

ABSTRACT

Title of Dissertation: AN EVALUATION OF CLUSTERING ALGORITHMS FOR MODELING GAME-BASED ASSESSMENT WORK PROCESSES

W. Austin Fossey, Doctor of Philosophy, 2017

Dissertation directed by: Professor Laura Stapleton and Professor Tracy Sweet, Department of Human Development and Quantitative Methodology

Game-based assessments (GBAs) use game design elements to make assessments more engaging for students and capture response data about work processes. GBA response data are often too complex to plan for every potential response pattern, so some researchers have turned to exploratory cluster analysis to classify students' work processes. This paper identifies the design elements specific to GBAs and investigates how well k -means, self-organizing maps (SOM), and robust clustering using links (ROCK) clustering algorithms group response patterns in prototypical GBA response data. Results from a simulation study are discussed, and a tutorial is provided with recommendations of general considerations and best practices for analyzing GBA data with clustering algorithms.

AN EVALUATION OF CLUSTERING ALGORITHMS FOR MODELING GAME-
BASED ASSESSMENT WORK PROCESSES

by

William Austin Fossey

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2017

Advisory Committee:
Professor Laura Stapleton, Chair
Professor Jeffrey Haring
Professor Kent Norman
Dr. Andre Rupp
Professor Tracy Sweet

© Copyright by
William Austin Fossey
2017

Table of Contents

Table of Contents	ii
List of Tables	v
List of Figures	vii
Chapter 1: Introduction	1
Evidence from Work Processes in GBAs	4
The Challenge of Interpreting Work Process Data in GBAs	6
Data Collection Considerations for Clustering GBA Data	8
Study Purpose	12
Chapter 2: Game-Based Assessment	15
Architectural Game Design Elements	15
Experiential Game Design Elements	19
GBAs: Assessments with Construct-Irrelevant Relatedness Elements	25
The Use of GBAs in Education	26
Challenges of Using Games for Assessment	30
Designing GBAs with Evidence-Centered Design	33
Measurement Models: Principled versus Empirical Approaches in GBAs	35
Clustering Algorithms	41
Chapter 3: Feasibility Study	60
Simulated Data for the Feasibility Study	60

GBA Data for the Feasibility Study.....	62
Procedure	66
Performance of the Clustering Algorithms for Datasets S1 and S2.....	75
Performance of the Clustering Algorithms for Datasets M1 and M2.....	82
Discussion of the Feasibility Study.....	87
Chapter 4: Methods for Full Study	90
Simulation Design Summary	92
Data Generation	93
Number of Clusters for the Simulation.....	95
Number of Cases in the Simulated Clusters	96
Data Type and Response Probabilities in Simulated Clusters	97
Amount of Overlap Between Clusters	99
Number of Response Variables in the Simulation.....	102
Methods and Outcome Measures.....	104
Tutorial Example	108
Chapter 5: Simulation Study Results	111
Research Question 1: Kappa Differences by Varying Individual Conditions	114
Research Question 2: Conditions Under Which Kappa is Expected to Exceed 0.80 .	122
Research Question 3: Conditions In Which Algorithms Can Validate Each Other ...	125
Research Question 4: Finding the Number of Clusters with DBI	127

Simulation Study Results Summary	132
Chapter 6: Tutorial Example Study	134
Beanstalk Game	135
Beanstalk Student Data	137
Step 1: Recode the Data and Select Variables	138
Step 2: Select Clustering Algorithms.....	142
Step 3. Plan Cross Validation	143
Step 4: Run the Algorithms.....	144
Step 5: Validating Across Algorithms	150
Step 6: Validating Against Holdout Sample and External Criterion	153
Step 7: Interpreting, Labeling, and Grouping Clusters	161
Step 8: Students' Changes in Cluster Membership Over Time	164
Tutorial Example Summary.....	168
Chapter 7: Discussion	170
Summary of Results	170
Study Limitations.....	174
Implications.....	177
Future of GBA	180
Appendix A: R Scripts for Full Study.....	185
References.....	196

List of Tables

<u>Table</u>	<u>Page</u>
1 S1 Simulated Data Parameters.....	61
2a S2 Simulated Data Parameters.....	61
2b Example of S2 Simulated Data (First Six Cases)	61
3 Excerpts of Number Jumble Log File Data (First Five Moves).....	64
4 Excerpt of M1 Data (First Six Cases)	66
5 Excerpt of M2 Simulated Data (First Six Cases)	67
6 Example of a Pivot Table Showing Classification Agreement Between Two Clustering Solutions.....	73
7 S1: True Cluster Recovery by Algorithm.....	76
8 S2: True Cluster Recovery by Algorithm.....	77
9 S1: Cluster Agreement Between Pairs of Algorithms.....	77
10 S2: Cluster Agreement Between Algorithms.....	79
11 DBI Results by Solution for S1 and S2.....	80
12 M1: Cluster Agreement Between Algorithms in a Two Cluster Solution....	85
13 M2: Cluster Agreement Between Algorithms in a Two Cluster Solution....	86
14 M2: Cluster Agreement Between Algorithms in a Three Cluster Solution...	87
15 DBI Results by Solution for M1 and M2.....	87
16 Total Number of Variables by Number of Clusters.....	105
17 Simulation Design of Four Varied Parameters Resulting in 48 Cells.....	109
18 Actual Number of Datasets Generated per Cell.....	113
19 Correlations between RMSD and Kappa by Study Cell and Clustering Algorithm.....	121

20	Percentage of Solutions with $Kappa \geq 0.80$ by Study Cell and Clustering Algorithm.....	123
21a	Algorithms' Agreement (Kappa) by k (Level 2-11)	152
21b	Algorithms' Fit (DBI) by k (Level 2-11)	152

List of Figures

<u>Figure</u>	<u>Page</u>
1	Example of the Classic 15 Puzzle..... 3
2	Example of a Time-Based Connection Element’s Influence on Game Performance..... 19
3	Conceptual Assessment Framework (CAF) Structure..... 35
4	Visualization of the Self-Organizing Map Learning Process..... 55
5	Level Mul_1 from Number Jumble..... 63
6	Example of Recoding Sequence of Two-Dimensional Cluster Solutions..... 72
7a	Cluster Center Magnitudes by Variable and Clustering Solution in S1..... 81
7b	Cluster Center Magnitudes by Variable and Clustering Solution in S2..... 81
8	Mean Kappa Rates by Algorithm for Different Numbers of Clusters k 116
9	Mean Kappa Rates by Algorithm for Two Different Average Cluster Sizes N_k 117
10	Mean Kappa Rates by Algorithm for Two Different Variances in the Data’s Cluster Sizes N_k 118
11	Mean Kappa Rates by Algorithm for Three Different Probabilities of Response in the Relevant Variables Defining Clusters..... 120
12	Kappa Distributions by Cluster Size, for Each Combination of Response Probability and k Clusters..... 124
13	Percentage of Datasets Correctly Identified, Overestimated, and Underestimated, as Determined by Davies-Bouldin Index (DBI)..... 129
14	DBI Values for Three Clustering Algorithms when Varying the Number of Requested Clusters from $k = 17$ -23 Clusters..... 132
15	Screenshot from Beanstalk..... 136
16	Screenshot of Beanstalk Log File Dataset..... 138
17	DBI Values by Clustering Algorithm and Number of Clusters (k) for Beanstalk Level 3-2..... 153

18	DBI Comparisons between Training Sample and Holdout Sample.....	156
19	Correct Outcome Rates (Rates of Beating the Level) for the Six Clusters Identified in Beanstalk Level 2-9.....	158
20a	Characterization of Flower Placement for Students in Cluster 4 on Beanstalk Level 2-9.....	159
20b	One Possible Correct Solution for Cluster 4 in Beanstalk Level 2-9.....	159
20c	The Other Possible Correct Solution for Cluster 4 in Beanstalk Level 2-9....	159
21	Flower Placement Rates for Cluster 4 in Beanstalk Level 2-9.....	160
22	Group Transition Map of Students' Clusters Across 75 Levels of Beanstalk.	167

Chapter 1: Introduction

While educational games are not new, the field has seen tremendous growth as technology has expanded the capabilities and availability of educational games. This growth mirrors a larger trend in gaming, where blockbuster video game franchises are being joined by “casual games” (games that are played for only a few minutes at a time, often on mobile devices), which are played by 40% of U.S. households (Schleiner, 2001).

Though skepticism persists, proponents of educational games observe that games can model learning in an engaging, challenging, and entertaining medium that can complement or enhance traditional learning methods (Teknibaş, 2014), and there is growing research demonstrating the benefits of games for student learning and classroom teaching (e.g., Shapiro, Teknibaş, Schwartz, & Darvasi, 2014). Part of the appeal of educational games is that they provide an environment where students can apply learning in a relevant way, but they can also record data and adapt the experience so that it keeps students engaged. If the right data are collected, games may even provide a view into how students learn (Shapiro, 2014b).

As the boundaries between games and learning continue to blur, another movement is gaining momentum in the assessment industry which blurs the lines between learning and assessment. Like educational games, this movement to use assessments as instructional devices exists at both a grassroots level in classrooms and in organized research initiatives like Educational Testing Services’ (ETS) Cognitively Based Assessments of, for, and as Learning (CBAL) project (Bennett, 2010). In this sphere, assessment designers are building assessments which not only measure the

outcomes *of* learning, but which also support instruction and serve as learning tools in their own right.

Given the current trends of increasing use of game-based learning and assessments as learning tools, it is not surprising that there is growing interest in game-based assessments (GBAs). Understanding the intersection of games and assessments is important for understanding the context of the issues GBA designers face. An assessment is any instrument that collects evidence from a sample of students' behavior and provides a quantitative value from which an inference about the students can be made (Crocker & Algina, 2008). The definition of a game is not quite as clear cut, and perspectives on games can vary based on how the games are designed and played (Mislevy et al., 2014; Salen & Zimmerman, 2004).

A game can become an assessment if it can yield quantitative data to support valid inferences about students, but making the leap from collecting gameplay data and evaluating the student results can be daunting because even the simplest games require a complex set of actions, decisions, strategizing, and mistakes in gameplay. Consider the simple, classic 15 Puzzle, which is a puzzle where the player slides tiles around to get the numbered tiles into the correct order (see Figure 1). Only one open space is available at any given time, so a player has a maximum of four possible moves at each step of his or her gameplay work process. Although the game is simple, a good player can think strategically about how to slide tiles to position them such that the puzzle can be solved in fewer moves—a goal which served as the test case for early computer search algorithm research (e.g., Brünger, Marzetta, Fukuda, & Nievergelt, 1999).

7	8	9	1
11	14	4	13
3	2	10	15
12	6	5	

Figure 1. Example of the classic 15 Puzzle where players must correctly order the tiles.

There are an astounding $16!/2$ possible configurations for the 15 Puzzle, and the most efficient solution for the hardest initial board state requires 80 moves (Brünger et al., 1999). If one were to record that gameplay work process, one might imagine a dataset where one simply records the tile that was moved, its original position, and its new position; however, one can go further and create variables recording the sequence of moves, the state of the puzzle board at each move, the time it took for the player to make his or her decision, etc. All of these variables may be useful evidence for evaluating what the player was thinking and how he or she played the game, and these data could serve to distinguish expert players from novices.

To add to the complexity of the gameplay data themselves, there is the added question of whether the game (used as an assessment instrument) is designed to provide any evaluation of the gameplay work process. The 15 Puzzle can only inform the player if the puzzle has been solved, but there is no mechanism for reporting how efficiently the player solved that puzzle. Mislevy et al. (2014) noted that the GBA only needs to collect evidence that can be used for assessment inferences—scoring and evaluation can occur outside of the game. This distinction implies that GBAs can take several forms, as is shown in the following section.

Evidence from Work Processes in GBAs

Mislevy et al. (2014) defined three GBA paradigms: assessment outside the game, assessment inside the game as pre-specified work products, and assessment inside the game using evidence from work processes. This third GBA paradigm uses assessment inside the game, but the data used for the assessment inferences come from data streams that reflect the work processes or gameplay of the students in the game (Shute, 2011). Students make many gameplay choices that manifest as patterns of responses and interactions with the game's architecture, and those response patterns are collected as evidence for assessment, but it is not necessarily feasible to plan for every possible response in a complex game.

To understand the potential of this third paradigm, it is worth considering what a work process represents. Newell and Simon (1972) described a work process arrangement as three parts of a problem: givens, goals, and operations. The *givens* are the information and resources provided to the student. The *goals* are the desired end state of the assessment, which may be described in instructions to the student. The *operations* are the work processes in which the student engages in their attempt to use the givens to get to the goals. In the 15 Puzzle example, the givens are the state of the puzzle board at each move, the goal is to put the numbered tiles on order, and the operations consist of both the observed movement of the tiles as well as the unobservable decision-making processes and strategizing that the player conducts before moving a tile. The entire work process consists of taking the givens, performing operations, and arriving at a goal state.

Gameplay certainly embodies the Newell and Simon's (1972) definition of work processes, and definitions of gameplay echo their assertion that work processes consist of givens, operations, and goals. For example, Suits (2005) defined gameplay as:

... [An] attempt to achieve a specific state of affairs (prelusory goal), using only means permitted by rules (lusory means), where the rules prohibit use of more efficient in favour of less efficient means (constitutive rules), and where the rules are accepted just because they make possible such activity (lusory attitude). (p.54-55)

What Suits' definition reveals about gameplay is that players accept the imposition of game rules which make it more difficult to engage in certain work processes, but the assumption is that the added challenge and experience of the game rules is entertaining and engaging. For example, if the goal of the 15 Puzzle is to put the tiles in order, the most efficient way to do this is to crack the case and arrange the tiles by hand, but players accept the rules of the game that limit them to moving only one tile at a time into the one available empty space.

But why do researchers care about work processes? Why is it valuable to identify how students take the givens and perform operations to reach a goal state? While many traditional assessments use the goal state (e.g., correct or incorrect) for assessment evidence, there is one common assessment use case for which the work process itself is valuable evidence: differentiating between novice and expert work processes. This application manifests frequently in literature relating to complex assessment designs, where researchers are either trying to discriminate between groups of students (e.g., Bruer, 1993; Stevens et al., 2012; Nash & Shaffer, 2011), or they are trying formatively

to help a novice student become an expert (e.g., Kerr, Chung, & Iseli, 2011; Stevens & Casillas, 2006).

In the third GBA paradigm described above, the GBA provides an environment with some rules and constraints that govern the student's actions. If the game environment becomes more complex and open, vast differences may exist in students' work processes—both in terms of operations and goals. These GBAs also open up possibilities for interpreting sub-processes or sequences within a student's work process as he or she performs gameplay operations to reach a goal (Mislevy et al., 2014). Multiplayer environments expand the possibilities to include gameplay where students interact together while working on individual goals or where students collaborate on a single group goal, possibly with competitive aspects of the play. Regardless of the scenario, the focus on work processes in this third GBA paradigm represents an exciting frontier in assessment design, as well as a challenge for process data collection and interpretation, and this GBA paradigm is the focus of this paper.

The Challenge of Interpreting Work Process Data in GBAs

Using work process data streams for assessment evidence is not a new concept. Assessment designers have successfully built complex performance assessments, simulations, and training systems for many years, especially in the certification and licensure fields (e.g., Braun, Bejar, & Williamson, 2006; Levy & Mislevy, 2004; Margolis & Clouser, 2006; Mislevy, 1996). Though these assessments are excellent examples of assessing work processes, they are not games. They are meant to assess and instruct, but they are not designed to be especially fun or engaging, and while complex, these assessments' designers must impose some rigidity to ensure that their measurement

models can accommodate all possible responses. This rigidity cannot exist in a GBA, which means GBA designers must balance the need to be able to plan for the evidence that can be fed into a measurement model while also avoiding a monotonous game experience.

From a game perspective, GBA designers seek to strike a balance between chaos, determinism, and opacity. Chaotic variation in the game experience can make the game seem more realistic, but it undermines standardization of students' work processes, and a game with too much chaos can frustrate students (for example, a 15 Puzzle with a random starting board is fun and challenging, but a 15 Puzzle where the numbers randomly change on tile faces throughout the game would be maddening). Designers may instead focus on deterministic game experiences (similar to the design of simulation assessments), but these may need to be very complex if they are going to be engaging for the students. The designer may also choose to obfuscate certain game mechanics so that the students can make discoveries and learn about the game as they play, which is different from most assessments where the goal is to ensure that students understand exactly how to respond to an item or perform a task (Mislevy et al., 2014).

Chaos, complex deterministic game elements, and opacity create variation in the game experience and the students' work processes that can make it challenging (though certainly not impossible) to interpret students' performance; however, GBA designers are now using data mining techniques to identify patterns in GBA response data while simultaneously loosening the constraints on the assessment design. In order to find patterns in subdimensions of these large, sparse datasets of GBA response data, researchers often turn to *clustering algorithms* to help group and interpret students' GBA

results. Clustering algorithms are processes that use observed similarities or densities in complex data to identify patterns and group similar observations (Berkhin, 2006).

Although the use of clustering algorithms to classify student response patterns may seem straightforward, researchers must still make decisions about which clustering algorithm to use and how to set any required parameters. Results may also be compared across multiple potential algorithms to help guide interpretation.

Data Collection Considerations for Clustering GBA Data

While there are existing studies that illustrate how to select proper clustering algorithms for general data mining work (e.g., Houle, Kriegel, Kröger, Schubert, & Zimek, 2010), there are additional considerations for researchers who are working with GBA data because the data must be linked by a theory to the *construct* of interest. A construct is a hypothetical concept that the test developers define based on theory, existing literature, and prior empirical work. A construct may be a skill (i.e., the capacity to do something under certain conditions), proficiency (i.e., competency within a domain), or knowledge (i.e., understanding of a subject) (Mislevy, Behrens, Dicerbo, & Levy, 2012). Assessment researchers must propose how a construct relates to a domain of observable behaviors and then design an assessment instrument to record those behaviors in a way to make an inference about the students in the context of the construct. As a simple example, a researcher will propose that the construct of math ability is related to correct answers on math questions, which one can observe directly. For GBAs, researchers must consider how gameplay and what aspects of gameplay relate to the construct of interest.

The researcher's ability to use observable gameplay in a GBA to make inferences about a construct is dependent on how well the GBA elicits behavior related to the construct and collects relevant data. The researcher must have data at an appropriate level of granularity and organization to be able to identify relevant response patterns, and they must understand all of the variables in how the data are created and collected.

Researchers interested in work processes may typically look at variables related to the architecture of the game (e.g., tasks or activities in which the student engaged, state of the game when an action occurred), but some analysis may extend to statistical variables as well (e.g., scores, normative comparisons with other students). Unlike inferences based solely on item responses or final work products, response patterns from process data (like gameplay) can include data representing actions, sequences, context, speed, timing—whole vectors of data representing multiple interacting or overlapping processes, only some of which may be of interest. For example, GBA data may have subsets of data that represent student response processes that are happening simultaneously, at different rates, or in different orders. If researchers want to have a chance at being able to untangle and identify these individual processes for analysis, they must understand how the GBA design impacts the collection and interpretation of gameplay data.

Mislevy et al. (2014) discussed different types of architectural game elements and provided insight on how these elements relate to how evidence is collected and interpreted from students' work processes or gameplay. Mislevy et al. observed that the fundamental elements of a game are the rules, where a reaction from the game is triggered when the student does an action at a given time and under certain conditions. There are several types of architectural elements in the game besides rules, but there are

two elements that relate to the rules and to how evidence is collected about the students' work processes: connections and state.

Connections define the relationships between objects and attributes in game design, which includes design elements like the rules of the game. Connections can also serve to define evidence from work processes in cases where sequences or timings matter. As an example, Mislevy et al. (2014) described a GBA where the sequence of students' actions was an important factor in making an inference about how well the student understood and planned for the consequences of their decisions in the game because the rules of the game stated that a set of actions had to be completed in a certain order to achieve the desired outcome. In the 15 Puzzle example, connections can represent which tiles can be moved at a given point in time, and which direction they can move in. Connections also represent the order in which the tiles must be placed in order for the puzzle to be solved (e.g., tile 5 must appear before tile 6).

Game state is another potential source of evidence for interpreting students' work processes. Game state defines what actions the student can take and the possible consequences to those actions, and the game state is updated by rules in the game. Game state may influence the student's actions as well. For example, a GBA with high autonomy may mean that some students do not get to explore the entire game environment—it is just too vast and complex—and if certain actions or processes are only possible under some game states, then it may be that those students never have an opportunity to demonstrate those processes. Mislevy et al. (2014) gave an example of a video game where the avatar's possible movements at a given moment in time depend on whether or not she is walking on land or swimming in water. In the 15 Puzzle example,

game state defines which tiles can be moved, but it also defines the most efficient solution. Recall that Brüngger et al. (1999) solved difficult 15 Puzzle game states with 80 moves, but a puzzle where only one tile was out of place can be solved instantly. While game state can force or suppress certain actions, it can also be the source of context for applying a rule. For example, in sports, the team with the highest score only wins the game (application of a rule) when the timer runs out (a game state). The same is true in many simulation assessments, where complex logical scoring trees provide the same rule-based structures (e.g., Margolis & Clauser, 2006).

Understanding how actions, sequences (connections), and game state may impact our understanding of a student's work process is an important step toward deciding how to collect and analyze GBA data. Different analyses will be appropriate for different types of data and inferences, but for researchers seeking to identify and then interpret patterns of behavior in data, some researchers suggest using data mining techniques like clustering algorithms (e.g., Kerr, Chung, & Iseli, 2011; Romero, Gonzalez, Ventura, del Jesus, & Herrera, 2009). Clustering algorithms are exploratory processes that group cases together based on observed similarities in the data (Berkhin, 2006). For researchers who are using clustering algorithms to data mine from GBAs, the choice of variables to analyze can impact the interpretations of the clusters, so it is helpful to understand how the inclusion or exclusion of sequence and game state data impacts the identification of clusters representing different work processes. Sequence and game state data may or may not be relevant to interpretation, but including these data usually increases the size of the dataset, leading to a clustering problem for high-dimensional, sparse, and often nominal data.

Study Purpose

To date, there has been excellent research into the design of GBAs (e.g., Kerr & Chung, 2012; Mislevy et al., 2014; Stevens & Casillas; 2006), and there is extensive guidance on the limitations and proper validation of clustering algorithm results (e.g., Kriegel, Kröger, & Zimek, 2009; Halkidi, Batistakis, & Vazirgiannis, 2001; Xu, Recker, Qi, Flann, & Ye, 2013). Unfortunately, little guidance is provided for the specific application of clustering algorithms to work process data (i.e., gameplay data) in GBAs. The result has been research papers that do little to explain the rationale for using a given clustering algorithm, and do less still to validate results with more than expert judgment. GBA researchers have made use of clustering algorithms and produced meaningful clusters of process data that can be interpreted, but as the field expands, GBA researchers' efforts will be more focused and defensible if they understand how GBA design affects gameplay and how the resulting work process data structure may affect the utility of different clustering algorithms. Furthermore, although there is guidance on designing GBAs and creating a valid assessment (e.g., Mislevy et al., 2014), there is not yet a tutorial providing instructions on how to validate and report clustering algorithm results for those GBA researchers who choose to use these tools. There is an opportunity to merge lessons and examples from both GBA researchers and other clustering algorithm research to create a unified set of instructions.

Given GBA design considerations and their potential impact on the interpretation of students' work processes, this study accomplishes two goals:

1. Investigate how well different clustering algorithms identify clusters of response patterns in GBA response data in order to inform researchers about

clustering algorithm limitations and impacts of varying conditions in datasets that mimic GBA scenarios.

2. Provide researchers with a tutorial for how to select clustering algorithms, compare results, and validate a clustering solution in the context of GBA development to ensure that inferences made about students on the basis of a clustering algorithm have been rigorously investigated and can support a validity argument.

Three clustering algorithms are compared in this study: *k*-means, self-organizing maps (SOM), and robust clustering using links (ROCK). The *k*-means algorithm was selected due to its general popularity and ease of use for education research. The SOM algorithm was selected because it has been used to analyze work process data in GBAs and simulation assessments (Stevens & Casillas, 2006; Stevens et al., 2012), and ROCK was chosen because it is specifically designed for binary data, which is a common way of tagging work process data. Chapter 3 discusses a small-scale feasibility simulation examining differences in these algorithms' clustering solutions and their performance with GBA data. A large-scale tutorial example study using empirical GBA data and a simulation study are outlined in Chapter 4. Chapter 5 examines how the three clustering algorithms perform under the varying simulated GBA conditions discussed in Chapter 4. Chapter 6 offers a tutorial for how cluster validation can be conducted with competing algorithms, by way of the tutorial example study discussed in Chapter 4.

It is important to acknowledge that this study does not generalize to all possible game designs or data conditions. Game designs and gameplay are enormously diverse, and it would be impossible to account for every possible scenario. Similarly, this study is

not expected to generalize to every possible clustering algorithm. Instead, this study serves to demonstrate that choices in clustering algorithms can impact the interpretability of solutions by comparing three specific algorithms. This study is intended to provide a framework for how to think about the use of clustering algorithms in GBA analysis rather than to yield claims or recommendations for the entire universe of games. For GBA researchers selecting clustering algorithms, the results illustrate how GBA design may create conditions that impact clustering algorithm results, as well as how to test, validate, and report clustering results in a GBA study.

Chapter 2: Game-Based Assessment

The intersection of games and assessments can be hazy, but there are many assessments that are clearly not games, and many games that would be difficult to use for assessment. This section examines the similarities and differences between the two mediums and seeks to identify the characteristics that make an instrument a GBA, as well as the considerations that arise in GBA scoring and evaluation through the use of clustering algorithms.

The criterion for when a game becomes an assessment is generally straightforward. A game can be used as an assessment instrument if the game is used to gather evidence from students' behavior, and if it yields a quantitative value from which an inference about the students can be made (Crocker & Algina, 2008). There is less clarity around when an assessment becomes a game. Mislevy et al. (2014) provided a detailed overview of game design elements, but many of these concepts could just as easily describe an assessment that would not be considered a game, such as a simulation assessment, a performance assessment, or an intelligent tutoring system. The following sections provide an overview of Mislevy et al.'s (2014) design elements and seek to pinpoint the design choices that differentiate between an assessment and a GBA.

Architectural Game Design Elements

Mislevy et al. (2014) defined four types of architectural elements of GBAs: rules, connections, game state, and mechanics. These elements are used to engineer the functionality of the game as an instrument, and each element has its own aspects and facets that apply to different areas of game design. This section explains what these

elements mean in game design and demonstrates how the same concepts can be found in some non-game assessments.

Rules. Rules define how the game reacts to students' behavior given the game state and moment in time that the behavior occurred. This is true for any game. In soccer, the difference between kicking a goal and kicking the goal that wins the game is based completely on context (cumulative frequency of goals per team, time on the game clock); however, the same definition of rules exists in many non-game assessments. For example, a computer-adaptive test (CAT) reacts to a student's response to an item by selecting the next item to present and by updating the estimate of the student's location on a latent scale. These two reactions will depend on the state of the assessment (the student's previously estimated location) and the moment in time on the assessment (how many items the student has already answered) (de Ayala, 2009).

Connections. Connections define the relationships between game elements, and Mislevy et al. (2014) observed that a GBA may have two sets of connections: one that maps out relationships needed for gameplay, and one that maps out relationships for assessment. A game rule can be seen as a special type of connection that defines relationships between game reactions, students' behaviors, and game state (Mislevy et al., 2014). Connections are the basis for the game's reaction to sequential behavior as well (e.g., solving a problem with multiple steps in a certain order), but this again does not differentiate between general assessments and GBAs. A simple example is the use of sequential relationships in common item formats, such as drag-and-drop items.

Game state. Game state defines what a student can do in a game at a specific moment in time, and game state can be an input for evaluating a rule in a game (Mislevy

et al., 2014). Game state can help to maintain a logical game environment for the user (e.g., when underwater, the student's avatar can only swim, not run or jump), or it may be used as a component in the game narrative. A typical example is levels—students often cannot advance to higher levels until they have completed initial levels and the game state is updated. Although the term “game state” is used here, the state of an instrument and the context of a student's action are also relevant in assessments. The rules described previously for CATs are clearly dependent on the state of the assessment at the time the rule is executed.

Mechanics. The final type of game element defined by Mislevy et al. (2014) is a game mechanic. Mechanics define elements of the game that are used by the student during gameplay, and they allow the student to make actions that are registered and reacted to by the game. Mechanics are selected so that they guide actions that are relevant to the target domain.

These mechanic devices exist in other assessment types—not just GBAs. Simulation assessments and intelligent tutoring systems require these types of mechanics to register a student's actions as evidence for the assessment. Even simple multiple choice items have a mechanic—specifically, a tool that allows a student to select a response or fill in a circle corresponding to their response. Thus, none of the architectural elements of games seem to differentiate games from assessments—all of these design elements are easy to find in traditional assessments—so if a differentiation does exist, it must lie in the experiential game design elements, discussed in the next section.

Nevertheless, the architectural elements of a game are important to consider when researchers use clustering algorithms to analyze work process data from gameplay.

Understanding the architectural elements can help researchers make informed decisions about which variables to analyze and which cases to exclude. Understanding these elements may also help with interpretation of the clusters, especially if a game is designed to guide students toward a common response pattern, as may be the case in instructional games. Most importantly, the architectural elements will help researchers plan for the data distributions they can expect to observe, which in turn will help them select an appropriate clustering algorithm. For example, if an algorithm assumes the data are spherically distributed, but activation of one connection blocks the student from accessing certain parts of the game, thus creating dependence between variables and resulting in a non-spherical distribution, then researchers can think about restructuring their data or using a different algorithm with more relaxed assumptions. For a very simple example, consider a game where a section of the game becomes unreachable after a short period of time—perhaps a bridge is collapsing or a door is closing—and one must race to get through it. This change in connections keeps slower players from participating in the game section that has been cut off, which could put a ceiling on their achievement in the game, although they may still play in other areas of the game environment. Figure 2 below shows how this hypothetical relation between speed of play and game achievement might look in this simple example.

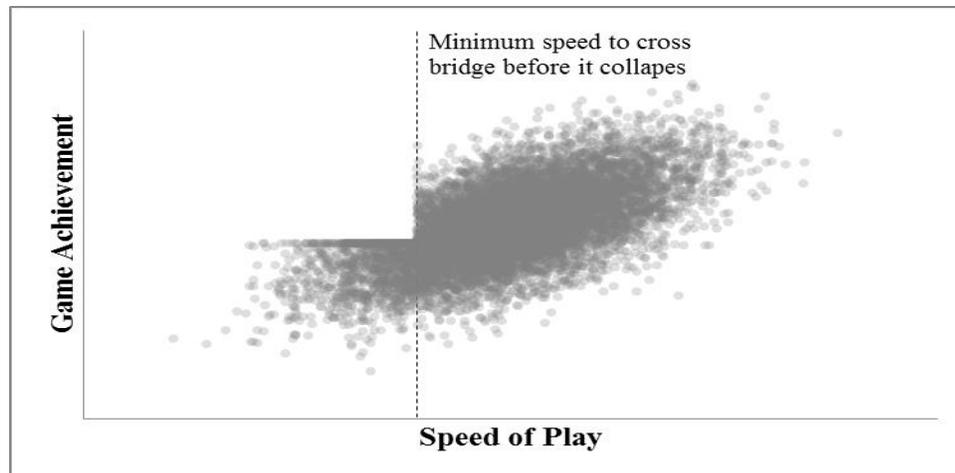


Figure 2. Hypothetical example of a time-based connection element that students can only experience if they play quick enough (e.g., get across a bridge before it collapses). This connection element puts a ceiling on slower players' possible achievement in the game and impacts the distribution of achievement scores.

Experiential Game Design Elements

In addition to architectural elements, there are three types of experiential game elements: autonomy, competence, and relatedness (Mislevy et al., 2014). These elements define how the student experiences and interacts with the game, and they are intended to elicit emotions, learning, engagement, and cognitive reactions from the student. These elements may contain the characteristics of the instrument that distinguish between something that is only an assessment versus something that is a GBA; i.e., they may help answer the question, when does an assessment become fun?

Autonomy. Autonomy refers to how much freedom a student has in his or her decisions about which actions to take in a game (Mislevy et al., 2014). Higher autonomy adds complexity which may make the game more engaging, but it also may expand the number of possible response patterns that can be observed, creating challenges for interpretation and analysis.

There are four aspects of game design that impact autonomy: authorship, chaos, determinism, and opacity (Mislevy et al., 2014). *Authorship* refers to the constraints on

how a student interacts with a game, and it exists on a spectrum. If all students have the same identical experience, then the authorship is very restricted. If the game simply provides an environment and students can make up their own experience within that environment, then authorship is very open. The second autonomy factor, *chaos*, refers to elements in the game that are allowed to vary randomly, which impacts the number of decisions and actions that a student could potentially make. Some chaotic elements make games feel more realistic, but too many chaotic elements may overwhelm students. Conversely, *deterministic* elements are those that are created by the game designer and follow predetermined rules, but a significant amount of work is needed to create complex, challenging deterministic elements. The final factor in autonomy is *opacity*, which refers to how much of the underlying mechanics of the game are initially exposed to the user. A fully transparent game makes it clear to students how the game is working and what they are supposed to do, whereas a completely opaque game is nearly impossible to figure out and may be frustrating. Some opacity can be desirable for creating an interesting game experience where students can discover things within the game and learn about the game as they play.

If autonomy is a distinguishing feature between GBAs and other assessment types, it is a subtle one. High varying degrees of authorship exist in both GBAs and other assessments. Chaos is less common in assessment instruments because chaos can make it difficult to compare responses between students (Mislevy et al., 2014), but it is not unheard of. While many assessment elements are deterministic, some formative assessments use chaotic elements like item templates that randomly generate values for predefined variables. For example, an assessment may have addition questions that

randomly select numbers (under some constraints) for the students to sum. Conversely, there are GBAs with very little chaos, game presentation is completely deterministic, and the same elements are applied for all students (e.g., Kerr et al., 2011). The final aspect of autonomy, opacity, also exists in simulation assessments. For example, there are assessments where the student does not initially know the complete scope of the task, and only through a process of discovery and learning through interaction with the assessment can the student arrive at a response (e.g., Margolis & Clauser, 2006; Mislavy, 1996).

Although autonomy does not seem to be a concept that is specific to GBAs, there is an architectural element to consider when evaluating the authorship of an instrument's design: rules. A designer can create an interface that supports many inputs from the students, thus creating high authorship, but if the instrument does not have rules to react to the student's actions, then the extensive autonomy is not very engaging for the student, and it serves no purpose for assessment or gameplay. Autonomy in the absence of a supporting rules structure can flood a dataset with irrelevant response data, making it difficult to analyze or interpret response patterns with a clustering algorithm. For this reason, any researcher using a clustering algorithm with a high autonomy instrument like a complex GBA should be prepared for the presence of data that are not relevant to the construct or for distinguishing between work processes, and this scenario should also be addressed in any comparison of the performance of different clustering algorithms.

Competence. Competence refers to the student's ability to apply and demonstrate their skills or knowledge within a game. This means that the game should have goals or challenges that the student is trying to achieve. The game should also have some degree of complexity (which relates to opacity) where the student has the opportunity to learn

about and engage with the architectural elements of the game—some of which may not be immediately transparent. As part of competence, students must receive feedback, which can be as simple as earning points or completing levels (Mislevy et al., 2014). The final competence factor is adaptability, which means that the game should be designed to be challenging and engaging for all players, regardless of ability (Gee, 2007).

Like autonomy elements, competence elements also create challenges for scoring and interpretation, especially with respect to adaptability. A well-designed game can provide experiences that are challenging to a variety of skill levels, but this can create issues for comparability. Depending on the game design, there may be very different game experiences for students with different abilities, increasing the sparseness of the dataset.

There are many well-designed assessments that exhibit the elements of competence, and it is not unique to GBAs. Goals of traditional assessments are often communicated to students in the form of performance level descriptors, rubrics, test blueprints, and other test documents that outline the purpose of the assessment, how performance is evaluated, and how that performance will be measured. For students, this may boil down to a goal of answering as many questions correctly as possible, but that is nevertheless a goal. The ability to learn about architectural elements in an assessment is also common, and some assessments are designed to have students work through different steps of discovery before arriving at a solution (e.g., Margolis & Clauser, 2006; Mislevy, 1996). Feedback and adaptability are also common to many assessment designs.

Competence elements are typically outcomes with feedback to the student, and the game designer expects most students to interact with each competence element

eventually; i.e., all students will beat a level, earn points, win the game, etc. if given enough time. In this sense, competence elements may not be an important consideration for including in a clustering algorithm that is seeking to describe work processes as opposed to outcomes; however, competence elements can serve to help interpret or validate clusters. Consider a case where there is one cluster representing a common work process for achieving a competence element goal but also a second cluster in which students also frequently—and perhaps unexpectedly—achieved the same goal. This would be evidence of an alternative solution to the game.

Relatedness. If there is an aspect of instrument design that best highlights the differences between an assessment and a GBA, it is relatedness. Relatedness refers to elements that help the student to identify with or empathize with the game. These can include narratives, settings, or even collaboration with other players or simulated characters. Relatedness often increases students' engagement and motivation to play and succeed in the game. SRI International, the organization responsible for independently studying the validity and effectiveness of GlassLab's games, observed that GBAs are being designed as a method for overcoming students' disengagement (SRI International, 2015).

It may be that relatedness causes an assessment to make the leap to GBA when the design elements are added with the intent to help the student identify with the instrument without any theoretical connection to the measured construct or the task model for the assessment. SRI International (2015) made a similar observation, noting that GBAs stand out because they are activities in which both students are motivated to participate. This sentiment was echoed by a teacher profiled by Schwartz (2014b), who

argued that as instruments map more closely and directly to educational standards, the less “game-like” it becomes. To put it simply, in GBAs, something has been added to the design of the instrument with the sole purpose of making it more fun.

D’Angelo et al. (2014) made the same distinction in an effort to distinguish between games and simulations. As part of their definition, they stipulated that simulations represented real-world or hypothetical scenarios; suggesting that a game might use scenarios that would be fantastical or even impossible in the real world as a form of relatedness elements. They added that games could include “non-learning-based goals” (i.e., relatedness elements) like levels, points, or other reward systems; whereas simulations focus the student’s experience onto the specific process or system that is being simulated and assessed. D’Angelo et al.’s distinctions between games and simulations echo Suits’ (2005) definition of gameplay, in which he noted that players accept the imposition of game rules which make it more difficult to achieve a work process goal than it would be under non-game conditions (e.g., it is easier to put a ball in a net if one does not have to play soccer to do so).

Relatedness does not need complex story lines and interesting characters however. Simple games also have design elements that impact relatedness. As an example, Number Jumble presents students with multiplication tasks. The student can earn points by solving the problems correctly, and the student can earn bonuses by answering quickly and accurately while the bonus clock ticks down (BrainPOP, 2015). These simple additions support notions of rewards, suspense, and urgency without having anything to do with the student’s ability to multiply numbers, assuming one does not evaluate a student’s multiplication aptitude on the basis of speededness.

From a scoring and interpretation standpoint, relatedness elements present one of the biggest challenges for the use of clustering algorithms in GBAs: irrelevant data. Relatedness elements are present to increase empathy, not to elicit evidence about the construct, yet students interact with these elements and data are collected. Even if a relatedness element does not result in collected data (e.g., a soundtrack for the game), there is no guarantee that it is not having an impact on the student's response patterns for those variables which are relevant to the construct. These relatedness elements will either produce their own variables which the researcher must choose to include or exclude, or they may influence the response patterns. From a cluster interpretation standpoint, it may be impossible to tell if a pattern of responses related to the work process that differentiates novices from experts or if it was just the student trying something for fun because of a well-designed relatedness element.

GBAs: Assessments with Construct-Irrelevant Relatedness Elements

The previous sections showed that all of the elements used in game design are found in regular, non-game assessments, but if an assessment is considered a GBA, it is due to the inclusion of construct-irrelevant relatedness elements: design elements that exist only to make the assessment fun. For the purposes of this discussion, we will consider an assessment instrument to be a GBA if a designer adds relatedness design elements to the assessment which have no theoretical connection to the task model and are intended only to help the student empathize with the game. This may include elements like narratives, navigation, animations, backgrounds, music, characters, avatars, timers, points, or other game elements. The addition of relatedness elements is the unique

differentiator between GBAs and other complex assessments or performance assessments.

These non-assessment relatedness elements are both a strength and a weakness from an assessment standpoint. As a strength, they make the instrument more enjoyable for students. This encourages increased engagement, thus improving the instrument's ability to be used as a learning tool as well as a tool for ongoing evidence collection for assessment. However, these relatedness elements also add superfluous elements to the task model. In traditional assessments, superfluous elements are surgically stripped from items to help focus the item on its content area and to avoid unnecessary complexity for students. Adding extraneous wording or stimuli to an item stem is often called "window dressing," and item writers are instructed to avoid this practice, because it can negatively impact the reliability and validity of an item (Nitko & Brookhart, 2007; Haladyna & Downing, 1989). The same concerns exist when adding construct-irrelevant relatedness elements in a GBA design. Mislevy et al. (2014) warned that relatedness elements can increase construct-irrelevant variance. In electronic GBAs, relatedness elements are also risky from an evidence collection perspective because they may produce irrelevant data that muddy the water for researchers seeking to interpret students' gameplay. This is why analysis of GBA data can be challenging for psychometricians who are interested only in identifying construct-relevant response patterns—it may not always be clear which data are relevant.

The Use of GBAs in Education

The use of games is already well established in U.S. classrooms. In a national survey of a stratified random sample of 648 K-8 teachers selected from a commercial

pool (a self-selected population of survey respondents maintained by VeraQuest), Takeuchi and Vaala (2014) found that 74% of the teachers surveyed use digital games, and 55% reported doing so on a weekly basis. In a different survey of 488 K-12 teachers, Fishman, Riconscente, Snider, Tsai, and Plass (2014) found similar results: 57% of teachers used games weekly, although they observed that that number tended to be higher for teachers working with younger students.

With games already embedded in instruction, it is easy to see how the next logical step would be GBA, but Takeuchi and Vaala (2014) found that assessment was a minority use case for teachers who use digital games. Of the 648 K-8 teachers surveyed, only 29% reported using games for formative assessment of content related to educational standards, and only 17% reported using games for summative assessment as well. Furthermore, there is no distinction in Takeuchi and Vaala's survey between the three aforementioned game paradigms described by Mislevy et al. (2014), so it may be that some of these teachers are using games that fall into the second paradigm, where traditional assessment tasks are woven into a game instrument. Of the 29% of teachers who said that they used games for assessments, only 31% of these (9% of the total sample) reported that they assess students' abilities by observing or discussing how the students played a game, implying that the remaining teachers only used the outcome of the game for assessment.

Although less than one-third of K-8 teachers reported using digital games for even formative assessment, three quarters of teachers reported using digital games on a weekly basis, suggesting that teachers at least perceive benefits of gameplay for learning (Takeuchi & Vaala, 2014), but whether those perceptions reflect reality has not been

rigorously studied. Granic, Lobel, and Engels (2014) argued that digital games may have cognitive, motivational, emotional, and social benefits for students, and they called for additional research to study these effect areas. Nevertheless, the potential benefits suggested by Granic et al. make sense from a game design perspective, because unlike other assessment instruments, games are designed to engage players emotionally or intellectually (Shapiro, Teknibaş, Schwartz, & Darvasi, 2014). Given the engagement design goals of games, Granic et al.'s (2014) proposed benefits of gameplay for learning are certainly plausible, which suggests that the same benefits may be realized for educators who use GBAs as assessments for learning.

Games (specifically GBAs) also have the added motivational benefit of not feeling like assessments. Many GBAs are examples of assessments as learning, meaning that they serve as learning devices as well as assessment instruments. By weaving the assessment features into the gameplay, students can learn continuously without having to pause to be evaluated. Students may also not feel as much anxiety about the assessment if it is embedded in their learning activities. This invisible integration of assessment and learning is sometimes called *stealth assessment* (Shute & Ke, 2012).

From an emotional standpoint, Granic, Lobel, and Engels (2014) note that there is no clear theory of how positive emotional states benefit people beyond the fact that they just feel good. Nevertheless, negative emotions like test anxiety are something that assessment designers seek to minimize, as they can introduce construct-irrelevant variance in responses (Mandler & Sarason, 1952). Wrapping an assessment in a game that is associated with positive emotional states can improve students' perception of

assessment and perhaps even improve measurement by minimizing anxiety. A review regarding the effect of anxiety on test performance can be found in Powers (1999).

Regardless of the perceived student benefits of using games, one thing is certain: GBAs are already taking the place of traditional assessments as measurement instruments; so it is worth asking whether there are measurement benefits to using GBAs. Admittedly, some of these instruments are little more than gamified versions of traditional assessments (e.g., keep solving multiplication problems to keep your race car going fast), but recall in Chapter 1 the discussion of the shift in GBA design to a paradigm where assessment is occurs inside the game architecture (Mislevy et al., 2014). Using this paradigm of GBAs, the assessment domain is now represented by the work processes or gameplay of the students (Shute, 2011).

Using work processes as domains creates an opportunity to measure constructs that may be difficult to represent in traditional assessments. For example, Shute and Wang (2016) explained that digital learning environments like digital GBAs are ideal tools for creating assessments that require the application of multiple competencies as part of the work process. They added that well-designed games are challenging (without being too difficult), provide feedback during gameplay, and reinforce students' sense of control over the task—all of which, they argued, make GBAs useful for teaching and assessing hard-to-measure constructs.

What are hard-to-measure constructs? Shute and Wang (2016) list creativity, problem solving, persistence, systems-thinking, gaming-the-system, and design thinking as examples. One might also add collaboration to this list, given the cooperative game designs of some GBAs already used in classrooms (e.g., Eseryel, Ge, Ifenthaler, & Law,

2011; Nash & Shaffer, 2011). Shute and Wang (2016) chose their list of hard-to-measure constructs from a set of skills that are currently seen as valuable competencies for students who are entering college or the workforce, but the list constructs that can be represented with work process domains could certainly be expanded.

Challenges of Using Games for Assessment

The previous section shows that educators and researchers perceive potential learning benefits from games, and the range of architectural and experiential game elements discussed above point towards new frontiers for the types of task models that can be used in assessments; however, the structure of games brings new challenges for those who seek to use games as measurement instruments. Researchers must consider how to evaluate students' gameplay in light of construct-irrelevant relatedness elements, unrecorded student actions, response dependencies on sequence, and response dependencies on game state.

Construct-irrelevant relatedness elements in games can cause work process response patterns to be a mixture of both construct-relevant actions and actions related to game skills. In complex gameplay, it can be difficult to determine when a student is exhibiting behavior related to the construct versus when the student's performance is dependent on his or her ability to play the game. Sequence and game state may also be relevant factors in interpreting a student's response.

For instance, Shute (2014) explained that the game's levels can become more difficult, not because the educational content has changed, but because the obstacles and goals in the game design become more difficult; i.e., the difficulty of the task is a construct-irrelevant relatedness element. As an example, Shute described a GBA where

students use knowledge of physics to create a system of simple machines (onscreen) to move a ball to a goal. The required knowledge of physics does not change throughout the game, but the levels still become more difficult because the designers add more obstacles around which the ball must be guided. Almond, Kim, Velasquez, and Shute (2014) expanded on this idea, noting that there is a difference between game difficulty due to complexity of the task and psychometric difficulty, which requires a higher level of the measured construct in order to succeed at the task. Almond et al. noted that these two concepts of difficulty are often but not always correlated. Keeping the distinction between task complexity and psychometric difficulty in mind, it becomes clear that a score or outcome in a GBA is only useful if the evidence for the work process is not drowned out by the enjoyable difficulties introduced as challenges and obstacles in the game's design.

Game design is not the only influence that might impact a researcher's deductions when analyzing process data from GBAs. One must also recognize that a game may not be an enjoyable or accessible format for everyone. While games are touted as a way to boost engagement, not all games will be engaging for all students. Some students may not ever figure out how to play the game, just as some students do not understand how to respond to certain item types, which then results in response bias (Schmeiser & Welch, 2006).

Students may also engage in behavior that has nothing to do with the construct of interest, yet also has little to do with scripted gameplay either. For example, *open-world* games are designed with such high autonomy and expansive game environments that students can spend time simply exploring ("free-roaming") the game space. This behavior

reflects students learning about the game design, testing its limits, and seeing what is possible in the rules and mechanisms that define the game (Muncy, 2015). In open-world games, game designers are intentionally encouraging this behavior by providing complexity and details that reward free-roaming, but this exploratory behavior is not limited to complex games. Students may experiment with the limitations of any task, testing the range of acceptable behaviors and responses and discovering the mechanics that underlay the assignment. From the researcher's perspective, the free-roaming work process data may be unintelligible.

To further complicate analysis of work process data, it is possible that some student actions in a work process may not even be recorded. Data are not collected for a student's work process unless that process includes interaction with the game interface. In their development of a GBA, Hoffman, John, and Makany (2014) noted much of the gameplay involved students observing and mentally interpreting the onscreen effects of an action in the game. The students then used those observations and interpretations to choose their next move onscreen, though no data were collected by the game about the cognitive decision-making processes that led up to that move.

Given the difficulties of evaluation resulting from construct-irrelevant relatedness elements, unrecorded student actions, response dependencies on sequence, and response dependencies on game state, GBA designers must plan evidence collection strategies early in the game design process using assessment development frameworks like evidence-centered design.

Designing GBAs with Evidence-Centered Design

An established framework for designing, delivering, and scoring assessments is *evidence-centered design* (ECD). ECD applies a broad range of assessment design resources (e.g., subject matter knowledge, software design, psychometrics, pedagogical knowledge) to the inferences that are drawn from the assessment scores, thus enabling interdisciplinary teams to use a common set of guiding concepts to reason through the design and implementation of an assessment (Mislevy, 2011). ECD has been used to develop many assessments across multiple disciplines, measurement models, and use cases, but Mislevy et al. (2014) specifically recommended the use of ECD for the development of GBAs, and there are several examples of GBAs designed with ECD (e.g., Hoffman et al., 2014; Stevens & Casillas, 2006).

ECD provides guidance on how to create an operational assessment for a latent construct through multiple, iterative development steps. Each step helps to build an assessment argument to support the validity of the assessment's inferences (Mislevy, 2011). In this sense, ECD is not just a guide for developing assessments, but also a conceptual framework linking the structure of an assessment to the inferences about the construct and intended applications of those inferences. The ECD component that relates to the design of the assessment is the *conceptual assessment framework* (CAF) (e.g., Mislevy et al., 2012). The CAF divides the operational elements of an assessment's design into three parts: the *student model*, the *task model*, and the *evidence model*, which are discussed below.

The student model in the CAF specifies the constructs and construct relationships that are being measured to describe the student's knowledge or abilities (Mislevy et al.,

2012). The goal of the assessment is to make inferences about a construct based on observed behavior in the assessment (Crocker & Algina, 2008). Assessments may measure a single construct, multiple constructs, or a hierarchy of constructs and sub-constructs at varying levels of granularity. Note that a student model does not necessarily describe all of the constructs that may be applied by the student when completing the assessment; it only describes those constructs for which we seek to make an inference.

The task model defines the assumptions and specifications for what the student can do in the assessment (the task) and the features of the environment in which the task takes place. (Mislevy et al., 2012). The task model also defines the specifications for any work products the student may be expected to create during the assessment. Even in high-autonomy assessments where there may be substantial variability in students' responses and work processes, the task model can act as an anchor for interpreting results. Regardless of the process a student executes, one can anticipate that most students are engaging in a process that relates to how they perceive and interpret the task model. This means that researchers may not know what the processes are, but there is a reasonable expectation that researchers know what the students are trying to accomplish because the researchers designed the task model.

The evidence model connects the task models and the student model in the CAF. It has two parts: the *evaluation / evidence identification component* and the *measurement model / evidence accumulation component*. The evaluation component defines how evidence is identified and collected in the work products, and the measurement model component defines how to aggregate those data to make inferences about the student

(Mislevy et al., 2012). Figure 3 illustrates how the evidence model relates to the task model and student model.

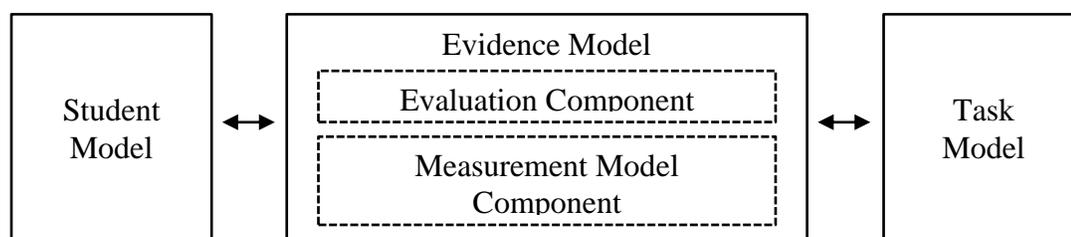


Figure 3. Relationship of the student model, evidence model, and task model in the CAF. The evidence model contains two parts: the evaluation component and the measurement model component.

As discussed previously, identifying evidence and creating a measurement model can be especially challenging in GBAs due to construct-irrelevant relatedness elements, unrecorded student actions, response dependencies on sequence, and response dependencies on game state. In a multiple choice item, the evaluation component simply depends on whether or not the student selected the correct choice. In a GBA, the evaluation component must make sense of large volumes of data that may come across as raw, granular pieces of information, such as keystrokes, time stamps, or text entry. The evidence model may need to define how to reduce the data to a set of scorable data points (a process called “tagging”), identify tagged data to be used for scoring the student’s final output (goal state), and possibly identify tagged data to be used for scoring the student’s work process. Once relevant evidence data have been identified, the GBA designers must then design a measurement model that can combine the evidence into an actionable inference about the student.

Measurement Models: Principled versus Empirical Approaches in GBAs

There are a variety of strategies to analyze response data from GBAs. This section discusses different methods that have been used for the measurement model and the

student model, as described by the ECD framework. Bejar and Braun (1994) identified two strategies for analyzing complex response data: a principled approach and an empirical approach. Simply put, one can think of these two approaches as a top-down versus bottom-up approach, where one can either work from a high-level theory to customize an instrument to collect evidence, or one can collect evidence and work up toward a theory that explains the observations, respectively. The following sections discuss these two approaches in further detail.

Principled approaches. With a principled approach, measurement models and student models are defined a priori based on the findings of a domain analysis and subject matter experts' understandings of how a construct will be manifested in the response data. This approach is appealing because the scoring rules are built on theory; however, there is a presumption that the test designers have considered every possible response pattern and how to score it in advance. This is usually not possible for high-autonomy task models, and measurement models and student models (as well as the tasks themselves) often have to be iteratively fine-tuned based on empirical approaches or usability studies of the GBA (Bejar & Braun, 1994).

Many GBAs use a principled approach in their measurement model as a matter of practicality. Rather than collecting every single key stroke of a student and then mining these data to determine which data might relate to the desired inferences, inputted response data (e.g., keystrokes, mouse movements, eye tracking data, time stamps) are often tagged or scored at some granular level. This initial tagging can be done with scoring trees, which are simply logical decision trees that return a value based on the observed result data. For example, if a student executes some action under a specific set

of circumstances, a scoring tree might return a value of 1 (and any other action might return a 0). That scoring tree must be defined by a test designer's understanding and interpretation of the student's performance in the context of the game state, so scoring trees count as a principled approach. Item response theory (IRT) and Bayes nets are other examples of principled approaches used to evaluate GBA data, though in practice these methods rely on a foundational layer of scoring trees to simplify the response data (e.g., Quellmalz, Timms, Silberglitt, & Buckley, 2011; Mislevy, 1996; Shute, Ventura, & Kim, 2013).

Conversely, exploratory approaches to GBA measurement models like clustering algorithms are often associated with empirical approaches for analyzing the complex response data, for which there may be only an inchoate theory about how response patterns will manifest as a result of the underlying construct. It is unlikely that these exploratory approaches would ever be used in a principled measurement approach because these methods are used to identify structures in multidimensional data that were previously unknown to the researcher. If the researcher has a theory for how a construct will manifest as a work process, then the assessment and measurement model can be built with a principled approach to capture and score evidence about the student's work process—there would be no need for exploratory approaches. The following section explains how researchers may use exploratory methods in an empirical approach to analyzing complex data and evaluating student performance when a theory of response patterns does not exist a priori.

Empirical approaches. With an empirical approach, subject matter experts analyze the results data and define a measurement model that will relate the work

process observations to their student model describing the construct. As a simple example, a test designer might use a regression analysis to relate observed game-based assessment data (either raw or tagged) to experts' ratings of the students' performance. That regression model could then be used as the measurement model for future administrations (Bejar & Braun, 1994). Clustering algorithms are also an example of an empirical approach to measurement because one clusters students based on the collected work process data and then makes post hoc inferences about students based on the characteristics of the clusters.

As an illustration, Kerr et al. (2011) took an empirical approach to analyzing GBA data with the use of fuzzy cluster analysis. Test designers found that the clusters corresponded to different solution strategies and common errors or misconceptions that manifested themselves in the response data. One benefit of their game was that different game levels had similar designs, and comparisons could be made between a student's cluster membership across levels.

Stevens and Casillas (2006) used the self-organizing map (SOM) clustering algorithm to analyze the paths students took through a game environment. The game tracked how many times the student visited each menu item in the game, the order of the visits, and the outcome of the game (right or wrong). The algorithm analyzed these paths and grouped the individual students' paths into different clusters. The researchers were able to go back and map the characteristics of these clusters to descriptions of theoretical strategies they expected students to exhibit during gameplay.

How principled and empirical approaches impact GBA design. It has been argued that the ideal GBA design is one where the test designers have tight control over

the assessment mechanisms without restricting the flow of the game (Mislevy et al., 2014). This is certainly the case for assessments designed with a framework such as ECD. Measurement models must be able to process the provided evidence data and yield results that are meaningful in the context of the student model. The test designers build the rules for what counts as evidence, which suggests a principled approach.

With an empirical approach, test designers have to inspect responses that already exist and then make decisions regarding which data count as evidence and how they should be used in a measurement model. For example, both Kerr et al. (2011) and Stevens and Casillas (2006) mined data with clustering algorithms to identify patterns in response data which they then used to classify students. Recall also that Mislevy et al. (2014) used the term “autonomy” to refer to the amount of freedom or variation students have in their work processes. By relying on an empirical approach, test designers do not need to have quite the same control over the assessment mechanism. They can increase the autonomy of the GBA’s task model, allowing for more variation in responses—including unexpected response patterns that may be considered valid. These patterns can then be discovered and evaluated later with an empirical approach like a clustering algorithm.

Even though a measurement model may use an exploratory empirical approach instead of a principled approach, one can still use the concepts and terminology of ECD to describe the development and design of a GBA. The researcher may not have a specific theory about precisely how the construct will manifest as a response pattern in the GBA until after the exploratory analysis, but the researcher still must base the design of the game environment (design elements) and task models on a theory for how the

game will elicit gameplay responses that will be evidence of the construct. The test designers may not know what responses will look like (which is why they may rely on tools like clustering algorithms), but they should have a theory that any response created and recorded within that environment could be used to inform the student model. Even if the measurement model relies on an exploratory approach, the model will presumably be locked down eventually. For example, clusters may be identified with a clustering algorithm, but then those cluster centers will likely remain static for evaluating future students' responses. In this sense, an exploratory measurement model may be seen as another part of the ECD development process for the CAF.

Unfortunately, even with careful design of a GBA environment using an approach like ECD, there is no guarantee that students will provide an interpretable or meaningful response. Students will often engage in response patterns in a high-autonomy environment that are difficult to interpret, even in relatively simple task models (Corrigan et al., 2014). The emergent behavior of students' work processes means that it may not be obvious how to create a statistical model that relates the observed response data back to the construct (Bejar & Braun, 1994; Johnson, 2001).

Exploratory approaches to measurement models may also obfuscate interpretations of students' responses. For example, a researcher clustering student response patterns with a clustering algorithm may find it difficult to understand why a student has been assigned to a given cluster or how to interpret what an entire cluster represents with respect to the construct. For example, Kerr et al. (2011) used fuzzy cluster analysis to classify the strategies and errors that students made in *Save Patch*, but 26.4% of the students' game results across 16 game levels had to be classified as

“unexplained errors.” Stevens and Casillas (2006) noted similar interpretive constraints, observing that there may be no way to understand how the SOM algorithm arrived at a prediction for a response pattern. The following sections explore empirical classification methods in more detail and discuss the perceived benefits of empirical approaches for work process data.

These unclassifiable response patterns underscore the need for GBA researchers to use clustering algorithms or other methods that can handle outlier cases well. GBA data will often be polluted with outlier response patterns (e.g., false starts, unengaged students), and these ideally should not impact the identification of clusters of legitimate responses that relate to the construct. Furthermore, the clustering solution needs to be reasonably reproducible. If cases are not consistently assigned to the same clusters, it may be difficult to trust the interpretations being made about individual students based on their cluster membership.

Nevertheless, accurate classification for all students may not be needed for GBAs in practice. The empirical approaches used by GBA researchers reflect and frame the reasons (discussed in Chapter 1) for why people use GBAs in the first place: to evaluate work processes to differentiate between novices and experts or to help novices become experts. GBAs are necessarily low-stakes, formative assessments, so accuracy at an individual level may not be as important as demonstrating the novice/expert distinction in work processes at an aggregate level.

Clustering Algorithms

The previous section discussed the utility of empirical methods for mining response data to identify assessment evidence when researchers cannot anticipate

precisely how students might respond in a task model, but if researchers cannot anticipate certain responses, it begs the question about whether or not clustering algorithms are even an appropriate tool for a measurement model, even in a low-stakes application.

In their paper detailing the facets of data mining, Romero et al. (2009) explained that one must use a data mining strategy that is appropriate for the type of data one wishes to identify, such as data mining to identify patterns of behaviors. Their explanation is valuable for assessment researchers because it indicates that data mining can indeed be a method for identifying cognitive or behavioral processes (Berkhin, 2006). Kerr et al. (2011) explained that identification of processes within response patterns is typically done with clustering algorithms, which can be considered a type of data mining.

Clustering algorithms are processes that use observed similarities or densities in data to identify patterns and group similar observations (Berkhin, 2006). Unlike global methods like Principal Component Analysis (PCA) which map the entire data space to a lower-dimensional space, clustering algorithms account for subspaces in the data. This means that some data dimensions or correlations may be relevant for defining a subset of clusters while remaining irrelevant to the definition of others. These types of subspace-specific cluster definitions are called *local feature relevance* and *local feature correlation*, respectively, and it implies that clusters can form in different subspaces of the data (Kriegel, Kröger, & Zimek, 2009). These subspace response patterns can be thought of as *unscalable classes* because observed responses on each variable do not form a scale (Goodman, 1975). For example, on a test where harder items are only answered correctly by students who are located high on a construct, item scores follow a logical scale where students are expected to answer items correctly until they become too

difficult—this is a scalable class (Braaten, 2009). When responses do not meet this scalable class criterion—as can be the case with work process data—the response patterns represent an unscalable class. Clustering algorithms are designed to work in these types of datasets, grouping data by similarities and correlation instead of ranking patterns based on response observations on more and more difficult (rarely observed) variables.

There are dozens of clustering algorithms that are used for different research questions and data scenarios [see Kriegel et al. (2009) and Berkhin (2006) for overviews and taxonomies of many of these algorithms]. The clustering methods used in GBA evaluation are called *soft-projected* clustering algorithms, which is a subset of *axis parallel* algorithms. These algorithms make the assumption that the subspaces in which clusters might exist are restricted to the spaces which are *parallel* to subsets of dimensions defined by the global data and that the number of clusters (k) is known in advance, although there is no mechanism in the algorithm for confirming or verifying that k is correctly specified (Kriegel et al., 2009). To put it another way, the clusters are defined by cases which share similar values on a subset of variables in the data—each variable is an axis in the data space. If the members of a cluster were “projected” (plotted) onto those axes that define that cluster, one would see those cases bunched together. The space defined by a cluster is not rotated or transformed in any way (as would be the case with clusters defined by a localized PCA), which is why it is said to be “parallel” to the axes defined by the variables. “Soft-projected” means that all of the variables presented to the algorithm are considered with equal weight, and the algorithm

has no method for dropping variables that may be irrelevant for the definition of a given cluster.

Unfortunately, the number of clusters is not known in advance in GBA research. Researchers often need to validate the number of clusters they select through a combination of fit statistics, validation against competing algorithms, and expert review of the interpretability of the results. It is also possible that multiple clusters, while representing different work processes, still share the same meaning in relation to the construct; e.g., groups of students who made different errors in a game which are all attributable to a single misconception or knowledge gap in the construct of interest. The validity of the clustering solution may also be influenced by the presence of irrelevant data and whether or not an appropriate clustering algorithm was selected (Berkhin, 2006; Kriegel et al., 2009; Xu et al., 2013).

There is ongoing research for data-mining applications in which an algorithm is able to discover and extract patterns or clusters within a dataset without any human guidance on the relevancy of variables or the nature and number of the patterns. Fayyad, Piatetsky-Shapiro, and Smyth (1996) called this field *knowledge discovery in databases* (KDD), and they provided a concise literature review of the field, its applications, and the limitations of data mining in KDD. Fayyad et al. describe many KDD initiatives, but all are designed for specific industries (e.g., healthcare, astronomy), and they did not mention any KDD work in education, assessment, or games. Furthermore, while KDD algorithms sound like a panacea for identifying hidden clusters in vast datasets, the accuracy of KDD algorithms is hindered by the same problems as other clustering algorithms, including high dimensionality, over-fitting models with “noisy” data, missing

data, complex relations, and the potential for uninterpretable clusters (Fayyad et al., 1996; Kriegel et al., 2009).

Dunietz (2016) echoed this reality check when discussing the limitations of deep neural networks that are able to identify their own high level data features, such as an algorithm that learns to detect curves in geometric images without having to be programmed to know which data represents curves. Dunietz explained that researchers still tweak these algorithms, selecting inputs, weights, and other parameters based on judgment and the interpretability of the results. In short, the challenges of selecting the number of clusters and the variables to include in the algorithm when analyzing GBA process data will not necessarily be solved through the use of more sophisticated data mining algorithms.

Additionally, it is important to remember that the use case—namely, grouping students based on similarities in response patterns—can also be addressed with parametric methods. For example, latent class analysis (LCA) creates conditional probabilities of whether or not a student is likely to belong to a given cluster. Xu et al. (2013) gathered usage data from 661 teachers working with an instruction design program and clustered them using both LCA and the k -means clustering algorithm. They found that the LCA clusters were better fitting and had better correlations with relevant external criterion variables than the k -means clusters. They also showed that when increasing the number of clusters (k), LCA tended to create new, small groups out of existing clusters, whereas k -means tended to create completely new, unrelated cluster solutions. While these methods are certainly valuable and promising, the fact remains that researchers are already using non-parametric, exploratory clustering algorithms in GBA

research, and the focus of this paper is on the performance, appropriateness, and proper application of clustering algorithms in a GBA context.

The following sections provide overviews of three axis parallel soft-projected clustering algorithms:

1. *k*-means – used in high dimensional datasets where data are expected to have roughly spherical distributions, clusters are similar sizes, and variables have equal variance (Robinson, 2017). For example, *k*-means would be appropriate for clustering people based on their responses on a set of Likert items.
2. SOM – used in high dimensional datasets, but with no assumptions about the data. SOM is used in a variety of applications, especially ones in which there is a need to inspect the similarities of clusters.
3. ROCK – used in high dimensional binary datasets, but with no assumptions about the data (beyond being binary). For example, ROCK can be used to group consumers into clusters based on the items they purchased, even though they may not buy exactly the same items.

GBA researchers are often working with high-dimensional datasets with sparse, nominal data, so researchers should select an algorithm that is able to find clusters under these conditions. These sections briefly describe these three algorithms' processes and their potential utility with respect to GBA evaluation.

***K*-means.** There are several versions of the *k*-means algorithm, but this paper focuses on the version defined by Hartigan and Wong (1979), which is generally accepted as the preferred *k*-means algorithm (Berkhin, 2006; R Core Team, 2014). In *k*-

means solutions, a cluster is defined by a center point which is the centroid for the members of that cluster. The algorithm begins with a set of k potential centers which can be defined by the researcher or by randomly selected k cases from the data to act as the initial centers of the clusters.

The choice of starting centers is deterministic: given the same starting centers, k -means will always return the same clustering solution (e.g., Pelleg & Moore, 2000; Ishioka, 2005). Lattin, Carroll, and Green (2003) warned that the k -means solution is only as good as its initial starting centers and that the selection of the initial centers heavily impacts the clustering solution for small datasets.

Once the centers are selected, the algorithm assigns all the cases to their closest centers and recalculates the new centers defined by these clusters. Distance is determined by a user-specified *similarity measure*—often Euclidean distance or Manhattan distance. The algorithm goes through multiple iterations of checking each case (e.g., student response pattern) to see if it should be moved to a different cluster based on the centers' updated coordinates. If so, it changes the case's cluster membership, updates the centers' coordinates, and continues to the next iteration until it converges on a solution where no points are being switched between clusters. In the case of Hartigan and Wong's (1979) algorithm, this process is handled by seven looping steps:

1. For each case x_i , such as student i 's vector of response pattern data, find the closest and second closest cluster centers m_{c1i} and m_{c2i} . Assign each case x_i to its cluster with the closest center, m_{c1i} . Starting centers for the clusters can be specified by the researcher or from cases randomly selected from the data. As the algorithm progresses and updates the coordinates of cluster centers, each

case's second closest cluster center (\mathbf{m}_{c2i}) is checked to see if it has become the closest cluster center to \mathbf{x}_i (see Step 7), in which case \mathbf{x}_i 's membership is changed and cluster centers are updated again.

2. Update all cluster centers \mathbf{m}_k by calculating the mean of each variable in vector \mathbf{x} for all cases \mathbf{x}_i currently assigned to \mathbf{m}_k .
3. Select a case, \mathbf{x}_i and define the "live set" of clusters, L .
 - a. If this is the initial cycle of the algorithm (i.e., the first time Step 3 has run), assign all k clusters to the live set of clusters, L .
 - b. If the algorithm has already completed at least one cycle, and if cluster \mathbf{m}_k was updated in the previous *quick-transfer* stage (Step 6 below) or if cluster \mathbf{m}_k was updated in any of the previous N *optimal-transfer* stages (Step 4 below), where N is the number of cases \mathbf{x}_i currently assigned to \mathbf{m}_k , then include \mathbf{m}_k in the live set of clusters, L .

4. Run the optimal-transfer stage of the algorithm.

- a. If \mathbf{x}_i 's assigned cluster, \mathbf{m}_{c1i} is in L based on the criteria defined in Step 3, calculate:

$$R_{2ki} = \min[N_k \times \text{dist}(\mathbf{x}_i, \mathbf{m}_k)^2] / (N_k + 1) \quad \forall \mathbf{m}_k \neq \mathbf{m}_{c1i}$$

(1)

Otherwise, if \mathbf{x}_i 's assigned cluster, \mathbf{m}_{c1} is not in L , calculate:

$$R_{2ki} = \min[N_k \times \text{dist}(\mathbf{x}_i, \mathbf{m}_k)^2] / (N_k + 1) \quad \forall \mathbf{m}_k \neq \mathbf{m}_{c1i} \in L$$

(2)

where N_k is the number of cases assigned to the k th cluster and $\text{dist}(\mathbf{x}_i, \mathbf{m}_k)$ is the distance between \mathbf{x}_i and the center of \mathbf{m}_k using the distance

measure chosen by the researcher (typically Euclidean distance by default).

- b. The cluster with the smallest R_{2ki} becomes m_{c2i} for x_i .
- c. If $R_{2ki} < [N_{c1i} \times \text{dist}(x_i, m_{c1i})^2] / (N_{c1i} - 1)$, where N_{c1i} is the number of cases in the cluster to which x_i is currently assigned, then m_{c2i} becomes m_{c1i} , m_{c1i} becomes m_{c2i} , and x_i is assigned to the cluster with the new m_{c1i} center. Update the cluster centers based on the new case assignment after each case x_i is evaluated.

5. If L is empty, stop the algorithm.

6. Run the quick-transfer stage of the algorithm.

- a. Consider all cases x_i whose clusters for m_{c1i} and m_{c2i} have changed in the past N cycles.
- b. For these cases, calculate:

$$R_1 = [N_{c1i} \times \text{dist}(x_i, m_{c1i})^2] / (N_{c1i} - 1)$$

(3)

$$R_2 = [N_{c2i} \times \text{dist}(x_i, m_{c2i})^2] / (N_{c2i} + 1)$$

(4)

- c. If $R_2 \geq R_1$, then m_{c2i} becomes m_{c1i} , m_{c1i} becomes m_{c2i} , and x_i is assigned to the cluster with the new m_{c1i} center. Update the cluster centers based on the new case assignment.

7. If no cases have changed cluster assignments in the past N cycles (i.e., the past N times these seven steps have been executed), then return to Step 4, otherwise return to Step 6.

The k -means algorithm is simple to understand, apply, and communicate, and it is likely the most widely-used clustering algorithm; however, it is very limiting. The algorithm can only be used for spherical interval/ratio data. Because the centroids are means, they can be influenced by outliers. The algorithm also does not scale well to high-dimensional datasets, especially when the data are sparsely populated. In fact, if data are not spherically distributed, variances within clusters are not equal, or some clusters are far larger than others, the k -means algorithm can fail, drawing cluster boundaries through true clusters in the data (Berkhin, 2006). The severity of such failures in practice will depend on the scenario and the application. Some researchers may still be able to interpret the clusters correctly, even if numerous cases were assigned by an incorrect boundary, but it is just as likely that the final k -means solution does not reflect the true clusters at all. If the researcher does not validate the findings, he or she may be missing the key interpretable structures of the data.

These limitations make k -means a poor choice for many GBA datasets which can easily have high-dimensional, sparsely populated, binary data. Alternate algorithms which are similar to k -means (e.g., k -median) can be more appropriate for binary data and have been used in some GBA development work (e.g., Corrigan et al., 2014; Kerr et al., 2011). Nevertheless, it is possible that some researchers use k -means clustering algorithm in inappropriate scenarios due to their own familiarity of the algorithm and its easy availability (Kriegel et al., 2009). For example, in their review of nine education studies using clustering algorithms, Xu et al. (2013) observed that five of those studies used k -means, and in their chapter discussing clustering algorithms for multivariate analysis, Lattin et al. (2003) only discussed Hartigan and Wong's (1979) k -means algorithm

without any mention of k -medians or discussion of other variants. Given its widespread use and easy availability despite the limitations discussed above, k -means is included in this study to see how it compares to other methods for the purposes of GBA classification.

Self-Organizing Maps (SOM). The SOM algorithm (also known as a Kohonen Map) is an artificial neural network algorithm where multidimensional data is mapped to a set of k clusters (“nodes”). One of the primary reasons SOM is popular is that the clusters can be mapped to a two-dimensional grid that shows which clusters are similar to each other. This is possible because the SOM uses a two dimensional array of nodes with weights representing cluster centers. For each presentation of the dataset (and there can be many), each case updates the location of the closest cluster center, as well as all of the cluster centers of nodes in a neighborhood around the winning node. The cumulative effect of doing this neighborhood update for each case over multiple presentations of the dataset is that the nodes self-organize into cluster centers where similarity is inversely related to the distance to each other on the grid (Kohonen, 1990). The axes and distances of the grid are meaningless, but it creates an organized map of cluster centers with similar clusters near each other in a grid. This is a valuable tool of visualizing data and validating clusters (Berkhins, 2006).

When mapping to a single set of clusters, the algorithm is very similar to the k -means algorithm. Like k -means, cases are mapped to the closest center, often determined by Euclidean distance. The researcher can specify a set of starting centers, or the algorithm will randomly select the centers from the data, and the researcher also has to specify how many clusters to identify (Bullinaria, 2004). As with the Hartigan and Wong

(1979) version of the k -means algorithm, SOM updates the centers after assigning a case to a center's cluster and continues to do so until cluster memberships stop changing (Berkhins, 2006).

The primary difference between k -means and SOM is that each time a case is assigned to a cluster, the center for that cluster is updated using a learning rate coefficient, but the centers for nearby clusters are also updated with the same learning rate coefficient to bring them closer to the winning cluster's center. This adaptive step is desirable because it allows the researcher to map k clusters onto a grid or table with k cells based on the similarities between their centers' coordinates; i.e., clusters can be displayed in a grid such that similar clusters are closer to each other. The axes and dimensions of the grid do not represent anything in of themselves (i.e., the researcher with 12 clusters could specify a 4×3 grid or a 6×2 grid), but the adaptive steps will assign each cluster to a space on the grid so that neighboring clusters are similar to each other. This tabular grid representation of the clusters cannot be achieved with k -means because it does not adjust nearby clusters based on a case's newly assigned membership. The actual dissimilarity between clusters in multivariate space will of course vary with both algorithms—some cluster pairs may be more similar than others. The SOM grid is simply a useful tool for interpretation and summarization.

The magnitude of the change of nearby clusters in the adaptive step depends on their proximity to the winning cluster center and the number of iterations (t) the algorithm has run. The SOM starts with a large learning rate coefficient (α_t) which is used to shift the cluster centers (\mathbf{m}_k) for all clusters in a large neighborhood (N_c) surrounding the winning cluster's center (\mathbf{m}_c). The initial radii of the neighborhoods can be set by the

researcher, but it diminishes as the algorithm runs. In other words, as time (iterations) progress, the neighborhood around each cluster shrinks to zero so that nearby clusters are not modified when a cluster is updated, and the clusters themselves are not changed as much by the presentation of new cases because the effect of these new cases is weighted by a decreasing learning algorithm. This is useful in cases where the researcher presents the same cases to the SOM network over and over again to achieve a more stable estimate of cluster centers. The initial cluster changes are large, with cluster centers being moved substantially by new cases and by changes in neighboring clusters. As the algorithm runs through its iterations, the learning rate coefficient and the size of the neighborhood shrink until eventually there are only minute, fine-tuning changes to the winning cluster's center (Bullinaria, 2004).

The size of the neighborhood and the rate of decrease can be set by the researcher. The rate of decrease may be linear or non-linear, and the neighborhood may exist for all of the SOM iterations, or it may be defined so that the neighborhood radius shrinks to zero after a set number of iterations have been completed. For example, in the default settings of the `som` package for R, the neighborhood's radius is chosen to be larger than $2/3$ of the unit-to-unit distances for all of the starting cluster centers (\mathbf{m}_k). The `som` package then linearly decreases the radius of the neighborhood over $1/3$ of the iterations chosen by the researcher (Wehrens & Buydens, 2007). If $2/3$ of the starting cluster centers are 100 Euclidean distance units away from each other, and the researcher specifies 300 iterations, then the radius of the neighborhood will decrease by one unit at each of the first 100 iterations, after which only the winning cluster's center (\mathbf{m}_c) will be updated. Once the neighborhood radius diminishes to zero, clusters near the winning

cluster are no longer updated when cases are reassigned, and the SOM algorithm solution is then identical to the logic used by the k -means algorithm (Kohonen, 1990).

$$\mathbf{m}_{k,t+1} = \begin{cases} \mathbf{m}_{k,t} + \alpha_t(\mathbf{x}_{i,t} - \mathbf{m}_{k,t}) & \text{if } i \in N_{c,t} \\ \mathbf{m}_{k,t} & \text{if } i \notin N_{c,t} \end{cases} \quad (5)$$

Equation 5 shows how these updates occur at every iteration t . The SOM algorithm begins by creating k clusters with centers \mathbf{m}_k by randomly selecting k cases from the data. At each iteration t , the SOM algorithm selects a case \mathbf{x}_i from the data and finds the cluster whose center is closest. This closest cluster has center \mathbf{m}_c . The algorithm then finds all of the clusters in the neighborhood of the winning cluster. The size of the neighborhood $N_{c,t}$ shrinks as iteration t increases. The centers \mathbf{m}_k of all of the clusters in $N_{c,t}$ are then updated with the learning rate coefficient α_t to be closer to $\mathbf{x}_{i,t}$. The algorithm then moves to the next iteration, selects a new case, decreases the size of the neighborhood radius and learning coefficient, and proceeds to update the cluster centers again (Kohonen, 1990). Note that a case's cluster membership is defined by its closest cluster, but the closest cluster may change as the cluster centers are updated in each iteration. The SOM algorithm completes when the cases' cluster membership stop changing. Figure 4 depicts the iterative cluster center updates in a two-dimensional space.

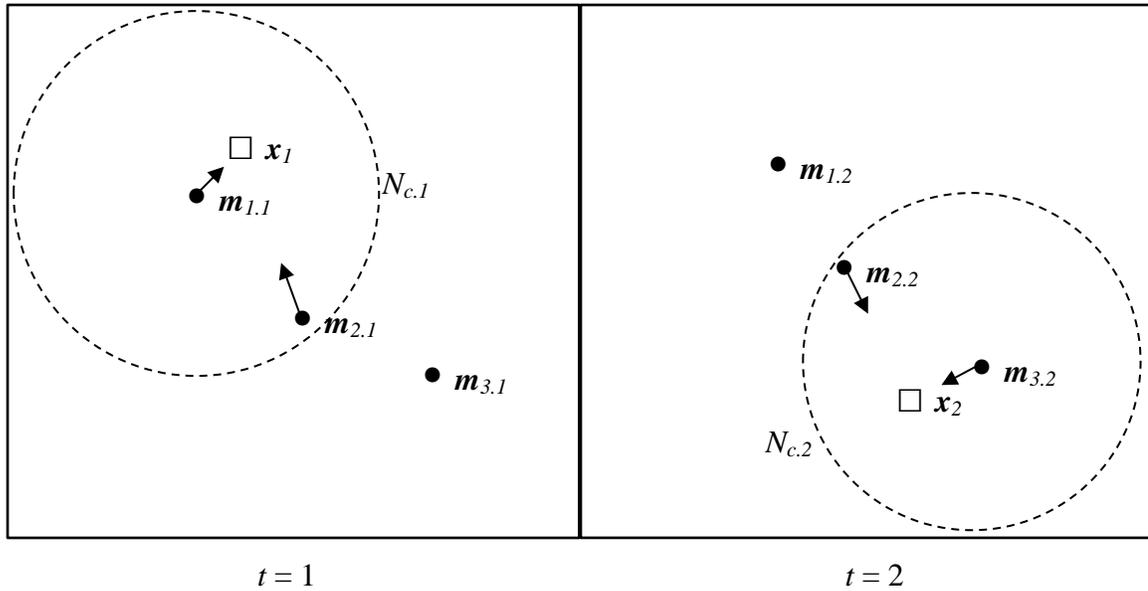


Figure 4. Visualization of the SOM learning process updating cluster centers across two iterations. Cluster centers within the winning cluster's neighborhood are updated to be closer to the case's data in each iteration.

In addition to the appeal of displaying topological relationships of the cluster centers in a two-dimensional grid, SOM has other benefits for GBA applications. First, SOM does not rely on any assumptions about the distributions of the data and the solutions are not heavily influenced by outliers (Wehrens & Buydens, 2007). This is because, unlike k -means, SOM does not ever calculate a cluster center's coordinates by taking the mean coordinates of all the cases assigned to the cluster. Instead, the cluster centers are moved incrementally depending on the case considered at each iteration.

On the downside, SOM, like k -means, is dependent on the starting centers, but it is not deterministic. The final clustering solution will be a product of the initial starting centers, the order in which cases were added to update the cluster centers, and the size of the neighborhood and learning rate coefficient at each of those iterations. Furthermore, researchers may not always have a theory to guide the parameter settings for the size of the neighborhood, the learning rate coefficient, or the rate at which these parameters

change, which might make the final solution somewhat arbitrary. Like k -means and other algorithms which rely on similarity measures, SOM may not work well for sparse datasets (Berkhins, 2006).

Robust Clustering Using Links (ROCK). One problem with GBA data is that data are frequently tagged as binary variables with a 1 representing the observation of an action and 0 representing no observation. This measurement scale has two implications: the datasets become sparse (large number of variables with lots of 0 values), and that sparsity means that many cases look similar simply because they have so many 0 values in common.

To handle these data scenarios, clustering algorithms like the Robust Clustering with Links (ROCK) algorithm have been developed which cluster cases based on the co-occurrence of categorical data. The ROCK algorithm also uses a similarity measure, and like k -means and SOM, the researcher can pick the metric, but the researcher must also define the threshold for labeling two cases as similar. If they are similar, they can be linked to each other and to other similar cases. Using this method, many links can be made between cases, but the ROCK algorithm is built around the concept that two similar cases are more likely to belong to the same cluster if they share a high number of links; i.e., they are similar to a lot of the same cases. ROCK uses a bottom-up hierarchical clustering to combine clusters until the algorithm finds k clusters (k being specified by the researcher). These clusters are defined by finding a clustering solution that maximizes the within-cluster links for all pairs of cases in the cluster (Guha, Rastogi, & Shim, 1999). The bottom-up hierarchical clustering steps are as follows:

1. Using a similarity measure (e.g., distance measure) specified by the researcher, assign all pairs of cases a similarity value between 0 and 1, where 1 represents identical cases and 0 represents maximum dissimilarity.
2. The researcher specifies a similarity threshold, θ . If a pair of cases' similarity value is equal to or greater than θ , then those two cases are considered neighbors.
3. For each case, identify all of its neighbors.
4. For each pair of cases, calculate the number of links, which is the number of common neighbors they share; i.e., the number of overlapping neighbors for the pair of cases. The concept behind ROCK is that if two points share a lot of neighbors (high number of links), then they are more likely to belong to the same cluster, even if the two cases are not similar to each other.
5. Assign all cases to a cluster. At the first iteration, each case is its own cluster.
6. For a pair of clusters C_i and C_j , calculate the number of links between the cases \mathbf{x}_q in C_i and cases \mathbf{x}_r in C_j :

$$link(C_i, C_j) = \sum link(\mathbf{x}_q, \mathbf{x}_r) \text{ such that } \mathbf{x}_q \in C_i \text{ and } \mathbf{x}_r \in C_j \quad (6)$$

7. To determine which two clusters should be merged next, calculate the *goodness measure* (g) for all pairs of clusters, where n_i is the number of cases in C_i and n_j is the number of cases in C_j :

$$g(C_i, C_j) = \frac{link(C_i, C_j)}{(n_i + n_j)^{[1+2(\frac{1-\theta}{1+\theta})]} - n_i^{[1+2(\frac{1-\theta}{1+\theta})]} - n_j^{[1+2(\frac{1-\theta}{1+\theta})]}} \quad (7)$$

8. Merge the two clusters with the highest goodness measure and update all link calculations between clusters. Continue iteratively merging clusters with this

bottom-up hierarchical clustering method until the desired number of clusters, k , is obtained.

ROCK is a good candidate for GBA data because of its ability to handle sparse datasets. It does not have any requirements for data distributions, and because of the method of identifying similarities and links, it essentially segregates and ignores outliers, which may be helpful in GBA process data if there are occasional, atypical response patterns. Like SOM, ROCK requires the researcher to specify the parameters, in this case the similarity measure to be used and the similarity threshold (θ) constraining the number of links in a case's neighborhood, and researchers may not know how to select these values.

To summarize, GBAs can be considered assessments that have construct-irrelevant game elements included to make the assessment more fun or engaging, but these elements may interact with relevant response patterns. Furthermore, high autonomy game environments increase the number of response patterns that may be observed, and as a result, GBA data often take the form of sparse, binary datasets. When designing the GBA, researchers may not be able to use a principled approach to plan out how to evaluate every possible response pattern. Instead, researchers must turn to empirical approaches to mine response data and identify interpretable patterns in the data. These empirical approaches typically employ clustering algorithms like k -means, SOM, and ROCK, but algorithms may be selected based on convenience, and researchers may not validate the results with multiple approaches.

To demonstrate the potential differences and between these algorithms and the potential impact on a researcher's interpretation of sparse, binary GBA data, this study

seeks to evaluate how well the SOM, k -means, and ROCK algorithms identify clusters in the presence of data that are irrelevant to the structure of the clustered data. Various data conditions (detailed in Chapter 4) are simulated to make these comparisons. In addition, the three clustering algorithms are used to identify clusters within real GBA process data from a math game, Beanstalk (Carnegie Mellon, 2013). These results are discussed in a tutorial study, examining the steps and choices that should be addressed when using clustering algorithms with GBA data. These inquiries illustrate how these algorithms can differ under varying data scenarios and provide guidance or cautions to researchers using empirical approaches to evaluate GBA performance.

Chapter 3: Feasibility Study

A small-scale feasibility study was conducted to explore how to analyze and compare the success of the k -means, SOM, and ROCK algorithms in high-dimensional, sparse, binary datasets that are common in GBA research. The feasibility study did not seek to address any research questions directly, but instead served to demonstrate that these three algorithms could yield results that could be compared and contrasted. It also served as a learning exercise to help explore practical issues before moving to a full-scale study (e.g., how to standardized cluster labels so that comparisons can be made between the algorithms' outputs). The feasibility study specifically focused on two small simulated datasets and a sample GBA dataset from a multiplication game. An artificial dataset was also appended to the GBA dataset to mimic a set of ten students who consistently used an incorrect process (addition instead of multiplication).

Simulated Data for the Feasibility Study

The first simulated dataset, S1, contained 50 cases and only three variables; these 50 cases were generated as coming from three clusters. The base probability of an observed binary response in any of the three variables, v_1-v_3 , was $Pr(v_i=1) = 0.05$, with exceptions for each cluster and variable shown in Table 1. Each case's observed response to the three variables was randomly generated from a binomial distribution using the `rbinom` function in R (R Core Team, 2014), using the specified probabilities reported in Table 1.

Table 1
S1 Simulated Data Parameters

Cluster	n	$Pr(v_1=1)$	$Pr(v_2=1)$	$Pr(v_3=1)$
1	15	0.90	0.05	0.05
2	15	0.05	0.90	0.05
3	20	0.05	0.05	0.90

The second simulated dataset, S2, also contained 50 cases but had six variables, v_1-v_6 . The probability of an observed response in any of the six variables was $Pr(v_i=1) = 0.05$, with exceptions for each cluster and variable shown in Table 2a. As with S1, each case's observed response to the six variables in S2 was randomly generated from a binomial distribution using the `rbinom` function in R (R Core Team, 2014), using the specified probabilities reported in Table 2a. An example of cases from this simulated dataset, S2, is shown in Table 2b below.

Table 2a
S2 Simulated Data Parameters

Cluster	N	$Pr(v_1=1)$	$Pr(v_2=1)$	$Pr(v_3=1)$	$Pr(v_4=1)$	$Pr(v_5=1)$	$Pr(v_6=1)$
1	15	0.90	0.50	0.05	0.05	0.05	0.05
2	15	0.05	0.05	0.90	0.50	0.05	0.05
3	20	0.05	0.05	0.05	0.05	0.90	0.50

Table 2b
Example of S2 Simulated Data (First Six Cases)

Case	Cluster	v_1	v_2	v_3	v_4	v_5	v_6
1	1	1	0	0	0	0	0
2	1	1	1	0	0	0	0
3	1	0	1	0	0	0	0
4	1	1	0	0	0	0	0
5	1	1	0	0	0	0	0
6	1	1	0	0	0	0	0

GBA Data for the Feasibility Study

GBA data were provided by Derek Lomas (personal communication, July 4, 2014) from the game Number Jumble (BrainPOP, 2015). The data are anonymous, and information was not available about the conditions under which they were collected or the sample of students who played the game.

Number Jumble is a multiplication game where students must move numbered tiles on a grid to coordinates so that the product of the coordinates equals the value on the tile (BrainPOP, 2015). For example, a student would move the 6 tile to the intersection of 2 and 3 because $2 \times 3 = 6$ (see Figure 5). In many ways, it is similar to the 15 Puzzle example discussed in Chapter 1, in that it has a grid of numbers, and there are ways to solve the Number Jumble grid more efficiently. Several grids with varying sizes and factors can be played in Number Jumble as different levels. Of the 328 response records provided for 14 levels, all but one of the levels had a sample size of 40 or fewer (mean = 16 results, s.d. = 14 results). The remaining level had 109 response records, and it also happened to be the simplest grid: a 3×3 grid with coordinates on both axes equal to [1, 2, 3], as is shown in Figure 5. Given the larger sample size and its simplicity, this remaining level was chosen for the demonstration and proof of concept in the feasibility study.



Figure 5. Level Mul_1 from Number Jumble (BrainPOP, 2015).

The tiles in Number Jumble are randomly placed at the start of the level with the stipulation that tiles cannot be placed on their solution coordinates. The student must then use his or her mouse to drag tiles to their correct coordinates (the displaced tile will switch positions with the tile moved by the student). Bonus points are earned by working quickly. The student may also pick up a tile and then return it to its starting point (BrainPOP, 2015).

Table 3 shows an example of log file data from Number Jumble that record every move made by the student. Recall in Chapter 1 how one could conceive of recording 15 Puzzle gameplay as tile movements, sequences, and move times—this is what Number Jumble has recorded in its work process data, along with additional information. The Number Jumble data represented individual moves made by students, and each move’s sequence and time stamp were recorded (see “Seq.” and “Time Stamp” columns in Table 3). Results were labeled with a student identification number (“SID”). The number of moves in a result ranged from 1 to 54, with an average of 8.7 moves (s.d. = 7.4). A player who made all correct moves would be able to complete this 3×3 level in eight moves (the ninth solution is automatic), assuming no added benefits like having a displaced tile also land in its solution coordinates.

Table 3

Excerpts of Number Jumble Log File Data (First Five Moves)

Time Stamp	SI D	Seq.	Pickup Numbe r	Picku p X	Picku p Y	Dro p X	Dro p Y	result	Level Name	Reaction Time
4/13/2013 15:34	999	1	2	1	1	2	1	correct	Mul_ 1	1.1
4/13/2013 15:35	999	2	6	1	1	3	2	correct	Mul_ 1	1.3
4/13/2013 15:35	999	3	4	3	3	1	2	in correc t	Mul_ 1	0.6
4/13/2013 15:35	999	4	2	2	2	2	3	in correc t	Mul_ 1	1.0
4/13/2013 15:35	999	5	9	1	3	2	3	in correc t	Mul_ 1	1.1

These data were uploaded into the R statistical software (R Core Team, 2014) for analysis. The dataset of Number Jumble response cases from the game level shown in Figure 5 was labeled M1. A second dataset, M2, was created by adding a small set of simulated cases representing a hypothetical incorrect response patterns to the real responses in M1. The following paragraphs detail the construction of M1 and M2.

M1: Responses. Each move was coded using the format <product>.<factor>.<factor>. This means that if a student moved the 1 tile to the coordinates (3, 2), the move was coded 1.3.2 (see Table 4). Data were coded binary, with 1 indicating the student made the move, and 0 indicating that the student did not make that move. No distinctions were made for students who made the same wrong move multiple times versus those who made a wrong move once, nor were distinctions were made between commutative factors. For example, a student must make the moves 6.3.2 and 6.2.3 in the game, but these were combined into two instances of the same multiplication problem and coded under the same variable: 6.3.2. Additionally, if a student picked up a tile and returned it to its starting position, the move was coded 999.

Table 4 shows an example of the data in M1. The first case, SID 13693303257821.1, shows that the student knew that $1 = 1 \times 1$ (i.e., 1.1.1 = 1), but also tried either $1 = 2 \times 1$ or $1 = 2 \times 1$ (i.e., 1.2.1 = 1). The student did not, however, try $1 = 2 \times 2$ (i.e., 1.2.2 = 0). For simplicity, no additional data about response order or game state are included in M1 for this initial exploration.

Table 4

Excerpt of M1 Data (First Six Cases)

SID	<product>.<factor ₁ >.<factor ₂ >				
	1.1.1	1.2.1	1.2.2	1.3.1	1.3.2
13693303257821.1	1	1	0	1	0
13693191284070.1	1	0	0	0	0
13693275705245.1	0	0	0	0	0
13693276443395.1	0	0	0	0	0
13693307246903.1	0	0	0	0	0
13693312091466.1	0	0	1	0	1

M2: Added simulated data. To help gauge the accuracy of clustering solutions with a set of responses from a known cluster, ten artificial cases were appended to the real data in the M1 dataset to create a second dataset, M2. These ten cases were copies of ten randomly selected cases in M1, but observations were changed from 0 to 1 for three variables: 2.1.1, 3.2.1, and 6.3.3. This pattern would be observed for a small cluster (i.e., these 10 results) where students mistakenly used addition instead of multiplication for three of their responses. For example, 2.1.1 = 1 represents that a student made a move representing $2 = 1 \times 1$, perhaps thinking instead that they were supposed to calculate $2 = 1 + 1$. Similarly, the correct products were marked as unobserved for the ten simulated cases (e.g., 2.2.1 = 0). Since these were artificial cases purposefully created to share similarities in their incorrect response patterns, they are expected to all be mapped to the same cluster in any clustering solution. If they are split between clusters, then the

clustering algorithm has failed to recover this simulated set, which may suggest that its cluster assignments are not useful. Table 5 shows an example of the response data for 2.1.1 and 2.2.1 for some of these artificial cases.

Table 5

Excerpt of M2 Simulated Data (First Six Cases)

SID	<product>.<factor ₁ >.<factor ₂ >				
	2.1.1*	2.2.1*	2.2.2	2.3.1	2.3.2
adder1	1	0	0	0	0
adder2	1	0	0	0	0
adder3	1	0	0	0	0
adder4	1	0	0	0	1
adder5	1	0	0	0	1
adder6	1	0	0	0	0

*Manipulated by researcher to define simulated group.

Procedure

An exploratory analysis was used to compare clustering solutions for the simulated datasets, S1 and S2, and the Number Jumble datasets, M1 and M2 (which included the simulated Number Jumble cases). The k -means and SOM algorithms were run through 20,000 cycles to yield the most stable solution. For the k -means algorithm, this means that the k -means algorithm was run 20,000 times using a different random sample of starting cluster centers each time. The most stable solution of those 20,000 is the solution that minimizes the within sum-of-squares distances for the clusters. For the SOM algorithm, this means that the dataset was presented to the SOM network 20,000 times, with each presentation resulting in iterative updates to the cluster centers that decrease as the neighborhood and learning coefficient decay over each iteration. The ROCK algorithm was run only once because it always yields the same results for a given dataset and theta parameter. At a high level, the analysis procedures followed this pattern:

1. Run 20,000 cycles of k -means and SOM for each dataset, specifying three clusters for S1 and S2 and two clusters for M1 and M2. Three clusters were selected for S1 and S2 since these were known from the simulated data. Two clusters were selected for M1 and M2 based on exploratory analysis of fit for different numbers of clusters, which is discussed further in the following section.
2. Run ROCK algorithm once for each dataset, adjusting theta downward after each solution until the hierarchical algorithm converges on three clusters for S1 and S2 or two clusters for M1 and M2.
3. Compare clustering solutions by looking at the degree of agreement between two solutions, their recovery of known clusters from the simulated data, and how well the solutions fit the data.

The following sections discuss the procedures developed around each of these steps, as well as the findings and decisions that were made through the process.

Setting the number of clusters. As noted previously, cluster validation often hinges on interpretability (Hand, Mannila, & Smyth; 2001; Kriegel et al., 2009), so the most useful clustering solution might not be the best-fitting solution. Even objective methods of identifying the number of clusters may still rely on a person's interpretation. For example, Hothorn and Everitt (2009) recommend plotting the within sum of squares for each clustering solution over a range of possible numbers of clusters. Several methods have been proposed for determining the optimal number of clusters based on the Bayesian Information Criterion (BIC) or the Akaike Information Criterion (AIC), and Pelleg and Moore (2000) developed an algorithm, x -means, which uses these information criteria to determine the optimal number of clusters. Extensions of this strategy have been

developed for non-spherical Gaussian distributions (Hamerly & Elkan, 2004) and for SOM algorithms (Wang, Yang, Mathee, & Narasimhan, 2005).

Pelleg and Moore (2000) found that x -means found better clustering solutions than repeated k -means trials on simulated Gaussian distributions, suggesting that the number of clusters determined by x -means might be a stable solution, but in a simulation study, Ishioka (2005) found that x -means did not always return the same recommended number of clusters, and in cases where a cluster was split evenly, x -means might return more clusters than expected. The x -means algorithm also performs well with large datasets (Pelleg & Moore, 2000), but it may not return optimal solutions when the data are not scaled in spherical, multivariate normal distributions around centroids (Hamerly & Elkan, 2004).

Given these constraints, x -means is not an acceptable method for selecting the number of clusters in sparse, binary GBA response data. Exploratory analysis of the Davies-Bouldin Index (Davies & Bouldin, 1979) for the clustering solutions for M1 suggest that there may not be any clusters (i.e., $k = 1$). This may reflect the simplicity of the Number Jumble level. Clustering algorithms will produce any number of clusters requested, even if it does not make sense to do so. For the purposes of this feasibility study, M1 can be analyzed with $k = 2$ clusters, even though there may not actually be two clusters in the data. This still provides a basis of comparison for when the clustering algorithms are run (also with $k = 2$) with the added simulated cases in M2. S1 and S2 are simulated to have $k = 3$ clusters.

Running clustering algorithms. For each dataset, three different analyses were run: k -means, SOM, and ROCK. The k -means algorithm was run 20,000 times, with each

cycle beginning with a different set of k starting centers randomly selected from the cases in the dataset. Similarly, the SOM algorithm was run through 20,000 cycles, where the dataset was presented again to the SOM network at each cycle, allowing the estimates of the cluster centers to be fine-tuned. SOM and k -means clustering solutions are both dependent on the starting centers. The ROCK algorithm was run once per dataset because it does not require starting values and can only return one possible solution for a given dataset. This allowed for comparisons of agreement between the three algorithms' results for each dataset, as well as comparisons of their results to the known cluster structures that were used to create datasets S1, S2, and M2.

For the k -means analyses, R was used to run the `kmeans` function in the `stats` package using k clusters (R Core Team, 2014). The default value of one random starting set per analysis was used, meaning that for each of the 20,000 iterations of `kmeans`, it randomly selected k of the cases from the dataset as its starting points. Following the recommendations in the `kmeans` documentation, the default k -means algorithm from Hartigan and Wong (1979) was used.

For the SOM analysis, R was used to run the `som` function in the `kohonen` package (Wehrens & Buydens, 2007). The grid for mapping the clusters was set to have dimensions of $1 \times k$ so that it could accommodate any k suggested by x -means. The default values were used for the number of times the dataset would be presented to the algorithm (i.e., 20,000 presentation updates) and for the SOM learning rate (a linear declination from 0.05 to 0.01 over each of the 20,000 updates). The default neighborhood radius was also used, meaning the starting radius value was set to cover $2/3$ of all unit-to-unit distances in the neighborhood of the potential cluster. A hexagonal grid was used (as

opposed to a rectangular grid) in a circular neighborhood, but there is no difference in visualizing the output when $k = 2$.

For the ROCK analysis, R was used to run the `rockCluster` function in the `cba` package (Buchta, & Hahsler, 2014). The similarity index, θ , was set to 0.95, which means that results needed to be very close together to create within-cluster links. If the ROCK algorithm could not converge on k clusters, then θ was decreased by increments of 0.05 until the ROCK algorithm did converge. Lower θ values indicate that there needs to be less overlap between two response patterns for a link to be established. Note that ROCK does not use starting centers, but its solution is based on the value of θ . This means that it returns the same clustering solution every time, given the same θ , which is why only one application of the ROCK was needed for each dataset.

Comparing clustering solutions. As Kriegel et al. (2009) observed, there is no universal method for comparing clustering solutions, and some clustering methods cannot be logically compared with others. Fortunately, k -means, SOM, and ROCK are very similar algorithms (i.e., they all are soft-projected clustering algorithms as described in Chapter 2, and they share the goal of segmenting data based on similarities), so one can examine the solutions to determine how well they agree and which solution fits best. For this feasibility study, comparisons were made between cases' individual cluster assignments from k -means, SOM, and ROCK in order to determine the consistency between algorithms.

Before comparisons could begin, the clustering solutions had to be recoded so that identical clusters (or similar clusters) were labeled the same way. For example, if one

solution put 75 cases in cluster 1 and a second solution put the same 75 cases into cluster 2, it would be helpful to change the labels in the second solution so that the cluster was labeled cluster 1 in both solutions. Unfortunately, clusters do not always match up perfectly, so a procedure was developed to recode the solutions using the following steps:

1. Take two clustering solutions, A and B, which show the k cluster memberships for n cases.
2. For each clustering solution, A and B, create a $k \times n$ membership matrix, where cell $k_i n_j = 1$ if the j^{th} result has been classified in the i^{th} cluster of the solution and $k_i n_j = 0$ otherwise.
3. For each cluster row in the membership matrix for clustering solution A, calculate the distance to each cluster row in membership matrix B in the cases' space.
4. Find the cluster membership rows which are closest to each other from solutions A and B. These two clusters have the best overlap in terms of the cases. Give them the same cluster label.
5. Find the next pair of closest membership rows, omitting the first two rows. Give this new pair the same cluster label.
6. Repeat step 5 until there are no clusters left to match.

Figure 6 provides an example of this recoding process in a two dimensional space.

In Figure 6, the two clustering solutions have three clusters each (A1-A3 and B1-B3). Solutions B3 and A2 are closest, so they are recoded to have the same label. Solutions B1 and A3 are the next closest, so they too are recoded to have the same label. A1 and B2 are left and are given the same label. There were clustering solutions that were previously closer to A1 and B2, but they have already been paired with other clusters that were an

even better match. For the matrices described above, there would be n axes (one for each case), and each cluster solution would have a coordinate of either 1 or 0 on each axis.

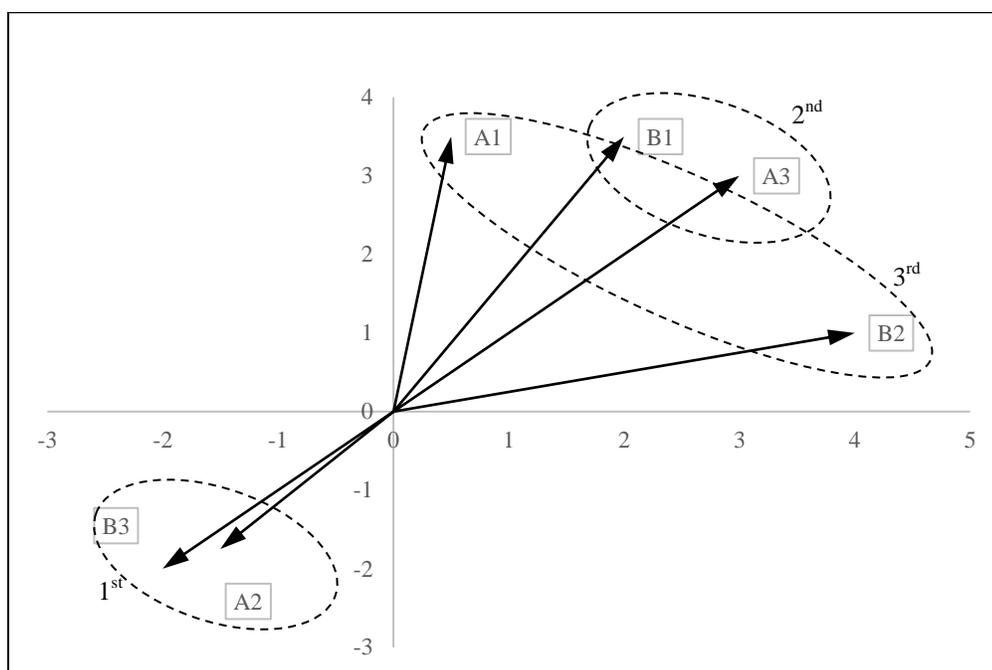


Figure 6. Example of recoding sequence of two-dimensional cluster solutions (A and B) with $k = 3$. The study data only had coordinate values of 0 or 1 and n dimensions.

Once the cluster solutions have their cluster identifiers matched, metrics are needed to compare the similarities between two clustering solutions. One simple comparison is the percentage agreement between two clustering solutions; i.e., the percentage of cases that were classified the same way in both algorithms. This was done by creating a pivot table showing the frequency of cases classified under each clustering solution. Table 6 shows an example of a hypothetical pivot table for a k -means clustering solution and a SOM clustering solution. Note that $89/109 = 82\%$ of the cases were classified in the same clusters by both algorithms in this comparison.

Table 6

Example of a Pivot Table Showing Classification Agreement Between Two Clustering Solutions

		Cluster Solution X	
		1	2
Cluster Solution Y	1	39	16
	2	4	50

Percentage comparison is a simple and practical measure for comparing the consistency between clustering solutions, but another common measure is Cohen's Kappa, which is often used to describe agreement between human raters. Cohen's Kappa is a measure of agreement that accounts for the amount of agreement that would have occurred by chance based on the individual raters' (or clustering algorithms') distribution of ratings. The unweighted kappa formula is used because the cluster categories are nominal (Cohen, 1960), and the formula for unweighted Kappa is:

$$\kappa = \frac{\Pr(a_o) - \Pr(a_e)}{1 - \Pr(a_e)} \quad (8)$$

where a_o is the observed rate of agreement and a_e is the expected rate of agreement based only on chance, given how many times two algorithms each assigned cases to each cluster. In the example in Table 6, the kappa value for this pair is 0.63. While 82% agreement is very useful for interpretation, it is also helpful to know that that agreement is not just an artifact of the clustering solutions. For example, two clustering solutions that dumped most cases into a single large cluster would have a high percentage agreement, but one might be interested on whether they agreed on which remaining cases did not belong to the large cluster.

For each cluster solution for each dataset in the study (S1, S2, M1, M2), the percentage agreement was calculated using R (R Core Team, 2014) and Cohen's Kappa

was calculated using the `cohen.kappa` formula in the `Rpsych` package (Revelle, 2015). The same statistics were calculated to evaluate how well the clustering solutions matched the expected “true” clusters used to create S1, S2, and M2.

Clustering solutions can also be evaluated and compared based on an internal criterion. One common method for evaluating a clustering solution is the Davies-Bouldin Index (Davies & Bouldin, 1979). The Davies-Bouldin Index (DBI) was originally proposed as a method for selecting the number of clusters, much as the *x*-means algorithm does, but unlike *x*-means results, the DBI can be used to compare multiple clustering solutions to find the one that provides the best result. The concept behind the DBI is that good clustering solutions should provide clusters that are not too similar to each other. If two clusters are close to each other or if clusters have a lot of within-cluster dispersion, then they are similar, but clustering solutions that are further apart and have little dispersion are more desirable. DBI is designed so that the clustering solution with the lowest index values is considered to be best (Davies & Bouldin, 1979). Another way to say this is that the best clustering solution is one where there is higher between-cluster variance and lower within-cluster variance. DBI is calculated with the following formula:

$$DBI = \frac{1}{k} \sum_{i=1}^k \max_{i \neq j} \frac{S_i + S_j}{M_{ij}} \quad (9)$$

where k is the total number of clusters, S_i is the average distance between the center of cluster i and its assigned cases (the same goes for S_j), M_{ij} is the distance between the centers of clusters i and j . Cluster j is the cluster that maximizes this ratio for cluster i , meaning that it is the most conservative scenario—cluster j is the cluster that is closest to cluster i or has high within-cluster variance (or both) (Davies & Bouldin, 1979). The

distance measure used for S_i , S_j , and M_{ij} is typically Euclidean distance, but DBI can use other distance measures as well.

From an application standpoint, this may not be the most useful solution. Interpretability may be more valuable than small differences in fit, and from the standpoint of GBA data, there is no reason that one would want to dismiss two similar yet different clusters of results. Recall that Amershi and Conati (2009) warned that clustering solutions might mask subtle, yet instructionally-relevant differences between student responses. One could imagine that this would be more likely if researchers blindly chose the output that yielded the biggest differences between clusters. With these cautions in mind, the Davies-Bouldin Index is still a suitable descriptive statistic for comparing the fit of different clustering solutions in this feasibility study, regardless of the algorithm.

For each of the four datasets in the study, the Davies-Bouldin Index was calculated for three clustering solutions using the `intCriteria` function in the `clusterCrit` package for R (Desgraupes, 2014). These indices were saved, along with identifiers indicating to which clustering algorithm solution they corresponded.

Performance of the Clustering Algorithms for Datasets S1 and S2

Before proceeding to the GBA datasets, M1 and M2, the clustering solutions were analyzed using the simulated data from S1 and S2. This provided insight for how these three algorithms compare with each other using simple, low-dimensional data.

First, the clustering solutions were compared to the true clusters used to generate S1 and S2 in order to evaluate how well they were able to recover the true clusters. Table 7 shows the true cluster recovery for S1, and Table 8 shows the true cluster recovery for S2. Kappa values and percentage agreement are shown below each matrix. Landis and

Koch (1977) indicated that Kappa values between 0.81 and 1.00 represent near perfect agreement. Notice in both tables that k -means and SOM perform just as well. They each have a good match with the true clusters in S1 (Kappa = 0.73, 82% agreement with the true clusters) and near perfect recovery in S2 (Kappa = 0.91, 94% agreement with the true clusters). Surprisingly, Tables 7 and 8 show that the ROCK algorithm tended to lump the results into a single large group, so the ROCK solution did a poor job recovering the three true clusters in S1 and S2 (Kappa = 0.07 and 0.43, respectively). This poor performance is unexpected for an algorithm designed to accommodate binary data, but it may be a reflection of the small number of variables in these datasets. The ROCK algorithm may perform better with more variables in the data, and this was considered in the proposal for the final study.

Consistency between clustering algorithms was investigated next. While perfect consistency might not be expected from different algorithms, it is worth seeing how well these three algorithms agree with each other if they are all potential candidates for evaluating the process data.

Table 7

S1: True Cluster Recovery by Algorithm

Cluster	k -means				ROCK				SOM		
	1	2	3		1	2	3		1	2	3
True 1	12	2	1	True 1	0	0	15	True 1	13	2	0
True 2	0	15	0	True 2	0	1	14	True 2	0	14	1
True 3	5	1	14	True 3	1	4	15	True 3	1	5	14
Kappa	0.73				0.07				0.73		
% Agreement	82%				36%				82%		

Table 8
S2: True Cluster Recovery by Algorithm

Cluster	<i>k</i> -means				ROCK				SOM					
	1	2	3		1	2	3		1	2	3			
True	1	18	2	0	True	1	0	0	15	True	1	19	1	0
	2	1	14	0		2	0	13	2		2	2	13	0
	3	0	0	15		3	1	1	19		3	0	0	15
Kappa	0.91				0.43				0.91					
% Agreement	94%				64%				94%					

Table 9 illustrates the level of agreement between the three clustering algorithms for their solutions for S1. The three matrices in Table 9 show the number of results classified into each cluster by the algorithms. One observes that the *k*-means and SOM solutions have good, but not perfect agreement (Kappa = 0.67). The ROCK solution has very poor agreement with the other two solutions because the ROCK solution grouped 44 of the 50 results into a single cluster. The ROCK and *k*-means solutions have a paltry 0.04 Kappa value, and the ROCK and SOM comparison yields an abysmal -0.17 Kappa. In this first example, *k*-means and SOM yield somewhat similar solutions, whereas the ROCK algorithm forced most cases into a single large cluster.

Table 9
S1: Cluster Agreement Between Pairs of Algorithms

Cluster	<i>k</i> -means				ROCK				SOM					
	1	2	3		1	2	3		1	2	3			
SOM	1	11	1	2	<i>k</i> -means	1	1	5	11	ROCK	1	0	1	0
	2	6	15	0		2	0	0	18		2	0	5	0
	3	0	2	13		3	0	0	15		3	14	15	15
Kappa	0.67				0.04				-0.17					
% Agreement	78%				38%				30%					

Table 10 shows the same algorithm comparisons for S2. Recall that S2 has six binary response variables whereas S1 had only three. Note the immediate improvement in agreement across all three comparisons shown in Table 8. For S2, the k -means and SOM algorithms had nearly perfect agreement ($\text{Kappa} = 0.94$). The agreement with the ROCK algorithm improved over the S1 solutions, but not drastically. Following the same behavior observed in S1, the ROCK algorithm again produced one large cluster, this time with 36 cases. The ROCK solution's agreement with the k -means solution was fair ($\text{Kappa} = 0.47$), and the agreement with the SOM solution was again the poorest ($\text{Kappa} = 0.06$).

These results yield some insights about how these algorithms perform under the data conditions in S1 and S2. First, notice that the consistency between SOM and k -means and their ability to recover the true clusters increases in S2. This is attributable to the number of variables—specifically the irrelevant variables where the probability of observation for a given cluster was 0.05. Since these algorithms are soft projection algorithms, they consider all of the variables equally. This means that one discrepant observation in a vector of process data holds less weight for assigning the cluster when there are more variables. Remember too that the unobserved variables are just as important as the observed variables for defining cluster membership.

Table 10
S2: Cluster Agreement Between Algorithms

Cluster	<i>k</i> -means				ROCK				SOM					
	1	2	3		1	2	3		1	2	3			
SOM	1	19	2	0	<i>k</i> -means	1	19	0	0	ROCK	1	21	13	2
	2	0	14	0		2	15	1	0		2	0	1	0
	3	0	0	15		3	2	0	13		3	0	0	13
Kappa	0.94				0.47				0.06					
% Agreement	96%				66%				42%					

Conversely, the ROCK algorithm struggled under these conditions. Although the ROCK algorithm is designed to handle binary data, there may not have been enough variables for the ROCK algorithm to perform well. Although Guha et al. (1999) do not specify a lower limit for the number of variables to use with the ROCK algorithm, its successful applications in research tend to have at least 15 (and as many as several hundred) variables (e.g., Guha et al. 2014; Song & Li, 2006). For datasets S1 and S2, it is possible that with so few variables, more links are identified in the ROCK algorithm, ultimately forcing many results into a single cluster in order to achieve the desired solution with $k = 3$ clusters. Because of this tendency to create one large cluster, the ROCK solution did not have good agreement with the other two algorithms and did a poor job at recovering the true clusters in S1 and S2.

SOM and *k*-means did an equally good job at recovering the true clusters in S1 and S2, but since they did not arrive at the same solution, which one was the best? According to the Davies-Bouldin Index (DBI), *k*-means performed slightly better. The DBI can be used to determine how well a given clustering solution creates well-defined clusters, as determined by small within cluster variance and larger between cluster

variance. Recall that lower values of DBI indicate better fit. Table 11 shows the DBI values for the three solutions for S1 and S2. Note that SOM had the best-fitting solution in S1, but k -means had a slightly better fitting solution in S2 (as indicated by having the lowest DBI). Given the high true cluster recovery rate of both SOM and k -means in S2, it is not surprising that their DBI indices are so close—they both returned results very similar to the true clusters.

Table 11
DBI Results by Solution for S1 and S2

Algorithm	DBI for S1	DBI for S2
SOM	0.51	1.02
k -means	0.61	0.96
ROCK	1.29	1.15

One can also inspect the true cluster recovery rate by looking at the cluster centers of the solutions. If they were a perfect recovery of the true clusters, these coordinates will be very close to the probabilities of observing variables in each of the clusters. Figures 7a and 7b show the magnitudes of the cluster centers for each solution and the true clusters in S1 and S2. Notice that for each cluster, the centers for the SOM and k -means solutions approximate the probabilities defining the true clusters, whereas the ROCK solution tended to have different cluster centers.

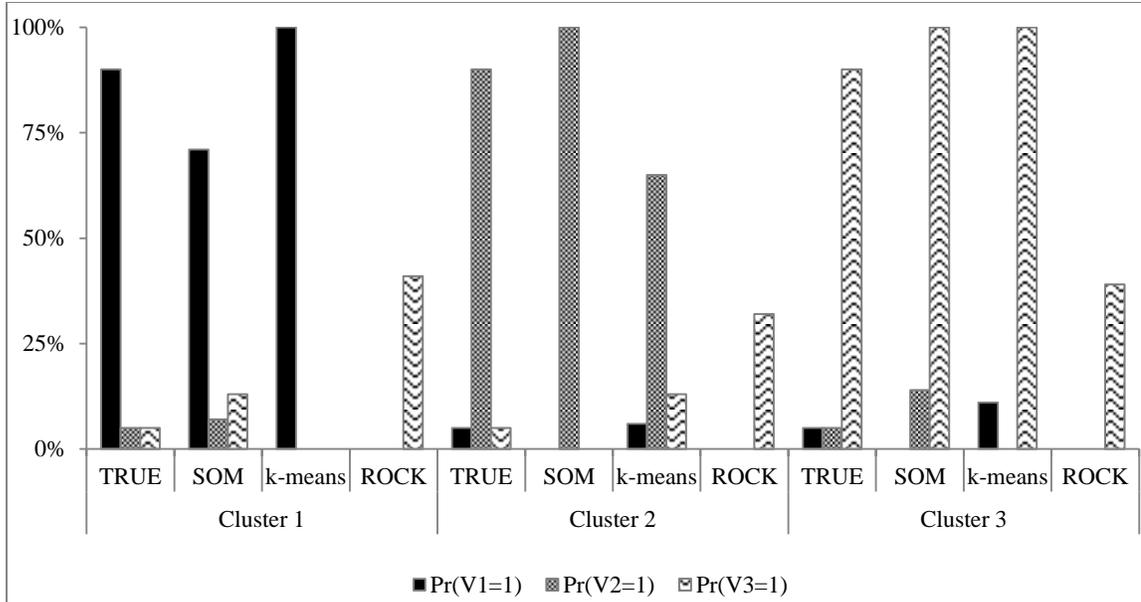


Figure 7a. Cluster center magnitudes (mean coordinates) by variable and clustering solution in S1. The magnitudes of the k -means and SOM centers roughly reflect the probabilities of observing a response in variables v_1 - v_3 in all three clusters, showing that k -means and SOM recover the clusters better than the ROCK algorithm.

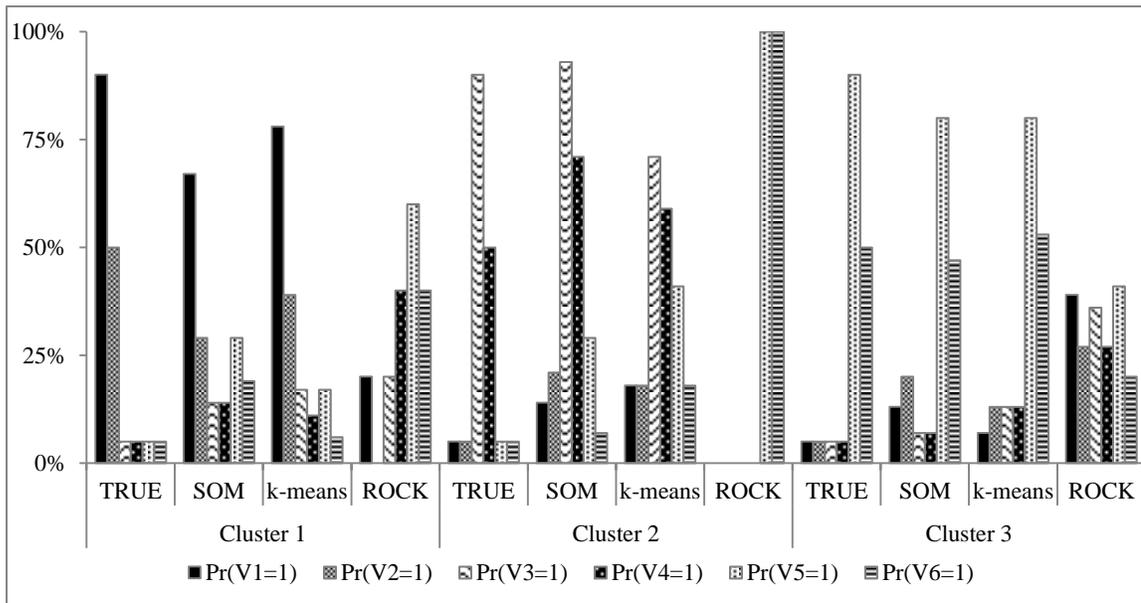


Figure 7b. Cluster center magnitudes (mean coordinates) by variable and clustering solution in S2. The magnitudes of the k -means and SOM centers roughly reflect the probabilities of observing a response in variables v_1 - v_6 in all three clusters, showing that k -means and SOM recover the clusters better than the ROCK algorithm.

The exploration into S1 and S2 shows that SOM and k -means both performed better than the ROCK algorithm in terms of true cluster recovery, fit (as measured by

DBI), and agreement with each other. SOM and k -means each recovered the true clusters equally well in both solutions, improving the true cluster recovery in S2 when more variables were presented. In S1, SOM found a better-fitting solution, whereas k -means found a slightly better solution in S2; however, the fit improvement was not substantial—the SOM solution would probably have the same practical suitability and utility for S2. In general, the improvement in true cluster recovery and fit from S1 to S2 across all three algorithms suggests that each will perform better with more variables.

Performance of the Clustering Algorithms for Datasets M1 and M2

The data in S1 and S2 were created with a known number of clusters, but there is no information about how many clusters are in M1 (the real GBA data from the Number Jumble game). Before comparing how well the clustering algorithms recover the ten simulated Number Jumble responses added to M1 to create M2, one must determine how many clusters are in M1 to begin with.

As discussed in Chapter 2, identifying the number of potential clusters is a challenge, and there is no agreement about the best way to identify the number of clusters in a dataset (Berkhin, 2006; Kriegel et al., 2009; Xu et al., 2013). One simple method is to compare the within sum-of-squares values for clustering solutions over varying numbers of k clusters. One can then plot these values and look for a sharp elbow in the chart, depicting the point at which additional clusters do not significantly lower the within sum-of-squares for the clustering solutions. Unfortunately, doing this with M1 yields no such elbow, suggesting that M1 may be a single cluster solution.

The `mclust` package in R provides a method for estimating the number of clusters by reporting a Bayesian Information Criterion (BIC) value for different values of

k clusters (Fraley & Raftery, 2007). Unfortunately, the package only uses hierarchical clustering that relies on linkages between cases, not a density partitioning method like k -means or SOM, so its recommendations might not be accurate for the algorithms used in this feasibility study. Furthermore, the BIC values are estimated under the assumption that the data are from a multivariate Gaussian distribution (`mclust` varies parameter assumptions for these distributions), but it is safe to assume that the binary data in M1 would not follow a Gaussian distribution. Even so, using `mclust` with the M1 dataset returns results that consistently suggest a one cluster solution across all of their Gaussian distribution parameter combinations.

For large sets of binary data, Newell, Cook, Hofmann, and Jannink (2013) recommend comparing Hubert's gamma statistic across varying levels of k clusters, where the maximum gamma value indicates the number of clusters to seek out in the data (Halkidi, Batistakis, & Vazirgiannis, 2001). The `fpc` package in R calculates Hubert's gamma; however, it does not evaluate whether or not there is a single cluster in the data (Hennig, 2015). Furthermore, the algorithm that calculates the gamma statistic does so by splitting the data into subsamples. Hennig and Liao (2011) recommended using subsamples with at least 200 cases in each. The sample size for M1 ($n = 109$) cannot support such large subsamples, and this becomes evident when trying to use the `fpc` package with the M1 dataset. The `fpc` package failed to reach convergence for the gamma statistic for $k > 4$ clusters, but showed declining gamma values for both SOM and k -means solutions as k increased from 2 to 4 clusters. In the absence of a gamma statistic at $k=1$, one can infer from these declining gamma values that M1 may have either 1 or 2 clusters, but not 3 or 4.

Given the results from comparing within sum-of-squares, BIC, and Hubert's gamma over varying numbers of clusters, it seems that M1 may include either one or two clusters in the Number Jumble data. Classical test theory item difficulty statistics show that the percentage of correct responses for the individual multiplication products in the Number Jumble game ranges from 49%-67%, meaning that about 1/3 to 1/2 of the moves for a given tile in the game are placed incorrectly, but ultimately most people do correctly place most of the tiles in this Number Jumble level. A single cluster solution would be interpreted to mean that most students made some mistakes, but that those mistakes were not systematic across the sample, and nearly every student was able to correctly identify the majority (if not all) of the multiplication products in the game level.

However, exploratory investigation of a two cluster solution using cluster assignments from both SOM and *k*-means shows interpretable results. In both solutions, students in cluster 2 identified the correct multiplication products at a higher rate than students in cluster 1, and they also had fewer "pick up and release" moves where they replace a tile at its starting point instead of picking a product from the grid, suggesting that students in Cluster 2 were somewhat more confident in their moves in the game. These differences may not provide enough discrimination to support a two cluster solution, but the interpretability is noteworthy, so a two cluster solution is at least worth exploring.

As a result of these findings, M1 was analyzed with a two cluster solution across the three algorithms (SOM, *k*-means, ROCK). An analysis of a one cluster solution is not needed, since there is no comparison to be made—all three algorithms would assign all results to the single cluster, and DBI cannot be calculated with $k = 1$. Next, with its

additional ten simulated cases, M2 was analyzed with a two and three cluster solution to see if the ten simulated cases were recovered consistently in the same cluster.

Table 12 shows the consistency between algorithms in M1 when there are $k = 2$ clusters. As was the case in the the results from the simulated datasets, SOM and k -means solutions have higher agreement with each other than they do with the solution found with the ROCK algorithm. In fact, Table 12 shows that SOM and k -means have identical solutions in the M1 dataset (Kappa = 1.00). As was seen with the simulated data, the ROCK algorithm again has a tendency to group most of the cases into a single, large cluster.

Table 12

M1: Cluster Agreement Between Algorithms in a Two Cluster Solution

Cluster	k -means			ROCK			SOM				
	1	2		1	2		1	2			
SOM	1	66	0	k -means	1	65	1	ROCK	1	65	43
	2	0	43		2	43	0		2	1	0
Kappa	1.00			-0.02			-0.02				
% Agreement	100%			60%			60%				

Table 13 shows that this consistency between SOM and k -means dissipates in M2 if only two clusters are specified (percent agreement = 72%, Kappa = 0.33), and ROCK continues to group most cases into one large cluster.

Unless the two different solutions are aligning with two different goals or interpretations, this lack of agreement between SOM and k -means when $k = 2$ could be a validity challenge in practice, perhaps suggesting that one or both solutions are statistically unstable. The SOM solution seems to suggest that there was primarily only one large cluster to begin with in M1, and the addition of the simulated data in M2

created a new cluster, to which three of the original cases were also assigned. The k -means solution presents a different interpretation, suggesting that there were indeed two clusters in M1, and the simulated cases are similar enough to 33 other cases that they should be included in one of the original clusters.

Table 13

M2: Cluster Agreement Between Algorithms in a Two Cluster Solution

Cluster	k -means		ROCK		SOM						
	1	2	1	2	1	2					
SOM	1	13	0	k -means	1	1	45	ROCK	1	0	1
	2	33	73		2	0	73		2	13	105
Kappa	0.33		0.03		-0.02						
% Agreement	72%		62%		88%						

Table 14 shows how the three algorithms compare in M2 when the number of clusters is increased to three. SOM and k -means snap back to the perfect alignment in clusters 1 and 2 that was seen when M1 was analyzed with a two cluster solution. All ten simulated cases are recovered in cluster 3, along with three of the real cases.

Table 15 summarizes the DBI fit values for the three algorithms' solutions for M1 and M2 for the varying value of k (recall that DBI cannot be calculated for $k = 1$). Surprisingly, ROCK was consistently the best-fitting solution. Recall that ROCK achieved this fit by grouping all but one response into a single cluster, thus exemplifying Kriegel et al's (2009) warning that the best-fitting solution is not necessarily the most useful or interpretable.

Table 14

M2: Cluster Agreement Between Algorithms in a Three Cluster Solution

Cluster	<i>k</i> -means				ROCK				SOM					
	1	2	3		1	2	3		1	2	3			
SOM	1	64	2	0	<i>k</i> -means	1	62	1	1	ROCK	1	62	42	13
	2	0	42	0		2	42	0	0		2	1	0	0
	3	0	0	13		3	13	0	0		3	1	0	0
Kappa	1.00				-0.02				-0.02					
% Agreement	100%				52%				52%					

Table 15 also shows that SOM and *k*-means had the same DBI value (3.02) for M1 with $k = 2$ because they created the same solution, but SOM had better fit in M2 for $k = 2$ and $k = 3$ (DBI = 1.86 and 1.82 respectively). Recall that when $k = 3$, cluster assignments for SOM and *k*-means in M2 were identical. The difference in DBI shows that SOM achieved this by selecting better fitting cluster centers, even though both algorithms yielded the same cluster assignments.

Table 15

DBI Results by Solution for M1 and M2

Algorithm	DBI for M1		DBI for M2	
	$k = 1$	$k = 2$	$k = 2$	$k = 3$
SOM	—	3.02	1.86	1.82
<i>k</i> -means	—	3.02	3.02	2.99
ROCK	—	0.80	0.84	0.67

Discussion of the Feasibility Study

The simple feasibility study offered a glimpse into how the SOM, *k*-means, and ROCK algorithms perform under different data scenarios, similar to those found in GBA response data. One can evaluate the findings by considering the algorithms' abilities to

recover true clusters (as measured by Kappa) and the fit of their solutions (as measured by DBI).

In S1 and S2, SOM and *k*-means were able to make good to near-perfect recoveries of the true clusters, as represented by their Kappa values (Landis & Koch, 1977). SOM and *k*-means performed similarly in S1 and S2 with the cleaner, completely simulated data. SOM and *k*-means created the best fitting solutions for S1 and S2, and since their solutions were so similar, there were not vast differences in their DBI values for the S1 and S2 solutions. ROCK tended to lump many cases together into one large cluster in S1 and S2 and yielded poorer DBI results for fit.

For M1, SOM and *k*-means yielded identical clustering solutions when $k = 2$ clusters. When the simulated cases were added to create M2, SOM and *k*-means diverged, with *k*-means creating similar partitions as were found in M1 and SOM isolating the simulated cases (along with three others) at the expense of combining all other cases into a single cluster. When k was increased to 3 for M2, SOM and *k*-means realigned, matching their two clusters from M1, and correctly clustering the ten simulated cases along with three of the real cases, although SOM's centers yielded a better DBI value. Inspection of those three cases showed that they each had observations for one or two of the three variables used to define the simulated cases: variables 2.1.1, 3.2.1, and 6.3.3. This suggests that these students may have made the addition instead of multiplication error that the simulated cases were meant to mimic.

The results of the feasibility study show that *k*-means and SOM may yield similar solutions in some scenarios, in which case they can be used in tandem to validate each other's solutions; however, there are situations like M2 where SOM outperforms *k*-means

in terms of fit (DBI). There may conceivably be cases where the reverse is true. The results from the Number Jumble data suggest that M1 and M2 may in fact have two and three clusters, respectively, and if the researcher specifies the incorrect number of clusters (e.g., $k=2$ for M2), SOM and k -means may arrive at drastically different results.

ROCK performed very poorly in the feasibility study, but this may be a factor of the high sparseness of the data and the low number of clusters: with so many zeroes many cases share similarities, which then create a large number of links between all responses, thus amalgamating all of the results into a single cluster when a low number of clusters (two or three) is desired. ROCK did produce better DBI values in the M1 and M2 solutions, but this may be because the ROCK solution had clusters where $n = 1$, which would lead to zero within cluster variance for those clusters, which in turn might lower the DBI.

Guha et al. (2014) and Song and Li (2006) have demonstrated that ROCK recovers known clusters well in higher dimensional binary datasets (16-548 variables), even when there are a low number of clusters, and Song and Li showed that ROCK's clusters were as good as or better than k -means clusters when creating sets of news articles that had been marked with binary topic tags. Despite the findings of the feasibility study, ROCK may still be a suitable algorithm when working with more variables.

Chapter 4: Methods for Full Study

The full study leverages the findings of the feasibility study to compare the performance of clustering algorithms in data that reflect common GBA response data; i.e., high-dimensional, sparse, binary data. The study seeks to evaluate how well SOM, k -means, and ROCK algorithms identify true latent clusters under varying conditions representing aspects of gameplay, including game state, autonomy, and relatedness. These conditions are explored by creating datasets which vary the numbers of clusters in the data, the number of cases in a cluster (cluster size), the probability of observing responses for some relevant variables in a cluster, and the amount of overlap between the variables that are relevant to defining the clusters.

The study seeks to illustrate the conditions under which recovery of true clusters yields Kappa values greater than 0.80 for a variety of GBA data scenarios—0.80 Kappa representing a commonly used threshold for “near perfect agreement” (Landis & Koch, 1977). With this goal in mind, the study addresses the following research questions:

1. Given the correct number of true clusters, k , what are the magnitudes of the differences in mean Kappa values for the SOM, k -means, and ROCK clustering solutions across multiple datasets as:
 - a. the number of clusters increases?
 - b. the number of cases in a cluster increases?
 - c. the number of cases in clusters are similar?
 - d. the probability of observing responses for relevant variables in a cluster increases?

- e. the amount of overlap between the variables that are relevant to defining the clusters increases?
2. Under what combinations of these conditions is the true cluster recovery poor (Kappa distributions generally below 0.80) for each algorithm, even when the researcher has correctly specified the number of clusters?
 3. Under what combinations of these conditions are the SOM, *k*-means, and ROCK solutions expected to agree with each other ($\text{Kappa} \geq 0.80$) so that they can be used to validate each other?
 4. When varying the number of clusters specified for an algorithm, does the best-fitting solution for the SOM, *k*-means, and ROCK algorithms (as measured by DBI) match the correct number of clusters in the data?

By answering these questions, this study helps researchers think about how to design GBAs to encourage response data that will be suitable for their desired accuracy of true cluster recovery, potentially by avoiding game designs that can result in data structures that yield poorly separated clusters of gameplay. For example, if overlapping relevant variables make it difficult to identify clusters, GBA designers may design the game with less autonomy to create less overlap in gameplay, or they may choose to omit variables that are observed in many students' response data.

For existing GBAs, this study provides guidance about the suitability of these algorithms for classifying student responses. Recall that these clustering algorithms do not help the researcher identify the number of clusters—a process that typically involves researchers comparing the interpretability of solutions across varying values of *k*—but this study demonstrates under which conditions these algorithms fail to recover true

clusters at an acceptable rate, even when the researchers have specified the correct number of clusters. The study also shows if better fitting solutions are found when k is misspecified and does not match the true number of clusters.

This study requires simulated data because these conditions cannot be easily generated in a real GBA scenario. Given a real GBA, there would be no way to ensure students were creating a desired number of response clusters without actually coaching them through every step of their gameplay. Crossing all five conditions from the first research question would be nearly impossible, and there would be logistical issues of getting students to participate in all of these variations. The time commitment and cost of recruiting such a sample would be prohibitive. A simulation allows for targeted manipulation of the response data conditions needed to address the research questions.

To help illustrate how the guidance from the study output relates to a real world application, the SOM, k -means, and ROCK algorithms have also been used to cluster data from the game Beanstalk (Carnegie Mellon, 2013). This example is intended to serve as a tutorial, discussing not just the specifics of the analyses, but also the step-by-step process of picking algorithms, coding log file data, comparing solutions, validating results, and interpreting the outcomes. The following sections discuss the design of the simulation, the study's output variables, and the tutorial structure in more detail.

Simulation Design Summary

The study simulates responses from a GBA in which each cluster is differentiated by having a higher probability of observed responses for certain variables in the dataset. Data are generated and analyzed 400 times for each cell in the study, meaning that each cell has 400 replications, each with hundreds or thousands of simulated response patterns,

depending on the number of clusters and cluster sample sizes specified for the cell. Since each cell has 400 datasets, 400 values for Kappa and DBI can be calculated in a given cell for each clustering algorithm. This sample size ensures more than 80% power to detect small effects between the average Kappa values and average DBI values for any two algorithms' solutions within a cell or between two cells. 400 datasets is also a multiple of 100, which helps simplify communication of the percentage of solutions that have certain characteristics (e.g., Kappa greater than 0.80).

The study varies four aspects of the data. The first three aspects are directly manipulated, varying the number of clusters (k), the number of cases in a cluster (cluster size; N_k), and the probability of observing responses in a cluster. These conditions create a 48-cell design ($4 \times 4 \times 3$) such that each dataset will have $k \times N_k$ cases (observed work process response patterns) and $k \times 20$ response variables. The fourth aspect, the amount of overlap between the relevant variables defining the clusters, also varies by allowing the 20 relevant variables for each cluster to be randomly selected, thus varying the between cluster variability. The following sections describe, in detail, the data generation, the parameters manipulated, the rationale for the specific parameter values, and the methods used in the simulation and the tutorial study using game data from Beanstalk (Carnegie Mellon, 2013; Koedinger et al., 2010).

Data Generation

Each of the 400 datasets in each cell of the study were generated using a latent class analysis (LCA) data simulator in the `poLCA` package in R (Linzer & Lewis, 2011). LCA models predict the probability of a student belonging to a given group, given the student's response pattern and the probabilities of responses on each variable for a given

group (cluster). Similarly, when generating simulated data in the `p0LCA` package, one must specify the number of clusters, the number of variables, the probability of being in a given cluster, the total sample size, and the probability of an observed response on each variable for each cluster.

For this simulation, the `p0LCA` data simulator was set to have k clusters, depending on the cell parameters. Each cluster is distinguished by simulated student responses on 20 variables in which there is a higher probability of a response for cases in that cluster. These 20 variables are considered “relevant” to the identification of the cluster, but some of these variables may also be used in the identification of multiple clusters (i.e., the same variable may be relevant in the identification of several clusters). To create these variables, an initial pool of $k \times 20$ variables was created. Each cluster had 20 variables randomly selected from the complete set of variables (without replacement) as their relevant variables (e.g., for four clusters, there are $4 \times 20 = 80$ total variables in the dataset, and each cluster has 20 of those 80 variables randomly assigned to be the clusters’ relevant variables). The probability for a response on each cluster’s 20 relevant variables was set to 0.30, 0.60, or 0.90, depending on the cell. The remaining variables were set to have a probability of 0.05. Each cluster’s 20 variables were picked independently, meaning that clusters may (and are expected to) share some of the same relevant variables. Thus, two clusters may each have a high probability of a response on the same variable.

The sample sizes for each of the k clusters was picked from a normal distribution with a mean of 50 or 100 and a variance of 9 or 100 using the `rnorm` function in R (R Core Team, 2014). The total sample size is simply the sum of the cluster sample sizes.

For the generation of the simulated data, the probability of a respondent belonging to a given cluster is simply the cluster's sample size divided by the total sample size: N_k/N . For example, if a cluster has a sample size of 50, and there are 200 cases in the dataset, then the probability of a respondent being in that cluster is 0.25.

Number of Clusters for the Simulation

Having more clusters in the dataset represents more autonomy in the game; i.e., there is more variation in the possible response patterns. In theory, a higher autonomy game can yield a higher variety of response patterns, which could correspond to more clusters of interpretable work processes. For example, the earlier feasibility study of the Number Jumble data suggested that there may be only two clusters of work processes in the data, but Number Jumble is a low autonomy game. The game Save Patch has moderately higher autonomy than Number Jumble, and in Kerr et al.'s (2011) fuzzy cluster analysis of the Save Patch response data, they determined each Save Patch level had seven or fewer clusters. They determined the number of clusters by increasing k until a cluster existed that contained only correct response patterns (Kerr & Chung, 2012). In a biology game with higher autonomy than Save Patch, Stevens and Casillas (2006) identified 36 clusters of responses using the SOM algorithm. In a similar study of brain state patterns of submarine pilot teams in a simulation (not a GBA), Stevens et al. (2012) identified 25 clusters of team brain state patterns, although these were inputs into a larger student model, and the clusters were not used individually to support inferences.

For the simulation conditions, there are four initial possibilities for the number of clusters (k) in each dataset: 4, 10, 20, and 35 clusters. These choices roughly mirror what has been observed in these other studies: ranging between a handful of distinct work

process clusters up to a few dozen. The highest number of clusters ($k = 35$) illustrates how these clustering algorithms might perform in an extremely high autonomy GBA. This is a reasonable inquiry, as some researchers have begun wrapping assessments around existing video games that have complex gameplay (e.g., Corrigan et al., 2014).

Number of Cases in the Simulated Clusters

In some GBAs, some work processes may be less common than others, and this will manifest in clusters with fewer cases than others. Smaller clusters may represent unexpected response patterns, and small clusters are a likely possibility in GBAs with high autonomy or where game state dependencies might limit the prevalence of certain response patterns. Of course, the number of cases in a cluster will also depend largely on how many students play the GBA, so for this simulation, the average cluster size and the variability in cluster size were both manipulated.

Varying the average number of response patterns (cases) in the simulated clusters helps identify whether certain clustering algorithms perform better as cluster size increases (i.e., does an algorithm recover true clusters better when clusters are very large?). Allowing for variability in cluster size within a dataset also demonstrates whether certain algorithms perform better in scenarios where some work processes are less common than others. This is of particular interest because variability in cluster size is likely common in GBAs; however, the k -means algorithm assumes that all clusters are roughly the same size (SOM and ROCK do not have this assumption).

Neither Kerr and Chung (2012) nor Stevens and Casillas (2006) reported how many cases were in each individual cluster, but an estimate can be made from the aggregations provided in their papers. For example, in a level with 900 total response

patterns, Kerr and Chung (2012) reported that three clusters represented 68% of those cases, thus one can infer that those three clusters had an average size of 204 response patterns. Similarly, given that Stevens and Casillas (2006) had 1,710 total responses and 36 clusters, the average cluster size was about 48 response patterns. These studies provided context for the choice of clusters sizes in this simulation study.

Each cell of the study has either smaller clusters defined as having an average cluster size of 50 cases, or they have larger clusters with an average size of 100 cases. It is expected that these results will generalize to larger cluster sizes, as the clustering algorithms are expected to perform better when clusters have even more cases. The variance of the cluster sizes is also manipulated in the study. Clusters either have a similar size ($\sigma^2 = 9$ cases) or have variability in cluster sizes ($\sigma^2 = 100$ cases). This implies that when clusters are similarly sized, clusters will not differ in size by more than 18 cases, but when they vary in size, they may differ in size by as much as 60 cases. The study results are then able to show whether clustering algorithms' recovery of true clusters changes as average cluster size goes up and when there is more variability between cluster sizes.

Data Type and Response Probabilities in Simulated Clusters

The work process data consist solely of binary data. For brevity, a value of 1 is called a *response*, but it could represent any dichotomous variable of interest in a game, such as an observed student action, a set of actions, the game state, or activation of a game connection. Binary data were used to represent structured-logging of data, meaning that game designers or researchers have tagged log file data with labels or codes that can be represented as binary indicators (Kerr & Chung, 2012). This is a common way to

organize GBA data to make it more manageable for analysis, which is why binary data were chosen for the simulation, and it is worth the reminder that the ROCK algorithm can only work with binary data. Nevertheless, it should be noted that GBA data do not need to be coded into binary data. Structurally logged data can also contain continuous measures (e.g., representing elapsed time, sequences of steps), and it is conceivable that some GBA work processes could be analyzed directly from the log file data without any additional tagging.

The probability of observing responses within a cluster manipulates the within-cluster variability, which helps determine if certain algorithms are better at clustering results that have some dissimilarity within a cluster. This aspect of the data also relates to autonomy (freedom in gameplay), in that students may be engaging in the same general work process, but the game design may allow for some flexibility or alternatives, even though the overall approach of the work process is the same. Of the studies reviewed here, only Stevens and Casillas (2006) touched on the relevancy of certain variables in defining a pattern. In sample diagrams, they showed bar charts from two of their 36 clusters, illustrating the selection frequency for each variable. For example, one of their clusters showed that the response patterns nearly always included observed responses from five of the variables in the game and almost never had responses on many of the other variables. Similarly, for this simulation clusters were created by having a higher probability of response in a subset of 20 variables in each simulated dataset.

To manipulate within-cluster variability, clusters were generated with low, moderate, and high probabilities of an observed response within their 20 relevant variables, as discussed in the Data Generation section. For the other variables not

manipulated in the definition of a cluster, the response probability was 0.05, meaning cases in a cluster have a very low (but not zero) probability of an observed response on the remaining non-relevant variables, thus creating a high probability of *not* observing responses in these variables and ensuring that they were largely dissimilar to the cluster's 20 relevant variables. This also means that this 0.05 probability of a response on non-relevant variables was shared across many clusters, thus making any given variable extremely redundant and irrelevant for most clusters' interpretations unless it is assigned to a cluster in which it has a high probability of a response.

One limitation of the simulation study is that there is no theory supporting the levels of the probability of a correct response used to create the simulated data. Using values of 0.30, 0.60, and 0.90 provides a wide range of probabilities which helps show how the clustering algorithms perform when there is higher within-cluster variability, but there remains the question of whether a 0.30 response probability actually is relevant to interpreting a cluster in practice. Stevens and Casillas (2006) looked at response frequencies by variable when evaluating the interpretability of their clusters, suggesting—at least for them—that an inconsistent presence of responses within a cluster might be grounds for discarding the cluster solution. In practice, researchers may be more interested in clusters where cases have high similarity or where there are correlations between the responses across multiple variables in a cluster, but these variations have been included to show how the algorithms perform as within-cluster variability changes.

Amount of Overlap Between Clusters

The previous section described the probability of observed responses in the variables that were considered relevant for differentiating one cluster from the others.

Allowing clusters to share these defining variables decreases the between-cluster variability because multiple clusters had high probabilities of responses on a set of overlapping variables, which is a likely scenario in practice. Stevens and Casillas (2006) showed that response probabilities for some variables were very similar in some (but not all) clusters, suggesting that those overlapping variables were important for the definition and interpretation of those subsets of clusters; however, overlap decreases between-cluster variability, which can make it more difficult to identify clusters.

Artificially creating overlap between clusters in a structured manner is a difficult task. One must consider whether a cluster shares relevant variables with multiple clusters, and if so, how many variables? Is a variable shared with just one other cluster, or many? Is the degree of overlap the same between all clusters, or is one cluster allowed to be completely unique?

Rather than trying to control these many aspects of overlap, it is simpler to let the clusters share relevant variables randomly. Each cluster had 20 variables randomly selected to have a higher response probability, and there is the expectation that each cluster would likely share at least one relevant variable with each of the other clusters. This creates a degree of overlap or similarity between the clusters—the more overlap, the less between-cluster variance there could be. Additionally, overlapping relevant variables were more prominent when there were fewer clusters and thus fewer variables. This is an important condition to simulate because it allows for the fact that many interpretively-distinct response patterns may share some features in log data; i.e., students across clusters likely share some of the same gameplay steps. It also depicts that the potential for that overlap decreases in games where there is more autonomy and thus the potential for

more distinct clusters, as well as more variables that can be included in the analysis. Allowing the clusters to randomly share some relevant variables provides a view into whether clustering algorithms work better as between-cluster variance increases.

Allowing clusters to overlap in their relevant variables helps illustrate how GBA data may be impacted by GBA design. When a subset of clusters has higher response probabilities in a shared set of variables, this can be representative of game state. For example, a specific initial work process or activation of a game element might be needed before a secondary work process can occur. If multiple clusters share that initial work process response pattern, then they might all be seen as co-dependent on those defining variables. Returning to the soccer example from the Rules section in Chapter 2, two clusters might share the defining variable of having the ball go into the goal, yet differ in the variables representing whether or not that occurred before or after the game clock ended (game state). Similar situations might occur in scenarios where a researcher tracks the order of actions; e.g. two clusters share the same variables representing actions in students' work processes, but they have differences in the variables representing the sequence in which those actions were conducted.

When *all* clusters have similar response probabilities in a shared set of variables, then this can be viewed as a representation of irrelevant variables, which might occur with the addition of relatedness elements in a GBA. For example, if bonus points are awarded to students who provide responses quicker in a game, even though speeded response is not a feature of the construct of interest, then the frequency of observed bonus points in the response data may not be a relevant variable in the definition of the clusters; i.e., in each cluster, some people got bonus points, and some did not. The bonus points

are irrelevant for interpretation of any cluster, which in turn decreases the between-cluster variability across all of the clusters.

Number of Response Variables in the Simulation

Interpretation of a student's work process is largely dependent on how his or her response pattern differs from other students' response patterns, so an interpretable cluster might have a unique probability of an observed response on a set of variables considered "relevant" to identifying that cluster. As mentioned above, each cluster had 20 relevant variables that have a higher probability of an observed response. The choice of 20 relevant variables per cluster is arbitrary and represents a limitation of the study. In practice, the number of variables that might be considered relevant to identifying a work process will likely vary between clusters and be heavily dependent on a GBA's design and the researcher's informed judgment. Furthermore, the number of variables can correspond to every action a student can make in the game, but it can be extended to include factors like the game state and sequence. The researcher makes decisions about which of these factors are relevant for clustering student responses. Nevertheless, having 20 variables that all have a higher probability of a response for a cluster can be considered a fairly complex game, and it is expected that the algorithms' performance in this simulation would generalize (or perhaps even improve) for GBAs with even more relevant variables for each cluster.

In the interest of keeping the number of varying simulation conditions to a minimum, the number of relevant variables is fixed at 20 variables for all clusters in the study. Despite this one limitation, 20 relevant variables for each cluster still result in a large number of total variables because the number of clusters (k) varies, and the total

number of variables was set to $k \times 20$. This creates enough total response variables so that there is a reasonable level of overlap in the 20 relevant variables in each cluster. Fewer total variables would increase the overlap, potentially to the point that there would be few distinguishing features of response patterns in different clusters and making many variables irrelevant due to having the same probability of a response across all clusters. Having more response variables would dilute the data with variables with equally low response rates across all clusters—essentially adding irrelevant variables. Table 16 shows the total number of GBA variables that exist in each simulated dataset by the number of clusters.

Table 16

Total Number of Variables by Number of Clusters

<i>k</i>	Total Number of Variables
4	80 variables
10	200 variables
20	400 variables
35	700 variables

The resulting range of total variables shown in Table 16 correspond to a reasonable range of the number of variables that might be used in a GBA cluster analysis. This in turn provides some justification for the choice of 20 relevant variables for each cluster. For example, Kerr and Chung (2012) clustered responses using only observed student actions from the Save Patch game and reported that their more complex, difficult game levels had 100-200 variables used in the cluster analysis. Kerr and Chung's game with 200 variables may seem high, but higher dimensionality may exist for a deceptively simple game. Stevens and Casillas' (2006) biology game identified clusters out of 2,862 variables. Mislevy et al. (2014) noted that game mechanics go beyond just the tools and

buttons that appear onscreen. Game mechanics encompass every possible action that students can take using those tools, so it would not be uncommon for GBA researchers to be working with hundreds of variables.

Given the large number of total variables that result from having 20 relevant variables for each cluster, there is an expectation for improved performance of the ROCK algorithm, as compared to its poor performance in the feasibility study. Recall too that game autonomy (the amount of freedom of actions a student has in the GBA) will increase the number of variables that might be available for analysis, but as autonomy increases, the results data will tend to be very sparse (Kerr & Chung, 2012). Stevens and Casillas' (2006) game may have had 2,862 variables, but some of the most efficient strategies could solve the game with fewer than ten of these variables. Thus the varying number of clusters and the corresponding changes to the total number of sparse variables in each dataset can be seen as a realistic representation of varying game autonomy.

Methods and Outcome Measures

Table 17 shows how the 48 cells of the simulation study are organized using the conditions discussed in the preceding sections. Recall that 400 datasets were generated in each cell, and each dataset had roughly $k \times N_k$ cases representing simulated student responses.

To address the first three research questions and discover how mean Kappa values differ between cells and when algorithms are expected to return Kappa values of 0.80 or higher, the 400 datasets in each of the 48 cells were evaluated with the SOM, k -means, and ROCK algorithms. The algorithms were instructed to identify k clusters in the data, where k is the known, true number of clusters in the data. Recall that researchers typically

choose the number of clusters based on the interpretability of clustering solutions that vary by k (e.g., Stevens & Casillas, 2006; Kriegel et al., 2009; Kerr & Chung, 2012), so by specifying the true k in the simulation study, these results demonstrate whether the algorithms perform poorly even when the researcher has the correct number of clusters.

Table 17

Simulation Design of Four Varied Parameters Resulting in 48 Cells

$k = 4$ clusters					$k = 10$ clusters				
Cluster Sizes N_k					Cluster Sizes N_k				
$Pr(1)$	$\sim N(50,9)$	$\sim N(50,100)$	$\sim N(100,9)$	$\sim N(100,100)$	$Pr(1)$	$\sim N(50,9)$	$\sim N(50,100)$	$\sim N(100,9)$	$\sim N(100,100)$
0.30					0.30				
0.60					0.60				
0.90					0.90				
$k = 20$ clusters					$k = 35$ clusters				
Cluster Sizes N_k					Cluster Sizes N_k				
$Pr(1)$	$\sim N(50,9)$	$\sim N(50,100)$	$\sim N(100,9)$	$\sim N(100,100)$	$Pr(1)$	$\sim N(50,9)$	$\sim N(50,100)$	$\sim N(100,9)$	$\sim N(100,100)$
0.30					0.30				
0.60					0.60				
0.90					0.90				

Note: $Pr(1)$ is the probability of an observed response on the 20 relevant variables randomly assigned to each cluster.

The k -means algorithm was set to run through 2,000 cycles, thus allowing it to pick a solution that minimizes the within sum-of-squares distances for the clusters. Similarly, each dataset was presented to the SOM algorithm 2,000 times, allowing the SOM algorithm to iteratively update the cluster centers with each presentation of the dataset. The ROCK algorithm was run with a theta value of 0.95, but if it could not

converge on k clusters, theta was lowered in 0.05 increments until the ROCK algorithm could produce exactly k clusters.

The distance measure used for all three algorithms was Euclidean distance. This was partly a necessity for comparisons—the SOM algorithm in the `kohonen` package in R currently only permits Euclidean distance (Wehrens & Buydens, 2007). The R packages for ROCK and k -means allow for other distance measures, but they default to Euclidean distance. When using binary data, Euclidean distance can be a problem for describing differences between cases because there is the potential for ties. For example, (0,1) and (1,0) are equidistant from (0,0) and (1,1). This becomes less of a concern when there are more variables in the data, which is one reason why it is beneficial that many of the cell conditions yield several hundred variables for each simulated dataset.

For each cell of the simulation, Cohen's Kappa was used to evaluate the true cluster recovery rate for all three algorithms, thus showing how well each algorithm was able to correctly group cases from each cluster. These statistics were used for comparisons of correct classification rates between the three algorithms under the varying conditions and parameters of the simulation. Since each cell will have 400 datasets, the distribution of Kappa values can be shown for each algorithm in each cell.

Each dataset also has a between-cluster variability measure of overlap calculated using the root mean square deviation (RMSD) of the response probability vectors for each cluster. In other words, each cluster's vector of response probabilities for all variables were compared to the overall average probability of someone having a response on a variable. This is comparable to measuring the deviation of clusters' expected centers from the global mean. When the RMSD is smaller (within a cell), it indicates more

overlap between the clusters' relevant variables; i.e., less between-cluster variability. Conversely, when the RMSD is larger within a cell, it indicates higher between-cluster variability. RMSD values cannot be meaningfully compared between cells since their magnitudes depend on the probabilities of responses and the numbers of clusters. RMSD is calculated using the observed cluster centers and the standard RMSD formula:

$$\sqrt{\frac{\sum_{c=1}^k (\mathbf{m}_c - \bar{\mathbf{m}})^2}{k}} \quad (10)$$

where \mathbf{m}_c is the center coordinates of cluster c and $\bar{\mathbf{m}}$ is the global mean of all k clusters' center coordinates.

The first research question seeks to determine if there are differences in mean Kappa values for the three algorithms under the varying conditions of the simulations (when k is correctly specified). The first research question was investigated by plotting mean Kappa values across manipulated conditions in the simulation, and correlations between RMSD values and Kappa within a cell show if higher between-cluster variability makes it easier to recover true cluster assignments. The second research questions was investigated by plotting distributions of Kappa values for each cell to illustrate the conditions under which each algorithm is expected to achieve Kappa values of at least 0.80. These distributional plots also address the third research question by showing when two algorithms fail to consistently yield Kappa values of 0.80 in a cell, meaning that their solutions would likely not validate each other under similar conditions in practice.

To address the fourth research question about whether the algorithms' best fitting solutions match the true number of clusters, the algorithms were run with a variety of k values ranging from $k_t - 3$ to $k_t + 3$, where k_t is the true number of clusters. The

distribution of DBI was plotted for each number of clusters for all three algorithms across the cells of the simulation study, categorized by whether the best-fitting solution overestimated, underestimated, or accurately identified the correct number of clusters. This illustrates whether the best-fitting solution for each algorithm reflects the true number of clusters used to simulate the 400 datasets in each cell. Kappa and percentage correct were not calculated for the clustering solutions when k was misspecified since the algorithms were not expected to correctly recover cluster memberships when the wrong number of clusters was provided.

The goal of comparing the fit at varying specifications of k in the fourth research question is to see if any algorithms routinely produce better-fitting solutions with a misspecified value of k or if DBI consistently tends to overestimate or underestimate the true number of clusters. Note that there are multiple methods for determining the number of clusters beyond DBI, but it is outside the scope of this study to compare them or to see if any are suitable methods for identifying the true number of clusters. Furthermore, some GBA researchers appear to prefer selecting the number of clusters based on interpretability rather than fit (e.g., Kerr et al., 2011; Stevens & Casillas, 2006). That being said, DBI was selected as the fit measure for comparisons in the fourth research question because it has been shown to be one of the best algorithms for identifying the number of clusters in binary data (Dimitriadou, Dolnicar, & Weingessel, 2002). DBI also has the added benefit of allowing for fit comparisons between algorithms' solutions.

Tutorial Example

The simulation study described previously serves to examine how the three algorithms perform under a variety for a variety of data scenarios, but the tutorial study is

intended to serve primarily as an example and tutorial for how GBA researchers should go about identifying and comparing clustering solutions as part of the validation of clustering algorithms as measurement models in the CFA. This tutorial includes examples of how to compare alignment between solutions from competing algorithms, select the potential number of clusters, compare fit, and evaluate differences in solutions in the context of the data and the algorithms themselves.

For the tutorial study, the k -means, SOM, and ROCK algorithms were used to classify student response patterns from the different levels in a mathematics game called Beanstalk (Carnegie Mellon, 2013). The tutorial study uses anonymized students' log file data from the 'Beanstalk_2013_05' dataset accessed via DataShop (Koedinger et al., 2010). The tutorial discusses the considerations in selecting algorithms and coding variables prior to analysis. For each Beanstalk game level, clustering solutions were calculated for varying numbers of k clusters, and DBI values were calculated for each solution. The retained clustering solution for each game level was selected based on both fit (as measured with DBI) and agreement between algorithms (as measured with Cohen's Kappa). The solutions were validated against a holdout sample and an external criterion—in this case, indicators of whether or not the student solved the level correctly. Finally, the tutorial discusses the process of labeling and grouping clusters for interpretation, as well as using changes in cluster membership to visualize student progress across the game.

To summarize, the full study consists of two parts: the simulation and the tutorial example. The simulation generated datasets with a known true number of clusters under varying conditions of the number of clusters, the size of the clusters, the probability of

observing responses in relevant variables in clusters, and the amount of overlap between clusters' relevant variables. These datasets were analyzed with the k -means, SOM, and ROCK algorithms to see how often they are able to adequately recover the true clusters, as determined with Kappa measures of agreement between cluster solutions and true cluster identifiers. The simulation study then used the same conditions but then used DBI to evaluate the number of clusters associated with the best-fitting solution (e.g., if the true number of clusters is 10, is an algorithm's best-fitting clustering solution also associated with $k = 10$, or does it over- or under-estimate the number of clusters needed?). The second part of the full study is the tutorial, which illustrates how one might implement a clustering algorithm in a GBA, walking through the steps and practical considerations from data identification and coding through cluster interpretation and reporting.

Chapter 5: Simulation Study Results

The simulation was run in R (R Core Team, 2014), and the final study scripts are provided in Appendix A. The simulation ran for approximately three months, and during that time, some adjustments were made to the code based on performance observations. It is important to note these changes here, as they are mentioned in the discussion of the results; however, none of these changes had an impact on the methods and outcome measures proposed in Chapter 4.

Early in the analysis, it was observed that the SOM algorithm occasionally did not return the correct number of clusters (k). For example, in the initial cells, the study was set to analyze $k = 4$ clusters, but the SOM algorithm occasionally returned only three clusters. This was occurring in approximately 1% of the results when the probability of a response in the relevant variables was 0.6 or 0.9. This phenomenon did not appear to be related to overlap between clusters' relevant variables, as measured by RMSD.

Further investigation suggested that this was occurring as a result of poor, randomly selected, starting centers for the SOM. Recall that (unless directed otherwise) the SOM algorithm selects its starting centers randomly from the observed data. If two of those centers are extremely similar, those clusters will be competing for the same cases, switching case assignments between the two clusters back and forth and eventually having nearly identical center coordinates. At the end of the data presentation iterations, the cases are all assigned to whichever center (node) was active when the algorithm stopped.

Similar issues can occur in k -means, where poor starting centers can lead to bizarre clustering solutions, but recall that the k -means algorithm in R runs itself multiple

times and selects the outcome that minimizes the within sum-of-squares (R Core Team, 2014). The SOM algorithm, by comparison, presents the data to the model over and over to keep fine tuning the centers as the neighborhood radius decreases, but the model never goes back to try the process with new starting centers. Thus the SOM algorithm still has a risk of returning a poor cluster partition, and savvy researchers should run try the SOM algorithm multiple times with different starting points to compare solutions (e.g., Amershi & Conati, 2009).

Despite this observation, the simulation was allowed to run as proposed, with the modification that if the SOM algorithm returned fewer clusters than the k clusters specified by the simulation cell, it was rerun with new starting centers. Recall that the researcher specifies the number of clusters required for analysis, and this was a necessary change to allow for proper calculation of Kappa when calculating the true cluster recovery. It also seemed like a reasonable correction to make, since in practice, a researcher who requested a k cluster solution would likely not accept a solution that produced fewer clusters.

Another observed issue was that in some datasets, the ROCK algorithm could not converge on the specified number of clusters. This only occurred when $k = 35$, $N_k = 50$, and probability of a response on the relevant variables was 0.3. In other words, the ROCK algorithm had difficulty converging on the desired number of clusters when there were a lot of small clusters with low response rates in the relevant variables. When $k = 35$, $N_k = 50$, probability of a response on the relevant variables was 0.3, and cluster size variance was nine, 397 datasets were created, and the ROCK algorithm converged in 24 (6%) of them. Similarly, under the same data conditions but with cluster size variance set

to 100, 391 datasets were generated, and the ROCK algorithm converged in only 34 (9%) of them.

The number of datasets mentioned in the preceding paragraph hints at the final issue, which was time. Despite running for three months, not all cells could be run with the desired number of datasets by the end date of the project. Datasets with more cases and more variables took more time to both simulate and analyze, and several cells did not have the 400 datasets desired. Nevertheless, the results are still interpretable and the trends are evident with the datasets that were completed. The study cells had the following final counts for simulated datasets as of the study's completion (Table 18):

Table 18
Actual Number of Datasets Generated per Cell

k	Size	Pr(1)		
		0.3	0.6	0.9
4	$\sim N(50,9)$	400	400	400
	$\sim N(50,100)$	400	400	400
	$\sim N(100,9)$	400	400	400
	$\sim N(100,100)$	400	400	400
10	$\sim N(50,9)$	400	400	400
	$\sim N(50,100)$	400	400	400
	$\sim N(100,9)$	400	400	400
	$\sim N(100,100)$	400	400	400
20	$\sim N(50,9)$	400	400	400
	$\sim N(50,100)$	400	400	400
	$\sim N(100,9)$	400	400	400
	$\sim N(100,100)$	400	400	400
35	$\sim N(50,9)$	397	400	400
	$\sim N(50,100)$	391	400	230
	$\sim N(100,9)$	140	400	120
	$\sim N(100,100)$	140	400	130

Research Question 1: Kappa Differences by Varying Individual Conditions

The first research question investigates the differences in mean Kappa values between the true clusters and the k -means, SOM, and ROCK algorithms when they are given the correct value of k under each of the individual conditions manipulated by the simulation study. It is critical to underscore that the first research question looks at individual aspects of the data manipulation in the simulation study, aggregating mean Kappa values that were calculated across many conditions. This is effectively looking at marginal effects, but this research question does not consider interaction effects. For example, the first research question investigates if mean Kappa values change as the number of clusters increases, without consideration for the size of the clusters or the probability of a response in the relevant variables. These combinations of conditions are explored in the second research question below, so the results of the first research question are intended to be exploratory, identifying the trends suggested by aggregated marginal effects before disaggregating and diving into the individual cell results in the second research question, which will illustrate the variability of the results.

As an additional note, recall that the ROCK algorithm could not converge with many datasets where $k = 35$, $N_k = 50$, and probability of a response on the relevant variables was 0.3. In the cases where it did converge, Kappa values were near zero. The lack of the ROCK results in these study cells means that the average ROCK Kappa values are likely inflated in some of the following figures. Had the ROCK algorithm converged in more datasets in these cells, one likely would have observed more low Kappa values, which would lower the mean Kappa scores. Where this has occurred, the plot point is marked with an asterisk (*).

To explore the marginal effects, one can first examine the suggested impact on mean Kappa values when the number of clusters changes, aggregated over the other conditions in the study. Figure 8 shows the mean Kappa values for the algorithms' true cluster recovery over the varying number of clusters in the study (4, 10, 20, and 35 clusters). Figure 8 suggests that the *k*-means and SOM algorithms can be expected to perform worse at recovering the true clusters when the number of clusters increases, with SOM performance declining faster than the *k*-means algorithm. Conversely, the average Kappa value for the ROCK algorithm's recovery of true clusters suggests that ROCK's true cluster recovery improves when the number of clusters increases, although recall that the ROCK algorithm often could not converge when there were 35 clusters with response probabilities of 0.3. When considering number of clusters alone, Figure 8 suggests that *k*-means may be the best algorithm (of the three compared here) for analyzing GBA data. When the game is low autonomy and there are minimal possible gameplay strategies (i.e., fewer clusters), both *k*-means and SOM are expected to perform well. When working with higher autonomy GBAs where many types of work processes may be observed, ROCK and *k*-means may be preferable.

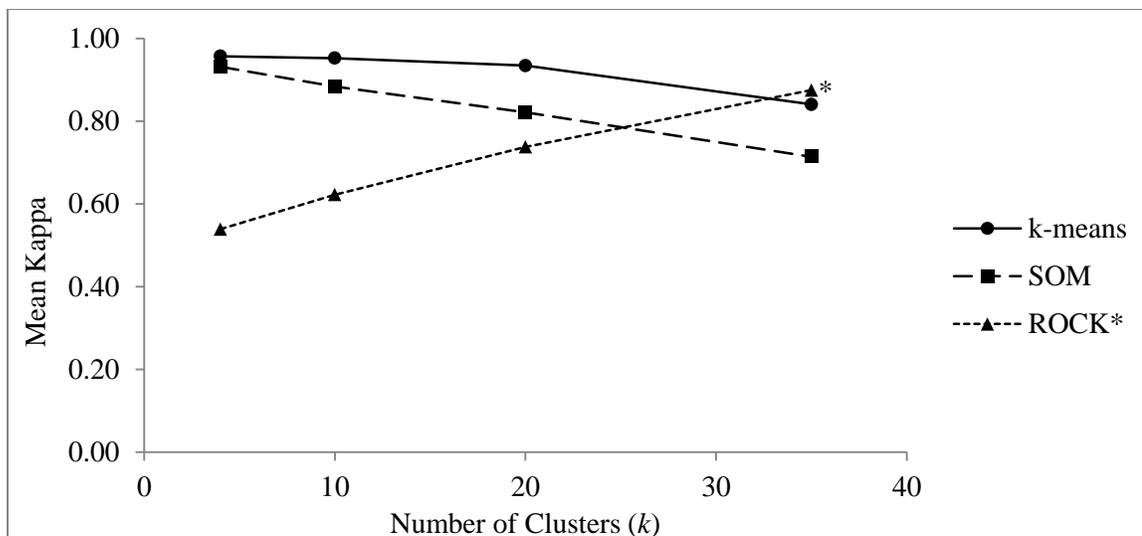


Figure 8. Mean Kappa rates by algorithm for different numbers of clusters k . The ROCK plot marked with an asterisk (*) may be inflated due to the lack of convergence at 35 clusters ($\bar{N}_k = 50$ cases) and probability of response = 0.3. Standard error ranges: k -means – 0.001-0.007, SOM – 0.002-0.009, ROCK – 0.006-0.009.

To further explore the marginal effects, one can next examine the suggested impact on mean Kappa values when the average size of the clusters changes, aggregated over the other conditions in the study. Recall that the study manipulated the average size of the clusters, specifying two levels: small cluster sizes (mean $N_k = 50$) and large cluster sizes (mean $N_k = 100$). Figure 9 suggests that both k -means and SOM are expected to perform well under both scenarios (discounting other data conditions), but that the mean Kappa values for recovering the true clusters often increased with larger sample sizes. The ROCK algorithm did not appear to perform as well as the other two algorithms, and ROCK's mean Kappa value when $N_k = 50$ is likely inflated due to the lack of convergence in the datasets where $k = 35$ and probability of a response was 0.3. Regardless, k -means generally performed the best on average with both cluster size options, when discounting other conditions of the data. When considering the sheer

volume of GBA work process data alone, Figure 9 suggests that k -means and SOM may be preferable algorithms for working with large datasets in some scenarios.

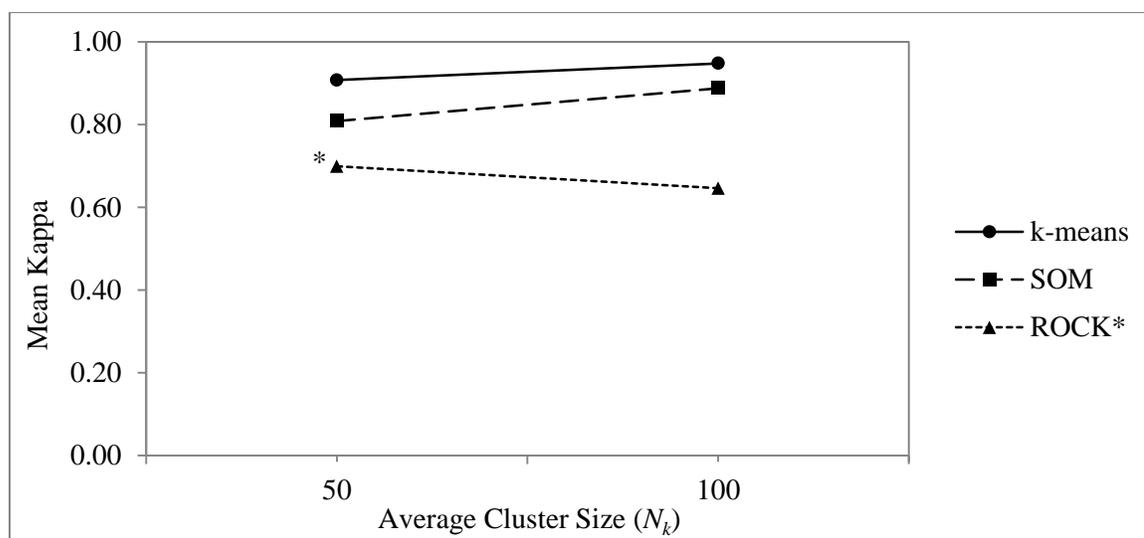


Figure 9. Mean Kappa rates by algorithm for two different average cluster sizes N_k . The ROCK plot marked with an asterisk (*) may be inflated due to the lack of convergence at 35 clusters ($\bar{N}_k = 50$ cases) and probability of response = 0.3. Standard error ranges: k -means – 0.002-0.003, SOM – 0.002-0.003, ROCK – 0.004 for both.

In addition to the general trends associated with the size of clusters, one can also look at marginal effects associated with the variance of the cluster sizes; i.e., if the average cluster had 50 cases in it, clusters could either be similar in size or vary in size. This condition was manipulated by letting the variance of the cluster sizes N_k be either 9 or 100 (e.g., if the mean cluster size was 50 cases, individual cluster sizes would be expected to range from 41-59 cases or from 20-80 cases). This was an important aspect of the simulation because equally sized clusters in GBA work process data are unlikely in practice and because Robinson (2017) claimed that the k -means algorithm will perform poorly when clusters vary in size, with k -means splitting up larger clusters and ignoring smaller clusters in an effort to minimize the sum-of-squared error in its solution. Figure 10, however, suggests that varying cluster sizes did not meaningfully impact the mean

Kappa values in any of three algorithms' recovery of the true clusters (ignoring, for the moment, other aspects of the data). In fact, k -means performed better than SOM and ROCK again. This was unexpected, given Robinson's (2017) claims that k -means requires similar cluster sizes and the fact that k -means is not intended to be used with binary data. Robinson's (2017) claim may hold true when one cluster is far smaller than another, as he demonstrated with an example, but this may cross into the realm of k -means simply doing a poor job with outliers (Berkhin, 2006). Under the aggregated conditions of this simulation study, k -means (as well as SOM and ROCK) were not notably impacted by increasing the cluster sample size variance from 9 to 100.

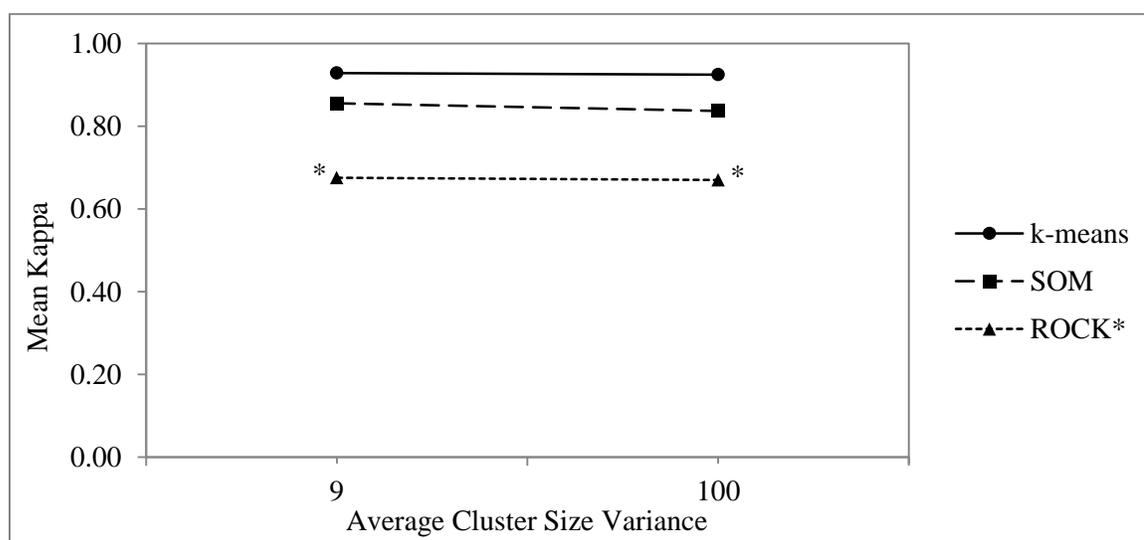


Figure 10. Mean Kappa rates by algorithm for two different variances in the data's cluster sizes N_k . The ROCK plots marked with an asterisk (*) may be inflated due to the lack of convergence at 35 clusters ($\bar{N}_k = 50$ cases) and probability of response = 0.3. Standard errors are: k -means – 0.00 for both, SOM – 0.003 for both, ROCK – 0.004 for both.

One can also explore the marginal effects on true cluster recovery over the different probabilities of response in the 20 relevant variables that defined each cluster, aggregating again across the other conditions in the study. Recall that all variables in each dataset had a default observation rate of 0.05, but then 20 variables were assigned to

each cluster that had probabilities of 0.3, 0.6, or 0.9 (selected randomly with replacement between clusters), depending on the specification of the simulation cell. Higher probabilities indicate that the cases in the cluster were all highly similar, as might occur in a very structured, low-autonomy GBA. Low probabilities represent clusters with more diversity in observed work processes, but where students in that cluster are all drawing from a common set of variables. When probabilities of observed responses are higher for clusters' relevant variables, the response patterns within a cluster correlate well with each other, generally making it easier to identify a cluster with an axis parallel soft-projected clustering algorithm. When the probability of a response on a cluster's relevant variables is lower, and the probability of a response is also very low on the remaining irrelevant variables in the dataset, then there is more similarity in all response patterns across clusters, and correlations with cases between clusters will be similar to cases within a cluster, thus making it more difficult to correctly identify clusters.

Following these expected trends, Figure 11 suggests that, when aggregated across conditions, mean Kappa values of true cluster recovery may be expected to increase as the probability of observing a response increases, with all three algorithms generally recovering the true clusters nearly flawlessly when the probability is 0.9. The *k*-means algorithm performs nearly perfectly under many scenarios when probability is 0.6, and SOM performance is also often good under this condition. All three algorithms struggle on average when probability of response is 0.3, although the mean Kappa values for *k*-means and SOM may be good enough to justify the use of these algorithms under some conditions when evaluating a low stakes GBA.

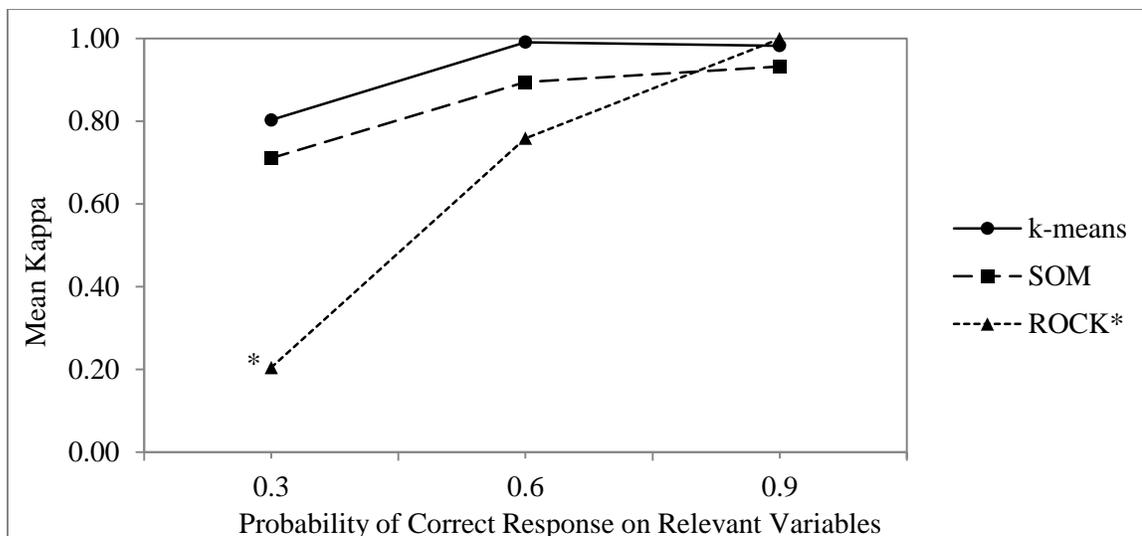


Figure 11. Mean Kappa rates by algorithm for three different probabilities of response in the relevant variables defining clusters. The ROCK plot marked with an asterisk (*) may be inflated due to the lack of convergence at 35 clusters ($\bar{N}_k = 50$ cases) and probability of response = 0.3. Standard error ranges: *k*-means – 0.001-0.004, SOM – 0.002-0.004, ROCK – 0.002-0.004.

The final condition of interest for the first research question was whether the amount of overlap between relative variables influenced the algorithms' abilities to recover the true clusters; i.e., would the algorithms perform better if the clusters were well-separated? The degree of overlap (or similarity) was measured with the RMSD between clusters, and it was not directly manipulated in the simulation code. Instead, the 20 relevant variables defining each cluster were randomly selected (with replacement), meaning that in some datasets, clusters might share multiple defining, relevant variables, whereas in other datasets, that overlap might be smaller.

The RMSD values were contingent on the number of total variables and probabilities of observed responses in clusters' relevant variables in each cell, so it is most illuminating to consider each cell individually. Since RMSD is a continuous measure (rather than the discrete levels set in the previous conditions), one can look at the Pearson correlations between RMSD and the algorithms' Kappa values for recovering the

true clusters in each cell. Table 19 shows these correlations. Overall, Table 19 shows that increasing the between-cluster variance had little effect on improving algorithm performance, with most cells showing only low positive or negligible correlations between RMSD and algorithms' Kappa values. The notable exceptions occurred for the k -means and SOM algorithms when there were only four clusters and the probability of a correct response was 0.3. In these cells, there were low to moderate positive correlations between RMSD and the k -means and SOM Kappa values, with higher correlations occurring when the clusters were larger. Note that lower correlations are associated with a lack of variation in Kappa values in the solutions, and correlations are omitted in cells where the correlation was not significant or the algorithm consistently returned Kappa values of 1.00 for all datasets.

Table 19

Correlations between RMSD and Kappa by Study Cell and Clustering Algorithm

k	Size	k -means			SOM			ROCK		
		Pr(1)			Pr(1)			Pr(1)		
		0.3	0.6	0.9	0.3	0.6	0.9	0.3	0.6	0.9
4	$\sim N(50,9)$	0.39	0.04	—	0.33	-0.05	-0.04	0.17	0.26	—
	$\sim N(50,100)$	0.38	0.08	—	0.29	0.00	0.00	0.19	0.18	0.06
	$\sim N(100,9)$	0.51	0.16	—	0.49	0.02	0.07	0.17	0.17	—
	$\sim N(100,100)$	0.48	0.18	—	0.48	-0.03	-0.04	0.18	0.16	0.06
10	$\sim N(50,9)$	0.17	0.04	—	0.09	0.01	0.07	0.13	0.13	—
	$\sim N(50,100)$	0.12	0.07	—	0.04	-0.01	-0.05	0.12	0.20	—
	$\sim N(100,9)$	0.28	0.04	—	0.03	0.09	-0.01	0.18	0.17	—
	$\sim N(100,100)$	0.28	0.01	—	0.07	0.12	-0.04	0.22	0.09	—
20	$\sim N(50,9)$	0.04	0.05	—	0.09	0.09	0.09	0.11	-0.01	0.03
	$\sim N(50,100)$	0.13	0.00	—	0.07	-0.08	0.08	0.11	0.03	—
	$\sim N(100,9)$	0.18	0.03	—	0.03	-0.04	0.01	0.18	0.00	—
	$\sim N(100,100)$	0.12	0.04	—	0.11	-0.01	-0.05	0.16	0.06	0.05
35	$\sim N(50,9)$	—	0.00	0.04	—	0.07	0.05	—	0.06	0.05
	$\sim N(50,100)$	—	0.01	—	—	-0.04	—	—	0.11	—
	$\sim N(100,9)$	—	-0.06	—	—	-0.08	—	—	0.08	—
	$\sim N(100,100)$	—	0.04	—	—	-0.12	—	—	0.08	—

Research Question 2: Conditions Under Which Kappa is Expected to Exceed 0.80

The second research question investigated the combinations of the study conditions that are expected to yield true cluster recovery Kappa values of at least 0.80. Recall that Landis and Koch (1977) suggested that Kappa values of 0.80 or higher represent near perfect agreement, or in this case, near perfect recovery of the true clusters. The 0.80 cutoff is admittedly arbitrary, but it serves as a useful reference point for comparing the performance of the algorithms across the different cells. With this in mind, a high level interpretation can be made by simply looking at the percentage of solutions in each cell that have Kappa values of 0.80 or higher. Table 20 shows the percentage of each algorithm's solutions that exceeded Landis and Koch's (1977) 0.80 Kappa threshold for near perfect true cluster recovery.

Table 20 shows that the *k*-means and SOM algorithms recover the true clusters reasonably well in most cases where the probability of a response in the relevant variables is 0.6 or 0.9, although the *k*-means algorithm performs better on average. When the probability of a response is 0.3, the *k*-means and SOM algorithms perform worse when there are a large number of small clusters. The SOM algorithm begins to perform poorly when the response level is 0.3 and there are 20 clusters, even when those clusters are large. Conversely, the ROCK algorithm has a near perfect recovery rate when the probability of response is 0.9 and a total failure rate when response probability is 0.3. When the response probability is 0.6, the ROCK algorithm recovers the true clusters much better when there are 35 clusters.

Table 20
Percentage of Solutions with Kappa ≥ 0.80 by Study Cell and Clustering Algorithm

<i>k</i>	Size	<i>k</i> -means			SOM			ROCK		
		Pr(1)			Pr(1)			Pr(1)		
		0.3	0.6	0.9	0.3	0.6	0.9	0.3	0.6	0.9
4	$\sim N(50,9)$	93%	100%	100%	90%	93%	96%	0%	35%	100%
	$\sim N(50,100)$	81%	100%	100%	79%	91%	88%	0%	37%	100%
	$\sim N(100,9)$	100%	100%	100%	100%	96%	97%	0%	23%	100%
	$\sim N(100,100)$	100%	100%	100%	100%	92%	94%	0%	24%	100%
10	$\sim N(50,9)$	95%	100%	100%	61%	85%	94%	0%	69%	100%
	$\sim N(50,100)$	82%	100%	100%	41%	77%	88%	0%	72%	100%
	$\sim N(100,9)$	100%	100%	100%	91%	89%	96%	0%	16%	100%
	$\sim N(100,100)$	100%	100%	100%	90%	88%	95%	0%	17%	100%
20	$\sim N(50,9)$	27%	100%	100%	0%	86%	89%	0%	63%	100%
	$\sim N(50,100)$	12%	100%	100%	0%	72%	85%	0%	64%	100%
	$\sim N(100,9)$	100%	100%	100%	47%	90%	88%	0%	82%	100%
	$\sim N(100,100)$	99%	100%	100%	36%	86%	88%	0%	82%	100%
35	$\sim N(50,9)$	0%	98%	81%	0%	89%	75%	0%	99%	100%
	$\sim N(50,100)$	0%	98%	90%	0%	78%	75%	0%	99%	100%
	$\sim N(100,9)$	—	100%	—	—	100%	—	—	100%	—
	$\sim N(100,100)$	—	90%	—	—	80%	—	—	100%	—

Visually, one can see these results by looking at boxplots of the Kappa values for each cell of the study. Figure 12 shows these distributions by cell, plotting boxplots with the same number of clusters and response probability next to each other, ordered by cluster size and cluster size variance. Figure 12 helps illustrate some of the effects for the interactions of the simulated data. For example, in the *k*-means and SOM algorithms, higher variability in cluster sizes does appear to lower Kappa values on true cluster

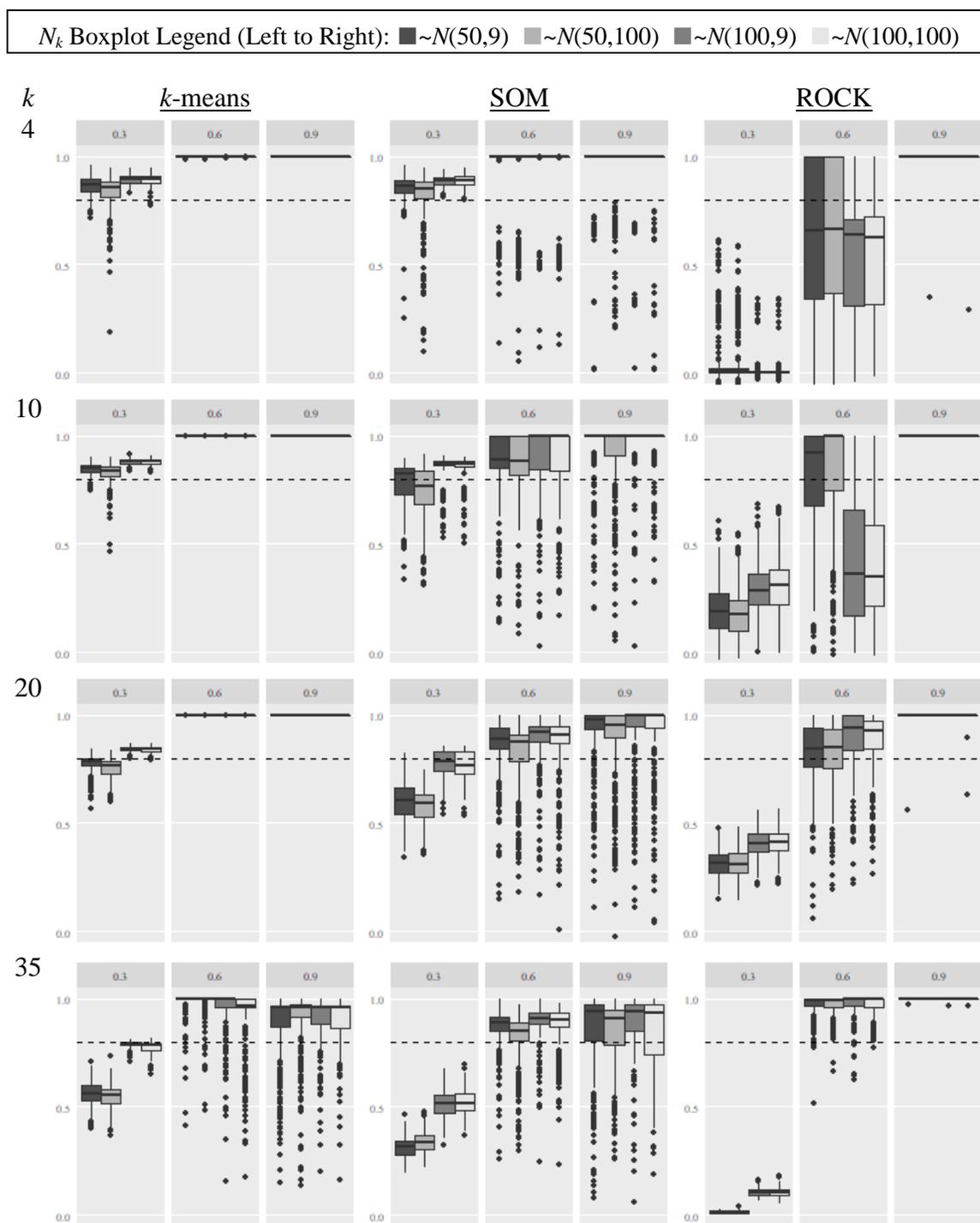


Figure 12. Kappa distributions by cluster size, for each combination of response probability and k clusters.

recovery when the cluster sizes are small (compare the two leftmost k -means and SOM boxplots when the cluster sizes are 50, comparing cluster size variance at 9 versus 100). One also observes a scattering of low Kappa outliers in the SOM solution, which, as discussed at the beginning of this chapter, appear to be a result of the random starting centers selected for these solutions, and not a direct effect from the cell parameters themselves.

Figure 12 also makes it easy to see which algorithms recover true clusters nearly perfectly. For example, notice how the k -means Kappa values are consistently close to 1.00 when response probabilities are 0.6 and 0.9, but that this performance begins to deteriorate when the number of clusters is increased to $k = 35$. Notice too how well the ROCK algorithm performs when the response probability is 0.9 except for a few outliers.

The ROCK boxplots when response probability is 0.6 in Figure 12 are also of interest, because one sees that the ROCK algorithm performs *worse* at recovering large clusters when $k = 4$ and $k = 10$. Based on these Kappa values, one might almost consider using the ROCK algorithm when response probability was near 0.6 and there were 10 clusters anticipated, but not if a large number of students were going to play the GBA. However, the ROCK algorithm begins to perform very well when response probability is 0.6 and $k = 20$ or $k = 35$, suggesting ROCK might be a great algorithm to use when there are high autonomy games and lots of observable strategies, with moderate or high consistency in the work process response patterns within clusters.

Research Question 3: Conditions In Which Algorithms Can Validate Each Other

By inspecting the boxplots in Figure 12, one can see that all three algorithms have high Kappa values when the probability of response in the relevant variables is 0.9. This

appears to be true regardless of the number of clusters, the size of the clusters, or the variation in cluster size in this study. When the response probability is 0.9, there is very little within-cluster variance, and all response patterns in a cluster are highly similar on a defining set of variables. This does not necessarily describe a low autonomy GBA, considering that one could have 35 clusters with response probabilities of 0.9, meaning that students are engaging in 35 different work processes; however, it does suggest that a GBA would be highly structured to minimize variation in work process patterns. These probabilities would not occur in a high autonomy game that was mixed with lots of chaos elements or relatedness elements to give the game a highly realistic, open-world feel.

Similarly, all three algorithms recover true clusters reasonably well (in terms of observed Kappa values) when the response probability is 0.6 and there are 20 clusters. In this scenario, all three algorithms again could be used to validate each other. This is encouraging news from a GBA standpoint, as these parameters suggest that a moderate amount of autonomy can be designed into the GBA and still support the use of these clustering algorithms. A GBA with 20 clusters and a 0.6 probability of a response on each cluster's relevant variables is one in which there is diversity in work processes (20 clusters) and some degree of diversity within those work processes—a response probability of 0.6 increases within-cluster variance compared to a response probability of 0.9.

Only the *k*-means and SOM algorithms have acceptable expected Kappa values when the response probability is 0.3, and even then, that only occurs when there is a small number of clusters: $k = 10$ or 4 clusters. The *k*-means algorithm produces reasonable Kappa values when $k = 20$, but the SOM algorithm does not, so they could not

validate each other. When there are only a small number of clusters ($k = 4$ or $k = 10$), the SOM and k -means algorithms both perform well at each of the three probabilities of observed response. This suggests that these two algorithms may be used to validate each other under these data conditions, and that these two algorithms may be preferable to the ROCK algorithm when analyzing binary GBA data in lower autonomy games.

Perhaps the most important observation overall is that there are very few conditions where Kappa values greater than 0.80 occurred in all datasets. The SOM algorithm tended to have outlier solutions with low Kappa values in almost every cell. As noted previously, these are likely caused by poor starting centers, and these failures could be obviated by running multiple SOM analyses with different starting centers, as the k -means algorithm does. In terms of informing practice, these results could be seen as evidence that researchers should analyze their GBA data with *more* than two clustering algorithms, since the algorithms have a susceptibility to converge on a poor solution in most of the simulation cells. Using three algorithms when analyzing GBA work process data gives the researcher a better chance of seeing at least two algorithms create similar solutions for validating the use of the cluster analysis, as is shown in the tutorial in the next chapter.

Research Question 4: Finding the Number of Clusters with DBI

The fourth research question was investigated by simulating datasets in each cell as before, but then asking the algorithm to try different values of k for each dataset. Because the data were simulated, the true number of clusters was known, so the algorithms were instructed to run through a range of k values from three below the true k to three above (two below when the true number of clusters was $k = 4$, since the

algorithms cannot create a single cluster solution where $k = 1$). This means that each algorithm created seven cluster solutions for each simulated dataset, each with a different number of clusters. The DBI values were calculated for each of these solutions to see which solution had the best fit (lowest DBI). This allowed for an investigation into whether the algorithms' best-fitting solution tended to overestimate, underestimate, or match exactly the true number of clusters.

Of course, over 400 datasets in each cell, some solutions might overestimate the number of clusters and some solutions might underestimate the number of clusters. It is helpful to consider whether an algorithm tends to overestimate or underestimate the number of clusters, using DBI as the decision criterion. Figure 13 represents these proclivities visually. The grey segments of the chart represent the percentage of times the lowest DBI corresponded to the correct number of clusters. The black and white segments represent the percentage of datasets where the lowest DBI value over- or underrepresented the true number of clusters, respectively.

Figure 13 shows that the algorithms tend to find better fitting solutions when they are instructed to pick more than the true number of clusters (with a few exceptions in the SOM algorithm where $k = 4$ and probability of an observed response is 0.3). This can be seen from the number of charts where the black segment (DBI overestimating the number of clusters) is the largest segment. Figure 13 shows two algorithm-level trends: first, that the ROCK algorithm's best-fitting solution never matches the true number of clusters, and second, that the k -means algorithm is the most likely to identify the correct number of clusters in each cell of the study when using DBI as the selection criterion (although it still tends to overestimate the number of clusters).

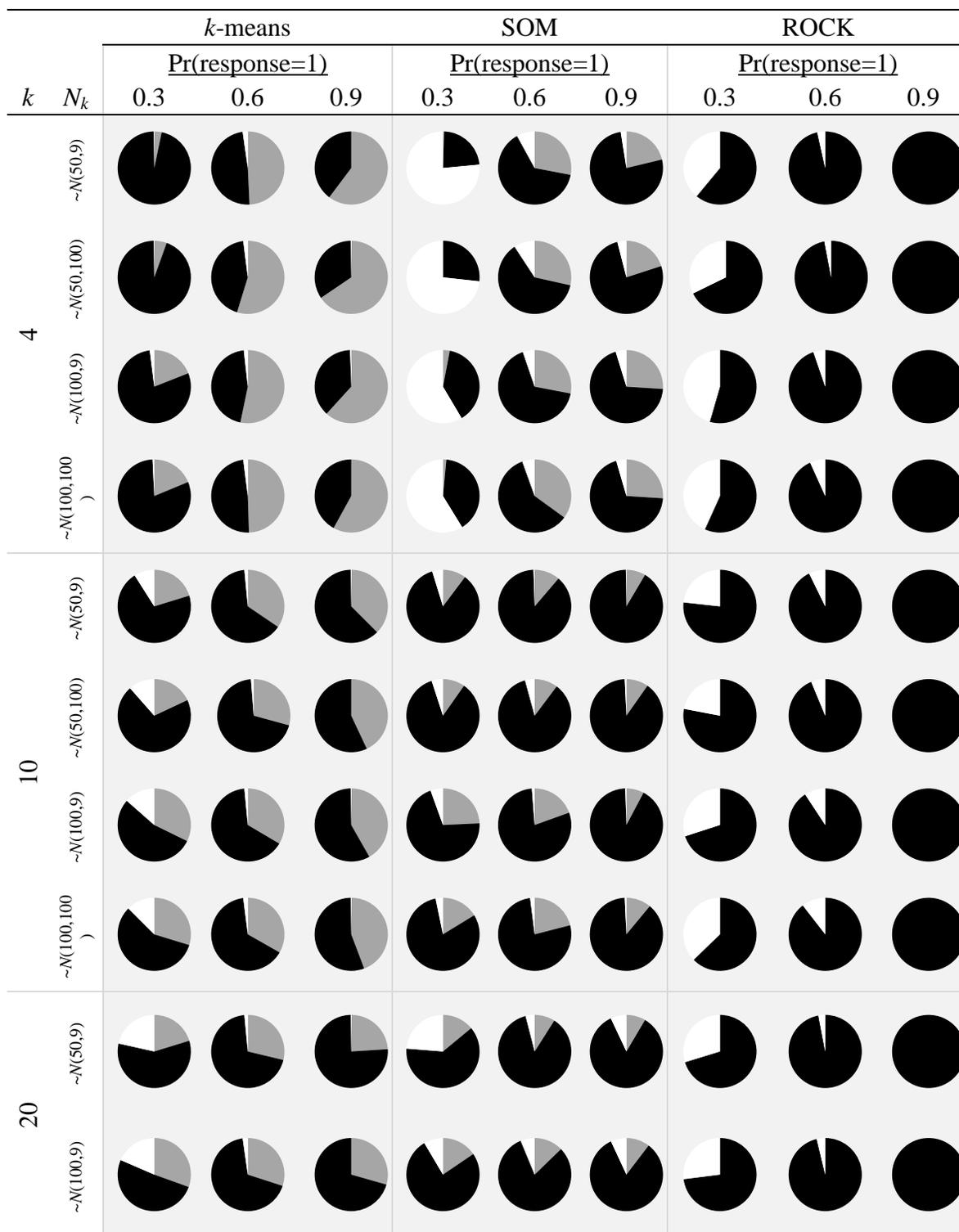


Figure 13. Pie charts depicting the percentage of datasets was correctly identified (■ grey segments), overestimated (■ black segments), and underestimated (□ white segments), as determined by the clustering solution with the lowest DBI. Due to the computational load of these comparisons, results are provided for the first 30 cells of the study.

The tendency to overestimate the number of clusters may be attributable to the use of DBI as the decision criterion, and not a feature of the algorithms themselves. Recall that DBI is lowest when there is higher between-cluster variance and lower within-cluster variance, but this may be achieved by creating more small clusters, tailor-fitted to a few similar cases. Recall in the feasibility study in Chapter 3 that the ROCK algorithm produced superior DBI values compared to the SOM and k -means algorithms by lumping most cases into a single large cluster and creating a few small clusters for the remaining outlying cases. The low within-cluster variance in these smaller clusters in those ROCK solutions helped yield a much lower DBI, even though the solution itself would not be a very useful partition of the cases.

Baker et al. (2017b) observed that adding more clusters will often lead to better coverage of the dataspace and lower within-cluster variance, and they recommend using a method like AIC or BIC to penalize models with more clusters. Recall in the discussion from Chapter 3 that Pelleg and Moore (2000) developed an algorithm for using these information criterion measures to select the number of clusters for k -means, and Wang et al. (2005) created a similar method for SOM; however, Ishioka (2005) showed that Pelleg and Moore's (2000) algorithm still could overestimate the true number of clusters. Recall too that Dimitriadou et al. (2002) claimed that DBI was the best measure for identifying the number of clusters in binary data, such as those created in these simulated GBA datasets, but their study showed very similar results, with DBI correctly identifying the number of clusters in only 35% of their binary datasets (analyzed using k -means and a hard competitive learning clustering algorithm). Dimitriadou et al. also observed a tendency to overestimate the true number of clusters when using DBI, and it is worth

noting that they did not evaluate the performance of AIC or BIC for selecting the number of clusters.

Figure 14 illustrates the challenges of using DBI for deciding the number of clusters. Figure 14 shows DBI values for the k -means, SOM, and ROCK algorithms across varying numbers of clusters (k) for a dataset that was simulated with 20 clusters, average cluster sizes of 50 cases (with cluster size variance set to 9), and a 0.6 probability of an observed response in the clusters' relevant variables. The DBI plots do not monotonically decrease like factor analysis scree plots, and there are clearly instances of local minima. Davies and Bouldin (1979) recommended using the global minimum DBI as the indicator of the best-fitting solution, but the plots in Figure 14 show that for each algorithm, many DBI plots are similar to the lowest value, and there is no DBI distribution that can be used to determine if these are statistically different. What Figure 14 does show is that for this simulated dataset, the use of the lowest DBI value for determining the number of clusters led to an overestimation of the number of clusters in all three algorithms.

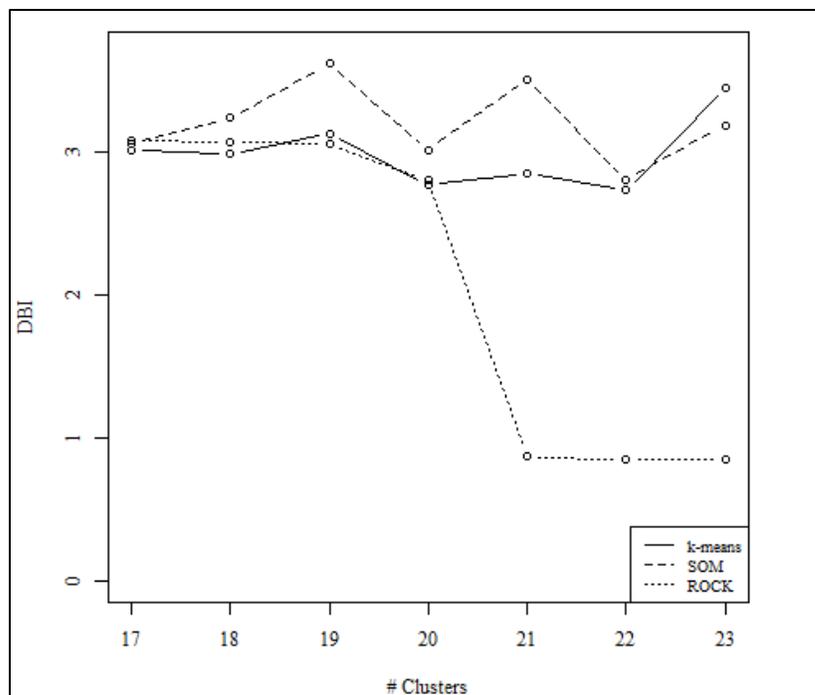


Figure 14. Plot of DBI values for three clustering algorithms when varying the number of requested clusters from $k = 17$ -23 clusters. The dataset used for this plot was simulated to have 20 clusters. The DBI plots have local minima, and the global minimum in this dataset overestimates the number of true clusters that were used to generate the data.

Simulation Study Results Summary

This study reviewed the performance of the k -means, SOM, and ROCK algorithms under a variety of data scenarios representing common aspects of GBA work processes, including the presence of construct-irrelevant variables, differences in the probabilities of responses, overlapping processes between clusters, varying cluster sizes, and differences in autonomy (i.e., different numbers of clusters). Under the simulated conditions, the k -means algorithm tended to be the safest bet, outperforming the SOM and ROCK algorithms' recovery of the true clusters in many cells. The SOM algorithm tended to perform nearly as well as the k -means algorithm, but it frequently had outlying simulated datasets where it converged on a solution that did not recover the true clusters well. This seems to be primarily attributable to the poor selection of starting centers, and

allowing the SOM algorithm to run through many sets of starting centers and then to choose its best-fitting solution (as the k -means algorithm does) would likely obviate these anomalies. The ROCK algorithm performed poorly unless the response probabilities were set to 0.9 for the relevant variables in each cluster, suggesting that the ROCK algorithm is best suited for data in which clusters are clearly distinguished from each other and where each clusters cases are almost guaranteed to have observations in a shared set of relevant variables. The ROCK algorithm's recovery of true clusters generally improved as the number of clusters increased, although it never performed well (or in some cases, converged at all) when the response probability was 0.3.

What this shows is that all three of these algorithms may yield misleading results under certain conditions, thus underscoring the need for GBA researchers to justify their selection of clustering algorithms and cross-validate across multiple algorithms. One cannot assume that one preferred clustering algorithm can work as a measurement model in all GBA designs or work process datasets. The practical application of the use of multiple algorithms is demonstrated in the following chapter (Chapter 6), and the implications for researchers are explored further in the Discussion chapter (Chapter 7).

Chapter 6: Tutorial Example Study

This tutorial is intended to provide examples and guidance on the steps and decisions one should consider when using clustering algorithms to evaluate students in a GBA. The steps here are not necessarily exhaustive, nor can one expect them to generalize to every possible game design. Moreover, the concepts discussed here are not original. These steps are, however, an amalgamation of best-practice recommendations or observed applications pooled from multiple sources, and by combining them here, this tutorial provides value by exemplifying these ideas in one place and explicitly stating how they relate to GBA research using clustering algorithms. The underlying purpose of these steps and decisions is for the researcher to be able to document and defend assessment inferences made on the basis of a clustering algorithm, essentially building supporting evidence and obviating rebuttals for a validity argument about the interpretation of students' cluster assignments. With this goal in mind, the tutorial will walk through the following eight steps:

1. Coding and selecting variables from the GBA log file.
2. Selecting clustering algorithms.
3. Creating a holdout sample for cross-validation.
4. Running the algorithms.
5. Validating across algorithms to pick the best clustering solution.
6. Validating against the holdout sample and an external criterion.
7. Interpreting, labeling, or grouping clusters for reporting.
8. Investigating changes in students' clusters over time or the course of the game.

Beanstalk Game

Beanstalk is a mathematics game designed to teach fulcrum principles, specifically the sum of cross products rule that determines whether a scale with a fulcrum (e.g., a see-saw) will balance, given the weights on the scale and their distance to the fulcrum. Carnegie Mellon (2013) explained that the game is intended to teach principles of balance, scientific inquiry, and social-emotional skills, which they aligned to K-3 objectives in the National Research Council's Framework for K-12 Science Education and Pennsylvania's Academic Standards. Although the game is aligned to K-3 objectives, the designers reported that the game is intended for children between ages 5 to 11 years old. The designers also noted that the levels in the game are intended to increase in difficulty as the mental models become more complex. The designers explained that there are four progressively complex mental models in the game, and that these are based on the research of Siegler (1976) and Siegler and Chen (2002). Carnegie Mellon (2013) reports that these mental models are:

- Paying attention to weight, but not the distances on the scale.
- Considering distance when weight is equal on both sides of the scale.
- Considering both weight and distance while being provided with cues for congruity.
- Using the sum of cross products rule.

The game storyline centers around a monster who lives on another planet and drops his teddy bear while he sleeps. The teddy bear lands in the room of the avatar controlled by the student, and a magic beanstalk begins to grow. Teetering atop the beanstalk is a platform, which the student must balance to make the beanstalk grow

further. The student is aided by a talking chicken and a talking crow that provide hints, interspersed with cheerful banter. The scale is tipped to one side with the weight of a collection of ladybugs. The student must grow flowers or use eggs from the talking chicken to balance the scale and make the beanstalk grow. The goal is to grow the beanstalk up into space so that the teddy bear can be returned to the extraterrestrial monster. The monster explains the solution to the scale if the student uses up the hints. The student's actions in the game primarily consist of picking where to place flowers or eggs and clicking on the birds to get a hint. The student can also undo a choice to rearrange the positions of flowers and eggs on the scale. Figure 15 shows a screenshot from the game (Carnegie Mellon, 2013).

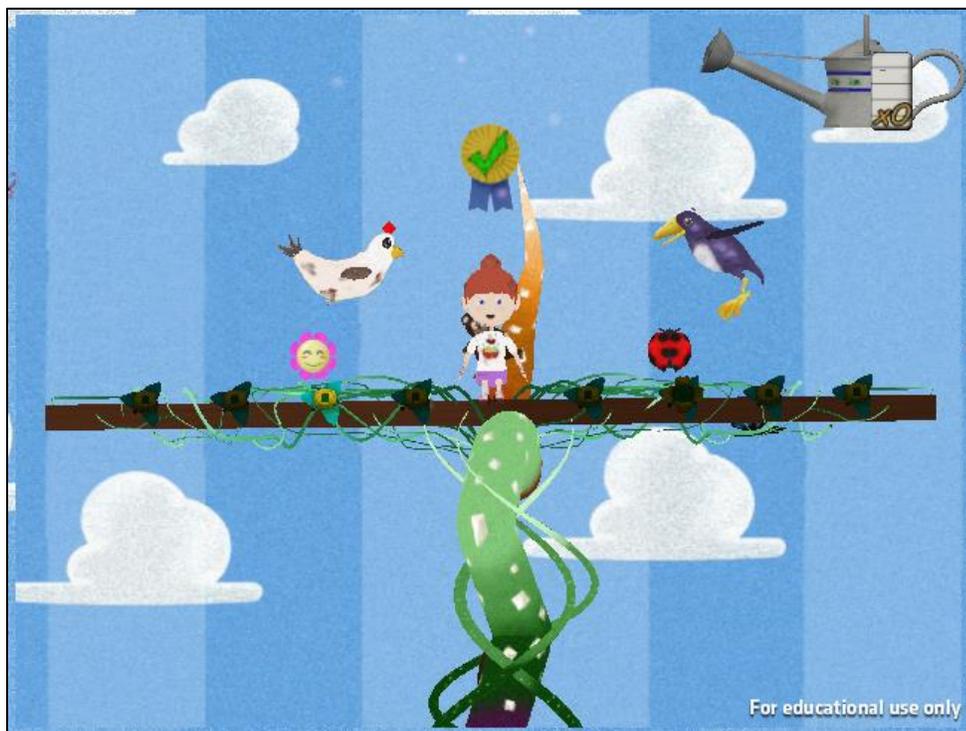


Figure 15. Screenshot from Beanstalk. The student must grow flowers on the left side of this scale to balance the weight of the ladybugs on the right side.

Beanstalk employs many of the game design elements defined by Mislevy et al. (2014). Rules, connections, game state, and mechanics all define what the student can do and what the outcomes will be. The game has low autonomy—there is relatively little variability in possible student actions, and there are no chaos elements—but that may be appropriate for the target age range. Competence elements, however, are featured prominently. The bulk of the game design is around creating a hierarchy of challenges and teaching the student to achieve those goals through experimentation and observation. All of this is saturated with relatedness elements (arguably the distinguishing factor between games and other instrument) such as music, background animations, a fantastical plot, avatars, quirky characters, and impossible scenarios (Carnegie Mellon, 2013).

Although Beanstalk is clearly a game, it is not an assessment, so in this sense, it is not strictly a GBA. The game tracks students anonymously, and there is no reported output or evaluation of the student at the end of the game. This, however, is the hypothetical scenario that is being explored by this tutorial: were one to begin using Beanstalk as an assessment, how could one create a measurement model with a clustering algorithm to evaluate students?

Beanstalk Student Data

The log file data in the 'Beanstalk_2013_05' dataset was accessed via DataShop (Koedinger et al., 2010). This dataset consists of 42,802 recorded student actions collected from 177 students in 371 gameplay sessions at two Pennsylvania elementary charter schools on May 8-9 and May 29-30, 2013. Beanstalk does not collect any identifying information, and anonymous IDs are used to represent students in the dataset (Carnegie Mellon, 2013). As such, there is no information about the ages or

demographics of the students in this dataset. The dataset is a log file format, with each row representing a recorded student action, tracking key data like the anonymous student ID, timestamps, attempt numbers, locations of items placed on the scale, the types of hints provided, and outcomes. Figure 16 shows a screenshot of excerpted columns from the 'Beanstalk_2013_05' dataset.

non Student Id	Session Id	Time	Duration (sec)	Tutor Res	Problem Name	Problem View	Problem Star	Attempt #	Is Last Att	Outcome	Selection	Action	Input	Feedback Text
tu_0022a3dfc22	fb7bcfb6-	5/8/13 16:13	8	RESULT	Tier 1, order 1		1	5/8/13 16:13	1	1	CORRECT	Plank -2	Add_Flower	1 [HelpVoice: Narrative]
tu_0022a3dfc22	fb7bcfb6-	5/8/13 16:14	5	RESULT	Tier 1, order 2		1	5/8/13 16:13	1	0	INCORRECT	Plank 1	Add_Flower	1 [HelpVoice: Fail1]
tu_0022a3dfc22	fb7bcfb6-	5/8/13 16:14	20	RESULT	Tier 1, order 2		2	5/8/13 16:14	1	1	CORRECT	Plank 2	Add_Flower	1 [HelpButtonForNoSEL: T
tu_0022a3dfc22	fb7bcfb6-	5/8/13 16:14	3	RESULT	Tier 1, order 3		1	5/8/13 16:14	1	1	CORRECT	Plank 4	Add_Flower	1 [HelpVoice: Narrative]
tu_0022a3dfc22	fb7bcfb6-	5/8/13 16:15	5		Tier 1, order 4		1	5/8/13 16:15				Plank -4	Add_Flower	1
tu_0022a3dfc22	fb7bcfb6-	5/8/13 16:15	4	RESULT	Tier 1, order 4		1	5/8/13 16:15	1	1	CORRECT	Plank -4	Add_Flower	2 [HelpVoice: Narrative]
tu_0022a3dfc22	fb7bcfb6-	5/8/13 16:15	3		Tier 1, order 5		1	5/8/13 16:15				Plank 1	Add_Flower	1
tu_0022a3dfc22	fb7bcfb6-	5/8/13 16:15	3	RESULT	Tier 1, order 5		1	5/8/13 16:15	1	1	CORRECT	Plank 2	Add_Flower	1 [HelpVoice: Narrative]
tu_0022a3dfc22	fb7bcfb6-	5/8/13 16:15	6	RESULT	Tier 1, order 6		1	5/8/13 16:15	1	1	CORRECT	Plank -3	Add_Flower	1 [HelpVoice: Narrative]
tu_0022a3dfc22	fb7bcfb6-	5/8/13 16:16	6		Tier 1, order 7		1	5/8/13 16:16				Plank 4	Add_Flower	1
tu_0022a3dfc22	fb7bcfb6-	5/8/13 16:16	4	RESULT	Tier 1, order 7		1	5/8/13 16:16	1	1	CORRECT	Plank 2	Add_Flower	1 [HelpVoice: Narrative]
tu_0022a3dfc22	fb7bcfb6-	5/8/13 16:16	4		Tier 1, order 8		1	5/8/13 16:16				Plank 2	Add_Flower	1
tu_0022a3dfc22	fb7bcfb6-	5/8/13 16:16	4	RESULT	Tier 1, order 8		1	5/8/13 16:16	1	1	CORRECT	Plank 2	Add_Flower	2 [HelpVoice: Narrative]
tu_0022a3dfc22	fb7bcfb6-	5/8/13 16:16	5		Tier 1, order 9		1	5/8/13 16:16				Plank 4	Add_Flower	1
tu_0022a3dfc22	fb7bcfb6-	5/8/13 16:16	4		Tier 1, order 9		1	5/8/13 16:16				Plank 1	Add_Flower	1
tu_0022a3dfc22	fb7bcfb6-	5/8/13 16:17	5		Tier 1, order 9		1	5/8/13 16:16				Plank 3	Add_Flower	1
tu_0022a3dfc22	fb7bcfb6-	5/8/13 16:17	3	RESULT	Tier 1, order 9		1	5/8/13 16:16	1	1	CORRECT	Plank 3	Add_Flower	2 [HelpVoice: Narrative]
tu_0022a3dfc22	fb7bcfb6-	5/8/13 16:19	5	RESULT	Tier 2, order 1		1	5/8/13 16:19	1	1	CORRECT	Plank 2	Add_Flower	1 [HelpVoice: Win1]
tu_0022a3dfc22	fb7bcfb6-	5/8/13 16:19	5	RESULT	Tier 2, order 2		1	5/8/13 16:19	1	1	CORRECT	Plank -3	Add_Flower	1 [HelpVoice: Win1]

Figure 16. Screenshot of Beanstalk log file dataset. The log file rows represent individual actions made by the students, such as asking for a hint and placing an item on the scale.

Step 1: Recode the Data and Select Variables

To create a numerical dataset that can be used by a clustering algorithm from log file data, one must first consider the unit of analysis. For this example, the unit of analysis is a student's attempt at a single level, with each game level representing an unbalanced scale scenario in the Beanstalk game. This means that if a student tried the same scenario twice, there would be two rows in the dataset used for analysis. This allows one to create clusters representing strategies at each level, recognizing that the student may come back and try a new strategy. In short, every attempt at a level gets a row in the dataset. The final recoded dataset for analysis is made up of 21,018 cases representing attempts at game levels in Beanstalk.

Kerr and Chung (2012) and Stevens and Casillas (2006) also used the attempts as the unit of analysis in their GBAs. This has the benefit of showing how individual

students change strategies across multiple attempts at a level, as is discussed in Step 8 below. Recall that one of the reasons to use a game as an assessment is that it should be fun and challenging enough that students will fail certain levels and keep trying them until they succeed, so most GBA research should focus on the level attempts as the unit of analysis.

This is not, however, the only way to structure the dataset. One could set up the dataset so that instead of attempts at each level, the rows represented a student, and the student's repeated attempts at a level would be documented in the columns. One could also look at the student's performance across levels in a single row, examining gameplay processes that span across the designer-defined segments of the game. If data were available, one could also analyze at the level of classrooms or schools, if desired. The data could also be turned around so that each row represented a level in the game, with student gameplay represented in columns. This could be used to examine which levels yield similar gameplay patterns, perhaps as an investigation into the effects of certain design decisions. The unit of analysis represented by rows in the dataset depends on the research question, and since this tutorial is interested in examining a student's work process, each row represents a single response pattern for a given level.

Next, one must decide which variables to include in the columns. These decisions should be guided by the researcher's theory or game design choices that are meant to be used as evidence in the CAF model. For this analysis, the dataset includes variables that depict:

- The game level (i.e., the specific scale scenario being solved).
- Where the student placed flowers or eggs on the scale to balance it.

- How many flowers or eggs the student placed at each location on the scale.
- Whether the student got hints or answers from the talking chicken, crow, or monster.

What is being left out? Student response times, attempt number, whether the item was answered correctly, and other data included in the log file. These data may still need to be retained in the dataset for data management or other reporting, but they were not used in the clustering algorithms. This is not to say that they could not be used as assessment evidence, but for the inference being made in this tutorial, the student's work process is only evaluated on where the student placed items on the scale and whether he or she asked for help from the game characters. Remember that the purpose of using GBAs should be to evaluate process, not outcome, since everyone should be able to eventually win a game, but the evidence of interest is how they do so over repeated gameplay.

Finally, the variables that were selected to be used in the clustering algorithms may have to be recoded. For each response pattern for a given item scenario in the game, binary variables were created that corresponded to each position on the scale, as well as the number of times the student used that position. For example, the variable for placing one flower at the third position to the right of the fulcrum was labeled x3.1. If the student placed a second flower on top of the first, this would be recorded in x3.2. The binary coding of this method allows one to use the ROCK algorithm, which only analyzes binary data; however, one could just as easily create a single variable with an integer representing the number of flowers the student placed.

The use of the hints had to be parsed from the Feedback Text variable shown on the right hand column in the log file data depicted in Figure 16 above. Not all Feedback Text was related to hints. For example, the characters would say non-hint statements when the student completed or failed a level. The characters would also make non-hint comments if the student clicked on the character immediately after having received a hint, presumably to provide some delay between hints and discourage hint abuse. If the student exhausted the hints or failed, the monster would provide guidance on the solution. Thus, additional binary variables were created to identify the number of hints the student requested and whether he or she got the answer from the monster.

With eight scale positions, each with up to four flower placements, plus four hints and the monster option, 37 variables were coded for use in the study. From a GBA standpoint, this is a small number of variables. This reflects the low autonomy of the game, as well as the fact that each level of the game is identical in terms of possible actions the student can take. If desired, and if theory supported it, one could add more variables representing things like sequence of actions, response times, and game state, but this also increases the risk of adding irrelevant variables that may impact clustering algorithms' solutions and their interpretations. Note also that this is a very simple example for the purpose of illustration, but many games will not have the same gameplay variables or even the same game design between scenarios in a game. The researcher needs to consider the structure of the game to know reasonable dividing points that may require new variable coding and separate clustering solutions, such as changes in levels or sections of a game.

Step 2: Select Clustering Algorithms

When selecting clustering algorithms, it is important that one justifies not just the use of a clustering algorithm as a measurement model, but also the choice of the specific algorithms that will be used in the analysis. When possible, one might frame this rationale in the context of the anticipated data structure, the assumptions of the algorithms, the desired structure of the output, and the research question itself. For example, if identifying outliers is critical for the research, then one should use a clustering algorithm that can identify and isolate outliers from the definition of clusters. As with any research, it is also helpful to consider whether other people have used the algorithm for similar applications.

For this tutorial, the k -means, SOM, and ROCK algorithms continued to be used, and the results of the simulation study in Chapter 5 can help support this decision. First, as investigated by the third research question, the distributions of Kappa values for true cluster recovery can be high in some data scenarios, but there are often cases where all three algorithms have a possibility of converging on a solution that does not represent the true clusters well. For this reason, it is recommended that all three algorithms be used in order to cast a broad net and to increase the likelihood of finding two solutions which validate each other.

Second, the data coding from Step 1 is all binary. Both the ROCK and SOM algorithms work with binary data, and despite being designed for numerical data, the simulation showed that k -means often outperformed both ROCK and SOM in terms of true cluster recovery and correct identification of the number of clusters, as determined

by DBI. As such, all three algorithms may be considered appropriate for analyzing the binary data created in Step 1 from the Beanstalk log file.

Third, Beanstalk is a low autonomy game. The student is limited in the actions he or she can make, and the coding from Step 1 only yields 37 variables—many of which were not active in a given game level (for example, the student can only place flowers on one side of the scale in any given level). Given this knowledge of the gameplay and game design, it was reasonable to expect that there would be small numbers of clusters of work processes in each Beanstalk level, and the probability of observing responses in the relevant variables would likely not be low for many variables. The results of the algorithms' true cluster recovery rates when $k = 4$ and $k = 10$ in Table 20 (reported above in Chapter 5) suggest that both k -means and SOM would perform well under the expected conditions of the Beanstalk data, with the expectation that the k -means algorithm would perform best. The ROCK algorithm is also expected to perform well if the work process data is very similar within a cluster (i.e., students in a cluster have nearly identical work process response data), so it may be a useful algorithm for cluster identification or validation in simple levels of the Beanstalk game.

Step 3. Plan Cross Validation

When using a clustering algorithm to evaluate students' work processes in a GBA, there is a danger that one might overfit the model to the data, and that the clusters may not generalize well to new student responses. One way to defend against this possibility is to cross-validate a clustering solution with subsets from the original dataset to see if the clustering algorithms fits the new data as well as it fit the training data.

The simplest way to do this is to simply split the dataset and keep a subsample as a holdout sample for testing the clustering solution from the remaining training data; however, Baker, Wang, and Andres (2017c) cautioned that, when analyzing repeated observations like multiple attempts at a game, one should make sure that the same students do not have data in both the training sample and the holdout sample, otherwise, instead of testing generalizability to new students, one may in fact be testing the generalizability of new data from the same students. For example, one would not want to have Student A's responses to Level 1 in the training data and his or her responses to Level 2 in the holdout data.

To create the cross-validation holdout sample in the Beanstalk data, one-third of the student IDs were randomly selected from the data, and a new binary variable was added to denote that these students' response rows all belonged to the holdout sample. This resulted in 59 students being identified for the holdout sample, and these 59 students were associated with 7,309 rows of response data across the different levels of Beanstalk. This left 118 students with 13,709 rows of response data for the training data, which would be used to create clustering solutions for each level in Beanstalk.

Step 4: Run the Algorithms

Running the algorithms sometimes requires some decisions and data manipulations on the part of the researcher, specifically around variables to exclude under certain conditions, parameters for the algorithms, values of k to try for the number of clusters, and stopping rules. This section breaks down these decisions below.

Excluding variables. If students' data within a variable do not vary much, the variable may not be meaningful for defining clusters that differentiate students' work

processes. If this happens across the entire game, the variable may not be worth including in the dataset at all, but it is often common to have a variable that varies a lot under some conditions and not others. This is the case in Beanstalk, where students would only place flowers and eggs on the side of the scale opposite of the bugs, which the student was trying to balance. If the bugs are on the left side of the scale on a given game level, the student could only place items on the right side of the scale, thus making any gameplay variables associated with the left side of the scale irrelevant for analysis of that level.

The previous example is a variable made irrelevant by the rules of the game, but variables may have low variation for other reasons. Consider a game where one option is so clearly wrong that nearly no one uses it, or alternatively, an obvious game move that nearly everyone uses. These variables may provide more discrimination in other levels of the game that are being analyzed separately, but they may not be distinguishing features for every game level.

For example, Kerr and Chung (2012) omitted variables from the clustering analysis if the variable was only activated in five or fewer attempts at a level. Given the similar sample sizes (Kerr and Chung had 209 students, the Beanstalk data has 177 students), the same approach was used in the analysis of the Beanstalk data. The one difference was that Beanstalk gameplay variables were excluded from the clustering analyses if five or fewer attempts had that variable activated *or* if all but five or fewer attempts used that variable. For example, if the hint button was used by only five students on a given Beanstalk level, that variable was omitted from the clustering analysis of that level. Conversely, if all the students except for five used the hint button, that variable would still be omitted from analysis. In both examples, the hint button likely would not

provide enough discrimination between students to be valuable in defining clusters of their response patterns.

Parameters for the algorithms. The simulation study reported in Chapter 5 showed the importance of experimenting with different parameters and starting values when using clustering algorithms for analysis. Recall that the *k*-means algorithm, as implemented in R, tries numerous starting centers, randomly selected from the observed data. The *k*-means algorithm then selects the clustering solution that minimized the within sum-of-squares distances for the clusters (R Core Team, 2014). Conversely, the SOM algorithm takes a random set of cases as starting centers, and then adjusts cluster center coordinates based on repeated presentations of the data to the SOM network (Wehrens & Buydens, 2007). The simulation study in Chapter 5 showed that these repeated presentations of data do not overcome the influence of poor starting centers, given the default parameters used in the `kohonen` package, and inclusion of an outlying case as a starting center can drastically change the convergent outcome of the SOM model. For this reason, researchers may wish to follow the *k*-means example, trying multiple random starting centers with the SOM algorithm, then comparing fit to choose the final solution.

The ROCK algorithm produces a single solution for a dataset, given a similarity threshold θ value. The feasibility study in Chapter 3 and the results of the simulation study in Chapter 5 show that the ROCK algorithm often cannot converge on a specific number of *k* clusters under higher ranges of θ values. Researchers who desire a certain number of clusters need to try multiple θ values when creating a ROCK solution. ROCK's ability to identify potential outliers is useful for some applications, but

it is important to have a set number of k clusters when validating the ROCK solution against solutions from other algorithms that have the same number of clusters. For this reason, researchers should be prepared to try multiple theta values in the ROCK algorithm to achieve their desired number of clusters, although as shown in the simulation results in Chapter 5, the ROCK algorithm may not be able to converge even at low values of theta if there are a large number of expected clusters with high within-cluster variability. As with the simulation study in Chapter 5, the ROCK algorithm was set to start at a theta value of 0.95 and decrease in 0.05 increments until the specified number of clusters, k , was attained in the Beanstalk data.

Exploring various numbers of clusters. As discussed in Chapter 3, interpretability of the results is often the key criterion for selecting the number of clusters (Hand et al., 2001; Kriegel et al., 2009; Stevens & Casillas, 2006); however, choosing a clustering solution—or at least narrowing one’s options—is considered a good practice, and many researchers have promoted this strategy or built automated clustering algorithms around this premise (e.g., Baker et al., 2017b; Hamerly & Elkan, 2004; Hothorn & Everitt, 2009; Pelleg & Moore, 2000; Wang et al., 2005). Following this guidance, and in the absence of any subject-matter expertise to guide interpretation of the results, the response data from each level in the Beanstalk dataset’s training data was analyzed with the k -means, SOM, and ROCK algorithms, with the number of clusters ranging from $k = 2$ through $k = 10$. DBI values were recorded for each algorithm’s solution at each value of k .

The upper limit was set to 10 clusters because of the low number of variables used in analysis (37 variables) and the size of the training sample (177 students). If one

assumes that students do not keep trying the same strategy within a level over repeated attempts, then using an upper limit of $k = 10$ implies that the average cluster is expected to contain attempt data from 18 students. Increasing the number of clusters in such cases arguably leads to a level of granularity that would not be useful for analysis of the Beanstalk data, although a researcher may be interested if there is reason to expect large variety in work process response patterns, as might be found in a high autonomy game. A lower limit of 2 clusters also seemed reasonable, since it was known in advance that some students successfully beat levels in the Beanstalk game and others failed them (at least on their initial attempt). This suggested that at least two strategies would be at work in any given level of Beanstalk. Recall that each case is an attempt at a level, so a single student may be represented multiple times (once for each attempt at the level); however, if the student changes his or her work process, each attempt would be assigned to a different cluster.

Stopping rules. The analysis of the Beanstalk data was automated with a “for” loop in R to go through each level of the Beanstalk data, identify the attempts from the students in the training sample, drop variables in that level with low variance in responses (which includes dropping variables for which there is no variance because they were not used in that level), and analyze the remaining response variables with the k -means, SOM, and ROCK algorithms for $k = 2-10$ clusters. The script was instructed to stop analysis and go to the next game level under three scenarios. First, if the number of attempts was less than 25 for a given level, that level was skipped. The cutoff sample size of 25 was chosen based on an inspection of the response frequencies in the data. Of the 93 levels in the Beanstalk data, 88 (95%) had 32 responses or more from the 177 students in the training

sample. There was then a gap in the number of observed responses, with the remaining 5 levels having 13 or fewer responses from the 177 students in the training sample. These samples were deemed to be too small to yield meaningful clusters, and any clusters derived from these small datasets would likely not generalize well to new students. The researcher should decide when or if a variable's observation is too rare to be of interest. Variables that are rarely observed may be of particular interest for identifying nuances in gameplay, but they will not help to define clusters. When in doubt, one can run clustering algorithms with and without the variable in question to see if its presence appreciably changes the clustering solution.

Secondly, if the number of variables selected for the analysis was less than three, then the level was skipped in the analysis. Recall that variables that were activated in 5 or fewer attempts or which were activated in all but 5 or fewer attempts were omitted from analysis because they did not have enough variability in the data to be useful for cluster identification. In some levels, such as the easier initial levels in the game, removing these variables left only one or two variables for analysis. This scenario did not seem to warrant use of clustering algorithms, due to the extremely low dimensionality of the remaining data.

Lastly, the script was instructed to proceed to the next level when k was greater than half the number of unique response patterns. For example, consider a level where eight binary variables have been selected for analysis. Theoretically, one might observe $2^8 = 256$ possible response patterns, but in practice only 10 unique patterns of responses are observed. This step was added after several solutions showed that the clustering algorithms were overfitting the data and creating clusters for one or two outlying

responses, which in turn created a very low DBI value. This choice was made based on the observations in this dataset, but it is not necessary. There will certainly be cases (such as extremely low autonomy GBAs) where a small number of unique response patterns correspond to large, meaningful clusters of gameplay.

Step 5: Validating Across Algorithms

Xu et al. (2013) recommended validating clustering solutions across two or more algorithms. This is good advice, but only if the algorithms are expected to perform similarly under the data conditions. Kerr and Chung (2012), for example, took the opposite approach, taking data from one game level and showing how a hard clustering algorithm did not yield the same interpretable results as the fuzzy clustering algorithm they used in their analysis. Their approach is good for building evidence supporting their choice of the fuzzy clustering algorithm; however, the argument could be stronger still had they demonstrated that another, similar algorithm produced a comparable clustering solution. Just as the holdout sample helps support the generalizability of a clustering solution to new students, one may want to build an argument that the clustering solution also generalizes to new clustering algorithms, effectively demonstrating that two different approaches have identified the same outcome.

The counterargument to Xu et al.'s (2013) recommendation is that it is possible that the best, most interpretable solution can only be found with one, specialized algorithm. This is a possibility, and if that is true, then it puts more pressure on the researcher to test a wide variety of algorithms and defend their choice of the best one for their data. While this is certainly a possibility, this example tutorial follows Xu et al.'s recommendation that two algorithms (both appropriate for the data and yielding

comparable cluster structures) should have relatively high agreement in whichever clustering solution is chosen for evaluating the students' response patterns.

For this study, the k -means, SOM, and ROCK algorithms were run for varying values of k for each level of Beanstalk. The selected clustering solution for each game level was the solution with the lowest DBI value which also had a Cohen's Kappa value of 0.80 or higher with one of the remaining two algorithms' clustering solutions at a given value of k . This process is illustrated by example below.

An example from Beanstalk Level 2-11. Tables 21a and 21b shows an example from Beanstalk level 2-11. Notice that the k -means solution has a slightly better fitting solution (DBI = 0.13) when $k = 5$, but it does match well to the other two algorithm's solutions (Kappa for k -means and SOM = 0.47, for k -means and ROCK = 0.69). The next best-fitting solution occurs at $k = 4$, where the ROCK algorithm produced a solution with DBI = 0.20, and in this case, ROCK and k -means had high agreement: Kappa = 0.95. In this case, the ROCK solution at $k = 4$ was chosen as the clustering solution for Beanstalk level 2-11 (note that in practice, one may wish to compare both solutions with subject matter experts to ascertain which solution is more interpretable in the context of the GBA's intended purpose).

Notice that the decision process used for selecting a clustering solution in Tables 21a and 21b does not require the researcher to use the same algorithm for all levels of the game. This option may be overlooked by many GBA researchers, but there is no reason to use the same clustering algorithm for all levels of a game. After all, gameplay (and the associated response data) may be very different across different levels or scenarios in a game. Just as one might use different IRT models for different item types, one should be

open to using different clustering algorithms for different sections of a GBA. In the example in Table 21, the ROCK solution for $k = 4$ is chosen and it agrees closely with the corresponding k -means solution, and the SOM solution agrees fairly well too. There are other cases in the data where a different algorithm performed well, agreed with one other algorithm, but the remaining algorithm did not agree with the first two. There was no theoretical basis for prohibiting the selection of whichever algorithm could produce the best-fitting solution at each level of Beanstalk.

Table 21a
Algorithms' Agreement (Kappa) by k
(Level 2-11)

k	<u>Cohen's Kappa</u>		
	k -means to SOM	k -means to ROCK	SOM to ROCK
2	1.00	-0.23	-0.23
3	0.37	0.22	-0.31
4	0.78	*0.95	0.74
5	0.47	0.69	0.85

Table 21b
Algorithms' Fit (DBI) by k
(Level 2-11)

k	<u>DBI</u>		
	k -means	SOM	ROCK
2	0.94	0.94	0.77
3	0.81	0.60	0.70
4	0.29	0.53	*0.20
5	**0.13	0.40	0.23

*Best-fitting solution with Kappa ≥ 0.80 for two algorithms.

**Best-fitting solution overall, but Kappa < 0.80 with other algorithms.

Note: Analysis of six gameplay variables, $n = 213$ response patterns.

Consider also the plot of number of clusters (k) and DBI values for each clustering algorithm in level 3-2, shown in Figure 17 below. The clustering solution for this level was the SOM clustering solution at $k = 4$, but if one looked solely at DBI fit and not cross-algorithm agreement with Cohen's Kappa, then one might come to a different decision. The lowest DBI value occurs for the ROCK solution at $k = 5$, but as has been seen in the feasibility study (Chapter 3) and the simulation study (Chapter 5), these low DBI values can be achieved by creating some small clusters and grouping the other cases into a large cluster, which was the case here. One might pick $k = 8$ because all three algorithms have similar, low DBI values, but although they all fit fairly well for those

numbers of clusters, they do not agree with each other (Kappa is less than 0.80). They are each telling different stories about the data, and without expert input, it would be difficult to say whether any of those solutions was useful or interpretable.

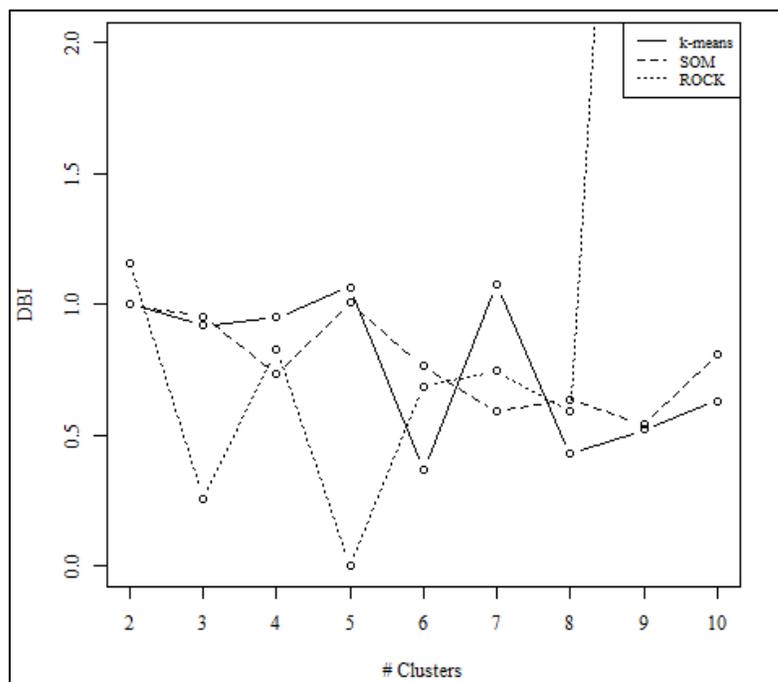


Figure 17. DBI values by clustering algorithm and number of clusters (k) for level 3-2, which analyzed 12 gameplay variables and had 156 attempts in the training data. The SOM solution for $k = 4$ was selected as the final clustering solution because of its low DBI and its high Kappa value (0.86) when compared with the k -means solution

In 51 (68%) of the 75 game levels analyzed, the selected clustering solution was also the lowest DBI observed at all levels of k for that level, meaning that the absolute best-fitting solution had a Kappa value of 0.80 or higher with another algorithm's solution approximately two-thirds of the time. The k -means solution was selected in 44 (59%) of the game levels, the SOM solution was selected in 26 (35%) of the levels, and the ROCK solution was selected in 5 (7%) of the levels.

Step 6: Validating Against Holdout Sample and External Criterion

Once the number of clusters and best-fitting clustering solution (that agrees well with another model) has been selected for each level in the game, one must examine the

validity of those solutions. Ultimately, each clustering solution should be interpretable by subject matter experts; however, validating with the holdout sample builds evidence of the clustering solutions' generalizability to new students, and validating against a criterion measure builds evidence of the solutions' accuracy and utility.

For the Beanstalk data, each level's clustering solution from the training sample was used to predict cluster membership for each response record in the holdout sample. The *k*-means, SOM, and ROCK functions in R each have the ability to predict cluster membership for new data, given a solution from training data (Buchta, & Hahsler, 2014; R Core Team, 2014; Wehrens & Buydens, 2007). The DBI values for the training data's clusters were compared to the DBI values from the clusters assigned to response data in the holdout samples. Figure 18 shows the DBI comparisons for the training data and the holdout sample for each of the 75 game levels in Beanstalk, with game levels listed in descending order by the training data's DBI. Notice that for most levels, the DBI value of the holdout sample's clusters was close to or lower than the DBI from the training sample. Exceptions occur at levels 2-7 and 1-5, where the DBI value for the holdout sample is noticeably higher than the DBI from the training sample. This suggests that these clustering solutions are overfitting the training data. Very low DBI values in the holdout sample are also noteworthy, but not necessarily cause for alarm. When the DBI value in the holdout sample is much lower, it indicates that certain clusters were not as prevalent or did not appear at all in the holdout sample; i.e., there were work processes observed in the training data that were not observed often in the holdout sample.

If the clustering solutions have similar fits to both the training and holdout samples, then one can validate the clusters against an external criterion; i.e., a related

measure that was not included in the clustering algorithms' analyses (e.g., Stevens & Casillas, 2006). This can be any meaningful criterion measure, such as scores from other tests, results from cognitive interviews, student engagement observations, or personality measures. Correlations between the criterion measures and the clusters help with interpretation and to validate the clusters as assessment outcomes.

A very simple validation criterion is simply the outcome of the game levels (i.e., win/loss, correct/incorrect). Stevens and Casillas (2006) looked at percentage correct as a criterion validation for their SOM analysis of a biology GBA. Kerr and Chung (2012) took a similar approach, but instead, they adjusted the number of clusters until clusters appeared that had only correct results stemming from the work processes. Kerr and Chung's approach does not count as validation because they specifically calibrated their cluster solution to match the correct/incorrect result outcomes, but the underlying concept is the same as Stevens and Casillas' approach: if students in a cluster are engaging in the same work process, then the GBA should give those students the same outcome.

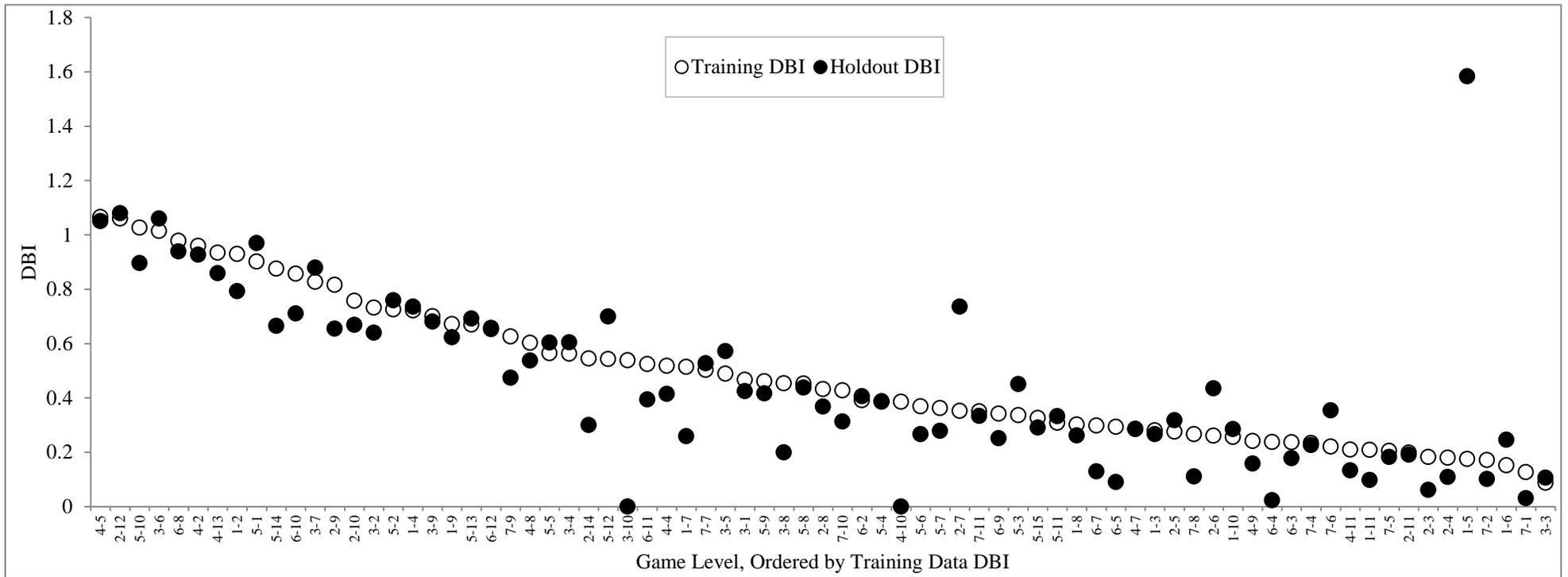


Figure 18. DBI comparisons between training sample and holdout sample. When the holdout sample's DBI is much higher than the training sample, it indicates that the clustering solution is overfitting the training data and a new clustering solution should be identified. When the holdout sample's DBI is much lower than the training sample, it suggests that some work processes observed in the training data were not observed as frequently (or at all) in the holdout sample.

Descriptive statistics were run for each cluster at each level using all cases in the dataset (both the training sample and the holdout sample). This provided a view of the average response patterns in each cluster as well as the percentage correct attained by each cluster. Across all 75 analyzed Beanstalk levels, 358 clusters were created, with each level having between 2 and 10 clusters. Out of all of the clusters, 297 (83%) reflected strategies that generally led to an incorrect or correct outcome: these clusters' response patterns yielded correct outcomes more than 85% of the time or less than 15% of the time. In inspecting the clusters where the correct or incorrect outcome happened less consistently, response patterns show students figuring out some of the pieces of the correct solution, then trying different ideas to finish the level, with mixed results.

An example from Beanstalk Level 2-9. To understand how one might validate against a criterion like whether the student beat a game level, consider Beanstalk level 2-9, which used a six cluster SOM solution (DBI = 0.82, Kappa with *k*-means solution = 1.00). Figure 19 shows the correct outcome rate for each of the six clusters in Level 2-9. Each cluster is given an arbitrary ID, thus labeling the clusters as Clusters 1 – 6. Notice that Clusters 5 and 6 have nearly perfect correct outcome rates—students in these clusters nearly always beat the level (the difference being that everyone in Cluster 6 got the answer from the Beanstalk monster after having an incorrect outcome on their previous attempt). Cluster 4, however, is not so clear-cut. In Cluster 4, 38% of the students using this work process beat the level, but 62% did not. How can the same work process beat the level for some cases, but not others?

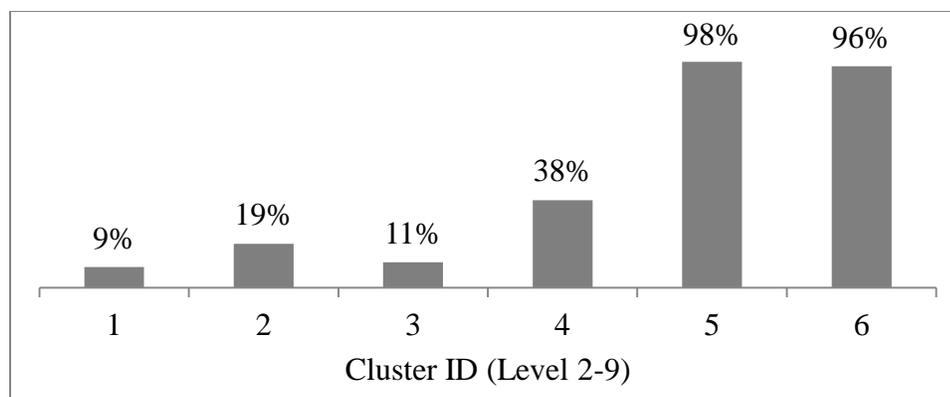


Figure 19. Correct outcome rates (rates of beating the level) for the six clusters identified in level Beanstalk 2-9.

The answer lies in the response variables themselves. In Cluster 4, most students got part of the solution and then experimented to find the correct response. Beanstalk allows students to “trim” their flowers, effectively letting them undo a move and try something different when they see that the scale is not balancing well. Some of these students were able to eventually figure out the solution by observing the feedback onscreen (the scale balancing) and using that to inform their experimentation. There are several ways to solve Beanstalk level 2-9, but all solutions require that the cross product of the weights and locations of the flowers sums to 11.

The defining feature of the cases in Cluster 4 for Beanstalk level 2-9 is that nearly every student placed two flowers at the end of the scale, four units from the fulcrum (see Figure 20a below). This flower placement yields a cross product of $4 \times 2 = 8$. In this level, the student’s final solution could only have four flowers in it, so for students who placed two flowers four units from the fulcrum, the remaining 3 points of the sum of the cross products had to come from the placement of one or two other flowers. Given the placement of the initial two flowers, there are only two solutions to the level that fit those criteria, shown in Figures 20b and 20c.



Figure 20a. Cases in Cluster 4 in level 2-9 were characterized by students who placed at least two flowers at the end of the scale, four units from the fulcrum. From this position, there are two ways to balance the scale (Figures 20b and 20c), but only 38% of students with cases in Cluster 4 identified a correct solution from this position.



Figure 20b. One possible correct solution for Cluster 4 in level 2-9 for students who have placed two flowers at the end of the scale, as depicted in Figure 20a. Note that the student is provided with four flowers, but this solution only requires three.

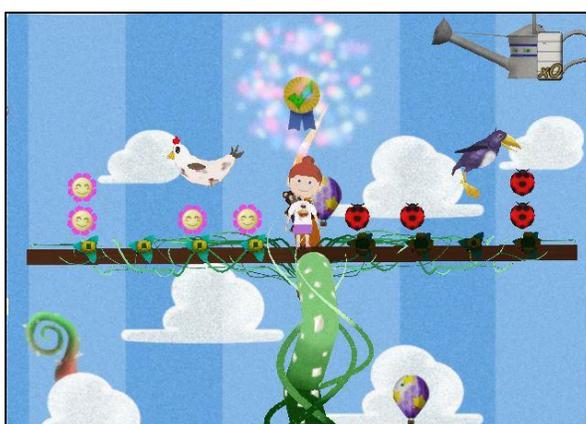


Figure 20c. The other possible correct solution for Cluster 4 in level 2-9 for students who have placed two flowers at the end of the scale, as depicted in Figure 20a. Note that in this solution, the student is simply mirroring the positions of the ladybugs.

Given that only 38% of cases in Cluster 4 (on level 2-9) were correct, one can infer that the remaining 62% of cases placed two flowers on the end of the scale and then deviated into another final placement. Figure 21 shows the distribution of flower placements for Cluster 4 in Beanstalk level 2-9, which could be stacked on top of each other up to a height of four flowers. Students with incorrect solutions continued stacking

higher, especially at units 3 and 4, which mirrors solutions in preceding levels. This explains how all these cases could be in the same cluster (a shared response pattern on two flower placements) even though only a minority of cases were correct—the differentiating factor is the placement of the remaining two flowers.

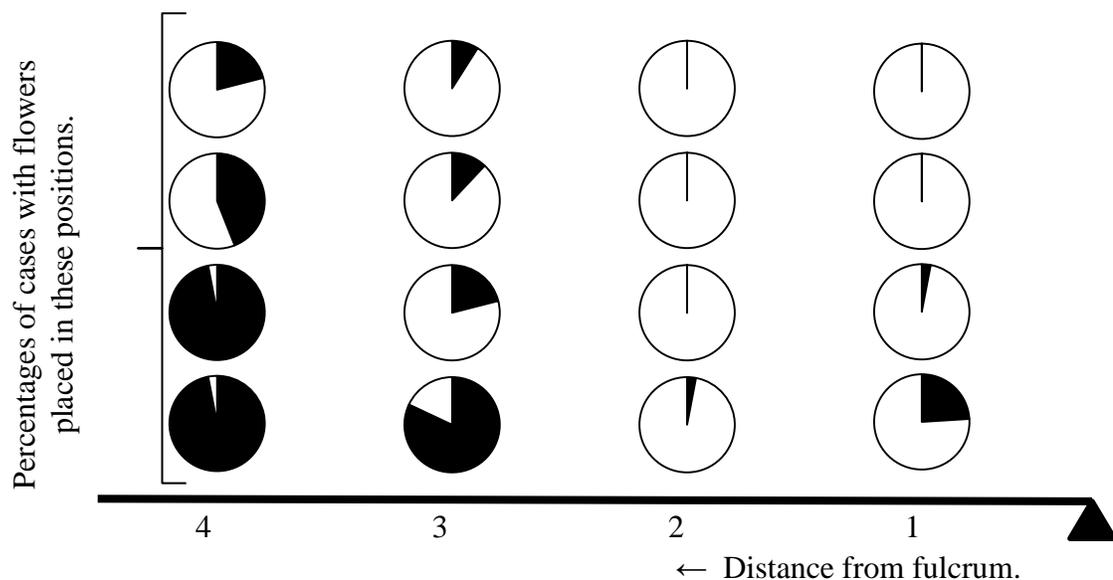


Figure 21. Flower placement rates for Cluster 4 in level 2-9. Nearly all responses in this cluster have two flowers placed 4 units away from the fulcrum (depicted on the right as a triangle). The sum of cross products must equal 11 to beat this level. Figures 20b and 20c above show the possible correct placements, given the placement of two flowers on the left of the scale.

This phenomenon was observed across many clusters where students did not have direct, concise solutions. Students in these clusters shared the key gameplay attributes that defined the cluster strategies, yet a minority of these students were able to turn that strategy into a correct solution. Whether that was luck or intelligent self-correction is impossible to say without looking at the students' gameplay across levels using a method like Bayesian Knowledge Tracing (e.g., Baker, Wang, & Andres, 2017a).

This already points to a weakness in using the clustering algorithms as the measurement model to evaluate the Beanstalk data, namely that, based on gameplay

alone, the clusters do not provide information about the work processes that is instructionally valuable; the clustering algorithms do not do a good job at differentiating between students who are self-correcting and getting to the correct answer versus students who are simply lost. This has everything to do with the game design's feedback. The scale balances in real time as more flowers are placed, and students have the opportunity to undo their moves and change course.

In this example, validation with an external criterion (the correct/incorrect outcome of the level) has shown that the clusters align well with performance and do a good job at differentiating between students who efficiently solve the level versus those who cast about a bit. In this sense, the clusters are doing a good job at differentiating between novice and expert gameplay, which was proposed in Chapter 1 as one of the main reasons someone would want to evaluate work process data in the first place. The external criterion, however, also shows that there are degrees of separation within the novices, and some novices are able to use the game feedback better than others. The clusters do not differentiate between these novices well. If one were interested in identifying such novices, one would likely need to include variables for sequence of response and game state to show the order of actions and how the student moved toward a correct response after incorporating the feedback data of the game state.

Step 7: Interpreting, Labeling, and Grouping Clusters

In a GBA, this step is best executed by subject matter experts who interpret the clusters in the context of the assessment's domain model. Both Kerr and Chung (2012) and Stevens and Casillas (2006) did this, linking the clusters back to behaviors that they associated with certain skills or misconceptions related to their games' measured

constructs. In a long game with high autonomy, this may be a tedious process, but there is no way around it—a human being must interpret the clusters and assign them meaning in the context of the construct. In this tutorial example, however, such experts are not available. For the purposes of illustration, this section simply focuses on the gameplay itself.

The work processes represented by each cluster can be interpreted through the variables that have high or low rates of occurrence in each cluster. For the Beanstalk data, recall that the clustering algorithms analyzed coded variables representing game actions (placement of flowers on the scale), hints, and use of the monster for getting the correct answer. These three groups of variables can serve as a rough basis for interpreting, labeling, and grouping clusters.

In this example, clusters were given one of four possible labels:

1. Correct – the student beat the level immediately with few missteps and hints.
2. Correct with Giveaway – the student beat the level, but only with help from the monster, who provides guidance on a solution.
3. Incorrect with Use of Hints – the student often did not beat the level, but made use of the hints in his or her efforts.
4. Incorrect – the student typically did not beat the level and made several missteps, but the student did not make use of the hints or have help from the monster.

Using common labels like these for the clusters is useful for analyzing gameplay across levels. One can see if students jump between cluster types or if certain clusters are

likely to result in another cluster type later in the game. Stevens and Casillas (2006) called this “strategic transition,” and mapped clusters to four different strategy types. They analyzed students longitudinally to demonstrate that students’ clusters advanced to different strategies as they played the game, and they used these findings as evidence that the GBA itself was helping students learn.

Recall from Chapter 2 that one of the benefits of using a game as an assessment is that it encourages students to keep playing, even if they fail at first. Shute and Wang (2016) explained that games should be challenging in order to be fun, and students expect to fail sometimes in a game. Repeated gameplay allows the researcher to map the evolution of a student’s gameplay as a series of failures and successes over time. This series becomes more meaningful and easier to interpret when one creates a unifying method for labeling clusters by strategy type across the entire GBA, even if there are multiple levels and scenarios. To put it another way, each level should not be interpreted in isolation. One should interpret the student’s path from the start of the game to the end. This necessitates cluster labels that can provide meaning about a student’s transition from one cluster to another, both within and between levels.

In the Beanstalk analysis, a cluster was labeled as “Correct” if at least 85% of the cases beat the level and there was no use of the monster for guidance. Clusters were labeled “Correct with Giveaway” using the same criteria, except that the attempts utilized the monster to find out the answer (either by exhausting the hints or following a failure on a previous attempt at the level). A cluster was labeled as “Incorrect with Use of Hints” if the correct outcome was less than 85% and at least one hint was used. All remaining clusters were labeled as “Incorrect.”

Step 8: Students' Changes in Cluster Membership Over Time

Once the clusters have been labeled and grouped into similar types, the students' pattern of clusters can be investigated, but this is optional. For the purposes of assessment, one could simply keep a student's cluster assignments for each level or scenario in the GBA as the assessment outcomes, but many games are not designed this way. Recall in the previous section that game levels are rarely designed to function in isolation. Instead, the levels in the game act as a structured path to move the student through the overall game arc, often moving through different concepts or getting more difficult as the game progresses in order to keep the gameplay challenging and fresh. Beanstalk is no different. In Beanstalk, the game levels progress through Siegler's (1976) and Siegler & Chen's (2002) mental models relating to the scale balancing rules, and the game levels get progressively more difficult in terms of both psychometric difficulty and game difficulty as the student progresses.

Although this step is optional, it is clear that ignoring the entire pattern of gameplay across the game is leaving some potentially valuable information on the table. For example, the pattern of clusters within and across game levels can show if the student is learning, if the student is adapting strategies, or if the student is gaming the system. If the value of looking at work processes is to differentiate between novices and experts, and if the value of using a game as an assessment is to engage students so that they keep playing and improve their performance (as is argued in Chapters 1 and 2), then the researcher should feel obligated to examine whether students are moving from novice to expert work processes over the course of gameplay.

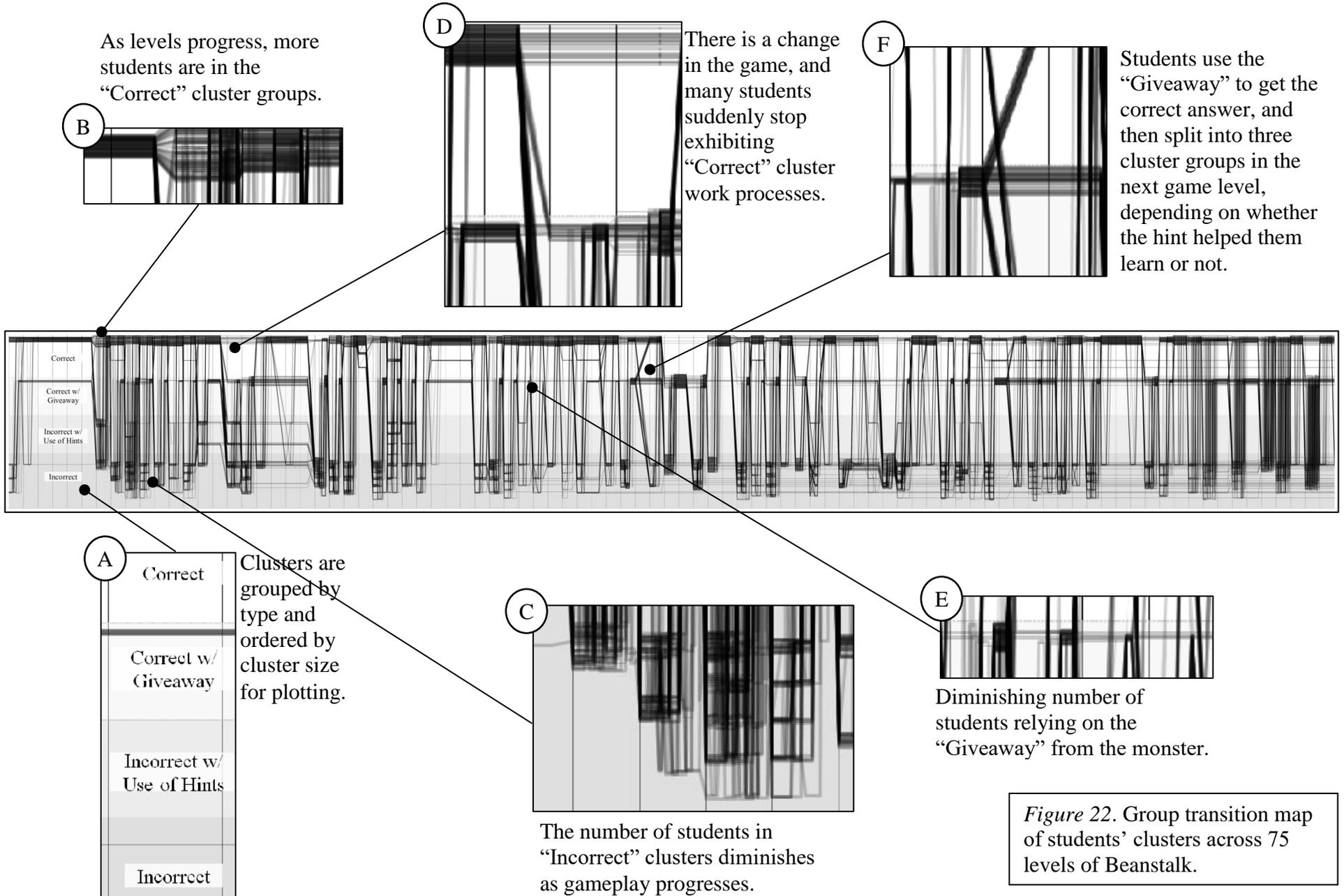
A simple way to investigate this is to plot students' paths through the clusters across all the levels of the game. Stevens and Casillas (2006) called this a *group transition map*. Figure 22 plots students' paths between the four types of clusters as they progress through the game, with game levels increasing along the horizontal axis.

The group transition map in Figure 22 shows high level patterns of student gameplay. The vertical axis represents the four types of clusters identified in the previous section: Correct, Correct with Giveaway, Incorrect with Use of Hints, and Incorrect. The horizontal axis represents the game levels of Beanstalk. The following gameplay features are noted with callouts on the figure:

- A. Within each horizontal band, clusters are ordered in terms of size, with larger clusters appearing at the top of a band and smaller clusters appearing at the bottom of the band. Thus, students' lines converging in a group on the bottom side of the Correct band represent students who used a rarer, perhaps less-obvious solution.
- B. In this game level, there are suddenly more lines in the Correct cluster band. More students are catching on and are easily beating this level.
- C. Early in the game, there are a variety of Incorrect clusters where students are not making use of the hints. These clusters have fewer students as the levels progress, suggesting students are learning to use the hint features or are solving the levels.
- D. At this point, something changed in the game and the strategies that had been working in earlier levels no longer apply. Many students drop down into the Incorrect with Hints and Incorrect clusters, although a few students are still present in the Correct clusters.

- E. These are clusters of students using the monster to get the correct answer. There are fewer and fewer students in these clusters as the levels progress, suggesting that many students do not keep relying on the monster to beat the levels.
- F. This cluster provides a clear look at the effectiveness of the monster's giveaway. This is a large Correct with Giveaway cluster where the student's got the correct answer from the monster. The students in this cluster go in three different directions after this level. Some students learn from seeing the solution and move up to the Correct clusters. Some students did not quite learn the solution strategy, and drop back down into the Incorrect clusters. Some students continue on and use the monster again in the next level.

The paths shown in Figure 22 are not just useful for investigating aggregate gameplay patterns. They can also be used for individual student reporting. In a simple game like Beanstalk where one has grouped clusters into four standard categories (as is the case above), little needs to be done to support individual student reporting. A researcher can report a single student's path through the game, tying level concepts to clusters and identifying where students struggled, where they changed processes, and where they ultimately succeeded.



Tutorial Example Summary

The tutorial provides an overview of the types of decisions, validations, analyses, and aggregations that must be made to begin to treat a game as an assessment. Even with a simple game like Beanstalk, there are many choices that must be made about how to select and code variables, choose the algorithms, choose the number of clusters, pick a final clustering solution, and interpret the results. Following an argument-based validity approach, one should be prepared to defend each of these decisions in the context of the inferences and uses of the game as an assessment. The steps demonstrated here will not generalize to every GBA design or clustering algorithm, but they frame the way one must think about using a clustering algorithm as a measurement model in a GBA, and from this perspective, these steps can serve as a rough guide to GBA researchers.

The tutorial showed that the first step, recoding and selecting variables, requires human guidance, which should imply that there is a theory or rationale at work from the start. Although the clustering algorithms are designed to uncover hidden patterns in large datasets, an assessment researcher is not absolved of his or her responsibility to link evidence to theories about constructs. The data fed to clustering algorithms will affect the quality and utility of the cluster outputs, so this is a critical step and should be treated as part of the evaluation component of the evidence model.

The tutorial also showed the potential value in validating results across different samples of students as well as different algorithms. Cross validating across groups of students helps build a case for the generalizability of the clustering solution to new students, whereas validating across cluster algorithms helps ensure that the result would generalize to other similar, appropriate clustering algorithms. The tutorial demonstrated

that using multiple algorithms for validation helps cast a broader net of solutions from which the researcher can choose, and it yields the possibility of switching between algorithms for different levels or sections of a GBA.

The tutorial also showed a critical weakness of clustering algorithms as measurement models in GBAs, namely that the clustering aggregation can mask meaningful, subtle differences between students' work processes. This possibility was underscored in Amershi and Conati's (2009) cautions against using clustering algorithms to evaluate students. The distribution of responses in Figure 22 in Step 6 above showed that the single cluster consisted of students who shared many aspects of a common response pattern, yet only a few of them were able to determine the correct solution. What made these students different, and were they perceptive or just lucky? The cluster obfuscates these potentially meaningful differences, yielding a coarse and perhaps misleading label to all of the work processes assigned to it.

These lessons illustrate why a clustering algorithm may be a poor measurement model for evaluating a student work process: they are simply too dull an instrument to provide defensible evaluation of a student work process in an assessment context. This does not, however, preclude their use for assessments. The inferences made from mapping changes in students' clusters over time (such as those discussed in Step 8 above) may still provide meaningful insights into a student's performance across multiple attempts or progressions of gameplay. These collections of clusters representing changes in work processes over time can provide a far richer picture of a student's gameplay in a GBA and can arguably serve to better differentiate between students with respect to the construct of interest.

Chapter 7: Discussion

This research sought to explore whether clustering algorithms could serve as measurement models in GBAs, and if so, the practical steps researchers should take to support the algorithms' use and to validate their results. Games are appealing as educational instruments because they are engaging, and their design encourages repeated gameplay that helps educators or researcher observe changes in work processes over time. The observations of work process data are useful for differentiating between novices and experts or to illustrate a student's path from novice to expert. By making repeated processes into a game, the student may be more engaged and more willing to participate for a longer period of time. Using a game as an assessment, however, presents challenges, because the assessment now includes construct-irrelevant relatedness elements that might impact performance without necessarily representing anything about the construct of interest. Performance on GBAs may also be dependent on both psychometric difficulty and game difficulty, and there may be no way to parse these concepts out when evaluating student performance. Nevertheless, GBAs may be valuable educational tools when used in low-stakes scenarios, and careful game design, evidence identification, and selection of measurement models can yield informative and valid GBAs.

Summary of Results

The findings of the simulation study underscore the need for selecting appropriate clustering algorithms for the GBA data and cross validating over multiple algorithms and datasets. The use of clustering algorithms as an assessment measurement model can be risky if the GBA inference occurs at the student level. For example, one might assign

students to a cluster based on the algorithm's analysis of the work process response data, but the inference may not be valid for outliers. If student-level reporting is needed, researchers may be able to provide a more valid and meaningful evaluation of a student by considering the student's clusters over multiple instances of gameplay rather than placing too much emphasis on the cluster membership of a single instance of a work process.

When considering the individual algorithms, the *k*-means algorithm performed unexpectedly well. In Chapter 2, the limitations of the *k*-means algorithm were discussed, and based on Berkhin's (2006) and Robinson's (2017) warnings, it was thought that *k*-means would perform poorly when applied to the sparse, binary datasets in the simulation and the tutorial. This was not the case. The *k*-means algorithm had been included principally with the intention of showing how this popular and easy to access algorithm could yield misleading results, but instead, the *k*-means algorithm often outperformed the more complicated SOM algorithm and the binary-specific ROCK algorithm. In the Beanstalk tutorial, 59% of the clustering solutions selected were generated by the *k*-means algorithm. In the simulation study, *k*-means was the most versatile algorithm, consistently recovering the true clusters in the most cells and identifying the correct number of clusters (using DBI) more often than the SOM and ROCK algorithms.

Why did the *k*-means algorithm perform so well when Berkhin (2006) and Robinson (2017) predicted it would perform poorly with binary data in non-spherical, differently-sized clusters? The simple explanation is the *k*-means software's design of trying multiple starting center choices, creating multiple solutions from them, and then returning the solution that minimized the within cluster sum-of-squares (R Core Team,

2014). To put it another way, *k*-means has the advantage over algorithms like SOM because it is actually making multiple attempts and cherry-picking the best one. It may also be that the conditions under which Robinson (2017) warns of *k*-means failure must also be vastly more extreme than what was created in this simulation study. Nevertheless, *k*-means' use and comparisons of multiple starting center choices provide two pieces of guidance:

1. When clustering algorithms' solutions are heavily based on the starting centers (e.g., *k*-means, SOM), researchers should try multiple starting center choices and systematically compare the fit of the competing solutions.
2. Extending that idea, when a clustering solution is heavily dependent on parameters for the algorithm (e.g., SOM's neighborhood radius and learning rate, ROCK's theta value for neighbor similarity), researchers should also systematically experiment with different parameter values and compare the fit of competing solutions.

Despite Robinson's (2017) warnings against the *k*-means algorithm for binary data, and given the results of the simulation study and that the *k*-means algorithm is simple to access, use, and understand, researchers should consider using it (or one of its derivations) as one of their clustering algorithms when analyzing binary GBA work process data.

The SOM algorithm performed well in many cases, but was susceptible to creating outlier solutions that did not recover true clusters well in the simulation. This was likely caused by a poor selection in starting centers, and the SOM algorithm performance could be improved by trying multiple starting centers and selecting the best-

fitting solution, as is the strategy used by the k -means algorithm in R (R Core Team, 2014). Researchers using SOM may also want to consider experimenting with different values for the SOM parameters, such as learning rate, neighborhood radius, grid dimensions, toroidal structure of the grid, and the number of data presentation iterations. The complexity of this algorithm, however, may make it difficult to use for some researchers, and when the GBA data scenario is similar to those in the simulation study, researchers may find that fine-tuning an SOM model does not offer noticeable benefits over the simple-to-use k -means algorithm,. Nevertheless, the SOM algorithm may be better-suited for more complex data scenarios and can be a useful second algorithm for validating results from the k -means algorithm.

Despite being designed for sparse, binary datasets, the ROCK algorithm is probably the least useful of the three algorithms studied for general GBA analyses. Only 7% of the solutions used in the tutorial were generated by the ROCK algorithm. The ROCK algorithm did, however, show excellent true cluster recovery in the simulation data when response probability was 0.9, and the ROCK algorithm also performed the best when there were a large number of clusters ($k = 35$). The ROCK algorithm may have also been at a disadvantage, since it is well-suited for identifying outliers and isolating them into many small clusters. When forced to converge on a smaller number of clusters, the ROCK algorithm must increase its threshold for similarity, which can result in many cases being assigned to a handful of clusters. There are certainly GBA scenarios where the ROCK algorithm may perform very well; however, researchers should be familiar with the limitations of ROCK and apply it only when it seems suitable for their data conditions or when identifying outliers is of interest.

Study Limitations

The study suffers from several limitations. First, it assumes that each cluster is defined by a high probability of a response in 20 variables and that that probability of a response is the same level for all clusters. In real world GBA data, clusters will likely have more variability, with clusters having different numbers of relevant variables and varying levels of response probability for those relevant variables. While this study's simulated data may not reflect that kind of diversity within a dataset, the controlled cluster structure has the benefit of allowing for investigation into the impact the other conditions have on the accuracy of the clustering algorithms' solutions.

Second, the fact that fit and interpretability are not synonymous emphasizes the limitation of using DBI as an evaluation of the clustering solutions. Just because a clustering algorithm can return the best-fitting solution does not mean that it will always be the most useful algorithm. While this study focused on the best-fitting solution, recall that *k*-means and SOM can arrive at different solutions based on the random cases that are chosen for the starting centers. If one were to collect multiple solutions from these two algorithms, one might observe groups of alternative solutions that may not be the globally best-fitting but which still reflect a local minimum in the DBI measures across solutions. These solutions might represent alternative interpretations which may or may not be useful for application.

The study also only represents data that have been coded with structured logging (Kerr & Chung, 2012); i.e., the simulated data and the tutorial data do not represent every mouse click and key stroke a student made. Binary datasets with these numbers of dimensions are far more likely to reflect logging in which a subject matter expert has

made a decision about which actions should be recorded, perhaps even combining variables into a new variable before the analysis. This is fine for representing many simple GBAs, but it may not be a realistic depiction of the vast datasets that could be generated in complex, high autonomy GBA environments like flight simulators. Similarly, this study is limited to only binary data, but some GBA researchers may be working with other values that better represent sequences or magnitudes of actions or game state.

Another limitation is the dependence between the number of clusters and the total number of variables in a dataset. With more clusters, more variables must be added to create new relevant variables for each cluster, but this means that the datasets with more clusters have more variables than those with fewer clusters. Kriegel et al. (2009) discuss the difficulties of identifying clusters when there are a lot of variables, although they were specifically discussing datasets with thousands or hundreds of thousands of variables (the simulation datasets never have more than 1,000 variables). Nevertheless, it is difficult to ascertain whether differences in algorithms' performance are due to changes in dimensionality (total number of variables) or differences in the number of clusters and their overlap. It is anticipated that the former has been the clear driver, but the differences in the number of variables may still limit interpretations. Despite this limitation, there are practical benefits to the decision to let the total number of variables vary: clusters in the study all have the same number of relevant variables, which supports comparisons, and smaller datasets were not padded with extra irrelevant variables, which would have hindered the identification of clusters.

Although the study manipulates several conditions of the simulated data, one must also consider that there are other aspects of the analysis that could be manipulated, particularly in the algorithms themselves. The SOM algorithm might return different results based on the starting values selected for the cluster centers, as well as the learning rate and neighborhood size. A researcher implementing the SOM algorithm might be well-advised to manipulate some of these variables to see how they impact the clustering solution. Additionally, the study only looked at three axis parallel soft-projected clustering algorithms. There are many other algorithms that could be compared, as well as methods that use probabilistic assignments to clusters, such as fuzzy clustering and latent class analysis.

Finally, the use of DBI as a measure of fit appears to have some drawbacks. The DBI values can be misleadingly low if the clustering algorithm creates a small cluster around a few outliers, and when selecting the number of clusters, DBI often results in an overestimation of the number of clusters that should be selected. Baker et al. (2017b) recommend using AIC or BIC for selecting the number of clusters, and algorithms like those developed by Pelleg and Moore (2000) and Wang et al. (2005) may be good options, especially if the researcher does not have a subject matter expert's help in interpreting clusters. Validating across several appropriate clustering algorithms and cross validating with other samples of data can also help avoid selecting a clustering solution with too many clusters that over-fits the training data. For the purposes of assessment, interpretability is the ultimate goal, and researchers should be able to defend the theoretical implications of their final clustering solution, without relying solely on measures of fit.

Implications

This study provides operational and analytical guidance for researchers pursuing empirical approaches to evaluating response patterns. While the context is primarily the use of GBAs, these findings also inform similar measurement efforts in other assessment types like simulations, automatically scored essays, or intelligent tutoring systems. Like GBAs, these complex assessment designs can yield high-dimensional response data that may need to be evaluated and categorized with an exploratory approach if there is high autonomy in these instruments. If meaningful interpretations are theorized to exist in subspaces of the response variables, clustering algorithms may be useful tools for these other instruments, just as they can be for GBAs (Kriegel et al., 2009).

When using an exploratory empirical approach—such as a clustering algorithm—to evaluate students' knowledge, skills, or abilities, it is recommended that the clusters be validated with several algorithms (Xu et al., 2013). If reasonable agreement does not exist between the clustering solutions, then one cannot be confident in individual classifications of students, and these methods may only serve to describe aggregate patterns. This study demonstrates how researchers should compare clustering solutions to validate their results, and it does so using three clustering algorithms (SOM, *k*-means, and ROCK) that are well-suited to the sparse, binary datasets common to GBAs.

At a minimum, this study exposes the relative strengths or weakness of these three algorithms for clustering sparse, binary datasets that would be typical for a GBA. The varying levels of the simulated data also point to cluster structures that are dependably identified between these three clustering algorithms. These findings will help inform GBA researchers' decisions about when to use these clustering algorithms, especially

with regard to their sensitivity to varying sized clusters and clusters that overlap due to the sparseness of the data—common scenarios in GBA data. These findings will also help inform GBA design, perhaps guiding the design of adaptive rules to help focus response patterns into well-separated clusters.

From a practical perspective, these inquiries have not necessarily cleared a path for GBA use or the use of clustering algorithms in GBAs. The use of a game as an assessment creates many opportunities by encouraging repeated gameplay and adapting difficulty to student abilities, but the introduction of construct-irrelevant relatedness elements and the potential tangling of psychometric difficulty versus game difficulty almost certainly preclude a game from any high stakes application. There is also the question of whether a game is appropriate for a high stakes decision, or whether high stakes assessments should be constrained to only serious, sterile assessment instruments.

The practical aspect of clustering algorithms and GBAs may also be scrutinized simply on the basis of the complexity of using a clustering algorithm. Although some clustering algorithms like *k*-means are easy to implement, the steps for using a clustering algorithm as a measurement model (Chapter 6) are far more complex—perhaps more than an assessment researcher would be willing to take on in practice. It may be that clustering algorithms should be relegated to a purely investigative, exploratory role in GBA design, but that the measurement model itself should retain a principled approach, which was method used by Corrigan et al. (2014). In short, even if one can build a strong validity case for a clustering solution for evaluating student work processes, the complexity required may simply be too daunting.

One can also argue that clustering algorithms are able to serve other roles as evaluation components in an evidence model, rather than measurement models themselves. Clustering solutions are valuable tools for evaluating the evidence from the work process, and the cluster IDs can feed into a non-clustering measurement model. The group transition map in Chapter 6 and the discussion of evaluating students' changes in cluster membership over time sets the stage for systematically evaluating a student's string of clusters as its own work process.

Based on the results of the simulation study (Chapter 5), future research needs to focus on fit-based methods for selecting the number of clusters. DBI appears to trend toward overestimating the number of clusters needed. Comparisons of DBI to AIC and BIC for selecting the number of clusters would be invaluable, and practitioners will benefit from examples of how to implement the findings. Just as this research recommended that researchers choose clustering algorithms that are appropriate for their data, researchers will find that they also need to be able to defend their choice in fit measures, especially if the fit measure is being used to pick the number of clusters or to compare competing cluster solutions from different algorithms.

It is also important for future research to investigate other conditions for the simulation study, such as clusters with different numbers of relevant variables or datasets with dozens of more clusters. This is in part because the direction of the research must lay the groundwork for vastly more complex games and technology. The GBAs discussed here are relatively simple, but educational game designers continue to develop more complex and (arguably) interesting games. Harnessing such games for assessment requires measurement research to stay abreast of game technology. If there is a desire to

make meaningful evaluations of students based on gameplay, researchers must prepare for measurement challenges in large games with many types of work processes and games that do not end (e.g., virtual worlds that students join and participate in as they wish). For those who seek to interpret gameplay as an assessment inference, the expansion of game capabilities will increase the complexity around foundational assessment practices like identifying the boundaries of domains and quantifying reliability.

These obstacles are not insurmountable, and the most valuable contributions will be made by researchers sharing their rationales, methodologies, and toolsets with each other. The inputs to the simulation study (Chapter 5) and many of the recommendations in the tutorial (Chapter 6) were culled from researchers who shared their process behind creating a clustering algorithm measurement model for an educational game, in effect creating a GBA. By continuing this kind of research and experience-sharing with more complex games, the lessons learned will help to inform future tutorials and best practices. Documentation that discusses both the GBA design and the development of a measurement model will be the most important, as ECD shows that these processes are dependent on each other for creating validity arguments.

Future of GBA

The use of games as assessment instruments and of machine learning techniques like clustering algorithms as assessment measurement models is exciting and represents a new frontier of educational assessment; however, one must not lose sight of assessment researchers' responsibilities to support validity and reliability of inferences. Machine learning in non-assessment applications (e.g., targeted online advertising) may have more

flexibility to be wrong or to improve slowly over time. For example, if using machine learning to recommend products to people browsing a website, the data scientist only needs to get a handful of suggestions right for that application to be useful, and there is little risk should the customer get an irrelevant recommendation. This is not the case in assessments. Researchers need to be able to make accurate inferences in the majority cases, and widespread inaccuracy in a machine learning output in a GBA precludes its use for assessment. Assessment researchers who are excited and tantalized about the power and flexibility of clustering algorithms will do well to balance that enthusiasm with a sobering reminder about assessment developers' responsibilities to their stakeholders to support valid inferences and uses of the results.

Similarly, assessment researchers using machine learning algorithms do not have the same freedom of data use as most data scientists. In many non-assessment applications, prediction or classification accuracy is prized, and any data that improve prediction and classification can be included in the machine learning model. Conversely, assessment researchers still need to be able to build a validity argument, which will include (among other things) a logical rationale for why certain data were included in the evidence accumulation for the measurement model. In short, assessment researchers cannot throw in variables to improve the model without also being able to explain how they link back to the domain of behavior associated with the construct.

Researchers must also consider the applications of GBA. For example, could GBA ever be used for a high stakes application? The simulation results in Chapter 5 showed that under some favorable conditions, clustering algorithms are able to recover true clusters with very high accuracy, but one must wonder if such results could be

expected in the field. Both Kerr and Chung (2012) and Stevens and Casillas (2006) reported identifying clusters that could not be easily interpreted in the context of their constructs, meaning one might reliably identify a cluster, but have no idea what it means. The relatedness elements designed to captivate students may actually turn off some students and cause them to disengage, or students may engage in off-task behavior like free-roaming the game. Admittedly, such behavior exists in traditional assessments, like when a student fills in an answer sheet with a pattern or draws a picture on an essay question, but it seems reasonable to expect that the lack of seriousness and potential higher autonomy in a game may invite a broader range of responses that may not be interpretable in the context of the construct of interest.

The potential for uninterpretable or irrelevant gameplay may be enough to preclude GBA from high stakes purposes, but one must also question whether stakeholders (e.g., students, parents, educators, employers) would accept the results of a game for a high stakes decision. If a high stakes outcome is dependent on a game, how fun can that game really be? Most high stakes assessment decisions (e.g., educational interventions, grade promotion, graduation, job hiring, licensure to practice) are not games—people’s lives are dependent on these decisions—and it may be inappropriate to make a game out of things that should be taken seriously.

If one can accept that GBAs may be relegated to the low stakes, formative assessment world for the time being, what can their future be? This paper has argued that the power of GBAs is in their ability to capture how students change gameplay over time, which in turn may suggest that lengthier games may be the direction researchers should take. This goes against the trend of traditional assessments, which have sought to be

shorter and leave a smaller footprint in the school day, such as the use computer-adaptive tests. One could, however, imagine shifting to a GBA which is enormous and ongoing—a game environment that students might interact with for years, across grade levels. Many entertainment video games already use this strategy, creating large game universes that draw players in for a long time. Entertainment games also capitalize on the engagement of their players, releasing new expansions or modules that build off the existing game and fuel interest. One can imagine the same thing occurring with GBA in the education space. Students might interact with a system of games throughout their education, and the games might be skinned or tailored to students' individual preferences without losing the key mechanics, rules, and connections that relate to the construct of interest. As unexpected clusters of gameplay are discovered by researchers, these might inform the development of new game expansions that either encourage and develop that behavior or limit and correct it, as deemed appropriate. A multiyear, adaptable, and expandable GBA—is such a thing possible? The concept exists in science fiction novels, but the practicality and utility of such an instrument may be outside the scope of what can or should be expected from an education system; however, if the GBA systems were engaging and adaptable enough, they could complement education efforts, creating a symbiotic relationship and data exchange between the initiatives within the classrooms and the play that occurs outside.

For the time being, GBAs, like traditional assessments, will likely continue to be time-bounded, single-purpose instruments. GBA researchers should maintain a foundation of validity in their work, linking aspects of game design and gameplay through argument structures back to inferences about the constructs of interest. At the

same time, GBA researchers should be encouraged to experiment with broader ranges of task models and measurement models through GBAs, which in turn will necessitate broader collaboration with experts from a variety of fields and the use of principled development approaches like ECD to facilitate a common understanding of the instrument. Games will continue to be a presence in formal and informal education, and to realize the full potential of the data created by educational gameplay, assessment researchers need to be prepared to be as creative as the games' designers.

Appendix A: R Scripts for Full Study

```

#TITLE: An Evaluation of Clustering Algorithms for Modeling Game-Based Assessment Work
#Processes
#AUTHOR: W. Austin Fossey, EDMS PhD Candidate, University of Maryland
#DATE: July 19, 2017
#DESCRIPTION: R script for simulating game-based assessment data and evaluating true
#cluster recovery of k-means, SOM, #and ROCK clustering algorithms.

#load libraries
library("poLCA")
library("kohonen")
library("psych")
library("clusterCrit")
library("cba")

#set working directory
setwd("C:/Users/afossey/Documents/disco")

#define cody function, which recodes cluster assignments to allow for Kappa calculations
#between two lists of clusters #of the same data.
cody=function(x,y){
  tab=as.data.frame.matrix(table(x,y))
  tab=tab[do.call(order, c(tab,decreasing=T)),]
  for(i in 1:nrow(tab)){
    subtab=tab[c(i:nrow(tab)),]
    subtab=subtab[order(subtab[,i], decreasing=T),]
    tab[c(i:nrow(tab)),]=subtab
  }
  while(nrow(tab)<ncol(tab)){
    tab=rbind(tab,rep(0,ncol(tab)))
  }
  return(as.table(as.matrix(tab)))
}

set.seed(52309)      #set seed for random numbers (used to select starting centers in k-
means and SOM)
nrel=20              #number of relevant variables defining each cluster
ndat=400             #number of datasets per cell
iter=2000           #number of iterations for k-means and SOM

#simulation condtions
simk=c(4,10,20,50)  #number of clusters
simn=matrix(c(      #matrix of clusters' sizes and variance of cluster sizes
  50,9,
  100,9,
  50,100,
  100,100),
  ncol=2,byrow=T)
simp=c(.3,.6,.9)    #probability of observing a response in the 20 relevant variables

#create output data frame to record results
output=data.frame(cell=integer(),k=integer(),prob=double(),size=character(),dataset=
integer(),nvars=integer(),
  n=integer(),rmsd=numeric(),kappak=numeric(),kappas=numeric(),kappar=numeric(),perk
=numeric(),pers=numeric(),
  perr=numeric(),dbik=numeric(),dbis=numeric(),dbir=numeric(),theta=double(),strings
AsFactors=F)

#set count variables for tracking
count=1
cell=1

for(a in 1:length(simk)){
  k=simk[a]          #set number of clusters k
  nvars=nrel*k      #number of variables (columns) in data set equals number of
                    #relevant variables times k
  for(b in 1:nrow(simn)){
    npar=simn[b,]    #set parameters for generating cluster sizes

```

```

for(c in 1:length(simp)){
  resp.prob=simp[c]      #set response probability
  for(d in 1:ndat){      #simulate and analyze 400 datasets for the
                        #cell

    #add identification data to output
    output[count,1:6]=c(cell,k, resp.prob,paste(npar[1],npar[2],
    sep="-"),d,nvars)

    #create list of response probabilities for each variable
    #and each cluster
    probs=list()
    for(i in 1:nvars){
      #default to pr(0)=.95, pr(1)=.05
      probs[[i]]=matrix(rep(c(.95,.05),k), ncol=2,
      byrow=T)
    }

    #randomly select variables that will have higher response
    #probability for each cluster
    for(i in 1:k){
      relevant=sample(1:length(probs),nrel,replace=F)

      for(j in 1:length(relevant)){
        probs[relevant[j]][[1]][i,]=c(1-
        resp.prob,resp.prob)
      }
    }

    #create cluster sizes using npar values
    clust.size=round(rnorm(k,npar[1],sqrt(npar[2])),0)

    #simulate data using LCA package, recode to 0/1 binary for
    #ROCK, record true cluster #assignments

    simdat=poLCA.simdata(N=sum(clust.size),probs,P=
    clust.size/sum(clust.size))
    simdat$dat=simdat$dat-1
    tclust=simdat$trueclass
    output[count,"n"]=nrow(simdat$dat)

    #create matrix of probabilities for each item on each
    #cluster

    #these are the expected cluster centers (columns are
    #clusters, rows are variables)
    probs=do.call("rbind",lapply(probs, function(x) x[,2]))

    #calculate RMSD for the expected cluster centers to
    #represent their degree of overlap
    #i.e., higher RMSD represents less overlap in the cluster
    #centers--more distinct #response types

    output[count,"rmsd"]=sqrt(sum((as.matrix(dist(t(cbind(
    probs,rowMeans(probs))))
    [k+1,1:(k)]^2)/k)

    #####Run K-Means#####
    kmeantest=NULL
    attempt=1
    while(is.null(kmeantest) && attempt<=10){

      try({kmeantest=invisible(kmeans(simdat$dat,centers=
      k, nstart=iter)))
      attempt=attempt+1
    }
    kclust=kmeantest$cluster

    output[count,"dbik"]=intCriteria(as.matrix(simdat$dat),
    kmeantest$cluster,"Davies_Bouldin")$davies_bouldin
  }
}

```

```

#####Run SOM#####
scheck=0
while(scheck!=k){
  somtest=invisible(som(as.matrix(simdat$dat), grid =
    somgrid(k, 1, "hexagonal"),
    rlen=iter,alpha=c(.05,.01)))
  scheck=nrow(table(somtest$unit.classif))
}
sclust=somtest$unit.classif

output[count,"dbis"]=intCriteria(as.matrix(simdat$dat),
somtest$unit.classif,"Davies_Bouldin")$davies_bouldin

#####Run ROCK#####
theta=.95
while(theta>=0.05){
rocktest=invisible(rockCluster(as.matrix(simdat$dat), n=k,
theta=theta))
  #authors recommend binary distance
  if(length(rocktest$size)==k){
    break
  }
  theta=theta-.05
}
rclust=as.integer(rocktest$c1)

output[count,"dbir"]=intCriteria(as.matrix(simdat$dat),
as.integer(rocktest$c1),"Davies_Bouldin")$davies_bouldin
output[count,"theta"]=theta

#####Check True Cluster Recovery#####

#kappa values

output[count,"kappak"]=try(cohen.kappa(cody(kclust,tclust))
$kappa)
output[count,"kappas"]=try(cohen.kappa(cody(sclust,tclust))
$kappa)
output[count,"kappar"]=try(cohen.kappa(cody(rclust,tclust))
$kappa)

#percentage match values

output[count,"perk"]=try(sum(diag(cody(kclust,tclust)))/sum
(cody(kclust,tclust)))
output[count,"pers"]=try(sum(diag(cody(sclust,tclust)))/sum
(cody(sclust,tclust)))
output[count,"perr"]=try(sum(diag(cody(rclust,tclust)))/sum
(cody(rclust,tclust)))

#save dataset and confusion matrices, if desired
#write.csv(simdat$dat, paste("dat",count,"csv",sep=".")
#write.csv(probs, paste("probs",count,"csv",sep=".")
#write.csv(cody(kclust,tclust),
#paste("ktab",count,"csv",sep=".")
#write.csv(cody(sclust,tclust),
#paste("stab",count,"csv",sep=".")
#write.csv(cody(rclust,tclust),
#paste("rtab",count,"csv",sep=".")

count=count+1

}
#write cell output to CSV file and progress to next cell

write.csv(output[which(output$cell==cell),],paste("output",cell,
"csv",sep=".")
cell=cell+1
}
}

```

```
}  
#write full output to CSV file  
write.csv(output,"output.full.csv")
```

```

#TITLE: An Evaluation of Clustering Algorithms for Modeling Game-Based Assessment Work
#Processes
#AUTHOR: W. Austin Fossey, EDMS PhD Candidate, University of Maryland
#DATE: July 19, 2017
#DESCRIPTION: R script for simulating game-based assessment data and evaluating true
#cluster recovery of k-means, SOM, #and ROCK clustering algorithms.

#load libraries
library("poLCA")
library("kohonen")
library("psych")
library("clusterCrit")
library("cba")

#set working directory
setwd("C:/Users/afossey/Documents/disco2")

set.seed(52309)      #set seed for random numbers (used to select starting centers in k-
means and SOM)
nrel=20             #number of relevant variables defining each cluster
ndat=400           #number of datasets per cell
iter=1000          #number of iterations for k-means and SOM

#simulation conditons
simk=c(4,10,20,50) #number of clusters
simn=matrix(c(      #matrix of clusters' sizes and variance of cluster sizes
  50,9,
  100,9,
  50,100,
  100,100),
  ncol=2,byrow=T)
simp=c(.3,.6,.9)   #probability of observing a response in the 20 relevant variables

#create output data frame to record results
output=data.frame(cell=integer(),k=integer(),prob=double(),size=character(),dataset=
integer(),nvars=integer(),n=integer(),rmsd=numeric(),mink=integer(),dbik=numeric(),mins=i
nteger(),dbis=numeric(),minr=integer(),dbir=numeric(),stringsAsFactors=F)

#set count variables for tracking
count=1
cell=1

for(a in 1:length(simk)){
  k=simk[a]          #set number of clusters k
  nvars=nrel*k      #number of variables (columns) in data set equals number of
                    #relevant variables times k
  for(b in 1:nrow(simn)){
    npar=simn[b,]    #set parameters for generating cluster sizes
    for(c in 1:length(simp)){
      resp.prob=simp[c] #set response probability
      for(d in 1:ndat){ #simulate and analyze 400 datasets
                        #for the cell

#add identification data to output

output[count,1:6]=c(cell,k, resp.prob,paste(npar[1],npar[2],
sep="-"),d,nvars)

#create list of response probabilities for each variable
#and each cluster
probs=list()
for(i in 1:nvars){
  #default to pr(0)=.95, pr(1)=.05
  probs[[i]]=matrix(rep(c(.95,.05),k), ncol=2,
byrow=T)
}

#randomly select variables that will have higher response

```

```

#probability for each cluster
for(i in 1:k){
  relevant=sample(1:length(probs),nrel,replace=F)

  for(j in 1:length(relevant)){
    probs[relevant[j]][[1]][i,]=c(1-
      resp.prob,resp.prob)
  }
}

#create cluster sizes using npar values
clust.size=round(rnorm(k,npar[1],sqrt(npar[2])),0)

#simulate data using LCA package, recode to 0/1 binary for
#ROCK, record true cluster assignments

simdat=poLCA.simdata(N=sum(clust.size),probs,P=
  clust.size/sum(clust.size))
simdat$dat=simdat$dat-1
tclust=simdat$trueclass
output[count,"n"]=nrow(simdat$dat)

#create matrix of probabilities for each item on each
#cluster. these are the expected cluster centers (columns
#are clusters, rows are variables)
probs=do.call("rbind",lapply(probs, function(x) x[,2]))

#calculate RMSD for the expected cluster centers to
#represent their degree of overlap; i.e., higher RMSD
#represents less overlap in the cluster centers--more
#distinct response types

output[count,"rmsd"]=sqrt(sum((as.matrix(dist(t(cbind(
  probs,rowMeans(probs))))
  [k+1,1:(k)]^2)/k)

#create comp dataframe to compare DBI values for clustering
#solutions at different values of k

comp=data.frame(dbik=numeric(),dbis=numeric(),dbir=numeric
  (),stringsAsFactors=F)

for(i in -3:3){
  #run through seven different
  #possible k values

  #####Run K-Means#####
  kmeantest=NULL
  attempt=1
  while(is.null(kmeantest) && attempt<=10){
    try({kmeantest=invisible(kmeans(simdat$dat,
      centers=(k+i), nstart=iter))})
    attempt=attempt+1
  }
  kclust=kmeantest$cluster

  comp[i+4,"dbik"]=intCriteria(as.matrix(simdat$dat),
    kmeantest$cluster,"Davies_Bouldin")$davies_bouldin

  #####Run SOM#####
  somtest=invisible(som(as.matrix(simdat$dat), grid =
    somgrid((k+i), 1,"hexagonal"),rlen=iter,
    alpha=c(.05,.01)))
  sclust=somtest$unit.classif

  comp[i+4,"dbis"]=intCriteria(as.matrix(simdat$dat),
    as.integer(somtest$unit.classif),"Davies_Bouldin")
  $davies_bouldin

```

```

#####Run ROCK#####
theta=.95
while(theta>=0.05){

    rocktest=invisible(rockCluster(as.matrix(
simdat$dat), n=(k+i),theta=theta))
    #authors recommend binary distance
    if(length(rocktest$size)==(k+i)){
        break
    }
    theta=theta-.05
}
rclust=as.integer(rocktest$c1)

comp[i+4,"dbir"]=intCriteria(as.matrix(simdat$dat),
as.integer(rocktest$c1),"Davies_Bouldin")
$davies_bouldin
}

#find value of k that yielded lowest DBI for each
#algorithm, record in output
comp[comp==0]=NA

output[count,c("dbik","dbis","dbir")]=apply(comp,2,
function(x) min(x, na.rm=T))

output[count,c("mink","mins","minr")]=apply(comp,2,
which.min)-4+k

count=count+1

}
#write cell output to CSV file and progress to next cell

write.csv(output[which(output$cell==cell),],paste("output2",cell,
"csv",sep="."))
cell=cell+1
}
}
#write full output to CSV file
write.csv(output,"output2a.full.csv")

```

```

#TITLE: An Evaluation of Clustering Algorithms for Modeling Game-Based Assessment Work
#Processes
#AUTHOR: W. Austin Fossey, EDMS PhD Candidate, University of Maryland
#DATE: August 16, 2017
#DESCRIPTION: R script for analyzing work process response data in the Beanstalk math
#game using k-means, SOM, and #ROCK clustering algorithms.

#set working directory
setwd("E:/Dissertation/Dissertation/Tutorial11")

#load libraries
library("poLCA")
library("kohonen")
library("psych")
library("clusterCrit")
library("cba")
library("flexclust")

#define cody function, which recodes cluster assignments to allow for Kappa calculations
#between two lists of clusters #of the same data.
cody=function(x,y){
  tab=as.data.frame.matrix(table(x,y))
  tab=tab[do.call(order, c(tab,decreasing=T)),]
  for(i in 1:nrow(tab)){
    subtab=tab[c(i:nrow(tab)),]
    subtab=subtab[order(subtab[,i], decreasing=T),]
    tab[c(i:nrow(tab)),]=subtab
  }
  while(nrow(tab)<ncol(tab)){
    tab=rbind(tab,rep(0,ncol(tab)))
  }
  return(as.table(as.matrix(tab)))
}

#read in cleaned, recoded Beanstalk attempt data for analysis
dat=read.csv("beandat.csv",header=T,stringsAsFactors=F)

#number of iterations for k-means and SOM
iter=2000

#identify gameplay variables in dataset that will be analyzed by clustering algorithms
gameplay=c("xn4.1","xn4.2","xn4.3","xn4.4","xn3.1","xn3.2","xn3.3","xn3.4","xn2.1",
"xn2.2","xn2.3","xn2.4","xn1.1","xn1.2","xn1.3","xn1.4","x1.1","x1.2","x1.3",
"x1.4","x2.1","x2.2","x2.3","x2.4","x3.1","x3.2","x3.3","x3.4","x4.1",
"x4.2","x4.3","x4.4","h.1","h.2","h.3","h.4","monster")

#identify holdout sample
holdout=sample(unique(dat$student),round(length(unique(dat$student))/3,0),replace=F)
dat$holdout=0
dat[which(dat$student %in% holdout),"holdout"]=1

#set game levels as factor in the data frame and set the desired order (for graphing)
dat$item=as.factor(dat$item)
dat$item=factor(dat$item, levels = c("1-1","1-2","1-3","1-4","1-5","1-6","1-7",
"1-8","1-9","1-10","1-11","1-12","1-13","1-14","1-15","2-1","2-2","2-3","2-4","2-5",
"2-6","2-7","2-8","2-9","2-10","2-11","2-12","2-13","2-14","3-1","3-2","3-3","3-4","3-
5","3-6","3-7","3-8","3-9","3-10","3-11","3-12","4-1","4-2","4-3","4-4","4-5","4-6","4-
7","4-8","4-9","4-10","4-11","4-12","4-13","5-1","5-2","5-3","5-4","5-5","5-6","5-7","5-
8","5-9","5-10","5-11","5-12","5-13","5-14","5-15","6-1","6-2","6-3","6-4","6-5","6-
6","6-7","6-8","6-9","6-10","6-11","6-12","7-1","7-2","7-3","7-4","7-5","7-6","7-7","7-
8","7-9","7-10","7-11","7-12"))

#create unique list of game levels
items=unique(dat$item)

#create itemout list to store models and solutions for each game level
itemout=list()

#set seed for random numbers (used to select starting centers in k-means and SOM)
set.seed(52309)

```

```

#create output dataframe to store the results of the analysis
output=data.frame(item=character(),k=integer(),nvars=integer(),n=integer(),kappa.k=numeric(),kappa.kr=numeric(),
  kappa.sr=numeric(),dbik=numeric(),dbis=numeric(),
  dbir=numeric(),stringsAsFactors=F)
count=1

for(i in 2:length(items)){

  #select cases for the given item, and pull only the gameplay variables for
  #analysis
  subs=which(dat$item==levels(dat$item)[i]&dat$holdout==0)
  resp=dat[subs,gameplay]

  #jump to next game level if there are fewer than 25 cases
  if(nrow(resp)<25) next

  #remove variables from analysis if they are activated/not activated in five or
  #fewer cases
  resp=resp[, (colSums(resp, na.rm=T)>=5&colSums(resp,na.rm=T)<(nrow(resp)-5))]

  #if only one or two variables are left for analysis, move to the next game level
  if(class(resp)=="integer") next
  if(ncol(resp)<3) next

  #create list to store models
  klist=list()

  #run clustering algorithms for k=2-10 clusters
  for(k in 2:10){

    #record level, number of clusters, number of variables analyzed, and
    #number of cases
    output[count,1:4]=c(levels(dat$item)[i],k,ncol(resp),nrow(resp))

    #set list for storing outputs from individual models
    model=list()
    model[[1]]=k

    #if number of clusters is greater than half the sample size, move to the
    #next game level
    if(k>nrow(unique(resp))/2) break

    #####Run K-Means#####
    kmeantest=NULL
    attempt=1
    while(is.null(kmeantest) && attempt<=10){
      try({kmeantest=invisible(kmeans(resp,centers=(k), nstart=iter))})
      attempt=attempt+1
    }
    kclust=kmeantest$cluster
    model[[2]]=c(kclust)
    model[[5]]=kmeantest

    #####Run SOM#####
    somcheck=0
    while(somcheck!=k){
      somtest=invisible(som(as.matrix(resp), grid = somgrid(k, 1,
        "hexagonal"),rlen=iter,alpha=c(.05,.01)))
      sclust=somtest$unit.classif
      somcheck=length(unique(sclust))
    }
    model[[3]]=sclust
    model[[6]]=somtest

    #####Run ROCK#####
    theta=.95
    while(theta>=0.05){
      rocktest=invisible(rockCluster(as.matrix(resp), n=k, theta=theta))
      #authors recommend binary distance
      if(length(rocktest$size)==(k+i)){

```

```

        break
    }
    theta=theta-.05
}
rclust=as.integer(rocktest$cl)
model[[4]]=rclust
model[[7]]=rocktest

#transform response data to numeric class
resp[]=lapply(resp,as.numeric)

#record DBI
output[count,"dbik"]=try(intCriteria(as.matrix(resp),kmeantest$cluster,
"Davies_Bouldin")$davies_bouldin)

output[count,"dbis"]=try(intCriteria(as.matrix(resp),as.integer(
somtest$unit.classif),"Davies_Bouldin")$davies_bouldin)

output[count,"dbir"]=try(intCriteria(as.matrix(resp),as.integer(
rocktest$cl),"Davies_Bouldin")$davies_bouldin)

#record Kappa between algorithms' solutions
output[count,"kappa.ks"]=try(cohen.kappa(cody(kclust,sclust))$kappa)
output[count,"kappa.kr"]=try(cohen.kappa(cody(kclust,rclust))$kappa)
output[count,"kappa.sr"]=try(cohen.kappa(cody(sclust,rclust))$kappa)

#move to next output row and save the model lists
count=count+1
klist[[k]]=model
}

#plot and save charts of DBI values by number of clusters (k)
png(filename=paste("DBI",levels(dat$item)[i],"png",sep=".")
par(family="serif",font=1,col="black")
matplot(output[which(output$item==levels(dat$item)[i]),c("dbik","dbis","dbir")],
type = c("b"),pch=1,col ="black",xlim=c(1,9),ylim=c(0,2),xlab="#Clusters",
ylab="DBI",xaxt="n")
legend("topright", legend=c("k-means","SOM","ROCK"), lty=1:3, cex=0.8)
axis(1, at=1:9, labels=c(2:10))
dev.off()

#save list of models for all levels of k for the game level and label by level
#name
itemout[[i]]=klist
names(itemout)[i]=levels(dat$item)[i]
}

#write output to CSV file
write.csv(output, "model comparison.csv")

# At this point, the researcher analyzed the output in MS Excel to identify clustering
# solutions with
# the lowest DBI and where the solution had a Kappa value of at least 0.80 with another
# algorithm's
# solution. From this analysis, a Model Selection file was created, identifying the #
# number of clusters
# to be used at each game level and the algorithm solution that should be used for that
# level. This
# file is read in as "orders" below.

#read in model selection orders
orders=read.csv("model selection.csv",header=T, stringsAsFactors=F)

#create column in the response data to record the cluster ID associated with the work
#process
dat$clust=NA

#create blank vector to record descriptive statistics for mean endorsement rate of the
#gameplay variables, by cluster
desc=c()

```

```

#use the orders to assign clusters to all responses at each game level
for(i in 1:nrow(orders)){

  #identify the algorithm to use, the game level to which it should be applied, and
  #the number of clusters
  alg=orders[i,"alg"]
  item.index=orders[i,"item"]
  k=orders[i,"k"]

  #assign cluster IDs to the training sample for the game level
  resp.m=names(itemout[[item.index]][[k]][[2]])
  dat[resp.m,"clust"]=itemout[[item.index]][[k]][[alg+1]]

  #assign cluster IDs to the holdout sample, using the selected algorithm and the
  #training data to predict the
  #holdout's cluster IDs
  switch(alg,
    "1"={dat[which(dat$item==orders[i,"item"]&dat$holdout==1),"clust"]=
      predict(as.kcca(itemout[[item.index]][[k]][[alg+4]],
        data=dat[which(dat$item==orders[i,"item"]&dat$holdout==0),colnames(
itemout[[item.index]]
[[k]][[alg+4]]$centers))),
      newdata=dat[which(dat$item==orders[i,"item"]&dat$holdout==1),colnam
es(itemout[[item.index]]
[[k]][[alg+4]]$centers))},
    "2"={dat[which(dat$item==orders[i,"item"]&dat$holdout==1),"clust"]=
      map(itemout[[item.index]][[k]][[alg+4]],
        newdata=as.matrix(dat[which(dat$item==orders[i,"item"]&dat$holdout=
=1),
      colnames(itemout[[item.index]][[k]][[alg+4]]$data))))$unit.classif}
    ,
    "3"={dat[which(dat$item==orders[i,"item"]&dat$holdout==1),"clust"]=
      as.integer(predict(itemout[[item.index]][[k]][[alg+4]],
        x=as.matrix(dat[which(dat$item==orders[i,"item"]&dat$holdout==1),
        colnames(itemout[[item.index]][[k]][[alg+4]]$x))))$cl}
  )

  #calculate DBI for holdout cluster assignments, add to "orders" dataframe
  subs=which(dat$item==orders[i,"item"]&dat$holdout==0)
  resp=dat[subs,gameplay]
  resp=resp[, (colSums(resp, na.rm=T)>=5&colSums(resp,na.rm=T)<(nrow(resp)-5))]
  subplay=names(resp)
  orders[i,"dbih"]=intCriteria(data.matrix(sapply(dat[which(dat$item==orders[i,
"item"]&dat$holdout==1),subplay],
    FUN=as.numeric)),
    as.integer(dat[which(dat$item==orders[i,"item"]&dat$holdout==1),"clust"]),
    "Davies_Bouldin")$davies_bouldin

  #calculate descriptives for gameplay variables by item and cluster
  for(j in 1:k){
    desc=rbind(desc,c(item.index,j,nrow(dat[which(dat$item==
item.index&dat$clust==j),]),),
      round(colMeans(dat[which(dat$item==item.index&dat$clust==j),
c("CORRECT",gameplay)],2)))
  }
}

#write CSV files for the output data and the descriptive statistics for each game level
colnames(desc)[1:3]=c("item","k","n")
desc[,"item"]=paste("'",desc[,"item"],"'",sep="")
write.csv(desc, "cluster_descriptives.csv")
write.csv(orders, "orders.csv")

```

References

- Almond, R. G., Kim, Y. J., Velasquez, G., & Shute, V. J. (2014). How task features impact evidence from assessments embedded in simulations and games. *Measurement, 12*, 1-33.
- Amershi, S., & Conati, C. (2009). Combining unsupervised and supervised classification to build user models for exploratory learning environments. *Journal of Educational Data Mining, 1*(1), 1-54.
- Baker, R., Wang, E., & Andres, M. (2017a). *Knowledge inference: Bayesian knowledge tracing* [PDF Document]. Available from: <https://courses.edx.org/courses/course-v1:PennX+BDE1x+1T2017>.
- Baker, R., Wang, E., & Andres, M. (2017b). *Clustering: Validation and selection of k* [PDF Document]. Available from: <https://courses.edx.org/courses/course-v1:PennX+BDE1x+1T2017>.
- Baker, R., Wang, E., & Andres, M. (2017c). *Cross-validation and over-fitting* [PDF Document]. Available from: <https://courses.edx.org/courses/course-v1:PennX+BDE1x+1T2017>.
- Bejar, I. I., & Braun, H. I. (1994). On the synergy between assessment and instruction: Early lessons from computer-based simulations. *Machine-Mediated Learning, 4*(1), 5-25.
- Bennett, R. E. (2010). Cognitively based assessment of, for, and as learning (CBAL): A preliminary theory of action for summative and formative assessment. *Measurement: Interdisciplinary Research and Perspectives, 8*(2), 70-91.

- Berkhin, P. (2006). A survey of clustering data mining techniques. In J. Kogan, C. Nicholas, & M. Teboulle (Eds.), *Grouping multidimensional data* (pp. 25–72). New York, NY: Springer.
- BrainPOP. (2015). Number Jumble [software]. Available from <https://www.brainpop.com/games/numberjumble/>.
- Braaten, K. N. (2009). An empirical investigation of unscalable components in scaling models (Doctoral dissertation). Available from: <http://pqdtopen.proquest.com/doc/304920693.html?FMT=AI>. (UMI Number: 3359337)
- Braun, H., Bejar, I. I., & Williamson, D. M. (2006). Rule-based methods for automated scoring: application in a licensing context. In D. M. Williamson, R. J. Mislevy, & I. I. Bejar (Eds.), *Automated scoring of complex tasks in computer-based testing* (pp. 83-122). Mahwah, NJ: Erlbaum.
- Bruer, J. T. (1993). The mind's journey from novice to expert: If we know the route, we can help students negotiate their way. *American Educator: The Professional Journal of the American Federation of Teachers*, 17(2), 38-46.
- Brünger, A., Marzetta, A., Fukuda, K., Nievergelt, J. (1999). The parallel search bench ZRAM and its applications. *Annals of Operations Research*, 90, 45-63.
- Buchta, C., & Hahsler, M. (2014). cba: Clustering for business analytics [software]. R package version 0.2-14. Available from: <http://CRAN.R-project.org/package=cba>.
- Bullinaria, J. A. (2004). *Self-organizing maps: Fundamentals* (PDF Document). Available from: <http://www.cs.bham.ac.uk/~jxb/NN/116.pdf>.

Bycer, J. (2012, January 4). Subjective difficulty: how plumbers can fight demons.

Gamasutra. Available from

http://gamasutra.com/view/feature/134950/examining_subjective_difficulty_.php.

Carnegie Mellon, Educational Technology Center's ENGAGE Program (2013).

Beanstalk: A see-saw game of balance [software]. Available from

<http://beanstalk.etc.cmu.edu/>.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1), 37-46.

Corrigan, S., Stephenson, S., Makany, T., Frenz, M., Hoffman, E., Bauer, M., ... Menon, U. (2014, April). *Combining fun, learning, and measurement: Game development in the classroom and beyond*. Workshop presented at the Annual Meeting for the National Council on Measurement in Education, Philadelphia, PA.

Crocker, L., & Algina, J. (2008). *Introduction to Classical and Modern Test Theory*.

Mason, OH: Cengage Learning

D'Angelo, C., Rutstein, D., Harris, C., Bernard, R., Borokhovski, E., & Haertel, G.

(2014). *Simulations for STEM learning: Systematic review and meta-analysis (executive summary)*. Menlo Park, CA: SRI International.

Davies, D. L. & Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1(2), 224–227.

de Ayala, R. J. (2009). *The theory and practice of Item Response Theory*. New York, NY: The Guildford Press.

Desgraupes, B. (2014). clusterCrit: Clustering indices (version 1.2.4) [computer software]. <http://CRAN.R-project.org/package=clusterCrit>.

- Dimitriadou, E., Dolnicar, S., & Weingessel, A. (2002). An examination of indexes for determining the number of clusters in binary data sets. *Psychometrika*, 67(1), 137-160.
- Dunietz, J. (2016). The fundamental limits of machine learning. *Nautilus*. Available from http://nautil.us/blog/the-fundamental-limits-of-machine-learning?utm_source=RSS_Feed&utm_medium=RSS&utm_campaign=RSS_Syndication
- Eseryel, D., Ge, X., Ifenthaler, D., & Law, V. (2011). Dynamic modeling as a cognitive regulation scaffold for developing complex problem-solving skills in an educational massively multiplayer online game environment. *Journal of Educational Computing Research*, 45(3), 265-286.
- Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, 17(3), 37-54.
- Fishman, B., Riconscente, M., Snider, R., Tsai, T., & Plass, J. (2014). Empowering educators: Supporting student progress in the classroom with digital games. Ann Arbor: University of Michigan.
- Fraley, C., & Raftery, A. E. (2007). Model-based methods of classification: Using the mclust software in chemometrics. *Journal of Statistical Software*, 18(6), 1-13.
- Gee, J. P. (2007). *What video games have to teach us about learning and literacy* (2nd ed.). New York: Palgrave.
- Gentile, D. A., Anderson, C. A., Yukawa, S., Ihori, N., Saleem, M., Ming, L. K., Shibuya, A., Liau, A. K., Khoo, A., Bushman, B. J., Huesmann, L. R., & Sakamoto, A. (2009). The effects of prosocial video games on prosocial

- behaviors: International evidence from correlational, longitudinal, and experimental studies. *Personality and Social Psychology Bulletin*, 35, 752–763.
- Goodman, L. A. (1975). A new model for scaling response patterns: An application of the quasi-independence concept. *Journal of the American Statistical Association*, 70, 755-768.
- Granic, I., Lobel, A., & Engels, C. M. E. (2014). The benefits of playing video games. *American Psychologist*, 69(1), 66-78.
- Guha, S., Rastogi, R., and Shim, K. (1999). ROCK: A robust clustering algorithm for categorical attributes. In Proceedings of the 15th ICDE, 512-521, Sydney, Australia.
- Haladyna, T. M., & Downing, S. M. (1989). A taxonomy of multiple-choice item-writing rules. *Applied Measurement in Education*, 2, 51-78.
- Halkidi, M., Batistakis, Y., & Vazirgiannis, M. (2001). On clustering validation techniques. *Journal of Intelligent Information Systems*, 17, 107-145.
- Hall, L. O, Bensaid, A. M., Clarke, L. P., Velthuizen, R. P., Silbiger, M. S., & Bezdek, J. C. (1992). A comparison of neural network and fuzzy clustering techniques in segmenting magnetic resonance images of the brain. *IEEE Transactions on Neural Networks*, 3(5), 672-682.
- Hamerly, G., & Elkan, C. (2004). Learning the k in k -means. *Advances in Neural Information Processing Systems*, 16.
- Hand, D., Mannila, H., & Smyth, P. (2001). *Principles of data mining*. Cambridge, MA: MIT Press.

- Hardin, L. E. (2003). Problem solving concepts and theories. *Journal of Veterinary Medical Evaluation*, 3(30), 226-229.
- Hartigan, J. A. & Wong, M. A. (1979). A K-means clustering algorithm. *Applied Statistics*, 28, 100-108.
- Heitin, L. (2014, April 24). NAEP uses video-game design for new technology and engineering test. Available from http://blogs.edweek.org/edweek/curriculum/2014/04/naep_uses_video-game_design_fo.html.
- Hennig, C. (2015). fpc: Flexible Procedures for Clustering (version 2.1-10) [computer software]. <https://CRAN.R-project.org/package=fpc>.
- Hennig, L., & Liao, T. F. (2011). How to find an appropriate clustering for mixed type variables with application to socioeconomic stratification. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 62(3), 309-514.
- Hoffman, E., John, M., & Makany, T. (2014, April). How do game design frameworks align with learning and assessment design frameworks? In A. Oranje (Chair), *Integrating games, learning, and assessment*. Symposium conducted at the Annual Meeting for the National Council on Measurement in Education, Philadelphia, PA.
- Hothorn, T., & Everitt, B. S. (2009). *A handbook of statistical analysis using R*. (2nd ed.). Boca Raton, FL: Chapman & Hall.
- Houle, M. E., Kriegel, H., Kröger, P., Schubert, E., & Zimek, A. (2010). Can shared-neighbor distances defeat the curse of dimensionality? In M. Gertz, & B. Ludäscher (Eds.), *Scientific and statistical database management: 22nd*

international conference, SSDBM 2010, Heidelberg, Germany, June 30-July 2, 2010 (pp. 482-500). Berlin: Springer.

Hynar, M., Burda, M., & Šarmanová, J. (2005). Unsupervised clustering with growing self-organizing neural network: A comparison with non-neural approach. In K. Richta, V. Snášel, & J. Pokorný (Eds.), *Dateso* (pp. 55-68). Available from <http://www.cs.vsb.cz/dateso/sbornik/dateso05.pdf#page=66>.

Ishioka, T. (2005). An expansion of x-means for automatically determining the optimal number of clusters. Proceedings from *The Fourth IASTED International Conference: Computational Intelligence (CI 2005), Calgary Canada, July 4-July 6, 2005* (pp. 91-96). Available from <http://www.rd.dnc.ac.jp/~tunenori/doc/487-053.pdf>.

Johnson, S. (2001). *Emergence: The connected lives of ants, brains, cities, and software*. New York, NY: Scribner.

K20 Center, University of Oklahoma. (2011, October 27). McLarin's Adventures – mapping scenario [video]. *McLarin's Adventures*. Available from <http://dgb1.ou.edu/mclarin/>.

Kato, P.M., Cole, S.W., Bradlyn, A.S., & Pollock, B.H. (2008). A video game improves behavioral outcomes in adolescents and young adults with cancer: A randomized trial. *Pediatrics*, *122*, 305-317.

Kerr, D., & Chung, G. (2012). Identifying key features of student performance in educational video games and simulations through cluster analysis. *Journal of Educational Data Mining*, *4*(1), 145-182.

- Kerr, D., Chung, G. K. W. K., & Iseli, M. R. (2011). *The feasibility of using cluster analysis to examine log data from educational video games*. (CRESST Report 790). Los Angeles, CA: University of California, National Center for Research on Evaluation, Standards, and Student Testing (CRESST).
- Koedinger, K. R., Baker, R. S. J.d., Cunningham, K., Skogsholm, A., Leber, B., Stamper, J. (2010) A data repository for the EDM community: The PSLC DataShop. In Romero, C., Ventura, S., Pechenizkiy, M., Baker, R. S. J.d. (Eds.) *Handbook of educational data mining* (pp. 43-56). Boca Raton, FL: CRC Press.
- Kohonen, T. (1990). The Self-Organizing Map. *Proceedings of the IEEE*, 78(9), 1464-1480.
- Kriegel, H. P., Kröger, P., & Zimek, A. (2009). Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data*, 3(1), 1-59.
- Landis, J. R., & Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, 33(1), 159-174.
- Lattin, J. M., Carroll, J. D., & Green, P. E. (2003). *Analyzing multivariate data*. Pacific Grove, CA: Brooks/Cole—Thomson Learning.
- Levy, R., & Mislavy, R. J. (2004). Specifying and refining a measurement model for a computer-based interactive assessment. *International Journal of Testing*, 4(4), 333-369.
- Linzer, D. A., & Lewis, J. B. (2011). poLCA: An R package for polytomous variable latent class analysis. *Journal of Statistical Software*, 41(10), 1-29.

- Maisuria, L. K., Ong, C. S., & Lai, W. K. (1999). A comparison of artificial neural networks and cluster analysis for typing biometrics authentication. *International Joint Conference on Neural Networks*, 5, 3295-3299.
- Mandler, G. & Sarason, S. B. (1952). A study of anxiety and learning. *Journal of Abnormal and Social Psychology*, 47, 166-173.
- Margolis, M. J., & Clauser, B. E. (2006). A regression-based procedure for automated scoring of a complex medical performance assessment. In D. M. Williamson, R. J. Mislevy, & I. I. Bejar (Eds.), *Automated scoring of complex tasks in computer-based testing* (pp. 123-167). Mahwah, NJ: Erlbaum.
- Milligan, G. W., & Cooper, M. C. (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2), 159-179.
- Mislevy, R. J. (1996). Test theory reconceived. *Journal of Educational Measurement*, 33(4), 379-416.
- Mislevy, R. J., Behrens, J. T., Dicerbo, K. E., & Levy, R. (2012). Design and discovery in educational assessment: Evidence-Centered Design, psychometrics, and Educational Data Mining. *Journal of Educational Data Mining*, 4(1), 11-48.
- Mislevy, R. J., Oranje, A., Bauer, M. I., von Davier, A., Hao, J., Corrigan, S., Hoffman, E., DiCerbo, K., John, M. (2014). *Psychometric considerations in game-based assessment*. Redwood City, CA: GlassLab Games.
- Mislevy, R. J., Steinberg, L. S., & Almond, R. G. (2003). On the structure of educational assessments. *Measurement: Interdisciplinary Research and Perspectives*, 1, 3-67.

- Mislevy, R.J. (2011). *Evidence-centered design for simulation-based assessment*. (CRESST Report 800). Los Angeles, CA: University of California, National Center for Research on Evaluation, Standards, and Student Testing (CRESST).
- Muncy, J. (2015, December 3). Open-world games are changing the way we play. Available from <https://www.wired.com/2015/12/open-world-games-2015/>.
- Nash, P., & Shaffer, D. W. (2011). Mentor modeling: the internalization of modeled professional thinking in an epistemic game. *Journal of Computer Assisted Learning, 27*, 173-189.
- National Council of Architectural Registration Boards. (2012). *4RE exam guide*. Available from http://www.ncarb.org/~media/Files/PDF/ARE-Exam-Guides/SPD_Exam_Guide.pdf.
- Newell, A., & Simon, H. A. (1972) *Human problem solving*. Englewood Cliffs, NJ: Prentice Hall.
- Newell, M. A., Cook, D., Hofmann, H., & Jannink, J. (2013). An algorithm for deciding the number of clusters and validation using simulated data with application to exploring crop population structure. *The Annals of Applied Statistics, 7*(4), 1898-1916.
- Nitko, A. J., & Brookhart, S. (2007). *Educational assessment of students* (5th ed.). Upper Saddle River, NJ: Pearson Education, Inc.
- Pelleg, D., & Moore, A. (2000). X-means: Extending *k*-means with efficient estimation of the number of clusters. Proceedings from the 17th *International Conference on Machine Learning, 2000* (pp. 727-734). Available from <https://www.cs.cmu.edu/~dpelleg/download/xmeans.pdf>.

- Perron, B., & Wolf, M. J. P. (Eds.). (2009). *The video game theory reader 2*. New York, NY: Routledge.
- Powers, D. E. (1999). *Test anxiety and test performance: comparing paper-based and computer adaptive versions of the GRE general test*. (ETS Report RR-99-15). Princeton, NJ: Educational Testing Service.
- Prensky, M. (2012). *From digital natives to digital wisdom: Hopeful essays for 21st century learning*. Thousand Oaks, CA: Corwin Press.
- Quellmalz, E. S., Timms, M. J., Silbergliitt, M. D., & Buckley, B. C. (2012). Science assessments for all: integrating science simulations into balanced state science assessment systems. *Journal of Research in Science Teaching*, 49(3), 363-393.
- R Core Team (2014). R: A language and environment for statistical computing (version 3.1.1) [computer software]. Vienna, Austria: R Foundation for Statistical Computing. Available from <http://www.R-project.org/>.
- Revelle, W. (2015) psych: Procedures for personality and psychological research (version 1.5.4) [computer software]. Evanston, IL: Northwestern University. Available from <http://CRAN.R-project.org/package=psych>.
- Robinson, D. (2017). K-means clustering is not a free lunch. Available from <http://varianceexplained.org/r/kmeans-free-lunch/>.
- Romero, C., Gonzalez, P., Ventura, S., del Jesus, M. J., & Herrera, F. (2009). Evolutionary algorithms for subgroup discovery in e-learning: A practical application using Moodle data. *Expert Systems With Applications*, 39, 1632–1644.
- Salen, K., & Zimmerman, E. (2004). *Rules of play: Game design fundamentals*. Cambridge, MA: MIT.

- Salminen, M., & Ravaja, N. (2008). Increased oscillatory theta activation evoked by violent digital game events. *Neuroscience Letters*, 435, 69–72.
- Schleiner, A. (2001). Does Lara Croft wear fake polygons? Gender and gender-role subversion in computer adventure games. *Leonardo*, 34(3), 221–226.
- Schmeiser, C. B., & Welch, C. J. (2006). Test development. In R. L. Brennan (Ed.), *Educational Measurement: Fourth Edition* (pp. 307-353). Westport, CT: Praeger Publishers.
- Schwartz, K. (2014a) Profile: Illinois. In T. Barseghian & K. Sung (Eds.), *Mind shift: Guide to digital games + learning* (pp. 30-31). Available from <http://www.kqed.org/assets/pdf/news/MindShift-GuidetoDigitalGamesandLearning.pdf>.
- Schwartz, K. (2014b) Profile: New York. In T. Barseghian & K. Sung (Eds.), *Mind shift: Guide to digital games + learning* (pp. 32). Available from <http://www.kqed.org/assets/pdf/news/MindShift-GuidetoDigitalGamesandLearning.pdf>.
- Shapiro, J. (2014a, December 18). Study shows video games' impact in face-to-face teaching. Available from <http://www.forbes.com/sites/jordanshapiro/2014/12/18/study-shows-video-games-impact-on-face-to-face-teaching/2/>.
- Shapiro, J. (2014b, May 30). Can games make high-stakes tests obsolete? Available from <http://blogs.kqed.org/mindshift/2014/05/can-games-make-high-stakes-tests-obsolete/>.

- Shapiro, J., Teknibaş, K. S., Schwartz, K., & Darvasi, P. (2014) What the research says about gaming and screen time. In T. Barseghian & K. Sung (Eds.), *Mind shift: Guide to digital games + learning* (pp. 6-13). Available from <http://www.kqed.org/assets/pdf/news/MindShift-GuidetoDigitalGamesandLearning.pdf>.
- Shute, V. J. & Wang, L. (2016). Assessing and supporting hard-to-measure constructs. To appear in A. Rupp, & J. Leighton (Eds.), *Handbook of cognition and assessment* (pp. 535-562). New York, NY: Springer.
- Shute, V. J. (2011). Stealth assessment in computer-based games to support learning. In S. Tobias, & J. D. Fletcher (Eds.), *Computer games and instruction* (pp. 503-524). Charlotte, NC: Information Age Publishers.
- Shute, V. J. (2014). *Stealth assessment in games*. Proceedings from the Maryland Assessment Research Center Conference. Available from <http://marces.org/conference/InnovativeAssessment/2Shute.pdf>.
- Shute, V. J., & Ke, F. (2012). Games, learning, and assessment. In D. Ifenthaler, D. Eseryel, & Ge, X. (Eds.), *Assessment in game-based learning: Foundations, innovations, and perspectives* (pp. 43-58). New York, NY: Springer.
- Shute, V. J., Ventura, M., & Kim, Y. J. (2013). Assessment and learning of qualitative physics in Newton's Playground. *The Journal of Education Research*, 106, 423-430.
- Shute, V. J., Ventura, M., Kim, Y. J. & Wang, L. (2014). Video games and learning. In W. G. Tierney, Z. Corwin, T. Fullerton, and G. Ragusa (Eds.), *Postsecondary*

play: The role of games and social media in higher education (pp. 217-235).

Baltimore, MD: John Hopkins University Press.

Siegler, R. S. (1976). Three aspects of cognitive development. *Cognitive Psychology*, 8, 481-520.

Siegler R. S., & Chen, Z. (2002). Development of rules and strategies: Balancing the old and the new. *Journal of Experimental Child Psychology*, 81, 446-457.

Silverman, S. (2011, November 29). Target's cashier game – is it really a game?

Available from <http://www.levelspro.com/targets-cashier-game-is-it-really-a-game/>.

Song, S. & Li, C. (2006). *Improved ROCK for text clustering using asymmetric proximity*. Proceedings from the 32nd Conference on Current Trends in Theory and Practice of Computer Science. Available from

https://www.researchgate.net/publication/221513060_Improved_ROCK_for_Text_Clustering_Using_Asymmetric_Proximity.

SRI International. (2015). *Independent research and evaluation on GlassLab games and assessments*. Available from <http://www.sri.com/work/projects/glasslab-research>.

Stevens, R. H., & Casillas, A. (2006). Artificial neural networks. In D. M. Williamson, R. J. Mislevy, & I. I. Bejar (Eds.), *Automated scoring of complex tasks in computer-based testing* (pp. 259-312). Mahwah, NJ: Erlbaum.

Stevens, R., Galloway, T., Wang, P., Berka, C., Tan, V., Wohlgemuth, T., Lamb, J., & Buckles, R. (2012). Modeling the neurodynamic complexity of submarine navigation teams. *Computational and Mathematical Organization Theory*, 19, 346-369.

- Suits, B. (2005). *The grasshopper: Games, life, and utopia* (2nd ed.). Toronto, ON: Broadview.
- Takeuchi, L. M., & Vaala, S. (2014). *Level up learning: A national survey on teaching with digital games*. New York: The Joan Ganz Cooney Center at Sesame Workshop.
- Teknibaş, K. S. (2014) Getting in the game. In T. Barseghian & K. Sung (Eds.), *Mind shift: Guide to digital games + learning* (pp. 4-5). Available from <http://www.kqed.org/assets/pdf/news/MindShift-GuidetoDigitalGamesandLearning.pdf>.
- Wang, Y., Yang, C., Mathee, K., & Narasimhan, G. (2005). Clustering using adaptive self-organizing maps (ASOM) and applications. In *Computational Science–ICCS 2005* (pp. 944-951). Berlin: Springer.
- Wehrens, R., & Buydens, L.M.C. (2007). Self- and super-organising maps in R: the Kohonen package. *Journal of Statistical Software*, 21(5), 1-19.
- Xu, B., Recker, M., Qi, X., Flann, N., & Ye, L. (2013). Clustering educational digital library usage data: A comparison of latent class analysis and k-means algorithms. *Journal of Educational Data Mining*, 5(2), 38-68.