Contents

Chapter 1

Data

Answers flagged with a ⋆ are available online for students.

1.1 Solutions to Problems

1.4 (p. 14) The only thing to remember is the placement of parentheses, and the need to use `*` for multiplication:

```
> 1 + 2*(3+4)
[1] 15
> 4^3 + 3^(2+1)
[1] 91
> sqrt((4+3)*(2+1))
[1] 4.583
> ( (1+2)/(3+4) )^2
[1] 0.1837
```

⋆ **1.5 (p. 14)** These would be $(2+3)-4$, $2+(3*4)$, $(2/3)/4$, and $2^{(3^4)}$; the last working right to left.

⋆ **1.6 (p. 14)** For example:

```
> p = c(2,3,5,7,11,13,17,19)
```

⋆ **1.7 (p. 15)** The `diff()` function returns the distance between fill-ups, so `mean(diff(gas))` is your average mileage per fill-up, and `mean(gas)` is the uninteresting average of the recorded mileage.

1.8 (p. 15) The data may be entered in using `c()` then manipulated in a natural way.

```
> x = c(2,5,4,10,8)
> x^2
[1]   4  25  16 100  64
> x - 6
[1] -4 -1 -2  4  2
> (x - 9)^2
```

```
[1] 49 16 25  1  1
```

⋆ **1.9 (p. 15)** Using `range()` makes these straightforward.

```
> mini = c(15.9, 21.4, 19.9, 21.9, 20.0, 16.5, 17.9, 17.5)
> range(mini)
[1] 15.9 21.9
> mean(mini)
[1] 18.88
> range(mini - mean(mini))
[1] -2.975  3.025
```

⋆ **1.10 (p. 15)** The sales figures are entered in with

```
> H2 = c(2700,2600, 3050, 2900, 3000, 2500, 2600, 3000, 2800,
+ 3200, 2800, 3400)
> cumsum(H2)
 [1]  2700  5300  8350 11250 14250 16750 19350 22350 25150 28350
[11] 31150 34550
> diff(H2)
 [1] -100  450 -150  100 -500  100  400 -200  400 -400  600
```

We see that there were 34,550 sold, December had the greatest increase (600), and June had the largest decrease (-500).

1.11 (p. 15) We compute the average number of calories due to fat and the average number of calories due to carbohydrates as follows:

```
> calories = c(2450, 2439, 2866, 2618)
> fat = c(37, 36.2, 34, 32.1)
> carbs = c(42.2, 43.1, 48.1, 50)
> carb.cals = carbs/100 * calories
> carb.cals
[1] 1034 1051 1379 1309
> fat.cals = fat/100 * calories
> fat.cals
[1] 906.5 882.9 974.4 840.4
```

The number of calories due to fat has decreased, and the number of calories due to carbohydrates has increased during this time span.

1.12 (p. 16) These can be done with

```
> rep(1,10)
> seq(1,99,by=2)
> rep(1:3,rep(3,3))
> rep(1:3,3:1)
> c(1:5,4:1)
```

1.13 (p. 16) These can be done with the following commands:

```
> fib = c(1,2,3,5,8,13,21,34)
> pos = 1:10
> recips = 1/pos
> cubes = (1:6)^3
> years = 1964:2003
> subway = c(14, 18, 23, 28, 34, 42, 50, 59, 66,
+ 72, 79, 86, 96, 103, 110)
```

```
> by25 = seq(0,1000, by=25)
```

1.14 (p. 22) First we store the data, then we analyze it.

```
> commutes =  c(17, 16, 20, 24, 22, 15, 21, 15, 17, 22)
> commutes[commutes == 24] = 18
> max(commutes)
[1] 24
> min(commutes)
[1] 15
> mean(commutes)
[1] 18.9
> sum(commutes >= 20)
[1] 5
> sum(commutes < 18)/length(commutes)
[1] 0.5
```

1.15 (p. 22) We need to know that the months with 31 days are 1, 3, 5, 7, 8, 10, and 12.

```
> dvd=c(79, 74, 161, 127, 133, 210, 99, 143, 249, 249, 368, 302)
> longmos = c(1, 3, 5, 7, 8, 10, 12)
> long = dvd[longmos]
> short = dvd[-longmos]
> mean(long)
[1] 166.6
> mean(short)
[1] 205.6
```

⋆ **1.16 (p. 22)** We use `scan()` as an alternative to `c()`.

```
> bill = scan()
1: 46 33 39 37 46 30 48 32 49 35 30 48
13:
Read 12 items
> sum(bill)
[1] 473
> max(bill)
[1] 49
> min(bill)
[1] 30
> sum(bill > 40)
[1] 5
> sum(bill > 40)/length(bill)
[1] 0.4167
```

⋆ **1.17 (p. 22)** Enter in the data as follows:

```
> x = c(0.57, 0.89, 1.08, 1.12, 1.18, 1.07, 1.17, 1.38, 1.441, 1.72)
```

Using `diff()` gives

```
> diff(x)
[1]  0.320  0.190  0.040  0.060 -0.110  0.100  0.210  0.061
[9]  0.279
```

So the jump between 1994 and 1995 was negative. The percentage difference is found by dividing by `x[-10]` and multiplying by 100. (Recall that `x[-10]` is all but the tenth (10th) number of `x`). The first year's jump was the largest.

```
> diff(x)/x[-10] * 100
```

```
[1] 56.140 21.348  3.704  5.357 -9.322  9.346 17.949  4.420
[9] 19.362
```

1.18 (p. 23) These result in 1 added to each entry in `x`; each entry in `y` multiplied by 2; 5 and 6 are the lengths of `x` and `y`; `x+y` will recycle the first element of `x` to add to the sixth element of `y`, resulting in 14 for the last entry (and a warning); `sum(x>5)` is 2 — the number of entries more than 5, `sum(x[x>5])` is 16 — the sum of 7 and 9; there are two entries in `x` more than 5 and one less than three so the sum returns 3; the third entry in `y` is 5; all *but* the third entry in `y` is the data vector containing 2, 3, 7, 11, 13; `y[x]` returns the first, third, fifth, seventh, and ninth entries of `y`, as `y` has length 6, the seventh and eighth are not defined resulting in `NA`; the last three elements of `y` are bigger than or equal 7 so these are returned.

⋆ **1.19 (p. 23)** The comparison of strings is done lexicographically. That is, comparisons are done character by character until a tie is broken. The comparison of characters varies due to the locale. This may be decided by ASCII codes—which yields alphabetically ordering—but need not be. See `?locale` for more detail.

1.20 (p. 29) Sorting with `decr=TRUE` give us

```
> sort(islands,decr=TRUE)[1:7]
         Asia        Africa North America South America
        16988         11506          9390          6795
   Antarctica        Europe     Australia
         5500          3745          2968
```

1.21 (p. 29) This can be answered using `sum()` and conditional statements as follows:

```
> sum(1 <= primes & primes <= 2003)
[1] 304
> sum(1 <= primes & primes <= 100)
[1] 25
> sum(100 <= primes & primes <= 1000)
[1] 143
```

1.22 (p. 29) The last construct, `primes[-1] - primes[-n]`, finds the differences between the primes. When this is 2 we have a twin primes. (This can also be done with `diff()`.)

1.23 (p. 29) The functions `length()`, `min()`, `max()`, and `sum()` give the answers.

```
> length(treering)
[1] 7980
> min(treering)
[1] 0
> max(treering)
[1] 1.908
> sum(treering > 1.5)
[1] 219
```

⋆ **1.24 (p. 30)** We can view the data with the commands

```
> mandms
                 blue brown green orange   red yellow
milk chocolate 10.00 30.00 10.00  10.00 20.00  20.00
Peanut         20.00 20.00 10.00  10.00 20.00  20.00
```

```
Peanut Butter  20.00 20.00 20.00   0.00 20.00  20.00
Almond         16.67 16.67 16.67  16.67 16.67  16.67
kid minis      16.67 16.67 16.67  16.67 16.67  16.67
```

From the table we see that orange is missing from peanut butter; almond and kid minis have equally likely colors; and brown is more likely than the other colors in milk chocolate packages.

1.25 (p. 30) To convert minutes into hours and minutes we can divide by 60 to get the hours, subtract the number of hours to get the fractional part, and then multiply by 60 to get the number of minutes.

```
> attach(nym.2002)
> length(time)                  # 1,000 times recorded (35,000 ran)
[1] 1000
> min(time)                     # 147 minutes
[1] 147.3
> 147.3/60                      # 2 hours
[1] 2.455
> (147.3/60 - 2) * 60           # and 27 minutes
[1] 27.3
> max(time)                     # 566 minutes
[1] 566.8
> 566.8/60                      # 9 hours
[1] 9.447
> (566.8/60 - 9)*60             # and 26 minutes
[1] 26.8
```

1.26 (p. 30) This is done by using `max()` or `min()` on the data set.

```
> max(rivers)
[1] 3710
> min(rivers)
[1] 135
```

⋆ **1.27 (p. 30)** We first add names to `uspop`.

```
> names(uspop) = seq(1790, 1970, by=10)
> uspop
> uspop                         # uspop is a time series
Time Series:
Start = 1790
End = 1970
...
> d = diff(uspop)
> names(d) = seq(1800, 1960, by=10) # add names
> d
...
 1800  1810  1820  1830  1840  1850  1860  1870  1880  1890
 1.38  1.93  2.40  3.26  4.20  6.10  8.20  8.40 10.40 12.70
 1900  1910  1920  1930  1940  1950  1960  1970
13.10 16.00 13.70 17.10  8.90 19.60 28.00 23.90
```

Scanning across we see that 1960 had the largest increase. To see if the numbers are always increasing we can call `diff()` again:

```
> d2 = diff(d)
> names(d2) = seq(1810, 1970, by=10)
> d2
```

```
...
 1810  1820  1830  1840  1850  1860  1870  1880  1890  1900
 0.55  0.47  0.86  0.94  1.90  2.10  0.20  2.00  2.30  0.40
 1910  1920  1930  1940  1950  1960  1970
 2.90 -2.30  3.40 -8.20 10.70  8.40 -4.10
```

The drop in population increase occurred during the times of World War I, the Great Depression, and the Vietnam War.

Chapter 2

Univariate data

2.1 Solutions to Problems

2.4 (p. 39) It contains the days when nothing much happened.

⋆ **2.5 (p. 39)** The graphics can be created as follows:

```
> browsers
         IE      Mozilla    Navigator Unidentified         <NA>
         86            4            5            1            4
> names(browsers) = c("IE","Mozilla","Navigator","Opera","Un.")
> browsers
       IE   Mozilla Navigator     Opera       Un.
       86         4         5         1         4
> barplot(browsers)
> pie(browsers)
> dotchart(browsers)
```

2.6 (p. 39) The answers come after the data is entered.

```
> mp3 = c(18,15,14.4,13.5,6.2)
> other = 100 - sum(mp3)
> mp3 = c(mp3, other)
> names(mp3) = c("Apple","RCE","Rio","iRiver","CL","other")
> 22000000 * mp3/100
  Apple     RCE     Rio  iRiver      CL   other
3960000 3300000 3168000 2970000 1364000 7238000
> barplot(mp3)
> dotchart(mp3)
> pie(mp3)
```

From the graphs we see that both the dot chart and the bar chart do a good job of presenting the data.

2.7 (p. 40) This graphic can be produced with

```
> dotchart(mtcars$mpg, labels= rownames(mtcars))
```

2.8 (p. 40) California is, as is indicated by

```
> sort(table(npdb$state))
```

⋆ **2.9 (p. 40)** This shows the number of doctors with that many awards. The great majority involve just one award

```
> table(table(npdb$ID))

   1    2    3    4    5    6    8   11   15   22   73
6105  235   12    4    7    1    1    1    1    1    1
> tmp = table(table(npdb$ID))
> sum(tmp[-1])/sum(tmp)         # percent 2 or more
[1] 0.04145
```

2.10 (p. 40) This is done using `names()` as follows:

```
> attach(MLBattend)
> yankees = wins[franchise == "NYA"]
> detach(MLBattend)
> names(yankees) = 1969:2000
> barplot(yankees, las = 2)     # turn names on side with las=2
> dotchart(yankees)
```

2.11 (p. 53) We have

```
> x = c(80,82,88,91,91,95,95,97,98,101,106,106,109,110,111)
> median(x)
[1] 97
```

⋆ **2.12 (p. 53)** All three measures should be similar, as the data is more or less symmetric about its center. A guess of 59—the identifiable median—seems reasonable. To check, we estimate the data, then use the correct functions.

```
> x = c(14, 18, 23, 28, 34, 42, 50, 59, 66, 72, 79, 86, 96, 103, 110)
> mean(x)
[1] 58.67
> median(x)
[1] 59
> mean(x, trim=.1)
[1] 58.15
```

⋆ **2.13 (p. 54)** The definition of the median is incorrect. Can you think of a shape for a distribution when this is actually okay?

⋆ **2.14 (p. 54)** The median is lower for skewed-left distributions. It makes an area look more affordable. For exclusive listings, the mean is often used to make an area seem more expensive.

2.15 (p. 54) We do the usual `sum(...)/length(...)` formula:

```
> sum(pi2000 <= 3)/length(pi2000) * 100
[1] 39.5
> sum(pi2000 >= 5)/length(pi2000) * 100
[1] 50.75
```

2.16 (p. 54) If the data set is loaded, using `library(UsingR)`, these can be found with:

```
> sum(rivers < 500)/length(rivers)
[1] 0.5816
```

```
> sum(rivers < mean(rivers))/length(rivers)
[1] 0.6667
> quantile(rivers,0.75)
75%
680
```

⋆ **2.17 (p. 54)** First grab the data and check the units (minutes). The top 10% is actually the 0.10 quantile in this case, as shorter times are better.

```
> times = nym.2002$time
> range(times)                  # looks like minutes
[1] 147.3 566.8
> sum(times < 3*60)/length(times) * 100 # 2.6% beat 3 hours
[1] 2.6
> quantile(times,c(.10, .25))   # 3:28 to 3:53
  10%   25%
208.7 233.8
> quantile(times,c(.90))        # 5:31
  90%
331.8
```

It is doubtful that the data is symmetric. It is much easier to be relatively slow in a marathon, as it requires little talent and little training—just doggedness.

2.18 (p. 54) Use the functions:

```
> mean(rivers)
[1] 591.2
> median(rivers)
[1] 425
> mean(rivers, trim=.25)
[1] 449.9
```

Yes, the data is skewed to the right.

⋆ **2.19 (p. 54)** We see

```
> stem(islands)
...                             # quite skewed
> mean(islands)
[1] 1253
> median(islands)
[1] 41
> mean(islands,trim=0.25)
[1] 51.08
```

The data set is quite skewed due to the seven continents.

⋆ **2.20 (p. 54)** We can find the *z*-score for Barry Bonds using the name as follows:

```
> (OBP['bondsba01']- mean(OBP)) / sd(OBP)
bondsba01
     5.99
```

2.21 (p. 54) Using `scale()` gives us the *z*-scores.

```
> z = scale(x)
> mean(z)
[1] -1.439e-16                  # this is 0
```

```
> sd(z)
[1] 1
```

Alternatively, we could have found the z-scores directly with

```
> z = (x - mean(x))/sd(x)
```

⋆ **2.22 (p. 55)** The function `mad()` returns the median deviation from the median. It is a measure of variation similar to the mean-squared distance from the mean (basically the variance) but more robust. For the long-tailed `exec.pay` data set we see that the standard deviation is very skewed by the outliers:

```
> attach(exec.pay)
> mad(exec.pay)
[1] 20.76
> IQR(exec.pay)
[1] 27.5
> sd(exec.pay)                  # skewed
[1] 207.0
```

2.23 (p. 55) As this distribution has a long tail, we find that the mean is much more than the median.

```
> amt = npdb$amount
> summary(amt)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
      50     8750    37500   166000   175000 25000000
> sum(amt < mean(amt))/length(amt) * 100
[1] 74.9
```

2.24 (p. 55) The function `summary()` shows the mean to be much more than the median. There are a few on the cabinet who saved *a lot* on taxes, skewing the mean amount.

```
> savings = cabinet$est.tax.savings
> summary(savings)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     30     445    6590   42200   50400  332000
```

2.25 (p. 55) The coefficient of variation is found using `sd()` and `mean()`.

```
> sd(rivers)/mean(rivers)
[1] 0.8354
> data(pi2000)
> sd(pi2000)/mean(pi2000)
[1] 0.6323
```

2.26 (p. 56) The lag plot of random data looks scattered. The lag plot of the sinusoidal data sits on a curve.

2.27 (p. 56) To prove these, write out the sums, factor out a in the first case, then regroup the summands in the second.

2.28 (p. 56) This follows from the facts $\sum c = nc$, $\sum(x_i + y_i) = \sum x_i + \sum y_i$, and $\sum x_i = n\bar{x}$.

2.29 (p. 56) We first rewrite the following:

$$\sum(x_i - \bar{x})^2 = \sum x_i^2 - 2x_i\bar{x} + \bar{x}^2 = \sum x_i^2 - 2n\overline{x\bar{x}} + \bar{x}^2 = \sum x_i^2 - n\bar{x}^2 = n\overline{x^2} - n(\bar{x})^2.$$

Divide by $n - 1$ to finish.

2.30 (p. 66) For `firstchi` this is done as follows:

```
> hist(firstchi)                # looks like 25 or so
> mean(firstchi)                # we were pretty close
[1] 23.98
```

2.31 (p. 66) It is *very* unlikely that you would get the same histogram—unless you forgot to rerun the command defining `x`. However, the new histogram should have a *similar* shape to the old one.

2.32 (p. 66) This is done with

```
> hist(pi2000-.1, prob=TRUE)
> lines(density(pi2000))
```

This distribution is "flat," as each digit is more or less equally likely. We subtracted 0.1 so the bins for 0 and 1 are not combined, something that is seen when making the histogram at first. Alternately, we could specify the argument `breaks=0:10-.5`.

2.33 (p. 66) This is done as follows:

```
> hist(normtemp$temperature)    # looks like its 98.2 -- not 98.6
> mean(normtemp$temperature)
[1] 98.25
```

⋆ **2.34 (p. 66)** The graphics can be produced with these commands:

```
> data(DDT, package="MASS")     # load from MASS package
> hist(DDT)
> boxplot(DDT)
```

The histogram shows the data to be roughly symmetric, with one outlying value, so the mean and median should be similar and the standard deviation about three-quarters the IQR. The median and IQR can be identified on the boxplot giving estimates of 3.2 for the mean and a standard deviation a little less than 0.5. We can check with this command:

```
> c(mean(DDT), sd(DDT))
[1] 3.3280 0.4372
```

⋆ **2.35 (p. 66)** First assign names. Then you can access the entries using the repective state abbreviations.

```
> names(state.area) = state.abb
> state.area['NJ']
  NJ
7836
> sum(state.area < state.area['NJ'])/50 * 100
[1] 8
> sum(state.area < state.area['NY'])/50 * 100
[1] 40
```

2.36 (p. 66) The histogram shows a skewed shape. A few stragglers at the end make the right tail a bit longer.

2.37 (p. 67) For a heavily skewed-right data set, such as this, the mean is significantly more than the median due to a relatively few large values. The median more accurately reflects the bulk of

the data. If your intention were to make the data seem outrageously large, then a mean might be used.

2.38 (p. 67) We make the histogram as follows:

```
> attach(babyboom)
> hist(diff(running.time))
> detach(babyboom)
```

The data is not symmetric. Rather, the frequencies decay steadily as time increases.

⋆ **2.39 (p. 67)** The histograms are all made in a similar manner to this:

```
> hist(hall.fame$HR)
```

The home run distribution is skewed right, the batting average is fairly symmetric, and on-base percentage is also symmetric but may have longer tails than the batting average.

2.42 (p. 67) The histograms can be made in a manner similar to this:

```
> attach(cfb)
> hist(AGE)
```

After making these, we see that `AGE` is short-tailed and somewhat symmetric; `EDUC` is not symmetric; `NETWORTH` is very skewed (some can get *really* rich, some can get *pretty* poor, most close to 0 on this scale; and `log(SAVING + 1)` is symmetric except for a spike of people at 0 who have no savings (the actual data is skewed—the logarithm changes this).

2.43 (p. 67) The histogram shows a fairly symmetric distribution centered near 8. (Star brightness is measured on a logarithmic scale—a difference of 5 is a factor of 100 in terms of brightness. Thus, the actual brightnesses are skewed.)

```
> hist(brightness)
```

2.44 (p. 68) The following commands produce the graphics:

```
> par(mfrow = c(2,1))
> hist(lawsuits)
> boxplot(lawsuits, horizontal=TRUE)
```

As well, similar commands produce the graphics for `log(lawsuits)`. Guessing the center is much easier after taking the logarithm.

⋆ **2.45 (p. 68)** After a log transform the data looks more symmetric. If you find the median of the transformed data, you can take its exponential to get the median of the untransformed data. Not so with the mean.

2.46 (p. 68) unlike the standard deviation, this formula uses the sign of $x_i - \bar{x}$. When the data is skewed right, say, the positive values of $x_i - \bar{x}$ will tend to be larger than the negative ones, making the formula for the skew large and positive. For symmetric data, we'd expect more cancellations, keeping the value closer to 0.

Chapter 3

Bivariate data

3.1 Solutions to Problems

⋆ **3.2 (p. 75)** The data can be entered using `rbind()` as

```
> all = c(125,210,375,475,590,700)
> spam = c(50,110,225,315,390,450)
> tab = rbind(spam,all)
> dimnames(tab) = list(c("spam","Total"),year=2000:2005)
```

To make the barplot, we make a new table, replacing `all` with `all-spam`, which is the number of e-mails that are not spam.

```
> newtab = rbind(spam,all-spam)
> dimnames(newtab) = list(c("spam","notspam"),year=2000:2005)
> barplot(newtab,legend=TRUE)
```

⋆ **3.3 (p. 76)** We can tabulate the coins then multiply by the amounts to find the total amount of money in the change bin.

```
> table(coins$value)
0.01 0.05  0.1 0.25
 203   59   42   67
> table(coins$value) * c(0.01, 0.05, 0.1, 0.25)
 0.01  0.05   0.1  0.25
 2.03  2.95  4.20 16.75
> sum(table(coins$value) * c(0.01, 0.05, 0.1, 0.25))
[1] 25.58
```

The barplot can be produced with the command

```
> barplot(table(coins$year))
```

It shows a generally increasing trend, as it is more likely to contain more recent coins. Finally, to get the coins by decade, a first attempt would be to try

```
> table(cut(coins$year, breaks = seq(1920,2010,by=10)))
```

But this has ranges like 1921-1930. Using the argument `right=FALSE` will cause the intervals to take the form $[a, b)$, for example, 1920-1929.

⋆ **3.4 (p. 76)** The barplots are made with the commands

```
> barplot(t(dvdsales), beside=T)
```

Reversing the time so that the numbers increase can be done using notation for data frames

introduced in Chapter 4:

```
> barplot(t(dvdsales[7:1,]), beside=T)
```

3.5 (p. 76) First `attach()` the data, then

```
> attach(florida)
> tab = rbind(BUSH/(BUSH+GORE),GORE/(BUSH+GORE))
> dimnames(tab) = list(candidate=c("Bush","Gore"),
+ County=as.character(County))  # factor needs coercion
> barplot(tab,las=3)            # las=3 puts names vertical
> detach()
```

Clearly, there is much variation from the 50% proportion. Some counties are Democratic strongholds' and some Republican ones.

3.6 (p. 76) This can be achieved with

```
> ca = c(76,70,68,52,48,46)
> nca = 100 - ca
> barplot(rbind(ca,nca), names.arg=1997:2002,
+ legend=c("cash","non-cash"))
> title("Shift in what welfare provides")
```

⋆ **3.7 (p. 76)** The table is made with

```
> data(UScereal, package="MASS")
> attach(UScereal)
> table(shelf,mfr)
```

It appears that `Q` (Quaker Oats) is underrepresented on the shelf 1, and `R` (Ralston Purina) is overrepresented there. Shelf 1 is less likely to be seen by casual consumers, so it has less chance of encouraging impulse buying.

3.8 (p. 81) This can be done with

```
> boxplot(time[control == "C"], time[control == "T"])
```

The spread is similar, but the center of the cell phone users is greater.

⋆ **3.9 (p. 81)** The boxplots are produced with a command like:

```
> attach(twins)
> boxplot(Foster, Biological, names=c("Foster","Biological"))
> detach(twins)
```

The centers and spreads appear to be similar.

⋆ **3.10 (p. 81)** The graphs are produced as follows:

```
> attach(stud.recs)
> plot(density(sat.m))
> lines(density(sat.v),lty=2)
> qqplot(sat.m,sat.v)
> detach(stud.recs)
```

We see from the densityplots that the center for `sat.v` appears to be larger. From the q-q plot it looks like the two distributions have similar shapes, as the points align fairly well.

3.11 (p. 81) Make the boxplots with the following commands:

```
> attach(morley)
> boxplot(list(one=Speed[Expt == 1], two=Speed[Expt == 2]))
```

```
> detach(morley)
```

From the boxplots we see similar spreads but different centers.

⋆ **3.12 (p. 82)** We can split the data up using the condition `gender==1` as follows:

```
> attach(normtemp)
> male = temperature[gender == 1]
> female = temperature[gender == 2]
> boxplot(list(male=male,female=female))
> detach()
```

From the graphic it appears that the centers of the distributions are different.

3.13 (p. 89) The histogram shows a bimodal distribution. One part of town had home values increase dramatically on a percentage basis, while the other side did not.

```
> attach(homedata)
> hist(y2000/y1970, probability=TRUE)
> lines(density(y2000/y1970))
> detach(homedata)
```

⋆ **3.14 (p. 89)** The correlations are found using `cor()`:

```
> names(galton)
[1] "child"  "parent"
> with(galton, cor(child,parent))
[1] 0.4588
> with(galton, cor(child,parent,method="spearman"))
[1] 0.4251
```

3.15 (p. 89) The scatterplot and correlation are found as follows:

```
> attach(normtemp)
> plot(temperature, hr)
> cor(temperature, hr)
[1] 0.2537
> detach(normtemp)
```

Although there appears to be a trend between the two, the correlation is not large.

3.16 (p. 89) The scatterplot and correlation are found with the commands

```
> attach(fat)
> plot(BMI, body.fat)
> cor(BMI, body.fat)
[1] 0.728
> detach(fat)
```

Except for a few outliers in the `BMI`, the data shows a fairly strong correlation.

⋆ **3.17 (p. 89)** The plot can be made with the commands

```
> attach(twins)
> plot(Foster, Biological)
> detach()
```

Based on this, a guess of 0.8 for the Pearson coefficient and a similar value for the Spearman coefficient seem reasonable. A check can be done with

```
> cor(Foster, Biological)
[1] 0.882
> cor(Foster, Biological,method="spearman")
```

```
[1] 0.8858
```

3.18 (p. 89) The scatterplots are produced using the model formula interface with the following commands:

```
## x77 attached as per instruction in problem
> plot(Population ~ Frost)
> plot(Murder ~ Population)
> plot(Population ~ Area)
> plot(Income ~ HS.Grad)
> detach()
```

We see some correlations between the variables. These do not imply causal relationships. Generally speaking, states with more frost have smaller populations, states with more population have higher murder rates, area and population is hard to read, and income increases with graduation percentages.

3.19 (p. 89) We might think, at first, that time would increase dramatically as age does, but the actual correlation is quite low.

```
> attach(nym.2002)
> names(nym.2002)
[1] "place"  "gender" "age"    "home"   "time"
> cor(age,time)
[1] 0.1899
```

3.20 (p. 90) These are the coordinates of the center for each of the fifty states in the United States.

⋆ **3.21 (p. 90)** The correlation is found with

```
> with(batting, cor(HR,SO))
[1] 0.7085
```

Although the two variables are quite correlated, there is a likely lurking variable—the number of at bats—that would explain the correlation.

3.22 (p. 90) Using `jitter()` with a factor of 5 mixes the values up quite well.

⋆ **3.24 (p. 102)** We fit the model and then make the prediction as follows:

```
> res = lm(abdomen ~ wrist, data = fat)
> predict(res, data.frame(wrist=17))
[1] 83.75
```

⋆ **3.25 (p. 103)** The `wtloss` data can be plotted as follows:

```
> attach(wtloss)
> cor(Weight, Days)             # close to -1
[1] -0.9853
> plot(Days, Weight)            # kinda linear
> res = lm(Weight ~ Days)
> abline(res)                   # add regression line
> plot(Days, residuals(res))    # A trend
> detach(wtloss)
```

We see from the scatterplot that a linear model may not explain the data well, despite the correlation coefficient that is close to -1. The residual plot amplifies this observation. This is to be expected as it is initially easy to lose a large amount of weight, but this gets progressively harder

as there is less excess weight to shed.

3.26 (p. 103) The plots can be produced with these commands:

```
## data frame x77 defined in problem
> plot(Illiteracy ~ HS.Grad, data = x77)
> plot(Life.Exp ~ Murder, data = x77)
> plot(Income ~ Illiteracy, data = x77)
```

All three plots show negative correlations. Are there confounding variables? Certainly, although it is not surprising that a higher high school graduation rate is directly correlated with lower illiteracy. It may seem obvious that a higher murder rate would lead to a loss of life expectancy, yet this is unlikely to make much of a difference, as murders are relatively rare. As well, although it seems that income should go down if illiteracy goes up, the relationship may be confounded by state wealth, spending on schools, or other unspecified variables.

3.27 (p. 103) The predicted amount and residual can be found using `predict()`:

```
> res = lm(RBI ~ HR, data=batting)
> predict(res, data.frame(HR=33))
[1] 104.1
> 98 - predict(res, data.frame(HR=33))
[1] -6.11
```

⋆ **3.28 (p. 103)** The calculations are performed as follows:

```
> lm(Female ~ Male, data = too.young)

Call:
lm(formula = Female ~ Male, data = too.young)
Coefficients:
(Intercept)         Male
      5.472        0.575
> plot(Female ~ Male, data = too.young)
> abline(lm(Female ~ Male, data = too.young))
> abline(7,1/2,lwd=2)
```

The result from the data set is a little more conservative than the rule of thumb, except for the teenage years.

3.29 (p. 103) The regression line and predictions are carried out using these commands:

```
> plot(price ~ carat, data = diamond, pch=5)
> res = lm(price ~ carat, data = diamond)
> abline(res)
> predict(res, data.frame(carat=1/3))
[1] 980.7
```

⋆ **3.30 (p. 104)** The slope appears in the output of `lm()`. The scatterplot shows a linear trend between the transformed variables that is not apparent in a scatterplot of the original variables.

```
> library(MASS)                 # loads data set and lqs()
> names(Animals)
[1] "body"  "brain"
> plot(log(brain) ~ log(body), data = Animals)
> res = lm(log(brain) ~ log(body), data = Animals)
> res
```

```
Call:
lm(formula = log(brain) ~ log(body), data = Animals)

Coefficients:
(Intercept)    log(body)
      2.555        0.496
```

To compare to the output of `lqs()`.

```
> lqs(log(brain) ~ log(body), data = Animals)
Call:
lqs.formula(formula = log(brain) ~ log(body), data = Animals)

Coefficients:
(Intercept)    log(body)
      1.816        0.776

Scale estimates 0.464 0.463
```

There are three influential points in the scatterplot of the log-transformed data causing the big change in the slope of the regression line.

3.31 (p. 104) This can be down with

```
> plot(log(time) ~ voltage, data=breakdown)
> res = lm(log(time) ~ voltage, data=breakdown)
> abline(res)
> res
Call:
lm(formula = log(time) ~ voltage, data = breakdown)
Coefficients:
(Intercept)      voltage
     18.946       -0.507
```

The spread around the regression line for each level of the voltage is symmetric and not too spread out.

⋆ **3.32 (p. 104)** From the scatterplot we see that four temperatures were used. It appears that only one data point is associated with a temperature of 150°C, but in fact there were ten:

```
> table(motors$temp)

150 170 190 220
 10  10  10  10
```

It is hard to tell whether the values for this temperature fit a linear model. Assuming they do, we can fit the model with

```
> res = lm(time ~ temp,  data=motors)
> res

Call:
lm(formula = time ~ temp, data = motors)

Coefficients:
(Intercept)         temp
      22999         -107
```

Then predictions can be made using `predict()`:

```
> predict(res, newdata=data.frame(temp=210))
[1] 580.6
```

⋆ **3.33 (p. 104)** We can make the plot and add the lines with

```
> attach(mw.ages)
> plot(1:103, Male + Female, type="l")
> lines(supsmu(1:103,Male),lty=2)
> lines(supsmu(1:103,Female),lty=3)
```

The town is a suburb of New York City. Most twenty somethings move away, returning after they are ready to settle down.

Chapter 4

Multivariate Data

4.1 Solutions to Problems

⋆ **4.1 (p. 112)** The tables can be made using `table()`. Different tables are produced for each level of the last variable specified. For example, to answer the first question we have:

```
> attach(samhda)
> table(amt.smoke, marijuana, gender)
, , gender = 1

         marijuana
amt.smoke 1   2   9
       1   13   3   0
       2    3   0   0
       3    5   0   0
       4    5   1   0
       5    2   5   0
       6   11  13   0
       7   25  38   1
       98   6 148   0
       99   4   4   2

...
> detach(samhda)
```

By looking at the help page for `samhda` to see how the variables are coded, we can see that a `marijuana` value of 2 is "not smoking", and an `amt.smoke` value of 7 is "no smoking in the last 30 days," whereas 1 indicates "smoking every day." The table for males (`gender` equal 1) shows that in this data set marijuana smokers are much more likely to be among the heavier cigarette smokers.

⋆ **4.2 (p. 112)** Once the variables `mpg` and `price` are made, this can be done with

```
> ftable(table(mpg, price, Cars93$Type))
```

⋆ **4.3 (p. 112)** The scatterplot can be made with

```
> plot(MPG.city ~ Price, pch=as.numeric(Type), data=Cars93)
```

There is a relationship between price and mileage. Additionally, you can note that certain types

are more likely to be cheap or expensive.

4.4 (p. 112) The scatterplot can be produced with the commands

```
> plot(Driver.deaths ~ Other.deaths, data=carsafety,
+ pch=as.numeric(type))
> x = carsafety$type
> legend(20,140, legend=levels(x), pch=1:length(x))
```

From the graphic, we can see that points for SUVs and Trucks are generally below the diagonal, and points for subcompacts are generally above. The larger the car, the greater the ratio of other deaths to driver deaths. Some might take this as safer, but the SUVs have higher death rates due to rollovers.

⋆ **4.5 (p. 113)** The boxplots are generated quickly, as `cancer` is a list of variables.

```
> boxplot(cancer)
```

From the graphic, we can see that breast cancer has the longest tail (a good thing for those with breast cancer as it means there is a chance of a long life still), bronchus cancer the smallest spread (not good for bronchus cancer sufferers), and that the centers are not comparable.
For more on this, please read the informative essay *The Median Isn't the Message* by Stephen J. Gould (`http://www.cancerguide.org/median_not_msg.html`).

⋆ **4.6 (p. 113)** A table shows the relationship between manufacturer and vitamin type by shelf location:

```
## load and attach data set
> library(MASS); attach(UScereal)
> table(mfr, vitamins, shelf)
, , shelf = 1

, , shelf = 3

   vitamins
mfr 100% enriched none
  G 3    6        0
  K 2    8        0
  N 0    1        0
  P 0    6        0
  Q 0    1        1
  R 0    1        0
```

The most obvious relationship is that 100% fortified cereal appears only on the highest shelf. This leaves space for the kid-desirable, sugar-laden cereals on the coveted second shelf.
The bubble plot is produced with the commands

```
> plot(calories ~ sugars, data=UScereal, cex=2*sqrt(fat))
```

There appears to be a linear relationship between sugars and calories, as expected. The larger bubbles appear to be above the smaller ones, indicating that more fat implies more calories as well.
A pairs plot (`pairs(UScereal)`) shows many relationships. For example, the `fibre`, `shelf` plot shows some relationship. A better plot than the scatterplot for showing a factor and a numeric variable is the boxplot:

```
> boxplot(fibre ~ shelf, data=UScereal)
```

The high-fiber cereals live on the top shelf for presumably the same reason that the fortified cereals do.

4.7 (p. 124) First attach the variables, then use `order()` to sort by `wt`:

```
> attach(mtcars)
> mtcars[order(wt,decreasing=TRUE),]
                     mpg cyl  disp  hp drat    wt  qsec
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  ...
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98
...
```

To compare for different indices, we can use the negative indexing convention, or create a new factor and a model formula.

```
> x = c(1:3,8:14,18:21,26:28,30:32)
> boxplot(mpg[x],mpg[-x],names=c("import","non-import"))
> f = rep(0,length(mpg))
> f[x]=1
> f = factor(f,labels=c("non-import","import"))
> boxplot(mpg ~ f)
```

Regardless of the method, the imported cars have better gas mileage.

Finally, the `pch=` argument can be used to label the points differently:

```
> plot(mpg ~ wt, data=mtcars, pch=cyl)
> tmp = sort(unique(cyl))
> legend(4,30,legend=tmp,pch=tmp)
> detach(mtcars)
```

We can see that mileage drops off with weight, and that heavier cars have more cylinders.

4.8 (p. 124) This can be done with matrix notation:

```
> df = cfb[cfb$INCOME > 0 & cfb$NETWORTH < 0, ]
> dim(df)
[1] 66 14
```

4.9 (p. 125) The plots can be made as follows:

```
> hist(hall.fame$HR)
> hf = subset(hall.fame, select=c("AB","hits","HR","RBI"),
+ subset = Hall.Fame.Membership != "not a member")
> boxplot(lapply(hf,scale))
```

The distribution of home runs has the most skew.

⋆ **4.10 (p. 125)** The data set is stored in a matrix, so we can reverse the years using indexing.

```
> barplot(t(dvdsales[7:1,]), beside=TRUE)
```

Using `apply()` on the rows, we get the yearly amounts:

```
> apply(dvdsales, 1, sum)
    2004     2003     2002     2001     2000     1999     1998
      NA 21994389 17089823 12706584  8498545  4019389  1089261
    1997
      NA
> apply(dvdsales, 1, function(x) sum(x,na.rm=TRUE)) # avoid NA
    2004     2003     2002     2001     2000     1999     1998
 5957387 21994389 17089823 12706584  8498545  4019389  1089261
    1997
  315136
```

Applying `sum()` to the columns gives the monthly sales. We do so only for years that have complete data.

```
> apply(dvdsales[2:6,], 2, sum)
    JAN     FEB     MAR     APR     MAY     JUN     JUL     AUG
2407354 2545896 4761189 3677919 3539725 5634658 3521254 3812934
    SEP     OCT     NOV     DEC
8362963 7551936 9323618 9169284
```

November is the big winner. (Christmas shopping?)

4.11 (p. 125) The row mean gives the monthly average, although it is not a weighted average, as it should be.

```
> apply(ewr[,3:10], 1, mean)
```

⋆ **4.12 (p. 125)** As the data is stored in a list, the boxplots are produced without any additional work:

```
> boxplot(u2)
```

However, we see that the titles are not legible. We can plot the titles along the side and give enough room for them to print with the commands:

```
> par(mar=c(5,15,4,2))          # increase to 15 from default 4
> boxplot(u2, las=2, horizontal=TRUE)
> par(mar=c(5,4,4,2))           # restore to original settings
```

The mean and median for each album can be found using `sapply()`:

```
> sapply(u2,mean)
> sapply(u2,median)
```

Comparisons can be made by subtracting the two. Finally, after unlisting we can sort to find the lengths:

```
> rev(sort(unlist(u2)))[1:3]
                 Zooropa. Lemon                Zooropa. Zooropa
                            416                             390
Rattle & Hum. All I Want Is You
                            390
```

4.13 (p. 125) The plots may be drawn one at a time as follows

```
> attach(normtemp)
> plot(density(temperature[gender==2])) # ladies first
> lines(density(temperature[gender==1]), lty=2)
> legend(96,0.5, legend=c("Females","Males"), lty=1:2)
> detach()
```

The densityplots show similar shapes but perhaps shifted centers.

4.14 (p. 125) Leaving an index blank implies that all the indices should be used, so this returns the entire data frame, same as `df` by itself.

⋆ **4.15 (p. 130)** The boxplots are easily made with the formula interface:

```
> boxplot(attendance ~ year, data = MLBattend)
```

Looking at the drops in attendance shows that 1972, 1981, and 1994 are likely candidates.

⋆ **4.16 (p. 130)** This can be done as follows:

```
> before.72 = year < 72 & year > 0
> tapply(runs.scored[before.72],league[before.72],mean)
   AL    NL
653.9 673.9
> tapply(runs.scored[!before.72],league[!before.72],mean)
```

```
   AL    NL
718.7 675.6
```

Alternatively, `tapply()` can use two factors to split up the results in one command.

```
> tapply(runs.scored,list(factor(before.72),league),mean)
         AL    NL
FALSE 718.7 675.6
TRUE  653.9 673.9
```

The `tapply()` syntax above uses a list of factors to split up the value of `runs.scored`.

⋆ **4.17 (p. 131)** The following commands will create the boxplots

```
> attach(npdb)
> tmp = split(amount,ID)
> df = data.frame(sum=sapply(tmp,sum),number=sapply(tmp,length))
> boxplot(sum ~ number, data = df) ## or even better
> boxplot(log(sum) ~ number, data = df)
> detach(npdb)
```

Based on the latter graph, the two or more awards appear higher; the total amounts aren't even comparable. To see this, again we can use `split()` and then `sapply()` as follows:

```
> attach(df)
> tmp =  sapply(split(sum,number),sum)
> tmp
         1          2          3          4          5
1034406350   81199650    4400500    2593750    1995000
         6          8         11         15         22
   1090000     960000     243550    1492500     855250
        73
    813500
> tmp/sum(tmp)
         1         2         3         4         5         6
0.9153633 0.0718549 0.0038941 0.0022953 0.0017654 0.0009646
         8        11        15        22        73
0.0008495 0.0002155 0.0013207 0.0007568 0.0007199
```

(An obvious complaint—that there aren't enough years in the data set to catch all the repeat offenders—is valid. The full data set shows a much less skewed picture.)

4.18 (p. 131) Using a model formula allows this to be done quickly, as with:

```
> boxplot(Speed ~ Expt, data=morley)
```

From the boxplots, there appears to be a difference in the centers of the distribution of measurements from experiment to experiment. As for the spreads, they appear to be more similar.

4.19 (p. 131) Looking at the boxplots with

```
> boxplot(weight ~ group, data = PlantGrowth)
```

We see the spreads are similar, but the centers appear to differ.

4.20 (p. 131) The following commands will create the boxplots:

```
> boxplot(count ~ spray, data=InsectSprays,
+ subset=spray %in% c("C","D","E"))
```

4.21 (p. 131) The following commands produce the plot:

```
> library(MASS);                # load data set
> pairs(~ calories + carbo + protein + fat + fibre + sugars, data=UScereal)
```

Most show a linear trend.

⋆ **4.22 (p. 134)** This command will produce the figure:

```
> xyplot(weight ~ height | cut(age,36*(0:4)), data = kid.weights)
```

4.23 (p. 134) Try this plot to determine it:

```
> bwplot(weight ~ gender | cut(age/12,3*(0:4)), data = kid.weights)
```

4.24 (p. 135) The boxplots may be produced using `bwplot()`:

```
> library(lattice)
> bwplot(income ~ race, data=female.inc) # too skewed
> bwplot(log(income) ~ race, data=female.inc) # better
```

After taking a logarithm, we see that, although the medians are comparable, the bottom 25% is not. The summary statistics can be produced using `tapply()`

```
> with(female.inc, tapply(income, race, summary))
$black
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    311    4350   13600   19500   27900  117000

$hispanic
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    311     311    6210   14900   20900   78600

$white
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    311    6210   13600   22000   30900  651000
```

⋆ **4.25 (p. 135)** These graphs can be produced as follows:

```
> bwplot(len ~ factor(dose) | supp, data = ToothGrowth)
> bwplot(len ~ supp | factor(dose), data = ToothGrowth)
```

4.26 (p. 135) Subcompacts have the most spread for driver deaths, but when the two are added together the difference in spreads is much less.

```
> bwplot(Driver.deaths ~ type, data=carsafety)
> bwplot(Driver.deaths + Other.deaths~ type, data=carsafety)
```

⋆ **4.27 (p. 135)** Scatterplots are made using `xyplot()`.

```
> xyplot(circumference ~ age | Tree, data = Orange)
```

From these, the growths seem to be similar.

4.28 (p. 135) The scatterplot shows a linear trend, with the hand size of females being generally smaller.

```
> xyplot(Wr.Hnd ~ NW.Hnd | Sex, data=survey)
```

The boxplots produced with

```
> bwplot(Pulse ~ Smoke | Sex, data=survey)
```

do not show a clear trend.

Chapter 5

Describing populations

5.1 Solutions to Problems

5.1 (p. 148) There are four equally likely outcomes: $\{HH, HT, TH, TT\}$. Thus the probability is $1/4$ that X or Y is 0, $1/4$ that X or Y is 2 and $1/2$ that X or Y is 1. That is, they are *identically distributed* (but not identical).

5.2 (p. 148) A command to simulate one value of X is

```
> max(sample(1:6,1),sample(1:6,1))
[1] 4
```

Repeat this five times. Alternately, knowing that the distribution of X is $\mathsf{P}(X = k) = (2k-1)/36$, we can use `sample()`, as follows, to generate a single value (this method easily extends to finding five values).

```
> k = 1:6
> sample(k, 1, prob= (2*k-1)/36)
[1] 6
```

⋆ **5.3 (p. 148)** The frequencies in `Balls` may be used for the probabilities, as shown:

```
> with(nba.draft,sample(Team,1,prob=Balls))
[1] Houston
13 Levels: Atlanta Chicago Cleveland Denver ... Washington
```

5.4 (p. 148) If we plot the density, f, over the interval $[0, \sqrt{2}]$ we see that the area associated with $\mathsf{P}(X \le b)$ is a right triangle with area $(1/2)b^2$.

5.5 (p. 148) Drawing the density of X shows that $\mathsf{P}(X \le b)$ is answered by the area of a rectangle with length b and height 1. So $\mathsf{P}(X \le b) = b$. Solving, we get $b = p$ for p in $[0,1]$.

5.6 (p. 148) Drawing the density of X shows that $\mathsf{P}(X \le b)$ is answered by the area of a triangle with length b and height b. So $\mathsf{P}(X \le b) = (1/2)b^2$. Solving we get $b = \sqrt{2p}$ for p in $[0,1]$.

5.7 (p. 149) No. For example, if we know that $X = 1$, then the distribution of Y has changed, as we

now know $Y = 1$. Algebraically, this can be shown as

$$\mathsf{P}(X = 1, Y = 0) = 0, \quad \text{but} \quad \mathsf{P}(X = 1)\mathsf{P}(Y = 0) = 1/2 \cdot 1/4.$$

5.8 (p. 158) This is a binomial problem. Let a roll of 4, 5, or 6 be considered a success. Then we are asked to find the probability of 3 or more success in 5 trials with success probability 1/2. This is found with

```
> sum(dbinom(3:5, size=5,prob=1/2))
[1] 0.5
```

⋆ **5.9 (p. 158)** If we think of each try as independent and identically distributed, then the number of strikes in 12 tries is binomial. Thus, we have

```
> dbinom(12,12,p=.3)
[1] 5.314e-07
```

⋆ **5.10 (p. 158)** We can answer this using a sum with the following:

```
> sum(dbinom(k, size=100000, prob=1/2))
[1] 0.7952
```

5.11 (p. 158) Using a binomial model with $n = 4$ and $p = 1/3$, this is $\mathsf{P}(X = 4)$ or

```
> dbinom(4,4,1/3)
[1] 0.01235
```

⋆ **5.12 (p. 158)** This is known as de Mere's problem. Although the expected number of successes in each case is identical, as $24/36 = 4/6$, the probabilities differ. Namely,

```
> 1 - pbinom(0,4,1/6)           # one or more 6 in 4 rolls
[1] 0.5177
> 1 - pbinom(0,24,1/36)         # one or more double sixes
[1] 0.4914
```

The rolling of four dice is better than even, and the rolling of 24 is worse.

5.13 (p. 158) If X is the number with the attribute, then X is binomial with $n = 100$ and $p = 40/100 = 0.4$. So $\mathsf{P}(X \leq 35)$ is answered by

```
> pbinom(35, size=100, prob=0.4)
[1] 0.1795
```

5.14 (p. 158) These are found from `pnorm()` and `qnorm()` using the default values for the parameters.

```
> pnorm(2.2)
[1] 0.986
> pnorm(2) - pnorm(-1)
[1] 0.8186
> pnorm(2.5, lower.tail=FALSE)  # or 1 - pnorm(2.5)
[1] 0.00621
> qnorm(.95)
[1] 1.645
```

The latter one uses symmetry of the normal distribution to argue that if $\mathsf{P}(-b < Z \leq b) = 0.90$, then $\mathsf{P}(Z < b) = .95$.

5.15 (p. 158) The probabilities are returned by `pnorm()`:

```
> 1 - pnorm(450, mean=350, sd=75)
[1] 0.09121
```

⋆ **5.16 (p. 158)** We assume that the scores are normally distributed. A student picked at random has a score that is assumed to be normally distributed with mean 20.6 and standard deviation 5.5. We compute $\mathsf{P}(X > 22)$ with

```
> 1 - pnorm(22,20.6,5.5)
[1] 0.3995
```

Thus, about 40% of the students passed the exam.

How many more would be expected to fail if the mark were moved? We multiply the probability times the sample size to get

```
> 1000000 * diff(pnorm(c(22,23), mean=20.6, sd=5.5))
[1] 68251
```

5.17 (p. 159) We use a "p" function to find the probability and a "q" function to return the quantile:

```
> pnorm(26, mean=24.9, sd=1.05)
[1] 0.8526
> qnorm(0.85, mean=24.9, sd=1.05)
[1] 25.99
```

5.18 (p. 159) This can be found using `pnorm()`.

```
> mu = 3.20; sigma = 0.35
> pnorm(4,mu,sigma) - pnorm(3.5,mu,sigma)
[1] 0.1845
```

⋆ **5.19 (p. 159)** The area under the standard normal density between -6 and 6 is characterized by

```
> pnorm(6) - pnorm(-6)
[1] 1
> 1 - (pnorm(6) - pnorm(-6))
[1] 1.973e-09
```

This is about 2 in a billion.

5.20 (p. 159) This is `pnorm(10.7,mean=12,sd=0.5)`, or 0.004661.

⋆ **5.21 (p. 159)** Making use of the `scale()` function we have

```
> x = scale(father.son$fheight)
> sum(abs(x) < 1)/length(x)
[1] 0.6753
> sum(abs(x) < 2)/length(x)
[1] 0.962
> sum(abs(x) < 3)/length(x)
[1] 0.999
```

5.22 (p. 159) The quintiles are found using `qnorm()`.

```
> qnorm((0:5)/5)
[1] -Inf -0.8416 -0.2533 0.2533 0.8416 Inf
```

5.23 (p. 159) For the uniform distribution we can find these probabilities as follows:

```
> mu = 1/2; sigma=sqrt(1/12)
> mu = 1/2; sigma=sqrt(1/12)
> k = 1; diff(punif(mu + c(-1,1)*k*sigma))
```

```
[1] 0.5774
> k = 2; diff(punif(mu + c(-1,1)*k*sigma))
[1] 1
> k = 3; diff(punif(mu + c(-1,1)*k*sigma))
[1] 1
```

For the exponential distribution, the answers are

```
> mu = 5; sigma=5
> k = 1; diff(pexp(mu + c(-1,1)*k*sigma, rate=1/mu))
[1] 0.8647
> k = 2; diff(pexp(mu + c(-1,1)*k*sigma, rate=1/mu))
[1] 0.9502
> k = 3; diff(pexp(mu + c(-1,1)*k*sigma, rate=1/mu))
[1] 0.9817
```

5.24 (p. 159) The q-q plots can be generated as follows:

```
> qqnorm(runif(100))
> qqnorm(rt(100, df=3))
> qqnorm(rnorm(100))
```

The short-tailed uniform should look "S"-like. The long-tailed t distribution (for 3 degrees of freedom) curves down on the left and up on the right. The normal distribution should be roughly a straight line.

5.25 (p. 160) The plot produced by

```
> curve(dnorm(x),-4,4)
> k = 5;   curve(dt(x,df=k), lty=k, add=TRUE)
> k = 10;  curve(dt(x,df=k), lty=k, add=TRUE)
> k = 25;  curve(dt(x,df=k), lty=k, add=TRUE)
> k = 50;  curve(dt(x,df=k), lty=k, add=TRUE)
> k = 100; curve(dt(x,df=k), lty=k, add=TRUE)
```

shows curves steadily approaching the initial one. By the time $k = 100$ it is hard to tell the curves apart.

5.26 (p. 160) The variance is $2k$.

⋆ **5.27 (p. 163)** Using the de Moivre-Laplace theorem for the normal approximation gives

```
> n = 100; p = 1/2; b = 42
> pbinom(b,n,p)
[1] 0.0666
> pnorm(b+1/2, n*p, sqrt(n*p*(1-p)))
[1] 0.06681
```

⋆ **5.28 (p. 163)** The binomial model assumes *i.i.d.* at bats. The answer can be found with the following:

```
> n = 600; p = .300; phat = .350; a = n*phat
> 1 - pnorm(a - 1/2, n*p, sqrt(n*p*(1-p)))
[1] 0.004294
```

5.29 (p. 164) We have $p = 1/2$, $n = 1,000$, and X, the number of people for the issue, is $\mathsf{Binomial}(n, p)$. Then $\mathsf{P}(X \geq 500)$ is approximately

```
> n = 1000; p = 1/2; mu = n*p; sigma=sqrt(n*p*(1-p))
> 1 - pnorm(550 - 1/2, mu, sigma)
```

```
[1] 0.000872
```

⋆ **5.30 (p. 164)** Let $X = X_1 + X_2 + \cdots + X_{15}$ be the total weight of the 15 occupants. The $\mathsf{P}(X > 3500)$ is equivalent to $\mathsf{P}(\bar{X} > 3500/15)$, which by the central limit theorem is

```
> 1 - pnorm(3500/15, mean=180, sd=25/sqrt(15))
[1] 1.110e-16
```

5.31 (p. 164) If the central limit theorem applies, then the average number of bottles sold per night is approximately normally distributed with mean 25 and standard deviation $2/\sqrt{30}$. Call the average $\bar{X}$, and the number of bottles sold in the 30 days X. We have $\bar{X} = X/30$ so that

$$\mathsf{P}(X > 775) = \mathsf{P}(\bar{X} > \frac{775}{30}).$$

This is found with

```
> 1 - pnorm(775/30, mean=25, sd=2/sqrt(30))
[1] 0.01124
```

5.32 (p. 164) Let X be the number of tickets written in 21 days, and $\bar{X} = X/21$ be the daily average. To say the central limit theorem applies means that the distribution of the random variable $\bar{X}$ is approximately normal with mean 4 and standard deviation $1/\sqrt{21}$. Thus, the probability of 75 or fewer is

$$\mathsf{P}(X \leq 75) = \mathsf{P}(\bar{X} \leq \frac{75}{21}).$$

This is given by

```
> pnorm(75/21, mean=4, sd = 1/sqrt(21))
[1] 0.02477
```

Chapter 6

Simulation

6.1 Solutions to Problems

6.1 (p. 177) The simulations can be done and viewed using these commands:

```
> n = 100; p = 0.02
> hist(rbinom(100, size=n, prob=p))
> n = 100; p = 0.2
> hist(rbinom(100, size=n, prob=p))
```

For $p = 0.02$ the distribution is still quite skewed; for $p = 0.2$ this is not the case. This agrees with the rule of thumb that when np and $n(1-p)$ are both 5 or more, the normal distribution can be used to approximate the binomial probabilities.

⋆ **6.2 (p. 178)** A few simulations using steps of size 50 produce an answer.

```
> data(lawsuits)
> res = c()
> n = 5
> for(i in 1:300) res[i] = mean(sample(lawsuits,n,replace=TRUE))
> plot(density(scale(res)), xlim=c(-4,4)) # xlim= sets plot window
> n = 50
> for(i in 1:300) res[i] = mean(sample(lawsuits,n,replace=TRUE))
> lines(density(scale(res)), lty=2)
> n = 150
> for(i in 1:300) res[i] = mean(sample(lawsuits,n,replace=TRUE))
> lines(density(scale(res)), lty=3)
> n = 350
> for(i in 1:300) res[i] = mean(sample(lawsuits,n,replace=TRUE))
> lines(density(scale(res)), lty=5)
> n = 500
> for(i in 1:300) res[i] = mean(sample(lawsuits,n,replace=TRUE))
> lines(density(scale(res)), lty=10)
> qqnorm(res)
```

The use of `xlim=c(-4,4)` with `plot()` gives a symmetric plot window for the subsequent density estimates. We see by $n = 350$ that the shape of the density becomes roughly symmetric. A q-q plot shows the approximate normality of the sample.

⋆ **6.3 (p. 178)** The uniform distribution is short-tailed and symmetric. $\bar{X}$ quickly becomes normally distributed, as can be verified with

```
> res = c(); for(i in 1:500) res[i] = mean(runif(5)); qqnorm(res)
```

⋆ **6.4 (p. 178)** The exponential distribution is skewed. It takes a larger sample than for a non-skewed distribution for the sampling distribution of $\bar{X}$ to become approximately normal. The size is somewhere in the 25 to 50 range. Verify with

```
> res = c();for(i in 1:500) res[i] = mean(rexp(25,1));qqnorm(res)
```

⋆ **6.5 (p. 178)** The t distribution with 3 degrees of freedom has quite a long tail. Because of this, it takes a large sample size for $\bar{X}$ to have an approximately normal sampling distribution. The following simulation should show that $n = 25$ is large enough, as measured by a q-q plot.

```
> res = c();for(i in 1:500) res[i] = mean(rt(25,3));qqnorm(res)
```

When n is much smaller than 25, the tails are still quite long.

6.6 (p. 178) We generate two random samples from the distributions and then compare with boxplots.

```
> res.mean=c()
> res.median=c()
> for(i in 1:200) res.mean[i]   = mean(  rt(10, df=3))
> for(i in 1:200) res.median[i] = median(rt(10, df=3))
> boxplot(list(mean=res.mean,median=res.median))
```

The boxplot shows that for this long-tailed parent distribution, the spread of the sample median value is less than that of the sample mean.

⋆ **6.7 (p. 178)** The central limit theorem says that the sampling distribution of the sample average of a large number of independent, identically distributed random numbers is approximately normal. This indicates that the chi-squared distribution should become bell-shaped for large values of n. The following simulation shows that this happens by $n = 100$ or so.

```
> res = c()
> n = 4;   for(i in 1:500) res[i] = sum(rnorm(n)^2); qqnorm(res)
> n = 10;  for(i in 1:500) res[i] = sum(rnorm(n)^2); qqnorm(res)
> n = 25;  for(i in 1:500) res[i] = sum(rnorm(n)^2); qqnorm(res)
> n = 50;  for(i in 1:500) res[i] = sum(rnorm(n)^2); qqnorm(res)
> n = 100; for(i in 1:500) res[i] = sum(rnorm(n)^2); qqnorm(res)
```

6.8 (p. 179) The simulation with a different distribution is nearly identical:

```
> xbar = c();std = c()
> for(i in 1:500) {
+ sam = rt(10, df=3)
+ xbar[i]=mean(sam); std[i] = sd(sam)
+ }
> plot(xbar,std)
> cor(xbar,std)
[1] 0.09986
```

The correlation might appear to be close to 0, but the scatterplot should convince us that the data is not independent, as it would have a "shotgun"-like pattern were it so.

Using the exponential distribution as the parent population is similar:

```
> for(i in 1:500) {
+ sam = rexp(10)
+ xbar[i]=mean(sam); std[i] = sd(sam)
+ }
```

```
> cor(xbar,std)
[1] 0.7935
> plot(xbar,std)
```

Here, the correlations are not small, and the graph shows the positive correlation—larger values of $\bar{x}$ are associated with large values of s.

6.9 (p. 179) The simulations require one only to change the line

```
> n = 3; m = 200
```

to

```
> n = 25; m = 200
```

Doing so shows that by $n = 50$ or 100 the q-q plot is basically a straight line.

⋆ **6.10 (p. 180)** We should compare the q-q plot with the quantile-normal plot provided by `qqnorm()` to see the difference for small values of n. First, we define a function, `T()`, to compute the statistic from a sample:

```
> T = function(x) mean(x)/(sd(x)/sqrt(length(x)))
```

Then the simulations can be performed as follows (making it easier to modify the lines using the arrow keys):

```
> n=3; m=1000; res=c()
> for(i in 1:m) res[i]=T(rnorm(n))
> qqplot(res, rt(m,df=n-1))
> qqnorm(res)
```

For $n = 3$ the long tail may produce outliers that skew the expected straight line for the q-q plot. This isn't so apparent by $n = 10$, yet the difference between the plot produced with `qqplot()` and that with `qqnorm()` should still be apparent.

6.11 (p. 180) We define the function `T()` to find the T statistic:

```
> T = function(x) mean(x)/(sd(x)/sqrt(length(x)))
```

The simulations can be performed using matrices as follows:

```
> m = 250; n=10
> tmp = matrix(rt(n*m, df = 2), nrow=m)
> res = apply(tmp,1,T)
> hist(res)                     # bell-shaped?
> qqnorm(res)                   # not normal?
> qqplot(res, rt(500, df = n-1)) # better fit than qqnorm()?
```

Replacing the line containing `rt()` with the line

```
> tmp = matrix(rexp(n*m)-1, nrow=m)
```

will use the exponential distribution instead of the t-distribution. The graphs should show that using the symmetric, long-tailed t distribution with 2 degrees of freedom for the population distribution does not greatly change the sampling distribution of T, whereas this is not the case when using the non-symmetric exponential distribution population distribution for $n = 10$.

6.12 (p. 180) Whether a given point is in the circle or not is a Bernoulli random variable with success probability $p = \pi/4$. The number of points inside the circle is then binomial with mean np and standard deviation $\sqrt{np(1-p)}$. The proportion of points inside the circle times 4 should have mean $\pi = 4p$ and standard deviation $4\sqrt{p(1-p)/n}$. For $n = 1000$ and $p = \pi$ this is

```
> n = 1000;p = pi/4; 4*sqrt(p*(1-p)/n)
[1] 0.05193
```

About 95% of all simulations are within 2σ or about 0.01 of π. Some simulations produce:

```
> res = c()
> for(i in 1:250) {n = 1000
```

```
+ x=runif(n,-1,1);y=runif(n,-1,1)
+ res[i]=4*sum(x^2 + y^2<1)/n
+ }
> sum(abs(res-pi)<0.1)/ 250
[1] 0.932
```

Chapter 7

Confidence intervals

7.1 Solutions to Problems

7.2 (p. 188) Clearly, people without phones would be missed. Furthermore, those who use *only* cellular phones would be missed as residential phone books do not list these (yet). Additionally, households with caller ID are unlikely to answer.

7.3 (p. 188) A "scientific" poll uses a random sample from a target population. In this case, people self-select themselves to vote.

7.4 (p. 188) There is no reason to believe that the `http://www.cnn.com` poll would even be accurate when used to predict how widespread an opinion in a nationwide target population.

⋆ **7.5 (p. 189)** The `prop.test` function gives

```
> n = 100; phat = 0.45
> prop.test(n*phat, n, conf.level = 0.8)
...
80 percent confidence interval:
 0.3827 0.5190
...
> prop.test(n*phat, n, conf.level = 0.9)
...
90 percent confidence interval:
 0.3658 0.5370
...
```

7.6 (p. 189) We use `prop.test()`, and its default 95% confidence interval, to answer the question "yes."

```
> prop.test(5,100)
...
95 percent confidence interval:
 0.01855 0.11830
...
```

7.7 (p. 189) The `prop.test()` function with the argument `conf.level=0.80` yields

```
> prop.test(x=9, n=10, conf.level=0.80)
...
80 percent confidence interval:
 0.6577 0.9901
...
```

Compare this to the similar output from `binom.test()`:

```
> binom.test(x=9, n=10, conf.level=0.80)
...
80 percent confidence interval:
 0.6632 0.9895
...
```

⋆ **7.8 (p. 189)** We have a 2 percent margin of error, or

$$.02 = z^* \mathsf{SE}(\widehat{p}).$$

But $z^* = 1.96$ and $\mathsf{SE}(\widehat{p}) = \sqrt{.54 * (1 - .54)}/\sqrt{n}$. Solving gives

```
> phat = .54; zstar = 1.96
> (zstar * sqrt(phat*(1-phat)) / 0.02)^2
[1] 2386
```

7.9 (p. 189) The one-sided widths of a 90% confidence interval are:

```
> n = 5;    SE = sqrt(.8*(.2)/n); qnorm(.95) * SE
[1] 0.2942
> n = 100;  SE = sqrt(.8*(.2)/n); qnorm(.95) * SE
[1] 0.0658
> n = 1000; SE = sqrt(.8*(.2)/n); qnorm(.95) * SE
[1] 0.02081
```

7.10 (p. 189) The margins of error are, respectively,

```
> n = 250;  SE = sqrt(phat*(1-phat)/n);  qnorm(.975) * SE
[1] 0.06167
> n = 1000; SE = sqrt(phat*(1-phat)/n); qnorm(.975) * SE
[1] 0.03083
>  0.06167/0.03083
[1] 2.000
```

That is, quadrupling the sample size halves the margin of error.

⋆ **7.11 (p. 189)** We have $z^* \mathsf{SE}(\widehat{p}) \leq 0.01$. We know z^* as α is given, so we have

$$\sqrt{\frac{\widehat{p}(1-\widehat{p})}{n}} \leq \frac{0.01}{z^*}.$$

Solving for n we get

$$\left(\frac{z^*}{0.01}\right)^2 \widehat{p}(1-\widehat{p}) \leq n.$$

No matter what $\widehat{p}$ is, the expression $\widehat{p}(1-\widehat{p})$ is largest when $\widehat{p} = 1/2$, so we have n is large

enough if

$$\left(\frac{z^*}{0.01}\right)^2 \frac{1}{2}\left(1-\frac{1}{2}\right) \leq n.$$

For our example, we have

```
## for 90% confidence
> alpha = 0.1; zstar = qnorm(1 - alpha/2)
> (zstar/0.01)^2/4
[1] 6764
## for 80% confidence
> alpha = 0.2; zstar = qnorm(1 - alpha/2)
> (zstar/0.01)^2/4
[1] 4106
```

⋆ **7.13 (p. 194)** To compute, we have

```
> Xbar = 9.5; s=1; n=15
> alpha=.1
> tstar = qt(1 - alpha/2, df = n-1)
> SE = s/sqrt(n)
> c(Xbar - tstar*SE, Xbar + tstar*SE)
[1] 9.045 9.955
```

We see that ten days is not in this interval.

7.14 (p. 194) The `t.test()` function will find this for us. First, we should check that the population is approximately normally distributed. A histogram shows this to be the case:

```
> hist(stud.recs$sat.m)
> t.test(stud.recs$sat.m, conf.level=0.80)
...
80 percent confidence interval:
 478.9 493.0
...
```

⋆ **7.15 (p. 195)** These can be found with

```
> t.test(homedata$y1970, conf.level = 0.9)
> t.test(homedata$y2000, conf.level = 0.9)
```

7.16 (p. 195) Once the indices are found, the `t.test()` function can be used to compute the requested confidence interval. A histogram is investigated first to check if the population assumptions are met.

```
> attach(kid.weights)
> ind = age < (5+1)*12 & age >= 5*12
> hist(weight[ind])             # not too skewed or long-tailed
> t.test(weight[ind], conf.level=0.90)
...
90 percent confidence interval:
 42.75 47.09
...
```

⋆ **7.17 (p. 195)** To properly make such a statistical inference, we assume that we can treat the data as a random sample from the population of visible (with the aid of a telescope) stars and then

use the sample to make an estimate for the mean brightness of these stars.
First, we make a histogram, to check that the sample appears to come from a normally distributed population. Once this is verified, then `t.test()` can be used to find the desired confidence interval.

```
> hist(brightness)              # looks good
> t.test(brightness, conf.level = 0.90)
...
90 percent confidence interval:
 8.349 8.486
...
```

For reference: in most suburbs the human eye can see stars with a brightness of 5.5 or less of which about 2,800 exist.

⋆ **7.18 (p. 195)** No, it does not. The mean appears to be 98.2 °F. (See `http://hypertextbook.com/facts/LenaWong.shtml` for some discussion on this.)

```
> t.test(normtemp$temperature,conf.level=.90)

        One Sample t-test

data:  normtemp$temperature
t = 1528, df = 129, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
90 percent confidence interval:
 98.14 98.36
sample estimates:
mean of x
    98.25
```

7.19 (p. 195) Here, $\bar{X} = 67.5$, $s = 2.54$, $n = 3$, and $\alpha = 0.05$. The confidence interval is found with

```
> Xbar = 67.5; s = 2.54; n=4; alpha = 0.05
> tstar = qt(1 - alpha/2,df = n -1 )
> SE = s/sqrt(n)
> c(Xbar -tstar*SE, Xbar + tstar*SE)
[1] 63.46 71.54
```

7.20 (p. 196) The simulations should show that the distribution of the T-statistic is resistant to small changes in the parent population. Only the use of the skewed exponential distribution or the heavy-tailed t-distribution with 3 degrees of freedom produces a noticeably different sampling distribution for T.

7.21 (p. 196) Using the boxplots or the densities, we see that the two distributions seem to be identical for values of $n \geq 25$. However, using the theoretical quantiles, we learn that a value of $n \geq 100$ seems more appropriate for the tail behavior to match. In many introductory statistics books, the value $n = 100$ is often the last tabulated value of n for tables evaluating the tail probabilities of the t-distribution.

⋆ **7.22 (p. 196)** The following will do the simulation:

```
> zs = c(); ts = c()
> for(i in 1:200) {
+ X = rnorm(10,0,2)
+ zs[i] = qnorm(.95)*2/sqrt(10); ts[i] = qt(.95,8)*sd(X)/sqrt(10)
```

```
+ }
> sum(ts > zs)
[1] 126
> sum(ts > zs)/200
[1] 0.63
```

The z one is usually smaller, but not as often as might have been guessed.

7.23 (p. 198) We have

```
> n = 10; m = 20; alpha=0.2
> s = (2.3/2.8)^2
> lr = qf(c(alpha/2, 1- alpha/2),df1=n-1,df2=m-1)
> lr
[1] 0.4338 1.9836
> sqrt(s/rev(lr))
[1] 0.5832 1.2471
```

So the confidence interval is $(0.5832, 1.2471)$.

⋆ **7.24 (p. 199)** The confidence interval can be found with:

```
> attach(nym.2002)
> n = length(place)
> alpha = .10
> (1-alpha)^(1/n)
[1] 0.9999
> max(place)/(1-alpha)^(1/n)
[1] 23664
> max(place)
[1] 23662
> detach(nym.2002)
```

So it is 90% certain that θ is between $23,662$ and $23,664$. (The real answer is 23,664.)

7.25 (p. 206) If we assume that the values are independent and that each group is a random sample from a normally distributed population, then we can make the statistical inference. With these assumptions, the confidence interval is found as follows:

```
> t.test(c1,c2, conf.level=0.80)
...
t = 0.2159, df = 10.79, p-value = 0.833
alternative hypothesis: true difference in means is not equal to 0
80 percent confidence interval:
 -0.5322  0.7322
...
```

By default, the population variances are not assumed to be equal. If this is assumed to be the case, then the values change slightly:

```
> t.test(c1,c2, conf.level=0.80, var.equal=TRUE)
...
t = 0.2159, df = 12, p-value = 0.8327
alternative hypothesis: true difference in means is not equal to 0
80 percent confidence interval:
 -0.5281  0.7281
...
```

7.26 (p. 206) To make the statistical inference, we assume that each group has values that can be viewed as a random sample from a normally distributed parent population with unknown mean

and variance, and that the samples themselves are independent. We make boxplots to check the distributional assumptions:

```
> gp.400 = c(7,0,8,1,10,12,2,9,5,2)
> gp.1200 = c(2,1,5,1,4,7,-1,8,7,3)
> boxplot(list(gp.400=gp.400, gp.1200 = gp.1200))
```

The assumptions on the distributions seem appropriate. Furthermore, the spreads seem similar, so we proceed with the assumption that the population variances are equal.

```
> t.test(gp.400, gp.1200, conf.level = 0.90, var.equal=TRUE)

        Two Sample t-test

data:  gp.400 and gp.1200
t = 1.162, df = 18, p-value = 0.2603
alternative hypothesis: true difference in means is not equal to 0
90 percent confidence interval:
 -0.9347  4.7347
...
```

The confidence interval includes 0, indicating it is plausible that there is no difference in the population means.

⋆ **7.27 (p. 206)** This data is paired off, as it is reasonable to assume that there is a correlation between the IQ scores of twins. We assume that differences in the scores can be viewed as a random sample from a normally distributed population. Then a confidence interval is found as follows:

```
> foster = c(80, 88,75, 113, 95, 82, 97, 94, 132, 108)
> biological = c(90, 91, 79, 97, 97, 82, 87, 94, 131, 115)
> plot(foster, biological)
> boxplot(foster - biological)
> t.test(foster, biological, conf.level=0.90, paired=TRUE)

        Paired t-test

data:  foster and biological
t = 0.041, df = 9, p-value = 0.9682
alternative hypothesis: true difference in means is not equal to 0
90 percent confidence interval:
 -4.369  4.569
```

The exploratory plots show the correlation between the two sets of scores and the distribution of the differences, indicating that a paired *t*-test is appropriate.

7.28 (p. 206) We would reasonably expect that the two parents are of a similar age. This leads us to use a paired *t*-test after checking the validity of the assumption that the differences in age are randomly sampled from a normally distributed population.

```
> plot(age ~ dage, data=babies)
> b = subset(babies, subset= dage < 99 & age < 99 ) # fix 99 = NA
> plot(age ~ dage, data=b)
> hist(b$dage - b$age)
> t.test(b$dage,b$age, conf.level=0.95, paired=TRUE)

        Paired t-test

data:  b$dage and b$age
```

```
t = 28.09, df = 1226, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 2.909 3.346
...
```

⋆ **7.29 (p. 211)** The data set implies that the population distribution is skewed and not normally distributed or symmetric. Thus `t.test()` and `wilcox.test()` are not appropriate. However, after a log transform, it appears that `wilcox.test()` will be.

```
> wilcox.test(log(commutes), conf.int=TRUE, conf.level=0.9)
...
 3.135 3.258
...
> exp(c(3.135,3.258))           # transform back
[1] 22.99 26.00
```

Or, that $[23, 26]$ is the confidence interval. The warning message is about the ties in the data. Notice that the sampling statistic is based on an assumption of a continuous distribution.

7.30 (p. 211) We first look at the distribution of values:

```
> stem(cabinet$est.tax.savings)

  The decimal point is 5 digit(s) to the right of the |

  0 | 0000000011112379
  1 | 12
  2 |
  3 | 3
```

The data is a bit skewed, though, the log-transformed data is not. We can find a confidence interval two ways. First by, using `wilcox.test()`:

```
> x = cabinet$est.tax.savings
> wilcox.test(log(x), conf.level=0.95, conf.int=TRUE)

        Wilcoxon signed rank test

data:  log(x)
V = 190, p-value = 3.815e-06
alternative hypothesis: true mu is not equal to 0
95 percent confidence interval:
  7.025 10.027
sample estimates:
(pseudo)median
         8.544

> exp(c(7.025, 10.027))
[1]  1124 22629
```

Using the binomial distribution we find the larger confidence interval:

```
> n = length(x)
> j = qbinom(0.025, n, 1/2)
> sort(x)[c(j,n+1-j)]
[1]   330 68370
```

⋆ **7.31 (p. 211)** We might envision the collection of songs on an album as coming from a population

of all possible songs the band could have written at the given time. A difference of means would indicate a tendency to write longer songs at one of the given times.
The only trick to finding the confidence interval is referencing the times from *The Joshua Tree*. This needs to be quoted.

```
> wilcox.test(u2$October,u2$"The Joshua Tree", conf.int=TRUE)
```

The calculated interval is $[-94, 10]$.

⋆ **7.32 (p. 212)** The AGE variable is symmetric, so no transformation is needed.

```
> hist(cfb$AGE)                 # symmetric
> wilcox.test(cfb$AGE, conf.int=TRUE)
...
95 percent confidence interval:
 48 50
...
```

The INCOME variable is skewed. We do a log-transform first.

```
> hist(log(cfb$INCOME +1))
> wilcox.test((log(cfb$INCOME +1)), conf.int=TRUE)
...
95 percent confidence interval:
 10.49 10.61
...
> exp(c(10.49,10.61)) - 1
[1] 35953 40537
```

7.33 (p. 212) The distribution is the same regardless of the parent population. In addition to a histogram with an added theoretical density, a q-q plot is also effective at showing this. For example:

```
> qqplot(res,rsignrank(100,n))
```

Chapter 8

Significance tests

8.1 Solutions to Problems

⋆ **8.1 (p. 221)** The FDA must assume that the drug is safe in the null hypothesis. The alternative would be that it is not safe.

⋆ **8.2 (p. 221)** We tabulate to get the counts.

```
> table(samhda$marijuana)
marijuana
  1   2
309 281
```

The hypothesis test is between

$$H_0 : p = 0.5, \quad H_A : p > 0.5.$$

The p-value is given by

```
> x = 309; n = 309 + 281
> prop.test(x,n,p=.5,alt="greater")

        1-sample proportions test with continuity correction

data:  x out of n, null probability 0.5
X-squared = 1.236, df = 1, p-value = 0.1332
alternative hypothesis: true p is greater than 0.5
...
```

which is a small, but not significant p-value.

8.3 (p. 221) What is being asked is to test the hypotheses

$$H_0 : p = .75, \quad H_A : p > .75$$

The test of proportion is done on the data where $\widehat{p} = .8$.

```
> prop.test(40,50, p = .75, alt="greater")$p.value
[1] 0.2568
```

which is not significant.

8.4 (p. 221) A test of significance is done on the hypotheses

$$H_0 : p = .344, \quad H_A : p \neq .344.$$

This is carried out with

```
> prop.test(40,75, p = .344, alt="two.sided")$p.value
[1] 0.0008681
```

This p-value is significant.

⋆ **8.5 (p. 222)** The normal approximation is said to be valid provided np and $n(1-p)$ are 5 or more. In this case $p = .999$ under H_0 and $n = 5,760$, leaving $n(1-p)$ just larger than 5. Assuming the approximation produces accurate p-values, we have the one-sided test, yielding

```
> prop.test(5731, 5760, p=.999, alt="less")

        1-sample proportions test with continuity correction

data:  5731 out of 5760, null probability 0.999
X-squared = 89.87, df = 1, p-value < 2.2e-16
alternative hypothesis: true p is less than 0.999
...
```

This is a very small p-value, indicating that the data is inconsistent with the null hypothesis.

⋆ **8.6 (p. 222)** The key to solving this is to find the value of z so that $P(\widehat{p} > z) = 0.05$. This is answered using the quantiles of the normal distribution, as $\widehat{p}$ is approximately normally distributed with mean $p = 0.113$ and standard deviation $\sqrt{p(1-p)/n}$.

```
> p = .113; n = 50000; qnorm(.95, mean=p,sd=sqrt(p*(1-p)/n))
[1] 0.1153
```

Multiplying by n produces the cutoffs

```
> n*0.1153
[1] 5765
```

Any value of 5,765 or larger would give a p-value less than 0.05.

8.7 (p. 222) This is found with

```
> prop.test(x=2700, n = 25000, p=.1, alt="greater" )

        1-sample proportions test with continuity correction

data:  2700 out of 25000, null probability 0.1
X-squared = 17.69, df = 1, p-value = 1.301e-05
alternative hypothesis: true p is greater than 0.1
95 percent confidence interval:
 0.1048 1.0000
sample estimates:
    p
0.108
```

8.8 (p. 222) Assuming naively that the European poll was a random sample from the entire population of Europeans, we perform this significance test as follows:

```
> prop.test(x=.16*200, n=200, alt="two.sided")

        1-sample proportions test with continuity correction
```

```
data:  0.16 * 200 out of 200, null probability 0.5
X-squared = 91.12, df = 1, p-value < 2.2e-16
alternative hypothesis: true p is not equal to 0.5
```

⋆ **8.9 (p. 225)** The test must be done by hand, as the data is summarized.

```
> xbar = 58260; n = 25; sd = 3250
> mu = 55000; SE = sd/sqrt(n)
> T = (xbar - mu)/SE
> T
[1] 5.015
> pt(T, df=n-1, lower.tail=FALSE)
[1] 1.999e-05
```

The significance test has a small *p*-value.

8.10 (p. 226) As n is large, we can assume that the sampling distribution of T is the normal distribution by the central limit theorem. With this observation, we can find the *p*-value as follows:

```
> xbar = 2.03; s = 0.22; n=800
> mu = 2; SE = s/sqrt(n); T = (xbar - mu)/SE
> pnorm(T, lower.tail=FALSE)
[1] 5.741e-05
```

8.11 (p. 226) The histogram shows that the data is not sampled from a long-tailed distribution, so the *t*-test is appropriate. The `t.test()` function returns a *p*-value less than 0.05, so we would "reject" the null hypothesis at this level.

```
> hist(stud.recs$sat.m)         # n is large, data is short-tailed
> t.test(stud.recs$sat.m, mu=500, alt="two.sided")

        One Sample t-test

data:  stud.recs$sat.m
t = -2.573, df = 159, p-value = 0.01099
alternative hypothesis: true mean is not equal to 500
...
```

⋆ **8.12 (p. 226)** First check normality, then use `t.test()` as follows:

```
> x =  babies$dht
> x = x[x<99]
> t.test(x, mu=68, alt="greater")

        One Sample t-test

data:  x
t = 20.80, df = 743, p-value < 2.2e-16
alternative hypothesis: true mean is greater than 68
...
```

8.13 (p. 226) As the data is presented in summarized form, we must use the relevant formulas to compute the *p*-value.

```
> xbar = 0.5; s = 3.77; n = 7; mu = 0
> SE = s/sqrt(n); T = (xbar - mu)/SE
```

```
> 2*pt(T, df = n-1, lower.tail=FALSE)
[1] 0.7377
```

The computed *p*-value indicates that the difference is not statistically significant.

8.14 (p. 226) We first investigate the data:

```
> hist(OBP)
> length(OBP)
[1] 438
```

Although the data has an outlier, n is quite large, so the *t*-test applies. This yields

```
> t.test(OBP, mu = 0.330, alt="two.sided")

        One Sample t-test

data:  OBP
t = -0.1685, df = 437, p-value = 0.8663
alternative hypothesis: true mean is not equal to 0.33
...
```

8.15 (p. 226) The *t*-test applies, as a histogram shows that the data appears to come from a normal distribution and n is large. The computed *p*-value is

```
> hist(normtemp$temperature)
> t.test(normtemp$temperature, mu = 98.6, alt="two.sided")

        One Sample t-test

data:  normtemp$temperature
t = -5.455, df = 129, p-value = 2.411e-07
alternative hypothesis: true mean is not equal to 98.6
...
```

The *p*-value is quite small. Redoing with the null hypothesis that $\mu = 98.2$ yields

```
> t.test(normtemp$temperature, mu = 98.2, alt="two.sided")

        One Sample t-test

data:  normtemp$temperature
t = 0.7656, df = 129, p-value = 0.4453
alternative hypothesis: true mean is not equal to 98.2
...
```

⋆ **8.16 (p. 227)** We do this all at once using a list for storage

```
> lst = list()
> m = 250; n = 10
> for(i in 1:m) {
+ lst$exp[i]  = t.test(rexp(n),     mu=1,  df=n-1)$p.value
+ lst$unif[i] = t.test(runif(n),    mu=.5, df=n-1)$p.value
+ lst$t4[i]   = t.test(rt(n, df=4), mu=0,  df=n-1)$p.value
+ }
> sapply(lst, function(x) sum(x<0.05)/length(x))
  exp  unif    t4
0.088 0.052 0.044
```

When the population distribution is skewed, the sampling distribution of T may differ a lot from the *t*-distribution.

⋆ **8.17 (p. 231)** The sign test of the hypotheses

$$H_0 : \text{median} = 220,000, \quad H_A : \text{median} > 220,000$$

is done with test statistic T, the number of data points more than 22. Larger values of T support the alternative. The p-value is then calculated as

```
> T = sum(exec.pay > 22)
> n = length(exec.pay)
> T
[1] 113
> pbinom(T - 1, n, .5, lower.tail=FALSE)
[1] 0.03252
```

⋆ **8.18 (p. 232)** The `log()` function can be used directly, as in

```
> wilcox.test(log(exec.pay), mu = log(22), alt="greater")

        Wilcoxon signed rank test with continuity correction

data:  log(exec.pay)
V = 10966, p-value = 0.004144
alternative hypothesis: true mu is greater than 3.091
```

We see a similarly small p-value as in the previous question, indicating that the null hypothesis is not supported.

If you tried to do a histogram to check for symmetry, you may have gotten an error message. This is because of the data values for which the compensation is 0. (The logarithm of 0 is treated as $-\infty$.) To avoid this error, you can exclude this case first as follows:

```
> ep = exec.pay[exec.pay > 0]
> hist(log(ep))                 # fairly symmetric
```

8.19 (p. 232) A boxplot of `smokers` reveals a population that is symmetric with longer tails than the normal distribution. The rank-sum test would be appropriate, whereas the t-test would not necessarily be. The p-value is found with

```
> wilcox.test(smokers, mu = 40*7)

        Wilcoxon signed rank test with continuity correction

data:  smokers
V = 44186, p-value = 0.003946
alternative hypothesis: true mu is not equal to 280
```

8.20 (p. 232) One run of the simulation shows the t-test to be more effective at rejecting when the null hypothesis is false.

```
> sum(res.t)/m                  # proportion rejected by t-test
[1] 0.405
> sum(res.sign)/m               # proportion rejected by sign-test
[1] 0.115
```

⋆ **8.21 (p. 235)** Assuming the phones are a random sample from the population of all phones, this can be done with `prop.test()`:

```
> prop.test(c(14,15),c(150,125),alt="less")
```

```
        2-sample test for equality of proportions with
        continuity correction

data:  c(14, 15) out of c(150, 125)
X-squared = 0.2702, df = 1, p-value = 0.3016
alternative hypothesis: less
...
```

The answer is "no."

8.22 (p. 235) This is a two-sided test of proportions. Using the default for `prop.test()` gives

```
> prop.test(c(250,250),c(600,500))$p.value
[1] 0.00687
```

8.23 (p. 235) The standard assumptions that the samples be random samples from the target population is suspicious in this instance, as we can reasonably assume that not everyone in the listening audience would be interested in attending an advance screening. Even if they were, the audience is not necessarily a random sample from the listening audience. Nonetheless, if we use `prop.test()` it yields a small *p*-value:

```
> prop.test(350*c(.82,.70), c(350,350))

        2-sample test for equality of proportions with
        continuity correction

data:  350 * c(0.82, 0.7) out of c(350, 350)
X-squared = 13.17, df = 1, p-value = 0.0002851
alternative hypothesis: two.sided
...
```

8.24 (p. 235) We compare with `prop.test()`:

```
> prop.test(c(153,196),c(30000,30000),alt="two.sided")

        2-sample test for equality of proportions with
        continuity correction

data:  c(153, 196) out of c(30000, 30000)
X-squared = 5.084, df = 1, p-value = 0.02415
alternative hypothesis: two.sided
...
```

The small *p*-value says that the differences are statistically significant.

8.25 (p. 236) The *p*-value computed from these numbers is found from `prop.test()` as follows:

```
> prop.test(c(18,3),c(22,22))

        2-sample test for equality of proportions with
        continuity correction

data:  c(18, 3) out of c(22, 22)
X-squared = 17.86, df = 1, p-value = 2.384e-05
alternative hypothesis: two.sided
95 percent confidence interval:
 0.4206 0.9430
```

```
sample estimates:
prop 1 prop 2
0.8182 0.1364
```

8.26 (p. 236) An assumption of `prop.test()` is that the normal distribution approximates the binomial values. We check using the rule of thumb that np and $n(1-p)$ are larger than 5, assuming p is no more than .99.

```
> 1250*(1-.99)
[1] 12.5
> percents = c(98.9,96.9); n = c(1250, 1100)
> prop.test(n*percents/100, n)

        2-sample test for equality of proportions with
        continuity correction

data:  n * percents/100 out of n
X-squared = 10.75, df = 1, p-value = 0.001042
alternative hypothesis: two.sided
95 percent confidence interval:
 0.007384 0.032616
sample estimates:
prop 1 prop 2
 0.989  0.969
```

⋆ **8.27 (p. 237)** We have $n_1 = n_2 = 10,000$ and $\widehat{p}_1 = .216$, $\widehat{p}_2 = .193$. A two-sided significance test is performed with `prop.test()` as follows:

```
> n = c(10000,10000)
> phat = c(.216,.193)
> x = n*phat
> prop.test(x,n,conf.level = 0.01)

        2-sample test for equality of proportions with
        continuity correction

data:  x out of n
X-squared = 16.12, df = 1, p-value = 5.952e-05
alternative hypothesis: two.sided
1 percent confidence interval:
 0.02283 0.02317
sample estimates:
prop 1 prop 2
 0.216  0.193
```

The difference is statistically significant. However, if the surveys were only of size 1,000m the difference would not have been statistically significant.

⋆ **8.28 (p. 246)** The assumptions are such that the t-statistic can be used to compute the small p-value. We assume that the population variances are equal.

```
> xbar1 =79; xbar2 = 110
> n1 = n2 = 250
> s1 = 25; s2 = 20
> sp = sqrt(( (n1-1)*s1^2 + (n2-1)*s2^2 )/(n1+n2-2))
> SE = sp * sqrt(1/n1 + 1/n2)
```

```
> T = (xbar1 - xbar2)/SE
> 2 * pt(T, df = n1+n2-2)       # T is negative, two sided
[1] 1.224e-43
```

8.29 (p. 246) We assume the two groups can be treated as random samples from the population of recovery times for children with colds who take the given treatment. Then, as n is large, we can compute the p-value using the two-sample t-test with the assumption of equal variances. The p-value is not significant.

```
> xbar1 = 5.3; xbar2 = 5.4
> sp = 2.5                      # clearly as pooled value is average
> n1 = 200; n2 = 207
> SE = sp*sqrt(1/n1 + 1/n2)
> T = (xbar1 - xbar2)/SE
> 2*pt(T, df = n1 + n2 - 2)     # T is negative
[1] 0.6868
```

8.30 (p. 247) The age of one spouse is not independent of the age of the other spouse. A scatterplot shows this and reminds us that values that should be `NA` are coded using `99`:

```
> plot(age ~ dage, data=babies)
> b = subset(babies, subset= dage < 99 & age < 99 )
> plot(age ~ dage, data=b)
```

A two-sample test would be inappropriate for this data. Instead, we assume that the age differences are a random sample from the population of age differences and apply the paired t-test. As n is large, we don't worry about the population distribution being normal.

```
> with(b, t.test(dage - age, alt="greater"))

        One Sample t-test

data:  dage - age
t = 28.09, df = 1226, p-value < 2.2e-16
alternative hypothesis: true mean is greater than 0
...
mean of x
    3.127
```

8.31 (p. 247) Splitting the data up and then performing the t-test can be done as follows:

```
> attach(normtemp)
> males = temperature[gender == 1]
> females = temperature[gender == 2]
> boxplot(list(males=males, females=females))
> t.test(males, females, alt="two.sided")

        Welch Two Sample t-test

data:  males and females
t = -2.285, df = 127.5, p-value = 0.02394
alternative hypothesis: true difference in means is not equal to 0
...
> detach(normtemp)
```

The small p-value indicates a statistically significant difference.

8.32 (p. 247) The data is paired off, as each student takes two exams. Assuming that the *t*-test applies to the differences in test scores, a *p*-value can be computed as follows.

```
> pre = c(17,12,20,12,20,21,23,10,15,17,18,18)
> post = c(19,25,18,18,26,19,27,14,20,22,16,18)
> t.test(post - pre, mu = 0, alt="greater")

        One Sample t-test

data:  post - pre
t = 2.563, df = 11, p-value = 0.01319
alternative hypothesis: true mean is greater than 0
95 percent confidence interval:
 0.9727    Inf
sample estimates:
mean of x
     3.25
```

8.33 (p. 247) The data is clearly paired off. Assuming the differences are a random sample from a normally distributed population, the *p*-value can be found as follows:

```
> method1 = c(45.9,57.6,54.9,38.7,35.7,39.2,45.9,43.2,45.4,54.8)
> method2 = c(48.2,64.2,56.8,47.2,43.7,45.7,53.0,52.0,45.1,57.5)
> t.test(method1 - method2, mu = 0, alt="two.sided")

        One Sample t-test

data:  method1 - method2
t = -5.078, df = 9, p-value = 0.0006649
alternative hypothesis: true mean is not equal to 0
...
```

The small *p*-value indicates that the data is not consistent with the null hypothesis that the differences have mean 0.

8.34 (p. 247) The scatterplot shows that *A* and *B* are related, as we would guess after realizing that the difference in wear between shoes per boy should quantify the materials difference. Thus a paired *t*-test is appropriate.

```
> library(MASS)                 # load data set
> names(shoes)
> plot(B ~ A, data=shoes)       # related
[1] "A" "B"
> with(shoes, t.test(A,B,paired=TRUE)$p.value)
[1] 0.008539
> with(shoes, t.test(A,B)$p.value)
[1] 0.7165
```

Not realizing the paired nature gives a completely different *p*-value.

⋆ **8.35 (p. 248)** The data is not independent among the samples as there are only 205 parents and 930 children represented. Treating it as though the pairs are independently drawn, we can use a paired *t*-test to get

```
> attach(galton)
> t.test(child,parent,paired=TRUE)
```

```
        Paired t-test

data:  child and parent
t = -2.966, df = 929, p-value = 0.003089
alternative hypothesis: true difference in means is not equal to 0
...
> detach(galton)
```

The small *p*-value indicates a difference in the means. The confidence interval suggests that it is between 0.07 inches and 0.37 inches.

8.36 (p. 248) The assumptions of normality were addressed in the example this problem refers to, so we can apply the `var.test()` function:

```
> x = c(284, 279, 289, 292, 287, 295, 285, 279, 306, 298)
> y = c(298, 307, 297, 279, 291, 335, 299, 300, 306, 291)
> var.test(x,y)

        F test to compare two variances

data:  x and y
F = 0.3418, num df = 9, denom df = 9, p-value = 0.1256
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.0849 1.3762
sample estimates:
ratio of variances
            0.3418
```

The reported *p*-value of 0.1256 shows no reason to doubt the null hypothesis of equal variances.

Chapter 9

Goodness of fit

9.1 Solutions to Problems

9.1 (p. 254) We can't answer the question about whether the die is fair, only whether the results are consistent with that assumption. With that in mind, we perform the chi-squared goodness-of-fit test. We use the default null hypothesis that all outcomes are equally likely.

```
> x = c(13, 17, 9, 17, 18, 26)
> chisq.test(x)

        Chi-squared test for given probabilities

data:  x
X-squared = 9.68, df = 5, p-value = 0.08483
```

The p-value is small, but it is not below the common 0.05 threshold for statistical significance.

9.2 (p. 255) We test the following null hypothesis:

$$H_0 : p_1 = .486, p_2 = .315, p_3 = .125, p_4 = .028, p_5 = .006, \text{ and } p_6 = .04$$

against the alternative hypothesis that these probabilities are not correct. We use `chisq.test()` as follows:

```
> obs = c(315, 197, 141, 39, 16, 79)
> p = c(.486, .315, .125, .028, .006, .040)
> chisq.test(obs, p=p)

        Chi-squared test for given probabilities

data:  obs
X-squared = 152.6, df = 5, p-value < 2.2e-16

Warning message:
Chi-squared approximation may be incorrect in: chisq.test(obs, p = p)
```

The warning is due to the expected cell counts for Huffington being less than 5 (they are 4.72). The small p-value indicates that it is likely that the sample is not consistent with the actual vote. It seems that voter perceptions changed between the time of the poll and the election.

⋆ **9.3 (p. 255)** The data set gives percentages, but `chisq.test()` wants proportions. We first define a quick function to compute proportions, then apply the two tests.

```
> prop = function(x) x/sum(x)
> bagfull = c(15,34,7,19,29,24)
> names(bagfull) = c("blue","brown","green","orange","red","yellow")
> chisq.test(bagfull,p = prop(mandms["milk chocolate",]))

        Chi-squared test for given probabilities

data:  bagfull
X-squared = 7.065, df = 5, p-value = 0.2158

> chisq.test(bagfull,p = prop(mandms["Peanut",]))

        Chi-squared test for given probabilities

data:  bagfull
X-squared = 13.33, df = 5, p-value = 0.02049
```

It appears that the bag is milk chocolate not peanut.

9.4 (p. 255) The null hypothesis for this problem is the assumption of equally likely digits. This is the default for `chisq.test()`. The *p*-value can be found after tabulating the data as follows:

```
> chisq.test(table(pi2000))

        Chi-squared test for given probabilities

data:  table(pi2000)
X-squared = 4.42, df = 9, p-value = 0.8817
```

The data is consistent with the null hypothesis of random digits.

9.5 (p. 256) We enter in the data and then use `chisq.test()`. We use probabilities with `chisq.test()`, so we need to turn our frequencies into proportions.

```
> counts = c(28,39,23,22,11)
> freq = c(9,12,9,8,4)
> chisq.test(counts,p=freq/sum(freq))$p.value
[1] 0.8968
```

The *p*-value gives no reason to doubt the null hypothesis that the frequencies are from the English language.

9.6 (p. 256) Once the letter distribution and assumed letter frequency are found, the chi-squared test can be performed with `chisq.test()` as follows:

```
> all.names  = paste(bright.stars$name, sep="", collapse="")
> x = unlist(strsplit(tolower(all.names), ""))
> letter.dist = sapply(letters, function(i) sum(x == i))
> p = scrabble$frequency[1:26];p=p/sum(p)
> chisq.test(letter.dist, p=p)

        Chi-squared test for given probabilities

data:  letter.dist
X-squared = 260.7, df = 25, p-value < 2.2e-16
```

The tiny p-value shows that the chi-squared test accurately detects that the letter distribution is not consistent with those implied by the Scrabble frequencies.

⋆ **9.7 (p. 257)** The first is done quite quickly using the defaults, as the uniform distribution is the null hypothesis.

```
> murder = c(53,42,51,45,36,36,65)
> chisq.test(murder)

        Chi-squared test for given probabilities

data:  murder
X-squared = 13.80, df = 6, p-value = 0.03189
```

The second we must do by hand. First note that if we specify p_w, the weekend probability, then p_d, the weekday probability, is $(1-2p_w)/5$. There is only 1 degree of freedom. We estimate p_w with the average of the weekend counts:

```
> n = sum(murder)
> phatw = (53 + 65)/(2*n)
> phatd = (1 - 2*phatw)/5
> e = n * c(phatw, rep(phatd,5), phatw)
> cs = sum ( (murder - e)^2/e )
> cs
[1] 5.077
> 1 - pchisq(cs,df=1)
[1] 0.02424
```

In both cases the p-value is small.

9.8 (p. 257) We have $H_0 : p_b = p_o$. Call this p. We estimate this with $(106+87)/2n$ and the others with the sample proportions. There are two degrees of freedom, as specifying two of the probabilities yields the third and fourth by the null hypothesis. Thus we can get the p-value as follows:

```
> colors = c(41,48,105,58)
> n = sum(colors)
> phat = mean(41/n,48/n)
> yellowhat = 105/n; greenhat = 58/n
> exp = n*c(phat,phat,yellowhat,greenhat)
> cs = sum( (colors - exp)^2/exp )
> cs
[1] 1.195
> pchisq(cs,df=2,lower.tail=FALSE)
[1] 0.5502
```

Despite the apparent difference, it is not statistically significant.

9.9 (p. 257) The histogram and density curves usually show a poor fit in the two instances shown. A formal significance test using `ks.test()`, as in

```
> ks.test(res, "pchisq", df=k-1)
```

can be used for further investigation.

9.10 (p. 257) Start with the formula for the chi-squared statistic, rewrite in terms of sample propor-

tions, then use the facts that $\widehat{p}_1 + \widehat{p}_2 = 1$ and $p_1 + p_2 = 1$:

$$\sum_{i=1}^{n} \frac{(e_i - f_i)^2}{e_i} = \frac{(np_1 - n\widehat{p}_1)^2}{np_1} + \frac{(np_2 - n\widehat{p}_2)^2}{np_2}$$

$$= n\left(\frac{(1-p_1)(p_1 - \widehat{p}_1)^2 + p_1(1 - p_1 - (1 - \widehat{p}_1))^2}{p_1(1-p_1)}\right)$$

$$= \frac{(p_1 - \widehat{p}_1)^2}{p_1(1-p_1)/n}.$$

⋆ **9.11 (p. 264)** The accident data can be entered in using `cbind` as follows:

```
> accidents = cbind(
+ none=c(67,42,75,56,57),
+ minor=c(10,6,8,4,15),
+ major=c(5,5,4,6,1))
> rownames(accidents)=c("<18","18-25","26-40","40-65","65>")
> accidents
      none minor major
<18     67    10     5
18-25   42     6     5
26-40   75     8     4
40-65   56     4     6
65>     57    15     1
> chisq.test(accidents)

        Pearson's Chi-squared test

data:  accidents
X-squared = 12.59, df = 8, p-value = 0.1269

Warning message:
Chi-squared approximation may be incorrect in: chisq.test(accidents)
```

9.12 (p. 264) We can investigate whether the data is consistent with the hypothesis that the two variables are independent with a chi-squared test of significance. For this data we get a very small p-value:

```
> sb.yes = c(12813, 647, 359, 42)
> sb.no = c(65963, 4000, 2642, 303)
> chisq.test(rbind(sb.yes,sb.no))

        Pearson's Chi-squared test

data:  rbind(sb.yes, sb.no)
X-squared = 59.22, df = 3, p-value = 8.61e-13
```

⋆ **9.13 (p. 265)** After massaging the data, a glimpse at a contingency table shows us that there appears to be some dependency between the variables. This is borne out by the tiny p-value returned by `chisq.test()`.

```
> aq = airquality[complete.cases(airquality),]
```

```
> attach(aq)
> te = cut(Temp, quantile(Temp))
> oz = cut(Ozone,quantile(Ozone))
> table(te,oz)
           oz
te          (1,18] (18,31] (31,62] (62,168]
  (57,71]   15      9       3       0
  (71,79]   10     10       7       1
  (79,84.5]  4      6      11       5
  (84.5,97]  0      0       6      22
> chisq.test(te,oz)

        Pearson's Chi-squared test

data:  te and oz
X-squared = 76.31, df = 9, p-value = 8.713e-13
```

9.14 (p. 265) The retention data can be entered using `rbind()`.

```
> retention = rbind(
+ nonblock=c(18,15,5,8,4),
+ block = c(10,5,7,18,10))
> colnames(retention) = c(1:4,"5+")
> retention
          1  2 3  4 5+
nonblock 18 15 5  8  4
block    10  5 7 18 10
> chisq.test(retention)

        Pearson's Chi-squared test

data:  retention
X-squared = 14.04, df = 4, p-value = 0.007179
```

The small p-value does not support the assumption of similarly distributed variables.

9.15 (p. 265) A peek at the data shows it to be sparse. It is best to heed the warning that the approximation might be incorrect in this case. The p-value found by simulation has a completely different interpretation than that found using the chi-squared approximation.

```
> oral.lesion
               Kerala Gujarat Andhra
Labial.Mucosa       0       1      0
Buccal.Mucosa       8       1      8
Commissure          0       1      0
Gingiva             0       1      0
Hard.Palate         0       1      0
Soft.Palate         0       1      0
Tongue              0       1      0
Floor.Mouth         1       0      1
Alveolar.Ridge      1       0      1
> chisq.test(oral.lesion)

        Pearson's Chi-squared test
```

```
data:  oral.lesion
X-squared = 22.1, df = 16, p-value = 0.1400

Warning message:
Chi-squared approximation may be incorrect in: chisq.test(oral.lesion)
> chisq.test(oral.lesion,simulate.p.value=TRUE)

        Pearson's Chi-squared test with simulated p-value (based
        on 2000 replicates)

data:  oral.lesion
X-squared = 22.1, df = NA, p-value = 0.02599
```

9.16 (p. 275) Assume that the samples are random samples from two different populations. The Kolmogorov-Smirnov test can be used to see whether the data is consistent with the assumption that these populations have equivalent distributions.

```
> y2001 = c(110, 12, 2.5, 98, 1017, 540, 4.3, 150, 432)
> y2002 = c(312, 316, 175, 200, 92, 201, 428, 51, 289, 1126, 822)
> ks.test(y2001, y2002)

        Two-sample Kolmogorov-Smirnov test

data:  y2001 and y2002
D = 0.4848, p-value = 0.1396
alternative hypothesis: two.sided
```

⋆ **9.17 (p. 275)** The two tests can be carried out with

```
> shapiro.test(babies$ht[babies$ht < 99])$p.value
[1] 4.898e-10
> shapiro.test(babies$wt[babies$wt < 999])$p.value
[1] 0.001192
```

In each case the null hypothesis would be rejected.

9.18 (p. 275) The histogram with empirical and theoretical densities shows that the fit is not convincing, although we see that the distribution appears to be bell-shaped with similar-sized tails. A better diagnostic is the quantile-normal plot, which in this case shows the non-normal distribution by its slight curve. The formal Shapiro-Wilk test for normality of the parent distribution quantifies this, yielding a tiny *p*-value:

```
> shapiro.test(brightness)

        Shapiro-Wilk normality test

data:  brightness
W = 0.9822, p-value = 1.825e-09
```

⋆ **9.19 (p. 275)** The Shapiro-Wilk test accepts a null hypothesis of normally distributed data.

```
> shapiro.test(normtemp$temperature)

        Shapiro-Wilk normality test

data:  normtemp$temperature
```

```
W = 0.9866, p-value = 0.2332
```

⋆ **9.20 (p. 275)** A plot of both the empirical density and the theoretical density of the gamma distribution with parameters chosen by `fitdistr()` can be produced as follows:

```
> library(MASS)
> fitdistr(rivers,"gamma")
     shape       rate
  2.5783541   0.0043609
 (0.2489438) (0.0004386)
... skip warnings ...
> plot(density(rivers))
> x = 0:4000
> lines(x, dgamma(x,shape=2.578, rate= 0.00436), lty=2)
```

The gamma shape matches the data, but the tail behavior does not seem that similar. A quantile-quantile plot will show this:

```
> qqplot(rivers, rgamma(100,shape=2.578, rate= 0.00436))
```

9.21 (p. 275) This can be done using `fitdistr()` from the `MASS` package. For the math SAT scores we have:

```
> library(MASS)
> fitdistr(stud.recs$sat.m,"normal")
    mean       sd
  485.938    68.914
 (  5.448) (  3.852)
```

The parameter labeled `mean` is the estimate for the parameter μ in the normal distribution, and that labeled `sd` is the estimate for σ.

9.22 (p. 276) We do the simulations with both distributions and check for how often we would reject at the 0.05 level.

```
> k = 1:100
> res.t = sapply(k, function(x) ks.test(rt(25,df=3),"pnorm")$p.value)
> res.exp = sapply(k, function(x) ks.test(rexp(25)-1,"pnorm")$p.value)
> sum(res.t < .05)/100
[1] 0.06
> sum(res.exp < .05)/100
[1] 0.17
```

The exponential distribution was only "rejected" 17 percent of the time, and the t-distribution only 6 percent of the time.

9.24 (p. 276) It is better to run repeated simulations, rather than a single one. To do this, we use the fact that the return value of `shapiro.test()` is a list containing a component labeled `p.value`.

```
> f = function(n,outlier=5) shapiro.test(c(rnorm(n),outlier))$p.value
> res.100  = sapply(1:500, function(x) f(n=100))
> res.1000 = sapply(1:500, function(x) f(n=1000))
> res.4000 = sapply(1:500, function(x) f(n=4000))
> sum(res.100  <= 0.05)/500*100
[1] 99.4
> sum(res.1000 <= 0.05)/500*100
[1] 91.4
> sum(res.4000 <= 0.05)/500*100
```

```
[1] 35.4
```

Compare this to what happens when no outlier is present:

```
> res.100.nooutlier = sapply(1:500,
+ function(x) f(n=100,outlier=rnorm(1)))
> sum(res.100.nooutlier <= 0.05)/500*100
[1] 4.6
```

We conclude that a single outlier can significantly affect the *p*-value computed by the test.

Chapter 10

Linear regression

10.1 Solutions to Problems

⋆ **10.1 (p. 283)** We begin by loading in the data set and looking at the names.

```
> library(MASS)                 # loads data set
```

For the model of highway mileage by horsepower we expect a negative correlation. A scatterplot confirms this.

```
> plot(MPG.highway ~ Horsepower, data = Cars93)
> res = lm(MPG.highway ~ Horsepower, data = Cars93)
> res

Call:
lm(formula = MPG.highway ~ Horsepower, data = Cars93)

Coefficients:
(Intercept)   Horsepower
     38.150       -0.063

> predict(res, newdata=data.frame(Horsepower=225))
[1] 23.97
```

Modeling highway mileage by automobile weight should have a similar negative correlation. Again we confirm and make the requested predictions.

```
> f = MPG.highway ~ Weight
> plot(f, data=Cars93)
> res = lm(f, data=Cars93)
> res

Call:
lm(formula = f, data = Cars93)

Coefficients:
(Intercept)       Weight
   51.60137     -0.00733

> predict(res, newdata=data.frame(Weight=c(2524, 6400)))
     1      2
```

```
33.108  4.708
```

The prediction for the MINI Cooper may be close, but there is no reason to expect the prediction for the HUMMER to be close, as the value of the predictor is outside the range of the data.

The variable `Min.Price` records the value of the stripped-down version of the car, and `Max.Price` records the fully equipped version. We'd expect that `Max.Price` would roughly be a fixed amount more than `Min.Price`, as the differences—the cost of leather seats, a bigger engine, perhaps— are roughly the same for each car. Checking, we have:

```
> f = Max.Price ~ Min.Price
> plot(f, data=Cars93)
> res = lm(f,data=Cars93)
> abline(res)
> res

Call:
lm(formula = f, data = Cars93)

Coefficients:
(Intercept)    Min.Price
       2.31         1.14
```

The slope of 1.14 indicates that perhaps add-ons for more expensive cars cost more, but in this case it appears to be due to the one large outlier, as robust regression estimates are much closer to 1:

```
> rlm(f, data=Cars93)
Call:
rlm(formula = f, data = Cars93)
Converged in 7 iterations

Coefficients:
(Intercept)   Min.Price
      3.609       1.029

Degrees of freedom: 93 total; 91 residual
Scale estimate: 3.18
```

A scatterplot matrix may show additional linear relationships. These are produced with the `pairs()` command, as in `pairs(Cars93)`. Doing so directly produces too many scatterplots. We can trim down the size of the data frame then plot again. Doing so using only the nonfactors can be done as follows:

```
> cars = Cars93[,sapply(Cars93, function(x) !is.factor(x))]
> pairs(cars)
```

Looking at the plots produced we see, for example, that variables 1 and 2, 2 and 3, 4 and 5, etc., are linearly related. These variables can be identified from the graphic if the monitor is large enough, or with the command `names(cars)`.

10.2 (p. 283) The slope, $\widehat{\beta}_1$ gives the increase for a given win; we need to multiply this by 10.

```
> lm(attendance ~ wins,MLBattend)
...
Coefficients:
(Intercept)         wins
    -378164        27345

> 27345*10
```

```
[1] 273450
```

10.3 (p. 283) We enter in the data and then use the regression coefficients directly to make a prediction.

```
> age.two = c(39, 30, 32, 34, 35, 36, 36, 30)
> adult   = c(71, 63, 63, 67, 68, 68, 70, 64)
> lm(adult ~ age.two)
...
Coefficients:
(Intercept)      age.two
     35.179        0.929

> 35.179 + .929*33
[1] 65.84
```

We see that the slope is not close to 2, although this is too small a data set to draw any generalizations from.

10.4 (p. 283) The `jitter()` function has an extra argument to increase the jitter amount. In this case a value of 5 produces a reasonable-looking plot. The regression line has a slope of 0.646. This would be 1 if children were the same height as their parents. The correlation of 0.4588 shows that the linear model explains only a small fraction of the relationship between points.

```
> attach(galton)
> plot(parent, child)
> res = lm(child ~ parent)
> abline(res)
> points(jitter(parent,5), jitter(child,5), pch=3)
> plot(parent, child)
> res = lm(child ~ parent)
> abline(res)
> points(jitter(parent,5), jitter(child,5), pch=3)
> res
...
Coefficients:
(Intercept)       parent
     23.942        0.646
> cor(parent,child)
[1] 0.4588
> detach(galton)                # tidy up
```

10.5 (p. 284) After using `scale()` to center and standardize the data, we see that $\widehat{\beta}_1$ and r have the same value (up to rounding differences).

```
> attach(galton)
> lm(scale(child) ~ scale(parent))
...
Coefficients:
  (Intercept)  scale(parent)
     4.34e-13       4.59e-01
> cor(scale(child), scale(parent))
       [,1]
[1,] 0.4588
> detach(galton)
```

⋆ **10.6 (p. 297)** The p-value is found by computing the t-statistic using $\widehat{\beta}_1$, and $\mathsf{SE}(\widehat{\beta}_1)$ is found from the `summary()` function.

```
> price = c(300, 250, 400, 550, 317, 389, 425, 289, 389, 559)
> no.bed = c(3, 3, 4, 5, 4, 3, 6, 3, 4, 5)
> res = lm(price ~ no.bed)
> summary(res)
...
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)     94.4       98.0    0.96    0.364
no.bed          73.1       23.8    3.08    0.015 *
...
> T = (73.1 - 60)/23.8
> pt(T, df = 8, lower.tail=FALSE)
[1] 0.2985
```

The large p-value is not significant.

10.7 (p. 297) After we fit the model, the significance test can be done using the reported standard error and estimate for $\widehat{\beta}_1$.

```
> no.beers = c(5, 2, 9, 8, 3, 7, 3, 5, 3, 5)
> bal = c(0.10, 0.03, 0.19, 0.12, 0.04, 0.095, 0.07, 0.06, 0.02, 0.05)
> res = lm(bal ~ no.beers)
> summary(res)
...
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.01850    0.01923   -0.96   0.3642
no.beers     0.01920    0.00351    5.47   0.0006 ***
...
> T = (0.01920 - .02)/0.00351
> 2*pt(T, df = length(no.beers)-2)
[1] 0.8254
```

The large p-value is consistent with the null hypothesis being true.

10.8 (p. 297) The marginal t-tests are two-sided significance tests of whether the coefficient is 0. This means that the p-value is contained in the output of `summary()`.

```
> no.beers = c(5, 2, 9, 8, 3, 7, 3, 5, 3, 5)
> bal = c(0.10, 0.03, 0.19, 0.12, 0.04, 0.095, 0.07, 0.06, 0.02, 0.05)
> summary(lm(bal ~ no.beers))
...
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.01850    0.01923   -0.96   0.3642
...
```

The p-value for this data is 0.3642.

⋆ **10.9 (p. 297)** For illustration, we type the data into a text file with two columns, the first being the elevation. Then we use `read.table()` to read it in.

```
> x = read.table(file="tmp.txt")
> names(x) = c("elev","Temp")
> res = lm(Temp ~ elev, x)
```

```
> summary(res)
...
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 59.290662   1.717329   34.52  3.9e-08 ***
elev        -0.005115   0.000921   -5.55   0.0014 **
---
...
> T = (-0.005115 - (-.00534)) / .000921
> T
[1] 0.2443
> 2*pt(T,df=6,lower.tail=FALSE) # two-sided
[1] 0.8151
```

The *p*-value is not statistically significant.

10.10 (p. 298) Despite the common use of exponential models for population growth, a scatterplot reveals a linear-like trend for the years indicated. However, 2004 is well beyond the range of the data, so any predictions based on this data would be suspicious. To find the 1963 predictions we can use the `predict()` function as shown.

```
> years = 1952:1962
> seal.counts = c(724, 756, 920, 1392, 1392, 1448, 1212, 1672,
+ 2068, 1980, 2116)
> plot(seal.counts ~ years)
> res = lm(seal.counts ~ years)
> predict(res, newdata=data.frame(years=1963))
[1] 2281
```

⋆ **10.11 (p. 298)** The predicted value can be found with `predict()`. Though you may not want to believe it, as the model has some issues. The simple plot of the residuals shows values that are scattered around 0, with nothing unusual. However, the second plot using the 1970 values on the *x*-axis instead of the index, shows that the variance increases with the price of the house.

```
> res = lm(y2000 ~ y1970, data=homedata)
> pred(res, newdata=dataframe(y1970=80000))
> predict(res, newdata=data.frame(y1970=80000))
[1] 321185
> plot(resid(res))              # simple plot
> plot(homedata$y1970,resid(res)) # shows spread
```

10.12 (p. 298) Not very, as a plot of residuals will show.

⋆ **10.13 (p. 299)** The linear relationship seems appropriate from a scatterplot. But the plot of the residuals versus the fitted values will show that the variance seems to increase for larger values of the defect size.

10.14 (p. 299) We have the following values:

```
> lm(scale(Time) ~ age, by.dist[['100']], subset = age < 70)
...
Coefficients:
(Intercept)          age
   -0.97239      0.00936
```

```
> lm(scale(Time) ~ age, by.dist[['400']], subset = age < 70)
...
Coefficients:
(Intercept)          age
    -1.3951       0.0156

> lm(scale(Time) ~ age, by.dist[['10000']], subset = age < 70)
...
Coefficients:
(Intercept)          age
    -1.6203       0.0201
```

10.15 (p. 299) The null hypothesis that $\beta_1 = 1$ is the model that a child's height is equal to that of his or her parents plus or minus some normally distributed error term (not necessarily mean 0). To see whether this model fits the data, we can compute the *p*-value. We use the output of `summary()` for the estimated values.

```
> res = lm(child ~ parent, data=galton)
> summary(res)
...
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  23.9415     2.8109    8.52   <2e-16 ***
parent        0.6463     0.0411   15.71   <2e-16 ***
...
> T = (0.6463-1)/0.0411
> 2*pt(T,df = 926)
[1] 3.212e-17
```

The small *p*-value cast doubt on the data having been produced by the null hypothesis.

⋆ **10.16 (p. 299)** If `res` stores the model, this can be done as follows:

```
> age.sort = sort(unique(age))
> pred.res = predict(res, newdata = data.frame(age = age.sort),int="conf")
> plot(mhr ~ age); abline(res)
> lines(age.sort,pred.res[,3], lty=2)
> lines(age.sort,pred.res[,2], lty=2)
```

The only change is the abbreviated `int="cont"`.

10.17 (p. 300) The residual plot can be produced with the commands

```
> res = lm(log(lab.defect) ~ log(field.defect), data=alaska.pipeline)
> plot(resid(res) ~ log(field.defect), data=alaska.pipeline)
```

The scatterplot is consistent with the assumption that the error terms have a common distribution.

10.18 (p. 300) A simulation should show quite clearly that the two variables are negatively correlated.

10.19 (p. 300) The confidence ellipse is drawn with these commands:

```
> library(ellipse)
> res = lm(Deflection ~ Load, data=deflection)
> plot(ellipse(res),type="l")
```

The plot shows the negative correlation between the two estimated values.

⋆ **10.20 (p. 310)** The output of `summary()` includes

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -101.9075    23.2918   -4.38  1.4e-05 ***
gestation      0.4503     0.0391   11.52  < 2e-16 ***
age            0.1350     0.1881    0.72    0.473
ht             1.2230     0.2852    4.29  2.1e-05 ***
wt1            0.0308     0.0343    0.90    0.369
dage           0.0603     0.1655    0.36    0.716
dht           -0.0783     0.2706   -0.29    0.772
dwt            0.0783     0.0331    2.37    0.018 *
```

So the coefficients for `gestation`, `ht`, and `dwt` are flagged.

The AIC choice is found with `stepAIC()`

```
> library(MASS)                # to load stepAIC
> stepAIC(res.lm)
...
Call:
lm(formula = wt ~ gestation + age + ht + dwt, data = b)

Coefficients:
(Intercept)    gestation          age           ht          dwt
  -109.1946       0.4532       0.2156       1.3016       0.0756
```

⋆ **10.21 (p. 310)** No. The p-value is 0.14.

```
> init.h = c(600,700,800,950,1100,1300,1500)
> h.d = c(253, 337, 395, 451, 495, 534, 573)
> res.lm3 = lm(h.d ~ init.h + I(init.h^2) + I(init.h^3))
> res.lm4 = update(res.lm3, . ~ . + I(init.h^4))
> anova(res.lm3, res.lm4)
Analysis of Variance Table

Model 1: h.d ~ init.h + I(init.h^2) + I(init.h^3)
Model 2: h.d ~ init.h + I(init.h^2) + I(init.h^3) + I(init.h^4)
  Res.Df  RSS Df Sum of Sq    F Pr(>F)
1      3 48.3
2      2 12.7  1      35.5 5.58   0.14
```

10.22 (p. 310) We can fit the additive model using `lm()`:

```
> res = lm(Volume ~ Girth + Height, data=trees)
```

This command produces a residual plot to assess the fit:

```
> plot(fitted(res),resid(res))
```

The residual plot shows that the assumption of *i.i.d.* error terms is a bit suspicious, as the variance in the error distribution seems to get larger as the fitted value gets larger.

10.23 (p. 311) The model is fit and the residuals plotted using these commands:

```
> res = lm(attendance ~ year + runs.scored + wins + games.behind,
+ data=MLBattend)
> plot(fitted(res), resid(res))
```

The residual plot shows that for high and low attendances the spread seems smaller, but the bulk of the data seems to satisfy the assumptions on the error terms. The `summary()` function reports the following:

```
> summary(res)

Call:
lm(formula = attendance ~ year + runs.scored + wins + games.behind,
    data = MLBattend)

Residuals:
     Min       1Q   Median       3Q      Max
-1353528  -434532   -92768   347736  2857735

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   -604375     245261   -2.46    0.014 *
year             5224       1198    4.36  1.4e-05 ***
runs.scored      2787        270   10.34  < 2e-16 ***
wins             3089       2900    1.07    0.287
games.behind   -15555       2563   -6.07  2.0e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 622000 on 833 degrees of freedom
Multiple R-Squared: 0.326,      Adjusted R-squared: 0.323
F-statistic:  101 on 4 and 833 DF,  p-value: <2e-16
```

The only variable not flagged as significant is the number of wins.

10.24 (p. 311) In R, the quadratic model can be fit using `I()` to insulate the powers.

```
> res.1 = lm(Deflection ~ Load, data=deflection)
> res.2 = update(res.1, . ~ . + I(Load^2))
> plot(fitted(res.1),resid(res.1)) # clearly a bad model
> plot(fitted(res.2),resid(res.2)) # looks good
```

The first residual plot shows that there is a further trend in the data not explained by the linear model. The second residual plot does not indicate this.

⋆ **10.25 (p. 311)** We can fit the models using `update()`:

```
> res.1 = lm(weight ~ age + height, data=kid.weights)
> res.2 = update(res.1, . ~ . + I(height^2))
> res.3 = update(res.2, . ~ . + I(height^3))
> res.4 = update(res.3, . ~ . + I(height^4))
> anova(res.1,res.2,res.3,res.4)
Analysis of Variance Table

Model 1: weight ~ age + height
Model 2: weight ~ age + height + I(height^2)
Model 3: weight ~ age + height + I(height^2) + I(height^3)
Model 4: weight ~ age + height + I(height^2) + I(height^3) + I(height^4)
  Res.Df   RSS  Df Sum of Sq    F  Pr(>F)
1    247 33408
2    246 30604   1      2803 25.4 9.1e-07 ***
3    245 27880   1      2724 24.7 1.3e-06 ***
4    244 26931   1       949  8.6  0.0037 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The ANOVA table shows that for each nested model the new term is statistically significant. The full model is the one selected by this criteria.

10.26 (p. 311) The model is fit directly with the following commands:

```
> res = lm(body.fat ~ age + weight + height + BMI + neck +
+ chest + abdomen + hip + thigh + knee + ankle + bicep + forearm +
+ wrist, data = fat)
```

The `stepAIC()` function selects the best model based on the AIC criteria. For the R^2 value, we find it in the output of `summary()`.

```
> stepAIC(res)                  # after library(MASS)
Start:  AIC= 712.7
...
Call:
lm(formula = body.fat ~ age + weight + neck + abdomen + hip +
   thigh + forearm + wrist, data = fat)

Coefficients:
(Intercept)          age       weight         neck      abdomen
   -20.0621       0.0592      -0.0841      -0.4319       0.8772
        hip        thigh      forearm        wrist
    -0.1864       0.2864       0.4825      -1.4049

> summary(res)
...
Multiple R-Squared: 0.749,      Adjusted R-squared: 0.734
F-statistic: 50.5 on 14 and 237 DF,  p-value: <2e-16
```

(The above modeling can be done with much less typing using the fact that the model involves all but the first, third, and fourth variables. With this insight, these values can be excluded and the model fit using the `.` to imply the remaining variables.

```
> res = lm(body.fat ~ ., data=fat[,-c(1,3,4)])
```

⋆ **10.27 (p. 311)** Only the `Weight` variable is selected.

```
> library(MASS)                 # load data set
> res = lm(MPG.city ~ EngineSize + Weight + Passengers + Price, data=Cars93)
> summary(res)

Call:
lm(formula = MPG.city ~ EngineSize + Weight + Passengers + Price,
    data = Cars93)

Residuals:
    Min      1Q  Median      3Q     Max
-6.1207 -1.9098  0.0522  1.1294 13.9580

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 46.38941    2.09752   22.12  < 2e-16 ***
EngineSize   0.19612    0.58888    0.33     0.74
Weight      -0.00821    0.00134   -6.11  2.6e-08 ***
Passengers   0.26962    0.42495    0.63     0.53
Price       -0.03580    0.04918   -0.73     0.47
```

```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
...
> stepAIC(res)
Start:  AIC= 212.9
 MPG.city ~ EngineSize + Weight + Passengers + Price
...
Call:
lm(formula = MPG.city ~ Weight, data = Cars93)

Coefficients:
(Intercept)       Weight
   47.04835     -0.00803
```

10.28 (p. 311) In a run of ten times, the full model was selected only six times.

⋆ **10.29 (p. 312)** This can be done as follows:

```
> n = length(year)
> yt = log(Nt[-1]/Nt[-n])
> nt = Nt[-n]
> detach(baycheck)
> lm(yt ~ nt)
  Coefficients:
(Intercept)           nt
   0.345810    -0.000409
```

So $\hat{r} = \widehat{\beta}_0 = 0.34$ and $\hat{K} = -\hat{r}/\widehat{\beta}_1 = 845.5$.

Chapter 11

Analysis of variance

11.1 Solutions to Problems

⋆ **11.1 (p. 321)** The data is stored so that the function `oneway.test()` is straightforward to use:

```
> names(morley)
[1] "Expt"  "Run"   "Speed"
> oneway.test(Speed ~ Expt, data=morley)

        One-way analysis of means (not assuming equal variances)

data:  Speed and Expt
F = 3.006, num df = 4.00, denom df = 47.04, p-value = 0.02738
```

The small p-value is consistent with the impression left after considering side-by-side boxplots that the centers are not the same.

11.2 (p. 322) The data is stored in a numeric variable containing the mileage and a factor recording the drive train. This is exactly how `oneway.test()` expects the data:

```
> library(MASS)                 # load data set
> oneway.test(MPG.highway ~ DriveTrain, data=Cars93)

        One-way analysis of means (not assuming equal variances)

data:  MPG.highway and DriveTrain
F = 10.72, num df = 2.00, denom df = 22.53, p-value = 0.0005342
```

The tiny p-value should be expected, since more economy cars are front-wheel drive, these cars tend to get better gas mileage.

⋆ **11.3 (p. 322)** A boxplot will show that the income data, like most income data, is heavily skewed.

```
> boxplot(income ~ race, data=female.inc)
```

An analysis of variance using `oneway.test()` would be inappropriate. However, the three population distributions appear to be roughly the same shape. The Kruskal-Wallis test then is appropriate.

```
> kruskal.test(income ~ race, data=female.inc)

        Kruskal-Wallis rank sum test
```

```
data:  income by race
Kruskal-Wallis chi-squared = 11.79, df = 2, p-value = 0.002748
```

The small p-value is not consistent with the assumption of equal mean wages for all three races represented in the data.

11.4 (p. 322) For `Driver.deaths`, the boxplot shows that an assumption of equal variances seems incorrect, but the normality assumption plausible, as the data is not very skewed for any of the types.

```
> boxplot(Driver.deaths ~ type, data=carsafety)
```

The `oneway.test()` returns a small p-value, as might have been guessed from the boxplot, as the data does not appear to come from populations with the same center.

```
> oneway.test(Driver.deaths ~ type, data=carsafety,var.equal=FALSE)

        One-way analysis of means (not assuming equal variances)

data:  Driver.deaths and type
F = 26.46, num df = 6.000, denom df = 8.785, p-value = 3.75e-05
```

The p-value for `Other.deaths` is surprisingly bigger than 0.10, indicating that the null hypothesis of equal populations means plausibly describes the data set.

```
> boxplot(Other.deaths ~ type, data=carsafety)
> oneway.test(Other.deaths ~ type, data=carsafety,var.equal=FALSE)

        One-way analysis of means (not assuming equal variances)

data:  Other.deaths and type
F = 2.431, num df = 6.000, denom df = 6.964, p-value = 0.1357
```

11.5 (p. 322) The boxplot shows that the parent populations likely do not have the same variances. Although the `not.a.member` population seems longer tailed than the normal, we perform the analysis of variance with `oneway.test()`. The resulting p-value is not consistent with the null hypothesis of equal batting averages for all three types of players.

```
> boxplot(BA ~ Hall.Fame.Membership, data=hall.fame)
> oneway.test(BA ~ Hall.Fame.Membership, data=hall.fame, var.equal=FALSE)

        One-way analysis of means (not assuming equal variances)

data:  BA and Hall.Fame.Membership
F = 160.9, num df = 2.00, denom df = 88.48, p-value < 2.2e-16
```

⋆ **11.6 (p. 322)** Before using either function, the data is put into the form of a numeric variable to record the values and a factor to record the laboratory.

```
> lab1 = c(4.13, 4.07, 4.04, 4.07, 4.05)
> lab2 = c(3.86, 3.85, 4.08, 4.11, 4.08)
> lab3 = c(4.00, 4.02, 4.01, 4.01, 4.04)
> lab4 = c(3.88, 3.89, 3.91, 3.96, 3.92)
> chems = stack(data.frame(lab1,lab2,lab3,lab4))
> boxplot(values ~ ind, data=chems) # var.equal unlikely
> oneway.test(values ~ ind, data=chems, var.equal=FALSE)

        One-way analysis of means (not assuming equal variances)
```

```
data:  values and ind
F = 18.71, num df = 3.000, denom df = 7.914, p-value = 0.0005927
```

The null hypothesis is that each lab has the same mean. The small *p*-value suggests that the data is not consistent with this assumption.

11.7 (p. 323) The data is entered into a list for viewing with boxplots. These suggest the default of not assuming equal variances is appropriate. The resulting *p*-value is consistent with the boxplots, which show much smaller values for Type 1, indicating that the null hypothesis of equal means does not describe the data well.

```
> type1 = c(303, 293, 296, 299, 298)
> type2 = c(322, 326, 315, 318, 320, 320)
> type3 = c(309, 327, 317, 315)
> wear = list(type1=type1,type2=type2,type3=type3)
> boxplot(wear)
> oneway.test(values ~ ind, data=stack(wear))

        One-way analysis of means (not assuming equal variances)

data:  values and ind
F = 46.47, num df = 2.000, denom df = 6.417, p-value = 0.0001522
```

11.8 (p. 323) The test returns a small *p*-value:

```
> kruskal.test(weight ~ group, PlantGrowth)$p.value
[1] 0.01842
```

11.9 (p. 323) The following commands produce the two *p*-values in a manner identical to the example.

```
> x = c(63, 64, 95, 64, 60, 85)
> y = c(58, 56, 51, 84, 77)
> z = c(85, 79, 59, 89, 80, 71, 43)
> d = stack(list("test 1"=x,"test 2"=y,"test 3"=z))
> kruskal.test(values ~ ind, data=d)

        Kruskal-Wallis rank sum test

data:  values by ind
Kruskal-Wallis chi-squared = 1.775, df = 2, p-value =
0.4116

> oneway.test(values ~ ind, data=d)

        One-way analysis of means (not assuming equal variances)

data:  values and ind
F = 0.3691, num df = 2.000, denom df = 9.645, p-value =
0.7007
```

Although the *p*-values are different, they suggest that the data could have been produced under the null hypothesis.

⋆ **11.10 (p. 330)** The commands to perform this analysis are

```
> summary(lm(attendance ~ league, data=MLBattend))
...
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1716321      36259   47.33   <2e-16 ***
leagueNL      127296      52093    2.44    0.015 *
...
```

We see that the mean difference of 127,296 fans is significant with a p-value of 0.015.

11.11 (p. 331) Enforcing the speed limit does have an effect:

```
> summary(lm(y ~ limit, data = Traffic))
...
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   23.130      0.799   28.95   <2e-16 ***
limityes      -4.217      1.305   -3.23   0.0015 **
---
...
> summary(lm(y ~ factor(year), data = Traffic))
...
Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)         22.76       0.91   25.02   <2e-16 ***
factor(year)1962    -2.42       1.29   -1.88    0.061 .
---
...
```

⋆ **11.12 (p. 331)** The data can be entered as follows:

```
> type1 = c(303, 293, 296, 299, 298)
> type2 = c(322, 326, 315, 318, 320, 320)
> type3 = c(309, 327, 317, 315)
> wear = list(type1=type1,type2=type2,type3=type3)
> wear.st = stack(wear)
```

The use of `stack()` stores the data in two variables: the numeric information in `values` and a factor indicating the type in `ind`. The p-value from `oneway.test()` can be returned succinctly with

```
> oneway.test(values ~ ind, data = wear.st)$p.value
[1] 0.0001522
```

Using `lm()`, this p-value is returned by the extractor function `summary()` in the section labeled `F-statistic`.

```
> summary(lm(values ~ ind, data = wear.st))
...
F-statistic:   31 on 2 and 12 DF,  p-value: 1.81e-05
```

Why the difference? The F-test assumes equal variances, whereas the default for `oneway.test()` assumes unequal variances. Changing the default produces equivalent answers.

```
> oneway.test(values ~ ind, data = wear.st, var.equal=TRUE)$p.value
[1] 1.810e-05
```

11.13 (p. 331) This can be done with the commands

```
> summary(lm(mpg ~ factor(cyl), data=mtcars))
```

The `factor()` function is used, as `cyl` is stored as a numeric variable.

⋆ **11.14 (p. 331)** The boxplot of `amount` broken up by year shows skewed data.

```
> boxplot(amount ~ factor(year), data=npdb)
```

Once a logarithm is taken, the data appears to come from a symmetric distribution.

```
> boxplot(log(amount) ~ factor(year), data=npdb)
```

A one-way analysis of variance can be performed using `lm()` as follows:

```
> res = lm(log(amount) ~ factor(year), subset= year < 2003, npdb)
> summary(res)

Call:
lm(formula = log(amount) ~ factor(year), data = npdb, subset = year <
    2003)

Residuals:
    Min      1Q  Median      3Q     Max
-6.7957 -1.2743  0.0014  1.4730  6.3266

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)       10.7078     0.0276  387.40   <2e-16 ***
factor(year)2001  -0.4872     0.0519   -9.39   <2e-16 ***
factor(year)2002  -1.2851     0.0955  -13.45   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.87 on 6789 degrees of freedom
Multiple R-Squared: 0.0336,     Adjusted R-squared: 0.0333
F-statistic:  118 on 2 and 6789 DF,  p-value: <2e-16
```

The p-value for the F-test is tiny, indicating that the null hypothesis is not likely to have yielded this data.

11.15 (p. 331) This can be done as follows. The difference is significant.

```
> summary(lm(mpg ~ factor(am), data=mtcars))
...
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    17.15       1.12   15.25  1.1e-15 ***
factor(am)1     7.24       1.76    4.11  0.00029 ***
...
```

⋆ **11.16 (p. 331)** We need to compare the following

```
> summary(lm(Speed ~ factor(Expt),data=morley))
...
Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)         909.0       16.6   54.76  < 2e-16 ***
factor(Expt)2       -53.0       23.5   -2.26  0.02625 *
factor(Expt)3       -64.0       23.5   -2.73  0.00763 **
factor(Expt)4       -88.5       23.5   -3.77  0.00028 ***
factor(Expt)5       -77.5       23.5   -3.30  0.00136 **
```

To the output of `TukeyHSD()`

```
> TukeyHSD(aov(Speed ~ factor(Expt), data=morley))
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = Speed ~ factor(Expt), data = morley)

$"factor(Expt)"
     diff     lwr    upr
2-1 -53.0 -118.28  12.28
3-1 -64.0 -129.28   1.28
4-1 -88.5 -153.78 -23.22
5-1 -77.5 -142.78 -12.22
3-2 -11.0  -76.28  54.28
4-2 -35.5 -100.78  29.78
5-2 -24.5  -89.78  40.78
4-3 -24.5  -89.78  40.78
5-3 -13.5  -78.78  51.78
5-4  11.0  -54.28  76.28
```

The latter flags `4-1`, `5-1` and `5-4`. The marginal tests find that all the means are different from the reference level `1`.

11.17 (p. 331) The `TukeyHSD()` function is an extractor function for models produced by `aov()`, not `lm()`.

```
> res.aov = aov(Other.deaths ~ type, data=carsafety)
> TukeyHSD(res.aov)
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = Other.deaths ~ type, data = carsafety)

$type
                       diff    lwr      upr
compact-SUV        -18.464 -45.50   8.5728
large-SUV          -17.048 -46.81  12.7192
...
subcompact-minivan  -1.429 -31.20  28.3383
subcompact-pickup  -59.929 -94.51 -25.3426
```

We can scan through the output or plot the results to find out which intervals do not straddle 0. A quick scan finds the following pairs involving pickups have statistically significant differences: `pickup-SUV`, `pickup-compact`, `pickup-large`, `pickup-midsize`, `pickup-minivan`, and `subcompact-pickup`.

⋆ **11.18 (p. 332)** The following commands will show that `B` and `F` are:

```
> plot(count ~ spray, InsectSprays)
> res.aov = aov(count ~ spray, InsectSprays)
> summary(res.aov)
> plot(TukeyHSD(res.aov))
```

⋆ **11.19 (p. 334)** The ANCOVA is performed as follows:

```
> res = lm(time ~ age + gender, nym.2002)
> summary(res)
...
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  241.450      6.250   38.63  < 2e-16 ***
age            1.226      0.155    7.92  6.4e-15 ***
genderMale   -29.397      3.633   -8.09  1.7e-15 ***
...
```

The difference is estimated to be a half-hour in total time.

11.20 (p. 334) The requested model is fit using `lm()`.

```
> res = lm(mpg ~ wt + factor(am), data=mtcars)
> summary(res)
...
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  37.3216     3.0546   12.22  5.8e-13 ***
wt           -5.3528     0.7882   -6.79  1.9e-07 ***
factor(am)1  -0.0236     1.5456   -0.02     0.99
...
```

The transmission type is not flagged as significant, although this is likely a limitation of the model and data set, as vehicles with manual transmissions consistently have higher mileage ratings than those with automatic transmissions.

11.21 (p. 334) The ANCOVA model is fit using `lm()` as follows:

```
> res = lm(wt ~ gestation + wt1 + ht + factor(smoke), data=babies)
> summary(res)
...
Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    84.40283    7.50839   11.24  < 2e-16 ***
gestation       0.01110    0.00667    1.66    0.096 .
wt1            -0.00526    0.00423   -1.24    0.214
ht              0.55888    0.11921    4.69  3.1e-06 ***
factor(smoke)1 -8.94199    1.09763   -8.15  9.1e-16 ***
factor(smoke)2  0.60447    1.95061    0.31    0.757
factor(smoke)3  0.94660    1.89125    0.50    0.617
factor(smoke)9  3.95806    5.59496    0.71    0.479
...
```

The variable `smoke()` is stored numerically but is treated as a factor in the model by using `factor()`. We see from the output of `summary()` that only the mother's height and her smoking status are flagged as statistically significant.

⋆ **11.22 (p. 334)** We fit the ANCOVA model first:

```
> library(MASS)                 # for stepAIC
> kid.weights$BMI = (kid.weights$weight/2.54)/ (kid.weights$height*2.54/100)^2
> res.full = lm(BMI ~ age + gender, kid.weights)
> res.age  = lm(BMI ~ age, kid.weights)
> res.gender  = lm(BMI ~ gender, kid.weights)
> anova(res.full, res.age)
Analysis of Variance Table

Model 1: BMI ~ age + gender
Model 2: BMI ~ age
```

```
  Res.Df  RSS  Df Sum of Sq    F Pr(>F)
1    247 9686
2    248 9686  -1     -0.25 0.01   0.94
> anova(res.full, res.gender)
Analysis of Variance Table

Model 1: BMI ~ age + gender
Model 2: BMI ~ gender
  Res.Df  RSS  Df Sum of Sq    F Pr(>F)
1    247 9686
2    248 9722  -1       -37 0.93   0.33
> stepAIC(res)

...
Call:
lm(formula = BMI ~ 1, data = kid.weights)

Coefficients:
(Intercept)
         17
```

We see that neither variable is significant by `stepAIC()`.

11.23 (p. 334) Without checking the model assumptions, we use `anova()` to see that this is so:

```
> anova(lm(log(INCOME+1) ~ AGE + factor(EDUC), data=cfb))
Analysis of Variance Table

Response: log(INCOME + 1)
              Df Sum Sq Mean Sq F value  Pr(>F)
AGE            1     14      14    12.3 0.00047 ***
factor(EDUC)  16    275      17    15.3 < 2e-16 ***
Residuals    982   1106       1
```

11.24 (p. 335) The ANCOVA model is fit as follows:

```
> res = lm(temperature ~ hr + factor(gender), data=normtemp)
> summary(res)
...
Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     96.25081    0.64872  148.37   <2e-16 ***
hr               0.02527    0.00876    2.88   0.0046 **
factor(gender)2  0.26941    0.12328    2.19   0.0307 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.702 on 127 degrees of freedom
Multiple R-Squared: 0.0983,     Adjusted R-squared: 0.0841
F-statistic: 6.92 on 2 and 127 DF,  p-value: 0.00141
```

Both variables are flagged as significant at the 0.05 level.

⋆ **11.25 (p. 342)** The data can be entered and models fit as follows:

```
> likability = c(-1,4,0,-1,4,1,6,2,7,1,2,2,7,5,2,3,6,1)
> web = factor(rep(c("N","Y"), c(9,9)))
```

```
> tv = factor(rep(c(0,"1-2","3+"), c(6,6,6)))
> res.web = lm(likability ~ web)
> res.tv = lm(likability ~ tv)
> res.both = lm(likability ~ tv + web)
```

To see whether the web itself has an effect we can look at the output of `summary()`:

```
> summary(res.web)
...
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    2.444      0.873    2.80    0.013 *
webY           0.778      1.235    0.63    0.538
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.62 on 16 degrees of freedom
Multiple R-Squared: 0.0242,     Adjusted R-squared: -0.0368
F-statistic: 0.397 on 1 and 16 DF,  p-value: 0.538
```

Web advertising is not effective in this assessment. However, controlling first for television exposure we have

```
> anova(res.tv, res.both)
Analysis of Variance Table

Model 1: likability ~ tv
Model 2: likability ~ tv + web
  Res.Df  RSS Df Sum of Sq    F Pr(>F)
1     15 86.2
2     14 69.5  1      16.7 3.36  0.088 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

That is, the effect of web advertising seems more likely. It may be worthwhile to perform this test on a bigger sample to try to detect any differences. It is difficult to detect differences with so few observations per category.

⋆ **11.26 (p. 342)** This can be carried out as follows:

```
> with(grip, interaction.plot(grip.type,person,UBP))
> res.int = lm(UBP ~ person * grip.type, data=grip)
> res.noint = lm(UBP ~ person + grip.type, data=grip)
> res.per = lm(UBP ~ person, data=grip)
> res.grip = lm(UBP ~ grip.type, data=grip)
> res.none = lm(UBP ~ 1, data=grip)
```

Now, we check to see what is statistically significant:

```
> anova(res.noint,res.int)
Analysis of Variance Table

Model 1: UBP ~ person + grip.type
Model 2: UBP ~ person * grip.type
  Res.Df RSS Df Sum of Sq    F Pr(>F)
1     30 509
2     24 484  6        25 0.21   0.97
```

This agrees with the interaction plot. No evidence of an interaction is present.

```
> anova(res.none,res.per)
Analysis of Variance Table
```

```
Model 1: UBP ~ 1
Model 2: UBP ~ person
  Res.Df RSS Df Sum of Sq    F Pr(>F)
1     35 876
2     32 848  3        27 0.34    0.8
```

No evidence that the difference varies among subjects (recall that this is simulated data).

```
> anova(res.none,res.grip)
Analysis of Variance Table

Model 1: UBP ~ 1
Model 2: UBP ~ grip.type
  Res.Df RSS Df Sum of Sq    F  Pr(>F)
1     35 876
2     33 536  2       339 10.4 0.00031 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The effect of the grip seems significant.

Finally, we see that `stepAIC()` returns the same conclusion.

```
> library(MASS)
> stepAIC(res.int)
...
Call:
lm(formula = UBP ~ grip.type, data = grip)
...
```

11.27 (p. 342) The two-way ANOVA with interaction is fit as follows:

```
> res.full = lm(mpg ~ factor(cyl) * factor(am), data=mtcars)
> summary(res.full)
...
(Intercept)              ***
factor(cyl)6
factor(cyl)8             ***
factor(am)1              *
factor(cyl)6:factor(am)1
factor(cyl)8:factor(am)1
...
```

The edited output shows that the two interactions are not flagged as statistically significant. We refit the models as follows:

```
> res.add = lm(mpg ~ factor(cyl) + factor(am), data=mtcars)
> res.cyl = lm(mpg ~ factor(cyl), data=mtcars)
> res.am  = lm(mpg ~ factor(am), data=mtcars)
> anova(res.am, res.add)
Analysis of Variance Table

Model 1: mpg ~ factor(am)
Model 2: mpg ~ factor(cyl) + factor(am)
  Res.Df RSS Df Sum of Sq    F Pr(>F)
1     30 721
2     28 264  2       456 24.2  8e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
> anova(res.cyl, res.add)
Analysis of Variance Table

Model 1: mpg ~ factor(cyl)
Model 2: mpg ~ factor(cyl) + factor(am)
  Res.Df   RSS Df Sum of Sq    F Pr(>F)
1     29 301.3
2     28 264.5  1      36.8 3.89  0.058 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The F-tests indicate that both variables are significant.

⋆ **11.28 (p. 342)** We proceed as follows:

```
> with(ToothGrowth, interaction.plot(dose, supp, len))
> res.full = lm(len ~ supp + dose, ToothGrowth)
> res.add = lm(len ~ supp * dose, ToothGrowth)
> anova(res.full, res.add)
Analysis of Variance Table

Model 1: len ~ supp + dose
Model 2: len ~ supp * dose
  Res.Df  RSS Df Sum of Sq    F Pr(>F)
1     57 1023
2     56  934  1        89 5.33  0.025 *
```

11.29 (p. 342) The interaction plot is made with the commands

```
> with(OrchardSprays, interaction.plot(rowpos,treatment,decrease))
```

At a glance, the lines do not appear to be “parallel,” indicating an interaction. The formal F-test for this is performed with `anova()`.

```
> res.full = lm(decrease ~ rowpos * treatment, data=OrchardSprays)
> res.add  = lm(decrease ~ rowpos + treatment, data=OrchardSprays)
> anova(res.full, res.add)
Analysis of Variance Table

Model 1: decrease ~ rowpos * treatment
Model 2: decrease ~ rowpos + treatment
  Res.Df   RSS Df Sum of Sq    F Pr(>F)
1     48 19718
2     55 20965 -7     -1247 0.43   0.88
```

The null hypothesis tested is that the interactions have no effect. In this case, the large p-value indicates that the data could likely have been produced by the additive model, and that there is no need for the multiplicative model to describe the data.

Chapter 12

Two extensions of the linear model

12.1 Solutions to Problems

⋆ **12.1 (p. 356)** The logistic regression is carried out as follows:

```
> res = glm(enjoyed ~ gender + age, data=tastesgreat,
+ family=binomial)
> summary(res)
...
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -8.1844     3.0964   -2.64   0.0082 **
genderMale    2.4224     0.9559    2.53   0.0113 *
age           0.1649     0.0652    2.53   0.0114 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 55.452  on 39  degrees of freedom
Residual deviance: 38.981  on 37  degrees of freedom
AIC: 44.98

Number of Fisher Scoring iterations: 5
```

This shows that both are flagged as significant.

12.2 (p. 357) This is performed with these commands:

```
> library(MASS)                 # load stepAIC
> stepAIC(glm(healthy ~ p + g, healthy, family=binomial))
...
Call:  glm(formula = healthy ~ p, family = binomial, data = healthy)

Coefficients:
(Intercept)            p
```

```
      -6.85         1.83

Degrees of Freedom: 31 Total (i.e. Null);  30 Residual
Null Deviance:      30.9
Residual Deviance: 24.8         AIC: 28.8
```

The preferred model by the AIC criteria involves p but not g.

12.3 (p. 357) We use `glm()` with the `family="binomial"` argument to perform the logistic regression.

```
> library(MASS)
> res.glm = glm(low ~ age + lwt + smoke + ht + ui, data=birthwt,
+ family = binomial)
> summary(res.glm)
...
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.39979    1.08041    1.30   0.1951
age         -0.03407    0.03367   -1.01   0.3116
lwt         -0.01545    0.00659   -2.35   0.0190 *
smoke        0.64754    0.33665    1.92   0.0544 .
ht           1.89327    0.68339    2.77   0.0056 **
ui           0.88461    0.44405    1.99   0.0464 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 234.67  on 188  degrees of freedom
Residual deviance: 211.78  on 183  degrees of freedom
AIC: 223.8

Number of Fisher Scoring iterations: 4
```

Only the age variable is not flagged as statistically significant. If we let `stepAIC()` select the model, we end up with the same conclusion.

```
> stepAIC(res.glm)
... Lots skipped ...
Call:  glm(formula = low ~ lwt + smoke + ht + ui, family = binomial,
           data = birthwt)

Coefficients:
(Intercept)          lwt        smoke           ht           ui
     0.7219      -0.0163       0.6530       1.9221       0.8963

Degrees of Freedom: 188 Total (i.e. Null);  184 Residual
Null Deviance:      235
Residual Deviance: 213  AIC: 223
```

12.4 (p. 357) The modeling is done using `glm()` with the `family="binomial"` argument.

```
> fm = hall.fame$Hall.Fame.Membership != "not a member"
> res = glm(fm ~ BA + HR + hits + games, data=hall.fame,
+ family="binomial")
> stepAIC(res)
```

```
Start:  AIC= 390.1
...
Call:  glm(formula = fm ~ BA + HR + hits, family = "binomial", data = hall.fame)

Coefficients:
(Intercept)           BA           HR         hits
  -21.84376     51.21222      0.00428      0.00250

Degrees of Freedom: 1339 Total (i.e. Null);  1336 Residual
Null Deviance:      826
Residual Deviance: 380  AIC: 388
```

The confounded variable `games` is dropped by this criteria.

12.5 (p. 357) Fitting the two models and then calling `AIC()` shows that the model without interactions is preferred.

```
> res.full <- glm(cbind(ncases, ncontrols) ~ agegp + tobgp * alcgp,
+ data = esoph, family = binomial())
> res.add  <- glm(cbind(ncases, ncontrols) ~ agegp + tobgp +  alcgp,
+ data = esoph, family = binomial())
> AIC(res.full)
[1] 237.0
> AIC(res.add)
[1] 225.5
```

12.6 (p. 357) This can be done by first plotting to guess initial estimates.

```
> plot(circumference ~ age, data=Orange, subset = Tree == 1)
```

From this we estimate `Y` to be 150, `k` to be about $1/300$, and `t0` to be 750. (See the help page `?Orange` to learn how to use the self-starting function `SSlogis()` to do this.)

```
> res.tree = nls(circumference ~ g(age,Y,k,t0),
+ start=c(Y=150,k=1/300,t0=750),
+ data=Orange, subset=Tree == 1)
> res.tree
Nonlinear regression model
  model:  circumference ~ g(age, Y, k, t0)
   data:  Orange
       Y         k        t0
1.542e+02 2.758e-03 6.272e+02
 residual sum-of-squares:  177
```

We can add this curve to our graph with

```
> age = 0:1500
> lines(age,predict(res.tree,data.frame(age=age)))
```

12.7 (p. 357) The logistic function is first defined; then a plot is drawn so that the initial guesses can be made.

```
> f = function(t,Y,k,t0)  Y * (1 + exp(-k*(t-t0)))^(-1)
> plot(weight ~ Time, data=ChickWeight, subset= Chick == 1)
```

The value of Y is the saturation growth amount. A value of 400 or more seems correct. The value of k is the growth rate. The data appears to grow by a factor of e initially in about 10 units of time, so that k is about $1/10$. Finally, t_0 is where the growth begins to "bend over," which appears to be happen after 20. We fit `nls()` with these values using the argument `subset=` to refer only to the data for chick number 1.

```
> nls(weight ~ f(Time, Y, k, t0), start=c(Y=400, k=1/10, t0=20),
+ subset= Chick == 1, data=ChickWeight)
Nonlinear regression model
  model:  weight ~ f(Time, Y, k, t0)
   data:  ChickWeight
       Y         k        t0
937.01502   0.08768  35.22270
 residual sum-of-squares:  76.66
```

Adding the curve to the graph can be done with `curve()`.

```
> curve(f(x, Y=937, k=.08768, t0=35.22270), add=T)
```

⋆ **12.8 (p. 358)** This model is actually done in the help page for the data set. We can fit the model by running the command

```
> library(MASS)                 # load in help page
> example(wtloss)
> wtloss.fm
Nonlinear regression model
  model:  Weight ~ b0 + b1 * 2^(-Days/th)
   data:  wtloss
    b0     b1     th
 81.37 102.68 141.91
 residual sum-of-squares:  39.24
> plot(Weight ~ Days, wtloss)   # make scatterplot
> lines(predict(wtloss.fm) ~ Days, data = wtloss)
```

The value of `b0` is the predicted long-term weight. If we want the predicted amount after 365 days we have

```
> predict(wtloss.fm, newdata=data.frame(Days=365))
[1] 98.64
```

(Of course, predictions beyond the range of the data are not a good idea.)

⋆ **12.9 (p. 358)** We can fit both using the same starting values and compare:

```
> l =  function(t,a,b,k,t0) (a + b*t)*(1 - exp(-k*(t-t0)))
> l1 = function(t,a,k,t0) l(t,a,0,k,t0)
> res.l = nls(length ~ l(age,a,b,k,t0), data=reddrum,
+  start=c(a=32,b=.25,k=.5,t0=0))
> res.l1 = nls(length ~ l1(age,a,k,t0), data=reddrum,
+ start=c(a=32,k=.5,t0=0))
> AIC(res.l)
[1] 306.7
> AIC(res.l1)
[1] 376.2
```

So the more complicated model is better by this criteria.

(The point of the paper that this data came from is to show that both of these models pale in comparison to the more complicated one they presented, which takes into account seasonal growth and a decrease in growth rate due to age.)

⋆ **12.10 (p. 358)** First we make a new year variable, then we fit the model. We use the new value of the car to estimate N.

```
> year = with(midsize, 2004-Year)
> f = function(x,N,r) N*exp(-r*x)
> with(midsize,nls(Accord ~ f(year,N,r),start=c(N=Accord[1],r=1/5)))
```

```
Nonlinear regression model
  model:  Accord ~ f(year, N, r)
   data:  parent.frame()
       N         r
1.670e+04 1.403e-01
 residual sum-of-squares:  1311037
> with(midsize,nls(Camry ~ f(year,N,r),start=c(N=Camry[1],r=1/5)))
Nonlinear regression model
  model:  Camry ~ f(year, N, r)
   data:  parent.frame()
       N         r
1.895e+04 1.578e-01
 residual sum-of-squares:  2288709
> with(midsize,nls(Taurus ~ f(year,N,r),start=c(N=Taurus[1],r=1/5)))
Nonlinear regression model
  model:  Taurus ~ f(year, N, r)
   data:  parent.frame()
       N         r
1.768e+04 2.602e-01
 residual sum-of-squares:  23029827
```

The Accord has the smallest decay rate, and the Taurus the largest.

Chapter 13

Programming in R

13.1 Solutions to Problems

⋆ **E.1 (p. 406)** A function to do so is

```
std = function(x) {
  n = length(x)
  S2 = (1/(n-1)) * sum( (x - mean(x))^2 )
  return(sqrt(S2))
}
```

⋆ **E.2 (p. 407)** These functions are useful for modeling a stack in computer science. They can each be written in one line:

```
> pop  = function(x) x[length(x)]
> push = function(x,val) c(x,val)
```

⋆ **E.3 (p. 407)** A function would look like

```
lag = function(x) {
  n = length(x)
  plot(x[-n],x[-1])
}
```

For the random data, we see a scatter, for the `sin(1:100)` data we see a regular shape.

⋆ **E.4 (p. 407)** We know the confidence interval has the form

$$\bar{X} \pm t^* \mathsf{SE}(\bar{X}).$$

We need to compute t^* for $n-1$ degrees of freedom

```
t.ci = function(Xbar,S,n,conf.level=0.95) {
  alpha = 1 - conf.level
  tstar = qt((1-alpha/2),df=n-1)
  c(Xbar - tstar*S,Xbar + tstar*S)
}
```

⋆ **E.5 (p. 407)** We write a function to implement the method as follows:

```
findzeroes = function(x) {
  delta = 1
  while(delta > .00001) {
    old.x = x
```

```
    x = x - (x^2 - sin(x))/(2*x - cos(x))
    delta = abs(x - old.x)
  }
  x
}
```

Starting at 0 gives 0 and starting at 1 gives 0.8767.

⋆ **E.6 (p. 407)** The `typeof()` is "logical," so we define the result as

```
size.logical = function(x) length(x)
```

Or, we can define the "size" to be the number of `TRUE` values.

⋆ **E.7 (p. 408)** Try this:

```
"/.String" = function(x,y) {unlist(strsplit(x,y))}
```

⋆ **E.8 (p. 408)** Try this (the result is coerced to be a string):

```
"-.string" = function(x,y) { as.string(paste(x/y,collapse=""))}
```

E.9 (p. 408) The following will work

```
setGeneric("upcase", function(object) standardGeneric("upcase"))
setMethod("upcase","String",function(object)   string(toupper(object@string)))

setGeneric("downcase", function(object) standardGeneric("downcase"))
setMethod("downcase","String",function(object)   string(tolower(object@string)))
```

⋆ **E.10 (p. 408)** This is easier to do with regular expressions, but if you don't know these you can split by `" "` and then throw out words of length 0. This can be done using the String class methods. For example:

```
String$defineMethod("strip",function() {
  lengthx = function(x) {
    x = String$new(x)
    if(x$length() > 0)
      TRUE
    else
      FALSE
  }
  y = split(" ")
  aword = sapply(y, lengthx)
  words = y[aword]
  ## join with a space
  paste(words,collapse=" ")
})
```

This would have been easier if the output of `split()` were a vector of type String, but this isn't possible. Instead, the function `sapply()` is used to apply the function `lengthx()`, which calculates the length of the string to each element in the output of `split()`.