Using the Devore6 package with R

Douglas Bates
Department of Statistics
University of Wisconsin-Madison

Contents

Chapter 0

Preliminaries

This document describes the use of the statistical package R as computing support in an introductory statistics course based on the text *Probability and Statistics for Engineering and the Sciences (6th edition)* by Jay Devore (Duxbury, 2004). We demonstrate how R can be used to reproduce the results in many of the examples in the text.

One of the desirable features of this text is the number of examples and exercises based on real data sets the Prof. Devore has culled from the engineering literature. As they are real data, some of the data sets are large and have a complex structure. Although it is not difficult to enter these data into a computer package like R, the process is tedious and error-prone. Furthermore, it is not much of a learning experience.

We have provided copies of the data sets for the examples and the exercises in a "package", named `Devore6`, that can be used with R. This document is also part of the `Devore6` package.

You may wish to try some of the examples in this section as you are reading it. We assume that you have both R and the Devore6 package for R installed. (See Appendix A for instructions if you need to do this first.)

Calculating a median

Suppose that we wish to reproduce the calculation of the median of the data on transferrin receptor concentration shown in Example 1.14 (p. 31 of the text). As there are only 12 concentrations, we could enter the data by hand. Start R and type

```
> conc = c(15.2, 9.3, 7.6, 11.9, 10.4, 9.7, 20.4, 9.4, 11.5,
+     16.2, 9.4, 8.3)
> str(conc)

 num [1:12] 15.2 9.3 7.6 11.9 10.4 9.7 20.4 9.4 11.5 16.2 ...

> median(conc)
```

```
[1] 10.05
```

The first line assigns the 12 data values as a numeric vector to the name `conc`, a short form of "concentration". The function named `"c"` concatenates a series of data values into a vector that can be assigned a name.

In the next line the `"str"` function is used to examine the structure of the object named `"conc"`. The output shows that this is a numeric vector of length 12 and displays the first several data values so you can check them against the data in the text.

Finally the `"median"` function, applied to the `conc` vector returns the median.

Using the Devore6 package

Entering the data, as shown above, is suitable for small data sets. An alternative and preferred way to access the data, especially for the larger data sets, is to use the `Devore6` package and load the data set. The data set for Example 1.14 is called `xmp01.14`. In general, data sets for examples in the text are named `xmp`*cc.nn*, where *cc* is the two-digit chapter number and *nn* is the two-digit example number. Data sets for exercises are named `ex`*cc.nn*. (Single digit chapter or example numbers have a `0` prepended as in `xmp01.14` so that the names sort in the correct order.)

You must attach the `Devore6` package every time you start R if you are to have access to the data sets from the textbook like this.

To attach the package to an R session use

```
> library(Devore6)
```

after starting R or select `Packages -> Load package -> Devore6` from the menu bar. (If this produces an error see Appendix A for instructions on installing the `Devore6` package.)

After attaching the package, you can load a data set with

```
> data(xmp01.14)
> str(xmp01.14)

`data.frame':        12 obs. of  1 variable:
 $ concentration: num  15.2 9.3 7.6 11.9 10.4 9.7 20.4 9.4 11.5 16.2 ...
```

The first line loads the data set into the current R session. The second line provides a description of the structure of the data set. It is a good practice always to use `str` on the data set after loading it.

Form of the data sets

The data set `xmp01.14` is not a single vector like `conc`. It is a data table (called a "data frame" in R) with one column and 12 rows. All the data sets in the `Devore6` package are data frames.

The output from `str` indicates that the name of the first column is `concentration`. To calculate the median we must give both the name of the data frame and the name of the column. We can do this in three ways, as described in Appendix **??**. We will focus on just one of these ways, which is to use the `"with"` function.

```
> with(xmp01.14, median(concentration))

[1] 10.05
```

The `with` function indicates which data set should be used to gain access to the column (or "variable") called `concentration`.

Summary

To recap:

1. You should have R installed on a computer and the `Devore6` package for R installed. (See Appendix A for instructions if you need to do this.)

2. At the beginning of each session use

   ```
   > library(Devore6)
   ```

   to allow access to the data sets from the package.

3. To load the data for a specific example or a specific exercise use a name of the form `xmp`*cc.nn* or `ex`*cc.nn* to `data()` then check the structure with `str()`.

   ```
   > data(xmp01.14)
   > str(xmp01.14)
   ```

In the remainder of this document we will not show these steps explicitly.

The R functions we have mentioned are shown in Table 1. See also Appendix **??**.

Function	Purpose
q()	quit R
help(name)	display help on an object (function or data set)
help.search("topic")	search for functions related to a topic
library(name)	Make data sets from a package available for loading
data(name)	Load a data set
str(name)	Display a brief description of the structure
with(dataset,...)	Use the variables in a data set

Table 1: R functions for general use

Chapter 1

Overview and Descriptive Statistics

We will follow the same sequence of topics and chapter headings as in the text and will begin each chapter with a table of R functions that are used in the chapter.

Table 1.1 lists functions used in chapter 1.

Function	Description
stem(x)	stem-and-leaf display
hist(x)	histogram
boxplot(x)	boxplot
mean(x)	mean (i.e. average) value of x
median(x)	median
var(x)	sample variance
sd(x)	sample standard deviation
log(x)	natural logarithm (works on entire vectors)
log(x, 10)	common (base 10) logarithm
sqrt(x)	square root
`x^(1/3)`	cube root

Table 1.1: R functions used with chapter 1

1.1 Example 1.1

Example 1.1 (p. 4) lists the ambient temperatures (°F) for each test firing or actual launch of the space shuttle prior to the *Challenger* tradgedy in 1986. In Figure 1.1 (p. 5) these data are displayed as a stem-and-leaf plot and as a histogram. There are 36 data values.

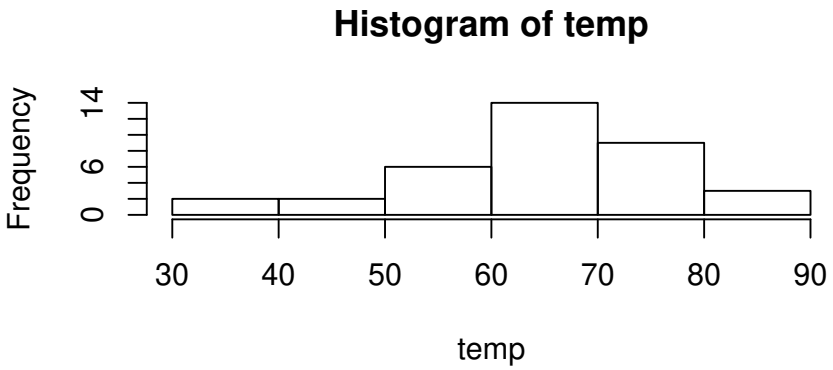
A stem-and-leaf plot similar to that in Figure 1.1 (p. 5) can be produced with

```
> with(xmp01.01, stem(temp))

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

  3 | 1
  4 | 059
  5 | 23788
  6 | 01136777789
  7 | 000023556689
  8 | 0134
```

(Remember that first you must attach the `Devore6` package, load the `xmp01.01` data set and check its structure. We do not show those steps here.)

The `hist` function produces a histogram.

```
> with(xmp01.01, hist(temp))
```



The stem-and-leaf plot and the histogram shown here are not exactly the same as those shown in Figure 1.1 (p. 5). In section 1.14 we show how optional arguments to `stem` and to `hist` could be used to produce displays similar to those in the text.

1.2 Example 1.5

The data in example 1.3 (p. 12), on the percentage of undergraduate students who are binge drinkers at 140 different campuses, are presented as a stem-and-leaf display in Figure 1.4 (p. 12).

```
> with(xmp01.05, stem(bingePct))
```

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

  0 | 4
  0 |
  1 | 134
  1 | 5678889
  2 | 12234
  2 | 56666777889999
  3 | 0112233344
  3 | 555666677777888899999
  4 | 11122222334444
  4 | 5566666677788888999
  5 | 001112222334
  5 | 55666667777888899
  6 | 011112444
  6 | 55666778

> with(xmp01.05, stem(bingePct, scale = 0.5))

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

  0 | 4
  1 | 1345678889
  2 | 1223456666777889999
  3 | 011223334455566667777788899999
  4 | 111222223344445566666677788888999
  5 | 00111222233455666667777888899
  6 | 01111244455666778
```

The first stem-and-leaf display is more spread out than the one in Figure 1.4 (p. 12). In the second call to `stem` we use the optional argument `scale` to shrink the scale by a factor of $\frac{1}{2}$ so the resulting display is similar to that in Figure 1.4.

1.3 Example 1.6

As in Example 1.5 a stem-and-leaf display is created, this time from data on yardages of golf courses as given in *Golf Magazine*.

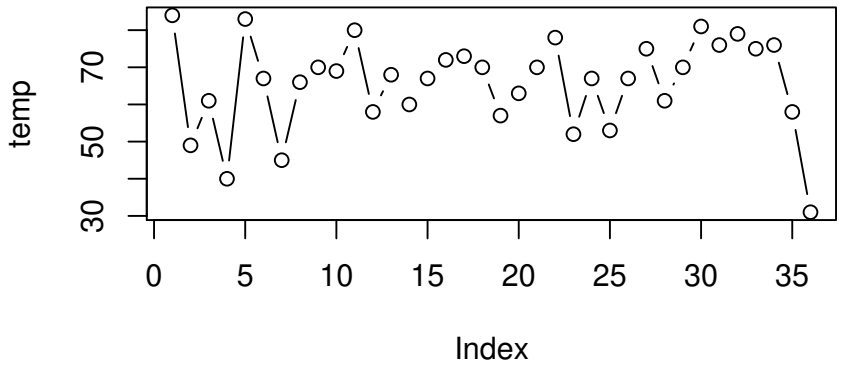
```
> with(xmp01.06, stem(yardage))

  The decimal point is 2 digit(s) to the right of the |

  64 | 3467
  65 | 1338
  66 | 119
  67 | 015779
```

```
68 | 05779
69 | 0034
70 | 112455
71 | 113777
72 | 18
```

1.4 Examples 1.7 and 1.8

Example 1.7 describes the *time-series plot* in Figure 1.6 (p. 14) of quarterly beer consumption. These data are not available so we show how a time-series plot of the shuttle launch temperatures from Example 1.1 could be created.

```
> with(xmp01.01, plot(temp, type = "b"))
```



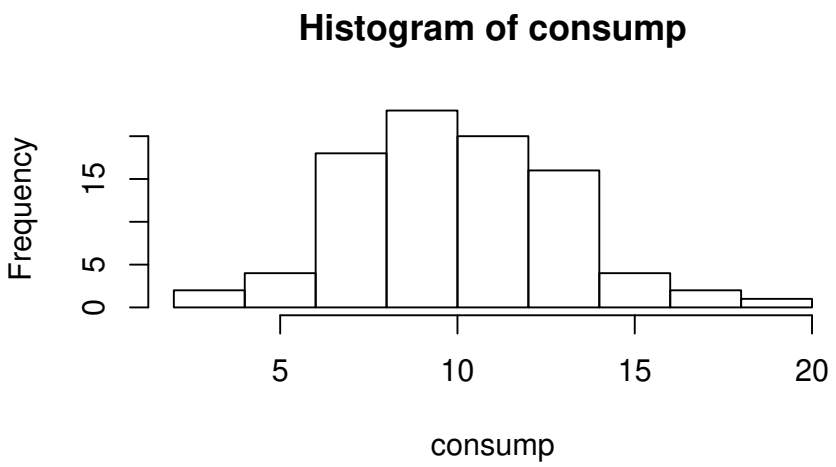
The optional argument `type="b"` to the `plot` function indicates that both the points and the connecting lines should be drawn on the plot.

There is no easy way to use R to produce the dotplot shown in Figure 1.7 (p. 14) for Example 1.8.

1.5 Example 1.10

A histogram such as shown in Figure 1.9 (p. 19) is produced by the `hist` function.

```
> with(xmp01.10, hist(consump))
```

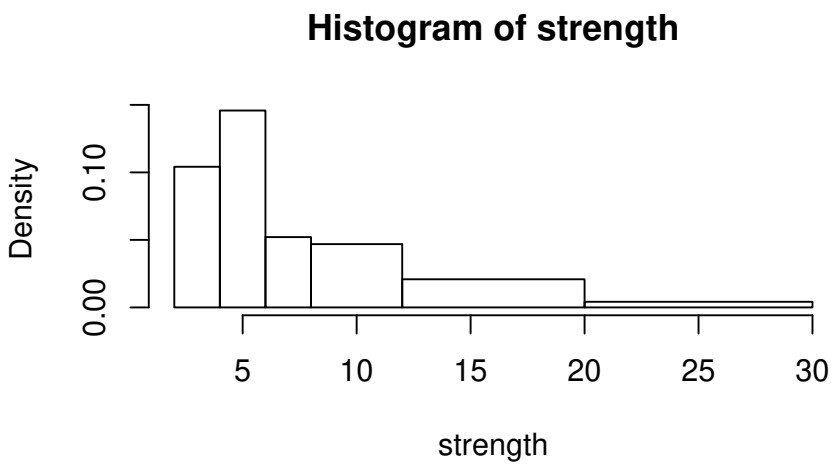


The vertical axis on this histogram is frequency. For a vertical axis on the scale of density (relative frequency divided by bin width) use the optional argument `freq = FALSE`.

1.6 Example 1.11

Figure 1.11 (p. 20) shows an example of a histogram with unequal bin widths. The optional argument `breaks` to the `hist` function is used to set the breakpoints for the bins. When unequal bin widths are used, the vertical axis switches to the density scale.

```
> with(xmp01.11, hist(strength, breaks = c(2, 4, 6, 8, 12,
+     20, 30)))
```

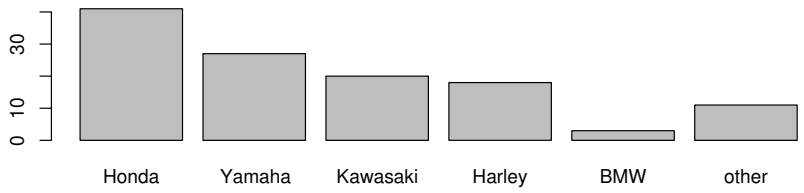


The `breaks` argument is a vector created by concatenating several numbers with the `c` function.

1.7 Example 1.12

Figure 1.13 (p. 22) shows a barplot for the motorcycle ownership data. As this data set is not available in the `Devore6` package, we create the vector of frequencies, assign names to the frequencies, and then use `barplot` to produce the plot

```
> cycles = c(41, 27, 20, 18, 3, 11)
> names(cycles) = c("Honda", "Yamaha", "Kawasaki", "Harley",
+     "BMW", "other")
> barplot(cycles)
```



1.8 Examples 1.13 and 1.14

Numeric measures of location are calculated with the functions `mean` and `median`. Another useful summary function is `sum`. A brief summary, including the mean, the median, the quartiles, the maximum and minimum is returned by `summary`.

```
> with(xmp01.13, mean(crackLength))

[1] 21.18095

> with(xmp01.13, sum(crackLength))

[1] 444.8

> with(xmp01.14, median(concentration))

[1] 10.05

> with(xmp01.13, summary(crackLength))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   8.90   14.00   21.20   21.18   25.80   45.00

> with(xmp01.14, summary(concentration))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  7.600   9.375  10.050  11.610  12.730  20.400
```

1.9 Example 1.15

The trimmed mean, described in Example 1.15 (p. 32) is obtained by using the optional `trim` argument to `mean`.

```
> with(xmp01.15, summary(lifetime))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  612.0   894.2  1010.0   965.0  1086.0  1201.0

> with(xmp01.15, mean(lifetime, trim = 0.1))

[1] 979.125
```

1.10 Example 1.16

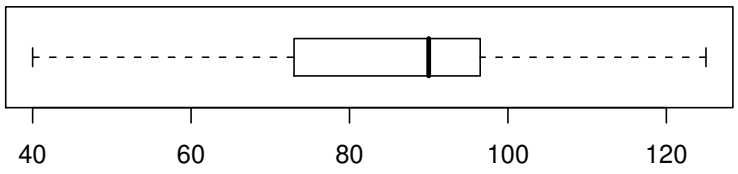
Functions `var` and `sd` provide the sample variance and sample standard deviation, respectively. The "computing formula" described on p. 39 is not used by these functions because that formula can have poor numerical properties.

```
> with(xmp01.16, var(Strength))

[1] 1.193580

> with(xmp01.16, sd(Strength))

[1] 1.092511
```

1.11 Examples 1.18

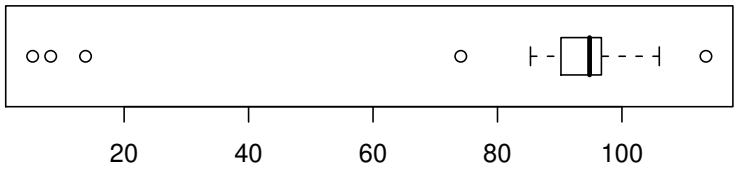
Function `boxplot` provides the boxplot. By default a vertical boxplot is constructed. Use the optional argument `horizontal=TRUE` to get a horizontal boxplot

```
> with(xmp01.18, boxplot(depth, horizontal = TRUE))
```



1.12 Example 1.19

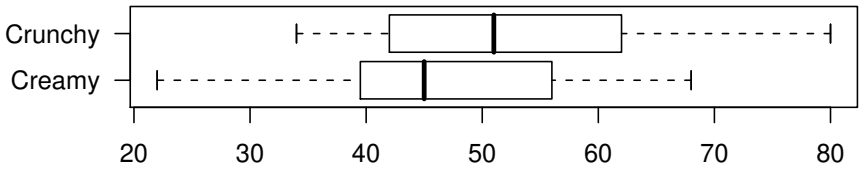
```
> with(xmp01.19, boxplot(C1, horizontal = TRUE))
```



1.13 Comparative boxplots

The data for Example 1.20 (p. 43) are not available so we use the data on scores for creamy and crunchy peanut butters (exercise 1.15, p. 24) to illustrate comparative boxplots. This data set has two columns and 37 rows. The score is the first column and the indicator of "Creamy" or "Crunchy" is the second column. We say that these data are in the stacked format (see Appendix **??** for details). In the call to `boxplot` we use the formula to indicate that `Score` is the response and `Type` defines the groups for the comparative boxplot.

```
> with(ex01.15, boxplot(Score ~ Type, horizontal = TRUE, las = 1))
```

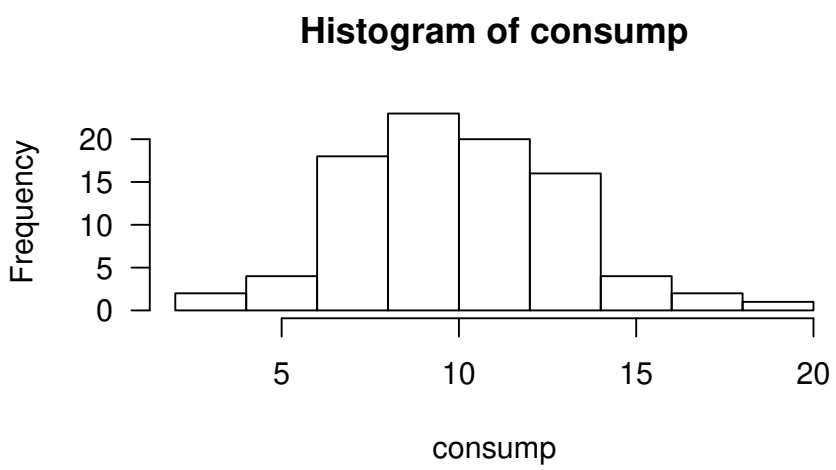


1.14 Enhancing graphical displays

In this chapter we have used several functions, such as `hist`, `barplot`, `plot`, and `boxplot` that produce graphical displays of data. These functions all can take optional arguments that provide more effective displays. For example, in § 1.13, we used the optional argument `las=1` to `boxplot` to change the label style on the vertical axis so the labels are horizontal rather than vertical.

The `las=1` argument can be used with other graphics functions. Compare

```
> with(xmp01.10, hist(consump, las = 1))
```



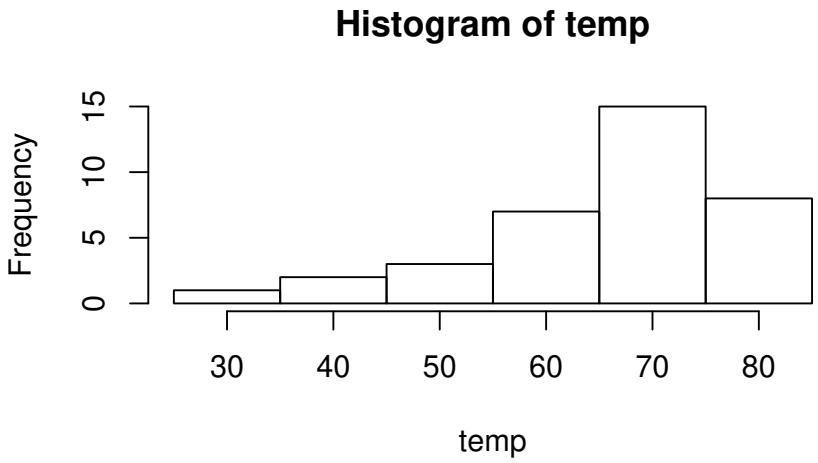
with the display in § 1.5

We have also seen how the `breaks` argument can be used with `hist` and how the `scale` argument can be used with `stem`. To reproduce a display like Figure 1.5 (p. 5) we would use

```
> with(xmp01.01, stem(temp, scale = 2))


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

  3 | 1
  3 |
  4 | 0
  4 | 59
  5 | 23
  5 | 788
  6 | 0113
  6 | 6777789
  7 | 000023
  7 | 556689
  8 | 0134
```

and

```
> with(xmp01.01, hist(temp, breaks = c(25, 35, 45, 55, 65,
+     75, 85)))
```


Histogram of temp
Frequency
0
5
10
15
30
40
50
60
70
80
temp

Chapter 2

Probability

Table 2.1 lists a function used in chapter 2.

Function	Description
choose(n,k)	calculate $\binom{n}{k}$

Table 2.1: R functions used in chapter 2

2.1 Example 2.23

In this chapter on probability there is little use for R functions except for the `choose` function that evaluates the number of combinations of k objects selected from n, written $\binom{n}{k}$ and described in section 2.3 (pp. 71–73). To calculate the first probability in example 2.23 (p. 73)

```
> choose(15, 3) * choose(10, 3)/choose(25, 6)

[1] 0.3083004
```

Chapter 3

Discrete Random Variables and Probability Distributions

To quote the document "Introduction to R"

> One convenient use of R is to provide a comprehensive set of statistical tables. Functions are provided to evaluate the cumulative distribution function P(X <= x), the probability density function and the quantile function (given q, the smallest x such that P(X <= x) > q), and to simulate from the distribution.

Distribution	R name	additional arguments
binomial	'binom'	'size, prob'
geometric	'geom'	'prob'
hypergeometric	'hyper'	'm, n, k'
negative binomial	'nbinom'	'size, prob'
Poisson	'pois'	'lambda'

> Prefix the name given here by 'd' for the density, 'p' for the CDF, 'q' for the quantile function and 'r' for simulation (**r**andom deviates). The first argument is 'x' for 'dXXX', 'q' for 'pXXX', 'p' for 'qXXX' and 'n' for 'rXXX' (except for 'rhyper' and 'rwilcox', for which it is 'nn').

These functions are more versatile and more accurate than using probability tables.

3.1 Example 3.30

For X having a binomial distribution with $n = 6$ and $p = 0.5$ we are to calculate $P(X = 3)$, $P(3 \leq X)$, and $P(X \leq 1)$.

```
> dbinom(3, size = 6, prob = 0.5)

[1] 0.3125

> dbinom(3:6, size = 6, prob = 0.5)

[1] 0.312500 0.234375 0.093750 0.015625

> sum(dbinom(3:6, size = 6, prob = 0.5))

[1] 0.65625

> 1 - pbinom(2, size = 6, prob = 0.5)

[1] 0.65625

> pbinom(2, size = 6, prob = 0.5, lower = FALSE)

[1] 0.65625

> pbinom(1, 6, 0.5)

[1] 0.109375
```

The first call evaluates $b(3; 6, .5)$. The second call evaluates the probability function at $3, \ldots, 6$ using the `":"` operator that generates the sequence from 3 to 6. If we sum this vector of probabilities we get $P(3 \leq X)$. An alternative is to use $P(3 \leq X) = 1 - P(X \leq 2)$ and evaluate $P(X \leq 2)$ with `pbinom`. Another alternative is to use cumulative probability in the upper tail, obtained with the optional argument `lower=FALSE` to `pbinom`. Finally `pbinom` is used to calculate $P(X \leq 1)$.

3.2 Example 3.31

```
> pbinom(8, 15, 0.2)

[1] 0.999215

> dbinom(8, 15, 0.2)

[1] 0.003454764

> 1 - pbinom(7, 15, 0.2)

[1] 0.00423975

> sum(dbinom(4:7, 15, 0.2))

[1] 0.3475981
```

3.3 Example 3.34

R uses a different, but equivalent, set of parameters for the hypergeometric distribution than does the text. In the text the parameters of the hypergeometric are N, the population size, n, the sample size, and M, the number of "successes" in the population. In R the sample size is called k, the parameter m corresponds to M in the text, and n is $N - M$.

Thus what is written in the text as $h(2; 5, 12, 20)$ becomes

```
> dhyper(2, 12, 8, 5)

[1] 0.2383901
```

3.4 Example 3.35

```
> dhyper(2, 5, 20, 10)

[1] 0.3853755

> phyper(2, 5, 20, 10)

[1] 0.6988142
```

3.5 Example 3.37

The negative binomial density function, `dnbinom`, shown in the text as $nb(10; 5, 2)$, has essentially the same calling sequence in R. The cumulative probability function is `pnbinom`.

```
> dnbinom(10, 5, 0.2)

[1] 0.0343941

> pnbinom(10, 5, 0.2)

[1] 0.1642337
```

3.6 Example 3.38

```
> dpois(5, lambda = 4.5)

[1] 0.1708269

> ppois(5, 4.5)

[1] 0.7029304
```

3.7 Example 3.39

The Poisson distribution can be used to approximate binomial probabilities with large n and small p. However, there is no need to do so because the exact binomial probabilities can be evaluated.

```
> dbinom(1, 400, 0.005)

[1] 0.2706694

> dpois(1, 2)

[1] 0.2706706

> pbinom(3, 400, 0.005)

[1] 0.8575767

> ppois(3, 2)

[1] 0.8571235
```

Chapter 4

Continuous Random Variables and Probability Distributions

The set of continous distributions available in R is

Distribution	R name	additional arguments
beta	'beta'	'shape1, shape2, ncp'
Cauchy	'cauchy'	'location, scale'
χ^2	'chisq'	'df, ncp'
exponential	'exp'	'rate'
F	'f'	'df1, df1, ncp'
gamma	'gamma'	'shape, scale'
log-normal	'lnorm'	'meanlog, sdlog'
logistic	'logis'	'location, scale'
normal	'norm'	'mean, sd'
Student's t	't'	'df, ncp'
uniform	'unif'	'min, max'
Weibull	'weibull'	'shape, scale'
Wilcoxon	'wilcox'	'm, n'

As with the discrete distributions, prefix the name given here by 'd' for the density, 'p' for the CDF, 'q' for the quantile function and 'r' for simulation (**r**andom deviates). The first argument is 'x' for 'dXXX', 'q' for 'pXXX', 'p' for 'qXXX' and 'n' for 'rXXX'

Not all the distributions shown above are discussed in the text.

4.1 Example 4.12

Function `pnorm` provides the standard normal cumulative distribution function by default. The optional arguments `mean` and `sd` can be set to values other than

0 and 1 to provide probabilities from any normal distribution.

$P(Z \leq 1.25)$ and $P(Z \leq -1.25)$ are evaluated as

```
> pnorm(1.25)

[1] 0.8943502

> pnorm(-1.25)

[1] 0.1056498
```

$P(Z > 1.25)$ can be evaluated in two ways

```
> 1 - pnorm(1.25)

[1] 0.1056498

> pnorm(1.25, lower = FALSE)

[1] 0.1056498
```

To evaluate probabilities of intervals, such as $P(-0.38 \leq Z \leq 1.25)$, apply `pnorm` to the endpoints as a vector (created with the `"c"` function) which returns a vector of probabilities. The `"diff"` function forms successive differences from which we obtain the probability in the interval.

```
> pnorm(c(-0.38, 1.25))

[1] 0.3519727 0.8943502

> diff(pnorm(c(-0.38, 1.25)))

[1] 0.5423775
```

4.2 Example 4.13

The inverse of the standard normal CDF, called the quantile function, is obtained with `qnorm`. Notice that the first argument to `qnorm` is a probability, not a percentage.

```
> qnorm(0.99)

[1] 2.326348
```

4.3 Example 4.14

To obtain z_α, use the optional argument `lower=FALSE` to `qnorm`.

```
> qnorm(0.05, lower = FALSE)

[1] 1.644854
```

4.4 Example 4.15

Nonstandard normal distribution probabilities or quantiles are obtained with the optional arguments `mean` and `sd` to `pnorm` and `qnorm`. In this example $\mu = 1.25$ and $\sigma = 0.46$ and we wish to evaluate $P(1.00 \leq X \leq 1.75)$

```
> diff(pnorm(c(1, 1.75), mean = 1.25, sd = 0.46))

[1] 0.5680717
```

4.5 Example 4.17

For $\mu = 64$ and $\sigma = 0.78$, the 99.5th percentile is

```
> qnorm(0.995, mean = 64, sd = 0.78)

[1] 66.00915
```

4.6 Example 4.19

The normal approximation to binomial probabilities can be calculated but, like the Poisson approximation, it is not necessary as the exact binomial probabilities can also be calculated.

```
> pnorm(10.5, mean = 12.5, sd = sqrt(12.5 * 0.75))

[1] 0.2568146

> pbinom(10, size = 50, prob = 0.25)

[1] 0.2622023

> diff(pnorm(c(4.5, 15.5), mean = 12.5, sd = sqrt(12.5 * 0.75)))

[1] 0.8319162

> diff(pbinom(c(4, 15), size = 50, prob = 0.25))

[1] 0.8348084
```

4.7 Example 4.20

The parameters α and β of the gamma distribution are named `shape` and `scale` respectively in R. In this example $\alpha = 2$ and β has the default value of 1.

```
> pgamma(c(3, 5), shape = 2)

[1] 0.8008517 0.9595723
```

```
> diff(pgamma(c(3, 5), shape = 2))

[1] 0.1587206

> pgamma(4, shape = 2, lower = FALSE)

[1] 0.0915782
```

4.8 Example 4.21

```
> diff(pgamma(c(60, 120), shape = 8, scale = 15))

[1] 0.4959056

> pgamma(30, shape = 8, scale = 15, lower = FALSE)

[1] 0.9989033
```

4.9 Examples 4.22 and 4.23

In R the parameter λ of the exponential distribution is called `rate`

```
> pexp(10, rate = 0.2)

[1] 0.8646647

> diff(pexp(c(5, 10), rate = 0.2))

[1] 0.2325442

> pexp(2, rate = 0.5, lower = FALSE)

[1] 0.3678794
```

4.10 Example 4.24

The parameters α and β of the Weibull distribution are called `shape` and `scale`.

```
> pweibull(10, shape = 2, scale = 10)

[1] 0.6321206

> qweibull(0.95, shape = 2, scale = 10)

[1] 17.30818
```

4.11 Example 4.26

The lognormal distribution takes two parameters named `meanlog` and `sdlog`.

```
> diff(plnorm(c(1, 2), meanlog = 0.375, sdlog = 0.25))
```

```
[1] 0.8316108
```

```
> qlnorm(0.99, meanlog = 0.375, sdlog = 0.25)
```

```
[1] 2.602798
```

4.12 Example 4.27

R provides probabilities and quantiles of the standard beta distribution with $A = 0$ and $B = 1$. The parameters α and β are called `shape1` and `shape2` respectively. To use other values of A and B the scaling must be done manually. In this example $A = 2$, $B = 5$, $\alpha = 2$ and $\beta = 3$. To transform to a standard beta distribution we replace x by $(x - 2)/(5 - 2)$

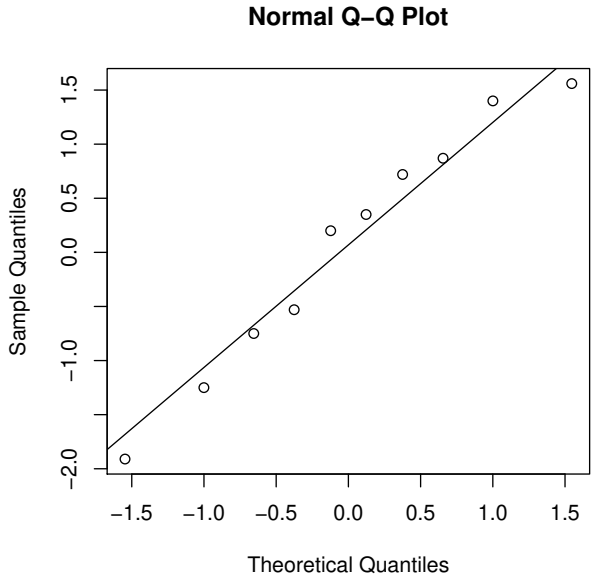
```
> pbeta((3 - 2)/(5 - 2), shape1 = 2, shape2 = 3)
```

```
[1] 0.4074074
```

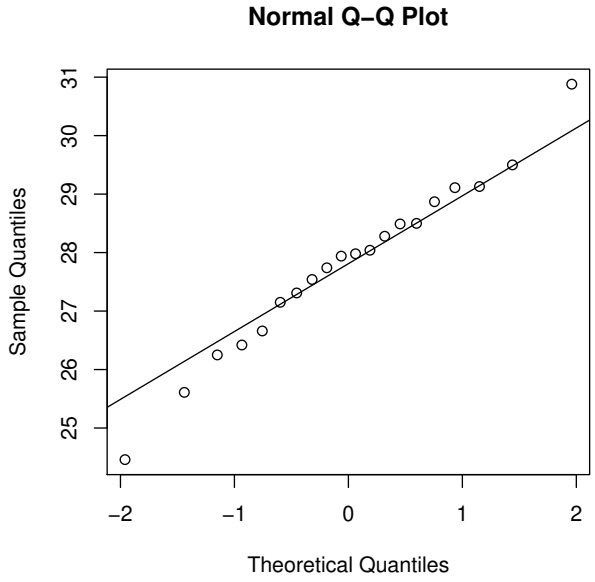
4.13 Examples 4.28 and 4.29

The normal probability plot is produced with `qqnorm`. A reference line can be added with `qqline`.

```
> with(xmp04.28, qqnorm(meas.err))
> with(xmp04.28, qqline(meas.err))
```



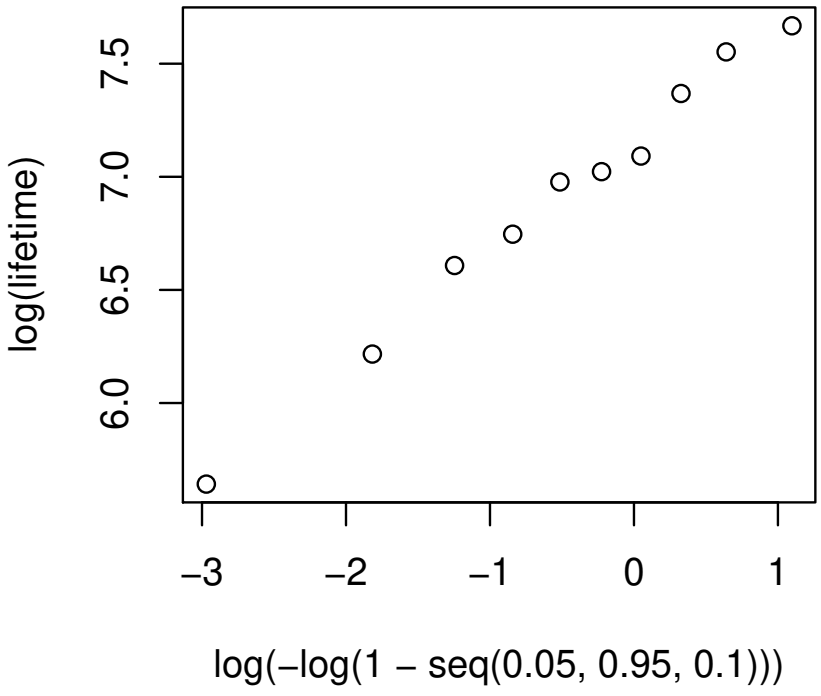

```
> with(xmp04.29, qqnorm(Voltage))
> with(xmp04.29, qqline(Voltage))
```



4.14 Example 4.30

The Weibull probability plot is not available directly in R. However, the plot can be created using the formula $\ln[-\ln(1-p)]$ for the 5th, 15th, ..., and 95th percentiles as given in text. The sequence $0.05, 0.15, \ldots, 0.95$ is generated with `seq(0.05, 0.95, 0.1)`.

```
> with(xmp04.30, plot(log(-log(1 - seq(0.05, 0.95, 0.1))),
+     log(lifetime)))
```



Chapter 5

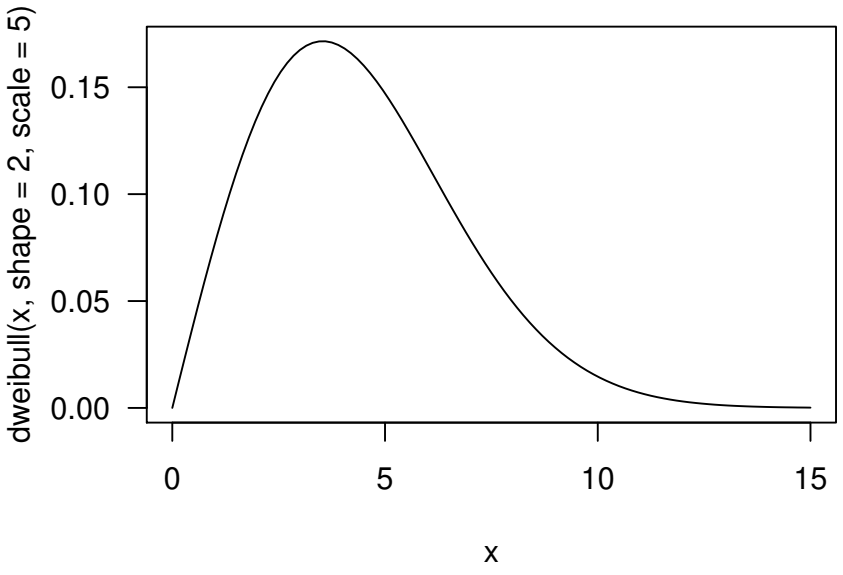
Joint Probability Distributions and Random Samples

The main use of R in this chapter is for simulation experiments as described in section 5.3.

5.1 Example 5.19

In this example six samples of size ten are drawn from a Weibull distribution with $\alpha = 2$ and $\beta = 5$. To reproduce the plot of the density shown in Figure 5.6 we can use

```
> curve(dweibull(x, shape = 2, scale = 5), 0, 15, las = 1)
```



To get a single sample of size 10 from this Weibull distribution we use

```
> rweibull(10, shape = 2, scale = 5)

 [1] 5.3396 7.4460 4.0282 7.9231 1.3094 5.6180 3.8558 4.0923 5.8237 3.5346
```

We could store such a sample as, say, `samp`, then evaluate its sample mean, sample median, and sample standard deviation.

```
> samp = rweibull(10, shape = 2, scale = 5)
> print(samp)

 [1] 2.0573 6.1328 7.4634 5.9205 4.6584 2.7243 3.1309 4.4855 2.0697 7.9331

> mean(samp)

[1] 4.6576

> median(samp)

[1] 4.572

> sd(samp)

[1] 2.1584
```

Notice that every time `rweibull` is called a new sample is generated.

This process of generating a sample and evaluating selected statistics on the sample could be repeated manually to get a total of 6 samples. For large simulation experiments this would quickly become tedious so we put these calculations in a loop.

```
> means = medians = sds = numeric(6)
> for (i in 1:6) {
+     samp = rweibull(10, shape = 2, scale = 5)
+     print(samp)
+     means[i] = mean(samp)
+     medians[i] = median(samp)
+     sds[i] = sd(samp)
+ }

 [1] 6.33990 1.45771 7.75343 2.91567 4.57116 1.65255 7.37469 4.24496
 [9] 0.69652 4.24561
 [1] 3.1949 2.9406 7.6942 5.3531 4.9792 3.4230 2.2851 3.7577 8.3336 1.6082
 [1] 5.76868 5.17182 1.65700 2.93255 5.22258 5.99908 3.51198 7.37884
 [9] 0.95013 1.51488
 [1]  6.8738  2.0723  5.5495  1.9942  3.7529  9.0547  1.1731 11.3804
 [9]  4.4835  4.4660
 [1] 2.90531 2.11623 0.64940 4.01813 2.41116 7.54389 5.71016 3.60282
 [9] 0.56603 4.02859
 [1] 6.1796 5.1305 3.5434 6.7727 8.9507 2.5021 5.2978 0.7854 5.4584 6.8814

> means

[1] 4.1252 4.3570 4.0108 5.0800 3.3552 5.1502

> medians

[1] 4.2453 3.5903 4.3419 4.4747 3.2541 5.3781

> sds

[1] 2.4789 2.2303 2.2049 3.2533 2.1544 2.3546
```

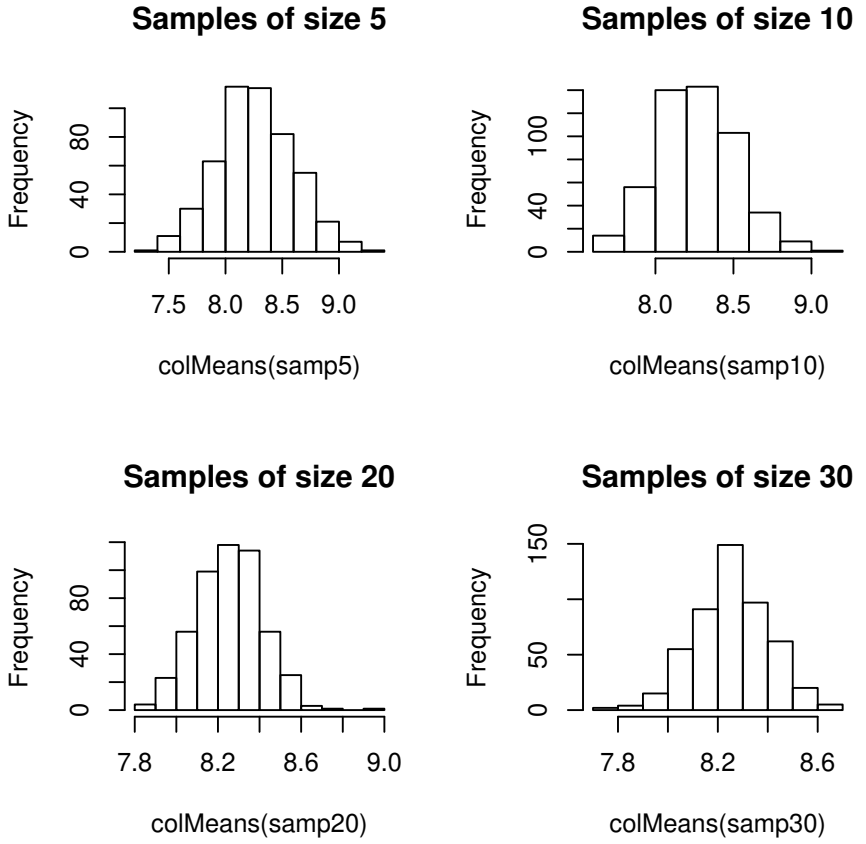
In the first line we assign the names `means`, `medians`, and `sds` to numeric vectors of length 6. Within the loop we assign individual elements in these vectors.

5.2 Example 5.22

In this example 500 samples of size $n = 5$ are generated from a normal distribution with $\mu = 8.25$ and $\sigma = 0.75$ and the mean of each sample is calculated. We could do this in a loop, as shown above. However, it is more compact to generate a matrix with 5 rows and 500 columns then generate a histogram of the means

of the columns of this matrix. Function `colMeans` calculates the means of the columns. Function `matrix` creates a matrix from a numeric vector. The user can specify the number of rows and the number of columns. If one of these is omitted, it is calculated from the length of the vector and the other dimension. We set the graphics parameter `"mfrow"` (multiple figures by row) to create a 2 by 2 array of plots.

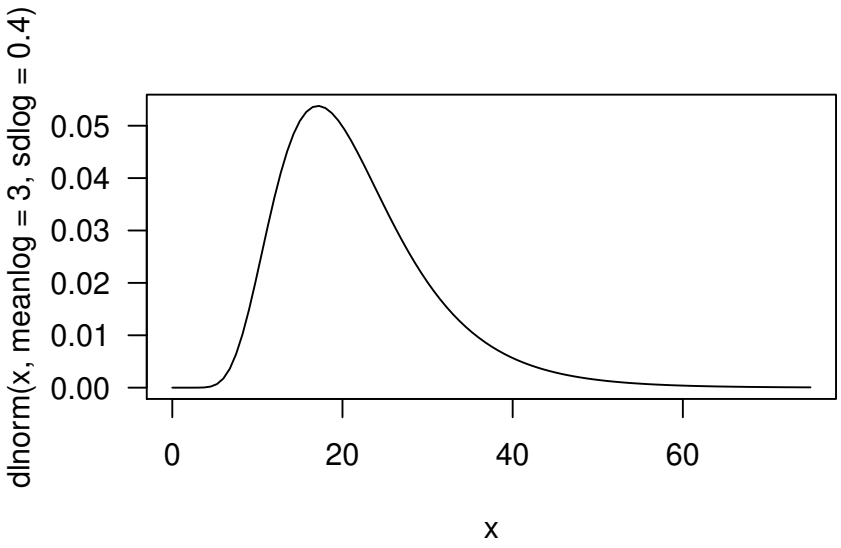
```
> par(mfrow = c(2, 2))
> samp5 = matrix(rnorm(500 * 5, mean = 8.25, sd = 0.75), ncol = 500)
> hist(colMeans(samp5), main = "Samples of size 5")
> samp10 = matrix(rnorm(500 * 10, mean = 8.25, sd = 0.75),
+     ncol = 500)
> hist(colMeans(samp10), main = "Samples of size 10")
> samp20 = matrix(rnorm(500 * 20, mean = 8.25, sd = 0.75),
+     ncol = 500)
> hist(colMeans(samp20), main = "Samples of size 20")
> samp30 = matrix(rnorm(500 * 30, mean = 8.25, sd = 0.75),
+     ncol = 500)
> hist(colMeans(samp30), main = "Samples of size 30")
```



5.3 Example 5.23

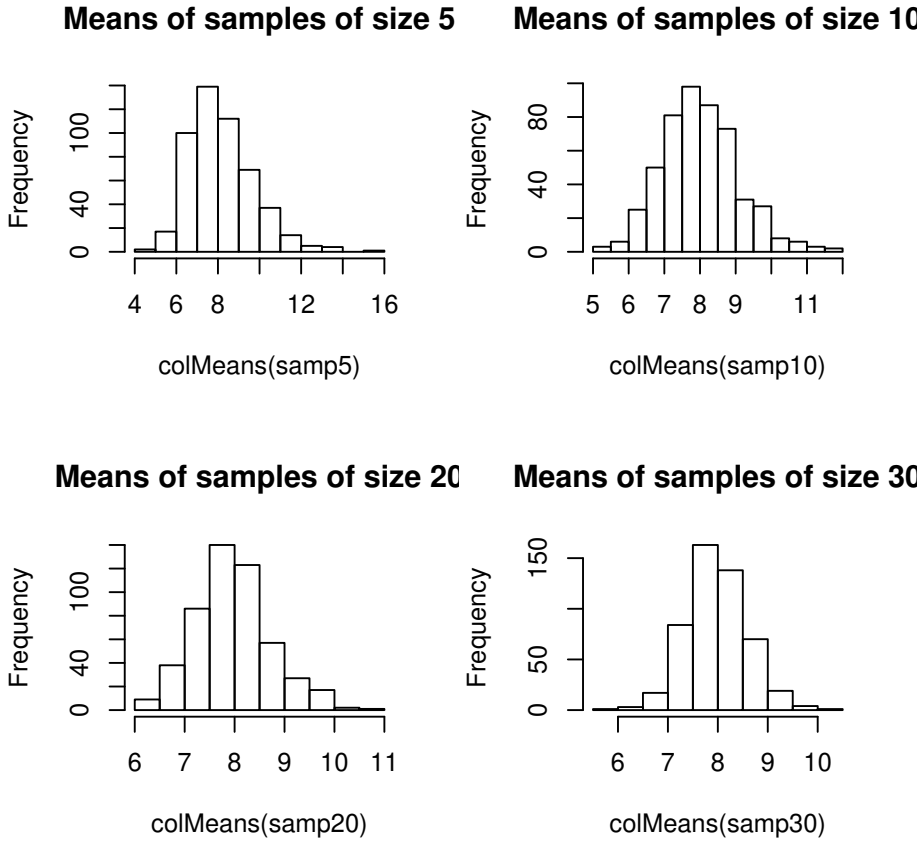
This example is similar to Example 5.22. To reproduce Figure 5.12 use

```
> curve(dlnorm(x, meanlog = 3, sdlog = 0.4), from = 0, to = 75,
+     las = 1)
```



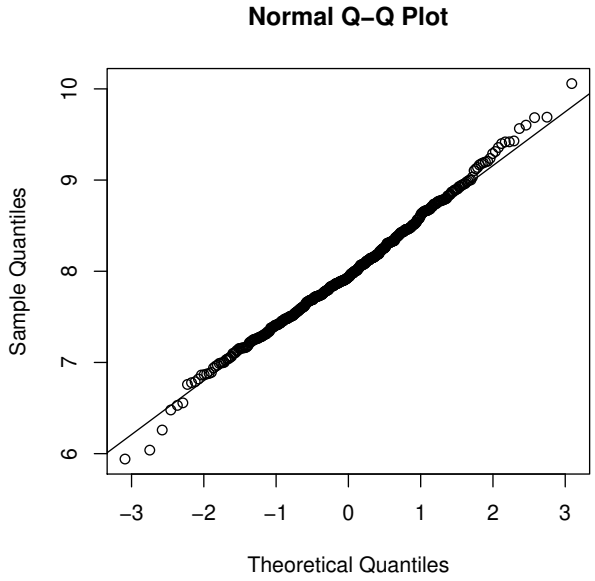
The samples and the histograms of the means are generated from

```
> par(mfrow = c(2, 2))
> samp5 = matrix(rlnorm(500 * 5, 2, 0.4), ncol = 500)
> hist(colMeans(samp5), main = "Means of samples of size 5")
> samp10 = matrix(rlnorm(500 * 10, 2, 0.4), ncol = 500)
> hist(colMeans(samp10), main = "Means of samples of size 10")
> samp20 = matrix(rlnorm(500 * 20, 2, 0.4), ncol = 500)
> hist(colMeans(samp20), main = "Means of samples of size 20")
> samp30 = matrix(rlnorm(500 * 30, 2, 0.4), ncol = 500)
> hist(colMeans(samp30), main = "Means of samples of size 30")
```



Finally the normal probability plot is generated by

```
> qqnorm(colMeans(samp30))
> qqline(colMeans(samp30))
```


Normal Q–Q Plot
Sample Quantiles
Theoretical Quantiles
6
7
8
9
10
−3
−2
−1
0
1
2
3

Chapter 6

Point Estimation

6.1 Example 6.2

The various estimators of location described in Example 6.2 (p. 255) can be evaluated as

```
> with(xmp06.02, mean(Voltage))

[1] 27.793

> with(xmp06.02, median(Voltage))

[1] 27.96

> with(xmp06.02, mean(range(Voltage)))

[1] 27.67

> with(xmp06.02, mean(Voltage, trim = 0.1))

[1] 27.83812
```

6.2 Example 6.3

Functions `var` and `sd` provide s^2 and s, the sample variance and standard deviation, respectively.

```
> with(xmp06.03, var(Strength))

[1] 0.25125

> with(xmp06.03, sd(Strength))

[1] 0.5012484
```

To evaluate the alternative estimator $\hat{\sigma} = \frac{\sum(X_i - \bar{X})}{n}$ we must evaluate the formula

```
> with(xmp06.03, sum((Strength - mean(Strength))^2)/length(Strength))

[1] 0.2198438
```

Function `length` applied to a vector returns n, the number of elements in the vector.

6.3 Example 6.12

The calculation of the method of moments estimates in Example 6.12 (p. 270) can be split into stages

```
> xbar = with(xmp06.12, mean(Survival))
> xsqb = with(xmp06.12, mean(Survival^2))
> xbar

[1] 113.45

> xsqb

[1] 14087.75

> xbar^2/(xsqb - xbar^2)

[1] 10.57725

> (xsqb - xbar^2)/xbar

[1] 10.72585
```

These estimates are slightly different from those shown in the text because the intermediate results $\bar{x}$ and $\sum x_i^2/n$ were rounded in the text.

Maximum likelihood estimates are discussed later in chapter 6. We can evaluate the maximum likelihood estimates of α and β for this example using the function `fitdistr` from the package `MASS` that supplements the book "Modern Applied Statistics with S (4th ed)" by Bill Venables and Brian Ripley (Springer, 2002). These estimates are determined by numerical optimization of the logarithm of the likelihood function and we must supply starting estimates. We use the method of moments estimates for this.

```
> library(MASS)
> with(xmp06.12, fitdistr(Survival, dgamma, list(shape = 10.577,
+     scale = 10.726)))

     shape       scale
   8.802043   12.888909
 ( 2.731807) ( 4.116383)
```

The MLEs are quite different from the method of moments estimates. The numbers in parentheses under the estimates are their standard errors.

Chapter 7

Statistical Intervals Based on a Single Sample

Table 7.1 lists functions used in chapters 7 and 8.

Function	Description
t.test(x)	Student's t test and confidence interval
prop.test(x,n)	Test and confidence interval on proportion
binom.test(x,n)	Test and confidence interval on proportion

Table 7.1: R functions used with chapters 7 and 8

7.1 Example 7.2

The calculation of the confidence interval for known σ, shown in Example 7.2, could be done in R as

```
> 80 + c(-1, 1) * 1.96 * 2/sqrt(31)
```

```
[1] 79.29595 80.70405
```

but it is probably just as easy to use a hand calculator for this.

7.2 Example 7.6

The calculation of the sample mean, the sample standard deviation, and the sample size can be combined into a single statement

```
> with(xmp07.06, mean(Voltage) + c(-1, 1) * 1.96 * sd(Voltage)/sqrt(length(Voltage)))
```

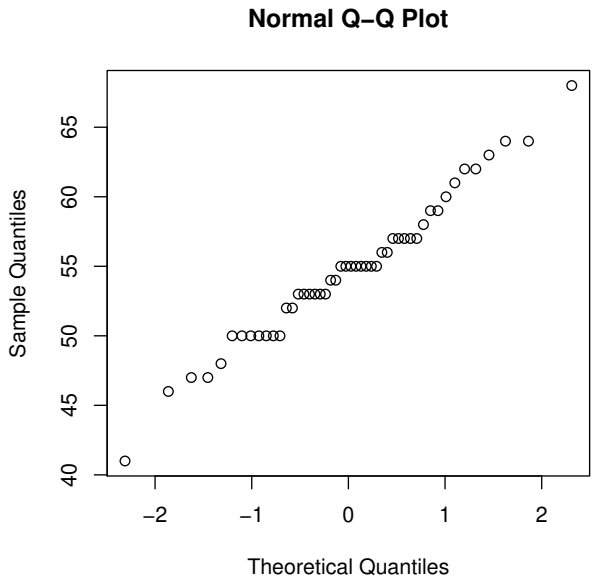
```
[1] 53.22857 56.18810
```

An alternative, and preferred way, of calculating the interval is to use `t.test`. With 48 observations the confidence interval on μ from the t distribution is nearly identical to that from the standard normal distribution.

```
> with(xmp07.06, t.test(Voltage))

        One Sample t-test

data:  Voltage
t = 72.4631, df = 47, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 53.18950 56.22716
sample estimates:
mean of x
 54.70833
```

It is always a good idea to check the normal probability plot when using `t.test`, even with a large sample.

```
> with(xmp07.06, qqnorm(Voltage))
```



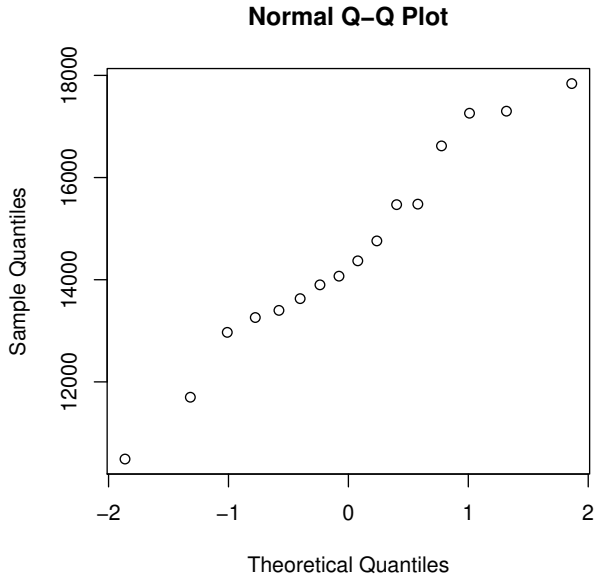
7.3 Example 7.8

R has two functions, `binom.test` and `prop.test`, that can be used to calculate a large-sample confidence interval on the binomial proportion, p. Neither of them corresponds exactly the the confidence interval described on p. 295.

```
> prop.test(16, 48)

        1-sample proportions test with continuity correction

data:  16 out of 48, null probability 0.5
X-squared = 4.6875, df = 1, p-value = 0.03038
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.2080794 0.4851357
sample estimates:
        p
0.3333333

> binom.test(16, 48)

        Exact binomial test

data:  16 and 48
number of successes = 16, number of trials = 48, p-value = 0.02930
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.2039597 0.4841083
sample estimates:
probability of success
             0.3333333
```

7.4 Example 7.11

```
> with(xmp07.11, t.test(Elasticity))

        One Sample t-test

data:  Elasticity
t = 28.2778, df = 15, p-value = 1.989e-14
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 13437.11 15627.89
sample estimates:
mean of x
  14532.5

> with(xmp07.11, qqnorm(Elasticity))
```

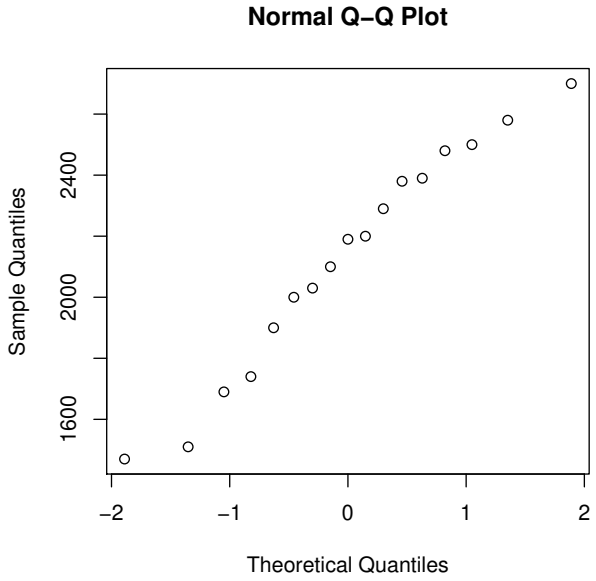


7.5 Example 7.15

Although there is no built-in confidence interval for σ^2 in R, the `qchisq` function can be used to obtain the critical values $\chi^2_{\alpha/2,n-1}$ and $\chi^2_{1-\alpha/2,n-1}$ used to calculate the interval.

```
> with(xmp07.15, qqnorm(voltage))
> with(xmp07.15, 16 * var(voltage)/qchisq(c(0.975, 0.025),
+     df = 16))
```

```
[1]  76171.31 318079.76
```


Normal Q–Q Plot
Sample Quantiles
1600
2000
2400
–2
–1
0
1
2
Theoretical Quantiles

Chapter 8

Tests of Hypotheses Based on a Single Sample

The functions described in chapter 7 are used for performing tests of hypotheses based on a single sample. Optional arguments are used to specify μ_0 or p_0 and to indicate the form of the alternative hypothesis. All of these tests return a p-value, described in §8.4 (pp. 344–350). From the p-value the result of the hypothesis test for any level α can be determined.

8.1 Example 8.8

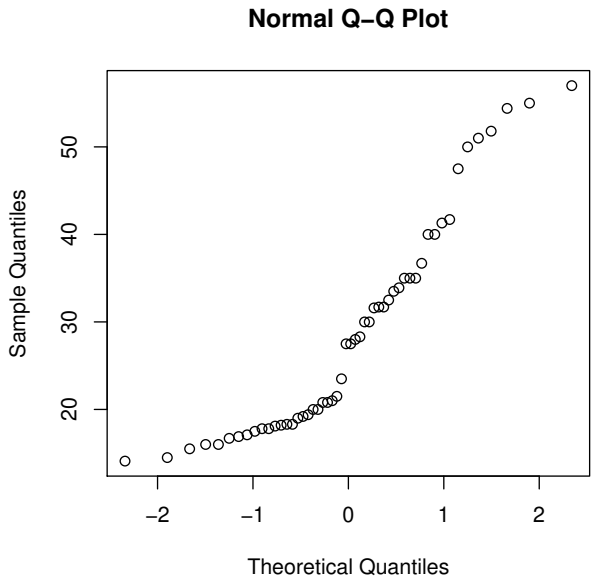
In R the `t.test` function can be used with any size of data set. For large n the t test is essentially equivalent to the large-sample z test.

```
> with(xmp08.08, t.test(DCP, mu = 30, alt = "less"))

        One Sample t-test

data:  DCP
t = -0.7282, df = 51, p-value = 0.2349
alternative hypothesis: true mean is less than 30
95 percent confidence interval:
     -Inf 31.61088
sample estimates:
mean of x
 28.76154
```

As the p-value of 0.2349 exceeds 0.05 we do not reject $H_0 : \mu = 30$ versus $H_a : \mu < 30$ at level $\alpha = 0.05$.

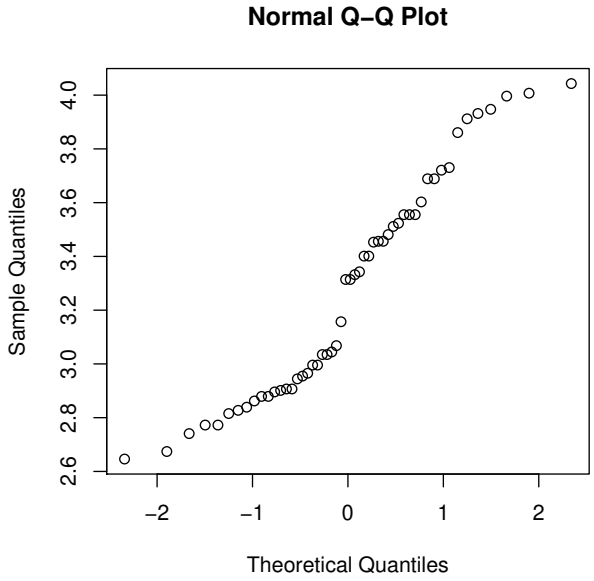
Although not shown in the text book, it is of interest to examine the normal probability plot for these data

```
> with(xmp08.08, qqnorm(DCP))
```



This plot shows considerable skewness in the data. If we transform to the logarithm of the DCP measurement the skewness is diminished.

```
> with(xmp08.08, qqnorm(log(DCP)))
```



and in the logarithm scale, the hypothesis test $H_0 : \mu_{log} = \log(30)$ versus $H_a : \mu_{log} < \log(30)$ is significant at level $\alpha = 0.05$.

```
> with(xmp08.08, t.test(log(DCP), mu = log(30), alt = "less"))

        One Sample t-test

data:  log(DCP)
t = -2.223, df = 51, p-value = 0.01534
alternative hypothesis: true mean is less than 3.401197
95 percent confidence interval:
     -Inf 3.370103
sample estimates:
mean of x
 3.274999
```

8.2 Example 8.9

```
> with(xmp08.09, t.test(MAWL, mu = 25, alt = "greater"))

        One Sample t-test

data:  MAWL
t = 1.0382, df = 4, p-value = 0.1789
alternative hypothesis: true mean is greater than 25
```

```
95 percent confidence interval:
 22.32433      Inf
sample estimates:
mean of x
    27.54
```

8.3 Example 8.10

```
> power.t.test(n = 10, delta = 0.1, sd = 0.1, type = "one.sample",
+     alt = "one.sided")

     One-sample t test power calculation

              n = 10
          delta = 0.1
             sd = 0.1
      sig.level = 0.05
          power = 0.897517
    alternative = one.sided

> power.t.test(delta = 0.1, sd = 0.1, power = 0.95, type = "one.sample",
+     alt = "one.sided")

     One-sample t test power calculation

              n = 12.32052
          delta = 0.1
             sd = 0.1
      sig.level = 0.05
          power = 0.95
    alternative = one.sided
```

8.4 Example 8.11

The large-sample test for a population proportion is most closely related to the result of the `prop.test` function

```
> prop.test(1276, 4115, p = 0.3, alt = "greater")

        1-sample proportions test with continuity correction

data:  1276 out of 4115, null probability 0.3
X-squared = 1.9453, df = 1, p-value = 0.08155
alternative hypothesis: true p is greater than 0.3
95 percent confidence interval:
 0.2982330 1.0000000
```

```
sample estimates:
        p
0.3100851
```

The value labeled `X-squared` is the square of the z statistic. This version of the test uses a continuity correction. If you wish to reproduce the test statistic as given in the text book, add the optional argument `correct=FALSE`

```
> prop.test(1276, 4115, p = 0.3, alt = "greater", correct = FALSE)

        1-sample proportions test without continuity correction

data:  1276 out of 4115, null probability 0.3
X-squared = 1.993, df = 1, p-value = 0.07901
alternative hypothesis: true p is greater than 0.3
95 percent confidence interval:
 0.2983532 1.0000000
sample estimates:
        p
0.3100851

> sqrt(1.993)

[1] 1.411737
```

In both cases the p-value is less than 0.1 so we reject $H_0 : p = 0.3$ versus $H_a : p > 0.3$ at level $\alpha = 0.1$.

8.5 Example 8.13

The small sample test is provided by `binom.test`

```
> binom.test(x = 14, n = 20, p = 0.9, alt = "less")

        Exact binomial test

data:  14 and 20
number of successes = 14, number of trials = 20, p-value = 0.01125
alternative hypothesis: true probability of success is less than 0.9
95 percent confidence interval:
 0.0000000 0.8604463
sample estimates:
probability of success
                   0.7
```

Chapter 9

Inference Based on Two Samples

```
> print(bwplot(type ~ strength, xmp09.07, xlab = "Tensile strength (psi)"))
```

```
> print(qqmath(~strength | type, xmp09.07, xlab = "Standard normal quantiles",
+     ylab = "Tensile strength (psi)", type = c("g", "p"),
+     aspect = 1))
```

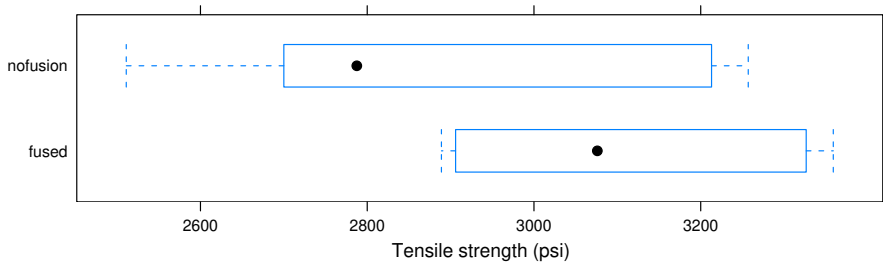
9.1 Example 9.7

When given vectors of numbers as arguments, the summary function returns the mean, min, max and quartiles of the given vector. The qqnorm function is used to generate a qqplot of its given argument. The simplest use of the boxplot function creates a boxplot for each vector argument passed to it.

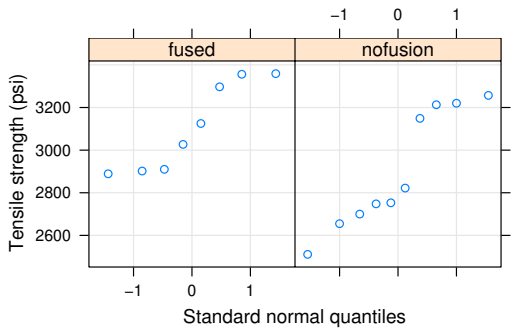
```
> str(xmp09.07)
```

```
`data.frame':        18 obs. of  2 variables:
 $ strength: num  2748 2700 2655 2822 2511 ...
 $ type    : Factor w/ 2 levels "fused","nofusion": 2 2 2 2 2 2 2 2 2 ...
```

```
> bwplot(type ~ strength, xmp09.07)
```



```
> qqmath(~strength | type, xmp09.07)
```

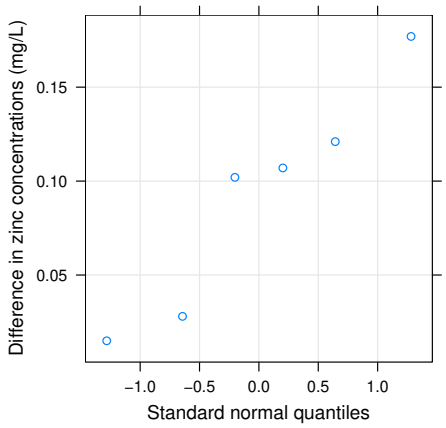



```
> t.test(strength ~ type, xmp09.07, alt = "greater")

        Welch Two Sample t-test

data:  strength by type
t = 1.8018, df = 15.944, p-value = 0.04526
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 6.332257      Inf
sample estimates:
   mean in group fused mean in group nofusion
               3108.125                2902.800

> print(qqmath(~I(bottom - surface), xmp09.08, type = c("g",
+     "p"), xlab = "Standard normal quantiles", aspect = 1,
+     ylab = "Difference in zinc concentrations (mg/L)"))
```

9.2 Example 9.8

The t.test function conducts various types of t tests. When given one vector argument, it performs a one sample test, when two vector arguments are given, and the paired option is specified as TRUE, the function can perform a paired two sample tests as well. Options can also present to specify if a test is one or two sided.

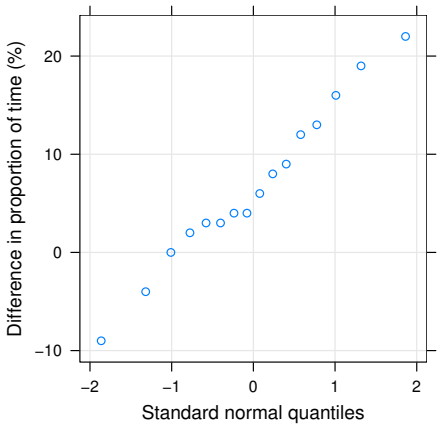
```
> str(xmp09.08)

`data.frame':        6 obs. of  2 variables:
 $ bottom : num  0.43 0.266 0.567 0.531 0.707 0.716
 $ surface: num  0.415 0.238 0.39 0.41 0.605 0.609

> qqmath(~I(bottom - surface), xmp09.08)
```




```
> with(xmp09.08, t.test(bottom, surface, paired = TRUE))

        Paired t-test

data:  bottom and surface
t = 3.6998, df = 5, p-value = 0.01400
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.02797823 0.15535510
sample estimates:
mean of the differences
             0.09166667

> print(qqmath(~Difference, xmp09.09, type = c("g", "p"), xlab = "Standard normal quantiles",
+     aspect = 1, ylab = "Difference in proportion of time (%)"))
```

9.3 Example 9.9

```
> str(xmp09.09)

`data.frame':        16 obs. of  4 variables:
 $ Subject   : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Before    : int  81 87 86 82 90 86 96 73 74 75 ...
 $ After     : int  78 91 78 78 84 67 92 70 58 62 ...
 $ Difference: int  3 -4 8 4 6 19 4 3 16 13 ...

> qqmath(~Difference, xmp09.09)
```




```
> with(xmp09.09, t.test(Before, After, paired = TRUE))

        Paired t-test

data:  Before and After
t = 3.2791, df = 15, p-value = 0.005072
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  2.362371 11.137629
sample estimates:
mean of the differences
                   6.75
```

9.4 Example 9.10

```
> Difference <- c(5, 19, 25, 10, 10, 10, 28, 46, 25, 38, 14,
+     23, 14)
> qqnorm(Difference)
> qqline(Difference)
> t.test(Difference)

        One Sample t-test

data:  Difference
t = 6.1904, df = 12, p-value = 4.654e-05
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 13.30957 27.76736
sample estimates:
mean of x
 20.53846
```

Chapter 10

The Analysis of Variance

Chapter 11

Multifactor Analysis of Variance

11.1 Example 11.01

An *interaction plot* shows the response versus the levels of one factor with the points joined according to the levels of the other factor. It an additive model is valid, the lines should be approximately parallel.

For a balanced data set, the order of the factors does not affect the calculations of the sums of squares nor any of the test statistics and conclusions. However, for unbalanced data the order of the factors is important. In general we put the blocking factor(s) first and the treatment factor(s) last.

The `TukeyHSD` function can be applied to aov models with more than one factor. If we are only interested in selected factors we can use the argument `which` to restrict the factors being considered.

Plots are useful aids in checking if assumptions have been satisfied. To assess the constant variance assumption, residual plots are used and the normal probability plot of the residuals is used to assess the assumption of a normal distribution of the noise term. The simplest way to obtain such plots in R is to plot the fitted model object. Use `which = 1` to get residuals versus fitted values, `which = 2` to get the qqnorm plot of the residuals, or `which = 1:2` to get both.

```
> with(xmp11.01, interaction.plot(treatment, brand, strength,
+     col = 2:4, lty = 1))
> with(xmp11.01, interaction.plot(treatment, brand, strength,
+     col = 2:4, lty = 1))
> with(xmp11.01, interaction.plot(brand, treatment, strength,
+     col = 2:4, lty = 1))
> with(xmp11.01, interaction.plot(brand, treatment, strength,
+     col = 2:4, lty = 1))
> anova(fm1 <- aov(strength ~ treatment + brand, xmp11.01))
```

```
Analysis of Variance Table

Response: strength
          Df  Sum Sq Mean Sq F value   Pr(>F)
treatment  3 0.47969 0.15990 11.0549 0.007399
brand      2 0.12822 0.06411  4.4323 0.065765
Residuals  6 0.08678 0.01446

> TukeyHSD(fm1, which = "treatment")

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

Fit: aov(formula = strength ~ treatment + brand, data = xmp11.01)

$treatment
           diff        lwr         upr     p adj
2-1 -0.46666667 -0.8065953 -0.12673802 0.0124339
3-1 -0.38000000 -0.7199286 -0.04007135 0.0315128
4-1 -0.50333333 -0.8432620 -0.16340469 0.0086330
3-2  0.08666667 -0.2532620  0.42659531 0.8141883
4-2 -0.03666667 -0.3765953  0.30326198 0.9806209
4-3 -0.12333333 -0.4632620  0.21659531 0.6185041

> plot(fm1, which = 1:2)
```

11.2 Example 11.05

```
> str(xmp11.05)

`data.frame':        20 obs. of  3 variables:
 $ power: int  685 722 733 811 828 792 806 802 888 920 ...
 $ humid: Ord.factor w/ 4 levels "1"<"2"<"3"<"4": 1 1 1 1 1 2 2 2 2 2 ...
 $ brand: Factor w/ 5 levels "1","2","3","4",..: 1 2 3 4 5 1 2 3 4 5 ...

> with(xmp11.05, interaction.plot(humid, brand, power, col = 2:6,
+     lty = 1))
> anova(fm1 <- aov(power ~ humid + brand, data = xmp11.05))

Analysis of Variance Table

Response: power
          Df Sum Sq Mean Sq F value    Pr(>F)
humid      3 116218   38739 278.199 2.364e-11
brand      4  53231   13308  95.567 5.419e-09
Residuals 12   1671     139
```

```
> TukeyHSD(fm1, which = "brand")

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

Fit: aov(formula = power ~ humid + brand, data = xmp11.05)

$brand
      diff        lwr       upr     p adj
2-1  46.00  19.403556  72.59644 0.0010268
3-1  41.50  14.903556  68.09644 0.0024312
4-1 116.50  89.903556 143.09644 0.0000001
5-1 139.75 113.153556 166.34644 0.0000000
3-2  -4.50 -31.096444  22.09644 0.9812528
4-2  70.50  43.903556  97.09644 0.0000175
5-2  93.75  67.153556 120.34644 0.0000008
4-3  75.00  48.403556 101.59644 0.0000092
5-3  98.25  71.653556 124.84644 0.0000005
5-4  23.25  -3.346444  49.84644 0.0978028

> plot(TukeyHSD(fm1, which = "brand"))
> plot(TukeyHSD(fm1, which = "brand"))
```

11.3 Example 11.06

```
> str(xmp11.06)

`data.frame':        24 obs. of  3 variables:
 $ Resp    : num  8 6.9 9.3 9.2 12 9.4 17.3 19.3 18.8 24.9 ...
 $ Stimulus: Factor w/ 6 levels "L1","L2","T",..: 1 2 3 4 5 6 1 2 3 4 ...
 $ Subject : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 2 2 2 2 ...

> anova(fm1 <- aov(Resp ~ Subject + Stimulus, data = xmp11.06))

Analysis of Variance Table

Response: Resp
          Df  Sum Sq Mean Sq F value    Pr(>F)
Subject    3 13444.6  4481.5 86.1112 1.110e-09
Stimulus   5  1428.3   285.7  5.4888   0.00455
Residuals 15   780.7    52.0

> with(xmp11.06, interaction.plot(Stimulus, Subject, Resp,
+     col = 2:6, lty = 1))
> TukeyHSD(fm1, which = "Stimulus")
```

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

Fit: aov(formula = Resp ~ Subject + Stimulus, data = xmp11.06)

$Stimulus
             diff         lwr      upr     p adj
L2-L1       3.050 -13.5235020 19.62350 0.9895813
T-L1        4.275 -12.2985020 20.84850 0.9553756
L1+L2-L1   15.525  -1.0485020 32.09850 0.0727869
L1+T-L1    16.400  -0.1735020 32.97350 0.0532377
L2+T-L1    20.225   3.6514980 36.79850 0.0129117
T-L2        1.225 -15.3485020 17.79850 0.9998651
L1+L2-L2   12.475  -4.0985020 29.04850 0.2019526
L1+T-L2    13.350  -3.2235020 29.92350 0.1528205
L2+T-L2    17.175   0.6014980 33.74850 0.0401628
L1+L2-T    11.250  -5.3235020 27.82350 0.2908899
L1+T-T     12.125  -4.4485020 28.69850 0.2248769
L2+T-T     15.950  -0.6235020 32.52350 0.0625789
L1+T-L1+L2  0.875 -15.6985020 17.44850 0.9999744
L2+T-L1+L2  4.700 -11.8735020 21.27350 0.9348445
L2+T-L1+T   3.825 -12.7485020 20.39850 0.9719095

> plot(fm1, which = 1)
> range(xmp11.06$Resp)

[1]  6.9 96.6

> anova(fm2 <- aov(log(Resp) ~ Subject + Stimulus, data = xmp11.06))

Analysis of Variance Table

Response: log(Resp)
          Df  Sum Sq Mean Sq F value    Pr(>F)
Subject    3 13.1315  4.3772 264.972 3.270e-13
Stimulus   5  1.1089  0.2218  13.425 4.331e-05
Residuals 15  0.2478  0.0165

> with(xmp11.06, interaction.plot(Stimulus, Subject, log(Resp),
+     col = 2:6, lty = 1))
> plot(fm2, which = 1)
> opar <- par(pty = "s")
> plot(fm2, which = 2)
> par(opar)
> plot(TukeyHSD(fm2, which = "Stimulus"))
> with(xmp11.06, interaction.plot(Stimulus, Subject, fitted(fm2),
+     col = 2:6, lty = 1))
```

11.4 Example 11.07

```
> str(xmp11.07)

`data.frame':        36 obs. of  3 variables:
 $ Yield  : num  10.5 9.2 7.9 12.8 11.2 13.3 12.1 12.6 14 10.8 ...
 $ Variety: Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
 $ Density: Factor w/ 4 levels "1","2","3","4": 1 1 1 2 2 2 3 3 3 4 ...

> xtabs(~Variety + Density, data = xmp11.07)

       Density
Variety 1 2 3 4
      1 3 3 3 3
      2 3 3 3 3
      3 3 3 3 3

> xtabs(Yield ~ Variety + Density, data = xmp11.07)

       Density
Variety    1    2    3    4
      1 27.6 37.3 38.7 32.4
      2 26.8 37.9 43.5 38.3
      3 48.9 54.3 59.8 54.5

> xtabs(Yield ~ Variety + Density, xmp11.07)/xtabs(~Variety +
+     Density, xmp11.07)

       Density
Variety         1         2         3         4
      1  9.200000 12.433333 12.900000 10.800000
      2  8.933333 12.633333 14.500000 12.766667
      3 16.300000 18.100000 19.933333 18.166667

> anova(fm1 <- aov(Yield ~ Density * Variety, data = xmp11.07))

Analysis of Variance Table

Response: Yield
                Df Sum Sq Mean Sq  F value    Pr(>F)
Density          3  86.69   28.90  18.2306 2.212e-06
Variety          2 327.60  163.80 103.3430 1.608e-12
Density:Variety  6   8.03    1.34   0.8445    0.5484
Residuals       24  38.04    1.58

> with(xmp11.07, interaction.plot(Density, Variety, Yield,
+     col = 2:6, lty = 1))
> anova(fm2 <- aov(Yield ~ Density + Variety, xmp11.07))
```

```
Analysis of Variance Table

Response: Yield
          Df Sum Sq Mean Sq F value    Pr(>F)
Density    3  86.69   28.90  18.816 4.690e-07
Variety    2 327.60  163.80 106.659 2.313e-14
Residuals 30  46.07    1.54

> TukeyHSD(fm2)

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

Fit: aov(formula = Yield ~ Density + Variety, data = xmp11.07)

$Density
          diff        lwr        upr     p adj
2-1  2.9111111  1.3226494  4.4995728 0.0001375
3-1  4.3000000  2.7115383  5.8884617 0.0000002
4-1  2.4333333  0.8448716  4.0217951 0.0013148
3-2  1.3888889 -0.1995728  2.9773506 0.1033936
4-2 -0.4777778 -2.0662395  1.1106839 0.8455586
4-3 -1.8666667 -3.4551284 -0.2782049 0.0163721

$Variety
        diff        lwr      upr     p adj
2-1 0.875000 -0.3722268 2.122227 0.2109855
3-1 6.791667  5.5444399 8.038893 0.0000000
3-2 5.916667  4.6694399 7.163893 0.0000000

> model.tables(fm2, type = "means")

Tables of means
Grand mean

13.88889

 Density
Density
     1      2      3      4
11.478 14.389 15.778 13.911

 Variety
Variety
     1      2      3
11.333 12.208 18.125
```

11.5 Example 11.11

```
> str(xmp11.11)

`data.frame':        96 obs. of  4 variables:
 $ Tempr : num  3.6 3.8 3.9 3.4 3.7 3.9 2.9 2.8 2.7 2.5 ...
 $ Period: Ord.factor w/ 4 levels "1"<"2"<"3"<"4": 1 1 1 1 1 1 1 1 1 1 ...
 $ Strain: Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
 $ Coat  : Factor w/ 4 levels "1","2","3","4": 1 1 1 2 2 2 3 3 3 4 ...

> xtabs(~Period + Coat + Strain, xmp11.11)

, , Strain = 1

      Coat
Period 1 2 3 4
     1 3 3 3 3
     2 3 3 3 3
     3 3 3 3 3
     4 3 3 3 3

, , Strain = 2

      Coat
Period 1 2 3 4
     1 3 3 3 3
     2 3 3 3 3
     3 3 3 3 3
     4 3 3 3 3

> anova(fm1 <- aov(Tempr ~ Period * Coat * Strain, xmp11.11))

Analysis of Variance Table

Response: Tempr
                   Df Sum Sq Mean Sq  F value    Pr(>F)
Period              3  0.490   0.163   4.1398 0.0095807
Coat                3 48.934  16.311 413.1671 < 2.2e-16
Strain              1  6.458   6.458 163.5910 < 2.2e-16
Period:Coat         9  1.608   0.179   4.5268 0.0001254
Period:Strain       3  0.020   0.007   0.1715 0.9152673
Coat:Strain         3  0.876   0.292   7.3975 0.0002489
Period:Coat:Strain  9  0.250   0.028   0.7039 0.7029981
Residuals          64  2.527   0.039

> anova(fm2 <- update(fm1, . ~ . - Period:Strain - Period:Coat:Strain))
```

```
Analysis of Variance Table

Response: Tempr
            Df Sum Sq Mean Sq  F value    Pr(>F)
Period       3  0.490   0.163   4.4408 0.0062620
Coat         3 48.934  16.311 443.2022 < 2.2e-16
Strain       1  6.458   6.458 175.4832 < 2.2e-16
Period:Coat  9  1.608   0.179   4.8559 3.905e-05
Coat:Strain  3  0.876   0.292   7.9353 0.0001135
Residuals   76  2.797   0.037
```

11.6 Example 11.12

```
> str(xmp11.12)

`data.frame':        36 obs. of  4 variables:
 $ abrasion: num  7.38 5.39 5.03 5.5 5.01 6.79 7.15 8.16 4.96 5.78 ...
 $ row     : Ord.factor w/ 6 levels "1"<"2"<"3"<"4"<..: 1 1 1 1 1 1 2 2 2 2 ...
 $ column  : Ord.factor w/ 6 levels "1"<"2"<"3"<"4"<..: 1 2 3 4 5 6 1 2 3 4 ...
 $ humidity: Factor w/ 6 levels "25%","37%","50%",..: 3 4 6 2 5 1 2 1 5 4 ...

> xtabs(as.integer(humidity) ~ row + column, xmp11.12)

   column
row 1 2 3 4 5 6
  1 3 4 6 2 5 1
  2 2 1 5 4 3 6
  3 4 6 3 5 1 2
  4 1 3 2 6 4 5
  5 6 5 1 3 2 4
  6 5 2 4 1 6 3

> xtabs(abrasion ~ row + column, xmp11.12)

   column
row    1    2    3    4    5    6
  1 7.38 5.39 5.03 5.50 5.01 6.79
  2 7.15 8.16 4.96 5.78 6.24 5.06
  3 6.75 5.64 6.34 5.31 7.81 8.05
  4 8.05 6.45 6.31 5.46 6.05 5.51
  5 5.65 5.44 7.27 6.54 7.03 5.96
  6 6.00 6.55 5.93 8.02 5.80 6.61

> anova(fm1 <- aov(abrasion ~ row + column + humidity, xmp11.12))

Analysis of Variance Table
```

```
Response: abrasion
          Df  Sum Sq Mean Sq F value   Pr(>F)
row        5  2.1897  0.4379  2.5106  0.06407
column     5  2.5743  0.5149  2.9516  0.03731
humidity   5 23.5301  4.7060 26.9789 3.03e-08
Residuals 20  3.4887  0.1744
```

```
> model.tables(fm1, cterms = "humidity", type = "mean")
```

```
Tables of means
Grand mean

6.305

 humidity
humidity
  25%   37%   50%   62%   75%   87%
7.683 6.765 6.593 5.977 5.372 5.440
```

```
> TukeyHSD(fm1, which = "humidity")
```

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

Fit: aov(formula = abrasion ~ row + column + humidity, data = xmp11.12)

$humidity
               diff        lwr         upr     p adj
37%-25% -0.91833333 -1.6762719 -0.16039478 0.0121211
50%-25% -1.09000000 -1.8479386 -0.33206145 0.0024825
62%-25% -1.70666667 -2.4646052 -0.94872811 0.0000097
75%-25% -2.31166667 -3.0696052 -1.55372811 0.0000001
87%-25% -2.24333333 -3.0012719 -1.48539478 0.0000001
50%-37% -0.17166667 -0.9296052  0.58627189 0.9782473
62%-37% -0.78833333 -1.5462719 -0.03039478 0.0385446
75%-37% -1.39333333 -2.1512719 -0.63539478 0.0001513
87%-37% -1.32500000 -2.0829386 -0.56706145 0.0002818
62%-50% -0.61666667 -1.3746052  0.14127189 0.1545364
75%-50% -1.22166667 -1.9796052 -0.46372811 0.0007299
87%-50% -1.15333333 -1.9112719 -0.39539478 0.0013767
75%-62% -0.60500000 -1.3629386  0.15293855 0.1683501
87%-62% -0.53666667 -1.2946052  0.22127189 0.2697646
87%-75%  0.06833333 -0.6896052  0.82627189 0.9997113
```

Appendix A

Installing R and the Devore6 package

A.1 What is R?

R is a freely available, open source, computer system for statistical analysis and graphics. It can be downloaded from the main R information site `http://www.r-project.org`, from the Comprehensive R Archive Network (CRAN) site `http://cran.r-project.org`, or from any of the mirrors of that site. Those in the United States, for example, are encouraged to use the U.S. mirror `http://cran.us.r-project.org`.

R provides facilities for data input and manipulation, for graphical and numerical summaries, for simulation and exploration of probability distributions, and for statistical analysis of data. It can be used for computing support for essentially all the topics in an introductory statistics course. This document describes how to use R for computing support in a course that uses the text *Probability and Statistics for Engineering and the Sciences (6th edition)* by Jay Devore (Duxbury, 2004).

Althought there are some limited graphical user interface (gui) capabilites for R, it is basically a command-line system. We will concentrate on the command-line interface, showing what the user types and what R responds. We will refer to what the user types as a "command" although, technically, we should use the term "function call".

A.2 Obtaining and Installing R

Binary versions of R are available for various operating systems including Microsoft Windows (Windows 95 or later), the Macintosh (OS X), and several Linux distributions. Complete source code for R is also available on the archives but it is quite unlikely that you will need to compile R for use with an intro-

ductory Statistics course.

As most students use R with Windows we will provide more detailed installation instructions for this operating system.

The current release of R is 1.7.1. The binary installer file for R-1.7.1 for Windows is called `rw1071.exe`. It is approximately 20MB in size and can be found at `http://cran.r-project.org/bin/windows/base/rw1071.exe`. If you have a fast network connection you should download and execute this file to install R. Without a network connection or with a slow network connection you will need to make other arrangements for obtaining a copy of this file.

The installation should provide a desktop icon or menu item for R-1.7.1. Use one of these to start R. The program should display a welcome message and a prompt `"> "`. At this point you could use it as a calculator. Try, for example,

```
> 2 + 2

[1] 4
```

A.3 Quitting R

To quit from R you can either select `File -> Exit` from the menu bar or type

```
> q()
```

at the prompt, as indicated in the startup message. It is necessary to type the parentheses. That is, typing `q` by itself is not sufficient.

Both of these methods will bring up a confirmation panel asking if you want to save the worksheet. In most cases you will not need to save the worksheet.

A.4 Using data sets

A standard R installation provides several data sets that are used to demonstrate different techniques. The `data` command provides a list of these

```
> data()
```

You can obtain a description of a data set with the `help` command or with the short form for help which is `?` followed by the name. Try, for example,

```
> help(pressure)
> "?"(pressure)
```

A.5 What is Devore6?

Notice that the description of the available data sets groups them into "packages" such as the "base" package, the "modreg" package, etc. Packages are groups of functions and data sets that extend the capabilities of R for specific purposes.

The "Devore6" package provides the data sets for the 6th edition of Devore's engineering statistics text book. By installing and attaching this package you will be able to use the data sets from the examples and exercises in this text book without having to enter the data by hand.

The Devore6 package also provides several "vignettes" - documents that describe particular aspects of the use of R. This document is one of the vignettes from the Devore6 package.

A.6 Installing and attaching Devore6

Installing and attaching a package are two different operations. Installation involves downloading the package from a web site and installing the files on the local hard drive. It only needs to be done once. A package that has been installed can be attached to an R session after which the data sets will be available in the session.

To install the Devore6 package on a computer with access to the internet, either use the command

```
> install.packages("Devore6")
```

or select `Packages -> Install package(s) from CRAN -> Devore6` from the menu bar.

If you do not have access to the Internet you will need to obtain a copy of the zip file whose name begins with `Devore6` in the directory `http://cran.r-project.org/bin/windows/contrib/1.7/`. (The exact name of the file changes as the package is updated but it will always begin with `Devore6` and end with `.zip`.) Use the menu selection `Packages -> Install package(s) fro local zip files` to install the package from the zip file.

We emphasize that it is only necessary to do the installation once.

To attach the package to an R session use

```
> library(Devore6)
```

after starting R or select `Packages -> Load package -> Devore6` from the menu bar.

You must attach the package every time you start R if you are to have access to the data sets from the textbook.

A.7 Form of the data sets

A.8 Names of the data sets

Data sets for exercises are named `ex`*cc.nn* where *cc* is two-digit chapter number and *nn* is the two-digit exercise number. Thus the data for exercise 27 in chapter 10 (p. 437) is called `ex10.27`. To provide the correct order when sorting the

data set names, single-digit chapter or exercise numbers have a zero prepended. The data for exercise 1 in chapter 6 (p. 265) is called `ex06.01`.

Data sets for examples in the text are named `xmp`*cc.nn*.

A listing of all the data sets in the package can be obtained with

```
> data(package = "Devore6")
```

A.9 Data sets as tables

All the data sets in the `Devore6` package are in a tabular form called a *data frame* in R. Rows correspond to observations and columns correspond to "variables". We use the tabular form even when there is only one variable.

The columns have names, usually reflecting the description of the data from the exercise or the example, although names like `C1` also occur frequently. (That name happens to be the default name of the first column assigned by another statistical computing system called Minitab.)

You can check the names and types of data in the data frame with `str`, which prints a concise summary of the structure of the data.

```
> data(xmp01.02)
> str(xmp01.02)

`data.frame':        27 obs. of  1 variable:
 $ strength: num  5.9 7.2 7.3 6.3 8.1 6.8 7 7.6 6.8 6.5 ...
```

This shows that the data for example 1.2 (p. 5) consists of 27 observations of 1 variable called `strength`, which is a numeric variable. The first several data values are printed so you can check that they correspond to the values in the text.

Most of the data sets discussed in chapters 1 to 8 are univariate (i.e. only one variable), numeric data like `xmp01.20`. There are a few examples of univariate, categorical data such as the health complaints discussed in exercise 1.29 (p. 27)

```
> data(ex01.29)
> str(ex01.29)

`data.frame':        60 obs. of  1 variable:
 $ complaint: Factor w/ 7 levels "B","C","F","J",..: 7 7 6 4 2 3 1 1 3 7 ...
```

These data are a set of observations of a variable that can take on only limited set of values named `B` for back pain, `C` for coughing, etc. In R such data are said to be a `factor`.

Some summary information about the variables in a data frame can be obtained with the `summary` function.

```
> summary(xmp01.02)
```

```
   strength
 Min.   : 5.90
 1st Qu.: 7.00
 Median : 7.70
 Mean   : 8.14
 3rd Qu.: 8.85
 Max.   :11.80
```

```
> summary(ex01.29)

 complaint
 B: 7
 C: 3
 F: 9
 J:10
 M: 4
 N: 6
 O:21
```

For a numeric variable `summary` provides a 'five-number' summary and the mean (making a total of 6 numbers in all). For a factor `summary` provides a frequency table.

A.10 Accessing individual variables

The `summary` function can be applied to entire data frames or to individual variables in a data frame. This is unusual. Most graphical or numerical functions apply to individual variables.

There are two ways to access a variable from within a data frame:

1. Use the name of the data set and the name of the variable separated by `$`
2. `attach` the data frame and use the variable name by itself

For example, the two ways to obtain the stem-and-leaf plot of the space shuttle launch ambient temperature data from example 1.1 are

```
> data(xmp01.01)
> str(xmp01.01)

`data.frame':        36 obs. of  1 variable:
 $ temp: int  84 49 61 40 83 67 45 66 70 69 ...
```

```
> stem(xmp01.01$temp)

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

  3 | 1
```

```
  4 | 059
  5 | 23788
  6 | 01136777789
  7 | 000023556689
  8 | 0134
```

and

```
> attach(xmp01.01)
> stem(temp)

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

  3 | 1
  4 | 059
  5 | 23788
  6 | 01136777789
  7 | 000023556689
  8 | 0134
```