

Malaria vaccine antigen identification for *Plasmodium vivax*

Renee Ti Chou

2023-06-06

Contents

Preface	1
1 Data engineering	2
1.1 purf package installization	2
1.2 Retrieving <i>P. vivax</i> protein variables	2
1.2.1 Immunological data sets	5
1.2.2 Proteomic data sets	7
1.2.3 Structural data sets	9
1.2.4 Genomic data sets	10
1.2.5 Final assembly	11
1.3 Generating ML input	11
2 Model training	13
2.1 Hyper-parameter tuning for PURF	13
2.1.1 Analysis	13
2.1.2 Plotting	18
2.1.3 Results for positive level = 0.5	28
3 Model evaluation	33
3.1 Known antigen prediction accuracy	33
3.1.1 Analysis	33
3.2 Adversarial controls	38
3.2.1 Analysis	38
3.2.2 Plotting	43
4 Model comparisons	53
4.1 Known antigen score comparisons	53
4.2 Known antigen prediction summary	57
4.3 Score and species association	62
4.4 Tree depth analysis	66
4.4.1 Analysis	66
4.4.2 Plotting	67
5 Model interpretation	78
5.1 Proximity matrix calculation	78
5.1.1 Analysis	78
5.1.2 Plotting	81
5.2 Clustering of predicted positives	85
5.2.1 Analysis	85

5.3	Comparison of amino acid frequencies between species	96
5.3.1	Analysis	96
5.4	Variable importance	99
5.4.1	Analysis	99
5.4.2	Plotting	106
5.4.3	Comparison of top 10 important variables	117
6	Top candidate comparisons	124
6.1	Clustering analysis	124
6.1.1	Analysis	124
6.2	Gene ontology analysis	130
6.2.1	Analysis	130
6.2.2	Plotting	130
6.3	Candidate antigen characterization	134
6.3.1	Processing table	134
7	Summary of candidates	138

Preface

Our research objective is to identify potential vaccine antigen candidates targeting the parasite *P. vivax*, the second most prevalent cause of malaria. To improve the performance of the autologous model for *P. vivax*, which has a constrained size of proteome and a small set of labeled antigens, we leverage heterologous data from *P. falciparum*. We utilized multiple models trained on various combinations of heterologous and autologous data using the positive-unlabeled random forest (PURF) algorithm. The research notebook contains both data and code generated in the study titled “*Plasmodium vivax* antigen candidate prediction improves with the addition of *Plasmodium falciparum* data.” Further, the notebook provides guidance on extracting protein variables and assembling machine learning input from the database, along with code for conducting experimental analyses and creating plots.

The notebook is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Section 1

Data engineering

1.1 **purf** package installization

See instructions on the GitHub webpage¹.

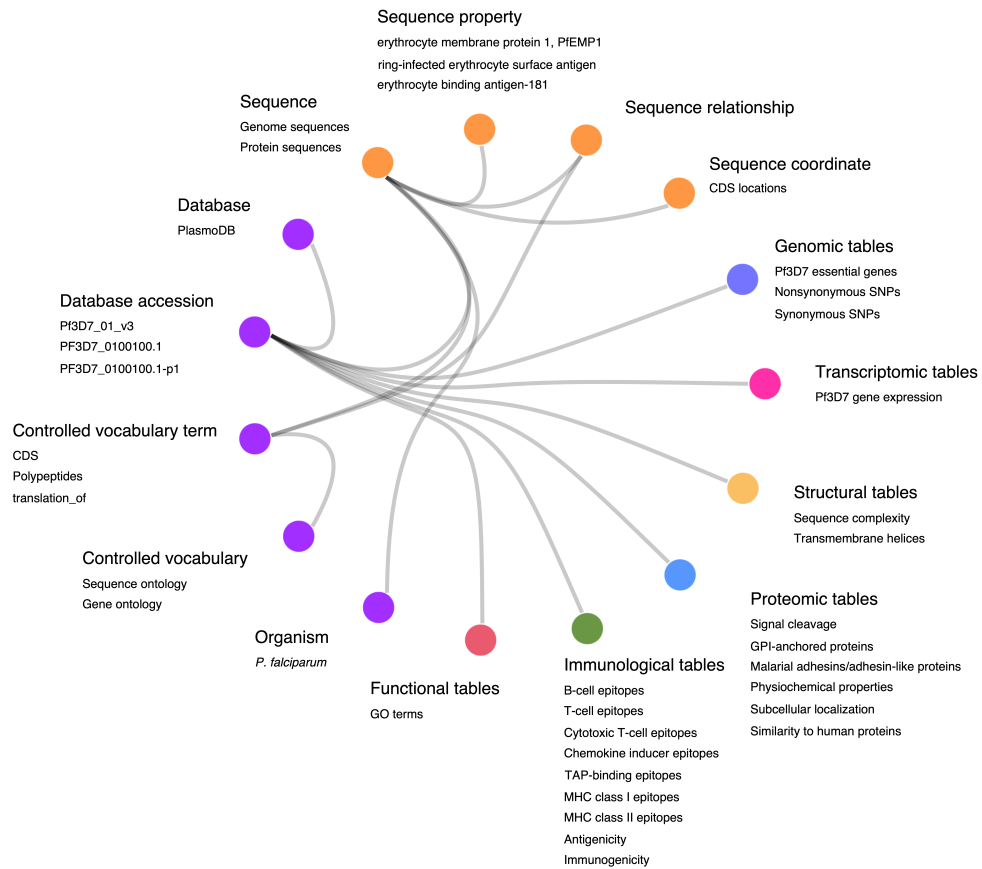
1.2 Retrieving *P. vivax* protein variables

In Bash:

```
echo "create database pfpv_reverse_vaccinology" | mariadb -u <dbuser> -p  
mariadb -u <dbuser> -p pfpv_reverse_vaccinology < ./other_data/pfpv_reverse_vaccinology.sql
```

Overall database schema:

¹<https://github.com/rtchou/purf>

**In R:**

```
library(RMariaDB)
library(DBI)
library(rlist)
```

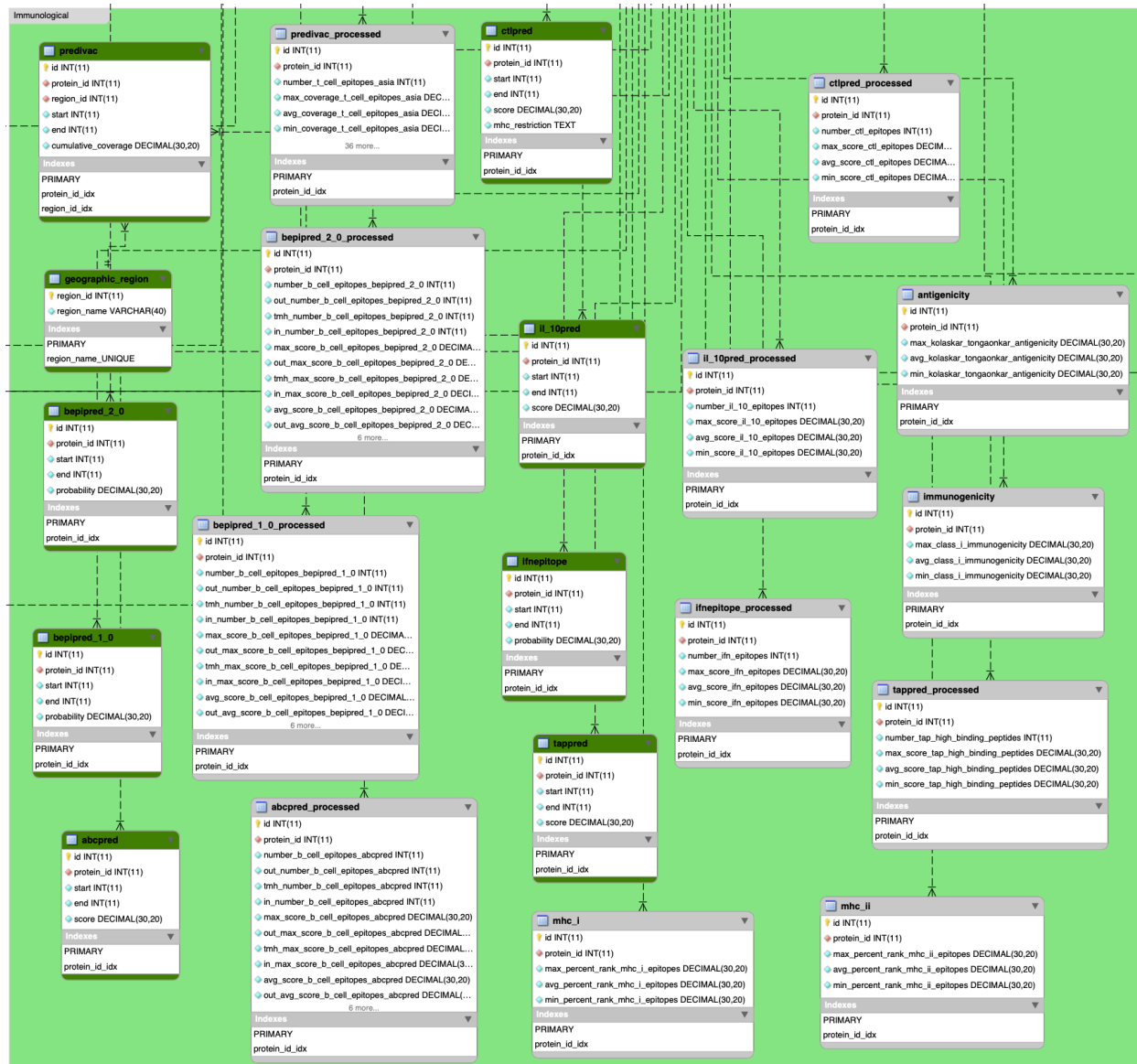
```
output_path <- "./other_data/pv_assembled_data.csv"
db <- dbConnect(RMariaDB::MariaDB(), user = "root", password = "", dbname = "pfpv_reverse_vaccinology")

get_data <- function(table_name, db) {
  sql <- sqlInterpolate(db, "SELECT * FROM ?table", table = dbQuoteIdentifier(db, table_name))
  return(subset(dbGetQuery(db, sql), select = -c(id)))
}

# protein ID and transcript ID map table
```

```
id_map <- dbGetQuery(db, 'SELECT dbxref_1.accession AS accession, dbxref_1.dbxref_id AS protein_id
↪ FROM dbxref AS dbxref_1
                        LEFT OUTER JOIN feature AS feature_1 ON dbxref_1.dbxref_id = feature_1.dbxref_id
                        LEFT OUTER JOIN cvterm ON feature_1.type_id = cvterm.cvterm_id
                        LEFT OUTER JOIN feature_relationship ON feature_1.feature_id =
↪ feature_relationship.subject_id
                        LEFT OUTER JOIN feature AS feature_2 ON feature_relationship.object_id =
↪ feature_2.feature_id
                        LEFT OUTER JOIN dbxref AS dbxref_2 ON feature_2.dbxref_id = dbxref_2.dbxref_id
                        LEFT OUTER JOIN organism ON feature_1.organism_id = organism.organism_id
                        WHERE feature_1.is_obsolete = 0 AND cvterm.name = "polypeptide" AND
↪ organism.species="vivax_P01")
```

1.2.1 Immunological data sets



In R:

```

immu <- list()

# predivac_processed result for T-cell epitopes
immu <- list.append(immu, get_data("predivac_processed", db))

```

```
# bepiped_2_0 result for B-cell epitopes
immu <- list.append(immu, get_data("bepiped_2_0_processed", db))

# bepiped_1_0 result for B-cell epitopes
immu <- list.append(immu, get_data("bepiped_1_0_processed", db))

# abcpred result for B-cell epitopes
immu <- list.append(immu, get_data("abcpred_processed", db))

# ctlpred result for cytotoxic T-cell epitopes
immu <- list.append(immu, get_data("ctlpred_processed", db))

# il_10pred result for interleukine-10 inducing epitopes
immu <- list.append(immu, get_data("il_10pred_processed", db))

# ifnepitope result for IFN-gamma inducing epitopes
immu <- list.append(immu, get_data("ifnepitope_processed", db))

# tappred for high binding affinity epitopes toward the TAP transporter
immu <- list.append(immu, get_data("tappred_processed", db))

# mhc_i for MHC class I epitopes
immu <- list.append(immu, get_data("mhc_i", db))

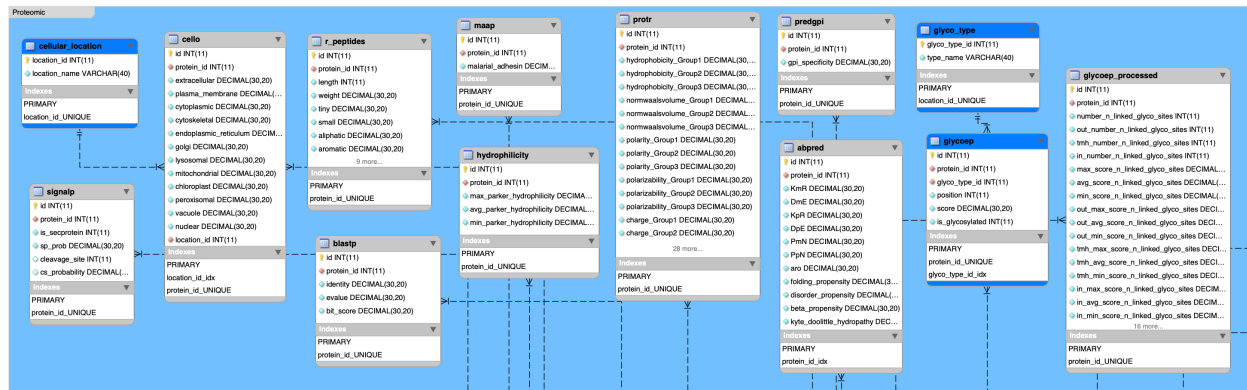
# mhc_ii for MHC class II epitopes
immu <- list.append(immu, get_data("mhc_ii", db))

# antigenicity
immu <- list.append(immu, get_data("antigenicity", db))

# immunogenicity
immu <- list.append(immu, get_data("immunogenicity", db))

# merge immunological data sets
immunological_ds <- id_map["protein_id"]
for (ds in immu) {
  immunological_ds <- merge(x = immunological_ds, y = ds, by = "protein_id", all.x = TRUE)
}
```

1.2.2 Proteomic data sets



In R:

```
pro <- list()

# cello result for subcellular localization
pro <- list.append(pro, subset(get_data("cello", db), select = -c(location_id)))

# maap result for malaria adhesin/adhesin-like proteins
pro <- list.append(pro, get_data("maap", db))

# peptides result for physicochemical properties
pro <- list.append(pro, get_data("r_peptides", db))

# protr result for physicochemical properties
pro <- list.append(pro, get_data("protr", db))

# hydrophilicity
pro <- list.append(pro, get_data("hydrophilicity", db))

# predgpi result for gpi anchors prediction
pro <- list.append(pro, get_data("predgpi", db))

# signalp result for signal cleavage prediction
pro <- list.append(pro, subset(get_data("signalp", db), select = -c(is_secprotein, cleavage_site,
  ↪ cs_probability)))

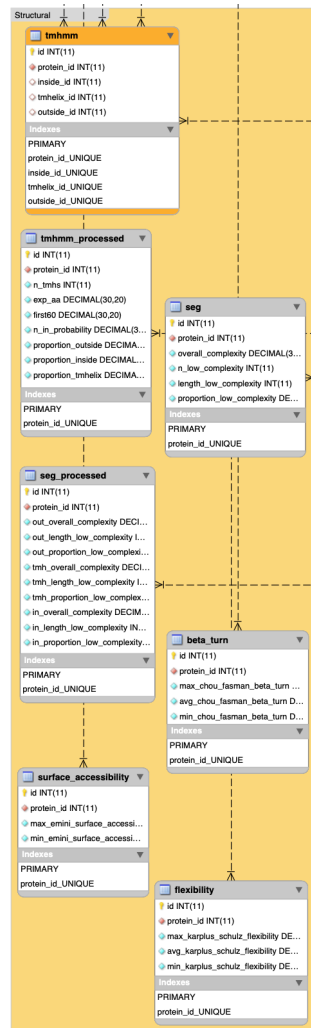
# abpred results for amino acid compositions and other variables
pro <- list.append(pro, get_data("abpred", db))
```

```
# glycoep results for N-linked and O-linked glycosylation
pro <- list.append(pro, get_data("glycoep_processed", db))

# blastp result for similarity to human proteins
pro <- list.append(pro, get_data("blastp", db))

# merge proteomic data sets
proteomic_ds <- id_map["protein_id"]
for (ds in pro) {
  proteomic_ds <- merge(x = proteomic_ds, y = ds, by = "protein_id", all.x = TRUE)
}
```

1.2.3 Structural data sets



In R:

```
struc <- list()

# tmhmm result for transmembrane helices prediction
struc <- list.append(struc, get_data("tmhmm_processed", db))

# seg result for sequence complexity
struc <- list.append(struc, get_data("seg", db))
```



```

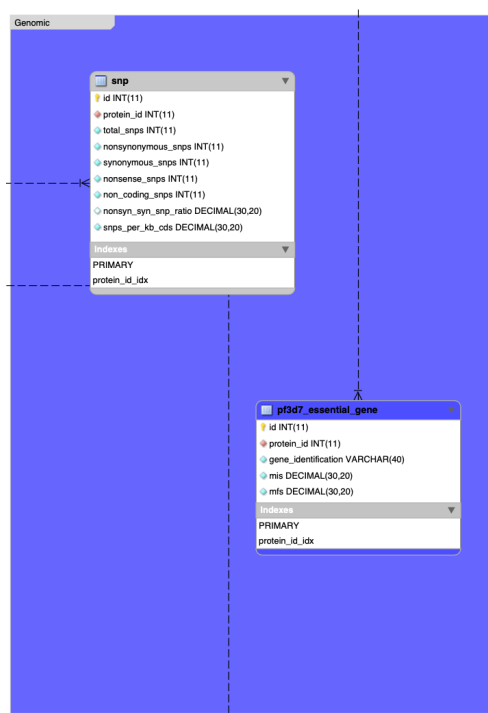
# seg + tmhmm result
struc <- list.append(struc, get_data("seg_processed", db))

# b_cell_epitope_methods for structural predictions
struc <- list.append(struc, get_data("beta_turn", db))
struc <- list.append(struc, get_data("surface_accessibility", db))
struc <- list.append(struc, get_data("flexibility", db))

# merge structural analysis data sets
structural_ds <- id_map["protein_id"]
for (ds in struc) {
  structural_ds <- merge(x = structural_ds, y = ds, by = "protein_id", all.x = TRUE)
}

```

1.2.4 Genomic data sets



In R:

```
geno <- list()
```

```
# snp result
geno <- list.append(geno, get_data("snp", db))

# merge structural analysis data sets
genomic_ds <- id_map["protein_id"]
for (ds in geno) {
  genomic_ds <- merge(x = genomic_ds, y = ds, by = "protein_id", all.x = TRUE)
}
```

1.2.5 Final assembly

In R:

```
# prepare predictor variables
data <- merge(x = immunological_ds, y = proteomic_ds, by = "protein_id", all = FALSE)
data <- merge(x = data, y = structural_ds, by = "protein_id", all = FALSE)
data <- merge(x = data, y = genomic_ds, by = "protein_id", all = FALSE)
data <- merge(x = id_map, y = data, by = "protein_id", all.x = TRUE)
accession <- data$accession
data <- subset(data, select = -c(protein_id, accession))
rownames(data) <- accession

# write to output
write.csv(data, output_path)

# close database connection
dbDisconnect(db)
```

1.3 Generating ML input

In R:

```
response_var <- read.csv("./other_data/pv_antigen_labels.csv", row.names = 1)
predictor_vars <- read.csv("./other_data/pv_assembled_data.csv", row.names = 1)
ml_input <- merge(x = response_var, y = predictor_vars, by = "row.names", all = TRUE)
row_names <- ml_input$Row.names
ml_input <- subset(ml_input, select = -c(Row.names))
```

```
rownames(ml_input) <- row_names
```

```
# write to output
```

```
write.csv(ml_input, "./data/supplementary_data_2_pv_ml_input.csv")
```

```
sessionInfo()
```

```
## R version 4.2.3 (2023-03-15)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur ... 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRlapack.dylib
##
## locale:
##  [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats      graphics  grDevices  utils      datasets  methods   base
##
## other attached packages:
## [1] rlist_0.4.6.2  DBI_1.1.3      RMariaDB_1.2.2
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.10      rstudioapi_0.14  knitr_1.42       magrittr_2.0.3
##  [5] hms_1.1.3        bit_4.0.5        R.cache_0.16.0   rlang_1.1.1
##  [9] fastmap_1.1.1    styler_1.9.1     tools_4.2.3      data.table_1.14.8
## [13] xfun_0.39        R.oo_1.25.0      cli_3.6.1        htmltools_0.5.5
## [17] yaml_2.3.7       bit64_4.0.5      digest_0.6.31    lifecycle_1.0.3
## [21] bookdown_0.34    purrr_1.0.1      vctrs_0.6.2      R.utils_2.12.2
## [25] codetools_0.2-19 glue_1.6.2       evaluate_0.21    rmarkdown_2.21
## [29] compiler_4.2.3   R.methodsS3_1.8.2 pkgconfig_2.0.3
```

Section 2

Model training

2.1 Hyper-parameter tuning for PURF

2.1.1 Analysis

In Python:

```
library(reticulate)
use_condaenv("/Users/renee/Library/r-miniconda/envs/purf/bin/python")
```

```
import pandas as pd
from sklearn.utils import shuffle
import numpy as np
import pickle
from sklearn.impute import SimpleImputer
from purf.pu_ensemble import PURandomForestClassifier
from sklearn.preprocessing import MinMaxScaler
from scipy.spatial import distance
from joblib import Parallel, delayed
from sklearn.ensemble._forest import _generate_unsampled_indices
import os
import re
import session_info
```

```
# private function for train_purf()
def _get_ref_antigen_stats(idx, tree, X, y, ref_indices, max_samples=None):
    if max_samples is None:
        max_samples = y.shape[0]
    oob_indices = _generate_unsampled_indices(tree.random_state, y.shape[0], max_samples)
    ref_oob = [i in oob_indices for i in ref_indices]
    ref_pred = list()
    pred = tree.predict_proba(X[ref_indices,:], check_input=False)
```

```

    ref_pred = pred[:,1]
    return ref_oob, ref_pred

def train_purf(features, outcome, pos_level=0.5, purf=None, tree_filtering=False, ref_antigens=None,
    ↪ n_jobs=1):
    features, outcome = shuffle(features, outcome, random_state=0)
    # Imputation
    imputer = SimpleImputer(strategy='median')
    X = imputer.fit_transform(features)
    X = pd.DataFrame(X, index=features.index, columns=features.columns)
    y = outcome
    features = X
    print('There are %d positives out of %d samples before feature space weighting.' % (sum(y), len(y)))
    # Feature space weighting
    lab_pos = X.loc[y==1,:]
    median = np.median(lab_pos, axis=0)
    # Feature space weighting
    lab_pos = X.loc[y==1,:]
    median = np.median(lab_pos, axis=0)
    scaler = MinMaxScaler(feature_range=(1,10))
    dist = list()
    for i in range(lab_pos.shape[0]):
        dist.append(distance.euclidean(lab_pos.iloc[i, :], median))
    dist = np.asarray(dist).reshape(-1, 1)
    counts = np.round(scaler.fit_transform(dist))
    counts = np.array(counts, dtype=np.int64)[: , 0]
    X_temp = X.iloc[y==1, :]
    X = X.iloc[y==0, :]
    y = np.asarray([0] * X.shape[0] + [1] * (sum(counts)))
    appended_data = [X]
    for i in range(len(counts)):
        appended_data.append(pd.concat([X_temp.iloc[[i]] * counts[i]))
    X = pd.concat(appended_data)
    print('There are %d positives out of %d samples after feature space weighting.' % (sum(y), len(y)))
    res = pd.DataFrame({'protein_id': X.index, 'antigen_label' : y})
    if tree_filtering is True:
        # get ref antigen indices
        ref_index_dict = {ref:list() for ref in list(ref_antigens.values())}
        for i in range(res.shape[0]):
            if res['protein_id'][i] in list(ref_antigens.values()):
                ref_index_dict[res['protein_id'][i]].append(res.index[i])

```

```

ref_indices = sum(ref_index_dict.values(), [])
# get OOB stats and predictions
X = X.astype('float32')
trees = purf.estimated_
idx_list = [i for i in range(len(trees))]
stats_res = Parallel(n_jobs=n_jobs)(
    delayed(_get_ref_antigen_stats)(idx, trees[idx], np.array(X), y, ref_indices) for
↪ idx in idx_list)
# ref_oob data structure:
# rows represent individual trees
# column represent reference antigens
# cells indicate whether the reference antigen is in the OOB samples of the tree
ref_oob = np.array([ref_oob for ref_oob, ref_pred in stats_res])
# ref_pred data structure:
# rows represent individual trees
# column represent reference antigens
# cells indicate the prediction of the reference antigen by the tree
ref_pred = np.array([ref_pred for ref_oob, ref_pred in stats_res])
# analyze duplicated reference antigens as a group
cumsum_num_ref = np.cumsum(np.array([len(v) for k,v in ref_index_dict.items()]))
ref_oob_all = np.array([ref_oob[:, 0:cumsum_num_ref[i]].any(axis=1) if i == 0 else \
    ref_oob[:, cumsum_num_ref[i - 1]:cumsum_num_ref[i]].any(axis=1) \
    for i in range(len(ref_antigens))]).T
ref_pred_all = np.array([ref_pred[:, 0:cumsum_num_ref[i]].any(axis=1) if i == 0 else \
    ref_pred[:, cumsum_num_ref[i - 1]:cumsum_num_ref[i]].sum(axis=1) \
    for i in range(len(ref_antigens))]).T
# calculate number of reference antigens as OOB samples for each tree
oob_total = ref_oob_all.sum(axis=1)
# assign score of 1 to trees that correctly predict all OOB reference antigens; otherwise,
↪ assign 0 score
weights = np.zeros(len(trees))
# iterate through the trees and calculate the stats
for i in range(len(trees)):
    oob_list = list(ref_oob_all[i,:])
    pred_list = list(ref_pred_all[i,:])
    if oob_total[i] == 0:
        weights[i] = 0
    else:
        if sum(np.array(pred_list)[oob_list] != 0) == oob_total[i]:
            weights[i] = 1
if tree_filtering is False:

```

```

# Training PURF
purf = PURandomForestClassifier(
    n_estimators = 100000,
    oob_score = True,
    n_jobs = 64,
    random_state = 42,
    pos_level = pos_level
)
purf.fit(X, y)
else:
    purf._set_oob_score_with_weights(np.array(X), y.reshape(-1,1), weights=weights)
# Storing results
res['OoB score'] = purf.oob_decision_function_[:,1]
res = features.merge(res.groupby('protein_id').mean(), left_index=True, right_on='protein_id')
res = res[['antigen_label', 'OoB score']]
if tree_filtering is False:
    return (purf, res)
else:
    return ({'model': purf, 'weights': weights}, res)

```

2.1.1.1 Pv data set

```

ref_antigens = {'CSP': 'PVP01_0835600.1-p1', 'DBP': 'PVP01_0623800.1-p1', 'MSP1':
    ↪ 'PVP01_0728900.1-p1'}
data = pd.read_csv('./other_data/pv_ml_input.csv', index_col=0)
features = data.iloc[:, 1:]
outcome = np.array(data.antigen_label)

for pos_level in [0.5, 0.4, 0.6, 0.3, 0.7, 0.2, 0.8, 0.1, 0.9]:
    (purf, res) = train_purf(features, outcome, pos_level=pos_level, tree_filtering=False)
    (purf_filtered, res_filtered) = train_purf(features, outcome, pos_level=pos_level, purf=purf,
    ↪ tree_filtering=True, ref_antigens=ref_antigens)
    res['OoB score filtered'] = res_filtered['OoB score']
    res.to_csv('~Downloads/pv_pos_level/%.1f_res.tsv' % pos_level)
    with open('~Downloads/pv_pos_level/%.1f_purf_tree_filtering.pkl' % pos_level, 'wb') as out:
        pickle.dump(purf_filtered, out, pickle.HIGHEST_PROTOCOL)

```

```

dir = '~/Downloads/pv_pos_level/'
files = os.listdir(dir)
tmp = pd.read_csv(dir + '0.1_res.tsv', sep='\t', index_col=0)['antigen_label']

data_frames = [pd.read_csv(dir + '%.1f_res.tsv' % pos_level, sep='\t', index_col=0)['00B score
↳ filtered'] for pos_level in np.arange(0.1, 1, 0.1)]
merged_df = pd.concat([tmp] + data_frames, join='outer', axis=1)
colnames = ['antigen_label'] + ['%.1f' % pos_level for pos_level in np.arange(0.1, 1, 0.1)]
merged_df.columns = colnames
merged_df.to_csv('./other_data/pv_pos_level_parameter_tuning.csv')

```

2.1.1.2 Pv + Pf combined data set

```

ref_antigens = {'CSP (Pf)': 'PF3D7_0304600.1-p1', 'RH5 (Pf)': 'PF3D7_0424100.1-p1', 'MSP5 (Pf)':
↳ 'PF3D7_0206900.1-p1',
                'P230 (Pf)': 'PF3D7_0209000.1-p1',
                'CSP (Pv)': 'PVP01_0835600.1-p1', 'DBP (Pv)': 'PVP01_0623800.1-p1', 'MSP1 (Pv)':
↳ 'PVP01_0728900.1-p1'}

data = pd.read_csv('./data/supplementary_data_4_pfpv_ml_input.csv', index_col=0)
features = data.iloc[:, 1:]
outcome = np.array(data.antigen_label)

for pos_level in [0.5, 0.4, 0.6, 0.3, 0.7, 0.2, 0.8, 0.1, 0.9]:
    (purf, res) = train_purf(features, outcome, pos_level=pos_level, tree_filtering=False)
    (purf_filtered, res_filtered) = train_purf(features, outcome, pos_level=pos_level, purf=purf,
↳ tree_filtering=True, ref_antigens=ref_antigens)
    res['00B score filtered'] = res_filtered['00B score']
    res.to_csv '~/Downloads/pfpv_pos_level/%.1f_res.tsv' % pos_level)
    with open '~/Downloads/pfpv_pos_level/%.1f_purf_tree_filtering.pkl' % pos_level, 'wb') as out:
        pickle.dump(purf_filtered, out, pickle.HIGHEST_PROTOCOL)

```

```

dir = '~/Downloads/pfpv_pos_level/'
files = os.listdir(dir)
tmp = pd.read_csv(dir + '0.1_res.tsv', sep='\t', index_col=0)['antigen_label']

data_frames = [pd.read_csv(dir + '%.1f_res.tsv' % pos_level, sep='\t', index_col=0)['00B score
↳ filtered'] for pos_level in np.arange(0.1, 1, 0.1)]
merged_df = pd.concat([tmp] + data_frames, join='outer', axis=1)

```



```
colnames = ['antigen_label'] + ['%.1f' % pos_level for pos_level in np.arange(0.1, 1, 0.1)]
merged_df.columns = colnames
merged_df.to_csv('./other_data/pfpv_pos_level_parameter_tuning.csv')
```

2.1.2 Plotting

In R:

```
library(mixR)
library(pracma)
library(rlist)
library(ggplot2)
library(cowplot)
library(grid)
library(gridExtra)
```

2.1.2.1 Pv data set

```
data <- read.csv("./other_data/pv_pos_level_parameter_tuning.csv", header = TRUE, row.names = 1,
  ↪ check.names = FALSE)
# Extract data with only unlabeled proteins
data_unl <- data[data$antigen_label == 0, ]
```

```
plot_list <- list()
plot_list2 <- list()
for (i in seq(0.1, 0.9, 0.1)) {
  fit <- mixfit(data_unl[[as.character(i)]], ncomp = 2)

  # Calculate receiver operating characteristic (ROC) curve
  # for putative positive and negative samples
  x <- seq(-0.5, 1.5, by = 0.01)
  neg_cum <- pnorm(x, mean = fit$mu[1], sd = fit$sd[1])
  pos_cum <- pnorm(x, mean = fit$mu[2], sd = fit$sd[2])
  fpr <- (1 - neg_cum) / ((1 - neg_cum) + neg_cum) # false positive / (false positive + true negative)
  tpr <- (1 - pos_cum) / ((1 - pos_cum) + pos_cum) # true positive / (true positive + false negative)

  p <- plot(fit, title = paste0(
```

```

    "Positive level = ", i,
    " (AUROC = ", round(trapz(-fpr, tpr), 2), ")"
  )) +
  scale_fill_manual(values = c("#0080FF", "#FF007F"), labels = c("Putative negative", "Putative
    ↪ positive")) +
  theme_bw() +
  {
    if (i == 0.1) {
      theme(
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.title = element_blank(),
        axis.text = element_text(colour = "black"),
        plot.title = element_text(hjust = 0.5, colour = "black"),
        legend.title = element_blank(),
        legend.text = element_text(colour = "black"),
        legend.position = c(0.7, 0.85),
        legend.background = element_blank()
      )
    } else {
      theme(
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.title = element_blank(),
        axis.text = element_text(colour = "black"),
        plot.title = element_text(hjust = 0.5, colour = "black"),
        legend.position = "none"
      )
    }
  } +
  xlim(-0.8, 1.5)

# Calculate percent rank for labeled positives
data_ <- data[c("antigen_label", as.character(i))]
colnames(data_) <- c("antigen_label", "O0B score")
data_$percent_rank <- rank(data_[["O0B score"]]) / nrow(data)
data_ <- data_[data$antigen_label == 1, ]
data_ <- data_[order(-data_$percent_rank), ]
data_$x <- 1:nrow(data_) / nrow(data_)
cat(paste0("EPR: ", sum(data_$`O0B score` >= 0.5) / nrow(data_), "\n"))

```

```

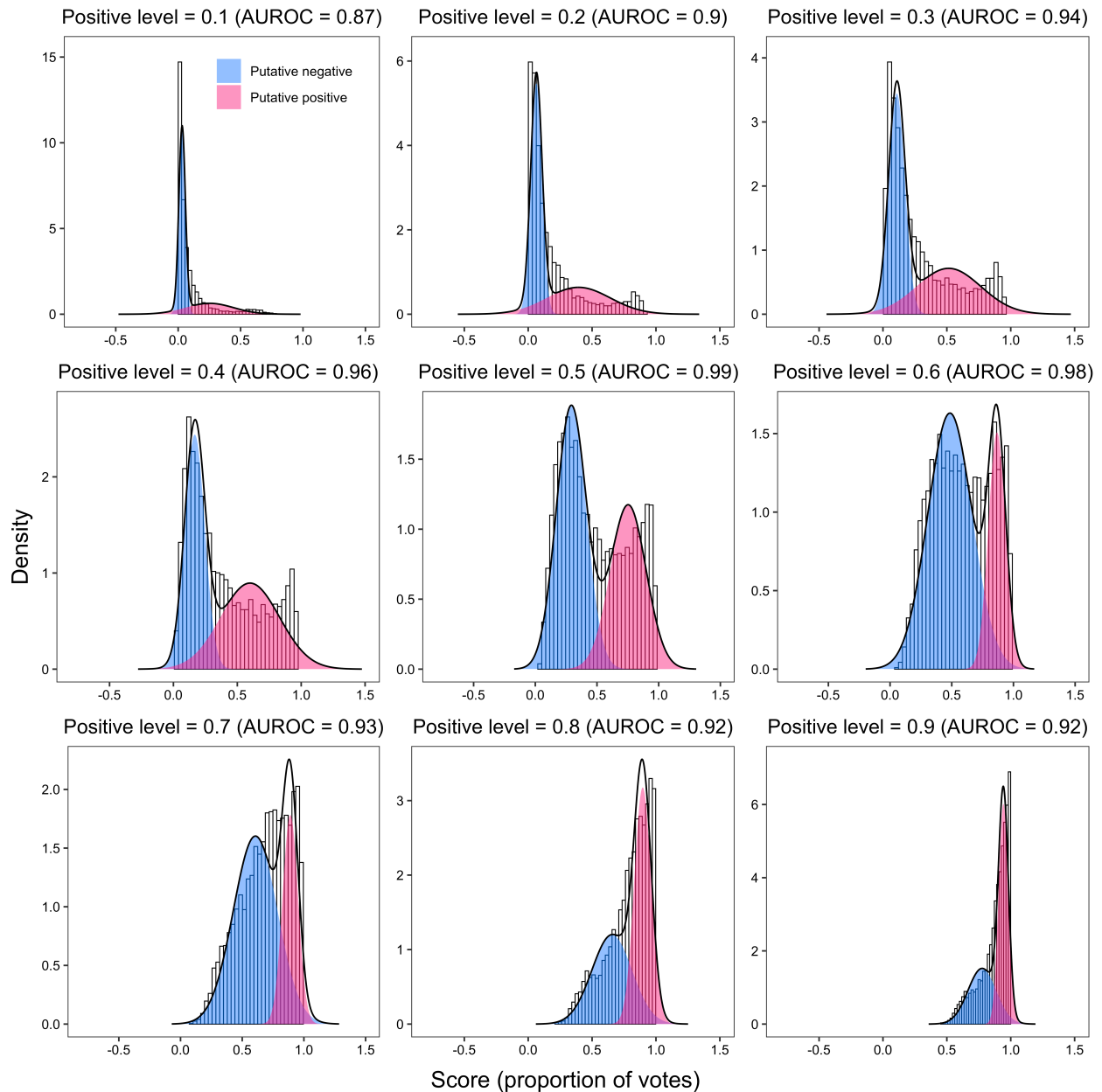
p2 <- ggplot(data_, aes(x = x, y = `percent_rank`)) +
  geom_hline(yintercept = 0.5, linetype = "dashed", color = "grey50") +
  geom_line(linewidth = 0.1, color = "grey30") +
  geom_point(aes(fill = `00B score`), size = 2.2, shape = 21, color = "grey30", stroke = 0.1) +
  scale_fill_gradient2(low = "#0080FF", mid = "white", high = "#FF007F", midpoint = 0.5, limits =
    ↪ c(0, 1)) +
  theme_bw() +
  {
    if (i == 0.1) {
      theme(
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.title = element_blank(),
        axis.text = element_text(colour = "black"),
        plot.title = element_text(hjust = 0.5, colour = "black"),
        legend.title = element_text(hjust = 0.5, colour = "black", angle = 0),
        legend.text = element_text(colour = "black"),
        legend.position = c(0.5, 0.15),
        legend.background = element_blank(),
        legend.title.align = -130
      )
    } else {
      theme(
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.title = element_blank(),
        axis.text = element_text(colour = "black"),
        plot.title = element_text(hjust = 0.5, colour = "black"),
        legend.position = "none"
      )
    }
  } +
  {
    if (i == 0.1) {
      guides(fill = guide_colourbar(title.position = "top", direction = "horizontal"))
    }
  } +
  {
    if (i == 0.1) {
      labs(fill = "Score (proportion of votes)")
    }
  }

```

```
  } +  
  ggtitle(paste0(  
    "Positive level = ", i,  
    " (AUC = ", round(trapz(c(0, data_$x, 1), c(1, data_$percent_rank, 0)), 2), ")")  
  )) +  
  ylim(0, 1)  
  
plot_list <- list.append(plot_list, p)  
plot_list2 <- list.append(plot_list2, p2)  
}
```

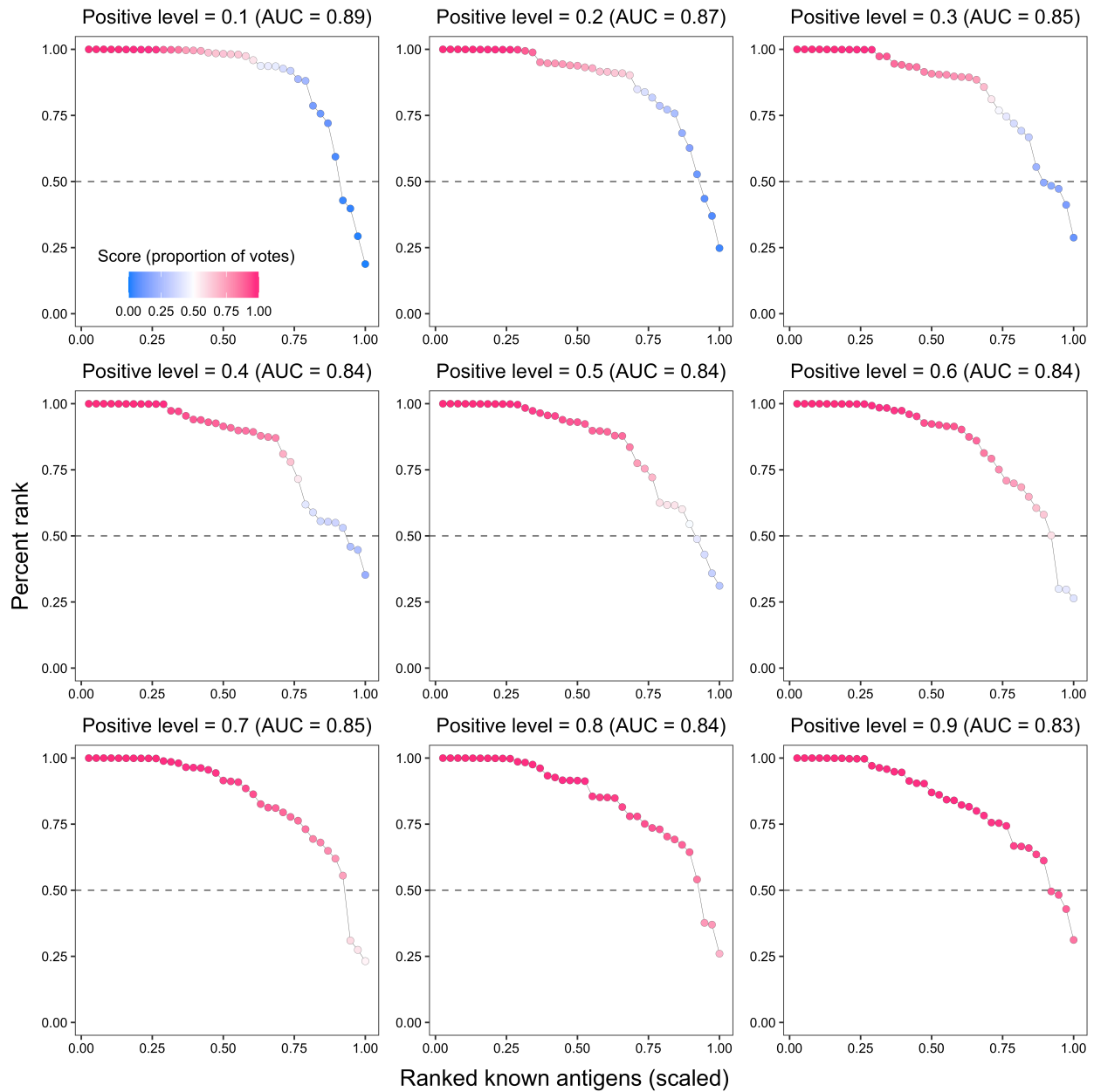
2.1.2.2 Bimodal distribution plot

```
x_grob <- textGrob("Score (proportion of votes)", gp = gpar(fontsize = 15))  
y_grob <- textGrob("Density", gp = gpar(fontsize = 15), rot = 90)  
grid.arrange(arrangeGrob(plot_grid(plotlist = plot_list, ncol = 3), left = y_grob, bottom = x_grob))
```



2.1.2.3 Known antigen ranking

```
x_grob2 <- textGrob("Ranked known antigens (scaled)", gp = gpar(fontsize = 15))
y_grob2 <- textGrob("Percent rank", gp = gpar(fontsize = 15), rot = 90)
grid.arrange(arrangeGrob(plot_grid(plotlist = plot_list2, ncol = 3), left = y_grob2, bottom = x_grob2))
```



2.1.2.4 Pv + Pf combined data set

```
data <- read.csv("./other_data/pfpv_pos_level_parameter_tuning.csv", header = TRUE, row.names = 1,
  ↪ check.names = FALSE)
# Extract data with only unlabeled proteins
```

```
data_unl <- data[data$antigen_label == 0, ]
```

```
plot_list3 <- list()
plot_list4 <- list()
for (i in seq(0.1, 0.9, 0.1)) {
  fit <- mixfit(data_unl[[as.character(i)]], ncomp = 2)

  # Calculate receiver operating characteristic (ROC) curve
  # for putative positive and negative samples
  x <- seq(-0.5, 1.5, by = 0.01)
  neg_cum <- pnorm(x, mean = fit$mu[1], sd = fit$sd[1])
  pos_cum <- pnorm(x, mean = fit$mu[2], sd = fit$sd[2])
  fpr <- (1 - neg_cum) / ((1 - neg_cum) + neg_cum) # false positive / (false positive + true negative)
  tpr <- (1 - pos_cum) / ((1 - pos_cum) + pos_cum) # true positive / (true positive + false negative)

  p <- plot(fit, title = paste0(
    "Positive level = ", i,
    " (AUROC = ", round(trapz(-fpr, tpr), 3), ")")
  )) +
    scale_fill_manual(values = c("#0080FF", "#FF007F"), labels = c("Putative negative", "Putative
    ↪ positive")) +
    theme_bw() +
    {
      if (i == 0.1) {
        theme(
          panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          axis.title = element_blank(),
          axis.text = element_text(colour = "black"),
          plot.title = element_text(hjust = 0.5, colour = "black"),
          legend.title = element_blank(),
          legend.text = element_text(colour = "black"),
          legend.position = c(0.7, 0.85),
          legend.background = element_blank()
        )
      } else {
        theme(
          panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
```

```

    axis.title = element_blank(),
    axis.text = element_text(colour = "black"),
    plot.title = element_text(hjust = 0.5, colour = "black"),
    legend.position = "none"
  )
}
} +
xlim(-0.8, 1.5)

# Calculate percent rank for labeled positives
data_ <- data[c("antigen_label", as.character(i))]
colnames(data_) <- c("antigen_label", "O0B score")
data_$percent_rank <- rank(data_[["O0B score"]]) / nrow(data)
data_ <- data_[data$antigen_label == 1, ]
data_ <- data_[order(-data_$percent_rank), ]
data_$x <- 1:nrow(data_) / nrow(data_)
cat(paste0("EPR: ", sum(data_$`O0B score` >= 0.5) / nrow(data_), "\n"))

p2 <- ggplot(data_, aes(x = x, y = `percent_rank`)) +
  geom_hline(yintercept = 0.5, linetype = "dashed", color = "grey50") +
  geom_line(linewidth = 0.1, color = "grey30") +
  geom_point(aes(fill = `O0B score`), size = 1.2, shape = 21, color = "grey30", stroke = 0.1) +
  scale_fill_gradient2(low = "#0080FF", mid = "white", high = "#FF007F", midpoint = 0.5, limits =
    ↪ c(0, 1)) +
  theme_bw() +
  {
    if (i == 0.1) {
      theme(
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.title = element_blank(),
        axis.text = element_text(colour = "black"),
        plot.title = element_text(hjust = 0.5, colour = "black"),
        legend.title = element_text(hjust = 0.5, colour = "black", angle = 0),
        legend.text = element_text(colour = "black"),
        legend.position = c(0.5, 0.15),
        legend.background = element_blank(),
        legend.title.align = -130
      )
    } else {
      theme(

```



```

    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.title = element_blank(),
    axis.text = element_text(colour = "black"),
    plot.title = element_text(hjust = 0.5, colour = "black"),
    legend.position = "none"
  )
}
} +
{
  if (i == 0.1) {
    guides(fill = guide_colourbar(title.position = "top", direction = "horizontal"))
  }
} +
{
  if (i == 0.1) {
    labs(fill = "Score (proportion of votes)")
  }
} +
ggtitle(paste0(
  "Positive level = ", i,
  " (AUC = ", round(trapz(c(0, data_$x, 1), c(1, data_$percent_rank, 0)), 2), ")")
)) +
ylim(0, 1)

plot_list3 <- list.append(plot_list3, p)
plot_list4 <- list.append(plot_list4, p2)
}

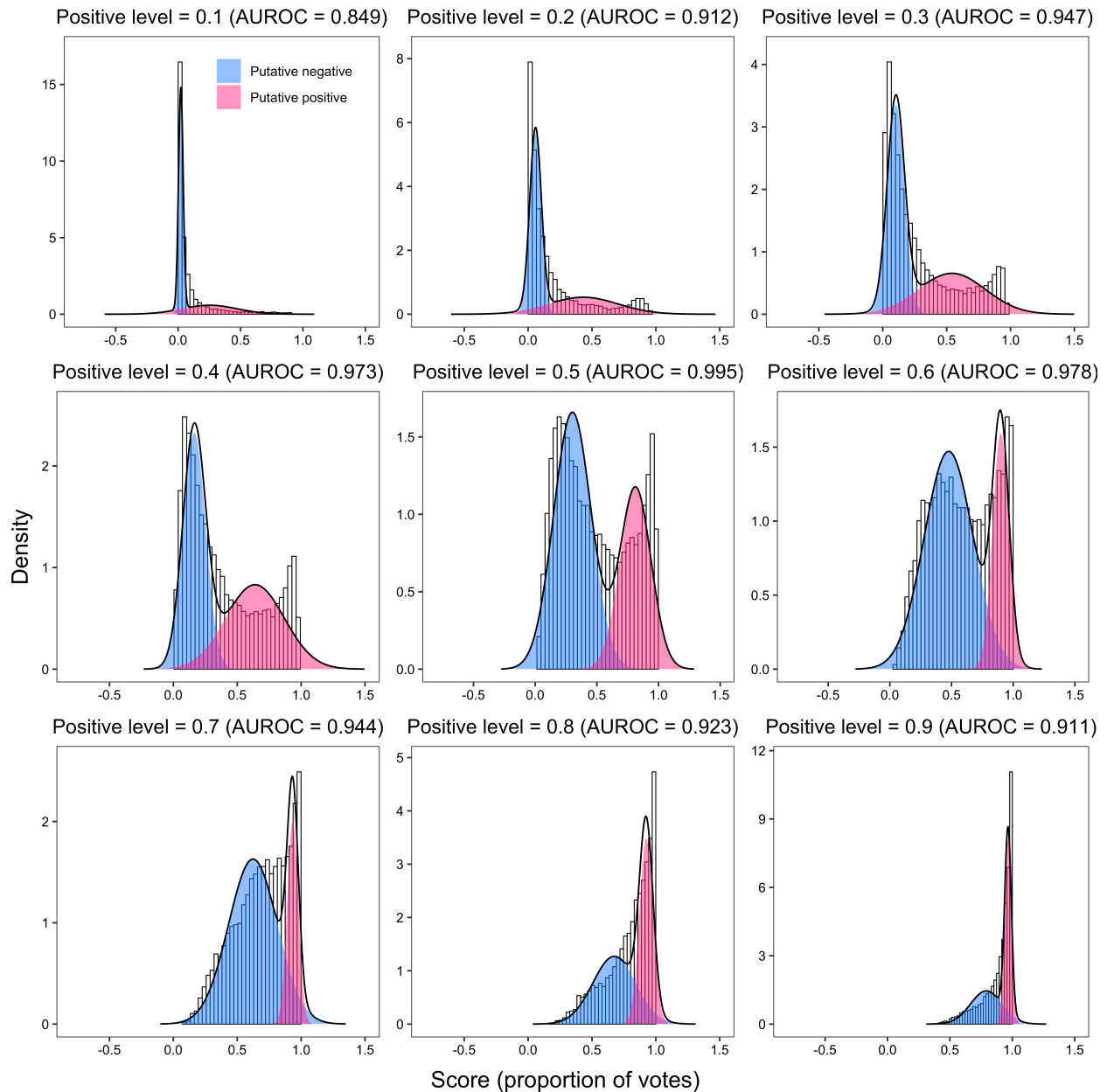
```

2.1.2.5 Bimodal distribution plot

```

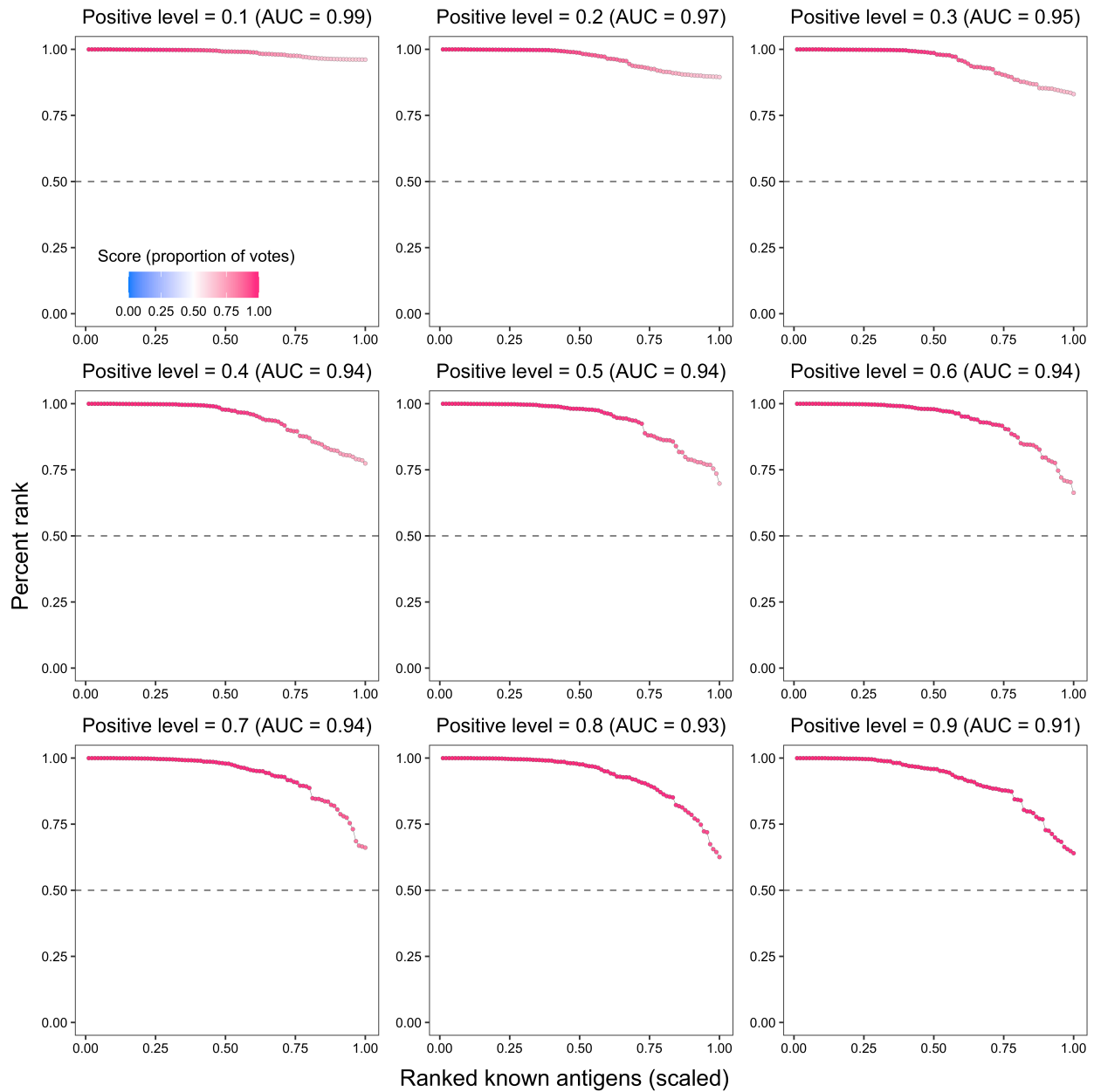
x_grob <- textGrob("Score (proportion of votes)", gp = gpar(fontsize = 15))
y_grob <- textGrob("Density", gp = gpar(fontsize = 15), rot = 90)
grid.arrange(arrangeGrob(plot_grid(plotlist = plot_list3, ncol = 3), left = y_grob, bottom = x_grob))

```



2.1.2.6 Known antigen ranking

```
x_grob2 <- textGrob("Ranked known antigens (scaled)", gp = gpar(fontsize = 15))
y_grob2 <- textGrob("Percent rank", gp = gpar(fontsize = 15), rot = 90)
grid.arrange(arrangeGrob(plot_grid(plotlist = plot_list4, ncol = 3), left = y_grob2, bottom = x_grob2))
```



2.1.3 Results for positive level = 0.5

```
plot_list[[5]]$labels$title <- ""
plot_list2[[5]]$labels$title <- ""
plot_list3[[5]]$labels$title <- ""
```

```

plot_list4[[5]]$labels$title <- ""

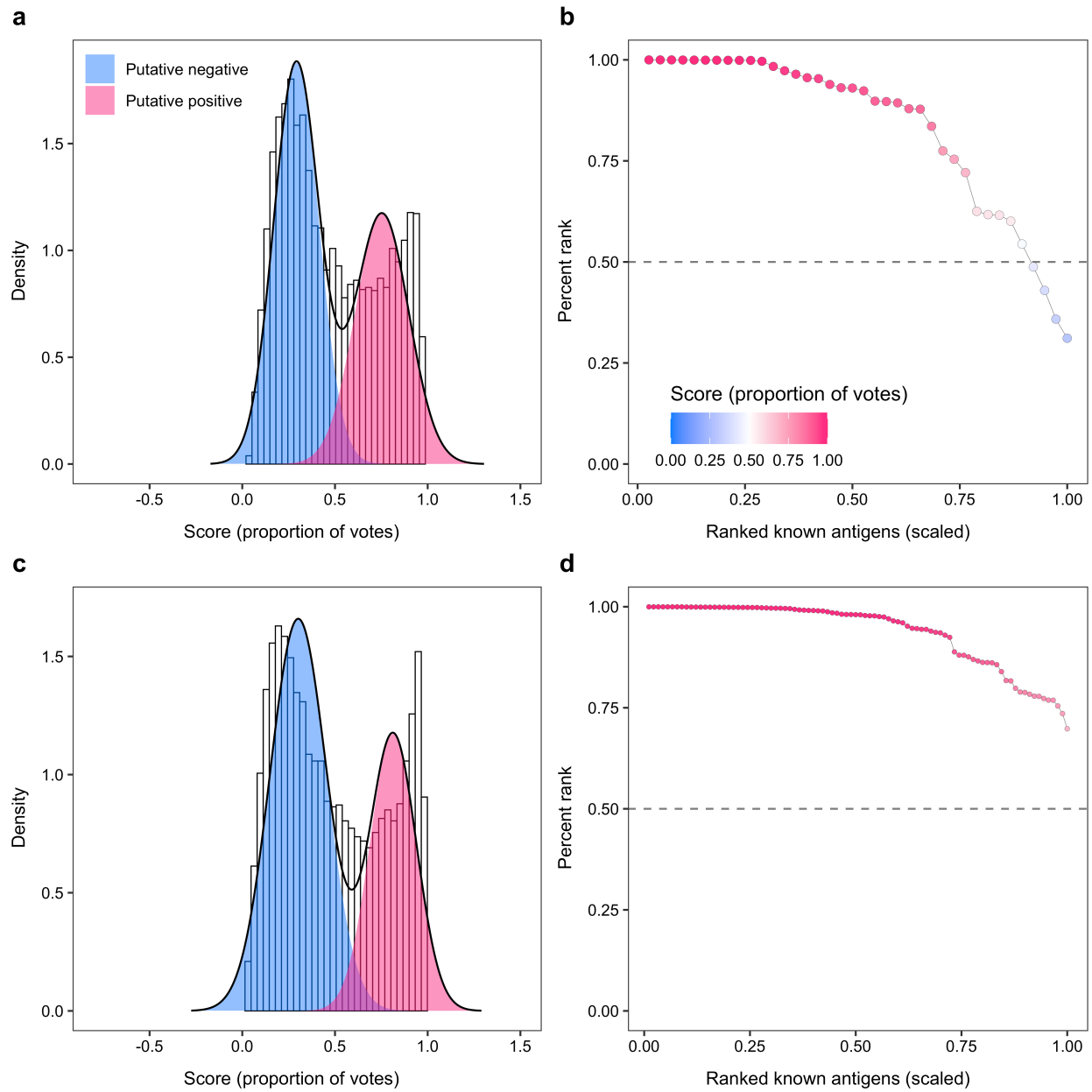
plot_list[[5]] <- plot_list[[5]] +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.title = element_blank(),
    axis.text = element_text(colour = "black"),
    plot.title = element_text(hjust = 0.5, colour = "black"),
    legend.title = element_blank(),
    legend.text = element_text(colour = "black"),
    legend.position = c(0.2, 0.91),
    legend.background = element_blank()
  )

plot_list2[[5]] <- plot_list2[[5]] +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.title = element_blank(),
    axis.text = element_text(colour = "black"),
    plot.title = element_text(hjust = 0.5, colour = "black"),
    legend.title = element_text(hjust = 0.5, colour = "black", angle = 0),
    legend.text = element_text(colour = "black"),
    legend.position = c(0.35, 0.13),
    legend.background = element_blank(),
    legend.title.align = -130
  ) +
  guides(fill = guide_colourbar(title.position = "top", direction = "horizontal")) +
  labs(fill = "Score (proportion of votes)")

x_grob <- textGrob("Score (proportion of votes)", gp = gpar(fontsize = 10))
y_grob <- textGrob("Density", gp = gpar(fontsize = 10), rot = 90)
x_grob2 <- textGrob("Ranked known antigens (scaled)", gp = gpar(fontsize = 10))
y_grob2 <- textGrob("Percent rank", gp = gpar(fontsize = 10), rot = 90)
plot_grid(
  arrangeGrob(plot_list[[5]], left = y_grob, bottom = x_grob),
  arrangeGrob(plot_list2[[5]], left = y_grob2, bottom = x_grob2),
  arrangeGrob(plot_list3[[5]], left = y_grob, bottom = x_grob),
  arrangeGrob(plot_list4[[5]], left = y_grob2, bottom = x_grob2),
  nrow = 2, labels = c("a", "b", "c", "d")
)

```

)



sessionInfo()

R version 4.2.3 (2023-03-15)

```
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur ... 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] grid      stats      graphics  grDevices  utils      datasets  methods
## [8] base
##
## other attached packages:
## [1] gridExtra_2.3   cowplot_1.1.1   ggplot2_3.4.2   rlist_0.4.6.2
## [5] pracma_2.4.2    mixR_0.2.0      reticulate_1.28
##
## loaded via a namespace (and not attached):
## [1] Rcpp_1.0.10      pillar_1.9.0     compiler_4.2.3    R.methodsS3_1.8.2
## [5] R.utils_2.12.2   tools_4.2.3      digest_0.6.31     tibble_3.2.1
## [9] jsonlite_1.8.4   evaluate_0.21    lifecycle_1.0.3   gtable_0.3.3
## [13] lattice_0.21-8   R.cache_0.16.0   pkgconfig_2.0.3   png_0.1-8
## [17] rlang_1.1.1      Matrix_1.5-4     cli_3.6.1         rstudioapi_0.14
## [21] yaml_2.3.7       xfun_0.39        fastmap_1.1.1     dplyr_1.1.2
## [25] withr_2.5.0      styler_1.9.1     knitr_1.42        generics_0.1.3
## [29] vctrs_0.6.2      tidyselect_1.2.0 rprojroot_2.0.3   data.table_1.14.8
## [33] glue_1.6.2       here_1.0.1       R6_2.5.1          fansi_1.0.4
## [37] rmarkdown_2.21   bookdown_0.34    purrr_1.0.1       magrittr_2.0.3
## [41] scales_1.2.1     codetools_0.2-19 htmltools_0.5.5   colorspace_2.1-0
## [45] utf8_1.2.3       munsell_0.5.0    R.oo_1.25.0
```

```
session_info.show()
```

```
## -----
## joblib          1.1.1
## numpy           1.19.0
## pandas          1.3.2
## purf            NA
## scipy           1.8.0
## session_info    1.0.0
## sklearn         0.24.2
```

```
## -----  
## Python 3.8.2 (default, Mar 26 2020, 10:45:18) [Clang 4.0.1 (tags/RELEASE_401/final)]  
## macOS-10.16-x86_64-i386-64bit  
## -----  
## Session information updated at 2023-05-18 12:30
```

Section 3

Model evaluation

3.1 Known antigen prediction accuracy

3.1.1 Analysis

In Python:

```
library(reticulate)
use_condaenv("/Users/renee/Library/r-miniconda/envs/purf/bin/python")
```

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler
from scipy.spatial import distance
from purf.pu_ensemble import PURandomForestClassifier
import pickle
import os
import re
import session_info
```

3.1.1.1 Pv model cross-species predictions

```
data_ = pd.read_csv('./other_data/pv_ml_input.csv', index_col=0)
X_ = data_.iloc[:,1:]
imputer = SimpleImputer(strategy='median')
imputer.fit(X_)

data = pd.read_csv('./other_data/pf_ml_input.csv', index_col=0)
X = data.iloc[:,1:]
y = np.array(data.antigen_label)
```



```

pickle_path = './pv_0.5_purf_tree_filtering.pkl'

with open(pickle_path, 'rb') as infile:
    purf = pickle.load(infile)

# Convert float type
X[X.select_dtypes(np.float64).columns] = X.select_dtypes(np.float64).astype(np.float32)
# Impute missing values
X = imputer.transform(X)

pred = purf['model'].predict_proba(X)[: ,1]

res = pd.DataFrame({'protein_id': data.index, 'antigen_label': y})
res['00B score filtered'] = pred
res.to_csv('~Downloads/pv_single_cross_predictions.csv', index=False)

```

3.1.1.2 Pf model cross-species predictions

```

data_ = pd.read_csv('./other_data/pf_ml_input.csv', index_col=0)
X_ = data_.iloc[:,1:]
imputer = SimpleImputer(strategy='median')
imputer.fit(X_)

data = pd.read_csv('./other_data/pv_ml_input.csv', index_col=0)
X = data.iloc[:,1:]
y = np.array(data.antigen_label)

pickle_path = './pf_0.5_purf_tree_filtering.pkl'

with open(pickle_path, 'rb') as infile:
    purf = pickle.load(infile)

# Convert float type
X[X.select_dtypes(np.float64).columns] = X.select_dtypes(np.float64).astype(np.float32)
# Impute missing values
X = imputer.transform(X)

pred = purf['model'].predict_proba(X)[: ,1]

```

```
res = pd.DataFrame({'protein_id': data.index, 'antigen_label': y})
res['00B score filtered'] = pred
res.to_csv('~Downloads/pf_single_cross_predictions.csv', index=False)
```

In R:

```
library(DT)
```

```
single_model_res <- read.csv("./other_data/pf_single_pv_single_scores.csv", check.names = FALSE)
cross_pred_res <- read.csv("./other_data/pf_single_pv_single_cross_predictions.csv", check.names =
  ↪ FALSE)
combined_model_res <- read.csv("./data/supplementary_data_5_pfpv_purf_oob_predictions.csv",
  ↪ check.names = FALSE)
```

```
df <- data.frame(
  "PURF model" = c("Pv single model", "Pf single model", "Pv + Pf combined model"),
  "Accuracy for Pv known antigens" = c(
    paste0(
      nrow(single_model_res[single_model_res$antigen_label == 1 & single_model_res$species == "pv" &
        ↪ single_model_res$`00B score filtered` >= 0.5, ]),
      " / ", nrow(single_model_res[single_model_res$antigen_label == 1 & single_model_res$species ==
        ↪ "pv", ]),
      " = ", sprintf("%.2f", nrow(single_model_res[single_model_res$antigen_label == 1 &
        ↪ single_model_res$species == "pv" & single_model_res$`00B score filtered` >= 0.5, ])) /
        ↪ nrow(single_model_res[single_model_res$antigen_label == 1 & single_model_res$species ==
        ↪ "pv", ]))
    ),
    paste0(
      nrow(cross_pred_res[cross_pred_res$antigen_label == 1 & cross_pred_res$species == "pv" &
        ↪ cross_pred_res$`00B score filtered` >= 0.5, ]),
      " / ", nrow(cross_pred_res[cross_pred_res$antigen_label == 1 & cross_pred_res$species == "pv", ]),
      " = ", sprintf("%.2f", nrow(cross_pred_res[cross_pred_res$antigen_label == 1 &
        ↪ cross_pred_res$species == "pv" & cross_pred_res$`00B score filtered` >= 0.5, ])) /
        ↪ nrow(cross_pred_res[cross_pred_res$antigen_label == 1 & cross_pred_res$species == "pv", ]))
    ),
    paste0(
      nrow(combined_model_res[combined_model_res$antigen_label == 1 & combined_model_res$species ==
        ↪ "pv" & combined_model_res$`00B score filtered` >= 0.5, ]),
```

```

" / ", nrow(combined_model_res[combined_model_res$antigen_label == 1 &
  ↪ combined_model_res$species == "pv", ]),
" = ", sprintf("%.2f", nrow(combined_model_res[combined_model_res$antigen_label == 1 &
  ↪ combined_model_res$species == "pv" & combined_model_res`00B score filtered` >= 0.5, ])) /
  ↪ nrow(combined_model_res[combined_model_res$antigen_label == 1 & combined_model_res$species
  ↪ == "pv", ]))
)
),
"Accuracy for Pf known antigens" = c(
  paste0(
    nrow(cross_pred_res[cross_pred_res$antigen_label == 1 & cross_pred_res$species == "pf" &
      ↪ cross_pred_res`00B score filtered` >= 0.5, ]),
    " / ", nrow(cross_pred_res[cross_pred_res$antigen_label == 1 & cross_pred_res$species == "pf", ]),
    " = ", sprintf("%.2f", nrow(cross_pred_res[cross_pred_res$antigen_label == 1 &
      ↪ cross_pred_res$species == "pf" & cross_pred_res`00B score filtered` >= 0.5, ])) /
      ↪ nrow(cross_pred_res[cross_pred_res$antigen_label == 1 & cross_pred_res$species == "pf", ]))
  ),
  paste0(
    nrow(single_model_res[single_model_res$antigen_label == 1 & single_model_res$species == "pf" &
      ↪ single_model_res`00B score filtered` >= 0.5, ]),
    " / ", nrow(single_model_res[single_model_res$antigen_label == 1 & single_model_res$species ==
      ↪ "pf", ]),
    " = ", sprintf("%.2f", nrow(single_model_res[single_model_res$antigen_label == 1 &
      ↪ single_model_res$species == "pf" & single_model_res`00B score filtered` >= 0.5, ])) /
      ↪ nrow(single_model_res[single_model_res$antigen_label == 1 & single_model_res$species ==
      ↪ "pf", ]))
  ),
  paste0(
    nrow(combined_model_res[combined_model_res$antigen_label == 1 & combined_model_res$species ==
      ↪ "pf" & combined_model_res`00B score filtered` >= 0.5, ]),
    " / ", nrow(combined_model_res[combined_model_res$antigen_label == 1 &
      ↪ combined_model_res$species == "pf", ]),
    " = ", sprintf("%.2f", nrow(combined_model_res[combined_model_res$antigen_label == 1 &
      ↪ combined_model_res$species == "pf" & combined_model_res`00B score filtered` >= 0.5, ])) /
      ↪ nrow(combined_model_res[combined_model_res$antigen_label == 1 & combined_model_res$species
      ↪ == "pf", ]))
  )
),
check.names = FALSE
)

```

```
save(df, file = "./rdata/known_antigen_pred_acc.RData")
```

```
load("./rdata/known_antigen_pred_acc.RData")
df %>%
  datatable(rownames = FALSE)
```

PURF model	Accuracy for Pf known antigens	Accuracy for Pv known antigens
---------------	-----------------------------------	-----------------------------------

```
pv_antigens <- single_model_res[single_model_res$antigen_label == 1 & single_model_res$species ==
  ↪ "pv", ]$protein_id
pv_model_pf_antigens <- cross_pred_res[cross_pred_res$antigen_label == 1 & cross_pred_res$species ==
  ↪ "pf" & cross_pred_res`00B score filtered` >= 0.5, ]$protein_id
pv_model_pv_antigens <- single_model_res[single_model_res$antigen_label == 1 &
  ↪ single_model_res$species == "pv" & single_model_res`00B score filtered` >= 0.5, ]$protein_id

pf_antigens <- single_model_res[single_model_res$antigen_label == 1 & single_model_res$species ==
  ↪ "pf", ]$protein_id
pf_model_pf_antigens <- single_model_res[single_model_res$antigen_label == 1 &
  ↪ single_model_res$species == "pf" & single_model_res`00B score filtered` >= 0.5, ]$protein_id
pf_model_pv_antigens <- cross_pred_res[cross_pred_res$antigen_label == 1 & cross_pred_res$species ==
  ↪ "pv" & cross_pred_res`00B score filtered` >= 0.5, ]$protein_id

pv_intersect <- intersect(pf_model_pv_antigens, pv_model_pv_antigens)
pf_intersect <- intersect(pf_model_pf_antigens, pv_model_pf_antigens)
pv_union <- union(pf_model_pv_antigens, pv_model_pv_antigens)
pf_union <- union(pf_model_pf_antigens, pv_model_pf_antigens)

df <- data.frame(
  "Set" = c("Pv model only", "Intersect", "Pf model only", "Complement"),
  "Pv known antigens" = c(
    paste(pv_model_pv_antigens[!pv_model_pv_antigens %in% pv_intersect], collapse = ", "),
    length(pv_intersect),
    paste(pf_model_pv_antigens[!pf_model_pv_antigens %in% pv_intersect], collapse = ", "),
    paste(pv_antigens[!pv_antigens %in% pv_union], collapse = ", ")
  ),
  "Pf known antigens" = c(
    paste(pv_model_pf_antigens[!pv_model_pf_antigens %in% pf_intersect], collapse = ", "),
```

```

length(pf_intersect),
paste(pf_model_pf_antigens[!pf_model_pf_antigens %in% pf_intersect], collapse = ", "),
paste(pf_antigens[!pf_antigens %in% pf_union], collapse = ", ")
),
check.names = FALSE
)

save(df, file = "./rdata/known_antigen_pred_set_analysis.RData")

```

```

load("./rdata/known_antigen_pred_set_analysis.RData")
df %>%
  datatable(rownames = FALSE)

```

Set Pf known antigens Pv known antigens

3.2 Adversarial controls

3.2.1 Analysis

In Python:

```

library(reticulate)
use_condaenv("/Users/renee/Library/r-miniconda/envs/purf/bin/python")

```

```

import pandas as pd
from sklearn.utils import shuffle
import numpy as np
import pickle
from sklearn.impute import SimpleImputer
from purf.pu_ensemble import PURandomForestClassifier
from sklearn.preprocessing import MinMaxScaler
from scipy.spatial import distance
from joblib import Parallel, delayed
from sklearn.ensemble._forest import _generate_unsampled_indices
import os
import re
import session_info

```

```

# private function for train_purf()
def _get_ref_antigen_stats(idx, tree, X, y, ref_indices, max_samples=None):
    if max_samples is None:
        max_samples = y.shape[0]
    oob_indices = _generate_unsampled_indices(tree.random_state, y.shape[0], max_samples)
    ref_oob = [i in oob_indices for i in ref_indices]
    ref_pred = list()
    pred = tree.predict_proba(X[ref_indices,:], check_input=False)
    ref_pred = pred[:,1]
    return ref_oob, ref_pred

def train_purf(features, outcome, pos_level=0.5, purf=None, tree_filtering=False, ref_antigens=None,
    ↪ n_jobs=1):
    features, outcome = shuffle(features, outcome, random_state=0)
    # Imputation
    imputer = SimpleImputer(strategy='median')
    X = imputer.fit_transform(features)
    X = pd.DataFrame(X, index=features.index, columns=features.columns)
    y = outcome
    features = X
    print('There are %d positives out of %d samples before feature space weighting.' % (sum(y), len(y)))
    # Feature space weighting
    lab_pos = X.loc[y==1,:]
    median = np.median(lab_pos, axis=0)
    # Feature space weighting
    lab_pos = X.loc[y==1,:]
    median = np.median(lab_pos, axis=0)
    scaler = MinMaxScaler(feature_range=(1,10))
    dist = list()
    for i in range(lab_pos.shape[0]):
        dist.append(distance.euclidean(lab_pos.iloc[i, :], median))
    dist = np.asarray(dist).reshape(-1, 1)
    counts = np.round(scaler.fit_transform(dist))
    counts = np.array(counts, dtype=np.int64)[: , 0]
    X_temp = X.iloc[y==1, :]
    X = X.iloc[y==0, :]
    y = np.asarray([0] * X.shape[0] + [1] * (sum(counts)))
    appended_data = [X]
    for i in range(len(counts)):
        appended_data.append(pd.concat([X_temp.iloc[[i]] * counts[i]))
    X = pd.concat(appended_data)

```

```

print('There are %d positives out of %d samples after feature space weighting.' % (sum(y),
    ↪ len(y)))
res = pd.DataFrame({'protein_id': X.index, 'antigen_label' : y})
if tree_filtering is True:
    # get ref antigen indices
    ref_index_dict = {ref:list() for ref in list(ref_antigens.values())}
    for i in range(res.shape[0]):
        if res['protein_id'][i] in list(ref_antigens.values()):
            ref_index_dict[res['protein_id'][i]].append(res.index[i])
    ref_indices = sum(ref_index_dict.values(), [])
    # get OOB stats and predictions
    X = X.astype('float32')
    trees = purf.estimators_
    idx_list = [i for i in range(len(trees))]
    stats_res = Parallel(n_jobs=n_jobs)(
        delayed(_get_ref_antigen_stats)(idx, trees[idx], np.array(X), y, ref_indices) for
    ↪ idx in idx_list)
    # ref_oob data structure:
    # rows represent individual trees
    # column represent reference antigens
    # cells indicate whether the reference antigen is in the OOB samples of the tree
    ref_oob = np.array([ref_oob for ref_oob, ref_pred in stats_res])
    # ref_pred data structure:
    # rows represent individual trees
    # column represent reference antigens
    # cells indicate the prediction of the reference antigen by the tree
    ref_pred = np.array([ref_pred for ref_oob, ref_pred in stats_res])
    # analyze duplicated reference antigens as a group
    cumsum_num_ref = np.cumsum(np.array([len(v) for k,v in ref_index_dict.items()]))
    ref_oob_all = np.array([ref_oob[:, 0:cumsum_num_ref[i]].any(axis=1) if i == 0 else \
        ref_oob[:, cumsum_num_ref[i - 1]:cumsum_num_ref[i]].any(axis=1) \
        for i in range(len(ref_antigens))]).T
    ref_pred_all = np.array([ref_pred[:, 0:cumsum_num_ref[i]].any(axis=1) if i == 0 else \
        ref_pred[:, cumsum_num_ref[i - 1]:cumsum_num_ref[i]].sum(axis=1) \
        for i in range(len(ref_antigens))]).T
    # calculate number of reference antigens as OOB samples for each tree
    oob_total = ref_oob_all.sum(axis=1)
    # assign score of 1 to trees that correctly predict all OOB reference antigens; otherwise,
    ↪ assign 0 score
    weights = np.zeros(len(trees))
    # iterate through the trees and calculate the stats

```

```

for i in range(len(trees)):
    oob_list = list(ref_oob_all[i,:])
    pred_list = list(ref_pred_all[i,:])
    if oob_total[i] == 0:
        weights[i] = 0
    else:
        if sum(np.array(pred_list)[oob_list] != 0) == oob_total[i]:
            weights[i] = 1
if tree_filtering is False:
    # Training PURF
    purf = PURandomForestClassifier(
        n_estimators = 100000,
        oob_score = True,
        n_jobs = 64,
        random_state = 42,
        pos_level = pos_level
    )
    purf.fit(X, y)
else:
    purf._set_oob_score_with_weights(np.array(X), y.reshape(-1,1), weights=weights)
# Storing results
res['OoB score'] = purf.oob_decision_function[:,1]
res = features.merge(res.groupby('protein_id').mean(), left_index=True, right_on='protein_id')
res = res[['antigen_label', 'OoB score']]
if tree_filtering is False:
    return (purf, res)
else:
    return ({'model': purf, 'weights': weights}, res)

```

3.2.1.1 Pv data set

```

ref_antigens = {'CSP': 'PVP01_0835600.1-p1', 'DBP': 'PVP01_0623800.1-p1', 'MSP1':
    ↪ 'PVP01_0728900.1-p1'}
data = pd.read_csv('./other_data/pv_ml_input.csv', index_col=0)
features = data.iloc[:, 1:]
outcome = np.array(data.antigen_label)

for (idx, (antigen, out)) in enumerate(zip(features.index, outcome)):
    if out == 1:

```



```

    if antigen in ref_antigens.values():
        continue
    outcome_ = outcome.copy()
    outcome_[idx] = 0
    (purf, res) = train_purf(features, outcome_, pos_level=0.5, tree_filtering=False)
    (purf_filtered, res_filtered) = train_purf(features, outcome_, pos_level=0.5, purf=purf,
    ↪ tree_filtering=True, ref_antigens=ref_antigens)
    res_filtered.to_csv('~Downloads/pv_jackknife/' + antigen + '_res.csv')

```

```

dir = '~Downloads/pv_jackknife/'
files = os.listdir(dir)

for file in files:
    if file.endswith('.csv'):
        tmp = pd.read_csv(dir + file, index_col=0)['antigen_label']
        break

data_frames = [pd.read_csv(dir + file, index_col=0)['00B score'] for file in files if
    ↪ file.endswith('.csv')]
merged_df = pd.concat([tmp] + data_frames, join='outer', axis=1)
colnames = ['antigen_label'] + [re.match('PVP01_[0-9]+\.[0-9]-p1', file)[0] for file in files if
    ↪ file.endswith('.csv')]
merged_df.columns = colnames
merged_df.to_csv('./other_data/pv_adversarial_controls.csv')

```

3.2.1.2 Pv + Pf combined data set

```

ref_antigens = {'CSP (Pf)': 'PF3D7_0304600.1-p1', 'RH5 (Pf)': 'PF3D7_0424100.1-p1', 'MSP5 (Pf)':
    ↪ 'PF3D7_0206900.1-p1',
                'P230 (Pf)': 'PF3D7_0209000.1-p1',
                'CSP (Pv)': 'PVP01_0835600.1-p1', 'DBP (Pv)': 'PVP01_0623800.1-p1', 'MSP1 (Pv)':
    ↪ 'PVP01_0728900.1-p1'}
data = pd.read_csv('./data/supplementary_data_4_pfpv_ml_input.csv', index_col=0)
features = data.iloc[:, 1:]
outcome = np.array(data.antigen_label)

for (idx, (antigen, out)) in enumerate(zip(features.index, outcome)):
    if out == 1:

```

```

    if antigen in ref_antigens.values():
        continue
    outcome_ = outcome.copy()
    outcome_[idx] = 0
    (purf, res) = train_purf(features, outcome_, pos_level=0.5, tree_filtering=False)
    (purf_filtered, res_filtered) = train_purf(features, outcome_, pos_level=0.5, purf=purf,
    ↪ tree_filtering=True, ref_antigens=ref_antigens)
    res_filtered.to_csv('~Downloads/pfpv_jackknife/' + antigen + '_res.csv')

```

```

dir = '~Downloads/pfpv_jackknife/'
files = os.listdir(dir)

for file in files:
    if file.endswith('.csv'):
        tmp = pd.read_csv(dir + file, index_col=0)['antigen_label']
        break

data_frames = [pd.read_csv(dir + file, index_col=0)['00B score'] for file in files if
    ↪ file.endswith('.csv')]
merged_df = pd.concat([tmp] + data_frames, join='outer', axis=1)
colnames = ['antigen_label'] + [re.match('PF3D7_[0-9]+\.[0-9]-p1', file)[0] if
    ↪ file.startswith('PF3D7') else re.match('PVP01_[0-9]+\.[0-9]-p1', file)[0] for file in files if
    ↪ file.endswith('.csv')]
merged_df.columns = colnames
merged_df.to_csv('./other_data/pfpv_adversarial_controls.csv')

```

3.2.2 Plotting

In R:

```

library(ggplot2)
library(ggdist)
library(gghalves)
library(cowplot)
library(grid)
library(ggbeeswarm)
library(ggpubr)

```

```

pv_data <- read.csv("./data/supplementary_data_3_pv_purf_oob_predictions.csv", check.names = FALSE,
  ↪ row.names = 1)
pf_data <- read.csv("./other_data/pf_single_pv_single_scores.csv", check.names = FALSE, row.names = 1)
pfpv_data <- read.csv("./data/supplementary_data_5_pfpv_purf_oob_predictions.csv", check.names =
  ↪ FALSE, row.names = 1)

pv_ad_ctrl <- read.csv("./other_data/pv_adversarial_controls.csv", check.names = FALSE, row.names = 1)
pf_ad_ctrl <- read.csv("./other_data/pf_adversarial_controls.csv", check.names = FALSE, row.names = 1)
pfpv_ad_ctrl <- read.csv("./other_data/pfpv_adversarial_controls.csv", check.names = FALSE, row.names
  ↪ = 1)

```

3.2.2.1 Mean differences in scores

```

pv_products <- read.csv("./other_data/pvp01_gene_products_v62.csv", row.names = 1)
pf_products <- read.csv("./other_data/pf3d7_gene_products_v62.csv", row.names = 1)
gene_products <- rbind(pf_products, pv_products)

calculate_known_antigen_scores <- function(validation_data, baseline_scores) {
  scores <- c()
  for (i in 2:ncol(validation_data)) {
    known_antigen <- sort(colnames(validation_data))[i]
    other_antigens <- sort(colnames(validation_data))[-c(1, i)]
    scores <- c(scores, mean(validation_data[other_antigens, known_antigen] -
  ↪ baseline_scores[other_antigens, ]))
  }
  names(scores) <- sort(colnames(validation_data))[2:ncol(validation_data)]
  return(scores)
}

pv_ad_ctrl_scores <- calculate_known_antigen_scores(pv_ad_ctrl, pv_data["00B score filtered"])
pf_ad_ctrl_scores <- calculate_known_antigen_scores(pf_ad_ctrl, pf_data[rownames(pf_ad_ctrl), "00B
  ↪ score filtered", drop = FALSE])
pfpv_ad_ctrl_scores <- calculate_known_antigen_scores(pfpv_ad_ctrl, pfpv_data["00B score filtered"])

pv_ad_ctrl_res <- data.frame(
  "score" = pv_ad_ctrl_scores,
  "gene_product" = gene_products[names(pv_ad_ctrl_scores), "gene_product"],
  row.names = names(pv_ad_ctrl_scores)
)

```

```

pv_ad_ctrl_res$source <- "Intersect"
literature <- c(
  "PVP01_0102300.1-p1", "PVP01_0118900.1-p1", "PVP01_0202200.1-p1", "PVP01_0205500.1-p1",
  "PVP01_0210500.1-p1", "PVP01_0304300.1-p1", "PVP01_0317900.1-p1", "PVP01_0418300.1-p1",
  "PVP01_0418400.1-p1", "PVP01_0529100.1-p1", "PVP01_0532400.1-p1", "PVP01_0616000.1-p1",
  "PVP01_0616100.1-p1", "PVP01_0701200.1-p1", "PVP01_0707700.1-p1", "PVP01_1026000.1-p1",
  "PVP01_1030900.1-p1", "PVP01_1129100.1-p1"
)
IEDB <- c(
  "PVP01_0106200.1-p1", "PVP01_0106300.1-p1", "PVP01_0215600.1-p1", "PVP01_0307400.1-p1",
  "PVP01_0522300.1-p1", "PVP01_0813700.1-p1", "PVP01_0829000.1-p1", "PVP01_0835600.1-p1",
  "PVP01_0923000.1-p1", "PVP01_0932400.1-p1", "PVP01_1240600.1-p1", "PVP01_1306000.1-p1",
  "PVP01_1412900.1-p1", "PVP01_1453900.1-p1"
)
pv_ad_ctrl_res$source[rownames(pv_ad_ctrl_res) %in% literature] <- "Literature"
pv_ad_ctrl_res$source[rownames(pv_ad_ctrl_res) %in% IEDB] <- "IEDB"

pfpv_ad_ctrl_res <- data.frame(
  "score" = pfpv_ad_ctrl_scores,
  "gene_product" = gene_products[names(pfpv_ad_ctrl_scores), "gene_product"],
  row.names = names(pfpv_ad_ctrl_scores)
)
pfpv_ad_ctrl_res$source <- "Intersect"
pfpv_ad_ctrl_res$source[rownames(pfpv_ad_ctrl_res) %in% literature] <- "Literature"
pfpv_ad_ctrl_res$source[rownames(pfpv_ad_ctrl_res) %in% IEDB] <- "IEDB"

write.csv(pv_ad_ctrl_res, file = "./other_data/pv_ad_ctrl_res.csv", row.names = TRUE)
write.csv(pfpv_ad_ctrl_res, file = "./other_data/pfpv_ad_ctrl_res.csv", row.names = TRUE)

```

```

pv_ad_ctrl_res <- read.csv("./other_data/pv_ad_ctrl_res.csv", row.names = 1)
pfpv_ad_ctrl_res <- read.csv("./other_data/pfpv_ad_ctrl_res.csv", row.names = 1)
data_ <- data.frame(
  group = factor(c(
    rep(0, length(pv_ad_ctrl_scores)), rep(1, length(pf_ad_ctrl_scores)),
    rep(2, length(pfpv_ad_ctrl_scores))
  )),
  score = c(pv_ad_ctrl_scores, pf_ad_ctrl_scores, pfpv_ad_ctrl_scores)
)

```

```

p <- ggplot(data_, aes(x = group, y = score)) +
  geom_hline(yintercept = 0, color = "grey80", linetype = "dashed") +
  ggdist::stat_halfeye(aes(color = group, fill = group), adjust = 0.5, width = 0.7, .width = 0,
  ↪ justification = -0.1, point_colour = NA, alpha = 0.9) +
  geom_boxplot(aes(color = group), width = 0.1, outlier.color = NA, lwd = 0.5, show.legend = FALSE) +
  gghalves::geom_half_point(aes(fill = group), side = "l", range_scale = 0.4, alpha = 0.6, shape = 21,
  ↪ color = "black", stroke = 0.1, size = 2) +
  coord_flip() +
  scale_color_manual(values = c("#03a1fc", "#984EA3", "#FF7F00")) +
  scale_fill_manual(values = c("#03a1fc", "#984EA3", "#FF7F00")) +
  scale_x_discrete(
    breaks = c(0, 1, 2),
    labels = c(
      expression(paste(italic("P. vivax"), " model")),
      expression(paste(italic("P. falciparum"), " model")),
      "Combined model"
    ),
    limits = rev
  ) +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.title.x = element_text(colour = "black"),
    axis.title.y = element_blank(),
    axis.text = element_text(colour = "black"),
    plot.title = element_text(hjust = 0.5, colour = "black"),
    plot.margin = ggplot2::margin(5, 5, 5, 5, "pt"),
    legend.title = element_blank(),
    legend.text = element_text(colour = "black"),
    legend.position = "none"
  ) +
  ylab("Mean difference in scores (proportion of votes)")

```

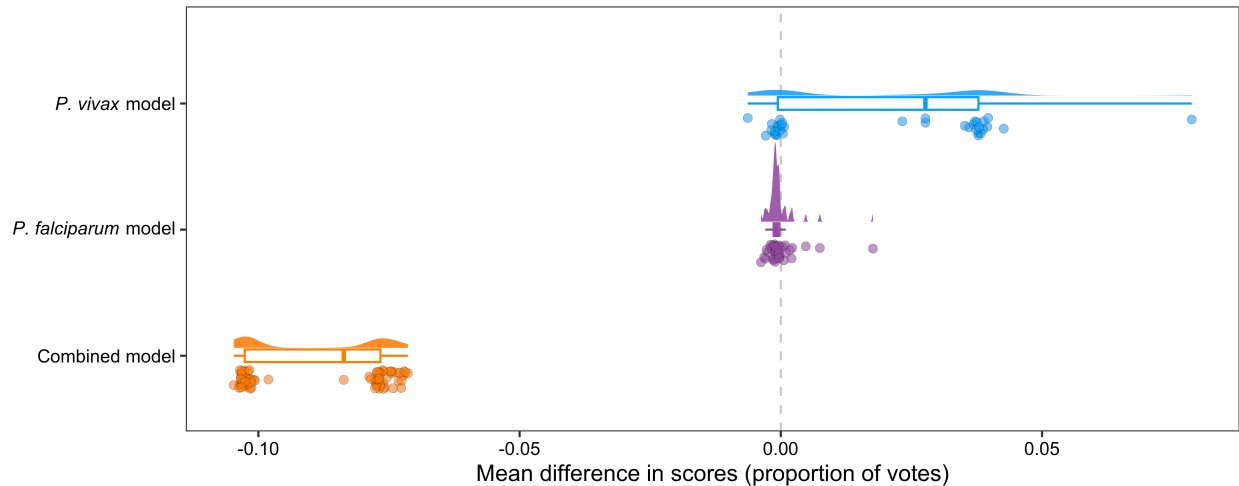
```

png(
  file = "../figures/Supplementary Fig 5.png",
  width = 5000, height = 2000, res = 600
)
print(p)

```

```
dev.off()

pdf(file = "../supplementary_figures/Supplementary Fig 5.pdf", width = 10, height = 4)
print(p)
dev.off()
```



3.2.2.2 Investigation of known antigen groups

```
library(umap)
library(ggplot2)
library(ggrepel)
library(cowplot)
```

```
pv_ad_ctrl_res <- read.csv("../other_data/pv_ad_ctrl_res.csv", row.names = 1)
# pv_ad_ctrl_res = pv_ad_ctrl_res[pv_ad_ctrl_res$source != 'Intersect', ]
pv_ad_ctrl_res$score_group <- ""
pv_ad_ctrl_res$score_group[pv_ad_ctrl_res$score < 0.01] <- "Group 1"
pv_ad_ctrl_res$score_group[pv_ad_ctrl_res$score >= 0.01] <- "Group 2"
table(pv_ad_ctrl_res$score_group, pv_ad_ctrl_res$source)
```

```
##
##          IEDB Intersect Literature
## Group 1   11         2         3
## Group 2    2         2        15
```

```
fisher.test(pv_ad_ctrl_res$score_group, pv_ad_ctrl_res$source)
```

```
##
## Fisher's Exact Test for Count Data
##
## data: pv_ad_ctrl_res$score_group and pv_ad_ctrl_res$source
## p-value = 0.0002597
## alternative hypothesis: two.sided
```

```
pfpv_ad_ctrl_res <- read.csv("./other_data/pfpv_ad_ctrl_res.csv", row.names = 1)
# pfpv_ad_ctrl_res = pfpv_ad_ctrl_res[pfpv_ad_ctrl_res$source != 'Intersect', ]
pfpv_ad_ctrl_res$score_group <- ""
pfpv_ad_ctrl_res$score_group[pfpv_ad_ctrl_res$score < -0.09] <- "Group 1"
pfpv_ad_ctrl_res$score_group[pfpv_ad_ctrl_res$score >= -0.09] <- "Group 2"
table(pfpv_ad_ctrl_res$score_group, pfpv_ad_ctrl_res$source)
```

```
##
##          IEDB Intersect Literature
## Group 1   12          27          2
## Group 2    1          25         16
```

```
fisher.test(pfpv_ad_ctrl_res$score_group, pfpv_ad_ctrl_res$source)
```

```
##
## Fisher's Exact Test for Count Data
##
## data: pfpv_ad_ctrl_res$score_group and pfpv_ad_ctrl_res$source
## p-value = 1.176e-05
## alternative hypothesis: two.sided
```

```
pfpv_ad_ctrl_res <- read.csv("./other_data/pfpv_ad_ctrl_res.csv", row.names = 1)
pfpv_ad_ctrl_res$score_group <- ""
pfpv_ad_ctrl_res$score_group[pfpv_ad_ctrl_res$score < -0.09] <- "Group 1"
pfpv_ad_ctrl_res$score_group[pfpv_ad_ctrl_res$score >= -0.09] <- "Group 2"
pfpv_ad_ctrl_res$species <- ""
pfpv_ad_ctrl_res$species[startsWith(rownames(pfpv_ad_ctrl_res), "PF3D7")] <- "Pf"
pfpv_ad_ctrl_res$species[startsWith(rownames(pfpv_ad_ctrl_res), "PVP01")] <- "Pv"
table(pfpv_ad_ctrl_res$score_group, pfpv_ad_ctrl_res$species)
```

```
##
##          Pf Pv
##   Group 1 25 16
##   Group 2 23 19

fisher.test(pfpv_ad_ctrl_res$score_group, pfpv_ad_ctrl_res$species)
```

```
##
## Fisher's Exact Test for Count Data
##
## data:  pfpv_ad_ctrl_res$score_group and pfpv_ad_ctrl_res$species
## p-value = 0.6584
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  0.4934941 3.3881568
## sample estimates:
## odds ratio
##  1.286776
```

3.2.2.3 Labeled known antigen accuracies

```
calculate_known_antigen_accuracies <- function(validation_data, baseline_scores) {
  accuracies <- c()
  for (i in 2:ncol(validation_data)) {
    known_antigen <- sort(colnames(validation_data))[i]
    other_antigens <- sort(colnames(validation_data))[-c(1, i)]
    accuracies <- c(accuracies, sum(validation_data[other_antigens, known_antigen] >= 0.5) /
  ↪ length(validation_data[other_antigens, known_antigen]))
  }
  return(accuracies)
}

pv_ad_ctrl_accuracies <- calculate_known_antigen_accuracies(pv_ad_ctrl, pv_data["00B score filtered"])
pf_ad_ctrl_accuracies <- calculate_known_antigen_accuracies(pf_ad_ctrl, pf_data[rownames(pf_ad_ctrl),
  ↪ "00B score filtered", drop = FALSE])
pfpv_ad_ctrl_accuracies <- calculate_known_antigen_accuracies(pfpv_ad_ctrl, pfpv_data["00B score
  ↪ filtered"])

df <- data.frame(
  "PURF model" = c("Pv single model", "Pf single model", "Pv + Pf combined model"),
```



```

"Accuracy (mean  $\pm$  SD)" = c(
  paste0(
    sprintf("%.2f", mean(pv_ad_ctrl_accuracies)), "  $\pm$  ",
    sprintf("%.2f", sd(pv_ad_ctrl_accuracies))
  ),
  paste0(
    sprintf("%.2f", mean(pf_ad_ctrl_accuracies)), "  $\pm$  ",
    sprintf("%.2f", sd(pf_ad_ctrl_accuracies))
  ),
  paste0(
    sprintf("%.2f", mean(pfpv_ad_ctrl_accuracies)), "  $\pm$  ",
    sprintf("%.2f", sd(pfpv_ad_ctrl_accuracies))
  )
),
check.names = FALSE
)

save(df, file = "./rdata/ad_ctrl_pred_accuracy.RData")

```

```

load("./rdata/ad_ctrl_pred_accuracy.RData")
df %>%
  datatable(rownames = FALSE)

```

PURF model Accuracy (mean $\pm$ SD)

```
sessionInfo()
```

```

## R version 4.2.3 (2023-03-15)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur ... 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:

```

```
## [1] grid      stats      graphics  grDevices utils      datasets  methods
## [8] base
##
## other attached packages:
## [1] ggrepel_0.9.3    umap_0.2.10.0    ggpubr_0.6.0     ggbeeswarm_0.7.2
## [5] cowplot_1.1.1    gghalves_0.1.4   ggdist_3.2.1     ggplot2_3.4.2
## [9] DT_0.27          reticulate_1.28
##
## loaded via a namespace (and not attached):
## [1] sass_0.4.6        tidyr_1.3.0       jsonlite_1.8.4
## [4] carData_3.0-5     R.utils_2.12.2    here_1.0.1
## [7] bslib_0.4.2       distributional_0.3.2 askpass_1.1
## [10] highr_0.10        vipor_0.4.5       yaml_2.3.7
## [13] pillar_1.9.0      backports_1.4.1    lattice_0.21-8
## [16] glue_1.6.2        digest_0.6.31      ggsignif_0.6.4
## [19] colorspace_2.1-0  htmltools_0.5.5    Matrix_1.5-4
## [22] R.oo_1.25.0       pkgconfig_2.0.3    broom_1.0.4
## [25] bookdown_0.34     purrr_1.0.1        scales_1.2.1
## [28] webshot_0.5.4     processx_3.8.1     RSpectra_0.16-1
## [31] tibble_3.2.1      openssl_2.0.6      styler_1.9.1
## [34] generics_0.1.3    farver_2.1.1       car_3.1-2
## [37] ellipsis_0.3.2    cachem_1.0.8       withr_2.5.0
## [40] cli_3.6.1         magrittr_2.0.3     evaluate_0.21
## [43] ps_1.7.5          R.methodsS3_1.8.2  fansi_1.0.4
## [46] R.cache_0.16.0    rstatix_0.7.2      beeswarm_0.4.0
## [49] tools_4.2.3       lifecycle_1.0.3    munsell_0.5.0
## [52] callr_3.7.3       compiler_4.2.3     jquerylib_0.1.4
## [55] rlang_1.1.1       rstudioapi_0.14    htmlwidgets_1.6.2
## [58] crosstalk_1.2.0   rmarkdown_2.21     gtable_0.3.3
## [61] codetools_0.2-19  abind_1.4-5        R6_2.5.1
## [64] knitr_1.42        dplyr_1.1.2        fastmap_1.1.1
## [67] utf8_1.2.3        rprojroot_2.0.3    Rcpp_1.0.10
## [70] vctrs_0.6.2       png_0.1-8          tidyselect_1.2.0
## [73] xfun_0.39
```

```
session_info.show()
```

```
## -----
## joblib          1.1.1
## numpy           1.19.0
## pandas          1.3.2
## purf            NA
```

```
## scipy                1.8.0
## session_info         1.0.0
## sklearn              0.24.2
## -----
## Python 3.8.2 (default, Mar 26 2020, 10:45:18) [Clang 4.0.1 (tags/RELEASE_401/final)]
## macOS-10.16-x86_64-i386-64bit
## -----
## Session information updated at 2023-05-18 12:30
```

Section 4

Model comparisons

4.1 Known antigen score comparisons

In R:

```
library(ggplot2)
library(reshape2)
library(ggdist)
library(gghalves)
library(cowplot)
```

```
pv_scores <- read.csv("./other_data/pv_known_antigen_scores.csv", row.names = 1)
pv_unl_scores <- read.csv("./other_data/pv_unlabeled_protein_scores.csv", row.names = 1)
pv_scores$label <- 2
pv_unl_scores$label <- 1
data_ <- melt(rbind(pv_scores, pv_unl_scores), id.vars = "label")
data_$variable <- factor(data_$variable, levels = c(
  "pf_single_model",
  "pv_single_model", "pv_pf_pos",
  "pfpv_combined_model",
  "pv_pf_unl"
))
data_$label[data_$value < 0.5 & data_$label == 1] <- 0
set.seed(12)
p1 <- ggplot(data_, aes(x = variable, y = value)) +
  geom_hline(yintercept = 0.5, color = "grey70", linetype = "dashed") +
  stat_halfeye(aes(fill = factor(label, levels = c(2, 1, 0))),
    slab_color = "black",
    slab_linewidth = 0.2, adjust = 0.5, width = 0.5, .width = 0,
    justification = -0.2, point_colour = NA, alpha = 0.6, normalize = "all"
  ) +
  geom_boxplot(aes(fill = factor(label, levels = c(2, 1, 0))),
    width = 0.2, outlier.color = NA,
```

```

    lwd = 0.3, show.legend = FALSE, alpha = 0.6, color = "black"
  ) +
  geom_half_point(
    data = data_[data_$label == 2, ], fill = "#FCB40A", side = "l",
    range_scale = 0.3, alpha = 0.8, shape = 21, color = "black", stroke = 0.1, size = 2
  ) +
  scale_fill_manual(
    name = "", values = c("#FCB40A", "#FF007F", "#0080FF"), breaks = c(2, 1, 0),
    labels = c("Known antigen", "Predicted antigen", "Predicted non-antigen")
  ) +
  scale_color_manual(
    name = "", values = c("#FCB40A", "#FF007F", "#0080FF"), breaks = c(2, 1, 0),
    labels = c("Known antigen", "Predicted antigen", "Predicted non-antigen")
  ) +
  coord_flip() +
  scale_x_discrete(
    breaks = c(
      "pf_single_model", "pv_single_model", "pv_pf_pos",
      "pfpv_combined_model", "pv_pf_unl"
    ),
    labels = c(
      "Heterologous model", "Autologous model",
      "Autologous with\n heterologous positives model",
      "Combined model",
      "Autologous with\n heterologous unlabeled model"
    ),
    limits = rev
  ) +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.title.x = element_text(colour = "black"),
    axis.title.y = element_blank(),
    axis.text = element_text(colour = "black"),
    plot.title = element_text(hjust = 0.5, colour = "black"),
    plot.margin = ggplot2::margin(5, 5, 5, 5, "pt"),
    legend.position = "none"
  ) +
  ylim(0, 1) +
  ylab("Score (proportion of votes)")

```

```

legend <- get_legend(p2 +
  theme(
    legend.direction = "horizontal",
    legend.position = "bottom",
    legend.title = element_blank()
  ) +
  guides(fill = guide_legend(title = "")))

```

```

pf_scores <- read.csv("./other_data/pf_known_antigen_scores.csv", row.names = 1)
pf_unl_scores <- read.csv("./other_data/pf_unlabeled_protein_scores.csv", row.names = 1)
pf_scores$label <- 2
pf_unl_scores$label <- 1
data_ <- melt(rbind(pf_scores, pf_unl_scores), id.vars = "label")
data_$variable <- factor(data_$variable, levels = c(
  "pv_single_model",
  "pf_single_model", "pf_pv_pos",
  "pfpv_combined_model",
  "pf_pv_unl"
))
data_$label[data_$value < 0.5 & data_$label == 1] <- 0
set.seed(12)
p2 <- ggplot(data_, aes(x = variable, y = value)) +
  geom_hline(yintercept = 0.5, color = "grey70", linetype = "dashed") +
  stat_halfeye(aes(fill = factor(label, levels = c(2, 1, 0))),
    slab_color = "black",
    slab_linewidth = 0.2, adjust = 0.5, width = 0.5, .width = 0,
    justification = -0.2, point_colour = NA, alpha = 0.6, normalize = "all"
  ) +
  geom_boxplot(aes(fill = factor(label, levels = c(2, 1, 0))),
    width = 0.2, outlier.color = NA,
    lwd = 0.3, show.legend = FALSE, alpha = 0.6, color = "black"
  ) +
  geom_half_point(
    data = data_[data_$label == 2, ], fill = "#FCB40A", side = "l",
    range_scale = 0.3, alpha = 0.8, shape = 21, color = "black", stroke = 0.1, size = 2
  ) +
  scale_fill_manual(
    name = "", values = c("#FCB40A", "#FF007F", "#0080FF"), breaks = c(2, 1, 0),

```

```

    labels = c("Known antigen", "Predicted antigen", "Predicted non-antigen")
  ) +
  scale_color_manual(
    name = "", values = c("#FCB40A", "#FF007F", "#0080FF"), breaks = c(2, 1, 0),
    labels = c("Known antigen", "Predicted antigen", "Predicted non-antigen")
  ) +
  coord_flip() +
  scale_x_discrete(
    breaks = c(
      "pv_single_model", "pf_single_model", "pf_pv_pos",
      "pfpv_combined_model", "pf_pv_unl"
    ),
    labels = c(
      "Heterologous model", "Autologous model",
      "Autologous with\n heterologous positives model",
      "Combined model",
      "Autologous with\n heterologous unlabeled model"
    ),
    limits = rev
  ) +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.title.x = element_text(colour = "black"),
    axis.title.y = element_blank(),
    axis.text = element_text(colour = "black"),
    plot.title = element_text(hjust = 0.5, colour = "black"),
    plot.margin = ggplot2::margin(5, 5, 5, 5, "pt"),
    legend.title = element_blank(),
    legend.text = element_text(colour = "black"),
    legend.position = "none"
  ) +
  ylim(0, 1) +
  ylab("Score (proportion of votes)")

```

```

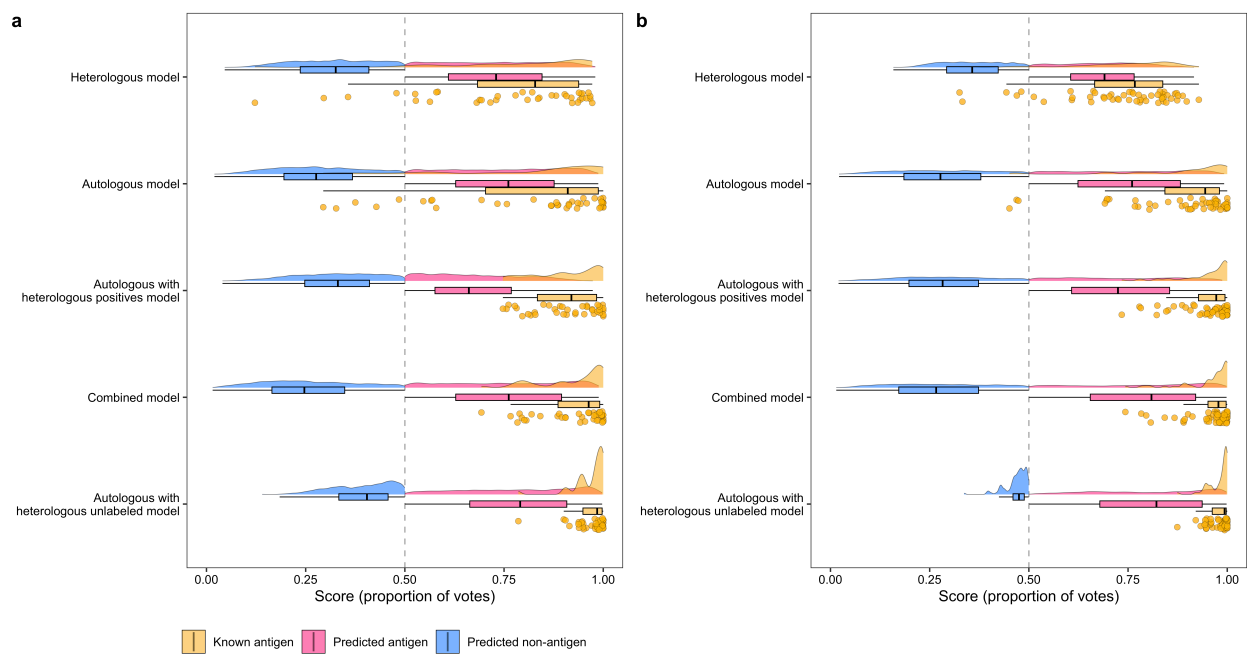
p_combined <- plot_grid(p1, p2, plot_grid(NULL, legend, rel_widths = c(0.2, 0.8)),
  ncol = 2,
  rel_heights = c(1, 0.1), labels = c("a", "b", "", ""))

```

```
)
p_combined
```

```
png(file = "../figures/Fig 2.png", width = 7500, height = 4000, res = 600)
print(p_combined)
dev.off()

pdf(file = "../figures/Fig 2.pdf", width = 13, height = 7)
print(p_combined)
dev.off()
```



4.2 Known antigen prediction summary

In R:

```
library(pracma)
library(ggsci)
```



```

pv_scores <- read.csv("./other_data/pv_known_antigen_scores.csv", row.names = 1)
pv_unl_scores <- read.csv("./other_data/pv_unlabeled_protein_scores.csv", row.names = 1)
pv_scores$label <- 1
pv_unl_scores$label <- 0
pv_all <- rbind(pv_scores, pv_unl_scores)

data <- data.frame(
  "antigen_label" = pv_all$label,
  "Heterologous model" = pv_all$pf_single_model,
  "Autologous model" = pv_all$pv_single_model,
  "Autologous with heterologous positives model" = pv_all$pv_pf_pos,
  "Combined model" = pv_all$pfpv_combined_model,
  "Autologous with heterologous unlabeled model" = pv_all$pv_pf_unl,
  check.names = FALSE
)

# Calculate percent rank for labeled positives
model_names <- c()
x <- c()
percent_rank <- c()
auc <- c()
for (model in colnames(data)[-1]) {
  data_ <- data[c("antigen_label", model)]
  percent_rank_ <- (rank(data_[[model]]) / nrow(data))[data$antigen_label == 1]
  percent_rank <- c(percent_rank, percent_rank_[order(-percent_rank_)])
  data_ <- data_[data$antigen_label == 1, ]
  data_ <- data_[order(-percent_rank_), ]
  x_ <- 1:nrow(data_) / nrow(data_)
  x <- c(x, x_)
  model_names <- c(model_names, rep(model, nrow(data_)))
  auc <- c(auc, round(trapz(c(0, x_, 1), c(1, percent_rank_, 0)), 2))
  cat(paste0("EPR: ", sum(data_[[model]] >= 0.5) / nrow(data_), "\n"))
}
names(auc) <- colnames(data)[-1]

res <- data.frame(model = model_names, x = x, percent_rank = percent_rank)

p1 <- ggplot(res, aes(x = x, y = `percent_rank`, group = model)) +
  geom_hline(yintercept = 0.5, linetype = "dashed", color = "grey50") +
  geom_line(linewidth = 0.1, color = "grey30") +
  geom_point(aes(fill = model),

```

```

    size = 2.2, shape = 21, color = "grey30", stroke = 0.1,
    alpha = 0.8
) +
scale_fill_jco(
  breaks = colnames(data)[-1][order(-auc)],
  labels = sapply(colnames(data)[-1][order(-auc)], function(x) {
    paste0(
      x, " (AUC = ",
      auc[x], ")"
    )
  })
) +
theme_bw() +
theme(
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  axis.title = element_text(colour = "black"),
  axis.text = element_text(colour = "black"),
  plot.title = element_text(hjust = 0.5, colour = "black"),
  legend.title = element_text(hjust = 0.5, colour = "black", angle = 0),
  legend.text = element_text(colour = "black"),
  legend.position = c(0.34, 0.16),
  legend.background = element_blank(),
  legend.key = element_rect(colour = "transparent", fill = "transparent")
) +
labs(fill = "") +
ylim(0, 1) +
xlab("Ranked known antigens (scaled)") +
ylab("Percent rank")

```

```

pf_scores <- read.csv("./other_data/pf_known_antigen_scores.csv", row.names = 1)
pf_unl_scores <- read.csv("./other_data/pf_unlabeled_protein_scores.csv", row.names = 1)
pf_scores$label <- 1
pf_unl_scores$label <- 0
pf_all <- rbind(pf_scores, pf_unl_scores)

data <- data.frame(
  "antigen_label" = pf_all$label,
  "Heterologous model" = pf_all$pv_single_model,

```

```

"Autologous model" = pf_all$pf_single_model,
"Autologous with heterologous positives model" = pf_all$pf_pv_pos,
"Combined model" = pf_all$pfpv_combined_model,
"Autologous with heterologous unlabeled model" = pf_all$pf_pv_unl,
check.names = FALSE
)

# Calculate percent rank for labeled positives
model_names <- c()
x <- c()
percent_rank <- c()
auc <- c()
for (model in colnames(data)[-1]) {
  data_ <- data[c("antigen_label", model)]
  percent_rank_ <- (rank(data_[[model]]) / nrow(data))[data$antigen_label == 1]
  percent_rank <- c(percent_rank, percent_rank_[order(-percent_rank_)])
  data_ <- data_[data$antigen_label == 1, ]
  data_ <- data_[order(-percent_rank_), ]
  x_ <- 1:nrow(data_) / nrow(data_)
  x <- c(x, x_)
  model_names <- c(model_names, rep(model, nrow(data_)))
  auc <- c(auc, round(trapz(c(0, x_, 1), c(1, percent_rank_, 0)), 2))
  cat(paste0("EPR: ", sum(data_[[model]] >= 0.5) / nrow(data_), "\n"))
}
names(auc) <- colnames(data)[-1]

res <- data.frame(model = model_names, x = x, percent_rank = percent_rank)

p2 <- ggplot(res, aes(x = x, y = `percent_rank`, group = model)) +
  geom_hline(yintercept = 0.5, linetype = "dashed", color = "grey50") +
  geom_line(linewidth = 0.1, color = "grey30") +
  geom_point(aes(fill = model),
    size = 2.2, shape = 21, color = "grey30", stroke = 0.1,
    alpha = 0.8
  ) +
  scale_fill_jco(
    breaks = colnames(data)[-1][order(-auc)],
    labels = sapply(colnames(data)[-1][order(-auc)], function(x) {
      paste0(
        x, " (AUC = ",
        auc[x], ")"
      )
    })
  )

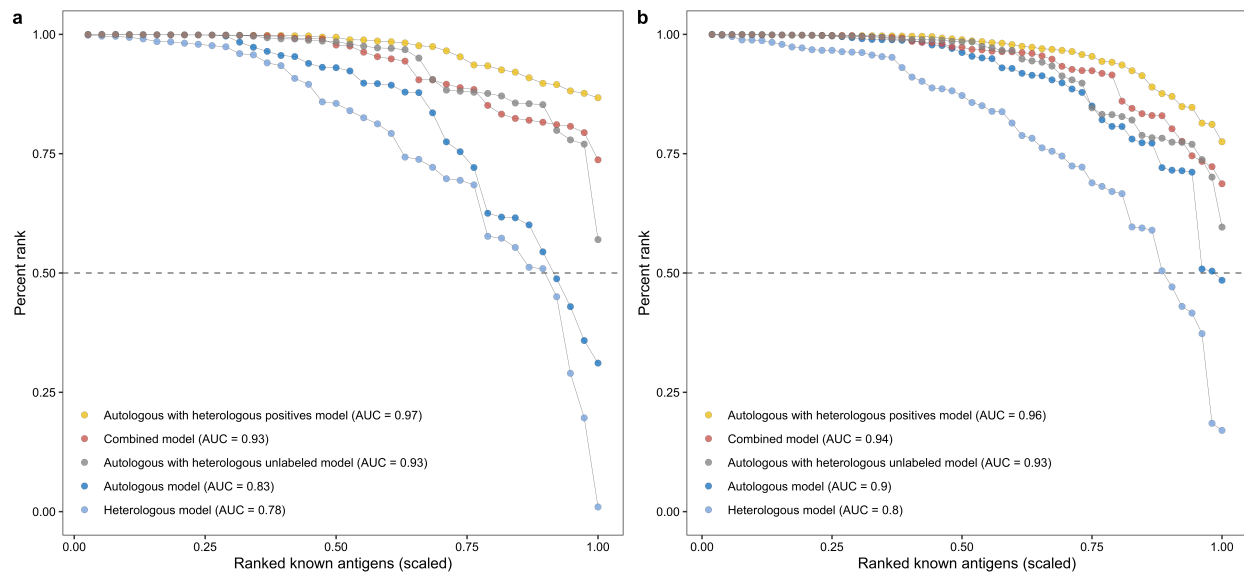
```

```
)
})
) +
theme_bw() +
theme(
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  axis.title = element_text(colour = "black"),
  axis.text = element_text(colour = "black"),
  plot.title = element_text(hjust = 0.5, colour = "black"),
  legend.title = element_text(hjust = 0.5, colour = "black", angle = 0),
  legend.text = element_text(colour = "black"),
  legend.position = c(0.34, 0.16),
  legend.background = element_blank(),
  legend.key = element_rect(colour = "transparent", fill = "transparent")
) +
labs(fill = "") +
ylim(0, 1) +
xlab("Ranked known antigens (scaled)") +
ylab("Percent rank")
```

```
p_combined <- plot_grid(p1, p2, nrow = 1, labels = c("a", "b"))
p_combined
```

```
png(file = "../figures/Supplementary Fig 6.png", width = 7600, height = 3500, res = 600)
print(p_combined)
dev.off()

pdf(file = "../supplementary_figures/Supplementary Fig 6.pdf", width = 15.2, height = 7)
print(p_combined)
dev.off()
```



4.3 Score and species association

In R:

```
library(scales)
library(rlist)
library(rcompanion)
library(DT)
```

```
scientific <- function(x) {
  ifelse(x == 0, "0", gsub("e", " * 10^", scientific_format(digits = 3)(x)))
}
contingency_table <- list()
chisq_pval <- c()
cramer_res <- list()
# Combined
data <- read.csv("./data/supplementary_data_5_pfpv_purf_oob_predictions.csv", check.names = FALSE,
  ↪ row.names = 1)
pred <- sapply(data$`00B score filtered`[data$antigen_label == 0], function(x) if (x >= 0.5) "pos"
  ↪ else "neg")
species <- data$species[data$antigen_label == 0]
M <- table(pred, species)
Xsq <- chisq.test(M, correct = FALSE)
```

```

cramerV <- cramerV(M, ci = TRUE)
contingency_table <- list.append(contingency_table, M)
chisq_pval <- c(chisq_pval, Xsq$p.value)
cramer_res <- list.append(cramer_res, sprintf("%.2f", cramerV))
# P. vivax & P. falciparum single models
data_1 <- read.csv("./other_data/pf_single_pv_single_scores.csv", check.names = FALSE, row.names = 1)
data_2 <- read.csv("./other_data/pf_single_pv_single_cross_predictions.csv", check.names = FALSE,
  ↪ row.names = 1)
pred_1 <- sapply(data_1$`00B score filtered`[data_1$antigen_label == 0], function(x) if (x >= 0.5)
  ↪ "pos" else "neg")
species_1 <- data_1$species[data_1$antigen_label == 0]
pred_2 <- sapply(data_2$`00B score filtered`[data_2$antigen_label == 0], function(x) if (x >= 0.5)
  ↪ "pos" else "neg")
species_2 <- data_2$species[data_2$antigen_label == 0]
## P. vivax
M <- table(
  c(pred_1[species_1 == "pv"], pred_2[species_2 == "pf"]),
  c(rep("pv", length(pred_1[species_1 == "pv"])), rep("pf", length(pred_2[species_2 == "pf"])))
)
Xsq <- chisq.test(M, correct = FALSE)
cramerV <- cramerV(M, ci = TRUE)
contingency_table <- list.append(contingency_table, M)
chisq_pval <- c(chisq_pval, Xsq$p.value)
cramer_res <- list.append(cramer_res, sprintf("%.2f", cramerV))
## P. falciparum
M <- table(
  c(pred_1[species_1 == "pf"], pred_2[species_2 == "pv"]),
  c(rep("pf", length(pred_1[species_1 == "pf"])), rep("pv", length(pred_2[species_2 == "pv"])))
)
Xsq <- chisq.test(M, correct = FALSE)
cramerV <- cramerV(M, ci = TRUE)
contingency_table <- list.append(contingency_table, M)
chisq_pval <- c(chisq_pval, Xsq$p.value)
cramer_res <- list.append(cramer_res, sprintf("%.2f", cramerV))
# P. vivax with heterologous positives
data <- read.csv("./other_data/pv_pf_pos_scores.csv", check.names = FALSE, row.names = 1)
pred <- sapply(data$`00B scores`[data$antigen_label == 0], function(x) if (x >= 0.5) "pos" else "neg")
species <- data$species[data$antigen_label == 0]
M <- table(pred, species)
Xsq <- chisq.test(M, correct = FALSE)
cramerV <- cramerV(M, ci = TRUE)

```

```

contingency_table <- list.append(contingency_table, M)
chisq_pval <- c(chisq_pval, Xsq$p.value)
cramer_res <- list.append(cramer_res, sprintf("%.2f", cramerV))
# P. falciparum with heterologous positives
data <- read.csv("./other_data/pf_pv_pos_scores.csv", check.names = FALSE, row.names = 1)
pred <- sapply(data$`00B scores`[data$antigen_label == 0], function(x) if (x >= 0.5) "pos" else "neg")
species <- data$species[data$antigen_label == 0]
M <- table(pred, species)
Xsq <- chisq.test(M, correct = FALSE)
cramerV <- cramerV(M, ci = TRUE)
contingency_table <- list.append(contingency_table, M)
chisq_pval <- c(chisq_pval, Xsq$p.value)
cramer_res <- list.append(cramer_res, sprintf("%.2f", cramerV))
# P. vivax with heterologous unlabeled
data <- read.csv("./other_data/pv_pf_unl_scores.csv", check.names = FALSE, row.names = 1)
pred <- sapply(data$`00B scores`[data$antigen_label == 0], function(x) if (x >= 0.5) "pos" else "neg")
species <- data$species[data$antigen_label == 0]
M <- table(pred, species)
Xsq <- chisq.test(M, correct = FALSE)
cramerV <- cramerV(M, ci = TRUE)
contingency_table <- list.append(contingency_table, M)
chisq_pval <- c(chisq_pval, Xsq$p.value)
cramer_res <- list.append(cramer_res, sprintf("%.2f", cramerV))
# P. falciparum with heterologous unlabeled
data <- read.csv("./other_data/pf_pv_unl_scores.csv", check.names = FALSE, row.names = 1)
pred <- sapply(data$`00B scores`[data$antigen_label == 0], function(x) if (x >= 0.5) "pos" else "neg")
species <- data$species[data$antigen_label == 0]
M <- table(pred, species)
Xsq <- chisq.test(M, correct = FALSE)
cramerV <- cramerV(M, ci = TRUE)
contingency_table <- list.append(contingency_table, M)
chisq_pval <- c(chisq_pval, Xsq$p.value)
cramer_res <- list.append(cramer_res, sprintf("%.2f", cramerV))
# Save results
df <- as.data.frame(cbind(
  c(
    "Combined", "P. vivax", "P. falciparum",
    "P. vivax with heterologous positives",
    "P. falciparum with heterologous positives",
    "P. vivax with heterologous unlabeled",
    "P. falciparum with heterologous unlabeled"
  )

```

```

    ),
    chisq_pval, do.call(rbind, cramer_res)
))
colnames(df) <- c("PURF model", "Chi-squared test p-value", "Cramer's V", "Lower CI", "Upper CI")
df$`Chi-squared test p-value` <- sapply(as.numeric(df$`Chi-squared test p-value`), scientific)
names(contingency_table) <- df$`PURF model`
save(contingency_table, df, file = "./rdata/score_species_association.RData")

```

```

load("./rdata/score_species_association.RData")
contingency_table

```

```

## $Combined
##      species
## pred   pf   pv
##  neg 2719 3814
##  pos 2622 2639
##
## $`P. falciparum`
##
##      pf   pv
##  neg 2897 3091
##  pos 2444 3362
##
## $`P. vivax`
##
##      pf   pv
##  neg 2451 3618
##  pos 2890 2835
##
## $`P. falciparum with heterologous positives`
##      species
## pred   pf   pv
##  neg 2932   63
##  pos 2409 6390
##
## $`P. vivax with heterologous positives`
##      species
## pred   pf   pv
##  neg    0 3695
##  pos 5341 2758

```



```
##
## $`P. falciparum with heterologous unlabeled`
##      species
## pred   pf   pv
##  neg  108 6314
##  pos 5233  139
##
## $`P. vivax with heterologous unlabeled`
##      species
## pred   pf   pv
##  neg 5133 1055
##  pos  208 5398
```

```
df %>%
  datatable(rownames = FALSE)
```

PURF model	Chi-squared test p-value	Cramer's V	Lower CI	Upper CI
---------------	-----------------------------	---------------	-------------	-------------

4.4 Tree depth analysis

4.4.1 Analysis

In Python:

```
library(reticulate)
use_condaenv("/Users/renee/Library/r-miniconda/envs/purf/bin/python")
```

```
import pandas as pd
import numpy as np
import pickle
from purf.pu_ensemble import PURandomForestClassifier
import session_info
```

```
with open('./pickle_data/pf_0.5_purf_tree_filtering.pkl', 'rb') as infile:
    pf_single_model = pickle.load(infile)
```

```
with open('./pickle_data/pv_0.5_purf_tree_filtering.pkl', 'rb') as infile:
```

```

pv_single_model = pickle.load(infile)

with open('./pickle_data/pfpv_0.5_purf_tree_filtering.pkl', 'rb') as infile:
    pfpv_combined_model = pickle.load(infile)

with open('./pickle_data/pf_pv_pos_purf.pkl', 'rb') as infile:
    pf_pv_pos_model = pickle.load(infile)

with open('./pickle_data/pf_pv_unl_purf.pkl', 'rb') as infile:
    pf_pv_unl_model = pickle.load(infile)

with open('./pickle_data/pv_pf_pos_purf.pkl', 'rb') as infile:
    pv_pf_pos_model = pickle.load(infile)

with open('./pickle_data/pv_pf_unl_purf.pkl', 'rb') as infile:
    pv_pf_unl_model = pickle.load(infile)

res = pd.DataFrame({'pf_single_model': [tree.get_depth() for tree in
↪ pf_single_model['model'].estimators_],
                    'pv_single_model': [tree.get_depth() for tree in
↪ pv_single_model['model'].estimators_],
                    'pfpv_combined_model': [tree.get_depth() for tree in
↪ pfpv_combined_model['model'].estimators_],
                    'pf_pv_pos_model': [tree.get_depth() for tree in
↪ pf_pv_pos_model['model'].estimators_],
                    'pf_pv_unl_model': [tree.get_depth() for tree in
↪ pf_pv_unl_model['model'].estimators_],
                    'pv_pf_pos_model': [tree.get_depth() for tree in
↪ pv_pf_pos_model['model'].estimators_],
                    'pv_pf_unl_model': [tree.get_depth() for tree in
↪ pv_pf_unl_model['model'].estimators_]})

res.to_csv('./other_data/tree_depths.csv', index=False)

```

In R

4.4.2 Plotting

[illegible]

```
##  
## Residual standard error: 0.4194 on 5 degrees of freedom  
## Multiple R-squared: 0.5965, Adjusted R-squared: 0.5158  
## F-statistic: 7.391 on 1 and 5 DF, p-value: 0.04184  
  
cat("~*****~\n")  
## ~*****~  
cat("Logit transform for proportion positives\n")  
## Logit transform for proportion positives  
cat("~*****~\n")  
## ~*****~  
res <- lm(mean_tree_depth ~ proportion_positives,  
  data = data.frame(  
    proportion_positives = logit(proportion_positives),  
    mean_tree_depth = mean_tree_depth  
  )  
)  
summary(res)  
##  
## Call:  
## lm(formula = mean_tree_depth ~ proportion_positives, data = data.frame(proportion_positives =  
  ↪ logit(proportion_positives),  
##     mean_tree_depth = mean_tree_depth))  
##  
## Residuals:  
##      1       2       3       4       5       6       7  
## -0.6182 -0.7549 -1.0271 -0.0954 -0.1369 -0.2818  2.9143  
##  
## Coefficients:  
##              Estimate Std. Error t value Pr(>|t|)  
## (Intercept)      22.7794      4.8653   4.682  0.00542 **  
## proportion_positives  3.3428      0.9906   3.375  0.01979 *  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## Residual standard error: 1.457 on 5 degrees of freedom  
## Multiple R-squared: 0.6949, Adjusted R-squared: 0.6339  
## F-statistic: 11.39 on 1 and 5 DF, p-value: 0.01979  
  
cat::~~~~~~\n")  
## ~~~~~
```

```
cat("Arc-sine transform for proportion positives\n")
## Arc-sine transform for proportion positives
cat("~~~~~\n")
## ~~~~~
res <- lm(mean_tree_depth ~ proportion_positives,
  data = data.frame(
    proportion_positives = asin(sqrt(proportion_positives)),
    mean_tree_depth = mean_tree_depth
  )
)
summary(res)
##
## Call:
## lm(formula = mean_tree_depth ~ proportion_positives, data = data.frame(proportion_positives =
  ↪ asin(sqrt(proportion_positives)),
##     mean_tree_depth = mean_tree_depth))
##
## Residuals:
##      1      2      3      4      5      6      7
## -0.37586 -0.90400 -1.11266  0.04187 -0.08305 -0.72069  3.15439
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -0.06706     2.29355  -0.029   0.9778
## proportion_positives 72.39644    24.52407   2.952   0.0318 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.592 on 5 degrees of freedom
## Multiple R-squared:  0.6354, Adjusted R-squared:  0.5625
## F-statistic: 8.715 on 1 and 5 DF,  p-value: 0.03181
cat("~~~~~\n")
## ~~~~~
cat("Logit transform for proportion positives & log transform for tree depth\n")
## Logit transform for proportion positives & log transform for tree depth
cat("~~~~~\n")
## ~~~~~
res <- lm(mean_tree_depth ~ proportion_positives,
  data = data.frame(
    proportion_positives = logit(proportion_positives),
```

```

mean_tree_depth = log2(mean_tree_depth)
)
)
summary(res)
##
## Call:
## lm(formula = mean_tree_depth ~ proportion_positives, data = data.frame(proportion_positives =
↪ logit(proportion_positives),
##     mean_tree_depth = log2(mean_tree_depth)))
##
## Residuals:
##      1      2      3      4      5      6      7
## -0.06899 -0.21823 -0.26484  0.09247 -0.05659 -0.11887  0.63505
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)      6.8270      1.1096   6.153  0.00165 **
## proportion_positives  0.8676      0.2259   3.840  0.01212 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3322 on 5 degrees of freedom
## Multiple R-squared:  0.7468, Adjusted R-squared:  0.6962
## F-statistic: 14.75 on 1 and 5 DF,  p-value: 0.01212

cat("~~~~~\n")
## ~~~~~
cat("Arc-sine transform for proportion positives & log transform for tree depth\n")
## Arc-sine transform for proportion positives & log transform for tree depth
cat("~~~~~\n")
## ~~~~~
res <- lm(mean_tree_depth ~ proportion_positives,
  data = data.frame(
    proportion_positives = asin(sqrt(proportion_positives)),
    mean_tree_depth = log2(mean_tree_depth)
  )
)
summary(res)
##
## Call:
## lm(formula = mean_tree_depth ~ proportion_positives, data = data.frame(proportion_positives =
↪ asin(sqrt(proportion_positives)),

```

```
##      mean_tree_depth = log2(mean_tree_depth))
##
## Residuals:
##      1      2      3      4      5      6      7
## -0.00527 -0.25295 -0.28949  0.12670 -0.03982 -0.23622  0.69705
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)      0.9066     0.5417   1.673   0.1551
## proportion_positives 18.6878     5.7926   3.226   0.0233 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3761 on 5 degrees of freedom
## Multiple R-squared:  0.6755, Adjusted R-squared:  0.6106
## F-statistic: 10.41 on 1 and 5 DF,  p-value: 0.02331
```

```
data <- data.frame(
  proportion_positives = logit(proportion_positives),
  mean_tree_depth = log2(mean_tree_depth),
  upper_error_tree_depth = log2(mean_tree_depth + sd_tree_depth),
  lower_error_tree_depth = log2(mean_tree_depth - sd_tree_depth)
)
data$model <- c(
  "italic(P.~falciparum)~model",
  "italic(P.~falciparum)~with~heterologous~positives~model",
  "italic(P.~falciparum)~with~heterologous~unlabeled~model",
  "italic(P.~vivax)~model",
  "italic(P.~vivax)~with~heterologous~positives~model",
  "italic(P.~vivax)~with~heterologous~unlabeled~model",
  "Combined~model"
)

lm_eqn <- function(data) {
  m <- lm(mean_tree_depth ~ proportion_positives, data)
  eq <- substitute(
    log[2](italic(y)) == a + b %.% logit(italic(x)) * "," ~ ~ italic(r^2) ~ "=" ~ r2,
    list(
      a = format(unname(coef(m)[1]), digits = 2),
```



```

    b = format(unname(coef(m)[2]), digits = 2),
    r2 = format(summary(m)$adj.r.squared, digits = 2)
  )
)
as.character(as.expression(eq))
}

p <- ggplot(data, aes(proportion_positives, mean_tree_depth)) +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE, color = "grey50", linetype = "dashed",
    ↪ linewidth = 0.5) +
  geom_point(color = "grey10", size = 1.5, alpha = 0.8) +
  geom_errorbar(aes(ymin = lower_error_tree_depth, ymax = upper_error_tree_depth), width = 0.05,
    ↪ linewidth = 0.2) +
  geom_text_repel(aes(label = model),
    size = 2.5, parse = TRUE, segment.size = 0.2,
    nudge_x = 0.02, nudge_y = -0.15, point.padding = 0.1
  ) +
  geom_text(x = -5.5, y = 3.5, label = lm_eqn(data), parse = TRUE, size = 3) +
  scale_x_continuous(
    breaks = logit(c(0.004, 0.008, 0.012, 0.016)),
    labels = c(0.004, 0.008, 0.012, 0.016)
  ) +
  scale_y_continuous(breaks = log2(c(0.25, 0.5, 2, 4, 8)), labels = c(0.25, 0.5, 2, 4, 8)) +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    plot.title = element_text(hjust = 0.5, colour = "black"),
    plot.margin = ggplot2::margin(5, 5, 5, 5, "pt"),
    axis.title = element_text(colour = "black"),
    axis.text = element_text(colour = "black")
  ) +
  xlab("Proportion of positives") +
  ylab("Mean tree depth")

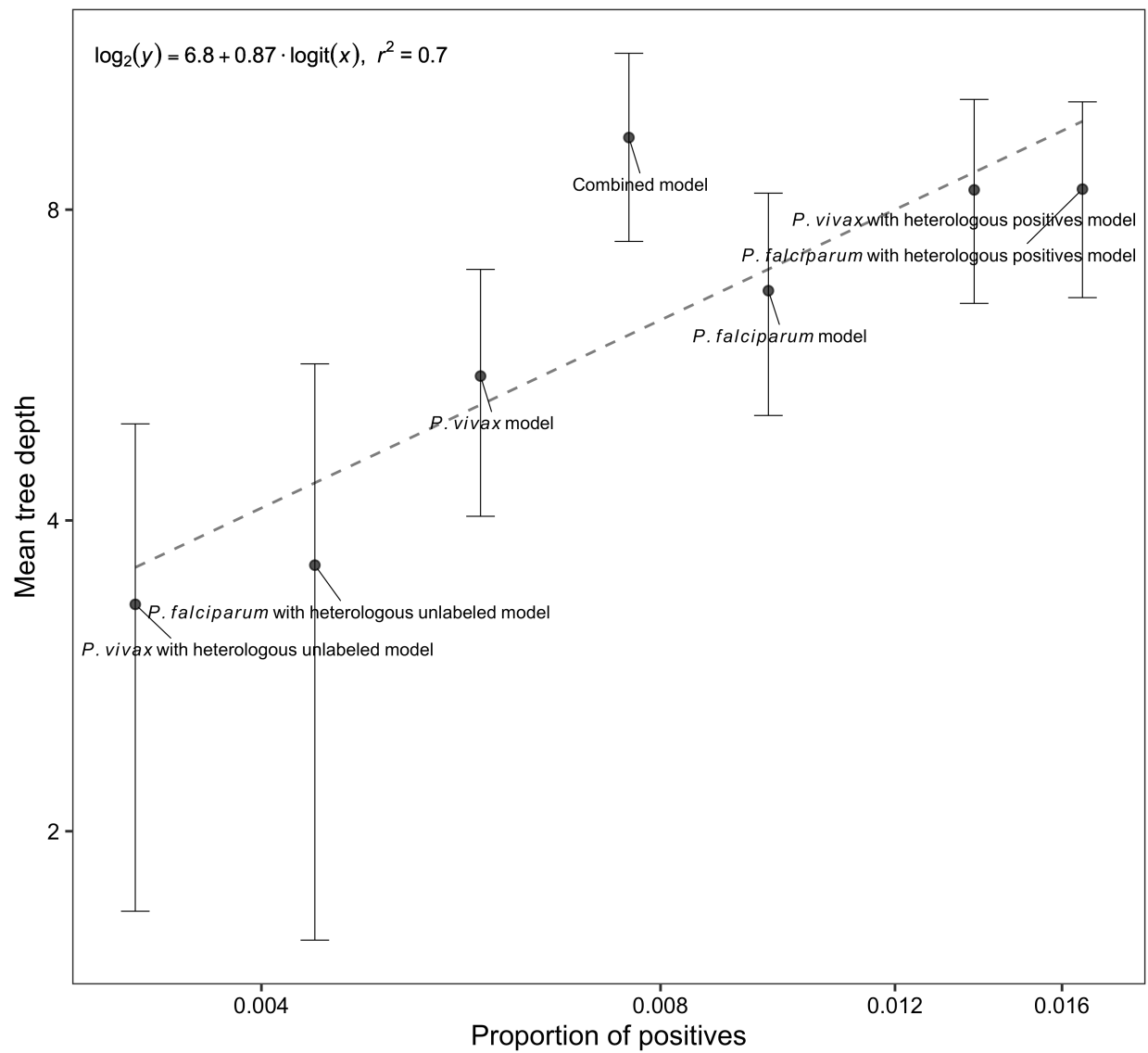
```

```

png(file = "./figures/Supplementary Fig 7.png", width = 3800, height = 3500, res = 600)
print(p)
dev.off()

```

```
pdf(file = "../supplementary_figures/Supplementary Fig 7.pdf", width = 7.6, height = 7)
print(p)
dev.off()
```



```
sessionInfo()
```

```
## R version 4.2.3 (2023-03-15)
```

```
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur ... 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats      graphics  grDevices  utils      datasets  methods   base
##
## other attached packages:
## [1] ggrepel_0.9.3      psych_2.3.3      reticulate_1.28   DT_0.27
## [5] rcompanion_2.4.30  rlist_0.4.6.2    scales_1.2.1      ggsci_3.0.0
## [9] pracma_2.4.2       cowplot_1.1.1    gghalves_0.1.4    ggdist_3.2.1
## [13] reshape2_1.4.4     ggplot2_3.4.2
##
## loaded via a namespace (and not attached):
## [1] nlme_3.1-162      matrixStats_0.63.0  webshot_0.5.4
## [4] http_1.4.6        rprojroot_2.0.3     R.cache_0.16.0
## [7] tools_4.2.3       bslib_0.4.2         utf8_1.2.3
## [10] R6_2.5.1          nortest_1.0-4       colorspace_2.1-0
## [13] withr_2.5.0       mnormt_2.1.1        tidyselect_1.2.0
## [16] Exact_3.2         processx_3.8.1      compiler_4.2.3
## [19] cli_3.6.1         expm_0.999-7        sandwich_3.0-2
## [22] bookdown_0.34     sass_0.4.6          lmtest_0.9-40
## [25] mvtnorm_1.1-3     callr_3.7.3         proxy_0.4-27
## [28] multcompView_0.1-9 stringr_1.5.0        digest_0.6.31
## [31] rmarkdown_2.21    R.utils_2.12.2      pkgconfig_2.0.3
## [34] htmltools_0.5.5   styler_1.9.1        fastmap_1.1.1
## [37] highr_0.10        htmlwidgets_1.6.2   rlang_1.1.1
## [40] readxl_1.4.2      rstudioapi_0.14     jquerylib_0.1.4
## [43] farver_2.1.1      generics_0.1.3      zoo_1.8-12
## [46] jsonlite_1.8.4    crosstalk_1.2.0     dplyr_1.1.2
## [49] R.oo_1.25.0       distributional_0.3.2 magrittr_2.0.3
## [52] modeltools_0.2-23 Matrix_1.5-4         Rcpp_1.0.10
## [55] DescTools_0.99.48 munsell_0.5.0       fansi_1.0.4
## [58] lifecycle_1.0.3   R.methodsS3_1.8.2   stringi_1.7.12
## [61] multcomp_1.4-23   yaml_2.3.7          MASS_7.3-60
## [64] rootSolve_1.8.2.3 plyr_1.8.8          grid_4.2.3
```

```
## [67] parallel_4.2.3      lmom_2.9             lattice_0.21-8
## [70] splines_4.2.3       knitr_1.42           ps_1.7.5
## [73] pillar_1.9.0        boot_1.3-28.1        gld_2.6.6
## [76] codetools_0.2-19    stats4_4.2.3         glue_1.6.2
## [79] evaluate_0.21       data.table_1.14.8    png_0.1-8
## [82] vctrs_0.6.2         cellranger_1.1.0     gtable_0.3.3
## [85] purrr_1.0.1         cachem_1.0.8         xfun_0.39
## [88] coin_1.4-2          libcoin_1.0-9        e1071_1.7-13
## [91] class_7.3-22        survival_3.5-5       tibble_3.2.1
## [94] TH.data_1.1-2       ellipsis_0.3.2       here_1.0.1
```

```
session_info.show()
```

```
## -----
## numpy          1.19.0
## pandas         1.3.2
## purf           NA
## session_info   1.0.0
## -----
## Python 3.8.2 (default, Mar 26 2020, 10:45:18) [Clang 4.0.1 (tags/RELEASE_401/final)]
## macOS-10.16-x86_64-i386-64bit
## -----
## Session information updated at 2023-05-18 12:31
```

Section 5

Model interpretation

5.1 Proximity matrix calculation

5.1.1 Analysis

In Python:

```
library(reticulate)
use_condaenv("/Users/renee/Library/r-miniconda/envs/purf/bin/python")
```

```
import pandas as pd
import numpy as np
import pickle
from purf.pu_ensemble import PURandomForestClassifier
from sklearn.impute import SimpleImputer
from sklearn.ensemble._forest import _generate_unsampled_indices
import multiprocessing
from joblib import Parallel, delayed
num_cores = multiprocessing.cpu_count()
import session_info
```

5.1.1.1 Pv data set

```
data = pd.read_csv('./other_data/pv_ml_input.csv', index_col=0)
features = data.iloc[:, 1:]
outcome = np.array(data.antigen_label)

# Imputation
imputer = SimpleImputer(strategy='median')
X = imputer.fit_transform(features)

# Load model
```

```

model_tree_filtered = pickle.load(open('./pickle_data/pv_0.5_purf_tree_filtering.pkl', 'rb'))
model = model_tree_filtered['model']
weights = model_tree_filtered['weights']

def calculate_proximity_(trees, y, leaf_indices):
    same_leaf_mat = np.zeros([y.shape[0], y.shape[0]])
    same_oob_mat = np.zeros([y.shape[0], y.shape[0]])
    for idx, tree in enumerate(trees):
        oob_indices = _generate_unsampled_indices(tree.random_state, y.shape[0], y.shape[0])
        for i in oob_indices:
            for j in oob_indices:
                same_oob_mat[i, j] += 1
                if leaf_indices[i, idx] == leaf_indices[j, idx]:
                    same_leaf_mat[i, j] += 1
    return (same_leaf_mat, same_oob_mat)

trees = model.estimators_
leaf_indices = model.apply(X)
idx_list = [i for i in range(len(trees)) if weights[i] == 1]
idx_list = np.array_split(idx_list, 1000)

res = Parallel(n_jobs=num_cores)(
    delayed(calculate_proximity_)(list(trees[i] for i in idx), outcome, leaf_indices[:, idx]) for idx
    ↪ in idx_list)

same_leaf_mat = sum([r[0] for r in res])
same_oob_mat = sum([r[1] for r in res])

prox_mat = pd.DataFrame(np.divide(same_leaf_mat, same_oob_mat, out=np.zeros_like(same_leaf_mat),
                                where=same_oob_mat!=0))
prox_mat.index = features.index
prox_mat.columns = features.index
prox_mat.to_csv('~Downloads/pv_proximity_values.csv', index=False)

```

5.1.1.2 Pv + Pf combined data set

```

data = pd.read_csv('./data/supplementary_data_4_pfpv_ml_input.csv', index_col=0)
features = data.iloc[:, 1:]
outcome = np.array(data.antigen_label)

```

```

# Imputation
imputer = SimpleImputer(strategy='median')
X = imputer.fit_transform(features)

# Load model
model_tree_filtered = pickle.load(open('./pickle_data/pfpv_0.5_purf_tree_filtering.pkl', 'rb'))
model = model_tree_filtered['model']
weights = model_tree_filtered['weights']

def calculate_proximity_(trees, y, leaf_indices):
    same_leaf_mat = np.zeros([y.shape[0], y.shape[0]])
    same_oob_mat = np.zeros([y.shape[0], y.shape[0]])
    for idx, tree in enumerate(trees):
        oob_indices = _generate_unsampled_indices(tree.random_state, y.shape[0], y.shape[0])
        for i in oob_indices:
            for j in oob_indices:
                same_oob_mat[i, j] += 1
                if leaf_indices[i, idx] == leaf_indices[j, idx]:
                    same_leaf_mat[i, j] += 1
    return (same_leaf_mat, same_oob_mat)

trees = model.estimators_
leaf_indices = model.apply(X)
idx_list = [i for i in range(len(trees)) if weights[i] == 1]
idx_list = np.array_split(idx_list, 1000)

res = Parallel(n_jobs=num_cores)(
    delayed(calculate_proximity_)(list(trees[i] for i in idx), outcome, leaf_indices[:, idx]) for idx
    ↪ in idx_list)

same_leaf_mat = sum([r[0] for r in res])
same_oob_mat = sum([r[1] for r in res])

prox_mat = pd.DataFrame(np.divide(same_leaf_mat, same_oob_mat, out=np.zeros_like(same_leaf_mat),
                                where=same_oob_mat!=0))
prox_mat.index = features.index
prox_mat.columns = features.index
prox_mat.to_csv('~Downloads/pfpv_proximity_values.csv', index=False)

```

In R:

```

# Multidimensional scaling
prox_mat <- read.csv("~/Downloads/pv_proximity_values.csv", check.names = FALSE)
mds <- cmdscale(as.dist(1 - prox_mat), k = ncol(prox_mat) - 1, eig = TRUE)
var_explained <- round(mds$eig * 100 / sum(mds$eig), 2)
print(paste0("Dimension 1 (", var_explained[1], "%)") # 41.44%
print(paste0("Dimension 2 (", var_explained[2], "%)") # 8.64%
print(paste0("Dimension 3 (", var_explained[3], "%)") # 4.72%

mds <- as.data.frame(mds$points)
write.csv(mds, "~/Downloads/mds_pv_proximity_values.csv")

prox_mat <- read.csv("~/Downloads/pfpv_proximity_values.csv", check.names = FALSE)
mds <- cmdscale(as.dist(1 - prox_mat), k = ncol(prox_mat) - 1, eig = TRUE)
var_explained <- round(mds$eig * 100 / sum(mds$eig), 2)
print(paste0("Dimension 1 (", var_explained[1], "%)") # 37.91%
print(paste0("Dimension 2 (", var_explained[2], "%)") # 7.88%
print(paste0("Dimension 3 (", var_explained[3], "%)") # 4.79%

mds <- as.data.frame(mds$points)
write.csv(mds, "~/Downloads/mds_pfpv_proximity_values.csv")

```

5.1.2 Plotting

In R:

```

library(umap)
library(ggplot2)
library(ggrepel)
library(cowplot)

```

5.1.2.1 Pv + Pf combined data set

```

data <- read.csv("~/Downloads/mds_pfpv_proximity_values.csv", row.names = 1, check.names = FALSE)

set.seed(22)
umap_res <- umap(data)
umap_df <- data.frame(umap_res$layout)
save(umap_df, file = "./rdata/pfpv_prox_mds_umap.RData")

```



```

data <- read.csv("../data/supplementary_data_5_pfpv_purf_oob_predictions.csv", row.names = 1,
  ↪ check.names = FALSE)[5394:11884, ]
load("../rdata/pfpv_prox_mds_umap.RData")
umap_df_ <- umap_df[1:5393, ]
umap_df <- umap_df[5394:11884, ]
umap_df$label <- data$antigen_label
umap_df$score <- data$`00B score filtered`
umap_df_text <- umap_df[umap_df$label == 1, ]
umap_df_text$label_text <- ""
umap_df_text$label_text[rownames(umap_df_text) == "PVP01_0835600.1-p1"] <- "CSP"
umap_df_text$label_text[rownames(umap_df_text) == "PVP01_0623800.1-p1"] <- "DBP"
umap_df_text$label_text[rownames(umap_df_text) == "PVP01_0728900.1-p1"] <- "MSP1"

umap_p1 <- ggplot() +
  geom_point(data = umap_df_, aes(x = X1, y = X2), color = "grey90", alpha = 0.5) +
  geom_point(data = umap_df[umap_df$label == 0, ], aes(x = X1, y = X2, color = score), alpha = 0.5) +
  geom_point(
    data = umap_df[umap_df$label == 1, ], aes(x = X1, y = X2, shape = factor(21)), color = "black",
    fill = "#FCB40A", stroke = 0.3, size = 2, alpha = 0.9
  ) +
  geom_text_repel(
    data = umap_df_text, aes(x = X1, y = X2, label = label_text), size = 3.5,
    nudge_x = 1, nudge_y = -1, point.padding = 0.1
  ) +
  scale_color_gradient2(low = "#3399FF", mid = "white", high = "#FF3399", midpoint = 0.5, limits = c(0,
    ↪ 1)) +
  scale_shape_manual(values = c(21), labels = c("Known antigens")) +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    plot.margin = ggplot2::margin(5, 5, 5, 5, "pt"),
    axis.title = element_text(colour = "black"),
    axis.text = element_text(colour = "black"),
    legend.position = "none"
  ) +
  xlab("Dimension 1") +
  ylab("Dimension 2")

```

```

data <- read.csv("../data/supplementary_data_5_pfpv_purf_oob_predictions.csv", row.names = 1,
  ↪ check.names = FALSE)[1:5393, ]
load("../rdata/pfpv_prox_mds_umap.RData")
umap_df_ <- umap_df[5394:11884, ]
umap_df <- umap_df[1:5393, ]
umap_df$label <- data$antigen_label
umap_df$score <- data$`O0B score filtered`
umap_df_text <- umap_df[umap_df$label == 1, ]
umap_df_text$label_text <- ""
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0304600.1-p1"] <- "CSP"
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0424100.1-p1"] <- "RH5"
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0206900.1-p1"] <- "MSP5"
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0209000.1-p1"] <- "P230"

umap_p2 <- ggplot() +
  geom_point(data = umap_df_, aes(x = X1, y = X2), color = "grey90", alpha = 0.5) +
  geom_point(data = umap_df[umap_df$label == 0, ], aes(x = X1, y = X2, color = score), alpha = 0.5) +
  geom_point(
    data = umap_df[umap_df$label == 1, ], aes(x = X1, y = X2), shape = 21, color = "black",
    fill = "#FCB40A", stroke = 0.3, size = 2, alpha = 0.9
  ) +
  geom_text_repel(
    data = umap_df_text, aes(x = X1, y = X2, label = label_text), size = 3.5,
    nudge_x = 1.5, nudge_y = -1, point.padding = 0.1
  ) +
  scale_color_gradient2(low = "#3399FF", mid = "white", high = "#FF3399", midpoint = 0.5, limits = c(0,
    ↪ 1)) +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    plot.margin = ggplot2::margin(5, 5, 5, 5, "pt"),
    axis.title = element_text(colour = "black"),
    axis.text = element_text(colour = "black"),
    legend.position = "none"
  ) +
  xlab("Dimension 1") +
  ylab("Dimension 2")

legend <- get_legend(umap_p2 +
  theme(

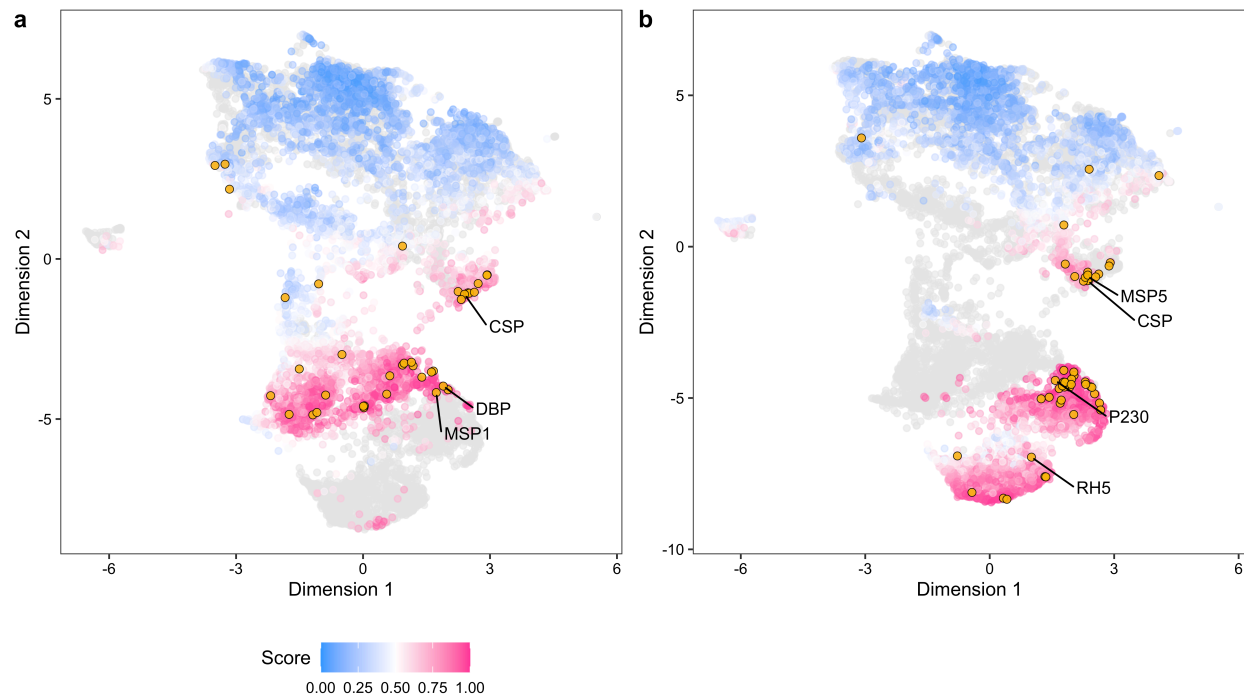
```

```
    legend.direction = "horizontal",
    legend.position = "bottom",
    legend.title = element_text(vjust = 0.8, colour = "black")
  ) +
  guides(
    color = guide_colorbar(title = "Score"),
    shape = guide_legend(title = "")
  )
))
```

```
umap_p_combined <- plot_grid(umap_p1, umap_p2,
  plot_grid(NULL, legend, rel_widths = c(0.15, 0.85)),
  ncol = 2,
  rel_heights = c(1, 0.2), labels = c("a", "b", "", ""))
)
umap_p_combined
```

```
png(file = "./figures/Fig 3.png", width = 6000, height = 3500, res = 600)
print(umap_p_combined)
dev.off()

pdf(file = "./figures/Fig 3.pdf", width = 12, height = 7)
print(umap_p_combined)
dev.off()
```



5.2 Clustering of predicted positives

5.2.1 Analysis

In R:

```
library(DT)
library(factoextra)
library(NbClust)
library(cluster)
library(dendextend)
library(umap)
library(ggplot2)
library(ggrepel)
library(cowplot)
library(scales)
library(rcompanion)
```

```

scientific <- function(x) {
  ifelse(x == 0, "0", gsub("e", " * 10^", scientific_format(digits = 3)(x)))
}

calculate_association <- function(clusters) {
  species <- sapply(names(clusters), function(x) if (startsWith(x, "PF3D7")) "pf" else "pv")
  M <- table(clusters, species)
  Xsq <- chisq.test(M, correct = FALSE)
  cramerV <- cramerV(M, ci = TRUE)
  return(list(
    M = M, xsq_pval = scientific(as.numeric(Xsq$p.value)),
    cramerV = sprintf("%.2f", cramerV)
  ))
}

```

```

pfpv_data <- read.csv("./data/supplementary_data_5_pfpv_purf_oob_predictions.csv", row.names = 1,
  ↪ check.names = FALSE)
pred_pos <- rownames(pfpv_data)[pfpv_data$`OOB score filtered` >= 0.5]
pfpv_prox <- read.csv("~/Downloads/pfpv_proximity_values.csv", check.names = FALSE)
pred_pos_dist <- 1 - pfpv_prox
rownames(pred_pos_dist) <- colnames(pred_pos_dist)
pred_pos_dist <- pred_pos_dist[rownames(pred_pos_dist) %in% pred_pos, colnames(pred_pos_dist) %in%
  ↪ pred_pos]

```

```

pred_pos_hc <- hclust(as.dist(pred_pos_dist), method = "ward.D2")
save(pred_pos_hc, file = "./rdata/pred_pos_clusters.RData")

```

```

load(file = "./rdata/pred_pos_clusters.RData")
clusters <- cutree(pred_pos_hc, k = 2)
calculate_association(clusters)

```

```

## $M
##      species
## clusters  pf  pv
##      1 1695 1918
##      2   979  759
##

```

```
## $xsq_pval
## [1] "1.11 * 10^-10"
##
## $cramerV
## [1] "0.09" "0.06" "0.12"
```

```
data <- read.csv("./data/supplementary_data_5_pfpv_purf_oob_predictions.csv", row.names = 1,
  ↪ check.names = FALSE)
load("./rdata/pfpv_prox_mds_umap.RData")
umap_df$label <- data$antigen_label
umap_df_ <- umap_df[!rownames(umap_df) %in% names(clusters), ]
umap_df <- rbind(umap_df[umap_df$label == 1, ], umap_df[rownames(umap_df) %in% names(clusters), ])
umap_df$group <- ""
umap_df$group[umap_df$label == 1] <- "Known antigen"
umap_df$group[rownames(umap_df) %in% c("PF3D7_0304600.1-p1", "PF3D7_0424100.1-p1",
  ↪ "PF3D7_0206900.1-p1", "PF3D7_0209000.1-p1", "PVP01_0835600.1-p1", "PVP01_0623800.1-p1",
  ↪ "PVP01_0728900.1-p1")] <- "Reference antigen"
umap_df$group[rownames(umap_df) %in% names(clusters)[clusters == 1]] <- "Group 1"
umap_df$group[rownames(umap_df) %in% names(clusters)[clusters == 2]] <- "Group 2"
umap_df_text <- umap_df[umap_df$label == 1, ]
umap_df_text$label_text <- ""
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0304600.1-p1"] <- "PfCSP"
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0424100.1-p1"] <- "RH5"
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0206900.1-p1"] <- "MSP5"
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0209000.1-p1"] <- "P230"
umap_df_text$label_text[rownames(umap_df_text) == "PVP01_0835600.1-p1"] <- "PvCSP"
umap_df_text$label_text[rownames(umap_df_text) == "PVP01_0623800.1-p1"] <- "DBP"
umap_df_text$label_text[rownames(umap_df_text) == "PVP01_0728900.1-p1"] <- "MSP1"
```

```
umap_p1 <- ggplot() +
  geom_point(data = umap_df_, aes(x = X1, y = X2), color = "grey90", alpha = 0.5) +
  geom_point(data = umap_df[umap_df$label == 0, ], aes(x = X1, y = X2, fill = group), shape = 21, alpha
  ↪ = 0.5) +
  scale_fill_manual(
    breaks = c("Group 1", "Group 2"), values = c("#A552FD", "#FD529B"),
    labels = c("Predicted antigen (Group 1)", "Predicted antigen (Group 2)")
  ) +
  geom_point(
    data = umap_df[umap_df$label == 1, ], aes(x = X1, y = X2), shape = 21, color = "black",
    fill = "#FFFF33", stroke = 0.3, size = 2, alpha = 0.5
  ) +
```

```

geom_text_repel(
  data = umap_df_text, aes(x = X1, y = X2, label = label_text), size = 3.5,
  nudge_x = 1.5, nudge_y = -1, point.padding = 0.1
) +
theme_bw() +
theme(
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  plot.margin = ggplot2::margin(5, 5, 5, 5, "pt"),
  axis.title = element_text(colour = "black"),
  axis.text = element_text(colour = "black"),
  legend.position = "bottom"
) +
guides(fill = guide_legend(title = "")) +
xlab("Dimension 1") +
ylab("Dimension 2")

```

```

load(file = "./rdata/pred_pos_clusters.RData")
pred_pos_hc_list <- get_subdendrograms(as.dendrogram(pred_pos_hc), 2)
pred_pos_hc_list_2 <- get_subdendrograms(as.dendrogram(pred_pos_hc_list[[1]]), 2)
clusters <- c(
  rep(1, nleaves(pred_pos_hc_list_2[[1]])),
  rep(2, nleaves(pred_pos_hc_list_2[[2]]))
)
names(clusters) <- c(labels(pred_pos_hc_list_2[[1]]), labels(pred_pos_hc_list_2[[2]]))
save(pred_pos_hc, pred_pos_hc_list, pred_pos_hc_list_2,
  file = "./rdata/pred_pos_clusters.RData"
)

```

```

load(file = "./rdata/pred_pos_clusters.RData")
clusters <- c(
  rep(1, nleaves(pred_pos_hc_list_2[[1]])),
  rep(2, nleaves(pred_pos_hc_list_2[[2]]))
)
names(clusters) <- c(labels(pred_pos_hc_list_2[[1]]), labels(pred_pos_hc_list_2[[2]]))
calculate_association(clusters)

```

```

## $M
##      species

```

```
## clusters   pf   pv
##           1 147 519
##           2 1548 1399
##
## $xsq_pval
## [1] "6.5 * 10^-46"
##
## $cramerV
## [1] "0.24" "0.21" "0.27"
```

```
data <- read.csv("./data/supplementary_data_5_pfpv_purf_oob_predictions.csv", row.names = 1,
  ↪ check.names = FALSE)
load("./rdata/pfpv_prox_mds_umap.RData")
umap_df$label <- data$antigen_label
umap_df_ <- umap_df[!rownames(umap_df) %in% names(clusters), ]
umap_df <- rbind(umap_df[umap_df$label == 1, ], umap_df[rownames(umap_df) %in% names(clusters), ])
umap_df$group <- ""
umap_df$group[umap_df$label == 1] <- "Known antigen"
umap_df$group[rownames(umap_df) %in% c("PF3D7_0304600.1-p1", "PF3D7_0424100.1-p1",
  ↪ "PF3D7_0206900.1-p1", "PF3D7_0209000.1-p1", "PVP01_0835600.1-p1", "PVP01_0623800.1-p1",
  ↪ "PVP01_0728900.1-p1")] <- "Reference antigen"
umap_df$group[rownames(umap_df) %in% names(clusters)[clusters == 1]] <- "Group 1"
umap_df$group[rownames(umap_df) %in% names(clusters)[clusters == 2]] <- "Group 2"
umap_df_text <- umap_df[umap_df$label == 1, ]
umap_df_text$label_text <- ""
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0304600.1-p1"] <- "PfCSP"
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0424100.1-p1"] <- "RH5"
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0206900.1-p1"] <- "MSP5"
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0209000.1-p1"] <- "P230"
umap_df_text$label_text[rownames(umap_df_text) == "PVP01_0835600.1-p1"] <- "PvCSP"
umap_df_text$label_text[rownames(umap_df_text) == "PVP01_0623800.1-p1"] <- "DBP"
umap_df_text$label_text[rownames(umap_df_text) == "PVP01_0728900.1-p1"] <- "MSP1"

umap_p2 <- ggplot() +
  geom_point(data = umap_df_, aes(x = X1, y = X2), color = "grey90", alpha = 0.5) +
  geom_point(data = umap_df[umap_df$label == 0, ], aes(x = X1, y = X2, fill = group), shape = 21, alpha
  ↪ = 0.5) +
  scale_fill_manual(
    breaks = c("Group 1", "Group 2"), values = c("#A552FD", "#53F4EF"),
    labels = c("Predicted antigen (Group 1)", "Predicted antigen (Group 2)")
  ) +
```



```

geom_point(
  data = umap_df[umap_df$label == 1, ], aes(x = X1, y = X2), shape = 21, color = "black",
  fill = "#FFFF33", stroke = 0.3, size = 2, alpha = 0.5
) +
geom_text_repel(
  data = umap_df_text, aes(x = X1, y = X2, label = label_text), size = 3.5,
  nudge_x = 1.5, nudge_y = -1, point.padding = 0.1
) +
theme_bw() +
theme(
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  plot.margin = ggplot2::margin(5, 5, 5, 5, "pt"),
  axis.title = element_text(colour = "black"),
  axis.text = element_text(colour = "black"),
  legend.position = "bottom"
) +
guides(fill = guide_legend(title = "")) +
xlab("Dimension 1") +
ylab("Dimension 2")

```

```

load(file = "./rdata/pred_pos_clusters.RData")
pred_pos_hc_list_3 <- get_subdendrograms(as.dendrogram(pred_pos_hc_list_2[[2]]), 2)
clusters <- c(
  rep(1, nleaves(pred_pos_hc_list_3[[1]])),
  rep(2, nleaves(pred_pos_hc_list_3[[2]]))
)
names(clusters) <- c(labels(pred_pos_hc_list_3[[1]]), labels(pred_pos_hc_list_3[[2]]))
save(pred_pos_hc, pred_pos_hc_list, pred_pos_hc_list_2, pred_pos_hc_list_3,
  file = "./rdata/pred_pos_clusters.RData"
)

```

```

load(file = "./rdata/pred_pos_clusters.RData")
clusters <- c(
  rep(1, nleaves(pred_pos_hc_list_3[[1]])),
  rep(2, nleaves(pred_pos_hc_list_3[[2]]))
)
names(clusters) <- c(labels(pred_pos_hc_list_3[[1]]), labels(pred_pos_hc_list_3[[2]]))

```

```
calculate_association(clusters)
```

```
## $M
##      species
## clusters  pf   pv
##      1 1505   40
##      2   43 1359
##
## $xsq_pval
## [1] "0"
##
## $cramerV
## [1] "0.94" "0.93" "0.95"
```

```
data <- read.csv("./data/supplementary_data_5_pfpv_purf_oob_predictions.csv", row.names = 1,
  ↪ check.names = FALSE)
load("./rdata/pfpv_prox_mds_umap.RData")
umap_df$label <- data$antigen_label
umap_df_ <- umap_df[!rownames(umap_df) %in% names(clusters), ]
umap_df <- rbind(umap_df[umap_df$label == 1, ], umap_df[rownames(umap_df) %in% names(clusters), ])
umap_df$group <- ""
umap_df$group[umap_df$label == 1] <- "Known antigen"
umap_df$group[rownames(umap_df) %in% c("PF3D7_0304600.1-p1", "PF3D7_0424100.1-p1",
  ↪ "PF3D7_0206900.1-p1", "PF3D7_0209000.1-p1", "PVP01_0835600.1-p1", "PVP01_0623800.1-p1",
  ↪ "PVP01_0728900.1-p1")] <- "Reference antigen"
umap_df$group[rownames(umap_df) %in% names(clusters)[clusters == 1]] <- "Group 1"
umap_df$group[rownames(umap_df) %in% names(clusters)[clusters == 2]] <- "Group 2"
umap_df_text <- umap_df[umap_df$label == 1, ]
umap_df_text$label_text <- ""
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0304600.1-p1"] <- "PfCSP"
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0424100.1-p1"] <- "RH5"
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0206900.1-p1"] <- "MSP5"
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0209000.1-p1"] <- "P230"
umap_df_text$label_text[rownames(umap_df_text) == "PVP01_0835600.1-p1"] <- "PvCSP"
umap_df_text$label_text[rownames(umap_df_text) == "PVP01_0623800.1-p1"] <- "DBP"
umap_df_text$label_text[rownames(umap_df_text) == "PVP01_0728900.1-p1"] <- "MSP1"

umap_p3 <- ggplot() +
  geom_point(data = umap_df_, aes(x = X1, y = X2), color = "grey90", alpha = 0.5) +
  geom_point(data = umap_df[umap_df$label == 0, ], aes(x = X1, y = X2, fill = group), shape = 21, alpha
  ↪ = 0.5) +
```

```

scale_fill_manual(
  breaks = c("Group 1", "Group 2"), values = c("#F49553", "#53F4EF"),
  labels = c("Predicted antigen (Group 1)", "Predicted antigen (Group 2)")
) +
geom_point(
  data = umap_df[umap_df$label == 1, ], aes(x = X1, y = X2), shape = 21, color = "black",
  fill = "#FFFF33", stroke = 0.3, size = 2, alpha = 0.5
) +
geom_text_repel(
  data = umap_df_text, aes(x = X1, y = X2, label = label_text), size = 3.5,
  nudge_x = 1.5, nudge_y = -1, point.padding = 0.1
) +
theme_bw() +
theme(
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  plot.margin = ggplot2::margin(5, 5, 5, 5, "pt"),
  axis.title = element_text(colour = "black"),
  axis.text = element_text(colour = "black"),
  legend.position = "bottom"
) +
guides(fill = guide_legend(title = "")) +
xlab("Dimension 1") +
ylab("Dimension 2")

```

```

load(file = "./rdata/pred_pos_clusters.RData")
pred_pos_hc_list_4 <- get_subdendrograms(as.dendrogram(pred_pos_hc_list_3[[1]]), 2)
clusters <- c(
  rep(1, nleaves(pred_pos_hc_list_4[[1]])),
  rep(2, nleaves(pred_pos_hc_list_4[[2]]))
)
names(clusters) <- c(labels(pred_pos_hc_list_4[[1]]), labels(pred_pos_hc_list_4[[2]]))
save(pred_pos_hc, pred_pos_hc_list, pred_pos_hc_list_2, pred_pos_hc_list_3, pred_pos_hc_list_4,
  file = "./rdata/pred_pos_clusters.RData"
)

```

```

load(file = "./rdata/pred_pos_clusters.RData")
clusters <- c(

```

```

rep(1, nleaves(pred_pos_hc_list_4[[1]])),
rep(2, nleaves(pred_pos_hc_list_4[[2]]))
)
names(clusters) <- c(labels(pred_pos_hc_list_4[[1]]), labels(pred_pos_hc_list_4[[2]]))
calculate_association(clusters)

```

```

## $M
##      species
## clusters pf  pv
##      1 563 18
##      2 942 22
##
## $xsq_pval
## [1] "3.28 * 10^-01"
##
## $cramerV
## [1] "0.02" "0.00" "0.08"

```

```

data <- read.csv("./data/supplementary_data_5_pfpv_purf_oob_predictions.csv", row.names = 1,
  ↪ check.names = FALSE)
load("./rdata/pfpv_prox_mds_umap.RData")
umap_df$label <- data$antigen_label
umap_df_ <- umap_df[!rownames(umap_df) %in% names(clusters), ]
umap_df <- rbind(umap_df[umap_df$label == 1, ], umap_df[rownames(umap_df) %in% names(clusters), ])
umap_df$group <- ""
umap_df$group[umap_df$label == 1] <- "Known antigen"
umap_df$group[rownames(umap_df) %in% c("PF3D7_0304600.1-p1", "PF3D7_0424100.1-p1",
  ↪ "PF3D7_0206900.1-p1", "PF3D7_0209000.1-p1", "PVP01_0835600.1-p1", "PVP01_0623800.1-p1",
  ↪ "PVP01_0728900.1-p1")] <- "Reference antigen"
umap_df$group[rownames(umap_df) %in% names(clusters)[clusters == 1]] <- "Group 1"
umap_df$group[rownames(umap_df) %in% names(clusters)[clusters == 2]] <- "Group 2"
umap_df_text <- umap_df[umap_df$label == 1, ]
umap_df_text$label_text <- ""
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0304600.1-p1"] <- "PfCSP"
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0424100.1-p1"] <- "RH5"
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0206900.1-p1"] <- "MSP5"
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0209000.1-p1"] <- "P230"
umap_df_text$label_text[rownames(umap_df_text) == "PVP01_0835600.1-p1"] <- "PvCSP"
umap_df_text$label_text[rownames(umap_df_text) == "PVP01_0623800.1-p1"] <- "DBP"
umap_df_text$label_text[rownames(umap_df_text) == "PVP01_0728900.1-p1"] <- "MSP1"

```

```

umap_p4 <- ggplot() +
  geom_point(data = umap_df_, aes(x = X1, y = X2), color = "grey90", alpha = 0.5) +
  geom_point(data = umap_df[umap_df$label == 0, ], aes(x = X1, y = X2, fill = group), shape = 21, alpha
    ↪ = 0.5) +
  scale_fill_manual(
    breaks = c("Group 1", "Group 2"), values = c("#F49553", "#D953F4"),
    labels = c("Predicted antigen (Group 1)", "Predicted antigen (Group 2)")
  ) +
  geom_point(
    data = umap_df[umap_df$label == 1, ], aes(x = X1, y = X2), shape = 21, color = "black",
    fill = "#FFFF33", stroke = 0.3, size = 2, alpha = 0.5
  ) +
  geom_text_repel(
    data = umap_df_text, aes(x = X1, y = X2, label = label_text), size = 3.5,
    nudge_x = 1.5, nudge_y = -1, point.padding = 0.1
  ) +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    plot.margin = ggplot2::margin(5, 5, 5, 5, "pt"),
    axis.title = element_text(colour = "black"),
    axis.text = element_text(colour = "black"),
    legend.position = "bottom"
  ) +
  guides(fill = guide_legend(title = "")) +
  xlab("Dimension 1") +
  ylab("Dimension 2")

```

```

umap_p_combined <- plot_grid(umap_p1, umap_p2, umap_p3, umap_p4,
  nrow = 2,
  labels = c("a", "b", "c", "d")
)
umap_p_combined

```

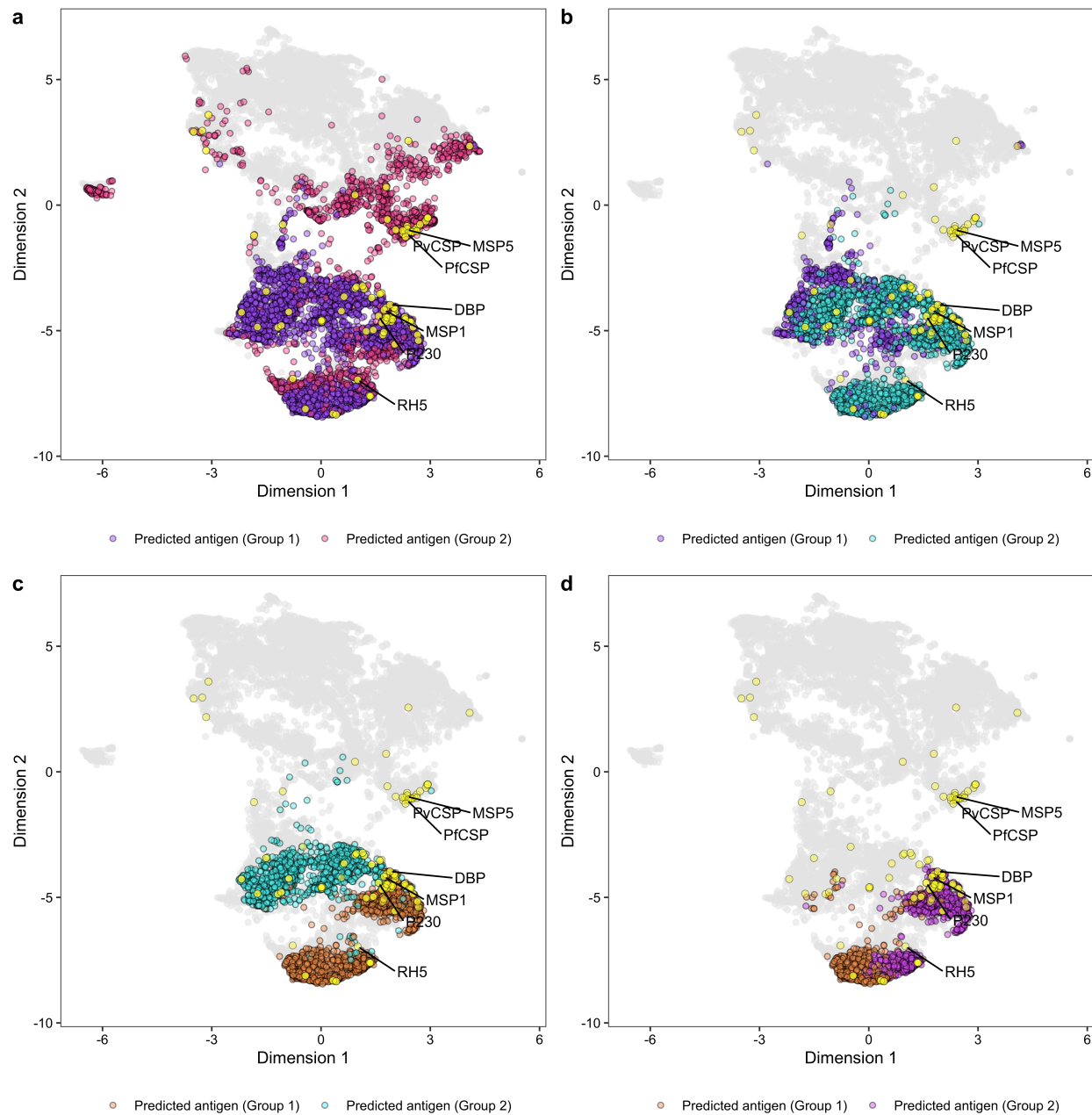
```

png(file = "./figures/Supplementary Fig 8.png", width = 6000, height = 6200, res = 600)
print(umap_p_combined)

```

```
dev.off()

pdf(file = "../supplementary_figures/Supplementary Fig 8.pdf", width = 12, height = 12.4)
print(umap_p_combined)
dev.off()
```



5.3 Comparison of amino acid frequencies between species

5.3.1 Analysis

Python:

```
library(reticulate)
use_condaenv("/Users/renee/Library/r-miniconda/envs/purf/bin/python")
```

```
from Bio import SeqIO
import pandas as pd
```

```
ds = pd.read_csv('./data/supplementary_data_5_pfpv_purf_oob_predictions.csv', index_col=0)
pv_records = list(SeqIO.parse('./other_data/combined_PlasmoDB-
    ↪ 45_PvivaxP01_AnnotatedProteins_no_pseudo_genes_special_sequences_modified.fasta',
    ↪ 'fasta'))
pf_records = list(SeqIO.parse('./other_data/combined_PlasmoDB-
    ↪ 43_Pfalciparum3D7_AnnotatedProteins_no_pseudo_genes_special_sequences_modified.fasta',
    ↪ 'fasta'))

pv_pos = ds[(ds['00B score filtered'] >= 0.5) & (ds['species'] == 'pv')].index
pf_pos = ds[(ds['00B score filtered'] >= 0.5) & (ds['species'] == 'pf')].index
pv_neg = ds[(ds['00B score filtered'] < 0.5) & (ds['species'] == 'pv')].index
pf_neg = ds[(ds['00B score filtered'] < 0.5) & (ds['species'] == 'pf')].index

pv_pos_records = [record for record in pv_records if record.id in pv_pos]
pf_pos_records = [record for record in pf_records if record.id in pf_pos]
pv_neg_records = [record for record in pv_records if record.id in pv_neg]
pf_neg_records = [record for record in pf_records if record.id in pf_neg]

aa_freq = {}
for i in 'ACDEFGHIKLMNPQRSTVWY':
    aa_freq[i] = [0, 0, 0, 0, 0, 0]

for record in pv_records:
    for char in record.seq:
        aa_freq[char][0] += 1

for record in pf_records:
    for char in record.seq:
        aa_freq[char][1] += 1

for record in pv_pos_records:
    for char in record.seq:
```



```

aa_freq[char][2] += 1

for record in pf_pos_records:
    for char in record.seq:
        aa_freq[char][3] += 1

for record in pv_neg_records:
    for char in record.seq:
        aa_freq[char][4] += 1

for record in pf_neg_records:
    for char in record.seq:
        aa_freq[char][5] += 1

res = pd.DataFrame.from_dict(aa_freq)
res.index = ['pv', 'pf', 'pv_pos', 'pf_pos', 'pv_neg', 'pf_neg']
res.to_csv('./other_data/pfpv_aa_freq.csv')

```

In R:

```

library(rcompanion)
library(rlist)
library(DT)

```

```

scientific <- function(x) {
  ifelse(x == 0, "0", gsub("e", " * 10^", scientific_format(digits = 3)(x)))
}

ds <- read.csv("./other_data/pfpv_aa_freq.csv", row.names = 1)
chisq_pval <- c()
cramer_res <- list()
# Comparison between Pv and Pf
M <- as.table(as.matrix(ds[1:2, ]))
Xsq <- chisq.test(M, correct = FALSE)
cramerV <- cramerV(M, ci = TRUE, R = 100)
chisq_pval <- c(chisq_pval, Xsq$p.value)
cramer_res <- list.append(cramer_res, sprintf("%.2f", cramerV))
# Comparison between Pv and Pf positives
M <- as.table(as.matrix(ds[3:4, ]))

```

```

Xsq <- chisq.test(M, correct = FALSE)
cramerV <- cramerV(M, ci = TRUE, R = 100)
chisq_pval <- c(chisq_pval, Xsq$p.value)
cramer_res <- list.append(cramer_res, sprintf("%0.2f", cramerV))
# Comparison between Pv and Pf negatives
M <- as.table(as.matrix(ds[5:6, ]))
Xsq <- chisq.test(M, correct = FALSE)
cramerV <- cramerV(M, ci = TRUE, R = 100)
chisq_pval <- c(chisq_pval, Xsq$p.value)
cramer_res <- list.append(cramer_res, sprintf("%0.2f", cramerV))
# Save results
df <- as.data.frame(cbind(
  c("Pv vs. Pf", "Pv pos vs. Pf pos", "Pv neg vs. Pf neg"),
  chisq_pval, do.call(rbind, cramer_res)
))
colnames(df) <- c("PURF model", "Chi-squared test p-value", "Cramer's V", "Lower CI", "Upper CI")
df$`Chi-squared test p-value` <- sapply(as.numeric(df$`Chi-squared test p-value`), scientific)
save(df, file = "./rdata/pfpv_aa_freq_comparisons.RData")

```

```

load("./rdata/pfpv_aa_freq_comparisons.RData")
df %>%
  datatable(rownames = FALSE)

```

PURF model Chi-squared test p-value Cramer's V

5.4 Variable importance

5.4.1 Analysis

In Python:

```

library(reticulate)
use_condaenv("/Users/renee/Library/r-miniconda/envs/purf/bin/python")

```

```

import pickle
import pandas as pd
import numpy as np

```

```

from sklearn.utils import shuffle
import pickle
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler
from scipy.spatial import distance
import multiprocessing
from joblib import Parallel, delayed
num_cores = multiprocessing.cpu_count()
from sklearn.ensemble._forest import _generate_unsampled_indices
import session_info

```

```

def calculate_raw_var_imp(idx, tree, X, y, weight, groups=None):
    rng = np.random.RandomState(idx)
    oob_indices = _generate_unsampled_indices(tree.random_state, y.shape[0], y.shape[0])
    oob_pos = np.intersect1d(oob_indices, np.where(y == 1)[0])
    noutall = len(oob_pos)
    pred = tree.predict_proba(X.iloc[oob_pos,:])[:, 1]
    nrightall = sum(pred == y[oob_pos])
    imprt, impsd = [], []
    if groups is None:
        for var in range(X.shape[1]):
            X_temp = X.copy()
            X_temp.iloc[:, var] = rng.permutation(X_temp.iloc[:, var])
            pred = tree.predict_proba(X_temp.iloc[oob_pos,:])[:, 1]
            nrightimpall = sum(pred == y[oob_pos])
            delta = (nrightall - nrightimpall) / noutall * weight
            imprt.append(delta)
            impsd.append(delta * delta)
    else:
        for grp in np.unique(groups):
            X_temp = X.copy()
            X_temp.iloc[:, groups == grp] = rng.permutation(X_temp.iloc[:, groups == grp])
            pred = tree.predict_proba(X_temp.iloc[oob_pos,:])[:, 1]
            nrightimpall = sum(pred == y[oob_pos])
            delta = (nrightall - nrightimpall) / noutall * weight
            imprt.append(delta)
            impsd.append(delta * delta)
    return (imprt, impsd)

```

```

def calculate_var_imp(model, features, outcome, num_cores, weights=None, groups=None):
    trees = model.estimators_
    idx_list = [i for i in range(len(trees))]
    if weights is None:
        weights = np.ones(len(trees))
    res = Parallel(n_jobs=num_cores)(
        delayed(calculate_raw_var_imp_)(idx, trees[idx], features, outcome, weights[idx], groups) for
    ↪ idx in idx_list)
    imprt, impsd = [], []
    for i in range(len(idx_list)):
        imprt.append(res[i][0])
        impsd.append(res[i][1])
    imprt = np.array(imprt).sum(axis=0)
    impsd = np.array(impsd).sum(axis=0)
    imprt /= sum(weights)
    impsd = np.sqrt(((impsd / sum(weights)) - imprt * imprt) / sum(weights))
    mda = []
    for i in range(len(imprt)):
        if impsd[i] != 0:
            mda.append(imprt[i] / impsd[i])
        else:
            mda.append(imprt[i])
    if groups is None:
        var_imp = pd.DataFrame({'variable': features.columns, 'meanDecreaseAccuracy': mda})
    else:
        var_imp = pd.DataFrame({'variable': np.unique(groups), 'meanDecreaseAccuracy': mda})
    return var_imp

```

5.4.1.1 Pv data set

In Python:

```

data = pd.read_csv('./other_data/pv_ml_input.csv', index_col=0)
features = data.iloc[:, 1:]
outcome = np.array(data.antigen_label)

features, outcome = shuffle(features, outcome, random_state=0)

# Imputation
imputer = SimpleImputer(strategy='median')

```

```

X = imputer.fit_transform(features)
X = pd.DataFrame(X, index=features.index, columns=features.columns)
y = outcome
features = X
print('There are %d positives out of %d samples before feature space weighting.' % (sum(y), len(y)))

# Feature space weighting
lab_pos = X.loc[y==1,:]
median = np.median(lab_pos, axis=0)

# Feature space weighting
lab_pos = X.loc[y==1,:]
median = np.median(lab_pos, axis=0)
scaler = MinMaxScaler(feature_range=(1,10))
dist = list()
for i in range(lab_pos.shape[0]):
    dist.append(distance.euclidean(lab_pos.iloc[i, :], median))

dist = np.asarray(dist).reshape(-1, 1)
counts = np.round(scaler.fit_transform(dist))
counts = np.array(counts, dtype=np.int64)[: , 0]
X_temp = X.iloc[y==1, :]
X = X.iloc[y==0, :]
y = np.asarray([0] * X.shape[0] + [1] * (sum(counts)))
appended_data = [X]
for i in range(len(counts)):
    appended_data.append(pd.concat([X_temp.iloc[[i]] * counts[i]))

X = pd.concat(appended_data)
print('There are %d positives out of %d samples after feature space weighting.' % (sum(y), len(y)))
features = X
outcome = y
X.to_csv('./other_data/pv_ml_input_processed_weighted.csv')

purf_model = pickle.load(open('./pickle_data/pv_0.5_purf_tree_filtering.pkl', 'rb'))
purf = purf_model['model']
weights = purf_model['weights']

metadata = pd.read_csv('./data/supplementary_data_1_protein_variable_metadata.csv')
groups = metadata.loc[np.isin(metadata['column name'], features.columns), 'category'].array

```

```

var_imp = calculate_var_imp(purf, features, outcome, 8, weights)
grp_var_imp = calculate_var_imp(purf, features, outcome, 8, weights, groups)
var_imp.to_csv('./other_data/pv_known_antigen_variable_importance.csv', index=False)
grp_var_imp.to_csv('./other_data/pv_known_antigen_group_variable_importance.csv', index=False)

```

In R:

```

prediction <- read.csv("./data/supplementary_data_3_pv_purf_oob_predictions.csv", check.names = FALSE)
known_antigens <- prediction[prediction$antigen_label == 1, ]$protein_id
other_proteins <- prediction[prediction$antigen_label == 0 & prediction$`00B score filtered` < 0.5,
↪ ]$protein_id

set.seed(22)
random_proteins <- sample(other_proteins, size = length(known_antigens), replace = FALSE)

# Load imputed data
data <- read.csv("./other_data/pv_ml_input_processed_weighted.csv")
data <- data[!duplicated(data), ]
compared_group <- sapply(data$protein_id, function(x) if (x %in% known_antigens) 1 else if (x %in%
↪ random_proteins) 0 else -1)
data <- data[, 2:ncol(data)]

# Min-max normalization
min_max <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}
data <- data.frame(lapply(data, min_max))

save(compared_group, data, file = "./rdata/pv_known_antigen_wilcox_data.RData")

load(file = "./rdata/pv_known_antigen_wilcox_data.RData")
pval <- c()
for (i in 1:ncol(data)) {
  pval <- c(pval, wilcox.test(data[compared_group == 1, i], data[compared_group == 0, i])$p.value)
}
adj_pval <- p.adjust(pval, method = "BH", n = length(pval))
wilcox_res <- data.frame(variable = colnames(data), adj_pval = adj_pval)
write.csv(wilcox_res, "./other_data/pv_known_antigen_wilcox_res.csv", row.names = FALSE)

```

5.4.1.2 Pv + Pf combined data set

In Python:

```
data = pd.read_csv('./data/supplementary_data_4_pfpv_ml_input.csv', index_col=0)
features = data.iloc[:, 1:]
outcome = np.array(data.antigen_label)

features, outcome = shuffle(features, outcome, random_state=0)

# Imputation
imputer = SimpleImputer(strategy='median')
X = imputer.fit_transform(features)
X = pd.DataFrame(X, index=features.index, columns=features.columns)
y = outcome
features = X
print('There are %d positives out of %d samples before feature space weighting.' % (sum(y), len(y)))

# Feature space weighting
lab_pos = X.loc[y==1,:]
median = np.median(lab_pos, axis=0)

# Feature space weighting
lab_pos = X.loc[y==1,:]
median = np.median(lab_pos, axis=0)
scaler = MinMaxScaler(feature_range=(1,10))
dist = list()
for i in range(lab_pos.shape[0]):
    dist.append(distance.euclidean(lab_pos.iloc[i, :], median))

dist = np.asarray(dist).reshape(-1, 1)
counts = np.round(scaler.fit_transform(dist))
counts = np.array(counts, dtype=np.int64)[: , 0]
X_temp = X.iloc[y==1, :]
X = X.iloc[y==0, :]
y = np.asarray([0] * X.shape[0] + [1] * (sum(counts)))
appended_data = [X]
for i in range(len(counts)):
    appended_data.append(pd.concat([X_temp.iloc[[i]] * counts[i]]))

X = pd.concat(appended_data)
print('There are %d positives out of %d samples after feature space weighting.' % (sum(y), len(y)))
```

```

features = X
outcome = y
X.to_csv('./other_data/pfpv_ml_input_processed_weighted.csv')

purf_model = pickle.load(open('./pickle_data/pfpv_0.5_purf_tree_filtering.pkl', 'rb'))
purf = purf_model['model']
weights = purf_model['weights']

metadata = pd.read_csv('./data/supplementary_data_1_protein_variable_metadata.csv')
groups = metadata.loc[np.isin(metadata['column name'], features.columns), 'category'].array

var_imp = calculate_var_imp(purf, features, outcome, 8, weights)
grp_var_imp = calculate_var_imp(purf, features, outcome, 8, weights, groups)
var_imp.to_csv('./other_data/pfpv_known_antigen_variable_importance.csv', index=False)
grp_var_imp.to_csv('./other_data/pfpv_known_antigen_group_variable_importance.csv', index=False)

```

In R:

```

prediction <- read.csv("./data/supplementary_data_5_pfpv_purf_oob_predictions.csv", check.names =
  ↪ FALSE)
known_antigens <- prediction[prediction$antigen_label == 1, ]$protein_id
other_proteins <- prediction[prediction$antigen_label == 0 & prediction$`OOB score filtered` < 0.5,
  ↪ ]$protein_id

set.seed(22)
random_proteins <- sample(other_proteins, size = length(known_antigens), replace = FALSE)

# Load imputed data
data <- read.csv("./other_data/pfpv_ml_input_processed_weighted.csv")
data <- data[!duplicated(data), ]
compared_group <- sapply(data$protein_id, function(x) if (x %in% known_antigens) 1 else if (x %in%
  ↪ random_proteins) 0 else -1)
data <- data[, 2:ncol(data)]

# Min-max normalization
min_max <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}
data <- data.frame(lapply(data, min_max))

```



```

save(compared_group, data, file = "./rdata/pfpv_known_antigen_wilcox_data.RData")

load(file = "./rdata/pfpv_known_antigen_wilcox_data.RData")
pval <- c()
for (i in 1:ncol(data)) {
  pval <- c(pval, wilcox.test(data[compared_group == 1, i], data[compared_group == 0, i])$p.value)
}
adj_pval <- p.adjust(pval, method = "BH", n = length(pval))
wilcox_res <- data.frame(variable = colnames(data), adj_pval = adj_pval)
write.csv(wilcox_res, "./other_data/pfpv_known_antigen_wilcox_res.csv", row.names = FALSE)

```

5.4.2 Plotting

In R:

```

library(ggplot2)
library(reshape2)
library(cowplot)
library(stringr)

colorset <- c("genomic" = "#0C1C63", "immunological" = "#408002", "proteomic" = "#0F80FF",
  ↪ "structural" = "#FEAE34")

```

5.4.2.1 Pv data set

```

# Variable importance
var_imp <- read.csv("./other_data/pv_known_antigen_variable_importance.csv")
var_imp <- var_imp[order(-var_imp$meanDecreaseAccuracy), ]
var_imp <- var_imp[1:10, ]

metadata <- read.csv("./data/supplementary_data_1_protein_variable_metadata.csv", check.names = FALSE)
metadata <- metadata[c("category", "column name")]
metadata <- metadata[metadata$`column name` %in% var_imp$variable, ]

var_imp <- merge(x = var_imp, y = metadata, by.x = "variable", by.y = "column name")
var_imp$category <- factor(var_imp$category, levels = names(colorset))
var_imp$color <- sapply(var_imp$category, function(x) colorset[x])

```

```

var_imp$variable_ <- c(
  "GPI-anchor specificity score",
  "Total length of the low\n complexity regions",
  "Maximum score of Chou and\n Fasman beta turn",
  "Maximum score of Kolaskar and\n Tongaonkar antigenicity",
  "Maximum score of B-cell\n epitopes (BepiPred-1.0)",
  "Maximum score of IFN-gamma\n inducing epitopes",
  "Minimum score of B-cell\n epitopes (BepiPred-1.0)",
  "Number of non-synonymous SNPs",
  "Number of IFN-gamma inducing\n epitopes",
  "Secretory signal peptide\n probability"
)

p1 <- ggplot(var_imp, aes(
  x = reorder(variable_, meanDecreaseAccuracy), y = meanDecreaseAccuracy,
  fill = category
)) +
  geom_point(size = 3, pch = 21, color = "black", alpha = 0.8) +
  scale_fill_manual(values = colorset, labels = c("Genomic", "Immunological", "Proteomic",
    ↪ "Structural")) +
  coord_flip() +
  ylim(min(var_imp$meanDecreaseAccuracy), max(var_imp$meanDecreaseAccuracy) + 1) +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.grid.major.y = element_line(color = "grey80", linewidth = 0.3, linetype = "dotted"),
    strip.background = element_blank(),
    panel.border = element_rect(color = "black"),
    legend.text = element_text(color = "black"),
    plot.title = element_text(hjust = 0.5, size = 20),
    plot.margin = ggplot2::margin(10, 10, 0, 10, "pt"),
    axis.title.x = element_text(color = "black"),
    axis.title.y = element_text(color = "black"),
    axis.text.x = element_text(color = "black"),
    axis.text.y = element_text(color = "black"),
    legend.title = element_blank(),
    legend.position = c(0.8, 0.2),
    legend.background = element_rect(colour = "black", linewidth = 0.2)
  ) +

```

```
xlab("") +
ylab("Mean decrease in accuracy")
```

```
# Group variable importance
grp_var_imp <- read.csv("./other_data/pv_known_antigen_group_variable_importance.csv")

grp_var_imp_ <- grp_var_imp
firstup <- function(x) {
  substr(x, 1, 1) <- toupper(substr(x, 1, 1))
  return(x)
}
grp_var_imp_$variable <- sapply(grp_var_imp_$variable, function(x) {
  x <- str_replace_all(x, "[_\\.]", " ")
  x <- firstup(x)
  return(x)
})

grp_var_imp_$category <- factor(tolower(grp_var_imp_$variable))

p2 <- ggplot(grp_var_imp_, aes(x = reorder(variable, meanDecreaseAccuracy), y = meanDecreaseAccuracy,
  ↪ fill = category)) +
  geom_point(size = 3, pch = 21, color = "black", alpha = 0.8) +
  scale_fill_manual(values = colorset, labels = c("Genomic", "Immunological", "Proteomic",
  ↪ "Structural")) +
  coord_flip() +
  ylim(min(grp_var_imp$meanDecreaseAccuracy), max(grp_var_imp$meanDecreaseAccuracy) + 5) +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.grid.major.y = element_line(color = "grey80", linewidth = 0.3, linetype = "dotted"),
    strip.background = element_blank(),
    panel.border = element_rect(color = "black"),
    legend.text = element_text(color = "black", size = 10),
    plot.title = element_text(hjust = 0.5, size = 20),
    plot.margin = ggplot2::margin(30, 10, 10, 90, "pt"),
    axis.title.x = element_text(color = "black"),
    axis.title.y = element_text(color = "black"),
    axis.text.x = element_text(color = "black"),
```

```

axis.text.y = element_text(color = "black"),
legend.position = "none"
) +
xlab("") +
ylab("Mean decrease in accuracy")

```

```

# Wilcoxon test
load(file = "../rdata/pv_known_antigen_wilcox_data.RData")
wilcox_res <- read.csv("../other_data/pv_known_antigen_wilcox_res.csv")
wilcox_data <- data
wilcox_data$compared_group <- compared_group
wilcox_data <- wilcox_data[wilcox_data$compared_group != -1, ]
wilcox_data <- melt(wilcox_data, id = c("compared_group"))
wilcox_data <- merge(x = wilcox_data, y = merge(x = var_imp, y = wilcox_res), by = "variable", all.y =
  ↪ TRUE)
wilcox_data$tile_pos <- rep(0, nrow(wilcox_data))
wilcox_data$compared_group <- factor(wilcox_data$compared_group)

wilcox_data$variable <- sapply(wilcox_data$variable, function(x) {
  x <- str_replace_all(x, "[_\\.]", " ")
  x <- firstup(x)
  return(x)
})

adj_pval_tmp <- c()
for (i in 1:nrow(wilcox_data)) {
  x <- wilcox_data$adj_pval[i]
  a <- strsplit(format(x, scientific = TRUE, digits = 3), "e")[[1]]
  res <- paste0(sprintf("%.2f", as.numeric(a[1])), " %%% 10^", as.integer(a[2]))
  adj_pval_tmp <- c(adj_pval_tmp, res)
}
wilcox_data$adj_pval <- adj_pval_tmp

```

```

p3 <- ggplot(wilcox_data, aes(x = reorder(variable, meanDecreaseAccuracy), y = value, fill =
  ↪ compared_group)) +
  geom_boxplot(outlier.color = NA, alpha = 0.3, lwd = 0.3) +
  geom_point(
    color = "black", shape = 21, stroke = 0.3, alpha = 0.5, size = 0.5,

```

```

    position = position_jitterdodge()
  ) +
  geom_text(aes(label = adj_pval),
    y = 1.1, size = 3, fontface = "plain", family = "sans",
    hjust = 0, parse = TRUE
  ) +
  geom_vline(xintercept = 1:9 + 0.5, color = "grey80", linetype = "solid", linewidth = 0.1) +
  coord_flip(ylim = c(0, 1), clip = "off") +
  scale_fill_manual(
    breaks = c("1", "0"), values = c("#FF007F", "#0080FF"),
    labels = c("Known antigens", "Random predicted non-antigens")
  ) +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    strip.background = element_blank(),
    panel.border = element_rect(linewidth = 0.2, colour = "black"),
    plot.title = element_text(hjust = 0.5),
    plot.margin = ggplot2::margin(10, 90, 0, -20, "pt"),
    legend.text = element_text(colour = "black"),
    axis.title.x = element_text(color = "black"),
    axis.title.y = element_text(color = "black"),
    axis.text.x = element_text(color = "black"),
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank(),
    legend.position = "none"
  ) +
  xlab("") +
  ylab("Normalized variable value")

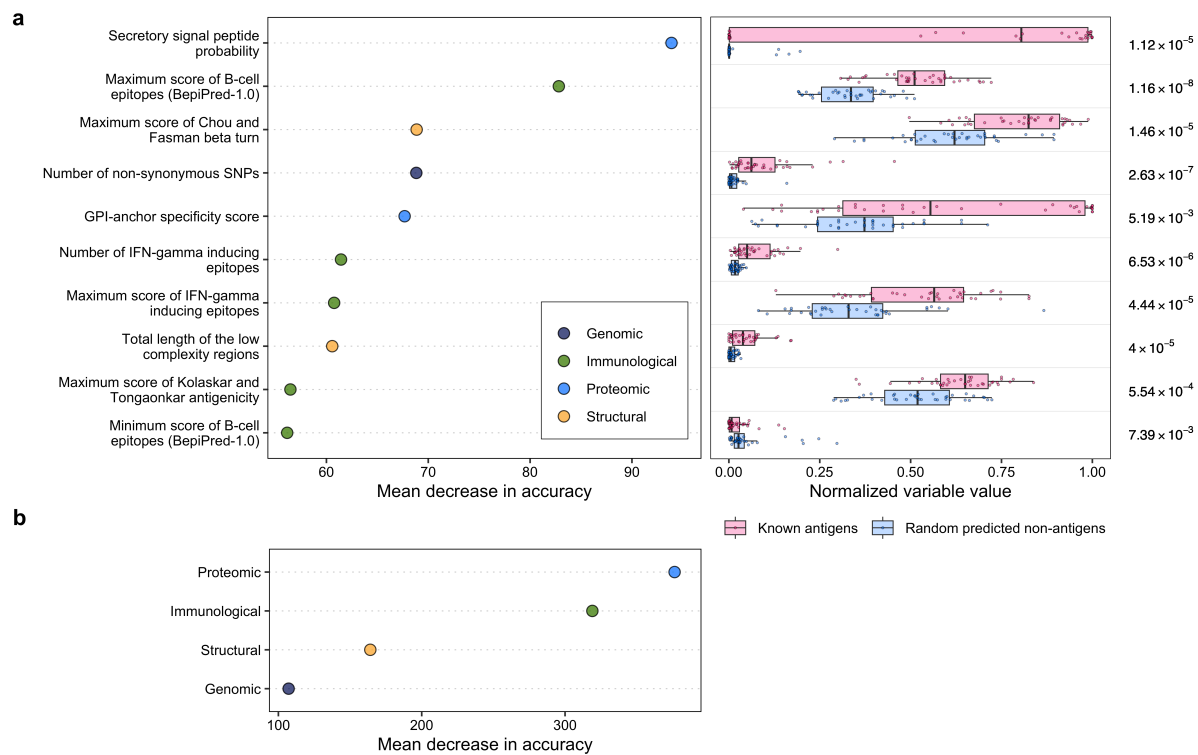
legend <- get_legend(p3 +
  theme(
    legend.title = element_blank(),
    legend.background = element_blank(),
    legend.key = element_blank(),
    legend.direction = "horizontal",
    legend.position = c(0.35, 0.9)
  ))

```

```

p_combined <- plot_grid(plot_grid(p1, p3, labels = c("a", ""), rel_widths = c(0.57, 0.43)),
  plot_grid(p2, NULL, legend,
    labels = c("b", "", "", ""),
    nrow = 1, rel_widths = c(0.57, 0.01, 0.42)
  ),
  ncol = 1, rel_heights = c(0.65, 0.35)
)
p_combined

```



5.4.2.2 Pv + Pf combined data set

```

# Variable importance
var_imp <- read.csv("./other_data/pfpv_known_antigen_variable_importance.csv")
var_imp <- var_imp[order(-var_imp$meanDecreaseAccuracy), ]
var_imp <- var_imp[1:10, ]

metadata <- read.csv("./data/supplementary_data_1_protein_variable_metadata.csv", check.names = FALSE)

```

```

metadata <- metadata[c("category", "column name")]
metadata <- metadata[metadata$`column name` %in% var_imp$variable, ]

var_imp <- merge(x = var_imp, y = metadata, by.x = "variable", by.y = "column name")
var_imp$category <- factor(var_imp$category, levels = names(colorset))
var_imp$color <- sapply(var_imp$category, function(x) colorset[x])

var_imp$variable_ <- c(
  "Percentage of aspartic acid\n minus percentage of glutamic acid",
  "GPI-anchor specificity score",
  "Total length of the low\n complexity regions",
  "Maximum score of Parker\n hydrophilicity",
  "Number of non-synonymous SNPs",
  "Percentage of amino acids with\n normalized van der Waals volume\n between 4.03-8.08",
  "Number of IFN-gamma inducing\n epitopes",
  "Percentage of amino acids with\n polarizability between 0.219-0.409",
  "Small amino acid percentage",
  "Secretory signal peptide\n probability"
)

p1 <- ggplot(var_imp, aes(x = reorder(variable_, meanDecreaseAccuracy), y = meanDecreaseAccuracy, fill
↵ = category)) +
  geom_point(size = 3, pch = 21, color = "black", alpha = 0.8) +
  scale_fill_manual(values = colorset, labels = c("Genomic", "Immunological", "Proteomic",
↵ "Structural")) +
  coord_flip() +
  ylim(min(var_imp$meanDecreaseAccuracy), max(var_imp$meanDecreaseAccuracy) + 1) +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.grid.major.y = element_line(color = "grey80", linewidth = 0.3, linetype = "dotted"),
    strip.background = element_blank(),
    panel.border = element_rect(color = "black"),
    legend.text = element_text(color = "black"),
    plot.title = element_text(hjust = 0.5, size = 20),
    plot.margin = ggplot2::margin(10, 10, 0, 10, "pt"),
    axis.title.x = element_text(color = "black"),
    axis.title.y = element_text(color = "black"),
    axis.text.x = element_text(color = "black"),
    axis.text.y = element_text(color = "black"),
  )

```

```

    legend.title = element_blank(),
    legend.position = c(0.8, 0.2),
    legend.background = element_rect(colour = "black", linewidth = 0.2)
  ) +
  xlab("") +
  ylab("Mean decrease in accuracy")

```

```

# Group variable importance
grp_var_imp <- read.csv("./other_data/pfpv_known_antigen_group_variable_importance.csv")

grp_var_imp_ <- grp_var_imp
firstup <- function(x) {
  substr(x, 1, 1) <- toupper(substr(x, 1, 1))
  return(x)
}
grp_var_imp_$variable <- sapply(grp_var_imp_$variable, function(x) {
  x <- str_replace_all(x, "[_\\.]", " ")
  x <- firstup(x)
  return(x)
})

grp_var_imp_$category <- factor(tolower(grp_var_imp_$variable))

p2 <- ggplot(grp_var_imp_, aes(x = reorder(variable, meanDecreaseAccuracy), y = meanDecreaseAccuracy,
  ↪ fill = category)) +
  geom_point(size = 3, pch = 21, color = "black", alpha = 0.8) +
  scale_fill_manual(values = colorset, labels = c("Genomic", "Immunological", "Proteomic",
  ↪ "Structural")) +
  coord_flip() +
  ylim(min(grp_var_imp$meanDecreaseAccuracy), max(grp_var_imp$meanDecreaseAccuracy) + 5) +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.grid.major.y = element_line(color = "grey80", linewidth = 0.3, linetype = "dotted"),
    strip.background = element_blank(),
    panel.border = element_rect(color = "black"),
    legend.text = element_text(color = "black", size = 10),
    plot.title = element_text(hjust = 0.5, size = 20),

```



```

plot.margin = ggplot2::margin(30, 10, 10, 90, "pt"),
axis.title.x = element_text(color = "black"),
axis.title.y = element_text(color = "black"),
axis.text.x = element_text(color = "black"),
axis.text.y = element_text(color = "black"),
legend.position = "none"
) +
xlab("") +
ylab("Mean decrease in accuracy")

```

```

# Wilcoxon test
load(file = "./rdata/pfpv_known_antigen_wilcox_data.RData")
wilcox_res <- read.csv("./other_data/pfpv_known_antigen_wilcox_res.csv")
wilcox_data <- data
wilcox_data$compared_group <- compared_group
wilcox_data <- wilcox_data[wilcox_data$compared_group != -1, ]
wilcox_data <- melt(wilcox_data, id = c("compared_group"))
wilcox_data <- merge(x = wilcox_data, y = merge(x = var_imp, y = wilcox_res), by = "variable", all.y =
  ↪ TRUE)
wilcox_data$tile_pos <- rep(0, nrow(wilcox_data))
wilcox_data$compared_group <- factor(wilcox_data$compared_group)

wilcox_data$variable <- sapply(wilcox_data$variable, function(x) {
  x <- str_replace_all(x, "[_\\.]", " ")
  x <- firstup(x)
  return(x)
})

adj_pval_tmp <- c()
for (i in 1:nrow(wilcox_data)) {
  x <- wilcox_data$adj_pval[i]
  a <- strsplit(format(x, scientific = TRUE, digits = 3), "e")[[1]]
  res <- paste0(sprintf("%.2f", as.numeric(a[1])), " %*% 10^", as.integer(a[2]))
  adj_pval_tmp <- c(adj_pval_tmp, res)
}
wilcox_data$adj_pval <- adj_pval_tmp

```

```

p3 <- ggplot(wilcox_data, aes(x = reorder(variable, meanDecreaseAccuracy), y = value, fill =
  ↪ compared_group)) +

```

```

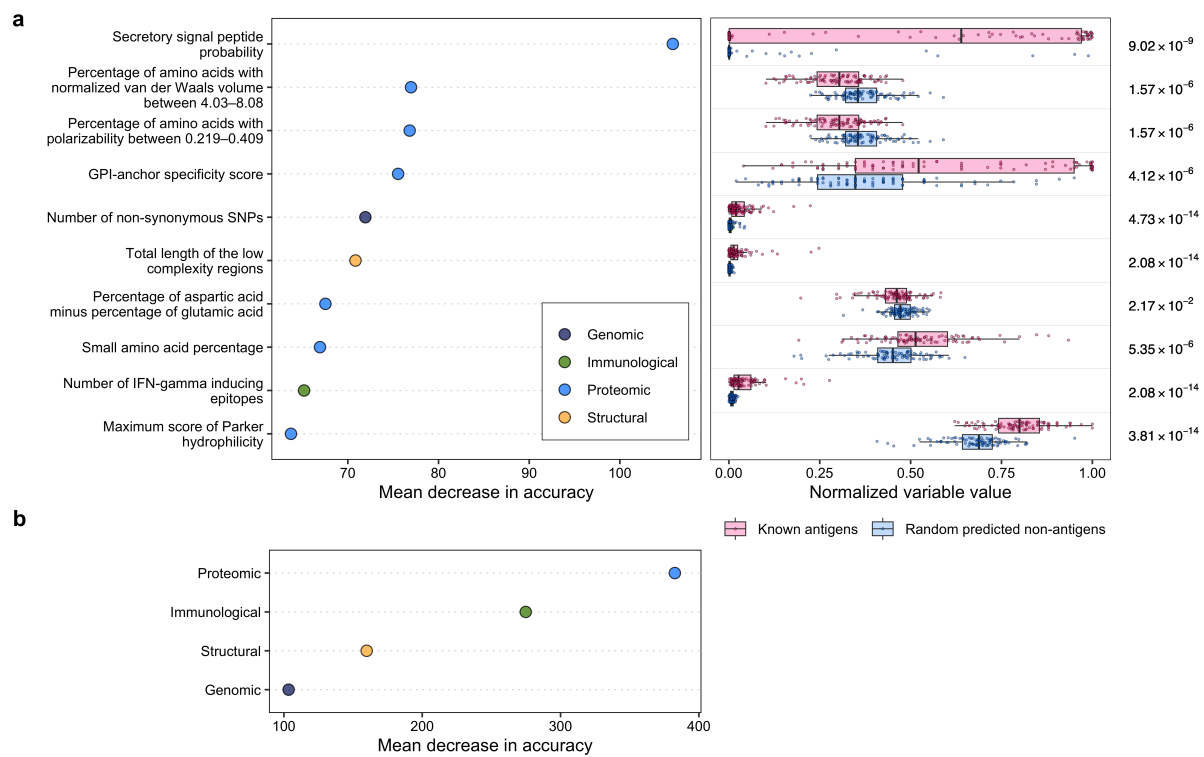
geom_boxplot(outlier.color = NA, alpha = 0.3, lwd = 0.3) +
geom_point(
  color = "black", shape = 21, stroke = 0.3, alpha = 0.5, size = 0.5,
  position = position_jitterdodge()
) +
geom_text(aes(label = adj_pval),
  y = 1.1, size = 3, fontface = "plain", family = "sans",
  hjust = 0, parse = TRUE
) +
geom_vline(xintercept = 1:9 + 0.5, color = "grey80", linetype = "solid", linewidth = 0.1) +
coord_flip(ylim = c(0, 1), clip = "off") +
scale_fill_manual(
  breaks = c("1", "0"), values = c("#FF007F", "#0080FF"),
  labels = c("Known antigens", "Random predicted non-antigens")
) +
theme_bw() +
theme(
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  strip.background = element_blank(),
  panel.border = element_rect(linewidth = 0.2, colour = "black"),
  plot.title = element_text(hjust = 0.5),
  plot.margin = ggplot2::margin(10, 90, 0, -20, "pt"),
  legend.text = element_text(colour = "black"),
  axis.title.x = element_text(color = "black"),
  axis.title.y = element_text(color = "black"),
  axis.text.x = element_text(color = "black"),
  axis.text.y = element_blank(),
  axis.ticks.y = element_blank(),
  legend.position = "none"
) +
xlab("") +
ylab("Normalized variable value")

legend <- get_legend(p3 +
  theme(
    legend.title = element_blank(),
    legend.background = element_blank(),
    legend.key = element_blank(),
    legend.direction = "horizontal",
    legend.position = c(0.35, 0.9)
  )

```

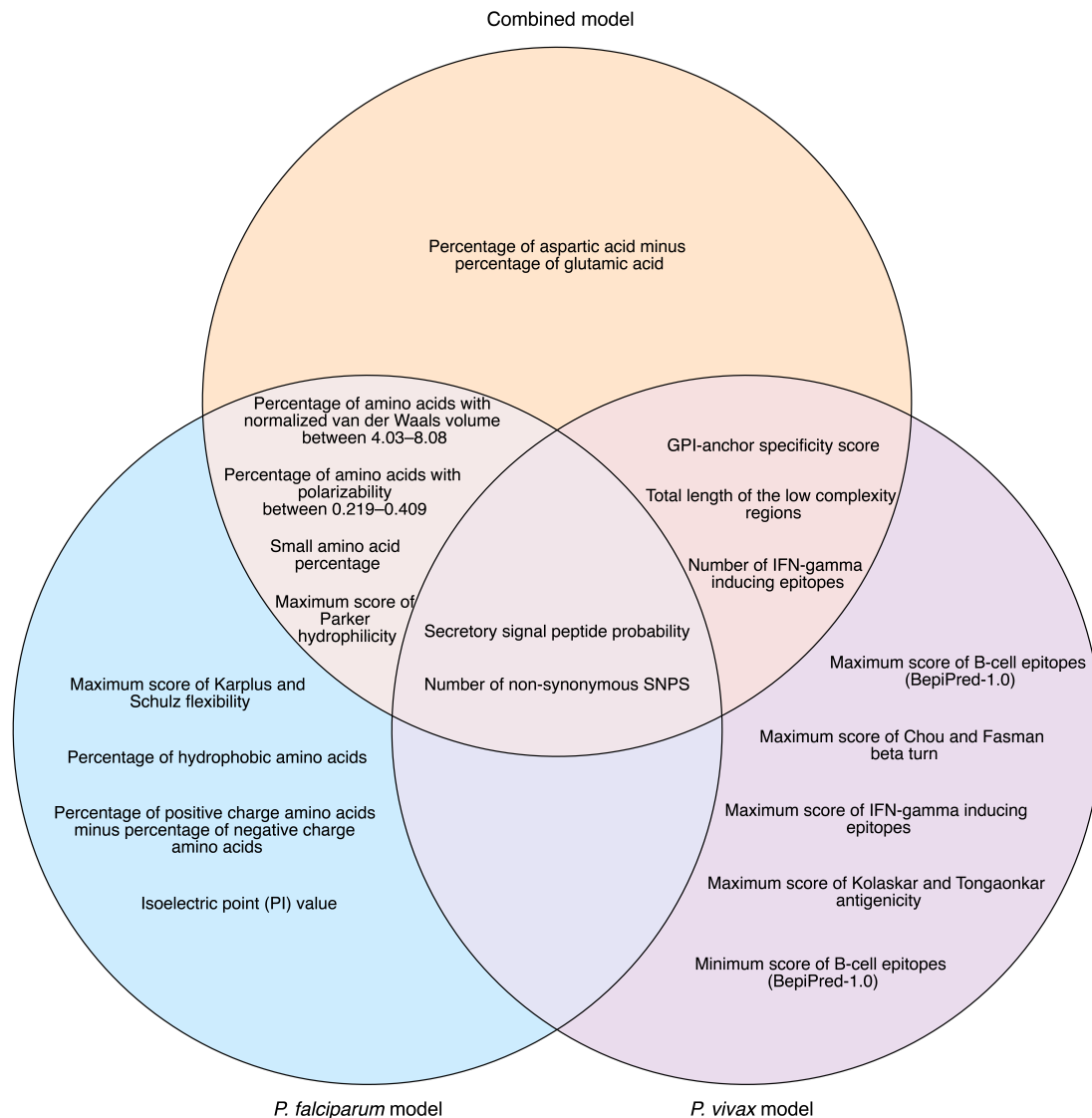
```
))
```

```
p_combined <- plot_grid(plot_grid(p1, p3, labels = c("a", ""), rel_widths = c(0.57, 0.43)),
  plot_grid(p2, NULL, legend,
    labels = c("b", "", "", ""),
    nrow = 1, rel_widths = c(0.57, 0.01, 0.42)
  ),
  ncol = 1, rel_heights = c(0.65, 0.35)
)
p_combined
```



5.4.3 Comparison of top 10 important variables

5.4.3.1 Venn diagram



5.4.3.2 Top 10 variables from combined model

In R

```

library(ggrepel)

data <- read.csv("./other_data/pfpv_top_10_imp_vars.csv")
data$variable <- c(
  "Secretory signal peptide\n probability",
  "Percentage of amino acids with\n normalized van der Waals volume\n between 4.03-8.08",
  "Percentage of amino acids with\n polarizability between 0.219-0.409",
  "GPI-anchor specificity score",
  "Number of non-synonymous SNPs",
  "Total length of the low\n complexity regions",
  "Percentage of aspartic acid\n minus percentage of glutamic acid",
  "Small amino acid percentage",
  "Number of IFN-gamma inducing\n epitopes",
  "Maximum score of Parker\n hydrophilicity"
)

p1 <- ggplot(data, aes(x = pf_single_model, y = pfpv_combined_model, label = variable)) +
  geom_abline(intercept = 0, slope = 1, alpha = 0.3) +
  geom_point() +
  geom_text_repel(
    size = 2.5, point.padding = 0, min.segment.length = 0,
    max.time = 1, max.iter = 1e5, seed = 42, box.padding = 0.3,
    segment.color = "grey30", segment.size = 0.2, lineheight = 1
  ) +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    strip.background = element_blank(),
    panel.border = element_rect(linewidth = 0.2, colour = "black"),
    plot.title = element_text(hjust = 0.5),
    plot.margin = ggplot2::margin(5, 0, 0, 5, "pt"),
    legend.text = element_text(colour = "black"),
    axis.title.x = element_blank(),
    axis.title.y = element_text(color = "black"),
    axis.text.x = element_blank(),
    axis.text.y = element_text(color = "black"),
    axis.ticks.x = element_blank(),
    legend.position = "none"
  )

```

```

) +
ylab("Combined model") +
xlim(0, 110) +
ylim(0, 110)

p2 <- ggplot(data, aes(x = pv_single_model, y = pfpv_combined_model, label = variable)) +
  geom_abline(intercept = 0, slope = 1, alpha = 0.3) +
  geom_point() +
  geom_text_repel(
    size = 2.5, point.padding = 0, min.segment.length = 0,
    max.time = 1, max.iter = 1e5, seed = 42, box.padding = 0.3,
    segment.color = "grey30", segment.size = 0.2, lineheight = 1
  ) +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    strip.background = element_blank(),
    panel.border = element_rect(linewidth = 0.2, colour = "black"),
    plot.title = element_text(hjust = 0.5),
    plot.margin = ggplot2::margin(5, 5, 0, 0, "pt"),
    legend.text = element_text(colour = "black"),
    axis.title.x = element_text(color = "black"),
    axis.title.y = element_blank(),
    axis.text.x = element_text(color = "black"),
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank(),
    legend.position = "none"
  ) +
  xlab(expression(paste(italic("P. vivax"), " model")))) +
  xlim(0, 110) +
  ylim(0, 110)

p3 <- ggplot(data, aes(x = pf_single_model, y = pv_single_model, label = variable)) +
  geom_abline(intercept = 0, slope = 1, alpha = 0.3) +
  geom_point() +
  geom_text_repel(
    size = 2.5, point.padding = 0, min.segment.length = 0,
    max.time = 1, max.iter = 1e5, seed = 42, box.padding = 0.3,
    segment.color = "grey30", segment.size = 0.2, lineheight = 1
  ) +

```

```

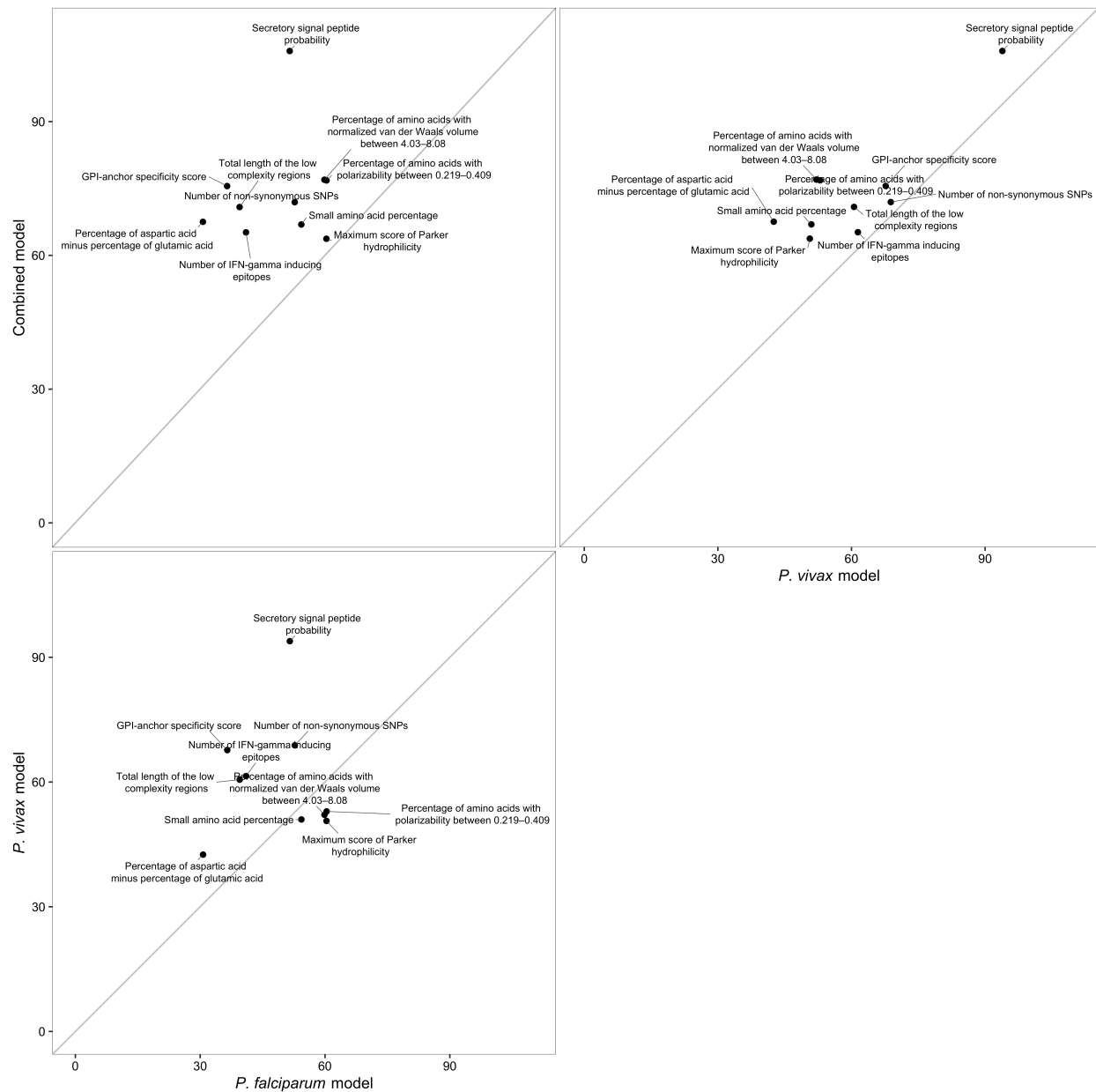
theme_bw() +
theme(
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  strip.background = element_blank(),
  panel.border = element_rect(linewidth = 0.2, colour = "black"),
  plot.title = element_text(hjust = 0.5),
  plot.margin = ggplot2::margin(0, 0, 5, 5, "pt"),
  legend.text = element_text(colour = "black"),
  axis.title.x = element_text(color = "black"),
  axis.title.y = element_text(color = "black"),
  axis.text.x = element_text(color = "black"),
  axis.text.y = element_text(color = "black"),
  legend.position = "none"
) +
xlab(expression(paste(italic("P. falciparum"), " model")))) +
ylab(expression(paste(italic("P. vivax"), " model")))) +
xlim(0, 110) +
ylim(0, 110)

```

```

p_combined <- plot_grid(plot_grid(p1, p3, ncol = 1),
  plot_grid(p2, NULL, ncol = 1, rel_heights = c(0.53, 0.47)),
  ncol = 2
)
p_combined

```



```
sessionInfo()
```

```
## R version 4.2.3 (2023-03-15)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur ... 10.16
##
```



```
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats      graphics  grDevices  utils      datasets  methods   base
##
## other attached packages:
## [1] stringr_1.5.0      reshape2_1.4.4     rlist_0.4.6.2      rcompanion_2.4.30
## [5] scales_1.2.1       dendextend_1.17.2  cluster_2.1.4      NbClust_3.0.1
## [9] factoextra_1.0.7   DT_0.27            cowplot_1.1.1      ggrepel_0.9.3
## [13] ggplot2_3.4.2      umap_0.2.10.0      reticulate_1.28
##
## loaded via a namespace (and not attached):
## [1] matrixStats_0.63.0 webshot_0.5.4      httr_1.4.6         rprojroot_2.0.3
## [5] R.cache_0.16.0     bslib_0.4.2        tools_4.2.3         utf8_1.2.3
## [9] R6_2.5.1           nortest_1.0-4      colorspace_2.1-0    withr_2.5.0
## [13] processx_3.8.1     tidyselect_1.2.0   gridExtra_2.3       Exact_3.2
## [17] compiler_4.2.3     cli_3.6.1          expm_0.999-7        sandwich_3.0-2
## [21] sass_0.4.6         bookdown_0.34      lmtest_0.9-40       mvtnorm_1.1-3
## [25] callr_3.7.3        proxy_0.4-27       askpass_1.1         multcompView_0.1-9
## [29] digest_0.6.31      rmarkdown_2.21     R.utils_2.12.2      pkgconfig_2.0.3
## [33] htmltools_0.5.5    styler_1.9.1       highr_0.10          fastmap_1.1.1
## [37] htmlwidgets_1.6.2  rlang_1.1.1        readxl_1.4.2        rstudioapi_0.14
## [41] jquerylib_0.1.4     generics_0.1.3     zoo_1.8-12          jsonlite_1.8.4
## [45] crosstalk_1.2.0    dplyr_1.1.2        R.oo_1.25.0         magrittr_2.0.3
## [49] modeltools_0.2-23  Matrix_1.5-4       Rcpp_1.0.10         DescTools_0.99.48
## [53] munsell_0.5.0      fansi_1.0.4        viridis_0.6.3       lifecycle_1.0.3
## [57] R.methodsS3_1.8.2  stringi_1.7.12     multcomp_1.4-23     yaml_2.3.7
## [61] MASS_7.3-60        rootSolve_1.8.2.3  plyr_1.8.8          grid_4.2.3
## [65] parallel_4.2.3     lmom_2.9           lattice_0.21-8      splines_4.2.3
## [69] ps_1.7.5           knitr_1.42         pillar_1.9.0        boot_1.3-28.1
## [73] gld_2.6.6          codetools_0.2-19   stats4_4.2.3        glue_1.6.2
## [77] evaluate_0.21      data.table_1.14.8  png_0.1-8           vctrs_0.6.2
## [81] cellranger_1.1.0   gtable_0.3.3       openssl_2.0.6       purrr_1.0.1
## [85] cachem_1.0.8       xfun_0.39          coin_1.4-2          libcoin_1.0-9
## [89] e1071_1.7-13       RSpectra_0.16-1    class_7.3-22        survival_3.5-5
## [93] viridisLite_0.4.2  tibble_3.2.1       ellipsis_0.3.2      TH.data_1.1-2
## [97] here_1.0.1
```

```
session_info.show()
```

```
## -----  
## Bio                1.78  
## joblib             1.1.1  
## numpy             1.19.0  
## pandas            1.3.2  
## purf              NA  
## scipy             1.8.0  
## session_info      1.0.0  
## sklearn           0.24.2  
## -----  
## Python 3.8.2 (default, Mar 26 2020, 10:45:18) [Clang 4.0.1 (tags/RELEASE_401/final)]  
## macOS-10.16-x86_64-i386-64bit  
## -----  
## Session information updated at 2023-05-18 12:31
```

Section 6

Top candidate comparisons

6.1 Clustering analysis

6.1.1 Analysis

In R:

```
library(DT)
library(factoextra)
library(NbClust)
library(cluster)
library(ggplot2)
library(cowplot)
```

```
pfpv_data <- read.csv("./data/supplementary_data_5_pfpv_purf_oob_predictions.csv", row.names = 1,
  ↪ check.names = FALSE)
pfpv_data <- pfpv_data[order(-pfpv_data$`00B score filtered`), ]
pfpv_threshold <- pfpv_data[pfpv_data$antigen_label == 1, ]$`00B score
  ↪ filtered`[sum(pfpv_data$antigen_label == 1) / 2]
top_pf_pfpv <- rownames(pfpv_data[pfpv_data$species == "pf" & pfpv_data$antigen_label == 0 &
  ↪ pfpv_data$`00B score filtered` >= pfpv_threshold, ])
top_pv_pfpv <- rownames(pfpv_data[pfpv_data$species == "pv" & pfpv_data$antigen_label == 0 &
  ↪ pfpv_data$`00B score filtered` >= pfpv_threshold, ])
```

```
pfpv_prox <- read.csv("~/Downloads/pfpv_proximity_values.csv", check.names = FALSE)
pfpv_dist <- 1 - pfpv_prox
rownames(pfpv_dist) <- colnames(pfpv_dist)
pfpv_dist <- pfpv_dist[
  rownames(pfpv_dist) %in% c(top_pf_pfpv, top_pv_pfpv),
  colnames(pfpv_dist) %in% c(top_pf_pfpv, top_pv_pfpv)
]
rm(pfpv_prox)
```

```

# Gap statistic
gap_stat <- clusGap(pfpv_dist, K.max = 10, hcut, B = 100, hc_method = "ward.D2")
p0_1 <- fviz_gap_stat(gap_stat, maxSE = list(method = "Tibs2001SEmax", SE.factor = 2)) +
  labs(title = "Gap statistic method")

# Silhouette method
p0_2 <- fviz_nbclust(pfpv_dist, k.max = 10, hcut, hc_method = "ward.D2", method = "silhouette") +
  labs(title = "Silhouette method")

# Elbow method
p0_3 <- fviz_nbclust(pfpv_dist, k.max = 10, hcut, hc_method = "ward.D2", method = "wss") +
  geom_vline(xintercept = 3, linetype = 2, color = "steelblue") +
  labs(title = "Elbow method")

save(p0_1, p0_2, p0_3, file = "./rdata/top_candidate_optimal_num_clusters.RData")

```

```

library(umap)
library(ggplot2)
library(ggrepel)
library(cowplot)

```

```

top_pfpv_hc <- hclust(as.dist(pfpv_dist), method = "ward.D2")
top_pfpv_2_clusters <- cutree(top_pfpv_hc, k = 2)
top_pfpv_3_clusters <- cutree(top_pfpv_hc, k = 3)
top_pfpv_2_clusters[top_pfpv_2_clusters == 1] <- 4
top_pfpv_2_clusters[top_pfpv_2_clusters == 2] <- 3
top_pfpv_2_clusters <- top_pfpv_2_clusters - 2
top_pfpv_3_clusters[top_pfpv_3_clusters == 1] <- 5
top_pfpv_3_clusters[top_pfpv_3_clusters == 2] <- 6
top_pfpv_3_clusters[top_pfpv_3_clusters == 3] <- 4
top_pfpv_3_clusters <- top_pfpv_3_clusters - 3
save(top_pfpv_2_clusters, top_pfpv_3_clusters,
  file = "./rdata/top_candidate_clusters.RData"
)

```

```

clusters <- top_pfpv_2_clusters
data <- read.csv("./data/supplementary_data_5_pfpv_purf_oob_predictions.csv", row.names = 1,
  ↪ check.names = FALSE)

```

```

load("./rdata/pfpv_prox_mds_umap.RData")
umap_df_ <- umap_df
umap_df$label <- data$antigen_label
umap_df <- rbind(umap_df[umap_df$label == 1, ], umap_df[rownames(umap_df) %in% names(clusters), ])
umap_df$group <- ""
umap_df$group[umap_df$label == 1] <- "Known antigen"
umap_df$group[rownames(umap_df) %in% c("PF3D7_0304600.1-p1", "PF3D7_0424100.1-p1",
↪ "PF3D7_0206900.1-p1", "PF3D7_0209000.1-p1")] <- "Reference antigen"
umap_df$group[rownames(umap_df) %in% names(clusters)[clusters == 1]] <- "Group 1"
umap_df$group[rownames(umap_df) %in% names(clusters)[clusters == 2]] <- "Group 2"
umap_df_text <- umap_df[umap_df$label == 1, ]
umap_df_text$label_text <- ""
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0304600.1-p1"] <- "PfCSP"
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0424100.1-p1"] <- "RH5"
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0206900.1-p1"] <- "MSP5"
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0209000.1-p1"] <- "P230"
umap_df_text$label_text[rownames(umap_df_text) == "PVP01_0835600.1-p1"] <- "PvCSP"
umap_df_text$label_text[rownames(umap_df_text) == "PVP01_0623800.1-p1"] <- "DBP"
umap_df_text$label_text[rownames(umap_df_text) == "PVP01_0728900.1-p1"] <- "MSP1"

umap_p1 <- ggplot() +
  geom_point(data = umap_df_, aes(x = X1, y = X2), color = "grey90", alpha = 0.5) +
  geom_point(
    data = umap_df[umap_df$label == 1, ], aes(x = X1, y = X2), shape = 21, color = "black",
    fill = "#FFFF33", stroke = 0.3, size = 2, alpha = 0.5
  ) +
  geom_point(data = umap_df[umap_df$label == 0, ], aes(x = X1, y = X2, fill = group), shape = 21, alpha
↪ = 0.5) +
  geom_text_repel(
    data = umap_df_text, aes(x = X1, y = X2, label = label_text), size = 3.5,
    nudge_x = 1.5, nudge_y = -1, point.padding = 0.1
  ) +
  scale_fill_manual(breaks = c("Group 1", "Group 2"), values = c("#03a1fc", "#984EA3")) +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    plot.margin = ggplot2::margin(5, 5, 5, 5, "pt"),
    axis.title = element_text(colour = "black"),
    axis.text = element_text(colour = "black"),
    legend.position = "bottom"
  )

```

```
) +
  guides(fill = guide_legend(title = "")) +
  xlab("Dimension 1") +
  ylab("Dimension 2")
```

```
clusters <- top_pfpv_3_clusters
data <- read.csv("./data/supplementary_data_5_pfpv_purf_oob_predictions.csv", row.names = 1,
  ↪ check.names = FALSE)
load("./rdata/pfpv_prox_mds_umap.RData")
umap_df_ <- umap_df
umap_df$label <- data$antigen_label
umap_df <- rbind(umap_df[umap_df$label == 1, ], umap_df[rownames(umap_df) %in% names(clusters), ])
umap_df$group <- ""
umap_df$group[umap_df$label == 1] <- "Known antigen"
umap_df$group[rownames(umap_df) %in% c("PF3D7_0304600.1-p1", "PF3D7_0424100.1-p1",
  ↪ "PF3D7_0206900.1-p1", "PF3D7_0209000.1-p1")] <- "Reference antigen"
umap_df$group[rownames(umap_df) %in% names(clusters)[clusters == 1]] <- "Group 1"
umap_df$group[rownames(umap_df) %in% names(clusters)[clusters == 2]] <- "Group 2"
umap_df$group[rownames(umap_df) %in% names(clusters)[clusters == 3]] <- "Group 3"
umap_df_text <- umap_df[umap_df$label == 1, ]
umap_df_text$label_text <- ""
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0304600.1-p1"] <- "PfCSP"
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0424100.1-p1"] <- "RH5"
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0206900.1-p1"] <- "MSP5"
umap_df_text$label_text[rownames(umap_df_text) == "PF3D7_0209000.1-p1"] <- "P230"
umap_df_text$label_text[rownames(umap_df_text) == "PVP01_0835600.1-p1"] <- "PvCSP"
umap_df_text$label_text[rownames(umap_df_text) == "PVP01_0623800.1-p1"] <- "DBP"
umap_df_text$label_text[rownames(umap_df_text) == "PVP01_0728900.1-p1"] <- "MSP1"

umap_p2 <- ggplot() +
  geom_point(data = umap_df_, aes(x = X1, y = X2), color = "grey90", alpha = 0.5) +
  geom_point(
    data = umap_df[umap_df$label == 1, ], aes(x = X1, y = X2), shape = 21, color = "black",
    fill = "#FFFF33", stroke = 0.3, size = 2, alpha = 0.5
  ) +
  geom_point(data = umap_df[umap_df$label == 0, ], aes(x = X1, y = X2, fill = group), shape = 21, alpha
    ↪ = 0.5) +
  geom_text_repel(
    data = umap_df_text, aes(x = X1, y = X2, label = label_text), size = 3.5,
```

```

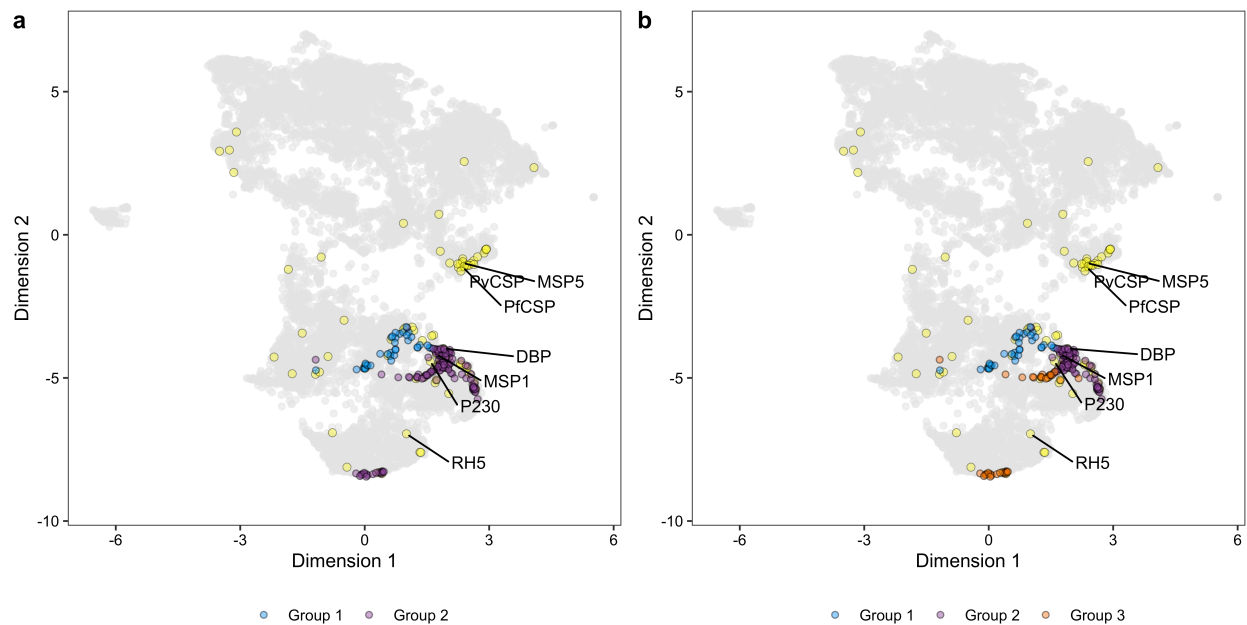
    nudge_x = 1.5, nudge_y = -1, point.padding = 0.1
  ) +
  scale_fill_manual(
    breaks = c("Group 1", "Group 2", "Group 3"),
    values = c("#03a1fc", "#984EA3", "#FF7F00")
  ) +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    plot.margin = ggplot2::margin(5, 5, 5, 5, "pt"),
    axis.title = element_text(colour = "black"),
    axis.text = element_text(colour = "black"),
    legend.position = "bottom"
  ) +
  guides(fill = guide_legend(title = "")) +
  xlab("Dimension 1") +
  ylab("Dimension 2")

```

```

umap_p_combined <- plot_grid(umap_p1, umap_p2, nrow = 1, labels = c("a", "b"))
umap_p_combined

```



```
load(file = "./rdata/top_candidate_clusters.RData")
species <- c(rep("pf", 145), rep("pv", 45))
cat("Two clusters\n")
```

```
## Two clusters
```

```
table(top_pfpv_2_clusters, species)
```

```
##              species
## top_pfpv_2_clusters pf  pv
##              1    0  35
##              2  145  10
```

```
cat("Three clusters\n")
```

```
## Three clusters
```

```
table(top_pfpv_3_clusters, species)
```

```
##              species
## top_pfpv_3_clusters pf  pv
##              1    0  35
##              2  107   9
##              3   38   1
```

```
protein_id_1 <- names(top_pfpv_3_clusters)[top_pfpv_3_clusters == 1]
protein_id_2 <- names(top_pfpv_3_clusters)[top_pfpv_3_clusters == 2]
protein_id_3 <- names(top_pfpv_3_clusters)[top_pfpv_3_clusters == 3]
save(protein_id_1, protein_id_2, protein_id_3, file = "./rdata/clustering_groups.RData")
```

```
gene_accession_1 <- str_replace_all(protein_id_1, "\\.[1-9]-p1", "")
gene_accession_2 <- str_replace_all(protein_id_2, "\\.[1-9]-p1", "")
gene_accession_3 <- str_replace_all(protein_id_3, "\\.[1-9]-p1", "")
```

```
write.table(data.frame("acc" = gene_accession_1),
  sep = ", ", row.names = FALSE, col.names = FALSE,
  file = "./other_data/top_candidates_gene_accession_1.csv"
)
```



```
write.table(data.frame("acc" = gene_accession_2),
  sep = ",", row.names = FALSE, col.names = FALSE,
  file = "./other_data/top_candidates_gene_accession_2.csv"
)
write.table(data.frame("acc" = gene_accession_3),
  sep = ",", row.names = FALSE, col.names = FALSE,
  file = "./other_data/top_candidates_gene_accession_3.csv"
)
```

6.2 Gene ontology analysis

6.2.1 Analysis

In Bash:

```
python ./other_data/goea.py -s "./other_data/top_candidates_gene_accession_1.csv" \
-p "./other_data/PlasmoDB-62_pfpv_G0.gaf" \
-n "./other_data/go-basic.obo" \
-o "./other_data/goea_result_1.xlsx"

python ./other_data/goea.py -s "./other_data/top_candidates_gene_accession_2.csv" \
-p "./other_data/PlasmoDB-62_pfpv_G0.gaf" \
-n "./other_data/go-basic.obo" \
-o "./other_data/goea_result_2.xlsx"

python ./other_data/goea.py -s "./other_data/top_candidates_gene_accession_3.csv" \
-p "./other_data/PlasmoDB-62_pfpv_G0.gaf" \
-n "./other_data/go-basic.obo" \
-o "./other_data/goea_result_3.xlsx"
```

6.2.2 Plotting

In R:

```
library(ggplot2)
library(gridExtra)
library(shadowtext)
library(ggforce)
library(reshape)
```

```
library(cowplot)
library(scales)
```

```
process_goea_res <- function(file_name) {
  data <- read.csv(file_name, header = TRUE, fill = TRUE, quote = '\'', stringsAsFactors = FALSE)
  # calculate percentage from the ratio column
  data$num_genes <- apply(data, 1, function(x) as.integer(unlist(strsplit(x["ratio_in_study"],
↪  "/" ))[1]))
  # select for enrichment (e) data rows
  data <- data[data$enrichment == "e", ]

  # select for biological process (BP) data rows
  res_bp <- data[data$NS == "BP", ]
  res_bp$`-Log10FDR` <- -log10(res_bp$p_fdr_bh)
  res_bp <- res_bp[order(res_bp$`-Log10FDR`), ]
  res_bp$group <- rep("Biological process", nrow(res_bp))

  # select for cellular component (CC) data rows
  res_cc <- data[data$NS == "CC", ]
  res_cc$`-Log10FDR` <- -log10(res_cc$p_fdr_bh)
  res_cc <- res_cc[order(res_cc$`-Log10FDR`), ]
  res_cc$group <- rep("Cellular component", nrow(res_cc))

  # select for molecular function (MF) data rows
  res_mf <- data[data$NS == "MF", ]
  res_mf$`-Log10FDR` <- -log10(res_mf$p_fdr_bh)
  res_mf <- res_mf[order(res_mf$`-Log10FDR`), ]
  res_mf$group <- rep("Molecular function", nrow(res_mf))

  ds <- do.call(rbind, list(res_mf, res_cc, res_bp))
  ds$name <- factor(ds$name, levels = ds$name)
  ds$group <- factor(ds$group, levels = c("Biological process", "Cellular component", "Molecular
↪  function"))

  return(ds)
}
```

```
ds_1 <- process_goea_res("./other_data/goea_result_1.csv")
ds_2 <- process_goea_res("./other_data/goea_result_2.csv")
ds_3 <- process_goea_res("./other_data/goea_result_3.csv")
```

```
p1 <- ggplot(ds_1, aes(x = name, y = `-Log10FDR`)) +
  geom_col(width = 0.05) +
  geom_point(size = 5, shape = 21, color = "black", fill = "grey20") +
  geom_text(aes(label = sprintf("%.0f", num_genes)), nudge_y = 0, size = 2.5, color = "white") +
  scale_x_discrete(labels = label_wrap(50)) +
  coord_flip() +
  ggforce::facet_col(facets = vars(group), scales = "free_y", space = "free") +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.spacing = unit(0, "pt"),
    strip.background = element_blank(),
    panel.border = element_rect(colour = "black", size = 0.5),
    plot.title = element_text(hjust = 0.5),
    plot.margin = ggplot2::margin(5, 5, 85, 5, "pt"),
    legend.text = element_text(colour = "black", ),
    axis.title.x = element_text(colour = "black", ),
    axis.title.y = element_text(colour = "black", ),
    axis.text.x = element_text(colour = "black", ),
    axis.text.y = element_text(colour = "black", )
  ) +
  xlab("") +
  ylab(expression("-Log"[10] * "FDR")) +
  ggtitle("Group 1 (35 candidates)")
```

```
p2 <- ggplot(ds_2, aes(x = name, y = `-Log10FDR`)) +
  geom_col(width = 0.05) +
  geom_point(size = 5, shape = 21, color = "black", fill = "grey20") +
  geom_text(aes(label = sprintf("%.0f", num_genes)), nudge_y = 0, size = 2.5, color = "white") +
  scale_x_discrete(labels = label_wrap(50)) +
  coord_flip() +
  ggforce::facet_col(facets = vars(group), scales = "free_y", space = "free") +
  theme_bw() +
```

```

theme(
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  panel.spacing = unit(0, "pt"),
  strip.background = element_blank(),
  panel.border = element_rect(colour = "black", size = 0.5),
  plot.title = element_text(hjust = 0.5),
  plot.margin = ggplot2::margin(5, 0, 5, 5, "pt"),
  legend.text = element_text(colour = "black", ),
  axis.title.x = element_text(colour = "black", ),
  axis.title.y = element_text(colour = "black", ),
  axis.text.x = element_text(colour = "black", ),
  axis.text.y = element_text(colour = "black", )
) +
xlab("") +
ylab(expression("-Log"[10] * "FDR")) +
ggtitle("Group 2 (116 candidates)")

```

```

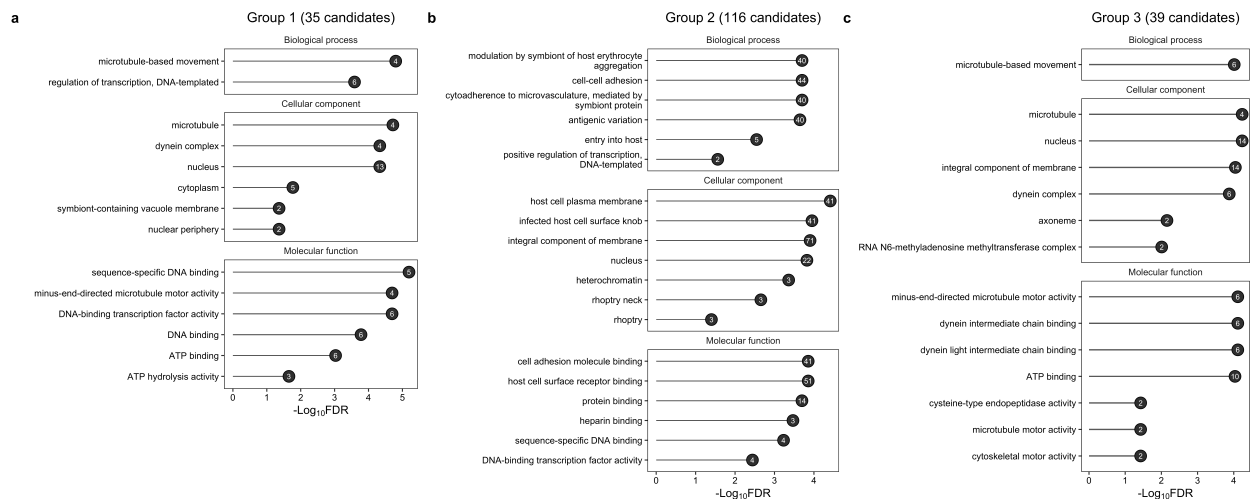
p3 <- ggplot(ds_3, aes(x = name, y = `-Log10FDR`)) +
  geom_col(width = 0.05) +
  geom_point(size = 5, shape = 21, color = "black", fill = "grey20") +
  geom_text(aes(label = sprintf("%.0f", num_genes)), nudge_y = 0, size = 2.5, color = "white") +
  scale_x_discrete(labels = label_wrap(50)) +
  coord_flip() +
  ggforce::facet_col(facets = vars(group), scales = "free_y", space = "free") +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.spacing = unit(0, "pt"),
    strip.background = element_blank(),
    panel.border = element_rect(colour = "black", size = 0.5),
    plot.title = element_text(hjust = 0.5),
    plot.margin = ggplot2::margin(5, 5, 5, 0, "pt"),
    legend.text = element_text(colour = "black", ),
    axis.title.x = element_text(colour = "black", ),
    axis.title.y = element_text(colour = "black", ),
    axis.text.x = element_text(colour = "black", ),
    axis.text.y = element_text(colour = "black", )
  )

```

```
) +
xlab("") +
ylab(expression("-Log"[10] * "FDR")) +
ggtitle("Group 3 (39 candidates)")
```

Final plot

```
p_combined <- plot_grid(p1, p2, p3, nrow = 1, rel_widths = c(0.3, 0.3, 0.3), labels = c("a", "b", "c"))
p_combined
```



6.3 Candidate antigen characterization

6.3.1 Processing table

In R:

```
library(tibble)
```

```
prediction <- read.csv("../data/supplementary_data_5_pfpv_purf_oob_predictions.csv", row.names = 1,
  ↪ check.names = FALSE)
prediction <- prediction[order(-prediction$`00B score filtered`), ]
threshold <- prediction[prediction$antigen_label == 1, ]$`00B score
  ↪ filtered`[sum(prediction$antigen_label == 1) / 2]
```

```
labeled_pos <- rownames(prediction)[prediction$antigen_label == 1]
top_pfpv <- rownames(prediction[prediction$antigen_label == 0 & prediction$`00B score filtered` >=
  ↪ threshold, ])
```

```
pfpv_prox <- read.csv("~/Downloads/pfpv_proximity_values.csv", check.names = FALSE)
pfpv_dist <- 1 - pfpv_prox
rm(pfpv_prox)
rownames(pfpv_dist) <- colnames(pfpv_dist)
pfpv_dist <- pfpv_dist[, labeled_pos]
```

```
ds <- prediction[c(labeled_pos, top_pfpv), "00B score filtered", drop = FALSE]
ds <- rownames_to_column(ds, "Protein ID")
colnames(ds)[2] <- "Score"
```

Add clustering groups

```
load(file = "./rdata/clustering_groups.RData")
ds$Group <- ""
ds$Group[ds$`Protein ID` %in% protein_id_1] <- "Group 1"
ds$Group[ds$`Protein ID` %in% protein_id_2] <- "Group 2"
ds$Group[ds$`Protein ID` %in% protein_id_3] <- "Group 3"
ds$Group[ds$`Protein ID` %in% labeled_pos] <- "Known antigen"
ds$Group[ds$`Protein ID` %in% c(
  "PF3D7_0304600.1-p1", "PF3D7_0424100.1-p1",
  "PF3D7_0206900.1-p1", "PF3D7_0209000.1-p1",
  "PVP01_0835600.1-p1", "PVP01_0623800.1-p1",
  "PVP01_0728900.1-p1"
)] <- "Reference antigen"
```

Add gene products

```
gene_products <- rbind(
  read.csv("./other_data/pf3d7_gene_products_v62.csv"),
  read.csv("./other_data/pvp01_gene_products_v62.csv")
)
colnames(gene_products) <- c("Protein ID", "Gene product")
ds <- merge(x = ds, y = gene_products, by = "Protein ID", all.x = TRUE)
```

Add closest reference antigen and the distance

```
pfpv_ad_ctrl_res <- read.csv("./other_data/pfpv_ad_ctrl_res.csv", check.names = FALSE, row.names = 1)
```

```

ds$`Closest known antigen` <- ""
ds$`Known antigen source` <- "Intersect"
ds$`Closest distance` <- -1
for (i in 1:nrow(ds)) {
  prot <- ds$`Protein ID`[i]
  tmp <- pfpv_dist[prot, ]
  ds[i, "Closest known antigen"] <- names(tmp)[which.min(tmp)]
  ds[i, "Closest known antigen"] <- paste0(
    ds[i, "Closest known antigen"], " (",
    gene_products[
      gene_products$`Protein ID` == ds[i, "Closest known antigen"],
      "Gene product"
    ], ")")
}
if (names(tmp)[which.min(tmp)] %in% c(
  "PF3D7_0304600.1-p1", "PF3D7_0424100.1-p1",
  "PF3D7_0206900.1-p1", "PF3D7_0209000.1-p1",
  "PVP01_0835600.1-p1", "PVP01_0623800.1-p1",
  "PVP01_0728900.1-p1"
)) {
  ds[i, "Known antigen source"] <- "Reference"
} else {
  ds[i, "Known antigen source"] <- pfpv_ad_ctrl_res[names(tmp)[which.min(tmp)], "source"]
}
ds[i, "Closest distance"] <- tmp[which.min(tmp)]
}

# Sort based on scores
ds <- ds[order(-ds$Score), ]
write.csv(ds, file = "./data/supplementary_data_6_antigen_characterization.csv", row.names = FALSE)

```

```
sessionInfo()
```

```

## R version 4.2.3 (2023-03-15)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur ... 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRlapack.dylib

```

```
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats      graphics  grDevices utils      datasets  methods   base
##
## other attached packages:
## [1] tibble_3.2.1    scales_1.2.1    reshape_0.8.9   ggforce_0.4.1
## [5] shadowtext_0.1.2 gridExtra_2.3    ggrepel_0.9.3   umap_0.2.10.0
## [9] cowplot_1.1.1   cluster_2.1.4    NbClust_3.0.1    factoextra_1.0.7
## [13] ggplot2_3.4.2   DT_0.27
##
## loaded via a namespace (and not attached):
## [1] reticulate_1.28  styler_1.9.1     tidyselect_1.2.0 xfun_0.39
## [5] purrr_1.0.1      lattice_0.21-8    colorspace_2.1-0 vctrs_0.6.2
## [9] generics_0.1.3   htmltools_0.5.5   yaml_2.3.7        utf8_1.2.3
## [13] rlang_1.1.1      R.oo_1.25.0       pillar_1.9.0      glue_1.6.2
## [17] withr_2.5.0      R.utils_2.12.2    tweenr_2.0.2      plyr_1.8.8
## [21] R.cache_0.16.0   lifecycle_1.0.3   munsell_0.5.0     gtable_0.3.3
## [25] R.methodsS3_1.8.2 htmlwidgets_1.6.2 codetools_0.2-19  evaluate_0.21
## [29] knitr_1.42       fastmap_1.1.1     fansi_1.0.4       Rcpp_1.0.10
## [33] openssl_2.0.6    jsonlite_1.8.4     farver_2.1.1      RSpectra_0.16-1
## [37] askpass_1.1       png_0.1-8          digest_0.6.31     bookdown_0.34
## [41] dplyr_1.1.2      polyclip_1.10-4    grid_4.2.3        cli_3.6.1
## [45] tools_4.2.3      magrittr_2.0.3     pkgconfig_2.0.3    MASS_7.3-60
## [49] Matrix_1.5-4     rmarkdown_2.21     rstudioapi_0.14    R6_2.5.1
## [53] compiler_4.2.3
```


Section 7

Summary of candidates

The table provides a summary of the top antigen candidates and the 90 known *P. vivax* and *P. falciparum* antigens, including the 7 reference antigens. The table contains columns for protein IDs, probability scores, antigen/candidate groups, gene products, closest known antigens, the source of the closest known antigen, and the distance to the closest antigens. Grey bars in the table show probability scores or closest euclidean distances, with values ranging from 0 to 1.

To access additional information in the rightmost columns, scroll horizontally. For column filtering, click on the white boxes below the headers and type in threshold criteria (e.g., “PVP01” to select *P. vivax* proteins). You can also arrange values in ascending or descending order by selecting the arrows adjacent to the headers.

Protein ID	Score	Group	Gene product	Closest known antigen	Known antigen source	Closest distance
All	All	All	All	All	All	All