Introduction to Probability and Statistics Using R

G. Jay Kerns

Second Edition

IPSUR: Introduction to Probability and Statistics Using R
 ISBN: 978-0-557-24979-4

Date: September 12, 2011

Contents

Preface

This book was expanded from lecture materials I use in a one semester upper-division undergraduate course entitled *Probability and Statistics* at Youngstown State University. Those lecture materials, in turn, were based on notes that I transcribed as a graduate student at Bowling Green State University. The course for which the materials were written is 50-50 Probability and Statistics, and the attendees include mathematics, engineering, and computer science majors (among others). The catalog prerequisites for the course are a full year of calculus.

The book can be subdivided into three basic parts. The first part includes the introductions and elementary *descriptive statistics*; I want the students to be knee-deep in data right out of the gate. The second part is the study of *probability*, which begins at the basics of sets and the equally likely model, journeys past discrete/continuous random variables, and continues through to multivariate distributions. The chapter on sampling distributions paves the way to the third part, which is *inferential statistics*. This last part includes point and interval estimation, hypothesis testing, and finishes with introductions to selected topics in applied statistics.

I usually only have time in one semester to cover a small subset of this book. I cover the material in Chapter 2 in a class period that is supplemented by a take-home assignment for the students. I spend a lot of time on Data Description, Probability, Discrete, and Continuous Distributions. I mention selected facts from Multivariate Distributions in passing, and discuss the meaty parts of Sampling Distributions before moving right along to Estimation (which is another chapter I dwell on considerably). Hypothesis Testing goes faster after all of the previous work, and by that time the end of the semester is in sight. I normally choose one or two final chapters (sometimes three) from the remaining to survey, and regret at the end that I did not have the chance to cover more.

In an attempt to be correct I have included material in this book which I would normally not mention during the course of a standard lecture. For instance, I normally do not highlight the intricacies of measure theory or integrability conditions when speaking to the class. Moreover, I often stray from the matrix approach to multiple linear regression because many of my students have not yet been formally trained in linear algebra. That being said, it is important to me for the students to hold something in their hands which acknowledges the world of mathematics and statistics beyond the classroom, and which may be useful to them for many semesters to come. It also mirrors my own experience as a student.

The vision for this document is a more or less self contained, essentially complete,

correct, introductory textbook. There should be plenty of exercises for the student, with full solutions for some, and no solutions for others (so that the instructor may assign them for grading). By `Sweave`'s dynamic nature it is possible to write randomly generated exercises and I had planned to implement this idea already throughout the book. Alas, there are only 24 hours in a day. Look for more in future editions.

Seasoned readers will be able to detect my origins: *Probability and Statistical Inference* by Hogg and Tanis [**?**], *Statistical Inference* by Casella and Berger [**?**], and *Theory of Point Estimation* and *Testing Statistical Hypotheses* by Lehmann [**?**, **?**]. I highly recommend each of those books to every reader of this one. Some R books with "introductory" in the title that I recommend are *Introductory Statistics with R* by Dalgaard [**?**] and *Using R for Introductory Statistics* by Verzani [**?**]. Surely there are many, many other good introductory books about R, but frankly, I have tried to steer clear of them for the past year or so to avoid any undue influence on my own writing.

I would like to make special mention of two other books: *Introduction to Statistical Thought* by Michael Lavine [**?**] and *Introduction to Probability* by Grinstead and Snell [**?**]. Both of these books are *free* and are what ultimately convinced me to release IPSUR under a free license, too.

Please bear in mind that the title of this book is "Introduction to Probability and Statistics Using R", and not "Introduction to R Using Probability and Statistics", nor even "Introduction to Probability and Statistics and R Using Words". The people at the party are Probability and Statistics; the handshake is R. There are several important topics about R which some individuals will feel are underdeveloped, glossed over, or wantonly omitted. Some will feel the same way about the probabilistic and/or statistical content. Still others will just want to learn R and skip all of the mathematics.

Despite any misgivings: here it is, warts and all. I humbly invite said individuals to take this book, with the GNU Free Documentation License (GNU-FDL) in hand, and make it better. In that spirit there are at least a few ways in my view in which this book could be improved.

Better data. The data analyzed in this book are almost entirely from the `datasets` package in base R, and here is why:

- I made a conscious effort to minimize dependence on contributed packages,
- The data are instantly available, already in the correct format, so we need not take time to manage them, and
- The data are *real*.

I made no attempt to choose data sets that would be interesting to the students; rather, data were chosen for their potential to convey a statistical point. Many of the data sets are decades old or more (for instance, the data used to introduce simple linear regression are the speeds and stopping distances of cars in the 1920's).

In a perfect world with infinite time I would research and contribute recent, *real* data in a context crafted to engage the students in *every* example. One day I hope to stumble over said time. In the meantime, I will add new data sets incrementally as time permits.

More proofs. I would like to include more proofs for the sake of completeness (I understand that some people would not consider more proofs to be improvement). Many proofs have been skipped entirely, and I am not aware of any rhyme or reason to the current omissions. I will add more when I get a chance.

More and better˜graphics. I have not used the `ggplot2` package [**?**] because I do not know how to use it yet. It is on my to-do list.

More and better exercises. There are only a few exercises in the first edition simply because I have not had time to write more. I have toyed with the `exams` package [**?**] and I believe that it is a right way to move forward. As I learn more about what the package can do I would like to incorporate it into later editions of this book.

About This Document

IPSURcontains many interrelated parts: the *Document*, the *Program*, the *Package*, and the *Ancillaries*. In short, the *Document* is what you are reading right now. The *Program* provides an efficient means to modify the Document. The *Package* is an R package that houses the Program and the Document. Finally, the *Ancillaries* are extra materials that reside in the Package and were produced by the Program to supplement use of the Document. We briefly describe each of them in turn.

The Document

The *Document* is that which you are reading right now – IPSUR's *raison d'être*. There are transparent copies (nonproprietary text files) and opaque copies (everything else). See the GNU-FDL in Appendix **??** for more precise language and details.

IPSUR.tex is a transparent copy of the Document to be typeset with a LaTeX distribution such as MikTeX or TeX Live. Any reader is free to modify the Document and release the modified version in accordance with the provisions of the GNU-FDL. Note that this file cannot be used to generate a randomized copy of the Document. Indeed, in its released form it is only capable of typesetting the exact version of IPSURwhich you are currently reading. Furthermore, the `.tex` file is unable to generate any of the ancillary materials.

IPSUR-xxx.eps, IPSUR-xxx.pdf are the image files for every graph in the Document. These are needed when typesetting with LaTeX.

IPSUR.pdf is an opaque copy of the Document. This is the file that instructors would likely want to distribute to students.

IPSUR.dvi is another opaque copy of the Document in a different file format.

The Program

The *Program* includes `IPSUR.lyx` and its nephew `IPSUR.Rnw`; the purpose of each is to give individuals a way to quickly customize the Document for their particular purpose(s).

IPSUR.lyx is the source LyX file for the Program, released under the GNU General Public License (GNU GPL) Version 3. This file is opened, modified, and compiled with LyX, a sophisticated open-source document processor, and may be used (together with `Sweave`) to generate a randomized, modified copy of the Document with brand new data sets for some of the exercises and the solution manuals (in the Second Edition). Additionally, LyX can easily activate/deactivate entire blocks of the document, /e.g./˜the proofs of the theorems, the student solutions to the exercises, or the instructor answers to the problems, so that the new author may choose which sections (s)he would like to include in the final Document (again, Second Edition). The `IPSUR.lyx` file is all that a person needs (in addition to a properly configured system – see Appendix **??**) to generate/compile/export to all of the other formats described above and below, which includes the ancillary materials `IPSUR.Rdata` and `IPSUR.R`.

IPSUR.Rnw is another form of the source code for the Program, also released under the GNU GPL Version 3. It was produced by exporting `IPSUR.lyx` into R/Sweave format (`.Rnw`). This file may be processed with Sweave to generate a randomized copy of `IPSUR.tex` – a transparent copy of the Document – together with the ancillary materials `IPSUR.Rdata` and `IPSUR.R`. Please note, however, that `IPSUR.Rnw` is just a simple text file which does not support many of the extra features that LyX offers such as WYSIWYM editing, instantly (de)activating branches of the manuscript, and more.

The Package

There is a contributed package on `CRAN`, called `IPSUR`. The package affords many advantages, one being that it houses the Document in an easy-to-access medium. Indeed, a student can have the Document at his/her fingertips with only three commands:

Another advantage goes hand in hand with the Program's license; since IPSUR is free, the source code must be freely available to anyone that wants it. A package hosted on `CRAN` allows the author to obey the license by default.

A much more important advantage is that the excellent facilities at R-Forge are building and checking the package daily against patched and development versions of the absolute latest pre-release of R. If any problems surface then I will know about it within 24 hours.

And finally, suppose there is some sort of problem. The package structure makes it *incredibly* easy for me to distribute bug-fixes and corrected typographical errors. As an author I can make my corrections, upload them to the repository at R-Forge, and they will be reflected *worldwide* within hours. We aren't in Kansas anymore, Toto.

Ancillary Materials

These are extra materials that accompany IPSUR. They reside in the `/etc` subdirectory of the package source.

IPSUR.RData is a saved image of the R workspace at the completion of the Sweave processing of IPSUR. It can be loaded into memory with File ▸ Load Workspace or with the command `load("/path/to/IPSUR.Rdata")`. Either method will make every single object in the file immediately available and in memory. In particular, the data BLANK from Exercise BLANK in Chapter BLANK on page BLANK will be loaded. Type BLANK at the command line (after loading `IPSUR.RData`) to see for yourself.

IPSUR.R is the exported R code from `IPSUR.Rnw`. With this script, literally every R command from the entirety of IPSURcan be resubmitted at the command line.

Notation

We use the notation `x` or `stem.leaf` notation to denote objects, functions, *etc*.. The sequence "Statistics ▸ Summaries ▸ Active Dataset" means to click the Statistics menu item, next click the Summaries submenu item, and finally click Active Dataset.

Acknowledgements

This book would not have been possible without the firm mathematical and statistical foundation provided by the professors at Bowling Green State University, including Drs. G{a}bor Sz{e}kely, Craig Zirbel, Arjun K. Gupta, Hanfeng Chen, Truc Nguyen, and James Albert. I would also like to thank Drs.˜Neal Carothers and Kit Chan.

I would also like to thank my colleagues at Youngstown State University for their support. In particular, I would like to thank Dr. G. Andy Chang for showing me what it means to be a statistician.

I would like to thank Richard Heiberger for his insightful comments and improvements to several points and displays in the manuscript.

Finally, and most importantly, I would like to thank my wife for her patience and understanding while I worked hours, days, months, and years on a *free book*. Looking back, I can't believe I ever got away with it.

List of Figures

List of Tables

1 An Introduction to Probability and Statistics

This chapter has proved to be the hardest to write, by far. The trouble is that there is so much to say – and so many people have already said it so much better than I could. When I get something I like I will release it here.

In the meantime, there is a lot of information already available to a person with an Internet connection. I recommend to start at Wikipedia, which is not a flawless resource but it has the main ideas with links to reputable sources.

In my lectures I usually tell stories about Fisher, Galton, Gauss, Laplace, Quetelet, and the Chevalier de Mere.

1.1 Probability

The common folklore is that probability has been around for millennia but did not gain the attention of mathematicians until approximately 1654 when the Chevalier de Mere had a question regarding the fair division of a game's payoff to the two players, if the game had to end prematurely.

1.2 Statistics

Statistics concerns data; their collection, analysis, and interpretation. In this book we distinguish between two types of statistics: descriptive and inferential.

Descriptive statistics concerns the summarization of data. We have a data set and we would like to describe the data set in multiple ways. Usually this entails calculating numbers from the data, called descriptive measures, such as percentages, sums, averages, and so forth.

Inferential statistics does more. There is an inference associated with the data set, a conclusion drawn about the population from which the data originated.

I would like to mention that there are two schools of thought of statistics: frequentist and bayesian. The difference between the schools is related to how the two groups interpret the underlying probability (see Section **??**). The frequentist school gained a lot of ground among statisticians due in large part to the work of Fisher, Neyman, and Pearson in the early twentieth century. That dominance lasted until inexpensive computing power

became widely available; nowadays the bayesian school is garnering more attention and at an increasing rate.

This book is devoted mostly to the frequentist viewpoint because that is how I was trained, with the conspicuous exception of Sections **??** and **??**. I plan to add more bayesian material in later editions of this book.

1.3 Exercises

2 Data Description

In this chapter we introduce the different types of data that a statistician is likely to encounter, and in each subsection we give some examples of how to display the data of that particular type. Once we see how to display data distributions, we next introduce the basic properties of data distributions. We qualitatively explore several data sets. Once that we have intuitive properties of data sets, we next discuss how we may numerically measure and describe those properties with descriptive statistics.

What do I want them to know?

- different data types, such as quantitative versus qualitative, nominal versus ordinal, and discrete versus continuous
- basic graphical displays for assorted data types, and some of their (dis)advantages
- fundamental properties of data distributions, including center, spread, shape, and crazy observations
- methods to describe data (visually/numerically) with respect to the properties, and how the methods differ depending on the data type
- all of the above in the context of grouped data, and in particular, the concept of a factor

2.1 Types of Data

Loosely speaking, a datum is any piece of collected information, and a data set is a collection of data related to each other in some way. We will categorize data into five types and describe each in turn:

Quantitative data associated with a measurement of some quantity on an observational unit,

Qualitative data associated with some quality or property of an observational unit,

Logical data which represent true or false and play an important role later,

Missing data which should be there but are not, and

Other types everything else under the sun.

In each subsection we look at some examples of the type in question and introduce methods to display them.

2.1.1 Quantitative data

Quantitative data are any data that measure or are associated with a measurement of the quantity of something. They invariably assume numerical values. Quantitative data can be further subdivided into two categories.

- *Discrete data* take values in a finite or countably infinite set of numbers, that is, all possible values could (at least in principle) be written down in an ordered list. Examples include: counts, number of arrivals, or number of successes. They are often represented by integers, say, 0, 1, 2, *etc*.
- *Continuous data* take values in an interval of numbers. These are also known as scale data, interval data, or measurement data. Examples include: height, weight, length, time, *etc*. Continuous data are often characterized by fractions or decimals: 3.82, 7.0001, $4\ \frac{5}{8}$, *etc*.

Note that the distinction between discrete and continuous data is not always clear-cut. Sometimes it is convenient to treat data as if they were continuous, even though strictly speaking they are not continuous. See the examples.

Example 2.1. Annual Precipitation in US Cities. The vector `precip` contains average amount of rainfall (in inches) for each of 70 cities in the United States and Puerto Rico. Let us take a look at the data:

```
str(precip)
```

```
  Named num [1:70] 67 54.7 7 48.5 14 17.2 20.7 13 43.4 40.2 ...
  - attr(*, "names")= chr [1:70] "Mobile" "Juneau" "Phoenix" "Little Rock" ...
```

```
precip[1:4]
```

```
     Mobile      Juneau     Phoenix Little Rock
       67.0        54.7         7.0        48.5
```

The output shows that `precip` is a numeric vector which has been *named*, that is, each value has a name associated with it (which can be set with the `names` function). These are quantitative continuous data.

Example 2.2. Lengths of Major North American Rivers. The U.S. Geological Survey recorded the lengths (in miles) of several rivers in North America. They are stored in the vector `rivers` in the `datasets` package (which ships with base R). See `?rivers`. Let us take a look at the data with the `str` function.

```
str(rivers)
```

```
  num [1:141] 735 320 325 392 524 ...
```

The output says that `rivers` is a numeric vector of length 141, and the first few values are 735, 320, 325, *etc*. These data are definitely quantitative and it appears that the measurements have been rounded to the nearest mile. Thus, strictly speaking, these are discrete data. But we will find it convenient later to take data like these to be continuous for some of our statistical procedures.

Example 2.3. Yearly Numbers of Important Discoveries. The vector `discoveries` contains numbers of "great" inventions/discoveries in each year from 1860 to 1959, as reported by the 1975 World Almanac. Let us take a look at the data:

```
str(discoveries)
```

```
  Time-Series [1:100] from 1860 to 1959: 5 3 0 2 0 3 2 3 6 1 ...
```

The output is telling us that `discoveries` is a *time series* (see Section 2.1.7 for more) of length 100. The entries are integers, and since they represent counts this is a good example of discrete quantitative data. We will take a closer look in the following sections.

2.1.2 Displaying Quantitative Data

One of the first things to do when confronted by quantitative data (or any data, for that matter) is to make some sort of visual display to gain some insight into the data's structure. There are almost as many display types from which to choose as there are data sets to plot. We describe some of the more popular alternatives.

Strip charts (also known as Dot plots) These can be used for discrete or continuous data, and usually look best when the data set is not too large. Along the horizontal axis is a numerical scale above which the data values are plotted. We can do it in R with a call to the `stripchart` function. There are three available methods.

overplot plots ties covering each other. This method is good to display only the distinct values assumed by the data set.

jitter adds some noise to the data in the y direction in which case the data values are not covered up by ties.

stack plots repeated values stacked on top of one another. This method is best used for discrete data with a lot of ties; if there are no repeats then this method is identical to overplot.

See Figure 2.1.1, which was produced by the following code.

```
stripchart(precip, xlab="rainfall")
stripchart(rivers, method="jitter", xlab="length")
stripchart(discoveries, method="stack", xlab="number")
```

The leftmost graph is a strip chart of the `precip` data. The graph shows tightly clustered values in the middle with some others falling balanced on either side, with perhaps slightly more falling to the left. Later we will call this a symmetric distribution, see Section 2.2.3. The middle graph is of the `rivers` data, a vector of length 141. There are several repeated values in the rivers data, and if we were to use the overplot method we would lose some of them in the display. This plot shows a what we will later call a right-skewed shape with perhaps some extreme values on the far right of the display. The third graph strip charts `discoveries` data which are literally a textbook example of a right skewed distribution.

Figure 2.1.1: Three stripcharts of three data sets. The first graph uses the `overplot` method, the second the `jitter` method, and the third the `stack` method.

The `DOTplot` function in the `UsingR` package [**?**] is another alternative.

Histogram These are typically used for continuous data. A histogram is constructed by first deciding on a set of classes, or bins, which partition the real line into a set of boxes into which the data values fall. Then vertical bars are drawn over the bins with height proportional to the number of observations that fell into the bin.

These are one of the most common summary displays, and they are often misidentified as "Bar Graphs" (see below.) The scale on the y axis can be frequency, percentage, or density (relative frequency). The term histogram was coined by Karl Pearson in 1891, see [**?**].

Example 2.4. Annual Precipitation in US Cities. We are going to take another look at the `precip` data that we investigated earlier. The strip chart in Figure **??** suggested a loosely balanced distribution; let us now look to see what a histogram says.

There are many ways to plot histograms in R, and one of the easiest is with the `hist` function. The following code produces the plots in Figure 2.1.2.

```
hist(precip, main = "")
hist(precip, freq = FALSE, main = "")
```

Notice the argument `main = ""` which suppresses the main title from being displayed – it would have said "Histogram of `precip`" otherwise. The plot on the left is a frequency histogram (the default), and the plot on the right is a relative frequency histogram (`freq = FALSE`).

```
m <- ggplot(as.data.frame(precip), aes(x = precip))
m + geom_histogram()
m + geom_histogram(binwidth = 5)

par(mfrow = c(1,2)) # 2 plots: 1 row, 2 columns
hist(precip, main = "")
hist(precip, freq = FALSE, main = "")
par(mfrow = c(1,1)) # back to normal
```

Figure 2.1.2: (Relative) frequency histograms of the `precip` data

Please mind the biggest weakness of histograms: the graph obtained strongly depends on the bins chosen. Choose another set of bins, and you will get a different histogram. Moreover, there are not any definitive criteria by which bins should be defined; the best choice for a given data set is the one which illuminates the data set's underlying structure (if any). Luckily for us there are algorithms to automatically choose bins that are likely to display well, and more often than not the default bins do a good job. This is not always the case, however, and a responsible statistician will investigate many bin choices to test the stability of the display.

Example 2.5. Recall that the strip chart in Figure **??** suggested a relatively balanced shape to the `precip` data distribution. Watch what happens when we change the bins slightly (with the `breaks` argument to `hist`). See Figure 2.1.3 which was produced by the following code.

```
hist(precip, breaks = 10, main = "")
hist(precip, breaks = 200, main = "")
```

Figure 2.1.3: More histograms of the `precip` data

The leftmost graph (with `breaks = 10`) shows that the distribution is not balanced at all. There are two humps: a big one in the middle and a smaller one to the left. Graphs like this often indicate some underlying group structure to the data; we could now investigate whether the cities for which rainfall was measured were similar in some way, with respect to geographic region, for example.

The rightmost graph in Figure 2.1.3 shows what happens when the number of bins is too large: the histogram is too grainy and hides the rounded appearance of the earlier histograms. If we were to continue increasing the number of bins we would eventually get all observed bins to have exactly one element, which is nothing more than a glorified strip chart.

Stemplots (more to be said in Section 2.4) Stemplots have two basic parts: *stems* and *leaves*. The final digit of the data values is taken to be a *leaf*, and the leading digit(s) is (are) taken to be *stems*. We draw a vertical line, and to the left of the line we list the stems. To the right of the line, we list the leaves beside their corresponding stem. There will typically be several leaves for each stem, in which case the leaves accumulate to the right. It is sometimes necessary to round the data values, especially for larger data sets.

Example 2.6. `UKDriverDeaths` is a time series that contains the total car drivers killed or seriously injured in Great Britain monthly from Jan 1969 to Dec 1984. See `?UKDriverDeaths`. Compulsory seat belt use was introduced on January 31, 1983. We construct a stem and leaf diagram in R with the `stem.leaf` function from the `aplpack` package[**?**].

```
library(aplpack)
stem.leaf(UKDriverDeaths, depth = FALSE)
```

```
1 | 2: represents 120
 leaf unit: 10
            n: 192
   10 | 57
   11 | 136678
   12 | 123889
   13 | 0255666888899
   14 | 000012223444445555566677888 89
   15 | 000011111222222344445555566677779
   16 | 012223334444455555556788888 89
   17 | 11233344566667799
   18 | 00011235568
   19 | 01234455667799
   20 | 0000113557788899
   21 | 145599
```

```
   22 | 013467
   23 | 9
   24 | 7
HI: 2654
```

The display shows a more or less balanced mound-shaped distribution, with one or maybe two humps, a big one and a smaller one just to its right. Note that the data have been rounded to the tens place so that each datum gets only one leaf to the right of the dividing line.

Notice that the `depth`s have been suppressed. To learn more about this option and many others, see Section 2.4. Unlike a histogram, the original data values may be recovered from the stemplot display – modulo the rounding – that is, starting from the top and working down we can read off the data values 1050, 1070, 1110, 1130, and so forth.

Index plots Done with the `plot` function. These are good for plotting data which are ordered, for example, when the data are measured over time. That is, the first observation was measured at time 1, the second at time 2, *etc*. It is a two dimensional plot, in which the index (or time) is the x variable and the measured value is the y variable. There are several plotting methods for index plots, and we mention two of them:

spikes draws a vertical line from the x-axis to the observation height (`type = "h"`).

points plots a simple point at the observation height (=type = "p"=).

Example 2.7. Level of Lake Huron 1875-1972. Brockwell and Davis [**?**] give the annual measurements of the level (in feet) of Lake Huron from 1875–1972. The data are stored in the time series `LakeHuron`. See `?LakeHuron`. Figure 2.1.4 was produced with the following code:

```
plot(LakeHuron, type = "h")
plot(LakeHuron, type = "p")
```

The plots show an overall decreasing trend to the observations, and there appears to be some seasonal variation that increases over time.

Figure 2.1.4: Index plots of the `LakeHuron` data

Density estimates Coming soon.

2.1.3 Qualitative Data, Categorical Data, and Factors

Qualitative data are simply any type of data that are not numerical, or do not represent numerical quantities. Examples of qualitative variables include a subject's name, gender, race/ethnicity, political party, socioeconomic status, class rank, driver's license number, and social security number (SSN).

Please bear in mind that some data *look* to be quantitative but are *not*, because they do not represent numerical quantities and do not obey mathematical rules. For example, a person's shoe size is typically written with numbers: 8, or 9, or 12, or $12\frac{1}{2}$. Shoe size is not quantitative, however, because if we take a size 8 and combine with a size 9 we do not get a size 17.

Some qualitative data serve merely to *identify* the observation (such a subject's name, driver's license number, or SSN). This type of data does not usually play much of a role in statistics. But other qualitative variables serve to *subdivide* the data set into categories; we call these *factors*. In the above examples, gender, race, political party, and socioeconomic status would be considered factors (shoe size would be another one). The possible values of a factor are called its *levels*. For instance, the factor *gender* would have two levels, namely, male and female. Socioeconomic status typically has three levels: high, middle, and low.

Factors may be of two types: *nominal* and *ordinal*. Nominal factors have levels that correspond to names of the categories, with no implied ordering. Examples of nominal factors would be hair color, gender, race, or political party. There is no natural ordering to "Democrat" and "Republican"; the categories are just names associated with different groups of people.

In contrast, ordinal factors have some sort of ordered structure to the underlying factor levels. For instance, socioeconomic status would be an ordinal categorical variable because the levels correspond to ranks associated with income, education, and occupation. Another example of ordinal categorical data would be class rank.

Factors have special status in R. They are represented internally by numbers, but even when they are written numerically their values do not convey any numeric meaning or obey any mathematical rules (that is, Stage III cancer is not Stage I cancer + Stage II cancer).

Example 2.8. The `state.abb` vector gives the two letter postal abbreviations for all 50 states.

```
str(state.abb)
```

```
  chr [1:50] "AL" "AK" "AZ" "AR" "CA" "CO" ...
```

These would be ID data. The `state.name` vector lists all of the complete names and those data would also be ID.

Example 2.9. U.S. State Facts and Features. The U.S. Department of Commerce of the U.S. Census Bureau releases all sorts of information in the *Statistical Abstract of the United States*, and the `state.region` data lists each of the 50 states and the region to which it belongs, be it Northeast, South, North Central, or West. See `?state.region`.

```
str(state.region)
state.region[1:5]
```

```
 Factor w/ 4 levels "Northeast","South",..: 2 4 4 2 4 4 1 2 2 2 ...
[1] South West  West  South West
Levels: Northeast South North Central West
```

The `str` output shows that `state.region` is already stored internally as a factor and it lists a couple of the factor levels. To see all of the levels we printed the first five entries of the vector in the second line.

2.1.4 Displaying Qualitative Data

Tables One of the best ways to summarize qualitative data is with a table of the data values. We may count frequencies with the `table` function or list proportions with the `prop.table` function (whose input is a frequency table). In the R Commander you can do it with Statistics ▸ Frequency Distribution... Alternatively, to look at tables for all factors in the `Active data set` you can do Statistics ▸ Summaries ▸ Active Dataset.

```
Tbl <- table(state.division)
```

```
Tbl
```

```
state.division
       New England    Middle Atlantic     South Atlantic
                 6                  3                  8
East South Central West South Central East North Central
                 4                  4                  5
West North Central           Mountain            Pacific
                 7                  8                  5
```

```
Tbl/sum(Tbl)      # relative frequencies
```

```
state.division
       New England    Middle Atlantic     South Atlantic
              0.12               0.06               0.16
East South Central West South Central East North Central
              0.08               0.08               0.10
West North Central           Mountain            Pacific
              0.14               0.16               0.10
```

```
prop.table(Tbl)   # same thing
```

```
 state.division
       New England    Middle Atlantic     South Atlantic
               0.12               0.06               0.16
 East South Central West South Central East North Central
               0.08               0.08               0.10
 West North Central           Mountain            Pacific
               0.14               0.16               0.10
```

Bar Graphs A bar graph is the analogue of a histogram for categorical data. A bar is displayed for each level of a factor, with the heights of the bars proportional to the frequencies of observations falling in the respective categories. A disadvantage of bar graphs is that the levels are ordered alphabetically (by default), which may sometimes obscure patterns in the display.

Example 2.10. U.S. State Facts and Features. The `state.region` data lists each of the 50 states and the region to which it belongs, be it Northeast, South, North Central, or West. See `?state.region`. It is already stored internally as a factor. We make a bar graph with the `barplot` function:

```
barplot(table(state.region), cex.names = 0.50)
barplot(prop.table(table(state.region)), cex.names = 0.50)
```

See Figure 2.1.5. The display on the left is a frequency bar graph because the y axis shows counts, while the display on the left is a relative frequency bar graph. The only difference between the two is the scale. Looking at the graph we see that the majority of the fifty states are in the South, followed by West, North Central, and finally Northeast. Over 30% of the states are in the South.

Notice the `cex.names` argument that we used, above. It shrinks the names on the x axis by 50% which makes them easier to read. See `?par` for a detailed list of additional plot parameters.

Figure 2.1.5: The left graph is a frequency barplot made with `table` and the right is a relative frequency barplot made with `prop.table`.

Pareto Diagrams A pareto diagram is a lot like a bar graph except the bars are rearranged such that they decrease in height going from left to right. The rearrangement is handy because it can visually reveal structure (if any) in how fast the bars decrease – this is much more difficult when the bars are jumbled.

Example 2.11. U.S. State Facts and Features. The `state.division` data record the division (New England, Middle Atlantic, South Atlantic, East South Central, West South Central, East North Central, West North Central, Mountain, and Pacific) of the fifty states. We can make a pareto diagram with either the `RcmdrPlugin.IPSUR` package or with the `pareto.chart` function from the `qcc` package [**?**]. See Figure 2.1.6. The code follows.

```
library(qcc)
pareto.chart(table(state.division), ylab="Frequency")
```

Figure 2.1.6: Pareto chart of the `state.division` data

Dot Charts These are a lot like a bar graph that has been turned on its side with the bars replaced by dots on horizontal lines. They do not convey any more (or less) information than the associated bar graph, but the strength lies in the economy of the display. Dot charts are so compact that it is easy to graph very complicated multi-variable interactions together in one graph. See Section 2.6. We will give an example here using the same data as above for comparison. The graph was produced by the following code.

Example 2.12.
```
x <- table(state.region)
dotchart(as.vector(x), labels = names(x))
```

Figure 2.1.7: Dot chart of the `state.region` data

See Figure 2.1.7. Compare it to Figure 2.1.5.

Pie Graphs These can be done with R and the R Commander, but they fallen out of favor in recent years because researchers have determined that while the human eye is good at judging linear measures, it is notoriously bad at judging relative areas (such as those displayed by a pie graph). Pie charts are consequently a very bad way of displaying information. A bar chart or dot chart is a preferable way of displaying qualitative data. See `?pie` for more information.

We are not going to do any examples of a pie graph and discourage their use elsewhere.

2.1.5 Logical Data

There is another type of information recognized by R which does not fall into the above categories. The value is either `TRUE` or `FALSE` (note that equivalently you can use `1 = TRUE`, `0 = FALSE`). Here is an example of a logical vector:

```
x <- 5:9
y <- (x < 7.3)
y
```

```
[1]  TRUE  TRUE  TRUE FALSE FALSE
```

Many functions in R have options that the user may or may not want to activate in the function call. For example, the `stem.leaf` function has the `depths` argument which is `TRUE` by default. We saw in Section 2.1.1 how to turn the option off, simply enter `stem.leaf(x, depths = FALSE)` and they will not be shown on the display.

We can swap `TRUE` with `FALSE` with the exclamation point `!`.

```
!y
```

```
[1] FALSE FALSE FALSE  TRUE  TRUE
```

2.1.6 Missing Data

Missing data are a persistent and prevalent problem in many statistical analyses, especially those associated with the social sciences. R reserves the special symbol `NA` to representing missing data.

Ordinary arithmetic with `NA` values give `NA`'s (addition, subtraction, *etc*.) and applying a function to a vector that has an `NA` in it will usually give an `NA`.

```
x <- c(3, 7, NA, 4, 7)
y <- c(5, NA, 1, 2, 2)
x + y
```

```
[1]  8 NA NA  6  9
```

Some functions have a `na.rm` argument which when `TRUE` will ignore missing data as if they were not there (such as `mean`, `var`, `sd`, `IQR`, `mad`, ...).

```
sum(x)
sum(x, na.rm = TRUE)
```

```
[1] NA
[1] 21
```

Other functions do not have a `na.rm` argument and will return `NA` or an error if the argument has `NA`s. In those cases we can find the locations of any `NA`s with the `is.na` function and remove those cases with the `[]` operator.

```
is.na(x)
z <- x[!is.na(x)]
sum(z)
```

```
 [1] FALSE FALSE  TRUE FALSE FALSE
 [1] 21
```

The analogue of `is.na` for rectangular data sets (or data frames) is the `complete.cases` function. See Appendix **??**.

2.1.7 Other Data Types

2.2 Features of Data Distributions

Given that the data have been appropriately displayed, the next step is to try to identify salient features represented in the graph. The acronym to remember is *C*enter, *U*nusual features, *S*pread, and *S*hape. (CUSS).

2.2.1 Center

One of the most basic features of a data set is its center. Loosely speaking, the center of a data set is associated with a number that represents a middle or general tendency of the data. Of course, there are usually several values that would serve as a center, and our later tasks will be focused on choosing an appropriate one for the data at hand. Judging from the histogram that we saw in Figure 2.1.3, a measure of center would be about 35.

2.2.2 Spread

The spread of a data set is associated with its variability; data sets with a large spread tend to cover a large interval of values, while data sets with small spread tend to cluster tightly around a central value.

2.2.3 Shape

When we speak of the *shape* of a data set, we are usually referring to the shape exhibited by an associated graphical display, such as a histogram. The shape can tell us a lot about any underlying structure to the data, and can help us decide which statistical procedure we should use to analyze them.

Symmetry and Skewness A distribution is said to be *right-skewed* (or *positively skewed*) if the right tail seems to be stretched from the center. A *left-skewed* (or *negatively skewed*) distribution is stretched to the left side. A symmetric distribution has a graph that is balanced about its center, in the sense that half of the graph may be reflected about a central line of symmetry to match the other half.

We have already encountered skewed distributions: both the discoveries data in Figure 2.1.1 and the `precip` data in Figure 2.1.3 appear right-skewed. The `UKDriverDeaths` data in Example 2.6 is relatively symmetric (but note the one extreme value 2654 identified at the bottom of the stemplot).

Kurtosis Another component to the shape of a distribution is how "peaked" it is. Some distributions tend to have a flat shape with thin tails. These are called *platykurtic*, and an example of a platykurtic distribution is the uniform distribution; see Section **??**. On the other end of the spectrum are distributions with a steep peak, or spike, accompanied by heavy tails; these are called *leptokurtic*. Examples of leptokurtic distributions are the Laplace distribution and the logistic distribution. See Section **??**. In between are distributions (called *mesokurtic*) with a rounded peak and moderately sized tails. The standard example of a mesokurtic distribution is the famous bell-shaped curve, also known as the Gaussian, or normal, distribution, and the binomial distribution can be mesokurtic for specific choices of p. See Sections **??** and **??**.

2.2.4 Clusters and Gaps

Clusters or gaps are sometimes observed in quantitative data distributions. They indicate clumping of the data about distinct values, and gaps may exist between clusters. Clusters often suggest an underlying grouping to the data. For example, take a look at the `faithful` data which contains the duration of `eruptions` and the `waiting` time between eruptions of the Old Faithful geyser in Yellowstone National Park. Do not be frightened by the complicated information at the left of the display for now; we will learn how to interpret it in Section 2.4.

```
library(aplpack)
with(faithful, stem.leaf(eruptions))
```

```
1 | 2: represents 1.2
 leaf unit: 0.1
            n: 272
   12     s | 667777777777
   51    1. | 888888888888888888888888888899999999999
   71    2* | 00000000000011111111
   87     t | 2222222222333333
```

```
 92     f | 44444
 94     s | 66
 97    2. | 889
 98    3* | 0
102     t | 3333
108     f | 445555
118     s | 6666677777
(16)   3. | 8888888889999999
138    4* | 00000000000000001111111111111111
107     t | 22222222222333333333333333333
 78     f | 44444444444445555555555555555555555
 43     s | 6666666666677777777777
 21    4. | 88888888888899999
  4    5* | 0001
```

There are definitely two clusters of data here; an upper cluster and a lower cluster.

2.2.5 Extreme Observations and other Unusual Features

Extreme observations fall far from the rest of the data. Such observations are troublesome to many statistical procedures; they cause exaggerated estimates and instability. It is important to identify extreme observations and examine the source of the data more closely. There are many possible reasons underlying an extreme observation:

- **Maybe the value is a typographical error.** Especially with large data sets becoming more prevalent, many of which being recorded by hand, mistakes are a common problem. After closer scrutiny, these can often be fixed.

- **Maybe the observation was not meant for the study**, because it does not belong to the population of interest. For example, in medical research some subjects may have relevant complications in their genealogical history that would rule out their participation in the experiment. Or when a manufacturing company investigates the properties of one of its devices, perhaps a particular product is malfunctioning and is not representative of the majority of the items.

- **Maybe it indicates a deeper trend or phenomenon.** Many of the most influential scientific discoveries were made when the investigator noticed an unexpected result, a value that was not predicted by the classical theory. Albert Einstein, Louis Pasteur, and others built their careers on exactly this circumstance.

2.3 Descriptive Statistics

2.3.1 Frequencies and Relative Frequencies

These are used for categorical data. The idea is that there are a number of different categories, and we would like to get some idea about how the categories are represented in the population.

2.3.2 Measures of Center

The *sample mean* is denoted $\overline{x}$ (read "x-bar") and is simply the arithmetic average of the observations:

$$\overline{x} = \frac{x_1 + x_2 + \cdots + x_n}{n} = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{2.3.1}$$

- Good: natural, easy to compute, has nice mathematical properties
- Bad: sensitive to extreme values

It is appropriate for use with data sets that are not highly skewed without extreme observations.

The *sample median* is another popular measure of center and is denoted $\tilde{x}$. To calculate its value, first sort the data into an increasing sequence of numbers. If the data set has an odd number of observations then $\tilde{x}$ is the value of the middle observation, which lies in position $(n + 1)/2$; otherwise, there are two middle observations and $\tilde{x}$ is the average of those middle values.

- Good: resistant to extreme values, easy to describe
- Bad: not as mathematically tractable, need to sort the data to calculate

One desirable property of the sample median is that it is *resistant* to extreme observations, in the sense that the value of $\tilde{x}$ depends only on those data values in the middle, and is quite unaffected by the actual values of the outer observations in the ordered list. The same cannot be said for the sample mean. Any significant changes in the magnitude of an observation x_k results in a corresponding change in the value of the mean. Consequently, the sample mean is said to be *sensitive* to extreme observations.

The *trimmed mean* is a measure designed to address the sensitivity of the sample mean to extreme observations. The idea is to "trim" a fraction (less than 1/2) of the observations off each end of the ordered list, and then calculate the sample mean of what remains. We will denote it by $\overline{x}_{t=0.05}$.

- Good: resistant to extreme values, shares nice statistical properties
- Bad: need to sort the data

How to do it with R

- You can calculate frequencies or relative frequencies with the `table` function, and relative frequencies with `prop.table(table())`.
- You can calculate the sample mean of a data vector `x` with the command `mean(x)`.
- You can calculate the sample median of `x` with the command `median(x)`.
- You can calculate the trimmed mean with the `trim` argument; `mean(x, trim = 0.05)`.

2.3.3 Order Statistics and the Sample Quantiles

A common first step in an analysis of a data set is to sort the values. Given a data set x_1, x_2, ..., x_n, we may sort the values to obtain an increasing sequence

$$x_{(1)} \leq x_{(2)} \leq x_{(3)} \leq \cdots \leq x_{(n)} \tag{2.3.2}$$

and the resulting values are called the *order statistics*. The k^{th} entry in the list, $x_{(k)}$, is the k^{th} order statistic, and approximately $100(k/n)\%$ of the observations fall below $x_{(k)}$. The order statistics give an indication of the shape of the data distribution, in the sense that a person can look at the order statistics and have an idea about where the data are concentrated, and where they are sparse.

The *sample quantiles* are related to the order statistics. Unfortunately, there is not a universally accepted definition of them. Indeed, R is equipped to calculate quantiles using nine distinct definitions! We will describe the default method (`type = 7`), but the interested reader can see the details for the other methods with `?quantile`.

Suppose the data set has n observations. Find the sample quantile of order p ($0 < p < 1$), denoted $\tilde{q}_p$, as follows:

First step: sort the data to obtain the order statistics $x_{(1)}$, $x_{(2)}$, ...,$x_{(n)}$.

Second step: calculate $(n-1)p + 1$ and write it in the form $k.d$, where k is an integer and d is a decimal.

Third step: The sample quantile $\tilde{q}_p$ is

$$\tilde{q}_p = x_{(k)} + d(x_{(k+1)} - x_{(k)}). \tag{2.3.3}$$

The interpretation of $\tilde{q}_p$ is that approximately $100p$ % of the data fall below the value $\tilde{q}_p$.

Keep in mind that there is not a unique definition of percentiles, quartiles, *etc*. Open a different book, and you'll find a different definition. The difference is small and seldom

plays a role except in small data sets with repeated values. In fact, most people do not even notice in common use.

Clearly, the most popular sample quantile is $\tilde{q}_{0.50}$, also known as the sample median, $\tilde{x}$. The closest runners-up are the *first quartile* $\tilde{q}_{0.25}$ and the *third quartile* $\tilde{q}_{0.75}$ (the *second quartile* is the median).

How to do it with R

At the command prompt we can find the order statistics of a data set stored in a vector `x` with the command `sort(x)`.

We can calculate the sample quantiles of any order p where $0 < p < 1$ for a data set stored in a data vector `x` with the `quantile` function, for instance, the command `quantile(x, probs = c(0, 0.25, 0.37))` will return the smallest observation, the first quartile, $\tilde{q}_{0.25}$, and the 37th sample quantile, $\tilde{q}_{0.37}$. For $\tilde{q}_p$ simply change the values in the `probs` argument to the value p.

With the R Commander we can find the order statistics of a variable in the `Active data set` by doing Data ▸ Manage variables in Active data set ... ▸ Compute new variable.... In the Expression to compute dialog simply type `sort(varname)`, where `varname` is the variable that it is desired to sort.

In `Rcmdr`, we can calculate the sample quantiles for a particular variable with the sequence Statistics ▸ Summaries ▸ Numerical Summaries... We can automatically calculate the quartiles for all variables in the `Active data set` with the sequence Statistics ▸ Summaries ▸ Active Dataset.

2.3.4 Measures of Spread

Sample Variance and Standard Deviation The *sample variance* is denoted s^2 and is calculated with the formula

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2. \tag{2.3.4}$$

The *sample standard deviation* is $s = \sqrt{s^2}$. Intuitively, the sample variance is approximately the average squared distance of the observations from the sample mean. The sample standard deviation is used to scale the estimate back to the measurement units of the original data.

- Good: tractable, has nice mathematical/statistical properties
- Bad: sensitive to extreme values

We will spend a lot of time with the variance and standard deviation in the coming chapters. In the meantime, the following two rules give some meaning to the standard deviation, in that there are bounds on how much of the data can fall past a certain distance from the mean.

Fact 2.13. *Chebychev's Rule: The proportion of observations within k standard deviations of the mean is at least* $1 - 1/k^2$*,* i.e.*, at least 75%, 89%, and 94% of the data are within 2, 3, and 4 standard deviations of the mean, respectively.*

Note that Chebychev's Rule does not say anything about when $k = 1$, because $1 - 1/1^2 = 0$, which states that at least 0% of the observations are within one standard deviation of the mean (which is not saying much).

Chebychev's Rule applies to any data distribution, *any* list of numbers, no matter where it came from or what the histogram looks like. The price for such generality is that the bounds are not very tight; if we know more about how the data are shaped then we can say more about how much of the data can fall a given distance from the mean.

Fact 2.14. *Empirical Rule: If data follow a bell-shaped curve, then approximately 68%, 95%, and 99.7% of the data are within 1, 2, and 3 standard deviations of the mean, respectively.*

Interquartile Range Just as the sample mean is sensitive to extreme values, so the associated measure of spread is similarly sensitive to extremes. Further, the problem is exacerbated by the fact that the extreme distances are squared. We know that the sample quartiles are resistant to extremes, and a measure of spread associated with them is the *interquartile range* (IQR) defined by $IQR = q_{0.75} - q_{0.25}$.

- Good: stable, resistant to outliers, robust to nonnormality, easy to explain
- Bad: not as tractable, need to sort the data, only involves the middle 50% of the data.

Median Absolute Deviation A measure even more robust than the IQR is the *median absolute deviation* (MAD). To calculate it we first get the median $\widetilde{x}$, next the *absolute deviations* $|x_1 - \tilde{x}|, |x_2 - \tilde{x}|, \ldots, |x_n - \tilde{x}|$, and the MAD is proportional to the median of those deviations:

$$MAD \propto \text{median}(|x_1 - \tilde{x}|,\ |x_2 - \tilde{x}|, \ldots, |x_n - \tilde{x}|). \tag{2.3.5}$$

That is, the $MAD = c \cdot \text{median}(|x_1 - \tilde{x}|,\ |x_2 - \tilde{x}|, \ldots, |x_n - \tilde{x}|)$, where c is a constant chosen so that the MAD has nice properties. The value of c in R is by default $c = 1.4286$. This value is chosen to ensure that the estimator of σ is correct, on the average, under suitable sampling assumptions (see Section **??**).

- Good: stable, very robust, even more so than the IQR.
- Bad: not tractable, not well known and less easy to explain.

Comparing Apples to Apples We have seen three different measures of spread which, for a given data set, will give three different answers. Which one should we use? It depends on the data set. If the data are well behaved, with an approximate bell-shaped distribution, then the sample mean and sample standard deviation are natural choices with nice mathematical properties. However, if the data have an unusual or skewed shape with several extreme values, perhaps the more resistant choices among the IQR or MAD would be more appropriate.

However, once we are looking at the three numbers it is important to understand that the estimators are not all measuring the same quantity, on the average. In particular, it can be shown that when the data follow an approximately bell-shaped distribution, then on the average, the sample standard deviation s and the MAD will be the approximately the same value, namely, σ, but the IQR will be on the average 1.349 times larger than s and the MAD. See **??** for more details.

How to do it with R

At the command prompt we may compute the sample range with `range(x)` and the sample variance with `var(x)`, where `x` is a numeric vector. The sample standard deviation is `sqrt(var(x))` or just `sd(x)`. The IQR is `IQR(x)` and the median absolute deviation is `mad(x)`.

With the R Commander we can calculate the sample standard deviation with the Statistics ▸ Summaries ▸ Numerical Summaries... combination. R Commander does not calculate the IQR or MAD in any of the menu selections, by default.

2.3.5 Measures of Shape

Sample Skewness The *sample skewness*, denoted by g_1, is defined by the formula

$$g_1 = \frac{1}{n}\frac{\sum_{i=1}^{n}(x_i - \overline{x})^3}{s^3}. \tag{2.3.6}$$

The sample skewness can be any value $-\infty < g_1 < \infty$. The sign of g_1 indicates the direction of skewness of the distribution. Samples that have $g_1 > 0$ indicate right-skewed distributions (or positively skewed), and samples with $g_1 < 0$ indicate left-skewed distributions (or negatively skewed). Values of g_1 near zero indicate a symmetric distribution. These are not hard and fast rules, however. The value of g_1 is subject to sampling variability and thus only provides a suggestion to the skewness of the underlying distribution.

We still need to know how big is "big", that is, how do we judge whether an observed value of g_1 is far enough away from zero for the data set to be considered skewed to the right or left? A good rule of thumb is that data sets with skewness larger than $2\sqrt{6/n}$ in magnitude are substantially skewed, in the direction of the sign of g_1. See Tabachnick & Fidell [**?**] for details.

Sample Excess Kurtosis The *sample excess kurtosis*, denoted by g_2, is given by the formula

$$g_2 = \frac{1}{n}\frac{\sum_{i=1}^{n}(x_i - \overline{x})^4}{s^4} - 3. \tag{2.3.7}$$

The sample excess kurtosis takes values $-2 \leq g_2 < \infty$. The subtraction of 3 may seem mysterious but it is done so that mound shaped samples have values of g_2 near zero. Samples with $g_2 > 0$ are called *leptokurtic*, and samples with $g_2 < 0$ are called *platykurtic*. Samples with $g_2 \approx 0$ are called *mesokurtic*.

As a rule of thumb, if $|g_2| > 4\sqrt{6/n}$ then the sample excess kurtosis is substantially different from zero in the direction of the sign of g_2. See Tabachnick & Fidell [**?**] for details.

Notice that both the sample skewness and the sample kurtosis are invariant with respect to location and scale, that is, the values of g_1 and g_2 do not depend on the measurement units of the data.

How to do it with R The `e1071` package [**?**] has the `skewness` function for the sample skewness and the `kurtosis` function for the sample excess kurtosis. Both functions have a `na.rm` argument which is `FALSE` by default.

Example 2.15. We said earlier that the `discoveries` data looked positively skewed; let's see what the statistics say:

```
library(e1071)
skewness(discoveries)
2*sqrt(6/length(discoveries))
```

```
 [1] 1.2076
 [1] 0.4898979
```

The data are definitely skewed to the right. Let us check the sample excess kurtosis of the `UKDriverDeaths` data:

```
kurtosis(UKDriverDeaths)
4*sqrt(6/length(UKDriverDeaths))
```

```
[1] 0.07133848
[1] 0.7071068
```

so that the UKDriverDeaths data appear to be mesokurtic, or at least not substantially leptokurtic.

2.4 Exploratory Data Analysis

This field was founded (mostly) by John Tukey (1915-2000). Its tools are useful when not much is known regarding the underlying causes associated with the data set, and are often used for checking assumptions. For example, suppose we perform an experiment and collect some data... now what? We look at the data using exploratory visual tools.

2.4.1 More About Stemplots

There are many bells and whistles associated with stemplots, and the stem.leaf function can do many of them.

Trim Outliers: Some data sets have observations that fall far from the bulk of the other data (in a sense made more precise in Section 2.4.4). These extreme observations often obscure the underlying structure to the data and are best left out of the data display. The trim.outliers argument (which is TRUE by default) will separate the extreme observations from the others and graph the stemplot without them; they are listed at the bottom (respectively, top) of the stemplot with the label HI (respectively LO).

Split Stems: The standard stemplot has only one line per stem, which means that all observations with first digit 3 are plotted on the same line, regardless of the value of the second digit. But this gives some stemplots a "skyscraper" appearance, with too many observations stacked onto the same stem. We can often fix the display by increasing the number of lines available for a given stem. For example, we could make two lines per stem, say, 3* and 3.. Observations with second digit 0 through 4 would go on the upper line, while observations with second digit 5 through 9 would go on the lower line. (We could do a similar thing with five lines per stem, or even ten lines per stem.) The end result is a more spread out stemplot which often looks better. A good example of this was shown on page **??**.

Depths: these are used to give insight into the balance of the observations as they accumulate toward the median. In a column beside the standard stemplot, the frequency of the stem containing the sample median is shown in parentheses. Next, frequencies are accumulated from the outside inward, including the outliers. Distributions that are more symmetric will have better balanced depths on either side of the sample median.

How to do it with R The basic command is `stem(x)` or a more sophisticated version written by Peter Wolf called `stem.leaf(x)` in the R Commander. We will describe `stem.leaf` since that is the one used by R Commander.

WARNING: Sometimes when making a stem plot the result will not be what you expected. There are several reasons for this:

- Stemplots by default will trim extreme observations (defined in Section 2.4.4) from the display. This in some cases will result in stemplots that are not as wide as expected.
- The leafs digit is chosen automatically by `stem.leaf` according to an algorithm that the computer believes will represent the data well. Depending on the choice of the digit, `stem.leaf` may drop digits from the data or round the values in unexpected ways.

Let us take a look at the `rivers` data set .

```
stem.leaf(rivers)

1 | 2: represents 120
 leaf unit: 10
            n: 141
    1     1 | 3
   29     2 | 01111333345555566667788888899
   64     3 | 000001111222233334555556666778889999
  (18)    4 | 011222233344566679
   59     5 | 000222234467
   47     6 | 0000112235789
   34     7 | 12233368
   26     8 | 04579
   21     9 | 0008
   17    10 | 035
   14    11 | 07
   12    12 | 047
    9    13 | 0
HI: 1450 1459 1770 1885 2315 2348 2533 3710
```

The stemplot shows a right-skewed shape to the `rivers` data distribution. Notice that the last digit of each of the data values were dropped from the display. Notice also that there were eight extreme observations identified by the computer, and their exact values are listed at the bottom of the stemplot. Look at the scale on the left of the stemplot and try to imagine how ridiculous the graph would have looked had we tried to include enough

stems to include these other eight observations; the stemplot would have stretched over several pages. Notice finally that we can use the depths to approximate the sample median for these data. The median lies in the row identified by (18), which means that the median is the average of the ninth and tenth observation on that row. Those two values correspond to 43 and 43, so a good guess for the median would be 430. (For the record, the sample median is $\widetilde{x} = 425$. Recall that stemplots round the data to the nearest stem-leaf pair.)

Next let us see what the precip data look like.

```
stem.leaf(precip)

1 | 2: represents 12
 leaf unit: 1
            n: 70
LO: 7 7.2 7.8 7.8
    8    1* | 1344
   13    1. | 55677
   16    2* | 024
   18    2. | 59
   28    3* | 0000111234
  (15)   3. | 555566677788899
   27    4* | 0000122222334
   14    4. | 56688899
    6    5* | 44
    4    5. | 699
HI: 67
```

Here is an example of split stems, with two lines per stem. The final digit of each datum has been dropped for the display. The data appear to be left skewed with four extreme values to the left and one extreme value to the right. The sample median is approximately 37 (it turns out to be 36.6).

2.4.2 Hinges and the Five Number Summary

Given a data set $x_1, x_2, \ldots, x_n$, the hinges are found by the following method:

- Find the order statistics $x_{(1)}, x_{(2)}, \ldots, x_{(n)}$.
- The *lower hinge* h_L is in position $L = \lfloor (n+3)/2 \rfloor / 2$, where the symbol $\lfloor x \rfloor$ denotes the largest integer less than or equal to x. If the position L is not an integer, then the hinge h_L is the average of the adjacent order statistics.
- The *upper hinge* h_U is in position $n + 1 - L$.

Given the hinges, the *five number summary* ($5NS$) is

$$5NS = (x_{(1)},\ h_L,\ \tilde{x},\ h_U,\ x_{(n)}). \tag{2.4.1}$$

An advantage of the $5NS$ is that it reduces a potentially large data set to a shorter list of only five numbers, and further, these numbers give insight regarding the shape of the data distribution similar to the sample quantiles in Section 2.3.3.

How to do it with R If the data are stored in a vector `x`, then you can compute the $5NS$ with the `fivenum` function.

2.4.3 Boxplots

A boxplot is essentially a graphical representation of the $5NS$. It can be a handy alternative to a stripchart when the sample size is large.

A boxplot is constructed by drawing a box alongside the data axis with sides located at the upper and lower hinges. A line is drawn parallel to the sides to denote the sample median. Lastly, whiskers are extended from the sides of the box to the maximum and minimum data values (more precisely, to the most extreme values that are not potential outliers, defined below).

Boxplots are good for quick visual summaries of data sets, and the relative positions of the values in the $5NS$ are good at indicating the underlying shape of the data distribution, although perhaps not as effectively as a histogram. Perhaps the greatest advantage of a boxplot is that it can help to objectively identify extreme observations in the data set as described in the next section.

Boxplots are also good because one can visually assess multiple features of the data set simultaneously:

Center can be estimated by the sample median, $\tilde{x}$.

Spread can be judged by the width of the box, $h_U - h_L$. We know that this will be close to the IQR, which can be compared to s and the MAD, perhaps after rescaling if appropriate.

Shape is indicated by the relative lengths of the whiskers, and the position of the median inside the box. Boxes with unbalanced whiskers indicate skewness in the direction of the long whisker. Skewed distributions often have the median tending in the opposite direction of skewness. Kurtosis can be assessed using the box and whiskers. A wide box with short whiskers will tend to be platykurtic, while a skinny box with wide whiskers indicates leptokurtic distributions.

Extreme observations are identified with open circles (see below).

2.4.4 Outliers

A *potential outlier* is any observation that falls beyond 1.5 times the width of the box on either side, that is, any observation less than $h_L - 1.5(h_U - h_L)$ or greater than $h_U + 1.5(h_U - h_L)$. A *suspected outlier* is any observation that falls beyond 3 times the width of the box on either side. In R, both potential and suspected outliers (if present) are denoted by open circles; there is no distinction between the two.

When potential outliers are present, the whiskers of the boxplot are then shortened to extend to the most extreme observation that is not a potential outlier. If an outlier is displayed in a boxplot, the index of the observation may be identified in a subsequent plot in `Rcmdr` by clicking the Identify outliers with mouse option in the Boxplot dialog.

What do we do about outliers? They merit further investigation. The primary goal is to determine why the observation is outlying, if possible. If the observation is a typographical error, then it should be corrected before continuing. If the observation is from a subject that does not belong to the population of interest, then perhaps the datum should be removed. Otherwise, perhaps the value is hinting at some hidden structure to the data.

How to do it with R The quickest way to visually identify outliers is with a boxplot, described above. Another way is with the `boxplot.stats` function.

Example 2.16. The `rivers` data. We will look for potential outliers in the `rivers` data.

```
boxplot.stats(rivers)$out
```

```
  [1] 1459 1450 1243 2348 3710 2315 2533 1306 1270 1885 1770
```

We may change the `coef` argument to 3 (it is 1.5 by default) to identify suspected outliers.

```
boxplot.stats(rivers, coef = 3)$out
```

```
 [1] 2348 3710 2315 2533 1885
```

2.4.5 Standardizing variables

It is sometimes useful to compare data sets with each other on a scale that is independent of the measurement units. Given a set of observed data $x_1, x_2, \ldots, x_n$ we get z scores, denoted $z_1, z_2, \ldots, z_n$, by means of the following formula

$$z_i = \frac{x_i - \overline{x}}{s}, \quad i = 1, 2, \ldots, n.$$

How to do it with R The `scale` function will rescale a numeric vector (or data frame) by subtracting the sample mean from each value (column) and/or by dividing each observation by the sample standard deviation.

2.5 Multivariate Data and Data Frames

We have had experience with vectors of data, which are long lists of numbers. Typically, each entry in the vector is a single measurement on a subject or experimental unit in the study. We saw in Section **??** how to form vectors with the `c` function or the `scan` function.

However, statistical studies often involve experiments where there are two (or more) measurements associated with each subject. We display the measured information in a rectangular array in which each row corresponds to a subject, and the columns contain the measurements for each respective variable. For instance, if one were to measure the height and weight and hair color of each of 11 persons in a research study, the information could be represented with a rectangular array. There would be 11 rows. Each row would have the person's height in the first column and hair color in the second column.

The corresponding objects in R are called *data frames*, and they can be constructed with the `data.frame` function. Each row is an observation, and each column is a variable.

Example 2.17. Suppose we have two vectors `x` and `y` and we want to make a data frame out of them.

```
x <- 5:8
y <- letters[3:6]
A <- data.frame(v1 = x, v2 = y)
```

Notice that `x` and `y` are the same length. This is *necessary*. Also notice that `x` is a numeric vector and `y` is a character vector. We may choose numeric and character vectors (or even factors) for the columns of the data frame, but each column must be of exactly one type. That is, we can have a column for `height` and a column for `gender`, but we will get an error if we try to mix function `height` (numeric) and `gender` (character or factor) information in the same column.

Indexing of data frames is similar to indexing of vectors. To get the entry in row i and column j do `A[i,j]`. We can get entire rows and columns by omitting the other index.

```
A[3, ]
A[1, ]
A[ , 2]
```

```
  v1 v2
3  7  e
```

```
  v1 v2
1  5  c
[1] c d e f
Levels: c d e f
```

There are several things happening above. Notice that `A[3,]` gave a data frame (with the same entries as the third row of `A`) yet `A[1, ]` is a numeric vector. `A[ ,2]` is a factor vector because the default setting for `data.frame` is `stringsAsFactors = TRUE`.

Data frames have a `names` attribute and the names may be extracted with the `names` function. Once we have the names we may extract given columns by way of the dollar sign.

```
names(A)
A$v1
```

```
[1] "v1" "v2"
[1] 5 6 7 8
```

The above is identical to `A[ ,1]`.

2.5.1 Bivariate Data

- Stacked bar charts
- odds ratio and relative risk
- Introduce the sample correlation coefficient.

The **sample Pearson product-moment correlation coefficient**:

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})}}$$

- independent of scale
- $-1 < r < 1$
- measures *strength* and *direction* of linear association
- Two-Way Tables. Done with `table`, or in the R Commander by following Statistics ▸ Contingency Tables ▸ Two-way Tables. You can also enter and analyze a two-way table.
- table

- prop.table
- addmargins
- rowPercents (Rcmdr)
- colPercents (Rcmdr)
- totPercents(Rcmdr)
- A <- xtabs(~ gender + race, data = RcmdrTestDrive)
- xtabs(Freq ~ Class + Sex, data = Titanic) from built in table
- barplot(A, legend.text=TRUE)
- barplot(t(A), legend.text=TRUE)
- barplot(A, legend.text=TRUE, beside = TRUE)
- spineplot(gender ~ race, data = RcmdrTestDrive)
- Spine plot: plots categorical versus categorical
- Scatterplot: look for linear association and correlation.
  - carb ~ optden, data = Formaldehyde (boring)
  - conc ~ rate, data = Puromycin
  - xyplot(accel ~ dist, data = attenu) nonlinear association
  - xyplot(eruptions ~ waiting, data = faithful) (linear, two groups)
  - xyplot(Petal.Width ~ Petal.Length, data = iris)
  - xyplot(pressure ~ temperature, data = pressure) (exponential growth)
  - xyplot(weight ~ height, data = women) (strong positive linear)

2.5.2 Multivariate Data

Multivariate Data Display

- Multi-Way Tables. You can do this with `table`,

or in R Commander by following Statistics ▸ Contingency Tables ▸ Multi-way Tables.

- Scatterplot matrix. used for displaying pairwise scatterplots simultaneously. Again, look for linear association and correlation.

- 3D Scatterplot. See Figure **??**
- `plot(state.region, state.division)`
- `barplot(table(state.division,state.region), legend.text=TRUE)`

2.6 Comparing Populations

Sometimes we have data from two or more groups (or populations) and we would like to compare them and draw conclusions. Some issues that we would like to address:

- Comparing centers and spreads: variation within versus between groups
- Comparing clusters and gaps
- Comparing outliers and unusual features
- Comparing shapes.

2.6.1 Numerically

I am thinking here about the Statistics ▸ Numerical Summaries ▸ Summarize by groups option or the Statistics ▸ Summaries ▸Table of Statistics option.

2.6.2 Graphically

- Boxplots
  - Variable width: the width of the drawn boxplots are proportional to $\sqrt{n_i}$, where n_i is the size of the i^{th} group. Why? Because many statistics have variability proportional to the reciprocal of the square root of the sample size.
  - Notches: extend to $1.58 \cdot (h_U - h_L)/\sqrt{n}$. The idea is to give roughly a 95% confidence interval for the difference in two medians. See Chapter **??**.
- Stripcharts
  - stripchart(weight ~ feed, method="stack", data=chickwts)
- Bar Graphs
  - barplot(xtabs(Freq ~ Admit + Gender, data = UCBAdmissions)) stacked bar chart
  - barplot(xtabs(Freq ~ Admit, data = UCBAdmissions))
  - barplot(xtabs(Freq ~ Gender + Admit, data = UCBAdmissions, legend = TRUE, beside = TRUE) oops, discrimination.

- barplot(xtabs(Freq ~ Admit+Dept, data = UCBAdmissions), legend = TRUE, beside = TRUE) different departments have different standards
- barplot(xtabs(Freq ~ Gender+Dept, data = UCBAdmissions), legend = TRUE, beside = TRUE) men mostly applied to easy departments, women mostly applied to difficult departments
- barplot(xtabs(Freq ~ Gender+Dept, data = UCBAdmissions), legend = TRUE, beside = TRUE)
- barchart(Admit ~ Freq, data = C)
- barchart(Admit ~ Freq|Gender, data = C)
- barchart(Admit ~ Freq | Dept, groups = Gender, data = C)
- barchart(Admit ~ Freq | Dept, groups = Gender, data = C, auto.key = TRUE)

- Histograms
  - ~ breaks | wool{*}tension, data = warpbreaks
  - ~ weight | feed, data = chickwts
  - ~ weight | group, data = PlantGrowth
  - ~ count | spray, data = InsectSprays
  - ~ len | dose, data = ToothGrowth
  - ~ decrease | treatment, data = OrchardSprays (or rowpos or colpos)
- Scatterplots

```
library(lattice)
xyplot(Petal.Width ~ Petal.Length, data = iris, group = Species)
```

Figure 2.6.1

- Scatterplot matrices
  - splom(~ cbind(GNP.deflator,GNP,Unemployed,Armed.Forces,Population,Year,Employed), data = longley)
  - splom(~ cbind(pop15,pop75,dpi), data = LifeCycleSavings)
  - splom(~ cbind(Murder, Assault, Rape), data = USArrests)
  - splom(~ cbind(CONT, INTG, DMNR), data = USJudgeRatings)
  - splom(~ cbind(area,peri,shape,perm), data = rock)

  - splom(~ cbind(Air.Flow, Water.Temp, Acid.Conc., stack.loss), data = stackloss)
  - splom(~ cbind(Fertility,Agriculture,Examination,Education,Catholic,Infant.Mortality), data = swiss)
  - splom(~ cbind(Fertility,Agriculture,Examination), data = swiss) (positive and negative)

- Dot charts
  - dotchart(USPersonalExpenditure)
  - dotchart(t(USPersonalExpenditure))
  - dotchart(WorldPhones) (transpose is no good)
  - freeny.x is no good, neither is volcano
  - dotchart(UCBAdmissions{[},,1{]})
  - dotplot(Survived ~ Freq | Class, groups = Sex, data = B)
  - dotplot(Admit ~ Freq | Dept, groups = Gender, data = C)

- Mosaic plot
  - mosaic(~ Survived + Class + Age + Sex, data = Titanic) (or just mosaic(Titanic))
  - mosaic(~ Admit + Dept + Gender, data = UCBAdmissions)

- Spine plots
  - spineplot(xtabs(Freq ~ Admit + Gender, data = UCBAdmissions))
  - # rescaled barplot

- Quantile-quantile plots: There are two ways to do this. One way is to compare two independent samples (of the same size). qqplot(x,y). Another way is to compare the sample quantiles of one variable to the theoretical quantiles of another distribution.

Given two samples $x_1, x_2, \ldots, x_n$, and $y_1, y_2, \ldots, y_n$, we may find the order statistics $x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n)}$ and $y_{(1)} \leq y_{(2)} \leq \cdots \leq y_{(n)}$. Next, plot the n points $(x_{(1)}, y_{(1)})$, $(x_{(2)}, y_{(2)}), \ldots, (x_{(n)}, y_{(n)})$.

It is clear that if $x_{(k)} = y_{(k)}$ for all $k = 1, 2, \ldots, n$, then we will have a straight line. It is also clear that in the real world, a straight line is NEVER observed, and instead we have a scatterplot that hopefully had a general linear trend. What do the rules tell us?

- If the y-intercept of the line is greater (less) than zero, then the center of the Y data is greater (less) than the center of the X data.

- If the slope of the line is greater (less) than one, then the spread of the Y data is greater (less) than the spread of the X data.

2.6.3 Lattice Graphics

The following types of plots are useful when there is one variable of interest and there is a factor in the data set by which the variable is categorized.

It is sometimes nice to set `lattice.options(default.theme = "col.whitebg")`

Side by side boxplots

```
library(lattice)
bwplot(~weight | feed, data = chickwts)
```

Figure 2.6.2: Boxplots of `weight` by `feed` type in the `chickwts` data

Histograms

```
histogram(~age | education, data = infert)
```

Figure 2.6.3: Histograms of `age` by `education` level from the `infert` data

Scatterplots

```
xyplot(Petal.Length ~ Petal.Width | Species, data = iris)
```

Figure 2.6.4: An `xyplot` of `Petal.Length` versus `Petal.Width` by `Species` in the `iris` data

Coplots

```
coplot(conc ~ uptake | Type * Treatment, data = CO2)
```

Figure 2.6.5: A coplot of conc versus uptake by Type and Treatment in the CO2 data

2.7 Exercises

Open R and issue the following commands at the command line to get started. Note that you need to have the RcmdrPlugin.IPSUR package installed, and for some exercises you need the e1071 package.

```
library(RcmdrPlugin.IPSUR)
data(RcmdrTestDrive)
attach(RcmdrTestDrive)
names(RcmdrTestDrive)
```

To load the data in the R Commander (Rcmdr), click the Data Set button, and select RcmdrTestDrive as the active data set. To learn more about the data set and where it comes from, type ?RcmdrTestDrive at the command line.

Exercise 2.1. Perform a summary of all variables in RcmdrTestDrive. You can do this with the command summary(RcmdrTestDrive).

Alternatively, you can do this in the Rcmdr with the sequence Statistics ▸ Summaries ▸ Active Data Set. Report the values of the summary statistics for each variable.

Exercise 2.2. Make a table of the race variable. Do this with Statistics ▸ Summaries ▸ IPSUR - Frequency Distributions...

1. Which ethnicity has the highest frequency?
2. Which ethnicity has the lowest frequency?
3. Include a bar graph of race. Do this with Graphs ▸ IPSUR - Bar Graph...

Exercise 2.3. Calculate the average salary by the factor gender. Do this with Statistics ▸ Summaries ▸ Table of Statistics...

1. Which gender has the highest mean salary?
2. Report the highest mean salary.
3. Compare the spreads for the genders by calculating the standard deviation of salary by gender. Which gender has the biggest standard deviation?
4. Make boxplots of salary by gender with the following method:

On the Rcmdr, click Graphs ▸ IPSUR - Boxplot... In the Variable box, select salary. Click the Plot by groups... box and select gender. Click OK. Click OK to graph the boxplot.

How does the boxplot compare to your answers to (1) and (3)?

Exercise 2.4. For this problem we will study the variable reduction.

1. Find the order statistics and store them in a vector x. *Hint:* x <- sort(reduction)
2. Find $x_{(137)}$, the 137^{th} order statistic.
3. Find the IQR.
4. Find the Five Number Summary (5NS).
5. Use the 5NS to calculate what the width of a boxplot of reduction would be.
6. Compare your answers (3) and (5). Are they the same? If not, are they close?
7. Make a boxplot of reduction, and include the boxplot in your report. You can do this with the boxplot function, or in Rcmdr with Graphs ▸ IPSUR - Boxplot...
8. Are there any potential/suspected outliers? If so, list their values. *Hint:* use your answer to (a).
9. Using the rules discussed in the text, classify answers to (8), if any, as *potential* or *suspected* outliers.

Exercise 2.5. In this problem we will compare the variables before and after. Don't forget library(e1071).

1. Examine the two measures of center for both variables. Judging from these measures, which variable has a higher center?
2. Which measure of center is more appropriate for before? (You may want to look at a boxplot.) Which measure of center is more appropriate for after?
3. Based on your answer to (2), choose an appropriate measure of spread for each variable, calculate it, and report its value. Which variable has the biggest spread? (Note that you need to make sure that your measures are on the same scale.)
4. Calculate and report the skewness and kurtosis for before. Based on these values, how would you describe the shape of before?
5. Calculate and report the skewness and kurtosis for after. Based on these values, how would you describe the shape of after?

6. Plot histograms of `before` and `after` and compare them to your answers to (4) and (5).

Exercise 2.6. Describe the following data sets just as if you were communicating with an alien, but one who has had a statistics class. Mention the salient features (data type, important properties, anything special). Support your answers with the appropriate visual displays and descriptive statistics.

1. Conversion rates of Euro currencies stored in `euro`.

2. State abbreviations stored in `state.abb`.