Nicholas J. Horton Daniel T. Kaplan Randall J. Pruim

# Compendium of Commands to Teach Statistics With

**Project MOSAIC** 

# Contents

1	Introduction 13	
2	One Quantitative Variable 15	
3	One Categorical Variable 25	
4	Two Quantitative Variables 33	
5	Two Categorical Variables 39	
6	Quantitative Response to a Categorical Predictor	45
7	Categorical Response to a Quantitative Predictor	53
8	Survival Time Outcomes 55	
9	More than Two Variables 57	
10	Probability Distributions and Random Variables	65
11	Power Calculations 69	

- 4 NICHOLAS J. HORTON, DANIEL T. KAPLAN AND RANDALL J. PRUIM
- 12 Data Management 73
- 13 Health Evaluation and Linkage to Primary Care (HELP) Study 87
- 14 Exercises and Problems 91
- 15 Bibliography 97

# About These Notes

These materials were initially created for a workshop entitled *Teaching Statistics Using R* prior to the 2011 United States Conference on Teaching Statistics and revised for the 2013 USCOTS. We organized these workshops to help instructors integrate R (as well as some related technologies) into their statistics courses at all levels. We got great feedback and many wonderful ideas from the participants and those that we've shared this with since the workshops.

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, data scraping from the internet and appropriate statistical analysis workflow. This follows the philosophy outlined by Nolan and Temple Lang<sup>1</sup>.

Throughout this book, 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

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.org with any comments, suggestions, corrections, etc. Updated versions will be posted at http://mosaic-web.org.

## What's Ours Is Yours - To a Point

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

<sup>1</sup> D. Nolan and D. Temple Lang. Computing in the statistics curriculum. *The American Statistician*, 64(2):97–107, 2010

### Caution!

Despite your 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.

licensing is available at this web page: http://www.mosaic-web.org/go/teachingRlicense.html.

### Two Audiences

The primary audience for these materials is instructors of statistics at the college or university level. A secondary 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 for 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 of this book 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 pretty straightforward for Mac, PC, or linux machines.

RStudio is an integrated development environment (IDE) that facilitates use of R for 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. There are several things we use that can only be done in RStudio (mainly things that make use manipulate() or RStudio's support for reproducible research). RStudio is available from http://www.rstudio.org/. RStudio can be installed as a desktop (laptop) application or as a server application that is accessible to others 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 (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.

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

# **Document Creation**

This document was created on July 27, 2013, using knitr and R version 3.0.0 (2013-04-03).

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 introduce upper level students to reproducible analysis methods. For beginners, we introduce knitr with RMardown, which produces HTML using a simpler syntax.

# Project MOSAIC

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 and 2013 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 and 2013.

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. A schedule of future M-casts and recordings of past M-casts are available at the Project MOSAIC web site, http://mosaic-web.org.

The development of a "concept inventory" to support teaching modeling. It is somewhat rare in today's curriculum for modeling to be taught. College and university catalogs are filled with descriptions of courses in statistics, computation, and calculus. There are many textbooks in these areas and most new faculty teaching statistics, computation, and calculus have a solid idea of what should be included. But modeling is different. It's generally recognized as important, but few if instructors have a clear view of the essential concepts.

The construction of syllabi and materials for courses that teach the MO-SAIC 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.

We welcome and encourage your participation in all of these initiatives.

# Statistical Computation and Computational Statistics

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.

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 had 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 and RStudio are 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.

Nevertheless, those who are unfamiliar with R or who have never used R for teaching are often cautious about using it with students. If you are in that category, then this book is for you. Our goal is to reveal some of what we have learned teaching with R and to make teaching statistics with R as rewarding and easy as possible – for both students and faculty. We will cover both technical aspects of R and RStudio (e.g. how do I get R to do thus and such?) as well as some perspectives on how to use computation to teach statistics. The latter will be illustrated in R but would be equally applicable with other statistical software.

Others have used R in their courses, but have perhaps left the course feeling like there must have been better ways to do this or that topic. If that sounds more like you, then this book is for you, too. As we have been working on this book, we have also been developing the mosaic R package (available on CRAN) to make certain aspects of statistical computation and computational statistics simpler for beginners. You will also find here some of our favorite activities, examples, and data sets, as well as answers to questions that we have heard frequently from both students and faculty colleagues. We invite you to scavenge from our materials and ideas and modify them to fit your courses and your students.

# Introduction

In this monograph, 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 trial of a novel way to link at-risk subjects with primary care. More information on the dataset can be found in chapter 13.

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 even if you present things in a different order. This is also a good organizational template to give your students to help them keep straight "what to do when".

Some data management is needed by students (and more by instructors). This material is reviewed in chapter 12.

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. A short summary of the R needed to teach introductory statistics can be found in the mosaic package vignette (http://cran.r-project.org/web/packages/mosaic/vignettes/MinimalR.pdf).

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<sup>1</sup> (see http://www.math.smith.edu/~nhorton/ips6e) as well as the third edition of the Statistical Sleuth<sup>2</sup> (see http://www.math.smith.edu/~nhorton/sleuth3).

To use a package within R, it must be installed (one time), and

<sup>&</sup>lt;sup>1</sup> D. S. Moore and G. P. McCabe. *Introduction to the Practice of Statistics*. W.H.Freeman and Company, 6th edition, 2007

<sup>&</sup>lt;sup>2</sup> Fred Ramsey and Dan Schafer. *Statistical Sleuth: A Course in Methods of Data Analysis*. Cengage, 3rd edition, 2013

loaded (each session). The  ${\tt mosaic}$  package can be installed using the following command:

install.packages("mosaic") # note the quotation marks

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:

require(mosaic)

# One Quantitative Variable

## 2.1 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, intra-quartile 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 ?options() for more configuration possibilities).

```
require(mosaic)
options(digits = 3)
```

Note that the mean() function in the mosaic package supports a modeling language common to lattice graphics and linear models (e.g. lm()). We will use this modeling language commands throughout this document.

```
mean(~cesd, data = HELPrct)
[1] 32.8
```

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

```
with(HELPrct, mean(cesd))
```

DIGGING DEEPER
The *Start Modeling with R* book will be helpful if you are unfamiliar with the modeling language.

```
[1] 32.8
mean(HELPrct$cesd)
[1] 32.8
```

Similar functionality exists for other summary statistics.

```
sd(~cesd, data = HELPrct)
[1] 12.5
```

```
sd(~cesd, data = HELPrct)^2

[1] 157

var(~cesd, data = HELPrct)

[1] 157
```

It is also straightforward to calculate quantiles of the distribution.

```
median(~cesd, data = HELPrct)
[1] 34
```

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

```
with(HELPrct, quantile(cesd))

0% 25% 50% 75% 100%
    1 25 34 41 60

with(HELPrct, quantile(cesd, c(0.025, 0.975)))
```

Caution!
Not all commands (including quantile()) have been upgraded to support the formula interface. These must be accessed using with() or the \$ operator.

```
2.5% 97.5%
 6.3 55.0
```

Finally, the favstats() function in the mosaic package provides a concise summary of many useful statistics.

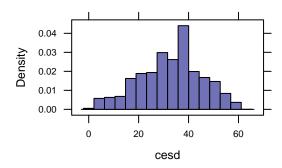
```
favstats(~cesd, data = HELPrct)
min Q1 median Q3 max mean sd
                               n missing
   1 25
           34 41 60 32.8 12.5 453
```

# Graphical summaries

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

x is for eXtra.

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



In the HELPrct dataset, approximately one quarter of the subjects are female.

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

```
female male Total
  107  346  453

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

female male Total
  23.6  76.4  100.0
```

It is straightforward to restrict our attention to just those subjects. If we are going to do many things with a subset of our data, it may be easiest to make a new data frame containing only the cases we are interested in. The subset() function can generate a new data frame containing just the women or just the men (see also section 12.4). Once this is created, we used the stem() function to create a stem and leaf plot.

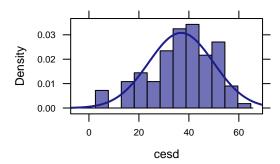
```
female <- subset(HELPrct, sex == "female")</pre>
male <- subset(HELPrct, sex == "male")</pre>
with(female, stem(cesd))
  The decimal point is 1 digit(s) to the right of the |
  0 | 3
  0 | 567
  1 | 3
  1 | 555589999
  2 | 123344
  2 | 66889999
  3 | 0000233334444
  3 | 5556666777888899999
  4 | 00011112222334
  4 | 555666777889
  5 | 011122222333444
  5 | 67788
  6 | 0
```

equal signs

Caution! Note that the equality operator is *two* 

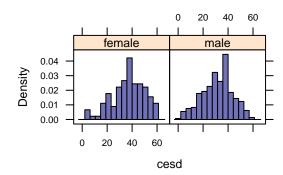
Subsets can also be generated and used on the fly (this time including an overlaid normal density):

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



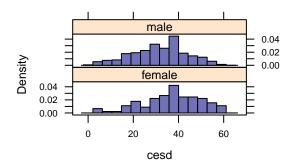
Alternatively, we can make side-by-side plots to compare multiple subsets.

histogram(~cesd | sex, data = HELPrct)



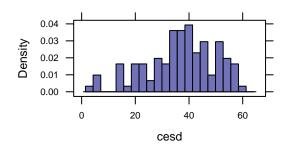
The layout can be rearranged.

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



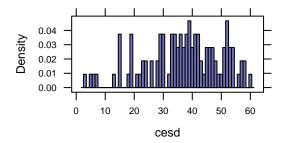
We can control the number of bins in a number of ways. These can be specified as the total number.

histogram(~cesd, nint = 20, data = female)



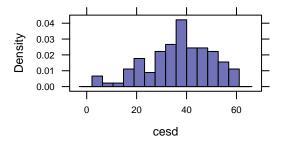
Or the width can be specified.

histogram(~cesd, width = 1, data = female)



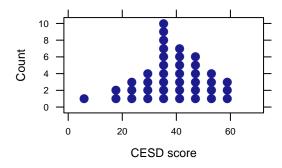
We could also have made our subset "on the fly", just for the purposes of graphing:

```
histogram(~cesd, data = HELPrct, subset = (sex == "female"))
```



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

```
dotPlot(~ cesd, xlab="CESD score",
  data=subset(HELPrct, sex=="female" & homeless=="homeless"))
```



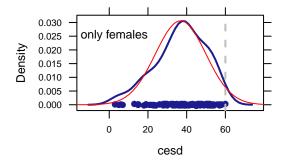
# 2.3 Density curves

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 gratuitous 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 can be specified, as well as line widths.

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

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



# DIGGING DEEPER The plotFun() function can also be used to annotate plots (see section 9.2.1).

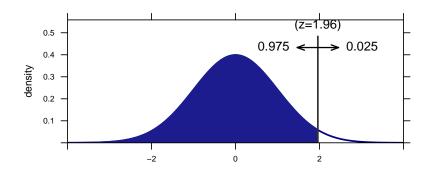
## 2.4 Normal distributions

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 10.

x is for eXtra.

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

```
If X \sim N(0,1), then
P(X \le 1.96) = P(Z \le 1.96) = 0.975
P(X > 1.96) = P(Z > 1.96) = 0.025
[1] 0.975
```



# 2.5 Inference for a single sample

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

```
with(female, t.test(cesd))
One Sample t-test
data: x
t = 29.3, df = 106, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to {\tt O}
95 percent confidence interval:
34.4 39.4
sample estimates:
mean of x
     36.9
confint(with(female, t.test(cesd)))
```

```
mean of x lower upper level 36.89 34.39 39.38 0.95
```

But it's also straightforward to calculate this using a bootstrap (using the approach described in the mosaic package Resampling Vignette). The statistic that we want to resample is the mean.

```
mean(~cesd, data = female)
[1] 36.9
```

One resampling trial can be carried out:

```
mean(~cesd, data = resample(female))
[1] 36
```

Another will yield different results:

```
mean(~cesd, data = resample(female))
[1] 36.5
```

Now conduct 1000 resampling trials, saving the results in an object called trials:

```
trials = do(1000) * mean(~cesd, data = resample(female))
with(trials, quantile(result, c(0.025, 0.975)))

2.5% 97.5%
34.3 39.4
```

TEACHING TIP
Even though a single trial is of little use, it's smart have students do the calculation to show that they are (usually!) getting a different result than without resampling.

# One Categorical Variable

# 3.1 Numerical summaries

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

```
tally(~homeless, data = HELPrct)

homeless housed Total
    209    244    453

tally(~homeless, margins = FALSE, data = HELPrct)

homeless housed
    209    244

tally(~homeless, format = "percent", data = HELPrct)

homeless housed Total
    46.1    53.9    100.0

tally(~homeless, format = "proportion", data = HELPrct)

homeless housed Total
    0.461    0.539    1.000
```

# 3.2 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.

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

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" "method"
[9] "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
                                          0.415
                  0.461
                  upper
                                          level
                  0.509
                                          0.950
pval(result)
p.value
   0.11
```

# *The proportion test*

A similar interval and test can be calculated using prop.test().

```
tally(~ homeless, data=HELPrct)
```

DIGGING DEEPER Most of the objects in R have a print() method. So when we get result, what we are seeing displayed in the console is print(result). There may be a good deal of additional information lurking inside the object itself. To make matter even more complicated, some objects are returned invisibly, so nothing prints. You can still assign the returned object to a variable and process it later, even if nothing shows up on the screen. This is the case for the lattice graphics functions, for example. You can save a plot into a variable, say myplot and display the plot again later using print(myplot).

```
homeless housed Total
209 244 453

prop.test(~ homeless=="homeless", correct=FALSE, data=HELPrct)

1-sample proportions test without continuity
correction

data: HELPrct$homeless == "homeless"

X-squared = 2.7, df = 1, p-value = 0.1001
alternative hypothesis: true p is not equal to 0.5

95 percent confidence interval:
0.416 0.507

sample estimates:
    p
0.461
```

It also accepts summarized data, the way binom.test() does.

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

1-sample proportions test without continuity
correction

data: x and n
X-squared = 2.7, df = 1, p-value = 0.1001
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
    0.416    0.507
sample estimates:
    p
0.461
```

To make things simpler still, we've added a formula interface in the mosaic package.

```
prop.test(~homeless, data = HELPrct)
```

prop.test() calculates a Chi-squared statistic. Most introductory texts use a *z*-statistic. They are mathematically equivalent in terms of inferential statements, but you may need to address the discrepancy with your students.

```
1-sample proportions test with continuity correction
data: HELPrct$homeless
X-squared = 2.55, df = 1, p-value = 0.1102
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.415 0.509
sample estimates:
    р
0.461
```

# 3.4 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 Total
  39.1 33.6 27.4 100.0
observed <- tally(~ substance, margins=FALSE, data=HELPrct)</pre>
observed
alcohol cocaine heroin
   177 152 124
```

```
CAUTION!
The margins=FALSE option is needed
here to include only the counts.
```

```
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 = 9.31, df = 2, p-value = 0.009508

total <- sum(observed); total

[1] 453

expected <- total*p; expected

[1] 151 151 151</pre>
```

We can also calculate this quantity manually, in terms of observed and expected values.

```
chisq <- sum((observed - expected)^2/(expected)); chisq
[1] 9.31
1 - pchisq(chisq, df=2)
[1] 0.00951</pre>
```

TEACHING TIP We don't encourage much manual calculations in our courses.

Alternatively, the  ${\tt mosaic}$  package provides a version of  ${\tt chisq.test()}$  with more verbose output.

x is for eXtra.

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

Chi-squared test for given probabilities

data: observed
X-squared = 9.31, df = 2, p-value = 0.009508

177     152     124
(151.00) (151.00) (151.00)
[4.4768] [0.0066] [4.8278]
< 2.116> < 0.081> <-2.197>
```

```
key:
observed
(expected)
[contribution to X-squared]
<residual>
# clean up variables no longer needed
rm(observed, p, total, chisq)
```

# 4

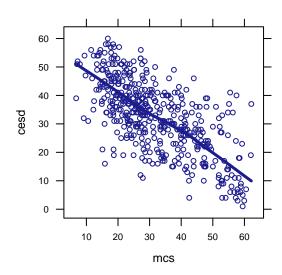
# Two Quantitative Variables

# 4.1 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) with a lowess (locally weighted scatterplot smoother) line, using a circle as the plotting character and slightly thicker line.

The lowess line can help to assess linearity of a relationship.

```
xyplot(cesd ~ mcs, type=c('p','smooth'), pch=1, cex=0.6,
  lwd=3, data=HELPrct)
```



# 4.2 Correlation

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

By default, Pearson correlations are provided, other variants (e.g. Spearman) can be specified using the method option.

```
cor(cesd, mcs, method = "spearman", data = HELPrct)
[1] -0.661
```

# 4.3 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  $\sim$  mcs.

We tend to introduce linear regression early in our courses, as a purely descriptive technique.

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

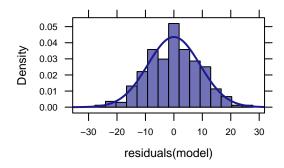
```
options(show.signif.stars = FALSE)
coef(model)
(Intercept)
              mcs
    53.902 -0.665
summary(model)
Call:
lm(formula = cesd ~ mcs, data = HELPrct)
Residuals:
   Min 1Q Median 3Q Max
-27.359 -6.728 -0.002 6.237 24.424
Coefficients:
         Estimate Std. Error t value Pr(>|t|)
(Intercept) 53.9022 1.1472 47.0 <2e-16
         -0.6647 0.0336 -19.8 <2e-16
Residual standard error: 9.16 on 451 degrees of freedom
Multiple R-squared: 0.465, Adjusted R-squared: 0.464
F-statistic: 392 on 1 and 451 DF, p-value: <2e-16
confint(model)
           2.5 % 97.5 %
(Intercept) 51.648 56.157
         -0.731 -0.599
mcs
r.squared(model)
[1] 0.465
```

```
class(model)
[1] "lm"
```

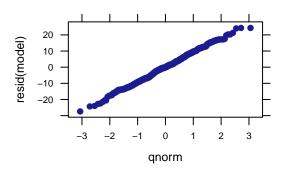
The return value from lm() is a linear model object. A number of functions can operate on these objects, as seen previously with coef(). The function residuals() returns a vector of the residuals.

histogram(~residuals(model), density = TRUE)

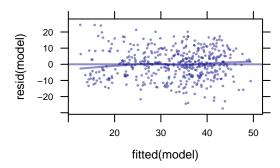
The function residuals() can be abbreviated resid(). Another useful function is fitted(), which returns a vector of predicted values.



# qqmath(~resid(model))

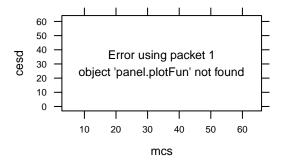


```
xyplot(resid(model) ~ fitted(model), type = c("p", "smooth",
    "r"), alpha = 0.5, cex = 0.3, pch = 20)
```



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)
```



## 5

# Two Categorical Variables

#### 5.1 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.

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

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

We can calculate the odds ratio directly from the table:

```
OR = (40*177)/(67*169); OR

[1] 0.625
```

The mosaic package has a function which will calculate odds ratios:

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

Odds Ratio
```

```
Proportions
Prop. 1: 0.1914
Prop. 2: 0.2746

Odds
Odds 1: 0.2367
Odds 2: 0.3785

Odds Ratio
Odds Ratio: 1.599

95 percent confidence interval:
1.025 < OR < 2.495
```

Note that the reference group is flipped (and that 1/1.599 = 0.625).

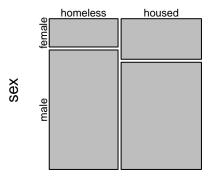
Graphical summaries of cross classification tables may be helpful in visualizing associations. Mosaic plots are one example (though the jury is still out regarding their utility, relative to the low data to ink ratio<sup>1</sup>). Here we see that males tend to be over-represented amongst the homeless subjects (as represented by the horizontal line which is higher for the homeless rather than the housed).

<sup>1</sup> E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT, 2nd edition, 2001

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

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

## mytab



homeless

## Chi-squared tests

```
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 = 4.32, df = 1, p-value = 0.03767
```

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 xchisq.test() function provides additional details to help with this process.

**x** is for eXtra.

```
xchisq.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 = 4.32, df = 1, p-value = 0.03767
   40
          169
(49.37) (159.63)
 [1.78]
        [0.55]
<-1.33> < 0.74>
   67
          177
(57.63) (186.37)
 [1.52] [0.47]
< 1.23> <-0.69>
key:
observed
(expected)
[contribution to X-squared]
<residual>
```

We observe that there are fewer homeless women, and more homeless men that would be expected.

### 5.3 Fisher's exact test

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

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

Fisher's Exact Test for Count Data

data: tally(~homeless + sex, margins = FALSE, data = HELPrct)
p-value = 0.0456
```

DIGGING DEEPER
Note the different estimate of the odds ratio from that seen in section 5.1. The fisher.test() function uses a different estimator (and different interval based on the profile likelihood).

```
alternative hypothesis: true odds ratio is not equal to \ensuremath{\text{1}}
95 percent confidence interval:
0.389 0.997
sample estimates:
odds ratio
     0.626
```

# Quantitative Response to a Categorical Predictor

### 6.1 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.

```
mean(cesd ~ sex, data = HELPrct)

female male
  36.9 31.6
```

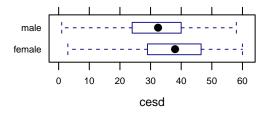
The favstats() function can provide more statistics by group.

```
favstats(cesd ~ sex, data = HELPrct)
   female.min female.Q1 female.median
                                      female.Q3
        3.0
              29.0
                         38.0
                                     46.5
   female.max female.mean female.sd
                                      female.n
       60.0
                  36.9
                              13.0
                                         107.0
              male.min
female.missing
                           male.Q1 male.median
                  1.0
                              24.0
                                          32.5
                       male.mean
     male.Q3
                male.max
                                       male.sd
       40.0
                 58.0
                          31.6
                                         12.1
      male.n male.missing
      346.0
```

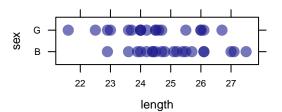
Boxplots are a particularly helpful graphical display to compare distributions. The bwplot() function can be used to display the boxplots for the CESD scores separately by sex. We see from both the

numerical and graphical summaries that women tend to have slightly higher CESD scores than men.

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.



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.

## 6.2 A dichotomous predictor: two-sample t

The Student's two sample t-test can be run without or with an equal variance assumption.

```
t.test(cesd ~ sex, var.equal=FALSE, data=HELPrct)
Welch Two Sample t-test
data: cesd by sex
t = 3.73, df = 167, p-value = 0.0002587
alternative hypothesis: true difference in means is not equal to \Theta
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.

The groups can also be compared using the lm() function (with an equal variance assumption).

```
summary(lm(cesd ~ sex, data = HELPrct))
Call:
lm(formula = cesd ~ sex, data = HELPrct)
Residuals:
  Min 10 Median 30 Max
-33.89 -7.89 1.11 8.40 26.40
Coefficients:
         Estimate Std. Error t value Pr(>|t|)
(Intercept) 36.89 1.19 30.96 < 2e-16
            -5.29 1.36 -3.88 0.00012
sexmale
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
```

### 6.3 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 = 23105, p-value = 0.0001033
alternative hypothesis: true location shift is not equal to 0
```

#### 6.4 Permutation test

Here we extend the methods introduced in section 2.5 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 <- compareMean(age ~ sex, data = HELPrct)
test.stat

[1] -0.784</pre>
```

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

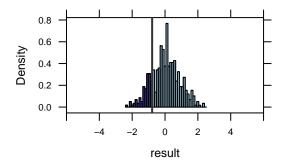
```
do(1) * compareMean(age ~ shuffle(sex), data = HELPrct)

result
1  1.23

do(1) * compareMean(age ~ shuffle(sex), data = HELPrct)
```

```
result
1 0.134
do(3) * compareMean(age ~ shuffle(sex), data = HELPrct)
  result
1 -1.139
2 -1.567
3 0.293
```

```
rtest.stats = do(500) * compareMean(age ~ shuffle(sex),
  data=HELPrct)
histogram(~ result, n=40, xlim=c(-6, 6),
  groups=result >= test.stat, pch=16, cex=.8,
  data=rtest.stats)
ladd(panel.abline(v=test.stat))
```

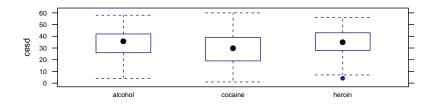


Here we don't see much evidence to contradict the null hypothesis that men and women have the same mean age in the population.

## 6.5 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).

#### bwplot(cesd ~ substance, data = HELPrct)



```
mean(cesd ~ substance, data = HELPrct)
alcohol cocaine heroin
  34.4   29.4   34.9
```

```
mod <- aov(cesd ~ substance, data = HELPrct)
summary(mod)

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.

```
mod1 <- lm(cesd ~ 1, data = HELPrct)
mod2 <- lm(cesd ~ substance, data = HELPrct)</pre>
```

The anova() command can summarize models.

```
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
```

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

```
anova(mod1, mod2)
Analysis of Variance Table
Model 1: cesd ~ 1
Model 2: cesd ~ substance
  Res.Df RSS Df Sum of Sq
                            F Pr(>F)
    452 70788
2
    450 68084 2
                    2704 8.94 0.00016
```

### 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 Difference (HSD). Other options are available within the multcomp package.

```
favstats(cesd ~ substance, data = HELPrct)
   alcohol.min
                 alcohol.Q1 alcohol.median
          4.0
                 26.0 36.0
    alcohol.Q3
                 alcohol.max alcohol.mean
         42.0
                       58.0
                                      34.4
    alcohol.sd
                  alcohol.n alcohol.missing
         12.1
                       177.0
                 cocaine.Q1 cocaine.median
   cocaine.min
                       19.0
                                      30.0
          1.0
    cocaine.Q3
                 cocaine.max
                              cocaine.mean
         39.0
                       60.0
                                      29.4
    cocaine.sd
                   cocaine.n cocaine.missing
         13.4
                       152.0
    heroin.min
                  heroin.Q1 heroin.median
                        28.0
                                      35.0
          4.0
     heroin.Q3
                  heroin.max heroin.mean
         43.0
                        56.0
     heroin.sd
                    heroin.n heroin.missing
         11.2
                       124.0
                                       0.0
```

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

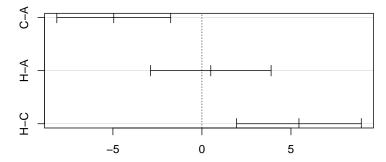
Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = x)

$subgrp
    diff lwr upr p adj
C-A -4.952 -8.15 -1.75 0.001
H-A 0.498 -2.89 3.89 0.936
H-C 5.450 1.95 8.95 0.001</pre>
```

```
plot(compare, cex.lab = 0.5)
```

#### 95% family-wise confidence level



Differences in mean levels of subgrp

Again, we see that the cocaine group has significantly lower CESD scores than the other two groups.

# Categorical Response to a Quantitative Predictor

## 7.1 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.

```
The glm() function has arguments
                                                                      family, which can take an option link.
logitmod <- glm(homeless ~ age + female, family=binomial,</pre>
                                                                      The logit link is the default link for the
  data=HELPrct)
                                                                      binomial family, so we don't need to
summary(logitmod)
                                                                      specify it here.
Call:
glm(formula = homeless ~ age + female, family = binomial, data = HELPrct)
Deviance Residuals:
   Min
           10 Median
                             30
                                     Max
-1.547 -1.202 0.918
                         1.123
                                   1.360
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 0.8926
                          0.4537
                                     1.97
                                              0.049
             -0.0239
                          0.0124 -1.92
                                              0.055
age
              0.4920
                          0.2282
                                     2.16
                                              0.031
female
(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
```

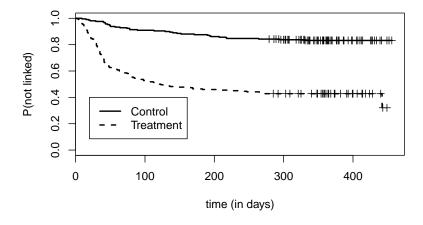
## Survival Time Outcomes

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

## 8.1 Kaplan-Meier plot

```
require(survival)
fit <- survfit(Surv(dayslink, linkstatus) ~ treat,
   data=HELPrct)
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)")</pre>
```

#### Product-Limit Survival Estimates (time to linkage)



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.

### 8.2 Cox proportional hazards model

```
require(survival)
summary(coxph(Surv(dayslink, linkstatus) ~ age + substance,
 data=HELPrct))
Call:
coxph(formula = Surv(dayslink, linkstatus) ~ age + substance,
   data = HELPrct)
 n= 431, number of events= 163
  (22 observations deleted due to missingness)
                   coef exp(coef) se(coef)
                                              z Pr(>|z|)
                                                  0.38
                0.00893 1.00897 0.01026 0.87
substancecocaine 0.18045 1.19775 0.18100 1.00
                                                   0.32
substanceheroin -0.28970 0.74849 0.21725 -1.33
                                                   0.18
                exp(coef) exp(-coef) lower .95 upper .95
                   1.009
                            0.991 0.989
                                                1.03
age
                                                 1.71
                   1.198
                             0.835
                                       0.840
substancecocaine
substanceheroin
                   0.748
                            1.336
                                     0.489
                                                 1.15
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
```

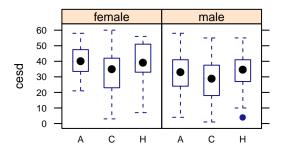
Neither age, nor substance group was significantly associated with linkage to primary care.

## 9

## More than Two Variables

## 9.1 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.



```
summary(aov(cesd ~ substance + sex, data = HELPrct))

Df Sum Sq Mean Sq F value Pr(>F)
```

```
    substance
    2
    2704
    1352
    9.27 0.00011

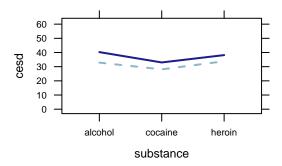
    sex
    1
    2569
    2569
    17.61 3.3e-05

    Residuals
    449 65515
    146
```

```
summary(aov(cesd ~ substance * sex, data = HELPrct))
               Df Sum Sq Mean Sq F value Pr(>F)
substance
                    2704
                             1352
                                     9.25 0.00012
                    2569
                             2569
                1
                                    17.57 3.3e-05
sex
                2
                              73
                                     0.50 0.60752
substance:sex
                     146
Residuals
              447
                   65369
                              146
```

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, type='a',
  data=HELPrct)
```



## 9.2 Multiple regression

Multiple regression is a logical extension of the prior commands, adding additional predictors (and allowing 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 also 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*.

```
lm1 <- lm(cesd ~ mcs + age + sex, data = HELPrct)</pre>
summary(lm1)
Call:
lm(formula = cesd ~ mcs + age + sex, data = HELPrct)
Residuals:
  Min 10 Median 30 Max
-26.924 -6.363 0.403 6.453 25.217
Coefficients:
          Estimate Std. Error t value Pr(>|t|)
(Intercept) 53.8303 2.3617 22.79 <2e-16
          -0.6548
                     0.0336 -19.50 <2e-16
mcs
           0.0553 0.0556 1.00 0.3200
age
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

```
lm2 <- lm(cesd ~ mcs + age + sex + mcs:sex, data = HELPrct)</pre>
summary(lm2)
Call:
lm(formula = cesd ~ mcs + age + sex + mcs:sex, data = HELPrct)
Residuals:
       1Q Median 3Q
   Min
                               Max
-26.667 -6.406 0.289 6.133 24.832
Coefficients:
          Estimate Std. Error t value Pr(>|t|)
(Intercept) 55.3906 2.9903 18.52 <2e-16
          -0.7082 0.0712 -9.95 <2e-16
mcs
age
           0.0549 0.0556 0.99 0.324
          -4.9421
                    2.6055 -1.90 0.058
sexmale
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(lm2)
Analysis of Variance Table
Response: cesd
          Df Sum Sq Mean Sq F value Pr(>F)
          1 32918 32918 398.27 <2e-16
mcs
          1
             107
                     107 1.29 0.2563
age
                      676 8.18 0.0044
sex
           1
               676
         1
               60
                      60 0.72 0.3952
mcs:sex
Residuals 448 37028
                       83
```

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

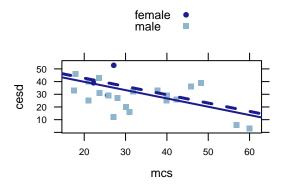
#### 9.2.1 Visualizing the results from the regression

The makeFun() and plotFun() functions from the mosaic package can be used to display the results from a regression model. For this example, we might display the predicted CESD values for a range of MCS values a 36 year old male and female subject from the parallel slopes model.

```
lm1fun = makeFun(lm1)
```

We can now plot this function for a range of values for MCS (mental component score), along with the observed data for 36 year olds.

```
xyplot(cesd ~ mcs, groups=sex, auto.key=TRUE,
  data=subset(HELPrct, age==36))
plotFun(lm1fun(mcs, age=36, sex="male") ~ mcs,
  xlim=c(0,60), lwd=2, ylab="predicted CESD", add=TRUE)
NULL
plotFun(lm1fun(mcs, age=36, sex="female") ~ mcs,
  xlim=c(0,60), lty=2, lwd=3, add=TRUE)
NULL
```



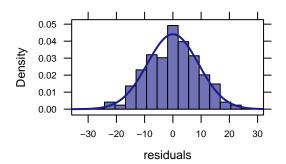
#### Residual diagnostics

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

```
HELPrct = transform(HELPrct, residuals = resid(lm1))
HELPrct = transform(HELPrct, pred = fitted(lm1))
```

```
Caution!
Be careful when fitting regression
models with missing values (see also
section 12.9).
```

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

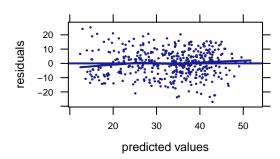


We can display observations with extremely large residuals.

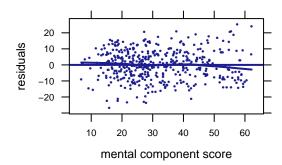
```
subset(HELPrct, abs(residuals) > 25)
    age anysubstatus anysub cesd d1 daysanysub dayslink
40
    43
                   0
                              16 15
                                           191
                                                    414
                         no
351 27
                  NA
                       <NA>
                              40 1
                                            NA
                                                    365
    drugrisk e2b female sex g1b homeless i1 i2 id indtot
40
           0 NA
                      0 male no homeless 24 36 44
                                                        41
           3
             2
                      0 male no homeless 18 18 420
                                                        37
    linkstatus link mcs pcs pss_fr racegrp satreat
40
                 no 15.9 71.4
                                       white
                                                  no
351
             0
                 no 57.5 37.7
                                   8
                                       white
                                                 yes
    sexrisk substance treat subgrp residuals pred
          7
                                 C
                                       -26.9 42.9
40
              cocaine
                        yes
351
          3
               heroin
                                 Н
                                        25.2 14.8
                         no
```

```
xyplot(residuals ~ pred, ylab="residuals", cex=0.3,
    xlab="predicted values", main="predicted vs. residuals",
    type=c("p", "r", "smooth"), data=HELPrct)
```

## predicted vs. residuals



```
xyplot(residuals ~ mcs, xlab="mental component score",
  ylab="residuals", cex=0.3,
  type=c("p", "r", "smooth"), data=HELPrct)
```



The assumptions of normality, linearity and homoscedasticity seem reasonable here.

# Probability Distributions and Random Variables

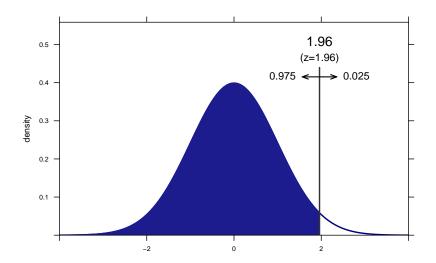
R can calculate quantities related to probability distributions of all types. It is straightforward to generate random variables from these distributions, which can be used for simulation and analysis.

```
xpnorm(1.96, mean = 0, sd = 1) # P(Z < 1.96)

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

P(X <= 1.96) = P(Z <= 1.96) = 0.975
P(X > 1.96) = P(Z > 1.96) = 0.025

[1] 0.975
```



```
# value which satisfies P(Z < z) = 0.975

qnorm(0.975, mean = 0, sd = 1)

[1] 1.96

integrate(dnorm, -Inf, 0) # P(Z < 0)

0.5 with absolute error < 4.7e-05
```

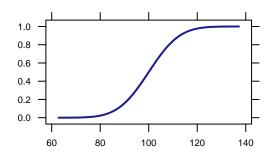
The following table displays the basenames for probability distributions available within base R. These functions can be prefixed by d to find the density function for the distribution, p to find the cumulative distribution function, q to find quantiles, and r to generate random draws. For example, to find the density function of a binomial random variable, use the command dbinom(). The qDIST() function is the inverse of the pDIST() function, for a given basename DIST.

Distribution	NAME
Beta	beta
binomial	binom
Cauchy	cauchy
chi-square	chisq
exponential	exp
F	f
gamma	gamma
geometric	geom
hypergeometric	hyper
logistic	logis
lognormal	lnorm
negative binomial	nbinom
normal	norm
Poisson	pois
Student's t	t
Uniform	unif
Weibull	weibull

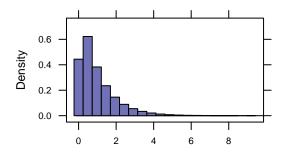
The plotDist() can be used to display distributions in a variety of ways.

DIGGING DEEPER
The fitdistr() within the MASS package facilitates estimation of parameter for many distributions.

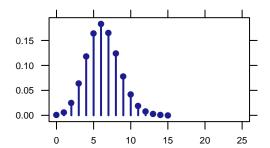
```
plotDist('norm', params=list(mean=100, sd=10),
  kind='cdf')
```



## plotDist('exp', kind='histogram')



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



## 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.

Let  $X_1,...,X_{25}$  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  $\alpha = .05$ . 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.

## 11.1 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.

```
qbinom(0.95, size = 25, prob = 0.5)

[1] 17

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</pre>
```

```
[4,] 18 0.00732
[5,] 19 0.00204
```

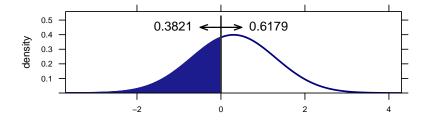
So we see that if we decide to reject when the number of positive values is 17 or larger, we will have an  $\alpha$  level of 0.054, which is near the nominal value in the problem.

We calculate the power of the sign test as follows. The probability that X > 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 <= 0) = P(Z <= -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.

#### 11.2 *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. First we find the rejection region.

```
alpha <- 1-pbinom(16, size=25, prob=0.5); alpha
[1] 0.0539
n <- 25; sigma <- 1 # given
stderr <- sigma/sqrt(n)</pre>
zstar <- qnorm(1-alpha, mean=0, sd=1)</pre>
zstar
[1] 1.61
crit <- zstar∗stderr
crit
[1] 0.322
```

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:

```
power <- 1 - pnorm(crit, mean = 0.3, sd = stderr)</pre>
power
[1] 0.457
```

Thus, 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 power.t.test() function.

```
power.t.test(n = 25, delta = 0.3, sd = 1, sig.level = alpha,
    alternative = "one.sided", type = "one.sample")$power
[1] 0.441
```

This 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.

Teaching Tip It's useful to have students calculate power empirically, to demonstrate the power of simulations.

## 12

## Data Management

Data management is a key capacity to allow students (and instructors) to "compute with data" or as Diane Lambert 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.

## 12.1 Adding new variables to a data frame

We can add additional variables to an existing data frame by simple assignment.

```
head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
                                   1.4
1
          5.1
                      3.5
                                                0.2 setosa
2
           4.9
                       3.0
                                    1.4
                                                0.2 setosa
3
          4.7
                      3.2
                                   1.3
                                                0.2 setosa
4
          4.6
                      3.1
                                   1.5
                                                0.2 setosa
5
           5.0
                       3.6
                                    1.4
                                                0.2 setosa
6
           5.4
                       3.9
                                    1.7
                                                0.4 setosa
```

```
# cut places data into bins
iris <- transform(iris, Length = cut(Sepal.Length, 4:8))</pre>
```

```
head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1
          5.1
                      3.5
                                   1.4
                                               0.2 setosa
2
          4.9
                      3.0
                                   1.4
                                               0.2 setosa
3
          4.7
                      3.2
                                   1.3
                                               0.2 setosa
4
          4.6
                      3.1
                                  1.5
                                               0.2 setosa
5
          5.0
                      3.6
                                   1.4
                                               0.2 setosa
6
          5.4
                      3.9
                                   1.7
                                               0.4 setosa
  Length
1 (5,6]
2 (4,5]
3 (4,5]
4 (4,5]
5 (4,5]
6 (5,6]
```

The CPS85 data frame contains data from a Current Population Survey (current in 1985, that is). Two of the variables in this data frame 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 this as a new variable in the data frame simply by assigning to it:

```
CPS85 <- transform(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</pre>
```

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.

```
with(CPS85, table(exper - workforce.years))

0  4
533  1
```

## 12.2 Dropping variables

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

```
names (CPS85)
                     "educ"
[1] "wage"
                                       "race"
                    "hispanic"
 [4] "sex"
                                      "south"
[7] "married"
                     "exper"
                                       "union"
[10] "age"
                     "sector"
                                       "workforce.years"
CPS1 <- subset(CPS85, select = -workforce.years)</pre>
names(CPS1)
 [1] "wage" "educ" "race"
                                    "sex"
                                              "hispanic"
[6] "south" "married" "exper"
                                    "union"
                                              "age"
[11] "sector"
```

Any number of variables can be dropped or kept in this manner by supplying a vectors of variables names.

```
CPS1 <- subset(CPS85, select = -c(workforce.years, exper))</pre>
```

If we only want to work with the first few variables, we can discard the rest in a similar way. Columns can be specified by number as well as name (but this can be dangerous if you are wrong about where the columns are):

```
CPSsmall <- subset(CPS85, select = 1:4)</pre>
head(CPSsmall, 2)
 wage educ race sex
1 9.0 10 W M
2 5.5 12 W M
```

## 12.3 Renaming variables

Both the column (variable) names and the row names of a data frames can be changed by simple assignment using names() or row.names().

```
ddd <- data.frame(number = 1:5, letter = letters[1:5])</pre>
row.names(ddd) <- c("Abe", "Betty", "Claire", "Don", "Ethel")</pre>
ddd # row.names affects how a data.frame prints
      number letter
           1
Abe
           2
                  b
Betty
           3
Claire
           4
                  d
Don
Ethel
           5
                  е
```

More interestingly, it is possible to reset just individual names with the following syntax.

```
# misspelled a name, let's fix it
row.names(ddd)[2] <- "Bette"
row.names(ddd)

[1] "Abe" "Bette" "Claire" "Don" "Ethel"</pre>
```

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

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

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

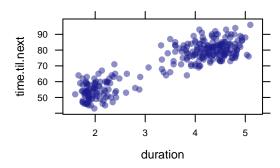
```
names(faithful) <- c("duration", "time.til.next")
head(faithful, 3)

duration time.til.next
1  3.60     79</pre>
```

Teaching Tip It's a good idea to start teach good practices for choice of variable names from day one.

```
2
      1.80
                        54
3
      3.33
                        74
```

```
xyplot(time.til.next ~ duration, alpha = 0.5, data = faithful)
```



If the variable containing a data frame is modified or used to store a different object, the original data from the package can be recovered using data().

```
data(faithful)
head(faithful, 3)
  eruptions waiting
       3.60
1
                 79
2
       1.80
                 54
3
       3.33
                 74
```

If we want to rename a variable, we can do this using names (). For example, perhaps we want to rename educ (the second column) to education.

```
names(CPS85)[2] <- "education"</pre>
CPS85[1, 1:4]
  wage education race sex
    9
               10
                     W
```

If we don't know the column number (or generally to make our code clearer), a few more keystrokes produces:

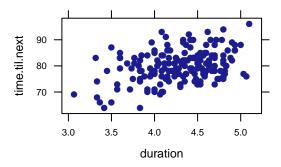
```
names(CPS85)[names(CPS85) == "education"] <- "educ"
CPS85[1, 1:4]

wage educ race sex
1  9  10  W  M</pre>
```

## 12.4 Creating subsets

We can also use subset() to reduce the size of a data set by selecting only certain rows.

```
data(faithful)
names(faithful) <- c("duration", "time.til.next")
# any logical can be used to create subsets
faithfulLong <- subset(faithful, duration > 3)
xyplot(time.til.next ~ duration, data = faithfulLong)
```



Of course, if all we want to do is produce a graph, there is no reason to create a new data frame. The plot above could also be made with:

CAUTION!
Unfortunately, not all functions in R support the subset=() or data=options.

```
xyplot(time.til.next ~ duration, subset=duration > 3,
  data=faithful)
```

## 12.5 Sorting data frames

Data frames can be sorted using the order() function.

```
head(faithful, 3)
  duration time.til.next
     3.60
1
                    79
2
     1.80
                     54
3
     3.33
                      74
sorted = faithful[order(faithful$duration), ]
head(sorted, 3)
    duration time.til.next
19
        1.60
58
        1.67
                        64
        1.70
                        59
115
```

## Merging datasets

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

```
require(fastR)
fusion1 = fusion1[order(fusion1$id), ]
head(fusion1, 3)
              marker markerID allele1 allele2 genotype
786 1002 RS12255372
                            1
                                    3
                                             3
                                                     GG
                                    3
                                             3
2319 1009 RS12255372
                            1
                                                     GG
```

```
384 1012 RS12255372
                                                    GG
     Adose Cdose Gdose Tdose
         0
               0
                     2
                           0
786
                     2
2319
         0
               0
               0
                           0
384
         0
head(pheno, 3)
          t2d bmi sex age smoker chol waist weight
1 1002
                     F 70.8 former 4.57 112.0
          case 32.9
                                                 85.6
2 1009
                     F 53.9 never 7.32 93.5
                                                77.4
          case 27.4
3 1012 control 30.5
                     M 53.9 former 5.02 104.0
                                                94.6
 height
          whr sbp dbp
    161 0.987 135 77
1
2
     168 0.940 158 88
3
    176 0.933 143 89
```

```
# merge fusion1 and pheno keeping only id's that are in both
fusion1m <- merge(fusion1, pheno, by.x='id', by.y='id',</pre>
  all.x=FALSE, all.y=FALSE)
head(fusion1m, 3)
    id
           marker markerID allele1 allele2 genotype Adose
1 1002 RS12255372
                         1
                                 3
                                         3
                                                 GG
                                                        0
2 1009 RS12255372
                         1
                                 3
                                         3
                                                 GG
                                                        0
                                 3
3 1012 RS12255372
                         1
                                         3
                                                 GG
  Cdose Gdose Tdose
                       t2d bmi sex age smoker chol waist
      0
            2
                       case 32.9 F 70.8 former 4.57 112.0
1
                  0
2
      0
            2
                  0
                                  F 53.9 never 7.32 93.5
                       case 27.4
3
      0
            2
                  0 control 30.5
                                   M 53.9 former 5.02 104.0
  weight height
                  whr sbp dbp
   85.6
           161 0.987 135
                          77
  77.4
           168 0.940 158 88
3
    94.6
           176 0.933 143 89
```

In this case, since the values are the same for each data frame, we could collapse by .x and by .y to by and collapse all .x and all .y to all. The first of these specifies which column(s) to use to identify matching cases. The second indicates whether cases in one data frame that do not appear in the other should be kept (TRUE)

or dropped (filling in NA as needed) or dropped from the merged data frame.

Now we are ready to begin our analysis.

```
tally(~t2d + genotype, data = fusion1m)
       genotype
t2d
          GG GT TT Total
         737 375 48 1160
 case
 control 835 309
                  27 1171
 Total 1572 684
                  75 2331
```

#### Slicing and dicing 12.7

The reshape() function 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 data frame, 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
             275 14.4 12.8 11.0 8.5
3 1953

      4 1954
      240 13.0 10.8 10.5 7.5

      5 1955
      325 13.5 14.0 11.8 10.0

            280 13.4 12.1 11.0 8.2
273 13.3 11.9 10.2 9.4
248 13.0 10.1 11.8 8.6
6 1956
7 1957
8 1958
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 <-</pre>
  reshape(traffic[,-2], idvar="year",
  ids=row.names(traffic),
  times=names(traffic)[3:6],
  timevar="state",
  varying=list(names(traffic)[3:6]),
  v.names="deathRate",
  direction="long")
head(longTraffic)
       year state deathRate
1951.ny 1951
                     13.9
              ny
1952.ny 1952 ny
                       13.8
1953.ny 1953
             ny
                       14.4
1954.ny 1954 ny
                      13.0
1955.ny 1955 ny
                      13.5
1956.ny 1956
               ny
                       13.4
```

And now we can reformat the other way, this time having all data for a given state form a row in the data frame.

```
stateTraffic <- reshape(longTraffic, direction='wide',</pre>
  v.names="deathRate", idvar="state", timevar="year")
stateTraffic
       state deathRate.1951 deathRate.1952 deathRate.1953
1951.ny
                      13.9
                                     13.8
                                                    14.4
        ny
                                    10.8
                                                    12.8
1951.cn
        cn
                      13.0
1951.ma
                       10.2
                                    10.0
                                                    11.0
                        8.0
1951.ri
                                      8.5
                                                     8.5
       deathRate.1954 deathRate.1955 deathRate.1956
               13.0
                              13.5
1951.ny
                                             13.4
1951.cn
                 10.8
                               14.0
                                              12.1
                 10.5
1951.ma
                               11.8
                                              11.0
1951.ri
                 7.5
                               10.0
                                               8.2
       deathRate.1957 deathRate.1958 deathRate.1959
                13.3
                              13.0
1951.ny
                                             12.9
1951.cn
                 11.9
                               10.1
                                              10.0
                 10.2
1951.ma
                               11.8
                                              11.0
1951.ri
                  9.4
                                 8.6
                                               9.0
```

In simpler cases, stack() or unstack() may suffice. The Hmisc package also provides reShape() as an alternative to reshape().

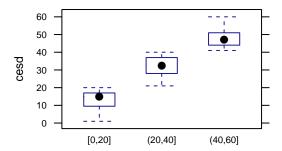
#### 12.8 Derived variable creation

A number of functions help facilitate the creation or recoding of variables.

### 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)
min Q1 median Q3 max mean
                             sd
                                  n missing
  1 25
            34 41 60 32.8 12.5 453
HELPrct = transform(HELPrct, cesdcut = cut(cesd,
 breaks=c(0, 20, 40, 60), include.lowest=TRUE))
bwplot(cesd ~ cesdcut, data=HELPrct)
```



## 12.8.2 Reordering factors

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

```
tally(~ substance, data=HELPrct)
alcohol cocaine heroin Total
   177
         152 124
                         453
coef(lm(cesd ~ substance, data=HELPrct))
    (Intercept) substancecocaine substanceheroin
         34.373
                -4.952
                               0.498
HELPrct = transform(HELPrct, subnew = relevel(substance,
  ref="heroin"))
coef(lm(cesd ~ subnew, data=HELPrct))
  (Intercept) subnewalcohol subnewcocaine
      34.871 -0.498
                            -5.450
```

## 12.9 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 <code>mosaic</code> package includes all n=470 subjects enrolled at baseline (including the n=17 subjects with some missing data who were not included in <code>HELPrct</code>).

```
Mean :32.9 Mean : 1.87 Mean :35.7
3rd Qu.:41.0 3rd Qu.: 1.00 3rd Qu.:41.0
Max. :60.0 Max. :21.00 Max. :45.0
           NA's :2
                       NA's :14
                        substance
   mcs
            pcs
Min. : 6.8 Min. :14.1 alcohol:185
Median :28.6 Median :48.9 heroin :128
Mean :31.5 Mean :48.1 missing: 1
3rd Qu.:40.6 3rd Qu.:57.0
Max. :62.2 Max. :74.8
NA's :2 NA's :2
favstats(~ mcs, data=smaller)
min Q1 median Q3 max mean sd n missing
6.76 21.7 28.6 40.6 62.2 31.5 12.8 468 2
with(smaller, sum(is.na(mcs)))
[1] 2
nomiss = na.omit(smaller)
dim(nomiss)
[1] 453 6
favstats(~ mcs, data=nomiss)
min Q1 median Q3 max mean sd n missing
6.76 21.8 28.6 40.9 62.2 31.7 12.8 453 0
```

# Health Evaluation and Linkage to Primary Care (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 (Ro1-AA10870, Samet PI) and National Institute on Drug Abuse (Ro1-DA10019, Samet PI). The details of the randomized trial along with the results from a series of additional analyses have been published<sup>1</sup>.

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 Services.

The mosaic package contains several forms of the de-identified HELP dataset. We will focus on HELPrct, which contains 27 variables for the 453 subjects with minimal missing data, primarily at baseline.

<sup>1</sup> J. H. Samet, M. J. Larson, N. J. Horton, K. Doyle, M. Winter, and R. Saitz. Linking alcohol and drug dependent adults to primary medical care: A randomized controlled trial of a multidisciplinary health intervention in a detoxification unit. Addiction, 98(4):509-516, 2003; J. Liebschutz, J. B. Savetsky, R. Saitz, N. J. Horton, C. Lloyd-Travaglini, and J. H. Samet. The relationship between sexual and physical abuse and substance abuse consequences. Journal of Substance Abuse Treatment, 22(3):121-128, 2002; and S. G. Kertesz, N. J. Horton, P. D. Friedmann, R. Saitz, and J. H. Samet. Slowing the revolving door: stabilization programs reduce homeless persons' substance use after detoxification. Journal of Substance Abuse Treatment, 24(3):197-207, 2003

Variables included in the HELP dataset are described in Table 13.1. More information can be found here<sup>2</sup>. A copy of the study instruments can be found at: http://www.math.smith.edu/help.

 $^{2}$  N. J. Horton and K. Kleinman. Using R for Data Management, Statistical Analysis, and Graphics. Chapman & Hall, 1st edition, 2011

Table 13.1: Annotated description of variables in the HELPrct dataset

VARIABLE	DESCRIPTION (VALUES)	NOTE	
age	age at baseline (in years) (range 19–60)		
anysub	use of any substance post-detox	see also daysanysub	
cesd	Center for Epidemiologic Studies De-		
	pression scale (range o-60, higher scores		
	indicate more depressive symptoms)		
d1	how many times hospitalized for medical		
	problems (lifetime) (range o-100)		
daysanysub	time (in days) to first use of any substance	see also any substatus	
	post-detox (range 0–268)		
dayslink	time (in days) to linkage to primary care	see also linkstatus	
	(range 0–456)		
drugrisk	Risk-Assessment Battery (RAB) drug risk	see also sexrisk	
	score (range 0–21)		
e2b	number of times in past 6 months entered		
	a detox program (range 1–21)		
female	gender of respondent (o=male, 1=female)		
g1b	experienced serious thoughts of suicide		
	(last 30 days, values 0=no, 1=yes)		
homeless	1 or more nights on the street or shelter in		
	past 6 months (o=no, 1=yes)		
i1	average number of drinks (standard units)	see also i2	
	consumed per day (in the past 30 days,		
	range 0–142)		
i2	maximum number of drinks (standard	see also i1	
	units) consumed per day (in the past 30		
	days range 0–184)		
id	random subject identifier (range 1-470)		
indtot	Inventory of Drug Use Consequences		
	(InDUC) total score (range 4-45)		
linkstatus	post-detox linkage to primary care (o=no,	see also dayslink	
	1=yes)		
mcs	SF-36 Mental Component Score (range	see also pcs	
	7-62, higher scores are better)		
pcs	SF-36 Physical Component Score (range	see also mcs	
	14-75, higher scores are better)		

pss_fr	perceived social supports (friends, range	
	0–14)	
racegrp	race/ethnicity (black, white, hispanic or	
	other)	
satreat	any BSAS substance abuse treatment at	
	baseline (o=no, 1=yes)	
sex	sex of respondent (male or female)	
sexrisk	Risk-Assessment Battery (RAB) sex risk	see also drugrisk
	score (range 0–21)	
substance	primary substance of abuse (alcohol, co-	
	caine or heroin)	
treat	randomization group (randomize to HELP	
	clinic, no or yes)	

Notes: Observed range is provided (at baseline) for continuous variables.

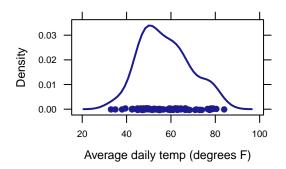
## Exercises and Problems

- **2.1** Using the HELPrct dataset, create side-by-side boxplots of the CESD scores by substance abuse group, just for the male subjects, with an overlaid normal density.
- **4.1** Using the HELPrct 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.
- **9.1** 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.

```
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)
```



- **9.2** The RailTrail dataset also includes a variable called cloudcover. Describe the distribution of this variable in terms of its center, spread and shape.
- 9.3 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.
- **9.4** 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?
- **9.5** 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?
- **9.6** Use overlapping densityplots to compare the distribution of volume by day type in the RailTrail dataset. What do you conclude?
- 9.7 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 relation-

ship?

- 9.8 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?
- **9.9** 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.

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

- 9.10 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).
- **9.11** Using the multiple regression model, generate a histogram (with overlaid normal density) to assess the normality of the residuals.
- 9.12 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.
- **9.13** 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.

- **10.1** Generate a sample of 1000 exponential random variables with rate parameter equal to 2, and calculate the mean of those variables.
- **10.2** Find the median of the random variable X, if it is exponentially distributed with rate parameter 10.
- **11.1** 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.
- **11.2** 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  $\alpha = 0.05$ ).
- **12.1** Using faithful data frame, make a scatter plot of eruption duration times vs. the time since the previous eruption.
- **12.2** 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.

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

Tidy up your data frame by dropping any columns that are redundant or that you just don't want to have in your final data frame.

# 15 Bibliography

- [HK11] N. J. Horton and K. Kleinman. *Using R for Data Management, Statistical Analysis, and Graphics*. Chapman & Hall, 1st edition, 2011.
- [KHF<sup>+</sup>o<sub>3</sub>] S. G. Kertesz, N. J. Horton, P. D. Friedmann, R. Saitz, and J. H. Samet. Slowing the revolving door: stabilization programs reduce homeless persons' substance use after detoxification. *Journal of Substance Abuse Treatment*, 24(3):197–207, 2003.
- [LSS<sup>+</sup>02] J. Liebschutz, J. B. Savetsky, R. Saitz, N. J. Horton, C. Lloyd-Travaglini, and J. H. Samet. The relationship between sexual and physical abuse and substance abuse consequences. *Journal of Substance Abuse Treatment*, 22(3):121–128, 2002.
- [MMo7] D. S. Moore and G. P. McCabe. *Introduction to the Practice of Statistics*. W.H.Freeman and Company, 6th edition, 2007.
- [NT10] D. Nolan and D. Temple Lang. Computing in the statistics curriculum. *The American Statistician*, 64(2):97–107, 2010.
- [RS13] Fred Ramsey and Dan Schafer. *Statistical Sleuth: A Course in Methods of Data Analysis*. Cengage, 3rd edition, 2013.
- [SLH<sup>+</sup>03] J. H. Samet, M. J. Larson, N. J. Horton, K. Doyle, M. Winter, and R. Saitz. Linking alcohol and drug dependent adults to primary medical care: A randomized controlled trial of a multidisciplinary health intervention in a detoxification unit. *Addiction*, 98(4):509–516, 2003.
  - [Tufo1] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT, 2nd edition, 2001.