Appendix F

Solutions manual

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.

1.9 (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).

1.16 (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
```

1.19 (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.

1.24 (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.

1.27 (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.

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)
```

2.9 (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
```

2.12 (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.

2.17 (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.

2.19 (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
```

2.22 (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
```

2.34 (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
```

2.39 (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.

2.45 (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.

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)
```

3.7 (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.

3.9 (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.

3.12 (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.

3.14 (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
```

3.17 (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
```

3.21 (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.

3.24 (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.

3.28 (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.

3.30 (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.

3.32 (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.

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.

4.5 (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.

4.10 (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?)

4.12 (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
```

4.15 (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.)

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

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

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)
```

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

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

From these, the growths seem to be similar.

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
```

5.9 (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
```

5.12 (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.

5.16 (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
```

5.19 (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.

5.21 (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
```

5.27 (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
```

5.30 (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
```

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.

6.7 (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)
```

6.10 (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.

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
```

```
...
```

7.8 (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
```

7.11 (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.

7.15 (p. 195) These can be found with

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

7.17 (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
```

7.22 (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.

7.24 (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.)

7.27 (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.

7.29 (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.

7.31 (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
```

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.

8.5 (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.

8.9 (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.

8.12 (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
...
```

8.16 (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
```

8.21 (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."

8.27 (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
```

8.35 (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.

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.

9.7 (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.

9.11 (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)
```

9.13 (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
```

9.17 (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.

9.19 (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))
```

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)`.

10.6 (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.

10.9 (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.

10.11 (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
```

10.13 (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.

10.16 (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"`.

10.20 (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
```

10.25 (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.

10.27 (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
```

10.29 (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 $\hat{K} = -\hat{r}/\widehat{\beta}_1 = 845.5$.

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.

11.3 (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.

11.6 (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.

11.10 (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.

11.12 (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
```

11.14 (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.

11.16 (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`.

11.18 (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.

11.22 (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()`.

11.25 (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)
...
```

11.28 (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 *
```

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.

12.8 (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.

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.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()`.