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 peremptory 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

List of Figures

List of Tables

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 reperesentation 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 accusor 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 it's 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

Although the focus of this text is the legal practice of peremptory challenges, these are a specific practice which may not be known in detail to the reader, 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. 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).

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

  (c) The venire member is accepted into the jury, and so becomes a juror

iv.) Steps i-iii are repeated until the desired number of jurors has been found

The details of each of these steps varies by region. Jury rolls can be collected from many different sources. In the United States, they are typically selected using lists of registered voters (see Van Dyke (1977) chapter two), but in Canada the practice varies province by province. 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).

While two presentation methods are observed 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 murder case. Hans and Vidmar (1986) suggest that this difference is due to a difference in philosophy. To borrow a quote from page 63 of Hans and Vidmar (1986):

> In Canada, for example, 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 opinion places a greater responsibility on the jurors themselves to overcome their biases and accept arguments in spite of them. The American opinion that certain prejudice cannot be overcome by jurors stands in stark contrast.

2.2 The Role 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 evidence. This has varied drastically in form throughout history. Consider that 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 practice of selecting the most "trustworthy" individuals of some reknown.

This is contrasted by the modern jury, which performs no collection of evidence, and is meant to be composed of 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 judge[1], to the case at hand. Evidence for guilt is then 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 law[1]. It should be clear that the jury therefore has tremendous power in the legal system. 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.

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 privileged removal of a venire member - to be replaced by a new randomly selected venire member - by either the prosecution or defence without justification has seen allegations of abuse.

In the United States, repeated allegations of racial discrimination have led to significant changes in their allowed use, through cases such as Supreme Court of the United States (1965) and Supreme Court of the United States (1986). The first of these cases, *Swain v. Alabama*, established in 1965 that the systematic removal of venire members of a particular race could 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 this. This placed a significant burden on the side taking issue with a challenge to demonstrate discrimination in the use of peremptory challenges.

However, this ruling was overturned only 21 years later in the 1986 case *Batson v. Kentucky*, which allowed the party ojecting 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 to use against peremptory challenge to keep a venire member: 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 gender and other characteristics of venire members.

In Canada, there have also been racial controversies. A report by a government inquiry in the province of Manitoba in 1991 (see Roach (2018)) was already reporting on possible

[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.

racial bias against First Nations venire members. More damning still was the Iacobucci (2013) Report on First Nations representation in juries 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). As it currently stands, the bill has not been approved by the Government of Canada, but seems likely to become law in the near future, which would abolish the peremptory challenge in Canada.

In doing so Canada would join the United Kingdom. Significant controversy around the use peremptory challenges there already led to the abolition of the practice by parliament in 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))[2].

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 remains controversial. The modern motivation and justification for the practice in spite of all of the controversy is 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 harks back to 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)[3].

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

[2]It should be noted that this did not abolish the use of "standing-aside" by the Crown, although the practice has been heavily curtailed to only national security trials with strict guidelines to its use, which are outlined by Attorney General's Office of the United Kingdom (2012).

[3]Such 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).

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 might appropriately begin 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).

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 present 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 on 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. Hoffman (1997) presents evidence of the selection of 81 Senators 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 in placing the power of removal with the litigant and suggesting no justification is necessary for their removal.

2.5.2 In English Law (1066–1988)

Peremptory challenge did not reach is modern form, as outlined above, 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"[4].

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 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.

Additionally, this is 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. As royal infallibilty grew out of favour, peremptory challenges seem to have been 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 precisely[5]. 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).

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

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

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 authorities[6].

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)).

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 bias (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 voire dire in the United States).

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 peers[7].

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 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

[6]For more detail on this development among the early colonists, it is instructive to read about the Zenger trial of 1734 (described form 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

[7]Indeed, 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

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 which would nullify peremptory ones.

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 the more egalitarian granting of limited 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 consitution 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 as in England. It is not until the Charter of Rights and Freedoms in 1982[8] that such rights were guaranteed in a founding 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 has existed in some form since at least the fourteenth century. After its inception in England, 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. Despite its abandonment by the English in 1988, it has remained highly prominent in the United States, accompanied by a voir dire process far more thorough and invasive than that performed in the English or Canadian jury selection process.

[8]This was the year of patriation of the Canadian constitution. As independence was granted by the Britsh 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 Consitution Act of 1982 that the Canadian constitution became a Canadian law. For a more detailed history see Sheppard (2018)

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 sky 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.

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 exmained 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 **?**). 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 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, were 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 a 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.

[1]The link provided in the Jury Sunshine Paper to the specific source (http://www.nccourts.org/Citizens/SRPlanning/Statistics/CAReports_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 aformentioned “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 variables[2].

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-alphnumeric 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

[2]One 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.

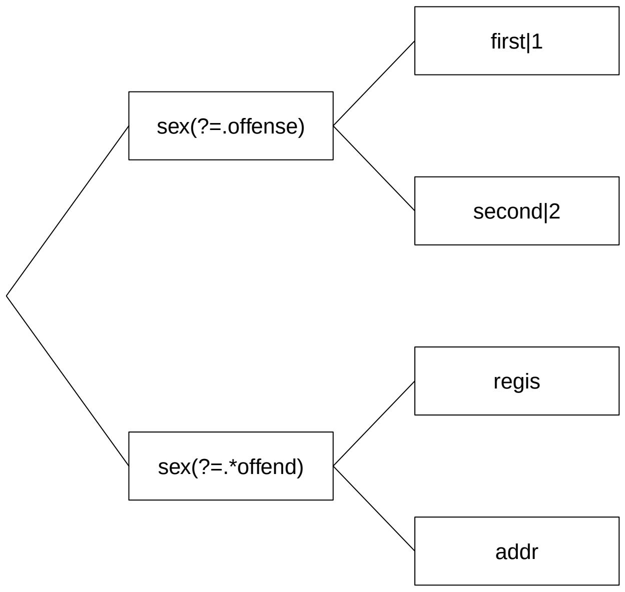

Figure 3.1: A example of a simple charge classification tree to separate the sexual offenses from charges leveled against previously known sex offenders. A charge would be classified from most general on the left to most specific on the right.

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 consulted[3]. 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

[3] As 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 1997[4]. 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% reliability[5].

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 confouding effects.

3.3.2 Cleaning

[4]This 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

[5]Additionally, 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 pereptory 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 attitudes[1]. Additionally, math-

[1]This 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	0.434	0.781	1.517
Stubborn			
Philadelphia			

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 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.

This visualization suffers from a number of limitations, some of which are obvious, and

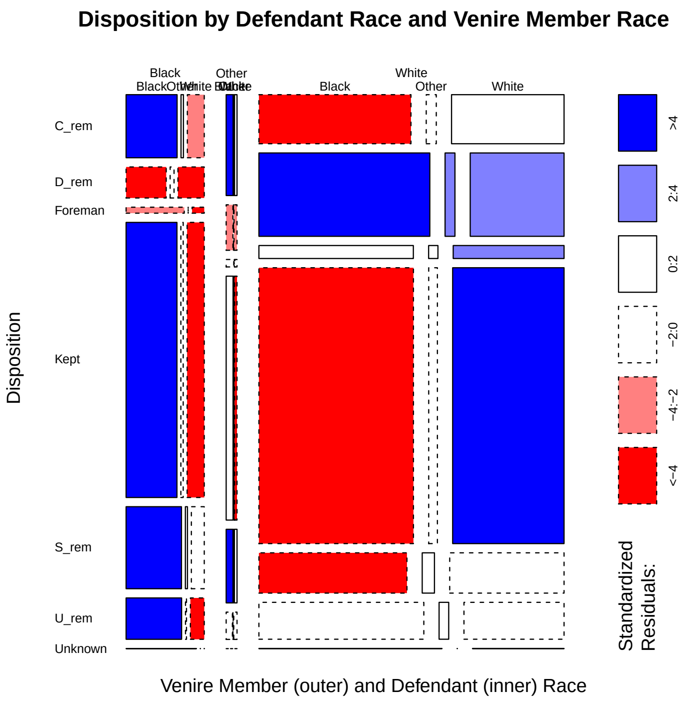

Figure 4.1: A mosaic plot of the simplified defendant and venire member race and their relation to the disposition of the venire member.

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.

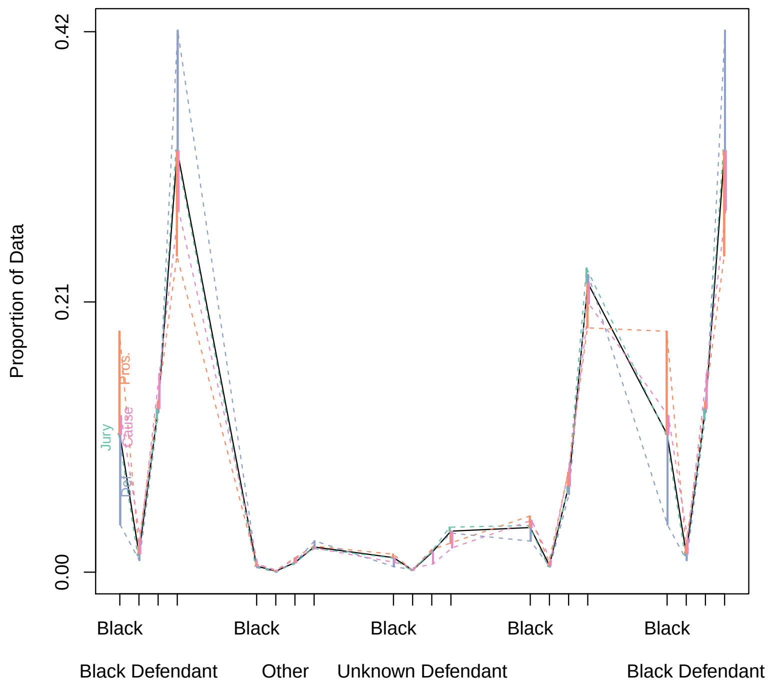

Figure 4.2: The first attempt at a parallel coordinate plot attempted. Note that the cramped display and unclear definition of the axis make interpretation even less intuitive than the mosaic plot, suggesting that this first attempt was a decided failure.

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, $P(Race, RaceDefendant|Disposition)$. 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 4.2.1.

4.2.1 The Mobile Plot

The primary data encoding method of the mobile plot is length. All lengths are anchored to the black horizontal lines, the vertical positions of which represent the expected values for the attached vertical lines, and the widths of which represent the relative proportion of the data represented by the marked combinaton on the horizontal axis. The vertical

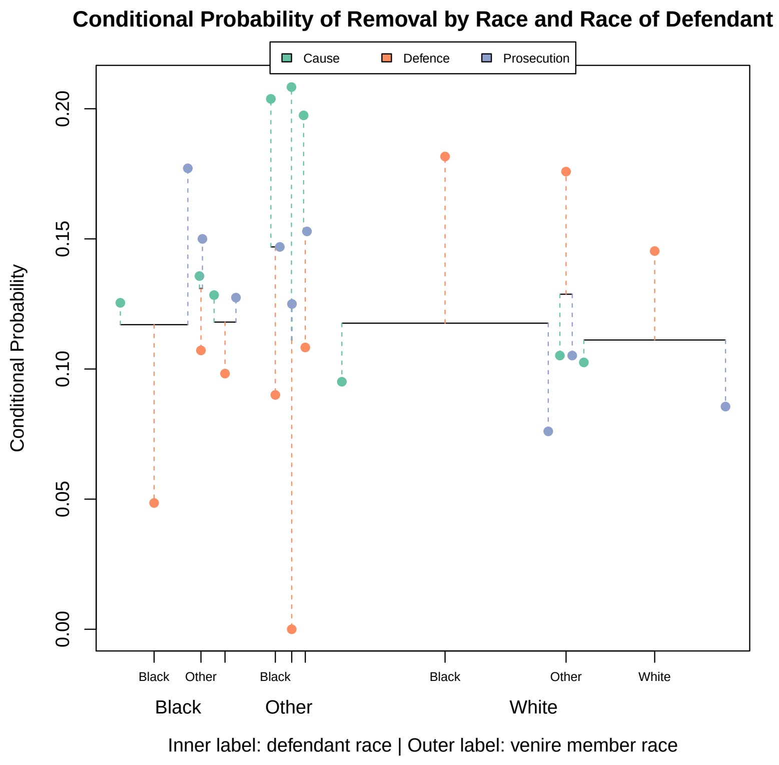

Figure 4.3: The “mobile plot” to display conditional distributions. Note that this plot is less cluttered than either the mosaic plot or the first parallel coordinate plot, despite displaying more information. It is also more efficient with data-ink, avoids displaying data with higher dimensions that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

axis gives the condition probability of a particular outcome, that is to say the distribution of outcomes given the combination indicated on the horizontal axis.

4.3 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.

4.4 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:

$$\log E[y_{ijk}] = \mathbf{x}_{ijk}\beta = \beta_o + \beta_R x_{i..} + \beta_D x_{.j.} + \beta_S x_{..k} + \beta_{R:D} x_{i..} x_{.j.} + \beta_{R:S} x_{i..} x_{..k} + \beta_{D:S} x_{.j.} x_{..k} + \beta_{R:D:S} x_{i..} x_{.j.} x_{..k} \quad (4.4.0.1)$$

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 independant 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.

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 combination[2].

The dominant pattern to these strikes is a tendency of the defense to preferentially reject white venire members and keep black venire members, and of the prosecution to do the opposite. It was already noted in the literatureWright 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

[2]Generated 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

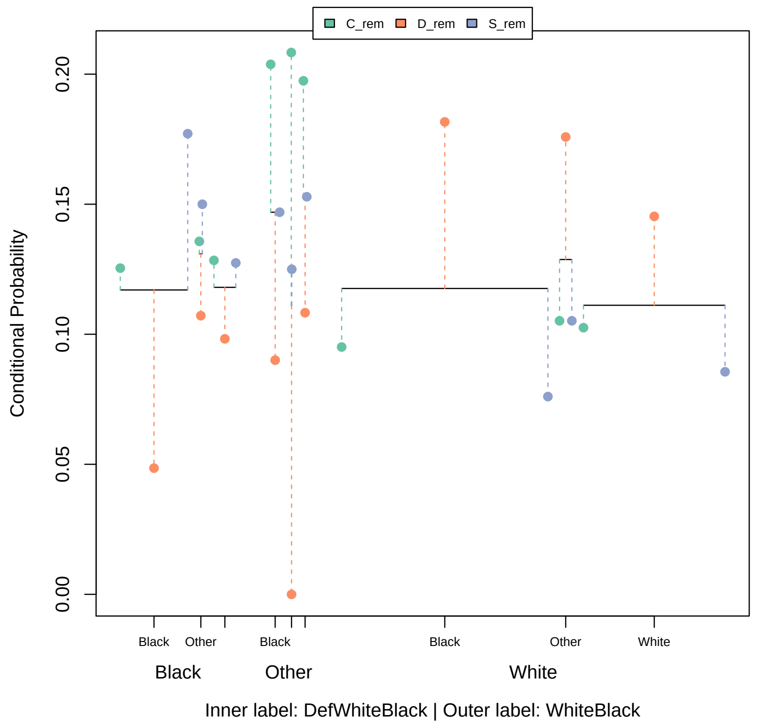

Figure 4.4: Parallel coordinate plot of racial strike tendencies

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.

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

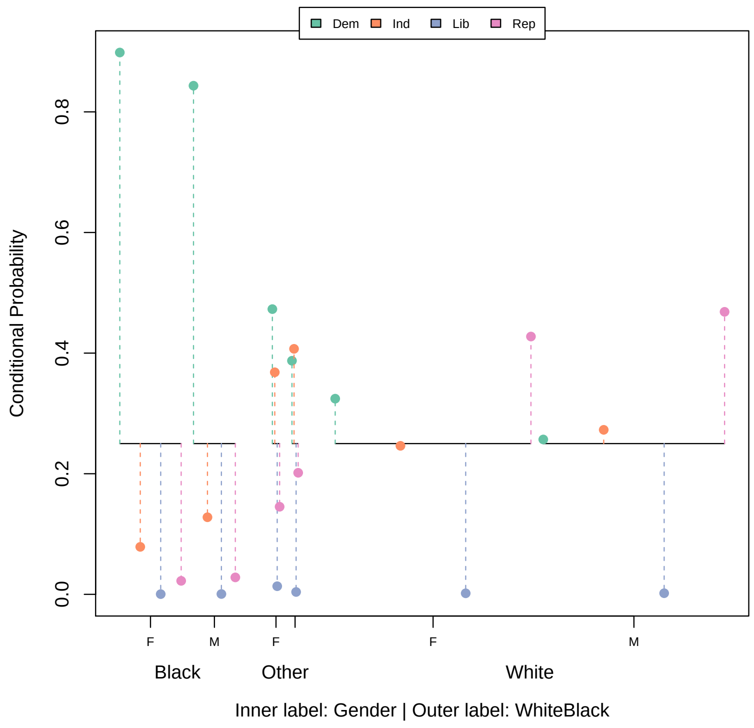

Figure 4.5: Conditional probabilities of political affiliation by race and gender

plotted in back-to-back histograms as shown in Figure 4.6.

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.

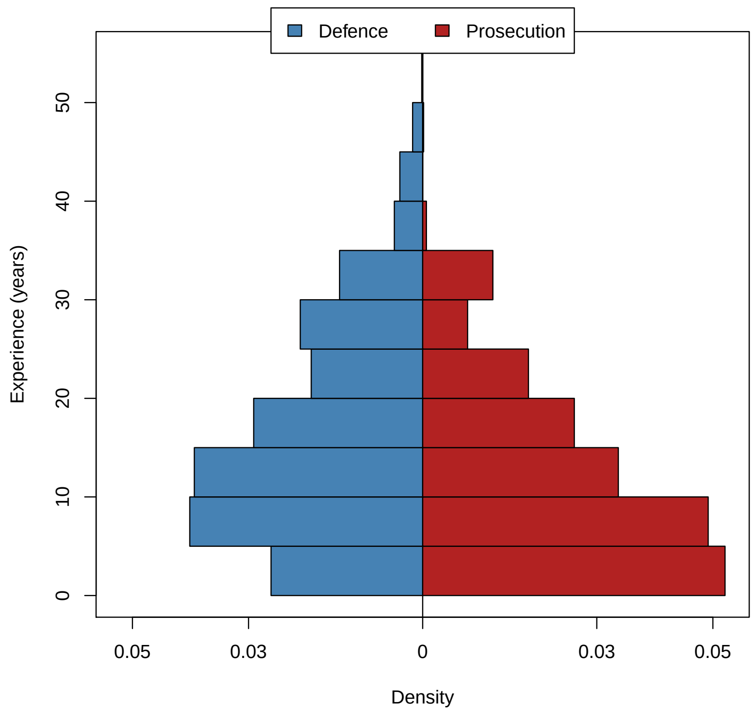

Figure 4.6: Distributions of lawyer experience for prosecutors and defence attorneys

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/charter-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 Consitutional 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 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_Representatio

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-colten-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.

Mirriam-Webster (2019a). Mirriam-Webster Dictionary Online. Accessed: https://www.merriam-webster.com/dictionary/venire.

Mirriam-Webster (2019b). Mirriam-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). Insitutionalized 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: Historica 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 Judgements. 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.

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 es   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.

```r
########################################

## THESIS DATA PROCESSING SCRIPT
## Christopher Salahub
## Sept 26, 2018

########################################

## PACKAGES ############################
library(readxl)
library(tm)
library(stringr)
library(grid)


## CONSTANTS ###########################

## start by defining file locations
ThesisDir ← "c:/Users/Chris/Documents/ETH Zurich/Thesis/Data"
SunshineFile ← paste0(ThesisDir, "/JurySunshineExcel.xlsx")
SunshineSheets ← excel_sheets(SunshineFile)

NorthCarFile ← paste0(ThesisDir,
                      "/Jury Study Data and Materials/NC Jury Selection Study Database6 Dec 2011.csv")

PhillyFile ← paste0(ThesisDir,
                    "/Voir Dire Data & Codebook/capital_venires.csv")

## next the factor level codes as given in the codebook and regularized here
## regularization: - political affiliation "N" replaced with "I" for all entries
LevRace ← sort(c("A","B","H","N","O","U","W"))
LevGen ← sort(c("F","M","U"))
LevPol ← sort(c("D","L","R","I","U"))

## create a charge tree with regex nodes to identify and clean charge text
chargeTree ← list("rape" = list("statutory", "first|1", "second|2"), "sex(?=.*offense)" = list("first|1", "second|2"),
                  "sex(?=.*offend)" = list("regis", "addr"), "murder" = list("first|1" = list("att"), "second|2" = list("att")),
                  "arson", "firearm" = list("pos", "disch"), "stole" = list("pos"),
                  "mari" = list("pos", "sell|sale", "man", "pwimsd"), "coca" = list("pos", "sell|sale", "man", "pwimsd"),
                  "cs" = list("pos", "sell|sale", "man", "pwimsd"), "hero" = list("pos", "sell|sale", "man", "pwimsd"),
                  "meth" = list("pos", "sell|sale", "man", "pwimsd"),
                  "oxycod" = list("pos", "sell|sale", "man", "pwimsd"), "mass" = list("pos"), "break" = list("enter"),
                  "assa" = list("serious bodily", "female", "strangul", "deadly", "official"),
                  "larceny" = list("motor", "felon", "merchant"), "false" = list("pretense"),
                  "driving" = list("impaired"), "kidnap" = list("first|1", "second|2"),
                  "robb" = list("dang"), "burg" = list("first|1", "second|2"), "indec" = list("liber"),
                  "embez", "manslaughter" = list("inv"), "flee" = list("arrest"),
                  "abuse|cruelty" = list("child", "anim"), "identity" = list("theft"))

## create a list of variables which can sensibly be summarized by trial
```

```r
TrialVars ← c("TrialNumberID", "DateOutcome", "JudgeID", "DefAttyType", "VictimName",
               "VictimRace", "VictimGender", "CrimeLocation", "PropertyType",
               "ZipCode.Trials", "StateTotalRemoved", "DefenseTotalRemoved",
               "CourtTotalRemoved", "JDistrict", "JName", "JRace", "JGender",
               "JPoliticalAff", "JVoterRegYr", "JYrApptd", "JResCity", "JResZip",
               "ChargeTxt", "Outcome", "Sentence.FullSunshine", "DefendantID.FullSunshine",
               "DefendantID.DefendantToTrial", "DefRace", "DefGender", "DefDOB", "DefAttyID",
               "DefAttyName", "DCRace", "DCGender", "DCPoliticalAff", "DCYrRegVote",
               "DCYrLicensed", "DCResideCity", "DCResideZip", "ProsecutorID", "ProsName",
               "ProsRace", "ProsGender", "ProsPoliticalAff", "PYrRegVote", "PYrLicensed",
               "PResideCity", "PResideZip", "Guilty", "CrimeType", "DefWhiteBlack")


## FUNCTIONS ###########################

## Loading and cleaning ################
## create a descriptive merge function for cleaning (essentially a 'merge' wrapper)
CleaningMerge ← function(x, y, ...) {
    ## start by creating the merge
    ## first match arguments
    MatchCall ← match.call(merge)
    MatchCall[[1]] ← quote(merge)
    ## get input names and ensure proper name structure
    xname ← MatchCall$x
    if (!is.symbol(xname)) xname ← as.symbol(paste0(xname[[2]],xname[[3]]))
    yname ← MatchCall$y
    if (!is.symbol(yname)) yname ← as.symbol(paste0(yname[[2]],yname[[3]]))
    ## use this to extract suffixes and fix MatchCall
    MatchCall$suffixes ← paste0(".", c(xname, yname))
    MatchCall$x ← xname
    MatchCall$y ← yname
    ## specify that the match should be an outer join
    MatchCall$all ← TRUE
    ## and use this to make a clean local assignment to modify
    assign(as.character(xname), cbind(x, Diag.x = 1), envir = environment())
    assign(as.character(yname), cbind(y, Diag.y = 1), envir = environment())
    ## now evaluate the call
    Merged ← eval(MatchCall, envir = environment())
    ## next perform some checks
    xExpInds ← is.na(Merged$Diag.x)
    yExpInds ← is.na(Merged$Diag.y)
    ## remove the diagnostic columns
    Merged$Diag.x ← NULL; Merged$Diag.y ← NULL
    ## summarize the diagnostic checks
    X_nexp ← sum(xExpInds)
    Y_nexp ← sum(yExpInds)
    X_missing ← Merged[xExpInds,]
    Y_missing ← Merged[yExpInds,]
    ## print the diagnostics
    cat("Joined ", paste(xname, yname, sep = " and "), " with ",
        X_nexp, " and ", Y_nexp, " failed matches respectively \n", sep = "")
    ## return the results, preferentially keeping the data which is present in x but missing from y
    if (X_nexp == 0 & Y_nexp == 0) {
        Merged
    } else list(Merge = Merged[!xExpInds,], Xfails = X_missing, Yfails = Y_missing)
}

## a function to identify and perform swaps with user input
SimpleSwapper ← function(data, CorrectLevs, auto = FALSE) {
    ## first match the data to the columns of interest
    colInds ← match(names(CorrectLevs), names(data))
    ## extract the levels of the columns of interest to check if there are any potential swaps
    swapCheck ← all(sapply(1:length(colInds),
                           function(ind) identical(sort(levels(as.factor(data[, colInds[ind]]))),
                                                   sort(CorrectLevs[[ind]]))))
    ## if no swaps are present end this check
    if (swapCheck) {
        cat("No errors found, exiting.")
        return(data)
    }
    ## if errors are found, further investigate them
    ## identify potential rows
    ## first those which have elements out of place
    SwapPoss ← sapply(1:length(colInds),
                      function(ind) !(data[,colInds[ind]] %in% CorrectLevs[[ind]]))
    ## now rows containing unknown entries
    Unknown ← sapply(1:length(colInds),
                     function(ind) data[,colInds[ind]] == "U")
    ## identify potential swaps by row
    Swaps ← apply(SwapPoss, 1, function(row) sum(row) > 1)
    ## identify the potential errors
    PotErr ← apply(SwapPoss, 1, function(row) sum(row) == 1)
    ## use the unknowns to account for some errors
    UnkInd ← apply(Unknown, 1, any)
    FalErr ← PotErr & UnkInd
    ## identify the indices to investigate
    SwapInds ← which(Swaps|FalErr)
    ErrInds ← which(PotErr & !UnkInd)
    ## communicate to the user and ask for input
    cat("There are ", sum(Swaps|FalErr), " swaps to check\n", sep = "")
    cat("Additionally, it seems there are ", sum(PotErr & !UnkInd), " errors in entries\n", sep = "")
    ## unless automated
    if (auto) ErrorReturn ← TRUE else ErrorReturn ← as.logical(readline("Return the errors? (T/F): "))
    ## now, if there are possible swaps investigate them
    if (sum(Swaps|FalErr) != 0) {
        ## create a temporary storage structure
        tempRows ← data[SwapInds, colInds]
        tempRows ← as.data.frame(lapply(tempRows, function(var) levels(var)[as.numeric(var)]),
                                 stringsAsFactors = FALSE)
        ## loop through and populate this
        for (ii in 1:nrow(tempRows)) {
            ## inspect the row
            print(tempRows[ii,])
            ## suggest corrections, first generate matches
            candComb ← lapply(tempRows[ii,],
                              function(el) which(sapply(CorrectLevs,
                                                        function(levs) el %in% levs)))
            reps ← unlist(lapply(candComb, length))
            ## now generate all swap combinations
            candComb[[1]] ← rep(candComb[[1]], each = max(reps[-1]))
            candComb ← as.data.frame(candComb, row.names = NULL)
            ## identify rows which contain all indices, in other words those valid as swaps
            compRows ← apply(candComb, 1, function(row) all(1:length(CorrectLevs) %in% row))
            goodComb ← candComb[compRows,]
            ## clean them up and print them
            colnames(goodComb) ← NULL
            rownames(goodComb) ← NULL
            cat("Potential combinations:\n")
            print(t(apply(goodComb,1,order)))
            ## take user input or automatically determine value
            if (auto) {
                if (!any(compRows)) acceptedComb ← 0 else acceptedComb ← 1
            } else acceptedComb ← as.numeric(readline("Enter a combination choice (0 for error, <enter> to accept first): "))
            ## handle special cases, 0 if a true error has been identified
            if (identical(acceptedComb,0)) { ## 0 if a true error has been identified
                ErrInds ← c(ErrInds, SwapInds[ii])
                cat("True error identified, adding ", SwapInds[ii], " to error list\n", sep = "")
            } else { ## the case where a swap has been correctly identified and selected, or enter has been pressed
                ## if enter has been pressed accept the first row
                if (is.na(acceptedComb)) acceptedComb ← 1
                ## print recombined row
                newRows ← tempRows[ii,order(as.matrix(goodComb[acceptedComb,]))]
                colnames(newRows) ← NULL
                rownames(newRows) ← NULL
                cat("Corrected row:")
                print(newRows)
                cat("------------------\n")
                ## correct entry
                tempRows[ii,] ← newRows
            }
        }
        ## fill the data
        ## first prevent factor level errors
        data[,colInds] ← lapply(colInds, function(ind) levels(data[,ind])[as.numeric(data[,ind])])
        ## now swap the data
        data[SwapInds,colInds] ← lapply(1:length(colInds), function(ind) tempRows[,ind])
        ## reconvert back to factors
        data[,colInds] ← lapply(colInds, function(ind) as.factor(data[,ind]))
    }
    ## in either case return the data and errors as specified
    if (ErrorReturn) {
        return(list(Data = data, Errors = ErrInds))
    } else {
        return(data)
    }
}

## now create a function to address the errors possibly identified in the above function automatically
SwapErrorFix ← function(errorData, CorrectLevs) {
    ## check if we are in the case without errors
    if (!identical(names(errorData), c("Data", "Errors"))) {
        cat("No errors\n")
        return(errorData)
    } else {
        ## extract the data and data in error
        fulldata ← errorData$Data
        ## get the relevant columns
        colInds ← match(names(CorrectLevs), names(fulldata))
        ## go through the specified variables and remove errors
        fixed ← lapply(1:length(colInds),
                        function(ind) {
                            var ← fulldata[,colInds[ind]]
                            var ← levels(var)[as.numeric(var)]
                            inds ← !(var %in% CorrectLevs[[ind]])
                            cat(names(CorrectLevs)[ind], ": ", sum(inds),
                                " errors\n", sep = "")
                            var[inds] ← "U"
                            as.factor(var)
                        })
        ## insert these fixed values
        fulldata[, colInds] ← fixed
        ## return this
        fulldata
    }
```

```r
}

## write a wrapper to perform this swapping and error correction in one call
SwapandError <- function(data, CorrectLevs) {
    swapped <- SimpleSwapper(data = data, CorrectLevs = CorrectLevs, auto = TRUE)
    fixed <- SwapErrorFix(errorData = swapped, CorrectLevs = CorrectLevs)
    fixed
}

## Variable Synthesis #################
## Kullback-Leibler divergence function
kldiv <- function(samp, dist) {
    ## convert to matrices
    mat1 <- as.matrix(samp)
    mat2 <- as.matrix(dist)
    ## make into proper distributions
    mat1 <- mat1/rowSums(mat1)
    mat2 <- mat2/rowSums(mat2)
    ## take the log ratio
    logratio <- log(mat1/mat2)
    ## multiply by correct matrix
    vals <- mat1*logratio
    ## take the row sums
    rowSums(vals, na.rm = TRUE)
}

## make a text-mining regularization function
StringReg <- function(strs) {
    ## first set everything to lowercase
    strs <- tolower(strs)
    ## replace specific patterns (noticed during early tests)
    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")
    strs <- str_replace_all(strs, "controlled substance", "cs")
    strs <- str_replace_all(strs, "dwi", "driving while impaired")
    strs <- str_replace_all(strs, "rwdw", "robbery with a deadly weapon")
    strs <- str_replace_all(strs, "pwisd|pwmsd|pwmsd|pwitd|pwid|pwmisd|pwosd", "
        pwimsd")
    strs <- str_replace_all(strs, "robery|rob ", "robbery")
    strs <- str_replace_all(strs, "bulgary", "burglary")
    strs <- str_replace_all(strs, "awdw", "assault with a deadly weapon")
    strs <- str_replace_all(strs, "(?<=[\\sa-z])[0-9]{2,}", "")
    strs <- str_replace_all(strs, "att ", "attempted ")
    strs <- str_replace_all(strs, "assult", "assault")
    strs <- str_replace_all(strs, "marj", "marijuana")
    ## replace punctuation
    strs <- gsub("[^[:alnum:][:space:]']", "", strs)
    ## return these
    strs
}

## create a function to process such a tree structure given a list of strings
stringTree <- function(strs, regexTree, inds = 1:length(strs), includeOther = TRUE
    ) {
    ## identify the sublists, and divide the data
    sublists <- sapply(regexTree, is.list)
    ## iterate over unnamed items (leaf nodes)
    listdiv <- lapply(regexTree[!sublists], function(el) inds[grepl(el, strs, perl
         = TRUE)])
    names(listdiv) <- unlist(regexTree[!sublists])
    ## check if there are any sublists
    if (!any(sublists)) {
        if (includeOther) listdiv <- c(listdiv, other = list(inds[!(inds %in%
            unlist(listdiv))]))
        ## in the case of none, treat the object as a list to iterate through
        listdiv
    } else {
        ## otherwise recurse over the branches
        finlist <- c(listdiv, lapply(names(regexTree)[sublists],
                                        function(name) stringTree(strs[grepl(name,
                                            strs, perl = TRUE)],
                                                                  regexTree[[name]],
                                                                  inds[grepl(name,
                                                                      strs, perl =
                                                                      TRUE)],
                                                                  includeOther)))
        names(finlist)[(length(listdiv) + 1):length(finlist)] ← names(regexTree)[
            sublists]
        c(finlist, other = list(inds[!(inds %in% unlist(finlist))]))
    }
}

## create a tree depth helper function
maxdepth ← function(tree, counter = 1) {
    max(sapply(tree, function(br) if (!is.list(br)) counter else maxdepth(br,
        counter + 1)))
}

## create a function to aggregate a tree as specified above at the desired depth
treeAgg ← function(tree, level = 1) {
    ## first check the max depth of the tree
    treedepth ← maxdepth(tree)
    ## compare this to requested aggregation level
    stopifnot(level <= treedepth)
    ## aggregate at desired level with a helper function
    agg ← function(tr, depth = 1) {
        if (depth == level) lapply(tr, function(el) setNames(unlist(el),NULL))
            else lapply(tr, function(br) agg(dr, depth + 1))
    }
    agg(tree)
}

## create a crime class aggregation function
CrimeClassify ← function(tree, regChar) {
    crimes ← list()
    crimes$Sex ← unique(c(unlist(tree[c("rape", "sex(?=.*offense)", "sex(?=.*
        offend)", "indec")]),
                          tree$other[grepl("sex", regChar[tree$other])]))
    crimes$Theft ← unique(unlist(tree[c("stole", "embez", "break", "larceny", "
        robb", "burg", "identity")]))
    crimes$Murder ← unique(unlist(tree[c("murder", "manslaughter")]))
    crimes$Drug ← unique(c(unlist(tree[c("mari", "coca", "cs", "hero", "meth", "
        oxycod")]),
                           tree$other[grepl("para|drug|substance|pwimsd",
                               regChar[tree$other])]))
    crimes$Violent ← unique(unlist(tree[c("arson", "assa", "abuse|cruelty")]))
    crimes$Driving ← unique(c(unlist(tree[c("driving")]),
                              tree$other[grepl("hit(?=.*run)|speeding", regChar[
                                  tree$other], perl = TRUE)]))
    crimes
}

## in order to make the process of pre-processing the data and adding desired
    columns, place the pre-processing into a
## flexible function and add operations as desired
SynCols ← function(data) {
    ## too busy, synthesize some variables to clearly indicate the results of
        defense and prosecution selection
    data$VisibleMinor ← data$Race != "White"
    data$PerempStruck ← grepl("S_rem|D_rem", data$Disposition)
    data$DefStruck ← data$Disposition == "D_rem"
    data$ProStruck ← data$Disposition == "S_rem"
    data$CauseRemoved ← data$Disposition == "C_rem"
    ## lets look at which race struck each juror
    data$StruckBy ← as.factor(sapply(1:nrow(data),
                                     function(ind) {
                                         dis ← as.character(data$
                                             Disposition[ind])
                                         if (dis == "S_rem") {
                                             as.character(data$ProsRace
                                                           [ind])
                                                   } else if (dis == "D_rem") {
                                                       as.character(data$DCRace[
                                                           ind])
                                                   } else "Not Struck"
                                               }))
    ## create a white black other indicator
    data$WhiteBlack ← FactorReduce(data$Race, tokeep = c("Black", "White", "U"))
    data$DefWhiteBlack ← FactorReduce(data$DefRace, tokeep = c("Black", "White",
        "U"))
    data$VicWhiteBlack ← FactorReduce(data$VictimRace, tokeep = c("Black", "White
        ", "U"))
    ## return the data with synthesized columns
    data
}

## write functions to process the sentences
SentenceProcess ← function(sentencing) {
    sents ← tolower(sentencing)
    ## identify sentences in months, years, and days
    monthsent ← str_extract(sents, "[0-9\\-]+\\s*(?=m)")
    daysent ← str_extract(sents, "[0-9\\-]+\\s*(?=d)")
    yearsent ← str_extract(sents, "[0-9\\-]+\\s*(?=y)")
    ## extract life without parole
    lwp ← str_extract(sents, "parol[e]*")
    ## and with parole
    life ← str_extract(sents, "life")
    life[!is.na(lwp)] ← NA
    ## get restitutions
    resti ← str_extract(sents, "[0-9,]+\\s*(?=restitu)|\\$[0-9,]+")
    ## get supervised probation
    suprob ← str_extract(sents, "sup.*pro")
}

## Summary Functions ##################
## make a function to summarize trial jury data
JurySummarize ← function(Varnames = c("Disposition", "Race", "Gender", "
    PoliticalAffiliation")) {
    ## check if a juror summary object exists already
    if (!("sun.juror" %in% ls(.GlobalEnv))) {
        ## first group the data for easy access
        Juries ← aggregate(sun.swap[, Varnames],
                            by = list(TrialNumberID = sun.swap$TrialNumberID,
                                JurorNumer = sun.swap$JurorNumber),
                            unique)
    } else Juries ← sun.juror
    ## in either case, perform aggregation by trial instance
    Juries ← aggregate(Juries[, Varnames],
                        by = list(TrialNumberID = Juries$TrialNumberID),
                        function(var) var)
    ## clean up the names
    names(Juries)[grepl("Polit", names(Juries))] ← "PolAff"
    Varnames[4] ← "PolAff"
    ## now summarize relevant features
    Summary ← apply(Juries[, Varnames], 1,
                     function(row) {
                         ## get final jury indices
                         disps ← unlist(row$Disposition)
                         foreman ← grepl("Foreman", disps)
                         finJur ← grepl("Foreman|Kept", disps)
                         defStruck ← grepl("D_rem", disps)
                         proStruck ← grepl("S_rem", disps)
                         ## process all variables
                         newrow ← sapply(row,
                                          function(el) {
                                              c(Jury = table(unlist(el)[finJur]),
                                                Venire = table(unlist(el)),
                                                DefRem = table(unlist(el)[
                                                    defStruck]),
                                                ProRem = table(unlist(el)[
                                                    proStruck]))
                                          })
                            newrow$Disposition <- NULL
                            newrow <- c(unlist(newrow), ForeRace = row$Race[foreman],
                                        ForeGender = row$Gender[foreman], ForePol =
                                            row$PolAff[foreman])
                            if (sum(foreman) > 1) {
                                names(newrow)[names(newrow) == "ForeRace1"] <- "
                                    ForeRace"
                                names(newrow)[names(newrow) == "ForeGender1"] <- "
                                    ForeGender"
                                names(newrow)[names(newrow) == "ForePol1"] <- "
                                    ForePol"
                            }
                            newrow
                       })
    ## perform some clean up
    longest <- sapply(Summary, length)
    longest <- which(longest == max(longest))[1]
    longNames <- names(Summary[[longest]])
    Summary <- lapply(names(Summary[[longest]]),
                      function(name) unname(sapply(Summary,
                                                   function(el) el[name])))
    names(Summary) <- longNames
    Summary <- lapply(longNames,
                      function(nm) {
                          if (grepl("ForeGender", nm)) {
                              Summary[[nm]] <- factor(Summary[[nm]], levels = 1:3,
                                   labels = LevGen)
                          } else if (grepl("ForePol", nm)) {
                              Summary[[nm]] <- factor(Summary[[nm]], levels = 1:5,
                                   labels = LevPol)
                          } else if (grepl("ForeRace", nm)) {
                              Summary[[nm]] <- factor(Summary[[nm]], levels = 1:7,
                                   labels = LevRace)
                          } else Summary[[nm]]
                      })
    names(Summary) <- longNames
    ## return these
    list(Juries = Juries, Summaries = as.data.frame(Summary))
}

## a generic simplification method to summarize a vector
Simplifier <- function(col, ...) {
    UseMethod("Simplifier")
}

## code up methods for the types to be seen
Simplifier.default <- function(col, collapse = "") paste0(col, collapse = collapse
    )
Simplifier.numeric <- function(col, na.rm = TRUE, trim = 0, ...) mean.default(col,
     trim = trim, na.rm = na.rm)
Simplifier.factor <- function(col, collapse = "", ...) paste0(sort(as.character(
    levels(col)[as.numeric(col)])),
                                                               collapse = collapse
                                                                   )
Simplifier.character <- function(col, collapse = "", ...) paste0(sort(col),
    collapse = collapse)

## create a grouping wrapper which does unique aggregation of a data set
UniqueAgg <- function(data, by, ...) {
    ## convert data to a data frame for regularity
    if (!is.data.frame(data)) data <- as.data.frame(data)
    ## identify the grouping column by in the data
    by.groups <- names(data) == by
    ## provide nice error handling
    stopifnot(sum(by.groups) > 0)
    ## first identify which rows are already unique
    groups <- as.numeric(as.factor(unlist(data[by.groups])))
    unqRows <- sapply(groups, function(el) sum(groups == el) == 1)
    ## consider grouping only the other rows using the unique function
    endata ← data[unqRows,]
    unqdata ← aggregate(data[!unqRows, !by.groups], by = list(data[!unqRows, by.
        groups]), unique)
    ## reorder to make sure everything is compatible
    names(unqdata)[1] ← by
    unqdata ← unqdata[,match(names(endata), names(unqdata))]
    ## now use the Simplifier helper defined above to process these results
    procdata ← lapply(unqdata, function(col) sapply(col, Simplifier, ...))
    ## append everything together
    endata ← lapply(1:length(endata),
                     function(n) c(if (is.factor(endata[[n]])) as.character(
                         endata[[n]]) else endata[[n]],
                                  procdata[[n]]))
    names(endata) ← names(data)
    ## convert to a data frame
    as.data.frame(endata)
}

## a simple helper to convert multiple factor levels into a single 'other' level
FactorReduce ← function(vals, tokeep) {
    chars ← as.character(vals)
    ## simply replace elements
    chars[!grepl(paste0(tokeep, collapse = "|"), chars)] ← "Other"
    chars
}

## write a function to re-level factor variables to make mosaic plots cleaner
MatRelevel ← function(data) {
    temp ← lapply(data, function(el) if (is.factor(el)) as.factor(levels(el)[as.
        numeric(el)]) else el)
    temp ← as.data.frame(temp)
    names(temp) ← names(data)
    temp
}

## another simple processing function to correct NA's given some other identifier
    and data set
FillNAs ← function(dataNAs, filldata, identifier) {
    ## extract the relevant column indices in a flexible way
    if (is.null(colnames(filldata))) {
        relcol ← grepl(identifier, names(filldata))
    } else relcol ← grepl(identifier, colnames(filldata))
    ## first identify the relevant rows in the data NAs
    relRows ← is.na(dataNAs)
    ## take the relevant rows of the filldata
    filldata ← matrix(unlist(filldata[relcol]), ncol = sum(relcol))
    rowfiller ← rowSums(filldata[relRows,])
    ## return the filled data
    dataNAs[relRows] ← rowfiller
    dataNAs
}

## write a wrapper to estimate the values of total removed jurors
RemovedJurorEstimates ← function(tofill, data, ident, plot = TRUE) {
    temp ← FillNAs(tofill, filldata = data, identifier = ident)
    temp2 ← rowSums(data[,grepl(ident, names(data))])
    ## let's see how accurate this is if plotting is desired
    if (plot) {
        plot(temp, temp2, xlab = "Observed and Filled", ylab = "Juror Sums")
        abline(0,1)
    }
    cat("= : ", sum(temp == temp2)/length(temp2), "\n", "< : ", sum(temp2 < temp)
        /length(temp2), "\n", sep = "")
    ## replace the filled values less than the estimated, for consistency
    temp[temp < temp2] ← temp2[temp < temp2]
    temp
}


```

```r
## LOADING AND PROCESSING DATA #########

## load the data
SunshineData ← lapply(SunshineSheets, function(nm) as.data.frame(read_excel(SunshineFile, sheet = nm)))
names(SunshineData) ← SunshineSheets
NorthCarData ← read.csv(NorthCarFile)
PhillyData ← read.csv(PhillyFile)

## clean non-informative columns
CleanSunshine ← lapply(SunshineData, function(dat) dat[, !apply(dat,2,function(col) all(is.na(col)))])

## the Sunshine data needs to be restructured into one table, rather than a relational database structure
## see the IDMatch function, this was created specifically to perform ID-based table joins
## the most appropriate global target is the juror table, start by matching this to the trial
FullSunshine ← with(CleanSunshine, CleaningMerge(Jurors, Trials, by = "TrialNumberID"))
## remove extra ID column, fix a misleading name
FullSunshine$CountyName ← FullSunshine$CountyID
FullSunshine$CountyID ← NULL
## clean up two additional columns which had inconsistencies
FullSunshine$Disposition ← toupper(FullSunshine$Disposition)
FullSunshine$Race[FullSunshine$Race == "?"] ← "U"
## before appending everything to this table, perform some other joins
TrialsToCharge ← with(CleanSunshine, CleaningMerge(Charges, Junction, by = "ACISID", all = TRUE))
DefendantToTrial ← with(CleanSunshine, CleaningMerge(Defendants, DefendantTrial, by = "DefendantID", all = TRUE))
AttorneyToTrial ← with(CleanSunshine, CleaningMerge(Attorney, AttorneyTrial, by = "DefAttyID", all = TRUE))
ProsecutorToTrial ← with(CleanSunshine, CleaningMerge(Prosecutor, ProsecutorTrial, by = "ProsecutorID", all = TRUE))
## merge issues:
##    - trials to charge: one charge is missing a trial ID, hopefully not important
##    - prosecutors to trials: 26 prosecutors without trials, however all entries were entirely uninformative
## given the above outputs, rename the failed clean merges to make the next section cleaner
TrialsToCharge ← TrialsToCharge$Merge
ProsecutorToTrial ← ProsecutorToTrial$Merge

## now perform some additional merges to create one sheet/data.frame
## add the judge descriptions (no issues)
FullSunshine ← CleaningMerge(FullSunshine, CleanSunshine$Judges, by = "JudgeID", all = TRUE)
## the charges
FullSunshine ← CleaningMerge(FullSunshine, TrialsToCharge, by = "TrialNumberID", all = TRUE)
## this leads to 22 jurors in trials without charges and 29 charges without trials, inspecting these:
##     - the jurors without charges are all related to a trial with ID number "710-01", thankfully the other data
##       for this case is complete, and so it may still be useful for viewing jury behaviour
##     - the charges without trials are all of the form "710-0xx", suggesting the omission of entire trials of some
##       relation, hopefully these were not too similar, or this exclusion can be explained later
FullSunshine ← FullSunshine$Merge
## the defendants
FullSunshine ← CleaningMerge(FullSunshine, DefendantToTrial, by = "TrialNumberID", all = TRUE)
## the attorneys
FullSunshine ← CleaningMerge(FullSunshine, AttorneyToTrial, by = "TrialNumberID", all = TRUE)
```

```r
## the prosecutors
FullSunshine ← CleaningMerge(FullSunshine, ProsecutorToTrial, by = "TrialNumberID", all = TRUE)
## 26 jurors appear to be lacking a prosecutor, these appear to be the uninformative prosecutors from earlier, included
## due to the preferential inclusion of the missing values in the first of the merged matrices
FullSunshine ← FullSunshine$Merge

## perform some cleanup
## start with some specific factor replacements
## replace the "N" with "I", as these factor levels are interchangeable in the codebook and prevent confusion with race
FullSunshine[,grepl("Pol", names(FullSunshine))] ← lapply(FullSunshine[,grepl("Pol", names(FullSunshine))],
                                                          function(var) {
                                                              var ← toupper(var)
                                                              var[var == "N"] ← "I"
                                                              var
                                                          })
## next save most variables as factors
FullSunshine ← lapply(FullSunshine,
                      function(el) if (is.character(el)) as.factor(el) else el)
## correct some overzealous assignment from above
FullSunshine[grepl("Notes", names(FullSunshine))] ← lapply(FullSunshine[grepl("Notes", names(FullSunshine))],
                                                           as.character)
## perform factor regularization according to the factor levels provided in the codebook
FullSunshine ← sapply(FullSunshine,
                      function(el) {
                          if (!is.factor(el)) {
                              el[el == 999] ← NA
                              el
                          } else {
                              el ← as.character(el)
                              el ← toupper(el)
                              el[is.na(el)] ← "U"
                              as.factor(el)
                          }
                      }, simplify = FALSE)
FullSunshine ← as.data.frame(FullSunshine)
## remove some unnecessary columns
FullSunshine$ID ← NULL
FullSunshine$TrialIDAuto ← NULL
## combine the name columns to produce more useful columns
FullSunshine$JName ← paste(FullSunshine$JFirstName, FullSunshine$JLastName)
FullSunshine$JName[FullSunshine$JName == "U U"] ← "U"
FullSunshine$DefAttyName ← paste(FullSunshine$DCFirstName, FullSunshine$DCLastName)
FullSunshine$DefAttyName[FullSunshine$DefAttyName == "U U"] ← "U"
FullSunshine$ProsName ← paste(FullSunshine$ProsecutorFirstName, FullSunshine$ProsecutorLastName)
FullSunshine$ProsName[FullSunshine$ProsName == "U U"] ← "U"

## Checkpoint 1: the clean data has been processed, none of the swaps, synthesis, or expansion has taken place
## save this
if (!("FullSunshine.csv" %in% list.files())) write.csv(FullSunshine, "FullSunshine.csv", row.names = FALSE)
## load if the desire is to start at checkpoint 1
if (!("FullSunshine" %in% ls())) FullSunshine ← read.csv("FullSunshine.csv")

## Note: the below swap functions have been set to auto as the function's performance in these cases has already
## been assessed, and so the swaps have already been inspected, it is critical for new data that "auto" be switched
## off to take full advantage of this functionality, and so the wrapper "SwapandError" should not be used
## in the juror data
sun.swapJuror ← SwapandError(FullSunshine, CorrectLevs = list(Race = LevRace,
                                                              Gender = LevGen,
                                                              PoliticalAffiliation = LevPol))
## in the judge data
sun.swap ← SimpleSwapper(sun.swapJuror, CorrectLevs = list(JRace = LevRace,
                                                           JGender = LevGen,
                                                           JPoliticalAff = LevPol
                                                           ))
## viewing the error report of these data, they are all related to one judge, Arnold O Jones II, who is verified
## as a male after a quick Google search
unique(sun.swap$Data[sun.swap$Errors, c("JFirstName", "JLastName")])
sun.swapJudge ← sun.swap$Data
sun.swapJudge$JGender[sun.swap$Errors] ← "M"
sun.swapJudge$JGender ← as.factor(levels(sun.swapJudge$JGender)[as.numeric(sun.swapJudge$JGender)])
## in the prosecutor data
sun.swap ← SimpleSwapper(sun.swapJudge, CorrectLevs = list(ProsRace = LevRace,
                                                           ProsGender = LevGen,
                                                           ProsPoliticalAff = LevPol
                                                           ))
## that found no errors
## a quick check of the levels of the defendant data finds only one error
levels(sun.swap$DefGender)
levels(sun.swap$DefRace)
sun.swap ← SwapandError(sun.swap, CorrectLevs = list(DefRace = LevRace,
                                                     DefGender = LevGen
                                                     ))
## next the attorney data
sun.swap ← SwapandError(sun.swap, CorrectLevs = list(DCRace = LevRace,
                                                     DCGender = LevGen,
                                                     DCPoliticalAff = LevPol))
## finally the victim data
sun.swap ← SwapandError(sun.swap, CorrectLevs = list(VictimRace = LevRace,
                                                     VictimGender = LevGen))
## this leaves the data error-free (in at least the race/gender/politics columns)

## fix the outcome data, which had some improper levels
sun.swap$Outcome[sun.swap$Outcome == "HC"] ← "U"
sun.swap$Outcome[sun.swap$Outcome == "G"] ← "GC"
sun.swap$Outcome ← as.factor(levels(sun.swap$Outcome)[as.numeric(sun.swap$Outcome)])

## lets make the levels more clear for some of the data (race, politics, disposition)
## start with the disposition
levels(sun.swap$Disposition) ← c("C_rem", "D_rem", "Foreman", "Kept", "U_rem",
                                 "S_rem", "Unknown")
## next the political affiliation
sun.swap ← lapply(sun.swap, function(el) {
    if (is.factor(el) & identical(levels(el), LevPol)) {
        levels(el) ← c("Dem", "Ind", "Lib", "Rep", "U")
        el
    } else el})
levels(sun.swap$JPoliticalAff) ← c("Dem", "Ind", "Rep", "U")
## now the race
sun.swap ← lapply(sun.swap, function(el) {
    if (is.factor(el) & identical(levels(el), LevRace)) {
        levels(el) ← c("Asian", "Black", "Hisp", "NatAm", "Other",
                       "U", "White")
        el
    } else el})
```

```r
levels(sun.swap$VictimRace) ← c("Asian", "Black", "Hisp", "NatAm",
                                "U", "White")
levels(sun.swap$JRace) ← c("Black", "Hisp", "NatAm", "U", "White")
levels(sun.swap$DCRace) ← c("Asian", "Black", "NatAm", "Other",
                            "U", "White")
## now the outcome/verdict
levels(sun.swap$Outcome) ← c("Acquittal", "Guilty as Charged",
                             "Guilty of Lesser", "Incomplete", "Mistrial",
                             "U")
## the defense attorney type
levels(sun.swap$DefAttyType) ← c("App Priv", "Public", "Private",
                                 "Ret Priv", "U", "Waived")

## add a guilt indicator
sun.swap$Guilty ← grepl("Guilty", sun.swap$Outcome)

## add a simple indicator of defendant race matching juror race if they are both known
sun.swap$RaceMatch ← sun.swap$Race == sun.swap$DefRace
sun.swap$RaceMatch[sun.swap$Race == "U" | sun.swap$DefRace == "U"] ← NA

## now perform tree classification of crimes
## first cast sun.swap as a data frame
sun.swap ← as.data.frame(sun.swap)
## regularize the charges
chargFact ← as.factor(sun.swap$ChargeTxt)
regCharg ← StringReg(levels(chargFact))[as.numeric(chargFact)]
## classify these into a charge tree and aggregate this at the coarsest level
aggCharg ← treeAgg(stringTree(regCharg, chargeTree))
## these can be further classified into crime classes
crimes.trial ← CrimeClassify(aggCharg, regCharg)
## convert these classes into a factor for the data, start with a generic "other" vector
sun.swap$CrimeType ← rep("Other", nrow(sun.swap))
## now populate it
for (nm in sort(names(crimes.trial))) sun.swap$CrimeType[crimes.trial[[nm]]] ← nm
sun.swap$CrimeType ← as.factor(sun.swap$CrimeType)

## synthesize additional columns
sun.swap ← SynCols(sun.swap)

## now organize this on the juror scale
sun.juror ← UniqueAgg(sun.swap, by = "JurorNumber", collapse = ",")

## Checkpoint 2: the swapped data has been processed and summarized to be on the scale of individual jurors
## save the swapped data
write.csv(sun.swap, "FullSunshine_Swapped.csv", row.names = FALSE)
## and the juror summarized data
saveRDS(sun.juror, "JurorAggregated.Rds")

## summarize by trial, get the unique trials
Trials ← unique(sun.swap$TrialNumberID)
## extract information about these trials, note that grouping occurs on the trial ID, defendant ID, and charge ID levels,
## as the trials frequency involve multiple charges and defendants, which makes them less clean
sun.trial ← aggregate(sun.swap[,TrialVars],
                      by = list(sun.swap$TrialNumberID, sun.swap$DefendantID.DefendantToTrial,
                                sun.swap$ID.Charges),
                      unique)
sun.trial$Group.1 ← NULL
sun.trial$Group.2 ← NULL
sun.trial$Group.3 ← NULL

## summarize the juries by trial as well
sun.jursum ← JurySummarize()

## merge the summaries to the trial sunshine data
```

```r
sun.trialsum ← merge(cbind(TrialNumberID = sun.jursum$Juries$TrialNumberID, sun.jursum$Summaries),
                     sun.trial, all = TRUE)

## notice that the total removed variables are incomplete, try to correct this where possible using the jury
## summarized data above
sun.trialsum$DefRemEst ← RemovedJurorEstimates(sun.trialsum$DefenseTotalRemoved, data = sun.trialsum,
                                               ident = "Gender.DefRem", plot = FALSE)
## perform this same procedure for the prosecution removals
sun.trialsum$ProRemEst ← RemovedJurorEstimates(sun.trialsum$StateTotalRemoved, data = sun.trialsum,
                                               ident = "Gender.ProRem", plot = FALSE)
## synthesize some other variables, simple race indicators
sun.trialsum$DefWhiteBlack ← as.factor(FactorReduce(sun.trialsum$DefRace, tokeep = c("Black", "White", "U")))
sun.trialsum$DefWhiteOther ← as.factor(FactorReduce(sun.trialsum$DefWhiteBlack, tokeep = c("White", "U")))
## the Kullback-Leibler divergence
sun.trialsum$KLdiv ← kldiv(sun.trialsum[,grepl("Jury", names(sun.trialsum))],
                           sun.trialsum[,grepl("Venire", names(sun.trialsum))])

## Checkpoint 3: the data has been set to the trial level and summarized
## save this
saveRDS(sun.trialsum, "TrialAggregated.Rds")
saveRDS(sun.jursum, "AllJuries.Rds")


## CHARGE CLASSIFICATION IMAGES ########
## presented here is the code to generate the appendix charge classification images

## extract relevant charges from the clean sunshine data, avoiding duplicates
## regularize the charges
chargFact.clean ← as.factor(CleanSunshine$Charges$ChargeTxt)
regCharg.clean ← StringReg(levels(chargFact.clean))[as.numeric(chargFact.clean)]
## classify these into a charge tree and aggregate this at the coarsest level
tree.clean ← stringTree(regCharg.clean, chargeTree)

## define padding
crimepad ← 0.01
areatop ← 0.94
hght ← ((0.94 - 0.035) - 5*crimepad)/5
wid ← ((0.975 - 0.025) - 7*crimepad)/6

## add overall box
grid.newpage()
grid.rect(width = 0.95, height = 0.95)
grid.text(label = "Charges", x = 0.025 + crimepad/2, y = 0.975 - crimepad/2, just = c("left","top"))
grid.text(label = "(198)", x = 0.975 - crimepad/2, y = 0.975 - crimepad/2, just = c("right","top"))

## inner crime boxes
xpos ← crimepad + 0.025 + wid/2 + (0:5)*(crimepad + wid)
ypos ← crimepad + 0.025 + hght/2 + (0:4)*(crimepad + hght)
coords ← cbind(rep(xpos, each = 5), rev(ypos))[1:length(tree.clean)-1,]
grid.rect(x = coords[,1], y = coords[,2], width = wid, height = hght)
grid.text(label = names(tree.clean)[1:(length(tree.clean)-1)], x = coords[,1] - (wid - crimepad)/2,
          y = coords[,2] + (hght-crimepad)/2, just = c("left","top"))

## add unclassified and top level counts
unclass ← sapply(tree.clean, function(el) if (is.list(el)) length(el$other) else 0)
grid.text(label = paste0("(", unclass, ")"), x = coords[,1] + (wid - crimepad)/2, y = coords[,2] + (hght-crimepad)/2,
          just = c("right", "top"))
toplev ← sapply(tree.clean, function(el) if (is.list(el)) "" else length(el))
grid.text(label = toplev, x = coords[,1], y = coords[,2])

## add sublist counts
for (ii in 1:length(tree.clean)) {
    if (is.list(tree.clean[[ii]])) {
        currlist ← tree.clean[[ii]]
        len ← length(currlist) - 1
        boty ← coords[ii,2] - hght/2
        newwid ← wid - 2*crimepad
        newhght ← (hght - 0.035 - len*crimepad)/len
        newy ← crimepad + newhght/2 + (0:(len-1))*(newhght + crimepad) + boty
        if (names(tree.clean)[ii] == "assa") {
            grid.rect(x = coords[ii,1], y = newy, width = newwid, height =
                newhght)
            grid.text(label = names(currlist), x = coords[ii,1] - (newwid-
                crimepad)/2, y = newy,
                      just = "left", gp = gpar(fontsize = 8))
            grid.text(sapply(currlist, length), x = coords[ii,1], y = newy, gp =
                gpar(fontsize = 8))
        } else if (names(tree.clean)[ii] == "murder") {
            grid.rect(x = coords[ii,1], y = newy, width = newwid, height =
                newhght)
            grid.text(label = names(currlist), x = coords[ii,1] - (newwid-
                crimepad)/2, y = newy + (newhght-crimepad)/2,
                      just = c("left","top"), gp = gpar(fontsize = 8))
            grid.text(paste0("(", sapply(currlist[1:len], function(el) length(el$
                other)), ")"), x = coords[ii,1] + (newwid-crimepad)/2,
                      y = newy + (newhght-crimepad)/2, just = c("right","top"),
                          gp = gpar(fontsize = 8))
            grid.rect(x = coords[ii,1], y = newy - 0.01, width = newwid - 2*
                crimepad, height = (newhght - 0.02)/2)
            grid.text(rep("att",2), x = coords[ii,1] - (newwid-2*crimepad-
                crimepad)/2, y = newy-0.01,
                      just = "left", gp = gpar(fontsize = 8))
            grid.text(sapply(currlist[1:len], function(el) length(el$att)), x =
                coords[ii,1],
                      y = newy - 0.01, gp = gpar(fontsize = 8))
        } else {
            grid.rect(x = coords[ii,1], y = newy, width = newwid, height =
                newhght)
            grid.text(label = names(currlist), x = coords[ii,1] - (newwid-
                crimepad)/2, y = newy + (newhght-crimepad)/2,
                      just = c("left","top"), gp = gpar(fontsize = 8))
            grid.text(sapply(currlist, length), x = coords[ii,1], y = newy, gp =
                gpar(fontsize = 10))
        }
    }
}

## a small example tree
firstx ← 0.33
secondx ← 0.75
firsty ← c(0.25,0.75)
secondy ← c(0.125,0.375,0.625,0.875)
wd ← 0.3
hg ← 0.1
grid.newpage()
grid.rect(x = firstx, y = firsty, width = wd, height = hg)
grid.rect(x = secondx, y = secondy, width = wd, height = hg)
grid.lines(x = c(0,firstx-wd/2), y = c(0.5,firsty[1]))
grid.lines(x = c(0,firstx-wd/2), y = c(0.5,firsty[2]))
grid.lines(x = c(firstx+wd/2,secondx-wd/2), y = c(firsty[2],secondy[3]))
grid.lines(x = c(firstx+wd/2,secondx-wd/2), y = c(firsty[2],secondy[4]))
grid.lines(x = c(firstx+wd/2,secondx-wd/2), y = c(firsty[1],secondy[1]))
grid.lines(x = c(firstx+wd/2,secondx-wd/2), y = c(firsty[1],secondy[2]))
grid.text(label = c("sex(?=.*offend)", "sex(?=.offense)"), x = firstx, y = firsty
    ,
          gp = gpar(fontsize = 16))
```

```
grid.text(label = c("first|1","second|2","regis","addr"), x = secondx, y = rev(
    secondy),
          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

```
########################################

## THESIS ANALYSIS SCRIPT
## Christopher Salahub
## Sept 26, 2018

########################################

## PACKAGES ###########################
library(readxl)
library(MASS)
library(eikosograms)
library(RColorBrewer)
library(stringr)
library(tm)
library(lme4)
library(lmerTest)

## CONSTANTS ##########################

## start by defining file locations
ThesisDir <- "c:/Users/Chris/Documents/ETH Zurich/Thesis/Data"
SunshineFile <- paste0(ThesisDir, "/JurySunshineExcel.xlsx")
SunshineSheets <- excel_sheets(SunshineFile)

NorthCarFile <- paste0(ThesisDir,
```

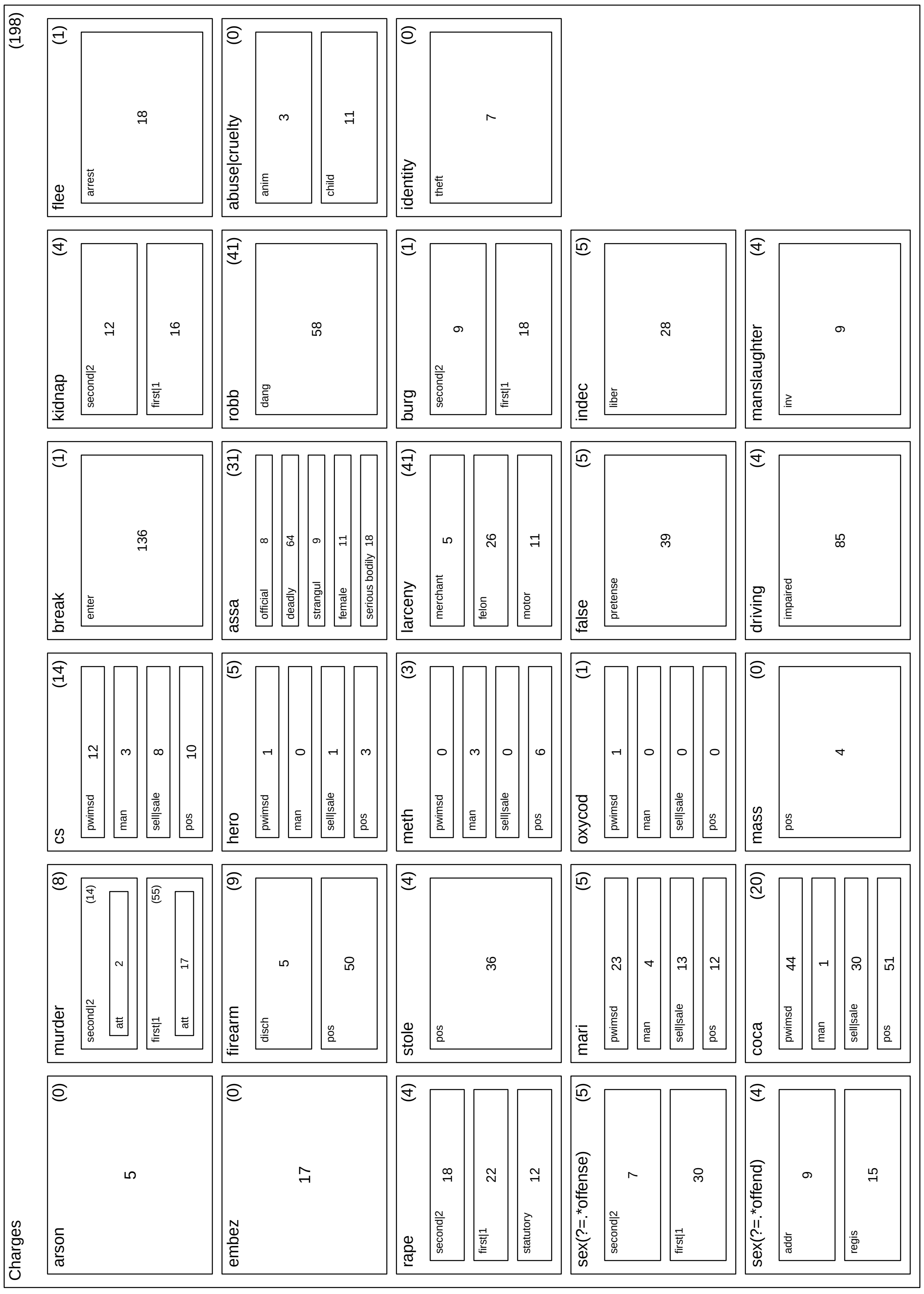
Charges (198)
arson (0) 5
embez (0) 17
rape (4) second|2 18 first|1 22 statutory 12
sex(?=.*offense) (5) second|2 7 first|1 30
sex(?=.*offend) (4) addr 9 regis 15
murder (8) second|2 (14) att 2 first|1 (55) att 17
firearm (9) disch 5 pos 50
stole (4) pos 36
mari (5) pwimsd 23 man 4 sell|sale 13 pos 12
coca (20) pwimsd 44 man 1 sell|sale 30 pos 51
cs (14) pwimsd 12 man 3 sell|sale 8 pos 10
hero (5) pwimsd 1 man 0 sell|sale 1 pos 3
meth (3) pwimsd 0 man 3 sell|sale 0 pos 6
oxycod (1) pwimsd 1 man 0 sell|sale 0 pos 0
mass (0) pos 4
break (1) enter 136
assa (31) official 8 deadly 64 strangul 9 female 11 serious bodily 18
larceny (41) merchant 5 felon 26 motor 11
false (5) pretense 39
driving (4) impaired 85
kidnap (4) second|2 12 first|1 16
robb (41) dang 58
burg (1) second|2 9 first|1 18
indec (5) liber 28
manslaughter (4) inv 9
flee (1) arrest 18
abuse|cruelty (0) anim 3 child 11
identity (0) theft 7

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

```r
                        "/Jury Study Data and Materials/NC Jury Selection Study
                            Database6 Dec 2011.csv")

PhillyFile ← paste0(ThesisDir,
                      "/Voir Dire Data & Codebook/capital_venires.csv")

## next the factor level codes as given in the codebook and regularized here
## regularization: - political affiliation "N" replaced with "I" for all entries
LevRace ← sort(c("Asian","Black","Hisp","NatAm","Other","U","White"))
LevGen ←  sort(c("F","M","U"))
LevPol ←  sort(c("Dem","Lib","Rep","Ind","U"))

## color constants
racePal ← brewer.pal(3, "Set2") # c("steelblue","grey50","firebrick")
whitePal ← c("steelblue","firebrick")
crimePal ← brewer.pal(7, "Set1")
dispPal ← brewer.pal(3, "Set2")


## FUNCTIONS ###########################

## create a plot which visualizes positional data patterns by a categorical
    variable
## could encode density as either box sizes or through alpha levels of colour
## the areal representation breaks the "dimensionality" rule of data in Edward
    Tufte's "The Visual Display of Information",
## to limit the dimensionality of a representation to at most the dimensionality
    of the data itself
## place the legend labels in the largest box instead of off to the side (didn't
    really work...)
posboxplot ← function(x, y, cats, boxcolours = NULL, boxwids = 0.8, alphaencoding
     = TRUE, alphamin = 0.1,
                        areaencoding = FALSE, xlim = range(x) + boxwids*c(-1/
                            1.05,1/1.05), inc.leg = TRUE,
                        ylim = range(y) + boxwids*c(-1/1.05,1/1.05), ...) {
    ## extract the number of categories to be displayed in each small multiple
    ncats ← length(unique(cats))
    ## automatically generate the category colours using color brewer
    if (is.null(boxcolours)) {
        boxcols ← brewer.pal(ncats, "Set2")
        boxcolours ← boxcols
    } else boxcols ← boxcolours
    ## first identify the unique coordinates for the small multiples
    unqPos ← unique(cbind(x,y))
    ## iterate through these, create tables of the categories at each position
    cattabs ← t(apply(unqPos, 1, function(pos) {
        ## generate a count a table
        table(cats[x == pos[1] & y == pos[2]])
    }))
    ## sum these counts to get the total at each position to scale the small
        multiples
    rowcounts ← rowSums(cattabs)
    ## convert the count table to cumulative proportions at each position
    catprops ← t(apply(cbind(0,cattabs/rowcounts), 1, cumsum))
    ## use these and the table of counts to generate the small multiples
    ## first in the case that opacity encodes density
    if (alphaencoding) {
        ## in the opacity-density case, convert the box colours to rgb and
            replicate them as necessary
        boxcols ← col2rgb(rep(boxcols, each = nrow(catprops)), alpha = FALSE)/255
        ## convert back to hex, adding the alpha encoding to control opacity
        boxcols ← rgb(t(boxcols),
                      alpha = rep(round((1-alphamin)*rowcounts/max(rowcounts) +
                          alphamin, digits = 4), times = ncats))
      ## in the case size encodes density, simply replicate the colors for the
          number of positions
    } else boxcols ← rep(boxcols, each = nrow(catprops))
    ## create an empty plot to place the small multiples
    plot(x, y, col = NA, xlim = xlim, ylim = ylim, ...)
    ## determine the width of the small multiple boxes
    if (areaencoding) boxwids ← boxwids*sqrt(rowcounts/max(rowcounts))
    ## define the bottom corner positions of the boxes
    bottomx ← unqPos[,1] - boxwids/2
    bottomy ← unqPos[,2] - boxwids/2
    ## use the bottom corner positions to calculate box extents, with internal
        borders defined as well
    rectx ← bottomx + catprops*boxwids
    recty ← cbind(rep(bottomy, times = ncats), rep(bottomy + boxwids, times =
        ncats))
    ## convert the x coordinates into a list of vectors specifying all positions
    xvec ← lapply(1:(ncats+1), function(n) rectx[,n])
    ## place the rectangles by unlisting this structure correctly
    rect(xleft = unlist(xvec[1:ncats]), ybottom = recty[,1], xright = unlist(xvec
        [2:(ncats+1)]),
         ytop = recty[,2], col = boxcols, border = boxcols)
    ## include a legend if desired
    if (inc.leg) legend(x = "top", legend = colnames(rectx)[-1],fill = boxcolours
        , bty = "n",
                        xpd = NA, horiz = TRUE)
}

## create a function for proportional parallel coordinate plots
## incorporate possibility to display in a non-proportional absolute way
propparcoord ← function(fact, cats, levs = NULL, proportional = TRUE, includerel
    = proportional, ylim = NULL,
                         colpal = NULL, ordering = NULL, legpos = "topleft",
                             brptpos = 1, brwid = 4, ...) {
    ## create the x label
    xnm ← deparse(substitute(fact))
    ## perform a type check
    if (!is.factor(fact)) fact ← as.factor(fact)
    ## check if levs have been supplied
    if (is.null(levs)) levs ← unique(cats)
    ## extract the levels and indices of interest
    levinds ← cats %in% levs
    ## get the length of the categories provided and a table of frequencies
    ctab ← table(as.character(cats[levinds]))
    len ← length(fact)
    lineLen ← length(levels(fact))
    ## check if order is null and allow reordering
    if (is.null(ordering)) ordering ← 1:lineLen
    ## set the ylim and other values based on whether a proportional plot is
        desired
    factab ← table(fact)
    if (proportional) factab ← factab/len else if (is.null(ylim)) ylim ← c(0, max
        (factab))
    ## generate a palette if one is not given
    if (is.null(colpal)) colpal ← brewer.pal(length(ctab), "Set2")
    colpal ← colpal[ordering]
    ## check for missing ylim value
    if (is.null(ylim)) yrng ← c(0,1) else yrng ← ylim
    ## now plot everything
    if (proportional) ynm ← "Proportion" else ynm ← "Count"
    plot(NA, xlim = c(1,lineLen), ylim = yrng, xlab = xnm, ylab = ynm, xaxt = 'n'
        , ...)
    lines(1:lineLen, factab[ordering])
    ## create positions for relative proportions bar chart if this is desired
    if (includerel) {
        ## specify bar chart rectangle bounds
        rectbounds ← seq(0.7, 0.7 - 0.03*(length(ctab)+3), by = -0.03)*yrng[2]
        ## add the reference "total population" bar
        rect(xleft = brptpos, xright = 1+brwid/4, ybottom = rectbounds[1], ytop =
             rectbounds[2], col = "black")
    }
    ## plot lines and relative size rectangles, depending on options
    for (ii in 1:length(ctab)) {
        ## get the counts for the subset selected by ii
        subsetab ← table(fact[cats == names(ctab)[ii]])
        ## set these proportional if desired
        if (proportional) subsetab ← subsetab/ctab[names(ctab)[ii]]
        ## place these in a line
        lines(1:lineLen, subsetab[ordering], col = colpal[ii])
        ## add the appropriate bar if desired
        if (includerel) {
            rect(xleft = brptpos, xright = 1+(brwid/4)*ctab[ii]/len, ybottom =
                rectbounds[ii+1], ytop = rectbounds[ii+2],
                 col = colpal[ii])
        }
    }
    ## add a legend and axis
    legend(x = legpos, legend = c("All", names(ctab)), lty = 1, col = c("Black",
        colpal), title = deparse(substitute(cats)))
    if (includerel) text("Relative Totals", x = 1, y = 0.72*yrng[2], pos = 4)
    axis(1, 1:lineLen, levels(fact)[ordering])
}

## DEPRECATED: a specific case of the above function which has been replaced by
    the following function
## make a specific line plot function
parcoordrace ← function() {
    ## clean up the defwhiteblack variable
    DefWhiteBlack_clean ← as.factor(as.character(gsub(",Other|,U", "", sun.juror$
        DefWhiteBlack)))
    ## first generate the necessary mixed factor
    jurorDef ← with(sun.juror, as.factor(paste(DefWhiteBlack_clean, WhiteBlack,
        sep = ":")))
    ## generate positions to associate these factor levels
    xpos ← rep(1:5, each = 4) + rep(c(-0.21,-0.07,0.07,0.21), times = 5)
    ## choose disposition levels
    displevs ← c("", "Kept", "S_rem", "D_rem", "C_rem")
    nicelevs ← c("All", "Jury", "Pros.", "Def.", "Cause")
    ## create a table based on the mixed factor
    mixtab ← with(sun.juror, lapply(displevs,
                                    function(disp) {
                                        tab ← table(jurorDef[grepl(disp,
                                            Disposition)])/sum(grepl(disp,
                                            Disposition))
                                        wraptab ← c(tab,tab[1:4])
                                        wraptab
                                    }))
    ## define a palette
    colpal ← brewer.pal(length(displevs) - 1, "Set2")
    ## extract the max value for plotting purposes
    maxtab ← max(unlist(mixtab))
    ## plot all tables using different colours
    lapply(1:length(mixtab), function(ind) {
        if (ind == 1) {
            plot(x = xpos, y = mixtab[[ind]], xlim = range(xpos), ylim = c(0,
                maxtab), xlab = "", xaxt = "n", ylab = "Proportion of Data",
                 col = "black", type = 'l', yaxt = 'n')
        } else lines(xpos+0.006*(ind-4)+0.003, mixtab[[ind]], col = colpal[ind
            -1], lty = 2)})
    ## add axes
    axis(side = 2, at = round(seq(0, maxtab, length.out = 3), digits = 2))
    axis(1, at = xpos, labels = rep(c("Black","Other","Unknown","White"), times =
         5))
    axis(1, at = 1:5, labels = c("Black Defendant","Other","Unknown Defendant","
        White Defendant","Black Defendant"),
         pos = -0.05, xpd = NA, tick = FALSE)
    ## add guide lines coloured by disposition
    lapply(2:length(displevs),
           function(ind) {
               sapply(1:20, function(n) rect(xleft = rep(xpos[n],2)+0.006*(ind-4)
                   , xright = rep(xpos[n],2)+0.006*(ind-3),
                                        ybottom = mixtab[[1]][n], ytop =
                                            mixtab[[ind]][n], border =
                                            colpal[ind-1],
                                        col = colpal[ind-1]))
           })
    ## add legend-ish text
    text(x = xpos[1]-0.01, y = mixtab[[2]][1] + 0.0075, labels = nicelevs[2], pos = 2, cex = 0.75, srt = 90,
         col = colpal[1])
    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])
    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])
    text(x = xpos[1]+0.03, y = mixtab[[5]][1], labels = nicelevs[5], pos = 1, cex = 0.75, srt = 90, col = colpal[4])
}

## the better version of the above function, takes an arbitrary three-way contingency table and plots the different conditional
## probabilities of the desired margins
parcoordracev2 ← function(tabl = NULL, tracemar = 1, deslev = NULL, wid = 0.02, addlines = FALSE,
                          space = 0.025, testlines = FALSE, ...) {
    ## in the default case (no table provided), look at the key race relationships, as these motivated this study
    if (is.null(tabl)) {
        ## for cleanliness, remove those jurors with unknown races
        temp.juror ← sun.juror[sun.juror$WhiteBlack != "U" & sun.juror$DefWhiteBlack != "U",]
        temp.juror$WhiteBlack ← as.factor(as.character(temp.juror$WhiteBlack))
        temp.juror$DefWhiteBlack ← as.factor(as.character(temp.juror$DefWhiteBlack))
        ## perform the same operation for defendant race
        temp.juror$DefWhiteBlack ← as.factor(as.character(gsub(",Other|,U", "", temp.juror$DefWhiteBlack)))
        ## make a table of disposition and both races
        outcometab ← table(temp.juror[,c("Disposition", "DefWhiteBlack", "WhiteBlack")])
    } else { ## if a table is provided simply copy it to the internal "outcometab" argument
        outcometab ← tabl
    }
    ## determine the "non-trace" margins; these specify margins which are combined to define the cases to which the horizontal
    ## line segments correspond
    nontrace ← (1:3)[1:3 != tracemar]
    ## get the dimension names
    tabnames ← dimnames(outcometab)
    ## handle a null desired level setting
    if (is.null(deslev)) deslev ← 1:length(tabnames[[tracemar]])
    ## create a palette
    temPal ← brewer.pal(length(deslev), "Set2")
    ## calculate the conditional probability distribution of outcome given non-trace margins using outcometab
    condout ← apply(outcometab, nontrace, function(margin) margin/sum(margin))
    ## and the sums
    marsums ← apply(outcometab, nontrace, sum)
    ## extract the desired levels from the margin of interest, write this flexibly to allow programmatic margin extraction later
    ## first define the local environment as the calling environment
    evEnv ← environment()
    ## create the language object
    condoutinds ← condoutcall ← quote(condout[,,])
    ## place the levels of interest as the subset argument
    condoutcall[[tracemar+2]] ← deslev
    ## evaluate this to subset the data
    condout ← eval(condoutcall, envir = evEnv)
    ## rename some useful values for ease of reference
    ## the dimensionality of the data
    dims ← dim(condout)
    ## the number of horizontal segments
    nseg ← prod(dims[nontrace])
    ## the number of 'inner' combination values
    innern ← dims[nontrace[1]]
    ## the number of 'outer' combination values (inner and outer refer to axis label positions)
    outern ← dims[nontrace[2]]
    ## add padding between segments to space everything nicely
    ## first replicate the trace margin sums to create a vector of the correct size for the horizontal line generation
    tempx ← rep(c(marsums), times = c(rep(2, nseg-1),1))
    ## replace every even element with the desired space size, then the cumulative sum automatically spaces
    tempx[2*(1:(nseg-1))] ← space*rep(c(rep(1,innern-1),3), length.out = nseg-1)*sum(marsums)
    ## take the cumulative sum to get the positions
    xpos_line ← c(0, cumsum(tempx)/sum(tempx))
    ##xpos ← rep(1:dims[nontrace[2]], each = dims[nontrace[1]]) +
    ##    rep(seq(-0.2, 0.2, length.out = dims[nontrace[1]]), times = dims[nontrace[2]])
    ##xpos ← cumsum(marsums)/sum(marsums)
    ## create the empty plot region
    plot(NA, xlim = range(xpos_line), ylim = c(0, max(condout[,,])), xaxt = 'n', xlab = "",
         ylab = "Conditional Probability", ...)
    ## calculate and plot the horizontal lines at the mean values
    ## the old way: calculating the mean of the conditional distributions
    meanline ← apply(condout, nontrace, mean)
    ## this way does not correspond to the hypothesis being tested: that there is no preference for race displayed by
    ## either side, rather under this hypothesis, the expected rate of each combination is given by the product of the
    ## overall rate for each side and the proportion of the venire of
    for(ii in 1:length(meanline)) lines(c(xpos_line[2*ii-1],xpos_line[2*ii]), rep(meanline[ii],2))
    ## add the vertical lines for each cell, save the positions for later
    xpos ← sort(unlist(lapply(1:length(deslev), function(ind) {
        ## first extract the relevant margin to give the y values
        tempind ← condoutinds
        tempind[[tracemar + 2]] ← ind
        yvals ← eval(tempind, envir = evEnv)
        ## use the horizontal line positions and the index to place the vertical lines
        ##adjx ← xpos + wid*(ind - (1/2)*(1 + length(deslev)))
        adjx ← xpos_line[2*(1:nseg)-1] + (ind-1)*diff(xpos_line)[2*(1:nseg)-1]/(dims[tracemar] - 1)
        ## add the corresponding end points
        points(adjx, yvals, col = temPal[ind], pch = 19)
        ## for aesthetics exclude lines if confidence intervals are plotted
        if (!testlines) for (ii in 1:length(adjx)) lines(x = rep(adjx[ii],2), y = c(meanline[ii], yvals[ii]),
                                                          lty = 2, col = temPal[ind])
        ## if trace lines (parallel axis plot) are desired plot these
        if (addlines) lines(adjx, yvals, col = temPal[ind], lty = 3)
        ##rect(xleft = adjx - (1/2)*wid, xright = adjx + (1/2)*wid, ybottom = meanline,
        ##     ytop = yvals, col = temPal[ind])
        return(adjx)
    })))
    ## now add the axes
    ## first determine axis label positions
    axpos ← sapply(1:nseg, function(ind) mean(xpos_line[c((2*ind-1),2*ind)]))
    outrpos ← sapply(1:outern, function(ind) mean(range(xpos_line[(2*(ind-1)*innern + 1):(2*ind*innern)])))
    ## add the axis ticks for the inner labels
    axis(1, at = axpos, labels = rep("", length(axpos)))
    ## add the labels to the inner axis
    axis(1, at = axpos, tick = FALSE, labels = rep(tabnames[[nontrace[1]]], times = outern), cex.axis = 0.7,
         pos = -0.02*max(condout))
    ## add the labels for the outer axis
    axis(1, at = outrpos, labels = tabnames[[nontrace[2]]], xpd = NA,
         tick = FALSE, pos = -0.08*max(condout[,,]))
    ## provide the axis title to give context
    axis(1, at = mean(range(xpos)), xpd = NA, tick = FALSE, pos = -0.15*max(
        condout),
         labels = paste0("Inner label: ", names(tabnames)[nontrace[1]], " | Outer
              label: ", names(tabnames)[nontrace[2]]))
    ## add testing lines if desired
    if (testlines) {
        ## get x positions
        errpos ← xpos
        ##errpos[3*(1:nseg) - 1] ← axpos
        ## reformat y positions
        erry ← c(condout)
        ## define error bar extensions
        ext ← c(-0.005, 0.005)
        ## add bars at each position
        invisible(sapply(1:length(erry), function(pos) {
            ## rename the conditional probability for readability
            p ← erry[pos]
            ## extract the relevant margin count
            n ← marsums[floor((pos-1)/dims[tracemar]) + 1]
            ## calculate the binomial error size
            err ← 2*sqrt((p/n)*(1-p))
            ## and the appropriate colour
            errcol ← temPal[(pos-1) %% dims[tracemar] + 1]
            ## add vertical lines and horizontal end lines
            lines(x = errpos[pos] + ext, y = rep(p + err, 2), col = adjustcolor(
                errcol, alpha.f = 0.5))
            lines(x = errpos[pos] + ext, y = rep(p - err, 2), col = adjustcolor(
                errcol, alpha.f = 0.5))
            lines(x = rep(errpos[pos], 2), y = c(p + err, p - err), col =
                adjustcolor(errcol, alpha.f = 0.5))
            ##lines(, meanline - sqrt((meanline/(3*marsums))*(1-3*meanline)), lty
                 = 2)
            ##lines(xpos, meanline + sqrt((meanline/(3*marsums))*(1-3*meanline)),
                 lty = 2)
            }))
    }
    ## add a legen to explain the colours
    legend(x = "top", horiz = TRUE, legend = tabnames[[tracemar]][deslev], col =
        temPal, inset = -0.04, cex = 0.7,
           fill = temPal, bg = "white", xpd = NA)
    ##invisible(sapply((0:4)*0.05, function(val) lines(x = c(0,max(xpos)+1), y =
        rep(val,2), col = "white", lwd = 2)))
}

## the better version of the above function, takes an arbitrary three-way
    contingency table and plots the different conditional
## probabilities of the desired margins
testplot ← function(tabl = NULL, tracemar = 1, deslev = NULL, wid = 0.02,
    addlines = FALSE,
                           space = 0.025, testlines = FALSE, expected = NULL,
                               ...) {
    ## in the default case (no table provided), look at the key race
        relationships, as these motivated this study
    if (is.null(tabl)) {
        ## for cleanliness, remove those jurors with unknown races
        temp.juror ← sun.juror[sun.juror$WhiteBlack != "U" & sun.juror$
            DefWhiteBlack != "U",]
        temp.juror$WhiteBlack ← as.factor(as.character(temp.juror$WhiteBlack))
        temp.juror$DefWhiteBlack ← as.factor(as.character(temp.juror$
            DefWhiteBlack))
        ## perform the same operation for defendant race
        temp.juror$DefWhiteBlack ← as.factor(as.character(gsub(",Other|,U", "",
            temp.juror$DefWhiteBlack)))
        ## make a table of disposition and both races
        outcometab ← table(temp.juror[,c("Disposition", "DefWhiteBlack", "
            WhiteBlack")])
    } else { ## if a table is provided simply copy it to the internal "outcometab
        " argument
        outcometab ← tabl
    }
    ## determine the "non-trace" margins; these specify margins which are
        combined to define the cases to which the horizontal
    ## line segments correspond
    nontrace ← (1:3)[1:3 != tracemar]
    ## get the dimension names
    tabnames ← dimnames(outcometab)
    ## handle a null desired level setting
    if (is.null(deslev)) deslev ← 1:length(tabnames[[tracemar]])
    ## create a palette
    temPal ← brewer.pal(length(deslev), "Set2")
    ## calculate the conditional probability distribution of outcome given non-
        trace margins using outcometab
    condout ← apply(outcometab, nontrace, function(margin) margin/sum(margin))
    ## and the sums
    marsums ← apply(outcometab, nontrace, sum)
    ## extract the desired levels from the margin of interest, write this
        flexibly to allow programmatic margin extraction later
    ## first define the local environment as the calling environment
    evEnv ← environment()
    ## create the language object
    condoutinds ← condoutcall ← quote(condout[,,])
    ## place the levels of interest as the subset argument
    condoutcall[[tracemar+2]] ← deslev
    ## evaluate this to subset the data
    condout ← eval(condoutcall, envir = evEnv)
    ## rename some useful values for ease of reference
    ## the dimensionality of the data
    dims ← dim(condout)
    ## the number of horizontal segments
    nseg ← prod(dims)
    ## the number of 'inner' combination values
    innern ← dims[nontrace[1]]
    ## the number of 'outer' combination values (inner and outer refer to axis
        label positions)
    outern ← dims[nontrace[2]]
    ## the trace dimension as well
    tracen ← dims[tracemar]
    ## add padding between segments to space everything nicely
    ## first replicate the trace margin sums to create a vector of the correct
        size for the horizontal line generation
    tempx ← rep(c(marsums)/tracen, times = c(rep(tracen + 1, outern*innern-1),
        tracen))
    ## replace certain elements with the  desired space size, then the cumulative
         sum automatically spaces
    tempx[(tracen+1)*(1:(innern*outern - 1))] ← space*rep(c(rep(1,innern-1),3),
        length.out = innern*outern-1)*sum(marsums)
    ## take the cumulative sum to get the positions
    xpos_line ← c(0, cumsum(tempx)/sum(tempx))
    ## get the middle positions using the filter function
    xpos ← c(filter(xpos_line, filter = c(1/2,1/2)))
    ## remove midsections of padding spaces
    xpos ← xpos[-(tracen + 1)*(1:(innern*outern - 1))]
    xpos ← xpos[-length(xpos)]
    ## use the xpos_line to get widths of sections
    xposwids ← diff(xpos_line)[-(tracen + 1)*(1:(innern*outern-1))]
    ## if the expected values are missing, take the original expectation: a
        uniform distribution conditioned on both
    ## races
    if (is.null(expected)) expected ← rep(apply(condout, nontrace, mean), each =
        tracen)
    ## create the empty plot region
    plot(NA, xlim = range(xpos_line), ylim = c(0, max(condout[,,])), xaxt = 'n',
        xlab = "",
         ylab = "Conditional Probability", ...)
    ## plot each line and its corresponding expectation
    invisible(lapply(1:length(xpos), function(ind) {
        ## plot the horizontal line using the relevant values
        lines(x = xpos[ind] + xposwids[ind]*c(-1/2,1/2), y = rep(expected[ind],2)
            )
        ## add the point
        points(x = xpos[ind], y = condout[ind], col = temPal[((ind - 1) %% tracen) + 1], pch = 19)
        ## for aesthetics exclude lines if confidence intervals are plotted
        if (!testlines) lines(x = rep(xpos[ind], 2), y = c(expected[ind], condout[ind]),
                              col = temPal[((ind-1) %% tracen) + 1], lty = 2)
    }))
    ## now add the axes
    ## first determine axis label positions
    axpos ← sapply(1:(innern*outern), function(n) mean(xpos[(tracen*(n-1) + 1):(tracen*n)]))
    outrpos ← sapply(1:outern, function(n) mean(xpos[(tracen*innern*(n-1) + 1):(tracen*innern*n)]))
    ## add the axis ticks for the inner labels
    axis(1, at = axpos, labels = rep("", length(axpos)))
    ## add the labels to the inner axis
    axis(1, at = axpos, tick = FALSE, labels = rep(tabnames[[nontrace[1]]], times = outern), cex.axis = 0.7,
         pos = -0.02*max(condout))
    ## add the labels for the outer axis
    axis(1, at = outrpos, labels = tabnames[[nontrace[2]]], xpd = NA,
         tick = FALSE, pos = -0.08*max(condout[,,]))
    ## provide the axis title to give context
    axis(1, at = mean(range(xpos_line)), xpd = NA, tick = FALSE, pos = -0.15*max(condout),
         labels = paste0("Inner label: ", names(tabnames)[nontrace[1]], " | Outer label: ", names(tabnames)[nontrace[2]]))
    ## add testing lines if desired
    if (testlines) {
        ## get x positions
        errpos ← xpos
        ##errpos[3*(1:nseg) - 1] ← axpos
        ## reformat y positions
        erry ← c(condout)
        ## define error bar extensions
        ext ← c(-0.005, 0.005)
        ## add bars at each position
        invisible(sapply(1:length(erry), function(pos) {
            ## rename the conditional probability for readability
            p ← erry[pos]
            ## extract the relevant margin count
            n ← marsums[floor((pos-1)/dims[tracemar]) + 1]
            ## calculate the binomial error size
            err ← 2*sqrt((p/n)*(1-p))
            ## and the appropriate colour
            errcol ← temPal[(pos-1) %% dims[tracemar] + 1]
            ## add vertical lines and horizontal end lines
            lines(x = errpos[pos] + ext, y = rep(p + err, 2), col = adjustcolor(errcol, alpha.f = 0.5))
            lines(x = errpos[pos] + ext, y = rep(p - err, 2), col = adjustcolor(errcol, alpha.f = 0.5))
            lines(x = rep(errpos[pos], 2), y = c(p + err, p - err), col = adjustcolor(errcol, alpha.f = 0.5))
            }))
    }
    ## add a legen to explain the colours
    legend(x = "top", horiz = TRUE, legend = tabnames[[tracemar]][deslev], col = temPal, inset = -0.04, cex = 0.7,
           fill = temPal, bg = "white", xpd = NA)
}

## back to back histogram plot
back2backh ← function(data1, data2, cols = NULL, legnames = NULL, ...) {
    ## get the data1 and data2 names for the legend if no others are provided
    if (is.null(legnames)) legnames ← c(deparse(substitute(data1)), deparse(substitute(data2)))
    ## generate and save the histograms of the data
    hist1 ← hist(data1, plot = FALSE)
    hist2 ← hist(data2, breaks = hist1$breaks, plot = FALSE)
    ## use these to generate some necessary plotting parameters
```

```r
    maxden <- max(c(hist1$density, hist2$density))
    nbins <- length(hist1$density)
    ## generate colours if none are provided
    if (is.null(cols)) cols <- c("steelblue","firebrick")
    ## create an empty plot area
    plot(NA, xlim = c(-maxden, maxden), ylim = range(hist1$breaks), xlab = "
        Density", xaxt = 'n', ...)
    ## add vertical separating line
    abline(v = 0)
    ## add an axis
    axispos <- round(seq(0, maxden, length.out = 3),2)
    axis(side = 1, labels = c(axispos[3:2], axispos), at = c(-axispos[3:2],
        axispos))
    ## plot the histograms back-to-back
    rect(xleft = -hist1$density, ybottom = hist1$breaks[1:nbins],
         xright = rep(0, nbins), ytop = hist1$breaks[2:(nbins+1)], col = cols[1])
    rect(xleft = rep(0, nbins), ybottom = hist2$breaks[1:nbins],
         xright = hist2$density, ytop = hist2$breaks[2:(nbins+1)], col = cols[2])
    ## add a legend
    legend(x = 0, y = max(hist1$breaks), legend = legnames, fill = cols, horiz =
        TRUE, xpd = NA, xjust = 0.5, yjust = 0,
           bg = "white")
}

## a function to re-level factor variables to make mosaic plots cleaner (useful
    helper generally)
MatRelevel <- function(data) {
    temp <- lapply(data, function(el) if (is.factor(el)) as.factor(levels(el)[as.
        numeric(el)]) else el)
    temp <- as.data.frame(temp)
    names(temp) <- names(data)
    temp
}

## a simple helper to convert multiple factor levels into a single 'other' level
FactorReduce <- function(vals, tokeep) {
    chars <- as.character(vals)
    ## simply replace elements
    chars[!grepl(paste0(tokeep, collapse = "|"), chars)] <- "Other"
    chars
}


## DATA INSPECTION #####################

## load the data
if ("FullSunshine_Swapped.csv" %in% list.files(ThesisDir)) {
    sun.swap <- read.csv(paste0(ThesisDir, "/FullSunshine_Swapped.csv"))
} else source(paste0(ThesisDir, "/DataProcess.R"))
FullSunshine <- read.csv(paste0(ThesisDir, "/FullSunshine.csv"))

## summarize onto the correct scale, the jurors
if ("JurorAggregated.Rds" %in% list.files(ThesisDir)) {
    sun.juror <- readRDS(paste0(ThesisDir, "/JurorAggregated.Rds"))
} else sun.juror <- UniqueAgg(sun.swap, by = "JurorNumber", collapse = ",")

## also load the data summarized onto the trial scale
if ("TrialAggregated.Rds" %in% list.files(ThesisDir)) {
    sun.trialsum <- readRDS(paste0(ThesisDir, "/TrialAggregated.Rds"))
} else warning(paste0("No trial aggregated data found in ", ThesisDir))

## there are two juries without charges or other info (noted in the early data
    cleaning but kept for other analysis), remove these
sun.trialsum <- sun.trialsum[!(sun.trialsum$TrialNumberID %in% c("590-128","710-01
    ")),]

## display information about juror rejection tendencies
mosaicplot(Race ~ Disposition, data = sun.juror, las = 2, shade = TRUE)

## create a race filtered data set
```

```r
sun.raceknown ← sun.juror[sun.juror$Race != "U" & sun.juror$DefRace != "U",]
sun.raceknown$DefWhiteBlack ← gsub(",U", "", sun.raceknown$DefWhiteBlack)
sun.raceknown ← MatRelevel(sun.raceknown)

## try plotting these
mosaicplot(Race ~ PerempStruck, data = sun.raceknown, main = "Race vs. Removal", shade = TRUE, las = 2)
mosaicplot(Race ~ Disposition, data = sun.raceknown, main = "Race by Trial Status", shade = TRUE, las = 2)
mosaicplot(Race ~ DefStruck, data = sun.raceknown, main = "Race by Defence Removal", shade = TRUE, las = 2)
mosaicplot(Race ~ ProStruck, data = sun.raceknown, main = "Race by Prosecution Removal", shade = TRUE, las = 2)
mosaicplot(Race ~ CauseRemoved, data = sun.raceknown, main = "Race by Removal with Cause", shade = TRUE, las = 2)
## it seems that there are significantly different strike habits between the defense and prosecution, but that
## generally the system does not strike at different rates on average
## recall the paper "Ideological Imbalance and the Peremptory Challenge"
par(mfrow = c(1,2))
mosaicplot(Race ~ PoliticalAffiliation, data = sun.raceknown[sun.raceknown$Gender == "M",],
           main = "Affiliation and Race (Men)", shade = TRUE, las = 2)
mosaicplot(Race ~ PoliticalAffiliation, data = sun.raceknown[sun.raceknown$Gender == "F",],
           main = "Affiliation and Race (Women)", shade = TRUE, las = 2)
mosaicplot(Race ~ DefStruck, data = sun.raceknown[sun.raceknown$Gender == "M",],
           main = "Defense Removals and Race (Men)", shade = TRUE, las = 2)
mosaicplot(Race ~ DefStruck, data = sun.raceknown[sun.raceknown$Gender == "F",],
           main = "Defense Removals and Race (Women)", shade = TRUE, las = 2)
mosaicplot(Race ~ ProStruck, data = sun.raceknown[sun.raceknown$Gender == "M",],
           main = "Prosecution Removals and Race (Men)", shade = TRUE, las = 2)
mosaicplot(Race ~ ProStruck, data = sun.raceknown[sun.raceknown$Gender == "F",],
           main = "Prosecution Removals and Race (Women)", shade = TRUE, las = 2)
par(mfrow = c(1,1))
## maybe the same forces are at play here, compare to simulation?
## alternatively, the strong relationship between race and political affiliation provides motivation for even an
## unbiased lawyer to preferentially strike one race or the other

## these mosaic plots can be confusing, and seemed ineffective upon first presentation, try parallel axis plots
## instead
## begin with an overall plot displaying the data at a high level
parcoordrace()
parcoordracev2(deslev = c(1,2,5))
## but are these differences significant?
parcoordracev2(deslev = c(1,2,5), testlines = TRUE)

## the independence we want to test here is that of (Race, Disposition)|(Defendant Race)
## filter the data to remove small categories
sun.chitest ← sun.raceknown
sun.chitest$Disposition ← gsub("U_rem", "Unknown", gsub("Foreman", "Kept", sun.chitest$Disposition))
## start by generating a table
dispTab ← table(sun.chitest[,c("DefWhiteBlack", "Disposition", "WhiteBlack")])
## now apply chi-square tests across the proper margin, start by simply generating the residuals
dispRes ← lapply(setNames(1:dim(dispTab)[1], dimnames(dispTab)[[1]]), function(ind) {
    ## extract the two way table of this index
    tab ← dispTab[ind,,]
    tabdf ← dim(tab) - 1
    ## calculate the expected values
    exp ← outer(rowSums(tab), colSums(tab))/sum(tab)
    ## and residuals
    resids ← (tab - exp)/sqrt(exp)
    ## calculate the observed chi-sq value
    chival ← sum(resids^2)
    ## and the p value
    pval ← 1 - pchisq(chival, df = tabdf[1]*tabdf[2])
    ## return these in a list
    list(pval = pval, chisq = chival, df = tabdf[1]*tabdf[2], residuals = resids)
})
## so, there is a significant difference in behaviour at the 5% level, and it is highly significant for white and black jurors

## but these results do not control for much, there could be many factors confounding this result
## first create a new data set for the model building
sun.jurmod ← sun.raceknown
sun.jurmod$DefVisMin ← sun.jurmod$DefWhiteBlack != "White"
sun.jurmod$VisMin ← sun.jurmod$WhiteBlack != "White"
sun.jurmod$DefStruck ← as.logical(sun.jurmod$DefStruck)
sun.jurmod$ProStruck ← as.logical(sun.jurmod$ProStruck)
## now the tricky part, predicting the rejection of a potential juror based on a host of factors, the problem is that we must
## perform multinomial regression on the data, but this multinomial regression makes comparison of certain parameters
## impossible, i.e. there is no mathematical way to compare the impact of race for prosecution and defense rejection statistically
## start by building separate defense and prosecution rejection models
mod.def1 ← glm(DefStruck ~ Race + DefRace + Gender + DefGender + CrimeType + DefAttyType + PoliticalAffiliation,
               data = sun.jurmod, family = binomial)
## very poorly fit model, but the reason should be fairly clear, the crime data in particular has very specific and small
## classes, try building up the model instead, and using the simpler race variable
mod.def2 ← glm(DefStruck ~ WhiteBlack*DefWhiteBlack, data = sun.jurmod, family = binomial)
mod.def3 ← update(mod.def2, formula = DefStruck ~ WhiteBlack + DefWhiteBlack)

## idea: instead of multinomial regression, do poisson regression on the dispTab above, this allows comparisons
sun.rdat ← data.frame(DefRace = rep(c("Black", "Other", "White"), times = 12),
                       Disposition = rep(rep(c("C_rem", "D_rem", "Kept", "S_rem"), each = 3), times = 3),
                       Race = rep(c("Black", "Other", "White"), each = 12),
                       Count = c(dispTab[,c("C_rem","D_rem","Kept","S_rem"),]))
## estimate the saturated model first
mod.rsat ← glm(Count ~ DefRace*Disposition*Race, family = poisson, data = sun.rdat)
## now test if the final interaction term can be removed
mod.r1 ← update(mod.rsat, formula = Count ~ DefRace*Disposition + DefRace*Race + Disposition*Race)
## look at the significance
1 - pchisq(mod.r1$deviance, mod.r1$df.residual)
## so, quite clearly, we cannot remove the three way interaction from the model, as it is highly significant
## the interpretation: the distribution of strikes, kept, etc. depends on both the venire member race and the defendant race
## still, this is perhaps not precise enough, if we change this data to only delineate between those kept and the behaviour of
## the lawyers
sun.rdat2 ← sun.rdat[sun.rdat$Disposition != "C_rem",]
mod.rsat2 ← glm(Count ~ DefRace*Disposition*Race, family = poisson, data = sun.rdat2)

## this is an interesting result, but perhaps it is related to political affiliation (as indicated by the ideological balance
## paper)
## create a table to test this hypothesis
dispTab.pol ← table(MatRelevel(sun.chitest[!(sun.chitest$PoliticalAffiliation %in% c("Lib","U")),
                                           c("Disposition","PoliticalAffiliation","WhiteBlack","DefWhiteBlack")])
                    )
dispTab.pol ← dispTab.pol[c("C_rem","D_rem","Kept","S_rem"),,,]
```

```
## convert to a data frame for fitting
sun.pdat ← data.frame(Disp_ = rep(c("C_rem", "D_rem", "Kept", "S_rem"), times =
    27),
                        Pol_ = rep(rep(c("Dem", "Ind", "Rep"), each = 4), times =
                            9),
                        Race_ = rep(rep(c("Black", "Other", "White"), each = 12),
                            times = 3),
                        Def_ = rep(c("Black", "Other", "White"), each = 36),
                        Count = c(dispTab.pol[,,,]))
## fit a model analogous to those fit above, now controlled for political choices
      in disposition
mod.psat ← glm(Count ∼ Def_*Disp_*Race_+ Pol_*Disp_, family = poisson, data = sun
    .pdat)
## now remove the third order interaction in the model
mod.psattest ← update(mod.psat, formula = Count ∼ Def_*Disp_*Race_ + Pol_*Disp_ -
      Def_:Disp_:Race_)
## test the models
anova(mod.psat, mod.psattest)
1 - pchisq(66.734, 12)

## ideological imbalance, look at politics
parcoordracev2(table(MatRelevel(sun.raceknown[sun.raceknown$PoliticalAffiliation
    != "U",
                                              c("Disposition","
                                                  PoliticalAffiliation","
                                                  WhiteBlack")])),
               deslev = c(1,2,5))

## use radial axis plots to view the lawyers tendencies, especially those who act
      as both defence and prosecution lawyers
## maybe remove the top lawyers and remodel
## look at the most prolific lawyers for both sides
## subset the data to only lawyers with one case to remove the lawyer dependency

## however, this suggests another question: is this strategy actually successful?
      That is, does there appear to
## be a relation between the number of peremptory challenges and the court case
    outcome?
## this may be difficult, there are a lot of factors to consider:
##                  - the lawyer and their track record
##                  - how to judge the success/failure of the case
## see if the presence of challenges is related to the verdict
mosaicplot(PerempStruck ∼ Guilty, data = sun.swap, main = "Strikes by Guilt",
    shade = TRUE)
## on the level of jurors, this is certainly not the case, but this is not the
    correct scale for the question being
## asked, this question will be addressed again in the case-summarized data

## a third obvious question is a comparison of which races strike or keep which
    others, used the synthesized variable
## above to try and identify this
mosaicplot(Race ∼ StruckBy, data = sun.raceknown, shade = TRUE, main = "Race of
    Juror to Race Removing Juror",
           las = 2)
mosaicplot(Race ∼ StruckBy, data = sun.raceknown[sun.raceknown$StruckBy != "Not
    Struck",], shade = TRUE,
           main = "Race to Race Removing (Only Removed)", las = 2)
## this plot shows no large systematic deviation between the races in their
    rejection habits, this suggests, that
## the rejection that occurs is not as simple as a group identity check
## this might be the wrong race to check, though, perhaps we are better comparing
      the defendant and victim races to
## strike habits
par(mfrow = c(1,3))
mosaicplot(Race ∼ DefRace, data = sun.raceknown[as.logical(sun.raceknown$
    DefStruck),], shade = TRUE,
           main = "Race of Defense-Struck Jurors to Defendant Race", las = 2)
mosaicplot(Race ∼ DefRace, data = sun.raceknown[as.logical(sun.raceknown$
    ProStruck),], shade = TRUE,
           main = "Race of Prosecution-Struck Jurors to Defendant Race", las = 2)
```

```r
mosaicplot(Race ~ DefRace, data = sun.raceknown, las = 2, shade = TRUE, main = "
    Race of Defendant to Venire Race")
par(mfrow = c(1,1))
## this makes the defense look as if they are not racist, though the comparison
    to the venire distributions in the third
## panel makes that clearer
## these distributions to the venire distribution relative to defendant race,
    first combine the smaller races into one
## category to make the plot less noisy and more identifiable
## now look at how the two behave relative in their rejections and their
    acceptance
eikos(WhiteBlack ~ DefWhiteBlack + DefStruck, data = sun.raceknown, xlab_rot =
    90,
      main = "Defense Challenges by Race of Venire Member and Defendant")
eikos(WhiteBlack ~ DefWhiteBlack + ProStruck, data = sun.raceknown, xlab_rot =
    90,
      main = "Prosecution Challenges by Race of Venire Member and Defendant")
## very interesting, the prosecution seems far more aggressive than the defense
sun.raceknown$DefWhiteBlack[sun.raceknown$DefWhiteBlack == "Black,U"] <- "Black"
sun.raceknown$DefWhiteBlack <- as.factor(as.character(sun.raceknown$DefWhiteBlack)
    )
mosaicplot(DefStruck ~ DefWhiteBlack + WhiteBlack, dir = c("v","v","h"), data =
    sun.raceknown, shade = TRUE, las = 2,
           xlab = "Defendant Race and Defence Removals", ylab = "Juror Race",
               main = "Defence Removal by Defendant Race")
mosaicplot(ProStruck ~ DefWhiteBlack + WhiteBlack, dir = c("v","v","h"),  data =
    sun.raceknown, shade = TRUE, las = 2,
           xlab = "Defendant Race and Prosecution Removals", ylab = "Juror Race",
                main = "Prosecution Removal by Defendant Race")

## that result is very interesting, the defense strike rates when conditioned on
    defendant race show no racial
## preference, with a preference to reject white jurors regardless of defendant,
    but those of the prosecution do,
## maybe by victim race?
mosaicplot(Race ~ VictimRace, data = sun.raceknown[as.logical(sun.raceknown$
    DefStruck),], shade = TRUE,
           main = "Race of Defense-Struck Jurors to Defendant Race", las = 2)
mosaicplot(Race ~ VictimRace, data = sun.raceknown[as.logical(sun.raceknown$
    ProStruck),], shade = TRUE,
           main = "Race of Prosecution-Struck Jurors to Defendant Race", las = 2)
## hard to see anything there, the majority of victim races are unknown, maybe
    looking at the races removed by defense
## attorney type
mosaicplot(DefAttyType ~ Race, data = sun.raceknown[as.logical(sun.raceknown$
    DefStruck),], shade = TRUE, las = 2,
           main = "Race of Defense-Struck Jurors to Defense Attorney Type")
mosaicplot(WhiteBlack ~ DefAttyType, data = sun.raceknown[as.logical(sun.
    raceknown$DefStruck),], shade = TRUE, las = 2,
           main = "Race of Defense-Strick Jurors to Defense Attorney Type")
eikos(WhiteBlack ~ DefWhiteBlack + DefAttyType, data = sun.raceknown[as.logical(
    sun.raceknown$DefStruck),],
      xlab_rot = 90)
## so what have we seen above is that the prosecuton and defense seem to behave
    very differently in their jury selection
## tactics, the defense seems to reject white individuals at a high rate
    regardless of the defendant, while the prosecution
## seems to prefer the rejection of venire members of the same race as the
    defendant

## this last plot shows that different types of lawyers may have different
    strategies, suggests a new investigation:
## that of lawyer strategy and success based on lawyer tendencies, aggregating by
     trial first will be easiest

## load the jury summaries
if ("AllJuries.Rds" %in% list.files(ThesisDir)) {
    sun.jursum <- readRDS(paste0(ThesisDir, "/AllJuries.Rds"))
} else cat(paste0("No file 'AllJuries.Rds' in ", ThesisDir))

```

```
## now look at removals across trials for defense and prosecution
with(sun.trialsum, plot(jitter(DefRemEst, factor = 2), jitter(ProRemEst, factor =
     2), pch = 20,
                        xlab = "Defense Strike Count (jittered)", ylab = "
                            Prosecution Strike Count (jittered)",
                        col = adjustcolor(racePal[as.numeric(DefWhiteBlack)],
                            alpha.f = 0.3)))
abline(0,1)
legend(x = "topleft", legend = levels(sun.trialsum$DefWhiteBlack), col = racePal,
     pch = 20, title = "Defendant Race")
## this is only somewhat informative, it is difficult to see any patterns, use
    the custom posboxplot function
## first encode relative size of point by alpha blending
with(sun.trialsum, posboxplot(DefRemEst, ProRemEst, DefWhiteBlack, boxcolours =
    racePal, xlab = "Defense Strike Count",
                              ylab = "Prosecution Strike Count", boxwids = 0.8,
                                  alphamin = 0.05))
## next by area, another encoding option in this function
with(sun.trialsum, posboxplot(DefRemEst, ProRemEst, DefWhiteBlack, boxcolours =
    racePal, xlab = "Defense Strike Count",
                              ylab = "Prosecution Strike Count", alphaencoding =
                                  FALSE, areaencoding = TRUE))

## break apart in more detail for the defense
DefStruckMeans ← with(sun.trialsum, sapply(levels(DefWhiteBlack),
                                            function(rc) c(mean((Race.DefRem.
                                                Black/Race.Venire.Black)[
                                                DefWhiteBlack == rc],
                                                                na.rm = TRUE),
                                                           mean((Race.DefRem.
                                                               White/Race.Venire.
                                                               White)[
                                                               DefWhiteBlack ==
                                                               rc],
                                                                na.rm = TRUE))))
with(sun.trialsum, plot(Race.DefRem.Black/Race.Venire.Black, Race.DefRem.White/
    Race.Venire.White, pch = 20,
                        xlab = "Black Venire Proportion Struck", ylab = "White
                            Venire Proportion Struck",
                        xlim = c(0,1), ylim = c(0,1), main = "Defense Strike
                            Proportions",
                        col = adjustcolor(racePal[as.numeric(DefWhiteBlack)],
                            alpha.f = 0.2)))
abline(0,1)
points(DefStruckMeans[1,], DefStruckMeans[2,], col = racePal, pch = 4, cex = 2,
    lwd = 1.5)
legend(x = "topright", title = "Defendant Race", col = c(racePal,"black"), pch =
    c(rep(20,3),4), bg = "white",
        legend = c(levels(sun.trialsum$DefWhiteBlack),"Mean"))
## hard to see the patterns at the lines, jitter the proportions
with(sun.trialsum, plot(Race.DefRem.Black/Race.Venire.Black + runif(nrow(sun.
    trialsum), min = -0.03, max = 0.03),
                        Race.DefRem.White/Race.Venire.White, pch = 20,
                        xlab = "Black Venire Proportion Struck", ylab = "White
                            Venire Proportion Struck",
                        xlim = c(0,1), ylim = c(0,1), main = "Defense Strike
                            Proportions",
                        col = adjustcolor(racePal[as.numeric(DefWhiteBlack)],
                            alpha.f = 0.1)))


## and for the prosecution
ProStruckMeans ← with(sun.trialsum, sapply(levels(DefWhiteBlack),
                                            function(rc) c(mean((Race.ProRem.
                                                Black/Race.Venire.Black)[
                                                DefWhiteBlack == rc],
                                                                na.rm = TRUE),
                                                           mean((Race.ProRem.
                                                               White/Race.Venire.
                                                               White)[
```

```r
                                                               DefWhiteBlack == rc],
                                                                na.rm = TRUE))))
with(sun.trialsum, plot(Race.ProRem.Black/Race.Venire.Black, Race.ProRem.White/Race.Venire.White, pch = 20,
                        xlab = "Black Venire Proportion Struck", ylab = "White Venire Proportion Struck",
                        xlim = c(0,1), ylim = c(0,1), main = "Prosecution Strike Proportions",
                        col = adjustcolor(racePal[as.numeric(DefWhiteBlack)], alpha.f = 0.2)))
abline(0,1)
points(ProStruckMeans[1,], ProStruckMeans[2,], col = racePal, pch = 4, cex = 2, lwd = 1.5)
legend(x = "topright", title = "Defendant Race", col = c(racePal,"black"), pch = c(rep(20,3),4), bg = "white",
       legend = c(levels(sun.trialsum$DefWhiteBlack),"Mean"))
## again hard to see, try jittering
with(sun.trialsum, plot(Race.ProRem.Black/Race.Venire.Black + runif(nrow(sun.trialsum), min = -0.03, max = 0.03),
                        Race.ProRem.White/Race.Venire.White, pch = 20,
                        xlab = "Black Venire Proportion Struck", ylab = "White Venire Proportion Struck",
                        xlim = c(0,1), ylim = c(0,1), main = "Prosecution Strike Proportions",
                        col = adjustcolor(racePal[as.numeric(DefWhiteBlack)], alpha.f = 0.1)))

## both of these plots show a much higher proportion of the black venire is usually struck for both sides, an unsurprising result
## given the the black venire was shown to be smaller in the aggregate statistics, looking at raw counts next:
## for the defense
with(sun.trialsum, plot(jitter(Race.DefRem.Black, factor = 2), jitter(Race.DefRem.White, factor = 2), pch = 20,
                        xlab = "Black Venire Strike Count (jittered)", ylab = "White Venire Strike Count (jittered)",
                        xlim = c(0,13), ylim = c(0,13), main = "Defense Strike Counts",
                        col = adjustcolor(racePal[as.numeric(DefWhiteBlack)], alpha.f = 0.2)))
legend(x = "topright", title = "Defendant Race", col = racePal, pch = 20, bg = "white", legend = levels(sun.trialsum$DefWhiteBlack))
## use custom plot here
with(sun.trialsum, posboxplot(Race.DefRem.Black, Race.DefRem.White, DefWhiteBlack, racePal,
                              xlab = "Black Venire Strike Count", ylab = "White Venire Strike Count",
                              xlim = c(0,13), ylim = c(0,13), main = "Defense Strike Counts"))
with(sun.trialsum, posboxplot(Race.DefRem.Black, Race.DefRem.White, DefWhiteBlack, racePal,
                              xlab = "Black Venire Strike Count", ylab = "White Venire Strike Count",
                              xlim = c(0,13), ylim = c(0,13), main = "Defence Strike Counts",
                              alphaencoding = FALSE, areaencoding = TRUE))

## for the prosecution
with(sun.trialsum, plot(jitter(Race.ProRem.Black, factor = 1.2), jitter(Race.ProRem.White, factor = 1.2), pch = 20,
                        xlab = "Black Venire Strike Count (jittered)", ylab = "White Venire Strike Count (jittered)",
                        xlim = c(0,8), ylim = c(0,8), main = "Prosecution Strike Counts",
                        col = adjustcolor(racePal[as.numeric(DefWhiteBlack)], alpha.f = 0.2)))
legend(x = "topright", title = "Defendant Race", col = racePal, pch = 20, bg = "white", legend = levels(sun.trialsum$DefWhiteBlack))
## more of the custom plot
```

```r
with(sun.trialsum, posboxplot(Race.ProRem.Black, Race.ProRem.White, DefWhiteBlack
    , racePal,
                              xlab = "Black Venire Strike Count", ylab = "White
                                  Venire Strike Count",
                              xlim = c(0,13), ylim = c(0,13), main = "Prosecution
                                   Strike Counts"))
with(sun.trialsum, posboxplot(Race.ProRem.Black, Race.ProRem.White, DefWhiteBlack
    , racePal,
                              xlab = "Black Venire Strike Count", ylab = "White
                                  Venire Strike Count",
                              xlim = c(0,13), ylim = c(0,13), main = "Prosecution
                                   Strike Counts",
                              alphaencoding = FALSE, areaencoding = TRUE))

## interesting, this shows some patterns in lawyer behaviour at the trial level

## so there are some obvious patterns we can see in the aggregated data and in
    the individual cases, see if these affect outcomes
with(sun.trialsum, plot(DefRemEst ~ Outcome))
with(sun.trialsum, plot(ProRemEst ~ Outcome))
## nothing obvious there, but there is no control for charges/crime type

## compare these to other variables
mosaicplot(DefRace ~ CrimeType, data = sun.trialsum, las = 2, main = "Crime and
    Race", shade = TRUE)
mosaicplot(Outcome ~ CrimeType, data = sun.trialsum, las = 2, main = "Crime and
    Outcome", shade = TRUE)
boxplot(DefRemEst ~ CrimeType, data = sun.trialsum)
with(sun.trialsum, posboxplot(as.numeric(CrimeType), DefRemEst, DefWhiteBlack,
    racePal, xaxt = "n",
                              ylab = "Defense Strike Count", xlab = "Crime Type")
                                   )
axis(side = 1, at = 1:7, labels = levels(sun.trialsum$CrimeType))
boxplot(ProRemEst ~ CrimeType, data = sun.trialsum)
with(sun.trialsum, posboxplot(as.numeric(CrimeType), ProRemEst, DefWhiteBlack,
    racePal, xaxt = "n",
                              ylab = "Prosecution Strike Count", xlab = "Crime
                                  Type"))
axis(side = 1, at = 1:7, labels = levels(sun.trialsum$CrimeType))

## try using the positional boxplots
with(sun.trialsum, posboxplot(DefRemEst, ProRemEst, CrimeType, crimePal))
## too many classes, maybe try drug, sex, theft, other
sun.trialsum$DrugSexTheft <- as.factor(FactorReduce(sun.trialsum$CrimeType, tokeep
     = c("Drug","Sex","Theft")))
with(sun.trialsum, posboxplot(DefRemEst, ProRemEst, DrugSexTheft, boxcolours =
    brewer.pal(4, "Set1")))
## also summarize this for the juror data
sun.juror$DrugSexTheft <- as.factor(FactorReduce(sun.juror$CrimeType, tokeep = c("
    Drug","Sex","Theft")))

## try something different, plot the tendency of the lawyers themselves
## idea: horizontal axis is lawyers, vertical is strikes
LawyerTends <- lapply(unique(c(sun.trialsum$DefAttyName,sun.trialsum$ProsName)),
                      function(name) list(Prosecution = sun.trialsum$ProRemEst[
                          sapply(sun.trialsum$DefAttyName,
                                                                                 function
                                                                                     (
                                                                                     nms
                                                                                     )

                                                                                     name

                                                                                     %
                                                                                     in
                                                                                     %

                                                                                     nms
                                                                                     )
                                                                                     ],
                                                  Defense = sun.trialsum$DefRemEst[sapply
                                                      (sun.trialsum$ProsName,
                                                                                              function
                                                                                                  (
                                                                                                  nms
                                                                                                  )

                                                                                                  name

                                                                                                  %
                                                                                                  in
                                                                                                  %

                                                                                                  nms
                                                                                                  )
                                                                                                  ])
                                                                                                  )
## order these by those who did both, then defense, then prosecution
LawyerOrder ← order(sapply(LawyerTends, function(lst) {
    lstlens ← sapply(lst, function(el) length(el) > 0)
    if (all(lstlens)) {
        0
    } else if (lstlens[[2]]) {
        1
    } else 2}
    ))
## reorder the lawyer tendencies
LawyerTends ← LawyerTends[LawyerOrder]
## plot these
plot(NA, xlim = c(1,length(LawyerTends)), ylim = c(0, max(unlist(LawyerTends), na
    .rm = TRUE)))
invisible(lapply(1:length(LawyerTends), function(ind) {
    vals ← LawyerTends[[ind]]
    points(rep(ind, length(vals$Defense)), vals$Defense, col = adjustcolor("
        steelblue", alpha.f = 0.1), pch = 20)
    points(rep(ind, length(vals$Prosecution)), vals$Prosecution, col =
        adjustcolor("red", alpha.f = 0.1), pch = 20)
}))
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.