



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 None

Adviser: Prof. Dr. Marloes Maathuis

Preface

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

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

Abstract

Short summary of my thesis.

Contents

Notation	ix
1 Introduction	1
2 Peremptory Challenges	3
2.1 Jury Selection Procedures	3
2.2 The Role of the Jury	4
2.3 Modern Peremptory Challenge Controversy	5
2.4 The Role of the Peremptory Challenge	6
2.5 History	7
2.5.1 Pre-English History	7
2.5.2 In English Law (1066–1988)	7
2.5.3 In American Law (ca. 1700–1986)	8
2.5.4 In Canadian Law (1867–1988)	8
3 Data	9
3.1 Jury Sunshine Project	9
3.2 North Carolina Data	9
3.3 Philadelphia Data	9
3.4 Data Cleaning	9
3.4.1 Sunshine Data	9
4 Analysis	11
4.1 Arguments for Peremptory Challenges	11
4.2 Modelling	12
4.3 To include a picture	16
4.4 To make a proof	16
4.5 To include R code	16
4.6 Other information	17
5 Summary	19
5.1 Future Work	19
Bibliography	21
A Complementary information	25
A.1 Including R code with verbatim	25
A.2 Data Processing Code	26
A.3 Using Sweave to include R code (and more) in your report	39
B Yet another appendix....	41
B.1 Description	41
B.2 Tables	41
Epilogue	43

List of Figures

4.1	Strike Tendency by Racial Combination Sunshine	13
4.2	Political Affiliation by Race and Gender (Sunshine)	14
4.3	Lawyer Experience (Sunshine)	15
4.4	Geyser data: binned histogram, Silverman's and another kernel	16
4.5	Geyser data: binned histogram, Silverman's and another kernel	16

List of Tables

B.1 Test results	41
----------------------------	--------------------

Notation and Terms

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

Prosecution/State The legal representation which argues for conviction

Defence The legal representation which argues against conviction

Court Reference to the judge, prosecution, and defence

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

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

Accused/Defendant The individual on trial for a crime

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

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

Litigants The accuser and the accused

Chapter 1

Introduction

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

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

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

This paper will proceed in five parts. Chapter 2 provides a brief history of the practice of peremptory challenges in jury trials, in particular explaining their original motivation, past implementations, and how they have developed in the United States, the United Kingdom, and Canada. Chapter 3 proceeds to discuss the three data sets obtained, explaining the sources and collection methods before detailing the cleaning and preprocessing. Chapter ?? 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, 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. 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 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 objecting to a challenge to use a *prima facie* argument which must be countered by a race-neutral reason that satisfies the judge. If no such reason can be supplied, the challenge would not be allowed. This created a new challenge 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 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 \(1997\)](#))¹.

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 \(2016\)](#)².

This defence suggests two modern justifications for the peremptory challenge. The first is that of removing venire members with “extreme” bias, and the second is the creation of a jury which is composed of jurors mutually acceptable to both the defense and the

¹It 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 \(2012\)](#).

²Such 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).

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 \(1997\)](#), with additional context and information certain details provided by [von Moschzisker \(1921\)](#), [Forsyth \(1994\)](#), [Brown, McGuire, and Winters \(1978\)](#), and [Brown \(2000\)](#). Information regarding the history of the Canadian system was provided by [Brown \(2000\)](#) and [Petersen \(1993\)](#).

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 Scandinavia, 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 its 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”³.

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 infallibility 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. From a maximum of 35 challenges allowed at their peak in the fourteenth century, the number of challenges allowed only decreased over time until their abolition in 1988 (discussed in [2.4](#)).

2.5.3 In American Law (ca. 1700–1986)

[von Moschzisker \(1921\)](#), [Hoffman \(1997\)](#), and [Van Dyke \(1977\)](#) all agree that the early English settlers that came to North America accepted the jury system with peremptory challenges as common law.

2.5.4 In Canadian Law (1867–1988)

³For a detailed explanation of this system see [Hoffman \(1997\)](#) and [Brown \(2000\)](#)

Chapter 3

Data

3.1 Jury Sunshine Project

3.2 North Carolina Data

3.3 Philadelphia Data

3.4 Data Cleaning

3.4.1 Sunshine Data

The data collected in North Carolina proved invaluable to this project [Wright et al. \(2018\)](#).

Problem: some columns of the data contained only NA values Solution: `lapply` to remove these uninformative columns

Problem: relational database provided did not have all data in one joined table Solution: creation of `CleaningMerge` function: a wrapper for `merge` which provides information about the mismatches which may be present in the two merged tables

Problem: inconsistently coded levels, e.g. inconsistent case or “?” instead of “U” for unknowns Solution: forcing levels to be uppercase and the replacement of obvious mis-specified levels

Problem: some columns seem to have swapped values, e.g. the gender column should be one of “M”, “F”, or “U” and the political affiliation column should be one of “D”, “R”, “I”, or “U”, but some individuals have the gender recorded as “R” and political affiliation as “M” Solution: the creation of the `IdentifySwap` function, which has two arguments: a data set and the acceptable or correct levels for the variables in the data set. It then identifies rows which have candidate swaps and presents them for review

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. The first is whether the results found by previous analyses which did not use statistics are statistically significant. Additionally, we may wonder whether the most common arguments posed in favour of peremptory challenge are satisfied in this data.

4.1 Arguments for Peremptory Challenges

The primary argument stated in favour of the continued use of peremptory challenges is that of the 'levelling' of the bias of the jury. [Cite Canadian news articles here](#) The argument, essentially states that peremptory challenges are necessary to remove those jurors which are somehow abnormally biased but which are not eligible for removal by cause, or have not been removed by cause out of error.

While the argument of recourse for an incorrect judgement of a challenge with cause is certainly valid, it seems unnecessary in light of the appeals process which already exists. Regardless, a precise comment on the validity of this statement cannot be easily made. The second assertion, however, permits a precise and straightforward mathematical analysis.

In the Jury Sunshine data, for example, the proportion of venire members rejected by peremptory challenge is roughly 0.43. In what sense can such a large proportion of the venire be judged as extreme?

A secondary argument is that of the creation of a jury which is mutually acceptable by giving both sides the privilege of removing any jurors they do not want assessing their case. The multiple American supreme court cases which address peremptory challenges and the outcome of the Gerald Stanley murder trial demonstrate quite clearly that this noble goal is not executed in practice. Rather, the privilege is a point of weakness that allows the politicization cases by doing the precise opposite of what it purports to achieve. It creates juries which are unacceptable to one party and society at large.

4.2 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:

$$\log E[y_{ijk}] = \mathbf{x}_{ijk}\beta = \beta_o + \beta_R x_{i..} + \beta_D x_{.j.} + \beta_S x_{..k} + \beta_{R:D} x_{i..} x_{.j.} + \beta_{R:S} x_{i..} x_{..k} + \beta_{D:S} x_{.j.} x_{..k} + \beta_{R:D:S} x_{i..} x_{.j.} x_{..k} \quad (4.2.0.1)$$

Where $x_{i..}$ indicates the race level of the ijk cell, and $x_{.j.}, x_{..k}$ are defined analogously for the defendant race and disposition. The interaction terms then serve to answer questions about the racial pattern of strikes which is utilized by each party given the defendant race. Most interesting to this investigation is the third order interaction term. This term indicates a significant difference in racial strike patterns given the party striking and the defendant race. In other words, this term accounts for different patterns for the different parties which are not independent of the defendant race.

When this term is tested using a nested model without the third order interaction, the third order interaction is found to be significant. This suggests that not only do the patterns present in the different parties vary, but they vary differently for different defendant races. This dependence can be viewed using a novel graphic presented in Figure 4.1.

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 combination¹.

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 literature [Wright et al. \(2018\)](#), but the addition of defendant race allows us to make a stronger statement, as this pattern remains across defendant races. It also adds nuance, however, as the race of the defendant has a clear impact on the lengths of the bars for both the defense and prosecution. The prosecution

¹Generated 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

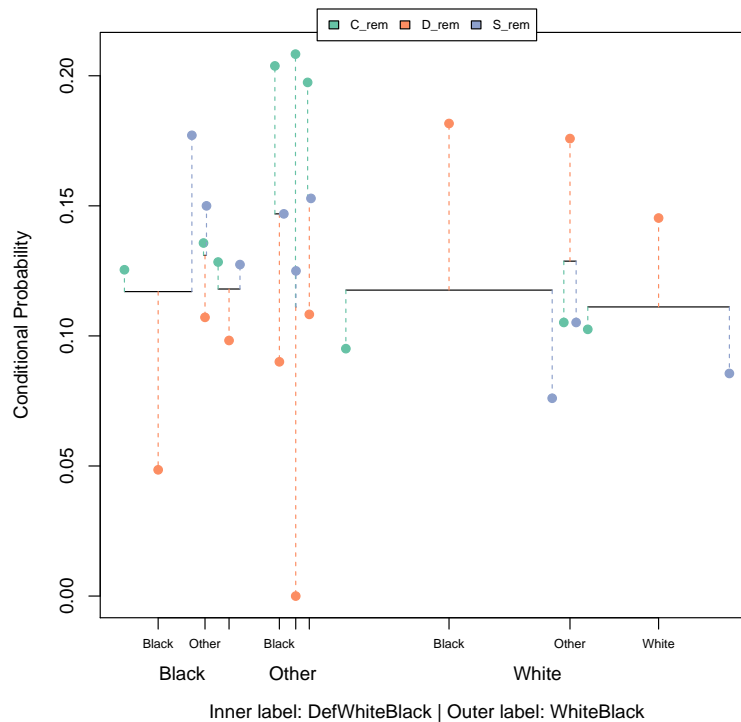


Figure 4.1: Parallel coordinate plot of racial strike tendencies

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 4.2 [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 response 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

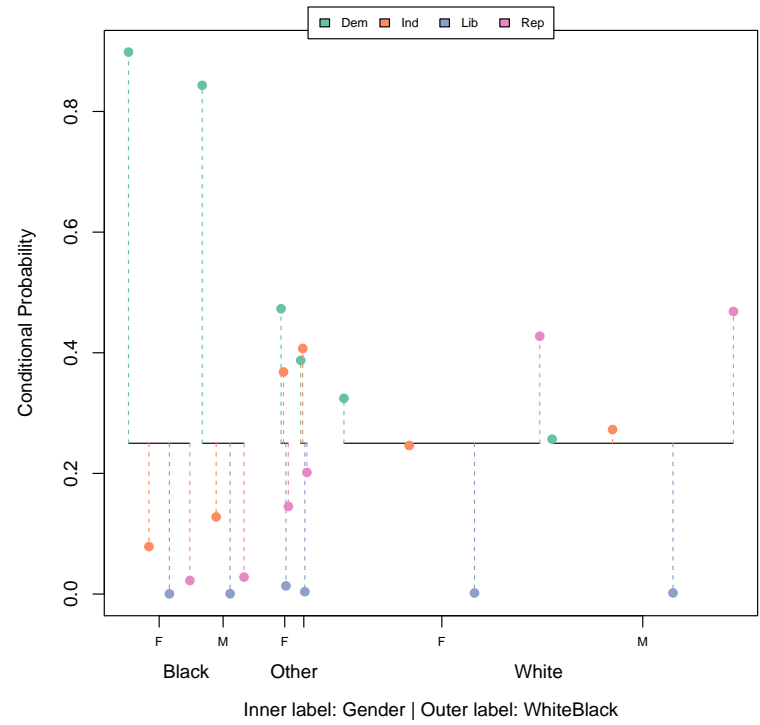


Figure 4.2: Conditional probabilities of political affiliation by race and gender

plotted in back-to-back histograms as shown in Figure 4.3.

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.

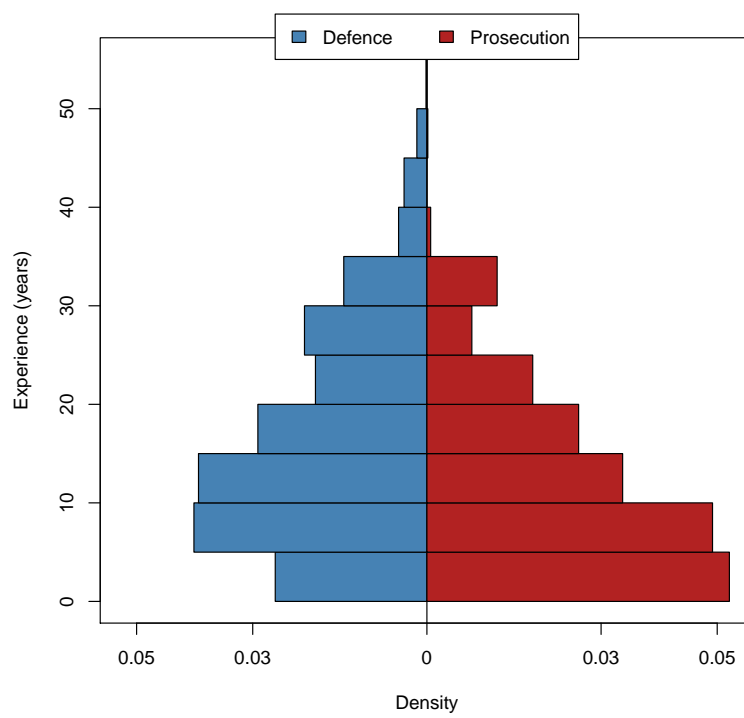


Figure 4.3: Distributions of lawyer experience for prosecutors and defence attorneys

4.3 To include a picture

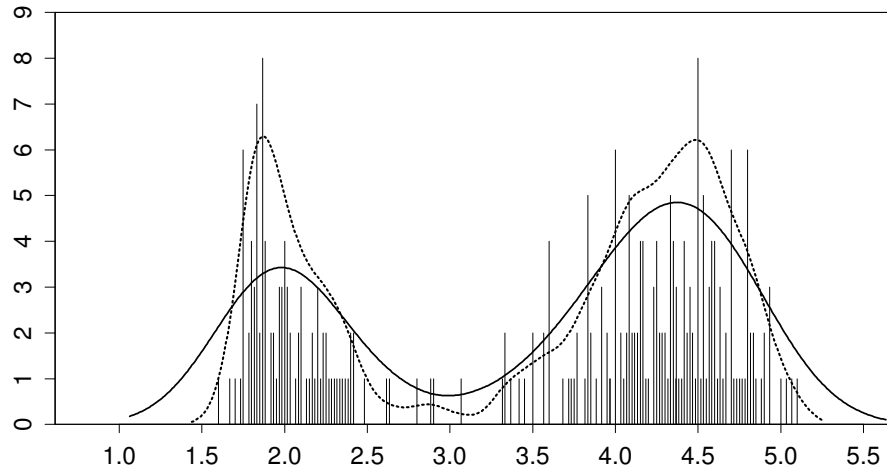


Figure 4.4: Old Faithful Geyser eruption lengths, $n = 272$; binned data and two (Gaussian) kernel density estimates ($\times 10$) with $h = h^* = .3348$ and $h = .1$ (dotted).

Or also with `includegraphics`:

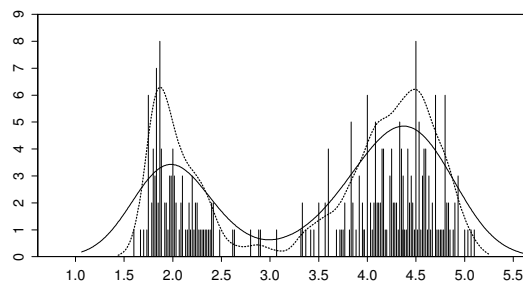


Figure 4.5: Old Faithful Geyser eruption lengths, $n = 272$; binned data and two (Gaussian) kernel density estimates ($\times 10$) with $h = h^* = .3348$ and $h = .1$ (dotted).

4.4 To make a proof

Proof. $1 + 1 = 2$

□

4.5 To include R code

See information in [Appendix A](#).

4.6 Other information

Put a text between quotes: make sure to use nice quotes, such as “quote”.

Cite a document in the bibliography (an example here): [Author and Author \(tion\)](#). Or mention that [Hampel](#) (a person) or [Stahel and Weisberg](#) (two persons) have already done quite a bit work.

Referencing a different part of your work: please refer to Appendix [A](#).

Chapter 5

Summary

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

5.1 Future Work

Possible ways to extend the work.

Bibliography

- 42nd Parliament of Canada (2018a, March). Bill C-75: An Act to Amend the Criminal Code, Youth Criminal Justice Act and other Acts and to make consequential amendments to other Acts. <http://www.justice.gc.ca/eng/csj-sjc/pl/charte-charte/c75.html>.
- 42nd Parliament of Canada (2018b, November). Bill C75. LEGISinfo. <http://www.parl.ca/LegisInfo>.
- Attorney General's Office of the United Kingdom (2012, November). Jury vetting right of stand-by guidelines. <https://www.gov.uk/guidance/jury-vetting-right-of-stand-by-guidelines-2>.
- Author, F. and S. Author (year of publication). Title of the article. *Journal where the article has been published volume of the journal*(issue number), firstpage–lastpage.
- Baldus, D. C., G. Woodworth, D. Zuckerman, and N. A. Weiner (2001). The Use of Peremptory Challenges in Capital Murder Trials: A Legal and Empirical Analysis. *University of Pennsylvania Journal of Constitutional Law* 3(1).
- Brown, F. L., F. T. McGuire, and M. S. Winters (1978). The peremptory challenge as a manipulative device in criminal trials: Traditional use or abuse. *New England Law Review* 14, 192.
- Brown, R. B. (2000). Challenges for cause, stand-asides, and peremptory challenges in the nineteenth century. *Osgoode Hall Law Journal* 38(3), 453–494.
- Burgess, G. (1992). The divine right of kings reconsidered. *The English Historical Review* 107(425), 837–861.
- 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.
- 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.
- Hampel, F. R. (1985). The breakdown points of the mean combined with some rejection rules. *Technometrics* 27(2), 95–107.
- Hans, V. P. and N. Vidmar (1986). *Judging the Jury* (1 ed.). Plenum Press.

- Harris, K. (2018, February). Liberals review jury selection process after Boushie case uproar. CBC News. <https://www.cbc.ca/news/politics/jury-selection-diversity-indigenous-1.4531792>.
- Hasan, N. R. (2018, April). Eliminating peremptory challenges makes trials less fair. The Star. <https://www.thestar.com/opinion/contributors/2018/04/10/eliminating-peremptory-challenges-make-trials-less-fair.html>.
- Hoffman, M. B. (1997). Peremptory Challenges Should Be Abolished: A Trial Judge's Perspective. *The University of Chicago Law Review* 64(3), 809.
- Iacobucci, F. (2013). First Nations Representation on Ontario Juries: Report of the Independent Review Conducted by the Honourable Frank Iacobucci. Ministry of the Attorney General. Accessed: https://www.attorneygeneral.jus.gov.on.ca/english/about/pubs/iacobucci/First_Nations_Representation
- 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.
- Miriam-Webster (2019a). Miriam-Webster Dictionary Online. Accessed: <https://www.merriam-webster.com/dictionary/venire>.
- Miriam-Webster (2019b). Miriam-Webster Dictionary Online. Accessed: <https://www.merriam-webster.com/dictionary/voir%20dire>.
- Morehead, J. W. (1994). When a peremptory challenge is no longer peremptory: Batson's unfortunate failure to eradicate invidious discrimination from jury selection. *DePaul Law Review* 43, 625.
- Page, A. (2005). Batson's blind spot: Unconscious stereotyping and the peremptory challenge. *Boston University Law Review* 85, 155.
- Petersen, C. (1993). Institutionalized racism: The need for reform of the criminal jury selection process. *McGill Law Journal* 38(1).
- Quenneville, G. (2018, February). What happened on Gerald Stanley's farm the day Colten Boushie was shot, as told by witnesses. CBC News. <https://www.cbc.ca/news/canada/saskatoon/what-happened-stanley-farm-boushie-shot-witnesses-colten-gerald-1.4520214>.
- Quenneville, G. and J. Warick (2018, February). Shouts of 'murderer' in courtroom after Gerald Stanley acquitted in Colten Boushie shooting. CBC

- News. <https://www.cbc.ca/news/canada/saskatoon/gerald-stanley-colten-boushie-verdict-1.4526313>.
- 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>.
- Stahel, W. and S. Weisberg (1991). *Directions in Robust Statistics and Diagnostics*, 2 vol. N. Y.: Springer-Verlag.
- Statistics Canada (2018, November). Table 35-10-0061-01: Crime severity index and weighted clearance rates, police services in Saskatchewan.
- Supreme Court of Canada (1991). R. v. sherratt. Supreme Court Judgments. SCC Case Number: 21501; Accessed: <https://scc-csc.lexum.com/scc-csc/scc-csc/en/item/734/index.do?q=21501>.
- Supreme Court of the United States (1965). Swain v. Alabama. Accessed: <https://supreme.justia.com/cases/federal/us/380/202/>.
- Supreme Court of the United States (1986). Batson v. Kentucky. Accessed: <https://www.law.cornell.edu/supremecourt/text/476/79>.
- 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).
- Woolley, A. (2018). An Ethical Jury? Reflections on the Acquittal of Gerald Stanley for the Murder/Manslaughter of Colten Boushie. *Slaw: Canada's online legal magazine*. Accessed: <http://www.slaw.ca/2018/02/20/an-ethical-jury-reflections-on-the-acquittal-of-gerald-stanley-for-the-murder-manslaughter-of-colten-boushie/>.
- Wright, R. F., K. Chavis, and G. S. Parks (2018, October). The Jury Sunshine Project: Jury Selection Data as a Political Issue. *University of Illinois Law Review* 2018(4), 1407.
- Zinchuk, B. (2018, March). Both sides wrong about Stanley trial. Prince George Citizen. <https://www.princegeorgecitizen.com/opinion/editorial/both-sides-wrong-about-stanley-trial-1.23199321>.

Appendix A

Complementary information

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

A.1 Including R code with verbatim

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

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

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

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

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

A.2 Data Processing Code

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

```

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

```

```

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

```

```

111  ## extract the levels of the columns of interest to check if there are any
      potential swaps
112  swapCheck ← all(sapply(1:length(colInds),
113                      function(ind) identical(sort(levels(as.factor(data[,
      colInds[ind]]))),
114                                                  sort(CorrectLevs[[ind]])))
115
116  ## if no swaps are present end this check
117  if (swapCheck) {
118      cat("No errors found, exiting.")
119      return(data)
120  }
121  ## if errors are found, further investigate them
122  ## identify potential rows
123  ## first those which have elements out of place
124  SwapPoss ← sapply(1:length(colInds),
      function(ind) !(data[,colInds[ind]] %in% CorrectLevs[[ind]]))
125
126  ## now rows containing unknown entries
127  Unknown ← sapply(1:length(colInds),
      function(ind) data[,colInds[ind]] == "U")
128
129  ## identify potential swaps by row
130  Swaps ← apply(SwapPoss, 1, function(row) sum(row) > 1)
131  ## identify the potential errors
132  PotErr ← apply(SwapPoss, 1, function(row) sum(row) == 1)
133  ## use the unknowns to account for some errors
134  UnkInd ← apply(Unknown, 1, any)
135  FalErr ← PotErr & UnkInd
136  ## identify the indices to investigate
137  SwapInds ← which(Swaps|FalErr)
138  ErrInds ← which(PotErr & !UnkInd)
139  ## communicate to the user and ask for input
140  cat("There are ", sum(Swaps|FalErr), " swaps to check\n", sep = "")
141  cat("Additionally, it seems there are ", sum(PotErr & !UnkInd), " errors in
      entries\n", sep = "")
142
143  ## unless automated
144  if (auto) ErrorReturn ← TRUE else ErrorReturn ← as.logical(readline("Return
      the errors? (T/F): "))
145
146  ## now, if there are possible swaps investigate them
147  if (sum(Swaps|FalErr) != 0) {
148      ## create a temporary storage structure
149      tempRows ← data[SwapInds, colInds]
150      tempRows ← as.data.frame(lapply(tempRows, function(var) levels(var)[as.
      numeric(var)]),
151                              stringsAsFactors = FALSE)
152
153      ## loop through and populate this
154      for (ii in 1:nrow(tempRows)) {
155          ## inspect the row
156          print(tempRows[ii,])
157          ## suggest corrections, first generate matches
158          candComb ← lapply(tempRows[ii,],
159                          function(el) which(sapply(CorrectLevs,
160                                                  function(levs) el %in%
161                                                  levs)))
162
163          reps ← unlist(lapply(candComb, length))
164          ## now generate all swap combinations
165          candComb[[1]] ← rep(candComb[[1]], each = max(reps[-1]))
166          candComb ← as.data.frame(candComb, row.names = NULL)
167          ## identify rows which contain all indices, in other words those
168          valid as swaps
169          compRows ← apply(candComb, 1, function(row) all(1:length(CorrectLevs)
170              %in% row))
171          goodComb ← candComb[compRows,]
172          ## clean them up and print them
173          colnames(goodComb) ← NULL
174          rownames(goodComb) ← NULL
175          cat("Potential combinations:\n")
176          print(t(apply(goodComb, 1, order)))
177          ## take user input or automatically determine value
178          if (auto) {
179              if (!any(compRows)) acceptedComb ← 0 else acceptedComb ← 1
180          }
181      }
182  }

```

```

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

```

```

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

```



```

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

```

```

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

```

```

418         })
419         newrow$Disposition ← NULL
420         newrow ← c(unlist(newrow), ForeRace = row$Race[foreman],
421                   ForeGender = row$Gender[foreman], ForePol =
422                     row$PolAff[foreman])
423         if (sum(foreman) > 1) {
424             names(newrow)[names(newrow) == "ForeRace1"] ← "
425               ForeRace"
426             names(newrow)[names(newrow) == "ForeGender1"] ← "
427               ForeGender"
428             names(newrow)[names(newrow) == "ForePol1"] ← "
429               ForePol"
430         }
431         newrow
432     })
433     ## perform some clean up
434     longest ← sapply(Summary, length)
435     longest ← which(longest == max(longest))[1]
436     longNames ← names(Summary[[longest]])
437     Summary ← lapply(names(Summary[[longest]]),
438                     function(name) unname(sapply(Summary,
439                                                   function(el) el[name])))
440     names(Summary) ← longNames
441     Summary ← lapply(longNames,
442                     function(nm) {
443                         if (grepl("ForeGender", nm)) {
444                             Summary[[nm]] ← factor(Summary[[nm]], levels = 1:3,
445                                                     labels = LevGen)
446                         } else if (grepl("ForePol", nm)) {
447                             Summary[[nm]] ← factor(Summary[[nm]], levels = 1:5,
448                                                     labels = LevPol)
449                         } else if (grepl("ForeRace", nm)) {
450                             Summary[[nm]] ← factor(Summary[[nm]], levels = 1:7,
451                                                     labels = LevRace)
452                         } else Summary[[nm]]
453                     })
454     names(Summary) ← longNames
455     ## return these
456     list(Juries = Juries, Summaries = as.data.frame(Summary))
457 }
458
459 ## a generic simplification method to summarize a vector
460 Simplifier ← function(col, ...) {
461     UseMethod("Simplifier")
462 }
463
464 ## code up methods for the types to be seen
465 Simplifier.default ← function(col, collapse = "") paste0(col, collapse = collapse
466 )
467 Simplifier.numeric ← function(col, na.rm = TRUE, trim = 0, ...) mean.default(col,
468   trim = trim, na.rm = na.rm)
469 Simplifier.factor ← function(col, collapse = "", ...) paste0(sort(as.character(
470   levels(col)[as.numeric(col)])),
471   collapse = collapse
472 )
473 Simplifier.character ← function(col, collapse = "", ...) paste0(sort(col),
474   collapse = collapse)
475
476 ## create a grouping wrapper which does unique aggregation of a data set
477 UniqueAgg ← function(data, by, ...) {
478     ## convert data to a data frame for regularity
479     if (!is.data.frame(data)) data ← as.data.frame(data)
480     ## identify the grouping column by in the data
481     by.groups ← names(data) == by
482     ## provide nice error handling
483     stopifnot(sum(by.groups) > 0)
484     ## first identify which rows are already unique
485     groups ← as.numeric(as.factor(unlist(data[by.groups])))
486     unqRows ← sapply(groups, function(el) sum(groups == el) == 1)
487     ## consider grouping only the other rows using the unique function

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```



```

jursum$Summaries),
764         sun.trial, all = TRUE)
765
766 ## notice that the total removed variables are incomplete, try to correct this
767 where possible using the jury
768 ## summarized data above
769 sun.trialsum$DefRemEst ← RemovedJurorEstimates(sun.trialsum$DefenseTotalRemoved,
        data = sun.trialsum,
        ident = "Gender.DefRem", plot =
        FALSE)
770 ## perform this same procedure for the prosecution removals
771 sun.trialsum$ProRemEst ← RemovedJurorEstimates(sun.trialsum$StateTotalRemoved,
        data = sun.trialsum,
        ident = "Gender.ProRem", plot =
        FALSE)
772
773 ## synthesize some other variables, simple race indicators
774 sun.trialsum$DefWhiteBlack ← as.factor(FactorReduce(sun.trialsum$DefRace, tokeep
        = c("Black", "White", "U")))
775 sun.trialsum$DefWhiteOther ← as.factor(FactorReduce(sun.trialsum$DefWhiteBlack,
        tokeep = c("White", "U")))
776 ## the Kullback-Leibler divergence
777 sun.trialsum$KLdiv ← kldiv(sun.trialsum[,grepl("Jury", names(sun.trialsum))],
778         sun.trialsum[,grepl("Venire", names(sun.trialsum))])
779
780 ## Checkpoint 3: the data has been set to the trial level and summarized
781 ## save this
782 saveRDS(sun.trialsum, "TrialAggregated.Rds")
783 saveRDS(sun.jursum, "AllJuries.Rds")

```

A.3 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.