Copyright (c) 2015 by Nicholas J. Horton, Randall Pruim, & Daniel Kaplan.
Edition 1.2, November 2015

This material is copyrighted by the authors under a Creative Commons Attribution 3.0 Unported License. You are free to Share (to copy, distribute and transmit the work) and to Remix (to adapt the work) if you attribute our work. More detailed information about the licensing is available at this web page: http://www.mosaic-web.org/go/teachingRlicense.html.

Cover Photo: Maya Hanna.
## Contents

1. **Introduction**  
   13

2. **Getting Started with RStudio**  
   15

3. **One Quantitative Variable**  
   27

4. **One Categorical Variable**  
   39

5. **Two Quantitative Variables**  
   45

6. **Two Categorical Variables**  
   55

7. **Quantitative Response, Categorical Predictor**  
   61

8. **Categorical Response, Quantitative Predictor**  
   69

9. **Survival Time Outcomes**  
   73
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>More than Two Variables</td>
<td>75</td>
</tr>
<tr>
<td>Probability Distributions &amp; Random Variables</td>
<td>83</td>
</tr>
<tr>
<td>Power Calculations</td>
<td>89</td>
</tr>
<tr>
<td>Data Management</td>
<td>93</td>
</tr>
<tr>
<td>Health Evaluation (HELP) Study</td>
<td>107</td>
</tr>
<tr>
<td>Exercises and Problems</td>
<td>111</td>
</tr>
<tr>
<td>Bibliography</td>
<td>115</td>
</tr>
<tr>
<td>Index</td>
<td>117</td>
</tr>
</tbody>
</table>
About These Notes

We present an approach to teaching introductory and intermediate statistics courses that is tightly coupled with computing generally and with R and RStudio in particular. These activities and examples are intended to highlight a modern approach to statistical education that focuses on modeling, resampling based inference, and multivariate graphical techniques. A secondary goal is to facilitate computing with data through use of small simulation studies and appropriate statistical analysis workflow. This follows the philosophy outlined by Nolan and Temple Lang\textsuperscript{1}. The importance of modern computation in statistics education is a principal component of the recently adopted American Statistical Association’s curriculum guidelines\textsuperscript{2}.

Throughout this book (and its companion volumes), we introduce multiple activities, some appropriate for an introductory course, others suitable for higher levels, that demonstrate key concepts in statistics and modeling while also supporting the core material of more traditional courses.

A Work in Progress

These materials were developed for a workshop entitled Teaching Statistics Using R prior to the 2011 United States Conference on Teaching Statistics and revised for US-COTS 2011, USCOTS 2013, eCOTS 2014, ICOTS 9, and USCOTS 2015. We organized these workshops to help instructors integrate R (as well as some related technologies) into statistics courses at all levels. We received great feedback and many wonderful ideas from the participants and those that we’ve shared this with since the workshops.


Caution!

Despite our best efforts, you WILL find bugs both in this document and in our code. Please let us know when you encounter them so we can call in the exterminators.
Consider these notes to be a work in progress. We appreciate any feedback you are willing to share as we continue to work on these materials and the accompanying mosaic package. Drop us an email at pis@mosaic-web.org with any comments, suggestions, corrections, etc.

Updated versions will be posted at http://mosaic-web.org.

Two Audiences

We initially developed these materials for instructors of statistics at the college or university level. Another audience is the students these instructors teach. Some of the sections, examples, and exercises are written with one or the other of these audiences more clearly at the forefront. This means that

1. Some of the materials can be used essentially as is with students.

2. Some of the materials aim to equip instructors to develop their own expertise in R and RStudio to develop their own teaching materials.

Although the distinction can get blurry, and what works “as is” in one setting may not work “as is” in another, we’ll try to indicate which parts fit into each category as we go along.

R, RStudio and R Packages

R can be obtained from http://cran.r-project.org/. Download and installation are quite straightforward for Mac, PC, or linux machines.

RStudio is an integrated development environment (IDE) that facilitates use of R for both novice and expert users. We have adopted it as our standard teaching environment because it dramatically simplifies the use of R for instructors and for students. RStudio can be installed as a desktop (laptop) application or as a server application that is accessible to users via the Internet.

In addition to R and RStudio, we will make use of several packages that need to be installed and loaded separately. The mosaic package (and its dependencies) will
be used throughout. Other packages appear from time to
time as well.

**Marginal Notes**

Marginal notes appear here and there. Sometimes these
are side comments that we wanted to say, but we didn’t
want to interrupt the flow to mention them in the main
text. Others provide teaching tips or caution about traps,
pitfalls and gotchas.

**What’s Ours Is Yours – To a Point**

This material is copyrighted by the authors under a Cre-
ative Commons Attribution 3.0 Unported License. You
are free to *Share* (to copy, distribute and transmit the
work) and to *Remix* (to adapt the work) if you attribute
our work. More detailed information about the licensing
is available at this web page: http://www.mosaic-web.
org/go/teachingRlicense.html.

**Document Creation**

This document was created on November 15, 2015, using

- **knitr**, version 1.11
- **mosaic**, version 0.12.9003
- **mosaicData**, version 0.12.9003
- **R version 3.2.2** (2015-08-14)

Inevitably, each of these will be updated from time to
time. If you find that things look different on your com-
puter, make sure that your version of R and your pack-
ages are up to date and check for a newer version of this
document.

**Have a great suggestion for a marginal note? Pass it along.**

**Digging Deeper**

If you know LaTeX as well as
R, then knitr provides a nice
solution for mixing the two. We
used this system to produce
this book. We also use it for
our own research and to intro-
duce upper level students to
reproducible analysis methods.
For beginners, we introduce
knitr with RMarkdown, which
produces PDF, HTML, or Word
files using a simpler syntax.
This book is a product of Project MOSAIC, a community of educators working to develop new ways to introduce mathematics, statistics, computation, and modeling to students in colleges and universities.

The goal of the MOSAIC project is to help share ideas and resources to improve teaching, and to develop a curricular and assessment infrastructure to support the dissemination and evaluation of these approaches. Our goal is to provide a broader approach to quantitative studies that provides better support for work in science and technology. The project highlights and integrates diverse aspects of quantitative work that students in science, technology, and engineering will need in their professional lives, but which are today usually taught in isolation, if at all.

In particular, we focus on:

**Modeling**  The ability to create, manipulate and investigate useful and informative mathematical representations of a real-world situations.

**Statistics**  The analysis of variability that draws on our ability to quantify uncertainty and to draw logical inferences from observations and experiment.

**Computation**  The capacity to think algorithmically, to manage data on large scales, to visualize and interact with models, and to automate tasks for efficiency, accuracy, and reproducibility.

**Calculus**  The traditional mathematical entry point for college and university students and a subject that still has the potential to provide important insights to today’s students.
Drawing on support from the US National Science Foundation (NSF DUE-0920350), Project MOSAIC supports a number of initiatives to help achieve these goals, including:

*Faculty development and training opportunities*, such as the USCOTS 2011, USCOTS 2013, eCOTS 2014, and ICOTS 9 workshops on *Teaching Statistics Using R and RStudio*, our 2010 Project MOSAIC kickoff workshop at the Institute for Mathematics and its Applications, and our *Modeling: Early and Often in Undergraduate Calculus* AMS PREP workshops offered in 2012, 2013, and 2015.

*M-casts*, a series of regularly scheduled webinars, delivered via the Internet, that provide a forum for instructors to share their insights and innovations and to develop collaborations to refine and develop them. Recordings of M-casts are available at the Project MOSAIC web site, [http://mosaic-web.org](http://mosaic-web.org).

*The construction of syllabi and materials* for courses that teach MOSAIC topics in a better integrated way. Such courses and materials might be wholly new constructions, or they might be incremental modifications of existing resources that draw on the connections between the MOSAIC topics.

More details can be found at [http://www.mosaic-web.org](http://www.mosaic-web.org). We welcome and encourage your participation in all of these initiatives.
There are at least two ways in which statistical software can be introduced into a statistics course. In the first approach, the course is taught essentially as it was before the introduction of statistical software, but using a computer to speed up some of the calculations and to prepare higher quality graphical displays. Perhaps the size of the data sets will also be increased. We will refer to this approach as statistical computation since the computer serves primarily as a computational tool to replace pencil-and-paper calculations and drawing plots manually.

In the second approach, more fundamental changes in the course result from the introduction of the computer. Some new topics are covered, some old topics are omitted. Some old topics are treated in very different ways, and perhaps at different points in the course. We will refer to this approach as computational statistics because the availability of computation is shaping how statistics is done and taught. Computational statistics is a key component of data science, defined as the ability to use data to answer questions and communicate those results.

In practice, most courses will incorporate elements of both statistical computation and computational statistics, but the relative proportions may differ dramatically from course to course. Where on the spectrum a course lies will be depend on many factors including the goals of the course, the availability of technology for student use, the perspective of the text book used, and the comfort-level of the instructor with both statistics and computation.

Among the various statistical software packages available, R is becoming increasingly popular. The recent addition of RStudio has made R both more powerful and more accessible. Because R and RStudio are free, they have become widely used in research and industry. Training in R

Students need to see aspects of computation and data science early and often to develop deeper skills. Establishing precursors in introductory courses help them get started.
and RStudio is often seen as an important additional skill that a statistics course can develop. Furthermore, an increasing number of instructors are using R for their own statistical work, so it is natural for them to use it in their teaching as well. At the same time, the development of R and of RStudio (an optional interface and integrated development environment for R) are making it easier and easier to get started with R.

We developed the mosaic R package (available on CRAN) to make certain aspects of statistical computation and computational statistics simpler for beginners, without limiting their ability to use more advanced features of the language. The mosaic package includes a modelling approach that uses the same general syntax to calculate descriptive statistics, create graphics, and fit linear models.

Information about the mosaic package, including vignettes demonstrating features and supplementary materials (such as this book) can be found at https://cran.r-project.org/web/packages/mosaic.
1

Introduction

In this reference book, we briefly review the commands and functions needed to analyze data from introductory and second courses in statistics. This is intended to complement the *Start Teaching with R* and *Start Modeling with R* books.

Most of our examples will use data from the HELP (Health Evaluation and Linkage to Primary Care) study: a randomized clinical trial of a novel way to link at-risk subjects with primary care. More information on the dataset can be found in chapter 14.

Since the selection and order of topics can vary greatly from textbook to textbook and instructor to instructor, we have chosen to organize this material by the kind of data being analyzed. This should make it straightforward to find what you are looking for. Some data management skills are needed by students\(^1\). A basic introduction to key idioms is provided in Chapter 13.

This work leverages initiatives undertaken by Project MOSAIC (http://www.mosaic-web.org), an NSF-funded effort to improve the teaching of statistics, calculus, science and computing in the undergraduate curriculum. In particular, we utilize the mosaic package, which was written to simplify the use of R for introductory statistics courses, and the mosaicData package which includes a number of data sets. A short summary of the R commands needed to teach introductory statistics can be found in the mosaic package vignette: https://cran.r-project.org/web/packages/mosaic.

Other related resources from Project MOSAIC may be helpful, including an annotated set of examples from the sixth edition of Moore, McCabe and Craig’s *Introduction to the Practice of Statistics*\(^2\) (see http://www.amherst.edu/)

---


~nhorton/ips6e), the second and third editions of the *Statistical Sleuth*\(^3\) (see http://www.amherst.edu/~nhorton/sleuth), and *Statistics: Unlocking the Power of Data* by Lock et al (see https://github.com/rpruim/Lock5withR).

To use a package within R, it must be installed (one time), and loaded (each session). The mosaic package can be installed using the following commands:

\begin{verbatim}
> install.packages("mosaic")  # note the quotation marks
\end{verbatim}

The `#` character is a comment in R, and all text after that on the current line is ignored.

Once the package is installed (one time only), it can be loaded by running the command:

\begin{verbatim}
> require(mosaic)
\end{verbatim}

The RMarkdown system provides a simple markup language and renders the results in PDF, Word, or HTML. This allows students to undertake their analyses using a workflow that facilitates “reproducibility” and avoids cut and paste errors.

We typically introduce students to RMarkdown very early, requiring students to use it for assignments and reports\(^4\).

---


2

Getting Started with RStudio

RStudio is an integrated development environment (IDE) for R that provides an alternative interface to R that has several advantages over other default R interfaces:

- **RStudio** runs on Mac, PC, and Linux machines and provides a simplified interface that *looks and feels identical on all of them.*

  The default interfaces for R are quite different on the various platforms. This is a distractor for students and adds an extra layer of support responsibility for the instructor.

- **RStudio** can run in a web browser.

  In addition to stand-alone desktop versions, RStudio can be set up as a server application that is accessed via the internet.

  The web interface is nearly identical to the desktop version. As with other web services, users login to access their account. If students logout and login in again later, even on a different machine, their session is restored and they can resume their analysis right where they left off. With a little advanced set up, instructors can save the history of their classroom R use and students can load those history files into their own environment.

- **RStudio** provides support for reproducible research.

  RStudio makes it easy to include text, statistical analysis (R code and R output), and graphical displays all in the same document. The RMarkdown system provides a simple markup language and renders the results in HTML. The knitr/LaTeX system allows users

A series of getting started videos are available at http://www.amherst.edu/~nhorton/rstudio.

---

**Caution!**

The desktop and server version of RStudio are so similar that if you run them both, you will have to pay careful attention to make sure you are working in the one you intend to be working in.

**Note**

Using RStudio in a browser is like Facebook for statistics. Each time the user returns, the previous session is restored and they can resume work where they left off. Users can login from any device with internet access.
to combine R and LaTeX in the same document. The reward for learning this more complicated system is much finer control over the output format. Depending on the level of the course, students can use either of these for homework and projects.

- RStudio provides an integrated support for editing and executing R code and documents.

- RStudio provides some useful functionality via a graphical user interface.
  
  RStudio is not a GUI for R, but it does provide a GUI that simplifies things like installing and updating packages; monitoring, saving and loading environments; importing and exporting data; browsing and exporting graphics; and browsing files and documentation.

- RStudio provides access to the manipulate package.
  
  The manipulate package provides a way to create simple interactive graphical applications quickly and easily.

While one can certainly use R without using RStudio, RStudio makes a number of things easier and we highly recommend using RStudio. Furthermore, since RStudio is in active development, we fully expect more useful features in the future.

We primarily use an online version of RStudio. RStudio is an innovative and powerful interface to R that runs in a web browser or on your local machine. Running in the browser has the advantage that you don’t have to install or configure anything. Just login and you are good to go. Furthermore, RStudio will “remember” what you were doing so that each time you login (even on a different machine) you can pick up right where you left off. This is “R in the cloud” and works a bit like GoogleDocs or Facebook for R.

R can also be obtained from http://cran.r-project.org/. Download and installation are pretty straightforward for Mac, PC, or Linux machines. RStudio is available from http://www.rstudio.org/.

To use Markdown or knitr/LaTeX requires that the knitr package be installed on your system.
2.1 Connecting to an RStudio server

RStudio servers have been set up at a number of schools to facilitate cloud-based computing.

Once you connect to the server, you should see a login screen:

![Login screen](https://r.amherst.edu/auth-sign-in)

Once you authenticate, you should see the RStudio interface:

![RStudio interface](https://r.amherst.edu/Rstudio)

Notice that RStudio divides its world into four panels. Several of the panels are further subdivided into multi-
ple tabs. Which tabs appear in which panels can be customized by the user.

\texttt{R} can do much more than a simple calculator, and we will introduce additional features in due time. But performing simple calculations in \texttt{R} is a good way to begin learning the features of \texttt{RStudio}.

Commands entered in the \texttt{Console} tab are immediately executed by \texttt{R}. A good way to familiarize yourself with the console is to do some simple calculator-like computations. Most of this will work just like you would expect from a typical calculator. Try typing the following commands in the console panel.

\begin{verbatim}
> 5 + 3
[1] 8

> 15.3 * 23.4
[1] 358.02

> sqrt(16)  # square root
[1] 4
\end{verbatim}

This last example demonstrates how functions are called within \texttt{R} as well as the use of comments. Comments are prefaced with the \texttt{#} character. Comments can be very helpful when writing scripts with multiple commands or to annotate example code for your students.

You can save values to named variables for later reuse.

\begin{verbatim}
> product = 15.3 * 23.4  # save result
> product                # display the result
[1] 358.02

> product <- 15.3 * 23.4  # <- can be used instead of =
> product
[1] 358.02
\end{verbatim}

Once variables are defined, they can be referenced in other operations and functions.

It’s probably best to settle on using one or the other of the right-to-left assignment operators rather than to switch back and forth. We prefer the arrow operator because it represents visually what is happening in an assignment and because it makes a clear distinction between the assignment operator, the use of \texttt{=} to provide values to arguments of functions, and the use of \texttt{==} to test for equality.
> 0.5 * product  # half of the product
[1] 179.01

> log(product)  # (natural) log of the product
[1] 5.880589

> log10(product)  # base 10 log of the product
[1] 2.553907

> log2(product)  # base 2 log of the product
[1] 8.483896

> log(product, base=2)  # base 2 log of the product, another way
[1] 8.483896

The semi-colon can be used to place multiple commands on one line. One frequent use of this is to save and print a value all in one go:

> product <- 15.3 * 23.4; product  # save result and show it
[1] 358.02

2.1.1 Version information

At times it may be useful to check what version of the mosaic package, R, and RStudio you are using. Running `sessionInfo()` will display information about the version of R and packages that are loaded and `RStudio.Version()` will provide information about the version of RStudio.

> sessionInfo()

R version 3.2.2 (2015-08-14)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.10.5 (Yosemite)

locale:
2.2 Working with Files

2.2.1 Working with R Script Files

As an alternative, R commands can be stored in a file. RStudio provides an integrated editor for editing these files and facilitates executing some or all of the commands. To create a file, select File, then New File, then R Script from the RStudio menu. A file editor tab will open in the Source panel. R code can be entered here, and buttons and menu items are provided to run all the code (called sourcing the file) or to run the code on a single line or in a selected section of the file.

2.2.2 Working with RMarkdown, and knitr/LATEX

A third alternative is to take advantage of RStudio’s support for reproducible research. If you already know LATEX,
you will want to investigate the \texttt{knitr}/\LaTeX{} capabili-
ties. For those who do not already know \LaTeX{}, the sim-
pler RMarkdown system provides an easy entry into the
world of reproducible research methods. It also provides
a good facility for students to create homework and re-
ports that include text, \texttt{R} code, \texttt{R} output, and graphics.

To create a new RMarkdown file, select File, then New
File, then RMarkdown. The file will be opened with a short
template document that illustrates the mark up language.

The mosaic package includes two useful RMarkdown
templates for getting started: fancy includes bells and
whistles (and is intended to give an overview of features),
while plain is useful as a starting point for a new analy-
sis. These are accessed using the Template option when
creating a new RMarkdown file.
Click on the Knit button to convert to an HTML, PDF, or Word file.

This will generate a formatted version of the document.
There is a button (marked with a question mark) which provides a brief description of the supported markup commands. The RStudio web site includes more extensive tutorials on using RMarkdown.

It is important to remember that unlike R scripts, which are executed in the console and have access to the console environment, RMarkdown and \textit{knitr/LaTeX} files do not have access to the console environment. This is a good feature because it forces the files to be self-contained, which makes them transferable and respects good reproducible research practices. But beginners, especially if they adopt a strategy of trying things out in the console and copying and pasting successful code from the console to their file, will often create files that are incomplete and therefore do not compile correctly.

\section*{2.3 The Other Panels and Tabs}

\subsection*{2.3.1 The History Tab}

As commands are entered in the console, they appear in the History tab. These histories can be saved and loaded, there is a search feature to locate previous commands, and individual lines or sections can be transferred back to the console. Keeping the History tab open will allow you to go back and see the previous several commands. This can be especially useful when commands produce a fair amount of output and so scroll off the screen rapidly.
2.3.2 Communication between tabs

RStudio provides several ways to move R code between tabs. Pressing the Run button in the editing panel for an R script or RMarkdown or other file will copy lines of code into the Console and run them.

2.3.3 The Files Tab

The Files tab provides a simple file manager. It can be navigated in familiar ways and used to open, move, rename, and delete files. In the browser version of RStudio, the Files tab also provides a file upload utility for moving files from the local machine to the server. In RMarkdown and knitr files one can also run the code in a particular chunk or in all of the chunks in a file. Each of these features makes it easy to try out code “live” while creating a document that keeps a record of the code.

In the reverse direction, code from the history can be copied either back into the console to run them again (perhaps after editing) or into one of the file editing tabs for inclusion in a file.

2.3.4 The Help Tab

The Help tab is where RStudio displays R help files. These can be searched and navigated in the Help tab. You can also open a help file using the ? operator in the console. For example the following command will provide the help file for the logarithm function.

```r
> ?log
```

2.3.5 The Environment Tab

The Environment tab shows the objects available to the console. These are subdivided into data, values (non-dataframe, non-function objects) and functions. The broom icon can be used to remove all objects from the environment, and it is good to do this from time to time, especially when running in RStudio server or if you choose
to save the environment when shutting down RStudio since in these cases objects can stay in the environment essentially indefinitely.

2.3.6 The Plots Tab

Plots created in the console are displayed in the Plots tab. For example, the following commands display the number of births in the United States for each day in 1978.

```
> # this will make lattice graphics available to the session
> # as well as the Births78 dataset
> require(mosaic)
> xyplot(births ~ dayofyear, data=Births78)
```

From the Plots tab, you can navigate to previous plots and also export plots in various formats after interactively resizing them.

2.3.7 The Packages Tab

Much of the functionality of R is located in packages, many of which can be obtained from a central clearing house called CRAN (Comprehensive R Archive Network). The Packages tab facilitates installing and loading packages. It will also allow you to search for packages that have been updated since you installed them.
3
One Quantitative Variable

3.1 Numerical summaries

R includes a number of commands to numerically summarize variables. These include the capability of calculating the mean, standard deviation, variance, median, five number summary, interquartile range (IQR) as well as arbitrary quantiles. We will illustrate these using the CESD (Center for Epidemiologic Studies–Depression) measure of depressive symptoms (which takes on values between 0 and 60, with higher scores indicating more depressive symptoms).

To improve the legibility of output, we will also set the default number of digits to display to a more reasonable level (see \texttt{options()} for more configuration possibilities).

\begin{verbatim}
> require(mosaic)
> require(mosaicData)
> options(digits=3)
> mean(~ cesd, data=HELPrc)
\end{verbatim}

\begin{verbatim}
[1] 32.8
\end{verbatim}

Note that the \texttt{mean()} function in the \texttt{mosaic} package supports a formula interface common to lattice graphics and linear models (e.g., \texttt{lm()}). The \texttt{mosaic} package provides many other functions that use the same notation, which we will be using throughout this document.

The same output could be created using the following commands (though we will use the MOSAIC versions when available).

\textbf{Digging Deeper}
If you have not seen the formula notation before, the companion book, \textit{Start Teaching with R} provides a detailed presentation. \textit{Start Modeling with R}, another companion book, details the relationship between the process of modeling and the formula notation.
Similar functionality exists for other summary statistics.

```r
> sd(~ cesd, data=HELPct)  # Variance
[1] 12.5
> sd(~ cesd, data=HELPct)^2
[1] 157
> var(~ cesd, data=HELPct)
[1] 157
```

It is also straightforward to calculate quantiles of the distribution.

```r
> median(~ cesd, data=HELPct)  # Median
[1] 34
```

By default, the `quantile()` function displays the quartiles, but can be given a vector of quantiles to display.

```r
> with(HELPct, quantile(cesd))
     0%   25%   50%   75%  100%
        1   25  34  41    60

> with(HELPct, quantile(cesd, c(.025, .975)))
     2.5%  97.5%
       6.3   55.0
```

---

**Caution!**

Not all commands have been upgraded to support the formula interface. For these functions, variables within dataframes must be accessed using `with()` or the `$` operator.
Finally, the `favstats()` function in the mosaic package provides a concise summary of many useful statistics.

```r
> favstats(~ cesd, data=HELPrct)

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>min</td>
<td>Q1</td>
<td>median</td>
<td>Q3</td>
<td>max</td>
<td>mean</td>
<td>sd</td>
<td>n</td>
</tr>
<tr>
<td>1</td>
<td>25</td>
<td>34</td>
<td>41</td>
<td>60</td>
<td>32.8</td>
<td>12.5</td>
<td>453</td>
</tr>
</tbody>
</table>
```

### 3.2 Graphical summaries

The `histogram()` function is used to create a histogram. Here we use the formula interface (as discussed in the Start Modeling with R book) to specify that we want a histogram of the CESD scores.

```r
> histogram(~ cesd, data=HELPrct)
```

![Histogram of CESD scores](image)

We can use the `width()` and `center()` options to control the location of the bins.

```r
> histogram(~ cesd, width=5, center=2.5, data=HELPrct)
```
In the HELPrct dataset, approximately one quarter of the subjects are female.

```r
> tally(~ sex, data=HELPrct)

defemale  male
107    346

> tally(~ sex, format="percent", data=HELPrct)

defemale  male
23.6    76.4
```

It is straightforward to restrict our attention to just the female subjects. If we are going to do many things with a subset of our data, it may be easiest to make a new dataframe containing only the cases we are interested in. The `filter()` function in the `dplyr` package can be used to generate a new dataframe containing just the women or just the men (see also section 13.5). Once this is created, the `stem()` function is used to create a stem and leaf plot.

```r
> female <- filter(HELPrct, sex=='female')
> male <- filter(HELPrct, sex=='male')
> with(female, stem(cesd))
```

The decimal point is 1 digit(s) to the right of the |
Subsets can also be generated and used “on the fly” (this time including an overlaid normal density):

```r
> histogram(~ cesd, fit="normal",
data=filter(HELPrct, sex=='female'))
```

Alternatively, we can make side-by-side plots to compare multiple subsets.

```r
> histogram(~ cesd | sex, data=HELPrct)
```
The layout can be rearranged.

> `histogram(~ cesd | sex, layout=c(1, 2), data=HELPrct)`

We can control the number of bins in a number of ways. These can be specified as the total number.

> `histogram(~ cesd, nint=20, data=female)`
The width of the bins can be specified.

```r
> histogram(~ cesd, width=2, data=female)
```

The `dotPlot()` function is used to create a dotplot for a smaller subset of subjects (homeless females). We also demonstrate how to change the x-axis label.

```r
> dotPlot(~ cesd, xlab="CESD score",
         data=filter(HELPrct, (sex=="female") & (homeless=="homeless")))
```
3.3 Density curves

One disadvantage of histograms is that they can be sensitive to the choice of the number of bins. Another display to consider is a density curve.

Here we adorn a density plot with some additions to demonstrate how to build up a graphic for pedagogical purposes. We add some text, a superimposed normal density as well as a vertical line. A variety of line types and colors can be specified, as well as line widths.

```r
> densityplot(~ cesd, data=female)
> ladd(grid.text(x=0.2, y=0.8, 'only females'))
> ladd(panel.mathdensity(args=list(mean=mean(cesd),
                       sd=sd(cesd)), col="red"), data=female)
> ladd(panel.abline(v=60, lty=2, lwd=2, col="grey"))
```

![Density curve with annotations](image)

Density plots are also sensitive to certain choices. If your density plot is too jagged or too smooth, try changing the adjust argument: larger than 1 for smoother plots, less than 1 for more jagged plots.

Digging Deeper

The `plotFun()` function can also be used to annotate plots (see section 10.2.1).

3.4 Frequency polygons

A third option is a frequency polygon, where the graph is created by joining the midpoints of the top of the bars of a histogram.
> `freqpolygon(~ cesd, data=female)`

3.5 **Normal distributions**

The most famous density curve is a normal distribution. The `xpnorm()` function displays the probability that a random variable is less than the first argument, for a normal distribution with mean given by the second argument and standard deviation by the third. More information about probability distributions can be found in section 11.

> `xpnorm(1.96, mean=0, sd=1)`

If $X \sim N(0,1)$, then

$P(X \leq 1.96) = P(Z \leq 1.96) = 0.975$

$P(X > 1.96) = P(Z > 1.96) = 0.025$

[1] 0.975
3.6 Inference for a single sample

We can calculate a 95% confidence interval for the mean CESD score for females by using a t-test:

```r
> t.test(~ cesd, data=female)
```

One Sample t-test

data: data$cesd
t = 30, df = 100, p-value <2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
  34.4 39.4
sample estimates:
mean of x
  36.9

> confint(t.test(~ cesd, data=female))

    mean of x lower upper level
1   36.9 34.4 39.4 0.95

But it’s also straightforward to calculate this using a bootstrap. The statistic that we want to resample is the mean.

```r
> mean(~ cesd, data=female)
```

[1] 36.9

One resampling trial can be carried out:

```r
> mean(~ cesd, data=resample(female))
```

[1] 37.7

Another will yield different results:

```r
> mean(~ cesd, data=resample(female))
```

[1] 34.9

Digging Deeper

More details and examples can be found in the mosaic package Resampling Vignette.

Here we sample with replacement from the original dataframe, creating a resampled dataframe with the same number of rows.

Even though a single trial is of little use, it’s smart having students do the calculation to show that they are (usually!) getting a different result than without resampling.
Now conduct 1000 resampling trials, saving the results in an object called trials:

```r
> trials <- do(1000) * mean(~ cesd, data=resample(female))
> head(trials, 3)

mean
1 36.7
2 35.5
3 35.9

> qdata(~ mean, c(.025, .975), data=trials)

        quantile     p
2.5% 34.3 0.025
97.5% 39.4 0.975
```
4
One Categorical Variable

4.1 Numerical summaries

The `tally()` function can be used to calculate counts, percentages and proportions for a categorical variable.

```r
> tally(~ homeless, data=HELPct)

homeless  housed
  209    244

> tally(~ homeless, margins=TRUE, data=HELPct)

homeless  housed  Total
  209    244     453

> tally(~ homeless, format="percent", data=HELPct)

homeless  housed
  46.1    53.9

> tally(~ homeless, format="proportion", data=HELPct)

homeless  housed
  0.461    0.539
```

Digging Deeper
The *Start Teaching with R* companion book introduces the formula notation used throughout this book. See also *Start Teaching with R* for the connections to statistical modeling.
4.2 The binomial test

An exact confidence interval for a proportion (as well as a test of the null hypothesis that the population proportion is equal to a particular value [by default 0.5]) can be calculated using the `binom.test()` function. The standard `binom.test()` requires us to tabulate.

```r
> binom.test(209, 209 + 244)
```

data: 209 out of 209 + 244
number of successes = 200, number of trials = 500, p-value = 0.1
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
0.415 0.509
sample estimates:
probability of success
0.461

The mosaic package provides a formula interface that avoids the need to pre-tally the data.

```r
> result <- binom.test(~ (homeless=="homeless"), data=HELPrct)
> result
```

data: HELPrct$homeless == "homeless" [with success = TRUE]
number of successes = 200, number of trials = 500, p-value = 0.1
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
0.415 0.509
sample estimates:
probability of success
0.461

As is generally the case with commands of this sort, there are a number of useful quantities available from the object returned by the function.
> names(result)

[1] "statistic" "parameter" "p.value" "conf.int"
[5] "estimate" "null.value" "alternative" "data.name"

These can be extracted using the $ operator or an extractor function. For example, the user can extract the confidence interval or p-value.

> result$statistic

number of successes
209

> confint(result)

probability of success lower upper level
1 0.461 0.415 0.509 0.95

> pval(result)

p.value
0.11

4.3 The proportion test

A similar interval and test can be calculated using the function prop.test(). Here is a count of the number of people at each of the two levels of homeless

> tally(~ homeless, data=HELPrct)

homeless housed
209 244

The prop.test() function will carry out the calculations of the proportion test and report the result.

> prop.test(~ (homeless=="homeless"), correct=FALSE, data=HELPrct)
1-sample proportions test without continuity correction

data: HELPrct$(homeless == "homeless") [with success = TRUE]
X-squared = 3, df = 1, p-value = 0.1
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.416 0.507
sample estimates:
p 0.461

In this statement, prop.test is examining the homeless variable in the same way that tally() would. prop.test() can also work directly with numerical counts, the way binom.test() does.

> prop.test(209, 209 + 244, correct=FALSE)

1-sample proportions test without continuity correction

data: 209 out of 209 + 244
X-squared = 3, df = 1, p-value = 0.1
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.416 0.507
sample estimates:
p 0.461

4.4 Goodness of fit tests

A variety of goodness of fit tests can be calculated against a reference distribution. For the HELP data, we could test the null hypothesis that there is an equal proportion of subjects in each substance abuse group back in the original populations.

> tally(~ substance, format="percent", data=HELPrct)

alcohol cocaine heroin
39.1 33.6 27.4
a student’s guide to r

> observed <- tally(~ substance, data=HELPrc)
> observed

<table>
<thead>
<tr>
<th>substance</th>
<th>alcohol</th>
<th>cocaine</th>
<th>heroin</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>177</td>
<td>152</td>
<td>124</td>
</tr>
</tbody>
</table>

> p <- c(1/3, 1/3, 1/3) # equivalent to rep(1/3, 3)
> chisq.test(observed, p=p)

Chi-squared test for given probabilities

data:  observed
X-squared = 9, df = 2, p-value = 0.01

> total <- sum(observed); total
[1] 453

> expected <- total*p; expected
[1] 151 151 151

We can also calculate the $\chi^2$ statistic manually, as a function of observed and expected values.

> chisq <- sum(((observed - expected)^2/(expected)); chisq
[1] 9.31

> 1 - pchisq(chisq, df=2)
[1] 0.00951

It may be helpful to consult a graph of the statistic, where the shaded area represents the value to the right of the observed value.

> plotDist("chisq", df=2, groups = x > 9.31, type="h")

Caution!
In addition to the format option, there is an option margins to include marginal totals in the table. The default in tally() is margins=FALSE. Try it out!
Alternatively, the mosaic package provides a version of `chisq.test()` with more verbose output.

```r
> xchisq.test(observed, p=p)
```

Chi-squared test for given probabilities

data:  x  
X-squared = 9, df = 2, p-value = 0.01

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>177</td>
<td>152</td>
<td>124</td>
</tr>
<tr>
<td>(151.00)</td>
<td>(151.00)</td>
<td>(151.00)</td>
</tr>
<tr>
<td>4.4768</td>
<td>0.0066</td>
<td>4.8278</td>
</tr>
<tr>
<td>&lt; 2.116</td>
<td>&lt; 0.081</td>
<td>&lt;-2.197</td>
</tr>
</tbody>
</table>

key:
- observed  
- (expected)  
- [contribution to X-squared]  
- <residual>

> # clean up variables no longer needed  
> rm(observed, p, total, chisq)

x in `xchisq.test()` stands for eXtra.

Objects in the workspace are listed in the Environment tab in RStudio. If you want to clean up that listing, remove objects that are no longer needed with `rm()`.
5
Two Quantitative Variables

5.1 Scatterplots

We always encourage students to start any analysis by graphing their data. Here we augment a scatterplot of the CESD (a measure of depressive symptoms, higher scores indicate more symptoms) and the MCS (mental component score from the SF-36, where higher scores indicate better functioning) for female subjects with a lowess (locally weighted scatterplot smoother) line, using a circle as the plotting character and slightly thicker line.

> females <- filter(HELPct, female==1)
> xyplot(cesd ~ mcs, type=c("p","smooth"), pch=1,
  cex=0.6, lwd=3, data=females)

The lowess line can help to assess linearity of a relationship. This is added by specifying both points (using ‘p’) and a lowess smoother.
It’s straightforward to plot something besides a character in a scatterplot. In this example, the USArrests can be used to plot the association between murder and assault rates, with the state name displayed. This requires a panel function to be written.

```r
> panel.labels <- function(x, y, labels='x', ...) {
  panel.text(x, y, labels, cex=0.4, ...)
}
> xyplot(Murder ~ Assault, panel=panel.labels,
  labels=rownames(USArrests), data=USArrests)
```

5.2 Correlation

Correlations can be calculated for a pair of variables, or for a matrix of variables.

```r
> cor(cesd, mcs, data=females)
[1] -0.674
```

```r
> smallHELP <- select(females, cesd, mcs, pcs)
> cor(smallHELP)

         cesd  mcs  pcs
cesd 1.000 -0.674 -0.369
mcs  -0.674 1.000 0.266
pcs  -0.369 0.266 1.000
```

By default, Pearson correlations are provided. Other variants (e.g., Spearman) can be specified using the method...
5.3 Pairs plots

A pairs plot (scatterplot matrix) can be calculated for each pair of a set of variables.

```r
> splom(smallHELP, cex=0.3)
```

The GGally package has support for more elaborate pairs plots.

5.4 Simple linear regression

Linear regression models are described in detail in Start Modeling with R. These use the same formula interface introduced previously for numerical and graphical summaries to specify the outcome and predictors. Here we consider fitting the model `cesd ~ mcs`.

```r
> cesdmodel <- lm(cesd ~ mcs, data=females)
> coef(cesdmodel)
```

We tend to introduce linear regression early in our courses, as a purely descriptive technique.
(Intercept)    mcs
  57.349   -0.707

To simplify the output, we turn off the option to display significance stars.

```R
> options(show.signif.stars=FALSE)
> coef(cesdmodel)

(Intercept)    mcs
  57.349   -0.707

> msummary(cesdmodel)

                         Estimate  Std. Error    t value  Pr(>|t|)
(Intercept)      57.3485      2.3806     24.09  < 2e-16
mcs              -0.7070      0.0757     -9.34   1.8e-15

Residual standard error: 9.66 on 105 degrees of freedom
Multiple R-squared:  0.454, Adjusted R-squared:  0.449
F-statistic: 87.3 on 1 and 105 DF,  p-value: 1.81e-15

> coef(summary(cesdmodel))

                         Estimate  Std. Error    t value  Pr(>|t|)
(Intercept)      57.3490      2.3806     24.09  1.42e-44
mcs              -0.7070      0.0757     -9.34   1.81e-15

> confint(cesdmodel)

                          2.5 %     97.5 %
(Intercept)      52.6280     62.0696
mcs              -0.8570     -0.5570

> rsquared(cesdmodel)

[1] 0.454

> class(cesdmodel)

[1] "lm"

The return value from `lm()` is a linear model object. A number of functions can operate on these objects, as seen previously with `coef()`. The function `residuals()` returns a vector of the residuals.

The function `residuals()` can be abbreviated `resid()`. Another useful function is `fitted()`, which returns a vector of predicted values.
> `histogram(~ residuals(cesdmodel), density=TRUE)`

> `qqmath(~ resid(cesdmodel))`

> `xyplot(resid(cesdmodel) ~ fitted(cesdmodel), type=c("p", "smooth", "r"),
        alpha=0.5, cex=0.3, pch=20)`
The \texttt{mplot()} function can facilitate creating a variety of useful plots, including the same residuals vs. fitted scatterplots, by specifying the \texttt{which=1} option.

\begin{verbatim}
> mplot(cesdmodel, which=1)
[[1]]
\end{verbatim}

It can also generate a normal quantile-quantile plot (\texttt{which=2}),

\begin{verbatim}
> mplot(cesdmodel, which=2)
\end{verbatim}
scale vs. location,

> mplot(cesdmodel, which=3)

Cook’s distance by observation number,

> mplot(cesdmodel, which=4)
residuals vs. leverage,

> mplot(cesdmodel, which=5)

and Cook’s distance vs. leverage.

> mplot(cesdmodel, which=6)
Prediction bands can be added to a plot using the `panel.lmbands()` function.

> `xyplot(cesd ~ mcs, panel=panel.lmbands, cex=0.2, band.lwd=2, data=HELPrct)`
6
Two Categorical Variables

6.1 Cross classification tables

Cross classification (two-way or $R$ by $C$) tables can be constructed for two (or more) categorical variables. Here we consider the contingency table for homeless status (homeless one or more nights in the past 6 months or housed) and sex.

```r
> tally(~ homeless + sex, margins=FALSE, data=HELPrct)

   homeless    sex
     female  male
  homeless  40  169
  housed   67  177
```

We can also calculate column percentages:

```r
> tally(~ sex | homeless, margins=TRUE, format="percent", data=HELPrct)

   homeless    sex
     homeless housed
   female   19.1  27.5
   male     80.9  72.5
 Total  100.0 100.0
```

We can calculate the odds ratio directly from the table:
The mosaic package has a function which will calculate odds ratios:

```r
> oddsRatio(tally(~ (homeless=="housed") + sex, margins=FALSE, data=HELPrc))
```

[1] 0.625

The CrossTable() function in the gmodels package also displays a cross classification table.

```r
> require(gmodels)
> with(HELPrc, CrossTable(homeless, sex, prop.r=FALSE, prop.chisq=FALSE, prop.t=FALSE))
```

Cell Contents
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-------------------------</td>
</tr>
<tr>
<td>N</td>
</tr>
<tr>
<td>N / Col Total</td>
</tr>
<tr>
<td>-------------------------</td>
</tr>
</tbody>
</table>

Total Observations in Table: 453

<table>
<thead>
<tr>
<th></th>
<th>sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>homeless</td>
<td>female</td>
</tr>
<tr>
<td></td>
<td>male</td>
</tr>
<tr>
<td>Row Total</td>
<td>---------</td>
</tr>
<tr>
<td>---------------</td>
<td>---------</td>
</tr>
<tr>
<td>homeless</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>169</td>
</tr>
<tr>
<td></td>
<td>209</td>
</tr>
<tr>
<td></td>
<td>0.374</td>
</tr>
<tr>
<td></td>
<td>0.488</td>
</tr>
<tr>
<td>housed</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>177</td>
</tr>
<tr>
<td></td>
<td>244</td>
</tr>
<tr>
<td></td>
<td>0.626</td>
</tr>
<tr>
<td></td>
<td>0.512</td>
</tr>
<tr>
<td>Column Total</td>
<td>107</td>
</tr>
<tr>
<td></td>
<td>346</td>
</tr>
<tr>
<td></td>
<td>453</td>
</tr>
<tr>
<td></td>
<td>0.236</td>
</tr>
<tr>
<td></td>
<td>0.764</td>
</tr>
</tbody>
</table>
Graphical summaries of cross classification tables may be helpful in visualizing associations. Mosaic plots are one example, where the total area (all observations) is proportional to one. Here we see that males tend to be over-represented amongst the homeless subjects (as represented by the horizontal line which is higher for the homeless rather than the housed).

```r
> mytab <- tally(~ homeless + sex, margins=FALSE, data=HELPrct)
> mosaicplot(mytab)
```

Caution!
The jury is still out regarding the utility of mosaic plots (also known as eikosograms), relative to the low data to ink ratio. We have found them to be helpful to reinforce understanding of a two way contingency table.


The `mosaic()` function in the vcd package also makes mosaic plots.

### 6.2 Creating tables from summary statistics

Tables can be created from summary statistics using the `do()` function.

```r
> HELPtable <- rbind(
  do(40) * data.frame(sex="female", homeless="homeless"),
  do(169) * data.frame(sex="male", homeless="homeless"),
  do(67) * data.frame(sex="female", homeless="housed"),
```

>```
\texttt{do(177) * data.frame(sex="male", homeless="housed")}

\texttt{> tally(~ homeless + sex, data=HELPtable)}

\begin{verbatim}
  sex
  homeless female male
  homeless     40 169
  housed       67 177
\end{verbatim}

### 6.3 Chi-squared tests

\texttt{> chisq.test(tally(~ homeless + sex, margins=FALSE, data=HELPrct), correct=FALSE)}

Pearson’s Chi-squared test

data: tally(~homeless + sex, margins = FALSE, data = HELPrct)
X-squared = 4, df = 1, p-value = 0.04

There is a statistically significant association found: it is unlikely that we would observe an association this strong if homeless status and sex were independent in the population.

When a student finds a significant association, it’s important for them to be able to interpret this in the context of the problem. The \texttt{xchisq.test()} function provides additional details (observed, expected, contribution to statistic, and residual) to help with this process.

\texttt{> xchisq.test(tally(~homeless + sex, margins=FALSE, data=HELPrct), correct=FALSE)}

Pearson’s Chi-squared test

data: x
X-squared = 4, df = 1, p-value = 0.04

\begin{verbatim}
  40 169
\end{verbatim}
We observe that there are fewer homeless women, and more homeless men that would be expected.

### 6.4 Fisher’s exact test

An exact test can also be calculated. This is computationally straightforward for 2 by 2 tables. Options to help constrain the size of the problem for larger tables exist (see `fisher.test()`).

```r
> fisher.test(tally(~homeless + sex, margins=FALSE, data=HELPrct))
```

Fisher's Exact Test for Count Data

data:  tally(~homeless + sex, margins = FALSE, data = HELPct)  
p-value = 0.05  
alternative hypothesis: true odds ratio is not equal to 1  
95 percent confidence interval:  
0.389 0.997  
sample estimates:  
odds ratio  
0.626

**Digging Deeper**
Note the different estimate of the odds ratio from that seen in section 6.1. The `fisher.test()` function uses a different estimator (and different interval based on the profile likelihood).
7
Quantitative Response, Categorical Predictor

7.1 A dichotomous predictor: numerical and graphical summaries

Here we will compare the distributions of CESD scores by sex. The `mean()` function can be used to calculate the mean CESD score separately for males and females.

```r
> mean(cesd ~ sex, data=HELPct)
female   male
  36.9   31.6
```

The `favstats()` function can provide more statistics by group.

```r
> favstats(cesd ~ sex, data=HELPct)
   sex min Q1 median Q3 max  mean   sd  n missing
1 female 3 29 38.0    46.5 60 36.9 13.0  107      0
2  male   1 24 32.5    40.0 58 31.6 12.1  346      0
```

Boxplots are a particularly helpful graphical display to compare distributions. The `bwplot()` function can be used to display the boxplots for the CESD scores separately by sex. We see from both the numerical and graphical summaries that women tend to have slightly higher CESD scores than men.

```r
> bwplot(sex ~ cesd, data=HELPct)
```

Although we usually put explanatory variables along the horizontal axis, page layout sometimes makes the other orientation preferable for these plots.
When sample sizes are small, there is no reason to summarize with a boxplot since `xyplot()` can handle categorical predictors. Even with 10–20 observations in a group, a scatter plot is often quite readable. Setting the alpha level helps detect multiple observations with the same value.

```r
> xyplot(sex ~ length, alpha=.6, cex=1.4, data=KidsFeet)
```

One of us once saw a biologist proudly present side-by-side boxplots. Thinking a major victory had been won, he naively asked how many observations were in each group. “Four,” replied the biologist.

### 7.2 A dichotomous predictor: two-sample t

The Student’s two sample t-test can be run without (default) or with an equal variance assumption.

```r
> t.test(cesd ~ sex, var.equal=FALSE, data=HELPct)
```

```
Welch Two Sample t-test

data:  cesd by sex
t = 4, df = 200, p-value = 3e-04
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  2.49 8.09
```
sample estimates:
mean in group female  mean in group male
36.9               31.6

We see that there is a statistically significant difference between the two groups.

We can repeat using the equal variance assumption.

> `t.test(cesd ~ sex, var.equal=TRUE, data=HELPct)`

Two Sample t-test

data:  cesd by sex
t = 4, df = 500, p-value = 1e-04
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  2.61 7.97
sample estimates:
mean in group female  mean in group male
36.9               31.6

The groups can also be compared using the `lm()` function (also with an equal variance assumption). The mosaic command `msummary()` provides a slightly terser version of the typical output from `summary()`.

> `msummary(lm(cesd ~ sex, data=HELPct))`

| Estimate | Std. Error | t value | Pr(>|t|) |
|----------|------------|---------|----------|
| (Intercept) | 36.89 | 1.19 | 30.96 | < 2e-16 |
| sexmale | -5.29 | 1.36 | -3.88 | 0.00012 |

Residual standard error: 12.3 on 451 degrees of freedom
Multiple R-squared:  0.0323, Adjusted R-squared:  0.0302
F-statistic: 15.1 on 1 and 451 DF,  p-value: 0.00012

### 7.3 Non-parametric 2 group tests

The same conclusion is reached using a non-parametric (Wilcoxon rank sum) test.
> wilcox.test(cesd ~ sex, data=HELPrct)

Wilcoxon rank sum test with continuity correction
data:  cesd by sex
W = 20000, p-value = 1e-04
alternative hypothesis: true location shift is not equal to 0

7.4 Permutation test

Here we extend the methods introduced in section 3.6 to undertake a two-sided test comparing the ages at baseline by gender. First we calculate the observed difference in means:

> mean(age ~ sex, data=HELPrct)
female male
36.3 35.5

> test.stat < - diffmean(age ~ sex, data=HELPrct)
> test.stat
diffmean
-0.784

We can calculate the same statistic after shuffling the group labels:

> do(1) * diffmean(age ~ shuffle(sex), data=HELPrct)
diffmean
1  0.268

> do(1) * diffmean(age ~ shuffle(sex), data=HELPrct)
diffmean
1  0.0357

> do(3) * diffmean(age ~ shuffle(sex), data=HELPrct)
diffmean
1  -0.515
2  -0.613
3   0.305
> rtest.stats <- do(500) * diffmean(age ~ shuffle(sex), data=HELPct)
> head(rtest.stats, 3)

  diffmean
  1  -0.1479
  2   0.3171
  3   0.0479

> favstats(~ diffmean, data=rtest.stats)

  min      Q1    median    Q3   max      mean     sd    n missing
-2.2  -0.527    0.0602  0.647  2.3  0.0502  0.871  500       0

> histogram(~ diffmean, n=40, xlim=c(-6, 6), 
          groups=diffmean >= test.stat, pch=16, cex=.8, 
          data=rtest.stats)
> ladd(panel.abline(v=test.stat, lwd=3, col="red"))

Here we don’t see much evidence to contradict the null hypothesis that men and women have the same mean age in the population.

7.5 One-way ANOVA

Earlier comparisons were between two groups. We can also consider testing differences between three or more groups using one-way ANOVA. Here we compare CESD scores by primary substance of abuse (heroin, cocaine, or
alcohol) with a line rather a dot to indicate the median.

```r
> bwplot(cesd ~ substance, pch="|", data=HELPrct)
```

```r
cesd
0 10 20 30 40 50 60
alcohol cocaine heroin
●
```

```r
> mean(cesd ~ substance, data=HELPrct)
```

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>alcohol</td>
<td>34.4</td>
<td>29.4</td>
<td>34.9</td>
<td></td>
</tr>
<tr>
<td>cocaine</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>heroin</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```r
> anovamod <- aov(cesd ~ substance, data=HELPrct)
> summary(anovamod)
```

```
Analysis of Variance Table

Response: cesd

Df  Sum Sq Mean Sq F value Pr(>F)
substance  2 2704 1352  8.94 0.00016   
Residuals 450 68084 151
```

While still high (scores of 16 or more are generally considered to be “severe” symptoms), the cocaine-involved group tend to have lower scores than those whose primary substances are alcohol or heroin.

```r
> modintercept <- lm(cesd ~ 1, data=HELPrct)
> modsubstance <- lm(cesd ~ substance, data=HELPrct)
```

The `anova()` command can summarize models.

```r
> anova(modsubstance)
```

```
Analysis of Variance Table

Response: cesd

Df  Sum Sq Mean Sq F value Pr(>F)
substance  2 2704 1352  8.94 0.00016   
Residuals 450 68084 151
```
In this setting the results are identical (since there is only one predictor, with 2 df).

The `anova()` function can also be used to formally compare two (nested) models.

```r
> anova(modintercept, modsubstance)
Analysis of Variance Table

Model 1: cesd ~ 1
Model 2: cesd ~ substance

  Res.Df RSS Df Sum of Sq F Pr(>F)
1   452 70788
2   450 68084  2    2704 8.94 0.00016
```

### 7.6 Tukey’s Honest Significant Differences

There are a variety of multiple comparison procedures that can be used after fitting an ANOVA model. One of these is Tukey’s Honest Significant Differences (HSD). Other options are available within the `multcomp` package.

```r
> favstats(cesd ~ substance, data=HELPrct)
substance min Q1 median Q3 max mean sd n missing
 1 alcohol 4 26 36 42 58 34.4 12.1 177     0
 2 cocaine 1 19 30 39 60 29.4 13.4 152     0
 3 heroin 4 28 35 43 56 34.9 11.2 124     0

> HELPrct <- mutate(HELPrct, subgrp = factor(substance,
levels=c("alcohol", "cocaine", "heroin"),
labels=c("A", "C", "H")))
> mod <- lm(cesd ~ subgrp, data=HELPrct)
> HELPHSD <- TukeyHSD(mod, "subgrp")
> HELPHSD

  Tukey multiple comparisons of means
  95% family-wise confidence level

  Fit: aov(formula = x)
Again, we see that the cocaine group has significantly lower CESD scores than either of the other two groups.
Categorical Response, Quantitative Predictor

8.1 Logistic regression

Logistic regression is available using the `glm()` function, which supports a variety of link functions and distributional forms for generalized linear models, including logistic regression.

```r
> logitmod <- glm(homeless ~ age + female, family=binomial, data=HELPrct)
> msummary(logitmod)

Coefficients:

|              | Estimate | Std. Error | z value | Pr(>|z|) |
|--------------|----------|------------|---------|----------|
| (Intercept)  | 0.8926   | 0.4537     | 1.97    | 0.049    |
| age          | -0.0239  | 0.0124     | -1.92   | 0.055    |
| female       | 0.4920   | 0.2282     | 2.16    | 0.031    |

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 625.28 on 452 degrees of freedom
Residual deviance: 617.19 on 450 degrees of freedom
AIC: 623.2

Number of Fisher Scoring iterations: 4

> exp(coef(logitmod))

(Intercept)   age   female
     2.442  0.976  1.636

> exp(confint(logitmod))

Waiting for profiling to be done...

The `glm()` function has argument `family`, which can take an option `link`. The `logit` link is the default link for the binomial family, so we don’t need to specify it here. The more verbose usage would be `family=binomial(link=logit).`
We can compare two models (for multiple degree of freedom tests). For example, we might be interested in the association of homeless status and age for each of the three substance groups.

```r
> mymodsubage <- glm(homeless=="homeless" ~ age + substance,
  family=binomial, data=HELPrct)
> mymoddage <- glm(homeless=="homeless" ~ age, family=binomial,
  data=HELPrct)
> msummary(mymodsubage)
```

Coefficients:

|                | Estimate | Std. Error | z value | Pr(>|z|) |
|----------------|----------|------------|---------|----------|
| (Intercept)    | -0.0509  | 0.5164     | -0.10   | 0.9215   |
| age            | 0.0100   | 0.0129     | 0.77    | 0.4399   |
| substancecocaine | -0.7496 | 0.2303     | -3.25   | 0.0011   |
| substanceheroin | -0.7780 | 0.2469     | -3.15   | 0.0016   |

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 625.28 on 452 degrees of freedom
Residual deviance: 607.62 on 449 degrees of freedom
AIC: 615.6

Number of Fisher Scoring iterations: 4

```r
> exp(coef(mymodsubage))
```

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>0.950</td>
<td>1.010</td>
<td>0.473</td>
<td>0.459</td>
</tr>
<tr>
<td>age</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>substancecocaine</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>substanceheroin</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```r
> anova(mymoddage, mymodsubage, test="Chisq")
```

Analysis of Deviance Table

Model 1: (homeless == "homeless") ~ age
Model 2: (homeless == "homeless") ~ age + substance

<table>
<thead>
<tr>
<th>Resid. Df</th>
<th>Resid. Dev</th>
<th>Df</th>
<th>Deviance</th>
<th>Pr(&gt;Chi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>622</td>
<td>2</td>
<td>14.3</td>
<td>0.00078</td>
</tr>
<tr>
<td>2</td>
<td>608</td>
<td>2</td>
<td>14.3</td>
<td>0.00078</td>
</tr>
</tbody>
</table>
We observe that the cocaine and heroin groups are significantly less likely to be homeless than alcohol involved subjects, after controlling for age. (A similar result is seen when considering just homeless status and substance.)

```r
> tally(~ homeless | substance, format="percent", margins=TRUE, data=HELPrct)

<table>
<thead>
<tr>
<th>substance</th>
<th>alcohol</th>
<th>cocaine</th>
<th>heroin</th>
</tr>
</thead>
<tbody>
<tr>
<td>homeless</td>
<td>58.2</td>
<td>38.8</td>
<td>37.9</td>
</tr>
<tr>
<td>housed</td>
<td>41.8</td>
<td>61.2</td>
<td>62.1</td>
</tr>
<tr>
<td>Total</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
</tr>
</tbody>
</table>
```
9 Survival Time Outcomes

Extensive support for survival (time to event) analysis is available within the survival package.

9.1 Kaplan-Meier plot

```r
> require(survival)
> fit <- survfit(Surv(dayslink, linkstatus) ~ treat,
    data=HELPct)
> plot(fit, conf.int=FALSE, lty=1:2, lwd=2,
    xlab="time (in days)", ylab="P(not linked)"
> legend(20, 0.4, legend=c("Control", "Treatment"),
    lty=c(1,2), lwd=2)
> title("Product-Limit Survival Estimates (time to linkage)"
```

![Product-Limit Survival Estimates (time to linkage)](image)
We see that the subjects in the treatment (Health Evaluation and Linkage to Primary Care clinic) were significantly more likely to link to primary care (less likely to “survive”) than the control (usual care) group.

### 9.2 Cox proportional hazards model

```r
> require(survival)
> summary(coxph(Surv(dayslink, linkstatus) ~ age + substance,
    data=HELPrct))
```

Call:

```r
coxph(formula = Surv(dayslink, linkstatus) ~ age + substance,
    data = HELPrct)
```

n= 431, number of events= 163
(22 observations deleted due to missingness)

| Variable      | Coef  | Exp(coef) | SE(coef) | z     | Pr(>|z|) |
|---------------|-------|-----------|----------|-------|----------|
| age           | 0.00893 | 1.009     | 0.01026  | 0.87  | 0.38     |
| substancecocaine | 0.18045  | 1.198     | 0.18100  | 1.00  | 0.32     |
| substanceheroin | -0.28970 | 0.748     | 0.21725  | -1.33 | 0.18     |

```

<table>
<thead>
<tr>
<th>exp(coef)</th>
<th>exp(-coef)</th>
<th>lower .95</th>
<th>upper .95</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>1.009</td>
<td>0.991</td>
<td>0.989</td>
</tr>
<tr>
<td>substancecocaine</td>
<td>1.198</td>
<td>0.835</td>
<td>0.840</td>
</tr>
<tr>
<td>substanceheroin</td>
<td>0.748</td>
<td>1.336</td>
<td>0.489</td>
</tr>
</tbody>
</table>

Concordance= 0.55 (se = 0.023)
Rsquare= 0.014 (max possible= 0.988)
Likelihood ratio test= 6.11 on 3 df, p=0.106
Wald test = 5.84 on 3 df, p=0.12
Score (logrank) test = 5.91 on 3 df, p=0.116

Neither age nor substance group was significantly associated with linkage to primary care.
10
More than Two Variables

10.1 Two (or more) way ANOVA

We can fit a two (or more) way ANOVA model, without or with an interaction, using the same modeling syntax.

\[
\text{median}(\text{cesd} \sim \text{substance} \mid \text{sex}, \text{data}=\text{HELPrct})
\]

<table>
<thead>
<tr>
<th>alcohol.female</th>
<th>cocaine.female</th>
<th>heroin.female</th>
<th>alcohol.male</th>
</tr>
</thead>
<tbody>
<tr>
<td>40.0</td>
<td>35.0</td>
<td>39.0</td>
<td>33.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cocaine.male</th>
<th>heroin.male</th>
<th>female</th>
<th>male</th>
</tr>
</thead>
<tbody>
<tr>
<td>29.0</td>
<td>34.5</td>
<td>38.0</td>
<td>32.5</td>
</tr>
</tbody>
</table>

\[
\text{bwplot}(\text{cesd} \sim \text{subgrp} \mid \text{sex}, \text{data}=\text{HELPrct})
\]

\[
\text{summary}(\text{aov}(\text{cesd} \sim \text{substance} + \text{sex}, \text{data}=\text{HELPrct}))
\]

<table>
<thead>
<tr>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>substance</td>
<td>2</td>
<td>2704.0</td>
<td>1352.0</td>
<td>9.27</td>
</tr>
</tbody>
</table>
> summary(aov(cesd ~ substance * sex, data=HELPrct))

<table>
<thead>
<tr>
<th></th>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>substance</td>
<td>2</td>
<td>2704</td>
<td>1352</td>
<td>9.25</td>
<td>0.00012</td>
</tr>
<tr>
<td>sex</td>
<td>1</td>
<td>2569</td>
<td>2569</td>
<td>17.57</td>
<td>3.3e-05</td>
</tr>
<tr>
<td>substance:sex</td>
<td>2</td>
<td>146</td>
<td>73</td>
<td>0.50</td>
<td>0.60752</td>
</tr>
<tr>
<td>Residuals</td>
<td>447</td>
<td>65369</td>
<td>146</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

There’s little evidence for the interaction, though there are statistically significant main effects terms for substance group and sex.

> xyplot(cesd ~ substance, groups=sex, lwd=2, auto.key=list(columns=2, lines=TRUE, points=FALSE), type='a', data=HELPrct)

10.2 Multiple regression

Multiple regression is a logical extension of the prior commands, where additional predictors are added. This allows students to start to try to disentangle multivariate relationships.

Here we consider a model (parallel slopes) for depressive symptoms as a function of Mental Component Score (MCS), age (in years) and sex of the subject.

We tend to introduce multiple linear regression early in our courses, as a purely descriptive technique, then return to it regularly. The motivation for this is described at length in the companion volume Start Modeling with R.
> lmnointeract <- lm(cesd ~ mcs + age + sex, data=HELPrct)
> msummary(lmnointeract)

|            | Estimate | Std. Error | t value | Pr(>|t|) |
|------------|----------|------------|---------|----------|
| (Intercept)| 53.8303  | 2.3617     | 22.79   | <2e-16   |
| mcs        | -0.6548  | 0.0336     | -19.50  | <2e-16   |
| age        | 0.0553   | 0.0556     | 1.00    | 0.3200   |
| sexmale    | -2.8993  | 1.0137     | -2.86   | 0.0044   |

Residual standard error: 9.09 on 449 degrees of freedom
Multiple R-squared: 0.476, Adjusted R-squared: 0.473
F-statistic: 136 on 3 and 449 DF, p-value: <2e-16

We can also fit a model that includes an interaction between MCS and sex.

> lminteract <- lm(cesd ~ mcs + age + sex + mcs:sex, data=HELPrct)
> msummary(lminteract)

|            | Estimate | Std. Error | t value | Pr(>|t|) |
|------------|----------|------------|---------|----------|
| (Intercept)| 55.3906  | 2.9903     | 18.52   | <2e-16   |
| mcs        | -0.7082  | 0.0712     | -9.95   | <2e-16   |
| age        | 0.0549   | 0.0556     | 0.99    | 0.324    |
| sexmale    | -4.9421  | 2.6055     | -1.90   | 0.058    |
| mcs:sexmale| 0.0687   | 0.0807     | 0.85    | 0.395    |

Residual standard error: 9.09 on 448 degrees of freedom
Multiple R-squared: 0.477, Adjusted R-squared: 0.472
F-statistic: 102 on 4 and 448 DF, p-value: <2e-16

> anova(lminteract)

Analysis of Variance Table

Response: cesd

<table>
<thead>
<tr>
<th></th>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>mcs</td>
<td>1</td>
<td>32918</td>
<td>32918</td>
<td>398.27</td>
<td>&lt;2e-16</td>
</tr>
<tr>
<td>age</td>
<td>1</td>
<td>107</td>
<td>107</td>
<td>1.29</td>
<td>0.2563</td>
</tr>
<tr>
<td>sex</td>
<td>1</td>
<td>676</td>
<td>676</td>
<td>8.18</td>
<td>0.0044</td>
</tr>
<tr>
<td>mcs:sex</td>
<td>1</td>
<td>60</td>
<td>60</td>
<td>0.72</td>
<td>0.3952</td>
</tr>
<tr>
<td>Residuals</td>
<td>448</td>
<td>37028</td>
<td>83</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

> anova(lmnointeract, lminteract)

Analysis of Variance Table
Model 1: cesd ~ mcs + age + sex
Model 2: cesd ~ mcs + age + sex + mcs:sex

<table>
<thead>
<tr>
<th>Res.Df</th>
<th>RSS</th>
<th>Df</th>
<th>Sum of Sq</th>
<th>F</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>449</td>
<td>37088</td>
<td></td>
<td>59.9</td>
<td>0.4</td>
</tr>
<tr>
<td>2</td>
<td>448</td>
<td>37028</td>
<td>1</td>
<td>0.72</td>
<td>0.4</td>
</tr>
</tbody>
</table>

There is little evidence for an interaction effect, so we drop this from the model.

10.2.1 Visualizing the results from the regression

The `makeFun()` and `plotFun()` functions from the `mosaic` package can be used to display the predicted values from a regression model. For this example, we might display the predicted CESD values for a range of MCS (mental component score) values a hypothetical 36 year old male and female subject might have from the parallel slopes (no interaction) model.

```r
> lmfunction <- makeFun(lmnointeract)
```

We can now plot the predicted values separately for male and female subjects over a range of MCS (mental component score) values, along with the observed data for all of the 36 year olds.

```r
> xypplot(cesd ~ mcs, groups=sex, auto.key=TRUE, data=filter(HELPrct, age==36))
> plotFun(lmfunction(mcs, age=36, sex="male") ~ mcs, xlim=c(0, 60), lwd=2, ylab="predicted CESD", add=TRUE)
> plotFun(lmfunction(mcs, age=36, sex="female") ~ mcs, xlim=c(0, 60), lty=2, lwd=3, add=TRUE)
```
10.2.2 Coefficient plots

It is sometimes useful to display a plot of the coefficients for a multiple regression model (along with their associated confidence intervals).

```r
> mplot(lmnointeract, rows=-1, which=7)
```

Darker dots indicate regression coefficients where the 95% confidence interval does not include the null hypothesis value of zero.

Caution!

Be careful when fitting regression models with missing values (see also section 13.11).

10.2.3 Residual diagnostics

It’s straightforward to undertake residual diagnostics for this model. We begin by adding the fitted values and residuals to the dataset.

The `mplot()` function can also be used to create these graphs.

Here we are adding two new variables into an existing dataset. It’s often a good practice to give the resulting dataframe a new name.
> HELPrct <- mutate(HELPrct, 
  residuals = resid(lmnointeract), 
  pred = fitted(lmnointeract))

> histogram(~ residuals, xlab="residuals", fit="normal", 
  data=HELPrct)

We can identify the subset of observations with extremely large residuals.

> filter(HELPrct, abs(residuals) > 25)

<table>
<thead>
<tr>
<th>age</th>
<th>anystatus</th>
<th>anysub</th>
<th>cesd</th>
<th>d1</th>
<th>daysanysub</th>
<th>dayslink</th>
<th>drugrisk</th>
<th>e2b</th>
<th>female</th>
<th>sex</th>
<th>g1b</th>
<th>homeless</th>
<th>i1</th>
<th>i2</th>
<th>indtot</th>
<th>linkstatus</th>
<th>link</th>
<th>mcs</th>
<th>pcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>43</td>
<td>0</td>
<td>no</td>
<td>16</td>
<td>15</td>
<td>191</td>
<td>414</td>
<td>0</td>
<td>NA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>27</td>
<td>NA</td>
<td>&lt;NA&gt;</td>
<td>40</td>
<td>1</td>
<td>365</td>
<td>3</td>
<td>2</td>
<td>NA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sexrisk</th>
<th>satreat</th>
<th>sexrisk</th>
<th>substance</th>
<th>treat</th>
<th>subgrp</th>
<th>residuals</th>
</tr>
</thead>
<tbody>
<tr>
<td>white</td>
<td>no</td>
<td>7</td>
<td>cocaine</td>
<td>yes</td>
<td>C</td>
<td>-26.9</td>
</tr>
<tr>
<td>white</td>
<td>yes</td>
<td>3</td>
<td>heroin</td>
<td>no</td>
<td>H</td>
<td>25.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>pred</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>42.9</td>
</tr>
<tr>
<td>2</td>
<td>14.8</td>
</tr>
</tbody>
</table>

> xyplot(residuals ~ pred, ylab="residuals", cex=0.3, 
  xlab="predicted values", main="predicted vs. residuals", 
  type=c("p", "r", "smooth"), data=HELPrct)
The assumptions of normality, linearity and homoscedasticity seem reasonable here.
R can calculate quantities related to probability distributions of all types. It is straightforward to generate random samples from these distributions, which can be used for simulation and exploration.

\[
> \text{xpnorm}(1.96, \text{mean}=0, \text{sd}=1) \quad \# P(Z < 1.96)
\]

If \(X \sim \text{N}(0, 1)\), then

\[
\begin{align*}
P(X \leq 1.96) &= P(Z \leq 1.96) = 0.975 \\
P(X > 1.96) &= P(Z > 1.96) = 0.025
\end{align*}
\]

[1] 0.975
A similar display is available for the F distribution.

> xpf(3, df1=4, df2=20)

[1] 0.957

The following table displays the basenames for probability distributions available within base R. These functions can be prefixed by d to find the density function for the distribution, p to find the cumulative distribution function, q to find quantiles, and r to generate random draws. For example, to find the density function of an exponential random variable, use the command dexp(). The qDIST() function is the inverse of the pDIST() function, for a given basename DIST.
<table>
<thead>
<tr>
<th>Distribution</th>
<th>Basename</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beta</td>
<td>beta</td>
</tr>
<tr>
<td>binomial</td>
<td>binom</td>
</tr>
<tr>
<td>Cauchy</td>
<td>cauchy</td>
</tr>
<tr>
<td>chi-square</td>
<td>chisq</td>
</tr>
<tr>
<td>exponential</td>
<td>exp</td>
</tr>
<tr>
<td>F</td>
<td>f</td>
</tr>
<tr>
<td>gamma</td>
<td>gamma</td>
</tr>
<tr>
<td>geometric</td>
<td>geom</td>
</tr>
<tr>
<td>hypergeometric</td>
<td>hyper</td>
</tr>
<tr>
<td>logistic</td>
<td>logis</td>
</tr>
<tr>
<td>lognormal</td>
<td>lnorm</td>
</tr>
<tr>
<td>negative binomial</td>
<td>nbinom</td>
</tr>
<tr>
<td>normal</td>
<td>norm</td>
</tr>
<tr>
<td>Poisson</td>
<td>pois</td>
</tr>
<tr>
<td>Student’s t</td>
<td>t</td>
</tr>
<tr>
<td>Uniform</td>
<td>unif</td>
</tr>
<tr>
<td>Weibull</td>
<td>weibull</td>
</tr>
</tbody>
</table>

The `plotDist()` can be used to display distributions in a variety of ways.

```r
> plotDist('norm', mean=100, sd=10, kind='cdf')

> plotDist('exp', kind='histogram', xlab="x")
```

**Digging Deeper**

The `fitdistr()` within the MASS package facilitates estimation of parameters for many distributions.
Multiple distributions can be plotted on the same plot.

```r
> plotDist('binom', size=25, prob=0.25, xlim=c(-1,26))
```

```r
> plotDist("binom", size=100, prob=.3, col="black", lwd=3, pch=16)
> plotDist("norm", mean=30, sd=sqrt(100 * .3 * .7),
  groups = abs(x - 30) > 6 , type="h", under=TRUE)
```
The `plotFun()` function can be used to plot an arbitrary function (in this case an exponential random variable).

```r
> f <- makeFun(2*exp(-2*x) ~ x)  # exponential with rate parameter 2
> x <- rexp(1000, rate=2)
> histogram(~ x, width=0.2, center=0.1)
> plotFun(f(x) ~ x, col="red", lwd=2, add=TRUE)
```
12

Power Calculations

While not generally a major topic in introductory courses, power and sample size calculations help to reinforce key ideas in statistics. In this section, we will explore how R can be used to undertake power calculations using analytic approaches. We consider a simple problem with two tests (t-test and sign test) of a one-sided comparison.

We will compare the power of the sign test and the power of the test based on normal theory (one sample one sided t-test) assuming that $\sigma$ is known. Let $X_1, \ldots, X_{25}$ be i.i.d. $N(0.3,1)$ (this is the alternate that we wish to calculate power for). Consider testing the null hypothesis $H_0: \mu = 0$ versus $H_A: \mu > 0$ at significance level $\alpha = .05$.

### 12.1 Sign test

We start by calculating the Type I error rate for the sign test. Here we want to reject when the number of positive values is large. Under the null hypothesis, this is distributed as a Binomial random variable with $n=25$ trials and $p=0.5$ probability of being a positive value. Let’s consider values between 15 and 19.

```r
> xvals <- 15:19
> probs <- 1 - pbinom(xvals, size=25, prob=0.5)
> cbind(xvals, probs)

  xvals  probs
 [1,] 15 0.11476
 [2,] 16 0.05388
 [3,] 17 0.02164
 [4,] 18 0.00732
 [5,] 19 0.00204
```
> `qbinom(.95, size=25, prob=0.5)`

[1] 17

So we see that if we decide to reject when the number of positive values is 17 or larger, we will have an $\alpha$ level of 0.054, which is near the nominal value in the problem.

We calculate the power of the sign test as follows. The probability that $X_i > 0$, given that $H_A$ is true is given by:

> `1 - pnorm(0, mean=0.3, sd=1)`

[1] 0.618

We can view this graphically using the command:

> `xpnorm(0, mean=0.3, sd=1, lower.tail=FALSE)`

If $X \sim N(0.3,1)$, then

\[
P(X \leq 0) = P(Z \leq -0.3) = 0.3821 \\
P(X > 0) = P(Z > -0.3) = 0.6179
\]

[1] 0.618

The power under the alternative is equal to the probability of getting 17 or more positive values, given that $p = 0.6179$:

> `1 - pbinom(16, size=25, prob=0.6179)`

[1] 0.338

The power is modest at best.
12.2 \textit{T-test}

We next calculate the power of the test based on normal theory. To keep the comparison fair, we will set our $\alpha$ level equal to 0.05388.

\begin{verbatim}
> alpha <- 1 - pb<binom(16, size=25, prob=0.5); alpha
[1] 0.0539

First we find the rejection region.
\end{verbatim}

\begin{verbatim}
> n <- 25; sigma <- 1 # given
> stderr <- sigma/sqrt(n)
> zstar <- qnorm(1-alpha, mean=0, sd=1)
> zstar
[1] 1.61

> crit <- zstar*stderr
> crit
[1] 0.322
\end{verbatim}

Therefore, we reject for observed means greater than 0.322.

To calculate the power of this one-sided test we find the probability under the alternative hypothesis to the right of this cutoff.

\begin{verbatim}
> power <- 1 - pnorm(crit, mean=0.3, sd=stderr)
> power
[1] 0.457
\end{verbatim}

The power of the test based on normal theory is 0.457. To provide a check (or for future calculations of this sort) we can use the \texttt{power.t.test()} function.

\begin{verbatim}
> power.t.test(n=25, delta=.3, sd=1, sig.level=alpha, alternative="one.sided",
   type="one.sample")$power
[1] 0.441
\end{verbatim}
This analytic (formula-based approach) yields a similar estimate to the value that we calculated directly. Overall, we see that the t-test has higher power than the sign test, if the underlying data are truly normal.

Calculating power empirically demonstrates the power of simulations.
13 Data Management

Data management is a key capacity to allow students (and instructors) to “compute with data” or as Diane Lambert of Google has stated, “think with data”. We tend to keep student data management to a minimum during the early part of an introductory statistics course, then gradually introduce topics as needed. For courses where students undertake substantive projects, data management is more important. This chapter describes some key data management tasks.

13.1 Inspecting dataframes

The `inspect()` function can be helpful in describing the variables in a dataframe (the name for a dataset in R).

```r
> inspect(iris)

categorical variables:
  name  class  levels     n         missing
1   Species factor  3   150 150

distribution
1  setosa (33.3%), versicolor (33.3%) ...

quantitative variables:
  name  class  min    Q1 median    Q3 max    mean   sd     n missing
1 Sepal.Length numeric 4.3  5.1  5.80  6.4 7.9  5.84 0.828 150 0
2  Sepal.Width numeric 2.0  2.8  3.00  3.3 4.4  3.06 0.436 150 0
3 Petal.Length numeric 1.0  1.6  4.35  5.1 6.9  3.76 1.765 150 0
4  Petal.Width numeric 0.1  0.3 1.30  1.8 2.5  1.20 0.762 150 0

The iris dataframe includes one categorical and four quantitative variables.
13.2 Adding new variables to a dataframe

We can add additional variables to an existing dataframe using `mutate()`. But first we create a smaller version of the `iris` dataframe.

```r
> irisSmall <- select(iris, Species, Sepal.Length)

> # cut places data into bins
> irisSmall <- mutate(irisSmall,
                      Length = cut(Sepal.Length, breaks=4:8))
```

Multiple commands can be chained together using the `%>%` (pipe) operator:

```r
> irisSmall <- iris %>%
              select(Species, Sepal.Length) %>%
              mutate(Length = cut(Sepal.Length, breaks=4:8))
```

Note that in this usage the first argument to `select()` is the first variable (as it inherits the data from the previous pipe).

```r
> head(irisSmall)
```

<table>
<thead>
<tr>
<th>Species</th>
<th>Sepal.Length</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>setosa</td>
<td>5.1</td>
<td>(5,6]</td>
</tr>
<tr>
<td>setosa</td>
<td>4.9</td>
<td>(4,5]</td>
</tr>
<tr>
<td>setosa</td>
<td>4.7</td>
<td>(4,5]</td>
</tr>
<tr>
<td>setosa</td>
<td>4.6</td>
<td>(4,5]</td>
</tr>
<tr>
<td>setosa</td>
<td>5.0</td>
<td>(4,5]</td>
</tr>
<tr>
<td>setosa</td>
<td>5.4</td>
<td>(5,6]</td>
</tr>
</tbody>
</table>

The `cut()` function has an option `labels` which can be used to specify more descriptive names for the groups.

The `CPS85` dataframe contains data from a Current Population Survey (current in 1985, that is). Two of the variables in this dataframe are `age` and `educ`. We can estimate the number of years a worker has been in the workforce if we assume they have been in the workforce since completing their education and that their age at graduation is 6 more than the number of years of education obtained. We can add this as a new variable in the dataframe using `mutate()`. 

...
> CPS85 <- mutate(CPS85, workforce.years = age - 6 - educ)
> favstats(~ workforce.years, data=CPS85)

    min Q1 median Q3 max mean   sd  n missing
-4   8  15  26  55 17.8 12.4 534      0

In fact this is what was done for all but one of the cases to create the exper variable that is already in the CPS85 data.

> tally(~ (exper - workforce.years), data=CPS85)

   0  4  533  1

### 13.3 Dropping variables

Since we already have the exper variable, there is no reason to keep our new variable. Let’s drop it. Notice the clever use of the minus sign.

> names(CPS85)

[1] "wage" "educ" "race" "sex" "hispanic" "south" "married" "exper" "union" "age" "sector" "workforce.years"

> CPS1 <- select(CPS85, select = -matches("workforce.years"))
> names(CPS1)

[1] "wage" "educ" "race" "sex" "hispanic" "south" "married" "exper" "union" "age" "sector"

Any number of variables can be dropped or kept in a similar manner.

> CPS1 <- select(CPS85, select = -matches("workforce.years|exper"))
13.4 Renaming variables

The column (variable) names for a dataframe can be changed using the `rename()` function in the `dplyr` package.

```r
> names(CPS85)
[1] "wage"     "educ"     "race"     "sex"      "hispanic"  "south"
[4] "married"  "exper"    "union"    "age"      "sector"    "workforce.years"

> CPSnew = rename(CPS85, workforce=workforce.years)
> names(CPSnew)
[1] "wage" "educ"  "race"  "sex"   "hispanic"  "south"
[6] "married" "exper" "union" "age"    "sector"    "workforce"
```

The row names of a dataframe can be changed by simple assignment using `row.names()`.

The faithful data set (in the datasets package, which is always available) has very unfortunate names.

```r
> names(faithful)
[1] "eruptions" "waiting"
```

It’s a good idea to establish practices for choice of variable names from day one.

The measurements are the duration of an eruption and the time until the subsequent eruption, so let’s give it some better names.

```r
> faithful <- rename(faithful, 
duration = eruptions, 
time.til.next=waiting)
> names(faithful)
[1] "duration"  "time.til.next"
```
If the variable containing a dataframe is modified or used to store a different object, the original data from the package can be recovered using `data()`.

```r
> data(faithful)
> head(faithful, 3)

  eruptions waiting
1   3.60       79
2   1.80       54
3   3.33       74
```

### 13.5 Creating subsets of observations

We can also use `filter()` to reduce the size of a dataframe by selecting only certain rows.

```r
> data(faithful)
> names(faithful) <- c('duration', 'time.til.next')
> # any logical can be used to create subsets
> faithfulLong <- filter(faithful, duration > 3)
> xyplot( time.til.next ~ duration, data=faithfulLong )
```
### 13.6 Sorting dataframes

Data frames can be sorted using the `arrange()` function.

```r
> head(faithful, 3)
  duration time.til.next
1    3.60         79
2    1.80         54
3    3.33         74

> sorted <- arrange(faithful, duration)
> head(sorted, 3)
  duration time.til.next
1    1.60         52
2    1.67         64
3    1.70         59
```

**Caution!** It is usually better to make new datasets rather than modifying the original.

### 13.7 Merging datasets

The fusion1 dataframe in the fastR package contains genotype information for a SNP (single nucleotide polymorphism) in the gene TCF7L2. The pheno dataframe contains phenotypes (including type 2 diabetes case/control status) for an intersecting set of individuals. We can join
(or merge) these together to explore the association between genotypes and phenotypes using `merge()`.

```r
> require(fastR)
> require(dplyr)
> fusion1 <- arrange(fusion1, id)
> head(fusion1, 3)

<table>
<thead>
<tr>
<th>id</th>
<th>marker</th>
<th>markerID</th>
<th>allele1</th>
<th>allele2</th>
<th>genotype</th>
<th>Adose</th>
<th>Cdose</th>
<th>Gdose</th>
<th>Tdose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1002</td>
<td>RS12255372</td>
<td>1</td>
<td>3</td>
<td>GG</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1009</td>
<td>RS12255372</td>
<td>1</td>
<td>3</td>
<td>GG</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1012</td>
<td>RS12255372</td>
<td>1</td>
<td>3</td>
<td>GG</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

> head(pheno, 3)

<table>
<thead>
<tr>
<th>id</th>
<th>t2d</th>
<th>bmi</th>
<th>sex</th>
<th>age</th>
<th>smoker</th>
<th>chol</th>
<th>waist</th>
<th>weight</th>
<th>height</th>
<th>whr</th>
<th>sbp</th>
<th>dbp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>case</td>
<td>32.9</td>
<td>F</td>
<td>70.8</td>
<td>former</td>
<td>4.57</td>
<td>112.0</td>
<td>85.6</td>
<td>161</td>
<td>0.987</td>
<td>135</td>
<td>77</td>
</tr>
<tr>
<td>2</td>
<td>case</td>
<td>27.4</td>
<td>F</td>
<td>53.9</td>
<td>never</td>
<td>7.32</td>
<td>93.5</td>
<td>77.4</td>
<td>168</td>
<td>0.940</td>
<td>158</td>
<td>88</td>
</tr>
<tr>
<td>3</td>
<td>control</td>
<td>30.5</td>
<td>M</td>
<td>53.9</td>
<td>former</td>
<td>5.02</td>
<td>104.0</td>
<td>94.6</td>
<td>176</td>
<td>0.933</td>
<td>143</td>
<td>89</td>
</tr>
</tbody>
</table>

> require(tidyr)
> fusion1m <- inner_join(fusion1, pheno, by='id')
> head(fusion1m, 3)

<table>
<thead>
<tr>
<th>id</th>
<th>marker</th>
<th>markerID</th>
<th>allele1</th>
<th>allele2</th>
<th>genotype</th>
<th>Adose</th>
<th>Cdose</th>
<th>Gdose</th>
<th>Tdose</th>
<th>t2d</th>
<th>bmi</th>
<th>sex</th>
<th>age</th>
<th>smoker</th>
<th>chol</th>
<th>waist</th>
<th>weight</th>
<th>height</th>
<th>whr</th>
<th>sbp</th>
<th>dbp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1002</td>
<td>RS12255372</td>
<td>1</td>
<td>3</td>
<td>GG</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>case</td>
<td>32.9</td>
<td>F</td>
<td>70.8</td>
<td>former</td>
<td>4.57</td>
<td>112.0</td>
<td>85.6</td>
<td>161</td>
<td>0.987</td>
<td>135</td>
<td>77</td>
</tr>
<tr>
<td>2</td>
<td>1009</td>
<td>RS12255372</td>
<td>1</td>
<td>3</td>
<td>GG</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>case</td>
<td>27.4</td>
<td>F</td>
<td>53.9</td>
<td>never</td>
<td>7.32</td>
<td>93.5</td>
<td>77.4</td>
<td>168</td>
<td>0.940</td>
<td>158</td>
<td>88</td>
</tr>
<tr>
<td>3</td>
<td>1012</td>
<td>RS12255372</td>
<td>1</td>
<td>3</td>
<td>GG</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>control</td>
<td>30.5</td>
<td>M</td>
<td>53.9</td>
<td>former</td>
<td>5.02</td>
<td>104.0</td>
<td>94.6</td>
<td>176</td>
<td>0.933</td>
<td>143</td>
<td>89</td>
</tr>
</tbody>
</table>

Now we are ready to begin our analysis.

```r
tally(~t2d + genotype, data=fusion1m)
```
13.8 Slicing and dicing

The tidyr package provides a flexible way to change the arrangement of data. It was designed for converting between long and wide versions of time series data and its arguments are named with that in mind.

A common situation is when we want to convert from a wide form to a long form because of a change in perspective about what a unit of observation is. For example, in the traffic dataframe, each row is a year, and data for multiple states are provided.

```
> traffic
   year cn.deaths ny cn ma ri
1 1951 265 13.9 13.0 10.2 8.0
2 1952 230 13.8 10.8 10.0 8.5
3 1953 275 14.4 12.8 11.0 8.5
4 1954 240 13.0 10.8 10.5 7.5
5 1955 325 13.5 14.0 11.8 10.0
6 1956 280 13.4 12.1 11.0 8.2
7 1957 273 13.3 11.9 10.2 9.4
8 1958 248 13.0 10.1 11.8 8.6
9 1959 245 12.9 10.0 11.0 9.0
```

We can reformat this so that each row contains a measurement for a single state in one year.

```
> longTraffic <- traffic %>%
  gather(state, deathRate, ny:ri)
>
> head(longTraffic)
   year cn.deaths state deathRate
1 1951   265    ny      13.9
2 1952   230    ny      13.8
3 1953   275    ny      14.4
4 1954   240    ny      13.0
5 1955   325    ny      13.5
6 1956   280    ny      13.4
```

We can also reformat the other way, this time having all data for a given state form a row in the dataframe.
> stateTraffic <- longTraffic %>%  
  select(year, deathRate, state) %>%  
  mutate(year=paste("deathRate.", year, sep="")) %>%  
  spread(year, deathRate)  
> stateTraffic

<table>
<thead>
<tr>
<th></th>
<th>deathRate.1951</th>
<th>deathRate.1952</th>
<th>deathRate.1953</th>
<th>deathRate.1954</th>
<th>deathRate.1955</th>
<th>deathRate.1956</th>
<th>deathRate.1957</th>
<th>deathRate.1958</th>
<th>deathRate.1959</th>
</tr>
</thead>
<tbody>
<tr>
<td>ny</td>
<td>13.9</td>
<td>13.8</td>
<td>14.4</td>
<td>13.0</td>
<td>13.5</td>
<td>13.4</td>
<td>13.3</td>
<td>13.0</td>
<td>12.9</td>
</tr>
<tr>
<td>cn</td>
<td>13.0</td>
<td>10.8</td>
<td>12.8</td>
<td>10.8</td>
<td>14.0</td>
<td>12.1</td>
<td>11.9</td>
<td>10.1</td>
<td>10.0</td>
</tr>
<tr>
<td>ma</td>
<td>10.2</td>
<td>10.0</td>
<td>11.0</td>
<td>10.5</td>
<td>11.8</td>
<td>11.0</td>
<td>10.2</td>
<td>11.8</td>
<td>11.0</td>
</tr>
<tr>
<td>ri</td>
<td>8.0</td>
<td>8.5</td>
<td>8.5</td>
<td>7.5</td>
<td>10.0</td>
<td>8.2</td>
<td>9.4</td>
<td>8.6</td>
<td>9.0</td>
</tr>
</tbody>
</table>

13.9 Derived variable creation

A number of functions help facilitate the creation or re-coding of variables.

13.9.1 Creating categorical variable from a quantitative variable

Next we demonstrate how to create a three-level categorical variable with cuts at 20 and 40 for the CESD scale (which ranges from 0 to 60 points).

> favstats(~ cesd, data=HELPrct)

<table>
<thead>
<tr>
<th></th>
<th>cesd, data=HELPrct</th>
</tr>
</thead>
<tbody>
<tr>
<td>min</td>
<td>1</td>
</tr>
<tr>
<td>Q1</td>
<td>25</td>
</tr>
<tr>
<td>median</td>
<td>34</td>
</tr>
<tr>
<td>Q3</td>
<td>41</td>
</tr>
<tr>
<td>max</td>
<td>60</td>
</tr>
<tr>
<td>mean</td>
<td>32.8</td>
</tr>
<tr>
<td>sd</td>
<td>12.5</td>
</tr>
<tr>
<td>n</td>
<td>453</td>
</tr>
<tr>
<td>missing</td>
<td>0</td>
</tr>
</tbody>
</table>

> HELPrct <- mutate(HELPrct, cesdcut = cut(cesd, breaks=c(0, 20, 40, 60), include.lowest=TRUE))  
> bwplot(cesd ~ cesdcut, data=HELPrct)
It might be preferable to give better labels.

```r
> HELPrct <- mutate(HELPrct, cesdcut = cut(cesd, 
labels=c("low", "medium", "high"), 
breaks=c(0, 20, 40, 60), include.lowest=TRUE))
> bwplot(cesd ~ cesdcut, pch="|", data=HELPrct)
```

The `ntiles()` function can be used to automate creation of groups in this manner.

The `derivedFactor()` function is even more general and can also be used for this purpose.

```r
> HELPrct <- mutate(HELPrct, 
anothercut = derivedFactor( 
  low = cesd >= 0 & cesd <= 20, 
  medium = cesd > 20 & cesd <= 40, 
  high = cesd > 40))
```
13.9.2 Reordering factors

By default R uses the first level in lexicographic order as the reference group for modeling. This can be overridden using the `relevel()` function (see also `reorder()`).

```r
> tally(~ substance, data=HELPct)

alcohol cocaine heroin
   177    152    124
```

```r
> coef(lm(cesd ~ substance, data=HELPct))

(Intercept) substancecocaine substanceheroin
     34.373     -4.952      0.498
```

```r
> HELPrct <- mutate(HELPct, subnew = relevel(substance, ref="heroin"))
> coef(lm(cesd ~ subnew, data=HELPrct))

(Intercept) subnewalcohol subnewcocaine
     34.871     -0.498     -5.450
```

13.10 Group-wise statistics

It can often be useful to calculate summary statistics by group, and add these into a dataset. The `group_by()` function in the `dplyr` package facilitates this process. Here we demonstrate how to add a variable containing the median age of subjects by substance group.

```r
> favstats(age ~ substance, data=HELPct)

substance min Q1 median Q3 max mean sd n missing
 1   alcohol  33 38.0 38.0 43.0 58 38.2 7.65  177    0
 2   cocaine  23 30.0 33.5 37.2 60 34.5 6.69  152    0
 3   heroin   19 27.0 33.0 39.0 55 33.4 7.99  124    0
```

```r
> ageGroup <- HELPct %>%
  group_by(substance) %>%
  summarise(agebygroup = mean(age))
> ageGroup
```
Source: local data frame [3 x 2]

<table>
<thead>
<tr>
<th>substance</th>
<th>agebygroup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(fctr)</td>
</tr>
<tr>
<td>alcohol</td>
<td>38.2</td>
</tr>
<tr>
<td>cocaine</td>
<td>34.5</td>
</tr>
<tr>
<td>heroin</td>
<td>33.4</td>
</tr>
</tbody>
</table>

> nrow(ageGroup)

[1] 3

> nrow(HELPrct)

[1] 453

> HELPmerged <- left_join(ageGroup, HELPrct, by="substance")
> favstats(agebygroup ~ substance, data=HELPmerged)

<table>
<thead>
<tr>
<th>substance</th>
<th>min</th>
<th>Q1</th>
<th>median</th>
<th>Q3</th>
<th>max</th>
<th>mean</th>
<th>sd</th>
<th>n</th>
<th>missing</th>
</tr>
</thead>
<tbody>
<tr>
<td>alcohol</td>
<td>38.2</td>
<td>38.2</td>
<td>38.2</td>
<td>38.2</td>
<td>38.2</td>
<td>38.2</td>
<td>0</td>
<td>177</td>
<td>0</td>
</tr>
<tr>
<td>cocaine</td>
<td>34.5</td>
<td>34.5</td>
<td>34.5</td>
<td>34.5</td>
<td>34.5</td>
<td>34.5</td>
<td>0</td>
<td>152</td>
<td>0</td>
</tr>
<tr>
<td>heroin</td>
<td>33.4</td>
<td>33.4</td>
<td>33.4</td>
<td>33.4</td>
<td>33.4</td>
<td>33.4</td>
<td>0</td>
<td>124</td>
<td>0</td>
</tr>
</tbody>
</table>

> nrow(HELPmerged)

[1] 453

### 13.11 Accounting for missing data

Missing values arise in almost all real world investigations. R uses the NA character as an indicator for missing data. The HELPmiss dataframe within the mosaicData package includes all \( n = 470 \) subjects enrolled at baseline (including the \( n = 17 \) subjects with some missing data who were not included in HELPrct).

> smaller <- select(HELPmiss, cesd, drugrisk, indtot, mcs, pcs, substance)
> dim(smaller)

[1] 470  6

> summary(smaller)
Of the 470 subjects in the 6 variable dataframe, only the drugrisk, indtot, mcs, and pcs variables have missing values.

```r
> favstats(~ mcs, data=smaller)

          min    Q1  median    Q3    max   mean     sd    n missing
cesd 6.76 21.7  28.6  40.6  62.2  31.5  12.8  468       2
drugrisk 1.00 1.00  2.00  4.00  6.00  3.7  1.87  408       1
indtot 2.00 2.00  12.0  34.0  60.0  35.7  7.37  468       1
mcs 2.00 2.00  14.0  32.0  44.0  29.9  5.50  468       1
pcs 6.00 11.0  21.0  36.0  55.0  36.9  9.42  468       1

> with(smaller, sum(is.na(mcs)))
[1] 2

> nomiss <- na.omit(smaller)
> dim(nomiss)
[1] 453 6
> nrow(nomiss)
[1] 453
> ncol(nomiss)
[1] 6
> favstats(~ mcs, data=nomiss)

          min    Q1  median    Q3    max   mean     sd    n missing
cesd 6.76 21.8  28.6  40.9  62.2  31.7  12.8  453       0
```
Alternatively, we could generate the same dataset using logical conditions.

```r
> nomiss <- filter(smaller,
>                  (!is.na(mcs) & !is.na(indtot) & !is.na(drugrisk)))
> dim(nomiss)

[1] 453    6
```
Health Evaluation (HELP) Study

Many of the examples in this guide utilize data from the HELP study, a randomized clinical trial for adult inpatients recruited from a detoxification unit. Patients with no primary care physician were randomized to receive a multidisciplinary assessment and a brief motivational intervention or usual care, with the goal of linking them to primary medical care. Funding for the HELP study was provided by the National Institute on Alcohol Abuse and Alcoholism (R01-AA10870, Samet PI) and National Institute on Drug Abuse (R01-DA10019, Samet PI). The details of the randomized trial along with the results from a series of additional analyses have been published.¹

Eligible subjects were adults, who spoke Spanish or English, reported alcohol, heroin or cocaine as their first or second drug of choice, resided in proximity to the primary care clinic to which they would be referred or were homeless. Patients with established primary care relationships they planned to continue, significant dementia, specific plans to leave the Boston area that would prevent research participation, failure to provide contact information for tracking purposes, or pregnancy were excluded.

Subjects were interviewed at baseline during their detoxification stay and follow-up interviews were undertaken every 6 months for 2 years. A variety of continuous, count, discrete, and survival time predictors and outcomes were collected at each of these five occasions. The Institutional Review Board of Boston University Medical Center approved all aspects of the study, including the creation of the de-identified dataset. Additional privacy protection was secured by the issuance of a Certificate of Confidentiality by the Department of Health and Human

The mosaicData package contains several forms of the de-identified HELP dataset. We will focus on HELPct, which contains 27 variables for the 453 subjects with minimal missing data, primarily at baseline. Variables included in the HELP dataset are described in Table 14.1. More information can be found at: http://www.amherst.edu/~nhorton/r2. A copy of the study instruments can be found at: http://www.amherst.edu/~nhorton/help.

Table 14.1: Annotated description of variables in the HELPct dataset

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>DESCRIPTION (VALUES)</th>
<th>NOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>age at baseline (in years) (range 19–60)</td>
<td></td>
</tr>
<tr>
<td>anysub</td>
<td>use of any substance post-detox</td>
<td>see also daysanysub</td>
</tr>
<tr>
<td>cesd</td>
<td>Center for Epidemiologic Studies Depression scale (range 0–60, higher scores indicate more depressive symptoms)</td>
<td></td>
</tr>
<tr>
<td>d1</td>
<td>how many times hospitalized for medical problems (lifetime) (range 0–100)</td>
<td></td>
</tr>
<tr>
<td>daysanysub</td>
<td>time (in days) to first use of any substance post-detox (range 0–268)</td>
<td>see also anysubstatus</td>
</tr>
<tr>
<td>dayslink</td>
<td>time (in days) to linkage to primary care (range 0–456)</td>
<td>see also linkstatus</td>
</tr>
<tr>
<td>drugrisk</td>
<td>Risk-Assessment Battery (RAB) drug risk score (range 0–21)</td>
<td>see also sexrisk</td>
</tr>
<tr>
<td>e2b</td>
<td>number of times in past 6 months entered a detox program (range 1–21)</td>
<td></td>
</tr>
<tr>
<td>female</td>
<td>gender of respondent (0=male, 1=female)</td>
<td></td>
</tr>
<tr>
<td>g1b</td>
<td>experienced serious thoughts of suicide (last 30 days, values 0=no, 1=yes)</td>
<td></td>
</tr>
<tr>
<td>homeless</td>
<td>1 or more nights on the street or shelter in past 6 months (0=no, 1=yes)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>Notes</td>
</tr>
<tr>
<td>---</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>i1</td>
<td>average number of drinks (standard units) consumed per day (in the past 30 days, range 0–142)</td>
<td>see also i2</td>
</tr>
<tr>
<td>i2</td>
<td>maximum number of drinks (standard units) consumed per day (in the past 30 days range 0–184)</td>
<td>see also i1</td>
</tr>
<tr>
<td>id</td>
<td>random subject identifier (range 1–470)</td>
<td></td>
</tr>
<tr>
<td>indtot</td>
<td>Inventory of Drug Use Consequences (InDUC) total score (range 4–45)</td>
<td></td>
</tr>
<tr>
<td>linkstatus</td>
<td>post-detox linkage to primary care (0=no, 1=yes)</td>
<td>see also dayslink</td>
</tr>
<tr>
<td>mcs</td>
<td>SF-36 Mental Component Score (range 7–62, higher scores are better)</td>
<td>see also pcs</td>
</tr>
<tr>
<td>pcs</td>
<td>SF-36 Physical Component Score (range 14–75, higher scores are better)</td>
<td>see also mcs</td>
</tr>
<tr>
<td>pss_fr</td>
<td>perceived social supports (friends, range 0–14)</td>
<td></td>
</tr>
<tr>
<td>racegrp</td>
<td>race/ethnicity (black, white, hispanic or other)</td>
<td></td>
</tr>
<tr>
<td>satreat</td>
<td>any BSAS substance abuse treatment at baseline (0=no, 1=yes)</td>
<td></td>
</tr>
<tr>
<td>sex</td>
<td>sex of respondent (male or female)</td>
<td></td>
</tr>
<tr>
<td>sexrisk</td>
<td>Risk-Assessment Battery (RAB) sex risk score (range 0–21)</td>
<td>see also drugrisk</td>
</tr>
<tr>
<td>substance</td>
<td>primary substance of abuse (alcohol, cocaine or heroin)</td>
<td></td>
</tr>
<tr>
<td>treat</td>
<td>randomization group (randomize to HELP clinic, no or yes)</td>
<td></td>
</tr>
</tbody>
</table>

Notes: Observed range is provided (at baseline) for continuous variables.
15
Exercises and Problems

The first part of the exercise number indicates which chapter it comes from.

3.1. Using the HELPrc dataset, create side-by-side histograms of the CESD scores by substance abuse group, just for the male subjects, with an overlaid normal density.

5.1. Using the HELPrc dataset, fit a simple linear regression model predicting the number of drinks per day as a function of the mental component score. This model can be specified using the formula: \( i1 \sim mcs \). Assess the distribution of the residuals for this model.

10.1. The RailTrail dataset within the mosaic package includes the counts of crossings of a rail trail in Northampton, Massachusetts for 90 days in 2005. City officials are interested in understanding usage of the trail network, and how it changes as a function of temperature and day of the week. Describe the distribution of the variable avgtemp in terms of its center, spread and shape.

```r
> favstats(~ avgtemp, data=RailTrail)

min  Q1 median  Q3 max mean    sd  n missing
 33 48.6  55.2  64.5  84 57.4 11.3 90    0

> densityplot(~ avgtemp, xlab="Average daily temp (degrees F)",
             data=RailTrail)
```
10.2. The RailTrail dataset also includes a variable called `cloudcover`. Describe the distribution of this variable in terms of its center, spread and shape.

10.3. The variable in the RailTrail dataset that provides the daily count of crossings is called `volume`. Describe the distribution of this variable in terms of its center, spread and shape.

10.4. The RailTrail dataset also contains an indicator of whether the day was a weekday (`weekday==1`) or a weekend/holiday (`weekday==0`). Use `tally()` to describe the distribution of this categorical variable. What percentage of the days are weekends/holidays?

10.5. Use side-by-side boxplots to compare the distribution of `volume` by day type in the RailTrail dataset. Hint: you’ll need to turn the numeric `weekday` variable into a factor variable using `as.factor()`. What do you conclude?

10.6. Use overlapping density plots to compare the distribution of `volume` by day type in the RailTrail dataset. What do you conclude?

10.7. Create a scatterplot of `volume` as a function of `avgtemp` using the RailTrail dataset, along with a regression line and scatterplot smoother (lowess curve). What do you observe about the relationship?

10.8. Using the RailTrail dataset, fit a multiple regression model for `volume` as a function of `cloudcover`, `avgtemp`,
weekday and the interaction between day type and average temperature. Is there evidence to retain the interaction term at the $\alpha = 0.05$ level?

10.9. Use `makeFun()` to calculate the predicted number of crossings on a weekday with average temperature 60 degrees and no clouds. Verify this calculation using the coefficients from the model.

```r
> coef(fm)

(Intercept) 378.83
cloudcover -17.20
avgtemp 2.31
weekday1 -321.12
avgtemp:weekday1 4.73
```

10.10. Use `makeFun()` and `plotFun()` to display predicted values for the number of crossings on weekdays and weekends/holidays for average temperatures between 30 and 80 degrees and a cloudy day (`cloudcover=10`).

10.11. Using the multiple regression model, generate a histogram (with overlaid normal density) to assess the normality of the residuals.

10.12. Using the same model generate a scatterplot of the residuals versus predicted values and comment on the linearity of the model and assumption of equal variance.

10.13. Using the same model generate a scatterplot of the residuals versus average temperature and comment on the linearity of the model and assumption of equal variance.

11.1. Generate a sample of 1000 exponential random variables with rate parameter equal to 2, and calculate the mean of those variables.

11.2. Find the median of the random variable $X$, if it is exponentially distributed with rate parameter 10.

12.1. Find the power of a two-sided two-sample $t$-test where both distributions are approximately normally distributed with the same standard deviation, but the group differ by 50% of the standard deviation. Assume
that there are 25 observations per group and an alpha level of 0.054.

12.2. Find the sample size needed to have 90% power for a two group t-test where the true difference between means is 25% of the standard deviation in the groups (with \( \alpha = 0.05 \)).

13.1. Using faithful dataframe, make a scatter plot of eruption duration times vs. the time since the previous eruption.

13.2. The fusion2 data set in the fastR package contains genotypes for another SNP. Merge fusion1, fusion2, and pheno into a single data frame.

Note that fusion1 and fusion2 have the same columns.

> names(fusion1)

[1] "id"       "marker"  "markerID"  "allele1"  "allele2"  "genotype"  "Adose"
[8] "Cdose"    "Gdose"    "Tdose"

> names(fusion2)

[1] "id"       "marker"  "markerID"  "allele1"  "allele2"  "genotype"  "Adose"
[8] "Cdose"    "Gdose"    "Tdose"

You may want to use the suffixes argument to merge() or rename the variables after you are done merging to make the resulting dataframe easier to navigate.

Tidy up your dataframe by dropping any columns that are redundant or that you just don’t want to have in your final dataframe.
16

Bibliography


# Index

<table>
<thead>
<tr>
<th>%&gt;%, 94</th>
<th>dropping variables, 95</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs(), 79, 80</td>
<td>exp(), 69</td>
</tr>
<tr>
<td>add option, 78, 87</td>
<td>factor reordering, 103</td>
</tr>
<tr>
<td>adding variables, 94</td>
<td>family option, 69</td>
</tr>
<tr>
<td>all.x option, 99</td>
<td>CPS85 dataset, 94</td>
</tr>
<tr>
<td>alpha option, 49, 62</td>
<td>failure time analysis, 73</td>
</tr>
<tr>
<td>analysis of variance, 65</td>
<td>faithful dataset, 96</td>
</tr>
<tr>
<td>anova(), 77</td>
<td>fit option, 31</td>
</tr>
<tr>
<td>anova(), 66, 70</td>
<td>fitted(), 79</td>
</tr>
<tr>
<td>aov(), 66, 75</td>
<td>format option, 30</td>
</tr>
<tr>
<td>arrange(), 98, 99</td>
<td>freqpolygon(), 30</td>
</tr>
<tr>
<td>auto.key option, 76, 78</td>
<td>function(), 46</td>
</tr>
<tr>
<td>band.lwd option, 53</td>
<td>gather(), 100</td>
</tr>
<tr>
<td>binom.test(), 40</td>
<td>glm(), 69</td>
</tr>
<tr>
<td>binomial test, 40</td>
<td>grid.text(), 34</td>
</tr>
<tr>
<td>bootstrapping, 36</td>
<td>group-wise statistics, 103</td>
</tr>
<tr>
<td>breaks option, 101</td>
<td>group_by(), 103</td>
</tr>
<tr>
<td>bwplot(), 61, 66, 75</td>
<td>groups option, 65, 78, 86</td>
</tr>
<tr>
<td>by.x option, 99</td>
<td>healthEvaluationAndLinkageToPrimaryCareStudy, 107</td>
</tr>
<tr>
<td>categorical variables, 39</td>
<td>head(), 94</td>
</tr>
<tr>
<td>center option, 87</td>
<td>data management, 93</td>
</tr>
<tr>
<td>cex option, 45, 62, 80</td>
<td>data(), 97</td>
</tr>
<tr>
<td>chisq.test(), 43, 58</td>
<td>dataframe, 94</td>
</tr>
<tr>
<td>class(), 48</td>
<td>dataframes</td>
</tr>
<tr>
<td>coef(), 47, 48, 103</td>
<td></td>
</tr>
<tr>
<td>coefficient plots, 79</td>
<td>density option, 48</td>
</tr>
<tr>
<td>col option, 34</td>
<td>densityplot(), 34</td>
</tr>
<tr>
<td>conf.int option, 73</td>
<td>derived variables, 101</td>
</tr>
<tr>
<td>confint(), 36, 41, 48</td>
<td>derivedFactor(), 102</td>
</tr>
<tr>
<td>contingency tables, 39, 55</td>
<td>diffmean(), 64</td>
</tr>
<tr>
<td>Cook's distance, 51</td>
<td>dim(), 104</td>
</tr>
<tr>
<td>cor(), 46</td>
<td>display first few rows, 94</td>
</tr>
<tr>
<td>correct option, 41</td>
<td>dnorm(), 83</td>
</tr>
<tr>
<td>correlation, 46</td>
<td>do(), 37</td>
</tr>
<tr>
<td>Cox proportional hazards</td>
<td>dotPlot(), 33</td>
</tr>
<tr>
<td>model, 74</td>
<td>dplyr package, 30, 31</td>
</tr>
<tr>
<td>coxph(), 74</td>
<td>drop option, 98, 99</td>
</tr>
<tr>
<td>CPS85 dataset, 94</td>
<td>failing time analysis, 73</td>
</tr>
<tr>
<td>creating subsets, 97</td>
<td>faithful dataset, 96</td>
</tr>
<tr>
<td>cross classification tables, 55</td>
<td>fitted(), 79</td>
</tr>
<tr>
<td>CrossTable(), 56</td>
<td>format option, 30</td>
</tr>
<tr>
<td>cut(), 94, 101</td>
<td>freqpolygon(), 30</td>
</tr>
<tr>
<td>data management, 93</td>
<td>function(), 46</td>
</tr>
<tr>
<td>data(), 97</td>
<td>gather(), 100</td>
</tr>
<tr>
<td>dataframe, 94</td>
<td>glm(), 69</td>
</tr>
<tr>
<td>density option, 48</td>
<td>grid.text(), 34</td>
</tr>
<tr>
<td>densityplot(), 34</td>
<td>group-wise statistics, 103</td>
</tr>
<tr>
<td>derived variables, 101</td>
<td>group_by(), 103</td>
</tr>
<tr>
<td>derivedFactor(), 102</td>
<td>groups option, 65, 78, 86</td>
</tr>
<tr>
<td>diffmean(), 64</td>
<td>healthEvaluationAndLinkageToPrimaryCareStudy, 107</td>
</tr>
<tr>
<td>dim(), 104</td>
<td>display first few rows, 94</td>
</tr>
<tr>
<td>display first few rows, 94</td>
<td>dnorm(), 83</td>
</tr>
<tr>
<td>do(), 37</td>
<td>dplyr package, 30, 31</td>
</tr>
<tr>
<td>drop option, 98, 99</td>
<td>failing time analysis, 73</td>
</tr>
<tr>
<td>faithful dataset, 96</td>
<td>fitted(), 79</td>
</tr>
<tr>
<td>fitted(), 79</td>
<td>format option, 30</td>
</tr>
<tr>
<td>format option, 30</td>
<td>freqpolygon(), 30</td>
</tr>
<tr>
<td>function(), 46</td>
<td>gather(), 100</td>
</tr>
<tr>
<td>glm(), 69</td>
<td>grid.text(), 34</td>
</tr>
<tr>
<td>group-wise statistics, 103</td>
<td>group_by(), 103</td>
</tr>
<tr>
<td>groups option, 65, 78, 86</td>
<td>healthEvaluationAndLinkageToPrimaryCareStudy, 107</td>
</tr>
</tbody>
</table>
HELP study, 107
HELPmiss dataset, 104
HELPPrct dataset, 27
histogram(), 29, 87
honest significant differences, 67
include.lowest option, 101
incomplete data, 104
inspect(), 93
inspecting dataframes, 93
install.packages(), 14
installing packages, 14
integrate(), 83
interactions, 77
iris dataset, 94
is.na(), 105
Kaplan-Meier plot, 73
knitr, 14
labels option, 67
ladd(), 34
layout option, 32
left_join(), 103
levels option, 67
leverage, 52
linear regression, 47
linearity, 45
lm(), 47, 67
loading packages, 14
logistic regression, 69
lowess, 45
lty option, 34
lwd option, 34, 45, 86
makeFun(), 78, 87
margins option, 39
markdown, 14
mean(), 27, 61
median(), 28, 75
merging dataframes, 98
missing data, 104
model comparison, 67
Modeling with R, 27
mosaic package, 27
mosaicplot(), 57
mplot(), 50, 68, 79
msummary(), 76
msummary(), 48, 63, 69
multiple comparisons, 67
multiple regression, 76
multivariate relationships, 76
mutate(), 67, 94, 101, 103
NA character, 104
na.omit(), 105
names(), 96
ncol(), 105
nint option, 32
nrow(), 103, 105
ntiles(), 102
oddsRatio(), 56
one-way ANOVA, 65
options(), 27
pairs plot, 47
panel.abline(), 34
panel.labels(), 46
panel.lmbands(), 53
panel.mathdensity(), 34
panel.text(), 46
paste(), 101
pbinom(), 90
pch option, 45, 86
pchisq(), 43
Pearson correlation, 46
permutation test, 64
pipe operator, 94
plotDist(), 43, 85
plotFun(), 78, 87
pnorm(), 83
polygons, 34
prediction bands, 53
print(), 41
prop.test(), 41
proportional hazards model, 74
pval(), 41
qdata(), 37
qnorm(), 83
qmath(), 49
quantile(), 28
quantiles, 28
random variables, 83
read.file(), 93
regression, 47
regression diagnostics, 79
relevel(), 103
rename(), 96
renaming variables, 96
reordering factors, 103
reproducible analysis, 14
require(), 14, 27
resample(), 36
resampling, 36, 64
reshaping dataframes, 100
resid(), 79
residual diagnostics, 79
residuals(), 48
rexp(), 87
rnorm(), 83
rownames(), 96
RStudio.Version(), 19
scale versus location, 51
scatterplot matrix, 47
scatterplots, 45
sd(), 28
select option, 95
select(), 94, 101, 103, 104
sessionInfo(), 19
shuffle(), 64
significance stars, 48
smoothers, 45
sorting dataframes, 98
Spearman correlation, 46
splom(), 47
spread(), 101
Start Modeling with R, 27
Start Teaching with R, 27
<table>
<thead>
<tr>
<th>Function</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>stem()</td>
<td>30</td>
</tr>
<tr>
<td>subsetting dataframes</td>
<td>97</td>
</tr>
<tr>
<td>sum()</td>
<td>43, 105</td>
</tr>
<tr>
<td>summarise()</td>
<td>103</td>
</tr>
<tr>
<td>summary()</td>
<td>63</td>
</tr>
<tr>
<td>Surv()</td>
<td>73</td>
</tr>
<tr>
<td>survfit()</td>
<td>73</td>
</tr>
<tr>
<td>survival analysis</td>
<td>73</td>
</tr>
<tr>
<td>t.test()</td>
<td>36, 62</td>
</tr>
<tr>
<td>tables</td>
<td>39, 55</td>
</tr>
<tr>
<td>tally()</td>
<td>30, 39, 55, 103</td>
</tr>
<tr>
<td>Teaching with R</td>
<td>27</td>
</tr>
<tr>
<td>test option</td>
<td>70</td>
</tr>
<tr>
<td>thinking with data</td>
<td>93</td>
</tr>
<tr>
<td>tidyr package</td>
<td>31, 99</td>
</tr>
<tr>
<td>time to event analysis</td>
<td>73</td>
</tr>
<tr>
<td>transforming dataframes</td>
<td>100</td>
</tr>
<tr>
<td>transposing dataframes</td>
<td>100</td>
</tr>
<tr>
<td>Tukey's HSD</td>
<td>67</td>
</tr>
<tr>
<td>TukeyHSD()</td>
<td>67</td>
</tr>
<tr>
<td>type option</td>
<td>45, 80, 86</td>
</tr>
<tr>
<td>under option</td>
<td>86</td>
</tr>
<tr>
<td>var()</td>
<td>28</td>
</tr>
<tr>
<td>var.equal option</td>
<td>62</td>
</tr>
<tr>
<td>vignettes</td>
<td>13</td>
</tr>
<tr>
<td>which option</td>
<td>50</td>
</tr>
<tr>
<td>width option</td>
<td>33, 87</td>
</tr>
<tr>
<td>wilcox.test()</td>
<td>63</td>
</tr>
<tr>
<td>with()</td>
<td>27, 105</td>
</tr>
<tr>
<td>xchisq.test()</td>
<td>44, 58</td>
</tr>
<tr>
<td>xlab option</td>
<td>73</td>
</tr>
<tr>
<td>xlim option</td>
<td>65</td>
</tr>
<tr>
<td>xpnorm()</td>
<td>83</td>
</tr>
<tr>
<td>xyplot()</td>
<td>45, 62, 78</td>
</tr>
<tr>
<td>ylab option</td>
<td>78</td>
</tr>
</tbody>
</table>