



Swiss Federal Institute of Technology Zurich

Seminar for
Statistics

Department of Mathematics

Master Thesis

Fall 2018

Christopher Salahub

Seen to be done

A statistical investigation of peremptory challenge

Submission Date: March 3 2019

Co-Adviser:

Adviser: Prof. Dr. Marloes Maathuis

Preface

This work would be nowhere near as polished or complete without the effort of Prof. Dr. Marloes Maathuis to ensure I was performing analysis with a clear direction and purpose. I would like to thank her for finding time in her busy schedule to allow for weekly meetings. The group meetings organized by her Ph.D. student Marco Eigenmann were also critical in the development of more nuanced analysis and intuitive visualizations. I thank Marco Eigenmann for organizing them, and Jinzhou Li, Armin Fingerle, Sanzio Monti, and Qikun Xiang for attending my presentations and listening attentively. A special thanks is extended to Cédric Bleutler and Leonard Henckel, both of whom were especially engaged and participated in lengthy discussions both during and outside of the group meetings.

I would like to acknowledge in particular Prof. Dr. Tilman Altwicker for his detailed suggestions on where to look for more legal context on the topic and Prof. Dr. Samuel Baumgartner for his research suggestions. They were very important at providing the necessary information to begin a first investigation of the topic. Of course, without the cooperation of Dr. Ronald Wright, Dr. George Woodworth, Dr. Barbara O'Brien, and Dr. Catherine Grosso for generously providing me with the data from their investigations on the subject of preemptory challenge. Without this data, the visualizations and modelling presented here simply would not have been possible, and so I am exceptionally grateful that they were so enthusiastic to share the fruits of their labour to help cultivate mine.

Abstract

Short summary of my thesis.

Contents

Notation	xi
1 Introduction	1
2 Peremptory Challenges	3
2.1 Jury Selection Procedures	3
2.2 The Role of the Jury	4
2.3 Modern Peremptory Challenge Controversy	5
2.4 The Role of the Peremptory Challenge	7
2.5 History	7
2.5.1 Pre-English History	8
2.5.2 In English Law (1066–1988)	8
2.5.3 In American Law (ca. 1700–1986)	9
2.5.4 In Canadian Law (ca 1800–2018)	10
2.6 Summary	11
3 Data	13
3.1 Jury Sunshine Project	13
3.1.1 Methodology	13
3.1.2 Cleaning	14
3.1.3 Variable Synthesis	18
3.2 Stubborn Legacy Data	18
3.2.1 Methodology	18
3.2.2 Cleaning	19
3.3 Philadelphia Data	19
3.3.1 Methodology	19
3.3.2 Cleaning	19
4 Analysis	21
4.1 Extremes of Partiality	21
4.2 The Impact of Race	22
4.3 Developing an Effective Visualization of Conditional Probability	23
4.3.1 The Mobile Plot	24
4.4 Case Level Summary	25
4.5 Modelling	26
5 Summary	31
5.1 Future Work	31
Bibliography	33
A Complementary information	37
A.1 Including R code with verbatim	37
A.2 Data Processing Code	38
A.3 Jury Sunshine Irregularities	53
A.4 Jury Sunshine Charge Classification	53
A.5 Analysis Code	53

A.6 Using Sweave to include R code (and more) in your report	71
B Yet another appendix....	73
B.1 Description	73
B.2 Tables	73
Epilogue	75

List of Figures

3.1	Simple Charge Tree Example	17
4.1	Mosaic Plot of Defendant and Venire Member Race	23
4.2	First Parallel Coordinate Attempt	25
4.3	The “Mobile Plot”	26
4.4	Strike Tendency by Racial Combination Sunshine	27
4.5	Political Affiliation by Race and Gender (Sunshine)	28
4.6	Lawyer Experience (Sunshine)	29
A.1	Regular expression charge tree visualized	54

List of Tables

4.1	Implied Rejection Boundaries	22
4.2	Strike Rate by Race	23
A.1	Jury Sunshine Irregularities	53
B.1	Test results	73

Notation and Terms

In order to facilitate clarity despite brevity, a list of terms used in this paper is presented here.

Prosecution/State The legal representation which argues for conviction

Defence The legal representation which argues against conviction

Court Reference to the judge, prosecution, and defence

Venire The population sample from which a jury is selected (according to [Mirriam-Webster \(2019a\)](#) derived from the latin *venire facias*: “may you cause to come”)

Jury The final group of (usually) twelve chosen venire members which judge the guilt or innocence of the accused/defendant

Accused/Defendant The individual on trial for a crime

Voir dire From old French “to speak the truth” (see [Mirriam-Webster \(2019b\)](#)), this is the questioning process used by the court to assess the suitability of a venire member to sit on the jury

Struck In the context of a venire member being rejected from the jury, struck indicates removal by peremptory challenge or challenge with cause

Litigants The accuser and the accused

Chapter 1

Introduction

The Gerald Stanley murder trial was noteworthy for all of the wrong reasons. The first reason was the crime itself. The rural region around Biggar, Saskatchewan ([Quenneville \(2018\)](#)) is not known for crime, indeed, the crime statistics collected by Statistics Canada suggest it is one of the safest in the province ([Statistics Canada \(2018\)](#)). Any murder at all would be worthy of attention and subject to plenty of drama. But beyond the damage this trial has done to the community, this trial is noteworthy because it led to a significant re-examination of the legal jurisprudence surrounding the jury selection process culminating in the proposition of Bill C-75 by the Canadian government in March of 2018 ([42nd Parliament of Canada \(2018a\)](#)), less than two months after the trial’s verdict ([Quenneville and Warick \(2018\)](#)).

Bill C-75, in part, aims to ameliorate one of the critical points of contention about the Gerald Stanley case: the use of peremptory challenges in jury selection. The outsized impact of the case was due, in large part, to its racial aspect. Gerald Stanley, a white man, was accused of second degree murder in the killing of Colten Boushie, a First Nations man. Given Canada’s troubled history with First Nations groups, this alone would have been enough to make the trial a flash point for race issues, but that was not the worst aspect of the trial. Rather, it was the alleged use of peremptory challenges to strike five potential jurors who “appeared” to be First Nations, resulting in an all-white jury, that proved to be the most controversial and influential facet of the entire affair ([Harris \(2018\)](#), [MacLean \(2018\)](#)).

With Bill C-75 currently moving through the Canadian parliamentary system, having completed its second reading in June 2018 ([42nd Parliament of Canada \(2018b\)](#)), a close re-examination of the practice of peremptory challenge is warranted. A great deal of ink has already been spilled on both sides of the debate (see [Hasan \(2018\)](#), [Zinchuk \(2018\)](#), and [Roach \(2018\)](#)), but startlingly little of this discussion has been based on any hard evidence on the impact of peremptory challenge in jury selection. This paper aims to provide analysis and evidence to illuminate the topic further by analyzing three separate peremptory challenge data sets collected in the United States, namely [Wright, Chavis, and Parks \(2018\)](#), [Grosso and O’Brien \(2012\)](#), and [Baldus, Woodworth, Zuckerman, and Weiner \(2001\)](#). While this data cannot tell us if challenges were racially motivated in the Stanley trial, stepping back from this fraught legal episode to take a wider view of the practice of peremptory challenge provides a more sober place to start the discussion of its place in modern jury trials.

This paper will proceed in five parts. Chapter 2 provides a brief history of the practice of peremptory challenges in jury trials, in particular explaining their original motivation, past implementations, and how they have developed in the United States, the United Kingdom, and Canada. Chapter 3 proceeds to discuss the three data sets obtained, explaining the sources and collection methods before detailing the cleaning and preprocessing. Chapter 4 then provides the details and results of the analysis performed on the different data sets. It begins discussing the Jury Sunshine data set, which was used as a 'test' set of sorts, where analysis could be flexibly performed before the final analysis methods were turned to the other two data sets. The results of this analysis are compared to previous works in Chapter ???. Finally, the results and findings are summarized in ??, and recommendations based on the observations obtained here are provided.

Chapter 2

Peremptory Challenges

The focus of this text is the practice of peremptory challenges in a jury trial system, a highly specific practice in a particular context which may not be known in detail to the reader. As a consequence, a brief exploration of their history, motivation, and current use is presented here. It is not meant to be exhaustive, but rather to provide context and references for an interested and motivated reader to learn more. Indeed, many details have been omitted from the summary of the history in particular.

2.1 Jury Selection Procedures

Before reviewing the history, it is best to give some context and an explanation for readers unfamiliar with the jury system and general courtroom procedures. While the process of jury selection varies by jurisdiction and crime severity, the general steps shared by jury trials are outlined below. More detail and a discussion of the diversity of jury selection procedures can be found in [Ford \(2010\)](#), [Hans and Vidmar \(1986\)](#), and [Van Dyke \(1977\)](#). To select a jury:

- i.) Eligible individuals are selected at random from the population (using a list known as the *jury roll*) of the region surrounding the location of the crime, the sampled individuals are called the *venire*
- ii.) The venire is presented to the court, either as a group or sequentially (borrowing the names of [Ford \(2010\)](#): the “struck-jury” system and the “sequential-selection” system, respectively)
- iii.) The presented venire member(s) are questioned in a process called *voir dire*, which can result in three possible outcomes for each venire member:
 - (a) The venire member is removed with cause, the cause provided by either the prosecutor or defence lawyer and admitted by the judge
 - (b) The venire member is removed by a *peremptory challenge* by the prosecutor or defence lawyer, where no reason need be provided to the court; such privileged rejections of a venire member are limited in number for both lawyers (in Canada a maximum of 20 such challenges per side per defendant are allowed [[Government of Canada \(1985\)](#)])

- (c) The venire member is accepted into the jury, and so becomes a juror
- iv.) Steps i-iii are repeated until the prosecution and defence fail to reject the desired number of jurors.

As mentioned above, the details in this process can vary greatly by region. One of the greatest sources of this variation is the creation of jury rolls. In the United States the method is somewhat homogeneous: they are typically selected using lists of registered voters (see [Van Dyke \(1977\)](#) chapter two), but in Canada their creation is far more varied. Ontario uses a combination of municipal voter lists and First Nations band lists (see [Ministry of the Attorney General of Ontario \(2018\)](#)), while in Saskatchewan - the province of the Gerald Stanley trial - the jury roll is created from the data in the central government health insurance agency in accordance with [Government of Saskatchewan \(1998\)](#).

Clearly, the variation in these methods will create differences in the universe of the sampled jury rolls relative to the population they are meant to reflect. Such differences are no doubt important to the coverage of the population which is present in the jury selection process (see [Iacobucci \(2013\)](#)), but these differences are not of primary interest to this paper. Rather, the steps presented afterwards are those to be investigated.

This leads to the two presentation methods presented in step ii, [Ford \(2010\)](#) and [Van Dyke \(1977\)](#) both note that the predominant method in the United States and Canada is the sequential-selection system. This is perhaps due to the relative efficiency of the method, as it is clear that in the sequential system voir dire need not be performed on the entire venire, only a subset. Contrast this with the struck-jury system, where the entire venire must be reviewed in every trial.

Finally, the scope of voir dire is radically different in the United States and much of the British Commonwealth. [Van Dyke \(1977\)](#) notes that Canada and the United Kingdom do not allow questions in areas of “non-specific” bias, or bias which is not directly related to the case before the court. That is to say, while it would be perfectly valid to ask a venire member for a murder case about their work history in the United States, such a question would only be allowed in Canada or the United Kingdom if occupation was specifically related to the crime.

This difference in procedure creates a far greater emphasis on the voir dire process in the United States, as noted by [Hans and Vidmar \(1986\)](#). [Hans and Vidmar](#) go further than this, and surmise that the key reason for this marked departure in procedure is a difference in philosophy. To borrow a quote from page 63 of [Hans and Vidmar \(1986\)](#):

In Canada... the courts have said that we must start with an initial presumption that “a juror will perform his duties in accordance with his oath”

This doctrine places a responsibility on the jurors themselves to overcome their biases and accept arguments in spite of them. This stands in stark contrast to the American attitude implied by the emphasis on expansive voir dire: that certain prejudice cannot be overcome by jurors themselves.

2.2 The Role of the Jury

Such a difference in viewpoint is especially relevant given the purpose of the jury. The central function of a jury in a jury trial system is to judge the innocence or guilt of an

accused in light of the presented evidence, a function which has had drastically different forms throughout history. In the distant past, [von Moschzisker \(1921\)](#) and [Hoffman \(1997\)](#) report that the central function of the jury was to collect evidence, essentially assuming the role commonly performed today by police detectives. Such a role justified the archaic practice of selecting only the most “trustworthy” individuals of some reknown.

This is contrasted by the modern jury, which performs no collection of evidence. It is, ideally, a panel of peers or “equals” of the accused sampled at random from the population, an idea which did not develop until 19th century Britain (see page 28 of [Hans and Vidmar \(1986\)](#)) and was not applied using random sampling until some time later (see [Hoffman \(1997\)](#) and page 29 of [Hans and Vidmar \(1986\)](#)). The modern jury is meant to apply the law, as told to them by the judge¹, to the case at hand. Evidence of the guilt of the accused is presented to the jury by the prosecutor, while evidence meant to exonerate is presented by the defence.

The jury listens to the evidence, considers the law as presented by the judge, and must (typically) reach a unanimous decision of guilt or acquittal. Such a decision cannot be overturned by the judge of the court, and the judge must then determine sentencing based on the decision of the jury and the letter of the law¹. It should be clear that the jury therefore has tremendous power in the judgement of any case. The philosophical and ethical justification for such power is well explained by [Woolley \(2018\)](#), and best summarized by a quote from [Supreme Court of Canada \(1991\)](#):

The jury, through its collective decision making, is an excellent fact finder; due to its representative character, it acts as the conscience of the community; the jury can act as the final bulwark against oppressive laws or their enforcement; it provides a means whereby the public increases its knowledge of the criminal justice system and it increases, through the involvement of the public, societal trust in the system as a whole.

While such enthusiastic support for juries has not been expressed by all countries which practice them, the justification is entirely consistent with the histories and discussions presented by [Hoffman \(1997\)](#), [von Moschzisker \(1921\)](#), [Hans and Vidmar \(1986\)](#), [Van Dyke \(1977\)](#), and others. This suggests that the [Supreme Court of Canada \(1991\)](#) lionization of the jury system is a fair representation of the perceived role of the jury throughout those countries which use them.

2.3 Modern Peremptory Challenge Controversy

If the general utility and importance of the jury is clear, the same cannot be said for peremptory challenges. The privileged removal of a venire member² without any justification has seen persistent allegations of abuse, often around the use of these challenges by state prosecutors.

In the United States, the criticism has focused on racial discrimination, and has led to significant changes in their allowed use, through cases such as *Swain v. Alabama* ([Supreme Court of the United States \(1965\)](#)) and *Batson v. Kentucky* ([Supreme Court of the United](#)

¹[Hans and Vidmar \(1986\)](#) note that this system actually varies throughout the US, though the jury and judge powers described here are consistent across Canada.

²To be replaced by another, *randomly selected* venire member

[States \(1986\)](#)). The first of these cases, *Swain v. Alabama*, established in 1965 that the systematic exclusion of venire members of a particular race would be unconstitutional discrimination under the Fourteenth Amendment, but argued that a “*prima facie*” (or “based on first impression”) argument of discrimination was not adequate to prove this³. This placed a significant burden on the side taking issue with a particular peremptory challenge to demonstrate that the choice had been discriminatory.

However, this ruling was overturned only 21 years later in the 1986 case *Batson v. Kentucky*, which allowed the party objecting to a challenge to use a *prima facie* argument which must be countered by a race-neutral reason that satisfies the judge. If no such reason can be supplied, the challenge would not be allowed. This created a new challenge which could be used to keep a venire member despite the use of a peremptory challenge: the so-called “Batson Challenge”. While the effectiveness of this system of additional challenges is questionable both practically and in abstract (see [Page \(2005\)](#) and [Morehead \(1994\)](#), and a particularly strong response in [Hoffman \(1997\)](#)), it has only been extended to allow Batson challenges for both the sex and race of venire members⁴.

In Canada, the controversy has also had a racial component. Racial bias in Manitoba against First Nations venire members was alleged in 1991 in a report produced after an inquiry by the provincial government (see [Roach \(2018\)](#)). More damning still was the [Iacobucci Report](#) on First Nations representation in juries. This report proposed an explicit restriction to the practice when it recommended:

an amendment to the Criminal Code that would prevent the use of peremptory challenges to discriminate against First Nations people serving on juries.

Despite these recommendations and allegations, there had not been a significant political effort to reform the peremptory challenge system until the Gerald Stanley trial culminated in the tabling of Bill C75 [42nd Parliament of Canada \(2018b\)](#), which would abolish the peremptory challenge outright. As of the writing of this paper, the bill has not been approved by the Government of Canada, but it seems likely to become law in the near future.

In doing so Canada would join the United Kingdom. Significant controversy around the use peremptory challenges in the United Kingdom has already resulted in the abolition of the practice by the Criminal Justice Act of 1988. The specific controversy was the result of the Cyprus spy case in the late 1970s, which led to a “sustained campaign in Parliament and in the press alleging that defence counsel were systematically abusing it” (see [Hoffman \(1997\)](#))⁵.

³In the actual case, not a single black juror had sat in Kentucky in the previous 15 years, despite composing 26% of the jury-eligible population. In Swain’s trial, six of the eight black venire members were rejected by state prosecutor peremptory challenges, and the other two removed for cause, leaving not a single black juror to judge Swain, a black man. This was the *prima facie* argument presented by Swain’s defence team against the state prosecutors of Alabama, and it was rejected as insufficient to prove discrimination

⁴The use of Batson Challenges for sex was established in [Supreme Court of the United States \(1993\)](#)

⁵It should be noted that this did not abolish the use of “standing-aside” by the Crown, although the practice was restricted to national security trials and heavily curtailed, with strict guidelines to its use outlined by [Attorney General’s Office of the United Kingdom \(2012\)](#).

2.4 The Role of the Peremptory Challenge

Despite these legal changes, recommendations, and a great deal of articles providing analysis against the practice (see, for example, [Hoffman \(1997\)](#)), the topic of the peremptory challenge remains controversial. The modern motivation and justification for the practice in spite of all of the controversy was perhaps best described by Justice Byron R. White in [Supreme Court of the United States \(1965\)](#):

The function of the challenge is not only to eliminate extremes of partiality on both sides, but to assure the parties that the jurors before whom they try the case will decide on the basis of the evidence placed before them, and not otherwise. In this way, the peremptory satisfies the rule that, “to perform its high function in the best way, justice must satisfy the appearance of justice.”

Such a justification is reminiscent of the now famous words of Lord Chief Justice Hewart in *R. v. Sussex Justices* in 1924: “Justice should not only be done, but should manifestly and undoubtedly be seen to be done” (as reported in [Richardson Oakes and Davies \(2016\)](#)). While these words originally only referred to the pecuniary interest of court staff involved in the case, they have since come to express the idealized expectation that both the defence and prosecution find the judge and jury acceptable, as explored by [Richardson Oakes and Davies \(2016\)](#)⁶.

This defence suggests two modern justifications for the peremptory challenge. The first is that of removing venire members with “extreme” bias, and the second is the creation of a jury which is composed of jurors mutually acceptable to both the defense and the prosecution. Those who defended the practice of peremptory challenges in Canada after the Gerald Stanley trial, including [Hasan \(2018\)](#) and [Macnab \(2018\)](#), seem to use this defence or some variant of it to argue in favour of keeping the practice. However philosophically appealing these two claims are, in light of all of the controversy surrounding the peremptory challenge, perhaps a critical and empirical examination of these assertions is warranted.

2.5 History

Such an analysis most appropriately begins with a historical explanation of the peremptory challenge. Roughly, the presentation of the history of jury trials here follows the comprehensive and exhaustively referenced description provided by [Hoffman \(1997\)](#). Two of the references [Hoffman](#) uses extensively, [Hans and Vidmar \(1986\)](#) and [Van Dyke \(1977\)](#), provided useful context while specific details provided by [von Moschzisker \(1921\)](#), [Forsyth \(1994\)](#), [Brown, McGuire, and Winters \(1978\)](#), and [Brown \(2000\)](#) helped to create a clearer picture of particular periods of jury history. Information regarding the history of the Canadian system was provided by [Brown \(2000\)](#) and [Petersen \(1993\)](#). For an excellent exploration of the nineteenth century, a formative time for the development of challenge law, see [Brown \(2000\)](#).

⁶Such grand generalizations and myth-making can also be seen in the common belief that the right to a trial by jury was originally established in the Magna Carta, an idea which is not supported by the relevant historical evidence (see [Hoffman \(1997\)](#) and [Van Dyke \(1977\)](#) for a detailed discussion and more accurate history).

It must be noted that certain important trials in the development of the peremptory challenge system have been excluded from the summary provided here. This was done deliberately, as the history presented here is only meant to explore the practice of peremptory challenges throughout history in broad terms. All of the sources listed above are much more thorough, by merit of their singular focus on the analysis of the practice from a legal and historical perspective, while this work devotes more to empirical and statistical analysis.

2.5.1 Pre-English History

Although precise timelines are hard to establish, there is evidence that jury trials have occurred in some form or another since antiquity. The concept, that of judgement by a group of peers, is so ancient that it is prevalent not only in historical records, but in myth. As [Hoffman \(1997\)](#) indicates, both Norse and Greek mythology feature groups of individuals assessing the guilt or collecting evidence about the actions of a peer.

Outside of the realm of myth, [Hoffman \(1997\)](#) reports that there is evidence of the use of juries in Ancient Egypt, Mycenae, Druid England, Greece, Rome, Viking Scandanavia, the Holy Roman Empire, and Saracen Jerusalem. It should be noted that in none of these areas was the jury trial the primary form of conflict resolution practiced. Nonetheless, it is clear the jury trial has a broad and long history of use.

Something similar to the modern peremptory challenge does not appear until Rome, however. The Roman *Judices* were groups of senators selected to judge the guilt of the accused in a legal case. According to [Hoffman \(1997\)](#), 81 Senators would be chosen to sit on one of these *Judices*, after which the litigants were permitted to remove fifteen of these Senators each. This egalitarian reduction of the jury size seems analogous to the modern peremptory challenge system, as it places the power of removal with the litigant and suggests no justification is necessary for their removal.

2.5.2 In English Law (1066–1988)

Peremptory challenge did not reach its modern form, as outlined in [2.1](#), until it was established in the English legal system. It should be noted that despite some previous debate on the topic, the most modern historical evidence suggests that the basis of the English practice was not related to the system used in the selection of *Judices* in Rome. The English system appears to be its own beast entirely.

The dominant historical interpretation is presented by [von Moschzisker \(1921\)](#) and [Hoffman \(1997\)](#): that the jury system was introduced to England during the Norman conquest of 1066 by William the Conqueror. The practice, however, was not made official until the Assize of Clarendon in 1166 by Henry II, and it was not until the outlaw of trials by ordeal (the most common method of trial at that time) in 1215, that peremptory challenges began to appear in England in the late thirteenth century. The challenges were officially recognized in 1305 when Parliament outlawed their use by the Crown, only to replace them with an analogous system of so-called “standing-aside”⁷.

⁷For a detailed explanation of this system see [Hoffman \(1997\)](#) and [Brown \(2000\)](#)

It should be noted here that although the challenges issued between the Assize of Clarendon and this 1305 act are called “peremptory,” they may not have served the same purpose, nor shared the same justification, as the modern challenges. Indeed, as [Hoffman \(1997\)](#) argues convincingly, these challenges may have been closer to modern challenges with cause. The argument hinges on the paradigm of royal infallibility and absolutism which was present in the late medieval period when the peremptory challenge first appeared (see [Burgess \(1992\)](#)).

Under royal absolutism and infallibility the argument for peremptory challenges is quite simple. If the king cannot be wrong in his judgement and he has some reason to feel that a venire member cannot serve on the jury, then he need not say why he thinks that is so, as his judgement is correct in any case. Indeed, asking for an explanation would be disrespectful and providing one undignified. The Crown prosecutors, as representatives of the king, would be similarly shielded from criticism.

Such an argument is further supported by the abolition of their royal use in 1305, the language of which suggests that peremptory challenges were originally the privilege of the Crown (see [Hoffman \(1997\)](#) and [Van Dyke \(1977\)](#)), with none being granted to the defence. [Hoffman \(1997\)](#) suggests that as royal infallibility grew out of favour, peremptory challenges were granted to the defence rather than being removed entirely.

Whatever the logic of the expansion of these challenges to the defence, their legal limits are recorded more precisely⁸. From a maximum of 35 challenges allowed at their peak in the fourteenth century, the number of challenges allowed only decreased over time until their abolition in 1988 (discussed in [2.4](#)).

2.5.3 In American Law (ca. 1700–1986)

[von Moschzisker \(1921\)](#), [Hoffman \(1997\)](#), and [Van Dyke \(1977\)](#) all agree that the early English colonists that came to North America accepted the jury system with peremptory challenges as common law well before the establishment of the United States of America. [Hans and Vidmar \(1986\)](#) note, however, that the difficulty of ocean travel and the overall indifference of appointed Crown representatives in the colonies led to an increased importance of the jury trial and the role of challenges to these early colonists as a way to exercise some degree of community control in the face of laws drafted in a distant country and implemented by unsympathetic authorities⁹.

It is somewhat interesting then, that the United States constitution makes no mention of the practice of peremptory challenges. The Sixth and Seventh Amendments specify a great deal of the jury system, including the right to public defense and an impartial jury drawn from the district of the crime, but make no mention of a right to the exercise of peremptory challenges, or any challenges whatsoever (see [Constitution of the United States \(1788\)](#)).

⁸see [Brown \(2000\)](#) for a detailed examination of the case law developing around challenges in the nineteenth century

⁹For more detail on this development among the early colonists, it is instructive to read about the Zenger trial of 1734 (described on pages 33-35 of [Hans and Vidmar \(1986\)](#)). Not only does this trial say a great deal about the attitudes of the colonists at the time, but it also presents the idea of a jury assessing guilt and “wrongness” using their own conscience rather than just settling fact. The precept of the modern jury trial in Canada (see [Woolley \(2018\)](#)) is based on this very idea

As [Hans and Vidmar \(1986\)](#) report on page 37, an original draft of the Sixth Amendment expressly included challenges for cause, but the debate around their inclusion resulted in the removal of their mention. They continue to say that at the time, even some proponents of the challenge considered the reference unnecessary, as the practice was implied by the text which remained, referring to a trial by an “impartial” jury. Another result of these debates was the adoption of the extensive voir dire process which allows questions of general bias¹⁰.

Critically, there appears to have been no discussion around the inclusion of peremptory challenges (see [Hans and Vidmar \(1986\)](#) and [Hoffman \(1997\)](#)). Despite the clear importance of the jury trial to the drafters of these amendments, it would seem the peremptory challenge was not considered to have anywhere near the same significance as judgement by an impartial jury of local peers¹¹.

Regardless of this, as [Brown \(2000\)](#) notes, the importance and use of challenges increased in the United States in the nineteenth century following American independence due to a desire to prevent the tyranny of the state. This desire also led to the adoption of a limited number of peremptory challenges for the prosecution, rather than the possibly unlimited stand-asides that were allowed under British law to prosecutors (see [Van Dyke \(1977\)](#), page 150).

While the specific numbers of peremptory challenges allowed to both sides and the required motivation of challenges for cause have varied over time (see [Hoffman \(1997\)](#) and [Brown \(2000\)](#)), they have remained a feature of the American legal system, and numerous Supreme court cases (detailed by [Hoffman \(1997\)](#)) have merely served to make the use of challenges more specific and codified. It was not until *Batson v. Kentucky* in 1986 that this system of challenges was drastically changed with the introduction of Batson challenges (described in 2.3).

2.5.4 In Canadian Law (ca 1800–2018)

Canadian law, inspired by a close relationship to both the British Crown and the United States, seems to have adopted elements of both legal systems in its development of peremptory challenges in the nineteenth century. As discussed by [Brown \(2000\)](#), Canada adopted the American practice of replacing prosecutorial stand-asides in favour of a more egalitarian limited number of peremptory challenges to both sides. Despite this, the Canadian voir dire process remains limited and much more similar to the British one, as does the system of challenges for cause (see page 48 of [Hans and Vidmar \(1986\)](#)).

One perfect demonstration of this departure is the Canadian constitution. As in the United States, the Canadian constitution fails to mention challenges. The British North America Act of 1867 (see [Constitution of Canada \(1982\)](#)), which established Canada’s independence from England, makes no mention of legal rights of the accused, indicating a deference to legal precedent in England. It is not until the Charter of Rights and Freedoms

¹⁰This is described on page 37-38 of [Hans and Vidmar \(1986\)](#), though [Brown \(2000\)](#) notes that 1807 Burr trial was also highly significant in the development of general voir dire in the United States

¹¹Indeed, as *Batson v. Kentucky* and *Swain v. Alabama* have both shown ([Supreme Court of the United States \(1986\)](#) and [Supreme Court of the United States \(1965\)](#)), the modern interpretation of “impartial” may preclude the use of peremptory challenges altogether

in 1982¹² that such rights were guaranteed in a legal Canadian document. Notably, its language is considerably more vague than the United States Sixth and Seventh Amendments, guaranteeing only “the benefit of trial by jury” (see [Constitution of Canada \(1982\)](#)).

This “eclectic” incorporation of both American and English case law, to borrow the term used by [Brown \(2000\)](#), led to a system somewhere between the English and American systems, but decidedly closer in operation to the English system. It should be noted, however, that as Canada grew more populous in the twentieth century and developed a greater legal precedent and more experienced judges of its own, this reliance upon its former colonial master and its more powerful southern neighbour seems to have diminished in importance. Viewing Supreme court rulings from recent decades reveals a great deal of reference to Canadian legal precedent rather than to the precedent of the other two countries.

2.6 Summary

The peremptory challenge, a practice of much controversy in the English-speaking world, seems to have started in its modern form as a privilege of the King of England in the thirteenth century. After its conception, it spread with English conquest and colonization, with new colonies and local governments accepting the practice based primarily on the adoption of English legal precedent. Though it was abolished in England in 1988, it remains a fixture of American jury trials, and is accompanied there by a thorough and invasive voir dire process which is not seen in Canada nor the United Kingdom.

Though the practice has historical longevity, it is not guaranteed by the constitutions of Canada or the United States, and has been a practice of considerable legal debate and significant change throughout its history. In England this culminated in the Cyprus spy trial, in the United States in *Batson v. Kentucky* and *Swain v. Alabama*, and in Canada in *R. v. Stanley*: the Gerald Stanley murder trial. As a consequence, the broad agreement of the importance and propriety of a jury has conferred little consensus on the place peremptory challenges in the selection of juries.

Indeed, it seems increasingly impossible for the jury to function in a way consistent with its demanding ideals with the peremptory challenge still present. Its spotted history and use to exclude certain minorities may undermine its purported use as a tool to ensure the acceptance of a trial’s outcome by both litigants. The three court cases mentioned above are a demonstration how the peremptory challenge can be used to create a jury which is actually unacceptable to one litigant in the case.

¹²This was the year of patriation of the Canadian constitution. As independence was granted by the British Parliament, the British North America Act outlining Canada’s laws was a British law and changing it was the prerogative of the British Parliament rather than the Canadian one. It was not until the Constitution Act of 1982 that the Canadian constitution became a Canadian law. For a more detailed history see [Sheppard \(2018\)](#)

Chapter 3

Data

Without data, performing an analysis that incorporated more than the history and legal argumentation presented in Chapter 2 is impossible. This proved problematic. While the motivation of this text was a Canadian case, no comprehensive Canadian data sets which examined jury selection in Canada could be found. The increased prominence of the jury selection process in the United States garnered a more fruitful search.

The author is heavily indebted to [Wright et al.](#); [Grosso and O'Brien](#); and [Baldus et al.](#). These authors shared their data freely with the author, providing him with a wealth of data to analyse empirically. As a consequence of the multiple separate data sets, however, care must be taken to describe each of the data sets separately in order to capture adequately the different methodologies and sources they represent. Critically, it should be noted that each of these papers represents effort on the part of the authors. As [Wright et al. \(2018\)](#) notes:

limited public access to court data reinforces the single-case focus of the legal doctrines related to jury selection. Poor access to records is the single largest reason why jury selection cannot break out of the litigato's framework to become a normal topic for political debate

Currently, the collection of jury data is difficult, as many courtrooms have not digitized past records and concerns over privacy limit the release of those records, which are stored as paper documents in the case file (see [Wright et al. \(2018\)](#)). This limits the ability of an individual to ask for summaries across numerous trials or to view the jury selection process on a scale beyond the basis of one case. Thus, to gather aggregate data the authors of these papers necessarily used different collection techniques dictated by the scope of collection desired and the procedures of the court systems from which data was collected.

3.1 Jury Sunshine Project

3.1.1 Methodology

The Jury Sunshine Project ([Wright et al. \(2018\)](#)), so named as it was carried out in order to shed light on the jury selection process, is the most extensive data set which was provided to the author. It endeavoured to collect jury data for all felony trial cases in

North Carolina in the year 2011, which ultimately resulted in a data set that detailed the simple demographic characteristics and trial information of 29,624 individuals summoned for jury duty in 1,306 trials. Note that not all entries were complete.

Due to the scope of the project, there are a number of problems which had to be solved by the authors. The first of these was simply identifying which court cases went to trial in 2011, in order to direct resources effectively. This was accomplished by downloading publicly available case data from the North Carolina Administrative Office of the Courts (NCAOC)¹ and determining the case numbers and counties of cases which went to trial. Wright et al. state that this likely missed some cases which went to trial, but that they were confident that a “strong majority” of trials was collected, which did not systematically differ from those excluded.

This list was then used to perform a pilot study to refine recording practices before undertaking a more general survey where “law students, law librarians, and undergraduate students” (called *collectors* for convenience) visited court clerk offices to collect the relevant case data, including the presiding judge, prosecutor, defence lawyer, defendant, venire members, charges, verdict, and sentence. The case files also included data about whether a venire member was removed by cause or peremptorily, and the party which challenged in the peremptory case. Using public voter databases, bar admission records, and judge appointment records, these collectors were able to determine demographic (race, gender, and date of birth) and political affiliation data for the venire members, lawyers, defendants, and judges. This data set was stored in a relational database provided to the author by Dr. Ronald Wright.

The analysis of the data provided in Wright et al. (2018) was limited to aggregate summaries of the trends at the venire member level. That is to say, they examined the strike trends for both the defence and the prosecution, conditioning on some additional variables. There was also spatial analysis performed, where different urban counties were directly compared. These analyses were not statistical in nature, and were displayed using contingency tables. Regardless the stark differences between prosecution and defence with regards to race were a key finding when the aggregate data was analyzed.

3.1.2 Cleaning

Flattening the Data

For greater expediency of analysis, the relational database of the Jury Sunshine Data was first flattened. The relational database was read into Microsoft Excel and the `readxl` package (Wickham and Bryan (2018)) was used to read the excel file into the programming language R. A wrapper for the `merge` function was developed which provided simple output detailing failed matches in an outer join in order to ensure that the flattening of the data into a matrix did not miss important data due to partial incompleteness. The code for this wrapper can be seen in A.2.

¹The link provided in the Jury Sunshine Paper to the specific source (http://www.nccourts.org/Citizens/SRPlanning/Statistics/CARports_fy16-17.asp) does not appear to be working as of January 2019, however the NCAOC seems to provide an API functionality at <https://data.nccourts.gov/api/v1/console/datasets/1.0/search/>

This wrapper revealed only a small number of irregularities in the data, which are detailed in [A.3](#):

- i.) Twenty-nine charges missing trial information such as the presiding judge (all of trials with IDs of the form 710-0XX)
- ii.) Twenty-six prosecutors not associated with any trials and missing demographic data
- iii.) One trial missing charge information

Ultimately, the jurors for trial ID 710-01, the trial missing a charge from above, were included in the data as their records were complete otherwise. The prosecutors and charges which could not be joined were excluded from any future analysis, as they could have easily been included by collectors by accident. Due to the small relative size of these inconsistencies relative to the size of the data set, they did not cause concern.

Uninformative Columns

Of course there were other irregularities in the data than the obvious ones that arose in the flattening process. There were a handful of likely sources for these errors. The first of these is the anonymization of the data for public use. The private data includes a wealth of privileged data such as juror name and address, and these were removed in the data given to the author.

As a consequence of this anonymization as well as the inclusion of rarely used columns such as those for additional notes, some columns of the data contained only NA values. Most baffling of these was the `BirthDate` variable in the `Jurors` table, as there was no clear reason for this data to be missing. Thankfully, none of the missing columns were relevant to the joins performed in flattening, and they would have been only secondary in data analysis. As a consequence, these uninformative columns were simply removed from the data.

Coding Inconsistencies

Related to this problem was the issue of inconsistently coded variable levels. An example of these inconsistencies would be levels recorded as both lower and upper case letters, or the presence of “?” instead of “U” for unknown values. It is very likely this inconsistency was a direct result of the data collection method which used many data collectors working independently in different places at different times. Thankfully, [Wright et al.](#) provided the codebook used by data collectors, which served as the authoritative reference for the admissible factor levels of demographic variables. Solving this problem was as simple as setting all demographic variable levels to be uppercase and replacing obviously mis-specified levels.

One specific inconsistency which should be noted is that of the outcome, which had a handful of entries recorded as “HC”, an inadmissible level not defined by the codebook. It is quite possible that this level represented a typo, as the “H” and “G” keys are adjacent on the American QWERTY keyboard layout, and “GC” was the code for ‘guilty as charged’, but out of a desire to be conservative with the data the occurrence of this outcome was replaced with U instead. Additionally, the inadmissible level “G” was replaced by GC.

Swaps

A more difficult problem of level misspecification was the presence of what appeared to be columns with swapped values, frequently occurring with the gender column (the admissible levels of which are “M”, “F”, and “U”) and the political affiliation column (the admissible levels of which are “D”, “R”, “I”, or “U”). The aforementioned “swaps” appeared as records in which, for example, the gender was recorded as “R” and political affiliation as “M”. More complicated swaps of three columns also occurred. To address this problem, the `IdentifySwap` function was written (see line 108 in [A.2](#)).

The `IdentifySwap` function accepts two arguments: a data frame with named columns and a named list of vectors of the acceptable levels for some of the column names. It then performs vectorized checks of the specified column names and presents any rows which may have swaps or errors interactively to the user, along with a suggested reorder to “un-swap” the row. The user can press enter to accept the suggested reordering, enter some other reordering, or enter 0 to indicate that the row was not a true swap, but simply an error. The un-swapped entries are then returned to the data, and the rows with errors have the erroneous values replaced by “U”, the universal code for “Unknown” within all data variables².

The source of these swaps is also most likely the data collection method. The codebook provided specifically notes that the data collection was meant to record the race (R), gender (G), and political affiliation (P) data in the form RGP, but it is not inconceivable that it would occasionally have been recorded or entered in some other ordering in the tedium of data entry. In any case, this problem affected only 431 records of the nearly 30,000, suggesting that the recorded error rate was not unacceptably large.

Charge Classification

Perhaps the least regular data in this data set was that of the charge text. Due to the lack of any codebook guidance about the standard way of recording a charge in a trial, identical charges were recorded in numerous ways. The first method used to combat this was removing non-alphanumeric characters, extra spaces, and converting all charges to lower case. This still left a considerable variation, however. Consider the charge of breaking and entering, for example, even with this simple preprocessing performed the entries varied significantly (e.g. “break or enter”, “breaking andor entering”, “breaking and or entering”, etc.).

As a consequence, the processing was more involved. First the most common versions of the charge text for the charges were all regularized to be identical (see `StringReg` in [A.2](#)). Next, a regular expression classification tree was developed, which would also account for specific features of a charge. When identifying murder, for example, it seemed important to ensure attempted murder was separated from murder itself, and separating first and second degree was also desired. This tree would, when presented with a charge, apply the regular expressions at each node to the charge. If the charge matched the expression at a node, the regular expressions of that node’s children were applied to the charge until it was classified to some leaf node, each of which had a standardized value which replaced

²One notable exception to this insertion of “Unknown” was the case of the judge Arnold O Jones II, whose gender was not recorded in the data, but who was identifiable as a man using a quick Google search of his unique name

the charge. A small example of this structure is displayed in Figure 3.1, and the full tree is visualized in A in Figure A.1.

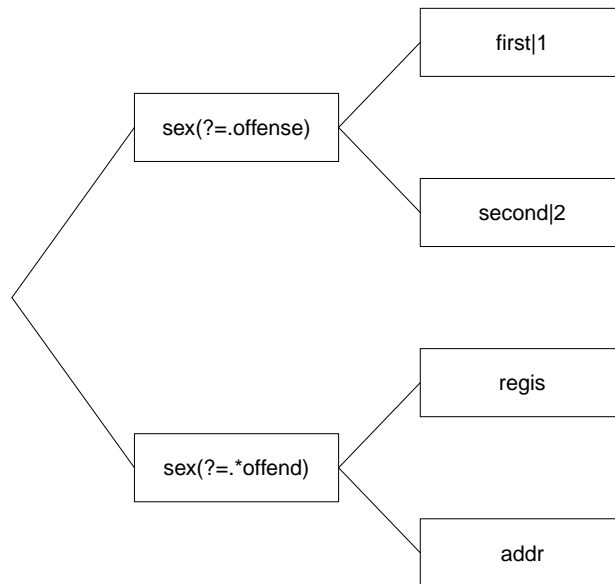


Figure 3.1: A example of a simple charge classification tree to separate the sexual offenses from charges leveled against previously known sex offenders. A charge would be classified from most general on the left to most specific on the right.

By performing regularization using this charge tree, regularized charges were guaranteed. The cost of this regularization was the inability to classify all crimes, however. Of the 1407 charges present in the data, the tree provides regularization for 1209. With additional time and inspection of the failed matches, the tree could conceivably be expanded to regularize all charges. As this was not the primary investigation of the report, however, such effort was not expended.

Instead, a number of helpful aggregation and extraction functions were developed to allow for the charges to be further simplified, as seen in A.2. In this report, they were aggregated by intuitive classes: sex-based offenses, thefts, murders, drug charges, violent offenses not otherwise classified, and driving charges. However, other classes, such as the North Carolina felony classes themselves (as provided by [North Carolina Sentencing and Policy Advisory Commission \(2017\)](#)), may provide a more informative classification rationale.

Variable Level Renaming

The final step of the data cleaning process was to convert the uninformative codes used to indicate variable values to more intuitive and clear names (for example to convert “I” in the political affiliation variable to “Ind”, a clearer indication of independent). Certain variables which were already clear, such as gender (codes “M”, “F”, “U”), were not renamed due to the clarity of the one letter representations.

3.1.3 Variable Synthesis

In order to expand the possible analysis and visualization potential, a number of variables were synthesized from the Jury Sunshine data set. They are detailed below.

Race Match A logical variable which is true for a venire member if they are the same race as the defendant, and false otherwise. This variable was motivated in particular by the Gerald Stanley trial, in which the implicit contention of those who have taken issue with peremptory challenge is that the First Nations venire members were removed by the defence as their race did not match that of Stanley in the racially charged trial.

Guilty Logical indicator indicating whether the trial outcome was guilty or not

Visible Minority Logical indicator of non-white venire member race

Race of Striking Party Factor variable which gives the race of the prosecution if the venire member was struck by the prosecution, the race of the defence if the venire member was struck by the defence

Simplified Race Due to the scarcity of the other minority races, this variable simplified the race provided to white, black, or other for the venire member

Simplified Defendant Race The same as the simplified race for the defendant races

3.2 Stubborn Legacy Data

3.2.1 Methodology

[Grosso and O'Brien \(2012\)](#) also provided data to the author, albeit a more limited set. This study, also based in North Carolina, focused on the trials of inmates on death row as of July 1, 2010, yielding a total of 173 cases. In each proceeding, the study examined only those venire members not excluded for cause, and critically the analysis of the study focused only on prosecutorial peremptory challenges. Besides collecting demographic data as in the Jury Sunshine Case, this study also collected attitudinal data for the venire members.

Staff attorneys from the Michigan State University College of Law were responsible for the data collection in this study. The work was performed similarly to the Jury Sunshine Data, using case files to collect information about the court proceedings such as the peremptory challenges used, presiding judge, prosecutor, and defence lawyer. Detailed verdict and charge information was not collected, as the preselection criteria of death row inmates made the verdict clear, and the death penalty can only be applied for certain crimes.

To collect demographic and attitudinal data, the juror questionnaire sheets were consulted³. These sheets are typically used as a component of voir dire, in order to make the process more efficient and determine venire members categorically ineligible for jury duty. As a result, they inquire about opinions on the death penalty, for example, as well as

³As [Grosso and O'Brien \(2012\)](#) observe, self-identified race may be the most accurate source of racial group identification

demographic questions. As not all jury questionnaires were available, additional information was collected from jury roll lists to determine the races of the final jury members. It should be noted that this collection was done blind and to high standards of proof, and a reliability study carried out in [Grosso and O'Brien \(2012\)](#) indicated that under this system the race coding was 97.9% accurate when the standards were met. Those for whom the standards were not met were marked as “Unknown.”

The analysis performed in this paper was more statistical than in the Jury Sunshine Data. Contingency tables generated using the data were tested using Chi-squared independence tests, and a simple logistic regression model was created to predict prosecutorial strikes. One minor criticism which could be made of their methodology is the lack of a consistent level to their tests. It seems that rather than class these tests as significant or not, these tests were simply performed to report the p-values they returned. Additionally, there are possible multiple testing issues as the study seems to indicate multiple tests were performed on each table, with the specific test used to generate the reported p-values not clearly indicated.

3.2.2 Cleaning

3.3 Philadelphia Data

3.3.1 Methodology

[Baldus et al. \(2001\)](#) presents a similar data set collected using similar means. Court files such as the juror questionnaire, voter registration, and census data were all used to complete juror demographic information for 317 venires consisting of 14,532 venire members in Philadelphia capital murder cases between 1981 and 1997⁴. It should be noted that this data included only those jurors kept or peremptorily struck, venire members struck for cause were not included in the data. The procedure used to determine race using the census and voter registration polls was quite complicated, but was rigorously performed using accepted census methods to a standard of 98% reliability⁵.

In their incredibly thorough analysis of the data, there were findings consistent with both the Jury Sunshine and Stubborn Legacy data. The defence and prosecution seem to follow mirrored patterns of racial preference in the use of peremptory challenges, even when controlling for other possible confounding effects.

3.3.2 Cleaning

⁴This study took into account the sampling error by reweighting venires based on the year of the trial and the defendant race, as court records showed that the sample coverage varied over these factors

⁵Additionally, imputation was only performed in a small minority of cases

Chapter 4

Analysis

With this data cleaned and processed, questions can now be posed and addressed through analysis. A few obvious questions come to mind, considering the previous work done on this subject and the modern controversy surrounding it. First, there is the obvious question of not only the possible racial imbalance of peremptory challenge use, but how this imbalance changes with the race of the defendant. In the Gerald Stanley trial, for example, the critical aspect of the trial was not the use of peremptory challenges in abstract, but how their use interacted with the race of Stanley.

Aside from these investigations, we may wonder whether the most common arguments posed in favour of peremptory challenge are satisfied in this data. As discussed in 2.4, there are two primary arguments. The first is the argument that the peremptory challenge is necessary to remove the “extremes of partiality” present in the venire for both sides, that is to remove the most extremely biased jurors. This goal is complemented by the ability of the judge to remove jurors with cause, which is also designed to remove those jurors with extreme bias. The second argument is the creation of a jury which is mutually acceptable to both parties in the trial.

4.1 Extremes of Partiality

While creating a quantitative judgement on the acceptability of a jury is somewhat difficult, measuring the extremality or abnormality of observations is a critical function of statistics. With this in mind, a very simple calculation was performed. The central claim of the advocates of the use of peremptory challenge is that it is only used to remove extreme cases of bias. If that is so, then the proportion of venire members removed by peremptory challenge should reflect this concept.

Of course, this cannot be rigorously tested, as there is no way of knowing the true distribution of bias among jurors. That does not mean nothing can be said, however. As [Nisbett and Kunda \(1985\)](#) notes, there is a tendency of people to guess that a distribution is normal when asked to guess the distribution of social attitudes¹. Additionally, math-

¹This problem is not helped by the notoriety of the normal distribution, as it is commonly the distribution used when performing tests (likely due to the utility of the Central Limit Theorem) and generating visualizations of a general distribution

Table 4.1: The implied statistical extremity bound for symmetric rejection in the datasets given different distributional assumptions

Data	Rejection Rate	Normal	Chebyshev Limit
Sunshine Stubborn Philadelphia	0.434	0.781	1.517

ematical constraints such as the Chebyshev inequality (see [Weisstein \(2018\)](#)) provide an upper limit to the dispersion of any distribution.

This study suggests that it would not be unreasonable to view the overall causal and peremptory challenge rates as the tails of a normal distribution, and the Chebyshev limit gives an estimate of the extremality of rejections given a maximally dispersed distribution of opinions. Table 4.1 provides a summary of the rejection rates of the different data sets and the implied standard distance from the centre that these imply for symmetric rejection.

Obviously, it is not possible to comment with authority on the presence of partiality in the population. Indeed, given the large divide that appears to be present politically in the United States and the rest of the Western world today, it may be easier to argue for a maximally spread distribution than a centralized one. Regardless of this difficulty, it is difficult to justify that 43% of a dataset is “extreme” statistically. In the normal case, this suggests that the rejection boundary is less than one standard deviation from the mean, i.e. that the typically sampled point will be too extreme. The Chebyshev limit is not much better, suggesting that the rejection boundary is at most 1.5 standard deviations from the mean in either direction.

This low rejection boundary given any distribution suggests that the peremptory challenge is not simply being used to remove “extremes of partiality.” Rather, it seems that the argument used to support and justify the practice cannot be reconciled well with the data, suggesting systemic over-use relative to its supported use. This leads naturally to the question of how exactly this legal instrument is over-used, and why.

4.2 The Impact of Race

The racially-motivated controversy surrounding peremptory provide one possible route to investigate the pattern of peremptory strike over-use. To begin, a simple marginal investigation was performed to explore the impact of the simplified race on the peremptory strike probability. The result of this investigation is displayed in Table 4.2. Of particular interest is whether any race is far more likely to be struck by peremptory challenge than the others, as this might suggest that race is the target of the over-use of strikes.

The differences in these probabilities are significant at $\alpha = 0.05$ by merit of the size of the Sunshine data set, but an effect size of 2% is hardly relevant if one considers that the size of each empanelled jury is only 12,

Table 4.2: The conditional probability of a venire member being struck peremptorily by the simplified venire member race across data sets

Venire Member Race	Sunshine	Stubborn	Philadelphia
Black	0.23		
Other	0.24		
White	0.25		

4.3 Developing an Effective Visualization of Conditional Probability

One deficiency of the results of the previous investigations was a failure to generate compelling and effective visualizations of the trends of peremptory challenges for different racial groups. While such visualizations are not necessarily critical to analysis, they can often be incredibly useful to not only communicate data, but to motivate further investigations and models in a way which is clearer and more intuitive than a simple table of values.

The first attempt at such a visualization was the mosaic plot (as discussed by [Friendly \(1994\)](#)) using the `mosaicplot` function in the `graphics` package in R ([R Core Team \(2018\)](#)). Figure 4.1 displays this first approach with disposition related to the simplified races of both the defendant and the venire member.

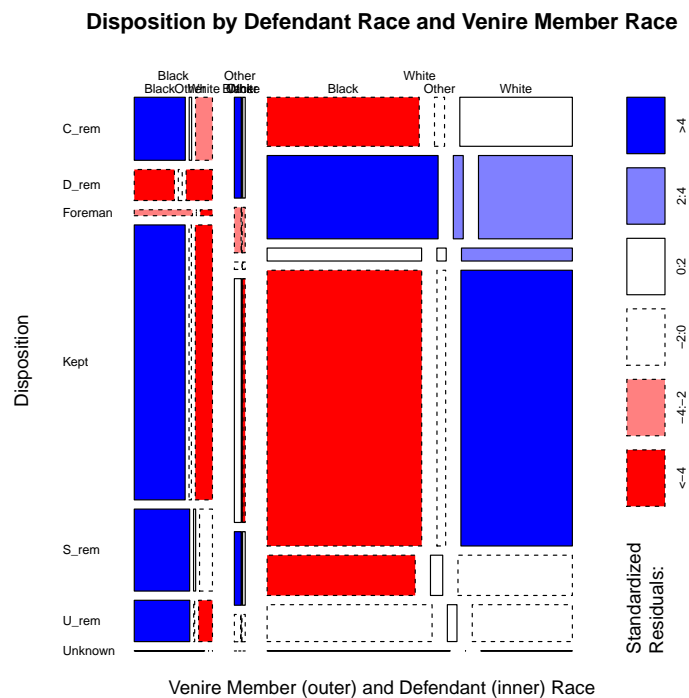


Figure 4.1: A mosaic plot of the simplified defendant and venire member race and their relation to the disposition of the venire member.

This visualization suffers from a number of limitations, some of which are obvious, and others of which are best explained by the hierarchy of accuracy of visual perception provided in [Cleveland and McGill \(1987\)](#). The obvious limitations are the lack of ability to perceive the differences for the smallest groups, which are compressed enough that their error is nearly imperceptible. Additionally, the ordering of the axes is incredibly important in how the different areas appear visually, and comparing the different areas is unclear if any specific comparisons are to be made.

This may be somewhat unsurprising. [Cleveland and McGill \(1987\)](#), in their ranking of visual displays by accuracy of perception place area low in the hierarchy, below angles, lengths, and positions along common scales. In *The Visual Display of Quantitative Information*, [Tufte](#) gives two more sources of possible criticism of the mosaic plot as displayed in Figure 4.1: the concept of data-ink and the dimensionality of visualization.

Of the mosaic plot, one may ask how much of the “ink”, or structure, on the page is necessary to communicate the information present. If one has a desire to “above all else show the data” as Tufte does, then these large shaded rectangles, which are likely not perceived accurately according to [Cleveland and McGill](#), seem unnecessary compared to a simpler visualization. This is the concept of “data-ink,” to reduce the complexity of the structures and chart used to display the data.

Hand-in-hand with this concept for this plot is [Tufte’s](#) rule that the dimensionality of the visualization should not be larger than the data. In the case of the mosaic plot this is not strictly violated, as the marginal lengths used to create the areas reflect a measurement of the data. Nonetheless, the areas of each rectangle correspond to a simple count in a contingency table, and perhaps an area is not the best way to represent such a singular value.

Motivated by these concepts, parallel coordinates (as in [Wegman \(1990\)](#)) were used to visualize the data next, as can be seen in Figure 4.2. This attempted visualization is arguably more difficult to interpret than the mosaic plot. It is cluttered by the parallel coordinate lines, the bars emanating from each point obscure the fact that the end point of the bar is the only feature of interest, and the meaning of the black reference line is entirely unclear without extensive explanation. Finally, by viewing the distribution of each disposition, the wrong conditional density is being examined, $P(\text{Race}, \text{RaceDefendant} | \text{Disposition})$. Multiple edits and re-conceptualizations of the concept eventually resulted in Figure 4.3, which will be called the “mobile plot” due to its passing resemblance to the mobiles hung above babies’ cribs.

An explanation of the features and encoding used in the mobile plot is presented in 4.3.1.

4.3.1 The Mobile Plot

The mobile plot consists of multiple grouped vertical lines anchored at one end to horizontal black lines, and at the other to points. Information is encoded using length, colour, and position relative to a common scale. The vertical axis is meant to show the value of a continuous variable, while the horizontal axis shows the value of a, possibly grouped, categorical variable. It can be used to display the relationship between three categorical variables and a continuous variable in a meaningful two-dimensional plot.

To show the grouping of categories on the horizontal axis, position is used. Those cate-

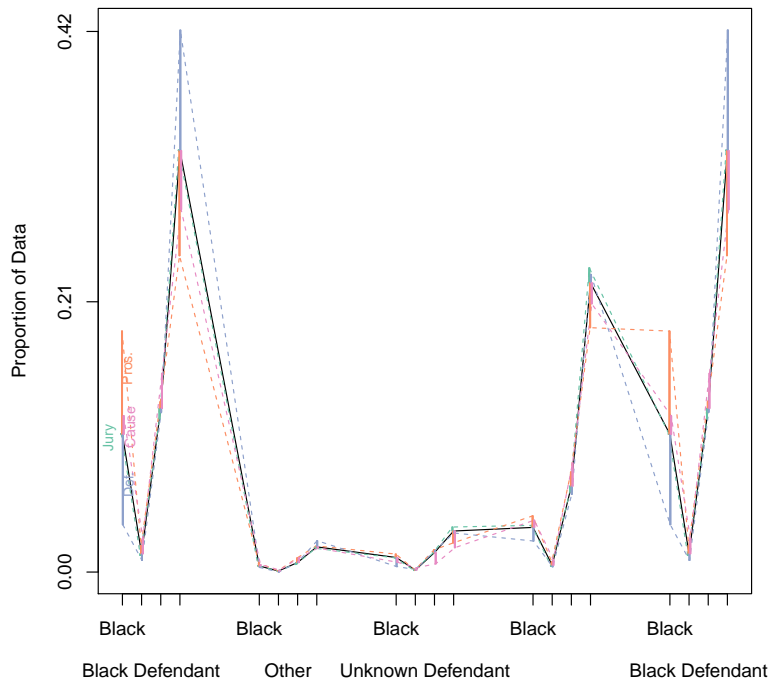


Figure 4.2: The first attempt at a parallel coordinate plot attempted. Note that the cramped display and unclear definition of the axis make interpretation even less intuitive than the mosaic plot, suggesting that this first attempt was a decided failure.

gorical levels which are grouped by some separate categorical variable are placed closer to each other than those which are not in the same group. Each categorical level

4.4 Case Level Summary

While [Wright et al. \(2018\)](#) reported a great deal of aggregate statistics about the venire members themselves, one piece of investigation which was lacking was an analysis which aggregated and viewed the trends for the cases, rather than simply for individual venire members. As we cannot know why a potential venire member is struck individually, and viewing their aggregate statistics tells us nothing about how different strikes relate to each other, it is possible we are viewing some effect which is not a result of persistent bias across trials, but is rather the result of some other effect.

By aggregating the venire members by trial and viewing the demographic trends in strikes and behaviour at this level, we gain a more detailed insight into the impact of challenges at a more relevant scale. Additionally, such aggregation allows for the synthesis of certain measures, such as a disitributional difference via the Kullback-Leibler divergence ([Kullback and Leibler \(1951\)](#)), which would otherwise not be well defined. This particular perspective of the data has also not been explored by any other studies known to the author.

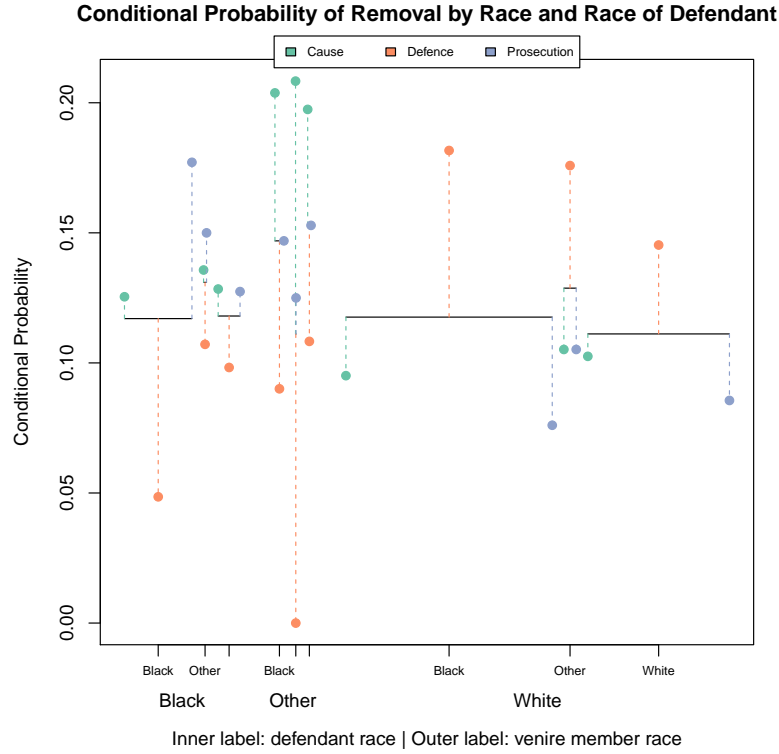


Figure 4.3: The “mobile plot” to display conditional distributions. Note that this plot is less cluttered than either the mosaic plot or the first parallel coordinate plot, despite displaying more information. It is also more efficient with data-ink, avoids displaying data with higher dimensions than the data itself, and uses redundant encoding of information in visual cues which are high in the hierarchy presented by [Cleveland and McGill \(1987\)](#).

4.5 Modelling

In order to create a single model to test the statistical significance of the differences observed for strike rates by race, defendant race, and party doing the striking, a saturated poisson regression model was fit to the data. Letting i denote the level of the venire member race, j the defendant race, and k the disposition, the numbers of observed venire members in each ijk combination, y_{ijk} were modelled as Poisson-distributed random variables with expectation λ_{ijk} . A saturated model was then fit to the data, that is a model described by the equation:

$$\log E[y_{ijk}] = \mathbf{x}_{ijk}\beta = \beta_o + \beta_R x_{i..} + \beta_D x_{.j.} + \beta_S x_{..k} + \beta_{R:D} x_{i..x.j.} + \beta_{R:S} x_{i..x..k} + \beta_{D:S} x_{.j.x..k} + \beta_{R:D:S} x_{i..x.j.x..k} \quad (4.5.0.1)$$

Where $x_{i..}$ indicates the race level of the ijk cell, and $x_{.j.}, x_{..k}$ are defined analogously for the defendant race and disposition. The interaction terms then serve to answer questions about the racial pattern of strikes which is utilized by each party given the defendant race. Most interesting to this investigation is the third order interaction term. This term indicates a significant difference in racial strike patterns given the party striking and the

defendant race. In other words, this term accounts for different patterns for the different parties which are not independent of the defendant race.

When this term is tested using a nested model without the third order interaction, the third order interaction is found to be significant. This suggests that not only do the patterns present in the different parties vary, but they vary differently for different defendant races. This dependence can be viewed using a novel graphic presented in Figure 4.4.

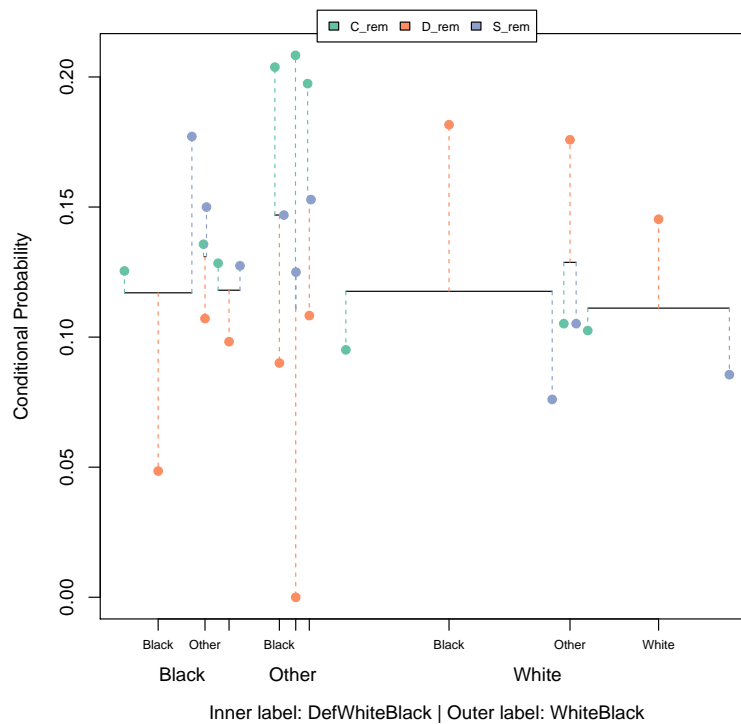


Figure 4.4: Parallel coordinate plot of racial strike tendencies

The conditional probability of a particular disposition given the racial combination of venire person and defendant is displayed on the y-axis, that is the count of individuals for a particular race, defendant race, and disposition combination divided by the number of individuals with the racial combination across all dispositions. The x-axis then displays the combinations, grouped by the venire member race to show the dominant pattern in the data.

The black line running across the plot is the mean, or expected, rejection probability that all parties would have if they acted identically. That is, the relative level of this line provides the relative strike rate on aggregate for a particular racial combination. The bars extending from this line at each point go from this line to the corresponding value of the party represented by the bar. Finally, the horizontal lines provide approximate confidence intervals for each combination².

The dominant pattern to these strikes is a tendency of the defense to preferentially reject

²Generated assuming a binomial distribution of struck (by any party) against kept, as when this data is modelled with a poisson distribution, the distribution of sub-processes given the overall count will be binomially distributed

white venire members and keep black venire members, and of the prosecution to do the opposite. It was already noted in the literature [Wright et al. \(2018\)](#), but the addition of defendant race allows us to make a stronger statement, as this pattern remains across defendant races. It also adds nuance, however, as the race of the defendant has a clear impact on the lengths of the bars for both the defense and prosecution. The prosecution seems to favour a jury which does not match the race of the defendant, while the defense seems to favour a jury which does.

While this second tendency seems to have no justification beyond race, the dominant tendency may have other justification than simply skin colour. As was noted by “Ideological Imbalance and Peremptory Challenge”, black individuals are more consistently aligned with the democratic party, and as a consequence a lawyer which suspects this political bias will impact the trial outcome would preferentially strike or keep black jurors in order to keep as many left wing individuals as possible. In this data, this political imbalance is incredibly prevalent, as can be seen in Figure 4.5 [Add the plot of this effect here, elaborate on this pattern more based on the plot.](#)

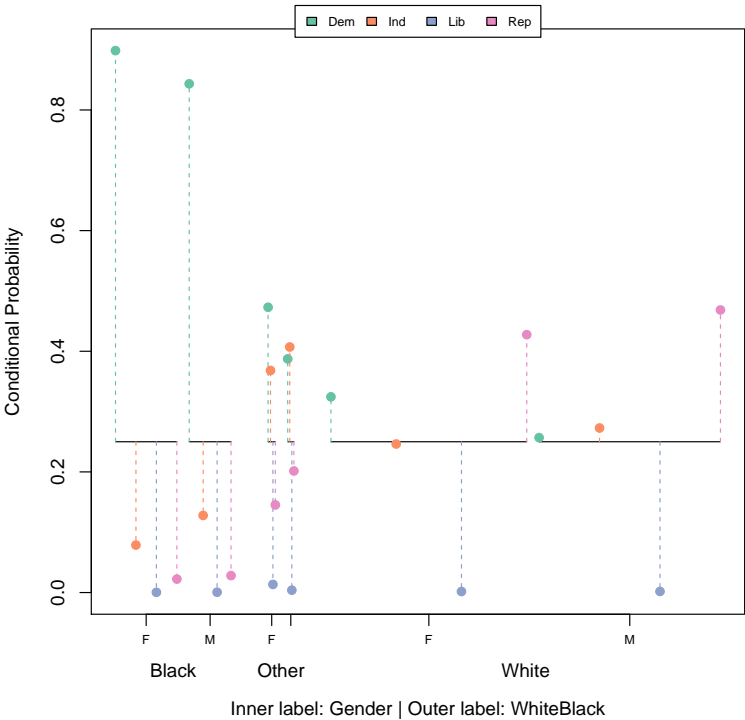


Figure 4.5: Conditional probabilities of political affiliation by race and gender

Perhaps more interestingly, the prosecution and judge seem to match in their tendency from the mean at every combination. This suggests that both challenges with cause and the prosecution tend to have the same effect on the jury composition, though the magnitudes can differ greatly for these two strikes. An immediate explanation to this is offered by [Hans and Vidmar \(1986\)](#), who outline, on pages 69-70, the skill and tact required to effectively propose challenges with cause. In order to determine an individual’s bias, it is frequently the case that a direct question will fail to garner an honest reponse due to social pressures. As a consequence, the questions asked of venire members must be carefully presented.

Using this as a motivation, an obvious possible explanation for the challenges with cause is that the prosecution is simply more experienced on average than the defence. To determine the veracity of this claim, the year licensed for each lawyer was subtracted from the outcome date of each trial. The resulting distribution of years of experience was then plotted in back-to-back histograms as shown in Figure 4.6.

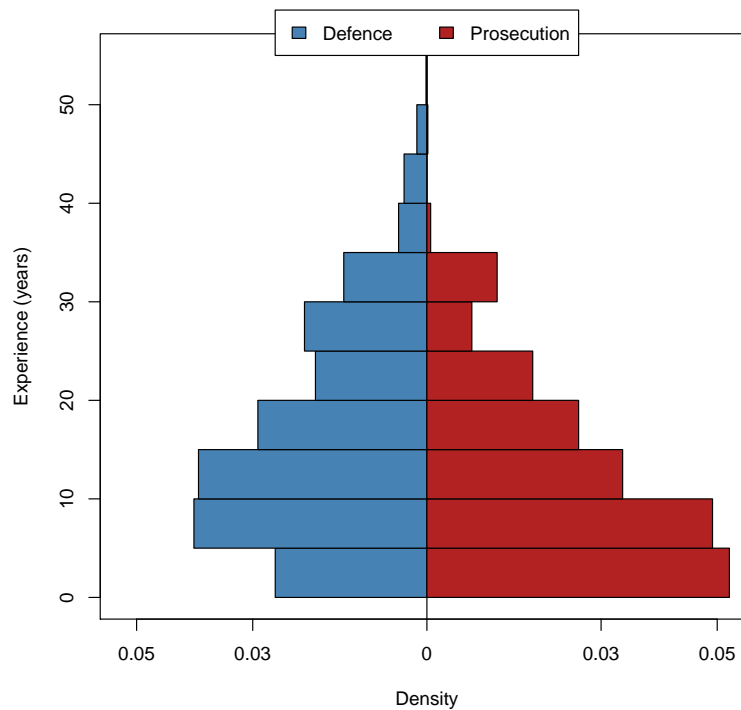


Figure 4.6: Distributions of lawyer experience for prosecutors and defence attorneys

Clearly, this hypothesis is false. It seems the typical defence lawyer is more experienced than the typical prosecutor, not less. Indeed, the prosecutors seem to be much more likely to be inexperienced than the defence lawyers.

Chapter 5

Summary

Summarize the presented work. Why is it useful to the research field or institute?

5.1 Future Work

Possible ways to extend the work.

Bibliography

- 42nd Parliament of Canada (2018a, March). Bill C-75: An Act to Amend the Criminal Code, Youth Criminal Justice Act and other Acts and to make consequential amendments to other Acts. <http://www.justice.gc.ca/eng/csj-sjc/pl/charte-charte/c75.html>.
- 42nd Parliament of Canada (2018b, November). Bill C75. LEGISinfo. <http://www.parl.ca/LegisInfo>.
- Attorney General's Office of the United Kingdom (2012, November). Jury vetting right of stand-by guidelines. <https://www.gov.uk/guidance/jury-vetting-right-of-stand-by-guidelines-2>.
- Baldus, D. C., G. Woodworth, D. Zuckerman, and N. A. Weiner (2001). The Use of Peremptory Challenges in Capital Murder Trials: A Legal and Empirical Analysis. *University of Pennsylvania Journal of Constitutional Law* 3(1).
- Brown, F. L., F. T. McGuire, and M. S. Winters (1978). The peremptory challenge as a manipulative device in criminal trials: Traditional use or abuse. *New England Law Review* 14, 192.
- Brown, R. B. (2000). Challenges for cause, stand-asides, and peremptory challenges in the nineteenth century. *Osgoode Hall Law Journal* 38(3), 453–494.
- Burgess, G. (1992). The divine right of kings reconsidered. *The English Historical Review* 107(425), 837–861.
- Cleveland, W. S. and R. McGill (1987). Graphical perception: The visual decoding of quantitative information on graphical displays of data. *Journal of the Royal Statistical Society. Series A (General)* 150(3), 192–229.
- Constitution of Canada (1982). Constitution of Canada. Accessed: <https://laws-lois.justice.gc.ca/eng/Const/index.html>.
- Constitution of the United States (1788). Constitution of the United States. Accessed: https://www.senate.gov/civics/constitution_item/constitution.htm.
- Ford, R. (2010). Modeling the effects of peremptory challenges on jury selection and jury verdicts. *George Mason Law Review* 17, 377.
- Forsyth, W. (1994). *History of Trial by Jury* (2 ed.). Lawbook Exchange.
- Friendly, M. (1994). Mosaic plots for multiway contingency tables. *Journal of the American Statistical Association* 89, 190–200.
- Government of Canada (1985). Criminal code section 634. <https://laws-lois.justice.gc.ca/eng/acts/C-46/section-634.html>.

- Government of Saskatchewan (1998). Jury act, 1998. Accessed: <http://www.qp.gov.sk.ca/documents/English/Statutes/Statutes/J4-2.pdf>.
- Grosso, C. M. and B. O'Brien (2012). A Stubborn Legacy: The Overwhelming Importance of Race in Jury Selection in 173 Post-Batson North Carolina Capital Trials. *Iowa Law Review* 97, 1531.
- Hans, V. P. and N. Vidmar (1986). *Judging the Jury* (1 ed.). Plenum Press.
- Harris, K. (2018, February). Liberals review jury selection process after Boushie case uproar. CBC News. <https://www.cbc.ca/news/politics/jury-selection-diversity-indigenous-1.4531792>.
- Hasan, N. R. (2018, April). Eliminating peremptory challenges makes trials less fair. The Star. <https://www.thestar.com/opinion/contributors/2018/04/10/eliminating-peremptory-challenges-make-trials-less-fair.html>.
- Hoffman, M. B. (1997). Peremptory Challenges Should Be Abolished: A Trial Judge's Perspective. *The University of Chicago Law Review* 64(3), 809.
- Iacobucci, F. (2013). First Nations Representation on Ontario Juries: Report of the Independent Review Conducted by the Honourable Frank Iacobucci. Ministry of the Attorney General. Accessed: https://www.attorneygeneral.jus.gov.on.ca/english/about/pubs/iacobucci/First_Nations_Representation
- Kullback, S. and R. Leibler (1951). On information and sufficiency. *The Annals of Mathematical Statistics* 22(1), 79–86.
- MacLean, C. (2018, February). Gerald Stanley acquittal renews calls for justice reform 27 years after Manitoba inquiry. CBC News. <https://www.cbc.ca/news/canada/manitoba/aboriginal-justice-inquiry-coltan-boushie-gerald-stanley-jury-1.4532394>.
- Macnab, A. (2018, February). Stanley acquittal should not lead to scrapping peremptory challenges, say criminal lawyers. Canadian Lawyer. <https://www.canadianlawyermag.com/legalfeeds/author/aidan-macnab/stanley-acquittal-should-not-lead-to-scrapping-peremptory-challenges-say-criminal-lawyers-15332/>.
- Ministry of the Attorney General of Ontario (2018). The annual jury selection process. Queen's Printer for Ontario. Accessed: https://www.attorneygeneral.jus.gov.on.ca/english/courts/jury/jury_selection_process.php.
- Miriam-Webster (2019a). Miriam-Webster Dictionary Online. Accessed: <https://www.merriam-webster.com/dictionary/venire>.
- Miriam-Webster (2019b). Miriam-Webster Dictionary Online. Accessed: <https://www.merriam-webster.com/dictionary/voir%20dire>.
- Morehead, J. W. (1994). When a peremptory challenge is no longer peremptory: Batson's unfortunate failure to eradicate invidious discrimination from jury selection. *DePaul Law Review* 43, 625.
- Nisbett, R. E. and Z. Kunda (1985). Perception of social distributions. *Journal of Personality and Social Psychology* 48(2), 297–311.

- North Carolina Sentencing and Policy Advisory Commission (2017). *Classification of a Sample of Offenses*. North Carolina Judicial Branch. Accessed: <https://www.nccourts.gov/assets/documents/publications/Sample-list-2017.pdf?MAZluXuS0FWod4nFt7zXecJ8Ifu0qEZx>.
- Page, A. (2005). Batson's blind spot: Unconscious stereotyping and the peremptory challenge. *Boston University Law Review* 85, 155.
- Petersen, C. (1993). Institutionalized racism: The need for reform of the criminal jury selection process. *McGill Law Journal* 38(1).
- Quenneville, G. (2018, February). What happened on Gerald Stanley's farm the day Colten Boushie was shot, as told by witnesses. CBC News. <https://www.cbc.ca/news/canada/saskatoon/what-happened-stanley-farm-boushie-shot-witnesses-colten-gerald-1.4520214>.
- Quenneville, G. and J. Warick (2018, February). Shouts of 'murderer' in courtroom after Gerald Stanley acquitted in Colten Boushie shooting. CBC News. <https://www.cbc.ca/news/canada/saskatoon/gerald-stanley-colten-boushie-verdict-1.4526313>.
- R Core Team (2018). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.
- Richardson Oakes, A. and H. Davies (2016). Justice must be seen to be done: a contextual reappraisal. *Adelaide Law Review* 37(2), 461–494.
- Roach, K. (2018, April). Ending peremptory challenges in jury selection is a good first step. The Ottawa Citizen. <https://ottawacitizen.com/opinion/columnists/roach-ending-peremptory-challenges-in-jury-selection-is-a-good-first-step>.
- Sheppard, R. (2018). Patriation of the constitution. The Canadian Encyclopedia: Historical Canada. Accessed: <https://www.thecanadianencyclopedia.ca/en/article/patriation-of-the-constitution>.
- Statistics Canada (2018, November). Table 35-10-0061-01: Crime severity index and weighted clearance rates, police services in Saskatchewan.
- Supreme Court of Canada (1991). *R. v. Sherratt*. Supreme Court Judgments. SCC Case Number: 21501; Accessed: <https://scc-csc.lexum.com/scc-csc/scc-csc/en/item/734/index.do?q=21501>.
- Supreme Court of the United States (1965). *Swain v. Alabama*. Accessed: <https://supreme.justia.com/cases/federal/us/380/202/>.
- Supreme Court of the United States (1986). *Batson v. Kentucky*. Accessed: <https://www.law.cornell.edu/supremecourt/text/476/79>.
- Supreme Court of the United States (1993). *J.E.B. v. Alabama*. Accessed: <https://supreme.justia.com/cases/federal/us/511/127/>.
- Tufte, E. R. (2001). *The Visual Display of Quantitative Information* (2 ed.). Graphics Press.
- Van Dyke, J. M. (1977). *Jury Selection Procedures: Our Uncertain Commitment to Representative Panels* (1 ed.). Ballinger Publishing.

- von Moschzisker, R. (1921). The historic origin of trial by jury. *University of Pennsylvania Law Review* 70(1).
- Wegman, E. J. (1990). Hyperdimensional analysis using parallel coordinates. *Journal of the American Statistical Association* 85(411), 664–675.
- Weisstein, E. W. (2018). Chebyshev inequality. MathWorld - A Wolfram Web Resource. Accessed: <http://mathworld.wolfram.com/ChebyshevInequality.html>.
- Wickham, H. and J. Bryan (2018). *readxl: Read Excel Files*. R package version 1.1.0.
- Woolley, A. (2018). An Ethical Jury? Reflections on the Acquittal of Gerald Stanley for the Murder/Manslaughter of Colten Boushie. *Slaw: Canada's online legal magazine*. Accessed: <http://www.slaw.ca/2018/02/20/an-ethical-jury-reflections-on-the-acquittal-of-gerald-stanley-for-the-murder-manslaughter-of-colten-boushie/>.
- Wright, R. F., K. Chavis, and G. S. Parks (2018, October). The Jury Sunshine Project: Jury Selection Data as a Political Issue. *University of Illinois Law Review* 2018(4), 1407.
- Zinchuk, B. (2018, March). Both sides wrong about Stanley trial. Prince George Citizen. <https://www.princegeorgecitizen.com/opinion/editorial/both-sides-wrong-about-stanley-trial-1.23199321>.

Appendix A

Complementary information

Additional material. For example long mathematical derivations could be given in the appendix. Or you could include part of your code that is needed in printed form. You can add several Appendices to your thesis (as you can include several chapters in the main part of your work).

A.1 Including R code with verbatim

A simple (rather too simple, see ??) way to include code or *R* output is to use `verbatim`. It just prints the text however it is (including all spaces, “strange” symbols,...) in a slightly different font.

```
## loading packages
library(RBGL)
library(Rgraphviz)
library(boot)
```

```
## global variables
X_MAX <- 150
```

```
    This allows me to put as many s p a c e s as I want.
I can also use \ and ' and & and all the rest that is usually only
accepted in the math mode.
```

```
I can also make as
                many
            line
        breaks as
I want... and
                where I want.
```

A.2 Data Processing Code

However, it is much nicer to use the *listings* package to include R code in your report. It allows you to number the lines, color the comments differently than the code, and so on.

```

1 #####
2
3 ## THESIS DATA PROCESSING SCRIPT
4 ## Christopher Salahub
5 ## Sept 26, 2018
6
7 #####
8
9 ## PACKAGES #####
10 library(readxl)
11 library(tm)
12 library(stringr)
13 library(grid)
14
15
16 ## CONSTANTS #####
17
18 ## start by defining file locations
19 ThesisDir ← "c:/Users/Chris/Documents/ETH Zurich/Thesis/Data"
20 SunshineFile ← paste0(ThesisDir, "/JurySunshineExcel.xlsx")
21 SunshineSheets ← excel_sheets(SunshineFile)
22
23 NorthCarFile ← paste0(ThesisDir,
24                       "/Jury Study Data and Materials/NC Jury Selection Study
25                       Database6 Dec 2011.csv")
26
27 PhillyFile ← paste0(ThesisDir,
28                     "/Voir Dire Data & Codebook/capital_venires.csv")
29
30 ## next the factor level codes as given in the codebook and regularized here
31 ## regularization: - political affiliation "N" replaced with "I" for all entries
32 LevRace ← sort(c("A", "B", "H", "N", "O", "U", "W"))
33 LevGen ← sort(c("F", "M", "U"))
34 LevPol ← sort(c("D", "L", "R", "I", "U"))
35
36 ## create a charge tree with regex nodes to identify and clean charge text
37 chargeTree ← list("rape" = list("statutory", "first|1", "second|2"), "sex(?=.*
38   offense)" = list("first|1", "second|2"),
39   "sex(?=.*offend)" = list("regis", "addr"), "murder" = list("
40     first|1" = list("att"), "second|2" = list("att")),
41   "arson", "firearm" = list("pos", "disch"), "stole" = list("pos
42     "),
43   "mari" = list("pos", "sell|sale", "man", "pwimsd"), "coca" =
44     list("pos", "sell|sale", "man", "pwimsd"),
45   "cs" = list("pos", "sell|sale", "man", "pwimsd"), "hero" =
46     list("pos", "sell|sale", "man", "pwimsd"),
47   "meth" = list("pos", "sell|sale", "man", "pwimsd"),
48   "oxycod" = list("pos", "sell|sale", "man", "pwimsd"), "mass" =
49     list("pos"), "break" = list("enter"),
50   "assa" = list("serious bodily", "female", "strangul", "deadly"
51     , "official"),
52   "larceny" = list("motor", "felon", "merchant"), "false" = list
53     ("pretense"),
54   "driving" = list("impaired"), "kidnap" = list("first|1", "
55     second|2"),
56   "robb" = list("dang"), "burg" = list("first|1", "second|2"), "
57     indec" = list("liber"),
58   "embezz", "manslaughter" = list("inv"), "flee" = list("arrest")
59     ,
60   "abuse|cruelty" = list("child", "anim"), "identity" = list("
61     theft"))
62
63 ## create a list of variables which can sensibly be summarized by trial

```

```

51 TrialVars <- c("TrialNumberID", "DateOutcome", "JudgeID", "DefAttyType", "
    VictimName",
52     "VictimRace", "VictimGender", "CrimeLocation", "PropertyType",
53     "ZipCode.Trials", "StateTotalRemoved", "DefenseTotalRemoved",
54     "CourtTotalRemoved", "JDistrict", "JName", "JRace", "JGender",
55     "JPoliticalAff", "JVoterRegYr", "JYrApptd", "JResCity", "JResZip",
56     "ChargeTxt", "Outcome", "Sentence.FullSunshine", "DefendantID.
        FullSunshine",
57     "DefendantID.DefendantToTrial", "DefRace", "DefGender", "DefDOB",
        "DefAttyID",
58     "DefAttyName", "DCRace", "DCGender", "DCPoliticalAff", "
        DCYrRegVote",
59     "DCYrLicensed", "DCResideCity", "DCResideZip", "ProsecutorID", "
        ProsName",
60     "ProsRace", "ProsGender", "ProsPoliticalAff", "PYrRegVote", "
        PYrLicensed",
61     "PResideCity", "PResideZip", "Guilty", "CrimeType", "DefWhiteBlack
        ")
62
63
64 ## FUNCTIONS #####
65
66 ## Loading and cleaning #####
67 ## create a descriptive merge function for cleaning (essentially a 'merge'
    wrapper)
68 CleaningMerge <- function(x, y, ...) {
69     ## start by creating the merge
70     ## first match arguments
71     MatchCall <- match.call(merge)
72     MatchCall[[1]] <- quote(merge)
73     ## get input names and ensure proper name structure
74     xname <- MatchCall$x
75     if (!is.symbol(xname)) xname <- as.symbol(paste0(xname[[2]], xname[[3]]))
76     yname <- MatchCall$y
77     if (!is.symbol(yname)) yname <- as.symbol(paste0(yname[[2]], yname[[3]]))
78     ## use this to extract suffixes and fix MatchCall
79     MatchCall$suffixes <- paste0(".", c(xname, yname))
80     MatchCall$x <- xname
81     MatchCall$y <- yname
82     ## specify that the match should be an outer join
83     MatchCall$all <- TRUE
84     ## and use this to make a clean local assignment to modify
85     assign(as.character(xname), cbind(x, Diag.x = 1), envir = environment())
86     assign(as.character(yname), cbind(y, Diag.y = 1), envir = environment())
87     ## now evaluate the call
88     Merged <- eval(MatchCall, envir = environment())
89     ## next perform some checks
90     xExpInds <- is.na(Merged$Diag.x)
91     yExpInds <- is.na(Merged$Diag.y)
92     ## remove the diagnostic columns
93     Merged$Diag.x <- NULL; Merged$Diag.y <- NULL
94     ## summarize the diagnostic checks
95     X_nexp <- sum(xExpInds)
96     Y_nexp <- sum(yExpInds)
97     X_missing <- Merged[xExpInds,]
98     Y_missing <- Merged[yExpInds,]
99     ## print the diagnostics
100    cat("Joined ", paste(xname, yname, sep = " and "), " with ",
101        X_nexp, " and ", Y_nexp, " failed matches respectively\n", sep = "")
102    ## return the results, preferentially keeping the data which is present in x
        but missing from y
103    if (X_nexp == 0 & Y_nexp == 0) {
104        Merged
105    } else list(Merge = Merged[!xExpInds,], Xfails = X_missing, Yfails = Y_
        missing)
106 }
107
108 ## a function to identify and perform swaps with user input
109 SimpleSwapper <- function(data, CorrectLevs, auto = FALSE) {
110     ## first match the data to the columns of interest

```

```

111 colInds ← match(names(CorrectLevs), names(data))
112 ## extract the levels of the columns of interest to check if there are any
    potential swaps
113 swapCheck ← all(sapply(1:length(colInds),
114                     function(ind) identical(sort(levels(as.factor(data[,
    colInds[ind]]))),
115                                             sort(CorrectLevs[[ind]])))
116 ## if no swaps are present end this check
117 if (swapCheck) {
118   cat("No errors found, exiting.")
119   return(data)
120 }
121 ## if errors are found, further investigate them
122 ## identify potential rows
123 ## first those which have elements out of place
124 SwapPoss ← sapply(1:length(colInds),
125                   function(ind) !(data[,colInds[ind]] %in% CorrectLevs[[ind]]))
126 ## now rows containing unknown entries
127 Unknown ← sapply(1:length(colInds),
128                  function(ind) data[,colInds[ind]] == "U")
129 ## identify potential swaps by row
130 Swaps ← apply(SwapPoss, 1, function(row) sum(row) > 1)
131 ## identify the potential errors
132 PotErr ← apply(SwapPoss, 1, function(row) sum(row) == 1)
133 ## use the unknowns to account for some errors
134 UnkInd ← apply(Unknown, 1, any)
135 FalErr ← PotErr & UnkInd
136 ## identify the indices to investigate
137 SwapInds ← which(Swaps|FalErr)
138 ErrInds ← which(PotErr & !UnkInd)
139 ## communicate to the user and ask for input
140 cat("There are ", sum(Swaps|FalErr), " swaps to check\n", sep = "")
141 cat("Additionally, it seems there are ", sum(PotErr & !UnkInd), " errors in
    entries\n", sep = "")
142 ## unless automated
143 if (auto) ErrorReturn ← TRUE else ErrorReturn ← as.logical(readline("Return
    the errors? (T/F): "))
144 ## now, if there are possible swaps investigate them
145 if (sum(Swaps|FalErr) != 0) {
146   ## create a temporary storage structure
147   tempRows ← data[SwapInds, colInds]
148   tempRows ← as.data.frame(lapply(tempRows, function(var) levels(var)[as.
    numeric(var)]),
149                             stringsAsFactors = FALSE)
150   ## loop through and populate this
151   for (ii in 1:nrow(tempRows)) {
152     ## inspect the row
153     print(tempRows[ii,])
154     ## suggest corrections, first generate matches
155     candComb ← lapply(tempRows[ii,],
156                      function(el) which(sapply(CorrectLevs,
157                                                function(levs) el %in%
158                                                    levs)))
159     reps ← unlist(lapply(candComb, length))
160     ## now generate all swap combinations
161     candComb[[1]] ← rep(candComb[[1]], each = max(reps[-1]))
162     candComb ← as.data.frame(candComb, row.names = NULL)
163     ## identify rows which contain all indices, in other words those
    valid as swaps
164     compRows ← apply(candComb, 1, function(row) all(1:length(CorrectLevs)
    %in% row))
165     goodComb ← candComb[compRows,]
166     ## clean them up and print them
167     colnames(goodComb) ← NULL
168     rownames(goodComb) ← NULL
169     cat("Potential combinations:\n")
170     print(t(apply(goodComb, 1, order)))
171     ## take user input or automatically determine value
    if (auto) {

```

```

172         if (!any(compRows)) acceptedComb ← 0 else acceptedComb ← 1
173     } else acceptedComb ← as.numeric(readline("Enter a combination choice
        (0 for error, <enter> to accept first): "))
174     ## handle special cases, 0 if a true error has been identified
175     if (identical(acceptedComb,0)) { ## 0 if a true error has been
        identified
176         ErrInds ← c(ErrInds, SwapInds[ii])
177         cat("True error identified, adding ", SwapInds[ii], " to error
            list\n", sep = "")
178     } else { ## the case where a swap has been correctly identified and
        selected, or enter has been pressed
179         ## if enter has been pressed accept the first row
180         if (is.na(acceptedComb)) acceptedComb ← 1
181         ## print recombined row
182         newRows ← tempRows[ii,order(as.matrix(goodComb[acceptedComb,]))]
183         colnames(newRows) ← NULL
184         rownames(newRows) ← NULL
185         cat("Corrected row:")
186         print(newRows)
187         cat("-----\n")
188         ## correct entry
189         tempRows[ii,] ← newRows
190     }
191 }
192 ## fill the data
193 ## first prevent factor level errors
194 data[,colInds] ← lapply(colInds, function(ind) levels(data[,ind])[as.
    numeric(data[,ind])])
195 ## now swap the data
196 data[SwapInds,colInds] ← lapply(1:length(colInds), function(ind) tempRows
    [,ind])
197 ## reconvert back to factors
198 data[,colInds] ← lapply(colInds, function(ind) as.factor(data[,ind]))
199 }
200 ## in either case return the data and errors as specified
201 if (ErrorReturn) {
202     return(list(Data = data, Errors = ErrInds))
203 } else {
204     return(data)
205 }
206 }
207
208 ## now create a function to address the errors possibly identified in the above
function automatically
209 SwapErrorFix ← function(errorData, CorrectLevs) {
210     ## check if we are in the case without errors
211     if (!identical(names(errorData), c("Data", "Errors"))) {
212         cat("No errors\n")
213         return(errorData)
214     } else {
215         ## extract the data and data in error
216         fulldata ← errorData$Data
217         ## get the relevant columns
218         colInds ← match(names(CorrectLevs), names(fulldata))
219         ## go through the specified variables and remove errors
220         fixed ← lapply(1:length(colInds),
221             function(ind) {
222                 var ← fulldata[,colInds[ind]]
223                 var ← levels(var)[as.numeric(var)]
224                 inds ← !(var %in% CorrectLevs[[ind]])
225                 cat(names(CorrectLevs)[ind], ": ", sum(inds),
226                     " errors\n", sep = "")
227                 var[inds] ← "U"
228                 as.factor(var)
229             })
230         ## insert these fixed values
231         fulldata[, colInds] ← fixed
232         ## return this
233         fulldata
234     }

```

```

235 }
236
237 ## write a wrapper to perform this swapping and error correction in one call
238 SwapandError <- function(data, CorrectLevs) {
239   swapped <- SimpleSwapper(data = data, CorrectLevs = CorrectLevs, auto = TRUE)
240   fixed <- SwapErrorFix(errorData = swapped, CorrectLevs = CorrectLevs)
241   fixed
242 }
243
244 ## Variable Synthesis #####
245 ## Kullback-Leibler divergence function
246 kldiv <- function(samp, dist) {
247   ## convert to matrices
248   mat1 <- as.matrix(samp)
249   mat2 <- as.matrix(dist)
250   ## make into proper distributions
251   mat1 <- mat1/rowSums(mat1)
252   mat2 <- mat2/rowSums(mat2)
253   ## take the log ratio
254   logratio <- log(mat1/mat2)
255   ## multiply by correct matrix
256   vals <- mat1*logratio
257   ## take the row sums
258   rowSums(vals, na.rm = TRUE)
259 }
260
261 ## make a text-mining regularization function
262 StringReg <- function(strs) {
263   ## first set everything to lowercase
264   strs <- tolower(strs)
265   ## replace specific patterns (noticed during early tests)
266   strs <- str_replace_all(strs, "b/e|break/enter|b&e|break or enter|b or e|b &/
    or e|b & e", "breaking and entering")
267   strs <- str_replace_all(strs, "controlled substance", "cs")
268   strs <- str_replace_all(strs, "dwi", "driving while impaired")
269   strs <- str_replace_all(strs, "rwdw", "robbery with a deadly weapon")
270   strs <- str_replace_all(strs, "pwisd|pwmsd|pwmsd|pwitd|pwid|pwmisd|pwsod", "
    pwimsd")
271   strs <- str_replace_all(strs, "robbery|rob ", "robbery")
272   strs <- str_replace_all(strs, "bulgary", "burglary")
273   strs <- str_replace_all(strs, "awdw", "assault with a deadly weapon")
274   strs <- str_replace_all(strs, "(?<=[\\sa-z])[0-9]{2,}", "")
275   strs <- str_replace_all(strs, "att ", "attempted ")
276   strs <- str_replace_all(strs, "assult", "assault")
277   strs <- str_replace_all(strs, "marj", "marijuana")
278   ## replace punctuation
279   strs <- gsub("[^[:alnum:][:space:]]'", "", strs)
280   ## return these
281   strs
282 }
283
284 ## create a function to process such a tree structure given a list of strings
285 stringTree <- function(strs, regexTree, inds = 1:length(strs), includeOther = TRUE) {
286   ## identify the sublists, and divide the data
287   sublists <- sapply(regexTree, is.list)
288   ## iterate over unnamed items (leaf nodes)
289   listdiv <- lapply(regexTree[!sublists], function(el) inds[grepl(el, strs, perl
    = TRUE)])
290   names(listdiv) <- unlist(regexTree[!sublists])
291   ## check if there are any sublists
292   if (!any(sublists)) {
293     if (includeOther) listdiv <- c(listdiv, other = list(inds[!(inds %in%
        unlist(listdiv))]))
294     ## in the case of none, treat the object as a list to iterate through
295     listdiv
296   } else {
297     ## otherwise recurse over the branches
298     finlist <- c(listdiv, lapply(names(regexTree)[sublists],
        function(name) stringTree(strs[grepl(name,

```

```

300         strs, perl = TRUE)],
301         regexTree[[name]],
302         inds[grepl(name,
303             strs, perl =
304                 TRUE)],
305         includeOther)))
306     names(finlist)[(length(listdiv) + 1):length(finlist)] ← names(regexTree)[
307         sublists]
308     c(finlist, other = list(inds[!(inds %in% unlist(finlist))]))
309 }
310 }
311
312 ## create a tree depth helper function
313 maxdepth ← function(tree, counter = 1) {
314     max(sapply(tree, function(br) if (!is.list(br)) counter else maxdepth(br,
315         counter + 1)))
316 }
317
318 ## create a function to aggregate a tree as specified above at the desired depth
319 treeAgg ← function(tree, level = 1) {
320     ## first check the max depth of the tree
321     treedepth ← maxdepth(tree)
322     ## compare this to requested aggregation level
323     stopifnot(level <= treedepth)
324     ## aggregate at desired level with a helper function
325     agg ← function(tr, depth = 1) {
326         if (depth == level) lapply(tr, function(el) setNames(unlist(el), NULL))
327         else lapply(tr, function(br) agg(dr, depth + 1))
328     }
329     agg(tree)
330 }
331
332 ## create a crime class aggregation function
333 CrimeClassify ← function(tree, regChar) {
334     crimes ← list()
335     crimes$Sex ← unique(c(unlist(tree[c("rape", "sex(?=.*offense)", "sex(?=.*
336         offend)", "indec")]),
337         tree$other[grepl("sex", regChar[tree$other])]))
338     crimes$Theft ← unique(unlist(tree[c("stole", "embez", "break", "larceny", "
339         robb", "burg", "identity")]))
340     crimes$Murder ← unique(unlist(tree[c("murder", "manslaughter")]))
341     crimes$Drug ← unique(c(unlist(tree[c("mari", "coca", "cs", "hero", "meth", "
342         oxycod")]),
343         tree$other[grepl("para|drug|substance|pwmsd",
344             regChar[tree$other])]))
345     crimes$Violent ← unique(unlist(tree[c("arson", "assa", "abuse|cruelty")]))
346     crimes$Driving ← unique(c(unlist(tree[c("driving")]),
347         tree$other[grepl("hit(?=.*run)|speeding", regChar[
348             tree$other], perl = TRUE)]))
349     crimes
350 }
351
352 ## in order to make the process of pre-processing the data and adding desired
353     columns, place the pre-processing into a
354     ## flexible function and add operations as desired
355     SynCols ← function(data) {
356         ## too busy, synthesize some variables to clearly indicate the results of
357             defense and prosecution selection
358         data$VisibleMinor ← data$Race != "White"
359         data$PerempStruck ← grepl("S_rem|D_rem", data$Disposition)
360         data$DefStruck ← data$Disposition == "D_rem"
361         data$ProStruck ← data$Disposition == "S_rem"
362         data$CauseRemoved ← data$Disposition == "C_rem"
363         ## lets look at which race struck each juror
364         data$StruckBy ← as.factor(sapply(1:nrow(data),
365             function(ind) {
366                 dis ← as.character(data$
367                     Disposition[ind])
368                 if (dis == "S_rem") {
369                     as.character(data$ProsRace

```

```

356                                     [ind])
357                                     } else if (dis == "D_rem") {
358                                         as.character(data$DCRace[
359                                             ind])
360                                     } else "Not Struck"
361                                     )))
362
363     ## create a white black other indicator
364     data$WhiteBlack ← FactorReduce(data$Race, tokeep = c("Black", "White", "U"))
365     data$DefWhiteBlack ← FactorReduce(data$DefRace, tokeep = c("Black", "White",
366         "U"))
367     data$VicWhiteBlack ← FactorReduce(data$VictimRace, tokeep = c("Black", "White",
368         "U"))
369     ## return the data with synthesized columns
370     data
371 }
372
373 ## write functions to process the sentences
374 SentenceProcess ← function(sentencing) {
375     sents ← tolower(sentencing)
376     ## identify sentences in months, years, and days
377     monthsent ← str_extract(sents, "[0-9\\-]+\\s*(?=m)")
378     daysent ← str_extract(sents, "[0-9\\-]+\\s*(?=d)")
379     yearsent ← str_extract(sents, "[0-9\\-]+\\s*(?=y)")
380     ## extract life without parole
381     lwp ← str_extract(sents, "parol[e]*")
382     ## and with parole
383     life ← str_extract(sents, "life")
384     life[!is.na(lwp)] ← NA
385     ## get restitutions
386     resti ← str_extract(sents, "[0-9,]+\\s*(?=restitu)|\\$[0-9,]+")
387     ## get supervised probation
388     supprob ← str_extract(sents, "sup.*pro")
389 }
390
391 ## Summary Functions #####
392 ## make a function to summarize trial jury data
393 JurySummarize ← function(Varnames = c("Disposition", "Race", "Gender", "
394     PoliticalAffiliation")) {
395     ## check if a juror summary object exists already
396     if (!("sun.juror" %in% ls(.GlobalEnv))) {
397         ## first group the data for easy access
398         Juries ← aggregate(sun.swap[, Varnames],
399             by = list(TrialNumberID = sun.swap$TrialNumberID,
400                 JurorNumer = sun.swap$JurorNumber),
401             unique)
402     } else Juries ← sun.juror
403     ## in either case, perform aggregation by trial instance
404     Juries ← aggregate(Juries[, Varnames],
405         by = list(TrialNumberID = Juries$TrialNumberID),
406         function(var) var)
407     ## clean up the names
408     names(Juries)[grepl("Polit", names(Juries))] ← "PolAff"
409     Varnames[4] ← "PolAff"
410     ## now summarize relevant features
411     Summary ← apply(Juries[, Varnames], 1,
412         function(row) {
413             ## get final jury indices
414             disps ← unlist(row$Disposition)
415             foreman ← grepl("Foreman", disps)
416             finJur ← grepl("Foreman|Kept", disps)
417             defStruck ← grepl("D_rem", disps)
418             proStruck ← grepl("S_rem", disps)
419             ## process all variables
420             newrow ← sapply(row,
421                 function(el) {
422                     c(Jury = table(unlist(el)[finJur]),
423                         Venire = table(unlist(el)),
424                         DefRem = table(unlist(el)[
425                             defStruck]),
426                         ProRem = table(unlist(el)[

```



```

419                                     proStruck)))
420                                     })
421     newrow$Disposition ← NULL
422     newrow ← c(unlist(newrow), ForeRace = row$Race[foreman],
423               ForeGender = row$Gender[foreman], ForePol =
424                 row$PolAff[foreman])
425     if (sum(foreman) > 1) {
426       names(newrow)[names(newrow) == "ForeRace1"] ← "
427         ForeRace"
428       names(newrow)[names(newrow) == "ForeGender1"] ← "
429         ForeGender"
430       names(newrow)[names(newrow) == "ForePol1"] ← "
431         ForePol"
432     }
433     newrow
434   })
435   ## perform some clean up
436   longest ← sapply(Summary, length)
437   longest ← which(longest == max(longest))[1]
438   longNames ← names(Summary[[longest]])
439   Summary ← lapply(names(Summary[[longest]]),
440     function(name) unname(sapply(Summary,
441       function(el) el[name])))
442   names(Summary) ← longNames
443   Summary ← lapply(longNames,
444     function(nm) {
445       if (grepl("ForeGender", nm)) {
446         Summary[[nm]] ← factor(Summary[[nm]], levels = 1:3,
447           labels = LevGen)
448       } else if (grepl("ForePol", nm)) {
449         Summary[[nm]] ← factor(Summary[[nm]], levels = 1:5,
450           labels = LevPol)
451       } else if (grepl("ForeRace", nm)) {
452         Summary[[nm]] ← factor(Summary[[nm]], levels = 1:7,
453           labels = LevRace)
454       } else Summary[[nm]]
455     })
456   names(Summary) ← longNames
457   ## return these
458   list(Juries = Juries, Summaries = as.data.frame(Summary))
459 }
460
461 ## a generic simplification method to summarize a vector
462 Simplifier ← function(col, ...) {
463   UseMethod("Simplifier")
464 }
465
466 ## code up methods for the types to be seen
467 Simplifier.default ← function(col, collapse = "") paste0(col, collapse = collapse)
468
469 Simplifier.numeric ← function(col, na.rm = TRUE, trim = 0, ...) mean.default(col,
470   trim = trim, na.rm = na.rm)
471
472 Simplifier.factor ← function(col, collapse = "", ...) paste0(sort(as.character(
473   levels(col)[as.numeric(col)])),
474   collapse = collapse)
475
476 Simplifier.character ← function(col, collapse = "", ...) paste0(sort(col),
477   collapse = collapse)
478
479 ## create a grouping wrapper which does unique aggregation of a data set
480 UniqueAgg ← function(data, by, ...) {
481   ## convert data to a data frame for regularity
482   if (!is.data.frame(data)) data ← as.data.frame(data)
483   ## identify the grouping column by in the data
484   by.groups ← names(data) == by
485   ## provide nice error handling
486   stopifnot(sum(by.groups) > 0)
487   ## first identify which rows are already unique
488   groups ← as.numeric(as.factor(unlist(data[by.groups])))
489   unqRows ← sapply(groups, function(el) sum(groups == el) == 1)

```

```

476   ## consider grouping only the other rows using the unique function
477   enddata <- data[unqRows,]
478   unqdata <- aggregate(data[!unqRows, !by.groups], by = list(data[!unqRows, by.
479     groups]), unique)
479   ## reorder to make sure everything is compatible
480   names(unqdata)[1] <- by
481   unqdata <- unqdata[,match(names(enddata), names(unqdata))]
482   ## now use the Simplifier helper defined above to process these results
483   procddata <- lapply(unqdata, function(col) sapply(col, Simplifier, ...))
484   ## append everything together
485   enddata <- lapply(1:length(enddata),
486     function(n) c(if (is.factor(enddata[[n]])) as.character(
487       enddata[[n]] else enddata[[n]],
488       procddata[[n]]))
487   names(enddata) <- names(data)
488   ## convert to a data frame
489   as.data.frame(enddata)
490 }
491
492 ## a simple helper to convert multiple factor levels into a single 'other' level
493 FactorReduce <- function(vals, tokeep) {
494   chars <- as.character(vals)
495   ## simply replace elements
496   chars[!grepl(paste0(tokeep, collapse = "|"), chars)] <- "Other"
497   chars
498 }
499
500 ## write a function to re-level factor variables to make mosaic plots cleaner
501 MatRelevel <- function(data) {
502   temp <- lapply(data, function(el) if (is.factor(el)) as.factor(levels(el)[as.
503     numeric(el)]) else el)
504   temp <- as.data.frame(temp)
505   names(temp) <- names(data)
506   temp
507 }
508
509 ## another simple processing function to correct NA's given some other identifier
510 and data set
511 FillNAs <- function(dataNAs, filldata, identifier) {
512   ## extract the relevant column indices in a flexible way
513   if (is.null(colnames(filldata))) {
514     relcol <- grepl(identifier, names(filldata))
515   } else relcol <- grepl(identifier, colnames(filldata))
516   ## first identify the relevant rows in the data NAs
517   relRows <- is.na(dataNAs)
518   ## take the relevant rows of the filldata
519   filldata <- matrix(unlist(filldata[relcol]), ncol = sum(relcol))
520   rowfiller <- rowSums(filldata[relRows,])
521   ## return the filled data
522   dataNAs[relRows] <- rowfiller
523   dataNAs
524 }
525
526 ## write a wrapper to estimate the values of total removed jurors
527 RemovedJurorEstimates <- function(tofill, data, ident, plot = TRUE) {
528   temp <- FillNAs(tofill, filldata = data, identifier = ident)
529   temp2 <- rowSums(data[,grepl(ident, names(data))])
530   ## let's see how accurate this is if plotting is desired
531   if (plot) {
532     plot(temp, temp2, xlab = "Observed and Filled", ylab = "Juror Sums")
533     abline(0,1)
534   }
535   cat("= : ", sum(temp == temp2)/length(temp2), "\n", "< : ", sum(temp2 < temp)
536     /length(temp2), "\n", sep = "")
537   ## replace the filled values less than the estimated, for consistency
538   temp[temp < temp2] <- temp2[temp < temp2]
539   temp
540 }

```

```

541 ## LOADING AND PROCESSING DATA #####
542
543 ## load the data
544 SunshineData <- lapply(SunshineSheets, function(nm) as.data.frame(read_excel(
    SunshineFile, sheet = nm)))
545 names(SunshineData) <- SunshineSheets
546 NorthCarData <- read.csv(NorthCarFile)
547 PhillyData <- read.csv(PhillyFile)
548
549 ## clean non-informative columns
550 CleanSunshine <- lapply(SunshineData, function(dat) dat[, !apply(dat, 2, function(
    col) all(is.na(col)))])
551
552 ## the Sunshine data needs to be restructured into one table, rather than a
    relational database structure
553 ## see the IDMatch function, this was created specifically to perform ID-based
    table joins
554 ## the most appropriate global target is the juror table, start by matching this
    to the trial
555 FullSunshine <- with(CleanSunshine, CleaningMerge(Jurors, Trials, by = "
    TrialNumberID"))
556 ## remove extra ID column, fix a misleading name
557 FullSunshine$CountyName <- FullSunshine$CountyID
558 FullSunshine$CountyID <- NULL
559 ## clean up two additional columns which had inconsistencies
560 FullSunshine$Disposition <- toupper(FullSunshine$Disposition)
561 FullSunshine$Race[FullSunshine$Race == "?"] <- "U"
562 ## before appending everything to this table, perform some other joins
563 TrialsToCharge <- with(CleanSunshine, CleaningMerge(Charges, Junction, by = "
    ACISID", all = TRUE))
564 DefendantToTrial <- with(CleanSunshine, CleaningMerge(Defendants, DefendantTrial,
    by = "DefendantID", all = TRUE))
565 AttorneyToTrial <- with(CleanSunshine, CleaningMerge(Attorney, AttorneyTrial, by =
    "DefAttyID", all = TRUE))
566 ProsecutorToTrial <- with(CleanSunshine, CleaningMerge(Prosecutor, ProsecutorTrial
    , by = "ProsecutorID", all = TRUE))
567 ## merge issues:
568 ## - trials to charge: one charge is missing a trial ID, hopefully not
    important
569 ## - prosecutors to trials: 26 prosecutors without trials, however all entries
    were entirely uninformative
570 ## given the above outputs, rename the failed clean merges to make the next
    section cleaner
571 TrialsToCharge <- TrialsToCharge$Merge
572 ProsecutorToTrial <- ProsecutorToTrial$Merge
573
574 ## now perform some additional merges to create one sheet/data.frame
575 ## add the judge descriptions (no issues)
576 FullSunshine <- CleaningMerge(FullSunshine, CleanSunshine$Judges, by = "JudgeID",
    all = TRUE)
577 ## the charges
578 FullSunshine <- CleaningMerge(FullSunshine, TrialsToCharge, by = "TrialNumberID",
    all = TRUE)
579 ## this leads to 22 jurors in trials without charges and 29 charges without
    trials, inspecting these:
580 ## - the jurors without charges are all related to a trial with ID number
    "710-01", thankfully the other data
581 ## for this case is complete, and so it may still be useful for viewing
    jury behaviour
582 ## - the charges without trials are all of the form "710-0xx", suggesting the
    omission of entire trials of some
583 ## relation, hopefully these were not too similar, or this exclusion can be
    explained later
584 FullSunshine <- FullSunshine$Merge
585 ## the defendants
586 FullSunshine <- CleaningMerge(FullSunshine, DefendantToTrial, by = "TrialNumberID"
    , all = TRUE)
587 ## the attorneys
588 FullSunshine <- CleaningMerge(FullSunshine, AttorneyToTrial, by = "TrialNumberID",
    all = TRUE)

```

```

589 ## the prosecutors
590 FullSunshine ← CleaningMerge(FullSunshine, ProsecutorToTrial, by = "TrialNumberID", all = TRUE)
591 ## 26 jurors appear to be lacking a prosecutor, these appear to be the
592 ##   uninformative prosecutors from earlier, included
593 ## due to the preferential inclusion of the missing values in the first of the
594 ##   merged matrices
595 FullSunshine ← FullSunshine$Merge
596
597 ## perform some cleanup
598 ## start with some specific factor replacements
599 ## replace the "N" with "I", as these factor levels are interchangeable in the
600 ##   codebook and prevent confusion with race
601 FullSunshine[,grepl("Pol", names(FullSunshine))] ← lapply(FullSunshine[,grepl("
602   Pol", names(FullSunshine))],
603   function(var) {
604     var ← toupper(var)
605     var[var == "N"] ← "I"
606     var
607   })
608
609 ## next save most variables as factors
610 FullSunshine ← lapply(FullSunshine,
611   function(el) if (is.character(el)) as.factor(el) else el)
612 ## correct some overzealous assignment from above
613 FullSunshine[grepl("Notes", names(FullSunshine))] ← lapply(FullSunshine[grepl("
614   Notes", names(FullSunshine))],
615   as.character)
616 ## perform factor regularization according to the factor levels provided in the
617 ##   codebook
618 FullSunshine ← sapply(FullSunshine,
619   function(el) {
620     if (!is.factor(el)) {
621       el[el == 999] ← NA
622       el
623     } else {
624       el ← as.character(el)
625       el ← toupper(el)
626       el[is.na(el)] ← "U"
627       as.factor(el)
628     }
629   }, simplify = FALSE)
630 FullSunshine ← as.data.frame(FullSunshine)
631 ## remove some unnecessary columns
632 FullSunshine$ID ← NULL
633 FullSunshine$TrialIDAuto ← NULL
634 ## combine the name columns to produce more useful columns
635 FullSunshine$JName ← paste(FullSunshine$JFirstName, FullSunshine$JLastName)
636 FullSunshine$JName[FullSunshine$JName == "U U"] ← "U"
637 FullSunshine$DefAttyName ← paste(FullSunshine$DCFirstName, FullSunshine$
638   DCLastName)
639 FullSunshine$DefAttyName[FullSunshine$DefAttyName == "U U"] ← "U"
640 FullSunshine$ProsName ← paste(FullSunshine$ProsecutorFirstName, FullSunshine$
641   ProsecutorLastName)
642 FullSunshine$ProsName[FullSunshine$ProsName == "U U"] ← "U"
643
644 ## Checkpoint 1: the clean data has been processed, none of the swaps, synthesis,
645 ##   or expansion has taken place
646 ## save this
647 if (!("FullSunshine.csv" %in% list.files())) write.csv(FullSunshine, "
648   FullSunshine.csv", row.names = FALSE)
649 ## load if the desire is to start at checkpoint 1
650 if (!("FullSunshine" %in% ls())) FullSunshine ← read.csv("FullSunshine.csv")
651
652 ## Note: the below swap functions have been set to auto as the function's
653 ##   performance in these cases has already
654 ## been assessed, and so the swaps have already been inspected, it is critical
655 ##   for new data that "auto" be switched
656 ## off to take full advantage of this functionality, and so the wrapper "
657 ##   SwapandError" should not be used

```

```

644 ## in the juror data
645 sun.swapJuror ← SwapandError(FullSunshine, CorrectLevs = list(Race = LevRace,
646                                                                Gender = LevGen,
647                                                                PoliticalAffiliation
                                                                = LevPol))
648 ## in the judge data
649 sun.swap ← SimpleSwapper(sun.swapJuror, CorrectLevs = list(JRace = LevRace,
650                                                                JGender =
651                                                                LevGen,
652                                                                JPoliticalAff
                                                                = LevPol
                                                                ))
653 ## viewing the error report of these data, they are all related to one judge,
654     Arnold O Jones II, who is verified
655 ## as a male after a quick Google search
656 unique(sun.swap$Data[sun.swap$Errors, c("JFirstName", "JLastName")])
657 sun.swapJudge ← sun.swap$Data
658 sun.swapJudge$JGender[sun.swap$Errors] ← "M"
659 sun.swapJudge$JGender ← as.factor(levels(sun.swapJudge$JGender)[as.numeric(sun.
660     swapJudge$JGender)])
661 ## in the prosecutor data
662 sun.swap ← SimpleSwapper(sun.swapJudge, CorrectLevs = list(ProsRace = LevRace,
663                                                                ProsGender =
664                                                                LevGen,
665                                                                ProsPoliticalAff
                                                                = LevPol
                                                                ))
666 ## that found no errors
667 ## a quick check of the levels of the defendant data finds only one error
668 levels(sun.swap$DefGender)
669 levels(sun.swap$DefRace)
670 sun.swap ← SwapandError(sun.swap, CorrectLevs = list(DefRace = LevRace,
671                                                                DefGender = LevGen
                                                                ))
672 ## next the attorney data
673 sun.swap ← SwapandError(sun.swap, CorrectLevs = list(DCRace = LevRace,
674                                                                DCGender = LevGen,
675                                                                DCPoliticalAff =
                                                                LevPol))
676 ## finally the victim data
677 sun.swap ← SwapandError(sun.swap, CorrectLevs = list(VictimRace = LevRace,
678                                                                VictimGender =
                                                                LevGen))
679 ## this leaves the data error-free (in at least the race/gender/politics columns)
680 ## fix the outcome data, which had some improper levels
681 sun.swap$Outcome[sun.swap$Outcome == "HC"] ← "U"
682 sun.swap$Outcome[sun.swap$Outcome == "G"] ← "GC"
683 sun.swap$Outcome ← as.factor(levels(sun.swap$Outcome)[as.numeric(sun.swap$Outcome
684     )])
685 ## lets make the levels more clear for some of the data (race, politics,
686     disposition)
687 ## start with the disposition
688 levels(sun.swap$Disposition) ← c("C_rem", "D_rem", "Foreman", "Kept", "U_rem",
689     "S_rem", "Unknown")
690 ## next the political affiliation
691 sun.swap ← lapply(sun.swap, function(el) {
692     if (is.factor(el) & identical(levels(el), LevPol)) {
693         levels(el) ← c("Dem", "Ind", "Lib", "Rep", "U")
694     } else el})
695 levels(sun.swap$JPoliticalAff) ← c("Dem", "Ind", "Rep", "U")
696 ## now the race
697 sun.swap ← lapply(sun.swap, function(el) {
698     if (is.factor(el) & identical(levels(el), LevRace)) {
699         levels(el) ← c("Asian", "Black", "Hispanic", "NatAm", "Other",
700             "U", "White")
701     } else el})

```

```

700 levels(sun.swap$VictimRace) ← c("Asian", "Black", "Hisp", "NatAm",
701                                "U", "White")
702 levels(sun.swap$JRace) ← c("Black", "Hisp", "NatAm", "U", "White")
703 levels(sun.swap$DCRace) ← c("Asian", "Black", "NatAm", "Other",
704                             "U", "White")
705 ## now the outcome/verdict
706 levels(sun.swap$Outcome) ← c("Acquittal", "Guilty as Charged",
707                              "Guilty of Lesser", "Incomplete", "Mistrial",
708                              "U")
709 ## the defense attorney type
710 levels(sun.swap$DefAttyType) ← c("App Priv", "Public", "Private",
711                                  "Ret Priv", "U", "Waived")
712
713 ## add a guilt indicator
714 sun.swap$Guilty ← grepl("Guilty", sun.swap$Outcome)
715
716 ## add a simple indicator of defendant race matching juror race if they are both
717   known
718 sun.swap$RaceMatch ← sun.swap$Race == sun.swap$DefRace
719 sun.swap$RaceMatch[sun.swap$Race == "U" | sun.swap$DefRace == "U"] ← NA
720
721 ## now perform tree classification of crimes
722 ## first cast sun.swap as a data frame
723 sun.swap ← as.data.frame(sun.swap)
724 ## regularize the charges
725 chargFact ← as.factor(sun.swap$ChargeTxt)
726 regCharg ← StringReg(levels(chargFact))[as.numeric(chargFact)]
727 ## classify these into a charge tree and aggregate this at the coarsest level
728 aggCharg ← treeAgg(stringTree(regCharg, chargeTree))
729 ## these can be further classified into crime classes
730 crimes.trial ← CrimeClassify(aggCharg, regCharg)
731 ## convert these classes into a factor for the data, start with a generic "other"
732   vector
733 sun.swap$CrimeType ← rep("Other", nrow(sun.swap))
734 ## now populate it
735 for (nm in sort(names(crimes.trial))) sun.swap$CrimeType[crimes.trial[[nm]]] ← nm
736 sun.swap$CrimeType ← as.factor(sun.swap$CrimeType)
737
738 ## synthesize additional columns
739 sun.swap ← SynCols(sun.swap)
740
741 ## now organize this on the juror scale
742 sun.juror ← UniqueAgg(sun.swap, by = "JurorNumber", collapse = ",")
743
744 ## Checkpoint 2: the swapped data has been processed and summarized to be on the
745   scale of individual jurors
746 ## save the swapped data
747 write.csv(sun.swap, "FullSunshine_Swapped.csv", row.names = FALSE)
748 ## and the juror summarized data
749 saveRDS(sun.juror, "JurorAggregated.Rds")
750
751 ## summarize by trial, get the unique trials
752 Trials ← unique(sun.swap$TrialNumberID)
753 ## extract information about these trials, note that grouping occurs on the trial
754   ID, defendant ID, and charge ID levels,
755 ## as the trials frequency involve multiple charges and defendants, which makes
756   them less clean
757 sun.trial ← aggregate(sun.swap[, TrialVars],
758                      by = list(sun.swap$TrialNumberID, sun.swap$DefendantID,
759                                .DefendantToTrial,
760                                sun.swap$ID.Charges),
761                      unique)
762
763 sun.trial$Group.1 ← NULL
764 sun.trial$Group.2 ← NULL
765 sun.trial$Group.3 ← NULL
766
767 ## summarize the juries by trial as well
768 sun.jursum ← JurySummarize()
769
770 ## merge the summaries to the trial sunshine data

```

```

764 sun.trialsum <- merge(cbind(TrialNumberID = sun.jursum$Juries$TrialNumberID, sun.
765     jursum$Summaries),
766     sun.trial, all = TRUE)
767 ## notice that the total removed variables are incomplete, try to correct this
768 where possible using the jury
769 ## summarized data above
769 sun.trialsum$DefRemEst <- RemovedJurorEstimates(sun.trialsum$DefenseTotalRemoved,
770     data = sun.trialsum,
771     ident = "Gender.DefRem", plot =
772     FALSE)
773 ## perform this same procedure for the prosecution removals
773 sun.trialsum$ProRemEst <- RemovedJurorEstimates(sun.trialsum$StateTotalRemoved,
774     data = sun.trialsum,
775     ident = "Gender.ProRem", plot =
776     FALSE)
777 ## synthesize some other variables, simple race indicators
777 sun.trialsum$DefWhiteBlack <- as.factor(FactorReduce(sun.trialsum$DefRace, tokeep
778     = c("Black", "White", "U")))
778 sun.trialsum$DefWhiteOther <- as.factor(FactorReduce(sun.trialsum$DefWhiteBlack,
779     tokeep = c("White", "U")))
779 ## the Kullback-Leibler divergence
779 sun.trialsum$KLdiv <- kldiv(sun.trialsum[,grepl("Jury", names(sun.trialsum))],
780     sun.trialsum[,grepl("Venire", names(sun.trialsum))])
781 ## Checkpoint 3: the data has been set to the trial level and summarized
782 ## save this
782 saveRDS(sun.trialsum, "TrialAggregated.Rds")
783 saveRDS(sun.jursum, "AllJuries.Rds")
784
785
786
787 ## CHARGE CLASSIFICATION IMAGES #####
788 ## presented here is the code to generate the appendix charge classification
789 images
790 ## extract relevant charges from the clean sunshine data, avoiding duplicates
791 ## regularize the charges
791 chargFact.clean <- as.factor(CleanSunshine$Charges$ChargeTxt)
792 regCharg.clean <- StringReg(levels(chargFact.clean))[as.numeric(chargFact.clean)]
793 ## classify these into a charge tree and aggregate this at the coarsest level
793 tree.clean <- stringTree(regCharg.clean, chargeTree)
794
795
796
797 ## define padding
797 crimepad <- 0.01
798 areatop <- 0.94
799 hght <- ((0.94 - 0.035) - 5*crimepad)/5
800 wid <- ((0.975 - 0.025) - 7*crimepad)/6
801
802
803 ## add overall box
803 grid.newpage()
804 grid.rect(width = 0.95, height = 0.95)
805 grid.text(label = "Charges", x = 0.025 + crimepad/2, y = 0.975 - crimepad/2, just
806     = c("left","top"))
807 grid.text(label = "(198)", x = 0.975 - crimepad/2, y = 0.975 - crimepad/2, just =
808     c("right","top"))
809
810 ## inner crime boxes
810 xpos <- crimepad + 0.025 + wid/2 + (0:5)*(crimepad + wid)
811 ypos <- crimepad + 0.025 + hght/2 + (0:4)*(crimepad + hght)
812 coords <- cbind(rep(xpos, each = 5), rev(ypos))[1:length(tree.clean)-1,]
813 grid.rect(x = coords[,1], y = coords[,2], width = wid, height = hght)
814 grid.text(label = names(tree.clean)[1:(length(tree.clean)-1)], x = coords[,1] - (
815     wid - crimepad)/2,
816     y = coords[,2] + (hght-crimepad)/2, just = c("left","top"))
817
818 ## add unclassified and top level counts
818 unclass <- sapply(tree.clean, function(e1) if (is.list(e1)) length(e1$other) else
819     0)
819 grid.text(label = paste0("(", unclass, ")"), x = coords[,1] + (wid - crimepad)/2,
820     y = coords[,2] + (hght-crimepad)/2,

```

```

820         just = c("right", "top"))
821 toplev ← sapply(tree.clean, function(el) if (is.list(el)) "" else length(el))
822 grid.text(label = toplev, x = coords[,1], y = coords[,2])
823
824 ## add sublist counts
825 for (ii in 1:length(tree.clean)) {
826   if (is.list(tree.clean[[ii]])) {
827     currlist ← tree.clean[[ii]]
828     len ← length(currlist) - 1
829     boty ← coords[ii,2] - hght/2
830     newwid ← wid - 2*crimepad
831     newhght ← (hght - 0.035 - len*crimepad)/len
832     newy ← crimepad + newhght/2 + (0:(len-1))*(newhght + crimepad) + boty
833     if (names(tree.clean)[ii] == "assa") {
834       grid.rect(x = coords[ii,1], y = newy, width = newwid, height =
835         newhght)
836       grid.text(label = names(currlist), x = coords[ii,1] - (newwid-
837         crimepad)/2, y = newy,
838         just = "left", gp = gpar(fontsize = 8))
839       grid.text(sapply(currlist, length), x = coords[ii,1], y = newy, gp =
840         gpar(fontsize = 8))
841     } else if (names(tree.clean)[ii] == "murder") {
842       grid.rect(x = coords[ii,1], y = newy, width = newwid, height =
843         newhght)
844       grid.text(label = names(currlist), x = coords[ii,1] - (newwid-
845         crimepad)/2, y = newy + (newhght-crimepad)/2,
846         just = c("left","top"), gp = gpar(fontsize = 8))
847       grid.text(paste0("(", sapply(currlist[1:len], function(el) length(el$
848         other)), ")"), x = coords[ii,1] + (newwid-crimepad)/2,
849         y = newy + (newhght-crimepad)/2, just = c("right","top"),
850         gp = gpar(fontsize = 8))
851       grid.rect(x = coords[ii,1], y = newy - 0.01, width = newwid - 2*
852         crimepad, height = (newhght - 0.02)/2)
853       grid.text(rep("att",2), x = coords[ii,1] - (newwid-2*crimepad-
854         crimepad)/2, y = newy-0.01,
855         just = "left", gp = gpar(fontsize = 8))
856       grid.text(sapply(currlist[1:len], function(el) length(el$att)), x =
857         coords[ii,1],
858         y = newy - 0.01, gp = gpar(fontsize = 8))
859     } else {
860       grid.rect(x = coords[ii,1], y = newy, width = newwid, height =
861         newhght)
862       grid.text(label = names(currlist), x = coords[ii,1] - (newwid-
863         crimepad)/2, y = newy + (newhght-crimepad)/2,
864         just = c("left","top"), gp = gpar(fontsize = 8))
865       grid.text(sapply(currlist, length), x = coords[ii,1], y = newy, gp =
866         gpar(fontsize = 10))
867     }
868   }
869 }
870
871 ## a small example tree
872 firstx ← 0.33
873 secondx ← 0.75
874 firsty ← c(0.25,0.75)
875 secondy ← c(0.125,0.375,0.625,0.875)
876 wd ← 0.3
877 hg ← 0.1
878 grid.newpage()
879 grid.rect(x = firstx, y = firsty, width = wd, height = hg)
880 grid.rect(x = secondx, y = secondy, width = wd, height = hg)
881 grid.lines(x = c(0,firstx-wd/2), y = c(0.5,firsty[1]))
882 grid.lines(x = c(0,firstx-wd/2), y = c(0.5,firsty[2]))
883 grid.lines(x = c(firstx+wd/2,secondx-wd/2), y = c(firsty[2],secondy[3]))
884 grid.lines(x = c(firstx+wd/2,secondx-wd/2), y = c(firsty[2],secondy[4]))
885 grid.lines(x = c(firstx+wd/2,secondx-wd/2), y = c(firsty[1],secondy[1]))
886 grid.lines(x = c(firstx+wd/2,secondx-wd/2), y = c(firsty[1],secondy[2]))
887 grid.text(label = c("sex(?.*offend)", "sex(?.*offense)"), x = firstx, y = firsty
888   ,
889   gp = gpar(fontsize = 16))

```



```

876 grid.text(label = c("first|1","second|2","regis","addr"), x = secondx, y = rev(
      secondy),
877          gp = gpar(fontsize = 16))

```

A.3 Jury Sunshine Irregularities

Table A.1: Jury sunshine data irregularities noted in data flattening

Charges without trial (ACISID)	08CRS50940, 08CRS52888, 09CRS000305, 09CRS1106, 09CRS50752, 10CR52031, 10CRS051975, 10CRS1215, 10CRS397, 10CRS51388, 10CRS51610, 10CRS52410, 11CRS051642, 11CRS051795, 11CRS1577, 11CRS1745, 11CRS1783, 11CRS51204, 11CRS51895, 11CRS52470, 08CRS54836, 08CRS50113
Prosecutors without trials (IDs)	1-000, 11B-000, 12-000, 14-000, 15B-000, 16A-000, 16B-000, 17A-000, 17B-000, 19A-000, 19B-000, 20A-000, 20B-000, 21-000, 22A-000, 22B-000, 24-000, 25-000, 27A-000, 27B-000, 28-000, 29A-000, 29B-000, 30-000, 6-000, 9-000
Trial missing charge (ID)	710-01

A.4 Jury Sunshine Charge Classification

A.5 Analysis Code

```

1 #####
2
3 ## THESIS ANALYSIS SCRIPT
4 ## Christopher Salahub
5 ## Sept 26, 2018
6
7 #####
8
9 ## PACKAGES #####
10 library(readxl)
11 library(MASS)
12 library(eikosograms)
13 library(RColorBrewer)
14 library(stringr)
15 library(tm)
16 library(lme4)
17 library(lmerTest)
18
19 ## CONSTANTS #####
20
21 ## start by defining file locations
22 ThesisDir <- "c:/Users/Chris/Documents/ETH Zurich/Thesis/Data"
23 SunshineFile <- paste0(ThesisDir, "/JurySunshineExcel.xlsx")
24 SunshineSheets <- excel_sheets(SunshineFile)
25
26 NorthCarFile <- paste0(ThesisDir,

```

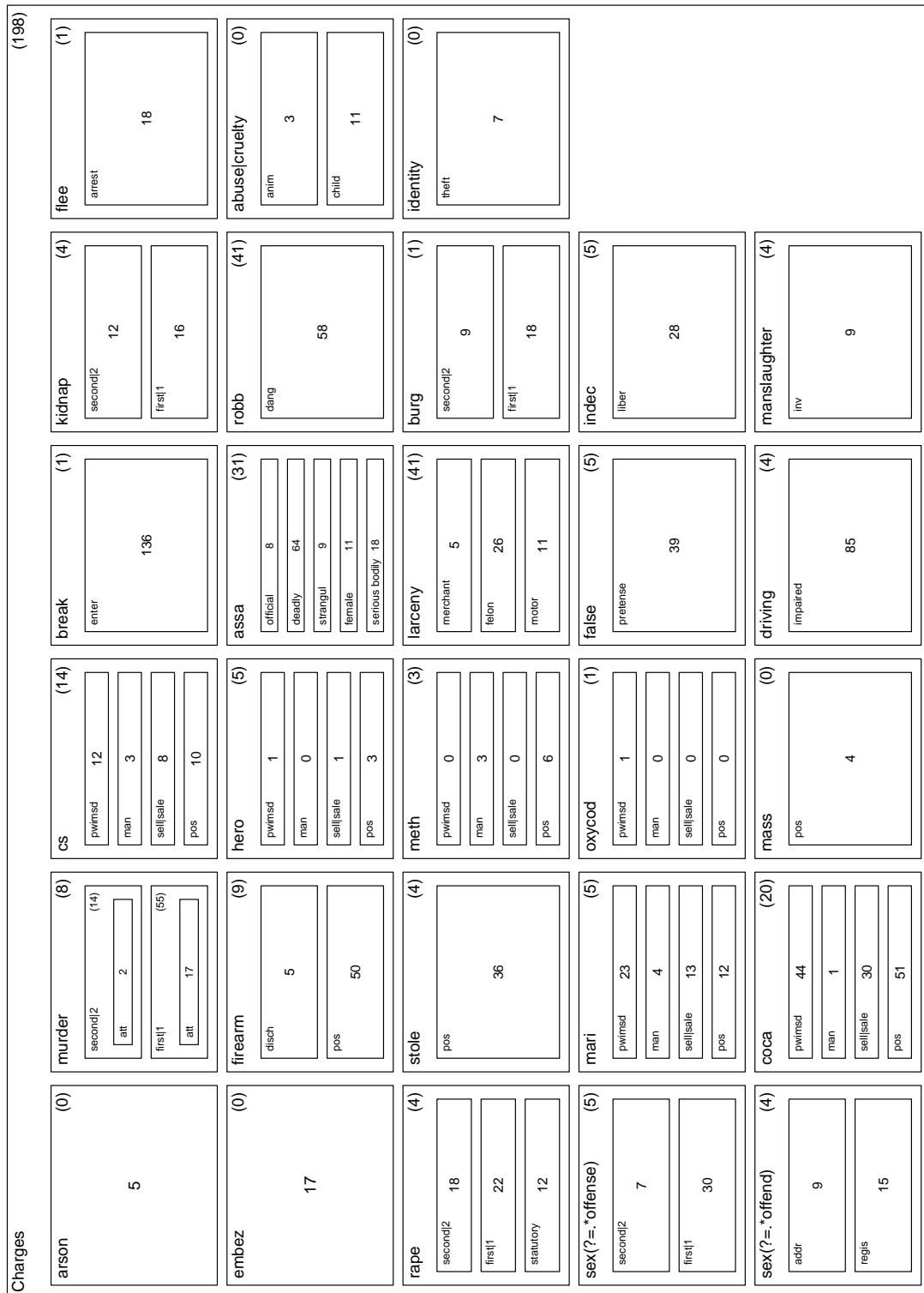


Figure A.1: The regular expression charge tree arranged by hierarchy with counts provided. The counts in brackets indicate the counts of charges which could not be classified to a lower level of the hierarchy

```

27         "/Jury Study Data and Materials/NC Jury Selection Study
          Database6 Dec 2011.csv")
28
29 PhillyFile ← paste0(ThesisDir,
30                      "/Voir Dire Data & Codebook/capital_venires.csv")
31
32 ## next the factor level codes as given in the codebook and regularized here
33 ## regularization: - political affiliation "N" replaced with "I" for all entries
34 LevRace ← sort(c("Asian", "Black", "Hisp", "NatAm", "Other", "U", "White"))
35 LevGen ← sort(c("F", "M", "U"))
36 LevPol ← sort(c("Dem", "Lib", "Rep", "Ind", "U"))
37
38 ## color constants
39 racePal ← brewer.pal(3, "Set2") # c("steelblue", "grey50", "firebrick")
40 whitePal ← c("steelblue", "firebrick")
41 crimePal ← brewer.pal(7, "Set1")
42 dispPal ← brewer.pal(3, "Set2")
43
44
45 ## FUNCTIONS #####
46
47 ## create a plot which visualizes positional data patterns by a categorical
   variable
48 ## could encode density as either box sizes or through alpha levels of colour
49 ## the areal representation breaks the "dimensionality" rule of data in Edward
   Tufte's "The Visual Display of Information",
50 ## to limit the dimensionality of a representation to at most the dimensionality
   of the data itself
51 ## place the legend labels in the largest box instead of off to the side (didn't
   really work...)
52 posboxplot ← function(x, y, cats, boxcolours = NULL, boxwids = 0.8, alphaencoding
   = TRUE, alphamin = 0.1,
53                      areaencoding = FALSE, xlim = range(x) + boxwids*c(-1/
   1.05, 1/1.05), inc.leg = TRUE,
54                      ylim = range(y) + boxwids*c(-1/1.05, 1/1.05), ...) {
55   ## extract the number of categories to be displayed in each small multiple
56   ncats ← length(unique(cats))
57   ## automatically generate the category colours using color brewer
58   if (is.null(boxcolours)) {
59     boxcols ← brewer.pal(ncats, "Set2")
60     boxcolours ← boxcols
61   } else boxcols ← boxcolours
62   ## first identify the unique coordinates for the small multiples
63   unqPos ← unique(cbind(x, y))
64   ## iterate through these, create tables of the categories at each position
65   cattabs ← t(apply(unqPos, 1, function(pos) {
66     ## generate a count a table
67     table(cats[x == pos[1] & y == pos[2]])
68   }))
69   ## sum these counts to get the total at each position to scale the small
   multiples
70   rowcounts ← rowSums(cattabs)
71   ## convert the count table to cumulative proportions at each position
72   catprops ← t(apply(cbind(0, cattabs/rowcounts), 1, cumsum))
73   ## use these and the table of counts to generate the small multiples
74   ## first in the case that opacity encodes density
75   if (alphaencoding) {
76     ## in the opacity-density case, convert the box colours to rgb and
       replicate them as necessary
77     boxcols ← col2rgb(rep(boxcols, each = nrow(catprops)), alpha = FALSE)/255
78     ## convert back to hex, adding the alpha encoding to control opacity
79     boxcols ← rgb(t(boxcols),
80                  alpha = rep(round((1-alphamin)*rowcounts/max(rowcounts) +
   alphamin, digits = 4), times = ncats))
81     ## in the case size encodes density, simply replicate the colors for the
       number of positions
82   } else boxcols ← rep(boxcols, each = nrow(catprops))
83   ## create an empty plot to place the small multiples
84   plot(x, y, col = NA, xlim = xlim, ylim = ylim, ...)
85   ## determine the width of the small multiple boxes

```

```

86   if (areaencoding) boxwids ← boxwids*sqrt(rowcounts/max(rowcounts))
87   ## define the bottom corner positions of the boxes
88   bottomx ← unqPos[,1] - boxwids/2
89   bottomy ← unqPos[,2] - boxwids/2
90   ## use the bottom corner positions to calculate box extents, with internal
     borders defined as well
91   rectx ← bottomx + catprops*boxwids
92   recty ← cbind(rep(bottomy, times = ncats), rep(bottomy + boxwids, times =
     ncats))
93   ## convert the x coordinates into a list of vectors specifying all positions
94   xvec ← lapply(1:(ncats+1), function(n) rectx[,n])
95   ## place the rectangles by unlisting this structure correctly
96   rect(xleft = unlist(xvec[1:ncats]), ybottom = recty[,1], xright = unlist(xvec
     [2:(ncats+1)]),
97       ytop = recty[,2], col = boxcols, border = boxcols)
98   ## include a legend if desired
99   if (inc.leg) legend(x = "top", legend = colnames(rectx)[-1], fill = boxcolours
     , bty = "n",
100                      xpd = NA, horiz = TRUE)
101 }
102
103 ## create a function for proportional parallel coordinate plots
104 ## incorporate possibility to display in a non-proportional absolute way
105 proparcoord ← function(fact, cats, levs = NULL, proportional = TRUE, includerel
     = proportional, ylim = NULL,
106                        colpal = NULL, ordering = NULL, legpos = "topleft",
                        brtpos = 1, brwid = 4, ...) {
107
108   ## create the x label
109   xnm ← deparse(substitute(fact))
110   ## perform a type check
111   if (!is.factor(fact)) fact ← as.factor(fact)
112   ## check if levs have been supplied
113   if (is.null(levs)) levs ← unique(cats)
114   ## extract the levels and indices of interest
115   levinds ← cats %in% levs
116   ## get the length of the categories provided and a table of frequencies
117   ctab ← table(as.character(cats[levinds]))
118   len ← length(fact)
119   lineLen ← length(levels(fact))
120   ## check if order is null and allow reordering
121   if (is.null(ordering)) ordering ← 1:lineLen
122   ## set the ylim and other values based on whether a proportional plot is
     desired
123   factab ← table(fact)
124   if (proportional) factab ← factab/len else if (is.null(ylim)) ylim ← c(0, max
     (factab))
125   ## generate a palette if one is not given
126   if (is.null(colpal)) colpal ← brewer.pal(length(ctab), "Set2")
127   colpal ← colpal[ordering]
128   ## check for missing ylim value
129   if (is.null(ylim)) yrng ← c(0,1) else yrng ← ylim
130   ## now plot everything
131   if (proportional) ynm ← "Proportion" else ynm ← "Count"
132   plot(NA, xlim = c(1,lineLen), ylim = yrng, xlab = xnm, ylab = ynm, xaxt = 'n'
     , ...)
133   lines(1:lineLen, factab[ordering])
134   ## create positions for relative proportions bar chart if this is desired
135   if (includerel) {
136     ## specify bar chart rectangle bounds
137     rectbounds ← seq(0.7, 0.7 - 0.03*(length(ctab)+3), by = -0.03)*yrng[2]
138     ## add the reference "total population" bar
139     rect(xleft = brtpos, xright = 1+brwid/4, ybottom = rectbounds[1], ytop =
     rectbounds[2], col = "black")
140   }
141   ## plot lines and relative size rectangles, depending on options
142   for (ii in 1:length(ctab)) {
143     ## get the counts for the subset selected by ii
144     subsetab ← table(fact[cats == names(ctab)[ii]])
145     ## set these proportional if desired
146     if (proportional) subsetab ← subsetab/ctab[names(ctab)[ii]]

```

```

146     ## place these in a line
147     lines(1:lineLen, subsettab[ordering], col = colpal[ii])
148     ## add the appropriate bar if desired
149     if (includerel) {
150         rect(xleft = brtpos, xright = 1+(brwid/4)*ctab[ii]/len, ybottom =
151             rectbounds[ii+1], ytop = rectbounds[ii+2],
152             col = colpal[ii])
153     }
154     ## add a legend and axis
155     legend(x = legpos, legend = c("All", names(ctab)), lty = 1, col = c("Black",
156         colpal), title = deparse(substitute(cats)))
157     if (includerel) text("Relative Totals", x = 1, y = 0.72*yrng[2], pos = 4)
158     axis(1, 1:lineLen, levels(fact)[ordering])
159 }
160 ## DEPRECATED: a specific case of the above function which has been replaced by
161     the following function
162 ## make a specific line plot function
163 parcoordrace <- function() {
164     ## clean up the defwhiteblack variable
165     DefWhiteBlack_clean <- as.factor(as.character(gsub(",Other|,U", "", sun.juror$
166         DefWhiteBlack)))
167     ## first generate the necessary mixed factor
168     jurorDef <- with(sun.juror, as.factor(paste(DefWhiteBlack_clean, WhiteBlack,
169         sep = ":")))
170     ## generate positions to associate these factor levels
171     xpos <- rep(1:5, each = 4) + rep(c(-0.21,-0.07,0.07,0.21), times = 5)
172     ## choose disposition levels
173     displevs <- c("", "Kept", "S_rem", "D_rem", "C_rem")
174     nicelevs <- c("All", "Jury", "Pros.", "Def.", "Cause")
175     ## create a table based on the mixed factor
176     mixtab <- with(sun.juror, lapply(displevs,
177         function(displ) {
178             tab <- table(jurorDef[grepl(displ,
179                 Disposition)])/sum(grepl(displ,
180                 Disposition))
181             wraptab <- c(tab, tab[1:4])
182             wraptab
183         })))
184     ## define a palette
185     colpal <- brewer.pal(length(displevs) - 1, "Set2")
186     ## extract the max value for plotting purposes
187     maxtab <- max(unlist(mixtab))
188     ## plot all tables using different colours
189     lapply(1:length(mixtab), function(ind) {
190         if (ind == 1) {
191             plot(x = xpos, y = mixtab[[ind]], xlim = range(xpos), ylim = c(0,
192                 maxtab), xlab = "", xaxt = "n", ylab = "Proportion of Data",
193                 col = "black", type = 'l', yaxt = 'n')
194         } else lines(xpos+0.006*(ind-4)+0.003, mixtab[[ind]], col = colpal[ind
195             -1], lty = 2))
196     ## add axes
197     axis(side = 2, at = round(seq(0, maxtab, length.out = 3), digits = 2))
198     axis(1, at = xpos, labels = rep(c("Black", "Other", "Unknown", "White"), times =
199         5))
200     axis(1, at = 1:5, labels = c("Black Defendant", "Other", "Unknown Defendant", "
201         White Defendant", "Black Defendant"),
202         pos = -0.05, xpd = NA, tick = FALSE)
203     ## add guide lines coloured by disposition
204     lapply(2:length(displevs),
205         function(ind) {
206             sapply(1:20, function(n) rect(xleft = rep(xpos[n], 2)+0.006*(ind-4)
207                 , xright = rep(xpos[n], 2)+0.006*(ind-3),
208                 ybottom = mixtab[[1]][n], ytop =
209                 mixtab[[ind]][n], border =
210                 colpal[ind-1],
211                 col = colpal[ind-1]))
212         })
213     ## add legend-ish text

```

```

202   text(x = xpos[1]-0.01, y = mixtab[[2]][1] + 0.0075, labels = nicelevs[2], pos
      = 2, cex = 0.75, srt = 90,
203       col = colpal[1])
204   text(x = xpos[1]+0.01, y = mixtab[[3]][1]-0.02, labels = nicelevs[3], pos =
      1, cex = 0.75, srt = 90, col = colpal[2])
205   text(x = xpos[1]+0.02, y = mixtab[[4]][1]+0.04, labels = nicelevs[4], pos =
      1, cex = 0.75, srt = 90, col = colpal[3])
206   text(x = xpos[1]+0.03, y = mixtab[[5]][1], labels = nicelevs[5], pos = 1, cex
      = 0.75, srt = 90, col = colpal[4])
207 }
208
209 ## the better version of the above function, takes an arbitrary three-way
      contingency table and plots the different conditional
210 ## probabilities of the desired margins
211 parcoordracev2 ← function(tabl = NULL, tracemar = 1, deslev = NULL, wid = 0.02,
      addlines = FALSE,
212                          space = 0.025, testlines = FALSE, ...) {
213   ## in the default case (no table provided), look at the key race
      relationships, as these motivated this study
214   if (is.null(tabl)) {
215     ## for cleanliness, remove those jurors with unknown races
216     temp.juror ← sun.juror[sun.juror$WhiteBlack != "U" & sun.juror$
      DefWhiteBlack != "U",]
217     temp.juror$WhiteBlack ← as.factor(as.character(temp.juror$WhiteBlack))
218     temp.juror$DefWhiteBlack ← as.factor(as.character(temp.juror$
      DefWhiteBlack))
219     ## perform the same operation for defendant race
220     temp.juror$DefWhiteBlack ← as.factor(as.character(gsub(",Other|,U", "",
      temp.juror$DefWhiteBlack)))
221     ## make a table of disposition and both races
222     outcometab ← table(temp.juror[,c("Disposition", "DefWhiteBlack", "
      WhiteBlack")])
223   } else { ## if a table is provided simply copy it to the internal "outcometab
      " argument
224     outcometab ← tabl
225   }
226   ## determine the "non-trace" margins; these specify margins which are
      combined to define the cases to which the horizontal
227   ## line segments correspond
228   nontrace ← (1:3)[1:3 != tracemar]
229   ## get the dimension names
230   tabnames ← dimnames(outcometab)
231   ## handle a null desired level setting
232   if (is.null(deslev)) deslev ← 1:length(tabnames[[tracemar]])
233   ## create a palette
234   temPal ← brewer.pal(length(deslev), "Set2")
235   ## calculate the conditional probability distribution of outcome given non-
      trace margins using outcometab
236   condout ← apply(outcometab, nontrace, function(margin) margin/sum(margin))
237   ## and the sums
238   marsums ← apply(outcometab, nontrace, sum)
239   ## extract the desired levels from the margin of interest, write this
      flexibly to allow programmatic margin extraction later
240   ## first define the local environment as the calling environment
241   evEnv ← environment()
242   ## create the language object
243   condoutinds ← condoutcall ← quote(condout[,])
244   ## place the levels of interest as the subset argument
245   condoutcall[[tracemar+2]] ← deslev
246   ## evaluate this to subset the data
247   condout ← eval(condoutcall, envir = evEnv)
248   ## rename some useful values for ease of reference
249   ## the dimensionality of the data
250   dims ← dim(condout)
251   ## the number of horizontal segments
252   nseg ← prod(dims[nontrace])
253   ## the number of 'inner' combination values
254   innern ← dims[nontrace[1]]
255   ## the number of 'outer' combination values (inner and outer refer to axis
      label positions)

```

```

256 outern ← dims[nontrace[2]]
257 ## add padding between segments to space everything nicely
258 ## first replicate the trace margin sums to create a vector of the correct
    size for the horizontal line generation
259 tempx ← rep(c(marsums), times = c(rep(2, nseg-1),1))
260 ## replace every even element with the desired space size, then the
    cumulative sum automatically spaces
261 tempx[2*(1:(nseg-1))] ← space*rep(c(rep(1,innern-1),3), length.out = nseg-1)*
    sum(marsums)
262 ## take the cumulative sum to get the positions
263 xpos_line ← c(0, cumsum(tempx)/sum(tempx))
264 ##xpos ← rep(1:dims[nontrace[2]], each = dims[nontrace[1]]) +
265 ## rep(seq(-0.2, 0.2, length.out = dims[nontrace[1]]), times = dims[
    nontrace[2]])
266 ##xpos ← cumsum(marsums)/sum(marsums)
267 ## create the empty plot region
268 plot(NA, xlim = range(xpos_line), ylim = c(0, max(condout[,])), xaxt = 'n',
    xlab = "",
269     ylab = "Conditional Probability", ...)
270 ## calculate and plot the horizontal lines at the mean values
271 ## the old way: calculating the mean of the conditional distributions
272 meanline ← apply(condout, nontrace, mean)
273 ## this way does not correspond to the hypothesis being tested: that there is
    no preference for race displayed by
274 ## either side, rather under this hypothesis, the expected rate of each
    combination is given by the product of the
275 ## overall rate for each side and the proportion of the venire of
276 for(ii in 1:length(meanline)) lines(c(xpos_line[2*ii-1],xpos_line[2*ii]), rep
    (meanline[ii],2))
277 ## add the vertical lines for each cell, save the positions for later
278 xpos ← sort(unlist(lapply(1:length(deslev), function(ind) {
279     ## first extract the relevant margin to give the y values
280     tempind ← condoutinds
281     tempind[[tracemar + 2]] ← ind
282     yvals ← eval(tempind, envir = evEnv)
283     ## use the horizontal line positions and the index to place the vertical
        lines
284     ##adjx ← xpos + wid*(ind - (1/2)*(1 + length(deslev)))
285     adjx ← xpos_line[2*(1:nseg)-1] + (ind-1)*diff(xpos_line)[2*(1:nseg)-1]/(
        dims[tracemar] - 1)
286     ## add the corresponding end points
287     points(adjx, yvals, col = temPal[ind], pch = 19)
288     ## for aesthetics exclude lines if confidence intervals are plotted
289     if (!testlines) for (ii in 1:length(adjx)) lines(x = rep(adjx[ii],2), y =
        c(meanline[ii], yvals[ii]),
290                                     lty = 2, col = temPal[
        ind])
291     ## if trace lines (parallel axis plot) are desired plot these
292     if (addlines) lines(adjx, yvals, col = temPal[ind], lty = 3)
293     ##rect(xleft = adjx - (1/2)*wid, xright = adjx + (1/2)*wid, ybottom =
        meanline,
294         ytop = yvals, col = temPal[ind])
295     return(adjx)
296 })))
297 ## now add the axes
298 ## first determine axis label positions
299 axpos ← sapply(1:nseg, function(ind) mean(xpos_line[c((2*ind-1),2*ind)]))
300 outrpos ← sapply(1:outern, function(ind) mean(range(xpos_line[(2*(ind-1)*
    innern + 1):(2*ind*innern)])))
301 ## add the axis ticks for the inner labels
302 axis(1, at = axpos, labels = rep("", length(axpos)))
303 ## add the labels to the inner axis
304 axis(1, at = axpos, tick = FALSE, labels = rep(tabnames[[nontrace[1]]], times
    = outern), cex.axis = 0.7,
305     pos = -0.02*max(condout))
306 ## add the labels for the outer axis
307 axis(1, at = outrpos, labels = tabnames[[nontrace[2]]], xpd = NA,
308     tick = FALSE, pos = -0.08*max(condout[,]))
309 ## provide the axis title to give context

```

```

310 axis(1, at = mean(range(xpos)), xpd = NA, tick = FALSE, pos = -0.15*max(
311   condout),
312   labels = paste0("Inner label: ", names(tabnames)[nontrace[1]], " | Outer
313     label: ", names(tabnames)[nontrace[2]]))
314 ## add testing lines if desired
315 if (testlines) {
316   ## get x positions
317   errpos <- xpos
318   ##errpos[3*(1:nseg) - 1] <- axpos
319   ## reformat y positions
320   erry <- c(condout)
321   ## define error bar extensions
322   ext <- c(-0.005, 0.005)
323   ## add bars at each position
324   invisible(sapply(1:length(erry), function(pos) {
325     ## rename the conditional probability for readability
326     p <- erry[pos]
327     ## extract the relevant margin count
328     n <- marsums[floor((pos-1)/dims[tracemar]) + 1]
329     ## calculate the binomial error size
330     err <- 2*sqrt((p/n)*(1-p))
331     ## and the appropriate colour
332     errcol <- temPal[(pos-1) %% dims[tracemar] + 1]
333     ## add vertical lines and horizontal end lines
334     lines(x = errpos[pos] + ext, y = rep(p + err, 2), col = adjustcolor(
335       errcol, alpha.f = 0.5))
336     lines(x = errpos[pos] + ext, y = rep(p - err, 2), col = adjustcolor(
337       errcol, alpha.f = 0.5))
338     lines(x = rep(errpos[pos], 2), y = c(p + err, p - err), col =
339       adjustcolor(errcol, alpha.f = 0.5))
340     ##lines(, meanline - sqrt((meanline/(3*marsums))*(1-3*meanline)), lty
341       = 2)
342     ##lines(xpos, meanline + sqrt((meanline/(3*marsums))*(1-3*meanline)),
343       lty = 2)
344   })))
345 }
346 ## add a legen to explain the colours
347 legend(x = "top", horiz = TRUE, legend = tabnames[[tracemar]][deslev], col =
348   temPal, inset = -0.04, cex = 0.7,
349   fill = temPal, bg = "white", xpd = NA)
350 ##invisible(sapply((0:4)*0.05, function(val) lines(x = c(0,max(xpos)+1), y =
351   rep(val,2), col = "white", lwd = 2)))
352 }
353
354 ## the better version of the above function, takes an arbitrary three-way
355 contingency table and plots the different conditional
356 probabilities of the desired margins
357 testplot <- function(tabl = NULL, tracemar = 1, deslev = NULL, wid = 0.02,
358   addlines = FALSE,
359   space = 0.025, testlines = FALSE, expected = NULL,
360   ...) {
361   ## in the default case (no table provided), look at the key race
362   relationships, as these motivated this study
363   if (is.null(tabl)) {
364     ## for cleanliness, remove those jurors with unknown races
365     temp.juror <- sun.juror[sun.juror$WhiteBlack != "U" & sun.juror$
366       DefWhiteBlack != "U",]
367     temp.juror$WhiteBlack <- as.factor(as.character(temp.juror$WhiteBlack))
368     temp.juror$DefWhiteBlack <- as.factor(as.character(temp.juror$
369       DefWhiteBlack))
370     ## perform the same operation for defendant race
371     temp.juror$DefWhiteBlack <- as.factor(as.character(gsub(",Other|,U", "",
372       temp.juror$DefWhiteBlack)))
373     ## make a table of disposition and both races
374     outcometab <- table(temp.juror[,c("Disposition", "DefWhiteBlack", "
375       WhiteBlack")])
376   } else { ## if a table is provided simply copy it to the internal "outcometab
377     " argument
378     outcometab <- tabl
379   }

```



```

362  ## determine the "non-trace" margins; these specify margins which are
      combined to define the cases to which the horizontal
363  ## line segments correspond
364  nontrace ← (1:3)[1:3 != tracemar]
365  ## get the dimension names
366  tabnames ← dimnames(outcometab)
367  ## handle a null desired level setting
368  if (is.null(deslev)) deslev ← 1:length(tabnames[[tracemar]])
369  ## create a palette
370  temPal ← brewer.pal(length(deslev), "Set2")
371  ## calculate the conditional probability distribution of outcome given non-
      trace margins using outcometab
372  condout ← apply(outcometab, nontrace, function(margin) margin/sum(margin))
373  ## and the sums
374  marsums ← apply(outcometab, nontrace, sum)
375  ## extract the desired levels from the margin of interest, write this
      flexibly to allow programmatic margin extraction later
376  ## first define the local environment as the calling environment
377  evEnv ← environment()
378  ## create the language object
379  condoutinds ← condoutcall ← quote(condout[,])
380  ## place the levels of interest as the subset argument
381  condoutcall[[tracemar+2]] ← deslev
382  ## evaluate this to subset the data
383  condout ← eval(condoutcall, envir = evEnv)
384  ## rename some useful values for ease of reference
385  ## the dimensionality of the data
386  dims ← dim(condout)
387  ## the number of horizontal segments
388  nseg ← prod(dims)
389  ## the number of 'inner' combination values
390  innern ← dims[nontrace[1]]
391  ## the number of 'outer' combination values (inner and outer refer to axis
      label positions)
392  outern ← dims[nontrace[2]]
393  ## the trace dimension as well
394  tracen ← dims[tracemar]
395  ## add padding between segments to space everything nicely
396  ## first replicate the trace margin sums to create a vector of the correct
      size for the horizontal line generation
397  tempx ← rep(c(marsums)/tracen, times = c(rep(tracen + 1, outern*innern-1),
      tracen))
398  ## replace certain elements with the desired space size, then the cumulative
      sum automatically spaces
399  tempx[(tracen+1)*(1:(innern*outern - 1))] ← space*rep(c(rep(1,innern-1),3),
      length.out = innern*outern-1)*sum(marsums)
400  ## take the cumulative sum to get the positions
401  xpos_line ← c(0, cumsum(tempx)/sum(tempx))
402  ## get the middle positions using the filter function
403  xpos ← c(filter(xpos_line, filter = c(1/2,1/2)))
404  ## remove midsections of padding spaces
405  xpos ← xpos[-(tracen + 1)*(1:(innern*outern - 1))]
406  xpos ← xpos[-length(xpos)]
407  ## use the xpos_line to get widths of sections
408  xposwids ← diff(xpos_line)[- (tracen + 1)*(1:(innern*outern-1))]
409  ## if the expected values are missing, take the original expectation: a
      uniform distribution conditioned on both
410  ## races
411  if (is.null(expected)) expected ← rep(apply(condout, nontrace, mean), each =
      tracen)
412  ## create the empty plot region
413  plot(NA, xlim = range(xpos_line), ylim = c(0, max(condout[,])), xaxt = 'n',
      xlab = "",
414  ylab = "Conditional Probability", ...)
415  ## plot each line and its corresponding expectation
416  invisible(lapply(1:length(xpos), function(ind) {
417    ## plot the horizontal line using the relevant values
418    lines(x = xpos[ind] + xposwids[ind]*c(-1/2,1/2), y = rep(expected[ind],2)
      )
419    ## add the point

```

```

420     points(x = xpos[ind], y = condout[ind], col = temPal[((ind - 1) %% tracen
421             ) + 1], pch = 19)
422     ## for aesthetics exclude lines if confidence intervals are plotted
423     if (!testlines) lines(x = rep(xpos[ind], 2), y = c(expected[ind], condout
424             [ind]),
425             col = temPal[((ind-1) %% tracen) + 1], lty = 2)
426   )))
427   ## now add the axes
428   ## first determine axis label positions
429   axpos ← sapply(1:(innern*outern), function(n) mean(xpos[(tracen*(n-1) + 1):(
430       tracen*n])))
431   outrpos ← sapply(1:outern, function(n) mean(xpos[(tracen*innern*(n-1) + 1):(
432       tracen*innern*n])))
433   ## add the axis ticks for the inner labels
434   axis(1, at = axpos, labels = rep("", length(axpos)))
435   ## add the labels to the inner axis
436   axis(1, at = axpos, tick = FALSE, labels = rep(tabnames[[nontrace[1]]], times
437       = outern), cex.axis = 0.7,
438       pos = -0.02*max(condout))
439   ## add the labels for the outer axis
440   axis(1, at = outrpos, labels = tabnames[[nontrace[2]]], xpd = NA,
441       tick = FALSE, pos = -0.08*max(condout[, ,]))
442   ## provide the axis title to give context
443   axis(1, at = mean(range(xpos_line)), xpd = NA, tick = FALSE, pos = -0.15*max(
444       condout),
445       labels = paste0("Inner label: ", names(tabnames)[nontrace[1]], " | Outer
446       label: ", names(tabnames)[nontrace[2]]))
447   ## add testing lines if desired
448   if (testlines) {
449     ## get x positions
450     errpos ← xpos
451     ##errpos[3*(1:nseg) - 1] ← axpos
452     ## reformat y positions
453     erry ← c(condout)
454     ## define error bar extensions
455     ext ← c(-0.005, 0.005)
456     ## add bars at each position
457     invisible(sapply(1:length(erry), function(pos) {
458         ## rename the conditional probability for readability
459         p ← erry[pos]
460         ## extract the relevant margin count
461         n ← marsums[floor((pos-1)/dims[tracemar]) + 1]
462         ## calculate the binomial error size
463         err ← 2*sqrt((p/n)*(1-p))
464         ## and the appropriate colour
465         errcol ← temPal[(pos-1) %% dims[tracemar] + 1]
466         ## add vertical lines and horizontal end lines
467         lines(x = errpos[pos] + ext, y = rep(p + err, 2), col = adjustcolor(
468             errcol, alpha.f = 0.5))
469         lines(x = errpos[pos] + ext, y = rep(p - err, 2), col = adjustcolor(
470             errcol, alpha.f = 0.5))
471         lines(x = rep(errpos[pos], 2), y = c(p + err, p - err), col =
472             adjustcolor(errcol, alpha.f = 0.5))
473     })))
474   }
475   ## add a legen to explain the colours
476   legend(x = "top", horiz = TRUE, legend = tabnames[[tracemar]][deslev], col =
477       temPal, inset = -0.04, cex = 0.7,
478       fill = temPal, bg = "white", xpd = NA)
479 }
480 ## back to back histogram plot
481 back2backh ← function(data1, data2, cols = NULL, legnames = NULL, ...) {
482   ## get the data1 and data2 names for the legend if no others are provided
483   if (is.null(legnames)) legnames ← c(deparse(substitute(data1)), deparse(
484       substitute(data2)))
485   ## generate and save the histograms of the data
486   hist1 ← hist(data1, plot = FALSE)
487   hist2 ← hist(data2, breaks = hist1$breaks, plot = FALSE)
488   ## use these to generate some necessary plotting parameters

```

```

478   maxden ← max(c(hist1$density, hist2$density))
479   nbins ← length(hist1$density)
480   ## generate colours if none are provided
481   if (is.null(cols)) cols ← c("steelblue", "firebrick")
482   ## create an empty plot area
483   plot(NA, xlim = c(-maxden, maxden), ylim = range(hist1$breaks), xlab = "
     Density", xaxt = 'n', ...)
484   ## add vertical separating line
485   abline(v = 0)
486   ## add an axis
487   axispos ← round(seq(0, maxden, length.out = 3), 2)
488   axis(side = 1, labels = c(axispos[3:2], axispos), at = c(-axispos[3:2],
     axispos))
489   ## plot the histograms back-to-back
490   rect(xleft = -hist1$density, ybottom = hist1$breaks[1:nbins],
491        xright = rep(0, nbins), ytop = hist1$breaks[2:(nbins+1)], col = cols[1])
492   rect(xleft = rep(0, nbins), ybottom = hist2$breaks[1:nbins],
493        xright = hist2$density, ytop = hist2$breaks[2:(nbins+1)], col = cols[2])
494   ## add a legend
495   legend(x = 0, y = max(hist1$breaks), legend = legnames, fill = cols, horiz =
     TRUE, xpd = NA, xjust = 0.5, yjust = 0,
496          bg = "white")
497 }
498
499 ## a function to re-level factor variables to make mosaic plots cleaner (useful
     helper generally)
500 MatRelevel ← function(data) {
501   temp ← lapply(data, function(el) if (is.factor(el)) as.factor(levels(el)[as.
     numeric(el)]) else el)
502   temp ← as.data.frame(temp)
503   names(temp) ← names(data)
504   temp
505 }
506
507 ## a simple helper to convert multiple factor levels into a single 'other' level
508 FactorReduce ← function(vals, tokeep) {
509   chars ← as.character(vals)
510   ## simply replace elements
511   chars[!grepl(paste0(tokeep, collapse = "|"), chars)] ← "Other"
512   chars
513 }
514
515
516 ## DATA INSPECTION #####
517
518 ## load the data
519 if ("FullSunshine_Swapped.csv" %in% list.files(ThesisDir)) {
520   sun.swap ← read.csv(paste0(ThesisDir, "/FullSunshine_Swapped.csv"))
521 } else source(paste0(ThesisDir, "/DataProcess.R"))
522 FullSunshine ← read.csv(paste0(ThesisDir, "/FullSunshine.csv"))
523
524 ## summarize onto the correct scale, the jurors
525 if ("JurorAggregated.Rds" %in% list.files(ThesisDir)) {
526   sun.juror ← readRDS(paste0(ThesisDir, "/JurorAggregated.Rds"))
527 } else sun.juror ← UniqueAgg(sun.swap, by = "JurorNumber", collapse = ",")
528
529 ## also load the data summarized onto the trial scale
530 if ("TrialAggregated.Rds" %in% list.files(ThesisDir)) {
531   sun.trialsum ← readRDS(paste0(ThesisDir, "/TrialAggregated.Rds"))
532 } else warning(paste0("No trial aggregated data found in ", ThesisDir))
533
534 ## there are two juries without charges or other info (noted in the early data
     cleaning but kept for other analysis), remove these
535 sun.trialsum ← sun.trialsum[!(sun.trialsum$TrialNumberID %in% c("590-128", "710-01
     ")),]
536
537 ## display information about juror rejection tendencies
538 mosaicplot(Race ~ Disposition, data = sun.juror, las = 2, shade = TRUE)
539
540 ## create a race filtered data set

```

```

541 sun.raceknown ← sun.juror[sun.juror$Race != "U" & sun.juror$DefRace != "U",]
542 sun.raceknown$DefWhiteBlack ← gsub("U", "", sun.raceknown$DefWhiteBlack)
543 sun.raceknown ← MatRelevel(sun.raceknown)
544
545 ## try plotting these
546 mosaicplot(Race ~ PerempStruck, data = sun.raceknown, main = "Race vs. Removal",
  shade = TRUE, las = 2)
547 mosaicplot(Race ~ Disposition, data = sun.raceknown, main = "Race by Trial Status",
  shade = TRUE, las = 2)
548 mosaicplot(Race ~ DefStruck, data = sun.raceknown, main = "Race by Defence
  Removal", shade = TRUE, las = 2)
549 mosaicplot(Race ~ ProStruck, data = sun.raceknown, main = "Race by Prosecution
  Removal", shade = TRUE, las = 2)
550 mosaicplot(Race ~ CauseRemoved, data = sun.raceknown, main = "Race by Removal
  with Cause", shade = TRUE, las = 2)
551 ## it seems that there are significantly different strike habits between the
  defense and prosecution, but that
552 ## generally the system does not strike at different rates on average
553 ## recall the paper "Ideological Imbalance and the Peremptory Challenge"
554 par(mfrow = c(1,2))
555 mosaicplot(Race ~ PoliticalAffiliation, data = sun.raceknown[sun.raceknown$Gender
  == "M",],
556   main = "Affiliation and Race (Men)", shade = TRUE, las = 2)
557 mosaicplot(Race ~ PoliticalAffiliation, data = sun.raceknown[sun.raceknown$Gender
  == "F",],
558   main = "Affiliation and Race (Women)", shade = TRUE, las = 2)
559 mosaicplot(Race ~ DefStruck, data = sun.raceknown[sun.raceknown$Gender == "M",],
560   main = "Defense Removals and Race (Men)", shade = TRUE, las = 2)
561 mosaicplot(Race ~ DefStruck, data = sun.raceknown[sun.raceknown$Gender == "F",],
562   main = "Defense Removals and Race (Women)", shade = TRUE, las = 2)
563 mosaicplot(Race ~ ProStruck, data = sun.raceknown[sun.raceknown$Gender == "M",],
564   main = "Prosecution Removals and Race (Men)", shade = TRUE, las = 2)
565 mosaicplot(Race ~ ProStruck, data = sun.raceknown[sun.raceknown$Gender == "F",],
566   main = "Prosecution Removals and Race (Women)", shade = TRUE, las = 2)
567 par(mfrow = c(1,1))
568 ## maybe the same forces are at play here, compare to simulation?
569 ## alternatively, the strong relationship between race and political affiliation
  provides motivation for even an
570 ## unbiased lawyer to preferentially strike one race or the other
571
572 ## these mosaic plots can be confusing, and seemed ineffective upon first
  presentation, try parallel axis plots
573 ## instead
574 ## begin with an overall plot displaying the data at a high level
575 parcoordrace()
576 parcoordracev2(deslev = c(1,2,5))
577 ## but are these differences significant?
578 parcoordracev2(deslev = c(1,2,5), testlines = TRUE)
579
580 ## the independence we want to test here is that of (Race, Disposition)|(
  Defendant Race)
581 ## filter the data to remove small categories
582 sun.chitest ← sun.raceknown
583 sun.chitest$Disposition ← gsub("U_rem", "Unknown", gsub("Foreman", "Kept", sun.
  chitest$Disposition))
584 ## start by generating a table
585 dispTab ← table(sun.chitest[,c("DefWhiteBlack", "Disposition", "WhiteBlack")])
586 ## now apply chi-square tests across the proper margin, start by simply
  generating the residuals
587 dispRes ← lapply(setNames(1:dim(dispTab)[1], dimnames(dispTab)[[1]]), function(
  ind) {
588   ## extract the two way table of this index
589   tab ← dispTab[ind,,]
590   tabdf ← dim(tab) - 1
591   ## calculate the expected values
592   exp ← outer(rowSums(tab), colSums(tab))/sum(tab)
593   ## and residuals
594   resids ← (tab - exp)/sqrt(exp)
595   ## calculate the observed chi-sq value
596   chival ← sum(resids^2)

```

```

597     ## and the p value
598     pval ← 1 - pchisq(chival, df = tabdf[1]*tabdf[2])
599     ## return these in a list
600     list(pval = pval, chisq = chival, df = tabdf[1]*tabdf[2], residuals = resids)
601 })
602 ## so, there is a significant difference in behaviour at the 5% level, and it is
603     highly significant for white and black jurors
604 ## but these results do not control for much, there could be many factors
605     confounding this result
606 ## first create a new data set for the model building
607 sun.jurmod ← sun.raceknown
608 sun.jurmod$DefVisMin ← sun.jurmod$DefWhiteBlack != "White"
609 sun.jurmod$VisMin ← sun.jurmod$WhiteBlack != "White"
610 sun.jurmod$DefStruck ← as.logical(sun.jurmod$DefStruck)
611 sun.jurmod$ProStruck ← as.logical(sun.jurmod$ProStruck)
612 ## now the tricky part, predicting the rejection of a potential juror based on a
613     host of factors, the problem is that we must
614 ## perform multinomial regression on the data, but this multinomial regression
615     makes comparison of certain parameters
616 ## impossible, i.e. there is no mathematical way to compare the impact of race
617     for prosecution and defense rejection statistically
618 ## start by building separate defense and prosecution rejection models
619 mod.def1 ← glm(DefStruck ~ Race + DefRace + Gender + DefGender + CrimeType +
620     DefAttyType + PoliticalAffiliation,
621     data = sun.jurmod, family = binomial)
622 ## very poorly fit model, but the reason should be fairly clear, the crime data
623     in particular has very specific and small
624 ## classes, try building up the model instead, and using the simpler race
625     variable
626 mod.def2 ← glm(DefStruck ~ WhiteBlack*DefWhiteBlack, data = sun.jurmod, family =
627     binomial)
628 mod.def3 ← update(mod.def2, formula = DefStruck ~ WhiteBlack + DefWhiteBlack)
629 ## idea: instead of multinomial regression, do poisson regression on the dispTab
630     above, this allows comparisons
631 sun.rdat ← data.frame(DefRace = rep(c("Black", "Other", "White"), times = 12),
632     Disposition = rep(rep(c("C_rem", "D_rem", "Kept", "S_rem"),
633     each = 3), times = 3),
634     Race = rep(c("Black", "Other", "White"), each = 12),
635     Count = c(dispTab[,c("C_rem", "D_rem", "Kept", "S_rem"),]))
636 ## estimate the saturated model first
637 mod.rsat ← glm(Count ~ DefRace*Disposition*Race, family = poisson, data = sun.
638     rdat)
639 ## now test if the final interaction term can be removed
640 mod.r1 ← update(mod.rsat, formula = Count ~ DefRace*Disposition + DefRace*Race +
641     Disposition*Race)
642 ## look at the significance
643 1 - pchisq(mod.r1$deviance, mod.r1$df.residual)
644 ## so, quite clearly, we cannot remove the three way interaction from the model,
645     as it is highly significant
646 ## the interpretation: the distribution of strikes, kept, etc. depends on both
647     the venire member race and the defendant race
648 ## still, this is perhaps not precise enough, if we change this data to only
649     delineate between those kept and the behaviour of
650     the lawyers
651 sun.rdat2 ← sun.rdat[sun.rdat$Disposition != "C_rem",]
652 mod.rsat2 ← glm(Count ~ DefRace*Disposition*Race, family = poisson, data = sun.
653     rdat2)
654 ## this is an interesting result, but perhaps it is related to political
655     affiliation (as indicated by the ideological balance
656     paper)
657 ## create a table to test this hypothesis
658 dispTab.pol ← table(MatRelevel(sun.chitest[!(sun.chitest$PoliticalAffiliation %in
659     % c("Lib", "U"))],
660     c("Disposition", "PoliticalAffiliation", "WhiteBlack", "DefWhiteBlack"))
661 )
662 dispTab.pol ← dispTab.pol[c("C_rem", "D_rem", "Kept", "S_rem"),,]

```

```

646 ## convert to a data frame for fitting
647 sun.pdat <- data.frame(Disp_ = rep(c("C_rem", "D_rem", "Kept", "S_rem"), times =
27),
648                               Pol_ = rep(rep(c("Dem", "Ind", "Rep"), each = 4), times =
9),
649                               Race_ = rep(rep(c("Black", "Other", "White"), each = 12),
times = 3),
650                               Def_ = rep(c("Black", "Other", "White"), each = 36),
651                               Count = c(dispNet.pol[,,,]))
652 ## fit a model analogous to those fit above, now controlled for political choices
in disposition
653 mod.psat <- glm(Count ~ Def_*Disp_*Race_ + Pol_*Disp_, family = poisson, data = sun
.pdat)
654 ## now remove the third order interaction in the model
655 mod.psatest <- update(mod.psat, formula = Count ~ Def_*Disp_*Race_ + Pol_*Disp_ -
Def_:Disp_:Race_)
656 ## test the models
657 anova(mod.psat, mod.psatest)
658 1 - pchisq(66.734, 12)
659
660 ## ideological imbalance, look at politics
661 parcoordracev2(table(MatRelevel(sun.raceknown[sun.raceknown$PoliticalAffiliation
!= "U",
662                               c("Disposition",
PoliticalAffiliation",
WhiteBlack")))),
663                               deslev = c(1,2,5))
664
665 ## use radial axis plots to view the lawyers tendencies, especially those who act
as both defence and prosecution lawyers
666 ## maybe remove the top lawyers and remodel
667 ## look at the most prolific lawyers for both sides
668 ## subset the data to only lawyers with one case to remove the lawyer dependency
669
670 ## however, this suggests another question: is this strategy actually successful?
That is, does there appear to
671 ## be a relation between the number of peremptory challenges and the court case
outcome?
672 ## this may be difficult, there are a lot of factors to consider:
673 ## - the lawyer and their track record
674 ## - how to judge the success/failure of the case
675 ## see if the presence of challenges is related to the verdict
676 mosaicplot(PerempStruck ~ Guilty, data = sun.swap, main = "Strikes by Guilt",
shade = TRUE)
677 ## on the level of jurors, this is certainly not the case, but this is not the
correct scale for the question being
678 ## asked, this question will be addressed again in the case-summarized data
679
680 ## a third obvious question is a comparison of which races strike or keep which
others, used the synthesized variable
681 ## above to try and identify this
682 mosaicplot(Race ~ StruckBy, data = sun.raceknown, shade = TRUE, main = "Race of
Juror to Race Removing Juror",
683           las = 2)
684 mosaicplot(Race ~ StruckBy, data = sun.raceknown[sun.raceknown$StruckBy != "Not
Struck",], shade = TRUE,
685           main = "Race to Race Removing (Only Removed)", las = 2)
686 ## this plot shows no large systematic deviation between the races in their
rejection habits, this suggests, that
687 ## the rejection that occurs is not as simple as a group identity check
688 ## this might be the wrong race to check, though, perhaps we are better comparing
the defendant and victim races to
689 ## strike habits
690 par(mfrow = c(1,3))
691 mosaicplot(Race ~ DefRace, data = sun.raceknown[as.logical(sun.raceknown$
DefStruck),], shade = TRUE,
692           main = "Race of Defense-Struck Jurors to Defendant Race", las = 2)
693 mosaicplot(Race ~ DefRace, data = sun.raceknown[as.logical(sun.raceknown$
ProStruck),], shade = TRUE,
694           main = "Race of Prosecution-Struck Jurors to Defendant Race", las = 2)

```

```

695 mosaicplot(Race ~ DefRace, data = sun.raceknown, las = 2, shade = TRUE, main = "
      Race of Defendant to Venire Race")
696 par(mfrow = c(1,1))
697 ## this makes the defense look as if they are not racist, though the comparison
to the venire distributions in the third
698 ## panel makes that clearer
699 ## these distributions to the venire distribution relative to defendant race,
first combine the smaller races into one
700 ## category to make the plot less noisy and more identifiable
701 ## now look at how the two behave relative in their rejections and their
acceptance
702 eikos(WhiteBlack ~ DefWhiteBlack + DefStruck, data = sun.raceknown, xlab_rot =
      90,
703       main = "Defense Challenges by Race of Venire Member and Defendant")
704 eikos(WhiteBlack ~ DefWhiteBlack + ProStruck, data = sun.raceknown, xlab_rot =
      90,
705       main = "Prosecution Challenges by Race of Venire Member and Defendant")
706 ## very interesting, the prosecution seems far more aggressive than the defense
707 sun.raceknown$DefWhiteBlack[sun.raceknown$DefWhiteBlack == "Black,U"] <- "Black"
708 sun.raceknown$DefWhiteBlack <- as.factor(as.character(sun.raceknown$DefWhiteBlack)
      )
709 mosaicplot(DefStruck ~ DefWhiteBlack + WhiteBlack, dir = c("v","v","h"), data =
      sun.raceknown, shade = TRUE, las = 2,
710           xlab = "Defendant Race and Defence Removals", ylab = "Juror Race",
              main = "Defence Removal by Defendant Race")
711 mosaicplot(ProStruck ~ DefWhiteBlack + WhiteBlack, dir = c("v","v","h"), data =
      sun.raceknown, shade = TRUE, las = 2,
712           xlab = "Defendant Race and Prosecution Removals", ylab = "Juror Race",
              main = "Prosecution Removal by Defendant Race")
713
714 ## that result is very interesting, the defense strike rates when conditioned on
defendant race show no racial
715 ## preference, with a preference to reject white jurors regardless of defendant,
but those of the prosecution do,
716 ## maybe by victim race?
717 mosaicplot(Race ~ VictimRace, data = sun.raceknown[as.logical(sun.raceknown$
      DefStruck)], shade = TRUE,
718           main = "Race of Defense-Struck Jurors to Defendant Race", las = 2)
719 mosaicplot(Race ~ VictimRace, data = sun.raceknown[as.logical(sun.raceknown$
      ProStruck)], shade = TRUE,
720           main = "Race of Prosecution-Struck Jurors to Defendant Race", las = 2)
721 ## hard to see anything there, the majority of victim races are unknown, maybe
looking at the races removed by defense
722 ## attorney type
723 mosaicplot(DefAttyType ~ Race, data = sun.raceknown[as.logical(sun.raceknown$
      DefStruck)], shade = TRUE, las = 2,
724           main = "Race of Defense-Struck Jurors to Defense Attorney Type")
725 mosaicplot(WhiteBlack ~ DefWhiteBlack + DefAttyType, data = sun.raceknown[as.logical(sun.
      raceknown$DefStruck)], shade = TRUE, las = 2,
726           main = "Race of Defense-Struck Jurors to Defense Attorney Type")
727 eikos(WhiteBlack ~ DefWhiteBlack + DefAttyType, data = sun.raceknown[as.logical(
      sun.raceknown$DefStruck)],
728       xlab_rot = 90)
729 ## so what have we seen above is that the prosecuton and defense seem to behave
very differently in their jury selection
730 ## tactics, the defense seems to reject white individuals at a high rate
regardless of the defendant, while the prosecution
731 ## seems to prefer the rejection of venire members of the same race as the
defendant
732
733 ## this last plot shows that different types of lawyers may have different
strategies, suggests a new investigation:
734 ## that of lawyer strategy and success based on lawyer tendencies, aggregating by
trial first will be easiest
735
736 ## load the jury summaries
737 if ("AllJuries.Rds" %in% list.files(ThesisDir)) {
738   sun.jursum <- readRDS(paste0(ThesisDir, "/AllJuries.Rds"))
739 } else cat(paste0("No file 'AllJuries.Rds' in ", ThesisDir))
740

```

```

741 ## now look at removals across trials for defense and prosecution
742 with(sun.trialsun, plot(jitter(DefRemEst, factor = 2), jitter(ProRemEst, factor =
743 2), pch = 20,
744 xlab = "Defense Strike Count (jittered)", ylab = "
745 Prosecution Strike Count (jittered)",
746 col = adjustcolor(racePal[as.numeric(DefWhiteBlack)],
747 alpha.f = 0.3)))
748 abline(0,1)
749 legend(x = "topleft", legend = levels(sun.trialsun$DefWhiteBlack), col = racePal,
750 pch = 20, title = "Defendant Race")
751 ## this is only somewhat informative, it is difficult to see any patterns, use
752 the custom posboxplot function
753 ## first encode relative size of point by alpha blending
754 with(sun.trialsun, posboxplot(DefRemEst, ProRemEst, DefWhiteBlack, boxcolours =
755 racePal, xlab = "Defense Strike Count",
756 ylab = "Prosecution Strike Count", boxwids = 0.8,
757 alphamin = 0.05))
758 ## next by area, another encoding option in this function
759 with(sun.trialsun, posboxplot(DefRemEst, ProRemEst, DefWhiteBlack, boxcolours =
760 racePal, xlab = "Defense Strike Count",
761 ylab = "Prosecution Strike Count", alphaencoding =
762 FALSE, areaencoding = TRUE))
763 ## break apart in more detail for the defense
764 DefStruckMeans <- with(sun.trialsun, sapply(levels(DefWhiteBlack),
765 function(rc) c(mean((Race.DefRem.
766 Black/Race.Venire.Black)[
767 DefWhiteBlack == rc],
768 na.rm = TRUE),
769 mean((Race.DefRem.
770 White/Race.Venire.
771 White)[
772 DefWhiteBlack ==
773 rc],
774 na.rm = TRUE))))
775 with(sun.trialsun, plot(Race.DefRem.Black/Race.Venire.Black, Race.DefRem.White/
776 Race.Venire.White, pch = 20,
777 xlab = "Black Venire Proportion Struck", ylab = "White
778 Venire Proportion Struck",
779 xlim = c(0,1), ylim = c(0,1), main = "Defense Strike
780 Proportions",
781 col = adjustcolor(racePal[as.numeric(DefWhiteBlack)],
782 alpha.f = 0.2)))
783 abline(0,1)
784 points(DefStruckMeans[1,], DefStruckMeans[2,], col = racePal, pch = 4, cex = 2,
785 lwd = 1.5)
786 legend(x = "topright", title = "Defendant Race", col = c(racePal,"black"), pch =
787 c(rep(20,3),4), bg = "white",
788 legend = c(levels(sun.trialsun$DefWhiteBlack),"Mean"))
789 ## hard to see the patterns at the lines, jitter the proportions
790 with(sun.trialsun, plot(Race.DefRem.Black/Race.Venire.Black + runif(nrow(sun.
791 trialsun), min = -0.03, max = 0.03),
792 Race.DefRem.White/Race.Venire.White, pch = 20,
793 xlab = "Black Venire Proportion Struck", ylab = "White
794 Venire Proportion Struck",
795 xlim = c(0,1), ylim = c(0,1), main = "Defense Strike
796 Proportions",
797 col = adjustcolor(racePal[as.numeric(DefWhiteBlack)],
798 alpha.f = 0.1)))
799 ## and for the prosecution
800 ProStruckMeans <- with(sun.trialsun, sapply(levels(DefWhiteBlack),
801 function(rc) c(mean((Race.ProRem.
802 Black/Race.Venire.Black)[
803 DefWhiteBlack == rc],
804 na.rm = TRUE),
805 mean((Race.ProRem.
806 White/Race.Venire.
807 White)[

```



```

782                                     DefWhiteBlack ==
783                                     rc],
                                     na.rm = TRUE))))
784 with(sun.trialsum, plot(Race.ProRem.Black/Race.Venire.Black, Race.ProRem.White/
    Race.Venire.White, pch = 20,
785                             xlab = "Black Venire Proportion Struck", ylab = "White
    Venire Proportion Struck",
786                             xlim = c(0,1), ylim = c(0,1), main = "Prosecution Strike
    Proportions",
    col = adjustcolor(racePal[as.numeric(DefWhiteBlack)],
        alpha.f = 0.2)))
787 abline(0,1)
788 points(ProStruckMeans[1,], ProStruckMeans[2,], col = racePal, pch = 4, cex = 2,
    lwd = 1.5)
789 legend(x = "topright", title = "Defendant Race", col = c(racePal,"black"), pch =
    c(rep(20,3),4), bg = "white",
790         legend = c(levels(sun.trialsum$DefWhiteBlack),"Mean"))
791 ## again hard to see, try jittering
792 with(sun.trialsum, plot(Race.ProRem.Black/Race.Venire.Black + runif(nrow(sun.
    trialsum), min = -0.03, max = 0.03),
793                             Race.ProRem.White/Race.Venire.White, pch = 20,
794                             xlab = "Black Venire Proportion Struck", ylab = "White
    Venire Proportion Struck",
795                             xlim = c(0,1), ylim = c(0,1), main = "Prosecution Strike
    Proportions",
796                             col = adjustcolor(racePal[as.numeric(DefWhiteBlack)],
    alpha.f = 0.1)))
797
798 ## both of these plots show a much higher proportion of the black venire is
799 usually struck for both sides, an unsurprising result
800 ## given the the black venire was shown to be smaller in the aggregate statistics
801 , looking at raw counts next:
802 ## for the defense
803 with(sun.trialsum, plot(jitter(Race.DefRem.Black, factor = 2), jitter(Race.DefRem
    .White, factor = 2), pch = 20,
804                             xlab = "Black Venire Strike Count (jittered)", ylab = "
    White Venire Strike Count (jittered)",
805                             xlim = c(0,13), ylim = c(0,13), main = "Defense Strike
    Counts",
806                             col = adjustcolor(racePal[as.numeric(DefWhiteBlack)],
    alpha.f = 0.2)))
807 legend(x = "topright", title = "Defendant Race", col = racePal, pch = 20, bg = "
    white", legend = levels(sun.trialsum$DefWhiteBlack))
808 ## use custom plot here
809 with(sun.trialsum, posboxplot(Race.DefRem.Black, Race.DefRem.White, DefWhiteBlack
    , racePal,
810                             xlab = "Black Venire Strike Count", ylab = "White
    Venire Strike Count",
811                             xlim = c(0,13), ylim = c(0,13), main = "Defense
    Strike Counts"))
812 with(sun.trialsum, posboxplot(Race.DefRem.Black, Race.DefRem.White, DefWhiteBlack
    , racePal,
813                             xlab = "Black Venire Strike Count", ylab = "White
    Venire Strike Count",
814                             xlim = c(0,13), ylim = c(0,13), main = "Defence
    Strike Counts",
815                             alphaencoding = FALSE, areaencoding = TRUE))
816 ## for the prosecution
817 with(sun.trialsum, plot(jitter(Race.ProRem.Black, factor = 1.2), jitter(Race.
    ProRem.White, factor = 1.2), pch = 20,
818                             xlab = "Black Venire Strike Count (jittered)", ylab = "
    White Venire Strike Count (jittered)",
819                             xlim = c(0,8), ylim = c(0,8), main = "Prosecution Strike
    Counts",
820                             col = adjustcolor(racePal[as.numeric(DefWhiteBlack)],
    alpha.f = 0.2)))
821 legend(x = "topright", title = "Defendant Race", col = racePal, pch = 20, bg = "
    white", legend = levels(sun.trialsum$DefWhiteBlack))
822 ## more of the custom plot

```

```

822 with(sun.trialsun, posboxplot(Race.ProRem.Black, Race.ProRem.White, DefWhiteBlack
823   , racePal,
824     xlab = "Black Venire Strike Count", ylab = "White
      Venire Strike Count",
825     xlim = c(0,13), ylim = c(0,13), main = "Prosecution
      Strike Counts"))
826 with(sun.trialsun, posboxplot(Race.ProRem.Black, Race.ProRem.White, DefWhiteBlack
827   , racePal,
828     xlab = "Black Venire Strike Count", ylab = "White
      Venire Strike Count",
829     xlim = c(0,13), ylim = c(0,13), main = "Prosecution
      Strike Counts",
830     alphaencoding = FALSE, areaencoding = TRUE))
831
832 ## interesting, this shows some patterns in lawyer behaviour at the trial level
833 ## so there are some obvious patterns we can see in the aggregated data and in
834 the individual cases, see if these affect outcomes
835 with(sun.trialsun, plot(DefRemEst ~ Outcome))
836 with(sun.trialsun, plot(ProRemEst ~ Outcome))
837 ## nothing obvious there, but there is no control for charges/crime type
838
839 ## compare these to other variables
840 mosaicplot(DefRace ~ CrimeType, data = sun.trialsun, las = 2, main = "Crime and
      Race", shade = TRUE)
841 mosaicplot(Outcome ~ CrimeType, data = sun.trialsun, las = 2, main = "Crime and
      Outcome", shade = TRUE)
842 boxplot(DefRemEst ~ CrimeType, data = sun.trialsun)
843 with(sun.trialsun, posboxplot(as.numeric(CrimeType), DefRemEst, DefWhiteBlack,
844   racePal, xaxt = "n",
845     ylab = "Defense Strike Count", xlab = "Crime Type")
      )
846 axis(side = 1, at = 1:7, labels = levels(sun.trialsun$CrimeType))
847 boxplot(ProRemEst ~ CrimeType, data = sun.trialsun)
848 with(sun.trialsun, posboxplot(as.numeric(CrimeType), ProRemEst, DefWhiteBlack,
849   racePal, xaxt = "n",
850     ylab = "Prosecution Strike Count", xlab = "Crime
      Type"))
851 axis(side = 1, at = 1:7, labels = levels(sun.trialsun$CrimeType))
852
853 ## try using the positional boxplots
854 with(sun.trialsun, posboxplot(DefRemEst, ProRemEst, CrimeType, crimePal))
855 ## too many classes, maybe try drug, sex, theft, other
856 sun.trialsun$DrugSexTheft ← as.factor(FactorReduce(sun.trialsun$CrimeType, tokeep
      = c("Drug", "Sex", "Theft")))
857 with(sun.trialsun, posboxplot(DefRemEst, ProRemEst, DrugSexTheft, boxcolours =
858   brewer.pal(4, "Set1")))
859 ## also summarize this for the juror data
860 sun.juror$DrugSexTheft ← as.factor(FactorReduce(sun.juror$CrimeType, tokeep = c("
      Drug", "Sex", "Theft")))
861
862 ## try something different, plot the tendency of the lawyers themselves
863 ## idea: horizontal axis is lawyers, vertical is strikes
864 LawyerTends ← lapply(unique(c(sun.trialsun$DefAttyName, sun.trialsun$ProsName)),
865   function(name) list(Prosecution = sun.trialsun$ProRemEst[
866     sapply(sun.trialsun$DefAttyName,

```

```

function
(
  nms
)

name

%
in
%

nms
)
],

```

```

862         Defense = sun.trialsun$DefRemEst[sapply
863             (sun.trialsun$ProsName,
function
  (
    nms
  )
    name
    %
    in
    %
    nms
  )
  ])
  )
864 ## order these by those who did both, then defense, then prosecution
865 LawyerOrder <- order(sapply(LawyerTends, function(lst) {
866     lstlens <- sapply(lst, function(el) length(el) > 0)
867     if (all(lstlens)) {
868         0
869     } else if (lstlens[[2]]) {
870         1
871     } else 2}
872     ))
873 ## reorder the lawyer tendencies
874 LawyerTends <- LawyerTends[LawyerOrder]
875 ## plot these
876 plot(NA, xlim = c(1,length(LawyerTends)), ylim = c(0, max(unlist(LawyerTends), na
  .rm = TRUE)))
877 invisible(lapply(1:length(LawyerTends), function(ind) {
878     vals <- LawyerTends[[ind]]
879     points(rep(ind, length(vals$Defense)), vals$Defense, col = adjustcolor("
  steelblue", alpha.f = 0.1), pch = 20)
880     points(rep(ind, length(vals$Prosecution)), vals$Prosecution, col =
  adjustcolor("red", alpha.f = 0.1), pch = 20)
881 })))
882 lines(1:length(LawyerTends), sapply(LawyerTends, function(el) mean(unlist(el), na
  .rm = TRUE)))

```

A.6 Using Sweave to include R code (and more) in your report

The easiest (and most elegant) way to include R code and its output (and have all your figures up to date with your report) is to use Sweave. You can find an introduction Sweave in `/u/sfs/StatSoftDoc/Sweave/Sweave-tutorial.pdf`.

Appendix B

Yet another appendix....

B.1 Description

Something details.

Something else other definition.

B.2 Tables

Refer to Table [B.1](#) to see a left justified table with caption on top.

Table B.1: Results.	
Student	Grade
Marie	6
Alain	5.5
Josette	4.5
Pierre	5

Epilogue

A few final words.

Declaration of Originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor .

Title of work (in block letters):

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):

First name(s):

Muster	Student

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the Citation etiquette information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work .
- I am aware that the work may be screened electronically for plagiarism.
- I have understood and followed the guidelines in the document *Scientific Works in Mathematics*.

Place, date:

Signature(s):

Zurich August 19th 2009	bla

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.