ABSTRACT

Title of Dissertation: INNOVATIONS IN TIME SERIES FORECASTING: NEW VALIDATION PROCEDURES TO IMPROVE FORECASTING ACCURACY AND A NOVEL MACHINE LEARNING STRATEGY FOR MODEL SELECTION

Gustavo Varela Alvarenga
Doctor of Philosophy, 2021

Dissertation Directed by: Professor Benjamin Kedem
Department of Mathematics

This dissertation is divided into two parts. The first part introduces the *p-Holdout* family of validation schemes for minimizing the generalization error rate and improving forecasting accuracy. More specifically, if one wants to compare different forecasting methods, or models, based on their performance, one may choose to use "out-of-sample tests" based on formal hypothesis tests, or "out-of-sample tests" based on data-driven procedures that directly compare the models using an error measure (e.g., MSE, MASE). To distinguish between the two "out-of-sample tests" terminologies seen in the literature, we will use the term "out-of-sample tests" for the former and "out-of-sample validation" for the latter. Both methods rely on some form of data split. We call these data partition methods "validation schemes." We also provide a history of their use with time-series data, along with their formulas and the formulas for the associated out-of-sample generalization errors. We also attempt to organize the different terminologies used in the

statistics, econometrics, and machine learning literature into one set of terms. Moreover, we noticed that the schemes used in a time series context overlook one crucial characteristic of this type of data: its seasonality. We also observed that deseasonalizing is not often done in the machine learning literature. With this in mind, we introduce the *p-Holdout family* of validation schemes. It has three new procedures that we have developed specifically to consider a series' periodicity. Our results show that when applied to benchmark data and compared to state-of-the-art schemes, the new procedures are computationally inexpensive, improve the forecast accuracy, and greatly reduce, on average, the forecast error bias, especially when applied to non-stationary time series.

In the second part of this dissertation, we introduce a new machine learning strategy to select forecasting models. We call it the GEARS (generalized and rolling sample) strategy.

The "generalized" part of the name is because we use generalized linear models combined with partial likelihood inference to estimate the parameters. It has been shown that partial likelihood inference enables very flexible conditions that allow for correct time series analysis using GLMs. With this, it becomes easy for users to estimate multivariate (or univariate) time series models. All they have to do is provide the right-hand side variable, the variables that should enter the left-hand side of the model, and their lags. GLMs also allow for the inclusion of interactions and all sorts of non-linear links. This easy setup is an advantage over more complicated models like state-space and GARCH. And the fact that we can include covariates and interactions is an advantage over ARIMA, Theta-method, and other univariate methods.

The "rolling sample" part relates to estimating the parameters over a sample of

a fixed size that "moves forward" at different "rounds" of estimation (also known as "folds"). This part resembles the "rolling window" validation scheme, but ours does not start at $T = 1$. The "best" model is taken from the set with all possible combinations of covariates - and their respective lags - included in the right-hand side of the forecasting model. Its selection is based on the minimization of the average error measure over all folds. Once this is done, the best model's estimated coefficients are used to get the out-of-sample forecasts.

We applied the GEARS method to all the 100,000 time-series used in the 2018's M-Competition, the M4 Forecasting Competition. We produced one-step-ahead forecasts for each series and compared our results with the submitted approaches and the benchmark methods. The GEARS strategy yielded the best results - in terms of the smallest overall weighted average of the forecast errors - more often than any of the twenty-five top methods in that competition. We had the best results in 8,750 cases out of the 100,000, while the procedure that won the competition had better results in fewer than 7,300 series.

Moreover, the GEARS strategy shows promise when dealing with multivariate time series. Here, we estimated several forecasting models based on a complex formulation that includes covariates with variable and fixed lags, quadratic terms, and interaction terms. The accuracy of the forecasts obtained with GEARS was far superior than the one observed for the predictions from an ARIMA. This result and the fact that our strategy for dealing with multivariate series is far simpler than VAR, State Space, or Cointegration approaches shines a light in the future of our procedure.

An R package was written for the GEARS strategy. A prototype web application - using the R package "Shiny" - was also developed to disseminate this method.

INNOVATIONS IN TIME SERIES FORECASTING: NEW VALIDATION PROCEDURES TO IMPROVE FORECASTING ACCURACY AND A NOVEL MACHINE LEARNING STRATEGY FOR MODEL SELECTION

by

Gustavo Varela Alvarenga

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2021

Advisory Committee:
Professor Benjamin Kedem, Chair/Advisor
Professor Yan Li
Professor Vincent Lyzinski
Professor Jean Opsomer
Professor Paul J. Smith

Dedication

I dedicate this dissertation to my insanity avoidance companion, Victoria Echeverria. Thank you for being the Samwise to my Frodo. This accomplishment is as much yours as it is mine.

To Marlene Ignez Calhes *in memoriam*. One of the most complicated and amazing people I have ever met. You have always been - and always shall be - with me.

Acknowledgments

I want to thank my advisor, Professor Benjamin Kedem. I chose him to be my advisor because he had the most interesting and outside-of-the-box ideas that I have ever seen. During the years we worked together, he taught me how to perceive and react to the world in a way that freed my mind from the strings that restrained it. I would not have been able to develop new procedures if he had not stilled in me that ”ideas are limitless.”

I am also grateful for the extinct “Science without Borders” program (Programa Ciência sem Fronteiras) created during the administration of the former Brazilian president Dilma Rousseff and managed by the Brazilian Federal Coordination for the Improvement of Higher Education Personnel foundation (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior or, simply, CAPES foundation). As a member of this program (process number 99999.013767/2013-05), I received a Ph.D. scholarship that allowed me to focus on my studies during the first four years of my degree.

Table of Contents

Preface ii

Foreword ii

Dedication ii

Acknowledgements iii

Table of Contents iv

List of Tables viii

List of Figures xii

I New Validation Procedures to Improve Forecasting Accuracy 2

Chapter 1: Introduction 3

Chapter 2: Validation Schemes and Forecast Evaluation - A Brief History 16

Chapter 3: Validation Schemes and the Selection and Assessment of Forecasting Models 27

3.1 Using Validation Schemes to Evaluate Generalization Performance 27

3.2 A framework for Forward validation schemes 34

3.2.1 Holdout 34

3.2.2 Rep-Holdout 37

3.2.3 Rolling Origin and Prequential Growing Window 39

3.2.4 Prequential Sliding Window 44

3.2.5 Prequential in Blocks 45

3.2.6 Prequential Sliding Blocks 48

3.2.7 Prequential Blocks with Gaps 50

3.3 A framework for Cross-validation schemes 51

3.3.1 Leave-one-out and k-Fold Cross-Validation 51

3.3.2 k-Fold Blocked Cross-Validation 55

3.3.3 *h*-Block and *hv*-Block Cross-Validation 56

3.3.4 Modified cross-validation 65

Chapter 4: Introducing the *p-Holdout* family of schemes 72
4.1 The *p*-Holdout validation scheme . . . 75
4.2 The *cp*-Holdout scheme . . . 78
4.3 The *cep*-Holdout scheme . . . 79

Chapter 5: Data & Methodology 83
5.1 Data . . . 83
5.1.1 Data from Cerqueira et al. (2020) . . . 83
5.1.2 Data from the M4 Competition . . . 84
5.1.3 Monte Carlo Simulation . . . 86
5.2 Methodology . . . 86
5.2.1 Terminology . . . 87
5.2.2 The Forecasting Model and the Embedded Matrix . . . 87
5.2.3 Length of the Sets and Number of Folds . . . 89
5.2.4 Stationarity . . . 89
5.2.5 Learning Algorithms . . . 90
5.2.6 Forecast Accuracy Measures . . . 93
5.2.7 Hypothesis Testing and Comparisonwise Error Rate . . . 95
5.2.8 The Complete Experimental Design . . . 100

Chapter 6: Results 103
6.1 Data from Cerqueira et al. (2020) . . . 103
6.1.1 Results from the RBR learning algorithm . . . 103
6.1.2 Results from the RF learning algorithm . . . 107
6.1.3 Results from the GLM learning algorithm . . . 111
6.2 Data from the M4 Competition . . . 115
6.2.1 Results from the RBR learning algorithm . . . 115
6.2.2 Results from the RF learning algorithm . . . 118
6.2.3 Results from the GLM learning algorithm . . . 120
6.3 Monte Carlo Simulation . . . 123
6.3.1 Results from the RBR learning algorithm . . . 123
6.3.2 Results from the RF learning algorithm . . . 128
6.3.3 Results from the GLM learning algorithm . . . 133

Chapter 7: Final Remarks 138

II A Novel Machine Learning Strategy for Forecasting Model Selection 142

Chapter 8: Introduction 143

Chapter 9: The Learning Procedure in Machine Learning 148

Chapter 10: The new GEARS strategy 155
10.1 The Basic Idea . . . 155

10.2 A General Framework . . . 157
10.3 Hyperparameters' Optimization . . . 162

Chapter 11: Data and Methodology 165
11.1 The M4 Competition and its data sets . . . 165
11.2 The multivariate "commodities" data set . . . 167
11.3 Partial Likelihood Estimation . . . 168

Chapter 12: Results 170
12.1 One-Step Ahead - All 100,000 series from the M4 Competition . . . 170
12.2 One-Step Ahead - The multivariate "commodities prices" data set . . . 175

Chapter 13: Final Remarks 179

Appendix 183

Appendix A: Part I - Complementary Results using RMSE 183
A.1 Data from Cerqueira et al. (2020) . . . 183
A.1.1 Results from the RBR learning algorithm . . . 183
A.1.2 Results from the RF learning algorithm . . . 185
A.1.3 Results from the GLM learning algorithm . . . 187
A.2 Data from the M4 Competition . . . 189
A.2.1 Results from the RBR learning algorithm . . . 189
A.2.2 Results from the RF learning algorithm . . . 191
A.2.3 Results from the GLM learning algorithm . . . 193

Appendix B: Part I - Results for all cases using MASE 195
B.1 Data from Cerqueira et al. (2020) . . . 195
B.1.1 Results from the RBR learning algorithm . . . 195
B.1.2 Results from the RF learning algorithm . . . 201
B.1.3 Results from the GLM learning algorithm . . . 207
B.2 Data from the M4 Competition . . . 213
B.2.1 Results from the RBR learning algorithm . . . 213
B.2.2 Results from the RF learning algorithm . . . 219
B.2.3 Results from the GLM learning algorithm . . . 225
B.3 Monte Carlo Simulation . . . 231
B.3.1 Results from the RBR learning algorithm . . . 231
B.3.2 Results from the RF learning algorithm . . . 235
B.3.3 Results from the GLM learning algorithm . . . 239

Appendix C: R Code 243
C.1 Motivation . . . 243
C.2 The p-Holdout Family . . . 249
C.3 Forecasts - Cerqueira et al. (2020) . . . 252
C.4 M4 Competition Sample Selection and Forecasts . . . 256

C.5 Monte Carlo Simulations and Forecasts 262
C.6 Create Plots - Cerqueira et al. and M4 Competition 269
C.7 Create Plots - Simulated Data 291
C.8 Create Tables - Cerqueira and M4 302
C.9 Create Bayesian Plots 328
C.10 Forecasts for the M4 Competition series using the GEARS strategy 346

Appendix D: Optimized results for a subset of Daily time series from the M4 Competition 370

Appendix E: List of the top 20 papers found under the keywords 'machine learning time series' in number of citations, and whether they mention seasonality in their text. 372

Bibliography **374**

List of Tables

5.1 Period and estimated periods for the univariate time series in Cerqueira et al.(2020)'s data set. . . . 85
5.2 List of the selected validation schemes . . . 87

11.1 Number of M4 series, minimum and maximum sample sizes, and forecast horizon per data periodicity. . . . 166

12.1 Number of times a method yielded the smallest OWA, per series periodicity type. . . . 174
12.2 Observed future values and respective forecasts with a 95% prediction interval, and forecast errors for the price of pork meat using the GEARS strategy and ARIMA. . . . 177

A.1 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to all 174 real-world time series using the RBR learning algorithm and the RMSE as error measure. . . . 183
A.2 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 97 stationary time series using the RBR learning algorithm and the RMSE as error measure. . . . 184
A.3 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 77 non-stationary time series using the RBR learning algorithm and the RMSE as error measure. . . . 184
A.4 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to all 174 real-world time series using the RF learning algorithm and the RMSE as error measure. . . . 185
A.5 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 97 stationary time series using the RF learning algorithm and the RMSE as error measure. . . . 185
A.6 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 77 non-stationary time series using the RF learning algorithm and the RMSE as error measure. . . . 186

A.7 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to all 174 real-world time series using the GLM-Ridge learning algorithm and the RMSE as error measure. 187
A.8 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 97 stationary time series using the GLM-Ridge learning algorithm and the RMSE as error measure. 188
A.9 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 77 non-stationary time series using the GLM-Ridge learning algorithm and the RMSE as error measure. 188
A.10 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the time series from the M4 Competition sample using the RBR learning algorithm and the RMSE as error measure. 189
A.11 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the stationary time series from the M4 Competition sample using the RBR learning algorithm and the RMSE as error measure. 190
A.12 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the non-stationary time series from the M4 Competition sample using the RBR learning algorithm and the RMSE as error measure. 190
A.13 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the time series from the M4 Competition sample using the RF learning algorithm and the RMSE as error measure. 191
A.14 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the stationary time series from the M4 Competition sample using the RF learning algorithm and the RMSE as error measure. 192
A.15 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the non-stationary time series from the M4 Competition sample using the RF learning algorithm and the RMSE as error measure. 192
A.16 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the time series from the M4 Competition sample using the GLM-RIDGE learning algorithm and the RMSE as error measure. 193
A.17 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the stationary time series from the M4 Competition sample using the GLM-RIDGE learning algorithm and the RMSE as error measure. 194

A.18 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the non-stationary time series from the M4 Competition sample using the GLM-RIDGE learning algorithm and the RMSE as error measure. 194

B.1 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to all 174 real-world time series using the RBR learning algorithm and the MASE as error measure. 195
B.2 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 97 stationary time series using the RBR learning algorithm and the MASE as error measure. 198
B.3 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 77 non-stationary time series using the RBR learning algorithm and the MASE as error measure. 198
B.4 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to all 174 real-world time series using the RF learning algorithm and the MASE as error measure. 203
B.5 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 97 stationary time series using the RF learning algorithm and the MASE as error measure. 203
B.6 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 77 non-stationary time series using the RF learning algorithm and the MASE as error measure. 204
B.7 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to all 174 real-world time series using the GLM-Ridge learning algorithm and the MASE as error measure. 209
B.8 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 97 stationary time series using the GLM-Ridge learning algorithm and the MASE as error measure. 209
B.9 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 77 non-stationary time series using the GLM-Ridge learning algorithm and the MASE as error measure. 210
B.10 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the time series from the M4 Competition sample using the RBR learning algorithm and the MASE as error measure. 215

B.11 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the stationary time series from the M4 Competition sample using the RBR learning algorithm and the MASE as error measure. 215
B.12 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the non-stationary time series from the M4 Competition sample using the RBR learning algorithm and the MASE as error measure. 216
B.13 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the time series from the M4 Competition sample using the RF learning algorithm and the MASE as error measure. 221
B.14 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the stationary time series from the M4 Competition sample using the RF learning algorithm and the MASE as error measure. 221
B.15 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the non-stationary time series from the M4 Competition sample using the RF learning algorithm and the MASE as error measure. 222
B.16 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the time series from the M4 Competition sample using the GLM-RIDGE learning algorithm and the MASE as error measure. 227
B.17 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the stationary time series from the M4 Competition sample using the GLM-RIDGE learning algorithm and the MASE as error measure. 227
B.18 P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the non-stationary time series from the M4 Competition sample using the GLM-RIDGE learning algorithm and the MASE as error measure. 228

D.1 Optimized results for a subset of Daily time series from the M4 Competition 370

List of Figures

1.1 Example of splitting a time series with and without accounting for its seasonality. . . . 9
1.2 The *lynx* time series divided according to different validation schemes. . . 11

2.1 Possible ways to partition a single data set with observed values of y. . . . 19

3.1 Partitioning a time series using the Holdout forward validation scheme. . . 35
3.2 Partitioning a time series using the Rep-Holdout forward validation scheme. 38
3.3 Partitioning a time series using the Rolling-Origin forward validation scheme. 41
3.4 Partitioning a time series using the Prequential Growing-Window validation scheme. . . . 43
3.5 Partitioning a time series using the Prequential Sliding-Window validation scheme. . . . 44
3.6 Partitioning a time series using the Prequential-in-Blocks validation scheme. 46
3.7 Partitioning a time series using the Prequential Sliding Blocks validation scheme. . . . 48
3.8 Partitioning a time series using the Prequential-in-Blocks with Gaps validation scheme. . . . 50
3.9 Partitioning a time series using the Leave-One-Out validation scheme. . . 52
3.10 Partitioning a time series using the *k*-fold cross-validation scheme. . . . 54
3.11 Partitioning a time series using the *k*-fold blocked cross-validation scheme used by Cerqueira et al. (2020). . . . 56
3.12 Partitioning a time series using the *h*-block cross-validation scheme. . . . 57
3.13 Partitioning a time series using the *hv*-block cross-validation scheme used by Rancine (2000). . . . 61
3.14 Partitioning a time series using the *hv*-block cross-validation scheme used by Cerqueira et al. (2020). . . . 65
3.15 Partitioning a time series using the modified cross-validation scheme used by Cerqueira et al. (2020). . . . 70

4.1 The Holdout procedure and the validation schemes from the new *p-Holdout* family applied to the USAccDeaths data set. . . . 74
4.2 Partitioning a time series using the *p*-Holdout forward validation scheme. 77
4.3 Partitioning a time series using the *cp*-Holdout forward validation scheme. 79

5.1 Example of the mechanics of the M5 model tree algorithm 91

6.1 Average APAE rank of each validation scheme. Data: all real-life time series from Cerqueira et al. (2020). Estimation method: RBR. Error measure: RMSE . . . 104
6.2 Log percentage difference of the estimated loss relative to the true loss for each validation scheme. Data: all real-life time series from Cerqueira et al. (2020). Estimation method: RBR. Error measure: RMSE . . . 105
6.3 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: stationary real-life time series from Cerqueira et al. (2020). Estimation method: RBR. Error measure: RMSE . . . 106
6.4 Proportion of probability of winning outcome when comparing the performance estimation ability of the cep-Holdout and the other validation schemes. Data: non-stationary real-life time series from Cerqueira et al. (2020). Estimation method: RBR. Error measure: RMSE . . . 107
6.5 Average APAE rank of each validation scheme. Data: all real-life time series from Cerqueira et al. (2020). Estimation method: RF. Error measure: RMSE . . . 108
6.6 Log percentage difference of the estimated loss relative to the true loss for each validation scheme. Data: all real-life time series from Cerqueira et al. (2020). Estimation method: RF. Error measure: RMSE . . . 109
6.7 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: stationary real-life time series from Cerqueira et al. (2020). Estimation method: RF. Error measure: RMSE . . . 110
6.8 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: non-stationary real-life time series from Cerqueira et al. (2020). Estimation method: RF. Error measure: RMSE . . . 111
6.9 Average APAE rank of each validation scheme. Data: all real-life time series from Cerqueira et al. (2020). Estimation method: RIDGE. Error measure: RMSE . . . 112
6.10 Log percentage difference of the estimated loss relative to the true loss for each validation scheme. Data: all real-life time series from Cerqueira et al. (2020). Estimation method: GLM-Ridge. Error measure: RMSE . . . 113
6.11 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: stationary real-life time series from Cerqueira et al. (2020). Estimation method: GLM-Ridge. Error measure: RMSE . . . 114
6.12 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: non-stationary real-life time series from Cerqueira et al. (2020). Estimation method: GLM-Ridge. Error measure: RMSE . . . 115
6.13 Average APAE rank of each validation scheme. Data: sample of 1,000 time series from the M4 competition. Estimation method: RBR. Error measure: RMSE . . . 116
6.14 Log percentage difference of the estimated loss relative to the true loss for each validation scheme. Data: sample of 1,000 time series from the M4 competition. Estimation method: RBR. Error measure: RMSE . . . 117

6.15 Average APAE rank of each validation scheme. Data: sample of 1,000 time series from the M4 competition. Estimation method: RF. Error measure: RMSE 118
6.16 Log percentage difference of the estimated loss relative to the true loss for each validation scheme. Data: sample of 1,000 time series from the M4 competition. Estimation method: RF. Error measure: RMSE 119
6.17 Average APAE rank of each validation scheme. Data: sample of 1,000 time series from the M4 competition. Estimation method: GLM-Ridge. Error measure: RMSE 121
6.18 Log percentage difference of the estimated loss relative to the true loss for each validation scheme. Data: sample of 1,000 time series from the M4 competition. Estimation method: GLM-Ridge. Error measure: RMSE . . 122
6.19 Average APAE rank of each validation scheme. Data: sample of 1,000 simulated time series. Estimation method: RBR. Error measure: RMSE . 124
6.20 Log percentage difference of the estimated loss relative to the true loss for each validation scheme. Data: sample of 1,000 simulated time series. Estimation method: RBR. Error measure: RMSE 125
6.21 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: sample of 1,000 time series simulated from SARIMA$(12,0,0)\times(1,0,0)_{12}$. Estimation method: RBR. Error measure: RMSE 126
6.22 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: sample of 1,000 time series simulated from SARIMA$(12,0,0)\times(1,1,0)_{12}$. Estimation method: RBR. Error measure: RMSE 127
6.23 Average APAE rank of each validation scheme. Data: sample of 1,000 simulated time series. Estimation method: RF. Error measure: RMSE . . 129
6.24 Log percentage difference of the estimated loss relative to the true loss for each validation scheme. Data: sample of 1,000 simulated time series. Estimation method: RF. Error measure: RMSE 130
6.25 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: sample of 1,000 time series simulated from SARIMA$(12,0,0)\times(1,0,0)_{12}$. Estimation method: RF. Error measure: RMSE 131
6.26 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: sample of 1,000 time series simulated from SARIMA$(12,0,0)\times(1,1,0)_{12}$. Estimation method: RF. Error measure: RMSE 132
6.27 Average APAE rank of each validation scheme. Data: sample of 1,000 simulated time series. Estimation method: GLM-Ridge. Error measure: RMSE 134
6.28 Log percentage difference of the estimated loss relative to the true loss for each validation scheme. Data: sample of 1,000 simulated time series. Estimation method: GLM-Ridge. Error measure: RMSE 135
6.29 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: sample of 1,000 time series simulated from SARIMA$(12,0,0)\times(1,0,0)_{12}$. Estimation method: GLM-Ridge. Error measure: RMSE . . . 136

6.30 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: sample of 1,000 time series simulated from SARIMA$(12,0,0)\times(1,1,0)_{12}$. Estimation method: GLM-Ridge. Error measure: RMSE . . . 137

9.1 Test and training error as a function of model complexity. 151

10.1 Example of $M = 5$ rolling samples of size $S = 6$, for $T = 15$ and $H = 1$. 159
10.2 Flowchart of the optimization algorithm for the GEARS strategy. 164

12.1 One-step ahead forecast accuracy of the top 25 methods from the M4 Competition and the GEARS strategy 173

B.1 Average APAE rank of each validation scheme. Data: all real-life time series from Cerqueira et al. (2020). Estimation method: RBR. Error measure: MASE 196
B.2 Log percentage difference of the estimated loss relative to the true loss for each validation scheme. Data: all real-life time series from Cerqueira et al. (2020). Estimation method: RBR. Error measure: MASE 197
B.3 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: stationary real-life time series from Cerqueira et al. (2020). Estimation method: RBR. Error measure: MASE 199
B.4 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: non-stationary real-life time series from Cerqueira et al. (2020). Estimation method: RBR. Error measure: MASE 200
B.5 Average APAE rank of each validation scheme. Data: all real-life time series from Cerqueira et al. (2020). Estimation method: RF. Error measure: MASE 201
B.6 Log percentage difference of the estimated loss relative to the true loss for each validation scheme. Data: all real-life time series from Cerqueira et al. (2020). Estimation method: RF. Error measure: MASE 202
B.7 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: stationary real-life time series from Cerqueira et al. (2020). Estimation method: RF. Error measure: MASE 205
B.8 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: non-stationary real-life time series from Cerqueira et al. (2020). Estimation method: RF. Error measure: MASE 206
B.9 Average APAE rank of each validation scheme. Data: all real-life time series from Cerqueira et al. (2020). Estimation method: GLM-Ridge. Error measure: MASE 207
B.10 Log percentage difference of the estimated loss relative to the true loss for each validation scheme. Data: all real-life time series from Cerqueira et al. (2020). Estimation method: GLM-Ridge. Error measure: MASE . . . 208
B.11 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: stationary real-life time series from Cerqueira et al. (2020). Estimation method: GLM-Ridge. Error measure: MASE 211

B.12 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: non-stationary real-life time series from Cerqueira et al. (2020). Estimation method: GLM-Ridge. Error measure: MASE . . . 212
B.13 Average APAE rank of each validation scheme. Data: sample of 1,000 time series from the M4 competition. Estimation method: RBR. Error measure: MASE . . . 213
B.14 Log percentage difference of the estimated loss relative to the true loss for each validation scheme. Data: sample of 1,000 time series from the M4 competition. Estimation method: RBR. Error measure: MASE . . . 214
B.15 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: stationary series from the sample of 1,000 time series from the M4 competition. Estimation method: RBR. Error measure: MASE 217
B.16 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: non-stationary series from the sample of 1,000 time series from the M4 competition. Estimation method: RBR. Error measure: MASE . . . 218
B.17 Average APAE rank of each validation scheme. Data: sample of 1,000 time series from the M4 competition. Estimation method: RF. Error measure: MASE . . . 219
B.18 Log percentage difference of the estimated loss relative to the true loss for each validation scheme. Data: sample of 1,000 time series from the M4 competition. Estimation method: RF. Error measure: MASE . . . 220
B.19 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: stationary series from the sample of 1,000 time series from the M4 competition. Estimation method: RF. Error measure: MASE 223
B.20 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: non-stationary series from the sample of 1,000 time series from the M4 competition. Estimation method: RF. Error measure: MASE . . . 224
B.21 Average APAE rank of each validation scheme. Data: sample of 1,000 time series from the M4 competition. Estimation method: GLM-Ridge. Error measure: MASE . . . 225
B.22 Log percentage difference of the estimated loss relative to the true loss for each validation scheme. Data: sample of 1,000 time series from the M4 competition. Estimation method: GLM-Ridge. Error measure: MASE . . . 226
B.23 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: stationary series from the sample of 1,000 time series from the M4 competition. Estimation method: GLM-Ridge. Error measure: MASE . . . 229
B.24 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: non-stationary series from the sample of 1,000 time series from the M4 competition. Estimation method: GLM-Ridge. Error measure: MASE . . . 230
B.25 Average APAE rank of each validation scheme. Data: sample of 1,000 simulated time series. Estimation method: RBR. Error measure: MASE . 231

B.26 Log percentage difference of the estimated loss relative to the true loss for each validation scheme. Data: sample of 1,000 simulated time series. Estimation method: RBR. Error measure: MASE 232
B.27 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: sample of 1,000 time series simulated from SARIMA$(12,0,0)\times(1,0,0)_{12}$. Estimation method: RBR. Error measure: MASE 233
B.28 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: sample of 1,000 time series simulated from SARIMA$(12,0,0)\times(1,1,0)_{12}$. Estimation method: RBR. Error measure: MASE 234
B.29 Average APAE rank of each validation scheme. Data: sample of 1,000 simulated time series. Estimation method: RF. Error measure: MASE . . 235
B.30 Log percentage difference of the estimated loss relative to the true loss for each validation scheme. Data: sample of 1,000 simulated time series. Estimation method: RF. Error measure: MASE 236
B.31 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: sample of 1,000 time series simulated from SARIMA$(12,0,0)\times(1,0,0)_{12}$. Estimation method: RF. Error measure: MASE 237
B.32 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: sample of 1,000 time series simulated from SARIMA$(12,0,0)\times(1,1,0)_{12}$. Estimation method: RF. Error measure: MASE 238
B.33 Average APAE rank of each validation scheme. Data: sample of 1,000 simulated time series. Estimation method: GLM-Ridge. Error measure: MASE 239
B.34 Log percentage difference of the estimated loss relative to the true loss for each validation scheme. Data: sample of 1,000 simulated time series. Estimation method: GLM-Ridge. Error measure: MASE 240
B.35 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: sample of 1,000 time series simulated from SARIMA$(12,0,0)\times(1,0,0)_{12}$. Estimation method: GLM-Ridge. Error measure: MASE . . . 241
B.36 Probability of winning outcome of the cep-Holdout vs. the other validation schemes. Data: sample of 1,000 time series simulated from SARIMA$(12,0,0)\times(1,1,0)_{12}$. Estimation method: GLM-Ridge. Error measure: MASE . . . 242

E.1 List of the top 20 papers found under the keywords 'machine learning time series' in number of citations, and whether they mention seasonality in their text. 373

To all those who suffer from impostor syndrome:
per aspera ad astra.

Part I

New Validation Procedures to Improve Forecasting Accuracy

Chapter 1: Introduction

Assessment of forecast accuracy is an essential step in evaluating the procedures used in a given scenario. Such procedures can be divided into two classes: forecasting **methods** and forecasting *models*[1]. We define a forecasting *model* as the equation that represents the relationship (linear, non-linear, or non-parametric) between the variables in the analysis. In contrast, a forecasting **method** includes the *model*, the estimation procedure (i.e., the learning algorithm) used to fit the *model*, and the data used for that estimation.

To compare different forecasting methods, or models, based on their forecasting ability, one may choose to use "out-of-sample tests" that rely on partitioning the time series into two (or more) non-overlapping subsamples. The different procedures used to split a data set belong to a class called *validation schemes*. Splitting the data is necessary because using the same sample to estimate and check a model's prediction ability might lead to over-fitting it without necessarily providing good forecasts. These data-partitioning procedures also aim at minimizing the combined bias and variance. They do it by adding some bias in return for a reduction in the sampling variance. This is known as the *bias-variance trade-off*.

[1] Here, we feel that it is helpful to formalize the distinction between forecasting **methods** and *models*, similarly to what was done by Giacomini and White [1].

The mechanics of using validation schemes to evaluate forecast accuracy are quite simple. If one already has a forecasting model and wants to compare the performance of different estimation methods (i.e., learning algorithms), the time series is partitioned into a *training set* and a *test set*. The test set is wholly removed from the estimation phase and used only to compute the "true" out-of-sample forecast errors. The remaining observations become the *training set*, which is used to fit the model according to the selected estimation method and obtain its forecasts.

The description above is usually the approach taken by **formal** "out-of-sample tests" to evaluate different forecasting **methods**. These are procedures based on formal hypothesis tests created to evaluate the conditional or unconditional predictive ability of each forecasting method (the former asks "[C]an we predict which forecast will be more accurate at a future date?"; the latter focuses on finding out "[W]hich forecast was more accurate on average?' - both quotes are from Giacomini and White [1, p. 1545]), like the DM tests [2], or the tests by Giacomini and White [1] or Clark and McCracken [3] [2].

If one wants to compare different forecasting *model* formulations or optimize a methods' hyperparameters, the usual approach is to evaluate the predictions using *data-driven* "out-of-sample tests." In this approach, after splitting the original time series into the training and test data sets, the training set is partitioned into an *estimation set* and a *validation set*. The validation set is completely removed from the estimation phase and used only to compute the "pseudo" out-of-sample forecast errors. Different model formulations (and with distinct hyperparameters, if that is the case) are fit to the data in

[2] Diebold and Mariano [2], Diebold [4], and Clark and McCracken [3] prefer the term "pseudo-out-of-sample tests."

the estimation set, and the validation set is used to calculate the accuracy of their forecasts. The model formulation (or set of hyperparameters) that yields the smallest error measure in the validation set is crowned the winner[3]. The best model is applied to the entire training set, and the test set is used the obtain the "true" out-of-sample forecast errors.

To distinguish between the two "out-of-sample tests" terminologies seen in the literature, we will use the term "out-of-sample tests" to refer to the class of procedures based on formal hypothesis tests. Oppositely, when forecast models are selected based on data-driven approaches, we will use the term "out-of-sample validation"[4]. We want to stress that the distinction we make here is merely to avoid terminological confusion, since formal tests and partitioning the time series only into training and test sets can be used to select forecast *models* [1], just as data-driven approaches that split the series into validation and estimation sets can be used to evaluate different forecasting **methods** [7].

As to which approach is the better one, it is hard to say. For instance, Clark and McCracken [3] argue that many of the tests of equal predictive ability (like the ones by Giacomini and White [1] and West [8]) ignore "the real-time nature of the data used in many applications" [3, p. 15]. Moreover, while Clark and McCracken [9] propose a test where that nature is taken into account, many results hold only in special cases or are not robust to the presence of some noise [3, p. 15-16]. Furthermore, Inoue and Kilian [10] compare in-sample and out-of-sample tests under the null of no predictability and conclude that the former produces more credible results (and with higher power) than the

[3]Such assessment would be more in line with the unconditional approach of the formal tests.

[4]This has a different meaning to the term "out-of-sample evaluation" by Tashman [5, p. 437], and the expression "last block validation" from Bergmeir and Benítez [6, p. 193]. These authors used those terms in reference only to a particular group of procedures that we define in Chapter 3 as "forward-validation schemes" (see also Section 3.2).

latter (provided that the proper critical values are used), especially when data mining[5] is used. Additionally, Diebold and Mariano [2, p. 253] stated that the formal comparison of forecast accuracy - and, consequently, the development of formal tests - is difficult due to the dependent nature of the forecast errors over time.

On the other hand, "out-of-sample validation" approaches mainly deal with the problem of obtaining the best prediction and often do a good job of capturing complicated relationships. Because of this, these methods have been outperforming classical approaches. For instance, instead of worrying about the data-generating process, the methods that won the last two M-competitions are machine-learning-based methods that use validation schemes to select the best forecast model [11, 12].

From our literature review (Chapter 2), what determines which method one should use is often the area in which one works. On the one side, econometricians seem to prefer to use formal "out-of-sample tests" (based on the econometric, or economic-related, journals in which they published their work). On the other, machine learning practitioners prefer to use "out-of-sample validation" schemes. And it is not like there is a debate between the two areas to figure out what approach works best. Sometimes, it actually seems like there is a giant gap between the two. For instance, in 2020, the first version of a pre-print published by the journal "Data Mining and Knowledge Discovery" started with the sentence "This paper introduces Time Series Regression (TSR): a little-studied task of which the aim is to learn the relationship between a time series and a continuous target variable" (Tan et al. [13] - the latest version has been corrected after these words

[5]That is, searching among different forecast model specifications and reporting only those with the highest predictive ability.

received backlash on social media).

In the previous example, it seems that terminology played a vital role in the confusion[6]. However, in a larger sense, statisticians might be at fault for such a gap. In the paper "To explain or to predict?", Shmueli [15] writes:

> "Recognizing that statistical methodology has focused mainly on inference indicates an important gap to be filled. (...) Currently, the predictive void has been taken up the field of machine learning and data mining. In fact, the differences, and some would say rivalry, between the fields of statistics and data mining can be attributed to their different goals of explaining versus predicting even more than to factors such as data size. While statistical theory has focused on model estimation, inference, and fit, machine learning and data mining have concentrated on developing computationally efficient predictive algorithms and tackling the bias–variance trade-off in order to achieve high predictive accuracy." (Shmueli, 2010, p. 306)

And in the abstract of the paper "Statistical modeling: The two cultures," Breiman [16] argues that:

> "There are two cultures in the use of statistical modeling to reach conclusions from data. One assumes that the data are generated by a given stochastic data model. The other uses algorithmic models and treats the data mechanism as unknown. The statistical community has been committed to the almost exclusive use of data models. This commitment has led to irrelevant theory, questionable conclusions, and has kept statisticians from working on a large range of interesting current problems. Algorithmic modeling, both in theory and practice, has developed rapidly in fields outside statistics. It can be used both on large complex data sets and as a more accurate and informative alternative to data modeling on smaller data sets. If our goal as a field is to use data to solve problems, then we need to move away from exclusive dependence on data models and adopt a more diverse set of tools." (Breiman, 2001, p. 199)

For those reasons, we will focus on "out-of-sample validation" schemes and will provide a discussion that is more practical and less focused on theoretical proofs. Other factors impacted this decision, and in Chapter 2, we discuss the hardships that dependent data brings to the development of theoretical results in this literature.

[6] See Hyndman [14].

We also propose new procedures to fill in a gap observed during our literature review. In our survey, we noticed that the validation schemes developed for time series often overlook one crucial characteristic of this type of data: its seasonality. Since a time series might contain cycles and seasonal patterns, disregarding them when dividing the series between training (estimation and validation) and test data might lead to an incorrect choice of the best forecasting model. This is the same argument that Bergmeir and Benítez [6] make when discussing the importance of taking non-stationarity into account. They write[7],

> "Non-stationarity has to be taken into account throughout the whole modeling process, not only during model selection. Depending on the type of stationarity, it can be easily removed by a preprocessing step (...). If non-stationarity cannot be removed by such a preprocessing step, the model building procedure may require a processing step that determines, which parts of the series to include in the modeling, as proposed by Deco et al. [22], or prediction of the series might even be an impossible task [22,33]. Furthermore, for non-stationary series last block evaluation might be misleading (...), as the block chosen for testing might be very different from the training data, and the unknown future may also be different from the training data, the test data, or from both of these." (Bergmeir and Benitez, 2012, p. 198)

We could easily replace "non-stationarity" with "seasonality" on this quote. Firstly, we say that seasonality (when it occurs) must be taken into account. However, in our review, we observed that more than half of the top 20 most cited papers under the keywords "machine learning time series" completely fail to mention anything about "season" or "cycles." Furthermore, a search on Google Scholar for the terms "machine learning time series deseasonalized" or "machine learning time series deseasonalize" yielded fewer than 1,300 hits (1,240 in the former and 1,230 in the latter). It seems, then, that, in general,

[7] In the quote, Deco et al. [22] refers to the paper: G. Deco, R. Neuneier, B. Schürmann, Non-parametric data selection for neural learning in non-stationary time series, Neural Networks 10 (3) (1997) 401–407. The citation [33] refers to: T.Y. Kim, K.J. Oh, C. Kim, J.D. Do, Artificial neural networks for non-stationary time series, Neurocomputing 61 (1–4) (2004) 439–447.

those in Academia who deal with time series methods and machine learning do not consider seasonal patterns. In the industry, the behavior seems the same. Only relatively recently have we seen a movement towards dealing with seasonal data. For instance, Google's TensorFlow - a software library for machine learning - was released in 2015, and in 2016 Google announced its capabilities to deal with time-series data [17]. However, only in 2019, they released a library for forecasting time series that accounts for a series' seasonality [18].

Secondly, for seasonal time series, last block evaluation might be misleading, as the block chosen for testing might be very different from the training data. However, we argue that *because* a seasonal series display a similar behavior over time, it is possible to obtain similar validation and test sets in a way that improves model selection and forecast accuracy, as shown in Figure 1.1.

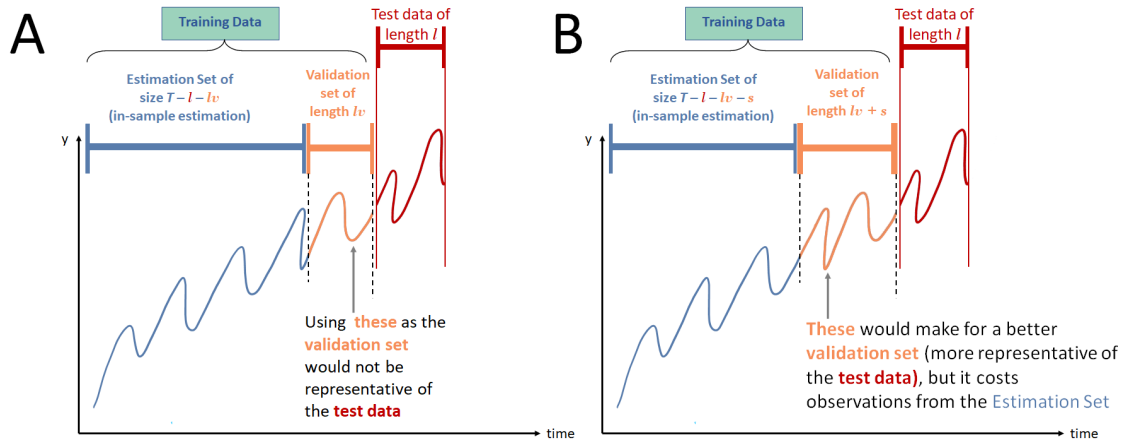


Figure 1.1: Example of splitting a time series without account for its seasonality (Panel A) and accounting for it (Panel B).

Based on this, we developed the *p-Holdout* family of validation schemes. The "p" in

p-Holdout stands for "period," an inspiration taken from Box et al. [19][8]. And "Holdout" was chosen because our schemes are simple modifications of the last block procedure, which is more commonly called the *holdout scheme* since the last part of the data is "held out" from the training set.

The *p-Holdout* family has three new validation schemes that take into account a series' periodicity. The first one simply called the `p-Holdout` scheme, is a simple modification of the `Holdout` scheme that incorporates the period in an additive way, while the `cp-Holdout` does that in a multiplicative manner. In both cases, the period is obtained using the `frequency` function from base `R`[9]. The `cep-Holdout` works similarly as the `cp-Holdout` but uses the dominant frequency of a time series estimated from a spectral analysis of the data[10]. A real-life data example of partitioning a data set under these procedures is given in Figure 1.2. In Panel A, the famous time series from Brockwell and Davis [20, p. 557] of the annual numbers of lynx trappings for 1821–1934 in Canada is split into training (70% of the data) and test sets (30%). We would like to obtain a validation set that is as similar as possible to the test set. Since the frequency is equal to one (i.e., `frequency(lynx)` returns 1), we see in Panels B, C, and D that the division according to the `Holdout` , `p-Holdout` , and `cp-Holdout` procedures yields similar validation sets, and none of them capture the spike seen in the test set. On the other hand, the `cep-Holdout` scheme can capture it, yielding a validation set that resembles the test set.

[8]Quoting Box et al. [19, p. 306] in their chapter *Analysis of Seasonal Time Series*: "In general, we say that a series exhibits periodic behavior with period s when similarities in the series occur after s basic time intervals."

[9]Here, the "frequency" is the number of sample observations before the seasonal pattern repeats.

[10]Given by the `findfrequency` function from the `forecast` package.

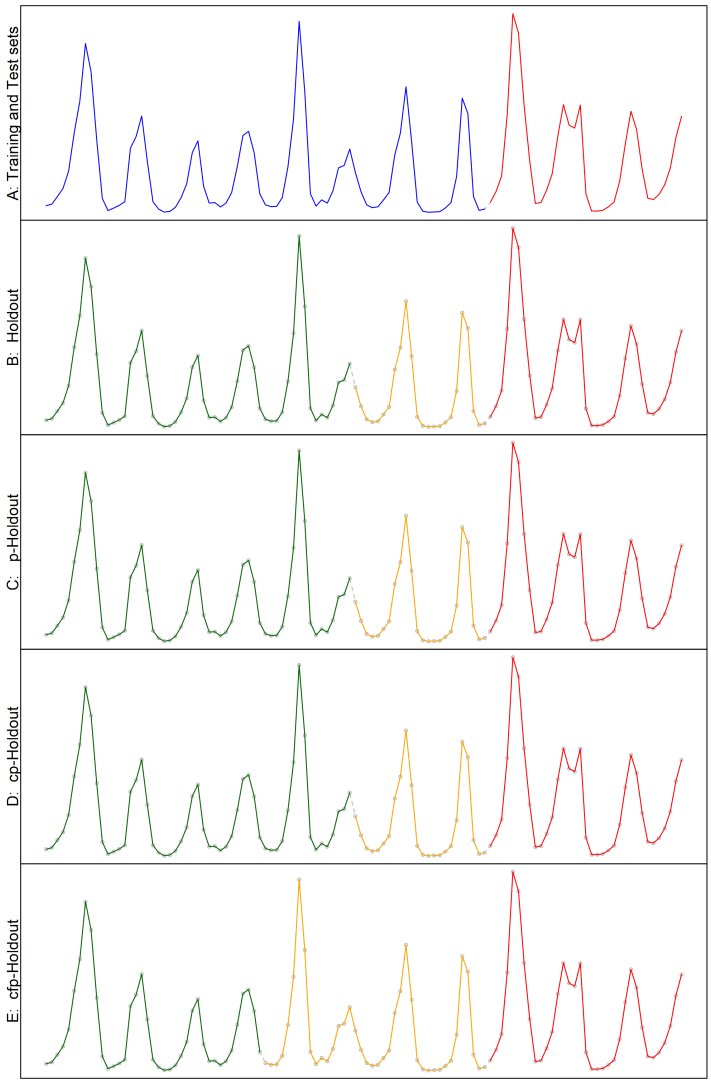


Figure 1.2: Time series with the annual number of lynx trapping for 1821-1934 in Canada, partitioned according to different validation schemes. Panel A shows the data split into the training set (blue) and the test set (red) using the holdout scheme. In Panel B, the training data is further divided into estimation (green) and validation sets (yellow) according to the holdout procedure. Panels C, D, and E show the division of the training set using the p-holdout, cp-holdout, and cep-holdout, respectively. In Panels B-E, the maximum length of the estimation set is 70% that of the training set.

To evaluate our procedures, we used an experimental design that was first introduced by Bergmeir and Benítez [6] and augmented by Bergmeir et al. [21] to include seasonal cases. The benchmarking data sets created via Monte Carlo simulation were also used by Cerqueira et al. [22] to compare the performance of several validation schemes. These authors also used real-life data sets in their study. Since the work of Cerqueira and Torgo and Mozetič [22] is the most recent one (2020) and includes more schemes and time series, we used their methodology to assess our proposed scheme. By using the same data and the same methodology, we can directly compare our results to theirs. Henceforth, we will refer to the work of Cerqueira et al. [22] as CTM.

In the experimental design by CTM, the goal is to evaluate the impact in forecast ability of the different data-splitting methods (the validation schemes). This is done by calculating the difference in the out-of-sample error in the *test set* (also called the "ground truth loss" [22] or "true out-of-sample generalization error" [23]) and the out-of-sample error in the **validation set** (the "pseudo out-of-sample generalization error").

To do that, we take a time series and split it into the training and test sets using the classic `Holdout` scheme. A given forecasting model is fitted to the training set using a given estimation method, and the forecasts are compared to the out-of-sample observations in the test set yielding the generalization. Then, the training set is further split into the estimation and validation sets using each one of the fourteen validation schemes considered here. The same forecasting model is fit using the same estimation method, and its forecasts are used to calculate the generalization error based on the **validation set**. Once both generalization errors are obtained, the absolute predictive accuracy error (APAE - which is the absolute difference between the errors) is calculated, and the vali-

dation schemes are ranked based on it - the scheme that yielded the smallest APAE value is the "best" for that time series and receives rank $= 1$, while the scheme with the maximum APAE is ranked 14-th. After going over this process for all time series, we take the average of the ranks obtained from each one, and we evaluate which scheme, on average, yielded the smallest values for the APAE metric.

However, the "pseudo" out-of-sample errors can be very close to the "true" out-of-sample errors, yet their difference can be far from zero. Ideally, both should be close to each other **and** close to zero. The APAE metric evaluates the former, and CTM uses the the log scaled of the predictive accuracy error (PAE) metric to evaluate the latter. The PAE metric is essentially the same as the APAE metric, it returns the difference (not the absolute difference) between the estimation error (obtained by applying a loss function to the forecasts from the model fit to estimation set and the actual observations from the validation set) and the "true" error (applying the same loss function to the forecasts from the model fit to the training set and the actual values from the test set).

Using the data from CTM, we observed that accounting for the periodicity has a considerable impact on reducing the average forecast error bias (measured by the predictive accuracy error - PAE - metric, especially in non-stationary series, as shown in Chapter 6. Moreover, when we focused only on periodic series (with seasons or cycles greater than 1) taken from a subsample of 1,000 real-life time series randomly selected from the 100,000 time series in the M4 Forecasting Competition database [24], we noticed that the new schemes yielded better results, both in terms of being the schemes that yielded the smallest forecast errors (quantified by the absolute predictive accuracy error - APAE - metric) more often than all of the other methods, and in terms of providing a

smaller error bias (PAE). And again, the best results were observed when non-stationary series were considered.

To complement our analysis, we also provide a brief history of the use of validation schemes with time-series data (Chapter 2). Based on this, we tried to organize all the different terminologies used in the statistics, econometrics, and machine learning literature into one set of terms.

In Chapter 3, we provide a more comprehensive account of the details for each validation scheme. Following Schnaubelt [23], who wrote the basic formulas for some of the validation procedures and how to use them to measure forecast accuracy, we write a general theory on how to use the schemes to evaluate forecasting methods and models. We expand on Schnaubelt's work by detailing each validation scheme and providing the associated formulas for each procedure's out-of-sample generalization errors, as Arlot and Celisse [25] did for the independent case. We also added information on schemes not covered by Schnaubelt [23], and present schematic illustrations - made specifically for the time series case - of partitioning the series using each one of them. From our literature survey, it seems that this is the first time that an organized, detailed, and properly illustrated survey of (most of) the state-of-the-art validation schemes used in time series is given *within* a time series context.

We expand on the motivation behind the creation of the `p-Holdout` family of validation schemes in Chapter 4. Similar to what was done for the other validation procedures, we provide the formulas for the three new methods that we are proposing and explain how to use them to evaluate forecasting models.

The methodological approach we used to evaluate our new schemes, including the

estimation methods (learning algorithms) used to fit the models and the details about the Monte Carlo simulations we ran, can be found in Chapter 5. In that same chapter, we give a brief description of the real-life data sets taken from CTM and the M4 Forecasting Competition that we used to complement our analysis. The results are presented in Chapter 6, and Chapter 7 contains our final remarks and suggestions for future research.

Chapter 2: Validation Schemes and Forecast Evaluation - A Brief History

One approach to evaluate forecast accuracy and compare forecasting methods and models is to use data-driven procedures to split the original time series into two (or more) non-overlapping sub-samples and use one part to fit the model and the other to evaluate its predictive accuracy. The method/model that yields the smallest error measure is considered the "best" one, as it was validated during this entire process. Thus, we call this approach "out-of-sample validation,"[1] and the class of procedures used to split the data receives the name "validation schemes."

The way we call the sub-samples has changed a lot over time. Nowadays, the first sub-sample is usually called the *training data* (or the "in-sample data" - IS - as it is often called in the statistics/econometrics literature). It is used to fit a selected forecasting model and obtain the regression coefficients of each covariate (i.e., it is used to learn the weights of the features). We then use the estimated coefficients to obtain the out-of-sample forecasts and compare these to the observed values in the second sub-sample,

[1] As mentioned in the Introduction (Chapter 1), some authors call the methods of evaluating the forecast accuracy based on data splits as "out-of-sample tests." It seems that this terminology started with Meese and Rogoff [26]. Other names have been used (for instance, Diebold [4] calls them "pseudo-out-of-sample procedures," Stock and Watson [27] call them "simulated out of sample" methods, and Makridakis [28] calls it "sliding simulation," to name a few), but "out-of-sample tests" seems to be the most used term. However, from our review, "out-of-sample tests" seem to encompass only a subset of the procedures, the ones called "forward-validation" schemes. More on this later.

the *test data set* (or the "out-of-sample data" - OOS). At this step, the method's forecasting accuracy is calculated based on some error measure selected by the analyst (RMSE, MAD, MASE). Figure 2.1 - Panel A shows an example of a data set split into a training set and a test set using the `Holdout` validation scheme (more about it in Section 3.2.1). This is the simplest validation scheme, and it has this name because the test data is "held out" of the data used for fitting. The same approach can also be used for model selection and hyperparameters' optimization. In those cases, the training data is further divided into an "estimation set" and a non-overlapping "validation set"[2]. This procedure is shown in Figure 2.1 - Panel B.

Splitting the data through validation schemes is necessary because using the same sample to estimate and to check a model's prediction ability might lead to over-fitting the model without necessarily providing good forecasts, a phenomenon that time series researchers have observed, at least, since the 1930s [30, in the text of Armstrong, 31][3]. More recently, White [32] warned us about the dangers of "data snooping" - when the same data set is used for model selection and inference.

The history of partitioning a time series to evaluate its forecasts is long. In the first half of the 1930s, Wilson [33] used data splits in conjunction with periodograms to search for hidden periods in a time series. To do so, the author used a large series with 1680 months spanning the years of 1790 to 1929 and divided the data into three blocks (1790-1859, 1825-1894, and 1860-1929), and used each block to obtain the forecasts.

[2]When these methods started to become popular, these sub-samples were known as "construction sample" and "validation sample" (Stone [29]). Stone preferred the term "assessment" to "validation" because the latter "has a ring of excessive confidence about it" [29, p. 111]. We agree with this terminology, but we shall keep using "validation" since it is the current norm.

[3]See Armstrong [31, p. 338-339] for more examples of authors that wrote in the first half of the twentieth century about partitioning a time series to evaluate the quality of the forecasts.

Regarding the middle block (1825-1894), Wilson [33] wrote that:

> "It is further seen that as a backward forecaster the series computed from the periods and coefficients indicated by the periodogram of the middle half, is also worse than useless; *as a forward forecaster it is not bad, and indeed forecasts the course of the index during the decade 1900-1909 better than it fits any decade on which the calculations are based.*" (Wilson, 1934, p. 408, emphasis added)

At the beginning of the 1950s, Ferber [34, in the text of Ferber, 35], used data until 1940 to forecast the value of the annual savings during the 1947-1949 period. The author calculated the average absolute percentage error of the forecasts and compared it with the sample coefficient of determination (r^2). The goal was to evaluate if r^2 could be used as an indicator of predictive accuracy.

Kirby [36] used data splits to compare the accuracy of forecasting methods (exponential smoothing, moving averages, and least squares) using monthly data that spanned 90 months. The author used 23 real and 23 simulated time series with and without seasonal and trend adjustments to obtain short-term (next month) and intermediate-range (6 months) forecasts. The first 36 months were used to fit the models and "allow the exponential smoothing bases to settle down" [36, [p. B-203]. Starting at the 4th year, the author computed forecasts at each month and used the actual observations to calculate the mean absolute forecast error.

Later, Williams and Goodman [37] used a method that closely resembles one of the validation schemes that we use today (see Section 3.2.3). To obtain a confidence limit for economic forecasts, they fitted a model to the first 24 observations and used it to predict the 25th and obtain its forecast error. Then, a model was fit using the first 25 observations to obtain the forecasts error of the 26th. This procedure was repeated until 18 forecasts

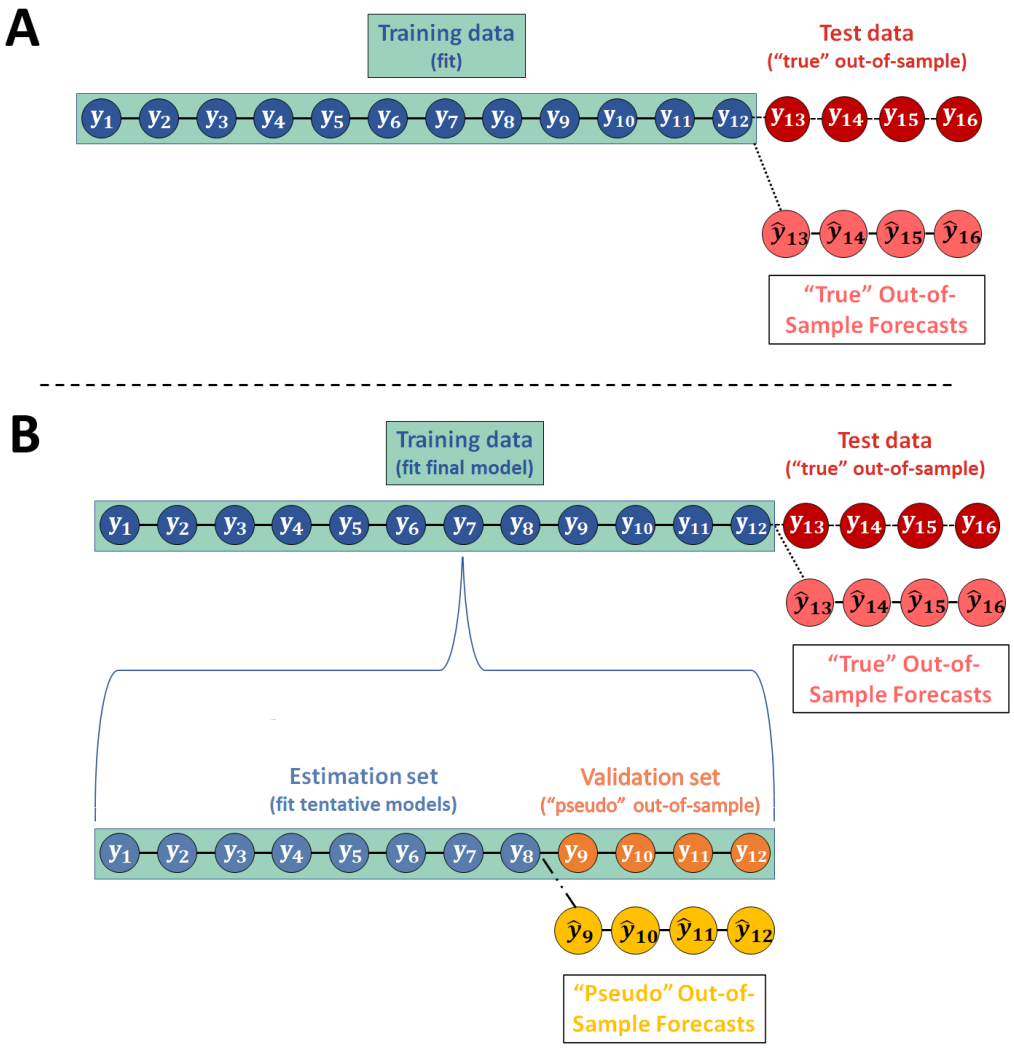


Figure 2.1: Possible ways to partition a single data set with observed values of y. Panel A displays the time series divided into training set and test set, and shows the relationship between the "true" out-of-sample data and the associated forecasts. Panel B shows the division of the training set into the estimation and validation sets.

were obtained, along with a confidence limit for each forecast.

In the following year, Nelson [38] divided his data into "sample" and "post-sample" sets and used them to respectively estimate and evaluate the prediction performance of the

FRB-MIT-PENN econometric model [39, 40] and of ARIMA models [19, 41] of the U.S. economy.[4] Also in 1972, Armstrong and Grohman [7] used a validation-like procedure to compare different methods[5] of forecasting the revenue of the U.S. air travel market.

Up until now, the papers that we have reviewed use procedures that keep only observations from the "future" in their test sets. Several authors kept using this "forward" approach [1, 2, 3, 8, 26, 27, 42, 43, 44]. But after the work of Stone [29], another "branch" of validation schemes started to be used within a time series context.

Stone [29] provided an extensive study on the use of the cross-validation (CV) approach that we now call the *leave-one-out* procedure (Section 3.3.1). It basically does what the name suggests. It removes one observation from the entire data set and makes that observation to be the entire test set. All the remaining observations lie in the estimation set. What set Stone's work apart is that this author was one of the first to use cross-validation schemes for model selection *and* prediction evaluation (with independent data). Stone [45] also showed that this scheme is asymptotically equivalent to the AIC (Akaike's Information Criterion; Akaike [46]). Since then, the use and development of CV techniques for dependent data have become more popular. Bergmeir and Benítez [6], Ansley and Kohn [47], Jong [48], Snijders [49], Burman [50], Zhang [51], Yao and Tong [52], Burman et al. [53], Kohavi [54], Racine [55], Kunst [56] - all these papers use the explicit terminology of "cross-validation" in a dependent data setting. We discuss some of these approaches in Chapter 3.3.

We see, then, that the basic idea of splitting the data to fit-and-predict to evaluate

[4] That is the idea behind the `Holdout` scheme. See Section 3.2.1.
[5] As we have defined on footnote 1. The original authors used "models."

a method's, or model's, forecasting ability is not new and has been evolving. Since the publication of the above papers, a lot has been done, and new authors have developed many schemes for dividing the samples. We classify the procedures into two classes: *cross-validation* and *forward validation* schemes.

Here, the term *cross-validation* (CV) refers to the group of schemes in which the time-series observations are randomly selected to be part of the training, estimation, and test sets. In this resampling procedure, observations from the past become part of the test set, so the order of the series is destroyed. We distinguish those from the *forward-validation* (FV) schemes, the group that encompasses the procedures where the data is divided in a way that only observations from the "future" are in the test set (i.e., its indexes are past the indexes of the observations in the training set). In such schemes, the order of the observations is kept intact within both training and test sets (one example is the `Holdout` scheme)[6]. Here, we consider that both groups comprise the larger class called *validation schemes*[7].

We indicate Rao and Wu [59] and Arlot and Celisse [25] for extensive surveys on cross-validation procedures. It is worth mentioning that since these schemes were not (at least, initially) developed within a time-series framework, the discussion by Arlot and Celisse [25] focuses on the cross-sectional uses of these methods, and the authors only briefly discuss the use of those procedures in a time-series context [25, see Section 8.1,

[6]Our definition of "forward validation" is different from the one by Hjorth [44]. Hjorth used this name to describe a procedure similar to the one discussed in Section 3.2.3 (in particular, the one discussed in Section 3.2.3), while here, we used it to define an entire class of schemes. Hjorth's procedure would, then, be a member of this class.

[7]There are other data validation procedures (for instance, those based on bootstrap methods) that could be included in this larger class, but they will not be covered here. We refer the reader to the work of Fukuchi [57], Kitamura [58], and Kunst [56].

p. 65-66]. Similarly, Rao and Wu [59] discuss cross-validation techniques in several contexts, but their approach to order selection in time series Rao and Wu [59, p. 31-36] does not include the schemes discussed here (they rather focus on AIC-like procedures).

Tashman [5] provided the first systematic review in a time-series context. However, only forward-validation schemes (dubbed as "out-of-sample tests" by the author) were discussed. Clark and McCracken [60]'s discussion is focused on *formal* "out-of-sample tests," but since these methods depend on some form of data split, the authors discuss a few of the forward validation schemes. More recently, Schnaubelt [23] wrote a more theoretically-focused review of validation schemes that includes both groups and used the schemes to compare the forecasting ability of machine learning models applied to non-stationary series.

As mentioned earlier, cross-validation schemes were not developed within a time-series context and often assume independent and identically distributed (i.i.d.) observations. The reason behind this assumption is that the reshuffling destroys the ordering of the observations. This is one of the main criticisms on using cross-validation with time-dependent data [6]. Moreover, in a data-dependent case, [61] discusses how cross-validation methods, when used to select a model (the smooth function) in a non-parametric case, produce an under smoothed estimate and leads to an overfitting of the data. Practical problems are also observed when dealing with missing observations [6]. These may be the reasons why statisticians do not often use these schemes in traditional forecasting [6, 62]. On the other hand, forward-validation procedures only use a part of the information available and might lose potentially important information (a problem that can also heavily affect cross-validation approaches, like the modified CV method

discussed in Section 3.3.4).

Given these characteristics, it is really difficult to prove some of the scheme's theoretical results when using dependent data[8], and it is fairly difficult to prove some of the results when applied to time series data because of the "ordered observations" component. Especially if one wants to avoid making restrictive assumptions, like assuming serially uncorrelated forecast errors [3, p. 10].[9]

Nevertheless, it is important to evaluate forecast accuracy using genuine forecasts alongside data that was not used to obtain said forecasts, and validation schemes provide a way to do that. Some advocate for the use of these schemes as is (under certain assumptions) [21], other procedures - or, modifications of the usual strategies - have been developed over the years (Jong [48], Snijders [49], Burman et al. [53], Racine [55], Kunst [56], Chu and Marron [67]) to properly account for the intrinsic dependency seen in time-series data. And while no scheme is perfect and free from issues, they seem to work in various real-life situations. Aside from the several papers already cited, recent papers have focused on the evaluation and comparison of the validation schemes.

Bergmeir and Benítez [6] have developed a rigorous and extensive experimental design to evaluate the consequences of using different validation schemes[10] on model selection and forecast accuracy. Their ultimate goal was to assess if using cross-validation

[8] Arlot and Celisse [25]'s extensive survey on cross-validation procedures presents the results (not their proofs) of theorems on the asymptotic properties of these schemes, as well as the closed forms of the expected values and the variance of the estimators of the risk (in the context of our paper, "risk" is the generalization error given in Eq.(3.4, Chapter 3). However, most of these results were proved under the assumptions of independent and identically distributed observations. For time-dependent data, only a few results for cross-validation methods were mentioned by the authors [25, p. 65-66], and these were specifically related to model selection procedures in a non-parametric setting where the errors are correlated.

[9] Yet that did not stop Giacomini and White [1], Diebold and Mariano [2], West [8], Clark and McCracken [9], Bergmeir et al. [21], Racine [55], Burman and Nolan [63], McCracken [64], Hirano and Wright [65], McDonald et al. [66], and many others from working on theoretical problems.

[10] Bergmeir and Benítez [6] used the term "model selection procedures" to refer to validation schemes.

methods - devised for independent data - on data that present dependencies would unduly affect the results. In other words, they aimed to assess if it is possible to obtain good predictions even after the time ordering of the observations was destroyed. The authors concluded that the theoretical problems that one might expect from using cross-validation methods with time-series data were not detected in the empirical results. Moreover, the use of the forward-validation schemes yielded worse results than using cross-validation.

Bergmeir et al. [68] extended [6] by evaluating to what extent cross-validation schemes are better than forward-validation procedures for directional forecast evaluation[11], using a Monte Carlo analysis. Aside from their results (they recommended using the blocked k-fold - see Section 3.3.2 - when dealing with directional forecast evaluation), the main output from Bergmeir et al. [68]'s paper are the data sets that they simulated. Those data sets were also used by Bergmeir et al. [21] and Cerqueira et al. [22], and their description can be found in Section 5.1.1.

Bergmeir et al. [21] wrote about the validity of using cross-validation for evaluating autoregressive time series prediction and mathematically proved that, under the assumption that the rows of the embedded matrix (the matrix with the past values of the covariates at different lag values and their respective response variables, see Section 5.2.2) are conditionally uncorrelated, the estimated prediction error for the k-fold cross-validation scheme (Section 3.3.1) converges to the real prediction error. They used this result and empirical results from Monte Carlo simulations to advocate for using this cross-validation procedure without any modifications. They also expanded on the experimental design from

[11] Directional accuracy measures if the the forecasts have the same direction as the actual "out-of-sample" values. In a very crude way, it measures if $\hat{y}_{t+1} - y_t > 0$ when $y_{t+1} - y_t > 0$.

Bergmeir and Benítez [6] and Bergmeir et al. [68] and added a simulated time series that contains seasonal patterns. We discuss the simulated series in Section 5.1.3 and use the same design to evaluate the *p-Holdout* family of schemes (Section 6.3).

Schnaubelt [23] compared the effectiveness of validation schemes applied specifically to non-stationary time series data. The author followed the experimental design from Bergmeir and Benítez [6] and a preprint of Cerqueira et al. [22] [69] and introduced time-dependent parameters as a perturbation of the stationarity of the process. The goal was to mimic the changing dynamics observed, for instance, in financial data. The author concluded that cross-validation schemes yielded estimates with the largest bias and variance vis-à-vis forward-validation schemes. Moreover, forward-validation procedures yielded better estimates of the out-of-sample error. In the end, Schnaubelt [23] stated that,

> "Using cross-validation for time-series applications comes at a great risk. While theoretically applicable, we find that cross-validation often is associated with the largest bias and variance when compared to all other validation schemes. In most cases, blocked variants of cross-validation have a similar or better performance, and should therefore be preferred if cross-validation is to be used. If global stationarity is perturbed by non-periodic changes in autoregression coefficients, we find that forward-validation may be preferred over cross-validation." (Schnaubelt, 2019, p.33)

Cerqueira et al. [22] reached similar conclusions after performing their experiment. These authors used the same design as Bergmeir et al. [68] and Bergmeir et al. [21] and added real-life data sets from [70] to their analysis. They also focused on time series with a high sampling frequency (like hourly and daily data) since this characteristic "is typically associated with more data, which is important for fitting the predictive models from a machine learning point of view."

All authors mentioned above conclude that stationarity is a crucial time series prop-

erty to taking into account when selecting the proper validation scheme. However, only Bergmeir et al. [21] evaluate the impact that the periodicity of a series has on this selection. They used a seasonal AR process as the data-generating process (DGP) with a significant lag 12 (seasonal lag 1) as a counterexample to show a counterexample where the cross-validation procedures break down. CTM also used this data set, but their paper has no mention of the impact that this characteristic has on the results.

Chapter 3: Validation Schemes and the Selection and Assessment of Forecasting Models

3.1 Using Validation Schemes to Evaluate Generalization Performance

Now that we have seen how these validation schemes evolved, it is time to understand their details and learn how we can use them to evaluate the generalization performance of a forecasting method or model.

But before getting into its details, let us start by discussing the form of the forecasting models. Let $\{Y_t\}_{t=1}^T$ be a time series of interest, and $\{y_t\}_{t=1}^T$ its observed values. We define the random covariate process $\mathbf{Z}_{t-1}$ as the p-dimensional vector of past explanatory variables:

$$\mathbf{Z}_{t-1} \equiv \left(Z_{(t-1)1}, \cdots, Z_{(t-1)p}\right)$$

The observed values of this process are $\{\mathbf{z}_{t-1}\}$, and in the machine learning literature, this is known as the p-dimensional vector of features used to predict the desired output y_t. The vector $\mathbf{Z}_{t-1}$ can also contain past values of the response variable Y_t and certain covariates X_t, W_t, known at $t-1$ (e.g., when these are deterministic or shifted processes). We denote by $\mathcal{F}_{t-1}$ the σ-algebra generated by all the covariates observed up until time

$t-1$:

$$\mathcal{F}_{t-1} \equiv (Y_{t-1}, Y_{t-2}, \ldots, X_t, W_t, \ldots, \mathbf{Z}_{t-1}, \mathbf{Z}_{t-2}, \cdots)$$

To forecast a time series y_t, one may use the available covariates to improve the results. In the statistics literature regarding generalized linear models [71], this can be done by defining the conditional expectation of the response given the past, μ_t, as:

$$\mu_t \equiv E\left[Y_t | \mathcal{F}_{t-1}\right]$$

At times, and when it is feasible, it is convenient to think of $\mathbf{Z}_{t-1}$ as already including past values of the response, and the known $X_t, W_t, \ldots$, so the time-dependent random covariate vector process $\{\mathbf{Z}_{t-1}\}$ may represent one or more time series and functions thereof that influence $\{Y_t\}$. Hence, we can use a monotone function $g(\cdot)$ to relate μ_t to the covariates:

$$g\left(\mu_t\right) = \mathbf{Z}_{t-1}^{\top}\boldsymbol{\theta} = \sum_{j=1}^{p} \theta_j Z_{(t-1)j}$$

where $\boldsymbol{\theta}$ is a p-dimensional vector of parameters associated with the covariates. If the link function $g(\cdot)$ is a *canonical link* and if the data is normally distributed, Fokianos and Kedem [72] showed that $\boldsymbol{\theta}$ could be estimated appropriately through partial likelihood methods that account for time-dependent data.

When applying machine-learning methods to time-series data, a typical way of representing the relationship between the observed y_t and the past values of the covariates

(features) is:

$$y_t = g(\mathbf{z}_{t-1}, \ldots, \mathbf{z}_{t-p}; \boldsymbol{\theta}) + \varepsilon_t \tag{3.1}$$

where ε_t is a shock or disturbance term, $\boldsymbol{\theta}$ is a p-dimensional parameter vector, and

$$g(\mathbf{z}_{t-1}, \ldots, \mathbf{z}_{t-p}; \boldsymbol{\theta}) = E_{\boldsymbol{\theta}}[Y_t | \mathbf{Z}_{t-1}, \ldots, \mathbf{Z}_{t-p}] \tag{3.2}$$

Here, $g(\cdot)$ could be any function: linear, nonlinear, or nonparametric, and we often do not know which one. A "learning procedure" associated with a validation scheme is often used to search for this function in machine learning approaches.

The learning procedure is a series of steps that allows us to use validation schemes to calculate the (expected) generalization errors of the methods and models. This method of evaluating the performance is of paramount importance since it guides us in choosing the best available forecasting method or model and gives us an idea of the quality of the final selection. It begins with the choice of a learning algorithm (i.e., a methodology that includes an embedded covariate/feature selection method, and a method to estimate $\boldsymbol{\theta}$) - like random forests (RF), the Rule-Based Regression (RBR) algorithm, the Box-Jenkins approach (ARIMA), or even a generalized linear models (GLM) with partial likelihood estimation. Then, we split the entire data set into two subsets, the training and test data sets, which will help us obtain the (expected) generalization errors.

The generalization error of a forecasting model $\hat{g}$ is the forecast error over an independent test set. If we partition the original time series into a training set of length N,

given by $\mathcal{D} \equiv \{(\mathbf{z}_t, y_t)\}_{t=1}^{N}$, and a test set of length[1] l, defined as $\mathcal{D}_{test} \equiv \{(\mathbf{z}_t, y_t)\}_{t=N+1}^{T}$, we can write this error as

$$\mathcal{L}_{\mathcal{D}} \equiv E_{\mathbf{Z}, Y \in \mathcal{D}_{test}} \left[\ell \left(Y, \hat{g}(\mathbf{Z}, \boldsymbol{\theta}_{\mathcal{D}}) \right) | \, \mathcal{D}, \boldsymbol{\theta}_{\mathcal{D}} \right] \tag{3.4}$$

where $\mathbf{Z} = \{\mathbf{Z}_{t-1}, \ldots, \mathbf{Z}_{t-p}\}$ and $Y = Y_t$ (we dropped the subscript for convenience). Moreover, $\boldsymbol{\theta}_{\mathcal{D}}$ is estimated using $\mathcal{D}$, and $\hat{g}(\mathbf{Z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}})$ is the out-of-sample prediction of Y after applying $\hat{\boldsymbol{\theta}}_{\mathcal{D}}$ to $\mathbf{Z} \in \mathcal{D}_{test}$. Here, $\ell(\cdot)$ is the loss function that measures the differences between Y and $\hat{g}(\cdot)$[3].

The generalization error given in Equation (3.4) is also known as the *test error*. The quality of the approximation between $\hat{g}(\cdot)$ and the true forecasting model $g(\cdot)$ would, ideally, be measured by it[4]. However, there is an issue with this metric since it is calculated for a single fixed training data set, $\mathcal{D}$. Hastie et al. [74, chapter 7] argue that this leads to a slightly larger mean absolute deviation. Their results show [74, p. 257] that, in practice, most validation schemes provide better estimates of the expected prediction error $\mathcal{L}$, given

[1] There are two ways to define the length of the training and test sets. In the first one, the analyst can define a constant value, l, for the forecast horizon. In this case, the length of the test set will be just l, and the length of the training data set will be $T - l$. In a different approach, if we let $q_t \in [1/T, 1)$ be the proportion of the entire data set that we will use as training data, then the length of the test set will be $l \equiv \lceil (1 - q_t) \cdot T \rceil$[2]. Usually, the size of the test set is equal to 20%-30% of the entire data (i.e., $(1 - q_t) \in [0.2, 0.3]$).

Therefore, there are two possibilities for the length of the training data set:

$$N \equiv \begin{cases} T - l, & \text{if } l \text{ is a constant forecast horizon} \\ \lfloor q_t \cdot T \rfloor, & \text{if } l = \lceil (1 - q_t) \cdot T \rceil, \forall q_t \in \left[\frac{1}{T}, 1\right) \end{cases} \tag{3.3}$$

[3] Typical choices are the *quadratic loss*, $\ell \left(y, \hat{g}(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}}) \right) = \left(y - \hat{g}(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}}) \right)^2$; and, the *absolute loss*, $\ell \left(y, \hat{g}(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}}) \right) = \left| y - \hat{g}(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}}) \right|$.

[4] The risk function is the expected value of the loss function [73, p. 349]. From Equation (3.4), the generalization error $\mathcal{L}_{\mathcal{D}}$ might be seen as the risk of $\hat{g}$, as mentioned in footnote 8 in the Introduction (Chapter 1).

by:

$$\mathcal{L} \equiv E_{\boldsymbol{\theta}} \left[\ell \left(Y, \hat{g}(\mathbf{Z}, \boldsymbol{\theta}) \right) \right] = E \left[\mathcal{L}_{\mathcal{D}} \right] \tag{3.5}$$

The above expectation is also known as the *expected* test error (or true error) and it is sometimes represented by *Err*, instead of $\mathcal{L}$ [74, p. 220]. Moreover, it takes the average over all random components, including the randomness in the estimation set that generated $\hat{g}$.

We can obtain a sample estimator for $\mathcal{L}$ by using validation schemes to partition the training data into an "estimation set", $\mathcal{D}\left(\mathcal{I}^e\right)$, and a non-overlapping "validation set", $\mathcal{D}\left(\mathcal{I}^v\right)$. Here, $\mathcal{I}^e$ is the non-empty proper subset of indexes from $\{1, \ldots, N\}$ that identify the observations used to **estimate** (hence, the e on $\mathcal{I}^e$) the parameters and obtain the forecasts. The set $\mathcal{I}^v$ is the complement of $\mathcal{I}^e$ and returns the index of the observations used to **validate** (accordingly, this puts the v on $\mathcal{I}^v$) the model by means of calculating its forecast accuracy. Thus, $\mathcal{I}^v \equiv \bar{\mathcal{I}}^e = \{1, \ldots, N\} \backslash \mathcal{I}^e$[5]. One way to find $\mathcal{D}\left(\mathcal{I}^e\right)$ and $\mathcal{D}\left(\mathcal{I}^v\right)$ is to take a similar approach to what was done to get $\mathcal{D}$, and $\mathcal{D}_{test}$. If we take a proportion $q_e \in [1/N, 1)$ of the training data to form the estimation set, then its length will be $N_e \equiv \lfloor q_e \cdot N \rfloor$. Consequently, the length of the validation set will be $l_v \equiv \lceil (1 - q_e) \cdot N \rceil$.

The above situation is the basis of the `Holdout` scheme (Section 3.2.1), and it is trivial to see from the discussion that this procedure partitions the training data only once. However, most schemes depend on several "rounds" of partitions made at different split points. The rationality behind this is that if one uses the `Holdout` scheme with only one

[5] Since $\mathcal{I}^e$ is a non-empty proper subset, then its complement $\mathcal{I}^v$ is also non-empty. Moreover, it is trivial to see that $\mathcal{I}^e \cap \mathcal{I}^v = \emptyset$

split, it is possible that the selected partitions that form the estimation and validation sets are not "representative.". Then, the quality of the forecast accuracy will be dependent on that single split point. By having different split points at each "round," it is hoped that the forecast accuracy of a model under a particular scheme (calculated by evaluating the mean loss of the model over all "rounds") would improve. But, since the models need to be estimated at each "round" (and considering that some machine learning algorithms already take a long time to learn the weights), the computational time of these procedures is longer than the one for the `Holdout` scheme.

Formally, these "rounds" are called *folds*. As discussed, at each fold, the training data set $\mathcal{D}$ is split differently into the estimation and validation sets. This way, the data sets used to fit and evaluate the forecast model in fold i are distinct from the ones used in fold $i+1$. The total number of folds, k, depends on the scheme (for the `Holdout` , $k = 1$).

For $i = 0, \ldots, k-1$, the sets $\mathcal{I}_i^e$ and $\mathcal{I}_i^e$ are the sets of indexes from $\{1, \ldots, N\}$ that indicate which observations will form the estimation and validation sets at the i-th fold: $\mathcal{D}\left(\mathcal{I}_i^e\right)$ and $\mathcal{D}\left(\mathcal{I}_i^v\right)$, respectively. Finally, we can define a general form for a validation scheme, $\mathcal{V}(\mathcal{D}; k)$, in those terms:

$$\mathcal{V}(\mathcal{D}; k) \equiv \left\{ \left(\mathcal{I}_i^e, \mathcal{I}_i^v\right) \middle| \, \mathcal{I}_i^e \text{ and } \mathcal{I}_i^v \subseteq \{1, \ldots, N\},\ \mathcal{I}_i^e \cap \mathcal{I}_i^v = \emptyset \right\}_{i=0}^{k-1} \tag{3.6}$$

Using the general form of a validation scheme from Equation (3.6), we can define an empirical estimator of $\mathcal{L}$ by taking the average of the mean out-of-sample losses over

all k splits [23]:

$$\hat{\mathcal{L}}(\mathcal{D}, \mathcal{V}, f) \equiv \frac{1}{k} \sum_{i=0}^{k-1} \frac{1}{\mathbf{Card}\left(\mathcal{I}_i^v\right)} \sum_{(\mathbf{z},y)\in\mathcal{D}\left(\mathcal{I}_i^v\right)} \ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}\right)\right] \tag{3.7}$$

where the forecast model f belongs to the set of all possible model formulations $\mathbb{G}$, and $\mathbf{Card}\left(\mathcal{I}_i^v\right)$ is the cardinality of the *i*-th validation index set. In essence, the above equation returns an estimate of the out-of-sample prediction error and can be used to select $\hat{g}(\cdot)$.

If one uses a *winner-takes-all* approach to evaluate between all possible $f \in \mathbb{G}$ models, then the selection of $\hat{g}$ is done by selecting the formulation f that yields the smallest *empirical* generalization error. That is,

$$\hat{g} = \underset{f\in\mathbb{G}}{\arg\min}\, \hat{\mathcal{L}}(\mathcal{D}, \mathcal{V}, f) \tag{3.8}$$

Then, the estimator of $\mathcal{L}$ for the best forecast model $\hat{g}$ is

$$\hat{\mathcal{L}}(\mathcal{D}, \mathcal{V}, \hat{g}) \equiv \frac{1}{k} \sum_{i=0}^{k-1} \frac{1}{\mathbf{Card}\left(\mathcal{I}_i^v\right)} \sum_{(\mathbf{z},y)\in\mathcal{D}\left(\mathcal{I}_i^v\right)} \ell\left[y, \hat{g}\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}\right)\right] \tag{3.9}$$

From Equations (3.7) and (3.8), we see that the choice of $\mathcal{V}$ has an impact on the result. CTM used several estimation methods with the same $\hat{g}$[6] to evaluate the performance of the different validation schemes. Here, we will use their approach with the addition of our proposed validation schemes to the analysis. By using the same $\hat{g}$ and the same estimation methods, we will be able to properly compare all the validation schemes. But

[6]However, CTM did not use the approach from Equation (3.8) to select $\hat{g}$. They used an auto-regressive process based on the time delay embedding method created by Takens [75]. More on this in Section 5.2.2.

before getting into the comparisons, it is essential to understand the characteristics of each procedure.

3.2 A framework for Forward validation schemes

3.2.1 Holdout

The most straightforward validation scheme is given by the `Holdout` approach, as shown in Figure 2.1 in the introduction (and replicated in Figure 3.1 below). It has this name because the validation data is "held out" of the data used for estimating the parameters. Some authors used this approach in a time-series context since the 1930s (as discussed in the Introduction, Sec. 1), but under different names. It probably got the name `Holdout` after the work of Devroye and Wagner [76, 77][7]. Nowadays, it is also called the *last-block* validation scheme [6, p. 193], and it relies on a single data split to create the sets that will be used to estimate the parameters and validate the forecasting model.

If we take only a proportion $q_e \in [1/N, 1)$ of the training data[8], $\mathcal{D} = \{(z_t, y_t)\}_{t=1}^{N}$, to create our estimation set, we can use the general validation scheme given in Equation (3.6) to write the `Holdout` validation scheme $\mathcal{V}^{\text{HO}}(\mathcal{D}; q_e)$ [9]:

$$\mathcal{V}^{\text{HO}}(\mathcal{D}; q_e) \equiv \{ (\mathcal{I}^e, \mathcal{I}^v) | \mathcal{I}^e = \{1, \ldots, \lfloor q_e \cdot N \rfloor\}, \mathcal{I}^v = \{\lfloor q_e \cdot N \rfloor + 1, \ldots, N\}\} \tag{3.10}$$

[7] It is worth mentioning that these authors developed their methodology using independent data.

[8] Since Equation(3.6) was defined in terms of the *estimation set* and the validation set, we are assuming that the data was already split into training data and test data by a proportion that may, or may not, be equal to q_t (this choice depends on the analyst).

[9] Since $i = 0, \ldots, k-1$, then $k = 1$ (because it is a single split), which implies that $i = 0$. Hence, we will suppress the i in the notation for the `Holdout` scheme.

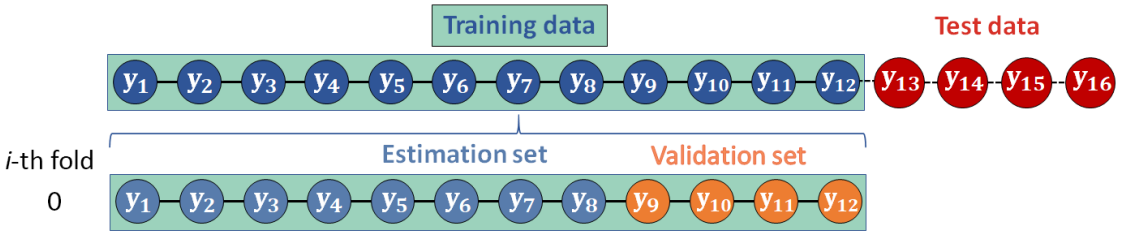


Figure 3.1: Example of a data set divided using the `Holdout` forward validation scheme with $T = 16$, $q_t = 0.8$, and $q_e = 0.7$.

From Equation (3.10) and Figure 3.1, it is trivial to see that the estimation and validation sets are composed by $\mathcal{D}\left(\mathcal{I}^e\right) = y_1, \ldots, y_{\lfloor q_e \cdot N \rfloor}$ and $\mathcal{D}\left(\mathcal{I}^v\right) = y_{\lfloor q_e \cdot N \rfloor + 1}, \ldots, y_N$, respectively.

There is a different way to write Equation (3.10) that will be useful when we discuss the *p-Holdout* family in Chapter 4. Recall from the previous section that the length of the validation set under the `Holdout` scheme is:

$$l_v^{\text{HO}} \equiv \lceil (1 - q_e) \cdot N \rceil, \text{ for } q_e \in [1/N, 1) \tag{3.11}$$

And if we notice that

$$\begin{aligned} N - l_v^{HO} &= N - \lceil (1 - q_e) \cdot N \rceil \\ &= N - N + \lfloor q_e \cdot N \rfloor \\ &= \lfloor q_e \cdot N \rfloor \end{aligned}$$

we can substitute $\lfloor q_e \cdot N \rfloor$ in Equation (3.10) for $N - l_v^{HO}$, and rewrite this equation as:

$$\mathcal{V}^{\text{HO}}(\mathcal{D}; q_e) = \Bigg\{ (\mathcal{I}^e, \mathcal{I}^v) |\ \mathcal{I}^e = \left\{1, \ldots, \left(N - l_v^{HO}\right)\right\}, \\ \mathcal{I}^v = \left\{\left(N - l_v^{HO} + 1\right), \ldots, N\right\} \Bigg\} \quad (3.12)$$

Finally, we can then use all this to write Equation (3.7), the average expected out-of-sample error, for the `Holdout` case:

$$\hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}^{\text{HO}}, f\right) \equiv \frac{1}{\lceil (1 - q_e) \cdot N \rceil} \sum_{(\mathbf{z}, y) \in \mathcal{D}(\mathcal{I}^v)} \ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}(\mathcal{I}^e)}\right)\right] \quad (3.13)$$

or

$$\hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}^{\text{HO}}, f\right) \equiv \frac{1}{l_v^{HO}} \sum_{(\mathbf{z}, y) \in \mathcal{D}(\mathcal{I}^v)} \ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}(\mathcal{I}^e)}\right)\right] \quad (3.14)$$

With the results from Equation (3.13) for each forecasting model f, we can find $\hat{g}$ using Equation (3.8). CTM used a different way to obtain $\hat{g}$, as we will discuss in Section 5.2.2. Nevertheless, with $\hat{g}$, we move forward to evaluate its forecasting accuracy on the test data. That is, we fit $\hat{g}$ using the entire training data $\mathcal{D}$, obtain the forecasts, and calculate $\hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}^{\text{HO}}, \hat{g}\right)$ based on Equation (3.14). Then, we evaluate the generalization performance by comparing this last measure to the error obtained from the test data, $\mathcal{D}_{test}$, by calculating:

$$\hat{\mathcal{L}}(\mathcal{D}, \mathcal{D}_{test}, \hat{g}) = \frac{1}{l} \sum_{(\mathbf{z}, y) \in \mathcal{D}_{test}} \ell\left[y, \hat{g}\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}}\right)\right] \quad (3.15)$$

CTM and Schnaubelt [23, p. 11] call the result from the above equation the "ground truth loss." Schnaubelt [23, p. 25] also calls it the "true out-of-sample generalization error." Both authors use Equation (3.15) as the basis to evaluate the out-of-sample performance of the validation schemes (see Section 5.2.6).

3.2.2 Rep-Holdout

The idea behind the *repeated holdout* validation scheme (or, `Rep-Holdout`) is based on taking several "rounds" (folds) of the holdout procedure, where the split point, a_i, is randomly selected (without replacement) from a sampling window at each fold.

To get this window's range, we need to select a proportion $q_e \in [1/N, 1)$ of the training data to be used as the estimation set and a proportion $q_v < (1 - q_e)$ for the validation set. So, in this case, the length of the estimation and validation sets are

$$N_e^{repHO} \equiv \lfloor q_e \cdot N \rfloor \tag{3.16}$$

$$l_v^{repHO} \equiv \lfloor q_v \cdot N \rfloor \tag{3.17}$$

Then, for every fold $i = 0, \ldots, k-1$, possible values of a_i are randomly taken from the window

$$\left\{ \left(N_e^{repHO} + 1 \right), \ldots, \left(N - l_v^{repHO} + 1 \right) \right\}$$

The split-point point a_i marks the beginning of the validation set for the i-th fold, while the end-point is given by $a_i + l_v^{repHO} - 1$. And since the selection is without replacement, the number of folds k is, at most, equal to the length of the above sampling

window. That is,

$$k \leq N - N_e^{repHO} - l_v^{repHO} + 1.$$

The indexes of the observations in the estimation set are also shifted, depending on a_i. This is done to make the length of the estimation set the same across the folds, and to avoid gaps between the estimation and validation sets. Thus, the last index of the estimation set is equal to $a_i - 1$, and the first is given by $a_i - N_e^{repHO}$. The final indexes can be seen in Equation (3.18) below. An example of this scheme applied to a data set is shown in Figure 3.2.

Figure 3.2: Example of a data set divided using the `Rep-Holdout` forward validation scheme with $T = 16$, $q_t = 0.8$, $q_e = 0.6$, $q_v = 0.2$, $k = 3$. The selection range for each a_i goes from the 8th observation to the 11th, and the values $a_1 = 10$, $a_2 = 9$, and $a_3 = 11$ were randomly selected.

Using Equation (3.6), the `Rep-Holdout` validation scheme, $\mathcal{V}^{\text{repHO}}(\mathcal{D}; q_e, q_v)$,

is:

$$\mathcal{V}^{\text{repHO}}(\mathcal{D}; q_e, q_v) \equiv \Bigg\{ (\mathcal{I}_i^e, \mathcal{I}_i^v) |\ \mathcal{I}_i^e = \{(a_i - \lfloor q_e \cdot N \rfloor), \ldots, (a_i - 1)\}, \\ \mathcal{I}_i^v = \{a_i, \ldots, (a_i + \lfloor q_v \cdot N \rfloor - 1)\} \Bigg\}_{i=0}^{k-1} \tag{3.18}$$

The estimated expected out-of-sample prediction error (Equation 3.7) becomes:

$$\hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}^{\text{repHO}}, f\right) = \frac{1}{k} \cdot \frac{1}{\lfloor q_v \cdot N \rfloor} \cdot \sum_{i=0}^{k-1} \sum_{(\mathbf{z},y) \in \mathcal{D}\left(\mathcal{I}_i^v\right)} \ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}\right)\right] \tag{3.19}$$

where $\mathcal{D}\left(\mathcal{I}_i^e\right)$ and $\mathcal{D}\left(\mathcal{I}_i^v\right)$ use the observations for which the indexes are given in Equation (3.18).

3.2.3 Rolling Origin and Prequential Growing Window

Small samples can cause problems to the previous procedures. One scheme that makes a more effective use of data is the *successive updating* procedure [7, 31], commonly called the *rolling-origin* validation scheme [5]. To explain it, we need to introduce the concept of *forecasting origin*, and to do so, let us take a data set divided into training data and test data (Figure 2.1 - A). If we want to produce forecasts for a lead time (or forecasting horizon) l, then the final time in the fit period is the point from which the forecasts are *originated*. This point is called the *forecasting origin*.

In other words, we first fit the model using all the information up until N (recall that $N \equiv T - l$, where T is the total length of the time series) and produce forecasts for all subsequent periods $N + 1, \ldots, N + l$. Then, we take the same model formulation and

its estimated parameters and apply them to an "updated" data set comprised of all data up until $N + 1$, and use it to produce forecasts for the periods $N + 2, \ldots, N + l$. Because of that, this scheme is also called the rolling origin *update* [6]. When the model is re-estimated (retrained) at each fold, the procedure is called the rolling origin **recalibration** scheme. In it, the estimated parameters at fold i are applied to the training data in the same fold only. We keep "rolling" the origins until $N + l - 1$ and obtain (in general) $l \cdot (l + 1)/2$ forecasts. The test data is then used to calculate the forecast accuracy. A schematic illustration of the rolling origin *update* scheme was adapted from Armstrong [31, p. 343] and is shown in Figure 3.3.

Figure 3.3 - Panel A shows the rolling origin *update* procedure, in which the parameters estimated from the data in fold $i = 0$ (blue points and cyan background) and are applied to the data in folds $1, 2,$ and 3 (blue points and white background) to obtain the forecasts (pink points). Panel B displays the data split according to the rolling origin **recalibration** scheme, where the parameters are estimated from the data at each fold (blue points and cyan background).

Both processes can easily be extended to be used with the estimation set and the validation set. It can also accommodate "rolls" (shifts) larger than 1. If we let $q_e \in \left[\frac{1}{N}, 1\right)$ be the minimum fraction of the training data used to form the estimation set, then the

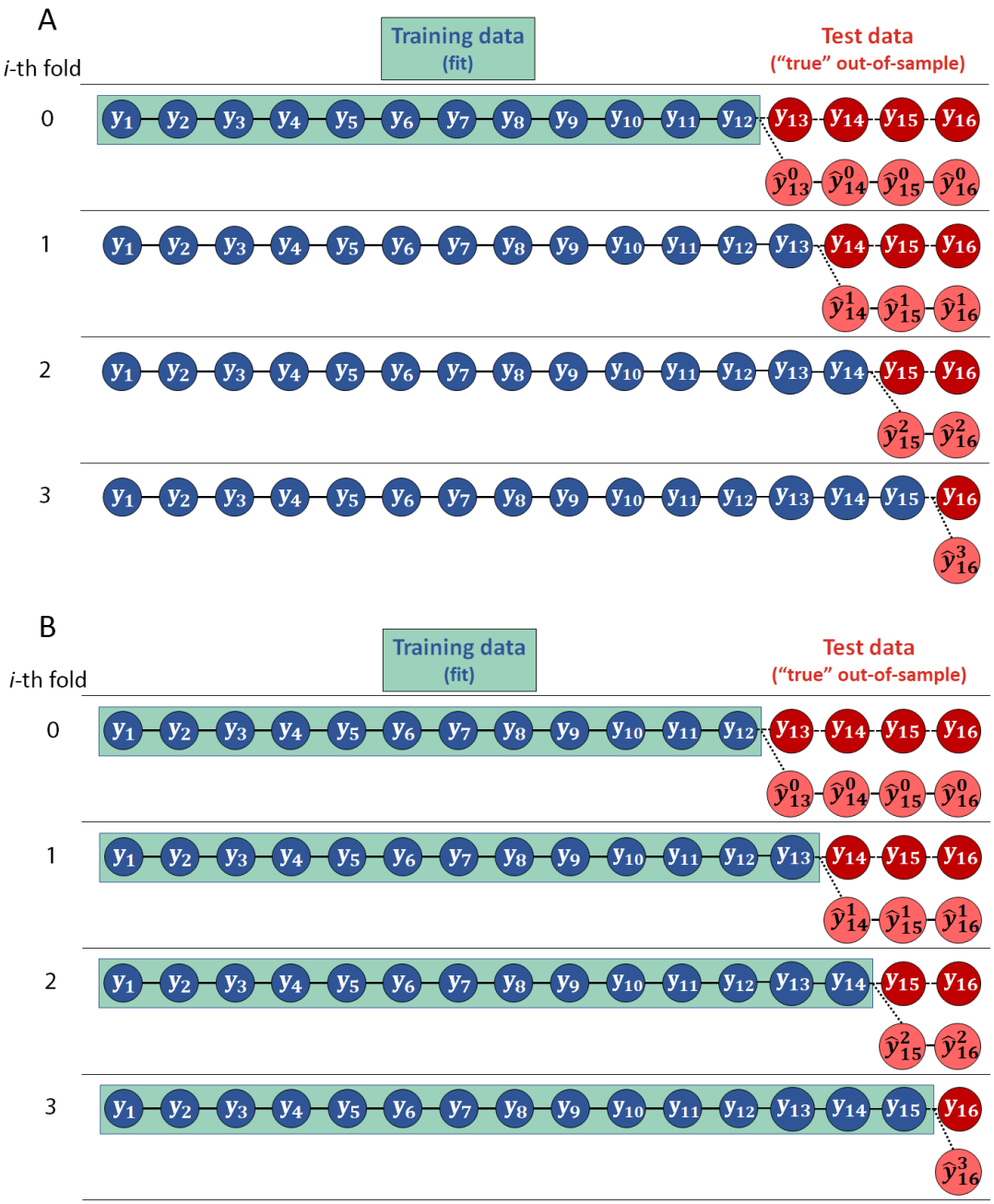


Figure 3.3: Example of the rolling origin forward validation scheme applied to a univariate data set with $T = 16$ and $l = 4$. Panel A shows the rolling origin *update* procedure, and Panel B depicts the rolling origin **recalibration** scheme.

number of shifts can be defined as[10]

$$\kappa^{RO} \equiv \frac{(1-q_e)\cdot N}{k}.$$

Then, the *rolling-origin validation scheme*, $\mathcal{V}^{\text{RO}}(\mathcal{D}; N, \kappa^{RO}, q_e)$, is written as:

$$\mathcal{V}^{\text{RO}}\left(\mathcal{D}; \kappa^{RO}, q_e\right) \equiv \left\{ (\mathcal{I}_i^e, \mathcal{I}_i^v) | \, \mathcal{I}_i^e = \left\{1, \ldots, \lfloor q_e \cdot N + i\kappa^{RO} \rfloor\right\}, \right.$$
$$\left. \mathcal{I}_i^v = \left\{\lfloor q_e \cdot N + i\kappa^{RO} \rfloor + 1, \ldots, N\right\} \right\}_{i=0}^{k-1} \tag{3.20}$$

Both the *update* and **recalibration** schemes split the sample using the scheme given in Equation (3.20). The distinction between the two will be seen in the estimation of $\hat{\boldsymbol{\theta}}$ inside the loss function from the out-of-sample generalization error equation (Equation 3.7). For the *update* scheme, $\hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_0^e\right)}$ is used to obtain the loss in all folds, while in the **recalibration** scheme, we re-estimate $\hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}$ at each fold.

The *prequential growing window* [22, p. 2009], or prequential landmark scheme[11], is a particular case of the rolling-origin **recalibration** procedure, where the forward shift is restricted to 1 (Figure 3.4). Therefore, $\kappa^{RO} = 1 \implies k = \lceil (1-q_e)\cdot N \rceil$.

With this, the *prequential growing window validation scheme*, $\mathcal{V}^{\text{PG}}(\mathcal{D}; N, q_e)$, can

[10] If the number of rolls, κ^{RO}, is a fixed integer instead, then

$$k = \left\lceil \frac{(1-q_e)\cdot N}{\kappa^{RO}} \right\rceil$$

[11] The prequential growing window scheme is different from the *growing window* procedures used by Hjorth [44] and Makridakis [28]. These authors used only the observation that immediately follows the estimation set as their validation set.

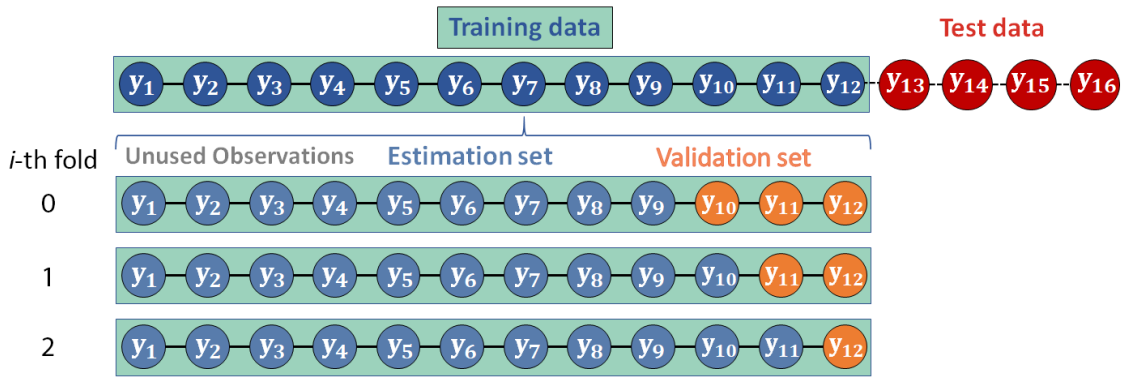


Figure 3.4: Example of the prequential growing window forward-validation scheme applied to a single univariate data set with $T = 16$, $q_t = 0.8$, $q_e = 0.8$ and $k = 3$.

be written as:

$$\mathcal{V}^{\text{PG}}(\mathcal{D}; N, q_e) \equiv \left\{ (\mathcal{I}_i^e, \mathcal{I}_i^v) |\ \mathcal{I}_i^e = \{1, \ldots, \lfloor q_e \cdot N + i \rfloor\}, \right. \\ \left. \mathcal{I}_i^v = \{\lfloor q_e \cdot N + i \rfloor + 1, \ldots, N\} \right\}_{i=0}^{k-1} \tag{3.21}$$

Using Equation (3.7), the estimated expected out-of-sample prediction error for the *prequential growing window validation scheme* (or prequential landmark scheme) is defined as:

$$\hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}^{\text{PG}}, f\right) = \frac{1}{k} \sum_{i=0}^{k-1} \frac{1}{(N - \lfloor q_e \cdot N + 1 \rfloor)} \sum_{(\mathbf{z},y) \in \mathcal{D}\left(\mathcal{I}_i^v\right)} \ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}\right)\right] \tag{3.22}$$

where $\mathcal{D}\left(\mathcal{I}_i^e\right)$ and $\mathcal{D}\left(\mathcal{I}_i^v\right)$ use the observations for which the indexes are given in Equation (3.21), and the same model specification f is used in every fold.

3.2.4 Prequential Sliding Window

The exhaustive *prequential sliding window* method works as a modification of the prequential landmark procedure (Section 3.2.3), where the first i observations are deleted at the i-th fold. An example is shown in Figure 3.5.

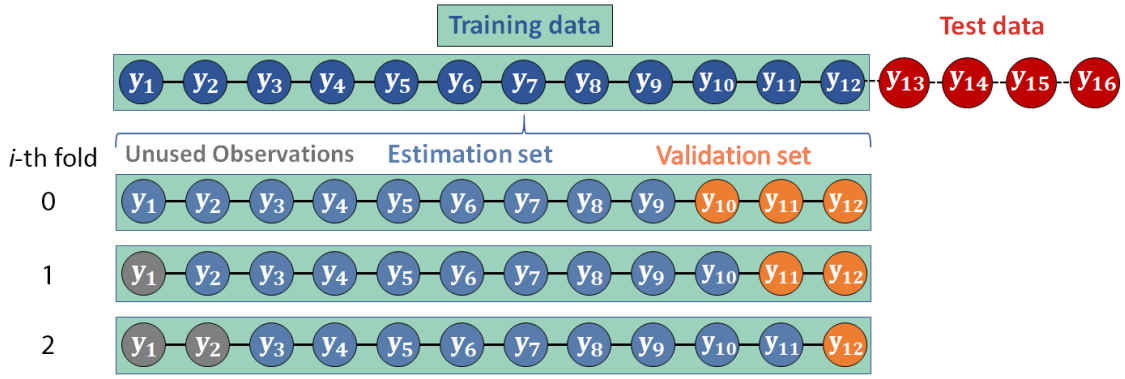


Figure 3.5: Example of the prequential sliding window validation scheme applied to a univariate data set with $T = 16$, $q_t = 0.8$, $q_e = 0.8$, and $\kappa^{RW} = 1$. The model is re-estimated at every estimation set (blue dots) and used to find the one-step-ahead forecast (orange dot).

The *prequential sliding window validation scheme*, $\mathcal{V}^{\text{PSW}}(\mathcal{D}; N, q_e, l)$, is defined as:

$$\mathcal{V}^{\text{PSW}}(\mathcal{D}; q_e) \equiv \left\{ (\mathcal{I}_i^e, \mathcal{I}_i^v) |\ \mathcal{I}_i^e = \{i+1, \ldots, \lfloor q_e \cdot N + i \rfloor\}, \right. \\ \left. \mathcal{I}_i^v = \{\lfloor q_e \cdot N + i \rfloor + 1, \ldots, N\} \right\}_{i=0}^{l-1} \quad (3.23)$$

where $k = \lceil (1 - q_e) \cdot N \rceil = l$.

And, the estimated expected out-of-sample prediction error for the *prequential slid-*

ing window scheme can be written:

$$\hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}^{\text{PSW}}, f\right) = \frac{1}{l}\sum_{i=0}^{l-1} \frac{1}{(N - \lfloor q_e \cdot N + i \rfloor)} \sum_{(\mathbf{z},y)\in\mathcal{D}\left(\mathcal{I}_i^v\right)} \ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}\right)\right] \quad (3.24)$$

where $\mathcal{D}\left(\mathcal{I}_i^e\right)$ and $\mathcal{D}\left(\mathcal{I}_i^v\right)$ use the observations for which the indices are given in Equation (3.21).

3.2.5 Prequential in Blocks

The idea behind the *prequential in blocks* validation scheme is due to Dawid [78], and it relies on the partition of the time series into sets of growing cardinality. Let $\kappa^{PBls} \approx N/k$ be the number of observations that each fold is rolled forward by, and let $A_0, \ldots, A_{k-2}$, be ordered partitions of the indexes $\{1, \ldots, N\}$ of the observations in $\mathcal{D}$ such that $A_0 \subset A_1 \subset \ldots \subset A_{k-2}$ with $\mathbf{Card}(A_i) \approx (i+2) \cdot \kappa^{PBls} \leq N$. That is,

$$A_i \approx \{1, \ldots, (i+2) \cdot \kappa^{PBls}\}, \text{ for } i = 0, \ldots, k-2. \quad (3.25)$$

In other words, the sets A_i give the indexes of the observations that will comprise both the estimation and validation sets at each fold. For example, let $N = 12$ and $k = 3$. Then, $\kappa^{PBls} = 4$ and for $i = 0, \ldots, k-2$:

$$A_0 = \{1, \ldots, (0+2) \cdot 4\} = \{1, 2, 3, 4, 5, 6, 7, 8\}$$

$$A_1 = \{1, \ldots, (1+2) \cdot 4\} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$$

At the i-th fold, the estimation set is formed of all the indexes of A_i up until the $(i+1)\cdot\kappa^{PBls}$-th case, while the validation set contains all the remaining cases. In the example above, the estimation sets at folds 0 and 1 contain the first 4 and 8 indexes, respectively. This example is represented in Figure 3.6. From it, it is easy to see that the validation set of the previous fold is incorporated into the estimation set of the current fold.

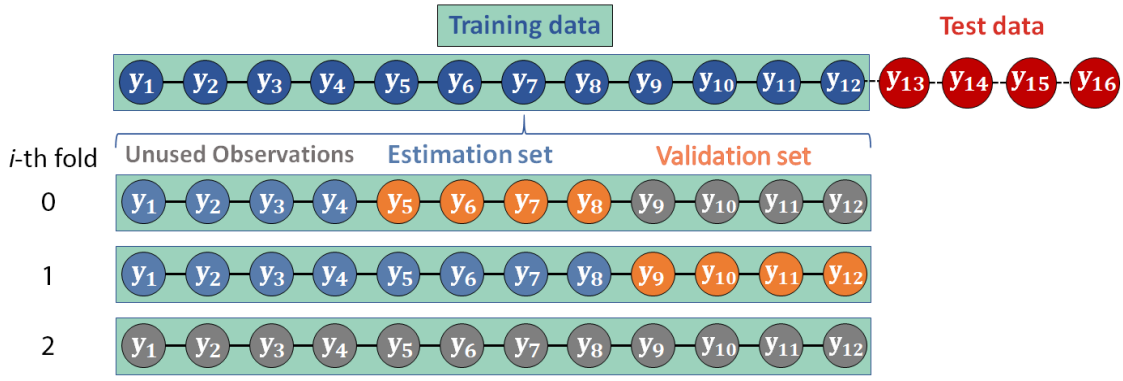


Figure 3.6: Example of a time series partitioned under the prequential in blocks validation scheme with $T = 16$, $q_t = 0.8$, $q_e = 0.8$, and $k = 3$. Fold number 2 is not used at all.

We write a general form of $\mathcal{V}^{\text{PBls}}(\mathcal{D};\kappa^{PBls})$, the *prequential in blocks validation scheme*, as:

$$\mathcal{V}^{\text{PBls}}(\mathcal{D};\kappa^{PBls}) \equiv \left\{ (\mathcal{I}_i^e, \mathcal{I}_i^v) |\ \mathcal{I}_i^e \approx \left\{1,\ldots,(i+1)\cdot\kappa^{PBls}\right\}, \right. \\ \left. \mathcal{I}_i^v \approx A_i \setminus \mathcal{I}_i^e \right\}_{i=0}^{k-2} \quad (3.26)$$

We stated that the sets $\mathcal{I}_i^e$ and $\mathcal{I}_i^v$ are approximately equal to their respective sets of

indexes because the ratio $\kappa^{PBls} \approx N/k$ might not return an integer, and depending on the computational device used, the value of κ^{PBls} can be either $\lfloor N/k \rfloor$ or $\lceil N/k \rceil$. Moreover, it is possible that κ^{PBls} is different for distinct values of i[12]. To account for that, we define a delta function as:

$$\delta(N,k,i) = \begin{cases} \lfloor N/k \rfloor, \text{ if the computer returns } \lfloor \kappa^{PBls} \rfloor \text{ for the } i\text{-th case} \\ \\ \lceil N/k \rceil, \text{ otherwise.} \end{cases} \tag{3.27}$$

Then we can rewrite Equation (3.26) as:

$$\begin{aligned} \mathcal{V}^{\text{PBls}}(\mathcal{D};k) = \Bigg\{ & (\mathcal{I}_i^e, \mathcal{I}_i^v) |\ \mathcal{I}_i^e = \{1,\ldots,(i+1)\cdot\delta(N,k,i)\}, \\ & \mathcal{I}_i^v = \{[(i+1)\cdot\delta(N,k,i)+1], \ldots, (i+2)\cdot\delta(N,k,i)\} \Bigg\}_{i=0}^{k-2} \end{aligned} \tag{3.28}$$

where N is the number of observations in $\mathcal{D}$.

Using Equation (3.26) to obtain the indices for $\mathcal{D}(\mathcal{I}_i^e)$ and $\mathcal{D}(\mathcal{I}_i^v)$, the estimated expected out-of-sample prediction error for the *prequential in blocks* can be written:

$$\begin{aligned} \hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}^{\text{PBls}}, f\right) &= \frac{1}{k-1}\sum_{i=0}^{k-2} \frac{1}{\textbf{Card}\left(A_i \setminus \mathcal{I}_i^e\right)} \sum_{(\mathbf{z},y)\in\mathcal{D}\left(\mathcal{I}_i^v\right)} \ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}\right)\right] \\ &= \frac{1}{k-1}\sum_{i=0}^{k-2} \frac{1}{\delta(N,k,i)} \sum_{(\mathbf{z},y)\in\mathcal{D}\left(\mathcal{I}_i^v\right)} \ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}\right)\right] \end{aligned} \tag{3.29}$$

[12]While code tracing CTM's program, we noticed that this is the behavior of the `R` function that they used

3.2.6 Prequential Sliding Blocks

The *prequential sliding blocks* validation scheme is a modification of the prequential in blocks procedure. The difference here is that, at each fold, all the observations in the estimation sets of the previous folds are discarded from the estimation set used in the current fold. From Figure 3.7, we see that the estimation set "slides" over the training set. And if we compare this figure with Figure 3.6 seen above, we see that observations y_1, y_2, y_3, and y_4 form the estimation set at the fold $i = 0$ but are unused in the estimation set at fold $i = 1$.

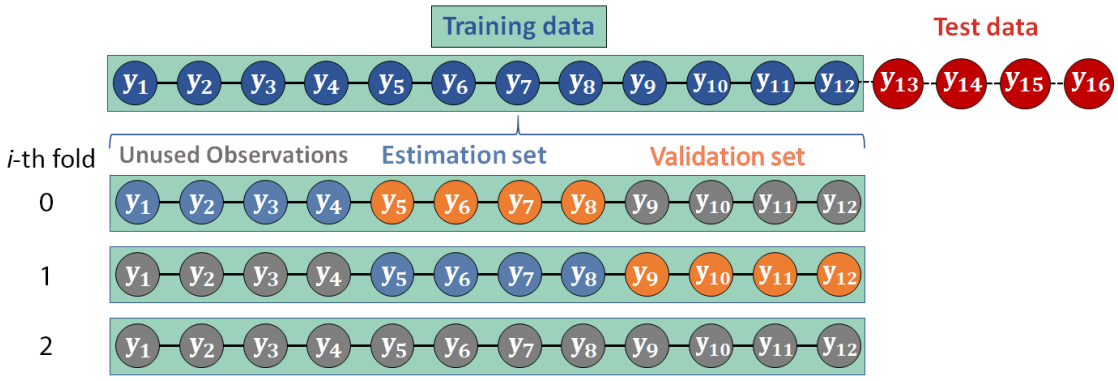


Figure 3.7: Example of a data set divided using the prequential sliding blocks forward validation scheme with $T = 16$, $q_t = 0.8$, $q_e = 0.8$ and $k = 3$. Fold 2 is not used.

Then, if $\kappa^{PBls} \approx N/k$ is the number of observations that each fold is rolled forward by, and $A_0, \ldots, A_{k-1}$, are ordered partitions of the indices $\{1, \ldots, N\}$ of the observations in $\mathcal{D}$ such that $A_0 \subset A_1 \subset \ldots \subset A_{k-1}$ with $\mathbf{Card}(A_i) \approx (i+2) \cdot \kappa^{PBls} \leq N$, then we

can write the *sliding prequential blocks validation scheme*, $\mathcal{V}^{\text{SPBls}}(\mathcal{D}; \kappa^{PBls})$ as:

$$\mathcal{V}^{\text{SPBls}}(\mathcal{D}; \kappa^{PBls}) \equiv$$
$$\left\{ (\mathcal{I}_i^e, \mathcal{I}_i^v) |\ \mathcal{I}_i^e \approx \left\{1, \ldots, (i+1)\kappa^{PBls}\right\} \setminus \left(\bigcup_{j=0}^{i-1} \mathcal{I}_j^e \right), \mathcal{I}_i^v \approx A_i \setminus \left(\bigcup_{j=0}^{i} \mathcal{I}_j^e \right) \right\}_{i=0}^{k-2} \tag{3.30}$$

where $\mathcal{I}_{-1}^f = \varnothing$.

The delta notation (Equation 3.27) greatly simplifies the above equation:

$$\mathcal{V}^{\text{SPBls}}(\mathcal{D}; N, k) = \left\{ (\mathcal{I}_i^e, \mathcal{I}_i^v) | \right.$$
$$\mathcal{I}_i^e = \left\{ [i \cdot \delta(N,k,i) + 1], \ldots, (i+1) \cdot \delta(N,k,i) \right\},$$
$$\left. \mathcal{I}_i^v = \left\{ [(i+1) \cdot \delta(N,k,i) + 1], \ldots, (i+2) \cdot \delta(N,k,i) \right\} \right\}_{i=0}^{k-2} \tag{3.31}$$

where N is the length of the set $\mathcal{D}$.

Taking the indexes from Equation (3.31), the error $\hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}^{\text{SPBls}}, f\right)$ is:

$$\begin{aligned} \hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}^{\text{SPBls}}, f\right) &= \frac{1}{k-1} \sum_{i=0}^{k-2} \frac{1}{\textbf{Card}\left[A_i \setminus \left(\bigcup_{j=0}^{i} \mathcal{I}_j^f \right) \right]} \sum_{(\mathbf{z},y) \in \mathcal{D}\left(\mathcal{I}_i^v\right)} \ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)} \right) \right] \\ &= \frac{1}{k-1} \sum_{i=0}^{k-2} \frac{1}{\delta(N,k,i)} \sum_{(\mathbf{z},y) \in \mathcal{D}\left(\mathcal{I}_i^v\right)} \ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)} \right) \right] \end{aligned} \tag{3.32}$$

3.2.7 Prequential Blocks with Gaps

Another modification of the prequential in blocks procedure is the *prequential blocks with gaps* validation scheme. Here, the same estimation set is used, but the validation set (as specified for prequential in blocks case) is skipped, and the next set of data is used instead. Figure 3.8 shows an example of the prequential blocks with gaps scheme. The rationale behind this procedure is that one may increase the independence between the estimation and validation sets by adding a gap between the two sets.

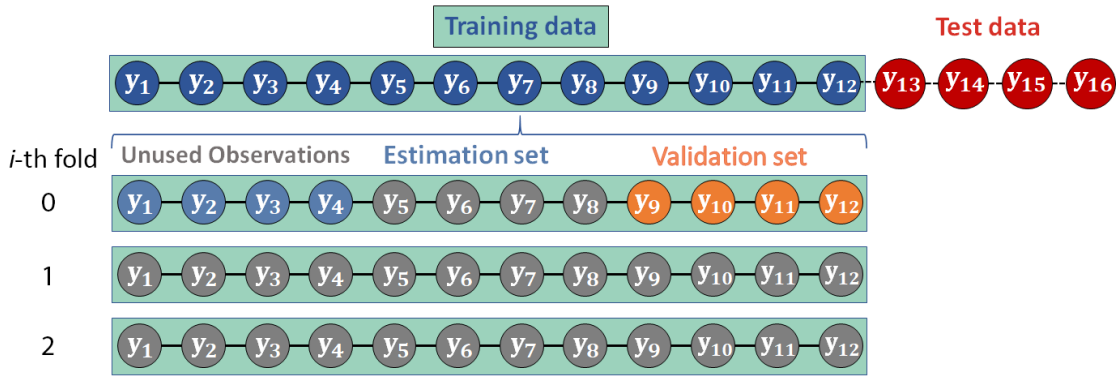


Figure 3.8: Example of a time series split according to the prequential in blocks with gaps validation scheme with $T = 16$, $q_t = 0.8$, $q_e = 0.8$, and $k = 3$. Folds 1 and 2 are not used at all.

Take the definitions of κ^{PBls} from Section 3.2.5 and A_i from Equation (3.25). Add to those definitions the following partitions of $\{1, \ldots, N\}$:

$$B_i \approx \{1, \ldots, (i+3) \cdot \kappa^{PBls}\}, \text{ for } i = 0, \ldots, k-3. \tag{3.33}$$

It is easy to see from the definition of A_i and B_i, that $A_i \subset B_i$ with $B_i \setminus A_i \neq \varnothing$,

for $i = 0, \ldots, k-3$. Inside this framework, we define the *prequential in blocks with gaps validation scheme*, $\mathcal{V}^{\text{PBG}}(\mathcal{D}; \kappa^{PBls})$ as:

$$\mathcal{V}^{\text{PBG}}(\mathcal{D}; \kappa^{PBls}) \equiv \left\{ (\mathcal{I}_i^e, \mathcal{I}_i^v) | \, \mathcal{I}_i^e \approx \left\{1, \ldots, (i+1)\kappa^{PBls}\right\}, \mathcal{I}_i^v \approx B_i \setminus A_i \right\}_{i=0}^{k-3}$$

Using the delta function defined in Equation (3.27), and with N obtained from $\mathcal{D}$, the scheme above simplifies to:

$$\begin{aligned} \mathcal{V}^{\text{PBG}}(\mathcal{D}; k) = \Bigg\{ (\mathcal{I}_i^e, \mathcal{I}_i^v) | \; & \mathcal{I}_i^e = \{1, \ldots, (i+1) \cdot \delta(N, k, i)\}, \\ & \mathcal{I}_i^v = \{[(i+2) \cdot \delta(N, k, i) + 1], \ldots, (i+3) \cdot \delta(N, k, i)\} \Bigg\}_{i=0}^{k-3} \end{aligned} \tag{3.34}$$

And the out-of-sample generalization error becomes:

$$\begin{aligned} \hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}^{\text{PBG}}, f\right) &= \frac{1}{k-2} \sum_{i=0}^{k-3} \frac{1}{\textbf{Card}\left(B_i \setminus A_i\right)} \sum_{(\mathbf{z}, y) \in \mathcal{D}\left(\mathcal{I}_i^v\right)} \ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}\right)\right] \\ &= \frac{1}{k-2} \sum_{i=0}^{k-3} \frac{1}{\delta(N, k, i)} \sum_{(\mathbf{z}, y) \in \mathcal{D}\left(\mathcal{I}_i^v\right)} \ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}\right)\right] \end{aligned} \tag{3.35}$$

3.3 A framework for Cross-validation schemes

3.3.1 Leave-one-out and k-Fold Cross-Validation

Cross-validation schemes are frequently used in the field of multivariate statistics when variables are independent and identically distributed (i.i.d.), especially for model se-

lection in classification and regression settings. In a regression setting, the *leave-one-out validation scheme* (LOO or LOOCV) was used by Stone [29][13]. This is the most classical exhaustive procedure since the number of folds is equal to the number of observations in the training set (i.e., $k = N$).

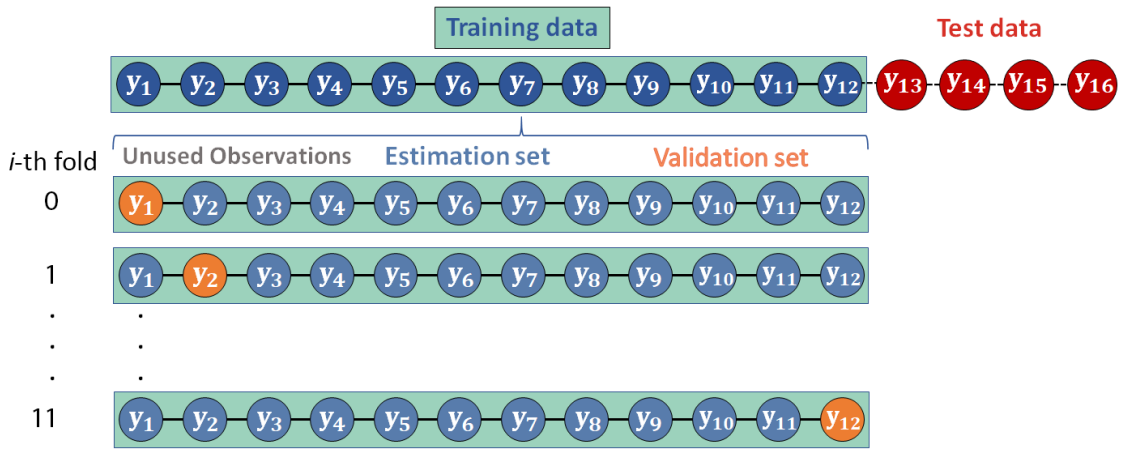


Figure 3.9: Example of the leave-one-out cross-validation scheme applied to a univariate time series with $T = 16$, $q_t = 0.8$, and $k = \lfloor q_t T \rfloor = 12$.

That is, at fold $i = 0, \ldots, k-1$, the validation set is comprised of only one observation - the one with the index equal to $i + 1$. The same $i + 1$-th index is deleted from the estimation set at each fold $i = 0, \ldots, k-1$. Consequently, the sets of indexes have the following cardinalities: $\mathbf{Card}\left(\mathcal{I}_i^e\right) = N - 1,$ and $\mathbf{Card}\left(\mathcal{I}_i^v\right) = 1,\ \forall i = 0, \ldots N-1$. A schematic example is shown in Figure 3.9. Based on the discussion, the *leave-one-out*

[13] Stone [29] called it "ordinary cross-validation." Independently, [79] used the same approach to create the extended PRESS (Prediction Sum of Squares) criterion for variable selection. Burman and Nolan [63] extended it to the dependent case when f is estimated via nonparametric techniques, under the assumption that the prediction errors are uncorrelated.

cross-validation scheme, $\mathcal{V}^{\text{LOO}}(\mathcal{D}; k)$ is defined as:

$$\mathcal{V}^{\text{LOO}}(\mathcal{D}; k) \equiv \left\{ (\mathcal{I}_i^e, \mathcal{I}_i^v) |\ \mathcal{I}_i^e = \{1, \ldots, N\} \setminus \mathcal{I}_i^v,\ \mathcal{I}_i^v = \{i+1\} \right\}_{i=0}^{N-1} \tag{3.36}$$

Breiman et al. [80, in the text of 50, p. 503] proposed the *k-fold validation scheme*[14] as an alternative to the computationally expensive leave-out-one procedure. It relies on shuffling the observations in the time series and dividing the shuffled training set into k mutually exclusive subsets of approximately equal size. Each subset works as the validation set for the respective fold. Figure 3.10 shows a schematic illustration of this method.

On Panel A in Figure 3.10, we have that y_4, y_5, y_7, and y_{10} where the observations were randomly selected to comprise the validation set at fold 0. But since the training set has been shuffled prior to selection, the actual ordering of the observations is shown in Panel B. Notice how the indexes in both the estimation and validation sets change in B. The time series with the ordering shown in B (blue dots) will be the ones used to fit the forecasting models at each fold, and at fold 0, the forecast errors will be calculated using y_5, y_{10}, y_7, y_4, in that order.

Formally, let $C_0, \ldots, C_{k-1}$ be the sets formed from random partitions of $\{1, \ldots, N\}$ taken without replacement, such that each set has approximately N/k elements (i.e., $\mathbf{Card}\,(C_i) = \delta(N, k, i)$, $i = 0, \ldots, k-1$, where $\delta(N, k, i)$ is the delta function defined

[14] Breiman et al. [80] named it "v-fold cross-validation." We changed it to k to be consistent with the rest of our notation and also because recent studies [22, 23] use this terminology.

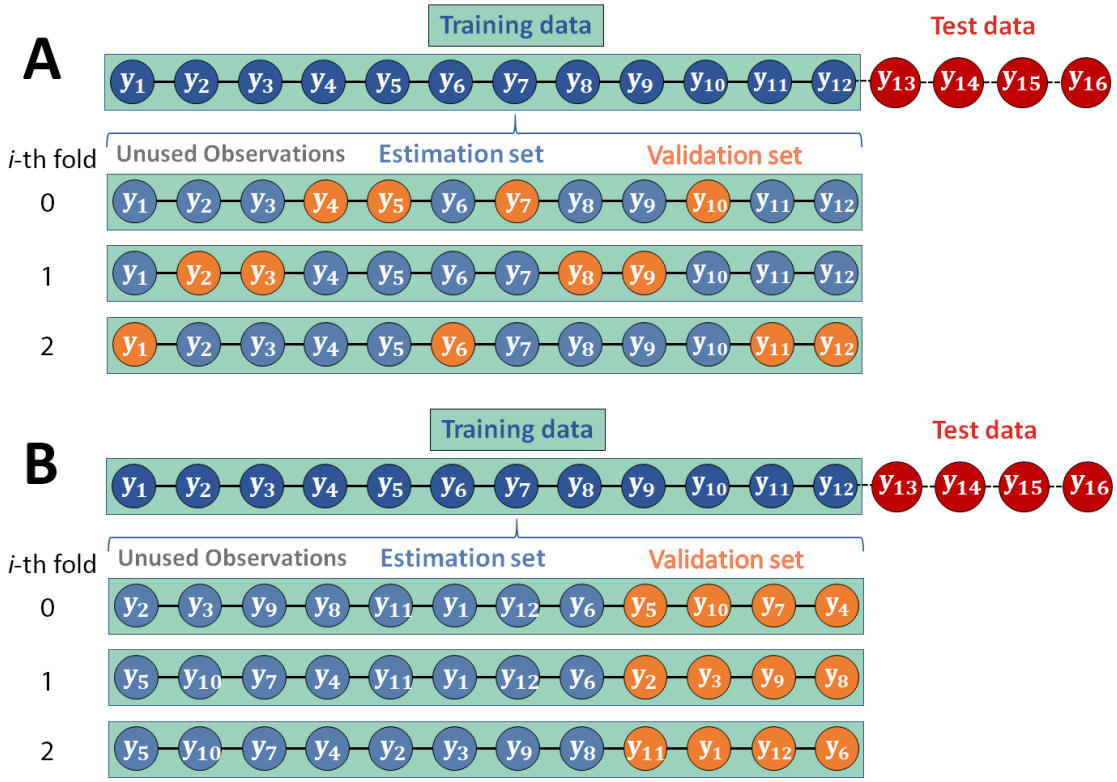


Figure 3.10: Example of the k-fold cross-validation scheme applied to a univariate time series with $T = 16$, $q_t = 0.8$, $q_e = 0.8$ and $k = 3$. Panel A shows the selected observations in their original ordering. Panel B shows them according to their selection order.

in Equation 3.27). Then, we define the *k-fold cross-validation scheme* as:

$$\mathcal{V}^{\text{CV}}(\mathcal{D}; k) \equiv \Big\{ (\mathcal{I}_i^e, \mathcal{I}_i^v) |\ \mathcal{I}_i^e = \{1, \ldots, N\} \setminus \mathcal{I}_i^v,\ \mathcal{I}_i^v = C_i\,; \\ C_i \subseteq \mathcal{P}_N,\ C_i \cap C_j = \varnothing,\ \forall i \neq j \Big\}_{i=0}^{k-1} \tag{3.37}$$

where $\mathcal{P}_N$ is the set of random permutations of $\{1, \ldots, N\}$.

The out-of-sample error for this scheme is:

$$\begin{aligned}\hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}^{\text{CV}}, f\right) &= \frac{1}{k}\sum_{i=0}^{k-1}\frac{1}{\textbf{Card}\left(\mathcal{I}_i^v\right)}\sum_{(\mathbf{z},y)\in\mathcal{D}\left(\mathcal{I}_i^v\right)}\ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}\right)\right] \\ &= \frac{1}{k}\sum_{i=0}^{k-1}\frac{1}{\textbf{Card}\left(C_i\right)}\sum_{(\mathbf{z},y)\in\mathcal{D}\left(\mathcal{I}_i^v\right)}\ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}\right)\right] \\ &= \frac{1}{k}\sum_{i=0}^{k-1}\frac{1}{\delta(N,k,i)}\sum_{(\mathbf{z},y)\in\mathcal{D}\left(\mathcal{I}_i^v\right)}\ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}\right)\right] \end{aligned} \tag{3.38}$$

3.3.2 k-Fold Blocked Cross-Validation

Snijders [49] (in the text of [23, p. 8]) proposed that instead of shuffling the observations, the validation sets would be created by taking continuous sequences, or blocks, of observations (the `Holdout` would be a special case of this approach). Bergmeir and Benítez [6] extended this idea and developed the *k-fold blocked cross-validation scheme* as a simple modification of the k-fold CV scheme. By using the idea of continuous indexes, the order of the time series observations is kept within the blocks but broken across the folds. A schematic illustration of this scheme is shown in Figure 3.11.

The *blocked k-fold cross-validation scheme*, $\mathcal{V}^{\text{CV-Bl}}(\mathcal{D}; k)$ is:

$$\begin{aligned}\mathcal{V}^{\text{CV-Bl}}(\mathcal{D}; k) \equiv \Big\{ &(\mathcal{I}_i^e, \mathcal{I}_i^v) |\ \mathcal{I}_i^e = \{1, \ldots, N\} \setminus \mathcal{I}_i^v, \\ &\mathcal{I}_i^v = \{[i \cdot \delta(N,k,i)] + 1, \ldots, (i+1) \cdot \delta(N,k,i)\}\Big\}_{i=0}^{k-1}\end{aligned} \tag{3.39}$$

where $\delta(N,k,i)$ is the delta function defined in Equation (3.27) (the same conditions apply here).

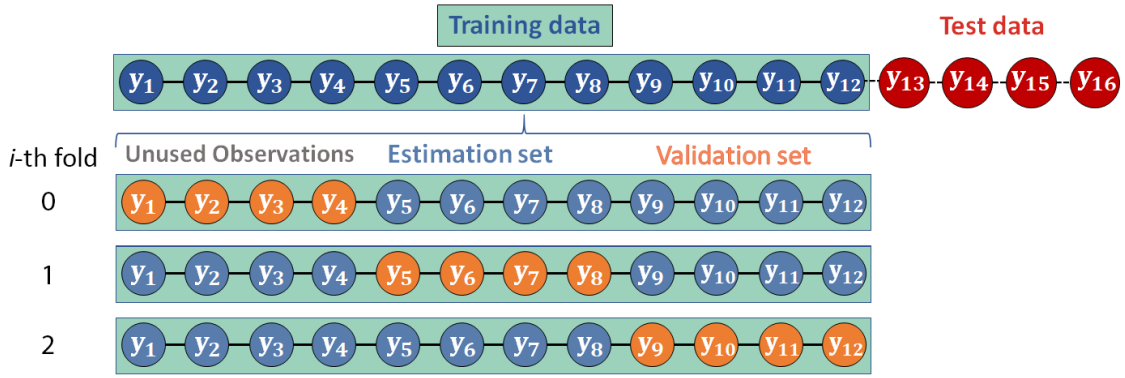


Figure 3.11: Example of the k-fold blocked cross-validation scheme applied to a univariate time series with $T = 16$, $q_t = 0.8$, $q_e = 0.8$ and $k = 3$.

The validation error, $\hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}^{\text{CV-Bl}}, f\right)$, for the *blocked k-fold cross-validation scheme* is:

$$\begin{aligned}\hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}^{\text{CV-Bl}}, f\right) &= \frac{1}{k}\sum_{i=0}^{k-1}\frac{1}{\textbf{Card}\left(\mathcal{I}_i^v\right)}\sum_{(\mathbf{z},y)\in\mathcal{D}\left(\mathcal{I}_i^v\right)}\ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}\right)\right] \\ &= \frac{1}{k}\sum_{i=0}^{k-1}\frac{1}{\delta(N,k,i)}\sum_{(\mathbf{z},y)\in\mathcal{D}\left(\mathcal{I}_i^v\right)}\ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}\right)\right] \end{aligned} \tag{3.40}$$

3.3.3 *h*-Block and *hv*-Block Cross-Validation

Burman et al. [53] developed a modification of the leave-one-out procedure for dependent data, called *h-block cross-validation*. Recall from our discussion of the leave-one-out scheme (Section 3.3.1), that the validation set is $\mathcal{I}_i^v = \{i+1\}$ and the estimation set is $\mathcal{I}_i^e = \{1, \ldots, N\} \setminus \mathcal{I}_i^v$, and we have $k = N$ folds.

In the h-block cross-validation procedure, the number of folds is still equal to N, and the validation set remains with only one observation. The difference lies in the defi-

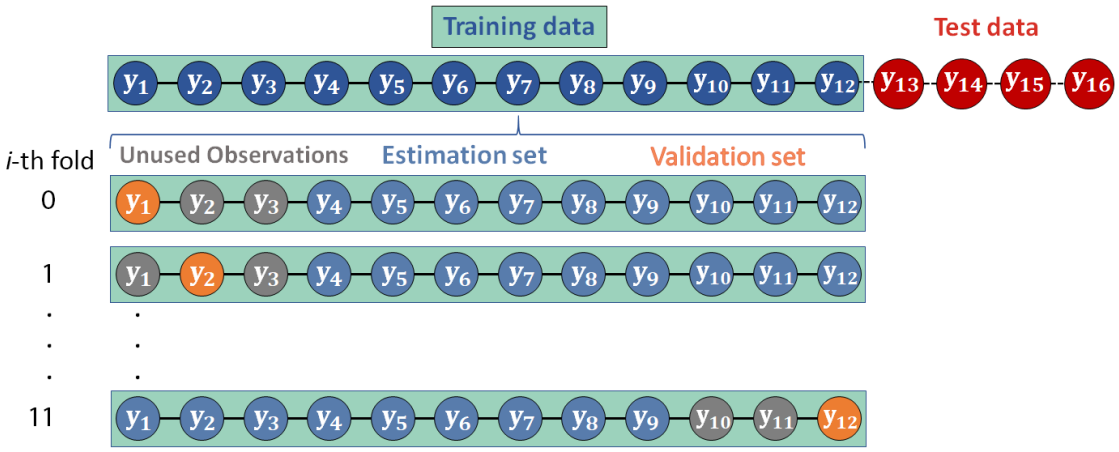


Figure 3.12: Example of a univariate time series with $T = 16$ partitioned accordingly to the h-block cross-validation scheme with $q_t = 0.8$, $q_e = 0.8$, $p = 3$, and $h = 1$.

nition of the estimation set. Now, aside from the deletion of the $i+1$-th index, $2h$ indexes are also deleted from the estimation set - h to the left of $i+1$, and h to its right[15]. Thus, it effectively creates "gaps" of size h around the observation $i+1$, and the estimation set uses only $N - 2h - 1$ observations to fit the model, instead of the $N - 1$ observations used in the leave-one-out approach (Figure 3.12). Burman et al. [53, p. 352] do this to achieve what they call a "near independence" setting between the estimation and validation sets (for large enough h)[16]. Burman et al. [53] proposed a rule-of-thumb for the h-block cross-validation scheme and suggested $h = \gamma N$, with $\gamma \in (0, 0.5)$[17].

Racine [55] proposed an improvement to the h-block procedure called the *hv-block cross-validation* scheme. In this scheme, Racine deletes h *and* v observations from either side of the $i+1$-th index, but instead of completely disregarding the v cases, the author

[15] If $i = 0$, it takes h from the right side only. Conversely, it takes only from the left if $i = k - 1$.
[16] In this context, near independence means that $E(\varepsilon_i - {}_h\varepsilon_j | Z_1, \ldots, Z_j) \approx 0$, for $i < j$ [55, p. 47-48].
[17] See Burman et al. [53, p. 352-354] for a discussion about selecting the proper size of h

adds them to the validation set. Hence, the sizes of the validation and estimation[18] sets become:

$$l_v^{\text{OG-hvCV}} = 2v + 1 \tag{3.41}$$

$$N_e^{\text{OG-hvCV}} = N - 2h - 2v - 1 \tag{3.42}$$

Racine [55, p. 49] used Burman et al. [53]'s rule-of-thumb for h (i.e., $h = \gamma N$), and report obtaining sensible results in the hv-block cross-validation procedure when $\gamma = 0.25$. For positive degrees of freedom $\left(N_e^{\text{OG-hvCV}} - p\right) > 0$[19], Racine [55, p. 46-47] suggests that the size of the estimation set should be equal to $\lfloor N^\delta \rfloor$, where δ is such that the ratio $\log(p)/\log(N) < \delta < 1$. In this case, $v = (N - N^\delta - 2h - 1)/2$.

The last missing value is the number of folds, k, that one can use. To find it, it helps to write all the cases for $\mathcal{I}_i^v$, and to define a sets D_i, $i = 0, \ldots, k-1$, of indexes from $\{1, \ldots, N\}$ that will be deleted from the estimation set. Since we first remove v from either side and then h from either side, we end up with five different cases.

For $i = 1, \ldots, v$:

$$\mathcal{I}_i^v = \{1, \ldots, 2v + 1\}$$

$$D_i = \{1, \ldots, 2v + 2h + 1\}$$

[18] Racine [55] used n_v for $l_v^{\text{OG-hvCV}}$, and n_c instead of $N_e^{\text{OG-hvCV}}$.

[19] Recall that p is the dimension of $\mathbf{Z}_{t-1}$, the vector of past explanatory variables, as defined in Section 3.

For $i = v + 1, \ldots, v + h$:

$$\mathcal{I}_i^v = \{i - v, \ldots, i + v\}$$

$$D_i = \{1, \ldots, 2v + 2h + 1\}$$

For $i = v + h + 1, \ldots, N - v - h$:

$$\mathcal{I}_i^v = \{i - v, \ldots, i + v\}$$

$$D_i = \{i - v - h, \ldots, i + v + h\}$$

For $i = N - v - h + 1, \ldots, N - v$:

$$\mathcal{I}_i^v = \{i - v, \ldots, i + v\}$$

$$D_i = \{N - 2v - 2h, \ldots, N\}$$

For $i = N - v + 1, \ldots, N$:

$$\mathcal{I}_i^v = \{N - 2v, \ldots, N\}$$

$$D_i = \{N - 2v - 2h, \ldots, N\}$$

Or, simply:

$$\mathcal{I}_i^v = \begin{cases} \{1, \ldots, 2v+1\}, & \text{for } i = \quad 1, \ldots, v; \\ \{i-v, \ldots, i+v\}, & \text{for } i = \quad v+1, \ldots, v+h, \\ & \qquad\quad v+h+1, \ldots, N-v-h, \\ & \qquad\quad N-v-h+1, \ldots, N-v; \\ \{N-2v, \ldots, N\}, & \text{for } i = \quad N-v+1, \ldots, N. \end{cases} \tag{3.43}$$

and

$$D_i = \begin{cases} \{1, \ldots, 2v+2h+1\}, & \text{for } i = 1, \ldots, v; \\ \{1, \ldots, 2v+2h+1\}, & \text{for } i = v+1, \ldots, v+h; \\ \{i-v-h, \ldots, i+v+h\}, & \text{for } i = v+h+1, \ldots, N-v-h; \\ \{N-2v-2h, \ldots, N\}, & \text{for } i = N-v-h+1, \ldots, N-v; \\ \{N-2v-2h, \ldots, N\}, & \text{for } i = N-v-1, \ldots, N. \end{cases} \tag{3.44}$$

It is easy to see that the first and last cases are redundant, so we could remove those. If we remove them, i goes from $v+1$ to $N-v$, when we consider the remaining cases.

Thus, the maximum number of folds is:

$$\begin{aligned} k &= N - v - (v + 1) + 1 \\ &= N - 2v \end{aligned} \tag{3.45}$$

Schnaubelt [23] confirms our calculations. This author wrote that in the hv-block cross-validation scheme, "the validation block is rolled forward by one observation, such that a total of $k = N - 2v$ validation folds is considered" [23, p. 9]. However, it is worth highlighting that in this case, i starts at $v + 1$, not 1 (nor 0), a remark that Schnaubelt did not make. An illustration of this scheme is given in Figure 3.13.

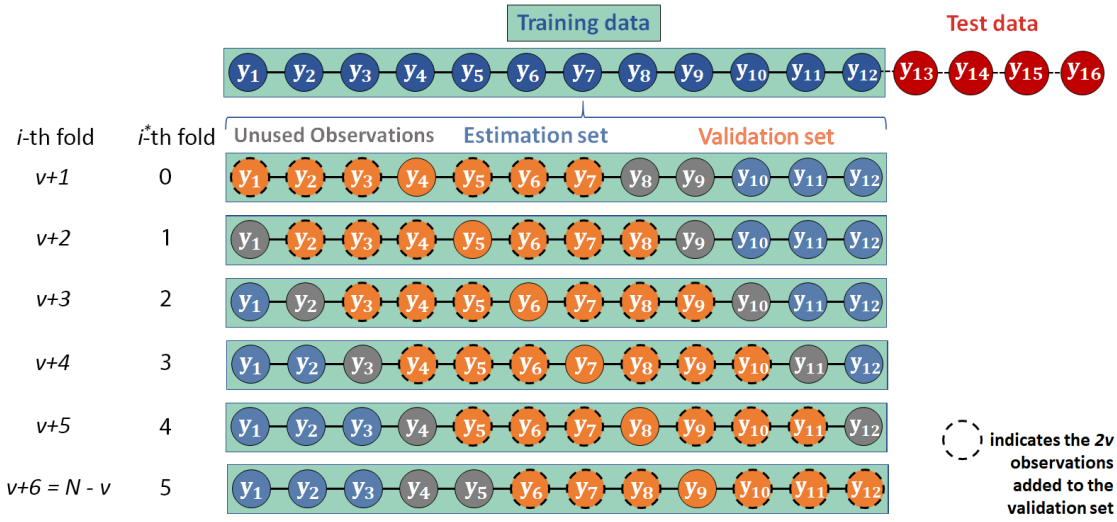


Figure 3.13: Example of the *hv*-block cross-validation scheme by Rancine (2000), applied to a univariate time series with $T = 16$, $q_t = 0.8$, $q_e = 0.8$, $p = 3$, $h = 1$, $\delta = 0.5$, and $k = 6$.

Based on the discussion above, $\mathcal{V}^{\text{OG-hvBl}}(\mathcal{D}; k)$, the original *hv-block cross-validation*

scheme is defined as:

$$\mathcal{V}^{\text{OG-hvBl}}(\mathcal{D}; v, h) \equiv \Bigg\{ (\mathcal{I}_i^e, \mathcal{I}_i^v) |\ \mathcal{I}_i^e = \{1, \ldots, N\} \setminus D_i,$$

$$\mathcal{I}_i^v = \begin{cases} \{1, \ldots, 2v+1\}, & \text{for } i = 1, \ldots, v; \\ \{i - v, \ldots, i + v\}, & \text{for } i = v+1, \ldots, N - v; \\ \{N - 2v, \ldots, N\}, & \text{for } i = N - v + 1, \ldots, N. \end{cases} \Bigg\}_{i=0}^{k-1} \tag{3.46}$$

where N is the length of $\mathcal{D}$, and D_i is given by Equation (3.44).

We called the validation scheme $\mathcal{V}^{\text{OG-hvBl}}$, where "OG" stands for "original." We did this to distinguish it from the way Cerqueira et al. [22] coded the *hv-block cross-validation* scheme. The way they programmed it uses different values for k, v, and h, and it also yields different sets $\mathcal{I}_i^e$ and $\mathcal{I}_i^v$. But before going into the details of those differences, we would like to stress that these authors did not include any of the validation equations in their paper, and our remark is the conclusion we reached after tracing their program. Any errors and omissions regarding the interpretation of their code and its subsequent translation into the equations seen below (and throughout Sections 3.2 and 3.3) are our own.

Cerqueira et al. [22] set a fixed value for k and used $v = (N - k)/(2k)$ and $h = p + 1$, with p being estimated using Takens [75]'s method (more details in Section 5.2.2). Since we want to compare our results to theirs, we shall use Cerqueira's, not Rancine's, approach.

With those choices of k and v, the number of observations in the validation set is:

$$\begin{aligned} l_v^{hvCV} &\equiv 2 \cdot v + 1 \\ &= 2\frac{N-k}{2k} + 1 \\ &= \frac{N}{k} \end{aligned} \tag{3.47}$$

The above equation depends on the ratio between N and k. As discussed before (Section 3.2.5), this result will depend on the software or function that one uses. Hence, we define the "nu" function:

$$\nu(N,k,i) = \begin{cases} \left\lfloor l_v^{hvCV} \right\rfloor, \text{ if the computer returns } \left\lfloor \frac{N}{k} \right\rfloor \text{ for the } i\text{-th case} \\ \\ \left\lceil l_v^{hvCV} \right\rceil, \text{ otherwise.} \end{cases} \tag{3.48}$$

Then, for $i = 0, \ldots, k-1$, the sets of indexes from $\{1, \ldots, N\}$ that will form the validation sets are given by:

$$\mathcal{I}_i^{v*} \equiv \begin{cases} \left\{1, \ldots, \nu(N,k,0)\right\}, & \text{if } i = 0 \\ \\ \left\{\left[\sum_{j=0}^{i-1} \nu(N,k,j)\right] + 1, \ldots, \sum_{j=0}^{i} \nu(N,k,j)\right\}, & \text{if } 0 < i \leq k-1 \end{cases} \tag{3.49}$$

Similarly, the indexes that will be deleted from the estimation set are:

$$D_i^* \equiv \begin{cases} \left\{1, \ldots, \nu(N,k,0)\right\}, & \text{if } i = 0 \\ \left\{\left[\sum_{j=0}^{i-1} \nu(N,k,j)\right] + 1 - h, \ldots, \left[\sum_{j=0}^{i} \nu(N,k,j)\right] + h\right\}, & \text{if } 0 < i < k-1 \\ \left\{\left[\sum_{j=0}^{i-1} \nu(N,k,j)\right] + 1 - h, \ldots, \sum_{j=0}^{i} \nu(N,k,j)\right\}, & \text{if } i = k-1 \end{cases} \tag{3.50}$$

The hv-block validation scheme in Cerqueira et al. [22]'s case is defined as:

$$\mathcal{V}^{\text{CV-hvBl}}(\mathcal{D}; k, h) \equiv \left\{ (\mathcal{I}_i^e, \mathcal{I}_i^v) |\ \mathcal{I}_i^e \approx \{1, \ldots, N\} \setminus D_i^*, \mathcal{I}_i^v = \mathcal{I}_i^{v*} \right\}_{i=0}^{k-1} \tag{3.51}$$

where N is the number of observations in $\mathcal{D}$, $h = p + 1$, and D_i^* and $\mathcal{I}_i^{v*}$ are given by the equations Equation (3.50) and Equation (3.49). An example of this scheme applied to a time series can be seen in Figure 3.14.

Finally, the validation error used by Cerqueira et al. [22] for the *hv-block cross-validation scheme* is:

$$\begin{aligned} \hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}^{\text{CV-hvBl}}, f\right) &= \frac{1}{k} \sum_{i=0}^{k-1} \frac{1}{\textbf{Card}\left(\mathcal{I}_i^v\right)} \sum_{(\mathbf{z},y) \in \mathcal{D}\left(\mathcal{I}_i^v\right)} \ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}\right)\right] \\ &= \frac{1}{k} \sum_{i=0}^{k-1} \frac{1}{\nu(N,k,i)} \sum_{(\mathbf{z},y) \in \mathcal{D}\left(\mathcal{I}_i^v\right)} \ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}\right)\right] \end{aligned} \tag{3.52}$$

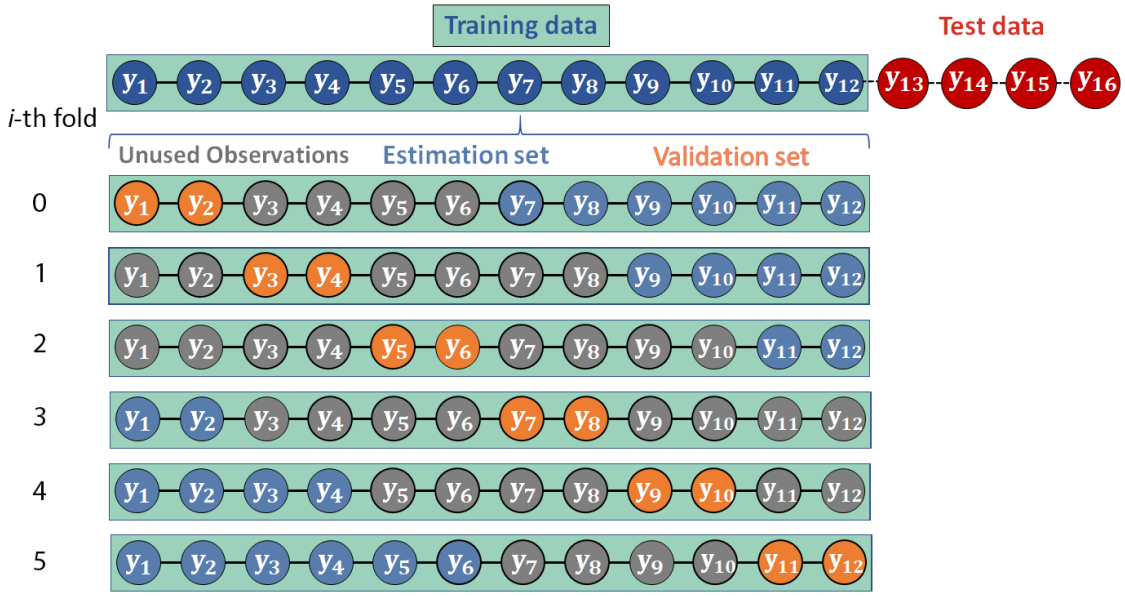


Figure 3.14: Example of the *hv*-block cross-validation scheme used by Cerqueira et al. (2020), applied to a univariate time series with $T = 16$, $q_t = 0.8$, $q_e = 0.8$, $p = 3$, and $k = 6$.

3.3.4 Modified cross-validation

McQuarrie and Tsai [81] showed that it is possible to obtain a better cross-validation criterion by modifying the k-fold procedure and deleting d observations from the estimation set at each fold. These authors called this scheme the *delete-d cross-validation*, CV(d), while Bergmeir and Benítez [6] called it *non-dependent cross-validation* and [22] used the term *modified cross-validation*.

McQuarrie and Tsai [81, p. 254] address the problem of choosing d, and propose $d = \lceil N - N^{\alpha} \rceil$ with $0 < \alpha < 1$. However, Kastens [82, p. 389] argues that this choice is not sufficient to indicate a desirable value for d, searches for an optimal value for it (in

the sense that, with it, CV(d) exhibits the highest rate of selecting the best model[20]), and discusses the use of $d = N - p - 1$ and $d = N - p$.

In their code, Cerqueira et al. [22] use a different approach and delete at most $d \leq \left[\left(2 \cdot \sqrt{\lfloor p+1 \rfloor}\right) + 1\right] \cdot N/k$[21] observations. In the way they programmed it, there can be overlaps in the indexes. Thus, the final value of d depends both on the number of unique indexes and on the number of indexes greater than 0.

Let us use the same framework as we did in Section 3.3.1 for the k-fold procedure. That is, let $C_0, \ldots, C_{k-1}$ be the sets formed from random partitions of $\{1, \ldots, N\}$ taken without replacement, such that each set has approximately N/k elements (i.e., $\mathbf{Card}\,(C_i) = \delta(N, k, i)$, $i = 0, \ldots, k-1$, where $\delta(N, k, i)$ is the delta function defined in Equation 3.27). Let $c_i^{(j)}$ be the indexes of the $j = 1, \ldots, \delta(N, k, i)$ elements in C_i. For example, back in the section on the k-fold, in the example shown in Figure 3.10, we have that $C_0 = \{5, 10, 7, 4\}$. Then, $c_0^{(1)} = 5$, $c_0^{(2)} = 10$, $c_0^{(3)} = 7$, $c_0^{(4)} = 4$.

In the *modified cross-validation scheme* used by CTM, at each $c_i^{(j)}$, we delete the observations in the range

$$E_i^{(j)} \equiv \left\{c_i^{(j)} - \lfloor\sqrt{p+1}\rfloor - 1, \ldots, c_i^{(j)} + \lfloor\sqrt{p+1}\rfloor - 1\right\} \tag{3.53}$$

It is easy to see that every set $E_i^{(j)}$, for $j = 1, \ldots, \delta(N, k, i)$, has cardinality equal to $\left(2 \cdot \sqrt{\lfloor p+1 \rfloor}\right) + 1$. And since we have $\delta(N, k, i) \approx N/k$ sets, the number of deleted observations at each fold i is (approximately) equal to $\left[\left(2 \cdot \sqrt{\lfloor p+1 \rfloor}\right) + 1\right] \cdot N/k$, as

[20] That is, the model with all of the predictors that contribute to the response, and only those.

[21] These values were obtained after tracing Cerqueira et al. [22]'s code. Thus, any errors or omissions are our own.

we have stated above. Moreover, since this value changes at every fold (because of the different results for the ratio N/k), we prefer writing this number as

$$d_i \leq d_{\text{max}} \equiv \left[\left(2 \cdot \sqrt{\lfloor p+1 \rfloor}\right) + 1\right] \cdot \delta(N,k,i) \tag{3.54}$$

The maximum value of d_i, d_{max}, assumes that $E_i^{(j)} \cap E_i^{(b)} = \varnothing$, for $j \neq b$ at each i, and that every element in $E_i^{(j)}$ is greater than zero. We did this to facilitate the definition of d_i and of the sets. In reality, the actual value of d_i might be smaller than that since we take only the unique elements of $E_i^{(j)}$ that are greater than 0.

For instance, let us revisit our example. Recall that we have been using $N = 12$, $k = 3$, and that C_0 has the elements $c_0^{(1)} = 5$, $c_0^{(2)} = 10$, $c_0^{(3)} = 7$, $c_0^{(4)} = 4$. Then, for $p = 2$, the $E_0^{(j)}$s are:

$$\begin{aligned} E_0^{(1)} &= \left\{c_0^{(1)} - \lfloor\sqrt{2+1}\rfloor - 1, \ldots, c_0^{(1)} + \lfloor\sqrt{2+1}\rfloor - 1\right\} \\ &= \left\{5 - 1 - 1, \ldots, 5 + 1 - 1\right\} \\ &= \left\{3, 4, 5\right\} \end{aligned}$$

$$\begin{aligned} E_0^{(2)} &= \left\{10 - 1 - 1, \ldots, 10 + 1 - 1\right\} \\ &= \left\{8, 9, 10\right\} \end{aligned}$$

$$\begin{aligned} E_0^{(3)} &= \left\{7 - 1 - 1, \ldots, 7 + 1 - 1\right\} \\ &= \left\{5, 6, 7\right\} \end{aligned}$$

and

$$E_0^{(4)} = \{4-1-1, \ldots, 4+1-1\}$$
$$= \{2, 3, 4\}$$

Thus, the indexes of the observations that will be deleted from the training set to form the estimation set are:

$$\{3, 4, 5\} \cup \{8, 9, 10\} \cup \{5, 6, 7\} \cup \{2, 3, 4\}$$

and it is clear to see that this number matches the value obtained from Equation (3.54),

$$d_0 \leq d_{\text{max}} = \left[\left(2 \cdot \sqrt{\lfloor 2+1 \rfloor}\right) + 1\right] \cdot 12/3 = 12$$

We would like to take only the unique indexes from the union above, and all the indexes that are greater than zero. Thus, let us also define a set that represents this union, but without the undesired indexes:

$$E_i \equiv \left\{ (e_1, ..., e_{d_i}) \in \bigcup_{j=1}^{\delta(N,k,i)} E_i^{(j)} \mid \right.$$
$$\left. \forall a, b \in \{1, \ldots, d_{\text{max}}\}, e_a > 0, e_b > 0, \text{ and } a \neq b \implies e_a \neq e_b \right\} \quad (3.55)$$

In our example,

$$E_0 = \{3, 4, 5, 8, 9, 10, 6, 7, 2\}$$

and ordering it yields

$$E_0 = \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

The above example - with E_1 and E_2 obtained in the same way as E_0 - is illustrated in Figure 3.15. This Figure was created exactly like Figure 3.10 for the k-fold scheme. The only difference is that the deleted observations were grayed out here.

From Equation (3.55) and Equation (3.53), it is easy to see that for every $i = 0, \ldots, k-1$, the sets $C_i \subset E_i$[22]. Then, we can modify the definition of the k-fold cross-validation scheme from Equation (3.37) to account for the deleted observations and write the *modified cross-validation scheme* used by Cerqueira et al. [22, 69], $\mathcal{V}^{\text{CV-Mod}}(\mathcal{D}; k)$, as:

$$\mathcal{V}^{\text{CV-Mod}}(\mathcal{D}; k) \equiv \Bigg\{ (\mathcal{I}_i^e, \mathcal{I}_i^v) |\ \mathcal{I}_i^e = \{1, \ldots, N\} \setminus E_i,\ \mathcal{I}_i^v = C_i\,; \\ C_i \subseteq \mathcal{P}_N,\ C_i \cap C_j = \varnothing,\ \forall i \neq j \Bigg\}_{i=0}^{k-1} \tag{3.56}$$

where $\mathcal{P}_N$ is the set of random permutations of $\{1, \ldots, N\}$, and N is the number of observations in the training set $\mathcal{D}$.

[22] A set C_i might not be a proper subset of E_i, since it is possible to have $C_i = E_i$. For instance, suppose that in the example given ($N = 12$, $k = 3$, $p = 2$), we obtain the set $C_0 = \{1, 2, 3, 4\}$. Then,

$$E_0^{(1)} = \{1 - 1 - 1, \ldots, 1 + 1 - 1\} = \{-1, 0, 1\}$$
$$E_0^{(2)} = \{2 - 1 - 1, \ldots, 2 + 1 - 1\} = \{0, 1, 2\}$$
$$E_0^{(3)} = \{3 - 1 - 1, \ldots, 3 + 1 - 1\} = \{1, 2, 3\}$$
$$E_0^{(4)} = \{4 - 1 - 1, \ldots, 4 + 1 - 1\} = \{2, 3, 4\}$$

Thus,

$$E_0 = \{1, 2, 3, 4\} = C_0$$

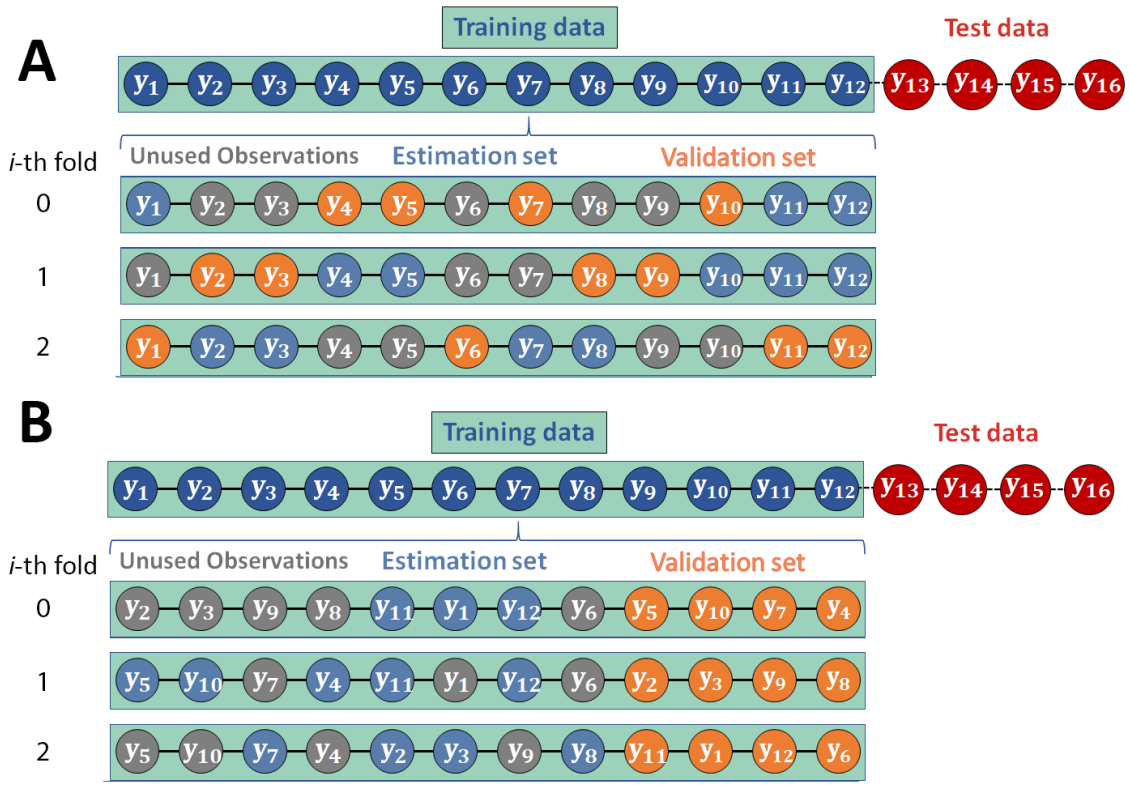


Figure 3.15: Example of the modified cross-validation scheme used by Cerqueira et al. (2020), applied to a univariate data set with $T = 16$, $q_t = 0.8$, $q_e = 0.8$, $p = 2$, and $k = 3$. Panel A shows the selected observations in their original ordering. Panel B shows them according to their selection order.

The out-of-sample error for this scheme is similar to the one for the k-fold. The difference is in the definition of $\mathcal{D}\left(\mathcal{I}_i^e\right)$ (which impacts the estimates of $\hat{\boldsymbol{\theta}}$), which now has fewer observations than the estimation set used in the k-fold procedure. We write this error as:

$$\begin{aligned}
\hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}^{\text{CV-Mod}}, f\right) &= \frac{1}{k}\sum_{i=0}^{k-1}\frac{1}{\mathbf{Card}\left(\mathcal{I}_i^v\right)}\sum_{(\mathbf{z},y)\in\mathcal{D}\left(\mathcal{I}_i^v\right)}\ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}\right)\right] \\
&= \frac{1}{k}\sum_{i=0}^{k-1}\frac{1}{\mathbf{Card}\left(C_i\right)}\sum_{(\mathbf{z},y)\in\mathcal{D}\left(\mathcal{I}_i^v\right)}\ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}\right)\right] \\
&= \frac{1}{k}\sum_{i=0}^{k-1}\frac{1}{\delta(N,k,i)}\sum_{(\mathbf{z},y)\in\mathcal{D}\left(\mathcal{I}_i^v\right)}\ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}\right)\right] \qquad (3.57)
\end{aligned}$$

Chapter 4: Introducing the *p-Holdout* family of schemes

After learning about the history of the validation schemes in Chapter 2 and going over their methodological details in Chapter 3, we noticed that none consider the seasonal aspect that a time series may present[1]. It is true that in the statistical literature some practitioners deseasonalize the data prior to producing the forecasts, following Kirby's suggestion [36, p. B-208], which generally leads to improvements in forecast accuracy. Yet, the "inconsistent handling of seasonality" [83] in machine learning applications (especially those based on neural networks) leads to mixed conclusions about whether deseasonalizing the series improves its forecasts [83, 84, 85]. On the other hand, when seasonality is properly accounted for, significant improvements can be seen in forecast accuracy [84, 86], and it also reduces the "computational time required to arrive at optimal weights and, therefore, learn faster" [85, p. 21].

Notwithstanding, recent papers in machine learning usually do not take into account a series' seasonal pattern. To arrive at this conclusion, we searched for the terms "machine learning time series" on Google Scholar and evaluated the top 20 papers in the number of citations - all published between 2001 and 2019. Out of the 20, eleven of them have no mention of the word "season" at all, one uses this word outside of a time series context,

[1] It is really difficult to be entirely accurate when affirming this since so many different names for validation schemes have been used since the 1930s, but to our knowledge, no method specifically accounts for seasonal or cyclic data.

and three only mention it in their literature review[2]. Moreover, a search on Google Scholar for the terms "machine learning time series deseasonalized" or "machine learning time series deseasonalize" yields fewer than 1,300 papers (1,240 in the former, and 1,230 in the latter case). Therefore, machine learning procedures are being used to select forecasting models, but without proper consideration of a series' period, a crucial characteristic of a time series.

Since a time series might contain cycles and seasonal patterns, disregarding them when preprocessing the data or when dividing between training data and test data might lead to an incorrect choice of the model. And while we are aware that one needs to be careful with over-fitting the model to the estimation and validation sets at hand, as the latter can be very different from the test set, we argue that *because* the observations are dependent and *because* a seasonal series displays a similar behavior over time, it is possible to obtain similar validation and test sets (Figure 4.1 - Panels D and E) in a way that improves forecast accuracy and, consequently, model selection.

Based on that, we propose three validation schemes that account for the periodicity of a time series. Figure 4.1 - Panels B, C, D, and E - returns the results from splitting the *USAccDeaths* data set using the `Holdout` , the `p-Holdout` , the `cp-Holdout` schemes, and the `cep-Holdout` , respectively. Another example can be seen in Figure 1.2 in the Introduction. From these pictures, we observe that the validation sets created by the `p-Holdout` and the `cp-Holdout` schemes are more similar to the test set than the one obtained from the classic `Holdout` method. In such cases, we hypothesize that the differences between the out-of-sample errors obtained with the validation and test sets

[2]The list of papers is available in Appendix E.

should be small. In fact, the results shown in Chapter 6 show that this is often the case.

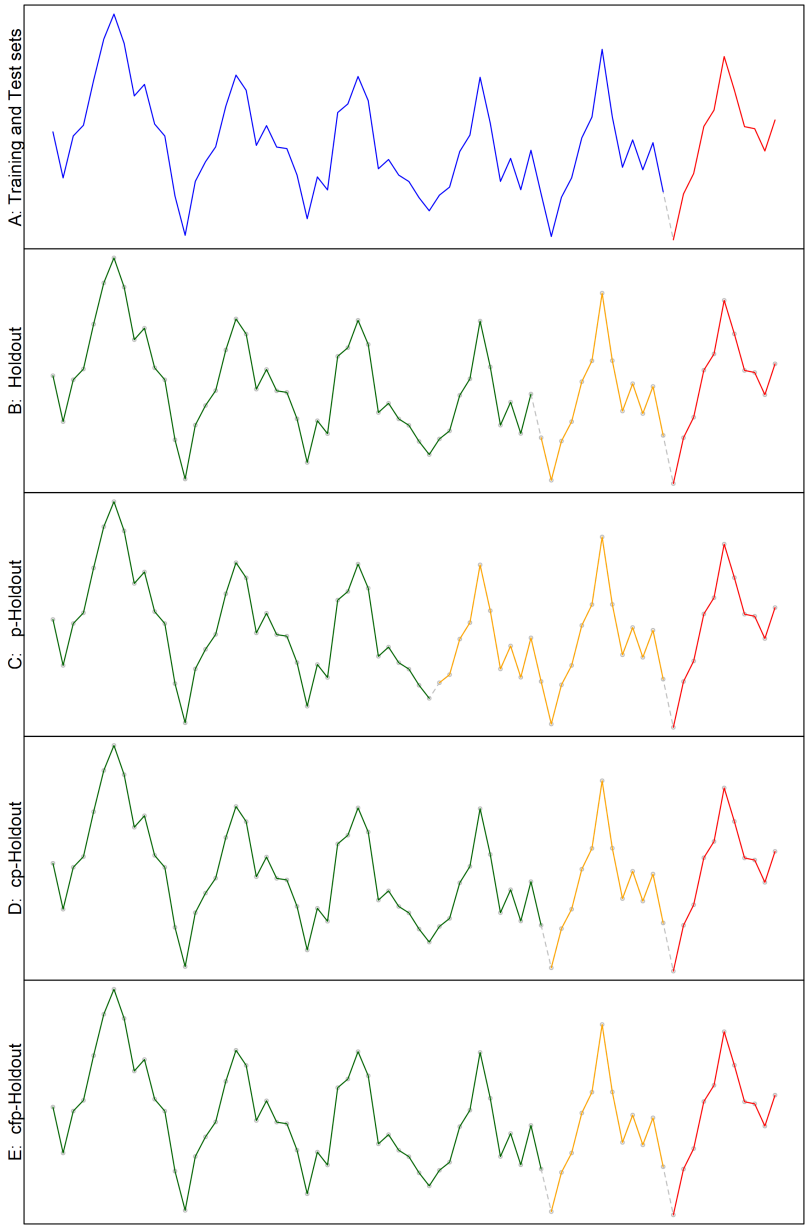


Figure 4.1: The `Holdout` , `p-Holdout` , `cp-Holdout` , and `cep-Holdout` schemes applied to the USAccDeaths data set. Panel A shows the entire series (blue) and the observations we would like to forecast (red). In Panel B, the test set was obtained with the `Holdout` method. In Panels C, D, and E, the test set was obtained applying the `p-Holdout` , `cp-Holdout` , and `cep-Holdout` schemes, respectively.

We compare our family of schemes with the procedures used by Bergmeir and Benítez [6], Bergmeir et al. [68], Cerqueira et al. [69], Bergmeir et al. [21], [23], and CTM. These papers use techniques to compare several validation schemes' performance and provide benchmark data sets that can be used to evaluate a new procedure. From the results in Chapter 6, our new schemes are computationally inexpensive, improve the forecast accuracy, and greatly reduce the average forecast bias without increasing the variability, especially when applied to non-stationary time series. Their details are given in the following sections.

4.1 The *p*-Holdout validation scheme

Our first proposal is the simple *period-holdout forward-validation scheme* or simply the `p-Holdout` scheme. It works much like the `Holdout` procedure in the sense that we take the last block of the training data to be our validation set. However, there are some key differences.

For the cases where a series exhibits a periodic behavior, the length of the validation set is not determined by a split-point chosen by the analyst, as it is done for the `Holdout` scheme[3], but depends directly on the length of the forecasting horizon, l. Hence, when similarities occur after s basic time intervals, the final length of the validation set is defined by the summation of l and s. If the series seems aperiodic, then we revert to the `Holdout` scheme. Formally, we define l_v^{pHO}, the length of the validation set under the

[3]Recall that in the `Holdout` procedure, $l_v = \lceil (1 - q_e) \cdot N \rceil$, for $q_e \in [1/N, 1)$.

p-Holdout scheme as,

$$l_v^{\text{pHO}} \equiv \begin{cases} \lceil (1 - q_e) \cdot N \rceil, & \text{if } s = 1 \\ l + s, & \text{otherwise.} \end{cases} \tag{4.1}$$

Recall that the length of the validation set under the Holdout scheme is $l_v^{\text{HO}} = \lceil (1 - q_e) \cdot N \rceil$ (see Equation (3.11)). Substituting this in the equation above yields,

$$l_v^{\text{pHO}} \equiv \begin{cases} l_v^{\text{HO}}, & \text{if } s = 1 \\ l + s, & \text{otherwise.} \end{cases} \tag{4.2}$$

In our analysis, the period s is obtained using the frequency function from base R. If the time series object has information on periodicity, this function returns the number of observations before the seasonal pattern repeats[4]. And even though the length of the validation set does not depend on the choice of q_e, the length of the test set might depend on the choice of q_t, as shown in Equation (3.3).

A schematic illustration of the p-Holdout scheme is given in Figure 4.2. For the sake of the example shown in this picture, take a time series of daily data with length $T = 16$. Suppose that this series has a pattern that repeats every 3 days. Thus, $s = 3$. If we want to forecast the next four observations, then we use Equation (3.3) to write $N = 16 - 4 = 12$, and Equation (4.2) to write $l_v^{pHO} = 4 + 3 = 7$.

We have developed the p-Holdout validation scheme to account for time series

[4]We developed the *cep*-Holdout (see Section 4.3) to account for the cases where the object does not have that information.

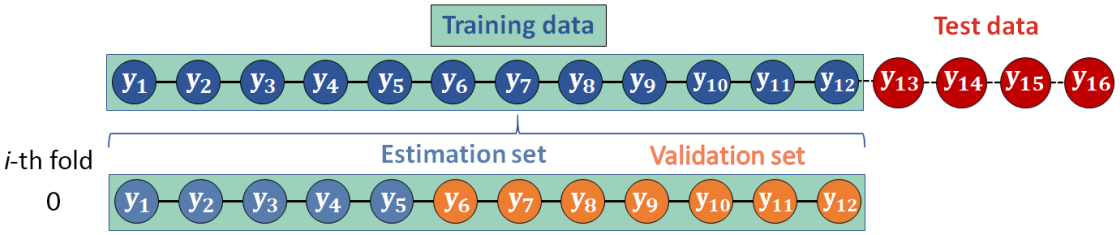


Figure 4.2: Example of the *p-Holdout* forward-validation scheme applied to a univariate time series with $T = 16$, $q_t = 0.8$, and $s = 3$.

with a period greater than 1. For those cases, $\mathcal{V}^{\text{pHO}}(\mathcal{D}; l, s)$, the `p-Holdout` validation scheme, is defined as:

$$\mathcal{V}^{\text{pHO}}(\mathcal{D}; l, s) \equiv \Big\{ (\mathcal{I}^e, \mathcal{I}^v) |\ \mathcal{I}^e = \{1, \ldots, (N - l - s)\}, \\ \mathcal{I}^v = \{(N - l - s + 1), \ldots, N\} \Big\} \quad (4.3)$$

Another way to write Equation (4.3) is to substitute $l + s$ for l_v^{pHO} from Equation (4.2). This makes it comparable to the `Holdout` procedure's equation, given by E.(3.12). Then, the proposed validation scheme $\mathcal{V}^{\text{pHO}}(\mathcal{D}; l, s)$ becomes:

$$\mathcal{V}^{\text{pHO}}(\mathcal{D}; l, s) = \Big\{ (\mathcal{I}^e, \mathcal{I}^v) |\ \mathcal{I}^e = \{1, \ldots, (N - l_v^{pHO})\}, \\ \mathcal{I}^v = \{(N - l_v^{pHO} + 1), \ldots, N\} \Big\} \quad (4.4)$$

The estimated out-of-sample average error (Eq.3.7) is written as:

$$\hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}^{\text{pHO}}, f\right) \equiv \frac{1}{l_v^{pHO}} \sum_{(\mathbf{z},y) \in \mathcal{D}(\mathcal{I}^v)} \ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}(\mathcal{I}^e)}\right)\right] \quad (4.5)$$

From Equations 4.3 and 4.5, it is easy to see that the `p-Holdout` scheme is **not** computationally expensive since it does not depend on several "rounds" of estimation, but on a single data split.

4.2 The *cp*-Holdout scheme

The `p-Holdout` procedure can "waste" many observations, as seen in Figure 4.1 - Panel B. To circumvent that, we developed the `cp-Holdout` procedure or *composite p-Holdout forward-validation scheme*. We called it *composite* because it considers three situations. If the series is aperiodic, then it returns the same validation set as the `Holdout` scheme. In the cases where the period is greater than 1, then we calculate the ratio between the length of the test set and s. If this division has a remainder, we take the ceiling of this ratio and multiply it by the period of the series. The result is the length of the validation set. Otherwise, it creates the validation set as the `p-Holdout` scheme does. In other words,

$$l_v^{\text{cpHO}} \equiv \begin{cases} l_v^{HO}, & \text{if } s = 1 \\ s \cdot \left\lceil l/s \right\rceil, & \text{if } s > 1 \text{ and } l \bmod s \neq 0 \\ l + s, & \text{otherwise.} \end{cases} \tag{4.6}$$

Using the same time series and conditions like those in the example from the `p-Holdout` scheme, we can show a simplified vision of the `cp-Holdout` method in Figure 4.3. In this case, we have that $l/s = 4/3 = 1.3333$, so l_v^{cpHO} is $3 \cdot \lceil 1.3333 \rceil = 6$.

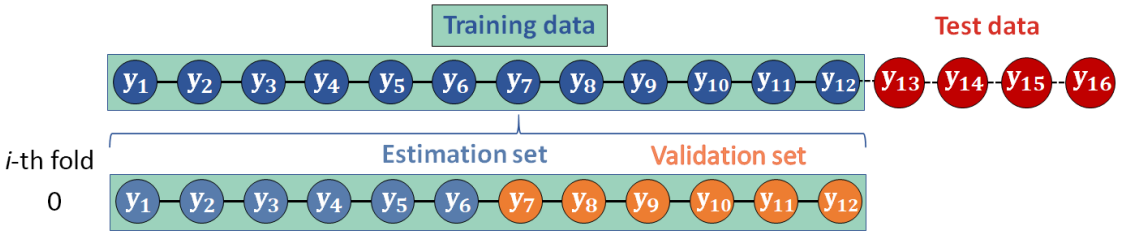


Figure 4.3: Example of the *cp-Holdout* forward-validation scheme applied to a univariate time series with $T = 16$, $q_t = 0.8$, and $s = 3$.

We define the `cp-Holdout` validation scheme, $\mathcal{V}^{\text{cpHO}}(\mathcal{D}; l, s)$ as:

$$\mathcal{V}^{\text{cpHO}}(\mathcal{D}; l, s) = \Big\{ (\mathcal{I}^e, \mathcal{I}^v) |\ \mathcal{I}^e = \left\{1, \ldots, \left(N - l_v^{cpHO}\right)\right\}, \\ \mathcal{I}^v = \left\{\left(N - l_v^{cpHO} + 1\right), \ldots, N\right\}\Big\} \quad (4.7)$$

And the equation for the average expected out-of-sample error is:

$$\hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}^{\text{cpHO}}, f\right) \equiv \frac{1}{l_v^{\text{cpHO}}} \sum_{(\mathbf{z},y)\in\mathcal{D}(\mathcal{I}^v)} \ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}(\mathcal{I}^e)}\right)\right] \quad (4.8)$$

4.3 The *cep*-Holdout scheme

In both the `p-Holdout` and `cp-Holdout` procedures, the period is obtained using the `frequency` function from base `R`. This function returns the number of observations s before the seasonal pattern repeats (its "frequency") by capturing an attribute of the time-series object used in the analysis. However, in some situations, the object might be of a different class (for instance, a "data.frame") that does not contain the "frequency"

attribute or might seem aperiodic (i.e., with the "frequency" attribute equal to one), but, in reality, it contains a cyclical or seasonal pattern. We developed the `cep-Holdout` scheme or *composite estimated period-holdout forward validation scheme* to account for these cases.

The `cep-Holdout` procedure works similarly to the `cp-Holdout` but uses the dominant frequency of a time series estimated from a spectral analysis of the data. We use the `findfrequency` function from the `forecast` package to obtain this dominant frequency since this function returns the seasonal period for seasonal data and the average cycle length for cyclic data. From its manual:

> "The dominant frequency is determined from a spectral analysis of the time series. First, a linear trend is removed, then the spectral density function is estimated from the best fitting autoregressive model (based on the AIC). If there is a large (possibly local) maximum in the spectral density function at frequency f, then the function will return the period 1/f (rounded to the nearest integer). If no such dominant frequency can be found, the function will return 1."

To avoid confusion with the forecasting model f, we shall write the dominant frequency as $\hat{s}$. With this, we write our first attempt at defining the length of the validation set for the `cep-Holdout` method as:

$$l_v^{1^{st}\text{cepHO}} \equiv \begin{cases} l_v^{\text{HO}}, & \text{if } \hat{s} = 1 \\ \hat{s} \cdot \left\lceil l/\hat{s} \right\rceil, & \text{if } \hat{s} > 1 \text{ and } l \bmod \hat{s} \neq 0 \\ l + \hat{s}, & \text{otherwise.} \end{cases} \tag{4.9}$$

When evaluating the results, we noticed that defining the length in the way shown in Equation (4.9) presented a few challenges. Firstly, we used the entire training data to

obtain $\hat{s}$. Having $\hat{s} = \texttt{findfrequency}(\mathcal{D})$ yielded some large values of $\hat{s}$ that proved problematic when used with the Random Forest estimation method (see Section 5.2.5). To obtain smaller values for l_v^{cfpHO}, we had to make two changes.

The first one is that instead of using the length of the test set, l, for all cases, we defined a condition that returns either $\lceil (1-q_t)\cdot N \rceil$ (the definition of l) or $\lceil (1-q_e)\cdot N \rceil$ $\left(\text{the definition of } l_v^{\text{HO}}\right)$ as the new length l^*, depending on the ratio between the two. The other change is a condition imposed on the estimated value $\hat{s} = \texttt{findfrequency}(\mathcal{D})$. If it is larger than l^*, then we obtain the dominant frequency from the validation set obtained using the classic `Holdout` scheme. That is, $\hat{s} = \texttt{findfrequency}\left(\mathcal{V}^{\text{HO}}(\mathcal{D}; q_e)\right)$. Formally, we have:

$$l^* \equiv \begin{cases} l_v^{\text{HO}}, & \text{if } l_v^{\text{HO}}/l < 0.5, \\ l, & \text{otherwise.} \end{cases} \tag{4.10}$$

and

$$\hat{s} \equiv \begin{cases} \texttt{findfrequency}(\mathcal{D}), & \text{if } \texttt{findfrequency}(\mathcal{D}) < l^*, \\ \texttt{findfrequency}\left(\mathcal{V}^{\text{HO}}(\mathcal{D}; q_e)\right), & \text{otherwise.} \end{cases} \tag{4.11}$$

With these, we write the length of the validation set in the `cep-Holdout` scheme:

$$l_v^{\text{cepHO}} \equiv \begin{cases} l_v^{\text{HO}}, & \text{if } \hat{s} = 1 \\ \hat{s} \cdot \left\lceil l^* / \hat{s} \right\rceil, & \text{if } \hat{s} > 1 \text{ and } l^* \bmod \hat{s} \neq 0 \\ l^* + \hat{s}, & \text{otherwise.} \end{cases} \tag{4.12}$$

Then, $\mathcal{V}^{\text{cepHO}}(\mathcal{D}; l^*, \hat{s})$, the *composite estimated period-holdout forward validation scheme* is defined as:

$$\mathcal{V}^{\text{cepHO}}(\mathcal{D}; l^*, \hat{s}) = \left\{ (\mathcal{I}^e, \mathcal{I}^v) |\ \mathcal{I}^e = \left\{ 1, \ldots, \left(N - l_v^{\text{cepHO}} \right) \right\}, \right. \\ \left. \mathcal{I}^v = \left\{ \left(N - l_v^{\text{cepHO}} + 1 \right), \ldots, N \right\} \right\} \tag{4.13}$$

and the associated out-of-sample generalization error is:

$$\hat{\mathcal{L}} \left(\mathcal{D}, \mathcal{V}^{\text{cepHO}}, f \right) \equiv \frac{1}{l_v^{\text{cepHO}}} \sum_{(\mathbf{z}, y) \in \mathcal{D}(\mathcal{I}^v)} \ell \left[y, f \left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}(\mathcal{I}^e)} \right) \right] \tag{4.14}$$

Chapter 5: Data & Methodology

5.1 Data

5.1.1 Data from Cerqueira et al. (2020)

It is important to notice that the selection of $\hat{g}(\cdot)$ in Eq.(3.8) does not depend uniquely on the validation scheme selected. From Equation (3.7), we see that it also depends on the training data ($\mathcal{D}$), on the learning algorithm (through $\hat{\boldsymbol{\theta}}$), and the loss function. Therefore, to compare our results to those from CTM, we will need to use the same training data, learning algorithm, and loss function that they used.

Out of the 174 real-world data sets used by CTM, 149 univariate time series with more than 500 observations without missing values were taken from the benchmark database *tsdl* from Hyndman and Yang [70]. The remaining 25 were taken from Cerqueira et al. [87]. CTM selected time series with at least 500 observations (the range for the sample size goes from 506 to 4000 observations) so that the machine learning algorithms had enough data to estimate the parameters properly. The 174 series were taken from many different fields, that include finance, physics, economy, energy, and meteorology [22, p. 2005]. Moreover, 97 are stationary, while the remaining 77 are non-stationary.

As for the seasonality, Table 5.1 shows s (the period from the `frequency` func-

tion) and $\hat{s}$ (obtained using the `forecast::findfrequency` function). We see from it that most series ($n = 60$ in each case) are either aperiodic ($s = 1$) or with monthly periodicity (i.e., with frequency $s = 12$). The function `forecast::findfrequency` was able to "correctly" identify 41 of the aperiodic series and 48 of the monthly time series. The highest period identified by s was 365 (daily), while $\hat{s}$ yielded a maximum estimated period of 499.

5.1.2 Data from the M4 Competition

The M4 Forecasting Competition took place in 2018. Participants in this competition had to forecast values for 100,000 real-life time series. The number of forecasts required was 6 for yearly data (minimum sample size of 13), 8 for quarterly data (minimum sample of 16), 18 for monthly time series (min. sample of 42), 13 for weekly data (min. sample of 80), 14 for daily (min. sample of 93), and 48 for hourly data (minimum sample size of 700).

We randomly selected 1,000 time series from these one hundred thousand and used them to evaluate the performance of the different validation schemes. Since we wanted to see how they behaved on periodic data alone, we selected only the series with at least 100 observations, and in which their period was greater than 1, and if this period was equal to the period of the dominant frequency of a time series (estimated using the `findfrequency` from the `forecast` package[1]). In the end, we had 164 series with $s = \hat{s} = 4$, 778 with both period and estimated period equal to 12, and 55 with similarities

[1] For cyclic data, this function returns the average cycle length. For seasonal data, it returns the seasonal period.

in the series occurring after 24 basic time intervals.

Table 5.1: Period and estimated periods for the univariate time series in Cerqueira et al.(2020)'s data set.

$\hat{s}$ \ s	1	12	13	24	48	365	Total
1	41	6	0	4	0	8	59
3	1	0	0	0	0	1	2
4	3	0	0	0	0	0	3
5	0	0	0	0	0	1	1
6	0	4	0	0	1	1	6
7	2	0	0	0	0	2	4
10	1	0	0	0	0	0	1
11	0	0	0	0	0	2	2
12	1	48	0	6	0	0	55
13	0	0	1	0	0	1	2
15	0	0	0	0	0	1	1
16	0	0	0	0	0	2	2
17	0	0	0	0	0	1	1
21	0	0	0	0	0	1	1
22	1	0	0	0	0	0	1
23	1	0	0	1	0	0	2
24	1	0	0	13	0	0	14
25	0	0	0	0	0	1	1
29	2	0	0	0	0	0	2
30	0	0	0	0	0	1	1
42	1	0	0	0	0	0	1
45	1	0	0	0	0	0	1
48	1	0	0	0	2	0	3
50	0	0	0	0	2	0	2
125	0	2	0	0	0	0	2
143	1	0	0	0	0	0	1
250	0	0	0	1	0	0	1
333	1	0	0	0	0	0	1
499	1	0	0	0	0	0	1
Total	60	60	1	25	5	23	174

5.1.3 Monte Carlo Simulation

The third data set, called $S3$, is a counterexample developed by [21] to work as a situation where the cross-validation procedures break down. The 1000 time series with 200 observations were generated following a seasonal AR process with a significant lag 12 (seasonal lag 1), that is, a SARIMA$(12,0,0) \times (1,0,0)_{12}$. The authors obtained the parameters for the data generating process by fitting the seasonal AR model to the `USAccDeaths` data set shown in Figure 4.1. This dataset is included in a standard installation of `R`.

The models used by both CTM and Bergmeir et al. [21] to analyze the simulated data only use up to the first five lags. By restricting the number of lags, Bergmeir et al. [21] expected the models to not fit well data. Curiously, CTM's results show that some cross-validation procedures yielded good results. We hypothesize that they used different forms of the validation schemes as we showed in Section 3.3.

We used the $S3$ dataset but also modified it to include a seasonal integration of order 1. In other words, we also simulated 1000 time series with 200 observations with parameters estimated using the `USAccDeaths` data set, but according to the specification SARIMA$(12,0,0) \times (1,1,0)_{12}$. The goal here was to evaluate how the validation schemes deal with an integrated seasonal process. We called this dataset $S4$.

5.2 Methodology

5.2.1 Terminology

To make the names of the schemes defined here compatible with the ones used by CTM, we use the short names shown in Table 5.2. The short names are also used in the plots with the final results (Chapter 6 and Appendix B).

Table 5.2: Names and short names of the validation schemes used here, alongside the number of the Sections with discussions about them, and the exact equation number used to define the indexes for the estimation and validation sets.

Type	Name	Short Name	Sec.	Eq. #
Forward Validation	Holdout	`Holdout`	3.2.1	3.10
	Repeated Holdout	`Rep-Holdout`	3.2.2	3.18
	Period-Holdout	`p-Holdout`	4.1	4.4
	Composite Period-Holdout	`cp-Holdout`	4.2	4.7
	Composite Estimated-Period-Holdout	`cep-Holdout`	4.3	4.13
	Prequential Growing Window	`Preq-Grow`	3.2.3	3.21
	Prequential Sliding Window	`Preq-Slide`	3.2.4	3.23
	Prequential in Blocks	`Preq-Bls`	3.2.5	3.28
	Prequential Sliding Blocks	`Preq-Slide-Blocks`	3.2.6	3.31
	Prequential Blocks with Gaps	`Preq-Bls-Gap`	3.2.7	3.34
Cross-Validation	Cross-validation (*k*-Fold CV)	`CV`	3.3.1	3.37
	Blocked *k*-Fold Cross-Validation	`CV-Bl`	3.3.2	3.39
	hv-Block Cross-Validation	`CV-hvBl`	3.3.3	3.51
	Modified cross-validation	`CV-Mod`	3.3.4	3.56

5.2.2 The Forecasting Model and the Embedded Matrix

Following CTM, we use a univariate autoregressive process to model the series. This process can be represented by the time delay embedding method proposed by Takens [75, *apud* CTM]. In this method, the lag order, p, of the model is estimated (see below), and the time series is embedded accordingly. The resulting matrix is similar to the one seen in Eq.(5.1) (this one was created for a lead time $l = 1$) and will be used as the input

for the learning algorithms (see Section 5.2.5).

$$Y_{[T,p]} = \begin{pmatrix} y_1 & y_2 & \cdots & y_p & y_{p+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_{t-p} & y_{t-p+1} & \cdots & y_{t-1} & y_t \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_{T-p} & y_{T-p+1} & \cdots & y_{T-1} & y_T \end{pmatrix} \tag{5.1}$$

In the above matrix, we see that each row is of the form $[\mathbf{z}_t^\top, y_t]$, where $\mathbf{z}_t^\top \in \mathbb{R}^p$ is the vector comprised of the lagged values of y_t. Thus, the first p columns of the matrix in Eq. (5.1) contain the predictors for the last column of the matrix.

The optimal embedding dimension, p, is estimated using the False Nearest Neighbor algorithm, developed by Kennel et al. [88]. The idea behind this algorithm is to search for regression vectors that are close to a vector that produces a good prediction. By similarity, it is expected that these regression vectors will produce forecasts that are close to each other. However, if those regression vectors produce vastly different future outputs, they are deemed *false neighbors* [89]. The embedding dimension is selected to be the one in which the number of false neighbors is dropped to an acceptably small percentage[2]. The algorithm uses the training data to search for p up to a maximum embedding dimension equal to 30. The tolerance to false nearest neighbors was set to 1% [22, p. 2008].

[2]In their code, CTM set a minimum value of 8 for the estimated dimension. Furthermore, when creating the embedded matrix, they added 1 to the estimated value. They did not discuss why they did this.

5.2.3 Length of the Sets and Number of Folds

Once we have the final embedded matrix for each of the simulated data sets, or the 174 time series from CTM, they will be divided into the training and test sets using the proportions of 70% and 30%, respectively (that is, $q_t = 0.7$). Then, following Bergmeir et al. [21], we take $q_e = 0.8$ for the simulated series. For the 174 real-life series, we use $q_e = 0.7$, as did CTM.

For the M4 Competition, the organizers had defined the length of the forecast horizon l for each type of series, as discussed in Section 5.1.2, so we used the provided lengths as the lengths of the test and validation sets for each one of the 1,000 selected series. After the competition was over, the organizer provided the true out-of-sample observations (with length l), and these were used as the test sets. The length of the training sets, N, was the maximum length of the series available before the end of the competition [11].

In all cases (Simulated, Cerqueira and M4), the proportion q_e used for the estimation set with the `Rep-Holdout` scheme was 60% of the training data, while the validation set contained the last 10% observations from the training data (i.e., $q_v = 0.1$), as it was done by [22, p. 2008]. For the `Rep-Holdout` and the other schemes that require the number of folds, this number was set to 10 [22, p. 2008].

5.2.4 Stationarity

We also divided the series into stationary and non-stationary cases to see how the validation schemes behave in each case. According to the original authors:

> "In order to test if a given time series is stationary we follow the wavelet spectrum test described by Nason (2013). This test starts by computing an

> evolutionary wavelet spectral approximation. Then, for each scale of this approximation, the coefficients of the Haar wavelet are computed. Any large Haar coefficient is evidence of a non-stationarity. An hypothesis test is carried out to assess if a coefficient is large enough to reject the null hypothesis of stationarity. In particular, we apply a multiple hypothesis test with a Bonferroni correction and a false discovery rate (Nason, 2013)." (Cerqueira, Torgo, and Mozetič, 2020, p. 2005)

5.2.5 Learning Algorithms

The learning algorithms (i.e., the methods used to estimate θ) applied by CTM are the following:

RBR: a rule-based regression algorithm from the Cubist R package [91], which is a variant of the M5 model tree algorithm [92, 93];

RF: a random-forest algorithm, which is an ensemble of decision trees [94];

GLM: a generalized linear model with a Gaussian distribution and a Ridge penalty mixing.

Quinlan's M5 model tree algorithm [93] is a supervised learning method used to predict continuous values. Tree-based models are constructed by the divide-and-conquer method. That is, they partition the covariate (feature) space of the training data into a set of rectangles and then fit independent models in each one (Figure 5.1 - bottom part). The same model can be represented by the tree in the upper part of Figure 5.1. The partition is based on the split-points that provide the best fit, i.e. the points that subset the training data in a way that minimizes the variability inside each rectangle. This metric depends on the standard deviation of the target/response variable taken into the entire training sample and taken inside each rectangle.

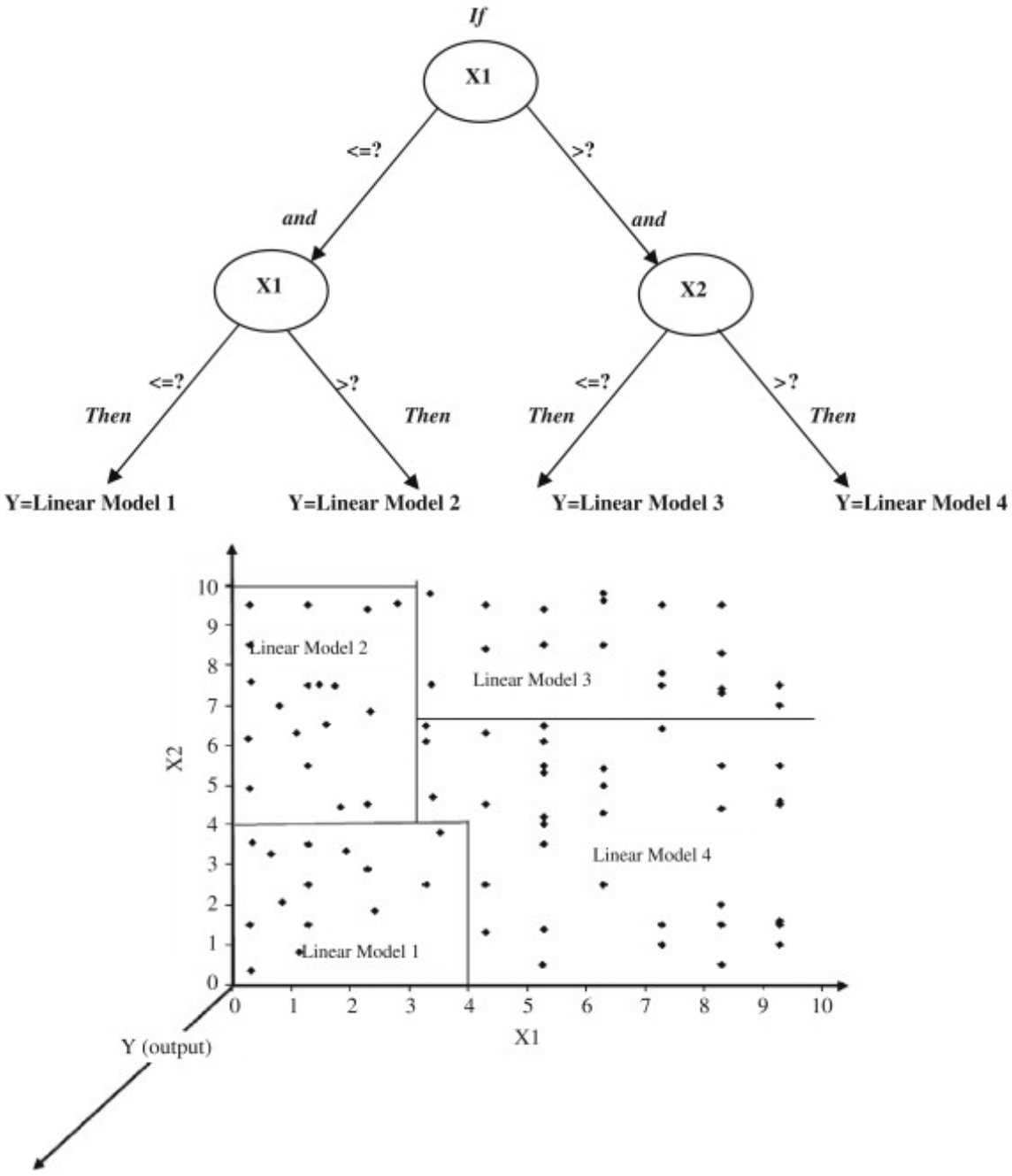


Figure 5.1: Example of splitting the input space $X_1 \times X_2$ by the M5 model tree algorithm. Image source: Etemad-Shahidi and Mahjoobi [95, p. 1177].

Cubist is a rule-based regression (RBR) model that combines the algorithms that Quinlan [92, 93] developed into an improved approach. The main improvements include an ensemble method for predictions called committees [96], where iterative model trees are created in sequence. The model in the first tree (i.e., in the first committee) uses the original response variable. Subsequent trees (committees) are created using adjusted versions of this response so that if the model over-predicted the outcome, its adjusted value becomes larger so that the model is pulled downwards for the next iteration in an attempt to stop over-predicting. The final prediction is the average of the predictions from each model tree (that is, "ensemble predictions are made by averaging over the committee model predictions", [96]). Following CTM [22, p. 2009], we use 5 committees.

The "Random Forests" (RF) algorithm basically builds a large collection of de-correlated trees, and takes their average [74, p. 587]. Because of this, it is often defined as an ensemble of decision tree algorithms. It differs from the model tree algorithms because the variables are randomly selected as candidates for splitting instead of a rule-based method. Another major difference is that it evaluates the models in a bootstrap sample of the training data instead of the rectangles created from a partition of the covariate/feature space as in the RBR model. This assures that the grown tree in each bootstrap sample (one tree per sample) yields uncorrelated outcomes from the other trees. We used 100 trees/bootstrap samples[3].

Lastly, we used a generalized linear model with a Gaussian distribution and a Ridge penalty mixing, or simply known as "ridge regression." This method shrinks the re-

[3]Due to the random nature of this method, and since Cerqueira et al. [22] did not set a seed in their code, our results in Chapter 6 are slightly different from theirs. Our discussion about this with the corresponding author can be seen here: `https://github.com/vcerqueira/performance_estimation/issues/1`.

gression coefficients by imposing a penalty on their size, which alleviates the effects of multicollinearity[74, p. 61-64]. On the one hand, this adds some degree of bias, but on the other, it reduces the standard errors. In this bias-variance trade-off, ridge regression, in general, provides more gains in terms of efficiency in exchange for a tolerable amount of bias.

5.2.6 Forecast Accuracy Measures

Following the work of Bergmeir et al. [21], CTM wanted to compare the different validation schemes by evaluating how close $\hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}, \hat{g}\right)$ would be to $\hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{D}_{test}, \hat{g}\right)$[4], the "ground truth loss" [22, p. 2009]. For that, they used the absolute predictive accuracy error (APAE), and the predictive accuracy error (PAE) metrics. The APAE metric evaluates the error size of a given validation scheme, and is defined in Eq.(5.2), below. On the other hand, PAE, as defined in Eq.(5.3), measures the error bias of a validation scheme - that is, if it is underestimates or overestimates the "true" error.

$$\text{APAE} \equiv \left|\hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}, \hat{g}\right) - \hat{\mathcal{L}}(\mathcal{D}, \mathcal{D}_{test}, \hat{g})\right| \tag{5.2}$$

$$\text{PAE} \equiv \hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}, \hat{g}\right) - \hat{\mathcal{L}}(\mathcal{D}, \mathcal{D}_{test}, \hat{g}) \tag{5.3}$$

[4]That is, Eq.(3.7) with $\mathcal{D}_{test}$ in place of $\mathcal{V}$.

where $\mathcal{V}$ is any of the validation schemes presented in Sections 3.2 and 3.3

The first loss function, ℓ, used inside each $\hat{\mathcal{L}}$ was the quadratic loss. We used it because it is a metric traditionally used to evaluate forecast accuracy. With it, the generalization error $\mathcal{L}$ becomes the mean square error, MSE. To control for its size, we calculated the root mean square error (RMSE). Using this error measure makes our results comparable to the ones by Bergmeir et al. [21]. These are the results shown in Chapter 6.

However, since the main paper that we are using to compare our results is the one by [22], we also use the mean absolute scaled error as calculated for the estimation and validation set as follows (that is, $\hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}, \hat{g}\right)$:

$$\text{MASE} \equiv \frac{1}{l_v} \cdot \frac{\sum_{t=l_e+1}^{l_e+l_v} |y_t - \hat{y}_t|}{\left(\frac{1}{N-s}\right) \sum_{t=s+1}^{N} |y_t - y_{t-s}|} \tag{5.4}$$

Calculating $\hat{\mathcal{L}}(\mathcal{D}, \mathcal{D}_{test}, \hat{g})$ using the MASE metric is similar. This metric is suitable since it allows us to consider the periodicity of a time series. Moreover, since it is scale-invariant, it "can be used to compare forecast methods on a single series and also to compare forecast accuracy between series" [97, 43].

Our final evaluation consists of taking the APAE metrics for all the validation schemes and sorting these values for a given time-series. The procedure that yields the smallest APAE values will receive a rank equal to 1 (meaning, it is the best for that specific series), while the "worst" scheme (with highest APAE) will be ranked the 14th method (i.e., the last place, since we have 14 procedures). Then, we take the average rank among all series and sort the procedures according to this average rank. These sorted ranks comprise the first set of results shown in Chapter 6 and Appendix B.

The second set of results is based on the PAE metric. For that we will calculate the log percentage difference of the estimated loss, $\hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}, f\right)$, relative to the true loss, $\hat{\mathcal{L}}(\mathcal{D}, \mathcal{D}_{test}, \hat{g})$. Values below zero represent under-estimations of the error. Conversely, values above it represent over-estimations of the error. Ideally, a scheme would produce a log percentage difference close to zero.

5.2.7 Hypothesis Testing and Comparisonwise Error Rate

Let us briefly review the types of errors. Recall that we have two types of errors related to hypothesis testing: Type I and Type II. Consider the case where we are testing a hypothesis about the mean of a population, and someone states that the real mean is equal to some value μ_0, and say that our alternative hypothesis is that this mean is different from μ_0. In this case, we have the following:

$$H_0 : \mu = \mu_0$$

$$H_A : \mu \neq \mu_0$$

From the way we have defined our hypotheses, H_0 is false if the real parameter μ is anything but μ_0. Since we have many possibilities for that (i.e., many possible values for the real parameter, μ, as long as it is different from μ_0), we say that H_A is a composite hypothesis. On the other hand, in the case H_0 has only one value and we say that it is a simple hypothesis.

Now, we define the Type II error as the error we make when we do not reject H_0 when it is false. Because μ can be anything but μ_0, the probability of making a Type II

error will be a function of all μ that are different from μ_0, and this function is usually denoted by $\beta(\cdot)$:

$$P(\text{Do not Reject } H_0 | H_0 \text{ false}) = \beta(\mu), \forall \mu \neq \mu_0$$

Type I error is the one we make when we reject H_0 when it is true, and the probability of making such error in that situation depends, in general, on the value specified in H_0. In other words,

$$P(\text{Reject } H_0 | H_0 \text{is true}) = P(\text{Type I error } | \mu = \mu_0)$$

And if the probability of type I error satisfies

$$P(\text{Type I error }) \leq \alpha, \quad \text{when } \mu = \mu_0,$$

Then we say that the test has a significance level α, and α is the ceiling for the probability of a Type I error.

Now, we have seen how to use the two-sample t-test to compare the means of two samples. In such case, we could have a null hypothesis that looks like this: $H_0 : \mu_1 = \mu_2$, where μ_1 and μ_2 are the population means for groups 1 and 2, respectively. However, when you have more groups, it is of interest to examine the differences between these groups. In such cases, we can have two types of null hypothesis: a complete null hypoth-

esis, or a partial null hypothesis: One example of a complete null hypothesis is:

$$H_0 : \mu_1 = \mu_2 = \ldots = \mu_K,$$

where K is the number of groups/levels/factors/categories that is, we are evaluating if all group means are the same. On the other hand, an example of a partial null hypothesis is:

$$H_0 : \mu_1 = \mu_2 = \ldots = \mu_{K-1} \neq \mu_K,$$

and, in this case, we are testing if the mean of one group is different from all other groups and that all other groups have the same mean.

To evaluate the probability of type I error in such cases, we could break apart these hypotheses into several simple hypothesis. In this case, we would have several probabilities of making a Type I error, one for each individual test between two means.

In a multiple comparison setting, we would be looking for the value of α at each simple case. One could use the same α for each simple case, and let us take a look at what happens when we do that. Recall our partial null hypothesis example and let us say that $K = 3$. Then,

$$H_0 : \mu_1 = \mu_2 \neq \mu_3$$

Breaking this up into three hypothesis yields:

$$H_0^1 : \mu_1 = \mu_2; \quad H_0^2 : \mu_1 \neq \mu_3; \quad H_0^3 : \mu_2 \neq \mu_3$$

If we use the $\alpha = 0.05$ for each of them - and, to ease the calculations, assume that the probabilities of a Type I error are independent among all comparisons -, then the probability that at least one of these tests yields an erroneous rejection raises to 0.143 (see below).

$$1 - P(\text{not one test yield an erroneous rejection of the null}) = 1 - 0.95^3$$
$$\approx 0.143$$

Thus, the above probability increases with the number of comparisons (single hypothesis) that we make. This is the multiple comparison problem. One way we can solve it is by controlling the overall type 1 error rate for all the comparisons. This overall Type 1 error rate is called *experimentwise* error rate.

It is difficult to calculate an experimentwise error rate's exact probability, but we can derive a pessimistic approximation by assuming that the comparisons are independent and giving an upper bound to it:

$$1 - (1 - \alpha_{CW})^C \leq \alpha_{EW} \tag{5.5}$$

where α_{EW} is the experimentwise error rate, α_{CW} is the *comparisonwise* error rate (i.e., the error rate between two comparisons), and C is the total number of comparisons.

In an experiment, if one wants to control the overall Type I error rate for all the comparisons, they are controlling the experimentwise error rate, α_{EW}. On the other hand, if they decide to control the individual type I error rates for each comparison, they are

controlling the individual or comparisonwise error rate, α_{CW}.

In situations where incorrectly rejecting one comparison may jeopardize the entire experiment or where the consequence of incorrectly rejecting one comparison is as serious as incorrectly rejecting a number of comparisons, the control of experimentwise error rate is more important. Now, when one erroneous conclusion will not affect other inferences in an experiment, the comparisonwise error rate is more pertinent.

One can control α_{EW} at the α level by setting α_{CW} to a sufficiently small value. The Bonferroni inequality has been widely used for this purpose. If

$$\alpha_{CW} = \frac{\alpha}{C},$$

then α_{EW} is less than α.

In the end, we choose what we will control by choosing the test we use. We have that individual tests (like individual two-sample t-tests) control α_{CW}. On the other hand, tests that yield confidence inequalities or confidence intervals (like Scheffe's) control α_{EW} under any complete or partial null hypotheses. Moreover, a preliminary F test controls α_{EW} under the complete null hypothesis but not under the partial null hypothesis.

Our evaluation of the PAE metric will be similar to a simple t-test with $H_0 = 0$. We do this because we are interested in knowing if the average log PAE metric for a scheme is close to zero. Here, we are interested in evaluating between the groups (the validation schemes), and incorrectly rejecting one comparison will not jeopardize our entire evaluation. Thus, we will control for the comparisonwise error rate, α_{CW}. We will use a non-parametric alternative to the t-test, called the Sign test. This non-parametric

test makes no assumptions regarding the distribution of the parameter, and is suitable for our case.

The number of comparisons is equal to the number of validation schemes. That is, $C = 14$. Moreover, the maximum value of α is set to 0.15. Then, $\alpha_{CW} = 0.15/14 \approx 0.011$.

Furthemore, we analyzed the statistical significance of the APAE metric using the same approach as CTM. According to these authors:

> "We also study the statistical significance of the obtained results in terms of error size (APAE) according to a Bayesian analysis [98]. Particularly, we applied the Bayes signed-rank test to compare pairs of methods across multiple problems. We arbitrarily define the *region of practical equivalence* (Benavoli et al. 2017) (ROPE) to be the interval [-2.5%, 2.5%] in terms of APAE. Essentially, this means that two methods show indistinguishable performance if the difference in performance between them falls within this interval. For a thorough description of the Bayesian analysis for comparing predictive models we refer to the work by Benavoli et al. (2017). In this analysis, it is necessary to use a scale invariant measure of performance. Therefore, we transform the metric APAE into the percentage difference of APAE relative to a baseline." (Cerqueira, Torgo, and Mozetič, 2020, p. 2013. Emphasis in the original.)

5.2.8 The Complete Experimental Design

In summary, CTM's experimental design (and the one used here), has the following steps for each one of the time series:

- Step 01: For the RF and GLM methods, calculate the number of differences required to make the series stationary, and take the differences. Otherwise, use the series as is;

- Step 02: Estimate p and get the embedded matrix. The columns of this matrix (other than the "target" column) yields $\hat{g}$;

- Step 03: Split the entire data set into training set ($\mathcal{D}$) and test data set ($\mathcal{D}_{test}$), following the `Holdout` validation scheme;

- Step 04: For each one of the estimation methods above (RBR, RF, GLM), calculate the "true loss," $\hat{\mathcal{L}}(\mathcal{D}, \mathcal{D}_{test}, \hat{g})$ (as both the RMSE and the MASE);

- Step 05: For each estimation method, partition the training data into the estimation and validation sets according to each validation scheme, and calculate $\hat{\mathcal{L}}(\mathcal{D}, \mathcal{V}, \hat{g})$ (using both RMSE and MASE);

- Step 06: For each method calculate the APAE (Eq.5.2) metric, and rank the validation schemes from smaller APAE (better, rank $= 1$) to higher (worse, rank $= 14$, as we evaluate 14 schemes);

- Step 07: For each method calculate the average rank (over the time series in the category[5]) based on the APAE measure;

- Step 08: Evaluate if the difference in the average ranks between the schemes is statistically significant using the Bayes signed-rank test;

- Step 09: For each method calculate the PAE (Eq. 5.3) metric;

[5]We have three major categories: data from CTM, from the M4 Competition sample, and from our Monte Carlo simulations. For the first two, we calculated the overall average, the average amongst only the stationary series, and the average over the non-stationary ones. For the latter, we divided it between $S3$: SARIMA$(12,0,0) \times (1,0,0)_{12}$ and $S4$: SARIMA$(12,0,0) \times (1,1,0)_{12}$ and calculated the averages inside each one of those.

- Step 10: Calculate the log percentage difference of the estimated loss, $\hat{\mathcal{L}}\left(\mathcal{D}, \mathcal{V}, \hat{g}\right)$, relative to the true loss, $\hat{\mathcal{L}}(\mathcal{D}, \mathcal{D}_{test}, \hat{g})$;

- Step 11: Evaluate if the (median) log percentage difference is statistically different from zero using the Sign test;

- Step 12: Plot the results from steps 07, 08 and 10. The plots are displayed in Chapter 6 for $\hat{\mathcal{L}}$ calculated using the RMSE, and in the Appendix B for the MASE;

The entire process described above was conducted in the software `R`. The final code used is a modification of the code from CTM, and it is fully available on our paper's corresponding author's GitHub repository [6]. All the data sets, results, and plots are also included inside this repository. The functions created specifically for this paper (i.e., the functions for the *p*-Holdout family and the evaluation of its results) can also be found in Appendix C.

[6] `https://github.com/gu-stat/validation_schemes`

Chapter 6: Results

6.1 Data from Cerqueira et al. (2020)

6.1.1 Results from the RBR learning algorithm

We start by analyzing the results for the RBR estimation method. Regarding all 174 real-life time-series and the RMSE as error measure, the `cep-Holdout` scheme came in fourth place, behind cross-validation methods like the `CV-hvBl` , `CV-Bl` , and `CV` (Figure 6.1). These procedures also yielded an average error bias closer to zero (Figure 6.2). The medians were also close to zero with p-values for the sign test greater than $\alpha_{CW} = 0.011$ (Table A.1).

However, when we use the MASE, this scenario changes (Figure B.1). For instance, `CV` now is the last-placed procedure, and the `cep-Holdout` comes in second place behind the `Rep-Holdout` (which came in 10th place with the RMSE). But this last procedure had a much larger average (log) error bias (0.1775, median = -0.7885 with p-value $= 0.7048$; Table B.1) than the `cep-Holdout` scheme (0.0498, median = -0.4583 with p-value = 0.5958). Only the `CV-Bl` and `cp-Holdout` schemes had an average log error smaller than the `cep-Holdout` procedure (0.0244 and 0.0362, respectively; Table B.1).

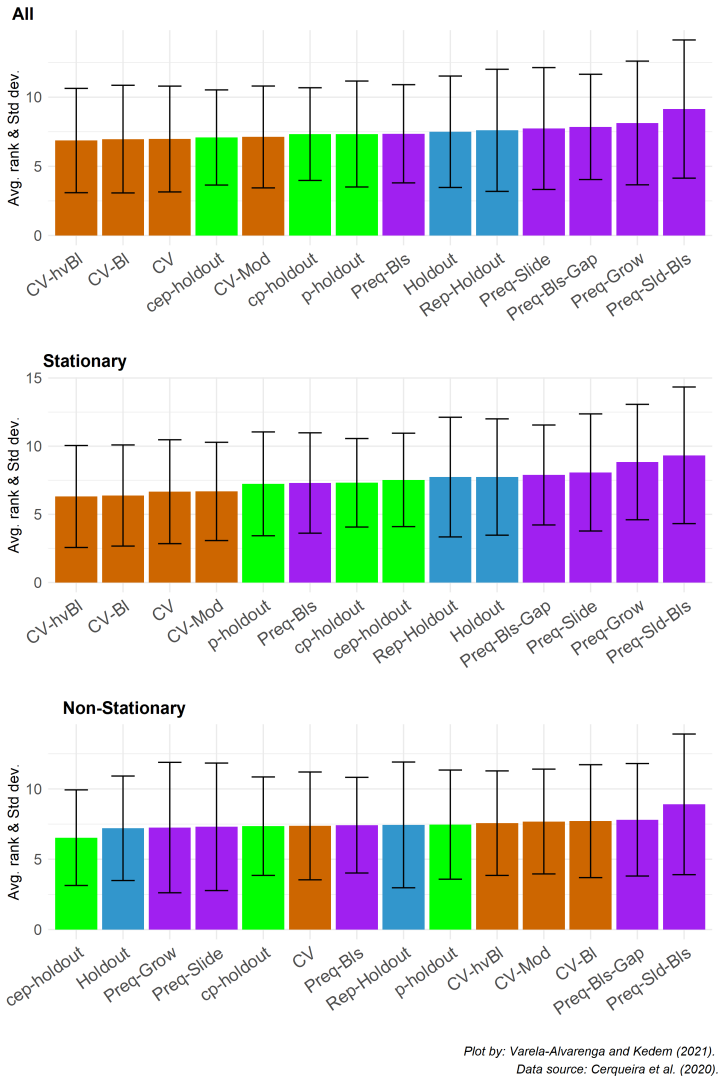


Figure 6.1: Average APAE rank of each validation scheme on 174 real-world time series using the RBR learning algorithm and RMSE as the error function. The black line represents -+ 1 standard deviation from the average.

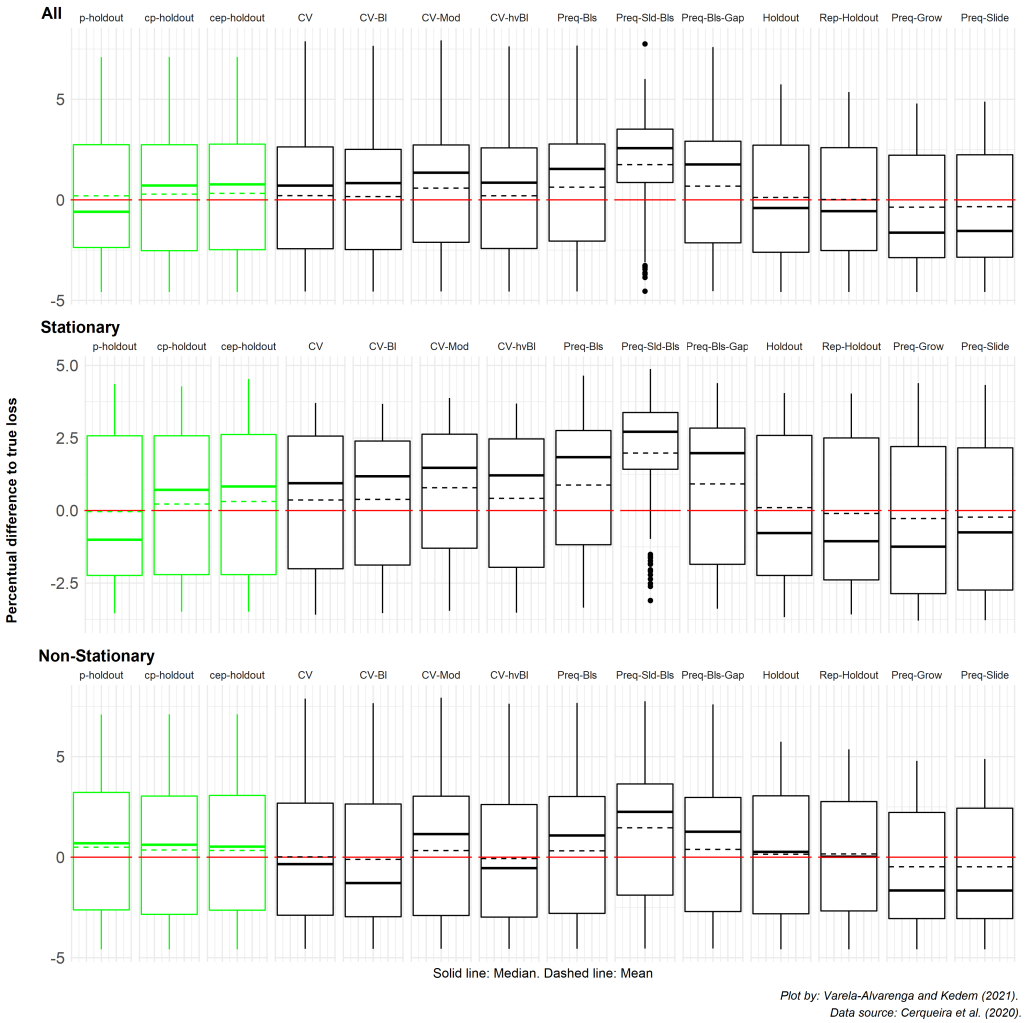


Figure 6.2: Log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to 174 real-world time series using the RBR learning algorithm. Values below the zero (red) line represent under-estimations of the error. Conversely, values above it represent over-estimations of the error.

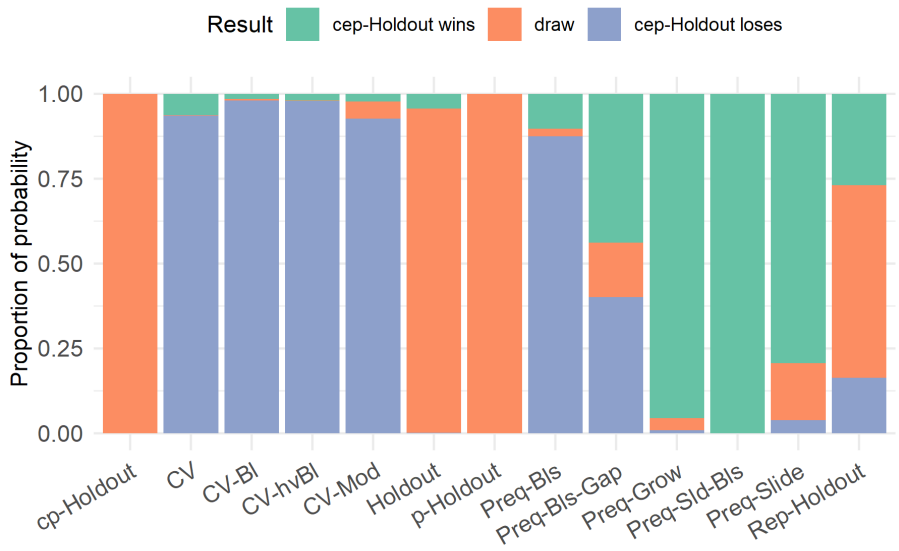


Figure 6.3: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation scheme when applied to the stationary time-series from Cerqueira et al. (2020), with parameters estimated via the RBR learning algorithm and error calculated using the RMSE. The probabilities are computed using the Bayes signed-rank test.

When we evaluate only the non-stationary series, the `cep-Holdout` scheme is the procedure that often returns the smallest prediction error (measured by the APAE metric) for both error measures (RMSE and MASE) (Figures 6.1 and B.1). The `cep-Holdout` yields a fairly large error bias under the RMSE (0.3344, 8th place), but a very small bias for when using the MASE (0.0263). In fact, all schemes in the *p-Holdout* family produced much smaller error bias under the MASE measure, than the other methods. For instance, the log percentage difference for the `p-Holdout` , `cep-Holdout` and `cp-Holdout` were equal to 0.0108, 0.0263, and -0.0263, respectively. In 4th place comes the `Rep-Holdout` scheme, with an average error bias of 0.2071.

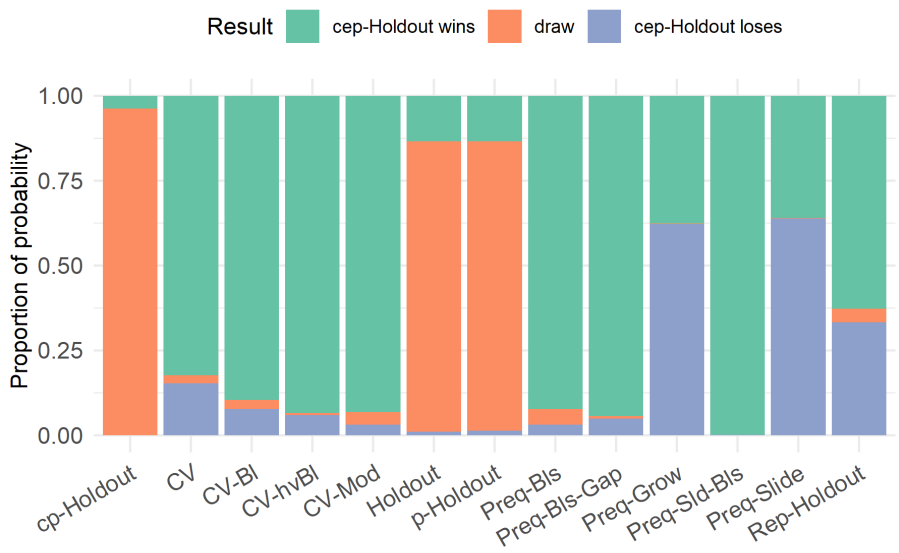


Figure 6.4: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation schemes when applied to the non-stationary time-series from Cerqueira et al. (2020), with parameters estimated via the RBR learning algorithm and error calculated using the RMSE. The probabilities are computed using the Bayes signed-rank test.

The plot in Figure 6.4 also shows the superiority of the cep-Holdout procedure in the non-stationary case. According to this plot, the `cep-Holdout` scheme has a significantly better estimation ability, since its probability of winning is larger then the ones for the cross-validation procedures. Based on Figure 6.4, the `Preq-Grow` and `Preq-Slide` schemes seem to be good competitors. However, these are computationally expensive and also yield a larger error bias (Table A.3).

6.1.2 Results from the RF learning algorithm

The results for the Rf learning algorithm are very similar to the ones for the RBR.

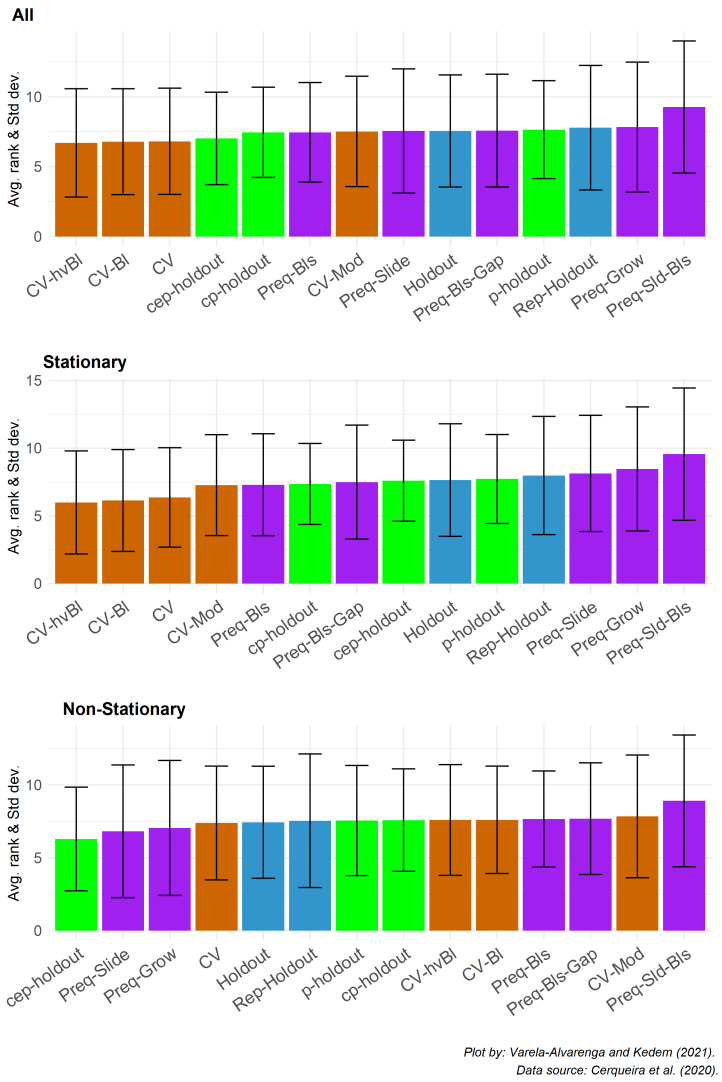


Figure 6.5: Average APAE rank of each validation scheme on 174 real-world time series using the RF learning algorithm and RMSE as the error function. The black line represents -+ 1 standard deviation from the average.

Overall, the `cep-Holdout` method ended up behind the `CV-hvBl` , `CV-Bl` , and `CV` schemes in terms of the average APAE metric when used in conjunction with the RMSE (Figure 6.5). In terms of the MASE, the `cep-Holdout` ended up in first place (versus the 2nd place it ended up when the RBR was used - Figure B.5).

When looking only at the 77 non-stationary series, the `cep-Holdout` scheme

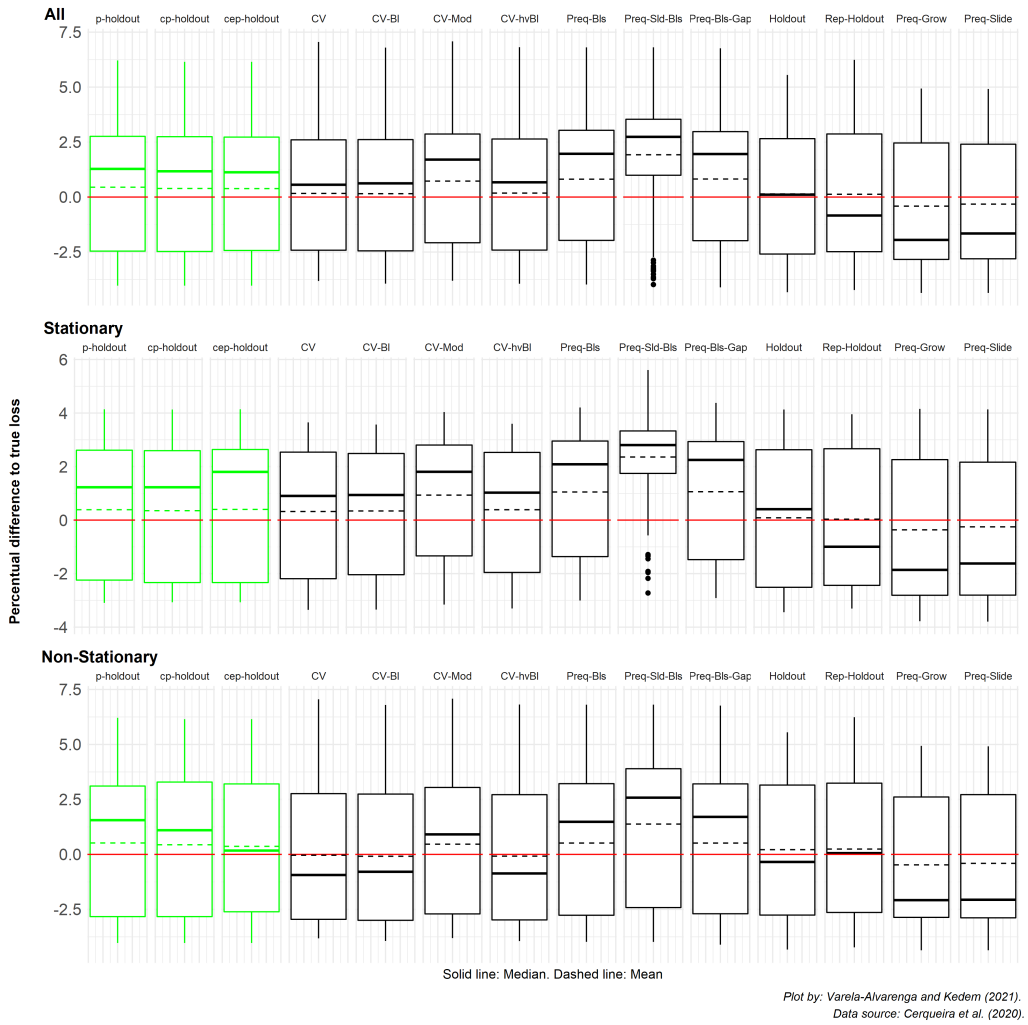


Figure 6.6: Log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to 174 real-world time series using the RF learning algorithm. Values below the zero (red) line represent under-estimations of the error. Conversely, values above it represent over-estimations of the error.

was the first, again (Figures 6.5 and B.5 - "Non-Stationary" panel), with the lowest APAE score, on average. However, it displayed, on average, a much larger bias than cross-validation procedures when the RMSE was used (Figures 6.6 - "Non-Stationary" panel).

On the other hand, when MASE was used (Figure B.6), our schemes had a per-

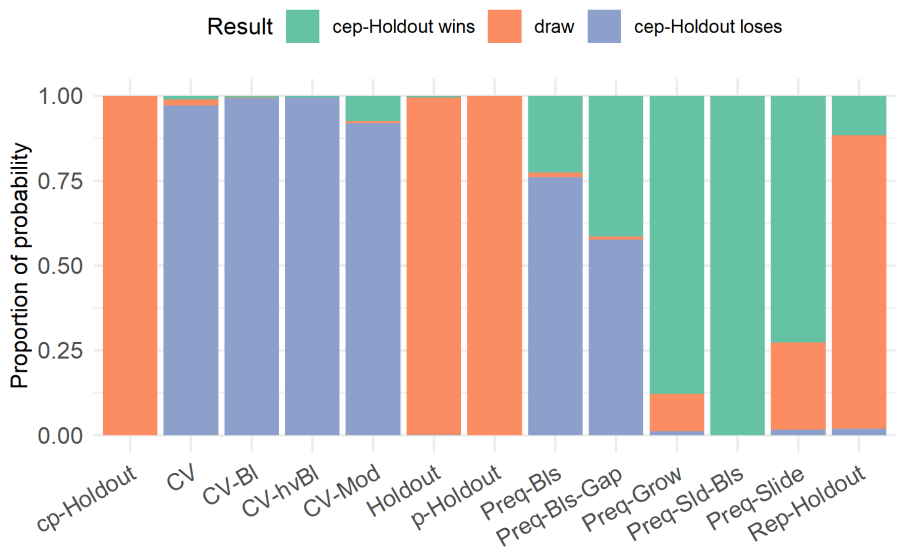


Figure 6.7: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation scheme when applied to the stationary time-series from Cerqueira et al. (2020), with parameters estimated via the RF learning algorithm and error calculated using the RMSE. The probabilities are computed using the Bayes signed-rank test.

centual difference to the true loss very close to zero, with the `p-Holdout` being the closest (0.0494), behind only to the `Holdout` scheme (-0.0273). Moreover, all three schemes in the *p-Holdout* family returned slightly pessimistic results (i.e., they over-estimated the error), with the average log percentages being equal to 0.0494, 0.1832, 0.1680, and for the `p-Holdout` , `cp-Holdout` , and `cep-Holdout` , respectively. The `Preq-Slide` procedure, that ended in second place when either the RMSE or MASE were used, yielded larger values of the error bias (-0.4075 in the RMSE case, and -0.3931 when the MASE was employed).

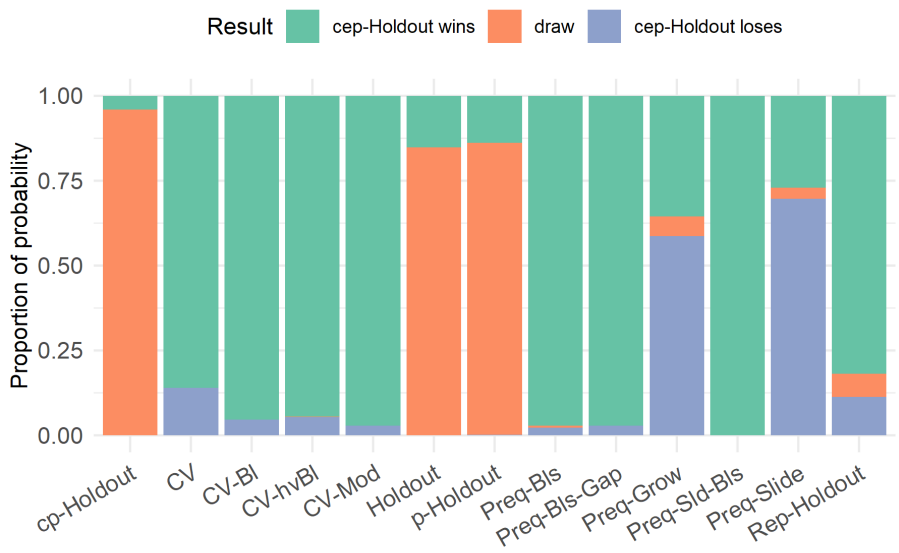


Figure 6.8: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation schemes when applied to the non-stationary time-series from Cerqueira et al. (2020), with parameters estimated via the RF learning algorithm and error calculated using the RMSE. The probabilities are computed using the Bayes signed-rank test.

6.1.3 Results from the GLM learning algorithm

The average APAE results for the 174 real-life time series using GLM-Ridge as the estimation method and the RMSE as error measure are shown in Figure 6.9. The novelty now is the presence of the `p-Holdout` scheme in second place in the "All" case and in third place in the "Non-Stationary" case. However, it ended up in 6th place, overall, in terms of the error bias, and in 10th place when only non-stationary series were evaluated (Figure 6.10), Another important result for our family is related to the `cep-Holdout` method. Not only did it yielded the best forecast accuracy in terms of the APAE metric when the RMSE was used, but also produced forecasts with the smallest average error

bias.

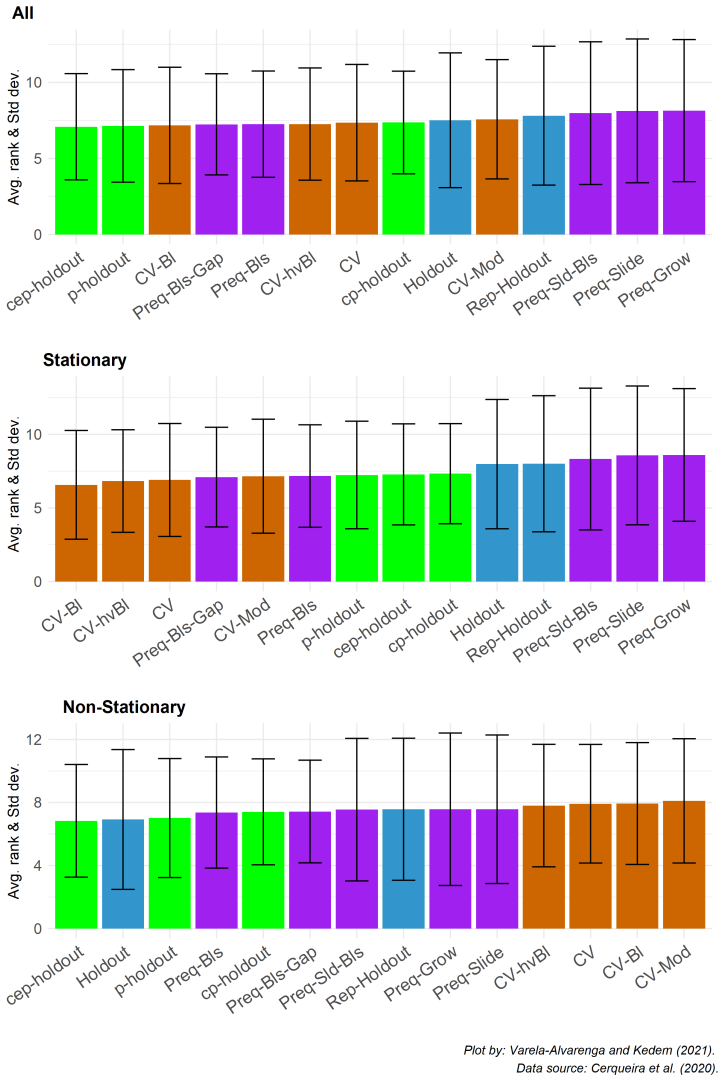


Figure 6.9: Average APAE rank of each validation scheme on 174 real-world time series using the RIDGE learning algorithm and RMSE as the error function. The black line represents -+ 1 standard deviation from the average.

When looking at the results with the MASE as error metric, the `cep-Holdout` was again first place in terms of the smallest rank. But, unlike the RMSE case, it did not provide the smallest log percentage difference. Here, using the `CV-Bl` , the `CV-Bl` and the `p-Holdout` schemes resulted in smaller error bias (Figure B.10). However, these

methods did poorly in terms of the APAE rank (Figure B.9).

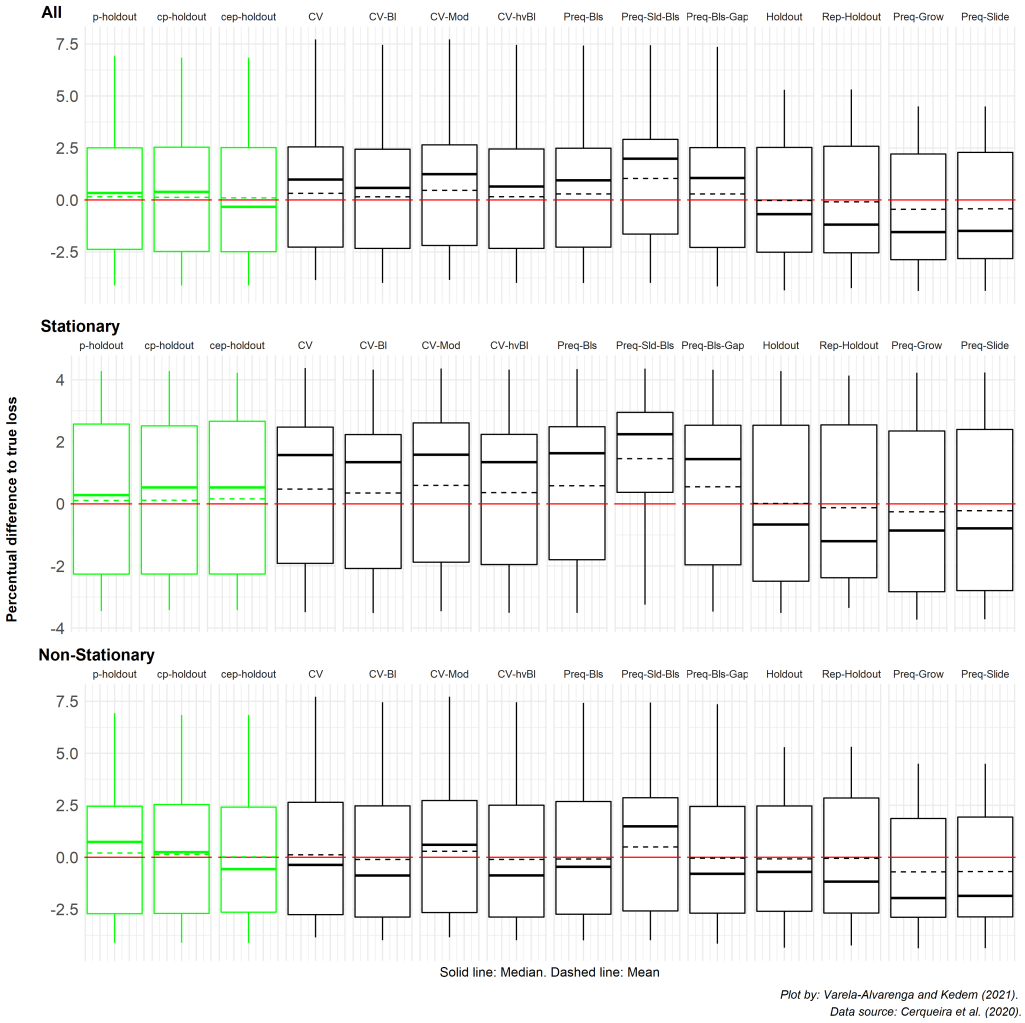


Figure 6.10: Log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to 174 real-world time series using the GLM-Ridge learning algorithm. Values below the zero (red) line represent under-estimations of the error. Conversely, values above it represent over-estimations of the error.

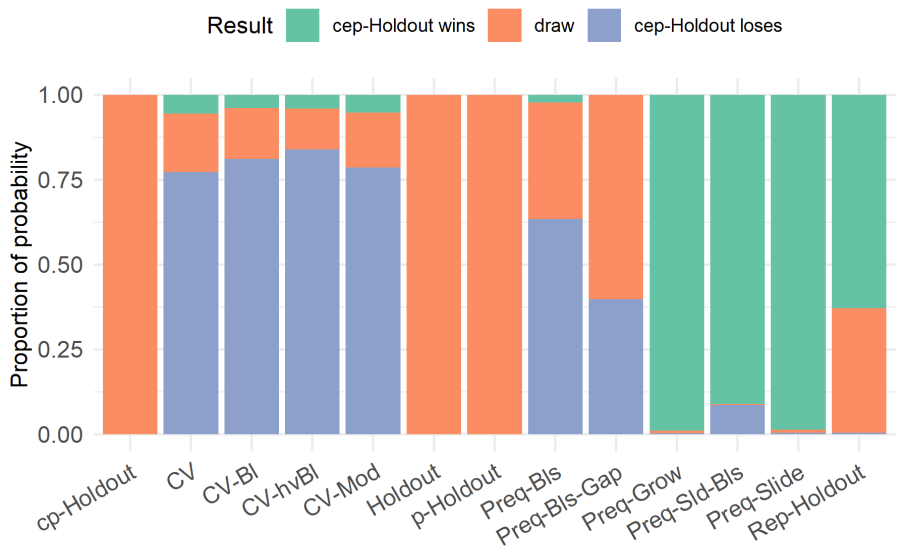


Figure 6.11: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation scheme when applied to the stationary time-series from Cerqueira et al. (2020), with parameters estimated via the GLM-Ridge learning algorithm and error calculated using the RMSE. The probabilities are computed using the Bayes signed-rank test.

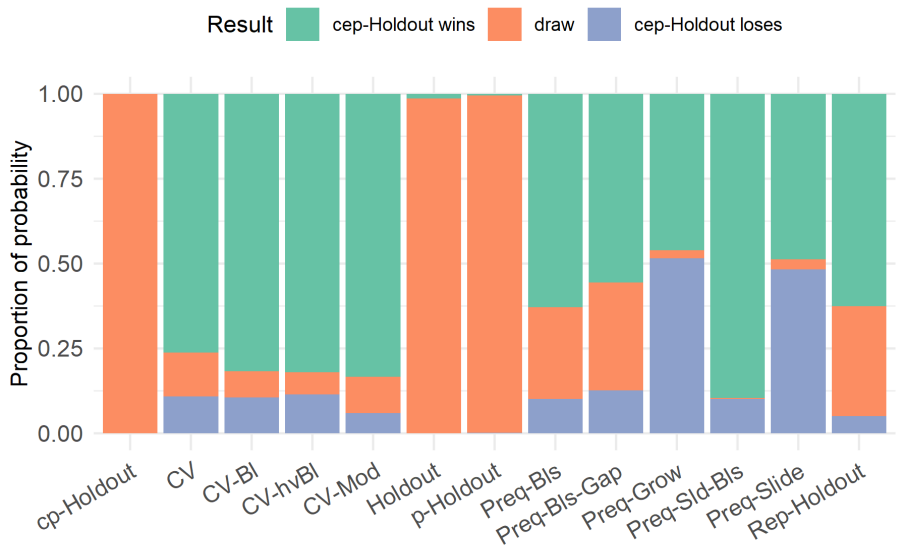


Figure 6.12: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation schemes when applied to the non-stationary time-series from Cerqueira et al. (2020), with parameters estimated via the GLM-Ridge learning algorithm and error calculated using the RMSE. The probabilities are computed using the Bayes signed-rank test.

6.2 Data from the M4 Competition

6.2.1 Results from the RBR learning algorithm

When evaluating the sample of periodic time series taken from the M4 Competition data set, we see that the procedures from the *p-Holdout* family perform quite well. Results using the RBR method indicate that, overall and for non-stationary series, the `cp-Holdout` and the `cep-Holdout` schemes are capable of producing the smallest forecast errors when both the RMSE (Figure 6.13) and the MASE (Figure B.13) error

metrics are used.

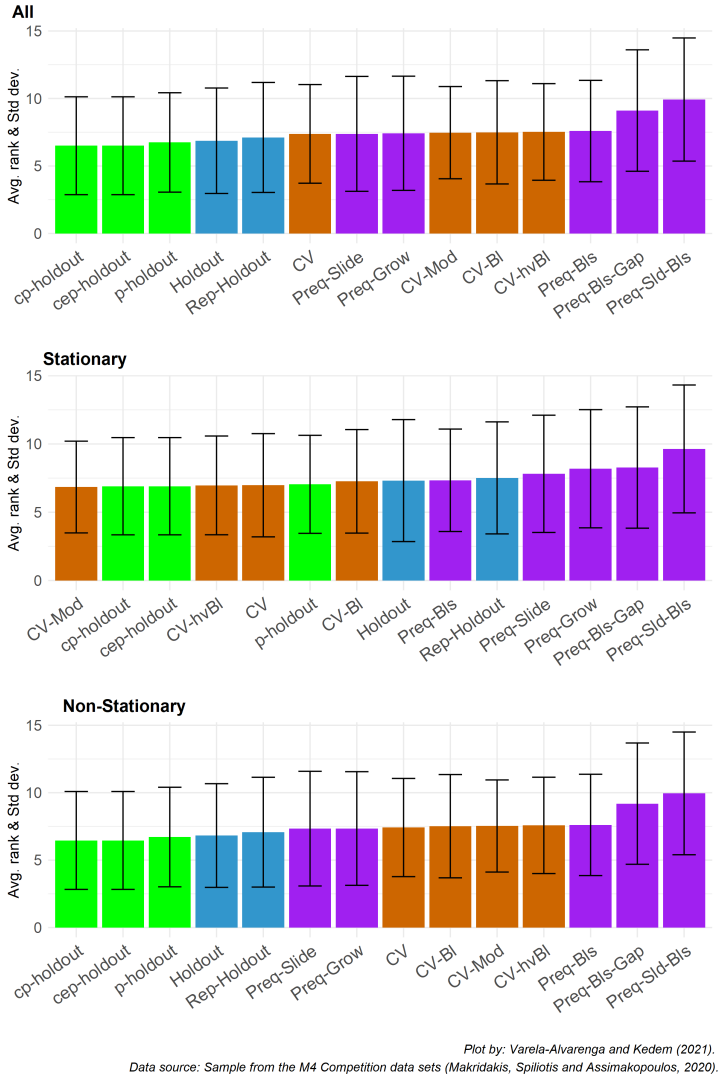


Figure 6.13: Average APAE rank of each validation scheme to the sample of 1,000 time series from the M4 competition using the RBR learning algorithm and RMSE as the error function. The black line represents -+ 1 standard deviation from the average.

For the "Stationary" case, the `CV-Mod` procedure yielded slightly better results in the RMSE case, albeit its error bias was larger (Figure 6.14). When MASE was used, the `CV-Bl` and the `CV-hvBl` placed better in terms of the average rank, but their error bias

was much larger (Figure B.14). For instance, the log percentage for the `cp-Holdout` and `cep-Holdout` were both equal to 0.0997, while the `CV-hvBl` and `CV-Bl` procedures produced an error bias equal to -0.4888 and -0.7591, respectively.

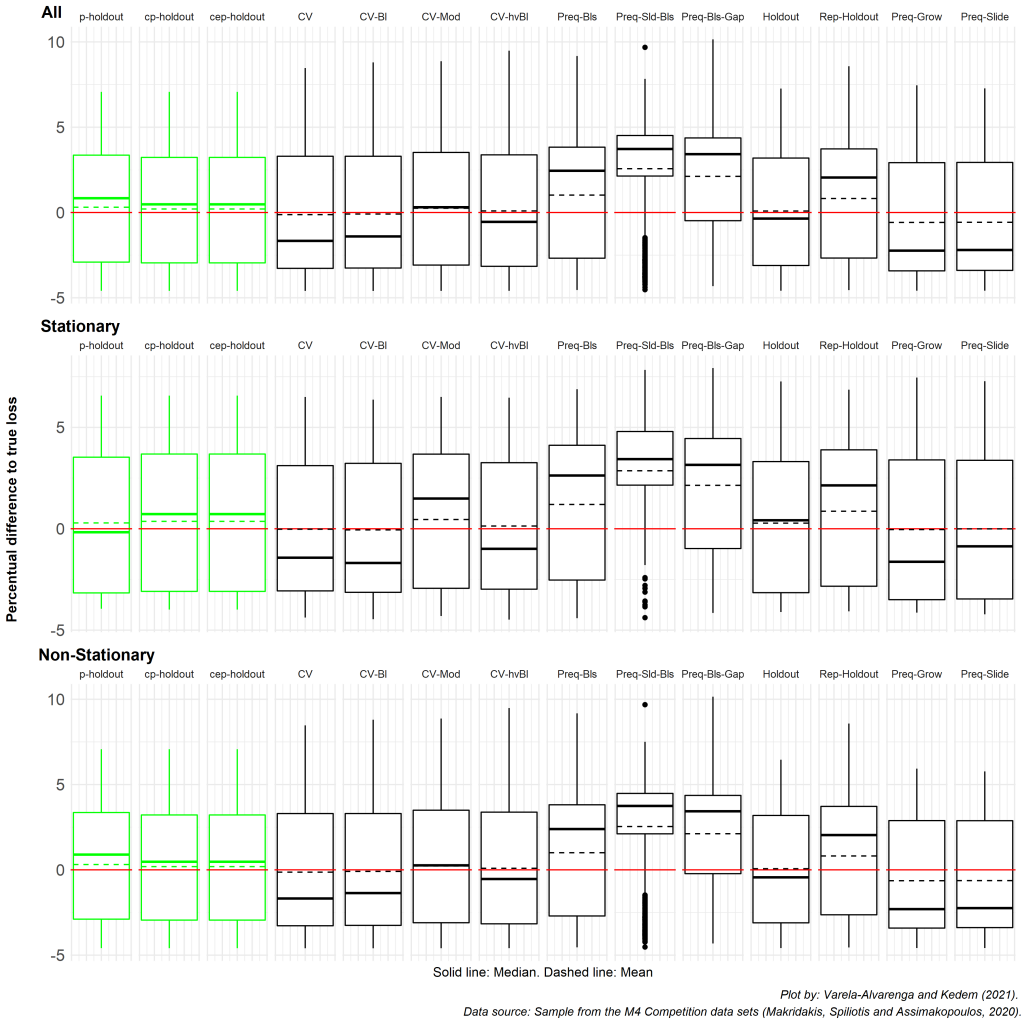


Figure 6.14: Log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the sample of 1,000 time series from the M4 competition using the RBR learning algorithm. Values below the zero (red) line represent under-estimations of the error. Conversely, values above it represent over-estimations of the error.

6.2.2 Results from the RF learning algorithm

When the RF learning algorithm was used with the sample of series from the M4 competition, the procedures in the *p-Holdout* family were unbeatable (according to the ranks based on the APAE metric - Figure 6.15).

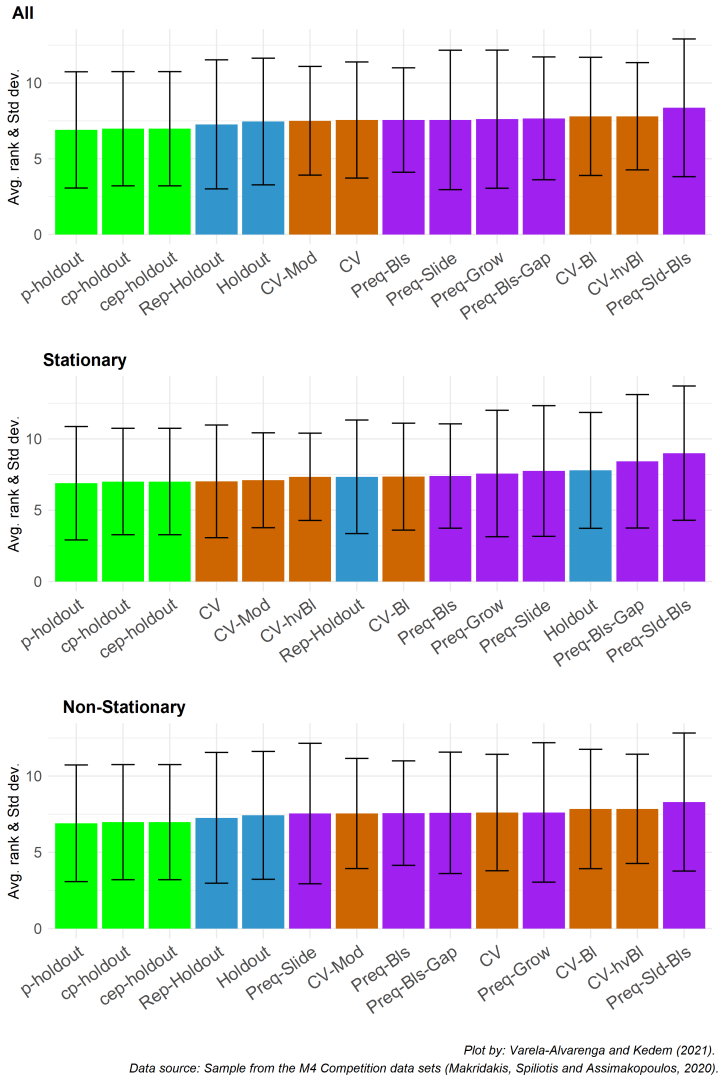


Figure 6.15: Average APAE rank of each validation scheme to the sample of 1,000 time series from the M4 competition using the RF learning algorithm and RMSE as the error function. The black line represents -+ 1 standard deviation from the average.

In particular, the `p-Holdout` scheme dominated all the results when the RMSE metric was used, and only lost to the `cp-Holdout` and `cep-Holdout` in the "Stationary" case with the MASE error (Figure B.17).

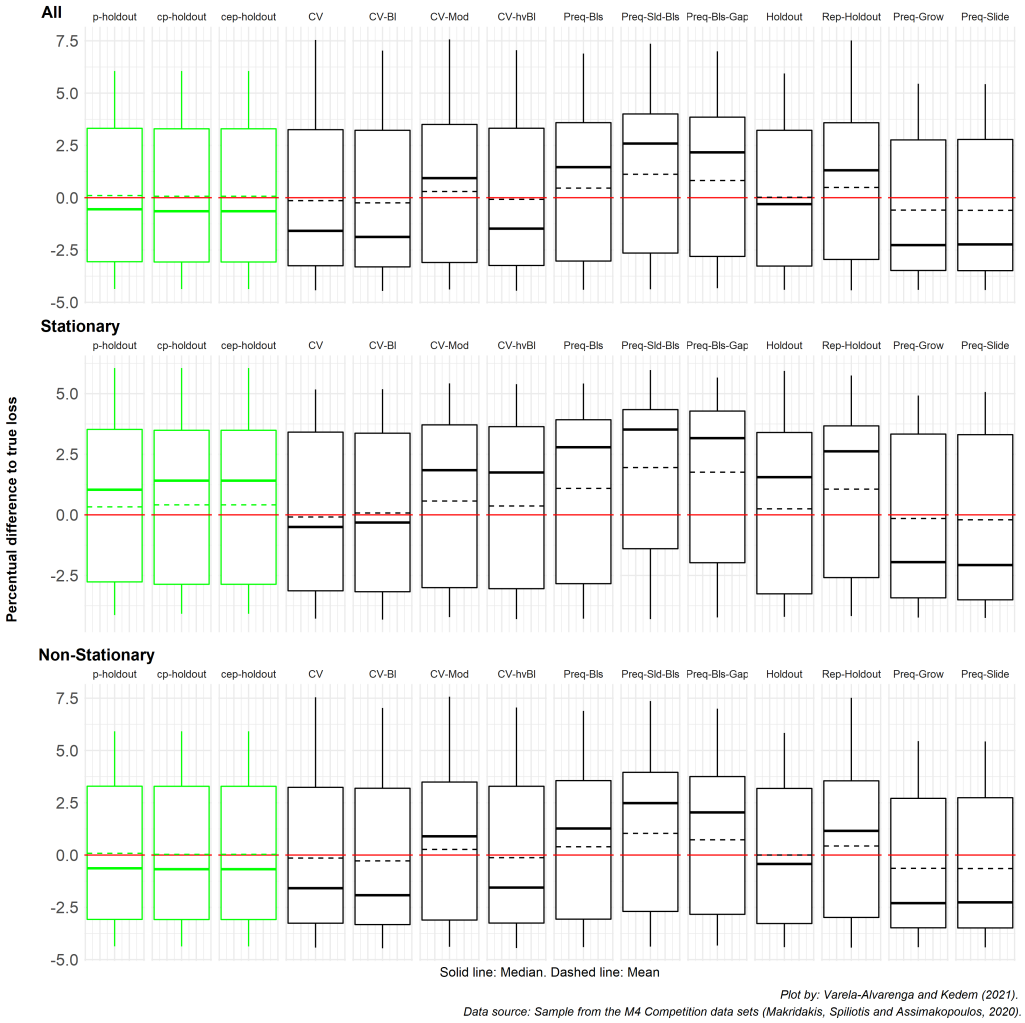


Figure 6.16: Log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the sample of 1,000 time series from the M4 competition using the RF learning algorithm. Values below the zero (red) line represent under-estimations of the error. Conversely, values above it represent over-estimations of the error.

The results regarding the error bias indicate that the `Holdout` performed a little

better than our schemes when we evaluate all series and only the non-stationary ones with the RMSE (Figure 6.16). However, our procedures provided a smaller log percentage difference for non-stationary series under the MASE error (Figure B.18).

6.2.3 Results from the GLM learning algorithm

With the GLM-Ridge regression as the estimation method, the new `cp-Holdout` scheme performed the best in all but stationary cases (according to the average APAE metric), followed by the `cep-Holdout` and the `p-Holdout` schemes (Figures 6.17 and B.21). As for the error bias, the holdout-based schemes provided values very close to zero in both the "All" and "Non-stationary" cases (Figure 6.18).

When evaluating only the 97 stationary series and the RMSE error, the `CV-Mod` scheme yielded the smallest average rank but only the 8th smallest value of the log percentage difference given by the PAE metric (Figure 6.18).

When the MASE was used, overall the `CV-Bl` procedure ended up in first place in terms of the average rank based on the APAE metric (Figure B.21), but had an error bias equal to -0.4889 (Figure B.22). The `cp-Holdout` , which ended up in second place, had a much smaller error (0.0997).

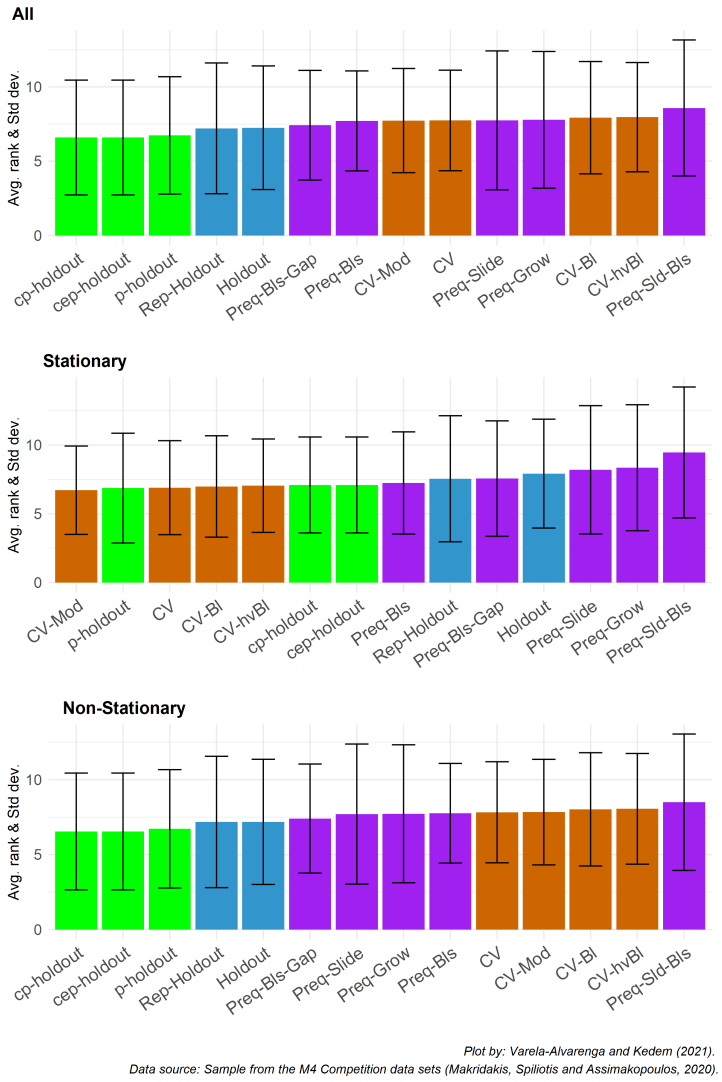


Figure 6.17: Average APAE rank of each validation scheme to the sample of 1,000 time series from the M4 competition using the GLM-Ridge learning algorithm and RMSE as the error function. The black line represents -+ 1 standard deviation from the average.

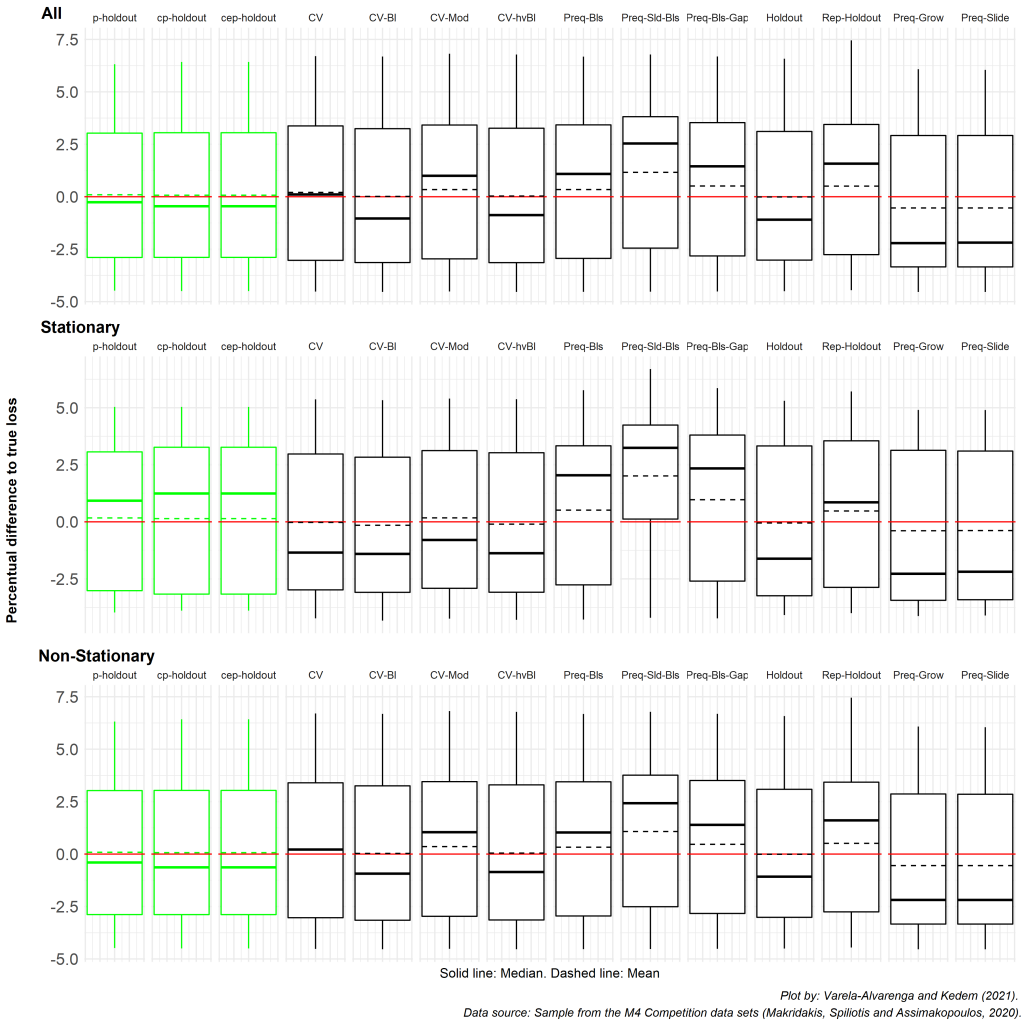


Figure 6.18: Log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the sample of 1,000 time series from the M4 competition using the GLM-Ridge learning algorithm. Values below the zero (red) line represent under-estimations of the error. Conversely, values above it represent over-estimations of the error.

6.3 Monte Carlo Simulation

6.3.1 Results from the RBR learning algorithm

When Bergmeir et al. [21] devised their experimental design, they constrained the estimation methods to allow only for $p = 5$, and expected the models to not fit well data, regardless of the validation scheme used.

Indeed, when we evaluate the sample of periodic time series taken from the Monte Carlo simulation, we see that the methods that previously did well start to break down (Figures 6.19 and B.25).

In the case of the RBR algorithm, we see that this is indeed the case when we evaluate the plots of the error bias (Figures 6.20 and B.26). From them, we see that cross-validation methods provided poor results for the $S3$ data set, and much worse results when integrated seasonal processes were evaluated ($S4$). On the other hand, the schemes in the *p-Holdout* family were the ones that yielded the smallest percentage differences to the true loss in both cases, and for both error measures (RMSE and MASE).

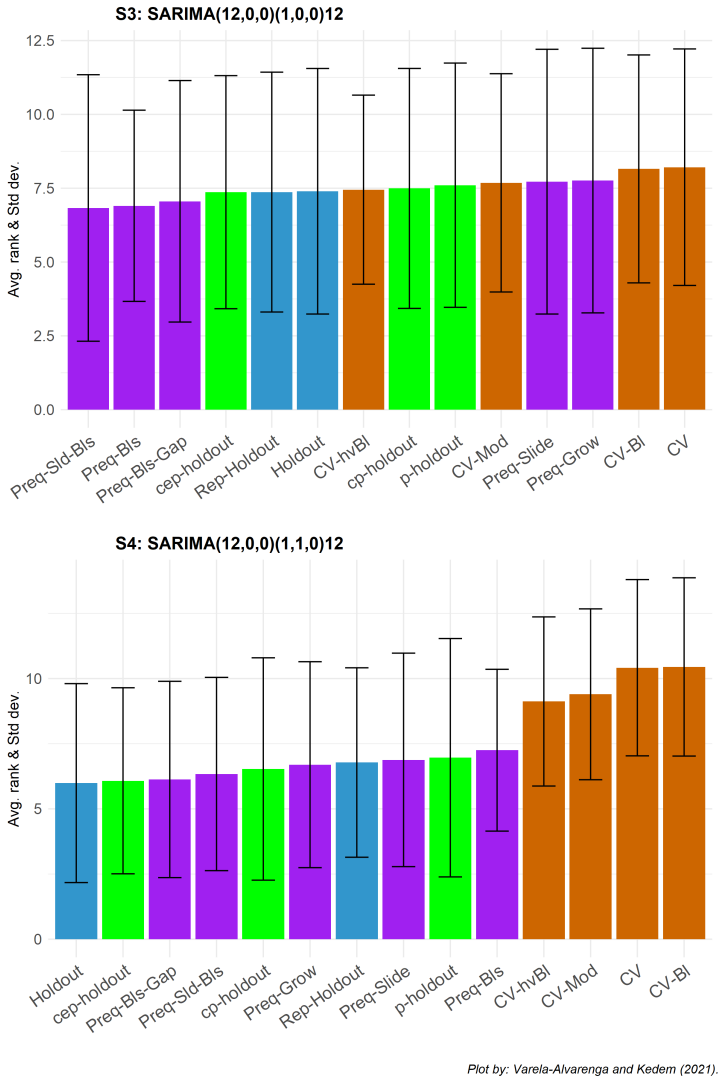


Figure 6.19: Average APAE rank of each validation scheme to the sample of 1,000 simulated time series using the RBR learning algorithm and RMSE as the error function. The black line represents -+ 1 standard deviation from the average.

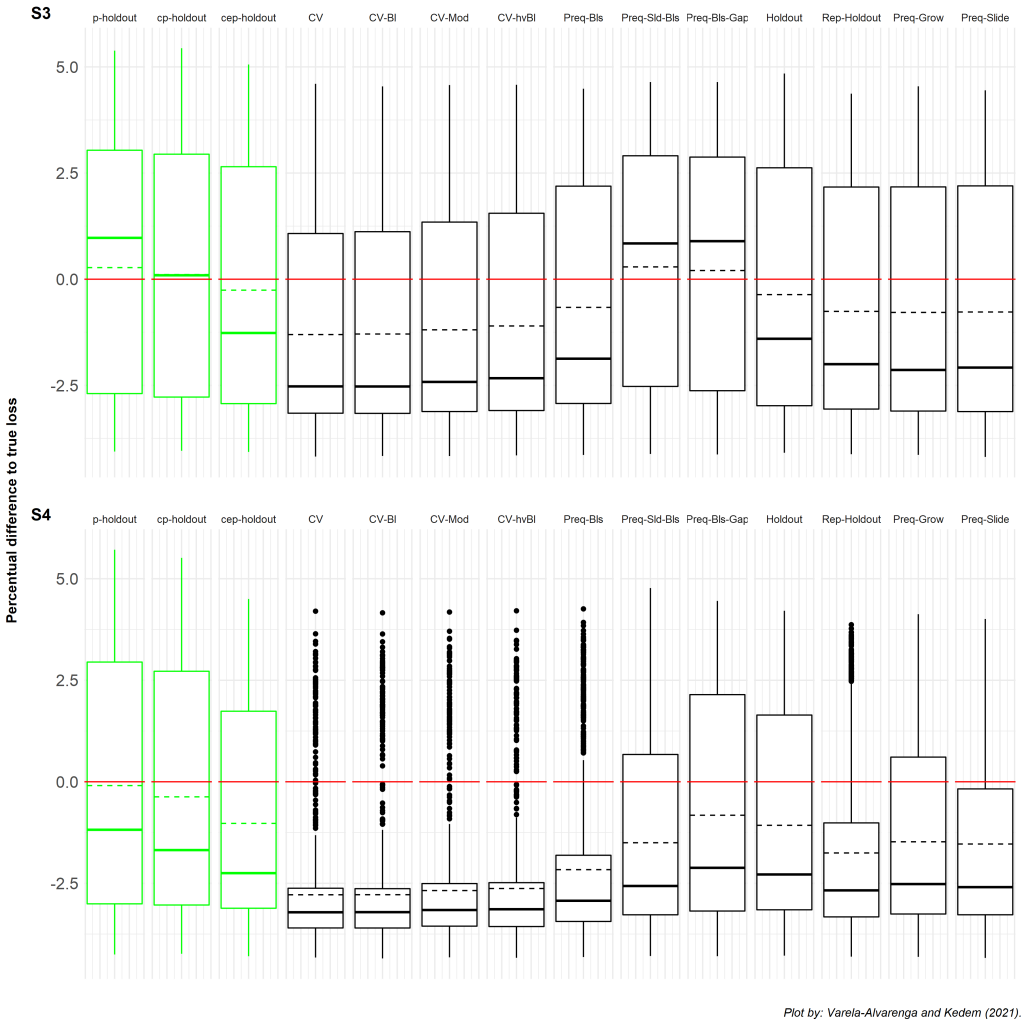


Figure 6.20: Log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the sample of 1,000 simulated time series using the RBR learning algorithm and error calculated using the RMSE. Values below the zero (red) line represent under-estimations of the error. Conversely, values above it represent over-estimations of the error.

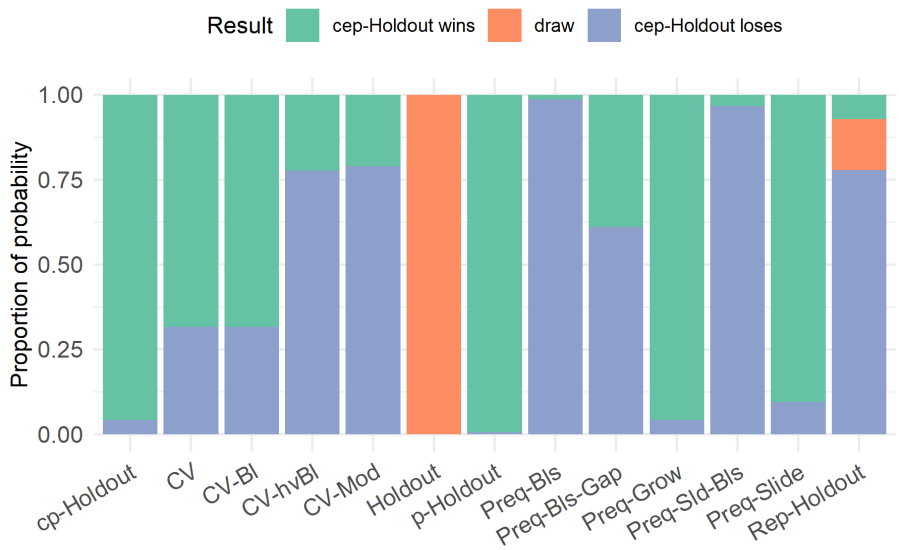


Figure 6.21: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation schemes when applied to the time-series from the $S3$ simulated data set, with parameters estimated via the RBR learning algorithm and error calculated using the RMSE. The probabilities are computed using the Bayes signed-rank test.

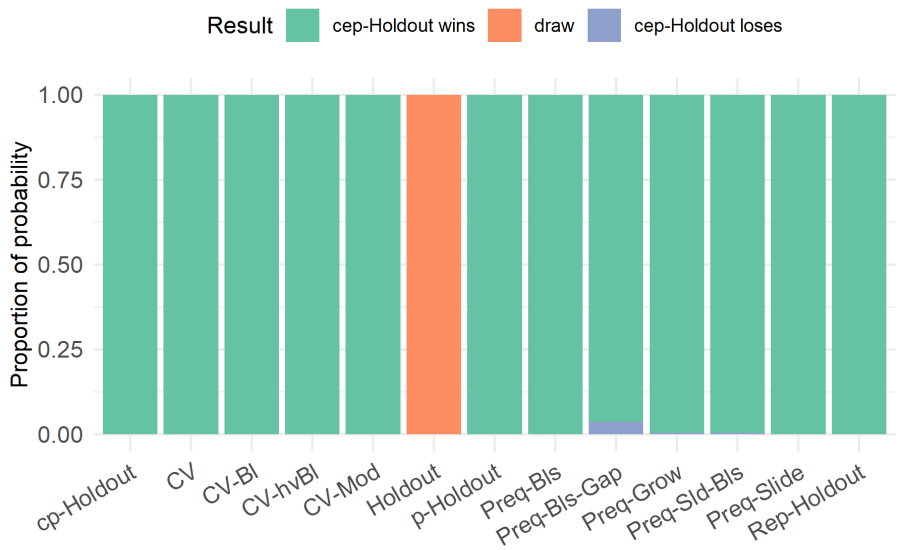


Figure 6.22: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation schemes when applied to the time-series from the $S4$ simulated data set, with parameters estimated via the RBR learning algorithm and error calculated using the RMSE. The probabilities are computed using the Bayes signed-rank test.

6.3.2 Results from the RF learning algorithm

The same problems on the error bias related to constraining the covariate space were observed when the RF learning algorithm was used (Figures 6.24 and B.30). Yet again, the procedures in the *p-Holdout* family were able to mitigate these effects. The `cp-Holdout` scheme provided the smallest percentage difference to the true loss when the RMSE was used in the $S3$ data set, and the second smallest when the $S4$ was used. The procedure that yielded the smaller error bias for $S4$ was the `p-Holdout` scheme (which was the 4th best method in terms of the average rank based on the APAE metric - Figure 6.23).

When evaluating the methods under the MASE error measure, the `Preq-Bls` procedure yielded the smallest error considering the SARIMA$(12,0,0) \times (1,0,0)_{12}$ case. In terms of the average rank, the `CV-hvBl` procedure ended up in first place (Figure B.29), but the boxplot in Figure B.30 shows that the majority of the distribution of its error bias does not cover the zero (red) line. The results for the $S4$ data set (SARIMA$(12,0,0) \times (1,1,0)_{12}$) show that the `cep-Holdout` was able to provide smaller forecasts errors more often (Figure B.29), and with the fourth smallest error bias (Figure B.30). In this metric, the `p-Holdout` provided the smallest log percentage difference to the true loss (-0.0880), followed by the `Preq-Bls-Gap` (0.2515), and the `cp-Holdout` (-0.3104).

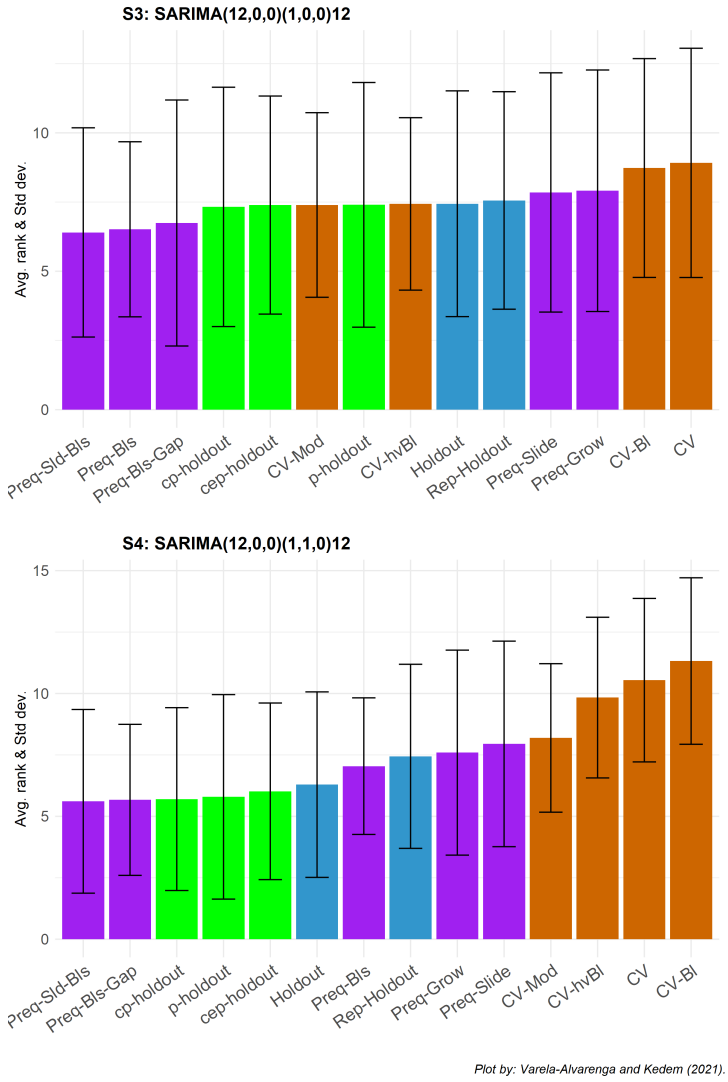


Figure 6.23: Average APAE rank of each validation scheme to the sample of 1,000 simulated time series using the RF learning algorithm and RMSE as the error function. The black line represents -+ 1 standard deviation from the average.

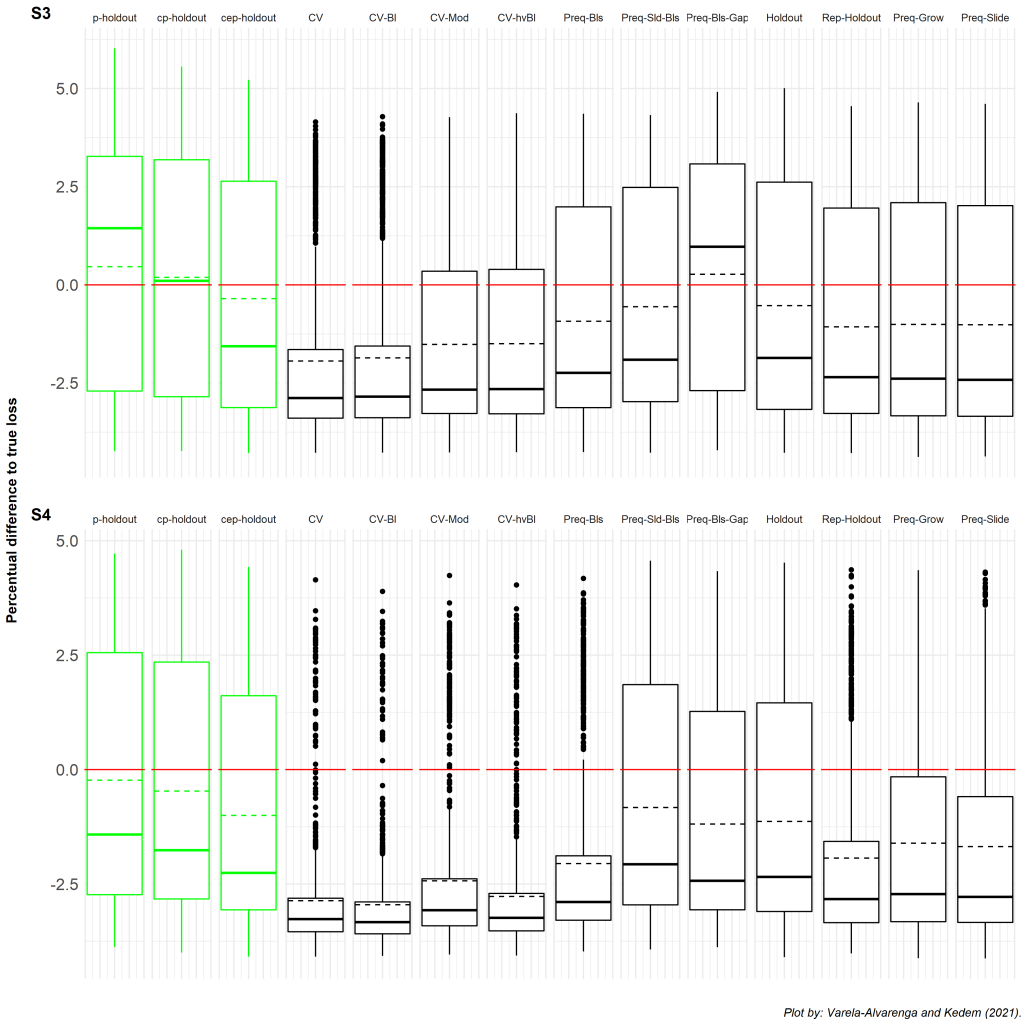


Figure 6.24: Log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the sample of 1,000 simulated time series using the RF learning algorithm and error calculated using the RMSE. Values below the zero (red) line represent under-estimations of the error. Conversely, values above it represent over-estimations of the error.

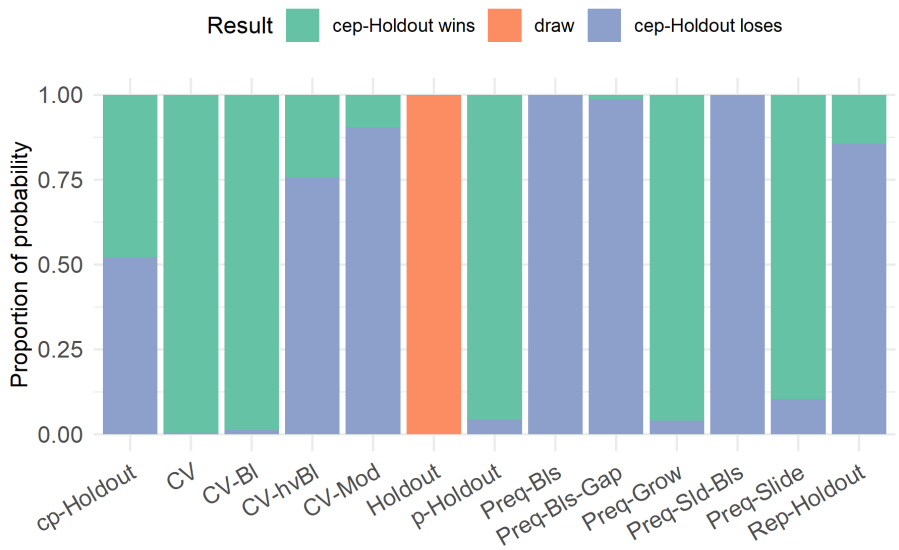


Figure 6.25: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation schemes when applied to the time-series from the $S3$ simulated data set, with parameters estimated via the RF learning algorithm and error calculated using the RMSE. The probabilities are computed using the Bayes signed-rank test.

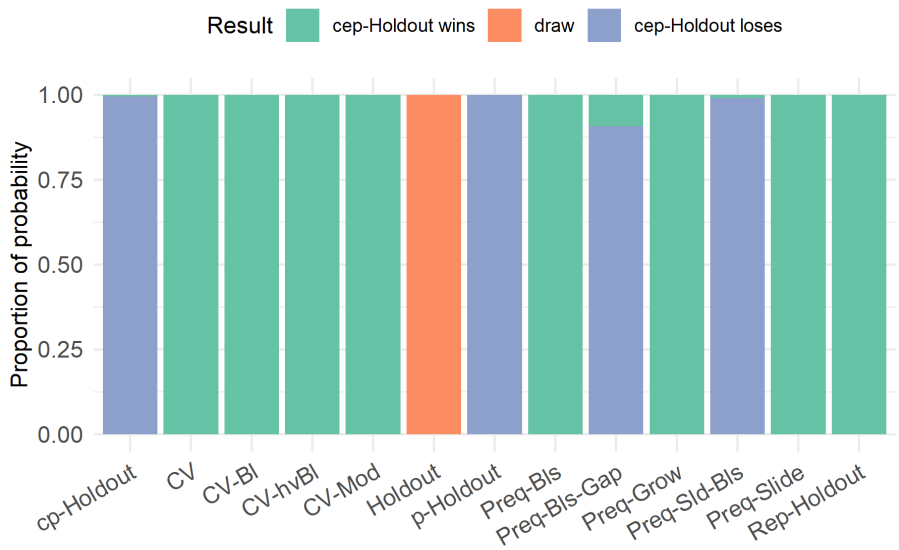


Figure 6.26: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation schemes when applied to the time-series from the $S4$ simulated data set, with parameters estimated via the RF learning algorithm and error calculated using the RMSE. The probabilities are computed using the Bayes signed-rank test.

6.3.3 Results from the GLM learning algorithm

Finally, the last results come from applying the GLM-Ridge regression as learning algorithm to the data sets from our Monte Carlo experiment with constrains on the covariate space. The most interesting result is related to the log percentage difference to the true loss with the $S4$: SARIMA$(12, 0, 0) \times (1, 1, 0)_{12}$ data set. From the bottom plots in Figures 6.28 and B.34, we see that all validation schemes provided poor results, with cross-validation methods being the worse ones. The schemes in the *p-Holdout* family also behaved poorly, despite showing good results for the rank (bottom part of Figures 6.27 and B.33). In this scenario, the `Preq-Grow` scheme was the one that provided the smallest percentage differences in both RMSE and MASE cases.

Much better results were obtained with the $S3$ data set. The `cp-Holdout` was the scheme that yielded the smallest error bias considering both error measures (RMSE - Figure 6.28, and MASE - Figure B.34). However, it did not do well in terms of the average rank based on the APAE metric (Figures 6.27 and B.33).

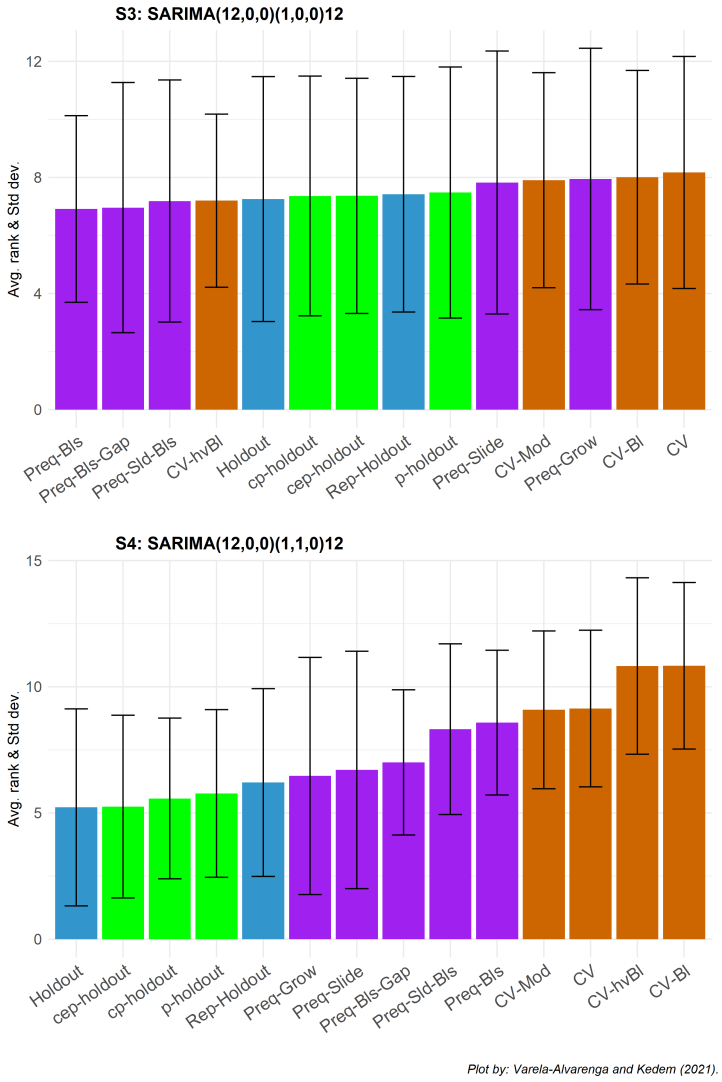


Figure 6.27: Average APAE rank of each validation scheme to the sample of 1,000 simulated time series using the GLM-Ridge learning algorithm and RMSE as the error function. The black line represents -+ 1 standard deviation from the average.

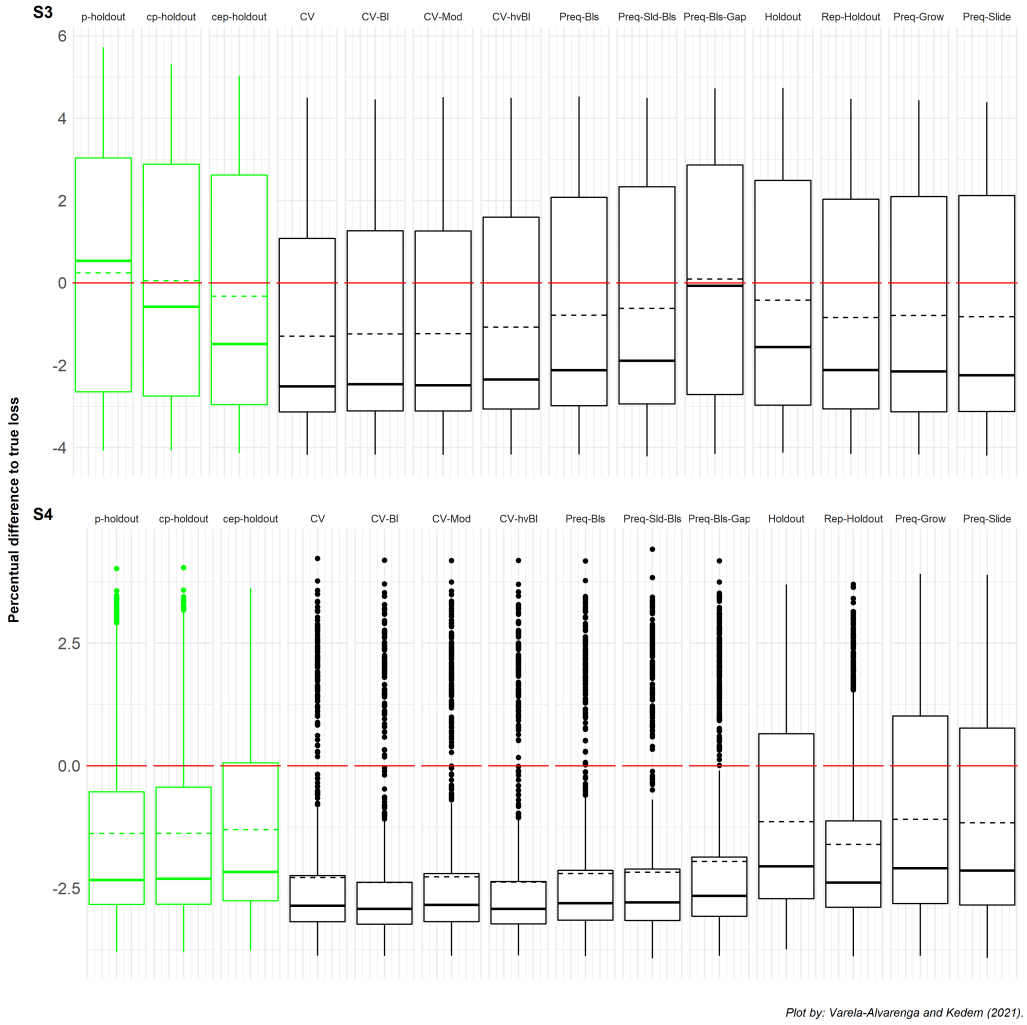


Figure 6.28: Log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the sample of 1,000 simulated time series using the GLM-Ridge learning algorithm and error calculated using the RMSE. Values below the zero (red) line represent under-estimations of the error. Conversely, values above it represent over-estimations of the error.

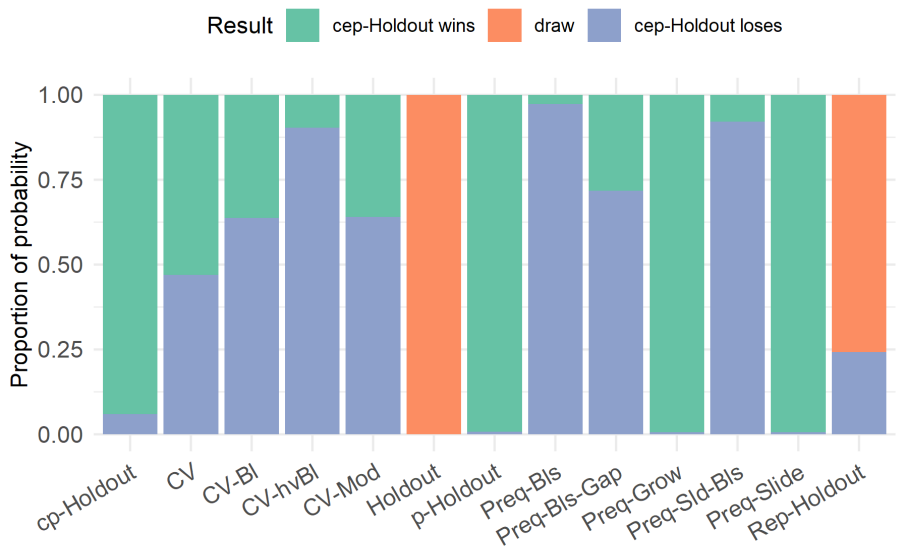


Figure 6.29: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation schemes when applied to the time-series from the $S3$ simulated data set, with parameters estimated via the GLM-Ridge learning algorithm and error calculated using the RMSE. The probabilities are computed using the Bayes signed-rank test.

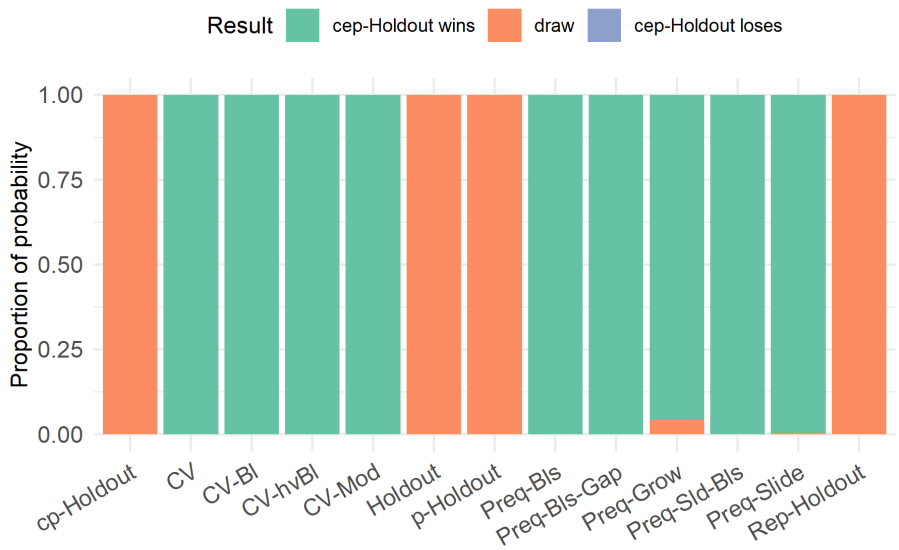


Figure 6.30: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation schemes when applied to the time-series from the $S4$ simulated data set, with parameters estimated via the GLM-Ridge learning algorithm and error calculated using the RMSE. The probabilities are computed using the Bayes signed-rank test.

Chapter 7: Final Remarks

One approach to evaluate forecast accuracy and compare forecasting methods and models is to use data-driven procedures to split the original time series into two (or more) non-overlapping sub-samples, and use one part to fit the model and the other to evaluate its predictive accuracy. These procedures are called "validation schemes."

We can divide the validation schemes usually associated with time-series data into two "main" categories: forward validation (or out-of-sample methods) and cross-validation schemes. The schemes in the former group preserve the time-ordering of the data, but do not use all the available data to "train" the model (i.e., to estimate its parameters), which complicates their implementation in small samples. On the other hand, cross-validation procedures use all available data, but they have to be adapted to deal with dependent data, as the "original" cross-validation schemes assume that the data are independent and identically distributed.

Notwithstanding, all these schemes fail to consider one important characteristic of a time-series, its periodicity. We argue that if the goal is to find a way to obtain more accurate forecasts and to decrease the error bias, one should consider the periodic behavior that a series might present. If a time-series does display such similar behavior over time, we argue that it is possible to obtain validation and test sets that have approximately

the same structure, which would lead to an improvement in model selection and forecast accuracy.

With that in mind, we proposed the `p-Holdout` family of validation schemes. The `p-Holdout` , `cp-Holdout` , and `cep-Holdout` schemes are the three members of this family, and were developed to take into account a series' periodicity. To assess their quality we followed [6]'s approach to evaluating new validation schemes. They said,

> "(...) researchers proposing a new method are interested in the question, whether the new method performs better than the state-of-the-art methods. This is usually determined by the application and comparison of all the methods on a set of benchmarking data or within competitions." (Bergmeir and Benitez, 2012, p. 192)

We evaluated our schemes using the benchmark data sets from Cerqueira et al. [22] and the Monte Carlo approach by Bergmeir et al. [21]. We also added a new simulated data set to see how the validation schemes deal with seasonal integrated processes. The last group of time-series that we evaluated came from the data sets of the M4 Forecasting Competition [11]. With those, we cover all types of data suggested by Bergmeir and Benítez [6].

In the end, we assessed the impact of 14 validation schemes on the forecast accuracy and forecast error bias. Since the out-of-sample error not only depends on the validation schemes, but also on the learning algorithm used, we used three estimation methods. The first one is known as RBR, and it is a rule-based regression based on the M5 model tree algorithm [92, 93]. The second one was the traditional Random Forest algorithm (RF). Lastly, we used a generalized linear model with a Gaussian distribution and a Ridge penalty mixing (GLM-Ridge).

Moreover, we considered two error metrics, the root mean square error (RMSE)

and the mean absolute scaled error (MASE). The former is a metric traditionally used to evaluate forecast accuracy, while the latter allows us to incorporate the seasonality into the calculation. Following Cerqueira et al. [22] (CTM), we wanted to compare the different validation schemes by evaluating how close the generalization error would be to the "ground truth loss." For that, we used the absolute predictive accuracy error (APAE) and the predictive accuracy error (PAE) metrics. The APAE metric evaluates the size of the forecast error of a given validation scheme, and gives us a metric of forecast accuracy. As a complement measure, the PAE metric returns the error bias of a validation scheme - that is, if it is underestimates or overestimates the "true" error.

After assessing all 14 schemes under 18 different scenarios (3 sets of time-series, 3 estimation methods, and 2 error measures), we concluded that our new schemes are computationally inexpensive, improve the forecast accuracy, and greatly reduce the average forecast bias without increasing the variability, specially when applied to non-stationary time series.

More specifically, the `cep-Holdout` validation scheme was the procedure that more often yielded the smallest average forecast error when applied to non-stationary time series (this result was observed in 7 out of the 18 scenarios), followed by the `cp-Holdout` procedure (4 times), and the `p-Holdout` scheme (2 out of 18). The remaining five cases were divided amongst the `Holdout` procedure (3 times), and the prequential grow (`Preq-Grow`) and prequential sliding blocks (`Preq-Slide-Blocks`) schemes. The last two are more computationally expensive and also do not provide large improvements in the error bias. Thus, when considering all aspects, the `cep-Holdout` seems to be a very good data-splitting scheme.

When it comes to the forecast error bias, the results show that our schemes usually tend to be pessimist, in the sense that they provide over-estimation of the errors (i.e., errors greater than zero). However, when assessing the series with integrated seasonality, our procedures tend to be more optimistic and under-estimate the bias.

Lastly, a careful evaluation of the equation for the `Holdout` scheme (Eq. 3.12) shows that it has the exact same structure as the equations for the `p-Holdout` (Eq. 4.4), the `cp-Holdout` (Eq. 4.7), and `cep-Holdout` (Eq. 4.13) schemes. The only difference is in the value of l_v that each procedure uses. By having the same structure, we argue that the theoretical results obtained for the `Holdout` procedure, including the asymptotic properties, proved by West [8] could be extended to the schemes in the *p-Holdout* family. We leave this for our future research.

Part II

A Novel Machine Learning Strategy for Forecasting Model Selection

Chapter 8: Introduction

There is an old saying that goes like this "a good craftsperson never blames their tools." This proverb might be true or not, depending on how one sees the world. What is undeniably true, though, is that selecting the proper tool greatly facilitates one's work and heavily influences the quality of the outcome. This is true for life and also for data analysis.

Sometimes, however, we might be in a situation where we do not know which tool is the best for a given task. This is especially true for data analysis and even more so for time-series forecasting.

When faced with such situations in life, we tend to use our previous knowledge to narrow down a set of tools that might work for a task. The next usual step is to go and try them all. Finally, we select the one that yields the best results in a cost-benefit analysis between the amount of work required and the outcome's quality.

That is the basic idea behind the Box-Jenkins procedure [41][1]. One uses their (usually incomplete) theoretical knowledge to indicate a suitable set of forecast models (i.e.,

[1] In the Box-Jenkins procedure, all model forms are based on a general linear process. Throughout this text, *forecast model*, or simply *model*, might also refer to the non-linear relationships. We assume an automatic distinction by the reader based on context. Furthermore, we distinguish *forecast models* from **forecast methods**, as did Giacomini and White [1]. Whenever we use the entire expression **forecast methods**, we mean the entire methodology to obtain the forecasts, which includes the selected forecast model, the method of parameter estimation and the data used for it.

the mathematical equations that indicate how many variables should be used as inputs and their relationship to the output) for evaluation.

Then, one fits these models using all the available data. Finally, one selects the model form that is both parsimonious (uses the smallest number of input variables) and fits the model well (through some criterion like the AIC – Akaike Information Criteria), and that ideally yields forecasts with sufficiently small forecast errors.

The problem with our analogy, and the reason we wrote the word ”ideally” in the previous sentence, is that the craftsperson is able to evaluate the result of their work throughout and at the end of the process. In contrast, in the Box-Jenkins procedure, one must monitor future cases to evaluate the model's forecast performance. Moreover, the same data that is used to fit and evaluate a model adequateness is also used to obtain its forecasts.

One possible solution to this conundrum is to use data-split techniques. The basic idea behind these procedures is not new. It dates back at least to the 1930s when Wilson [33] used it in conjunction with periodograms to search for hidden periods in a time series. Here, the forecasting models are estimated recursively using parts of the data. Then, their one-period-ahead[2] predictions are calculated and compared with “future uses” (i.e., one-step-ahead observations that are out of the sample used to fit the model). The model specification that maximizes the predictive likelihood is selected, and its final parameters are then estimated using the entire dataset.

It is hard to say if current authors stood on Wilson's shoulders to develop the “learning procedures” used in the Machine Learning literature, but it is undeniable that there are

[2]It is worth mentioning that Wilson [33] calculated multi-step-ahead forecasts.

striking similarities between the two approaches.

A learning procedure can be summarized as an approach that uses data-split techniques called *validation schemes* to find the forecasting model that yields the smallest average out-of-sample prediction error and the hyperparameters associated with it according to the Empirical Risk Minimization principle (ERM - see Chapter 9).

In Machine Learning, the parameters (or weights) of the forecasting models are estimated (learned) using methods like backpropagated gradient-descent for feedforward neural networks or rule-based regression algorithms. However, such methods have been labeled *black-box* algorithms, since "we know what goes in, we know what comes out, but very few understand what happens in-between" [99, p. 35]. The authors from the previous quote also argue that,

> "If the data do not contain problematic sampling biases, then building the black boxes – the predictive models that mathematically elucidate a relationship between the outcome and the predictors – can significantly benefit from the large amount of information available about the underlying population. Exploratory statistical methods – restricted cubic splines[100] for example – can help empirically derive the functional form of a predictor within a model which may be difficult to obtain with moderate to small sample sizes. Big data, in short, with good statistical methodology, help to find the form of a mathematical relationship which can usefully sit within the black box." (Kuhn and Johnson, 2014, p. 36)

However, we often see in the news [101] and in Academia [102, 103] cases of sampling bias and the problematic outcomes yielded by machine learning algorithms when that happens. Granted that most of those cases lie *outside* the time series context, but we argue that we still need to be careful with these black box methods, and that hybrid methods that combine aspects from both the machine learning and the statistics literature - like the ones that won the last two M competitions [12, 24] - should be preferred.

In light of this discussion, we introduce a new machine learning strategy for forecasting model selection. One that uses a modification of the prequential sliding window forward validation scheme (see Section 3.2.4) - that we call *rolling sample* validation scheme - and results from the theory behind generalized linear models (GLM) with partial likelihood estimation developed by Kedem and Fokianos [71, 72].

We call our new strategy the *generalized and rolling sample* method - or *GEARS*, for short - since it combines the benefits of the validation schemes and the GLM approach. With the latter, it is easy to accommodate many functional forms for the forecasting methods, including those with covariates and interaction terms. It is so simple to use that we have developed a web application where anyone can upload their data set and obtain forecasts for univariate and multivariate time series in a manner of seconds using statistically sound methods.

We apply the GEARS strategy to the 100,000 time series from the M4 Forecasting Competition and compare its results against the other methods submitted to competition. We had the best results in 8,750 cases out of the 100,000, and the method that won the competition had better results in fewer than 7,300 series.

Moreover, traditional statistical approaches like VAR, State Space, or cointegration methods are often overly complicated and require many steps until a forecasting model is finally ready to be estimated. One of the advantages of the GEARS strategy over those methods is its simplicity in dealing with multivariate series. Our approach allows us to estimate simple models with one covariate as well as more complex model formulations that include covariates with variable and fixed lags, quadratic terms, and interaction terms. The accuracy of the forecasts obtained with GEARS was far superior than the one

observed for the predictions obtained using the ARIMA method.

In Chapter 9 we go over the details of the learning procedure in machine learning and how it can be used to selected the best forecasting model. We introduce the GEARS strategy in Chapter 10. In Chapter 11 we briefly describe the M4 Competition data set and the details related to the partial likelihood estimation of generalized linear models, as developed by Kedem and Fokianos [71]. We display the results of applying GEARS to the 100,000 time series in Chapter 12. Finally, we present our final remarks in Chapter 13.

Chapter 9: The Learning Procedure in Machine Learning

For $t = 1, \ldots, T$, let $\{Y_t\}$ be a time series of interest, and $\{y_t\}$ its observed values. We define the covariate process $\mathbf{Z}_{t-1}$ as:

$$\mathbf{Z}_{t-1} \equiv (Z_{(t-1)1}, \cdots, Z_{(t-1)m})$$

The observed values of this process are $\mathbf{z}_t$, and in the ML literature this is known as the m-dimensional vector of features used to predict the desired output. $\mathbf{Z}_{t-1}$ can also contain past values of the response variable Y_t.

A typical way of representing the relationship between $\{y_t\}$ and the past values of the covariates is:

$$y_t = g(\mathbf{z}_{t-1}, \boldsymbol{\theta}) + \varepsilon_t \tag{9.1}$$

where ε_t is a shock or disturbance term, $\boldsymbol{\theta}$ is a parameter vector, and

$$g(\mathbf{z}_{t-1}, \boldsymbol{\theta}) = E_{\boldsymbol{\theta}}[Y_t | \mathbf{Z}_{t-1}] \tag{9.2}$$

Here, $g(\cdot)$ could be any function: linear, nonlinear, or nonparametric, but we often

do not know which one. The forecasting problem, then, can be stated in terms of $g(\cdot)$, and it will the same in both the Statistics field and the Machine Learning field: how do we relate $g(\cdot)$ to the covariates?

In statistics, the definition of $g(\cdot)$ often depends on the type of Y variable that we are evaluating (continuous, binary, count, etc.), how it behaves, and on the characteristics of $\mathbf{Z}$ (univariate, multivariate). The decision to look at the type, behavior, and characteristics of the data is important here because the statistical properties of the data points us in the direction of which estimation method [STAT] - learning algorithm/learner [ML] - we should use (non-linear least squares, Random Forests, rule-based regression, to name a few).

In machine learning, we search for a model specification through a process called "learning procedure", which has two goals [104]:

- Structural identification: the algorithm needs to choose among a parametric family of model specifications $f : \mathbf{Z}_{t-1} \mapsto Y_t$ the one that gives a good approximation of the unknown function $g(\cdot)$;
- Parametric identification : within the family $f(\cdot)$, the algorithm needs to estimate on the basis of the training set, $\mathcal{D}$, the parameters $\boldsymbol{\theta}_{\mathcal{D}}$ which best approximates g.

In the case where machine learning algorithms are used to accomplish the above tasks, the learning procedure consists of two nested loops: i) an inner parametric identification loop which searches for the best parameter vector within a model specification; ii) an external structural identification loop which goes through different model specifications and returns the one that yields the smallest forecast error.

For each f in the set $\mathbb{G}$ of all possible model specifications, the parametric identification is done according to the Empirical Risk Minimization (ERM) principle where

$$\hat{\boldsymbol{\theta}}_{\mathcal{D}} = \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta}}{\arg\min}\ \overline{\text{err}}(\boldsymbol{\theta}_{\mathcal{D}}) \tag{9.3}$$

where $\overline{\text{err}}(\theta_{\mathcal{D}})$ is the *training error*, calculated as

$$\overline{\text{err}}(\theta_{\mathcal{D}}) \equiv \frac{1}{\mathbf{Card}(\mathcal{D})} \sum_{(\mathbf{z}_{t-1}, y_t) \in \mathcal{D}} \ell\left[y_t, f(\mathbf{z}_{t-1}, \boldsymbol{\theta}_{\mathcal{D}})\right] \tag{9.4}$$

where ℓ is a loss function selected by the analyst and $\mathbf{Card}(\mathcal{D})$ is the cardinality of the training set, usually described as N.

Ultimately, we are interested in finding the forecasting model $\hat{g}$ with the "best" generalization performance. That is, we aim to assess its capacity to produce accurate forecasts using new and independent data. Such performance can be measured by the test error,

$$\mathcal{L}_{\mathcal{D}} \equiv E_{\mathbf{Z}, Y \in \mathcal{D}_{test}}\left[\ell\left(Y, \hat{g}(\mathbf{Z}, \boldsymbol{\theta}_{\mathcal{D}})\right) | \mathcal{D}, \boldsymbol{\theta}_{\mathcal{D}}\right] \tag{9.5}$$

where $\mathcal{D}_{test}$ refers to the test set. We omitted the t subscripts for easiness.

A related quantity is the estimated test error,

$$\mathcal{L} \equiv E_{\boldsymbol{\theta}}\left[\ell\left(Y, \hat{g}(\mathbf{Z}, \boldsymbol{\theta})\right)\right] = E\left[\mathcal{L}_{\mathcal{D}}\right] \tag{9.6}$$

One could try to use the training error to estimate the test error, but $\overline{\text{err}}$ consistently

decreases with model complexity, and may reach zero if we increase the model complexity enough (Figure 9.1). But, selecting a model with zero training error does not mean that the generalization performance in a new data set will be any good. Usually, those models overfit to the training data and generalize poorly [74, p. 221]. Hastie et al. [74, chapter 7] argue that Eq.(9.5) leads to a slightly larger mean absolute deviation. Their results show [74, p. 257] that, in practice, better estimations are obtained using the expected test error $\mathcal{L}$ with validation schemes.

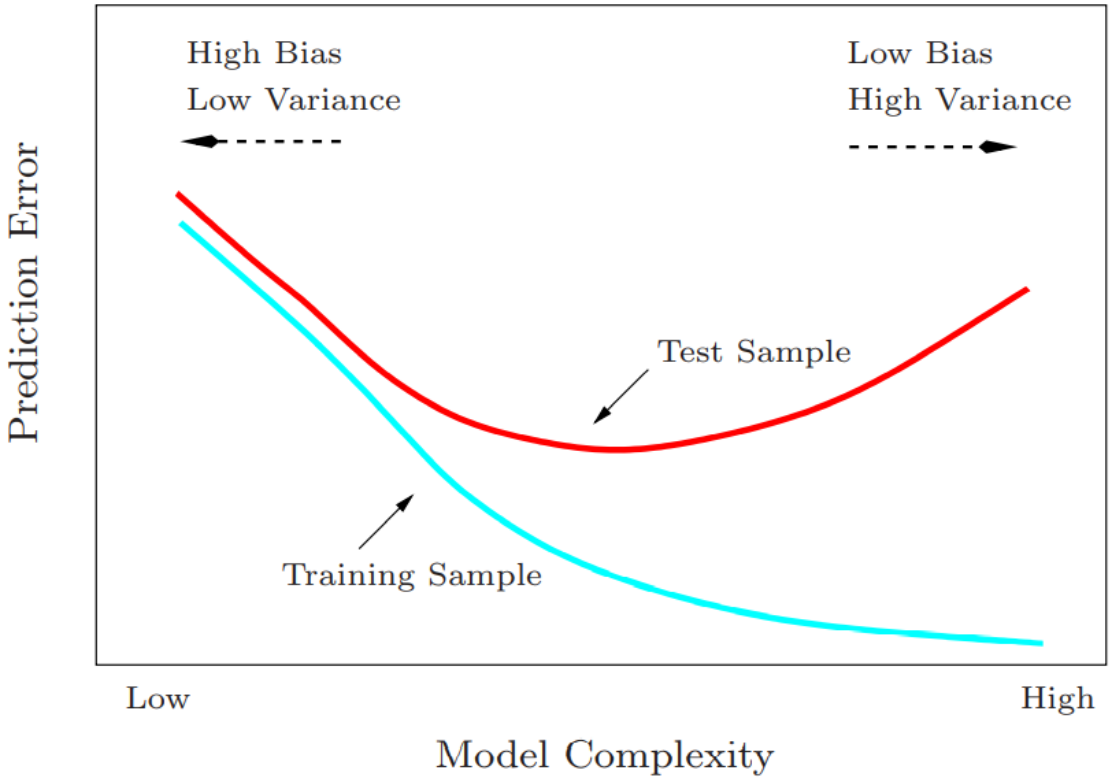


Figure 9.1: Example of the expected test and expected training error curves as a function of model complexity. Image source: Hastie et al. [74, p. 38].

In a data-rich environment, we can partition the training set into an estimation set

and a validation set at each of the k folds. For $i = 0, \ldots, k-1$, the sets $\mathcal{I}_i^e$ and $\mathcal{I}_i^e$ are the sets of indexes from $\{1, \ldots, N\}$ that indicate which observations will form the estimation and validation sets at the i-th fold: $\mathcal{D}\left(\mathcal{I}_i^e\right)$ and $\mathcal{D}\left(\mathcal{I}_i^v\right)$, respectively. Finally, we can define a general form for a validation scheme, $\mathcal{V}(\mathcal{D}; k)$, in those terms:

$$\mathcal{V}(\mathcal{D}; k) \equiv \left\{ \left(\mathcal{I}_i^e, \mathcal{I}_i^v\right) \middle| \, \mathcal{I}_i^e \text{ and } \mathcal{I}_i^v \subseteq \{1, \ldots, N\}, \; \mathcal{I}_i^e \cap \mathcal{I}_i^v = \emptyset \right\}_{i=0}^{k-1} \tag{9.7}$$

Using the general form of a validation scheme from Eq.(9.7), we can define an empirical estimator of $\mathcal{L}$ by taking the average of the mean out-of-sample losses over all k splits [23]:

$$\hat{\mathcal{L}}(\mathcal{D}, \mathcal{V}, f) \equiv \frac{1}{k} \sum_{i=0}^{k-1} \frac{1}{\mathbf{Card}\left(\mathcal{I}_i^v\right)} \sum_{(\mathbf{z}, y) \in \mathcal{D}\left(\mathcal{I}_i^v\right)} \ell \left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}\right) \right] \tag{9.8}$$

where the forecast model f belongs to the set of all possible model formulations $\mathbb{G}$, and $\mathbf{Card}\left(\mathcal{I}_i^v\right)$ is the cardinality of the i-th validation index set. In essence, the above equation returns an estimate of the out-of-sample prediction error and can be used to select $\hat{g}(\cdot)$.

If one uses a *winner-takes-all* approach to evaluate between all possible $f \in \mathbb{G}$ models, then:

$$\hat{g} = \underset{f \in \mathbb{G}}{\arg\min} \, \hat{\mathcal{L}}(\mathcal{D}, \mathcal{V}, f) \tag{9.9}$$

Here is a pseudo-code for the *winner-takes-all* approach:

1. Structural identification loop: for each f in $\mathbb{G}$

Parametric identification loop:

a) for each $i = 0, \ldots, k-1$

$$\hat{\boldsymbol{\theta}}^{f}_{\mathcal{D}\left(\mathcal{I}_i^e\right)} = \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}^f}{\arg\min}\ \frac{1}{\mathbf{Card}\left(\mathcal{I}_i^e\right)} \sum_{(\mathbf{z},y)\in\mathcal{D}\left(\mathcal{I}_i^e\right)} \ell\left[y, f\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}\right)\right]$$

b) calculate $\hat{\mathcal{L}}(\mathcal{D}, \mathcal{V}, f)$ using $\hat{\boldsymbol{\theta}}^{f}_{\mathcal{D}\left(\mathcal{I}_i^e\right)}$

2. Model selection: $\hat{g} = \arg\min_{f\in\mathbb{G}} \hat{\mathcal{L}}(\mathcal{D}, \mathcal{V}, f)$

3. Final parametric identification:

$$\hat{\boldsymbol{\theta}}^{\hat{g}}_{\mathcal{D}} = \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}^f}{\arg\min}\ \frac{1}{N} \sum_{(\mathbf{z},y)\in\mathcal{D}} \ell\left[y, \hat{g}\left(\mathbf{z}, \hat{\boldsymbol{\theta}}_{\mathcal{D}}\right)\right]$$

4. The final forecasting model is : $\hat{g}(\boldsymbol{z}_t, \hat{\boldsymbol{\theta}}^{\hat{g}}_{\mathcal{D}})$

The "parametric identification" part is done differently from what is shown in Eq.(9.3) if one uses a statistical procedure. Instead of obtaining $\hat{\boldsymbol{\theta}}_{\mathcal{D}}$ by trying different values (the attempts can either be based on a rule or not) that minimize the training error, there are estimation methods that depend on the maximization of the likelihood. If it is possible to obtain a form for the likelihood function, statistical procedures like iterated reweighed least squares tend to be more computationally efficient than traditional machine learning methods. This happens because, for the latter, the complexity of the optimization depends on the form of $f(\cdot)$ and the optimization problem may very well be an NP-hard problem[1]

[1] In theoretical computer science, and in mathematical complexity theory, nondeterministic polynomial (NP) time problems is a class of problems for which the solutions can be checked in polynomial time by a nondeterministic Turing machine. They are opposed to the class of P problems, which can be solved by a deterministic Turing machine in polynomial time. An NP-hard problem is a part of the NP set that contains its hardest problems.

[104]. Moreover, statistical methods might be preferred to traditional machine learning algorithms because these are sometimes deemed as "black boxes," as discussed in the previous chapter.

For the above reasons, we chose to use Kedem and Fokianos's methodology [71, 72] as our "learning algorithm." By taking advantage of the GLM formulation, we are able to accommodate several types of (linear) model specifications while speeding up the computations.

Chapter 10: The new GEARS strategy

10.1 The Basic Idea

The idea behind the GEARS strategy is to train our model by breaking down the time series in sections of length s, taking the last ℓ sections and fitting/predicting the model at each of these sections. Then, we use the prediction errors to calculate the mean absolute deviation (MAD) and select the covariates that minimize MAD. Once we have done it, we will take the estimated coefficients from the last fitted model and use it to predict out-of-sample values.

To clarify this idea, say we have y_t, a response variable for $t = 1, 2, \ldots$, and a vector of covariates $\mathbf{z}_t = [\ x_{t,1}\ \cdots\ x_{t,k}]^\top$, which might include a constant, past values of y, and/or other exogenous covariates. The main problem is that we observe $y_1, \ldots, y_n$ and $\mathbf{z}_1, \ldots, \mathbf{z}_n$, and want to predict y_{n+1}. To do so, let us define the relationship between y and $\mathbf{z}$ to be the following:

$$y_{t+1} \equiv \boldsymbol{\beta}^\top \mathbf{z}_t + \varepsilon_t; \qquad t = 1, \ldots, n-1$$

Now, suppose $n = 100$, $s = 12$, $\ell = 20$, and take one of the possible combinations of the k covariates. Then:

1. Fit $y_{t+1} = \boldsymbol{\beta}^\top \mathbf{z}_t + \varepsilon_t,\ t = 68, \ldots, 68 + 19 = 87.$

Get the prediction $\hat{y}_{89} = \hat{\boldsymbol{\beta}}^\top \mathbf{z}_{88}$. Get the prediction error $y_{89} - \hat{y}_{89}$.

2. Fit $y_{t+1} = \boldsymbol{\beta}^\top \mathbf{z}_t + \varepsilon_t,\ t = 69, \ldots, 69 + 19 = 88.$

Get the prediction $\hat{y}_{90} = \hat{\boldsymbol{\beta}}^\top \mathbf{z}_{89}$. Get prediction error $y_{90} - \hat{y}_{90}$.

3. Fit $y_{t+1} = \boldsymbol{\beta}^\top \mathbf{z}_t + \varepsilon_t,\ t = 70, \ldots, 70 + 19 = 89.$

Get the prediction $\hat{y}_{91} = \hat{\boldsymbol{\beta}}^\top \mathbf{z}_{90}$. Get $y_{91} - \hat{y}_{91}$.

$\vdots$

12. Fit $y_{t+1} = \boldsymbol{\beta}^\top \mathbf{z}_t + \varepsilon_t,\ t = 79, \ldots, 79 + 19 = 98.$

Get the prediction $\hat{y}_{100} = \hat{\boldsymbol{\beta}}^\top \mathbf{z}_{99}$. Get the prediction error $y_{100} - \hat{y}_{100}$.

From the 12 prediction errors, get

$$\text{MAD} = \frac{1}{12} \sum_{i=89}^{100} |y_i - \hat{y}_i|$$

Do this for all possible combinations of k covariates and select the one that minimizes the MAD (other measures like the MSE, MASE, and sMAPE can also be used). Then, finally, get $\hat{\boldsymbol{\beta}}$ from No. 12 and set

$$\hat{y}_{101} = \hat{\boldsymbol{\beta}}^\top \mathbf{z}_{100} \tag{10.1}$$

Observe that this $\hat{\boldsymbol{\beta}}$ was estimated using the observations $y_{80}, \ldots y_{99}$ and $\mathbf{z}_{79}, \ldots, \mathbf{z}_{98}$, and

provided the **out-of-sample** residual $y_{100} - \hat{y}_{100}$ (which we are calling *prediction error*), which was used to select the best model. Had we used $y_{81}, \ldots y_{100}$ and $\mathbf{z}_{80}, \ldots, \mathbf{z}_{99}$ to estimate $\boldsymbol{\beta}$, we would have gotten the **in-sample** residual $y_{100} - \hat{y}_{100}$, and should not use this to select the best model. If we use this in-sample residual in selecting the best set of covariates this would defeat the purpose of training the model to select the best $\boldsymbol{\beta}$.

An R package was written for the GEARS strategy. It can be installed using the following piece of code,

Listing 10.1: R code to install the GEARS package.

```
# install.packages("devtools")
library(devtools)

# Install the GEARS package

## Access Token:
GITHUB_PAT <- "b9b7b8b9d384ff89000d1ba40cb0d2e761c273b3"
install_github("gu-stat/gears", auth_token = GITHUB_PAT)

## Call the package
library(gears)
```

An example of its use can be seen in Section 12.2. A prototype web application was also developed to democratize the access to this method. In it, anyone, regardless of their knowledge in programming, can forecast the values of a time series using the GEARS method with just a few clicks. There are a few things that need to be done, and some other bugs that need fixing, but the prototype is available at `https://shiny.ogustavo.com/gears`.

10.2 A General Framework

For a time series y observed up until some time $T > 0$ and for each forecast lead $h = 1, \ldots, H$ and forecast horizon $H \in \mathbb{N}$, we want to find the out-of-sample forecast

$\hat{y}_{T+H}$. We define the following model to accomplish such task:

$$y_{T+h} = \boldsymbol{\beta}_c^{\intercal} \mathbf{X}_{c,T} + \varepsilon_{T+h} \tag{10.2}$$

where c is the c-th element from the σ-algebra $\mathfrak{C}$ that contains all combinations of k covariates - i.e., all possible combinations of elements from the set $\{\mathbf{x}_{1,T}, \ldots, \mathbf{x}_{k,T}\}$, with each $\mathbf{x}_{.,T}$ being a vector of variables observed up until time T. Therefore, $\mathfrak{C}$ is given by $\{[\mathbf{x}_{1,T}], \ldots, [\mathbf{x}_{k,T}], \ldots, [\mathbf{x}_{1,T}, \mathbf{x}_{k,T}], \ldots, [\mathbf{x}_{1,T} \cdots \mathbf{x}_{k,T}]\}$ and has cardinality $G \equiv \#\mathfrak{C} = \sum_{i=0}^{k} \binom{k}{i}$. From this, we have $\mathbf{X}_{1,T} \equiv [\mathbf{x}_{1,T}]$, and $\mathbf{X}_{G,T} \equiv [\mathbf{x}_{1,T} \cdots \mathbf{x}_{k,T}]$. Moreover, $\boldsymbol{\beta}_c$ is the set of parameters associated with the set of covariates $\mathbf{X}_{c,T}$, and ε_{T+h} is a mean zero error.

The selection of $\mathbf{X}_{c,T}$ is made by estimating an *ex-ante* version of Eq. 10.2 (given by Eq. 10.4) for all $c \in \mathfrak{C}$ and selecting the set of covariates that minimizes a given error measure (that can be MSE, sMAPE, MASE, MAD) for M subsamples of size S taken sequentially and in order from the set of indexes $\{1, \cdots, T-1\}$. Because they are taken sequentially and in order, we call each subsample $m = 1, \ldots, M$ a **rolling sample**.

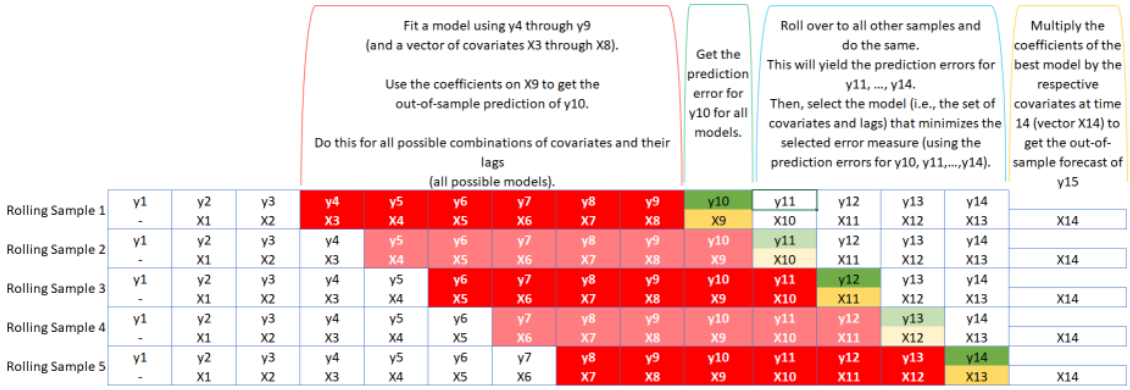
For each rolling sample $m = 1, \ldots, M$, we define the starting time t_m as an index from $\{1, \ldots, T-1\}$ given by:

$$t_m \equiv (T-h) - M - S + m, \quad h = 1, \ldots, H \tag{10.3}$$

For example, take $T = 15$ and $h = H = 1$. If we have $M = 5$ rolling samples of size $S = 6$, their starting times t_m will be $t_1 = 4$, $t_2 = 5$, $t_3 = 6$, $t_4 = 7$, $t_5 = 8$, where

$\{4, 5, 6, 7, 8\}$ are indexes from $\{1, \ldots, T-1\}$ (see Figure 10.1).

Figure 10.1: Example of $M = 5$ rolling samples of size $S = 6$, for $T = 15$ and $H = 1$.



Under this theoretical framework, we write the *ex-ante* version of Eq. 10.2 for a rolling sample $m \in 1, \ldots, M$ and forecast lead $h \in \{1, \ldots, H\}$:

$$y_{(t_m+s)+h} = \boldsymbol{\beta}^{\intercal}_{c|(t_m,S)} \mathbf{x}_{c,(t_m+s)} + \varepsilon_{(t_m+s)+h}, \quad s = 0, \ldots, S-1 \tag{10.4}$$

where $\mathbf{x}_{c,(t_m+s)}$ is the vector of covariates from $\mathbf{X}_{c,T}$ observed at time $t_m + s$ for $s = 0, \ldots, S-1$, and $\boldsymbol{\beta}_{c|(t_m,S)}$ is the vector of parameters associated with $\mathbf{x}_{c,t_m+s}$. [1]

We can write model 10.4 in matrix form for each $m \in 1, \ldots, M$ and forecast lead $h \in \{1, \ldots, H\}$ as:

$$\mathbf{y}_{(t_m,h)|S} = \boldsymbol{\beta}^{\intercal}_{c|(t_m,S)} \mathbf{X}_{(c,t_M)|S} + \boldsymbol{\varepsilon}_{(t_m,h)|S} \tag{10.5}$$

[1] It is also worth mentioning that when $m = M$ and $s = S-1$, we have that $t_m + s = t_M + (S-1)$, and using t_M from Eq. 10.3 we see that $t_M + (S-1) = (T-h) - M - S + M + (S-1) = (T-h) - 1$. Substituting this in the index for y in Eq.10.4 yields $y_{(T-h-1)+h} = y_{T-1}$. Doing the same in the index for $\mathbf{x}$ gives $\mathbf{x}_{c,(T-h-1)}$. If we define $T' \equiv T - 1 - h \implies T = T' + 1 + h$, then the indexes for y and $\mathbf{x}$ become $y_{T'+1+h-1} = y_{T'+h}$ and $\mathbf{x}_{c,(T'+1+h-h-1)} = \mathbf{x}_{c,T'}$. With these, we ca re-write Eq.10.4 as $y_{T'+h} = \boldsymbol{\beta}^{\intercal}_{c|(t_M,S)} \mathbf{x}_{c,T'} + \varepsilon_{T'+h}$, and this *ex-post* model is similar to the model given by Eq.10.2.

where

$$\mathbf{y}_{(t_m,h)|S} \equiv \begin{bmatrix} y_{(t_m+0)+h} \\ \vdots \\ y_{(t_m+S-1)+h} \end{bmatrix}; \quad \mathbf{X}_{(c,t_M)|S} \equiv \begin{bmatrix} \mathbf{x}_{c,(t_m+0)} & \dots & \mathbf{x}_{c,(t_m+S-1)} \end{bmatrix}$$

The estimates $\hat{\boldsymbol{\beta}}_{c|(t_m,S)}$ are obtained via partial likelihood[2] using S observations that start at the index t_m. The indexes c, t_m, S, albeit cumbersome, serve as reminders that for each set of covariates $c \in \mathfrak{C}$, each rolling sample $m \in \{1, \dots, M\}$ and each sample size S, we have different estimates $\boldsymbol{\beta}$. In other words, the $\boldsymbol{\beta}$'s are function of c, S, and M.

After obtaining $\hat{\boldsymbol{\beta}}_{c|(t_m,S)}$, we move on to calculate the out-of-estimation-sample [3] prediction error for $h \in \{1, \dots, H\}$ for each rolling sample $m = 1, \dots, M$:

$$\hat{e}_{h|(c,t_m,S)} \equiv y_{(t_m+S)+h} - \hat{y}_{(t_m+S)+h} \tag{10.6}$$

where $\hat{y}_{(t_m+S)+h} = \hat{\boldsymbol{\beta}}^{\intercal}_{c|(t_m,S)} \mathbf{x}_{c,(t_m+S)}$. Therefore, Eq.10.6 becomes

$$\hat{e}_{h|(c,t_m,S)} = y_{(t_m+S)+h} - \hat{\boldsymbol{\beta}}^{\intercal}_{c|(t_m,S)} \mathbf{x}_{c,(t_m+S)} \tag{10.7}$$

Having $\hat{e}_{h|(c,t_1,S)}, \dots \hat{e}_{h|(c,t_M,S)}$, we can calculate the selected error measure that we need to minimize. We define a generic function $f_e\left(\hat{e}_{h|(c,t_1,S)}, \dots \hat{e}_{h|(c,t_M,S)}\right)$ to account for

[2] Using the methodology defined in [71].

[3] We wrote "out-of-estimation-sample" because we wanted to emphasize that this prediction error uses information on y that is observed in the complete sample - $y_{(t_m+S)+h}$ -, but that lies outside the estimation sample used to get $\hat{\boldsymbol{\beta}}_{c|(t_m,S)}$, which uses information up until $y_{(t_m+S-1)+h}$ (see Eq.10.5 and comments below). We also want to stress that this out-of-estimation-sample prediction error is different from the out-of-sample prediction error, defined as $\hat{e}_T(h) \equiv y_{T+h} - \hat{y}_{T+h}$. To avoid confusion, we will refer to $\hat{e}_T(h)$ as the out-of-sample *forecast* error.

any selected error measure e. In the case of the MSE:

$$f_{MSE}\left(\hat{e}_{h|(c,t_1,S)},\dots\hat{e}_{h|(c,t_M,S)}\right) \equiv \frac{\sum_{m=1}^{M}\left(\hat{e}_{h|(c,t_m,S)}\right)^2}{M} = \frac{\sum_{m=1}^{M}\left[y_{(t_m+S)+h}-\hat{\boldsymbol{\beta}}'_{c|(t_m,S)}\mathbf{x}_{c,(t_m+S)}\right]^2}{M} \quad (10.8)$$

Finally, we want to find the set of covariates $c \in \mathfrak{C}$ that, given M and S, minimizes the selected error measure:

$$c^* \equiv \underset{c\in\mathfrak{C}|M,S}{\arg\min}\, f_e\left(\hat{e}_{h|(c,t_1,S)},\dots\hat{e}_{h|(c,t_M,S)}\right) \quad (10.9)$$

For the MSE, the above becomes:

$$c^* \equiv \underset{c\in\mathfrak{C}|M,S}{\arg\min}\, \frac{\sum_{m=1}^{M}\left[y_{(t_m+S)+h}-\hat{\boldsymbol{\beta}}'_{c|(t_m,S)}\mathbf{x}_{c,(t_m+S)}\right]^2}{M} \quad (10.10)$$

To get the *ex-post* forecast based on the best *ex-ante* forecasts, we calculate the expected value of y_{T+h} in the out-of-sample forecast equation, Eq.10.2:

$$\hat{y}_{T+h} = \hat{\boldsymbol{\beta}}'_{c^*,h|(M,S)}\mathbf{X}_{c^*,T} \quad (10.11)$$

where $\hat{\boldsymbol{\beta}}_{c^*,h|(M,S)}$ is either $\hat{\boldsymbol{\beta}}_{c^*,h|t_M,S}$, obtained from estimating Eq.10.5 using the last rolling sample only, t_M; or, the average of the $\hat{\boldsymbol{\beta}}_{c^*,h|t_m,S}$ estimated for all rolling samples.

10.3 Hyperparameters' Optimization

The results from the GEARS method vary significantly for different numbers of samples (M) and different sample sizes (S). While trying to mitigate these effects, we developed an optimization algorithm that borrows ideas from strategies used by Machine-Learning methods and uses validation schemes to search for the "best" values of these hyperparameters. The entire flowchart of the process can be seen in Figure 10.2.

We start the optimization algorithm by checking if the time series needs to be de-seasonalized. We follow the same procedures as the organizers of the M4 Competition used (see Section 11.1). That is, we first check the periodicity of the series using the `frequency` function from R. Then, a 90% autocorelation test is performed to decide whether the data are seasonal. If it is, we apply a classical multiplicative decomposition.

After that first step, we split the data into training and test sets using the holdout forward validation scheme. The former set is used for parameter estimation and calculation of the forecasts via GEARS using different numbers of samples and different sample sizes.

Then, the algorithm performs a "modified" grid search on the range of numbers of samples and sample sizes provided by the user. Grid search algorithms are not always the best option, since they can waste a lot of computational power exploring non-optimal value for the hyperparameters.

However, we feel that this approach works better than a random search strategy because we do not have a probability distribution for the hyperparameters, and even if we had, we would not be able to define an optimal point upon which the search algorithm

would stop. This issue arises from the fact that the algorithm selects the best values of S and M that minimizes the forecast error using a measure chosen by the user (MSE, MAD, MASE, or sMAPE - see Section 11.1 for the last two). Since some of these metrics are scale-dependent – or have problems with values of Y that are close to zero – an optimal point is data set-dependent, and setting its value beforehand may be unfeasible (as it is the case when we apply the GEARS strategy to the 100,000 time series from the M4 Forecasting Competition - see Chapter 11).

Due to the limitation of not being able to define an optimal stopping point, we say that our algorithm is a "modified" grid search algorithm. Instead of stopping when the error is sufficiently small, our algorithm evaluates all possible combinations of the given hyperparameters and obtains the best model specification for each one of them. Then, it calculates the desired error measure for all of them and returns the values for S and M associated with the smallest error.

Moreover, the algorithm is also capable of evaluating whether the best model should contain an intercept term, and if we should use the last set of $\hat{\beta}$'s or their average.

It is worth noting that the optimization algorithm is not worried about issues with adequacy[4] and diversity[5] since the it returns the number of samples and the sizes that best reflect the characteristics seen in the data set. And while these characteristics may not be present in future data, the purpose of this optimization approach is to give the user a workable solution to the problem of defining the "correct" values for the number of samples and the sample sizes.

[4]The number of forecasts at each lead time.

[5]Forecast error measures do not depend on special events and specific phases within the data set.

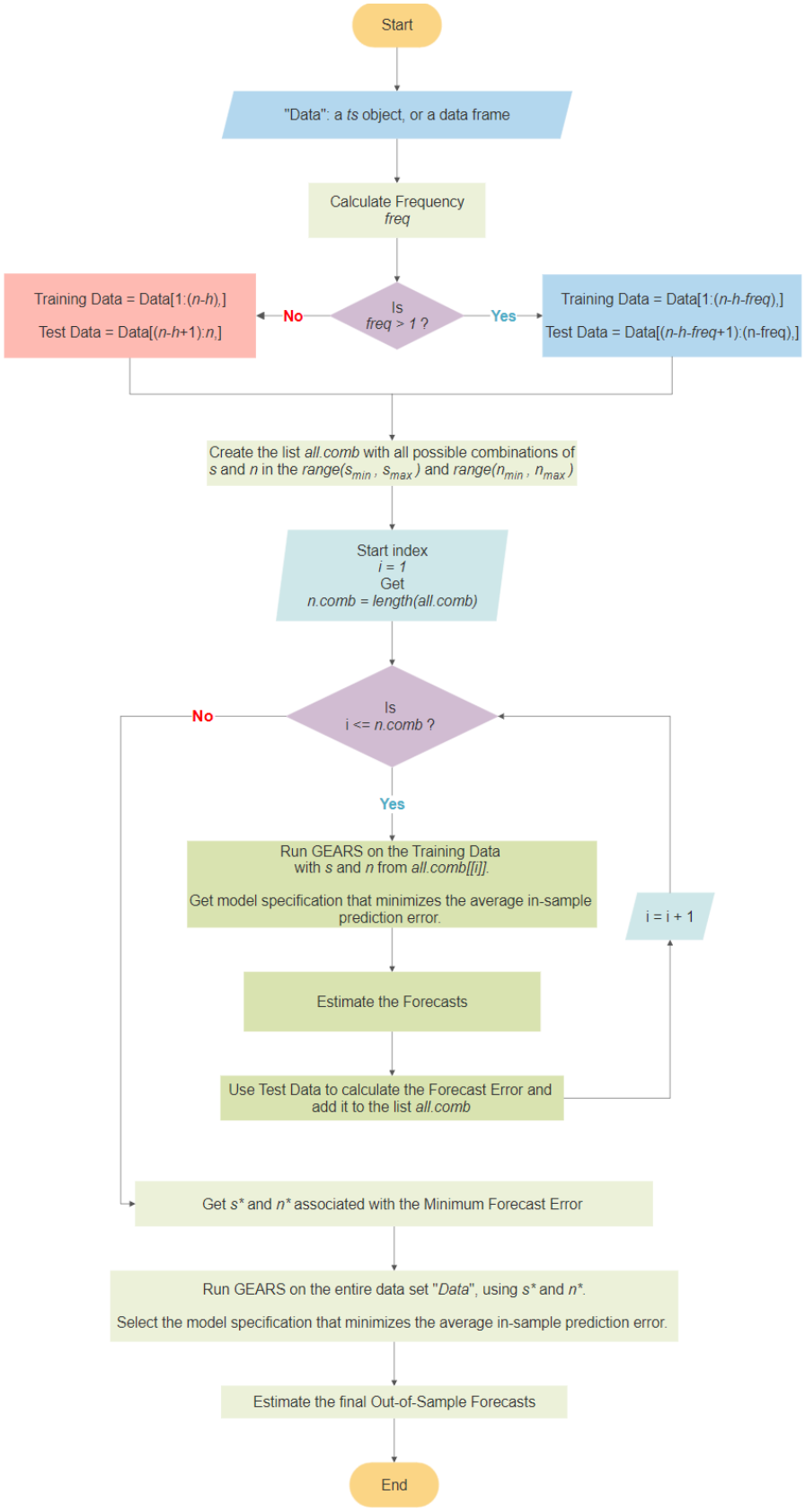


Figure 10.2: Flowchart of the optimization algorithm for the GEARS strategy.

Chapter 11: Data and Methodology

11.1 The M4 Competition and its data sets

The M4 Forecasting Competition took place in 2018. Participants in this competition had to forecast values for 100,000 real-life time series. It is a continuation of the competitions that started almost half a decade ago with the purpose of finding new models and methods to improve forecast accuracy.

The organizers had made available 100,000 univariate time series from different areas (demographic, finance, industry, macro, micro, and others) and different periodicity (yearly, quarterly, monthly, weekly, daily, and hourly), and participants were tasked with providing forecasts for each one of them. The number of forecasts varied by periodicity. Table 11.1 shows the number of series in each category along with the number of forecasts required by the organizers, and the minimum and maximum sample size of the series.

Forecast accuracy was measured by the *Overall Weighted Average* (OWA) of two accuracy measures: the Mean Absolute Scaled Error (MASE) and the symmetric Mean Absolute Percentage Error (sMAPE). These two metrics were calculated as

$$\text{sMAPE} = \frac{1}{h}\sum_{t=1}^{h}\frac{2|Y_t - \hat{Y}_t|}{|Y_t| + |\hat{Y}_t|} \tag{11.1}$$

Table 11.1: Number of M4 series, minimum and maximum sample sizes, and forecast horizon per data periodicity.

Periodicity	Number of Series	Min. Sample Size	Max. Sample Size	Forecast Horizon
Yearly	23,000	13	835	6
Quarterly	24,000	16	866	8
Monthly	48,000	42	2794	18
Weekly	359	80	2597	13
Daily	4,227	93	9919	14
Hourly	414	700	960	48
Total	100,000	-	-	-

and

$$\text{MASE} = \frac{1}{h} \frac{\sum_{t=1}^{h} |Y_t - \hat{Y}_t|}{\frac{1}{n-m} \sum_{t=m+1}^{n} |Y_t - Y_{t-m}} \tag{11.2}$$

where Y_t is the true out-of-sample value of a time series at point t (not available to participants at the time of estimation), and $\hat{Y}_t$ is the respective forecast. The number of observations in the training set available to the competitors is given by n, while h stands for the forecast horizon, and m is the periodicity of the data (i.e., 12 for monthly series).

After calculating the MASE and sMAPE for each submitted forecasting method, the organizers divided all errors by that of the *Naïve 2* forecasting method to obtain the Relative MASE and the Relative sMAPE. Then, the average between the Relative MASE and Relative sMAPE yields the OWA.

In the *Naïve 2* method, the forecasts are simply $\hat{Y}_{n+i} = Y_n$, for $i = 1, \ldots, h$, "but the data are seasonally adjusted, if needed, by applying a classical multiplicative decomposition. A 90% autocorrelation test is performed to decide whether the data are seasonal" [24, p. 57].

Other important statistical benchmark methods used in this competition include[1]:

- SES: Exponentially smoothing the data and extrapolating assuming no trend. Seasonal adjustments are considered as per *Naïve 2*;
- Holt: Holt's Exponential Smoothing method. The data are exponentially smoothed and extrapolated assuming a linear trend. Seasonal adjustments are considered as per *Naïve 2*;
- Damped: Exponentially smoothing the data and extrapolating assuming a damped trend. Seasonal adjustments are considered as per *Naïve 2*;
- Comb: The simple arithmetic average of the SES, Hold, and Damped methods. It is used as the *single benchmark for evaluating all other methods*.

A total of 49 valid forecasting methods were submitted, and several used a *combination approach*, in which several forecasts are obtained for a time series - either using different forecasting methods or distinct forecasting models - and the final forecast is the average between them. However, the method that won the 9,000€ first prize is a "hybrid" approach that utilized both statistical (exponential smoothing) and machine learning ("a 'black-box' recurrent neural network" [11, p. 2]) features.

11.2 The multivariate "commodities" data set

Traditional statistical approaches like VAR, State Space, or cointegration methods are often overly complicated and require many steps until a forecasting model is finally ready to be estimated. One of the advantages of the GEARS strategy over those methods is its simplicity in dealing with multivariate series. As long as the model is linear, GEARS is able to accommodate any model specification, including those with interaction terms.

Moreover, by allowing covariates in its forecasting equation, the GEARS strategy may present more accurate forecasts when compared to univariate approaches like

[1]The full list can be seen at Makridakis et al. [24, p. 57].

ARIMA or recurrent neural networks.

To evaluate if that is the case, we have included the "commodities_prices" data set as part of the `gears` package. It was created by taking price data (US$) on beef, swine (pork) meat, poultry (chicken) meat, maize (corn), and wheat from the International Monetary Fund on Primary Commodities Prices data set[2]. The data set has monthly data from January, 1980 until March, 2020 (483 time points). For more information, run the command `gears::commodities_prices_data_dictionary` in R.

11.3 Partial Likelihood Estimation

Partial likelihood inference for time series following generalized linear models is the natural extension of GLM methods to deal with dependent data. One of the main outcomes from the theory devised by Kedem and Fokianos [71, 72] is that the main inferential features appropriate for independent data can be transported to time series data.

Such transportation is enabled by the use of partial likelihood methods, since it allows for "temporal or sequential conditional inference with respect to a filtration generated by all that is known to the observer at the time of observation" [71, p. 1]. This means that the main difference between "regular" GLM inference and inference based on partial likelihood (PL) lies in the interpretation of the results. That is, the same estimation procedures are carried out in both methods, but, in the latter, the outcome needs to be interpreted as the response conditioned to the all that it is observed at time t. This works well for our purposes, since we want to use data-driven methods to obtain forecasts for a

[2] `https://www.imf.org/~/media/Files/Research/CommodityPrices/Monthly/Table3.ashx` . Access on April 17, 2020.

specific training set, and we are only interested in making predictions - not inference.

Another advantage gained by resorting to PL inference is that "[t] he definition of PL does not require the joint distribution of the response and the covariates. Thus, in PL inference, the joint distribution of the response and covariates is left unspecified. In addition, any stationarity assumptions may be dropped" [72, p. 174].

The combination of PL and GLMs provide a suitable framework to our GEARS strategy, since it is easy to implement (since a number of existing software packages can already be used to analyze GLMs) and does not require stationarity. The use of this method is a more transparent way to estimate the parameters of the forecasting models, than traditional machine learning methods. This contributes for a more reproducible approach to the learning procedure seen in Chapter 9.

Chapter 12: Results

12.1 One-Step Ahead - All 100,000 series from the M4 Competition

The GEARS strategy can be a data-expensive method if the total sample size of the series is relatively small when compared to the forecast horizon, and also if one wants to include larger lags on the right-hand side of the forecasting model. We observe the former issue in the "Yearly" series, for example. The minimum sample size is equal to 13 and we need to provide six-steps ahead forecasts (Table 11.1) for those series. The GEARS strategy cannot produce forecasts with that few observations, so we restricted our analysis to one-step ahead forecasts only. For the number of lags, due to the limitation in the sample sizes, we also had to restrict the maximum number of lags. Here, the maximum was equal to 2.

To obtain the forecasts for the time series from the M4 Competition, we had to select the initial values for the sample sizes and number of rolling samples for each type of data (based on their periodicitity). Since the computational time to perform a grid search considering several values of S (sample size) and M (number of rolling samples) for all 100,000 series would be exorbitant, we randomly selected only a few series for a first round of evaluation. After that, we selected the set of initial values used here.

For "Hourly" series, we used 24, 36, and 48 as the number of rolling samples, with

sizes equal to 144 450 as starting points for the optimization algorithm.

The numbers of sample sizes used for the "Daily" data sets were equal to 30 and 60, and the number of rolling samples was fixed in 12.

When evaluating the "Weekly" series, we had to break it down by the length of the time series. For those series with total length equal to 80, we used $S = 47$ and the number of sample equal to 5 and 6. Otherwise, we tested the values $S = 156$ and $M = 52$. Since we have only one value for S and M when $T > 80$, the optimization algorithm was used to decide whether the model should have an intercept, and if we should consider the last betas or their average.

The "Yearly" and "Quarterly" series were complicated ones. With some series being short, we had to make several adjustments to the values of S and M.

For the "Yearly" cases, when $T \leq 15$, we used $S = 6$ and $M = 3$. For $16 < T < 50$, $M = 5$ and the sample sizes were equal to 5 and 6. In the cases where $50 \leq T < 100$, M was also equal to 5, while $S = 10, 20$. For the other cases, $M = 30$ and $S = 20, 60$.

In the case of the "Quarterly" series, we used $S = 5$ and $M = 3$ when $T \leq 18$. For the cases where $18 < T \leq 25$, we used four rolling samples with size equal to 8. The same number of rolling samples ($M = 4$) was used in the cases where $25 < T \leq 50$, but with $S = 14$. For all other cases, we used $S = 20$ and $M = 4, 12$.

Despite the fact that "Monthly" series had time series with minimum lengths larger than the ones for the "Yearly" and "Quarterly" series (Table 11.1), these were the most complicated ones in terms of adjusting the sample sizes and the number of rolling samples. We ended up setting $M = 4$ for all series with $T \leq 100$; and $M = 10$, otherwise. For series with $T > 100$, $S = 48$. In the cases where $59 < T < 100$, the sample size

used was equal to 36. The value $S = 30$ was used when $53 < T \leq 59$, and $S = 22$ when $45 < T \leq 53$. Finally, for the cases with $T \leq 45$, $S = 18$.

The order of the methods in the rows of Table 12.1 is the same as the final order (considering the multi-step ahead forecasts). Our method was only able to beat one method (the purely statistical method with user ID number equal to 239). This means that it did not perform better than the simple *Naïve 2* method or the "Comb" method (the overall benchmark considered by the organizers - see Section 11.1).

The results using the GEARS strategy with the M4 Forecasting Competition data were not great (Table 12.1). However, the GEARS strategy produced the smallest results more often (Table 12.1). Out of the 100,000 cases, the GEARS strategy was the best method in 8,750. However, when it misses, it produces very large values of the OWA.

We believe that the problematic cases are due to the initial choice of hyperparameters, or the many adjustments that we did, or because we had to restrict the maximum number of lags to 2. For example, we selected 59 cases of daily series that had a numerical value for the OWA greater than 10, and we optimized them using a range for S and M greater than what was used before. We observed major reductions in the OWA for all series, with some cases going from an OWA equal to 137.91 to 1.31 (Table D.1). Only in four out of the 59 cases we were not able to reduce the OWA to values smaller than 10. However, we were able to obtain a reduction of at least 65% in those cases. Since the number of optimized cases (59) is small compared to the number of series evaluated, the result of the optimization had no impact on the results from Figure 12.1.

Method			sMAPE	MASE	OWA	Overview				
User ID	Affiliation	Type of Method	Total	Total	Total	Better than Comb	General Rank (OWA)	General Rank (sMAPE)	General Rank (MASE)	% improvement over Comb
118	Uber Technologies	Hybrid	6.281	0.738	0.897	1	4	2	6	4.15
245	University of A Coruña & Monash University	Combination (S & ML)	6.186	0.721	0.880	1	1	1	1	6.00
237	ProLogistica Soft	Combination (S)	6.300	0.722	0.889	1	2	3	2	5.03
72	Individual	Combination (S & ML)	6.312	0.735	0.898	1	5	4	5	4.12
69	University of Brasília & University of São Paulo	Combination (S)	6.330	0.731	0.896	1	3	5	3	4.28
36	University of Bath & Lancaster University	Combination (S)	6.342	0.734	0.899	1	6	6	4	3.97
78	Harvard Extension School	Combination (S)	6.890	0.797	0.977	0	21	21	17	-4.32
260	National Technical University of Athens	Statistical	6.505	0.760	0.927	1	9	9	9	1.00
238	University of Oxford	Combination (S)	6.382	0.757	0.916	1	7	7	8	2.17
39	University of Castilla-La Mancha	Combination (S)	6.656	0.768	0.942	0	15	17	13	-0.62
5	National Technical University of Athens	Statistical	6.546	0.768	0.935	1	10	11	14	0.16
132	Washington State Employment Security Department	Combination (S)	6.789	0.798	0.970	0	19	20	18	-3.60
251	Georgia Institute of Technology	Statistical	6.555	0.800	0.954	0	18	12	20	-1.95
250	University of Tartu	Combination (S & ML)	7.056	0.837	1.013	0	25	24	25	-8.20
243	Universiti Tun Hussein Onn Malaysia	Combination (S)	6.443	0.752	0.917	1	8	8	7	2.01
235	Business Forecast Systems (Forecast Pro)	Statistical	6.527	0.799	0.952	0	16	10	19	-1.67
104	Pontifical Catholic University of Rio de Janeiro	Combination (S)	6.944	0.825	0.997	0	22	22	24	-6.52
Theta	Benchmark	Statistical	6.593	0.774	0.942	0	14	14	15	-0.59
Com	Benchmark	Combination (S)	6.586	0.766	0.936	0	11	13	12	0.00
ARIMA	Benchmark	Statistical	6.645	0.762	0.938	0	12	16	10	-0.15
223	Individual	Combination (S & ML)	6.747	0.811	0.975	0	20	19	22	-4.11
Damped	Benchmark	Statistical	6.677	0.783	0.953	0	17	18	16	-1.76
ETS	Benchmark	Statistical	6.622	0.764	0.938	0	13	15	11	-0.15
239	Universidad Miguel Hernández & Universitat de Valencia	Combination (S)	7.584	0.938	1.112	0	27	27	27	-18.80
211	Individual	Machine Learning	7.161	0.804	1.000	0	24	25	21	-6.83
Naive2	Benchmark	Statistical	7.025	0.820	1.000	0	23	23	23	-6.82
GEARS			7.480	0.890	1.075	0	26	26	26	-14.85

sMAPE: symmetric mean absolute percentage error
MASE: mean absolute scaled error
OWA: overall weighted average of the relative sMAPE and the relative MASE

Figure 12.1: One-step ahead forecast accuracy of the top 25 methods from the M4 Competition and the GEARS strategy considering all 100,000 time series from the competition.

Table 12.1: Number of times a method yielded the smallest OWA, per series periodicity type.

Method ID	Hourly	Daily	Yearly	Weekly	Quarterly	Monthly	TOTAL
118	20	451	1108	35	1460	4187	7261
245	19	55	566	11	683	1350	2684
237	14	751	404	9	436	1247	2861
72	18	27	300	15	434	829	1623
69	7	8	320	2	366	760	1463
36	21	32	296	5	364	618	1336
78	9	213	472	12	725	2505	3936
260	5	15	484	2	438	648	1592
238	21	208	1170	22	1320	3025	5766
39	52	11	317	1	886	1542	2809
5	2	12	399	1	441	860	1715
132	9	387	1036	11	1411	2827	5681
251	5	16	446	3	389	503	1362
250	23	227	1406	38	1573	2403	5670
243	24	120	614	19	527	1131	2435
235	19	59	649	16	1083	2219	4045
104	20	272	1397	20	1296	2932	5937
Theta	1	10	1494	1	428	492	2426
Com	8	11	555	3	383	523	1483
ARIMA	18	95	803	14	1286	2860	5076
223	17	160	709	14	1065	2094	4059
Damped	17	19	535	2	643	1044	2260
ETS	22	15	601	9	666	1243	2556
239	12	184	2468	15	1415	2707	6801
211	19	385	2229	45	2167	3568	8413
GEARS	12	484	2222	34	2115	3883	8750
TOTAL	414	4227	23000	359	24000	48000	100000

12.2 One-Step Ahead - The multivariate "commodities prices" data set

The goal with the "commodities_prices" data set is to find the forecasts for the pork price variable. The "maximum" tentative model is:

$$\begin{aligned}\text{PORK\_PRICE}_{t+h} = \text{Intercept} &+ \text{PORK\_PRICE}_{t} + \text{PORK\_PRICE}_{t-1} \\ &+ \text{BEEF\_PRICE}_{t} + \text{BEEF\_PRICE}_{t-1} + \text{BEEF\_PRICE}_{t-2} \\ &+ \text{CORN\_PRICE}_{t-4} + \text{POULTRY\_PRICE}_{t-5} \\ &+ \text{CORN\_PRICE}^2_{t-1} + \text{WHEAT\_PRICE}_{t-2} \cdot \text{BEEF\_PRICE}_{t-3} \end{aligned} \quad (12.1)$$

Here, the lags up to $t-1$ of the variable PORK_PRICE are included in the right-hand side of the forecasting model. The covariate BEEF_PRICE also has a "variable" lag count, with values up to $t-2$ being included in the above equation. The covariates CORN_PRICE and POULTRY_PRICE have "fixed" lags, with only the $t-4$ and $t-5$ values - respectively - being included in Eq.(12.1). A quadratic term for CORN_PRICE is also included. Finally, we also have an interaction term between WHEAT_PRICE (at the "fixed" lag $t-2$) and BEEF_PRICE (lat $t-3$).

We wrote "maximum" because the GEARS strategy works by going over all possible combinations of the variables in the right-hand side of Eq.(12.1) - a total of 1022 distinct equations - in the search for the best forecasting model formulation. The "maximum" tentative model is the one that includes all variables (shown in Eq.(12.1), and the

"minimum" models are the ones that have only one covariate. For example,

$$\text{PORK\_PRICE}_{t+h} = \text{PORK\_PRICE}_t$$

or

$$\text{PORK\_PRICE}_{t+h} = \text{CORN\_PRICE}^2_{t-1}.$$

We can easily obtain the forecasts for the above models using the `gears` package. For instance, if we want to obtain the one-step-ahead forecast ($h = 1$) for $T = 150$ using $M = 5$ random samples with size $S = 25$, we only need to provide the following piece of code,

Listing 12.1: R code to run the GEARS strategy with the "commodities prices" multivariate time series.

```
example_gears <- gears(
  DATA                = gears::commodities_prices,
  forecast.horizon    = 1,
  size.rs             = 25,
  number.rs           = 5,

  y.name              = "PORK_PRICE",
  y.max.lags          = 1,

  x.names             = list("BEEF_PRICE"),
  x.max.lags          = list(2),

  x.fixed.names       = list("CORN_PRICE", "POULTRY_PRICE"),
  x.fixed.lags        = list(4, 5),

  x.interaction.names =
      list("CORN_PRICE*CORN_PRICE", "WHEAT_PRICE*BEEF_PRICE"),
  x.interaction.lags  = list(c(1, 1), c(2,3)),

  error.measure       = "mse",

  last.obs            = 150
)
```

When considering 5 random samples, the total number of estimated models is equal to $5 \cdot 1022 = 5110$. It took 3.5 seconds to estimate all these models using a Intel Core i7 with 4.8 GHz turbo frequency. If the number of models that need to be estimated is

large, the user could benefit from using parallel computing. To do so, simply add the code `use.parallel = TRUE`, and specify the number of cores with `num.cores =` the desired number.

The "best" selected model using the GEARS strategy is

$$\text{PORK\_PRICE}_{t+1} = \text{PORK\_PRICE}_t + \text{BEEF\_PRICE}_t + \text{BEEF\_PRICE}_{t-2} \quad (12.2)$$

The results from applying the GEARS strategy to the "commodities prices" data set can be viewed in Table 12.2, below. For comparison, we added the results from an automatically selected ARIMA model[1] and from the Theta-method, the method that won the M3 Forecasting Competition [105].

Table 12.2: Observed future values and respective forecasts with a 95% prediction interval, and forecast errors for the price of pork meat obtained from a multivariate GEARS strategy and an ARIMA model.

Procedure	Observed	Forecast	Lower 95 PI	Upper 95 PI	Absolute Error	Squared Error
ARIMA	61.04	64.28	41.31	87.25	3.24	10.49
Theta-Method	61.04	60.87	37.21	84.52	0.17	0.03
GEARS - Last betas	61.04	60.75	42.09	79.42	0.29	0.08
GEARS - Avg. betas	61.04	60.82	42.16	79.49	0.22	0.05

We see that the forecasts from the GEARS strategy are closer to the observed value of PORK_PRICE at $T = 151$ and yielded a smaller forecast error than the one from the ARIMA method. Also, the 95% prediction interval from the ARIMA procedure seems to be more conservative than the one from the GEARS strategy. However, the Theta-Method

[1] Obtained from the function `forecast::auto.arima` and with the option `stationary = TRUE`. The selected model was an AR(2).

produced a more precise forecast, albeit with the most conservative prediction interval. Its range was equal to 47.3 versus 43.94 from the ARIMA and 37.33 from the GEARS strategy.

Chapter 13: Final Remarks

One of the issues with time series modeling is selecting the covariates that will provide the best out-of-sample forecasts. The development of information criteria (AIC, BIC) facilitated selecting predictors and their lags. However, not always the chosen model by minimizing information criteria is the one that provides the best forecasts. With the above in mind, we propose the GEARS method, a Generalized And Rolling Sample method that focuses on selecting the best set of covariates (and their lags) for forecasting.

The "generalized" part of the name is because we use generalized linear models combined with partial likelihood inference to estimate the parameters. It was showed that partial likelihood inference enables very flexible conditions that allow for correct time series analysis using GLMs. With this, it becomes easy for users to estimate multivariate (or univariate) time series models. All they have to do is provide the right-hand side variable, the variables that should enter the left-hand side of the model, and their lags. GLMs also allow for the inclusion of interactions and all sorts of non-linear links. This easy setup is an advantage over more complicated models like state-space and GARCH. And the fact that we can include covariates and interactions is an advantage over ARIMA, Theta-method, and other univariate methods.

The "rolling sample" part relates to estimating the parameters over a rolling window

of a fixed size. The idea is to "train" our model by breaking down our sample in sections of length S, taking the last M sections, and fitting/predicting several models at each of these sections. Each fitted model is taken from the set with all possible combinations of covariates and lags included in the right-hand side of the forecasting model. Then, we use the out-of-sample prediction errors to calculate an error measure (e.g., MSE) for all the models in that set, and select the one that minimizes this error measure. Once this is done, the best model's estimated coefficients are used to get the out-of-sample forecasts.

An R package was written for the GEARS method. A prototype web application was also developed to democratize the access to this method. In it, anyone, regardless of their knowledge in programming, can forecast the values of a time series using the GEARS method with just a few clicks. There are a few things that need to be done, but the prototype is available at `https://shiny.ogustavo.com/gears`.

We applied the GEARS method to all the time series used in the 2018's M-Competition, the M4 Competition. Participants in this competition had to forecast values for 100,000 real-life time series. Due to the nature of the GEARS strategy, we had to focus on one-step ahead forecasts. Given the supplied hyperparameters, the performance of the GEARS strategy on these data sets was not great. After optimizing the hyperparameters for a subset of daily series, we were able to detect significant decreases in the OWA values. Moreover, we had the best results in 8,750 cases out of the 100,000, and the method that won the competition had better results in fewer than 7,300 series.

Due to these problems in performance, we learned about the limitations of the GEARS strategy. First, it is important to observe that $\hat{\boldsymbol{\beta}}_{c^*,h|(M,S)}$ directly depends on the number of rolling samples M and the size of each sample S. Hence, finding proper

values for M and S is a major concern.

Using a grid search approach - like we did - is inefficient since the size of the grid is given by $\#\mathfrak{C} \times \#\mathcal{M} \times \#\mathcal{S} \times H$, where $\#\mathcal{M}$ is the cardinality of the set of all numbers of rolling samples considered, and $\#\mathcal{S}$ is the cardinality of the set of sample sizes to use. Testing for all possible combinations of M and S would take an unreasonable long time, and selecting values at random is not a good approach.

Finding a formula for the expected (effective) sample size given M could improve this search. Maybe we could use adaptive sampling methods to find a predictor $\hat{y}_{t+h}$ that is model-unbiased for y_{t+h} - i.e., given a sample s, the conditional expectation of $\hat{y}_{t+h}$ equals the expectation of y_{t+h}, given s. We leave these topics for future research.

On the other hand, the GEARS strategy shows promise when dealing with multivariate time series, but caution is needed when dealing with a high-dimensional space of covariates. The way GEARS was developed goes over all possible model specifications, and if several covariates need to be included, the computational cost makes the entire process unfeasible. If one wants to use the GEARS strategy in such scenario, a prior selection of the variables - either by the user or by some sort of principal component analysis on dependent data - is advised.

Here, we estimated several forecasting models based on a complex formulation that includes covariates with variable and fixed lags, quadratic terms, and interaction terms. Despite its complexity, the full model had ten covariates only. The accuracy of the forecasts obtained with GEARS was far superior than the one observed for the predictions from an ARIMA. This result and the fact that our strategy for dealing with multivariate series is far simpler than VAR, State Space, or Cointegration approaches shines a light in

the future of our procedure. Further investigation is required to confirm this.

Appendix A: Part I - Complementary Results using RMSE

A.1 Data from Cerqueira et al. (2020)

A.1.1 Results from the RBR learning algorithm

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.5788	-2.4769	0.7705	0.3224	2.7703	7.1041	0.3244
cp-holdout	-4.5788	-2.5268	0.7124	0.2834	2.7403	7.1041	0.4044
CV	-4.5566	-2.4297	0.7087	0.2096	2.6315	7.8831	0.3244
CV-Bl	-4.5587	-2.4710	0.8332	0.1645	2.5098	7.6575	0.4044
CV-hvBl	-4.5583	-2.4183	0.8541	0.2035	2.5845	7.6305	0.2554
CV-Mod	-4.5538	-2.1114	1.3502	0.5835	2.7296	7.9304	0.0577
Holdout	-4.5788	-2.6061	-0.4060	0.1219	2.7191	5.7406	0.8202
p-holdout	-4.5788	-2.3682	-0.5929	0.1993	2.7437	7.0945	0.8202
Preq-Bls	-4.5534	-2.0508	1.5375	0.6307	2.7773	7.6699	0.0185
Preq-Bls-Gap	-4.5398	-2.1370	1.7604	0.6832	2.9142	7.5973	0.0049
Preq-Grow	-4.5785	-2.8729	-1.6308	-0.3656	2.2195	4.7890	0.1973
Preq-Sld-Bls	-4.5432	0.8633	2.5709	1.7509	3.5157	7.7528	< 0.0001
Preq-Slide	-4.5786	-2.8526	-1.5470	-0.3395	2.2411	4.8812	0.1973
Rep-Holdout	-4.5788	-2.5204	-0.5614	0.0150	2.5947	5.3605	0.4952

Table A.1: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to all 174 real-world time series using the RBR learning algorithm and the RMSE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-3.4843	-2.2082	0.8286	0.3128	2.6194	4.5326	0.3099
cp-holdout	-3.4843	-2.2082	0.7124	0.2245	2.5758	4.2740	0.5426
CV	-3.5890	-2.0057	0.9418	0.3637	2.5654	3.7063	0.1548
CV-Bl	-3.5337	-1.8785	1.1798	0.3838	2.3944	3.6755	0.1038
CV-hvBl	-3.5185	-1.9562	1.2119	0.4209	2.4671	3.6849	0.0417
CV-Mod	-3.4540	-1.2981	1.4713	0.7854	2.6299	3.8795	0.025
Holdout	-3.6721	-2.2363	-0.7761	0.1008	2.5885	4.0479	0.6849
p-holdout	-3.5389	-2.2363	-1.0056	-0.0368	2.5758	4.3604	0.4168
Preq-Bls	-3.3461	-1.1780	1.8384	0.8788	2.7570	4.6476	0.0042
Preq-Bls-Gap	-3.3830	-1.8530	1.9774	0.9172	2.8406	4.3903	0.0022
Preq-Grow	-3.7929	-2.8638	-1.2456	-0.2775	2.2045	4.3903	0.6849
Preq-Sld-Bls	-3.1020	1.4224	2.7139	1.9799	3.3774	4.8757	< 0.0001
Preq-Slide	-3.7788	-2.7395	-0.7522	-0.2288	2.1604	4.3266	0.8392
Rep-Holdout	-3.5789	-2.3903	-1.0569	-0.1022	2.5010	4.0311	0.3099

Table A.2: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 97 stationary time series using the RBR learning algorithm and the RMSE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.5788	-2.6355	0.5254	0.3344	3.0704	7.1041	0.8199
cp-holdout	-4.5788	-2.8427	0.6184	0.3576	3.0397	7.1041	0.6488
CV	-4.5566	-2.8861	-0.3445	0.0153	2.6861	7.8831	1
CV-Bl	-4.5587	-2.9603	-1.2855	-0.1117	2.6442	7.6575	0.6488
CV-hvBl	-4.5583	-2.9789	-0.5433	-0.0705	2.6183	7.6305	0.6488
CV-Mod	-4.5538	-2.8987	1.1487	0.3293	3.0322	7.9304	0.8199
Holdout	-4.5788	-2.8135	0.2651	0.1485	3.0514	5.7406	1
p-holdout	-4.5788	-2.6204	0.6928	0.4967	3.2208	7.0945	0.6488
Preq-Bls	-4.5534	-2.7940	1.0798	0.3182	3.0166	7.6699	0.8199
Preq-Bls-Gap	-4.5398	-2.7024	1.2664	0.3884	2.9698	7.5973	0.4944
Preq-Grow	-4.5785	-3.0528	-1.6551	-0.4766	2.2246	4.7890	0.1711
Preq-Sld-Bls	-4.5432	-1.8865	2.2544	1.4625	3.6415	7.7528	0.0013
Preq-Slide	-4.5786	-3.0522	-1.6592	-0.4790	2.4350	4.8812	0.11
Rep-Holdout	-4.5788	-2.6724	0.0236	0.1626	2.7651	5.3605	1

Table A.3: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 77 non-stationary time series using the RBR learning algorithm and the RMSE as error measure.

A.1.2 Results from the RF learning algorithm

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.0289	-2.4280	1.1269	0.3841	2.7261	6.1486	0.4952
cp-holdout	-4.0289	-2.4744	1.1721	0.3907	2.7452	6.1486	0.3244
CV	-3.8173	-2.4167	0.5600	0.1630	2.6019	7.0524	0.7048
CV-Bl	-3.9369	-2.4460	0.6234	0.1531	2.6130	6.7960	0.9396
CV-hvBl	-3.9409	-2.4121	0.6709	0.1804	2.6401	6.8141	0.4952
CV-Mod	-3.8067	-2.0778	1.7031	0.7264	2.8666	7.0800	0.0078
Holdout	-4.3271	-2.5913	0.1077	0.1426	2.6562	5.5521	1
p-holdout	-4.0289	-2.4584	1.2800	0.4463	2.7606	6.2120	0.3244
Preq-Bls	-3.9760	-1.9701	1.9670	0.8140	3.0358	6.8069	0.0049
Preq-Bls-Gap	-4.1050	-1.9836	1.9569	0.8191	2.9756	6.7653	0.0078
Preq-Grow	-4.3603	-2.8352	-1.9490	-0.4130	2.4588	4.9343	0.0809
Preq-Sld-Bls	-3.9794	0.9916	2.7371	1.9242	3.5376	6.8094	< 0.0001
Preq-Slide	-4.3595	-2.8067	-1.6564	-0.3204	2.4035	4.9139	0.1973
Rep-Holdout	-4.2271	-2.4835	-0.8393	0.1260	2.8689	6.2388	0.8202

Table A.4: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to all 174 real-world time series using the RF learning algorithm and the RMSE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-3.0703	-2.3338	1.8007	0.3984	2.6378	4.1409	0.4168
cp-holdout	-3.0703	-2.3338	1.2271	0.3527	2.5942	4.1311	0.4168
CV	-3.3534	-2.1897	0.9031	0.3218	2.5391	3.6503	0.2229
CV-Bl	-3.3433	-2.0402	0.9381	0.3419	2.4884	3.5695	0.4168
CV-hvBl	-3.3019	-1.9557	1.0256	0.3877	2.5289	3.5988	0.1548
CV-Mod	-3.1564	-1.3374	1.8047	0.9337	2.8042	4.0367	0.0042
Holdout	-3.4415	-2.5109	0.4072	0.0861	2.6289	4.1311	1
p-holdout	-3.0930	-2.2421	1.2271	0.3879	2.6114	4.1395	0.4168
Preq-Bls	-3.0050	-1.3642	2.0852	1.0505	2.9536	4.2071	0.0042
Preq-Bls-Gap	-2.9122	-1.4770	2.2479	1.0610	2.9335	4.3786	0.008
Preq-Grow	-3.7775	-2.8086	-1.8606	-0.3640	2.2594	4.1543	0.5426
Preq-Sld-Bls	-2.7243	1.7439	2.8032	2.3567	3.3309	5.6070	< 0.0001
Preq-Slide	-3.7930	-2.8035	-1.6217	-0.2512	2.1638	4.1357	0.8392
Rep-Holdout	-3.3054	-2.4382	-0.9957	0.0345	2.6665	3.9545	0.6849

Table A.5: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 97 stationary time series using the RF learning algorithm and the RMSE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.0289	-2.6097	0.1735	0.3661	3.2044	6.1486	1
cp-holdout	-4.0289	-2.8292	1.0996	0.4385	3.2888	6.1486	0.6488
CV	-3.8173	-2.9534	-0.9368	-0.0370	2.7629	7.0524	0.4944
CV-Bl	-3.9369	-2.9977	-0.7905	-0.0848	2.7430	6.7960	0.4944
CV-hvBl	-3.9409	-2.9798	-0.8663	-0.0809	2.7158	6.8141	0.6488
CV-Mod	-3.8067	-2.7064	0.9095	0.4652	3.0445	7.0800	0.4944
Holdout	-4.3271	-2.7599	-0.3439	0.2139	3.1521	5.5521	1
p-holdout	-4.0289	-2.8292	1.5567	0.5198	3.1075	6.2120	0.6488
Preq-Bls	-3.9760	-2.7673	1.4814	0.5161	3.2158	6.8069	0.362
Preq-Bls-Gap	-4.1050	-2.6992	1.7053	0.5145	3.2062	6.7653	0.362
Preq-Grow	-4.3603	-2.8647	-2.0801	-0.4748	2.6105	4.9343	0.0675
Preq-Sld-Bls	-3.9794	-2.4177	2.5796	1.3793	3.8962	6.8094	0.0059
Preq-Slide	-4.3595	-2.8840	-2.0627	-0.4075	2.7179	4.9139	0.11
Rep-Holdout	-4.2271	-2.6409	0.0567	0.2412	3.2395	6.2388	1

Table A.6: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 77 non-stationary time series using the RF learning algorithm and the RMSE as error measure.

A.1.3 Results from the GLM learning algorithm

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.1097	-2.4885	-0.3307	0.0999	2.5174	6.8405	0.7048
cp-holdout	-4.1097	-2.4778	0.3881	0.1270	2.5355	6.8405	0.9396
CV	-3.8529	-2.2680	0.9806	0.3183	2.5502	7.7182	0.2554
CV-Bl	-3.9883	-2.3304	0.5779	0.1493	2.4375	7.4567	0.9396
CV-hvBl	-3.9870	-2.3300	0.6453	0.1563	2.4486	7.4569	0.9396
CV-Mod	-3.8457	-2.1897	1.2392	0.4596	2.6485	7.7211	0.0809
Holdout	-4.3466	-2.5133	-0.6838	-0.0265	2.5251	5.2947	0.4044
p-holdout	-4.1097	-2.3760	0.3347	0.1527	2.5054	6.9337	0.8202
Preq-Bls	-3.9943	-2.2698	0.9435	0.2885	2.4859	7.4220	0.1973
Preq-Bls-Gap	-4.1550	-2.2828	1.0548	0.2872	2.5160	7.3584	0.1973
Preq-Grow	-4.3757	-2.8720	-1.5454	-0.4517	2.2089	4.4979	0.0404
Preq-Sld-Bls	-3.9882	-1.6415	1.9821	1.0318	2.9126	7.4408	< 0.0001
Preq-Slide	-4.3726	-2.8197	-1.4889	-0.4261	2.2856	4.4965	0.0809
Rep-Holdout	-4.2393	-2.5412	-1.1865	-0.0915	2.5811	5.3116	0.1973

Table A.7: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to all 174 real-world time series using the GLM-Ridge learning algorithm and the RMSE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-3.4215	-2.2602	0.5296	0.1625	2.6610	4.2216	1
cp-holdout	-3.4215	-2.2602	0.5296	0.1170	2.5131	4.2792	1
CV	-3.4883	-1.9162	1.5736	0.4770	2.4733	4.3768	0.1038
CV-Bl	-3.5159	-2.0807	1.3440	0.3502	2.2331	4.3234	0.5426
CV-hvBl	-3.5080	-1.9543	1.3448	0.3617	2.2395	4.3234	0.5426
CV-Mod	-3.4576	-1.8785	1.5837	0.5964	2.6087	4.3612	0.0417
Holdout	-3.5140	-2.4918	-0.6650	0.0129	2.5333	4.2792	0.8392
p-holdout	-3.4504	-2.2624	0.2833	0.1105	2.5714	4.2792	1
Preq-Bls	-3.5115	-1.7991	1.6316	0.5850	2.4859	4.3430	0.0417
Preq-Bls-Gap	-3.4693	-1.9625	1.4428	0.5510	2.5334	4.3204	0.0671
Preq-Grow	-3.7284	-2.8298	-0.8587	-0.2543	2.3485	4.2259	0.5426
Preq-Sld-Bls	-3.2465	0.3720	2.2441	1.4564	2.9481	4.3573	< 0.0001
Preq-Slide	-3.7171	-2.7939	-0.7888	-0.2195	2.3978	4.2331	0.8392
Rep-Holdout	-3.3519	-2.3796	-1.2021	-0.1237	2.5447	4.1333	0.3099

Table A.8: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 97 stationary time series using the GLM-Ridge learning algorithm and the RMSE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.1097	-2.6411	-0.5686	0.0210	2.4112	6.8405	0.4944
cp-holdout	-4.1097	-2.7000	0.2465	0.1396	2.5357	6.8405	1
CV	-3.8529	-2.7582	-0.3675	0.1184	2.6423	7.7182	1
CV-Bl	-3.9883	-2.8749	-0.8770	-0.1039	2.4703	7.4567	0.6488
CV-hvBl	-3.9870	-2.8746	-0.8727	-0.1026	2.5037	7.4569	0.6488
CV-Mod	-3.8457	-2.6583	0.6001	0.2873	2.7284	7.7211	0.8199
Holdout	-4.3466	-2.5977	-0.7025	-0.0763	2.4665	5.2947	0.362
p-holdout	-4.1097	-2.7132	0.7382	0.2058	2.4525	6.9337	0.8199
Preq-Bls	-3.9943	-2.7387	-0.4579	-0.0850	2.6817	7.4220	0.8199
Preq-Bls-Gap	-4.1550	-2.6879	-0.7946	-0.0451	2.4428	7.3584	1
Preq-Grow	-4.3757	-2.8852	-1.9598	-0.7003	1.8648	4.4979	0.022
Preq-Sld-Bls	-3.9882	-2.5803	1.4863	0.4971	2.8625	7.4408	0.362
Preq-Slide	-4.3726	-2.8671	-1.8560	-0.6865	1.9331	4.4965	0.022
Rep-Holdout	-4.2393	-2.6826	-1.1710	-0.0510	2.8502	5.3116	0.4944

Table A.9: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 77 non-stationary time series using the GLM-Ridge learning algorithm and the RMSE as error measure.

A.2 Data from the M4 Competition

A.2.1 Results from the RBR learning algorithm

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.5917	-2.9524	0.4796	0.2035	3.2303	7.0715	0.5909
cp-holdout	-4.5917	-2.9524	0.4796	0.2035	3.2303	7.0715	0.5909
CV	-4.5951	-3.2729	-1.6629	-0.1237	3.2978	8.4690	0.0149
CV-Bl	-4.5976	-3.2546	-1.4026	-0.0889	3.2973	8.8001	0.0341
CV-hvBl	-4.5902	-3.1541	-0.5491	0.0937	3.3798	9.4853	0.548
CV-Mod	-4.5948	-3.0814	0.3085	0.2503	3.5211	8.8682	0.776
Holdout	-4.5856	-3.1060	-0.3559	0.0864	3.1924	7.2647	0.681
p-holdout	-4.5883	-2.9078	0.8380	0.3108	3.3620	7.0715	0.1067
Preq-Bls	-4.5420	-2.6779	2.4471	1.0198	3.8298	9.1734	< 0.0001
Preq-Bls-Gap	-4.3151	-0.4799	3.4204	2.1198	4.3714	10.1493	< 0.0001
Preq-Grow	-4.5834	-3.4204	-2.2355	-0.5813	2.9223	7.4561	< 0.0001
Preq-Sld-Bls	-4.5243	2.1373	3.7232	2.5694	4.5144	9.6810	< 0.0001
Preq-Slide	-4.5835	-3.3925	-2.1972	-0.5741	2.9363	7.2773	< 0.0001
Rep-Holdout	-4.5495	-2.6693	2.0504	0.8187	3.7284	8.5748	< 0.0001

Table A.10: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the time series from the M4 Competition sample using the RBR learning algorithm and the RMSE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.3651	-2.9729	-0.6754	0.0763	3.2617	6.0731	0.5594
cp-holdout	-4.3651	-2.9729	-0.6754	0.0763	3.2617	6.0731	0.5594
CV	-4.3742	-3.2596	-1.8215	-0.2359	3.1845	6.3080	0.0097
CV-Bl	-4.3722	-3.2406	-1.6933	-0.1922	3.2827	5.9256	0.0097
CV-hvBl	-4.3651	-3.1577	-1.2666	-0.0223	3.3595	6.4974	0.1332
CV-Mod	-4.2688	-3.0755	-1.0444	0.1054	3.4307	6.4068	0.4043
Holdout	-4.4864	-3.1084	-1.0543	-0.0057	3.2012	5.4942	0.2782
p-holdout	-4.3651	-2.9523	0.8292	0.2720	3.3354	6.0731	0.1332
Preq-Bls	-4.3515	-2.7069	2.1134	0.9098	3.8038	6.6895	< 0.0001
Preq-Bls-Gap	-4.3151	-0.7745	3.3990	2.0736	4.2897	7.2154	< 0.0001
Preq-Grow	-4.4693	-3.4250	-2.3280	-0.6457	2.8993	5.9318	< 0.0001
Preq-Sld-Bls	-4.2280	1.9815	3.6946	2.5404	4.4395	7.0249	< 0.0001
Preq-Slide	-4.4352	-3.3977	-2.2750	-0.6483	2.8767	5.7708	< 0.0001
Rep-Holdout	-4.3582	-2.6007	1.9663	0.7639	3.6361	5.7947	2e-04

Table A.11: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the stationary time series from the M4 Competition sample using the RBR learning algorithm and the RMSE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.5917	-2.8732	1.1455	0.3755	3.2279	7.0715	0.1205
cp-holdout	-4.5917	-2.8732	1.1455	0.3755	3.2279	7.0715	0.1205
CV	-4.5951	-3.2736	-1.0673	0.0282	3.3835	8.4690	0.4971
CV-Bl	-4.5976	-3.2757	-0.7363	0.0508	3.3257	8.8001	0.8462
CV-hvBl	-4.5902	-3.1441	1.3322	0.2506	3.4265	9.4853	0.4377
CV-Mod	-4.5948	-3.1310	1.6525	0.4463	3.5802	8.8682	0.1455
Holdout	-4.5856	-3.0823	0.5553	0.2109	3.1462	7.2647	0.5606
p-holdout	-4.5883	-2.8815	0.8779	0.3634	3.4116	7.0715	0.4971
Preq-Bls	-4.5420	-2.6060	2.5388	1.1685	3.9087	9.1734	< 0.0001
Preq-Bls-Gap	-4.1520	-0.3619	3.4230	2.1824	4.4196	10.1493	< 0.0001
Preq-Grow	-4.5834	-3.4047	-2.1696	-0.4941	3.0458	7.4561	2e-04
Preq-Sld-Bls	-4.5243	2.2839	3.7803	2.6087	4.5558	9.6810	< 0.0001
Preq-Slide	-4.5835	-3.3875	-2.1384	-0.4738	3.0533	7.2773	7e-04
Rep-Holdout	-4.5495	-2.8358	2.3129	0.8928	3.8142	8.5748	7e-04

Table A.12: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the non-stationary time series from the M4 Competition sample using the RBR learning algorithm and the RMSE as error measure.

A.2.2 Results from the RF learning algorithm

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.3683	-3.0737	-0.6461	0.0688	3.2951	6.0577	0.467
cp-holdout	-4.3683	-3.0737	-0.6461	0.0688	3.2951	6.0577	0.467
CV	-4.4250	-3.2524	-1.5849	-0.1394	3.2530	7.5430	0.0072
CV-Bl	-4.4575	-3.3073	-1.8755	-0.2461	3.2232	7.0328	2e-04
CV-hvBl	-4.4479	-3.2402	-1.4783	-0.0806	3.3213	7.0533	0.0177
CV-Mod	-4.3865	-3.0975	0.9363	0.2963	3.5030	7.5706	0.3591
Holdout	-4.4038	-3.2678	-0.3064	0.0215	3.2229	5.9394	0.681
p-holdout	-4.3635	-3.0606	-0.5523	0.1048	3.3173	6.0577	0.728
Preq-Bls	-4.4013	-3.0282	1.4613	0.4588	3.5884	6.8916	0.0247
Preq-Bls-Gap	-4.3270	-2.8070	2.1684	0.8229	3.8514	6.9983	< 0.0001
Preq-Grow	-4.4057	-3.4788	-2.2655	-0.5930	2.7636	5.4483	< 0.0001
Preq-Sld-Bls	-4.3769	-2.6465	2.5897	1.1209	4.0041	7.3619	< 0.0001
Preq-Slide	-4.4137	-3.4869	-2.2331	-0.6032	2.7851	5.4307	< 0.0001
Rep-Holdout	-4.4260	-2.9519	1.3120	0.4891	3.5840	7.5171	0.0537

Table A.13: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the time series from the M4 Competition sample using the RF learning algorithm and the RMSE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.3043	-3.1987	-1.5933	-0.1646	3.1744	5.9160	0.0155
cp-holdout	-4.3043	-3.1987	-1.5933	-0.1646	3.1744	5.9160	0.0155
CV	-4.2817	-3.2947	-2.0440	-0.3159	3.1582	6.1654	0.0026
CV-Bl	-4.3362	-3.3144	-2.1401	-0.4187	3.0935	6.0249	1e-04
CV-hvBl	-4.3347	-3.2661	-1.8480	-0.2544	3.1771	6.0669	0.0035
CV-Mod	-4.2475	-3.1135	-0.6095	0.1169	3.4445	6.2005	0.6168
Holdout	-4.3509	-3.3500	-1.2457	-0.1452	3.2025	5.7655	0.2109
p-holdout	-4.2477	-3.1269	-1.3553	-0.0949	3.2705	5.9160	0.055
Preq-Bls	-4.2799	-3.0590	1.3166	0.3428	3.5348	6.0778	0.3589
Preq-Bls-Gap	-4.2192	-2.8326	2.0478	0.7161	3.7489	6.2391	0.0011
Preq-Grow	-4.4025	-3.5273	-2.4453	-0.7135	2.6991	5.3743	< 0.0001
Preq-Sld-Bls	-4.2519	-2.7275	2.5094	1.0169	3.9954	6.6408	< 0.0001
Preq-Slide	-4.4044	-3.5383	-2.3937	-0.7330	2.6827	5.4106	< 0.0001
Rep-Holdout	-4.2662	-2.9519	0.3998	0.3400	3.4846	5.6100	0.7387

Table A.14: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the stationary time series from the M4 Competition sample using the RF learning algorithm and the RMSE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.3683	-2.9742	1.4114	0.3846	3.3915	6.0577	0.099
cp-holdout	-4.3683	-2.9742	1.4114	0.3846	3.3915	6.0577	0.099
CV	-4.4250	-3.1955	-0.5468	0.0994	3.4132	7.5430	0.5606
CV-Bl	-4.4575	-3.2916	-1.3597	-0.0126	3.3664	7.0328	0.2072
CV-hvBl	-4.4479	-3.1977	-0.3026	0.1544	3.4206	7.0533	0.8462
CV-Mod	-4.3865	-3.0466	1.5635	0.5390	3.6121	7.5706	0.0415
Holdout	-4.4038	-3.1633	1.2192	0.2471	3.2266	5.9394	0.4377
p-holdout	-4.3635	-2.9742	1.4312	0.3751	3.3486	6.0577	0.099
Preq-Bls	-4.4013	-2.9815	1.8266	0.6157	3.7150	6.8916	0.0198
Preq-Bls-Gap	-4.3270	-2.7253	2.3016	0.9673	3.9106	6.9983	2e-04
Preq-Grow	-4.4057	-3.3027	-1.9308	-0.4300	2.8436	5.4483	0.0026
Preq-Sld-Bls	-4.3769	-2.5138	2.8276	1.2615	4.0176	7.3619	< 0.0001
Preq-Slide	-4.4137	-3.3177	-1.9453	-0.4277	2.9269	5.4307	0.0036
Rep-Holdout	-4.4260	-2.9433	1.8410	0.6909	3.7233	7.5171	0.0116

Table A.15: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the non-stationary time series from the M4 Competition sample using the RF learning algorithm and the RMSE as error measure.

A.2.3 Results from the GLM learning algorithm

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.4977	-2.8912	-0.4528	0.0732	3.0518	6.4250	0.728
cp-holdout	-4.4977	-2.8912	-0.4528	0.0732	3.0518	6.4250	0.728
CV	-4.5206	-3.0319	0.1078	0.2050	3.3758	6.7023	0.9748
CV-Bl	-4.5365	-3.1370	-1.0369	0.0134	3.2443	6.6840	0.242
CV-hvBl	-4.5364	-3.1415	-0.8752	0.0348	3.2662	6.7793	0.1739
CV-Mod	-4.5164	-2.9614	0.9986	0.3407	3.4187	6.8172	0.1547
Holdout	-4.5034	-3.0210	-1.0939	-0.0116	3.1119	6.5797	0.2174
p-holdout	-4.4770	-2.8949	-0.2615	0.0941	3.0318	6.3170	0.8744
Preq-Bls	-4.5331	-2.9391	1.0874	0.3435	3.4257	6.6791	0.1547
Preq-Bls-Gap	-4.5159	-2.8224	1.4500	0.5113	3.5327	6.6856	0.0086
Preq-Grow	-4.5384	-3.3441	-2.2116	-0.5346	2.9187	6.0797	< 0.0001
Preq-Sld-Bls	-4.5266	-2.4489	2.5382	1.1617	3.8156	6.7809	< 0.0001
Preq-Slide	-4.5385	-3.3454	-2.1871	-0.5346	2.9195	6.0475	< 0.0001
Rep-Holdout	-4.4537	-2.7598	1.5774	0.5079	3.4454	7.4545	0.0014

Table A.16: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the time series from the M4 Competition sample using the GLM-RIDGE learning algorithm and the RMSE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.2659	-2.9836	-1.1778	-0.0672	2.9599	5.3166	0.2109
cp-holdout	-4.2659	-2.9836	-1.1778	-0.0672	2.9599	5.3166	0.2109
CV	-4.2877	-3.0369	-0.8544	0.0718	3.2285	5.5304	0.4529
CV-Bl	-4.3250	-3.1219	-1.4217	-0.1360	3.0995	5.4313	0.037
CV-hvBl	-4.3256	-3.1321	-1.4177	-0.1135	3.1153	5.4581	0.03
CV-Mod	-4.2682	-2.9674	0.1940	0.2023	3.3175	5.5615	0.9335
Holdout	-4.3536	-3.0675	-1.5334	-0.1383	3.0812	5.3572	0.0952
p-holdout	-4.2659	-2.9588	-0.4276	0.0072	2.9694	5.3799	0.6168
Preq-Bls	-4.2987	-2.9501	0.4977	0.2228	3.2572	6.1020	0.8675
Preq-Bls-Gap	-4.2787	-2.7929	1.0189	0.4031	3.4084	5.7739	0.1562
Preq-Grow	-4.3647	-3.3884	-2.2890	-0.6418	2.7972	5.3904	< 0.0001
Preq-Sld-Bls	-4.2516	-2.4848	2.3662	1.0462	3.6985	6.4157	< 0.0001
Preq-Slide	-4.3541	-3.3845	-2.2934	-0.6399	2.7989	5.3897	< 0.0001
Rep-Holdout	-4.1904	-2.7529	1.5253	0.4469	3.3204	5.4815	0.0242

Table A.17: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the stationary time series from the M4 Competition sample using the GLM-RIDGE learning algorithm and the RMSE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.4977	-2.7566	1.0714	0.2632	3.1635	6.4250	0.3826
cp-holdout	-4.4977	-2.7566	1.0714	0.2632	3.1635	6.4250	0.3826
CV	-4.5206	-3.0287	1.0267	0.3852	3.5113	6.7023	0.332
CV-Bl	-4.5365	-3.1703	0.9397	0.2157	3.3868	6.6840	0.5606
CV-hvBl	-4.5364	-3.1520	0.9020	0.2355	3.4068	6.7793	0.698
CV-Mod	-4.5164	-2.9511	1.5349	0.5280	3.5557	6.8172	0.0415
Holdout	-4.5034	-2.9102	0.2526	0.1598	3.1941	6.5797	1
p-holdout	-4.4770	-2.8551	0.7024	0.2118	3.1478	6.3170	0.7711
Preq-Bls	-4.5331	-2.9201	1.7547	0.5067	3.5152	6.6791	0.0522
Preq-Bls-Gap	-4.5159	-2.8557	1.8345	0.6578	3.6286	6.6856	0.0198
Preq-Grow	-4.5384	-3.2556	-2.0411	-0.3894	3.0444	6.0797	0.0087
Preq-Sld-Bls	-4.5266	-2.4197	2.7623	1.3180	3.9452	6.7809	0
Preq-Slide	-4.5385	-3.2392	-2.0483	-0.3921	3.0220	6.0475	0.0087
Rep-Holdout	-4.4537	-2.7662	1.6796	0.5904	3.5620	7.4545	0.0255

Table A.18: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the non-stationary time series from the M4 Competition sample using the GLM-RIDGE learning algorithm and the RMSE as error measure.

Appendix B: Part I - Results for all cases using MASE

B.1 Data from Cerqueira et al. (2020)

B.1.1 Results from the RBR learning algorithm

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.5723	-2.5755	-0.4583	0.0498	2.6956	6.4568	0.5958
cp-holdout	-4.5723	-2.6252	-0.0724	0.0244	2.7307	5.9954	0.8202
CV	-4.6032	-4.0570	-3.6145	-3.1518	-3.0563	6.5720	< 0.0001
CV-Bl	-4.5371	-2.5622	0.3650	0.0362	2.4823	7.3776	0.8202
CV-hvBl	-4.5368	-2.5481	0.8733	0.1204	2.4952	7.3856	0.4044
CV-Mod	-4.6031	-3.9966	-3.5288	-2.9220	-2.9475	6.6844	< 0.0001
Holdout	-4.5723	-2.7067	-1.2840	-0.1086	2.7038	5.2328	0.4952
p-holdout	-4.5723	-2.7051	-0.5560	-0.0509	2.6577	7.2400	0.4952
Preq-Bls	-4.5376	-2.2538	1.4321	0.6898	3.0297	6.9546	0.0577
Preq-Bls-Gap	-4.5146	-2.2472	1.5659	0.8087	3.0944	6.9282	0.0185
Preq-Grow	-4.5684	-2.8964	-1.5932	-0.3239	2.6028	5.1708	0.0577
Preq-Sld-Bls	-4.5120	2.0185	3.2030	2.5119	3.8669	7.8570	< 0.0001
Preq-Slide	-4.5678	-2.8132	-1.4559	-0.1874	2.5851	4.7553	0.2554
Rep-Holdout	-4.5689	-2.4770	-0.7885	0.1775	2.7797	8.1900	0.7048

Table B.1: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to all 174 real-world time series using the RBR learning algorithm and the MASE as error measure.

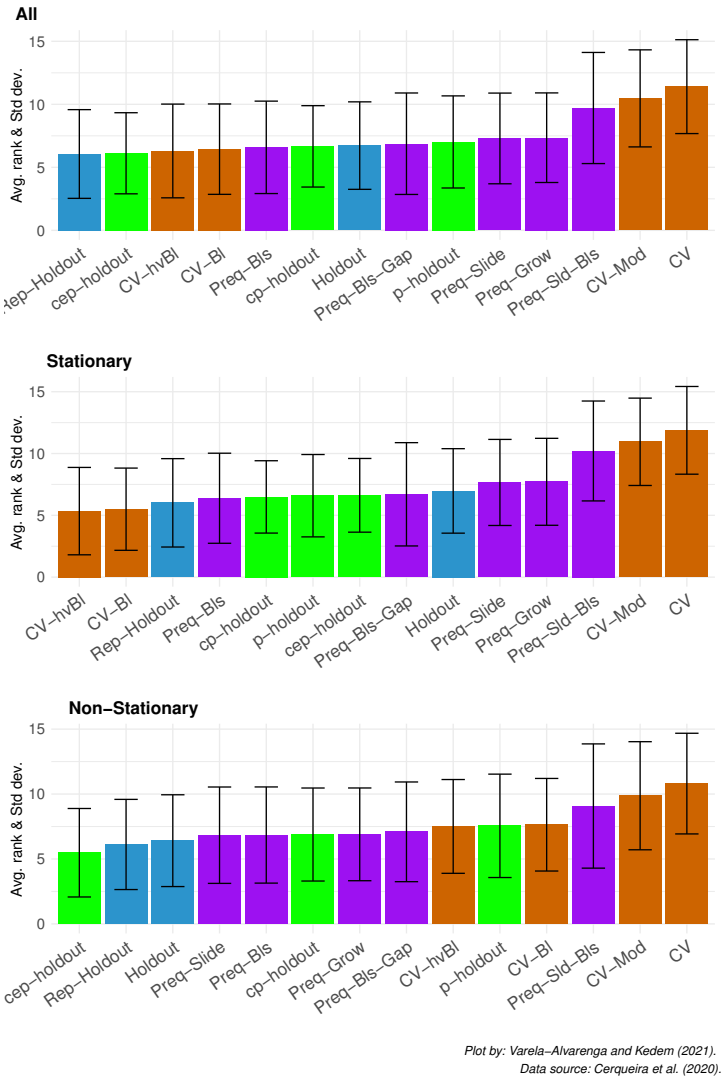


Figure B.1: Average APAE rank of each validation scheme on 174 real-world time series using the RBR learning algorithm and MASE as the error function. The black line represents -+ 1 standard deviation from the average.

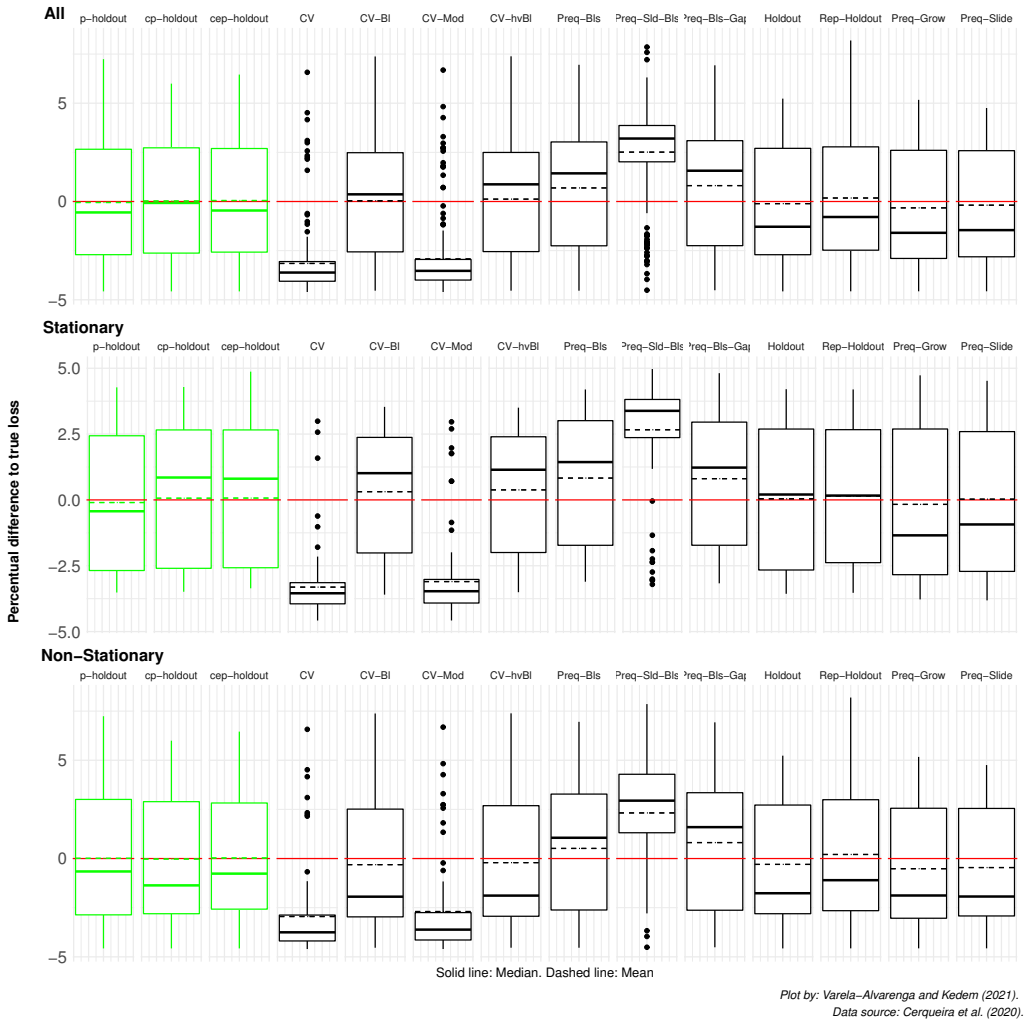


Figure B.2: Log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to 174 real-world time series using the RBR learning algorithm. Values below the zero (red) line represent under-estimations of the error. Conversely, values above it represent over-estimations of the error.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-3.3569	-2.5759	0.8053	0.0685	2.6573	4.8705	0.8392
cp-holdout	-3.4879	-2.5978	0.8502	0.0647	2.6573	4.2869	0.8392
CV	-4.5798	-3.9461	-3.5451	-3.3124	-3.1427	2.9898	< 0.0001
CV-Bl	-3.5970	-2.0139	1.0152	0.3116	2.3736	3.5332	0.2229
CV-hvBl	-3.5050	-1.9978	1.1456	0.3812	2.3974	3.5029	0.0671
CV-Mod	-4.5784	-3.9155	-3.4674	-3.1014	-3.0201	2.9640	< 0.0001
Holdout	-3.5652	-2.6615	0.2044	0.0420	2.6910	4.2043	0.8392
p-holdout	-3.5163	-2.6819	-0.4311	-0.0999	2.4360	4.2740	0.8392
Preq-Bls	-3.1065	-1.7238	1.4364	0.8289	3.0096	4.1937	0.025
Preq-Bls-Gap	-3.1673	-1.7219	1.2271	0.8062	2.9538	4.8129	0.0417
Preq-Grow	-3.7782	-2.8400	-1.3461	-0.1705	2.6928	4.7301	0.4168
Preq-Sld-Bls	-3.2078	2.3659	3.3821	2.6636	3.8145	4.9703	< 0.0001
Preq-Slide	-3.8148	-2.7149	-0.9317	0.0281	2.5935	4.5225	0.8392
Rep-Holdout	-3.5285	-2.3827	0.1644	0.1540	2.6671	4.1948	1

Table B.2: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 97 stationary time series using the RBR learning algorithm and the MASE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.5723	-2.5744	-0.7677	0.0262	2.8260	6.4568	0.2543
cp-holdout	-4.5723	-2.8068	-1.3651	-0.0263	2.8931	5.9954	0.4944
CV	-4.6032	-4.1921	-3.7528	-2.9495	-2.8750	6.5720	< 0.0001
CV-Bl	-4.5371	-2.9674	-1.9399	-0.3107	2.5136	7.3776	0.362
CV-hvBl	-4.5368	-2.9361	-1.8853	-0.2081	2.6890	7.3856	0.4944
CV-Mod	-4.6031	-4.1438	-3.6161	-2.6960	-2.7501	6.6844	< 0.0001
Holdout	-4.5723	-2.8068	-1.7653	-0.2982	2.7208	5.2328	0.1711
p-holdout	-4.5723	-2.8645	-0.6571	0.0108	3.0075	7.2400	0.4944
Preq-Bls	-4.5376	-2.6173	1.0564	0.5146	3.2786	6.9546	0.8199
Preq-Bls-Gap	-4.5146	-2.6292	1.5972	0.8118	3.3487	6.9282	0.2543
Preq-Grow	-4.5684	-3.0365	-1.8800	-0.5171	2.5575	5.1708	0.0675
Preq-Sld-Bls	-4.5120	1.3117	2.9447	2.3208	4.2867	7.8570	< 0.0001
Preq-Slide	-4.5678	-2.9212	-1.9325	-0.4589	2.5488	4.7553	0.1711
Rep-Holdout	-4.5689	-2.6525	-1.1010	0.2071	2.9921	8.1900	0.4944

Table B.3: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 77 non-stationary time series using the RBR learning algorithm and the MASE as error measure.

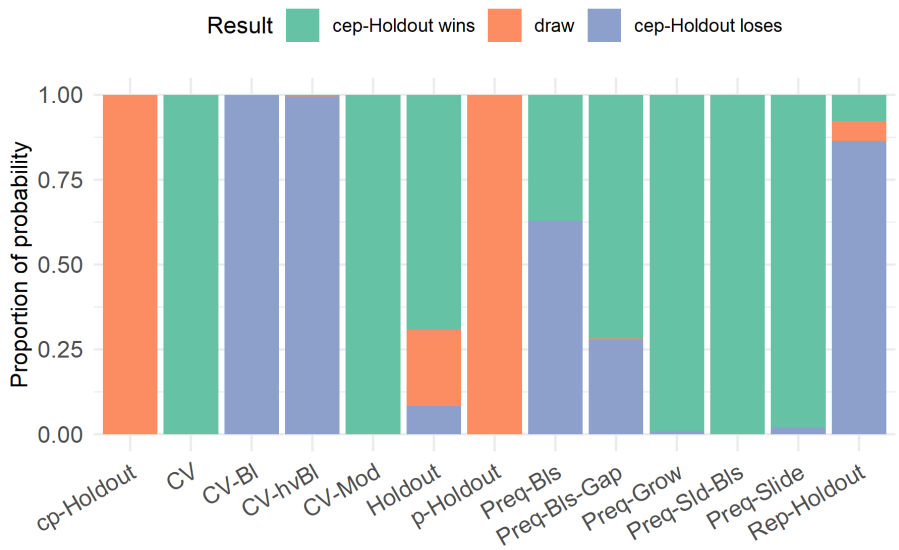


Figure B.3: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation scheme when applied to the stationary time-series from Cerqueira et al. (2020), with parameters estimated via the RBR learning algorithm and error calculated using the MASE. The probabilities are computed using the Bayes signed-rank test.

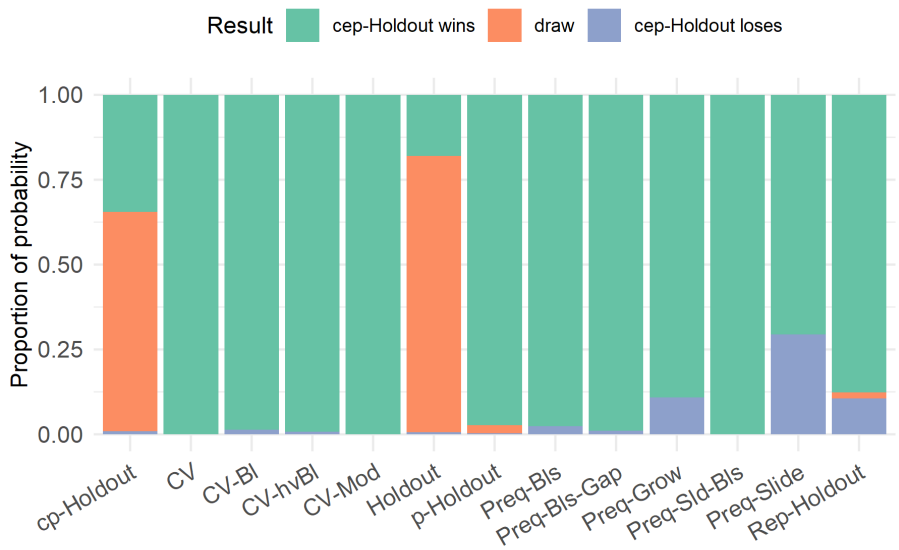


Figure B.4: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation schemes when applied to the non-stationary time-series from Cerqueira et al. (2020), with parameters estimated via the RBR learning algorithm and error calculated using the MASE. The probabilities are computed using the Bayes signed-rank test.

B.1.2 Results from the RF learning algorithm

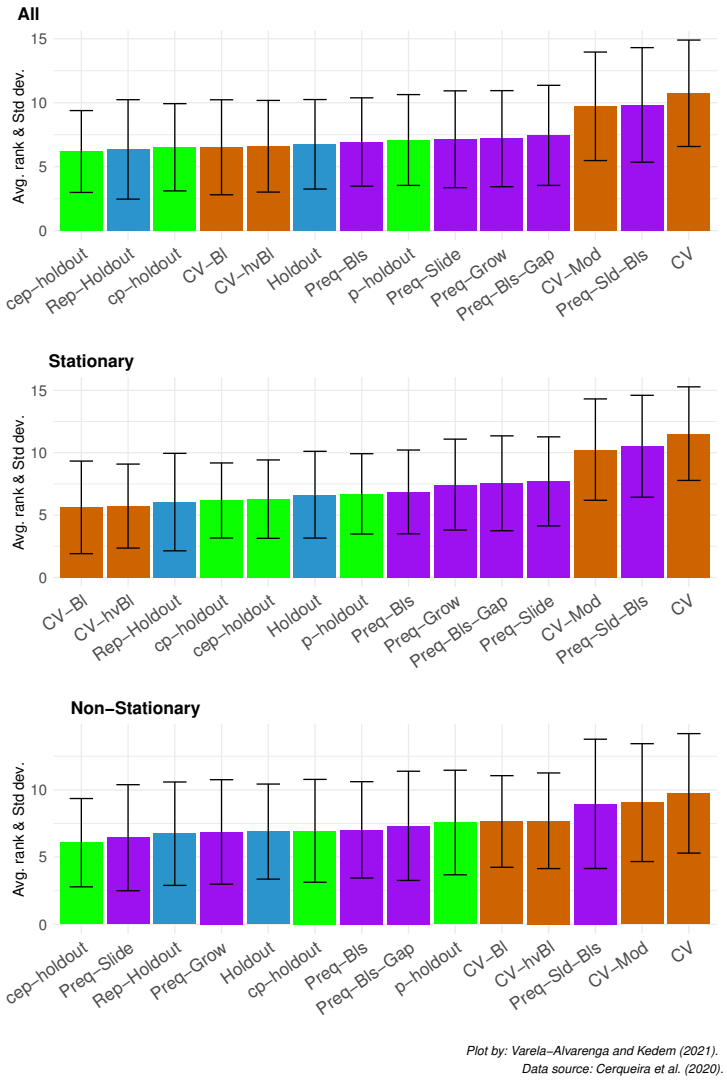


Figure B.5: Average APAE rank of each validation scheme on 174 real-world time series using the RF learning algorithm and MASE as the error function. The black line represents -+ 1 standard deviation from the average.

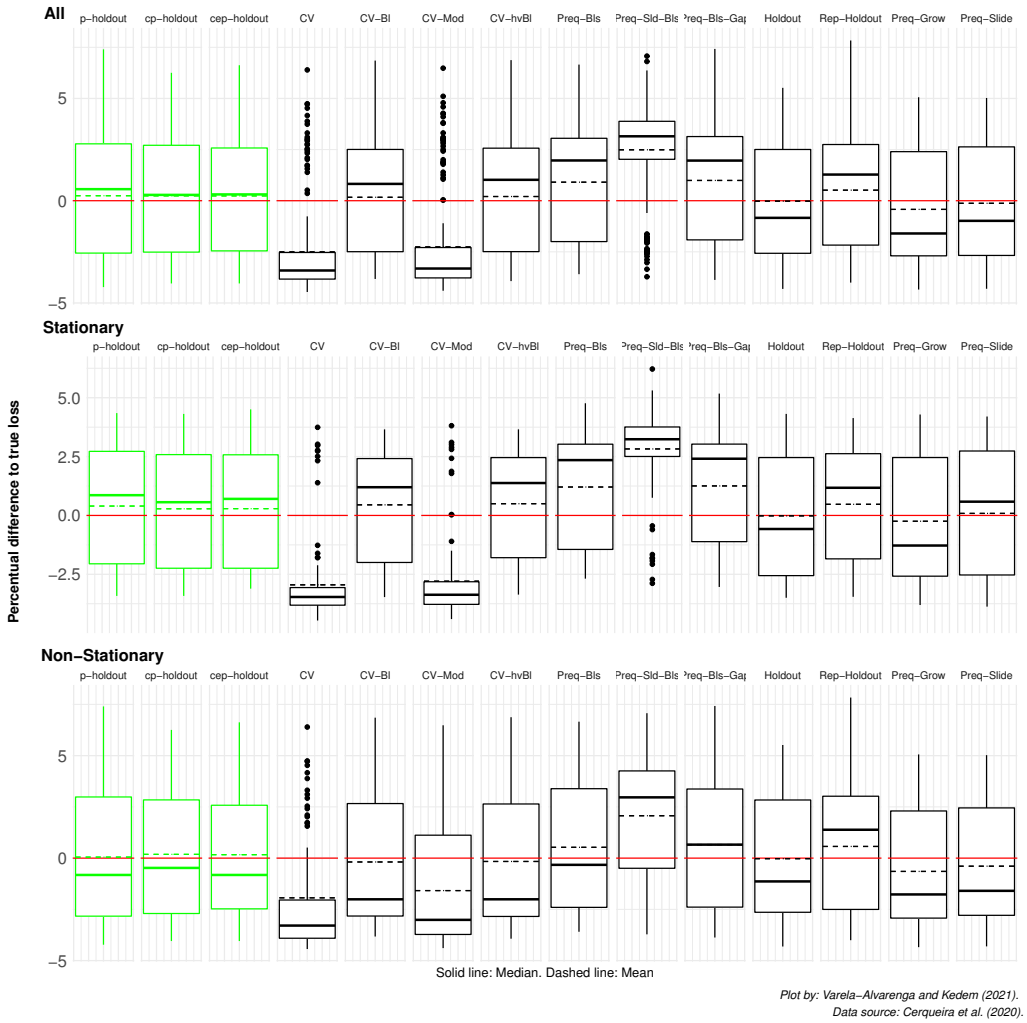


Figure B.6: Log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to 174 real-world time series using the RF learning algorithm. Values below the zero (red) line represent under-estimations of the error. Conversely, values above it represent over-estimations of the error.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.0452	-2.4522	0.3085	0.2351	2.5787	6.6253	0.9396
cp-holdout	-4.0452	-2.5129	0.2852	0.2385	2.7101	6.2548	0.9396
CV	-4.4630	-3.8328	-3.4115	-2.5006	-2.5318	6.3957	< 0.0001
CV-Bl	-3.8252	-2.4927	0.8222	0.1678	2.5070	6.8520	0.5958
CV-hvBl	-3.9305	-2.4889	1.0221	0.2055	2.5698	6.8748	0.4044
CV-Mod	-4.4020	-3.7744	-3.3154	-2.2574	-2.2897	6.4808	< 0.0001
Holdout	-4.3119	-2.5707	-0.8383	-0.0218	2.5054	5.5194	0.4044
p-holdout	-4.2242	-2.5608	0.5651	0.2440	2.7810	7.4009	0.8202
Preq-Bls	-3.5924	-1.9985	1.9680	0.9066	3.0526	6.6564	0.0276
Preq-Bls-Gap	-3.8724	-1.9149	1.9625	0.9909	3.1372	7.4208	0.0018
Preq-Grow	-4.3438	-2.6949	-1.6014	-0.4246	2.3988	5.0591	0.0049
Preq-Sld-Bls	-3.7184	2.0250	3.1501	2.4896	3.8825	7.0703	< 0.0001
Preq-Slide	-4.3054	-2.6757	-0.9839	-0.1257	2.6327	5.0239	0.3244
Rep-Holdout	-4.0041	-2.1707	1.2788	0.5159	2.7481	7.8342	0.1111

Table B.4: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to all 174 real-world time series using the RF learning algorithm and the MASE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-3.1095	-2.2424	0.7032	0.2884	2.5763	4.5045	0.5426
cp-holdout	-3.4183	-2.2424	0.5624	0.2824	2.5840	4.3126	0.6849
CV	-4.4630	-3.8117	-3.4638	-2.9477	-3.0641	3.7423	< 0.0001
CV-Bl	-3.4676	-1.9974	1.1986	0.4503	2.4156	3.6555	0.0671
CV-hvBl	-3.3622	-1.7943	1.3768	0.4983	2.4567	3.6569	0.025
CV-Mod	-4.4020	-3.7769	-3.3695	-2.7922	-2.8163	3.8111	< 0.0001
Holdout	-3.4967	-2.5562	-0.5759	-0.0175	2.4578	4.3126	0.6849
p-holdout	-3.4183	-2.0544	0.8605	0.3984	2.7218	4.3473	0.4168
Preq-Bls	-2.6844	-1.4435	2.3501	1.2073	3.0260	4.7647	0.0022
Preq-Bls-Gap	-3.0378	-1.1133	2.4090	1.2532	3.0296	5.1751	2e-04
Preq-Grow	-3.8047	-2.5800	-1.2784	-0.2463	2.4587	4.2873	0.2229
Preq-Sld-Bls	-2.8833	2.5048	3.2367	2.8247	3.7576	6.2192	< 0.0001
Preq-Slide	-3.8732	-2.5286	0.5847	0.0867	2.7369	4.2021	1
Rep-Holdout	-3.4561	-1.8441	1.1751	0.4732	2.6233	4.1358	0.2229

Table B.5: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 97 stationary time series using the RF learning algorithm and the MASE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.0452	-2.4738	-0.8212	0.1680	2.5795	6.6253	0.6488
cp-holdout	-4.0452	-2.7016	-0.4780	0.1832	2.8407	6.2548	0.8199
CV	-4.4349	-3.9084	-3.2907	-1.9375	-2.0436	6.3957	< 0.0001
CV-Bl	-3.8252	-2.8252	-2.0090	-0.1881	2.6622	6.8520	0.2543
CV-hvBl	-3.9305	-2.8449	-2.0086	-0.1634	2.6417	6.8748	0.2543
CV-Mod	-4.3899	-3.7237	-3.0123	-1.5838	1.1190	6.4808	1e-04
Holdout	-4.3119	-2.6419	-1.1363	-0.0273	2.8362	5.5194	0.4944
p-holdout	-4.2242	-2.8313	-0.8212	0.0494	2.9837	7.4009	0.6488
Preq-Bls	-3.5924	-2.4026	-0.3275	0.5278	3.3873	6.6564	1
Preq-Bls-Gap	-3.8724	-2.3902	0.6579	0.6605	3.3713	7.4208	0.6488
Preq-Grow	-4.3438	-2.9261	-1.7716	-0.6492	2.3009	5.0591	0.0059
Preq-Sld-Bls	-3.7184	-0.4954	2.9643	2.0674	4.2547	7.0703	2e-04
Preq-Slide	-4.3054	-2.7906	-1.5958	-0.3931	2.4499	5.0239	0.11
Rep-Holdout	-4.0041	-2.5025	1.3845	0.5697	3.0195	7.8342	0.362

Table B.6: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 77 non-stationary time series using the RF learning algorithm and the MASE as error measure.

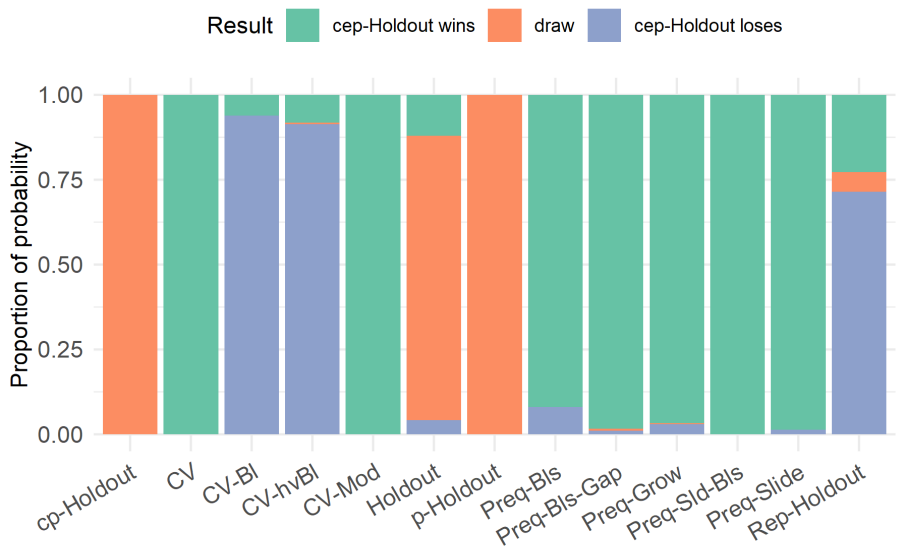


Figure B.7: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation scheme when applied to the stationary time-series from Cerqueira et al. (2020), with parameters estimated via the RF learning algorithm and error calculated using the MASE. The probabilities are computed using the Bayes signed-rank test.

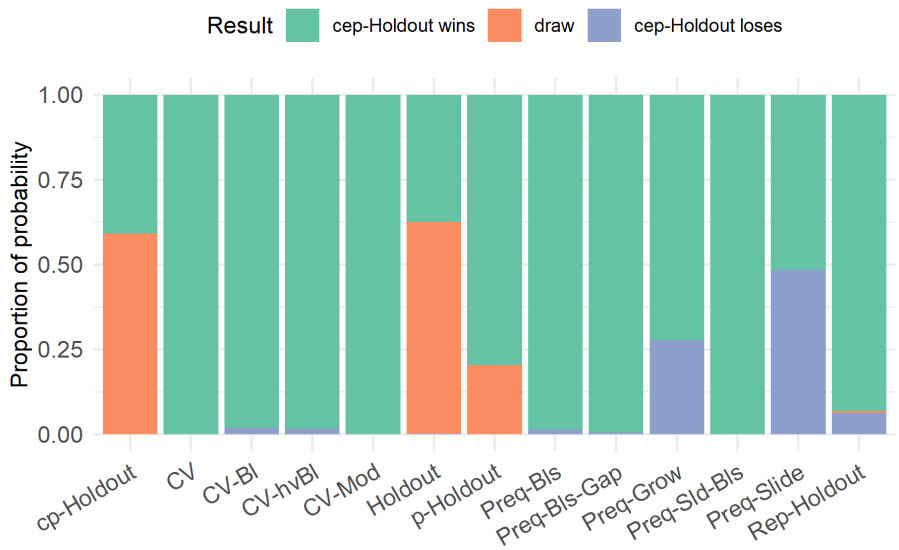


Figure B.8: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation schemes when applied to the non-stationary time-series from Cerqueira et al. (2020), with parameters estimated via the RF learning algorithm and error calculated using the MASE. The probabilities are computed using the Bayes signed-rank test.

B.1.3 Results from the GLM learning algorithm

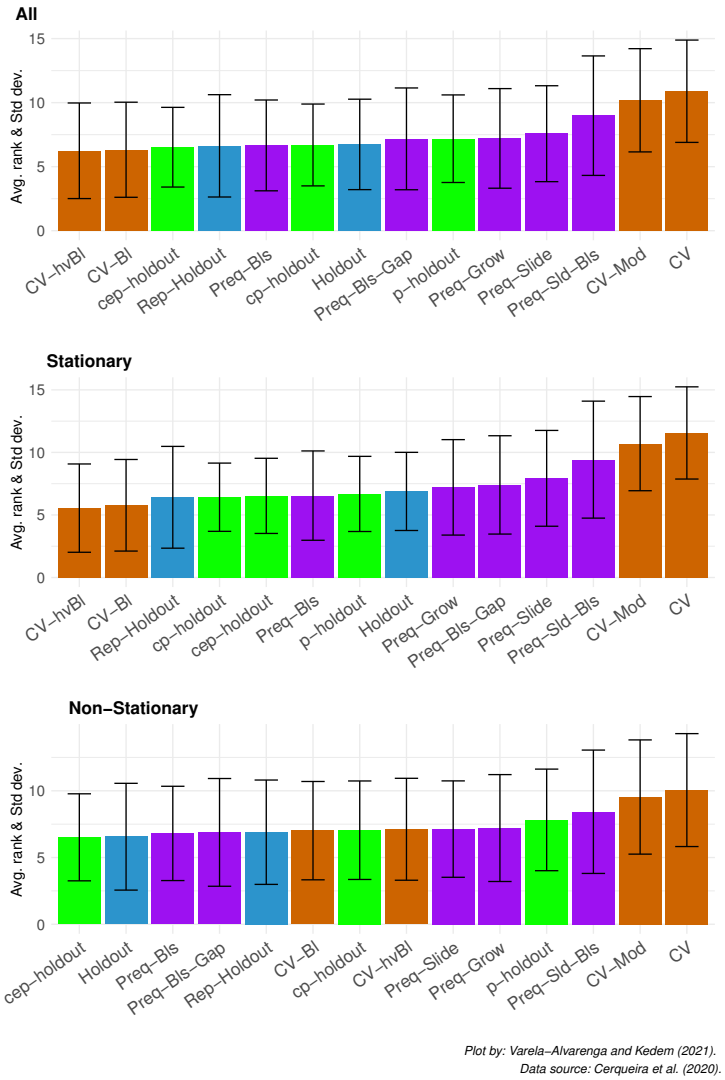


Figure B.9: Average APAE rank of each validation scheme on 174 real-world time series using the GLM-Ridge learning algorithm and MASE as the error function. The black line represents -+ 1 standard deviation from the average.

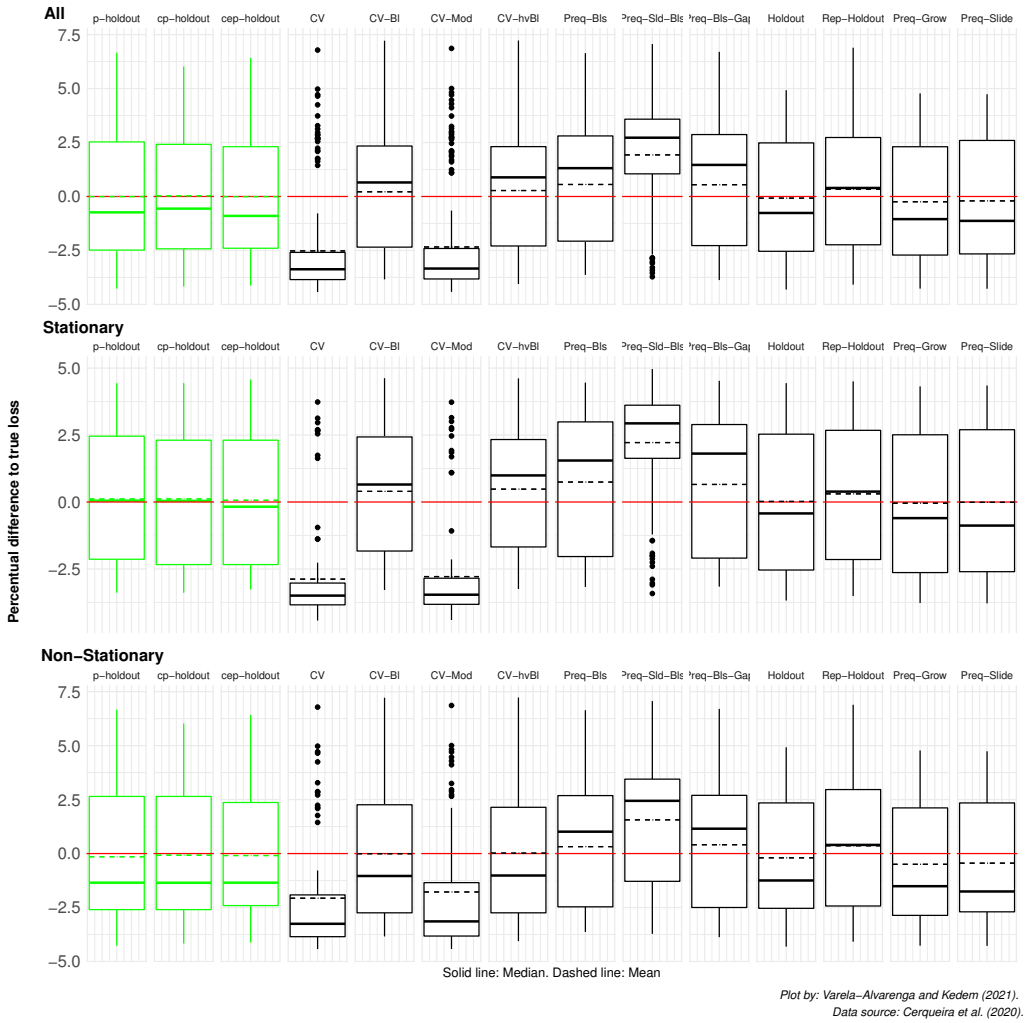


Figure B.10: Log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to 174 real-world time series using the GLM-Ridge learning algorithm. Values below the zero (red) line represent under-estimations of the error. Conversely, values above it represent over-estimations of the error.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.1262	-2.3978	-0.8995	-0.0050	2.3076	6.4257	0.4952
cp-holdout	-4.1684	-2.4256	-0.5627	0.0300	2.4157	6.0219	0.9396
CV	-4.4242	-3.8497	-3.3711	-2.5201	-2.5901	6.7832	< 0.0001
CV-Bl	-3.8388	-2.3461	0.6500	0.2179	2.3389	7.2172	0.4044
CV-hvBl	-4.0557	-2.2928	0.8862	0.2763	2.3101	7.2266	0.2554
CV-Mod	-4.4182	-3.8203	-3.3385	-2.3392	-2.4087	6.8601	< 0.0001
Holdout	-4.3088	-2.5386	-0.7634	-0.0761	2.4826	4.9240	0.3244
p-holdout	-4.2708	-2.4840	-0.7346	-0.0037	2.5280	6.6673	0.8202
Preq-Bls	-3.6326	-2.0697	1.3126	0.5568	2.8058	6.6408	0.0577
Preq-Bls-Gap	-3.8718	-2.2742	1.4629	0.5459	2.8698	6.7019	0.1495
Preq-Grow	-4.2664	-2.7134	-1.0482	-0.2447	2.3046	4.7756	0.1973
Preq-Sld-Bls	-3.7203	1.0490	2.7238	1.9278	3.5816	7.0618	< 0.0001
Preq-Slide	-4.2744	-2.6599	-1.1299	-0.2039	2.5963	4.7368	0.0577
Rep-Holdout	-4.0829	-2.2367	0.3946	0.3300	2.7330	6.8926	0.4952

Table B.7: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to all 174 real-world time series using the GLM-Ridge learning algorithm and the MASE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-3.2664	-2.3362	-0.1756	0.0642	2.3076	4.5735	1
cp-holdout	-3.3877	-2.3362	0.0433	0.1147	2.3076	4.4372	0.8392
CV	-4.4242	-3.8393	-3.4945	-2.8789	-3.0264	3.7327	< 0.0001
CV-Bl	-3.2845	-1.8304	0.6528	0.3967	2.4292	4.6228	0.2229
CV-hvBl	-3.2481	-1.6781	0.9926	0.4802	2.3321	4.6151	0.1038
CV-Mod	-4.4056	-3.8203	-3.4606	-2.7840	-2.8449	3.7268	< 0.0001
Holdout	-3.6792	-2.5392	-0.4265	0.0207	2.5320	4.4372	0.6849
p-holdout	-3.3877	-2.1372	0.0512	0.1114	2.4562	4.4372	0.8392
Preq-Bls	-3.1709	-2.0364	1.5484	0.7446	2.9908	4.4556	0.0144
Preq-Bls-Gap	-3.1587	-2.0941	1.8076	0.6553	2.8919	4.5238	0.1548
Preq-Grow	-3.7711	-2.6346	-0.6020	-0.0445	2.5094	4.3157	0.8392
Preq-Sld-Bls	-3.4167	1.6307	2.9361	2.2190	3.6136	4.9646	< 0.0001
Preq-Slide	-3.7827	-2.6011	-0.8803	-0.0132	2.6988	4.3486	0.3099
Rep-Holdout	-3.5104	-2.1471	0.3885	0.3048	2.6758	4.5017	0.5426

Table B.8: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 97 stationary time series using the GLM-Ridge learning algorithm and the MASE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.1262	-2.4093	-1.3489	-0.0922	2.3649	6.4257	0.362
cp-holdout	-4.1684	-2.5962	-1.3524	-0.0767	2.6464	6.0219	0.6488
CV	-4.4221	-3.8500	-3.2538	-2.0682	-1.9150	6.7832	< 0.0001
CV-Bl	-3.8388	-2.7446	-1.0385	-0.0073	2.2603	7.2172	1
CV-hvBl	-4.0557	-2.7451	-1.0170	0.0195	2.1431	7.2266	1
CV-Mod	-4.4182	-3.8205	-3.1421	-1.7790	-1.3468	6.8601	< 0.0001
Holdout	-4.3088	-2.5367	-1.2467	-0.1981	2.3459	4.9240	0.362
p-holdout	-4.2708	-2.5962	-1.3489	-0.1486	2.6464	6.6673	0.4944
Preq-Bls	-3.6326	-2.4664	1.0140	0.3202	2.6854	6.6408	1
Preq-Bls-Gap	-3.8718	-2.4985	1.1515	0.4080	2.6962	6.7019	0.6488
Preq-Grow	-4.2664	-2.8632	-1.5121	-0.4969	2.1151	4.7756	0.11
Preq-Sld-Bls	-3.7203	-1.2861	2.4418	1.5610	3.4472	7.0618	0.0028
Preq-Slide	-4.2744	-2.6991	-1.7563	-0.4441	2.3450	4.7368	0.11
Rep-Holdout	-4.0829	-2.4284	0.4007	0.3617	2.9647	6.8926	0.8199

Table B.9: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the 77 non-stationary time series using the GLM-Ridge learning algorithm and the MASE as error measure.

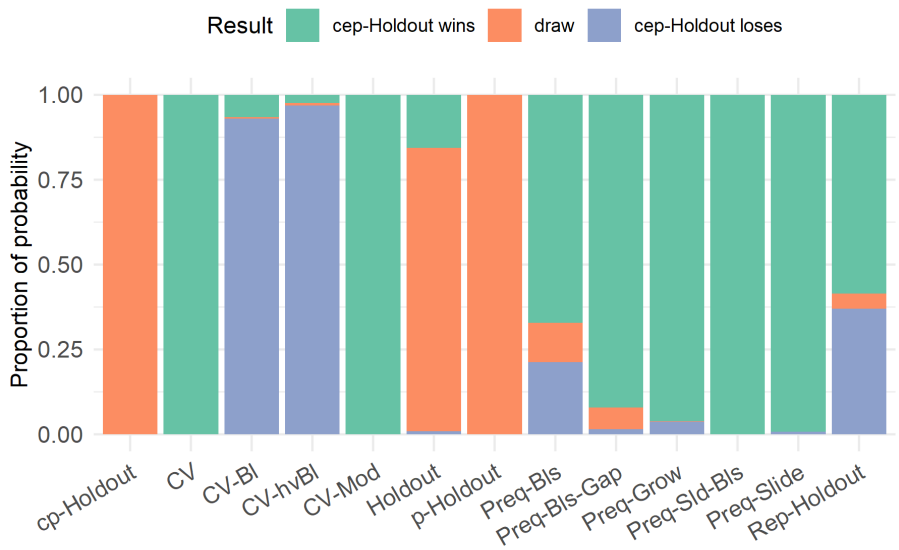


Figure B.11: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation scheme when applied to the stationary time-series from Cerqueira et al. (2020), with parameters estimated via the GLM-Ridge learning algorithm and error calculated using the MASE. The probabilities are computed using the Bayes signed-rank test.

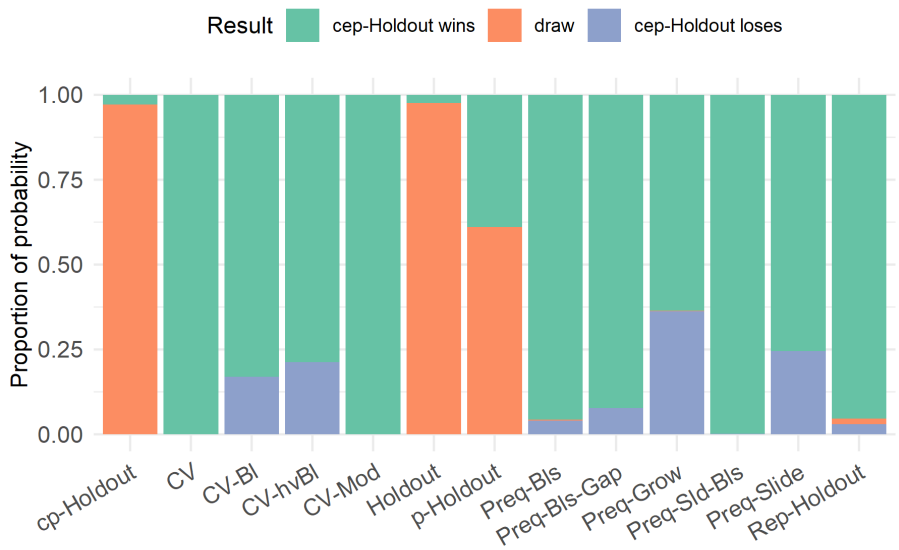


Figure B.12: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation schemes when applied to the non-stationary time-series from Cerqueira et al. (2020), with parameters estimated via the GLM-Ridge learning algorithm and error calculated using the MASE. The probabilities are computed using the Bayes signed-rank test.

B.2 Data from the M4 Competition

B.2.1 Results from the RBR learning algorithm

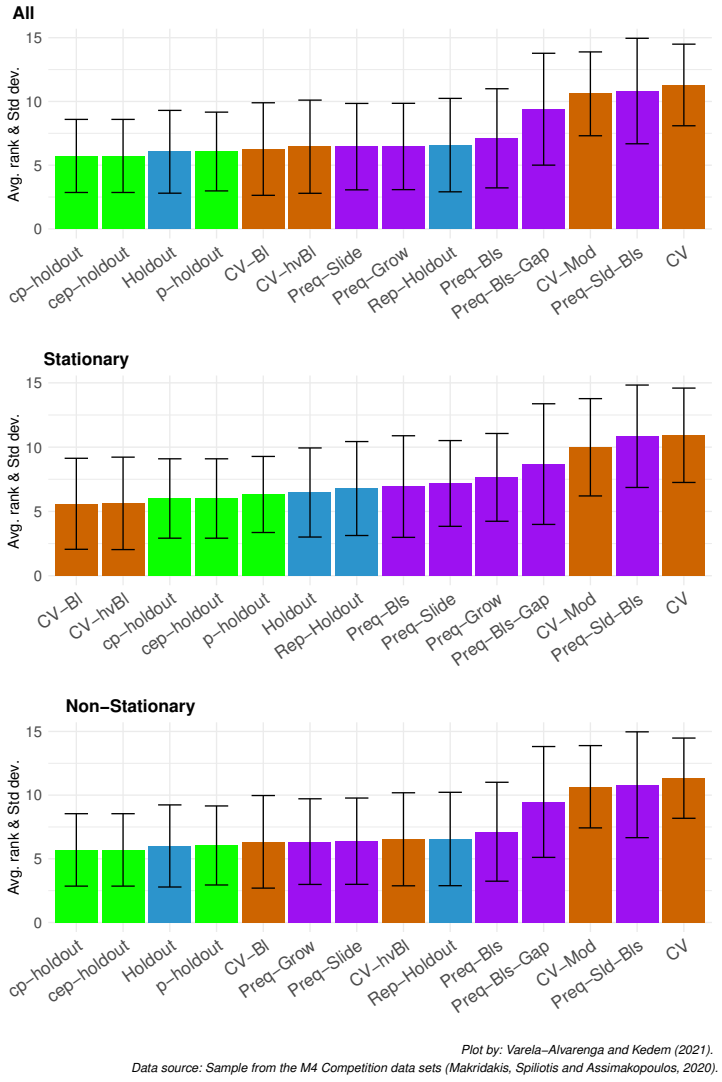


Figure B.13: Average APAE rank of each validation scheme to the sample of 1,000 time series from the M4 competition using the RBR learning algorithm and MASE as the error function. The black line represents -+ 1 standard deviation from the average.

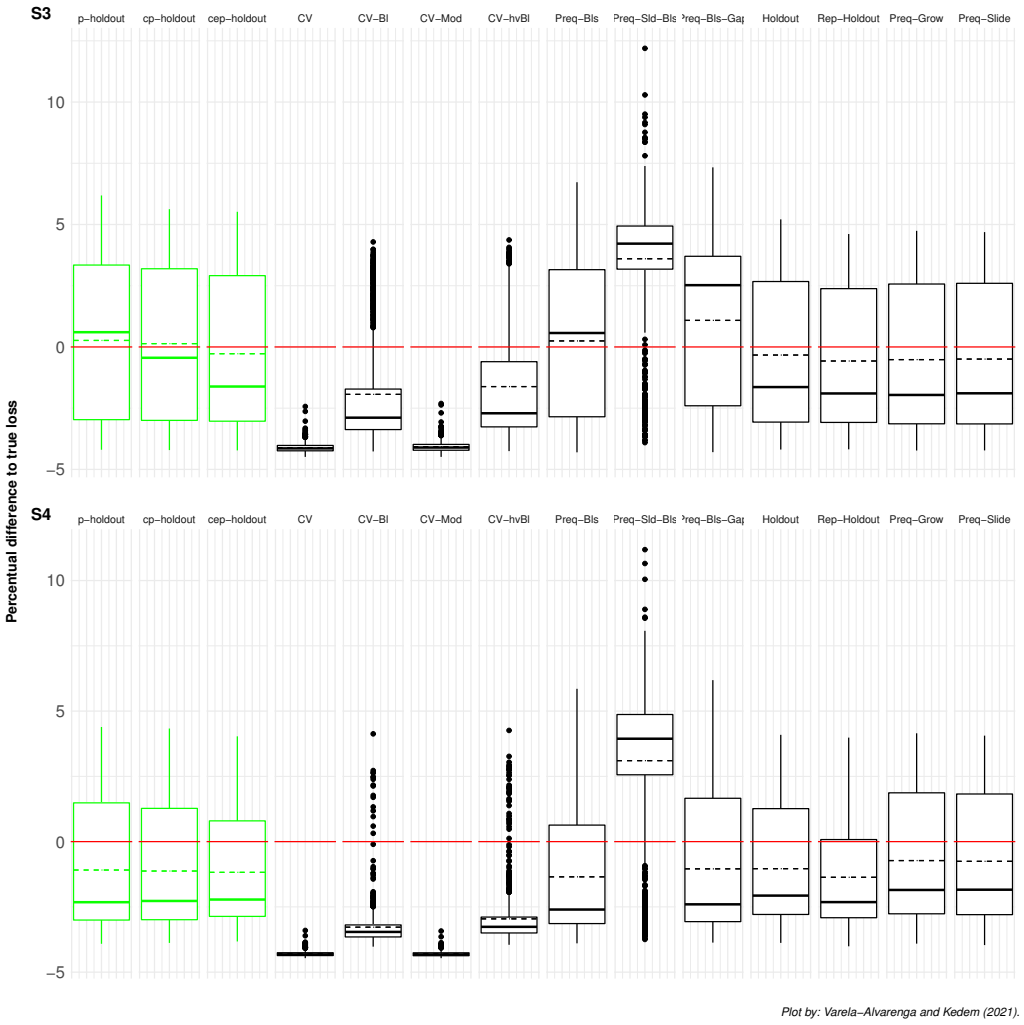


Figure B.14: Log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the sample of 1,000 time series from the M4 competition using the RBR learning algorithm. Values below the zero (red) line represent under-estimations of the error. Conversely, values above it represent over-estimations of the error.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.5795	-2.9299	0.4439	0.2002	3.3110	6.5327	0.728
cp-holdout	-4.5795	-2.9299	0.4439	0.2002	3.3110	6.5327	0.728
CV	-4.6127	-4.4619	-4.2972	-4.0383	-4.0521	5.6846	< 0.0001
CV-Bl	-4.5921	-3.4503	-2.4319	-0.6585	2.8523	8.4530	< 0.0001
CV-hvBl	-4.5828	-3.4265	-2.3631	-0.5729	2.9685	9.6575	< 0.0001
CV-Mod	-4.6125	-4.4393	-4.2651	-3.9442	-3.9949	6.7614	< 0.0001
Holdout	-4.5670	-3.1529	-0.8269	0.0633	3.2760	6.6033	0.3269
p-holdout	-4.5770	-2.9895	0.4232	0.2575	3.3777	6.5327	0.728
Preq-Bls	-4.2153	-1.9776	3.0625	Inf	4.1987	Inf	< 0.0001
Preq-Bls-Gap	-4.1389	2.3808	3.9618	Inf	4.9054	Inf	< 0.0001
Preq-Grow	-4.5664	-3.3398	-1.7586	-0.2818	3.1555	6.9907	3e-04
Preq-Sld-Bls	-4.3239	3.4905	4.4786	Inf	5.1546	Inf	< 0.0001
Preq-Slide	-4.5672	-3.3333	-1.8415	-0.2882	3.1405	6.8414	2e-04
Rep-Holdout	-4.4875	-2.6319	2.3265	0.9898	3.7825	8.5848	< 0.0001

Table B.10: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the time series from the M4 Competition sample using the RBR learning algorithm and the MASE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.3782	-3.0349	0.7064	0.1731	3.2642	6.2152	0.5594
cp-holdout	-4.3782	-3.0349	0.7064	0.1731	3.2642	6.2152	0.5594
CV	-4.6075	-4.4570	-4.2995	-4.0783	-4.0629	4.1261	< 0.0001
CV-Bl	-4.3540	-3.4315	-2.4488	-0.7868	2.7848	5.7453	< 0.0001
CV-hvBl	-4.3552	-3.4406	-2.4373	-0.7001	2.9438	6.8117	< 0.0001
CV-Mod	-4.6055	-4.4397	-4.2717	-3.9866	-4.0087	4.4632	< 0.0001
Holdout	-4.4533	-3.2102	-1.2202	0.0159	3.2687	5.4386	0.3169
p-holdout	-4.3782	-2.9805	0.6429	0.2667	3.3628	6.2152	0.5594
Preq-Bls	-4.1195	-2.0404	2.9020	Inf	4.1680	Inf	< 0.0001
Preq-Bls-Gap	-4.0964	2.3304	3.8976	Inf	4.8378	Inf	< 0.0001
Preq-Grow	-4.4095	-3.3232	-1.9077	-0.3468	3.1293	5.8924	5e-04
Preq-Sld-Bls	-4.1638	3.4625	4.4048	Inf	5.1176	Inf	< 0.0001
Preq-Slide	-4.4010	-3.2943	-2.0541	-0.3623	3.0805	5.7150	6e-04
Rep-Holdout	-4.2533	-2.6187	2.1759	0.9416	3.6959	5.9270	< 0.0001

Table B.11: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the stationary time series from the M4 Competition sample using the RBR learning algorithm and the MASE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.5795	-2.8861	-0.1334	0.2367	3.3895	6.5327	0.9227
cp-holdout	-4.5795	-2.8861	-0.1334	0.2367	3.3895	6.5327	0.9227
CV	-4.6127	-4.4660	-4.2871	-3.9841	-4.0465	5.6846	< 0.0001
CV-Bl	-4.5921	-3.4549	-2.2261	-0.4849	3.0126	8.4530	7e-04
CV-hvBl	-4.5828	-3.4195	-2.2488	-0.4007	2.9722	9.6575	0.0036
CV-Mod	-4.6125	-4.4384	-4.2571	-3.8868	-3.9864	6.7614	< 0.0001
Holdout	-4.5670	-3.0770	-0.3140	0.1274	3.2925	6.6033	0.7711
p-holdout	-4.5770	-3.0064	-0.1444	0.2450	3.4087	6.5327	0.9227
Preq-Bls	-4.2153	-1.7258	3.1894	Inf	4.2764	Inf	< 0.0001
Preq-Bls-Gap	-4.1389	2.4127	4.0944	Inf	4.9702	Inf	< 0.0001
Preq-Grow	-4.5664	-3.3644	-1.5904	-0.1939	3.2264	6.9907	0.1455
Preq-Sld-Bls	-4.3239	3.5347	4.5208	Inf	5.2419	Inf	< 0.0001
Preq-Slide	-4.5672	-3.3816	-1.4619	-0.1880	3.2122	6.8414	0.0806
Rep-Holdout	-4.4875	-2.7016	2.5831	1.0551	3.8375	8.5848	< 0.0001

Table B.12: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the non-stationary time series from the M4 Competition sample using the RBR learning algorithm and the MASE as error measure.

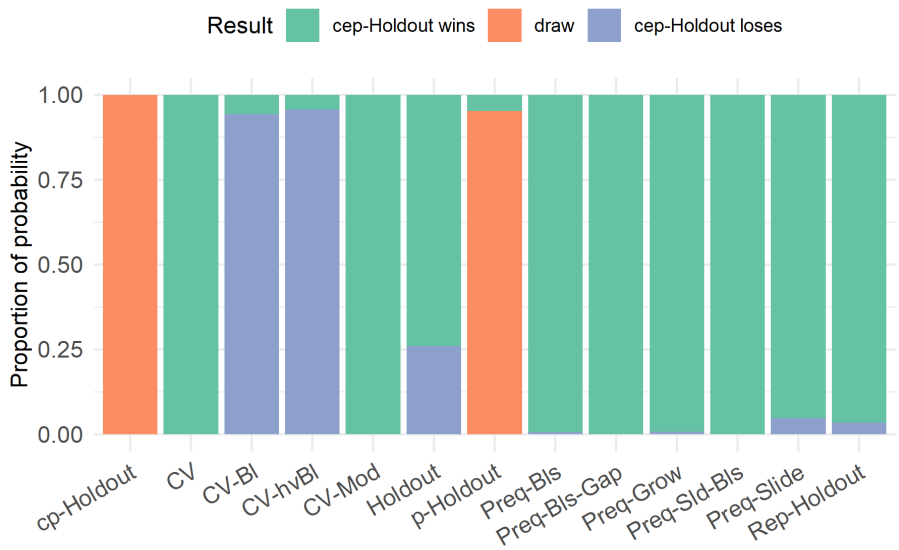


Figure B.15: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation scheme when applied to the stationary series from the sample of 1,000 time series from the M4 competition, with parameters estimated via the RBR learning algorithm and error calculated using the MASE. The probabilities are computed using the Bayes signed-rank test.

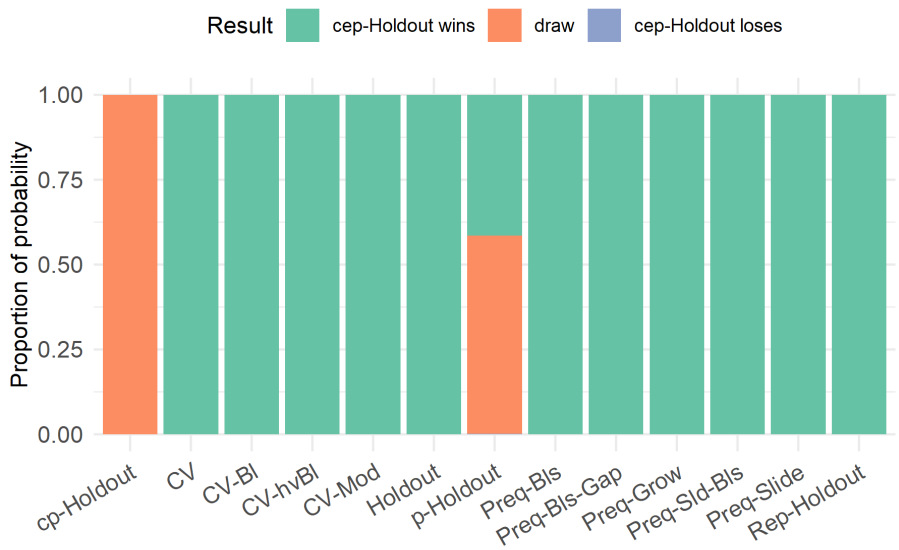


Figure B.16: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation schemes when applied to the non-stationary series from the sample of 1,000 time series from the M4 competition, with parameters estimated via the RBR learning algorithm and error calculated using the MASE. The probabilities are computed using the Bayes signed-rank test.

B.2.2 Results from the RF learning algorithm

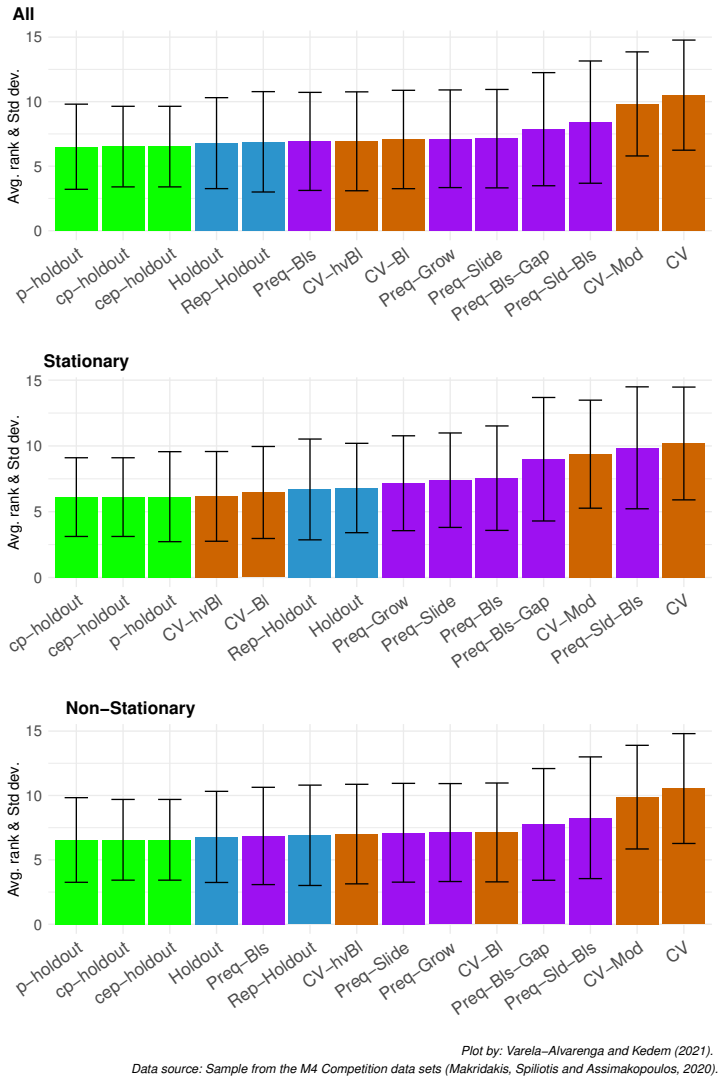


Figure B.17: Average APAE rank of each validation scheme to the sample of 1,000 time series from the M4 competition using the RF learning algorithm and MASE as the error function. The black line represents -+ 1 standard deviation from the average.

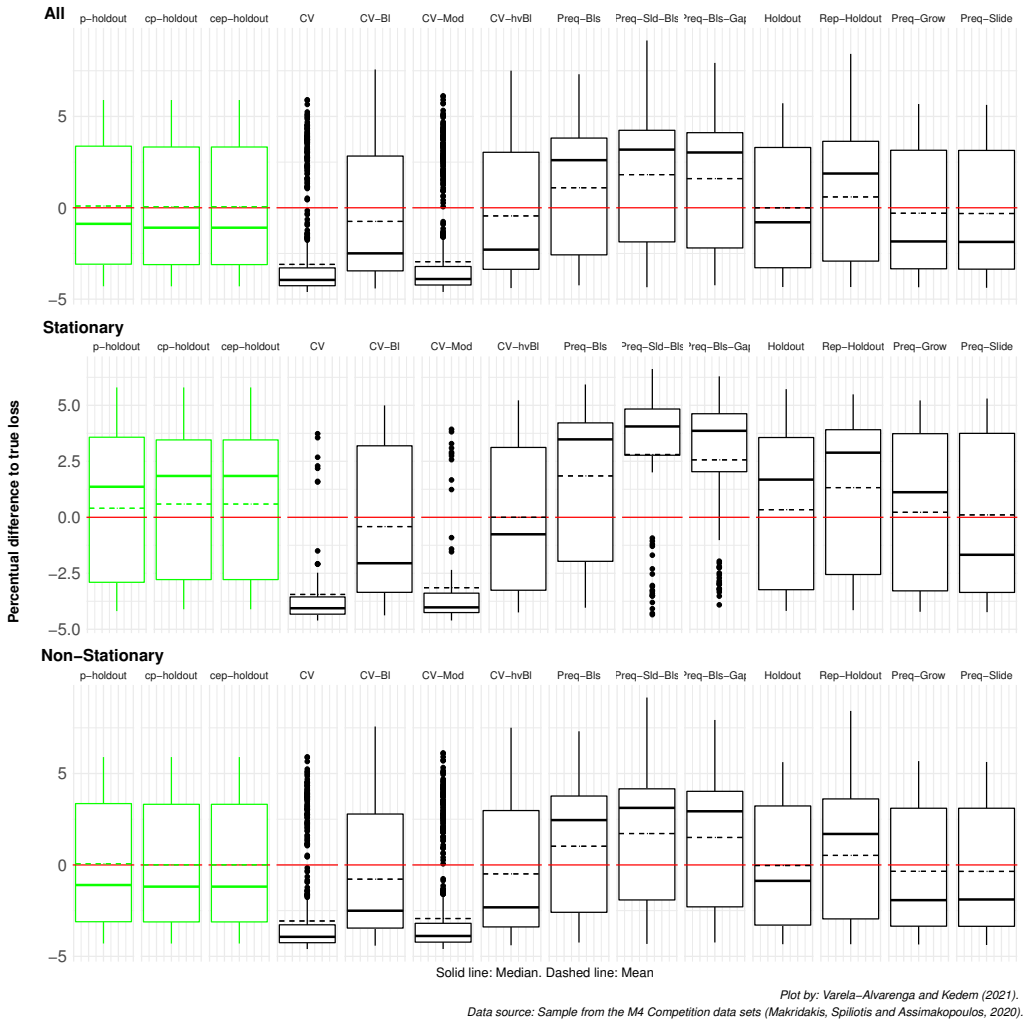


Figure B.18: Log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the sample of 1,000 time series from the M4 competition using the RF learning algorithm. Values below the zero (red) line represent under-estimations of the error. Conversely, values above it represent over-estimations of the error.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.2983	-3.1091	-1.0897	0.0606	3.3269	5.8987	0.1372
cp-holdout	-4.2983	-3.1091	-1.0897	0.0606	3.3269	5.8987	0.1372
CV	-4.6085	-4.2654	-3.9484	-3.0992	-3.2894	5.9089	< 0.0001
CV-Bl	-4.4149	-3.4521	-2.4935	-0.7465	2.8318	7.5705	< 0.0001
CV-hvBl	-4.3938	-3.3676	-2.2907	-0.4448	3.0375	7.5050	< 0.0001
CV-Mod	-4.6061	-4.2261	-3.8989	-2.9506	-3.2134	6.1251	< 0.0001
Holdout	-4.3359	-3.2810	-0.7936	0.0024	3.3012	5.7206	0.2967
p-holdout	-4.2959	-3.0869	-0.8786	0.0963	3.3718	5.8987	0.4292
Preq-Bls	-4.2431	-2.5343	2.6466	Inf	3.8444	Inf	< 0.0001
Preq-Bls-Gap	-4.2402	-2.1717	3.0450	Inf	4.1573	Inf	< 0.0001
Preq-Grow	-4.3496	-3.3386	-1.8420	-0.2990	3.1477	5.6790	1e-04
Preq-Sld-Bls	-4.3476	-1.7305	3.2492	Inf	4.2864	Inf	< 0.0001
Preq-Slide	-4.3784	-3.3586	-1.8682	-0.3127	3.1398	5.6314	6e-04
Rep-Holdout	-4.3369	-2.9182	1.8753	0.5943	3.6380	8.4230	0.0021

Table B.13: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the time series from the M4 Competition sample using the RF learning algorithm and the MASE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.2349	-3.1597	-1.5520	-0.1749	3.1855	5.8670	0.0045
cp-holdout	-4.2349	-3.1597	-1.5520	-0.1749	3.1855	5.8670	0.0045
CV	-4.5946	-4.2664	-3.9402	-3.1472	-3.3302	5.8738	< 0.0001
CV-Bl	-4.3784	-3.4575	-2.6338	-0.9461	2.5268	6.2103	< 0.0001
CV-hvBl	-4.3616	-3.4037	-2.3897	-0.5990	2.8557	6.3341	< 0.0001
CV-Mod	-4.5924	-4.2279	-3.9012	-3.0118	-3.2881	6.1251	< 0.0001
Holdout	-4.2365	-3.3622	-1.6271	-0.1974	3.2322	5.6221	0.0242
p-holdout	-4.1936	-3.2117	-1.3326	-0.0703	3.3215	5.8670	0.0952
Preq-Bls	-4.1395	-2.5679	2.4032	Inf	3.8008	Inf	< 0.0001
Preq-Bls-Gap	-4.1408	-2.2138	2.9716	Inf	4.0569	Inf	< 0.0001
Preq-Grow	-4.3496	-3.4518	-2.2136	-0.4522	3.0672	5.5024	1e-04
Preq-Sld-Bls	-4.1313	-1.9038	3.1774	Inf	4.2404	Inf	< 0.0001
Preq-Slide	-4.3784	-3.4595	-2.2320	-0.4758	3.0557	5.4900	1e-04
Rep-Holdout	-4.1757	-2.9393	1.8670	0.5242	3.6121	5.4169	0.037

Table B.14: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the stationary time series from the M4 Competition sample using the RF learning algorithm and the MASE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.2983	-3.0472	1.3112	0.3791	3.5300	5.8987	0.332
cp-holdout	-4.2983	-3.0472	1.3112	0.3791	3.5300	5.8987	0.332
CV	-4.6085	-4.2635	-3.9562	-3.0343	-3.2110	5.9089	< 0.0001
CV-Bl	-4.4149	-3.4144	-2.3632	-0.4764	3.1236	7.5705	1e-04
CV-hvBl	-4.3938	-3.2903	-2.0566	-0.2362	3.2332	7.5050	0.0116
CV-Mod	-4.6061	-4.2207	-3.8754	-2.8679	-3.0584	6.0848	< 0.0001
Holdout	-4.3359	-3.1674	1.0748	0.2728	3.3662	5.7206	0.332
p-holdout	-4.2959	-2.9685	1.1562	0.3218	3.4358	5.8987	0.4971
Preq-Bls	-4.2431	-2.4978	2.8488	Inf	3.9256	Inf	< 0.0001
Preq-Bls-Gap	-4.2402	-2.1024	3.2511	Inf	4.2180	Inf	< 0.0001
Preq-Grow	-4.3357	-3.1889	-1.1992	-0.0916	3.2590	5.6790	0.2072
Preq-Sld-Bls	-4.3476	-1.2746	3.3340	Inf	4.3000	Inf	< 0.0001
Preq-Slide	-4.3419	-3.2304	-1.1612	-0.0920	3.2743	5.6314	0.4377
Rep-Holdout	-4.3369	-2.8443	1.9072	0.6892	3.7113	8.4230	0.0255

Table B.15: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the non-stationary time series from the M4 Competition sample using the RF learning algorithm and the MASE as error measure.

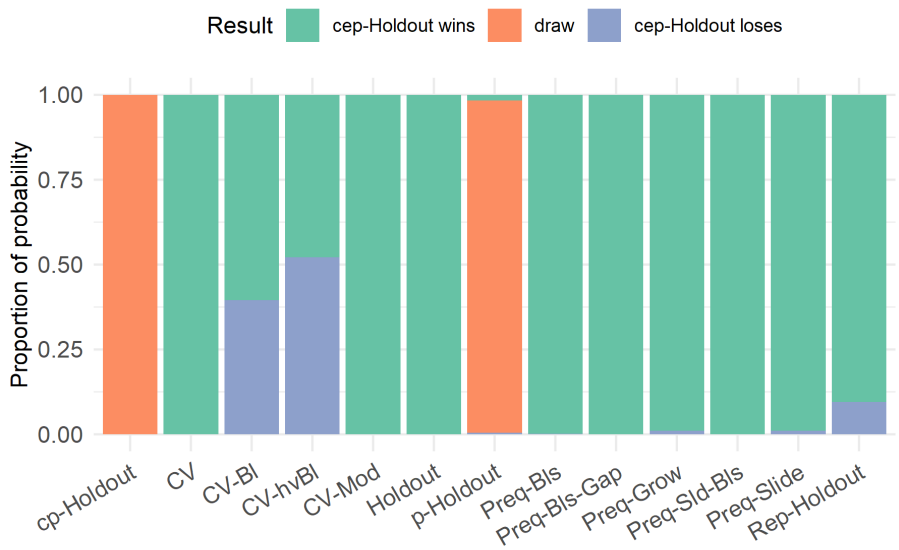


Figure B.19: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation scheme when applied to the stationary series from the sample of 1,000 time series from the M4 competition, with parameters estimated via the RF learning algorithm and error calculated using the MASE. The probabilities are computed using the Bayes signed-rank test.

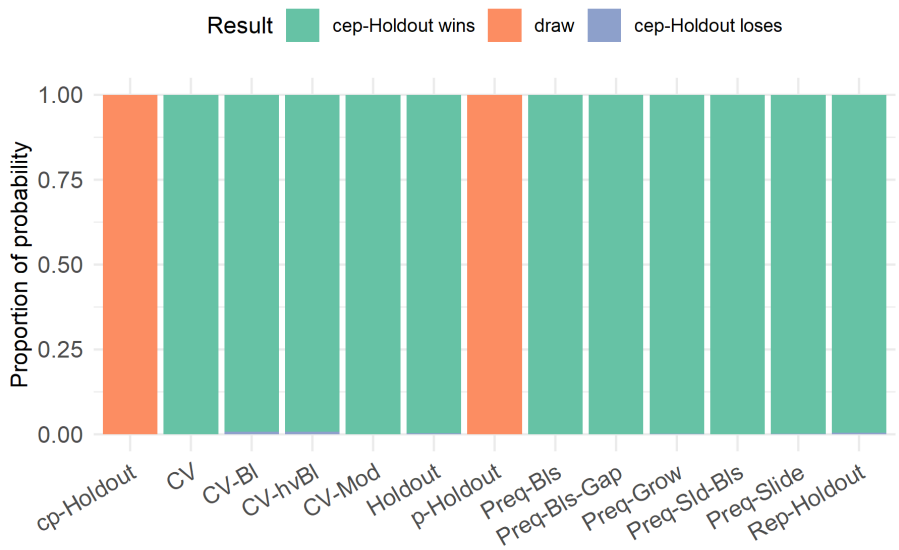


Figure B.20: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation schemes when applied to the non-stationary series from the sample of 1,000 time series from the M4 competition, with parameters estimated via the RF learning algorithm and error calculated using the MASE. The probabilities are computed using the Bayes signed-rank test.

B.2.3 Results from the GLM learning algorithm

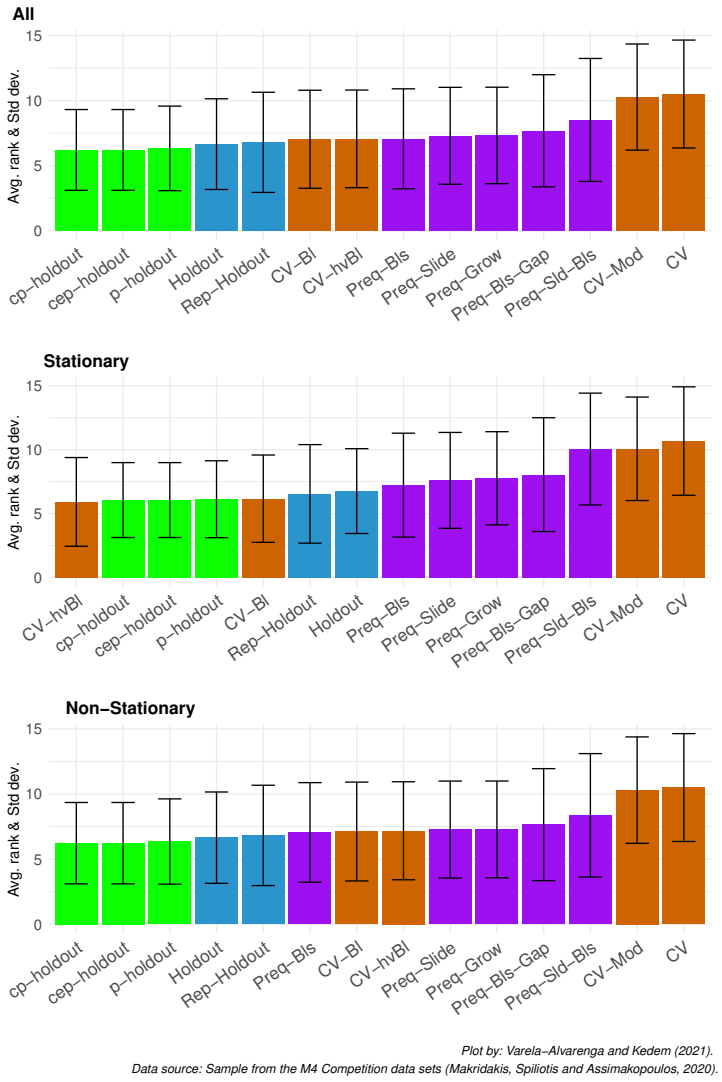


Figure B.21: Average APAE rank of each validation scheme to the sample of 1,000 time series from the M4 competition using the GLM-Ridge learning algorithm and MASE as the error function. The black line represents -+ 1 standard deviation from the average.

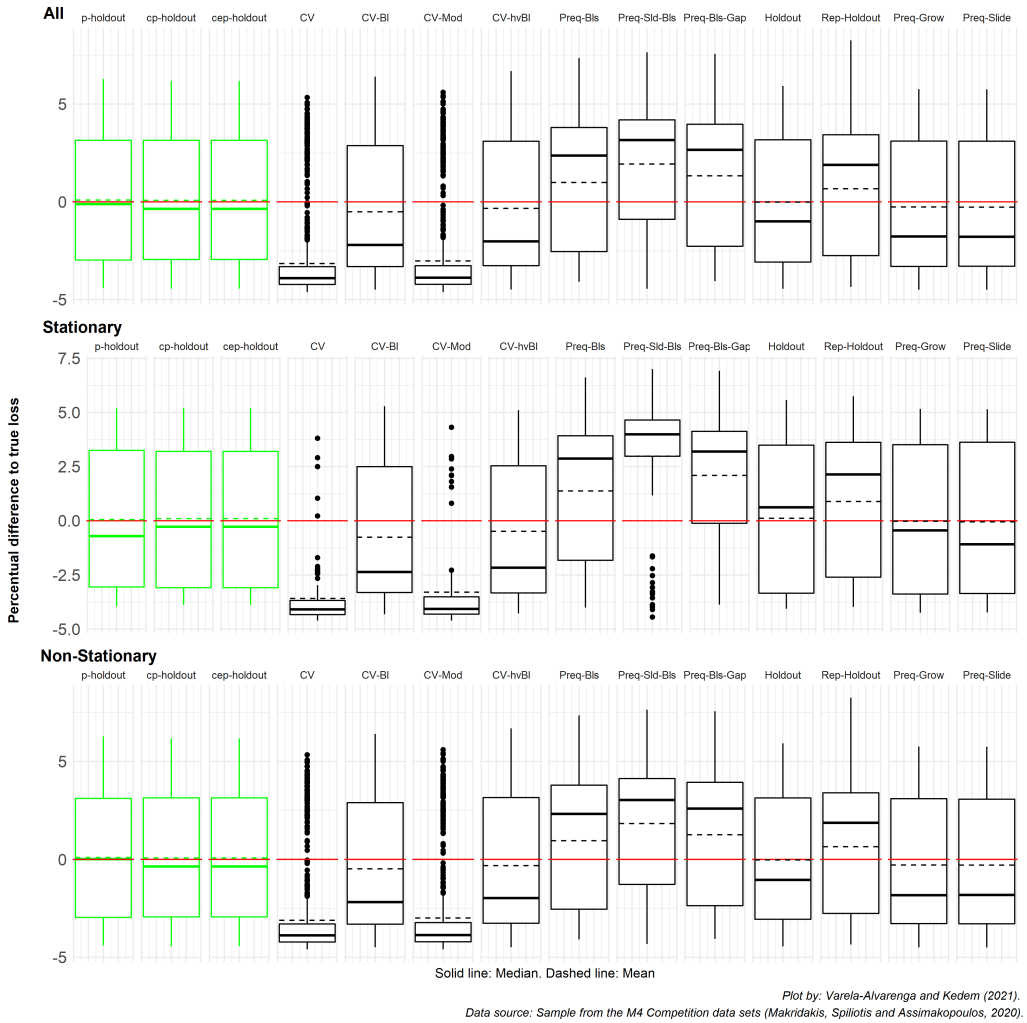


Figure B.22: Log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the sample of 1,000 time series from the M4 competition using the GLM-Ridge learning algorithm. Values below the zero (red) line represent under-estimations of the error. Conversely, values above it represent over-estimations of the error.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.4369	-2.9482	-0.3593	0.0753	3.1487	6.1833	0.6353
cp-holdout	-4.4369	-2.9482	-0.3593	0.0753	3.1487	6.1833	0.6353
CV	-4.6080	-4.2246	-3.9029	-3.1556	-3.3146	5.3390	< 0.0001
CV-Bl	-4.4894	-3.3111	-2.2039	-0.5091	2.8751	6.4085	< 0.0001
CV-hvBl	-4.4857	-3.2638	-2.0213	-0.3347	3.0985	6.6908	< 0.0001
CV-Mod	-4.6080	-4.2182	-3.8798	-3.0229	-3.2666	5.6017	< 0.0001
Holdout	-4.4450	-3.0800	-0.9980	-0.0145	3.1733	5.9277	0.1067
p-holdout	-4.3981	-2.9781	-0.1149	0.0914	3.1477	6.2853	0.9748
Preq-Bls	-4.0868	-2.5359	2.3808	Inf	3.8261	Inf	< 0.0001
Preq-Bls-Gap	-4.0577	-2.2591	2.6912	Inf	3.9889	Inf	< 0.0001
Preq-Grow	-4.4962	-3.3039	-1.7719	-0.2620	3.1069	5.7649	6e-04
Preq-Sld-Bls	-4.4441	-0.6312	3.2135	Inf	4.2300	Inf	< 0.0001
Preq-Slide	-4.4978	-3.2934	-1.7861	-0.2714	3.1019	5.7528	9e-04
Rep-Holdout	-4.3486	-2.7494	1.8906	0.6691	3.4315	8.2590	< 0.0001

Table B.16: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the time series from the M4 Competition sample using the GLM-RIDGE learning algorithm and the MASE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.2143	-3.0228	-0.7194	0.0035	3.1446	5.4322	0.5594
cp-holdout	-4.2143	-3.0228	-0.7194	0.0035	3.1446	5.4322	0.5594
CV	-4.5905	-4.2298	-3.9028	-3.2043	-3.3862	4.9959	< 0.0001
CV-Bl	-4.3598	-3.3194	-2.3067	-0.6650	2.7220	5.3448	< 0.0001
CV-hvBl	-4.3563	-3.2559	-2.0800	-0.4417	2.9847	5.4041	1e-04
CV-Mod	-4.5904	-4.2194	-3.8786	-3.0766	-3.3251	5.1343	< 0.0001
Holdout	-4.3058	-3.1390	-1.2557	-0.0848	3.1758	5.6044	0.1562
p-holdout	-4.2105	-3.0428	0.3155	0.0917	3.1055	5.4322	0.8024
Preq-Bls	-4.0350	-2.4423	2.3805	Inf	3.6871	Inf	< 0.0001
Preq-Bls-Gap	-4.0577	-2.1914	2.6608	Inf	3.8643	Inf	< 0.0001
Preq-Grow	-4.3190	-3.3190	-1.9690	-0.3262	3.0765	5.6249	0.0035
Preq-Sld-Bls	-4.1751	-0.9669	3.0659	Inf	4.1745	Inf	< 0.0001
Preq-Slide	-4.3060	-3.3008	-1.9052	-0.3520	3.0457	5.5836	0.0011
Rep-Holdout	-4.2346	-2.7547	1.8915	0.6762	3.3913	5.7761	1e-04

Table B.17: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the stationary time series from the M4 Competition sample using the GLM-RIDGE learning algorithm and the MASE as error measure.

Scheme	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	p-value
cep-holdout	-4.4369	-2.8636	-0.0162	0.1723	3.1615	6.1833	1
cp-holdout	-4.4369	-2.8636	-0.0162	0.1723	3.1615	6.1833	1
CV	-4.6080	-4.2204	-3.9068	-3.0896	-3.2042	5.3390	< 0.0001
CV-Bl	-4.4894	-3.2961	-2.0105	-0.2982	3.0637	6.4085	0.0327
CV-hvBl	-4.4857	-3.2693	-1.9145	-0.1899	3.1964	6.6908	0.0255
CV-Mod	-4.6080	-4.2167	-3.8918	-2.9503	-3.1429	5.6017	< 0.0001
Holdout	-4.4450	-3.0093	-0.7491	0.0805	3.1692	5.9277	0.4377
p-holdout	-4.3981	-2.9435	-0.5265	0.0909	3.1696	6.2853	0.698
Preq-Bls	-4.0868	-2.5515	2.4396	Inf	3.9269	Inf	< 0.0001
Preq-Bls-Gap	-4.0222	-2.4000	2.7617	Inf	4.1009	Inf	< 0.0001
Preq-Grow	-4.4962	-3.2706	-1.4496	-0.1751	3.1947	5.7649	0.0652
Preq-Sld-Bls	-4.4441	0.3797	3.3660	Inf	4.3153	Inf	< 0.0001
Preq-Slide	-4.4978	-3.2783	-1.4777	-0.1624	3.1607	5.7528	0.2072
Rep-Holdout	-4.3486	-2.7329	1.8832	0.6595	3.5244	8.2590	0.0152

Table B.18: P-value for the Sign Test and summary of the log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the non-stationary time series from the M4 Competition sample using the GLM-RIDGE learning algorithm and the MASE as error measure.

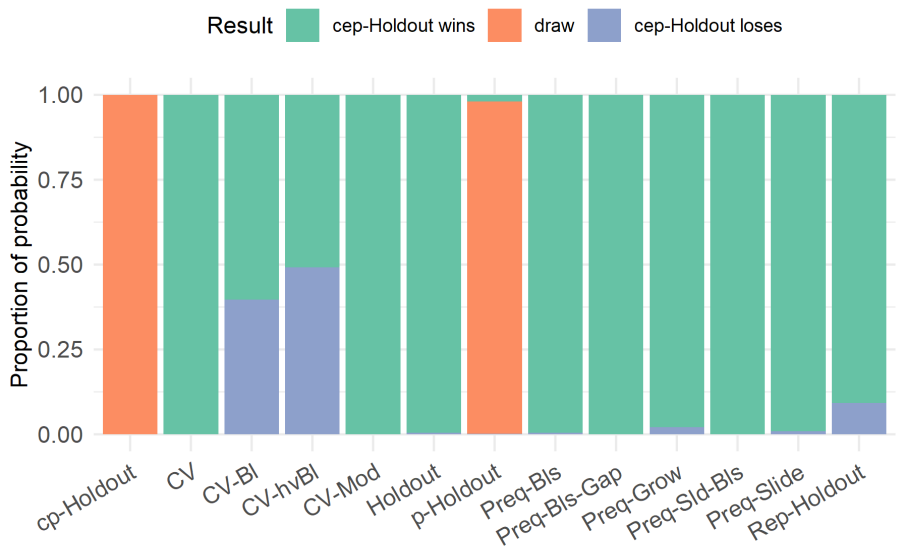


Figure B.23: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation scheme when applied to the stationary series from the sample of 1,000 time series from the M4 competition, with parameters estimated via the GLM-Ridge learning algorithm and error calculated using the MASE. The probabilities are computed using the Bayes signed-rank test.

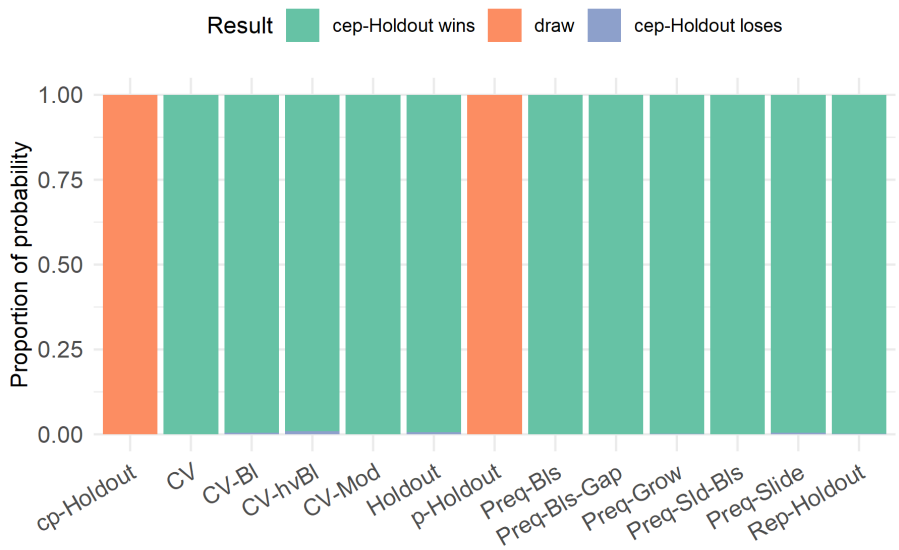


Figure B.24: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation schemes when applied to the non-stationary series from the sample of 1,000 time series from the M4 competition, with parameters estimated via the GLM-Ridge learning algorithm and error calculated using the MASE. The probabilities are computed using the Bayes signed-rank test.

B.3 Monte Carlo Simulation

B.3.1 Results from the RBR learning algorithm

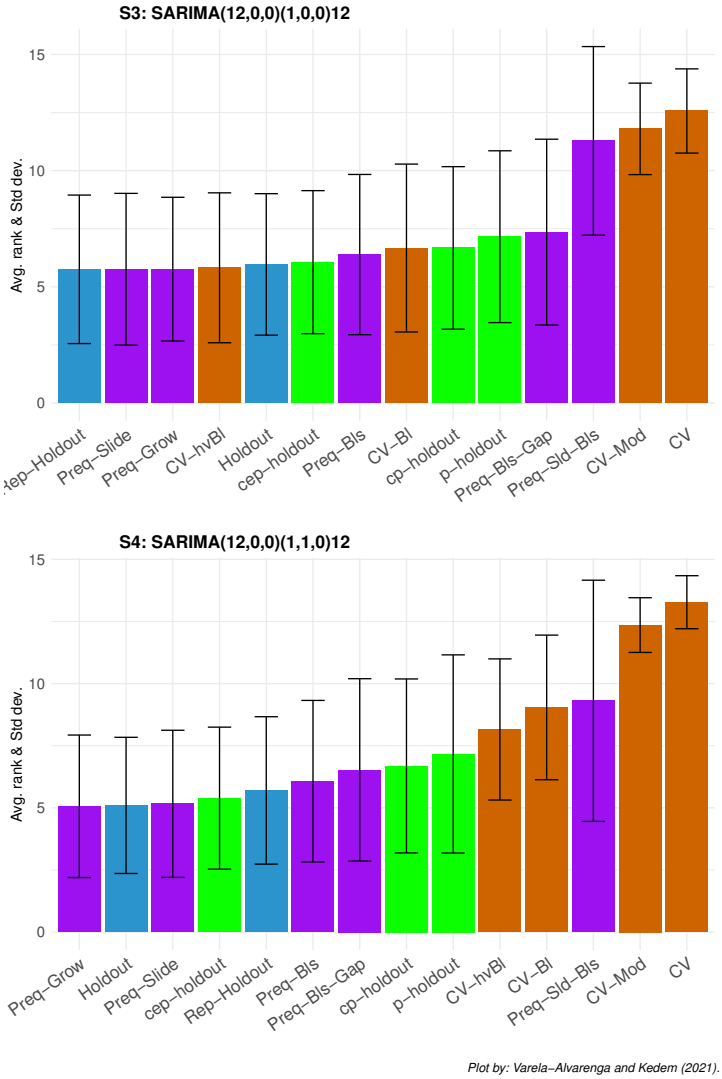


Figure B.25: Average APAE rank of each validation scheme to the sample of 1,000 simulated time series using the RBR learning algorithm and MASE as the error function. The black line represents -+ 1 standard deviation from the average.

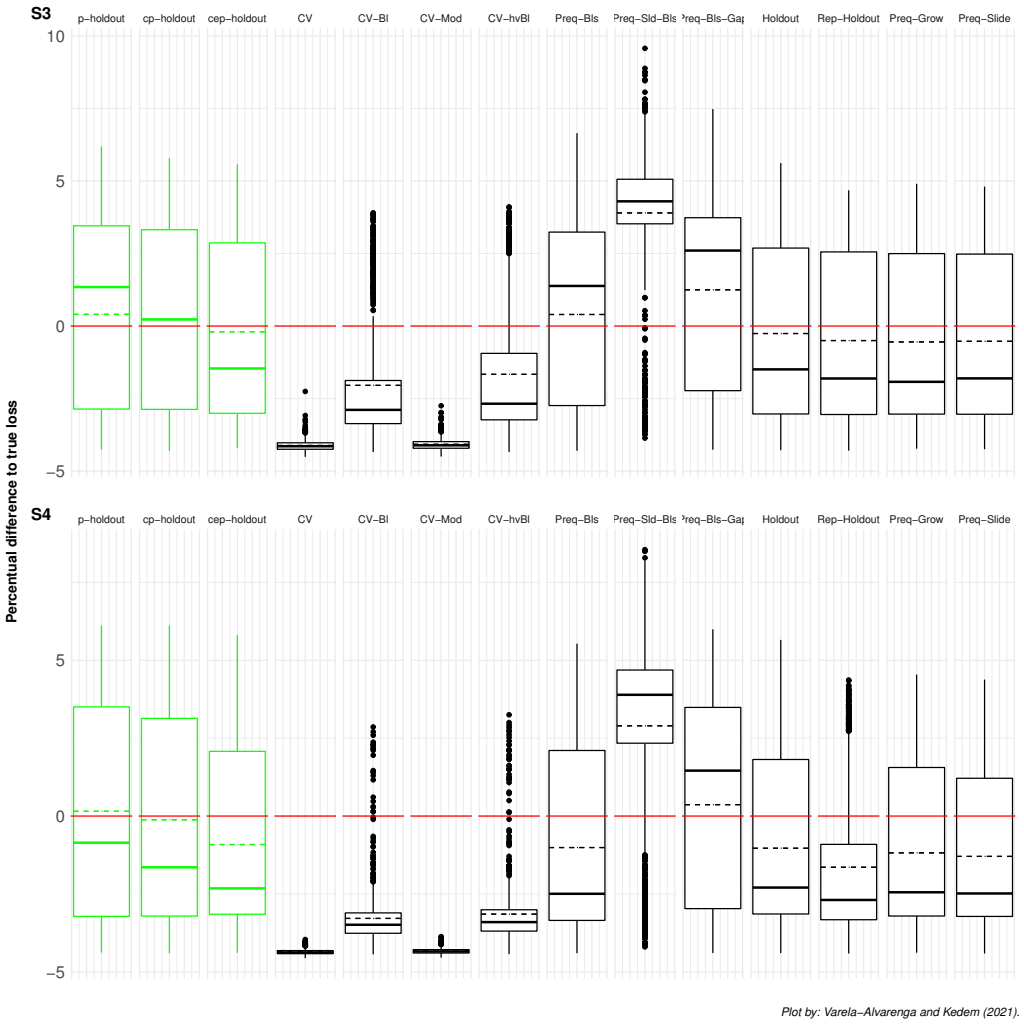


Figure B.26: Log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the sample of 1,000 simulated time series using the RBR learning algorithm. Values below the zero (red) line represent under-estimations of the error. Conversely, values above it represent over-estimations of the error.

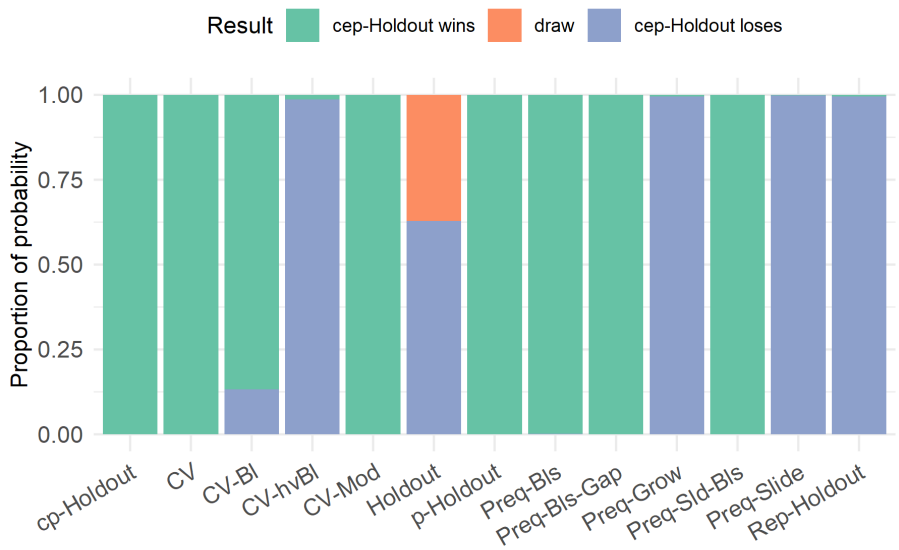


Figure B.27: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation schemes when applied to the time-series from the $S3$ simulated data set, with parameters estimated via the RBR learning algorithm and error calculated using the MASE. The probabilities are computed using the Bayes signed-rank test.

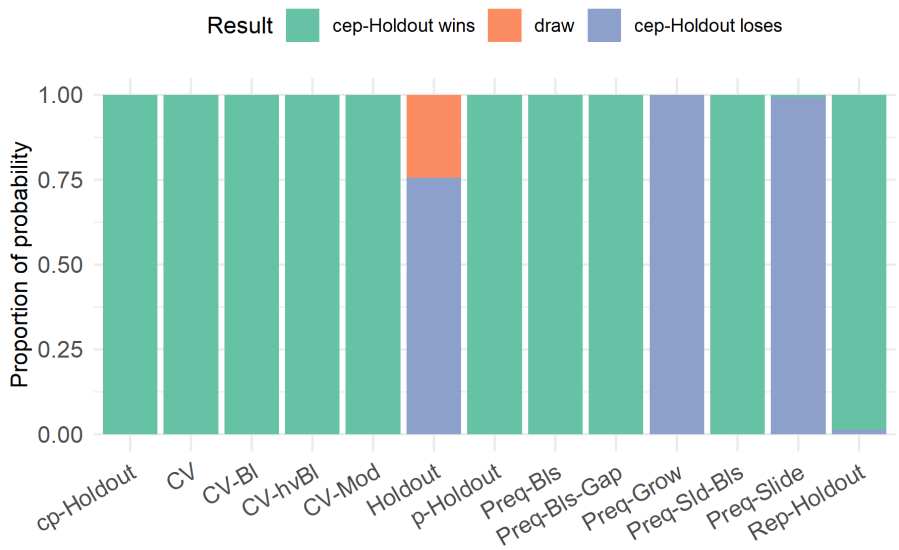


Figure B.28: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation schemes when applied to the time-series from the $S4$ simulated data set, with parameters estimated via the RBR learning algorithm and error calculated using the MASE. The probabilities are computed using the Bayes signed-rank test.

B.3.2 Results from the RF learning algorithm

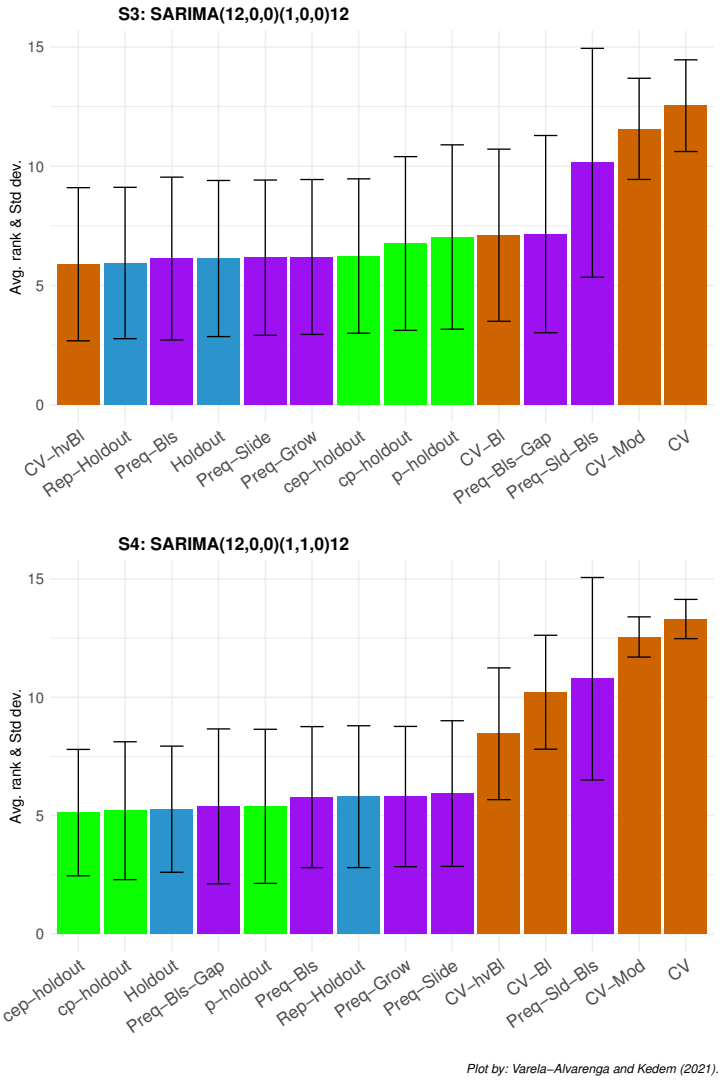


Figure B.29: Average APAE rank of each validation scheme to the sample of 1,000 simulated time series using the RF learning algorithm and MASE as the error function. The black line represents -+ 1 standard deviation from the average.

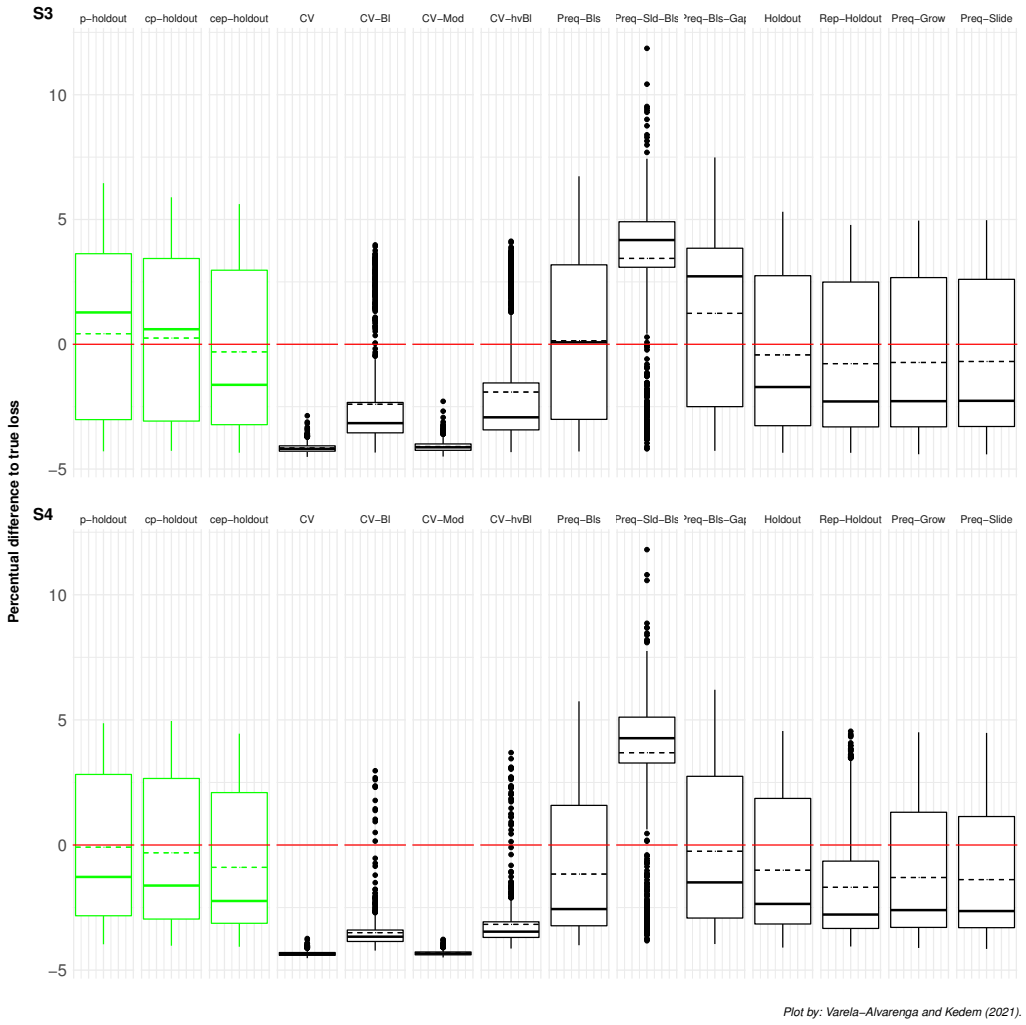


Figure B.30: Log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the sample of 1,000 simulated time series using the RF learning algorithm. Values below the zero (red) line represent under-estimations of the error. Conversely, values above it represent over-estimations of the error.

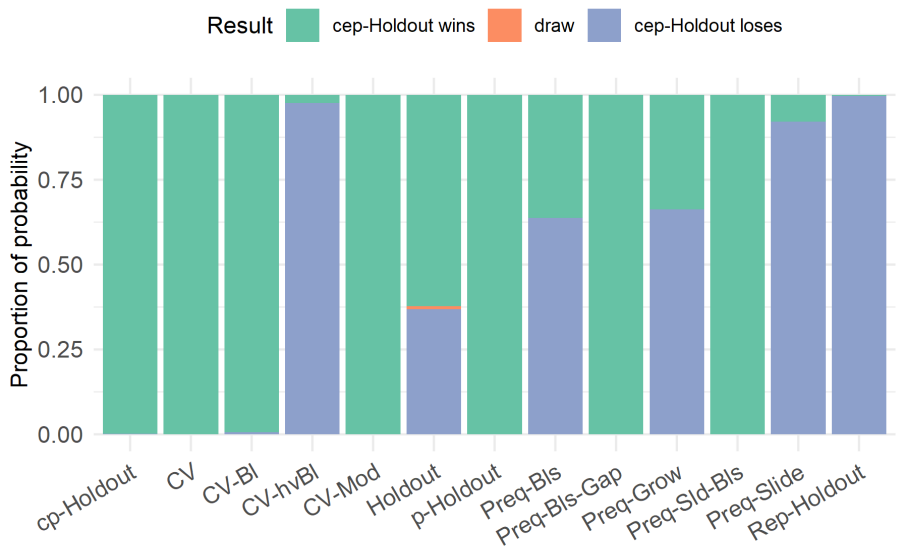


Figure B.31: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation schemes when applied to the time-series from the $S3$ simulated data set, with parameters estimated via the RF learning algorithm and error calculated using the MASE. The probabilities are computed using the Bayes signed-rank test.

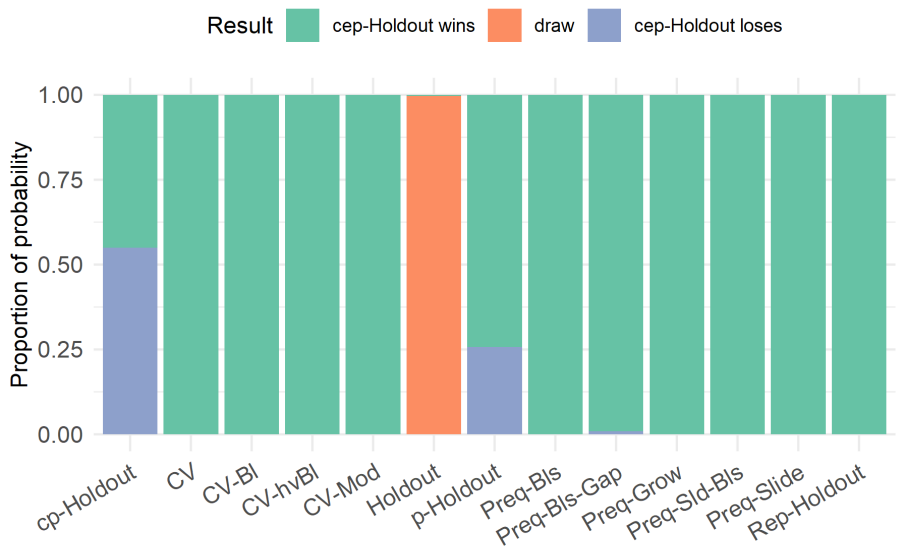


Figure B.32: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation schemes when applied to the time-series from the $S4$ simulated data set, with parameters estimated via the RF learning algorithm and error calculated using the MASE. The probabilities are computed using the Bayes signed-rank test.

B.3.3 Results from the GLM learning algorithm

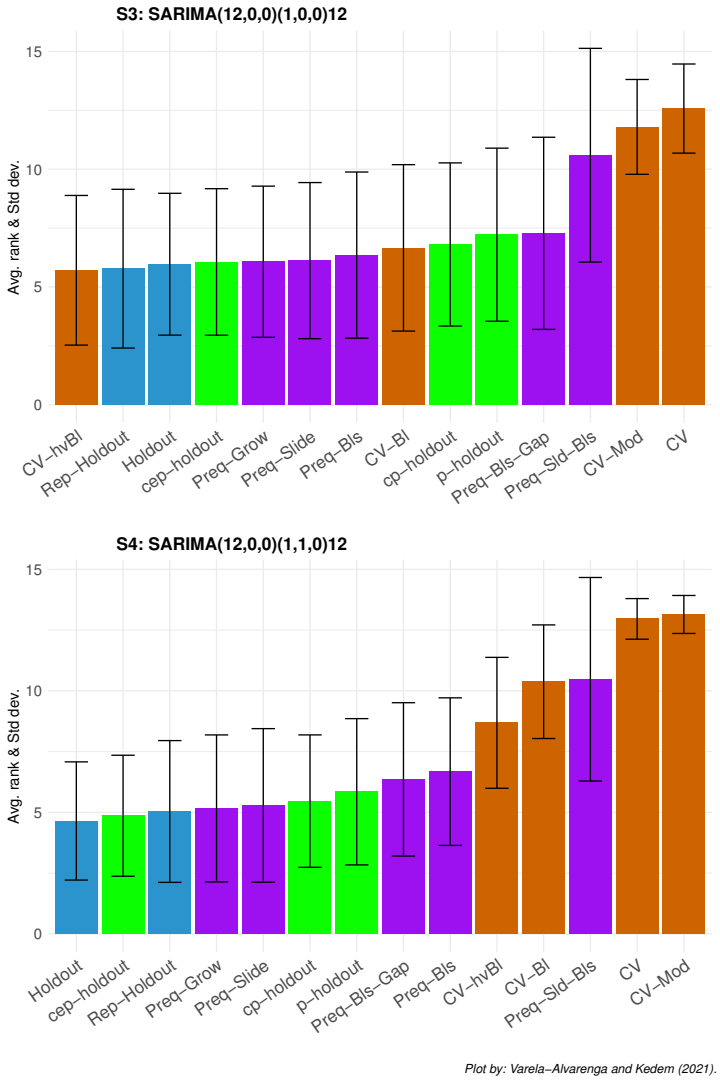


Figure B.33: Average APAE rank of each validation scheme to the sample of 1,000 simulated time series using the RIDGE learning algorithm and MASE as the error function. The black line represents -+ 1 standard deviation from the average.

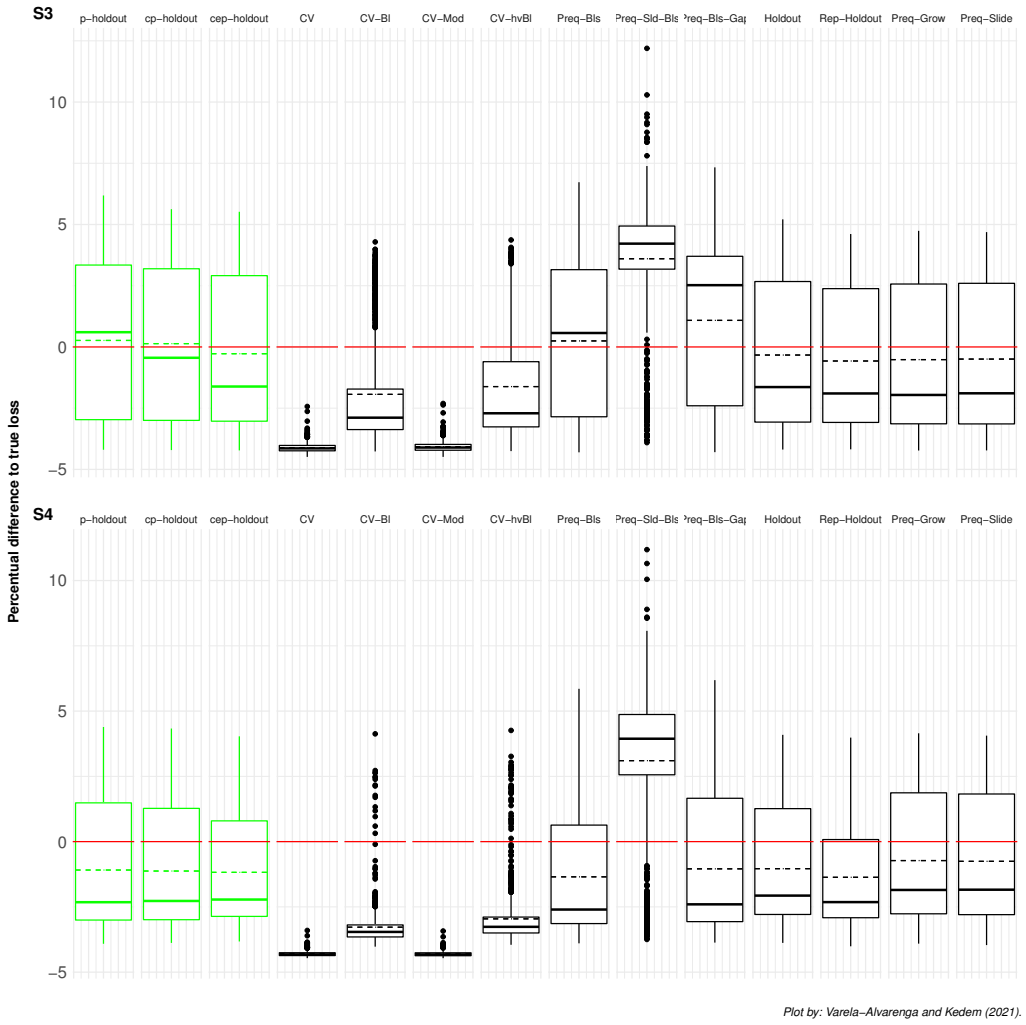


Figure B.34: Log percentage difference of the estimated loss relative to the true loss for each validation scheme applied to the sample of 1,000 simulated time series using the RIDGE learning algorithm. Values below the zero (red) line represent under-estimations of the error. Conversely, values above it represent over-estimations of the error.

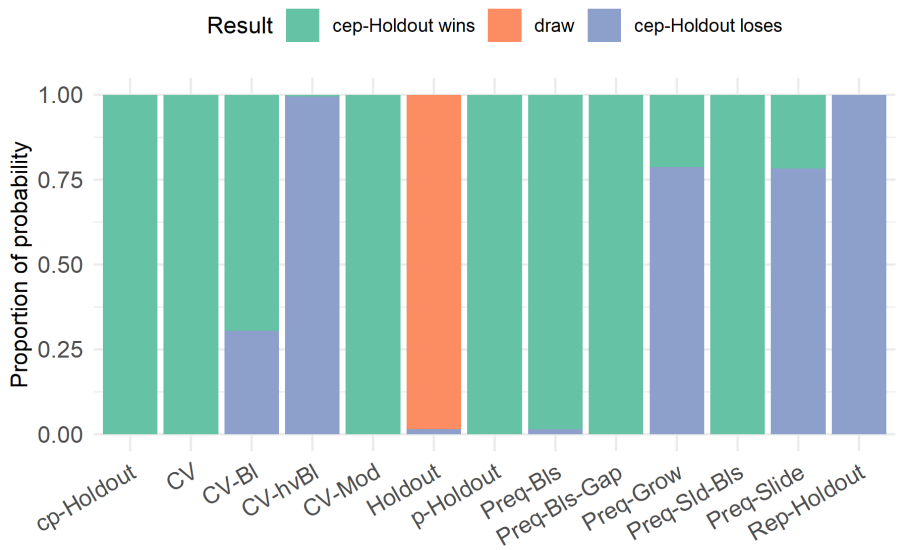


Figure B.35: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation schemes when applied to the time-series from the $S3$ simulated data set, with parameters estimated via the RIDGE learning algorithm and error calculated using the MASE. The probabilities are computed using the Bayes signed-rank test.

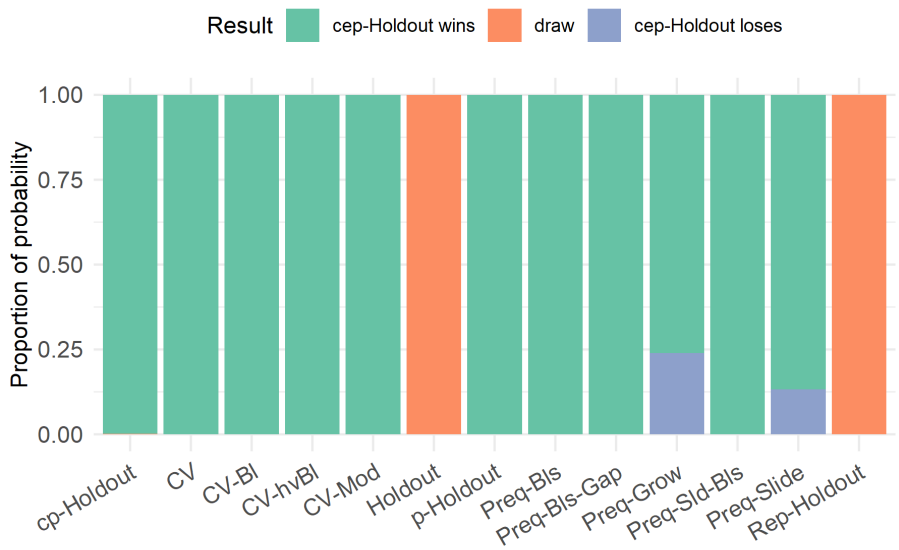


Figure B.36: Proportion of probability of winning outcome when comparing the performance estimation ability of the `cep-Holdout` and the respective validation schemes when applied to the time-series from the $S4$ simulated data set, with parameters estimated via the RIDGE learning algorithm and error calculated using the MASE. The probabilities are computed using the Bayes signed-rank test.

Appendix C: R Code

C.1 Motivation

code/motivation_p_holdout_V2021082001.R

```r
#' Motivation for the new p-Holdout family of validation schemes
#'
#' Creates the plots that show the motivation behind the p-holdout, the
#' cp-holdout, and the cep-holdout validation schemes.
#'
#' @author Gustavo Varela-Alvarenga
#'
#' @param og_TS an object with the original time series
#' @param q_e numeric value between 0 and 1. It is the percentage of
   the
#'     training data that will be used as the estimation set (i.e.,
#'     the inner split).
#' @param q numeric value between 0 and 1. It is the percentage of the
#'     original time series that will be used as the training data
#'     (i.e., the outer split). If \code{NULL}, then \code{forecast.h}
#'     must be specified.
#' @param forecast.h integer. It is the length of the forecast horizon.
#'     To be used when the test data has a specific length, instead of
#'     being defined by \code{ceiling(length(og_TS)*q)}.
#'     If \code{NULL}, then \code{q} must be specified.
#'
#' @return a panel with five plots. The first one shows the time series
#'    split into training data and test data using the Holdout scheme.
#'    The second displays the test data, and the training data split
#'    into estimation set and validation set via the original holdout
#'    validation scheme.
#'    The third, fourth, and fifth plots use the procedures in the
#'    p-Holdout family. Respectively, they display the training set
#'    partitioned into the estimation and validation sets using the
#'    p-Holdout, cp-Holdout, and cep-Holdout schemes.
#'
plot_motivation_p_holdout <- function(og_TS,
                                      q_e,
                                      q = NULL,
                                      forecast.h = NULL){

  # ============================================================== #
  # Length of the Training Data and the Test Data ----------------
  if (is.null(forecast.h)){
```

```
    if (is.null(q)) {
      error("You must provide either 'q' or 'forecast.h'.")
    }
    length.test.data  <- ceiling(length(og_TS)*(1-q))
    length.train.data <- floor(length(og_TS)*q)
  } else {
    length.test.data  <- forecast.h
    length.train.data <- length(og_TS) - length.test.data
  }

  # Create the Training Data and Test Data set -----------------------
  trainingData <- head(og_TS, length.train.data)
  testData     <- tail(og_TS, length.test.data)

  # ================================================================= #
  # Priodicity/frequency ---------------------------------------------
  ## number of samples (observations) per unit of time
  tmpFrequency <- stats::frequency(og_TS)

  tmpPeriod   <- forecast::findfrequency(og_TS)

  # ================================================================= #
  # Holdout Scheme: Lengths of the sets --------
  length.validation.set.HO  <- ceiling(length(trainingData)*(1-q_e))
  length.estimation.set.HO  <- floor(length(trainingData)*q_e)

  # Holdout Scheme: Create the Estimation and Validation Sets ---------
  estimationSetHO <- head(trainingData, length.estimation.set.HO)
  validationSetHO <- tail(trainingData, length.validation.set.HO)

  # ================================================================= #
  # p-Holdout: Length of the Estimation Set and the Validation Set ----
  if (tmpFrequency > 1) {
    length.validation.set.PHO <- length.test.data + tmpFrequency
  } else {
    length.validation.set.PHO <- length.validation.set.HO
  }

  length.estimation.set.PHO <-
    length.train.data - length.validation.set.PHO

  # p-Holdout: Create the Estimation Set and the Validation Set -------
  estimationSetSPHO <- head(trainingData, length.estimation.set.PHO)
  validationSetSPHO <- tail(trainingData, length.validation.set.PHO)

  # ================================================================= #
  # cp-Holdout: Length of the Estimation and Validation Sets ----

  if (tmpFrequency == 1){
    length.validation.set.CPHO <- length.validation.set.HO
  } else if (isFALSE((length.test.data/tmpFrequency)%%1 == 0)){
    length.validation.set.CPHO <-
      tmpFrequency*ceiling(length.test.data/tmpFrequency)
  } else {
```

```
  length.validation.set.CPHO <- length.test.data + tmpFrequency
}

length.estimation.set.CPHO <-
  length.train.data - length.validation.set.CPHO

# cp-Holdout: Create the Estimation Set and the Validation Set ------
estimationSetCPHO <- head(trainingData, length.estimation.set.CPHO)
validationSetCPHO <- tail(trainingData, length.validation.set.CPHO)

# ================================================================= #
# cep-Holdout: Priodicity/frequency ---------------------------------
## Create the test set
length.test.og <- ceiling(length(og_TS)*(1-q))
test.set.og    <- tail(og_TS, length.test.og)

period.test    <- forecast::findfrequency(validationSetHO)

## Get Length
tmpLength <- ifelse(length.validation.set.HO/length.test.data < 0.5,
                    length.validation.set.HO, length.test.data)

## Get Frequency
tmpFrequency <- ifelse(tmpPeriod < tmpLength, tmpPeriod, period.test)

# cep-Holdout: Length of the Estimation and Validation Sets ---------

if (tmpFrequency == 1){
  length.validation.set.CEPHO <- length.validation.set.HO
} else if (isFALSE((tmpLength/tmpFrequency)%%1 == 0)){
  length.validation.set.CEPHO <-
    tmpFrequency*ceiling(tmpLength/tmpFrequency)
} else {
  length.validation.set.CEPHO <- tmpLength + tmpFrequency
}

length.estimation.set.CEPHO <-
  length.train.data - length.validation.set.CEPHO

# cep-Holdout: Create the Estimation Set and the Validation Set -----
estimationSetCEPHO <- head(trainingData, length.estimation.set.CEPHO)
validationSetCEPHO <- tail(trainingData, length.validation.set.CEPHO)

# ================================================================= #
# Plots ----
original_pars <- par('mfrow', 'mar', 'oma')

par(mfrow = c(5, 1), mar = c(0, 0, 0, 1), oma = c(2, 2, 2, 2))

on.exit(par(original_pars)) # return par to original values

## TRAINING DATA AND TEST DATA ---------------------------------------
plot(og_TS, col = 'gray', t = "l", lty = 2, xaxt = 'n', yaxt = 'n')
mtext(text = "A: Training and Test sets", side = 2)
```

```
  lines(trainingData, col = 'blue')
  lines(testData,     col = 'red')

  ## Holdout: ESTIMATION SET, VALIDATION SET AND TEST DATA ------------
  plot(og_TS, col = 'gray', t = "p", xaxt = 'n', yaxt = 'n')
  lines(og_TS, col = 'gray', t = "l", lty = 2, xaxt = 'n', yaxt = 'n')
  mtext(text = "B:  Holdout", side = 2)
  lines(estimationSetHO,  col = 'darkgreen')
  lines(validationSetHO,  col = 'orange')
  lines(testData,         col = 'red')

  ## p-Holdout: ESTIMATION SET, VALIDATION SET AND TEST DATA ----------
  plot(og_TS, col = 'gray', t = "p", xaxt = 'n', yaxt = 'n')
  lines(og_TS, col = 'gray', t = "l", lty = 2, xaxt = 'n', yaxt = 'n')
  mtext(text = "C:   p-Holdout", side = 2)
  lines(estimationSetSPHO, col = 'darkgreen')
  lines(validationSetSPHO, col = 'orange')
  lines(testData,         col = 'red')

  # ## cp-Holdout: ESTIMATION SET, VALIDATION SET AND TEST DATA -------
  plot(og_TS, col = 'gray', t = "p", xaxt = 'n', yaxt = 'n')
  lines(og_TS, col = 'gray', t = "l", lty = 2, xaxt = 'n', yaxt = 'n')
  mtext(text = "D:  cp-Holdout", side = 2)
  lines(estimationSetCPHO, col = 'darkgreen')
  lines(validationSetCPHO, col = 'orange')
  lines(testData,          col = 'red')

  # ## cfp-Holdout: ESTIMATION SET, VALIDATION SET AND TEST DATA ------
  plot(og_TS, col = 'gray', t = "p", xaxt = 'n', yaxt = 'n')
  lines(og_TS, col = 'gray', t = "l", lty = 2, xaxt = 'n', yaxt = 'n')
  mtext(text = "E:  cfp-Holdout", side = 2)
  lines(estimationSetCEPHO, col = 'darkgreen')
  lines(validationSetCEPHO, col = 'orange')
  lines(testData,          col = 'red')
}

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
## >  Path to save plots to ----
tmpPath <- "results/plots/"
tmpVersion  <- "v2021082001"

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
## >  Final Plots ----

## USAccDeaths - Monthly data

## > Export as png ----
png(
  filename = paste0(
    tmpPath, "motivation_USAccDeaths_plot_", tmpVersion, ".png"
  ),
  type     = "cairo",
  width    = 8,
  height   = 12,
```

```
  units     = "in",
  res       = 300
)
plot_motivation_p_holdout(og_TS = USAccDeaths, q = 0.85, q_e = 0.8)
dev.off()

# ## > Export as eps ----
setEPS()
postscript(
  file = paste0(
    tmpPath, "motivation_USAccDeaths_plot_", tmpVersion, ".eps"
  ),
  width      = 8,
  height     = 12,
  horizontal = F
)
plot_motivation_p_holdout(og_TS = USAccDeaths, q = 0.85, q_e = 0.8)
dev.off()

## --- ##

## taylor (forecast library) - Half-Hourly data

## > Export as png ----
png(
  filename = paste0(
    tmpPath, "motivation_taylor_half_hour_plot_", tmpVersion, ".png"
  ),
  type     = "cairo",
  width    = 8,
  height   = 12,
  units    = "in",
  res      = 300
)
plot_motivation_p_holdout(og_TS = forecast::taylor, q = 0.8, q_e = 0.8)
dev.off()

# ## > Export as eps ----
setEPS()
postscript(
  file       = paste0(
    tmpPath, "motivation_taylor_half_hour_plot_", tmpVersion, ".eps"
  ),
  width      = 8,
  height     = 12,
  horizontal = F
)
plot_motivation_p_holdout(og_TS = forecast::taylor, q = 0.8, q_e = 0.8)
dev.off()

## lynx - Annual data

## > Export as png ----
png(
```

```
  filename = paste0(
    tmpPath, "motivation_linx_annual_plot_", tmpVersion, ".png"
  ),
  type     = "cairo",
  width    = 8,
  height   = 12,
  units    = "in",
  res      = 300
)
plot_motivation_p_holdout(og_TS = lynx, q = 0.7, q_e = 0.7)
dev.off()

# ## > Export as eps ----
setEPS()
postscript(
  file = paste0(
    tmpPath, "motivation_linx_annual_plot_", tmpVersion, ".eps"
  ),
  width      = 8,
  height     = 12,
  horizontal = F
)
plot_motivation_p_holdout(og_TS = lynx, q = 0.7, q_e = 0.7)
dev.off()
```

C.2 The *p*-Holdout Family

code/p_holdout_v2021082001.R

```r
#' The p-Holdout family of validation schemes
#'
#' Partitions the time series according to the p-Holdout, cp-Holdout,
#' and cep-Holdout validation schemes.
#'
#' @author Gustavo Varela-Alvarenga
#'
#' @inheritParams performance_estimation
#'
#' @details These are the same functions seen in the
#' motivation_p_holdout.R file, but adapted to receive the
#' parameter values from the performance_estimation function (inside
#' the workflows.R file).

# ================================================================= #
p_holdout <- function(DATA, test, FUN, form, inner_split, freq, seed,
                      og_TS, error_metric,...){

  # Length of the Training Data and the Test Data --------------------
  length.test.data  <- nrow(test)
  length.train.data <- nrow(DATA)

  # Holdout: Length of the Training Data and the Test Data -----------
  length.validation.set.HO  <-
    ceiling(length.train.data*(1-inner_split))

  # Priodicity/frequency ---------------------------------------------

  tmpFrequency <- freq

  # sp-Holdout: Length of the Estimation Set and the Validation Set----
  if (tmpFrequency > 1) {
    length.validation.set.SPHO <- length.test.data + tmpFrequency
  } else {
    length.validation.set.SPHO <- length.validation.set.HO
  }

  length.estimation.set.SPHO <-
    length.train.data - length.validation.set.SPHO

  # sp-Holdout: Create the Estimation Set and the Validation Set ------
  estimationSetSPHO <- head(DATA, length.estimation.set.SPHO)
  validationSetSPHO <- tail(DATA, length.validation.set.SPHO)

  # Run the pred_model on the split data set  ------------------------
  FUN(estimationSetSPHO, validationSetSPHO, form, seed = seed,
      og_TS = og_TS, error_metric = error_metric)
}

# ================================================================= #
```

```
cp_holdout <- function(DATA, test, FUN, form, inner_split, freq, seed,
                       og_TS, error_metric,...){

  # Length of the Training Data and the Test Data ---------------------
  length.test.data  <- nrow(test)
  length.train.data <- nrow(DATA) # floor(length(og_TS)*outer_split)

  # Holdout: Length of the Training Data and the Test Data ------------
  length.validation.set.HO  <-
    ceiling(length.train.data*(1-inner_split))

  # Priodicity/frequency ----------------------------------------------

  tmpFrequency <- freq

  # cp-Holdout: Length of the Estimation Set and the Validation Set ---

  if (tmpFrequency == 1){
    length.validation.set.CPHO <- length.validation.set.HO
  } else if (isFALSE((length.test.data/tmpFrequency)%%1 == 0)){
    length.validation.set.CPHO <-
      tmpFrequency*ceiling(length.test.data/tmpFrequency)
  } else {
    length.validation.set.CPHO <- length.test.data + tmpFrequency
  }

  length.estimation.set.CPHO <-
    length.train.data - length.validation.set.CPHO

  trainingData <- head(DATA, length.estimation.set.CPHO)
  testData     <- tail(DATA, length.validation.set.CPHO)

  FUN(trainingData, testData, form, seed = seed, og_TS = og_TS,
      error_metric = error_metric)
}

# ================================================================== #
cep_holdout <- function(DATA, test, FUN, form, inner_split, freq,
                        period, seed,
                        outer_split, og_TS, error_metric, ...){

  # Length of the Training Data and the Test Data ---------------------
  length.test.data  <- nrow(test)
  length.train.data <- nrow(DATA) # floor(length(og_TS)*outer_split)

  # Holdout: Length of the Validation Set -----------------------------
  length.validation.set.HO  <-
    ceiling(length.train.data*(1-inner_split))

  # Holdout Scheme: Create the Validation Set -------------------------
  validationSetHO <- tail(DATA, length.validation.set.HO)

  # Periodicity/frequency ---------------------------------------------
  ## Create the test set
```

```
  length.test.og <- ceiling(length(og_TS)*(1-outer_split))
  test.set.og    <- tail(og_TS, length.test.og)

  period.test    <- forecast::findfrequency(validationSetHO)

  ## Get Length
  tmpLength <- ifelse(length.validation.set.HO/length.test.data < 0.5,
                      length.validation.set.HO, length.test.data)

  ## Get Frequency
  tmpFrequency <- ifelse(period < tmpLength, period, period.test)

  # cep-Holdout: Length of the Estimation and Validation SetS ---------

  if (tmpFrequency == 1){
    length.validation.set.CEPHO <- length.validation.set.HO
  } else if (isFALSE((tmpLength/tmpFrequency)%%1 == 0)){
    length.validation.set.CEPHO <-
      tmpFrequency*ceiling(tmpLength/tmpFrequency)
  } else {
    length.validation.set.CEPHO <- tmpLength + tmpFrequency
  }

  length.estimation.set.CEPHO <-
    length.train.data - length.validation.set.CEPHO

  trainingData <- head(DATA, length.estimation.set.CEPHO)
  testData     <- tail(DATA, length.validation.set.CEPHO)

  FUN(trainingData, testData, form, seed = seed, og_TS = og_TS,
      error_metric = error_metric)
}
```

C.3 Forecasts - Cerqueira et al. (2020)

code/perfestimation–rw_v2021082201.R

```
#
    ================================================================================
     #
# Files ----

load("../data/tsl_uni_90_mix.rdata")

#source("../src/utils.r")
#source("../src/estimation-procedures.r")
#source("../src/workflows.r")
#source("../src/metrics.r")
#source("../src/learning-models.r")

source("src/utils_gus.r")
source("src/metrics_V2021071201.r")

source("src/workflows_v2021082001.r")
source("src/p_holdout_v2021082001.r")
source("src/get_ranks_v2021070901.r")
source("src/learning-models_v2021082001.r")
source("src/estimation-procedures_v2021082001.r")
#
    ================================================================================
     #
# Packages ----
library(tsensembler)
library(ranger)
library(Cubist)
library(glmnet)
library(kernlab)
library(nnet)

library(RcppRoll)
library(tseriesChaos)
library(forecast)

#library(parallel)
library("future.apply")

# install.packages("beepr")
# library("beepr") # plays notification sound when R finishes running

#
    ================================================================================
     #
# Initial options -----

form     <- target~.
nfolds   <- 10
tmpSeed  <- 3L
```

```r
tmpOuter <- 0.7 # q_t (percentual of the data that will be training data)
tmpInner <- 0.7 # q_e (percentual of the training data that will be used
                #      for estimation)

#    ============================================================================
# RBR ----
tmpVersionRBR <- "v2021082201"
tmpFileRBR    <- paste0(
  "results/results_cerqueira_RMSE_rbr_", tmpVersionRBR, ".rdata"
)

plan(multisession, workers = 16)

time_rbr_174 <- system.time({
results_cerqueira_rbr <- future_lapply(
  1:length(ts_list),
  function(i) {
    #cat(i, "\n\n")
    ds <- ts_list[[i]]

    x <- workflow(
       ds                   = ds,
       form                 = form,
       predictive_algorithm = "rbr",
       nfolds               = nfolds,
       outer_split          = tmpOuter,
       inner_split          = tmpInner,
       set_seed             = tmpSeed,
       error_metric         = "rmse"
      )

    x
  },
  future.seed = 0xBEEF
)
})
plan(sequential)
time_rbr_174[3]/3600 # new system + future_lapply: ~1.404739 hrs, 1.987 with 10 workers

results_cerqueira_RMSE_rbr_ranks     <- get_ranks_gus(results_cerqueira_rbr)
results_cerqueira_RMSE_rbr_mean_rank <- rowMeans(results_cerqueira_RMSE_rbr_ranks$fr_abs_rank)
sort(results_cerqueira_RMSE_rbr_mean_rank)

save(results_cerqueira_rbr, file = tmpFileRBR)
beepr::beep("fanfare")

```

```
#
    ============================================================================
     #
# RF ----
tmpVersionRF <- "v2021082201"
tmpFileRF    <- paste0(
  "results/results_cerqueira_RMSE_rf_", tmpVersionRF, ".rdata"
)

plan(multisession, workers = 16)

time_rf_174 <- system.time({
  results_cerqueira_rf <- future_lapply(
    1:length(ts_list),
    function(i) {
      #cat(i, "\n\n")
      ds <- ts_list[[i]]

      x <- workflow(
        ds                    = ds,
        form                  = form,
        predictive_algorithm = "rf",
        nfolds                = nfolds,
        outer_split           = tmpOuter,
        inner_split           = tmpInner,
        set_seed              = tmpSeed,
        error_metric          = "rmse"
      )

      x
    },
    future.seed = 0xBEEF
  )
})
plan(sequential)
time_rf_174[3]/60 # new system + future_lapply: ~ 33.25 min (54.76 if
    half power)

results_cerqueira_RMSE_rf_ranks     <- get_ranks_gus(results_cerqueira_
    rf)
results_cerqueira_RMSE_rf_mean_rank <- rowMeans(results_cerqueira_RMSE_
    rf_ranks$fr_abs_rank)
sort(results_cerqueira_RMSE_rf_mean_rank)

save(results_cerqueira_rf, file = tmpFileRF)
beepr::beep("fanfare")

#
    ============================================================================
     #
# RIDGE ----
tmpVersionRIDGE <- "v2021082201"
tmpFileRIDGE    <- paste0(
  "results/results_cerqueira_RMSE_ridge_", tmpVersionRIDGE, ".rdata"
```

```
)

plan(multisession, workers = 16)

time_ridge_174 <- system.time({
  results_cerqueira_ridge <- future_lapply(
    1:length(ts_list),
    function(i) {
      #cat(i, "\n\n")
      ds <- ts_list[[i]]

      x <- workflow(
        ds                    = ds,
        form                  = form,
        predictive_algorithm = "lasso", # name is lasso, but it's
    running ridge
        nfolds                = nfolds,
        outer_split           = tmpOuter,
        inner_split           = tmpInner,
        set_seed              = tmpSeed,
        error_metric          = "rmse"
      )

      x
    },
    future.seed = 0xBEEF
  )
})
plan(sequential)
time_ridge_174[3]/60 # new system + future_lapply: ~9.297 min

results_cerqueira_RMSE_ridge_ranks     <- get_ranks_gus(results_
    cerqueira_ridge)
results_cerqueira_RMSE_ridge_mean_rank <- rowMeans(results_cerqueira_
    RMSE_ridge_ranks$fr_abs_rank)
sort(results_cerqueira_RMSE_ridge_mean_rank)

save(results_cerqueira_ridge, file = tmpFileRIDGE)
beepr::beep("fanfare")
```

C.4 M4 Competition Sample Selection and Forecasts

code/get_M4_datasets_v2021071001.R

```r
# ========================================================================== #

# Install the M4comp2018 package
#install.packages("https://github.com/carlanetto/M4comp2018/releases/
    download/0.2.0/M4comp2018_0.2.0.tar.gz",
#                 repos=NULL)

# Load the M4 list with all 100k data sets
M4data <- M4comp2018::M4

# Get sample sizes
M4_sample <- sapply(
  X = 1:length(M4data),
  function(X) M4data[[X]]$n
)

# Get their frequencies
M4_freqs <- sapply(
  X = 1:length(M4data),
  function(X) stats::frequency(M4data[[X]]$x)
)
#table(M4_freqs) # frequency > 1: freqs 4, 12, 24

# Get estimated frequencies
M4_freqs_forecast <- sapply(
  X = 1:length(M4data),
  function(X) forecast::findfrequency(M4data[[X]]$x)
)
#table(M4_freqs_forecast)

# Get  series with either period greater than 1,
# and minimum sample size of 100
M4_periodic4 <- which(
  M4_freqs_forecast == M4_freqs & M4_sample >= 100 & M4_freqs > 1
)

head(sort(M4_periodic))
head(sort(M4_periodic2))
head(sort(M4_periodic3))
head(sort(M4_periodic4))

# Randomly select 1,000 series from the list of series
tmpSeed <- 3
set.seed(tmpSeed)
M4_periodic_sample_number_v02 <-
  sample(M4_periodic, size = 1000, replace = F)

# Get the selected time series
M4_periodic_sample_series_v02 <- lapply(
  X = 1:length(M4_periodic_sample_number_v02),
```

```
  function(X) {
    i <- M4_periodic_sample_number_v02[[X]]
    M4data[[i]]
  }
)

# Save

## Sample Numbers
save(
  M4_periodic_sample_number_v02,
  file = "data/M4_periodic_sample_number_v02.rdata"
)

## Selected series
save(
  M4_periodic_sample_series_v02,
  file = "data/M4_periodic_sample_series_v02.rdata"
)
```

code/perfestimation–rw_M4_v2021082201.R

```
#
    ================================================================================
     #
# Files ----

load("data/M4_periodic_sample_series_v02.rdata")

#source("../src/utils.r")
#source("../src/estimation-procedures.r")
#source("../src/workflows.r")
#source("../src/metrics.r")
#source("../src/learning-models.r")

source("src/utils_gus.r")
source("src/metrics_V2021071201.r")

source("src/workflows_v2021082001.r")
source("src/p_holdout_v2021082001.r")
source("src/get_ranks_v2021070901.r")
source("src/learning-models_v2021082001.r")
source("src/estimation-procedures_v2021082001.r")

#
    ================================================================================
     #
# Packages ----
library(tsensembler)
library(ranger)
library(Cubist)
library(glmnet)
library(kernlab)
library(nnet)
```

```r
library(RcppRoll)
library(tseriesChaos)
library(forecast)

#library(parallel)
library("future.apply")

# install.packages("beepr")
# library("beepr") # plays notification sound when R finishes running
#
    ================================================================================
     #
# Initial options -----

form     <- target~.
nfolds   <- 10
tmpSeed  <- 3L

#
    ================================================================================
     #
# RBR ----
tmpM4VersionRBR <- "v2021082201"
tmpM4FileRBR    <- paste0(
  "results/results_M4_RMSE_rbr_", tmpM4VersionRBR, ".rdata"
)

plan(multisession, workers = 8)

time_RMSE_rbr_M4 <- system.time({
  results_M4_rbr <- future_lapply(
    seq_along(M4_periodic_sample_series_v02),
    function(i) {
      #cat(i, "\n\n")
      M4.ds <- M4_periodic_sample_series_v02[[i]]

      ds <- ts(
        c(M4.ds$x, M4.ds$xx),
        start      = tsp(M4.ds$x)[1],
        end        = tsp(M4.ds$xx)[2], ## xx not x
        frequency  = tsp(M4.ds$x)[3]
      )

      x <- workflow(
        ds                   = ds,
        form                 = form,
        predictive_algorithm = "rbr",
        nfolds               = nfolds,
        outer_split          = 1 - (M4.ds$h/length(ds)),
        inner_split          = 1 - (M4.ds$h/M4.ds$n),
        set_seed             = tmpSeed,
        n_M4                 = M4.ds$n,
        error_metric         = "rmse"
```

```
      )

      x
    },
    future.seed = 0xBEEF
  )
})
plan(sequential)
time_RMSE_rbr_M4[3]/60 # new system + future_lapply: ~ 7.777 min (11.8 min)

results_M4_RMSE_rbr_ranks     <- get_ranks_gus(results_M4_rbr)
results_M4_RMSE_rbr_mean_rank <- rowMeans(results_M4_RMSE_rbr_ranks$fr_abs_rank)
sort(results_M4_RMSE_rbr_mean_rank)

save(results_M4_rbr, file = tmpM4FileRBR)
beepr::beep("fanfare")

# ============================================================================ #
# RF ----

tmpM4VersionRF <- "v2021082201"
tmpM4FileRF    <- paste0(
  "results/results_M4_RMSE_rf_", tmpM4VersionRF, ".rdata"
)

plan(multisession, workers = 16)

time_RMSE_rf_M4 <- system.time({
  results_M4_rf <- future_lapply(
    seq_along(M4_periodic_sample_series_v02),
    function(i) {
      #cat(i, "\n\n")
      M4.ds <- M4_periodic_sample_series_v02[[i]]

      ds <- ts(
        c(M4.ds$x, M4.ds$xx),
        start      = tsp(M4.ds$x)[1],
        end        = tsp(M4.ds$xx)[2], ## xx not x
        frequency  = tsp(M4.ds$x)[3]
      )

      x <- workflow(
        ds                    = ds,
        form                  = form,
        predictive_algorithm = "rf",
        nfolds                = nfolds,
        outer_split           = 1 - (M4.ds$h/length(ds)),
        inner_split           = 1 - (M4.ds$h/M4.ds$n),
        set_seed              = tmpSeed,
        n_M4                  = M4.ds$n,
```

```
        error_metric         = "rmse"
      )

      x
    },
    future.seed = 0xBEEF
  )
})
plan(sequential)
time_RMSE_rf_M4[3]/60 # new system + future_lapply: ~ 5.319 min (8 min)

results_M4_RMSE_rf_ranks     <- get_ranks_gus(results_M4_rf)
results_M4_RMSE_rf_mean_rank <- rowMeans(results_M4_RMSE_rf_ranks$fr_
    abs_rank)
sort(results_M4_RMSE_rf_mean_rank)

save(results_M4_rf, file = tmpM4FileRF)
beepr::beep("fanfare")

#
    ================================================================================
     #
# RIDGE ----

tmpM4VersionRIDGE <- "v2021082201"
tmpM4FileRIDGE    <- paste0(
  "results/results_M4_RMSE_ridge_", tmpM4VersionRIDGE, ".rdata"
)

plan(multisession, workers = 16)

time_RMSE_ridge_M4 <- system.time({
  results_M4_ridge <- future_lapply(
    seq_along(M4_periodic_sample_series_v02),
    function(i) {
      #cat(i, "\n\n")
      M4.ds <- M4_periodic_sample_series_v02[[i]]

      ds <- ts(
        c(M4.ds$x, M4.ds$xx),
        start      = tsp(M4.ds$x)[1],
        end        = tsp(M4.ds$xx)[2], ## xx not x
        frequency  = tsp(M4.ds$x)[3]
      )

      x <- workflow(
        ds                   = ds,
        form                 = form,
        predictive_algorithm = "lasso", # name is lasso, but it's
    running ridge
        nfolds               = nfolds,
        outer_split          = 1 - (M4.ds$h/length(ds)),
        inner_split          = 1 - (M4.ds$h/M4.ds$n),
        set_seed             = tmpSeed,
```

```
        n_M4                  = M4.ds$n,
        error_metric          = "rmse"
      )
      x
    },
    future.seed = 0xBEEF
  )
})
plan(sequential)
time_RMSE_ridge_M4[3]/60 # new system + future_lapply: 1.82 min (2.71
    min)

results_M4_RMSE_ridge_ranks     <- get_ranks_gus(results_M4_ridge)
results_M4_RMSE_ridge_mean_rank <- rowMeans(results_M4_RMSE_ridge_ranks
    $fr_abs_rank)
sort(results_M4_RMSE_ridge_mean_rank)

save(results_M4_ridge, file = tmpM4FileRIDGE)
beepr::beep("fanfare")
```

C.5 Monte Carlo Simulations and Forecasts

code/perfestimation–TS3_v2021070901.R

```
# ========================================================================= #
# Files ----

#source("../src/utils.r")
#source("../src/estimation-procedures.r")
#source("../src/workflows.r")
#source("../src/metrics.r")
#source("../src/learning-models.r")

source("src/utils_gus.r")
source("src/metrics_gus.r")

source("src/workflows_v2021070901.r")
source("src/p_holdout_v2021070901.r")
source("src/get_ranks_v2021070901.r")
source("src/learning-models_v2021070901.r")
source("src/estimation-procedures_v2021070901.r")

# ========================================================================= #
# Packages ----
library(tsensembler)
library(ranger)
library(Cubist)
library(glmnet)
library(kernlab)
library(nnet)

library(RcppRoll)
library(tseriesChaos)
library(forecast)

#library(parallel)
library("future.apply")

# install.packages("beepr")
# library("beepr") # plays notification sound when R finishes running

# ========================================================================= #
# Initial options -----

form     <- target~.
nfolds   <- 10
tmpSeed  <- 3L
tmpOuter <- 0.7 # q_t (percentual of the data used training data)
tmpInner <- 0.8 # q_e (percentual of the training data that will be
                # used for estimation)

# ========================================================================= #
# Simulate the data set -----
```

```r
mcreps <- 1000
seq.   <- seq_len(mcreps)
ts.len <- 200

data(USAccDeaths)

# TS3 ----

arima.fit <- forecast::Arima(
  y        = USAccDeaths,
  order    = c(12,0,0),
  seasonal = list(order = c(1, 0, 0), period = frequency(USAccDeaths))
)

set.seed(tmpSeed)
TS3 <- lapply(
  seq.,
  function(j) simulate(object = arima.fit, nsim = ts.len)
)

#check_freq_TS3 <- sapply(TS3, frequency)
#table(check_freq_TS3)

#check_period_TS3 <- sapply(TS3, forecast::findfrequency)
#table(check_period_TS3)

# ------------------------------------------------------------------ #
# TS4 ----

arima.fit2 <- forecast::Arima(
  y        = USAccDeaths,
  order    = c(12,0,0),
  seasonal = list(order = c(1, 1, 0), period = frequency(USAccDeaths))
)

set.seed(tmpSeed)
TS4 <- lapply(
  seq.,
  function(j) simulate(object = arima.fit2, nsim = ts.len)
)

# ================================================================== #
# RBR ----
tmpVersionRBR <- "v2021070901"
tmpFileRBR    <- paste0(
  "results/results_TS3_rbr_", tmpVersionRBR, ".rdata"
)

plan(multisession, workers = 12)

time_rbr_174 <- system.time({
  results_TS3_rbr <- future_lapply(
    seq_along(TS3),
    function(i) {
```

```
      #cat(i, "\n\n")
      ds <- TS3[[i]]

      x <- workflow(
        ds                    = ds,
        form                  = form,
        predictive_algorithm = "rbr",
        nfolds                = nfolds,
        outer_split           = tmpOuter,
        inner_split           = tmpInner,
        set_seed              = tmpSeed,
        is_embedded           = TRUE,
        is_TS3                = TRUE,
        is_TS4                = FALSE
      )

      x
    },
    future.seed = 0xBEEF
  )
})
#plan(sequential)
time_rbr_174[3]/60 # new system + future_lapply: ~4.681667 min

results_TS3_rbr_ranks     <- get_ranks_gus(results_TS3_rbr)
results_TS3_rbr_mean_rank <- rowMeans(
  results_TS3_rbr_ranks$fr_abs_rank
)
sort(results_TS3_rbr_mean_rank)

save(results_TS3_rbr, file = tmpFileRBR)
beepr::beep("fanfare")

# ============================================================== #
# RBR ----
tmpVersionRBR <- "v2021070901"
tmpFileRBR    <- paste0(
  "results/results_TS4_rbr_", tmpVersionRBR, ".rdata"
)

#plan(multisession, workers = 16)

time_rbr_174 <- system.time({
  results_TS4_rbr <- future_lapply(
    seq_along(TS4),
    function(i) {
      #cat(i, "\n\n")
      ds <- TS4[[i]]

      x <- workflow(
        ds                    = ds,
        form                  = form,
        predictive_algorithm = "rbr",
        nfolds                = nfolds,
```

```
        outer_split           = tmpOuter,
        inner_split           = tmpInner,
        set_seed              = tmpSeed,
        is_embedded           = TRUE,
        is_TS3                = FALSE,
        is_TS4                = TRUE
      )

      x
    },
    future.seed = 0xBEEF
  )
})
plan(sequential)
time_rbr_174[3]/60 # new system + future_lapply: ~4.68 min

results_TS4_rbr_ranks     <- get_ranks_gus(results_TS4_rbr)
results_TS4_rbr_mean_rank <- rowMeans(
  results_TS4_rbr_ranks$fr_abs_rank
)
sort(results_TS4_rbr_mean_rank)

save(results_TS4_rbr, file = tmpFileRBR)
beepr::beep("fanfare")

# ============================================================================ #
# RF ----
tmpVersionRF <- "v2021070901"
tmpFileRF    <- paste0(
  "results/results_TS3_rf_", tmpVersionRF, ".rdata"
)

plan(multisession, workers = 16)

time_rf_174 <- system.time({
  results_TS3_rf <- future_lapply(
    seq_along(TS3),
    function(i) {
      #cat(i, "\n\n")
      ds <- TS3[[i]]

      x <- workflow(
        ds                    = ds,
        form                  = form,
        predictive_algorithm = "rf",
        nfolds                = nfolds,
        outer_split           = tmpOuter,
        inner_split           = tmpInner,
        set_seed              = tmpSeed,
        is_embedded           = TRUE,
        is_TS3                = TRUE,
        is_TS4                = FALSE
      )
```

```
      x
    },
    future.seed = 0xBEEF
  )
})
plan(sequential)
time_rf_174[3]/60 # new system + future_lapply: ~4.332667min

results_TS3_rf_ranks     <- get_ranks_gus(results_TS3_rf)
results_TS3_rf_mean_rank <- rowMeans(results_TS3_rf_ranks$fr_abs_rank)
sort(results_TS3_rf_mean_rank)

save(results_TS3_rf, file = tmpFileRF)
beepr::beep("fanfare")

# ============================================================== #
# RF ----
tmpVersionRF <- "v2021070901"
tmpFileRF    <- paste0(
  "results/results_TS4_rf_", tmpVersionRF, ".rdata"
)

plan(multisession, workers = 16)

time_rf_174 <- system.time({
  results_TS4_rf <- future_lapply(
    seq_along(TS4),
    function(i) {
      #cat(i, "\n\n")
      ds <- TS4[[i]]

      x <- workflow(
        ds                   = ds,
        form                 = form,
        predictive_algorithm = "rf",
        nfolds               = nfolds,
        outer_split          = tmpOuter,
        inner_split          = tmpInner,
        set_seed             = tmpSeed,
        is_embedded          = TRUE,
        is_TS3               = FALSE,
        is_TS4               = TRUE
      )

      x
    },
    future.seed = 0xBEEF
  )
})
plan(sequential)
time_rf_174[3]/60 # new system + future_lapply: ~4.225 min

results_TS4_rf_ranks     <- get_ranks_gus(results_TS4_rf)
results_TS4_rf_mean_rank <- rowMeans(results_TS4_rf_ranks$fr_abs_rank)
```

```r
sort(results_TS4_rf_mean_rank)

save(results_TS4_rf, file = tmpFileRF)
beepr::beep("fanfare")

# ============================================================== #
# RIDGE ----
tmpVersionRIDGE <- "v2021070901"
tmpFileRIDGE    <- paste0(
  "results/results_TS3_ridge_", tmpVersionRIDGE, ".rdata"
)

plan(multisession, workers = 16)

time_ridge_174 <- system.time({
  results_TS3_ridge <- future_lapply(
    seq_along(TS3),
    function(i) {
      #cat(i, "\n\n")
      ds <- TS3[[i]]

      x <- workflow(
        ds                    = ds,
        form                  = form,
        predictive_algorithm = "lasso", # name is lasso, but it's
                                        # running ridge
        nfolds                = nfolds,
        outer_split           = tmpOuter,
        inner_split           = tmpInner,
        set_seed              = tmpSeed,
        is_embedded           = TRUE,
        is_TS3                = TRUE,
        is_TS4                = FALSE
      )

      x
    },
    future.seed = 0xBEEF
  )
})
plan(sequential)
time_ridge_174[3]/60 # new system + future_lapply: ~1.6025 min

results_TS3_ridge_ranks     <- get_ranks_gus(results_TS3_ridge)
results_TS3_ridge_mean_rank <- rowMeans(
  results_TS3_ridge_ranks$fr_abs_rank
)
sort(results_TS3_ridge_mean_rank)

save(results_TS3_ridge, file = tmpFileRIDGE)
beepr::beep("fanfare")

# ============================================================== #
# RIDGE ----
```

```r
tmpVersionRIDGE <- "v2021070901"
tmpFileRIDGE    <- paste0(
  "results/results_TS4_ridge_", tmpVersionRIDGE, ".rdata"
)

plan(multisession, workers = 16)

time_ridge_174 <- system.time({
  results_TS4_ridge <- future_lapply(
    seq_along(TS4),
    function(i) {
      #cat(i, "\n\n")
      ds <- TS4[[i]]

      x <- workflow(
        ds                    = ds,
        form                  = form,
        predictive_algorithm = "lasso", # name is lasso, but it's
                                        # running ridge
        nfolds                = nfolds,
        outer_split           = tmpOuter,
        inner_split           = tmpInner,
        set_seed              = tmpSeed,
        is_embedded           = TRUE,
        is_TS3                = FALSE,
        is_TS4                = TRUE
      )

      x
    },
    future.seed = 0xBEEF
  )
})
plan(sequential)
time_ridge_174[3]/60 # new system + future_lapply: ~1.6025 min

results_TS4_ridge_ranks     <- get_ranks_gus(results_TS4_ridge)
results_TS4_ridge_mean_rank <- rowMeans(
  results_TS4_ridge_ranks$fr_abs_rank
)
sort(results_TS4_ridge_mean_rank)

save(results_TS4_ridge, file = tmpFileRIDGE)
beepr::beep("fanfare")
```

C.6 Create Plots - Cerqueira et al. and M4 Competition

code/rw_analysis_v2021_08_22_01.R

```
#
    ##############################################################################
     #
#' Evaluating the performance of estimation methods
#
#' This is an extension of the code from Cerqueira, Torgo, and Mozetic.
#' "Evaluating time series forecasting models: an empirical study
#' on performance estimation methods".
#' In: Machine Learning (2020) 109:1997-2028
#'
#' Modified by: Gustavo Varela-Alvarenga
#' Date:        05/30/2021
#
    ##############################################################################
     #
#
    \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
     ####
# > Packages
    ============================================================================
# library(forecast)
# library(tsensembler)
# library(ranger)
# load code for `avg_rank_plot`
source("src/plots_v2021071301.r")
#library(scmamp)
# --- #

#
    \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
     ####
## >  Path to save plots to ----
tmpSavePath <- "results/plots"
tmpSavePathTables <- "results/tables/"

#
    \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
     ####
# > Helpers
         ####
# |_ Helper Function: get_ranks_p_holdout
    ====================================
# Gets final estimation errors, and ranks
get_ranks_p_holdout <- function(final_results_data){
  # --- #
  # get estimation errors
  err_estimation <- lapply(
    X = final_results_data,
    function(X) tryCatch(X$err_estimation, error =function(e) {NULL})
  )
```

```
38
   err_estimation <- err_estimation[!sapply(err_estimation, is.null)]
40 # --- #
   # create df with final estimation errors
42 fr <- do.call(rbind, err_estimation)
   fr <- as.data.frame(fr)
44 rownames(fr) <- NULL

46 colnames(fr) <-
     c("p-holdout",      # <---- new method
48     "cp-holdout",     # <---- new method
       "cep-holdout",    # <---- new method
50     "CV", "CV-Bl", "CV-Mod","CV-hvBl",
       "Preq-Bls", "Preq-Sld-Bls",
52     "Preq-Bls-Gap","Holdout", "Rep-Holdout",
       # "Preq-Slide","Preq-Grow" # the order in the original is
     switched
54     "Preq-Grow", "Preq-Slide"
     )
56 # --- #
   # get ranks for each estimation procedure
58 fr_abs <- abs(fr)
   fr_abs_rank <- apply(fr_abs, 1, rank)
60
   # --- #
62 # return df with final estimation errors, and another one with the
     ranks
   return(list(fr = fr, fr_abs_rank = fr_abs_rank))
64 }

66 # |_ Helper Function: plot_avg_ranks_ts_types
     ==================================
   plot_avg_ranks_ts_types <- function(results, is_stat = NULL, df.source)
     {
68
   # break data into stationary data or not (or both)
70 tmpAll           <- results
   if(!is.null(is_stat)){
72   tmpStationary    <- results[is_stat]
     tmpNonStationary <- results[!is_stat]
74 }


76
   # ---- #
78 # get ranks
   ranksAll <- get_ranks_p_holdout(tmpAll)
80 if(!is.null(is_stat)){
     ranksStationary    <- get_ranks_p_holdout(tmpStationary)
82   ranksNonStationary <- get_ranks_p_holdout(tmpNonStationary)
   }
84
   # ---- #
86 # plot ranks
   ## function 'avg_rank_plot' comes from
```

```
  ## source("../src/plots_gus.r")
  ## I've changed the original 'plots.r' to add a greeb color to my
   method's
  ## results' bar plot.
  pRanksAll <- avg_rank_plot(
    avg  = rowMeans(ranksAll$fr_abs_rank),
    sdev = apply(ranksAll$fr_abs_rank,1, sd)
  )

  if(!is.null(is_stat)){
    pRanksStationary <- avg_rank_plot(
      avg  = rowMeans(ranksStationary$fr_abs_rank),
      sdev = apply(ranksStationary$fr_abs_rank,1, sd)
    )

    pRanksNonStationary <- avg_rank_plot(
      avg  = rowMeans(ranksNonStationary$fr_abs_rank),
      sdev = apply(ranksNonStationary$fr_abs_rank,1, sd)
    )

    finalPlotRanks <- ggpubr::ggarrange(
      plotlist = list(pRanksAll, pRanksStationary, pRanksNonStationary)
   ,
      ncol     = 1,
      nrow     = 3,
      labels   = c("All", "Stationary", "Non-Stationary")
    )
  } else {
    finalPlotRanks <- ggpubr::ggarrange(
      plotlist = list(pRanksAll),
      ncol     = 1,
      nrow     = 1,
      labels   = c("All")
    )
  }
  # ---- #
  # return plot with annotations
  ggpubr::annotate_figure(
    finalPlotRanks,
    bottom = ggpubr::text_grob(
      label = paste0(
        "Plot by: Varela-Alvarenga and Kedem (2021). \n",
        "Data source: ", df.source
      ),
      hjust = 1,
      x     = 1,
      face  = "italic",
      size  = 10
    )
  )
}

#
    ------------------------------------------------------------------------
```

```
    #
# helper function - get data and plot ----
plot_log_diff_ts_types <- function(results, is_stat = NULL, df.source){

  # break data into stationary data or not (or both)
  tmpAll           <- results
  if(!is.null(is_stat)){
    tmpStationary    <- results[is_stat]
    tmpNonStationary <- results[!is_stat]
  }
  # ---- #
  # get ranks
  ranksAll           <- get_ranks_p_holdout(tmpAll)
  if(!is.null(is_stat)){
    ranksStationary    <- get_ranks_p_holdout(tmpStationary)
    ranksNonStationary <- get_ranks_p_holdout(tmpNonStationary)
  }
  # ---- #
  # plot log diff

  ## function 'percdiff_plot_log' comes from
  ## source("../src/plots_gus.r")
  pLogDiffAll           <- percdiff_plot_log(ranksAll$fr)
  if(!is.null(is_stat)){
    pLogDiffStationary    <- percdiff_plot_log(ranksStationary$fr)
    pLogDiffNonStationary <-
      percdiff_plot_log(ranksNonStationary$fr) +
      labs(x = "Solid line: Median. Dashed line: Mean")

    finalPlotLogDiff <- ggpubr::ggarrange(
      plotlist = list(pLogDiffAll, pLogDiffStationary,
   pLogDiffNonStationary),
      ncol     = 1,
      nrow     = 3,
      labels   = list("All", "Stationary", "Non-Stationary"),
      hjust    = c(-1, -0.25, -0.15),
      vjust    = c(1.5, 0.25, 0.25)
    )
  } else {
    finalPlotLogDiff <- ggpubr::ggarrange(
      plotlist = list(pLogDiffAll),
      ncol     = 1,
      nrow     = 1,
      labels   = c("All")
    )
  }
  # ---- #

  # ---- #
  # return plot with annotations
  ggpubr::annotate_figure(
    finalPlotLogDiff,
    bottom = ggpubr::text_grob(
      label = paste0(
```

```
        "Plot by: Varela-Alvarenga and Kedem (2021). \n",
        "Data source: ", df.source
      ),
      hjust = 1,
      x     = 1,
      face  = "italic",
      size  = 10
    ),
    left = ggpubr::text_grob(
      label = "Percentual difference to true loss",
      face  = "bold",
      size  = 12,
      rot   = 90
    )
  )
}

#    ------------------------------------------------------------------------
     #
# helper function - returns log diff values ----

log_diff_ts_types <- function(results,
                              is_stat,
                              statistic = c("Mean", "Median", "Std.Dev.
    ", "IQR")
                              ){

  # break data into stationary data or not (or both)
  tmpAll           <- results
  tmpStationary    <- results[is_stat]
  tmpNonStationary <- results[!is_stat]

  # ---- #
  # get ranks
  ranksAll           <- get_ranks_p_holdout(tmpAll)
  ranksStationary    <- get_ranks_p_holdout(tmpStationary)
  ranksNonStationary <- get_ranks_p_holdout(tmpNonStationary)

  # ---- #
  # calculate log diff
  log_trans <- function(x) sign(x) * log(abs(x) + 1)

  rAll               <- reshape2::melt(ranksAll$fr, id.vars = NULL)
  rAll$log           <- log_trans(rAll$value)
  rStationary        <- reshape2::melt(ranksStationary$fr, id.vars =
    NULL)
  rStationary$log    <- log_trans(rStationary$value)
  rNonStationary     <- reshape2::melt(ranksNonStationary$fr, id.vars =
     NULL)
  rNonStationary$log <- log_trans(rNonStationary$value)

  # ---- #
  # calculate summary by scheme
```

```
summaryAll <- aggregate(
  x   = rAll$log,
  by  = list(scheme = rAll$variable),
  FUN = summary
)
summaryStationary <- aggregate(
  x   = rStationary$log,
  by  = list(scheme = rStationary$variable),
  FUN = summary
)
summaryNonStationary <- aggregate(
  x   = rNonStationary$log,
  by  = list(scheme = rNonStationary$variable),
  FUN = summary
)

# ---- #
# calculate std dev by scheme
sdAll <- aggregate(
  x   = rAll$log,
  by  = list(scheme = rAll$variable),
  FUN = sd
)
sdStationary <- aggregate(
  x   = rStationary$log,
  by  = list(scheme = rStationary$variable),
  FUN = sd
)
sdNonStationary <- aggregate(
  x   = rNonStationary$log,
  by  = list(scheme = rNonStationary$variable),
  FUN = sd
)
# ---- #
# calculate IQR by scheme
iqrAll <- aggregate(
  x   = rAll$log,
  by  = list(scheme = rAll$variable),
  FUN = IQR
)
iqrStationary <- aggregate(
  x   = rStationary$log,
  by  = list(scheme = rStationary$variable),
  FUN = IQR
)
iqrNonStationary <- aggregate(
  x   = rNonStationary$log,
  by  = list(scheme = rNonStationary$variable),
  FUN = IQR
)
# ---- #
# Get final summary table (with std dev and IQR)
summaryAll <- cbind.data.frame(
  summaryAll,
```

```
    "Std.Dev." = sdAll$x,
    "IQR"      = iqrAll$x
  )
  summaryStationary <- cbind.data.frame(
    summaryStationary,
    "Std.Dev." = sdStationary$x,
    "IQR"      = iqrStationary$x
  )
  summaryNonStationary <- cbind.data.frame(
    summaryNonStationary,
    "Std.Dev." = sdNonStationary$x,
    "IQR"      = iqrNonStationary$x
  )
  # ---- #
  # Order final summary table by statistic
  summaryAll <- summaryAll[order(abs(summaryAll$x[, statistic])), ]

  summaryStationary <-
    summaryStationary[order(abs(summaryStationary$x[, statistic])), ]

  summaryNonStationary <-
    summaryNonStationary[order(abs(summaryNonStationary$x[, statistic])
    ), ]

  # ---- #
  # return final list
  list(
    All           = summaryAll,
    Stationary    = summaryStationary,
    NonStationary = summaryNonStationary
  )
}

#    \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
     ####

#    ============================================================================
     #
#    ============================================================================
     #
#####                            Stationary data
        #####
#    ============================================================================
     #
#    ============================================================================
     #

# |_ Import Original Data set
     ================================================
```

```r
load("../stationarity_tsdl.rdata")

# |_ Indicator if the series is stationary =====================================
tmp_is_stat <- is_stationary_2ensemble
#table(is_stationary_2ensemble) # 97 stationary time series (out of 174, ~56%)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ============================================================================ #
# ============================================================================ #
####                          Results RBR                                   ####
# ============================================================================ #
# ============================================================================ #
## \___ Import Data ------------------------------------------------------------
tmpVersionRBR <- "v2021082201"
tmpFileRBR    <- paste0(
  "results/results_cerqueira_RMSE_rbr_", tmpVersionRBR, ".rdata"
)
tmpRBR <- get(load(tmpFileRBR))

# |_ Plot the average ranks ====================================================
plot_RBR_avg_rank <- plot_avg_ranks_ts_types(
  results   = tmpRBR,
  is_stat   = tmp_is_stat,
  df.source = "Cerqueira et al. (2020)."
)
plot_RBR_avg_rank

## \___ Export as PNG ----------------------------------------------------------
ggplot2::ggsave(
  filename = paste0("RBR_ranks_plot_RMSE_",tmpVersionRBR,".png"),
  path     = tmpSavePath,
  plot     = plot_RBR_avg_rank,
  device   = "png",
  type     = "cairo",
```

```
  width    = 8,
  height   = 12,
  units    = "in"
)

## \___ Export as EPS
    ------------------------------------------------------------
ggplot2::ggsave(
  filename = paste0("RBR_ranks_plot_RMSE_",tmpVersionRBR,".eps"),
  path     = tmpSavePath,
  plot     = plot_RBR_avg_rank,
  device   = "eps",
  width    = 8,
  height   = 12,
  units    = "in"
)

# |_ Plot the Log-Diff
    ============================================================
plot_RBR_log_diff <- plot_log_diff_ts_types(
  results = tmpRBR,
  is_stat = tmp_is_stat,
  df.source = "Cerqueira et al. (2020)."
)
plot_RBR_log_diff

## \___ Export as PNG
    ------------------------------------------------------------
ggplot2::ggsave(
  filename = paste0("RBR_log_diff_plot_RMSE_",tmpVersionRBR,".png"),
  path     = tmpSavePath,
  plot     = plot_RBR_log_diff,
  device   = "png",
  type     = "cairo",
  width    = 12,
  height   = 12,
  units    = "in"
)

## \___ Export as EPS
    ------------------------------------------------------------
ggplot2::ggsave(
  filename = paste0("RBR_log_diff_plot_RMSE_",tmpVersionRBR,".eps"),
  path     = tmpSavePath,
  plot     = plot_RBR_log_diff,
  device   = "eps",
  width    = 12,
  height   = 12,
  units    = "in"
)

# |_ Table with values of the Log-Diff
    ========================================
log_diff_RBR_Mean   <- log_diff_ts_types(tmpRBR, tmp_is_stat, "Mean")
```

```r
log_diff_RBR_Median <- log_diff_ts_types(tmpRBR, tmp_is_stat, "Median")

## \___ Export as TXT ---------------------------------------------------------
capture.output(
  log_diff_RBR_Mean,
  file = paste0(tmpSavePathTables, "RBR_log_diff_Mean_RMSE_", tmpVersionRBR,".txt")
)
capture.output(
  log_diff_RBR_Median,
  file = paste0(tmpSavePathTables, "RBR_log_diff_Median_RMSE_", tmpVersionRBR,".txt")
)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# =============================================================================#
# =============================================================================#
####                           Results RF                                  ####
# =============================================================================#
# =============================================================================#
## \___ Import Data -----------------------------------------------------------
tmpVersionRF <- "v2021082201"
tmpFileRF    <- paste0(
  "results/results_cerqueira_RMSE_rf_", tmpVersionRF, ".rdata"
)
tmpRF <- get(load(tmpFileRF))

# |_ Plot the average ranks ===================================================
plot_RF_avg_rank <- plot_avg_ranks_ts_types(
  results   = tmpRF,
  is_stat   = tmp_is_stat,
  df.source = "Cerqueira et al. (2020)."
)
plot_RF_avg_rank

## \___ Export as PNG ---------------------------------------------------------
ggplot2::ggsave(
  filename = paste0("RF_ranks_plot_RMSE_",tmpVersionRF,".png"),
  path     = tmpSavePath,
```

```
  plot     = plot_RF_avg_rank,
  device   = "png",
  type     = "cairo",
  width    = 8,
  height   = 12,
  units    = "in"
)

## \___ Export as EPS
    ------------------------------------------------------------
ggplot2::ggsave(
  filename = paste0("RF_ranks_plot_RMSE_",tmpVersionRF,".eps"),
  path     = tmpSavePath,
  plot     = plot_RF_avg_rank,
  device   = "eps",
  width    = 8,
  height   = 12,
  units    = "in"
)

# |_ Plot the Log-Diff
    ============================================================
plot_RF_log_diff <- plot_log_diff_ts_types(
  results = tmpRF,
  is_stat = tmp_is_stat,
  df.source = "Cerqueira et al. (2020)."
)
plot_RF_log_diff

## \___ Export as PNG
    ------------------------------------------------------------
ggplot2::ggsave(
  filename = paste0("RF_log_diff_plot_RMSE_",tmpVersionRF,".png"),
  path     = tmpSavePath,
  plot     = plot_RF_log_diff,
  device   = "png",
  type     = "cairo",
  width    = 12,
  height   = 12,
  units    = "in"
)

## \___ Export as EPS
    ------------------------------------------------------------
ggplot2::ggsave(
  filename = paste0("RF_log_diff_plot_RMSE_",tmpVersionRF,".eps"),
  path     = tmpSavePath,
  plot     = plot_RF_log_diff,
  device   = "eps",
  width    = 12,
  height   = 12,
  units    = "in"
)

```

```r
# |_ Table with values of the Log-Diff =========================================
log_diff_RF_Mean   <- log_diff_ts_types(tmpRF, tmp_is_stat, "Mean")
log_diff_RF_Median <- log_diff_ts_types(tmpRF, tmp_is_stat, "Median")

## \___ Export as TXT ----------------------------------------------------------
capture.output(
  log_diff_RF_Mean,
  file = paste0(tmpSavePathTables, "RF_log_diff_Mean_RMSE_", tmpVersionRF,".txt")
)
capture.output(
  log_diff_RF_Median,
  file = paste0(tmpSavePathTables, "RF_log_diff_Median_RMSE_", tmpVersionRF,".txt")
)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ============================================================================= #
# ============================================================================= #
####                              Results RIDGE                              ####
# ============================================================================= #
# ============================================================================= #
## \___ Import Data ------------------------------------------------------------
tmpVersionRIDGE <- "v2021082201"
tmpFileRIDGE    <- paste0(
  "results/results_cerqueira_RMSE_ridge_", tmpVersionRIDGE, ".rdata"
)
tmpRIDGE <- get(load(tmpFileRIDGE))

# |_ Plot the average ranks ====================================================
plot_RIDGE_avg_rank <- plot_avg_ranks_ts_types(
  results = tmpRIDGE,
  is_stat = tmp_is_stat,
  df.source = "Cerqueira et al. (2020)."
)
plot_RIDGE_avg_rank

## \___ Export as PNG ----------------------------------------------------------
```

```r
ggplot2::ggsave(
  filename = paste0("RIDGE_ranks_plot_RMSE_",tmpVersionRIDGE,".png"),
  path     = tmpSavePath,
  plot     = plot_RIDGE_avg_rank,
  device   = "png",
  type     = "cairo",
  width    = 8,
  height   = 12,
  units    = "in"
)

## \___ Export as EPS
    ------------------------------------------------------------
ggplot2::ggsave(
  filename = paste0("RIDGE_ranks_plot_RMSE_",tmpVersionRIDGE,".eps"),
  path     = tmpSavePath,
  plot     = plot_RIDGE_avg_rank,
  device   = "eps",
  width    = 8,
  height   = 12,
  units    = "in"
)

# |_ Plot the Log-Diff
    ============================================================
plot_RIDGE_log_diff <- plot_log_diff_ts_types(
  results = tmpRIDGE,
  is_stat = tmp_is_stat,
  df.source = "Cerqueira et al. (2020)."
)
plot_RIDGE_log_diff

## \___ Export as PNG
    ------------------------------------------------------------
ggplot2::ggsave(
  filename = paste0("RIDGE_log_diff_plot_RMSE_",tmpVersionRIDGE,".png")
    ,
  path     = tmpSavePath,
  plot     = plot_RIDGE_log_diff,
  device   = "png",
  type     = "cairo",
  width    = 12,
  height   = 12,
  units    = "in"
)

## \___ Export as EPS
    ------------------------------------------------------------
ggplot2::ggsave(
  filename = paste0("RIDGE_log_diff_plot_RMSE_",tmpVersionRIDGE,".eps")
    ,
  path     = tmpSavePath,
  plot     = plot_RIDGE_log_diff,
  device   = "eps",
```

```r
  width    = 12,
  height   = 12,
  units    = "in"
)

# |_ Table with values of the Log-Diff ==========================================
log_diff_RIDGE_Mean   <- log_diff_ts_types(tmpRIDGE, tmp_is_stat, "Mean")
log_diff_RIDGE_Median <- log_diff_ts_types(tmpRIDGE, tmp_is_stat, "Median")

## \____ Export as TXT ---------------------------------------------------------
capture.output(
  log_diff_RIDGE_Mean,
  file = paste0(
    tmpSavePathTables, "RIDGE_log_diff_Mean_RMSE_",tmpVersionRIDGE,".txt"
  )
)
capture.output(
  log_diff_RIDGE_Median,
  file = paste0(
    tmpSavePathTables, "RIDGE_log_diff_Median_RMSE_",tmpVersionRIDGE,".txt"
  )
)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ============================================================================ #
# ============================================================================ #
#####                          Stationary data - M4                        #####
# ============================================================================ #
# ============================================================================ #

# |_ Import Original Data set ==================================================
load("results/stationarity_tsdl_M4_v02.rdata")

# |_ Indicator if the series is stationary ====================================
tmp_is_stat_M4 <- is_stationary_2ensemble_M4_v02
```

```r
#table(is_stationary_2ensemble) # 97 stationary time series (out of 174, ~56%)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ============================================================================= #
# ============================================================================= #
####                             Results RBR - M4                            ####
# ============================================================================= #
# ============================================================================= #
## \___ Import Data -------------------------------------------------------------
tmpM4VersionRBR <- "v2021082201"
tmpM4FileRBR    <- paste0(
  "results/results_M4_RMSE_rbr_", tmpM4VersionRBR, ".rdata"
)
tmpRBR_M4 <- get(load(tmpM4FileRBR))

# |_ Plot the average ranks =====================================================
plot_M4_RMSE_rbr_avg_rank <- plot_avg_ranks_ts_types(
  results = tmpRBR_M4,
  is_stat = tmp_is_stat_M4,
  df.source = paste0("Sample from the M4 Competition data sets ",
                     "(Makridakis, Spiliotis and Assimakopoulos, 2020).")
)
plot_M4_RMSE_rbr_avg_rank

## \___ Export as PNG -----------------------------------------------------------
ggplot2::ggsave(
  filename = paste0("M4_RBR_ranks_plot_RMSE_",tmpM4VersionRBR,".png"),
  path     = tmpSavePath,
  plot     = plot_M4_RMSE_rbr_avg_rank,
  device   = "png",
  type     = "cairo",
  width    = 8,
  height   = 12,
  units    = "in"
)

## \___ Export as EPS -----------------------------------------------------------
```

```
ggplot2::ggsave(
  filename = paste0("M4_RBR_ranks_plot_RMSE_",tmpM4VersionRBR,".eps"),
  path     = tmpSavePath,
  plot     = plot_M4_RMSE_rbr_avg_rank,
  device   = "eps",
  width    = 8,
  height   = 12,
  units    = "in"
)

# |_ Plot the Log-Diff
    ========================================================
plot_M4_RBR_log_diff <- plot_log_diff_ts_types(
  results = tmpRBR_M4,
  is_stat = tmp_is_stat_M4,
  df.source = paste0("Sample from the M4 Competition data sets ",
                     "(Makridakis, Spiliotis and Assimakopoulos, 2020).
    ")

)
plot_M4_RBR_log_diff

## \___ Export as PNG
    ------------------------------------------------------------
ggplot2::ggsave(
  filename = paste0("M4_RBR_log_diff_plot_RMSE_",tmpM4VersionRBR,".png"
    ),
  path     = tmpSavePath,
  plot     = plot_M4_RBR_log_diff,
  device   = "png",
  type     = "cairo",
  width    = 12,
  height   = 12,
  units    = "in"
)

## \___ Export as EPS
    ------------------------------------------------------------
ggplot2::ggsave(
  filename = paste0("M4_RBR_log_diff_plot_RMSE_",tmpM4VersionRBR,".eps"
    ),
  path     = tmpSavePath,
  plot     = plot_M4_RBR_log_diff,
  device   = "eps",
  width    = 12,
  height   = 12,
  units    = "in"
)

# |_ Table with values of the Log-Diff
    ========================================
log_diff_M4_RMSE_rbr_Mean   <- log_diff_ts_types(tmpRBR_M4, tmp_is_stat
    _M4, "Mean")
```

```r
log_diff_M4_RMSE_rbr_Median <- log_diff_ts_types(tmpRBR_M4, tmp_is_stat_M4, "Median")

## \___ Export as TXT ----------------------------------------------------------
capture.output(
  log_diff_M4_RMSE_rbr_Mean,
  file = paste0(
    tmpSavePathTables, "M4_RBR_log_diff_Mean_RMSE_",tmpM4VersionRBR,".txt"
  )
)
capture.output(
  log_diff_M4_RMSE_rbr_Median,
  file = paste0(
    tmpSavePathTables, "M4_RBR_log_diff_Median_RMSE_",tmpM4VersionRBR,".txt"
  )
)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ============================================================================ #
# ============================================================================ #
####                            Results RF - M4                            ####
# ============================================================================ #
# ============================================================================ #
## \___ Import Data ------------------------------------------------------------
tmpM4VersionRF <- "v2021082201"
tmpM4FileRF    <- paste0(
  "results/results_M4_RMSE_rf_", tmpM4VersionRF, ".rdata"
)
tmpRF_M4 <- get(load(tmpM4FileRF))

# |_ Plot the average ranks ====================================================
plot_M4_RMSE_rf_avg_rank <- plot_avg_ranks_ts_types(
  results = tmpRF_M4,
  is_stat = tmp_is_stat_M4,
  df.source = paste0("Sample from the M4 Competition data sets ",
                     "(Makridakis, Spiliotis and Assimakopoulos, 2020).")
```

```
)
plot_M4_RMSE_rf_avg_rank

## \___ Export as PNG
    ------------------------------------------------------------
ggplot2::ggsave(
  filename = paste0("M4_RF_ranks_plot_RMSE_",tmpM4VersionRF,".png"),
  path     = tmpSavePath,
  plot     = plot_M4_RMSE_rf_avg_rank,
  device   = "png",
  type     = "cairo",
  width    = 8,
  height   = 12,
  units    = "in"
)

## \___ Export as EPS
    ------------------------------------------------------------
ggplot2::ggsave(
  filename = paste0("M4_RF_ranks_plot_RMSE_",tmpM4VersionRF,".eps"),
  path     = tmpSavePath,
  plot     = plot_M4_RMSE_rf_avg_rank,
  device   = "eps",
  width    = 8,
  height   = 12,
  units    = "in"
)

# |_ Plot the Log-Diff
    ============================================================
plot_M4_RF_log_diff <- plot_log_diff_ts_types(
  results = tmpRF_M4,
  is_stat = tmp_is_stat_M4,
  df.source = paste0("Sample from the M4 Competition data sets ",
                     "(Makridakis, Spiliotis and Assimakopoulos, 2020).
    ")

)
plot_M4_RF_log_diff

## \___ Export as PNG
    ------------------------------------------------------------
ggplot2::ggsave(
  filename = paste0("M4_RF_log_diff_plot_RMSE_",tmpM4VersionRF,".png"),
  path     = tmpSavePath,
  plot     = plot_M4_RF_log_diff,
  device   = "png",
  type     = "cairo",
  width    = 12,
  height   = 12,
  units    = "in"
)
```

```r
## \___ Export as EPS ----------------------------------------------------------
ggplot2::ggsave(
  filename = paste0("M4_RF_log_diff_plot_RMSE_",tmpM4VersionRF,".eps"),
  path     = tmpSavePath,
  plot     = plot_M4_RF_log_diff,
  device   = "eps",
  width    = 12,
  height   = 12,
  units    = "in"
)

# |_ Table with values of the Log-Diff =========================================
log_diff_M4_RMSE_rf_Mean   <- log_diff_ts_types(tmpRF_M4, tmp_is_stat_M4, "Mean")
log_diff_M4_RMSE_rf_Median <- log_diff_ts_types(tmpRF_M4, tmp_is_stat_M4, "Median")

## \___ Export as TXT ----------------------------------------------------------
capture.output(
  log_diff_M4_RMSE_rf_Mean,
  file = paste0(
    tmpSavePathTables, "M4_RF_log_diff_Mean_RMSE_",tmpM4VersionRF,".txt"
  )
)
capture.output(
  log_diff_M4_RMSE_rf_Median,
  file = paste0(
    tmpSavePathTables, "M4_RF_log_diff_Median_RMSE_",tmpM4VersionRF,".txt"
  )
)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ============================================================================ #
# ============================================================================ #
####                           Results RIDGE - M4                           ####
# ============================================================================ #
# ============================================================================ #
```

```
## \___ Import Data
    ------------------------------------------------------------
tmpM4VersionRIDGE <- "v2021082201"
tmpM4FileRIDGE    <- paste0(
  "results/results_M4_RMSE_ridge_", tmpM4VersionRIDGE, ".rdata"
)
tmpRIDGE_M4 <- get(load(tmpM4FileRIDGE))

# |_ Plot the average ranks
    ====================================================
plot_M4_RMSE_ridge_avg_rank <- plot_avg_ranks_ts_types(
  results = tmpRIDGE_M4,
  is_stat = tmp_is_stat_M4,
  df.source = paste0("Sample from the M4 Competition data sets ",
                     "(Makridakis, Spiliotis and Assimakopoulos, 2020).
    ")

)
plot_M4_RMSE_ridge_avg_rank

## \___ Export as PNG
    ------------------------------------------------------------
ggplot2::ggsave(
  filename = paste0("M4_RIDGE_ranks_plot_RMSE_",tmpM4VersionRIDGE,".png
    "),
  path     = tmpSavePath,
  plot     = plot_M4_RMSE_ridge_avg_rank,
  device   = "png",
  type     = "cairo",
  width    = 8,
  height   = 12,
  units    = "in"
)

## \___ Export as EPS
    ------------------------------------------------------------
ggplot2::ggsave(
  filename = paste0("M4_RIDGE_ranks_plot_RMSE_",tmpM4VersionRIDGE,".eps
    "),
  path     = tmpSavePath,
  plot     = plot_M4_RMSE_ridge_avg_rank,
  device   = "eps",
  width    = 8,
  height   = 12,
  units    = "in"
)

# |_ Plot the Log-Diff
    ============================================================
plot_M4_RIDGE_log_diff <- plot_log_diff_ts_types(
  results = tmpRIDGE_M4,
  is_stat = tmp_is_stat_M4,
  df.source = paste0("Sample from the M4 Competition data sets ",
```

```
872                       "(Makridakis, Spiliotis and Assimakopoulos, 2020).
    ")

874 )
plot_M4_RIDGE_log_diff
876
## \___ Export as PNG
    ------------------------------------------------------------
878 ggplot2::ggsave(
  filename = paste0("M4_RIDGE_log_diff_plot_RMSE_",tmpM4VersionRIDGE,".
    png"),
880   path     = tmpSavePath,
  plot     = plot_M4_RIDGE_log_diff,
882   device   = "png",
  type     = "cairo",
884   width    = 12,
  height   = 12,
886   units    = "in"
)
888
## \___ Export as EPS
    ------------------------------------------------------------
890 ggplot2::ggsave(
  filename = paste0("M4_RIDGE_log_diff_plot_RMSE_",tmpM4VersionRIDGE,".
    eps"),
892   path     = tmpSavePath,
  plot     = plot_M4_RIDGE_log_diff,
894   device   = "eps",
  width    = 12,
896   height   = 12,
  units    = "in"
898 )

900 # |_ Table with values of the Log-Diff
    ========================================
log_diff_M4_RMSE_ridge_Mean   <- log_diff_ts_types(
902   tmpRIDGE_M4, tmp_is_stat_M4, "Mean"
)
904
log_diff_M4_RMSE_ridge_Median <- log_diff_ts_types(
906   tmpRIDGE_M4, tmp_is_stat_M4, "Median"
)
908
## \___ Export as TXT
    ------------------------------------------------------------
910 capture.output(
  log_diff_M4_RMSE_ridge_Mean,
912   file = paste0(
    tmpSavePathTables, "M4_RIDGE_log_diff_Mean_RMSE_",tmpM4VersionRIDGE
    ,".txt"
914   )
)
916 capture.output(
  log_diff_M4_RMSE_ridge_Median,
```

```
  file = paste0(
    tmpSavePathTables, "M4_RIDGE_log_diff_Median_RMSE_",
    tmpM4VersionRIDGE,".txt"
  )
)
```

C.7 Create Plots - Simulated Data

code/rw_analysis_TS3_v2021_07_09_01.R

```
# ######################################################################## #
#' Evaluating the performance of estimation methods
#
#' This is an extension of the code from ts3, Torgo, and Mozetic.
#' "Evaluating time series forecasting models: an empirical study
#' on performance estimation methods".
#' In: Machine Learning (2020) 109:1997-2028
#'
#' Modified by: Gustavo Varela-Alvarenga
#' Date:        07/09/2021
# ######################################################################## #
# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# > Packages ============================================================
# library(forecast)
# library(tsensembler)
# library(ranger)
# load code for `avg_rank_plot`
source("src/plots_gus.r")
#library(scmamp)
# --- #

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
## >  Path to save plots to ----
tmpSavePath <- "results/plots"
tmpSavePathTables <- "results/tables/"

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# > Helpers                                                            ####
# |_ Helper Function: get_ranks_p_holdout ==============================
# Gets final estimation errors, and ranks
get_ranks_p_holdout_sim <- function(final_results_data){
  # --- #
  # get estimation errors
  err_estimation <- lapply(
    X = final_results_data,
    function(X) tryCatch(X$err_estimation, error =function(e) {NULL})
  )

  err_estimation <- err_estimation[!sapply(err_estimation, is.null)]
  # --- #
  # create df with final estimation errors
  fr <- do.call(rbind, err_estimation)
  fr <- as.data.frame(fr)
  rownames(fr) <- NULL

  colnames(fr) <-
    c("p-holdout",      # <---- new method
      "cp-holdout",     # <---- new method
      "cep-holdout",    # <---- new method
      "CV", "CV-Bl", "CV-Mod","CV-hvBl",
```

```
      "Preq-Bls", "Preq-Sld-Bls",
      "Preq-Bls-Gap","Holdout", "Rep-Holdout",
      # "Preq-Slide","Preq-Grow" #the order in the original is switched
      "Preq-Grow", "Preq-Slide"
    )
  # --- #
  # get ranks for each estimation procedure
  fr_abs <- abs(fr)
  fr_abs_rank <- apply(fr_abs, 1, rank)

  # --- #
  # return df with final estimation errors,
  # and another one with the ranks
  return(list(fr = fr, fr_abs_rank = fr_abs_rank))
}

# |_ Helper Function: plot_avg_ranks_ts_types ==========================
plot_avg_ranks_ts_types <- function(results_ts3, results_ts4){

  # ---- #
  # get ranks
  ranksTS3 <- get_ranks_p_holdout_sim(results_ts3)
  ranksTS4 <- get_ranks_p_holdout_sim(results_ts4)

  # ---- #
  # plot ranks
  ## function 'avg_rank_plot' comes from
  ## source("../src/plots_gus.r")
  ## I've changed the original 'plots.r' to add a green color to my
  ## schemes' results' bar plot.
  pRanksTS3 <- avg_rank_plot(
    avg  = rowMeans(ranksTS3$fr_abs_rank),
    sdev = apply(ranksTS3$fr_abs_rank,1, sd)
  )
  pRanksTS4 <- avg_rank_plot(
    avg  = rowMeans(ranksTS4$fr_abs_rank),
    sdev = apply(ranksTS4$fr_abs_rank,1, sd)
  )

  finalPlotRanks <- ggpubr::ggarrange(
    plotlist = list(pRanksTS3, pRanksTS4),
    ncol     = 1,
    nrow     = 2,
    labels   = c(
      "S3: SARIMA(12,0,0)(1,0,0)12", "S4: SARIMA(12,0,0)(1,1,0)12"
    )
  )

  # ---- #
  # return plot with annotations
  ggpubr::annotate_figure(
    finalPlotRanks,
    bottom = ggpubr::text_grob(
      label = "Plot by: Varela-Alvarenga and Kedem (2021).",
```

```
      hjust = 1,
      x     = 1,
      face  = "italic",
      size  = 10
    )
  )
}

# ------------------------------------------------------------------ #
# helper function - get data and plot ----
plot_log_diff_ts_types <- function(results_ts3, results_ts4){

  # ---- #
  # get ranks
  ranksTS3 <- get_ranks_p_holdout_sim(results_ts3)
  ranksTS4 <- get_ranks_p_holdout_sim(results_ts4)
  # ---- #
  # plot log diff

  ## function 'percdiff_plot_log' comes from
  ## source("../src/plots_gus.r")
  pLogDiffTS3           <- percdiff_plot_log(ranksTS3$fr)
  pLogDiffTS4           <- percdiff_plot_log(ranksTS4$fr)

  finalPlotLogDiff <- ggpubr::ggarrange(
    plotlist = list(pLogDiffTS3, pLogDiffTS4),
    ncol     = 1,
    nrow     = 2,
    labels   = list("S3","S4")
  )
  # ---- #

  # ---- #
  # return plot with annotations
  ggpubr::annotate_figure(
    finalPlotLogDiff,
    bottom = ggpubr::text_grob(
      label = "Plot by: Varela-Alvarenga and Kedem (2021).",
      hjust = 1,
      x     = 1,
      face  = "italic",
      size  = 10
    ),
    left = ggpubr::text_grob(
      label = "Percentual difference to true loss",
      face  = "bold",
      size  = 12,
      rot   = 90
    )
  )
}

# ------------------------------------------------------------------ #
# helper function - returns log diff values ----
```

```r

log_diff_ts_types <- function(results_ts3, results_ts4,
                              statistic = c(
                                "Mean", "Median", "Std.Dev.", "IQR"
                              )
                      ){

  # get ranks
  ranksTS3 <- get_ranks_p_holdout_sim(results_ts3)
  ranksTS4 <- get_ranks_p_holdout_sim(results_ts4)
  # ---- #

  # ---- #
  # calculate log diff
  log_trans <- function(x) sign(x) * log(abs(x) + 1)

  rTS3     <- reshape2::melt(ranksTS3$fr, id.vars = NULL)
  rTS3$log <- log_trans(rTS3$value)
  rTS4     <- reshape2::melt(ranksTS4$fr, id.vars = NULL)
  rTS4$log <- log_trans(rTS4$value)
  # ---- #
  # calculate summary by scheme
  summaryTS3 <- aggregate(
    x   = rTS3$log,
    by  = list(scheme = rTS3$variable),
    FUN = summary
  )
  summaryTS4 <- aggregate(
    x   = rTS4$log,
    by  = list(scheme = rTS4$variable),
    FUN = summary
  )
  # ---- #
  # calculate std dev by scheme
  sdTS3 <- aggregate(
    x   = rTS3$log,
    by  = list(scheme = rTS3$variable),
    FUN = sd
  )
  sdTS4 <- aggregate(
    x   = rTS4$log,
    by  = list(scheme = rTS4$variable),
    FUN = sd
  )
  # ---- #
  # calculate IQR by scheme
  iqrTS3 <- aggregate(
    x   = rTS3$log,
    by  = list(scheme = rTS3$variable),
    FUN = IQR
  )
  iqrTS4 <- aggregate(
    x   = rTS4$log,
    by  = list(scheme = rTS4$variable),
```

```
    FUN = IQR
  )
  # ---- #
  # Get final summary table (with std dev and IQR)
  summaryTS3 <- cbind.data.frame(
    summaryTS3,
    "Std.Dev." = sdTS3$x,
    "IQR"      = iqrTS3$x
  )
  summaryTS4 <- cbind.data.frame(
    summaryTS4,
    "Std.Dev." = sdTS4$x,
    "IQR"      = iqrTS4$x
  )
  # ---- #
  # Order final summary table by statistic
  summaryTS3 <- summaryTS3[order(abs(summaryTS3$x[, statistic])), ]

  summaryTS4 <- summaryTS4[order(abs(summaryTS4$x[, statistic])), ]
  # ---- #
  # return final list
  list(
    TS3 = summaryTS3,
    TS4 = summaryTS4
  )
}

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ========================================================================== #
# ========================================================================== #
####                            Results RBR                               ####
# ========================================================================== #
# ========================================================================== #
## \___ Import Data -------------------------------------------------------
tmpVersionRBR <- "v2021070901"

tmpFileRBR_TS3    <- paste0(
  "results/results_ts3_rbr_", tmpVersionRBR, ".rdata"
)
tmpRBR_TS3 <- get(load(tmpFileRBR_TS3))

tmpFileRBR_TS4    <- paste0(
  "results/results_ts4_rbr_", tmpVersionRBR, ".rdata"
)
tmpRBR_TS4 <- get(load(tmpFileRBR_TS4))

# |_ Plot the average ranks ================================================
plot_RBR_avg_rank <- plot_avg_ranks_ts_types(
  results_ts3 = tmpRBR_TS3,
  results_ts4 = tmpRBR_TS4
)
plot_RBR_avg_rank

## \___ Export as PNG -----------------------------------------------------
```

```r
ggplot2::ggsave(
  filename = paste0("SIM_RBR_ranks_plot_MSE_",tmpVersionRBR,".png"),
  path     = tmpSavePath,
  plot     = plot_RBR_avg_rank,
  device   = "png",
  type     = "cairo",
  width    = 8,
  height   = 12,
  units    = "in"
)

## \___ Export as EPS ------------------------------------------------
ggplot2::ggsave(
  filename = paste0("SIM_RBR_ranks_plot_MSE_",tmpVersionRBR,".eps"),
  path     = tmpSavePath,
  plot     = plot_RBR_avg_rank,
  device   = "eps",
  width    = 8,
  height   = 12,
  units    = "in"
)

# |_ Plot the Log-Diff ==============================================
plot_RBR_log_diff <- plot_log_diff_ts_types(
  results_ts3 = tmpRBR_TS3,
  results_ts4 = tmpRBR_TS4
)
plot_RBR_log_diff

## \___ Export as PNG ------------------------------------------------
ggplot2::ggsave(
  filename = paste0("SIM_RBR_log_diff_plot_MSE_",tmpVersionRBR,".png"),
  path     = tmpSavePath,
  plot     = plot_RBR_log_diff,
  device   = "png",
  type     = "cairo",
  width    = 12,
  height   = 12,
  units    = "in"
)

## \___ Export as EPS ------------------------------------------------
ggplot2::ggsave(
  filename = paste0("SIM_RBR_log_diff_plot_MSE_",tmpVersionRBR,".eps"),
  path     = tmpSavePath,
  plot     = plot_RBR_log_diff,
  device   = "eps",
  width    = 12,
  height   = 12,
  units    = "in"
)

# |_ Table with values of the Log-Diff ================================
log_diff_RBR_Mean   <-
```

```r
  log_diff_ts_types(tmpRBR_TS3, tmpRBR_TS4, "Mean")

log_diff_RBR_Median <-
  log_diff_ts_types(tmpRBR_TS3, tmpRBR_TS4, "Median")

## \___ Export as TXT ----------------------------------------------
capture.output(
  log_diff_RBR_Mean,
  file = paste0(tmpSavePathTables,
                "SIM_log_diff_Mean_MSE_",tmpVersionRBR,".txt")
)
capture.output(
  log_diff_RBR_Median,
  file = paste0(tmpSavePathTables,
                "SIM_RBR_log_diff_Median_MSE_",tmpVersionRBR,".txt")
)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ========================================================================= #
# ========================================================================= #
####                              Results RF                             ####
# ========================================================================= #
# ========================================================================= #
## \___ Import Data ------------------------------------------------
tmpVersionRF <- "v2021070901"

tmpFileRF_TS3    <- paste0(
  "results/results_ts3_rf_", tmpVersionRF, ".rdata"
)
tmpRF_TS3 <- get(load(tmpFileRF_TS3))

tmpFileRF_TS4    <- paste0(
  "results/results_ts4_rf_", tmpVersionRF, ".rdata"
)
tmpRF_TS4 <- get(load(tmpFileRF_TS4))

# |_ Plot the average ranks ===================================================
plot_RF_avg_rank <- plot_avg_ranks_ts_types(
  results_ts3 = tmpRF_TS3,
  results_ts4 = tmpRF_TS4
)
plot_RF_avg_rank

## \___ Export as PNG ----------------------------------------------
ggplot2::ggsave(
  filename = paste0("SIM_RF_ranks_plot_MSE_",tmpVersionRF,".png"),
  path     = tmpSavePath,
  plot     = plot_RF_avg_rank,
  device   = "png",
  type     = "cairo",
  width    = 8,
  height   = 12,
  units    = "in"
)
```

```
375
## \___ Export as EPS ------------------------------------------------
377 ggplot2::ggsave(
  filename = paste0("SIM_RF_ranks_plot_MSE_",tmpVersionRF,".eps"),
379   path     = tmpSavePath,
  plot     = plot_RF_avg_rank,
381   device   = "eps",
  width    = 8,
383   height   = 12,
  units    = "in"
385 )

387 # |_ Plot the Log-Diff ============================================
plot_RF_log_diff <- plot_log_diff_ts_types(
389   results_ts3 = tmpRF_TS3,
  results_ts4 = tmpRF_TS4
391 )
plot_RF_log_diff
393
## \___ Export as PNG ------------------------------------------------
395 ggplot2::ggsave(
  filename = paste0("SIM_RF_log_diff_plot_MSE_",tmpVersionRF,".png"),
397   path     = tmpSavePath,
  plot     = plot_RF_log_diff,
399   device   = "png",
  type     = "cairo",
401   width    = 12,
  height   = 12,
403   units    = "in"
)
405
## \___ Export as EPS ------------------------------------------------
407 ggplot2::ggsave(
  filename = paste0("SIM_RF_log_diff_plot_MSE_",tmpVersionRF,".eps"),
409   path     = tmpSavePath,
  plot     = plot_RF_log_diff,
411   device   = "eps",
  width    = 12,
413   height   = 12,
  units    = "in"
415 )

417 # |_ Table with values of the Log-Diff ============================
log_diff_RF_Mean   <- log_diff_ts_types(tmpRF_TS3, tmpRF_TS4, "Mean")
419 log_diff_RF_Median <- log_diff_ts_types(tmpRF_TS3, tmpRF_TS4, "Median")

421 ## \___ Export as TXT ------------------------------------------------
capture.output(
423   log_diff_RF_Mean,
  file = paste0(tmpSavePathTables,
425                 "SIM_RF_log_diff_Mean_MSE_",tmpVersionRF,".txt")
)
427 capture.output(
  log_diff_RF_Median,
```

```
  file = paste0(tmpSavePathTables,
                "SIM_RF_log_diff_Median_MSE_",tmpVersionRF,".txt")
)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ========================================================================== #
# ========================================================================== #
####                              Results RIDGE                           ####
# ========================================================================== #
# ========================================================================== #
## \___ Import Data ------------------------------------------------------
tmpVersionRIDGE <- "v2021070901"

tmpFileRIDGE_TS3    <- paste0(
  "results/results_ts3_ridge_", tmpVersionRIDGE, ".rdata"
)
tmpRIDGE_TS3 <- get(load(tmpFileRIDGE_TS3))

tmpFileRIDGE_TS4    <- paste0(
  "results/results_ts4_ridge_", tmpVersionRIDGE, ".rdata"
)
tmpRIDGE_TS4 <- get(load(tmpFileRIDGE_TS4))

# |_ Plot the average ranks ================================================
plot_RIDGE_avg_rank <- plot_avg_ranks_ts_types(
  results_ts3 = tmpRIDGE_TS3,
  results_ts4 = tmpRIDGE_TS4
)
plot_RIDGE_avg_rank

## \___ Export as PNG ----------------------------------------------------
ggplot2::ggsave(
  filename = paste0(
    "SIM_RIDGE_ranks_plot_MSE_",tmpVersionRIDGE,".png"
  ),
  path     = tmpSavePath,
  plot     = plot_RIDGE_avg_rank,
  device   = "png",
  type     = "cairo",
  width    = 8,
  height   = 12,
  units    = "in"
)

## \___ Export as EPS ----------------------------------------------------
ggplot2::ggsave(
  filename = paste0(
    "SIM_RIDGE_ranks_plot_MSE_",tmpVersionRIDGE,".eps"
  ),
  path     = tmpSavePath,
  plot     = plot_RIDGE_avg_rank,
  device   = "eps",
  width    = 8,
  height   = 12,
```

```
  units    = "in"
)

# |_ Plot the Log-Diff ================================================
plot_RIDGE_log_diff <- plot_log_diff_ts_types(
  results_ts3 = tmpRIDGE_TS3,
  results_ts4 = tmpRIDGE_TS4
)
plot_RIDGE_log_diff

## \___ Export as PNG ------------------------------------------------
ggplot2::ggsave(
  filename = paste0(
    "SIM_RIDGE_log_diff_plot_MSE_",tmpVersionRIDGE,".png"
  ),
  path     = tmpSavePath,
  plot     = plot_RIDGE_log_diff,
  device   = "png",
  type     = "cairo",
  width    = 12,
  height   = 12,
  units    = "in"
)

## \___ Export as EPS ------------------------------------------------
ggplot2::ggsave(
  filename = paste0(
    "SIM_RIDGE_log_diff_plot_MSE_",tmpVersionRIDGE,".eps"
  ),
  path     = tmpSavePath,
  plot     = plot_RIDGE_log_diff,
  device   = "eps",
  width    = 12,
  height   = 12,
  units    = "in"
)

# |_ Table with values of the Log-Diff ================================
log_diff_RIDGE_Mean   <- log_diff_ts_types(
  tmpRIDGE_TS3, tmpRIDGE_TS4, "Mean"
)
log_diff_RIDGE_Median <- log_diff_ts_types(
  tmpRIDGE_TS3, tmpRIDGE_TS4, "Median"
)

## \___ Export as TXT ------------------------------------------------
capture.output(
  log_diff_RIDGE_Mean,
  file = paste0(
    tmpSavePathTables, "SIM_RIDGE_log_diff_Mean_MSE_",tmpVersionRIDGE,
    ".txt"
  )
)
capture.output(
```

```
  log_diff_RIDGE_Median,
  file = paste0(
    tmpSavePathTables, "SIM_RIDGE_log_diff_Median_MSE_",
    tmpVersionRIDGE,
    ".txt"
  )
)
```

C.8 Create Tables - Cerqueira and M4

code/hypothesis_tests_SignedRank_v2021071502.R

```
# ######################################################################## #
#' Evaluating the performance of estimation methods
#
#' This is an extension of the code from Cerqueira, Torgo, and Mozetic.
#' "Evaluating time series forecasting models: an empirical study
#' on performance estimation methods".
#' In: Machine Learning (2020) 109:1997-2028
#'
#' Modified by: Gustavo Varela-Alvarenga
#' Date:        07/15/2021
# ######################################################################## #
# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# > Packages ============================================================
#install.packages("devtools", dependencies = TRUE)
#devtools::install_github('alanarnholt/BSDA')
#install.packages("xtable")

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
## >  Path to save tables to ----
tmpSavePathTables <- "results/tables/"

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# > Helpers                                                            ####
# |_ Helper Function: get_ranks_p_holdout ================================
# Gets final estimation errors, and ranks
get_ranks_p_holdout <- function(final_results_data){
  # --- #
  # get estimation errors
  err_estimation <- lapply(
    X = final_results_data,
    function(X) tryCatch(X$err_estimation, error =function(e) {NULL})
  )

  err_estimation <- err_estimation[!sapply(err_estimation, is.null)]
  # --- #
  # create df with final estimation errors
  fr <- do.call(rbind, err_estimation)
  fr <- as.data.frame(fr)
  rownames(fr) <- NULL

  colnames(fr) <-
    c("p-holdout",      # <---- new method
      "cp-holdout",     # <---- new method
      "cep-holdout",    # <---- new method
      "CV", "CV-Bl", "CV-Mod","CV-hvBl",
      "Preq-Bls", "Preq-Sld-Bls",
      "Preq-Bls-Gap","Holdout", "Rep-Holdout",
      # "Preq-Slide","Preq-Grow" #the order in the original is switched
      "Preq-Grow", "Preq-Slide"
    )
```

```r
  # --- #
  # get ranks for each estimation procedure
  fr_abs <- abs(fr)
  fr_abs_rank <- apply(fr_abs, 1, rank)

  # --- #
  # return df with final estimation errors,
  # and another one with the ranks
  return(list(fr = fr, fr_abs_rank = fr_abs_rank))
}

# ------------------------------------------------------------------------ #
# helper function - returns log diff values ----

hypothesis_tests_schemes <- function(results, is_stat){

  # break data into stationary data or not (or both)
  tmpAll           <- results
  tmpStationary    <- results[is_stat]
  tmpNonStationary <- results[!is_stat]

  # ---- #
  # get ranks
  ranksAll           <- get_ranks_p_holdout(tmpAll)
  ranksStationary    <- get_ranks_p_holdout(tmpStationary)
  ranksNonStationary <- get_ranks_p_holdout(tmpNonStationary)

  # ---- #
  # calculate log diff
  log_trans <- function(x) sign(x) * log(abs(x) + 1)

  rAll               <- reshape2::melt(ranksAll$fr, id.vars = NULL)
  rAll$log           <- log_trans(rAll$value)
  rStationary        <- reshape2::melt(
    ranksStationary$fr, id.vars = NULL
  )
  rStationary$log    <- log_trans(rStationary$value)
  rNonStationary     <- reshape2::melt(
    ranksNonStationary$fr, id.vars = NULL
  )
  rNonStationary$log <- log_trans(rNonStationary$value)

  # ---- #
  # calculate summary by scheme
  summaryAll <- aggregate(
    x   = rAll$log,
    by  = list(scheme = rAll$variable),
    FUN = summary
  )
  summaryAll[,-1] <- round(summaryAll[,-1], 4)

  summaryStationary <- aggregate(
    x   = rStationary$log,
    by  = list(scheme = rStationary$variable),
```

```
  FUN = summary
)
summaryStationary[,-1] <- round(summaryStationary[,-1], 4)

summaryNonStationary <- aggregate(
  x   = rNonStationary$log,
  by  = list(scheme = rNonStationary$variable),
  FUN = summary
)
summaryNonStationary[,-1] <- round(summaryNonStationary[,-1], 4)

# ---- #
# calculate p-value by scheme

tmpSchemes <- as.character(unique(rAll$variable))

pvalueAll <- lapply(
  seq_along(tmpSchemes),
  function(X){
    tmpScheme <- tmpSchemes[X]
    tmpDF     <- rAll[rAll$variable == tmpScheme, ]
    tmpTest   <- wilcox.test(
      tmpDF$log, mu = 0, alternative = "two.sided"
    )
    list(
      "scheme"     = tmpScheme,
      "WMW_pvalue" = round(tmpTest$p.value, 4)
    )
  }
)
pvalueAll <- do.call(rbind, pvalueAll)

pvalueStationary <- lapply(
  seq_along(tmpSchemes),
  function(X){
    tmpScheme <- tmpSchemes[X]
    tmpDF     <- rStationary[rStationary$variable == tmpScheme, ]
    tmpTest   <- wilcox.test(
      tmpDF$log, mu = 0, alternative = "two.sided"
      )
    list(
      "scheme"     = tmpScheme,
      "WMW_pvalue" = round(tmpTest$p.value, 4)
    )
  }
)
pvalueStationary <- do.call(rbind, pvalueStationary)

pvalueNonStationary <- lapply(
  seq_along(tmpSchemes),
  function(X){
    tmpScheme <- tmpSchemes[X]
    tmpDF     <-
      rNonStationary[rNonStationary$variable == tmpScheme, ]
```

```
159
      tmpTest   <-
161     wilcox.test(tmpDF$log, mu = 0, alternative = "two.sided")

163   list(
        "scheme"      = tmpScheme,
165     "WMW_pvalue" = round(tmpTest$p.value, 4)
      )
167   }
  )
169 pvalueNonStationary <- do.call(rbind, pvalueNonStationary)

171 # ---- #
  # calculate p-value by scheme
173 pvalueAllSign <- lapply(
    seq_along(tmpSchemes),
175   function(X){
      tmpScheme <- tmpSchemes[X]
177   tmpDF     <- rAll[rAll$variable == tmpScheme, ]

      tmpTest   <-
179     BSDA::SIGN.test(tmpDF$log, mu = 0, alternative = "two.sided")
181
      c("scheme" = tmpScheme, "p_value" = round(tmpTest$p.value, 4))
183   }
  )
185 pvalueAllSign <- do.call(rbind, pvalueAllSign)

187 pvalueStationarySign <- lapply(
    seq_along(tmpSchemes),
189   function(X){
      tmpScheme <- tmpSchemes[X]
191   tmpDF     <- rStationary[rStationary$variable == tmpScheme, ]

      tmpTest   <-
193     BSDA::SIGN.test(tmpDF$log, mu = 0, alternative = "two.sided")
195
      c("scheme" = tmpScheme, "p_value" = round(tmpTest$p.value, 4))
197   }
  )
199 pvalueStationarySign <- do.call(rbind, pvalueStationarySign)

201 pvalueNonStationarySign <- lapply(
    seq_along(tmpSchemes),
203   function(X){
      tmpScheme <- tmpSchemes[X]
205
      tmpDF     <-
207     rNonStationary[rNonStationary$variable == tmpScheme, ]

      tmpTest   <-
209     BSDA::SIGN.test(tmpDF$log, mu = 0, alternative = "two.sided")
211
      c("scheme" = tmpScheme, "p_value" = round(tmpTest$p.value, 4))
```

```
    }
  )
  pvalueNonStationarySign <- do.call(rbind, pvalueNonStationarySign)

  # ---- #
  # Final Sets
  finalAll        <- merge(summaryAll, pvalueAllSign, by="scheme")

  finalStationary <-
    merge(summaryStationary, pvalueStationarySign, by="scheme")

  finalNonStationary <-
    merge(summaryNonStationary, pvalueNonStationarySign, by="scheme")


  # ---- #
  # return final list
  list(
    All           = finalAll,
    Stationary    = finalStationary,
    NonStationary = finalNonStationary
  )
}

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####

# ========================================================================= #
# ========================================================================= #
#####                            Stationary data                       #####
# ========================================================================= #
# ========================================================================= #

# |_ Import Original Data set ============================================
load("../stationarity_tsdl.rdata")

# |_ Indicator if the series is stationary ===============================
tmp_is_stat <- is_stationary_2ensemble

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ========================================================================= #
# ========================================================================= #
####                            Results RBR                           ####
# ========================================================================= #
# ========================================================================= #
## \___ Import Data ----------------------------------------------------

tmpVersionRBR_MSE  <- "v2021070901"
tmpVersionRBR_MASE <- "v2021071201"

tmpFileRBR_MSE    <- paste0(
  "results/results_cerqueira_rbr_", tmpVersionRBR_MSE, ".rdata"
)
tmpRBR_MSE <- get(load(tmpFileRBR_MSE))
```

```r
tmpFileRBR_MASE    <- paste0(
  "results/results_cerqueira_MASE_rbr_", tmpVersionRBR_MASE, ".rdata"
)
tmpRBR_MASE <- get(load(tmpFileRBR_MASE))

# |_ Table with values of the Log-Diff ================================
test_RBR_MSE   <- hypothesis_tests_schemes(
  results = tmpRBR_MSE,
  is_stat = tmp_is_stat
)

test_RBR_MASE   <- hypothesis_tests_schemes(
  results = tmpRBR_MASE,
  is_stat = tmp_is_stat
)

## \___ Export as Latex Table ---------------------------------------
print(
  xtable::xtable(
    x = t(t(test_RBR_MSE$All)),
    type = "latex",
    label = "tab:RBR:cerqueira:bias:all",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      " relative to the true loss for each validation scheme applied ",
      "to all 174 real-world time series ",
      "using the ",
      "RBR ",
      "learning algorithm and the ",
      "MSE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "RBR_test_MSE_All_",tmpVersionRBR_MSE,".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_RBR_MSE$Stationary)),
    type = "latex",
    label = "tab:RBR:cerqueira:bias:stationary",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      " relative to the true loss for each validation scheme applied ",
      "to the 97 stationary time series ",
      "using the ",
      "RBR ",
      "learning algorithm and the ",
      "MSE ",
      "as error measure."
```

```
    )
  ),
  file = paste0(
    tmpSavePathTables, "RBR_test_MSE_Stationary_",tmpVersionRBR_MSE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_RBR_MSE$NonStationary)),
    type = "latex",
    label = "tab:RBR:cerqueira:bias:nonstationary",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      " relative to the true loss for each validation scheme applied ",
      "to the 77 non-stationary time series ",
      "using the ",
      "RBR ",
      "learning algorithm and the ",
      "MSE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "RBR_test_MSE_NonStationary_",tmpVersionRBR_MSE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

# ---- #
# MASE ----

print(
  xtable::xtable(
    x = t(t(test_RBR_MASE$All)),
    type = "latex",
    label = "tab:RBR:cerqueira:bias:all:mase",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to all 174 real-world time series ",
      "using the ",
      "RBR ",
      "learning algorithm and the ",
      "MASE ",
      "as error measure."
    )
  ),
  file = paste0(
```

```
    tmpSavePathTables, "RBR_test_MASE_All_",tmpVersionRBR_MASE,".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_RBR_MASE$Stationary)),
    type = "latex",
    label = "tab:RBR:cerqueira:bias:stationary:mase",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the 97 stationary time series ",
      "using the ",
      "RBR ",
      "learning algorithm and the ",
      "MASE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "RBR_test_MASE_Stationary_",tmpVersionRBR_MASE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_RBR_MASE$NonStationary)),
    type = "latex",
    label = "tab:RBR:cerqueira:bias:nonstationary:mase",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the 77 non-stationary time series ",
      "using the ",
      "RBR ",
      "learning algorithm and the ",
      "MASE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "RBR_test_MASE_NonStationary_",
    tmpVersionRBR_MASE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)
```

```
429
# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
431 # ========================================================================== #
# ========================================================================== #
433 ####                            Results RF                                ####
# ========================================================================== #
435 # ========================================================================== #
## \___ Import Data ------------------------------------------------
437
tmpVersionRF_MSE  <- "v2021070901"
439 tmpVersionRF_MASE <- "v2021071201"

441 tmpFileRF_MSE    <- paste0(
  "results/results_cerqueira_rf_", tmpVersionRF_MSE, ".rdata"
443 )
tmpRF_MSE <- get(load(tmpFileRF_MSE))
445
tmpFileRF_MASE    <- paste0(
447   "results/results_cerqueira_MASE_rf_", tmpVersionRF_MASE, ".rdata"
)
449 tmpRF_MASE <- get(load(tmpFileRF_MASE))

451 # |_ Table with values of the Log-Diff ================================
test_RF_MSE   <- hypothesis_tests_schemes(
453   results = tmpRF_MSE,
  is_stat = tmp_is_stat
455 )

457 test_RF_MASE   <- hypothesis_tests_schemes(
  results = tmpRF_MASE,
459   is_stat = tmp_is_stat
)
461
## \___ Export as Latex Table --------------------------------------
463 print(
  xtable::xtable(
465     x = t(t(test_RF_MSE$All)),
    type = "latex",
467     label = "tab:RF:cerqueira:bias:all",
    caption = paste0(
469       "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
471       "to all 174 real-world time series ",
      "using the ",
473       "RF ",
      "learning algorithm and the ",
475       "MSE ",
      "as error measure."
477     )
  ),
479   file = paste0(
    tmpSavePathTables, "RF_test_MSE_All_", tmpVersionRF_MSE, ".txt"
481   ),
  booktabs = TRUE,
```

```
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_RF_MSE$Stationary)),
    type = "latex",
    label = "tab:RF:cerqueira:bias:stationary",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the 97 stationary time series ",
      "using the ",
      "RF ",
      "learning algorithm and the ",
      "MSE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "RF_test_MSE_Stationary_", tmpVersionRF_MSE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_RF_MSE$NonStationary)),
    type = "latex",
    label = "tab:RF:cerqueira:bias:nonstationary",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the 77 non-stationary time series ",
      "using the ",
      "RF ",
      "learning algorithm and the ",
      "MSE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "RF_test_MSE_NonStationary_", tmpVersionRF_MSE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

# ---- #
# MASE ----
```

```
print(
  xtable::xtable(
    x = t(t(test_RF_MASE$All)),
    type = "latex",
    label = "tab:RF:cerqueira:bias:all:mase",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to all 174 real-world time series ",
      "using the ",
      "RF ",
      "learning algorithm and the ",
      "MASE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "RF_test_MASE_All_",tmpVersionRF_MASE, ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_RF_MASE$Stationary)),
    type = "latex",
    label = "tab:RF:cerqueira:bias:stationary:mase",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the 97 stationary time series ",
      "using the ",
      "RF ",
      "learning algorithm and the ",
      "MASE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "RF_test_MASE_Stationary_",tmpVersionRF_MASE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_RF_MASE$NonStationary)),
    type = "latex",
    label = "tab:RF:cerqueira:bias:nonstationary:mase",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
```

```
      "relative to the true loss for each validation scheme applied ",
      "to the 77 non-stationary time series ",
      "using the ",
      "RF ",
      "learning algorithm and the ",
      "MASE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "RF_test_MASE_NonStationary_",tmpVersionRF_MASE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ========================================================================== #
# ========================================================================== #
####                              Results GLM                               ####
# ========================================================================== #
# ========================================================================== #
## \___ Import Data ------------------------------------------------------

tmpVersionGLM_MSE  <- "v2021070901"
tmpVersionGLM_MASE <- "v2021071201"

tmpFileGLM_MSE    <- paste0(
  "results/results_cerqueira_ridge_", tmpVersionGLM_MSE, ".rdata"
)
tmpGLM_MSE <- get(load(tmpFileGLM_MSE))

tmpFileGLM_MASE    <- paste0(
  "results/results_cerqueira_MASE_ridge_", tmpVersionGLM_MASE, ".rdata"
)
tmpGLM_MASE <- get(load(tmpFileGLM_MASE))

# |_ Table with values of the Log-Diff ================================
test_GLM_MSE   <- hypothesis_tests_schemes(
  results = tmpGLM_MSE,
  is_stat = tmp_is_stat
)

test_GLM_MASE   <- hypothesis_tests_schemes(
  results = tmpGLM_MASE,
  is_stat = tmp_is_stat
)

## \___ Export as Latex Table --------------------------------------
print(
  xtable::xtable(
    x = t(t(test_GLM_MSE$All)),
    type = "latex",
```

```
    label = "tab:GLM:cerqueira:bias:all",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to all 174 real-world time series ",
      "using the ",
      "GLM-Ridge ",
      "learning algorithm and the ",
      "MSE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "GLM_test_MSE_All_",tmpVersionGLM_MSE,".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_GLM_MSE$Stationary)),
    type = "latex",
    label = "tab:GLM:cerqueira:bias:stationary",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the 97 stationary time series ",
      "using the ",
      "GLM-Ridge ",
      "learning algorithm and the ",
      "MSE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "GLM_test_MSE_Stationary_",tmpVersionGLM_MSE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_GLM_MSE$NonStationary)),
    type = "latex",
    label = "tab:GLM:cerqueira:bias:nonstationary",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the 77 non-stationary time series ",
      "using the ",
      "GLM-Ridge ",
```

```
      "learning algorithm and the ",
      "MSE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "GLM_test_MSE_NonStationary_",tmpVersionGLM_MSE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

# ---- #
# MASE ----

print(
  xtable::xtable(
    x = t(t(test_GLM_MASE$All)),
    type = "latex",
    label = "tab:GLM:cerqueira:bias:all:mase",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to all 174 real-world time series ",
      "using the ",
      "GLM-Ridge ",
      "learning algorithm and the ",
      "MASE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "GLM_test_MASE_All_",tmpVersionGLM_MASE,".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_GLM_MASE$Stationary)),
    type = "latex",
    label = "tab:GLM:cerqueira:bias:stationary:mase",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the 97 stationary time series ",
      "using the ",
      "GLM-Ridge ",
      "learning algorithm and the ",
      "MASE ",
      "as error measure."
    )
```

```
  ),
  file = paste0(
    tmpSavePathTables, "GLM_test_MASE_Stationary_",tmpVersionGLM_MASE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_GLM_MASE$NonStationary)),
    type = "latex",
    label = "tab:GLM:cerqueira:bias:nonstationary:mase",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the 77 non-stationary time series ",
      "using the ",
      "GLM-Ridge ",
      "learning algorithm and the ",
      "MASE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "GLM_test_MASE_NonStationary_",
    tmpVersionGLM_MASE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ========================================================================= #
# ========================================================================= #
#####                          Stationary data - M4                   #####
# ========================================================================= #
# ========================================================================= #

# |_ Import Original Data set ============================================
load("results/stationarity_tsdl_M4_v02.rdata")

# |_ Indicator if the series is stationary ===============================
tmp_is_stat_M4 <- is_stationary_2ensemble_M4_v02

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ========================================================================= #
# ========================================================================= #
####                          Results M4-RBR                           ####
# ========================================================================= #
# ========================================================================= #
## \___ Import Data ---------------------------------------------------
```

```
807
tmpVersionM4_RBR_MSE  <- "v2021070901"
809 tmpVersionM4_RBR_MASE <- "v2021071201"

811 tmpFileM4_RBR_MSE    <- paste0(
  "results/results_M4_rbr_", tmpVersionM4_RBR_MSE, ".rdata"
813 )
tmpM4_RBR_MSE <- get(load(tmpFileM4_RBR_MSE))
815
tmpFileM4_RBR_MASE    <- paste0(
817   "results/results_M4_MASE_rbr_", tmpVersionM4_RBR_MASE, ".rdata"
)
819 tmpM4_RBR_MASE <- get(load(tmpFileM4_RBR_MASE))

821 # |_ Table with values of the Log-Diff ================================
test_M4_RBR_MSE   <- hypothesis_tests_schemes(
823   results = tmpM4_RBR_MSE,
  is_stat = tmp_is_stat
825 )

827 test_M4_RBR_MASE   <- hypothesis_tests_schemes(
  results = tmpM4_RBR_MASE,
829   is_stat = tmp_is_stat
)
831
## \___ Export as Latex Table ----------------------------------------
833 print(
  xtable::xtable(
835     x = t(t(test_M4_RBR_MSE$All)),
    type = "latex",
837     label = "tab:RBR:M4:bias:all",
    caption = paste0(
839       "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
841       "to the time series from the M4 Competition sample ",
      "using the ",
843       "RBR ",
      "learning algorithm and the ",
845       "MSE ",
      "as error measure."
847     )
  ),
849   file = paste0(
    tmpSavePathTables, "M4_RBR_test_MSE_All_",tmpVersionM4_RBR_MSE,
851     ".txt"
  ),
853   booktabs = TRUE,
  include.rownames=FALSE
855 )

857 print(
  xtable::xtable(
859     x = t(t(test_M4_RBR_MSE$Stationary)),
    type = "latex",
```

```
    label = "tab:RBR:M4:bias:stationary",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the stationary time series from the M4 Competition sample ",
      "using the ",
      "RBR ",
      "learning algorithm and the ",
      "MSE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "M4_RBR_test_MSE_Stationary_",
    tmpVersionM4_RBR_MSE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_M4_RBR_MSE$NonStationary)),
    type = "latex",
    label = "tab:RBR:M4:bias:nonstationary",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the non-stationary time series from the M4 Competition ",
      "sample using the ",
      "RBR ",
      "learning algorithm and the ",
      "MSE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "M4_RBR_test_MSE_NonStationary_",
    tmpVersionM4_RBR_MSE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

# ---- #
# MASE ----

print(
  xtable::xtable(
    x = t(t(test_M4_RBR_MASE$All)),
    type = "latex",
    label = "tab:RBR:M4:bias:all:mase",
```

```
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the time series from the M4 Competition sample ",
      "using the ",
      "RBR ",
      "learning algorithm and the ",
      "MASE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "M4_RBR_test_MASE_All_",tmpVersionM4_RBR_MASE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_M4_RBR_MASE$Stationary)),
    type = "latex",
    label = "tab:RBR:M4:bias:stationary:mase",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the stationary time series from the M4 Competition sample ",
      "using the ",
      "RBR ",
      "learning algorithm and the ",
      "MASE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "M4_RBR_test_MASE_Stationary_",
    tmpVersionM4_RBR_MASE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_M4_RBR_MASE$NonStationary)),
    type = "latex",
    label = "tab:RBR:M4:bias:nonstationary:mase",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the non-stationary time series from the M4 Competition ",
      "sample using the ",
```

```
      "RBR ",
      "learning algorithm and the ",
      "MASE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "M4_RBR_test_MASE_NonStationary_",
    tmpVersionM4_RBR_MASE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ========================================================================== #
# ========================================================================== #
####                            Results M4-RF                             ####
# ========================================================================== #
# ========================================================================== #
## \___ Import Data ------------------------------------------------------

tmpVersionM4_RF_MSE  <- "v2021070901"
tmpVersionM4_RF_MASE <- "v2021071201"

tmpFileM4_RF_MSE    <- paste0(
  "results/results_M4_rf_", tmpVersionM4_RF_MSE, ".rdata"
)
tmpM4_RF_MSE <- get(load(tmpFileM4_RF_MSE))

tmpFileM4_RF_MASE    <- paste0(
  "results/results_M4_MASE_rf_", tmpVersionM4_RF_MASE, ".rdata"
)
tmpM4_RF_MASE <- get(load(tmpFileM4_RF_MASE))

# |_ Table with values of the Log-Diff ================================
test_M4_RF_MSE   <- hypothesis_tests_schemes(
  results = tmpM4_RF_MSE,
  is_stat = tmp_is_stat
)

test_M4_RF_MASE   <- hypothesis_tests_schemes(
  results = tmpM4_RF_MASE,
  is_stat = tmp_is_stat
)

## \___ Export as Latex Table ---------------------------------------------
print(
  xtable::xtable(
    x = t(t(test_M4_RF_MSE$All)),
    type = "latex",
    label = "tab:RF:M4:bias:all",
    caption = paste0(
```

```
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the time series from the M4 Competition sample ",
      "using the ",
      "RF ",
      "learning algorithm and the ",
      "MSE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "M4_RF_test_MSE_All_",tmpVersionM4_RF_MSE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_M4_RF_MSE$Stationary)),
    type = "latex",
    label = "tab:RF:M4:bias:stationary",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the stationary time series from the M4 Competition sample ",
      "using the ",
      "RF ",
      "learning algorithm and the ",
      "MSE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "M4_RF_test_MSE_Stationary_",
    tmpVersionM4_RF_MSE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_M4_RF_MSE$NonStationary)),
    type = "latex",
    label = "tab:RF:M4:bias:nonstationary",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the non-stationary time series from the M4 Competition ",
      "sample using the ",
      "RF ",
```

```
      "learning algorithm and the ",
      "MSE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "M4_RF_test_MSE_NonStationary_",
    tmpVersionM4_RF_MSE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

# ---- #
# MASE ----

print(
  xtable::xtable(
    x = t(t(test_M4_RF_MASE$All)),
    type = "latex",
    label = "tab:RF:M4:bias:all:mase",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the time series from the M4 Competition sample ",
      "using the ",
      "RF ",
      "learning algorithm and the ",
      "MASE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "M4_RF_test_MASE_All_",tmpVersionM4_RF_MASE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_M4_RF_MASE$Stationary)),
    type = "latex",
    label = "tab:RF:M4:bias:stationary:mase",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the stationary time series from the M4 Competition sample ",
      "using the ",
      "RF ",
      "learning algorithm and the ",
      "MASE ",
```

```
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "M4_RF_test_MASE_Stationary_",
    tmpVersionM4_RF_MASE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_M4_RF_MASE$NonStationary)),
    type = "latex",
    label = "tab:RF:M4:bias:nonstationary:mase",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the non-stationary time series from the M4 Competition ",
      "sample using the ",
      "RF ",
      "learning algorithm and the ",
      "MASE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "M4_RF_test_MASE_NonStationary_",
    tmpVersionM4_RF_MASE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ========================================================================== #
# ========================================================================== #
####                            Results M4-GLM                            ####
# ========================================================================== #
# ========================================================================== #
## \___ Import Data ------------------------------------------------------

tmpVersionM4_GLM_MSE  <- "v2021070901"
tmpVersionM4_GLM_MASE <- "v2021071201"

tmpFileM4_GLM_MSE    <- paste0(
  "results/results_M4_ridge_", tmpVersionM4_GLM_MSE, ".rdata"
)
tmpM4_GLM_MSE <- get(load(tmpFileM4_GLM_MSE))

tmpFileM4_GLM_MASE    <- paste0(
```

```
  "results/results_M4_MASE_ridge_", tmpVersionM4_GLM_MASE, ".rdata"
)
tmpM4_GLM_MASE <- get(load(tmpFileM4_GLM_MASE))

# |_ Table with values of the Log-Diff ================================
test_M4_GLM_MSE   <- hypothesis_tests_schemes(
  results = tmpM4_GLM_MSE,
  is_stat = tmp_is_stat
)

test_M4_GLM_MASE   <- hypothesis_tests_schemes(
  results = tmpM4_GLM_MASE,
  is_stat = tmp_is_stat
)

## \___ Export as Latex Table ----------------------------------------
print(
  xtable::xtable(
    x = t(t(test_M4_GLM_MSE$All)),
    type = "latex",
    label = "tab:GLM:M4:bias:all",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the time series from the M4 Competition sample ",
      "using the ",
      "GLM-RIDGE ",
      "learning algorithm and the ",
      "MSE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "M4_GLM_test_MSE_All_",tmpVersionM4_GLM_MSE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_M4_GLM_MSE$Stationary)),
    type = "latex",
    label = "tab:GLM:M4:bias:stationary",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the stationary time series from the M4 Competition sample ",
      "using the ",
      "GLM-RIDGE ",
      "learning algorithm and the ",
      "MSE ",
      "as error measure."
```

```
    )
  ),
  file = paste0(
    tmpSavePathTables, "M4_GLM_test_MSE_Stationary_",
    tmpVersionM4_GLM_MSE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_M4_GLM_MSE$NonStationary)),
    type = "latex",
    label = "tab:GLM:M4:bias:nonstationary",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the non-stationary time series from the M4 Competition ",
      "sample using the ",
      "GLM-RIDGE ",
      "learning algorithm and the ",
      "MSE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "M4_GLM_test_MSE_NonStationary_",
    tmpVersionM4_GLM_MSE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

# ---- #
# MASE ----

print(
  xtable::xtable(
    x = t(t(test_M4_GLM_MASE$All)),
    type = "latex",
    label = "tab:GLM:M4:bias:all:mase",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the time series from the M4 Competition sample ",
      "using the ",
      "GLM-RIDGE ",
      "learning algorithm and the ",
      "MASE ",
      "as error measure."
    )
```

```
  ),
  file = paste0(
    tmpSavePathTables, "M4_GLM_test_MASE_All_",tmpVersionM4_GLM_MASE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_M4_GLM_MASE$Stationary)),
    type = "latex",
    label = "tab:GLM:M4:bias:stationary:mase",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the stationary time series from the M4 Competition sample ",
      "using the ",
      "GLM-RIDGE ",
      "learning algorithm and the ",
      "MASE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "M4_GLM_test_MASE_Stationary_",
    tmpVersionM4_GLM_MASE,
    ".txt"
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)

print(
  xtable::xtable(
    x = t(t(test_M4_GLM_MASE$NonStationary)),
    type = "latex",
    label = "tab:GLM:M4:bias:nonstationary:mase",
    caption = paste0(
      "Summary of the log percentage difference of the estimated loss",
      "relative to the true loss for each validation scheme applied ",
      "to the non-stationary time series from the M4 Competition ",
      "sample using the ",
      "GLM-RIDGE ",
      "learning algorithm and the ",
      "MASE ",
      "as error measure."
    )
  ),
  file = paste0(
    tmpSavePathTables, "M4_GLM_test_MASE_NonStationary_",
    tmpVersionM4_GLM_MASE,
    ".txt"
```

```
  ),
  booktabs = TRUE,
  include.rownames=FALSE
)
```

C.9 Create Bayesian Plots

code/bayesian_plots_v2021071601.R

```r
# ######################################################################## #
#' Evaluating the performance of estimation methods
#
#' This is an extension of the code from Cerqueira, Torgo, and Mozetic.
#' "Evaluating time series forecasting models: an empirical study
#' on performance estimation methods".
#' In: Machine Learning (2020) 109:1997-2028
#'
#' Modified by: Gustavo Varela-Alvarenga
#' Date:        07/16/2021
# ######################################################################## #
# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# > Packages ===========================================================
# install.packages("remotes")
# remotes::install_github("b0rxa/scmamp")
#
# install.packages("ggplot2")
#
# install.packages("Cairo")
#
# install.packages("beepr")

library("scmamp")
library("ggplot2")
library("Cairo")

# load code for `proportion_plot`
source("src/plots_v2021071301.r")

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
## >  Path to save plots to ----
tmpSavePath <- "results/plots"

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# > Helpers                                                             ####
# |_ Helper Function: get_proportion_plot ==============================
get_proportion_plot <- function(final_results_data, baseline){

  # --- #
  # get estimation errors
  err_estimation <- lapply(
    X = final_results_data,
    function(X) tryCatch(X$err_estimation, error =function(e) {NULL})
  )

  err_estimation <- err_estimation[!sapply(err_estimation, is.null)]
  # --- #
  # create df with final estimation errors
  fr <- do.call(rbind, err_estimation)
  fr <- as.data.frame(fr)
```

```
  rownames(fr) <- NULL

  colnames(fr) <-
    c("p-Holdout",      # <---- new method
      "cp-Holdout",     # <---- new method
      "cep-Holdout",    # <---- new method
      "CV", "CV-Bl", "CV-Mod","CV-hvBl",
      "Preq-Bls", "Preq-Sld-Bls",
      "Preq-Bls-Gap","Holdout", "Rep-Holdout",
      # "Preq-Slide","Preq-Grow" #the order in the original is switched
      "Preq-Grow", "Preq-Slide"
    )
  # --- #
  # get ranks for each estimation procedure
  fr_abs <- abs(fr)

  # --- #
  cID <- which(colnames(fr_abs) %in% baseline)

  PerfDiff <- lapply(as.data.frame(fr_abs),
                     function(x) x-fr_abs[,cID,drop=T])

  PerfDiff <- as.data.frame(PerfDiff[-cID])

  rope <- 2.5
  baout <- lapply(
    PerfDiff,
    function(u) {
      bSignedRankTest(u, rope=c(-rope,rope))$posterior.probabilities
    }
  )

  baout <- lapply(baout,unlist)
  baout <- do.call(rbind, baout)
  rownames(baout) <- gsub("\\.","-",rownames(baout))
  colnames(baout) <- c("probLeft","probRope","probRight")
  proportion_plot(baout, baseline) + scale_fill_brewer(palette="Set2")
}

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####

# ========================================================================= #
# ========================================================================= #
#####                        Stationary data                          #####
# ========================================================================= #
# ========================================================================= #

# |_ Import Original Data set ============================================
load("../stationarity_tsdl.rdata")

# |_ Indicator if the series is stationary ===============================
is_stat <- is_stationary_2ensemble

# 97 stationary time series (out of 174, ~56%)
```

```r
#table(is_stationary_2ensemble)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ========================================================================== #
# ========================================================================== #
####                        Results RBR                                   ####
# ========================================================================== #
# ========================================================================== #

## \___ MSE ------------------------------------------------------------

tmpVersionRBR_MSE  <- "v2021070901"

tmpFileRBR_MSE    <- paste0(
  "results/results_cerqueira_rbr_", tmpVersionRBR_MSE, ".rdata"
)
tmpRBR_MSE <- get(load(tmpFileRBR_MSE))

# ---- #
# Stationary ----
ggsave(
  filename = paste0(
    tmpSavePath, "/RBR_bayes_MSE_Stationary_", tmpVersionRBR_MSE,
    ".png"
  ),
  plot     = get_proportion_plot(tmpRBR_MSE[is_stat], "cep-Holdout"),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

# ---- #
# NonStationary ----
ggsave(
  filename = paste0(
    tmpSavePath, "/RBR_bayes_MSE_NonStationary_", tmpVersionRBR_MSE,
    ".png"
  ),
  plot     = get_proportion_plot(tmpRBR_MSE[!is_stat], "cep-Holdout"),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

## \___ MASE -----------------------------------------------------------

tmpVersionRBR_MASE <- "v2021071201"

tmpFileRBR_MASE    <- paste0(
  "results/results_cerqueira_MASE_rbr_", tmpVersionRBR_MASE, ".rdata"
)
tmpRBR_MASE <- get(load(tmpFileRBR_MASE))
```

```r

# ---- #
# Stationary ----
ggsave(
  filename = paste0(
    tmpSavePath, "/RBR_bayes_MASE_Stationary_", tmpVersionRBR_MASE,
    ".png"
  ),
  plot     = get_proportion_plot(tmpRBR_MASE[is_stat], "cep-Holdout"),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

# ---- #
# NonStationary ----
  ggsave(
    filename = paste0(
      tmpSavePath,
      "/RBR_bayes_MASE_NonStationary_",
      tmpVersionRBR_MASE,
      ".png"
    ),
    plot     = get_proportion_plot(
      tmpRBR_MASE[!is_stat], "cep-Holdout"
    ),
    device   = "png",
    type     = "cairo",
    width    = 6,
    height   = 4
  )

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ========================================================================== #
# ========================================================================== #
####                          Results RF                                  ####
# ========================================================================== #
# ========================================================================== #

## \___ MSE ------------------------------------------------------------

tmpVersionRF_MSE  <- "v2021070901"

tmpFileRF_MSE    <- paste0(
  "results/results_cerqueira_rf_", tmpVersionRF_MSE, ".rdata"
)
tmpRF_MSE <- get(load(tmpFileRF_MSE))

# ---- #
# Stationary ----
ggsave(
  filename = paste0(
    tmpSavePath, "/RF_bayes_MSE_Stationary_", tmpVersionRF_MSE, ".png"
```

```
  ),
  plot     = get_proportion_plot(tmpRF_MSE[is_stat], "cep-Holdout"),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

# ---- #
# NonStationary ----
ggsave(
  filename = paste0(
    tmpSavePath, "/RF_bayes_MSE_NonStationary_", tmpVersionRF_MSE,
    ".png"
  ),
  plot     = get_proportion_plot(tmpRF_MSE[!is_stat], "cep-Holdout"),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

## \___ MASE -------------------------------------------------------------

tmpVersionRF_MASE <- "v2021071201"

tmpFileRF_MASE    <- paste0(
  "results/results_cerqueira_MASE_rf_", tmpVersionRF_MASE, ".rdata"
)
tmpRF_MASE <- get(load(tmpFileRF_MASE))

# ---- #
# Stationary ----
ggsave(
  filename = paste0(
    tmpSavePath, "/RF_bayes_MASE_Stationary_", tmpVersionRF_MASE,
    ".png"
  ),
  plot     = get_proportion_plot(tmpRF_MASE[is_stat], "cep-Holdout"),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

# ---- #
# NonStationary ----
ggsave(
  filename = paste0(
    tmpSavePath, "/RF_bayes_MASE_NonStationary_", tmpVersionRF_MASE,
    ".png"
  ),
  plot     = get_proportion_plot(tmpRF_MASE[!is_stat], "cep-Holdout"),
  device   = "png",
```

```
  type     = "cairo",
  width    = 6,
  height   = 4
)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ========================================================================== #
# ========================================================================== #
####                           Results GLM                                  ####
# ========================================================================== #
# ========================================================================== #

## \___ MSE -------------------------------------------------------------

tmpVersionGLM_MSE  <- "v2021070901"

tmpFileGLM_MSE    <- paste0(
  "results/results_cerqueira_ridge_", tmpVersionGLM_MSE, ".rdata"
)
tmpGLM_MSE <- get(load(tmpFileGLM_MSE))

# ---- #
# Stationary ----
ggsave(
  filename = paste0(
    tmpSavePath, "/RIDGE_bayes_MSE_Stationary_", tmpVersionGLM_MSE,
    ".png"
  ),
  plot     = get_proportion_plot(tmpGLM_MSE[is_stat], "cep-Holdout"),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

# ---- #
# NonStationary ----
ggsave(
  filename = paste0(
    tmpSavePath, "/RIDGE_bayes_MSE_NonStationary_", tmpVersionGLM_MSE,
    ".png"
  ),
  plot     = get_proportion_plot(tmpGLM_MSE[!is_stat], "cep-Holdout"),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

## \___ MASE ------------------------------------------------------------

tmpVersionGLM_MASE <- "v2021071201"

tmpFileGLM_MASE    <- paste0(
```

```
  "results/results_cerqueira_MASE_ridge_", tmpVersionGLM_MASE, ".rdata"
)
tmpGLM_MASE <- get(load(tmpFileGLM_MASE))

# ---- #
# Stationary ----
ggsave(
  filename = paste0(
    tmpSavePath, "/RIDGE_bayes_MASE_Stationary_", tmpVersionGLM_MASE,
    ".png"
  ),
  plot     = get_proportion_plot(tmpGLM_MASE[is_stat], "cep-Holdout"),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

# ---- #
# NonStationary ----
ggsave(
  filename = paste0(
    tmpSavePath, "/RIDGE_bayes_MASE_NonStationary_",
    tmpVersionGLM_MASE,
    ".png"
  ),
  plot     = get_proportion_plot(tmpGLM_MASE[!is_stat], "cep-Holdout"),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ========================================================================= #
# ========================================================================= #
#####                         Stationary data - M4                      #####
# ========================================================================= #
# ========================================================================= #

# |_ Import Original Data set ============================================
load("results/stationarity_tsdl_M4_v02.rdata")

# |_ Indicator if the series is stationary ===============================
is_stat_M4 <- is_stationary_2ensemble_M4_v02

# 91 stationary time series (out of 1000, ~0.091%)
#table(is_stationary_2ensemble_M4_v02)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ========================================================================= #
# ========================================================================= #
####                          Results M4-RBR                           ####
# ========================================================================= #
```

```r
# ========================================================================== #

## \___ MSE ------------------------------------------------------------

tmpVersionM4_RBR_MSE  <- "v2021070901"

tmpFileM4_RBR_MSE    <- paste0(
  "results/results_M4_rbr_", tmpVersionM4_RBR_MSE, ".rdata"
)
tmpM4_RBR_MSE <- get(load(tmpFileM4_RBR_MSE))

# ---- #
# Stationary ----
ggsave(
  filename = paste0(
    tmpSavePath, "/M4_RBR_bayes_MSE_Stationary_", tmpVersionM4_RBR_MSE,
    ".png"
  ),
  plot     = get_proportion_plot(
    tmpM4_RBR_MSE[is_stat_M4], "cep-Holdout"
  ),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

# ---- #
# NonStationary ----
ggsave(
  filename = paste0(
    tmpSavePath, "/M4_RBR_bayes_MSE_NonStationary_",
    tmpVersionM4_RBR_MSE,
    ".png"
  ),
  plot     = get_proportion_plot(
    tmpM4_RBR_MSE[!is_stat_M4], "cep-Holdout"
  ),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

## \___ MASE
    ------------------------------------------------------------------

tmpVersionM4_RBR_MASE <- "v2021071201"

tmpFileM4_RBR_MASE    <- paste0(
  "results/results_M4_MASE_rbr_", tmpVersionM4_RBR_MASE, ".rdata"
)
tmpM4_RBR_MASE <- get(load(tmpFileM4_RBR_MASE))
```

```r
# ---- #
# Stationary ----
ggsave(
  filename = paste0(
    tmpSavePath, "/M4_RBR_bayes_MASE_Stationary_",
    tmpVersionM4_RBR_MASE,
    ".png"
  ),
  plot     = get_proportion_plot(
    tmpM4_RBR_MASE[is_stat_M4], "cep-Holdout"
  ),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

# ---- #
# NonStationary ----
ggsave(
  filename = paste0(
    tmpSavePath, "/M4_RBR_bayes_MASE_NonStationary_",
    tmpVersionM4_RBR_MASE,
    ".png"
  ),
  plot     = get_proportion_plot(
    tmpM4_RBR_MASE[!is_stat_M4], "cep-Holdout"
  ),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ========================================================================== #
# ========================================================================== #
####                            Results M4-RF                             ####
# ========================================================================== #
# ========================================================================== #

## \___ MSE ------------------------------------------------------------

tmpVersionM4_RF_MSE  <- "v2021070901"

tmpFileM4_RF_MSE    <- paste0(
  "results/results_M4_rf_", tmpVersionM4_RF_MSE, ".rdata"
)
tmpM4_RF_MSE <- get(load(tmpFileM4_RF_MSE))

# ---- #
# Stationary ----
ggsave(
  filename = paste0(
```

```
    tmpSavePath, "/M4_RF_bayes_MSE_Stationary_", tmpVersionM4_RF_MSE,
    ".png"
  ),
  plot     = get_proportion_plot(
    tmpM4_RF_MSE[is_stat_M4], "cep-Holdout"
  ),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

# ---- #
# NonStationary ----
ggsave(
  filename = paste0(
    tmpSavePath, "/M4_RF_bayes_MSE_NonStationary_",
    tmpVersionM4_RF_MSE,
    ".png"
  ),
  plot     = get_proportion_plot(
    tmpM4_RF_MSE[!is_stat_M4], "cep-Holdout"
  ),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

## \___ MASE -------------------------------------------------------------

tmpVersionM4_RF_MASE <- "v2021071201"

tmpFileM4_RF_MASE    <- paste0(
  "results/results_M4_MASE_rf_", tmpVersionM4_RF_MASE, ".rdata"
)
tmpM4_RF_MASE <- get(load(tmpFileM4_RF_MASE))

# ---- #
# Stationary ----
ggsave(
  filename = paste0(
    tmpSavePath, "/M4_RF_bayes_MASE_Stationary_", tmpVersionM4_RF_MASE,
    ".png"
  ),
  plot     = get_proportion_plot(
    tmpM4_RF_MASE[is_stat_M4], "cep-Holdout"
  ),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)
```

```r
# ---- #
# NonStationary ----
ggsave(
  filename = paste0(
    tmpSavePath, "/M4_RF_bayes_MASE_NonStationary_",
    tmpVersionM4_RF_MASE,
    ".png"
  ),
  plot     = get_proportion_plot(
    tmpM4_RF_MASE[!is_stat_M4], "cep-Holdout"
  ),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ========================================================================== #
# ========================================================================== #
####                              Results M4-GLM                            ####
# ========================================================================== #
# ========================================================================== #

## \___ MSE ---------------------------------------------------------------

tmpVersionM4_RIDGE_MSE  <- "v2021070901"

tmpFileM4_RIDGE_MSE    <- paste0(
  "results/results_M4_ridge_", tmpVersionM4_RIDGE_MSE, ".rdata"
)
tmpM4_RIDGE_MSE <- get(load(tmpFileM4_RIDGE_MSE))

# ---- #
# Stationary ----
ggsave(
  filename = paste0(
    tmpSavePath, "/M4_RIDGE_bayes_MSE_Stationary_",
    tmpVersionM4_RIDGE_MSE,
    ".png"
  ),
  plot     = get_proportion_plot(
    tmpM4_RIDGE_MSE[is_stat_M4], "cep-Holdout"
  ),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

# ---- #
# NonStationary ----
ggsave(
  filename = paste0(
```

```
    tmpSavePath, "/M4_RIDGE_bayes_MSE_NonStationary_",
    tmpVersionM4_RIDGE_MSE,
    ".png"
  ),
  plot     = get_proportion_plot(
    tmpM4_RIDGE_MSE[!is_stat_M4], "cep-Holdout"
  ),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

## \___ MASE
    ----------------------------------------------------------------------

tmpVersionM4_RIDGE_MASE <- "v2021071201"

tmpFileM4_RIDGE_MASE    <- paste0(
  "results/results_M4_MASE_rf_", tmpVersionM4_RIDGE_MASE, ".rdata"
)
tmpM4_RIDGE_MASE <- get(load(tmpFileM4_RIDGE_MASE))

# ---- #
# Stationary ----
ggsave(
  filename = paste0(
    tmpSavePath, "/M4_RIDGE_bayes_MASE_Stationary_",
    tmpVersionM4_RIDGE_MASE,
    ".png"
  ),
  plot     = get_proportion_plot(
    tmpM4_RIDGE_MASE[is_stat_M4], "cep-Holdout"
  ),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

# ---- #
# NonStationary ----
ggsave(
  filename = paste0(
    tmpSavePath, "/M4_RIDGE_bayes_MASE_NonStationary_",
    tmpVersionM4_RIDGE_MASE,
    ".png"
  ),
  plot     = get_proportion_plot(
    tmpM4_RIDGE_MASE[!is_stat_M4], "cep-Holdout"
  ),
  device   = "png",
  type     = "cairo",
  width    = 6,
```

```
  height    = 4
)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ========================================================================== #
# ========================================================================== #
####                              Results SIM-RBR                           ####
# ========================================================================== #
# ========================================================================== #

## \___ MSE ------------------------------------------------------------------

# ---- #
# S3 ----
tmpVersionS3_RBR_MSE  <- "v2021070901"

tmpFileS3_RBR_MSE    <- paste0(
  "results/results_ts3_rbr_", tmpVersionS3_RBR_MSE, ".rdata"
)
tmpS3_RBR_MSE <- get(load(tmpFileS3_RBR_MSE))

ggsave(
  filename = paste0(
    tmpSavePath, "/SIM_RBR_bayes_MSE_S3_", tmpVersionS3_RBR_MSE, ".png"
  ),
  plot     = get_proportion_plot(tmpS3_RBR_MSE, "cep-Holdout"),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)


# ---- #
# S4 ----
tmpVersionS4_RBR_MSE  <- "v2021070901"

tmpFileS4_RBR_MSE    <- paste0(
  "results/results_ts4_rbr_", tmpVersionS4_RBR_MSE, ".rdata"
)
tmpS4_RBR_MSE <- get(load(tmpFileS4_RBR_MSE))

ggsave(
  filename = paste0(
    tmpSavePath, "/SIM_RBR_bayes_MSE_S4_", tmpVersionS4_RBR_MSE, ".png"
  ),
  plot     = get_proportion_plot(tmpS4_RBR_MSE, "cep-Holdout"),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)


```

```r
## \___ MASE ------------------------------------------------------------

# ---- #
# S3 ----
tmpVersionS3_RBR_MASE <- "v2021071201"
tmpFileS3_RBR_MASE    <- paste0(
  "results/results_TS3_MASE_rbr_", tmpVersionS3_RBR_MASE, ".rdata"
)
tmpS3_RBR_MASE <- get(load(tmpFileS3_RBR_MASE))

ggsave(
  filename = paste0(
    tmpSavePath, "/SIM_RBR_bayes_MASE_S3_", tmpVersionS3_RBR_MASE,
    ".png"
  ),
  plot     = get_proportion_plot(tmpS3_RBR_MASE, "cep-Holdout"),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

# ---- #
# S4 ----
tmpVersionS4_RBR_MASE <- "v2021071201"
tmpFileS4_RBR_MASE    <- paste0(
  "results/results_TS4_MASE_rbr_", tmpVersionS4_RBR_MASE, ".rdata"
)
tmpS4_RBR_MASE <- get(load(tmpFileS4_RBR_MASE))

ggsave(
  filename = paste0(
    tmpSavePath, "/SIM_RBR_bayes_MASE_S4_", tmpVersionS4_RBR_MASE,
    ".png"
  ),
  plot     = get_proportion_plot(tmpS4_RBR_MASE, "cep-Holdout"),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ========================================================================= #
# ========================================================================= #
####                            Results SIM-RF                           ####
# ========================================================================= #
# ========================================================================= #

## \___ MSE ------------------------------------------------------------

# ---- #
# S3 ----
tmpVersionS3_RF_MSE  <- "v2021070901"
```

```r

tmpFileS3_RF_MSE    <- paste0(
  "results/results_ts3_rf_", tmpVersionS3_RF_MSE, ".rdata"
)
tmpS3_RF_MSE <- get(load(tmpFileS3_RF_MSE))

ggsave(
  filename = paste0(
    tmpSavePath, "/SIM_RF_bayes_MSE_S3_", tmpVersionS3_RF_MSE, ".png"
  ),
  plot     = get_proportion_plot(tmpS3_RF_MSE, "cep-Holdout"),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

# ---- #
# S4 ----
tmpVersionS4_RF_MSE  <- "v2021070901"

tmpFileS4_RF_MSE    <- paste0(
  "results/results_ts4_rf_", tmpVersionS4_RF_MSE, ".rdata"
)
tmpS4_RF_MSE <- get(load(tmpFileS4_RF_MSE))

ggsave(
  filename = paste0(
    tmpSavePath, "/SIM_RF_bayes_MSE_S4_", tmpVersionS4_RF_MSE, ".png"
  ),
  plot     = get_proportion_plot(tmpS4_RF_MSE, "cep-Holdout"),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)
## \___ MASE -------------------------------------------------------------

# ---- #
# S3 ----
tmpVersionS3_RF_MASE <- "v2021071201"
tmpFileS3_RF_MASE    <- paste0(
  "results/results_TS3_MASE_rf_", tmpVersionS3_RF_MASE, ".rdata"
)
tmpS3_RF_MASE <- get(load(tmpFileS3_RF_MASE))

ggsave(
  filename = paste0(
    tmpSavePath, "/SIM_RF_bayes_MASE_S3_", tmpVersionS3_RF_MASE, ".png"
  ),
  plot     = get_proportion_plot(tmpS3_RF_MASE, "cep-Holdout"),
  device   = "png",
  type     = "cairo",
  width    = 6,
```

```
  height    = 4
)

# ---- #
# S4 ----
tmpVersionS4_RF_MASE <- "v2021071201"
tmpFileS4_RF_MASE    <- paste0(
  "results/results_TS4_MASE_rf_", tmpVersionS4_RF_MASE, ".rdata"
)
tmpS4_RF_MASE <- get(load(tmpFileS4_RF_MASE))

ggsave(
  filename = paste0(
    tmpSavePath, "/SIM_RF_bayes_MASE_S4_", tmpVersionS4_RF_MASE, ".png"
  ),
  plot     = get_proportion_plot(tmpS4_RF_MASE, "cep-Holdout"),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

# \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ####
# ========================================================================= #
# ========================================================================= #
####                            Results SIM-RIDGE                        ####
# ========================================================================= #
# ========================================================================= #

## \___ MSE ---------------------------------------------------------------

# ---- #
# S3 ----
tmpVersionS3_RIDGE_MSE  <- "v2021070901"

tmpFileS3_RIDGE_MSE    <- paste0(
  "results/results_ts3_ridge_", tmpVersionS3_RIDGE_MSE, ".rdata"
)
tmpS3_RIDGE_MSE <- get(load(tmpFileS3_RIDGE_MSE))

ggsave(
  filename = paste0(
    tmpSavePath, "/SIM_RIDGE_bayes_MSE_S3_", tmpVersionS3_RIDGE_MSE,
    ".png"
  ),
  plot     = get_proportion_plot(tmpS3_RIDGE_MSE, "cep-Holdout"),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)


# ---- #
```

```
# S4 ----
tmpVersionS4_RIDGE_MSE  <- "v2021070901"

tmpFileS4_RIDGE_MSE    <- paste0(
  "results/results_ts4_ridge_", tmpVersionS4_RIDGE_MSE, ".rdata"
)
tmpS4_RIDGE_MSE <- get(load(tmpFileS4_RIDGE_MSE))

ggsave(
  filename = paste0(
    tmpSavePath, "/SIM_RIDGE_bayes_MSE_S4_", tmpVersionS4_RIDGE_MSE,
    ".png"
  ),
  plot     = get_proportion_plot(tmpS4_RIDGE_MSE, "cep-Holdout"),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

## \___ MASE -------------------------------------------------------------

# ---- #
# S3 ----
tmpVersionS3_RIDGE_MASE <- "v2021071201"
tmpFileS3_RIDGE_MASE    <- paste0(
  "results/results_TS3_MASE_ridge_", tmpVersionS3_RIDGE_MASE, ".rdata"
)
tmpS3_RIDGE_MASE <- get(load(tmpFileS3_RIDGE_MASE))

ggsave(
  filename = paste0(
    tmpSavePath, "/SIM_RIDGE_bayes_MASE_S3_", tmpVersionS3_RIDGE_MASE,
    ".png"
  ),
  plot     = get_proportion_plot(tmpS3_RIDGE_MASE, "cep-Holdout"),
  device   = "png",
  type     = "cairo",
  width    = 6,
  height   = 4
)

# ---- #
# S4 ----
tmpVersionS4_RIDGE_MASE <- "v2021071201"
tmpFileS4_RIDGE_MASE    <- paste0(
  "results/results_TS4_MASE_ridge_", tmpVersionS4_RIDGE_MASE, ".rdata"
)
tmpS4_RIDGE_MASE <- get(load(tmpFileS4_RIDGE_MASE))

ggsave(
  filename = paste0(
    tmpSavePath, "/SIM_RIDGE_bayes_MASE_S4_", tmpVersionS4_RIDGE_MASE,
    ".png"
```

```
  ),
  plot      = get_proportion_plot(tmpS4_RIDGE_MASE, "cep-Holdout"),
  device    = "png",
  type      = "cairo",
  width     = 6,
  height    = 4
)
beepr::beep("fanfare")
```

C.10 Forecasts for the M4 Competition series using the GEARS strategy

code/evaluation_M4_v2020080702.R

```
#' Applying the GEARS strategy to the 100,000 time series from the
#' M4 Forecasting Competition

# > Packages ============================================================

# install.packages("devtools")
library(devtools)

# Install the GEARS package

## Access Token:
GITHUB_PAT <- "b9b7b8b9d384ff89000d1ba40cb0d2e761c273b3"
install_github("gu-stat/gears", auth_token = GITHUB_PAT)

## Call the package
library(gears)

# Install the M4comp2018 package with data from the M4 Competition
# install.packages(
#   "https://github.com/carlanetto/M4comp2018/releases/download/0.2.0/
   M4comp2018_0.2.0.tar.gz",
#   repos=NULL
# )

## Call the package
library(M4comp2018)

# Install the future.apply package
# install.packages("future.apply")

## Call the package
library("future.apply")

# > Path ================================================================

tmpPathM4 <- "./M4GearsResults"

# ********************************************************************* ----
# DATA                                                                  ----
# >                                                                     ----

# |__ M4 Data Sets ======================================================

# M4 <- M4comp2018::M4

# "st"     : series_name
# "period" : periodicity
# "n"      : sample_size
# "h"      : forecast.horizon
# "x"      : data_train
```

```r
# "xx"     : data_test
# "pt_ff"  : Point Forecasts (top 25 submissions, one on each row)
# "low_ff" : Prediction Interval - Lower Bound (top 25 submissions)
# "up_ff"  : Prediction Interval - Upper Bound (top 25 submissions)

# \____ Hourly ----------------------------------------------------------

M4.Hourly <- Filter(function(l) l$period == "Hourly", M4comp2018::M4)

M4.Hourly.forecasts <- Map(function(l) l$pt_ff, M4.Hourly)

# \____ Daily -----------------------------------------------------------

M4.Daily <- Filter(function(l) l$period == "Daily", M4comp2018::M4)

M4.Daily.forecasts <- Map(function(l) l$pt_ff, M4.Daily)

# \____ Yearly ----------------------------------------------------------

M4.Yearly <- Filter(function(l) l$period == "Yearly", M4comp2018::M4)

M4.Yearly.forecasts <- Map(function(l) l$pt_ff, M4.Yearly)

# \____ Weekly ----------------------------------------------------------

M4.Weekly <- Filter(function(l) l$period == "Weekly", M4comp2018::M4)

M4.Weekly.forecasts <- Map(function(l) l$pt_ff, M4.Weekly)

# \____ Quarterly -------------------------------------------------------

M4.Quarterly <-
  Filter(function(l) l$period == "Quarterly", M4comp2018::M4)

M4.Quarterly.forecasts <- Map(function(l) l$pt_ff, M4.Quarterly)

# \____ Monthly ---------------------------------------------------------

M4.Monthly <-
  Filter(function(l) l$period == "Monthly", M4comp2018::M4)

M4.Monthly.forecasts <- Map(function(l) l$pt_ff, M4.Monthly)

# ************************************************************ ----
# Analysis                                                     ----
# >                                                            ----

# |__ Hourly ==============================================================

tmpVersionHourly <- "v2020080702"
tmpFileHourlyAll    <- paste0(
  tmpPathM4, "/Hourly_All_One_Step_", tmpVersionHourly, ".rdata"
)
```

```r
tmpFileHourlySummary    <- paste0(
  tmpPathM4, "/Hourly_Summary_One_Step_", tmpVersionHourly, ".rdata"
)


plan(multisession, workers = 16)

timeHourly <- system.time({
  hourlyOptim <- future_lapply(
    X = 1:length(M4.Hourly),
    function(X) {

      tmpDeseason <- deseason(
        ts.data          = M4.Hourly[[X]]$x,
        ts.frequency     = stats::frequency(M4.Hourly[[X]]$x),
        alpha.level      = 0.05,
        forecast.horizon = 1 #M4.Hourly[[X]]$h
      )

      tmpOptim <- gears_optim(
        DATA             = tmpDeseason$deseasonTS,
        forecast.horizon = 1, #M4.Hourly[[X]]$h
        search.size.rs   = c(144, 450),
        search.number.rs = c(24, 36, 48),
        last.obs         = M4.Hourly[[X]]$n,
        y.max.lags       = 2,
        use.intercept    = "both",
        error.measure    = "smape",
        betas.selection  = "both",
        use.parallel     = FALSE
      )

      # Estimation
      tmpGears <- gears(
        DATA             = tmpDeseason$deseasonTS,
        forecast.horizon = 1, #M4.Hourly[[X]]$h,
        size.rs          = tmpOptim[1, "size.rs"],
        number.rs        = tmpOptim[1, "number.rs"],
        last.obs         = M4.Hourly[[X]]$n,
        y.max.lags       = 2,
        use.intercept    = as.character(tmpOptim[1, "intercept"]),
        error.measure    = "smape",
        betas.selection  = as.character(tmpOptim[1, "betas"]),
        use.parallel     = FALSE
      )

      return(list(
        forecasts =
          tmpGears$out_sample_forecasts * tmpDeseason$seasonalComp,

        lower     = tmpGears$lower * tmpDeseason$seasonalComp,
        upper     = tmpGears$upper * tmpDeseason$seasonalComp
      ))
    },
```

```
    future.seed = 0xBEEF
  )
})

plan(sequential)

timeHourly[3]/60 # ~ 5.477333 min

save(hourlyOptim, file = tmpFileHourlyAll)

# \____ OWA Results - M4 + Gears ---------------------------------------

allHourlyForecasts <- lapply(
  X = 1:length(hourlyOptim),
  function(X) {
    t(t(c(M4.Hourly.forecasts[[X]][, 1], hourlyOptim[[X]]$forecasts)))
  }
)

allResultsHourly <- evaluationM4_One_Step(
  DATA = M4.Hourly,
  forecast.list = allHourlyForecasts,
  alpha = 0.05
)

rownames(allResultsHourly) <-
  c(as.character(M4comp2018::submission_info$ID[1:25]), "GEARS")

# allResultsHourly

save(allResultsHourly, file = tmpFileHourlySummary)
beepr::beep("fanfare")

# \____ Forecasts Naive2 -----------------------------------------------

plan(multisession, workers = 16)

tmpForecastsHourlyNaive2 <- future_lapply(
  X = 1:length(M4.Hourly),
  function(X) {
    forecast_naive2(
      ts.data          = M4.Hourly[[X]]$x,
      ts.frequency     = stats::frequency(M4.Hourly[[X]]$x),
      forecast.horizon = 1,
      alpha.level      = 0.05
    )
  },
  future.seed = 0xBEEF
)

## SMAPE for naive2

smapeM4HourlyNaive2 <- future_lapply(
  X = 1:length(M4.Hourly),
```

```
  function(X) {

    tmp <- error_measures(
      forecasts         = tmpForecastsHourlyNaive2[[X]],
      outsample         = M4.Hourly[[X]]$xx[1],
      insample          = M4.Hourly[[X]]$x,
      ts.frequency      = stats::frequency(M4.Hourly[[X]]$x),
      forecast.horizon  = 1,
      alpha.level       = 0.05,
      error.measure     = "smape"
    )

    unlist(tmp)
  },
  future.seed = 0xBEEF
)

smapeM4HourlyNaive2 <- mean(unlist(smapeM4HourlyNaive2))

## MASE for naive2

maseM4HourlyNaive2 <- future_lapply(
  X = 1:length(M4.Hourly),
  function(X) {

    tmp <- error_measures(
      forecasts         = tmpForecastsHourlyNaive2[[X]],
      outsample         = M4.Hourly[[X]]$xx[1],
      insample          = M4.Hourly[[X]]$x,
      ts.frequency      = stats::frequency(M4.Hourly[[X]]$x),
      forecast.horizon  = 1,
      alpha.level       = 0.05,
      error.measure     = "mase"
    )

    unlist(tmp)
  },
  future.seed = 0xBEEF
)

maseM4HourlyNaive2 <- mean(unlist(maseM4HourlyNaive2))

plan(sequential)

# |__ Weekly ============================================================

tmpVersionWeekly <- "v2020080702"
tmpFileWeeklyAll    <- paste0(
  tmpPathM4, "/Weekly_All_One_Step_", tmpVersionWeekly, ".rdata"
)

tmpFileWeeklySummary    <- paste0(
  tmpPathM4, "/Weekly_Summary_One_Step_", tmpVersionWeekly, ".rdata"
)
```

```
266
268 plan(multisession, workers = 16)
270 timeWeekly <- system.time({
  weeklyOptim <- future_lapply(
272   X = 1:length(M4.Weekly),
    function(X) {
274
      if (M4.Weekly[[X]]$n == 80) {
276     tmp.search.size.rs   <- c(26, 43)
        tmp.search.number.rs <- c(5, 10)
278   } else {
        tmp.search.size.rs   <- c(80, 150)
280     tmp.search.number.rs <- c(26, 52)
      }
282
      tmpDeseason <- deseason(
284     ts.data          = M4.Weekly[[X]]$x,
        ts.frequency     = stats::frequency(M4.Weekly[[X]]$x),
286     alpha.level      = 0.05,
        forecast.horizon = 1 #M4.Weekly[[X]]$h
288   )

290   tmpOptim <- gears_optim(
        DATA             = tmpDeseason$deseasonTS,
292     forecast.horizon = 1, #M4.Weekly[[X]]$h
        search.size.rs   = tmp.search.size.rs,
294     search.number.rs = tmp.search.number.rs,
        last.obs         = M4.Weekly[[X]]$n,
296     y.max.lags       = 2,
        use.intercept    = "both",
298     error.measure    = "smape",
        betas.selection  = "both",
300     use.parallel     = FALSE
      )
302
      # Estimation
304   tmpGears <- gears(
        DATA             = tmpDeseason$deseasonTS,
306     forecast.horizon = 1, #M4.Weekly[[X]]$h,
        size.rs          = tmpOptim[1, "size.rs"],
308     number.rs        = tmpOptim[1, "number.rs"],
        last.obs         = M4.Weekly[[X]]$n,
310     y.max.lags       = 2,
        use.intercept    = as.character(tmpOptim[1, "intercept"]),
312     error.measure    = "smape",
        betas.selection  = as.character(tmpOptim[1, "betas"]),
314     use.parallel     = FALSE
      )
316
      return(list(
318     forecasts =
          tmpGears$out_sample_forecasts * tmpDeseason$seasonalComp,
```

```
320
        lower     = tmpGears$lower * tmpDeseason$seasonalComp,
322     upper     = tmpGears$upper * tmpDeseason$seasonalComp
      ))
324   },
    future.seed = 0xBEEF
326 )
})
328
plan(sequential)
330
timeWeekly[3]/60 # ~ 5.477333 min
332
save(weeklyOptim, file = tmpFileWeeklyAll)
334
# \____ OWA Results - M4 + Gears ---------------------------------------
336
allWeeklyForecasts <- lapply(
338   X = 1:length(weeklyOptim),
  function(X) {
340   t(t(c(M4.Weekly.forecasts[[X]][, 1], weeklyOptim[[X]]$forecasts)))
  }
342 )

344 allResultsWeekly <- evaluationM4_One_Step(
  DATA = M4.Weekly,
346   forecast.list = allWeeklyForecasts,
  alpha = 0.05
348 )

350 rownames(allResultsWeekly) <-
  c(as.character(M4comp2018::submission_info$ID[1:25]), "GEARS")
352
# allResultsWeekly
354
save(allResultsWeekly, file = tmpFileWeeklySummary)
356 beepr::beep("fanfare")

358 # \____ Forecasts Naive2 -----------------------------------------------

360 plan(multisession, workers = 16)

362 tmpForecastsWeeklyNaive2 <- future_lapply(
  X = 1:length(M4.Weekly),
364   function(X) {
    forecast_naive2(
366       ts.data          = M4.Weekly[[X]]$x,
      ts.frequency     = stats::frequency(M4.Weekly[[X]]$x),
368       forecast.horizon = 1,
      alpha.level      = 0.05
370   )
  },
372   future.seed = 0xBEEF
)
```

```
374
## SMAPE for naive2
376
smapeM4WeeklyNaive2 <- future_lapply(
378   X = 1:length(M4.Weekly),
   function(X) {
380
     tmp <- error_measures(
382       forecasts         = tmpForecastsWeeklyNaive2[[X]],
       outsample         = M4.Weekly[[X]]$xx[1],
384       insample          = M4.Weekly[[X]]$x,
       ts.frequency      = stats::frequency(M4.Weekly[[X]]$x),
386       forecast.horizon  = 1,
       alpha.level       = 0.05,
388       error.measure     = "smape"
     )
390
     unlist(tmp)
392   },
   future.seed = 0xBEEF
394 )

396 smapeM4WeeklyNaive2 <- mean(unlist(smapeM4WeeklyNaive2))

398 ## MASE for naive2

400 maseM4WeeklyNaive2 <- future_lapply(
   X = 1:length(M4.Weekly),
402   function(X) {

404     tmp <- error_measures(
       forecasts         = tmpForecastsWeeklyNaive2[[X]],
406       outsample         = M4.Weekly[[X]]$xx[1],
       insample          = M4.Weekly[[X]]$x,
408       ts.frequency      = stats::frequency(M4.Weekly[[X]]$x),
       forecast.horizon  = 1,
410       alpha.level       = 0.05,
       error.measure     = "mase"
412     )

414     unlist(tmp)
   },
416   future.seed = 0xBEEF
)
418
maseM4WeeklyNaive2 <- mean(unlist(maseM4WeeklyNaive2))
420
plan(sequential)
422
# |__ Daily ============================================================
424
tmpVersionDaily <- "v2020080702"
426 tmpFileDailyAll    <- paste0(
   tmpPathM4, "/Daily_All_One_Step_", tmpVersionDaily, ".rdata"
```

```
)

tmpFileDailySummary    <- paste0(
  tmpPathM4, "/Daily_Summary_One_Step_", tmpVersionDaily, ".rdata"
)

# Sample sizes
tmpNDaily <- sapply(
  X = 1:length(M4.Daily),
  function(X) M4.Daily[[X]]$n
)

table(tmpNDaily)

plan(multisession, workers = 16)

timeDaily <- system.time({
  dailyOptim <- future_lapply(
    X = 1:length(M4.Daily),

    function(X) {

      tmp.search.size.rs   <- c(30, 60)
      tmp.search.number.rs <- c(12)

      if (X %in% c(34, 2211)) {
        tmp.search.betas = "last"
      } else {
        tmp.search.betas = "both"
      }

      if (X %in% c(131, 2085, 2211, 2219) ) {
        tmp.intercept = "without"
      } else {
        tmp.intercept = "both"
      }

      tmpDeseason <- deseason(
        ts.data          = M4.Daily[[X]]$x,
        ts.frequency     = stats::frequency(M4.Daily[[X]]$x),
        alpha.level      = 0.05,
        forecast.horizon = 1 #M4.Daily[[X]]$h
      )

      tmpOptim <- gears_optim(
        DATA             = tmpDeseason$deseasonTS,
        forecast.horizon = 1, #M4.Daily[[X]]$h
        search.size.rs   = tmp.search.size.rs,
        search.number.rs = tmp.search.number.rs,
        last.obs         = M4.Daily[[X]]$n,
        y.max.lags       = 2,
        use.intercept    = tmp.intercept,
        error.measure    = "smape",
        betas.selection  = tmp.search.betas,
```

```
        use.parallel     = FALSE
      )

      # Estimation
      tmpGears <- gears(
        DATA             = tmpDeseason$deseasonTS,
        forecast.horizon = 1, #M4.Daily[[X]]$h,
        size.rs          = tmpOptim[1, "size.rs"],
        number.rs        = tmpOptim[1, "number.rs"],
        last.obs         = M4.Daily[[X]]$n,
        y.max.lags       = 2,
        use.intercept    = as.character(tmpOptim[1, "intercept"]),
        error.measure    = "smape",
        betas.selection  = as.character(tmpOptim[1, "betas"]),
        use.parallel     = FALSE
      )
      #cat(X)
      return(list(
        forecasts =
          tmpGears$out_sample_forecasts * tmpDeseason$seasonalComp,

        lower     = tmpGears$lower * tmpDeseason$seasonalComp,
        upper     = tmpGears$upper * tmpDeseason$seasonalComp
      ))
    },
    future.seed = 0xBEEF
  )
})

plan(sequential)

timeDaily[3]/60 # ~ 6.286167 min

save(dailyOptim, file = tmpFileDailyAll)

# \____ OWA Results - M4 + Gears ------------------------------------

allDailyForecasts <- lapply(
  X = 1:length(dailyOptim),
  function(X) {
    t(t(c(M4.Daily.forecasts[[X]][, 1], dailyOptim[[X]]$forecasts)))
  }
)

allResultsDaily <- evaluationM4_One_Step(
  DATA = M4.Daily,
  forecast.list = allDailyForecasts,
  alpha = 0.05
) # ~ 7 min

rownames(allResultsDaily) <-
  c(as.character(M4comp2018::submission_info$ID[1:25]), "GEARS")

# allResultsDaily
```

```
536
save(allResultsDaily, file = tmpFileDailySummary)
538 beepr::beep("fanfare")

540 # \____ Forecasts Naive2 ----------------------------------------------

542 plan(multisession, workers = 16)

544 tmpForecastsDailyNaive2 <- future_lapply(
  X = 1:length(M4.Daily),
546   function(X) {
    forecast_naive2(
548       ts.data          = M4.Daily[[X]]$x,
      ts.frequency     = stats::frequency(M4.Daily[[X]]$x),
550       forecast.horizon = 1,
      alpha.level      = 0.05
552     )
  },
554   future.seed = 0xBEEF
)
556
## SMAPE for naive2
558
smapeM4DailyNaive2 <- future_lapply(
560   X = 1:length(M4.Daily),
  function(X) {
562
    tmp <- error_measures(
564       forecasts         = tmpForecastsDailyNaive2[[X]],
      outsample         = M4.Daily[[X]]$xx[1],
566       insample          = M4.Daily[[X]]$x,
      ts.frequency      = stats::frequency(M4.Daily[[X]]$x),
568       forecast.horizon  = 1,
      alpha.level       = 0.05,
570       error.measure     = "smape"
    )
572
    unlist(tmp)
574   },
  future.seed = 0xBEEF
576 )

578 smapeM4DailyNaive2 <- mean(unlist(smapeM4DailyNaive2))

580 ## MASE for naive2

582 maseM4DailyNaive2 <- future_lapply(
  X = 1:length(M4.Daily),
584   function(X) {

586     tmp <- error_measures(
      forecasts         = tmpForecastsDailyNaive2[[X]],
588       outsample         = M4.Daily[[X]]$xx[1],
      insample          = M4.Daily[[X]]$x,
```

```
      ts.frequency      = stats::frequency(M4.Daily[[X]]$x),
      forecast.horizon  = 1,
      alpha.level       = 0.05,
      error.measure     = "mase"
    )

    unlist(tmp)
  },
  future.seed = 0xBEEF
)

maseM4DailyNaive2 <- mean(unlist(maseM4DailyNaive2))

plan(sequential)

# |__ Yearly ============================================================

tmpVersionYearly <- "v2020080702"
tmpFileYearlyAll    <- paste0(
  tmpPathM4, "/Yearly_All_One_Step_", tmpVersionYearly, ".rdata"
)

tmpFileYearlySummary    <- paste0(
  tmpPathM4, "/Yearly_Summary_One_Step_", tmpVersionYearly, ".rdata"
)

# Sample sizes
tmpNYearly <- sapply(
  X = 1:length(M4.Yearly),
  function(X) M4.Yearly[[X]]$n
)

table(tmpNYearly)
which(tmpNYearly == 13)

plan(multisession, workers = 16)

beepr::beep_on_error({timeYearly <- system.time({
  yearlyOptim <- future_lapply(
    X = 1:length(M4.Yearly),
    #lapply(
    #X = 21547,
    function(X) {

      if (M4.Yearly[[X]]$n <= 15) {
        tmp.search.size.rs   <- c(6)
        tmp.search.number.rs <- c(3)

      } else if (M4.Yearly[[X]]$n < 50) {
        tmp.search.size.rs   <- c(5, 6)
        tmp.search.number.rs <- c(5)

      } else if (M4.Yearly[[X]]$n < 100) {
        tmp.search.size.rs   <- c(10, 20)
```

```
    #tmp.search.size.rs   <- c(20)
    tmp.search.number.rs <- c(5)
  } else {
    tmp.search.size.rs   <- c(20, 60)
    #tmp.search.size.rs   <- c(60)
    tmp.search.number.rs <- c(30)
  }

  if (X %in% c(609, 9012, 9875, 10289, 10033, 12143, 12147, 15088,
               17087, 21124)) {
    tmp.search.betas = "last"
  } else {
    tmp.search.betas = "both"
  }

  if (X %in% c(3472, 3792, 9012, 9861, 10289, 10033, 12143, 12146,
               12149, 13143, 13332, 13335, 14244, 14833, 15088,
               17084, 17086, 17087, 21124,
               21168, 21547, 22380, 22466) ) {
    tmp.intercept = "without"
  } else {
    tmp.intercept = "both"
  }

  #tmp.search.betas = "both"
  #tmp.intercept = "both"

  tmpDeseason <- deseason(
    ts.data          = M4.Yearly[[X]]$x,
    ts.frequency     = stats::frequency(M4.Yearly[[X]]$x),
    alpha.level      = 0.05,
    forecast.horizon = 1 #M4.Yearly[[X]]$h
  )

  tmpOptim <- gears_optim(
    DATA             = tmpDeseason$deseasonTS,
    forecast.horizon = 1, #M4.Yearly[[X]]$h
    search.size.rs   = tmp.search.size.rs,
    search.number.rs = tmp.search.number.rs,
    last.obs         = M4.Yearly[[X]]$n,
    y.max.lags       = 2,
    use.intercept    = tmp.intercept,
    error.measure    = "smape",
    betas.selection  = tmp.search.betas,
    use.parallel     = FALSE
  )

  # Estimation
  tmpGears <- gears(
    DATA             = tmpDeseason$deseasonTS,
    forecast.horizon = 1, #M4.Yearly[[X]]$h,
    size.rs          = tmpOptim[1, "size.rs"],
    number.rs        = tmpOptim[1, "number.rs"],
    last.obs         = M4.Yearly[[X]]$n,
```

```
        y.max.lags       = 2,
        use.intercept    = as.character(tmpOptim[1, "intercept"]),
        error.measure    = "smape",
        betas.selection  = as.character(tmpOptim[1, "betas"]),
        use.parallel     = FALSE
      )
      cat(paste0(X, "."))
      return(list(
        forecasts =
          tmpGears$out_sample_forecasts * tmpDeseason$seasonalComp,

        lower     = tmpGears$lower * tmpDeseason$seasonalComp,
        upper     = tmpGears$upper * tmpDeseason$seasonalComp
      ))
    }
    #)
    , future.seed = 0xBEEF
  )
})
},
sound = 9)

plan(sequential)

timeYearly[3]/60 # ~ 12.75983 min

save(yearlyOptim, file = tmpFileYearlyAll)
beepr::beep("fanfare")

# \____ OWA Results - M4 + Gears ---------------------------------------

allYearlyForecasts <- lapply(
  X = 1:length(yearlyOptim),
  function(X) {
    t(t(c(M4.Yearly.forecasts[[X]][, 1], yearlyOptim[[X]]$forecasts)))
  }
)

allResultsYearly <- evaluationM4_One_Step(
  DATA = M4.Yearly,
  forecast.list = allYearlyForecasts,
  alpha = 0.05
) # ~ 7 min

rownames(allResultsYearly) <-
  c(as.character(M4comp2018::submission_info$ID[1:25]), "GEARS")

# allResultsYearly

save(allResultsYearly, file = tmpFileYearlySummary)
beepr::beep("fanfare")

# \____ Forecasts Naive2 -----------------------------------------------
```

```r
plan(multisession, workers = 16)

options(future.globals.maxSize= 891289600)

tmpForecastsYearlyNaive2 <- future_lapply(
  X = 1:length(M4.Yearly),
  function(X) {
    forecast_naive2(
      ts.data          = M4.Yearly[[X]]$x,
      ts.frequency     = stats::frequency(M4.Yearly[[X]]$x),
      forecast.horizon = 1,
      alpha.level      = 0.05
    )
  },
  future.seed = 0xBEEF
)

## SMAPE for naive2

smapeM4YearlyNaive2 <- future_lapply(
  X = 1:length(M4.Yearly),
  function(X) {

    tmp <- error_measures(
      forecasts         = tmpForecastsYearlyNaive2[[X]],
      outsample         = M4.Yearly[[X]]$xx[1],
      insample          = M4.Yearly[[X]]$x,
      ts.frequency      = stats::frequency(M4.Yearly[[X]]$x),
      forecast.horizon  = 1,
      alpha.level       = 0.05,
      error.measure     = "smape"
    )

    unlist(tmp)
  },
  future.seed = 0xBEEF
)

smapeM4YearlyNaive2 <- mean(unlist(smapeM4YearlyNaive2))

## MASE for naive2

maseM4YearlyNaive2 <- future_lapply(
  X = 1:length(M4.Yearly),
  function(X) {

    tmp <- error_measures(
      forecasts         = tmpForecastsYearlyNaive2[[X]],
      outsample         = M4.Yearly[[X]]$xx[1],
      insample          = M4.Yearly[[X]]$x,
      ts.frequency      = stats::frequency(M4.Yearly[[X]]$x),
      forecast.horizon  = 1,
      alpha.level       = 0.05,
      error.measure     = "mase"
```

```
    )

    unlist(tmp)
  },
  future.seed = 0xBEEF
)

maseM4YearlyNaive2 <- mean(unlist(maseM4YearlyNaive2))

plan(sequential)

# |__ Quarterly ============================================================

tmpVersionQuarterly <- "v2020080702"
tmpFileQuarterlyAll    <- paste0(
  tmpPathM4, "/Quarterly_All_One_Step_", tmpVersionQuarterly, ".rdata"
)

tmpFileQuarterlySummary    <- paste0(
  tmpPathM4,
  "/Quarterly_Summary_One_Step_",
  tmpVersionQuarterly,
  ".rdata"
)

# Sample sizes
tmpNQuarterly <- sapply(
  X = 1:length(M4.Quarterly),
  function(X) M4.Quarterly[[X]]$n
)

table(tmpNQuarterly)
which(tmpNQuarterly == 26)

plan(multisession, workers = 16)

beepr::beep_on_error({timeQuarterly <- system.time({
  quarterlyOptim <- future_lapply(
    X = 1:length(M4.Quarterly),
    #lapply(
    #X = 19636,
    function(X) {

      if (M4.Quarterly[[X]]$n <= 18) {
        tmp.search.size.rs   <- c(5)
        tmp.search.number.rs <- c(3)

      } else if (M4.Quarterly[[X]]$n <= 25) {
         tmp.search.size.rs   <- c(8)
         tmp.search.number.rs <- c(4)
      } else if (M4.Quarterly[[X]]$n <= 50) {
        tmp.search.size.rs   <- c(14)
        #tmp.search.size.rs   <- c(20)
        tmp.search.number.rs <- c(4)
```

```
860       } else {
            tmp.search.size.rs   <- c(20)
862         #tmp.search.size.rs   <- c(60)
            tmp.search.number.rs <- c(4, 12)
864       }

866       if (X %in% c(19636, 19680)) {
            tmp.search.betas = "last"
868       } else {
            tmp.search.betas = "both"
870       }

872       if (X %in% c(5619, 14727, 19636, 19680) ) {
            tmp.intercept = "without"
874       } else {
            tmp.intercept = "both"
876       }

878       #tmp.search.betas = "both"
          #tmp.intercept = "both"
880
          tmpDeseason <- deseason(
882         ts.data          = M4.Quarterly[[X]]$x,
            ts.frequency     = stats::frequency(M4.Quarterly[[X]]$x),
884         alpha.level      = 0.05,
            forecast.horizon = 1 #M4.Quarterly[[X]]$h
886       )

888       tmpOptim <- gears_optim(
            DATA             = tmpDeseason$deseasonTS,
890         forecast.horizon = 1, #M4.Quarterly[[X]]$h
            search.size.rs   = tmp.search.size.rs,
892         search.number.rs = tmp.search.number.rs,
            last.obs         = M4.Quarterly[[X]]$n,
894         y.max.lags       = 2,
            use.intercept    = tmp.intercept,
896         error.measure    = "smape",
            betas.selection  = tmp.search.betas,
898         use.parallel     = FALSE
          )
900
          # Estimation
902       tmpGears <- gears(
            DATA             = tmpDeseason$deseasonTS,
904         forecast.horizon = 1, #M4.Quarterly[[X]]$h,
            size.rs          = tmpOptim[1, "size.rs"],
906         number.rs        = tmpOptim[1, "number.rs"],
            last.obs         = M4.Quarterly[[X]]$n,
908         y.max.lags       = 2,
            use.intercept    = as.character(tmpOptim[1, "intercept"]),
910         error.measure    = "smape",
            betas.selection  = as.character(tmpOptim[1, "betas"]),
912         use.parallel     = FALSE
          )
```

```
      cat(paste0(X, "."))
      return(list(
        forecasts =
          tmpGears$out_sample_forecasts * tmpDeseason$seasonalComp,

        lower     = tmpGears$lower * tmpDeseason$seasonalComp,
        upper     = tmpGears$upper * tmpDeseason$seasonalComp
      ))
    }
    #)
    , future.seed = 0xBEEF
  )
})
},
sound = 9)

plan(sequential)

timeQuarterly[3]/60 # ~ 7.161167 min (2: ~20.094 min)

save(quarterlyOptim, file = tmpFileQuarterlyAll)
beepr::beep("fanfare")

# \____ OWA Results - M4 + Gears ---------------------------------------

allQuarterlyForecasts <- lapply(
  X = 1:length(quarterlyOptim),
  function(X) {
    t(t(c(
      M4.Quarterly.forecasts[[X]][, 1], quarterlyOptim[[X]]$forecasts
    )))
  }
)

allResultsQuarterly <- evaluationM4_One_Step(
  DATA = M4.Quarterly,
  forecast.list = allQuarterlyForecasts,
  alpha = 0.05
) # ~ 7 min

rownames(allResultsQuarterly) <-
  c(as.character(M4comp2018::submission_info$ID[1:25]), "GEARS")

# allResultsQuarterly

save(allResultsQuarterly, file = tmpFileQuarterlySummary)
beepr::beep("fanfare")

# \____ Forecasts Naive2 -----------------------------------------------

plan(multisession, workers = 16)

options(future.globals.maxSize= 891289600)
```

```
tmpForecastsQuarterlyNaive2 <- future_lapply(
  X = 1:length(M4.Quarterly),
  function(X) {
    forecast_naive2(
      ts.data          = M4.Quarterly[[X]]$x,
      ts.frequency     = stats::frequency(M4.Quarterly[[X]]$x),
      forecast.horizon = 1,
      alpha.level      = 0.05
    )
  },
  future.seed = 0xBEEF
)

## SMAPE for naive2

smapeM4QuarterlyNaive2 <- future_lapply(
  X = 1:length(M4.Quarterly),
  function(X) {

    tmp <- error_measures(
      forecasts         = tmpForecastsQuarterlyNaive2[[X]],
      outsample         = M4.Quarterly[[X]]$xx[1],
      insample          = M4.Quarterly[[X]]$x,
      ts.frequency      = stats::frequency(M4.Quarterly[[X]]$x),
      forecast.horizon  = 1,
      alpha.level       = 0.05,
      error.measure     = "smape"
    )

    unlist(tmp)
  },
  future.seed = 0xBEEF
)

smapeM4QuarterlyNaive2 <- mean(unlist(smapeM4QuarterlyNaive2))

## MASE for naive2

maseM4QuarterlyNaive2 <- future_lapply(
  X = 1:length(M4.Quarterly),
  function(X) {

    tmp <- error_measures(
      forecasts         = tmpForecastsQuarterlyNaive2[[X]],
      outsample         = M4.Quarterly[[X]]$xx[1],
      insample          = M4.Quarterly[[X]]$x,
      ts.frequency      = stats::frequency(M4.Quarterly[[X]]$x),
      forecast.horizon  = 1,
      alpha.level       = 0.05,
      error.measure     = "mase"
    )

    unlist(tmp)
  },
```

```
  future.seed = 0xBEEF
)

maseM4QuarterlyNaive2 <- mean(unlist(maseM4QuarterlyNaive2))

plan(sequential)

# |__ Monthly ============================================================

tmpVersionMonthly <- "v2020080702"
tmpFileMonthlyAll    <- paste0(
  tmpPathM4, "/Monthly_All_One_Step_", tmpVersionMonthly, ".rdata"
)

tmpFileMonthlySummary    <- paste0(
  tmpPathM4,
  "/Monthly_Summary_One_Step_",
  tmpVersionMonthly,
  ".rdata"
)

# Sample sizes
tmpNMonthly <- sapply(
  X = 1:length(M4.Monthly),
  function(X) M4.Monthly[[X]]$n
)

table(tmpNMonthly)
which(tmpNMonthly == 26)

options(future.globals.maxSize= 891289600)

plan(multisession, workers = 16)

beepr::beep_on_error({timeMonthly <- system.time({
  monthlyOptim <- future_lapply(
    X = 1:length(M4.Monthly),
    #lapply(
    #X = 47981:length(M4.Monthly),
    #X = 47810,
    function(X) {

      if (M4.Monthly[[X]]$n <= 45) {
        tmp.search.size.rs   <- c(18)
        tmp.search.number.rs <- c(8)
      } else if (M4.Monthly[[X]]$n <= 53) {
        tmp.search.size.rs   <- c(22)
        tmp.search.number.rs <- c(8)
      } else if (M4.Monthly[[X]]$n <= 59) {
        tmp.search.size.rs   <- c(30)
        tmp.search.number.rs <- c(8)
      } else if (M4.Monthly[[X]]$n <= 100) {
        tmp.search.size.rs   <- c(36)
        tmp.search.number.rs <- c(8)
```

```
  } else {
    tmp.search.size.rs   <- c(48)
    tmp.search.number.rs <- c(10)
  }

  # } else if (M4.Monthly[[X]]$n <= 25) {
  #   tmp.search.size.rs   <- c(8)
  #   tmp.search.number.rs <- c(4)
  # } else if (M4.Monthly[[X]]$n <= 50) {
  #   tmp.search.size.rs   <- c(14)
  #   #tmp.search.size.rs   <- c(20)
  #   tmp.search.number.rs <- c(4)
  # } else {
  #   tmp.search.size.rs   <- c(20)
  #   #tmp.search.size.rs   <- c(60)
  #   tmp.search.number.rs <- c(4, 12)
  # }
  #
  if (X %in% c(3006, 16993, 34815, 38911)) {
    tmp.search.betas = "last"
  } else {
    tmp.search.betas = "both"
  }

  if (X %in% c(3006, 16993, 34815, 38911) ) {
    tmp.intercept = "without"
  } else {
    tmp.intercept = "both"
  }

  #tmp.search.betas = "both"
  #tmp.intercept = "both"

  tmpDeseason <- deseason(
    ts.data          = M4.Monthly[[X]]$x,
    ts.frequency     = stats::frequency(M4.Monthly[[X]]$x),
    alpha.level      = 0.05,
    forecast.horizon = 1 #M4.Monthly[[X]]$h
  )

  tmpOptim <- gears_optim(
    DATA             = tmpDeseason$deseasonTS,
    forecast.horizon = 1, #M4.Monthly[[X]]$h
    search.size.rs   = tmp.search.size.rs,
    search.number.rs = tmp.search.number.rs,
    last.obs         = M4.Monthly[[X]]$n,
    y.max.lags       = 2,
    use.intercept    = tmp.intercept,
    error.measure    = "smape",
    betas.selection  = tmp.search.betas,
    use.parallel     = FALSE
  )

  # Estimation
```

```
      tmpGears <- gears(
        DATA             = tmpDeseason$deseasonTS,
        forecast.horizon = 1, #M4.Monthly[[X]]$h,
        size.rs          = tmpOptim[1, "size.rs"],
        number.rs        = tmpOptim[1, "number.rs"],
        last.obs         = M4.Monthly[[X]]$n,
        y.max.lags       = 2,
        use.intercept    = as.character(tmpOptim[1, "intercept"]),
        error.measure    = "smape",
        betas.selection  = as.character(tmpOptim[1, "betas"]),
        use.parallel     = FALSE
      )
      #if (X %in% c(seq(1:48)*1000)) cat(paste0(X, "."))
      cat(paste0(X, "."))
      return(list(
        forecasts =
          tmpGears$out_sample_forecasts * tmpDeseason$seasonalComp,

        lower     = tmpGears$lower * tmpDeseason$seasonalComp,
        upper     = tmpGears$upper * tmpDeseason$seasonalComp
      ))
    }
    #)
    , future.seed = 0xBEEF
  )
})
},
sound = 9)

plan(sequential)

timeMonthly[3]/60 # ~ 7.161167 min (2: ~20.094 min)

save(monthlyOptim, file = tmpFileMonthlyAll)
beepr::beep("fanfare")

# \____ OWA Results - M4 + Gears ------------------------------------

allMonthlyForecasts <- lapply(
  X = 1:length(monthlyOptim),
  function(X) {
    t(t(c(
      M4.Monthly.forecasts[[X]][, 1], monthlyOptim[[X]]$forecasts
    )))
  }
)

allResultsMonthly <- evaluationM4_One_Step(
  DATA = M4.Monthly,
  forecast.list = allMonthlyForecasts,
  alpha = 0.05
) # ~ 7 min

rownames(allResultsMonthly) <-
```

```
  c(as.character(M4comp2018::submission_info$ID[1:25]), "GEARS")

# allResultsMonthly

save(allResultsMonthly, file = tmpFileMonthlySummary)
beepr::beep("fanfare")

# \____ Forecasts Naive2 ------------------------------------------

plan(multisession, workers = 16)

options(future.globals.maxSize= 891289600)

tmpForecastsMonthlyNaive2 <- future_lapply(
  X = 1:length(M4.Monthly),
  function(X) {
    forecast_naive2(
      ts.data          = M4.Monthly[[X]]$x,
      ts.frequency     = stats::frequency(M4.Monthly[[X]]$x),
      forecast.horizon = 1,
      alpha.level      = 0.05
    )
  },
  future.seed = 0xBEEF
)

## SMAPE for naive2

smapeM4MonthlyNaive2 <- future_lapply(
  X = 1:length(M4.Monthly),
  function(X) {

    tmp <- error_measures(
      forecasts         = tmpForecastsMonthlyNaive2[[X]],
      outsample         = M4.Monthly[[X]]$xx[1],
      insample          = M4.Monthly[[X]]$x,
      ts.frequency      = stats::frequency(M4.Monthly[[X]]$x),
      forecast.horizon  = 1,
      alpha.level       = 0.05,
      error.measure     = "smape"
    )

    unlist(tmp)
  },
  future.seed = 0xBEEF
)

smapeM4MonthlyNaive2 <- mean(unlist(smapeM4MonthlyNaive2))

## MASE for naive2

maseM4MonthlyNaive2 <- future_lapply(
  X = 1:length(M4.Monthly),
  function(X) {
```

```
1238
    tmp <- error_measures(
1240      forecasts         = tmpForecastsMonthlyNaive2[[X]],
      outsample         = M4.Monthly[[X]]$xx[1],
1242      insample          = M4.Monthly[[X]]$x,
      ts.frequency      = stats::frequency(M4.Monthly[[X]]$x),
1244      forecast.horizon  = 1,
      alpha.level       = 0.05,
1246      error.measure     = "mase"
    )
1248
    unlist(tmp)
1250  },
  future.seed = 0xBEEF
1252)

1254maseM4MonthlyNaive2 <- mean(unlist(maseM4MonthlyNaive2))

1256plan(sequential)
```

Appendix D: Optimized results for a subset of Daily time series from the M4 Competition

Table D.1: Daily time-series case number and their respective new and old values of the OWA, alongside the new values for S, M, intercept choice, and betas choice.

TS Number	New OWA	Previous OWA	New S	New M	Intercept	Betas
38	5.86	115.80	7	90	without	average
261	5.81	16.19	8	5	without	last
493	4.21	26.31	8	10	with	average
681	4.09	11.65	50	19	without	last
697	1.31	137.91	50	20	without	average
700	4.56	12.42	50	17	without	last
701	1.47	92.17	8	15	without	average
704	5.36	14.35	50	17	without	last
709	4.85	13.11	50	17	without	last
809	7.19	12.82	50	17	without	last
838	3.90	10.06	50	16	without	last
863	4.97	12.88	21	16	without	last
1025	4.96	29.98	7	14	with	average
1063	8.71	25.65	9	12	without	average
1289	1.16	32.81	50	15	without	last
1451	0.32	29.17	13	12	without	last
1681	6.34	11.85	50	15	without	last
1686	6.11	12.02	50	15	without	last
1948	7.31	10.77	50	18	without	average
1993	8.23	23.70	50	19	with	average
2116	2.94	13.12	50	15	without	last
2218	4.29	12.54	11	14	without	average
2232	2.27	33.30	11	16	without	average
2256	0.44	14.56	50	16	without	last
2317	5.78	38.35	80	7	without	last

2378	6.06	103.47	12	6	with	last
2463	10.26	63.12	11	14	with	last
2487	7.26	81.69	11	14	without	average
2504	6.45	53.99	7	20	with	average
2532	1.00	30.56	11	14	without	last
2543	4.17	10.26	50	20	without	average
2548	0.36	21.30	9	7	with	average
2566	3.90	34.56	13	10	with	average
2569	2.34	16.56	50	15	without	last
2583	2.25	11.40	50	20	with	average
2640	14.21	40.93	80	7	without	last
2642	11.19	54.54	80	7	without	last
2691	1.78	11.35	50	15	without	last
2697	4.78	13.68	50	15	with	average
2744	2.42	13.08	8	10	with	average
2766	8.48	19.52	60	13	with	last
2789	4.70	15.62	30	15	with	last
2845	4.87	13.59	13	8	without	last
2974	7.99	11.85	50	19	with	average
3002	4.19	16.04	11	14	with	last
3016	1.40	38.48	50	15	with	average
3022	7.55	37.97	8	10	with	last
3023	0.44	20.05	120	7	without	last
3064	4.02	14.49	50	16	with	last
3102	11.87	86.32	6	8	with	average
3160	7.56	13.60	50	16	without	last
3232	4.34	36.83	7	30	with	last
3292	0.15	22.29	14	8	with	average
3398	1.74	49.56	6	7	with	average
3457	0.99	14.68	7	20	with	average
3559	1.52	39.14	50	15	with	average
3649	2.64	18.10	7	30	with	last
3706	5.94	10.17	50	20	with	average
4207	0.96	12.14	50	15	without	last

Appendix E: List of the top 20 papers found under the keywords 'machine learning time series' in number of citations, and whether they mention seasonality in their text.

Paper Number	Author	Title	Citations	Year	Cites Season/Seasonality/Deseason?
1	Cao, LJ and Tay, FEH	Support vector machine with adaptive parameters in financial time series forecasting	1099	2003	no
2	Sapankevych, NI and Sankar, R	Time series prediction using support vector machines: a survey	948	2009	yes, cite 1 paper that uses
3	Ahmed, AF and Atiya, NE and El-Shishiny, H	An empirical comparison of machine learning models for time series forecasting	535	2010	yes
4	Giles, CL, and Lawrence, S, and Tsoi, AC	Noisy time series prediction using recurrent neural networks and grammatical inference	487	2001	no
5	Bontempi, G and Taieb, SB and Le Borgne, YA	Machine learning strategies for time series forecasting	354	2012	no
6	Laptev, N and Yosinski, J and Li, LE and Smyl, S	Time-series extreme event forecasting with neural networks at uber	252	2017	yes
7	Recknagel, F	Applications of machine learning to ecological modelling	230	2001	not in a TS context
8	Wang, X and Han, M	Online sequential extreme learning machine with kernels for nonstationary time series prediction	183	2014	no
9	Cai, Y and Guan, K and Peng, J and Wang, S and Seifer, C and Wardlow, B and Li Z	A high-performance and in-season classification system of field-level crop types using time-series Landsat data and a machine learning approach	168	2018	yes
10	Siami-Namini, S and Tavakoli, N and Namin, AS	A comparison of ARIMA and LSTM in forecasting time series	164	2018	yes
11	Formisano, E and De Martino, F and Valente, G	Multivariate analysis of fMRI time series: classification and regression of brain responses using machine learning	160	2008	no
12	Kadous, MW, and Sammut, C	Classification of multivariate time series and structured data using constructive induction	150	2005	yes (but only mentions it)
13	Krollner, B and Vanstone, BJ and Finnie, GR	Financial time series forecasting with machine learning techniques: a survey.	141	2010	no
14	Singh, R and Balasundaram, S	Application of extreme learning machine method for time series analysis	111	2007	yes (but only mentions it)
15	Ak, R and Fink, O and Zio, E	Two machine learning approaches for short-term wind speed time-series prediction	105	2015	yes, but no mention of deseasonalizing it
16	Chou, JS and Ngo, NT	Time series analytics using sliding window metaheuristic optimization-based machine learning system for identifying building energy consumption patterns	100	2016	yes
17	Costello, Z and Martin, HG	A machine learning approach to predict metabolic pathway dynamics from time-series multiomics data	93	2018	no
18	Wang, Z and Zhang, M and Wang, D and Song, C and Liu, M and Li, J and Lou, L, and Liu, Z	Failure prediction using machine learning and time series in optical network	93	2017	no
19	Khosravi, A and Machado, L and Nunes, RO	Time-series prediction of wind speed using machine learning algorithms: A case study Osorio wind farm, Brazil	93	2018	no
20	Shih, SY, and Sun, FK and Lee, H	Temporal pattern attention for multivariate time series forecasting	88	2019	yes

Figure E.1: List of the top 20 papers found under the keywords 'machine learning time series' in number of citations, and whether they mention seasonality in their text.

Bibliography

[1] Raffaella Giacomini and Halbert White. Tests of conditional predictive ability. *Econometrica*, 74(6):1545–1578, 2006.

[2] Francis X. Diebold and Robert S. Mariano. Comparing predictive accuracy. *Journal of Business & Economic Statistics*, 13(3):253–263, 1995.

[3] Todd Clark and Michael McCracken. Testing for unconditional predictive ability. Working Paper 2010-031A, Federal Reserve Bank of St. Louis, 2010.

[4] Francis X. Diebold. Comparing predictive accuracy, twenty years later: A personal perspective on the use and abuse of diebold–mariano tests. *Journal of Business & Economic Statistics*, 33(1):1–1, 2015.

[5] Leonard J. Tashman. Out-of-sample tests of forecasting accuracy: an analysis and review. *International journal of forecasting*, 16(4):437–450, 2000.

[6] Christoph Bergmeir and José M. Benítez. On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191:192–213, 2012.

[7] J. Scott Armstrong and Michael C. Grohman. A comparative study of methods for long-range market forecasting. *Management Science*, 19(2):211–221, 1972.

[8] Kenneth D West. Asymptotic inference about predictive ability. *Econometrica: Journal of the Econometric Society*, pages 1067–1084, 1996.

[9] Todd Clark and Michael McCracken. Tests of equal predictive ability with real-time data. *Journal of Business & Economic Statistics*, 27(4):441–454, 2009.

[10] Atsushi Inoue and Lutz Kilian. In-sample or out-of-sample tests of predictability: Which one should we use? *Econometric Reviews*, 23(4):371–402, 2005.

[11] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802–808, 2018.

[12] Spyros Makridakis, Evangelos Spiliotis, Vassilios Assimakopoulos, Zhi Chen, Anil Gaba, Ilia Tsetlin, and Robert L. Winkler. The m5 uncertainty competition: Results, findings and conclusions. *International Journal of Forecasting*, pages 1–24, 2020.

[13] Chang Wei Tan, Christoph Bergmeir, François Petitjean, and Geoffrey I Webb. Time series extrinsic regression. *Data Mining and Knowledge Discovery*, 35(3): 1032–1060, 2021.

[14] Rob J. Hyndman. Terminology matters. https://robjhyndman.com/hyndsight/terminology-matters/, 2020. URL https://robjhyndman.com/hyndsight/terminology-matters/.

[15] Galit Shmueli. To explain or to predict? *Statistical science*, 25(3):289–310, 2010.

[16] Leo Breiman. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*, 16(3):199–231, 2001.

[17] Corrie Elston. Tensorflow machine learning with financial data on google cloud platform, 2016. URL https://cloud.google.com/blog/products/gcp/tensorflow-machine-learning-with-financial-data-on-google-cloud-p

[18] Dave Moore and Jacob Burnim. Structural time series modeling in tensorflow probability, 2019. URL https://blog.tensorflow.org/2019/03/structural-time-series-modeling-in.html.

[19] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[20] Peter J Brockwell and Richard A Davis. *Time series: Theory and Methods*. Springer Series in Statistics, 2nd edition, 1991.

[21] Christoph Bergmeir, Rob J. Hyndman, and Bonsoo Koo. A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis*, 120:70–83, 2018.

[22] Vitor Cerqueira, Luis Torgo, and Igor Mozetič. Evaluating time series forecasting models: An empirical study on performance estimation methods. *Machine Learning*, 109(11):1997–2028, 2020.

[23] Matthias Schnaubelt. A comparison of machine learning model validation schemes for non-stationary time series data. Technical report, FAU Discussion Papers in Economics, 2019.

[24] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020.

[25] Sylvain Arlot and Alain Celisse. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79, 2010.

[26] Richard A. Meese and Kenneth Rogoff. Empirical exchange rate models of the seventies: Do they fit out of sample? *Journal of International Economics*, 14(1):3–24, 1983. ISSN 0022-1996. doi: https://doi.org/10.1016/0022-1996(83)90017-X. URL https://www.sciencedirect.com/science/article/pii/002219968390017X.

[27] James H Stock and Mark W Watson. A comparison of linear and nonlinear univariate models for forecasting macroeconomic time series. Working paper 6607, National Bureau of Economic Research, 1998.

[28] Spyros Makridakis. Note—sliding simulation: A new approach to time series forecasting. *Management Science*, 36(4):505–512, 1990.

[29] Mervyn Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133, 1974.

[30] U.S. National Resources Committee. *Patterns of Resource Use*. U.S. Government Printing Office, (preliminary edition for technical criticism) edition, 1938.

[31] J. Scott Armstrong. *Long-Range Forecasting: From Crystal Ball to Computer*. Wiley-Interscience, second edition, 1985.

[32] Halbert White. A reality check for data snooping. *Econometrica*, 68(5):1097–1126, 2000.

[33] Edwin B. Wilson. The periodogram of american business activity. *The Quarterly Journal of Economics*, 48(3):375–417, 1934.

[34] Robert Ferber. Front matter to a study of aggregate consumption functions. In *A Study of Aggregate Consumption Functions*, pages 7–5. NBER, 1953.

[35] Robert Ferber. Are correlations any guide to predictive value? *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 5(2):113–121, 1956.

[36] Robert M Kirby. A comparison of short and medium range statistical forecasting methods. *Management Science*, 13(4):B–202, 1966.

[37] William H Williams and MLi Goodman. A simple method for the construction of empirical confidence limits for economic forecasts. *Journal of the American Statistical Association*, 66(336):752–754, 1971.

[38] Charles R. Nelson. The prediction performance of the frb-mit-penn model of the us economy. *The American Economic Review*, 62(5):902–917, 1972.

[39] Frank De Leeuw and Edward Gramlich. The federal reserve-mit economic model. *Federal Reserve Bulletin*, (Jan):11–40, 1968.

[40] Robert H Rasche and Harold T Shapiro. The frb-mit econometric model: its special features. *The American Economic Review*, 58(2):123–149, 1968.

[41] George EP Box and Gwilym M Jenkins. Time series analysis. forecasting and control. *Holden-Day Series in Time Series Analysis*, 1976.

[42] Ray C. Fair. Estimating the expected predictive accuracy of econometric models. *International Economic Review*, pages 355–378, 1980.

[43] Richard Ashley, Clive WJ Granger, and Richard Schmalensee. Advertising and aggregate consumption: An analysis of causality. *Econometrica: Journal of the Econometric Society*, pages 1149–1167, 1980.

[44] Urban Hjorth. Model selection and forward validation. *Scandinavian Journal of Statistics*, 9(6):95–105, 1982.

[45] Mervyn Stone. An asymptotic equivalence of choice of model by cross-validation and akaike's criterion. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):44–47, 1977.

[46] Hirotugu Akaike. Information theory and an extension of the maximum likelihood principle. In B. N. Petrov and F. Czáki, editors, *2nd International Symposium on Information Theory*, pages 267–281, 1973.

[47] Craig F Ansley and Robert Kohn. Efficient generalized cross-validation for state space models. *Biometrika*, 74(1):139–148, 1987.

[48] Piet De Jong. A cross-validation filter for time series models. *Biometrika*, 75(3): 594–600, 1988.

[49] Tom A.B. Snijders. On cross-validation for predictor evaluation in time series. In *On model uncertainty and its statistical implications*, pages 56–69. Springer, 1988.

[50] Prabir Burman. A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods. *Biometrika*, 76(3):503–514, 1989.

[51] Ping Zhang. Model selection via multifold cross validation. *The annals of statistics*, pages 299–313, 1993.

[52] Qiwei Yao and Howell Tong. On subset selection in non-parametric stochastic regression. *Statistica Sinica*, pages 51–70, 1994.

[53] Prabir Burman, Edmond Chow, and Deborah Nolan. A cross-validatory method for dependent data. *Biometrika*, 81(2):351–358, 1994.

[54] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[55] Jeff Racine. Consistent cross-validatory model-selection for dependent data: hv-block cross-validation. *Journal of Econometrics*, 99(1):39–61, 2000.

[56] Robert M. Kunst. Cross validation of prediction models for seasonal time series by parametric bootstrapping. *Austrian Journal of Statistics*, 37(3&4):271–284, 2008.

[57] J-I Fukuchi. Subsampling and model selection in time series analysis. *Biometrika*, 86(3):591–604, 1999. ISSN 0006-3444. doi: 10.1093/biomet/86.3.591. URL https://doi.org/10.1093/biomet/86.3.591.

[58] Yuichi Kitamura. Predictive inference and the bootstrap. *Yale University*, 2001.

[59] CR Rao and Y Wu. On model selection. In *Model selection*, pages 1–57. Institute of Mathematical Statistics, 2001.

[60] Todd Clark and Michael McCracken. Advances in forecast evaluation. *Handbook of economic forecasting*, 2:1107–1201, 2013.

[61] Jean Opsomer, Yuedong Wang, and Yuhong Yang. Nonparametric regression with correlated errors. *Statistical Science*, pages 134–153, 2001.

[62] Rob J. Hyndman. Why every statistician should know about cross-validation. https://robjhyndman.com/hyndsight/crossvalidation/, 2010. URL https://robjhyndman.com/hyndsight/crossvalidation/.

[63] Prabir Burman and Deborah Nolan. Data-dependent estimation of prediction functions. *Journal of Time Series Analysis*, 13(3):189–207, 1992.

[64] Michael McCracken. Robust out-of-sample inference. *Journal of Econometrics*, 99(2):195–223, 2000.

[65] Keisuke Hirano and Jonathan H Wright. Forecasting with model uncertainty: Representations and risk reduction. *Econometrica*, 85(2):617–643, 2017.

[66] Daniel J McDonald, Cosma Rohilla Shalizi, and Mark Schervish. Nonparametric risk bounds for time-series forecasting. *The Journal of Machine Learning Research*, 18(1):1044–1083, 2017.

[67] C.-K. Chu and J. S. Marron. Comparison of two bandwidth selectors with dependent errors. *The Annals of Statistics*, 19(4):1906–1918, 1991.

[68] Christoph Bergmeir, Mauro Costantini, and José M. Benítez. On the usefulness of cross-validation for directional forecast evaluation. *Computational Statistics & Data Analysis*, 76:132–143, 2014.

[69] Vitor Cerqueira, Luis Torgo, Jasmina Smailović, and Igor Mozetič. A comparative study of performance estimation methods for time series forecasting. In *2017 IEEE international conference on data science and advanced analytics (DSAA)*, pages 529–538. IEEE, 2017.

[70] Rob J. Hyndman and Yangzhuoran Yang. *tsdl: Time series Data Library. v0.1.0*. URL https://pkg.yangzhuoranyang./tsdl/.

[71] Benjamin Kedem and Konstantinos Fokianos. *Regression models for time series analysis*, volume 488. John Wiley & Sons, 2005.

[72] Konstantinos Fokianos and Benjamin Kedem. Partial likelihood inference for time series following generalized linear models. *Journal of Time Series Analysis*, 25(2): 173–197, 2004.

[73] George Casella and Roger L. Berger. *Statistical inference*. Cengage Learning, 2021.

[74] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[75] Floris Takens. Detecting strange attractors in turbulence. In *Dynamical systems and turbulence, Warwick 1980*, pages 366–381. Springer, 1981.

[76] Luc Devroye and Terry Wagner. Distribution-free inequalities for the deleted and holdout error estimates. *IEEE Transactions on Information Theory*, 25(2):202–207, 1979.

[77] Luc Devroye and Terry Wagner. Distribution-free performance bounds for potential function rules. *IEEE Transactions on Information Theory*, 25(5):601–604, 1979.

[78] A Philip Dawid. Present position and potential developments: Some personal views statistical theory the prequential approach. *Journal of the Royal Statistical Society: Series A (General)*, 147(2):278–290, 1984.

[79] D Allen. The relationship between variable selection and data augmentation and slow feature analysis. *Technometrics*, 16:125–127, 1974.

[80] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[81] Allan D.R. McQuarrie and Chih-Ling Tsai. *Regression and time series model selection*. World Scientific, 1998.

[82] Jude H Kastens. Small sample behaviors of the delete-d cross validation statistic. *Open Journal of Statistics*, 5(05):382, 2015.

[83] Michael Nelson, Tim Hill, William Remus, and Marcus O'Connor. Time series forecasting using neural networks: Should the data be deseasonalized first? *Journal of forecasting*, 18(5):359–367, 1999.

[84] G Peter Zhang and Min Qi. Neural network forecasting for seasonal and trend time series. *European journal of operational research*, 160(2):501–514, 2005.

[85] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. Statistical and machine learning forecasting methods: Concerns and ways forward. *PloS one*, 13(3):e0194889, 2018.

[86] Ben Taieb Souhaib, Gianluca Bontempi, Amir F. Atiya, and Antti Sorjamaa. A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. *Expert Systems with Applications*, 39(8):7067–7083, 2012. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2012.01.039. URL https://www.sciencedirect.com/science/article/pii/S0957417412000528.

[87] Vitor Cerqueira, Luis Torgo, and Carlos Soares. Machine learning vs statistical methods for time series forecasting: Size matters. *arXiv preprint arXiv:1909.13316*, 2019.

[88] Matthew B Kennel, Reggie Brown, and Henry DI Abarbanel. Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Physical review A*, 45(6):3403, 1992.

[89] Carl Rhodes and Manfred Morari. The false nearest neighbors algorithm: An overview. *Computers & Chemical Engineering*, 21:S1149–S1154, 1997.

[90] Guy Nason. A test for second-order stationarity and approximate confidence intervals for localized autocovariances for locally stationary time series. *Journal of the Royal Statistical Society: SERIES B: Statistical Methodology*, pages 879–904, 2013.

[91] Max Kuhn, Steve Weston, Chris Keefer, Nathan Coulter, and Ross Quinlan. *Cubist: Rule- And Instance-Based Regression Modeling. Version 0.3.0*. R Foundation for Statistical Computing. URL https://cran.r-project.org/package=Cubist.

[92] John Ross Quinlan. Combining instance-based and model-based learning. In *Proceedings of the tenth international conference on machine learning*, pages 236–243, 1993.

[93] John Ross Quinlan. Learning with continuous classes. In *5th Australian joint conference on artificial intelligence*, volume 92, pages 343–348. World Scientific, 1992.

[94] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[95] A Etemad-Shahidi and Javad Mahjoobi. Comparison between m5' model tree and neural networks for prediction of significant wave height in lake superior. *Ocean Engineering*, 36(15-16):1175–1181, 2009.

[96] Max Kuhn. Modern rule-based models, May 2020. URL https://rviews.rstudio.com/2020/05/21/modern-rule-based-models/.

[97] Rob J. Hyndman. Another look at forecast-accuracy metrics for intermittent demand. *Foresight: The International Journal of Applied Forecasting*, 4(4):43–46, 2006.

[98] Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *The Journal of Machine Learning Research*, 18(1):2653–2688, 2017.

[99] Max Kuhn and Kjell Johnson. Who's afraid of the big black box?: Statisticians' vital role in big data and predictive modelling. *Significance*, 11(3):35–37, 2014. doi: https://doi.org/10.1111/j.1740-9713.2014.00753.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1740-9713.2014.00753.x.

[100] Frank E. Harrell Jr. *Regression modeling strategies: with applications to linear models, logistic and ordinal regression, and survival analysis*. springer, 2015.

[101] Agbolade Omowole. Research shows ai is often biased. here's how to make algorithms work for all of us, Jul 2021. URL https://www.weforum.org/agenda/2021/07/ai-machine-learning-bias-discrimination/.

[102] Adrienne Yapo and Joseph Weiss. Ethical implications of bias in machine learning. In *Proceedings of the 51st Hawaii International Conference on System Sciences (HICSS 2018)*, pages 5365–5372. Association for Information Systems.

[103] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, 54(6):1–35, 2021.

[104] Gianluca Bontempi. Machine learning strategies for time series prediction. https://www.researchgate.net/profile/Gianluca-Bontempi/publication/304627726_Machine_learning_strategies_for_time_series_forecasting/links/577568a008aead7ba06ffe7a/Machine-learning-strategies-for-time-series-forecasting.pdf, 2013. Accessed: 2021–02-21.

[105] Spyros Makridakis and Michele Hibon. The m3-competition: results, conclusions and implications. *International journal of forecasting*, 16(4):451–476, 2000.