Nicholas J. Horton
Daniel T. Kaplan
Randall Pruim

A Compendium of Commands to Teach Statistics with R

Project MOSAIC

Contents

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 USCOTS 2011 and eCOTS 2014. We organized these workshops to help instructors integrate R (as well as some related technologies) into their statistics courses at all levels. We received great feedback and many wonderful ideas from the participants and those that we've shared this with since the workshops.

We present an approach to teaching introductory and intermediate statistics courses that is tightly coupled with computing generally and with R and RStudio in particular. These activities and examples are intended to highlight a modern approach to statistical education that focuses on modeling, resampling based inference, and multivariate graphical techniques. A secondary goal is to facilitate computing with data through use of small simulation studies and appropriate statistical analysis workflow. This follows the philosophy outlined by Nolan and Temple Lang[1].

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

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

A Work in Progress

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`.

CAUTION!
Despite our best efforts, you WILL find bugs both in this document and in our code. Please let us know when you encounter them so we can call in the exterminators.

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 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 to develop their own teaching materials.

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

R, RStudio and R Packages

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

RStudio is an integrated development environment (IDE) that facilitates use of R for both novice and expert users. We have adopted it as our standard teaching environment because it dramatically simplifies the use of R for instructors and for students. 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 users via the Internet.

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

Marginal Notes

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

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

Document Creation

This document was created on August 22, 2014, using `knitr` and R version 3.1.1 Patched (2014-08-16 r66408).

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 RMarkdown, which produces PDF, HTML, or Word files 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, USCOTS 2013, eCOTS 2014, and ICOTS 9 workshops on

Teaching Statistics Using R *and* RStudio, our 2010 Project MOSAIC kickoff workshop at the Institute for Mathematics and its Applications, and our *Modeling: Early and Often in Undergraduate Calculus* AMS PREP workshops offered in 2012, 2013, and 2015.

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

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

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

Statistical Computation, Computational Statistics, and Data Science

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. This is a key capacity of **data science**, defined as the ability to use data to answer questions and communicate those results.

Our students need to see aspects of computation and data science early and often to develop deeper skills. Establishing precursors in introductory courses will help them get started.

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

Among the various statistical software packages available, R is becoming increasingly popular. The recent addition of RStudio has made R both more powerful and more accessible. Because R and RStudio are free, they have become widely used in research and industry. Training in R and RStudio is often seen as an important additional skill that a statistics course can develop. Furthermore, an increasing number of instructors are using R for their own statistical work, so it is natural for them to use it in their teaching as well. At the same time, the development of R and of RStudio (an optional interface and integrated development environment for R) are making it easier and easier to get started with R.

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.

1
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 clinical 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 Rfor introductory statistics courses, and the `mosaicData` package which includes a number of data sets. A short summary of the R commands needed to teach introductory statistics can be found in the mosaic package vignette (`http://cran.r-project.org/web/packages/mosaic/vignettes/mosaic-resources.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*[1] (see `http://www.amherst.edu/~nhorton/ips6e`), the second and third editions of the *Statistical Sleuth*[2] (see `http://www.amherst.edu/`

[1] D. S. Moore and G. P. McCabe. *Introduction to the Practice of Statistics*. W.H.Freeman and Company, 6th edition, 2007

[2] Fred Ramsey and Dan Schafer. *Statistical Sleuth: A Course in Methods of Data Analysis*. Cengage, 2nd edition, 2002

~nhorton/sleuth), and *Statistics: Unlocking the Power of Data* by Lock et al (see https://github.com/rpruim/Lock5withR).

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

```
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)
require(mosaicData)
```

TEACHING TIP
RStudio features a simplified package installation tab (on the bottom right panel).

Using Markdown or knitr/LaTeX requires that the markdown package be installed on your system.

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

We typically introduce students to RMarkdown very early, requiring students to use it for assignments and reports[3].

Depending on the level of the course, students can use either of these for homework and projects.

[3] Ben Baumer, Mine Çetinkaya Rundel, Andrew Bray, Linda Loi, and Nicholas J. Horton. R Markdown: Integrating a reproducible analysis tool into introductory statistics. *Technology Innovations in Statistics Education*, 8(1):281–283, 2014

TEACHING TIP
The knitr/LaTeX system allows users to combine R and LaTeX in the same document. The reward for learning this more complicated system is much finer control over the output format.

2
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, interquartile range (IQR) as well as arbitrary quantiles. We will illustrate these using the CESD (Center for Epidemiologic Studies–Depression) measure of depressive symptoms (which takes on values between 0 and 60, with higher scores indicating more depressive symptoms).

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

```
require(mosaic)
require(mosaicData)
options(digits = 3)
mean(~cesd, data = HELPrct)

[1] 32.8
```

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 commands using variants of this modeling language throughout this document. Those already familiar with R may be surprised by the form of this command.

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

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

[1] 32.8

mean(HELPrct$cesd)

[1] 32.8
```

Digging Deeper
The *Start Modeling with R* companion book will be helpful if you are unfamiliar with the modeling language. The *Start Teaching with R* also provides useful guidance in getting started.

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.

CAUTION!
Not all commands have been upgraded to support the formula interface. For these functions, variables within dataframes must be accessed using `with()` or the $ operator.

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

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

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

 2.5% 97.5%
  6.3  55.0
```

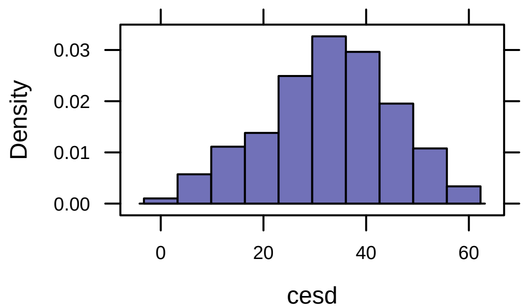
Finally, the `favstats()` function in the `mosaic` package provides a concise summary of many useful statistics.

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

 min Q1 median Q3 max mean   sd   n missing
   1 25     34 41  60 32.8 12.5 453       0
```

2.2 *Graphical summaries*

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

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



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

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


female   male
   107    346

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


female   male
  23.6   76.4
```

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

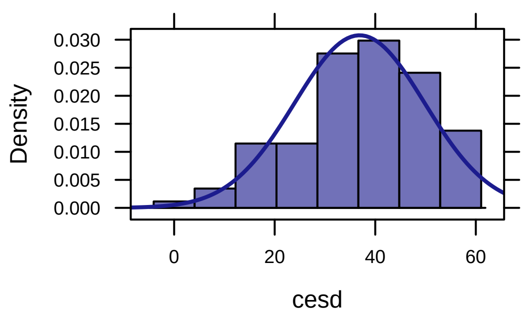
CAUTION!
Note that the tests for equality use *two* equal signs

```
female <- filter(HELPrct, sex == "female")
male <- filter(HELPrct, sex == "male")
with(female, stem(cesd))


  The decimal point is 1 digit(s) to the right of the |

  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
```

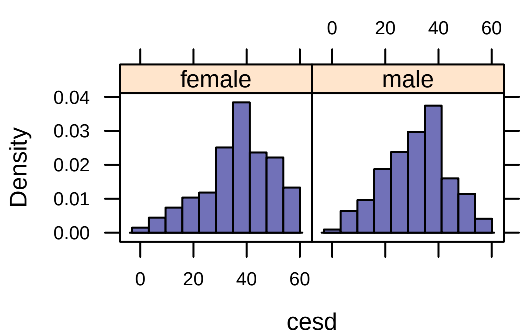
Subsets can also be generated and used “on the fly" (this time including an overlaid normal density):

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



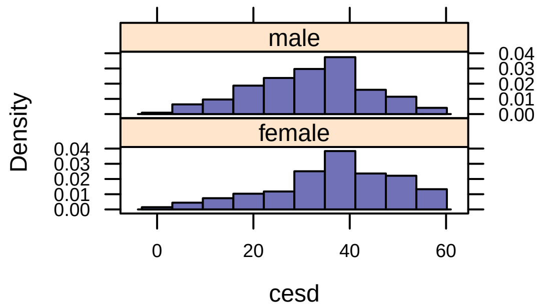
Alternatively, we can make side-by-side plots to compare multiple subsets.

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



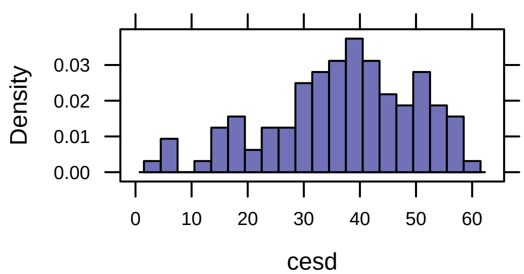
The layout can be rearranged.

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



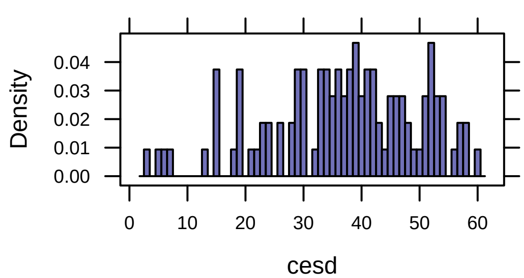
We can control the number of bins in a number of ways. These can be specified as the total number.

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



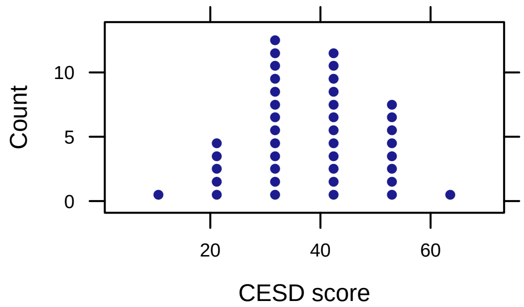
The width of the bins can be specified.

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



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

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



2.3 Density curves

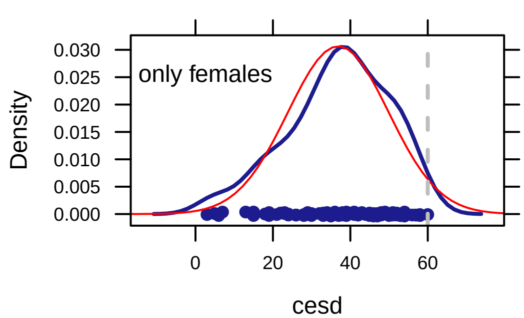
One disadvantage of histograms is that they can be sensitive to the choice of the number of bins. Another display to consider is a density curve.

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 and colors 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).

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

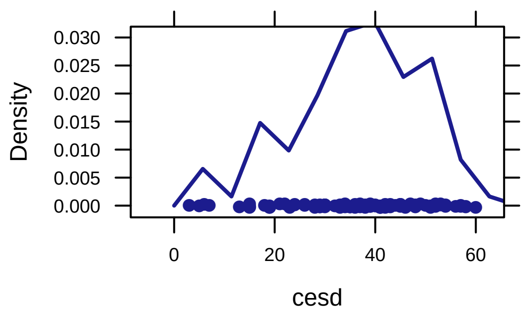
```
densityplot(~ cesd, data=female)
ladd(grid.text(x=0.2, y=0.8, 'only females'))
ladd(panel.mathdensity(args=list(mean=mean(cesd),
  sd=sd(cesd)), col="red"), data=female)
ladd(panel.abline(v=60, lty=2, lwd=2, col="grey"))
```



2.4 *Frequency polygons*

A third option is a frequency polygon, where the graph is created by joining the midpoints of the top of the bars of a histogram.

```
freqpolygon(~ cesd, data=female)
```

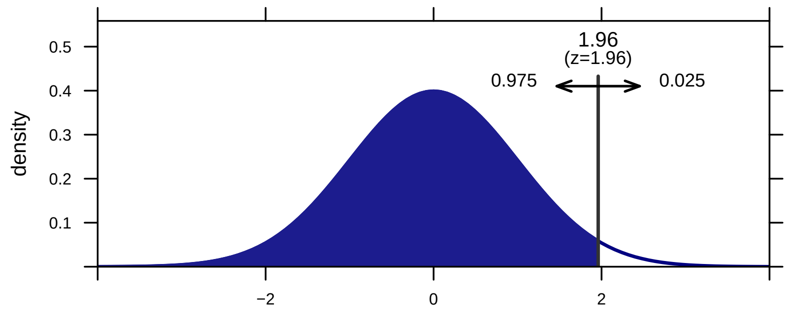


2.5 *Normal distributions*

The most famous density curve is a normal distribution. The `xpnorm()` function displays the probability that a random variable is less than the first argument, for a normal distribution with mean given by the second argument and standard deviation by the third. More information about probability distributions can be found in section 10.

`x` is for eXtra.

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


If X ~ 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
```



2.6 *Inference for a single sample*

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

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


One Sample t-test

data:  data$cesd
t = 29.3, df = 106, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 34.4 39.4
sample estimates:
mean of x
     36.9

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

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

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

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

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

[1] 36.9
```

One resampling trial can be carried out:

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

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

[1] 34.9
```

Another will yield different results:

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

[1] 35.2
```

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

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

```
trials = do(1000) * mean(~cesd, data = resample(female))

Loading required package:  parallel

qdata(c(0.025, 0.975), ~result, data = trials)

      quantile     p
2.5%      34.5 0.025
97.5%     39.3 0.975
```

3
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
     209      244

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


homeless   housed    Total
     209      244      453

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


homeless   housed
    46.1     53.9

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


homeless   housed
   0.461    0.539
```

DIGGING DEEPER
The *Start Modeling with R* companion book will be helpful if you are unfamiliar with the modeling language. The *Start Teaching with R* also provides useful guidance in getting started.

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.

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


Exact binomial test

data:  x and n
number of successes = 209, number of trials = 453,
p-value = 0.1101
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.415 0.509
sample estimates:
probability of success
                 0.461
```

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

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


Exact binomial test

data:  n$(homeless == "homeless")
number of successes = 209, number of trials = 453,
p-value = 0.1101
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.415 0.509
sample estimates:
probability of success
                 0.461
```

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

```
names(result)

[1] "statistic"   "parameter"   "p.value"     "conf.int"
[5] "estimate"    "null.value"  "alternative" "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
                 0.461                    0.415
                 upper                    level
                 0.509                    0.950

pval(result)

p.value
   0.11
```

3.3 *The proportion test*

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

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


homeless   housed
     209      244

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
```

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 sometimes helpful for `lattice` graphics functions.

`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.

```
sample estimates:
    p
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.

Caution!
The `margins=FALSE` option is the default for the `tally()` function.

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


alcohol cocaine  heroin
   39.1    33.6    27.4

observed <- tally(~ substance, data=HELPrct)
observed


alcohol cocaine  heroin
    177     152     124
```

```
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
```

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

Teaching Tip
We don't have students do much if any manual calculations in our courses.

```
chisq <- sum((observed - expected)^2/(expected)); chisq

[1] 9.31

1 - pchisq(chisq, df=2)

[1] 0.00951
```

Alternatively, the `mosaic` package provides a version of `chisq.test()` with more verbose output.

```
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>
```

TEACHING TIP
The `pchisq()` function calculates the probability that a χ^2 random variable with `df()` degrees is freedom is less than or equal to a given value. Here we calculate the complement to find the area to the right of the observed Chi-square statistic.

`x` is for eXtra.

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

TEACHING TIP
Objects in the workspace that are no longer needed can be removed.

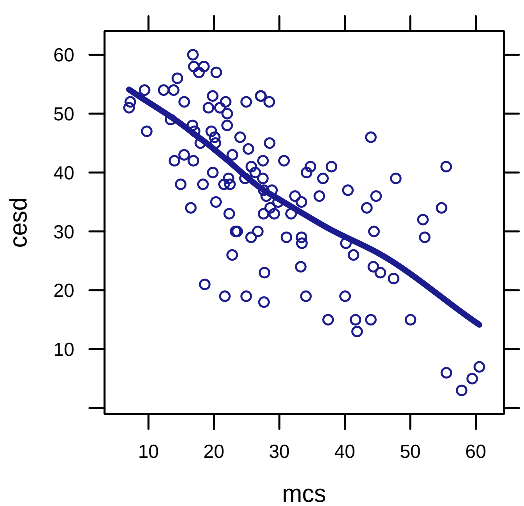
4 Two Quantitative Variables

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) for female subjects 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. This is added by specifying both points (using 'p') and a lowess smoother.

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

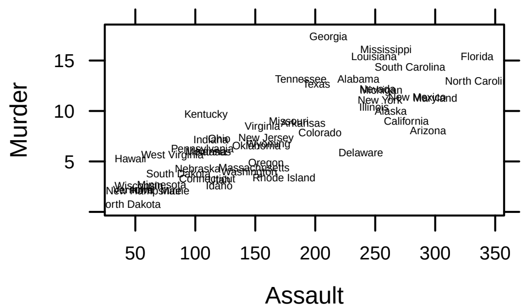


Digging Deeper

The *Start Modeling with R* companion book will be helpful if you are unfamiliar with the modeling language. The *Start Teaching with R* also provides useful guidance in getting started.

It's straightforward to plot something besides a character in a scatterplot. In this example, the `USArrests` can be used to plot the association between murder and assault rates, with the state name displayed. This requires a panel function to be written.

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



4.2 *Correlation*

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

```
cor(cesd, mcs, data = females)

[1] -0.674

smallHELP = select(females, cesd, mcs, pcs)
cor(smallHELP)

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

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

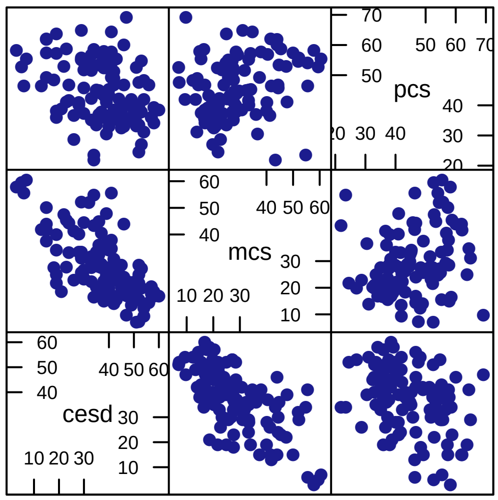
```
cor(cesd, mcs, method = "spearman", data = females)

[1] -0.666
```

4.3 Pairs plots

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

Teaching Tip
The GGally package has support for more elaborate pairs plots.

```
splom(smallHELP)
```



4.4 Simple linear regression

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

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

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

(Intercept)         mcs
     57.349      -0.707
```

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

It's important to pick good names for modeling objects. Here the output of `lm()` is saved as `cesdmodel`, which denotes that it is a regression model of depressive symptom scores.

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

(Intercept)         mcs
     57.349      -0.707

summary(cesdmodel)
```

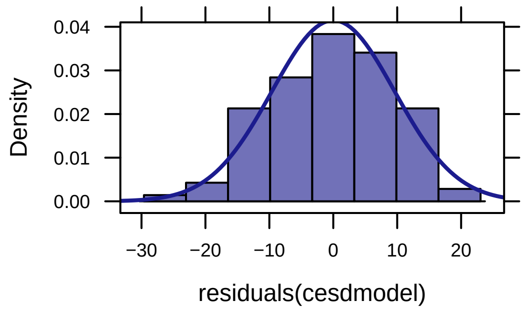
```
Call:
lm(formula = cesd ~ mcs, data = females)

Residuals:
    Min      1Q  Median      3Q     Max 
-23.202  -6.384   0.055   7.250  22.877 

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

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

coef(summary(cesdmodel))

            Estimate Std. Error t value Pr(>|t|)
(Intercept)   57.349     2.3806   24.09 1.42e-44
mcs           -0.707     0.0757   -9.34 1.81e-15

confint(cesdmodel)

             2.5 % 97.5 %
(Intercept) 52.628 62.069
mcs         -0.857 -0.557

rsquared(cesdmodel)

[1] 0.454
```

```
class(cesdmodel)

[1] "lm"
```

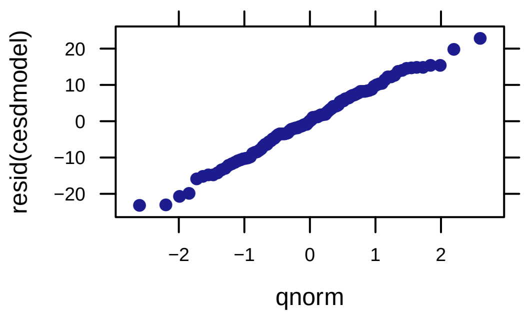
The return value from `lm()` is a linear model object. A number of functions can operate on these objects, as seen previously with `coef()`. The function `residuals()` returns a vector of the residuals.

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

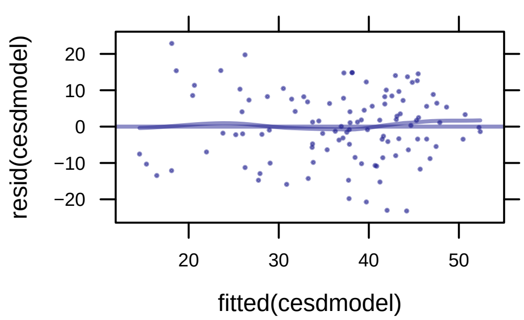
```
histogram(~residuals(cesdmodel), density = TRUE)
```




```
qqmath(~resid(cesdmodel))
```

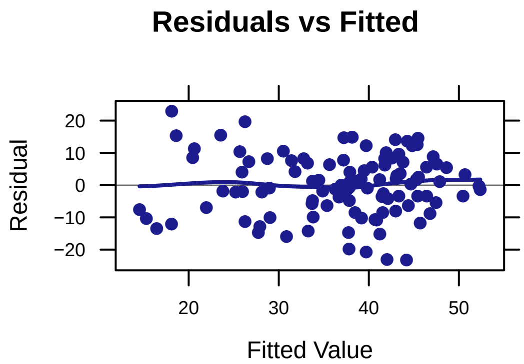



```
xyplot(resid(cesdmodel) ~ fitted(cesdmodel), type = c("p", "smooth",
    "r"), alpha = 0.5, cex = 0.3, pch = 20)
```



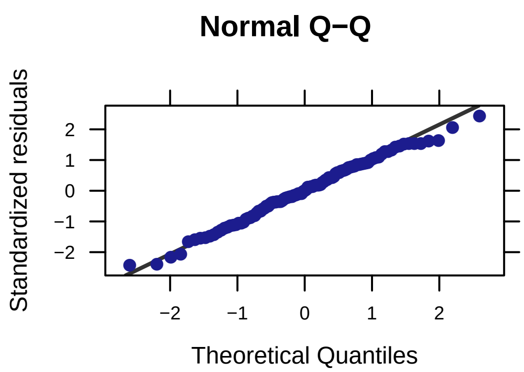
The `mplot()` function can facilitate creating a variety of useful plots, including the same residuals vs. fitted scatterplots, by specifying the `which=1` option.

```
mplot(cesdmodel, which = 1)
```



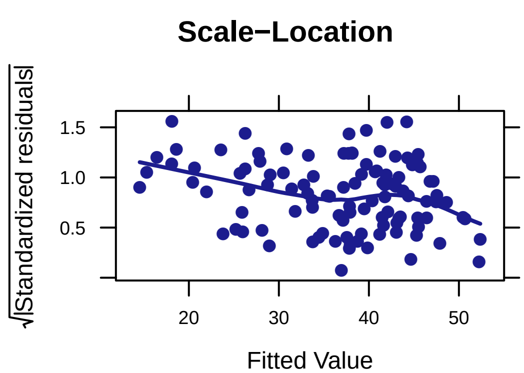
It can also generate a normal quantile-quantile plot (`which=2`),

```
mplot(cesdmodel, which = 2)
```



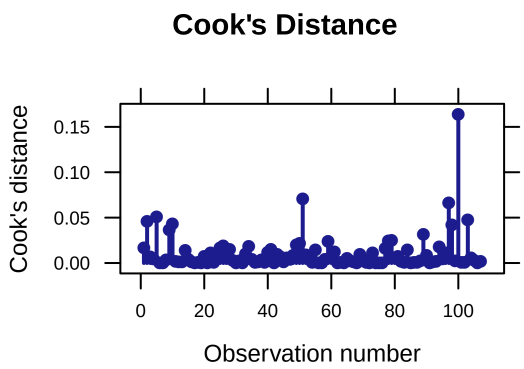
scale vs. location,

```
mplot(cesdmodel, which = 3)
```



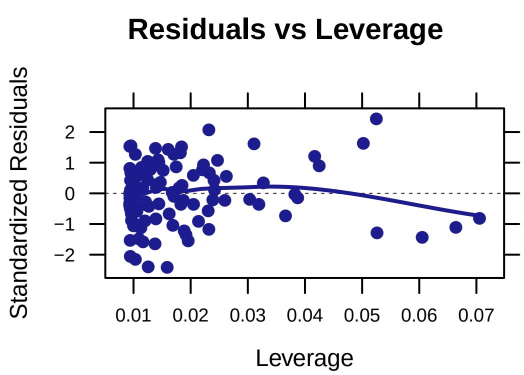
Cook's distance by observation number,

```
mplot(cesdmodel, which = 4)
```



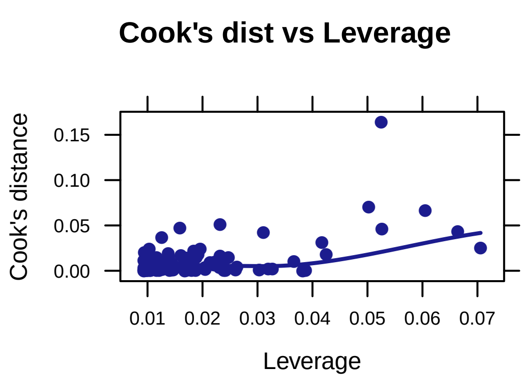
residuals vs. leverage, and

```
mplot(cesdmodel, which = 5)
```



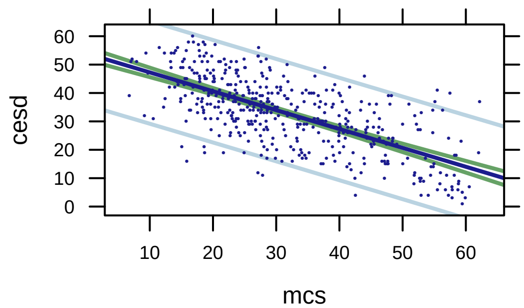
Cook's distance vs. leverage.

```
mplot(cesdmodel, which = 6)
```



Prediction bands can be added to a plot using the `panel.lmbands()` function.

```
xyplot(cesd ~ mcs, panel = panel.lmbands, cex = 0.2, band.lwd = 2,
    data = HELPrct)
```



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 also calculate column percentages:

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

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

We can calculate the odds ratio directly from the table:

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

[1] 0.625
```

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

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

[1] 0.625
```

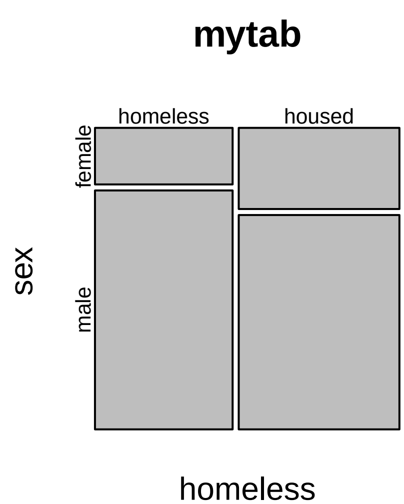
Graphical summaries of cross classification tables may be helpful in visualizing associations. Mosaic plots are one example, where the total area (all observations) is proportional to one. Here we see that males tend to be over-represented amongst the homeless subjects (as represented by the horizontal line which is higher for the homeless rather than the housed).

CAUTION!
The jury is still out regarding the utility of mosaic plots, relative to the low data to ink ratio . But we have found them to be helpful to reinforce understanding of a two way contingency table.

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)
```



5.2 *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 (observed, expected, contribution to statistic, and residual) 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 computationally straightforward for 2 by 2 tables. Options to help constrain the size of the problem for larger tables exist (see `?fisher.test()`).

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).

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

6
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
```

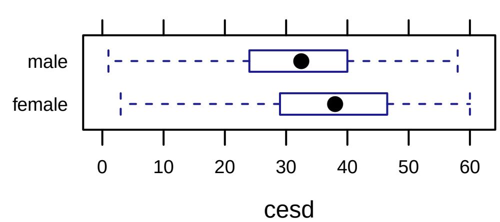
The `favstats()` function can provide more statistics by group.

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

  .group min Q1 median   Q3 max mean   sd   n missing
1 female   3 29   38.0 46.5  60 36.9 13.0 107       0
2   male   1 24   32.5 40.0  58 31.6 12.1 346       0
```

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

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

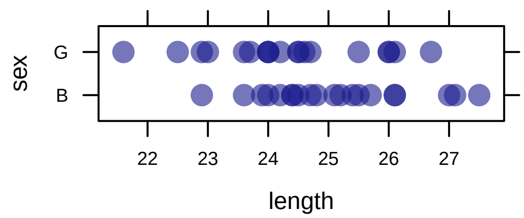
Although we usually put explanatory variables along the horizontal axis, page layout sometimes makes the other orientation preferable for these plots.



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.

```
xyplot(sex ~ length, KidsFeet, alpha = 0.6, cex = 1.4)
```



6.2 *A dichotomous predictor: two-sample t*

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

```
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 0
95 percent confidence interval:
 2.49 8.09
sample estimates:
mean in group female   mean in group male
                36.9                 31.6
```

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

We can repeat using the equal variance assumption.

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


Two Sample t-test
```

```
data:  cesd by sex
t = 3.88, df = 451, p-value = 0.00012
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 2.61 7.97
sample estimates:
mean in group female   mean in group male
                36.9                 31.6
```

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

```
summary(lm(cesd ~ sex, data = HELPrct))


Call:
lm(formula = cesd ~ sex, data = HELPrct)

Residuals:
   Min     1Q Median     3Q    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
sexmale        -5.29       1.36   -3.88  0.00012

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

Teaching Tip
While it requires use of the equal variance assumption, the `lm()` function is part of a much more flexible modeling framework (while `t.test()` is essentially a dead end).

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 2.6 to undertake a two-sided test comparing the ages at baseline by gender. First we calculate the observed difference in means:

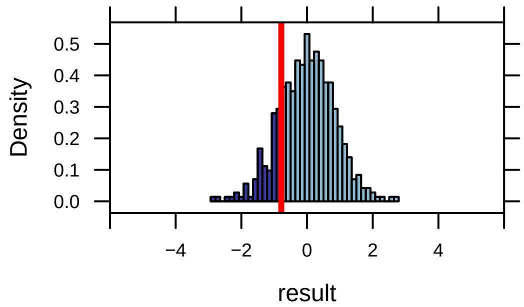
```
mean(age ~ sex, data = HELPrct)

female   male
  36.3   35.5

test.stat <- compareMean(age ~ sex, data = HELPrct)
test.stat

[1] -0.784
```

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

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

  result
1  -1.32

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

  result
1  0.966

do(3) * compareMean(age ~ shuffle(sex), data = HELPrct)

  result
1 -0.185
2  1.125
3 -0.209
```

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

```
rtest.stats = do(500) * compareMean(age ~ shuffle(sex),
  data=HELPrct)
favstats(~ result, data=rtest.stats)

   min     Q1 median    Q3  max   mean    sd   n missing
 -2.85 -0.564 0.0234 0.574 2.73 0.0057 0.846 500       0

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, lwd=3, col="red"))
```

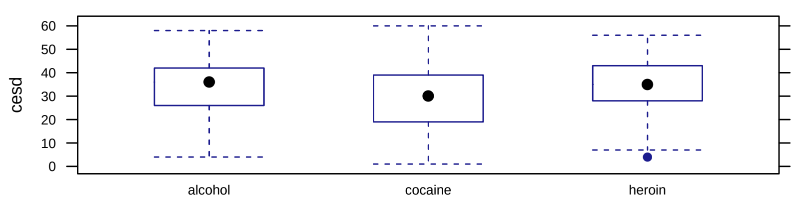


Here we don't see much evidence to contradict the null hypothesis that men and women have the same mean age in the population.

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
```

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

             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.

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

The `anova()` command can summarize models.

```
anova(modsubstance)

Analysis of Variance Table

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

It can also be used to formally compare two (nested) models.

```
anova(modintercept, modsubstance)

Analysis of Variance Table

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

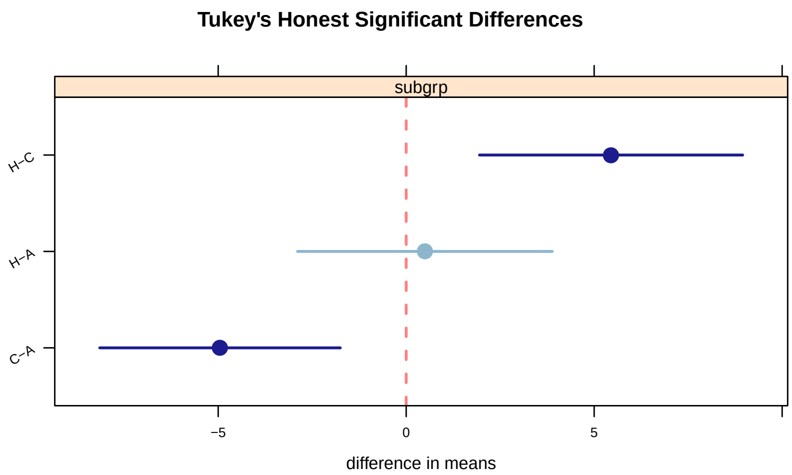
6.6 *Tukey's Honest Significant Differences*

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

```
favstats(cesd ~ substance, data = HELPrct)

   .group min Q1 median Q3 max mean   sd   n missing
1 alcohol   4 26     36 42  58 34.4 12.1 177       0
2 cocaine   1 19     30 39  60 29.4 13.4 152       0
3  heroin   4 28     35 43  56 34.9 11.2 124       0
```

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

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

Fit: aov(formula = x)

$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
```

```
mplot(HELPHSD)
```



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

7 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 argument `family`, which can take an option `link`. The `logit` link is the default link for the binomial family, so we don't need to specify it here. The more verbose usage would be `family=binomial(link=logit)`.

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


Call:
glm(formula = homeless ~ age + female, family = binomial, data = HELPrct)

Deviance Residuals:
   Min      1Q  Median      3Q     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
age          -0.0239     0.0124   -1.92    0.055
female        0.4920     0.2282    2.16    0.031

(Dispersion parameter for binomial family taken to be 1)

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

Number of Fisher Scoring iterations: 4

exp(coef(logitmod))

(Intercept)         age      female
      2.442       0.976       1.636

exp(confint(logitmod))
```

```
Waiting for profiling to be done...

            2.5 % 97.5 %
(Intercept) 1.008   5.99
age         0.953   1.00
female      1.050   2.57
```

We can compare two models (for multiple degree of freedom tests). For example, we might be interested in the association of homeless status and age for each of the three substance groups.

```
mymodsubage = glm((homeless=="homeless") ~ age + substance,
  family=binomial, data=HELPrct)
mymodage = glm((homeless=="homeless") ~ age, family=binomial,
  data=HELPrct)
summary(mymodsubage)


Call:
glm(formula = (homeless == "homeless") ~ age + substance, family = binomial,
    data = HELPrct)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.409  -1.001  -0.947   1.086   1.458

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

(Dispersion parameter for binomial family taken to be 1)

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

Number of Fisher Scoring iterations: 4

exp(coef(mymodsubage))

     (Intercept)              age substancecocaine
           0.950            1.010            0.473
 substanceheroin
           0.459

anova(mymodage, mymodsubage, test="Chisq")

Analysis of Deviance Table
```

```
Model 1: (homeless == "homeless") ~ age
Model 2: (homeless == "homeless") ~ age + substance
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1       451        622
2       449        608  2     14.3  0.00078
```

We observe that the cocaine and heroin groups are significantly less likely to be homeless than alcohol involved subjects, after controlling for age. (A similar result is seen when considering just homeless status and substance.)

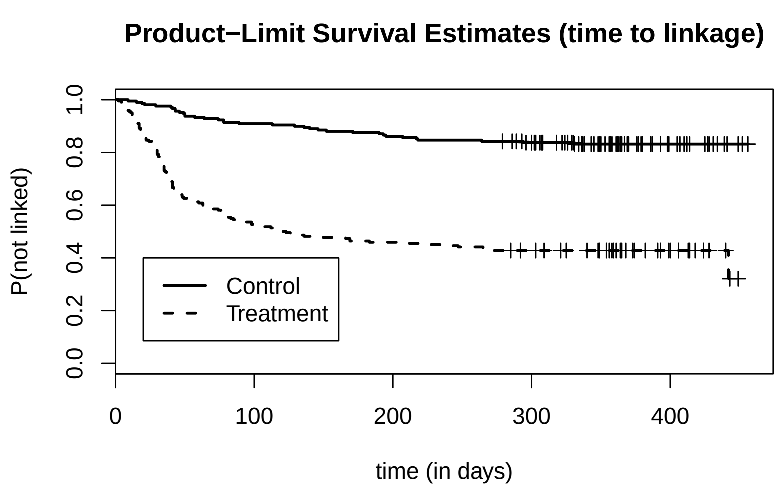
```
tally(~homeless | substance, format = "percent", margins = TRUE,
    data = HELPrct)

          substance
homeless   alcohol cocaine heroin
  homeless    58.2    38.8   37.9
  housed      41.8    61.2   62.1
  Total      100.0   100.0  100.0
```

8
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)")
```



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|)
age               0.00893   1.00897  0.01026  0.87     0.38
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
age                  1.009      0.991     0.989      1.03
substancecocaine     1.198      0.835     0.840      1.71
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
```

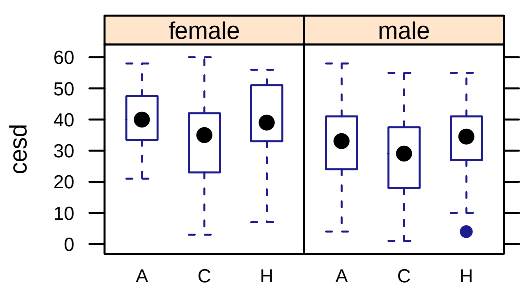
Neither age or substance group was significantly associated with linkage to primary care.

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.

```
median(cesd ~ substance | sex, data = HELPrct)

alcohol.female cocaine.female  heroin.female   alcohol.male
          40.0           35.0           39.0           33.0
  cocaine.male    heroin.male         female           male
          29.0           34.5           38.0           32.5

bwplot(cesd ~ subgrp | sex, data = HELPrct)
```

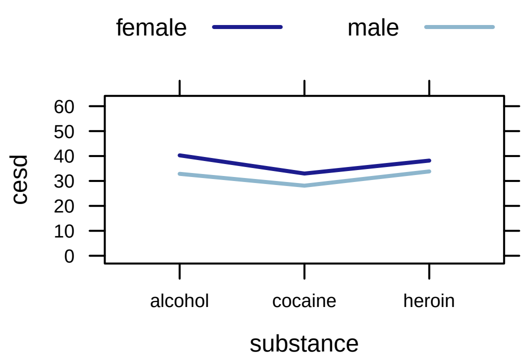



```
summary(aov(cesd ~ substance + sex, data = HELPrct))

             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       2   2704    1352    9.25 0.00012
sex             1   2569    2569   17.57 3.3e-05
substance:sex   2    146      73    0.50 0.60752
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,
  auto.key=list(columns=2, lines=TRUE, points=FALSE), type='a',
  data=HELPrct)
```



9.2 *Multiple regression*

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

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

We tend to introduce multiple linear regression early in our courses, as a purely descriptive technique, then return to it regularly. The motivation for this is described at length in the companion volume *Start Modeling with R*.

```
lmnointeract <- lm(cesd ~ mcs + age + sex, data = HELPrct)
summary(lmnointeract)


Call:
lm(formula = cesd ~ mcs + age + sex, data = HELPrct)

Residuals:
    Min      1Q  Median      3Q     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
mcs          -0.6548     0.0336  -19.50   <2e-16
age           0.0553     0.0556    1.00   0.3200
sexmale      -2.8993     1.0137   -2.86   0.0044

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

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

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


Call:
lm(formula = cesd ~ mcs + age + sex + mcs:sex, data = HELPrct)

Residuals:
    Min      1Q  Median      3Q     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
mcs          -0.7082     0.0712   -9.95   <2e-16
age           0.0549     0.0556    0.99    0.324
sexmale      -4.9421     2.6055   -1.90    0.058
mcs:sexmale   0.0687     0.0807    0.85    0.395

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

anova(lminteract)

Analysis of Variance Table

Response: cesd
           Df Sum Sq Mean Sq F value Pr(>F)
mcs         1  32918   32918  398.27 <2e-16
age         1    107     107    1.29 0.2563
sex         1    676     676    8.18 0.0044
mcs:sex     1     60      60    0.72 0.3952
Residuals 448  37028      83
```

```
anova(lmnointeract, lminteract)

Analysis of Variance Table

Model 1: cesd ~ mcs + age + sex
Model 2: cesd ~ mcs + age + sex + mcs:sex
  Res.Df   RSS Df Sum of Sq    F Pr(>F)
1    449 37088
2    448 37028  1      59.9 0.72    0.4
```

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

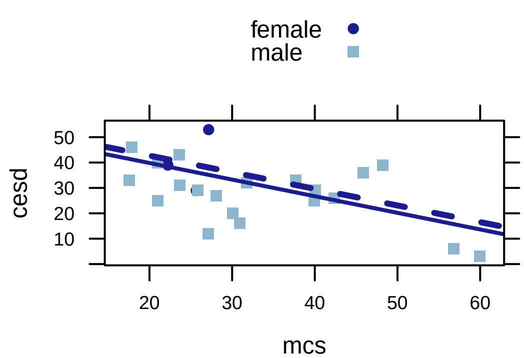
9.2.1 *Visualizing the results from the regression*

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 (no interaction) model.

```
lmfunction = makeFun(lmnointeract)
```

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

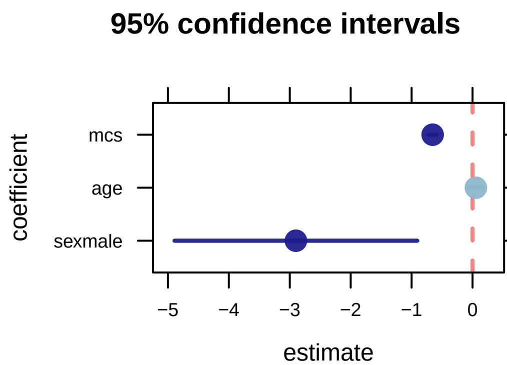
```
xyplot(cesd ~ mcs, groups=sex, auto.key=TRUE,
  data=filter(HELPrct, age==36))
plotFun(lmfunction(mcs, age=36, sex="male") ~ mcs,
  xlim=c(0, 60), lwd=2, ylab="predicted CESD", add=TRUE)
plotFun(lmfunction(mcs, age=36, sex="female") ~ mcs,
  xlim=c(0, 60), lty=2, lwd=3, add=TRUE)
```



9.2.2 *Coefficient plots*

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

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



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

9.2.3 *Residual diagnostics*

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

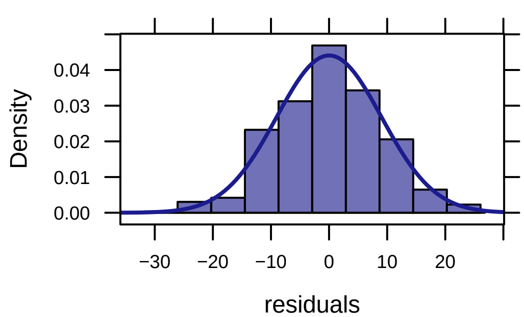
CAUTION!
Be careful when fitting regression models with missing values (see also section 12.10).

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

```
HELPrct = mutate(HELPrct, residuals = resid(lmnointeract),
  pred = fitted(lmnointeract))
```

Here we are adding two new variables into an existing dataset. It's often a good practice to give the resulting dataframe a new name.

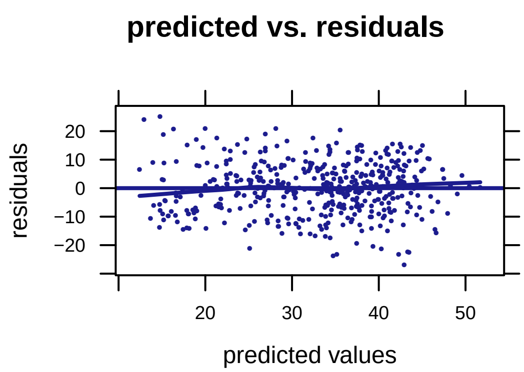
```
histogram(~ residuals, xlab="residuals", fit="normal",
  data=HELPrct)
```



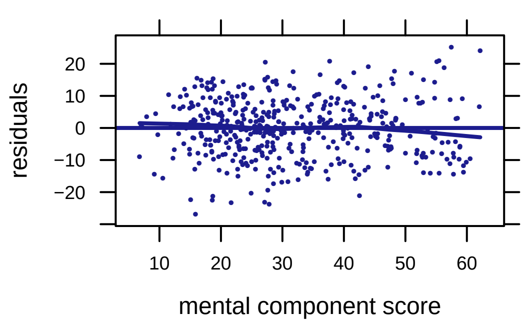
We can identify the subset of observations with extremely large residuals.

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

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

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




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



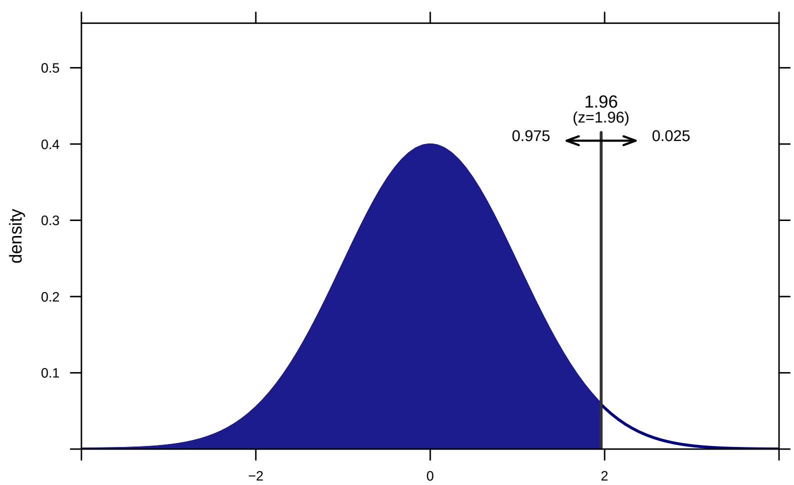
The assumptions of normality, linearity and homoscedasticity seem reasonable here.

10

Probability Distributions and Random Variables

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

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


If X ~ 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
```

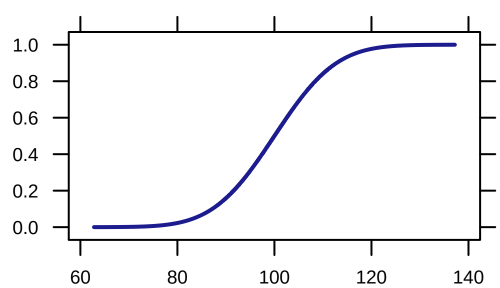
The following table displays the basenames for probability distributions available within base R. These functions can be prefixed by d to find the density function for the distribution, p to find the cumulative distribution function, q to find quantiles, and r to generate random draws. For example, to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

Distribution	Basename
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

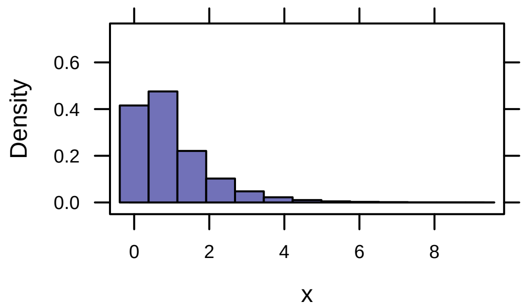
DIGGING DEEPER
The `fitdistr()` within the `MASS` package facilitates estimation of parameters for many distributions.

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

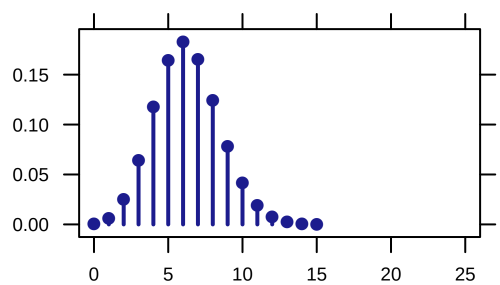
```
plotDist('norm', mean=100, sd=10, kind='cdf')
```



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




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



11
Power Calculations

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

We will compare the power of the sign test and the power of the test based on normal theory (one sample one sided t-test) assuming that σ is known. 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 $\alpha = .05$.

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.

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

     xvals   probs
[1,]    15 0.11476
[2,]    16 0.05388
[3,]    17 0.02164
[4,]    18 0.00732
[5,]    19 0.00204

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

[1] 17
```

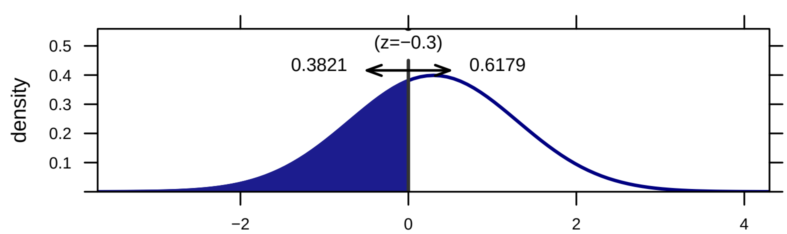
So we see that if we decide to reject when the number of positive values is 17 or larger, we will have an α level of 0.054, which is near the nominal value in the problem.

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

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

[1] 0.618
```

We can view this graphically using the command:

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


If X ~ 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 α level equal to 0.05388.

```
alpha <- 1-pbinom(16, size=25, prob=0.5); alpha

[1] 0.0539
```

First we find the rejection region.

```
n <- 25; sigma <- 1 # given
stderr <- sigma/sqrt(n)
zstar <- qnorm(1-alpha, mean=0, sd=1)
zstar

[1] 1.61

crit <- zstar*stderr
crit

[1] 0.322
```

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)
power

[1] 0.457
```

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 analytic (formula-based approach) yields a similar estimate to the value that we calculated directly.

Overall, we see that the t-test has higher power than the sign test, if the underlying data are truly normal.

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

12
Data Management

Teaching Tip
The *Start Teaching with R* book features an extensive section on data management, including use of the `read.file()` function to load data into R and RStudio.

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

Teaching Tip
The `dplyr` and `tidyr` packages provide an elegant approach to data management and facilitate the ability of students to compute with data. Hadley Wickham, author of the packages, suggests that there are six key idioms (or verbs) implemented within these packages that allow a large set of tasks to be accomplished: filter (keep rows matching criteria), select (pick columns by name), arrange (reorder rows), mutate (add new variables), summarise (reduce variables to values), and group by (collapse groups).

12.1 Adding new variables to a dataframe

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

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

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

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

```
head(irisSmall)

  Species Sepal.Length Length
1  setosa          5.1  (5,6]
2  setosa          4.9  (4,5]
3  setosa          4.7  (4,5]
4  setosa          4.6  (4,5]
5  setosa          5.0  (4,5]
6  setosa          5.4  (5,6]
```

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

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

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

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

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



  0   4
533   1
```

12.2 Dropping variables

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

```
names(CPS85)

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

CPS1 <- select(CPS85, select = -matches("workforce.years"))
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 a similar manner.

```
CPS1 <- select(CPS85, select = -matches("workforce.years|exper"))
```

12.3 *Renaming variables*

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

```
names(CPS85)

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

CPSnew = rename(CPS85, workforce = workforce.years)
names(CPSnew)

 [1] "wage"      "educ"      "race"      "sex"
 [5] "hispanic"  "south"     "married"   "exper"
 [9] "union"     "age"       "sector"    "workforce"
```

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

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

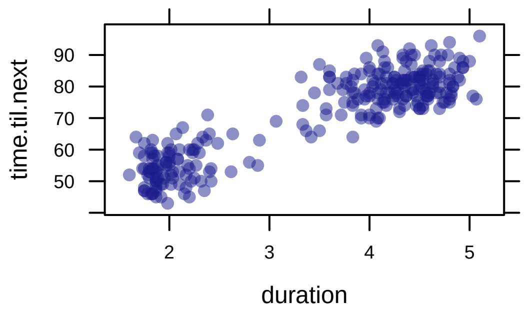
TEACHING TIP
It's a good idea to start teaching good practices for choice of variable names from day one.

```
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.

```
faithful = rename(faithful,
  duration = eruptions,
  time.til.next=waiting)
names(faithful)

[1] "duration"      "time.til.next"
```

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



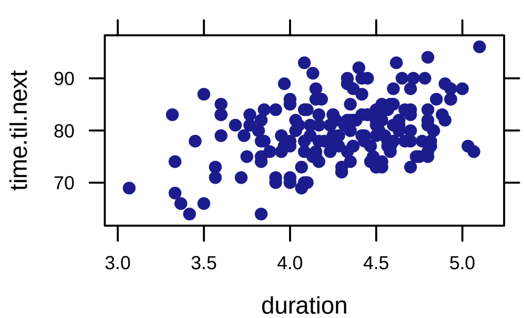
If the variable containing a dataframe is modified or used to store a different object, the original data from the package can be recovered using `data()`.

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

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

12.4 *Creating subsets of observations*

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

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



12.5 Sorting dataframes

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

```
head(faithful, 3)

  duration time.til.next
1     3.60            79
2     1.80            54
3     3.33            74

sorted = arrange(faithful, duration)
head(sorted, 3)

  duration time.til.next
1     1.60            52
2     1.67            64
3     1.70            59
```

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

12.6 Merging datasets

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

```
require(fastR)
require(dplyr)
fusion1 = arrange(fusion1, id)
head(fusion1, 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 1012 RS12255372        1       3       3       GG     0
  Cdose Gdose Tdose
1     0     2     0
2     0     2     0
3     0     2     0

head(pheno, 3)

    id     t2d  bmi sex  age smoker chol waist weight
1 1002    case 32.9   F 70.8 former 4.57 112.0   85.6
```

```
2 1009    case 27.4   F 53.9  never 7.32  93.5   77.4
3 1012 control 30.5   M 53.9 former 5.02 104.0   94.6
  height   whr sbp dbp
1    161 0.987 135  77
2    168 0.940 158  88
3    176 0.933 143  89
```

```
require(tidyr)
fusion1m <- inner_join(fusion1, pheno, by='id')
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 1012 RS12255372        1       3       3       GG     0
  Cdose Gdose Tdose     t2d  bmi sex  age smoker chol waist
1     0     2     0    case 32.9   F 70.8 former 4.57 112.0
2     0     2     0    case 27.4   F 53.9  never 7.32  93.5
3     0     2     0 control 30.5   M 53.9 former 5.02 104.0
  weight height   whr sbp dbp
1   85.6    161 0.987 135  77
2   77.4    168 0.940 158  88
3   94.6    176 0.933 143  89
```

Now we are ready to begin our analysis.

```
tally(~t2d + genotype, data = fusion1m)

         genotype
t2d        GG  GT  TT
  case    737 375  48
  control 835 309  27
```

12.7 *Slicing and dicing*

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

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

Teaching Tip
The vignettes that accompany the `tidyr` and `dplyr` packages feature a number of useful examples of common data manipulations.

```
traffic

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

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

```
longTraffic <- traffic %>%
  gather(state, deathRate, ny:ri)
head(longTraffic)

  year cn.deaths state deathRate
1 1951       265    ny      13.9
2 1952       230    ny      13.8
3 1953       275    ny      14.4
4 1954       240    ny      13.0
5 1955       325    ny      13.5
6 1956       280    ny      13.4
```

We can also reformat the other way, this time having all data for a given state form a row in the dataframe.

```
stateTraffic <- longTraffic %>%
  select(year, deathRate, state) %>%
  mutate(year=paste("deathRate.", year, sep="")) %>%
  spread(year, deathRate)
stateTraffic

  state deathRate.1951 deathRate.1952 deathRate.1953
1    ny           13.9           13.8           14.4
2    cn           13.0           10.8           12.8
3    ma           10.2           10.0           11.0
4    ri            8.0            8.5            8.5
  deathRate.1954 deathRate.1955 deathRate.1956
1           13.0           13.5           13.4
2           10.8           14.0           12.1
3           10.5           11.8           11.0
4            7.5           10.0            8.2
  deathRate.1957 deathRate.1958 deathRate.1959
1           13.3           13.0           12.9
2           11.9           10.1           10.0
```

```
3           10.2           11.8           11.0
4            9.4            8.6            9.0
```

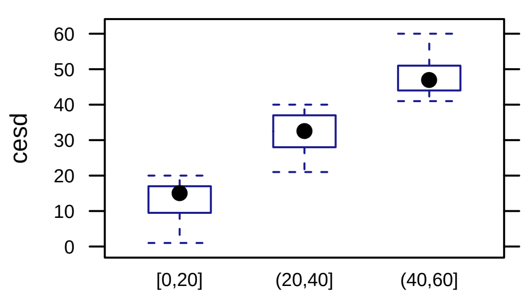
12.8 Derived variable creation

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

12.8.1 Creating categorical variable from a quantitative variable

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

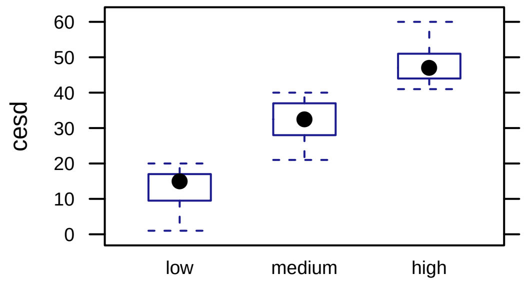
```
favstats(~ cesd, data=HELPrct)

 min Q1 median Q3 max mean   sd   n missing
   1 25     34 41  60 32.8 12.5 453       0

HELPrct = mutate(HELPrct, cesdcut = cut(cesd,
  breaks=c(0, 20, 40, 60), include.lowest=TRUE))
bwplot(cesd ~ cesdcut, data=HELPrct)
```



It might be preferable to give better labels.

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

```
HELPrct = mutate(HELPrct, cesdcut = cut(cesd,
  labels=c("low", "medium", "high"),
  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
    177     152     124

coef(lm(cesd ~ substance, data=HELPrct))

     (Intercept) substancecocaine  substanceheroin
          34.373           -4.952            0.498

HELPrct = mutate(HELPrct, subnew = relevel(substance,
  ref="heroin"))
coef(lm(cesd ~ subnew, data=HELPrct))

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

12.9 *Group-wise statistics*

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

```
favstats(age ~ substance, data=HELPrct)

   .group min Q1 median   Q3 max mean   sd   n missing
1 alcohol  20 33   38.0 43.0  58 38.2 7.65 177       0
```

```
2 cocaine  23 30   33.5 37.2  60 34.5 6.69 152       0
3  heroin  19 27   33.0 39.0  55 33.4 7.99 124       0

ageGroup <- HELPrct %>%
  group_by(substance) %>%
  summarise(agebygroup = mean(age))
ageGroup

Source: local data frame [3 x 2]

  substance agebygroup
1   alcohol       38.2
2   cocaine       34.5
3    heroin       33.4

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

   .group  min   Q1 median   Q3  max mean sd   n missing
1 alcohol 38.2 38.2   38.2 38.2 38.2 38.2  0 177       0
2 cocaine 34.5 34.5   34.5 34.5 34.5 34.5  0 152       0
3  heroin 33.4 33.4   33.4 33.4 33.4 33.4  0 124       0
```

12.10 *Accounting for missing data*

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

```
smaller = select(HELPmiss, cesd, drugrisk, indtot, mcs, pcs,
  substance)
dim(smaller)

[1] 470   6

summary(smaller)

      cesd         drugrisk         indtot
 Min.   : 1.0   Min.   : 0.00   Min.   : 4.0
 1st Qu.:25.0   1st Qu.: 0.00   1st Qu.:32.0
 Median :34.0   Median : 0.00   Median :37.5
 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
      mcs            pcs          substance
 Min.   : 6.8   Min.   :14.1   alcohol:185
 1st Qu.:21.7   1st Qu.:40.3   cocaine:156
```

```
 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
```

Of the 470 subjects in the 6 variable dataframe, only the `drugrisk`, `indtot`, `mcs`, and `pcs` variables have missing values.

```
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
```

Alternatively, we could generate the same dataset using logical conditions.

```
nomiss = filter(smaller,
  (!is.na(mcs) & !is.na(indtot) & !is.na(drugrisk)))
dim(nomiss)

[1] 453   6
```

13
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 (R01-AA10870, Samet PI) and National Institute on Drug Abuse (R01-DA10019, Samet PI). The details of the randomized trial along with the results from a series of additional analyses have been published[1].

[1] 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

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 `mosaicData` 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.

Variables included in the HELP dataset are described in Table 13.1. More information can be found here[2]. A copy of the study instruments can be found at: `http://www.amherst.edu/~nhorton/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 Depression scale (range 0–60, higher scores indicate more depressive symptoms)	
`d1`	how many times hospitalized for medical problems (lifetime) (range 0–100)	
`daysanysub`	time (in days) to first use of any substance post-detox (range 0–268)	see also `anysubstatus`
`dayslink`	time (in days) to linkage to primary care (range 0–456)	see also `linkstatus`
`drugrisk`	Risk-Assessment Battery (RAB) drug risk score (range 0–21)	see also `sexrisk`
`e2b`	number of times in past 6 months entered a detox program (range 1–21)	
`female`	gender of respondent (0=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 (0=no, 1=yes)	
`i1`	average number of drinks (standard units) consumed per day (in the past 30 days, range 0–142)	see also `i2`
`i2`	maximum number of drinks (standard units) consumed per day (in the past 30 days range 0–184)	see also `i1`
`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 (0=no, 1=yes)	see also `dayslink`
`mcs`	SF-36 Mental Component Score (range 7-62, higher scores are better)	see also `pcs`
`pcs`	SF-36 Physical Component Score (range 14-75, higher scores are better)	see also `mcs`

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 (0=no, 1=yes)	
sex	sex of respondent (male or female)	
sexrisk	Risk-Assessment Battery (RAB) sex risk score (range 0–21)	see also drugrisk
substance	primary substance of abuse (alcohol, cocaine or heroin)	
treat	randomization group (randomize to HELP clinic, no or yes)	

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

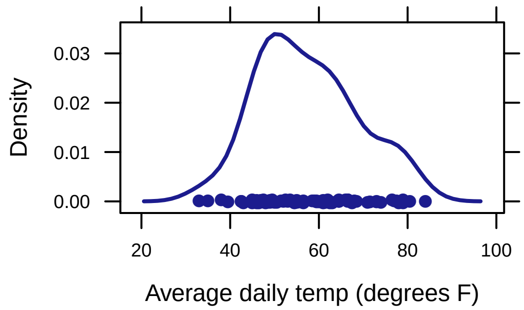
14
Exercises and Problems

2.1 Using the HELPrct dataset, create side-by-side histograms 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 relationship?

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
         378.83           -17.20             2.31
       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` dataframe, 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.

```
names(fusion1)

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

names(fusion2)

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

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

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

15
Bibliography

[BcRB⁺14] Ben Baumer, Mine Çetinkaya Rundel, Andrew Bray, Linda Loi, and Nicholas J. Horton. R Markdown: Integrating a reproducible analysis tool into introductory statistics. *Technology Innovations in Statistics Education*, 8(1):281–283, 2014.

[HK11] N. J. Horton and K. Kleinman. *Using R for Data Management, Statistical Analysis, and Graphics*. Chapman & Hall, 1st edition, 2011.

[KHF⁺03] 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⁺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.

[MM07] 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.

[RS02] Fred Ramsey and Dan Schafer. *Statistical Sleuth: A Course in Methods of Data Analysis*. Cengage, 2nd edition, 2002.

[SLH⁺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.

[Tuf01] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT, 2nd edition, 2001.

16 Index