SOLUTIONS MANUAL FOR

USING R FOR INTRODUCTORY STATISTICS

SECOND EDITION

by

John Verzani

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

SOLUTIONS MANUAL FOR

USING R FOR INTRODUCTORY STATISTICS

SECOND EDITION

by

John Verzani

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
A CHAPMAN & HALL BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed on acid-free paper
Version Date: 20140508

International Standard Book Number-13: 978-1-4665-9077-9 (Ancillary)

Contents

1

Getting Started

1.1 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] 4.582576
```

```
( (1+2)/(3+4) )^2
```

```
## [1] 0.1836735
```

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

1.3 Translating this to R requires attention to the use of parentheses and using an asterisk for multiplication:

```
(1 + 2*3^4) / (5/6 - 7)
```

```
## [1] -26.43243
```

1.4 We use the 1/2 power as an alternative to the `sqrt` function:

```
(0.25 - 0.2) / (0.2 * (1 - 0.2)/100)^(1/2)
```

```
## [1] 1.25
```

1.5 We don't use `c` below, as it is a very commonly used function in R:

```
a <- 2; b <- 3; d <- 4; e <- 5
a * b * d * e
```

```
## [1] 120
```

1.6 It is 1770.

1.7 It is `2510`. Instead of scanning, this can be automated:

```
require(UsingR)
```

```
## Loading required package: UsingR
## Loading required package: MASS
##
## Attaching package: 'UsingR'
##
## The following object is masked from 'package:ggplot2':
##
##    movies
##
## The following object is masked from 'package:survival':
##
##    cancer
```

```
max(exec.pay)
```

```
## [1] 2510
```

1.8 These values come from:

```
require(UsingR)
mean(exec.pay)

## [1] 59.88945

min(exec.pay)

## [1] 0

max(exec.pay)

## [1] 2510
```

1.9 This is done with:

```
require(UsingR)
mean(exec.pay)

## [1] 59.88945

mean(exec.pay, trim=0.10)

## [1] 29.96894
```

The big difference is due to the fact that the very large salaries that are trimmed have big influence on the average of the data set computed by `mean`.

1.10 The variable names are printed when the data set is displayed. They are `Tree`, `age`, and `circumference`.

1.11 The only trick is to reference the variable appropriately:

```
mean(Orange$age)
```

```
## [1] 922.1429
```

1.12 The largest value in a collection is returned by `max`:

```
max(Orange$circumference)

## [1] 214
```

2

Univariate data

2.1 For example:

```
p <- c(2, 3, 5, 7, 11, 13, 17, 19)
```

2.2 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.

2.3 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
```

2.4 These can be done with

```
rep("a", 10)
```

```
##  [1] "a" "a" "a" "a" "a" "a" "a" "a" "a" "a"
```

```
seq(1, 99, by=2)
```

```
##  [1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39
## [21] 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79
## [41] 81 83 85 87 89 91 93 95 97 99
```

```
rep(1:3, rep(3,3))
```

```
## [1] 1 1 1 2 2 2 3 3 3
```

```
rep(1:3, 3:1)
```

```
## [1] 1 1 1 2 2 3
```

```
c(1:5, 4:1)
```

```
## [1] 1 2 3 4 5 4 3 2 1
```

2.5 These can be done with the following commands:

```
primes_under_20 <- c(1, 2, 3, 5, 8, 13, 21, 34)
ns <- 1:10
recips <- 1/ns
cubes <- (1:6)^3
years <- 1964:2014
subway <- c(14, 18, 23, 28, 34, 42, 50, 59, 66, 72, 79, 86, 96, 103, 110)
by25 <- seq(0,1000, by=25)
```

2.6 We have:

```
sum(abs(rivers - mean(rivers))) / length(rivers)
```

```
## [1] 313.5508
```

To elaborate, `rivers - mean(rivers)` centers the values and is a data vector. Calling `abs` makes all the values non-negative, and `sum` reduces the result to a single number, which is then divided by the length.

2.7 The unary minus is evaluated before the colon:

```
-1:3                                    # like (-1):3
```

```
## [1] -1  0  1  2  3
```

However, the colon is evaluated before multiplication in the latter:

```
1:2*3                                   # not like 1:(2*3)
```

```
## [1] 3 6
```

2.8 If we know the cities starting with a "J" then this is just an exercise in indexing by the names attribute, as with:

```
precip["Juneau"]
```

```
## Juneau
##   54.7
```

Getting the cities with the names beginning with "J" can be done by sorting and inspecting, say with `sort(names(precip))`. This gives:

```
j_cities <- c("Jackson", "Jacksonville", "Juneau")
precip[j_cities]
```

```
##      Jackson Jacksonville       Juneau
##         49.2         54.5         54.7
```

The inspection of the names by scanning can be tedious for large data sets. The `grepl` function can be useful here, but requires the specifica-

tion of a regular expression to indicate words that start with "J". As a teaser, here is how this could be done:

```
precip[grepl("^J", names(precip))]
```

```
##       Juneau Jacksonville      Jackson
##         54.7         54.5         49.2
```

Regular expressions are described in the help page ?regexp.

2.9 There are many ways to do this, the following uses `paste`:

```
paste("Trial", 1:10)
```

```
##  [1] "Trial 1"  "Trial 2"  "Trial 3"  "Trial 4"  "Trial 5"
##  [6] "Trial 6"  "Trial 7"  "Trial 8"  "Trial 9"  "Trial 10"
```

2.10 This answer will very depending on the underlying system. One answer is:

```
paste(dname, fname, sep=.Platform$file.sep)
```

```
## [1] "/Library/Frameworks/R.framework/Versions/3.2/Resources/library/UsingR/DESCRIPTION"
```

2.11 The number of levels and number of cases are returned by:

```
require(MASS)
man <- Cars93$Manufacturer
length(man)                             # number of cases
```

```
## [1] 93
```

```
length(levels(man))                     # number of levels
```

```
## [1] 32
```

2.12 Looking at the levels, we see that one is `rotary`, which is clearly not numeric. As for the 5-cylinder cars, we can get them as follows:

```
cyl <- Cars93$Cylinders
levels(cyl)                             # "rotary"

## [1] "3"      "4"      "5"      "6"      "8"      "rotary"

which(cyl == "5")                       # just 5 is also okay

## [1] 89 93

Cars93$Manufacturer[ which(cyl == 5) ]  # which companies

## [1] Volkswagen Volvo
## 32 Levels: Acura Audi BMW Buick Cadillac Chevrolet ... Volvo
```

2.13 The `factor` function allows this to be done by specifying the `labels` argument:

```
mtcars$am <- factor(mtcars$am, labels=c("automatic", "manual"))
```

This produces a modified, local copy of `mtcars`. The ordering of the labels should match the following: `sort(unique(as.character(mtcars$am)))`.

2.14 The answer is no:

```
require(HistData)
any(Arbuthnot$Female > Arbuthnot$Male)

## [1] FALSE
```

Read the help page to see how this could be construed to show the "guiding hand of a devine being."

2.15 We have:

```
A <- c(TRUE, FALSE, TRUE, TRUE)
B <- c(TRUE, FALSE, TRUE, TRUE)
!(A & B)

## [1] FALSE  TRUE FALSE FALSE

!A | !B

## [1] FALSE  TRUE FALSE FALSE
```

It is not necessary to express the latter as `(!A) | (!B)`, as the unary `!` operator has higher precedence than the binary `|` operator.

2.16 We use logical extraction for this task:

```
names(precip[precip > 50])

## [1] "Mobile"       "Juneau"       "Jacksonville" "Miami"
## [5] "New Orleans"  "San Juan"
```

2.17 After parsing the question, it can be seen that this expression answers it:

```
m <- mean(precip)
trimmed_m <- mean(precip, trim=0.25)
any(precip > m + 1.5 * trimmed_m)

## [1] FALSE
```

A similar question is used for the algorithmic determination of "outliers" in a data set.

2.18 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.

2.19 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] 22

min(commutes)

## [1] 15

mean(commutes)

## [1] 18.3

sum(commutes >= 20)

## [1] 4

sum(commutes < 18)/length(commutes)

## [1] 0.5
```

2.20 We need to know that the months with 31 days are 1, 3, 5, 7, 8, 10, and 12.

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

## [1] 166.5714

mean(short)

## [1] 205.6
```

2.21 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)
names(x) <- 1990:1999
```

Using `diff` gives

```
diff(x)
```

```
##   1991   1992   1993   1994   1995   1996   1997   1998   1999
##  0.320  0.190  0.040  0.060 -0.110  0.100  0.210  0.061  0.279
```

We can see that one year was negative:

```
which(diff(x) < 0)
```

```
## 1995
##    5
```

The jump between 1994 and 1995 was negative (there was a work stoppage that year). 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
```

```
##      1991      1992      1993      1994      1995      1996
## 56.140351 21.348315  3.703704  5.357143 -9.322034  9.345794
##      1997      1998      1999
## 17.948718  4.420290 19.361554
```

2.22 We have:

```
mean_distance <- function(x) {
  distances <- abs(x - mean(x))
  mean(distances)
}
```

2.23 This can be done through:

```
f <- function(x) {
  mean(x^2) - mean(x)^2
}
f(1:10)


## [1] 8.25
```

2.24 A simple answer is just given by:

```
iseven <- function(x) x %%2 == 0
```

Then `isodd` would be:

```
isodd <- function(x) x%%2 == 1
```

The following implementation ensures integers are used, and adds names:

```
iseven <- function(x) {
  x <- as.integer(x)
  ans <- x %% 2 == 0
  setNames(ans, x)                      # add names
}
iseven(1:10)


##     1     2     3     4     5     6     7     8     9    10
## FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE
```

Restricting a function to handle only integer inputs can be achieved by using generic functions, such as described in Appendix **??**.

2.25 A simple implementation looks like this. One could improve it by only looking at integer factors less or equal the square-root of x.

```
isprime <- function(x){
  !any(x %% 2:(x-1) == 0)
}
```

Though this isn't a terribly efficient means to generate a list of primes, it can be used to check if one number is prime.

2.26 The package containing the data set is no longer maintained, so this problem becomes quite hard to do! Here we copy the data:

```
time <- c(169, 125, 210, 118, 117, 135, 128, 120, 122, 164, 174, 155, 120, 159,
          121, 144, 129, 136, 124, 138, 195, 141, 156, 179, 109, 112, 167, 113,
          133, 153, 141, 150, 126, 202, 165, 139, 164, 162, 171, 154, 147, 148,
          137, 144, 139, 159, 128, 181, 181, 146, 161, 157, 130, 121, 122, 135,
          150, 151, 177, 168, 180, 136, 230, 153, 275, 204, 245, 177, 187, 237,
          119, 166, 205, 167, 153, 204, 156, 303, 158, 163, 155, 80, 303, 165,
          240, 130, 190, 62, 185, 286, 167, 148, 121, 140, 124, 213, 232, 102,
          106, 177, 160, 241, 166, 145, 195, 270, 188, 253, 162, 175, 191, 495,
          194)
album <- rep(c("BBC_tapes", "Rubber_Soul", "Revolver", "Magical Mystery Tour",
               "Seargent Peper", "The White album"),
             c(31, 11, 14, 14,13,30))
beatles <- data.frame(time=time, album=album)
```

We first convert time to minutes, then compute:

```
lengths <- beatles$time / 60
c(mean=mean(lengths), median=median(lengths),
  longest=max(lengths), shortest=min(lengths))
```

```
##     mean   median  longest shortest
## 2.773009 2.616667 8.250000 1.033333
```

2.27 We need to take a weighted mean, which we do as follows:

```
nk <- ChestSizes$count
yk <- ChestSizes$chest
n <- sum(nk)
wk <- nk/n
sum(wk * yk)
```

```
## [1] 39.83182
```

2.28 We have

```
x <- c(80,82,88,91,91,95,95,97,98,101,106,106,109,110,111)
median(x)


## [1] 97
```

2.29 The `LearnEDA` package is no longer available. The data for `farms` is reproduced with:

```
state <- c("Al", "Als", "Ar", "Ark", "Ca", "Col", "Conn", "De", "Fl", "Ge",
           "Ha", "Id", "Ill", "Ind", "Io", "Kan", "Ken", "Lou", "Ma", "Mary",
           "Mass", "Mi", "Minn", "Miss", "Misso", "Mon", "Neb", "Nev", "NH",
           "NJ", "NM", "NY", "NC", "ND", "Oh", "Ok", "Or", "PA", "RI", "SC",
           "SD", "Te", "Tex", "Ut", "Ver", "Vir", "Wa", "WV", "Wi", "Wy")
count <- c(48, 1, 8, 49, 89, 29, 4, 3, 45, 50, 6, 25, 79, 65, 98, 65, 91, 30,
           7, 12, 6, 53, 81, 43, 110, 28, 55, 3, 3, 10, 16, 39, 58, 31, 80,
           84, 41, 59, 1, 25, 33, 91, 227, 16, 7, 50, 40, 21, 78, 9)
farms <- data.frame(state=state, count=count)
```

The stem and leaf plot is produced by:

```
stem(farms$count)


##
##   The decimal point is 1 digit(s) to the right of the |
##
##    0 | 1133346677890266
##    2 | 155890139
##    4 | 013589003589
##    6 | 5589
##    8 | 0149118
##   10 | 0
##   12 |
##   14 |
##   16 |
##   18 |
##   20 |
##   22 | 7
```

It is hard to gauge the influence of the outlier, but otherwise, the balance point is likely in the stem labeled 4, or 4000 farms. A check shows it is 44.04.

2.30 The value is `2.3e-4` or $2.3 \cdot 10^{-4}$:

```
2.3 * 10^(-4)


## [1] 0.00023
```

2.31 For `firstchi` this is done as follows:

```
hist(firstchi)                # looks like 25 or so
mean(firstchi)                # we were pretty close ...


## [1] 23.97701
```

2.32 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 This is done as follows:

```
hist(normtemp$temperature)    # looks like its 98.2 -- not 98.6
mean(normtemp$temperature)


## [1] 98.24923
```

2.34 The graphics can be produced with these commands:

```
require(MASS)
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=mean(DDT), sd=sd(DDT))


##      mean        sd
## 3.3280000 0.4371531
```

2.35 The `hist` function needs the data to be in a data vector, not tabulated. We pad it out using `rep`, then plot. The histogram is very symmetric.

```
x <- rep(ChestSizes$chest, ChestSizes$count)
hist(x)
```

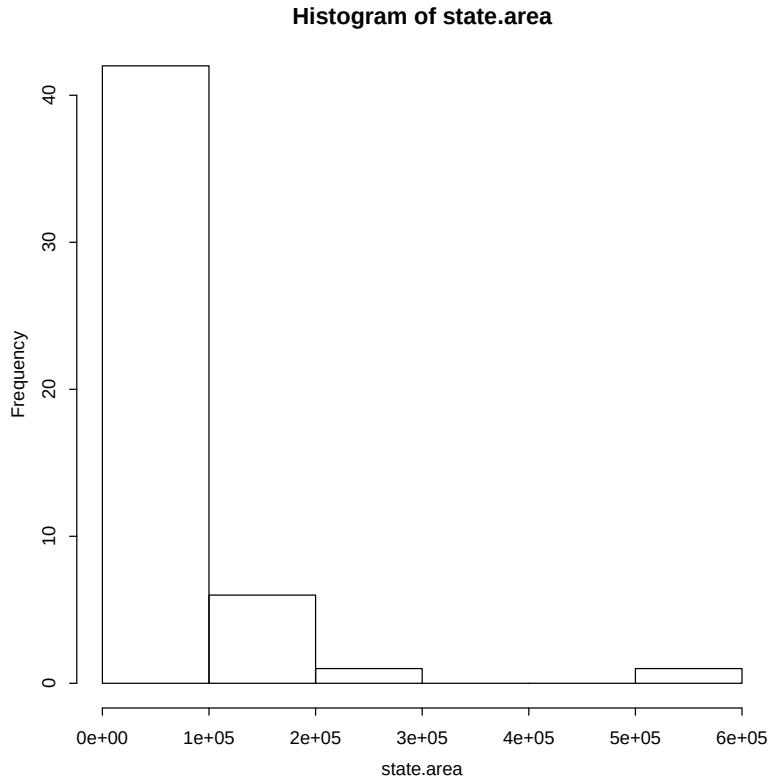
2.36 The histogram has a rather wide range (about 3 times from smallest to largest. Some year had over 93 feet of snow fall!

2.37 First assign names. Then you can access the entries using the respective 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
```

To see that Alaska is the big state, we could look at this histogram then query:

```
hist(state.area)                          # 50,000 cuts of last case
state.area[state.area > 5e5]

##     AK
## 589757
```



2.38 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.

2.39 The definition of the median is incorrect. Can you think of a shape for a distribution when this is actually okay?

2.40 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.

2.41 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
```

2.42 These values are found with:

```
sum(rivers < 500) / length(rivers)

## [1] 0.5815603

sum(rivers < mean(rivers)) / length(rivers)

## [1] 0.6666667

quantile(rivers, 0.75)

## 75%
## 680
```

2.43 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                  # easier to use
range(times)                            # looks like minutes

## [1] 147.3333 566.7833
```

```
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%
## 208.695 233.775

quantile(times,c(.90))                  # 5:31

##    90%
## 331.75
```

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.

2.44 Use the functions:

```
mean(rivers)

## [1] 591.1844

median(rivers)

## [1] 425

mean(rivers, trim=.25)

## [1] 449.9155
```

Yes, the data is skewed to the right.

2.45 We see

```
stem(islands)                              # quite skewed


##
##   The decimal point is 3 digit(s) to the right of the |
##
##    0 | 000000000000000000000000000000111111222338
##    2 | 07
##    4 | 5
##    6 | 8
##    8 | 4
##   10 | 5
##   12 |
##   14 |
##   16 | 0


c(mean=mean(islands),
  median=median(islands),
  trimmed=mean(islands,trim=0.25))


##       mean     median    trimmed
## 1252.72917   41.00000   51.08333
```

The data set is quite skewed due to the seven continents. We wouldn't expect the mean and median to agree, but note that after trimming the mean and median are similar.

2.46 We can find the z-score for Barry Bonds using the name as follows:

```
(OBP[bondsba01]- mean(OBP)) / sd(OBP)


## bondsba01
##  5.990192
```

This is a decidedly "non-normal" data set, as we wouldn't expect z-scores larger than 3 in that case.

2.47 Using `scale` gives us the z-scores.

```
z <- scale(x)[,1]                          # matrix notation
mean(z)                                    # basically 0
```

```
## [1] -1.340544e-17

sd(z)

## [1] 1
```

Alternatively, we could have found the z-scores directly with `(x - mean(x))/sd(x)`.

2.48 No, as the data is skewed heavily to the right, the standard deviation is quite different:

```
c(mad=mad(exec.pay), IQR=IQR(exec.pay), sd=sd(exec.pay))

##      mad      IQR       sd
##  20.7564  27.5000 207.0435
```

2.49 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   166300   175000 25000000

sum(amt < mean(amt))/length(amt) * 100

## [1] 74.90069
```

2.50 The value is relatively close to 1, which is the value for exponentially distributed data:

```
sd(rivers) / mean(rivers)

## [1] 0.8353922
```

2.51 Yes, fairly close:

```
ia_times <- diff(babyboom$running.time)
sd(ia_times) / mean(ia_times)

## [1] 0.8889017
```

2.52 The skew of `wt` is negative indicating a slight left skew. The skew of the inter-arrival times is twice as much and to the right.

```
skew(babyboom$wt)

## [1] -1.078636

ia_times <- diff(babyboom$running.time)
skew(ia_times)

## [1] 1.829281
```

Inter-arrival times are typically exponentially distributed. As such, they should have a coefficient of variation that is nearly 1.

2.53 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.

2.54 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.

2.55 The data is tabulated, so we first create a data vector through `rep`, then plot.

```
require(HistData)
chest <- rep(ChestSizes$chest, ChestSizes$count)
qqnorm(chest)
```

The steps are due to truncation in the measurement. Jittering can smooth this out (try `qqnorm(jitter(chest,3))`). This data hews closely to a line, and was used by Quetelet in 1846 to demonstrate "normally-distributed" data.

2.56 The graphic is made with `qqnorm(Michelson$velocity)`. It falls fairly close to a straight line.

2.57 The histograms can be made in a manner similar to this:

```
hist(cfb$AGE)
```

After making the graphics, 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).

2.58 The histogram (`hist(brightness)`) 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.)

2.59 The `Price` variable is:

```
skew(Cars93$Price) > skew(Cars93$MPG.highway)


## [1] TRUE
```

(Both are skewed right.)

2.60 This can be done as follows (using a different name from the built-in `mode` function):

```
Mode <- function(x) {
  tbl <- table(x)
  ind <- which(tbl == max(tbl))
```

```
  vals <- names(ind)
  as(vals, class(x)[1])                    # unnecessary!
}
```

Outside of the last line, this is a simple translation of the example give. The last line is not necessary. It simply generalizes the call to `as.numeric` in the example by coercing the output to the class of the input variable.

2.61 From this command we see the answer is 1001-2000 dollars:

```
bumps <- cut(bumpers, c(0, 1000, 2000, 3000, 4000))
table(bumps)


## bumps
##     (0,1e+03] (1e+03,2e+03] (2e+03,3e+03] (3e+03,4e+03]
##             2             8             7             6
```

2.62 The output of `summary` is a table:

```
summary(Cars93$Cylinders)


##      3      4      5      6      8 rotary
##      3     49      2     31      7      1
```

This seems a good choice—factors are used to categorize values and a table of counts is a useful summary.

2.63 Applying the idiom to `lorem` we have:

```
chars <- unlist(strsplit(lorem, split=""))
table(chars)


## chars
##  \n       ,   ;   .   a   A   b   c   C   d   D   e   E   f   F
##  10 589  48   1  73 251   3  38 156   5 102   8 370   3  22   2
##   g   h   i   I   j   l   L   m   M   n   N   o   p   P   q   Q
##  45  17 343   8   4 220   3 142   7 211  13 174  80   6  30   2
##   r   s   S   t   u   U   v   V   x
## 183 272   7 289 289   3  49   5   3
```

Scanning we see that `e` is the most common. To avoid scanning, the function `sort` can be called on the output of `table`:

```
sort(table(chars))
```

```
## chars
##   ;   F   Q   A   E   L   U   x   j   C   V   P   M   S   D   I
##   1   2   2   3   3   3   3   3   4   5   5   6   7   7   8   8
##  \n   N   h   f   q   b   g   ,   v   .   p   d   m   c   o   r
##  10  13  17  22  30  38  45  48  49  73  80 102 142 156 174 183
##   n   l   a   s   t   u   i   e
## 211 220 251 272 289 289 343 370 589
```

2.64 This is done with

```
require(MASS)
dotchart(table(Cars93$Cylinders))
```

```
## Warning in dotchart(table(Cars93$Cylinders)): 'x' is neither a
vector nor a matrix: using as.numeric(x)
```

The graphic shows that 4-cylinder cars were the most popular in 1993. Was this the case in 1974 (cf. `mtcars$cyl`)?

2.68 It contains the days when nothing much happened.

3

Bivariate data

3.1 The levels of the supp variable are:

```
levels(ToothGrowth$supp)


## [1] "OJ" "VC"
```

With this we create two variables:

```
len_oj <- ToothGrowth$len[ToothGrowth$supp == "OJ"]
len_vc <- ToothGrowth$len[ToothGrowth$supp == "VC"]
```

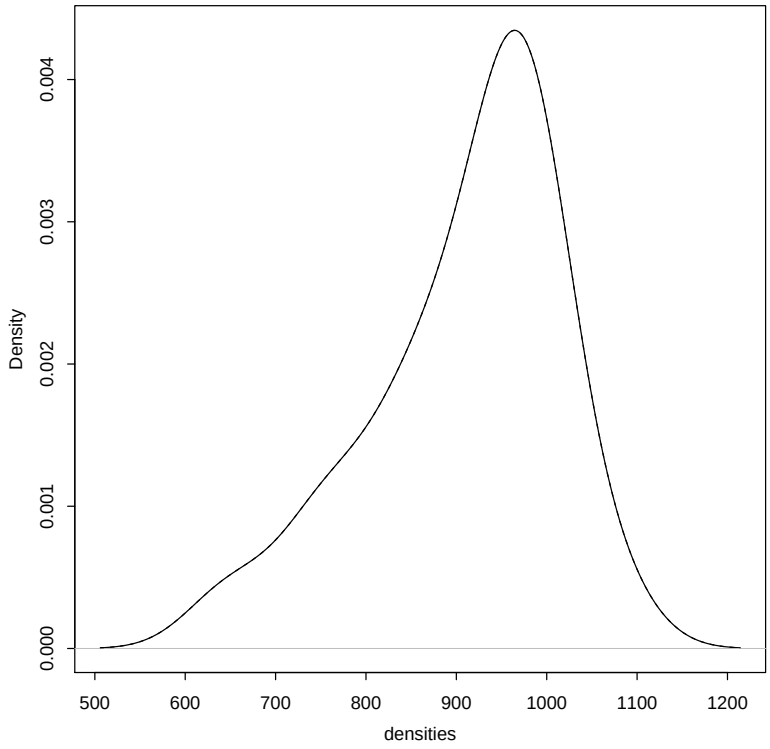
Boxplots are then produced by:

```
boxplot(len_oj, len_vc, names=c("OJ", "VC"))
```

The median for the OJ group is larger, perhaps this is better at encouraging tooth growth.

3.2 This follows the example in the text, save for replacing 4 and 5 with 1 and 2:

```
first <- speed[expt == 1]
second <-  speed[expt == 1]
d_1 <- density(first)
d_2 <- density(second)
xrange <- range(c(d_1$x, d_2$x))
yrange <- range(c(d_1$y, d_2$y))
plot(d_1, xlim=xrange, ylim=yrange, xlab="densities", main="")
lines(d_2, lty=2)
```



3.3 If the quantile-quantile plot follows a straight line this would be the case. However, it clearly isn't:

```
qqplot(rivers, islands)                # not straigh
```

3.4 The list and graphic can be produced with:

```
HW_time <- list(Marsha= c(25, 0, 45, 90, 0),
                Bill=c(30, 30, 30, 30),
                Holly=c(15, 0, 90, 0))
boxplot(HW_time)
```

3.5 The data is already in the format expected, so we have:

```
vals <- data.frame(temp=c(98.2, 98.6, 98.2),
                   pulse=c(96, 56, 76),
                   systolic=c(134, 120, 150),
                   diastolic=c(90, 80, 95))
rownames(vals) <- c("Jackie", "Florence", "Mildred")
```

The last command shows how one can add rownames. (For data frames the rownames must be unique.) The `plot` function produces scatter plots of each pair of variables, not a boxplot of each variable, the way `boxplot` does.

3.6 The output will summarize each experiment by a sample size and average and the overall data set by a sample size and the overall average.

3.7 This can be verified by the command:

```
split(mtcars$mpg, mtcars$cyl)
```

```
## $`4`
##  [1] 22.8 24.4 22.8 32.4 30.4 33.9 21.5 27.3 26.0 30.4 21.4
##
## $`6`
## [1] 21.0 21.0 21.4 18.1 19.2 17.8 19.7
##
## $`8`
##  [1] 18.7 14.3 16.4 17.3 15.2 10.4 10.4 14.7 15.5 15.2 13.3 19.2
## [13] 15.8 15.0
```

We will see in the next chapter how to easily apply a function to each component of the returned list.

3.8 This is done as follows:

```
split(ToothGrowth$len, list(ToothGrowth$supp, ToothGrowth$dose))
```

```
## $OJ.0.5
##  [1] 15.2 21.5 17.6  9.7 14.5 10.0  8.2  9.4 16.5  9.7
##
## $VC.0.5
##  [1]  4.2 11.5  7.3  5.8  6.4 10.0 11.2 11.2  5.2  7.0
##
## $OJ.1
```

```
##  [1] 19.7 23.3 23.6 26.4 20.0 25.2 25.8 21.2 14.5 27.3
##
## $VC.1
##  [1] 16.5 16.5 15.2 17.3 22.5 17.3 13.6 14.5 18.8 15.5
##
## $OJ.2
##  [1] 25.5 26.4 22.4 24.5 24.8 30.9 26.4 27.3 29.4 23.0
##
## $VC.2
##  [1] 23.6 18.5 33.9 25.5 26.4 32.5 26.7 21.5 23.3 29.5
```

3.9 This can be done with

```
time <- reaction.time$time; control <- reaction.time$control
boxplot(time[control == "C"], time[control == "T"])
```

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

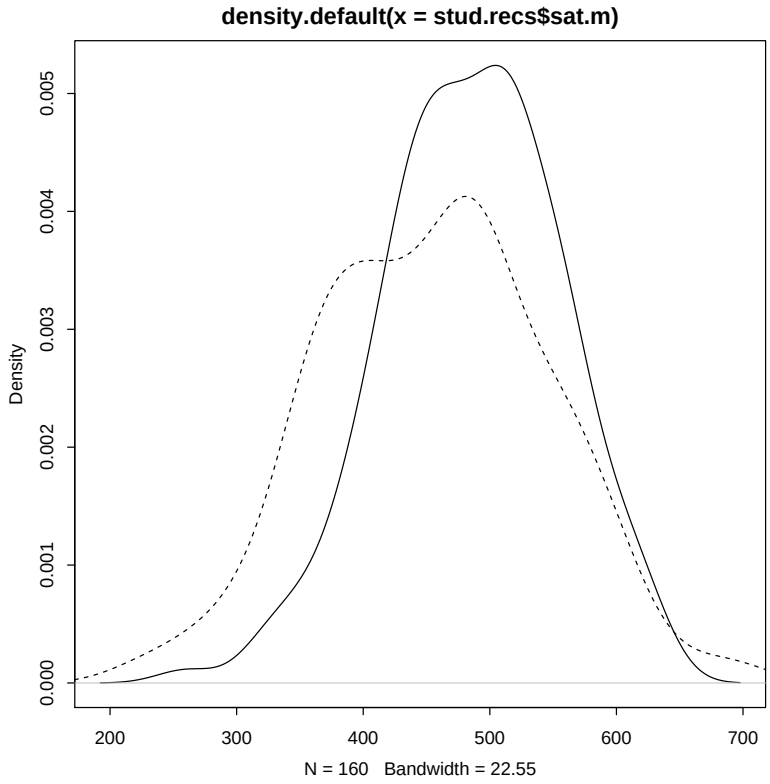
3.10 The boxplots may be produced with a command like:

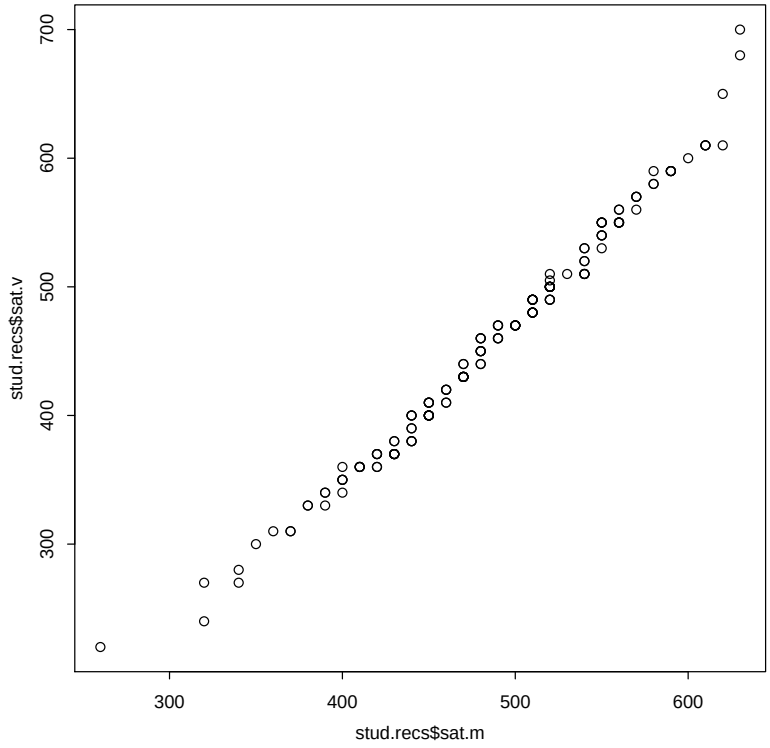
```
boxplot(twins$Foster, twins$Biological, names=c("Foster","Biological"))
```

The centers and spreads appear to be similar.

3.11 If we don't account for potential clipping, the graphs can be produced with

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

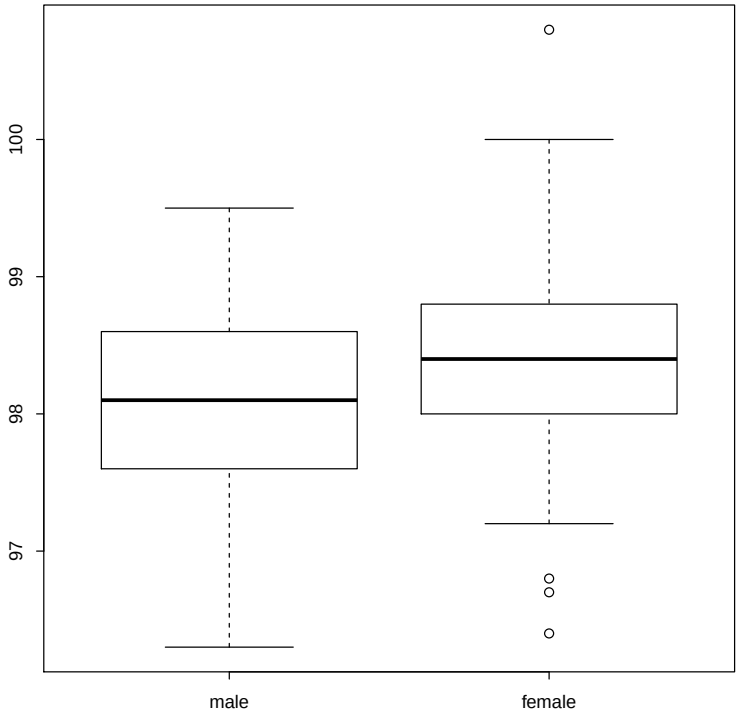

density.default(x = stud.recs$sat.m)
Density
0.000
0.001
0.002
0.003
0.004
0.005
200
300
400
500
600
700
N = 160 Bandwidth = 22.55


stud.recs$sat.v
300
400
500
600
700
300
400
500
600
stud.recs$sat.m

We see from the density plots 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 We can split the data up using the condition `gender==1` as follows:

```
temperature <- normtemp$temperature; gender <- normtemp$gender
male <- temperature[gender == 1]
female <- temperature[gender == 2]
boxplot(list(male=male, female=female))
```



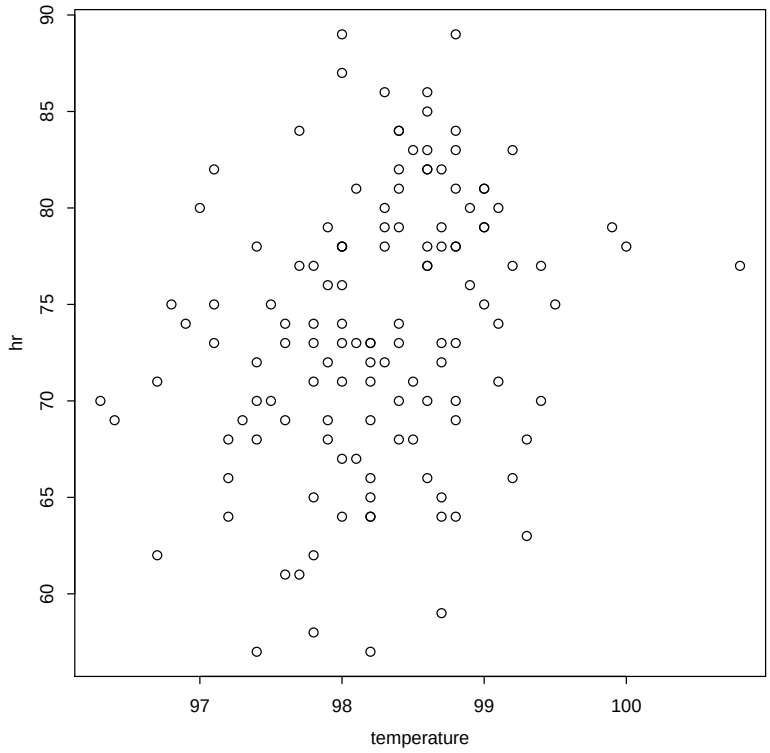
From the graphic it appears that the centers of the distributions are different.

3.13 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. If we don't worry about clipping, a graphic can be produced with:

```
hd_ratio <- homedata$y2000 / homedata$y1970
hist(hd_ratio, probability=TRUE)
lines(density(hd_ratio))
```

3.14 The scatter plot and correlation are found as follows:

```
plot(hr ~ temperature, normtemp)
cor(normtemp$temperature, normtemp$hr)


## [1] 0.2536564
```

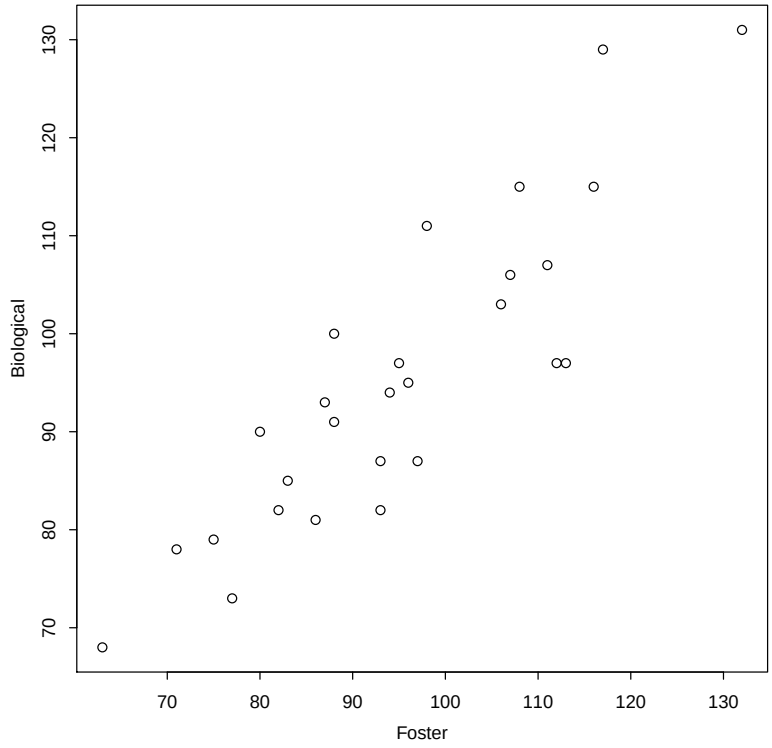


Although there appears to be a trend between the two, the correlation is not large.

3.15 The scatter plot and correlation are found with the commands plot(body.fat BMI, data=fat) cor(fatBMI, fatbody.fat) @ Except for a few outliers in the `BMI`, the data shows a fairly strong correlation.

3.16 The plot can be made with the commands

```
plot(Biological ~ Foster, data=twins)
```



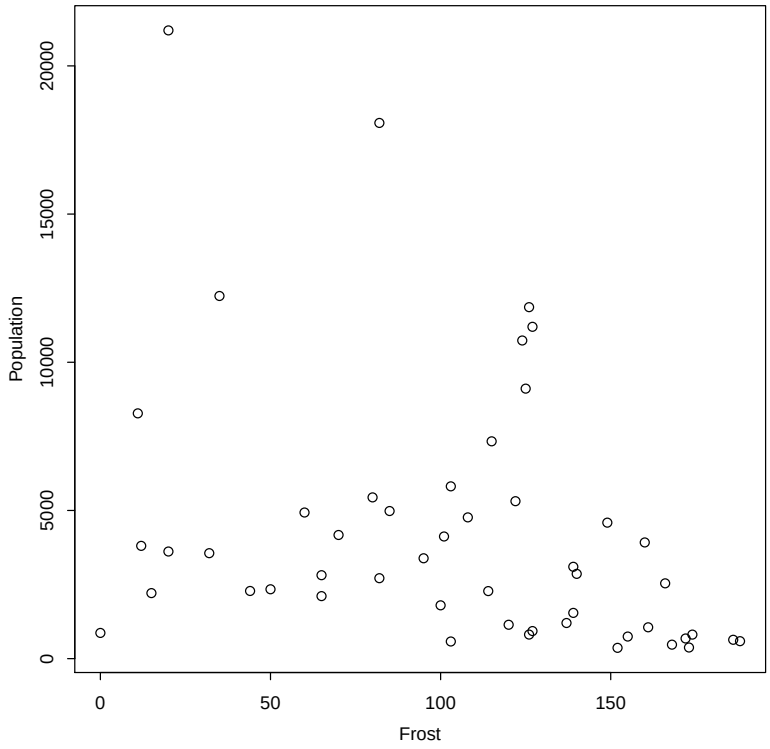
Based on this, a guess of 0.8 for the Pearson coefficient and a similar value for the Spearman coefficient seem reasonable. A check can be done with

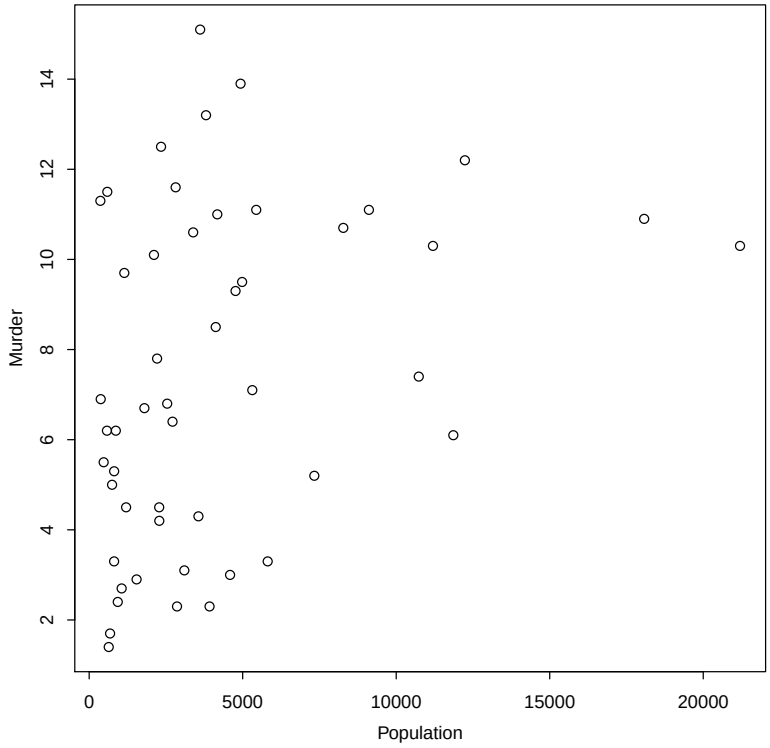
```
cor(twins$Foster, twins$Biological)

## [1] 0.8819877
```

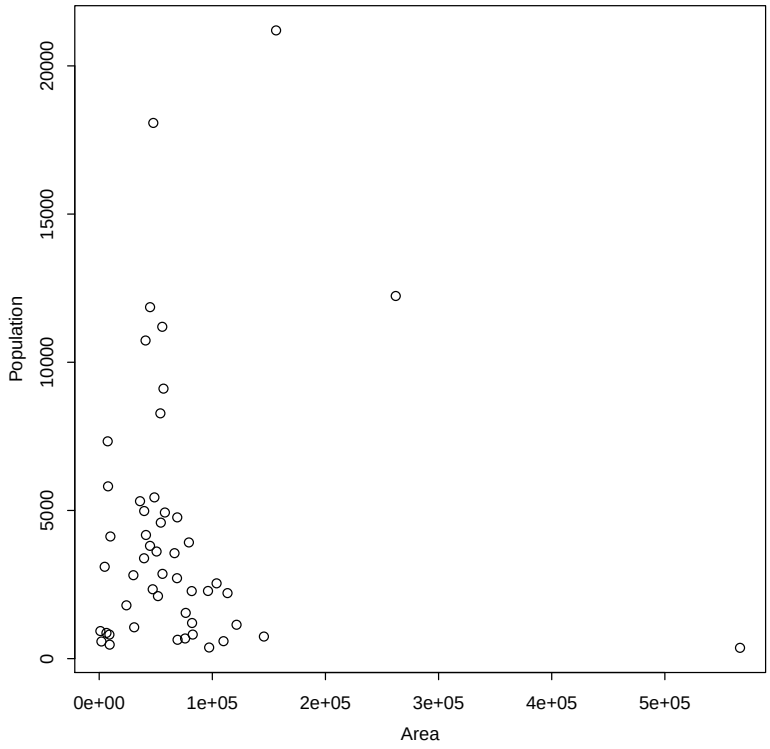
```
cor(twins$Foster, twins$Biological, method="spearman")


## [1] 0.8858324
```

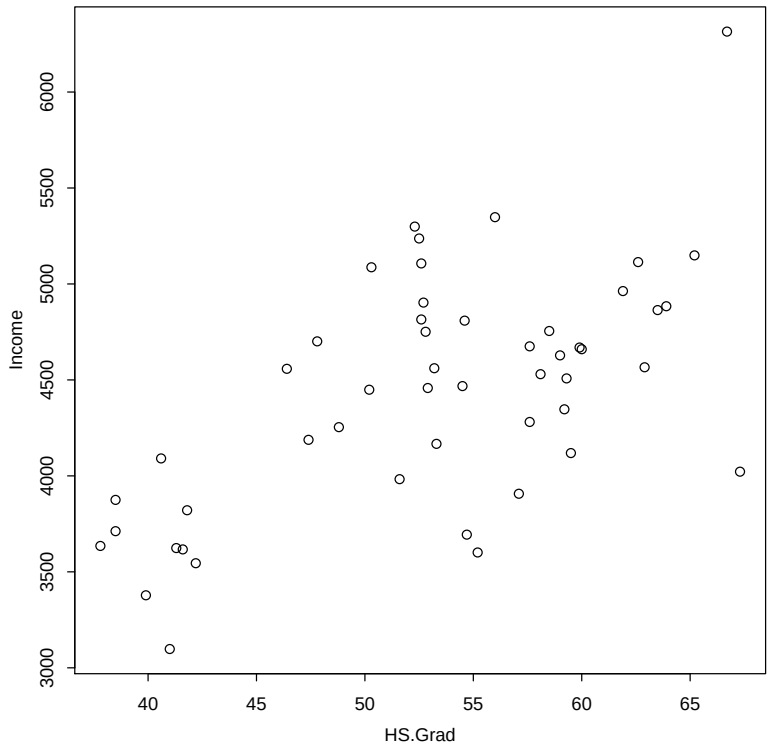
3.17 The scatter plots are produced using the model formula interface with the following commands:

```
plot(Population ~ Frost, data=x77)
plot(Murder ~ Population, data=x77)
plot(Population ~ Area, data=x77)
plot(Income ~ HS.Grad, data=x77)
```


Population
0
5000
10000
15000
20000
0
50
100
150
Frost


Murder
2
4
6
8
10
12
14
0
5000
10000
15000
20000
Population


Population
0
5000
10000
15000
20000
0e+00
1e+05
2e+05
3e+05
4e+05
5e+05
Area


Income
3000
3500
4000
4500
5000
5500
6000
40
45
50
55
60
65
HS.Grad

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.

3.18 We might think, at first, that time would increase dramatically as age does, but the actual correlation is quite low.

```
names(nym.2002)                          # variable names

## [1] "place"  "gender" "age"    "home"   "time"

cor(nym.2002$age,  nym.2002$time)

## [1] 0.1898672
```

3.19 These are the coordinates of the center for each of the fifty states in the United States.

3.20 The correlation is found with

```
cor(batting$HR, batting$SO)

## [1] 0.7084697
```

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

3.22 We fit the model and then make the prediction as follows:

```
res <- lm(abdomen ~ wrist, data = fat)
predict(res, data.frame(wrist=17))

##        1
## 83.75187
```

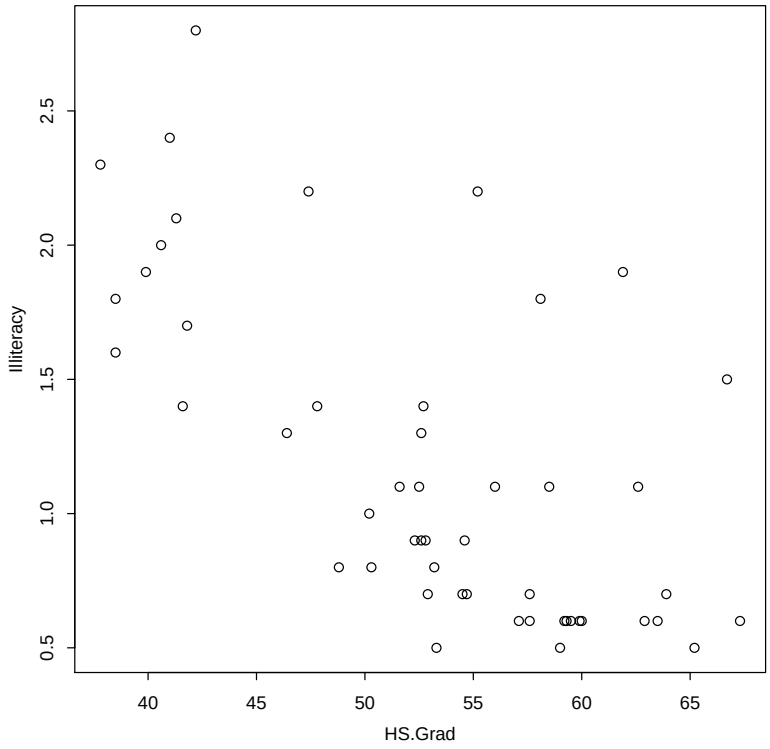
3.23 The `wtloss` data can be plotted as follows:

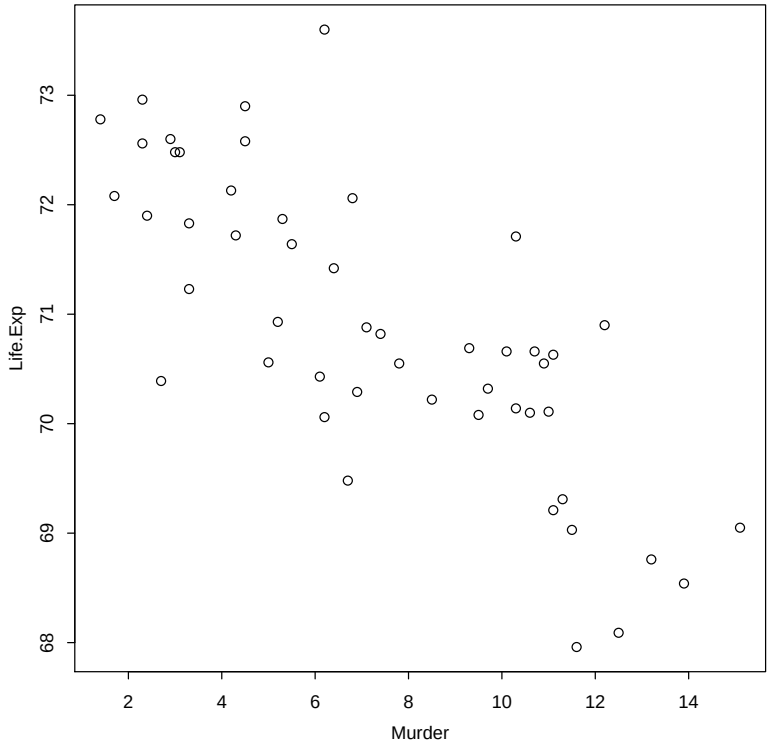
```
cor(wtloss$Weight, wtloss$Days)             # close to -1


## [1] -0.9853149


plot(Weight ~ Days, data=wtloss)
res <- lm(Weight ~ Days, data=wtloss)
abline(res)
plot(residuals(res) ~ Days, data=wtloss)    # A trend
```

We see from the scatter plot 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.

3.24 The plots can be produced with these commands:

```
plot(Illiteracy ~ HS.Grad, data = x77)
plot(Life.Exp ~ Murder, data = x77)
plot(Income ~ Illiteracy, data = x77)
```


Illiteracy
0.5
1.0
1.5
2.0
2.5
40
45
50
55
60
65
HS.Grad


Life.Exp
68
69
70
71
72
73
2
4
6
8
10
12
14
Murder

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.

3.25 The predicted amount and residual can be found using `predict`:

```
res <- lm(RBI ~ HR, data=batting)
predict(res, data.frame(HR=33))
```

```
##        1
## 104.1099
```

```
98 - predict(res, data.frame(HR=33))
```

```
##       1
## -6.1099
```

3.26 The calculations are performed as follows:

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

```
##
## Call:
## lm(formula = Female ~ Male, data = too.young)
##
## Coefficients:
## (Intercept)         Male
##      5.4720       0.5754
```

```
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.

3.27 The regression line and predictions are carried out using these commands:

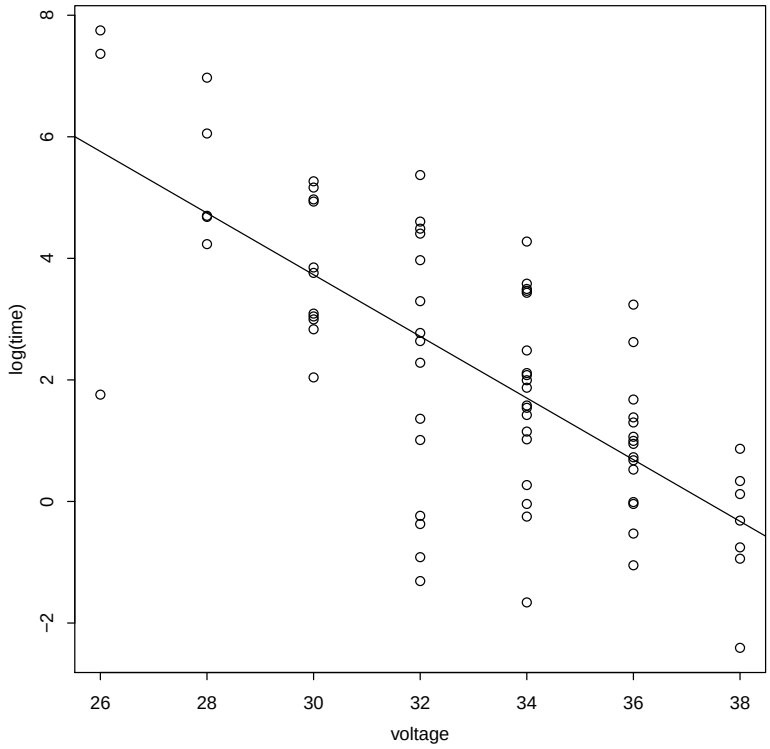
```
plot(price ~ carat, data = diamond, pch=5)
res <- lm(price ~ carat, data = diamond)
abline(res)
predict(res, data.frame(carat=1/3))
```

```
##        1
## 980.7157
```

3.28 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
##     18.9461      -0.5072
```



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

3.29 From the scatter plot 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
##     22999.1       -106.8
```

Then predictions can be made using `predict`:

```
predict(res, newdata=data.frame(temp=210))


##        1
## 580.5888
```

3.31 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] 25.93
```

The barplot of all years 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.

To get the coins by decade, a first attempt would be to try

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

```
##
## (1.92e+03,1.93e+03] (1.93e+03,1.94e+03] (1.94e+03,1.95e+03]
##                   3                   2                   0
## (1.95e+03,1.96e+03] (1.96e+03,1.97e+03] (1.97e+03,1.98e+03]
##                   3                  16                  43
## (1.98e+03,1.99e+03]    (1.99e+03,2e+03]    (2e+03,2.01e+03]
##                  99                 147                  58
```

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.

Finally, the two way table can be made just by passing the data frame to `table`, as in `table(coins)`. Alternatively, one could reference the variables individually, as with `table(coins$year, coins$value)`. Either way, the marginal distributions are certainly not uniform, but there is very little change in the proportions from year to year. A check of the correlation yields:

```
cor(coins$year, coins$value)
```

```
## [1] 0.0595
```

3.32 The table is made with

```
require(MASS)
table(UScereal$shelf, UScereal$mfr)
```

```
##
##      G  K  N  P  Q  R
##   1  6  4  2  2  0  4
##   2  7  7  0  1  3  0
##   3  9 10  1  6  2  1
```

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.

3.33 Producing the graphic with the `shade=TRUE` argument results in the area being drawn in dark blue. The coloring is based on an expectation of no association of the variables. In this case, the size of area for blond females with blue eyes is larger than would be guessed if there was no dependence.

4

Multivariate data

4.1 These can be answered with:

```
require(MASS)
dim(UScereal)                           # rows, columns

## [1] 65 11

length(levels(UScereal$mfr))            # number of manufacturers

## [1] 6

length(levels(UScereal["vitamins"]))    # vitamin categories

## [1] 0

sum(UScereal[, "sugars"] > 10)          # sugar levels above 10

## [1] 39

mean(UScereal[UScereal$fat > 5, "calories"]) # conditional mean

## [1] 291.8

mean(UScereal[UScereal$fat <= 5, "calories"]) # conditional mean

## [1] 144.9

mean(UScereal[UScereal["shelf"] == 2, "calories"])
```

```
## [1] 129.8
```

4.2 We make the list, then answer the questions:

```
l <- lm(mpg ~ wt, data=mtcars)
length(l)                               # no. components
```

```
## [1] 12
```

```
names(l)
```

```
##  [1] "coefficients"  "residuals"     "effects"
##  [4] "rank"          "fitted.values" "assign"
##  [7] "qr"            "df.residual"   "xlevels"
## [10] "call"          "terms"         "model"
```

```
l[["residuals"]]                        # numeric, named data
```

```
##           Mazda RX4       Mazda RX4 Wag          Datsun 710
##            -2.28261            -0.91977            -2.08595
##      Hornet 4 Drive   Hornet Sportabout             Valiant
##             1.29735            -0.20014            -0.69325
##          Duster 360           Merc 240D            Merc 230
##            -3.90536             4.16374             2.34996
##            Merc 280           Merc 280C           Merc 450SE
##             0.29986            -1.10014             0.86687
##          Merc 450SL         Merc 450SLC  Cadillac Fleetwood
##            -0.05025            -1.88302             1.17335
## Lincoln Continental   Chrysler Imperial            Fiat 128
##             2.10329             5.98107             6.87271
##         Honda Civic      Toyota Corolla       Toyota Corona
##             1.74620             6.42198            -2.61100
##    Dodge Challenger         AMC Javelin          Camaro Z28
##            -2.97259            -3.72687            -3.46236
##    Pontiac Firebird           Fiat X1-9       Porsche 914-2
##             2.46437             0.35643             0.15204
##        Lotus Europa      Ford Pantera L        Ferrari Dino
##             1.20106            -4.54315            -2.78094
##       Maserati Bora          Volvo 142E
##            -3.20536            -1.02750
```

4.3 The values `mtcars$cyl`, `mtcars[["car""]]`, and `mtcars[,"car"]` return a data vector, whereas `mtcars["car"]` and `mtcars[,"car", drop=FALSE]` return a data frame with one column.

4.4 Only `mtcars$cy` and `mtcars[["cy", exact=FALSE]]`? will match. The call `mtcars$c` is ambiguous with the `carb` variable, and `[` access doesn't employ partial matching.

4.5 They all will, though the usage may vary. The `colnames` function is used for matrix-like objects (there are also functions `rownames<-` and `row.names<-`—just for data frames). The `names<-` and `dimnames<-` functions come from their use on lists.

```
d <- data.frame(a=1, b="two")
names(d) <- c("A", "B")
dimnames(d) <- list("1", c("eh", "bee"))
colnames(d) <- c("EH", "BEE")
d <- setNames(d, c("ahh", "buh"))
```

4.6 We use `with`:

```
with(fat, mean(neck) / mean(wrist))

## [1] 2.084

with(fat, mean(neck / wrist))

## [1] 2.084
```

4.7 We simply chain the filters together with `&`:

```
subset(Cars93, Origin == "non-USA" & Cylinders == 4 & Max.Price <= 15)

##    Manufacturer   Model   Type Min.Price Price Max.Price
## 40          Geo   Storm Sporty      11.5  12.5      13.5
## 44      Hyundai   Excel  Small       6.8   8.0       9.2
## 45      Hyundai Elantra  Small       9.0  10.0      11.0
## 46      Hyundai  Scoupe Sporty       9.1  10.0      11.0
## 53        Mazda     323  Small       7.4   8.3       9.1
```

```
## 54        Mazda Protege  Small      10.9  11.6      12.3
## 62   Mitsubishi  Mirage  Small       7.7  10.3      12.9
## 64       Nissan  Sentra  Small       8.7  11.8      14.9
## 81       Subaru  Loyale  Small      10.5  10.9      11.3
## 84       Toyota  Tercel  Small       7.8   9.8      11.8
## 88   Volkswagen     Fox  Small       8.7   9.1       9.5
##    MPG.city MPG.highway     AirBags DriveTrain Cylinders
## 40       30          36 Driver only      Front         4
## 44       29          33        None      Front         4
## 45       22          29        None      Front         4
## 46       26          34        None      Front         4
## 53       29          37        None      Front         4
## 54       28          36        None      Front         4
## 62       29          33        None      Front         4
## 64       29          33 Driver only      Front         4
## 81       25          30        None        4WD         4
## 84       32          37 Driver only      Front         4
## 88       25          33        None      Front         4
##    EngineSize Horsepower  RPM Rev.per.mile Man.trans.avail
## 40        1.6         90 5400         3250             Yes
## 44        1.5         81 5500         2710             Yes
## 45        1.8        124 6000         2745             Yes
## 46        1.5         92 5550         2540             Yes
## 53        1.6         82 5000         2370             Yes
## 54        1.8        103 5500         2220             Yes
## 62        1.5         92 6000         2505             Yes
## 64        1.6        110 6000         2435             Yes
## 81        1.8         90 5200         3375             Yes
## 84        1.5         82 5200         3505             Yes
## 88        1.8         81 5500         2550             Yes
##    Fuel.tank.capacity Passengers Length Wheelbase Width
## 40               12.4          4    164        97    67
## 44               11.9          5    168        94    63
## 45               13.7          5    172        98    66
## 46               11.9          4    166        94    64
## 53               13.2          4    164        97    66
## 54               14.5          5    172        98    66
## 62               13.2          5    172        98    67
## 64               13.2          5    170        96    66
## 81               15.9          5    175        97    65
## 84               11.9          5    162        94    65
## 88               12.4          4    163        93    63
##    Turn.circle Rear.seat.room Luggage.room Weight  Origin
## 40          37           24.5           11   2475 non-USA
## 44          35           26.0           11   2345 non-USA
```

```
## 45          36           28.0           12   2620 non-USA
## 46          34           23.5            9   2285 non-USA
## 53          34           27.0           16   2325 non-USA
## 54          36           26.5           13   2440 non-USA
## 62          36           26.0           11   2295 non-USA
## 64          33           26.0           12   2545 non-USA
## 81          35           27.5           15   2490 non-USA
## 84          36           24.0           11   2055 non-USA
## 88          34           26.0           10   2240 non-USA
##                 Make
## 40         Geo Storm
## 44     Hyundai Excel
## 45   Hyundai Elantra
## 46    Hyundai Scoupe
## 53         Mazda 323
## 54     Mazda Protege
## 62 Mitsubishi Mirage
## 64     Nissan Sentra
## 81     Subaru Loyale
## 84     Toyota Tercel
## 88    Volkswagen Fox
```

4.8 The variables `Alcohol.consumption`, `Working.hours`, and `Military.spending`.

4.9 The package containing the data set is no longer maintained, so this problem becomes quite hard to do! Here we reproduce the data, then proceed:

```
time <- c(169, 125, 210, 118, 117, 135, 128, 120, 122, 164, 174, 155, 120, 159,
          121, 144, 129, 136, 124, 138, 195, 141, 156, 179, 109, 112, 167, 113,
          133, 153, 141, 150, 126, 202, 165, 139, 164, 162, 171, 154, 147, 148,
          137, 144, 139, 159, 128, 181, 181, 146, 161, 157, 130, 121, 122, 135,
          150, 151, 177, 168, 180, 136, 230, 153, 275, 204, 245, 177, 187, 237,
          119, 166, 205, 167, 153, 204, 156, 303, 158, 163, 155, 80, 303, 165,
          240, 130, 190, 62, 185, 286, 167, 148, 121, 140, 124, 213, 232, 102,
          106, 177, 160, 241, 166, 145, 195, 270, 188, 253, 162, 175, 191, 495,
          194)
album <- rep(c("BBC_tapes", "Rubber_Soul", "Revolver", "Magical Mystery Tour",
               "Seargent Peper", "The White album"),
             c(31, 11, 14, 14,13,30))
beatles <- data.frame(time=time, album=album)
#
l <- with(beatles, split(time,  album))
```

```
sapply(l, length)

##            BBC_tapes Magical Mystery Tour             Revolver
##                   31                   14                   14
##          Rubber_Soul       Seargent Peper      The White album
##                   11                   13                   30
```

4.10 This is an easy task for `sapply` or `Vectorize`:

```
sapply(mtcars, sd)

##         mpg         cyl        disp          hp        drat
##   6.0269481   1.7859216 123.9386938  68.5628685   0.5346787
##          wt        qsec          vs          am        gear
##   0.9784574   1.7869432   0.5040161   0.4989909   0.7378041
##        carb
##   1.6152000

Vectorize(sd)(mtcars)

##         mpg         cyl        disp          hp        drat
##   6.0269481   1.7859216 123.9386938  68.5628685   0.5346787
##          wt        qsec          vs          am        gear
##   0.9784574   1.7869432   0.5040161   0.4989909   0.7378041
##        carb
##   1.6152000
```

4.11 First we filter our non-numeric rows and then apply the `sd` function:

```
sapply(Filter(is.numeric, Cars93), sd)

##          Min.Price              Price          Max.Price
##           8.746029           9.659430          11.030457
##           MPG.city        MPG.highway         EngineSize
##           5.619812           5.331726           1.037363
##         Horsepower                RPM       Rev.per.mile
##          52.374410         596.731690         496.506525
## Fuel.tank.capacity         Passengers             Length
##           3.279370           1.038979          14.602382
```

```
##          Wheelbase              Width        Turn.circle
##           6.819674           3.778986           3.223265
##     Rear.seat.room       Luggage.room             Weight
##                 NA                 NA         589.896510
```

To pass in `na.rm=TRUE` to `sd`, we can add it as an argument to `sd` with:

```
sapply(Filter(is.numeric, Cars93), sd, na.rm=TRUE)
```

```
##          Min.Price              Price          Max.Price
##           8.746029           9.659430          11.030457
##           MPG.city        MPG.highway         EngineSize
##           5.619812           5.331726           1.037363
##         Horsepower                RPM       Rev.per.mile
##          52.374410         596.731690         496.506525
## Fuel.tank.capacity         Passengers             Length
##           3.279370           1.038979          14.602382
##          Wheelbase              Width        Turn.circle
##           6.819674           3.778986           3.223265
##     Rear.seat.room       Luggage.room             Weight
##           2.989072           2.997967         589.896510
```

4.12 Using `Filter` is very quick, `Filter(is.factor, DF)` or `Filter(is.integer, DF)`. To keeps those columns that are either a factor or integer data, one can write a special function, e.g, `function(x) is.factor(x) | is.integer(x)`.

4.13 This can be done by splitting on the team ID variable and then applying a function to each resulting data frame.

```
teams <- split(batting, batting$teamID)
team_avg <- sapply(teams, function(DF) with(DF, sum(H) / sum(AB)))
sort(team_avg)
```

```
##       BAL       DET       PIT       CLE       CHN       TBA
## 0.2485346 0.2497458 0.2518990 0.2545050 0.2546267 0.2552230
##       KCA       OAK       SDN       CIN       TOR       NYN
## 0.2608696 0.2629318 0.2629957 0.2631579 0.2637260 0.2642434
##       PHI       MIL       ATL       CHA       HOU       FLO
## 0.2673943 0.2674995 0.2682297 0.2691599 0.2709572 0.2716430
##       SFN       LAN       MIN       TEX       MON       SLN
```

```
## 0.2724966 0.2725216 0.2730257 0.2732106 0.2737418 0.2748584
##       SEA       COL       NYA       BOS       ARI       ANA
## 0.2759270 0.2770161 0.2773313 0.2778684 0.2787719 0.2848981
```

4.14 A player will have played for more than one team if they have more than one `teamID` values:

```
players <- split(batting, batting$playerID)
traded_players <- Filter(function(x) nrow(x) > 1, players)
names(traded_players)
```

```
##  [1] "alomasa02" "branyru01" "durhara01" "encarju01" "floydcl01"
##  [6] "giambje01" "guilljo01" "hollato01" "jimenda01" "kaplega01"
## [11] "loftoke01" "maciajo01" "mondera01" "paytoja01" "penaca01"
## [16] "polanpl01" "rolensc01" "stevele01" "trubych01"
```

4.15 Each component is a data vector, we simply need a function which takes the first element:

```
sapply(d, function(x) x[1])
```

```
##   1,2   3,4   5,6
## "1,2" "3,4" "5,6"
```

For lists, this is a fairly common operation. This function will "pluck" out the *i*th entry from each component (simplifying the anonymous function used above):

```
pluck <- function(lst, i=1) sapply(lst, "[", i=i)
```

4.16 A variable has an `NA` value if `any(is.na(x))` is TRUE. This allows us to perform this task with `Filter`:

```
d <- data.frame(a=1:3, b=c(1, NA, 3), c=c("one", "two", NA))
Filter(function(x) !any(is.na(x)), d)
```

```
##   a
```

```
## 1 1
## 2 2
## 3 3
```

4.17 As `class` can sometimes return more than one element, we use just the first below.

```
f <- function(nm, x) sprintf("Variable %s has class %s", nm, class(x)[1])
our_func <- function(DF) mapply(f, names(DF), DF)
our_func(mtcars[1:3])
```

```
##                               mpg
##  "Variable mpg has class numeric"
##                               cyl
##  "Variable cyl has class numeric"
##                              disp
## "Variable disp has class numeric"
```

4.18 One can work with indices, as with:

```
sapply(seq_along(fruits), function(i) paste(fruits[i], "are fruit number", i))
```

```
## [1] "Bananas are fruit number 1"   "Oranges are fruit number 2"
## [3] "Avocados are fruit number 3"  "Celeries? are fruit number 4"
```

Alternatively, one can use `mapply`:

```
mapply(paste, fruits, "are fruit number", seq_along(fruits))
```

```
##                        Bananas                        Oranges
##   "Bananas are fruit number 1"   "Oranges are fruit number 2"
##                        Avocados                       Celeries?
##  "Avocados are fruit number 3" "Celeries? are fruit number 4"
```

4.19 Subtraction is done left to right, as in $(2-3)-4$, and powers are done right to left: $2^{(3^4}$.

5

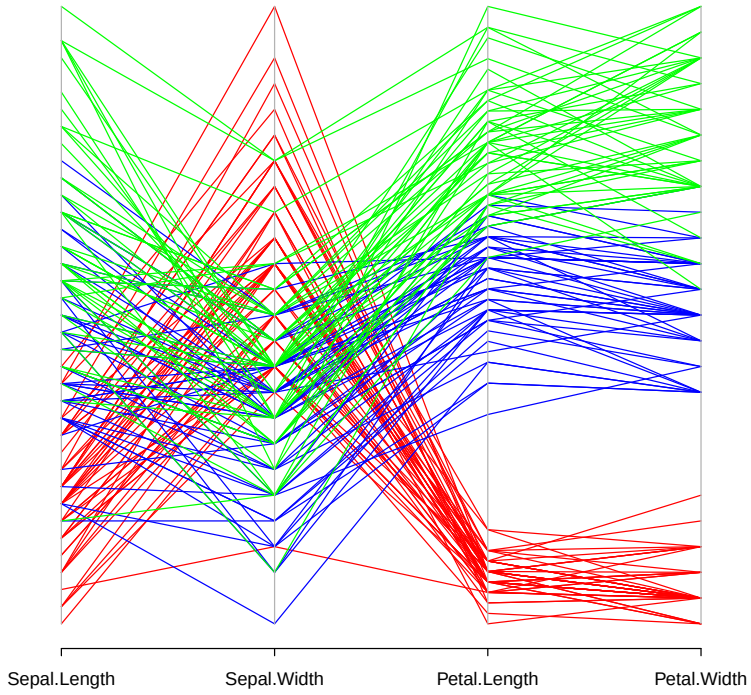
Multivariate graphics

5.1 The graphic can be produced with

```
plot(calories ~ sugars, pch=shelf, data=UScereal)
legend(0, 400, legend=paste("shelf", 1:3),  pch=1:3)
```

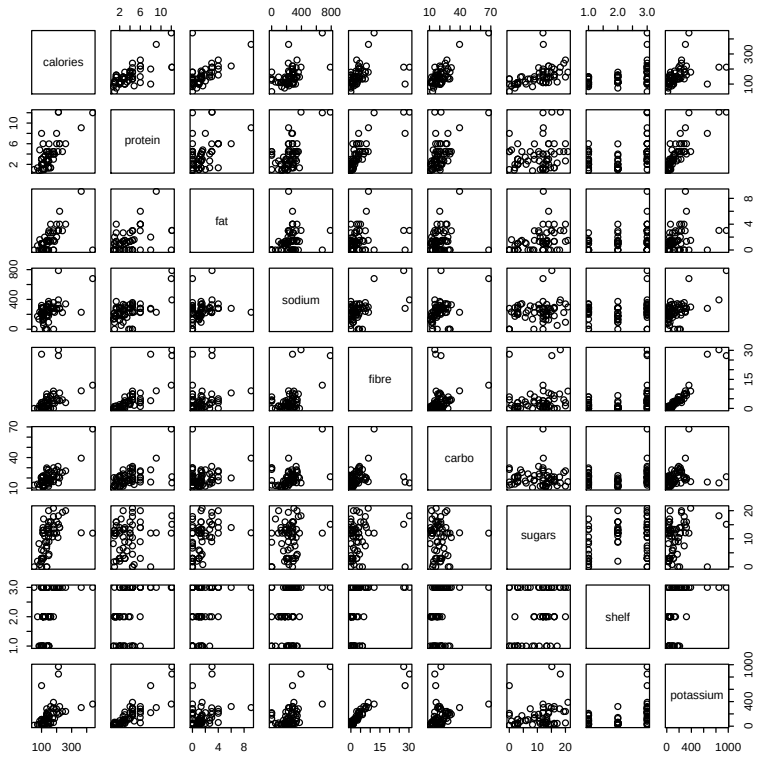
Outside of a few exceptions, the higher calorie cereals are on the third shelf.

5.2
```
values <- Filter(is.numeric, iris)
cols <- c("red", "blue", "green")
cols <- cols[as.numeric(iris$Species)]
parcoord(values, col=cols)
```



The red species (*setosa*) is quite different.

5.3
```
require(MASS)
d <- Filter(is.numeric, UScereal)
pairs(d)
```



The correlation between fat and calories looks stronger:

```
with(UScereal, c(calories=cor(fat, calories), sugars=cor(fat, sugars)))
```

```
##  calories    sugars
## 0.5901757 0.4156740
```

5.4 The graphic is made with

```
plot(HR ~ H, batting, cex=sqrt(SO)/10)
```

The size of the points seems to track the number of home runs hit.

5.5 The graphic can be produced with:

```
require(MASS)
require(lattice)
dotplot(Expt ~ Speed, data=michelson)
```

The first experiment has more spread.

5.6 We can compute the batting average in the formula:

```
require(lattice)
bwplot(teamID ~ I(H/AB), batting)
```

The formula in `bwplot` actually doesn't need the `I` here, we add it. (It would had we added variables, as the `+` symbol has an overloaded meaning.)

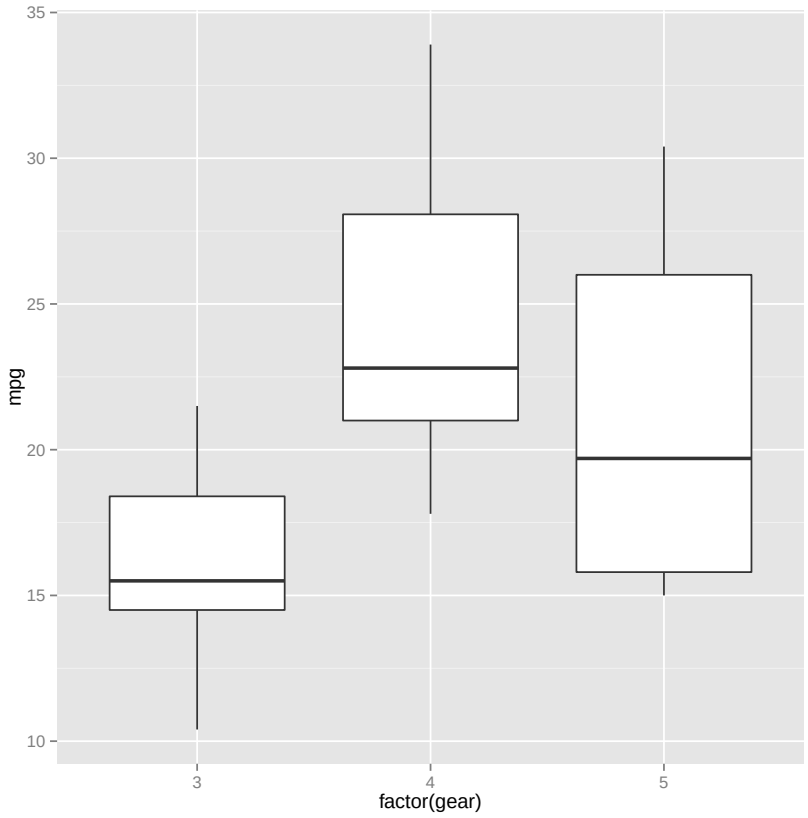
In the graphic, we see that Anaheim (ANA) has the largest median, which we could have found by sorting this output:

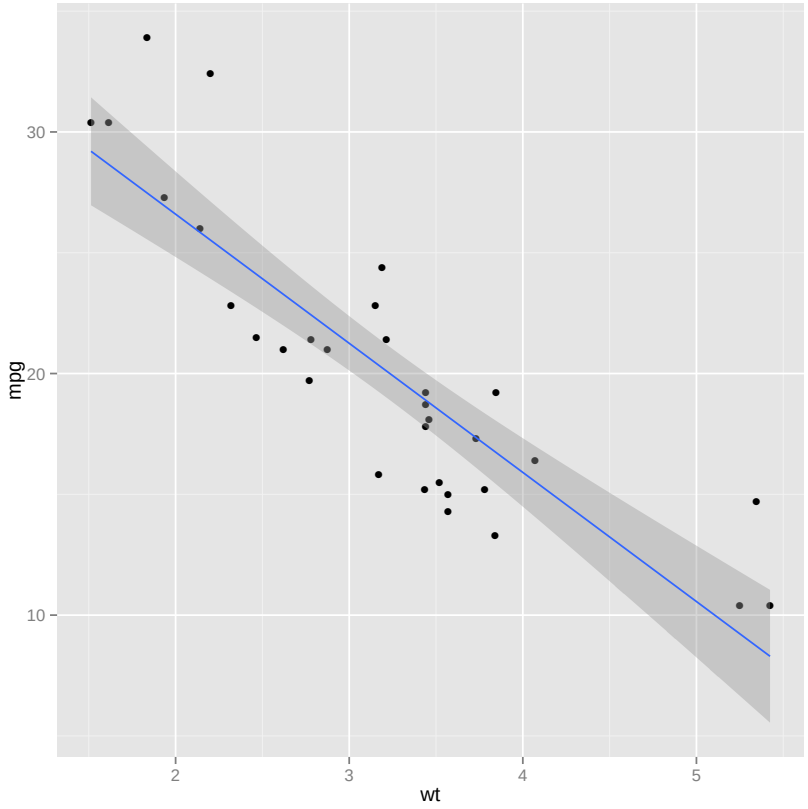
```
sapply(split(batting, batting$teamID),
       function(DF) with(DF, median(H/AB)))
```

```
##       ANA       ARI       ATL       BAL       BOS       CHA
## 0.2873793 0.2766483 0.2665768 0.2386478 0.2657658 0.2585227
##       CHN       CIN       CLE       COL       DET       FLO
## 0.2526316 0.2543554 0.2410714 0.2646048 0.2406639 0.2701299
##       HOU       KCA       LAN       MIL       MIN       MON
## 0.2686567 0.2450142 0.2772512 0.2566964 0.2642945 0.2632979
##       NYA       NYN       OAK       PHI       PIT       SDN
## 0.2602740 0.2602740 0.2663317 0.2617960 0.2449412 0.2621359
##       SEA       SFN       SLN       TBA       TEX       TOR
## 0.2698597 0.2603049 0.2620978 0.2523044 0.2678571 0.2633286
```

5.7 These are done with:

```
p <- ggplot(mtcars)
p + aes(x=factor(gear), y=mpg) + geom_boxplot() # boxplots
p + aes(x=wt, y=mpg) + geom_point() + geom_smooth(method="lm") # scatter plot
p + aes(x=hp, y=mpg, col=factor(am), pch=factor(am)) + # add color and shape for transmissi
  geom_point() +
  facet_grid(gear ~ cyl)
```


35
30
25
20
15
10
mpg
3
4
5
factor(gear)


30
20
10
mpg
2
3
4
5
wt


4
6
8
35
30

6

Populations

6.1 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).

6.2 A command to simulate one value of X is

```
 max(sample(1:6,1),sample(1:6,1))
```

```
## [1] 4
```

Repeat this command 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
```

6.3 The frequencies in `Balls` may be used for the probabilities, as shown:

```
with(nba.draft, sample(Team, 1, prob=Balls))
```

```
## [1] Memphis
## 13 Levels: Atlanta Chicago Cleveland Denver ... Washington
```

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

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

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

6.7 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

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

6.8 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
```

6.9 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, size=12, prob=.3)
```

```
## [1] 5.31441e-07
```

6.10 We can answer this using a sum with the following:

```
sum(dbinom(k, size=1e6, prob=1/2))
```

```
## [1] 0
```

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

```
dbinom(4, size=4, prob=1/3)

## [1] 0.01234568
```

6.12 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] 0.5177469

1 - pbinom(0,24,1/36)         # one or more double sixes

## [1] 0.4914039
```

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

6.13 If we roll a pair of dice, the odds of getting a pair of sizes is $p = 1/36$. This can be answered as a binomial question: Let X record the number of double sixes in n rolls, what is the smallest n with $\mathsf{P}(X \geq 1) \geq 1/2$?

As we have `pbinom` answers $\mathsf{P}(X <= k)$, we translate the problem to: what is the smallest n for which $\mathsf{P}(X \leq 0) \leq 1/2$?

```
n <- 18:30                              # some range, may be wrong
names(n) <- n                           # helps, not necessary
pbinom(0, size=n, p=1/36)               # use vectorized n

##        18        19        20        21        22        23
## 0.6022541 0.5855248 0.5692603 0.5534475 0.5380739 0.5231274
##        24        25        26        27        28        29
## 0.5085961 0.4944685 0.4807332 0.4673795 0.4543968 0.4417746
##        30
## 0.4295031
```

Scanning the answers, we see that $n = 25$, not 18. (This can also be solved directly from $(35/36)^n \leq 1/2$, why?)

6.14 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] 0.1794694
```

6.15 These are found from `pnorm` and `qnorm` using the default values for the parameters.

```
pnorm(2.2)
```

```
## [1] 0.9860966
```

```
pnorm(2) - pnorm(-1)
```

```
## [1] 0.8185946
```

```
pnorm(2.5, lower.tail=FALSE)  # or 1 - pnorm(2.5)
```

```
## [1] 0.006209665
```

```
qnorm(.95)
```

```
## [1] 1.644854
```

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

6.16 The probabilities are returned by `pnorm`:

```
1 - pnorm(450, mean=350, sd=75)
```

```
## [1] 0.09121122
```

6.17 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 $\mathrm{P}(X > 22)$ with

```
1 - pnorm(22, mean=20.6, sd=5.5)
```

```
## [1] 0.3995371
```

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] 68250.64
```

6.18 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] 0.8525929
```

```
qnorm(0.85, mean=24.9, sd=1.05)
```

```
## [1] 25.98826
```

6.19 This can be found using `pnorm`.

```
mu <- 3.20
sigma <- 0.35
pnorm(4, mean=mu, sd=sigma) - pnorm(3.5, mean=mu, sd=sigma)
```

```
## [1] 0.1845475
```

6.20 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] 1.973175e-09
```

This is about 2 in a billion.

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

6.22 First we scale the data:

```
x <- father.son$fheight
x <- (x - mean(x))/sd(x)                # or scale(x)[,1]
sum(abs(x) < 1)/length(x)

## [1] 0.6753247

sum(abs(x) < 2)/length(x)

## [1] 0.9619666

sum(abs(x) < 3)/length(x)

## [1] 0.9990724
```

6.23 The quintiles are found using `qnorm`.

```
ps <- seq(0, 1, by=0.2)
qnorm(ps)

## [1]       -Inf -0.8416212 -0.2533471  0.2533471  0.8416212
```

```
## [6]        Inf
```

6.24 For the uniform distribution we can find these probabilities as follows:

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

```
## [1] 0.5773503
```

```
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 different, but found in a similar manner:

```
mu <- 5
sigma <- 5
k <- 1; diff(pexp(mu + c(-1,1)*k*sigma, rate=1/mu))
```

```
## [1] 0.8646647
```

```
k <- 2; diff(pexp(mu + c(-1,1)*k*sigma, rate=1/mu))
```

```
## [1] 0.9502129
```

```
k <- 3; diff(pexp(mu + c(-1,1)*k*sigma, rate=1/mu))
```

```
## [1] 0.9816844
```

6.25 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.

6.26 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.

6.27 The variance is $2k$.

6.28 Using the de Moivre-Laplace theorem for the normal approximation gives

```
n <- 100; p <- 1/2; b <- 42
pbinom(b,n,p)

## [1] 0.06660531

pnorm(b+1/2, n*p, sqrt(n*p*(1-p)))

## [1] 0.0668072
```

6.29 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] 0.004293556
```

6.30 We have $p = 1/2$, $n = 1{,}000$, and X, the number of people for the issue, is 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] 0.000872085
```

6.31 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] 1.110223e-16
```

6.32 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] 0.01123944
```

6.33 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] 0.02476731
```

7

Statistical inference

7.1 The answer is 7, but we can simulate to see:

```
res <- replicate(1000, sum(sample(1:6, 2, replace=TRUE)))
sum(res == 7) > sum(res == 8)


## [1] TRUE
```

7.2 We take a sample with:

```
sam <- sample(rivers, 10, replace=TRUE)
```

and many samples withs

```
sams <- replicate(1000,  mean(sample(rivers, 10, replace=TRUE)))
```

We can then compare these samples to the sample mean:

```
c("mean of samples"=mean(sams), mean=mean(rivers))


## mean of samples            mean
##        596.4608        591.1844
```

8.2 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.

8.3 A "scientific" poll uses a random sample from a target population. In this case, people self-select themselves to vote.

8.4 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.

8.5 The `prop.test` function gives

```
n <- 100; phat <- 0.45
prop.test(n*phat, n, conf.level = 0.8)


##
##  1-sample proportions test with continuity correction
##
## data:  n * phat out of n, null probability 0.5
## X-squared = 0.81, df = 1, p-value = 0.3681
## alternative hypothesis: true p is not equal to 0.5
## 80 percent confidence interval:
##  0.3827104 0.5190311
## sample estimates:
##    p
## 0.45


prop.test(n*phat, n, conf.level = 0.9)


##
```

```
##  1-sample proportions test with continuity correction
##
## data:  n * phat out of n, null probability 0.5
## X-squared = 0.81, df = 1, p-value = 0.3681
## alternative hypothesis: true p is not equal to 0.5
## 90 percent confidence interval:
##  0.3657761 0.5370170
## sample estimates:
##    p
## 0.45
```

8.6 We use `prop.test`, and its default 95% confidence interval, to answer the question "yes."

```
prop.test(5,100)
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  5 out of 100, null probability 0.5
## X-squared = 79.21, df = 1, p-value < 2.2e-16
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
##  0.01855256 0.11829946
## sample estimates:
##    p
## 0.05
```

8.7 The `prop.test` function with the argument `conf.level=0.80` yields

```
prop.test(x=9, n=10, conf.level=0.80)$conf.int
```

```
## [1] 0.6577132 0.9901076
## attr(,"conf.level")
## [1] 0.8
```

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

```
binom.test(x=9, n=10, conf.level=0.80)$conf.int

## [1] 0.6631523 0.9895193
## attr(,"conf.level")
## [1] 0.8
```

8.8 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] 2385.634
```

8.9 The one-sided widths of a 90% confidence interval are:

```
n <- 5;    SE <- sqrt(.8*(.2)/n); qnorm(.95) * SE

## [1] 0.2942404

n <- 100;  SE <- sqrt(.8*(.2)/n); qnorm(.95) * SE

## [1] 0.06579415

n <- 1000; SE <- sqrt(.8*(.2)/n); qnorm(.95) * SE

## [1] 0.02080594
```

8.10 The margins of error are, respectively,

```
n <- 250;  SE <- sqrt(phat*(1-phat)/n);  qnorm(.975) * SE

## [1] 0.06178085
```

```
n <- 1000; SE <- sqrt(phat*(1-phat)/n); qnorm(.975) * SE
```

```
## [1] 0.03089043
```

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

8.11 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 95% confidence
alpha <- 0.05; zstar <- qnorm(1 - alpha/2)
(zstar/0.01)^2/4
```

```
## [1] 9603.647
```

```
## for 90% confidence
alpha <- 0.1; zstar <- qnorm(1 - alpha/2)
(zstar/0.01)^2/4
```

```
## [1] 6763.859
```

```
## for 80% confidence
alpha <- 0.2; zstar <- qnorm(1 - alpha/2)
(zstar/0.01)^2/4
```

```
## [1] 4105.936
```

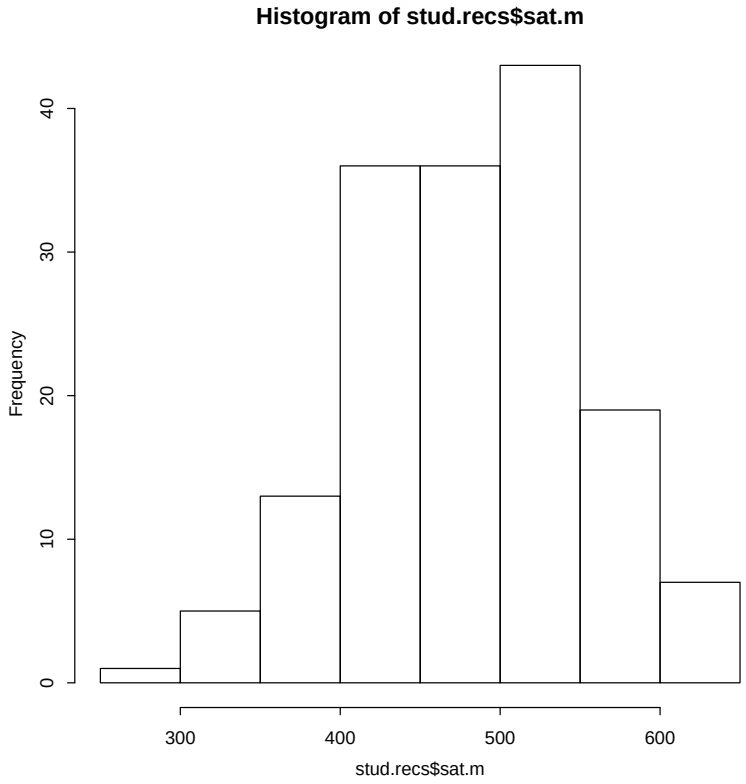
8.13 To compute, we have

```
xbar <- 9.5; s <- 1; n <- 15
alpha <- 0.1
tstar <- qt(1 - alpha/2, df=n-1)
SE <- s/sqrt(n)
c(xbar - tstar*SE, xbar + tstar*SE)


## [1] 9.045232 9.954768
```

We see that ten days is not in this interval.

8.14 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)


##
##  One Sample t-test
##
## data:  stud.recs$sat.m
## t = 88.914, df = 159, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 80 percent confidence interval:
##  478.9043 492.9707
## sample estimates:
## mean of x
##  485.9375
```



8.15 These can be found with

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

##
##  One Sample t-test
##
## data:  homedata$y1970
## t = 262.87, df = 6840, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 90 percent confidence interval:
##  70377.72 71264.14
## sample estimates:
## mean of x
##  70820.93

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

```
##
##  One Sample t-test
##
## data:  homedata$y2000
## t = 169.79, df = 6840, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 90 percent confidence interval:
##  265769.5 270970.0
## sample estimates:
## mean of x
##  268369.8
```

To assess appropriateness, we should ask if either the population appears normal, or n seems large:

```
qqnorm(homedata$y2000)                  # no, skewed
length(homedata$y2000)                  # yes, n=6841
```

8.16 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.

```
hist(yr5$weight)                        # not long tailed, but skewed
length(yr5$weight)                      # not large -- may be issues!
```

```
## [1] 25
```

```
t.test(yr5$weight, conf.level=0.90)
```

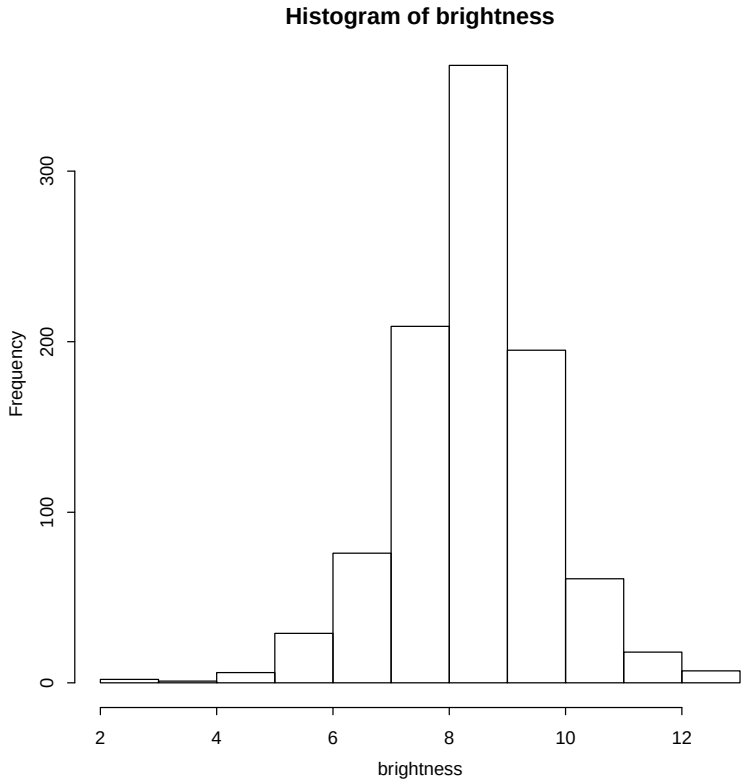
```
##
##  One Sample t-test
##
## data:  yr5$weight
## t = 35.478, df = 24, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 90 percent confidence interval:
##  42.75381 47.08619
## sample estimates:
## mean of x
##     44.92
```

The value is returned, centered around 44.92. The shape of the distribution may give us pause for concern about the inference though.

8.17 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)              # bell-shaped, looks good
t.test(brightness, conf.level = 0.90)$conf.int


## [1] 8.349184 8.486303
## attr(,"conf.level")
## [1] 0.9
```



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.

8.18 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 = 1527.9, df = 129, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 90 percent confidence interval:
##  98.14269 98.35577
## sample estimates:
## mean of x
```

```
##  98.24923
```

8.19 A simulation can be made as follows:

```
require(HistData)
finger <- with(Macdonell, rep(finger, frequency))
res <- replicate(750, mean(sample(finger, 4, replace=TRUE)))
quantile(res, c(0.025, 0.975))


##     2.5%    97.5%
## 11.04312 12.12500
```

Comparing to the following:

```
n <- 4
x <- sample(finger, n, replace=TRUE)
t.test(x, conf.level=0.95)$conf.int


## [1] 10.71299 12.88701
## attr(,"conf.level")
## [1] 0.95
```

In this simulation the latter is a bit wider.

8.20 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.

8.21 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.

8.22 The following commands will do the simulation:

```
set.seed(10)
```

```
M <- 200; n <- 10
alpha <- 1 - 0.90
res <- replicate(M, {
  x <- rnorm(n, mean=0, sd=2)
  c(z=qnorm(1-alpha/2) * 2/sqrt(n),
    t=qt(1 - alpha/2, df=n-1) * sd(x)/sqrt(n))
})                                          # a matrix
sum(res["z",] < res["t", ]) / M

## [1] 0.64
```

In this simulation it is 64% of the time. The z one is usually smaller, but not as often as might have been guessed.

8.23 We have

```
n <- 10; m <- 20
alpha <- 1 - 0.80
lr <- qf(c(alpha/2, 1- alpha/2), df1=n-1, df2=m-1)
lr

## [1] 0.4338415 1.9836388

s <- (2.3/2.8)^2
sqrt(s/rev(lr))

## [1] 0.5832282 1.2471086
```

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

8.24 The confidence interval can be found with:

```
place <- nym.2002$place
n <- length(place)
alpha <- 1 - 0.90
(1-alpha)^(1/n)
```

```
## [1] 0.9998946

max(place)/(1-alpha)^(1/n)

## [1] 23664.49

max(place)

## [1] 23662
```

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

8.25 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:

```
c1 <- c(3.1, 3.3, 1.7, 1.2, 0.7, 2.3, 2.9)
c2 <- c(1.8, 2.3, 2.2, 3.5, 1.7, 1.6, 1.4)
t.test(c1,c2, conf.level=0.80)

##
##  Welch Two Sample t-test
##
## data:  c1 and c2
## t = 0.21592, df = 10.788, p-value = 0.8331
## alternative hypothesis: true difference in means is not equal to 0
## 80 percent confidence interval:
##  -0.5322402  0.7322402
## sample estimates:
## mean of x mean of y
##  2.171429  2.071429
```

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)$conf.int

## [1] -0.5281055  0.7281055
```

```
## attr(,"conf.level")
## [1] 0.8
```

8.26 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. Even though the spreads are similar, it could be expected that there may be a difference in the population variances, so the conservative choice (the default) is made:

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

```
##
##  Welch Two Sample t-test
##
## data:  gp.400 and gp.1200
## t = 1.1623, df = 16.354, p-value = 0.2618
## alternative hypothesis: true difference in means is not equal to 0
## 90 percent confidence interval:
##  -0.9502007  4.7502007
## sample estimates:
## mean of x mean of y
##       5.6       3.7
```

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

8.27 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 = 0.041019, df = 9, p-value = 0.9682
## alternative hypothesis: true difference in means is not equal to 0
## 90 percent confidence interval:
##  -4.368937  4.568937
## sample estimates:
## mean of the differences
##                     0.1
```

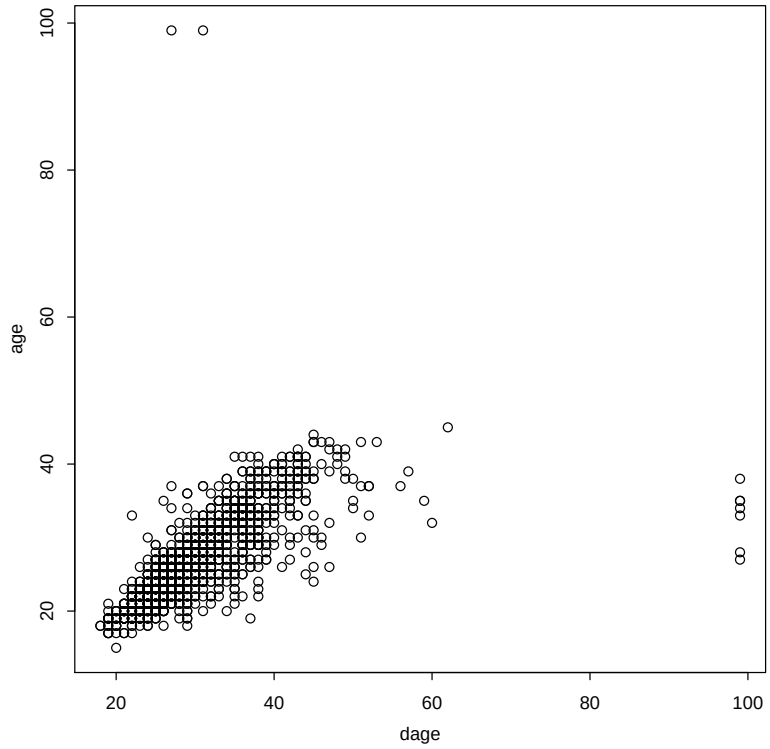
The exploratory plots show the correlation between the two sets of scores and the distribution of the differences, indicating that a paired *t*-test is appropriate.

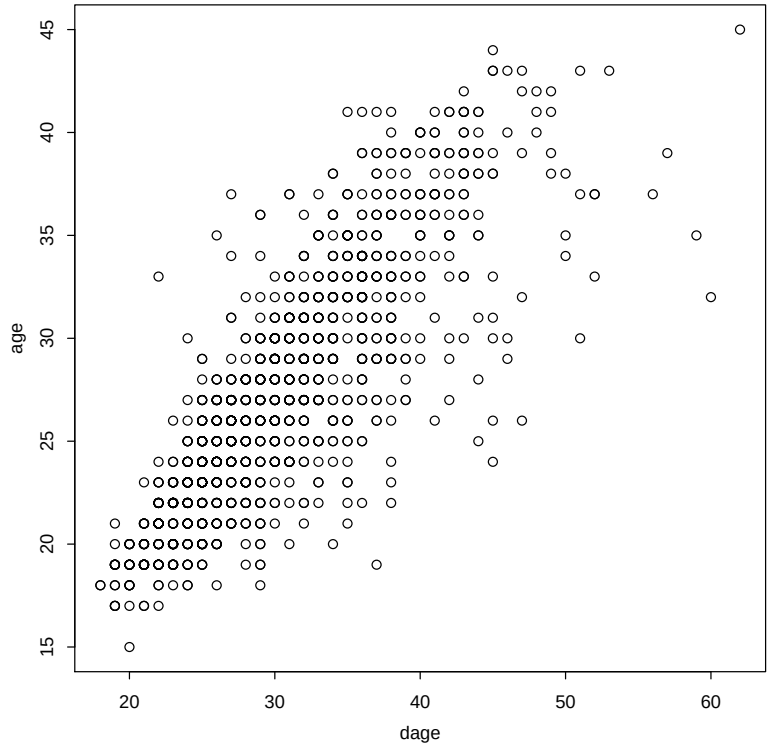
8.28 We would reasonably expect that the two parents are of a similar age. This leads us to use a paired *t*-test after checking the validity of the assumption that the differences in age are randomly sampled from a normally distributed population.

```
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 = 28.092, df = 1226, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  2.908749 3.345530
## sample estimates:
## mean of the differences
```

```
##                 3.127139
```





Histogram of b$dage − b$age

8.29 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.

```
commutes <- rep(c(21,22,23,24,25,26,29,31,33),
  c(3,2,2,6,2,1,1,2,1))
ans <- wilcox.test(log(commutes), conf.int=TRUE, conf.level=0.9)

## Warning in wilcox.test.default(log(commutes), conf.int = TRUE,
conf.level = 0.9): cannot compute exact p-value with ties
## Warning in wilcox.test.default(log(commutes), conf.int = TRUE,
conf.level = 0.9): cannot compute exact confidence interval with
ties

confint(ans)

## (3.13, 3.26) with 90 percent confidence

confint(ans, transform=exp)             # also exp(ans$conf.int)

## (22.98, 26.11) with 90 percent confidence
```

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.[1]

8.30 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)

## Warning in wilcox.test.default(u2$October, u2$"The Joshua Tree",
conf.int = TRUE): cannot compute exact p-value with ties
```

[1] The `coin` package has several functions that can work with tied data.

```
## Warning in wilcox.test.default(u2$October, u2$"The Joshua Tree",
conf.int = TRUE): cannot compute exact confidence intervals with
ties

##
##  Wilcoxon rank sum test with continuity correction
##
## data:  u2$October and u2$"The Joshua Tree"
## W = 26.5, p-value = 0.02778
## alternative hypothesis: true location shift is not equal to 0
## 95 percent confidence interval:
##  -94.000025  -9.999975
## sample estimates:
## difference in location
##              -48.99996
```

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

8.31 The AGE variable is symmetric, so no transformation is needed.

```
hist(cfb$AGE)                 # symmetric
wilcox.test(cfb$AGE, conf.int=TRUE)

##
##  Wilcoxon signed rank test with continuity correction
##
## data:  cfb$AGE
## V = 500500, p-value < 2.2e-16
## alternative hypothesis: true location is not equal to 0
## 95 percent confidence interval:
##  47.99992 50.00005
## sample estimates:
## (pseudo)median
##       48.99996
```

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

```
hist(log(cfb$INCOME +1))
ans <- wilcox.test((log(cfb$INCOME +1)), conf.int=TRUE)
exp(ans$conf.int) - 1

## [1] 35960.81 40470.11
```

```
## attr(,"conf.level")
## [1] 0.95
```

8.32 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))
```

9

Significance tests

9.1 The FDA must assume that the drug is safe in the null hypothesis. The alternative would be that it is not safe.

9.2 We tabulate to get the counts.

```
cnts = table(samhda$marijuana)
cnts
```

```
##
##   1   2   9
## 134 463   3
```

We ignore the cases coded with 9, as they presumably are for individuals who chose not to answer. We see that there are 134 who answered "yes" and 463 who answered "no." The hypothesis test is between

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

The p-value is given by

```
x <- cnts[1]
n <- cnts[1] + cnts[2]
prop.test(x, n, p=.5, alternative="greater")
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  x out of n, null probability 0.5
## X-squared = 180.21, df = 1, p-value = 1
## alternative hypothesis: true p is greater than 0.5
## 95 percent confidence interval:
##  0.1968507 1.0000000
```

```
## sample estimates:
##         p
## 0.2244556
```

This finds a p-value of 1. Decidely not significant. The data do not support the alternative, rather they suggest a decrease in the percentage of those who have tried marijuana.

9.3 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 $\widehat{p} = .8$.

```
out <- prop.test(40,50, p = .75, alternative="greater")
out$p.value


## [1] 0.2568146
```

which is not significant.

9.4 A test of significance is done on the hypotheses

$$H_0 : p = 0.281, \quad H_A : p \neq 0.281.$$

This is carried out by computing the count from the percent and passing this value off to `prop.test`:

```
out <- prop.test(0.40 * 75, 75, p = 0.281, alternative="two.sided")
out$p.value


## [1] 0.03043972
```

This p-value is significant.

9.5 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

```
out <- prop.test(5731, 5760, p=0.999, alternative="less")
out$p.value


## [1] 1.274574e-21
```

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

9.6 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 $p = 0.1500$ and standard deviation $\sqrt{p(1-p)/n}$.

```
p <- 0.1500; n <- 150000
out <- qnorm(.95, mean=p,sd=sqrt(p*(1-p)/n))
```

Multiplying by n produces the cutoff:

```
n * out


## [1] 22727.47
```

9.7 This is found with

```
out <- prop.test(x=2700, n=25000, p=0.1, alternative="greater" )
out$p.value


## [1] 1.300633e-05
```

9.8 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] 5.015385
```

```
pt(T, df=n-1, lower.tail=FALSE)


## [1] 1.998964e-05
```

The significance test has a small p-value.

9.9 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 <- 4.03; s <- 0.42; n <- 800
mu <- 4.00
SE <- s/sqrt(n)
obs <- (xbar - mu)/SE
pnorm(obs, lower.tail=FALSE)


## [1] 0.02167588
```

9.10 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$sat.m)         # n is large, data is short-tailed
out <- t.test(stud.recs$sat.m, mu=500, alternative="two.sided")
out$p.value <= 0.05                     # TRUE, reject


## [1] TRUE
```

9.11 First check normality, then use `t.test` as follows:

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


##
##  One Sample t-test
##
```

```
## data:  x
## t = 20.796, df = 743, p-value < 2.2e-16
## alternative hypothesis: true mean is greater than 68
## 95 percent confidence interval:
##  70.02973      Inf
## sample estimates:
## mean of x
##   70.2043
```

9.12 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)
obs <- (xbar - mu)/SE
2*pt(obs, df = n-1, lower.tail=FALSE)


## [1] 0.7376588
```

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

9.13 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 should apply. This yields

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


##
##  One Sample t-test
##
## data:  OBP
## t = -0.16846, df = 437, p-value = 0.8663
```

```
## alternative hypothesis: true mean is not equal to 0.33
## 95 percent confidence interval:
##  0.3257100 0.3336126
## sample estimates:
## mean of x
## 0.3296613
```

9.14 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

```
x <- normtemp$temperature
hist(x)
t.test(x, mu = 98.6, alternative="two.sided")


##
##  One Sample t-test
##
## data:  x
## t = -5.4548, df = 129, p-value = 2.411e-07
## alternative hypothesis: true mean is not equal to 98.6
## 95 percent confidence interval:
##  98.12200 98.37646
## sample estimates:
## mean of x
##  98.24923
```

The p-value is quite small. The computed confidence interval contains 98.2, but not 98.6.

9.15 We compute the p-value directly for a one-sided test with a null hypothesis of \$200:

```
xbar <- 650
n <- 8
s <- 100
mu <- 200
obs <- (xbar - mu) / (s/sqrt(n))
pt(obs, df=n-1, lower.tail=FALSE)


## [1] 2.139251e-06
```

The small p-value indicates the difference is statistically significant.

9.16 We need to fill in all the known values for `power.t.test` and let, it compute the unknown minimimal sample size:

```
alpha <- 0.05; beta <- 0.20
delta <- 1; std <- 1
power.t.test(sd=std, sig.level=alpha, power=1-beta, delta=delta,
             type="one.sample", alternative="one.sided")


##
##      One-sample t test power calculation
##
##               n = 7.727622
##           delta = 1
##              sd = 1
##       sig.level = 0.05
##           power = 0.8
##     alternative = one.sided
```

9.17 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] 0.03252251
```

9.18 The `log` function can be used directly, as in

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


##
##  Wilcoxon signed rank test with continuity correction
##
## data:  log(exec.pay)
## V = 10966, p-value = 0.004144
## alternative hypothesis: true location is greater than 3.091042
```

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))
```

9.19 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

```
smokers <- subset(babies, smoke == 1 & gestation != 999)
wilcox.test(smokers$gestation, mu = 40*7)


## Error in wilcox.test.default(smokers$gestation, mu = 40 * 7):
not enough (finite) 'x' observations
```

9.20 Genrally the t-test is more effective at rejecting when the null hypothesis is false. Again, the t-test makes strong assumptions on the data, so one would expect the sampling distribution to be "tighter."

9.21 Assuming the phones are a random sample from the population of all phones, this can be done with `prop.test`, as follows:

```
x <- c("A"=14, "B"=15)
n <- c("A"=150, "B"=125)
```

```
prop.test(x, n, alternative="less")


##
##  2-sample test for equality of proportions with
##  continuity correction
##
## data:  x out of n
## X-squared = 0.27016, df = 1, p-value = 0.3016
## alternative hypothesis: less
## 95 percent confidence interval:
##  -1.00000000  0.04240782
## sample estimates:
##     prop 1     prop 2
## 0.09333333 0.12000000
```

The p value is not small, the answer is "no."

9.22 This is a two-sided test of proportions. We need to compute the counts for the 2013 values, as the problem gives us a proportion. Then we can call `prop.test`.

```
n <- c(600, 1050)
x <- c(250, n[2] * 0.52)
prop.test(x, n)                          # use default alternative


##
##  2-sample test for equality of proportions with
##  continuity correction
##
## data:  x out of n
## X-squared = 15.917, df = 1, p-value = 6.619e-05
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.1543351 -0.0523316
## sample estimates:
##    prop 1    prop 2
## 0.4166667 0.5200000
```

The small p-value suggests that the null hypothesis of no difference of opinion does not describe the data well.

9.23 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:

```
phat <- c(0.82, 0.70)
n <- c(350, 350)
x <- n * phat
prop.test(x, n)                          # using default alternative


##
##  2-sample test for equality of proportions with
##  continuity correction
##
## data:  x out of n
## X-squared = 13.166, df = 1, p-value = 0.0002851
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  0.05449405 0.18550595
## sample estimates:
## prop 1 prop 2
##   0.82   0.70
```

9.24 We compare with `prop.test`:

```
x <- c("had"=153, "didnot"=196)
n <- c("had"=30000, "didnot"=30000)
prop.test(x, n)                          # use two.sided default


##
##  2-sample test for equality of proportions with
##  continuity correction
##
## data:  x out of n
## X-squared = 5.084, df = 1, p-value = 0.02415
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.0026835628 -0.0001831039
## sample estimates:
##      prop 1      prop 2
## 0.005100000 0.006533333
```

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

9.25 The p-value computed from these numbers is found from `prop.test` as follows:

```
x <- c(18, 3)
n <- c(22, 22)
prop.test(x, n)


##
##  2-sample test for equality of proportions with
##  continuity correction
##
## data:  x out of n
## X-squared = 17.855, df = 1, p-value = 2.384e-05
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  0.4206344 0.9430019
## sample estimates:
##    prop 1    prop 2
## 0.8181818 0.1363636
```

The small p-value reflects the obvious difference between the samples.

9.26 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
```

Okay, seems like a normal approximation will fit the data under our assumptions. The p value is:

```
percents <- c(98.9,96.9)
n <- c(1250, 1100)
p <- percents/100
x <- n * p
prop.test(x, n)                          # use two.sided default


##
```

```
##  2-sample test for equality of proportions with
##  continuity correction
##
## data:  x out of n
## X-squared = 10.752, df = 1, p-value = 0.001042
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  0.007383812 0.032616188
## sample estimates:
## prop 1 prop 2
##  0.989  0.969
```

9.27 We suppose the survey of 100 was a random sample from the population of all students, and the 1,000 in the CDC survey was as well, so we can use `prop.test` to compute a p-value.

Given that, we have the one-sided test performed as:

```
p <- c(0.31, 0.41)
n <- c(1000, 100)
x <- n * p
prop.test(x, n, alternative="less")
```

```
##
##  2-sample test for equality of proportions with
##  continuity correction
##
## data:  x out of n
## X-squared = 3.7365, df = 1, p-value = 0.02662
## alternative hypothesis: less
## 95 percent confidence interval:
##  -1.00000000 -0.01009961
## sample estimates:
## prop 1 prop 2
##   0.31   0.41
```

The small p-value indicates the difference is "statistically significant."

9.28 We have $n_1 = n_2 = 10{,}000$ and $\widehat{p}_1 = 0.216$, $\widehat{p}_2 = .193$. A two-sided significance test is performed with `prop.test` as follows:

```
p <- c(.216,.193)
n <- c(10000,10000)
x <- n * p
out <- prop.test(x,n)
out$p.value < 0.01                          # TRUE

## [1] TRUE
```

The difference is statistically significant. However, if the surveys were only of size 1,000 the difference would not have been statistically significant. As always, differences must be assessed in terms of variability which depends on sample size.

9.29 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] 1.224317e-43
```

9.30 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] 0.6868484
```

9.31 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, alternative="greater"))
```

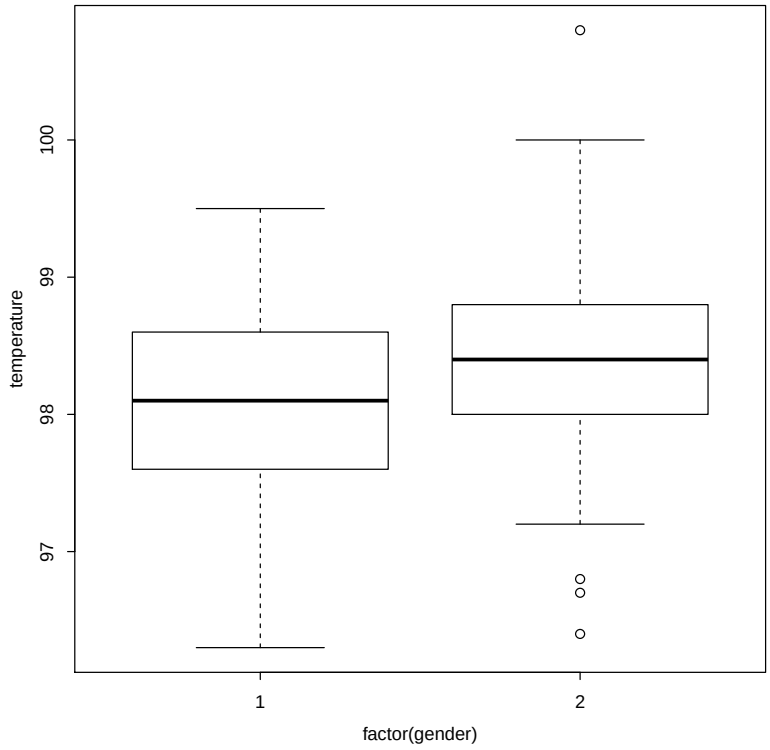
```
##
##  One Sample t-test
##
## data:  dage - age
## t = 28.092, df = 1226, p-value < 2.2e-16
## alternative hypothesis: true mean is greater than 0
## 95 percent confidence interval:
##  2.943902      Inf
## sample estimates:
## mean of x
##  3.127139
```

9.32 This data lends itself to the formula interface for `t.test`:

```
plot(temperature ~ factor(gender), data=normtemp)
t.test(temperature ~ factor(gender), data=normtemp)
```

```
##
##  Welch Two Sample t-test
##
## data:  temperature by factor(gender)
## t = -2.2854, df = 127.51, p-value = 0.02394
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.53964856 -0.03881298
## sample estimates:
```

```
## mean in group 1 mean in group 2
##        98.10462        98.39385
```



The small p-value indicates a statistically significant difference.

9.33 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, alternative="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:
##  0.9727256       Inf
## sample estimates:
## mean of x
##      3.25
```

The p-value indicates that the null hypothesis of "no improvement" is not consistent with the data.

9.34 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, alternative="two.sided")
```

```
##
##  One Sample t-test
##
## data:  method1 - method2
## t = -5.0778, df = 9, p-value = 0.0006649
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -7.531073 -2.888927
## sample estimates:
## mean of x
##     -5.21
```

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

9.35 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)
```

```
## [1] "A" "B"
```

```
plot(B ~ A, data=shoes)       # related (not shown)
with(shoes, t.test(A,B,paired=TRUE)$p.value)

## [1] 0.008538781

with(shoes, t.test(A,B)$p.value)

## [1] 0.7165011
```

Not realizing the paired nature gives a completely different (and incorrect) p-value.

9.36 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

```
with(Galton, t.test(child,parent, paired=TRUE))

##
##  Paired t-test
##
## data:  child and parent
## t = -2.8789, df = 927, p-value = 0.004082
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.36949983 -0.06993982
## sample estimates:
## mean of the differences
##              -0.2197198
```

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

9.37 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 = 0.34183, 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:
##  0.0849059 1.3762082
## sample estimates:
## ratio of variances
##          0.3418306
```

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

10

Goodness of fit

10.1 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.

10.2 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)


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


##
```

```
##  Chi-squared test for given probabilities
##
## data:  obs
## X-squared = 152.56, df = 5, p-value < 2.2e-16
```

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.

10.3 The data set gives percentages, but `chisq.test` wants proportions. We first define a quick function to compute proportions, then apply the two tests.

The data set is a data frame, where each row is of interest. Some care must be taken when extracting the probabilities, as `chisq.test` needs a vector not a data frame. Here we use `unlist` to provide that.

```
prob <- mandms["milk chocolate", ]
prob <- unlist(prob)
bagful <- c(15, 34, 7, 19, 29, 24)
names(bagful) = c("blue", "brown", "green", "orange", "red", "yellow")
chisq.test(bagful, p=prob/sum(prob))


##
##  Chi-squared test for given probabilities
##
## data:  bagful
## X-squared = 7.0651, df = 5, p-value = 0.2158
```

We repeat with "Peanut:"

```
prob <- mandms["Peanut", ]
prob <- unlist(prob)
chisq.test(bagful, p=prob/sum(prob))


##
##  Chi-squared test for given probabilities
##
## data:  bagful
## X-squared = 13.328, df = 5, p-value = 0.02049
```

It appears that the bag is milk chocolate not peanut, though repeatedly performing significance tests requires an adjustment to the significance level.

10.4 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.

10.5 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))


##
##  Chi-squared test for given probabilities
##
## data:  counts
## X-squared = 1.084, df = 4, p-value = 0.8968
```

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

10.6 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$name, sep="", collapse="")
x <- unlist(strsplit(tolower(all.names), ""))
letter.dist <- sapply(letters, function(i) sum(x == i))
ps <- scrabble$frequency[1:26]
chisq.test(letter.dist, p=ps/sum(ps))
```

```
##
##  Chi-squared test for given probabilities
##
## data:  letter.dist
## X-squared = 260.73, 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.

10.7 The first test is done quite quickly using the defaults, as the uniform distribution is the null hypothesis.

```
murder <- c(63 , 53 , 50 , 51 , 55 , 52 , 56)
chisq.test(murder)


##
##  Chi-squared test for given probabilities
##
## data:  murder
## X-squared = 2.1263, df = 6, p-value = 0.9077
```

The p-value is not small. It may be said there is no statistically significant difference in the day.

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] 0.7099884


1 - pchisq(cs, df=1)


## [1] 0.3994477
```

Again, the p-value is not small.

10.8 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] 1.195122


pchisq(cs,df=2,lower.tail=FALSE)


## [1] 0.5501518
```

Despite the apparent difference, it is not statistically significant.

10.9 The histogram and density curves usually show a poor fit in the two instances shown.

10.10 Start with the formula for the chi-squared statistic, rewrite in terms of sample proportions, then use the facts that $\widehat{p}_1 + \widehat{p}_2 = 1$ and $p_1 + p_2 = 1$:

$$\begin{aligned}\sum_{i=1}^{n} \frac{(e_i - f_i)^2}{e_i} &= \frac{(np_1 - n\widehat{p}_1)^2}{np_1} + \frac{(np_2 - n\widehat{p}_2)^2}{np_2} \\ &= n\left(\frac{(1-p_1)(p_1 - \widehat{p}_1)^2 + p_1(1 - p_1 - (1 - \widehat{p}_1))^2}{p_1(1-p_1)}\right) \\ &= \frac{(p_1 - \widehat{p}_1)^2}{p_1(1-p_1)/n}.\end{aligned}$$

10.11 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)
```

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

```
##
##  Pearson's Chi-squared test
##
## data:  accidents
## X-squared = 12.586, df = 8, p-value = 0.1269
```

10.12 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),]
aq <- transform(aq,
  te = cut(Temp, quantile(Temp)),
  oz = cut(Ozone,quantile(Ozone))
)
xtabs(~ te + oz, data=aq)
```

```
##            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(xtabs(~ te + oz, data=aq))
```

```
##
##  Pearson's Chi-squared test
##
## data:  xtabs(~te + oz, data = aq)
## X-squared = 76.309, df = 9, p-value = 8.713e-13
```

10.13 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 = 59.224, df = 3, p-value = 8.61e-13
```

10.14 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)


## Warning in chisq.test(oral.lesion): Chi-squared approximation
may be incorrect


##
##  Pearson's Chi-squared test
##
## data:  oral.lesion
## X-squared = 22.099, df = 16, p-value = 0.14


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 = 22.099, df = NA, p-value = 0.02299
```

10.15 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 = 14.037, df = 4, p-value = 0.007179
```

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

10.16 The data are not a sample, rather a census. As such the question doesn't make much sense, unless it is viewed as a sample from a counterfactual population. Regardless, the following p-value indicates that the assumption of homogeneity is supported by the data.

```
y2011 <- c(63, 53, 50, 51, 55, 52, 56)
y2003 <- c(53, 42, 51, 45, 36, 37, 65)
x <- rbind(y2011, y2003)
chisq.test(x)


##
##  Pearson's Chi-squared test
##
## data:  x
## X-squared = 6.0479, df = 6, p-value = 0.4178
```

10.17 The two tests can be carried out with

```
shapiro.test(babies$ht[babies$ht < 99])$p.value


## [1] 4.893686e-10


shapiro.test(babies$wt[babies$wt < 999])$p.value


## [1] 0.001191711
```

In each case the null hypothesis would be rejected.

10.18 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 = 0.98222, p-value = 1.825e-09
```

10.19 The Shapiro-Wilk test accepts a null hypothesis of normally distributed data.

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

```
##
##  Shapiro-Wilk normality test
##
## data:  normtemp$temperature
## W = 0.98658, p-value = 0.2332
```

10.20 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
##   2.578975570   0.004363394
##  (0.268578387) (0.000488203)
```

```
plot(density(rivers))
curve(dgamma(x,shape=2.578, rate= 0.00436), add=TRUE, 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))
```

10.21 This can be done using `fitdistr` from the `MASS` package. For the math SAT scores we have:

```
fitdistr(stud.recs$sat.m, "normal")


##       mean          sd
##   485.937500    68.913867
##  (  5.448120) (  3.852402)
```

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

10.22 We do the simulations with both distributions and check for how often we would reject at the 0.05 level.

```
M <- 1000
res.t   <- replicate(M, ks.test(rt(25, df=3), "pnorm")$p.value)
res.exp <- replicate(M, ks.test(rexp(25)-1,  "pnorm")$p.value)
sum(res.t < .05)/M


## [1] 0.081


sum(res.exp < .05)/M


## [1] 0.264
```

The exponential distribution was only "rejected" 26 percent of the time, and the t-distribution only 8 percent of the time.

10.24 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
M <- 500
res.100  = replicate(M, f(n=100))
res.1000 = replicate(M, f(n=1000))
res.4000 = replicate(M, f(n=4000))
sum(res.100  <= 0.05)/M


## [1] 0.988


sum(res.1000 <= 0.05)/M
```

```
## [1] 0.908
```

```
sum(res.4000 <= 0.05)/M
```

```
## [1] 0.402
```

Compare this to what happens when no outlier is present:

```
res.100.nooutlier <- replicate(M, f(n=100, outlier=rnorm(1)))
sum(res.100.nooutlier <= 0.05)/500
```

```
## [1] 0.05
```

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

11

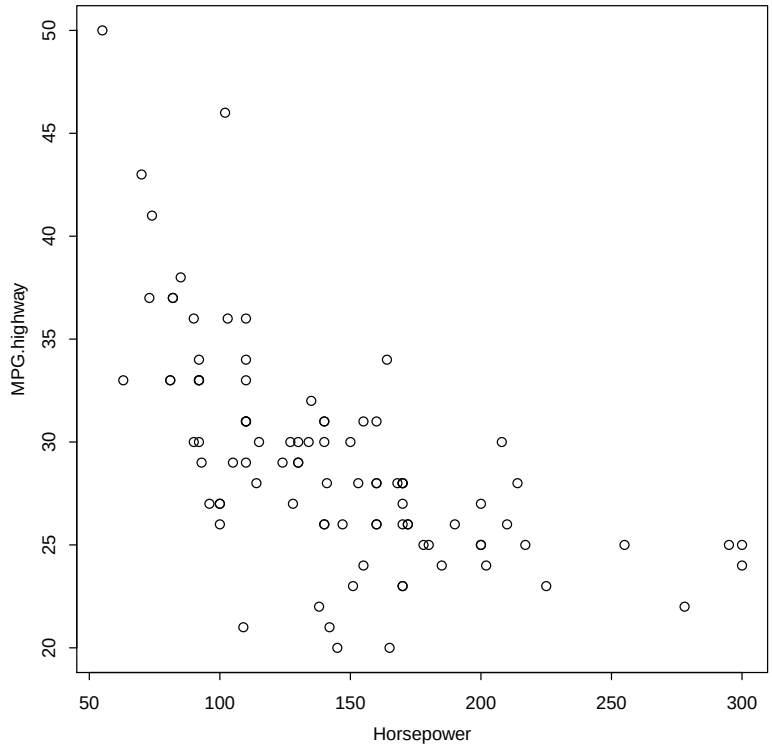
Linear regression

11.1 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
##    38.14988     -0.06302

predict(res, newdata=data.frame(Horsepower=225))


##        1
## 23.97066
```

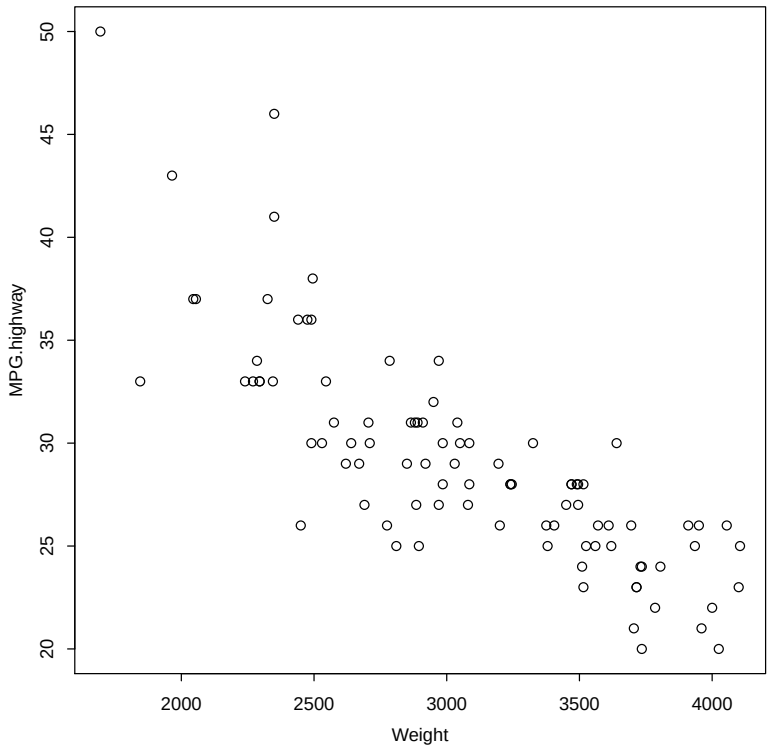


Modeling highway mileage by automobile weight should have a similar negative correlation. Again we confirm and make the requested predictions.

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


##
## Call:
## lm(formula = f, data = Cars93)
##
## Coefficients:
## (Intercept)       Weight
##   51.601365    -0.007327
```

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

```
##         1         2
## 33.107868  4.708186
```

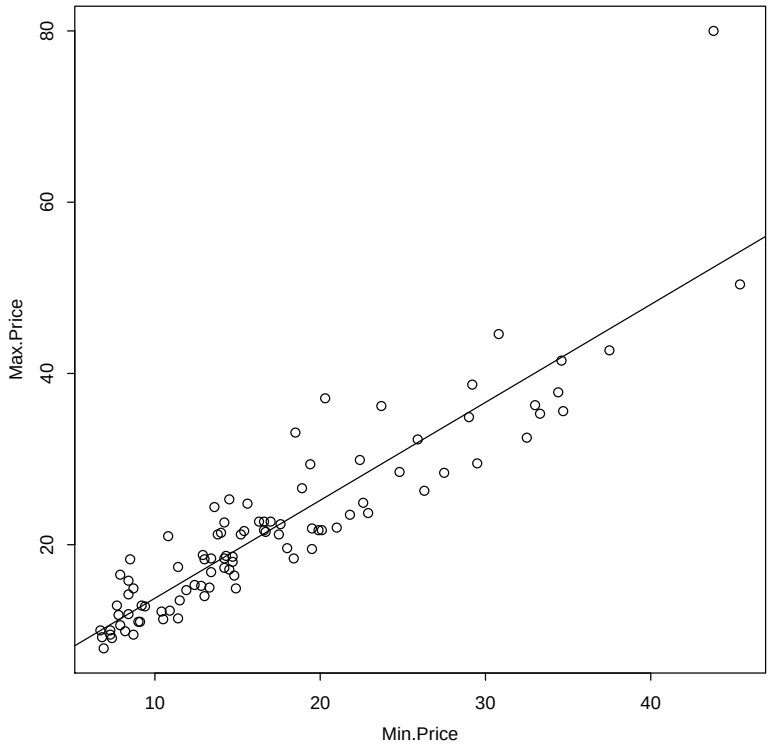


The prediction for the MINI Cooper may be close, but there is no reason to expect the prediction for the HUMMER to be close, as the value of the predictor is outside the range of the data.

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
##       2.314        1.144
```



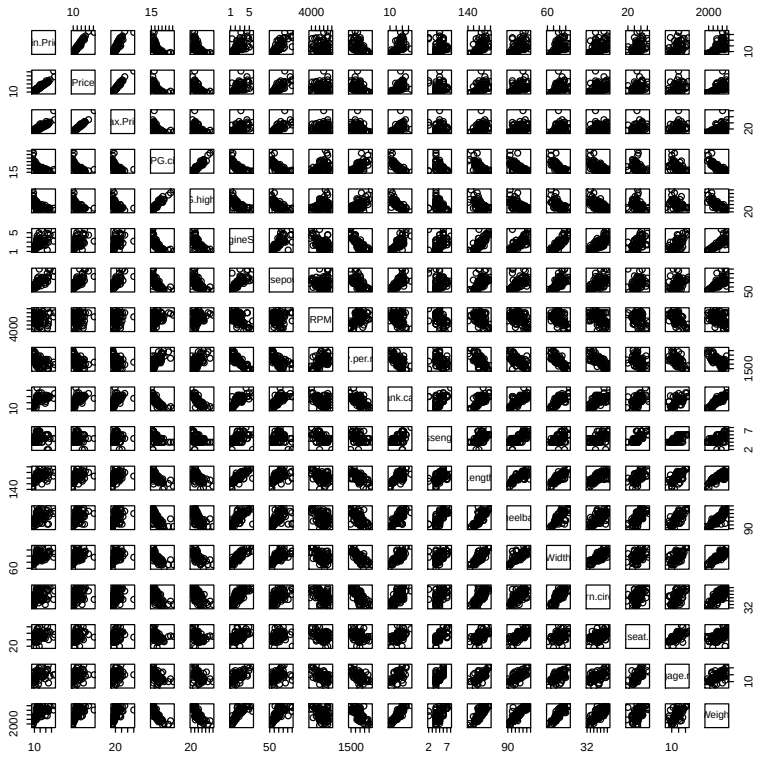
The slope of 1.14 indicates that perhaps add-ons for more expensive cars cost more, but in this case it appears to be due to the one large outlier, as robust regression estimates are much closer to 1:

```
rlm(f, data=Cars93)
```

```
## Call:
## rlm(formula = f, data = Cars93)
## Converged in 7 iterations
##
## Coefficients:
## (Intercept)   Min.Price
##    3.609198    1.028727
##
## 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)`.

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

```
lm(attendance ~ wins,MLBattend)
```

```
##
## Call:
## lm(formula = attendance ~ wins, data = MLBattend)
##
## Coefficients:
## (Intercept)         wins
##     -378164        27345
```

```
27345*10


## [1] 273450
```

11.3 We enter in the data and then use the regression coefficients directly to make a prediction.

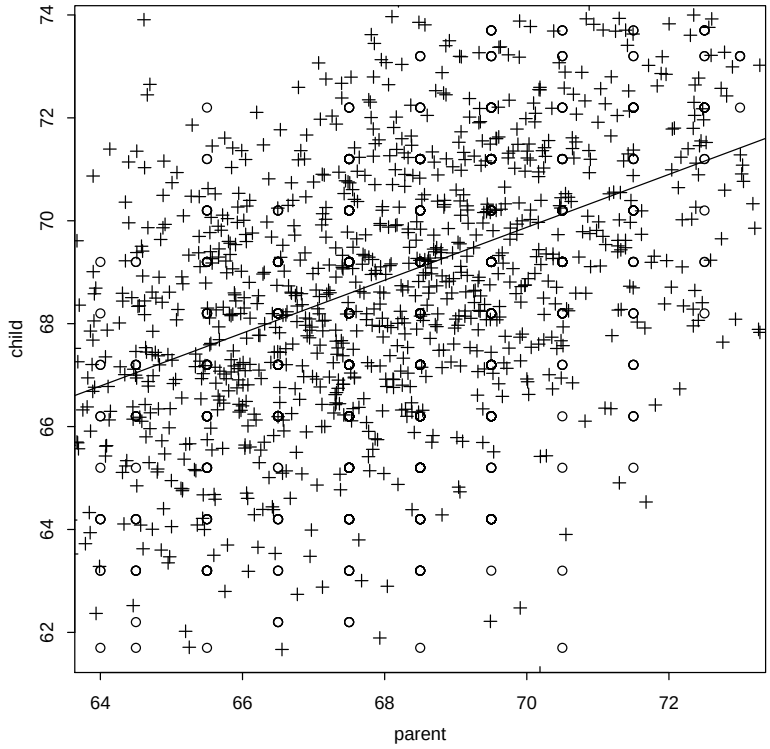
```
age.two <- c(39, 30, 32, 34, 35, 36, 36, 30)
adult <- c(71, 63, 63, 67, 68, 68, 70, 64)
lm(adult ~ age.two)


##
## Call:
## lm(formula = adult ~ age.two)
##
## Coefficients:
## (Intercept)      age.two
##     35.1786       0.9286


35.179 + .929*33


## [1] 65.836
```

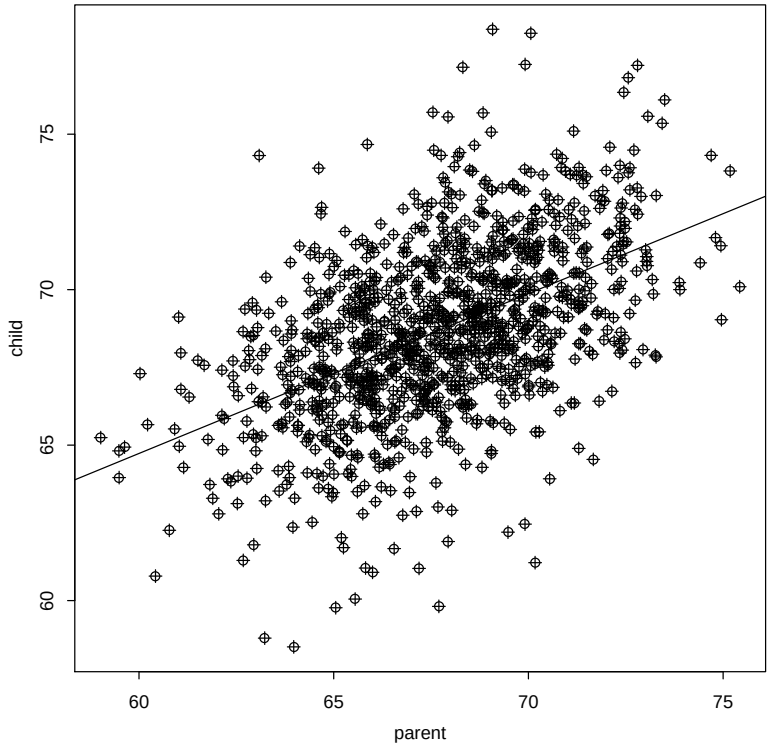
We see that the slope is not close to 2, although this is too small a data set to draw any generalizations from.

11.4 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.

```
plot(child ~ parent, data=galton)
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


##
## Call:
## lm(formula = child ~ parent)
##
## Coefficients:
## (Intercept)       parent
##     33.8866       0.5141


cor(parent,child)


## [1] 0.5013383
```


child
parent
62
64
66
68
70
72
74
64
66
68
70
72


child
parent
60
65
70
75
60
65
70
75

11.5 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).

```
lm(scale(child) ~ scale(parent), data=galton)
```

```
##
## Call:
## lm(formula = scale(child) ~ scale(parent), data = galton)
##
## Coefficients:
##   (Intercept)  scale(parent)
##     2.983e-15      4.588e-01
```

```
with(galton, cor(scale(child), scale(parent)))
```

```
##           [,1]
## [1,] 0.4587624
```

11.6 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)
```

```
##
## Call:
## lm(formula = price ~ no.bed)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -108.00  -53.95   -5.75   59.77   99.10
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    94.40      97.98   0.963   0.3635
## no.bed         73.10      23.76   3.076   0.0152 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 75.15 on 8 degrees of freedom
## Multiple R-squared:  0.5419,Adjusted R-squared:  0.4846
## F-statistic: 9.462 on 1 and 8 DF,  p-value: 0.01521


T <- (73.1 - 60)/23.8
pt(T, df=8, lower.tail=FALSE)


## [1] 0.2985304
```

The large p-value is not significant.

11.7 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)


##
## Call:
## lm(formula = BAL ~ no.beers)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.0275 -0.0187 -0.0071  0.0194  0.0357
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.018500   0.019230  -0.962 0.364200
## no.beers     0.019200   0.003511   5.469 0.000595 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02483 on 8 degrees of freedom
## Multiple R-squared:  0.789,Adjusted R-squared:  0.7626
## F-statistic: 29.91 on 1 and 8 DF,  p-value: 0.0005953


T <- (0.01920 - .02)/0.00351
2*pt(T, df = length(no.beers)-2)


## [1] 0.8254257
```

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

11.8 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)
short_summary(lm(BAL ~ no.beers))


##               Estimate Std. Error t value  Pr(>|t|)
## (Intercept) -0.0185000  0.0192299 -0.9620 0.3641997
## no.beers     0.0192000  0.0035109  5.4687 0.0005953 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-value for this data is 0.3642.

11.9 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.

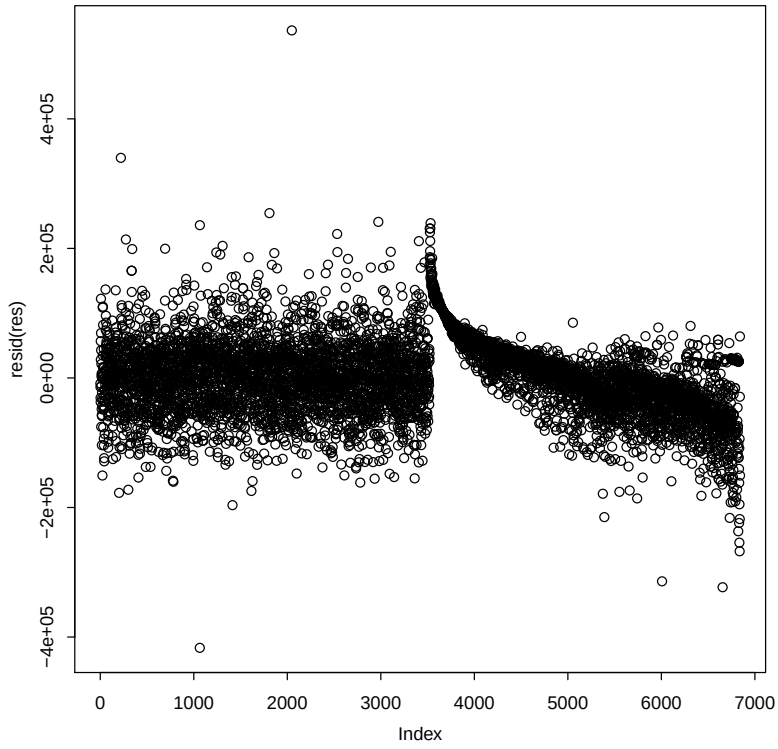
```
x <- read.table(file="tmp.txt")
names(x) <- c("elev", "Temp")
res <- lm(Temp ~ elev, x)
summary(res)


##
## Call:
## lm(formula = Temp ~ elev, data = x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0844 -0.4433  0.1369  0.5072  3.1025
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 59.2906619  1.7173294  34.525 3.93e-08 ***
## elev        -0.0051146  0.0009214  -5.551  0.00144 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.879 on 6 degrees of freedom
## Multiple R-squared:  0.837,Adjusted R-squared:  0.8099
```

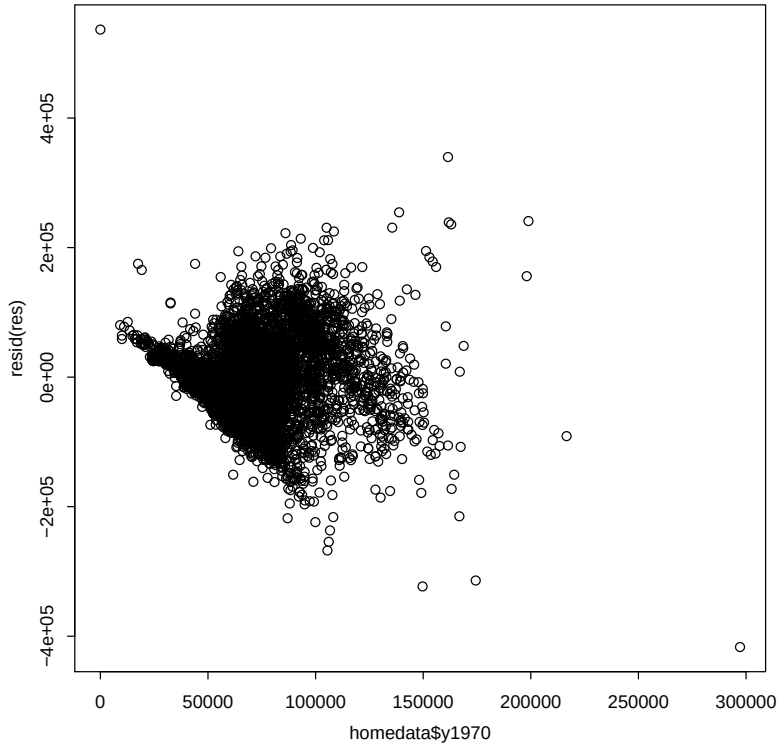
```
## F-statistic: 30.81 on 1 and 6 DF,  p-value: 0.001445

T <- (-0.005115 - (-0.00534)) / 0.000921
T

## [1] 0.2442997

2*pt(T, df=6, lower.tail=FALSE) # two-sided

## [1] 0.8151384
```

The p-value is not statistically significant.

11.10 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.

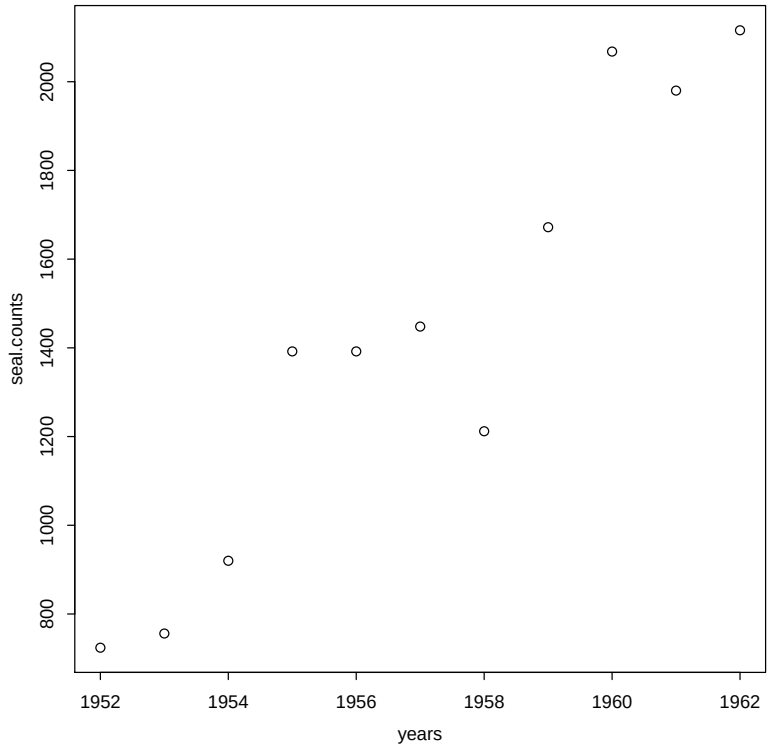
```
res <- lm(y2000 ~ y1970, data=homedata)
predict(res, newdata=dataframe(y1970=80000))

## Error in predict.lm(res, newdata = dataframe(y1970 = 80000)):
could not find function "dataframe"

predict(res, newdata=data.frame(y1970=80000))

##        1
## 316633.1

plot(resid(res))                  # simple plot
plot(homedata$y1970, resid(res))  # shows spread
```


resid(res)
4e+05
2e+05
0e+00
−2e+05
−4e+05
0
1000
2000
3000
4000
5000
6000
7000
Index


resid(res)
4e+05
2e+05
0e+00
−2e+05
−4e+05
0
50000
100000
150000
200000
250000
300000
homedata$y1970

11.11 Despite the common use of exponential models for population growth, a scatterplot reveals a linear-like trend for the years indicated. However, 20014 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
## 2280.727
```



11.12 Not very, as a plot of residuals will show.

11.13 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.

11.14 We have the following values:

```
sapply(c("100", "400", "10000"), function(distance) {
  res <- lm(scale(Time) ~ age, by.dist[[distance]], subset = age < 70)
  coef(res)[2]
})
```

```
##     100.age     400.age   10000.age
## 0.009362635 0.015577566 0.020083747
```

11.15 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)
```

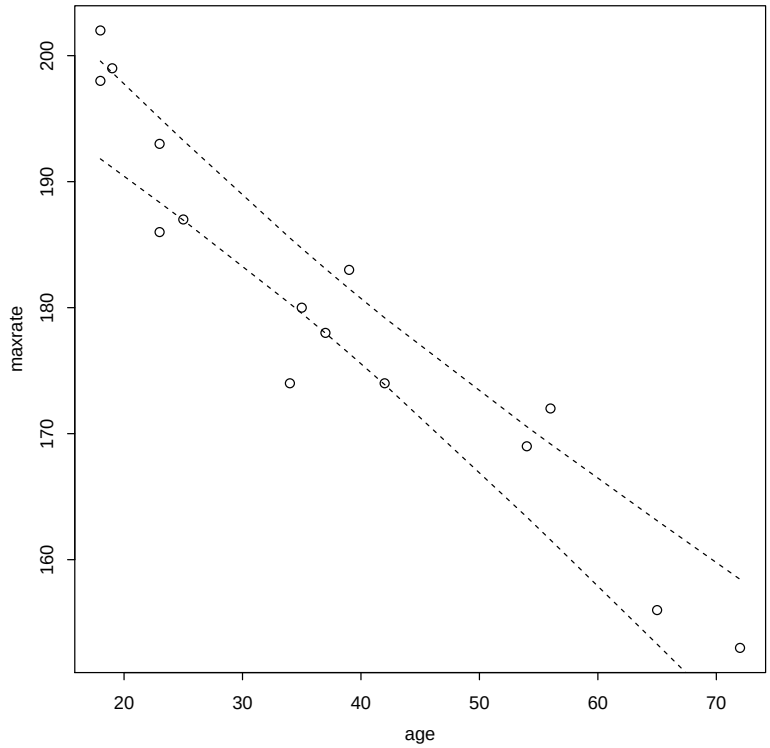
```
##
## Call:
## lm(formula = child ~ parent, data = galton)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.8050 -1.3661  0.0487  1.6339  5.9264
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 23.94153    2.81088   8.517   <2e-16 ***
## parent       0.64629    0.04114  15.711   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.239 on 926 degrees of freedom
## Multiple R-squared:  0.2105,Adjusted R-squared:  0.2096
## F-statistic: 246.8 on 1 and 926 DF,  p-value: < 2.2e-16
```

```
T <- (0.6463-1)/0.0411
2*pt(T, df=926)


## [1] 3.212167e-17
```

The small p-value cast doubt on the data being consistent with the null hypothesis.

11.16 If `res.mhr` stores the model, this can be done as follows:

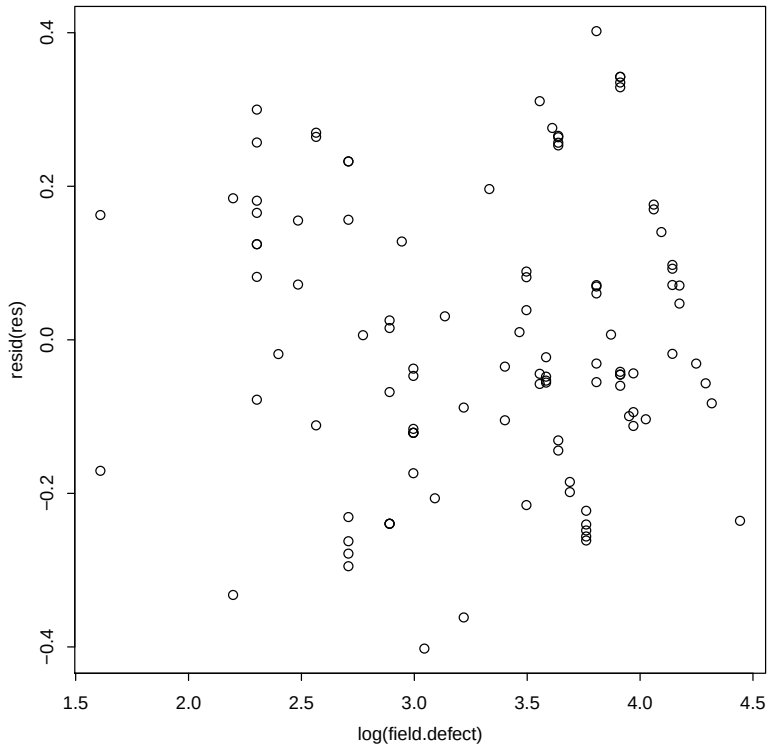
```
age.sort <- sort(unique(heartrate$age))
pred.res <- predict(res.mhr, newdata=data.frame(age=age.sort), int="conf")
plot(maxrate ~ age, heartrate); abline(res)
lines(age.sort,pred.res[,3], lty=2)
lines(age.sort,pred.res[,2], lty=2)
```



The only change is the abbreviated `int="cont"`.

11.17 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)
```



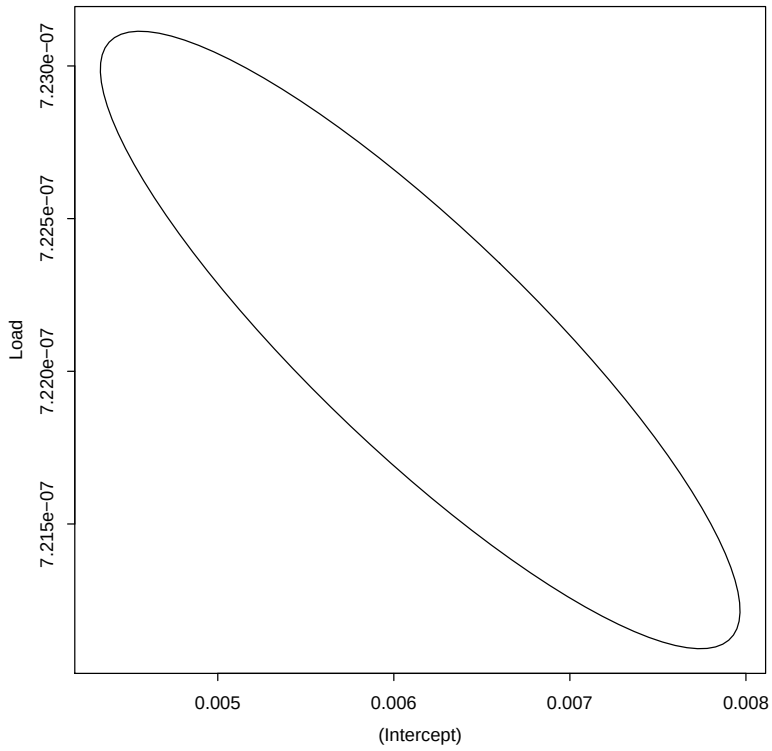
The scatterplot is consistent with the assumption that the error terms have a common distribution.

11.18 A simulation should show quite clearly that the two variables are negatively correlated.

11.19 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.

11.20 The corresponding p-value is for $\widehat{\beta}_1$. This test is a two-sided investigation of $H_0 : \widehat{\beta}_1 = 0$. If the slope were 0, then there would be no difference between the means of the two groups, which is the question being addressed in the two-sample t test.

11.21 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 48.254
## 2      2 12.732  1    35.522 5.5799  0.142
```

11.22 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.

11.23 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:

```
short_summary(res)
```

```
##                Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  -604374.61  245261.41 -2.4642   0.01393 *
## year            5224.52    1197.46  4.3630 1.444e-05 ***
## runs.scored     2787.20     269.58 10.3390 < 2.2e-16 ***
## wins            3088.57    2899.54  1.0652   0.28710
## games.behind  -15554.71    2562.93 -6.0691 1.952e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The only variable not flagged as significant is the number of wins.

11.24 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.

11.25 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   2803.37 25.3988 9.085e-07 ***
## 3    245 27880  1   2723.95 24.6792 1.274e-06 ***
## 4    244 26931  1    949.08  8.5988  0.003684 **
## ---
## 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.

11.26 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, trace=0)            # after library(MASS)


##
## Call:
## lm(formula = body.fat ~ age + weight + neck + abdomen + hip +
##     thigh + forearm + wrist, data = fat)
##
## Coefficients:
## (Intercept)          age       weight         neck      abdomen
##   -20.06213      0.05922     -0.08414     -0.43189      0.87721
##         hip        thigh      forearm        wrist
##    -0.18641      0.28644      0.48255     -1.40487
```

(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)])
```

11.27 Only the `Weight` variable is selected.

```
library(MASS)                 # load data set
res = lm(MPG.city ~ EngineSize + Weight + Passengers + Price, data=Cars93)
short_summary(res)


##               Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 46.3894133  2.0975157 22.1164 < 2.2e-16 ***
## EngineSize   0.1961189  0.5888804  0.3330    0.7399
## Weight      -0.0082072  0.0013429 -6.1115 2.633e-08 ***
## Passengers   0.2696217  0.4249510  0.6345    0.5274
## Price       -0.0358039  0.0491785 -0.7280    0.4685
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
stepAIC(res, trace=0)


##
## Call:
## lm(formula = MPG.city ~ Weight, data = Cars93)
##
## Coefficients:
## (Intercept)       Weight
##   47.048353    -0.008032
```

11.28 In a run of ten times, the full model was selected only six times.

11.29 This can be done as follows:

```
d <- with(baycheck, {
  n <- length(year)
  yt <- log(Nt[-1]/Nt[-n])
  nt <- Nt[-n]
  data.frame(yt, nt)
})
lm(yt ~ nt, data=d)


##
## Call:
## lm(formula = yt ~ nt, data = d)
##
## Coefficients:
## (Intercept)           nt
##   0.3458097   -0.0004088
```

So $\hat{r} = \widehat{\beta}_0 = 0.34$ and $\hat{K} = -\hat{r}/\widehat{\beta}_1 = 845.5$.

12

Analysis of variance

12.1 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 = 3.0061, num df = 4.000, denom df = 47.044, p-value =
## 0.02738
```

12.2 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 = 10.725, num df = 2.000, denom df = 22.527, 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.

12.3 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 = 11.794, 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.

12.4 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 = 26.465, num df = 6.0000, denom df = 8.7853, 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 = 6.0000, denom df = 6.9637, p-value =
## 0.1357
```

12.5 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 = 160.93, num df = 2.000, denom df = 88.485, p-value <
## 2.2e-16
```

12.6 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 = 18.715, num df = 3.0000, denom df = 7.9144, 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.

12.7 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 = 46.475, num df = 2.0000, denom df = 6.4165, p-value =
## 0.0001522
```

12.8 The test returns a small p-value:

```
kruskal.test(weight ~ group, PlantGrowth)$p.value
```

```
## [1] 0.01842376
```

12.9 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 = 1.7753, 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 = 0.36912, num df = 2.0000, denom df = 9.6449, p-value
## = 0.7007
```

Although the p-values are different, they both suggest that the data could have been produced under the null hypothesis. A boxplot of the data shows slightly skewed data, but nothing so skewed that one would expect a big difference between these two test due to a egregious difference from the assumptions of each.

12.10 We enter the data and then use `stack` to put it in the proper format:

```
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))
plot(values ~ ind, data=d, xlab = "test", ylab="grade")
```

The boxplots show that the assumption of independent samples from a common population, which perhaps is shifted, is appropriate.

The Kruskal–Wallis test returns

```
kruskal.test(values ~ ind, data=d)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  values by ind
## Kruskal-Wallis chi-squared = 1.7753, df = 2, p-value =
## 0.4116
```

This large p-value indicates no reason to doubt the null hypothesis of equally difficult exams.

12.11 The commands to perform this analysis are

```
short_summary(lm(attendance ~ league, data=MLBattend))
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1716321      36259 47.3345  < 2e-16 ***
## leagueNL      127296      52093  2.4436  0.01475 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see that the mean difference of 127,296 fans is significant with a p-value of 0.015.

12.12 Enforcing the speed limit does have an effect:

```
short_summary(lm(y ~ limit, data = Traffic))
```

```
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 23.13043    0.79896 28.9505 < 2.2e-16 ***
## limityes    -4.21739    1.30470 -3.2325  0.001457 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
short_summary(lm(y ~ factor(year), data = Traffic))
```

```
##                 Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)      22.76087    0.90972 25.0195  < 2e-16 ***
## factor(year)1962 -2.42391    1.28654 -1.8841  0.06115 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

12.13 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] 0.0001521839
```

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))


##
## Call:
## lm(formula = values ~ ind, data = wear.st)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.0000 -2.0833 -0.1667  1.5167 10.0000
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  297.800      2.205 135.077  < 2e-16 ***
## indtype2      22.367      2.985   7.493  7.3e-06 ***
## indtype3      19.200      3.307   5.806  8.4e-05 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.93 on 12 degrees of freedom
## Multiple R-squared:  0.838,Adjusted R-squared:  0.811
## F-statistic: 31.03 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] 1.810398e-05
```

12.14 This can be done with the commands

```
short_summary(lm(mpg ~ factor(cyl), data=mtcars))


##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept)   26.6636     0.9718 27.4373 < 2.2e-16 ***
## factor(cyl)6  -6.9208     1.5583 -4.4411 0.0001195 ***
## factor(cyl)8 -11.5636     1.2986 -8.9045 8.568e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

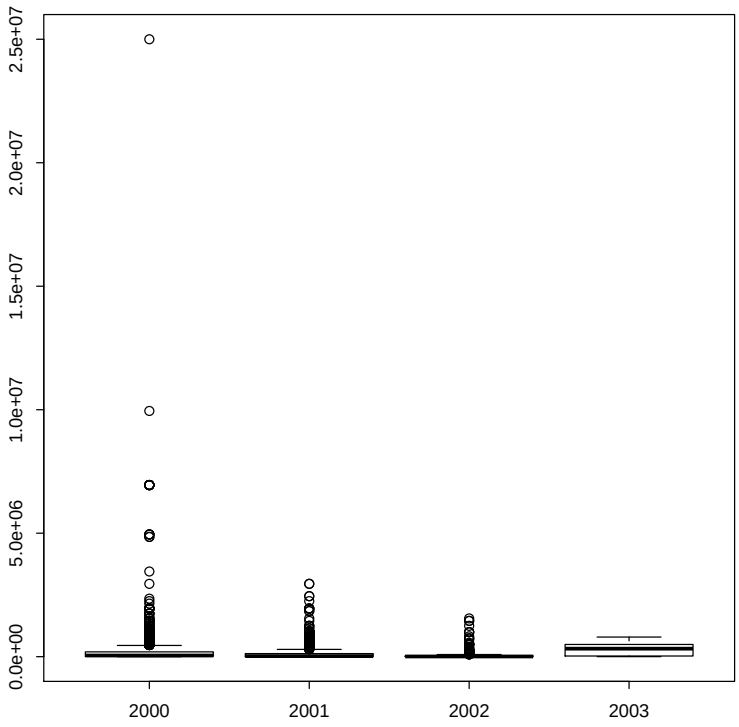
The `factor` function is used, as `cyl` is stored as a numeric variable.

12.15 This can be done as follows. The difference is significant.

```
short_summary(lm(mpg ~ factor(am), data=mtcars))


##                 Estimate Std. Error t value  Pr(>|t|)
## (Intercept)       17.1474     1.1246 15.2475 1.134e-15 ***
## factor(am)manual   7.2449     1.7644  4.1061  0.000285 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

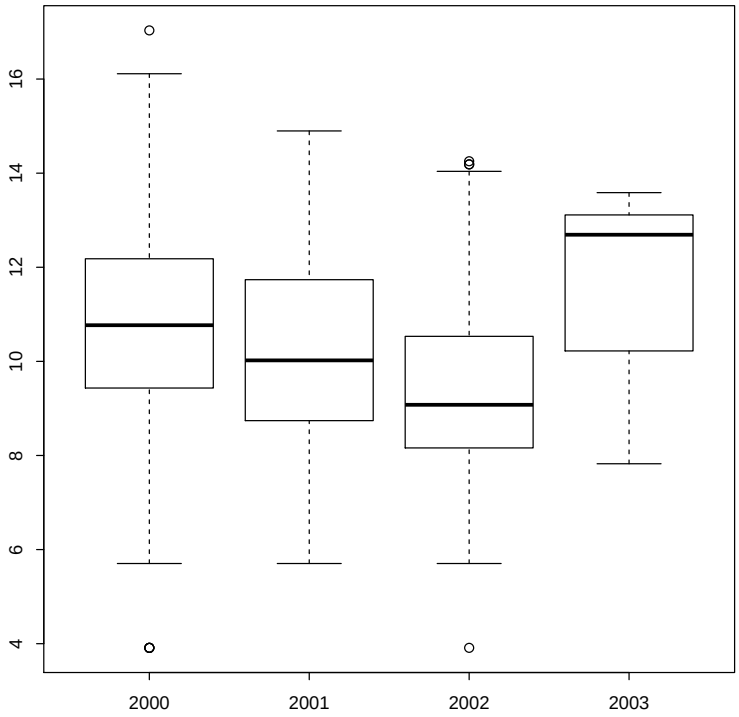
12.16 The boxplot of `amount` broken up by year shows skewed data.

```
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)
short_summary(res)
```

```
##                   Estimate Std. Error  t value  Pr(>|t|)
## (Intercept)      10.707768   0.027640 387.4048 < 2.2e-16 ***
## factor(year)2001 -0.487228   0.051887  -9.3902 < 2.2e-16 ***
## factor(year)2002 -1.285061   0.095546 -13.4497 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-value for the F-test is tiny, indicating that the null hypothesis is not likely to have yielded this data.

12.17 We need to compare the following

```
short_summary(lm(Speed ~ factor(Expt),data=morley))


##               Estimate Std. Error t value  Pr(>|t|)
## (Intercept)    909.000     16.599 54.7619 < 2.2e-16 ***
## factor(Expt)2  -53.000     23.475 -2.2577 0.0262509 *
## factor(Expt)3  -64.000     23.475 -2.7263 0.0076266 **
## factor(Expt)4  -88.500     23.475 -3.7700 0.0002835 ***
## factor(Expt)5  -77.500     23.475 -3.3014 0.0013561 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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     p adj
## 2-1 -53.0 -118.28006  12.280058 0.1679880
## 3-1 -64.0 -129.28006   1.280058 0.0574625
## 4-1 -88.5 -153.78006 -23.219942 0.0025733
## 5-1 -77.5 -142.78006 -12.219942 0.0115793
## 3-2 -11.0  -76.28006  54.280058 0.9899661
## 4-2 -35.5 -100.78006  29.780058 0.5571665
## 5-2 -24.5  -89.78006  40.780058 0.8343360
## 4-3 -24.5  -89.78006  40.780058 0.8343360
## 5-3 -13.5  -78.78006  51.780058 0.9784065
## 5-4  11.0  -54.28006  76.280058 0.9899661
```

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`.

12.18 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     p adj
## compact-SUV        -18.464286 -45.501378   8.5728067 0.3402130
## large-SUV          -17.047619 -46.814478  12.7192401 0.5428769
## midsize-SUV        -22.571429 -45.628738   0.4858814 0.0579194
## minivan-SUV        -18.714286 -48.481145  11.0525734 0.4350403
## pickup-SUV          39.785714   5.199749  74.3716792 0.0166036
## subcompact-SUV     -20.142857 -43.200167   2.9144528 0.1165746
## large-compact        1.416667 -31.529209  34.3625424 0.9999993
## midsize-compact     -4.107143 -31.144235  22.9299496 0.9988563
## minivan-compact     -0.250000 -33.195876  32.6958758 1.0000000
## pickup-compact      58.250000  20.892888  95.6071117 0.0006346
## subcompact-compact  -1.678571 -28.715664  25.3585210 0.9999938
## midsize-large       -5.523810 -35.290669  24.2430496 0.9965191
## minivan-large       -1.666667 -36.887289  33.5539560 0.9999988
## pickup-large        56.833333  17.455480  96.2111866 0.0016353
## subcompact-large    -3.095238 -32.862097  26.6716210 0.9998704
## minivan-midsize      3.857143 -25.909716  33.6240020 0.9995372
## pickup-midsize      62.357143  27.771178  96.9431077 0.0000864
## subcompact-midsize   2.428571 -20.628738  25.4858814 0.9998604
## pickup-minivan      58.500000  19.122147  97.8778533 0.0011587
## subcompact-minivan  -1.428571 -31.195431  28.3382877 0.9999987
## subcompact-pickup  -59.928571 -94.514536 -25.3426065 0.0001532
```

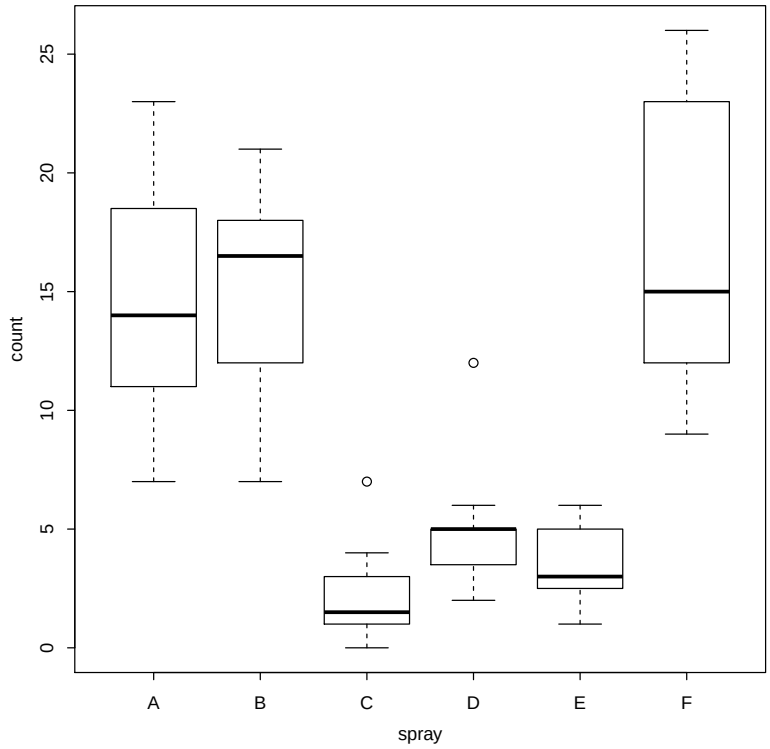
We can scan through the output or plot the results to find out which intervals do not straddle 0. A quick scan finds the following pairs involving pickups have statistically significant differences: `pickup-SUV`, `pickup-compact`, `pickup-large`, `pickup-midsize`, `pickup-minivan`, and `subcompact-pickup`.

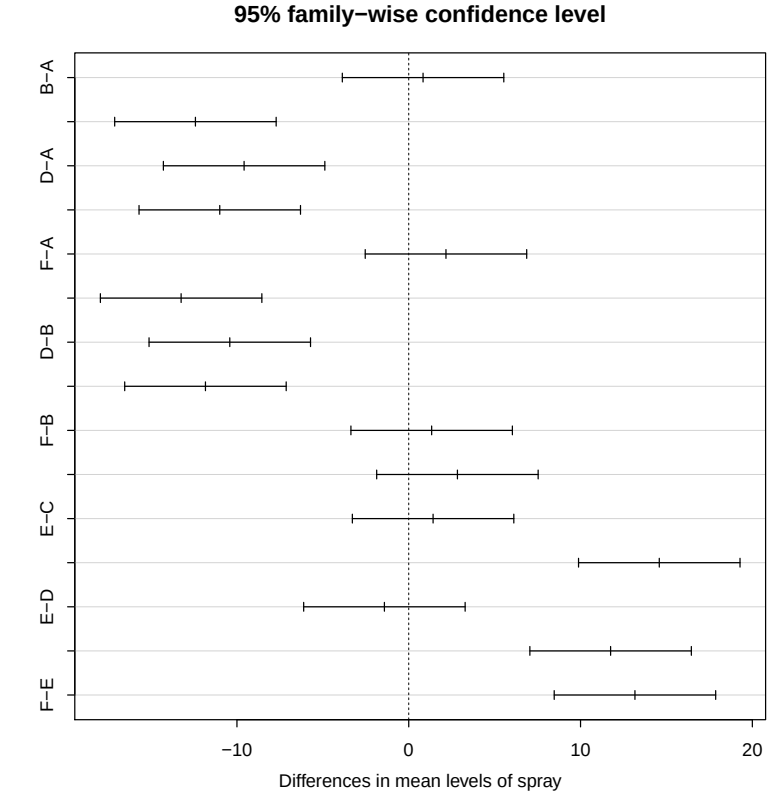
12.19 The following commands will show that `B` and `F` are:

```
plot(count ~ spray, data=InsectSprays)
res.aov <- aov(count ~ spray, data=InsectSprays)
summary(res.aov)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## spray        5   2669   533.8    34.7 <2e-16 ***
## Residuals   66   1015    15.4
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

plot(TukeyHSD(res.aov))
```


count
0
5
10
15
20
25
A
B
C
D
E
F
spray


95% family−wise confidence level
B−A
D−A
F−A
D−B
F−B
E−C
E−D
F−E
−10
0
10
20
Differences in mean levels of spray

12.20 The ANCOVA is performed as follows:

```
res <- lm(time ~ age + gender, data=nym.2002)
short_summary(res)

##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 241.45016    6.25031 38.6301 < 2.2e-16 ***
## age           1.22592    0.15483  7.9178 6.419e-15 ***
## genderMale  -29.39666    3.63343 -8.0906 1.716e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The difference is estimated to be a half-hour in total time.

12.21 The requested model is fit using `lm`.

```
res = lm(mpg ~ wt + factor(am), data=mtcars)
short_summary(res)

##                   Estimate Std. Error t value  Pr(>|t|)
## (Intercept)      37.321551   3.054638 12.2180 5.843e-13 ***
## wt               -5.352811   0.788244 -6.7908 1.867e-07 ***
## factor(am)manual -0.023615   1.545645 -0.0153    0.9879
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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.

12.22 The ANCOVA model is fit using `lm` as follows:

```
res <- lm(wt ~ gestation + wt1 + ht + factor(smoke), data=babies)
short_summary(res)

##                 Estimate Std. Error t value  Pr(>|t|)
## (Intercept)    84.4028290  7.5083897 11.2411 < 2.2e-16 ***
## gestation       0.0111011  0.0066690  1.6646   0.09625 .
## wt1            -0.0052558  0.0042302 -1.2424   0.21431
## ht              0.5588764  0.1192086  4.6882 3.064e-06 ***
```

```
## factor(smoke)1 -8.9419872  1.0976273 -8.1467 9.127e-16 ***
## factor(smoke)2  0.6044698  1.9506106  0.3099   0.75670
## factor(smoke)3  0.9465991  1.8912498  0.5005   0.61680
## factor(smoke)9  3.9580598  5.5949594  0.7074   0.47943
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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.

12.23 We fit the ANCOVA model first:

```
kid.weights$BMI = with(kid.weights, (weight/2.2) / (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 12911
## 2    248 12911 -1  -0.33189 0.0063 0.9366

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 12911
## 2    248 12960 -1   -48.791 0.9334 0.3349
```

We see that neither variable is significant by `stepAIC`:

```
require(MASS)
stepAIC(res)


## Start:  AIC=7087.79
## wt ~ gestation + wt1 + ht + factor(smoke)
##
##                 Df Sum of Sq    RSS    AIC
## - wt1            1     474.4 377907 7087.3
## <none>                       377433 7087.8
## - gestation      1     851.6 378284 7088.6
## - ht             1    6755.5 384188 7107.7
## - factor(smoke)  4   25053.9 402487 7159.2
##
## Step:  AIC=7087.34
## wt ~ gestation + ht + factor(smoke)
##
##                 Df Sum of Sq    RSS    AIC
## <none>                       377907 7087.3
## - gestation      1     831.3 378739 7088.1
## - ht             1    7473.5 385381 7109.5
## - factor(smoke)  4   25128.5 403036 7158.9
##
## Call:
## lm(formula = wt ~ gestation + ht + factor(smoke), data = babies)
##
## Coefficients:
##    (Intercept)       gestation              ht  factor(smoke)1
##       89.37866         0.01097         0.46988        -8.93897
## factor(smoke)2  factor(smoke)3  factor(smoke)9
##        0.55280         1.07946         4.01091
```

12.24 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   13.85 13.8518  12.299 0.0004737 ***
## factor(EDUC)  16  275.42 17.2140  15.284 < 2.2e-16 ***
```

```
## Residuals    982 1105.97  1.1262
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

12.25 The ANCOVA model is fit as follows:

```
res = lm(temperature ~ hr + factor(gender), data=normtemp)
short_summary(res)


##                   Estimate Std. Error  t value  Pr(>|t|)
## (Intercept)     96.2508140  0.6487172 148.3710 < 2.2e-16 ***
## hr               0.0252667  0.0087619   2.8837  0.004618 **
## factor(gender)2  0.2694061  0.1232772   2.1854  0.030696 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Both variables are flagged as significant at the 0.05 level.

12.26 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 <- gl(2, 9, 18, c("N", "Y"))
tv <- gl(3, 6, 18, labels=c(0, "1-2", "3+"))

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)


##
## Call:
## lm(formula = likability ~ web)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4444 -2.0278 -0.8333  1.7222  4.5556
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.4444     0.8731    2.80   0.0129 *
## webY          0.7778     1.2348    0.63   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.619 on 16 degrees of freedom
## Multiple R-squared:  0.0242,Adjusted R-squared:  -0.03679
## F-statistic: 0.3968 on 1 and 16 DF,  p-value: 0.5377
```

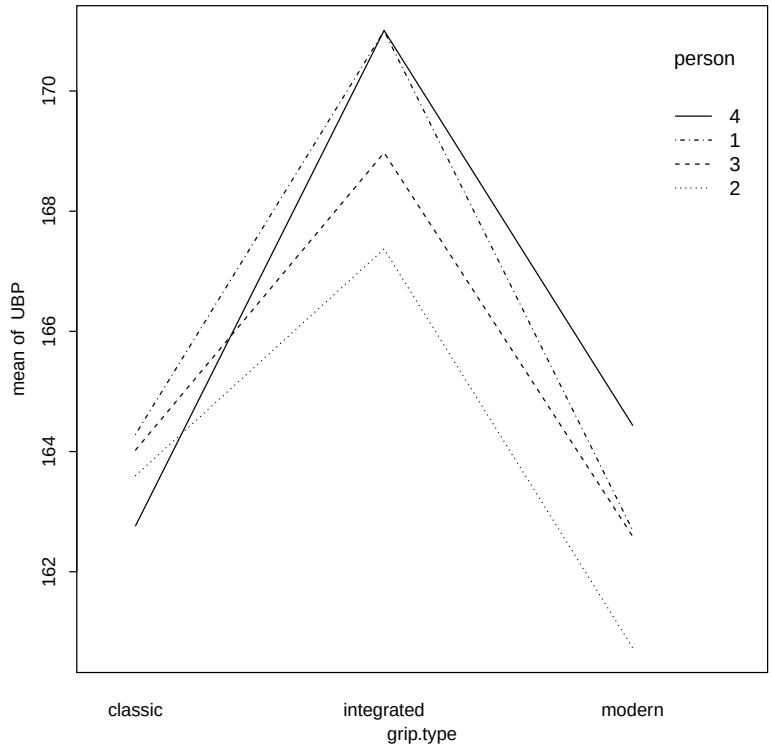
Web advertising is not effective in this assessment. However, controlling first for television exposure we have

```
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 86.167
## 2     14 69.500  1    16.667 3.3573 0.08826 .
## ---
## 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.

12.27 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     30 509.01
## 2     24 483.86  6     25.15 0.2079 0.9709
```

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     35 875.55
## 2     32 848.19  3     27.35 0.3439 0.7937
```

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     35 875.55
## 2     33 536.36  2    339.18 10.434 0.0003079 ***
## ---
## 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, verbose=FALSE)
stepAIC(res.int)
```

```
## Start:  AIC=117.54
## UBP ~ person * grip.type
##
##                     Df Sum of Sq    RSS    AIC
## - person:grip.type  6     25.15 509.01 107.36
## <none>                          483.86 117.54
##
## Step:  AIC=107.36
## UBP ~ person + grip.type
##
##             Df Sum of Sq    RSS    AIC
## - person     3     27.35 536.36 103.25
## <none>                   509.01 107.36
```

```
## - grip.type  2    339.18 848.19 121.75
##
## Step:  AIC=103.25
## UBP ~ grip.type
##
##             Df Sum of Sq    RSS    AIC
## <none>                   536.36 103.25
## - grip.type  2    339.18 875.55 116.89
##
## Call:
## lm(formula = UBP ~ grip.type, data = grip)
##
## Coefficients:
##         (Intercept)  grip.typeintegrated      grip.typemodern
##             163.669                5.919               -1.055
```

12.28 The two-way ANOVA with interaction is fit as follows:

```
res.full = lm(mpg ~ factor(cyl) * factor(am), data=mtcars)
short_summary(res.full)


##                              Estimate Std. Error t value
## (Intercept)                   22.9000     1.7507 13.0807
## factor(cyl)6                  -3.7750     2.3159 -1.6300
## factor(cyl)8                  -7.8500     1.9573 -4.0106
## factor(am)manual               5.1750     2.0528  2.5209
## factor(cyl)6:factor(am)manual -3.7333     3.0948 -1.2063
## factor(cyl)8:factor(am)manual -4.8250     3.0948 -1.5591
##                               Pr(>|t|)
## (Intercept)                  6.057e-13 ***
## factor(cyl)6                 0.1151546
## factor(cyl)8                 0.0004548 ***
## factor(am)manual             0.0181761 *
## factor(cyl)6:factor(am)manual 0.2385526
## factor(cyl)8:factor(am)manual 0.1310693
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The shortende output shows that the two interactions are not flagged as statistically significant. We refit the models as follows:

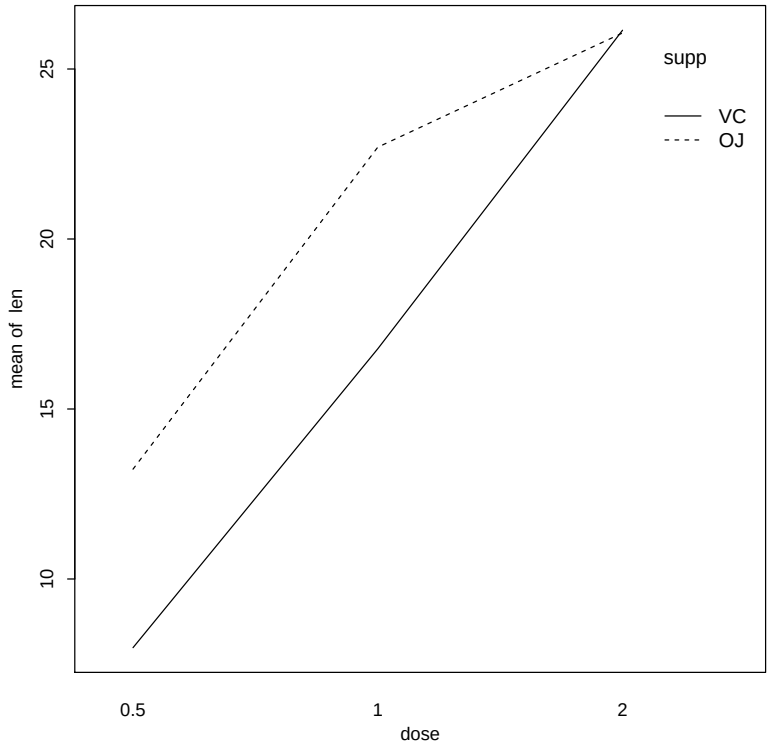
```
res.add <- lm(mpg ~ factor(cyl) + factor(am), data=mtcars)
res.cyl <- lm(mpg ~ factor(cyl),              data=mtcars)
res.am <- lm(mpg ~ factor(am),                data=mtcars)
anova(res.am, res.add)

## 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     30 720.9
## 2     28 264.5  2     456.4 24.158 8.01e-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     29 301.26
## 2     28 264.50  1    36.767 3.8922 0.05846 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The F-tests indicate that both variables are significant.

12.29 We proceed as follows:

```
with(ToothGrowth, interaction.plot(dose, supp, len))
res.full <- lm(len ~ supp + dose, data=ToothGrowth)
res.add  <- lm(len ~ supp * dose, data=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 1022.56
```

```
## 2     56  933.63  1     88.92 5.3335 0.02463 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



12.30 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   -1247.4 0.4338 0.8761
```

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.

12.31 The coefficients do get estimated, and should be easy enough to understand. However, as there is insufficient data to estimate the residual variance, the statistical significance of the estimates can not be performed. R responds by putting `NA` for the values that can't be estimated.

13

Extensions of linear model

13.1 The logistic regression is carried out as follows:

```
res <- glm(enjoyed ~ gender + age, data=tastesgreat,
           family=binomial)
summary(res)
```

```
##
## Call:
## glm(formula = enjoyed ~ gender + age, family = binomial, data = tastesgreat)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.84192  -0.88512  -0.06624   0.74655   2.55961
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.18443    3.09644  -2.643  0.00821 **
## genderMale   2.42241    0.95590   2.534  0.01127 *
## age          0.16491    0.06519   2.530  0.01142 *
## ---
## 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: 44.981
##
## Number of Fisher Scoring iterations: 5
```

This shows that both are flagged as significant.

13.2 This is performed with these commands:

```
library(MASS)                 # load stepAIC
stepAIC(glm(healthy ~ p + g, healthy, family=binomial))


## Start:  AIC=28.97
## healthy ~ p + g
##
##        Df Deviance    AIC
## - g     1   24.840 28.840
## <none>      22.971 28.971
## - p     1   28.945 32.945
##
## Step:  AIC=28.84
## healthy ~ p
##
##        Df Deviance    AIC
## <none>      24.840 28.840
## - p     1   30.885 32.885
##
## Call:  glm(formula = healthy ~ p, family = binomial, data = healthy)
##
## Coefficients:
## (Intercept)            p
##      -6.845        1.827
##
## Degrees of Freedom: 31 Total (i.e. Null);  30 Residual
## Null Deviance:     30.88
## Residual Deviance: 24.84  AIC: 28.84
```

The preferred model by the AIC criteria involves `p` but not `g`.

13.3 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)


##
## Call:
## glm(formula = low ~ age + lwt + smoke + ht + ui, family = binomial,
```

```
##     data = birthwt)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6549  -0.8356  -0.6462   1.1251   1.9925
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.399794   1.080407   1.296   0.1951
## age         -0.034073   0.033674  -1.012   0.3116
## lwt         -0.015447   0.006587  -2.345   0.0190 *
## smoke        0.647540   0.336650   1.923   0.0544 .
## ht           1.893274   0.683392   2.770   0.0056 **
## ui           0.884607   0.444051   1.992   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: 223.78
##
## 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, trace=0)


##
## Call:  glm(formula = low ~ lwt + smoke + ht + ui, family = binomial,
##     data = birthwt)
##
## Coefficients:
## (Intercept)          lwt        smoke           ht           ui
##     0.72186     -0.01631      0.65300      1.92213      0.89627
##
## Degrees of Freedom: 188 Total (i.e. Null);  184 Residual
## Null Deviance:     234.7
## Residual Deviance: 212.8  AIC: 222.8
```

13.4 The modeling is done using `glm` with the `family="binomial"` argument.

```
hfm <- hall.fame$Hall.Fame.Membership != "not a member"
res <- glm(hfm ~ BA + HR + hits + games, data=hall.fame,
  family="binomial")
stepAIC(res, trace=0)

##
## Call:  glm(formula = hfm ~ BA + HR + hits, family = "binomial", data = hall.fame)
##
## Coefficients:
## (Intercept)           BA           HR         hits
##  -21.843762    51.212224     0.004285     0.002495
##
## Degrees of Freedom: 1339 Total (i.e. Null);  1336 Residual
## Null Deviance:     826.4
## Residual Deviance: 380.4  AIC: 388.4
```

The confounded variable `games` is dropped by this criteria.

13.5 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] 236.9645

AIC(res.add)

## [1] 225.454
```

13.6 This can be done by first plotting to guess initial estimates.

```
plot(circumference ~ age, data=Orange, subset = Tree == 1)
```

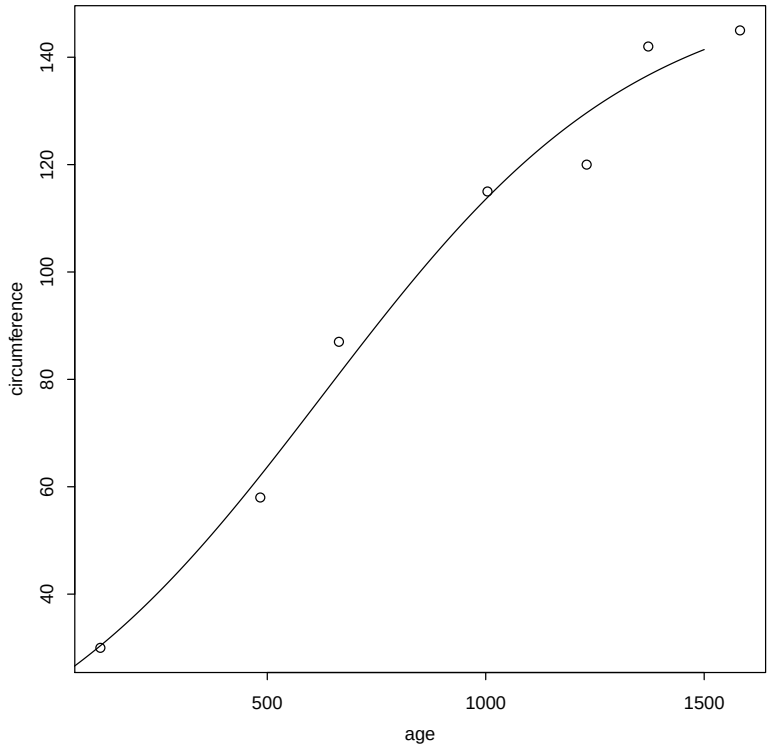
From this graphic, we estimate `Y` to be 150, `k` to be about 1/300, and `t0` to be 750. (See the help page `?Orange` to learn how to use the self-starting function `SSlogis` to do this.)

```
g <- function(t, Y, k, t_0) Y*( 1 + exp(-k*(t-t_0)))^(-1)
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
##
## Number of iterations to convergence: 5
## Achieved convergence tolerance: 9.188e-06
```

We can add this curve to our graph with

```
age <- 0:1500
plot(circumference ~ age, data=Orange, subset = Tree == 1)
lines(age, predict(res.tree, data.frame(age=age)))
```



13.7 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
## 937.02199   0.08768  35.22282
##  residual sum-of-squares: 76.66
##
## Number of iterations to convergence: 7
## Achieved convergence tolerance: 1.042e-06
```

Adding the curve to the graph can be done with curve.

```
curve(f(x, Y=937, k=.08768, t0=35.22270), add=T)
```

13.8 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> wtloss.fm <- nls(Weight ~ b0 + b1*2^(-Days/th),
## wtloss+     data = wtloss, start = list(b0=90, b1=95, th=120))
##
## 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
##
## Number of iterations to convergence: 3
## Achieved convergence tolerance: 4.324e-06
##
## wtloss> plot(wtloss)
##
## wtloss> with(wtloss, lines(Days, fitted(wtloss.fm)))


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
##
## Number of iterations to convergence: 3
## Achieved convergence tolerance: 4.324e-06


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] 98.64172
```

(Of course, predictions beyond the range of the data are not a good idea.)

13.9 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] 308.6806


AIC(res.l1)


## [1] 378.1829
```

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.)

13.10 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
##
## Number of iterations to convergence: 4
## Achieved convergence tolerance: 1.602e-06

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
##
## Number of iterations to convergence: 5
## Achieved convergence tolerance: 1.087e-06

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
##
```

```
## Number of iterations to convergence: 8
## Achieved convergence tolerance: 5.472e-06
```

The Accord has the smallest decay rate, and the Taurus the largest.

14

Thanks

Thanks to Professor William Kitto from Antelope Valley College for pointing out some errors. (8/2015)