



Western Michigan University
ScholarWorks at WMU

Dissertations

Graduate College

12-2017

High Performance Computing Techniques for Analyzing Risky Decision Making

Vinay B. Gavirangaswamy
Western Michigan University, vinay.babu.g@hotmail.com

Follow this and additional works at: <https://scholarworks.wmich.edu/dissertations>



Part of the Computer Sciences Commons

Recommended Citation

Gavirangaswamy, Vinay B., "High Performance Computing Techniques for Analyzing Risky Decision Making" (2017). *Dissertations*. 3196.

<https://scholarworks.wmich.edu/dissertations/3196>

This Dissertation-Open Access is brought to you for free and open access by the Graduate College at ScholarWorks at WMU. It has been accepted for inclusion in Dissertations by an authorized administrator of ScholarWorks at WMU. For more information, please contact wmu-scholarworks@wmich.edu.



HIGH PERFORMANCE COMPUTING TECHNIQUES FOR
ANALYZING RISKY DECISION MAKING

by

Vinay B. Gavirangaswamy

A dissertation submitted to the Graduate College
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
Computer Science
Western Michigan University
December 2017

Doctoral Committee:

Ajay Gupta, Ph.D., Chair
Cynthia J. Pietras, Ph.D.
Li Yang, Ph.D.

HIGH PERFORMANCE COMPUTING TECHNIQUES FOR ANALYZING RISKY DECISION MAKING

Vinay B. Gavirangaswamy, Ph.D.

Western Michigan University, 2017

The process or activity of making choices when subject to gain or loss can be understood as risky decision making (RDM). Risky Decisions consists of outcomes of decisions that may probabilistically result in unfavorable results. Every organism that lives faces this challenge and recent research suggests that there is a computational process involved in making these decisions. This has led to new approaches in the study of RDM. My dissertation is towards contributing to expand on the existing knowledge of RDM processes.

The core contribution of my work is an analysis and development of high performance computing techniques that improves contemporary research by providing new tools. An open source toolkit called RDMTk was developed as a part of this work that is available for use in the form of Software as a Service. RDMTk uses cloud based computing resources for running analysis code on demand by a researcher.

Computational challenges are addressed for a reinforcement learning algorithm currently used in identifying individual differences in RDM [71]. A central objective was an exhaustive analysis for improving this technology using shared and distributed memory. Algorithms for MPI (Message Passing Interface) based distributed memory and CUDA-GPU (Compute Unified Device Architecture-Graphic Processing Unit) based shared memory are developed and tested using extensive experiments. Our implementation on distributed architecture was able to achieve

almost a linear speedup (e.g. 44.79x using 48 MPI threads). And showed a 130x speedup for CPU-GPU based shared memory implementation over CPU only. We also discuss a novel Floor Tiles Planning theoretical approach to further reduce the computational overhead in RDM algorithms. This approach exploits spatial & temporal dependencies in computing resource allocation along with associated data dependencies. Data for our RDM research is collected through open source RDMTk toolkit, developed as a part of the dissertation work.

ACKNOWLEDGMENTS

I would like to thank my sister, Monica Gavirangaswamy; without her support I probably would not have finished my PhD. I am thankful to my adviser Dr. Ajay Gupta for bearing with my mood swings and giving support from grass root level into research. He introduced me into the research community via letting me work for IEEE-TCPP. I got to learn many things from him not only academically. He has given me advice on personal matters when asked. I thank Aakash Gupta, Wharton School of the University of Pennsylvania, for help in conducting the experiments in the Stanford Behavioral Lab that provided the datasets for our work. I am also grateful towards Lee Newman for his work on clustering and generous sharing of the original algorithms and datasets.

Vinay B. Gavirangaswamy

© 2017 Vinay B. Gavirangaswamy

TABLE OF CONTENTS

ACKNOWLEDGMENTS	ii
LIST OF TABLES	vii
LIST OF FIGURES	ix
CHAPTER	
I. INTRODUCTION	1
1. Introduction.....	1
2. Significance of the Research.....	3
3. Research Objectives.....	4
4. Methodology.....	5
5. Main Results	5
II. BACKGROUND	7
1. Introduction.....	7
2. Tools and Techniques.....	7
3. High Performance Computing for RDM Models	13
4. Empirical Analysis of RDM Datasets	18
4.1 Individual Differences vs. Group Behavior	19
4.2 Preferences in RDM.....	20
4.3 Predictive Modeling of RDM	21
III. RDMTk: A TOOLKIT FOR RISKY DECISION MAKING.....	23
1. Introduction.....	23

Table of Contents—Continued

CHAPTER

2. RDMTk- An Overview	26
3. Technical Details	29
4. Integrating New Tasks and Games into the RDMTk Framework	32
5. Risky Decision-Making Tasks	37
6. Experiments	40
7. Collecting Data	41
8. Experiment-monitor	42
9. Data Analysis	43
10. Bootstrapping RDMTk and Amazon Elastic Compute Cloud	44
11. Contributing to RDMTk Development	46
12. Summary	46
IV. REINFORCED LEARNING MODEL USED IN ANALYZING RISKY DECISION MAKING	48
1. Introduction	48
2. Ensemble Clustering as Reinforcement Learning Algorithm	50
3. Model Formulation	56
4. Summary	60
V. DISTRIBUTED MEMORY PARALLELIZATION FOR THE REINFORCED LEARNING ALGORITHM TO FIND INDIVIDUAL DIFFERENCES	61
1. Introduction	61
2. Experimental Results	66

Table of Contents—Continued

CHAPTER

3. Summary	70
VI. SHARED MEMORY PARALLELIZATION FOR THE REINFORCED LEARNING ALGORITHM TO FIND INDIVIDUAL DIFFERENCES	72
1. Introduction.....	72
2. CUDA, Libraries and Related Tools	73
3. Implementation – An Overview.....	75
4. Experimental Results	77
5. Summary	85
VII. DISCRETIZATION AND SCHEDULING MODEL EXECUTION OF RDM RL ALGORITHM STEPS AS FLOOR TILES PLANNING.....	87
1. Introduction.....	87
2. FTiP Problem Statement for Reinforced Learning Algorithm	89
3. Solution Approaches	95
4. Summary	96
VIII. CLASSIFICATION AND PREDICTION OF PREFERENCES SHIFTS IN CUPS TASK.....	98
1. Introduction.....	98
2. Models in RDM	98
3. CUPS Task	100
4. Research Objective	102
5. Experimental Setup.....	102

Table of Contents—Continued

CHAPTER

6. Model Specifications	103
7. Experimental Results	111
8. Summary	118
IX. GENERAL DISCUSSION, CONCLUSIONS AND FUTURE WORK	120
REFERENCES	124
APPENDIX.....	136

LIST OF TABLES

1. Some available toolkits used in RDM and their constituent tools.....	8
2. Toolkits and their features comparison.....	9
3. Iowa Gambling Tasks payout scale included in RDMTk toolkit.....	37
4. CUPS tasks payout scale included in RDMTk toolkit.....	38
5. Base clustering algorithms in the ensemble and their time complexities.....	54
6. PST structure to configure ensemble clustering	55
7. Ensemble clustering model variation parameters.....	56
8. Notations, used throughout the chapter	62
9. Time complexity for different parallelization approaches	63
10. Ensemble clustering configuration parameters used for distributed memory parallelization experiments.....	67
11. Run times, speedup and efficiency of ensemble clustering after parallelization for 1000 participants' data on penguin cluster.....	68
12. Computer specification used for evaluation.....	79
13. Ensemble clustering configuration parameters used for shared memory parallelization experiments.....	79
14. Run times in seconds for executing ensemble clustering algorithm on CPU & CPU-GPU.....	82
15. Dependency list and link strength.....	91
16. Models execution sequence as floorplan	95
17. Properties of ACF and PACF to determine ARIMA model	108

List of Tables—Continued

18. Measure of average number of risky decisions across gain, loss domains and overall. advantageous and disadvantageous risky decisions are also listed.....	112
19. Validation metrics of predicted preference shifts using supervised model	118
20. Validation metrics of predicted preference shifts using unsupervised model	118

LIST OF FIGURES

1. Different aspects of my dissertation where computer science is incorporating or building upon previous contributions from data science, economic, mathematics, machine learning, statistics, and psychology.	2
2. Word cloud of article titles that refer to original ensemble clustering algorithm publication	14
3. Overall design and architecture for RDMTk	27
4. RDMTk's four-layered architecture	30
5. Step involved in integrating a new task into RDMTk	32
6. RDMTk step 1 - form to add a new task to the toolkit	33
7. RDMTk step 2 - form to add a new task to the toolkit	34
8. RDMTk step 3 - form to add new task to the toolkit	35
9. RDMTk experiments page	40
10. RDMTk create experiment screen	41
11. RDMTk form to monitor real-time experimental status and progress tracker.....	43
12. RDMTk's data analysis module on amazon AWS EC2 infrastructure	44
13. RDMTk and amazon EC2 bootstrapping.....	44
14. Summary statistics for an RDM experiment.....	45
15. Conceptual overview of ensemble clustering solution.	49
16. Execution times and speedup for parallelized RDM reinforcement learning algorithm using ensemble clustering for 1000 participants on a penguin computing POD cluster.	66
17. Call graph of agglomerative, kmeans, kmedians and spectral clustering with relative percentage computation time for each step.....	69

List of Figures—Continued

18. GPU memory hierarchy and CPU-GPU heterogeneous programming model.	73
19. Example: CUDA thread assignment with jagged indices calculation.	80
20. Speedup achieved for running ensemble clustering on CPU vs. CPU-GPU	82
21. Comparison of running distance matrix kernels using texture vs. non-texture memory.	82
22. Comparison of running (1) agglomerative, (2) kmeans, (3) spectral, and (4) kmedians clustering kernels using texture vs. non-texture memory.....	82
23. Normalized criteria value for the number of supported clusters for nine validity criteria.	83
24. Validity criteria summary showing consensus for 5 or 8 clusters.....	83
25. Run times for unordered vs. batch reconfigured model sequence execution.	83
26. GPGPU threads and resources as floor tiles	89
27. Summary of CUPS task trial type and expected values in each.	100
28. Model to predict risk taking preference shifts	104
29. ACF and PACF for data without differencing.	109
30. ACF and PACF for data with differencing.	109
31. Means of risky choices in (a) the loss and (b) the gain domain, as a function of subject group and expected-value (EV) level (risk advantageous trials; risk equal expected value trials; risk disadvantageous trials). Subjects received 15 gain trials and 15 loss trials for each of the three EV levels.....	112
32. Categorization of subjects of prospects theory experiment into twelve risky preferences types as suggested in the literature.	113
33. Typical data sample for subject's performance on CUPS task, revealing seasonality, and trend.	114

List of Figures—Continued

34. Seasonality patterns for four random subjects selected from survey pool of 324 subjects from prospects theory experiment. All subjects showed a definite sign of seasonal pattern across 90 trials as shown in the figure.	115
35. Reinforced learning model output showing valid number of clustering result for subject’s performance in prospects theory experiment.	116
36. Example for predicted reference shifts in subjects’ performance using supervised model.	117
37. Example for predicted preference shifts in subjects’ performance using unsupervised model.	117
38. RDMTk application bird’s eye view.....	121
39. Typical process for analyzing RDM from experiments	136
40. RDMTk login screen.....	138
41. RDMTk account registration page.....	139
42. RDMTk researcher’s dashboard view	140
43. RDMTk participant’s dashboard view.....	141
44. RDMTk menu item “experiments”	143
45. RDMTk menu item: experiments->add experiment.....	144
46. RDMTk create experiment form.....	144
47. RDMTk experiments list view.....	146
48. RDMTk “So” button to view experiment details.....	147
49. RDMTk form listing experiment details.....	147
50. RDMTk “Ed” button to edit existing experiment details.....	148
51. RDMTk edit experiment details form.....	149

List of Figures—Continued

52. RDMTk “Dlt” button to delete an experiment.....	150
53. RDMTk menu item for “experimental design”	150
54. RDMTk existing experimental design list and link to add new relations	151
55. RDMTk create experimental relationship form.....	151
56. Amazon mTurk interface to setup RDMTk experiments.....	152
57. RDMTk download experiment results menu item.....	153
58. RDMTk form to download experiment’s data.....	154
59. RDMTk step 1 form to add new task to toolkit	155
60. RDMTk step 2 form to add new task to toolkit	155
61. RDMTk step 3 form to add new task to toolkit	156
62. RDMTk step 1 form to monitor real time experimental status and progress tracker	159
63. RDMTk menu item to access IGT analysis models.....	161
64. RDMTk base model results sample	161
65. RDMTk random model results sample	162
66. RDMTk EVL model results sample.....	162

CHAPTER I

INTRODUCTION

1. Introduction

Regardless as to whether individuals are making personal or professional choices, every choice involves a decision to assess the advantages and disadvantages of a given choice, and every choice involves, by necessity, a negation of the other possibility. Decision making then assumes an onerous burden, particularly when some choices carry with them not only the possibility of a greater benefit, but also the possibility of a substantial risk, aside from the factor of negation and the loss of the other potential opportunity cost. Yet, individuals are involved in decision making at all phases in their lives: to attend a certain university; to pursue a particular career path; to marry a certain individual; to divorce, among many others. The results of decision making – the anticipation or expectation of a gain or a reward that prompts an individual to engage in “risky” behavior – is a factor that is not well understood. The ability to study risky decision making (RDM) and to potentially predict or forecast the evolution of such behavior presents researchers with fertile opportunities for exploration – the objective of this research.

Traditionally research in RDM was spearheaded by people with a background in psychology; however recent developments indicate increased interest by other domains including economics and neuroscience. Computer Science is a recent addition among other fields that can contribute towards developing richer understanding of RDM through intelligent and scalable algorithms to detect and predict RDM patterns. Various new fields of study have stemmed out

recently in response to latest developments in RDM such as, Neuroeconomics, Neurodevelopment, and Emotion Regulation (ER) etc. Neuroeconomics researchers have formalized a process of economic decision making with respect to activity center in the brain and related models/axioms thereof. In neurodevelopment people study about RDM across different ages starting from early childhood to old age. These new studies consider the existence of computational model for decision making, which involves modeling interaction among neurons. Neuropsychology tries to study structure and process in brain corresponding to decision making and behavior. Computational modeling of RDM is an extension of neuropsychology where people have tried to formulate decision making in terms of neural activity and brain structures for corresponding stimuli. All extensions of RDM research use and benefit from a variety of tools and techniques.

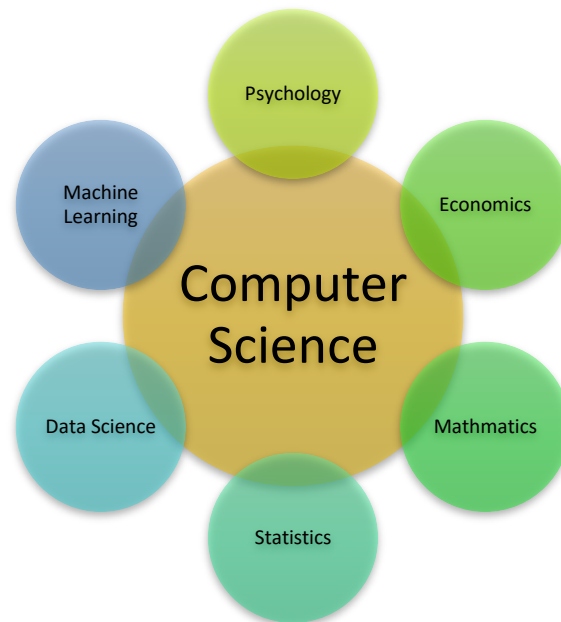


Figure 1. Different aspects of my dissertation where computer science is incorporating or building upon previous contributions from data science, economic, mathematics, machine learning, statistics, and psychology.

Contemporary researchers are trying to better understand individual preferences of people while making a decision under risky situations. Much of the theoretical work in understanding risky decision making comes from economics and psychological literature. Researchers have developed theories using statistical and mathematical structures to account for risky decision making and its various aspects. Computer science can offer to further extend RDM understanding by incorporating machine learning on top of the existing knowledge base. Theories for decision under uncertainty have been broadly classified as descriptive, prescriptive and normative. Descriptive theories try to explain how and why a pattern of decisions were taken. Prescriptive theories try to suggest what course of decisions one should make. And normative theories try to come up with the most profitable decision making patterns. However, currently literature lacks a good understanding of individual's preferences and changes in preferences while making these decisions.

My dissertation is contributing towards the existing literature for risky decision making by providing an enhanced toolkit and techniques for analyzing behavioral data from a computational perspective.

2. Significance of the Research

This research is an effort to better understand different preferences in people's decision making. There are two aspects towards this, one is to collect relevant data that would enable us with the second part, to develop models for the data to verify hypotheses under investigation. Models used to analyze data are computationally intensive and we address some of the problems that would arise from running these on large datasets.

Recent developments in computer science offers a significant improvement to existing technology and methodology by which these studies are conducted by researchers studying RDM. Games/tasks developed requires software installation and sometimes development of additional features. Also, a typical study involves collecting data from different sources/sensors. Most of the toolkits currently available are commercial and require license to use; this limits collaboration among researchers and hinders further improvements of task/game used. Our proposed Risky Decision-Making Toolkit is designed and built to the state-of-the-art as open-source toolkit. We used web and cloud technologies to make the toolkit available as “Software as a Service” (SaaS), thus making global research studies, data collection and analysis easier.

Developing models that predict patterns in RDM data will benefit researchers across various disciplines. These models could be based on analytical, machine learning or both algorithms. Researchers can derive more consistent and productive results if they have access to a wider range of datasets. Establishing relationship between analytical or machine learning models with psychological theories is the central challenge.

3. Research Objectives

Computer scientists have a plethora of innovations to contribute towards researching RDM. My dissertation work is an effort to answer some of the open challenges faced by researchers in RDM. Below we list our main contributions:

- a) Designing and building a state-of-the-art open-source toolkit to make it easier to collect the data. Exploit web and cloud technologies to make the toolkit as a SaaS - this would allow scalability, availability, collaboration, reachability, diversity, and larger-data-collection (making even global data collection easier). This way envisioned Risky Decision-Making

Toolkit will provide the necessary capability to researchers who study RDM computationally. It also includes tools to analyze data collected through experiments. Our approach thus enables a researcher to identify an individual, as well group decision making approaches.

- b) Improved computational performance of the existing popular RDM reinforcement algorithm through HPC techniques.

4. Methodology

A. Toolkit

Developing a toolkit from scratch is a challenging task. Principles of software engineering including modern practices such as agile scrum in conjunction with iterative development process and Feature Driven Development are used.

B. High Performance Computing Versions of Algorithms

Enabling RDM algorithms to use, a high-performance computing environment was done using test driven software development process.

5. Main Results

The primary goal of my dissertation was to improve risky decision making research from a systems perspective. Towards this end I worked at improving the techniques currently being used by both researchers and practitioners. More precisely following are the key take away from my work

- Development of RDMTk, an open source software as a service application tool to use in studying RDM.

- Analysis and improvement of reinforcement learning algorithm on distributed systems that is 40x faster compared to sequential version.
- Analysis and improvement of reinforcement learning algorithm on shared memory systems that is 130x faster compared to sequential version.
- Development of a Floor Tiles Planning (FTiP) theoretical framework for optimizing computational resource utilization in RDM algorithms.

The rest of the document is organized as follows. In Chapter II we briefly discuss previous work related to existing tools used in RDM empirical experiments, existing approaches in RDM data analysis and its high-performance computing solutions. Chapter III introduces an open source toolkit developed as part of the dissertation work. A formal model specification for reinforced learning algorithm for RDM is given in Chapter IV. In Chapter V we discuss a distributed memory based implementations and reduction in time complexities. Chapter VI lists the implementation of the algorithm using CPU-GPU combination. Further improvements to reduce the RDM algorithm in regard to computational resource and complexity is highlighted in Chapter VII. Chapter VIII introduces new approach to predict risk preferences in CUPS task. We then conclude with a brief discussion on directions for future advances in Chapter IX.

CHAPTER II

BACKGROUND

1. Introduction

We summarize the related previous work into three sections. The first, few paragraphs discuss tools, techniques and the need for a comprehensive toolkit for RDM researchers. The second section is about the need for improving computational performance in a RDM data analysis approaches. We foresee a necessity for improvement in terms of analysis methods and discuss major work done towards RDM predictions at the last section.

2. Tools and Techniques

Initially, RDM researchers relied primarily on self-reporting measures. Self-reports, however, were prey to both falsification of information and biases. Subsequently, many RDM researchers began to experimentally control participant behavior [59, 100, 7, 35, 92, 108] in order to assess the decision-making process. This experimental control often took the form of games or tasks. These tasks/games were shown to improve both the breadth and depth of assessment of risk-taking behavior. A number of tools have been designed and developed as a result. **Error! Reference source not found.** lists a selection of the most prominent assessments that were developed, along with the toolkits in which they are implemented. **Error! Reference source not found.** compares the toolkits and their features. Unfortunately, not all are available for use under

open source. Even for those that are open source developing applications to study different aspects of RDM is time-consuming. Therefore, we believe providing a prebuilt set of tasks/games will ease the data-collection process. We will now describe a few of the existing toolkits presently used by the RDM community.

Table 1 Some available toolkits used in RDM and their constituent tools.

	Angling Risk Task (ART)	Balloon Analog Risk Task (BART)	Delayed Discounting	Cups Task	Iowa Gambling Task	Go/No-Go Task	Stroop	N-Back
jsPsych	✓	✓				✓		
E-Prime		✓		✓	✓	✓	✓	✓
Millisecond Lib	✓	✓	✓	✓	✓	✓	✓	✓
Paradigm		✓		✓	✓			
PEBL		✓			✓			
Presentation							✓	✓
PsychMate		✓		✓	✓		✓	✓
PsychoPy		✓		✓	✓			
RDMTk		✓	✓	✓	✓		✓	✓
SoPHIE						✓		
SuperLab								✓
Tatool							✓	
Webexp2							✓	

Table 2 Toolkits and their features comparison.

	Customize expr. Start/End	Data Analysis	HPC Backend	Data Management	Eye Tracking	fMRI	Keyboard Input	Multiple Monitor Displays	Multi-Stimuli	Open Source	Platform Independent	Programmability	Requires Installation	Sound Recording	Speech Recognition	Timing	Web Based Availability
jsPsych	√			√			√			√	√						√
E-Prime				√					√			√				√	
Millisecond Lib	√			√			√				√	√	√			√	√
Paradigm	√			√	√		√				√	√	√				√
PEBL				√			√			√		√				√	
Presentation				√	√	√	√	√	√			√	√	√	√	√	
PsychMate		√		√			√						√			√	
PsychoPy				√			√			√		√	√	√	√	√	
RDMTk	√	√	√	√			√	√	√	√	√	√				√	√
SoPHIE	√			√			√			√	√						√
SuperLab	√			√	√	√	√					√	√				
Tatool	√			√			√			√	√						√
Webexp2	√			√			√			√	√						√

A. jsPsych

jsPsych is a library for developing psychological experiments using web-based technology. The JavaScript library is used to formalize the experimental setup and uses graphic notations that can be used during code implementation. It is an open source library made available to the research community by Josh de Leeuw at Indiana University [22].

B. E-Prime 2.0

E-Prime [85] toolkit provides a platform to design, data collection and analyze experiments. It has a graphical user interface to design games/tasks, which is back by a scripting language similar to visual basic.

C. Inquisit

Inquisit 4 [26] is web enabled application that gives flexibility to work from anywhere in the globe. It houses a wide range of psychological experiments and more can be developed using the software.

D. The Paradigm Experiment Builder

The Paradigm [48] experiment builder claims to be “One of a Kind”. Paradigm allows creating experiments without needing to learn complex user interface and scripting language. The developers think it is the easiest experiment builder available.

E. PEBL

PEBL [69] is a toolkit that is designed with its own programming language. This toolkit implements many standard tasks, and more can be implemented. It is supported on Windows, Linux and Macintosh operating systems and is implemented in C++, flex and bison. PEBL is provided freely under GPL license.

F. Presentation

Presentation [70] is a toolkit designed to run on Windows PC. It delivers many experiment/stimuli. It integrates fMRI, ERP, MEG, psychophysics, eye movements, reaction times and other performance measures. Presentation is the world's most popular experimental control software with 75172 registrations and 167505 downloads.

G. PsychMate

PsychMate [30] consists of classic and current experiments in psychology. It is used to teach students about research method and data collection. Students can also participate in experiments as real participants. Their data is submitted automatically over internet and complete analysis is provided.

H. PsychoPy

PsychoPy [76] is used in neuroscience, psychology and psychophysics experiments. It is an open-source application, free, powerful alternative to Presentation or e-Prime. It was developed in Python.

I. Psychtoolbox-3

Psychtoolbox [54] is free software provided under GNU GPL-2 licence. It is used to study psychophysics experiments in Matlab. Tasks/Experiments can be programmed using technologies included e.g. OpenGL, C etc. It supports sub-millisecond timing, vertical screen trace, and low latency audio. Data analysis is done through Matlab. It is designed to work under Windows, Linux and Mac OS.

J. SoPHIE

SoPHIE (Software Platform for Human Interaction Experiments) is developed and maintained by Achim Hendriks. This is an open source software product with a web interface. The project was initially intended to provide computerized support for experiments in economics, but has progressed into multiple development cycles with wider usage. It uses a Berkeley Software Distribution (BSD) style license [42].

K. SuperLab

SuperLab was initially designed to work on Max OS as a general purpose psychological experimentation toolkit. It allows to participation in experiments and data collection. Stimuli input can be given through RB Series, Lumina, SV-1, keyboard, mouse and etc. Output includes Stimtraker, Cedrus Response Device. Latest version of SuperLab also delivers cross-platform freedom: a license allows you to run it on either Mac or Windows [40].

L. Tatool

Tatool (Training and Testing tool) is a Java-based tool that was developed to assist researchers who wanted to develop computer-based training software, experiments, and questionnaires. It uses a platform-independent, object-oriented framework for designing experiments. It also provides predefined functions for configurable training schedules, adaptive training algorithms, and individual training statistics [96].

Other toolkits not reviewed in this paper also tend to focus on a limited set of tasks and do not scale well for global-scale studies of RDM. The RDMTk hopes to fill that gap.

Some of the tasks require additional software installation, and some require the development or customization of additional features. Custom-building an application for each aspect of a study is a time-consuming task. Hence, providing a prebuilt set of tasks/games under one umbrella that ease the data-collection process should allow researchers to focus on research questions and experimental manipulations.

Furthermore, nearly all the existing packages lack any features to manage data. In this period of free information exchange, active and healthy research should be supported by providing a platform on which reusable research data could be shared. This will not only reduce costs when conducting empirical studies but also diversify research activities globally.

3. High Performance Computing for RDM Models

A number of statistical and machine learning models are used to analyze data collected from empirical experiments studying risky decision making. For example the Iowa Gambling Task (IGT) task is studied using models based on reinforcement learning such as EV model, baseline model, Prospect Valence Learning (PVL) model, and PVL-Delta [1, 8, 17, 29]. For instance, reinforced learning (RL) models were successfully applied to categorize IGT experiment data for participants in healthy and substance abusers (cannabis users) by Fridberg et al. in [11]. Their analysis concentrated around how the controlled group performed compared to cannabis abusers and what were the individual differences. A very compelling case for using hierarchical Bayesian methods in cognitive science is made in [25]. Their survey work is interesting as they clearly layout the need for applying advanced statistical and machine learning algorithms. In short, they surmise that models should provide insight and understanding, facilitate prediction and generalization,

direct empirical exploration and account for group vs. individual performances. Even though each model discussed gave interesting insight into either individual or group performance; none of the models used was able to capture all or most of the individual level differences and subsets of groups. According to Steingroever et al. in [26]; none of the models considered (EV, PVL, EV-PU combination of EV & PVL) provide good fit across different experiment data and concluded that search for better analysis technique is still a work in progress. Authors in [27] support Steingroever et al.'s conclusion in [26], while distinguishing models into post hoc absolute fit and simulation methods. They emphasize accounting for models with absolute performance before using its results from comparisons to other model's adequacy (which is the case with post hoc absolute fit models).



Figure 2. Word cloud of article titles that refer to original ensemble clustering algorithm publication.

Ensemble clustering algorithm can be defined as a computational technique devised from using an agglomeration of clustering algorithms for meta-analysis on the data and was first proposed by Strehl and Ghosh [91]. Analysis of the word cloud (see Figure 2) from article titles obtained using Google scholar search for works directly citing Strehl and Ghosh's original publication on ensemble clustering algorithm gives us interesting insights. The most common constituent clustering algorithms in the ensemble so far are Spectral, Hierarchical/Agglomerative, Kmeans, Cmeans, Bayesian and Fuzzy clustering. It is used in the context of analyzing, learning, detection, and searching or discovering knowledge from images, gene expression, social text, brain, cancer, chemical and web datasets. The analysis also reveals that algorithm commonly uses a graph, and other high-dimensional structures stored as sparse or dense matrix data structures of different formats. It is interesting that little work is done in parallelizing it despite its popularity in data analysis scenarios which could easily lead to important problems in the big data arena [14, 36, 63, 66, 67, 89, 109, 111]. A few notable examples of algorithm's usage, tools and parallelization efforts are discussed next.

Glerean et al. [37] have used the ensemble clustering algorithm to identify group level analysis for finding differences among individuals suffering from autism spectrum disorder. Xu et al. applied the algorithm to recognize different individuals' handwriting [106]. Works such as [3, 62, 80, 97] are a few examples that discuss the application of the ensemble clustering algorithm to specific contexts and are made up of either single base clustering or more than one. Benmounah and Batouche [8] discussed the implementation of ensemble clustering algorithm using MATLAB's Parallel Computing Toolbox to analyze gene expression profiling. Literature also reveals MATLAB and R based tools such as LinkCluE [47], CLUE [44] and MATLAB Cluster

Ensemble Toolbox [83] in existence for specific problem scenarios. Ozyer and Alhajj [21] discuss a parallel implementation of a multi-objective clustering based on a genetic algorithm using a divide and conquer approach for analyzing large datasets. Their article does not give details of parallelization techniques but mentions that they used libraries from R statistical software as well as MATLAB [73]. Our two works are similar in the sense that both are trying to parallelize multi-objective clustering algorithms but are different as their approach is to partition dataset into smaller pieces to run in a distributed fashion whereas ours is designed for shared memory and CPU/GPU hybrid. Arnaldo et al. implemented an ensemble learning system called FLASH that utilizes GPU technology [4]. However, their work is very specific to genetic programming regression in shared memory and does not combine other classifiers.

We separate previous work specifically related to ensemble clustering into two sub topics. The first is, fundamental research work that produced improvements in the effectiveness of the algorithm to accurately partition the given dataset. The second is subsidiary research that undertook the necessary steps of making these algorithms run faster by addressing computational requirements.

The primary problem to improve algorithm effectiveness has been tackled by several researchers from an optimization perspective since its original proposal by Strehl and Ghosh [28]. These improvements to ensemble clustering algorithm are categorized as theoretical contributions via variations in the modeling and implementation algorithms. For example, optimizations can be based on graph partitioning algorithms, genetic algorithms, ant colony, greedy methods etc. [2, 9, 21, 24]. In general, to improve performance one needs to address each of the 2 aspects (clustering

ensemble and consensus function) either individually or together. We refer readers to one of the survey publications for detailed insight [12, 14, 30].

Secondly, one must also consider practical implications of using such computationally intensive algorithm in applications. Very little work is done for performance improvements that capitalize on today's readily available computation power; our work falls into this second category. There is some work that is an exception to above. For example, authors in [15] applied map-reduce technique to ensemble clustering. In their approach a given dataset is first partitioned and these data partitions are distributed in the map-phase that computes locally optimal labels, namely, partial output from objective functions, and in the reduce-phase partial outputs are condensed to form the final consensus. Their experimental results compared number of dataset size, models and nodes. They achieved linear speedup when models were kept constant, but data size was varied. However, researchers did not get encouraging results for varying number of models. Mohamad et al. [22] propose the use of parallelization techniques to improve performance of neural ensembles. ANN were trained on different partitions of data and combined to form consensus. Individual ANN of the ensemble learns different patterns by varying initial conditions, network architecture, training dataset, and training algorithm. Mohamad et al. also saw increased speedup with increasing number of processors. In [10] Fern et al. have published results from an empirical study of ensemble clustering techniques' capability to handle high dimensionality with respect to ensemble construction methods, and consensus functions. Their study found that random projections produce better diversity, and consensus functions based on bipartite graph partitioning performed better. To the best of our knowledge, utilization of parallelization technology for

improving computational performance of algorithms analyzing risky decision making is yet to be researched exhaustively; not much work has addressed this space.

4. Empirical Analysis of RDM Datasets

Researchers have applied several formal models and analysis techniques in the decision making domain. Throughout our background research we came across repeated discussions of not having adequate models that would account for variations in analyzing RDM data [26, 27]. Data analysis is, of course, one of the primary aspects for any experimental study. Analysis techniques must be tailored to specific application domain. Thus, a person performing data analysis should have prior knowledge of the data in context. Newman [71] used an exhaustive ensemble of clustering algorithms to generate better clusters and inferences. Other previous research related to analyzing decision making has typically used classification techniques such as k-means, agglomerative hierarchical, spectral clustering, and general linear model [93, 71, 17]. Much of the focus has been to find behavioral patterns in terms of groups of participants; in other words performance assessment is generalized in most cases and geared towards group averages. These participants were derived from a particular background or nature of study; for instance [24] discusses implication of substance use disorder (SUD, alcohol in this case) over risky decision making. Data analysis was done through general linear model using ANOVA toolkit. Acar and Yener in [1] surveyed use of multiway data analysis in neuroscience domain. Their study highlights that computational neuroscience is best analyzed with multiway models such as PARAFAC. Toolboxes such as ERPWAVELAB [68] have been developed earlier that runs on MATLAB platform. Toolbox was helpful in modeling electroencephalogram (EEG)/Event-Related Potentials (ERP) signals, functional Magnetic Resonance Imaging (fMRI) data. We have found

that most research work adapts similar approaches. The next section discusses approaches typically applied to analyze data.

4.1 Individual Differences vs. Group Behavior

Lauriola et al, [56] work is the first paper that discusses individual differences and predicting RDM among participants. Suhr and Tsanadis studied IGT in terms of Behavioral Inhibition Scale / Behavioral Activation Scale (BIS/BAS scale) and Positive Affect, Negative Affect Schedule (PANAS) [93]. Their results show that individuals higher in BAS Fun Seeking¹ performed worst on IGT, relative to individuals who were lower on BAS scales, and individuals high on only BAS Drive and Reward. They suggested a need to assess different dimensions when trying to assess IGT performance. One can also use graph-based approaches. Hofmans and Mullet develop in [43] a series of clustering procedures that can be used to study individual differences, integration rules, and also individual differences in other stages of information processing.

Horstmann et al. [45] hypothesized that performance on IGT depends on a combination of features rather than a single feature of participants. They used a system of linear equations to model IGT data that estimates weights to quantify the influence of each individual feature on decision making in IGT. Their model disentangles the individual features and quantifies the impact of features. Their cluster analysis of estimated feature weights also revealed sub-groups of participants with different weight patterns and hence decision behavior.

¹ BIS and BAS are components of Gray's Reinforcement Sensitivity Theory (RST). BIS & BAS indicate anxiety and reward situations, respectively. Also BAS is divided into three categories namely, Drive, Reward Responsiveness and Fun Seeking.

According to [98] risky choices or risky decisions that might be inferred directly or indirectly are domain specific, instead of a stable attitude or trait across scenarios. This work further explains:

- The content-specificity of risk taking, especially with respect to individual differences.
- A new approach that allows researchers and practitioners to assess both conventional risk attitudes and perceived-risk attitudes.

In [65], Maia and McClelland argue that there is a relation between working memory (WM) and performance on IGT. Working memory refers to a person's ability to remember symbols for a short term. The number of distinct symbols that one person remembers is called their capacity. Their study shows participants prior knowledge/bias towards the game. Most of their studies are conducted to explicitly test this behavior among participants. They also express interest in techniques that will be able to detect similar kind of participants exhibiting risk indifference.

4.2 Preferences in RDM

Recent research has suggested that one or more intuitive process contribute towards risky decision making [38, 5]. These intuitive processes are termed as implicit associations in neural mappings. In [84], implications for conceptualizations of value, risk aversion, inter temporal choice, and a dual-process theory of decision making is addressed. This work elaborates on relation / influence between intuitive process and deliberativeness on each other. IGT and Stroop have been used in situations that evoke emotional experience that may exert a covert influence on behavior [92]. It would be useful if we can derive implicit association between performed actions and map them to discrete states to quantify risky decision making. Examples listed below further

substantiate other similar work that attempts to predict hidden patterns from the data without prior configuration about expected patterns.

Data driven classification approach offers a fundamental difference from the existing algorithms such as k-means, SOM and GMM where one assumes prior knowledge of patterns or expected functional curve. In case of hierarchical clustering, algorithms prune data in an ad-hoc manner. For instance, Ma et al. in [64] introduced Smoothing Spline Clustering (SSC) model that uses properties of gene expression over time. It allowed discovery of related pattern of gene expression and underlying functions without prior specification of either the cluster number or the functional form.

In [39] Golland et al. applied a hypothesis-free, unsupervised two-class clustering algorithm (k-means) to a large set of fMRI data. Their study mapped cortical activates according to tasks or decision making. Their clustering results confirmed that the intrinsic–extrinsic subdivision constitutes a fundamental cortical divide. Their approach contrasts with others by demonstrating the benefit of using clustering to construct a top-down model of global activation patterns in the brain. Their study was repeated for different values of k and found encouraging results of delineation at finer levels (when $k=3$) whereas it was consistent when $k=2$. This approach demonstrated applicability of neuroanatomical research in revealing data-driven subdivisions within the human cortex.

4.3 Predictive Modeling of RDM

Dougherty and Thomas propose General Monotone Model (GeMM) as a predicting algorithm for predicting rank orders from a set of predictor variables. They used it to study

psychological data that has nonlinear relations in the data without a need to make precise assumptions. This is important to our work as this substantiates the need for researchers to have tools that allow flexible modeling [25].

In [71] L. Newman used linear discriminant based methodology to validate reinforcement learning model's fit by predicting cluster. He used data from IGT, and his model accounted for individual differences. Using his models Newman was able to identify the existence of multiple decision making styles previously unknown.

Data collected from MRI sensors while participants completed the BART task was used in [41] to predict if individuals would make risky or safe decisions. This method was able to predict correctly 71.8% of the time based on activity centers in the brain. This study also shed some light on the amount of required data density for prediction accuracy.

CHAPTER III

RDMTk: A TOOLKIT FOR RISKY DECISION MAKING

1. Introduction

Decision Making (DM), and in particular Risky Decision Making (RDM), has become a cross-disciplinary field of interest. Many people ask "Why do some people have better life outcomes than others?" The economist might ask "Why do some people make certain financial decisions?" Psychologists wonder "Why do certain people have a higher appetite for risk than others?" The organizational behaviorist ask "How will these decisions affect the organization?" Neuroscientists wonder "Are there certain areas of the brain that are tied to risky decision-making processes?" The computer scientist ask "Can networks predict when actors will make risky decisions?" And these are just a few of the many ways in which risky decision making has become a widely studied phenomenon. Unfortunately, this analysis is often done independently across different fields, using disparate toolkits, disparate analysis methods, and only moderate cross disciplinary pollination. There are no universally accepted tools and measures for risky decision-making. Historically, assessments of risky decision making were made using self-reports as measuring instruments.

As the field has evolved, data from previously published research has shown that not all individuals possess the ability to assess and accurately report on their behavior. Our assumption is that, under uncertain situations, even the most sophisticated response formats, including multiple-choice, multiple-selection, short and extended constructed-response and performance task, etc. are

inadequate to account for the complex cognitive processes involved in the decision. Diagnostic instruments constructed in laboratories offered better self-report tools at the cost of smaller test subject pools. Since then, the use of experimental manipulation has become prevalent. For example, psychologists would often present participants with hypothetical decision games [18]. Economists began to present participants with lists of hypothetical scenarios [50]. These experimental manipulations naturally found their way to computerized interfaces, which enabled ease of use in manipulation, data collection, and scale. More recent developments show test takers prefer internet based assessments over paper based (Chapter 7) [86]. Virtualized versions of in laboratory measuring techniques using computer technologies offer cost effective and enhanced replicas of the same.

Reservations against incorporating computer and information technologies in psychometric measurements were much debated during the 1980's and 1990's; including over what is possible and the accuracy of results, compared to paper based methods. The situation has changed today, as computers are now available in much wider forms such as smart phones, tablets, surface and touch devices when compared to previous decades [86]. With the widespread acceptance of computer usage, International Test Commission, Inc. (ITC) [87] has developed guidelines for computer-based and internet-based testing in psychometric assessments.

As a result, the number of techniques and packages exploded. E-Prime [85], Inquisit 4 [26], MouseLabWeb [102], PEBL [69], PsychMate [30], PsychoPy [76], The Paradigm experiment builder [48], Presentation [70], SuperLab [40], SurveyWiz and FactorWiz [11], Visual DMDX [33], Webexp2 [52], WEXTOR [81] all provide tasks that can be used for RDM analysis experiments. There are also several laboratories such as the Laboratory for Cognitive and Decision

Sciences [78], Laboratory of Biological Dynamics and Theoretical Medicine [75], and The Brain and Mind Research Institute, which have historically focused on developing such tools. Experimenters often pair the tools with specialized add-ons. For instance, The Black Box Toolkit [77] is designed to give precise timing control and tracking for psychology researchers to remove timing errors.

This proliferation of disparate packages for RDM research has created a burden for researchers in the design and operationalization phases. Further, almost none of these tools have been designed to scale to a global community and plug into a variety of experimental methods, incorporate modern analytical tools geared for machine learning. None have been accepted as standards.

In this chapter, we introduce Risky Decision Making Toolkit (RDMTk), a new toolkit used for researching decision-making processes and related activities made under risky situations. RDMTk provides researchers with a scalable, reliable, open-source experimentation and analysis framework. There has been significant work dedicated to producing tools to support RDM research, but there are several areas still to be addressed. Many of the single instruments currently used in a laboratory environment fail to capture RDM's multidimensional nature on their own and researchers are required to use a multitude of packages or design their own kits. By providing it all in one place, at scale, the toolkit hopes to improve the quality of Risky Decision Making research. Also, RDMTk tries to incorporate suggestions from ITC and can be used on multiple computing devices types such as smart phones, tablets, surfaces devices, etc.

2. RDMTk- An Overview

Technological innovations are changing the dynamics of conducting empirical experiments for behavior analysis. We created this toolkit in response to the open challenges faced by RDM researchers, such as the burden created during the design and operationalization phases because of the proliferation of disparate packages. The RDMTk toolkit is designed around a one-stop-shop design philosophy. It offers ways, to not just build and present experimental tasks but also to store and analyze data, using statistical, mathematical and analytical (machine learning) packages. RDMTk is available under the “Software as a Service” (SaaS) model and is provided under a General Public License (GPL); hence, RDMTk is an open source project. It is a free environment that can be used to conduct experiments on a global basis. RDMTk allows resources to scale, and can be used in a variety of environments in order to obtain different types of measurements. The initial release of the toolkit incorporates the six most popular experimental paradigms. However, it is the authors’ intention to develop and incorporate much more over time through community collaboration. Our aspiration is for RDMTk to become the preferred toolkit for studying risky decision making, which will encourage global research studies, as well as simplified data collection and sharing.

Like many open source projects before, our toolkit is a proposal that needs to be nurtured through community collaboration, and contribution. By incorporating best practices, appropriate tools, and relevant resources, it is our intention to develop RDMTk as a state of the art expert system geared towards studying and analyzing risky decision making in the future.

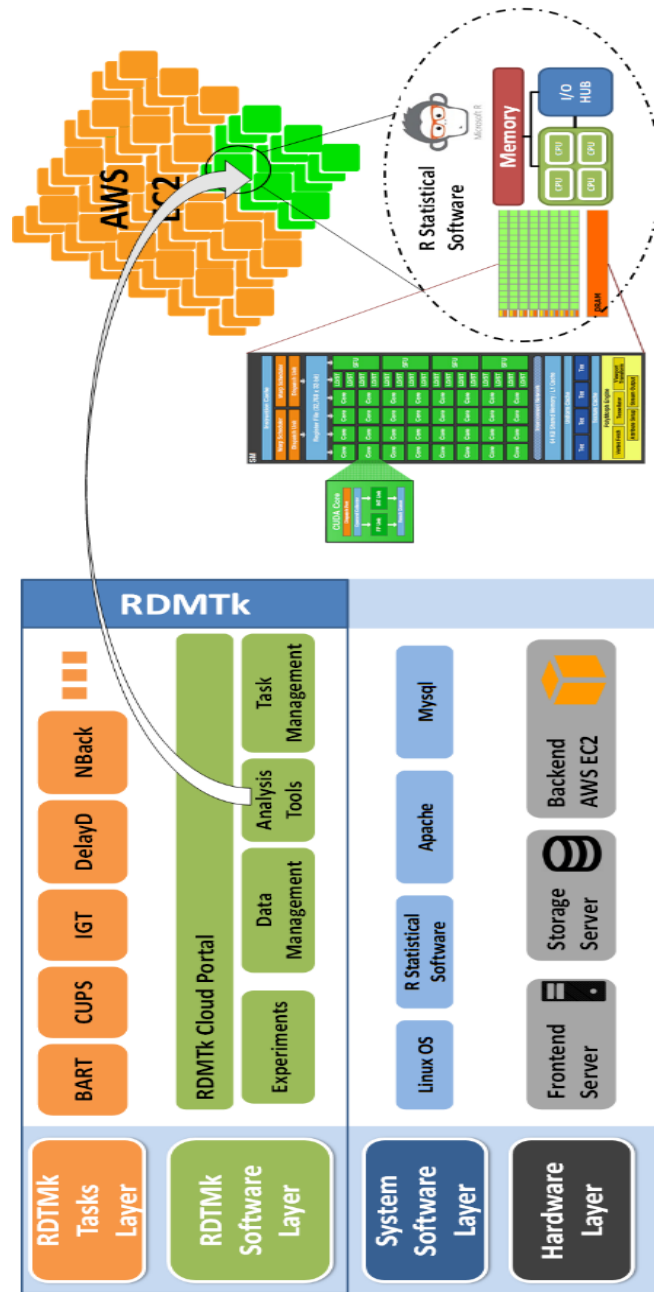


Figure 3. Overall design and architecture for RDMTk.

Recent advances in Computer Science bring a number of innovations that haven't found their way into existing RDM tools. Development in cloud computing bring new options for resource and infrastructure sharing. The open source framework of RDMTk provides extended

analytical capabilities to help researchers create, manage, conduct, monitor, collaborate, and analyze their experiments on a global basis. RDMTk studies can be integrated with such popular crowd-sourcing labor markets as Amazon's Mechanical Turk (mTurk), Google Forms, Qualtrics, and Survey Monkey. As shown in Figure 3, the analytical framework of RDMTk is built on MRAN, a Microsoft R statistical programming platform running on Amazon Web Services Elastic Computing Cloud (AWS EC2), a computer cluster on the cloud.

By bundling robust data analysis tools with features for supporting empirical experimentation, we hope to open doors for collaboration between researchers from diverse backgrounds across political boundaries. This toolkit should significantly improve the quality of research for everyone in the RDM / DM areas.

Consistent with recommendation from ITC, toolkits like RDMTk should support three types of accounts: Administrators, Researchers, and Participants. Our emphasis has been on the last two account types - researchers who are designing and conducting empirical experiments and those who are participating in the experiments. The interface for each account type takes the form of a customized dashboard with a toolbar at the top, menu items on the left, and a central display section on the right. Below is a description of available account types.

a. Administrators

The administrator is the person who takes responsibility for the overall operation of the RDMTk application. The administrator is the point of contact for troubleshooting problems and for resolving issues related to deployment and accessibility.

b. Researchers

RDMTk distinguishes between two types of researchers:

- 1) Researchers who are focusing on task design and development; these can be categorized as task or paradigm developers.
- 2) Researchers who are conducting empirical studies, running tests, interpreting data, and communicating results to the wider community.

c. Participants

People using RDMTk for taking part in an experiment are categorized as participants. There are multiple ways for participants to access experiments. Participants can access the experiment at their convenience via mTurk, Qualtrics or through another online labor and data collection framework.

The use of a standardized interface to access the software makes using the toolkit simple and intuitive.

3. Technical Details

When implementing RDMTk, we relied on the most current technologies and practices. This is expected to facilitate contributions from other developers in future development cycles and also to encourage contributions from people who have a non-technical background. As illustrated in Figure 4, the design of the toolkit uses a four-layered architecture: (1) RMDTk tasks layer, (2) RDMTk software layer, (3) system software layer, and (4) the hardware layer. RDMTk is currently provided as a SaaS (Software as a Service) product, hosted by the WiSe (Wireless Sensornets) Lab within Western Michigan University's Computer Science Department.

Each task within the toolkit is implemented as an independent application that integrates itself as an add-on to the remaining layers. Tasks in RDMTk can be developed using the latest web development technologies, such as HTML5, web 3D rendering libraries, and PHP, with MySQL providing the underlying database support. Tasks use web animation technologies such as GreenSock's TweenLite and TweenMax in combination with Three.js. This provides a lightweight, extremely fast and flexible animation framework for realizing new tasks in the RDM domain.

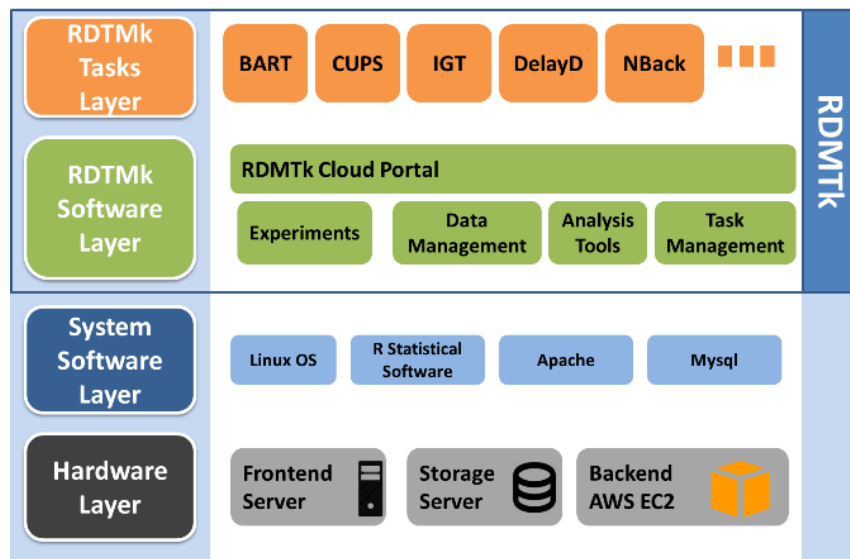


Figure 4. RDMTk's four-layered architecture.

A. The RDMTk Software Layer

RDMTk was developed using PHP 5.3, JavaScript, Laravel 4.1/5, and MySQL. The toolkit is structured around a dashboard-based UI design that provides support for the three different types of accounts (Administrators, Researchers, and Participants) described in Section 2.

B. The System Software Layer

To the extent possible, open source technologies were used to implement RDMTk. Currently, RDMTk is hosted on a LAMP server running the Linux-based Ubuntu 14.04 operating system, Apache web server, and MySQL database. The R statistical software is used as the main workhorse for analysis. R is exposed to the web using the OpenCPU library.

C. The Hardware Layer

The underlying server architecture is divided into three layers - a front end server, a storage server, and Amazon EC2 instances for analysis on the backend. Instances in Amazon EC2 are configured to use HPC technologies such as OpenMP, MPI, and GPGPU-CUDA depending on the analysis model. We expect this configuration will change over time as computing needs evolve.

RDMTk is provided under GPL and is an open source project. We hope the community will find the idea and principals behind developing this toolkit appealing and that they will contribute towards its success. We refer the reader to the related technical documentation on the toolkit website for more details.

D. Security and Authentication

RDMTk implements strict application-level security through two access measures:

- 1) A unique ID is used by unregistered users to access the tasks landing site. In the case of mTurk, it is the mTurk Identification (MID).
- 2) A unique username and password is needed for full access to RDMTk, with login controlled using the SSL (secure socket layer) protocol.

Administrator accounts are created manually on an as-needed basis. Other accounts can be created from the RDMTk login screen; the individual creating the account will be required to select either "researcher" or "participant" access.

RDMTk stores all sensitive information, including account passwords, using a Bcrypt hash. User access to dashboard operations is restricted on the basis of the user's specific account type. User logins are monitored for inactivity, and are automatically logged out after 3 minutes of inactivity. Finally, access to the high-performance computing resources on AWS EC2 is tracked on the basis of pre-configured instance credentials associated with the specific analysis being performed.

4. Integrating New Tasks and Games into the RDMTk Framework

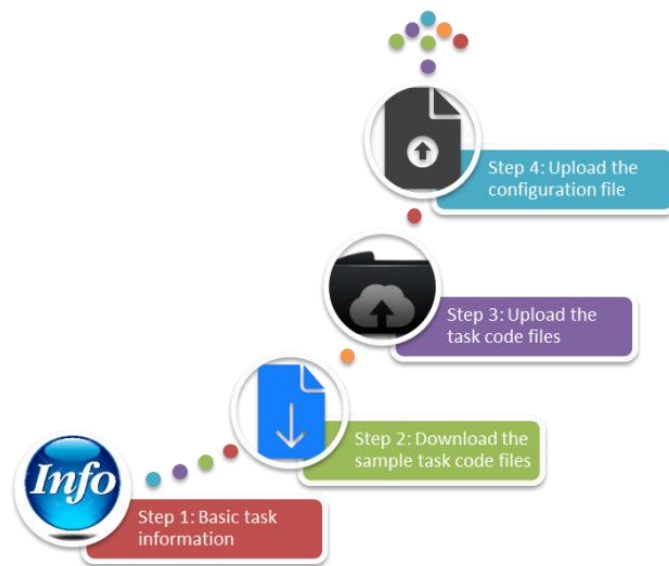


Figure 5. Step involved in integrating a new task into RDMTk.

The current configuration of RDMTk includes the six most commonly used RDM tasks. However, it is our intention is to allow collaborating researchers to readily integrate new tasks into

the existing toolkit deployment. To support this, we have developed an easy to use interface for adding new tasks to the toolkit.

The addition of a new task, which is only available to researchers, is initiated via the ‘Add a New Task’ menu option; Figure 5 shows the four steps required to add a new task; the individual steps are shown below.

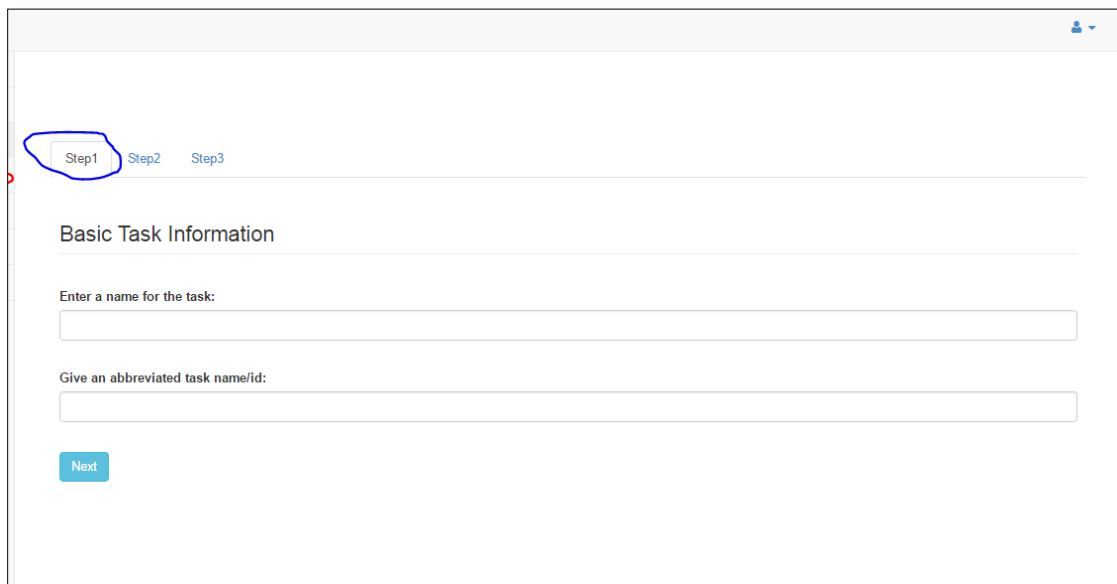
The image shows a web-based form titled 'Basic Task Information'. At the top, there are three tabs: 'Step1', 'Step2', and 'Step3'. 'Step1' is the active tab and is highlighted with a blue circle. Below the tabs, the form has two text input fields. The first field is labeled 'Enter a name for the task:' and the second field is labeled 'Give an abbreviated task name/id:'. Both fields are empty. At the bottom left of the form, there is a blue button labeled 'Next'. The form is enclosed in a light gray border with a small user icon in the top right corner.

Figure 6. RDMTk step 1 - form to add a new task to the toolkit.

Step 1: Provide basic task information–

The researcher must first enter the task name and the task id. Since the task name and task id must be unique, RDMTk will prompt the user to provide different values if either value is already in use (Figure 6).

The screenshot shows a web interface for Step 2 of the RDMTk process. At the top, there is a progress bar with three steps: Step 1 (active), Step 2, and Step 3. Below this, the heading "Upload task's code" is displayed. Under the heading, the text "Task's codebase" is followed by a "Choose File" button and the text "No file chosen". Below this, a paragraph states: "Upload a .zip file consisting of all the files related to new task's code. This codebase should be written in php and be able to operate as a self sufficient web application." At the bottom of the form, there are two buttons: "Next" and "Download template codebase as zip file". A yellow warning box at the very bottom contains a bell icon and the text: "Please name the index page for the task as 'task.php'".

Figure 7. RDMTk step 2 - form to add a new task to the toolkit.

Steps 2 & 3: Upload the task code files –

The researcher next uploads the zip file that includes all the task files to be uploaded. A sample zip file can be downloaded as an example for specific details needed (Figure 7). This uploaded file should reflect the modified computer code primarily written in php and JavaScript programming language.

Figure 8. RDMTk step 3 - form to add new task to the toolkit.

Step 4: Upload the configuration file –

Finally, the researcher needs to provide a description of the database table (or tables) needed to support the task. This description is provided as an xml file named config.xml. Note that any table names are restricted to contain only letters, digits, and the underscore character. For further information, you can examine the sample file that is available on the upload web page.

The xml file is expected to have the following tree-like structure:

```
<tables>
<table name="table_name">
    <field_name type="data_type">Field Name 1 </field_name>
    <field_name type="data_type"> Field Name 2 </field_name>
    ...
    ...
</table>
<table name="table_name">
    ...
    ...
</table>
...
...
</tables>
```

The supported data types are restricted to 'integer,' 'float,' 'string,' and 'DateTime.' Also, the script automatically creates a primary key field named 'S_no' for each new table. This primary key field assigns a unique sequence number to each new record that is added to the table.

As presently configured, RDMTk includes six commonly used risky decision making tasks - four "risky decision-making tasks" and two "lower-level cognitive tasks." A major feature of RDMTk, though, is the ability to readily add new tasks to this set. The tasks are implemented in a

manner that allows a researcher to readily configure the task for specific applications. Assessment processes and delivery are designed to be consistent across all experiments and participants. The specific tasks already included in RDMTk are listed discussed next.

5. Risky Decision-Making Tasks

A. Iowa Gambling Task

The Iowa Gambling Task (IGT) is one the more popular tasks used to study RDM [7]. Participants are given four decks of cards in the game. Each deck has a monetary reward as a payout and offers varying amounts as payouts during trials. A payout can be advantageous, disadvantageous or both. Participants initially begin with \$4000 and try to make as much money as possible. Note that the original design of this task [6] used an initial stake of \$2000. Our current implementation supports only one payout scale as given in Table 3; future releases will accommodate multiple scales.

Table 3 Iowa Gambling Tasks payout scale included in RDMTk toolkit.

Deck	Gain amounts	Loss frequency	Loss amounts	Expected value
A	\$80 to \$170	50%	-\$150 to -\$350	-\$72
B	\$80 to \$170	10%	-\$1250 to -\$2500	-\$72
C	\$40 to \$95	50%	-\$25 to -\$75	+\$32
D	\$40 to \$95	10%	-\$250 to -\$375	+\$32

B. Balloon Analog Risk Task

The Balloon Analog Risk Task (BART) is a digital game that was initially developed for use in the laboratory. The participant is presented with a balloon that can earn a larger payout with each user click. The participant can cash-out at any time. With each click, the balloon becomes more inflated. At some point, a threshold is reached, the balloon pops, and the participant gets no

payout. The task measures actual risky behavior similar to real-world situations involving impulsivity. Riskiness is rewarded up to a point at which further risk results in poorer outcomes [59]. RDMTk's BART implementation uses Three.js in combination with TwelineLite and TimelineMax to create a 3-dimensional balloon that spins around its vertical central axis like a real balloon. During each trial, a new balloon of a different color is presented to the participant.

C. Cups Task

The Cups Task was developed to assess decisions about potential gains and potential losses as well as relationship of these decisions to different neural structures [100, 108]. In their study, the authors used cups to analyze adaptive decision making under risk. They examined whether an individual's ability to make adaptive decisions differentially for gains and losses is affected by either damage to neural structures, or by changes in the subject's emotional state. The Cups Task is a simulation that forces participants to make risky decisions by forcing them to choose from gain and loss domain cups. These cups can either contain a reward or deduct an amount based on the domain from which they are derived. The participant's goal is to increase the reward amount as much as possible. The payout scale for the implementation in RDMTk is as given in Table 4.

Table 4 CUPS tasks payout scale included in RDMTk toolkit.

Gain Domain		Loss Domain	
Number of Cups	Points	Number of Cups	Points
2	2	2	-2
3	3	3	-3
5	5	5	-5
2	3	2	-3
2	5	2	-5
3	5	3	-5
3	2	3	-2
5	2	5	-2
5	3	5	-3

D. Delay Discounting Task

In this task, the participant is presented with a series of choices related to the receipt of differing amounts of money. For example, the participant might be asked to choose between "Receive \$1.00 now" or "Receive \$10.00 in a year's time". In this task, there is a predefined set of sample questions that can be used by the researcher or the researcher can create a unique set of questions which cannot be seen by other researchers. These questions can only be modified by the researcher who created them.

Lower-Level Cognitive Tasks

E. Stroop Task

The Stroop task was designed by Stroop [92] for studying the effect of interference on performance. During the test, participants are presented with words written in different ink. In each trial, the participant is required to recognize the color of the ink rather than the word. For example, a word written in blue ink may be more difficult to recognize than a word written in red ink. The time required to recognize the ink color when given an advantageous trial (the word and ink in the same color), a disadvantageous trial (the word and ink in different colors), and a neutral trial (the word and ink have no correlation) is recorded.

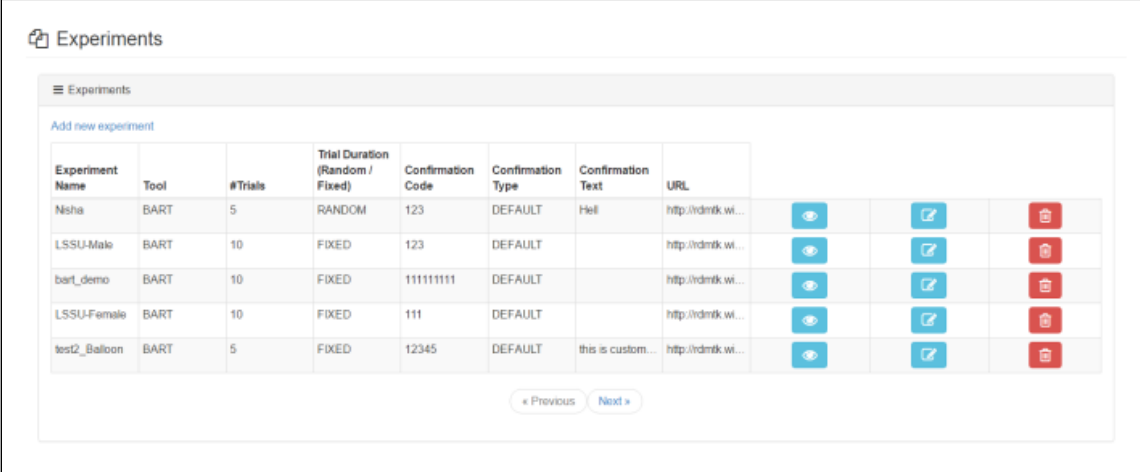
F. N-Back Task

In this task, participants are presented with a series of stimuli in the form of alphabetic letters. The participant's task is to detect whether the current letter stimulus matches with the letter shown N times earlier. The challenge can be made more or less difficult by adjusting the value N, the number of letters the participant should remember. The N-back task was developed by Kirchner as part of his research into short-term memory [53].

Researchers can create experiments based off any of these six preinstalled tasks or other tasks that were integrated into RDMTk by other contributors.

6. Experiments

The 'Experiment' phrase is a generic term used inside RDMTk that involves a number of different phases of the experimental design procedure. Traditionally, a research group decides on a research question or hypothesis before a target audience or any other specifics are identified. RDMTk has features allowing the researcher to navigate through this experimental design process.



Experiment Name	Tool	#Trials	Trial Duration (Random / Fixed)	Confirmation Code	Confirmation Type	Confirmation Text	URL			
Nisha	BART	5	RANDOM	123	DEFAULT	Hel	http://rdmtk.wi...			
LSSU-Male	BART	10	FIXED	123	DEFAULT		http://rdmtk.wi...			
bart_demo	BART	10	FIXED	111111111	DEFAULT		http://rdmtk.wi...			
LSSU-Female	BART	10	FIXED	111	DEFAULT		http://rdmtk.wi...			
tesQ_Balloon	BART	5	FIXED	12345	DEFAULT	this is custom...	http://rdmtk.wi...			

Figure 9. RDMTk experiments page.

Upon selecting the Experiments menu item, all of the experiments that have been created are listed in the main display area (see Figure 9). This screen allows new experiments to be set up as shown in Figure 10. Researchers are given more control over the experiments during the creation phase. Researchers can select different types of tasks, the number of trials, the trial response type, and the experiment end type. These experiments can be integrated into modern survey tools such as mTurk and Qualtrics by accessing the unique URL created for each experiment and is accessible through the view button next to each experiment in the list of experiments (Figure 9).

The screenshot shows the RDMTk 'Create Experiment' interface. It features a sidebar on the left with a 'Experiments' link. The main content area is titled 'Create Experiment' and contains several form fields: 'Experiment Name' (text input), 'Task name' (dropdown menu with 'Please select one option'), 'Number of Trials' (text input), 'Trial End Type' (dropdown menu with 'Please select one option'), 'Confirmation Page Type' (dropdown menu with 'Please select one option'), 'Confirmation Code' (text input), and 'Custom Text' (checkbox and text area). At the bottom, there is an 'Add-On Features' section with a checkbox for 'Enable Mouse Tracking' and a note stating 'Mouse Tracking is currently not supported with STROOP task.' Below this are 'Submit' and 'Cancel' buttons.

Figure 10. RDMTk create experiment screen.

Some of the procedures from experimental design such as deciding on the data required for the experiment can be performed at this stage. The toolkit allows users to choose one among the tasks described below. We have also implemented an optional feature that records the participant's mouse tracking.

7. Collecting Data

Participants' performance on the task is monitored to gather relevant information and this data is stored for later analysis. Researchers can derive a more meaningful conclusion if they have detailed insight into a participant's behavior during the experiment. RDMTk enables geographically-separate research teams to collaborate through sharing their experiments and data as participants are not restricted to a single location. Participants' performance data are kept

confidential and is only made available to the administrators and researchers who own the experiment. Data can currently be downloaded as an Excel file. Currently, toolkit does not support researcher specific access control on the experimental data. However, future versions will incorporate such features.

A participant's decisions are captured during a task and stored in a MySQL database for future analysis. RDMTk can collect additional data during an experiment in addition to the task-specific data. For example, collecting response times, mouse movement, or eye tracking provide a much more detailed perspective on a participant's behavior during an experiment. The toolkit currently tracks the time taken by participants during each trial. Tracking mouse movements for each participant is an optional feature that can be enabled at any point. Additional sensory features are planned to be integrated in the future.

Generally, data have variable characteristics. For instance, the data collected from participants have different variability and visualization features. These variations exist because each researcher may have his own unique approach to the same problem. Using RDMTk, many researchers may use similar tools/tasks to conduct experiments with minor or no modifications.

8. Experiment-monitor

The experiment-monitoring feature in RDMTk gives real-time status and a progress tracker for a selected experimental design. This tool helps a researcher to assess the number of participants that took part in the experiment and gives cues on when to stop recording data. The experiment monitor is built based on power analysis or power test. Unlike traditional methods, this feature allows statistical significance of the data collected dynamically or on the fly compared to a statically predetermined sample size. The experiment monitor feature needs to be configured by

creating an experimental design type for a task type. Previously set experimental designs are listed in the right-most drop-down list illustrated in Figure 11. Selecting one of the experimental designs will start the monitoring feature as shown in the figure.

Figure 11. RDMTk form to monitor real-time experimental status and progress tracker.

9. Data Analysis

RDMTk provides data analysis and visualization capabilities primarily through R statistical software, which is exposed to the web using the Open CPU library [72]. Analytical features are run on Amazon AWS EC2 in the back end. RDMTk needs to be bootstrapped with AWS EC2 before a researcher can use the analytical features on the cloud. Because high computational power is a necessity, an analysis back end integrated with Amazon's AWS EC2 allows usage of the latest HPC technologies and opens large-scale big data computer capabilities to RDMTk. A researcher is no longer restricted to the limitations of his or her workstation and can now leverage Amazon's cloud computing capabilities. The current RDMTk version implements Base Model, Random Model and EVL model [12] for the IGT paradigm.

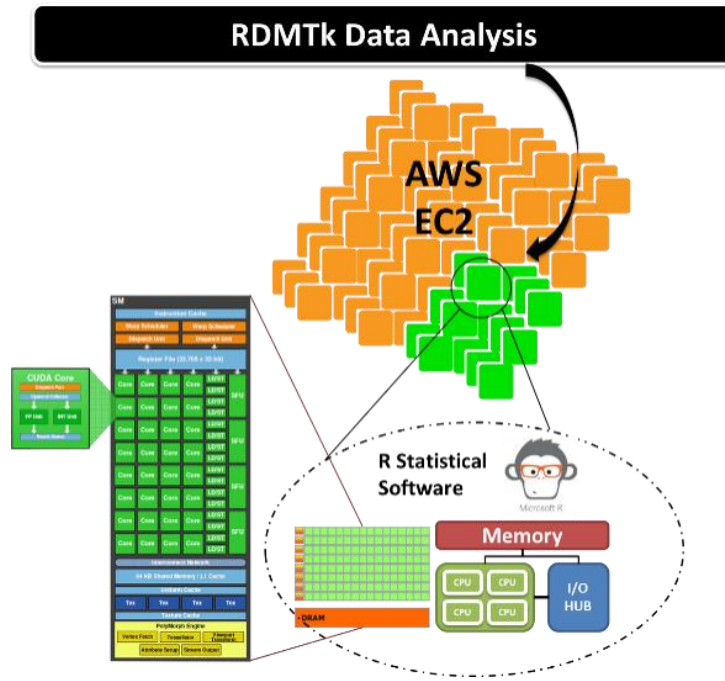


Figure 12. RDMTk's data analysis module on amazon AWS EC2 infrastructure.

10. Bootstrapping RDMTk and Amazon Elastic Compute Cloud

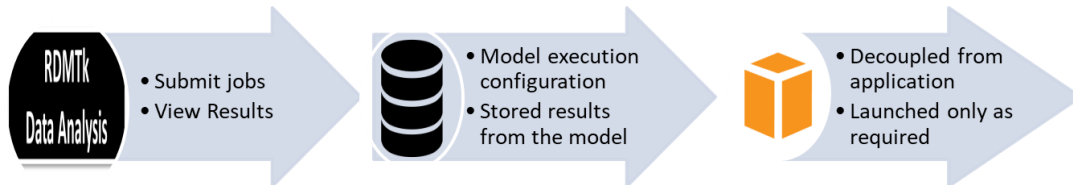


Figure 13. RDMTk and amazon EC2 Bootstrapping.

Researchers are able to use the analysis features after creating an AWS account and will need to bootstrap their RDMTk account with an instance of an AWS EC2 machine. RDMTk decouples the analysis model selection, submission, and execution internally into three different stages as shown in Figure 13. Submitted jobs and results are available on the toolkit dashboard. Once models are selected for execution, appropriate configuration entries are created in the database. Upon successfully configuring the database, a pre-configured EC2 instance is launched

to execute these models. Results from successfully executed models are stored back in the database and made available to the researcher.

One typically conducts these studies involving a sample of participants. Analyzing experimental data for appropriate inferences is an extensive process that usually requires the use of enhanced tools from a statistical or mathematical software package. Different models are used to analyze the data collected from these studies. Modeling RDM can be approached from various perspectives.

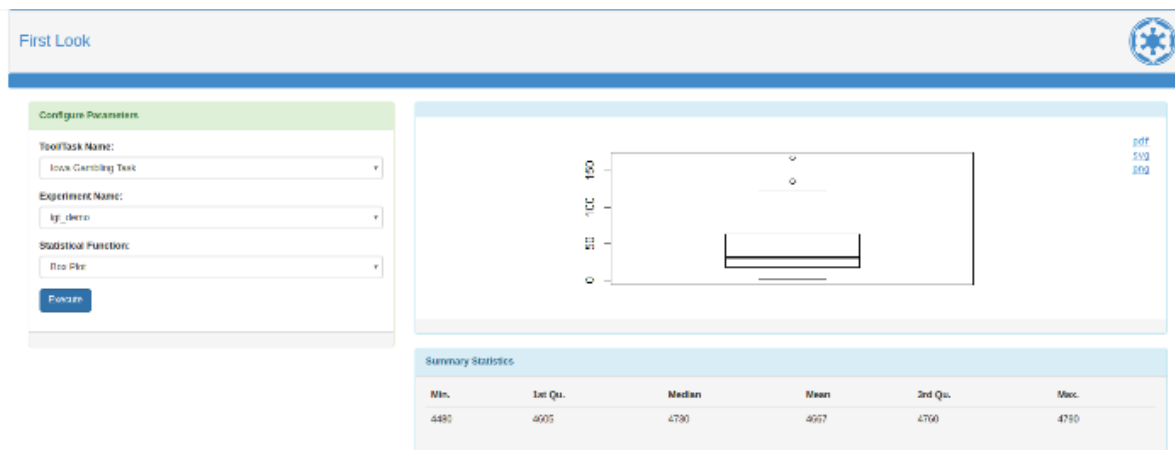


Figure 14. Summary statistics for an RDM experiment.

Our approach is to enhance these techniques and automate them in comparison to what currently exists - whether a researcher is interested in the identification of individual differences in participant performance [71, 57] or looking for traditional generic group behavior. As more advanced techniques are still currently being developed, we have provided some commonly used models in the current release of the toolkit. We have implemented tools to give summary statistics (see Figure 14) for selected experiments.

11. Contributing to RDMTk Development

The source code of RDMTk is provided under GPL license. It is available on Github.

Code: <https://github.com/guptagithub/RDMTk>

Live: <https://rdmtk.wise.cs.wmich.edu/>

12. Summary

Research into Risky Decision Making (RDM) has become a multidisciplinary effort. Conversations cut across fields such as psychology, economics, insurance, and marketing. How and why an individual makes decisions concerning risk is an important concern. This broad interest highlights the necessity for collaborative investigation of RDM to understand and manipulate the situations within which it manifests. Technological innovations modify and transform traditional methods to fit the needs of this new paradigm and open new possibilities. A holistic understanding of RDM has been impeded by the independent development of diverse RDM research methodologies across different fields. Many behavioral assessment tools have been used, including behavior observation, self-reports, assessments, interviews, and tests. However, there is no software specific to RDM that combines paradigms and analytical tools based on recent developments in high-performance computing technologies. This paper presents a toolkit called RDMTk, developed specifically for the study of risky decision making. RDMTk provides a free environment that can be used to manage globally-based experiments while fostering collaborative research. The toolkit's one-stop-shop philosophy provides access to tests and analytical features used in the context of RDM. The integration of RDMTk with external tools such as Amazon AWS, mTurk, and R further facilitates this. The incorporation of machine learning and high-performance computing (HPC) technologies in the toolkit further open additional possibilities such as scalable

algorithms for analyzing non-experimental data sets and big data problems arising from global scale experiments. By fostering collaboration through community support, we hope RDMTk becomes the preferred toolkit for studying RDM.

CHAPTER IV

REINFORCED LEARNING MODEL USED IN ANALYZING RISKY DECISION MAKING

1. Introduction

Researchers design and use game/task(s) that allow one to capture data for specific aspects of decision making. Balloon analog risk task (BART), Cups task (CUPS), and Iowa gambling task (IGT) are some commonly used tasks in studying decision making behaviors [3, 19, 31]. RDM research involves analyzing huge amount of data collected from experiments based on individuals taking these or similar tests and recording various parameters.

Analyzing RDM data collected from experiments involves various steps. Through analysis one might want to (i) discover associations between participant's performance/behavior to task's core phenomena, (ii) identify individuals and groups for similarity, (iii) identify individual differences, etc. The discussion revolves around developing better statistical and/or machine-learning techniques that could be used to identify similar groups as well as individual differences in data collected through empirical studies [4, 25].

Combinatorial and multi-objective data analyses are gaining traction among many data scientists. This is particularly true in life sciences and related research areas, such as analyzing risky decision making (RDM) behaviors, where initial sample sizes are small when compared to contemporary counterparts involving big data scenarios. While efforts are also under way to increase logistical capabilities for collecting and analyzing data from a large pool of participants

both at experimental and industrial scales, underlying algorithms will be same or similar in nature.

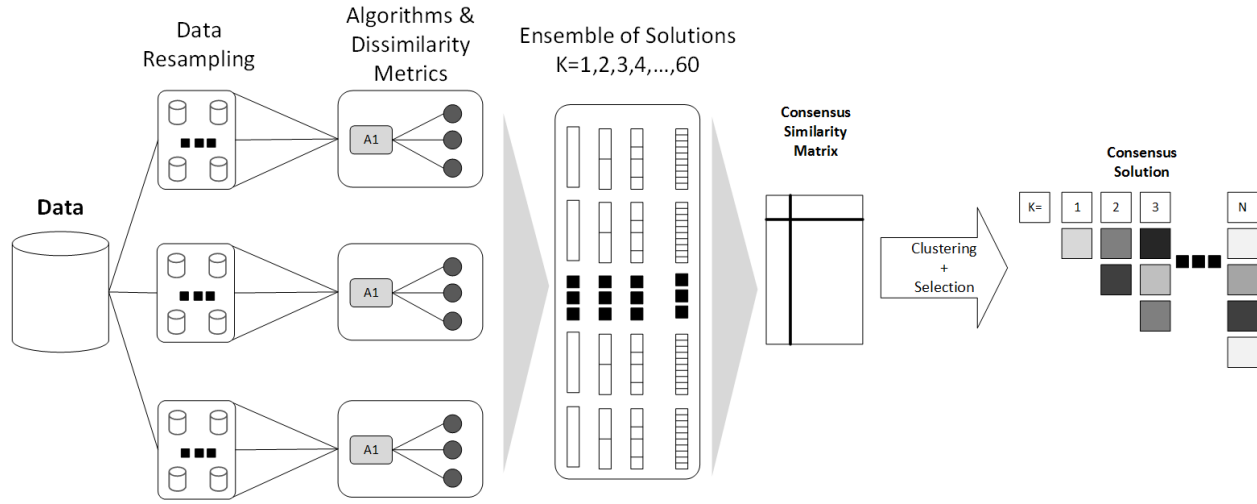


Figure 15. Conceptual overview of ensemble clustering solution [23].

The ensemble clustering algorithm, which is also known as meta-clustering or ensemble approaches, was originally proposed by Strehl and Ghosh [91]. Using an ensemble of clustering methods in combination instead of individual classifiers yields a superior technique. It improves the robustness, quality of the classification, and experimental results [79, 112].

A promising reinforcement learning model is applied to RDM by Newman in his PhD dissertation in [23] and was shown to provide better insights compared to other RDM studies (see Figure 15). Taking a new approach to studying RDM, he applied a number of clustering algorithms on resample data forming an ensemble. Results from the individual algorithms are gathered and summarized through a consensus function that is customized for IGT. We will explain the algorithm in greater detail in the reminder of the chapter. Among the computational intelligence community this approach is commonly known as ensemble clustering. Newman's work provides greater insights into RDM and is different from traditionally used models. Even though his

approach marks an important milestone for applying machine learning algorithms to RDM, it is a computationally intensive process.

Ideally one expects a clustering algorithm to accurately partition a given dataset into appropriate subsets with minimal error irrespective of the permutations. Most clustering algorithms operate on the premise of a multi objective optimization problem. It is a well-known fact that most optimization problems can be solved in polynomial time, however in worst case scenarios, converging to the most optimal solution takes time and has no known polynomial solution. Essentially speaking, clustering for optimality can be treated as NP-Hard, as it's related or can be transformed into one of the known NP-Complete problems like 3-coloring, minimum edge coloring, minimum cut or knapsack [16, 20]. Hence clustering techniques on large data sets tend to be computationally expensive and resource intensive.

2. Ensemble Clustering as Reinforcement Learning Algorithm

A central focus of the algorithm considered is to account for individual differences among participants through reinforced learning. Application of ensemble clustering as a reinforcement learning algorithm has proven to be good fit for the IGT task [23]. It analyzes the dataset from multiple dimensions, which is achieved by creating and analyzing numerous partitions of the original data collected through empirical experiments.

The complete ensemble algorithm can be broadly classified into three separate parts. The first step is to preprocess data and generate data bootstraps. The second step is to create the clustering ensemble and the latter phase is used to extract relevant information giving insight into participant's performance on the task. The process of creating bootstrap boot samples is given in

Algorithm 2. Bootstrap resamples are created after removing outliers. Constructing co-association graph COA on Π can be summarized as shown in Algorithm 3.

A. Removing outliers from Data

Removing outliers is important because of the bootstrapping step in the algorithm. It involves computing inter-object distances for (X, X) using either Euclidean or Mahalanobis distance measure.

$$euclidean_distance\langle X_i, X_j \rangle = \sqrt{\sum_k^F (X_{ik} - X_{jk})^2} \dots\dots\dots (1)$$

$$mahalanobis_distance\langle X \rangle = \sqrt{\sum_k^F (X_{ik} - \mu_k) S^{-1} (X_{ik} - \mu_k)} \dots\dots\dots (2)$$

Any data object whose computed z-score (based on either Euclidean or Mahalanobis distance) is greater than or equal to cut off is removed from further analysis. Z-score can be computed as

$$zscore = \frac{X - \mu}{\sigma} \dots\dots\dots (3)$$

B. IID Bootstrap Resampling

For 1:n*Bootstraps*

- a. Simulate N *i.i.d* in uniform distribution for generating integers $u_i \mid 1 \leq i \leq N$ (with replacement)
- b. Construct a bootstrap resample
 - i. $X_b^* = \{X_{u_1}, X_{u_2}, \dots, X_{u_N}\}$
- c. Compute the mean for X_b^* (*optional*)
- d. Estimate the standard error for X_b^* (*optional*)

End

Algorithm 1. Method to resample data using i.i.d jackknife method.

Given input data is resampled using bootstrapping. Bootstrap resampling is currently done as given in Algorithm 1. We assume an underlying statistical distribution that explains the statistical properties of the observations do not change. As the algorithm shows data resampling is done using I.I.D Jackknife approach. This processing done during the preprocessing step of Algorithm 2. Jackknife method was originally discussed by Efron [28] and subsequently been researched extensively. The algorithm assumes that data are independent and identically distributed (IID).

A. Create Clustering Ensemble

1. Preprocess data
2. Determine statistical variation
3. Remove outliers
4. Create Bootstrap resampling
5. Reorganize models
6. Create Ensemble:
7. For($nModels;\mu$)
 - a. calcDistance()
 - b. runModel()
8. end
9. Extract Solution
10. Construct co-association graph *COA*
11. For($nPartitions;K_P$)
12. Partition *COA* using spectral clustering
13. Convert spectral indices to data object centers
14. Count object assignments to labels
15. Compute SSE
16. end

Algorithm 2. Ensemble clustering algorithm using co-association consensus function.

In the first phase of reinforcement learning algorithm for RDM we partition original experimental data into multiple partitions through variation in dimensions across a number of data re-samples (bootstraps, see Algorithm 1). Ensemble creation step also involves running different clustering algorithm on the bootstrap dataset to get cluster label assignments. clustering algorithms

that are reinitialized multiples times to avoid local optima, the number of clusters and various distance measures. By doing this one can enhance the statistical significance of the original data and also increase the probability of discovering weak correlations. These label assignments, in turn, become input to ensemble solution extraction step (step 9, in Algorithm 2). The process of extracting solution i.e. the number of individual groups supported by data is given in Algorithm 3. At a broad level, ensemble creation step encompasses sub process that is responsible for generating model variations. Another major parameter is different inter-object distances from which we drive models. After constructing object similarities (X, X) matrix, different base clustering algorithms are applied with variation in configurations.

1. Find co-association counts for all partitions.
2. Construct co-association graph:
3. For 1: K_p Partitions
 - a. Construct co-associations for partition k_p
 - b. Accumulate counts across partition k_p
4. End
5. Find edge weights for the graph

Algorithm 3. Constructing co-association graph on cluster labels for ensemble models.

Clustering result from each individual run is stored temporarily along with its dimensions (meta-data). Accumulated results from all the clustering runs across dimensions are passed to the third phase (see Algorithm 3).

B. Extract Result from Ensemble

Final solution is extracted by analyzing a matrix produced by a consensus function; and is called consensus matrix. It is derived from the results computed during first phase. Consensus function's algorithmic logic is based on the counting principle. For the sake of brevity, we do not

expand on various ideas for deriving consensus matrix except that in simplest of terms, it essentially operates by counting the number of times a participant was assigned to the same cluster across all the partitions in the ensemble.

Table 5 Base clustering algorithms in the ensemble and their time complexities.

Clustering Algorithm	Time Complexity (one distance measure)
K-means	$O(NKF)$
K-medians	$O(NKF)$
Agglomerative	$O(N^2 \log N)$
GMM	$O(NKF)$
Spectral	$O(N^3)$

The sequential time complexity of ensemble algorithm can be seen as follows. Running each model in M and collecting results will depend on the base clustering algorithm's time complexity (see Table 5). Algorithm 3 takes $O(N^2)$ -time. Converting spectral indices to centers can be done in $O(K_p NF)$ time. Finding the sum of square involves creating scatter matrix on the input data objects and has a sequential time complexity of $O(N^3)$.

It can thus be easily seen that for a reasonable size of input data, these could lead to long computation times, and can be measured in terms of days.

C. Ensembles configuration

Ensemble configuration has arguments that determine a total number of partitions involved in the algorithm. Ensemble clustering complexity is directly proportional to these partitions. As mentioned in Table 6, one will need to specify a list of values for K with parameter values as `kList`, base clustering procedures in the format of `gmm`, `kmeansxxx`, `medoidxxx`, `spectralxxx` and

aggxxxxyy where xxx denotes distance metric and yyy denotes linkage metric. Arguments nBootstraps and nReps control a number of bootstrap samples and replications respectively.

Table 6 PST structure to configure ensemble clustering.

Parameter Name	Values
outlier cutoff	Can be varied and typically specified by researcher.
outlier metric	mahal or euc
ensemble.kList	Eg. [1:15,20:5:60]
ensemble.modelList	Eg. 'kmeanseuc','kmeanscit','kmeanscor','kmeanscos', 'spectraleuc','spectralcit','spectralcor','spectralcos', 'aggeucwar','aggeucavg','aggeucom','aggcitavg','aggcitcom','ag gcoravg','aggcorcom','aggcosavg','aggcoscom'
ensemble.nBootstraps	Can be varied and typically specified by researcher.
ensemble.nReps	Can be varied and typically specified by researcher.
extract.kList	extract solutions at these k
extract.consensusMethod	'coassoc' or 'vote'
extract.extractionMethods	spectral or agglom

D. Solution extraction configurations

Configuration parameters for extracting solution from partitions include a list of K_p (kList) for many partitioning of the COA graph. Consensus extraction method (consensusMethod) – current implementation only supports co-association matrix. However, other methods can be incorporated as easily. Extraction method (extractionMethods) can be either spectral or agglomerative.

3. Model Formulation

Table 7 Ensemble clustering model variation parameters.

Data objects X	$\{x_1, x_2, \dots, x_N\}$
Data bootstrap B	$\{b_1, b_2, \dots, b_\beta\}$
Base cluster Algorithms C	$\{c_1, c_2, \dots, c_\gamma\}$
Inter-object distance D	$\{d_1, d_2, \dots, d_\delta\}$
Number of clusters	$1 \leq k \leq K$
Given number of partitions for COA hyper-graph.	$1 \leq p \leq K_p$

Mathematically ensemble clustering can be summarized based on different models (M). Models can be derived from variations on parameters shown in Table 6.

Each data object has F features associated with it. Models derived from the above parameters can be denoted as $M = \{m_1, m_2, \dots, m_\mu\}$. Output from running these models gives the different labeling (Π) for data objects in X. Clustering labels in $\Pi = \{\pi_1, \pi_2, \dots, \pi_\mu\}$ are extracted from running individual models in $M = \{m_1, m_2, \dots, m_\mu\}$. Inner mathematical workings of each model are specific to clustering algorithms and detailed discussion of such clustering algorithms is out of the scope for this dissertation. A reader is referred to one of the many survey articles already published on various clustering algorithms such as [9, 31, 107].

After running models in M, the solution is extracted from Π , a $\mu * N$ matrix whose (i. j)th cell value is the clustering label assigned by model i to object j, for $1 \leq i \leq \mu$ and $1 \leq j \leq N$. A co-association hypergraph representation COA (V, E) is constructed from objects labeling (Π) that co-occurred in the clusters. Where,

V- Vertices are objects and its associated model in M.

E – Edges connect these vertices with weights equal to a percentage of co-occurrence.

More formally, graph COA can be constructed by applying the following operations: Let the co-occurrence of objects i and j by model k be defined as

$$COO_k(i, j) = \begin{cases} 1 & \text{objects } i \text{ \& } j \text{ get same label by model } k \\ 0 & \text{otherwise} \end{cases} \dots\dots\dots (4)$$

for $1 \leq k \leq \mu$, and $1 \leq i \leq N$, $1 \leq j \leq N$. Then

$$COA(i, j) = \frac{\sum_k COO_k(i, j)}{|\Pi(*, j)|} \dots\dots\dots (5)$$

where $|\Pi(*, j)|$ indicates the number of entries in column j of Π , i.e., number of times object j is present in the μ models. This is like finding Jaccard index and can be expressed as.

$$COA(i, j) = \frac{\text{\#of models giving same label to objects } i \text{ \& } j}{\text{total \#models labeling objects } i \text{ \& } j} \dots\dots\dots (6)$$

A. COA - co-association and co-occurrence (COO) computation example

To illustrate construction of COA graph let's consider the example with $\mu = 5$ and $N = 5$.

For the purpose of giving an brief explanation of steps in the algorithm we are starting with a randomly generated Π matrix.

$$\Pi = \begin{bmatrix} 1 & 1 & 2 & 3 & 3 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 2 & 1 \\ 1 & 2 & 3 & 4 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

Following equation (4) we construct COO_k for each row of Π as shown below.

$$COO_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$COO_2 = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$COO_3 = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$COO_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$COO_5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Final process in calculating COA matrix is further divided into two steps. In the first step we perform summation over all the COO matrices. And in the next step and final step do a dot division on the resulting matrix with number rows in Π .

$$COO_1 + COO_2 + COO_3 + COO_4 + COO_5 = \begin{bmatrix} 0 & 2 & 2 & 1 & 3 \\ 2 & 0 & 1 & 2 & 1 \\ 2 & 1 & 0 & 1 & 2 \\ 1 & 2 & 1 & 1 & 1 \\ 3 & 1 & 2 & 2 & 0 \end{bmatrix}$$

$$COA = \begin{bmatrix} 0 & 2 & 2 & 1 & 3 \\ \frac{0}{5} & \frac{2}{5} & \frac{2}{5} & \frac{1}{5} & \frac{3}{5} \\ 2 & 0 & 1 & 2 & 1 \\ \frac{2}{5} & \frac{0}{5} & \frac{1}{5} & \frac{2}{5} & \frac{1}{5} \\ 2 & 1 & 0 & 1 & 2 \\ \frac{2}{5} & \frac{1}{5} & \frac{0}{5} & \frac{1}{5} & \frac{2}{5} \\ 1 & 2 & 1 & 1 & 1 \\ \frac{1}{5} & \frac{2}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ 3 & 1 & 2 & 2 & 0 \\ \frac{3}{5} & \frac{1}{5} & \frac{2}{5} & \frac{2}{5} & \frac{0}{5} \end{bmatrix} \text{ or } \begin{bmatrix} 0 & 0.4 & 0.4 & 0.2 & 0.6 \\ 0.4 & 0 & 0.2 & 0.4 & 0.2 \\ 0.4 & 0.2 & 0 & 0.2 & 0.4 \\ 0.2 & 0.4 & 0.2 & 0.2 & 0.2 \\ 0.6 & 0.2 & 0.4 & 0.4 & 0 \end{bmatrix}$$

Another important metric that is of interest is the sum of squared error (SSE) with respect to each partitioning p of COA ($1 \leq p \leq K_p$). SSE is used to validate the number of clusters supported by the data. p^{th} partition is obtained by performing a spectral cut over COA. SSE can then be found by first centering data objects in the p^{th} partition of graph COA. Let n' be the total number of objects in a sub-graph of COA induced by partition p and x_{ij} be the value of the j th feature of object i in this sub-graph. Then $x_{ij} - \bar{x}_j$ is the normalization (or centering) of x_{ij} for $1 \leq i \leq n'$ and $1 \leq j \leq F$ which gives us an $n' * F$ matrix X_C , where \bar{x}_j is the average feature over the objects in the same partition of COA, and calculated as

$$\bar{x}_j = \frac{1}{n'} \sum_i x_{ij} \dots\dots\dots (7)$$

We find SSE for each partition by adding diagonal elements of the scatter matrix (SM) over X_C . Which is

$$SM = X_C^T * X_C \dots\dots\dots (8)$$

$$SSE = \sum_{f=1}^F SM_{ff} \dots\dots\dots (9)$$

A more descriptive explanation can be found in [46]. Work published by Singh et al. in [88] is a good place to start on the mathematical proofs behind ensemble clustering algorithm. Even though this technique has numerous merits, it faces challenges while cross-validation and recent work by Brusco and Steinley highlight the same [15].

4. Summary

Ensemble clustering (which Newman has shown to be a better than previously proposed research methodologies and promising approach for RDM model analyses) applies a number of clustering techniques thereby consumes even more computational resources, which certainly limits even further the size of RDM datasets that can be analyzed using existing algorithms on conventional hardware. For example, in terms of program execution time it took more than 3457 minutes (about 57 hours) for 1000 participants on an Intel Xeon processor, 8 GB RAM, 120 GB hard drive workstation running Ubuntu 12.04 – a typical workstation on a researcher’s desk. Therefore, in order to obtain a high performance technique for RDM, we first propose parallelizing this step. Currently proposed ensemble consists of Agglomerative, Kmeans, Kmedoids, GMM, and Spectral clustering algorithms at its core. Thus, we discuss parallelization of ensemble clustering technique for RDM on distributed and shared memory systems in the next chapters.

CHAPTER V

DISTRIBUTED MEMORY PARALLELIZATION FOR THE REINFORCED LEARNING ALGORITHM TO FIND INDIVIDUAL DIFFERENCES

1. Introduction

Researchers have employed different approaches to improving the computational performance of individual clustering algorithms. A number of parallel implementations of widely used clustering algorithms (such as kmeans, gmm, kmedoids etc.) can be found in [5, 7]. However very few have looked into the parallelization of ensemble clusters as a whole. Ensemble clustering algorithm composes of well-known and extensively researched machine learning techniques that are executed over multiple dimensions.

Ensemble clustering computational cost can be easily derived from the constituent clustering algorithms and the cost of the consensus function. An ensemble of clustering algorithm results in a solution that uses different formats to represent clusters. Aggregating these to form a final result can be difficult and falls in the category of solving correspondence problem [28]. This is because ensemble clustering algorithm is a multi-objective algorithm working on different number of data dimensions and is classified as an NP-Hard problem for variable number of data partitions. However, in our case we are using a simpler version of a consensus function, whose cost primarily depends on the number of dimensions and partitions computed. Hence, for this reason we assume cost of computing a consensus function will be negligible in comparison to computing the ensemble. Also, distance calculation is another subtle aspect. Although computation

cost varies across different distance measures, execution times are within tolerable range of each other, for consideration towards further optimization.

In general, we evaluated parallelization of ensemble clustering algorithm from 3 different perspectives. One could Parallelize individual clustering algorithms (approach 1); Concurrently execute ensemble clustering across the number of data partitions resulting from variations in parameters forming dimensions (approach 2); or Use a hybrid approach resulting from doing both (approach 3) i.e. individual clustering algorithms are parallelized and operate on different data partition concurrently.

Table 8. Notations, used throughout the chapter.

n	number of data points
d	dimensions or features
t	number of nearest neighbors
m	number of edges in the graph or Arnoldi length in using an eigen solver
h	number iterations required to converge or -#restarted Arnoldi in ARPACK [18]
k	Number of desired clusters
p, q	Number of MPI threads used for parallelization
Δ_S	Number of models based on spectral clustering
Δ_K	Number of models based on k-means plus k-medoids clustering
Δ_A	Number of models based on agglomerative clustering

Table 9 Time complexity for different parallelization approaches.

Sequential
$O\left(\Delta_s(n^2d + n^2 \log t + (m^3 + (nm + nt) * (m - k)) * h + (nk^2) * \#k \text{ means})\right) \\ + O(\Delta_k(nkd)) + O(\Delta_A(n^2 \log n))$
Parallelize across models
$O\left(\frac{\Delta_s}{p}\left(n^2d + n^2 \log t + (O(m^3) + (O(nm) + O(nt)) * O(m - k)) * h - O(nk^2) \right. \right. \\ \left. \left. * (\#k \text{ means})\right)\right) + O\left(\frac{\Delta_k}{p}(nkd)\right) + O\left(\frac{\Delta_A}{p}(n^2 \log n)\right)$
Parallelize Individual Clustering Algorithm
$O\left(\Delta_s(n^2d/q + n^2 \log t/q + (m^3 + (nm/q + nt/q) * (m - k)) * h + (nk^2/q) * \#k \text{ means})\right) \\ + O(\Delta_k(nkd/q)) + O(\Delta_A(n * n/q \log n))$
Hybrid Parallelization
$O\left(\Delta_s/p(n^2d/q + n^2 \log t/q + (m^3 + (nm/q + nt/q) * (m - k)) * h + (nk^2/q) \right. \\ \left. * \#k \text{ means})\right) + O(\Delta_k/p(nkd/q)) + O(\Delta_A/p(n * n/q \log n))$

Analysis on the time complexity for each of these approaches can be found in Table 9. Each approach has its strengths and weaknesses depending on the data size and resulting overhead from communication. In our case, individual data partition in the ensemble is fairly small compared to the number of models, hence we chose to parallelize using second approach and listing in Algorithm 4 corresponds to it. But if the size of individual partition is large then analyzing it across various models might not be feasible on low compute capable machines. And opting for distributed memory architecture might be inevitable hence one will have to use approach 3 (i.e., the hybrid approach).

In our approach to diffuse the computation load across worker threads, each thread needs to know configuration parameters of the model being executed. This meta data is maintained in a FIFO queue; steps 1 through 9 in the Algorithm 4 shows populating this queue with correct parameter settings. Each thread fetches a model to execute after successful completion of previous model execution. Making this approach dynamic in nature. Note in Algorithm 4 different clustering algorithms are run variation on the number of clusters (K). Bootstrap data sample is used to run clustering algorithms. As there are number of these bootstrap data in the model's configuration each clustering algorithm is run over same bootstrapped data resample with other variations. Chapter IV give details on these variations in the algorithm.


```

1. Load data
2. Compose cluster parameters
3. // Make preparations to run ensemble in accordance to (ii) i.e
4. // Create FIFO data structure to record Models
5. FIFO_Q_MODELS = EMPTY; //
6. for m = 1 : nClustMthd
7.   for k=1 : kList
8.     for b = 1 : nBootstraps
9.       a. for r = 1 : loopReplicates
10.        b. switch(clustMthd)
11.         c. case KMEANS:
12.           i. insert_into(FIFO_Q_MODELS);
13.         d. case KMEDIAN:
14.           i. insert_into(FIFO_Q_MODELS);
15.         e. case AGGLOMETATIVE:
16.           i. insert_into(FIFO_Q_MODELS);
17.         f. case SPECTRAL:
18.           i. insert_into(FIFO_Q_MODELS);
19.     EN_RESULT = NULL;
20. // Create Ensemble: cluster_ensemble_create() – Start processing in Parallel
21. model_parameters = get_frm_queue(FIFO_Q_MODELS, PID);
22. model = get(model_parameters)
23. temp_result = null;
24. switch(model)
25. case KMEANS:
26.   a. temp_result =doKmeans();
27. case KMEDIAN:
28.   a. temp_result =doKmedian();
29. case AGGLOMETATIVE:
30.   a. temp_result =doAgglo();
31. case SPECTRAL:
32.   a. temp_result =doSpectral();
33. Append temp_result to EN_RESULT
34. // END Parallel code
35. // Extract Result: cluster_ensemble_extract() – Sequential Code
36. for m= 1 : nMethod
37. C = get_partition_centers(EN_RESULT);
38. D = distance_partition_centers(EN_RESULT);
39. for k =1 : kList
40. get cluster indices for centers using spectral clustering
41. for n =1 : nSamples
42.   a. find closest centers to each sample in EN_RESULT

```

Algorithm 4. Parallel ensemble clustering algorithm used for analyzing RDM data.
In the above ALGORITHM 2, sections highlighted in gray forms the code for worker threads.

2. Experimental Results

As mentioned earlier, we analyzed performance of a multi-objective machine learning algorithm namely ensemble clustering to analyze data collected through IGT for RDM. All experiments were run on M40 nodes of the penguin computing pod cluster [6] which has QDR Infiniband interconnect, 10GigE data network, and minimum 4GB RAM per core. Sequential execution times over different data set sizes (200, 400, 600, 800 and 1000 participants) were recorded to identify bottlenecks in the performance. Data size from a RDM experiment study was varied for different number of participants while the number of data partitions that are analyzed in the ensemble is kept constant at 20400 to resemble Newman approach [23]. Recall that, our primary goal for this study was to first identify performance bottlenecks and then design a HPC solution to analyzing RDM.

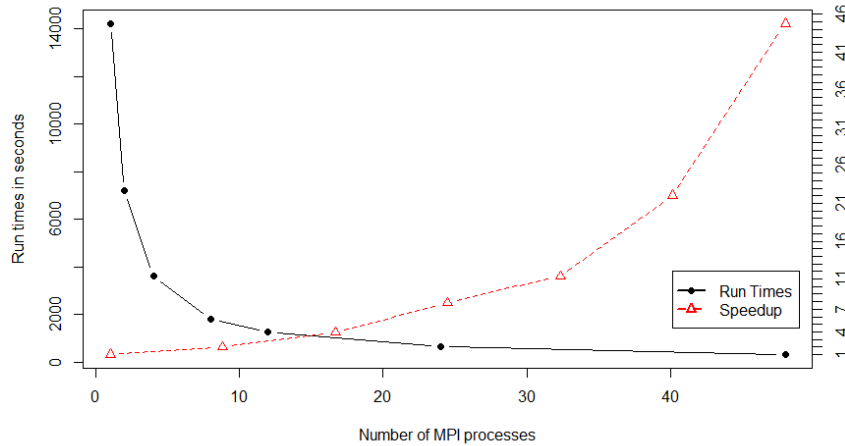


Figure 16. Execution times and speedup for parallelized RDM reinforcement learning algorithm using ensemble clustering for 1000 participants on a penguin computing POD cluster.

Ensemble Configuration parameter used during the experiments for measure execution times on distributed execution environment is as given in the Table 10.

Table 10 Ensemble clustering configuration parameters used for distributed memory parallelization experiments.

Parameter Name	Values
outlier cutoff	3
outlier metric	mahal
ensemble.kList	1, 2, 3, 4, 5, 6,7,8,9,10,11,12,13,14,15,20,25,30,35,40,45,50,55,60
ensemble.modelList	kmeanseuc,kmeanscit,kmeanscor,kmeanscos,spectraleuc,spectralcit,spectralcor,spectralcos,aggeucwar,aggeucavg,aggeucocom,aggcitavg,aggcitcom,aggcoravg,aggcorcom,aggcosavg,aggcoscom
ensemble.nBootstraps	50
ensemble.nReps	20
extract.kList	1,2,3,4,5,6,7,8,9,10
extract.consensusMethod	'coassoc'
extract.extractionMethods	spectral

Initial analysis of algorithm's experimental results revealed that for 1000 participants on a single machine it took over 3457 minutes in total (about 57 hours or over 2 days), where the machine was Intel Xeon processor, 8 GB RAM, 120 GB hard drive running Ubuntu 12.04. Individually k-means, agglomerative and spectral clustering algorithms took 75, 5.5, and 3376 minutes, respectively. This trend follows across different data set sizes. To improve the overall

computational performance of the algorithm, we ran different models of ensemble concurrently. Figure 16 is an example plot for run times taken for experimental results where data size is 1000 participants on a penguin computing pod cluster.

Table 11 Run times, speedup and efficiency of ensemble clustering after parallelization for 1000 participants' data on penguin cluster.

#MPI threads	1	2	4	8	12	24	48
Exec Time (in sec)	14237	7198	3603	1808	1251	645	317
Speedup	1	1.97	3.95	7.87	11.37	22.04	44.79
Efficiency	1	0.98	0.98	0.98	0.94	0.91	0.93

Above Table 11 lists corresponding runtimes. Runtime using one MPI process was 14327 seconds which decreased to 1251 seconds using 12 MPI threads – an almost linear speedup. Similar trend can be seen in the plot in Figure 16, which shows a direct linear proportionality between runtimes and the number of threads, i.e., runtimes decreases proportional to the number of threads. We chose this parallelization approach because of the nature of problem. In the current example our initial data set is small, i.e., only 1000 participants data. However, algorithm analyzes it from several dimensions by partitioning. If one supposes to use approach 1 for parallelization, speedup achieved would be offset by the communication overhead between different MPI threads, resulting in almost negligible performance improvement. This suggests that shared memory implementations should also be explored for ensemble clustering in addition to distributed memory approaches.

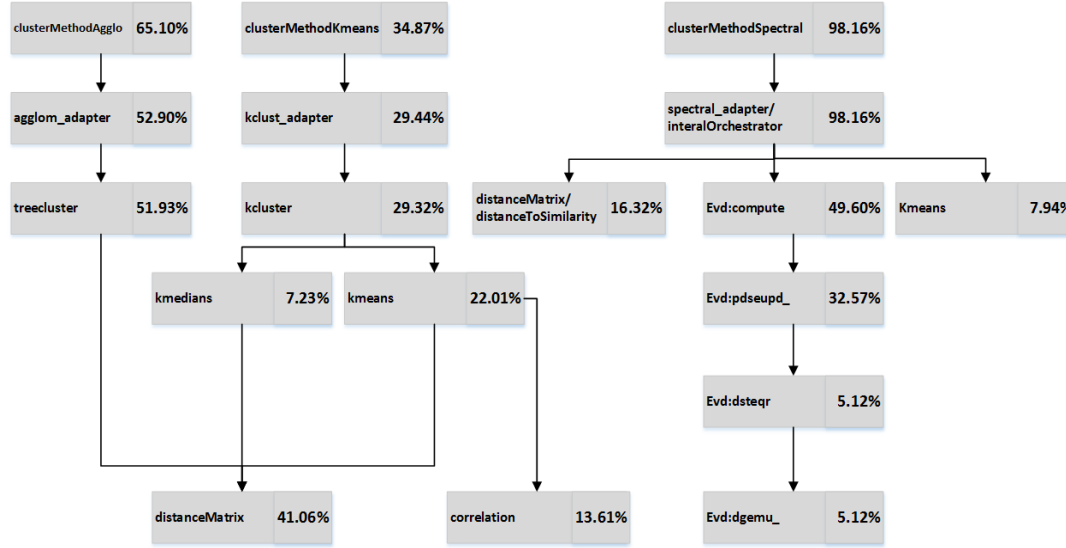


Figure 17. Call graph of agglomerative, kmeans, kmedians and spectral clustering with relative percentage computation time for each step.

Execution time's perspective alone is not enough. We also analyzed individual clustering algorithm's call hierarchy, and memory requirements along with execution times. Figure 17 shows the call graph for agglomerative, kmeans, kmedians, and spectral clustering algorithms. Each node in the figure also shows the relative percentage of time spent in the step. Analyzing the time spent in each step shows that agglomerative clustering costs almost 2x that of kmeans. And each clustering algorithm spends significant amount of time in calculating distance, among which cosine seems to be the most expensive at 27%. Hence, one can strategize techniques to generate bootstraps (resample data sets) such that one can minimize repetitive computation over same or statistically similar datasets without compromising output from the consensus function.

Analysis of memory requirements for each algorithm was also collected through experiments. Agglomerative, kmeans, kmedians and spectral clustering was run across all the variants in dimensions. All the algorithms used just one thread to get accurate memory footprint. Our experimental results reveal variation in cluster number (k) did not have any impact on the memory requirement for any of the considered clustering algorithms. However, we see that memory requirement gradually increases with time (iteration) for kmeans and agglomerative clustering across dimensions. This is an interesting finding as theoretical approximation do not give any indication for such behavior. System and program implementation can be considered probable cause for such symptoms in memory footprint; however, it would be worthwhile to investigate further as a separate research effort.

This behavior has little impact when data size is small but will be clearly evident when large data sets are considered, resulting in significant implications on the scalability of RDM techniques wherein experiments are envisioned to be designed on a global scale for a wide range of studies.

The code and relevant datasets for relevant datasets for reproduce results have been kept in the repository

https://github.com/vinaybabug/distributed_impl_reinforced_learning_algorithm_rdm.git

3. Summary

Analyzing datasets for Risky Decision Making (RDM) is a challenging task involving the identification of varied decision making patterns and the categorization of individuals. Researchers from various fields as diverse as psychology and marketing are actively working to identify

suitable techniques, which will allow understanding decision making processes better. Researchers have commonly used machine learning algorithms to model decision making processes. However, the high computational costs of most machine learning algorithms make such endeavors challenging for increasingly large datasets. One of the most promising approaches is to use ensemble clustering for RDM analysis. Ensemble clustering is computationally intensive and thus we propose to improve its performance. Our study reveals that computational overhead is introduced through the use of dimensions in ensemble cluster RDM analyses. Improving performance requires more than the parallelization of individual clustering techniques of the ensemble. We therefore propose a FIFO queue based implementation for analyzing RDM datasets using a HPC cluster on a distributed system. Our technique is able to achieve almost a linear speedup (e.g. 44.79x using 48 MPI threads). Possible shortcomings of the proposed method, opportunities for future work, and alternative parallelization scenarios are also discussed later in this dissertation.

CHAPTER VI

SHARED MEMORY PARALLELIZATION FOR THE REINFORCED LEARNING ALGORITHM TO FIND INDIVIDUAL DIFFERENCES

1. Introduction

Ensemble Clustering has been cited approximately two-hundred times per year since its original announcement in 2003 and has a total of 2900+ citations. Scholarly works are referring to Strehl and Ghosh [91], in turn, has been listed as reference 52000 times, indicating the level of influence this technique has during data analysis. Computer scientists, statisticians, mathematicians, and researchers from many other fields are developing analytical & machine learning models, based on ensemble clustering. The work in references [19, 71, 79] show some of the examples from the decision-making analysis domain. Work in [3, 12] uses ensemble clustering for medical applications such as magnetic resonance images, cancerous cells identification using FTIR spectroscopy, medical diagnostics and DNA data, [106] uses it for handwriting recognition, and [10] for application identification based on network traffic, etc. Although our discussion is focused more towards the RDM domain, in particular, the algorithm proposed by Newman in [71] which allows the identification of individual differences among participants making decisions under uncertainty, performance improvements in ensemble clustering is applicable to a wide variety of other problems and disciplines.

Currently, scientists studying decision making involving risk (Risky Decision Making, RDM), work with a relatively small sample of experimental data and use ensemble clustering

techniques. Data for studying RDM is collected during empirical studies. However, using an ensemble clustering algorithm on a typical off-the-shelf desktop machine entails unacceptably long computation times. This is a typical case for researchers with non-computational backgrounds who may not be able to justify investing in high-performance computing (HPC) machines. Our previous work in Chapter VI improved execution times in a distributed memory system using MPI. In this chapter, we explore performance improvements on a shared-memory desktop computer using CPU & GPU parallelization.

2. CUDA, Libraries and Related Tools

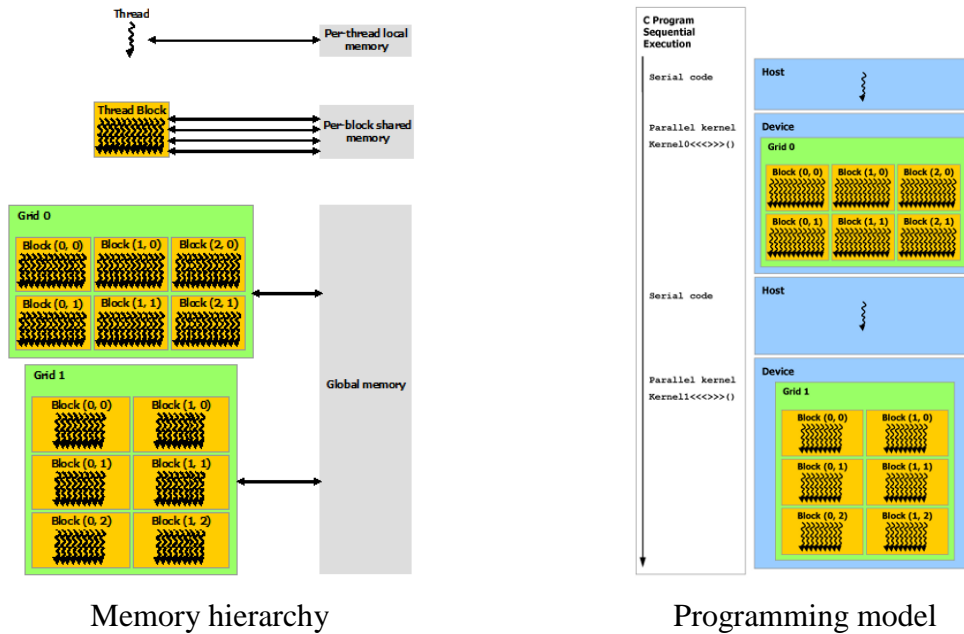


Figure 18. GPU memory hierarchy and CPU-GPU heterogeneous programming model [105].

Graphics processing hardware technology has made herculean improvements over the past decade. It is now being used for general purpose computation in many applications. It is best suited for algorithms designed for single instruction multiple data (SIMD) type of operations. Nvidia's graphics cards are commonplace and implement SIMD functionality using several threads running

on streaming multiprocessors [105]. It is commonly known as single instruction multiple thread (SIMT) model in the community. Threads on the GPU device can be assigned in a three-dimensional space. Our implementation tries to assign threads in these dimensions for maximum utilization of the device. The ratio of transistors in GPU's hardware is purposed for high arithmetic operations instead of data access and caching. This is different from CPU's as GPU device lacks built-in optimizations for memory operations compared to CPU. This burden is mitigated to the programmer. Because of which the performance of an application depends on efficient vectorization of data structures and its operations. As shown in Figure 18 programs written in CUDA are organized into three-level memory hierarchical model. It primarily consists of global memory, shared memory, thread local memory and texture memory. We have used texture memory to keep data objects that are most commonly accessed. Using shared memory as intermediary storage appropriately reduces memory latency as compared to accessing from global memory.

Not all parts of a program are suitable for efficient vectorization and execution on GPU. Hence typical programming model in CPU-GPU based technique is as shown in Figure 18. Our implementation also utilizes programmer productivity tools and libraries such as Thrust, and NSight. Thrust is based on C++ standard template library but for GPU devices. Below are some of the tools that support our implementation.

A. cuBLAS

It is a set of linear algebra routines optimized for GPU architecture and comes bundled with Nvidia's CUDA software development kit (SDK). It is also a part of a much wider array of

GPU-accelerated libraries² such as cuSPARSE, Thrust, etc. provided by Nvidia. cuBLAS is reported to be much faster compared to other CPU based BLAS libraries.

B. ARPACK, LAPACK, and OpenBLAS

ARPACK is a set of Fortran-based routines designed specifically for solving large scale eigenproblems [58]. It is designed to compute a eigenvalues efficiently. ARPACK provides reverse communication interface making it possible to integrate with third party technologies and tools such as CUDA.

C. MATLAB/OCTAVE

MATLAB and OCTAVE both are software used in scientific and engineering disciplines. MATLAB is a proprietary software whereas OCTAVE is its open source alternative. They are important in our research as application scientists use them often for their built-in clustering algorithms and their capability to work efficiently with matrices and plotting routines. Our implementation can be easily linked against these two software's libraries to provide the capability to read and write data understood in this software.

3. Implementation – An Overview

Our implementation of ensemble clustering is targeted to be generic to facilitate data analysis from different domains. The program accepts MATLAB structure called parameter structure (PST) with inputs to facilitate easy configuration of the models that will compose the ensemble (see Table 6, Chapter IV).

² <https://developer.nvidia.com/gpu-accelerated-libraries>

Models in ensemble clustering algorithm are composite of the certain base clustering algorithms. Running these models is at least equivalent if not more expensive than running individual clustering algorithms. To have a parallel implementation of the ensemble clustering algorithm, one will need to have parallelized individual clustering algorithms. We found many parallel implementations of the popular cluster algorithms such as kmeans, gmm and spectral clustering algorithm.

Our ensembles consist of GMM, k-means, k-medoids, agglomerative and spectral clustering. Our base clustering algorithms' CUDA implementation uses a modified version of algorithms implemented by Andrew D. Pangborn [74] for GMM, by Wei-keng Liao & Serban Giuroiu [99] for kmeans and by Yu Jin [49] for spectral clustering. CUDA implementation for kmedoid and agglomerative was done by us as part of this work. Some parts of the code have been based off Open Source Cluster Project: Cluster 3.0 by De Hoon et al. [21]. Again, for brevity, details on the individual clustering algorithm's CUDA implementation are left to those references. Several other subsequent articles have also been published which have in depth discussion on these topics.

Upon computing, cluster indices / labels for each model, cluster indices for bootstrap data need to be converted to cluster indices for original data. This is done using `cluster_util_bootpartition2partition()` method and takes $O(N)$ linear time.

E. Ensemble Solution Extraction Step

Extracting optimal solution from the ensemble turns out to be an expensive process, and previous literature has proved it to be NP-Hard. In this work, as mentioned earlier, we have only implemented acceptable-solution extraction through co-associations.

The second half of Algorithm 3 describes the solution extraction process at a higher level. The COA graph is constructed only once using the method in Algorithm 2, and method `cluster_ensemble2cam()` implements the algorithm. This method, in turn, uses `cluster_util_partition2cam()` which is used to track and identify partitions where a given object was assigned the same label and takes $O(N^2)$ time for sequential CPU execution. As `cluster_ensemble2cam()` is computed over all the partition the overall time complexity of constructing COA is $O(\mu N^2)$. As mentioned in Chapter V, μ is the total number of models in M or the number of partitions specified in the PST configuration $(\beta, \gamma, \delta, K, p)$.

Once COA graph is constructed, we extract solutions for a requested number of subgraphs of COA via K_p parameter list in PST. The overall computational cost of this phase is more than $O(N^3)$. It includes performing a spectral cut for all K_p listed in COA. Followed by reverse tracing of spectral labels to an original data object in methods `cluster_util_indices2centers()` and `cluster_util_ssw()`. It is used to calculate SSE for each K_p . We utilized `cublasSgeam` from `cuBLAS` for calculating SSE.

The output from the solution extraction phase is written back to files system as a MATLAB structure (.mat) file. Program's input and output interface are made to be through MATLAB or octave structure files because domain scientists tend to use MATLAB and our intention was to provide a seamless interface to them.

4. Experimental Results

The parallelized CPU-GPU version of the ensemble clustering algorithm presented is primarily being used for detecting the presence of different categorical groups in the given data. These datasets in our experiments were derived from the risky decision-making domain, where a

common practice is to collect data from participants via surveys and computer based paradigms. These paradigms are analogous to computer games and collect data about the person’s behavior (decision making). The solution from the algorithm is validated using several validations criteria (in our experiments we used eleven validation criteria). Mean, and median of error rate is among these and give the final summary. Summary of validation criteria gives an indication of the valid number of clusters supported by the solution. For a detailed description of the validation procedures, the reader is referred to the doctoral dissertation by Newman [71], as a discussion on model validation is extensive and out of the scope of this article.

As can be easily seen from our discussion so far, implementing ensemble clustering in a native language is an immense undertaking. In the following section, we only highlight some of the salient features. Table 14 and Figure 20 present and compare speedups gained by running shared memory based CUDA plus CPU (i.e., CPU-GPU) vs. CPU only implementations.

A. Datasets: Iowa Gambling Task / Paradigm

Iowa gambling task (IGT) is one of the popular and widely used computer based paradigm for RDM analysis. IGT is a card playing game, where the goal is to make more money by the game end. All participants start with the predefined initial amount. During the game, a participant repeatedly selects a card from one of the four decks. Decks reward can be positive, negative or both. The game simulates reinforcement learning by design [7]. In task the main dependent variable is the payout during each trail and the deck selected. Datasets for our experiments was collected using an open source toolkit called RDMTk.

Table 12 Computer specification used for evaluation.

CPU Model	Intel Xeon i7
CPU Cores	4
DRAM Size	64GB
GPU Model	Quadro K1200; Tesla K80
Device Memory Size	4GB GDDR5
SMs and SPs	4 and 128
Compute Capability	5.0
CUDA SDK	7.5
PCIe Bus interconnect	PCIe x 16 Gen2
OS	Ubuntu 14.04 LTS

The ensemble configuration parameter used during the experiments for measure execution times on shared memory execution environment is as given in the Table 13.

Table 13 Ensemble clustering configuration parameters used for shared memory parallelization experiments.

Parameter Name	Values
outlier cutoff	3
outlier metric	mahal
ensemble.kList	1, 2, 5
ensemble.modelList	kmeanseuc,kmeanscit,kmeanscor,kmeanscos,medoideuc,medoidcit,medoidcor,medoidcos,spectraleuc,spectralcit,spectralcor,spectralcos,aggeucwar,aggeucavg,aggeucum,aggcitavg,aggcitcom,aggcoravg,aggcorcom,aggcosavg,aggcoscom,gmm
ensemble.nBootstraps	5
ensemble.nReps	2
extract.kList	1,2,3,4,5,6,7,8,9,10
extract.consensusMethod	'coassoc'
extract.extractionMethods	spectral

B. Pairwise Distance Matrix Computation

Our implementation currently supports pairwise distance computation using Euclidean, squared Euclidean, City Block, Pearson, Weighted Pearson, Kendall, Cosine, Mahalanobis, Jaccard, Chebyshev, and Hamming distance measures. A distance matrix is symmetrical across the diagonal; one will only keep track of either the upper or the lower half triangle. As only unique distance values are stored, data storage results in a jagged array and is dynamic memory allocation intensive. Our CUDA implementation utilizes texture memory on the device to cache data. Computation on GPU using non-texture memory vs. texture memory is compared in Figure 21. Using texture memory on average is 1000 times faster compared to performance on global and shared memory based implementations.

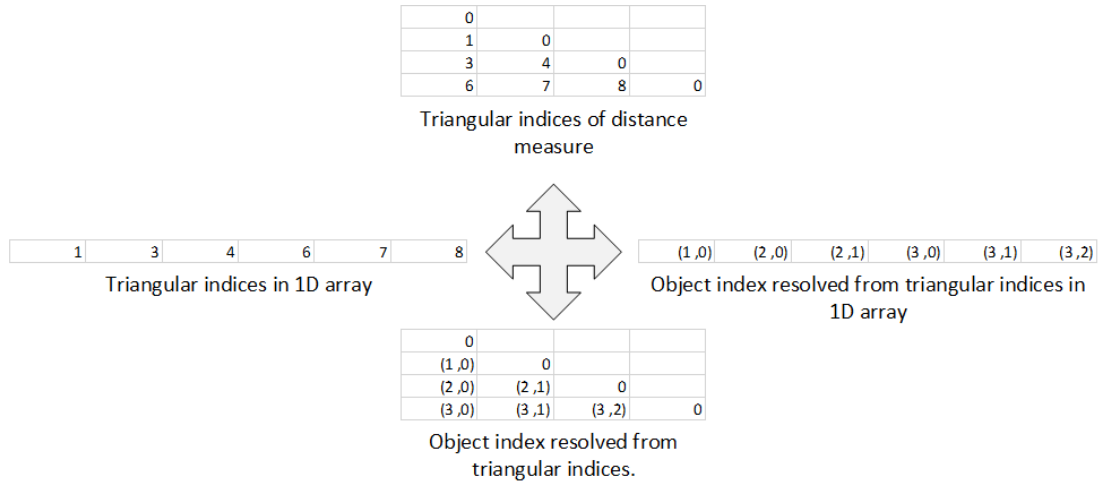


Figure 19. Example: CUDA thread assignment with jagged indices calculation.

This improvement is achieved in two folds by enhancing data locality [23]. We need an efficient data allocation to threads that support coalesced data access while calculating distance metrics. As we only need to compute jagged distances measures, storing this data in a 2D array

will result in increased paging problems internally on the CUDA device. Computing distance measure across different models (M) requires repeated access to data. Repeated allocation and deallocation of dynamic 2D memory is bad by design [104] for spatial and spatiotemporal data locality. Therefore, we use 1D arrays.

We also employed the following strategy to localize the data for the thread and memory hierarchy of the CUDA device. Using roots of the quadratic equation $ax^2+bx+c=0$ to allocate the two data objects on which the thread computes distance metric increases spatiotemporal locality. It localizes data access based on `tIdx`, ensuring fetch and store during calculation of all pairwise distance is uniformly distributed across different blocks.

$$object1 = floorf\left(\frac{(-1 + sqrtf(1 - (4 * 1 * (-tIdx * 2))))}{2}\right).....(10)$$

$$object2 = tIdx - \left(\frac{(row*(row + 1))}{2}\right).....(11)$$

Figure 19 gives an example of thread assignment with jagged indices calculation. Where `tIdx` is the thread id calculated by using `blockIdx.x`, `blockDim.x`, and `threadIdx.x`.

Table 14 Run times in seconds for executing ensemble clustering algorithm on CPU & CPU-GPU.

#Participants	Parallel BLAS-ARPACK				Sequential CUBLAS-ARPACK			
	Preprocessing	Ensemble Creation	Ensemble Extraction	Total	Preprocessing	Ensemble Creation	Ensemble Extraction	Total
512	0.000002	0.027031	0.200994	0.228027	0.008813	0.920119	0.596379	1.52531
1024	0.000004	0.065028	0.829694	0.894726	0.034078	3.890402	1.2575	5.18198
2048	0.000008	0.179218	1.74629185	1.9255178	0.133376	24.678988	3.91411	28.7264
4096	0.000025	0.353409	4.55996992	4.9134039	0.5259915	178.714399	10.220642	189.460
8192	0.000097	0.918421	16.4327205	17.351238	2.0748	1325.326938	36.832031	1364.23
10240	0.000197	1.4468	25.6589273	27.105924	3.240139	2550.207621	57.5115	2610.95
11264	0.000228	1.762412	31.1415162	32.904156	3.89515	3374.181781	69.8000851	3447.87
12288	0.000216	1.773678	36.1456167	37.919510	4.637481	4361.379918	81.016194	4447.03
13312	0.000254	8.05293	42.3747132	50.427897	5.440463	5515.57373	94.977989	5615.99
14336	0.000309	2.353111	48.4853123	50.838732	6.315224	6858.803793	108.674186	6973.79

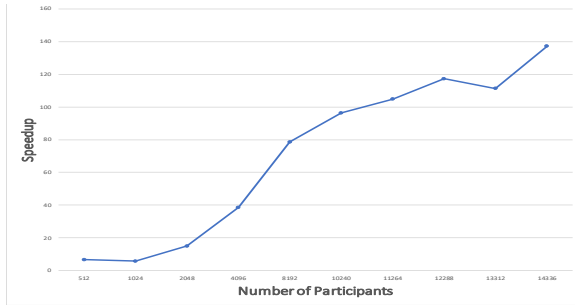


Figure 20. Speedup achieved for running ensemble clustering on CPU vs. CPU-GPU.

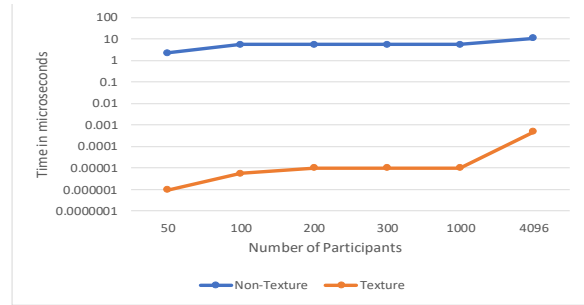


Figure 21. Comparison of running distance matrix kernels using texture vs. non-texture memory.

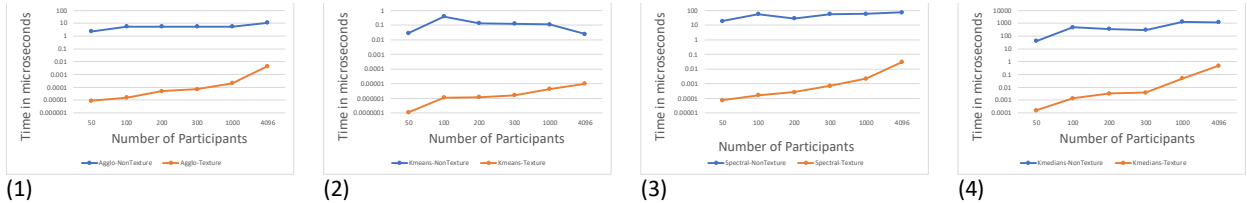


Figure 22. Comparison of running (1) agglomerative, (2) kmeans, (3) spectral, and (4) kmedians clustering kernels using texture vs. non-texture memory.

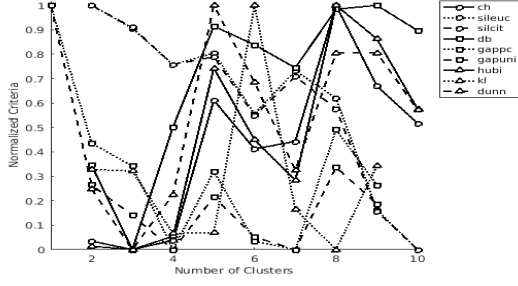


Figure 23. Normalized criteria value for the number of supported clusters for nine validity criteria.

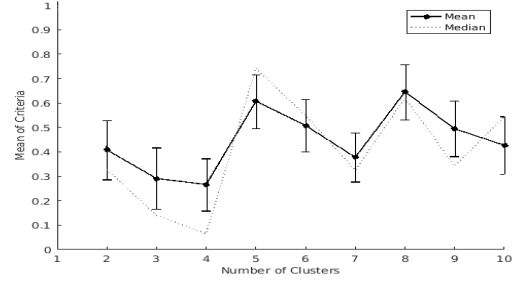


Figure 24. Validity criteria summary showing consensus for 5 or 8 clusters.

C. Memory Optimizations for the Clustering Algorithms

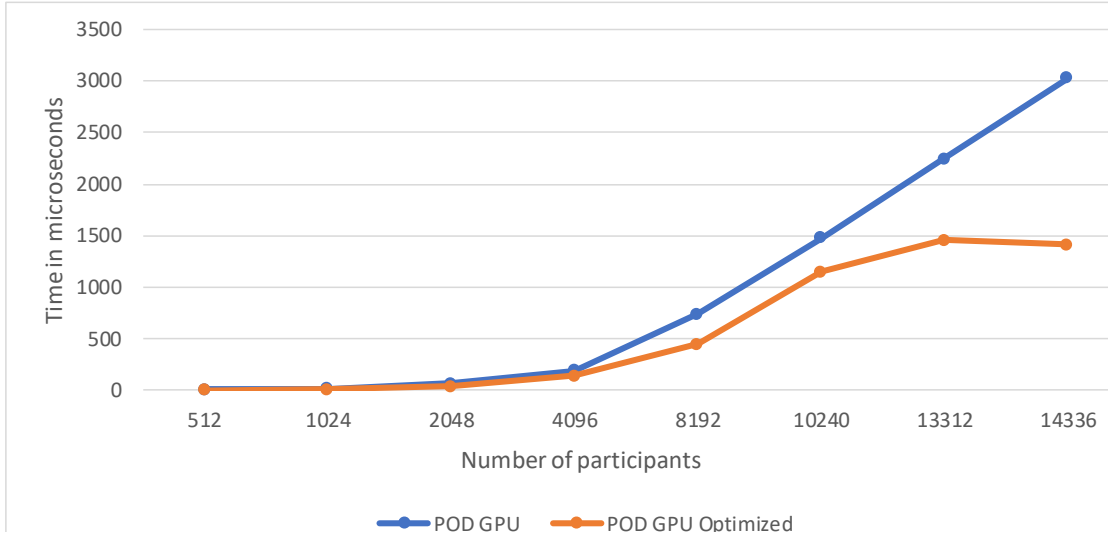


Figure 25. Run times for unordered vs. batch reconfigured model sequence execution.

Our implementation uses previously developed codes for individual clustering algorithms except for Kmedians and Agglomerative as mentioned earlier. Individual clustering algorithms, e.g., Kmeans, Kmedoids, and Spectral were also modified from original to use shared memory and texture memory. We compared the performance of original and modified versions. The distance

metric and number of clusters were kept constant while collecting execution time measurements. As seen in Figure 22 use of texture memory speeds up execution by order of 100 – 1000 times.

D. Unordered vs. Batch Reconfigured Model Execution Sequence

The complexity of ensemble clustering algorithm depends on parameter values for β , γ , δ , K and p . These parameters determine the limit on μ (Table 7) or the total number of model iterations. Calculation of distances for each model is redundant. Similarly, there are several calculations that are repeated over the order of the model's execution cycle. Such as in agglomerative clustering, tree construction for dendrogram is computed each time a model variation occurs. In spectral clustering, Eigen values are calculated from scratch for each model variation. These operations would have been repeated during each iteration of M in a naïve implementation. Also, setting up texture memory, launching CUDA kernels and data transfer between CPU and device are required during each iteration. These operations are expensive and are limited by the PCIe bus interconnect bandwidth.

Our implementation reduces some of these redundancies. This is achieved by reconfiguring and batching the model execution order. The configuration of models is similar in principle to software engineering 101, however, modified for HPC scenarios. Figure 25 shows that execution times for optimized version is 2-5x faster than unordered sequence for configuration parameter values of $\beta=5$, $\gamma=5$, $\delta=14$, $K = 60$ and $p=10$.

Researchers want to identify the number of clusters data supports, upon successful completion of the algorithm. Figure 23 and 24 show results for an RDM dataset with 1700+ participants having 22 features. Figure 23 shows normalized criteria values for different number of clusters for the 9 validation criteria (Calinski-Harabasz (ch) [16], Silhouette Euclidean (sileuc),

and Cityblock (silcit) [51], Davies-Bouldin (db) [20], Tibshirani's Gap criterion (gappc), and with uniform distribution (gapuni) [95], Improved Hubert Gamma (hubi) [110], Krzanowski-Lai (ki) [55] and Dunn (dunn) [27]), and the best supported clusters are those whose criteria value is the largest. For example, Calinski-Harabasz (ch) suggests a solution with 8 clusters (having a peak normalized criteria value nearly equal to 1; note that zero or no cluster solutions are ignored). Figure 24 plots mean and median validities of the clustering suggestions from individual validation criteria of Figure 23. We can now infer that the data supports 5 or 8 valid clusters from looking at the peaks of the mean criteria value in Figure 24.

The code and relevant datasets for relevant datasets for reproduce results have been kept in the repository

Sequential: <https://github.com/vinaybabug/EnsembleClusteringSequential.git>

Parallel: <https://github.com/vinaybabug/EnsembleClusteringParallelCUDA.git>

5. Summary

Ensemble clustering algorithm is frequently used in machine learning algorithms, but it is also one of the most computationally intensive components thereby limiting their scalability. Parallel implementations of these algorithms enable researchers to pose bigger questions using larger datasets. The combinatorial and multi-objective nature of ensemble clustering algorithm makes running a large number of ensemble models a time-consuming task. In this chapter, we present a CPU-GPU based implementation of ensemble clustering algorithm. Ensemble consists of agglomerative, gmm, kmeans, kmedians, and spectral clustering algorithms. Our

implementation shows a 130x speedup over CPU only implementation. Primary usage and datasets used in our research are derived from risky decision making (RDM) domain.

CHAPTER VII

DISCRETIZATION AND SCHEDULING MODEL EXECUTION OF RDM RL ALGORITHM STEPS AS FLOOR TILES PLANNING

1. Introduction

It is becoming increasingly evident that current processor/hardware technologies are beginning to fail at satisfying Moore's Law. Researcher and practitioners are now considering new possibilities. New construction materials that reduce heat dissipation in circuits, computing technologies based on quantum computing, neural networks, and other advances made towards finding a suitable replacement for general purpose computing. There is another paradigm, as with IBM's Blue Gene, in which computers built specific to applications or processing of particular kinds of data or algorithms.

We firmly believe that the software aspect of computing begins to pull its weight by pushing the limits of scalability. In these efforts, we propose a conceptual framework around algorithms belonging to a particular class, i.e. combinatorial, bootstrap based algorithms similar to reinforced learning algorithm for RDM based on ensemble clustering discussed in previous chapters. One can reduce computational time by controlling execution sequence of different codes and considering their spatial and temporal data dependencies. We call this approach Floor Tiles Planning (FTiP), as it is analogous to fitting together tiles of different shapes. The same name was coined when engineers tried to fit scores of circuits on a VLSI design. The basic idea is to divide

the algorithm into small computing entities and assign a different amount of computing resources to it based on spatial and temporal dependencies in data.

The problem is a variation on the dynamic knapsack problem [32]. Consider the algorithm that is parallelized as an object that needs to be fit in a bag. Filling the portion of the bag is analogous to executing the algorithm's steps proportional to the bag filled. One of the methods of trying to fill the bag is to try and put all same size objects. Another method would be to use variable sized objects. We can easily see various optimization strategies to the problem. Likewise, analogous to what most parallelization efforts do, we instead propose to divide the parallelization effort by dividing it into units of various forms. These units can then be blown up to different sizes, representing some resources assigned to each model's consumption (refer to Chapter V, for explanation of model representation), the question then arises, can we fit all these models into one bag (i.e. can we execute them at the same time). If not, they will be executed in batches. If these models cannot fit into a single "bag," we will need multiple bags. If so, how many bags do we need? How big should each model be? How will this affect the total number of bags required to hold these models?

In our framework, we assume multiple kernels can be executed simultaneously. Each kernel represents running a model with different launch configurations. Models are derived from variations in different clustering algorithms (see Chapter IV, section on Model Formulation). A set of models executing on GPGPU execution simultaneously for a determinant period T is considered one floor.

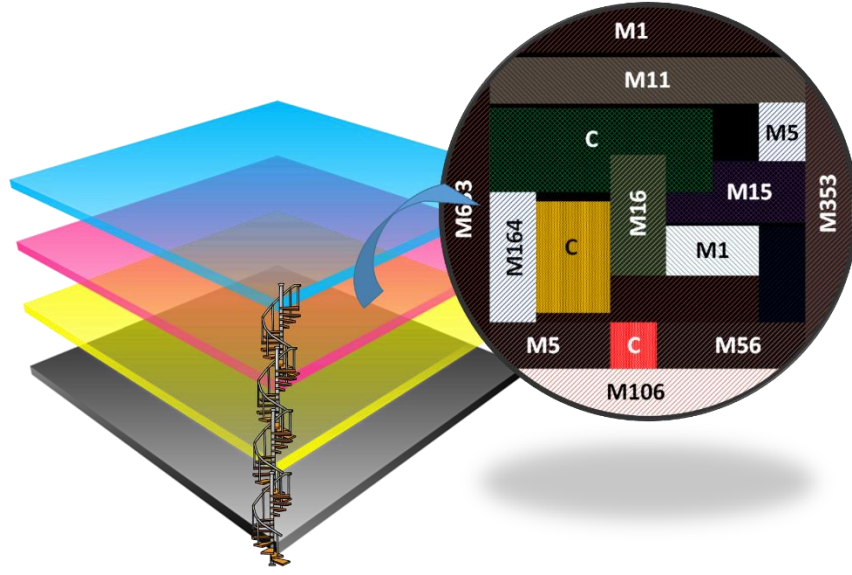


Figure 26. GPGPU threads and resources as floor tiles.

2. FTiP Problem Statement for Reinforced Learning Algorithm

Given the RDM reinforced learning algorithm; which can be represented by M models and its discretization into K sub models $M = \{m_1, m_2, m_3, \dots, m_k\}$ & N a list of dependencies, represented as $(n_1, n_2, n_3, \dots, n_k)$; where m_i has a corresponding dependency n_i . Find a floorplan F such that $\text{cost}(F)$ is minimized over period (time) T . Our goal is to have $\text{cost}(F)$ to be smaller than $T_{\text{Sequential}}$; where $T_{\text{Sequential}}$ is the time taken to execute all models and its sub-models m_i on the cuda device (SIMD) using T_{max} threads in a sequential manner i.e. one model after another [94, 90, 29].

Where,

A model m_i is launched as a kernel, which is a programmer-defined C function. When m_i is executed, it runs in parallel with a specified number of CUDA threads and other resources as dictated by the programmer.

The number of threads on the GPU device executing code in parallel is organized into,

- Thread groups which can be identified using a 1D, 2D or 3D indexes - $tx_i (n_x, n_y, n_z)$.
- Thread in 1D, 2D or 3D form thread block of the corresponding dimension- $bx_i(n_x, n_y, n_z)$
- Blocks are organized into 1D, 2D or 3D grid of thread blocks- $gx_i(n_x, n_y, n_z)$

The model m_i is a generic term used to represent a group of threads executing a kernel code. Also, an assignment of tx_i threads to model m_i can be in the form of a group of threads arranged in 1D, 2D or 3D within blocks and grids of size (Bx_i, Gx_i) . And $\{m_1, m_2, m_3, \dots, m_k\}$ is a set of models derived from the discretization of a bigger program M .

As mentioned earlier, we define dependencies N as a list $(n_1, n_2, n_3, \dots, n_k)$; each n_i in turn is made up of vectors. Dependency list size equals the number of models k . Each n_i is a vector of dependencies on other models for m_i .

Dependency for model m_i is based on the subset d_j of data D it operates on.

Data D is bootstrapped to subsets of the same size by resampling. Any two subset bootstraps d_j and d_k are related to each other by the number of participants (rows) they share in common because of being resampled data from the same base set.

Given two subsets d_j and d_k resampling D , we define π_{jk} the link strength (redundancy coefficient). Redundancy factor gives the relationship (link) strength between any two-data bootstrapped from same base superset D .

A similar relation is drawn for any two models m_j and m_k , where π_{jk} will represent the dependency based on the link strength between the constituent datasets of the models. Extending this definition further, we define matrix π

Table 15 Dependency list and link strength.

Π_{11}	Π_{12}	Π_{13}	Π_{14}
Π_{21}	Π_{22}	Π_{23}	Π_{24}
Π_{31}	Π_{32}	Π_{33}	Π_{34}
Π_{41}	Π_{42}	Π_{43}	Π_{44}

The above matrix in Table 15 is a sample of a dependency matrix for an instance of a problem with four models, where π_{jk} represents dependency degree and strength between model i and model j .

A floorplan (F) is an assignment/execution of models in M on a CUDA device plane over a period T such that no models overlap one another and are executed at most once while satisfying dependencies listed in N . For example, model m_6 may depend on $\{m_1, m_{25}, m_{31}\}$. Dependency imposes a constraint on the order of models execution. It could mean any of the following three

- The models that m_i depends on could/must be executed concurrently
- Those must complete before m_i
- Dependent models that must execute after m_i

A floorplan is described as a sequence of floors $\{f_1, f_2, f_3, \dots, f_m\} = F$ over corresponding period T divided as $\{t_1, t_2, t_3, \dots, t_p\}$.

The floorplan has a cost associated with it. $\text{Cost}(f_i)$ – is measured by the expense of the largest subset of models in M satisfying dependencies in N that can be scheduled at t_i .

The total cost of F is the sum of costs overall individual constituent floors f_i in the optimal solution or an approximation of the optimal solution.

A. Variables affecting the cost of the various component floors f_i

Cost for running a model m_i as a kernel on a GPU device with CUDA depends on the following variables

D –Given data with dimensions $\mathbf{P} \times \mathbf{Y}$, where P accounts for the number of participants and Y is the number of columns for features.

Given data D (super set) is bootstrapped, i.e. resampled to create subsets $\{d_1, d_2, d_3, \dots, d_n\}$; each of the subset d_i is of same dimensions $\mathbf{P} \times \mathbf{Y}$

Model m_i 's time and space complexity K are denoted as $\text{time}(m_i)$ and $\text{space}(m_i)$, respectively. Time and space complexity of each model is different. However, in practice, each model is derived from one of a few base model types B_1, B_2, \dots, B_t , where t is small. For simplicity, we will assume $t \leq 5$ for the rest of our discussion.

Note: Both the execution order of threads within a block and blocks within a grid is undefined while all the threads in a block will be scheduled at once.

B. Type I variables cost for executing model m_i ignoring dependencies N

Variables in type I costs are regarding number warps. If w represents unit warp cost, then warp cost for running model m_i for type I variables will be (G is the number of variable in Type I)

$$\text{cost_type1}(m_i)_{f_j, t_k} = \sum_{h=1}^G \mu_h * w \dots \dots \dots (12)$$

Example for Type I -Variables that put constraints on the system because of user

μ_1 - #warps required for sync data access for model m_i numdataUniform

μ_2 -#warps required that are async data access for model m_i numdataDivergent

μ_3 #warp required when all threads execute same/ uniform code for model m_i

μ_4 #warp required where thread code is async/divergent for model m_i

μ_6 #warp shuffle (exchange data) in model m_i

μ_7 #warps branches in model for model m_i

A. Barriers and synchronization

μ_8 -lock-free synchronization

μ_9 - lock-based synchronization

μ_{10} – Number of Fences

μ_{11} – Number of Barriers

μ_{12} - Shared memory access times * amount of memory access for model m_i

μ_{13} - Global memory access times * amount of memory access for model m_i

μ_{14} - Thread local access times * amount of memory access for model m_i

μ_{15} - GPU coalesce concurrent reads by the threads in a group for model m_i

C. Type II variables cost for executing model m_i ignoring dependencies N

Based on variables identified above, the cost of executing model m_i in floor f_j at time t_k is written as (H is the total Type II variables)

Some portion of C_i * cost of C_i

$$cost_type2(m_i)_{f_j, t_k} = \sum_{h=1}^H \gamma_h * C_h \dots \dots \dots (13)$$

In the above equation $cost(m_i)_{f_j, t_k}$ represents the cost of scheduling / executing model m_i as part of the floor f_j at time t_k . Cost of executing m_i on a different floor at different instance of time will remain same if dependencies are ignored and is given by (2).

The term γ represents the portion of total resource of a particular kind. A portion of the resource that is allocated should be strictly integer, as the fractional amount of resource cannot be allocated.

Example for Type II - Variables that put resource constraints on the application

C_1 # of GPU multiprocessor

C_2 #in device memory allocation

C_3 Per block shared memory

C_4 Per thread private memory

C_5 maxnreg- the maximum number of registers to be allocated to a single thread args (n - #regs)

C_6 maxntid - the maximum number of threads in a thread block (CTA) args(n_x, n_y, n_z)

C_7 reqntid – the required number of threads in a thread block (CTA) args(n_x, n_y, n_z)

C_8 minnctapersm – the minimum number of threads block to be scheduled on a single multiprocessor (SM)

C_9 nregx – the current number of registers allocated per thread.

Total cost for model m_i can be given as

$$cost(m_i)_{f_j, t_k} = cost_type1(m_i)_{f_j, t_k} + cost_type2(m_i)_{f_j, t_k} \dots\dots\dots (14)$$

3. Solution Approaches

A. Method 1

One idea might be a graph approach to solving above and perform LP optimization [2] to find the static schedule.

B. Method 2

Use set-partitioning approach to finding an optimal set if we have multiple tables shown above.

C. Using the partitioning method to solve for the optimal space, using integer programming approach

In the matrix – F below each row represents a floor plan. A column of each row accounts for model m_i $1 \leq i \leq |M|$. Only following two values can be assigned to a cell

$$F(f_i, m_j) = \begin{cases} 0, & m_j \text{ cannot be executed in floor } f_i \\ 1, & m_j \text{ can be executed in floor } f_i \end{cases} \dots\dots\dots (15)$$

Table 16. Models execution sequence as floorplan.

	m_1	m_2	...	m_j
F_1	1	1		0
F_2	1	0		0
F_3				
.				
.				
.				
F_n	1	0		1

Each floor has a limited amount of resources. These resources are dependent variables, and listed as Type II in the previous section. Being constrained on the available resource puts a bound on the number of models it can fit. Each model m_i has different costs, and hence each floor can accommodate multiple numbers and combinations of models.

D. Digital Logic Circuit based solution on filling floor plan

One method to find a feasible solution might be to utilize a digital logic circuit made up of logical gates to fill each floor. We might also need to put a bound on the number of levels.

E. Bound for each floor for variables based upon user control (Type I)

There is no upper limit on the Type I variables for the total warps cost for model m_i . However, the solution should try to minimize the warp cost for Type I variables across the floors. i.e. for each floor, we want to reduce below equation's resulting warp cost.

$$\text{Min} \left| \sum_{h=1}^M f(f_i, m_h) * \text{cost_type1}(m_h)_{f_j, t_k} \right| \dots\dots\dots (16)$$

F. Bound for each floor on variables based upon maximum available resource (Type II)

Each variable listed under Type II, for each variable C_x the total portion of resource allocated in frame f_i at time t_k should be less than or equal to 1.

$$\sum_{h=1}^M f(f_i, m_h) * \gamma_{f_j, t_k, m_h} \leq 1 \dots\dots\dots (17)$$

Variables that put resource constraint on the system have a maximum of the upper bound.

4. Summary

Floor Tiles Planning (FTiP) is a conceptual structure used in scheduling an RDM algorithm's code execution by considering spatial and temporal dependencies. FTiP can improve the performance of computer algorithms for problems falling into the class of combinatorial, bootstrap based algorithms. The computational overhead in these algorithms is introduced using dimensions and redundant computations in data analyses. Hence, we propose original floor tiles planning to engineer solutions for shared and distributed memory systems, minimizing

calculations across algorithm steps. Computational steps such as distance matrix, Eigenvector calculations, etc. are prevalent in most data analysis algorithms. It is done by discretization of the algorithm into models and scheduling them for execution for efficient utilization of computing resources. This approach will also apply to various hardware software architectures like GPGPU, multiprocessors systems, smart memory systems, etc. The work detailed in this chapter is unique to GPGPU architecture and programming environments.

CHAPTER VIII

CLASSIFICATION AND PREDICTION OF PREFERENCES SHIFTS IN CUPS TASK

1. Introduction

An individual's decision making is influenced by various factors such as cognitive, memory, and neurological. Culmination of these results in differences in decision making under uncertainty. Decision making uncertainty can be broadly classified into three categories [101].

- 1) Decision Making Under Certainty (DMUC)
- 2) Decision Making Under Ignorance (DMUI)
- 3) Decision Making Under Risk (DMUR)

Different individual's risky decision making can be again classified into

- 1) Risk-avoidant
- 2) Risk-aversive
- 3) Risk-seeking

2. Models in RDM

Previous research has established that people's behaviors is adaptive. Behavior or decision making adaptability is subject to individual preferences. Framing effects is used to describe shifts in risk preferences [82]. A contemporary research argument is towards using conceptual models such as fuzzy trace theory for explaining framing effects. Below is a list of some of these models

- association theories
- axiomatic utility theory
- expected utility theory
- fuzzy-trace theory
- MINERVA-DM model
- Multiple Regression Model
- normative analysis
- Protection-motivation theory
- prototype/willingness model
- schema theory
- security-potential/aspiration theory
- self-regulation model
- subjective expected utility
- weighted utility function

Detailed explanation and analysis for each of the above-mentioned models is out of scope for current work. However, we encourage readers to refer one of many articles published on the same. Current literature still lacks a detailed survey article on risk preferences. Typically, scientists and practitioners studying decision making collect data for analysis either using non-experimental (real world) or experimental procedures. Experimental procedures include Q&A based self-reports and psychological tasks or games. One such task is CUPS task.

3. CUPS Task

The CUPS task is a computer based psychological simulation environment. It is used to measure individuals risk preferences [60]. On each side of the screen, you will see a certain number of cups (either 2, 3, or 5). The cups will have a return value over them, either positive or negative. For each trial, you will be given the option of choosing a cup from either side by clicking on your choice. The side with multiple cups has one cup with the return value under it. The other cups have nothing under them. So, your goal is to choose the best cups to maximize your score. Figure 27 is showing all the variations of trials belonging to gain or loss domain. Each of the domain (gain or loss) in turn is composed of three distinct payout scales. These payout scales can be categorized into risk advantageous, disadvantageous or neutral; depending on the expected value for making risky choice.

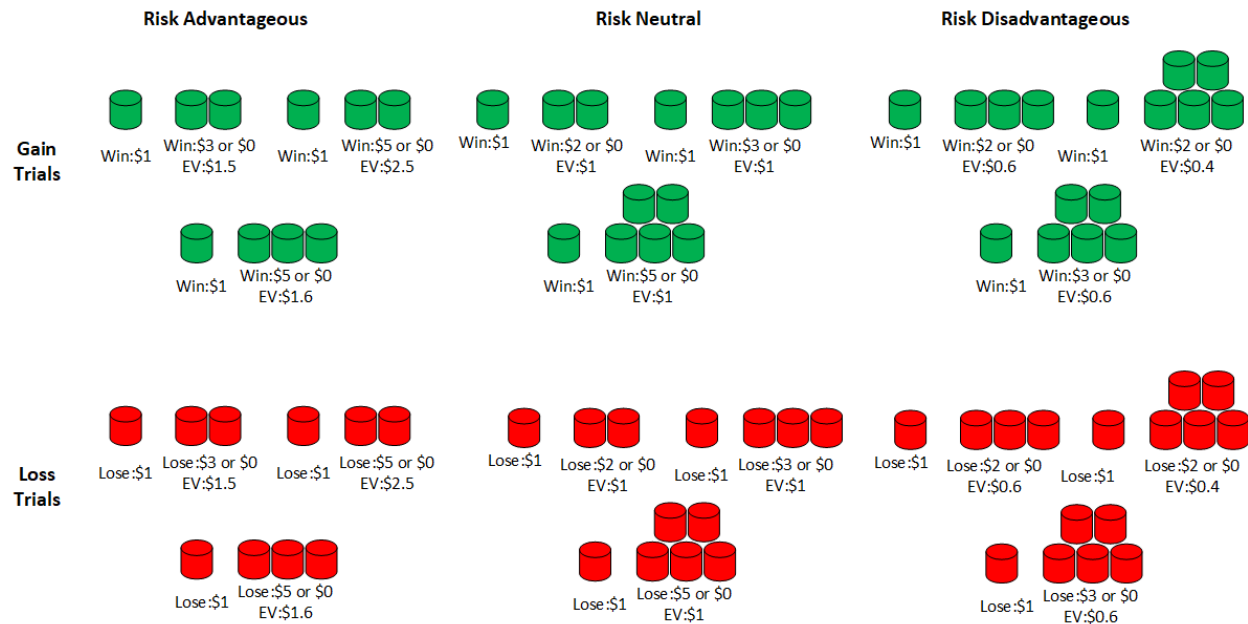


Figure 27. Summary of CUPS task trial type and expected values in each.

Commonly measured quantities in CUPS TASK are:

- Total number of decisions
- Total number of risky decisions
- Number of risky decisions – Gain domain
- Number of risky decisions – Loss domain
- Total number of advantageous risky decisions
- Number of advantageous risky decision – gain domain
- Number of advantageous risky decisions – loss domain
- Total number of disadvantageous risky decisions
- Number of disadvantageous risky decision – gains domain
- Number of disadvantageous risky decision – loss domain
- Total risk adjustment (i.e. # of advantageous risky decisions - # of disadvantageous risky decisions)
- Risk adjustment – gain domain
- Risk adjustment – loss domain

And interpreted measure of performance are

1. Optimal performance / Non- optimal performance = risky advantageous is high, and risky disadvantageous is low
2. Risk seeking / Risk aversion = total number of risky decisions is high / total number of risky decisions is low
3. Loss seeking = Total number of risky decision is high in loss domain.

4. Loss aversion = optimal performance in gain domain, risk aversion profile in loss domain.
5. Risk insensitive / Risk sensitive (Gain Domain) = Risk adjustment index near zero in gain domain.
6. Risk insensitive / Risk sensitive (Loss Domain) = Risk adjustment index near zero in loss domain.
7. Risk insensitive / Risk sensitive (Both Domains) = Risk adjustment index near zero in both domains.

Overall a participant can be categorized into one of twelve categories mentioned above.

4. Research Objective

Currently, state of the art research is lacking in terms of computational and mathematical models to identify preference shifts. There is no implementation of such for CUPS task.

In this project we want to apply advance mathematical and machine learning approach to

1. Identify patterns in preference shifts for CUPS tasks
2. Predict preferences in CUPS task

We would like to implement a deep neural network based model to analyze data collected for CUPS task to study prospect theory.

5. Experimental Setup

We collected data on preference in risky decision making and also time taken to make these decisions during each trial. CUPS task was used to collect data from 325 subjects. The task was published in an online survey using Amazon mTurk. Amazon mTurk is a workforce market place

to recruit human intelligence. On mTurk we created a human intelligence task (HIT) to recruit participants. Subjects from anywhere on the planet could perform the HIT. Participants were paid real money proportional to the final score on the task. This was done to keep the experiment in line with prospect theory and motivate the subjects for taking risky choices.

The CUPS Task was administered through RDMTk toolkit. The task was designed to have 90 trials in total. In turn both gain and loss domain had 45 trials each. Our experiment was designed to administer these trial domains in random order. Cups containing a risky option was also randomized for the cup containing the points along with the side for multiple cups placement. We also kept records of time taken during each trial.

6. Model Specifications

The model specified in this section is formulated generic across all tasks instead of being more specific to cups task alone. We want to develop a model which would help us detect presence or absence of seasonality of predictable pattern in subjects' preferences. For this purpose, notation and approach can be reused for others with little or no modification. Hence, we formulate cups task data in terms of trial data for N subjects. Which in turn consists of

T a set consisting of trials per subject. $T \in t_1, t_2, t_3, \dots, t_\tau$; where τ is the total number of trials for the subject. It is same across all the subjects in an experiment.

F is a set of features for each t_i , $F \in \{f_1, f_2, f_3, \dots, f_\epsilon\}$; where ϵ is the number of features (data types) collected per trial. Variations in values of f_i for a trial categorizes subject's preference for risky decision making.

P is a set $P \in p_1, p_2, p_3, \dots, p_p$; which represents distinct measured preferences learned from trials for all N subjects. The total number of trials across all the subjects is determined as $N \times \tau$.

The purpose of the model (Figure 28) is to predict preference labels of future trials using the data available from past trial.

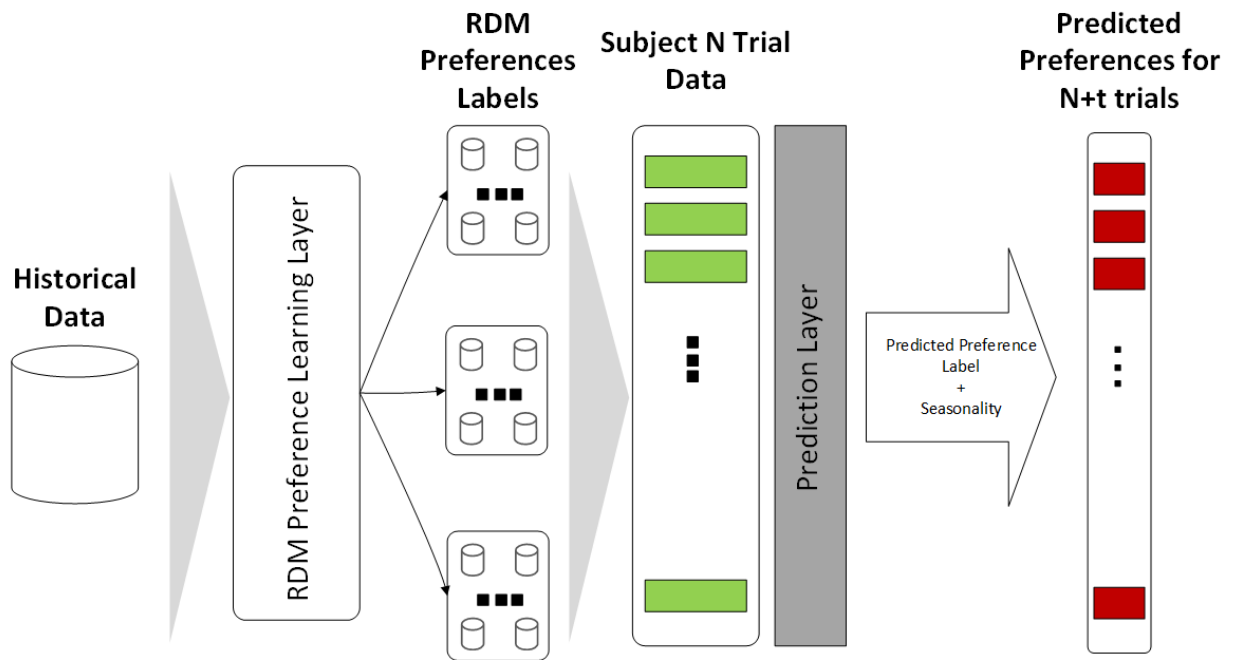


Figure 28. Model to predict risk taking preference shifts.

The proposed model can be categorized into two variations depending on the method used to determine set P from the past trials of the same individual.

- Unsupervised prediction model for preference shifts if P is determined from analyzing historical data first.
- Supervised prediction model for preference shifts if P is a given as an input parameter to the model.

In the current model implementation for the CUPS task, P is given as an input parameter and also learned using a reinforcement-learning algorithm [71].

The model has two parts; in the first stage participants trial data is compared against global trial record to determine preference category for the trial. And the second stage is to check for seasonality and predict future preferences. Changes in the preference labels across trials gives us hints on the rate at which subject's preferences shifts.

A. Supervised Prediction for Preference Shifts

Subject's performance in CUPS task is categorized as one of the twelve trial preferences domains [60, 61] (see Section 3). In our model we are identifying preferences P as a set of numerical labels ranging from {1:12}.

We identify trials that are closest to each other and give same preference label. One naïve method to find trials closest to each other can be done by reducing the squared distance among trials that are given same label. This can be achieved by grouping all the trial data into twelve categories. Mathematically, we would like to minimize error from grouping similar trials across all subjects and can be written as

$$\text{minimize } (e) = \sum_{i=1}^{12} \sum_{j=1}^{N^* \tau} (|t_i - t_j|)^2 \dots\dots\dots (18)$$

Where i and j are indices of trials being compared. The trials representing centroids are chosen at random (i.e. trials for index i). In terms of features in the trials the above equation becomes

$$\text{minimize } (e) = \sum_{i=1}^{12} \sum_{j=1}^{N^* \tau} \sum_{k=1}^{\epsilon} (|f_{ik} - f_{jk}|)^2 \dots\dots\dots (19)$$

Here, $|t_i - t_j|$ or $|f_{ik} - f_{jk}|$ represents the Euclidean distance between trials i and j or features of those trials respectively. This clustering can be accomplished by applying kmeans clustering algorithm.

a) KMeans

1. Define the initial group of k centroids (selected at random).
2. Assign each trial to the closest cluster centroid.
3. Recalculate cluster centroids.
4. Repeat steps 2 and 3 iteratively until trials assigned to same cluster is constant.

Algorithm 5. KMeans clustering algorithm.

It is one of the most common clustering technique, and steps in the algorithm are as described below. It uses an iterative refinement approach, as seen above

The K-Means is a greedy, computationally efficient technique, being the most popular representative-based clustering algorithm.

B. Unsupervised Prediction for Preference Shifts

The unsupervised prediction model is similar to supervised approach except that the size of set and preference labels are learned through analyzing historical data across participants. There are a number of different approaches to determine preference labels from the analysis of features. In our current work we discuss results obtained from using an ensemble clustering model.

We first divide the data into 70:30% ratio; where 70% is used for training and 30% for testing. Our model uses patterns in the preference shifts among 70% trials per subject of the

training data to predict preferences in the other 30% of trials. We used ARIMA based prediction as we were interested in finding presence of seasonality.

C. Prediction Layer

a) ARIMA

Most parametric models such as averaging, weighted moving average, exponentially weighted moving average, ARMA (auto regressive moving average) and ARIMA (auto regressive integrated moving average) proposed by [13] work on the principle that future/predicted values depend only on immediate historical value. For predicting preference shifts, in ARIMA we also assume that data consists of preference label at trial t , P_t , and a random noise (white noise ε_t of zero mean and standard deviation). General form of ARIMA with order p and q to determine vectors a and d respectively, namely ARIMA(p, q) is

$$P_t = a_1 P_{t-1} + \dots + a_p P_{t-p} + \varepsilon_t + d_1 \varepsilon_{t-1} + \dots + d_q \varepsilon_{t-q} \dots \dots \dots (20)$$

Using backshift operator B , the model can be succinctly written as $A_p(B)X_t = D_q(B)\varepsilon_t$ where $A_p(\cdot)$ and $D_q(\cdot)$ are polynomials

$$A_p(B) = 1 - a_1 B - a_2 B^2 - \dots - a_p B^p \dots \dots \dots (21)$$

$$D_q(B) = 1 + d_1 B + d_2 B^2 \dots + d_q B^q \dots \dots \dots (22)$$

with $B^k Z_t = Z_{t-k}$, for $K = 1, 2, \dots$ $Z = X$ or ε .

ARIMA consists of AR, MA and a combination of both (ARMA); order of p and q in ARIMA determines the selection of appropriate model. This is achieved by analyzing

autocorrelation function (ACF) and partial autocorrelation function (PACF). Table 17 gives properties of ACF and PACF for the corresponding models.

Table 17. Properties of ACF and PACF to determine ARIMA model.

	MA(q)	AR(p)	ARMA(p, q)
ACF	Spikes up to lag q, and cuts off afterwards.	Slow decay, infinite tails off, declined exponential and/or cosine waves	Declined exponential and/or cosine waves after p-q
PACF	Slow decay, infinite tails off, declined exponential and/or cosine waves	Spikes up to lag p, and cuts off afterwards	Declined exponential and/or cosine waves after p-q

Autocorrelation function measures the linear relationship between values of a time series lagged by k units. Estimate of the kth lag autocorrelation γ_k is

$$\gamma_k = \frac{c_k}{c_0}, \quad \text{where } c_k = \frac{1}{N} \sum_{t=1}^{N-k} (z_t - \bar{z})(z_{t+k} - \bar{z}), \quad k = 0, 1, 2, \dots, K \text{ and } \bar{z} \text{ is the mean, with } z = X \text{ or } \varepsilon.$$

Auto covariance is used to make series stationary and also explains correlation between neighboring pair of time series values. The covariance between z_t and its value z_{t+k} separated by k intervals of time is called auto-covariance at lag k.

Figure 29's ACF and PACF corresponds to ARMA(p, q) for data without differencing to account for seasonal factor. And Figure 30 corresponds to same data with differencing d=1 that removes seasonality, it has a corresponding AR(p) model.

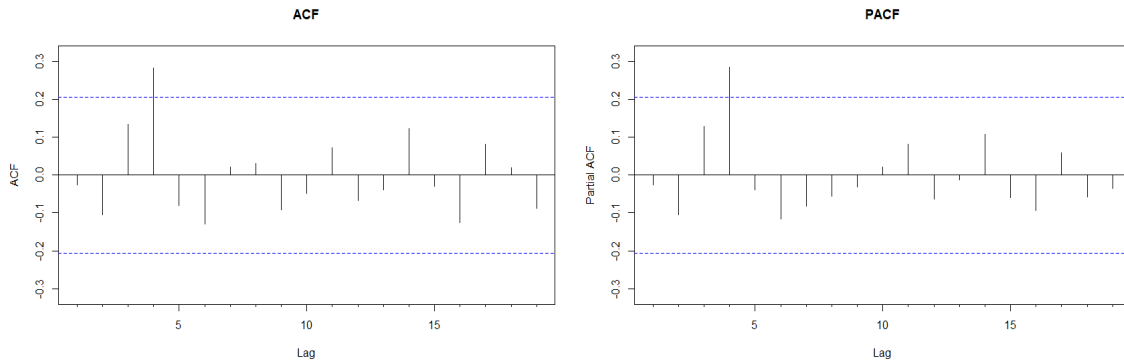


Figure 29. ACF and PACF for data without differencing.

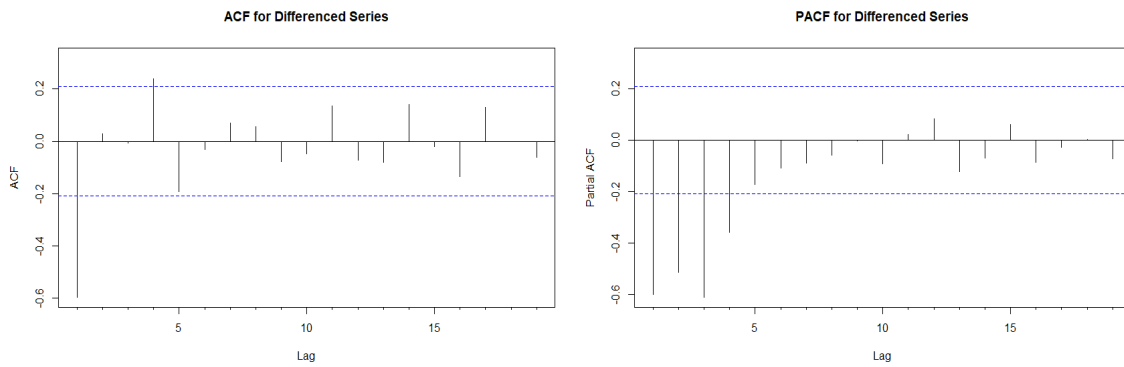


Figure 30. ACF and PACF for data with differencing.

ARIMA also assumes that series is linear and stationary with respect to statistical properties of the given preference sequence. If autocorrelation cuts off fairly quickly, or dies down quickly then the time series is considered to be stationary and on the contrary if it dies down very slowly then it is considered non-stationary. We can convert a non-stationary series into stationary by differencing. When differencing, d , is used in ARIMA to make it a stationary time series it is denoted as $ARIMA(p,d,q)$.

Precise estimate of fairly quickly or slowly is not defined and is contextual. For this reason, historical data is preprocessed by smoothing and de-trending. As can be seen in the ACF and PACF plots (Figure 29 and Figure 30) data with no differencing is fitted by an $ARIMA(p, q)$ model whereas after differencing it is fitted by an $AR(p)$. For preferences in CUPS task trial data,

developing such contextual knowledge is difficult and nearly impractical when developing a model for large scale implementation across subjects. For this reason ARIMA is run iteratively for various values of p, d, and q to account for all permutations of model order and its implications.

Time complexity for ARIMA(p,d,q) can be computed as follows. Using the ARIMA formulation given in [13],

$$\alpha = inv(M_A^T M_A) M_A^T M_b \dots\dots\dots (23)$$

where, α is the vector of AR coefficients, matrices M_A and M_b are of the following form,

$$M_A = \begin{bmatrix} X_p & X_{p-1} & \dots & X_1 \\ X_{p+1} & X_p & \dots & X_2 \\ X_{p+2} & X_{p+1} & \dots & X_3 \\ \vdots & \vdots & \dots & \vdots \\ X_{t-1} & X_{t-2} & \dots & X_{t-p} \end{bmatrix} \quad M_b = \begin{bmatrix} X_{p+1} \\ X_{p+2} \\ X_{p+3} \\ \vdots \\ X_t \end{bmatrix}$$

Equation 1 is used to compute both the p and q values. Computation of AR and MA coefficients thus takes $O((N - p)p^2)$ and $O((N - q)q^2)$ time, respectively, where N is the length of historical values. Hence the total time will be

$$O((N - p)p^2 + (N - q)q^2)$$

We can easily see that model complexity grows significantly as we consider higher order values for p and q.

b) SARIMA

SARIMA incorporates both seasonal and non-seasonal aspects of a time series as a multiplicative model of the two as discussed in [103]. Mathematically it can be represented by

$$a_p(B) A_p(B^s) X_t = d_q(B) D_q(B^s) \varepsilon_t \dots\dots\dots (24)$$

where, $a_p(B)$ and $d_q(B)$ is the non-seasonal and $A_p(B^s)$ and $D_q(B^s)$ represent the seasonal part for the data and white noise. Time complexity can be computed similar to ARIMA, except that SARIMA also consists of ARIMA as its subcomponent, therefore we can easily see that SARIMA time complexity is

$$O\left(\sum_{i=1}^{L_s} \left(\left(\frac{N}{S_i} - p_i \right) p_i^2 + \left(\frac{N}{S_i} - q_i \right) q_i^2 \right) + (N - p)p^2 + (N - q)q^2\right)$$

where, L_s is number of seasonality being considered and S_i is the seasonality.

Preference labels for CUPS task data shows different seasonality patterns depending on the subject. Most common patterns are across trials. We can also mine for seasonality in different dimensions.

7. Experimental Results

Historically CUPS task has been analyzed for mean number of times risky decision was made by subjects in gain and loss domain. Figure 31 shows the number of risky decisions took for gain vs. loss domains and each trial in turn is divided three sub categories i.e. risk advantageous, risk disadvantageous and risk neutral. From the graph we can infer that subjects took most risky decisions during risk advantageous trials for gain domain. And risk disadvantageous trials have the most risky decisions for the loss domain.

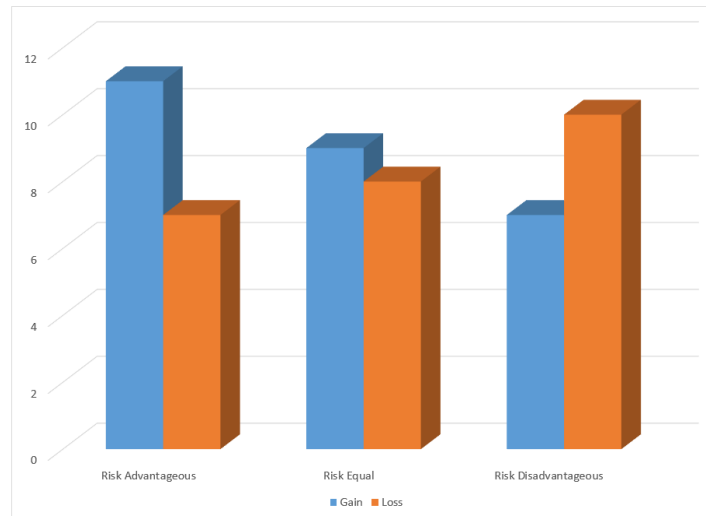


Figure 31. Means of risky choices in (a) the loss and (b) the gain domain, as a function of subject group and expected-value (EV) level (risk advantageous trials; risk equal expected value trials; risk disadvantageous trials). Subjects received 15 gain trials and 15 loss trials for each of the three EV levels.

If we drill down into data we can realize more details.

Table 18 Measure of average number of risky decisions across gain, loss domains and overall. advantageous and disadvantageous risky decisions are also listed.

Total number of decisions	90
Total number of risky decisions	55
Number of risky decisions – Gain domain	27
Number of risky decisions – Loss domain	27
Total number of advantageous risky decisions	29
Number of advantageous risky decision – gain domain	15
Number of advantageous risky decisions – loss domain	14
Total number of disadvantageous risky decisions	25
Number of disadvantageous risky decision – gains domain	12
Number of disadvantageous risky decision – loss domain	13

Measuring number of risky decisions in each domain and keeping track of advantageous ones verses disadvantageous per subject categorizes him / her (as given in Table 18) into one among twelve interpreted risk preferences (as shown in figure 32).

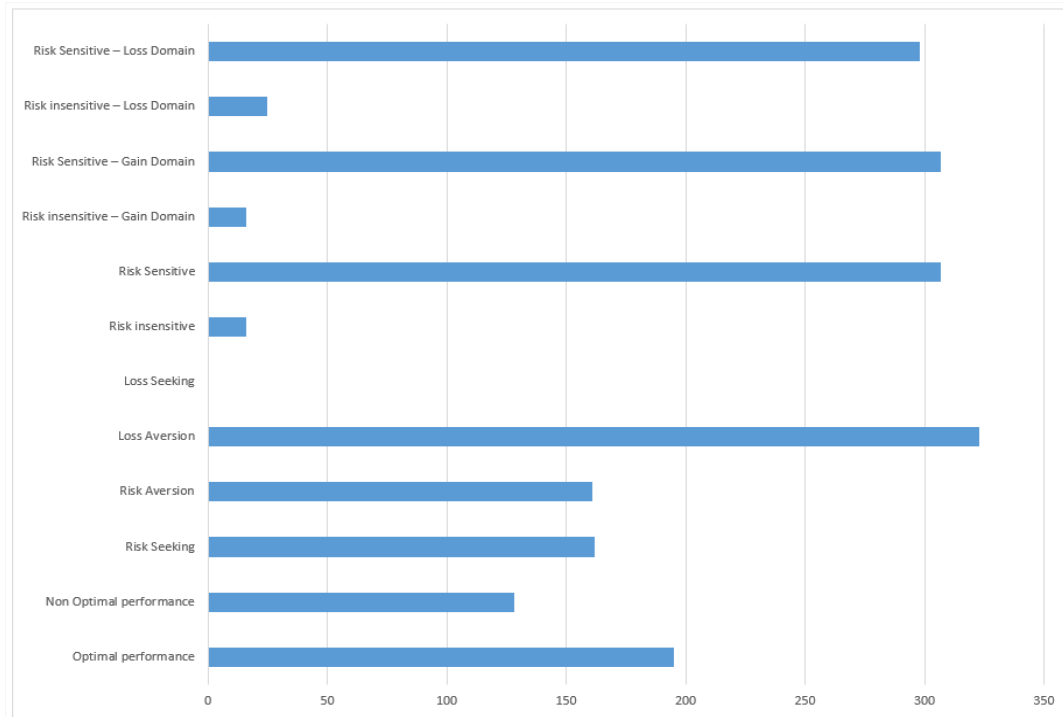


Figure 32. Categorization of subjects of prospects theory experiment into twelve risky preferences types as suggested in the literature.

Calculating counting statistics for the number of, advantageous and disadvantageous risky decisions made by each subject allows categorization into different risk preference types. In the below experiments results we categorized each trial by subject according to risk preference type it would lead. We fixed the total number of distinct preference types at 12, for supervised prediction of preference shifts.

We are showing results from a few representative subjects; as our model predicts future risky preferences types per subject and listing all 324 subjects is unnecessary. Figure 33 represents

a typical sample of subject's performance on the CUPS task. It is important to notice presence of a seasonal pattern and trend in the subject response to different risk domains. In Figure 33 the x-axis is the trial number and y-axis is different for each sub graph. For data y-axis represents the numeric label given to preference measures.

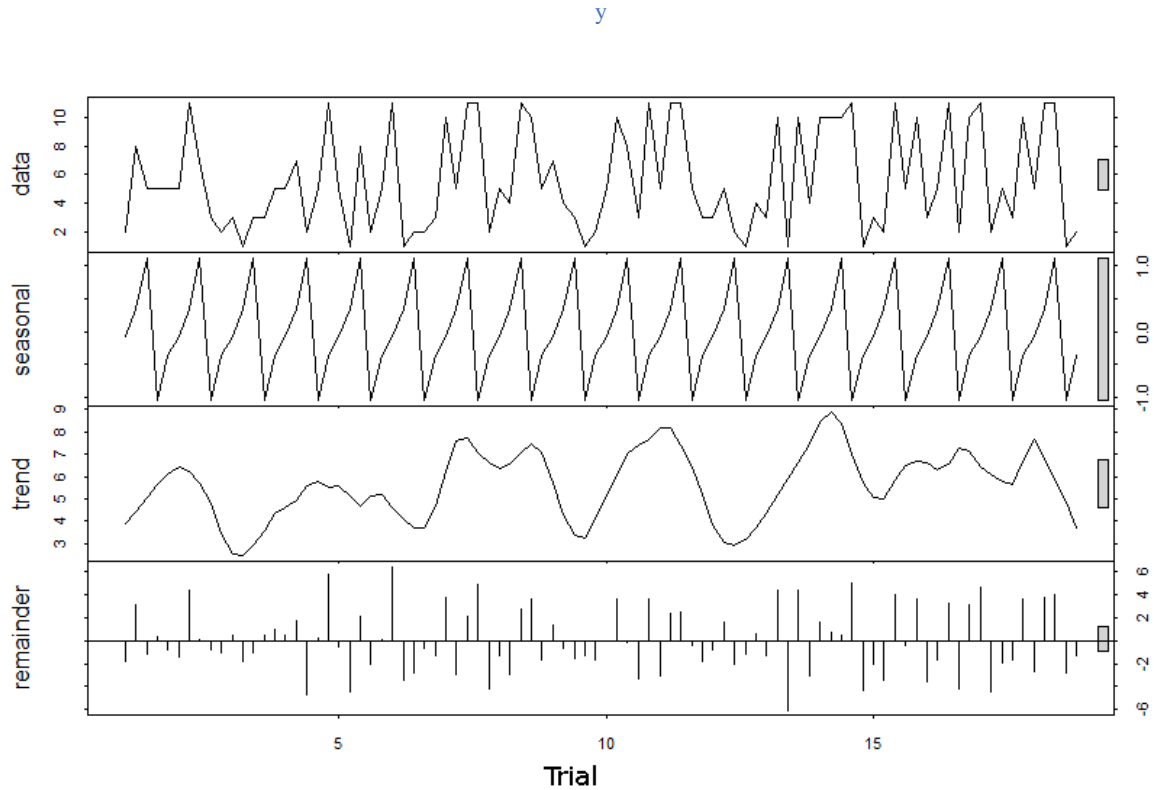


Figure 33. Typical data sample for subject's performance on CUPS task, revealing seasonality, and trend.

A. Presence of seasonality in risky preferences

An important result of our analysis of CUPS task data is identification of seasonality presence. Each subject tends to cycle through a set of preference types at frequent intervals.

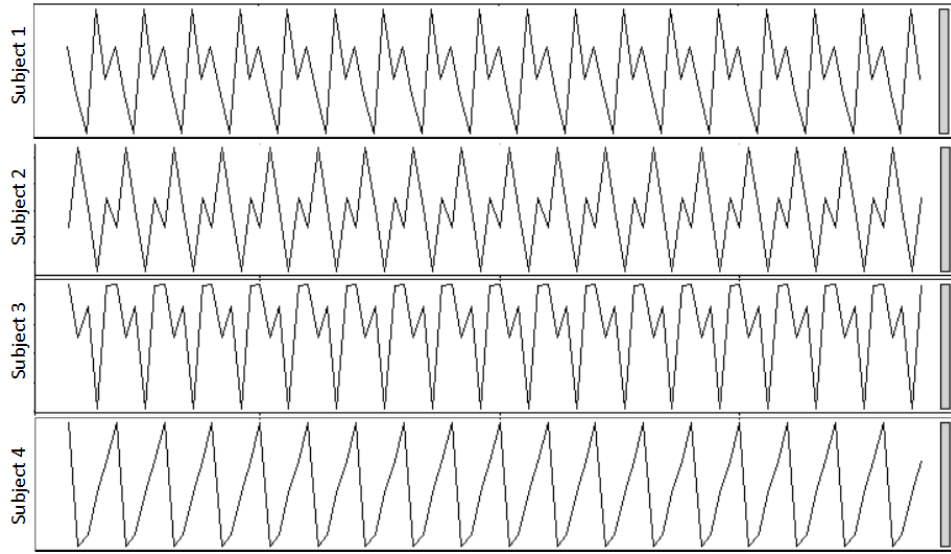


Figure 34. Seasonality patterns for four random subjects selected from survey pool of 324 subjects from prospects theory experiment. All subjects showed a definite sign of seasonal pattern across 90 trials as shown in the figure.

Our analysis revealed that each subject showed evidence for having seasonal pattern (see Figure 34). From the presence of seasonal patterns, one can infer that subjects go through a set of risk preference types. Their performance on the task can be further modeled as transitions through these fixed number of risk preferences. Risk preferences for each participant is different. Analysis of same data using supervised modeling showed that switching pattern between preference types is more frequent. On the other hand, same data when analyzed using unsupervised modeling showed more stability. This contradiction warrants further investigation. Our initial hypothesis is that unsupervised learning yielded fewer categories thereby resulting in fewer preference type shifts.

B. Predicted vs. actual risk preferences

The number of risk preference types were fixed to twelve for supervised model. And unsupervised model learned the number of risk preference types in the data to be five (see Figure 35). We are ignoring two clustering solutions as that would suggest only a binary risk preference

type. For CUPS task that would translate to checking if participants choose to take risk or go with certain (fixed) gain or loss option.

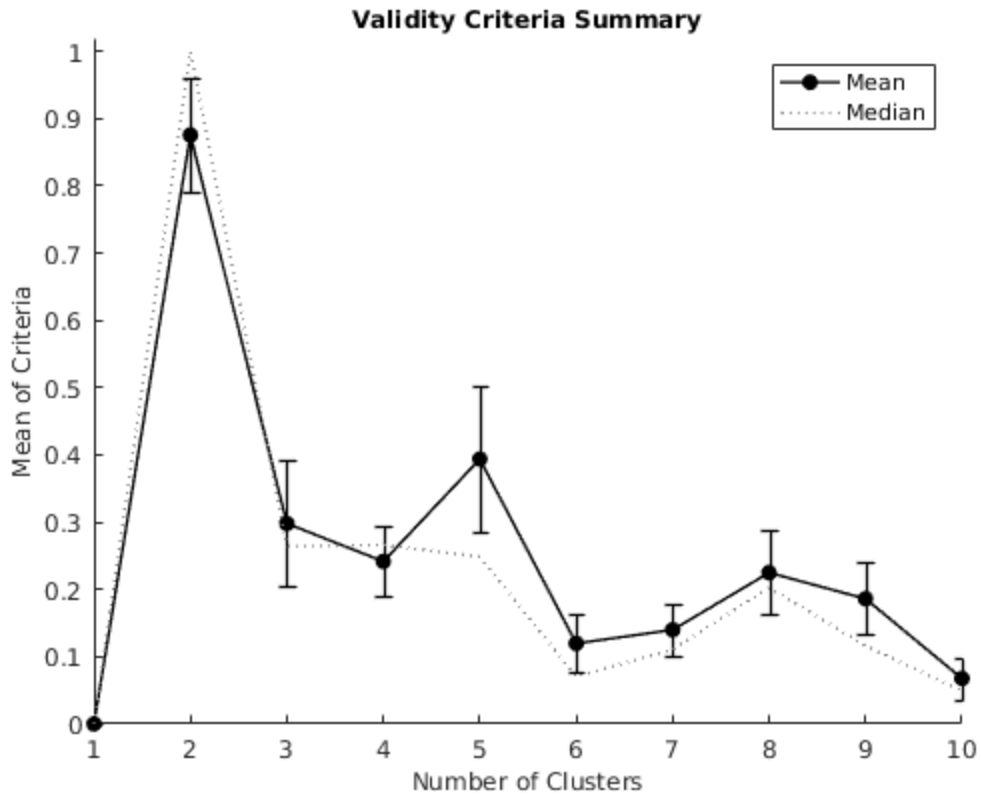


Figure 35. Reinforced learning model output showing valid number of clustering result for subject's performance in prospects theory experiment.

Figure 36 and 37 are samples of predictions, after running supervised and unsupervised version of our prediction model. The x-axis represents the trial number and corresponding values on the y-axis gives the preference label. The darker and lighter shaded regions represent the 80 and 90 percentile prediction intervals. Our work establishes the capability to predict shifts in preferences in risky decision making. However, visual inspection and validation metric shows room for improvement in the accuracy and thus the need for more thorough investigation

Validation metrics used in our analysis were AIC (Akaike information criterion), BIC (Bayesian Information Criterion), and RMSE (Root Mean Square Error).

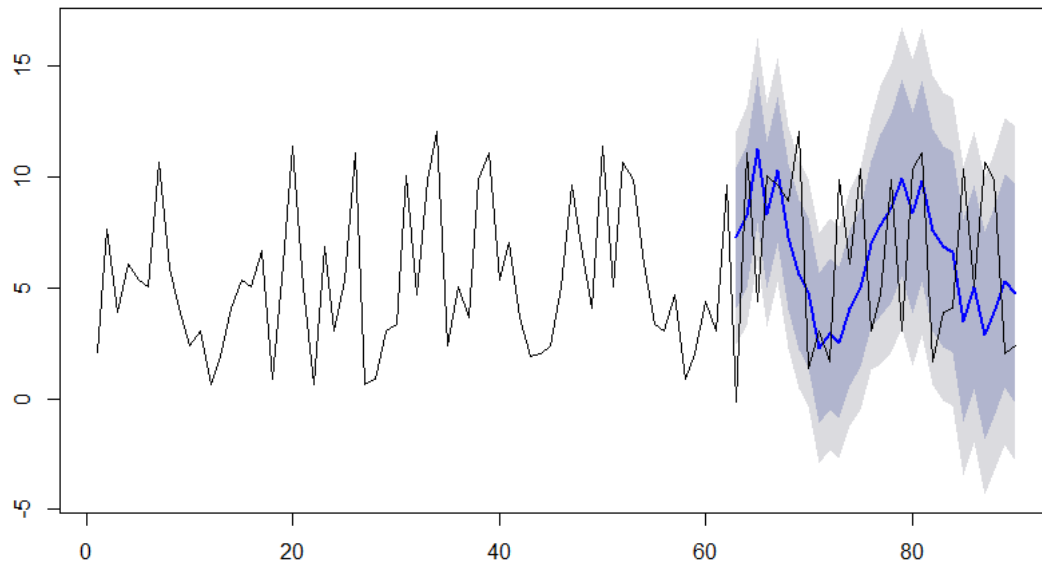


Figure 36. Example for predicted reference shifts in subjects' performance using supervised model.

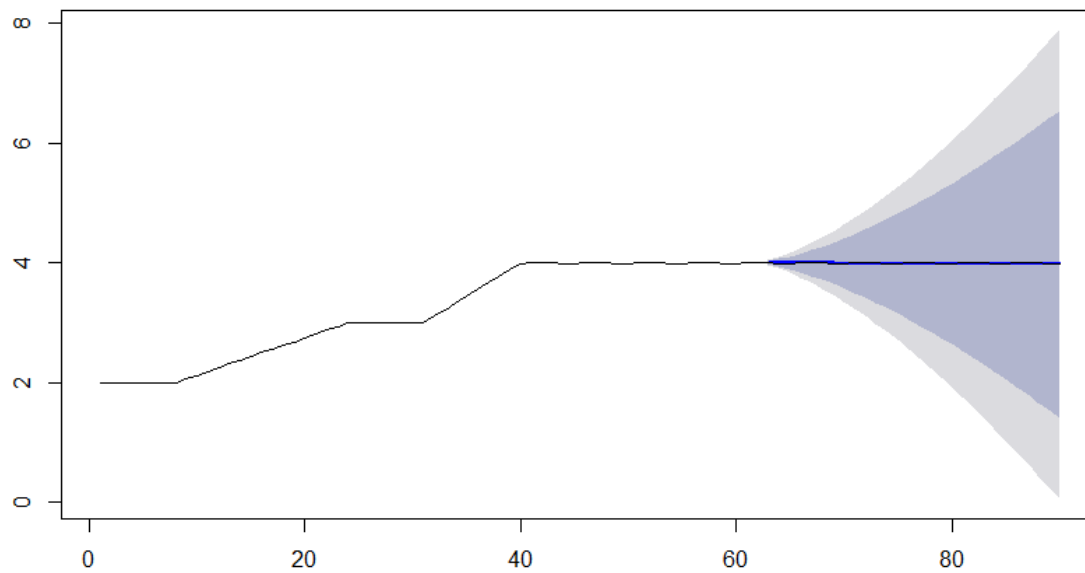


Figure 37. Example for predicted preference shifts in subjects' performance using unsupervised model.

Table 19 and 20 lists validation metrics of predicted values for supervised and unsupervised models respectively. The fitting models are chosen as that values of validation metrics i.e. AIC, BIC and RMSE are minimized.

Table 19 Validation metrics of predicted preference shifts using supervised model.

Subject#	AIC	BIC	RMSE
1	353.943	402.867	2.429392
2	369.918	431.604	2.483127
3	355.747	398.29	2.807285
4	348.092	386.381	2.667318

Table 20 Validation metrics of predicted preference shifts using unsupervised model.

Subject#	AIC	BIC	RMSE
2	-323.81	-260	0.008554
4	-194.73	-188.14	0.04179
5	-241.03	-219.76	0.026996
33	-275.89	-265.25	0.021981

8. Summary

In this work we analyzed 324 subject's performances on the CUPS task. We developed two variations of a model to identify and predict preference shifts in subject's choices. The results revealed a frequently changing risk preference characteristic in the supervised model; whereas unsupervised model showed that changes in risk preference were more stable. However, both showed that there is definite change in preference patterns. Also, the number of risk preference types present in data detected by unsupervised version of the model was far lesser at five than literature suggested twelve.

Currently developed model identifies the presence of risk preferences by matching across various subject's trial performance for similarity. However, we know from the empirical study that subject's performance and risk preference in subsequent trials is based on past experiences learned over time. This temporal dimension is currently missing in the current model to account for varying risk preferences types. Also, the prediction model used in this work is rather a naïve seasonal ARIMA model with very less accuracy. Our model fails to improve on prediction accuracy for subjects who change risk preferences more frequently over the course of the task. In the future a more accurate and robust model needs to be developed both in terms of identification of risk preference type from the historical data and also the prediction model.

CHAPTER IX

GENERAL DISCUSSION, CONCLUSIONS AND FUTURE WORK

The main goal of my dissertation was to pave way for future research in RDM analysis through interesting works in various directions. We addressed two main problem areas among RDM researchers: One to collect data globally and second to improve computational performance of RDM analysis algorithms. The second aspect of my research is to enable analysis on even smaller devices to enable non-computational scientists to run RDM analyses.

RDMTk was developed with a different design principle than contemporary toolkits. It allows the researcher to spend more time in the design phase of a DM study as opposed to the creation and data collection phases. It also provides a new methodology to enable a participant's decision-making behavior to be visually analyzed. An advanced clustering and deep neural network learning algorithm are currently being developed and planned to be incorporated in the near future. We hope its ease of use and innovativeness will encourage the RDM community to contribute and develop the tool further. We believe all researchers in the decision-making community will benefit, as it is an open source toolkit.

In the future, RDMTk can be applied to a host of domains that involve risk. We will integrate multiple data sources, including fields such as intelligent transportation systems (ITS) to help avoid risky situations on the road. In the finance arena, risky situations can be applied to loan applications, loan recovery, and stocks. Marketing information can be used to deter or motivate

purchasing patterns. The toolkit can also be used to make better administrative decisions. Figure 38 depicts our vision and future development road map for RDMTk.

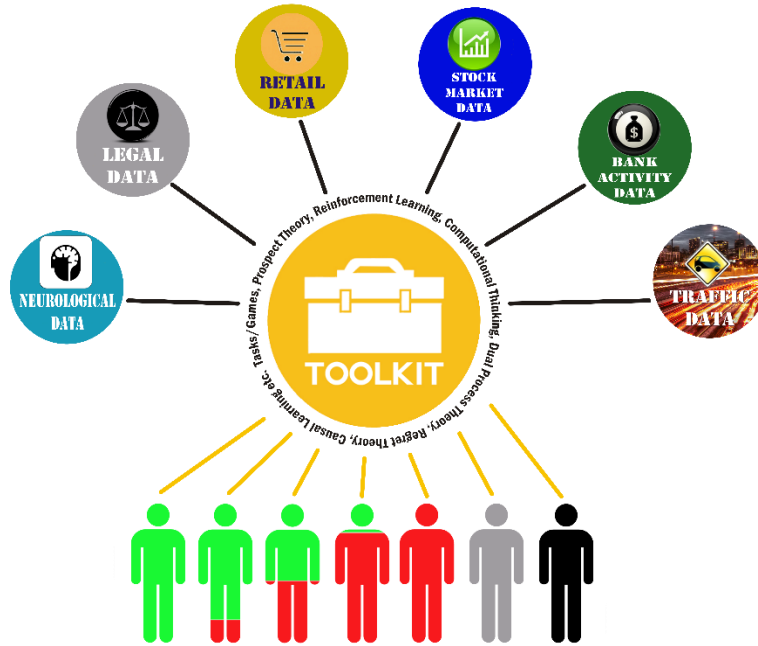


Figure 38. RDMTk application bird's eye view.

In our work to analyze computation requirement for the commonly used RDM techniques, we identified ensemble clustering part (that uses Agglomerative, K-means, K-medians and Spectral clustering) to be the main limiting factor for the high scalability of RDM analysis tools. Through empirical analysis we identified spectral clustering to be one of the most expensive among the three clustering algorithms used in Newman's work. Using distributed memory based parallel implementation and parallelized algorithm across different models in the ensemble gave us excellent speedups and improved efficiency for varying the number of models/partitions while keeping data size constant. Contrary to Huang et al., our approach scales well across different models when data size is kept constant. However, for studying large datasets using ensemble

clustering with the suggested approach still remains to be carefully investigated, as a number of big-data challenges will need to be tackled. In future work we also plan to parallelize individual clustering algorithms on distributed systems along with models in ensemble for further improvement. Also, minimizing computations across ensemble, such as distance matrix, eigen vector calculations etc. should be explored. Reduction in the number of bootstraps required to give existing consensus function result will also be a challenging task and worth investigating. Parallelization of ML algorithms such as this is still a new domain and obviously a lot of challenges need to be addressed, especially for the big-data situations (volume, storage, transfer and analytics), and we intend to continue our pursuit to push the scalability limits by improving the performance in the context of RDM techniques and the corresponding toolkits.

Shared memory CPU-GPU implementation of ensemble clustering algorithm achieves a speedup of 130x over its counterpart CPU based sequential version. The parallel implementations discussed here are planned to be incorporated as a part of RDMTk [34]. Researchers in RDM can then run ensemble cluster analysis just by clicking few buttons. Implementation of multiple CPU threads using OpenMP and GPU and using MPI to run even bigger datasets across different machines and accumulating single result is the next phase of progression for this line of research work.

We proposed a floor tiles planning based framework for a new and novel approach to minimizing computations by reducing redundancy in algorithms, specifically in the class of combinatorial, bootstrap based algorithms. More research is needed to evaluate and address the applicability of the approach through more precise theoretical formulations. Simulations will test the validity of the approach on different computer processor and system architectures such as

multi-core processors, GPGPU devices, and smart devices (or smart memory devices). Questions remain, such as how best to implement this approach or whether it can be driven from processor architecture through runtime systems? Answers to these issues could lead to a truly reconfigurable system. Dynamic code injection can result in a more efficient runtime system. There should also be an investigation into applying this approach to both distributed and shared memory systems. This topic needs more thorough examination, and many more questions should be asked to verify its validity and applicability.

REFERENCES

- [1] Evrim Acar and Bülent Yener. Unsupervised multiway data analysis: A literature survey. *Knowledge and Data Engineering, IEEE Transactions on*, 21(1):6–20, 2009.
- [2] Emmanuel Agullo, Olivier Beaumont, Lionel Eyraud-Dubois, and Suraj Kumar. Are static schedules so bad? a case study on cholesky factorization. 2015.
- [3] Muna Al-Razgan and Carlotta Domeniconi. Weighted clustering ensembles. In *Proceedings of the 2006 SIAM International Conference on Data Mining*, pages 258–269. SIAM, 2006.
- [4] Ignacio Arnaldo, Kalyan Veeramachaneni, and Una-May O’Reilly. Flash: A gp-gpu ensemble learning system for handling large datasets. In *European Conference on Genetic Programming*, pages 13–24. Springer, 2014.
- [5] Antoine Bechara. Decision making, impulse control and loss of willpower to resist drugs: a neurocognitive perspective. *Nature neuroscience*, 8(11):1458–1463, 2005.
- [6] Antoine Bechara, Antonio R Damasio, Hanna Damasio, and Steven W Anderson. Insensitivity to future consequences following damage to human prefrontal cortex. *Cognition*, 50(1):7–15, 1994.
- [7] Antoine Bechara, Hanna Damasio, Daniel Tranel, and Antonio R Damasio. The iowa gambling task and the somatic marker hypothesis: some questions and answers. *Trends in cognitive sciences*, 9(4):159–162, 2005.
- [8] Zakaria Benmounah and Mohamed Batouche. A parallel distributed system for gene expression profiling based on clustering ensemble and distributed optimization. In *International*

Conference on Algorithms and Architectures for Parallel Processing, pages 176–185. Springer, 2013.

[9] Pavel Berkhin et al. A survey of clustering data mining techniques. *Grouping multidimensional data*, 25:71, 2006.

[10] Laurent Bernaille, Renata Teixeira, and Kave Salamatian. Early application identification. In *Proceedings of the 2006 ACM CoNEXT conference*, page 6. ACM, 2006.

[11] Michael H Birnbaum. Surveywiz and factorwiz: Javascript web pages that make html forms for research on the internet. *Behavior Research Methods*, 32(2):339–346, 2000.

[12] A. J. Bishara. Selected code, data, and supplementary materials, 2017.

[13] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[14] Andrei Broder. A taxonomy of web search. In *ACM Sigir forum*, volume 36, pages 3–10. ACM, 2002.

[15] Michael J Brusco and Douglas Steinley. Cross validation issues in multiobjective clustering. *British Journal of Mathematical and Statistical Psychology*, 62(2):349–368, 2009.

[16] Tadeusz Calinski and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27, 1974.

[17] C-Y Peter Chiu, Sarah J Tlustos, Nicolay Chertkoff Walz, Scott K Holland, James C Eliassen, Lori Bernard, and Shari L Wade. Neural correlates of risky decision making in adolescents with and without traumatic brain injury using the balloon analog risk task. *Developmental neuropsychology*, 37(2):176–183, 2012.

[18] Lee J Cronbach and Paul E Meehl. Construct validity in psychological tests. *Psychological bulletin*, 52(4):281, 1955.

- [19] Peng Dai. *Decision Making under Uncertainty: Scalability and Applications*. PhD thesis, University of Washington, 2011.
- [20] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.
- [21] Michiel JL de Hoon, Seiya Imoto, John Nolan, and Satoru Miyano. Open source clustering software. *Bioinformatics*, 20(9):1453–1454, 2004.
- [22] Joshua R de Leeuw. jspsych: A javascript library for creating behavioral experiments in a web browser. *Behavior research methods*, pages 1–12, 2014.
- [23] Peter J Denning. The locality principle. *Communications of the ACM*, 48(7):19–24, 2005.
- [24] Geert Dom, Bieke De Wilde, Wouter Hulstijn, Wim Van Den Brink, and Bernard Sabbe. Decision-making deficits in alcohol-dependent patients with and without comorbid personality disorder. *Alcoholism: Clinical and Experimental Research*, 30(10):1670–1677, 2006.
- [25] Michael R Dougherty and Rick P Thomas. Robust decision making in a nonlinear world. *Psychological review*, 119(2):321, 2012.
- [26] S Draine. Inquisit [computer software]. *Seattle, WA: Millisecond Software*, 1998.
- [27] Joseph C Dunn. Well-separated clusters and optimal fuzzy partitions. *Journal of cybernetics*, 4(1):95–104, 1974.
- [28] Bradley Efron. Bootstrap methods: another look at the jackknife. *The annals of Statistics*, pages 1–26, 1979.
- [29] Bradley Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in Statistics*, pages 569–593. Springer, 1992.

- [30] Amy Eschman, James St James, Walter Schneider, and Anthony Zuccolotto. Psychmate: Providing psychology majors the tools to do real experiments and learn empirical methods. *Behavior research methods*, 37(2):301–311, 2005.
- [31] Maurizio Filippone, Francesco Camastra, Francesco Masulli, and Stefano Rovetta. A survey of kernel and spectral methods for clustering. *Pattern recognition*, 41(1):176–190, 2008.
- [32] Arnaud Fréville. The multidimensional 0–1 knapsack problem: An overview. *European Journal of Operational Research*, 155(1):1–21, 2004.
- [33] Pablo Garaizar and Ulf-Dietrich Reips. Visual dmdx: A web-based authoring tool for dmdx, a windows display program with millisecond accuracy. *Behavior research methods*, 47(3):620–631, 2015.
- [34] Vinay B Gavirangaswamy. The risky decision making toolkit (rdmtk), 2017.
- [35] AS Gevins and BC Cutillo. Neuroelectric evidence for distributed processing in human working memory. *Electroencephalogr. Clin. Neurophysiol*, 87:128–143, 1993.
- [36] Stamatios Giannoulakis and Nicolas Tsapatsoulis. Instagram hashtags as image annotation metadata. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 206–220. Springer, 2015.
- [37] Enrico Glerean, Raj K Pan, Juha Salmi, Rainer Kujala, Juha M Lahnakoski, Ulrika Roine, Lauri Nummenmaa, Sami Leppämäki, Taina Nieminen-von Wendt, Pekka Tani, et al. Reorganization of functionally connected brain subnetworks in high-functioning autism. *Human brain mapping*, 37(3):1066–1079, 2016.
- [38] Joshua I Gold and Michael N Shadlen. The neural basis of decision making. *Annu. Rev. Neurosci.*, 30:535–574, 2007.

- [39] Yulia Golland, Polina Golland, Shlomo Bentin, and Rafael Malach. Data-driven clustering reveals a fundamental subdivision of the human cortex into two global systems. *Neuropsychologia*, 46(2):540–553, 2008.
- [40] James V Haxby, Raja Parasuraman, François Lalonde, and Hisham Abboud. Superlab: General-purpose macintosh software for human experimental psychology and psychological testing. *Behavior Research Methods, Instruments, & Computers*, 25(3):400–405, 1993.
- [41] Sarah M Helfinstein, Tom Schonberg, Eliza Congdon, Katherine H Karlsgodt, Jeanette A Mumford, Fred W Sabb, Tyrone D Cannon, Edythe D London, Robert M Bilder, and Russell A Poldrack. Predicting risky choices from brain activity patterns. *Proceedings of the National Academy of Sciences*, 111(7):2470–2475, 2014.
- [42] Achim Hendriks. Software platform for human interaction experiments, 3 2015.
- [43] Joeri Hofmans and Etienne Mullet. Towards unveiling individual differences in different stages of information processing: A clustering-based approach. *Quality & Quantity*, 47(1):455–464, 2013.
- [44] Kurt Hornik. A clue for cluster ensembles. *Journal of Statistical Software*, 14(12):1–25, 2005.
- [45] Annette Horstmann, Arno Villringer, and Jane Neumann. Iowa gambling task: There is more to consider than long-term outcome. using a linear equation model to disentangle the impact of outcome and frequency of gains and losses. *Frontiers in neuroscience*, 6, 2012.
- [46] Natthakan Iam-On, Tossapon Boongoen, Simon Garrett, and Chris Price. A link-based approach to the cluster ensemble problem. *IEEE transactions on pattern analysis and machine intelligence*, 33(12):2396–2409, 2011.

- [47] Natthakan Iam-on, Simon Garrett, et al. Linkclue: A matlab package for link-based cluster ensembles. *Journal of Statistical Software*, 36(9):1–36, 2010.
- [48] Perception Research Systems Incorporated. The paradigm experiment builder, 2015.
- [49] Yu Jin and Joseph F Jaja. A high performance implementation of spectral clustering on cpu-gpu platforms. In *Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International*, pages 825–834. IEEE, 2016.
- [50] Daniel Kahneman and Amos Tversky. Intuitive prediction: Biases and corrective procedures. Technical report, DTIC Document, 1977.
- [51] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.
- [52] F Keller, M Corley, S Corley, L Konieczny, and A Todirascu. Webexp: A java toolbox for web-based psychological experiments (technical report no. hcrc/tr-99). *Human Communication Research Centre, University of Edinburgh*, 1998.
- [53] Wayne K Kirchner. Age differences in short-term retention of rapidly changing information. *Journal of experimental psychology*, 55(4):352, 1958.
- [54] Mario Kleiner, David Brainard, Denis Pelli, Allen Ingling, Richard Murray, and Christopher Broussard. What’s new in psyctoolbox-3. *Perception*, 36(14):1–1, 2007.
- [55] Wojtek J Krzanowski and YT Lai. A criterion for determining the number of groups in a data set using sum-of-squares clustering. *Biometrics*, pages 23–34, 1988.
- [56] Marco Lauriola, Irwin P Levin, and Stephanie S Hart. Common and distinct factors in decision making under ambiguity and risk: A psychometric study of individual differences. *Organizational Behavior and Human Decision Processes*, 104(2):130–149, 2007.

- [57] Marco Lauriola, Angelo Panno, Irwin P Levin, and Carl W Lejuez. Individual differences in risky decision making: A meta-analysis of sensation seeking and impulsivity with the balloon analogue risk task. *Journal of Behavioral Decision Making*, 27(1):20–36, 2014.
- [58] Richard B Lehoucq, Danny C Sorensen, and Chao Yang. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM, 1998.
- [59] CW Lejuez, Jennifer P Read, Christopher W Kahler, Jerry B Richards, Susan E Ramsey, Gregory L Stuart, David R Strong, and Richard A Brown. Evaluation of a behavioral measure of risk taking: the balloon analogue risk task (bart). *Journal of Experimental Psychology: Applied*, 8(2):75, 2002.
- [60] Irwin P Levin and Stephanie S Hart. Risk preferences in young children: Early evidence of individual differences in reaction to potential gains and losses. *Journal of Behavioral Decision Making*, 16(5):397–413, 2003.
- [61] Irwin P Levin, Joshua A Weller, Ashley A Pederson, and Lyndsay A Harshman. Age-related differences in adaptive decision making: Sensitivity to expected value in risky choice. *Judgment and Decision Making*, 2(4):225, 2007.
- [62] Hongfu Liu, Tongliang Liu, Junjie Wu, Dacheng Tao, and Yun Fu. Spectral ensemble clustering. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 715–724. ACM, 2015.
- [63] Steve Lohr. The age of big data. *New York Times*, 11(2012), 2012.
- [64] Ping Ma, Cristian I Castillo-Davis, Wenxuan Zhong, and Jun S Liu. A data-driven clustering method for time course gene expression data. *Nucleic Acids Research*, 34(4):1261–1269, 2006.

- [65] Tiago V Maia and James L McClelland. A reexamination of the evidence for the somatic marker hypothesis: What participants really know in the iowa gambling task. *Proceedings of the National Academy of Sciences of the United States of America*, 101(45):16075–16080, 2004.
- [66] Lev Manovich. Trending: The promises and the challenges of big social data. *Debates in the digital humanities*, 2:460–475, 2011.
- [67] Vivien Marx. Biology: The big challenges of big data. *Nature*, 498(7453):255–260, 2013.
- [68] Morten Mørup, Lars Kai Hansen, and Sidse M Arnfred. Erpwavelab: A toolbox for multi-channel analysis of time–frequency transformed event related potentials. *Journal of neuroscience methods*, 161(2):361–368, 2007.
- [69] ST Mueller. Pebl: The psychology experiment building language (version 0.10)[computer experiment programming language]. *Retrieved Jan*, 2009.
- [70] NeuroBehavioralSystems. Presentation, 2015.
- [71] Lee I Newman. *Decision Making under Uncertainty: Revealing, Characterizing and Modeling Individual Differences in the Iowa Gambling Task*. PhD thesis, The University of Michigan, 2009.
- [72] Jeroen Ooms. The opencpu system: Towards a universal interface for scientific computing through separation of concerns. *arXiv preprint arXiv:1406.4806*, 2014.
- [73] Tansel Özyer and Reda Alhajj. Parallel clustering of high dimensional data by integrating multi-objective genetic algorithm with divide and conquer. *Applied Intelligence*, 31(3):318–331, 2009.
- [74] Andrew D Pangborn. Scalable data clustering using gpus. Master’s thesis, Rochester Institute of Technology, 2010.
- [75] Martin P Paulus. Laboratory of biological dynamics and theoretical medicine, 2015.

- [76] Jonathan W Peirce. Psychopy—psychophysics software in python. *Journal of neuroscience methods*, 162(1):8–13, 2007.
- [77] Richard R. Plant. The black box toolkit, 2015.
- [78] Timothy J. Pleskac. Laboratory for cognitive and decision sciences, 2015.
- [79] Robi Polikar. Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, 6(3):21–45, 2006.
- [80] Girish Ravunnikutty. Ensemble k-means on modern many core hardware. 2012.
- [81] Ulf-Dietrich Reips and Christoph Neuhaus. Wextor: A web-based tool for generating and visualizing experimental designs and procedures. *Behavior Research Methods, Instruments, & Computers*, 34(2):234–240, 2002.
- [82] Valerie F Reyna. A new intuitionism: Meaning, memory, and development in fuzzy-trace theory. *Judgment and decision making*, 7(3):332, 2012.
- [83] Vincent De Sapio and Philip Kegelmeyer. Matlab cluster ensemble toolbox. Technical report, Sandia National Laboratories, 2009.
- [84] Dan R Schley and Ellen Peters. Assessing “economic value” symbolic-number mappings predict risky and riskless valuations. *Psychological science*, page 0956797613515485, 2014.
- [85] E Schneider and A Zuccoloto. E-prime 2.0 [computer software]: Pittsburg. PA: *Psychological Software Tools*, 2007.
- [86] From K Schweizer and Christine DiStefano. Principles and methods of test construction, 2016.
- [87] secretary@intestcom.org. International test commission, 09 2017.
- [88] Vikas Singh, Lopamudra Mukherjee, Jiming Peng, and Jinhui Xu. Ensemble clustering using semidefinite programming with applications. *Machine learning*, 79(1):177–200, 2010.

- [89] Zachary D Stephens, Skylar Y Lee, Faraz Faghri, Roy H Campbell, Chengxiang Zhai, Miles J Efron, Ravishankar Iyer, Michael C Schatz, Saurabh Sinha, and Gene E Robinson. Big data: astronomical or genomical? *PLoS biology*, 13(7):e1002195, 2015.
- [90] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.
- [91] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research*, 3:583–617, 2003.
- [92] J Ridley Stroop. Studies of interference in serial verbal reactions. *Journal of experimental psychology*, 18(6):643, 1935.
- [93] Julie A Suhr and John Tsanadis. Affect and personality correlates of the iowa gambling task. *Personality and Individual Differences*, 43(1):27–36, 2007.
- [94] Maolin Tang and Xin Yao. A memetic algorithm for vlsi floorplanning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(1):62–69, 2007.
- [95] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.
- [96] Claudia C von Bastian, André Locher, and Michael Rufin. Tatool: A java-based open-source programming framework for psychological studies. *Behavior research methods*, 45(1):108–115, 2013.
- [97] Hongjun Wang, Hanhuai Shan, and Arindam Banerjee. Bayesian cluster ensembles. *Statistical Analysis and Data Mining*, 4(1):54–70, 2011.

- [98] Elke U Weber, Ann-Renee Blais, and Nancy E Betz. A domain-specific risk-attitude scale: Measuring risk perceptions and risk behaviors. *Journal of behavioral decision making*, 15(4):263–290, 2002.
- [99] Serban Giuroiu Wei-keng Liao. Cuda k-means clustering.
<http://users.eecs.northwestern.edu/~wkliao/Kmeans/index.html>.
- [100] Joshua A Weller, Irwin P Levin, Baba Shiv, and Antoine Bechara. Neural correlates of adaptive decision making for risky gains and losses. *Psychological Science*, 18(11):958–964, 2007.
- [101] Thomas Whalen and Geoffrey Churchill. Decisions under uncertainty. 1971.
- [102] MC Willemssen and EJ Johnson. Mouselabweb: Performing sophisticated process tracing experiments in the participants home. In *Society for Computers in Psychology annual meeting, Minneapolis*, 2004.
- [103] Billy M Williams and Lester A Hoel. Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. *Journal of transportation engineering*, 129(6):664–672, 2003.
- [104] Paul R Wilson, Mark S Johnstone, Michael Neely, and David Boles. Dynamic storage allocation: A survey and critical review. In *Memory Management*, pages 1–116. Springer, 1995.
- [105] www.nvidia.com. Cuda c programming guide, 2017.
- [106] Lei Xu, Adam Krzyzak, and Ching Y Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on systems, man, and cybernetics*, 22(3):418–435, 1992.
- [107] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

- [108] Gui Xue, Zhonglin Lu, Irwin P Levin, Joshua A Weller, Xiangrui Li, and Antoine Bechara. Functional dissociations of risk and reward processing in the medial prefrontal cortex. *Cerebral Cortex*, 19(5):1019–1027, 2009.
- [109] Mehrdad Yazdani and Lev Manovich. Predicting social trends from non-photographic images on twitter. In *Big Data (Big Data), 2015 IEEE International Conference on*, pages 1653–1660. IEEE, 2015.
- [110] Heng Zhao, Jimin Liang, and Haihong Hu. Clustering validity based on the improved hubert\gamma statistic and the separation of clusters. In *Innovative Computing, Information and Control, 2006. ICICIC'06. First International Conference on*, volume 2, pages 539–543. IEEE, 2006.
- [111] Sanduo Zhou, Kefei Cheng, and Lijun Men. The survey of large-scale query classification. In *AIP Conference Proceedings*, volume 1834, page 040045. AIP Publishing, 2017.
- [112] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.

APPENDIX

RDMTK USER MANUAL

1. Introduction

There is no systematic structure to serve as a guideline to assist researchers to conduct RDM studies. The toolkit attempts to encourage organized thinking and implementation of

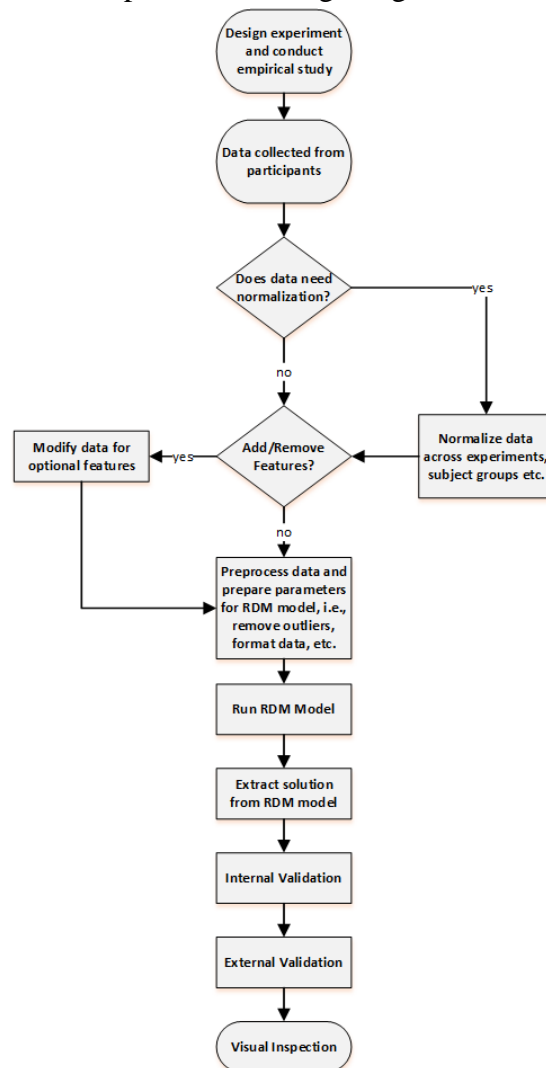


Figure 39. Typical process for analyzing RDM from experiments.

repetitive tasks. We hope RDMTk will reduce errors and optimize resource usage. Our unified toolkit standardizes methodologies for reusability and sharing. Results can be made available to a larger community base where best practices and lessons learned will be easier to integrate because RDMTk is a centralized collaborative tool that can be used for conducting studies globally. Figure 39 describes a typical flow chart for studying RDM using empirically collected data.

2. RDMTk Toolkit Accounts

Toolkit primarily supports 3 different types of accounts:

- Administrators
- Researchers
- Participants

Toolkit is primarily intended for the later two, i.e. people conducting empirical experiments and participating in them.

A. Administrators

Administrator is the person who plays vital roles in overall operation of the RDMTk application. The Administrator will be the point of contact to troubleshoot problems or issues related to deployment and accessibility.

B. Researchers

Toolkit categorizes researchers into two different types of audience:

- Researchers focusing on task design and development
- Researchers conducting empirical studies

C. Participants

People using the toolkit for participating in an experiment are categorized as participants. There are two interfaces for participants to access experiments. Participants can be access the experiment via mTurk or Qualtrics or through another online labor and data collection framework.

Use of one common interface to access the software makes using Toolkit simple. Figure 40 shows the login page, which has a link to sign up for a new participant account as well as the login screen.

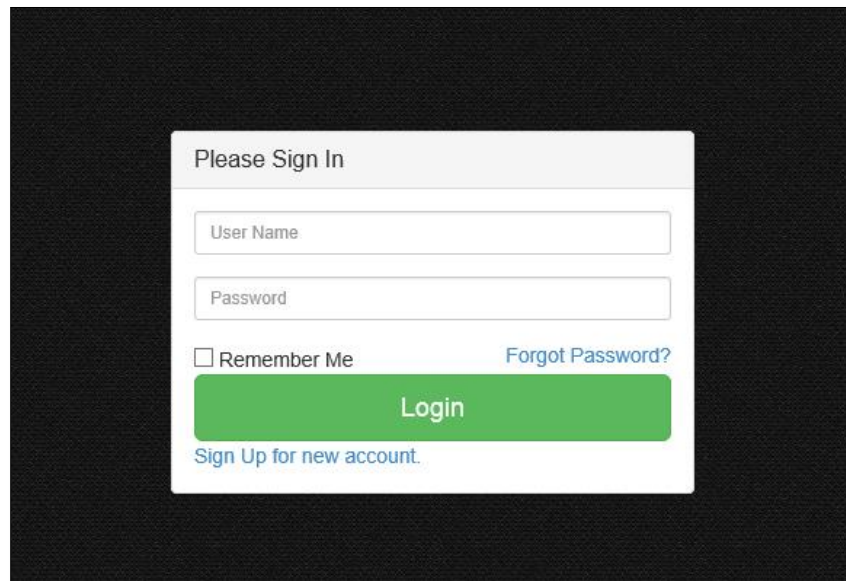


Figure 40. RDMTk login screen.

A participant will have to click on the link to sign up for a new account.

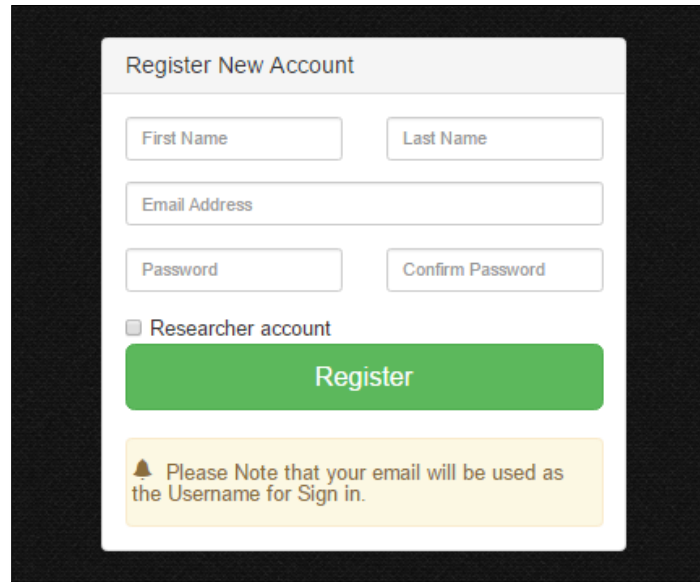
The image shows a web form titled "Register New Account". It contains several input fields: "First Name", "Last Name", "Email Address", "Password", and "Confirm Password". Below these fields is a checkbox labeled "Researcher account". A large green button labeled "Register" is positioned below the checkbox. At the bottom of the form, there is a yellow warning box with a bell icon and the text: "Please Note that your email will be used as the Username for Sign in."

Figure 41. RDMTk account registration page.

In the account registration page, one can create an account for participating in the tasks or by checking the Research Account check box to create a researcher's account. Participant accounts will only allow taking tests and provide a dashboard for them only listing available experiments that are set up. However, a researcher's account will allow one to manage different experiments, collect data and analyze results.

3. Dashboard

Dashboard has a simple user interface; it is divided into three sections: a top toolbar, left side menu items, and on the right side there is a main display section.

A. A quick start with the Researcher's account

As showing in Figure 42`, a researcher account's dashboard allows one to access advanced features to conduct psychological experiments for RDM globally. They have advanced features compared to participants, such as access to:

- Experiments
- Adding new tasks
- Data management
- Analysis Tools

Each of these features is discussed in detail during later sections.

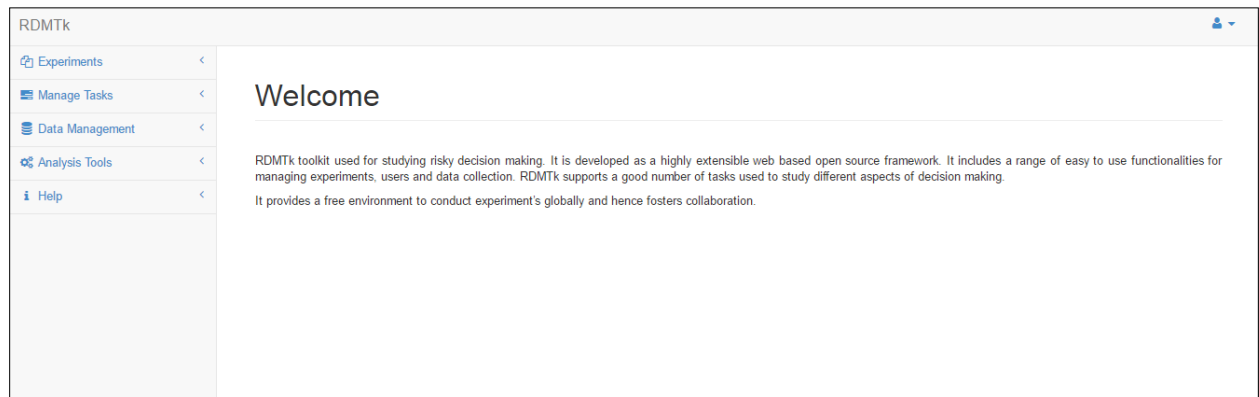


Figure 42. RDMTk researcher's dashboard view.

B. A quick start with the Participant account

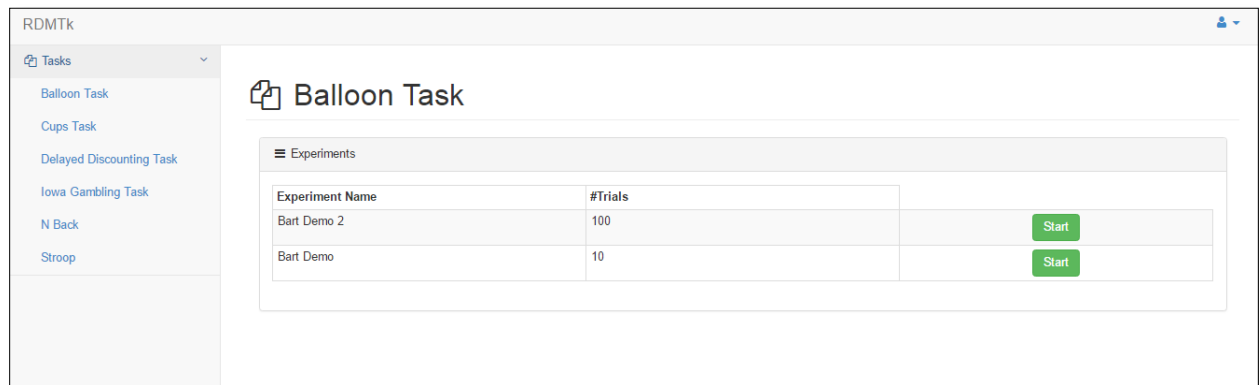


Figure 43. RDMTk participant's dashboard view.

As illustrated in Figure 43, a Participant account provides the user with a task list in the left navigation menu. On clicking a task, all of its experiments are visible to the user, among which the user may choose and participate in one. The participant can start the experiment by clicking the Start button next to the experiment name.

4. Tasks

Following are the default tasks in the toolkit:

1. Balloon Task – In this task, participant can inflate the balloon by clicking. The user can either keep inflating or collect points. If the user collects the points before the balloon explode, he gets all the points, which is proportional to the number of clicks. If the balloon explodes, the user will get no points in that trial.
2. Iowa Gambling Task - In this experiment, you will be asked to repeatedly select a card from one of four decks. You select a card by clicking the mouse on one of the decks. With each card, you can win some money, but you may also lose some. Some decks will be more profitable than others. You try to choose cards from the

most profitable decks so that your total winnings will be as high as possible. You will get 100 chances to select a card from the decks that you think will give you the highest winnings. Your total earnings and the number of cards selected will be displayed on screen. You start with \$4000.

3. Cups Task – In this task, you are going to choose cups to get the highest score. On each side of the screen, you will see a certain number of cups (2, 3, or 5). The cups will have a return value over them, either positive or negative. For each trial, you will be given the option of choosing a cup from either side by clicking on your choice. The side with multiple cups has one cup with the return value under it. The other cups have nothing under them. So your goal is to choose the right cups to maximize your score. Please read the payouts for each trial carefully.
4. N Back - Using your right hand, you will put your thumb on the spacebar. You will see a string of letters presented one at a time. If the letter you saw is the same as the letter before the last one, press the spacebar as soon as you can. For example, if you see a sequence line '...m k h k p...', then you should press the spacebar on the second 'k'. The user gets a short practice session before the experiment begins.
5. Stroop - In this task, you will be asked to name the color of the ink the words are printed in as fast as you can, ignoring the actual word that is printed in each item. You will put your left middle finger on 'D', left index finger on 'F', right index finger on 'J' and right middle finger on 'K'. The user should memorize which button to press in correspondence to different ink colors before the experiment begins.
6. Delayed Discounting Task - You will be presented with a series of choices in which you must indicate preference in a form to receive a given quantity of money. For

example, choose between "R\$1.00 now" or "R\$10.00 in a year's time." In this task, there is predefined set of sample questions that can be used by the user or the user can make his own set of questions that can be only used by him and cannot be seen by other researchers. These questions further can be modified by him only.

5. Experiments

Inside RDMTk, experiment is a generic term that is used to describe an instance that enables researchers to collect data from participants for a particular task. A task can be one among BART, CUPS, IGT, etc. Each experiment can be configured to suit individual necessities. Basic CRUD (create, read, update, delete) operations can be performed on experiments as described below.

A. Creating Experiments

Step 1: Navigate to Experiments menu.

Click on the Experiments tab in the left side menu panel.

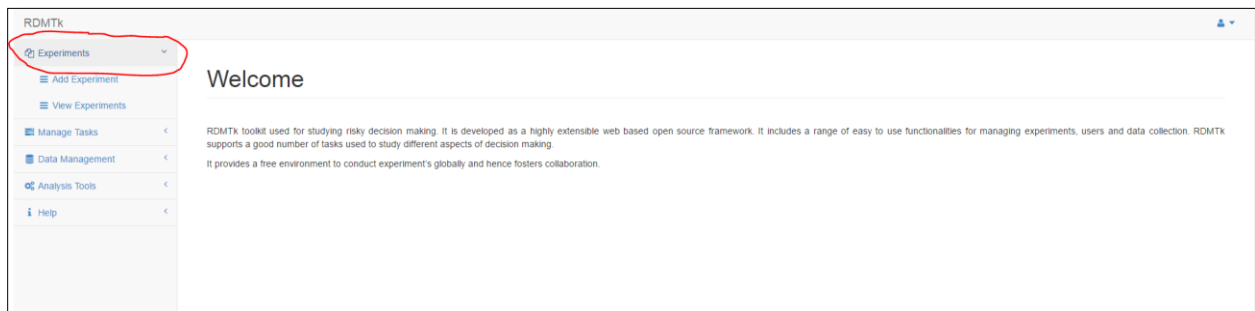


Figure 44. RDMTk menu item “experiments”.

Step 2: Click on Add Experiment.

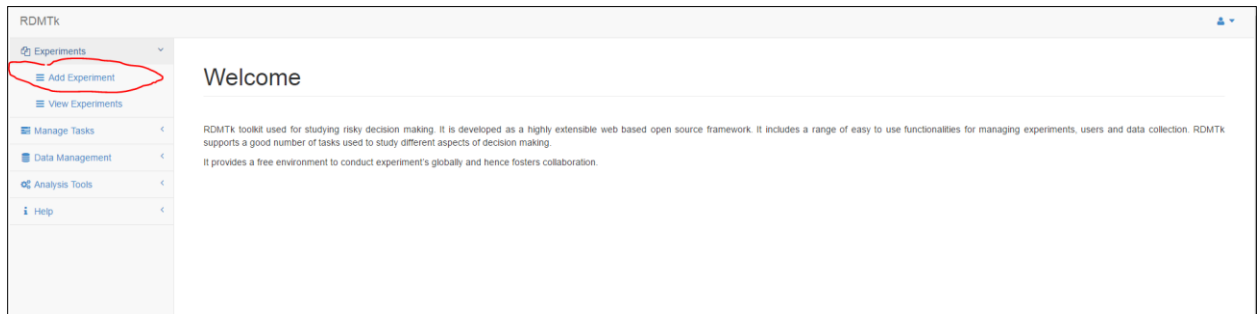


Figure 45. RDMTk menu item: experiments->add experiment.

Step 3: Fill in the form fields to create an RDMTk experiment for your study.

A screenshot of the 'Create Experiment' form in the RDMTk application. The form is titled 'Create Experiment' and contains several input fields and checkboxes. The fields are: 'Experiment Name' (text input), 'Task name' (dropdown menu with 'Please select one option'), 'Number of Trials' (text input with 'Trial#' placeholder), 'Trial End Type' (dropdown menu with 'Please select one option'), 'Confirmation Page Type' (dropdown menu with 'Please select one option'), and 'Confirmation Code' (text input with 'Confirmation Code' placeholder). There is a checkbox for 'Custom Text' with a text area below it. At the bottom, there is a section for 'Add-On Features' with a checkbox for 'Enable Mouse Tracking' and a note: 'Mouse Tracking is currently not supported with STROOP task.' At the very bottom are 'Submit' and 'Cancel' buttons.

Figure 46. RDMTk create experiment form.

Experiment Name: Researcher decides a name for the experiment and the experiment will be stored in the toolkit with this name.

Task Name: Choose a task (Balloon, IGT, etc.) from a drop-down menu.

Number of trials: The researcher selects the number of trials for the participants of the experiment being created.

Trial Duration Type: Each trial in the experiment can have a randomized outcome or predetermined. For example, in case of BART task, balloon burst points can be randomized for each participant and trial (RANDOM) or predetermined fixed balloon burst points across all participants (FIXED). Select the one appropriate to your needs.

- Random: The questions would be displayed in random order.
- Fixed: The questions would be displayed in the proper sequence.

Confirmation Page Type: Upon completion of all the trials, the participant is shown a confirmation message. This message can be either a default message which gives participants a code for their successful participation or a customized message for that particular experiment. If the researcher is integrating an experiment with mTurk or Qualtrics, it's suggested they use the default option with a confirmation code.

Default: A default text that is displayed to the participant after the completion of the task.

Confirmation Code: It is the code that would be given to the participant after his completion of the experiment.

Custom Text: If the researcher selects Confirmation Page Type as "CUSTOM_TXT" ,he can define the text to be displayed in this text box.

Add-On features: Additional features that could be included in the experiments.

Enable mouse tracking: It is used to store the data regarding the mouse locations of the participant while going through the experiment, which can be studied and analyzed further by the researcher.

Step 4: Click Submit.

B. View Experiments

To view, edit or delete an experiment, go to Experiments->View Experiments.

Experiment Name	Tool	#Trials	Trial Duration (Random / Fixed)	Confirmation Code	Confirmation Type	Confirmation Text	URL	So	Ed	De
LSSU-Male	BART	10	FIXED	123	DEFAULT		http://rdmtk.wise.cs...	So	Ed	De
bart_demo	BART	10	FIXED	111111111	DEFAULT		http://rdmtk.wise.cs...	So	Ed	De
LSSU-Female	BART	10	FIXED	111	DEFAULT		http://rdmtk.wise.cs...	So	Ed	De
cups_demo	CUPS	10	FIXED	22222222	DEFAULT		http://rdmtk.wise.cs...	So	Ed	De
igt_demo	IGT	10	FIXED	787878787	DEFAULT		http://rdmtk.wise.cs...	So	Ed	De

Figure 47. RDMTk experiments list view.

This will show you all the experiments created by the logged in user along with the options to view details, edit and delete an experiment. Each page shows 5 experiments at a time and if there are more than five, one can browse these experiments using navigation links highlighted in blue circle buttons.

C. Show Experiment

To see details for an existing experiment, click on the highlighted “So” button.

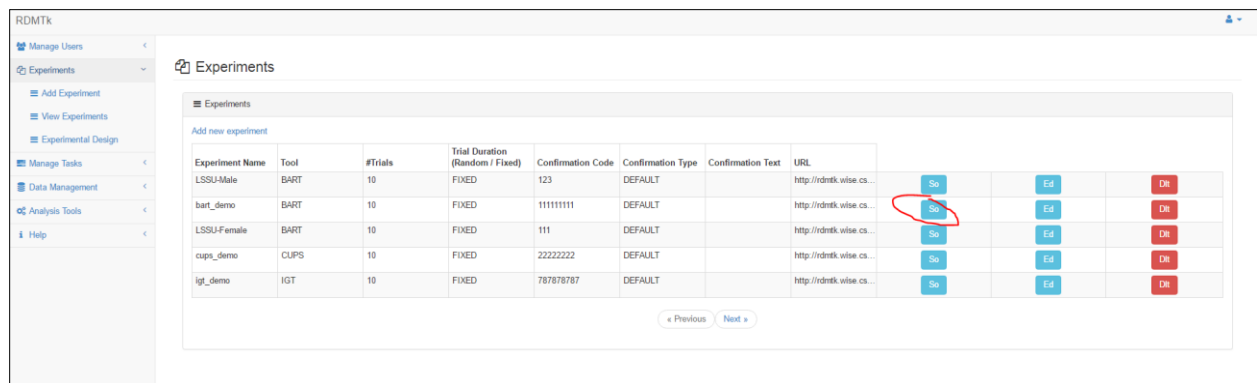


Figure 48. RDMTk “So” button to view experiment details.

The “So” button will give access to the experiment URL, which is used to integrate with mTurk and Qualtrics. It also lists other relevant details for the experiment as shown below.



Figure 49. RDMTk form listing experiment details.

D. Edit Experiment

To edit details for an existing experiment, click on the highlighted “Ed” button corresponding to it.

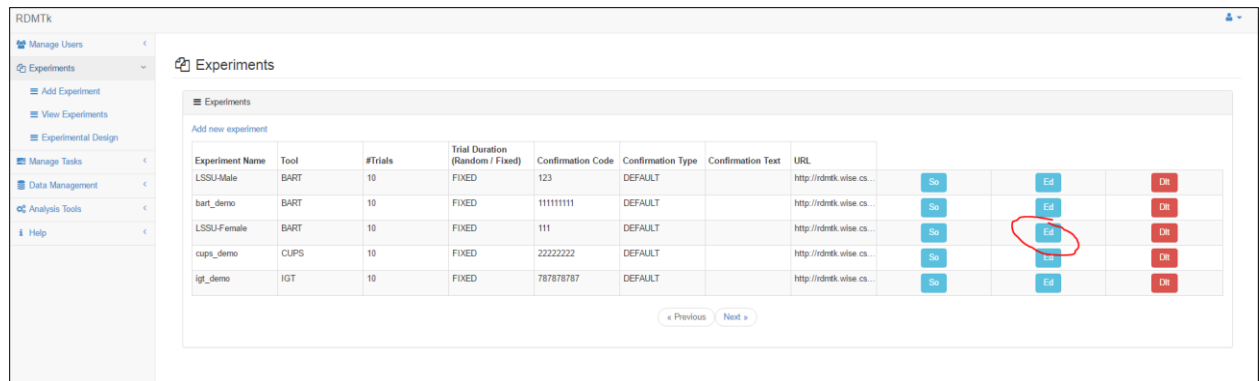


Figure 50. RDMTk “Ed” button to edit existing experiment details.

The “Ed” button will pull up a form very similar to one in “create experiment”. Here, a researcher can modify experiment parameters for an already existing experiment. It should be noted that these changes will not reflect in the data collected for participants who already took a test.

The screenshot shows the RDMTk web interface. On the left is a sidebar with navigation links: Manage Users, Experiments, Manage Tasks, Data Management, Analysis Tools, and Help. The main content area is titled 'Edit Experiment Details'. The form contains the following fields and options:

- Experiment Name:** A text input field containing 'Cups Demo'.
- Tool:** A dropdown menu with 'Cups Task' selected.
- Number of Trials:** A text input field containing '20'.
- Trial End Type:** A dropdown menu with 'Random' selected.
- Confirmation Page Type:** A dropdown menu with 'DEFAULT' selected.
- Confirmation Code:** A text input field containing '78787887'.
- Custom Text:** A checkbox labeled 'Custom Text' is checked, followed by a large text area containing the text 'Custom Text'.
- Add-On Features:** A section containing a checkbox for 'Enable Mouse Tracking'. Below it, a note states: 'Mouse Tracking is currently not supported with STROOP task.'
- Buttons:** At the bottom are two buttons: a green 'Submit' button and a red 'Cancel' button.

Figure 51. RDMTk edit experiment details form.

E. Delete Experiment

To delete an experiment permanently, click on the highlighted “Dlt” button corresponding to it.

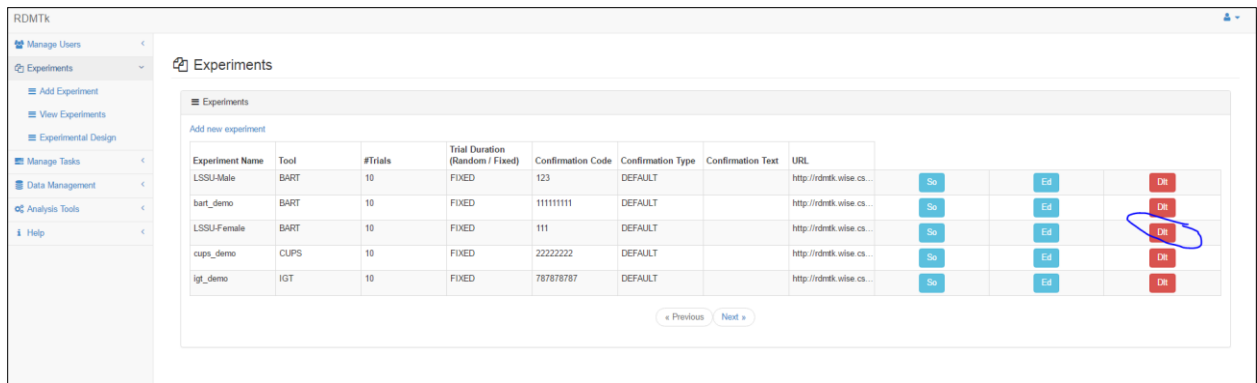


Figure 52. RDMTk “Del” button to delete an experiment.

F. Experimental Design

A researcher can create an experimental design before a study by selecting the highlighted “Experimental Design” link in the menu.

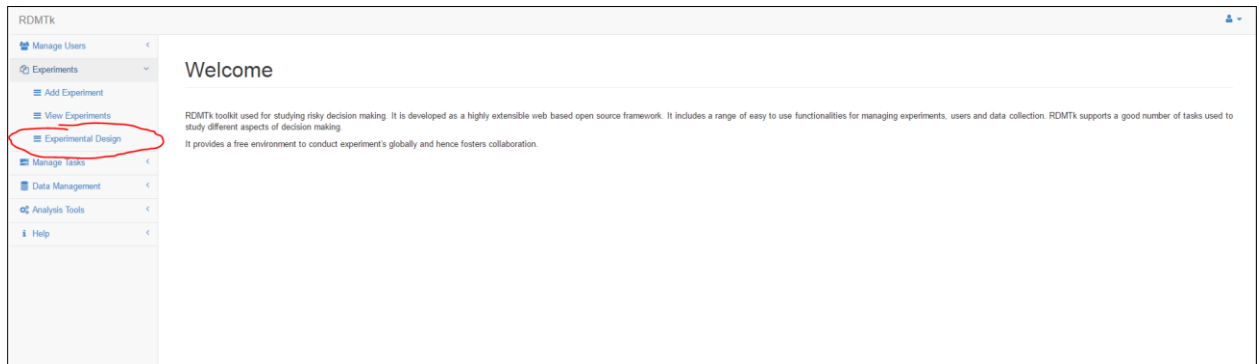


Figure 53. RDMTk menu item for “experimental design”.

Upon selecting to design a study using preexisting experiments, first currently existing study designs are listed.

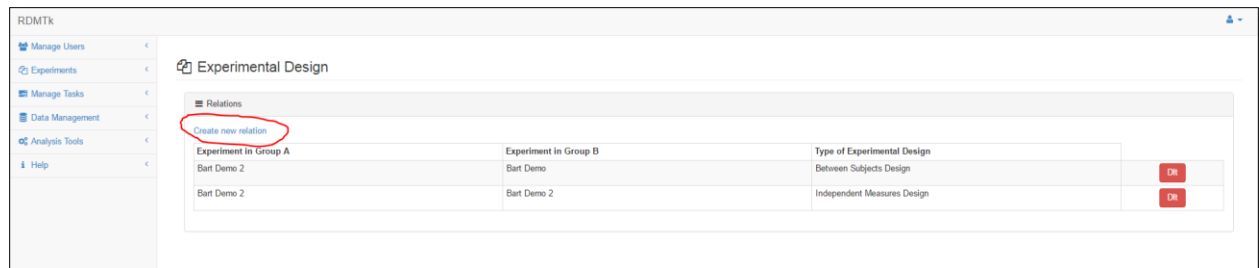


Figure 54. RDMTk existing experimental design list and link to add new relations.

Click on “Create new relation” to add a new experimental design. This link will give access to a form that will let the researcher design a study.

The screenshot shows the 'Create Experimental Relationship' form in the RDMTk interface. The form has a sidebar on the left with the same links as Figure 54. The main content area contains the following fields:

- Experiment design type:** A dropdown menu with the text 'Please select one option'.
- Task type:** A dropdown menu with the text 'Please select one option'.
- Experiment in Group A:** A dropdown menu.
- Experiment in Group B:** A dropdown menu.
- Submit** and **Cancel** buttons at the bottom.

Figure 55. RDMTk create experimental relationship form.

In the above form, a researcher can select either the “Between Subjects” or “Independent Measures” design. Existing experiments are listed in group A and B list, upon selecting a particular task type. Experiments listed in group A and B will be of the same type. For creating a “Between Subjects Design,” the experiment in “Group A” needs to be different from the one in group B. However, to create an “Independent Measure Design” study, the experiments in group A and B should be same.

6. Integration with Amazon mTurk

Researcher can recruit workers once experiment is created and experimental design finalized.

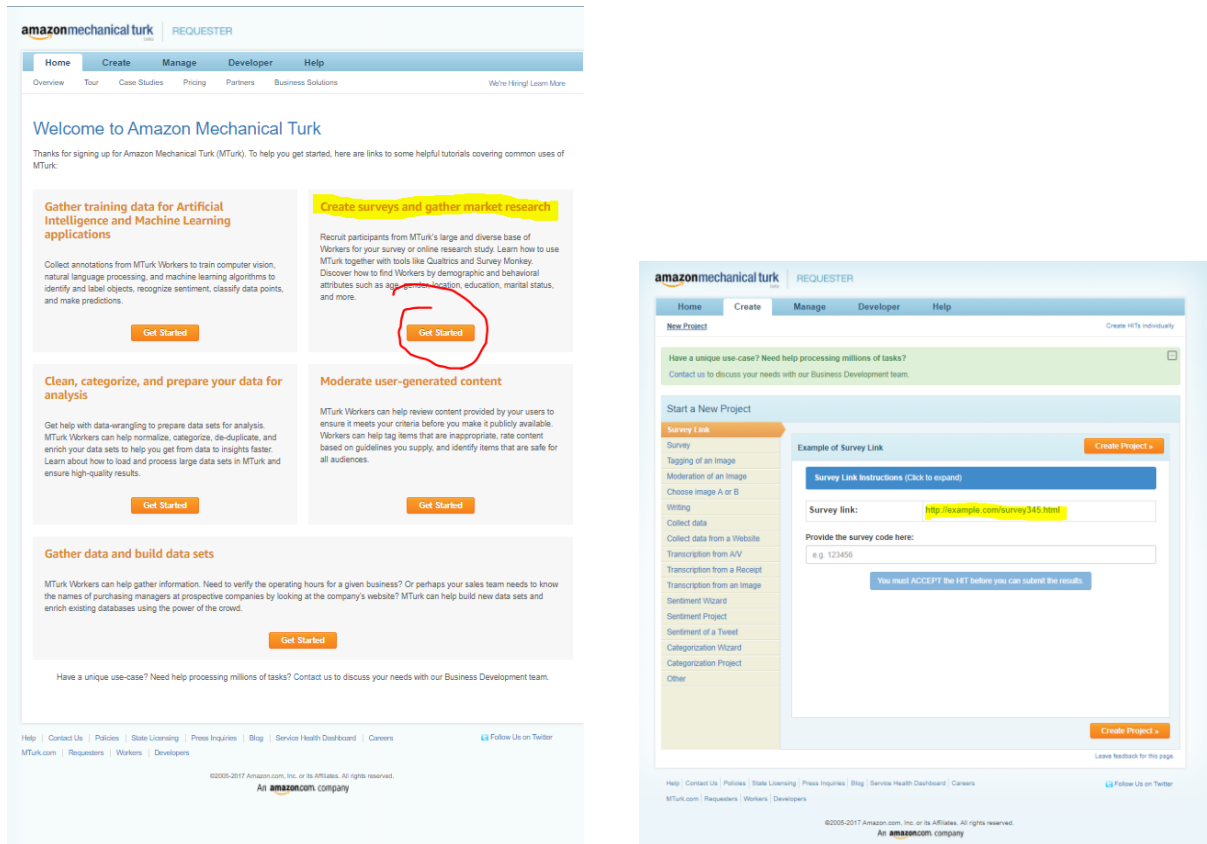


Figure 56. Amazon mTurk interface to setup RDMTk experiments.

RDMTk experiment is given as input in the highlighted box, see Figure 56 the right-side image.

7. Data Management

Data from all the individual experiments is stored in one central database. Described next are the menu options that give access and features to this data.

A. Download Experiment Results

The current version of toolkit only supports the ability to download data as an Excel workbook. To accomplish this, follow the steps below:

Step 1: Navigate to the download section: Data Management->Download Result.

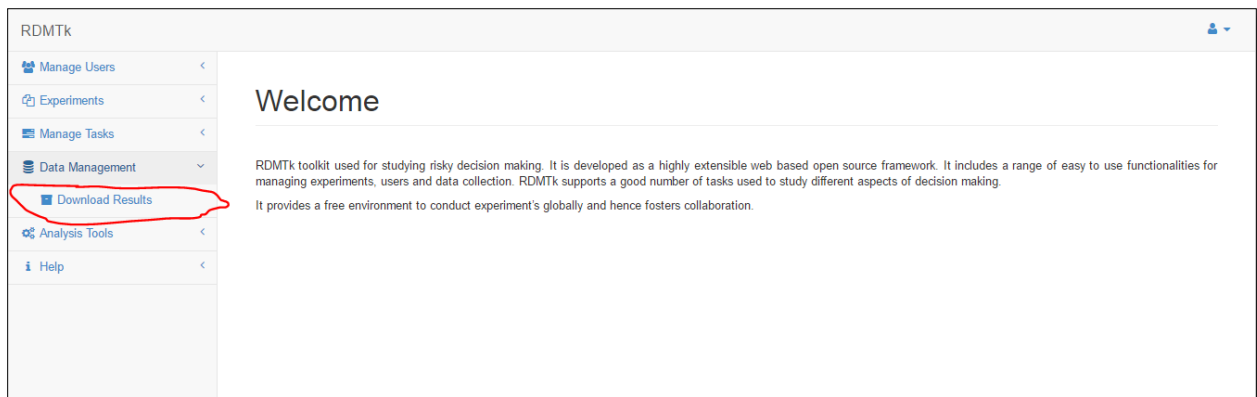


Figure 57. RDMTk download experiment results menu item.

Step 2: Choose the task name which was used to create the desired experiment.

Step 3: Choose the experiment's name from the drop-down list.

Step 4: Click on download.

Figure 58. RDMTk form to download experiment's data.

8. Adding a New Task to RDMTk

RDMTk currently supports six tasks; however, this is definitely not a complete list. To enhance toolkit's usability, we have developed an easy to use interface that allows adding new tasks to toolkit. Adding a new task is a three-step process. This option is available only to researchers. To access this feature, click on Manage Tasks->Add a New Task, as highlighted in red below.

Figure 59. RDMTk step 1 form to add new task to toolkit.

Step 1: Basic task information -

Enter the task name and task id. If any of them is already in use, then the user will get the message to enter different details.

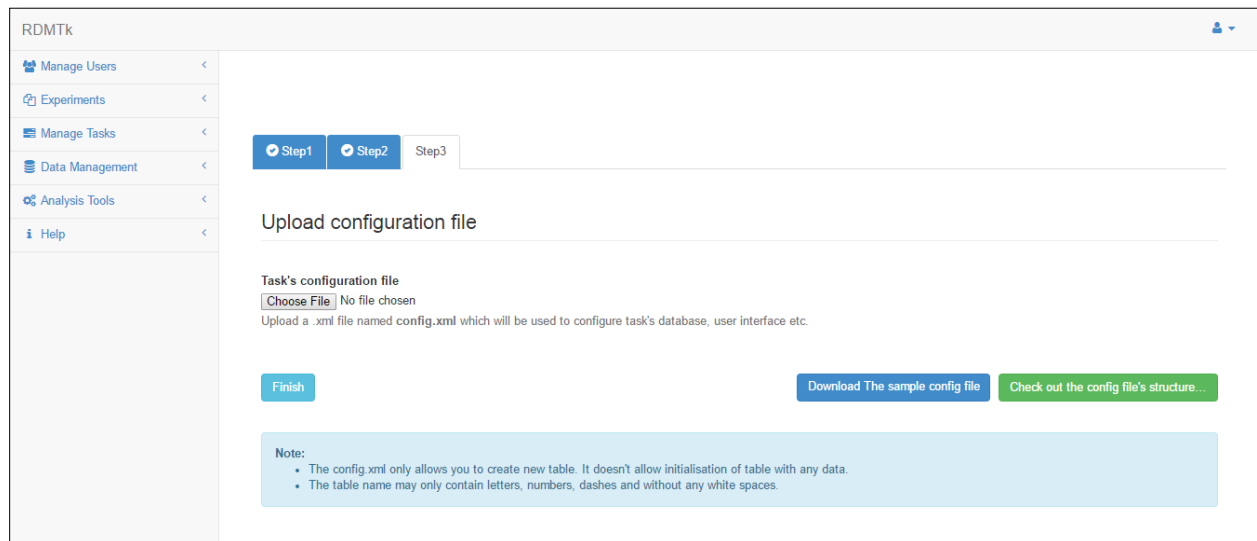
Figure 60. RDMTk step 2 form to add new task to toolkit.

Step 2: Upload the task code files -

Upload the zip file that includes all of the task files to be uploaded. A sample zip file can be downloaded for more understanding.

NOTE: It is required that the index page in the task is a file named 'task.php'.

Step 3: Upload the config.xml file which is required to create a new table in the database for the task. The table name may only contain letters, numbers, and dashes. No spaces are allowed. A sample config.xml file is also available on the page to provide better understanding.



The screenshot shows the RDMTk interface. On the left is a sidebar with navigation links: Manage Users, Experiments, Manage Tasks, Data Management, Analysis Tools, and Help. The main content area is titled 'Upload configuration file' and features a progress bar with three steps: Step 1, Step 2 (active), and Step 3. Below the progress bar, there is a section for 'Task's configuration file' with a 'Choose File' button and the text 'No file chosen'. A note below this states: 'Upload a .xml file named config.xml which will be used to configure task's database, user interface etc.' At the bottom of the form, there are three buttons: 'Finish', 'Download The sample config file', and 'Check out the config file's structure...'. A light blue note box at the bottom contains the following text: 'Note: The config.xml only allows you to create new table. It doesn't allow initialisation of table with any data. The table name may only contain letters, numbers, dashes and without any white spaces.'

Figure 61. RDMTk step 3 form to add new task to toolkit.

The config file will provide the toolkit information about the tables that it needs to create for the new task to be integrated.

The xml file needs to adhere to the following tree structure:

```
<tables>

<table name="table_name">

    <field_name type="data_type">Field Name 1 </field_name>

    <field_name type="data_type"> Field Name 2 </field_name>

    ...

    ...

</table>

<table name="table_name">

    ...

    ...

</table>

...

...

</tables>
```

The data types should be strictly chosen from the following: 'integer' , 'float' , 'string' , 'dateTime' . The script automatically creates an auto increment primary key field called 'S_no' for each new table.

9. Analysis Tools

RDMTk integrates data analysis features alongside other tools used to conduct empirical studies to reduce work load on the researcher. This feature will enable RDMTk to be eventually

grown towards being an expert system over time. Currently, it supports a feature to monitor an experiment using power test and a model to analyze data from IGT. Models implemented for IGT are to show proof of concept on the capabilities of RDMTk.

RDMTk's analysis backend is implemented to run on Amazon Web Services (AWS) to support compute-intensive operations. AWS is a collection of computing services that can scale to meet varying application resources. It is commonly referred to as cloud computing.

Its integration with advanced data analysis tools such as R statistical package will empower researchers. Even though RDMTk does not provide an extensive set of models to complement currently implemented six tasks, it is designed to incorporate new techniques and models written in the R programming language seamlessly.

10. Experiment Monitor

Experiment monitoring feature give real time status and progress tracker for a selected experimental design. This tool helps a researcher to assess the number of participants that took part in the experiment(s), and also gives cues on when to stop recording data. This tool is built based upon power analysis or power test. Traditionally, the researcher would perform power test prior to conducting an actual experiment to determine the minimum sample size (in our case, participants) that would give good statistical significance for desired properties. The implementation in RDMTk is different from contemporary approaches because, in this case, the researcher would start the experiment without any predetermined sample size. During the course of an experiment, the researcher would assess statistical significance of the data collected so far by looking at effect size and power value. Based on these two values, it can be determined whether to stop the experiment or encourage larger sample.

The experiment monitor feature can be accessed by clicking on the “Experiment Monitor” link in the menu. Upon clicking, the menu dashboard shows the screen for monitoring an experiment. It consists of two sections, where in the first part, the researcher needs to select the experiment design and task type, upon which previously configured experimental designs are listed in the third dropdown list. Selecting one of the experimental designs starts the monitoring feature as shown in the figure below.

Figure 62. RDMTk step 1 form to monitor real time experimental status and progress tracker.

11. Analysis Models for IGT

RDMTk implements the following three most commonly used analysis models for IGT as proof of concept of its capacity to analyze data fetched from the database on the R statistical platform running on Amazon Web Service (AWS). RDMTk implements a two-step approach in analyzing data collected through experiments. In the first step, the researcher can request to execute a specified model. Results after the computation are stored in a database as the model execution can be sometimes time consuming and results are not available immediately. These results are made available at a later time when the computation is completed.

To run one of these models, the researcher would click on the highlighted link in the menu and select the appropriate model. Currently implemented models are [12]:

- A. Base Model
- B. Random Model
- C. Expectancy Valence Learning (EVL) Model

Once the model is selected, all the IGT experiments are listed. The researcher has the following three options:

1. View model results – by clicking on “View” button. The button is enabled only if the model result is available to view.
2. Download model results as a csv file – by clicking on “Dwnld” button. The button is enabled only if the model result is available to download.
3. Execute model – by clicking the “Exec” button, the researcher submits a job to AWS for running the specified model on the cloud computing system. Resubmitting a job will override any results from previous runs.

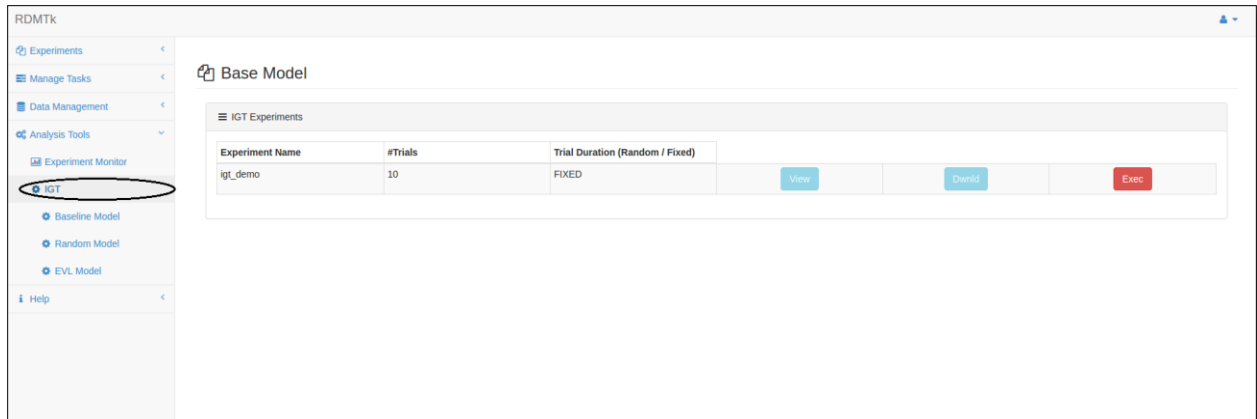


Figure 63. RDMTk menu item to access IGT analysis models.

a) Base Model Results

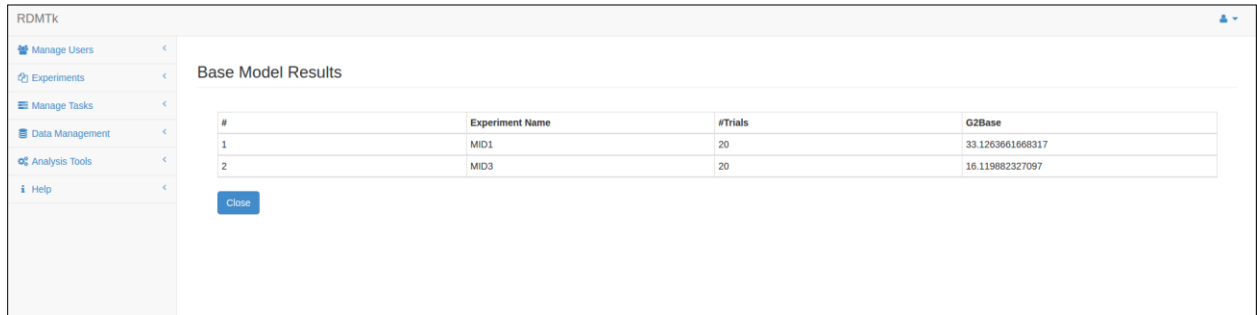
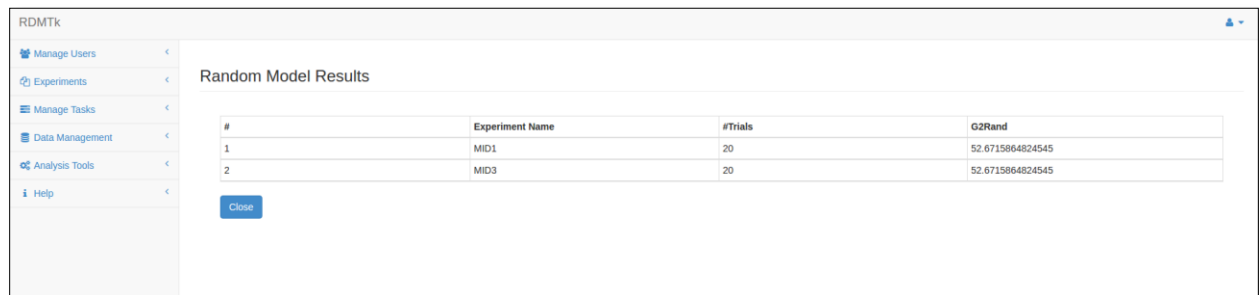


Figure 64. RDMTk base model results sample.

b) Random Model Results



The screenshot shows the RDMTk application window with a sidebar on the left containing menu items: Manage Users, Experiments, Manage Tasks, Data Management, Analysis Tools, and Help. The main content area is titled "Random Model Results" and contains a table with the following data:

#	Experiment Name	#Trials	G2Rand
1	MID1	20	52.6715864824545
2	MID3	20	52.6715864824545

Below the table is a blue "Close" button.

Figure 65. RDMTk random model results sample.

c) EVL Model Results



The screenshot shows the RDMTk application window with a sidebar on the left containing menu items: Manage Users, Experiments, Manage Tasks, Data Management, Analysis Tools, and Help. The main content area is titled "Expectancy Valance Learning Model Results" and contains a table with the following data:

#	Experiment Name	#Trials	G2EVL	Recenc	At floss	Consist
1	MID1	20	44.3220007139709	0.199436747081128	0.317717803216143	4.72698645528018
2	MID3	20	32.7948354475281	0.143701839643668	1.38375536500857e-07	4.9999999999927

Below the table is a blue "Close" button.

Figure 66. RDMTk EVL model results sample.

