Big Data Applications in Genetics and Sports

by

Nirodha Epasinghege Dona

M.Sc., University of Manitoba, Canada, 2019
B.Sc., University of Colombo, Sri Lanka, 2015

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
Department of Statistics and Actuarial Science
Faculty of Science

Declaration of Committee

Name:	**Nirodha Epasinghege Dona**
Degree:	**Doctor of Philosophy**
Thesis title:	**Big Data Applications in Genetics and Sports**
Committee:	**Chair:** Liangliang Wang Associate Professor, Statistics and Actuarial Science

Tim Swartz
Co-Supervisor
Professor, Statistics and Actuarial Science

Jinko Graham
Co-Supervisor
Professor, Statistics and Actuarial Science

Harsha Perera
Committee Member
Lecturer, Statistics and Actuarial Science

Haolun Shi
Examiner
Assistant Professor, Statistics and Actuarial Science

Andrew Swift
External Examiner
Associate Professor, Mathematical and Statistical Sciences
University of Nebraska Omaha

Abstract

This thesis consists of five distinct chapters; each considers various applications within the domain of big data. In the opening chapter, an application of genetics is presented. It discusses how to simulate exome-sequencing data for 150 families from a North American admixed population, containing at least four members affected with lymphoid cancer. These data encompass details regarding the ascertained families, along with information about single-nucleotide variants found in the exome of the affected family members.

The subsequent chapters focus on sports analytics through the lens of big data applications. In the second chapter, the expected goals concept is extended to limited overs cricket where ideas are illustrated using the economy rate statistic. The approach is based on the estimation of batting outcome probabilities given detailed data on each ball that is bowled in a match. Through the utilization of machine learning techniques, estimation of batting outcomes is carried out. From the analysis, distinctions between men's and women's T20 cricket are observed. One such finding is that there is a higher frequency of sixes occurring in the men's game than in the women's game.

In the third chapter, the focus shifts to examining the issue of pace of play in soccer. In this study, the key question revolves around whether employing a fast-paced playing style offers an advantageous strategy in the game. This is a question that remains insufficiently addressed in both soccer and hockey. The investigation is enabled through the utilization of tracking data which provides the locations of players measured at frequent intervals (i.e. 10 times per second). The chapter begins by formulating a definition of pace. In this study, we use methods of causal inference to investigate the relationship between pace in soccer and shots. The analysis reveals that maintaining a higher pace than the opponent throughout a match results in an advantage of approximately two additional shots per game.

The fourth chapter entails an assessment of the optimal locations for throw-ins in soccer. The investigation is also enabled through the utilization of tracking data which provides the locations of players measured at frequent intervals (i.e. 10 times per second). The methods for the investigation are necessarily causal since there are confounding variables that impact both the throw- in location and the result of the throw-in. A simple causal analysis indicates that on average, backwards throw-ins are beneficial and lead to an extra 2.5 shots per 100 throw-ins. We also observe that there is a benefit to long throw-ins where on average, they result in roughly 4.0 more shots per 100 throw-ins. These results are confirmed by a more complex causal analysis that relies on the spatial structure of throw-ins.

The last chapter proposes increasingly complex models based on publicly available data involving rally length in tennis. The models provide insights regarding player characteristics involving the ability to extend rallies and relates these characteristics to performance measures. The analysis highlights some important features that make a difference between winning and losing, and therefore provides feedback on how players may improve. Bayesian models are introduced where posterior estimation is carried out using Markov chain Monte Carlo methods.

Keywords: Family Studies; Exome Sequencing; Lymphoid Cancer; Ascertained Pedigrees; Sports Analytics; Player Tracking Data; Causal Inference; Machine Learning

Acknowledgements

I am profoundly grateful for the support and guidance I have received throughout this journey in completing my PhD thesis. First and foremost, I extend my heartfelt gratitude to my advisors, Dr. Tim Swartz and Dr. Jinko Graham, whose constant dedication, insightful feedback, and encouragement have been instrumental in shaping the trajectory of my research. This thesis owes its existence to their invaluable contributions.

I am also indebted to the members of my thesis committee, Dr. Andrew Swift, Dr. Haolun Shi and Dr. Harsha Perera for their invaluable insights and constructive criticism that have immensely enriched my work. I extend special gratitude to Dr. Liangliang Wang for graciously chairing my thesis defense.

I would like to express my heartfelt gratitude to my husband, Buwan Nawulla, for his enduring support, understanding, and patience throughout my journey in completing this thesis. His encouragement, love, and belief in me have been my pillars of strength, enabling me to navigate the challenges and triumphs of this academic pursuit. His sacrifices, whether big or small, have played an immeasurable role in making this achievement possible. This thesis is not just a reflection of my efforts but a testament to the unwavering partnership we share. Thank you for being my rock and my constant source of inspiration. Also, I am profoundly grateful to my parents and my sister, whose love, guidance, and sacrifices have shaped me into the person I am today. Their strong support has been the cornerstone of my journey, and I owe them an immeasurable debt of gratitude for the values they instilled in me and the opportunities they provided. Without their presence and influence, this achievement would not have been possible.

I am grateful to all the faculty members in the department of Statistics and Actuarial Science who oversaw a kid hanging around for years

I am indebted to my best friend, Dr. Puwasala Gamakumara, friends and my colleagues for their camaraderie, stimulating discussions, and shared experiences. Their support has provided both solace and motivation during the challenging phases of this thesis journey.

Lastly, this achievement is not just mine, but a culmination of the efforts of many, and I am humbled by the contributions of each individual who has been part of this journey.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

I am Nirodha Epasinghege Dona, originally from Sri Lanka. Since my early days at school, I developed a strong passion for mathematics. This was because, it consistently enhanced my logical thinking skills and the ability to engage in research. With this interest, I always believed that mathematics would shape my path in the future. As I grew older and able to plan my future, I found my enthusiasm directed towards statistics. Furthermore, I was drawn to being a researcher and to the unique position it offers as an educator to share knowledge and explore the world. I took my first step towards becoming a researcher in statistics, when I graduated from the University of Colombo with First Class honors for the B.Sc. (Hons) degree in statistics. The subjects I took during my degree enhanced my drive towards being a researcher. I was drawn towards analyzing data, finding patterns, fluctuations and the explorations behind it. This field of study broadened my view of the world and allowed me to find my path to success. The next step of my academic life was getting an M.Sc. degree and it was one of the best decisions that I have ever made in my life. I was selected as a graduate student in the Department of Statistics at the University of Manitoba. This helped me to grow up not only with the knowledge of statistics but also with lots of life experiences. During my master's studies, I was motivated to continue my studies and I applied for a PhD in statistics at Simon Fraser University. I got the opportunity of being a PhD candidate and I started my research studies in the field of genetics. I gained lots of skills and enjoyed my work in research. After two years, I wanted to shift my research interest to the field of sports analytics. As a researcher and a statistician, I believe we are capable to handle any field of studies to achieve the objectives that interest us. This thesis is a perfect example of this belief because it is a combination of these two areas of expertise.

Sport and genetics are both interesting fields that significantly impact on human ability and potential. In sports analytics, we are interested in analyzing player movements, player statistics, game outcomes and evaluate team strategies, etc.. These analyses permit to make evidence-based decisions and gain a deeper understanding of player and team dynamics. In genetics, data analyses consider identifying variations and associations between genes and traits. These analyses help to identify genetic factors associated with diseases, individual differences, and evolutionary history.

Both of these areas require the management and analysis of substantial amounts of data. In both domains, the ability to effectively handle big data is essential as it enables the extraction of valuable insights that contribute to decision-making and research progress. The applications that are discussed in this thesis, enhance the capability of handling big data and drive deeply into the analysis of significant discoveries that contribute to both fields. The following subsection explains the layout of the thesis corresponding to my PhD journey.

1.2 Organization of the thesis

Not including the introduction, this thesis is structured into five independent chapters. The underlying thread that ties these chapters together is the exploration of big data challenges. Within the domains of sports and genetics, we identify research questions and address them through the application of statistical analysis and computational techniques. The following is the sequence of chapters presented in this thesis.

Chapter 2 presents simulated exome-sequencing data for 150 families from a North American admixed (i.e. mixed ancestry) population, ascertained to contain at least four members affected with lymphoid cancer. These data include information on the ascertained families as well as single-nucleotide variants on the exome of affected family members. We provide an outline of the simulation process, accompanied by references to the relevant software scripts used in the procedure. The resulting data are useful to identify genomic patterns and disease inheritance in families with multiple disease-affected members. This chapter has been peer-reviewed and published as the following research article:

- Epasinghege Dona, N., & Graham, J. (2022). Datasets for a simulated family-based exome-sequencing study. *Data in Brief*, 42, 108311. https://doi.org/10.1016/j.dib.2022.108311

Note that this article, unlike others in this thesis, does not contain a Discussion section in accordance with the specific requirements outlined by the journal for data reports.

Chapter 3 is dedicated to quantifying the economy rate within limited-over cricket by using the concept of expected goals in soccer. In cricket, the economy rate of a bowler is defined as the average number of runs conceded by a bowler for each over bowled. This statistic proves invaluable in assessing a bowler's performance. A lower economy rate is typically regarded as advantageous, signifying that the bowler is giving away fewer runs per over, thus contributing to their effectiveness on the field. However, when calculating the economy rate, the element of luck associated with players' performance is not completely eliminated.

Therefore, we introduce the statistic, "Expected Economy Rate" (xER), which attempts to eliminate the influence of luck on players' performance within the cricket context. Through this study, our findings indicate that, xER could potentially be a more reliable statistic than the conventional economy rate and it is particularly valuable for inexperienced players who do not have a long statistical history. This chapter has been peer-reviewed and published as the following research article:

- Epasinghege Dona, N., Nguyen, R., Gill, P.S. and Swartz, T.B. (2022). Expected economy rate. *Studies of Applied Economics*, 40(1), 1-14.

Chapter 4 discusses how pace of play impacts the game of soccer. The examination of pace within soccer represents an unexplored topic. The reason for this is pace is a fundamental element of a team's playing style, which is difficult to quantify since it depends on the unique dynamics of each team. In order to measure team dynamics, we need to consider the actions of multiple players in teams whose movements are based on both time and space. By utilizing tracking data, we can evaluate player movements and capture essential team dynamics, enabling us to measure the pace of play in soccer. In this study, we quantify pace and assess whether it is an advantageous strategy that teams can adopt within the framework of their playing style. To achieve this objective, we define two quantities that measure the pace and can be calculated using the readily available player tracking data. We use causal inference methods to investigate the relationship between pace and shots. Through this study we found that maintaining a higher pace than the opponent throughout a match provides an advantage of around two additional shots per game. This chapter has been peer-reviewed and published as the following research article:

- Epasinghege Dona, N. and Swartz, T.B. (2023). A causal investigation of pace of play in soccer. *Statistica Applicata - Italian Journal of Applied Statistics*, 35(1), Article 6.

Chapter 5 considers the exploration of identifying the optimal target throw-in location in the context of soccer. Our primary focus in this study is to assess whether backward or forward throw-ins are advantageous, as well as to evaluate the benefits of utilizing long or short throw-ins. With the availability of tracking data, we investigate these ideas by using causal inference techniques. In this chapter, we begin by presenting a causal analysis pertaining to the optimal locations for throw-ins in relation to the position of the throw-in on the pitch. This is achieved through three distinct approaches. The initial two approaches are straightforward, involving the definition of binary variables for backward/forward throw-ins and short/long throw-ins. Finally, we consider both of these aspects in a comprehensive analysis that is based on a fully spatial analysis. This chapter has been reviewed, revised and resubmitted for publication:

- Epasinghege Dona, N. and Swartz, T.B.(2023). A causal investigation of throw-ins in soccer. *Under review at IMA Journal of Management Mathematics.*

In this thesis, we investigate big data application across diverse sports. In the final chapter, Chapter 6, the focus is centered around the sport of tennis. It considers how the length of rallies in tennis provides insights into player characteristics. We define three different models that estimate tennis characteristics. The initial phase involves developing a straightforward model to assess overarching tennis characteristics and how they differ between men's and women's matches, as well as between first and second serves. Then in the second model, we consider the examination of both serve and rally characteristics in relation to specific players of interest. Lastly, we expand our analysis to explore how extending rallies serves as a pivotal component of achieving success in the realm of tennis. This chapter is under preparation for submission to a journal.

Chapter 2

Datasets for a Simulated Family-Based, Exome-Sequencing Study

2.1 Introduction

We present simulated exome-sequencing data for 150 families from a North American admixed population, ascertained to contain at least four members affected with lymphoid cancer. These data include information on the ascertained families as well as single-nucleotide variants on the exome of affected family members. We provide a brief overview of the simulation steps and links to the associated software scripts. The resulting data are useful to identify genomic patterns and disease inheritance in families with multiple disease-affected members.

Specifications Table

Subject	Biostatistics
Specific subject area	Exome sequencing
Type of data How data were acquired	Plain text files and PLINK files containing simulated genetic-sequence and pedigree data in exome-sequencing of 150 families ascertained to contain at least four members affected with lymphoid cancer. Simulations done with software SLiM and with R packages SimRVPedigree and SimRVSequences.
Data format	Raw

Description of data collection	Extended pedigrees ascertained for at least four members affected with lymphoid cancer in a simulated North-American admixed population.
Data source location	Simon Fraser University
Data accessibility	Repository name: Simulated exome-sequencing data for a family study of lymphoid cancer (Zenodo). Data identification number: 10.5281/zenodo.6499208 Direct URL to data: https://zenodo.org/record/6499208 Source code repository:https://github.com/SFUStatgen /SeqFamStudy. This GitHub Repository is also archived on Zenodo at https://zenodo.org/record/6505385.

Value of the Data

- Next-generation sequencing data from families ascertained to contain multiple relatives diagnosed with the same disease can identify rare, causal DNA variants. For example, whole-exome sequencing data has been used to prioritize several rare variants for further investigation in large multi-generational pedigrees ascertained for multiple cases of bipolar disorder Cruceanu et al. (2013).

- Data from realistic simulations has the potential to advance the understanding of genomic patterns of disease inheritance, while avoiding costly recruitment of ascertained families and issues of patient confidentiality. Simulated data are easily shared, enabling researchers who analyse sequences collected from ascertained families to test and evaluate different methods.

- We present simulated exome-sequencing data in 150 families ascertained to contain four or more relatives affected with lymphoid cancer. These data should be useful for evaluating genomic patterns and disease inheritance in ascertained families, and for validation and bench-marking of statistical analysis methods in family-based sequencing.

- Our data and simulation scripts answer important calls in genetic epidemiology Riggs et al. (2021) for the reuse of existing datasets to compare statistical methods and maximize benefit from research investment and for the sharing of source code and simulated data sets in public repositories to facilitate reuse and reproducibility.

2.2 Data Description

This article describes data simulated according to the work-flow in Figure 2.1 and available in the following files:

- SLiM_output.txt - a 6.3GB text file that contains exome-wide, single-nucleotide variant (SNV) sequences for an American-admixed population of 53876 individuals generated under an American-admixture demographic model Browning et al. (2018) with the genetic simulation software SLiM Haller and Messer (2019).

- SLiM_output_chr8&9.txt - a 1.3GB text file that contains the SLiM-simulated data above for all source populations as well as the American-admixed sub-population, but only for chromosomes 8 and 9. The total number of individuals in each source population is displayed in Table 2.2.

- sample_info.txt - a 37.8KB text file giving pedigree information for the disease-affected individuals and individuals connecting them along a line of descent, for all 150 pedigrees. The file contains a total of 1247 individuals, 686 of whom are disease-affected. The remaining 561 individuals in the file connect affected individuals within a pedigree along a line of descent.

- Genotypes.zip - a 5.4MB zip-file that contains 22 chromosome-specific text files of genotypes for rare variants (RVs) on the exome. RVs are defined to be SNVs with population minor-allele frequencies less than 0.01. The RV genotypes are reported in gene-dosage format, as 0, 1 or 2 copies. These files are summarized in Table **??** below.

- SNVmaps.zip - a 2.5MB zip-file that contains 22 chromosome-specific text files giving SNV information. The contents of this zip-file is summarized in Table 2.4 below.

- familial_cRV.txt - a 2.4KB text file that contains information about the causal RVs (cRVs) in all 150 ascertained pedigrees. Three of the 150 pedigrees are “sporadic"; i.e., all affected individuals have sporadically occurring disease.

- study_peds.txt - a 552.7KB text file that contains the 150 pedigrees ascertained to contain four or more relatives affected with lymphoid cancer.

- PLINKfiles.zip - a 2.8MB zip-file that contains PLINK .fam, .bim and .bed files for all 22 of the chromosomes.

- Chromwide.Rdata - a 217MB .Rdata file that can be used as an intermediate file to save the user substantial time when running the associated RMarkdown script for the simulation. We recommend loading Chromwide.Rdata into your R work-space rather than generating it from scratch.

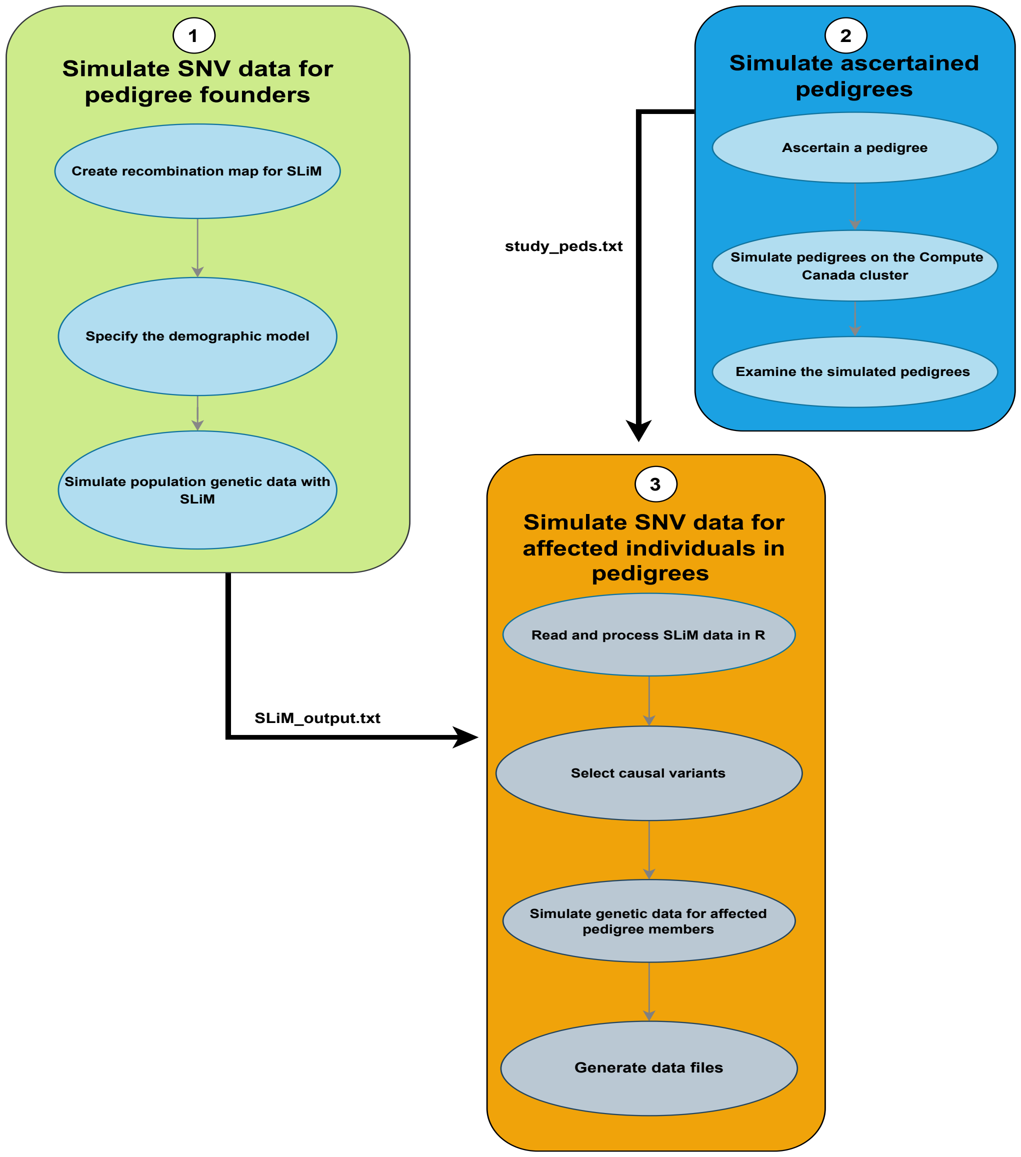

1
Simulate SNV data for pedigree founders
Create recombination map for SLiM
Specify the demographic model
Simulate population genetic data with SLiM
2
Simulate ascertained pedigrees
Ascertain a pedigree
Simulate pedigrees on the Compute Canada cluster
Examine the simulated pedigrees
study_peds.txt
3
Simulate SNV data for affected individuals in pedigrees
Read and process SLiM data in R
Select causal variants
Simulate genetic data for affected pedigree members
Generate data files
SLiM_output.txt

Figure 2.1: Work-flow for simulating the exome-sequencing data for ascertained pedigrees.

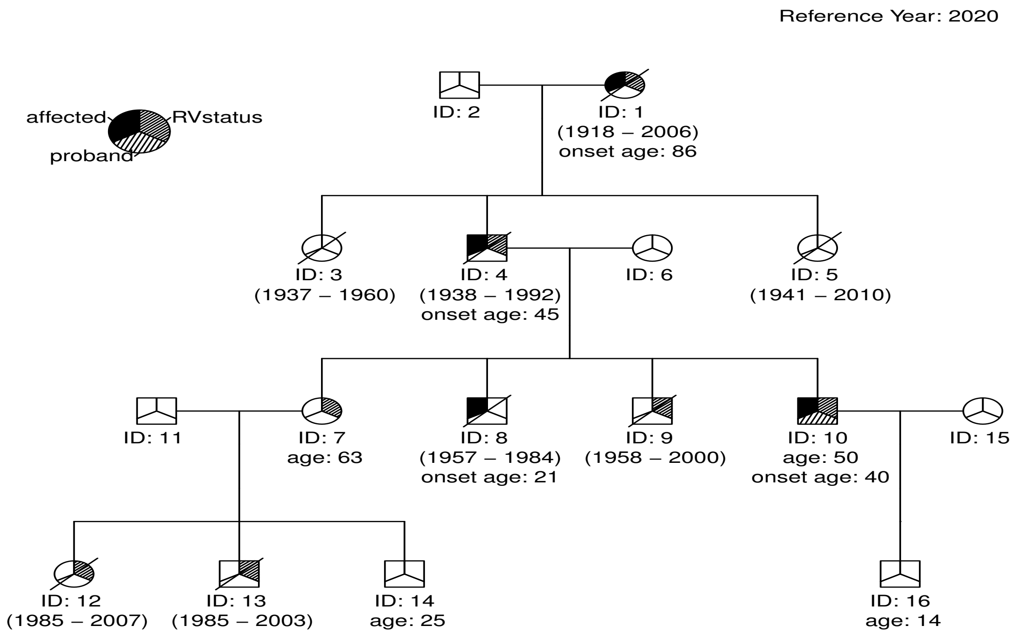


Figure 2.2: An example pedigree (pedigree 39 out of 150). The legend identifies affected individuals, the proband, and the cRV status of the individuals. Disease-affected individuals have solid shading in the upper-left third of their symbol (IDs 1, 4, 8 and 10). The proband (ID 10) has shading in the lower portion of their symbol. Individuals carrying a causal genetic variant (IDs 1, 4, 7, 9, 10, 12 and 13) have shading in the upper-right portion of their symbol. The birth year and the death year of dead individuals are displayed in parentheses. The age of the individuals who are alive at the end of the reference year of 2020 displays under their symbol. Any individual with disease onset before the end of the reference year has a disease-onset year given under their symbol. Following standard practice in medical genetics, individuals who have died as of the reference year have slashes through their symbols. The age of the individuals who are alive at the end of the reference year displays under their symbol. Any individual with disease onset before the end of the reference year has a disease-onset year given under their symbol.

Figure 2.2 shows an example family (family ID 39 out of 150) from study_peds.txt. This family contains 16 individuals across 4 generations, 4 of whom are affected (IDs 1, 4, 8 and 10). Across all 150 families, the family size varies between 8 and 207 individuals (see Figure 2.3), the number of affected members varies between 4 and 8 individuals (see Figure 2.4) and the number of generations varies between 1 and 7 (see Figure 2.6). Further, as the number of generations increases, so does the pedigree size (see Figure 2.7). Note that, due to high computational cost, we did these calculations only for a single set of 150 pedigrees. However, we can increase the number of replications in this simulation and a detailed description about how users can generate more pedigrees can be found in our second supplementary material.

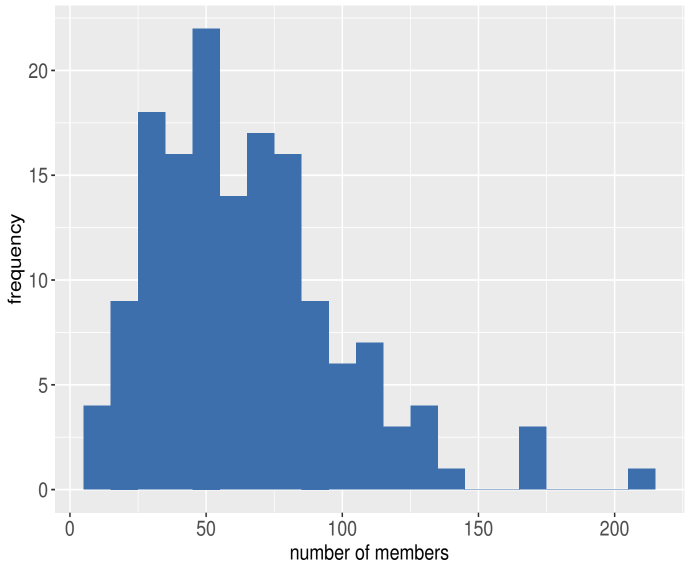


Figure 2.3: Distribution of pedigree size.

Figure 2.4 suggests that the size of the pedigrees is uncorrelated with the number of affected members. Also, as expected and shown in Figure 2.5, so-called “sporadic" pedigrees, in which all affected members have sporadically occurring disease (family IDs 72, 95, 103), tend to be larger than non-sporadic pedigrees.

The above data files have been created by simulation, performed in three major steps as outlined in Figure 2.1 and implemented in the RMarkdown files provided as supplementary materials. These supplementary materials are contained in Appendix A, Appendix B Appendix C and Appendix D. The RMarkdown files to simulate the data are:

- Supplementary Material 1-A: Simulate SNV sequence data for pedigree founders. Contains the code and methods to get the `SLiM_output.txt` file.

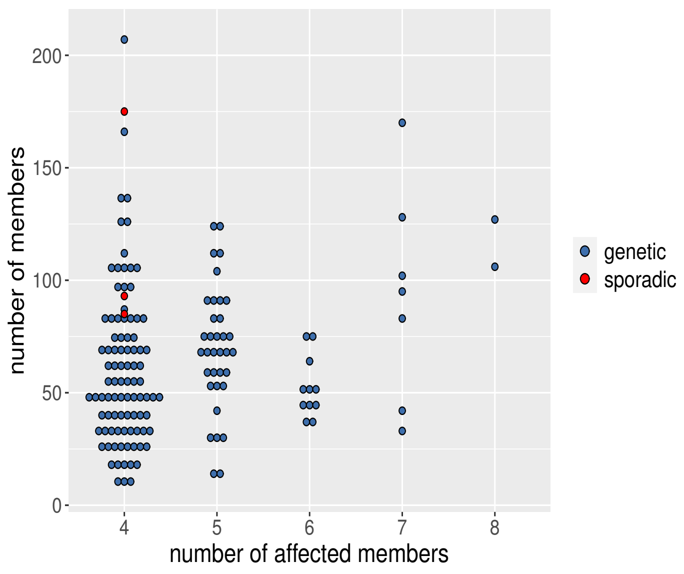


Figure 2.4: Distribution of pedigree size by number of affected members.

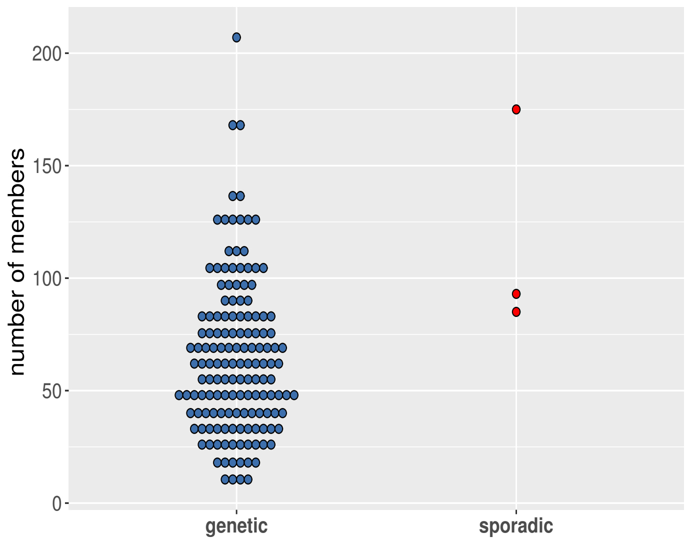


Figure 2.5: Distribution of pedigree size in pedigrees with genetic and sporadically occurring disease.

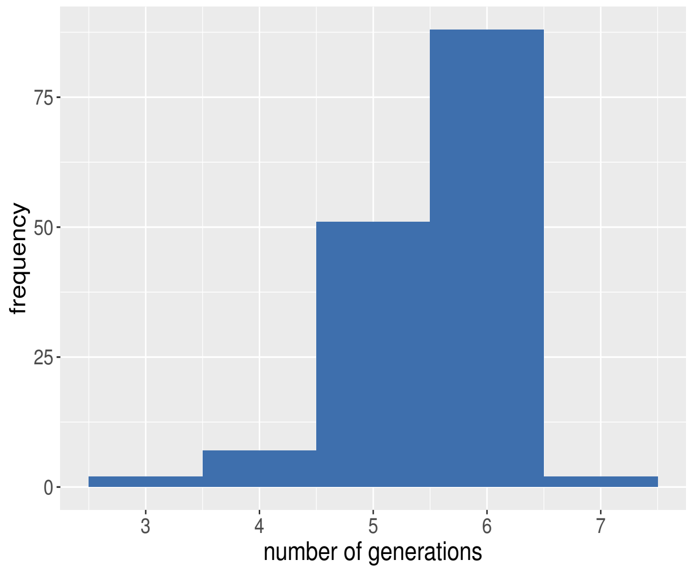


Figure 2.6: Distribution of number of generations in pedigrees.

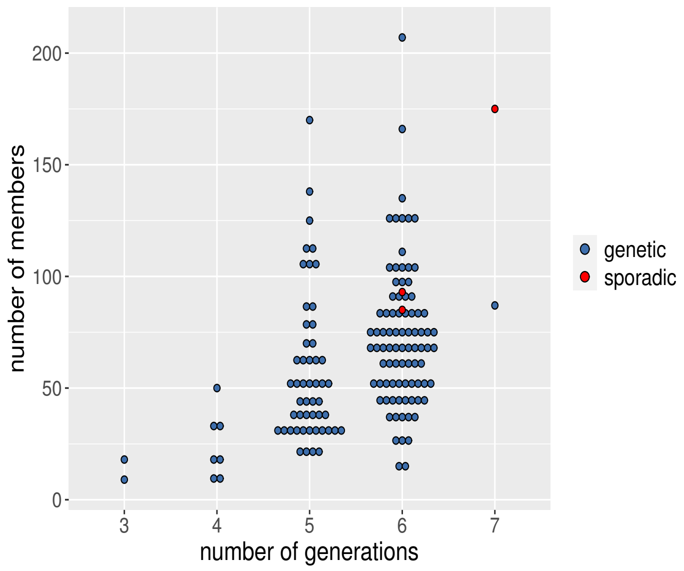


Figure 2.7: Distribution of pedigree size by the number of generations.

- Supplementary Material 1-B: Combining Source Populations with the American-Admixed Population in SLiM. Contains the code and methods to get the `SLiM_output_chr8&9.txt` file.
- Supplementary Material 2: Simulate ascertained pedigrees. Contains the code and methods to get the `study_peds.txt` file.
- Supplementary Material 3: Simulate SNV data for affected individuals in pedigrees. Contains the code and methods to get the data files `sample_info.txt`, `Genotypes.zip`, `SNVmaps.zip` `familial_cRV.txt` and `PLINKfiles.zip`. Also contains a detailed description of the formats of these files.

Population	**size**
African	14475
European	35815
Asian	48765
Admix	53876
Total	**152931**

Table 2.2: Source Population sizes.

Chromosome	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
No. of RVs	19393	14132	11634	8096	9681	10137	9653	7107
Chromosome	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
No. of RVs	8061	8356	10629	10150	4266	6108	6977	7536
Chromosome	**17**	**18**	**19**	**20**	**21**	**22**		
No. of RVs	10168	3474	10298	5007	2185	4305		

Table 2.3: Number of rare variants (RVs) in each genotype text file.

Chromosome	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
Pathway RVs	4	90	12	15	24	38	23	50	0	28	5	0	0
No. of cRVs	1	18	4	3	7	7	5	8	0	6	2	0	0
Chromosome	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**				
Pathway RVs	0	0	8	19	30	6	0	37	18				
No. of cRVs	0	0	3	3	5	0	0	3	5				

Table 2.4: Number of rare variants in apoptosis sub-pathway and number of causal rare variants (cRVs) in SNV map files.

2.3 Experimental Design, Materials and Methods

To acquire these data, we simulated exome sequences in pedigrees ascertained to have four or more relatives affected with lymphoid cancer, according to the work-flow summarized in

Figure 2.1. The simulation involved three major steps which we describe in the subsections below.

2.3.1 Simulate SNV data for pedigree founders

Simulating the exome sequences in ascertained pedigrees requires SNV sequences for pedigree founders. We assume these founders are sampled randomly from an American-Admixed population, which we simulate with the evolutionary simulation package SLiM Haller and Messer (2019). To mimic exome sequencing, we simulate genome-wide sequences of exons only. A detailed account of this part of the simulation is provided in Supplementary Material 1-A.

As shown in Figure 2.1, simulating the founder sequences involves three steps. First, we provide a recombination map to SLiM giving the exon positions in chromosomes. We use the `create_SlimMap()` function in the `SimRVSequence` Nieuwoudt et al. (2020) R package for this task.

Second, we specify a demographic model for the American-Admixed population Browning et al. (2018) because our work is motivated by a family-based exome-sequencing study of lymphoid cancer in a North American population Jones et al. (2017). The American- Admixture demographic model of Browning et al. (2018) is compiled in stdpopsim, a standard library of population-genetic simulation models Adrion et al. (2020).

Finally, we use the Compute Canada cluster (http://www.computecanada.ca) to avoid the computational cost of running this large simulation on a personal computer. The batch scripts that we ran on the cluster can be found in the first supplementary materials. The resulting SNV data for pedigree founders is used as an input to simulate the SNV data for affected pedigree members in the third step of our work flow, described below.

2.3.2 Simulate ascertained pedigrees

The second step of our workflow is to simulate ascertained pedigrees with disease-affected relatives. We simulated 150 pedigrees ascertained to contain four or more relatives affected with lymphoid cancer using the `SimRVPedigree` R package Nieuwoudt et al. (2018). A complete description of this step of the workflow can be found in our Supplementary Material 2. As the simulation is time-consuming, we use the Compute Canada cluster. The final outcome of this step is a set of ascertained pedigrees in which to generate exome-sequencing data for the third and final step of the workflow, described next.

2.3.3 Simulate SNV data for affected individuals in pedigrees

To simulate exome-sequencing data for the affected members of ascertained pedigrees, we use the outcomes from the previous two steps of the workflow, together with the gene-dropping functions in the `SimRVSequences` R package Nieuwoudt et al. (2020). These functions require sparse matrices of SNV sequences but, unfortunately, the population size and number of SNVs is too large to accommodate within a single sparse matrix. Therefore, we create the sparse matrices chromosome-by-chromosome. A complete description of how we implemented this final step of our workflow can be found in Supplementary Material 3.

As shown in the Figure 2.1, we divide this step into four sections. First we read the SLiM-simulated sequencing data into R and process them chromosome-by-chromosome. Then we select the rare variants (RVs) in genes on an apoptosis sub-pathway as candidates for causal rare variants (cRVs), as described in Nieuwoudt et al. (2020). Specifically, we use the apoptosis sub-pathway centered about the TNFSF10 gene in the UCSC Genome Browser's Gene Interaction Tool. This sub-pathway contains 23 genes: *DAP3*, *CFLAR*, *CASP10*, *CASP8*, *TNFSF10*, *CASP3*, *MAP3K1*, *TNF TNFRSF10A*, *FOXO3*, *IFNGR1*, *CYCS*, *TNFRSF10B*, *TNFRSF11B*, *FAS*, *FADD*, *TRADD*, *TP53*, *RALBP1*, *BCL2*, *BAX*, *IFNAR1*, *IFNGR2* and *BID*. These genes and their chromosomal locations are provided in the `hg_apopPath` data set of the `SimRVSequences` package. Among the SNVs in this pathway, cRVs are selected from population singletons, on the basis of their absolute selection coefficients, until the cumulative probability of a sequence carrying any cRV is 0.001.

We then simulate exome sequences for the affected individuals in the 150 ascertained pedigrees by using the `SimRVSequences` R package. The `sim_RVstudy()` function of this package simulates genetic sequence data in pedigrees, but expects only a single population database of sequences as an argument in the form of a sparse matrix of SNV haplotypes and an associated mutation data frame. Unfortunately, we cannot use `sim_RVstudy()` without modification because the number of individuals and RVs in our American-admixed population is too large. The fundamental problem is that the population sequences of RVs cannot be contained in a single sparse matrix without exceeding the memory capacity of R. We therefore modify `sim_RVstudy()` and various supporting functions in the `SimRVSequences` package to handle chromosome-specific databases as described in the Supplementary Material 3.

Finally, we deliver our data in human-readable flat-file formats. We use the simulated data to create a `.sam` file containing information about genotyped individuals in the ascertained pedigrees, chromosome-specific `.geno` files containing RV genotypes and chromosome-specific `.var` files containing information about RVs. The data files are in flat-file format and also in PLINK file formats Purcell et al. (2007). These text and PLINK file formats are discussed in the Supplementary Material 3.

Chapter 3

Expected Economy Rate

3.1 Introduction

Expected goals (xG) is a concept that has gained rapid adoption in professional sport, particularly in soccer. The statistic xG attempts to quantify what is most likely to have happened in a match given the opportunities that occurred during the match. For example, imagine that Team A drew with Team B with the scoreline 1-1 but the expected goals for the match was 3.8 to 0.5 in favour of Team A. In this case, one would conclude that Team A outplayed Team B, and that Team B was fortunate to achieve the draw. Therefore, xG provides a measure of dominance in matches, and attempts to remove what might be described as the "luck" element of sport. Not only does xG describe match dominance, but xG is predictive of future results.

Although xG is intuitive, its calculation is viewed as a black-box procedure from the point of view of the general public. For this reason, and for the technical underpinnings used in the calculation of expected goals, xG falls under the category of advanced analytics. Further, the "black-boxness" of xG is magnified since there are many implementations of xG, and these implementations are typically proprietary. In soccer and hockey, the basic idea behind xG is that there are scoring opportunities on the field and the rink, respectively. The scoring opportunities have associated goal scoring probabilities where goals resulting from shots that are further away and taken from more extreme angles are less probable. These probabilites are summed over the opportunities for each team, leading to the team's xG. The literature that does exist concerning xG is mostly found on blog sites, Twitter feeds and conference proceedings. Some of the more detailed contributions related to xG include Pollard, Ensum and Taylor (2004), Rudd (2011), Macdonald (2012), Decroos et al. (2018) and Fernández, Bornn, and Cervonne (2019).

In this chapter, we introduce the the concept of xG to limited overs cricket in the context of the economy rate statistic. In limited overs cricket, the economy rate for a bowler is defined as the average number of runs conceded per over where there are six balls per over. The average is typically calculated over a match, a series, a year or a career. For illustration, we consider economy rate based on a match in which case the economy rate for a bowler is defined as 6*(the number of runs conceded in a match) divided by the number of balls bowled by the bowler in the match. Smaller values of the economy rate are indicative of good bowling.

The development of xER (expected economy rate) is conceptually simple. Based on the characteristics of a ball that has been bowled, we estimate the probabilities of the 8 batting outcomes: wicket, 0 runs, 1 run, 2 runs, 3 runs, 4 runs, 5 runs and 6 runs where we note that there is negligible probability of scoring 3 runs and 5 runs. Using obvious notation, we define the *expected economy rate* in a match for a bowler as

$$\text{xER} = (6/M) \sum_{i=1}^{M} \left(e_i + p_i(1) + 2p_i(2) + 3p_i(3) + 4p_i(4) + 5p_i(5) + 6p_i(6)\right) \tag{3.1}$$

where i corresponds to the ball number, e_i is the actual number of extras accumulated on the ith ball and M is the number of balls bowled by the bowler in the match. Analogous to xG, xER represents the expected economy rate performance of the bowler in the match. Whereas the p_i's in (3.1) have been estimated, we do not estimate extras. We consider the observed extras e_i as penalty terms that are directly attributable to the bowler and are added to the expected economy rate formula. There is no luck aspect associated with extras; bad balls are simply bad balls.

Now, xER will only be informative and useful provided that the estimated probabilities p_i in (3.1) are realistic. To motivate the approach, suppose that $p_i(4) = 0.5$ and the actual result of the bowled ball was a wicket. What might have happened in this scenario is that the bowler's delivery ought to have been exploited by the batsmen. Based on the characteristics of how the ball was bowled, the probability of scoring four runs was high. With a ball of this type, the bowler is typically punished. Instead, the bowler was "lucky" in the sense that a wicket occurred. For example, perhaps a fabulous catch was made. Therefore xER is an attempt to represent what the bowler would have achieved under ordinary circumstances given the performance. The actual achievement is subject to both performance and the luck/stochastic element of sport.

The utility of the xER statistic (3.1) occurs when a bowler's actual economy rate differs considerably from their xER statistic. For example, suppose that there is a relatively young bowler whose actual economy rate is 8.4 but their xER = 6.6. This would signal that the bowler has been unlucky, and that this is a player for whom team selectors should give attention. This may be a promising bowler.

The elimination of "luck" from performance is a driving force in this research. Whereas luck does not seem to have been addressed in cricket, luck has been investigated in other sports analytics research. For example, luck has been explored in soccer (Sarkar and Kamath 2022), golf (Connolly and Rendleman 2008), baseball (Bailey, Loeppky and Swartz 2020), and hockey (Weissbock 2014).

In this chapter the probability estimates p_i in (3.1) are based on detailed ball-by-ball data. Ball-by-ball data have been utilized in various research initiatives in cricket (Swartz 2017). In these projects, ball-by-ball results and covariates have been parsed from match commentaries provided by www.cricinfo.com. However, the ball-by-ball data used in these investigations are not sufficiently detailed for the current investigation. In this project, Cricket Australia has provided us with even more detailed ball-by-ball data from the Twenty20 (T20) format. However, such data are not widely available and do not currently exist for competitions outside of the Australian context. For this reason, the work presented here is explored as a proof of concept. We demonstrate that the approach is feasible and informative, and is something that can be fully developed when detailed ball-by-ball data become widespread. With the advent of player tracking data and analyses across major sports (Gudmundsson and Horton 2017), we expect that the availability of detailed data in cricket is only a matter of time.

Clearly, xER can be extended to applications over series, matches, years and careers. Also, it is clear that other standard bowling statistics such as bowling average and bowling strike rate have xG adaptations. With respect to batting, we might similarly define xG statistics corresponding to batting average and the batting strike rate. Many statistics have been developed for the sport of cricket; see Swartz (2017) for a review of various measures of player evaluation.

In Section 3.2, we describe the detailed data that have been provided by Cricket Australia. Subjective decisions on retaining and excluding variables are part of the feature identification process. Whereas the focus of sports analytics research has typically involved "big" sports that involve male participation, a feature of this work is that we also analyze women's T20 data. This has the added benefit of assessing differences in how T20 is played between the men's and women's game. In Section 3.3, we provide a description of the random forest approach used to estimate the probabilities in (3.1). The procedure falls under the topic of supervised learning. In Section 3.4, we investigate the quality of estimation in various ways. For example, we examine the overall rate of correct predictions, we examine the related confusion matrices and we qualitatively assess the variables of importance. We also compare our random forest predictions against predictions made by two other methods using Brier scores. We observe that the proposed estimation technique is superior to the other methods. In Section 3.5, we apply the approach to datasets involving T20 cricket matches. Some insights are obtained which demonstrate the potential of utilizing xER in player evaluation. Finally, a short discussion is provided in Section 3.6.

3.2 Data

Detailed ball-by-ball data have been collected by Cricket Australia over the years 2007 through 2019. The data correspond to international matches involving the Australian national teams (men and women) in Test, ODI and T20 formats. Data have also been collected for Australian domestic matches such as the Big Bash competition. For illustration, we have restricted our attention to T20 matches. There are 532 matches for the men and 725 matches for the women. For a given match, there are an astounding 360 variables collected on every ball that is bowled. The data were coded manually by data entry specialists who watch video broadcasts of matches. It is believed that the data have a high level of accuracy. After some minor data management, our dataset of T20 matches consist of 123,067 bowled balls (men) and 166,898 bowled balls (women) that occurred during the first and second innings.

In our investigation of xER, we are interested in bowling performance. Hence we seek variables that relate bowling performance to the batting outcome. The beauty of sport is that domain knowledge is often high, and feature selection may be assisted by this subjective knowledge. In Table 3.1, we list $k = 25$ covariates (features) that we believe are predictive of the batting outcome. We have taken the point of view to err on the generous side and include all variables that may have a chance of improving supervised learning predictions; machine learning algorithms have been designed to detect the most important variables. The potential variables that we have omitted from Table 3.1 include many redundant variables such as the result of the match and cumulative totals as the match proceeds. We have omitted match identifiers (e.g. match Id and season Id) which are unrelated to the outcome of the particular ball that has been bowled. We have also omitted many variables that are deemed irrelevant to the batting outcome such as non-striker characteristics, fielder characteristics and the occurrence of injuries.

Note that the variables in Table 3.1 are categorized as either bowling variables or batting variables. A bowling variable is one that is entirely due to the bowler or the conditions of the match. For example, the *ball speed* variable is a bowling variable. As another example, the *wickets lost* variable is a bowling variable. A batting variable concerns something that the batsman did. For example, *batsman handedness* is a bowling variable (because this is a condition of the match) whereas *hit angle* of the batted ball is a batting variable.

Feature (Data Details)	Bowling Variable	Batting Variable	Percent Missing Observations
bowling team venue (home, neutral, road)	Y	N	0.00
innings (1, 2)	Y	N	0.00
time of day (hour using 24 hour clock)	Y	N	0.00
batsman handedness (L, R)	Y	N	0.00
bowler handedness (L, R)	Y	N	0.00
bowler's style (pace, spin)	Y	N	0.00
bowler's spell (1, 2, 3, 4)	Y	N	0.00
bowler's speed (fast, medium, slow)	Y	N	0.00
powerplay (Y, N)	Y	N	0.00
over (1, ..., 20)	Y	N	0.00
ball in over (1, ..., 6)	Y	N	0.00
wickets lost (0, ..., 9)	Y	N	0.00
resources remaining (0.0 to 100.0)	Y	N	0.00
deficit (integer)	Y	N	0.00
ball speed (continuous in km/hr)	Y	N	69%
ball rpm (continuous)	Y	N	99%
pitchx (horizontal ball landing rel to wicket)	Y	N	0.00
pitchy (vertical ball landing rel to wicket)	Y	N	0.00
batsmanx (horizontal ball landing rel to batsman)	Y	N	0.00
batsmany (vertical ball landing rel to batsman)	Y	N	0.00
hit angle (angle ball hit by batsman)	N	Y	0.00
hit length (distance ball stopped after hit)	N	Y	0.00
temperature (cold, moderate, hot)	Y	N	2%
humidity (dry, moderate, humid)	Y	N	2%
cloud cover (light, medium, heavy)	Y	N	2%

Table 3.1: Selected features (covariates) and related information used in the estimation of batting outcomes.

In Table 3.1, we have added two variables that were not included in the Cricket Australia database which we believe are predictive of the batting outcome. The first is the *resources remaining* variable (Duckworth and Lewis 1998) adapted for T20. The resources remaining at the time that a ball is bowled provides an indication of how aggressive a batsman may bat. To also account for batting aggression, we include the *deficit* variable which is the number of runs by which the batting team is trailing in the second innings (i.e. deficit = target score less current batting score). In the first innings, we define *deficit* as the average runs scored in the first innings of a T20 match less the current score. In men's T20 cricket, the average first innings score is 160 runs whereas in women's T20 cricket, the average first innings score is 131 runs.

In Table 3.1, we observe some redundancies in the variables. For example, the D/L *resources remaining* variable is a function of *wickets lost* and *overs*. In theory, the specification of redundant variables is unnecessary since machine learning techniques are "smart" and are able to detect functional relationships. However, in our experience, over-specification of variables is sometimes useful to assist the performance of algorithms. We also note that the *hit angle* variable has been standardized to account for lefthanded and righthanded batsmen. There were some very minor data management issues such as correcting entries with 10, 11 or 12 wickets lost. These are obvious coding errors where the correct values can be imputed by looking at the data corresponding to adjacent balls.

3.3 Random Forests

Recall that our problem involves the estimation of the batting outcome probabilities $p_i(1), \ldots,$ $p_i(6)$ in (3.1). These probabilities are estimated in a supervised learning context the batting outcomes and the features (Table 3.1) are known for each ball in our massive dataset.

A first thought involving the prediction of categorical outcomes may be the use of "classical" methods such as multinomial logistic regression. However, an important limitation of such methods is the reliance on a linear model structure. In this application, we have many covariates that may interact in non-linear ways where appropriate correlations may be difficult to specify. Moreover, classical parametric models have an underlying stochastic structure which is unknown.

A rationale for machine learning methods in prediction is that complex phenomenon are often difficult to model explicitly. Here, we have a categorical response variable y with 8 categories, and a moderate-dimensional explanatory vector $x = (x_1, x_2, \ldots, x_k)$, with $k = 25$. The reduction of potential covariates in the complete dataset to $k = 25$ variables was carried out in Section 3.2 and based on subject domain knowledge of cricket. We have little apriori knowledge about the relationship between y and x. For example, the relationship may only involve a subset of the variables x, the components of x may be correlated, and most importantly, the relationship $y \approx f(x)$ involves an unknown and possibly complex function f. In addition, the stochastic aspect of the relationship is typically unknown and big data sets may introduce computational challenges. Miraculously, machine algorithms provide black box predictions based on the features of interest.

For this application, we use random forests as the chosen machine learning algorithm. Random forests (Genuer and Poggi 2020) are particularly easy to implement using the *randomForest* package (Liaw and Wiener 2002) in the R programming language (R Core Team 2022). The basic idea is that a random forest is a collection of many decision trees where prediction results are aggregrated over trees. The use of multiple trees improves prediction and makes inference less reliant on a single tree. The splits in the trees accommodate non-linear relationships and terminal nodes provide the estimated probabilities $p_i(1), \ldots, p_i(6)$.

In choosing the tuning parameters, we have a preference for simpler models (i.e. smaller trees). For example, if a more expansive tree has similar prediction accuracy to a modest tree, we choose the modest tree. To assess accuracy (Section 3.4), the data were randomly divided where 20% of the observations were used for training and the remaining 80% of the observations were used for validation and prediction. The 20/80 ratio is low compared to many applications. However, we want a large validation (prediction) set so that there are enough balls to reliably estimate xER for many of the bowlers.

With 20% of the data restricted to training, this still provides a large enough dataset to obtain a good model. When modifying the training set from 20% to 50% of the observations, we found little change in predictions. In the training component, 10-fold cross-validation was utilized, and this allowed us to set tuning parameters. For example, the number of variables randomly selected at each split was set at mtry = 10 to maximize accuracy. We specified 500 trees in the random forest which is the default value. We also used default values for tree depth and the maximum number of nodes. For missing values in our dataset (of which there are few - see Table 3.1), we chose the argument *na.roughfix* which involves a simple imputation scheme.

3.4 Model Validation

This section investigates the quality of estimation in various ways.

Their are four analyses that are of interest. First, we have the T20 data divided into the men's game and the women's game. And then, within each of these two formats, we consider an analysis A based on both bowling and batting features (see Table 3.1). And then we consider an analysis B based only on bowling features (see Table 3.1). The appeal of Analysis A is that it includes the variables *hit angle* and *hit length*. One might presume that these are very predictive features, which provide more accurate probabilities $p_i(1), \ldots, p_i(6)$, and hence, better estimates of xER. The appeal of analysis B is that it removes the quality of the opposition from the analysis. For example, suppose you have a bowler who competes primarily against inferior opponents. Then this bowler's observed economy rate would be lower than if the bowler competed against more challenging competition. But this would not be a problem for xER (under Analysis B) since only the characteristics of the bowled ball and the state of the match are considered; the quality of the opposition is eliminated from the evaluation of xER.

3.4.1 Confusion Matrices

Once a model is trained (fitted), we consider each ball from the validation set. The features of the ball are fed to the model and the batting probabilities are estimated. The batting outcome with the maximum probability is considered the predicted outcome.

For each ball in the validation set, we compare the actual batting outcome with the predicted outcome, and summarize the results in a confusion matrix. The entry (i, j) in the confusion matrix records the number of times an actual batting outcome j was predicted as outcome i.

Tables 3.2, 3.3, 3.4 and 3.5 provide the confusion matrices for Analysis A (Men), Analysis B (Men), Analysis A (Women) and Analysis B (Women), respectively. The first thing that can be calculated from the confusion matrices is that the overall percentage rates of accuracy for the four analyses are 76%, 45%, 78%, and 47%, respectively. Therefore, as anticipated, the batting features *angle* and *hit_length* greatly assist in the accuracy of the predictions. In Analysis B, one prediction that is particularly poor concerns wickets. Although the wicket calculation does not directly appear in the xER formula (3.1), its underestimation impacts the other batting outcomes. For example, in Table 3.3, only 124 wickets were predicted in the roughly 5000 cases where wickets actually occurred. The prediction of wickets is much improved in Table 3.2 (Analysis A). The same comment also applies to the women's game. In all analyses, we observe that 3's and 5's are almost never predicted. In fact, this is sensible, as they almost always result from some sort of fielding error.

Prediction	Actual Outcome							
	Wicket	0	1	2	3	4	5	6
Wicket	1232	123	44	23	0	6	0	0
0	1377	27361	5483	83	4	4	11	0
1	2368	3873	31693	7182	410	305	4	12
2	146	3	543	1042	265	102	0	3
3	1	0	1	8	10	0	0	0
4	44	0	112	195	100	9923	3	96
5	0	0	0	0	0	0	0	0
6	35	1	66	79	15	44	1	4015

Table 3.2: Confusion matrix corresponding to Analysis A (bowling and batting features) for men based on a validation set of 98,450 observations.

Prediction	Actual Outcome							
	Wicket	0	1	2	3	4	5	6
Wicket	124	100	183	64	2	34	0	31
0	1697	16007	9594	2079	344	4174	10	885
1	3313	14820	27714	6294	429	5693	9	3056
2	31	51	121	65	4	38	0	29
3	0	0	1	0	0	1	0	0
4	30	375	309	100	25	432	0	104
5	0	0	0	1	0	0	0	0
6	8	8	20	9	0	12	0	21

Table 3.3: Confusion matrix corresponding to Analysis B (bowling features only) for men based on a validation set of 98,450 observations.

One of the important observations distinguishing Tables 3.2 and 3.3 (men) from Tables 3.4 and 3.5 (women) is the rate at which 6's occur and are predicted. In the men's game, 6's occur $4123/98450 \rightarrow 4.2\%$ of the time whereas in the women's game, 6's occur $1654/133515 \rightarrow 1.2\%$ of the time. This is a reminder that there are significant differences between men's and women's T20 cricket, and that they should be studied separately.

Prediction	Actual Outcome							
	Wicket	0	1	2	3	4	5	6
Wicket	1610	170	32	8	0	0	0	0
0	2683	46023	9924	162	11	4	11	1
1	2267	4687	38943	7879	412	4	7	7
2	143	7	915	1772	324	34	0	2
3	0	0	6	24	26	0	0	0
4	5	0	33	28	10	13564	1	70
5	0	0	0	0	0	0	0	0
6	9	1	40	32	9	41	0	1574

Table 3.4: Confusion matrix corresponding to Analysis A (bowling and batting features) for women based on a validation set of 133,515 observations.

Prediction	Actual Outcome							
	Wicket	0	1	2	3	4	5	6
Wicket	34	49	61	4	1	10	1	34
0	2765	31533	18581	3576	373	6598	11	2765
1	3897	19141	31017	6255	414	6865	7	3897
2	14	44	71	19	3	26	0	14
3	0	2	1	1	0	1	0	0
4	7	118	158	47	1	144	1	7
5	0	0	0	0	0	0	0	0
6	0	1	4	3	0	3	0	0

Table 3.5: Confusion matrix corresponding to Analysis B (bowling features only) for women based on a validation set of 133,515 observations.

We note that the xER statistic does not distinguish between wickets and 0s; each contributes no runs to the xER statistic (3.1). Therefore, it may be interesting to reproduce the confusion matrix given in Table 3.3 by combining wickets and 0s. This new table is provided in Table 3.6. When comparing Table 3.3 and Table 3.6, we observe that the predictions are generally less accurate when wickets and 0s are combined.

Prediction	Actual Outcome						
	W & 0	1	2	3	4	5	6
W & 0	20695	13474	2909	414	4849	11	1341
1	15199	23855	5469	366	4947	8	2573
2	121	182	88	3	73	0	56
3	1	4	0	0	2	0	0
4	522	395	129	20	490	0	132
5	0	0	0	0	0	0	0
6	26	32	17	1	23	0	24

Table 3.6: Confusion matrix corresponding to Analysis B (bowling features only) for men based on a validation set of 98,450 observations. Here, the wickets and 0 categories have been combined.

3.4.2 Features of Importance

One of the informative outputs from the *randomForest* package (Liaw and Wiener 2002) are importance plots of the model features. Features are listed from top to bottom according to their impact on prediction.

We provide importance plots for the men's game (Figure 3.1) and for the women's game (Figure 3.2). Importance plots are provided for Analysis A (bowling and batting features) and Analysis B (bowling features only). As expected, we observe in the leftmost plots that the variables *hit length* and *hit angle* are the most influential as they describe the characteristics of the batted ball. For example, balls that are hit far are most likely to result in runs scored.

In both Analysis A and Analysis B, we observe that the landing location of the bowling delivery (*pitchx* and *pitchy*) impact the batting outcome prediction. The same is true for (*batsmanx* and *batsmany*) which describe the landing location of the ball relative to the batsman. For example, if the ball is too close to the batsman, the batsman is unable to apply full torque on the batted ball, and is less likely to generate 4's and 6's. Our introduced variables *deficit* and *resources remaining* which together describe the urgency and aggressiveness of batting are also variables which help the prediction of batting outcomes.

Comparing Figure 3.1 and Figure 3.2, we do not observe many meaningful differences between the men's game and the women's game. It appears that the features that are influential in predicting batting outcomes are similar.

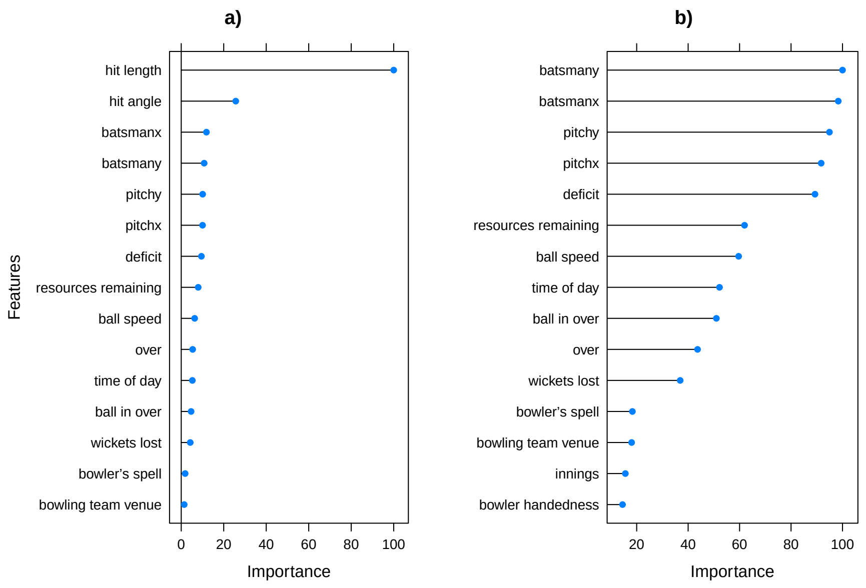


Figure 3.1: Importance plots of the features for the men's game corresponding to Analysis A and Analysis B.

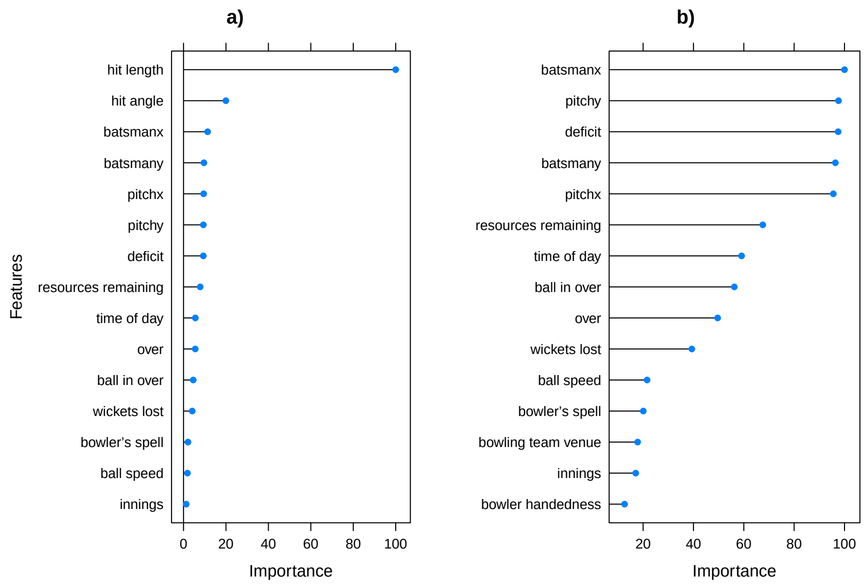


Figure 3.2: Importance plots of the features for the women's game corresponding to Analysis A and Analysis B.

3.4.3 Brier Scores

As mentioned previously, the utility of xER is only as good as the reliability of the estimates of the probabilities p_i in (3.1). To assess the estimates, we calculate Brier scores (Brier 1950) based on three forecasts.

The first forecast is naive and is not expected to be accurate. However, it does provide a sense of the magnitude of differences with respect to Brier scores. We refer to the first forecast as Uniform Discrete where the associated probabilities are given in Table 3.7. Here, we set all of the six non-neglible probabilites in (3.1) equal to 1/6. We emphasize that the probabilities are the same for every ball that is bowled.

The second forecast which we refer to as T20 Proportions is based on the observed proportions of T20 batting outcomes from a much larger dataset. It includes more than 500,000 balls from international men's T20 matches from 2015-2020. Again, we assign zero probability to the rare batting events corresponding to 3 and 5 runs. The probabilities are given in Table 3.7. Since these probabilities are associated with the men's game, our Brier score analysis will only consider batting predictions for men.

Forecast	p(wicket)	$p(0)$	$p(1)$	$p(2)$	$p(3)$	$p(4)$	$p(5)$	$p(6)$
Uniform Discrete	0.166	0.166	0.166	0.166	0.000	0.166	0.000	0.166
T20 Proportions	0.056	0.305	0.404	0.071	0.000	0.112	0.000	0.048

Table 3.7: Probability estimates associated with two competing forecasts.

Now, for every ball $i = 1, \ldots, N$ that is bowled in the dataset, we define $o_i(j) = 1$ if event j occurred and $o_i(j) = 0$ if event j did not occur where the event j takes on the values $w, 0, \ldots, 6$. With event probabilites $p_i(j)$ corresponding to a particular forecasting method, the Brier score is then given by

$$B = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=w,0,\ldots,6} (p_i(j) - o_i(j))^2 \ . \tag{3.2}$$

For the men's game, we calculate Brier scores using (3.2) for the Uniform Discrete forecast, the T20 Proportions forecast and the random forests forecast (Analysis A), We obtain Brier scores of 0.84, 0.73 and 0.33, respectively. The results suggest that the random forests algorithm (Analysis A) predicts the batting outcome much better than the other two methods that do not consider the circumstances of the match and the characteristics of the ball that was bowled. In turn, and most importantly, this suggests that xER informs us about what might reasonably have happened in matches involving economy rate had the batting outcomes proceeded as expected.

3.5 Results - xER in T20 Cricket

For a given bowler, we are interested in the comparison of xER with their career economy rate. Career economy rates in T20 were obtained from the stats.espncricinfo.com website.

For this analysis, we are only interested in bowlers who have bowled sufficiently in our validation (prediction) dataset. We therefore restricted our attention to bowlers who have bowled at least 500 balls in the validation set. From these bowlers (57 men and 89 women), we randomly selected 10 men and 10 women and calculated their corresponding xER. The calculation was based on the estimates $p_i(1), \ldots, p_i(6)$ in (3.1) for each ball i that was bowled in the validation set. The xER statistic in (3.1) was calculated two ways; using Analysis A which relied on bowling and batting features, and Analysis B which only used bowling features. The results are presented in Table 3.8 (men) and Table 3.9 (women).

Our initial observation is that the expected economy rates xER are in line with the career economy rates. Sometimes the xER statistic is smaller (the bowler has been slightly unlucky in actual matches) and sometimes the xER statistic is larger (the bowler has been slightly lucky in actual matches). We have argued up to this point that Analysis A is most likely better than Analysis B since Analysis A has more accurate predictions. Perhaps the most interesting bowler according to Analysis A is Renee Chappel from Table 3.9. Her xER (Analysis A) is a full 2.81 runs lower than her career economy rate. This is a large difference which suggests that she is a better bowler than her record indicates. She is an experienced bowler, 38 years of age, having made her international debut in 2013. In 2016-17, Chappel received the Karen Read Medal as the best player in WACA Female A Grade and T20 competitions. It appears that her excellence as highlighted by the xER statistic has been appreciated.

We also observe that the men bowlers tend to have higher economy rates on average than the women bowlers. This could be related to the observation from Section 3.4.1 where we observed that the men score 6's more frequently than the women.

Bowler	Country	Balls Bowled in Validation Set	Career Economy Rate	xER (Analysis A)	xER (Analysis B)
Steketee, Mark	Australia	595	8.90	8.37	7.53
Lyon, Nathan Michael	Australia	620	7.21	7.28	7.74
Abbott, Sean	Australia	1185	8.54	8.54	7.82
Hauritz, Nathan	Australia	533	7.56	7.75	7.79
Archer, Jofra	England	512	7.65	7.43	7.70
Maxwell, Glenn	Australia	802	7.71	7.68	7.36
Ahmed, Fawad	Australia	933	6.96	6.88	8.13
Zampa, Adam	Australia	1306	7.29	6.76	7.94
McKay, Clinton	Australia	950	8.07	8.41	7.46
Boyce, Cameron	Australia	1109	7.65	7.72	7.96

Table 3.8: The expected economy rate statistic xER for 10 randomly selected men bowlers.

Bowler	Country	Balls Bowled in Validation Set	Career Economy Rate	xER (Analysis A)	xER (Analysis B)
Pike, Kirsten	Australia	708	7.34	5.93	5.67
Hepburn, Brooke	Australia	1454	6.97	6.70	6.15
King, Emma	Australia	1379	6.41	6.06	6.30
Birkett, Haidee	Australia	553	6.75	6.67	6.28
Kearney, Emma	Australia	657	6.22	6.33	5.73
Chappell, Renee	Australia	634	8.28	5.47	6.45
Elwiss, Georgia	England	604	5.92	6.90	6.55
Biss, Emma	Australia	521	5.22	6.01	6.08
Elliott, Sarah	Australia	921	6.20	5.72	6.24
Coyte, Sarah	Australia	2521	6.10	6.25	6.12

Table 3.9: The expected economy rate statistic xER for 10 randomly selected women bowlers.

3.6 Discussion

This chapter introduces expected economy rate xER to the sport of cricket. The idea borrows on the expected goals concept which has become especially popular in soccer. As xER attempts to reduce the luck element from bowling, xER may be a diagnostic that informs us of the true quality of a bowler, perhaps a more trustworthy statistic than the actual economy rate. This may be particularly valuable in the context of unproven bowlers who have not established a clear reputation. For example, xER could alert team selectors to promising bowlers whose results have not yet matched their quality.

This chapter is intended to be a proof of concept of a potentially valuable statistic. Naturally, the statistic could improve with better data. For example, the positioning of fielders surely has an impact on batting outcomes. We suggest that the availability of such data is not far down the road as player tracking data becomes more widely available across major sports.

Of course, different models and prediction schemes could also be investigated. This is a possible avenue for future research. Another point of reference for future research in cricket analytics is a call for more work on the women's game. This research has pointed out that there are significant differences between the men's and women's games.

Chapter 4

A Causal Investigation of Pace of Play in Soccer

4.1 Introduction

In team sports, playing style is a much discussed topic and an important component of success. For example, in soccer, we hear about the gegenpress (Tweedale 2022), total football (McLellan 2010) and parking the bus (Guan, Cao and Swartz 2022). However, playing style is notoriously difficult to quantify in soccer. It is difficult to quantify since playing style is a team concept, which relies on the actions of multiple players whose movements are fluid in both time and space.

However, the landscape for studying playing style has changed in recent years with the advent of player tracking data. With player tracking data, the location coordinates for every player on the field are recorded frequently (e.g. 10 times per second in soccer). With such detailed data, the opportunity to explore novel questions in sport has never been greater. The massive datasets associated with player tracking also introduce data management issues and the need to develop modern data science methods beyond traditional statistical analyses. Gudmundsson and Horton (2017) provide a review of spatio-temporal analyses that have been used in invasion sports where player tracking data are available.

This chapter is concerned with "pace of play" in soccer, a relatively underexplored topic. In some sports, pace is readily defined. For example, in basketball, team pace may be defined as the average number of possessions per game. In the NBA, this is a well-studied statistic which is available from various websites including https://www.nba.com/stats/teams/advanced/

In American football, although there is a clear notion of pace of play, there is no commonly reported statistic that directly measures pace. In the National Football League (NFL), the number of plays per game is available for each team from standard box scores. Although this statistic is related to pace, it is obvious that poor offensive teams who rarely make first downs have fewer plays per game. Therefore, in football, the average number of plays per game for a team is confounded with offensive strength, and consequently, the number of plays is not a pure measure of pace. Pace in football can be increased for a team by using a "hurry-up offense" which affords more plays in a given period of time provided that the team continues to make first downs. Furthermore, teams that frequently pass the ball (as opposed to run the ball) typically use up less of the clock and have more plays from scrimmage.

In ice hockey, the definition of pace is even less clear. See, for example, Silva, Davis and Swartz (2018) where various definitions of pace are considered. Yu et al. (2018) revisit the hockey problem and suggest an alternative definition of pace.

The sport of soccer shares some of the same challenges as hockey with respect to the definition of pace. For example, how is possession determined? How do successful passes contribute to pace and should pace calculations involving a pass be counted differently than when dribbling? Shen, Santo and Akande (2022) builds on the aforementioned hockey papers and uses event data to investigate pace in soccer.

This chapter differs from Shen, Santo and Akande (2022) in a number of key directions. First, this chapter uses tracking data rather than event data to study pace. Second, alternative definitions of pace are provided. In particular, we define *attacking pace* which is related to "direct play", a much discussed tactic in soccer. Third, we provide various sporting implications associated with pace. Finally, our primary goal addresses the key question of whether playing with pace is strategicly sound. Many soccer experts believe that moving the ball quickly is advantageous. When you move the ball quickly, the logic is that it affords the defensive team less time to transition to solid defensive formations. However, to our knowledge, this basic tenet of soccer has never been tested. Is it better to play with pace? We address this question by using methods of causal inference (Pearl 2009). Obviously, decisions that are made on the field are often instantaneous. Therefore, it is impossible to use traditional randomized trials to determine the cause-and-effect relationship between playing with pace and success. With match data, we have "studies" as opposed to "experiments". Fortunately, the methods of causal inference allow us to address causality in studies provided that confounding variables can be identified and measured. With tracking data and our subject knowledge of soccer, confounding variables are accessible.

Related to our investigation of pace, they have been many investigations of determinants of success in soccer. A sample of recent papers include Lepschy, Wäsche and Woll (2021), Merlin et al. (2020), and in the women's game, de Jong et al. (2020).

In soccer, the most investigated aspect of playing style concerns formations. For example, the book "Inverting the Pyramid" (Wilson 2013) considers the history of soccer tactics throughout the world with an emphasis on positional play and player roles. It is also now common during television broadcasts to provide graphical statistics that depict the average location of each player during a match. Such information is useful in determining match strategy as it can point out features such as gaps in player alignment. There have also been many technical papers written on player formation. For example, Shaw and Glickman (2019) use tracking data and clustering methods to determine a team's offensive and defensive formations. This is useful as the fluidity of the sport and changing tactics sometimes makes it difficult to distinguish between formations (e.g. 4-4-2 versus 3-5-2). Goes et al. (2021) also identify formations using tracking data and relate attacking success to formations.

The association between style and results in soccer has been well investigated. For example, in their Table 1, Kempe et al. (2014) list various ball possession and passing metrics which have been explored in the literature. Kempe et al. (2014) also propose aggregate metrics and relate these to success. However, a distinguishing feature of our work is that we consider a causal approach rather than one of association. This is made possible by the availability of player tracking data.

In Section 4.2, we introduce and motivate two definitions of pace. We contrast these definitions with alternative definitions that have been presented in the literature. In Section 4.3, we describe the player tracking dataset and discuss the challenges involved in pace calculations. One of the challenges is the determination of possession. In Section 4.4, we provide exploratory data analyses which provides various sporting insights on pace. The sporting insights are highlighted with the letters A-E. This section is also useful in identifying confounding variables that are related to pace. In Section 4.5, we present a causal analysis concerning the benefit of playing with pace. This involves the fitting of a MANOVA model which is the foundation for the determination of propensity scores and matching. The main result of this section is that playing with pace is a beneficial team strategy in soccer in terms of generating more shots. We conclude with a short discussion in Section 4.6.

4.2 Definitions of Pace in Soccer

Dan Blank's paperback on soccer (Blank 2002) provides 54 chapters on different tactics and advice on playing the game well. The first chapter which is titled the "Holy Grail" provides an inspiration for our investigation. In this chapter, Blank claims that playing fast is better than playing slow. In other words, Blank argues that teams should play with pace. Although the heuristic may be appealing, it does not seem that the belief has ever been corroborated against data. If the belief is true, then a measurable and sensible definition of pace may lead to important soccer insights.

First, we review some of the previous definitions of pace. In the original investigation of pace in hockey, Silva and Swartz (2018) were limited to the analysis of event data. With event data, a finite number of event types are recorded along with a timestamp. A shortcoming of the analysis is that the skating paths between events (which are relevant to pace) are unknown. Consequently, Silva and Swartz (2018) only measured horizontal distances (i.e. down the length of the rink) during which possession was maintained. Furthermore, Silva and Swartz (2018) only evaluated pace for a game and did not differentiate pace of play between the two teams. Yu et al. (2019) used more extensive event data with events recorded approximately every second on average. With this data, they were able to define pace in various directions and considered zonal, league, team-level and player-level analyses. The pace metric defined by Yu et al. (2019) appears to be an average of velocities over event intervals and therefore differs conceptually from the Silva and Swartz (2018) definition which is based on total distance travelled. In soccer, Shen, Santo and Akande (2022) also used velocity as a pace measurement but restricted analyses to sequences where possession is retained over three or more events.

A commonality amongst all of the above pace analyses is that they were based on event data. With event data, distance calculations between events assume that the ball/puck travels in a straight line. Shen, Santo and Akande (2022) described the assumption as a major limitation. In this chapter, the more detailed tracking data allows us to consider the actual paths where the ball travelled.

We begin with an analogy related to our definition of pace. We suggest that a painter is painting quickly (i.e. with pace) if they are able to apply a lot of paint on a canvas in a short period of time. In soccer, we view the brush strokes as the paths where players carry the ball and the paths where a ball is successfully passed. If a team is able to move the ball quickly, then they are playing with pace. The concept of possession is important; if a team is simply punting the ball downfield, in our view, they are not playing with pace. To operationalize these ideas, we consider the non-contiguous time intervals $(t_1, t_2), \ldots, (t_n, t_{n+1})$ in a match where a team has possession. During the possession interval i, the team moves distance d_i, $i = 1, \ldots, n$. Then, following the painting analogy, we refer to the team's *general pace* in the match as

$$GP = \frac{\sum_{i=1}^{n} d_i}{\sum_{i=1}^{n}(t_{i+1} - t_i)} \ . \tag{4.1}$$

Similarly, there is a corresponding pace formula (4.1) for the opponent. Note that the two teams will differ in the amount of time possession during the match. Therefore, the pace measure (4.1) is reflective of their style of play while in possession, and is insensitive to their total time of possession. Although the general pace metric (4.1) is defined in terms of a match, it can also be calculated for shorter periods of time (e.g. a half) or even for a single possession.

We contrast the general pace metric (4.1) with the quantity

$$P_{\text{SSA}} = \frac{1}{n} \sum_{i=1}^{n} \frac{d_i}{t_{i+1} - t_i} \tag{4.2}$$

which is related to the velocity concept of pace utilized by Shen, Santo and Akande (2022); note, however that Shen, Santo and Akande (2022) used medians rather than means. When comparing (4.1) with (4.2), we observe that (4.2) is sensitive to and is inflated by very fast passes (i.e. typically large d_i that occur over moderate time intervals $t_{i+1} - t_i$). We believe that (4.1) better reflects pace as the totality of distance covered with respect to the cumulative time of possession.

We now introduce a variation of the general pace metric GP defined in (4.1). We note that there are differences in scoring intent based on the type of passes and dribbling. For example, the "tiki-taka" approach adopted by the Spanish National team in 2006 relied on many consecutive short passes that emphasized possession. Based on the metric GP, the tiki-taka approach would be characterized as a pacey style since passes typically have larger pace contributions than dribbling. But is tiki-taka pacey?

The aforementioned tiki-taka style allows one to reflect on stylistic differences between hockey and soccer. In hockey, the playing surface is smaller and players skate at great speeds. Therefore, it is more difficult to retain possession in hockey. Consequently, possession sequences tend to be of shorter duration than in soccer. To investigate pace in soccer with an emphasis on direct play, we modify GP and introduce *attacking pace* AP where the distances d_i now correspond to displacements down the field in the direction of the opposing goal. For example, passes back to the keeper (which have no attacking intent) do not positively contribute to attacking pace AP. Large positive contributions to the statistic AP will involve transitions such as the counter-attack.

To define AP, we refer to Figure 4.1 where an AP contribution is illustrated. In the plot, the "most attacking" pass that could possibly be made from point A would be to the middle of the opponent's goal line C.

This potential pass has associated distance d_{AC}. Instead, the pass was made from A to B, and the attacking distance from this new point B to C is denoted d_{BC}. Therefore, the contribution (in terms of attacking) from A to B is given by the residual distance

$$d = \begin{cases} d_{AC} - d_{BC} & d_{AC} \geq d_{BC} \\ 0 & d_{AC} < d_{BC} \end{cases} . \tag{4.3}$$

In (4.3), $d_{AC} - d_{BC}$ represents the reduction of the greatest attacking distance that was made due to the pass from A to B. Therefore, the new statistic AP has the same form as GP in equation (4.1) where the d in equation (4.3) assumes a subscript i corresponding to the ith possession. We also note that the same type of calculation is carried out whether a possession involves passes or dribbles. It is important to note that the tracking data allows us to deal with path curvature when dribbling by breaking up dribbling sequences into small time intervals. If event data had been used (in contrast to tracking data), only the starting point and ending point of a dribbling sequence would be known.

A feature of the construction of the metric AP is illustrated through a possession sequence where the ball travels in a forward direction from A to B to C and where we denote the center of the goal by G. Using obvious notation for distances, the total attacking dis-

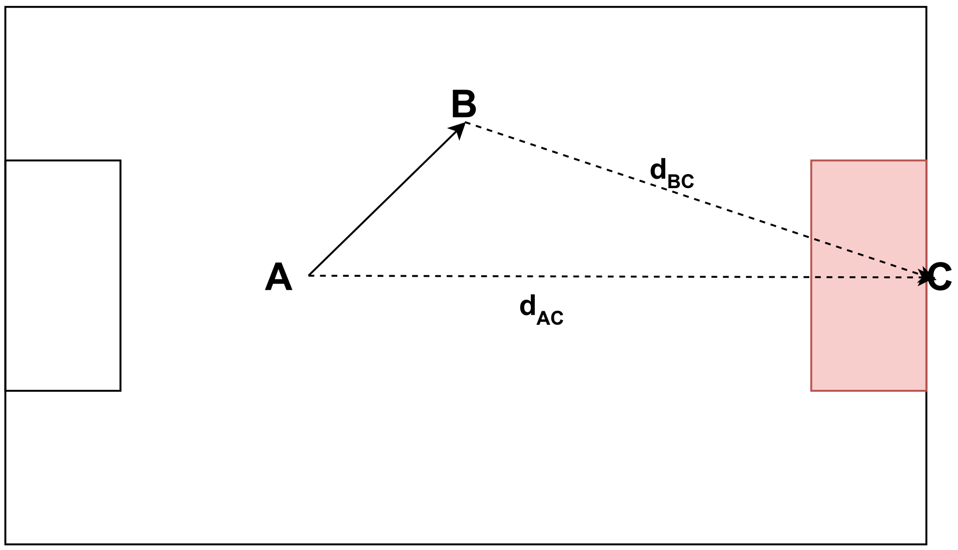


Figure 4.1: The plot illustrates a pass with attacking intent. A is the starting point of the pass, B is the end point of the pass and C denotes the middle of the goal line of the opponent. The values d_{BC} and d_{AC} represent the distances from B to C, and A to C, respectively. Attacking pace AP for this component of play is obtained using the distance $d_{AC} - d_{BC}$.

tance (4.3) is given by $d = (d_{AG} - d_{BG}) + (d_{BG} - d_{CG}) = d_{AG} - d_{CG}$ which demonstrates that the metric is additive over the possession path.

Whereas the metrics GP and AP describe style of play while a team is in possession, insights may also be provided by considering the extent to which teams play a given style. For example, if a team is rarely in possession, then they are rarely executing their style. Therefore, we could also introduce the metric GP^* which is similar to GP except that we omit the denominator in (4.1). Therefore, GP^* may be thought of as total distance travelled by the team. Therefore, while GP is a statistic that describes pace during possession, GP^* takes possession into account such that teams with little possession are not playing with pace. Similarly, we could introduce the metric AP^* which is the total attacking distance during the match. However, for the remainder of our investigation, we only focus on the general pace statistic GP and the attacking pace statistic AP. Note that all of the proposed definitions of pace are properties of the possessing team.

4.3 Data

For this investigation, we have a big data problem where both event data and player tracking data are available for 237 regular season matches (three matches missing) from the 2019 season of the Chinese Super League (CSL). The schedule is balanced where each of the 16 teams plays every opponent twice, once at home and once on the road.

Event data and tracking data were collected independently where event data consists of occurrences such as tackles and passes, and these are recorded along with auxiliary information whenever an "event" takes place. The events are manually recorded by technicians who view film. Both event data and tracking data have timestamps so that the two files can be compared for internal consistency. There are various ways in which tracking data are collected. One approach involves the use of RFID technology where each player and the ball have tags that allow for the accurate tracking of objects. In the CSL dataset, tracking data are obtained from video and the use of optical recognition software. The tracking data consists of roughly one million rows per match measured on 7 variables where the data are recorded every 1/10th of a second. Each row corresponds to a particular player at a given instant in time. Although the inferences gained via our analyses are specific to the CSL, we suggest that the methods are applicable to any soccer league which collects tracking data.

4.3.1 Possession

A possession is defined as a period where a team has control of the ball. The event data is used to identify the possession sequences of a team. The event data contains all the events that occurred during a match and therefore tells us when possession sequences began and ended. Events where neither team is determined to have possession include injuries, cards, out-of-bounds, preliminary time to the beginning of set pieces (eg corners, throw-ins, free kicks, penalties, etc). Also, when determining possession sequences, we exclude time beyond 90 minutes since different matches have different amounts of added time. Among all the matches, there is an average of 373 possessions per match.

In Figure 4.2, we provide histograms (GP and AP) of the length of the possession sequences in metres. The histogram is right skewed. We observe a mean length of 46.6 (21.0) metres, minimum length 0.1 (0.0) metres, and maximum length 456.8 (82.2) metres corresponding to GP and AP, respectively.

4.3.2 The Pace Datasets

We pre-processed the CSL tracking and event data. Originally, the data were provided in xml files and we extracted the content using the `read_xml` function from the `XML` package using R software. The resulting tracking and event data were written into csv file format.

Ultimately, we constructed a pace dataframe for each match. This is a comprehensive dataset that allows us to investigate various questions of interest. The pace dataset is a

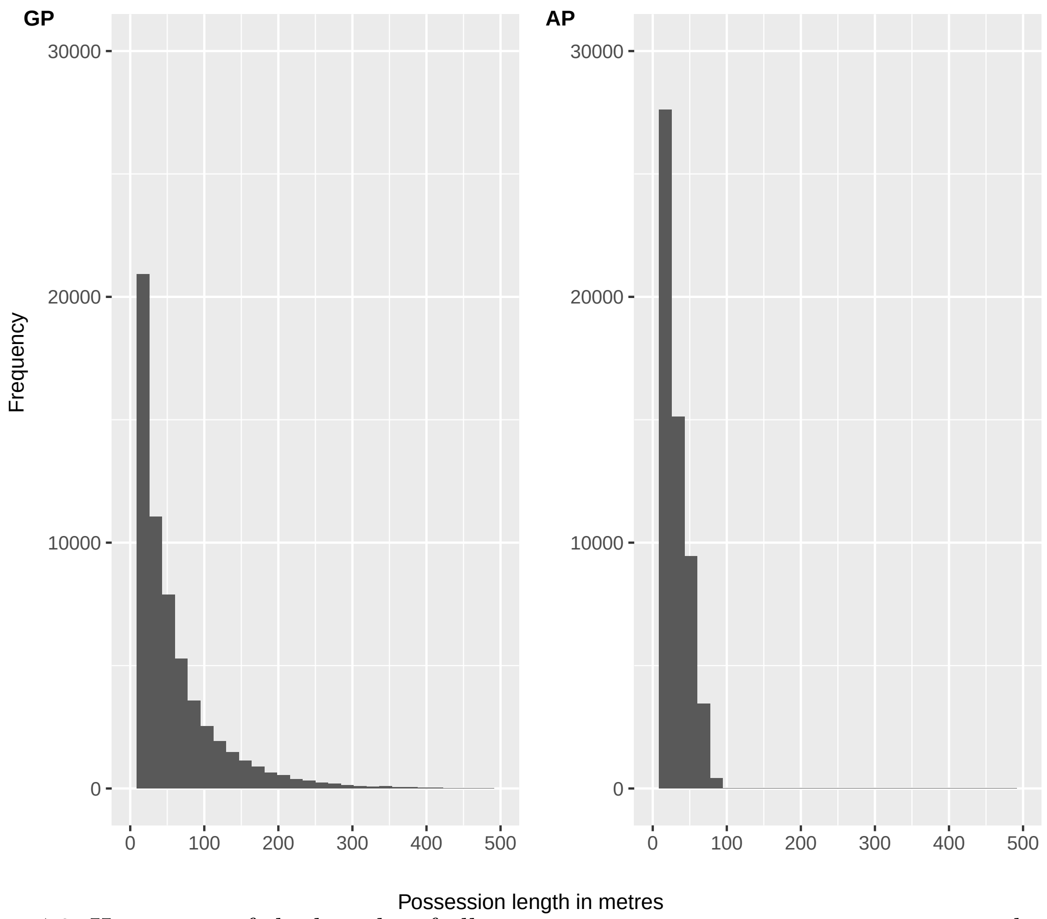


Figure 4.2: Histogram of the lengths of all possession sequences in metres corresponding to GP and AP, respectively.

matrix where the rows correspond to pace contributions made by an individual player during a possession. The columns consist of the following variables: start time of pace contribution, end time of pace contribution, the displacement d_i (both Euclidean distance and attacking distance (4.3)), match score at the beginning of the pace contribution, match score at the end of the pace contribution, the player who contributed to the pace contribution, the team of the player who made the pace contribution, whether the contributing player plays for the home or road team, and the number of playing minutes during the match for the player. We note that Yu et al. (2018) shared the pace contribution equally between the player who made the pass and the player who received the pass. In our construction, we assign credit only to the player who made the pass.

To create the pace dataframe, we looped frame by frame through the tracking data, where we matched events and time using the event data. This permitted the calculation of the relevant distances during each possession. The process required approximately 15 minutes of computation for all 237 matches. Another challenge related to the calculation of

AP involved slight differences in pitch size where the coordinates of point C in Figure 4.1 varied across pitches.

4.4 Exploratory Data Analyses

A main objective of exploratory data analysis (EDA) is to reveal insights that can be more thoroughly investigated via modelling and inferential techniques. In this section, we use EDA to gain insights related to the pace statistics GP and AP together with other variables of interest. Below, EDA reveals five insights, labelled A-E.

In Figure 4.3, we produce scatterplots of GP and AP related to the home and road teams for all of the 237 available matches during the 2019 season of the CSL. We obtain a mean value of 0.66 (0.66) metres/sec, minimum value 0.42 (0.48) metres/sec and maximum value 0.78 (0.82) for the home and road team, respectively, using GP. We obtain a mean value of 0.25 (0.26) metres/sec, minimum value 0.14 (0.15) metres/sec and maximum value 0.38 (0.53) for the home and road team, respectively, using AP. Therefore, we observe that the pace statistics differentiating home and road teams are minor.

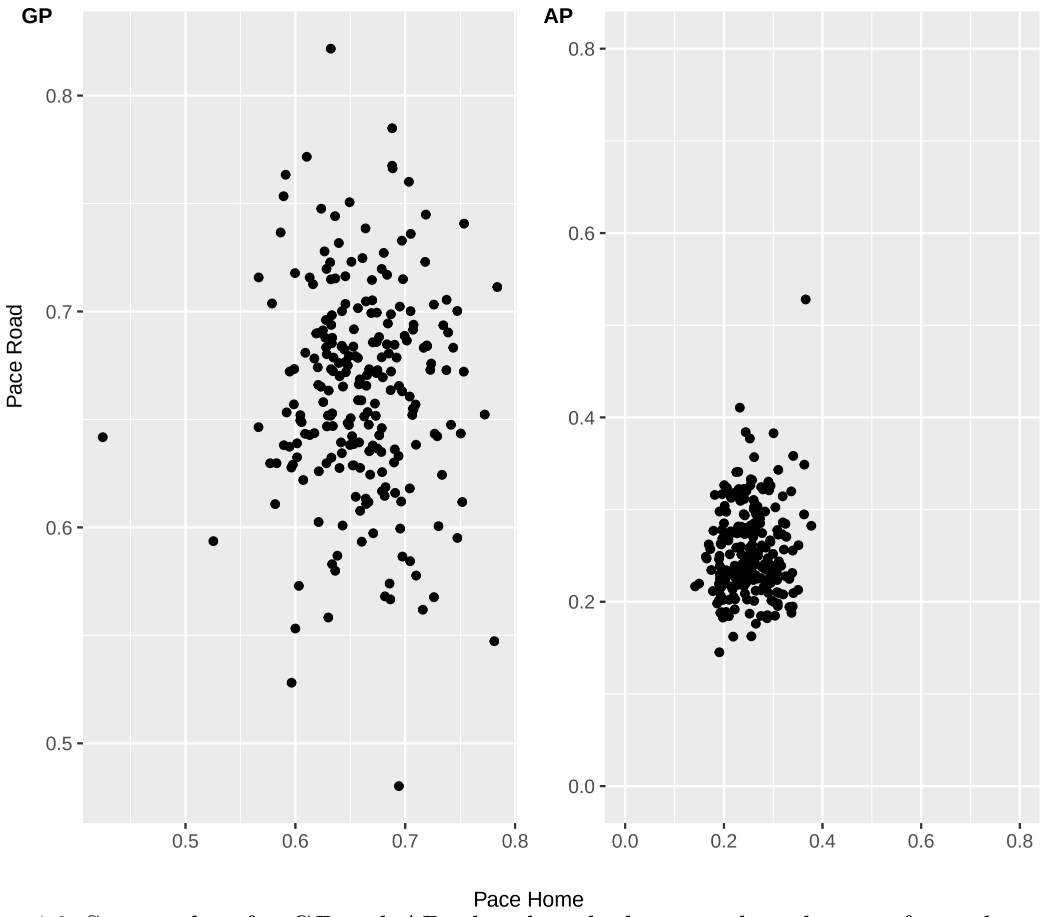


Figure 4.3: Scatterplots for GP and AP related to the home and road teams for each match.

Initially, we were unsure whether pace was a property of the match (e.g. both teams play at high pace due to the particular style of the game) or whether each team has control of their respective pace. We observe that the sample correlation coefficients for GP and AP are 0.16 and 0.06, respectively. It is possible to carry out a test of correlation $H_0 : \rho = 0$ in the two cases. The p-values are given by 0.014 and 0.358, for GP and AP, respectively. Although the first correlation is statistically significant, it is not strong in magnitude. The lack of strong correlations lead to the following insight.

Insight A: In a given match, each team has control of whether they play a pacey style. The pace of one team is not dictated by the pace of its opposition.

Next, we are interested in whether pace is a characteristic that can be attributed to teams. In Figure 4.4, we produce boxplots of the pace (GP and AP) for each of the 16 teams in the CSL where a single datapoint refers to the pace calculation in a match. We observe that there are only minor differences in the pace distributions across teams.

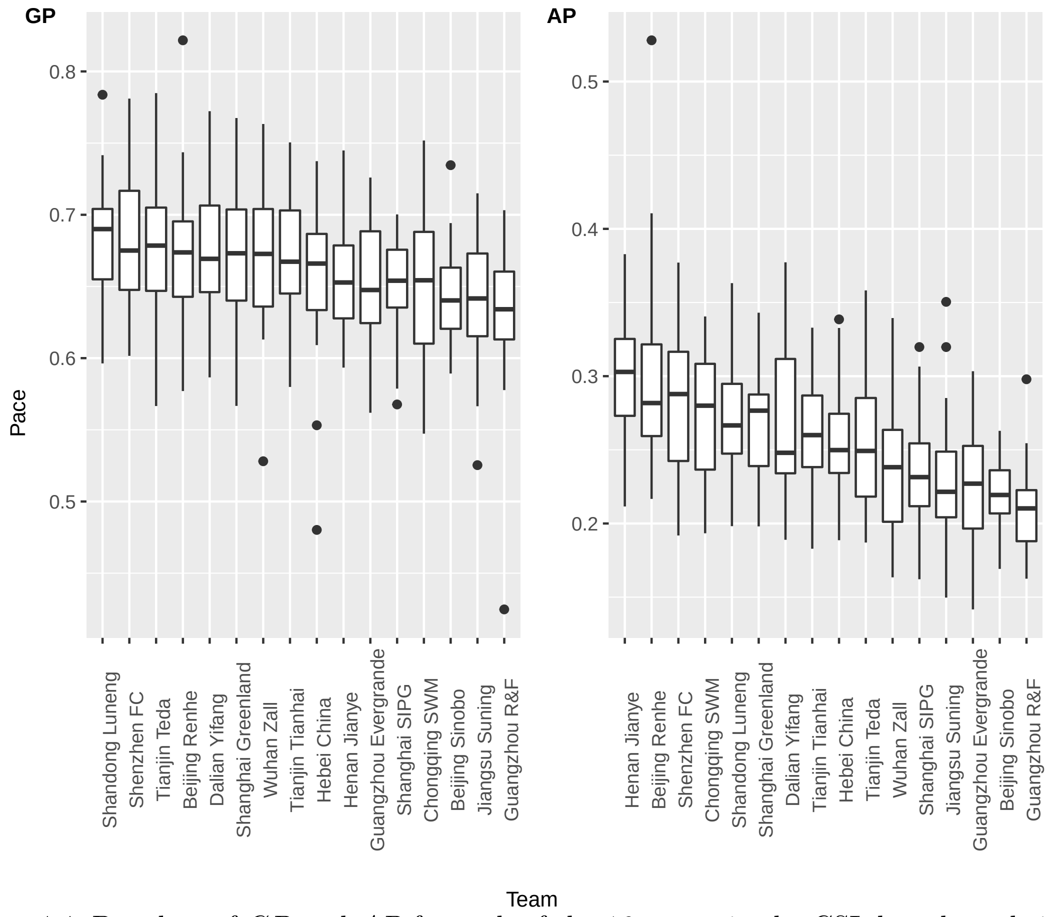


Figure 4.4: Boxplots of GP and AP for each of the 16 teams in the CSL based on their 30 matches.

Using a one-way ANOVA design for testing differences across teams, we obtain p-values of 0.0278 and 0.0106, for GP and AP, respectively. This leads to the following insight.

Insight B: Although some teams may play at slightly different average pace than other teams, such differences are small (particularly with GP. Pace is primarily a property of how a team plays in a particular match rather than a general property of the team.

Next, we are interested in whether pace depends on playing position. In Figure 4.5, we produce boxplots of the pace (GP and AP) for defenders, midfielders and forwards in the CSL where a single datapoint refers to the pace calculation in a match. We observe differences across the three positions. Due to the constraints of the field and positioning, it is logical that defenders have more open space in front of them than midfielders, and that midfielders have more open space in front of them than forwards. Therefore, it coincides with our intuition that pace should decrease according to defenders, midfielders and forwards, respectively.

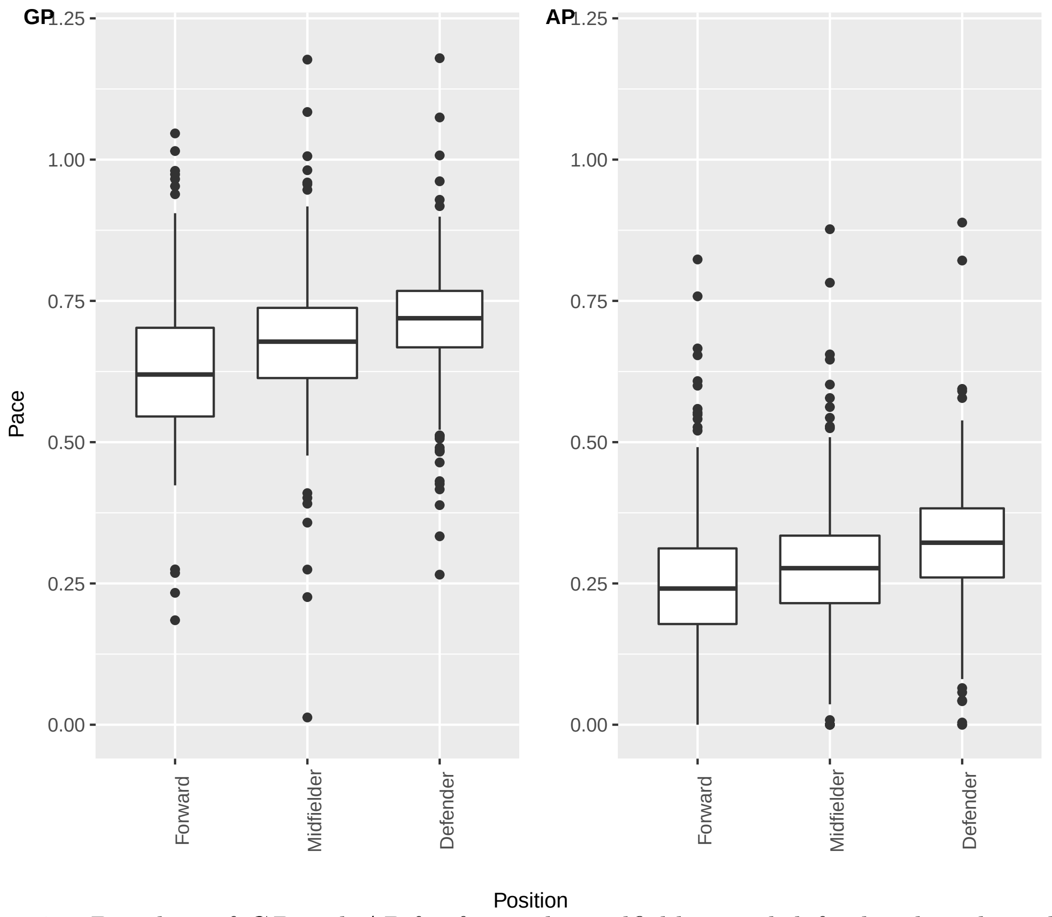


Figure 4.5: Boxplots of GP and AP for forwards, midfielders and defenders based on their individual match statistics.

Using a one-way ANOVA design for testing differences across positions, we obtain highly significant test results with p-values of $4.13e^{-10}$ and $5.18e^{-6}$, for GP and AP, respectively. This leads to the following broad insight.

Insight C: Defenders play at higher pace levels than midfielders who in turn play at higher pace levels than forwards.

Next, we are interested in whether pace is related to the time of the match. In Figure 4.6, we produce boxplots of the total pace by both teams (GP and AP) according to the time of the match broken into 15-minute intervals from 0 to 90 minutes. Although pace changes throughout the match, we observe different patterns according to GP and AP. With general pace GP, when a match begins, we expect that teams are alert and maintain defensive discipline. As the match continues, players tire, and they discontinue running with the same pace as before. At halftime, there is a rest period where teams recover slightly, and then they continue to tire during the second half.

With attacking pace AP, the interplay between exhaustion and defensive discipline is expressed differently. As the match continues, players tire and this allows for more open space and the opportunity to seek gaps downfield. This causes a gradual increase in attacking pace with more pronounced increases in the latter stages.

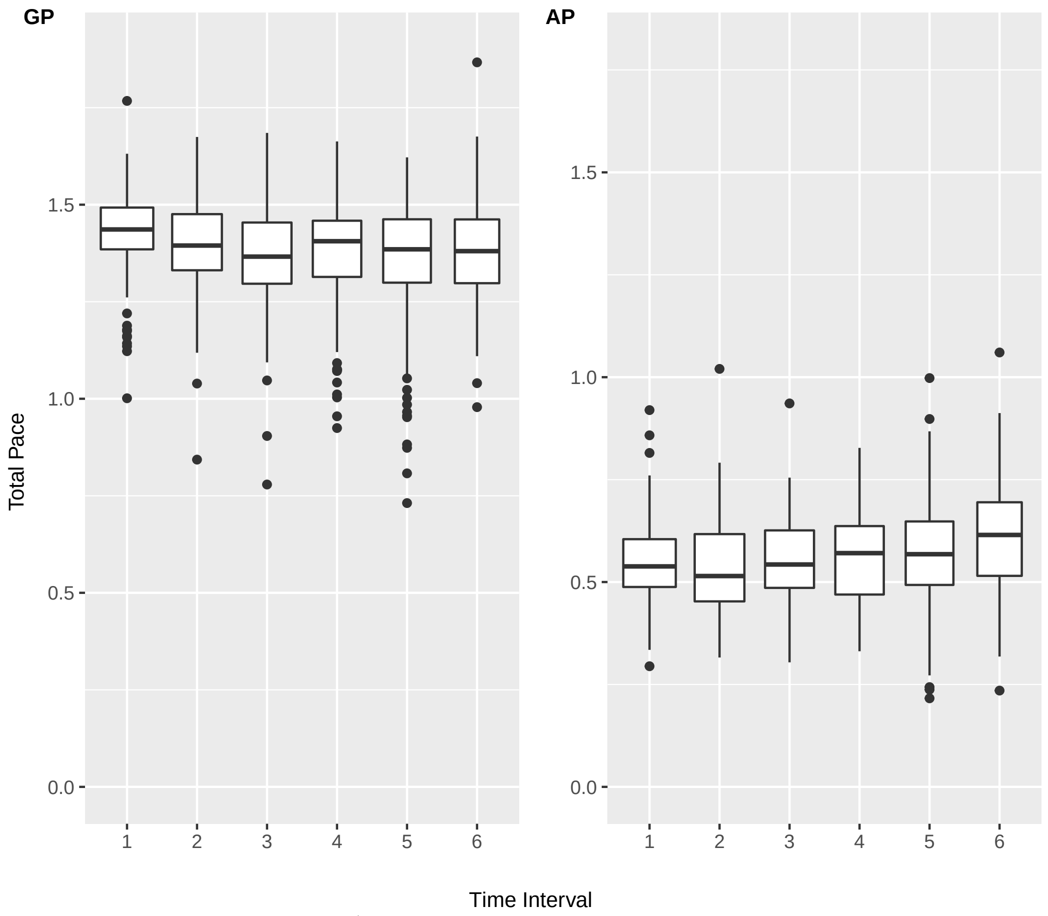


Figure 4.6: Boxplots of GP and AP according to the time of the match where time is divided into six 15-minute intervals from 0 to 90 minutes. The pace calculations are the total pace corresponding to both teams.

Using a one-way ANOVA design for testing differences across time intervals, we obtain highly significant test results with p-values of $3.66e^{-8}$ and $4.56e^{-5}$, for GP and AP, respectively. This leads to the following insight.

Insight D: Teams plays at higher attacking pace as the match progresses.

Next, we are interested in whether pace is related to goal differential. In Figure 4.7, we produce boxplots of GP and AP corresponding to five goal differential categories as explained in the caption. The calculation of pace is taken over five-minute intervals for all teams and matches during the season. When a goal is scored during a five-minute interval, then the pace observation for that interval is excluded since the goal differential during the interval is not constant. With respect to the home team, we observe an interesting pattern with a slight increase in the median value of AP from $GD = -2$ to $GD = -1$, from $GD = -1$ to $GD = 0$, from $GD = 0$ to $GD = 1$, followed by a drop in pace at $GD = 2$. Note that due to the home team advantage, $GD = 2$ is a more common situation than $GD = -2$. Our nuanced intuition corresponding to these observations begins with the case $GD = -2$ where the home team is losing badly. In this case, we expect that the road team is playing defensively as argued by Guan, Cao and Swartz (2002). The home team is therefore dominant in their offensive zone (i.e. near the road team's goal). On average, there is little room downfield for the home team, and consequently, they will be unable to make significant positive contributions to AP, and the AP measurement (as observed), will be low. As GD changes from -2 through 1, we would expect the road team to play less defensively, and as previously argued, AP will increase (as observed). However, a different behavioural mechanism occurs when $GD = 2$. In this case, the home team has a dominant lead. Their lead is so great, that they have little fear of losing. Hence, when $GD = 2$, the home team is not playing ultra-defensive (i.e. largely contained in their own zone, with predominantly long passes having a high AP contribution). Rather, when $GD = 2$, the home team is playing free, and this causes a reduction in AP from $GD = 1$ to $GD = 2$.

Using a one-way ANOVA design for testing differences in pace across goal differentials, we obtain significant test results with p-values of 0.041 and 0.028 for GP and AP, respectively. This leads to the following insight.

Insight E: Teams play at different pace levels depending on the goal differential.

4.5 Causal Analysis

In this section, we return to our primary question whether it is advantageous to play with pace. With a sensible definition of pace and the availability of tracking data, the issue can be addressed.

Recall that questions of cause and effect are traditionally addressed using randomization in experimental contexts. For our problem, this would require the random assignment of

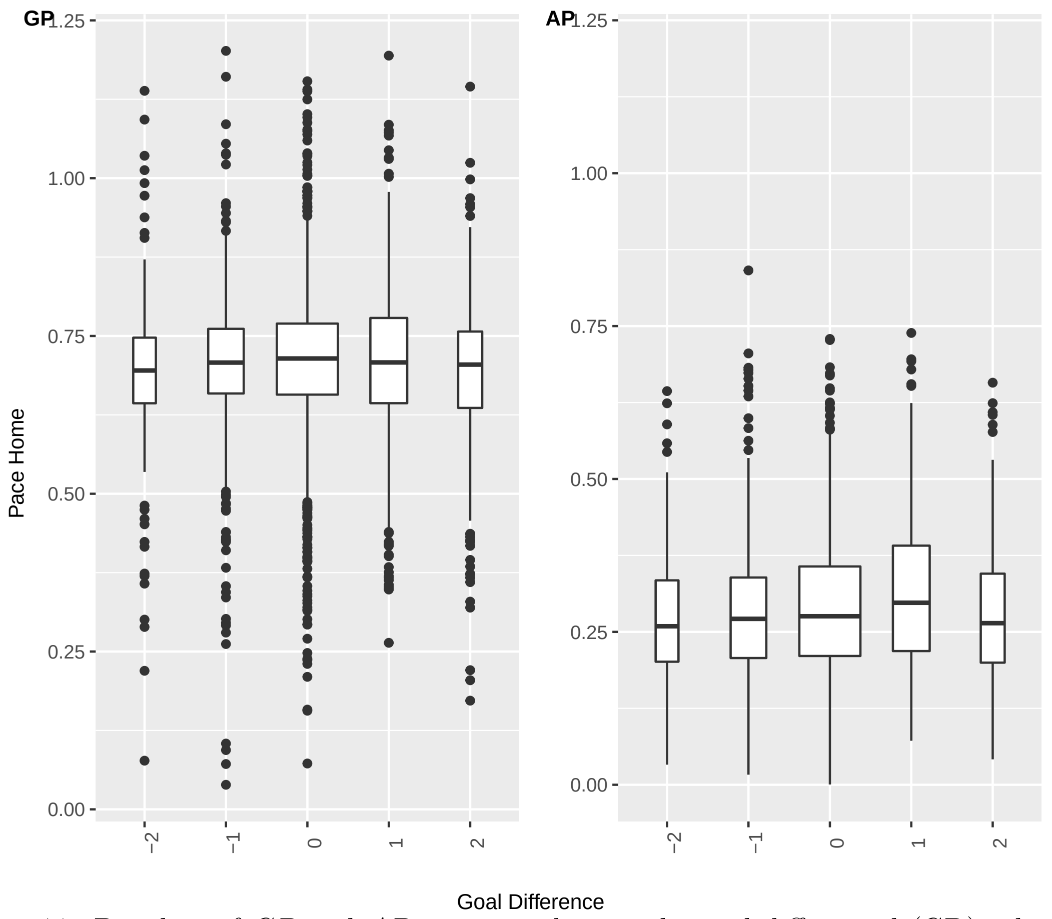


Figure 4.7: Boxplots of GP and AP corresponding to the goal differential (GD) taken at 5-minute intervals where -2 corresponds to the home team losing by 2 or more goals, -1 corresponds to the home team losing by 1 goal, 0 indicates a tied match, 1 corresponds to the home team winning by 1 goal and 2 corresponds to the home team winning by 2 or more goals.

pace to the two teams. Of course, matches are not experiments, but rather observational studies where randomization does not occur. Therefore, we address cause and effect through methods of causal inference (Pearl 2009). Although causal inference has received great attention, the methods are often difficult to implement due to the necessity of specifying and measuring relevant confounding variables. Fortunately, sport is much simpler in its objectives than many other scientific domains, and via the spatio-temporal tracking data and the EDA investigations of Section 4.2, confounding variables are accessible. Therefore, together with some novel ideas, and referring to the approach introduced in Wu et al. (2021), we are able to address cause and effect associated with pace.

4.5.1 Propensity Scores

Using causal terminology, we think of pace as the treatment which we denote X_h and X_r, corresponding to the home and road teams, respectively. We denote W as the vector of confounding variables which we believe are predictive of the pace $X = (X_h, X_r)'$. With this structure, we wish to specify propensity scores $\text{Prob}(X_h - X_r > 0 \mid W)$ that describe the probability that the home team plays at greater pace than the road team given the relevant circumstances of the match. With insights gained from the EDA of Section 4.2, we specify a statistical model that leads to propensity scores. For reference, we define all of our relevant variables below.

$$\begin{array}{rl} t & \equiv \text{time of the match in minutes, } \; t \in (0, 90) \\ X(t) & \equiv \text{pace vector for home and road teams at time } t\text{; either } GP \text{ or } AP \\ GD(t) & \equiv \text{goal differential in favour of the home team at time } t \\ O & \equiv \text{pre-match betting odds corresponding to the home team} \\ Y((t_1, t_2)) & \equiv \text{excess shots by home team compared to road team during } (t_1, t_2) \end{array} \tag{4.4}$$

Looking ahead, our interest is in determining a cause-effect relationship regarding the impact of pace X on success Y. A natural success variable would be goals. However, in soccer, goals are rare events with less than three goals per game on average in top professional leagues. We therefore use the surrogate variable shots as defined in (4.4) to assess success. Of course, not all shots lead to goals but shots are an indication of success. However, let's return to the first step of the causal investigation which involves the construction of a propensity score model.

We first bin the data to define levels for each of the three confounding variables $W(t) = (t, GD(t), O)'$. We segment the time t into 18 five-minute intervals: $(0, 5)$, $(5, 10)$, $\ldots$, $(85, 90)$. We do not include added time beyond 90 minutes since the amount of added time differs across matches.

For the second variable, we restrict $GD(t)$ to five states with goal differentials -2, -1, 0, 1 and 2 corresponding to the home team at time t. Note that $GD(t) = -2$ corresponds to the home team losing by two or more goals and that $GD(t) = 2$ corresponds to the home team winning by two or more goals. For a given match, we consider each of the 18 time intervals, and if the goal differential is constant throughout the interval (either -2, -1, 0, 1 or 2), then an observation is recorded.

For the third variable O, we access pre-match betting odds available from the website https://www.oddsportal.com/soccer/china/super-league-2019/results/ . The betting odds (reported in decimal format) provide us with the relative strength of the two teams. Ignoring the vigorish imposed by the bookmaker, the interpretation of betting odds o for a team is that the team has a pre-match probability $1/o$ of winning the match. Therefore, values of o slightly greater than 1.0 indicate a strong favourite whereas large values of o indicate an *underdog*. For a given match, we define four bins for the decimal odds of the home team: [1.3,1.7), [1.7,2.3), [2.3,3.0) and [3.0,8.0). The odds are restricted so that only competitive matches are included, and the endpoints are selected to provide comparable numbers of observations across bins. Note that the bettings odds O do not depend on the time t.

The variable O was obtained using the standard three-way betting odds for soccer corresponding to home wins, draws and losses. Ideally, relative strength would be better measured with *moneyline* odds corresponding to wins where wagers corresponding to draws are refunded. The reason why three-way betting odds are not ideal is that two matches can have identical win odds yet different draw and loss odds. However, the difference in odds in these two situations is typically minor.

For the response variable in the propensity score model, we calculate $X_h(t)$ and $X_r(t)$ during the time interval which is intended to convey the style of pace over the time period. We use attacking pace AP for the pace calculation, as it is more definitive and perhaps more interesting that general pace GP. We also emphasize that the response variable $X(t) = (X_h(t), X_r(t))'$ is bivariate which makes the causal investigation nonstandard.

To illustrate the variables in the propensity score model, consider a match where the score is 2-0 just prior to the 70-th minute. Following conventional notation where the first team in the scoreline is the home team, this implies that the home team is leading by two goals. Assume further that the home team is the favoured team with pre-match decimal betting odds $o = 1.5$. In this match, suppose that neither team scores during the time interval $(65, 70)$ minutes, and that the AP statistics for the home and road teams during this period are 2.34 and 2.07, respectively. Then, for this time interval, we have the observed response $X = (2.34, 2.07)$ and covariates $W = (14, 2, 1)$ where $t \in (65, 70)$ corresponds to the 14th time category, $GD = 2$ is the goal differential (categorical), and odds $o = 1.5 \in (1.3, 1.7)$ corresponds to the first category.

Based on the above considerations, we have 3679 observations recorded across $18 \times 5 \times 4 = 360$ cells. For linear models based on categorical data, it is prudent to have adequate numbers of observations in each cell. For this reason, we consider a reduction in the number of cells by instead defining six time categories, $(0, 15), (15, 30), \ldots, (75, 90)$ minutes. In this case, we have 944 observations recorded across $6 \times 5 \times 4 = 120$ cells. The cell counts are provided in Table 4.1. In most cells, we have the recommended minimum number of five counts per cell; exceptions tend to occur with large goal differentials (i.e. $GD = -2$ and $GD = 2$), especially early in matches.

Our propensity score model is a multivariate analysis of variance (MANOVA) model where the response variable X is two-dimensional and the covariate $W = (t, GD, O)$ has $6 \times 5 \times 4 = 120$ cells (as described above). The MANOVA model is preferred to two separate ANOVA models for X_h and X_r since the MANOVA model permits a covariance structure between X_h and X_r. Details on MANOVA models are given by Smith, Gnanadesikan and Hughes (1962).

We used MANOVA software using the `manova` function in the `Stats` R package (R Core Team 2022). One of the assumptions of MANOVA concerns the normality of observations. A quantile plot of the residuals does not suggest any serious departures from normality. In Table 4.2, we present the results of fitting the MANOVA model where we have allowed for

	$GD=-2$				$GD=-1$				$GD=0$			
O	[1.3,1.7)	[1.7,2.3)	[2.3,3)	[3,8)	[1.3,1.7)	[1.7,2.3)	[2.3,3)	[3,8)	[1.3,1.7)	[1.7,2.3)	[2.3,3)	[3,8)
$t \in (00,15)$	0	0	0	2	0	0	0	3	48	51	33	37
$t \in (15,30)$	0	0	0	3	6	3	2	8	32	32	19	24
$t \in (30,45)$	1	0	0	3	6	3	4	5	24	24	17	20
$t \in (45,60)$	0	0	2	8	5	5	6	7	18	18	11	16
$t \in (60,75)$	1	1	3	13	3	4	3	12	15	15	7	7
$t \in (75,90)$	1	3	4	7	3	4	4	11	9	16	6	8

	$GD=1$				$GD=2$			
O	[1.3,1.7)	[1.7,2.3)	[2.3,3)	[3,8)	[1.3,1.7)	[1.7,2.3)	[2.3,3)	[3,8)
$t \in (00,15)$	6	2	1	2	0	0	0	0
$t \in (15,30)$	14	6	5	7	2	1	0	0
$t \in (30,45)$	17	9	7	10	6	2	1	0
$t \in (45,60)$	18	14	5	8	12	1	2	2
$t \in (60,75)$	12	16	8	5	14	5	4	3
$t \in (75,90)$	13	9	6	4	16	3	3	1

Table 4.1: Cell counts for the $6 \times 5 \times 4$ covariate categories where the categories correspond to the time t, the goal differential GD and the betting odds O.

the possibility of first-order interaction terms. The main takeaway is that the time of the match t, the goal differential in favour of the home team GD and the relative strength of the home team O are strongly associated with attacking pace X. There is also mild evidence of some first-order interactions involving t, GD and O.

Variable	Df	Pillai	approx F	num Df	den Df	Pr($> F$)
t	5	0.062411	5.7077	10	1772	1.801e-08 ***
GD	4	0.075761	8.7208	8	1772	9.575e-12 ***
O	3	0.126651	19.9665	6	1772	<2.2e-16 ***
$t{*}GD$	18	0.050637	1.2786	36	1772	0.12531
$t{*}O$	12	0.054758	2.0784	24	1772	0.00164 **
$GD{*}O$	15	0.035338	1.0624	30	1772	0.37509
Error	886					

Table 4.2: Results from the MANOVA which relates pace X to the covariates $W = (t, GD, O)$.

Finally, we need to induce the required probability $\text{Prob}(X_h - X_r > 0 \mid W)$ from the fitted MANOVA model. The calculation is based on a simple result from mathematical statistics using properties of the normal distribution. For example, for a given match situation W, suppose that the MANOVA model yields $X \sim \text{Normal}_2(\mu, \Sigma)$ where $\mu = (\mu_1, \mu_2)'$ and $\Sigma = (\sigma_{ij})$. Then $\text{Prob}(X_h - X_r > 0 \mid W) = \Phi((\mu_1 - \mu_2)/\sqrt{\sigma_{11} + \sigma_{22} - 2\sigma_{12}})$ where Φ is the cumulative distribution function of the standard normal distribution. Note that the bivariate normal parameters are estimated through the fitting of the MANOVA model. For example, the estimated values of σ_{11}, σ_{22} and σ_{12} are 0.0051, 0.0061 and 0.0003, respectively. Therefore, the estimated correlation between home and road attacking pace is $\sigma_{12}/(\sqrt{\sigma_{11}\sigma_{22}}) = 0.054$ which indicates that the MANOVA formulation (which takes into account the relationship between home and road pace) provides only a slight improvement over ANOVA.

4.5.2 Matching and Results

In the most basic randomized experiment, an experimenter randomly assigns M subjects from a population to receive a treatment and M subjects from the population to receive the control. The hope is that through random assignment, the treatment group will on average be similar to the control group, and that differences in the response between the two groups can be attributed to the treatment.

The use of propensity scores and matching (Austin 2011, Imbens 2004) attempts to mimic the basic randomized experiment in the context of observational studies. A propensity score for a subject in a clinical trial is the probability that the subject receives the treatment. In the pace problem, $\text{Prob}(X_h - X_r > 0 \mid W)$ is the estimated probability that the home team will play at a higher pace than the road team. Therefore, $\text{Prob}(X_h - X_r > 0 \mid W)$ serves as the relevant propensity score in the pace application.

In our problem, we have a dataset involving 944 pace observations (see Section 4.5.1) resulting in $M_1 = 450$ cases where the home team plays at greater pace (the treatment) and $M_2 = 494$ cases where the home team plays at lesser pace (the control). Since $M_1 < M_2$, the matching idea is that we attempt to match each of the M_1 treatment cases with a corresponding control case so that each pair has a similar estimated propensity score based on the underlying match circumstances W. Then the resulting two groups (M_1 treatments and M_2 controls) will be similar in the match characteristics, and that differences between the two groups can be attributed to the treatment (i.e. pace).

There are many ways that the matching of propensity scores can be carried out (Stuart 2010), and caution ought to be exercised in the process. In our application, we begin with the M_1 cases where the home team plays at a greater pace, and we use a nearest neighbor method for selecting the matched cases where the home team plays at a lesser pace. Specifically, we use the *Matching* package (Sekhon 2011) in the statistical programming language R (R Core Team 2022) to randomly select (with replacement) control cases that fall within a specified tolerance of the propensity scores for the treatment cases. Sampling with replacement tends to increase the quality of matching when compared to sampling without replacement. Unlike deterministic matching procedures, the random aspect of the nearest neighbor procedure allows us to repeat analyses to check the sensitivity of the inferences.

Following the implementation of the matching procedure, Figure 4.8 displays the balance between the two groups with respect to the propensity scores. The similarity in the histograms is important as it provides confidence that the two groups are similar according to the characteristics that affect whether the home team plays at greater pace.

The inferential component of the investigation begins with a simple paired two-sample test between the two groups based on the response Y (excess shots by the home team) as described in (4.4). Again, we prefer to use shots rather than goals since goals are rare events. The quantity of interest is the average treatment effect $\text{ATE} = \bar{Y}(1) - \bar{Y}(0)$ where $\bar{Y}(1)$ is

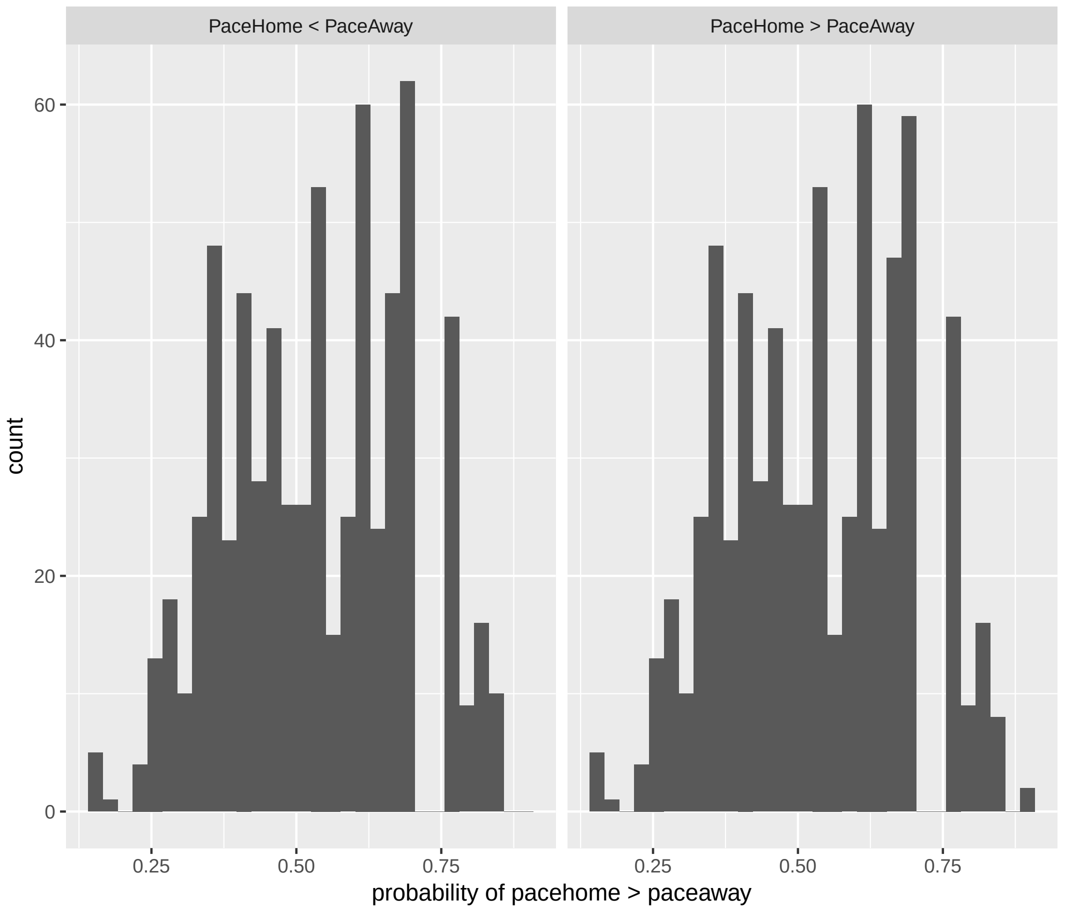


Figure 4.8: After matching, histograms of the two groups (treatment and control) are depicted where the horizontal variable is the propensity score.

the excess number of resultant shots by the home team when they are playing at greater pace, and $\bar{Y}(0)$ is the excess number of resultant shots by the home team when they are playing at lesser pace. We obtain ATE $= 0.73 - 0.41 = 0.32$ with standard error 0.103. The result is significant and suggests that pace is beneficial in the sense of playing at a higher attacking pace.

To put the above result into context, suppose that the home team outpaces the road team during all six 15-minute intervals during the match. Then, we would expect the home team to have roughly $6(0.32) = 2$ more shots during the match than the road team. Note also that we have been careful to distinguish the home and road teams. If we flipped the analysis to consider the average treatment effect due to the road team playing at pace, we would obtain ATE $= -0.41 - (-0.73) = 0.32$. Therefore, the benefit of outpacing the opposition applies to either team.

In Figure 4.9, we present a more nuanced view of the situation. For each group (treatment and control), we smooth the variable Y with respect to the propensity score. We observe that as the propensity score increases (i.e. conditions become more favourable for the home team to play at greater pace), the excess shots for the home team increases for both groups. We also observe that the excess shots by the home team remains relatively constant across the two groups as the propensity score increases. In practice, this means that the advantage of playing at pace persists no matter the circumstances that dictate whether a team should play at pace.

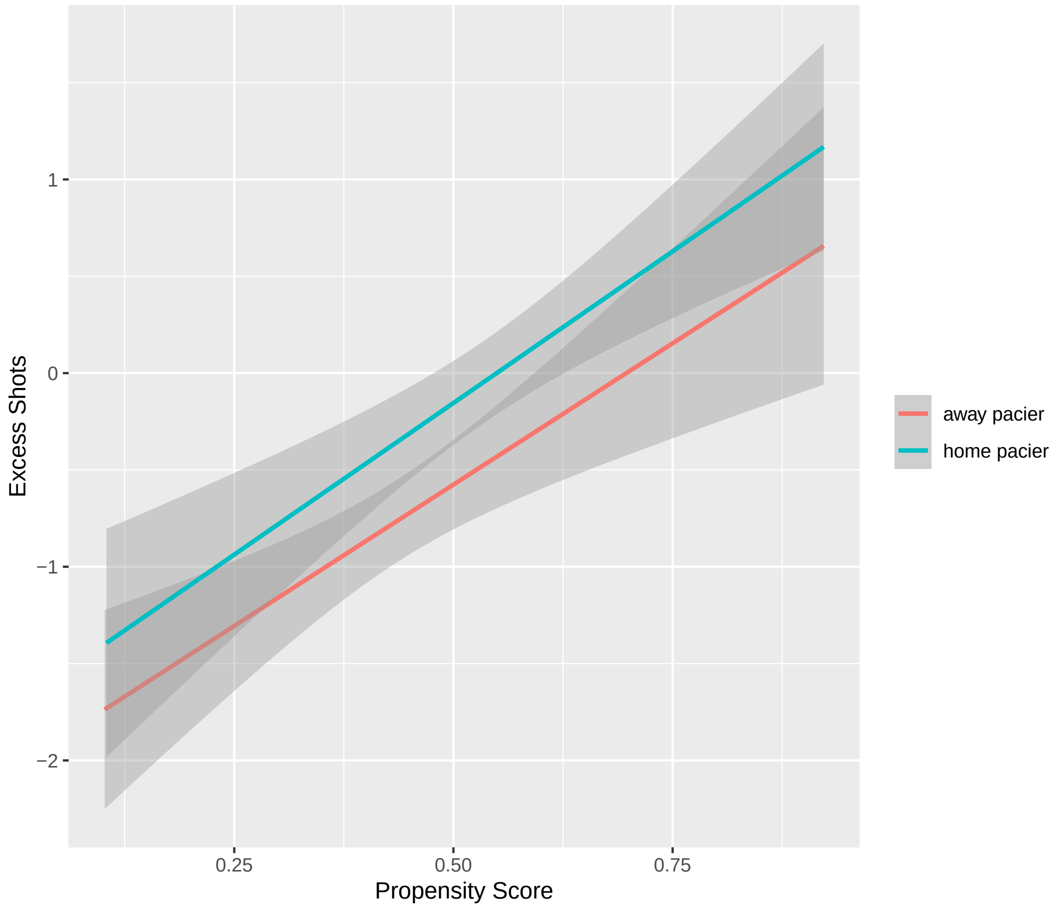


Figure 4.9: After matching, smoothed plots of the excess shot variable Y for the home team with respect to the propensity score under the treatment (blue) and the control (red).

Therefore, the takeaway message is that playing with pace is a good strategy. It leads to more shots for than against. This provides support to Blank's thesis - the Holy Grail of tactics (in Chapter 1 of Blank (2012)) that fast is better than slow.

4.6 Discussion

Despite its importance, style of play is an understudied aspect in team sport. In this chapter, we investigate pace of play as it relates to soccer. Although the analyses were restricted to the study of tracking data in the Chinese Super League, it is conjectured that the broad results hold true for other high-level professional soccer leagues.

In particular, we found that teams that play at higher attacking pace are more advantaged in producing shots than teams that play at lower pace. For a team that outpaces its opponent throughout a match, this translates to roughly two extra shots. The conclusion was facilitated through the adaptation of causal methods. In particular, we sought confounding variables that were important in determining propensity scores. Furthermore, the propensity scores were obtained by reducing a bivariate normal distribution to a relevant Bernoulli distribution. The EDA produced additional sporting insights (A-E) related to pace.

There are possible future investigations related to pace of play. For example, we believe that similar analyses may be carried out in other invasion sports where tracking data are available. Also, it may be interesting to analyze pace separately in terms of passing and dribbling. The ball generally moves more quickly when passing, and there may be stylistic differences between teams in terms of how much they pass relative to how much they dribble.

A limitation in our work is that the response variable Y (shots) in the causal analysis correspond to rare events and is known to be noisy. A better response variable may be expected goals (Spearman 2018, Anzer and Bauer 2021), and this could be considered in future investigations. Another limitation of our work is the restriction to matches from the CSL. It would be good to see if the results also hold in top-level European leagues where the best players from all over the world compete. Although we argue that confounding variables can be identified with tracking data, for sure, there are latent variables that we have not utilized (e.g. level of player fatigue). This is also a limitation.

Chapter 5

Causal analysis of tactics in soccer: The Case of Throw-ins

5.1 Introduction

The investigation of cause and effect relationships is a fundamental research topic in both the sciences and the social sciences. Traditionally, cause and effect relationships are studied through experiments where randomization is the primary technical tool for investigation.

In sport, cause and effect relationships are also important. For example, teams and individuals want to know whether particular tactics are effective winning strategies. However, in sport, data do not typically arise through randomized experiments. Rather, data are usually collected from matches, in non-experimental settings.

Fortunately, the development of causal methods (Pearl 2009) has provided opportunities to investigate cause and effect relationships in non-experimental settings. Whereas the identification and measurement of relevant confounding variables is a necessary and challenging component of causal methods, the hurdle appears less imposing in sport. In sport, objectives are often clear (e.g. score more goals than the opponent), matches terminate in reasonable timeframes (e.g. often two to four hours), and rules are well defined. Most importantly, with the advent of detailed player tracking data (e.g. spatio-temporal data), our sporting intuition often permits the identification and measurement of relevant confounding variables.

Causal inference in sport assisted by player tracking data is a relatively new but potentially fruitful research area. Wu et al. (2021) provided a template for such analyses in soccer where the benefit of crossing the ball was investigated. In this investigation, the response variable Y (resultant shot) was binary, and the treatment X (crossing) was binary. Wu et al. (2021) generated conclusions that were contradictory to some of the existing literature, where they indicated that crossing is a valuable tactic. Epasinghege Dona and Swartz (2023) expanded on these ideas to carry out a causal analysis regarding pace of play in soccer. In this investigation, the response variable Y (excess shots) was discrete, and the treatment

X (pace) was bivariate and continuous. Epasinghege Dona and Swartz (2023) established that playing with pace is a valuable strategy, a conclusion that had not been previously established in soccer.

This chapter extends the causal investigations of Wu et al. (2021) and Epasinghege Dona and Swartz (2023). Our analysis investigates the optimal locations of throw-ins in soccer. In this investigation, the response variable Y (resultant shot) is binary, and the treatment X (throw-in reception location) is spatial. Stone, Smith and Barry (2021) have previously studied the throw-in problem using data from the English Premier League where they obtained the surprising result that backward throw-ins are more successful in terms of shot creation. Notably, Stone, Smith and Barry (2021) did not have access to player tracking data. Without tracking data, it is not possible to assess the extent to which the recipient of a throw-in is open. Unlike our chapter, Stone, Smith and Barry (2021) did not carry out a causal analysis.

As mentioned, the availability of player tracking data provides the opportunity for a deep-dive analysis of throw-ins in soccer. With player tracking data, the location coordinates for every player on the field are recorded frequently (e.g. 10 times per second in soccer). With such detailed data, the opportunity to explore novel questions in sport has never been greater. The massive datasets associated with player tracking also introduce data management issues and the need to develop modern data science methods beyond traditional statistical analyses. Gudmundsson and Horton (2017) provide a review of spatio-temporal analyses that have been used in invasion sports where player tracking data are available.

In Section 5.2, we describe the player tracking data and discuss related challenges. We then describe how we construct the throw-in datasets from the tracking data. The throw-in datasets are the source files which are used for the causal analysis. Some exploratory data analyses are also provided. In Section 5.3, we discuss the use of propensity scores in causal investigations. Propensity scores describe the probability of the treatment X (spatial location of throw-in) given underlying covariates W. In Section 5.4, we present a causal analysis concerning the optimal locations of throw-ins relative to the position on the pitch. This is done in three ways. The first two approaches are simple as they are based on defining a binary variable corresponding to backward/forward throw-ins and (2) defining a binary variable corresponding to short/long throw-ins. We then consider a more complex analysis based on the full spatial treatment X. The main result from the analyses is that both backward throw-ins and long throw-ins confer a competitive advantage. We provide some concluding remarks in Section 5.5.

Apart from tactics, there have been many recent investigations in the literature related to soccer. A sample of diverse topics include match fixing (Forrest and McHale 2019), the evaluation of passing (Håland et al. 2020), competitive balance (Manasis, Ntzoufras and Reade 2022) and the forecasting of match results (Hubáček, Šourek and železný 2022).

5.2 Data

For this investigation, we have a big data problem where both event data and player tracking data are available for 237 regular season matches (three matches missing) from the 2019 season of the Chinese Super League (CSL). The schedule is balanced where each of the 16 teams plays every opponent twice, once at home and once on the road.

Event data and tracking data were collected independently where event data consists of occurrences such as tackles and passes, and these are recorded along with auxiliary information whenever an "event" takes place. The events are manually recorded by technicians who view film. Both event data and tracking data have timestamps so that the two files can be compared for internal consistency. There are various ways in which tracking data are collected. One approach involves the use of Radio Frequency Identification (RFID) technology where each player and the ball have tags that allow for the accurate tracking of objects. In the CSL dataset, tracking data are obtained from video and the use of optical recognition software. The tracking data consists of roughly 1.3 million rows per match measured on 7 variables where the data are recorded every 1/10th of a second. Each row corresponds to a particular player at a given instant in time. Although the inferences gained via our analyses are specific to the CSL, we suggest that the methods are applicable to any soccer league which collects tracking data.

5.2.1 The Throw-in Datasets

We pre-processed the CSL tracking and event data. Originally, the data were provided in xml files and we extracted content using the `read_xml` function from the `XML` package using R software. The resulting tracking and event data were written into csv file format.

Ultimately, we constructed throw-in dataframes for each match. These are comprehensive datasets that allows us to investigate various questions of interest related to throw-ins. A throw-in dataframe is a matrix where the rows correspond to throw-ins. Each throw-in has been translated and standardized such that throw-in angles and distances downfield are consistent according to direction that a team is attacking. The columns include the following basic variables: the identification of the throw-in team, the identification of the opponent team, the identification of the player who made the throw-in and the binary variable Y according to whether the end of possession from the throw-in resulted in a shot for the throw-in team. The variable Y serves as the response variable which indicates success related to a throw-in. An end of possession for the throw-in team occurs when the opponent gains possession, a whistle occurs, there is a stoppage in play or when the ball goes out of bounds. Although goals are a more direct measure of success, we note that goals are rare events in soccer with less than three goals per match on average in most professional leagues. We also record the relative spatial location $X = (r, \theta)$ of the received throw-in. By relative spatial location, we mean the length of the throw-in and its radius angle given the location on the field where the throw-in occurred to where the ball was received. The measurement is standardized with respect to the side of the field where the throw-in occurs. Using polar coordinates, the radius arm r is the length of the throw-in measured in metres and θ is the angle of the throw-in measured in degrees. For example, $X = (10, 90)$ describes a throw-in of length 10m that is thrown perpendicular to the touch line.

For the propensity scores described in Section 5.3, we wish to relate covariates W which have a potential impact on the relative spatial location X of the received throw-in. These variables are derived from our soccer intuition and are viewed as confounding variables in the causal analysis. In proposing covariates W, we take a broad perspective and introduce variables that may have even a hint of impacting the spatial locations of throw-ins. We introduce additional column variables $W = (t, d, f, o, b, r)$ to the throw-in dataframes where

$$\begin{aligned}
t \quad &\equiv \text{time of the throw-in in minutes, } \ t \in (0, 90) \\
d(t) \quad &\equiv \text{score differential in favour of the throw-in team} \\
f(t) \quad &\equiv \text{field location of the throw-in, } \ f(t) \in (0, 100) \\
o(t) \quad &\equiv \text{openness of the receiver of the throw-in} \\
b \quad &\equiv \text{pre-match betting odds corresponding to the throw-in team} \\
r \quad &\equiv \text{red card variable corresponding to manpower advantage of throw-in team}
\end{aligned} \tag{5.1}$$

In (5.1), we define the time variable t such that throw-ins that occur during extra time in the first half are set to $t = 45$. For throw-ins that occur during extra time in the second half, we set $t = 90$. Therefore t is a mixed variable (both continuous and discrete).

The score differential $d(t)$ is a discrete variable and expresses the lead by the throw-in team. For example, $d(t) = -2$ indicates that the throw-in team is losing by two goals.

The field location variable $f(t)$ has been standardized to the interval $(0, 100)$ to account for fields of different length. For example, $f(t) = 50.0$ corresponds to a throw-in taken from midfield.

The openness variable $o(t)$ in (5.1) describes the degree to which the receiver of the throw-in is open. An open receiver is more likely to be targeted for a throw-in. To obtain $o(t)$, we first consider the shaded region in Figure 5.1 where only defenders in this region are assumed to pose a threat of intercepting the throw-in. The idea is that defenders who are behind the receiver can be "boxed out" by the receiver, and are not involved. Our experience is that a receiver with a defender on his back will move towards the ball, be able to keep the defender behind, and obtain possession. Thus, such a defender is not a threat to possession. We define a defender as "boxed out" if the defender is situated within 45 degrees from the perpindicular from the throw-in location to the receiver. Admittedly, the 45 degree angle is a bit arbitrary. The variable $o(t)$ is then calculated by taking the distance from the nearest defender in the shaded region to the receiver. In this way, longer distances convey greater openness of the receiver. If the throw-in is intercepted, $o(t) = 0.0$. Openness is related to the more complex notion of pitch control or field ownership. Pitch control was first introduced using Voronoi tesselations (Voronoi 1907, Kim 2004). More advanced metrics for field ownership are discussed in Wu and Swartz (2022).

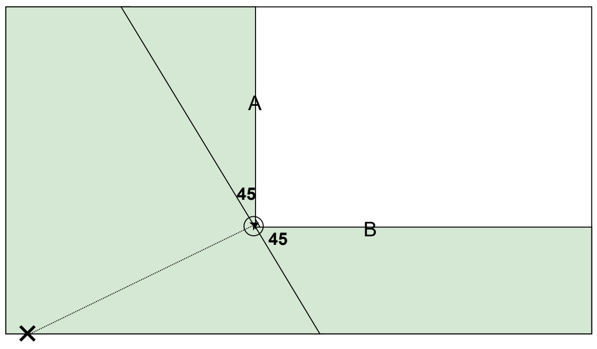


Figure 5.1: The figure depicts the line between thrower (X) and receiver (open dot), and two rays labelled A and B that are situated 45 degrees from the perpindicular to the line. The resultant shaded region corresponds to the area where defenders are assumed to pose a threat of intercepting the throw-in.

For the fifth variable b in (5.1), we accessed pre-match betting odds available from the website https://www.oddsportal.com/soccer/china/super-league-2019/results/ . The betting odds (reported in decimal format and also known as European odds) provide us with the relative strength of the two teams. For simplicity, we temporarily ignore the vigorish imposed by the bookmaker. [1] In this case, the interpretation of betting odds b for a team is that the team has a pre-match probability $1/b$ of winning the match. Therefore, values of b slightly greater than 1.0 indicate a strong favourite whereas large values of b indicate an *underdog*. To better understand betting odds, consider fair odds b, a wager of x dollars and probability p of winning the bet. The expected profit from such a wager is $-x * (1 - p) + (xb - x) * p$ and setting this equal to zero yields $p = 1/b$. Our sporting intuition is that stronger teams may have different throw-in strategies than weaker teams. In general, stronger teams tend to play differently than weaker teams (see for instance, Silva and Swartz (2016)). Figure 5.2 depicts some of the variables described above.

The final variable r is binary and is set according to whether the throw-in team has a manpower advantage. One might expect the defensive team to behave differently (e.g. more players lined up behind the ball) in this scenario.

[1]For the actual analysis, consider bettings odds b_w, b_d and b_l corresponding to a team win, draw and loss respectively. For profitability, the bookmaker introduces a vigorish whereby $1/b_w + 1/b_d + 1/b_l > 1$. Therefore, the implied probability of a win is given by $p = (1/b_w)/(1/b_w + 1/b_d + 1/b_l)$. To measure the strength of a team, we instead base our analysis using betting odds defined as the reciprocal of p.

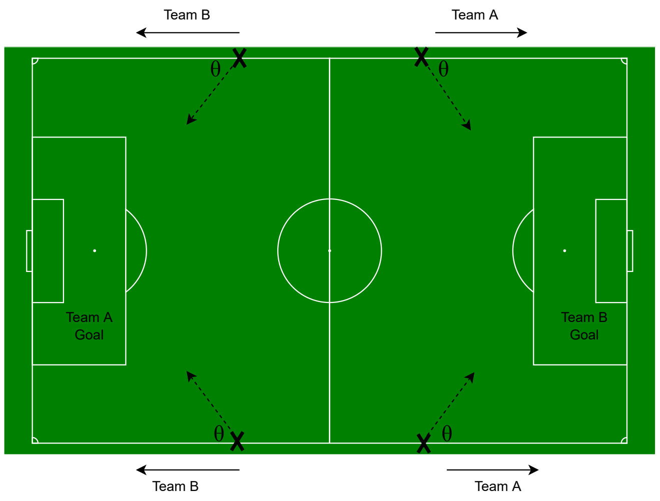


Figure 5.2: The plot illustrates some of the key variables corresponding to throw-ins. Four throw-ins are depicted where the angle θ is standardized accounting for the side of the field and the attacking direction. The four throw-ins each correspond to $r = 15$ metres and $f(t) = 60.0$.

To create the throw-in dataframe, we looped frame by frame through the tracking data, where we matched events and time using the event data. The process required approximately 15 minutes of computation for all 237 matches.

To illustrate the propensity score variables W given by (5.1), consider a match where the score is 1-0. In the 70-th minute, teams are full-strength, a throw-in takes place at midfield for the leading team who are the favoured team with pre-match decimal betting odds $b = 1.5$ (with the vigorish removed). Further, suppose that the nearest defender to the receiver is standing along the sideline 8 metres away from the receiver. In this case, $W = (t, d, f, o, b, r) = (70, 1, 50, 8.0, 1.5, 0)$.

The vector $W = (t, d, f, o, b, r)$ corresponds to our soccer intuition as a driver of the throw-in decision. We did consider other variables which did not have significant effects. For example, the speed of the target receiver was considered and this was obtained by taking the Euclidean distance of the player two frames before and two frames prior to the throw-in. We also experimented with the red card variable r in a categorical setting corresponding to manpower advantage, no advantage and manpower disadvantage. However, only manpower advantage proved significant, and hence it was reduced to a binary variable.

5.2.2 Data Management

With increasingly complex and large datasets in data science, the importance of data integrity cannot be overstated. Without accurate data, reliable inferences cannot be achieved. In this application, we constructed the throw-in datasets from the event and tracking data where the following issues presented challenges.

- Some throw-ins were identified as impossibly short (i.e. 0 metres in length). These were a consequence of foul throws (59 occurrences) and were removed from the dataset.
- Some throw-ins were identified as impossibly long (i.e. exceeding 40 metres in length). These were a consequence of an event happening during the throw-in which invalidated the throw-in (e.g. a foul), and from which the next event took place at a location other than the free-throw location. These events were rare (13 occurrences) and were removed from the dataset.
- In our event dataset, a throw-in is labelled before the throw-in occurred, and therefore, we should consider the next event to obtain the location where the throw-in was received. There are some events like player substitutions, red/yellow cards that occur before the throw-in. Therefore, we had to carefully eliminate these events before calculating the throw-in length r and angle θ.
- There were two throw-ins where the target receiver was not identified.

From the original 8467 throw-ins occurring in the matches, 8393 were useable for data analysis.

5.2.3 Exploratory Data Analysis

Exploratory data analyses often guide the development of formal statistical models. We present several plots related to our investigation.

In Figure 5.3, we provide a histogram of the direction variable θ associated with all throw-ins in the dataset. We observe that there are more forward throw-ins (i.e. $\theta < 90$) than backward throw-ins (i.e. $\theta > 90$). This corresponds to our intuition since the attacking team typically wishes to advance the ball downfield to a more threatening scoring position. The symmetric modes that we observe at approximately $\theta = 22.5$ and $\theta = 157.5$ are interesting. These throw-ins are close to the touch line.

In Figure 5.4, we provide a histogram of the radius arm r (or simply the length) associated with all throw-ins in the dataset. We observe a right-skewed histogram where the modal throw-in length is roughly 10 metres. There are several long throw-ins of approaching 40 metres in length. Whereas such throw-in distances may seem unlikely, these throw-ins may be the result of a ball that was not initially received and travelled for a period.

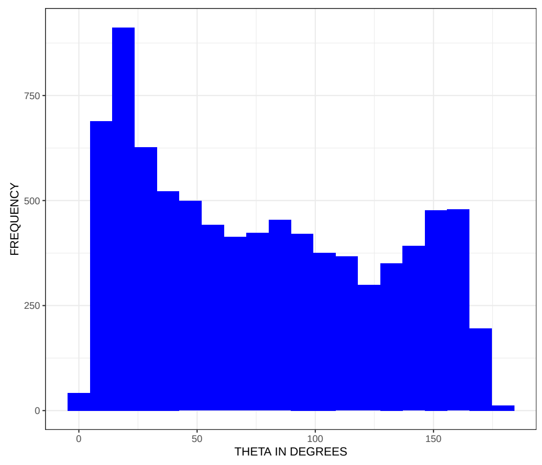


Figure 5.3: The histogram of the variable θ describes the throw-in angle relative to the sideline in the direction that the team is attacking.

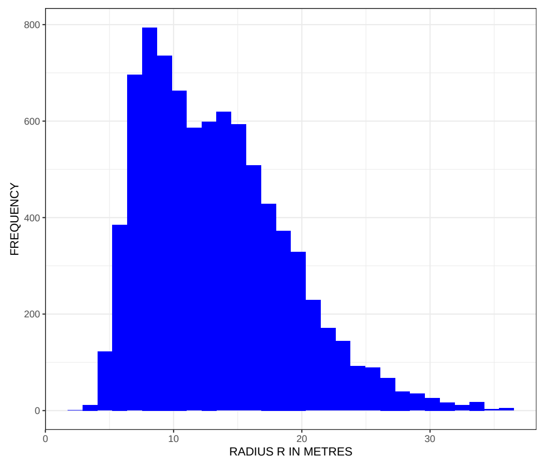


Figure 5.4: The histogram of the variable r describes the throw-in length.

In Figure 5.5, we provide a histogram of the openness variable o which gives the distance from the receiver to the receiver's nearest opponent who is positioned in the shaded region of Figure 5.1. We observe that the median distance is roughly 10 metres and that the histogram is right-skewed. From a practical point of view, a player is comfortably open whether $o = 10$ metres or $o = 30$ metres, for example. Note that there were 126 cases in the throw-in dataset where there were no opponents in the shaded area (see Figure 5.1). These observations are not reflected in Figure 5.5, although they were retained for the causal analysis.

5.3 Propensity Scores

Imagine temporarily that the throw-in problem was designed as a randomized experiment. For each throw-in, we would randomize the location $X = (r, \theta)$ of the received throw-in, and we would then relate Y (whether the completion of the possession resulted in a shot) to X.

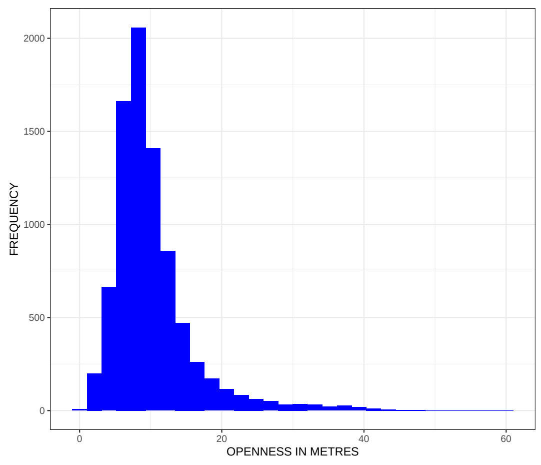


Figure 5.5: The histogram of the openness variable o describes the degree to which the receiver is open (see formal definition in Section 5.2.1).

This would allow us to determine an optimal X. With randomization, the idea is that the underlying conditions leading to X would be nearly uniform across different realizations of X.

Of course, with match data, X is not randomized. And it is quite possible that not only does X depends on the covariate W in (5.1) but also Y depends on W. In other words, W is a confounding variable when investigating the relationship between Y and X.

We therefore wish to obtain propensity scores $\mathrm{P}(X \mid W)$ that describe how the probability of the treatment X (i.e. relative spatial location of the throw-in) is related to the confounding variable W. If we are able to do this, then through matching, we can compare Y_1 under a treatment X_1 relative to Y_2 under a treatment X_2 if X_1 and X_2 have similar propensity scores. This is the essential logic of the causal approach where propensity scores are used as a substitute for randomization.

Whereas we utilize a propensity score matching (PSM) approach, it is also possible to carry out analyses based on weighted propensity scores (PSW). The latter has the advantage that all observations can be used in the analysis. Narita, Tena and Detotto (2023) provide an insightful tutorial on the use of propensity score analyses with particular attention to PSW analyses.

We consider three causal analyses in Section 5.4: (1) backward throw-ins versus forward throw-ins, (2) long throw-ins versus short throw-ins and (3) a composite analysis based on the full spatial variable $X = (r, \theta)$. The propensity score models that we use in these three analyses are logistic, logistic and random forests, respectively.

5.4 Causal Analyses

We return to the primary question concerning the optimal locations involving throw-ins. In Section 5.3, we have developed a propensity score model which yields scores $\mathrm{P}(X \mid W)$. Our objective now is to investigate the causal relationship between the binary response variable Y (whether the throw-in possession results in a shot) and the spatial treatment variable $X = (r, \theta)$.

We present three analyses where the first two analyses address the following simple questions: (1) Is a forward throw-in preferable to a backward throw-in? (2) Is a long throw-in preferable to a short throw-in? For the third analysis, we use more sophisticated methods to investigate the impact of the full spatial variable $X = (r, \theta)$ on Y.

For the purposes of the simple causal analyses in Sections 5.4.1 and 5.4.2, we make some adjustments to the confounding variable $W = (t, d, f, o, b, r)$ presented in equation (5.1). First, we discretize the time variable t according to the two categories $t < 45$ minutes and $t > 45$ minutes. This has been done since we are doubtful that the corresponding response variables are linear with respect to t, and we believe that the two halves of a match reflect different playing styles. We also categorize the score differential to $d(t) = -2, -1, 0, 1, 2$ where $d(t) = -2$ indicates that the throw-in team is losing by a large margin (two or more goals) and $d(t) = 2$ indicates that the throw-in team is winning by a large margin (two or more goals). With respect to goal differential, we attempted expanding the categories to $d(t) = -3, -2, -1, 0, 1, 2, 3$. However, we observed several insignificant effects and we believe this was due to few observations corresponding to the cells $d = -3$ and $d = 3$. We discretize the field location variable f according to $f < 67$ and $f \geq 67$ since there are tactical differences in the final third of the pitch. Third, we truncate the openness variable o such that values of $o > 10$ metres are set according to $o = 10$. This is done because we believe there is a meaningful difference in openness between $o = 1$ metres and $o = 2$ metres, for example. However, for $o > 10$, all throw-in receivers are effectively open. We did experiment with different thresholds of openness (e.g. $o > 12$ metres) but found that this made little difference in the causal analyses.

5.4.1 Causal Analysis based on Throw-in Direction

We simplify the problem involving the spatial causation variable $X = (r, \theta)$ to a binary context such that the control $0 < \theta < 90$ corresponds to a forward throw-in and the treatment $90 < \theta < 180$ corresponds to a backward throw-in. Therefore, the corresponding propensity score becomes $P(90 < \theta < 180 \mid W)$ which we fit using logistic regression. In this framework, there are $n_0 = 5023$ control observations and $n_1 = 3370$ treatment observations. Of course, there may be other classes of interest with respect to throw-in direction (e.g. sideways throw-ins); the full spectrum of throw-in directions are analyzed in Section 5.4.3.

In Table 5.1, we provide the results of logistic regression based on the variable $W = (t, d, f, o, b, r)$ described in Section 5.2.1. We observe that the time t of the match is significant where more throw-ins go backward as the time progresses. This may be a function of teams tiring and less willing to move forward up the pitch. The goal differential d is highly significant where we observe that greater leads are associated with forward throw-ins. This may be a consequence of the leading team playing better, having more confidence and energy, and consequently moving downfield more frequently. The throw-in position variable f is highly significant. As the throw-in location moves into the attacking third, there are more backwards throw-ins due to the constraints of the endlines. The openness variable o is significant and this aligns with our intuition. With teams defending their own goal, we expect more openness with backwards throw-ins. The betting odds variable b is highly significant and indicates that weaker teams tend to have more forward throw-ins. This is an interesting result and may be explained that these teams have less confidence and perhaps feel that the only way for them to succeed is to move forward. Conversely, stronger teams are typically more comfortable on the ball, willing to build up play through structured passing and possession, and thus, more likely to throw backwards. With respect to the red card variable, we observe that the throw-in team (with a manpower advantage) tends to have more backward throw-ins. This may be explained by the defensive team playing a more cautious style with more defenders behind the ball.

Variable	Estimate	Std Error	p-value
intercept	-0.184	0.134	0.171
time $t(45, 90)$	0.108	0.050	0.030 *
goal differential $d(-1)$	-0.208	0.091	0.021 *
goal differential $d(0)$	-0.575	0.088	5e-11 ***
goal differential $d(1)$	-1.185	0.101	2e-16 ***
goal differential $d(2)$	-1.250	0.122	2e-16 ***
field location $f(\geq 67)$	0.706	0.047	2e-16 ***
openness o	0.026	0.011	0.018 *
betting odds b	-0.050	0.008	5e-10 ***
red card r	0.641	0.176	3e-04 ***

Table 5.1: Results from logistic regression in Section 5.4.1 which determines the propensity scores $P(90 < \theta < 180 \mid W)$.

Since $n_1 < n_0$, the matching concept (Austin 2011, Imbens 2004) is that we attempt to match each of the n_1 treatment cases with a corresponding control case so that each pair has a similar estimated propensity score based on the underlying match circumstances W. Then the intention is that the resulting two groups (controls and treatments) are similar in the match characteristics, and the differences between the two groups can be attributed to the treatment (i.e. backward throw-in). There are many ways that the matching of propensity scores can be carried out (Stuart 2010). For example, matching may be carried out either with or without replacement. Matching may also be greedy (where each treatment case is matched with the closest eligible control case) or performed to optimize some global criterion. Further, randomization can be introduced in the matching procedure so that sensitivity due to the matching can be assessed. There are some downsides of propensity score matching (Guo, Fraser and Chen 2020). For example, with unequal treatment and control groups, matching results in a loss of data. Also, propensity score matching reduces the dimensionality of the covariate vector to a single dimension.

In our application, we begin with the n_1 cases where the throw-ins are backward, and we use a nearest neighbor method for selecting the matched cases where the throw-ins are forward. Specifically, we use the *Matching* package (Sekhon 2011) in the statistical programming language R to randomly select (with replacement) control cases that fall within a specified tolerance of the propensity scores for the treatment cases. Sampling with replacement tends to increase the quality of matching when compared to sampling without replacement. Unlike deterministic matching procedures, the random aspect of the nearest neighbor procedure allows us to repeat analyses to check the sensitivity of the inferences. We repeated the analyses 1000 times. To get a sense of the matching, for each of the 1000 analyses, we recorded the maximum absolute difference in propensity scores. This quantity was then averaged over the 1000 analyses and yielded the difference 0.0048.

We tested for balance in the covariate distributions $W = (t, d, f, o, b, r)$ of the matched treatment and control groups using the two-sample t-test (Rosenbaum and Rubin 1985). For a particular matching (selected randomly), the p-values corresponding to t, d, f, o, b, r were 0.222, 0.396, 0.116, 0.962, 0.190 and 0.640, respectively. The lack of significance suggests that there is balance in the matching across the confounding variables.

Following the implementation of the matching procedure, we calculate the average treatment effect ATE $= \bar{Y}(1) - \bar{Y}(0)$ where $\bar{Y}(1)$ is the average number of resultant shots from a backward throw-in and $\bar{Y}(0)$ is the average number of resultant shots from a forward throw-in. We obtained ATE $= 0.018$ with standard error 0.006 leading to p-value 0.001. This was based on the 1000 iterations of the matching procedure using $n_1 = 3370$ matched pairs. The result is significant and suggests that backward throw-ins are beneficial. This corroborates the findings of Stone, Smith and Barry (2021). Specifically, our causal analysis indicates that from 100 backward throw-ins, roughly two more will result in a shot than if the throw-ins had been forward. This is a meaningful result in terms of gaining a competitive advantage.

In Figure 5.6, we present a more nuanced view of the situation for a randomly selected case involving matching. For each group (treatment and control), we smooth the variable Y with respect to the propensity score. On average, under our model's specifications, we observe that there is no advantage to executing a forward throw-in. As the propensity scores increase (i.e. conditions more favorable to making a backward throw-in), the benefit of the backward throw-in (in terms of shots) increases compared to making a forward throw-in.

This implies that players tend to make the correct decisions with respect to the direction of throw-ins. As backward throw-ins become more probable, backward throw-ins have higher probabilities of successful outcomes.

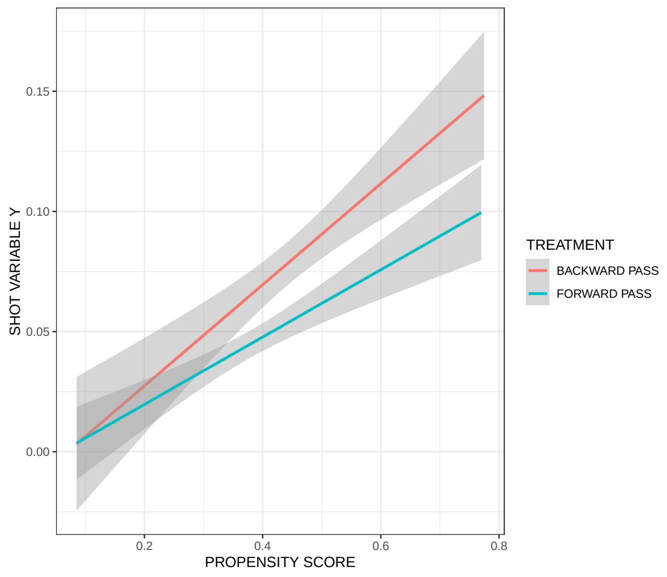


Figure 5.6: Smoothed plots of the shot variable Y with respect to the propensity score for backward throw-ins (treatment red) and for forward throw-ins (control blue).

5.4.2 Causal Analysis based on Throw-in Length

In this section, we consider a second inferential question involving the length of throw-ins. In recent years, the "long" throw-in has gained popularity in professional soccer, and we wish to investigate whether the long throw-in confers an advantage. We again simplify the problem involving the causation variable $X = (r, \theta)$ to a binary context. In this case, we define the treatment of a long throw-in as $r > 15$ metres and $60 < \theta < 120$. The dual condition is imposed so that long throw-ins do not correspond to throw-ins along the touchline (i.e. θ near 0 or 180). Rather our interest is focused on long throw-ins that are directed towards the middle of the pitch. Such long throw-ins appear to be an increasingly common tactic. Therefore, the corresponding propensity score becomes $P((r > 15) \cap (60 < \theta < 120) \mid W)$ which we fit using logistic regression. In this framework, there were $n_0 = 7831$ control observations and $n_1 = 562$ treatment observations.

In Table 5.2, we provide the results of logistic regression based on the variable $W = (t, d, f, o, b)$. Note that the red card variable r was not included in the analysis of Table 5.2 since it was not statistically significant. In this analysis, we do not have as many statistically significant terms as in Table 5.1. However, we do note that the coefficient estimates generally correspond to our soccer intuition. For example, we expect more longer throw-ins in the second half where one of the teams may be desperate and in need of a goal. This same pattern appears with respect to the goal differential where a team trailing by one goal (desperate) is more likely to make a long throw-in. When a team is tied or leading (i.e. $d = 0, 1, 2$), they are less likely to make a long throw-in. The positive coefficient for f is also sensible as long throws are more common in the attacking third. The openness variable o is highly significant; in order to retain possession on the throw-in, it is reasonable that long throw-ins require receivers to be more open than with short throw-ins.

Variable	Estimate	Std Error	p-value
intercept	-3.613	0.274	2e-16 ***
time $t(45, 90)$	0.355	0.095	2e-04 ***
goal differential $d(-1)$	0.116	0.161	0.472
goal differential $d(0)$	-0.113	0.160	0.477
goal differential $d(1)$	-0.602	0.194	0.002 **
goal differential $d(2)$	-0.444	0.228	0.052
field location $f(\geq 67)$	0.212	0.089	0.017 *
openness o	0.111	0.023	2e-06 ***
betting odds b	-0.024	0.015	0.135

Table 5.2: Results from logistic regression in Section 5.4.2 which determines the propensity scores $P((r > 15) \cap (60 < \theta < 120))$.

We carry out the matching procedure as described in Section 5.4.1. To check the sensitivity of the inferences, we repeated the analyses 1000 times. For each of the 1000 analyses, we recorded the maximum absolute difference in propensity scores. This quantity was then averaged over the 1000 analyses and yielded the difference 0.0002. We then obtained the average treatment effect ATE $= \bar{Y}(1) - \bar{Y}(0) = 0.042$ with standard error 0.016 and corresponding p-value 0.004. This was based on the 1000 iterations of the matching procedure using $n_1 = 562$ matched pairs. Here, $\bar{Y}(1)$ is the average number of resultant shots from a long throw-in and $\bar{Y}(0)$ is the average number of resultant shots otherwise. The result indicates that long throw-ins are beneficial as they lead to approximately four more shots per 100 throw-ins. In this analysis, the p-value is larger than in Section 5.4.1 but is still significant. We note that the distance analysis presented here involves fewer observations than the directional analysis of Section 5.4.1. The result indicates that the recent trend involving more long throw-ins is a sound tactic.

In Figure 5.7, we consider a randomly selected case involving matching. We smooth the variable Y with respect to the propensity score for each group (treatment and control). We again see that professional soccer players are making the correct decisions. As it becomes more likely for executing a longer throw-in, the benefits of doing so increase. Inspecting the propensity scores, we observe that long throw-ins are relatively rare. This suggests that teams may consider increasing the frequency of long throw-ins.

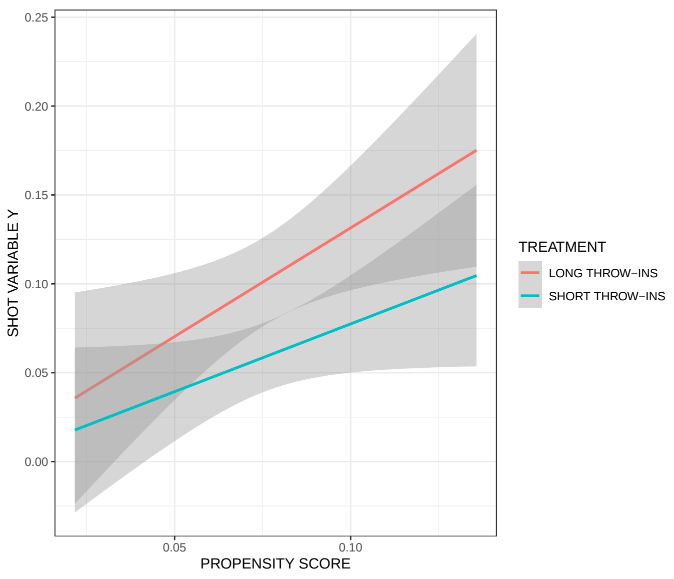


Figure 5.7: Smoothed plots of the shot variable Y with respect to the propensity score for the treatment group involving long throw-ins (red) and for the control group (blue).

5.4.3 Causal Analysis based on Full Spatial Location of Throw-in

In this section, we utilize the full spatial variable $X = (r, \theta)$ to gain insight on the causal relationship between X and the shot variable Y. Gelman and Meng (2004) consider structures beyond the simple binary X as analyzed in Sections 5.4.1 and 5.4.2. Here, we use machine learning methods to obtain the propensity scores.

A rationale for machine learning methods in prediction is that complex phenomena are often difficult to model explicitly. Here, we have a two-dimensional spatial response variable $X = (r, \theta)$, and an explanatory vector W described by (5.1). We have little apriori knowledge about the relationship between X and W. For example, the relationship may only involve a subset of the variables W, the components of W may be correlated, and most importantly, the relationship $X \approx g(W)$ involves an unknown and possibly complex function g. In addition, the stochastic aspect of the relationship is typically unknown.

For this application, we use random forests as the chosen machine learning algorithm. Random forests (Genuer and Poggi 2020) are particularly easy to implement using the *randomForest* package (Liaw and Wiener 2002) in the R programming language (R Core Team 2022). The basic idea is that a random forest is a collection of many decision trees where prediction results are aggregrated over trees. The use of multiple trees improves prediction and makes inference less reliant on a single tree. The splits in the trees accommodate non-linear relationships and terminal nodes provide the estimated probabilities of discretized values of X.

A feature of random forest procedures is that the cutpoints for component variables in W (which determine nodes in trees) are obtained optimally by the algorithm. The random forests procedure provides us with propensity scores $P(X \mid W)$ for data (X, W) which is a necessary ingredient of the spatial causal analysis. In this analysis, $P(X \mid W)$ is the probability that the ball is received at spatial location $X = (r, \theta)$ given the match situation W.

In choosing the tuning parameters of the random forest procedure, we aimed for predictive accuracy. We used the grid search method to obtain the optimized hyperparameters of the random forest model. From that, we obtained ntrees = 300, mtry = 1 and nodesize = 2. All the other hyperparameters were set to their default values. For the evaluation of model performance, we used 10-fold cross validation.

In Figure 5.8, we present the feature importance plot of the variables $W = (t, d, f, o, b, r)$ used in the random forest procedure. The plot is provided as part of the *randomForest* package. As in Section 5.4.1 and Section 5.4.2, we observe that the variables t, d, f, o, b are important. In particular, t, f, o, b are roughly of the same importance with goal differential d slightly less important. The red card variable r does not appear important in the full spatial analysis.

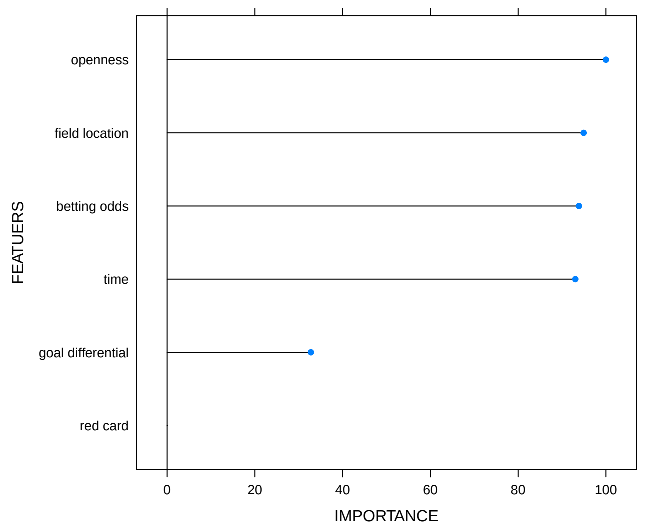


Figure 5.8: A plot of feature importance for the random forest procedure of Section 5.4.3.

Our causal investigation begins by discretizing the two-dimensional space X where throw-ins may be received. The region is truncated such that we only include observations extending 18 metres vertically from the throw-in location and 20 metres from the throw-in location horizontally (both left and right). Therefore, the area of the region is 720 squared metres (i.e. 18 metres by 40 metres). The region is then divided into rectangles of dimension 4 metres (horizontally) by 2 metres (vertically). This leads to 720/(2*4) = 90 rectangles. Based on $n = 7704$ throw-ins in the truncated region, we would expect $7704/90 \approx 85$ throw-ins on average, per rectangle. We amalgamate neighbouring rectangles when the number of observations in a rectangle is less than 30. There were 19 rectangles with fewer than 30 observations. After joining rectangles with insufficient observations, we were left with 71 rectangles. For every throw-in received, there is a propensity score $P(X \mid W)$ obtained using the machine learning methods based on random forests.

The matching idea used previously in the binary analyses of Sections 5.4.1 and 5.4.2 is now extended for the grid structure. We begin by randomly selecting a throw-in and noting its propensity score $p = P(X \mid W)$. Within each of the remaining 70 rectangles, we then select the throw-in whose propensity score is closest to p; these are the matching observations.

The process in the preceding paragraph is repeated $M = 30$ times. This means that there are 30 observations within a given rectangle, and each of these observations is matched to an observation in each of the remaining 70 rectangles. For each rectangle, we calculate the average number of shots $\bar{Y}$ generated by the throw-ins within the rectangle.

There is variability in the procedure due to the initial $M = 30$ observations that were randomly selected. Therefore, the entire procedure is repeated 100 times with $\bar{Y}$ for each rectangle averaged over the 100 iterations. We investigated the matching of propensity scores by calculating the maximum absolute difference of propensity scores across the $\binom{71}{2}$ pairs. This was then averaged over the $M = 30$ observations and the 100 iterations giving a value of 0.003. Therefore, the small difference suggests that the matching was successful.

To investigate the causal effect of X on Y, we produce a smoothed heat map of the average treatment effect $\bar{Y}$. In Figure 5.9, the heatmap is smoothed using the function *interp.loess* in the R package *tgp*. We observe darker regions (i.e. larger $\bar{Y}$) to the left (backward throw-ins) that are not too long (i.e. less than 5 metres outward). We also observe darker regions near the top (longish throw-ins from roughly 10 metres to 17 metres). This corroborates our findings from Section 5.4.1 and Section 5.4.2.

Our investigation of variability associated with the matching procedure involves calculating the standard deviation $s(\bar{Y})$ for each rectangle. For a particular rectangle, we obtain $\bar{Y}_i$ for the i-th iteration, $i = 1, \ldots, 100$. The quantity $s(\bar{Y})$ is the resultant sample standard deviation corresponding to the $\bar{Y}_i$ values. Then $s(\bar{Y})$ is averaged over the 71 rectangles where we obtain $\bar{s}(\bar{Y}) = 0.064$. From the color coding legend in Figure 9, it is apparent that the variability due to matching does not lead to maps with meaningful colour differences.

5.5 Discussion

The evaluation of tactics is a difficult and important problem for teams seeking to gain a competitive edge. This chapter uses causal methods facilitated by tracking data to investigate throw-ins in soccer. Our results suggest the surprising result that backward throw-ins are more effective than forward throw-ins. It is surprising since the receptor of a backward throw-in is in a less threatening offensive position on the field. We also demonstrate the

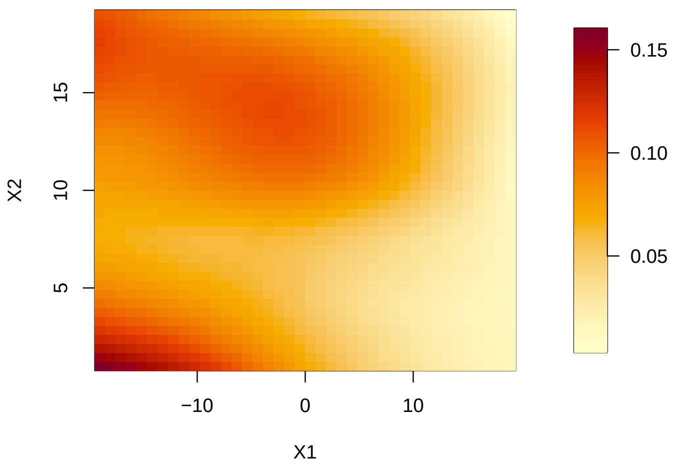


Figure 5.9: A smoothed heat map of the average treatment effect $\bar{Y}$ over the grid of the reception locations of throw-ins according to the algorithm of Section 5.4.3. The point $(X1, X2) = (0, 0)$ refers to the throw-in location.

benefit of the long throw-in, a tactic that appears to be increasing in usage but whose benefits have not been previously quantified.

We see this work as a template for the use of causal methods in sport to assess tactics. Of course, a limitation of causal methods is the latency of confounding variables W that effect the tactic X and the response Y. An underlying premise of our work is that sport specific knowledge and tracking data permit the identification of the important confounding variables.

There is an important and practical question related to our work. Given a particular game situation $W^* = (t, d, f, o, b, r)$, how should the throw-in be executed X to optimize Y? Let's assume that all confounding variables have been identified. Then we wish to compare a throw-in tactic X_0 against a throw-in tactic X_1 both occurring under W^*. There would be n_0 observations under X_0 and n_1 observations under X_1. Unfortunately, n_0 and n_1 would be small (likely 0 or 1), and therefore a meaningful comparison could not be carried out. Perhaps there is some way around this, maybe by categorizing W^*, X_0 and X_1 into larger classes of interest. With the provision of more data, this may be a future research direction.

Whereas tactics related to set plays are perhaps the easiest and first investigations that come to mind, we also wish to continue the analyses of tactics to more complex scenarios and across various sports.

Chapter 6

What does Rally Length tell us about Player Characteristics in Tennis?

6.1 Introduction

As is the case with many sports, tennis has seen an upsurge of work in analytics. One of the first serious statistical contributions to tennis analytics was the work by Klassen and Magnus (2001) which concerned an investigation of the iid assumption that points are independent and identically distributed. They concluded that there is a positive correlation between successive outcomes and that servers are less likely to win a point in important situations.

More recently, tennis analytics has been assisted by the availability of tracking data. With tracking data, player and ball locations are recorded with high frequency (i.e. spatio-temporal data), and these detailed datasets have contributed to explorations of many sporting problems that were previously unimaginable. Gudmundsson and Horton (2017) provide a review on spatio-temporal analyses used in invasion sports where player tracking data are available. In tennis, there are a growing number of papers that provide statistical analyses and rely on tracking data. Tea and Swartz (2022) investigate serving tendencies which may allow a player to anticipate the nature of the opponent's serve. The approach relies on hierarchical models in a Bayesian framework. Kovalchik and Albert (2022) also used a Bayesian framework where they investigate serve returns by introducing a semiparametric mixture model. Other papers that use tracking data but focus exclusively on the serve include Mecheri et al. (2016) and Wei et al. (2015). The text by Albert et al. (2017) provides a flavour for sports statistics across major sports.

Whereas tracking data analyses are on the increase, tracking data are often proprietary. In this chapter, we use publicly available non-tracking data to analyze an aspect of tennis that does not seem to have been previously investigated. Specifically, we consider rally length as the response variable with the added information whether the last volley was touched or not touched (i.e. a winner). This simple and readily available information allows us to analyze the rally characteristics of players. Various models of increasing complexity are introduced where we consider player characteristics, the identification of the server and a reduction in serve advantage as the rally proceeds.

In Section 6.2, we describe the data and associated issues with the dataset. In Section 6.3, three models of increasing complexity are proposed which take into account increasingly realistic features of the sport of tennis. In Section 6.4, priors are introduced for the models and computation is discussed. We fit the models using large datasets based on the ATP (Association of Tennis Professionals) tour for men and the WTA (Women' Tennis Association) tour for women. We consider separate analyses for first and second serves. Insights are obtained for the various models. For example, we consider overall tennis characteristics and how these vary across the the men's and the women's games, and in first versus second serves. We then investigate serve and rally characteristics with respect to various players of interest. We also observe how extending rallies is an important component of success in tennis. We conclude with a short discussion in Section 6.5.

6.2 Tennis Data

The data analysed in this chapter is based on 947,821 and 422,776 serves from men's and women's professional tennis matches, respectively. The data were obtained from the Match Charting Project (https://github.com/JeffSackmann/tennis_MatchChartingProject) maintained by Jeff Sackman. The data involve matches from 1970 through 2022 and contain shot-by-shot outcomes involving 710 distinct men and 489 distinct women. This public dataset provides information on shot type, shot direction, depth of returns, types of errors, and more. The data were collected by volunteers after watching the video recordings of matches. To the best of our knowledge, there is no other source of publicly-available data of this type. At present, these data appear to be under utilized for statistical modelling of tennis outcomes.

The data covers matches from all the major Grand Slam events, the Davis Cup and many minor tournaments. The best players of this time period are included in the dataset. Since the charting of matches was at the discretion of volunteers, data are highly skewed in favour of later years. Also, better (i.e. winning) players have a higher representation since they frequently reached the latter rounds of tournaments.

We have chosen to use limited and relatively simple data to facilitate modelling. As will be seen in Section 6.4, important tennis insights can be achieved with such data. The data collected for each point are of the form (T, I) where T is the number of touches leading to the point and I is an indicator function as to whether the server won the point. For example, suppose that the serve is in play and the receiver hit the serve out of bounds. In this case, there are two touches yielding $T = 2$ and $I = 1$. Alternatively, suppose that the serve is in play, the receiver returned the serve and the server then hit the ball into the net. In this case, there are three touches yielding $T = 3$ and $I = 0$. Note there is a technicality in the definition of (T, I). If $(T = 1, I = 0)$, then the event corresponds to a fault on a first serve. However, $(T = 1, I = 0)$ corresponds to a point for the receiver on a second serve (i.e., a winner).

It may seem that the number of touches T is a nonstandard choice for a data variable. We have selected T over the related variable "rally length" (or rally count) since there appears to be some confusion over the definition of rally length. For example, some people refer to an ace as a rally of length zero while others refer to an ace as a rally of length one. We also found that the variable T is more intuitive for modelling purposes.

6.2.1 Data Management Issues

When analyzing data, it is obviously important that the data are accurate. Therefore, we carried out various procedures to check data accuracy.

In the MatchChartingProject dataset, rows correspond to points awarded. Therefore, we augmented the dataset to include all serves. For example, whenever a second serve occurred, this implied that there was a first serve that resulted in a fault, for which no point was awarded.

Also, it is expected that in a huge dataset involving volunteer coding, some mistakes occur in data entry. For example, in the rallyCount variable, there were five non-numeric characters out of more than 500,000 serves in the ATP data. We simply removed the corresponding rows when this occurred.

In our model formulation, we need to determine the number of touches T. However, in the MatchChartingProject dataset, T is not directly provided and is obtained through the rallyCount variable RC. We set $T = RC$ if the IsRallyWinner variable is TRUE, and we set $T = RC + 1$ if the IsRallyWinner variable is FALSE. According to Jeff Sackman, RC is the number of shots excluding errors.

6.3 Models

In this section, we present three models. The first model is simple, concise, and assumes commond characteristics across all players. The model serves as a baseline case for the more realistic models to follow. The second model introduces the realism of individual player characteristics. The third model is a further extension, which distinguishes return characteristics according to whether a player is the server.

None of the models below distinguish first serves from second serves. However, there is a quick fix to this limitation; we simply use the same model analyzed separately for the two situations with model parameters interpreted according to the serve number. Naturally, there are fewer second serves. We analyze first and second serves separately in Section 6.4 for both the men's and women's games.

6.3.1 Model 1

Although this model is not meant to be realistic, it introduces notation and provides a baseline case for comparison purposes. This model also permits straightforward generalizations for more complex models. Later, we use results from Model 1 to elicit prior information for more realistic Bayesian models.

We introduce a parameter vector (a, f, s, w, r, m) which describes player characteristics. There is an implicit assumption that all players have the same characteristics. Specifically, we define

$a =$	Prob(player serves an *ace*)
$f =$	Prob(player serves a *fault*)
$s =$	Prob(player serves a ball that is neither an ace nor a fault)
$w =$	Prob(player returns a *winner*)
$r =$	Prob(player *returns* a ball in play which is subsequently touched by the opponent)
$m =$	Prob(player makes a *mistake* by touching but not returning a ball in play)

The parameterization includes service parameters a and f, critical components of tennis. The return parameters (w, r, m) are convenient since they describe all possibilities that can occur on a return that is touched, and consequently, $w + r + m = 1$. Quality of return capability is characterized by large w and small m.

We note that the proposed parameters have a different focus than standard statistics reported in the tennis literature. For example, with respect to the parameter a (aces), a commonly reported statistic is aces per match. A problem with aces per match is that players may not have the same number of average serves per match. A player with more serves (i.e. longer matches) will have an inflated aces/match statistic. Also, career aces favour players with longevity. Our proposed parameters and their estimates standardize performance with respect to the number of opportunities.

Consider then a first serve that results in a fault. In this case, neither player wins the point and we have

$$\text{Prob}(T = 1) = f \ . \tag{6.1}$$

Alternatively, consider a one touch event (i.e., $T = 1$) that may be either a first or second serve, but is not a first serve fault. In this case, we have

$$\text{Prob}(T = 1, I) = a^I \ f^{1-I} \ . \tag{6.2}$$

For two touches, a conditional probability expansion involving the two events yields

$$\text{Prob}(T = 2, I) = s \ m^I \ w^{1-I}$$

and, in general, for rallies with $t = 2, 3, \ldots,$ touches,

$$\text{Prob}(T = t, I) = \begin{cases} s \ r^{t/2-1} \ r^{t/2-1} \ m^I \ w^{1-I} & t \text{ even} \\ s \ r^{t/2-1/2} \ r^{t/2-3/2} \ w^I \ m^{1-I} & t \text{ odd} \end{cases} \tag{6.3}$$

Therefore, the likelihood based on the observed data is formed by taking the product over all serves using the expressions (6.1), (6.2) and (6.3). Model 1 has a parameterization that is 4-dimensional.

6.3.2 Model 2

We extend Model 1 by introducing a parameter vector $(a_i, f_i, s_i, w_i, r_i, m_i)$ for each player $i = 1, \ldots, N$. The vector describes playing characteristics of player i. The generalization is needed since some players, for example, are better servers than other players. For each $i = 1, \ldots, N$, the parameters satisfy the constraints in Model 1.

Without loss of generality, we assume that player i serves to player j. Consider then a first serve that results in a fault. In this case, neither player wins the point and we have

$$\text{Prob}(T = 1) = f_i \ . \tag{6.4}$$

Alternatively, consider a one touch event (i.e., $T = 1$) that may be either a first or second serve, but is not a first serve fault. In this case, we have

$$\text{Prob}(T = 1, I) = a_i^I \ f_i^{1-I} \ . \tag{6.5}$$

Extending (6.3) for specific players, for rallies with $t = 2, 3, \ldots,$ touches, we have

$$\text{Prob}(T = t, I) = \begin{cases} s_i \ r_j^{t/2-1} \ r_i^{t/2-1} \ m_j^I \ w_j^{1-I} & t \text{ even} \\ s_i \ r_j^{t/2-1/2} \ r_i^{t/2-3/2} \ w_i^I \ m_i^{1-I} & t \text{ odd} \end{cases} \tag{6.6}$$

Therefore, the likelihood based on the observed data is formed by taking the product over all tennis points using the expressions (6.4), (6.5) and (6.6), and by introducing appropriate subscripts i and j for specific players. With n players, Model 2 has a parameterization that is $4N$-dimensional.

6.3.3 Model 3

Although convenient, the Model 2 does not realistically account for differences in return characteristics according to the number of touches. For example, suppose again that player i serves to player j. It is well known in tennis that the probability of j hitting a winner on the immediate serve return (touch number two) is less than the probability of j hitting a winner on touch number four. The reason is that serves often place the returner in vulnerable situations where it is difficult for the returner to hit a quality shot. By the time player j reaches touches $4, 6, 8, \ldots$, the impact of the serve begins to dissipate.

Accordingly, we augment the return parameters (w_i, r_i, m_i) to $(w_i^{(t)}, r_i^{(t)}, m_i^{(t)})$ where the superscript $t = 2, 3, \ldots$, corresponds to the touch number. For example, we would expect that $w_i^{(2)} < w_i^{(4)} < w_i^{(6)}$ and that $w_i^{(3)} > w_i^{(5)} > w_i^{(7)}$.

With this modelling enhancement, the probabilities (6.4) and (6.5) remain the same. However, the probabilities in (6.6), for touches $t = 2, 3, \ldots$, become

$$\text{Prob}(T = t, I) = \begin{cases} s_i \left(\prod_{k=1}^{t/2-1} r_j^{(2k)} \; r_i^{(2k+1)}\right) \; (m_j^{(t)})^I \; (w_j^{(t)})^{1-I} & t \text{ even} \\ s_i \left(\prod_{k=1}^{t/2-1/2} r_j^{(2k)}\right) \left(\prod_{k=1}^{t/2-3/2} r_i^{(2k+1)}\right) \; (w_i^{(t)})^I \; (m_i^{(t)})^{1-I} & t \text{ odd} \end{cases} \tag{6.7}$$

A challenge involving (6.7) is the dimensionality of the parametrization. With N players and with maximum number of touches $t_{\max}$, the extended parametrization in Model 3 is $(2 + 2(t_{\max} - 1))N$-dimensional. A possible solution to this high-dimensional challenge involves limiting the parametrization. We may determine a cutoff value c for which the serve impact on returns dissipates; i.e. $w_i^{(t)} = w_i^{(c)}$, $r_i^{(t)} = r_i^{(c)}$, and $m_i^{(t)} = m_i^{(c)}$, for all $t \geq c$. Another idea is the introduction of exponential decay models for the specification of $w_i^{(t)}$, $r_i^{(t)}$ and $m_i^{(t)}$.

6.4 Analyses and Results

The analyses and results are provided according to the three models of increasing complexity. All of our models are developed in the Bayesian framework, and consequently require the specification of prior distributions. We discuss prior selection and associated computational issues. We also provide a comparison of fit among the three models.

6.4.1 Analysis of Model 1

We begin with the simplest model which provides overall insights but later serves in prior development for more complex models.

For the Bayesian approach, we consider flat priors given by

$$(a, f, s) \sim \text{Dirichlet}(1, 1, 1) \tag{6.8}$$

which is assumed independent of

$$(w, r, m) \sim \text{Dirichlet}(1, 1, 1) \ . \tag{6.9}$$

Note that the Dirichlet distributions in (6.8) and (6.9) are 2-dimensional where $a+f+s=1$ and $w+r+m=1$. The parameter s is a characteristic of lesser interest and is not specifically investigated. To obtain posterior inferences, we use the programming language *Stan* (Stan Development Team 2023). A main benefit of *Stan* is that a user supplies only the statistical model, the prior specification and the data. The associated Markov chain Monte Carlo (MCMC) aspects of the Bayesian implementation are carried out in the background.

To assess the robustness of Bayesian inferences with respect to prior selection, we calculate classical estimates of the parameters that are based on proportions. Recall that T is the number of touches and $I = 1/0$ corresponds to the server/receiver winning the point. Let n be the corresponding number of serves and let t_k be the number of touches on the kth serve. Then the sample proportions are given as follows:

$$\begin{aligned}
\hat{a} &= \frac{\#\{T=1, I=1\}}{n} \\
\hat{s} &= \frac{\#\{T>1\}}{n} \\
\hat{w} &= \frac{\#\{T=2, I=0\} + \#\{T=3, I=1\} + \#\{T=4, I=0\} + \cdots}{t_1 - 1 + t_2 - 1 + \cdots + t_n - 1} \\
\hat{m} &= \frac{\#\{T=2, I=1\} + \#\{T=3, I=0\} + \#\{T=4, I=1\} + \cdots}{t_1 - 1 + t_2 - 1 + \cdots + t_n - 1}
\end{aligned}$$

In Table 6.1, we provide estimated posterior means and proportions for Model 1. This is done for first and second serves using both the ATP and WTA data. We observe that there is agreement between the posterior estimates and the proportions which indicates that the information contained in the data dominates the prior. This is a consequence of having a rich dataset with many observations.

In Table 6.1, we observe some interesting features regarding first serves. It seems that aces occur in the men's game (0.08) at roughly double the rate observed in the women's game (0.04). This may be explained by the higher average speed of men's serves. The remaining displayed parameter estimates are comparable between men and women. We observe that first serve faults occur roughly forty percent of the time ($f = 0.40$ for men and $f = 0.37$ for women). This high rate may be explained by the desire to hit a first serve that is difficult to return - difficult serves are typically directed near the boundaries and are close to faults. If a player is able to get their racket on the ball, the return rate is high ($r = 0.74$ for men and

$r = 0.76$ for women). And if a player is able to get their racket on the ball, the probability m of making a mistake is roughly 2.5 times the probability w of hitting a winner.

For second serves, the occurrences of aces and faults are much different from first serves. Aces are reduced in second serves because the server is less aggressive and wants to ensure that the serve is in play. Similarly, faults are greatly reduced with second serves. In searching the literature, we could not find any reports of second serve ace percentage other than that second serve aces are extremely rare. Here, we estimate second serve ace percentage where there are 1% second serve aces for men and 0.4% second serve aces for women.

When comparing men to women on second serves, the biggest parameter differences involve aces ($a = 0.01$ for men and $a = 0.004$ for women) and faults ($f = 0.09$ for men and $f = 0.13$ for women). The lower ace percentage for women compared to men may again be due to lower serve speed. However, the higher fault percentage for women compared to men is not so readily explained.

		Parameters				
Serve	n	a	f	w	r	m
ATP 1st	692385	0.08 (0.08)	0.40 (0.40)	0.07 (0.07)	0.74 (0.74)	0.19 (0.19)
WTA 1st	307750	0.04 (0.04)	0.37 (0.37)	0.07 (0.07)	0.76 (0.76)	0.17 (0.17)
ATP 2nd	255436	0.01 (0.01)	0.09 (0.09)	0.05 (0.05)	0.80 (0.80)	0.15 (0.15)
WTA 2nd	115026	0.004 (0.004)	0.13 (0.13)	0.06 (0.06)	0.77 (0.77)	0.17 (0.17)

Table 6.1: Estimated posterior means (sample proportions) for the parameters a, f, w, r, m corresponding to Model 1. The posteriors are based on flat priors. The sample size n for the number of serves is also reported.

6.4.2 Analysis of Model 2

Model 2 extends Model 1 by introducing player specific characteristics. That is, we extend the parameter vector (a, f, s, w, r, m) to $(a_i, f_i, s_i, w_i, r_i, m_i)$ for all players $i = 1, \ldots, N$. The modelling philosophy is that individual player characteristics arise from a population of player characteristics given by

$$(a_i, f_i, s_i) \sim \text{Dirichlet}(ka, kf, ks) \tag{6.10}$$

which is assumed independent of

$$(w_i, r_i, m_i) \sim \text{Dirichlet}(kw, kr, km) \tag{6.11}$$

where (a, f, s, w, r, m) are the posterior means obtained through the previous analysis of Model 1, and $k > 0$ is a specified constant. Hence, the approach is empirical Bayes where larger values of k impose greater knowledge through the prior distributions (6.10) and (6.11). More specifically, larger values of k decrease the prior variance.

Our rationale for setting $k > 0$ is that we want players with longevity to have posterior parameter estimates that reflect their actual playing performance. On the other hand, for players with short professional careers who have served infrequently (say $n_i < 100$), we want their posterior estimates to revert closer to population averages. Accordingly, we have set $k = 200$ after some trial and error.

Under the higher parameterization associated with Model 2, computational demands increase. Running *Stan* with 2000 iterations using the men's first serve dataset requires 4.6 hours of computation on a laptop computer.

We are interested in the variability of the player characteristics for the four datasets (ATP 1st Serve, WTA 1st Serve, ATP 2nd Serve, WTA 2nd Serve). In Figures 6.1-6.3, we provide the boxplots of the posterior means of the parameters a_i, f_i and w_i for the four datasets. From Figure 6.1, we observe that there is a lower ace percentage on the second serve compared to the first serve. This is a consequence of more cautious (less aggressive) behaviour on the second serve since players are trying to avoid double faults. It is also interesting that the men (with higher service speeds) have a higher percentage of aces on the first serve than the women and that there is more variability in ace percentage amongst men than amongst women.

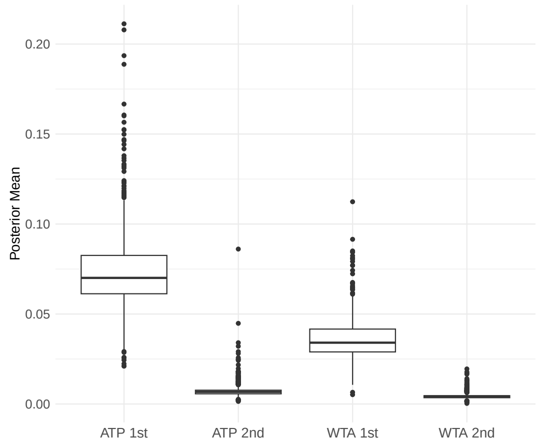


Figure 6.1: Boxplots of the posterior probability of an ace a_i by players for the four datasets.

From Figure 6.2, we observe a similar pattern amongst faults as with aces. That is, due to cautiousness on the second serve, the fault probability f_i is greatly decreased on the second serve compared to the first serve. This is true for both the men and the women. Variability amongst men and variability amongst women seems to be similar when comparing first serves and when comparing second serves. However, a conspicuous feature of Figure 6.2 is that the women fault on the second serve at a higher rate (roughly 50% more often) than the men. This may be a consequence of greater serving skill by the men.

From Figure 6.3, we observe the distributions of the percentage of winning shots w_i. With the women, these distributions do not differ greatly between first and second serves.

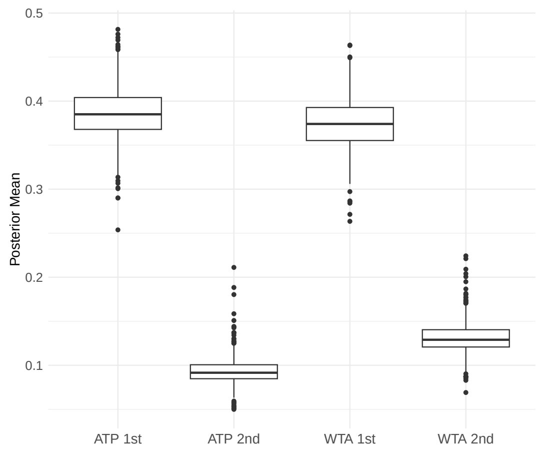


Figure 6.2: Boxplots of the posterior probability of a fault f_i by players for the four datasets.

However, with the men, the distributions differ where there is a higher percentage of winning volleys using the first serve data compared to the second serve data. However, these differences are difficult to interpret since a winner on the return of a serve is a markedly different situation than a winner on the server's first return. For example, we would expect a player to be more off balance in the former situation than in the latter situation due to the influence of a powerful serve. This aspect is investigated in Section 6.3 using Model 3 where return characteristics are allowed to vary according to touch number. Because of the importance of touch number, we do not investigate the distributions of return probabilities r_i and mistake probabilities m_i under Model 2.

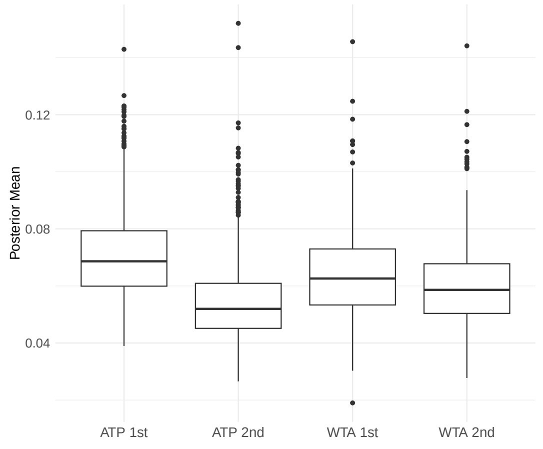


Figure 6.3: Boxplots of the posterior probability of a winning shot w_i by players for the four datasets.

With an understanding of the distribution of parameter estimates, we wish to observe which players distinguish themselves with respect to the player characteristics. In Tables 6.2-6.4, we highlight the top five players with respect to their posterior estimates a_i, f_i and

w_i under the four datasets. From Table 6.2, we observe that the top five list of ATP players who serve high ace percentages (first serve) are familiar names. The posterior standard errors are such that there exist differences between the players. Reilly Opelka is 6 feet 11 inches tall and this affords a serve trajectory that can maximize serving speed. He is a current player who is not highly ranked (#643 in 2023). It will be interesting to see if his overall game catches up to his devastating serve. John Isner is likewise tall (6 feet 10 inches) and is reknown for his serve. Goran Ivanisevic was a left hander which is unusual and may have accounted for the difficulty that players experienced when returning his serve. It is noteworthy that four of the top-five ATP players are still active in 2023 (Opelka, Isner, Kyrgios) or recently retired (Karlovic). Interestingly, three of the players from the first serve list reside on the second serve list. With second serves, Maxime Cressy is an unusual case with an ace percentage that is comparable to the mean first serve ace percentage. It seems that Cressy has discovered/perfected something, and that he differs from the population of ATP professionals. In the women's game, Serena Williams sits atop the first serve ace percentage list. Similar to the men, four of the top five players with respect to first serve ace percentage (excepting Lucie Hradecka) are recent players. With WTP second serve ace percentage, there is not much to discuss since all players (even the top ones) have very few aces.

ATP 1st Serve		WTA 1st Serve		ATP 2nd Serve		WTA 2nd Serve	
1. R.Opelka	0.21(0.006)	1. S.Williams	0.11(0.004)	1. M.Cressy	0.08(0.008)	1. C.Harrison	0.02(0.008)
2. I.Karlovic	0.20(0.007)	2. M.Keys	0.09(0.005)	2. I.Karlovic	0.04(0.006)	2. J.Ostapenko	0.02(0.003)
3. J.Isner	0.19(0.004)	3. L.Hradecka	0.09(0.020)	3. A.Bublik	0.03(0.005)	3. C.Paquet	0.02(0.007)
4. G.Ivanisevic	0.18(0.006)	4. K.Pliskova	0.08(0.004)	4. R.Opelka	0.03(0.005)	4. S.Lisicki	0.02(0.004)
5. N.Kyrgios	0.17(0.004)	5. C.Vandeweghe	0.08(0.009)	5. N.Kyrgios	0.03(0.003)	5. J.Niemeier	0.01(0.005)

Table 6.2: Estimated posterior means (Model 2) for the top five players in the four datasets for the probability of serving an ace a_i. Estimated posterior standard errors are given in parantheses.

From Table 6.3, we first remind ourselves that low fault percentage is considered good. Generally speaking, the players in these lists are not specifically known for their low fault percentage. It is remarkable that John Isner appears on this list (ATP first serves) and also in Table 6.2 for first serve ace percentage. His appearance on both lists confirms that he is an outstanding server. During his playing days, Mats Wilander must have had outstanding serve control; he has low fault percentage on both first and second serves. For the women, we observe that Chris Evert (an iconic player) faulted infrequently on first serve. Given the nickname "Ice Maiden", perhaps this reflected her ability not to succumb to the pressure of faulting.

From Table 6.4, we investigate the ability to hit winners during the rally. It is noteworthy here that most of the players who appear on the top lists are from an earlier era. Maybe this is a consequence of the game being faster today (especially with respect to the serve) and the consequent difficulty of hitting winners from faster shots. It is also noteworthy that

ATP 1st Serve		WTA 1st Serve		ATP 2nd Serve		WTA 2nd Serve	
1. A.Berasategui	0.25(0.018)	1. S.Errani	0.26(0.010)	1. M.Wilander	0.04(0.007)	1. A.Radwanska	0.07(0.006)
2. M.Wilander	0.28(0.007)	2. C.Evert	0.27(0.010)	2. G.Forget	0.05(0.009)	2. M.Keys	0.08(0.007)
3. S.Baez	0.29(0.012)	3. N.Parrizas Diaz	0.28(0.020)	3. J.Brooksby	0.05(0.008)	3. K.Juvan	0.08(0.010)
4. J.Isner	0.30(0.005)	4. A.Rus	0.29(0.006)	4. A.Chesnokov	0.05(0.009)	4. A.Sanchez Vicario	0.09(0.009)
5. M.Cecchinato	0.30(0.011)	5. M.Niculescu	0.29(0.010)	5. M.Safin	0.05(0.005)	5. K.Muchova	0.09(0.008)

Table 6.3: Estimated posterior means (Model 2) for the top five players in the four datasets for the probability of committing a fault f_i. Estimated posterior standard errors are given in parantheses.

Novak Djokovic widely regarded as the GOAT (greatest of all time) in men's tennis does not appear on this list nor in the previous tables. We posit that Djokovic has an all-around game that leads to his excellence. Prominent names that appear in Table 6.4 include Pat Rafter, Stefan Edberg, Rod Laver, Hana Mandilikova and Anna Kournikova. Since hitting winners off of rallies is not a statistic that is routinely collected and analyzed, it good to see some excellent all-time players appearing in these lists. However, as mentioned with Djokovic, these individual parameters, studied on their own do not portend success. As we observe later in this section, it is the combination of skills which lead to success.

ATP 1st Serve		WTA 1st Serve		ATP 2nd Serve		WTA 2nd Serve	
1. K.Carlsen	0.14(0.016)	1. K.Scott	0.15(0.020)	1. R.Laver	0.15(0.010)	1. K.Scott	0.14(0.020)
2. K.Curren	0.13(0.010)	2. J.Goerges	0.12(0.005)	2. K.Carlsen	0.14(0.020)	2. J.Ostapenko	0.12(0.006)
3. P.Rafter	0.12(0.004)	3. H.Sukova	0.12(0.009)	3. R.Krajicek	0.13(0.007)	3. J.Goerges	0.12(0.006)
4. S.Edberg	0.12(0.002)	4. S.Waltert	0.11(0.010)	4. D.Brown	0.12(0.009)	4. H.Mandlikova	0.11(0.010)
5. R.Laver	0.12(0.001)	5. J.Ostapenko	0.11(0.005)	5. J.Siemerink	0.11(0.020)	5. A.Kournikova	0.10(0.010)

Table 6.4: Estimated posterior means (Model 2) for the top five players in the four datasets for the probability of hitting a winner w_i. Estimated posterior standard errors are given in parantheses.

We have suggested that the game of tennis may have changed over time. With the various parameters introduced in Model 2 which characterize aspects of the game, we see that longitudinal tennis analyses are readily possible. For example, consider the ATP mistake volley parameter m_i on first serves. In Figure 6.4, we plot the posterior mean of m_i versus the year of entry into professional tennis for the corresponding player. There are several things to observe from Figure 6.4. First, there are more players in our dataset in recent years. This is a consequence of greater availability of recent recordings for charting. There also seems to be tighter variability in the mistake parameter m_i in recent years. There is also an indication that volley mistakes have slightly decreased over time. With tighter scrutiny on tennis analytics and performance, this is perhaps understandable.

We now investigate how the parameters a, f, w and m correlate individually to success. We collect the Official Pepperstone ATP player ranking points obtained from the ATP website at https://www.atptour.com/en/rankings/singles. There are 100 players on this list and we take the points corresponding to June 6/2022. These points are used to determine player

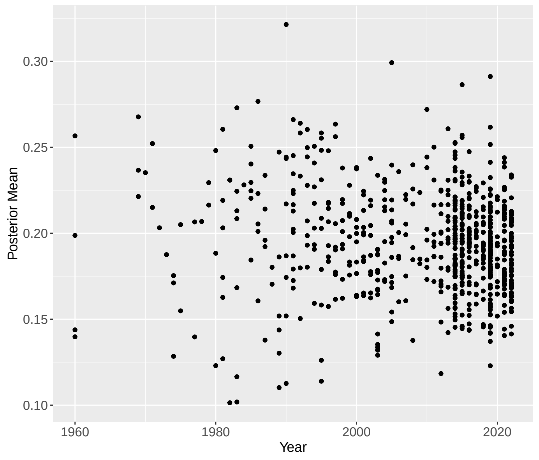


Figure 6.4: Scatterplot of the posterior mean of m_i for first serve ATP data versus the year that player began professional tennis.

rankings. For example, the 1st, 50th and 100th players on the list are Novak Djokovic, Roger Federer and Alexei Popyrin with 8770, 1030 and 626 points, respectively. We obtain sample correlation coefficients between the player ranking points and their posterior estimates. The results are reported in Table 6.5. The message again is that individual parameters do not correlate strongly with success. We do note that although the correlations are small (i.e. close to zero), they tend to correlate in the correct directions. For example, high probabilities of aces a_i are beneficial, and here, the associated correlations are positive with respect to the first serve. We would expect to also see a positive correlation with winners w_i. However, this is not the case since winner occur infrequently (see Figure 6.3) and there is not much variability in winners across players. Also, as mentioned previously, the analysis of winners needs to be considered in tandem with touch number T. For example, winners on $T = 2$ are unlikely since this corresponds to returning a serve where the receiver is often off-balance. On the other hand, a winner on $T = 3$ is likely since the server retains some advantage from the serve.

Low probabilities of faults f_i and mistakes m_i are beneficial, and here, the associated correlations are negative with respect to the first serve. The correlations involving second serves are less interpretable where fewer aces and faults occur. Again, a limiting factor in the correlation study involving w_i and m_i is that touch number is not considered; this is remedied with Model 3 in Section 6.3.

Now, although the previous analyses are informative, it is obviously the case that player excellence is a function of various skills. To investigate the combination of skills, we carry out a regression analysis of the previously mentioned Pepperstone points against the variables $(a_{i1}, f_{i1}, w_{i1}, m_{i1}, a_{i2}, f_{i2}, w_{i2}, m_{i2})$ where the second subscript in the pair of subscripts refers

Parameter	ATP 1st Serve	ATP 2nd Serve
a	0.11	-0.05
f	-0.25	0.01
w	-0.05	-0.14
m	-0.14	-0.22

Table 6.5: Correlation coefficients between 2022 ATP player ranking points and the posterior estimates of a, f, w, m for first and second serves.

to serve number. Retaining only the significant variables, we obtain the fitted equation

$$y = 7539 + 19872 * a_1 - 14217 * f_1 - 25706 * m_2 \tag{6.12}$$

where the correlation between the fitted line and the Pepperstone points is an improved $r = 0.50$.

Equation (6.12) can assist in player evaluation. For example, suppose that a player has first serve ace percentage $a_1 = 0.08$. The player's points might be expect to rise by 199 points if they could increase their first serve ace percentage to $a_1 = 0.09$. However, it must be kept in mind that there are reasonable restrictions (see Figure 6.1) as to the extent that one might increase their first serve ace percentage. It is interesting that the fitted equation (6.12) contains the characteristics a_1, f_1 and m_2. Although the inclusion of a_1 and f_1 appear obvious, it is worth considering why mistakes occurring on the second serve m_2 are important whereas mistakes on the first serve m_1 are not important. Our data reveal that 72% of first serves involve $T = 3$ touches or less. When there are only $T = 3$ touches, this means that mistakes of the type m_1 can only be made on the second or third touches. After a powerful serve, the second touch (i.e. the receiver's return) will be difficult, and therefore, mistakes m_1 will be made at a high rate. Conversely, on the third touch, the server will continue to benefit from the serve as mistakes m_1 will occur at a low rate. These two phenomena will tend to cancel each other out, and this is why m_1 is not statistically significant. Again, the importance of touch number is considered in Section 6.3.

6.4.3 Analysis of Model 3

The enhancement of Model 2 to Model 3 involves the introduction of return characteristics w, r, m that vary according to touch number. With the player characteristics i, and the touch number t, this leads to parameters $w_i^{(t)}, r_i^{(t)}, m_i^{(t)}$. The motivation is that the effect of the serve dissipates as the rally progresses. We wish to quantify the effect.

We use the same prior distribution as given in (6.10) and (6.11) for each touch $t = 2, 3, \ldots$. We then average posterior estimates of $w_i^{(t)}, r_i^{(t)}, m_i^{(t)}$ over all players i to give varying return characteristics $w^{(t)}, r^{(t)}, m^{(t)}$. For the ATP first serve dataset, we plot the estimates of the winner probability $w^{(t)}$ with respect to the touch number t in Figure 6.5. We observe that the server retains an advantage in hitting a winner on touches $T = 3$ and $T = 5$. The advantage then quickly dissipates on subsequent touches for the server where the posterior probability of a winner approaches the prior mean probability. This is expected since there are fewer cases of long rallies. For example, the number of cases where $T = 15$ in the ATP first serve dataset is only $n = 2,245$. We also observe that as the rally continues, the probability of the server and the receiver hitting a winner is the same.

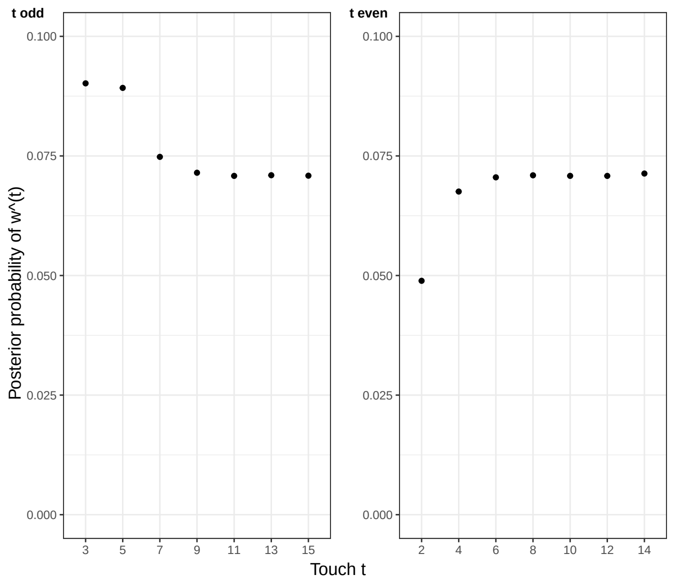


Figure 6.5: Plot of the posterior estimate $w^{(t)}$ versus t according to Model 3 with respect to the ATP first serve data. The first plot concerns t odd (server) and the second plot concerns t even (receiver).

In Figure 6.6, we plot the estimates of the mistake probability $m^{(t)}$ with respect to the touch number t. Here, we observe that the server's mistake probabilities increase only slightly with touch number. This seems intuitive as mistakes are exaggerated when a player is put in difficult situations, and there are no touch numbers where the server is consistently in distress. We also observe that mistake probabilities decrease for the receiver as the rally continues with stabilization around $T = 6$. This is the point at which the residual influence of the serve has dissipated.

Under the higher parameterization associated with Model 3, computational demands increase. Running *Stan* with 2000 iterations using the men's first serve dataset requires roughly 18 hours of computation on a laptop computer.

6.5 Discussion

Using a massive dataset of coarse observations in tennis, we have produced models that describe various features of the sport. These features are explored with respect to both the men's and women's games, and with respect to both the first and second serves.

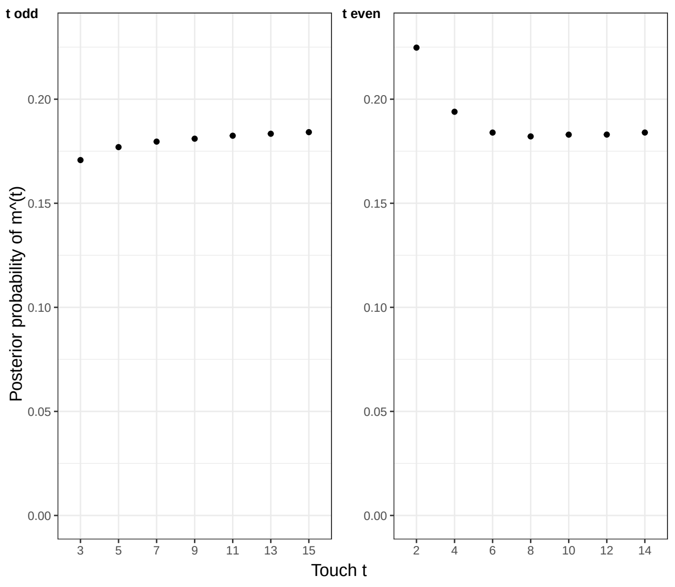


Figure 6.6: Plot of the posterior estimate $m^{(t)}$ versus t according to Model 3 with respect to the ATP first serve data. The first plot concerns t odd (server) and the second plot concerns t even (receiver).

An instructive aspect of this research is that players have their own playing characteristics a and f corresponding to the serve, and their characteristics $w^{(t)}$ and $m^{(t)}$ corresponding to touch $T = t$. These probabilities summarize all components of a player's game, and can be viewed with the purpose of evaluation with respect to average player performance.

Also, given these estimated parameters, a researcher can use simulation techniques to investigate all sorts of interesting questions. For example, one may simulate matches between two competitors to obtain the probability of a match lasting beyond two sets. As another example, one may be interested in the probability that a player breaks service in a particular match. With parameters describing inter-related aspects of tennis, the models provide great scope for analysis both at the game level and the player level.

In terms of future directions, it would be interesting to see how characteristics vary across different surfaces (e.g., grass, clay and hardcourt). With fewer matches (and data) on grass, one may extend the proposed hierarchical models where grass parameters are related to hardcourt parameters, for example. More work could also be done on identifying aspects of the game which make players truly exceptional. In this chapter, the framework has been proposed to address extended questions in tennis that have not been previously investigated.

A further avenue for model enhancement concerns the parameters a and w corresponding to the probability of an ace and the probability of a winner, respectively. In Model 2 and Model 3, these parameters have a subscript i corresponding to the player of interest who is involved in the shot. However, it is apparent that these probabilities are also impacted by i's opponent. It would be useful to incorporate this feature in enhanced future models.

Bibliography

Adrion, J. R., Cole, C. B., Dukler, N., Galloway, J. G., Gladstein, A. L., Gower, G., Kyriazis, C. C., Ragsdale, A. P., Tsambos, G., Baumdicker, F., Carlson, J., Cartwright, R. A., Durvasula, A., Gronau, I., Kim, B. Y., McKenzie, P., Messer, P. W., Noskova, E., Ortega-Del Vecchyo, D., Racimo, F., Struck, T. J., Gravel, S., Gutenkunst, R. N., Lohmueller, K. E., Ralph, P. L., Schrider, D. R., Siepel, A., Kelleher, J., and Kern, A. D. (2020). A community-maintained standard library of population genetic models. *eLife*, 9:e54967.

Albert, J.A., Glickman, M.E., Swartz, T.B. and Koning, R.H., Editors (2017). *Handbook of Statistical Methods and Analyses in Sports*, Chapman & Hall/CRC Handbooks of Modern Statistical Methods, Boca Raton.

Anzer, G. and Bauer, P. (2021). A goal scoring probability model for shots based on synchronized positional and event data in football (soccer). *Frontiers in Sport and Active Living*, 3, Article 624475.

Austin, P.C. (2011). An introduction to propensity score methods for reducing the effects of confounding in observational studies. *Multivariate Behavioral Research*, 46, 399-424.

Bailey, S.R., Loeppky, J. and Swartz, T.B. (2020). The prediction of batting averages in Major League Baseball. *Stats*, 9, 84-93.

Baker, R.D. and McHale, I. (2014). A dynamic paired comparisons model: Who is the greatest tennis player? *European Journal of Operational Research*, 236 (2), 677-684.

Blank, D. (2012). *Soccer IQ*, www.soccerpoet.com

Brier, G.W. (1950). "Verification of forecasts expressed in terms of probability", *Monthly Weather Review*, 78, 1-3.

Browning, S. R., Browning, B. L., Daviglus, M. L., Durazo-Arvizu, R. A., Schneiderman, N., Kaplan, R. C., and Laurie, C. C. (2018). Ancestry-specific recent effective population size in the Americas. *PLoS Genetics.*

Connoly, R.A. and Rendleman, Jr, R.J. (2008). Skill, luck and streaky play on the PGA tour. *Journal of the American Statistical Association*, 103(481), 74-78.

Cruceanu, C., Ambalavanan, A., Spiegelman, D., Gauthier, J., Lafrenière, R. G., Dion, P. a., Alda, M., Turecki, G., and Rouleau, G. a. (2013). Variants for Lithium-Responsive Bipolar Disorder 1. *Genome*, 640(September):634–640.

De Jong, L.M.S., Gastin, P.B., Angelova, M., Bruce, L. and Dwyer, B. (2020). Technical determinants of success in professional women's soccer: A wider range of variables reveals new insights. *PLOS ONE*, 15(10), e0240992.

Decroos, T., Bransen, L., Van Haaren, J. and Davis, J. (2018). "Actions speak louder than goals: Valuing player actions in soccer", *Proceedings of the 25th ACM Conference on Knowledge Discovery and Data Mining (KDD 2019)*, retrieved Sep 16/20 at https://people.cs.kuleuven.be/∼tom.decroos/reports/kdd19_tomd.pdf

Duckworth, F.C. and Lewis, A.J. (1998). A fair method of resetting targets in one-day cricket matches. *Journal of the Operational Research Society*, 49, 220-227.

Epasinghege Dona, N. and Swartz, T.B. (2023). A causal investigation of pace of play in soccer. *Statistica Applicata - Italian Journal of Applied Statistics*, 35(1), Article 6.

Fernández, J., Bornn, L. and Cervonne, D. (2019). "Decomposing the immeasurable sport: a deep learning expected possession value framework for soccer", *MIT Sloan Analytics Conference*, retrieved Sep 16/20 at www.sloansportsconference.com/wp-content/uploads/2019/02/Decomposing-the-Immeasurable-Sport.pdf

Gelman, A. and Meng, X-L., Editors (2004). *Applied Bayesian Modeling and Causal Inference from Incomplete-Data Perspectives*, Wiley: New York.

Genuer, R. and Poggi, J-M. (2020). *Random Forests with R*, Springer: New York.

Goes, F.R., Brink, M.S., Elferink-Gemser, M.T., Kempe, M. and Lemmink, K.A.P.M. (2021). The tactics of successful attacks in professional association football: Large-scale spatiotemporal analysis of dynamic subgroups using position tracking data. *Journal of Sports Sciences*, 39(5), 523-532.

Guan, T., Cao, J. and Swartz, T.B. (2023). Parking the bus. *Journal of Quantitative Analysis in Sports*, 19(2), https://doi.org/10.1515/jqas2021-0059

Gudmundsson, J. and Horton, M. (2017). Spatio-temporal analysis of team sports. *ACM Computing Surveys*, 50(2), Article 22.

Haller, B. C. and Messer, P. W. (2019). SLiM 3: Forward Genetic Simulations Beyond the Wright–Fisher Model. *Molecular Biology and Evolution*, 36(3):632–637.

Imbens, G.W. (2004). Nonparametric estimation of average treatment effects under exogeneity: A review. *The Review of Economics and Statistics*, 86, 4-29.

Jones, S. J., Voong, J., Thomas, R., English, A., Schuetz, J., Slack, G. W., Graham, J., Connors, J. M., and Brooks-Wilson, A. (2017). Nonrandom occurrence of lymphoid cancer types in 140 families. *Leukemia & Lymphoma*, 58(9):2134–2143. PMID: 28278712.

Kempe, M., Vogelbein, M., Memmert, D. and Nopp, S. (2014). Possession vs. direct play: Evaluating tactical behavior in elite soccer. *International Journal of Sports Science* 4(6A), 35-41.

Kim, S. (2004). Voronoi analysis of a soccer game. *Nonlinear Analysis: Modelling and Control*, 9(3), 233-240.

Klassen, F.J.G.M. and Magnus, J.R. (2001). Are points in tennis independent and identically distributed? Evidence from a dynamic binary panel model. *Journal of the American Statistical Association*, 96(454), 500-509.

Kovalchik, S.A. and Albert, J. (2022). A statistical model of serve return impact patterns in professional tennis. Accessed October 13, 2022 at https://arxiv.org/abs/2202.00583

Lepschy, H., Wäsche, H. and Woll, A. (2021). Success factors in football: an analysis of the German Bundesliga. *International Journal of Performance Analysis in Sport*, 20(2), 150-164.

Liaw, A. and Wiener, M. (2002). Classification and regression by randomForest. *R News, The Newsletter of the R Project*, Vol 2/3, 18-22.

Macdonald, B. (2012). "An expected goals model for evaluating NHL teams and players", *MIT Sloan Analytics Conference*, retrieved Sep 16/20 at www.hockeyanalytics.com/Research _files/NHL-Expected-Goals-Brian-Macdonald.pdf

McLellan, I. (2010). Total football: Whatever happened to the beautiful game? *Bleacher Report: World Football*, Accessed June 2, 2022 at https://bleacherreport.com/articles/321814-beautiful-game-what-ever-happened-to-total-football

Mecheri, S., Rioult, F., Mantel, B., Kauffmann, F. and Benguigui, N. (2016). The serve impact in tennis: First large-scale study of big hawk-eye data. *Statistical Analysis and Data Mining*, 9(5), 310-325.

Merlin, M., Cunha, S.A., Moura, F.A., Torres, R., Gonçalves, B. and Sampaio, J. (2020). Exploring the determinants of success in different clusters of ball possession sequences in soccer. *Research in Sports Medicine*, 28(3), 339-350.

Nieuwoudt, C., Brooks-Wilson, A., and Graham, J. (2020). SimRVSequences: An R package to simulate genetic sequence data for pedigrees.*Bioinformatics*, 36(7):2295–2297.

Nieuwoudt, C., Jones, S. J., Brooks-Wilson, A., and Graham, J. (2018). Simulating pedigrees ascertained for multiple disease-affected relatives. *Source Code for Biology and Medicine*, 13:2.

Pearl, J. (2009). *Causality: Models, Reasoning and Inference, Second Edition.* Cambridge University Press: Cambridge.

Pollard, R., Ensum, J. and Taylor, S. (2004). "Estimating the probability of a shot resulting in a goal: The effects of distance, angle and space", *International Journal of Soccer and Science*, 2, 50-55.

Purcell, S., Neale, B., Todd-Brown, K., Thomas, L., Ferreira, M. A., Bender, D., Maller, J., Sklar, P., De Bakker, P. I., Daly, M. J., and Sham, P. C. (2007). PLINK: A tool set for whole-genome association and population-based linkage analyses. *American Journal of Human Genetics*, 81(3):559–575.

R Core Team (2022). R: *A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Riggs, K., Chen, H. S., Rotunno, M., Li, B., Simonds, N. I., Mechanic, L. E., and Peng, B. (2021). On the application, reporting, and sharing of in silico simulations for genetic studies.*Genetic Epidemiology*, 45(2):131–141.

Rudd, S. (2011). "A framework for tactical analysis and individual offensive production assessment in soccer using Markov chains", In *New England Symposium on Statistics in Sports*, retrieved Sep 16/20 at http://nessis.org/nessis11/rudd.pdf

Sarkar, S. and Kamath, S. (2022). Does luck play a role in the determination of the rank positions in football leagues? A study of Europe's "big five". *Annals of Operations Research*, To appear.

Sekhon, J.S. (2011). Multivariate and propensity score matching software with automated balance optimization: The matching package for R. *Journal of Statistical Software*, 42, 1-52.

Shaw, L. and Glickman, M. (2019). Dynamic analysis of team strategy in professional football. *Barça Sports Analytics Summit*

Shen, E., Santo, S. and Akande, O. (2022). Analyzing pace-of-play in soccer using spatio-temporal event data. *Journal of Sports Analytics*, 8(2), 127-139.

Silva, R., Davis, J. and Swartz, T.B. (2018). The evaluation of pace of play in hockey. *Journal of Sports Analytics*, 4, 145-151.

Smith, H., Gnanadesikan, R. and Hughes, J.B. (1962). Multivariate analysis of variance (MANOVA). *Biometrics* 18(1), 22–41.

Spearman, W. (2018). Beyond expected goals. *Proceedings of the 2018 MIT Sloan Sports Analytics Conference*, Accessed October 6, 2022 at https://www.researchgate.net /publication /327139841_Beyond_Expected_Goals

Stan Development Team (2023). *Stan Modeling Language User's Guide and Reference Manual*, Version 2.32. https://mc-stan.org

Stone, J.A., Smith, A. and Barry, A. (2021). The undervalued set piece: Analysis of soccer throw-ins during the English Premier League 2018-2019 season.*International Journal of Sports Science and Coaching*, 16(3), 830-839.

Stuart, E.A. (2010). Matching methods for causal inference. A review and a look forward. *Statistical Science*, 25(1), 1-21.

Swartz, T.B. (2017). "Research directions in cricket", *Handbook of Statistical Methods and Analyses in Sports*, Editors Albert, J.A., Glickman, M.E., Swartz, T.B. and Koning, R.H., Chapman & Hall/CRC Handbooks of Modern Statistical Methods, Boca Raton, 445-460.

Tea, P. and Swartz, T.B. (2022). The analysis of serve decisions in tennis using Bayesian hierarchical models. *Annals of Operations Research*, 325(1), 633-648.

Tweedale, A. (2022). Counter-pressing and the gegenpress: football tactics explained. *The Coaches Voice: Coaching Knowledge*, Accessed June 2, 2022 at https://www. coachesvoice .com/cv/counter-pressing-gegenpressing-football-tactics-explained-klopp-guardiola-bielsa-hasenhuttl/

Wei, X., Lucey, P., Morgan, S., Carr, P., Reid, M. and Sridharan, S. (2015). Predicting serves in tennis using style priors. *Proceedings of the 21th ACM SIGKDD International Conference on Discovery and Data Mining*, 2207-2215.

Weissbock, J. (2014). Forecasting success in the National Hockey League using in-game statistics and textual data. *PhD dissertation, Université d'Ottawa.*

Wilson, J. (2013). *Inverting the Pyramid*, Nation Books, New York.

Wu, Y. and Swartz, T.B. (2022). A new metric for pitch control based on an intuitive motion model. *Manuscript under review.*

Wu, Y., Danielson, A., Hu, J. and Swartz, T.B. (2021). A contextual analysis of crossing the ball in soccer. *Journal of Quantitative Analysis in Sports*, 17(1), 57-66.

Yu, D., Boucher, C., Bornn, L. and Javan, M. (2019). Evaluating team-level pace of play in hockey using spatio-temporal possession data. *Proceedings of the 2019 MIT Sloan Sports Analytics Conference*, accessed October 18, 2021 at https://arxiv.org/pdf/1902.02020.pdf

Appendix A

Supplementary Material 1-A: Simulate SNV sequence data for pedigree founders

Supplementary Material 1-A: Simulate SNV sequence data for pedigree founders

Nirodha Epasinghege Dona, Jinko Graham

2022-04-16

Contents

This is the first in a series of RMarkdown documents describing how we simulated exome-sequencing data in pedigrees ascertained to have four or more relatives affected with lymphoid cancer. The overall workflow for this project is shown below.

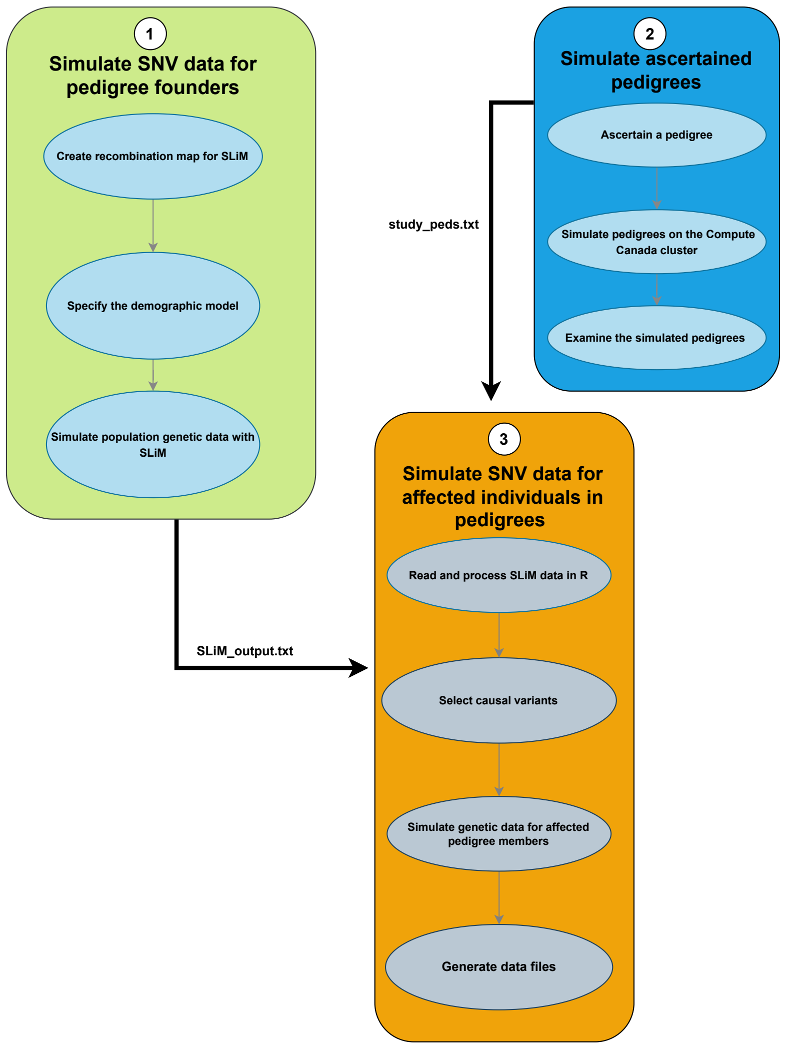

1
Simulate SNV data for pedigree founders
Create recombination map for SLiM
Specify the demographic model
Simulate population genetic data with SLiM
2
Simulate ascertained pedigrees
Ascertain a pedigree
Simulate pedigrees on the Compute Canada cluster
Examine the simulated pedigrees
study_peds.txt
SLiM_output.txt
3
Simulate SNV data for affected individuals in pedigrees
Read and process SLiM data in R
Select causal variants
Simulate genetic data for affected pedigree members
Generate data files

Figure 1: Work-flow for simulating the exome-sequencing data for ascertained pedigrees.

This document focuses on the part of the flowchart labelled as 1 (the green box). To start, we require single-nucleotide variant (SNV) sequences for pedigree founders. These founders are assumed to be sampled from an American Admixed population, which we simulate with the evolutionary simulation package `SLiM` (Haller et al. 2019). In particular, we simulate genome-wide sequences of exons only, to mimic exome sequencing.

The outline of this document as follows. Section 1 explains how we create the `SLiM` recombination map using the `create_SlimMap()` function in the `SimRVSequence` R package (Nieuwoudt, Brooks-Wilson, and Graham

2020). Section 2 explains the demographic model for the source population of the pedigree founders. Section 3 discusses how we set the parameters in our SLiM model to simulate the exon-only SNV sequences.

The final outcome of this RMarkdown document is the file `SLiM_output` containing SNV exon sequences from a simulated American Admixed population. In the third RMarkdown document of this series, the population sequences file is sampled to get the founder sequences to drop down the ascertained pedigrees. The final gene-dropping step generates the exome-sequencing data in the family-based study of lymphoid cancer families.

1 Create recombination map for SLiM

To simulate the genome-wide exon-only sequences with SLiM, we need to supply a recombination map which reads the exon positions in chromosomes. We use the `create_SlimMap()` function in the `SimRVSequence` (Nieuwoudt, Brooks-Wilson, and Graham 2020) R package, as shown in the next code chunk.

```r
library(SimRVSequences)

# Load hg_exons data set in SimRVSequence package
data("hg_exons")

# Create recombination map for exon-only data using the hg_exons dataset
s_map <- create_slimMap(exon_df = hg_exons)
head(s_map)
```

```
##   chrom segLength  recRate mutRate  exon simDist endPos
## 1     1     11873 0.00e+00   0e+00 FALSE       1      1
## 2     1       354 1.00e-08   1e-08  TRUE     354    355
## 3     1       385 3.85e-06   0e+00 FALSE       1    356
## 4     1       109 1.00e-08   1e-08  TRUE     109    465
## 5     1       499 4.99e-06   0e+00 FALSE       1    466
## 6     1      1609 1.00e-08   1e-08  TRUE    1609   2075
```

We use the `hg_exons` dataset in the `SimRVSequence` package to specify the exon positions of each of the 22 human autosomes, based on the hg38 reference genome from the UCSC Genome Browser (Nieuwoudt, Brooks-Wilson, and Graham 2020). As shown above, the call to `create_SlimMap()` returns a data frame with information about the genetic segments in each chromosome. As an example, the first row in the output above represents information about the genetic segment before the first exon on chromosome 1. The second row represents information about the first exon on chromosome 1. The exon contains 354 base pairs and the recombination and mutation rates in this exon are 10^{-8} per site per generation. The other columns of the data frame are described in the `SimRVSequences` documentation. The recombination rate between adjacent exons is set to the number of base pairs in the intervening intronic segment (`segLength`) multiplied by 10^{-8} per base pair per generation (`recomb_rate`). Further, the gap between two unlinked chromosomes is set to be a single base pair and the recombination rate between them is set to be 0.5 per base pair per generation (Harris and Nielsen 2016). Since we are interested in exon-only data, the mutation rate outside exons is set to zero and mutation rates inside exons is set to 10^{-8} per base pair per generation (Nieuwoudt, Brooks-Wilson, and Graham 2020).

We need three variables from `s_map` to create the recombination map for simulating exon-only data by SLiM: `recRate`, `mutRate` and `endPos`. We select these three variables and shift the `endPos` variable forward by one unit because SLiM reads arrays starting at position as 0 rather than 1. We save the resulting output as a text file (`Slim_Map_chr.txt`) to be used as a recombination map for SLiM.

```r
# Restrict output to the variables required by SLiM
slimMap <- s_map[, c("recRate", "mutRate", "endPos")]

# Shift endPos up by one unit
```

```
slimMap$endPos <- slimMap$endPos - 1

# Print first four rows of slimMap
head(slimMap, n = 4)
```

```
##    recRate mutRate endPos
## 1 0.00e+00   0e+00      0
## 2 1.00e-08   1e-08    354
## 3 3.85e-06   0e+00    355
## 4 1.00e-08   1e-08    464
```

```
# Write the results to a text file
write.table(slimMap, file ="Slim_Map_chr.txt")
```

The next section explains the demographic model we will use to simulate the population-level, exon-only SNV sequences. These sequences will be randomly sampled from the population to be assigned to the founders of our ascertained pedigrees in later steps of the workflow.

2 Specify the demographic model

Demographic models play a major role in understanding the genetic patterns in human populations. Throughout human evolution, different demographic events such as expansion, migration, splitting etc. have occurred, affecting genetic diversity (Ragsdale and Gravel 2019). The population-genetics literature has several established demographic models inferred from genetic data (Gutenkunst et al. 2009). Some of these models have been compiled in `stdpopsim`, a standard library of population-genetic simulation models (Adrion et al. 2020). At the time of writing, this library contains around nine demographic models. Among these, we select the **American Admixture** demographic model of Browning et al. (2018) because the family-based study motivating our work is in a North American population.

2.1 American admixture demographic model

In the American-Admixture model (Browning et al. 2018), the pre-admixture model parameters are selected from the Out-of-Africa model of Gravel et al. (2011) illustrated below.

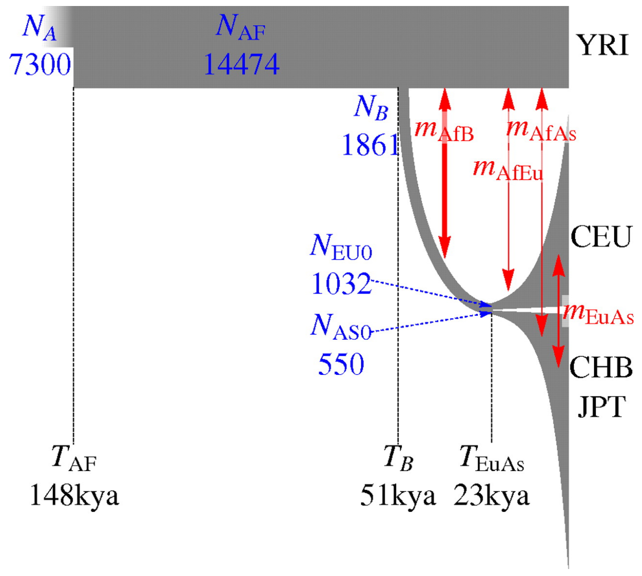


Figure 2: The inferred Out-of-Africa demographic model.

In the figure, the parameter estimates have been rounded and times are expressed in kilo-years before present (kya). The demographic model has three populations representing Africa, Europe and Asia. The initial effective population size of Africa was 7310 individuals which then increased to $14,474$ individuals 5920 generations ago (148 kya, assuming a generation time of 25 years). About 2040 generations ago (51 kya), the out-of-Africa migration event occurred with a migrating effective population size of 1861 individuals. Then migration occurred between Africa and out-of-Africa populations with a rate of 1.5×10^{-4} per generation. About 920 generations ago (23 kya), the out-of-Africa population split into two populations, Europe and Asia, with effective sizes of 1032 and 554 individuals, respectively. These two populations then grew at rates of 3.8×10^{-3} per generation for Europe and 4.8×10^{-3} per generation for Asia. Further, between these three populations, (Africa , Europe and Asia) migrations occurred. The migration rates per generation were 2.5×10^{-5} between Africa and Europe, 7.8×10^{-6} between Africa and Asia, and 3.11×10^{-5} between Europe and Asia (Browning et al. 2018). Admixing started about 12 generations ago (0.3 kya) with the initial effective size of the admixed population being $30,000$ individuals. The growth rate of the admixed population was 5% per generation with $\frac{1}{6}$ of the admixed population originating from African ancestry, $\frac{1}{3}$ from European ancestry and $\frac{1}{2}$ from Asian ancestry (Browning et al. 2018).

As described in the next section, we use SLiM together with the inferred American-admixture demographic model to simulate population-level exon-only SNV sequences.

3 Simulate population genetic data with SLiM

The following SLiM script generates genome-wide exon-only SNV sequences for a population under the American-Admixture demographic model. The script is a `.slim` file, which we have embedded in an R code chunk (that is not run). The original `SLiM_American_Admixture.slim` file can be found on our GitHub page at https://github.com/SFUStatgen/SeqFamStudy/.

```
initialize() {

// Seed number which helps to reproduce the same result
setSeed(2181144364021);

// Read recombination map created by SimRVSequence R package

lines = readFile("~/Slim_Map_chr.txt");
Rrates = NULL;
Mrates = NULL;
ends = NULL;

for (line in lines)
{
components = strsplit(line);
ends = c(ends, asInteger(components[3]));
Rrates = c(Rrates, asFloat(components[1]));
Mrates = c(Mrates, asFloat(components[2]));
}
Exomelength = ends[size(ends)-1];

initializeRecombinationRate(Rrates, ends);

initializeMutationRate(Mrates, ends);

initializeSex("A"); // Specifies modeling of an autosome

initializeMutationType("m1", 0.5, "g", -0.043, 0.23); //non-synonymous
initializeMutationType("m2", 0.5, "f", 0.0); // synonymous

m1.mutationStackPolicy = "l";
m2.mutationStackPolicy = "l";

initializeGenomicElementType("g1", m1, 1); // positions 1 and 2
initializeGenomicElementType("g2", m2, 1); // positions 3

starts = repEach(seqLen(asInteger(round(Exomelength/3))) * 3, 2) +
  rep(c(0,2), asInteger(round(Exomelength/3)));
end_pos = starts + rep(c(1,0), asInteger(round(Exomelength/3)));
types = rep(c(g1,g2), asInteger(round(length(starts)/2)));

initializeGenomicElement(types, starts, end_pos);

}

// Initialize the ancestral African population
1 { sim.addSubpop("p1", asInteger(round(7310.370867595234))); }

// End the burn-in period; expand the African population
73105 { p1.setSubpopulationSize(asInteger(round(14474.54608753566))); }

// Split Eurasians (p2) from Africans (p1) and set up migration
76968 {
sim.addSubpopSplit("p2", asInteger(round(1861.288190027689)), p1);
p1.setMigrationRates(c(p2), c(15.24422112e-5));
p2.setMigrationRates(c(p1), c(15.24422112e-5));
}

// Split p2 into European (p2) and East Asian (p3); resize; migration
78084 {
sim.addSubpopSplit("p3", asInteger(round(553.8181989)), p2);
p2.setSubpopulationSize(asInteger(round(1032.1046957333444)));
p1.setMigrationRates(c(p2, p3), c(2.54332678e-5, 0.7770583877e-5));
p2.setMigrationRates(c(p1, p3), c(2.54332678e-5, 3.115817913e-5));
p3.setMigrationRates(c(p1, p2), c(0.7770583877e-5, 3.115817913e-5));
}

// Set up exponential growth in Europe (p2) and East Asia (p3)
78084:79012{
t = sim.generation - 78084;
p2_size = round(1032.1046957333444 * (1 + 0.003784324268)^t);
p3_size = round(553.8181989 * (1 + 0.004780219543)^t);
p2.setSubpopulationSize(asInteger(p2_size));
p3.setSubpopulationSize(asInteger(p3_size));
}

// Create the admixed population
79012 early(){
sim.addSubpop("p4", 30000);
//This new subpopulation is created with 30000 new empty individuals
p4.setMigrationRates(c(p1, p2, p3), c(0.1666667, 0.3333333, 0.5));
}
```

```
//After this early() event, SLiM will generate offspring, and the empty individuals in p4
will be // discarded and replaced by migrant offspring from p1, p2 and p3 as requested.
79012 late(){
p4.setMigrationRates(c(p1, p2, p3), c(0, 0, 0));
}

// Set up exponential growth in admixture (p4)
79012:79024 {
t = sim.generation - 79012;
p4_new_size = round(30000 * (1 + 0.05)^t);
p4.setSubpopulationSize(asInteger(p4_new_size));
}

// Output and terminate
79024 late() {
p4.individuals.genomes.output(filePath = "~/SLiM_output.txt");
}
```

Before the simulation starts, we need to initialize the mutation rate, recombination rate, genomic structure and so forth as the simulation parameters (Haller et al. 2019). We read the recombination map into SLiM using the `readFile()` function. Inside this function, we supply the path to our recombination map text file. Then we create three null vectors named `Rrates`, `Mrates` and `ends` to save the recombination rates, mutation rates and end positions of each exon in our recombination map, respectively.

Next, we use a `for`-loop to move along the genome, reading each line of the recombination map and:

- save the recombination rate, mutation rate and end position of each genomic segment,
- initialize the recombination rate for each genomic segment with the `initializeRecombinationRate()` function, by specifying the rate and the end position of the genomic segment,
- initialize the mutation rate for each genomic segment with the `initializeMutationRate()` function,
- specify that the genomic segment belongs to an autosomal chromosome with the `initializeSex()` function,
- specify the mutation type for each genomic segment with the `initializeMutationType()` function (see below),
- specify the mutation stacking policy for each genomic segment with the `mutationStackPolicy` command (see below),
- specify the type for each genomic segment with the `initializeGenomicElementType()` function (see below).

In exons, the last base-pair position in a three base-pair codon (coding for an amino acid in a protein) is a synonymous site. Synonymous sites are viewed as selectively neutral in comparison to the first two base-pair positions in a codon, which are non-synonymous. Therefore, we simulate two types of mutations: synonymous and non-synonymous. The `initializeMutationType`("m1," 0.5, "g," -0.043, 0.23) callback in the for-loop explains all the parameters that are held by the "m1" mutation type. We use "m1" to represent the non-synonymous mutations. These non-synonymous mutations have a dominance coefficient of 0.5 and the selection coefficient is generated from a gamma distribution with mean -0.043 and shape parameter is 0.23 (Harris and Nielsen 2016). We initialize the synonymous mutations separately with another call to the `initializeMutationType()` function. In `initializeMutationType`("m2," 0.5, "f," 0.0) callback, the "m2" mutation type represents the synonymous mutations and they have a fixed selection coefficient denoted by "f." The selection coefficient of this type of mutation is always 0, as seen in the fourth argument of the function. The dominance coefficient in the second argument of the function is 0.5.

In SLiM (as in biology), the individuals rather than the mutations are under selection. Selection acts on the individual, through their fitness value. The fitness value of an individual is calculated from the fitness effects of all the mutations carried by that individual (based upon their selection coefficient, dominance coefficient, and heterozygous/homozygous state). All the fitness effects are multiplied together to produce the individual fitness. The individual fitness value then affects selection. Specifically, in the default Wright-Fisher (WF) model of SLiM (which we use), lower fitness means a lower probability of mating. As a result, deleterious mutations tend to decrease in frequency and beneficial mutations tend to increase in frequency.

SLiM allows for recurrent mutations at a given base position on a given sequence (Haller et al. 2019). By default, SLiM "stacks" any mutations that occur in the same location as pre-existing mutations on a given sequence. This default behaviour of "mutation stacking" ("s" for stacked), is changed to "l" (last) with the command `m1.mutationStackPolicy = "l"`, so that new mutations occurring in the same location as pre-existing mutations on a given sequence replace the pre-existing mutations.

The next initialization task is to create the chromosome structure. In SLiM we can model different genomic structures in the chromosomes. We consider exons only, which have two genomic element types: one for non-synonymous sites (base positions 1 and 2 of a codon) and the other for synonymous sites (base position 3 of a codon). These genomic element types are called "g1" and "g2" and alternate as g1, g2, g1, g2, g1, etc. along the exome until the end position of a chromosome is reached. The first genomic-element type corresponds to non-synonymous sites, is initialized as "m1" and could have mutations with selection coefficients that come from the negative gamma distribution. The second genomic element type corresponds to the synonymous sites, is initialized as "m2" and could have neutral mutations. We use the `initializeGenomicElementType()` function to specify these two genomic elements and our exome structure. For example, `initializeGenomicElementType("g1", m1, 1)` specifies that genomic element type "g1" is defined as using mutation type "m1" for all of its mutations. The second genomic element type "g2" is defined as using mutation type "m2" for all its mutations. Then we create the alternating start and end positions of the "g1" and "g2" genomic elements along the exome. Finally, we initialize the two genomic elements "g1" and "g2" with `initializeGenomicElement()`, supplying their starting and ending positions along the exome.

After the `initialize()` callbacks end, we run our simulation under the Out-of-Africa model described in the SLiM manual (Haller et al. 2019), with exact parameter estimates from Gravel et al. (2011). In the first generation, the African ancestral population, labelled "p1," is created with the function `sim.addSubpop()` and initial effective population size of 7310 individuals. Haller et al. (2019) start the model at 79024 generations back from the present (gbp) and set it to be generation 0 in the forwards simulation. The simulation then takes 10*`African ancestral population size` generations as the neutral burn-in time (Haller et al. 2019). At generation 73105 in the simulation (5919 gbp), the ancestral population, "p1," increases in effective size from $\sim$ 7310 to $\sim$ 14474 individuals. Subsequently, at generation 76968 of the simulation (2056 gbp), the African ancestral population, "p1," splits into the Eurasian ancestral sub-population, "p2," and migration starts between these two sub-populations. The command `p1.setMigrationRates` sets the migration rate from the African to the Eurasian ancestral sub-population, while `p2.setMigrationRates` sets the migration rate from the Eurasian ancestral to the African sub-population. Then at 78084 generations in the simulation (940 gbp), the "p2" Eurasian sub-population splits into European and Asian sub-populations. We create a new sub-population,"p3," to represent the Asian sub-population and let the Eurasian ancestral sub-population become the European sub-population. After the Eurasian ancestral population becomes the Asian and European sub-populations, we allow for migration between the African, Asian and European sub-populations, setting the migration rates according to the literature. Then, starting from 78084 generations in the simulation (940 gbp), we specify exponential growth in European (p2) and Asian (p3) sub-populations, until 79012 generations in the simulation (12 gbp). At 79012 generations (12 gbp), we create the American admixed sub-population with an initial effective population size of 30000 individuals and set the migration rates between the admixed and the other three sub-populations according to Browning et al. (2018). Once the admixed sub-population is created, migration into and out of it is stopped and it grows exponentially at rate 5% per generation until the present at 79024 generations into the simulation. Finally in generation 79024 of the simulation (the present) we terminate our SLiM simulation and collect the output.

We only consider the SLiM output for the American admixed sub-population. We extract the genomic sequences of all individuals in the admixed sub-population with the function `p4.individuals.genomes.output()` obtaining output formatted as follows:

```
#OUT: 79024 GS 107752 /project/6007536/epasiedn/SLiM/American__Admixture/SLiM__output.txt
Mutations:
7229 50171 m2 51287555 0 0.5 p1 5 60626 ...
13218 484904 m2 39812003 0 0.5 p1 45 9536 ...
5202 762125 m2 36490340 0 0.5 p1 70 64099 ...
...
Genomes:
p*:0 A 0 1 2 3 4 5 6 7 8 9 10 11 ...
p*:1 A 10605 1 2 3 10606 4 5 6 8 10607 10608 9 ...
p*:2 A 10605 1 2 4 15639 15640 6 10608 15641 15642 15643 15644
p*:3 A 0 1 2 19096 19097 4 6 19098 19099 9 10 19100
...
```

In the above output, the first line starts with "# OUT:" followed by the generation of the simulation (79024) from which the output is obtained. Then "GS" tells us the data is formatted as "genomes SLiM format" and this is followed by the number of haploid genomes (2* number of individuals). Finally, the full path where we save the output is printed.

The second line of the output starts the mutation section. In the mutation section, each line represents a mutation which is currently segregating in the population and the nine fields on a line represent the mutation properties. The first field is the SLiM-generated identifier number which helps to identify the mutation easily within the program. The second field is the mutation's identification number. The third field represents the type of the mutation. The fourth field is the base-pair position of the mutation on the chromosome. The fifth and sixth fields represent selection and dominance coefficients, respectively. The seventh field is the sub-population in which the mutation originated. The eighth field is the generation of the simulation when the mutation arose. Finally, the ninth field represents the number of copies of the mutation in the sub-population.

The last section in the output represents the genomes section. In the genome section, a line corresponds to a haploid genome in the sub-population. For example, in the first line, "p*: 0," means the 0th genome of the sub-population . Then "A" represents autosome, the type of the genome. This is followed by the SLiM-generated identification numbers of all the mutations carried by this haploid genome. Recall that the SLiM-generated identification numbers are in the first field of the mutation section.

3.1 Simulation on Compute Canada Cluster

This SLiM simulation is highly memory intensive and not suitable for most personal computers. We therefore use the Compute Canada cluster (http://www.computecanada.ca) as described next. First, we need to install the SLiM software. The way we install the software is exactly the same as how we install the software on our own computer. Use the SLiM manual guidelines for this task. After we install SLiM, we use a job scheduler on the compute cluster to run our jobs. On the Compute Canada Cluster, the job scheduler is the **Slurm Workload Manager** . Slurm helps to allocate resources and time, and provides methods to execute our work. To run the SLiM script we write the following Slurm script, `job_serial.sh`:

```
#!/bin/bash
#SBATCH --account=def-jgraham
#SBATCH --ntasks=1
#SBATCH --time=7-05:05:00
#SBATCH --mem=64000M

module load StdEnv/2020 gcc/9.3.0 slim/3.4.0

slim SLiM_American_Admixture.slim
```

The content of `job_serial.sh` is described as follows.

- `#SBATCH-account=def-jgraham` specifies the account name. In this example, the account name is "def-jgraham."

- `#SBATCH-ntasks=1` defines the number of processors. We request 1 processor to run the program.
- `#SBATCH-time=7-05:05:00` specifies the time limit for the job. To avoid the simulation being stopped prematurely for running over the allocated time, we try to err on the side of allocating too much.
- `#SBATCH-mem=640000M` specifies memory required for our simulation. We request 64GB. Next, the executable commands are aligned in the script file:
- `module load StdEnv/2020 gcc/9.3.0 slim/3.4.0` loads the SLiM version we installed on the Cluster.
- `slim SLiM_American_Admixture.slim` calls SLiM to run the SLiM script, `SLiM_American_Admixture.slim`.

To submit the slurm script to the cluster, we type the following in our log-in node:

```
[epasiedn@gra-login2 American_Admixture]$ sbatch job_serial.sh
```

This SLiM simulation took approximately 3 days to complete on the Compute Canada cluster. The SLuRM script allocated 64GB to run the simulation. Out of this 64GB, the job utilized 43.61GB. The output of this SLiM simulation is saved as `SLiM_output.txt` and we use this file as one of the inputs for the third step of our workflow. We will return to the `SLiM_output.txt` output in our third RMarkdown document.

3.2 Summary Statistics

We use R and the output in `SLiM_output.txt` to obtain summary statistics for the American admixed population. We start by reading `SLiM_output.txt` into R.

```
library(SimRVSequences)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(Matrix)
```

```
##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
# Read the SLiM output text file to R
# Note: Change the path for the file as necessary.
exDat <- readLines("D:/SFU_Vault/SLiM_Output/SLiM_output.txt")
```

The file size of `SLiM_output.txt` is approximately 6 GB and takes approximately 1 minute to load into R on a Windows OS with an i7-8550U @ 1.8GHz,16GB of RAM.

```r
# Read the mutations and genomic sections in the output
MutHead <- which(exDat == "Mutations:")
GenHead <- which(exDat == "Genomes:")

# Get the population count in sequences
popCount <- as.numeric(unlist(strsplit(exDat[1],
                                       split = " ", fixed = TRUE))[4])

# Population count in individuals
popCount/2
```

```
## [1] 53876
```

The number of individuals in the simulated American admixed population is 53,876.

```r
# Extract mutation data from SLiM's Mutation output
# only retaining the tempID, type, position,
# selection coefficient and count of each mutation
MutOut <- do.call(rbind, strsplit(exDat[(MutHead + 1):(GenHead - 1)], split = " ",
fixed = TRUE))
MutData <- data.frame(tempID = as.numeric(MutOut[, 1]),
                      type = MutOut[, 3],
                      position = as.numeric(MutOut[, 4]),
                      selCoef = as.numeric(MutOut[, 5]),
                      count = as.numeric(MutOut[, 9]),
                      stringsAsFactors = TRUE)
nrow(MutData)
```

```
## [1] 862243
```

The number of mutations segregating in the American admixed population is 862,243. We next examine what percentage of these have derived allele frequency less than 1% in the population.

```r
# Add 1 to temp ID so that we can easily associate mutations to columns.
# By default SLiM's first tempID is 0, not 1.
MutData$tempID <- MutData$tempID + 1
# First position in SLiM is 0, not 1
MutData$position <- MutData$position + 1

# Calculate the population derived allele frequency.
# Divide the allele count by the population size.
MutData$afreq <- MutData$count/(popCount)

# Get the percentage of SNVs whose allele frequency < 0.01
af_less <- which(MutData$afreq < 0.01)
af_less_per <- length(af_less)/ nrow(MutData)

af_less_per
```

```
## [1] 0.9426565
```

Among the 862,243 mutations segregating in the simulated American-admixed population, approximately 94% have derived allele frequencies less than 1%. This is slightly less than the approximately 97% of single-nucleotide variants observed to have alternate allele frequencies less than 1% in the TopMed study of the American population (Taliun et al. 2021).

Next, we check the percentage of mutations that are singletons in the simulated American admixed population.

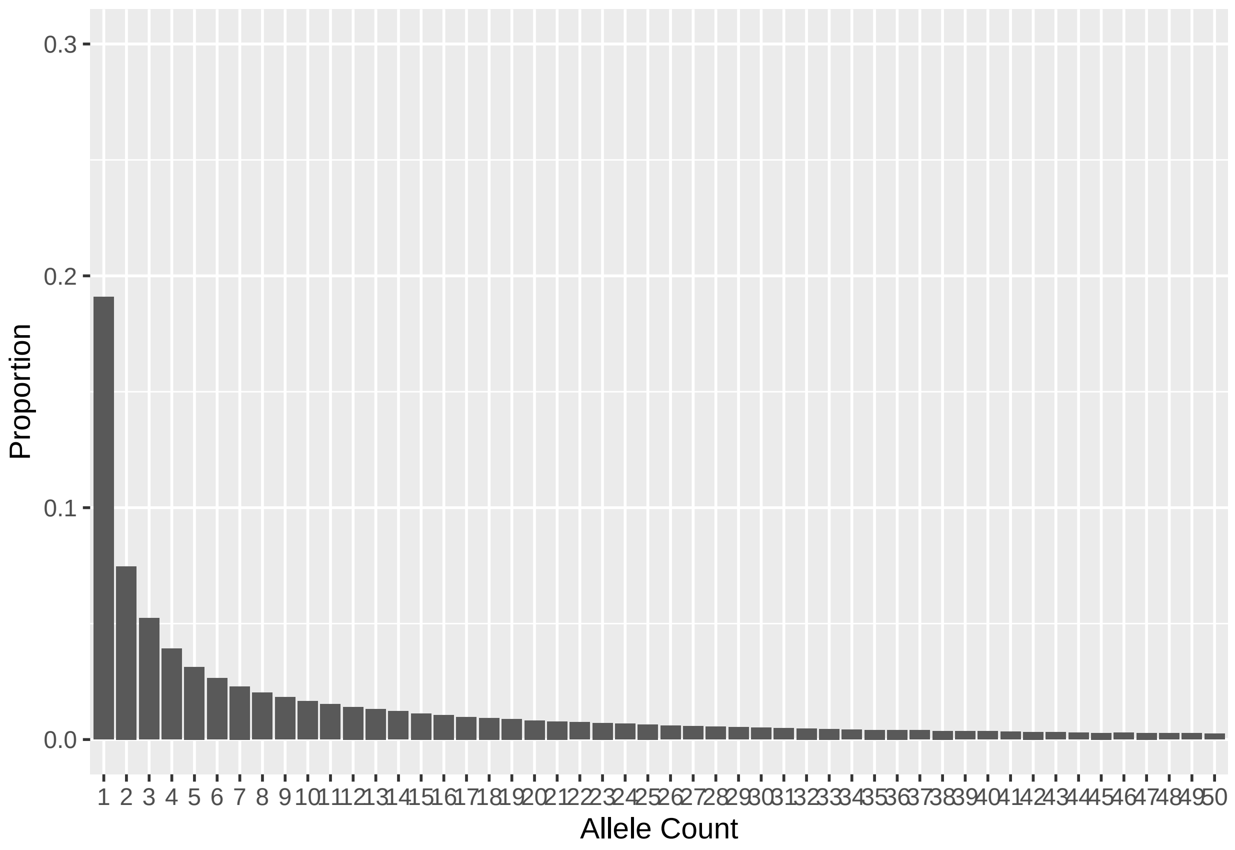
```r
# Use the prevalence (the number of times that the mutation occurs in any genome)
# column in MutData dataframe to calculate the singleton percentage
singleton <- MutData %>%
  count(count) %>%
  mutate(percentage = n/nrow(MutData))

colnames(singleton) <- c("number_of_allele", "count", "proportion")
head(singleton)
```

```
##   number_of_allele  count proportion
## 1                1 164624 0.19092530
## 2                2  64401 0.07469008
## 3                3  45210 0.05243301
## 4                4  33804 0.03920473
## 5                5  27019 0.03133571
## 6                6  22881 0.02653660
```

Among the 862,243 mutations, only 19% are singletons. By contrast, in the TopMed study, about half the variants are singletons (Taliun et al. 2021). The following figure illustrates the allele frequency spectrum in the simulated population.

```r
# Plot the first 50 sites in the allele frequency spectrum
ggplot(singleton) +
  geom_bar(mapping = aes(x = as.factor(number_of_allele),
                                              y = proportion),
                              stat="identity",
                              position="dodge") +
  xlab("Allele Count") +
  ylab("Proportion") +
  ylim(0, 0.3) +
  scale_x_discrete(limits= as.character(1:50))
```



We believe that our simulated American-admixed population has fewer rare variants and singletons than the TopMed population because it lacks TopMed's variety of source populations. Our SLiM simulation has only three source populations and we collect mutation data from only the American-admixed population. By contrast, the TopMed study considers the entire American population consisting of many more source populations as well as an admixed population. To investigate this hypothesis, we combined data from all four populations in our SLiM simulation to see if the singleton percentage increased. Due to the high computational cost, we simulated mutations for chromosomes 8 and 9 only. Combining all four populations, the percentage of singletons increased from 19 to 26% of mutations. An additional RMarkdown document, for supplementary material 1-B, discusses the commands to generate and summarize all the source populations and check the proportions of singletons, after combining the four populations.

References

Adrion, Jeffrey R, Christopher B Cole, Noah Dukler, Jared G Galloway, Ariella L Gladstein, Graham Gower, Christopher C Kyriazis, et al. 2020. "A Community-Maintained Standard Library of Population Genetic Models." Edited by Graham Coop, Patricia J Wittkopp, John Novembre, Arun Sethuraman, and Sara Mathieson. *eLife* 9 (June): e54967. https://doi.org/10.7554/eLife.54967.

Browning, Sharon R., Brian L. Browning, Martha L. Daviglus, Ramon A. Durazo-Arvizu, Neil Schneiderman, Robert C. Kaplan, and Cathy C. Laurie. 2018. "Ancestry-specific recent effective population size in the Americas." *PLoS Genetics*. https://doi.org/10.1371/journal.pgen.1007385.

Gravel, Simon, Brenna M. Henn, Ryan N. Gutenkunst, Amit R. Indap, Gabor T. Marth, Andrew G. Clark, Fuli Yu, Richard A. Gibbs, and Carlos D. Bustamante. 2011. "Demographic history and rare allele sharing among human populations." *Proceedings of the National Academy of Sciences of the United States of America*. https://doi.org/10.1073/pnas.1019276108.

Gutenkunst, Ryan N., Ryan D. Hernandez, Scott H. Williamson, and Carlos D. Bustamante. 2009. "Inferring the joint demographic history of multiple populations from multidimensional SNP frequency data." *PLoS Genetics*. https://doi.org/10.1371/journal.pgen.1000695.

Haller, Benjamin C, Philipp W Messer, Yoann Buoro, Deborah Charlesworth, Jeremy Van Cleve, Jean Cury, Michael Degiorgio, et al. 2019. "SLiM : An Evolutionary Simulation Framework CookBook," no. May.

Harris, Kelley, and Rasmus Nielsen. 2016. "The genetic cost of Neanderthal introgression." *Genetics*. https://doi.org/10.1534/genetics.116.186890.

Nieuwoudt, Christina, Angela Brooks-Wilson, and Jinko Graham. 2020. "SimRVSequences: An R package to simulate genetic sequence data for pedigrees." *Bioinformatics* 36 (7): 2295–97. https://doi.org/10.1093/bioinformatics/btz881.

Ragsdale, Aaron P., and Simon Gravel. 2019. "Models of archaic admixture and recent history from two-locus statistics." *PLoS Genetics* 15 (6): 1–19. https://doi.org/10.1371/journal.pgen.1008204.

Taliun, Daniel, Daniel N. Harris, Michael D. Kessler, Jedidiah Carlson, Zachary A. Szpiech, Raul Torres, Sarah A.Gagliano Taliun, et al. 2021. "Sequencing of 53,831 diverse genomes from the NHLBI TOPMed Program." *Nature* 590 (7845): 290–99. https://doi.org/10.1038/s41586-021-03205-y.

Appendix B

Supplementary Material 1-B: Combining Source Populations with the American-Admixed Population in SLiM

Supplementary Material 1-B: Combining Source Populations with the American-Admixed Population in SLiM

Nirodha Epasinghege Dona, Jinko Graham

2021-12-22

Supplementary Material 1-A discusses the SLiM simulation of the American-admixed population and the processing of its data. This document discusses how to obtain and process the data not just for the American-admixed population but also for all four populations in the simulation. We focus on chromosomes 8 and 9 only to reduce computational cost, and alter the original SLiM script as follows: 1. at the top of the script, we provide an abbreviated recombination map, `Slim_Map8n9.txt`, containing only those lines of the original recombination map (in the data frame, `s_map`) pertaining to chromosomes 8 and 9, and 2. at the bottom of the script, we call `sim.outputFull()` rather than `p4.individuals.genomes.output()` to obtain the output. Otherwise, the SLiM script remains exactly the same as the original in Supplementary Material 1-A.

The following R code chunk creates the abbreviated recombination map.

```r
# Create the file Slim_Map8n9.txt containing the recombination map
# for chromosomes 8 and 9 only. NB: We assume that you have already run
# SupplementaryMaterial_1A.Rmd so that the s_map object it creates is
# already in your R workspace.
library(SimRVSequences)
data("hg_exons")
s_map<-create_slimMap(exon_df = hg_exons)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.1     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(Matrix)
```

```
##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```r
library(data.table)
```

```
##
## Attaching package: 'data.table'
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
slimMap8n9<- s_map[(s_map[,"chrom"]==8 | s_map[,"chrom"]==9), c("recRate","mutRate","endPos")]
slimMap8n9$endPos <- slimMap8n9$endPos-1
write.table(slimMap8n9, file="Slim_Map8n9.txt")
```

Now we are ready to run the following SLiM script.

```
initialize() {

// Read in the abbreviated recombination map for chromosome 8 and 9.
lines = readFile("~/Slim_Map8n9.txt");
Rrates = NULL;
Mrates = NULL;
ends = NULL;

for (line in lines)
{
components = strsplit(line);
ends = c(ends, asInteger(components[3]));
Rrates = c(Rrates, asFloat(components[1]));
Mrates = c(Mrates, asFloat(components[2]));
}
Exomelength = ends[size(ends)-1];

initializeRecombinationRate(Rrates, ends);

initializeMutationRate(Mrates, ends);

initializeSex("A"); // Specifies modeling of an autosome

initializeMutationType("m1", 0.5, "g", -0.043, 0.23); //non-synonymous
initializeMutationType("m2", 0.5, "f", 0.0); // synonymous

m1.mutationStackPolicy = "l";
m2.mutationStackPolicy = "l";


initializeGenomicElementType("g1", m1, 1); // positions 1 and 2
initializeGenomicElementType("g2", m2, 1); // positions 3


starts = repEach(seqLen(asInteger(round(Exomelength/3))) * 3, 2) +
  rep(c(0,2), asInteger(round(Exomelength/3)));
end_pos = starts + rep(c(1,0), asInteger(round(Exomelength/3)));
types = rep(c(g1,g2), asInteger(round(length(starts)/2)));
initializeGenomicElement(types, starts, end_pos);

}
```

```
// Initialize the ancestral African population
1 { sim.addSubpop("p1", asInteger(round(7310.370867595234))); }

// End the burn-in period; expand the African population
73105 { p1.setSubpopulationSize(asInteger(round(14474.54608753566))); }

// Split Eurasians (p2) from Africans (p1) and set up migration
76968 {
sim.addSubpopSplit("p2", asInteger(round(1861.288190027689)), p1);
p1.setMigrationRates(c(p2), c(15.24422112e-5));
p2.setMigrationRates(c(p1), c(15.24422112e-5));
}

// Split p2 into European (p2) and East Asian (p3); resize; migration
78084 {
sim.addSubpopSplit("p3", asInteger(round(553.8181989)), p2);
p2.setSubpopulationSize(asInteger(round(1032.1046957333444)));
p1.setMigrationRates(c(p2, p3), c(2.54332678e-5, 0.7770583877e-5));
p2.setMigrationRates(c(p1, p3), c(2.54332678e-5, 3.115817913e-5));
p3.setMigrationRates(c(p1, p2), c(0.7770583877e-5, 3.115817913e-5));
}

// Set up exponential growth in Europe (p2) and East Asia (p3)
78084:79012{
t = sim.generation - 78084;
p2_size = round(1032.1046957333444 * (1 + 0.003784324268)^t);
p3_size = round(553.8181989 * (1 + 0.004780219543)^t);
p2.setSubpopulationSize(asInteger(p2_size));
p3.setSubpopulationSize(asInteger(p3_size));
}

// Create the admixed population
79012{
p2_new_size = p2.individualCount;
p3_new_size = p3.individualCount;
defineConstant("pop_size", c(p2_new_size, p3_new_size));
sim.addSubpop("p4", 30000);
p4.setMigrationRates(c(p1, p2, p3), c(0.1666667, 0.3333333, 0.5));
}
79012 late(){
p4.setMigrationRates(c(p1, p2, p3), c(0, 0, 0));
}

// Setup exponential growth in Europe (p2) and East Asia (p3)
79012:79024 {
t = sim.generation - 79012;
p2_new_size = round(pop_size[0] * (1 + 0.003784324268)^t);
p3_new_size = round(pop_size[1] * (1 + 0.004780219543)^t);
p4_new_size = round(30000 * (1 + 0.05)^t);
p2.setSubpopulationSize(asInteger(p2_new_size));
p3.setSubpopulationSize(asInteger(p3_new_size));
p4.setSubpopulationSize(asInteger(p4_new_size));
}
```

```
// Output for all populations (not just p4) and terminate
79024 late() {
sim.outputFull("~/SLiM_output_chr8&9.txt");
}
```

We read `SLiM_output_chr8&9.txt` into R and obtain the number of individuals in each population and in all populations combined.

```
# Read the SLiM output text file to R
# Note:Change the path for the file as necessary.
exDat <- readLines("/Users/jgraham/OneDrive - Simon Fraser University (1sfu)/NirodhaStuff/Data/SLiM_outp
# Read the mutations and genomic sections in the output
MutHead <- which(exDat == "Mutations:")
GenHead <- which(exDat == "Genomes:")
PopHead <- which(exDat == "Populations:")
IndHead <- which(exDat == "Individuals:")

# Get the population count for each source population

popCount_1 <- as.numeric(unlist(strsplit(exDat[PopHead + 1], split = " "))[2])
popCount_2 <- as.numeric(unlist(strsplit(exDat[PopHead + 2], split = " "))[2])
popCount_3 <- as.numeric(unlist(strsplit(exDat[PopHead + 3], split = " "))[2])
popCount_4 <- as.numeric(unlist(strsplit(exDat[PopHead + 4], split = " "))[2])


# Get the total population count
popCount <- popCount_1 + popCount_2 + popCount_3 + popCount_4
```

The following table of population sizes summarizes the output of the above commands.

Table 1: Population sizes.

Population	**size**
African	14,475
European	35,815
Asian	48,765
Admix	53,876
Total	**152,931**

```
# Extract mutation data from SLiM's Mutation output
# only retaining the tempID, type, position, selection coefficient and prevalence of each mutation
MutOut <- do.call(rbind, strsplit(exDat[(MutHead + 1):(IndHead - 1)], split = " ", fixed = TRUE))
MutData <- data.frame(tempID = as.numeric(MutOut[, 1]),
                      type = MutOut[, 3],
                      position = as.numeric(MutOut[, 4]),
                      selCoef = as.numeric(MutOut[, 5]),
                      count = as.numeric(MutOut[, 9]),
                      stringsAsFactors = TRUE)

nrow(MutData)
```

```
## [1] 142549
```

On chromosomes 8 and 9, the number of mutations segregating in all four populations is 142,549.

```
# Add 1 to temp ID so that we can easily associate mutations to columns.
# By default SLiM's first tempID is 0, not 1.
MutData$tempID <- MutData$tempID + 1
# First position in SLiM is 0, not 1
MutData$position <- MutData$position + 1

# Calculate the population derived-allele frequency.
# Divide the derived-allele count by the population size.
MutData$afreq <- MutData$count/(popCount)

# Get the percentage of SNVs whose derived-allele frequency is < 0.01
af_less <- which(MutData$afreq < 0.01)
af_less_per <- length(af_less)/ nrow(MutData)

af_less_per
```

```
## [1] 0.9616693
```

Among the 142,549 mutations on chromosomes 8 and 9, approximately 96% have frequencies less than 1%.

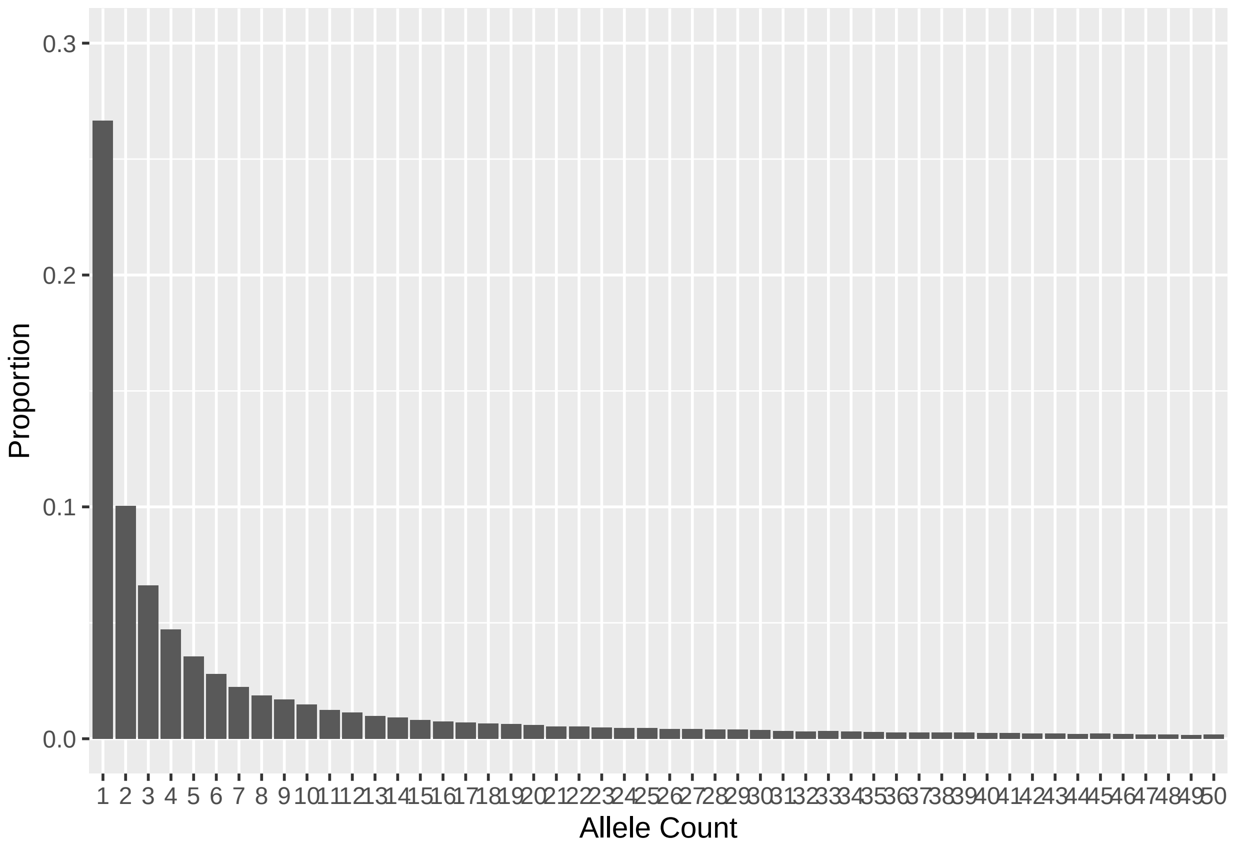
In Supplementary Material 1-A, we discuss how 26% of the variants in the combined populations were singletons. The following commands are used to calculate this percentage.

```
# Use the prevalence (the number of times that the mutation occurs in any genome)
# column in MutData dataframe to calculate the singleton percentage
singleton <- MutData %>% count(count) %>% mutate(percentage = n/nrow(MutData))
colnames(singleton) <- c("number_of_allele", "count", "proportion")
head(singleton)
```

```
##   number_of_allele count proportion
## 1                1 38012 0.26665918
## 2                2 14312 0.10040056
## 3                3  9436 0.06619478
## 4                4  6731 0.04721885
## 5                5  5051 0.03543343
## 6                6  3979 0.02791321
```

The following figure illustrates the derived-allele frequency spectrum for chromosomes 8 and 9.

```
# Plot derived-allele counts of up to 50 in the allele-frequency spectrum
ggplot(singleton) +
  geom_bar(mapping = aes(x = as.factor(number_of_allele),
                                         y = proportion),
                              stat="identity",
                              position="dodge") +
  xlab("Allele Count") +
  ylab("Proportion") +
  ylim(0, 0.3) +
  scale_x_discrete(limits= as.character(1:50))
```



0.3
0.2
0.1
0.0
Proportion
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
Allele Count

Appendix C

Supplementary Material 2: Simulate ascertained pedigrees

Supplementary Material 2: Simulate ascertained pedigrees

Nirodha Epasinghege Dona, Jinko Graham

2022-04-16

Contents

The second major step in our work-flow simulates 150 pedigrees ascertained to have four or more relatives affected with lymphoid cancer (the blue box labelled 2).

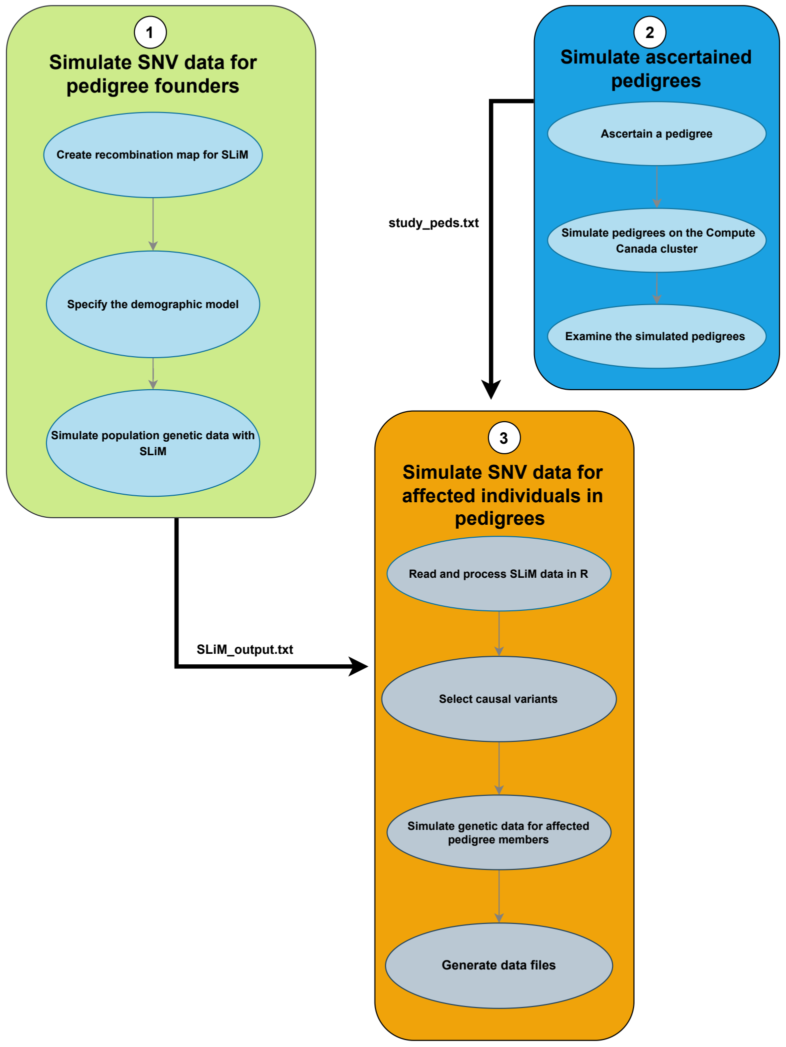

1
Simulate SNV data for pedigree founders
Create recombination map for SLiM
Specify the demographic model
Simulate population genetic data with SLiM
2
Simulate ascertained pedigrees
Ascertain a pedigree
Simulate pedigrees on the Compute Canada cluster
Examine the simulated pedigrees
study_peds.txt
3
Simulate SNV data for affected individuals in pedigrees
Read and process SLiM data in R
Select causal variants
Simulate genetic data for affected pedigree members
Generate data files
SLiM_output.txt

Figure 1: Work-flow for simulating the exome-sequencing data for ascertained pedigrees.

1 Ascertain a pedigree

We use the SimRVPedigree (Nieuwoudt et al. 2018) R package to ascertain a single pedigree simulated to contain four or more relatives affected with lymphoid cancer. Affected pedigree members can have either sporadically occurring disease or genetic disease caused by a single rare variant that is segregating in the pedigree. We refer to the causal rare variants as cRVs. The package constructs a pedigree by growing it from a single starting individual or "seed founder" and obtaining the seed founder's descendants. A cRV may be introduced into the seed founder with probability equal to either one or the carrier probability of a cRV in the population. When a cRV is introduced into the seed founder with probability equal to the carrier probability of a cRV in the population, the pedigree is ascertained from the **general population**. A *genetic* pedigree is defined as a pedigree in which the seed founder carries a cRV, whereas a *sporadic* pedigree is defined as a pedigree in which the seed founder does not carry a cRV. Ascertained pedigrees may be either genetic or sporadic, and may contain both genetic and sporadic cases. Once a cRV is introduced into a seed founder, it is transmitted from parent to offspring according to Mendel's law. The age-specific life events of the seed founder and his/her descendants such as birth, disease onset and death are modelled according to the cRV carrier status of the individual. The modeling requires specification of the age-specific incidence rates of disease, the age-specific hazard rates of death and the genetic relative-risk (GRR) of disease.

We ascertain a single pedigree from the **general population** using `simRV_ped()`, the core function of the `SimRVPedigree` package. The `sim_RVped()` function simulates all life events of a seed founder and his/her descendants, as described by Nieuwoudt et al. (2018). Starting at the birth of an individual, the waiting times of the possible next life events of the individual – disease onset, reproduction and death – are generated. The event with the minimum waiting time is selected as the individual's next life event. The waiting time is added to the current age of the individual and the corresponding life event is recorded. These steps are repeated until the individual dies or the study reaches its stop year. Further details of this function can be found in Nieuwoudt and Graham (2018).

The required arguments of `simRV_ped()` are: `hazard_rates`, `GRR`, `num_affected`, `ascertain_span`, `FamID`, and `founder_byears`. Non-required arguments of specific interest to us are: `stop_year`, `carrier_prob`, `RV_founder`, `recall_probs` and `first_diagnosis`. A short description of these arguments follows.

- **`hazard_rates-`** We use the `AgeSpecific_Hazards` dataset in the `SimRVPedigree` package. The first column of the `AgeSpecific_Hazards` data-frame gives the age-specific hazard rates for the disease in the general population. The second column gives the age-specific hazard rates for death in the unaffected population. The third column gives the age-specific hazard rates for death in the affected population.
- **`GRR-`** the genetic relative-risk; i.e, the risk of disease for individuals who carry a copy of a cRV relative to those who carry no copies of a cRV. We use 50 as the GRR.
- **`num_affected-`** the minimum number of disease-affected members needed to ascertain the pedigree is set to 4.
- **`ascertain_span-`** the period of ascertainment of the pedigree; i.e., (start-year, end-year) is set to (2000 , 2010).
- **`FamID-`** the family identity number of the simulated pedigree. We assign a vector that contains values 1 to 150 since we need to generate 150 pedigrees.
- **`founder_byears-`** the period for the possible birth year of a seed founder is set to be (1880, 1920).
- **`stop_year-`** 2020 is set to the year in which we stop collecting data.
- **`carrier_prob-`** the probability that an individual in the general population carries a cRV is set to 0.001.
- **`RV_founder-`** is set to `FALSE`, i.e., the seed founder carries a cRV with probability equal to the carrier probability (0.001) of a cRV in the population.
- **`recall_probs-`** the proband's recall probabilities of relatives in the pedigree is set to (1, 1, 1, 1, 0.75, 0.5, 0.25, 0.125, 0). These probabilities imply that first to fourth-degree relatives of the proband (e.g. fourth degree = great aunt) are recalled with probability 1, all fifth-degree relatives (e.g. first cousin once removed) of the proband are recalled with probability 0.75, and so forth.
- **`first_diagnosis-`** the earliest year after which reliable diagnoses can be made regarding the disease-affection status is set as 1940.

We set the above values for the arguments in the `sim_RVped()` function and use the Compute Canada cluster for the simulation. We use an array job on the cluster, with a processor (CPU) to simulate each pedigree. Due to the requirement that at least four relatives be known to be affected, the ascertainment of each pedigree is time-consuming and the simulation time is variable across pedigrees. However, as discussed below, we allocate up to 24 hours to simulate a pedigree.

2 Simulate pedigrees on the Compute Canada cluster

We use the following slurm batch file to submit an array job with a processor for each of the 150 pedigrees.

```
#!/bin/bash
#SBATCH --account=def-jgraham
#SBATCH --array= 1-150
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=4000M
#SBATCH --time=23:59:00

module load  nixpkgs/16.09 gcc/5.4.0  r/3.5.0

echo "This is job $SLURM_ARRAY_TASK_ID out of $SLURM_ARRAY_TASK_COUNT jobs."

R CMD BATCH --no-save SimRVpedigree.R
```

In the batch script above, the parameters are set as follows.

- `#SBATCH-account=def-jgraham` - specifies the project account on Compute Canada. In this example, the project account is “def-jgraham”.
- `# SBATCH -array= 1-150` - specifies an array of tasks with indices 1 through 150, one for each of the 150 pedigrees. More generally, users can specify indices x through y to obtain $y - x + 1$ pedigrees.
- `#SBATCH-ntasks=1` - defines the number of array tasks per processor. We request 1 processor to run each task (i.e. each one of our 150 pedigrees, will use 1 CPU).
- `#SBATCH-mem=40000M` - specifies memory that we require to run each task in the array. We request 4GB.
- `#SBATCH-time=23:59:00` - specifies the time limit for each task. If we allocate 24 hours or more to run a task, we must wait longer in the queue to start a task than if we allocate less than 24 hours.
- `module load nixpkgs/16.09 gcc/5.4.0 r/3.5.0` - loads the R version that we installed.
- `echo "This is job $ SLURM_ARRAY_TASK_ID out of $ SLURM_ARRAY_TASK_COUNT jobs."` - prints the job number out of 150 tasks.

We use the `R CMD BATCH` command to submit the `SimRVpedigree.R` script to the cluster. The contents of the script is given below.

```
# load the SimRVPedigree library
library(SimRVPedigree)

# Create hazard object from AgeSpecific_Hazards data
data(AgeSpecific_Hazards)
my_HR = hazard(AgeSpecific_Hazards)

# Get the Unix environmental variable for array job id.
# This id is created by the cluster for each job.
dID = Sys.getenv("SLURM_ARRAY_TASK_ID")

# Set a seed value to assure the reproducibility.
```

```
seed = as.numeric(dID)
set.seed(seed)

generatePeds = function(dataID){

  # Read the R function that do the analysis
  out =  sim_RVped(hazard_rates = my_HR,
            GRR = 50, FamID = dataID,
            RVfounder = FALSE,
            founder_byears = c(1880, 1920),
            ascertain_span = c(2000, 2010),
            stop_year = 2020,
            recall_probs = c(1, 1, 1, 1, 0.75, 0.5, 0.25, 0.125, 0),
            carrier_prob = 0.001,
            num_affected = 4,
            first_diagnosis = 1940)[[2]]

  # Save the results separately for each dataset.
  write.table(out, file = paste0("/project/6007536/epasiedn/Array_jobs/",
                          dataID,".txt"))
}

# Run the function.
generatePeds(dID)
```

In the above script, we create a function `generatePeds()` which calls the `sim_RVped()` function. The `generatePeds()` function has one argument,`dataID`, for the task identifier created by the cluster scheduler. The environment variable, `SLURM_ARRAY_TASK_ID`, identifies each task in the array. In the last two lines of the R script above, we use `dID = Sys.getenv("SLURM_ARRAY_TASK_ID")` to get the task identifier and assign it as the argument to the `generatePeds()` function. `dID` is also assigned as the random seed for each pedigree. Since each pedigree is a task that is run separately on a different CPU in the cluster, we want to assign a different seed value each time.

Among the 150 tasks, 140 manage to run within the allocated time period. Some tasks take longer because some pedigrees take a longer time to ascertain. The unfinished tasks are run again, with a different random seed. We use the linux command `ls` to identify the finished jobs. This command returns all the files in our directory, so that we can see which task IDs are missing. Among all 150 tasks, IDs 14, 30, 48, 50, 63, 73, 83, 94, 102 and 129 are unfinished. The following code chunk shows how we identify the unfinished tasks.

```
[epasiedn@cedar1 Array_jobs_check]$ ls
100.txt  140.txt  45.txt  88.txt          slurm-19422255_124.out    slurm-19422255_26.out
101.txt  141.txt  46.txt  89.txt          slurm-19422255_125.out    slurm-19422255_27.out
103.txt  142.txt  47.txt  8.txt           slurm-19422255_126.out    slurm-19422255_28.out
104.txt  143.txt  49.txt  90.txt          slurm-19422255_127.out    slurm-19422255_29.out
105.txt  144.txt  4.txt   91.txt          slurm-19422255_128.out    slurm-19422255_2.out
106.txt  145.txt  51.txt  92.txt          slurm-19422255_129.out    slurm-19422255_30.out
107.txt  146.txt  52.txt  93.txt          slurm-19422255_12.out     slurm-19422255_31.out
108.txt  147.txt  53.txt  95.txt          slurm-19422255_130.out    slurm-19422255_32.out
109.txt  148.txt  54.txt  96.txt          slurm-19422255_131.out    slurm-19422255_33.out
10.txt   149.txt  55.txt  97.txt          slurm-19422255_132.out    slurm-19422255_34.out
110.txt  150.txt  56.txt  98.txt          slurm-19422255_133.out    slurm-19422255_35.out
111.txt  15.txt   57.txt  99.txt          slurm-19422255_134.out    slurm-19422255_36.out
112.txt  16.txt   58.txt  9.txt           slurm-19422255_135.out    slurm-19422255_37.out
113.txt  17.txt   59.txt  job_array.sh    slurm-19422255_136.out    slurm-19422255_38.out
```

```
114.txt  18.txt  5.txt   SIMrvpedigree.R         slurm-19422255_137.out  slurm-19422255_39.out
115.txt  19.txt  60.txt  SIMrvpedigree.Rout      slurm-19422255_138.out  slurm-19422255_3.out
116.txt  1.txt   61.txt  slurm-19422255_100.out  slurm-19422255_139.out  slurm-19422255_40.out
117.txt  20.txt  62.txt  slurm-19422255_101.out  slurm-19422255_13.out   slurm-19422255_41.out
118.txt  21.txt  64.txt  slurm-19422255_102.out  slurm-19422255_140.out  slurm-19422255_42.out
119.txt  22.txt  65.txt  slurm-19422255_103.out  slurm-19422255_141.out  slurm-19422255_43.out
11.txt   23.txt  66.txt  slurm-19422255_104.out  slurm-19422255_142.out  slurm-19422255_44.out
120.txt  24.txt  67.txt  slurm-19422255_105.out  slurm-19422255_143.out  slurm-19422255_45.out
121.txt  25.txt  68.txt  slurm-19422255_106.out  slurm-19422255_144.out  slurm-19422255_46.out
122.txt  26.txt  69.txt  slurm-19422255_107.out  slurm-19422255_145.out  slurm-19422255_47.out
123.txt  27.txt  6.txt   slurm-19422255_108.out  slurm-19422255_146.out  slurm-19422255_48.out
124.txt  28.txt  70.txt  slurm-19422255_109.out  slurm-19422255_147.out  slurm-19422255_49.out
125.txt  29.txt  71.txt  slurm-19422255_10.out   slurm-19422255_148.out  slurm-19422255_4.out
126.txt  2.txt   72.txt  slurm-19422255_110.out  slurm-19422255_149.out  slurm-19422255_50.out
126.txt  31.txt  74.txt  slurm-19422255_111.out  slurm-19422255_14.out   slurm-19422255_51.out
127.txt  32.txt  75.txt  slurm-19422255_112.out  slurm-19422255_150.out  slurm-19422255_52.out
12.txt   34.txt  76.txt  slurm-19422255_113.out  slurm-19422255_15.out   slurm-19422255_53.out
130.txt  35.txt  77.txt  slurm-19422255_114.out  slurm-19422255_16.out   slurm-19422255_54.out
131.txt  36.txt  78.txt  slurm-19422255_115.out  slurm-19422255_17.out   slurm-19422255_55.out
132.txt  37.txt  79.txt  slurm-19422255_116.out  slurm-19422255_18.out   slurm-19422255_56.out
133.txt  38.txt  7.txt   slurm-19422255_117.out  slurm-19422255_19.out   slurm-19422255_57.out
134.txt  39.txt  80.txt  slurm-19422255_118.out  slurm-19422255_1.out    slurm-19422255_58.out
135.txt  3.txt   81.txt  slurm-19422255_119.out  slurm-19422255_20.out   slurm-19422255_59.out
136.txt  40.txt  82.txt  slurm-19422255_11.out   slurm-19422255_21.out   slurm-19422255_5.out
137.txt  41.txt  84.txt  slurm-19422255_120.out  slurm-19422255_22.out   slurm-19422255_60.out
138.txt  42.txt  85.txt  slurm-19422255_121.out  slurm-19422255_23.out   slurm-19422255_61.out
139.txt  43.txt  86.txt  slurm-19422255_122.out  slurm-19422255_24.out   slurm-19422255_62.out
13.txt   44.txt  87.txt  slurm-19422255_123.out  slurm-19422255_25.out   slurm-19422255_63.out
```

For these unfinished tasks, we need to assign a new seed value which we set to be `job number * 20`; i.e. `seed = as.numeric(dID)*20`. We select 20 as the multiplier to avoid repeating the seed values. For example, a multiplier of 10 doesn't work because if we multiply task ID 14 by 10, we get 140 as the seed, which has already been used for pedigree ID 140 in the previous run. In this way, we obtain the 150 simulated pedigrees in separate files (i.e. 1.txt, 2.txt,. . . ,150.txt), read them all into R and save them in a single file called `study_peds.txt`. We use a `for`-loop to read in the pedigrees and save them in a list. Then we combine all 150 list elements into a single data frame as shown in the next code chunk.

```r
## Load all 150 simulated pedigrees, save them in a single list, and write them to a text file.

study_peds <- list()

for(i in 1:150){

  study_peds[[i]] <- read.table(paste0(i,".txt"))

  }

study_peds <- do.call("rbind", study_peds)

write.table(study_peds, file = "study_peds.txt")
```

3 Examine the simulated pedigrees

In the next code chunk, we read `study_peds.txt` into R as a data frame and convert it to class `ped` using the `new.ped()` function of the `SimRVPedigree` R package.

```
library(SimRVPedigree)

# import study peds
study_peds <- read.table("study_peds.txt", header=TRUE, sep= " ")

# create an object of class ped, from a data.frame,
study_peds <- new.ped(study_peds)

head(study_peds)
```

```
##   FamID ID sex dadID momID affected DA1 DA2 birthYr onsetYr deathYr available
## 1     1  1   1    NA    NA     TRUE   0   1    1881    1952    1955      TRUE
## 2     1  2   0    NA    NA    FALSE   0   0      NA      NA      NA     FALSE
## 3     1  3   1     2     1     TRUE   0   1    1901    1970    1981      TRUE
## 5     1  4   0     2     1     TRUE   0   1    1910    2000    2002      TRUE
## 6     1  5   1     2     1    FALSE   0   0    1913      NA    1991      TRUE
## 8     1  7   1     6     3    FALSE   0   1    1924      NA    1956      TRUE
##   Gen proband
## 1   1   FALSE
## 2   1   FALSE
## 3   2   FALSE
## 5   2    TRUE
## 6   2   FALSE
## 8   3   FALSE
```

The rows of `study_peds` represent individuals and the columns are:

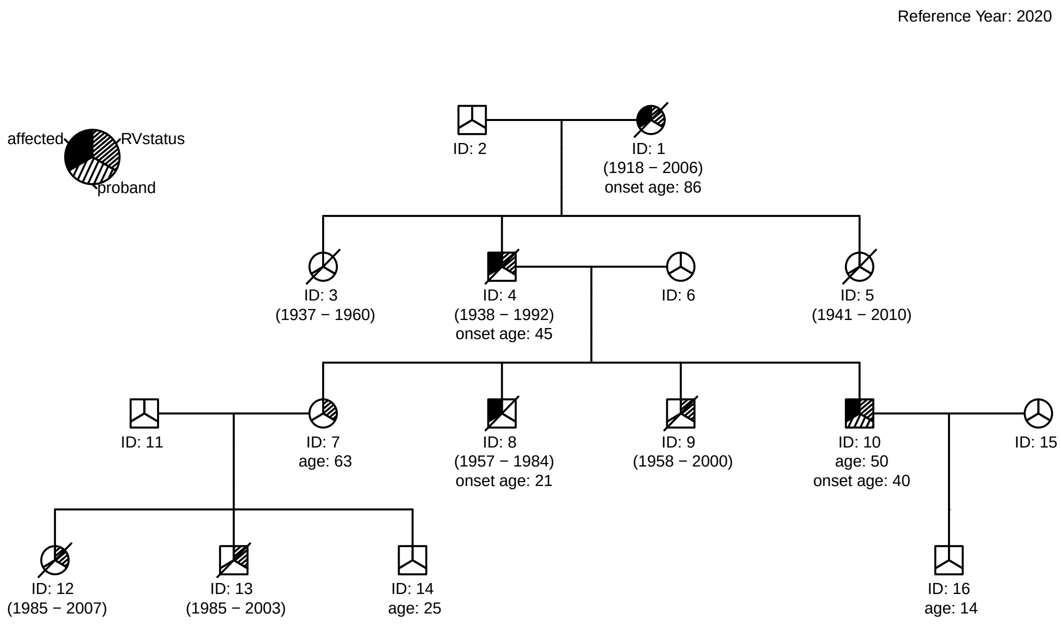
1. `FamID`- the identity number of the ascertained pedigree.
2. `ID`- the individual identity number.
3. `sex`- sex of the individual, with `sex` = 0 for males and `sex` = 1 for females.
4. `dadID`- individual identity number of the father.
5. `momID`- individual identity number of the mother.
6. `affected`- the disease status of the individual, with `affected = TRUE` if the individual has developed disease and `affected = FALSE` otherwise.
7. `DA1`- the cRV status of the paternally inherited allele, with `DA1 = 1` if the cRV is inherited and 0 otherwise.
8. `DA2`- the cRV status of the maternally inherited allele, with `DA2 = 1` if the cRV is inherited and 0 otherwise.
9. `birthYr`- the birth year of the individual.
10. `onsetYr`- the disease-onset year of the individual, when applicable, and NA otherwise.
11. `deathYr`- the death year of the individual, when applicable, and NA otherwise.
12. `RR`- the genetic relative-risk of disease for carriers of the cRV.
13. `available`- the availability of life-events information on the individual. Specifically, if an individual descends from the seed founder and is recalled by the proband then `available = TRUE`. If an individual descends from the seed founder and is not recalled by the proband then `available = FALSE`. Finally,

if an individual is not descended from the seed founder (i.e. has married into the pedigree) `available = FALSE`.

14. `Gen`- the generation of the individual within the pedigree.
15. `proband`- the proband status, with `proband = TRUE` if the individual is the proband and `FALSE` otherwise.

Let's use `SimRVPedigree`'s built-in `plot()` function to draw a pedigree in `study_peds`, in the year 2020. We will take the 39th pedigree out of the 150 generated:

```r
plot(study_peds[study_peds$FamID == 39, ], ref_year = 2020, cex = 0.5)
```



The legend identifies affected individuals, the proband, and the cRV status of the individuals. Disease-affected individuals have solid shading in the upper-left third of their symbol (IDs 1, 4, 8 and 10). The proband (ID 10) has shading in the lower portion of their symbol. Carrier individuals (IDs 1, 4, 7, 9, 10, 12 and 13) have shading in the upper-right portion of their symbol. The seed founder is the individual with ID 1 and he and all his descendants have ages relative to the reference year of 2020. We create age labels at a selected reference year by providing the argument `ref_year` to the `plot()` function. The birth year and the death year of dead individuals are displayed in parentheses. Following standard practice in medical genetics, individuals who have died as of the reference year have slashes through their symbols. The age of the individuals who are alive at the end of the reference year displays under their symbol. Any individual with disease onset before the end of the reference year has a disease-onset year given under their symbol.

For reference, session information giving the versions of R and packages used by the `SimRVPedigree` package is as follows.

```r
# Get the session information
sessionInfo()
```

```
## R version 4.2.3 (2023-03-15 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 22621)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_Canada.utf8  LC_CTYPE=English_Canada.utf8
## [3] LC_MONETARY=English_Canada.utf8 LC_NUMERIC=C
## [5] LC_TIME=English_Canada.utf8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] SimRVPedigree_0.4.4
##
## loaded via a namespace (and not attached):
##  [1] quadprog_1.5-8  lattice_0.20-45 digest_0.6.29   grid_4.2.3
##  [5] magrittr_2.0.3  evaluate_0.18   rlang_1.0.6     stringi_1.7.8
##  [9] cli_3.4.0       rstudioapi_0.14 kinship2_1.9.6  Matrix_1.5-1
## [13] rmarkdown_2.18  tools_4.2.3     stringr_1.4.1   xfun_0.35
## [17] yaml_2.3.6      fastmap_1.1.0   compiler_4.2.3  htmltools_0.5.3
## [21] knitr_1.41
```

In the third and final step of our workflow, to be discussed next, the file `study_peds.text` will be used to simulate the exome-sequencing data for the 150 ascertained pedigrees.

References

Nieuwoudt, Christina, and Jinko Graham. 2018. "SimRVPedigree: Simulate Pedigrees Ascertained for a Rare Disease. R package version 0.1.0. https://CRAN.R-project.org/package=SimRVPedigree." https://doi.org/10.1101/234153%3E.Depends.

Nieuwoudt, Christina, Samantha J. Jones, Angela Brooks-Wilson, and Jinko Graham. 2018. "Simulating pedigrees ascertained for multiple disease-affected relatives." *Source Code for Biology and Medicine*. https://doi.org/10.1186/s13029-018-0069-6.

Appendix D

Supplementary Material 3 : Simulate SNV data for affected individuals in pedigrees

Supplementary Material 3 : Simulate SNV data for affected individuals in pedigrees

Nirodha Epasinghege Dona, Jinko Graham

2022-04-22

Contents

This document discusses the gene-dropping step in our work-flow (the orange box labelled 3). Gene-dropping in the ascertained pedigrees is the third and final step required to simulate the exome-sequencing data of affected individuals and their connecting relatives along a line of descent in the ascertained pedigrees.

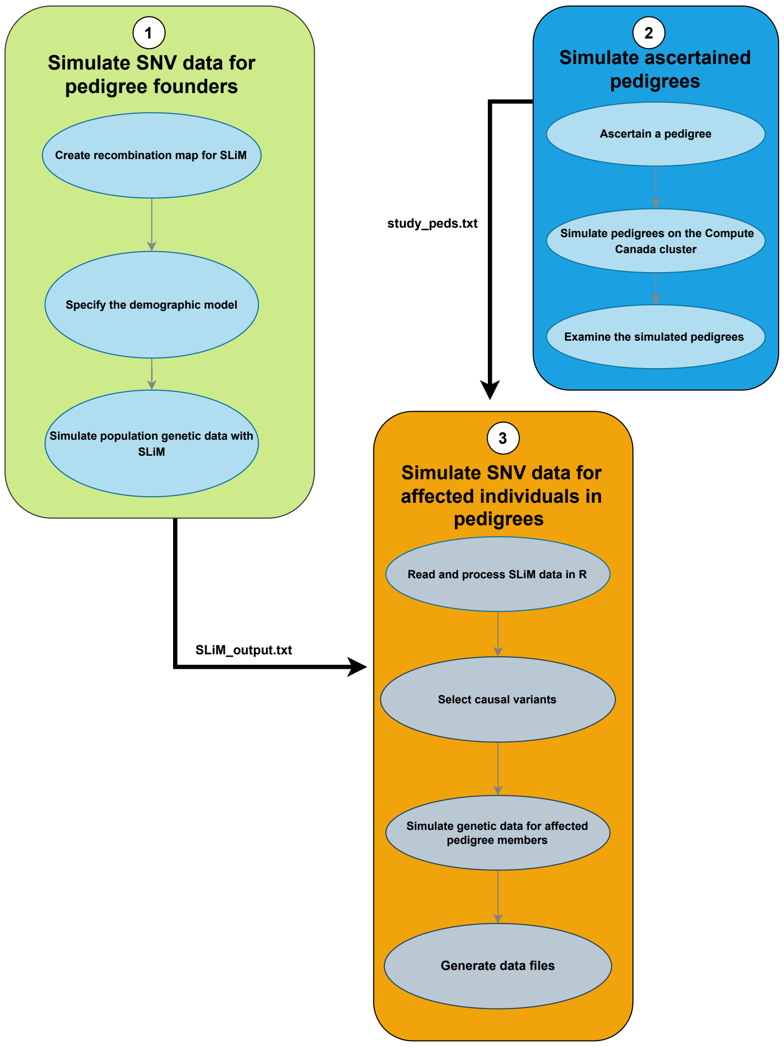

1
Simulate SNV data for pedigree founders
Create recombination map for SLiM
Specify the demographic model
Simulate population genetic data with SLiM
2
Simulate ascertained pedigrees
Ascertain a pedigree
Simulate pedigrees on the Compute Canada cluster
Examine the simulated pedigrees
study_peds.txt
SLiM_output.txt
3
Simulate SNV data for affected individuals in pedigrees
Read and process SLiM data in R
Select causal variants
Simulate genetic data for affected pedigree members
Generate data files

Figure 1: Work-flow for simulating the exome-sequencing data for ascertained pedigrees.

1 Read and process SLiM data in R

Our final goal is to simulate exome sequences for disease-affected members of the ascertained pedigrees. In the first supplementary materials document we obtained exome sequences for an American-admixed population. In the second supplementary materials document we obtained the ascertained pedigrees. Now, we have only to select sequences for the pedigree founders from the population, and then "drop" them through the pedigrees to descendants. We use the gene-dropping functions available in the `SimRVSequences` (Nieuwoudt, Brooks-Wilson, and Graham 2020) R package, which require sparse matrices of SNV sequences. Unfortunately, the large population size and number of single-nucleotide variants (SNVs) exceeds R's memory capacity for a single sparse matrix. Therefore, we read in the population sequences and create the sparse matrices chromosome-by-chromosome, as described next.

We start by reading the SLiM simulation output, `SLiM_output.txt`, into R. This text file is of size approximately 6 GB and contains all the exome sequences in the American-admixed population. The file takes approximately 1 minute to read on a Windows OS with an i7-8550U @ 1.8GHz,16GB of RAM.

```
library(Matrix) #this package is required throughout this document
# Read the text file to R.
# Note: Change the path for the file as necessary.
exData <- readLines("D:/SFU_Vault/SLiM_Output/SLiM_output.txt")
```

Next, we select rare SNVs based on their population derived (mutated) allele frequencies, as described in the next subsection.

1.1 Extract the rare variants

First, we find the line numbers of the mutation and genome header sections in the SLiM output. The formatting of the SLiM output is described in first supplementary materials document.

```
# Find heading location (i.e. file line number) for mutations.
MutHead <- which(exData == "Mutations:")

# Find heading location (i.e. file line number) for genomes.
GenHead <- which(exData == "Genomes:")
```

We create a data frame to store all the SNVs as follows.

```
# Extract mutation data from SLiM's Mutation output.
# Only retaining the tempID, position, selection coefficients and count
# of each mutation.
MutOut <- do.call(rbind, strsplit(exData[(MutHead + 1):(GenHead - 1)],
                                   split = " ", fixed = TRUE))

MutData <- data.frame(tempID = as.numeric(MutOut[, 1]),
                      type = MutOut[, 3],
                      position = as.numeric(MutOut[, 4]),
                      selCoef = as.numeric(MutOut[, 5]),
                      count = as.numeric(MutOut[, 9]),
                      stringsAsFactors = TRUE)
head(MutData)
```

```
##   tempID type position selCoef count
## 1   7229   m2 51287555       0 60626
## 2  13218   m2 39812003       0  9536
## 3   5202   m2 36490340       0 64099
## 4  25103   m2 41991968       0  3732
## 5  14264   m2 54793604       0 46668
## 6   4333   m2 30267920       0 79908
```

The `MutData` data frame contains all the SNVs in the simulated American-admixed population. The rows of this data frame correspond to SNVs and the columns to the following SNV characteristics of interest:

1. `tempID`- specifies the SLiM-generated identifier number which helps to identify the SNV.
2. `type`- represents the type of the SNV."m_1" and "m_2" catalog the non-synonymous and synonymous SNVs, respectively.

3. `position`- indicates the base-pair position of the SNV on the chromosome.
4. `selCoef`- represents the selection coefficient of the SNV.
5. `count`- specifies the number of copies of the SNV in the population.

Next we change the default behavior of the starting positions in SLiM. The starting position is zero in SLiM, but R starts its indexing at position one. To accommodate R indexing, we add one to the `tempID` and `position` columns in the `MutData` data frame.

```
# Add 1 to temp ID so that we can easily associate mutations to columns.
# By default SLiM's first tempID is 0, not 1.
MutData$tempID <- MutData$tempID + 1

# First position in slim is 0, not 1
MutData$position <- MutData$position + 1
```

Then we calculate the population derived-allele frequencies of the SNVs. We divide the number of copies of the SNV in the population (the `count` column in the `MutData`) by the total number of sequences in the population.

```
# Get the population count of sequences.
popCount <- as.numeric(unlist(strsplit(exData[1], split = " ",
                                        fixed = TRUE))[4])
# Calculate the population derived allele frequency.
# Divide the allele count by the population size.
MutData$afreq <- MutData$count/(popCount)

# Order Mutation data set by tempID, so that (later) we can order
# the mutations on each haplotype by their genomic position.
MutData <- MutData[order(MutData$tempID), ]
```

After calculating the population derived-allele frequencies, we keep the SNVs which are rare in the population. To select only the rare variants (RVs), we assign a `colID` to each based on a threshold value for its minor-allele frequency (MAF). RVs with MAFs below the threshold are assigned non-zero `colID`s that increase according to their physical order on the exome. Common SNVs are assigned a `colID` of zero, as they will be discarded.

```
# Create a threshold value
maf <- 0.01
keep_SNVs <- (MutData$afreq <= maf | MutData$afreq >= (1 - maf))

# Variants with MAF below the threshold are assigned non-zero
# colIDs according their physical order on the exome.

MutData$colID <- cumsum(keep_SNVs)*(keep_SNVs)
```

We create a new data frame, `RareMutData`, of RVs in the American-admixed population.

```
# Using the identified colID, create data frame of rare mutations only.
RareMutData <- MutData[MutData$colID > 0, ]
```

We identify the chromosome of each RV with the `reMap_mutations()` internal function in the `SimRVSequences` package. This function requires a recombination map identifying the exon positions on chromosomes. The recombination map is obtained by calling the `create_slimMap()` function in the `SimRVSequences` R package.

```
# Create recombination map for exon-only data using
# the hg_exons dataset.(From SimRVSequences.)
recomb_map <- SimRVSequences::create_slimMap(exon_df = hg_exons)
```

```r
# Use reMap_mutations function to identify the chromosome
# number on which each SNV resides.
RareMutData <- SimRVSequences:::reMap_mutations(mutationDF = RareMutData,
                                                recomb_map)
```

The call to `reMap_mutation()` adds a new column, `chr`, to `RareMutData`. We are now in a position to extract RV sequences from the SLiM output, as discussed in the next subsection.

1.2 Extract sequences of RVs

First we get the line number of the genome-header section in the SLiM output:

```r
# Find heading location (i.e. file line number) for genomes.
GenHead <- which(exData == "Genomes:")
```

In the genomes section of the SLiM output, rows and columns represent, respectively, exome sequences and SLiM-generated identifier numbers for SNVs. We extract RV sequences with the `extract_tempIDs()` internal function of the `SimRVSequences` package.

```r
# Determine future row and column position of each mutation
# listed in genomes.
RareGenomes <- lapply(1:(popCount), function(x){
    SimRVSequences:::extract_tempIDs(mutString = exData[GenHead + x],
                    rarePos = MutData$colID)
  })
```

For the American-admixed population, we now have a database of the RVs (`RareMutdata`) as well as a catalog of the sequences containing them (`RareGenomes`). These sequences will be separated by chromosome in the next subsection.

1.3 Prepare chromosome-specific population data

The code chunk below separates the large number of RVs and sequences in the American-admixed population by chromosome. As the code chunk takes approximately 16 hours to run on a Windows OS with an i7-8550U @ 1.8GHz,16GB, we recommend against running it to knit the document. Instead, load the `Chromwide.Rdata` file which can be found in the Zenodo repository.

We use the `foreach()` function to parallelize the looping over the 22 chromosomes. To start, we create a "haplotypes" matrix for the corresponding chromosome. The haplotypes matrix is a sparse matrix of class `dgCMatrix` defined in the `SimRVSequences` package. The rows of the matrix correspond to sequences in the population and the columns to RVs that lie on the targeted chromosome. For any chromosome, the number of rows in the haplotypes matrix is the number of sequences in the population. We get the IDs for RVs that lie on a particular chromosome from the `RareMutData` R object and match them to the column IDs of the haplotypes in the `RareGenome` R object. The RVs are ordered according to their base-pair position along the chromosome. The RVs in the columns of the chromosome-specific haplotypes matrix `GenoData` and in the rows of the chromosome-specific mutation data-frame `RareMutData_new` are named according to their chromosome and base-pair position. The following code chunk implements these steps.

```r
#-------------------#
# Get the results by chromosome
#-------------------#
# Load required libraries to parallel the code
library(foreach)
library(doParallel)

# Get unique chromosome IDs.
chrID <- unique(RareMutData$chrom)
```

```
# Create empty lists to save the results
chrby_haplotype  <- list()
chrby_SNVs <- list()
output <- list()

# Since we have a large number of SNVs in each chromosome,
# we parallelize the function to speed up the simulation time.
# Make the clusters.
cl <- makeCluster(detectCores() - 1)
# Register the clusters.
registerDoParallel(cl)

# Create a foreach loop.
out <- foreach(k= 1:length(chrID),
               .packages = c("Matrix", "tidyverse", "data.table",
                             "SimRVSequences"),
               .multicombine = TRUE)%dopar%{
    #-----------#
    # Genotypes #
    #-----------#
    # Get the column positions of the sparse matrix for the kth chromosome.
    # We use the jpos output that contains the
    # data from the genome section of the SLiM output.
    jpos_chr <- lapply(RareGenomes,function(x){
                     x[RareMutData[RareMutData$colID
                                   %in% x, ]$chrom == chrID[k]]})

    # Get the rows of the sparse matrix for the kth chromosome.
    ipos_chr <- lapply(1:length(jpos_chr), function(x){
                       rep(x, length(jpos_chr[[x]]))})

    # Create sparse matrix containing SNVs (columns)
    # for each genome (row).
    GenoData <- sparseMatrix(i = unlist(ipos_chr),
                             j = unlist(jpos_chr),
                             x = rep(1, length(unlist(jpos_chr))))

    GenoData <- GenoData[, -which(colSums(GenoData) == 0)]

    #-----------#
    # SNVs #
    #-----------#

    # Identify the SNV matrix for each chromosome.
    Mute_uni <- unlist(jpos_chr)
    RareSNVs <- RareMutData[RareMutData$colID %in% Mute_uni, ]

    # Order by genomic position of rare SNV.
    GenoData <- GenoData[, order(RareSNVs$position)]
    RareSNVs <- RareSNVs[order(RareSNVs$position), ]
    RareSNVs$colID <- 1:nrow(RareSNVs)

    # Remove the old tempID.
    RareSNVs <-  RareSNVs[, -1]

    # Change the row names and column names of the mutation data frame.
    RareMutData_new <- RareSNVs
    row.names(RareMutData_new) = NULL

    # Create unique SNV names.
    RareMutData_new$SNV <- make.unique(paste0(RareMutData_new$chrom,
                                              sep = "_",
                                              RareMutData_new$position))

    # Reduce RareMutData, to the columns we actually need.
    RareMutData_new <- RareMutData_new[, c("colID", "chrom", "position",
                                           "afreq", "SNV", "type",
                                           "selCoef")]
    # Store the SNVs and haplotypes by chromosome.
    output[[k]] <- list(Haplotypes = GenoData,
                        Mutations = RareMutData_new)

               }

stopCluster(cl)

# Save the result
save(out, file = "Chromwide.Rdata")
```

1.4 Identify pathway RVs

To identify the RVs that lie on the pathway of interest, we use the `identify_pathwaySNVs()` function in the `SimRVSequences` package. We supply the apoptosis sub-pathway centered about the TNFSF10 gene in the UCSC Genome Browser's Gene Interaction Tool as discussed in Nieuwoudt, Brooks-Wilson, and Graham (2020). The data in this sub-pathway are contained in the `hg_apopPath` data set in `SimRVSequences` R package.

```
# Load the output generated from the previous code chunk.
# Note: Change the path for the file as necessary.

load("Chromwide.Rdata")

#----------------------#
# Identify Pathway SNVs #
#----------------------#
pathway_out <- lapply(out, function(x){
  RareMutData_pathway = SimRVSequences:::identify_pathwaySNVs(markerDF =
                             x$Mutations, pathwayDF = hg_apopPath )})
```

The call to `identify_pathwaySNVs()` adds an additional column to the mutation data frame labelled `pathwaySNV`. This column identifies RVs that lie on the pathway as `TRUE`.

We combine the chromosome-specific haplotypes matrices with the chromosome-specific mutation data frames to get list elements for chromosomes. We then combine the chromosome-specific list elements into a list of chromosomes as follows.

```
# Create a list of 22 elements representing chromosomes.
# Each element is itself a list which contains the haplotypes
# matrix and the mutation data frame for that chromosome.

slim_out <- lapply(1:22, function(x){list(Haplotypes = out[[x]]$Haplotypes,
                                          Mutations = pathway_out[[x]])})
```

The format of `slim_out` is discussed in the next subsection.

1.5 Discuss format of chromosome-specfic population data

The structure of the first element of the `slim_out`, for chromosome 1, is shown below.

```
# Get the structure of each list elements of output.
str(slim_out[[1]])
```

```
## List of 2
##  $ Haplotypes:Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
##   .. ..@ i       : int [1:8343173] 86549 579 1814 3424 4089 4909 5912 6565 7663 8422 ...
##   .. ..@ p       : int [1:84665] 0 1 144 146 189 295 349 362 366 367 ...
##   .. ..@ Dim     : int [1:2] 107752 84664
##   .. ..@ Dimnames:List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : NULL
##   .. ..@ x       : num [1:8343173] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..@ factors : list()
##  $ Mutations :'data.frame':  84664 obs. of  8 variables:
##   ..$ colID     : int [1:84664] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ chrom     : int [1:84664] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ position  : num [1:84664] 11951 11975 12224 12626 13231 ...
##   ..$ afreq     : num [1:84664] 9.28e-06 1.33e-03 1.86e-05 3.99e-04 9.84e-04 ...
##   ..$ SNV       : chr [1:84664] "1_11951" "1_11975" "1_12224" "1_12626" ...
##   ..$ type      : Factor w/ 2 levels "m1","m2": 1 1 1 1 2 2 1 1 1 1 ...
##   ..$ selCoef   : num [1:84664] -3.64e-03 -1.33e-06 -2.45e-02 -4.53e-03 0.00 ...
##   ..$ pathwaySNV: logi [1:84664] FALSE FALSE FALSE FALSE FALSE FALSE ...
```

The object `slim_out` is a list of 22 elements. Each element corresponds to a chromosome and is itself a list with two elements, a haplotypes matrix and a mutation data frame. The haplotypes matrix contains the chromosome-specific exome sequences of all 107,752 individuals in the simulated American-admixed population. Exome sequences for pedigree founders are sampled from the haplotypes matrix. The mutation data frame contains information on the SNVs that reside on the chromosome. For chromosome 1, the dimensions of these elements are as follows.

```
# dimensions of the list element 1
dim(slim_out[[1]]$Haplotypes)
```

```
## [1] 107752  84664
```

```
dim(slim_out[[1]]$Mutations)
```

```
## [1] 84664     8
```

The number of columns in haplotypes matrix is equal to the number of rows in the mutation data frame. The first chromosome has 84,664 SNVs. Let's print the first four rows and 30 columns of its haplotypes matrix.

```
slim_out[[1]]$Haplotypes[1:6, 1:30]
```

```
## 6 x 30 sparse Matrix of class "dgCMatrix"
##
## [1,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## [2,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## [3,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## [4,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## [5,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## [6,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```

The haplotypes matrix is a sparse matrix of class `dgCMatrix` from the `Matrix` package. Rows correspond to individuals and columns correspond to RVs on chromosome 1. Entries with "1" and "." indicate the derived (mutated) allele and the ancestral allele, respectively.

Let's print the first six rows of the mutation data frame for chromosome 1.

```
head(slim_out[[1]]$Mutations)
```

```
##   colID chrom position        afreq     SNV type       selCoef pathwaySNV
## 1     1     1    11951 9.280570e-06 1_11951   m1 -3.638366e-03      FALSE
## 2     2     1    11975 1.327122e-03 1_11975   m1 -1.328507e-06      FALSE
## 3     3     1    12224 1.856114e-05 1_12224   m1 -2.454038e-02      FALSE
## 4     4     1    12626 3.990645e-04 1_12626   m1 -4.526557e-03      FALSE
## 5     5     1    13231 9.837404e-04 1_13231   m2  0.000000e+00      FALSE
## 6     6     1    13411 5.011508e-04 1_13411   m2  0.000000e+00      FALSE
```

The rows and columns of the mutation data frame represent the RVs and their characteristics, respectively. The column variable `colID` links the rows in the mutation data frame to the columns of haplotypes matrix, `chrom` is the chromosome of the RV, `position` is the position of the RV along the chromosome in base pairs, `afreq` is the RV's population derived allele frequency, `SNV` is an unique character identifier for the RV, and `pathwaySNV` identifies whether or not RVs are located within the apoptosis sub-pathway of interest.

The next task is to select the causal rare variants (cRVs).

2 Select causal variants

We create a function, `select_cRV()`, to select cRVs. The functions considers RVs in genes on an apoptosis sub-pathway as candidates for cRVs. Among these, cRVs are selected from population singletons, on the basis of their absolute selection coefficients, until the cumulative probability of a sequence carrying a cRV in the population is 0.001. Note that selection coefficients are less than or equal to zero because mutations are deleterious or selectively neutral in the SLiM simulation. The function has two required arguments:

1. `chrm_by_out` - represents the chromosome-by-chromosome results in the `slim_out` R object.
2. `cumAF` - specifies the cumulative probability of a sequence carrying a risk variant in the population.

`select_cRV()` selects the relevant mutation data frames and haplotypes matrices from the `slim_out` R object. Singleton SNVs which lie on the specified pathway (apoptosis sub-pathway) are determined from the haplotypes matrices of the `cumAF` argument. A weight is assigned to each singleton in the pathway according to the value of its selection coefficient (which is ≤ 0 because mutations are set to be deleterious or selectively neutral in the SLiM simulation). The weights are calculated as:

$$w_i = \frac{|S_i|}{|\sum_i^N S_i|},$$

where w_i is weight of i^{th} SNV; S_i is the selection coefficient of the i^{th} SNV and N is the total number of singletons in the specified pathway. These weights are used as the sampling probabilities for drawing causal rare variants (cRVs) from the pool of singleton SNVs. The cRVs are sampled randomly with these weights until their cumulative, derived-allele frequency in the population is 0.001.

The `select_cRV()` function is:

```r
select_cRV <- function(chrm_by_out, cumAF){

  # Select all the mutation data frames in the slim_out object.
  SNV_df <- lapply( chrm_by_out, `[[`, 'Mutations')

  # Select all haplotypes matrices in the slim_out object.
  haplo <- lapply( chrm_by_out, `[[`, 'Haplotypes')

  # Get the ColIDs of singletons under each chromosome.
  sing_colID <- lapply(lapply(haplo, function(x){
    which(colSums(x) == 1)}), unlist)

  # Select only singletons in the mutation data frames.
  singletons <- lapply(1:22, function(x){
    SNV_df[[x]][SNV_df[[x]]$colID %in% sing_colID[[x]], ]})

  # Combine all the 22 data frames (contains only singletons)
  # into one data frame.
  SNV_singletons <- do.call(rbind, singletons)

  # Assign weights to each marker based on their selection coefficient values.
  SNV_singletons$weight <-
    abs(SNV_singletons$selCoef)/abs(sum(SNV_singletons$selCoef))

  # Select SNVs(singletons) which lie on our pathway of interest.
  SNV_pathway <- SNV_singletons[SNV_singletons$pathwaySNV == TRUE, ]

  # Initialize vectors to store the cumulative sum of allele frequencies and
  # cRVs.
  cum <- 0
  cSNVs <- c()

  # The loop runs while the cumulative sum of the allele frequency
  # is less than or equal to CumAF (0.001).
  while (cum <= cumAF) {
    # Select a SNV proportional to its weights.
    selected_cRVs <- sample(SNV_pathway$SNV, 1,
                            prob = c(SNV_pathway$weight))
    # Get the allele frequency of the selected SNV.
    af <-  SNV_pathway[SNV_pathway$SNV == selected_cRVs, 4]
    # Update the cumulative sum of the allele frequencies of causal SNVs.
    cum <- cum + af
    # Remove the selected SNV from the mutation data frame.
    # If not we may get this same SNV again.
    SNV_pathway <- SNV_pathway[-(which(SNV_pathway$SNV == selected_cRVs)),
                               ]
    # Save the selected cRVs in a vector
    cSNVs <- c(cSNVs, selected_cRVs)
  }
  # Create a new variable in our mutation data frame to represent
  # whether a SNV is a cRV or not.
  SNV_combine <- do.call("rbind", SNV_df)
  SNV_combine$is_CRV <- SNV_combine$SNV %in% cSNVs
  return(SNV_combine)
}
```

Below is an example call to `select_cRV()`.

```
# Set a seed value.
set.seed(1987)

# Run the function.
cRV_data <- select_cRV(chrm_by_out = slim_out , cumAF = 0.001)

# Display the function output.
head(cRV_data)
```

```
##   colID chrom position        afreq     SNV type       selCoef pathwaySNV
## 1     1     1    11951 9.280570e-06 1_11951   m1 -3.638366e-03      FALSE
## 2     2     1    11975 1.327122e-03 1_11975   m1 -1.328507e-06      FALSE
## 3     3     1    12224 1.856114e-05 1_12224   m1 -2.454038e-02      FALSE
## 4     4     1    12626 3.990645e-04 1_12626   m1 -4.526557e-03      FALSE
## 5     5     1    13231 9.837404e-04 1_13231   m2  0.000000e+00      FALSE
## 6     6     1    13411 5.011508e-04 1_13411   m2  0.000000e+00      FALSE
##   is_CRV
## 1  FALSE
## 2  FALSE
## 3  FALSE
## 4  FALSE
## 5  FALSE
## 6  FALSE
```

The output of the function is a mutation data frame with an additional column, `is_CRV`. This column gives the RVs selected as causal variants. We may then print the number of cRVs in the population, their cumulative allele frequencies in the population and their chromosomes, as follows.

```
# Display the number of selected cRVs in the population.
length(which(cRV_data$is_CRV == TRUE))
```

```
## [1] 108
```

```
# Print the cumulative allele frequency in the population.
round(sum(cRV_data$afreq[which(cRV_data$is_CRV==TRUE)]), 5)
```

```
## [1] 0.001
```

```
# Display the number of cRVs that are selected from each chromosome.
table(cRV_data[cRV_data$is_CRV ==  TRUE, ]$chrom)
```

```
##
##  1  2  3  4  5  6  7  8 10 11 16 17 18 19 21 22
##  1 21  5  3  8  8  5 11  8  4  4  4 13  1  7  5
```

According to the above outputs, 108 RVs are sampled as cRVs. Their cumulative derived-allele frequency in the population is 0.001. The table summarizes the number of cRVs on each chromosome. For example, only one cRV is on chromosome 1; 21 cRVs reside in chromosome 2 and so forth.

Then we add the `is_CRV` column to all 22 mutation data frames in the `slim_out` object as follows.

```
# Add is_CRV column to all 22 mutation data frames
slim_out <- lapply(1:22, function(x){
  list(Haplotypes = slim_out[[x]]$Haplotypes,
       Mutations = cRV_data[cRV_data$chrom == x, ])})
```

We are now ready to simulate exome sequences for the affected individuals in the 150 ascertained pedigrees, as described in the next section.

3 Simulate genetic data for affected pedigree members

The `sim_RVstudy()` function of the `SimRVSequences` R package simulates genetic sequence data in pedigrees, but expects only a single population database of sequences as an argument in the form of a sparse matrix of SNV haplotypes and an associated mutation data frame. Unfortunately, we cannot use `sim_RVstudy()` without modification because the number of individuals and RVs in our American-admixed population far exceeds R's memory. The fundamental problem is that the population sequences of RVs cannot be contained in a single sparse matrix. We therefore modify `sim_RVstudy()` and various supporting functions in the `SimRVSequences` package to handle chromosome-specific databases, as described in the next subsections.

3.1 Modify `sim_RVstudy()` to simulate data by chromosome

We add a new argument, `fam_RVs`, to `sim_RVstudy()` that identifies the familial cRVs. This argument is used to check whether or not the familial cRV lies on the targeted chromosome. If the familial cRV is on the targeted chromosome, the chromosome is segregated through the pedigree with conditional gene-dropping (Nieuwoudt, Brooks-Wilson, and Graham 2020). Otherwise, the chromosome is segregated through the pedigree according to Mendelian law. Below, the updated `sim_RVstudy_new()` function has these changes as marked in the comments.

```
sim_RVstudy_new <- function(ped_files, SNV_data, fam_RVs,
                            affected_only = TRUE,
                            remove_wild = TRUE,
                            pos_in_bp = TRUE,
                            gamma_params = c(2.63, 2.63/0.5),
                            burn_in = 1000,
                            SNV_map = NULL, haplos = NULL){

  if (!(is.null(SNV_map)) | !is.null(haplos)) {
    stop("Arguments 'SNV_map' and 'haplos' have been deprecated.
         \n Instead, please supply to argument 'SNV_data' an object of class SNVdata.
         Execute help(SNVdata) for more information." )
  }

  if (!(SimRVSequences:::is.SNVdata(SNV_data))) {
    stop("Expecting SNV_data to be an object of class SNVdata")
  }

  #check to see if DA1 and DA2 are both missing, if so
  #assume fully sporadic and issue warning
  if (is.null(ped_files$DA1) & is.null(ped_files$DA2)) {
    ped_files$DA1 <- 0
    ped_files$DA2 <- 0
    warning("\n The variables DA1 and DA2 are missing from ped_files.
            \n Assuming fully sporadic ...
            \n...setting DA1 = DA2 = 0 for all pedigrees.")
  }

  #check ped_files for possible issues
```

```
SimRVSequences:::check_peds(ped_files)

#assign generation number if not included in ped_file
if(!"Gen" %in% colnames(ped_files)){
  ped_files$Gen <- unlist(lapply(unique(ped_files$FamID),
                          function(x){
                            SimRVSequences:::assign_gen(ped_files[
                              ped_files$FamID == x, ])}))
}

# save mutations and haplotypes in SNV_map and haplos objects
SNV_map = SNV_data$Mutations
haplos = SNV_data$Haplotypes

#check to see that the sample contains affected relatives when the
#affected_only setting is used
if (affected_only & all(ped_files$affected  == FALSE)) {
  stop("\n There are no disease-affected relatives in this sample of pedigrees.
       \n To simulate data for pedigrees without disease-affected
       relatives use affected_only = FALSE.")
}

#collect list of FamIDs
FamIDs <- unique(ped_files$FamID)

#check for pedigree formatting issues
for (i in FamIDs){
  SimRVSequences:::check_ped(ped_files[ped_files$FamID == i, ])
}

#Reduce to affected-only pedigrees
if (affected_only) {
  #reduce pedigrees to contain only disease-affected relative and
  #the individuals who connect them along a line of descent.
  Afams <- lapply(FamIDs, function(x){
  SimRVSequences:::affected_onlyPed(ped_file = ped_files[which(ped_files$FamID == x),])
  })

  #combine the reduced pedigrees
  ped_files <- do.call("rbind", Afams)
  pedfiles <- ped_files
  #check to see if any pedigrees were removed due to lack of
  #disease affected relatives and issue warning for removed pedigrees
  removed_peds <- setdiff(FamIDs, unique(ped_files$FamID))

  if (length(removed_peds) > 0){
    FamIDs <- unique(ped_files$FamID)
    warning("\n There are no disease-affected relatives in the pedigrees with FamID: ",
            paste0(removed_peds, collapse = ", "),
            "\n These pedigrees have been removed from ped_files.")
  }
}
```

```
  # Add is_CRV column again if it is not present
  if (is.null(SNV_map$is_CRV)) {
    SNV_map$is_CRV = FALSE
    # warning("The variable is_CRV is missing from SNV_map.",
    #         "\n ... randomly sampling one SNV to be the cRV for all pedigrees.")
  }

 # Check whether any candidates for the familial cRV lie on the chromosome.
 # This next block of code is changed from the original.
  for(k in 1:length(FamIDs)){
    if(any(SNV_map$SNV == fam_RVs[k])){
      ped_files[ped_files$FamID == k, ]$DA1 <- ped_files[ped_files$FamID ==
                                                        k, ]$DA1
      ped_files[ped_files$FamID == k, ]$DA2 <- ped_files[ped_files$FamID ==
                                                        k, ]$DA2
    } else {
      ped_files[ped_files$FamID == k, ]$DA1 <- 0
      ped_files[ped_files$FamID == k, ]$DA2 <- 0
    }
  }

  #Given the location of familial risk variants, sample familial founder
  #haplotypes from conditional haplotype distribution
   f_genos <- lapply(c(1:length(FamIDs)), function(x){
   sim_FGenos(founder_ids = ped_files$ID[which(ped_files$FamID == FamIDs[x]
                                          & is.na(ped_files$dadID))],
              RV_founder = ped_files$ID[which(ped_files$FamID == FamIDs[x]
                                         & is.na(ped_files$dadID)
                                         & (ped_files$DA1 + ped_files$DA2) != 0)],
              founder_pat_allele = ped_files$DA1[which(ped_files$FamID == FamIDs[x]
                                                & is.na(ped_files$dadID))],
              founder_mat_allele = ped_files$DA2[which(ped_files$FamID == FamIDs[x]
                                                & is.na(ped_files$dadID))],
              haplos, RV_col_loc = which(SNV_map$SNV == fam_RVs[x]),
              RV_pool_loc = SNV_map$colID[SNV_map$is_CRV])
  })

  #If desired by user, reduce the size of the data by removing
  #markers not carried by any member of the study.
  if (remove_wild) {
    reduced_dat <- SimRVSequences:::remove_allWild(f_haps = f_genos, SNV_map)
    f_genos <- reduced_dat[[1]]
    SNV_map <- reduced_dat[[2]]
  }

  #create chrom_map, this is used to determine the segments over
  #which we will simulate genetic recombination
  chrom_map <- SimRVSequences:::create_chrom_map(SNV_map)

  #convert from base pairs to centiMorgan
  if (pos_in_bp) {
    options(digits = 9)
    chrom_map$start_pos <- SimRVSequences:::convert_BP_to_cM(chrom_map$start_pos)
    chrom_map$end_pos <- SimRVSequences:::convert_BP_to_cM(chrom_map$end_pos)
    SNV_map$position <- SimRVSequences:::convert_BP_to_cM(SNV_map$position)
  }

  #simulate non-founder haploypes via conditional gene drop
  ped_seqs <- lapply(c(1:length(FamIDs)), function(x){
    sim_seq(ped_file = ped_files[ped_files$FamID == FamIDs[x], ],
            founder_genos = f_genos[[x]],
            SNV_map, chrom_map,
            RV_marker = fam_RVs[x],
            burn_in, gamma_params)
  })

  ped_haplos <- do.call("rbind", lapply(ped_seqs, function(x){x$ped_genos}))
  haplo_map <- do.call("rbind", lapply(ped_seqs, function(x){x$geno_map}))

  #convert back to base pairs if we converted to CM
  if (pos_in_bp) {
    options(digits = 9)
    SNV_map$position <- SimRVSequences:::convert_CM_to_BP(SNV_map$position)
  }

  return(SimRVSequences:::famStudy(list(ped_files = pedfiles, ped_haplos = ped_haplos,
                       haplo_map = haplo_map, SNV_map = SNV_map)))
}
```

`sim_RVstudy()` requires the argument `SNV_data`, an object of class `SNVdata` as defined by `SimRVSequences` package. The `SNVdata()` function in the `SimRVSequences` R package converts haplotypes matrices and mutation data frames into an object of class `SNVdata`. We modified the `SNVdata()` and the `check_SNV_map()` functions of the package to align with our changes in `sim_RVstudy_new()`. The `SNVdata()` and `check_SNV_map()` functions remain the same except that the mutation data frame provided as an argument is now expected to have a column named `SNV` rather than `marker`. When the targeted chromosome contains no cRV, the original `check_SNV_map()` function exits with an error. The original function exits because it inappropriately checks whether the `is_CRV` column is FALSE for all the SNVs in the mutation data frame. To avoid the inappropriate exit, we remove this check. The modified versions of `SNVdata()` and `check_SNV_map()` are renamed as `SNVdata_new()` and `check_SNV_map_new()` and defined in the next code chunk.

```
# Define the SNVdata_new() and check_SNV_map_new() functions

# Constructor function for an object of class SNVdata
SNVdata_new <- function(Haplotypes, Mutations, Samples = NULL) {

  #check SNV_map for possible issues
  check_SNV_map_new(Mutations)

  if (!"SNV" %in% colnames(Mutations)) {
    Mutations$SNV <- make.unique(paste0(Mutations$chrom, sep = "_", Mutations$position))
  }

  if (nrow(Mutations) != ncol(Haplotypes)) {
    stop("\n nrow(Mutations) != ncol(Haplotypes).
      \n Mutations must catalog every SNV in Haplotypes.")
  }
  #create list containing all relevant of SNVdata information
  SNV_data = list(Haplotypes = Haplotypes,
                  Mutations = Mutations,
                  Samples = Samples)

  class(SNV_data) <- c("SNVdata", class(SNV_data))
  return(SNV_data)
}

# Check SNV_map for possible issues: modified version
check_SNV_map_new <- function(SNV_map){
  #check to see if SNV_map contains the column information we expect
  # and check to see if we have any missing values.

  ## Check colID variable
  if (!"colID" %in% colnames(SNV_map)) {
    stop('The variable "colID" is missing from SNV_map.')
  }
  if (any(is.na(SNV_map$colID))) {
    stop('Error SNV_map: The variable "colID" contains missing values.')
  }
  if (any(duplicated(SNV_map$colID))) {
    stop('Error SNV_map: The variable "colID" contains duplicate values.')
  }

  ## Check chrom variable
  if (!"chrom" %in% colnames(SNV_map)) {
    stop('The variable "chrom" is missing from SNV_map.')
  }
  if (any(is.na(SNV_map$chrom))) {
    stop('Error SNV_map: The variable "chrom" contains missing values.')
  }

  ## Check position variable
  if (!"position" %in% colnames(SNV_map)) {
    stop('The variable "position" is missing from SNV_map.')
  }

  if (any(is.na(SNV_map$position))) {
    stop('Error SNV_map: The variable "position" contains missing values.')
  }

  # Check to see if marker variable exists, and if so do all SNVs have a unique name
  if ("SNV" %in% colnames(SNV_map)) {
    if (length(unique(SNV_map$SNV)) != nrow(SNV_map)) {
      stop('Expecting each SNV to have a unique SNV name in SNV_map.')
    }
    if (any(is.na(SNV_map$SNV))) {
      stop('Error SNV_map: The variable "marker" contains missing values.')
    }
  }
}
```

The next subsection discusses how we set the arguments of `sim_RVstudy_new()`.

3.2 Set arguments to `sim_RVstudy_new()`

`sim_RVstudy_new()` requires three arguments: `fam_RVs` giving the cRV for each ascertained pedigree, `ped_files` giving the ascertained pedigrees and `SNV_data` giving the chromosome-specific exome sequences and associated mutation data frames for everyone in the American admixed population. We prepare these three arguments as follows.

(1). `fam_RVs`

Familial cRVs are sampled on the basis of their population derived-allele frequencies as follows.

```r
# Load all 150 pedigrees.
# Note: Change the path for the file as necessary.
study_peds <- read.table("study_peds.txt", header=TRUE, sep= " ")

# Collect list of FamIDs.
FamIDs <- unique(study_peds$FamID)

# Set the sampling probabilities for causal RVs.
# When the derived-allele frequencies are provided, we sample cRVs
# according to their derived-allele frequency.
sample_prob <- cRV_data$afreq[cRV_data$is_CRV]/
    sum(cRV_data$afreq[cRV_data$is_CRV])

set.seed(1987)
# Sample the familial cRV from the pool of potential cRVs with replacement.
familial_RVs <- sample(x = cRV_data$SNV[cRV_data$is_CRV],
                  size = length(FamIDs),
                  prob = sample_prob,
                  replace = TRUE)
# Display first five candidates for familial cRVs.
familial_RVs[1:5]
```

```
## [1] "2_201170321" "18_63125128" "10_89015222" "6_108683999" "1_155738619"
```

The final output, `familial_RVs` is a vector of length 150 that contains the familial cRVs for each of the ascertained pedigrees.

(2). `ped_files` is a data frame that represents the ascertained pedigrees. We have loaded this data frame previously, in the object `study_peds`.

(3). `SNV_data` gives a database of exome sequences for a single chromosome, for everyone in the American-admixed population. This argument is an object of class `SNVdata`. Objects of class `SNVdata` are comprised of a sparse matrix of exome sequences together with an associated data frame of mutation information.

We first make a list, by chromosome, of objects of class `SNVdata` by applying the `SNVdata_new()` function:

```r
# Apply the SNVdata_new function to 22 SNVdata objects comprised
# of sparse matrices of SNV sequences and mutation data frames.
chrom_data <- lapply(slim_out, function(x){
  SNV_data = SNVdata_new(Haplotypes = x$Haplotypes,
                    Mutations = x$Mutations)})
```

The `SNV_data` argument of `sim_RVstudy_new()` will be extracted from the appropriate list element of the `chrom_data` object.

The remaining arguments to `sim_RVstudy_new()` are optional and set with the defaults of the original `sim_RVstudy()` function. For example, we use the default value `affected_only = TRUE` to simulate sequence data for disease-affected members in the pedigree, as is typical for exome-sequencing studies of families ascertained for multiple affected relatives. Affected relatives from such families are more likely to carry a cRV. We also set the default value of `remove_wild = TRUE`, to shrink the sequence data for the study by removing monomorphic SNVs.

To obtain the genetic sequences for disease-affected family members, we loop over chromosomes and apply the `simRV_study_new()` function to each. First, however, we load functions required by `simRV_study_new()`. These functions are slightly modified versions of `sim_FGenos()` and `sim_seq()`, two non-exported functions from the `SimRVSequences` R package. The modified versions are the same as their counterparts in `SimRVsequences`, except they use the `Matrix` package's `which()` function instead of base R's.

```r
# Draw founder genotypes from haplotype distribution given familial RV
sim_FGenos <- function(founder_ids, RV_founder,
                       founder_pat_allele, founder_mat_allele,
                       haplos, RV_col_loc, RV_pool_loc) {

  #Determine which haplotypes carry the familial RV and which do not
  #Determine which haplotypes carry the familial cRV
  RV_hap_loc <- which(haplos[, RV_col_loc] == 1)

  #Determine which haplotypes do not carry ANY cRV in the pool
  no_CRVrows <- SimRVSequences:::find_no_cSNV_rows(haplos, RV_pool_loc)

  #here we handle the fully sporadic families
  #i.e. families that do not segregate any cSNVs
  #In this case, the haplotypes for ALL founders
  #is sampled from no_CRVhaps
  if(length(RV_founder) == 0){
    #sample all founder data from this pool
    founder_genos <- haplos[sample(x = no_CRVrows,
                                   size = 2*length(founder_ids),
                                   replace = TRUE), ]
  } else {
    #sample the paternally inherited founder haplotypes
    pat_inherited_haps <- sapply(founder_pat_allele, function(x){
      if(x == 0){
        SimRVSequences:::resample(x = no_CRVrows, size = 1)
      } else {
        SimRVSequences:::resample(x = RV_hap_loc, size = 1)
      }})

    #sample the maternally inherited founder haplotypes
    mat_inherited_haps <- sapply(founder_mat_allele, function(x){
      if(x == 0){
        SimRVSequences:::resample(x = no_CRVrows, size = 1)
      } else {
        SimRVSequences:::resample(x = RV_hap_loc, size = 1)
      }})

    #pull the sampled haplotypes from the haplos matrix
    founder_genos <- haplos[c(pat_inherited_haps, mat_inherited_haps), ]
  }

  #create IDs to associate founders to rows in founder_genos
  founder_genos_ID <- rep(founder_ids, 2)

  #re-order so that founder haplotypes appear in order
  founder_genos <- founder_genos[order(founder_genos_ID), ]
  founder_genos_ID <- founder_genos_ID[order(founder_genos_ID)]

  return(list(founder_genos, founder_genos_ID))
}

#Now the modified version of the sim_seq() function.
sim_seq <- function(ped_file, founder_genos,
                    SNV_map, chrom_map, RV_marker,
                    burn_in = 1000, gamma_params = c(2.63, 2.63/0.5)){

  #Get parent/offspring information
  #i.e. for each offspring find RV_status,
  #parent IDs, and parent alleles at RV locus
  PO_info <- SimRVSequences:::get_parOffInfo(ped_file)
  PO_info <- PO_info[order(PO_info$Gen, PO_info$offspring_ID),]

  ped_genos <- founder_genos[[1]]
  ped_geno_IDs <- founder_genos[[2]]

  #determine the chromosome number and location of the familial RV locus
  #then store as a data frame with chrom in the first column
  RVL <- SNV_map[which(SNV_map$SNV == RV_marker),
                 which(colnames(SNV_map) %in% c("chrom", "position"))]

  if(colnames(RVL[1]) != "chrom"){
    RVL <- RVL[, c(2, 1)]
  }

  #for each offspring simulate transmission of parental data
  for (i in 1:nrow(PO_info)) {
    #simulate recombination events for this parent offspring pair
    loop_gams <- SimRVSequences:::sim_gameteInheritance(RV_locus = RVL,
                                      parent_RValleles = PO_info[i, c(6, 7)],
                                      offspring_RVstatus = PO_info[i, 5],
                                      chrom_map,
                                      allele_IDs = c(1, 2),
                                      burn_in, gamma_params)

    #construct offspring's inherited material from this parent
    loop_seq <- lapply(c(1:nrow(chrom_map)),
                       function(x){
                         SimRVSequences:::reconstruct_fromHaplotype(
                           parental_genotypes = ped_genos[which(ped_geno_IDs == PO_info[i,
                           4]),
                                                        which(SNV_map$chrom == chrom_map
                           $chrom[x])], CSNV_map = SNV_map[which(SNV_map$chrom == chrom_map
                           $chrom[x]),], inherited_haplotype = loop_gams$haplotypes[[x]],
                           chiasmata_locations = loop_gams$cross_locations[[x]],
                           REDchrom_map = chrom_map[x, ])
                       })
    #append ID for this haplotype to the list of IDs
    ped_geno_IDs <- c(ped_geno_IDs, PO_info[i, 1])
    ped_genos <- rbind(ped_genos, unlist(loop_seq))
  }

  #Determine if this is a sporadic pedigree
  printed_FamRV <- ifelse(all(ped_file[, c("DA1", "DA2")] == 0), "no_CRV", RV_marker)

  #create a data.frame to store identifying info
  geno_map <- data.frame(FamID = rep(ped_file$FamID[1], length(ped_geno_IDs)),
                         ID = ped_geno_IDs,
                         affected =  rep(FALSE, length(ped_geno_IDs)),
                         FamCRV = rep(printed_FamRV, length(ped_geno_IDs)),
                         stringsAsFactors = FALSE)

  #identify affected individuals
  geno_map$affected[geno_map$ID %in% ped_file$ID[ped_file$affected]] <- TRUE

  #Return the genomes matrix and a data.frame containing identifying
  #information for the of IDs to identify the
  #family member to whom
  return(list(ped_genos = ped_genos, geno_map = geno_map))
}
```

We are now ready to call `sim_RVstudy_new()` on each chromosome.

```
# Simulate exome sequences of SNVs for affected family members
set.seed(1987)

study_seq <- lapply(1:22, function(x){sim_RVstudy_new(fam_RVs = familial_RVs,
                                                      ped_files = study_peds,
                                                      SNV_data = chrom_data[[x]]
                                                      )})
```

Simulating exome-wide sequences for disease-affected members in the study families takes about 6 minutes on a Windows OS with an i7-8550U @ 1.8GHz,16GB of RAM. The times to simulate chromosomes 1, 2, 8 and 9 are shown in Table 1.

Table 1: Simulation time for selected chromosomes.

Chromosome	**No. of RVs**	**No. of cRVs**	**Time (s)**
1	84664	1	36.23
2	60995	21	53.63
8	31396	11	22.35
9	34248	0	19.57

From the table, we see that chromosome 1 takes less time to simulate than chromosome 2, despite having more rare variants. We attribute this to chromosome 1 having fewer cRVs than chromosome 2. By contrast, chromosome 8 has more cRVs than chromosome 1 yet takes less time to simulate because it has fewer RVs overall. Simulation time therefore depends on both the overall number of RVs and the number of cRVs on chromosome.

The `sim_RVstudy_new()` function returns the same set of outputs as the `sim_RVstudy()` function, as discussed in the next subsection.

3.3 Discuss the `sim_RVstudy_new()` output

The output `study_seq` from the call to `sim_RVstudy_new()` is a list containing 22 elements, one for each chromosome. As the output format of each chromosome is the same, we focus on the first chromosome. Each element of the list `study_seq` is itself a list containing four elements as follows.

(1). The `ped_files` data frame gives details about the individuals in the pedigrees. When we set `affected_only = TRUE`, the results contain only the affected individuals and the individuals who connect them along a line of descent within a pedigree. Note that the `ped_files` data frame is exactly the same for all 22 chromosomes; though wasteful of space, this unnecessary repetition is convenient for looping.

```
# View the first 4 individuals in the ped_files data frame (the same regardless of chromosome).
head(study_seq[[1]]$ped_files, n = 4)
```

```
##    FamID ID sex dadID momID affected DA1 DA2 birthYr onsetYr deathYr available
## 1      1  1   1    NA    NA     TRUE   0   1    1881    1952    1955      TRUE
## 3      1  3   1     2     1     TRUE   0   1    1901    1970    1981      TRUE
## 5      1  4   0     2     1     TRUE   0   1    1910    2000    2002      TRUE
## 25     1 20   0    19     8     TRUE   0   1    1957    1997    2016      TRUE
##    Gen proband
## 1    1   FALSE
## 3    2   FALSE
## 5    2    TRUE
## 25   4   FALSE
```

(2). The sparse matrix `ped_haplos` contains simulated SNVs on the exome sequences of the disease-affected individuals and the individuals connecting them in the ascertained pedigrees.

```
# View the first 30 SNVs of the first 6 exome sequences on the first chromosome.
study_seq[[1]]$ped_haplos[1:6, 1:30]
```

```
## 6 x 30 sparse Matrix of class "dgCMatrix"
##
## [1,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## [2,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## [3,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## [4,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## [5,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## [6,] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```

Rows of this sparse matrix correspond to exomes and columns to RVs on the first chromosome. The entry "1" represents the derived (mutated) allele and "." the ancestral allele.

(3). The `SNV_map` data frame contains information about RVs in the study. Since the `remove_wild` argument of `sim_RVstudy_new()` is set to its default value of `TRUE`, this data frame contains only RVs carried by at least one study individual.

```
# View the first 4 rows of SNV_map
head(study_seq[[1]]$SNV_map, n = 4)
```

```
##   colID chrom position          afreq     SNV type        selCoef pathwaySNV
## 1     1     1    12626 0.000399064519 1_12626   m1 -0.00452655694      FALSE
## 2     2     1    13411 0.000501150791 1_13411   m2  0.00000000000      FALSE
## 3     3     1    14230 0.001568416364 1_14230   m2  0.00000000000      FALSE
## 4     4     1    14231 0.000631078773 1_14231   m1 -0.00146789732      FALSE
##   is_CRV
## 1  FALSE
## 2  FALSE
## 3  FALSE
## 4  FALSE
```

The rows of the data frame represent the RVs carried by at least one individual in the study. The columns are characteristics of the RVs explained in subsection 1.5 of this document.

(4). The `haplo_map` data frame maps the exome sequences in `ped_haplos` to the individuals in `ped_files`. The rows of `haplo_map` correspond to sequences and the columns to characteristics of individuals to which these sequences belong. Let's look at the first family's information on chromosome 1.

```
# View family 1's entries of haplo_map
fam1 <- (study_seq[[1]]$haplo_map[,"FamID"]==1)
study_seq[[1]]$haplo_map[fam1,]
```

```
##    FamID ID affected FamCRV
## 1      1  1     TRUE no_CRV
## 2      1  1     TRUE no_CRV
## 3      1  2    FALSE no_CRV
## 4      1  2    FALSE no_CRV
## 5      1  6    FALSE no_CRV
## 6      1  6    FALSE no_CRV
## 7      1 19    FALSE no_CRV
## 8      1 19    FALSE no_CRV
## 9      1  3     TRUE no_CRV
## 10     1  3     TRUE no_CRV
## 11     1  4     TRUE no_CRV
## 12     1  4     TRUE no_CRV
## 13     1  8    FALSE no_CRV
## 14     1  8    FALSE no_CRV
## 15     1 20     TRUE no_CRV
## 16     1 20     TRUE no_CRV
```

We can see that the two sequences of an individual are stored in consecutive rows of the data frame. The `FamCRV` column of the data frame gives the identifier of the familial cRV and is the same for all family members. If a family does not have a cRV on the selected chromosome, the entry of `FamCRV` is `no_CRV`. For example, family ID 1 does not carry a cRV on chromosome 1.

With the complete data now available in the list `study_seq`, our final task is to deliver it in human-readable flat-file formats, as described next.

4 Generate data files

Throughout this section, we will refer to the list `study_seq` generated in the previous subsection. The list element for chromosome 21 has the following structure.

```
# The study_seq object is a list of length 22 elements.
# We print the 21st element of study_seq, for chromosome 21.
str(study_seq[[1]])
```

```
## List of 4
##  $ ped_files :'data.frame':  1247 obs. of  14 variables:
##   ..$ FamID    : int [1:1247] 1 1 1 1 1 1 1 1 2 2 ...
##   ..$ ID       : int [1:1247] 1 3 4 20 2 8 19 6 1 3 ...
##   ..$ sex      : int [1:1247] 1 1 0 0 0 1 0 0 1 0 ...
##   ..$ dadID    : int [1:1247] NA 2 2 19 NA 6 NA NA NA 2 ...
##   ..$ momID    : int [1:1247] NA 1 1 8 NA 3 NA NA NA 1 ...
##   ..$ affected : logi [1:1247] TRUE TRUE TRUE TRUE FALSE FALSE ...
##   ..$ DA1      : int [1:1247] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ DA2      : int [1:1247] 1 1 1 1 0 1 0 0 1 1 ...
##   ..$ birthYr  : int [1:1247] 1881 1901 1910 1957 NA 1924 NA NA 1914 1931 ...
##   ..$ onsetYr  : int [1:1247] 1952 1970 2000 1997 NA NA NA NA 1987 1979 ...
##   ..$ deathYr  : int [1:1247] 1955 1981 2002 2016 NA 1957 NA NA 1990 2014 ...
##   ..$ available: logi [1:1247] TRUE TRUE TRUE TRUE FALSE TRUE ...
##   ..$ Gen      : int [1:1247] 1 2 2 4 1 3 3 2 1 2 ...
##   ..$ proband  : logi [1:1247] FALSE FALSE TRUE FALSE FALSE FALSE ...
##  $ ped_haplos:Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
##   .. ..@ i       : int [1:193252] 1689 1700 1706 983 986 990 918 924 2246 2256 ...
##   .. ..@ p       : int [1:19394] 0 3 6 10 14 15 19 25 29 38 ...
##   .. ..@ Dim     : int [1:2] 2494 19393
##   .. ..@ Dimnames:List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : NULL
##   .. ..@ x       : num [1:193252] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..@ factors : list()
##  $ haplo_map :'data.frame':  2494 obs. of  4 variables:
##   ..$ FamID   : int [1:2494] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ ID      : int [1:2494] 1 1 2 2 6 6 19 19 3 3 ...
##   ..$ affected: logi [1:2494] TRUE TRUE FALSE FALSE FALSE FALSE ...
##   ..$ FamCRV  : chr [1:2494] "no_CRV" "no_CRV" "no_CRV" "no_CRV" ...
##  $ SNV_map   :'data.frame':  19393 obs. of  9 variables:
##   ..$ colID     : int [1:19393] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ chrom     : int [1:19393] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ position  : num [1:19393] 12626 13411 14230 14231 15836 ...
##   ..$ afreq     : num [1:19393] 0.000399 0.000501 0.001568 0.000631 0.000343 ...
##   ..$ SNV       : chr [1:19393] "1_12626" "1_13411" "1_14230" "1_14231" ...
##   ..$ type      : Factor w/ 2 levels "m1","m2": 1 2 2 1 2 1 1 2 2 2 ...
##   ..$ selCoef   : num [1:19393] -0.00453 0 0 -0.00147 0 ...
##   ..$ pathwaySNV: logi [1:19393] FALSE FALSE FALSE FALSE FALSE FALSE ...
##   ..$ is_CRV    : logi [1:19393] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  - attr(*, "class")= chr [1:2] "famStudy" "list"
```

We use `study_seq` to create a `.sam` file containing information about genotyped individuals in the ascertained pedigrees, chromosome-specific `.geno` files containing RV genotypes and chromosome-specific `.var` files containing information about RVs. As described next, the data files are in flat-file format similar to PLINK files (Purcell et al. 2007).

4.1 .sam file

The `.sam` file contains pedigree information about the disease-affected individuals and the individuals connecting them along a line of descent in their pedigrees. These individuals are prioritized for exome sequencing in our family study. The function `plink_format_samp()` generates the `.sam` file using the argument, `peds`. The argument `peds` is a data frame giving information on the study pedigrees. The function selects specific columns of the `peds` data frame and aligns them in a format similar to the `.psam` PLINK file.

```
# Get the information for .sam file
plink_format_samp <- function(peds){
  # Convert sex. In PLINK 1 is male 2 is female.
  # We have 0 s to represent male and 1 for female.
  peds$sex[peds$sex == 1] <- c(2)
  peds$sex[peds$sex == 0] <- c(1)

  # Affected variable consists logical values.
  # Need to change it as character to assign values to
```

```r
  # represent the phenotype.
  peds$affected <- as.character(peds$affected)

  # If affected is NA consider it as missing.
  # In PLINK missing is denoted as 0 or -9.
  peds$affected[is.na(peds$affected)] <- c(0)
  # Non-affected is represented as 1 in PLINK.
  peds$affected[peds$affected == "FALSE"] <- c(1)
  # Affected is represented as 2 in PLINK.
  peds$affected[peds$affected == "TRUE"] <- c(2)

  peds$FamID <- as.numeric(peds$FamID)
  peds$affected <- as.numeric(peds$affected)

  # Create the data frame with required columns.
  psam_file <- data.frame(peds$FamID, peds$ID, peds$dadID, peds$momID,
                        peds$sex, peds$affected,
                        peds$birthYr, peds$deathYr, peds$proband)

  colnames(psam_file) <- c("FID", "IID", "PAT", "MAT", "SEX", "PHENO1",
                         "BIRTHYr", "DEATHYr", "PROBAND")
  return(psam_file)
}
```

To get the `.sam` file, we apply the `plink_format_samp()` function to chromosome 21. Note that the `.sam` file is the same regardless of which chromosome is used.

```r
# Call the function for chromosome 21
sample_data <- plink_format_samp(study_seq[[21]]$ped_files)

# How many individuals.
nrow(sample_data)
```

```
## [1] 1247
```

```r
# Print the first 6 individuals
head(sample_data, n= 6)
```

```
##   FID IID PAT MAT SEX PHENO1 BIRTHYr DEATHYr PROBAND
## 1   1   1  NA  NA   2      2    1881    1955   FALSE
## 2   1   3   2   1   2      2    1901    1981   FALSE
## 3   1   4   2   1   1      2    1910    2002    TRUE
## 4   1  20  19   8   1      2    1957    2016   FALSE
## 5   1   2  NA  NA   1      1      NA      NA   FALSE
## 6   1   8   6   3   2      1    1924    1957   FALSE
```

The rows of `sample_data` are the 1247 genotyped individuals in the study pedigrees. The individuals are either disease-affected or connect disease-affected individuals along a line of descent in a study pedigree. The columns of `sample_data` contain information about the individuals as follows:

1. `FID`- the identification number of the family that the individual belongs to.
2. `IID`- the individual identification number.
3. `PAT`- the father's identification number.
4. `MAT`- the mother's identification number.

5. SEX- the individual's sex, with 1 and 2 corresponding to male and female, respectively.
6. PHENO1- the disease-affected status, with 1 and 2 corresponding to unaffected and affected, respectively.
7. BIRTHYr- the individual's birth year.
8. DEATHYr- the death year of the individual, with NA indicating that the individual is still alive at the end of the study.
9. PROBAND- a logical value indicating whether or not the individual is the proband for their pedigree.

We save the .sam file as a text file, sample_info.txt, as follows. The text file can be found in our Zenodo repository.

```
# Write the sample information to a single text file
write.table(sample_data, "sample_info.txt", row.names=FALSE, quote = FALSE)
```

4.2 .geno files

A .geno file gives the RV genotypes in gene-dosage format. An individual's dosage of the derived allele is the number of copies they inherited from their parents (i.e. 0, 1 or 2). The get_geno_data() function below converts RV-haplotype pairs into genotypes in gene-dosage format.

```
# Convert haplotype pairs into genotypes in gene-dosage format.
get_geno_data <- function(haps){
  gene_dosage <- list()
  IDs <- seq(from = 1, to = nrow(haps), by = 2)

  # Get the column sums.
  for(i in 1: length(IDs)){
    gene_dosage[[i]] <- colSums(haps[IDs[i]:(IDs[i] + 1), ])
    genotypes <- do.call(rbind, gene_dosage)
  }
  genotypes <- do.call(rbind, gene_dosage)
  return(genotypes)
}
```

Let's call get_geno_data() on chromosome 21 as an example. The function's argument, haps, is filled with the sparse matrix ped_haplos from the study_seq output.

```
# Apply the function to 21st chromosome
genotype_data <-  get_geno_data(study_seq[[21]]$ped_haplos)
```

To convert chromosome 21 haplotypes to individual genotypes in gene-dosage format, get_geno_data() takes approximately 15 seconds on a Windows OS with an i7-8550U @ 1.8GHz,16GB of RAM. Let's view the first few rows and columns of the data frame that is returned.

```
# View the first four rows and 12 columns
genotype_data[1:4, 1:12]
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
## [1,]    0    0    0    0    0    0    0    0    0     0     0     0
## [2,]    0    0    0    0    0    0    0    0    0     0     0     0
## [3,]    0    0    0    0    0    0    0    0    0     0     0     0
## [4,]    0    0    0    0    0    0    0    0    0     0     0     0
```

The rows of the data frame represent the 1247 genotyped individuals in our study. The columns represent RVs that reside on the exome of chromosome 21. Each entry of the data frame gives the dosage of the derived allele of an RV (i.e. 0, 1 or 2). Most of the entries are 0, as would be expected for RVs.

The `get_geno_data()` function is applied to all the chromosomes as follows.

```r
# Apply function to all chromosomes
genotype_data <-  lapply(study_seq, function(x){
  result <- get_geno_data(x$ped_haplos)
  colnames(result) <- x$SNV_map$SNV
  result
})
```

Below, the resulting chromosome-specific `.geno` files are written to text files named `genotypes_chr_i.txt`, where "i" indicates the chromosome number. These text files can be found in the Zenodo repository.

```r
# Write the results to 22 text files
for(i in 1:22){
  write.table(genotype_data[[i]],
              paste0("genotypes_chr_",i,".txt"),
              row.names=FALSE, quote = FALSE)
}
```

4.3 .var files

A `.var` file contains information about the RVs in the columns of the associated `.geno` file. The `get_variant_data()` function below selects the relevant characteristics of the RVs and stores them in a data frame.

```r
# Get the variant information to create the .var file
get_variant_data <- function(variant){
  # Chromosome number.
  CHROM <- variant$chrom
  # Position.
  POS <- variant$position
  # Reference allele.
  REF <- rep("A", length(CHROM))
  # Alternate allele.
  ALT <- rep("T", length(CHROM))
  # Selection coefficient.
  sel_coef <- variant$selCoef
  # Population allele frequency.
  pop_afreq <- variant$afreq
  # Pathway SNV or not.
  pathwaySNV <- variant$pathwaySNV
  # Causal SNV or not.
  C_SNV <- variant$is_CRV
  # label the type as NS and S where NS- non-synonymous and
  # S- Synonymous.
  levels(variant$type) <- c("NS", "S")
  # Type of the SNV
  Type <- variant$type

  # Create the data frame.
  SNV <- data.frame(CHROM, POS, REF, ALT,
                    pop_afreq, sel_coef, pathwaySNV, C_SNV, Type)

  return(SNV)
}
```

Let's call `get_variant_data()` on chromosome 21 as an example. The function's argument, `variant`, is filled with the `SNV_map` data frame from the `study_seq` output.

```
# Run the function on chromosome 21 SNV_map data
variant_info <- get_variant_data(study_seq[[21]]$SNV_map)
```

The function returns the data frame `variant_info`. Let's view information about the first four RVs in `variant_info`.

```
# View the first 4 rows of the resulting data frame
head(variant_info, n = 4)
```

```
##   CHROM     POS REF ALT      pop_afreq        sel_coef pathwaySNV C_SNV Type
## 1    21 5591572   A   T 0.000371222808  0.00000000e+00      FALSE FALSE    S
## 2    21 5591931   A   T 0.000575395352 -8.08714731e-08      FALSE FALSE   NS
## 3    21 6066994   A   T 0.000092805702  0.00000000e+00      FALSE FALSE    S
## 4    21 6112121   A   T 0.000566114782  0.00000000e+00      FALSE FALSE    S
```

The rows of `variant_info` contain exomic RVs on chromosome 21 that are carried by at least one study participant. The columns give the following information about these RVs:

1. `CHROM`- the chromosome number of the RV.
2. `POS`- the RV position, in base pairs, on the chromosome.
3. `REF`- the reference allele for the RV
4. `ALT`- the alternate allele for the RV
5. `pop_afreq`- the population alternate allele frequency for the RV.
6. `sel_coef`- the selection coefficient for the RV.
7. `pathwaySNV`- whether or not the RV comes from a gene in our disease pathway.
8. `C_SNV`- whether or not the RV is causal.
9. `Type`- whether the RV is a synonymous (S) or non-synonymous (NS) mutation.

The `get_variant_data()` function is applied to all the chromosomes as follows.

```
# Apply function to all 22 chromosomes
SNV_map <- lapply(study_seq, function(x){
  get_variant_data(x$SNV_map)
})
```

Below, the resulting chromosome-specific `.var` files are written to text files named `SNV_map_chr_i.txt`, where "i" indicates the chromosome number. These text files can be found in the Zenodo repository.

```
# Write the results separately to text files
for(i in 1:22){
  write.table(SNV_map[[i]],
              paste0("SNV_map_chr_",i,".txt"),
              row.names=FALSE, quote = FALSE)
}
```

The next section provides a data frame listing the cRVs for each ascertained family.

4.4 List the familial cRVs

First, we obtain a list of family-specific cRVs by chromosome. Each list item corresponds to a chromosome and is a data frame with the family identifiers and the familial cRVs on that chromosome. We print the first two chromosomes in this list for illustration.

```
# Get the FamIDs and their familial cRVs, by chromosome
famcRV_bychrom <- lapply(study_seq, function(x){
  unique(x$haplo_map[, c("FamID", "FamCRV")])})

str(famcRV_bychrom[1:2])
```

```
## List of 2
##  $ :'data.frame':    150 obs. of  2 variables:
##   ..$ FamID : int [1:150] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ FamCRV: chr [1:150] "no_CRV" "no_CRV" "no_CRV" "no_CRV" ...
##  $ :'data.frame':    150 obs. of  2 variables:
##   ..$ FamID : int [1:150] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ FamCRV: chr [1:150] "2_201170321" "no_CRV" "no_CRV" "no_CRV" ...
```

From the output of `str()`, we see that each chromosome in the list `famcRV_bychrom` has a data frame containing the family identifiers and the family's cRV, if any, on that particular chromosome.

Next, we create the function `familial_cRV()` to collapse `famRV_bychrom` into a **single data frame** containing the familial identifier and cRV for each family **across all chromosomes**.

```
# Get the familial_cRVs
familial_cRV <- function(cRV_bychrom){
  # From haplomap data frame get the familial
  # cRVs from the last column FamCRV.
  f_CRV <- lapply(cRV_bychrom, function(x){
    unique(x[which(x$FamCRV != "no_CRV"), ])
  })

  # Combine all of them into a single data frame.
  family_CRV <- do.call("rbind", f_CRV)
  # Order them the data frame according to the family ID.
  family_CRV <- family_CRV[order(as.numeric(family_CRV$FamID)), ]
  # Get the family IDS that are not carrying a cRV,
  # by comparing two data frames.
  no_CRV <- as.numeric(setdiff(cRV_bychrom[[1]]$FamID, family_CRV$FamID))
  no_CRVfam <- rep(c("no_CRV"), length(no_CRV))
  # Create a data frame with families without a cRV.
  df <- data.frame(no_CRV, no_CRVfam)
  # Give the same column names as in families with a cRV.
  colnames(df) <- colnames(family_CRV)
  # Combine both of the data frames.
  family_CRV <- rbind(family_CRV, df)
  # Order the final data frame based on the family ID.
  family_CRV <- family_CRV[order(as.numeric(family_CRV$FamID)), ]

  return(family_CRV)
}
```

We apply the function as follows and get the familial cRV for each family.

```
# Apply the function.
cRVS <- familial_cRV(famcRV_bychrom)

# View the output
head(cRVS, n=5)
```

```
##    FamID      FamCRV
## 1      1 2_201170321
## 17     2 18_63125128
## 31     3 10_89015222
## 51     4 6_108683999
## 67     5 1_155738619
```

The output gives the cRVs for each family. As an example, family ID 1 has a cRV labeled "2_201170321" indicating that it is on chromosome 2 in base-pair position 20117032. Although not shown in the output, there are a few families without a cRV:

```
# Select families which do not carry cRVs
cRVS[cRVS$FamCRV == "no_CRV", ]
```

```
##    FamID FamCRV
## 11    72 no_CRV
## 2     95 no_CRV
## 3    103 no_CRV
```

Three families with IDs 72, 95 and 103 do not carry a cRV. These families have affected individuals with **sporadically** occurring disease.

We save the results in a text file, `familial_cRV.txt`, which can be found in the Zenodo repository.

```
# Save results in a text file.
write.table(cRVS,
            "familial_cRV.txt",
            row.names=FALSE, quote = FALSE)
```

4.5 Create PLINK files

We convert .sam, .var and .geno files to PLINK file formats, .fam, .bim and .bed file formats, respectively. For this task we use the `write.plink()` function in the `snpStats` R package. For example, let's write chromosome 21's .bim, .fam and .bed files.

First we get the sample information with the `plink_format_samp()` function from section 4.2 of this document:

```
# Call the plink_format_samp function for chromosome 21
sample_data <- plink_format_samp(study_seq[[21]]$ped_files)
```

Then we get the variant information for chromosome 21 with the `variant_data()` function from section 4.3 of this document.

```
# Run the function on chromosome 21 SNV_map data
variant_info <- get_variant_data(study_seq[[21]]$SNV_map)
# Add row names to the variant data data frame
row.names(variant_info) <- study_seq[[21]]$SNV_map$SNV
```

Finally, we get the genotypes data for chromosome 21 by modifying the `get_geno_data()` function from section 4.2 to code genotype dosage following the `SnpMatrix` convention of 0 for missing values, 1 for no copies of the alternate allele, 2 for a single copy and 3 for two copies of the alternate allele. This modification allows an object of class `SnpMatrix` to be supplied as an argument to the `write.plink()` function, as required.

```
# Convert haplotype pairs into genotypes in gene-dosage format.

get_geno_data_new <- function(haps){
  gene_dosage <- list()
  IDs <- seq(from = 1, to = nrow(haps), by = 2)
  # Get the column sums.
  for(i in 1: length(IDs)){
    gene_dosage[[i]] <- colSums(haps[IDs[i]:(IDs[i] + 1), ]) + 1
    genotypes <- do.call(rbind, gene_dosage)
  }
  genotypes <- do.call(rbind, gene_dosage)
  return(genotypes)
}
```

We apply the `get_geno_data_new()` function to the SNV haplotypes for chromosome 21.

```
# Apply the function to 21st chromosome
genotype_data <-  get_geno_data_new(study_seq[[21]]$ped_haplos)
```

Let's view the first few rows and columns of the data frame `genotype_data`.

```
# View the first four rows and 12 columns
genotype_data[1:4, 1:12]
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
## [1,]    1    1    1    1    1    1    1    1    1     1     1     1
## [2,]    1    1    1    1    1    1    1    1    1     1     1     1
## [3,]    1    1    1    1    1    1    1    1    1     1     1     1
## [4,]    1    1    1    1    1    1    1    1    1     1     1     1
```

In the output above, we see genotype dosages of 1, indicating that the four individuals carry no copies of the alternate (derived) allele at the first twelve SNVs; this is to be expected as most SNVs are rare.

We convert the `genotype_data` object into a `SnpMatrix` object as follows.

```
# Add row and column names
rownames(genotype_data) <- 1:nrow(genotype_data)
colnames(genotype_data)<- study_seq[[21]]$SNV_map$SNV

# Change the mode of the genotype_data to 'raw'
mode(genotype_data) <- "raw"

# Change the class of the genotype_data object to SnpMatrix and rename it 'genotypes'
genotypes <- new("SnpMatrix", genotype_data)
```

To obtain PLINK .fam, .bim and .bed files for chromosome 21, we apply the `write.plink()` function as follows.

```
write.plink(file.base = "chr_21", snp.major = TRUE,
              snps = genotypes,
              subject.data = sample_data, pedigree = as.numeric(FID),
              id = as.numeric(IID), father = as.numeric(PAT) ,
              mother = as.numeric(MAT), sex = as.numeric(SEX),
              phenotype = as.numeric(PHENO1),
              snp.data = variant_info, chromosome = as.numeric(CHROM),
              position = as.numeric(POS), allele.1 = REF,
              allele.2 = ALT,
              na.code = 0, human.genome=TRUE)
```

```
## Writing FAM file to chr_21.fam
## Writing extended MAP file to chr_21.bim
## Writing BED file to chr_21.bed (SNP-major mode)
## NULL
```

We may apply the above steps to all the chromosomes as follows. First, we apply the `get_variant_data()` function to all the chromosomes.

```
# Apply function to all 22 chromosomes
SNV_map <- lapply(study_seq, function(x){
   result <- get_variant_data(x$SNV_map)
   rownames(result)<- x$SNV_map$SNV
   result
})
```

Second, we apply the `get_geno_data_new()` function to all the chromosomes.

```
# Apply function to all chromosomes
genotype_all <-  lapply(study_seq, function(x){
  result <- get_geno_data_new(x$ped_haplos)
  colnames(result)<- x$SNV_map$SNV
  rownames(result) <- 1:nrow(result)
  result
})
```

Next, convert `genotype_all` to an object of class `SnpMatrix`.

```
# Convert objects as the class of SnpMatrix
genotypes <- lapply(genotype_all, function(x){
  mode(x) <- "raw"
  new("SnpMatrix", x)})
```

To finish, we apply the `write.plink()` function to all chromosomes.

```
# Apply the write.plink function for all the chromosomes
for(i in 1:22){
  write.plink(file.base=paste0("PLINK/chr_", i), snp.major = TRUE,
              snps = genotypes[[i]],
              subject.data = sample_data, pedigree = as.numeric(FID),
              id = as.numeric(IID), father = as.numeric(PAT) ,
              mother = as.numeric(MAT), sex = as.numeric(SEX),
              phenotype = as.numeric(PHENO1),
              snp.data = SNV_map[[i]], chromosome = as.numeric(CHROM),
              position = as.numeric(POS), allele.1 = REF,
              allele.2 = ALT,
              na.code = 0, human.genome=TRUE)
}
```

As a final step, for future reference, we provide the R version and the version of the `SimRVSequences` R package that was used to generate this document.

```
library(SimRVSequences)
sessionInfo()
```

```
## R version 4.1.1 (2021-08-10)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 22000)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_Canada.1252  LC_CTYPE=English_Canada.1252
## [3] LC_MONETARY=English_Canada.1252 LC_NUMERIC=C
## [5] LC_TIME=English_Canada.1252
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] snpStats_1.44.0      survival_3.2-11      reshape2_1.4.4
##  [4] doRNG_1.8.2          rngtools_1.5.2       doParallel_1.0.16
##  [7] iterators_1.0.13     foreach_1.5.1        data.table_1.14.0
## [10] Matrix_1.2-12        forcats_0.5.1        stringr_1.4.0
## [13] dplyr_1.0.7          purrr_0.3.4          readr_2.0.1
## [16] tidyr_1.1.3          tibble_3.1.4         ggplot2_3.3.5
## [19] tidyverse_1.3.1      SimRVSequences_0.2.7
##
## loaded via a namespace (and not attached):
##  [1] httr_1.4.2          jsonlite_1.7.2      splines_4.1.1
##  [4] modelr_0.1.8        assertthat_0.2.1    cellranger_1.1.0
##  [7] yaml_2.2.1          pillar_1.7.0        backports_1.2.1
## [10] lattice_0.20-44     glue_1.4.2          quadprog_1.5-8
## [13] digest_0.6.27       rvest_1.0.2         colorspace_2.0-2
## [16] htmltools_0.5.2     plyr_1.8.6          pkgconfig_2.0.3
## [19] broom_0.8.0         SimRVPedigree_0.4.4 haven_2.4.3
## [22] zlibbioc_1.40.0     scales_1.2.0        intervals_0.15.2
## [25] tzdb_0.1.2          generics_0.1.2      ellipsis_0.3.2
## [28] withr_2.5.0         BiocGenerics_0.40.0 cli_3.1.0
## [31] magrittr_2.0.1      crayon_1.5.1        readxl_1.3.1
## [34] evaluate_0.15       kinship2_1.8.5      fs_1.5.0
## [37] fansi_0.5.0         xml2_1.3.3          tools_4.1.1
## [40] hms_1.1.1           lifecycle_1.0.1     munsell_0.5.0
## [43] reprex_2.0.1        compiler_4.1.1      rlang_1.0.2
## [46] grid_4.1.1          rstudioapi_0.13     rmarkdown_2.13
## [49] gtable_0.3.0        codetools_0.2-18    DBI_1.1.2
## [52] R6_2.5.1            lubridate_1.7.10    knitr_1.38
## [55] fastmap_1.1.0       utf8_1.2.2          stringi_1.7.4
## [58] Rcpp_1.0.7          vctrs_0.3.8         dbplyr_2.1.1
## [61] tidyselect_1.1.2    xfun_0.30
```

References

Nieuwoudt, Christina, Angela Brooks-Wilson, and Jinko Graham. 2020. "SimRVSequences: An R package to simulate genetic sequence data for pedigrees." *Bioinformatics* 36 (7): 2295–97. https://doi.org/10.1093/bioinformatics/btz881.

Purcell, Shaun, Benjamin Neale, Kathe Todd-Brown, Lori Thomas, Manuel A. R. Ferreira, David Bender, Julian Maller, et al. 2007. "PLINK: A tool set for whole-genome association and population-based linkage analyses." *American Journal of Human Genetics* 81 (3): 559–75. https://doi.org/10.1086/519795.