

# R Programming

## Table of Contents

Pre-requisites .....	7
Part 1 .....	8
Chapter 2: Numerics, Arithmetic, Assignment, And Vectors .....	8
Exercise 2.1.....	8
Solution 2.1.....	8
Exercise 2.2.....	10
Solution 2.2.....	10
Exercise 2.3.....	11
Solution 2.3.....	11
Exercise 2.4.....	12
Solution 2.4.....	12
Exercise 2.5.....	14
Solution 2.5.....	14
Chapter 3: Matrices and Arrays.....	15
Exercises 3.1.....	15
Solution 3.1.....	15
Exercise 3.2.....	17
Solution 3.2.....	18
Exercise 3.3.....	20
Solution 3.3.....	20
Chapter 4: Non-Numeric Values.....	26
Exercise 4.1.....	26
Solution 4.1.....	26
Exercise 4.2.....	30
Solution 4.2.....	30
Exercise 4.3.....	32
Solution 4.3.....	32
Exercise 4.4.....	34
Solution 4.4.....	34
Exercise 4.5.....	36
Solution 4.5.....	36
Chapter 5: Lists and Dataframes.....	39
Exercise 5.1.....	39
Solution 5.1.....	39

<b>Exersise 5.2.....</b>	41
<b>Solution 5.2.....</b>	42
<b>Chapter 6: Special Values, Classes, Coercion.....</b>	48
<b>Exercise 6.1.....</b>	48
<b>Solution 6.1.....</b>	48
<b>Exercise 6.2.....</b>	50
<b>Solution 6.2.....</b>	50
<b>Exercise 6.3.....</b>	52
<b>Solution 6.3.....</b>	53
<b>Chapter 7: Basic Plotting.....</b>	58
<b>Exercise 7.1.....</b>	58
<b>Solution 7.1.....</b>	59
<b>Chapter 8: Reading and Writing Files .....</b>	63
<b>Exercise 8.1.....</b>	63
<b>Solution 8.1.....</b>	64
<b>Part 2 .....</b>	67
<b>Chapter 9: Calling Functions .....</b>	67
<b>Exercise 9.1.....</b>	67
<b>Solution 9.1.....</b>	67
<b>Exercise 9.2.....</b>	68
<b>Solution 9.2.....</b>	68
<b>Chapter 10: Conditions and Loops .....</b>	70
<b>Exercise 10.1.....</b>	70
<b>Solution 10.1.....</b>	71
<b>Exercise 10.2.....</b>	73
<b>Solution 10.2.....</b>	75
<b>Exercise 10.3.....</b>	77
<b>Solution 10.3.....</b>	77
<b>Exercise 10.4.....</b>	80
<b>Solution 10.4.....</b>	82
<b>Exercise 10.5.....</b>	87
<b>Solution 10.5.....</b>	87
<b>Exercise 10.6.....</b>	91
<b>Solution 10.6.....</b>	92
<b>Chapter 11: Writing Functions .....</b>	99
<b>Exercise 11.1.....</b>	99

<b>Solution 11.1</b>	100
<b>Exercise 11.2</b>	102
<b>Solution 11.2</b>	104
<b>Exercise 11.3</b>	108
<b>Solution 11.3</b>	110
<b>Chapter 12: Exceptions, Timings, and Visibility</b>	112
<b>Exercise 12.1</b>	112
<b>Solution 12.1</b>	114
<b>Exercise 12.2</b>	119
<b>Solution 12.2</b>	119
<b>Part 3</b>	122
<b>Chapter 13: Elementary Statistics</b>	122
<b>Exercise 13.1</b>	122
<b>Solution 13.1</b>	122
<b>Exercise 13.2</b>	124
<b>Solution 13.2</b>	124
<b>Exercise 13.3</b>	126
<b>Solution 13.3</b>	126
<b>Exercise 13.4</b>	128
<b>Solution 13.4</b>	129
<b>Chapter 14: Basic Data Visualization</b>	134
<b>Exercise 14.1</b>	134
<b>Solution 14.1</b>	136
<b>Chapter 15: Probability</b>	147
<b>Exercise 15.1</b>	147
<b>Solution 15.1</b>	147
<b>Exercise 15.2</b>	149
<b>Solution 15.2</b>	150
<b>Chapter 16: Common Probability Distributions</b>	154
<b>Exercise 16.1</b>	154
<b>Solution 16.1</b>	155
<b>Exercise 16.2</b>	157
<b>Solution 16.2</b>	157
<b>Exercise 16.3</b>	158
<b>Solution 16.3</b>	158
<b>Exercise 16.4</b>	164

<b>Solution 16.4</b>	164
<b>Exercise 16.5</b>	170
<b>Solution 16.5</b>	170
<b>PART 4</b>	174
<b>Chapter 17: Sampling Distributions and Confidence</b>	174
<b>Exercise 17.1</b>	174
<b>Solution 17.1</b>	175
<b>Exercise 17.2</b>	177
<b>Solution 17.2</b>	178
<b>Chapter 18: Hypothesis Testing</b>	182
<b>Exercise 18.1</b>	182
<b>Solution 18.1</b>	182
<b>Exercise 18.2</b>	184
<b>Solution 18.2</b>	185
<b>Exercise 18.3</b>	190
<b>Solution 18.3</b>	191
<b>Exercise 18.4</b>	196
<b>Solution 18.4</b>	196
<b>Exercise 18.5</b>	198
<b>Solution 18.5</b>	199
<b>Exercise 18.6</b>	201
<b>Solution 18.6</b>	201
<b>Chapter 19: Analysis of Variance</b>	208
<b>Exercise 19.1</b>	208
<b>Solution 19.1</b>	208
<b>Exercise 19.2</b>	220
<b>Solution 19.2</b>	220
<b>Chapter 20: Simple Linear Regression</b>	223
<b>Exercise 20.1</b>	223
<b>Solution 20.1</b>	224
<b>Exercise 20.2</b>	234
<b>Solution 20.2</b>	235
<b>Chapter 21: Multiple Linear Regression</b>	242
<b>Exercise 21.1</b>	242
<b>Solution 21.1</b>	244
<b>Exercise 21.2</b>	258

<b>Solution 21.2</b>	259
<b>Exercise 21.3</b>	273
<b>Solution 21.3</b>	273
<b>Chapter 22: Linear Model Selection and Diagnostics</b>	282
<b>Exercise 22.1</b>	282
<b>Solution 22.1</b>	283
<b>Exercise 22.2</b>	305
<b>Solution 22.2</b>	306
<b>PART 5</b>	332
<b>Chapter 23: Advanced Plot Customization</b>	332
<b>Exercise 23.1</b>	332
<b>Solution 23.1</b>	334
<b>Exercise 23.2</b>	345
<b>Solution 23.2</b>	347
<b>Chapter 24: Going Further with the Grammar of Graphics</b>	352
<b>Exercise 24.1</b>	352
<b>Solution 24.1</b>	354
<b>Exercise 24.2</b>	361
<b>Solution 24.2</b>	362
<b>Chapter 25: Defining Colors and Plotting in Higher Dimensions</b>	368
<b>Exercise 25.1</b>	368
<b>Solution 25.1</b>	370
<b>Exercise 25.2</b>	375
<b>Solution 25.2</b>	375
<b>Exercise 25.3</b>	378
<b>Solution 25.3</b>	381
<b>Exercise 25.4</b>	390
<b>Solution 25.4</b>	392
<b>Exercise 25.5</b>	397
<b>Solution 25.5</b>	398
<b>Chapter 26: Interactive 3D Plots</b>	405
<b>Exercise 26.1</b>	405
<b>Solution 26.1</b>	407
<b>Exercise 26.2</b>	410
<b>Exercise 26.2</b>	412
<b>Exercise 26.3</b>	419

## Pre-requisites

Before start you need to install next packages by using command

`install.packages("package name"):`

`car, ggplot2, faraway, boot, spatstat, rgl, ggsave, shape, scatterplot3d, gridExtra, ggvis`

## Part 1

# Chapter 2: Numerics, Arithmetic, Assignment, And Vectors

Estimated time to complete: **30 minutes**

### Exercise 2.1

a. Using R, verify that

$$\frac{(6a + 42)}{(34.2 - 3.62)} = 29.50556$$

when  $a = 2.3$ .

b. Which of the following squares negative 4 and adds 2 to the result?

- i.  $(-4)^2+2$
- ii.  $-4^2+2$
- iii.  $(-4)^{(2+2)}$
- iv.  $-4^{(2+2)}$

c. Using R, how would you calculate the square root of half of the average of the numbers 25.2, 15, 16.44, 15.3, and 18.6?

d. Find  $\log_e 0.3$

e. Compute the exponential transform of your answer to (d).

f. Identify R's representation of  $-0.0000000423546322$  when printing this number to the console.

### Solution 2.1

#(a)

R> (6\*2.3+42)/(34.2-3.62)

[1] 1.824722

#(b)

R> (-4)^2+2

[1] 18

#(c)

R>sqrt(x=0.5\*((25.2+15+16.44+15.3+18.6)/5))

[1] 3.008987

```
#(d)
R>log(x=0.3)
[1] -1.203973
#(e)
R>exp(x=-1.203973)
[1] 0.2999999
#(f)
R>-0.0000000423546322
[1] -4.235463e-09
```

## Exercise 2.2

- a. Create an object that stores the value  $3^2 \times 4^{1/8}$
- b. Overwrite your object in (a) by itself divided by 2.33. Print the result to the console.
- c. Create a new object with the value  $-8.2 \times 10^{-13}$ .
- d. Print directly to the console the result of multiplying (b) by (c).

## Solution 2.2

#(a)

```
R>foo <- 3^2*4^(1/8)
```

#(b)

```
R>foo <- foo/2.33
```

```
R>foo
```

```
[1] 4.593504
```

#(c)

```
R>bar <- -8.2e-13
```

#(d)

```
R>foo*bar
```

```
[1] -3.766673e-12
```

## Exercise 2.3

- a. Create and store a sequence of values from 5 to  $-11$  that progresses in steps of 0.3.
  - b. Overwrite the object from (a) using the same sequence with the order reversed.
  - c. Repeat the vector  $c(-1,3,-5,7,-9)$  twice, with each element repeated 10 times, and store the result. Display the result sorted from largest to smallest.
  - d. Create and store a vector that contains, in any configuration, the following:
    - i. A sequence of integers from 6 to 12 (inclusive)
    - ii. A threefold repetition of the value 5.3
    - iii. The number  $-3$
    - iv. A sequence of nine values starting at 102 and ending at the number that is the total length of the vector created in (c)
  - e. Confirm that the length of the vector created in (d) is 20.

## Solution 2.3

## Exercise 2.4

- a. Create and store a vector that contains the following, in this order:
  - i. A sequence of length 5 from 3 to 6 (inclusive)
  - ii. A twofold repetition of the vector  $c(2, -5.1, -33)$
  - iii. The value  $\frac{7}{42} + 2$
- b. Extract the first and last elements of your vector from (a), storing them as a new object.
- c. Store as a third object the values returned by omitting the first and last values of your vector from (a).
- d. Use only (b) and (c) to reconstruct (a).
- e. Overwrite (a) with the same values sorted from smallest to largest.
- f. Use the colon operator as an index vector to reverse the order of (e), and confirm this is identical to using *sort* on (e) with *decreasing=TRUE*.
- g. Create a vector from (c) that repeats the third element of (c) three times, the sixth element four times, and the last element once.
- h. Create a new vector as a copy of (e) by assigning (e) as is to a newly named object. Using this new copy of (e), overwrite the first, the fifth to the seventh (inclusive), and the last element with the values 99 to 95 (inclusive), respectively.

## Solution 2.4

#(a)

```
R>foo <- c(seq(from=3,to=6,length.out=5),rep(c(2,-5.1,-33),times=2),7/42+2)
R>foo
[1] 3.000000 3.750000 4.500000 5.250000 6.000000 2.000000 -5.100000 -33.000000
2.000000 -5.100000 -33.000000 2.166667
```

#(b)

```
R>bar <- foo[c(1,length(x=foo))]
```

```
R>bar
```

```
[1] 3.000000 2.166667
```

#(c)

```
R>baz <- foo[-c(1,length(x=foo))]
```

```
R>baz
```

```
[1] 3.75 4.50 5.25 6.00 2.00 -5.10 -33.00 2.00 -5.10 -33.00
```

#(d)

```
R>c(bar[1],baz,bar[2])
```

```
[1] 3.000000 3.750000 4.500000 5.250000 6.000000 2.000000 -5.100000 -33.000000
2.000000 -5.100000 -33.000000 2.166667
```

#(e)

```
R>foo <- sort(x=foo,decreasing=FALSE)
```

```
R>foo
```

```
[1] -33.000000 -33.000000 -5.100000 -5.100000 2.000000 2.000000 2.166667 3.000000
3.750000 4.500000 5.250000 6.000000
```

```
#(f)
R>foo[length(x=foo):1]
R>sort(x=foo,decreasing=TRUE)
[1] 6.000000 5.250000 4.500000 3.750000 3.000000 2.166667 2.000000 2.000000 -
5.100000 -5.100000 -33.000000 -33.000000

#g)
R>baz[c(rep(x=3,times=3),rep(x=6,times=4),length(x=baz))]
[1] 5.25 5.25 5.25 -5.10 -5.10 -5.10 -5.10 -33.00
#(h)
R>qux <- foo
R>qux[c(1,5:7,12)] <- 99:95
R>qux
[1] 99.00 -33.00 -5.10 -5.10 98.00 97.00 96.00 3.00 3.75 4.50 5.25 95.00
```

## Exercise 2.5

- a. Convert the vector  $c(2,0.5,1,2,0.5,1,2,0.5,1)$  to a vector of only 1s, using a vector of length 3.
- b. The conversion from a temperature measurement in degrees Fahrenheit F to Celsius C is performed using the following equation:
$$C = \frac{5}{9}(F - 32)$$
Use vector-oriented behavior in R to convert the temperatures 45, 77, 20, 19, 101, 120, and 212 in degrees Fahrenheit to degrees Celsius.
- c. Use the vector  $c(2,4,6)$  and the vector  $c(1,2)$  in conjunction with rep and \* to produce the vector  $c(2,4,6,4,8,12)$ .
- d. Overwrite the middle four elements of the resulting vector from (c) with the two recycled values -0.1 and -100, in that order.

## Solution 2.5

#(a)

```
R>c(2,0.5,1,2,0.5,1,2,0.5,1)/c(2,0.5,1)
```

```
[1] 1 1 1 1 1 1 1 1 1
```

#(b)

```
R>farens <- c(45,77,20,19,101,120,212)
```

```
R>cel <- 5/9*(farens-32)
```

```
R>cel
```

```
[1] 7.222222 25.000000 -6.666667 -7.222222 38.333333 48.888889 100.000000
```

#(c)

```
R>foo <- rep(x=c(2,4,6),times=2)*rep(x=c(1,2),each=3)
```

```
R>foo
```

```
[1] 2 4 6 4 8 12
```

#(d)

```
R>foo[2:5] <- c(-0.1,-100)
```

```
R>foo
```

```
[1] 2.0 -0.1 -100.0 -0.1 -100.0 12.0
```

# Chapter 3: Matrices and Arrays

Estimated time to complete: **30 minutes**

## Exercises 3.1

- a. Construct and store a  $4 \times 2$  matrix that's filled row-wise with the values 4.3, 3.1, 8.2, 8.2, 3.2, 0.9, 1.6, and 6.5, in that order.
- b. Confirm the dimensions of the matrix from (a) are  $3 \times 2$  if you remove any one row.
- c. Overwrite the second column of the matrix from (a) with that same column sorted from smallest to largest.
- d. What does R return if you delete the fourth row and the first column from (c)? Use matrix to ensure the result is a single-column matrix, rather than a vector.
- e. Store the bottom four elements of (c) as a new  $2 \times 2$  matrix.
- f. Overwrite, in this order, the elements of (c) at positions (4,2), (1,2), (4,1), and (1,1) with one of the two values on the diagonal of (e).>

## Solution 3.1

```
#(a)
> mymat <- matrix(data=c(4.3,3.1,8.2,8.2,3.2,0.9,1.6,6.5),nrow=4,ncol=2,byrow=TRUE)

> mymat
 [,1] [,2]
[1,] 4.3  3.1
[2,] 8.2  8.2
[3,] 3.2  0.9
[4,] 1.6  6.5

> #(b)

> dim(mymat[-2,])
[1] 3 2

> #(c)

> mymat[,2] <- sort(x=mymat[,2])

> mymat
 [,1] [,2]
[1,] 4.3  0.9
[2,] 8.2  3.1
[3,] 3.2  6.5
[4,] 1.6  8.2
```

```

> #(d)
> mymat[-4,-1]
[1] 0.9 3.1 6.5
> matrix(data=mymat[-4,-1])
[,1]
[1,] 0.9
[2,] 3.1
[3,] 6.5
> #(e)
> mymat2 <- mymat[3:4,]
> mymat2
[,1] [,2]
[1,] 3.2 6.5
[2,] 1.6 8.2
> #(f)
> mymat[c(4,1),2:1] <- -0.5*diag(mymat2)
> mymat
[,1] [,2]
[1,] -4.1 -4.1
[2,] 8.2 3.1
[3,] 3.2 6.5
[4,] -1.6 -1.6

```

## Exercise 3.2

- a. Calculate the following:

$$\frac{2}{7} \left( \begin{bmatrix} 1 & 2 \\ 2 & 4 \\ 7 & 6 \end{bmatrix} - \begin{bmatrix} 10 & 20 \\ 30 & 40 \\ 50 & 60 \end{bmatrix} \right)$$

- b. Store these two matrices:

$$A = \begin{bmatrix} 1 \\ 2 \\ 7 \end{bmatrix} \quad B = \begin{bmatrix} 3 \\ 4 \\ 8 \end{bmatrix}$$

Which of the following multiplications are possible? For those that are, compute the result.

- i.  $A \cdot B$
- ii.  $A^T \cdot B$
- iii.  $B^T \cdot (A \cdot A^T)$
- iv.  $(A \cdot A^T) \cdot B^T$
- v.  $[(B \cdot B^T) + (A \cdot A^T) - 100I_3]^{-1}$

- c. For

$$A = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix},$$

confirm that  $A^{-1} \cdot A - I_4$  provides a  $4 \times 4$  matrix of zeros.

## Solution 3.2

```
> #(a)  
> 2/7*(cbind(c(1,2,7),c(2,4,6))-cbind(c(10,30,50),c(20,40,60)))  
 [,1]   [,2]  
[1,] -2.571429 -5.142857  
[2,] -8.000000 -10.285714  
[3,] -12.285714 -15.428571  
> #(b)  
> A <- matrix(data=c(1,2,7))  
> B <- matrix(data=c(3,4,8))  
> ##(i) Not possible  
> ##(ii)  
> t(A)%*%B  
 [,1]  
[1,] 67  
> ##(iii)  
> t(B)%*%(A%*%t(A))  
 [,1] [,2] [,3]  
[1,] 67 134 469  
> ##(iv) Not possible  
> ##(v)  
> solve(B%*%t(B)+A%*%t(A)-100*diag(3))  
 [,1]   [,2]   [,3]  
[1,] -0.007923676 0.003123274 0.007843334  
[2,]  0.003123274 -0.005350239 0.011483806  
[3,]  0.007843334  0.011483806 0.017584735
```

```
> #(c)
> A <- rbind(c(2,0,0,0),c(0,3,0,0),c(0,0,5,0),c(0,0,0,-1))
> solve(A) %*% A-diag(4)
 [,1] [,2] [,3] [,4]
[1,] 0 0 0 0
[2,] 0 0 0 0
[3,] 0 0 0 0
[4,] 0 0 0 0
```

### Exercise 3.3

- a. Create and store a three-dimensional array with six layers of a  $4 \times 2$  matrix, filled with a decreasing sequence of values between 4.8 and 0.1 of the appropriate length.
- b. Extract and store as a new object the fourth- and first-row elements, in that order, of the second column only of all layers of (a).
- c. Use a fourfold repetition n of the second row of the matrix formed in (b) to fill a new array of dimensions  $2 \times 2 \times 2 \times 3$ .
- d. Create a new array comprised of the results of deleting the sixth layer of (a).
- e. Overwrite the second and fourth row elements of the second column of layers 1, 3, and 5 of (d) with -99.

### Solution 3.3

```
> #(a)
```

```
> AR <- array(data=seq(from=4.8,to=0.1,length.out=48),dim=c(4,2,6))
```

```
> AR
```

```
, , 1
```

```
[,1] [,2]
```

```
[1,] 4.8 4.4
```

```
[2,] 4.7 4.3
```

```
[3,] 4.6 4.2
```

```
[4,] 4.5 4.1
```

```
, , 2
```

```
[,1] [,2]
```

```
[1,] 4.0 3.6
```

```
[2,] 3.9 3.5
```

```
[3,] 3.8 3.4
```

```
[4,] 3.7 3.3
```

```
, , 3
```

```
[,1] [,2]
```

```
[1,] 3.2 2.8
```

```
[2,] 3.1 2.7
```

```
[3,] 3.0 2.6
```

```
[4,] 2.9 2.5
```

```
, , 4
```

```
[,1] [,2]
```

```
[1,] 2.4 2.0
```

```
[2,] 2.3 1.9
```

```
[3,] 2.2 1.8
```

```
[4,] 2.1 1.7
```

```
, , 5
```

```
[,1] [,2]
```

```
[1,] 1.6 1.2
```

```
[2,] 1.5 1.1
```

```
[3,] 1.4 1.0
```

```
[4,] 1.3 0.9
```

```
, , 6
```

```
[,1] [,2]
```

```
[1,] 0.8 0.4
```

```
[2,] 0.7 0.3
```

```
[3,] 0.6 0.2
```

```
[4,] 0.5 0.1
```

```
> #(b)
```

```
> BR <- AR[c(4,1),2,]
```

```
> BR
```

```
[,1] [,2] [,3] [,4] [,5] [,6]
```

```
[1,] 4.1 3.3 2.5 1.7 0.9 0.1
```

```
[2,] 4.4 3.6 2.8 2.0 1.2 0.4
```

```
> #(c)
```

```
> CR <- array(data=rep(x=BR[2,],times=4),dim=c(2,2,2,3))
```

```
> CR
```

```
, , 1, 1
```

```
[,1] [,2]
```

```
[1,] 4.4 2.8
```

```
[2,] 3.6 2.0
```

```
, , 2, 1
```

[,1] [,2]  
[1,] 1.2 4.4

[2,] 0.4 3.6  
, , 1, 2

[,1] [,2]  
[1,] 2.8 1.2

[2,] 2.0 0.4  
, , 2, 2

[,1] [,2]  
[1,] 4.4 2.8

[2,] 3.6 2.0  
, , 1, 3

[,1] [,2]  
[1,] 1.2 4.4

[2,] 0.4 3.6  
, , 2, 3

[,1] [,2]  
[1,] 2.8 1.2

[2,] 2.0 0.4

```

> #(d)
> DR <- AR[,,-6]
> DR
, , 1
[,1] [,2]
[1,] 4.8 4.4
[2,] 4.7 4.3
[3,] 4.6 4.2
[4,] 4.5 4.1
, , 2
[,1] [,2]
[1,] 4.0 3.6
[2,] 3.9 3.5
[3,] 3.8 3.4
[4,] 3.7 3.3
, , 3
[,1] [,2]
[1,] 3.2 2.8
[2,] 3.1 2.7
[3,] 3.0 2.6
[4,] 2.9 2.5
, , 4
[,1] [,2]
[1,] 2.4 2.0
[2,] 2.3 1.9
[3,] 2.2 1.8
[4,] 2.1 1.7

```

```

,,5
[,1] [,2]
[1,] 1.6 1.2
[2,] 1.5 1.1
[3,] 1.4 1.0
[4,] 1.3 0.9
> #(e)
> DR[c(2,4),2,c(1,3,5)] <- -99
> DR
,,1
[,1] [,2]
[1,] 4.8 4.4
[2,] 4.7 -99.0
[3,] 4.6 4.2
[4,] 4.5 -99.0
,,2
[,1] [,2]
[1,] 4.0 3.6
[2,] 3.9 3.5
[3,] 3.8 3.4
[4,] 3.7 3.3
,,3
[,1] [,2]
[1,] 3.2 2.8
[2,] 3.1 -99.0
[3,] 3.0 2.6
[4,] 2.9 -99.0

```

, , 4

[,1] [,2]

[1,] 2.4 2.0

[2,] 2.3 1.9

[3,] 2.2 1.8

[4,] 2.1 1.7

, , 5

[,1] [,2]

[1,] 1.6 1.2

[2,] 1.5 -99.0

[3,] 1.4 1.0

[4,] 1.3 -99.0

# Chapter 4: Non-Numeric Values

Estimated time to complete: **30 minutes**

## Exercise 4.1

- a. Store the following vector of 15 values as an object in your workspace:  
 $c(6,9,7,3,6,7,9,6,3,6,6,7,1,9,1)$ . Identify the following elements:
  - i. Those equal to 6
  - ii. Those greater than or equal to 6
  - iii. Those less than  $6 + 2$
  - iv. Those not equal to 6
- b. Create a new vector from the one used in (a) by deleting its first three elements. With this new vector, fill a  $2 \times 2 \times 3$  array.

Examine the array for the following entries:

- i. Those less than or equal to 6 divided by 2, plus 4
- ii. Those less than or equal to 6 divided by 2, plus 4, *after* increasing every element in the array by 2
- c. Confirm the specific locations of elements equal to 0 in the  $10 \times 10$  identity matrix  $I_{10}$  (see Section 3.3).
- d. Check whether *any* of the values of the logical arrays created in (b) are *TRUE*. If they are, check whether they are *all TRUE*.
- e. By extracting the diagonal elements of the logical matrix created in (c), use *any* to confirm there are no *TRUE* entries.

## Solution 4.1

```
> #(a)  
  
> foo <- c(6,9,7,3,6,7,9,6,3,6,6,7,1,9,1)  
  
> foo  
  
[1] 6 9 7 3 6 7 9 6 3 6 6 7 1 9 1  
  
> foo==6  
  
[1] TRUE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE TRUE TRUE  
FALSE FALSE FALSE FALSE  
  
> foo>=6  
  
[1] TRUE TRUE TRUE FALSE TRUE TRUE TRUE FALSE TRUE TRUE TRUE  
FALSE TRUE FALSE  
  
> foo<(6+2)  
  
[1] TRUE FALSE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE  
TRUE FALSE TRUE  
  
> foo!=6  
  
[1] FALSE TRUE TRUE TRUE FALSE TRUE TRUE FALSE TRUE FALSE FALSE  
TRUE TRUE TRUE TRUE
```

```
> #(b)  
> bar <- foo[-(1:3)]  
> bar <- array(data=bar,dim=c(2,2,3))  
> bar  
, , 1  
[,1] [,2]  
[1,] 3 7  
[2,] 6 9  
, , 2  
[,1] [,2]  
[1,] 6 6  
[2,] 3 6  
, , 3  
[,1] [,2]  
[1,] 7 9  
[2,] 1 1  
> bar<=(6/2+4)  
, , 1  
[,1] [,2]  
[1,] TRUE TRUE  
[2,] TRUE FALSE
```

```
, , 2  
[1] [2]  
[1,] TRUE TRUE  
[2,] TRUE TRUE  
, , 3  
[1] [2]  
[1,] TRUE FALSE  
[2,] TRUE TRUE
```

> (bar+2)<=(6/2+4)

```
, , 1  
[1] [2]  
[1,] TRUE FALSE  
[2,] FALSE FALSE  
, , 2  
[1] [2]  
[1,] FALSE FALSE  
[2,] TRUE FALSE  
, , 3  
[1] [2]  
[1,] FALSE FALSE  
[2,] TRUE TRUE
```

> #(c)

> diag(10)==0

```
[1] [2] [3] [4] [5] [6] [7] [8] [9] [10]  
[1,] FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
[2,] TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
[3,] TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE  
[4,] TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE  
[5,] TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE
```

```
[6,] TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE  
[7,] TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE  
[8,] TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE  
[9,] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE  
[10,] TRUE FALSE TRUE  
> #(d)  
  
> any(bar<=(6/2+4))  
  
[1] TRUE  
  
> all(bar<=(6/2+4))  
  
[1] FALSE  
  
> any((bar+2)<=(6/2+4))  
  
[1] TRUE  
  
> all((bar+2)<=(6/2+4))  
  
[1] FALSE  
  
> #(e)  
  
> any(diag(diag(10)==0))  
  
[1] FALSE  
  
>
```

## Exercise 4.2

- a. Store the vector  $c(7,1,7,10,5,9,10,3,10,8)$  as  $foo$ . Identify the elements greater than 5 OR equal to 2.
- b. Store the vector  $c(8,8,4,4,5,1,5,6,6,8)$  as  $bar$ . Identify the elements less than or equal to 6 AND not equal to 4.
- c. Identify the elements that satisfy (a) in  $foo$  AND satisfy (b) in  $bar$ .
- d. Store a third vector called  $baz$  that is equal to the element-wise sum of  $foo$  and  $bar$ . Determine the following:
  - i. The elements of  $baz$  greater than or equal to 14 but not equal to 15
  - ii. The elements of the vector obtained via an element-wise division of  $baz$  by  $foo$  that are greater than 4 OR less than or equal to 2
- f. Confirm that using the long version in all of the preceding exercises performs only the first comparison (that is, the results each match the first entries of the previously obtained vectors).

## Solution 4.2

#(a)

```
> foo <- c(7,1,7,10,5,9,10,3,10,8)  
> (foo>5)|(foo==2)
```

[1] TRUE FALSE TRUE TRUE FALSE TRUE TRUE FALSE TRUE TRUE

> #(b)

```
> bar <- c(8,8,4,4,5,1,5,6,6,8)  
> (bar<=6)&(bar!=4)
```

[1] FALSE FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE FALSE

> #(c)

```
> ((foo>5)|(foo==2))&((bar<=6)&(bar!=4))
```

[1] FALSE FALSE FALSE FALSE FALSE TRUE TRUE TRUE FALSE TRUE FALSE

> #(d)

```
> baz <- foo+bar  
> baz
```

[1] 15 9 11 14 10 10 15 9 16 16

> ##(i)

```
> (baz>=14)&(baz!=15)
```

[1] FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE TRUE TRUE

> ##(ii)

```
> (baz/foo>4)|(baz/foo<=2)
```

[1] FALSE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE

```
> #(e)
> (foo>5)|(foo==2)
[1] TRUE
> (bar<=6)&&(bar!=4)
[1] FALSE
> ((foo>5)|(foo==2))&&((bar<=6)&&(bar!=4))
[1] FALSE
> (baz>=14)&&(baz!=15)
[1] FALSE
> (baz/foo>4)|(baz/foo<=2)
[1] FALSE
```

## Exercise 4.3

- a. Store this vector of 10 values:  $\text{foo} <- \text{c}(7,5,6,1,2,10,8,3,8,2)$ . Then, do the following:
  - i. Extract the elements greater than or equal to 5, storing the result as bar.
  - ii. Display the vector containing those elements from foo that remain after omitting all elements that are greater than or equal to 5.
- b. Use bar from (a)(i) to construct a  $2 \times 3$  matrix called baz, filled in a row-wise fashion. Then, do the following:
  - i. Replace any elements that are equal to 8 with the squared value of the element in row 1, column 2 of baz itself.
  - ii. Confirm that all values in baz are now less than or equal to 25 AND greater than 4.
- c. Create a  $3 \times 2 \times 3$  array called qux using the following vector of 18 values:  $\text{c}(10,5,1,4,7,4,3,3,1,3,4,3,1,7,8,3,7,3)$ . Then, do the following:
  - i. Identify the dimension-specific index positions of elements that are either 3 OR 4.
  - ii. Replace all elements in qux that are less than 3 OR greater than or equal to 7 with the value 100.
- d. Return to *foo* from (a). Use the vector  $\text{c}(F,T)$  to extract every second value from *foo*. In Section 4.1.4, you saw that in some situations, you can substitute 0 and 1 for TRUE and FALSE. Can you perform the same extraction from *foo* using the vector  $\text{c}(0,1)$ ? Why or why not? What does R return in this case?

## Solution 4.3

```
#(a)
> foo <- c(7,5,6,1,2,10,8,3,8,2)

> ##(i)

> bar <- foo[foo>=5]

> ##(ii)

> foo[-which(x=foo>=5)]

[1] 1 2 3 2

> #(b)

> baz <- matrix(data=bar,nrow=2,ncol=3,byrow=T)

> ##(i)

> baz[baz==8] <- baz[1,2]^2

> ##(ii)

> all(baz<=25&baz>4)

[1] TRUE

> #(c)

> qux <- array(data=c(10,5,1,4,7,4,3,3,1,3,4,3,1,7,8,3,7,3),dim=c(3,2,3))

> ##(i)

> which(x=qux==3|qux==4,arr.ind=T)

dim1 dim2 dim3
```

```
[1,] 1 2 1
[2,] 3 2 1
[3,] 1 1 2
[4,] 2 1 2
[5,] 1 2 2
[6,] 2 2 2
[7,] 3 2 2
[8,] 1 2 3
[9,] 3 2 3
> ##(ii)
> qux[qux<3|qux>=7] <- 100
> #(d)
> foo[c(F,T)]
[1] 5 1 10 3 2
> foo[c(0,1)]
[1] 7
```

## Exercise 4.4

- a. Re-create exactly the following output:

```
"The quick brown fox  
jumped over  
the lazy dogs"
```

- b. Suppose you've stored the values `num1 <- 4` and `num2 <- 0.75`.

Write a line of R code that returns the following string:

```
[1] "The result of multiplying 4 by 0.75 is 3"
```

Make sure your code produces a string with the correct multiplication result for *any* two numbers stored as `num1` and `num2`.

- c. On my local machine, the directory for my work on this book is specified in R as `"/Users/tdavies/Documents/RBook/"`. Imagine it is your machine—write a line of code that replaces `tdavies` in this string with your first initial and surname.

- d. In Section 4.2.4, you stored the following string:

```
R> bar <- "How much wood could a woodchuck chuck"
```

- i. Store a new string by gluing onto `bar` the words `"if a woodchuck could chuck wood"`.
  - ii. In the result of (i), replace all instances of `wood` with `metal`.
- e. Store the string `"Two 6-packs for $12.99"`. Then do the following:
- i. Use a check for equality to confirm that the substring beginning with character 5 and ending with character 10 is `"6-pack"`.
  - ii. Make it a better deal by changing the price to \$10.99.

## Solution 4.4

```
#(a)
```

```
> cat("\\"The quick brown fox\n\tjumped over\n\tthe lazy dogs\\")
```

```
"The quick brown fox
```

```
        jumped over
```

```
            the lazy dogs"> #(b)
```

```
> num1 <- 4
```

```
> num2 <- 0.75
```

```
> paste("The result of multiplying",num1,"by",num2,"is",num1*num2)
```

```
[1] "The result of multiplying 4 by 0.75 is 3"
```

```
> #(c)
```

```
> sub(pattern="tdavies",replacement="aschwarzenegger",x="/Users/tdavies/Documents/RBook")
```

```
[1] "/Users/aschwarzenegger/Documents/RBook"
```

```
> #(d)

> bar <- "How much wood could a woodchuck chuck"
> ##(i)

> baz <- paste(bar,"if a woodchuck could chuck wood")
> ##(ii)

> gsub(pattern="wood",replacement="metal",x=baz)
[1] "How much metal could a metalchuck chuck if a metalchuck could chuck metal"

> #(e)

> foo <- "Two 6-packs for $12.99"
> #(i)

> substr(x=foo,start=5,stop=10)=="6-pack"
[1] TRUE

> #(ii)

> substr(x=foo,start=19,stop=19) <- "0"
> foo
[1] "Two 6-packs for $10.99"
```

## Exercise 4.5

The New Zealand government consists of the political parties National, Labour, Greens, and Maori, with several smaller parties labeled as Other. Suppose you asked 20 New Zealanders which of these they identified most with and obtained the following data:

- There were 12 males and 8 females; the individuals numbered 1, 5–7, 12, and 14–16 were females.
- The individuals numbered 1, 4, 12, 15, 16, and 19 identified with Labour; no one identified with Maori; the individuals numbered 6, 9, and 11 identified with Greens; 10 and 20 identified with Other; and the rest identified with National.

- a. Use your knowledge of vectors (for example, subsetting and overwriting) to create two character vectors: *sex* with entries "M" (male) and "F" (female) and *party* with entries "National", "Labour", "Greens", "Maori", and "Other". Make sure the entries are placed in the correct positions as outlined earlier.
- b. Create two different factor vectors based on *sex* and *party*. Does it make any sense to use *ordered=TRUE* in either case? How has R appeared to arrange the levels?
- c. Use factor subsetting to do the following:
  - i. Return the factor vector of chosen parties for only the male participants.
  - ii. Return the factor vector of genders for those who chose National.
- d. Another six people joined the survey, with the results *c("National","Maori","Maori","Labour","Greens","Labour")* for the preferred party and *c("M","M","F","F","F","M")* as their gender.  
Combine these results with the original factors from (b).  
Suppose you also asked all individuals to state how confident they were that Labour will win more seats in Parliament than National in the next election and to attach a subjective percentage to that confidence. The following 26 results were obtained: 93, 55, 29, 100, 52, 84, 56, 0, 33, 52, 35, 53, 55, 46, 40, 40, 56, 45, 64, 31, 10, 29, 40, 95, 18, 61.
  - e. Create a factor with levels of confidence as follows: Low for percentages [0,30]; Moderate for percentages (30,70]; and High for percentages (70,100].
  - f. From (e), extract the levels corresponding to those individuals who originally said they identified with Labour. Do this also for National. What do you notice?

## Solution 4.5

#(a)

```
> party <- rep("National",20)  
> party[c(1,4,12,15,16,19)] <- "Labour"  
> party[c(6,9,11)] <- "Greens"  
> party[c(10,20)] <- "Other"  
  
> party
```

```
[1] "Labour" "National" "National" "Labour" "National" "Greens" "National" "National"  
"Greens" "Other" "Greens" "Labour" "National" "National" "Labour" "Labour" "National"
```

```
[18] "National" "Labour" "Other"
```

```
> sex <- rep("M",20)
```

```
> sex[c(1,5:7,12,14:16)] <- "F"
```

```
> sex
```

```
[1] "F" "M" "M" "M" "M" "F" "F" "M" "M" "M" "M" "F" "M" "F" "F" "F" "M" "M" "M" "M"
```

```
> #(b)
```

```
> sex.fac <- factor(x=sex)
```

```
> sex.fac
```

```
[1] F M M M F F F M M M M F M F F F M M M M
```

Levels: F M

```
> party.fac <- factor(x=party,levels=c("National","Labour","Greens","Maori","Other"))
```

```
> party.fac # Should not use ordered=TRUE, there is no 'natural' or 'low-to-high' ordering here.  
Factor levels are arranged in the order specified in the 'levels' argument.
```

```
[1] Labour National National Labour National Greens National National Greens Other  
Greens Labour National National Labour Labour National National Labour Other
```

Levels: National Labour Greens Maori Other

```
> #(c)
```

```
> ##(i)
```

```
> party.fac[sex.fac=="M"]
```

```
[1] National National Labour National Greens Other Greens National National National  
Labour Other
```

Levels: National Labour Greens Maori Other

```
> ##(ii)
```

```
> sex.fac[party.fac=="National"]
```

```
[1] M M F F M M F M M
```

Levels: F M

```
> #(d)
```

```
> sex.newvals <- factor(x=c("M","M","F","F","F","M"))
```

```
> sex.fac <- factor(x=levels(sex.fac)[c(sex.fac,sex.newvals)])
```

```
> sex.fac
```

```
[1] F M M M F F F M M M M F M F F F M M M M M F F F M
```

Levels: F M

```

> party.newvals <-  

factor(x=c("National","Maori","Maori","Labour","Greens","Labour"),levels=levels(party.fac))  

> party.fac <- factor(x=levels(party.fac)[c(party.fac,party.newvals)])  

> party.fac  

[1] Labour National National Labour National Greens National National Greens Other  

Greens Labour National National Labour Labour National National Labour Other  

National  

[22] Maori Maori Labour Greens Labour  

Levels: Greens Labour Maori National Other  

> #(e)  

> conf <- c(93,55,29,100,52,84,56,0,33,52,35,53,55,46,40,40,56,45,64,31,10,29,40,95,18,61)  

> conf.fac <-  

cut(x=conf,breaks=c(0,30,70,100),include.lowest=TRUE,labels=c("Low","Moderate","High"))  

> #(f)  

> conf.fac[party.fac=="Labour"]  

[1] High High Moderate Moderate Moderate Moderate High Moderate  

Levels: Low Moderate High  

> conf.fac[party.fac=="National"] # Theres an indication that those who identify as "Labour" have  

greater confidence than those who identify as "National" when it comes to guessing how well  

Labour will do in the next election.  

[1] Moderate Low Moderate Moderate Low Moderate Moderate Moderate Moderate Low  

Levels: Low Moderate High

```

# Chapter 5: Lists and Dataframes

Estimated time to complete: **30 minutes**

## Exercise 5.1

- a. Create a list that contains, in this order, a sequence of 20 evenly spaced numbers between  $-4$  and  $4$ ; a  $3 \times 3$  matrix of the logical vector  $c(F,T,T,T,F,T,T,F,F)$  filled column-wise; a character vector with the two strings "*don*" and "*quixote*"; and a factor vector containing the observations  $c("LOW", "MED", "LOW", "MED", "MED", "HIGH")$ .  
Then, do the following:
  - i. Extract row elements 2 and 1 of columns 2 and 3, in that order, of the logical matrix.
  - ii. Use *sub* to overwrite "*quixote*" with "*Quixote*" and "*don*" with "*Don*" inside the list.  
Then, using the newly overwritten list member, concatenate to the console screen the following statement exactly:  
*"Windmills! ATTACK!"*  
*-\Don Quixote/-*
  - iii. Obtain all values from the sequence between  $-4$  and  $4$  that are greater than 1.
  - iv. Using *which*, determine which indexes in the factor vector are assigned the "*MED*" level.
- b. Create a new list with the factor vector from (a) as a component named "*facs*"; the numeric vector  $c(3,2.1,3.3,4,1.5,4.9)$  as a component named "*nums*"; and a nested list comprised of the first three members of the list from (a) (use list slicing to obtain this), named "*oldlist*". Then, do the following:
  - i. Extract the elements of "*facs*" that correspond to elements of "*nums*" that are greater than or equal to 3.
  - ii. Add a new member to the list named "*flags*". This member should be a logical vector of length 6, obtained as a twofold repetition of the third column of the logical matrix in the "*oldlist*" component.
  - iii. Use "*flags*" and the logical negation operator *!* to extract the entries of "*num*" corresponding to *FALSE*.
  - iv. Overwrite the character string vector component of "*oldlist*" with the single character string "*Don Quixote*".

## Solution 5.1

```
> #(a)

> foo <- list(seq=-
4,to=4,length=20),matrix(c(F,T,T,T,F,T,T,F,F),nrow=3,ncol=3),c("don","quixote"),factor(x=c("L
OW","MED","LOW","MED","MED","HIGH")))

> ##(i)

> foo[[2]][2:1,2:3]

 [,1] [,2]

[1,] FALSE FALSE

[2,] TRUE TRUE
```

```

> ##(ii)

> foo[[3]][1] <- sub(pattern="d",replacement="D",x=foo[[3]][1])
> foo[[3]][2] <- sub(pattern="q",replacement="Q",x=foo[[3]][2])
> cat("\Windmills! ATTACK!\n\t-",foo[[3]][1]," ",foo[[3]][2],"/-",sep="")
"Windmills! ATTACK!"

-\\Don Quixote/->

##(iii)

> foo[[1]][foo[[1]]>1]
[1] 1.052632 1.473684 1.894737 2.315789 2.736842 3.157895 3.578947 4.000000

> ##(iv)

> which(x=foo[[4]]=="MED")
[1] 2 4 5

> #(b)

> bar <- list(facs=foo[[4]],nums=c(3,2.1,3.3,4,1.5,4.9),oldlist=foo[1:3])
> ##(i)

> bar$fac<-bar$fac>=3
[1] LOW LOW MED HIGH

Levels: HIGH LOW MED

> ##(ii)

> bar$flags <- rep(x=bar$oldlist[[2]][,3],times=2)
> ##(iii)

> bar$nums[!bar$flags]
[1] 2.1 3.3 1.5 4.9

> ##(iv)

> bar$oldlist[[3]] <- "Don Quixote"

```

## Exersise 5.2

- a. Create and store this data frame as *dframe* in your R workspace:

person	sex	funny
Stan	M	High
Francine	F	Med
Steve	M	Low
Roger	M	High
Hayley	F	Med
Klaus	M	Med

The variables *person*, *sex*, and *funny* should be identical in nature to the variables in the *mydata* object studied throughout

Section 5.2. That is, *person* should be a character vector, *sex* should be a factor with levels *F* and *M*, and *funny* should be a factor with levels *Low*, *Med*, and *High*.

- b. Stan and Francine are 41 years old, Steve is 15, Hayley is 21, and Klaus is 60. Roger is extremely old—1,600 years. Append these data as a new numeric column variable in *dframe* called *age*.
- c. Use your knowledge of reordering the column variables based on column index positions to overwrite *dframe*, bringing it in line with *mydata*. That is, the first column should be *person*, the second column *age*, the third column *sex*, and the fourth column *funny*.
- d. Turn your attention to *mydata* as it was left after you included the *age.mon* variable in Section 5.2.2. Create a new version of *mydata* called *mydata2* by deleting the *age.mon* column.
- e. Now, combine *mydata2* with *dframe*, naming the resulting object *mydataframe*.
- f. Write a single line of code that will extract from *mydataframe* just the names and ages of any records where the individual is female and has a level of funniness equal to *Med* OR *High*.
- g. Use your knowledge of handling character strings in R to extract all records from *mydataframe* that correspond to people whose names start with *S*. Hint: Recall *substr* from Section 4.2.4 (note that *substr* can be applied to a vector of multiple character strings).

## Solution 5.2

In order to start Solution for 5.2 you need to complete exercises from the section 5.2

Section 5.2 #

## 5.2.1 ##

```
> mydata <-  
data.frame(person=c("Peter", "Lois", "Meg", "Chris", "Stewie"), age=c(42, 40, 17, 14, 1), sex=factor(c("M", "F", "F", "M", "M")))
```

```
> mydata
```

person age sex

```
1 Peter 42 M  
2 Lois 40 F  
3 Meg 17 F  
4 Chris 14 M  
5 Stewie 1 M
```

```
>
```

```
> #
```

```
>
```

```
> mydata[2,2]
```

```
[1] 40
```

```
>
```

```
> #
```

```
>
```

```
> mydata[3:5,3]
```

```
[1] F M M
```

Levels: F M

```
>
```

```
> #
```

```
>
```

```
> mydata[,c(3,1)]
```

sex person

```
1 M Peter
```

```
2 F Lois
```

```

3 F Meg
4 M Chris
5 M Stewie
> mydata$age
[1] 42 40 17 14 1
>
> #
>
> mydata$age[2]
[1] 40
>
> #
>
> nrow(mydata)
[1] 5
> ncol(mydata)
[1] 3
> dim(mydata)
[1] 5 3
>
> #
>
> mydata$person
[1] Peter Lois Meg Chris Stewie
Levels: Chris Lois Meg Peter Stewie
>
> #
>
> mydata <-
data.frame(person=c("Peter","Lois","Meg","Chris","Stewie"),age=c(42,40,17,14,1),sex=factor(c("M","F","F","M","M")),stringsAsFactors=FALSE)
> mydata

```

```

person age sex
1 Peter 42 M
2 Lois 40 F
3 Meg 17 F
4 Chris 14 M
5 Stewie 1 M
> mydata$person
[1] "Peter" "Lois" "Meg" "Chris" "Stewie"

>
>
> ## 5.2.2 ##
>
> newrecord <- data.frame(person="Brian",age=7,sex=factor("M",levels=levels(mydata$sex)))
> newrecord

person age sex
1 Brian 7 M
>
> #
>
> mydata <- rbind(mydata,newrecord)
> mydata

person age sex
1 Peter 42 M
2 Lois 40 F
3 Meg 17 F
4 Chris 14 M
5 Stewie 1 M
6 Brian 7 M
>
> #
>
> funny <- c("High","High","Low","Med","High","Med")
> funny <- factor(x=funny,levels=c("Low","Med","High"))
> funny
[1] High High Low Med High Med

```

Levels: Low Med High

```
>  
> #  
>  
> mydata <- cbind(mydata,funny)  
> mydata  
  person age sex funny  
1 Peter 42 M High  
2 Lois 40 F High  
3 Meg 17 F Low  
4 Chris 14 M Med  
5 Stewie 1 M High  
6 Brian 7 M Med  
>  
> #  
>  
> mydata$age.mon <- mydata$age*12  
> mydata  
  person age sex funny age.mon  
1 Peter 42 M High 504  
2 Lois 40 F High 480  
3 Meg 17 F Low 204  
4 Chris 14 M Med 168  
5 Stewie 1 M High 12  
6 Brian 7 M Med 84  
  
#a  
> dframe <-  
  data.frame(person=c("Stan","Francine","Steve","Roger","Hayley","Klaus"),sex=factor(x=c("M","F","M","M","F","M")),funny=factor(x=c("High","Med","Low","High","Med","Med")),levels=c("Low","Med","High")),stringsAsFactors=F)  
> dframe  
  person sex funny  
1 Stan M High  
2 Francine F Med  
3 Steve M Low  
4 Roger M High  
5 Hayley F Med  
6 Klaus M Med  
#b
```

```

> dframe$age <- c(41,41,15,1600,21,60)
> dframe
  person sex funny age
1 Stan M High 41
2 Francine F Med 41
3 Steve M Low 15
4 Roger M High 1600
5 Hayley F Med 21
6 Klaus M Med 60

#c
> dframe <- dframe[,c(1,4,2,3)]
> dframe
  person age sex funny
1 Stan 41 M High
2 Francine 41 F Med
3 Steve 15 M Low
4 Roger 1600 M High
5 Hayley 21 F Med
6 Klaus 60 M Med

#d
> mydata2 <- mydata[,-5] #(Assuming the presence of the 'mydata' object as left in Section 5.2.2)

#e
> mydataframe <- rbind(mydata2,dframe)
> mydataframe

  person age sex funny
1 Peter 42 M High
2 Lois 40 F High
3 Meg 17 F Low
4 Chris 14 M Med
5 Stewie 1 M High
6 Brian 7 M Med
7 Stan 41 M High
8 Francine 41 F Med
9 Steve 15 M Low
10 Roger 1600 M High
11 Hayley 21 F Med
12 Klaus 60 M Med

```

```
#f  
>  
mydataframe[mydataframe$sex=="F"&(mydataframe$funny=="Med"|mydataframe$funny=="High"),c("person","age")]  
person age  
2 Lois 40  
8 Francine 41  
11 Hayley 21  
  
#g  
> mydataframe[substr(x=mydataframe$person,start=1,stop=1)=="S",]  
person age sex funny  
5 Stewie 1 M High  
7 Stan 41 M High  
9 Steve 15 M Low  
>
```

# Chapter 6: Special Values, Classes, Coercion

Estimated time to complete: **30 minutes**

## Exercise 6.1

- a. Store the following vector:

```
foo <- c(13563,-14156,-14319,16981,12921,11979,9568,8833,-12968,8133)
```

Then, do the following:

- i. Output all elements of *foo* that, when raised to a power of 75, are NOT infinite.
- ii. Return the elements of *foo*, excluding those that result in negative infinity when raised to a power of 75.
- b. Store the following  $3 \times 4$  matrix as the object *bar*:

$$\begin{bmatrix} 77875.40 & 27551.45 & 23764.30 & -36478.88 \\ -35466.25 & -73333.85 & 36599.69 & -70585.69 \\ -39803.81 & 55976.34 & 76694.82 & 47032.00 \end{bmatrix}$$

Now, do the following:

- i. Identify the coordinate-specific indexes of the entries of *bar* that are *NaN* when you raise *bar* to a power of 65 and divide by infinity.
- ii. Return the values in *bar* that are NOT *NaN* when *bar* is raised to a power of 67 and infinity is added to the result. Confirm this is identical to identifying those values in *bar* that, when
- iii. raised to a power of 67, are not equal to negative infinity.
- iv. Identify those values in *bar* that are either negative infinity OR finite when you raise *bar* to a power of 67.

## Solution 6.1

```
> #(a)

> foo <- c(13563,-14156,-14319,16981,12921,11979,9568,8833,-12968,8133)

> ##(i)

> foo[is.finite(foo^75)]

[1] 11979 9568 8833 8133

> ##(ii)

> foo[-which(foo^75==Inf)]

[1] 13563 16981 12921 11979 9568 8833 8133

> #(b)

> bar <- matrix(c(77875.4,-35466.25,-39803.81,27551.45,-73333.85,55976.34,23764.3,36599.69,76694.82,-36478.88,-70585.69,47032),nrow=3,ncol=4)
```

```

> ##(i)
> which(is.nan(bar^65/Inf),arr.ind=T)
  row col
[1,] 1  1
[2,] 2  2
[3,] 3  2
[4,] 3  3
[5,] 2  4
> ##(ii)
> bar[!is.nan(bar^67+Inf)]
[1] 77875.40 -35466.25 -39803.81 27551.45 55976.34 23764.30 36599.69 76694.82 -
36478.88 47032.00
> bar[bar^67!=Inf]
[1] 77875.40 -35466.25 -39803.81 27551.45 55976.34 23764.30 36599.69 76694.82 -
36478.88 47032.00
> ##(iii)
> bar[bar^67==Inf|is.finite(bar^67)]
[1] -35466.25 -39803.81 27551.45 -73333.85 23764.30 36599.69 -36478.88 -70585.69
>

```

## Exercise 6.2

- a. Consider the following line of code:

```
foo <- c(4.3,2.2,NULL,2.4,NaN,3.3,3.1,NULL,3.4,NA)
```

Decide yourself which of the following statements are true and which are false and then use R to confirm:

- i. The length of foo is 8.
  - ii. Calling `which(x=is.na(x=foo))` will not result in 4 and 8.
  - iii. Checking `is.null(x=foo)` will provide you with the locations of the two *NULL* values that are present.
  - iv. Executing `is.na(x=foo[8])+4/NULL` will not result in *NA*.
- b. Create and store a list containing a single member: the vector `c(7,7,NA,3,NA,1,1,5,NA)`. Then, do the following:
- i. Name the member "*alpha*".
  - ii. Confirm that the list doesn't have a member with the name "*beta*" using the appropriate logical valued function.
  - iii. Add a new member called *beta*, which is the vector obtained by identifying the index positions of *alpha* that are *NA*.

## Solution 6.2

```
> #(a)  
  
> foo <- c(4.3,2.2,NULL,2.4,NaN,3.3,3.1,NULL,3.4,NA)  
  
> ##(i)  
  
> length(x=foo)  
  
[1] 8  
  
> ##(ii)  
  
> which(x=is.na(x=foo))  
  
[1] 4 8  
  
> ##(iii)  
  
> is.null(x=foo)  
  
[1] FALSE  
  
> ##(iv)  
  
> is.na(x=foo[8])+4/NULL  
  
numeric(0)  
  
> #(b)  
  
> bar <- list(c(7,7,NA,3,NA,1,1,5,NA))  
  
> ##(i)  
  
> names(bar) <- "alpha"
```

```
> ##(ii)
> is.null(x=bar$beta)
[1] TRUE
> ##(iii)
> bar$beta <- which(x=is.na(x=bar$alpha))
> bar
$alpha
[1] 7 7 NA 3 NA 1 1 5 NA

$beta
[1] 3 5 9
```

## Exercise 6.3

- a. Identify the class of the following objects. For each object, also state whether the class is explicitly or implicitly defined.
- i. `foo <- array(data=1:36,dim=c(3,3,4))`
  - ii. `bar <- as.vector(foo)`
  - iii. `baz <- as.character(bar)`
  - iv. `qux <- as.factor(baz)`
  - v. `quux <- bar+c(-0.1,0.1)`
- b. For each object defined in (a), find the sum of the result of calling `is.numeric` and `is.integer` on it separately. For example, `is.numeric(foo)+is.integer(foo)` would compute the sum for (i). Turn the collection of five results into a factor with levels 0, 1, and 2, identified by the results themselves. Compare this factor vector with the result of coercing it to a numeric vector.
- c. Turn the following:

```
[,1] [,2] [,3] [,4]  
[1,] 2 5 8 11  
[2,] 3 6 9 12  
[3,] 4 7 10 13
```

into the following:

```
[1] "2" "5" "8" "11" "3" "6" "9" "12" "4" "7" "10" "13"
```

- d. Store the following matrix:

$$\begin{bmatrix} 34 & 0 & 1 \\ 23 & 1 & 2 \\ 33 & 1 & 1 \\ 42 & 0 & 1 \\ 41 & 0 & 2 \end{bmatrix}$$

Then, do the following:

- i. Coerce the matrix to a data frame.
- ii. As a data frame, coerce the second column to be logicalvalued.
- iii. As a data frame, coerce the third column to be factor-valued.

### Solution 6.3

```
> #(a)  
> ##(i)  
> foo <- array(data=1:36,dim=c(3,3,4))  
> foo  
, , 1
```

```
[,1] [,2] [,3]  
[1,] 1 4 7  
[2,] 2 5 8  
[3,] 3 6 9
```

```
, , 2
```

```
[,1] [,2] [,3]  
[1,] 10 13 16  
[2,] 11 14 17  
[3,] 12 15 18
```

```
, , 3
```

```
[,1] [,2] [,3]  
[1,] 19 22 25  
[2,] 20 23 26  
[3,] 21 24 27
```

```
, , 4
```

```
[,1] [,2] [,3]  
[1,] 28 31 34  
[2,] 29 32 35
```

```
[3,] 30 33 36
```

```
> class(foo)
[1] "array"
> attributes(foo) #implicit
$dim
[1] 3 3 4

> ##(ii)
> bar <- as.vector(foo)
> bar
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
33 34 35 36
> class(bar)
[1] "integer"
> attributes(bar) #implicit
NULL
> ##(iii)
> baz <- as.character(bar)
> baz
[1] "1" "2" "3" "4" "5" "6" "7" "8" "9" "10" "11" "12" "13" "14" "15" "16" "17" "18" "19"
"20" "21" "22" "23" "24" "25" "26" "27" "28" "29" "30" "31" "32" "33" "34" "35" "36"
> class(baz)
[1] "character"
> attributes(baz) #implicit
NULL
> ##(iv)
> qux <- as.factor(baz)
> qux
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
33 34 35 36
```

```

Levels: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3 30 31 32 33 34 35 36
4 5 6 7 8 9

> class(qux)
[1] "factor"

> attributes(qux) #explicit
$levels
[1] "1" "10" "11" "12" "13" "14" "15" "16" "17" "18" "19" "2" "20" "21" "22" "23" "24" "25"
"26" "27" "28" "29" "3" "30" "31" "32" "33" "34" "35" "36" "4" "5" "6" "7" "8" "9"

$class
[1] "factor"

> ##(v)
> quux <- bar+c(-0.1,0.1)
> quux
[1] 0.9 2.1 2.9 4.1 4.9 6.1 6.9 8.1 8.9 10.1 10.9 12.1 12.9 14.1 14.9 16.1 16.9 18.1 18.9 20.1
20.9 22.1 22.9 24.1 24.9 26.1 26.9 28.1 28.9 30.1 30.9 32.1 32.9 34.1 34.9 36.1

> class(quux)
[1] "numeric"

> attributes(quux) #implicit
NULL
> #(b)

> foo.sum <- is.numeric(foo)+is.integer(foo)
> bar.sum <- is.numeric(bar)+is.integer(bar)
> baz.sum <- is.numeric(baz)+is.integer(baz)
> qux.sum <- is.numeric(qux)+is.integer(qux)
> quux.sum <- is.numeric(quux)+is.integer(quux)
> myfac <- factor(x=c(foo.sum,bar.sum,baz.sum,qux.sum,quux.sum),levels=c(0,1,2))
> myfac
[1] 2 2 0 0 1

Levels: 0 1 2

> as.numeric(myfac)

```

```

[1] 3 3 1 1 2
> #(c)

> foo <- matrix(data=2:13,nrow=3,ncol=4)
> foo
 [,1] [,2] [,3] [,4]
[1,]  2   5   8   11
[2,]  3   6   9   12
[3,]  4   7   10  13

> as.character(as.vector(t(foo)))
[1] "2"  "5"  "8"  "11" "3"  "6"  "9"  "12" "4"  "7"  "10" "13"

> #(d)

> foo <- cbind(c(34,23,33,42,41),c(0,1,1,0,0),c(1,2,1,1,2))
> foo
 [,1] [,2] [,3]
[1,] 34   0   1
[2,] 23   1   2
[3,] 33   1   1
[4,] 42   0   1
[5,] 41   0   2

> ##(i)

> foo <- as.data.frame(foo)
> foo
  V1 V2 V3
1 34  0  1
2 23  1  2
3 33  1  1
4 42  0  1
5 41  0  2

> ##(ii)

> foo[,2] <- as.logical(foo[,2])
> foo

```

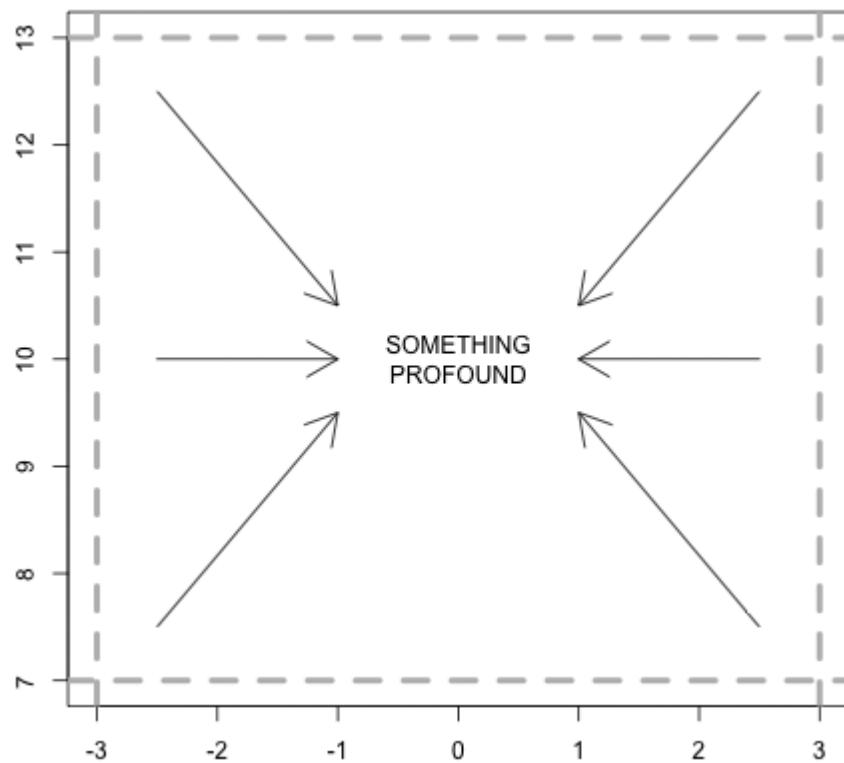
```
V1  V2 V3
1 34 FALSE 1
2 23 TRUE 2
3 33 TRUE 1
4 42 FALSE 1
5 41 FALSE 2
> ##(iii)
> foo[,3] <- as.factor(foo[,3])
> foo
V1  V2 V3
1 34 FALSE 1
2 23 TRUE 2
3 33 TRUE 1
4 42 FALSE 1
5 41 FALSE 2
> foo$V3
[1] 1 2 1 1 2
Levels: 1 2
```

# Chapter 7: Basic Plotting

Estimated time to complete: **15 minutes**

## Exercise 7.1

- a. As closely as you can, re-create the following plot:



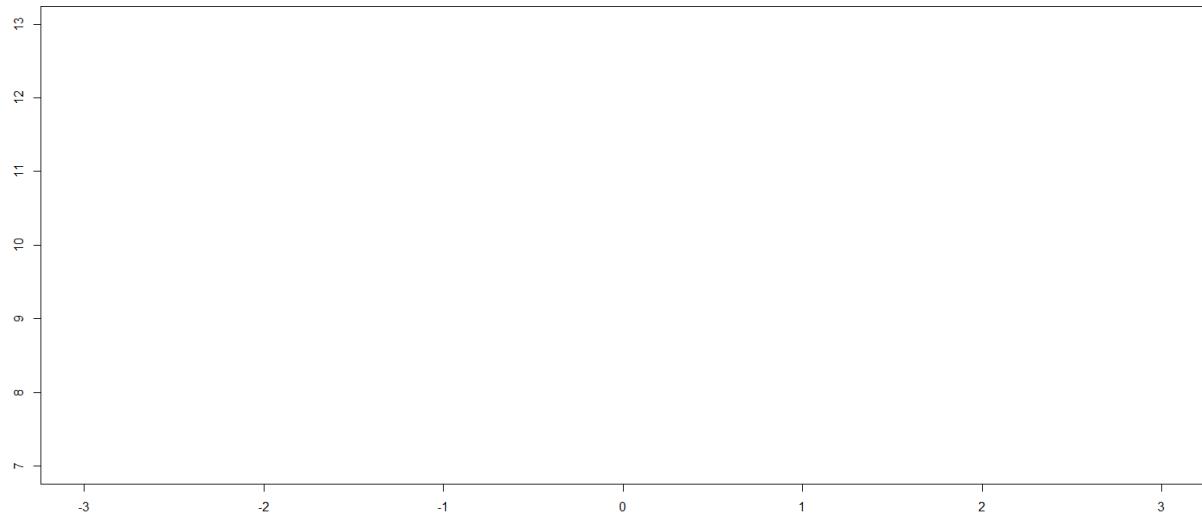
With the following data, create a plot of weight on the x-axis and height on the y-axis. Use different point characters or colors to distinguish between males and females and provide a matching legend. Label the axes and give the plot a title.

Weight (kg)	Height (cm)	Sex
55	161	female
85	185	male
75	174	male
42	154	female
93	188	male
63	178	male
58	170	female
75	167	male
89	181	male
67	178	female

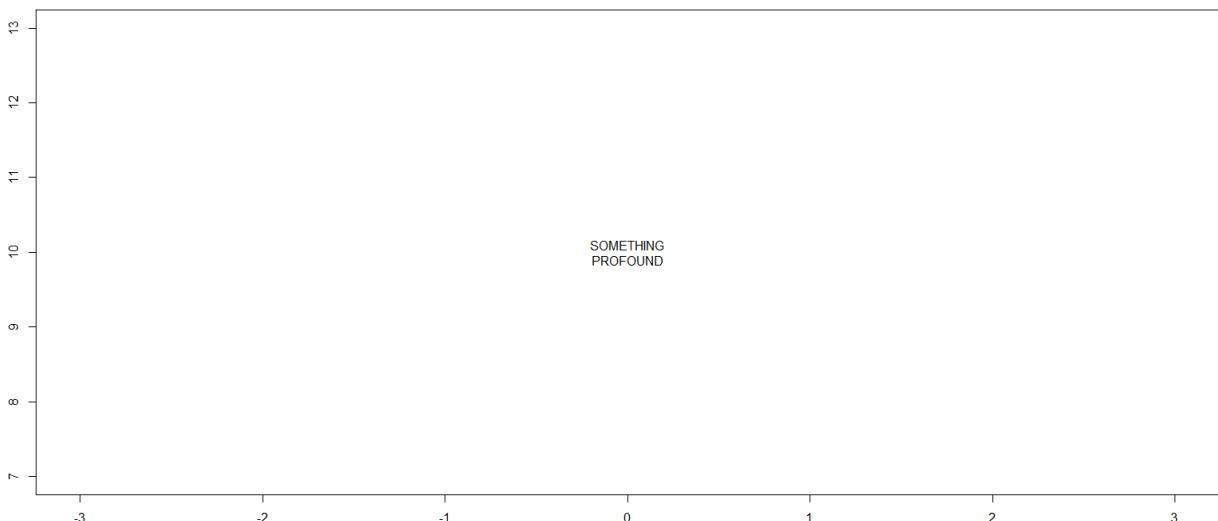
## Solution 7.1

```
> #(a)
```

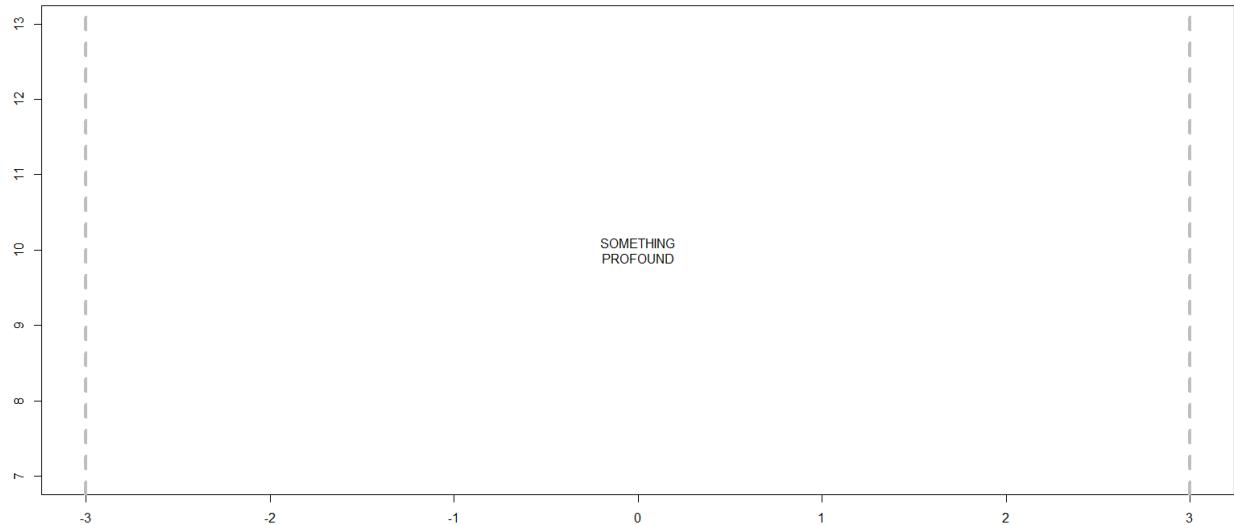
```
> plot(-3:3,7:13,type="n",xlab="",ylab="")
```



```
> text(x=0,y=10,labels="SOMETHING\nPROFOUND")
```



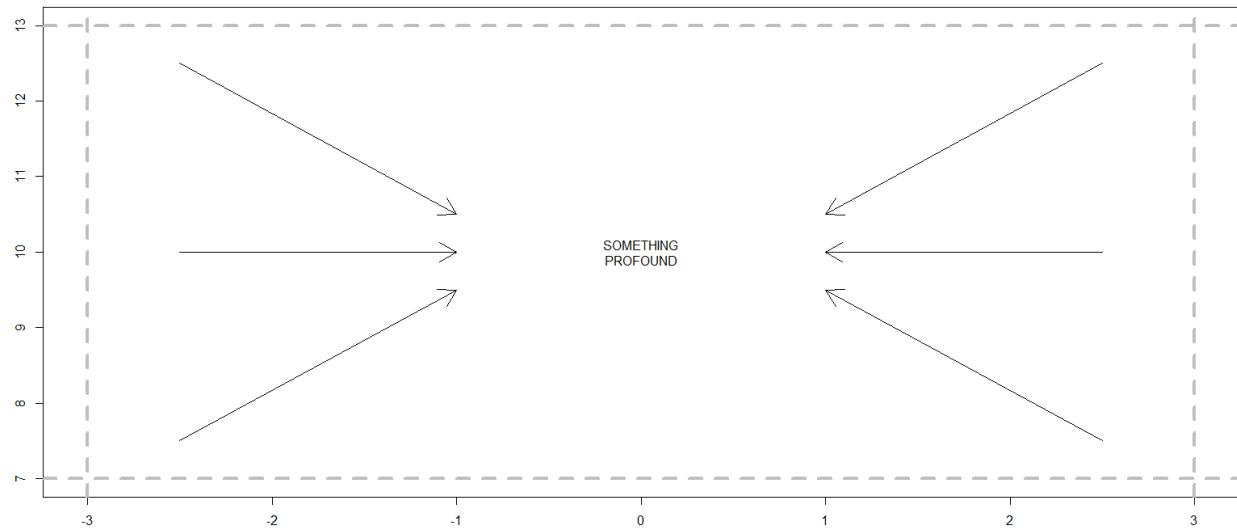
```
> abline(v=c(-3,3),lty=2,lwd=4,col=8)
```



```
> abline(h=c(7,13),lty=2,lwd=4,col=8)
```

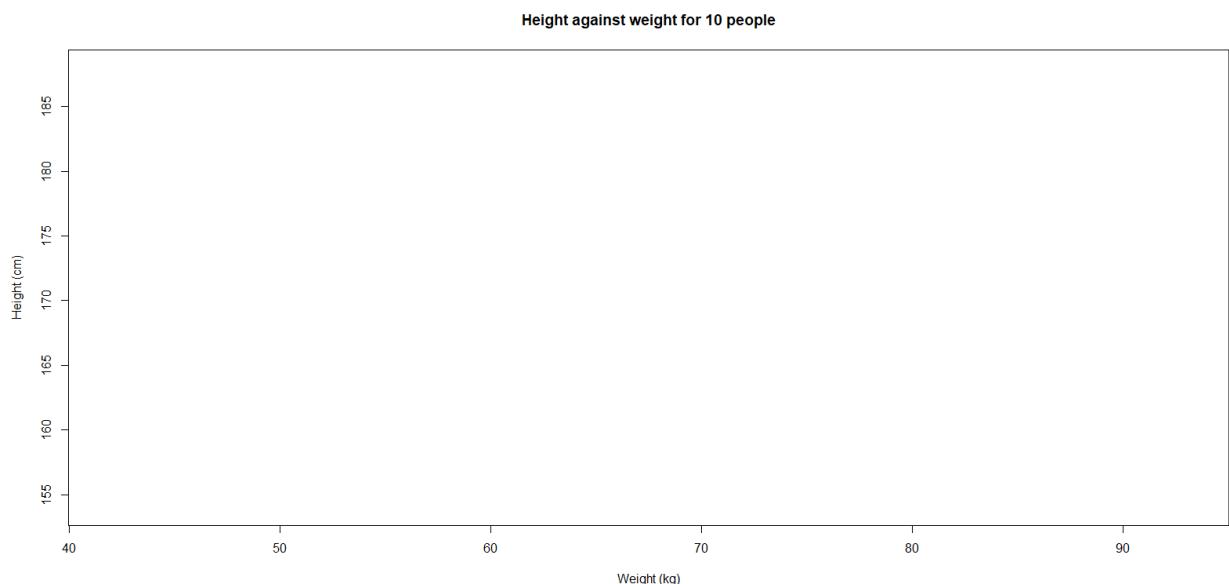


```
> arrows(x0=c(-2.5,-2.5,-2.5,2.5,2.5,2.5),y0=c(7.5,10,12.5,7.5,10,12.5),x1=c(-1,-1,-1,1,1,1),y1=c(9.5,10,10.5,9.5,10,10.5))
```



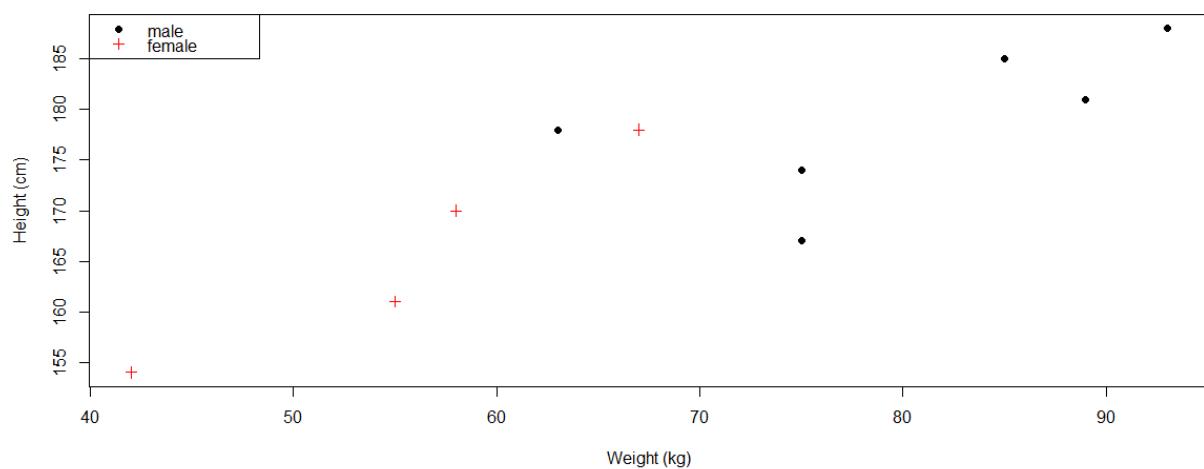
```
> #(b)
```

```
> w <- c(55,85,75,42,93,63,58,75,89,67)
> h <- c(161,185,174,154,188,178,170,167,181,178)
> s <- c("female","male","male","female","male","male","female","male","male","female")
> plot(w,h,type="n",xlab="Weight (kg)",ylab="Height (cm)",main="Height against weight for 10 people")
```



```
> points(w[s=="male"],h[s=="male"],pch=19)
> points(w[s=="female"],h[s=="female"],pch=3,col=2)
> legend("topleft",legend=c("male","female"),pch=c(19,3),col=c(1,2))
```

Height against weight for 10 people



# Chapter 8: Reading and Writing Files

Estimated time to complete: **15 minutes**

## Exercise 8.1

- a. In R's built-in *datasets* library is the data frame *quakes*. Make sure you can access this object and view the corresponding help file to get an idea of what this data represents. Then, do the following:
  - i. Select only those records that correspond to a magnitude (*mag*) of greater than or equal to 5 and write them to a table-format file called *q5.txt* in an existing folder on your machine. Use a delimiting character of ! and do not include any row names.
  - ii. Read the file back into your R workspace, naming the object *q5.dframe*.
- b. In the contributed package *car*, there's a data frame called *Duncan*, which provides historical data on perceived job prestige in 1950.

Install the *car* package and access the *Duncan* data set and its help file. Then, do the following:

- i. Write R code that will plot *education* on the x-axis and *income* on the y-axis, with both x- and y-axis limits fixed to be [0,100]. Provide appropriate axis labels. For jobs with *prestige* value of less than or equal to 80, use a black  $\circ$  as the point character. For jobs with *prestige* greater than 80, use a blue  $\bullet$ .
- ii. Add a legend explaining the difference between the two types of points and then save a 500  $\times$  500 pixel *.png* file of the image.
- c. Create a list called *exer* that contains the three data sets *quakes*, *q5.dframe*, and *Duncan*. Then, do the following:
  - i. Write the list object directly to disk, calling it *Exercise8-1.txt*.
  - ii. Briefly inspect the contents of the file in a text editor.
  - iii. Read *Exercise8-1.txt* back into your workspace; call the resulting object *list.of.dataframes*. Check that *list.of.dataframes* does indeed contain the three data frame objects.
- d. In Section 7.4.3, you created a *ggplot2* graphic of 20 observations displayed as the bottom image of Figure 7-11 on page 144. Use *ggsave* to save a copy of this plot as a *.tiff* file.

## Solution 8.1

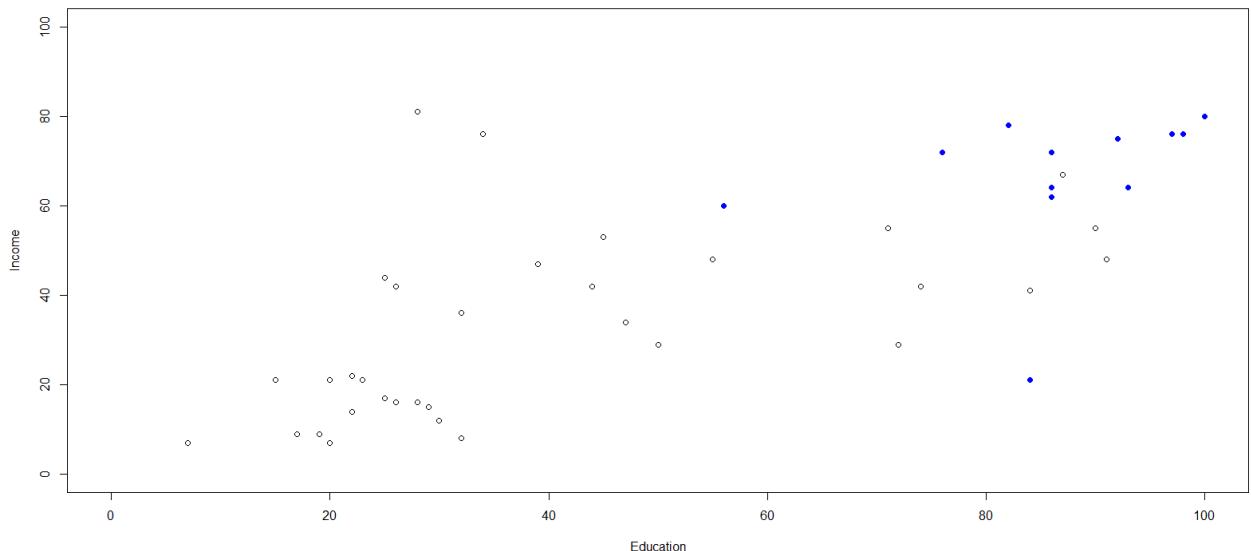
Pre-requisites: you should check your working directory before start of this exercise by executing command:

```
>getwd()

#(a)
##(i)
>
write.table(x=quakes[quakes$mag>=5,],file="/Users/user/Documents/q5.txt",sep="!",row.names=F)

##(ii)
>q5.dframe <- read.table(file="/Users/user/Documents/q5.txt",sep="!",header=T)
>q5.dframe
   lat long depth mag stations
1 -26.00 184.10  42 5.4    43
2 -20.70 169.92 139 6.1    94
3 -13.64 165.96  50 6.0    83
4 -19.66 180.28 431 5.4    57
5 -16.46 180.79 498 5.2    79
6 -18.97 185.25 129 5.1    73
7 -13.82 172.38 613 5.0    61
8 -21.96 179.62 627 5.0    45
9 -15.46 187.81  40 5.5    91
10 -23.74 179.99 506 5.2    75
11 -28.98 181.11 304 5.3    60

Continued...
#(b)
>install.packages("car")
library("car")
data(Duncan)
##(i)
plot(Duncan$education[Duncan$prestige<=80],Duncan$income[Duncan$prestige<=80],xlim=c(0,100),ylim=c(0,100),xlab="Education",ylab="Income")
points(Duncan$education[Duncan$prestige>80],Duncan$income[Duncan$prestige>80],pch=19,col="blue")
```



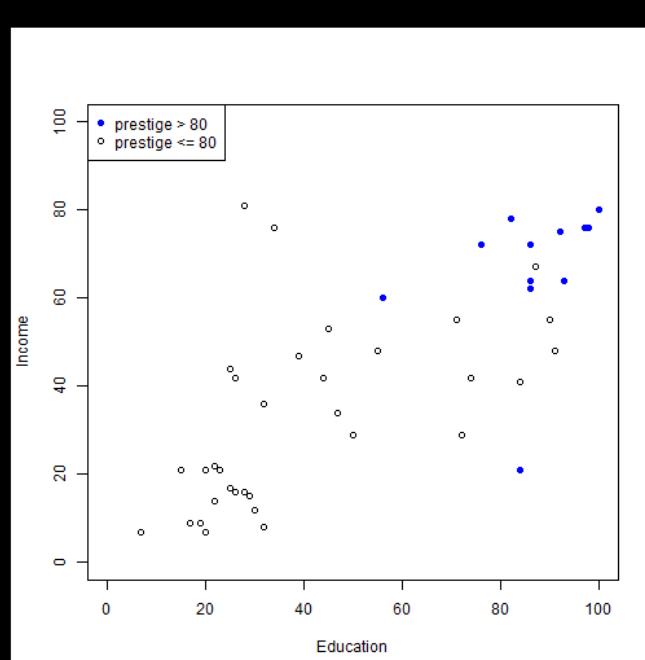
##(ii)

```

png("/Users/user/Documents/dunc.png",width=500,height=500)
plot(Duncan$education[Duncan$prestige<=80],Duncan$income[Duncan$prestige<=80],xlim=c(0,
100),ylim=c(0,100),xlab="Education",ylab="Income")
points(Duncan$education[Duncan$prestige>80],Duncan$income[Duncan$prestige>80],pch=19,co
l="blue")
legend("topleft",legend=c("prestige > 80","prestige <= 80"),pch=c(19,1),col=c("blue","black"))
dev.off()

```

Open working directory and check if file dunc.png was saved there

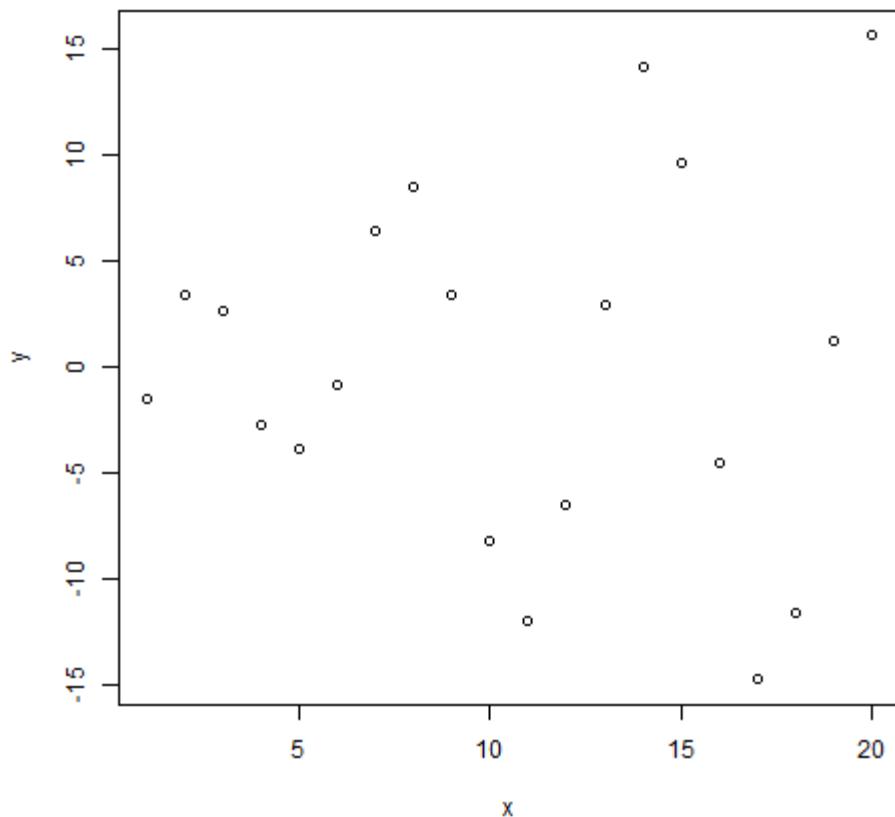


```

#(c)
exer <- list(quakes,q5.dframe,Duncan) #####!!!
##(i)
dput(x=exer,file="/Users/user/Documents/Exercise8-1Data.txt") #####
##(ii)
list.of.dataframes <- dget("/Users/user/Documents/Exercise8-1Data.txt")
list.of.dataframes
#(d)
x <- 1:20
y <- c(-1.49,3.37,2.59,-2.78,-3.94,-0.92,6.43,8.51,3.41,-8.23,-12.01,-6.58,2.87,14.12,9.63,-4.58,-14.78,-11.67,1.17,15.62)
ptype <- rep(NA,length(x=x))
ptype[y>=5] <- "too_big"
ptype[y<=-5] <- "too_small"
ptype[(x>=5&x<=15)&(y>-5&y<5)] <- "sweet"
ptype[(x<5|x>15)&(y>-5&y<5)] <- "standard"
ptype <- factor(x=ptype)
plot(x,y,color=ptype,shape=ptype) + geom_point(size=4) +
  geom_line(mapping=aes(group=1),color="black",lty=2) +
  geom_hline(mapping=aes(yintercept=c(-5,5)),color="red") +
  geom_segment(mapping=aes(x=5,y=-5,xend=5,yend=5),color="red",lty=3) +
  geom_segment(mapping=aes(x=15,y=-5,xend=15,yend=5),color="red",lty=3)
png(filename="/Users/user/Documents/elaborateqplot.tiff")

```

You should close your R studio and find file **elaborateqplot.tiff** in the folder



## Part 2

# Chapter 9: Calling Functions

Estimated time to complete: **30 minutes**

### Exercise 9.1

- a. Identify the first 20 items contained in the built-in and automatically loaded *methods* package. How many items are there in total?
- b. Determine the environment that owns each of the following functions:
  - i. `read.table`
  - ii. `data`
  - iii. `matrix`
  - iv. `jpeg`
- c. Use `ls` and a test for character string equality to confirm the function `smoothScatter` is part of the *graphics* package.

### Solution 9.1

```
> #(a)  
  
> ls("package:methods")[1:20]  
[1] "addNextMethod"      "allGenerics"       "allNames"        "Arith"          "as"  
"as<-"                 "asMethodDefinition" "assignClassDef"  
  
[9] "assignMethodsMetaData" "balanceMethodsList"  "body<-"  
"cacheGenericsMetaData"  "cacheMetaData"      "cacheMethod"     "callGeneric"  
"callNextMethod"  
  
[17] "canCoerce"         "cbind2"           "checkAtAssignment" "checkSlotAssignment"  
  
> length(ls("package:methods"))  
[1] 218  
  
> #(b)  
  
> ##(i)  
  
> environment(read.table)  
<environment: namespace:utils>  
  
> ##(ii)  
  
> environment(data)  
<environment: namespace:utils>  
  
> ##(iii)
```

```

> environment(matrix)
<environment: namespace:base>
> ##(iv)
> environment(jpeg)
<environment: namespace:grDevices>
> #(c)
> any(ls("package:graphics")=="smoothScatter")
[1] TRUE

```

## Exercise 9.2

- Use positional matching with *seq* to create a sequence of values between  $-4$  and  $4$  that progresses in steps of  $0.2$ .
- In each of the following lines of code, identify which style of argument matching is being used: exact, partial, positional, or mixed. If mixed, identify which arguments are specified in each style.
  - array(8:1,dim=c(2,2,2))*
  - rep(1:2,3)*
  - seq(from=10,to=8,length=5)*
  - sort(decreasing=T,x=c(2,1,1,2,0.3,3,1.3))*
  - which(matrix(c(T,F,T,T),2,2))*
  - which(matrix(c(T,F,T,T),2,2),a=T)*
- Suppose you explicitly ran the plotting function *plot.default* and supplied values to arguments tagged *type*, *pch*, *xlab*, *ylab*, *lwd*, *lty*, and *col*. Use the function documentation to determine which of these arguments fall under the umbrella of the ellipsis.

## Solution 9.2

```

> #(a)
> seq(-4,4,0.2)
[1] -4.0 -3.8 -3.6 -3.4 -3.2 -3.0 -2.8 -2.6 -2.4 -2.2 -2.0 -1.8 -1.6 -1.4 -1.2 -1.0 -0.8 -0.6 -0.4 -0.2
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0 3.2 3.4
[39] 3.6 3.8 4.0

```

```

> #(b)
> ##(i)
> array(8:1,dim=c(2,2,2)) # MIXED: 'data' positional, 'dim' exact.
, , 1

```

```

[,1] [,2]
[1,] 8 6
[2,] 7 5

```

```
, , 2
```

```
[,1] [,2]
```

```
[1,] 4 2
```

```
[2,] 3 1
```

```
> ##(ii)
```

```
> rep(1:2,3) # POSITIONAL
```

```
[1] 1 2 1 2 1 2
```

```
> ##(iii)
```

```
> seq(from=10,to=8,length=5) # MIXED: 'from' and 'to' exact, 'length.out' partial.
```

```
[1] 10.0 9.5 9.0 8.5 8.0
```

```
> ##(iv)
```

```
> sort(decreasing=T,x=c(2,1,1,2,0.3,3,1.3)) # EXACT
```

```
[1] 3.0 2.0 2.0 1.3 1.0 1.0 0.3
```

```
> ##(v)
```

```
> which(matrix(c(T,F,T,T),2,2)) # POSITIONAL
```

```
[1] 1 3 4
```

```
> ##(vi)
```

```
> which(matrix(c(T,F,T,T),2,2),a=T) # MIXED: 'x', 'data', 'nrow', 'ncol' positional, 'arr.ind' partial
```

```
row col
```

```
[1,] 1 1
```

```
[2,] 1 2
```

```
[3,] 2 2
```

```
> #(c)
```

```
> # 'pch', 'lwd', 'lty' and 'col' are part of the ellipsis.
```

```
>
```

# Chapter 10: Conditions and Loops

Estimated time to complete: **90 minutes**

## Exercise 10.1

- a. Create the following two vectors:

```
vec1 <- c(2,1,1,3,2,1,0)
vec2 <- c(3,8,2,2,0,0,0)
```

Without executing them, determine which of the following *if* statements would result in the string being printed to the console. Then confirm your answers in R.

- i. if((vec1[1]+vec2[2]==10){ cat("Print me!") }
  - ii. if(vec1[1]>=2&&vec2[1]>=2){ cat("Print me!") }
  - iii. if(all((vec2-vec1)[c(2,6)]<7)){ cat("Print me!") }
  - iv. if(!is.na(vec2[3])){ cat("Print me!") }
- b. Using *vec1* and *vec2* from (a), write and execute a line of code that multiplies the corresponding elements of the two vectors together *if* their sum is greater than 3. Otherwise, the code should simply sum the two elements.
- c. In the editor, write R code that takes a square character matrix and checks *if* any of the character strings on the diagonal (top left to bottom right) begin with the letter g, lowercase or uppercase.

If satisfied, these specific entries should be overwritten with the string "*HERE*". Otherwise, the entire matrix should be replaced with an identity matrix of the same dimensions. Then, try your code on the following matrices, checking the result each time:

- i. mymat <- matrix(as.character(1:16),4,4)
- ii. mymat <- matrix(c("DANDELION","Hyacinthus","Gerbera",
 "MARIGOLD","geranium","ligularia",
 "Pachysandra","SNAPDRAGON","GLADIOLUS"),3,3)
- iii. mymat <- matrix(c("GREAT","exercises","right","here"),2,2,
 byrow=T)

Hint: This requires some thought—you will find the functions *diag* from Section 3.2.1 and *substr* from Section 4.2.4 useful.

## Solution 10.1

```
> #(a)

  vec1 <- c(2,1,1,3,2,1,0)
  vec2 <- c(3,8,2,2,0,0,0)

> ##(i)

> if((vec1[1]+vec2[2]==10){ cat("Print me!") } # condition SATISFIED
Print me!>

##(ii)

> if(vec1[1]>=2&&vec2[1]>=2){cat("Print me!") } # condition SATISFIED
Print me!>

##(iii)

> if(all((vec2-vec1)[c(2,6)]<7)){cat("Print me!") } # condition NOT SATISFIED
> ##(iv)

> if(!is.na(vec2[3])){ cat("Print me!") } # condition SATISFIED
Print me!>

#(b)

> ifelse(vec1+vec2>3,vec1*vec2,vec1+vec2)
[1] 6 8 3 6 2 1 0

> #(c)

if(any(substr(diag(mymat),1,1)=="g")||any(substr(diag(mymat),1,1)=="G")){
  indexes <- which(substr(diag(mymat),1,1)=="g"|substr(diag(mymat),1,1)=="G")
  diag(mymat)[indexes] <- "HERE"
} else {
  mymat <- diag(nrow(mymat))
}

> mymat
 [,1] [,2] [,3] [,4]
[1,] 1 0 0 0
[2,] 0 1 0 0
[3,] 0 0 1 0
[4,] 0 0 0 1
```

```
> ##(i)

> mymat <- matrix(as.character(1:16),4,4)

> ##(ii)

> mymat <-
matrix(c("DANDELION","Hyacinthus","Gerbera","MARIGOLD","geranium","ligularia","Pachys
andra","SNAPDRAGON","GLADIOLUS"),3,3)

> ##(iii)

> mymat <- matrix(c("GREAT","exercises","right","here"),2,2,byrow=T)

> mymat

[,1] [,2]

[1,] "GREAT" "exercises"

[2,] "right" "here"
```

## Exercise 10.2

- a. Write an explicit stacked set of *if* statements that does the same thing as the integer version of the *switch* function illustrated earlier. Test it with *mynum* <- 3 and *mynum* <- 0, as in the text.
- b. Suppose you are tasked with computing the precise dosage amounts of a certain drug in a collection of hypothetical scientific experiments. These amounts depend upon some predetermined set of “dosage thresholds” (*lowdose*, *meddose*, and *highdose*), as well as a predetermined dose level factor vector named *doselevel*. Look at the following items (i–iv) to see the intended form of these objects. Then write a set of nested *if* statements that produce a new numeric vector called *dosage*, according to the following rules:

- First, *if* there are any instances of “*High*” in *doselevel*, perform the following operations:
  - \* Check *if lowdose* is greater than or equal to 10. If so, overwrite *lowdose* with 10; *otherwise*, overwrite *lowdose* by itself divided by 2.
  - \* Check *if meddose* is greater than or equal to 26. If so, overwrite *meddose* by 26.
  - \* Check *if highdose* is less than 60. If so, overwrite *highdose* with 60; *otherwise*, overwrite *highdose* by itself multiplied by 1.5.
  - \* Create a vector named *dosage* with the value of *lowdose* repeated (*rep*) to match the *length* of *doselevel*.
  - \* Overwrite the elements in *dosage* corresponding to the index positions of instances of “*Med*” in *doselevel* by *meddose*.
  - \* Overwrite the elements in *dosage* corresponding to the index positions of instances of “*High*” in *doselevel* by *highdose*.
- Otherwise (in other words, if there are no instances of “*High*” in *doselevel*), perform the following operations:
  - \* Create a new version of *doselevel*, a factor vector with levels “*Low*” and “*Med*” only, and label these with “*Small*” and “*Large*”, respectively (refer to Section 4.3 for details or see *?factor*).
  - \* Check to see *if lowdose* is less than 15 AND *meddose* is less than 35. If so, overwrite *lowdose* by itself multiplied by 2 and overwrite *meddose* by itself plus *highdose*.
  - \* Create a vector named *dosage*, which is the value of *lowdose* repeated (*rep*) to match the *length* of *doselevel*.
  - \* Overwrite the elements in *dosage* corresponding to the index positions of instances of “*Large*” in *doselevel* by *meddose*.

Now, confirm the following:

- i. Given

```
lowdose <- 12.5
meddose <- 25.3
highdose <- 58.1
doselevel <- factor(c("Low", "High", "High", "High", "Low", "Med",
"Med"), levels=c("Low", "Med", "High"))
```

the result of *dosage* after running the nested *if* statements is as follows:

```
R> dosage
[1] 10.0 60.0 60.0 60.0 10.0 25.3 25.3
```

ii. Using the same *lowdose*, *meddose*, and *highdose* thresholds as in ( i ) Given  
`doselevel <- factor(c("Low","Low","Low","Med","Low","Med","Med"),levels=c("Low","Med","High"))`

the result of *dosage* after running the nested *if* statements is as follows:

```
R> dosage
[1] 25.0 25.0 25.0 83.4 25.0 83.4 83.4
```

Also, *doselevel* has been overwritten as follows:

```
R> doselevel
```

```
[1] Small Small Small Large Small Large Large
Levels: Small Large
```

iii. Given

```
lowdose <- 9
meddose <- 49
highdose <- 61
doselevel <- factor(c("Low","Med","Med"),
levels=c("Low","Med","High"))
```

the result of *dosage* after running the nested *if* statements is as follows:

```
R> dosage
[1] 9 49 49
```

Also, *doselevel* has been overwritten as follows:

```
R> doselevel
[1] Small Large Large
Levels: Small Large
```

iv. Using the same *lowdose*, *meddose*, and *highdose* thresholds as (iii), as well as the same *doselevel* as (i), the result of *dosage* after running the nested *if* statements is as follows:

```
R> dosage
[1] 4.5 91.5 91.5 91.5 4.5 26.0 26.0
```

c. Assume the object *mynum* will only ever be a single integer between 0 and 9. Use *ifelse* and *switch* to produce a command that takes in *mynum* and returns a matching character string for all possible values 0, 1, . . . , 9. Supplied with 3, for example, it should return "three"; supplied with 0, it should return "zero".

## Solution 10.2

```
> mynum <- 3
> mynum <- 0
#(a)
>if (mynum==1) {
  foo <- 12
} else if (mynum==2) {
  foo <- 34
} else if (mynum==3) {
  foo <- 56
} else if (mynum==4) {
  foo <- 78
} else if (mynum==5) {
  foo <- NA
} else {
  foo <- NULL
}
> foo
NULL

> #(b)
if (any(doselevel=="High")) {
  if (lowdose>=10) {
    lowdose <- 10
  } else {
    lowdose <- lowdose/2
  }
  if (meddose>=26) {
    meddose <- 26
  }
  if (highdose<60) {
    highdose <- 60
  } else {
    highdose <- highdose*1.5
  }
  dosage <- rep(lowdose,length(doselevel))
  dosage[doselevel=="Med"] <- meddose
  dosage[doselevel=="High"] <- highdose
} else {
  doselevel <-
factor(doselevel,levels=c("Low","Med"),labels=c("Small","Large"))
  if (lowdose<15 && meddose<35) {
    lowdose <- lowdose*2
    meddose <- meddose+highdose
  }
  dosage <- rep(lowdose,length(doselevel))
  dosage[doselevel=="Large"] <- meddose
}
> dosage
[1] 10.0 60.0 60.0 60.0 10.0 25.3 25.3

>doselevel
[1] Low Low Low Med Low Med Med
```

*Levels: Low Med High*

```
##(i)
> lowdose <- 12.5
> meddose <- 25.3
> highdose <- 58.1
> doselevel <-
factor(c("Low","High","High","High","Low","Med","Med"),levels=c("Low","Med","High"))
> ##(ii)
> lowdose <- 12.5
> meddose <- 25.3
> highdose <- 58.1
> doselevel <-
factor(c("Low","Low","Low","Med","Low","Med","Med"),levels=c("Low","Med","High"))
> ##(iii)
> lowdose <- 9
> meddose <- 49
> highdose <- 61
> doselevel <- factor(c("Low","Med","Med"),levels=c("Low","Med","High"))
> ##(iv)
> lowdose <- 9
> meddose <- 49
> highdose <- 61
> doselevel <-
factor(c("Low","High","High","High","Low","Med","Med"),levels=c("Low","Med","High"))
> #(c)
> mynum <- 3
>
ifelse(mynum>0,switch(mynum,"one","two","three","four","five","six","seven","eight","nine"),"zero")
[1] "three"
> mynum <- 0
>
ifelse(mynum>0,switch(mynum,"one","two","three","four","five","six","seven","eight","nine"),"zero")
```

```
[1] "zero"
```

## Exercise 10.3

- a. In the interests of efficient coding, rewrite the nested loop example from this section, where the matrix *foo* was filled with the multiples of the elements of *loopvec1* and *loopvec2*, using only a single *for* loop.
- b. In Section 10.1.5, you used the command

```
switch(EXPR=mystring,Homer=12,Marge=34,Bart=56,Lisa=78,Maggie=90,NA)
```

to return a number based on the supplied value of a single character string. This line won't work if *mystring* is a character vector. Write some code that will take a character vector and return a vector of the appropriate numeric values. Test it on the following vector:

```
c("Peter","Homer","Lois","Stewie","Maggie","Bart")
```

- c. Suppose you have a list named *mylist* that can contain other lists as members, but assume those “member lists” cannot themselves contain lists. Write nested loops that can search any possible *mylist* defined in this way and count how many matrices are present. Hint: Simply set up a counter before commencing the loops that is incremented each time a matrix is found, regardless of whether it is a straightforward member of *mylist* or it is a member of a member list of *mylist*.

Then confirm the following:

- i. That the answer is 4 if you have the following:

```
mylist <- list(aa=c(3.4,1),bb=matrix(1:4,2,2),
cc=matrix(c(T,T,F,T,F,F),3,2),
dd="string here",
ee=list(c("hello","you"),matrix(c("hello","there"))),
ff=matrix(c("red","green","blue","yellow")))
```

- ii. That the answer is 0 if you have the following:

```
mylist <- list("tricked you",as.vector(matrix(1:6,3,2)))
```

- iii. That the answer is 2 if you have the following:

```
mylist <- list(list(1,2,3),list(c(3,2),2),
list(c(1,2),matrix(c(1,2))),
rbind(1:10,100:91))
```

## Solution 10.3

```
> #(a)  
> loopvec1 <- 5:7  
> loopvec2 <- 9:6  
> foo <- matrix(NA,length(loopvec1),length(loopvec2))
```

```

> for(i in 1:length(loopvec1)){
  foo[i,] <- loopvec1[i]*loopvec2
}
> foo
[,1] [,2] [,3] [,4]
[1,] 45 40 35 30
[2,] 54 48 42 36
[3,] 63 56 49 42
> #(b)

> mystrings
[1] "Peter"   "Homer"   "Lois"     "Stewie"   "Maggie"   "Bart"

mynums <- rep(NA,length(mystrings))
for(i in 1:length(mystrings)){
  mynums[i] <-
switch(EXPR=mystrings[i],Homer=12,Marge=34,Bart=56,Lisa=78,Maggie=90,NA)
}
> mynums
[1] NA 12 NA NA 90 56
#(c)
> counter <- 0
for(i in 1:length(mylist)){
  member <- mylist[[i]]
  if(is.matrix(member)){
    counter <- counter+1
  } else if(is.list(member)){
    for(j in 1:length(member)){
      if(is.matrix(member[[j]])){
        counter <- counter+1
      }
    }
  }
}
> member
[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,] 1 2 3 4 5 6 7 8 9 10
[2,] 100 99 98 97 96 95 94 93 92 91
##(i)
mylist <-
list(aa=c(3.4,1),bb=matrix(1:4,2,2),cc=matrix(c(T,T,F,T,F,F),3,2),dd="string
here",ee=list(c("hello","you"),matrix(c("hello","there"))),ff=matrix(c(
"red","green","blue","yellow")))
##(ii)
mylist <- list("tricked you",as.vector(matrix(1:6,3,2)))
##(iii)

> mylist <-
list(list(1,2,3),list(c(3,2),2),list(c(1,2),matrix(c(1,2))),rbind(1:10,
100:91))
> mylist

```

```

[[1]]
[[1]][[1]]
[1] 1

[[1]][[2]]
[1] 2

[[1]][[3]]
[1] 3

[[2]]
[[2]][[1]]
[1] 3 2

[[2]][[2]]
[1] 2

[[3]]
[[3]][[1]]
[1] 1 2

[[3]][[2]]
[,1]
[1,] 1
[2,] 2

[[4]]
[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,] 1 2 3 4 5 6 7 8 9 10
[2,] 100 99 98 97 96 95 94 93 92 91

```

## Exercise 10.4

- a. Based on the most recent example of storing identity matrices in a list, determine what the resulting *mylist* would look like for each of the following possible *mynumbers* vectors, without executing anything:

- i. *mynumbers* <- *c*(2,2,2,2,5,2)
- ii. *mynumbers* <- 2:20
- iii. *mynumbers* <- *c*(10,1,10,1,2)

Then, confirm your answers in R (note you'll also have to reset the initial values of *mylist*, *counter*, and *mycondition* each time, just as in the text).

- b. For this problem, I'll introduce the *factorial* operator. The factorial of a non-negative integer *x*, expressed as *x!*, refers to *x* multiplied by the product of all integers less than *x*, down to 1.

Formally, it is written like this:

$$\text{"x factorial"} = x! = x \times (x - 1) \times (x - 2) \times \dots \times 1$$

Note that there is a special case of *zero factorial*, which is always 1. That is:  
 $0! = 1$

For example, to work out 3 factorial, you compute the following:

$$3 \times 2 \times 1 = 6$$

To work out 7 factorial, you compute the following:

$$7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 5040$$

Write a *while* loop that computes and stores as a new object the factorial of any non-negative integer *mynum* by decrementing *mynum* by 1 at each repetition of the braced code.

Using your loop, confirm the following:

- i. That the result of using *mynum* <- 5 is 120
- ii. That using *mynum* <- 12 yields 479001600
- iii. That having *mynum* <- 0 correctly returns 1
- c. Consider the following code, where the operations in the braced area of the *while* loop have been omitted:

```
mystring <- "Rfever"  
index <- 1  
ecount <- 0  
result <- mystring  
while(ecount<2 && index<=nchar(mystring)){  
# several omitted operations #  
}  
result
```

Your task is to complete the code in the braced area so it inspects *mystring* character by character until it reaches the second instance of the letter *e* or the end of the string, whichever comes first. The *result* object should be the entire character string if there is no second *e* or the character string made up of all the characters up to, but not including, the second *e* if there is one. For example, *mystring* <- "Rfever" should provide *result* as "Rfev". This must be achieved by following these operations in the braces:

1. Use *substr* (Section 4.2.4) to extract the single character of *mystring* at position *index*.

2. Use a check for equality to determine whether this singlecharacter string is either "*e*" OR "*E*". If so, increase *ecount* by 1.
3. Next, perform a separate check to see whether *ecount* is equal to 2. If so, use *substr* to set *result* equal to the characters between *l* and *index-1* inclusive.
4. Increment *index* by 1.

Test your code—ensure the previous *result* for *mystring* <- "Rfever". Furthermore, confirm the following:

- Using *mystring* <- "beautiful" provides *result* as "beautiful"
- Using *mystring* <- "ECCENTRIC" provides *result* as "ECC"
- Using *mystring* <- "ElAbOrAte" provides *result* as "ElAbOrAt"
- Using *mystring* <- "eeeeek!" provides *result* as "e"

## Solution 10.4

```
> #(a)  
> ##(1)  
  
>mylist <- list()  
counter <- 1  
mynumbers <- c(2,2,2,2,5,2)  
mycondition <- mynumbers[counter]<=5  
mycondition  
while (mycondition) {  
  mylist[[counter]] <- diag(mynumbers[counter])  
  counter <- counter+1  
  if(counter<=length(mynumbers)) {  
    mycondition <- mynumbers[counter]<=5  
  } else {  
    mycondition <- FALSE  
  }  
}  
>mylist  
[[1]]  
 [,1] [,2]  
[1,] 1 0  
[2,] 0 1  
  
[[2]]  
 [,1] [,2]  
[1,] 1 0  
[2,] 0 1  
  
[[3]]  
 [,1] [,2]  
[1,] 1 0  
[2,] 0 1  
  
[[4]]  
 [,1] [,2]  
[1,] 1 0  
[2,] 0 1
```

[[5]]

```
[,1] [,2] [,3] [,4] [,5]  
[1,] 1 0 0 0 0  
[2,] 0 1 0 0 0  
[3,] 0 0 1 0 0  
[4,] 0 0 0 1 0  
[5,] 0 0 0 0 1
```

[[6]]

```
[,1] [,2]  
[1,] 1 0  
[2,] 0 1
```

#(ii)

```
>mylist <- list()  
counter <- 1  
mynumbers <- 2:20  
mycondition <- mynumbers[counter]<=5  
mycondition  
[1] TRUE  
while (mycondition) {  
  mylist[[counter]] <- diag(mynumbers[counter])  
  counter <- counter+1  
  if(counter<=length(mynumbers)) {  
    mycondition <- mynumbers[counter]<=5  
  } else {  
    mycondition <- FALSE  
  }  
}  
> mylist  
[[1]]
```

```
[,1] [,2]  
[1,] 1 0  
[2,] 0 1
```

[[2]]

```
[,1] [,2] [,3]
```

```
[1,] 1 0 0
```

```
[2,] 0 1 0
```

```
[3,] 0 0 1
```

```
[[3]]
```

```
,1] [,2] [,3] [,4]
```

```
[1,] 1 0 0 0
```

```
[2,] 0 1 0 0
```

```
[3,] 0 0 1 0
```

```
[4,] 0 0 0 1
```

```
[[4]]
```

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

```
[1,] 1 0 0 0 0
```

```
[2,] 0 1 0 0 0
```

```
[3,] 0 0 1 0 0
```

```
[4,] 0 0 0 1 0
```

```
[5,] 0 0 0 0 1
```

##(iii) below, the loop braced-area code won't even be entered -- the first element of 'mynumbers' is greater than 5 -- resulting list will be empty

```
mylist <- list()
counter <- 1
mynumbers <- c(10,1,10,1,2)
mycondition <- mynumbers[counter]<=5
mycondition
[1] FALSE
while(mycondition){
  mylist[[counter]] <- diag(mynumbers[counter])
  counter <- counter+1
  if(counter<=length(mynumbers)){
    mycondition <- mynumbers[counter]<=5
  } else {
    mycondition <- FALSE
  }
}
mylist
```

```

list()

# (b)
mynum.fac <- 1
while(mynum>1) {
  mynum.fac <- mynum.fac*mynum
  mynum <- mynum-1
}
Error in mynum : object 'mynum' not found

> mynum.fac

[1] 1

##(i)

mynum <- 5

##(ii)

mynum <- 12

> mynum
[1] 12
##(iii)
>mynum <- 0
> mynum

[1] 0

# (c)

mystring <- "R fever"
mystring <- "beautiful"
mystring <- "ECCENTRIC"
mystring <- "ElAbOrAte"
mystring <- "eeeeek!"

> mystring

[1] "eeeeek!"

#--#
index <- 1
ecount <- 0
result <- mystring
while(ecount<2 && index<=nchar(mystring)) {
  temp.char <- substr(mystring,index,index)
  if(temp.char=="e"||temp.char=="E") {
    ecount <- ecount+1
  }
  if(ecount==2) {
    result <- substr(mystring,1,index-1)
  }
  index <- index + 1
}

```

```
 }  
result  
[1] "e"
```

## Exercise 10.5

- a. Continuing on from the most recent example in the text, write an implicit loop that calculates the product of all the column elements of the matrix returned by the call to `apply(foo, 1, sort, decreasing=TRUE)`.
- b. Convert the following `for` loop to an implicit loop that does exactly the same thing:

```
matlist <- list(matrix(c(T,F,T,T),2,2),
                 matrix(c("a","c","b","z","p","q"),3,2),
                 matrix(1:8,2,4))
```

```
matlist
for(i in 1:length(matlist)){
  matlist[[i]] <- t(matlist[[i]]) }
```

```
matlist
```

- c. In R, store the following  $4 \times 4 \times 2 \times 3$  array as the object `qux`:

```
R> qux <- array(96:1,dim=c(4,4,2,3))
```

That is, it is a four-dimensional array comprised of three blocks, with each block being an array made up of two layers of  $4 \times 4$  matrices. Then, do the following:

- i. Write an implicit loop that obtains the diagonal elements of all second-layer matrices only to produce the following matrix:

```
[,1] [,2] [,3]
[1,] 80 48 16
[2,] 75 43 11
[3,] 70 38 6
[4,] 65 33 1
```

- i. Write an implicit loop that will return the *dimensions* of each of the three matrices formed by accessing the fourth column of every matrix in `qux`, regardless of layer or block, wrapped by another implicit loop that finds the row sums of that returned structure, resulting simply in the following vector:

```
[1] 12 6
```

## Solution 10.5

```
> #(a)

> foo <- matrix(1:12,4,3)

> apply(apply(foo,1,sort,decreasing=TRUE),2,prod)

[1] 45 120 231 384

> #(b)

> matlist <- list(matrix(c(T,F,T,T),2,2),matrix(c("a","c","b","z","p","q"),3,2),matrix(1:8,2,4))

> matlist

[[1]]

[,1] [,2]

[1,] TRUE TRUE

[2,] FALSE TRUE
```

```
[[2]]  
[,1] [,2]  
[1,] "a" "z"  
[2,] "c" "p"  
[3,] "b" "q"
```

```
[[3]]  
[,1] [,2] [,3] [,4]  
[1,] 1 3 5 7  
[2,] 2 4 6 8
```

```
> for(i in 1:length(matlist)){  
  matlist[[i]] <- t(matlist[[i]])  
}  
> matlist  
[[1]]  
[,1] [,2]  
[1,] TRUE FALSE  
[2,] TRUE TRUE
```

```
[[2]]  
[,1] [,2] [,3]  
[1,] "a" "c" "b"  
[2,] "z" "p" "q"
```

```
[[3]]  
[,1] [,2]  
[1,] 1 2  
[2,] 3 4  
[3,] 5 6  
[4,] 7 8
```

```
> matlist <- list(matrix(c(T,F,T,T),2,2),matrix(c("a","c","b","z","p","q"),3,2),matrix(1:8,2,4))
```

```
> matlist
```

```
[[1]]
```

```
 [,1] [,2]
```

```
[1,] TRUE TRUE
```

```
[2,] FALSE TRUE
```

```
[[2]]
```

```
 [,1] [,2]
```

```
[1,] "a" "z"
```

```
[2,] "c" "p"
```

```
[3,] "b" "q"
```

```
[[3]]
```

```
 [,1] [,2] [,3] [,4]
```

```
[1,] 1 3 5 7
```

```
[2,] 2 4 6 8
```

```
> matlist <- lapply(matlist,t)
```

```
> matlist
```

```
[[1]]
```

```
 [,1] [,2]
```

```
[1,] TRUE FALSE
```

```
[2,] TRUE TRUE
```

```
[[2]]
```

```
 [,1] [,2] [,3]
```

```
[1,] "a" "c" "b"
```

```
[2,] "z" "p" "q"
```

```
[[3]]
```

```

[,1] [,2]
[1,] 1 2
[2,] 3 4
[3,] 5 6
[4,] 7 8

> #(c)
> qux <- array(96:1,dim=c(4,4,2,3))
> ##(i)
> apply(qux[,2,],3,diag)
[,1] [,2] [,3]
[1,] 80 48 16
[2,] 75 43 11
[3,] 70 38 6
[4,] 65 33 1
> ##(ii)
> apply(apply(qux[,4,,],3,dim),1,sum)
[1] 12 6

```

## Exercise 10.6

- a. Using the same objects from Section 10.3.1,

```
foo <- 5  
bar <- c(2,3,1,1,4,0,4,1,3)  
do the following:
```

- i. Write a *while* loop—*without* using *break* or *next*—that will reach exactly the same result as the *break* example in Section 10.3.1. That is, produce the same vector as *loop2.result* in the text.
  - ii. Obtain the same result as *loop3.result*, the example concerning *next*, using an *ifelse* function instead of a loop.
- b. To demonstrate *while* loops in Section 10.2.2, you used the vector *mynumbers* <- *c*(4,5,1,2,6,2,4,6,6,2) to progressively fill *mylist* with identity matrices whose dimensions matched the values in *mynumbers*. The loop was instructed to stop when it reached the end of the numeric vector or a number that was greater than 5.
- i. Write a *for* loop using a *break* declaration that does the same thing.
  - ii. Write a *repeat* statement that does the same thing.
- c. Suppose you have two lists, *matlist1* and *matlist2*, both filled with numeric matrices as their members. Assume that all members have finite, nonmissing values, but *do not* assume that the dimensions of the matrices are the same throughout. Write a nested pair of *for* loops that aim to create a result list, *reslist*, of all possible *matrix products* (refer to Section 3.3.5) of the members of the two lists according to the following guidelines:
- The *matlist1* object should be indexed/searched in the outer loop, and the *matlist2* object should be indexed/searched in the inner loop.
  - You’re interested only in the possible matrix products of the members of *matlist1* with the members of *matlist2* in that order.
  - If a particular multiple isn’t possible (that is, if the *ncol* of a member of *matlist1* doesn’t match the *nrow* of a member of *matlist2*), then you should skip that multiplication, store the string “*not possible*” at the relevant position in *reslist*, and proceed directly to the *next* matrix multiplication.
  - You can define a *counter* that is incremented at each comparison (inside the inner loop) to keep track of the current position of *reslist*.

Note, therefore, that the *length* of *reslist* will be equal to *length(matlist1)\*length(matlist2)*. Now, confirm the following results:

- i. If you have

```
matlist1 <- list(matrix(1:4,2,2),matrix(1:4),matrix(1:8,4,2))  
matlist2 <- matlist1
```

then all members of *reslist* should be “*not possible*” apart from members [[1]] and [[7]].

- ii. If you have

```
matlist1 <- list(matrix(1:4,2,2),matrix(2:5,2,2),  
matrix(1:16,4,2))
```

```

matlist2 <- list(matrix(1:8,2,4),matrix(10:7,2,2),
matrix(9:2,4,2))
then only the "not possible" members of reslist should be
[[3]], [[6]], and [[9]].
```

## Solution 10.6

```

> #(a)

> foo <- 5

> foo

[1] 5

> bar <- c(2,3,1.1,4,0,4.1,3)

> bar

[1] 2.0 3.0 1.1 4.0 0.0 4.1 3.0

> ##(i)

> loop2.result <- rep(NA,length(bar) )
condition <- TRUE
counter <- 1
while(condition) {
  temp <- foo/bar[counter]
  if(is.finite(temp)){
    loop2.result[counter] <- temp
  } else {
    condition <- FALSE
  }
  counter <- counter+1
}
> loop2.result

[1] 2.500000 1.666667 4.545455 1.250000     NA     NA     NA

> ##(ii)

> loop3.result <- ifelse(is.finite(foo/bar),foo/bar,NA)

> loop3.result

[1] 2.500000 1.666667 4.545455 1.250000     NA 1.219512 1.666667

> #(b)

> mynumbers <- c(4,5,1,2,6,2,4,6,6,2)

> mynumbers

[1] 4 5 1 2 6 2 4 6 6 2

> ##(i)

> mylist <- list()
```

```

>for(i in 1:length(mynumbers)) {
  if(mynumbers[i]<=5) {
    mylist[[i]] <- diag(mynumbers[i])
  } else {
    break
  }
}
>mylist
[[1]]
 [,1] [,2] [,3] [,4]
[1,] 1 0 0 0
[2,] 0 1 0 0
[3,] 0 0 1 0
[4,] 0 0 0 1

[[2]]
 [,1] [,2] [,3] [,4] [,5]
[1,] 1 0 0 0 0
[2,] 0 1 0 0 0
[3,] 0 0 1 0 0
[4,] 0 0 0 1 0
[5,] 0 0 0 0 1

[[3]]
 [,1]
[1,] 1

[[4]]
 [,1] [,2]
[1,] 1 0
[2,] 0 1

>##(ii)
> mylist <- list()
>counter <- 0
repeat{

```

```
counter <- counter+1
if(counter<=length(mynumbers)) {
  if(mynumbers[counter]<=5) {
    mylist[[counter]] <- diag(mynumbers[counter])
  } else {
    break
  }
} else {
  break
}
}
> mylist
```

[[1]]

```
[,1] [,2] [,3] [,4]
[1,] 1 0 0 0
[2,] 0 1 0 0
[3,] 0 0 1 0
[4,] 0 0 0 1
```

[[2]]

```
[,1] [,2] [,3] [,4] [,5]
[1,] 1 0 0 0 0
[2,] 0 1 0 0 0
[3,] 0 0 1 0 0
[4,] 0 0 0 1 0
[5,] 0 0 0 0 1
```

[[3]]

```
[,1]
[1,] 1
```

[[4]]

```
[,1] [,2]
[1,] 1 0
[2,] 0 1
```

```
> #(c)
```

```
>matlist1 <- list(matrix(1:4,2,2),matrix(1:4),matrix(1:8,4,2))
>matlist2 <- matlist1
```

```
>counter <- 0
reslist <- list()
for(i in 1:length(matlist1)){
  for(j in 1:length(matlist2)){
    counter <- counter+1
    if(ncol(matlist1[[i]])!=nrow(matlist2[[j]])){
      reslist[[counter]] <- "not possible"
      next
    }
    reslist[[counter]] <- matlist1[[i]]%*%matlist2[[j]]
  }
}
```

```
Error in length(matlist1) :object 'matlist1' not found
```

```
> reslist
```

```
[[1]]
```

```
 [,1] [,2]
```

```
[1,] 7 15
```

```
[2,] 10 22
```

```
[[2]]
```

```
[1] "not possible"
```

```
[[3]]
```

```
[1] "not possible"
```

```
[[4]]
```

```
[1] "not possible"
```

```
[[5]]
```

```
[1] "not possible"
```

```
[[6]]
```

```
[1] "not possible"
```

```
[[7]]  
 [,1] [,2]  
[1,] 11 23  
[2,] 14 30  
[3,] 17 37  
[4,] 20 44
```

```
[[8]]  
[1] "not possible"
```

```
[[9]]  
[1] "not possible" > ##(i)  
> matlist1 <- list(matrix(1:4,2,2),matrix(1:4),matrix(1:8,4,2))  
> matlist2 <- matlist1  
>matlist2
```

```
[[1]]  
 [,1] [,2]  
[1,] 1 3  
[2,] 2 4
```

```
[[2]]  
 [,1]  
[1,] 1  
[2,] 2  
[3,] 3  
[4,] 4
```

```
[[3]]  
 [,1] [,2]  
[1,] 1 5
```

```

[2,] 2 6
[3,] 3 7
[4,] 4 8
> ##(ii)

> matlist1 <- list(matrix(1:4,2,2),matrix(2:5,2,2),matrix(1:16,4,2))
> matlist1

[[1]]
 [,1] [,2]
[1,] 1 3
[2,] 2 4

[[2]]
 [,1] [,2]
[1,] 2 4
[2,] 3 5

[[3]]
 [,1] [,2]
[1,] 1 5
[2,] 2 6
[3,] 3 7
[4,] 4 8

> matlist2 <- list(matrix(1:8,2,4),matrix(10:7,2,2),matrix(9:2,4,2))
> matlist2

[[1]]
 [,1] [,2]
[1,] 1 3
[2,] 2 4

[[2]]
 [,1]

```

[1,] 1

[2,] 2

[3,] 3

[4,] 4

[[3]]

[,1] [,2]

[1,] 1 5

[2,] 2 6

[3,] 3 7

[4,] 4 8

# Chapter 11: Writing Functions

Estimated time to complete: **30 minutes**

## Exercise 11.1

- a. Write another Fibonacci sequence function, naming it *myfib4*.

This function should provide an option to perform either the operations available in *myfib2*, where the sequence is simply printed to the console, or the operations in *myfib3*, where a vector of the sequence is formally returned. Your function should take two arguments: the first, *thresh*, should define the limit of the sequence (just as in *myfib2* or *myfib3*); the second, *printme*, should be a logical value. If *TRUE*, then *myfib4* should just print; if *FALSE*, then *myfib4* should return a vector. Confirm the correct results arise from the following calls:

- *myfib4(thresh=150,printme=TRUE)*
- *myfib4(1000000,T)*
- *myfib4(150,FALSE)*
- *myfib4(1000000,printme=F)*

- b. In Exercise 10.4 on page 203, you were tasked with writing a *while* loop to perform integer factorial calculations.
- i. Using your factorial *while* loop (or writing one if you didn't do so earlier), write your own R function, *myfac*, to compute the factorial of an integer argument *int* (you may assume *int* will always be supplied as a non-negative integer). Perform a quick test of the function by computing 5 factorial, which is 120; 12 factorial, which is 479,001,600; and 0 factorial, which is 1.
  - ii. Write another version of your factorial function, naming it *myfac2*. This time, you may still assume *int* will be supplied as an integer but not that it will be non-negative. If negative, the function should return *NaN*. Test *myfac2* on the same three values as previously, but also try using *int=-6*.

## Solution 11.1

> #(a)

```
>myfib4 <- function(thresh,printme) {
  if(printme) {
    fib.a <- 1
    fib.b <- 1
    cat(fib.a," ",fib.b," ",sep="")
    repeat{
      temp <- fib.a+fib.b
      fib.a <- fib.b
      fib.b <- temp
      cat(fib.b," ",sep="")
      if(fib.b>thresh){
        cat("BREAK NOW...")
        break
      }
    }
  } else {
    fibseq <- c(1,1)
    counter <- 2
    repeat{
      fibseq <- c(fibseq,fibseq[counter-1]+fibseq[counter])
      counter <- counter+1
      if(fibseq[counter]>thresh){
        break
      }
    }
    return(fibseq)
  }
}
> myfib4(thresh=150,printme=TRUE)
```

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, BREAK NOW...> myfib4(1000000,T)

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393, 196418, 317811, 514229, 832040, 1346269, BREAK NOW...> myfib4(150,FALSE)

[1] 1 1 2 3 5 8 13 21 34 55 89 144 233

> myfib4(1000000,printme=F)

```
[1] 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
987 1597 2584 4181 6765 10946 17711 28657 46368
```

[25] 75025 121393 196418 317811 514229 832040 1346269

> #(b)

> ##(i)

```
> myfac <- function(int) {
  result <- 1
  while(int>1){
    result <- result*int
    int <- int-1
```

```
    }
    return(result)
}
> myfac(5)
[1] 120

> myfac(12)
[1] 479001600

> myfac(0)
[1] 1

> ##(ii)

> myfac2 <- function(int) {
+   if(int<0) {
+     return(NaN)
+   }
+   result <- 1
+   while(int>1) {
+     result <- result*int
+     int <- int-1
+   }
+   return(result)
}
>myfac2(5)
[1] 120

> myfac2(12)
[1] 479001600

> myfac2(0)
[1] 1

> myfac2(-6)
[1] NaN
```

## Exercise 11.2

a. Accruing annual compound interest is a common financial benefit for investors. Given a principal investment amount  $P$ , an interest rate per annum  $i$  (expressed as a percentage), and a frequency of interest paid per year  $t$ , the final amount  $F$  after  $y$  years is given as follows:

$$F = P \left(1 + \frac{i}{100t}\right)^{ty}$$

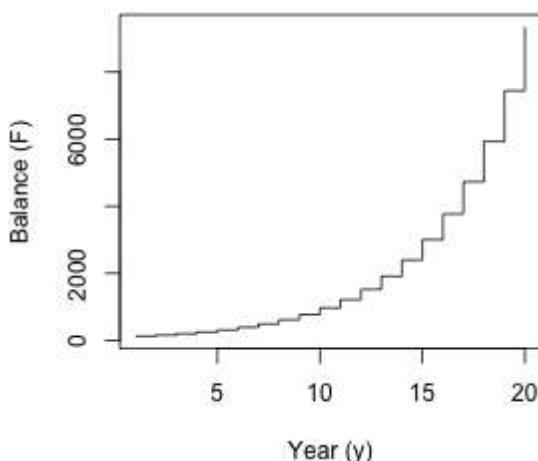
Write a function that can compute  $F$  as per the following notes:

- Arguments must be present for  $P$ ,  $i$ ,  $t$ , and  $y$ . The argument for  $t$  should have a default value of 12.
- Another argument giving a logical value that determines whether to *plot* the amount  $F$  at each integer time should be included. For example, if *plotit*=TRUE (the default) and  $y$  is 5 years, the plot should show the amount  $F$  at  $y = 1, 2, 3, 4, 5$ .
- If this function is plotted, the plot should always be a stepplot, so *plot* should always be called with *type*="s".
- If *plotit*=FALSE, the final amount  $F$  should be returned as a numeric vector corresponding to the same integer times, as shown earlier.
- An ellipsis should also be included to control other details of plotting, if it takes place.

Now, using your function, do the following:

- i. Work out the final amount after a 10-year investment of a principal of \$5000, at an interest rate of 4.4 percent per annum compounded monthly.
- ii. Re-create the following step-plot, which shows the result of \$100 invested at 22.9 percent per annum, compounded monthly, for 20 years:

**Compound interest calculator**



- iii. Perform another calculation based on the same parameters as in (ii), but this time, assume the interest is compounded annually. Return and store the results as a numeric vector.
  - iv. Then, use *lines* to add a second step-line, corresponding to this annually accrued amount, to the plot created previously. Use a different color or line type and make use of the *legend* function so the two lines can be differentiated.
- b. A *quadratic equation* in the variable  $x$  is often expressed in the following form:

$$k_1x^2 + k_2x + k_3 = 0$$

Here,  $k_1$ ,  $k_2$ , and  $k_3$  are constants. Given values for these constants, you can attempt to find up to two *real roots*—values of  $x$  that satisfy the equation. Write a function that takes  $k_1$ ,  $k_2$ , and  $k_3$  as arguments and finds and returns any solutions (as a numeric vector) in such a situation. This is achieved as follows:

- Evaluate  $k_2^2 - 4k_1k_3$ . If this is negative, there are no solutions, and an appropriate message should be printed to the console.
- If  $k_2^2 - 4k_1k_3$  is zero, then there is one solution, computed by  $-k_2/2k_1$ .
- If  $k_2^2 - 4k_1k_3$  is positive, then there are two solutions, given by  $(-k_2 - (k_2^2 - 4k_1k_3)^{0.5})/2k_1$  and  $(-k_2 + (k_2^2 - 4k_1k_3)^{0.5})/2k_1$ .
- No default values are needed for the three arguments, but the function should check to see whether any are missing.

If so, an appropriate character string message should be returned to the user, informing the user that the calculations are not possible.

Now, test your function.

- i. Confirm the following:

\*  $2x^2 - x - 5$  has roots 1.850781 and -1.350781.

\*  $x^2 + x + 1$  has no real roots.

- ii. Attempt to find solutions to the following quadratic equations:

\*  $1.3x^2 - 8x - 3.13$

\*  $2.25x^2 - 3x + 1$

\*  $1.4x^2 - 2.2x - 5.1$

\*  $-5x^2 + 10.11x - 9.9$

- iii. Test your programmed response in the function if one of the arguments is missing.

## Solution 11.2

```
> #(a)

> comp <- function(P,i,t=12,y,plotit=TRUE,...) {
  yseq <- 1:y
  values <- P*(1+i/(100*t))^^(t*yseq)

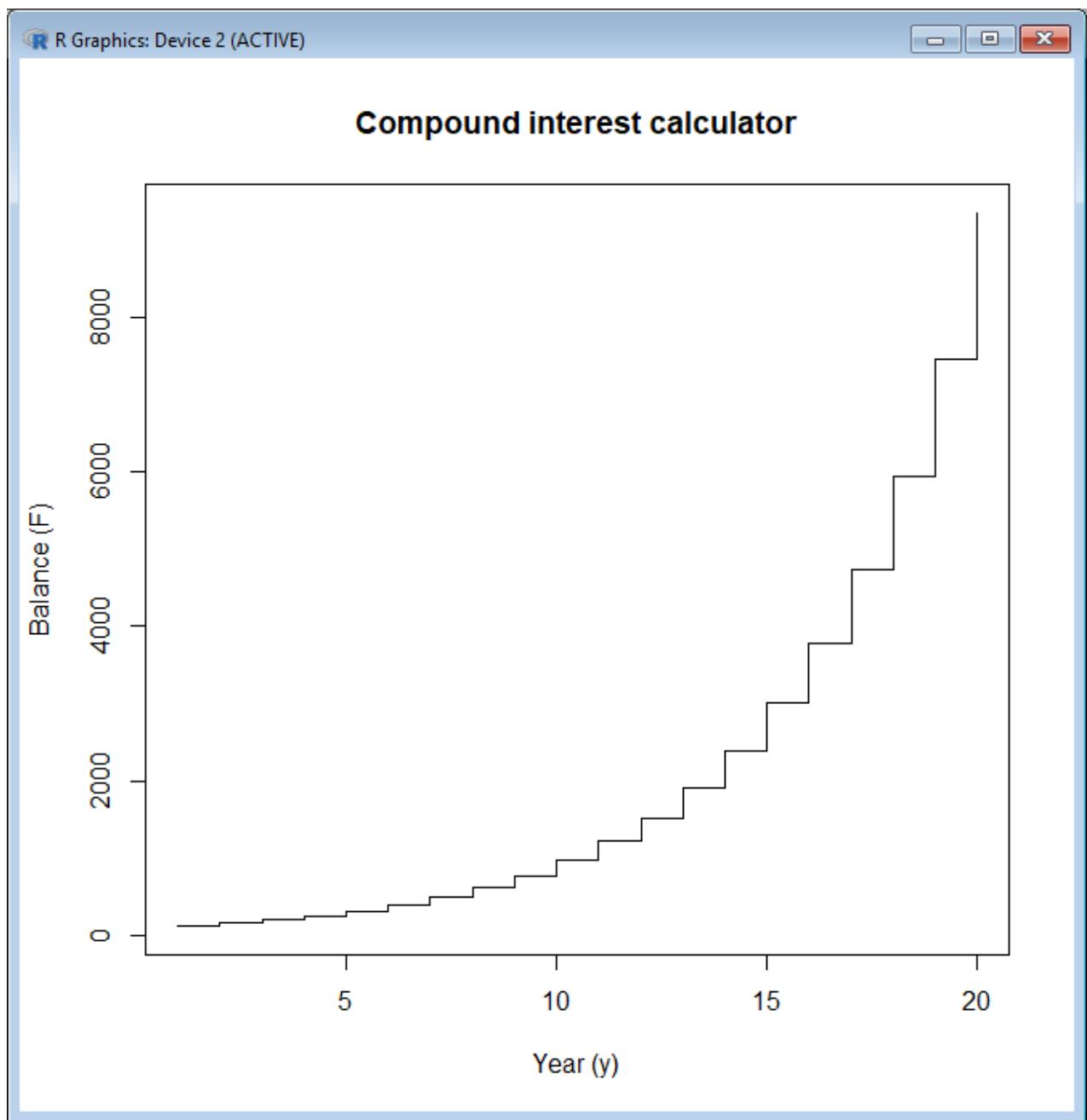
  if(plotit) {
    plot(yseq,values,type="s",...)
  } else {
    return(values)
  }
}

> ##(i)

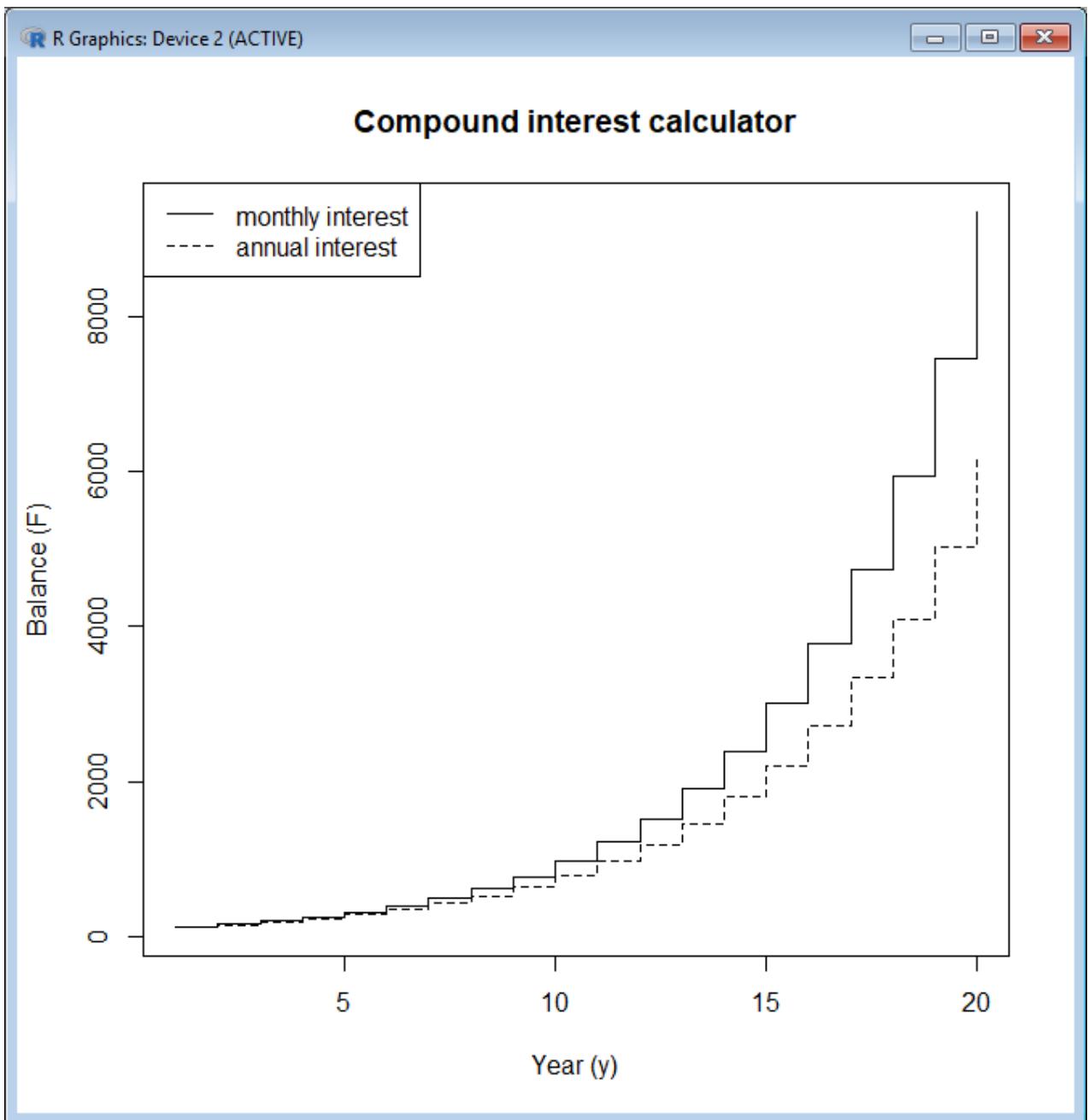
> comp(5000,4.4,y=10,plotit=F)[10]
[1] 7757.291

> ##(ii)

> comp(100,22.9,12,20,plotit=T,main="Compound interest calculator",ylab="Balance
(F)",xlab="Year (y)")
```



```
> ##(iii)  
> ann <- comp(100,22.9,1,20,plotit=F)  
> lines(1:20,ann,lty=2,type="s")  
> legend("topleft",lty=c(1,2),legend=c("monthly interest","annual interest"))
```



```
> #(b)
```

```
> quad <- function(k1,k2,k3) {
  if(any(c(missing(k1),missing(k2),missing(k3)))) {
    return("At least one of k1, k2, k3 was missing")
  }
  x <- k2^2-4*k1*k3
  if(x<0) {
    cat("No real roots\n")
  } else if(x==0) {
    return(-k2/(2*k1))
  } else {
    return(c((-k2-x^0.5)/(2*k1), (-k2+x^0.5)/(2*k1)))
  }
}
```

```
> ##(i)
> quad(k1=2,k2=-1,k3=-5)
[1] -1.350781 1.850781
> quad(1,1,1)
No real roots
> ##(ii)
> quad(k1=1.3,k2=-8,k3=-3.13)
[1] -0.3691106 6.5229567
> quad(2.25,-3,1)
[1] 0.6666667
> quad(1.4,-2.2,-5.1)
[1] -1.278312 2.849740
> quad(-5,10.11,-9.9)
No real roots
> ##(iii)
> quad(0)
[1] "At least one of k1, k2, k3 was missing"
```

## Exercise 11.3

- a. Given a list whose members are character string vectors of varying lengths, use a disposable function with *lapply* to paste an exclamation mark onto the end of each element of each member, with an empty string as the separation character (note that the default behavior of *paste* when applied to character vectors is to perform the concatenation on each element). Execute your line of code on the list given by the following:

```
foo <- list("a",c("b","c","d","e"),"f",c("g","h","i"))
```

- b. Write a recursive version of a function implementing the non-negative integer factorial operator (see Exercise 10.4 on page 203 for details of the factorial operator). The stopping rule should return the value 1 if the supplied integer is 0. Confirm that your function produces the same results as earlier.

- i. 5 factorial is 120.
- ii. 120 factorial is 479,001,600.
- iii. 0 factorial is 1.

- c. For this problem, I'll introduce the *geometric mean*. The geometric mean is a particular measure of centrality, different from the more common arithmetic mean. Given n observations denoted  $x_1, x_2, \dots, x_n$ , their geometric mean  $\bar{g}$  is computed as follows:

$$\bar{g} = (x_1 \times x_2 \times \dots \times x_n)^{1/n} = \left( \prod_{i=1}^n x_i \right)^{1/n}$$

For example, to find the geometric mean of the data 4.3, 2.1, 2.2, 3.1, calculate the following:

$$\bar{g} = (4.3 \times 2.1 \times 2.2 \times 3.1)^{1/4} = 61.58460.25 = 2.8$$

(This is rounded to 1 d.p.)

Write a function named *geolist* that can search through a specified list and compute the geometric means of each member per the following guidelines:

- Your function must define and use an internal helper function that returns the geometric mean of a vector argument.
- Assume the list can only have numeric vectors or numeric matrices as its members. Your function should contain an appropriate loop to inspect each member in turn.
- If the member is a vector, compute the geometric mean of that vector, overwriting the member with the result, which should be a single number.
- If the member is a matrix, use an implicit loop to compute the geometric mean of *each row* of the matrix, overwriting the member with the results.
- The final list should be returned to the user.

Now, as a quick test, check that your function matches the following two calls:

i.

```
R> foo <- list(1:3,matrix(c(3.3,3.2,2.8,2.1,4.6,4.5,3.1,9.4),4,2),
matrix(c(3.3,3.2,2.8,2.1,4.6,4.5,3.1,9.4),2,4))
```

```
R> geolist(foo)
```

```
[[1]]
```

```
[1] 1.817121
```

```
[[2]]
```

```
[1] 3.896152 3.794733 2.946184 4.442972
```

```
[[3]]
```

```
[1] 3.388035 4.106080
```

ii.

```
R> bar <- list(1:9,matrix(1:9,1,9),matrix(1:9,9,1),matrix(1:9,3,3))
R> geolist(bar)
[[1]]
[1] 4.147166
[[2]]
[1] 4.147166
[[3]]
[1] 1 2 3 4 5 6 7 8 9
[[4]]
[1] 3.036589 4.308869 5.451362
```

### Solution 11.3

```
> #(a)

> foo <- list("a",c("b","c","d","e"),"f",c("g","h","i"))

> lapply(foo,function(x) paste(x,"!",sep=""))

[[1]]

[1] "a!"

[[2]]

[1] "b!" "c!" "d!" "e!"

[[3]]

[1] "f!"

[[4]]

[1] "g!" "h!" "i!"
```

```
> #(b)

> facrec <- function(x) {
  if(x==0) {
    return(1)
  } else {
    return(x*facrec(x-1))
  }
}

> ##(i)
```

```
> facrec(5)
```

```
[1] 120
```

```
> ##(ii)
```

```
> facrec(12)
```

```
[1] 479001600
```

```
> ##(iii)
```

```
> facrec(0)
```

```
[1] 1
```

```
> #(c)
```

```
> geolist <- function(x) {
  geo <- function(nums) {
    return(prod(nums)^{(1/length(nums))})
  }

  for(i in 1:length(x)) {
```

```

if(!is.matrix(x[[i]])){
  x[[i]] <- geo(x[[i]])
} else {
  x[[i]] <- apply(x[[i]], 1, geo)
}
}
return(x)
}
> ##(i)

> foo <-
list(1:3,matrix(c(3.3,3.2,2.8,2.1,4.6,4.5,3.1,9.4),4,2),matrix(c(3.3,3.2,2.8,2.1,4.6,4.5,3.1,9.4),2,4))

> geolist(foo)

[[1]]
[1] 1.817121

[[2]]
[1] 3.896152 3.794733 2.946184 4.442972

[[3]]
[1] 3.388035 4.106080

> ##(ii)

> bar <- list(1:9,matrix(1:9,1,9),matrix(1:9,9,1),matrix(1:9,3,3))

> geolist(bar)

[[1]]
[1] 4.147166

[[2]]
[1] 4.147166

[[3]]
[1] 1 2 3 4 5 6 7 8 9

[[4]]
[1] 3.036589 4.308869 5.451362

```

# Chapter 12: Exceptions, Timings, and Visibility

Estimated time to complete: **30 minutes**

## Exercise 12.1

- a. In Exercise 11.3 (b) on page 238, your task was to write a recursive R function to compute integer factorials, given some supplied non-negative integer  $x$ . Now, modify your function so that it throws an error (with an appropriate message) if  $x$  is negative.

Test your new function responses by using the following:

- i.  $x$  as 5
- ii.  $x$  as 8
- iii.  $x$  as -8

- b. The idea of *matrix inversion*, briefly discussed in Section 3.3.6, is possible only for certain square matrices (those with an equal number of columns as rows). These inversions can be computed using the *solve* function, for example:

```
R> solve(matrix(1:4,2,2))
[,1] [,2]
[1,] -2 1.5
[2,] 1 -0.5
```

Note that *solve* throws an error if the supplied matrix cannot be inverted. With this in mind, write an R function that attempts to invert each matrix in a list, according to the following guidelines:

- The function should take four arguments.
  - \* The list  $x$  whose members are to be tested for matrix inversion
  - \* A value *noninv* to fill in results where a given matrix member of  $x$  cannot be inverted, defaulting to *NA*
  - \* A character string *nonmat* to be the result if a given member of  $x$  is not a matrix, defaulting to "not a matrix"
  - \* A logical value *silent*, defaulting to *TRUE*, to be passed to *try* in the body code
- The function should first check whether  $x$  is in fact a list. If not, it should throw an error with an appropriate message.
- Then, the function should ensure that  $x$  has at least one member. If not, it should throw an error with an appropriate message.
- Next, the function should check whether *nonmat* is a character string. If not, it should try to coerce it to a character string using an appropriate “as-dot” function (see Section 6.2.4), and it should issue an appropriate warning.
- After these checks, a loop should search each member  $i$  of the list  $x$ .
  - \* If member  $i$  is a matrix, attempt to invert it with *try*. If it’s invertible without error, overwrite member  $i$  of  $x$  with the result. If an error is caught, then member  $i$  of  $x$  should be overwritten with the value of *noninv*.
  - \* If member  $i$  is not a matrix, then member  $i$  of  $x$  should be overwritten with the value of *nonmat*.
- Finally, the modified list  $x$  should be returned.

Now, test your function using the following argument values to make sure it responds as expected:

- i.  $x$  as
  - `list(1:4,matrix(1:4,1,4),matrix(1:4,4,1),matrix(1:4,2,2))`
  - and all other arguments at default.

- ii.  $x$  as in (i),  $noninv$  as *Inf*,  $nonmat$  as 666,  $silent$  at default.
- iii. Repeat (ii), but now with  $silent=FALSE$ .
- iv.  $x$  as  
 $list(diag(9),matrix(c(0.2,0.4,0.2,0.1,0.1,0.2),3,3),$   
 $rbind(c(5,5,1,2),c(2,2,1,8),c(6,1,5,5),c(1,0,2,0)),$   
 $matrix(1:6,2,3),cbind(c(3,5),c(6,5)),as.vector(diag(2)))$  and  $noninv$  as "unsuitable matrix";  
all other values at default.

Finally, test the error messages by attempting calls to your function with the following:

- v.  $x$  as "hello"
- vi.  $x$  as *list()*

## Solution 12.1

```
> #(a)

>facrec2 <- function(x) {
  if(x<0) {
    stop("'x' must be a positive integer")
  }

  if(x==0) {
    return(1)
  } else {
    return(x*facrec2(x-1))
  }
}

>##(i)

>facrec2(5)

[1] 120

>##(ii)

>facrec2(8)

[1] 40320

>##(iii)

>facrec2(-8)
```

Error in facrec2(-8) : 'x' must be a positive integer

```
> #(b)

>matinv <- function(x, noninv=NA, nonmat="not a matrix", silent=TRUE) {
  if(!is.list(x)) {
    stop("'x' must be a list")
  }

  n <- length(x)
  if(n==0) {
    stop("'x' appears to be empty")
  }

  if(!is.character(nonmat)) {
    warning("attempting to coerce 'nonmat' to a character string")
    nonmat <- as.character(nonmat)
  }

  for(i in 1:n) {
    if(is.matrix(x[[i]])) {
      attempt <- try(solve(x[[i]]), silent=silent)
      if(class(attempt)=="try-error") {
        x[[i]] <- noninv
      } else {
        x[[i]] <- attempt
      }
    } else {
```

```

        x[[i]] <- nonmat
    }
}

return(x)
}
> ##(i)

> x <- list(1:4,matrix(1:4,1,4),matrix(1:4,4,1),matrix(1:4,2,2))
> matinv(x)

[[1]]
[1] "not a matrix"

[[2]]
[1] NA

[[3]]
[1] NA

[[4]]
[,1] [,2]
[1,] -2 1.5
[2,]  1 -0.5

> ##(ii)

> matinv(x,noninv=Inf,nonmat=666)

[[1]]
[1] "666"

[[2]]
[1] Inf

[[3]]
[1] Inf

```

```
[[4]]  
[,1] [,2]  
[1,] -2 1.5  
[2,] 1 -0.5
```

Warning message:

```
In matinv(x, noninv = Inf, nonmat = 666) :  
  attempting to coerce 'nonmat' to a character string > ##(iii)  
> matinv(x,noninv=Inf,nonmat=666,silent=F)
```

Error in solve.default(x[[i]]) : 'a' (1 x 4) must be square

In addition: Warning message:

```
In matinv(x, noninv = Inf, nonmat = 666, silent = F) :  
  attempting to coerce 'nonmat' to a character string
```

Error in solve.default(x[[i]]) : 'a' (4 x 1) must be square

```
[[1]]  
[1] "666"
```

```
[[2]]  
[1] Inf
```

```
[[3]]  
[1] Inf
```

```
[[4]]  
[,1] [,2]  
[1,] -2 1.5  
[2,] 1 -0.5  
> ##(iv)  
> x <-  
list(diag(9),matrix(c(0.2,0.4,0.2,0.1,0.1,0.2),3,3),rbind(c(5,5,1,2),c(2,2,1,8),c(6,1,5,5),c(1,0,2,0)),  
matrix(1:6,2,3),cbind(c(3,5),c(6,5)),as.vector(diag(2)))  
> matinv(x,noninv="unsuitable matrix")
```

[[1]]

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]

[1,]	1	0	0	0	0	0	0	0	0
[2,]	0	1	0	0	0	0	0	0	0
[3,]	0	0	1	0	0	0	0	0	0
[4,]	0	0	0	1	0	0	0	0	0
[5,]	0	0	0	0	1	0	0	0	0
[6,]	0	0	0	0	0	1	0	0	0
[7,]	0	0	0	0	0	0	1	0	0
[8,]	0	0	0	0	0	0	0	1	0
[9,]	0	0	0	0	0	0	0	0	1

[[2]]

[1] "unsuitable matrix"

[[3]]

[,1] [,2] [,3] [,4]

[1,]	0.019900498	-0.2288557	0.35820896	-0.79104478
[2,]	0.203980100	0.1542289	-0.32835821	0.64179104
[3,]	-0.009950249	0.1144279	-0.17910448	0.89552239
[4,]	-0.054726368	0.1293532	0.01492537	-0.07462687

[[4]]

[1] "unsuitable matrix"

[[5]]

[,1] [,2]

[1,]	-0.3333333	0.4
[2,]	0.3333333	-0.2

[[6]]

```
[1] "not a matrix"
```

```
> ##(v)
```

```
> x <- "hello"
```

```
> matinv(x)
```

```
Error in matinv(x) : 'x' must be a list> ##(vi)
```

```
> x <- list()
```

```
> matinv(x)
```

```
Error in matinv(x) : 'x' appears to be empty>
```

## Exercise 12.2

- a. Modify *prog\_test* from Section 12.2.1 to include an ellipsis in its argument list, intended to take values for the additional arguments in *txtProgressBar*; name the new function *prog\_test\_fancy*.

Time how long it takes a call to *prog\_test\_fancy* to execute. Set 50 as *n*, instruct the progress bar (through the ellipsis) to use *style*=3, and set the bar character to be "r".

- b. In Section 12.1.2, you defined a function named *myfibvectorTRY* (which itself calls *myfibrec2* from Section 12.1.1) to return multiple terms from the Fibonacci sequence based on a supplied “term vector” *nvec*. Write a new version of *myfibvectorTRY* that includes a progress bar of *style*=3 and a character of your choosing that increments at each pass of the internal *for* loop. Then, do the following:
- Use your new function to reproduce the results from the text where *nvec*=*c*(3,2,7,0,9,13).
  - Time how long it takes to use your new function to return the first 35 terms of the Fibonacci sequence. What do you notice, and what does this say about your recursive Fibonacci functions?
- c. Remain with the Fibonacci sequence. Write a stand-alone *for* loop that can compute, and store in a vector, the same first 35 terms as in (b)(ii). Time it. Which approach would you prefer?

## Solution 12.2

```
> #(a)
```

```
> prog_test_fancy <- function(n,...) {
  result <- 0
  progbar <- txtProgressBar(min=0,max=n,...)
  for(i in 1:n){
    result <- result + 1
    Sys.sleep(0.5)
    setTxtProgressBar(progbar,value=i)
  }
  close(progbar)
  return(result)
}
ence <- Sys.time()
prog_test_fancy(50,style=3,char="r")
```

```
[1] 50
```

```
> differ <- Sys.time()
```

```
> differ-ence
```

```
Time difference of 25.92701 secs
```

```
> #(b)
```

```
> myfibrec2 <- function(n) {
  if(n<0) {
    warning("Assuming you meant 'n' to be positive -- doing that
instead")
    n <- n*-1
  } else if(n==0) {
    stop("'n' is uninterpretable at 0")
  }

  if(n==1 || n==2) {
```

```

        return(1)
    } else {
        return(myfibrec2(n-1)+myfibrec2(n-2))
    }
}
myfibvectorTRY2 <- function(nvec) {
    nterms <- length(nvec)
    result <- rep(0,nterms)
    progbar <- txtProgressBar(min=0,max=nterms,style=3,char="-")
    for(i in 1:nterms){
        attempt <- try(mfibrec2(nvec[i]),silent=T)
        if(class(attempt)=="try-error"){
            result[i] <- NA
        } else {
            result[i] <- attempt
        }
        setTxtProgressBar(progbar,value=i)
    }
    close(progbar)
    return(result)
}

```

```

> ##(i)
> myfibvectorTRY2(nvec=c(3,2,7,0,9,13))

```

```
[1] 2 1 13 NA 34 233
```

```
> ##(ii)
```

```

> t1 <- Sys.time()
> myfibvectorTRY2(1:35)

```

```

[1]   1   1   2   3   5   8   13  21  34  55  89 144 233 377 610
987 1597 2584 4181 6765 10946 17711 28657 46368

```

```

[25] 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887
9227465

```

```
> t2 <- Sys.time()
```

```
> t2-t1
```

Time difference of 37.24528 secs

```
> ### This takes almost 1 minute on my machine... execution slows down as the recursion gets
deeper... recursion perhaps not so good for computing Fibonacci sequence
```

```

> #(c)
t1 <- Sys.time()
fibvec <- c(1,1,rep(NA,33))
for(i in 3:35){
    fibvec[i] <- fibvec[i-2]+fibvec[i-1]
}
fibvec

```

```
[1]   1   1   2   3   5   8   13  21  34  55  89 144 233 377 610  
987 1597 2584 4181 6765 10946 17711 28657 46368  
[25] 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887  
9227465
```

```
> t2 <- Sys.time()
```

```
> t2-t1
```

Time difference of 0.07639289 secs

```
> ### This is substantially quicker than recursion!
```

## Part 3

# Chapter 13: Elementary Statistics

Estimated time to complete: **60 minutes**

### Exercise 13.1

- a. For each of the following, identify the type of variable described: numeric-continuous, numeric-discrete, categorical-nominal, or categorical-ordinal:
  - i. The number of blemishes on the hood of a car coming off a production line
  - ii. A survey question that asks the participant to select from Strongly agree, Agree, Neutral, Disagree, and Strongly disagree
  - iii. The noise level (in decibels) at a concert
  - iv. The noise level out of three possible choices: high, medium, low
  - v. A choice of primary color
  - vi. The distance between a cat and a mouse
- b. For each of the following, identify whether the quantity discussed is a population parameter or a sample statistic. If the latter, also identify what the corresponding population parameter is.
  - i. The percentage of 50 New Zealanders who own a gaming console
  - ii. The average number of blemishes found on the hoods of three cars in the No Dodgy Carz yard
  - iii. The proportion of domestic cats in the United States that wear a collar
  - iv. The average number of times per day a vending machine is used in a year
  - v. The average number of times per day a vending machine is used in a year, based on data collected on three distinct days in that year

### Solution 13.1

```
> #(a)  
  
> ##(i) Numeric-discrete  
> ##(ii) Categorical-ordinal  
> ##(iii) Numeric-continuous  
> ##(iv) Categorical-ordinal  
> ##(v) Categorical-nominal  
> ##(vi) Numeric-continuous  
  
> #(b)  
  
> ##(i) Sample statistic. Population parameter is the proportion of NZers who own a gaming console.  
  
> ##(ii) Sample statistic. Population parameter is the average number of blemishes on the hoods of all cars at No Dodgy Carz.  
  
> ##(iii) Population parameter.
```

> ##(iv) Population parameter.

> ##(v) Sample statistic. Population parameter is (iv).

## Exercise 13.2

- a. Obtain, rounded to two decimal places, the proportion of seismic events in the *quakes* data frame that occurred at a depth of 300 km or deeper.
- b. Remaining with the *quakes* data set, calculate the mean and median magnitudes of the events that occurred at a depth of 300 km or deeper.
- c. Using the *chickwts* data set, write a *for* loop that gives you the mean weight of chicks for each feed type—the same as the results given by the *tapply* function in Section 13.2.1. Display the results rounded to one decimal place and, when printing, ensure each mean is labeled with the appropriate feed type.

Another ready-to-use data set (in the automatically loaded *datasets* package) is *InsectSprays*. It contains data on the number of insects found on various agricultural units, as well as the type of insect spray that was used on each unit. Ensure you can access the data frame at the prompt; then study the help file *?InsectSprays* to get an idea of R’s representation of the two variables.

- d. Identify the two variable types in *InsectSprays* (as per the definitions in Section 13.1.1 and Section 13.1.2).
- e. Calculate the modes of the distribution of insect counts, regardless of spray type.
- f. Use *tapply* to report the total insect counts by each spray type.
- g. Using the same kind of for loop as in (c), compute the percentage of agricultural units in each spray type group that had at least five bugs on them. When printing to the screen, round the percentages to the nearest whole number.
- h. Obtain the same numeric results as in (g), with rounding, but use *tapply* and a disposable function.

## Solution 13.2

```
> #(a)  
  
> round(mean(quakes$depth>=300),2)  
[1] 0.45  
  
> #(b)  
  
> mean(quakes$mag[quakes$depth>=300])  
[1] 4.527373  
  
> median(quakes$mag[quakes$depth>=300])  
[1] 4.5  
  
> #(c)  
  
> for(i in levels(chickwts$feed)){  
+   cat(i,": ",round(mean(chickwts$weight[chickwts$feed==i]),1)," grams\n",sep="")  
+ }  
  
casein: 323.6 grams  
  
horsebean: 160.2 grams  
  
linseed: 218.8 grams  
  
meatmeal: 276.9 grams
```

```

soybean: 246.4 grams
sunflower: 328.9 grams
> #(d) count: numeric-discrete; spray: categorical-nominal
> #(e)
> Ctab <- table(InsectSprays$count)
> Ctab[Ctab==max(Ctab)]
3
8
> #(f)
> tapply(InsectSprays$count, INDEX=InsectSprays$spray, FUN=sum)
A  B  C  D  E  F
174 184 25 59 42 200
> #(g)
> for(i in levels(InsectSprays$spray)){
+   cat("Spray ",i,"; at least 5 bugs:  

",round(mean(InsectSprays$count[InsectSprays$spray==i]>=5)*100,0),"% \n",sep="")
+ }
Spray A; at least 5 bugs: 100%
Spray B; at least 5 bugs: 100%
Spray C; at least 5 bugs: 8%
Spray D; at least 5 bugs: 58%
Spray E; at least 5 bugs: 33%
Spray F; at least 5 bugs: 100%
> #(h)
> tapply(InsectSprays$count, INDEX=InsectSprays$spray, FUN=function(x)
round(mean(x>=5)*100,0))
A  B  C  D  E  F
100 100 8 58 33 100
>

```

## Exercise 13.3

- a. Using the *chickwts* data frame, compute the 10th, 30th, and 90th percentiles of all the chick weights and then use *tapply* to determine which feed type is associated with the highest sample variance of weights.
- b. Turn to the seismic event data in *quakes* and complete the following tasks:
  - i. Find the IQR of the recorded depths.
  - ii. Find the five-number summary of all magnitudes of seismic events that occur at a depth of 400 km or deeper. Compare this to the summary values found in Section 13.2.3 of those events occurring at less than 400 km and briefly comment on what you notice.
  - iii. Use your knowledge of *cut* (Section 4.3.3) to create a new factor vector called *depthcat* that identifies four evenly spaced categories of *quakes\$depth* so that when you use *levels(depthcat)*, it gives the following:  
*R> levels(depthcat)*  
[1] "[40,200]" "[200,360]" "[360,520]" "[520,680]"
  - iv. Find the sample mean and standard deviation of the magnitudes of the events associated with each category of depth according to *depthcat*.
  - v. Use *tapply* to compute the 0.8th quantile of the magnitudes of the seismic events in *quakes*, split by *depthcat*.

## Solution 13.3

```
> #(a)  
  
> quantile(chickwts$weight,c(0.1,0.3,0.9))  
  
10% 30% 90%  
  
153 217 359  
  
> chickvars <- tapply(chickwts$weight,INDEX=chickwts$feed,FUN=var)  
  
> chickvars[chickvars==max(chickvars)]  
  
meatmeal  
  
4212.091  
  
> #(b)  
  
> ##(i)  
  
> IQR(quakes$depth)  
  
[1] 444  
  
> ##(ii)  
  
> summary(quakes$mag[quakes$depth>=400]) ### Magnitudes at <400 have a similar amount of  
spread as those at >=400 (and an equivalent IQR of 0.5), but are centered at slightly higher values.  
  
Min. 1st Qu. Median Mean 3rd Qu. Max.  
  
126
```

```

4.000 4.200 4.500 4.545 4.700 5.900

> ##(iii)

> dmin <- min(quakes$depth)
> dmax <- max(quakes$depth)
> depthcat <-
cut(quakes$depth,breaks=seq(dmin,dmax,length=5),include.lowest=TRUE,right=FALSE)
> levels(depthcat)
[1] "[40,200)" "[200,360)" "[360,520)" "[520,680)"

> ##(iv)

> tapply(quakes$mag,INDEX=depthcat,FUN=mean)
[40,200) [200,360) [360,520) [520,680]
4.735492 4.552201 4.500000 4.546488

> tapply(quakes$mag,INDEX=depthcat,FUN=sd)
[40,200) [200,360) [360,520) [520,680]
0.4042286 0.3788451 0.3612255 0.3908510

> ##(v)

> tapply(quakes$mag,INDEX=depthcat,FUN=quantile,prob=0.8)
[40,200) [200,360) [360,520) [520,680]
5.1      4.8      4.8      4.9

```

## Exercise 13.4

- a. In Exercise 7.1 (b) on page 139, you plotted height against weight measurements. Compute the correlation coefficient based on the observed data of these two variables.
- b. Another of R's built-in, ready-to-use data sets is *mtcars*, containing a number of descriptive details on performance aspects of 32 automobiles.
  - i. Ensure you can access this data frame by entering *mtcars* at the prompt. Then inspect its help file to get an idea of the types of data present.
  - ii. Two of the variables describe a vehicle's horsepower and shortest time taken to travel a quarter-mile distance. Using base R graphics, plot these two data vectors with horsepower on the x-axis and compute the correlation coefficient.
  - iii. Identify the variable in *mtcars* that corresponds to transmission type. Use your knowledge of factors in R to create a new factor from this variable called *tranfac*, where manual cars should be labeled "*manual*" and automatic cars "*auto*".
  - iv. Now, use *qplot* from *ggplot2* in conjunction with *tranfac* to produce the same scatterplot as in (ii) so that you're able to visually differentiate between manual and automatic cars.
  - v. Finally, compute separate correlation coefficients for horsepower and quarter-mile time based on the transmission of the vehicles and, comparing these estimates with the overall value from (ii), briefly comment on what you note.
- c. Return to *chickwts* to complete the following tasks:
  - i. Produce a plot like the left panel of Figure 13-7, based on the weights of chicks on the sunflower diet only. Note that one of the sunflower-fed chicks has a far lower weight than the others.
  - ii. Compute the standard deviation and IQR of the weights of the sunflower-fed chicks.
  - iii. Now, suppose you're told that the lowest weight of the sunflower-fed chicks was caused by a certain illness, irrelevant to your research. Delete this observation and recalculate the standard deviation and IQR of the remaining sunflower chicks. Briefly comment on the difference in calculated values.

## Solution 13.4

```
> #(a)
```

```
> w <- c(55,85,75,42,93,63,58,75,89,67)
```

```
> h <- c(161,185,174,154,188,178,170,167,181,178)
```

```
> cor(w,h)
```

```
[1] 0.8621007
```

```
> #(b)
```

```
> ##(i)
```

```
> mtcars
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1

```
Dodge Challenger 15.5 8 318.0 150 2.76 3.520 16.87 0 0 3 2
AMC Javelin     15.2 8 304.0 150 3.15 3.435 17.30 0 0 3 2
Camaro Z28      13.3 8 350.0 245 3.73 3.840 15.41 0 0 3 4
Pontiac Firebird 19.2 8 400.0 175 3.08 3.845 17.05 0 0 3 2
Fiat X1-9       27.3 4 79.0 66 4.08 1.935 18.90 1 1 4 1
Porsche 914-2    26.0 4 120.3 91 4.43 2.140 16.70 0 1 5 2
Lotus Europa     30.4 4 95.1 113 3.77 1.513 16.90 1 1 5 2
Ford Pantera L   15.8 8 351.0 264 4.22 3.170 14.50 0 1 5 4
Ferrari Dino     19.7 6 145.0 175 3.62 2.770 15.50 0 1 5 6
Maserati Bora    15.0 8 301.0 335 3.54 3.570 14.60 0 1 5 8
Volvo 142E       21.4 4 121.0 109 4.11 2.780 18.60 1 1 4 2
```

> ?mtcars

starting httpd help server ... done

The screenshot shows an R documentation browser window with the following details:

- Address bar: 127.0.0.1:15361/library/datasets/ht
- Title bar: R Motor Trend Car Road Tests
- Page title: mtcars {datasets}
- Page header: R Documentation
- Section: Motor Trend Car Road Tests
- Section: Description

The text content under "Description" is:

The data was extracted from the 1974 *Motor Trend* US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models).

- Section: Usage

The text content under "Usage" is:

```
mtcars
```

- Section: Format

The text content under "Format" is:

A data frame with 32 observations on 11 variables.

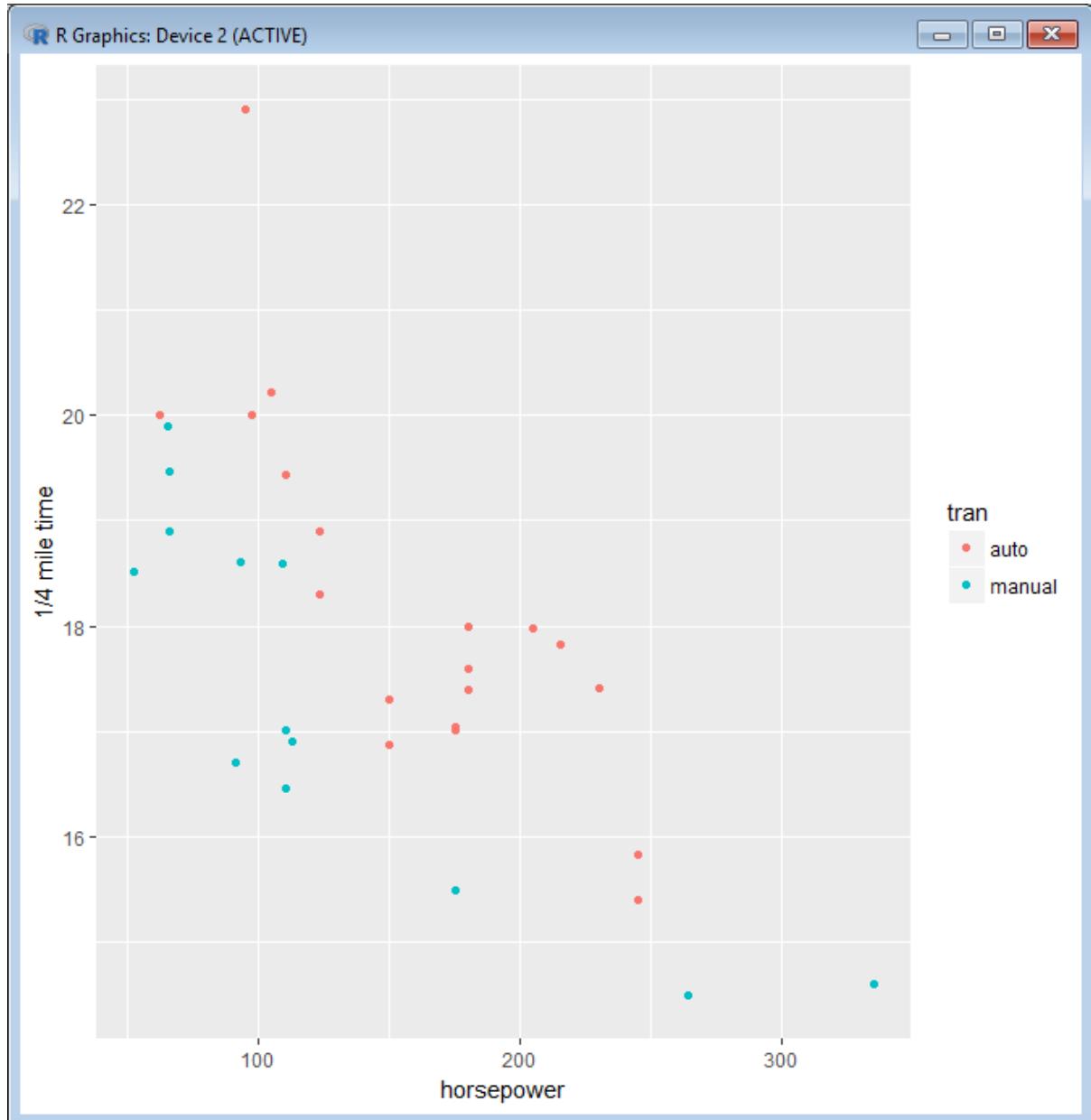
Variable descriptions:

  - [, 1] mpg Miles/(US) gallon
  - [, 2] cyl Number of cylinders
  - [, 3] disp Displacement (cu.in.)
  - [, 4] hp Gross horsepower
  - [, 5] drat Rear axle ratio
  - [, 6] wt Weight (1000 lbs)
  - [, 7] qsec 1/4 mile time
  - [, 8] vs V/S
  - [, 9] am Transmission (0 = automatic, 1 = manual)
  - [,10] gear Number of forward gears
  - [,11] carb Number of carburetors

```
> ##(ii)
> plot(mtcars$hp,mtcars$qsec,xlab="horsepower",ylab="1/4 mile time")
> cor(mtcars$hp,mtcars$qsec)
[1] -0.7082234
> ##(iii)
> tran <- factor(mtcars$am,labels=c("auto","manual"))
> ##(iv)
```

```
> library(ggplot2)
```

```
> qplot(mtcars$hp,mtcars$qsec,col=tran,xlab="horsepower",ylab="1/4 mile time")
```



```
> ##(v)
```

```
> cor(mtcars$hp[tran=="manual"],mtcars$qsec[tran=="manual"])
```

```
[1] -0.8494566
```

```
> cor(mtcars$hp[tran=="auto"],mtcars$qsec[tran=="auto"]) ### Separated by transmission type,  
correlations are more extreme than when pooled. Manual cars have a slightly more extreme  
negative correlation between the two variables than automatic cars.
```

```
[1] -0.8040275
```

```
> #(c)
```

```
> ##(i)
```

```
> sunchicks <- chickwts$weight[chickwts$feed=="sunflower"]  
> plot(sunchicks,rep(0,length(sunchicks)),yaxt="n",bty="n",xlab="sunflower chick  
weights",ylab="")  
> abline(h=0,col="gray",lty=2)  
> ##(ii)  
> sd(sunchicks)  
[1] 48.83638  
> IQR(sunchicks)  
[1] 27.5  
> ##(iii)  
> sd(sunchicks[-which(sunchicks==min(sunchicks))])  
[1] 38.31473  
> IQR(sunchicks[-which(sunchicks==min(sunchicks))]) ### Both measures of spread decrease  
with the deletion of the lowest weight; though the sd is affected more than the IQR...  
[1] 21.5
```

# Chapter 14: Basic Data Visualization

Estimated time to complete: **30 minutes**

## Exercise 14.1

Recall the built-in *InsectSprays* data frame, containing counts of insects on various agricultural units treated with one of six sprays.

- a. Produce a histogram of the counts of insects using base R graphics.
- b. Obtain the total number of insects found according to each spray (this was also asked in Exercise 13.2 (f) on page 273). Then, use base R graphics to produce a vertical barplot and a pie chart of these totals, labeling each plot appropriately.
- c. Use *ggplot2* functionality to generate side-by-side boxplots of the counts of insects according to each spray type and include appropriate axis labels and a title.

Yet another of R's useful ready-to-use data sets is *USArrests*, containing data on the number of arrests for murder, rape, and assault per 100,000 individuals in each of the 50 states of the United States, recorded in 1973 (see, for example, McNeil, 1977). It also includes a variable giving the percentage of urban-based population in each state. Briefly inspect the data frame object and the accompanying documentation *?USArrests*. Then complete the following:

- d. Use *ggplot2* functionality to generate a right-exclusive histogram of the proportion of urban population for the states. Set your breaks to be 10 units each, between 0 and 100. Have the histogram show the first quartile, the median, and the third quartile; then provide a matching legend. Use colors as you like and include appropriate axis annotation.
- e. The code `t(as.matrix(USArrests[,-3]))` creates a matrix of the *USArrests* data without the urban population column, and the built-in R object *state.abb* provides the two-letter state abbreviations, in alphabetical order, as a character vector. Use these two structures and base R graphics to produce a horizontal, stacked barplot with the horizontal bars labeled with state abbreviations and with each bar split according to the type of crime (murder, rape, and assault). Include a legend.
- f. Define a new factor vector *urbancat* that is set to *1* if the corresponding state has an urban population percentage greater than the median percentage and is set to *0* otherwise.
- g. Create a new copy of *USArrests* in your workspace, after deleting the *UrbanPop* column, leaving just the three crime rate variables.

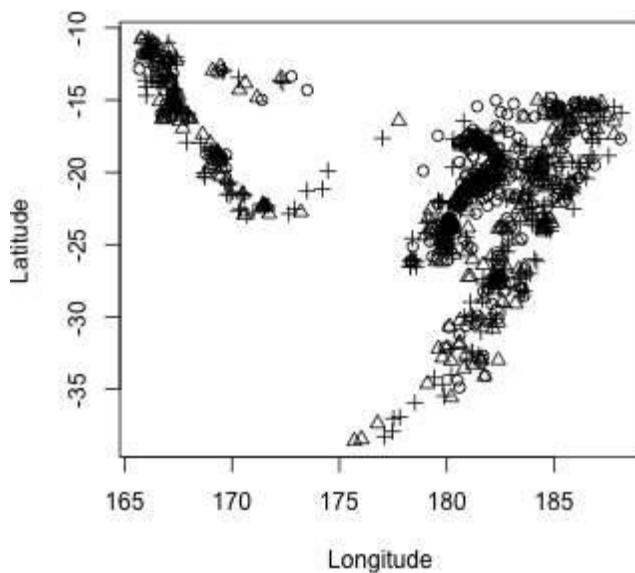
Then insert a new, fourth column in this object with *urbancat*.

- h. Use the data frame from (g) to produce a scatterplot matrix and other associated plots of the three crime rates against one another via *GGally* functionality. Use color to split the crime rates according to the two levels of *urbancat*.

Return to the built-in *quakes* data set.

- i. Create a factor vector corresponding to the magnitudes. Each entry should assume one of three categories based on breaks marked by the minimum magnitude, the  $\frac{1}{3}$ th quantile, the  $\frac{2}{3}$ th quantile, and the maximum magnitude.
- j. Re-create the plot shown next, where low-, medium-, and highmagnitude

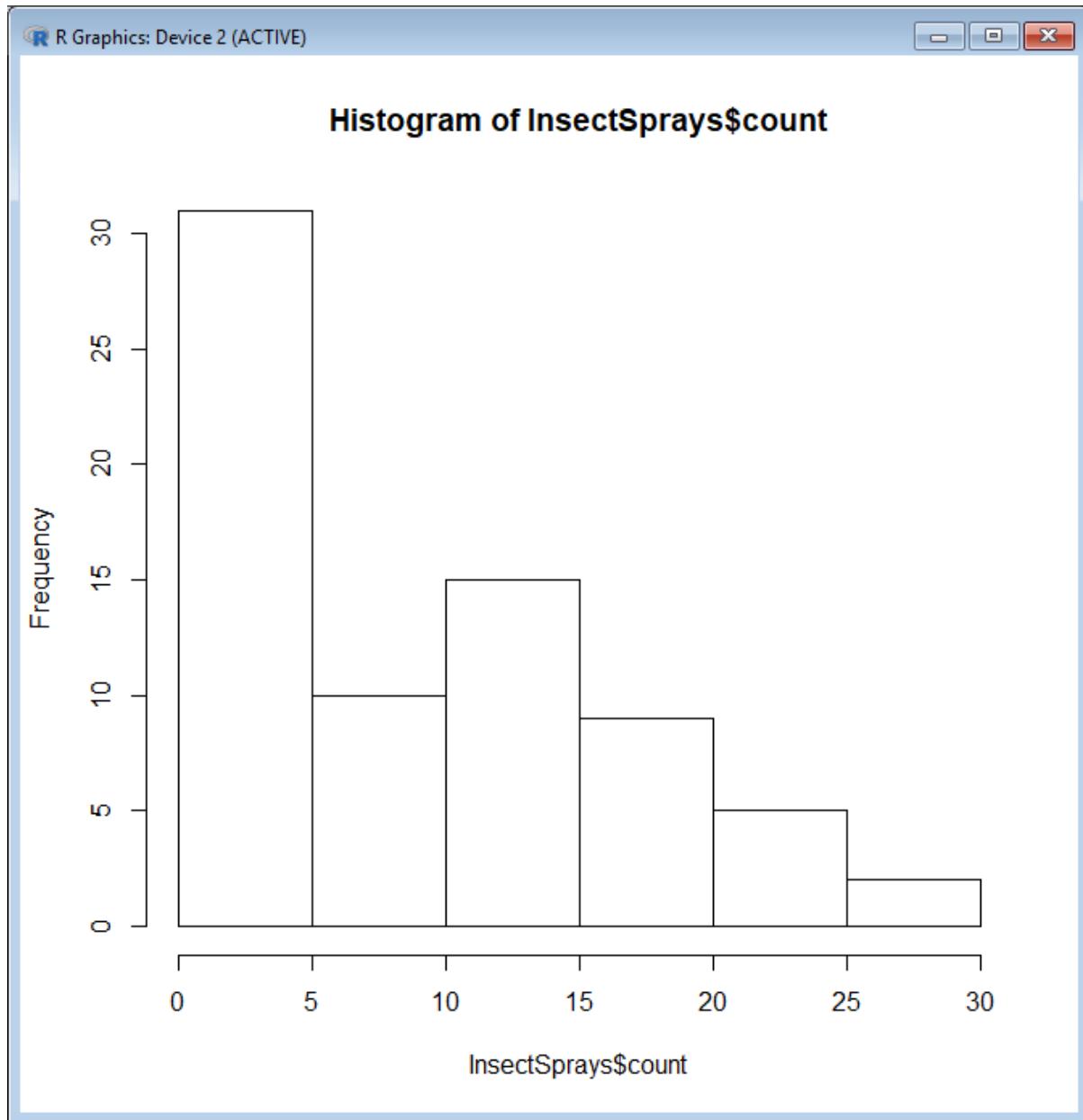
events, according to your factor vector from (i), are plotted with  $pch$  being assigned 1, 2, and 3, respectively.



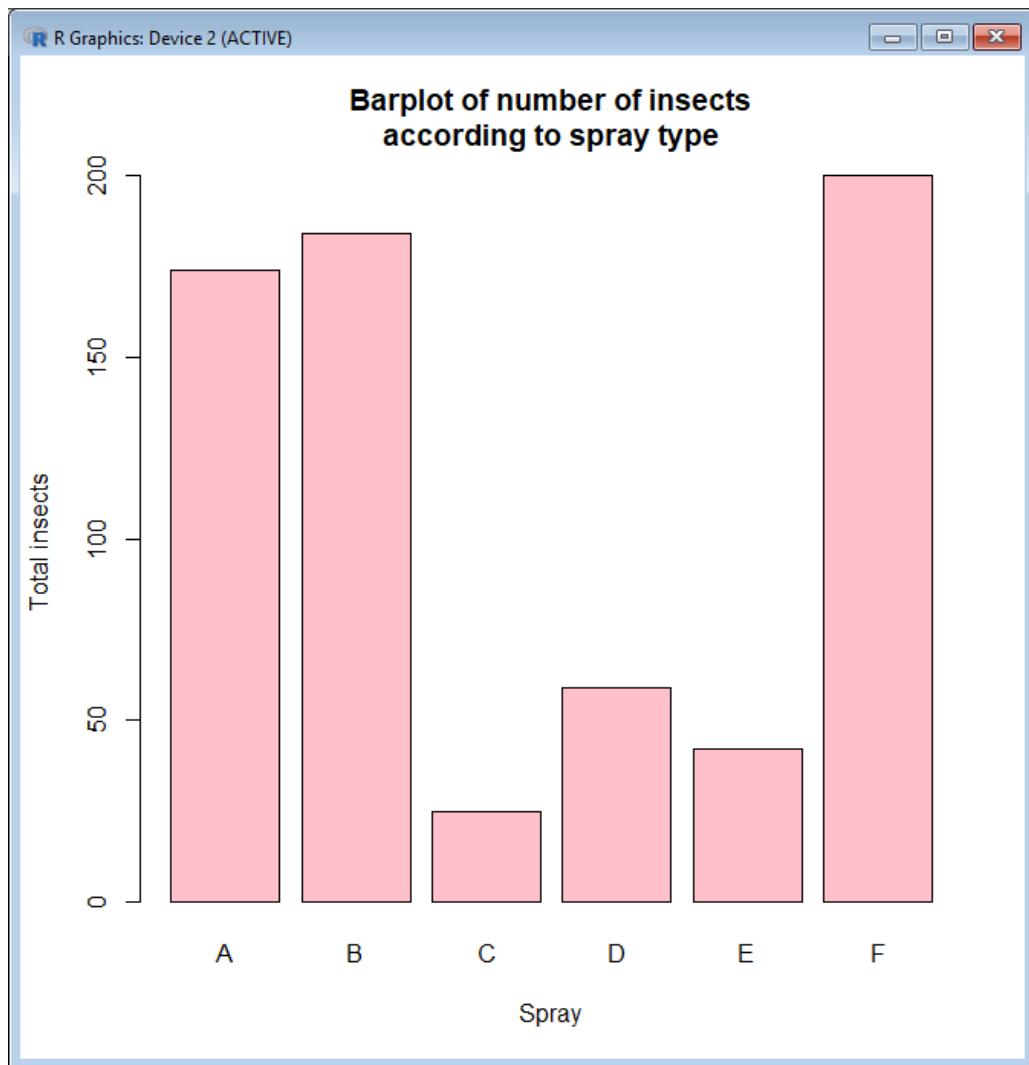
k. Add a legend to the plot from (j) to reference the three  $pch$  values.

## Solution 14.1

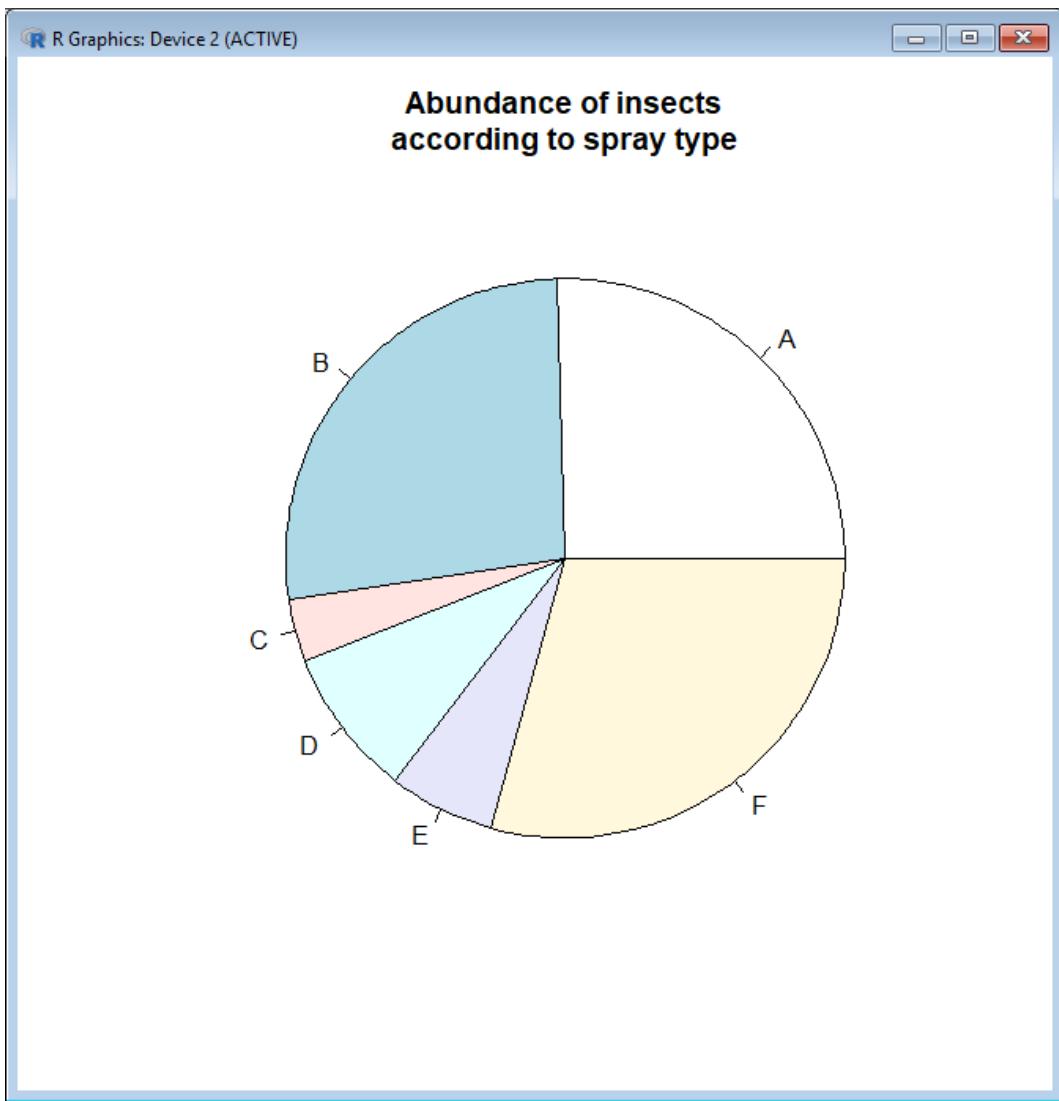
```
> library("ggplot2")
> library("GGally")
> #(a)
> hist(InsectSprays$count)
```



```
> #(b)
> inspray <- levels(InsectSprays$spray)
> incount <- tapply(InsectSprays$count, INDEX=InsectSprays$spray, FUN=sum)
> barplot(incount, names.arg=inspray, xlab="Spray", ylab="Total insects", main="Barplot of number of insects\naccording to spray type", col="pink")
```

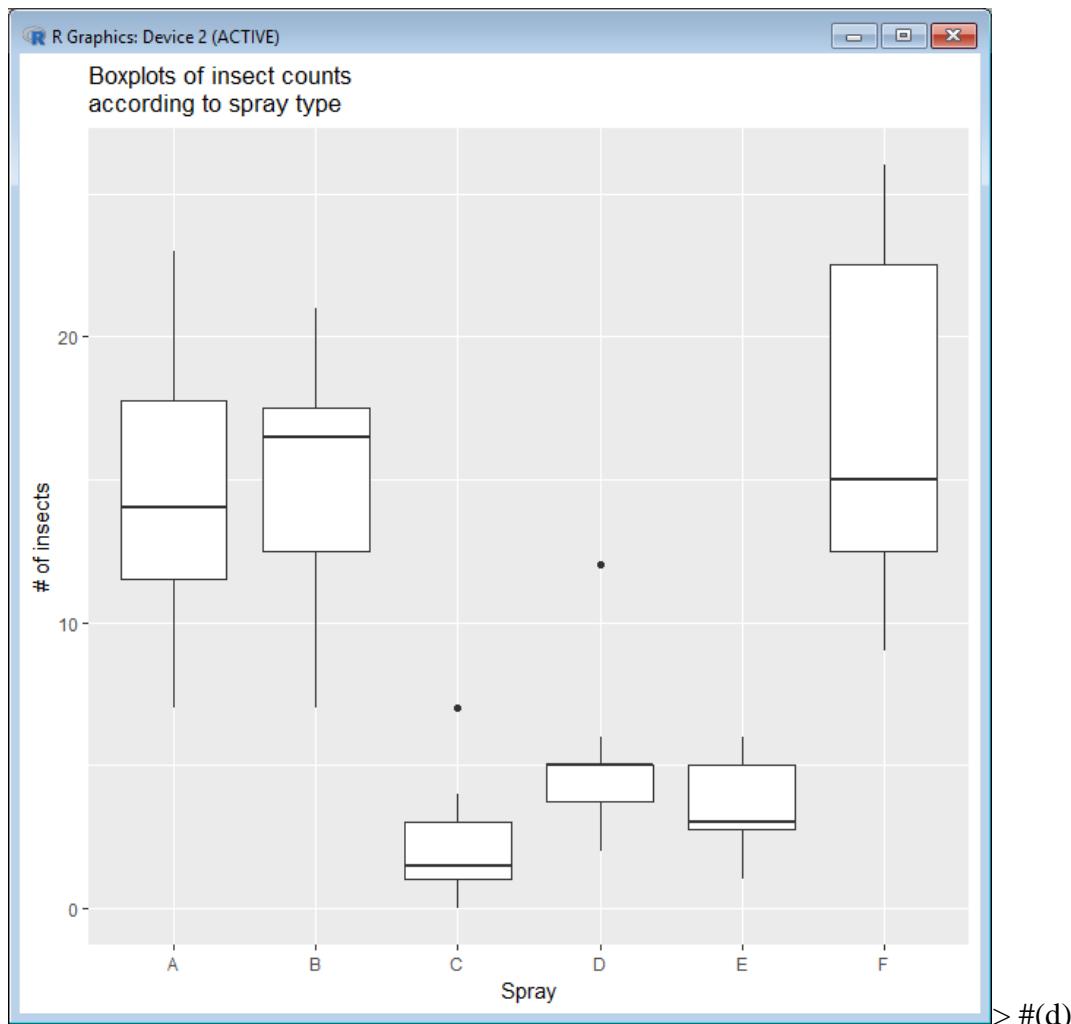


```
> pie(incount,labels=inspray,main="Abundance of insects\naccording to spray type")
```



```
> #(c)
```

```
> qplot(InsectSprays$spray,InsectSprays$count,geom="boxplot",xlab="Spray",ylab="# of insects",main="Boxplots of insect counts\naccording to spray type")
```



> USArests

	Murder	Assault	UrbanPop	Rape
Alabama	13.2	236	58	21.2
Alaska	10.0	263	48	44.5
Arizona	8.1	294	80	31.0
Arkansas	8.8	190	50	19.5
California	9.0	276	91	40.6
Colorado	7.9	204	78	38.7
Connecticut	3.3	110	77	11.1
Delaware	5.9	238	72	15.8
Florida	15.4	335	80	31.9
Georgia	17.4	211	60	25.8
Hawaii	5.3	46	83	20.2
Idaho	2.6	120	54	14.2
Illinois	10.4	249	83	24.0
Indiana	7.2	113	65	21.0
Iowa	2.2	56	57	11.3
Kansas	6.0	115	66	18.0
Kentucky	9.7	109	52	16.3
Louisiana	15.4	249	66	22.2
Maine	2.1	83	51	7.8
Maryland	11.3	300	67	27.8

Massachusetts	4.4	149	85	16.3
Michigan	12.1	255	74	35.1
Minnesota	2.7	72	66	14.9
Mississippi	16.1	259	44	17.1
Missouri	9.0	178	70	28.2
Montana	6.0	109	53	16.4
Nebraska	4.3	102	62	16.5
Nevada	12.2	252	81	46.0
New Hampshire	2.1	57	56	9.5
New Jersey	7.4	159	89	18.8
New Mexico	11.4	285	70	32.1
New York	11.1	254	86	26.1
North Carolina	13.0	337	45	16.1
North Dakota	0.8	45	44	7.3
Ohio	7.3	120	75	21.4
Oklahoma	6.6	151	68	20.0
Oregon	4.9	159	67	29.3
Pennsylvania	6.3	106	72	14.9
Rhode Island	3.4	174	87	8.3
South Carolina	14.4	279	48	22.5
South Dakota	3.8	86	45	12.8
Tennessee	13.2	188	59	26.9
Texas	12.7	201	80	25.5
Utah	3.2	120	80	22.9
Vermont	2.2	48	32	11.2
Virginia	8.5	156	63	20.7
Washington	4.0	145	73	26.2
West Virginia	5.7	81	39	9.3
Wisconsin	2.6	53	66	10.8
Wyoming	6.8	161	60	15.6

> ?USAArrests

## Violent Crime Rates by US State

### Description

This data set contains statistics, in arrests per 100,000 residents for assault, murder, and rape in each of the 50 US states in 1973. Also given is the percent of the population living in urban areas.

### Usage

```
USArests
```

### Format

A data frame with 50 observations on 4 variables.

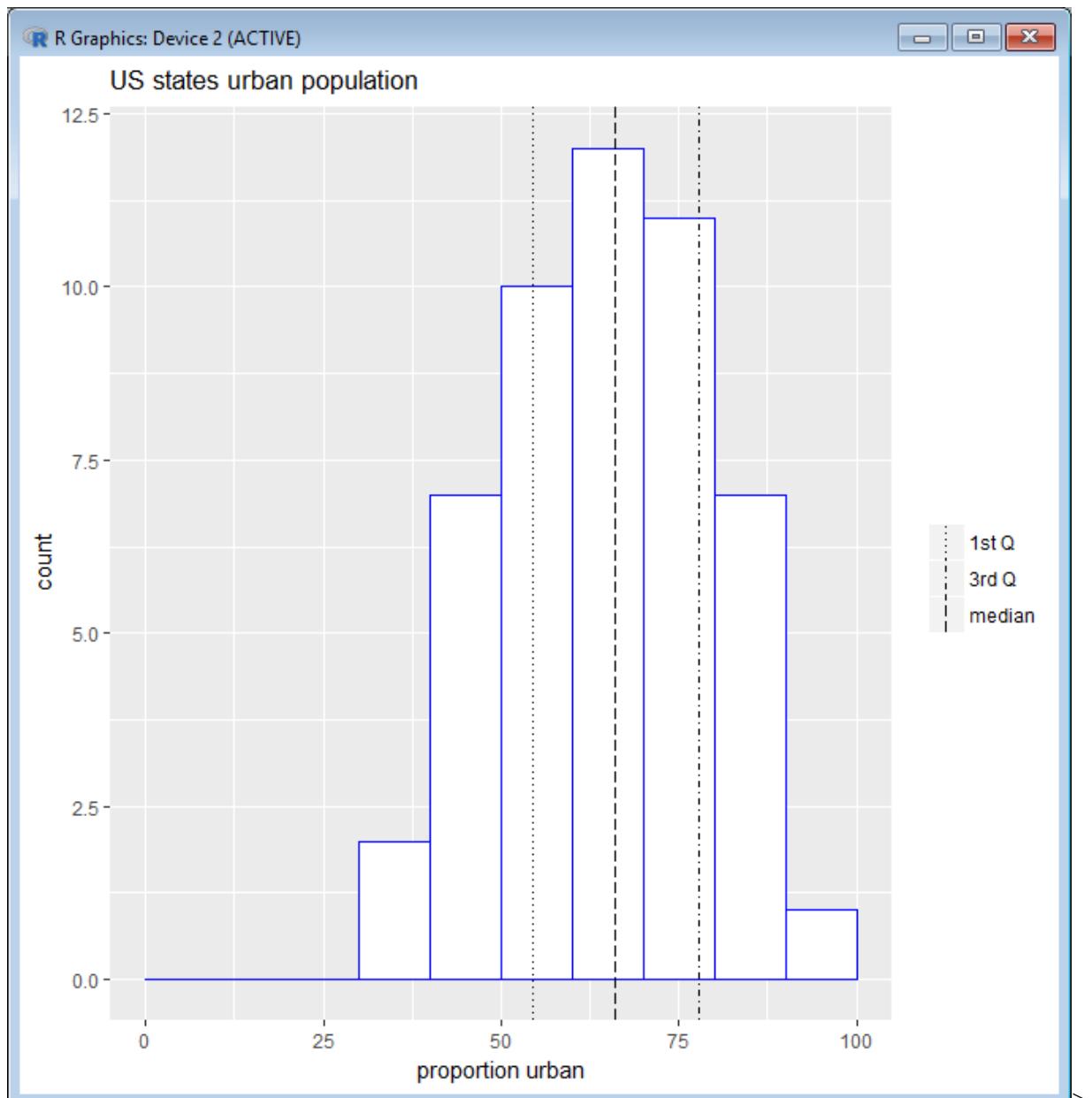
```
[1] Murder    numeric Murder arrests (per 100,000)
[2] Assault   numeric Assault arrests (per 100,000)
[3] UrbanPop  numeric Percent urban population
[4] Rape      numeric Rape arrests (per 100,000)
```

### Note

`USArests` contains the data as in McNeil's monograph. For the `UrbanPop` percentages, a review of the table (No. 21) in the Statistical Abstracts 1975 reveals a transcription error for Maryland (and that McNeil used the same “round to even” rule that R's `round()` uses), as found by Daniel S Coven (Arizona).

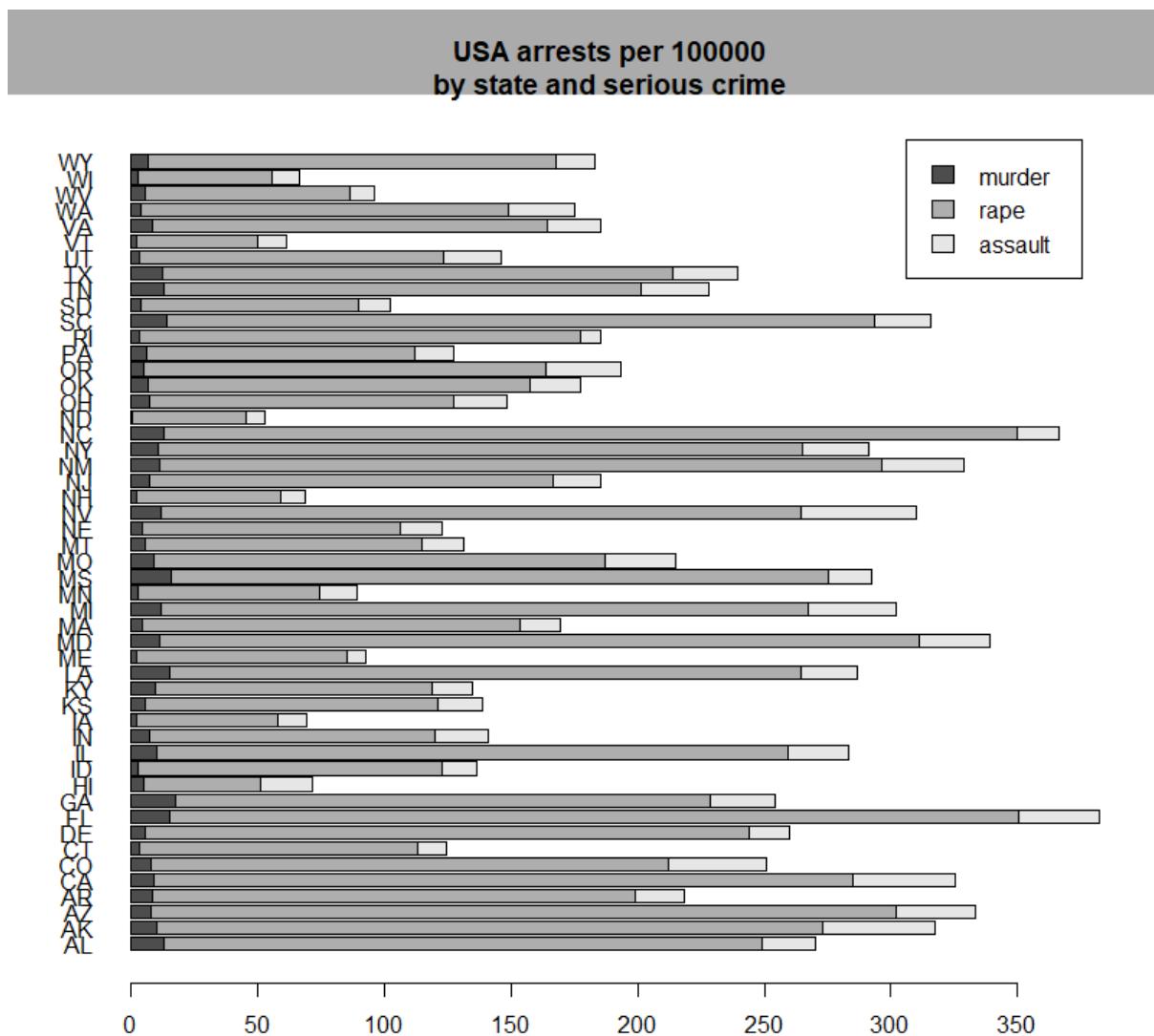
See the example below on how to correct the error and improve accuracy for the ‘<n>.5’ percentages.

```
> qplot(USArests[,3],geom="blank",main="US states urban population",xlab="proportion urban") + geom_histogram(color="blue",fill="white",breaks=seq(0,100,10),closed="right") + geom_vline(mapping=aes(xintercept=c(quantile(USArests[,3],0.25),median(USArests[,3]),quantile(USArests[,3],0.75)),linetype=factor(c("1st Q","median","3rd Q"))),show.legend=TRUE) + scale_linetype_manual(values=3:5) + labs(linetype="")
```



#(e)

```
> ustab <- t(as.matrix(USArrests[,-3]))  
>  
barplot(ustab,names.arg=state.abb,horiz=T,las=1,legend.text=c("murder","rape","assault"),main="USA arrests per 100000\nby state and serious crime")
```



```
> #(f)
```

```
> urbancat <- rep(1,50)
```

```
> urbancat[USArrests$UrbanPop<=median(USArrests$UrbanPop)] <- 0
```

```
> urbancat <- factor(urbancat)
```

```
> #(g)
```

```
> myUSArrests <- USArrests[,-3]
```

```
> myUSArrests$urbancat <- urbancat
```

```
> #(h)
```

```
> ggpairs(myUSArrests,aes(col=urbancat),axisLabels="internal")
```



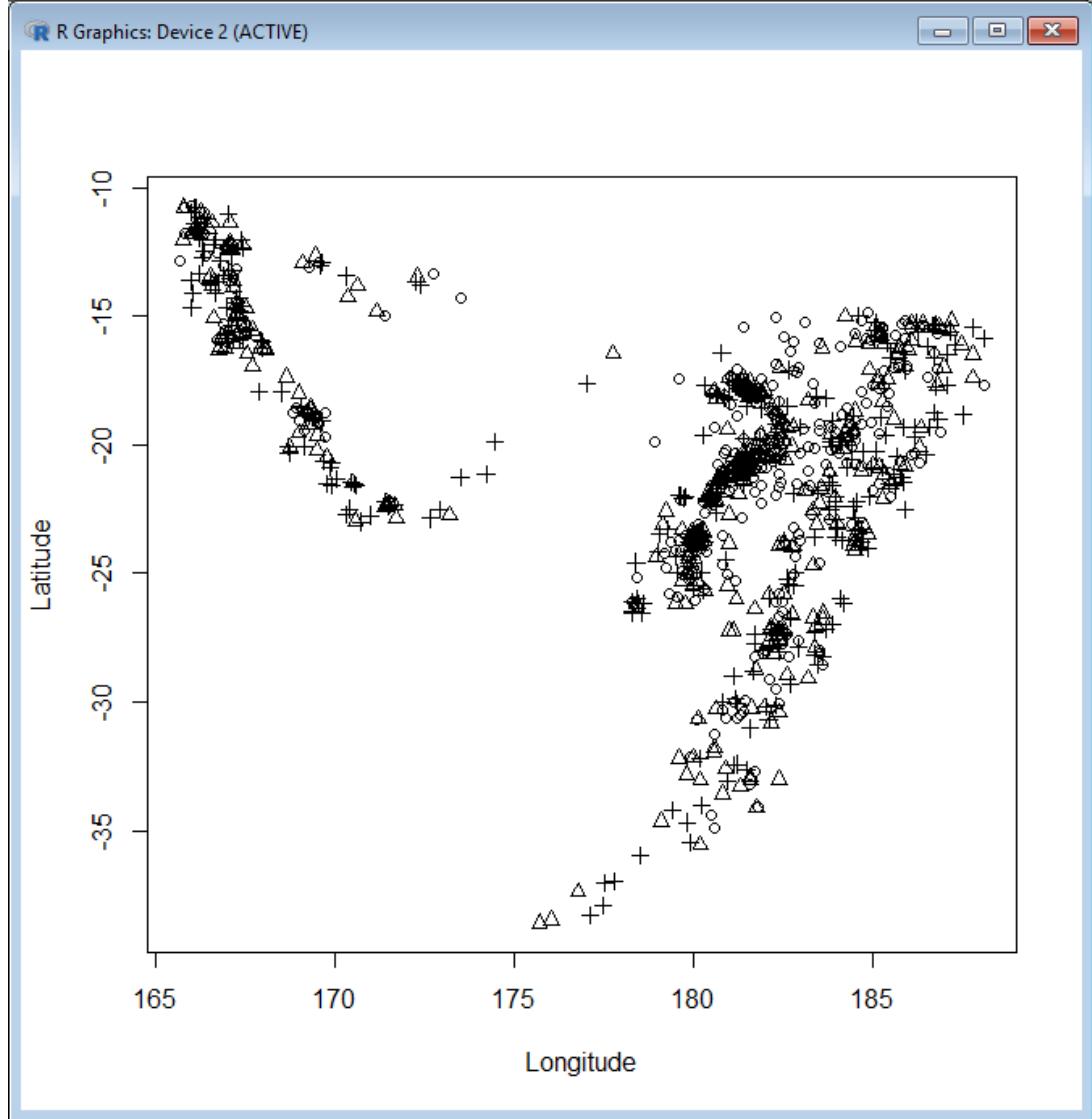
```
> #(i)
```

```
> magquan <- quantile(quakes$mag,c(1/3,2/3))
```

```
> magfac <-
cut(quakes$mag,breaks=c(min(quakes$mag),magquan[1],magquan[2],max(quakes$mag)),include.
lowest=TRUE)
```

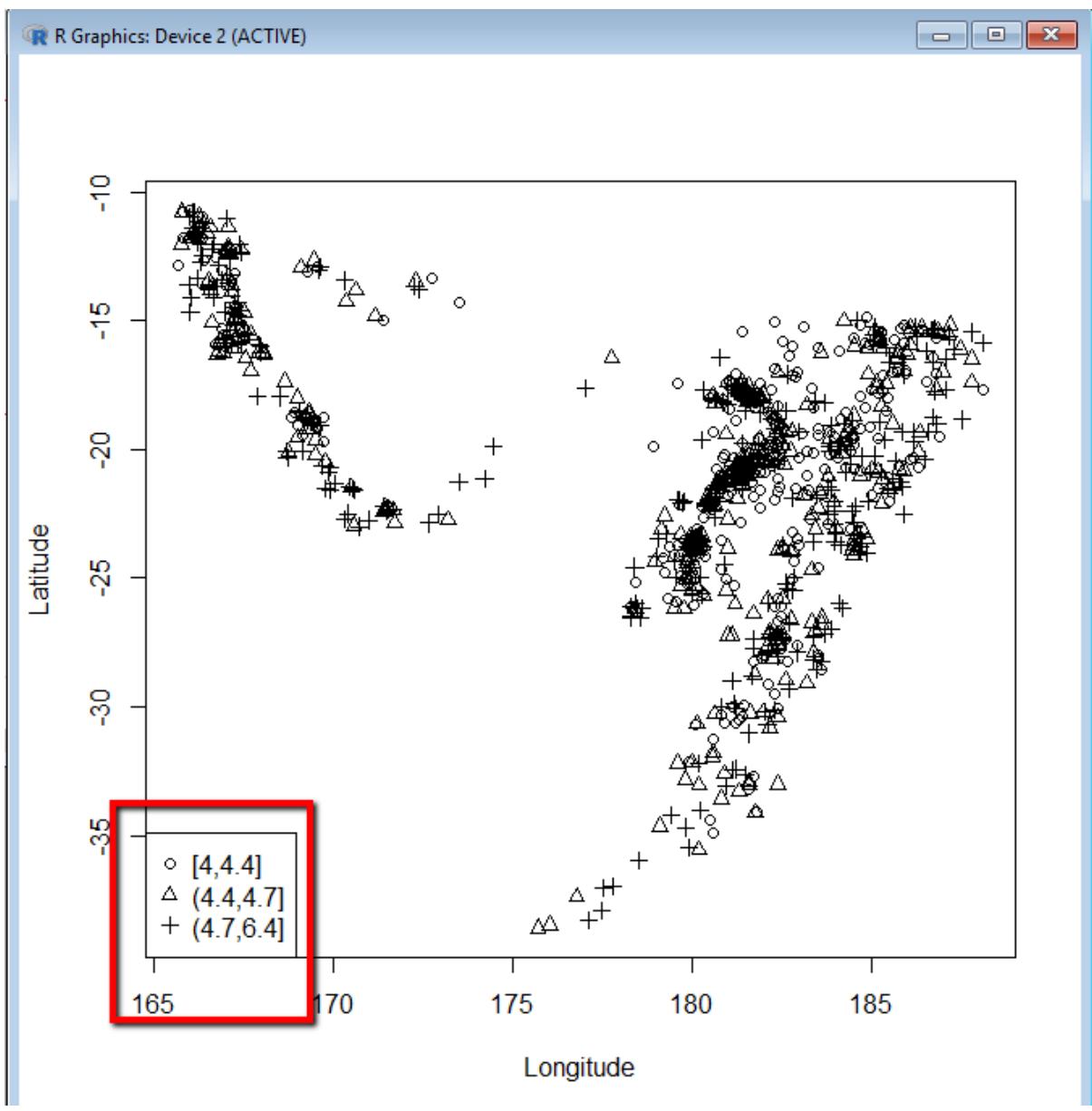
```
> #(j)
```

```
> plot(quakes[,2],quakes[,1],pch=(1:3)[magfac],xlab="Longitude",ylab="Latitude")
```



```
> #(k)
```

```
> legend("bottomleft",legend=levels(magfac),pch=1:3)
```



>

# Chapter 15: Probability

Estimated time to complete: **30 minutes**

## Exercise 15.1

You have a standard deck of 52 playing cards. There are two colors (black and red) and four suits (spades are black, clubs are black, hearts are red, and diamonds are red). Each suit has 13 cards, in which there is an ace, numbered cards from 2 to 10, and three face cards (jack, queen, and king).

- a. You randomly draw and then replace a card. What's the probability it's an ace? What's the probability it's the 4 of spades?
- b. You randomly draw a card, and after replacing it, you draw another. Let A be the event that the card is a club; let B be the event that the card is red. What is  $\Pr(A|B)$ ? That is, what is the probability the second card is a club, *given* the first one was a red card? Are the two events independent?
- c. Repeat (b), this time assuming that when the first (club) card is drawn, it is not replaced. Would this change your answer to (b) in terms of independence?
- d. Let C be the event a card is a face card, and let D be the event a card is black. You draw a single card. Evaluate  $\Pr(C \cap D)$ . Are the two events mutually exclusive?

## Solution 15.1

> #(a)

> 4/52 # Pr(Ace)

[1] 0.07692308

> 1/52 # Pr(4 of spades)

[1] 0.01923077

> #(b)

> 13/52 # Pr(A|B)==Pr(A), so the two events are independent

[1] 0.25

> #(c)

> 13/52 # Pr(A)

[1] 0.25

> 13/51 # Pr(A|B) Pr(A|B)!=Pr(A), so the two events are no longer independent

[1] 0.254902

> #(d)

> 12/52 # Pr(C)

[1] 0.2307692

> 26/52 # Pr(D)

[1] 0.5

>  $6/52 \# \Pr(C \text{ and } D) = \Pr(C|D)*\Pr(D) = (6/26)*(26/52) = 6/52 \neq 0$  therefore C and D are not mutually exclusive

[1] 0.1153846

## Exercise 15.2

- a. For each of the following definitions, identify whether it's best described as a random variable or as a *realization* of a random variable. Furthermore, identify whether each statement describes a continuous or a discrete quantity.

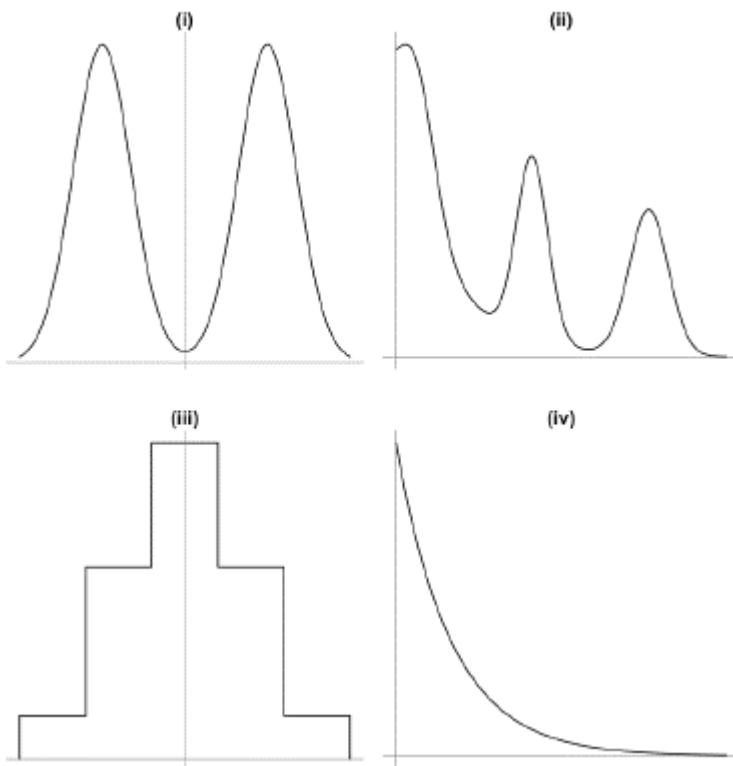
- i. The number of coffees  $x$  made by your local shop on June 3, 2016
- ii. The number of coffees  $X$  made by your local shop on any given day
- iii.  $Y$ , whether or not it rains tomorrow
- iv.  $Z$ , the amount of rain that falls tomorrow
- v. How many crumbs  $k$  on your desk right now
- vi. Total collective weight  $W$  of the crumbs on your desk at any specified time

- b. Suppose you construct the following table providing probabilities associated with the random variable  $S$ , the total stars given to any movie in a particular genre by a certain critic:

$s$	1	2	3	4	5
$\Pr(S = s)$	0.10	0.13	0.21	???	0.15

- i. Assuming this table describes the complete set of outcomes, evaluate the missing probability  $\Pr(S = 4)$ .
  - ii. Obtain the cumulative probabilities.
  - iii. What is the mean of  $S$ , the expected number of stars this critic will award any given movie in this genre?
  - iv. What is the standard deviation of  $S$ ?
  - v. What is the probability that any given movie in this genre will be given at least three stars?
  - vi. Visualize, and briefly comment on the appearance of, the probability mass function.
- c. Return to the picnic temperature example based on the random variable  $W$  defined in Section 15.2.3.
- i. Write an R function to return  $f(w)$  as per Equation (15.5) for any numeric vector of values supplied as  $w$ . Try to avoid using a loop in favor of vector-oriented operations.
  - ii. Write an R function to return  $F(w)$  as per Equation (15.6) for any numeric vector of values supplied as  $w$ . Again, try to avoid using a loop, either explicit or implicit.
  - iii. Use your functions from (i) and (ii) to confirm the results from the text, in other words, that  $f(55.2) = 0.02432$  and that  $F(55.2) = 0.184832$ .
  - iv. Make use of your function for  $F(w)$  to compute  $\Pr(W > 60)$ .
  - v. Hint: Note that because the total area underneath  $f(w)$  is one,  $\Pr(W > 60) = 1 - \Pr(W \leq 60)$ .

- d. Assume each of the following plots labeled (i)–(iv) shows the general appearance of a probability distribution. Use terminology from Section 15.2.4 to describe the shape of each.



## Solution 15.2

```
> #(a)
> ##(i) Realization of discrete random variable
> ##(ii) Discrete random variable
> ##(iii) Discrete random variable
> ##(iv) Continuous random variable
> ##(v) Realization of discrete random variable
> ##(vi) Continous random variable
```

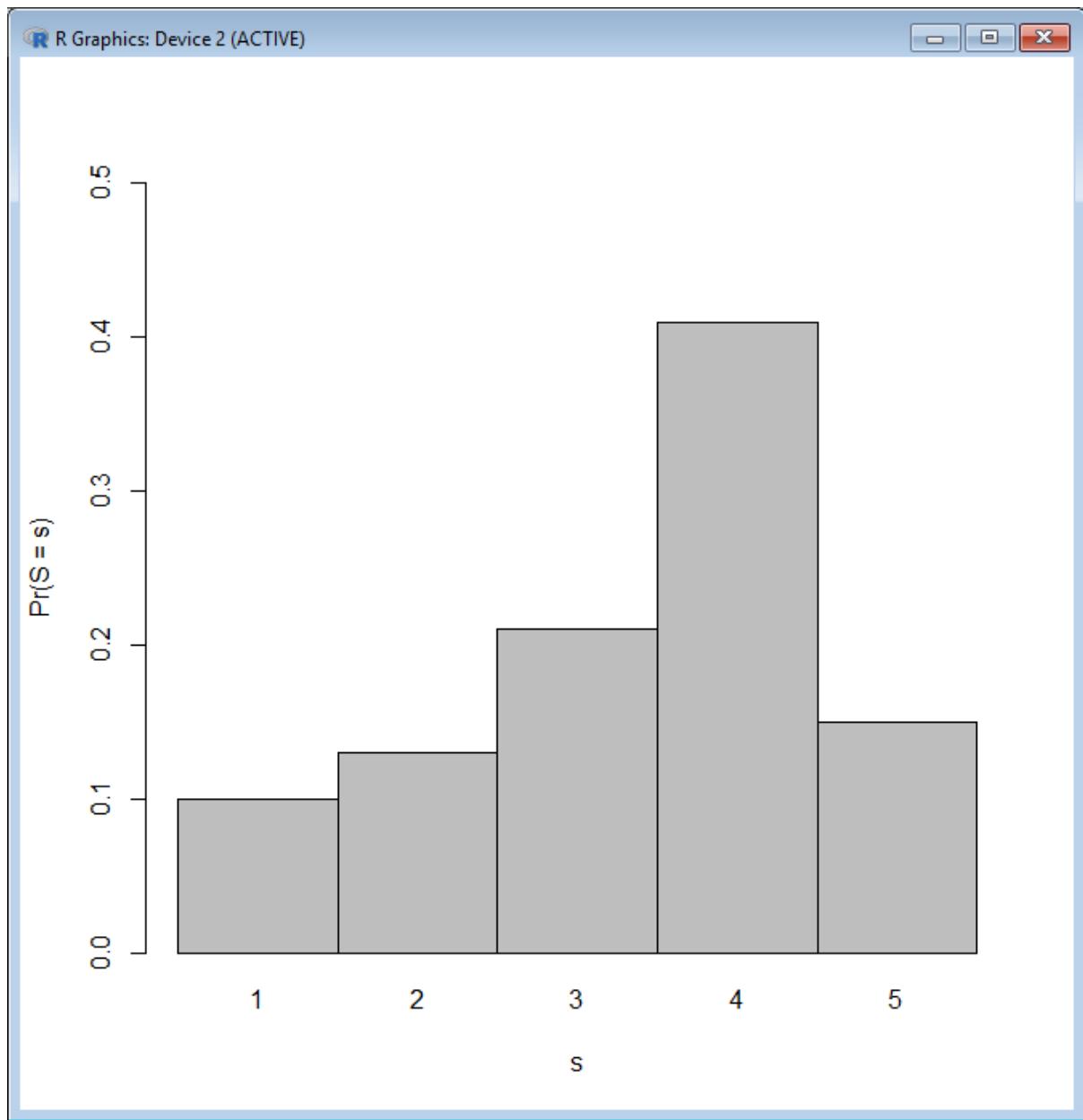
```
> #(b)
```

```
> S.outcomes <- 1:5
> ##(i)
> 1-0.1-0.13-0.21-0.15 # Pr(S=4)
[1] 0.41
> S.prob <- c(0.1,0.13,0.21,0.41,0.15)
> ##(ii)
```

```

> cumsum(S.prob)
[1] 0.10 0.23 0.44 0.85 1.00
> ##(iii)
> mu.S <- sum(S.prob*S.outcomes)
> mu.S
[1] 3.38
> ##(iv)
> sqrt(sum(S.prob*(S.outcomes-mu.S)^2))
[1] 1.181355
> ##(v)
> sum(S.prob[3:5])
[1] 0.77
> ##(vi)
> barplot(S.prob,ylim=c(0,0.5),names.arg=S.outcomes,space=0,xlab="s",ylab="Pr(S = s)") ####
Unimodal, asymmetric -- slight left skew

```



```

> #(c)

> ##(i)

>fw <- function(w) {
  w.upper <- w>65 & w<=90
  w.lower <- w>=40 & w<=65

  result <- rep(0,length(w))
  result[w.upper] <- (90-w[w.upper])/625
  result[w.lower] <- (w[w.lower]-40)/625
  return(result)
}

> ##(ii)

> Fw <- function(w) {
  w.upper <- w>65 & w<=90
  w.lower <- w>=40 & w<=65

```

```

result <- rep(0,length(w))
result[w.upper] <- (180*w[w.upper]-w[w.upper]^2-6850)/1250
result[w.lower] <- (w[w.lower]^2-80*w[w.lower]+1600)/1250
result[w>90] <- 1

return(result)
}
>##(iii)

>fw(55.2)

[1] 0.02432

>Fw(55.2)

[1] 0.184832

>##(iv)

>1-Fw(60)

[1] 0.68

>##(v)

>Fw(76.89)-Fw(60.3)

[1] 0.5328303

>#(d)

>##(i) Bimodal, symmetric

>##(ii) Trimodal, asymmetric -- right skew

>##(iii) Unimodal, symmetric

>##(iv) Unimodal, asymmetric -- right skew

```

# Chapter 16: Common Probability Distributions

Estimated time to complete: **60 minutes**

## Exercise 16.1

A forested nature reserve has 13 bird-viewing platforms scattered throughout a large block of land. The naturalists claim that at any point in time, there is a 75 percent chance of seeing birds at each platform. Suppose you walk through the reserve and visit every platform. If you assume that all relevant conditions are satisfied,

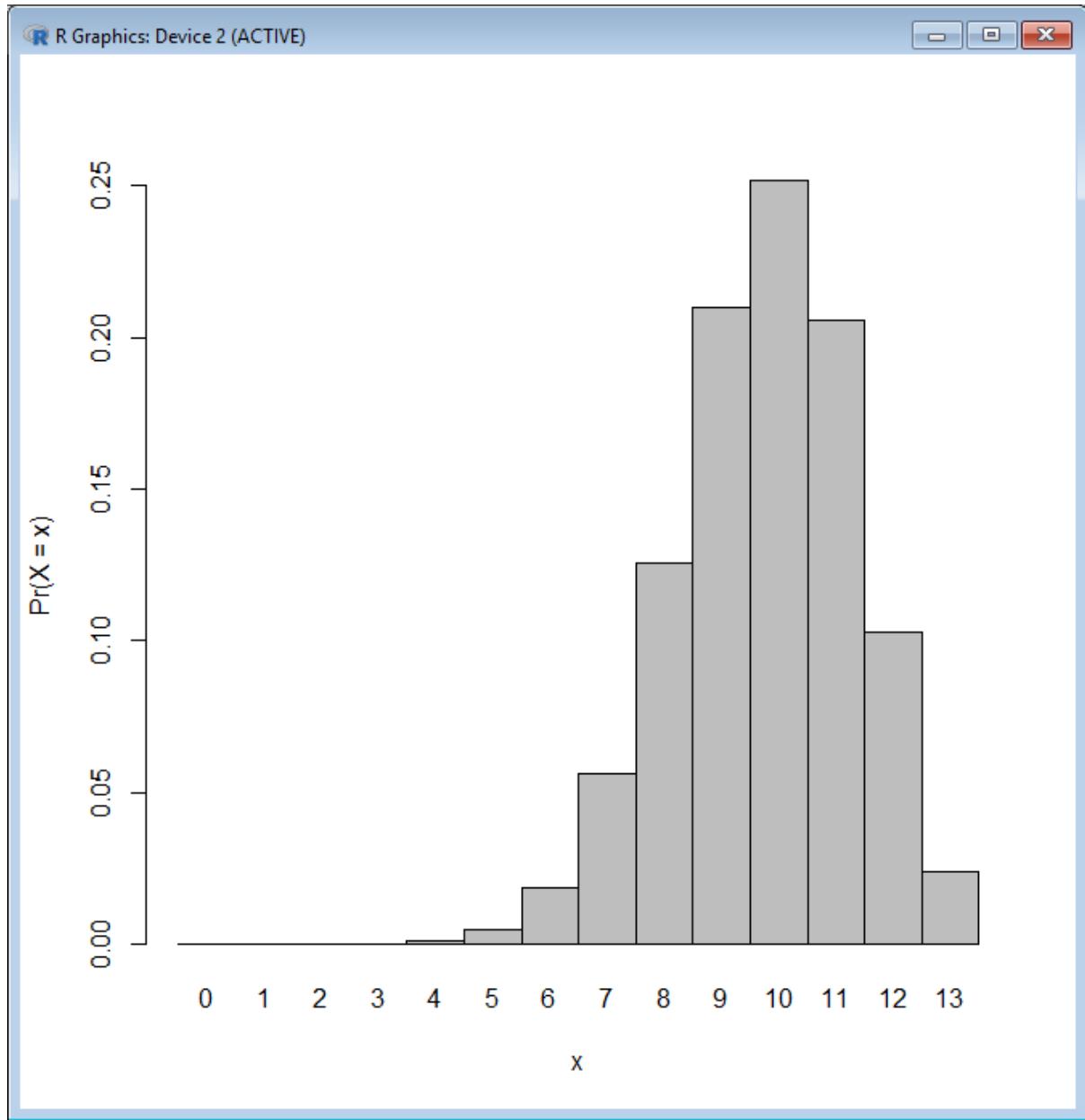
let  $X$  be a binomial random variable representing the total number of platforms at which you see birds.

- a. Visualize the probability mass function of the binomial distribution of interest.
- b. What is the probability you see birds at all sites?
- c. What is the probability you see birds at more than 9 platforms?
- d. What is the probability of seeing birds at between 8 and 11 platforms (inclusive)? Confirm your answer by using only the  $d$ -function and then again using only the  $p$ -function.
- e. Say that, before your visit, you decide that if you see birds at fewer than 9 sites, you'll make a scene and demand your entry fee back. What's the probability of your embarrassing yourself in this way?
- f. Simulate realizations of  $X$  that represent 10 different visits to the reserve; store your resulting vector as an object.
- g. Compute the mean and standard deviation of the distribution of interest.

## Solution 16.1

```
> #(a)
```

```
> barplot(dbinom(x=0:13,size=13,prob=0.75),names.arg=0:13,space=0,xlab="x",ylab="Pr(X = x)")
```



```
> #(b)
```

```
> dbinom(13,13,0.75)
```

```
[1] 0.02375726
```

```
> #(c)
```

```
> 1-pbinom(q=9,13,0.75)
```

```
[1] 0.5842527
```

```
> #(d)
```

```
> sum(dbinom(8:11,13,0.75))
[1] 0.793082
> pbinom(11,13,0.75)-pbinom(7,13,0.75)
[1] 0.793082
> #(e)
> pbinom(8,13,0.75)
[1] 0.2060381
> #(f)
> visits <- rbinom(10,13,0.75)
> visits
[1] 8 10 11 8 11 11 10 8 12 9
> #(g)
> mu.X <- 13*0.75
> mu.X
[1] 9.75
> sigma.X <- sqrt(mu.X*0.25)
> sigma.X
[1] 1.561249
```

## Exercise 16.2

Every Saturday, at the same time, an individual stands by the side of a road and tallies the number of cars going by within a 120-minute window. Based on previous knowledge, she believes that the mean number of cars going by during this time is exactly 107. Let  $X$  represent the appropriate Poisson random variable of the number of cars passing her position in each Saturday session.

- a. What is the probability that more than 100 cars pass her on any given Saturday?
- b. Determine the probability that no cars pass.
- c. Plot the relevant Poisson mass function over the values in  $60 \leq x \leq 150$ .
- d. Simulate 260 results from this distribution (about five years of weekly Saturday monitoring sessions). Plot the simulated results using `hist`; use `xlim` to set the horizontal limits from 60 to 150.

Compare your histogram to the shape of your mass function from (c).

## Solution 16.2

```
> #(a)  
> 1-ppois(100,107)  
[1] 0.7319128  
> #(b)  
> dpois(0,107)  
[1] 3.39227e-47  
> #(c)  
> barplot(dpois(x=60:150,lambda=107),names.arg=60:150,space=0,xlab="x",ylab="Pr(X = x)")  
> #(d)  
> traffic <- rpois(n=260,107)  
> hist(traffic,xlim=c(60,150))
```

## Exercise 16.3

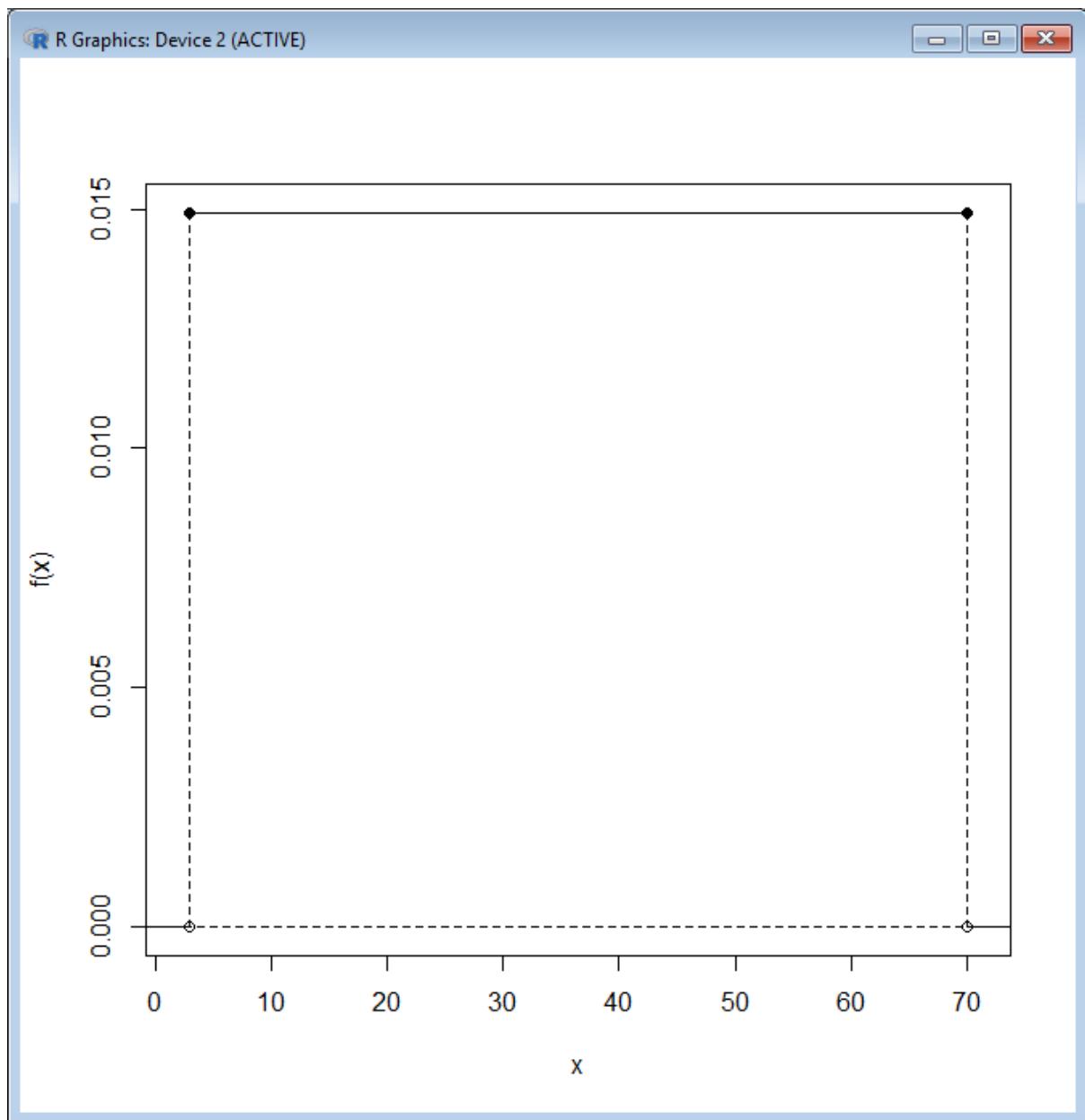
You visit a national park and are informed that the height of a certain species of tree found in the forest is uniformly distributed between 3 and 70 feet.

- a. What is the probability you encounter a tree shorter than  $5\frac{1}{2}$  feet?
- b. For this probability density function, what is the height that marks the cutoff point of the tallest 15 percent of trees?
- c. Evaluate the mean and standard deviation of the tree height distribution.
- d. Using (c), confirm that the chance that you encounter a tree with a height that is within half a standard deviation (that is, below or above) of the mean height is roughly 28.9 percent.
- e. At what height is the density function itself? Show it in a plot.
- b. Simulate 10 observed tree heights. Based on these data, use *quantile* (refer to Section 13.2.3) to estimate the answer you arrived at in (b). Repeat your simulation, this time generating 1,000 variates, and estimate (b) again. Do this a handful of times, taking a mental note of your two estimates each time.
- c. Overall, what do you notice of your two estimates (one based on 10 variates at a time and the other based on 1,000) with respect to the “true” value in (b)?

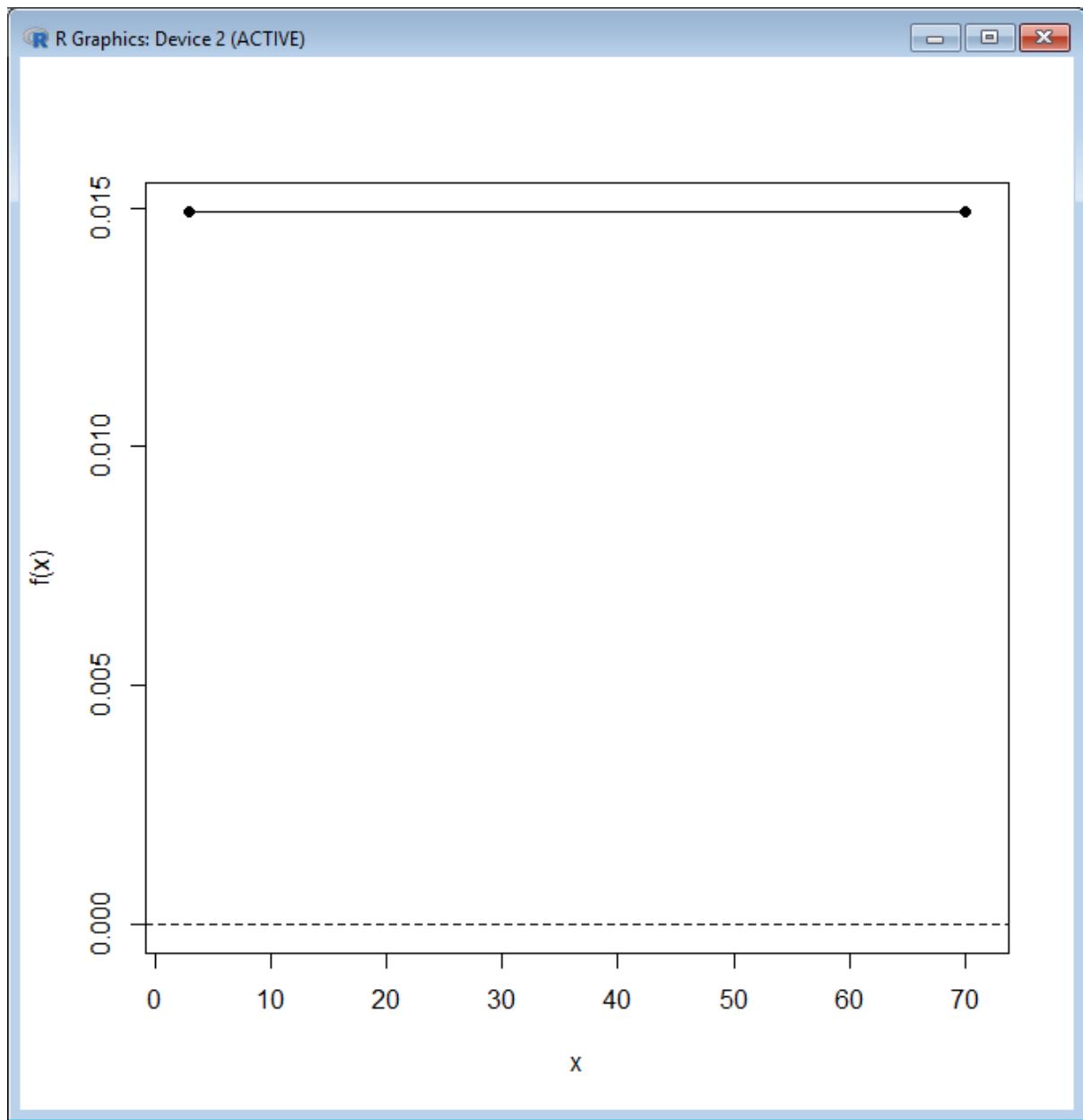
## Solution 16.3

```
> a <- 3
> b <- 70
> #(a)
> punif(q=5.5,min=a,max=b)
[1] 0.03731343
> #(b)
> qunif(p=1-0.15,min=a,max=b)
[1] 59.95
> #(c)
> mu.X <- (a+b)/2 #mean
> mu.X
[1] 36.5
> sigma.X <- sqrt((b-a)^2/12) #sd
> sigma.X
[1] 19.34123
> #(d)
> punif(mu.X+0.5*sigma.X,a,b) - punif(mu.X-0.5*sigma.X,a,b)
[1] 0.2886751
> #(e)
```

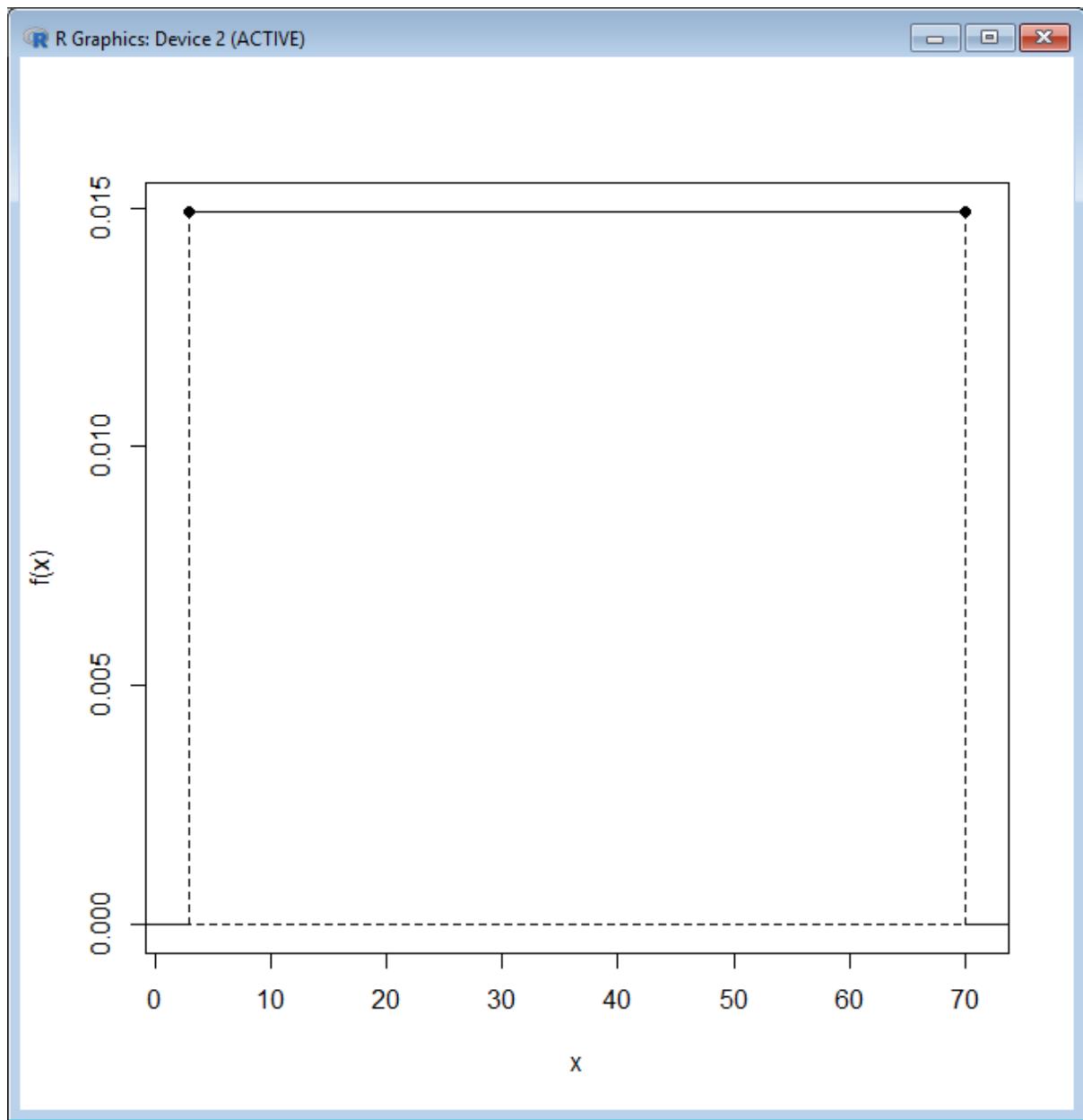
```
> X.dens <- dunif(mu.X,a,b)  
> X.dens  
[1] 0.01492537  
> plot(c(a,b),rep(X.dens,2),type="o",pch=19,xlim=c(a-  
1,b+1),ylim=c(0,X.dens),ylab="f(x)",xlab="x")
```



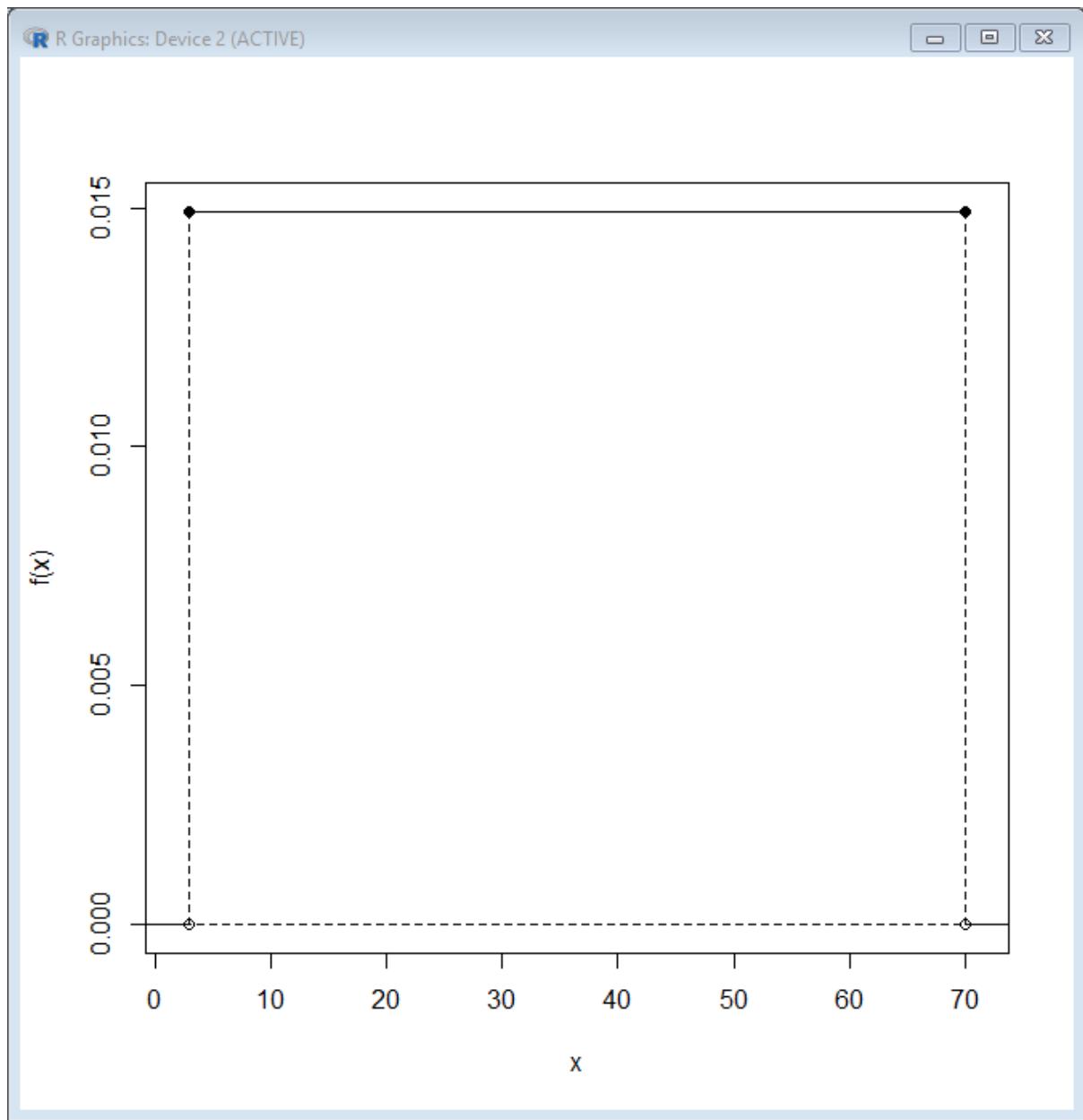
```
> abline(h=0,lty=2)
```



```
> segments(c(a-,b+5,a,b),rep(0,4),rep(c(a,b),2),rep(c(0,X.dens),each=2),lty=rep(1:2,each=2))
```



```
> points(c(a,b),c(0,0))
```



```
> #(f)
> sim1 <- runif(10,a,b)
> sim1
[1] 26.622855 5.166131 48.508436 60.881629 32.754983 37.061884 9.004976 37.387800
13.399594 11.637715
> quan1 <- quantile(sim1,prob=1-0.15)
> quan1
85%
44.61621
> sim2 <- runif(1000,a,b)
> quan2 <- quantile(sim2,prob=1-0.15)
```

> quan2    ### Overall, both estimates seem to be centered on the 'true' value from (b), but those based on sim1 (the smaller samples) are more variable.

85%

59.90529

## Exercise 16.4

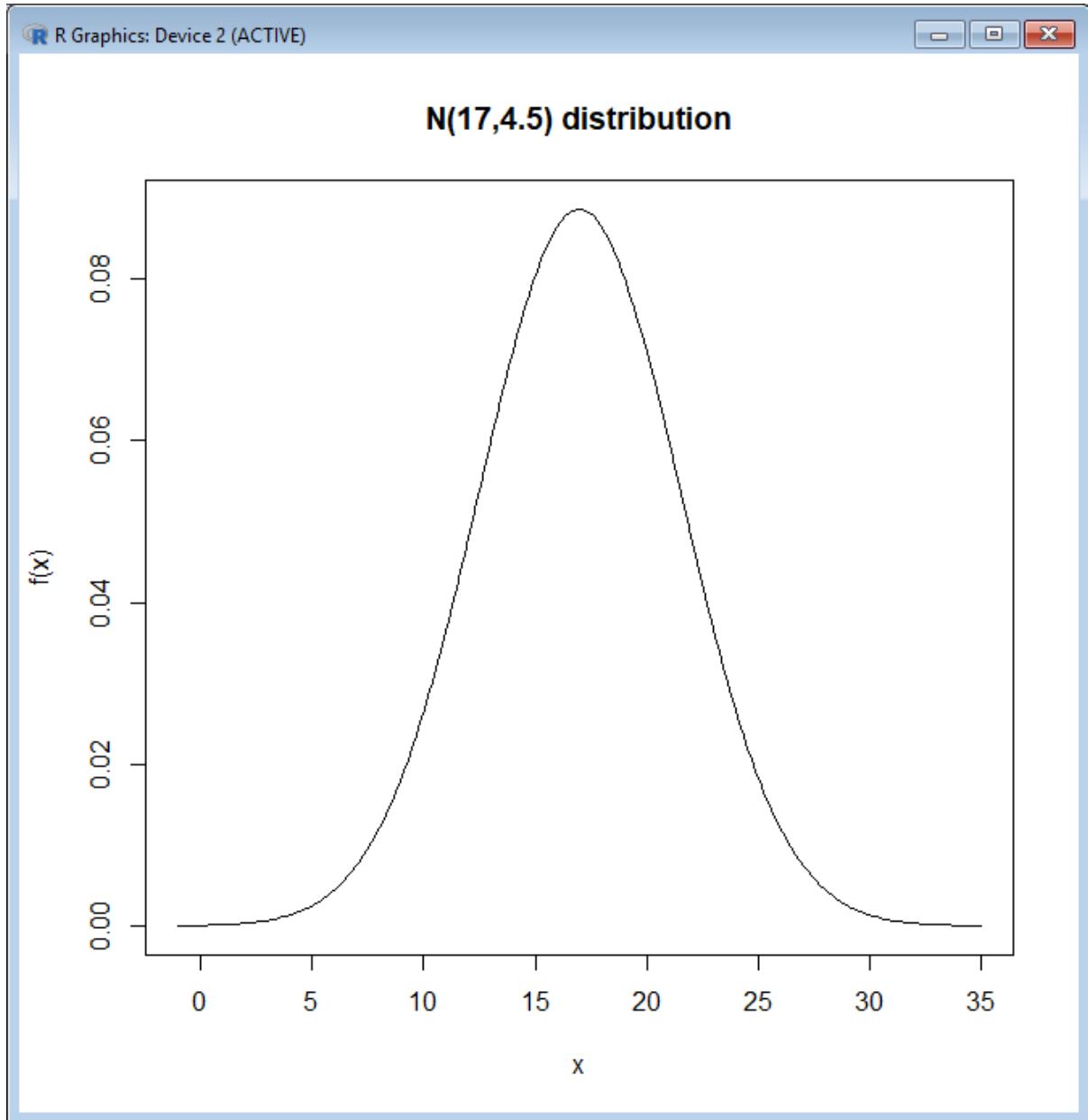
- a. A tutor knows that the length of time taken to complete a certain statistics question by first-year undergraduate students,  $X$ , is normally distributed with a mean of 17 minutes and a standard deviation of 4.5 minutes.
  - i. What is the probability a randomly selected undergraduate takes more than 20 minutes to complete the question?
  - ii. What's the chance that a student takes between 5 and 10 minutes to finish the question?
  - iii. Find the time that marks off the slowest 10 percent of students.
  - iv. Plot the normal distribution of interest between  $\pm 4\sigma$  and shade in the probability area of (iii), the slowest 10 percent of students.
  - v. Generate a realization of times based on a class of 10 students completing the question.
- b. A meticulous gardener is interested in the length of blades of grass on his lawn. He believes that blade length  $X$  follows a normal distribution centered on 10 mm with a variance of 2 mm.
  - i. Find the probability that a blade of grass is between 9.5 and 11 mm long.
  - ii. What are the standardized values of 9.5 and 11 in the context of this distribution? Using the standardized values, confirm that you can obtain the same probability you found in (i) with the standard normal density.
  - iii. Below which value are the shortest 2.5 percent of blade lengths found?
  - iv. Standardize your answer from (iii).

## Solution 16.4

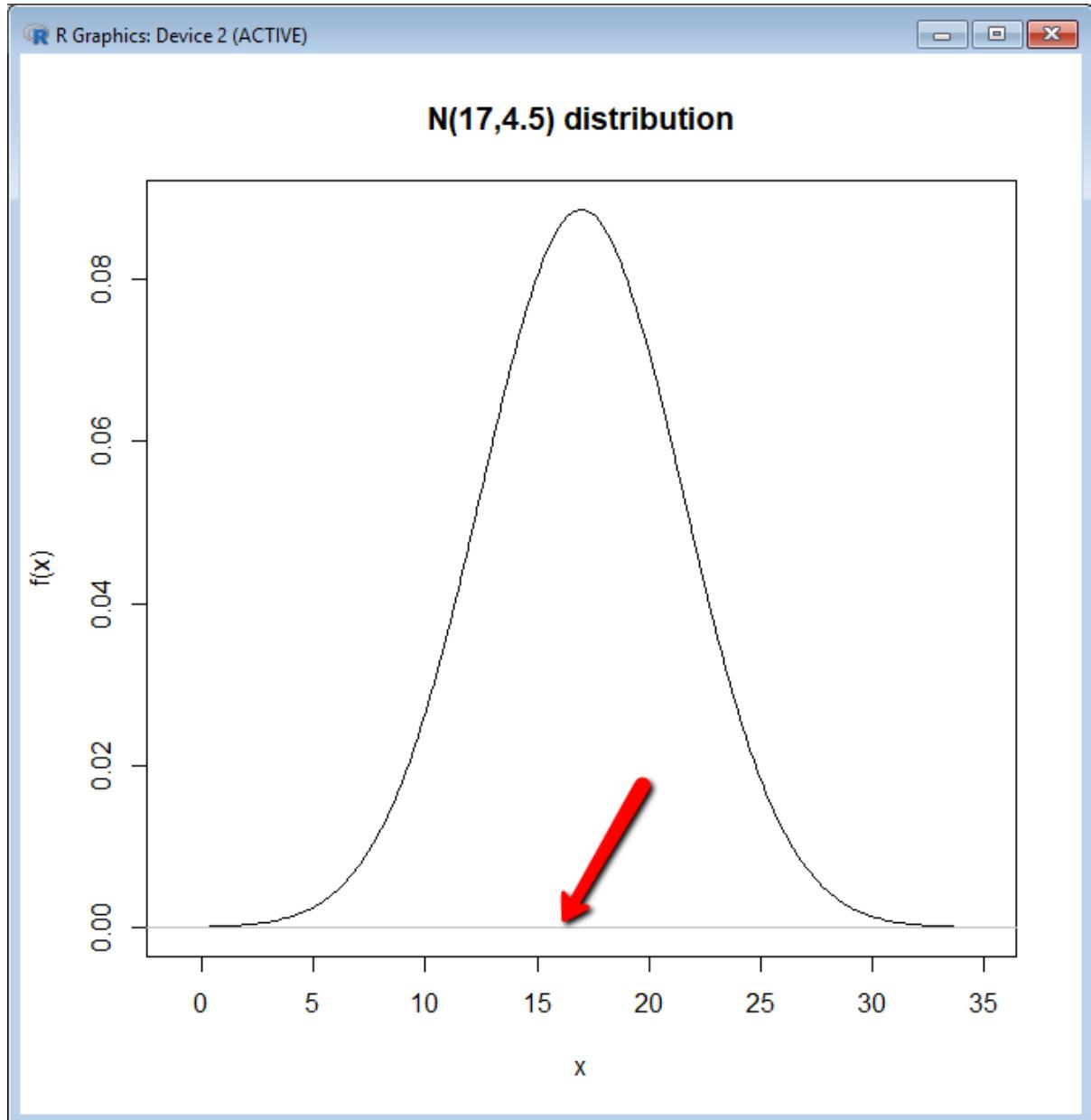
```
> #(a)  
  
> mu <- 17  
  
> sigma <- 4.5  
  
> ##(i)  
  
> 1-pnorm(20,mu,sigma)  
[1] 0.2524925  
  
> ##(ii)  
  
> pnorm(10,mu,sigma)-pnorm(5,mu,sigma)  
[1] 0.05607653  
  
> ##(iii)  
  
> slow10 <- qnorm(1-0.1,mu,sigma)  
  
> slow10  
[1] 22.76698  
  
> ##(iv)  
  
xvals <- seq(mu-4*sigma,mu+4*sigma,length=200)  
fx <- dnorm(xvals,mu,sigma)  
xvals.sub <- xvals[xvals>=slow10]
```

```
fx.sub <- fx[xvals>=slow10]
```

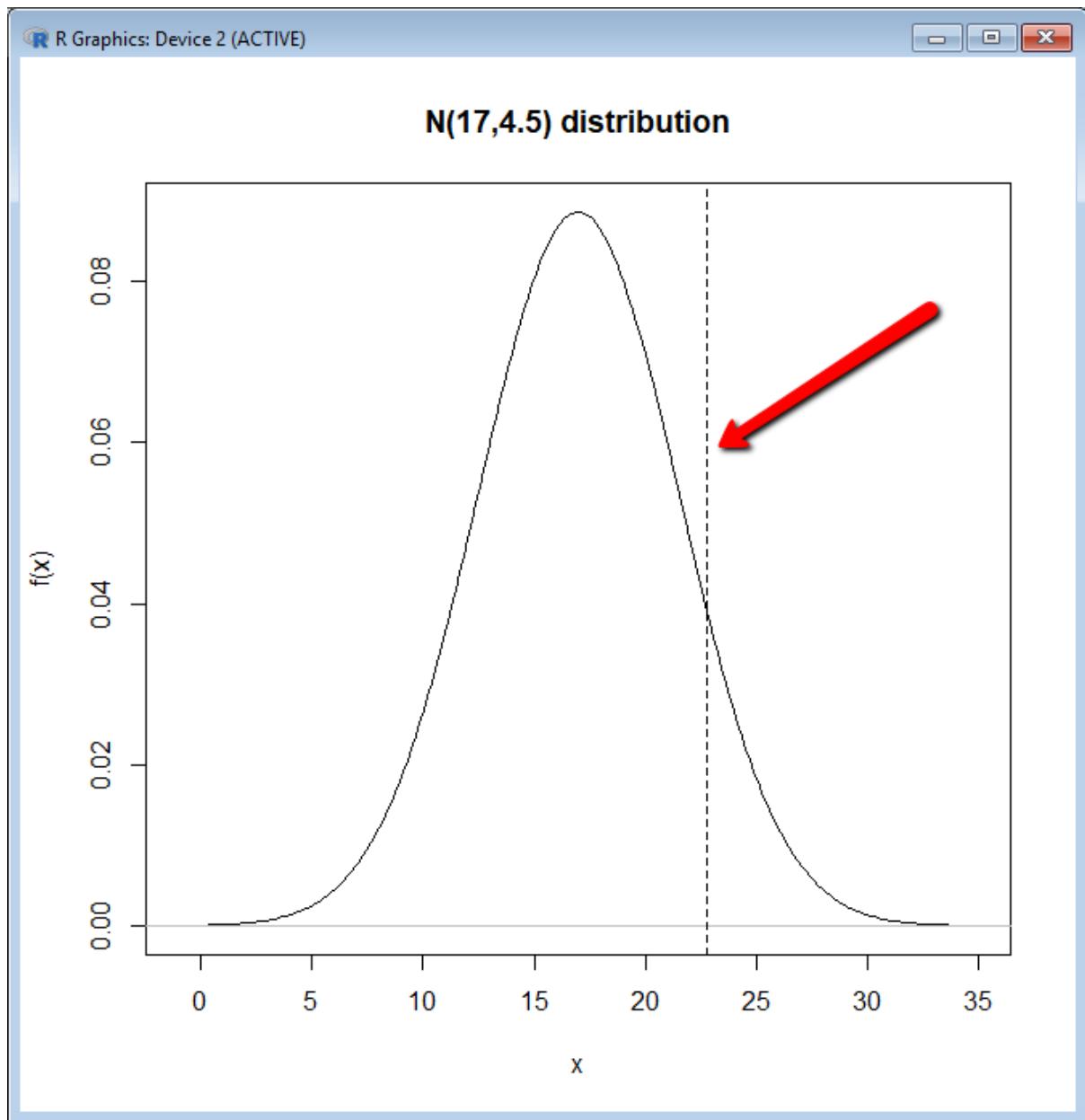
```
plot(xvals,fx,type="l",main="N(17,4.5) distribution",xlab="x",ylab="f(x)")
```



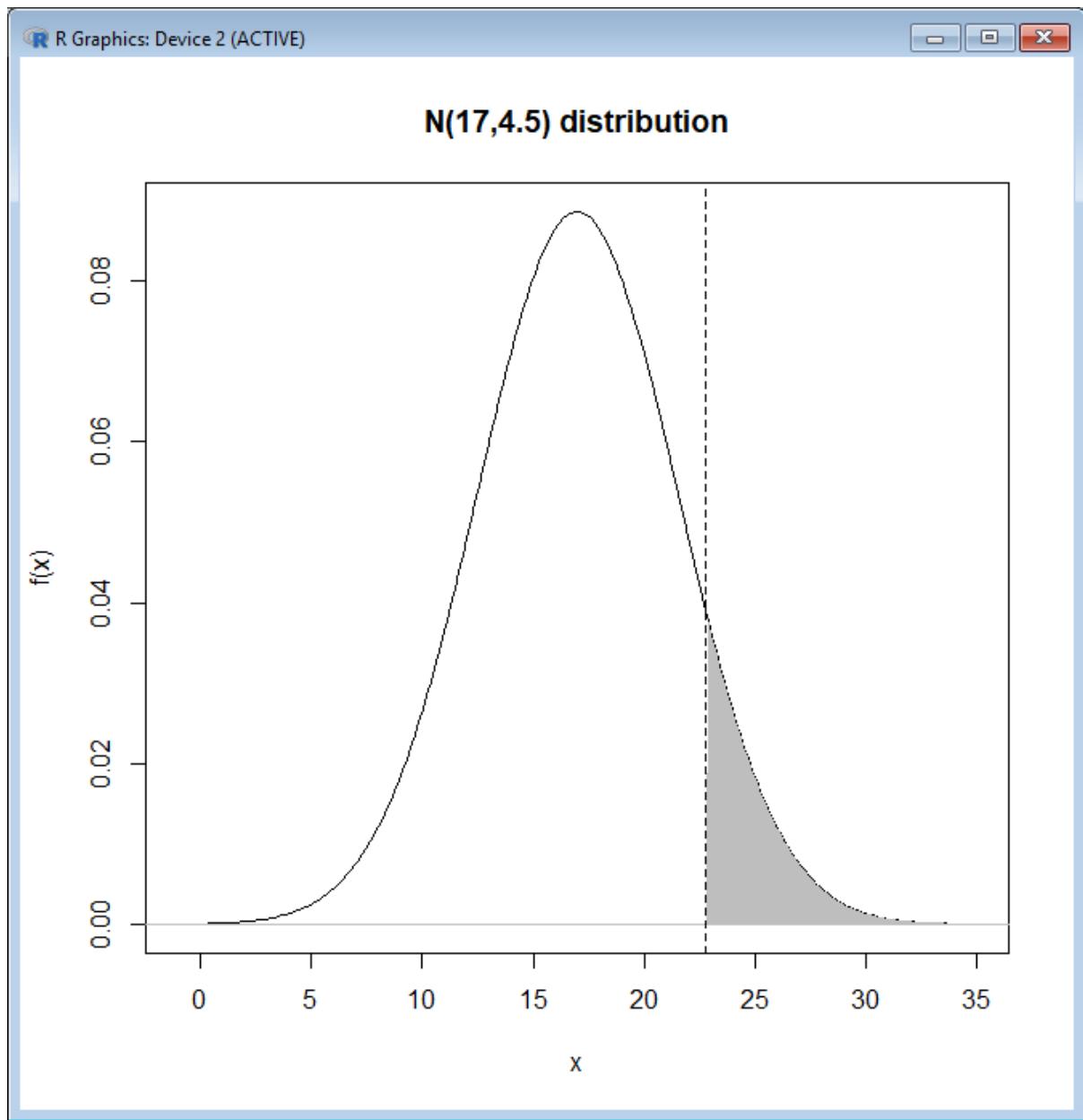
```
abline(h=0,col="gray")
```



```
abline(v=slow10,lty=2)
```



```
> polygon(rbind(c(slow10,0),cbind(xvals.sub,fx.sub),c(max(xvals),0)),border=NA,col="gray")
```



```

> ##(v)
> rnorm(10,mu,sigma)
[1] 6.724849 12.435094 14.485262 18.330514 18.675714 14.210625 13.140441 12.824613
15.442698 19.195762

> #(b)
> mu <- 10
> sigma <- sqrt(2)
> ##(i)
> pnorm(11,mu,sigma)-pnorm(9.5,mu,sigma)
[1] 0.3984131
> ##(ii)

```

```
> stan9.5 <- (9.5-mu)/sigma
> stan9.5
[1] -0.3535534
> stan11 <- (11-mu)/sigma
> stan11
[1] 0.7071068
> pnorm(stan11)-pnorm(stan9.5)
[1] 0.3984131
> ##(iii)
> short2.5 <- qnorm(0.025,mu,sigma)
> short2.5
[1] 7.228192
> ##(iv)
> (short2.5-mu)/sigma
[1] -1.959964
>
```

## Exercise 16.5

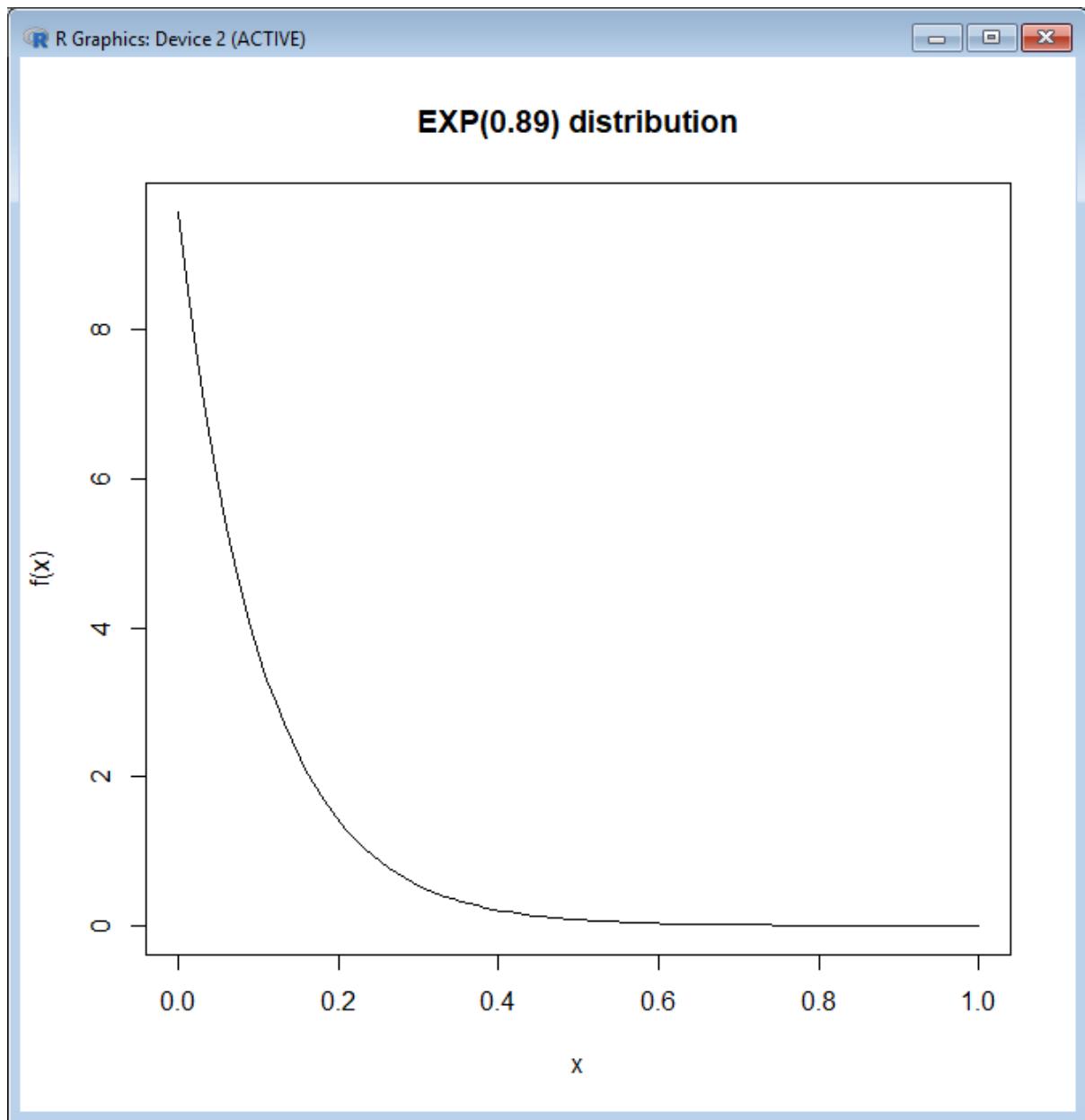
- a. Situated in the central north island of New Zealand, the Pohutu geyser is said to be the largest active geyser in the southern hemisphere. Suppose that it erupts an average of 3,500 times every year.
- With the intention of modeling a random variable  $X$  as the time between consecutive eruptions, evaluate the parameter value  $\lambda e$  with respect to a time scale in days (assume 365.25 days per year to account for leap years).
  - Plot the density function of interest. What's the mean wait in days between eruptions?
  - What's the probability of waiting less than 30 minutes for the next eruption?
  - What waiting time defines the longest 10 percent of waits?

Convert your answer to hours.

- b. You can also use the exponential distribution to model certain product survival times, or “time-to-failure” type of variables. Say a manufacturer of a particular air conditioning unit knows that the product has an average life of 11 years before it needs any type of repair callout. Let the random variable  $X$  represent the time until the necessary repair of one of these units and assume  $X$  follows an exponential distribution with  $\lambda e = 1/11$ .
- The company offers a five-year full repair warranty on this unit. What's the probability that a randomly selected air conditioner owner makes use of the warranty?
  - A rival company offers a six-year guarantee on its competing air conditioning unit but knows that its units last, on average, only nine years before requiring some kind of repair. What are the chances of making use of that warranty?
  - Determine the probabilities that the units in (i) and the units in (ii) last more than 15 years.

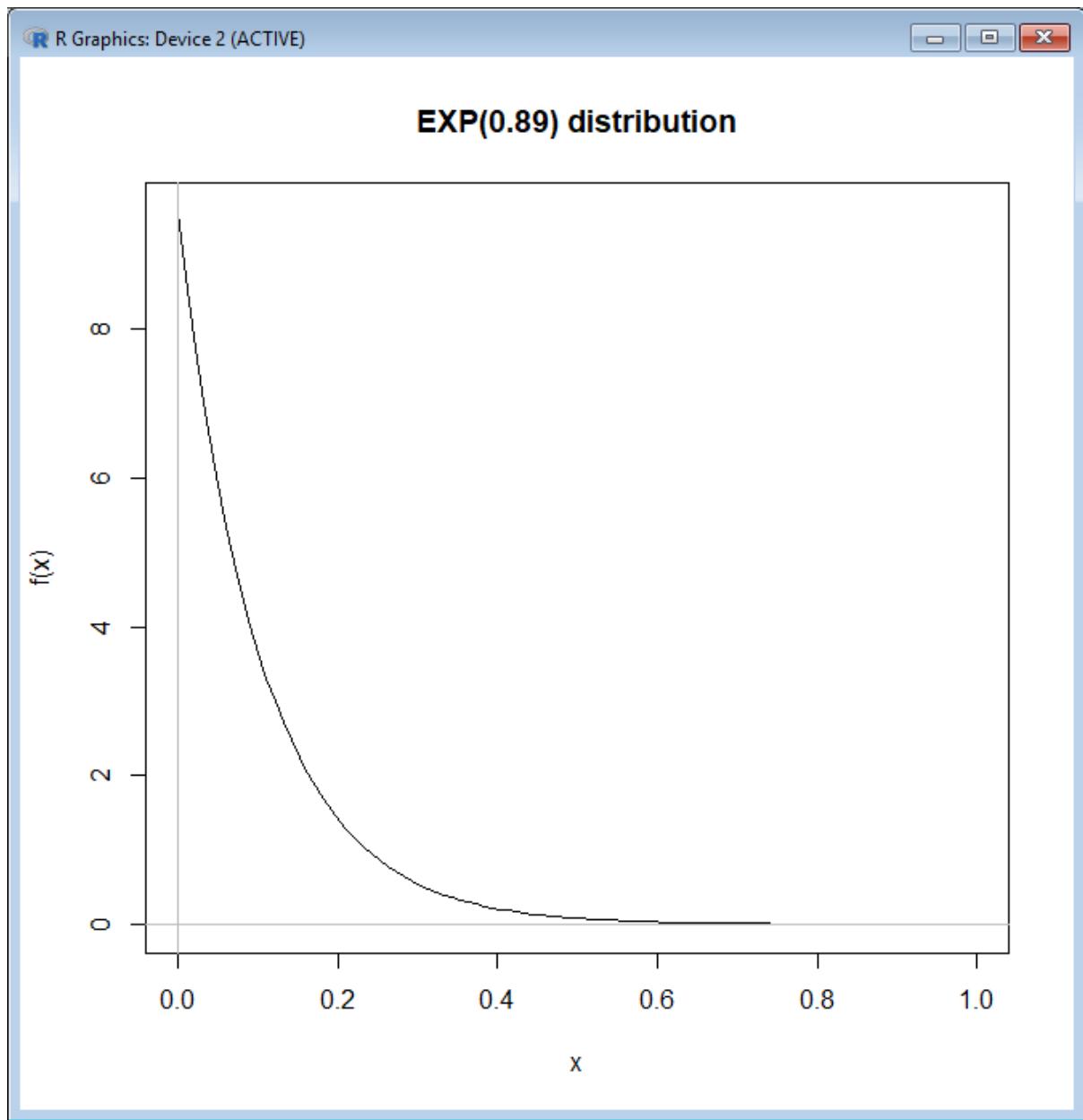
## Solution 16.5

```
> #(a)  
> ##(i)  
  
> lambda.day <- 3500/365.25  
  
> lambda.day  
[1] 9.582478  
  
> ##(ii)  
  
> xvals <- seq(0,1,length=100)  
  
> plot(xvals,dexp(xvals,lambda.day),type="l",xlab="x",ylab="f(x)",main="EXP(0.89)  
distribution")
```



```
> abline(h=0,col="gray")
```

```
> abline(v=0,col="gray")
```



```
> ##(iii)
> pexp(0.5/24,rate=lambda.day)
[1] 0.1809703
> ##(iv)
> qexp(1-0.1,rate=lambda.day)*24
[1] 5.766989
> #(b)
> ##(i)
> pexp(5,1/11)
[1] 0.3652636
```

```
> ##(ii)  
> pexp(6,1/9)  
[1] 0.4865829  
> ##(iii)  
> 1-pexp(15,1/11)  
[1] 0.2557292  
> 1-pexp(15,1/9)  
[1] 0.1888756
```

## PART 4

# Chapter 17: Sampling Distributions and Confidence

Estimated time to complete: **30 minutes**

### Exercise 17.1

A teacher wants to test all of the 10th-grade students at his school to gauge their basic mathematical understanding, but the photocopier breaks after making only six copies of the test. With no other choice, he chooses six students at random to take the test. Their results, recorded as a score out of 65, have a sample mean of 41.1. The standard deviation of the marks of this test is known to be 11.3.

- a. Find the standard error associated with the mean test score.
- b. Assuming the scores themselves are normally distributed, evaluate the probability that the mean score lies between 45 and 55 if the teacher took another sample of the same size.
- c. A student who gets less than half the questions correct receives a failing grade (F). Find the probability that the average score is an F based on another sample of the same size.

A marketing company wants to find out which of two energy drinks teenagers prefer—drink A or drink B. It surveys 140 teens, and the results indicate that only 35 percent prefer drink A.

- d. Use a quick check to decide whether it is valid to use the normal distribution to represent the sampling distribution of this proportion.
- e. What is the probability that in another sample of the same size, the proportion of teenagers who prefer drink A is greater than 0.4?
- f. Find the two values of this sampling distribution that identify the central 80 percent of values of the proportion of interest.

In Section 16.2.4, the time between cars passing an individual's location was modeled using an exponential distribution. Say that on the other side of town, her friend is curious about a similar problem.

Standing outside her house, she records 63 individual times between cars passing. These sampled times have a mean of  $\bar{x} = 37.8$  seconds with a standard deviation of  $s = 34.51$  seconds.

- g. The friend inspects a histogram of her raw measurements and notices that her raw data are heavily right-skewed. Briefly identify and describe the nature of the sampling distribution
- h. with respect to the sample mean and calculate the appropriate standard error.
- i. Using the standard error from (g) and the appropriate probability distribution, calculate the probability that in another sample of the same size, the sample mean time between cars passing is as follows:
  - i. More than 40 seconds
  - ii. Less than half a minute

iii. Between the given sample mean and 40 seconds

### Solution 17.1

```
> xbar <- 41.1  
> #(a)  
> se <- 11.3/sqrt(6)  
> se  
[1] 4.613206  
> #(b)  
> pnorm(55,mean=41.1,sd=se)-pnorm(45,mean=41.1,sd=se)  
[1] 0.197651  
> #(c)  
> pnorm(32.5,41.1,se)  
[1] 0.03114587  
> #(d)  
> 140*0.35  
[1] 49  
> 140*(1-0.35) # Both are > 5 so rule-of-thumb says using the normal distribution to represent the sampling distribution of the sample proportion is okay.  
[1] 91  
> #(e)  
> 1-pnorm(0.4,mean=0.35,sd=sqrt(0.35*0.65/140))  
[1] 0.1074235  
> #(f)  
> qnorm(0.9,0.35,sqrt(0.35*0.65/140)) # Upper limit - tail above this value has probability 0.1  
[1] 0.401661  
> qnorm(0.1,0.35,sqrt(0.35*0.65/140)) # Lower limit - tail below this value has probability 0.1.  
Together, these two limits therefore mark of a central area under the curve of exactly 0.8.  
[1] 0.298339  
> #(g)  
> # Even though raw data are not normal, sample size is large ( $n>30$  by rule-of-thumb). Standard deviation estimated from sample, so sampling distribution for the mean should be a t-distribution with  $63-1=62$  df.  
> se <- 34.51/sqrt(63)
```

```
> se  
[1] 4.347851  
> #(h)  
> ##(i)  
> 1-pt((40-37.8)/se,df=62)  
[1] 0.3073266  
> ##(ii)  
> pt((30-37.8)/se,df=62)  
[1] 0.03884552  
> ##(iii)  
> pt((40-37.8)/se,df=62)-0.5  
[1] 0.1926734
```

## Exercise 17.2

A casual runner records the average time it takes him to sprint 100 meters. He completes the dash 34 times under identical conditions and finds that the mean of these is 14.22 seconds. Assume that he knows the standard deviation of his runs is  $\sigma_X = 2.9$  seconds.

- a. Construct and interpret a 90 percent confidence interval for the true mean time.
- b. Repeat (a), but this time, assume that the standard deviation is not known and that  $s = 2.9$  is estimated from the sample. How, if at all, does this change the interval?

In a particular country, the true proportion of citizens who are left handed or ambidextrous is unknown. A random sample of 400 people is taken, and each individual is asked to identify with one of three options: right-handed only, left-handed only, or ambidextrous.

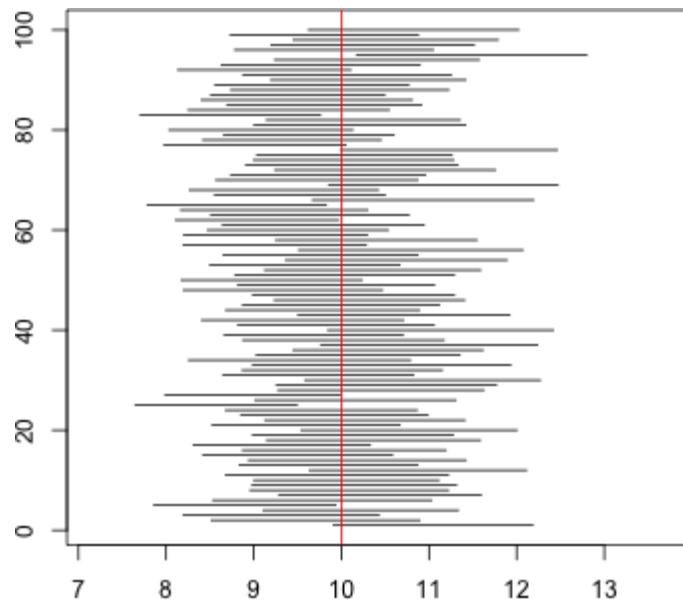
The results show that 37 selected left-handed and 11 selected ambidextrous.

- c. Calculate and interpret a 99 percent CI for the true proportion of left-handed-only citizens.
- d. Calculate and interpret a 99 percent CI for the true proportion of citizens who are either left-handed *or* ambidextrous.

In Section 17.2.4, the technical interpretation of a CI with respect to its confidence level was described as the proportion of many similar intervals (that is, when calculated for samples of the same size from the same population) that contain the true value of the parameter of interest.

- e. Your task is to write an example to demonstrate this behavior of confidence intervals using simulation. To do so, follow these instructions:
  - Set up a matrix (see Chapter 3) filled with *NAs* (Chapter 6) that has 5,000 rows and 3 columns.
  - Use skills from Chapter 10 to write a *for* loop that, at each of 5,000 iterations, generates a random sample of size 300 from an exponential distribution with rate parameter  $\lambda_e = 0.1$  (Section 16.2.4).
  - Evaluate the sample mean and sample standard deviation of each sample, and use these quantities with the critical values from the appropriate sampling distribution to calculate a 95 percent CI for the true mean of the distribution.
  - Within the *for* loop, the matrix should now be filled, row by row, with your results. The first column will contain the lower limit, the second will contain the upper limit, and the third column will be a logical value that is *TRUE* if the corresponding interval contains the true mean of  $1/\lambda_e$  and that is *FALSE* otherwise.
  - When the loop is completed, compute the proportion of *TRUEs* in the third column of the filled matrix. You should find that this proportion is close to 0.95; this will vary randomly each time you rerun the loop.
- f. Create a plot that draws the first 100 of your estimated confidence intervals as separate horizontal lines drawn from  $l$  to  $u$ , one on top of another. One way to do this is to first create an empty plot with preset x- and y-limits (the latter as  $c(1,100)$ ) and then progressively add each line using *lines* with appropriate coordinates (this could be done using another *for* loop). As a final touch, add to the plot a red vertical line that denotes the true mean. Confidence intervals that do not include the true mean will not intersect that vertical line.

The following shows an example of this plot:



## Solution 17.2

```
> x.bar <- 14.22
```

```
> sigma <- 2.9
```

```
> #(a)
```

```
> x.bar+c(-1,1)*qnorm(0.95)*sigma/sqrt(34) # When sd is known, sampling distribution of sample  
mean is normal. 90% confident that the true mean run time lies somewhere between 13.40 and  
15.04 seconds (rounded 2 d.p.)
```

```
[1] 13.40194 15.03806
```

```
> #(b)
```

```
> x.bar+c(-1,1)*qt(0.95,df=33)*sigma/sqrt(34) # When sd is estimated from sample, sampling  
distribution is t with n-1 df. This means more extreme critical values and wider 90% CI when  
compared to normal version.
```

```
[1] 13.37831 15.06169
```

```
> #(c)
```

```
> p.hat <- 37/400
```

```
> p.hat
```

```
[1] 0.0925
```

```
> p.hat+c(-1,1)*qnorm(0.995)*sqrt(p.hat*(1-p.hat)/400) # 99% confident that the true proportion  
of left-handedness only is somewhere between 0.055 and 0.130 (rounded 3 d.p.)
```

```
[1] 0.05518519 0.12981481
```

```
> #(d)
```

```
> p.hat <- (37+11)/400
```

```

> p.hat
[1] 0.12

> p.hat+c(-1,1)*qnorm(0.995)*sqrt(p.hat*(1-p.hat)/400) # 99% confident that the true proportion
of left-handed or ambidextrous citizens is somewhere between 0.078 and 0.162 (rounded 3 d.p.)

[1] 0.07814773 0.16185227

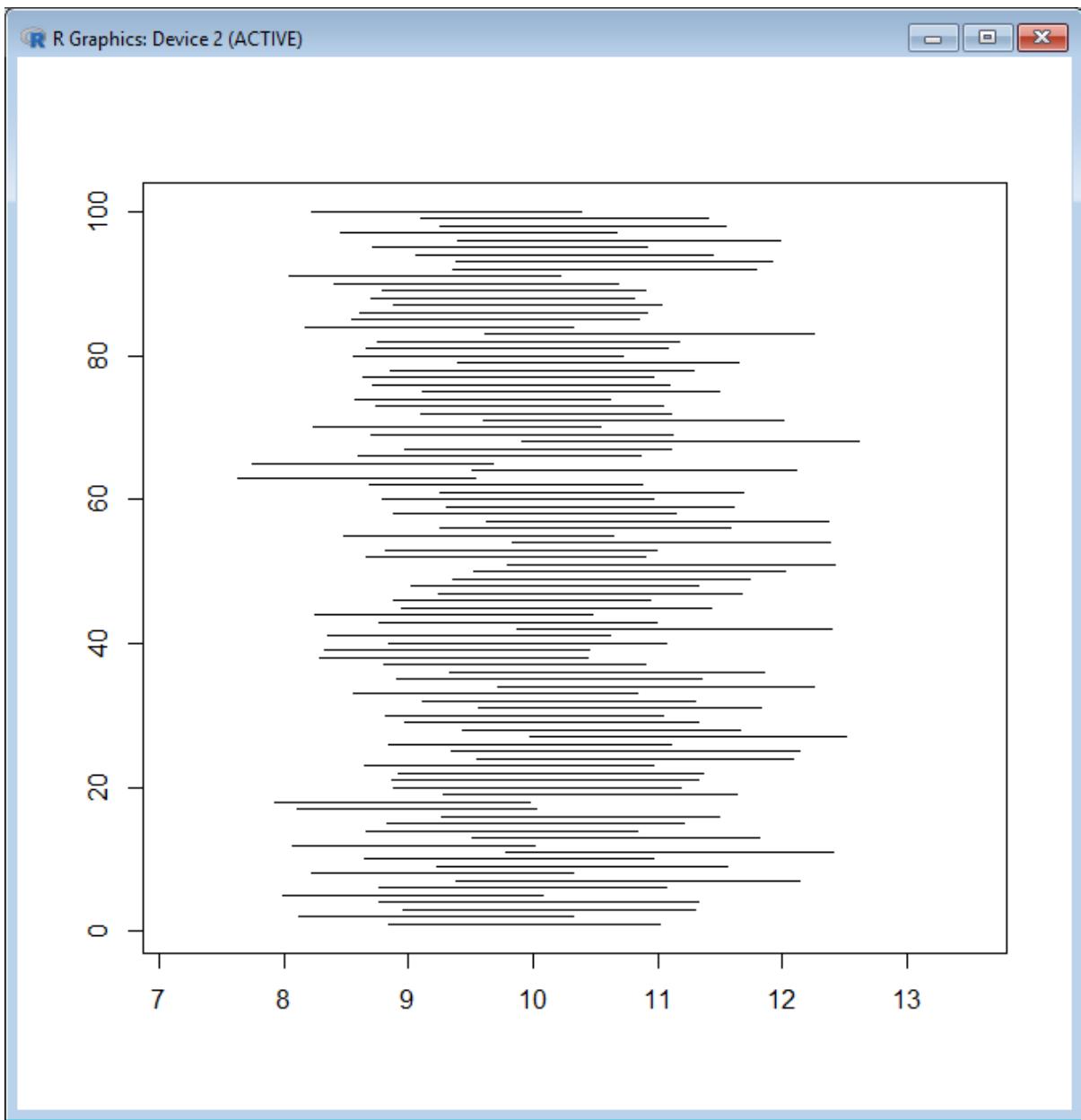
> #(e)

>ci.mat <- matrix(NA,5000,3)
n <- 300
lambda.e <- 0.1
mu <- 1/lambda.e
for(i in 1:5000){
  samp <- rexp(n,rate=lambda.e)
  samp.ci <- mean(samp)+c(-1,1)*qt(0.975,n-1)*sd(samp)/sqrt(n)
  ci.mat[i,1:2] <- samp.ci
  ci.mat[i,3] <- mu>=samp.ci[1] && mu<=samp.ci[2]
}
mean(ci.mat[,3])
[1] 0.9488

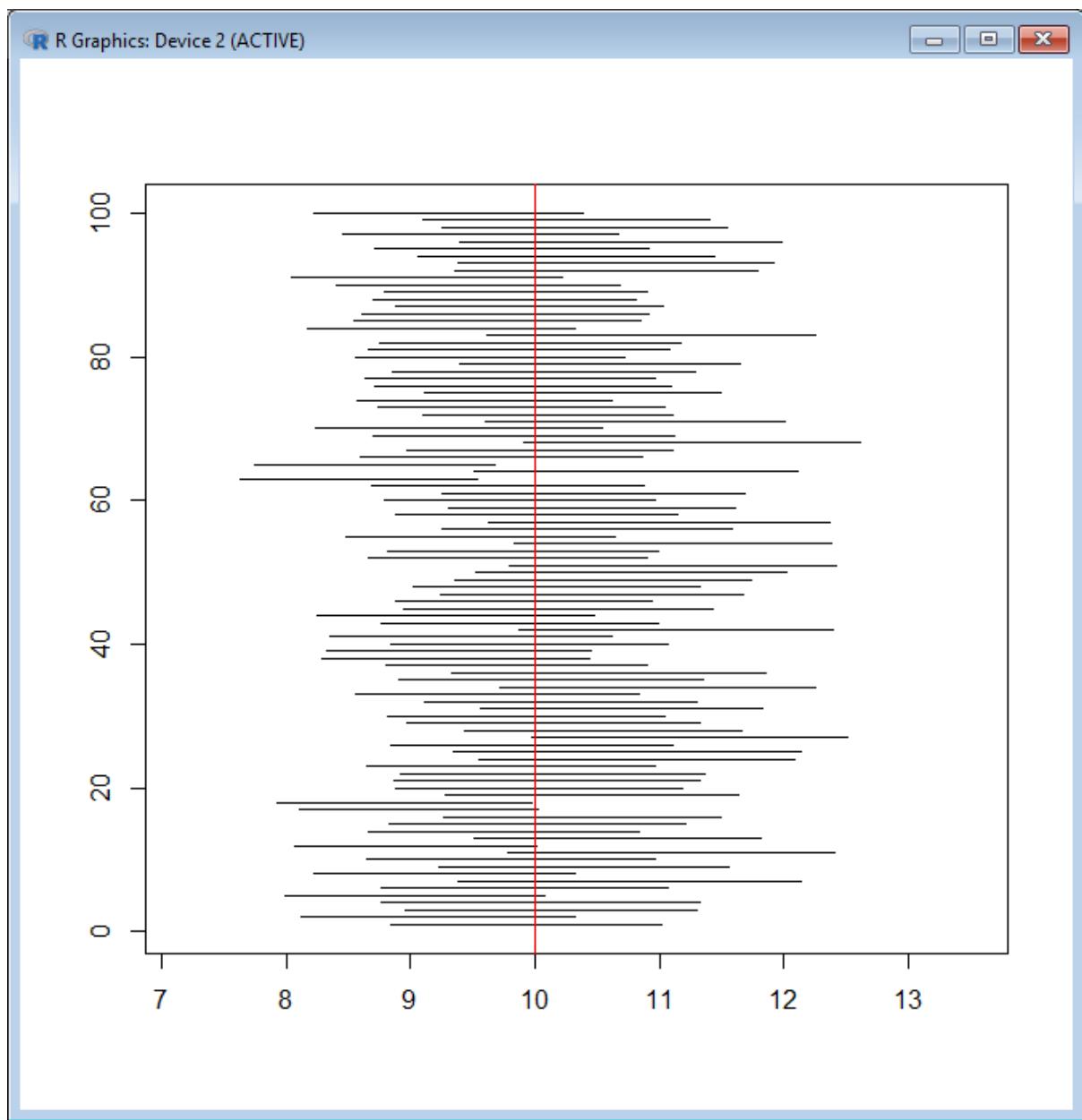
> #(f)

>plot(ci.mat[1,1:2],c(1,1),xlim=range(ci.mat[,1:2]),ylim=c(1,100),type=
"l",xlab="",ylab="")
for(i in 2:100){
  lines(ci.mat[i,1:2],c(i,i))
}

```



```
> abline(v=mu,col=2)
```



>

# Chapter 18: Hypothesis Testing

Estimated time to complete: **90 minutes**

## Exercise 18.1

- a. Adult domestic cats of a certain breed are said to have an average weight of 3.5 kilograms. A feline enthusiast disagrees and collects a sample of 73 weights of cats of this breed. From her sample, she calculates a mean of 3.97 kilograms and a standard deviation of 2.21 kilograms. Perform a hypothesis test to test her claim that the true mean weight  $\mu$  is *not* 3.5 kilograms by setting up the appropriate hypothesis, carrying out the analysis, and interpreting the p-value (assume the significance level is  $\alpha = 0.05$ ).
- b. Suppose it was previously believed that the mean magnitude of seismic events off the coast of Fiji is 4.3 on the Richter scale. Use the data in the *mag* variable of the ready-to-use *quakes* data set, providing 1,000 sampled seismic events in that area, to test the claim that the true mean magnitude is in fact *greater* than 4.3.

Set up appropriate hypotheses, use *t.test* (conduct the test at a significance level of  $\alpha = 0.01$ ), and draw a conclusion.

- c. Manually compute a two-sided confidence interval for the true mean of (b).

## Solution 18.1

```
> #(a)  
> ### H0: mu = 3.5; HA: mu != 3.5 (two-sided test)  
> tstat <- (3.97-3.5)/(2.21/sqrt(73))  
> tstat  
[1] 1.817051  
[1] 0.07337077  
> ### p-value is around 0.073, this is > than alpha=0.05, therefore insufficient evidence to reject  
the null hypothesis. There is no evidence that the true mean cat weight is different to 3.5.  
> #(b)  
> # H0: mu = 4.3; HA: mu > 4.3 (one-sided test)  
> t.test(quakes$mag,mu=4.3,alternative="greater",conf.level=0.99)
```

One Sample t-test

```
data: quakes$mag  
t = 25.155, df = 999, p-value < 2.2e-16  
alternative hypothesis: true mean is greater than 4.3  
99 percent confidence interval:  
 4.590722     Inf  
sample estimates:  
mean of x  
4.6204
```

> # p-value very small; strong evidence to reject the null. There is evidence to suggest that the true mean magnitude is greater than 4.3.

```
> #(c)  
> mean(quakes$mag)+c(-1,1)*qt(0.995,df=999)*sd(quakes$mag)/sqrt(1000)  
[1] 4.587529 4.653271  
>
```

## Exercise 18.2

In the package *MASS* you'll find the data set *anorexia*, which contains data on pre- and post-treatment weights (in pounds) of 72 young women suffering from the disease, obtained from Hand et al. (1994).

One group of women is the control group (in other words, no intervention), and the other two groups are the cognitive behavioral program and family support intervention program groups. Load the library and ensure you can access the data frame and understand its contents. Let  $\mu_d$  denote the mean difference in weight, computed as (*post-weight* – *pre-weight*).

- a. Regardless of which treatment group the participants fall into, conduct and conclude an appropriate hypothesis test with  $\alpha = 0.05$  for the entire set of weights for the following hypotheses:

$$H_0 : \mu_d = 0$$

$$H_A : \mu_d > 0$$

- b. Next, conduct three separate hypothesis tests using the same defined hypotheses, based on which treatment group the participants fall into. What do you notice?

Another ready-to-use data set in R is *PlantGrowth* (Dobson, 1983), which records a continuous measure of the yields of a certain plant, looking at the potential effect of two supplements administered during growth to increase the yield when compared to a control group with no supplement.

- c. Set up hypotheses to test whether the mean yield for the control group is less than the mean yield from a plant given either of the treatments. Determine whether this test should proceed using a pooled estimate of the variance or whether Welch's *t*-test would be more appropriate.
- d. Conduct the test and make a conclusion (assuming normality of the raw observations). As discussed, there is a rule of thumb for deciding whether to use a pooled estimate of the variance in an unpaired *t*-test.
- e. Your task is to write a wrapper function that calls *t.test* after deciding whether it should be executed with *var.equal=FALSE* according to the rule of thumb. Use the following guidelines:
  - Your function should take four defined arguments: *x* and *y* with no defaults, to be treated in the same way as the same arguments in *t.test*; and *var.equal* and *paired*, with defaults that are the same as the defaults of *t.test*.
  - An ellipsis (Section 9.2.5) should be included to represent any additional arguments to be passed to *t.test*.
  - Upon execution, the function should determine whether *paired=FALSE*.
    - \* If *paired* is TRUE, then there is no need to proceed with the check of a pooled variance.
    - \* If *paired* is FALSE, then the function should determine the value for *var.equal* automatically by using the rule of thumb.
  - If the value of *var.equal* was set automatically, you can assume it will override any value of this argument initially supplied by the user.
  - Then, call *t.test* appropriately.
- f. Try your new function on all three examples in the text of Section 18.2.2, ensuring you reach identical results.

## Solution 18.2

```
> #(a)  
> library("MASS")  
> ?anorexia  
> t.test(anorexia[,3],anorexia[,2],alternative="greater",paired=TRUE)
```

Paired t-test

```
data: anorexia[, 3] and anorexia[, 2]  
t = 2.9376, df = 71, p-value = 0.002229  
alternative hypothesis: true difference in means is greater than 0  
95 percent confidence interval:  
 1.195825     Inf  
sample estimates:  
mean of the differences  
 2.763889
```

> # p-value ~0.0023. Less than 0.05; some evidence to reject H0. There is evidence to suggest that the mean post-weight is greater than the mean pre-weight.

```
> #(b)  
>  
t.test(anorexia[anorexia$Treat=="Cont",3],anorexia[anorexia$Treat=="Cont",2],alternative="greater",paired=TRUE)
```

Paired t-test

```
data: anorexia[anorexia$Treat == "Cont", 3] and anorexia[anorexia$Treat == "Cont", 2]  
t = -0.28723, df = 25, p-value = 0.6118  
alternative hypothesis: true difference in means is greater than 0  
95 percent confidence interval:  
 -3.126168     Inf  
sample estimates:
```

mean of the differences

-0.45

>

t.test(anorexia[anorexia\$Treat=="CBT",3],anorexia[anorexia\$Treat=="CBT",2],alternative="greater",paired=TRUE)

Paired t-test

data: anorexia[anorexia\$Treat == "CBT", 3] and anorexia[anorexia\$Treat == "CBT", 2]

t = 2.2156, df = 28, p-value = 0.01751

alternative hypothesis: true difference in means is greater than 0

95 percent confidence interval:

0.6981979 Inf

sample estimates:

mean of the differences

3.006897

>

t.test(anorexia[anorexia\$Treat=="FT",3],anorexia[anorexia\$Treat=="FT",2],alternative="greater",paired=TRUE)

Paired t-test

data: anorexia[anorexia\$Treat == "FT", 3] and anorexia[anorexia\$Treat == "FT", 2]

t = 4.1849, df = 16, p-value = 0.0003501

alternative hypothesis: true difference in means is greater than 0

95 percent confidence interval:

4.233975 Inf

sample estimates:

mean of the differences

7.264706

```
> # There is no statistical evidence to reject the claim that there is no difference between the pre-and post-weight means in the control group, mild evidence to reject at the 5% level for the CBL treatment, and strong evidence to reject in favor of HA for the FT treatment. It would seem that the FT treatment is the most effective based on these data.
```

```
> #(c)
```

```
> ?PlantGrowth
```

```
> control <- PlantGrowth$weight[PlantGrowth$group=="ctrl"]
```

```
> treated <- PlantGrowth$weight[PlantGrowth$group!="ctrl"]
```

```
> # H0: mu_control - mu_treated = 0; HA: mu_control - mu_treated < 0
```

```
> max(c(sd(control),sd(treated)))/min(c(sd(control),sd(treated)))
```

```
[1] 1.315153
```

```
> # Ratio of (large sd) / (small sd) is less than 2 so use pooled variance according to rule-of-thumb.
```

```
> #(d)
```

```
> t.test(x=control,y=treated,alternative="less",var.equal=TRUE)
```

## Two Sample t-test

data: control and treated

t = -0.22272, df = 28, p-value = 0.4127

alternative hypothesis: true difference in means is less than 0

95 percent confidence interval:

-Inf 0.4082382

sample estimates:

mean of x mean of y

5.0320 5.0935

```
> # Large p-value ~0.41. There is no evidence to reject H0. There is insufficient evidence to conclude that the mean treated weight is more than the mean untreated weight.
```

```
> #(e)
```

```
> myt.test <- function(x,y,paired=FALSE,var.equal=FALSE,...) {
  if(!paired) {
    sdx <- sd(x)
```

```

sdy <- sd(y)
sd.big <- max(c(sdx,sdy) )
sd.small <- min(c(sdx,sdy) )
var.equal <- (sd.big/sd.small)<2
}
return(t.test(x=x,y=y,paired=paired,var.equal=var.equal,...))
}
> #(f)

> # Example 1

snacks <-
c(87.7,80.01,77.28,78.76,81.52,74.2,80.71,79.5,77.87,81.94,80.7,82.32,
75.78,80.19,83.91,79.4,77.52,77.62,81.4,74.89,82.95,73.59,77.92,77.18,
79.83,81.23,79.28,78.44,79.01,80.47,76.23,78.89,77.14,69.94,78.54,79.7,
82.45,77.29,75.52,77.21,75.99,81.94,80.41,77.7)
snacks2 <-
c(80.22,79.73,81.1,78.76,82.03,81.66,80.97,81.32,80.12,78.98,79.21,
81.48,79.86,81.06,77.96,80.73,80.34,80.01,81.82,79.3,79.08,79.47,
78.98,80.87,82.24,77.22,80.03,79.2,80.95,79.17,81)
myt.test(x=snacks2,y=snacks,alternative="greater",conf.level=0.9)

```

### Welch Two Sample t-test

data: x and y

t = 2.4455, df = 60.091, p-value = 0.008706

alternative hypothesis: true difference in means is greater than 0

90 percent confidence interval:

0.5859714 Inf

sample estimates:

mean of x mean of y

80.15710 78.91068

> # Example 2

> men <- c(102,87,101,96,107,101,91,85,108,67,85,82)

> women <- c(73,81,111,109,143,95,92,120,93,89,119,79,90,126,62,92,77,106,105,111)

> myt.test(x=men,y=women,alternative="two.sided",conf.level=0.95)

### Two Sample t-test

data: x and y

t = -0.93758, df = 30, p-value = 0.3559

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:

-19.016393 7.049727

sample estimates:

mean of x mean of y

92.66667 98.65000

> # Example 3

> rate.before <- c(52,66,89,87,89,72,66,65,49,62,70,52,75,63,65,61)

> rate.after <- c(51,66,71,73,70,68,60,51,40,57,65,53,64,56,60,59)

> myt.test(x=rate.after,y=rate.before,alternative="less",paired=TRUE,conf.level=0.95)

### Paired t-test

data: x and y

t = -4.8011, df = 15, p-value = 0.0001167

alternative hypothesis: true difference in means is less than 0

95 percent confidence interval:

-Inf -4.721833

sample estimates:

mean of the differences

-7.4375

## Exercise 18.3

An advertisement for a skin cream claims nine out of ten women who use it would recommend it to a friend. A skeptical salesperson in a department store believes the true proportion of women users who'd recommend it,  $\pi$ , is much smaller than 0.9. She follows up with 89 random customers who had purchased the skin cream and asks if they would recommend it to others, to which 71 answer yes.

- a. Set up an appropriate pair of hypotheses for this test and determine whether it will be valid to carry out using the normal distribution.
- b. Compute the test statistic and the p-value and state your conclusion for the test using a significance level of  $\alpha = 0.1$ .
- c. Using your estimated sample proportion, construct a two-sided 90 percent confidence interval for the true proportion of women who would recommend the skin cream.

The political leaders of a particular country are curious as to the proportion of citizens in two of its states that support the decriminalization of marijuana. A small pilot survey taken by officials reveals that 97 out of 445 randomly sampled voting-age citizens residing in state 1 support the decriminalization and that 90 out of 419 votingage citizens residing in state 2 support the same notion.

- d. Letting  $\pi_1$  denote the true proportion of citizens in support of decriminalization in state 1, and  $\pi_2$  the same measure in state 2, conduct and conclude a hypothesis test under a significance level of  $\alpha = 0.05$  with reference to the following hypotheses:

$$\begin{aligned} H_0 : \pi_2 - \pi_1 &= 0 \\ H_A : \pi_2 - \pi_1 &\neq 0 \end{aligned}$$

- e. Compute and interpret a corresponding CI.

Though there is standard, ready-to-use R functionality for the *t*-test, at the time of this writing, there is no similar function for the Z-test (in other words, the normal-based test of proportions described here) except in contributed packages.

- f. Your task is to write a relatively simple R function, *Z.test*, that can perform a one- or two-sample Z-test, using the following guidelines:
  - The function should take the following arguments: *p1* and *n1* (no default) to pose as the estimated proportion and sample size; *p2* and *n2* (both defaulting to *NULL*) that contain the second sample proportion and sample size in the event of a two-sample test; *p0* (no default) as the null value; and *alternative* (default "two.sided") and *conf.level* (default 0.95), to be used in the same way as in *t.test*.
  - When conducting a two-sample test, it should be *p1* that is tested as being smaller or larger than *p2* when *alternative*= "less" or *alternative*= "greater", the same as in the use of *x* and *y* in *t.test*.
  - The function should perform a one-sample Z-test using *p1*, *n1*, and *p0* if either *p2* or *n2* (or both) is *NULL*.
  - The function should contain a check for the rule of thumb to ensure the validity of the normal distribution in both one- and two-sample settings. If this is violated, the function should still complete but should issue an appropriate warning message (see Section 12.1.1).
  - All that need be returned is a list containing the members *Z* (test statistic), *P* (appropriate p-value—this can be determined by *alternative*; for a two-sided test, determining whether *Z* is positive or not can help), and *CI* (two-sided CI with respect to *conf.level*).
- g. Replicate the two examples in the text of Sections 18.3.1 and 18.3.2 using *Z.test*; ensure you reach identical results.

h. Call  $Z.test(p1=0.11, n1=10, p0=0.1)$  to try your warning message in the one-sample setting.

### Solution 18.3

```
> #(a)  
> # H0: p=0.9; HA: p<0.9  
> n <- 89  
> n*0.9  
[1] 80.1  
> n*0.1  
[1] 8.9  
> # Both np and n(p-1) > 5 so OK to continue with normal distribution according to the rule-of-thumb.  
> #(b)  
> p.hat <- 71/n  
> p.hat  
[1] 0.7977528  
> Z <- (p.hat-0.9)/(sqrt(0.9*0.1/n))  
> Z  
[1] -3.215327  
> pnorm(Z)  
[1] 0.0006514802  
> prop.test(x=71, n=n, p=0.9, alternative="less", conf.level=0.9, correct=FALSE)
```

1-sample proportions test without continuity correction

```
data: 71 out of n, null probability 0.9  
X-squared = 10.338, df = 1, p-value = 0.0006515  
alternative hypothesis: true p is less than 0.9  
90 percent confidence interval:  
0.0000000 0.8466949  
sample estimates:
```

```
p  
0.7977528
```

> # p-value very small; less than 0.1. There is evidence to reject H0 and conclude the true proportion of women who would recommend in samples of size 89 is less than 0.9.

```
> #(c)  
> p.hat+c(-1,1)*qnorm(0.95)*sqrt(p.hat*(1-p.hat)/n)
```

```
[1] 0.7277190 0.8677866
```

```
> #(d)
```

```
> x1 <- 97
```

```
> n1 <- 445
```

```
> p.hat1 <- x1/n1
```

```
> p.hat1
```

```
[1] 0.2179775
```

```
> x2 <- 90
```

```
> n2 <- 419
```

```
> p.hat2 <- x2/n2
```

```
> p.hat2
```

```
[1] 0.2147971
```

```
> p.star <- (x1+x2)/(n1+n2)
```

```
> p.star
```

```
[1] 0.2164352
```

```
> Z <- (p.hat2-p.hat1)/sqrt(p.star*(1-p.star)*(1/n1+1/n2))
```

```
> Z
```

```
[1] -0.1134513
```

```
> 2*pnorm(Z)
```

```
[1] 0.9096728
```

> # p-value is very large; much greater than 0.05. No evidence to reject H0. Retain H0 and conclude there is no evidence to suggest the proportion of citizens who support decriminalization varies between the two states.

```
> #(e)
```

```
> (p.hat2-p.hat1)+c(-1,1)*qnorm(0.975)*sqrt(p.star*(1-p.star)*(1/n1+1/n2))
```

```
[1] -0.05812427 0.05176349
```

```
> # We are 95% confident that the true difference in the proportion of support between State 2 and State 1 lies somewhere between -0.058 and 0.051. CI includes zero; reflects the same result as the hypothesis (no evidence of a difference).
```

```
> #(f)
```

```
> Z.test <-  
function(p1,n1,p2=NULL,n2=NULL,p0,alternative="two.sided",conf.level=0.  
95){  
  if(is.null(p2) || is.null(n2)){  
    cat("One-sample Z-test.\n")  
    if(p1*n1<=5 || n1*(1-p1)<=5){  
      warning("Normal distribution may not be valid; np or n(1-p) <= 5  
detected.")  
    }  
    Z <- (p1-p0)/sqrt(p0*(1-p0)/n1)  
    CI <- (p1)+c(-1,1)*qnorm(conf.level+(1-conf.level)/2)*sqrt(p0*(1-  
p0)/n1)  
  } else {  
    cat("Two-sample Z-test.\n")  
    if(p1*n1<=5 || n1*(1-p1)<=5 || p2*n2<=5 || n2*(1-p2)<=5){  
      warning("Normal distribution may not be valid; np or n(1-p) <= 5  
detected.")  
    }  
    p.star <- (p1*n1+p2*n2)/(n1+n2)  
    Z <- (p1-p2-p0)/sqrt(p.star*(1-p.star)*(1/n1+1/n2))  
    CI <- sort((p1-p2)+c(-1,1)*qnorm(conf.level+(1-  
conf.level)/2)*sqrt(p.star*(1-p.star)*(1/n1+1/n2)))  
  }  
  
  P <- pnorm(Z)  
  if(alternative=="greater"){  
    P <- 1-P  
  } else if(alternative=="two.sided"){  
    if(Z<0){  
      P <- 2*P  
    } else {  
      P <- 2*(1-P)  
    }  
  }  
  return(list(Z=Z,P=P,CI=CI))  
}  
> #(g)
```

```
> # Example 1
```

```
> sick <- c(0,0,1,1,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,1,1,1,0,0,0,1)
```

```
> Z.test(p1=mean(sick),n1=length(sick),p0=0.2,alternative="two.sided",conf.level=0.95)
```

```
One-sample Z-test.
```

```
$Z
```

```
[1] 1.021324
```

```
$P  
[1] 0.3071008
```

```
$CI  
[1] 0.1302796 0.4214445
```

```
> # Example 2  
> x1 <- 180  
> n1 <- 233  
> p.hat1 <- x1/n1  
> x2 <- 175  
> n2 <- 197  
> p.hat2 <- x2/n2  
> Z.test(p.hat2,n2,p.hat1,n1,p0=0,alternative="greater",conf.level=0.95) # ...or you could flip the  
order of differencing and use alternative="less"
```

Two-sample Z-test.

```
$Z  
[1] 3.152693
```

```
$P  
[1] 0.0008088606  
  
$CI  
[1] 0.04380676 0.18777861
```

```
> #(h)  
> Z.test(p1=0.11,n1=10,p0=0.1)
```

One-sample Z-test.

```
$Z  
[1] 0.1054093
```

\$P

[1] 0.9160511

\$CI

[1] -0.07593851 0.29593851

Warning:

B Z.test(p1 = 0.11, n1 = 10, p0 = 0.1) :

Normal distribution may not be valid; np or n(1-p) <= 5 detected.

## Exercise 18.4

*HairEyeColor* is a ready-to-use data set in R that you haven't yet come across. This  $4 \times 4 \times 2$  array provides frequencies of hair and eye colors of 592 statistics students, split by sex (Snee, 1974).

- a. Perform and interpret, at a significance level of  $\alpha = 0.01$ , a chisquared test of independence for hair against eye color for all students, regardless of their sex.

In Exercise 8.1 on page 161, you accessed the *Duncan* data set of the contributed package *car*, which contains markers of job prestige collected in 1950. Install the package if you haven't already and load the data frame.

- b. The first column of *Duncan* is the variable *type*, recording the type of job as a factor with three levels: *prof* (professional or managerial), *bc* (blue collar), and *wc* (white collar). Construct appropriate hypotheses and perform a chi-squared GOF test to determine whether the three job types are equally represented in the data set.
  - i. Interpret the resulting p-value with respect to a significance level of  $\alpha = 0.05$ .
  - ii. What conclusion would you reach if you used a significance level of  $\alpha = 0.01$ ?

## Solution 18.4

```
> #(a) H0: No relationship between hair and eye color; HA: There is a relationship.
```

```
> ?HairEyeColor
```

```
> chisq.test(x=HairEyeColor[,1]+HairEyeColor[,2])
```

Pearson's Chi-squared test

```
data: HairEyeColor[, , 1] + HairEyeColor[, , 2]
```

```
X-squared = 138.29, df = 9, p-value < 2.2e-16
```

```
> # Very small P-value. Very strong evidence against the null. Reject H0 and conclude there does appear to be a relationship between hair and eye color of statistics students.
```

```
> #(b) H0: p1=p2=p3=1/3; HA: H0 is incorrect
```

```
> library("car")
```

```
> data(Duncan)
```

```
> ?Duncan
```

```
> jobtype <- Duncan$type
```

```
> jobtype.tab <- table(jobtype)
```

```
> jobtype.tab
```

jobtype

  bc prof wc

  21 18 6

```
> chisq.test(jobtype.tab)
```

## Chi-squared test for given probabilities

data: jobtype.tab

X-squared = 8.4, df = 2, p-value = 0.015

> ##(i) With a significance level of 0.05 and a p-value of 0.015, there is weak evidence to reject H0 and we therefore conclude that the three job types do not appear to be uniformly represented in the data set.

> ##(ii) With a significance level of 0.01 and a P-value of 0.015, there is no evidence to reject H0 and we therefore conclude that the three job types appear to be uniformly represented in the data set.

>

## Exercise 18.5

- a. Write a new version of *typeI.tester* called *typeI.mean*. The new function should be able to simulate the Type I error rate for tests of a single mean in any direction (in other words, one or two-sided). The new function should take an additional argument, *test*, which takes a character string "less", "greater", or "two.sided" depending on the type of desired test. You can achieve this by modifying *typeI.tester* as follows:
    - Instead of calculating and storing the p-values directly in the *for* loop, simply store the test statistic.
    - When the loop is complete, set up stacked *if-else* statements that cater to each of the three types of test, calculating the p-value as appropriate.
    - For the two-sided test, remember that the p-value is defined as twice the area “more extreme” than the null. Computationally, this means you must use the upper-tail area if the test statistic is positive and the lower-tail area otherwise.If this area is less than half of  $\alpha$  (since it is subsequently multiplied by 2 in a “real” hypothesis test), then a rejection of the null should be flagged.
    - If the value of *test* is not one of the three possibilities, the function should throw an appropriate error using *stop*.
  - i. Experiment with your function using the first example setting in the text with  $\mu_0 = 0$ ,  $\sigma = 1$ ,  $n = 40$ , and  $\alpha = 0.05$ . Call *typeI.mean* three times, using each of the three possible options for *test*. You should find that all simulated results sit close to 0.05.
  - ii. Repeat (i) using the second example setting in the text with  $\mu_0 = -4$ ,  $\sigma = 0.3$ ,  $n = 60$ , and  $\alpha = 0.01$ . Again, you should find that all simulated results sit close to the value of  $\alpha$ .
- b. Modify *typeII.tester* in the same way as you did *typeI.tester*; call the new function *typeII.mean*. Simulate the Type II error rates for the following hypothesis tests. As per the text, assume  $\mu_A$ ,  $\sigma$ ,  $\alpha$ , and  $n$  denote the true mean, standard deviation of raw observations, significance level, and sample size, respectively.
    - i.  $H_0 : \mu = -3.2$ ;  $H_A : \mu > -3.2$  with  $\mu_A = -3.3$ ,  $\sigma = 0.1$ ,  $\alpha = 0.05$ , and  $n = 25$ .
    - ii.  $H_0 : \mu = 8994$ ;  $H_A : \mu < 8994$  with  $\mu_A = 5600$ ,  $\sigma = 3888$ ,  $\alpha = 0.01$ , and  $n = 9$ .
    - iii.  $H_0 : \mu = 0.44$ ;  $H_A : \mu > 0.44$  with  $\mu_A = 0.4$ ,  $\sigma = 2.4$ ,  $\alpha = 0.05$ , and  $n = 68$ .

## Solution 18.5

```
> #(a)

>typeI.mean <-
function(mu0,sigma,n,alpha,test="two.sided",ITERATIONS=10000) {
  tstats <- rep(NA,ITERATIONS)
  for(i in 1:ITERATIONS) {
    temporary.sample <- rnorm(n=n,mean=mu0,sd=sigma)
    temporary.mean <- mean(temporary.sample)
    temporary.sd <- sd(temporary.sample)
    tstats[i] <- (temporary.mean-mu0)/(temporary.sd/sqrt(n))
  }
  pvals <- pt(tstats,df=n-1)
  if(test=="less"){
    return(mean(pvals<alpha))
  } else if(test=="greater"){
    return(mean((1-pvals)<alpha))
  } else if(test=="two.sided"){
    result <- pvals
    result[tstats>0] <- 1-pvals[tstats>0]
    return(mean(result<alpha/2))
  } else {
    stop("'test' argument not recognised")
  }
}
>##(i)

> typeI.mean(mu0=0,sigma=1,n=40,alpha=0.05,test="less")

[1] 0.0515

> typeI.mean(mu0=0,sigma=1,n=40,alpha=0.05,test="greater")

[1] 0.0462

> typeI.mean(mu0=0,sigma=1,n=40,alpha=0.05,test="two.sided")

[1] 0.0514

>##(ii)

> typeI.mean(mu0=-4,sigma=0.3,n=60,alpha=0.01,test="less")

[1] 0.0102

> typeI.mean(mu0=-4,sigma=0.3,n=60,alpha=0.01,test="greater")

[1] 0.0126

> typeI.mean(mu0=-4,sigma=0.3,n=60,alpha=0.01,test="two.sided")

[1] 0.0092
```

```

> #(b)

>typeII.mean <-
function(mu0,muA,sigma,n,alpha,test="two.sided",ITERATIONS=10000) {
  tstats <- rep(NA,ITERATIONS)
  for(i in 1:ITERATIONS) {
    temporary.sample <- rnorm(n=n,mean=muA,sd=sigma)
    temporary.mean <- mean(temporary.sample)
    temporary.sd <- sd(temporary.sample)
    tstats[i] <- (temporary.mean-mu0)/(temporary.sd/sqrt(n))
  }
  pvals <- pt(tstats,df=n-1)
  if(test=="less") {
    return(mean(pvals>=alpha))
  } else if(test=="greater") {
    return(mean((1-pvals)>=alpha))
  } else if(test=="two.sided") {
    result <- pvals
    result[tstats>0] <- 1-pvals[tstats>0]
    return(mean(result>=alpha/2))
  } else {
    stop("'test' argument not recognised")
  }
}
>##(i)

> typeII.mean(mu0=-3.2,muA=-3.3,sigma=0.1,n=25,alpha=0.05,test="two.sided")
[1] 0.0025

>##(ii)

> typeII.mean(mu0=8994,muA=5600,sigma=3888,n=9,alpha=0.01,test="less")
[1] 0.5557

>##(iii)

> typeII.mean(mu0=0.44,muA=0.4,sigma=2.4,n=68,alpha=0.05,test="greater")
[1] 0.9626

>

```

## Exercise 18.6

- a. For this exercise you'll need to have written *typeII.mean* from Exercise 18.5 (b). Using this function, modify *power.tester* so that a new function, *power.mean*, calls *typeII.mean* instead of calling *typeII.tester*.
- i. Confirm that the power of the test given by  $H_0 : \mu = 10$ ;
  - ii.  $H_A : \mu > 10$ , with  $\mu_A = 10.5$ ,  $\sigma = 0.9$ ,  $\alpha = 0.01$ , and  $n = 50$ , is roughly 88 percent.
  - iii. Remember the hypothesis test in Section 18.2.1 for the mean net weight of an 80-gram pack of snacks, based on the  $n = 44$  observations provided in the *snack* vector. The hypotheses were as follows:

$$\begin{aligned} H_0 : \mu &= 80 \\ H_A : \mu &< 80 \end{aligned}$$

If the true mean is  $\mu_A = 78.5$  g and the true standard deviation of the weights is  $\sigma = 3.1$  g, use *power.mean* to determine whether the test is statistically powerful, assuming  $\alpha = 0.05$ . Does your answer to this change if  $\alpha = 0.01$ ?

- b. Staying with the snacks hypothesis test, using the *sample.sizes* vector from the text, determine the minimum sample size required for a statistically powerful test using both  $\alpha = 0.05$  and  $\alpha = 0.01$ . Produce a plot showing the two power curves.

## Solution 18.6

```
> #(a)

> power.mean <- function(nvec, ...) {
  nlen <- length(nvec)
  result <- rep(NA, nlen)
  pbar <- txtProgressBar(min=0, max=nlen, style=3)
  for(i in 1:nlen) {
    result[i] <- 1-typeII.mean(n=nvec[i], ...)
    setTxtProgressBar(pbar, i)
  }
  close(pbar)
  return(result)
}
> ##(i)

> power.mean(nvec=50, mu0=10, muA=10.5, sigma=0.9, alpha=0.01, test="two.sided")
```

[1] 0.8881

```
> ##(ii)
```

```
> power.mean(nvec=44, mu0=80, muA=78.5, sigma=3.1, alpha=0.05, test="less") #Yes, seems
statistically powerful.
```

[1] 0.9397

```
> power.mean(nvec=44, mu0=80, muA=78.5, sigma=3.1, alpha=0.01, test="less") #No, power
appears less than 80%, but only just...
```

[1] 0.779

```
> #(b)

> sample.sizes <- 5:100

> pow <- power.mean(nvec=sample.sizes,mu0=80,muA=78.5,sigma=3.1,alpha=0.05,test="less")

> minimum.n <- sample.sizes[min(which(pow>=0.8))]

> minimum.n

[1] 28

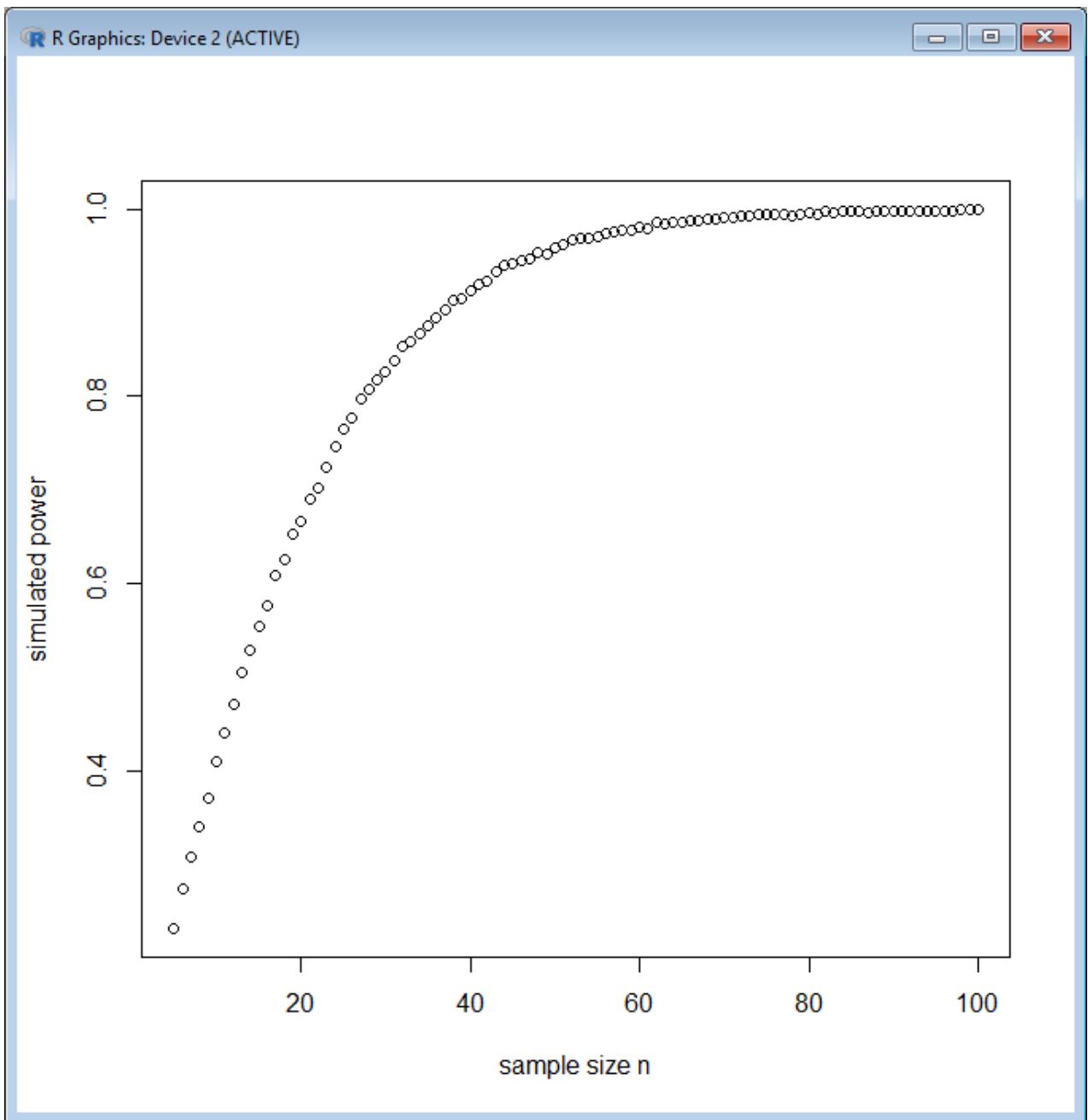
> pow2 <- power.mean(nvec=sample.sizes,mu0=80,muA=78.5,sigma=3.1,alpha=0.01,test="less")

> minimum.n2 <- sample.sizes[min(which(pow2>=0.8))]

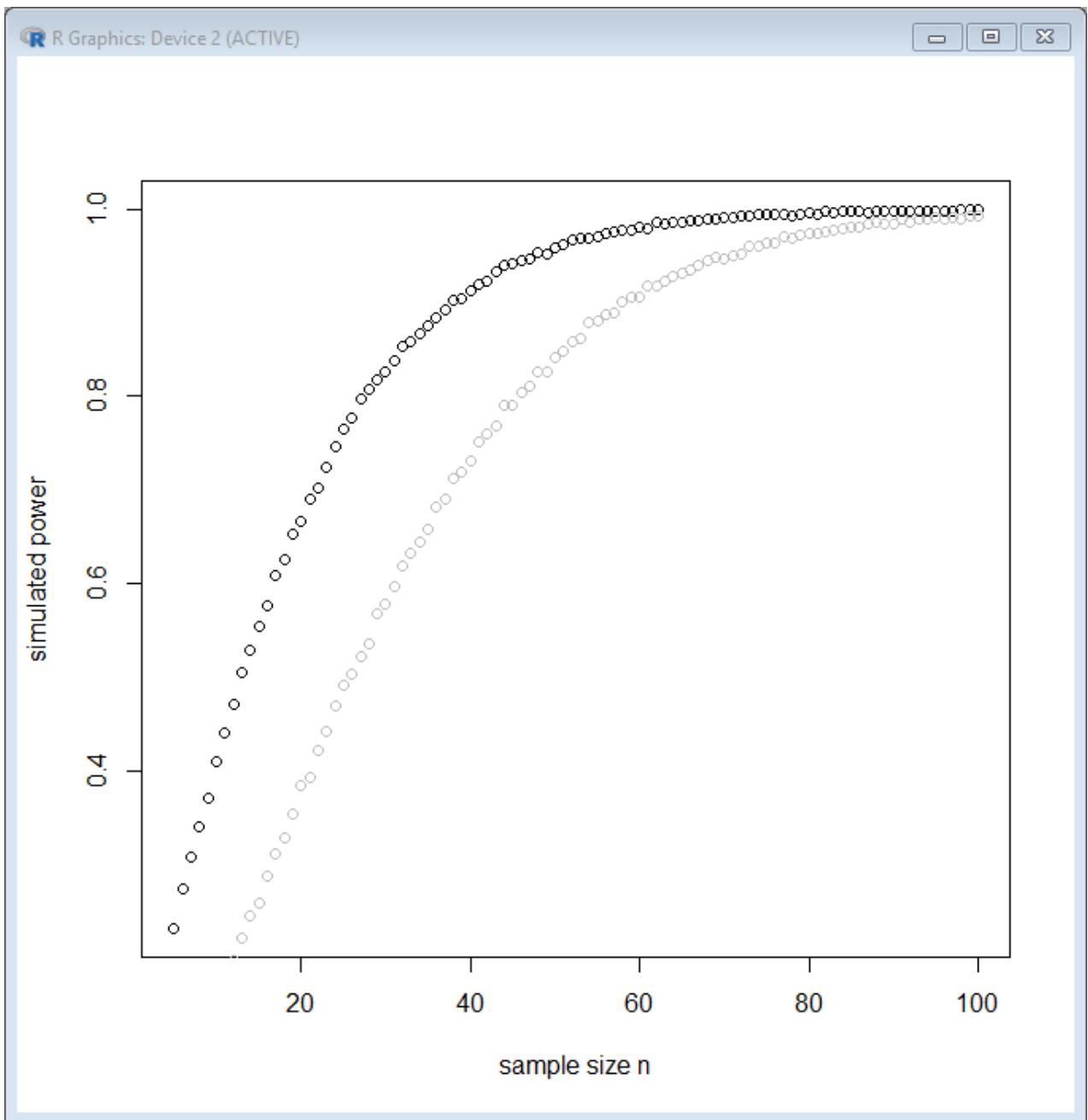
> minimum.n2

[1] 46

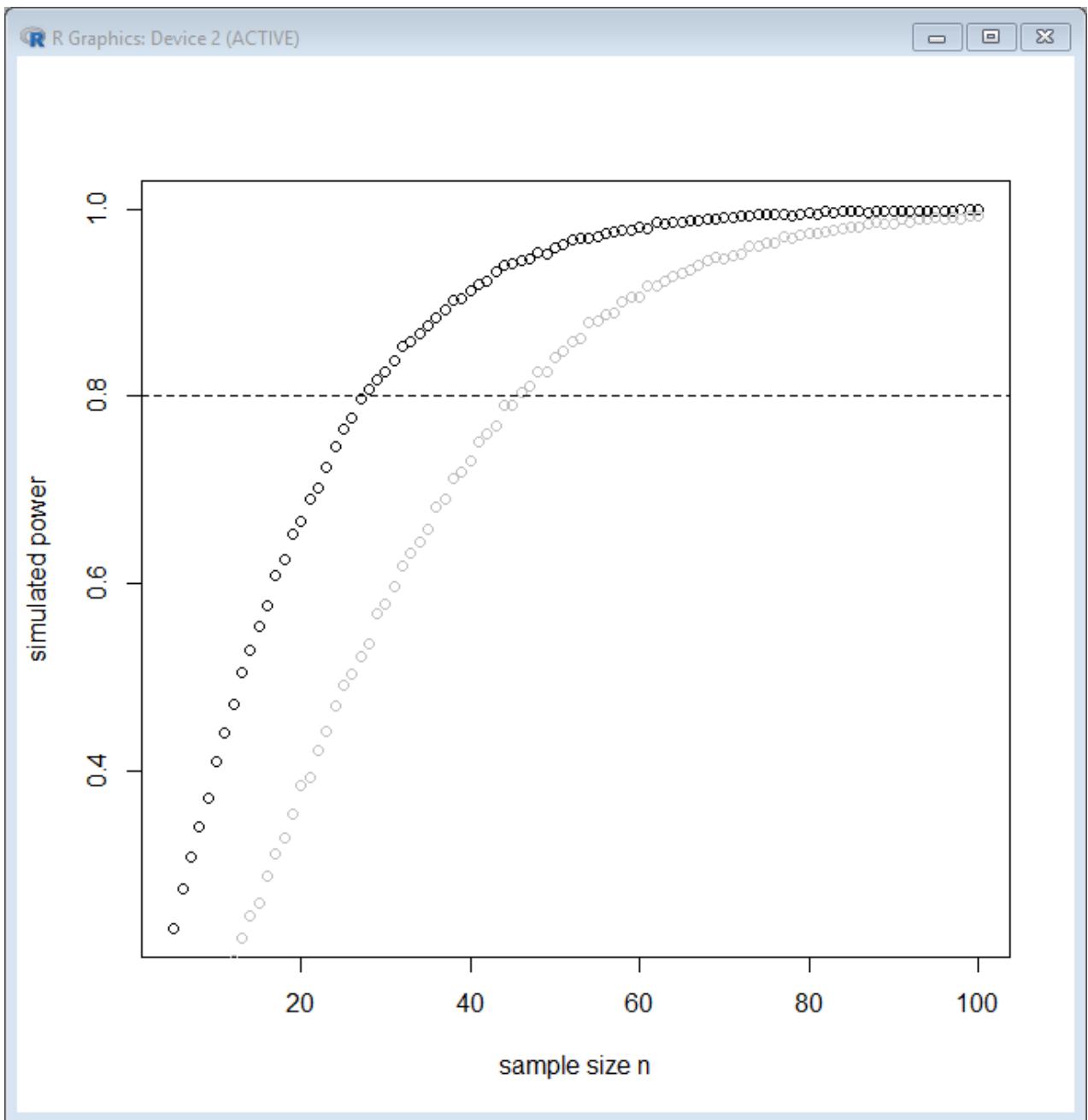
> plot(sample.sizes,pow,xlab="sample size n",ylab="simulated power")
```



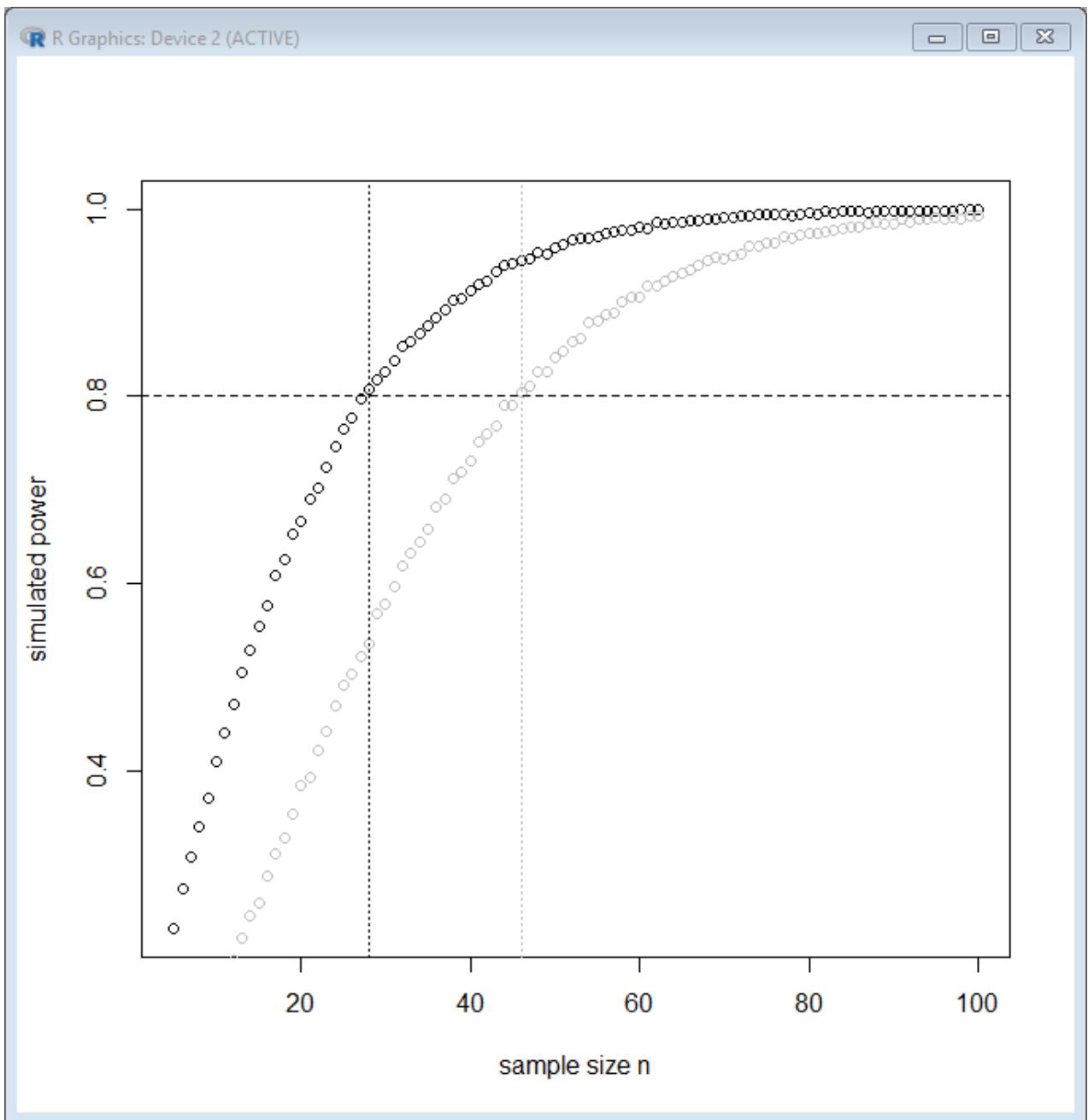
```
> points(sample.sizes,pow2,col="grey")
```



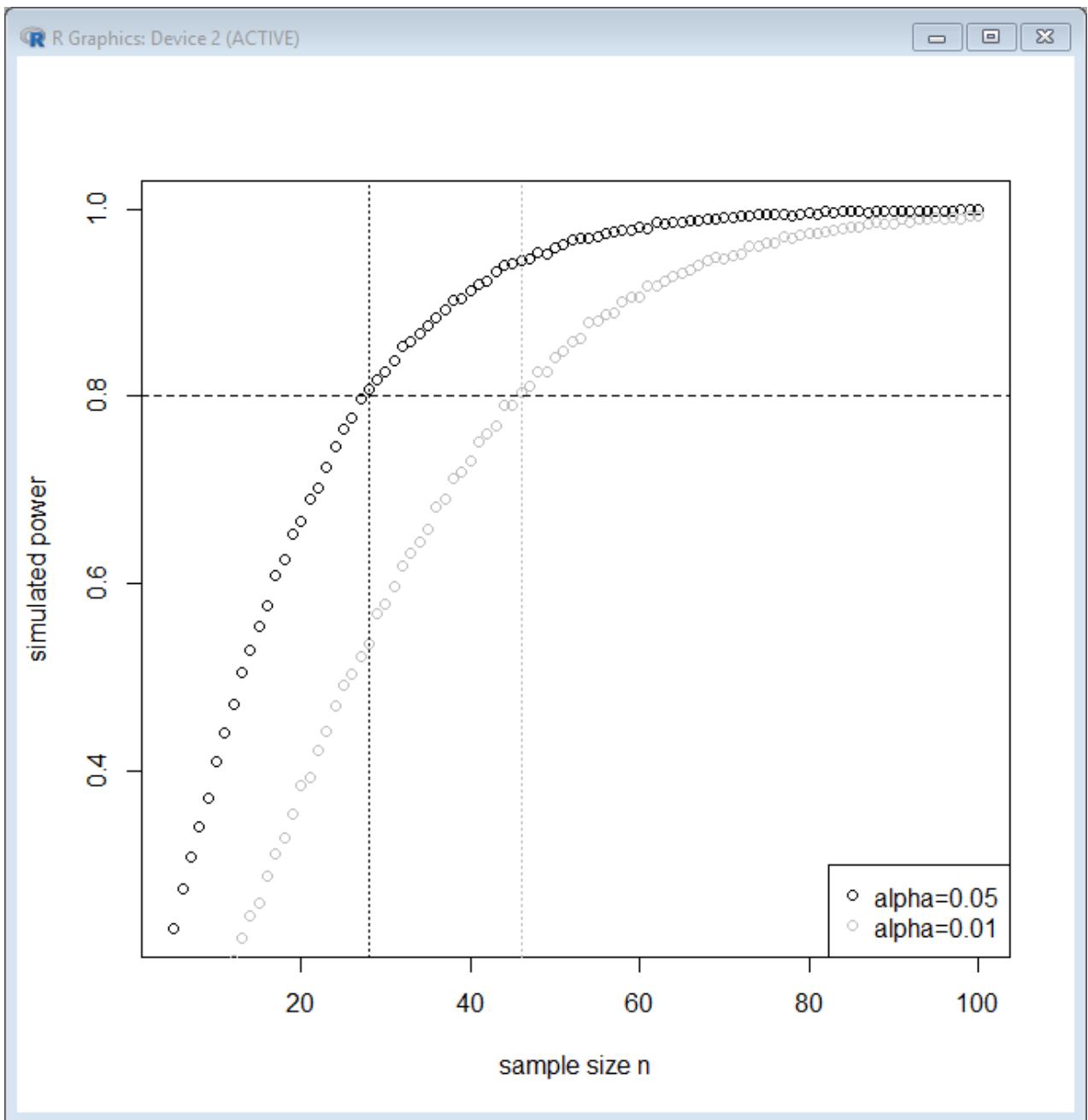
```
> abline(h=0.8,lty=2)
```



```
> abline(v=c(minimum.n,minimum.n2),lty=3,col=c("black","grey"))
```



```
> legend("bottomright",legend=c("alpha=0.05","alpha=0.01"),col=c("black","grey"),pch=1)
```



# Chapter 19: Analysis of Variance

Estimated time to complete: **30 minutes**

## Exercise 19.1

Consider the following data:

Site I	Site II	Site III	Site IV
93	85	100	96
120	45	75	58
65	80	65	95
105	28	40	90
115	75	73	65
82	70	65	80
99	65	50	85
87	55	30	95
100	50	45	82
90	40	50	
78		45	
95		55	
93			
88			
110			

These figures provide the depths (in centimeters) at which important archaeological finds were made at four sites in New Mexico (see Woosley and McIntyre, 1996). Store these data in your R workspace, with one vector containing depth and the other vector containing the site of each observation.

- Produce side-by-side boxplots of the depths split by group, and use additional points to mark the locations of the sample means.
- Assuming independence, execute diagnostic checks for normality and equality of variances.
- Perform and conclude a one-way ANOVA test for evidence of a difference between the means.

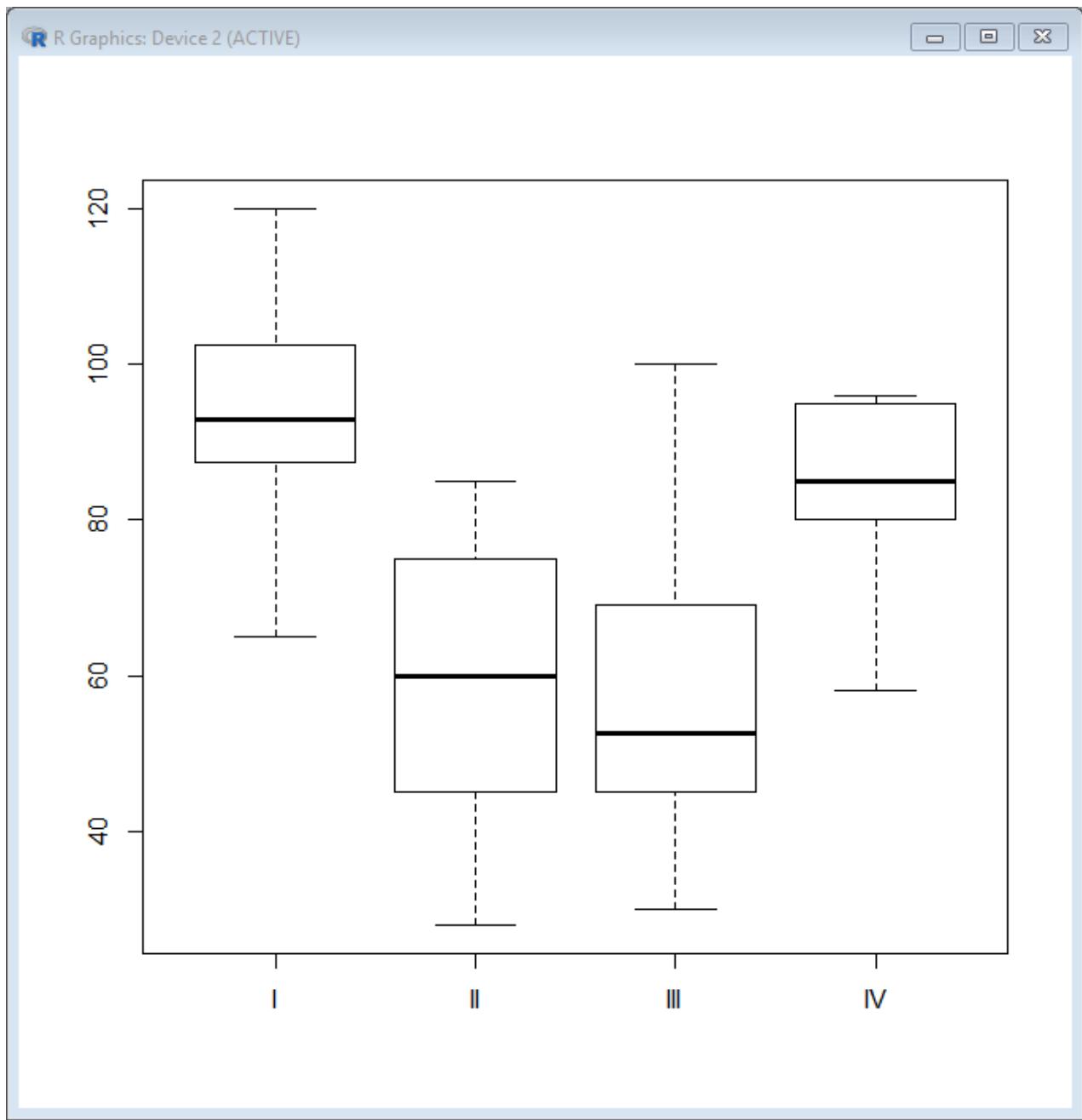
In Section 14.4, you looked at the data set providing measurements on petal and sepal sizes for three species of iris flowers. This is available in R as *iris*.

- Based on diagnostic checks for normality and equality of variances, decide which of the four outcome measurements (sepal length/width and petal length/width) would be suitable for ANOVA (using the species as the group variable).
- Carry out one-way ANOVA for any suitable measurement variables.

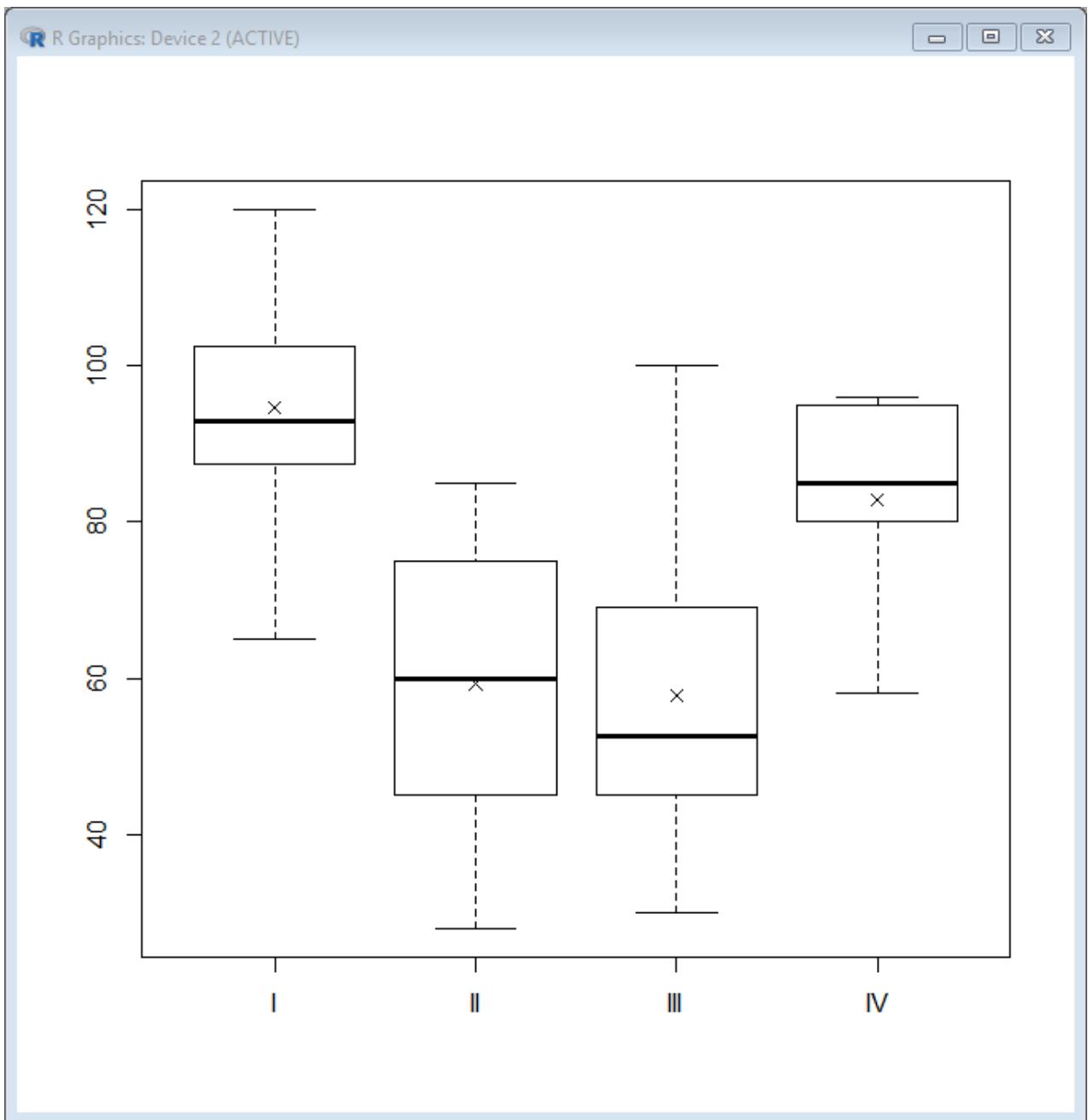
## Solution 19.1

```
> mim <-  
c(93,120,65,105,115,82,99,87,100,90,78,95,93,88,110,85,45,80,28,75,70,65,55,50,40,100,75,65,4  
0,73,65,50,30,45,50,45,55,96,58,95,90,65,80,85,95,82)  
  
> site <- c(rep("I",15),rep("II",10),rep("III",12),rep("IV",9))
```

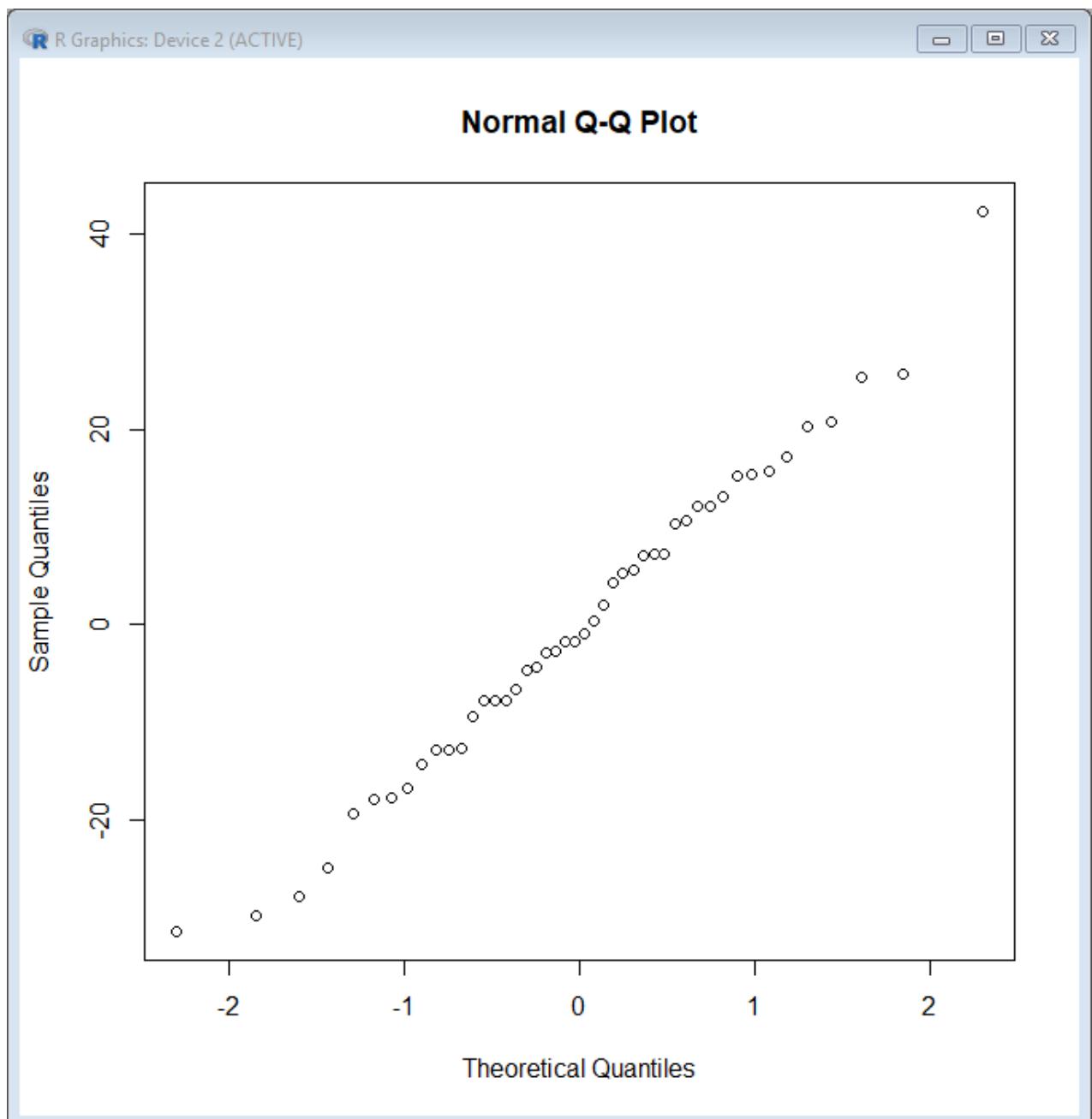
```
> #(a)  
> boxplot(mim~site)
```



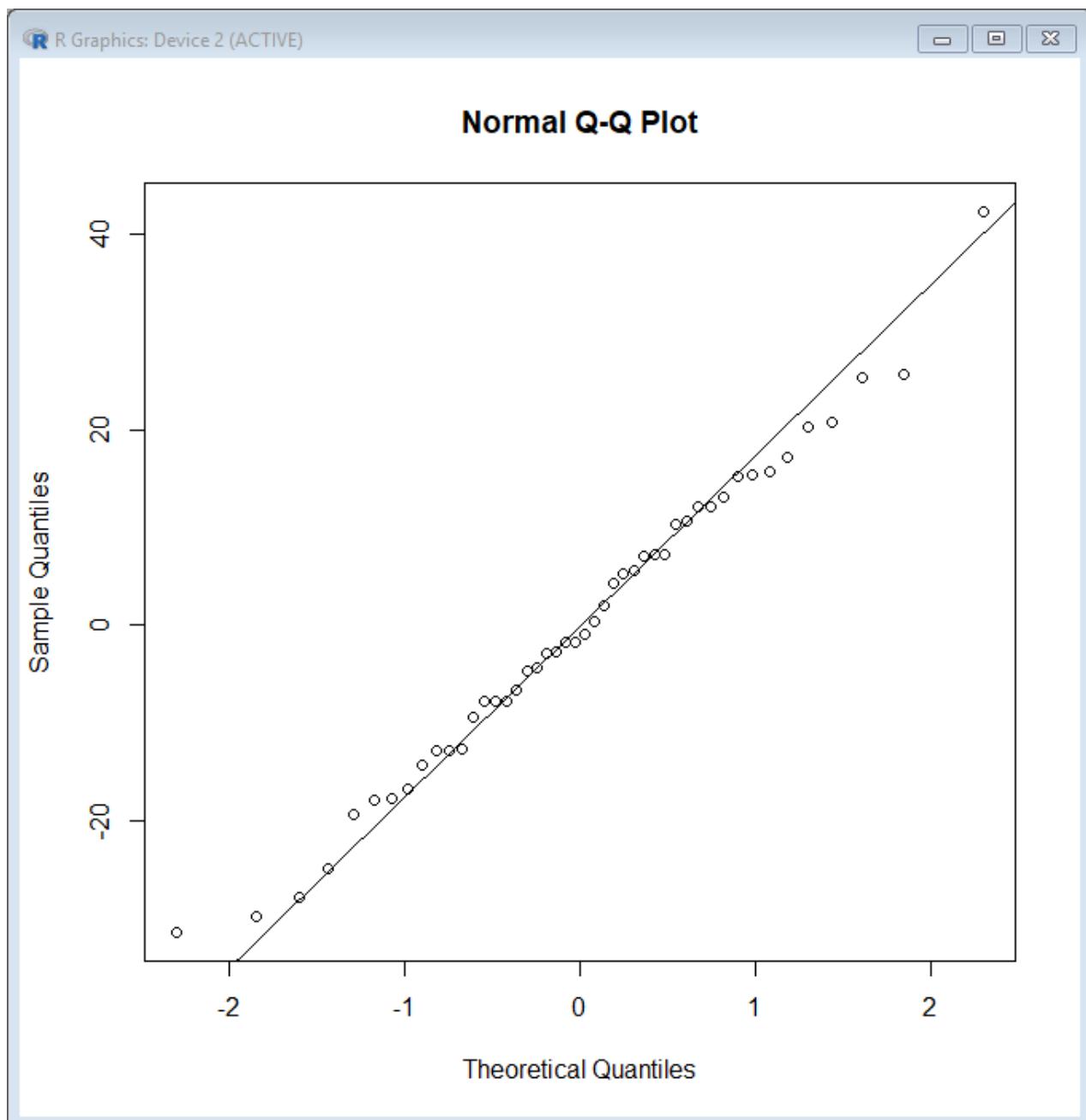
```
> mim.means <- tapply(mim,INDEX=site,FUN=mean)  
> points(1:4,mim.means,pch=4)
```



```
> #(b)  
> mim.meancen <- mim.rep(mim.means,table(site))  
> qqnorm(mim.meancen)
```



```
> qqline(mim.meancen) # Seems consistent with normality based on a visual inspection.
```



```
> sds <- tapply(mim, INDEX=site, FUN=sd)
> max(sds)/min(sds) # Less than 2 so variances can be assumed equal according to the rule-of-thumb.
```

```
[1] 1.399928
```

```
> #(c)
```

```
> summary(aov(mim~site)) # Small p-value. Very strong evidence against H0. There is evidence to reject the null and conclude there is a difference in the mean depths of the finds across the four sites.
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
site	3	12397	4132	15.14	7.99e-07 ***

```
Residuals 42 11465 273
```

```
---
```

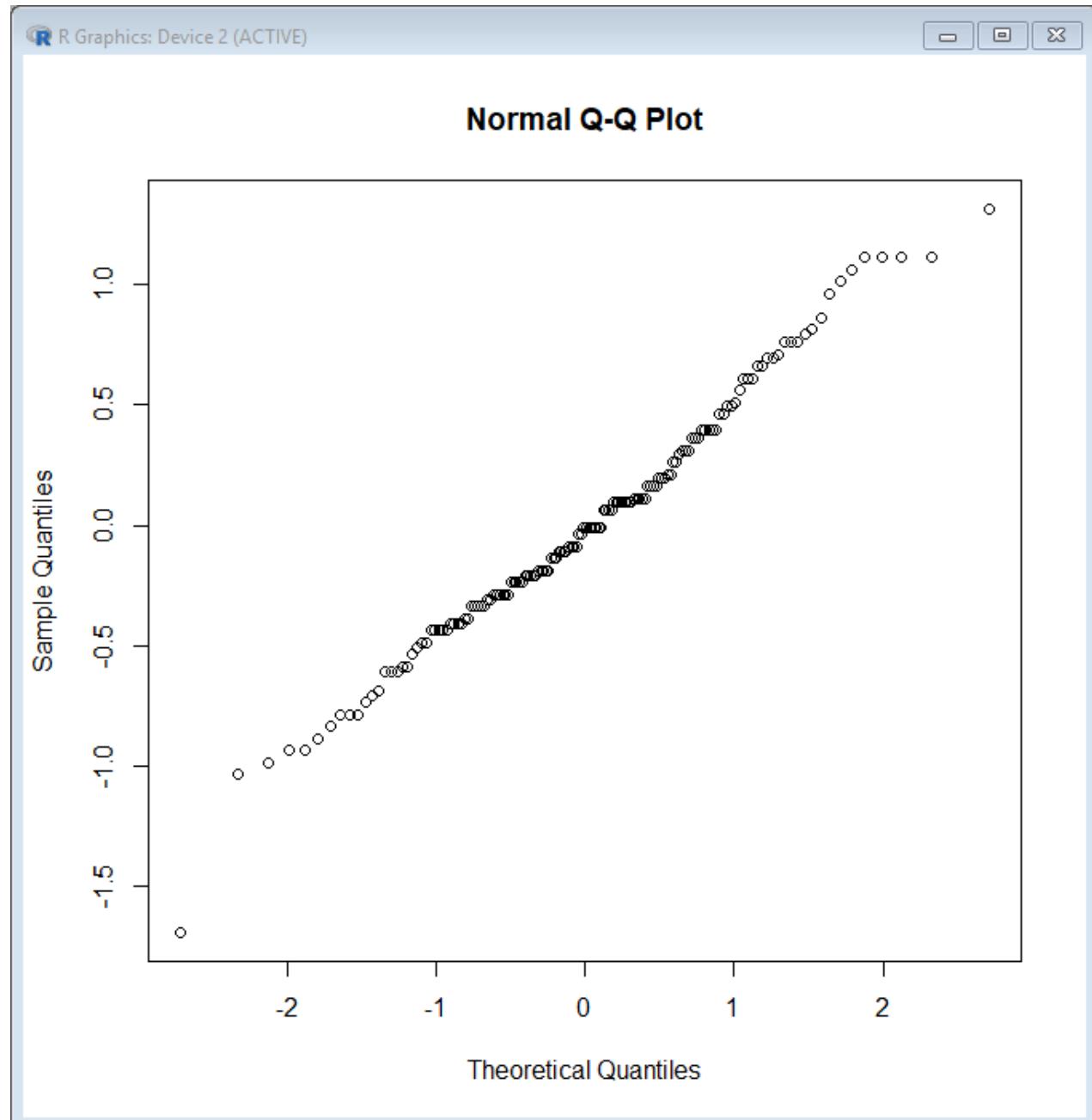
```
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
> #(d)
```

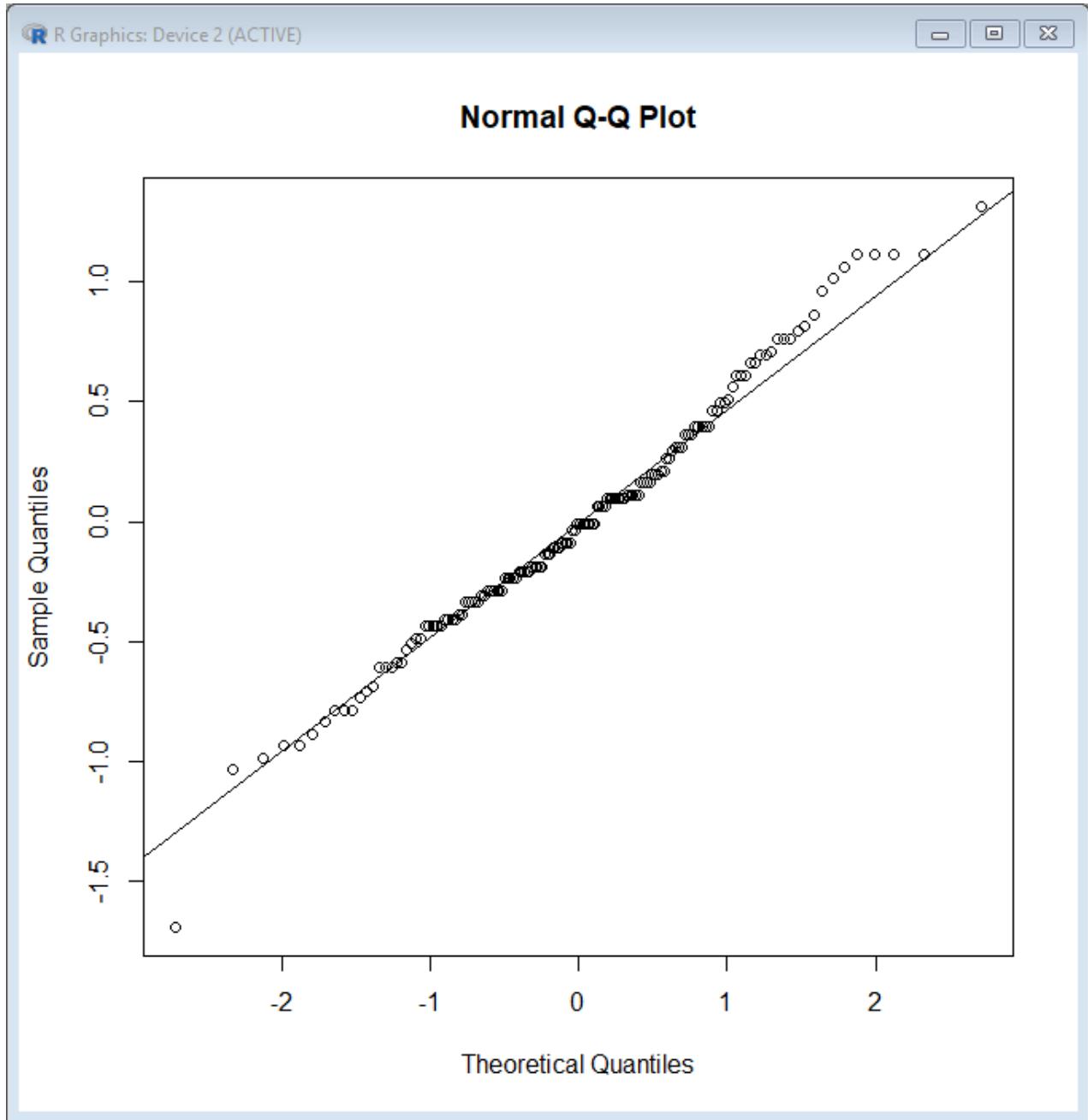
```
> m <- tapply(iris$Sepal.Length,iris$Species,mean)
```

```
> mc <- iris$Sepal.Length-m[as.numeric(iris$Species)]
```

```
> qqnorm(mc)
```



```
> qqline(mc) # Looks approximately normal.
```

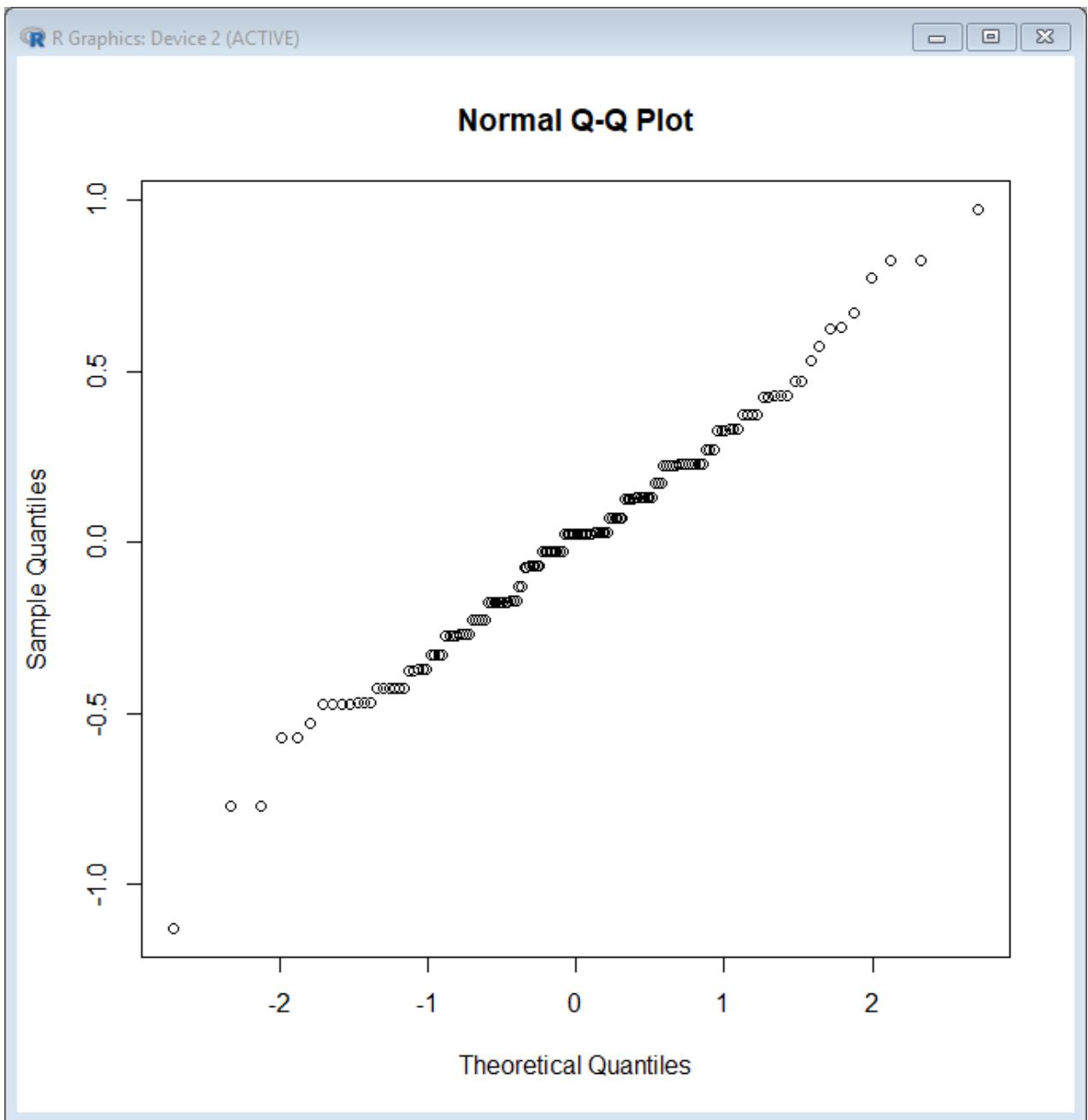


```
> tapply(iris$Sepal.Length,iris$Species,sd) # max(sd)/min(sd) < 2 so variances may be assumed equal.
```

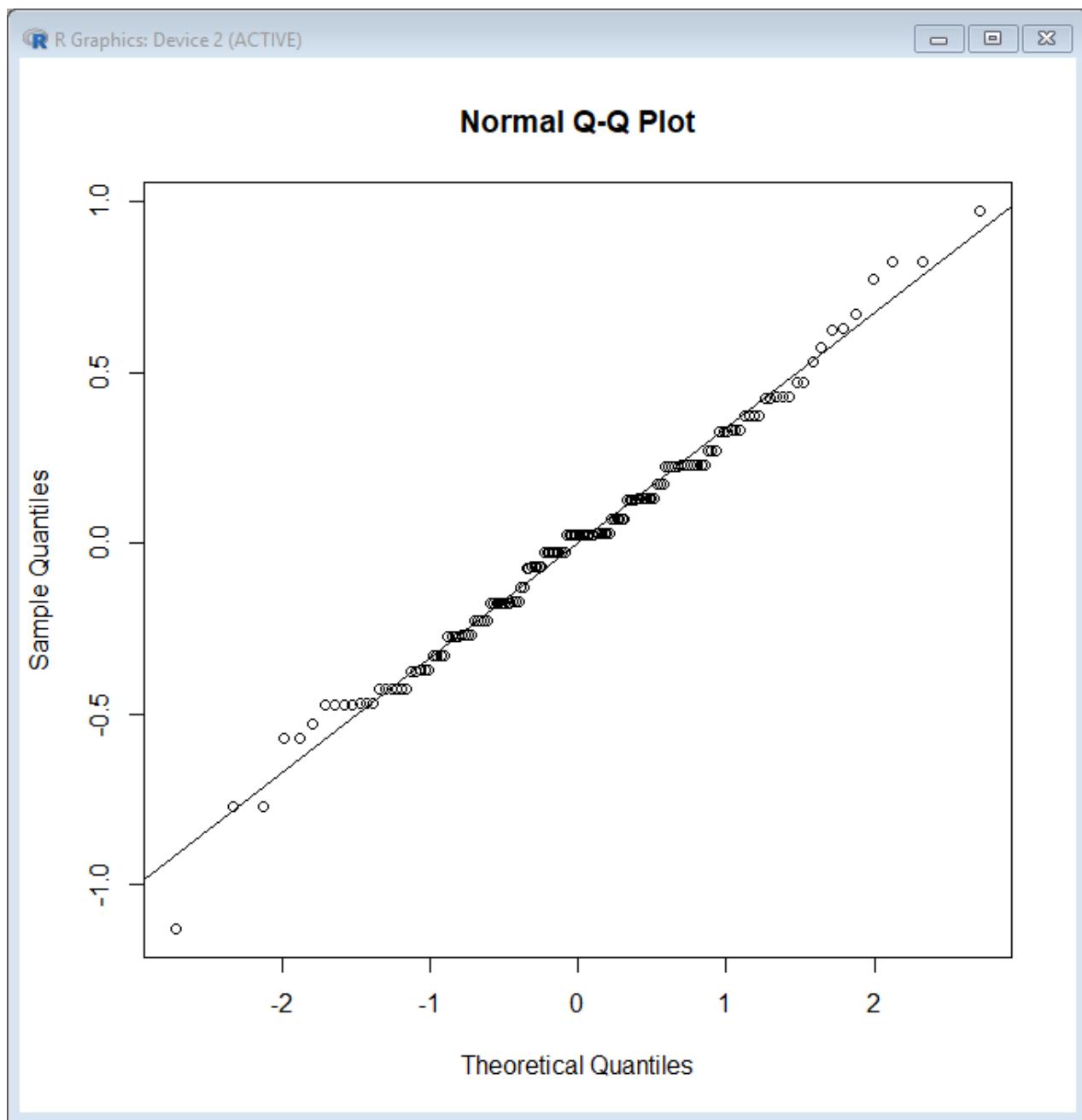
```
setosa versicolor virginica
```

```
0.3524897 0.5161711 0.6358796
```

```
> m <- tapply(iris$Sepal.Width,iris$Species,mean)  
> mc <- iris$Sepal.Width-m[as.numeric(iris$Species)]  
> qqnorm(mc)
```



```
> qqline(mc) # Looks approximately normal.
```

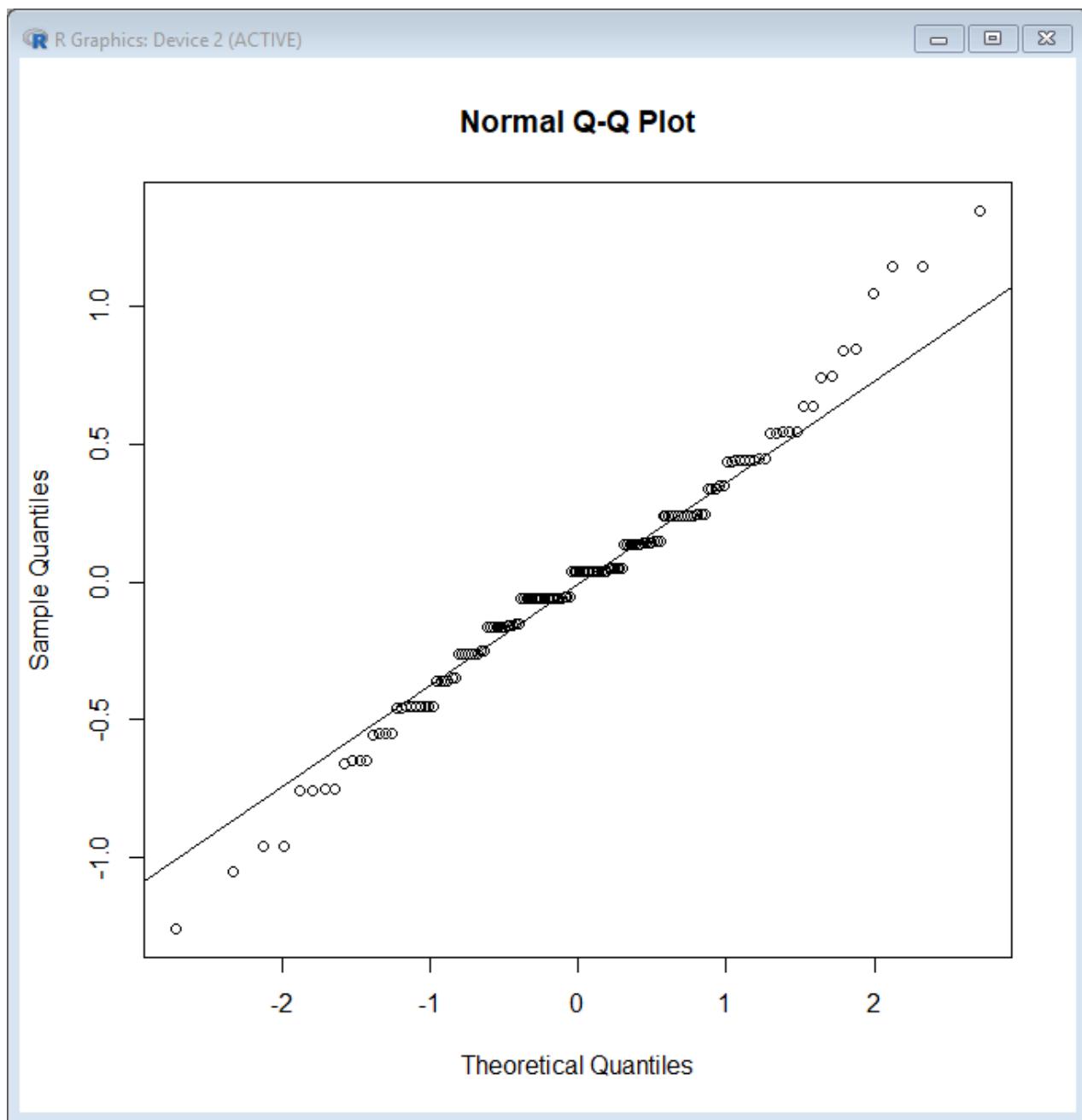


```
> tapply(iris$Sepal.Width,iris$Species,FUN=sd) # max(sd)/min(sd) < 2 so variances may be assumed equal.
```

```
setosa versicolor virginica
```

```
0.3790644 0.3137983 0.3224966
```

```
> m <- tapply(iris$Petal.Length,iris$Species,mean)
> mc <- iris$Petal.Length-m[as.numeric(iris$Species)]
> qqnorm(mc)
> qqline(mc) # Looks approximately normal; some deviation though.
```

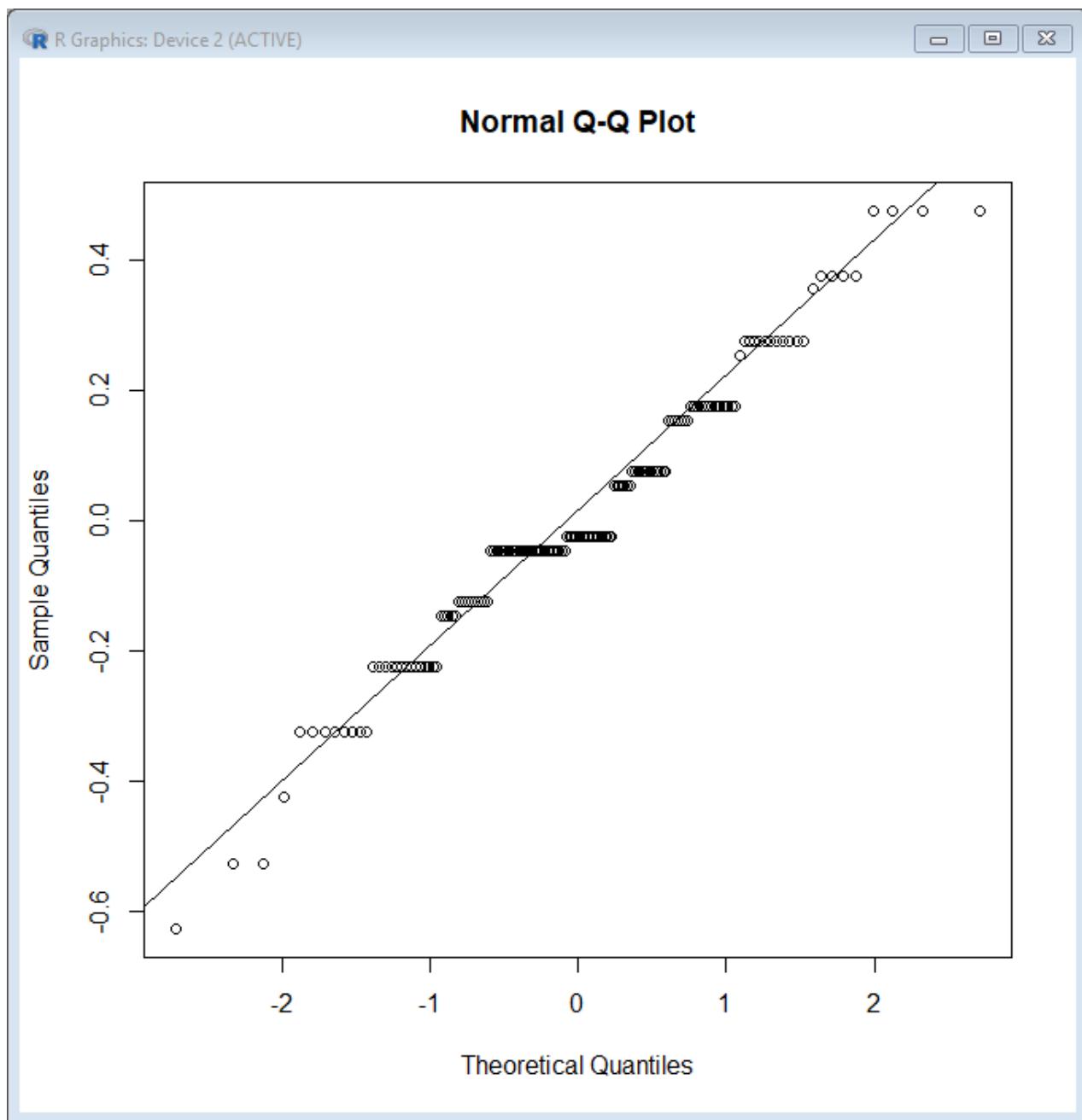


```
> tapply(iris$Petal.Length,iris$Species,sd) # max(sd)/min(sd) > 2 so variances may not be assumed equal.
```

```
setosa versicolor virginica
```

```
0.1736640 0.4699110 0.5518947
```

```
> m <- tapply(iris$Petal.Width,iris$Species,mean)
> mc <- iris$Petal.Width-m[as.numeric(iris$Species)]
> qqnorm(mc)
> qqline(mc) # Looks approximately normal.
```



```
> tapply(iris$Petal.Width,iris$Species,FUN=sd) # max(sd)/min(sd) > 2 so variances may not be assumed equal.
```

```
setosa versicolor virginica
0.1053856 0.1977527 0.2746501
> #(e)
> summary(aov(Sepal.Length~Species,data=iris))

Df Sum Sq Mean Sq F value Pr(>F)
Species     2  63.21  31.606  119.3 <2e-16 ***
Residuals 147  38.96  0.265
---
```

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> summary(aov(Sepal.Width~Species,data=iris))

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Species	2	11.35	5.672	49.16 <2e-16	***
Residuals	147	16.96	0.115		

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> # Both p-values are very small. Strong evidence to reject H0 and conclude that the mean sepal lengths and widths do vary according to species.

## Exercise 19.2

Bring up the *quakes* data frame again, which describes the locations, magnitudes, depths, and number of observation stations that detected 1,000 seismic events off the coast of Fiji.

- a. Use *cut* (see Section 4.3.3) to create a new factor vector defining the depths of each event according to the following three categories: (0,200], (200,400], and (400,680].
- b. Decide whether a one-way ANOVA or a Kruskal-Wallis test is more appropriate to use to compare the distributions of the number of detecting stations, split according to the three categories in (a).
- c. Perform your choice of test in (b) (assume a  $\alpha = 0.01$  level of significance) and conclude.  
Load the *MASS* package with a call to *library("MASS")* if you haven't already done so in the current R session. This package includes the ready-to-use *Cars93* data frame, which contains detailed data on 93 cars for sale in the United States in 1993 (Lock, 1993; Venables and Ripley, 2002).
- d. Use *aggregate* to compute the mean length of the 93 cars, split by two categorical variables: *AirBags* (type of airbags available—levels are *Driver & Passenger*, *Driver only*, and *None*), and *Man.trans.avail* (whether the car comes in a manual transmission—levels are *Yes* and *No*).
- e. Produce an interaction plot using the results in (d). Does there appear to be an interactive effect of *AirBags* with *Man.trans.avail* on the mean length of these cars (if you consider only these variables)?
- f. Fit a full two-way ANOVA model for the mean lengths according to the two grouping variables (assume satisfaction of all relevant assumptions). Is the interactive effect statistically significant? Is there evidence of any main effects?

## Solution 19.2

```
> #(a)  
  
> depth.fac <- cut(quakes$depth,breaks=c(0,200,400,680))  
  
> table(depth.fac)  
  
depth.fac  
  
(0,200] (200,400] (400,680]  
  
    418     185     397  
  
> #(b)  
  
> m <- tapply(quakes$stations,depth.fac,mean)  
  
> mc <- quakes$stations-m[as.numeric(depth.fac)]  
  
> hist(mc)  
  
> qqnorm(mc)  
  
> qqline(mc) # Data appear non-normal... Kruskal-Wallis preferred over parametric one-way  
ANOVA  
  
> #(c)
```

```
> kruskal.test(quakes$stations~depth.fac) # P-value > 0.01 (just barely); retain null. Minimal evidence, at best, of a difference in median number of detecting stations according to depth.fac categories.
```

Kruskal-Wallis rank sum test

```
data: quakes$stations by depth.fac
```

```
Kruskal-Wallis chi-squared = 9.0943, df = 2, p-value = 0.0106
```

```
> #(d)
```

```
> library("MASS")
```

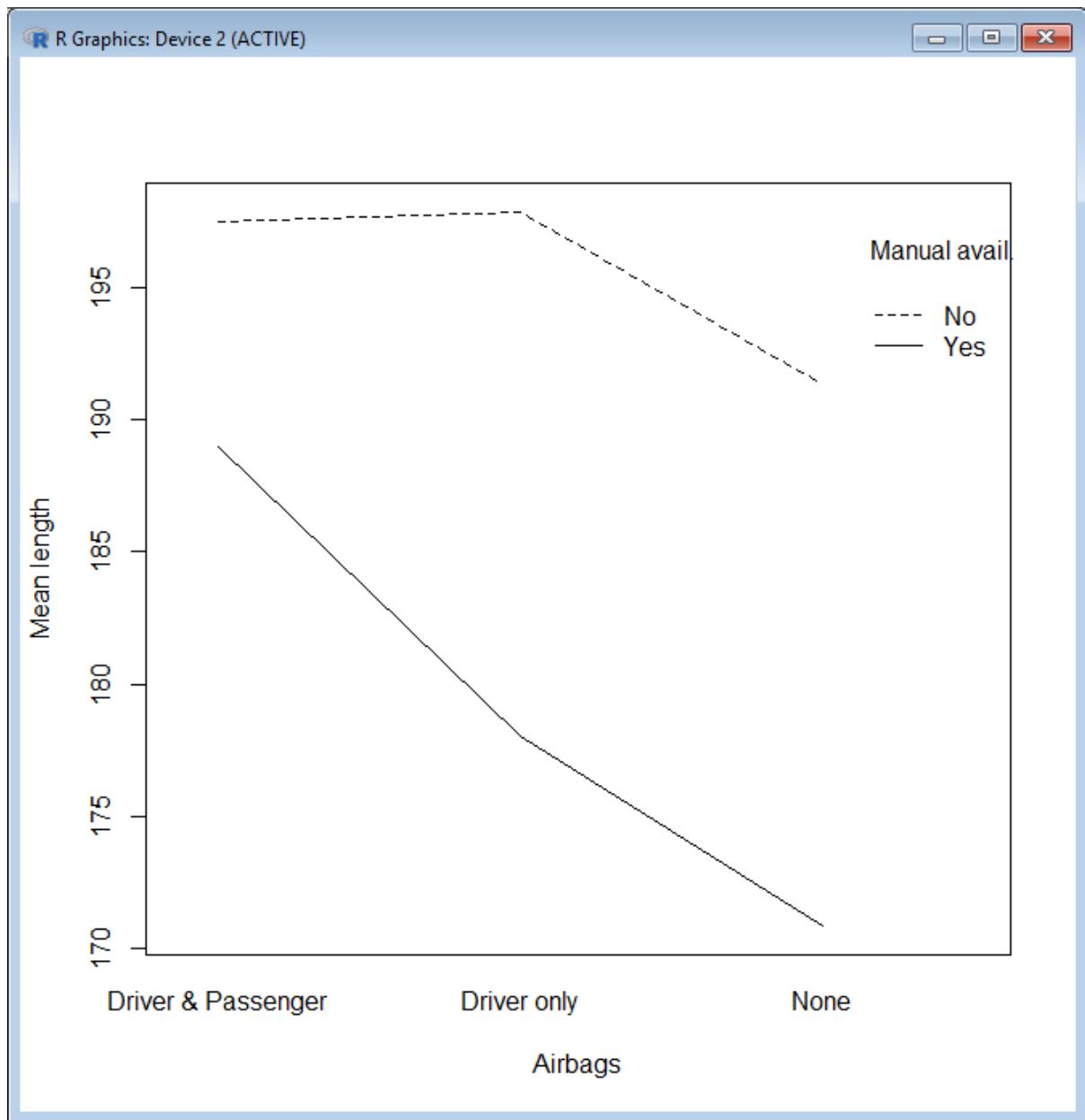
```
> cars.means <-  
aggregate(Cars93$Length,by=list(Cars93$AirBags,Cars93$Man.trans.avail),FUN=mean)
```

```
> cars.means
```

	Group.1	Group.2	x
1	Driver & Passenger	No	197.5000
2	Driver only	No	197.8750
3	None	No	191.3750
4	Driver & Passenger	Yes	189.0000
5	Driver only	Yes	178.0370
6	None	Yes	170.8462

```
> #(e)
```

```
>  
interaction.plot(x.factor=cars.means[,1],trace.factor=cars.means[,2],respons=cars.means$x,trace.label="Manual avail.",xlab="Airbags",ylab="Mean length") # There is some visual indication of interactive behavior owing to the non-parallel nature of the two lines; but rememeber there are no measures of variability of the various group means displayed on this plot...
```



```
> #(f)
```

```
> summary(aov(Length~AirBags+Man.trans.avail+AirBags:Man.trans.avail,data=Cars93)) # No formal statistical evidence of an interactive effect. Strong evidence of main effects, however -- both 'Airbags' and 'Man.trans.avail' appear to be related to car length.
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)						
AirBags	2	3752	1876	18.048	2.78e-07 ***						
Man.trans.avail	1	6388	6388	61.447	1.05e-11 ***						
AirBags:Man.trans.avail	2	433	217	2.084	0.131						
Residuals	87	9044	104								
	---										
Signif. codes:	0	***	0.001	**	0.01	*	0.05	.	0.1	''	1

# Chapter 20: Simple Linear Regression

Estimated time to complete: **30 minutes**

## Exercise 20.1

Continue to use the *survey* data frame from the package *MASS* for the next few exercises.

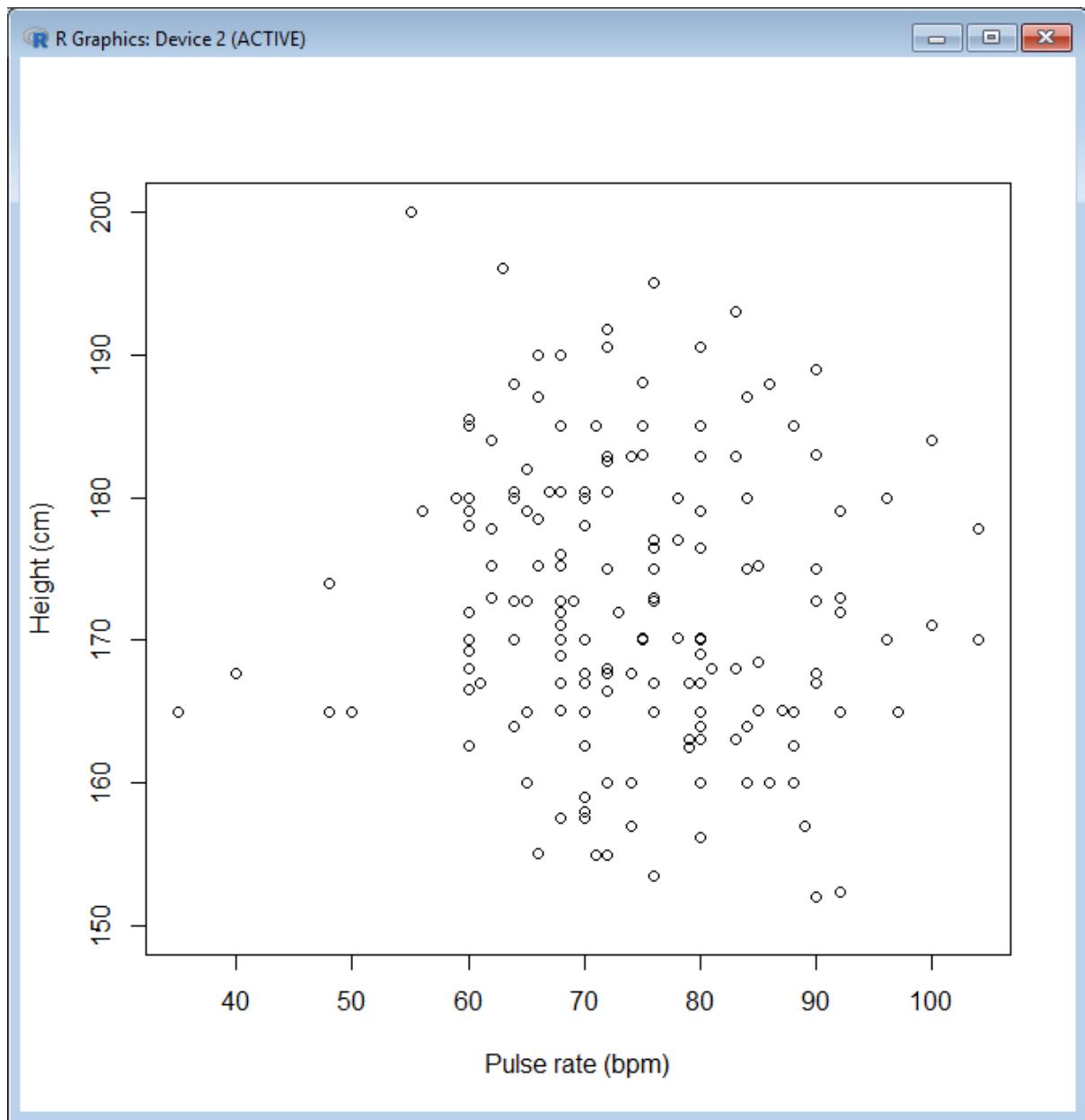
- a. Using your fitted model of student height on writing handspan, *survfit*, provide point estimates and 99 percent confidence intervals for the mean student height for handspans of 12, 15.2, 17, and 19.9 cm.
- b. In Section 20.1, you defined the object *incomplete.obs*, a numeric vector that provides the records of *survey* that were automatically removed from consideration when estimating the model parameters. Now, use the *incomplete.obs* vector along with *survey* and Equation (20.3) to calculate  $\hat{\beta}_0$  and  $\hat{\beta}_1$  in R. (Remember the functions *mean*, *sd*, and *cor*. Ensure your answers match the output from *survfit*.)
- c. The *survey* data frame has a number of other variables present aside from *Height* and *Wr.Hnd*. For this exercise, the end aim is to fit a simple linear model to predict the mean student height, but this time from their pulse rate, given in *Pulse* (continue to assume the conditions listed in Section 20.2 are satisfied).
  - i. Fit the regression model and produce a scatterplot with the fitted line superimposed upon the data. Make sure you can write down the fitted model equation and keep the plot open.
  - ii. Identify and interpret the point estimate of the slope, as well as the outcome of the test associated with the hypotheses  $H_0 : \beta_1 = 0$ ;  $H_A : \beta_1 \neq 0$ . Also find a 90 percent CI for the slope parameter.
  - iii. Using your model, add lines for 90 percent confidence and prediction interval bands on the plot from (i) and add a legend to differentiate between the lines.
  - iv. Create an *incomplete.obs* vector for the current “height on pulse” data. Use that vector to calculate the sample mean of the height observations that were used for the model fitted in (i). Then add a perfectly horizontal line to the plot at this mean (use color or line type options to avoid confusion with the other lines present). What do you notice? Does the plot support your conclusions from (ii)?

Next, examine the help file for the *mtcars* data set, which you first saw in Exercise 13.4 on page 287. For this exercise, the goal is to model fuel efficiency, measured in miles per gallon (MPG), in terms of the overall weight of the vehicle (in thousands of pounds).

- d. Plot the data—*mpg* on the y-axis and *wt* on the x-axis.
- e. Fit the simple linear regression model. Add the fitted line to the plot from (d).
- f. Write down the regression equation and interpret the point estimate of the slope. Is the effect of *wt* on mean *mpg* estimated to be statistically significant?
- g. Produce a point estimate and associated 95 percent PI for a car that weighs 6,000 lbs. Do you trust the model to predict observations accurately for this value of the explanatory variable? Why or why not?

## Solution 20.1

```
> library("MASS")
> survfit <- lm(Height~Wr.Hnd,data=survey)
> #(a)
>
predict(survfit,newdata=data.frame(Wr.Hnd=c(12,15.2,17,19.9)),interval="confidence",level=0.99
)
      fit     lwr     upr
1 151.3530 146.0901 156.6159
2 161.3262 158.3051 164.3473
3 166.9361 164.9985 168.8737
4 175.9743 174.3066 177.6420
> #(b)
> incomplete.obs <- which(is.na(survey$Height)|is.na(survey$Wr.Hnd))
> rho.xy <- cor(survey$Wr.Hnd,survey$Height,use="complete.obs")
> b1 <- sd(survey$Height[-incomplete.obs])/sd(survey$Wr.Hnd[-incomplete.obs])*rho.xy
> b1
[1] 3.116617
> b0 <- mean(survey$Height[-incomplete.obs])-b1*mean(survey$Wr.Hnd[-incomplete.obs])
> b0
[1] 113.9536
> #(c)
> ##(i)
> plot(survey$Height~survey$Pulse,xlab="Pulse rate (bpm)",ylab="Height (cm)")
```



```
> survfit <- lm(Height~Pulse,data=survey)  
> survfit # Model equation is y = 177.86 - 0.072x
```

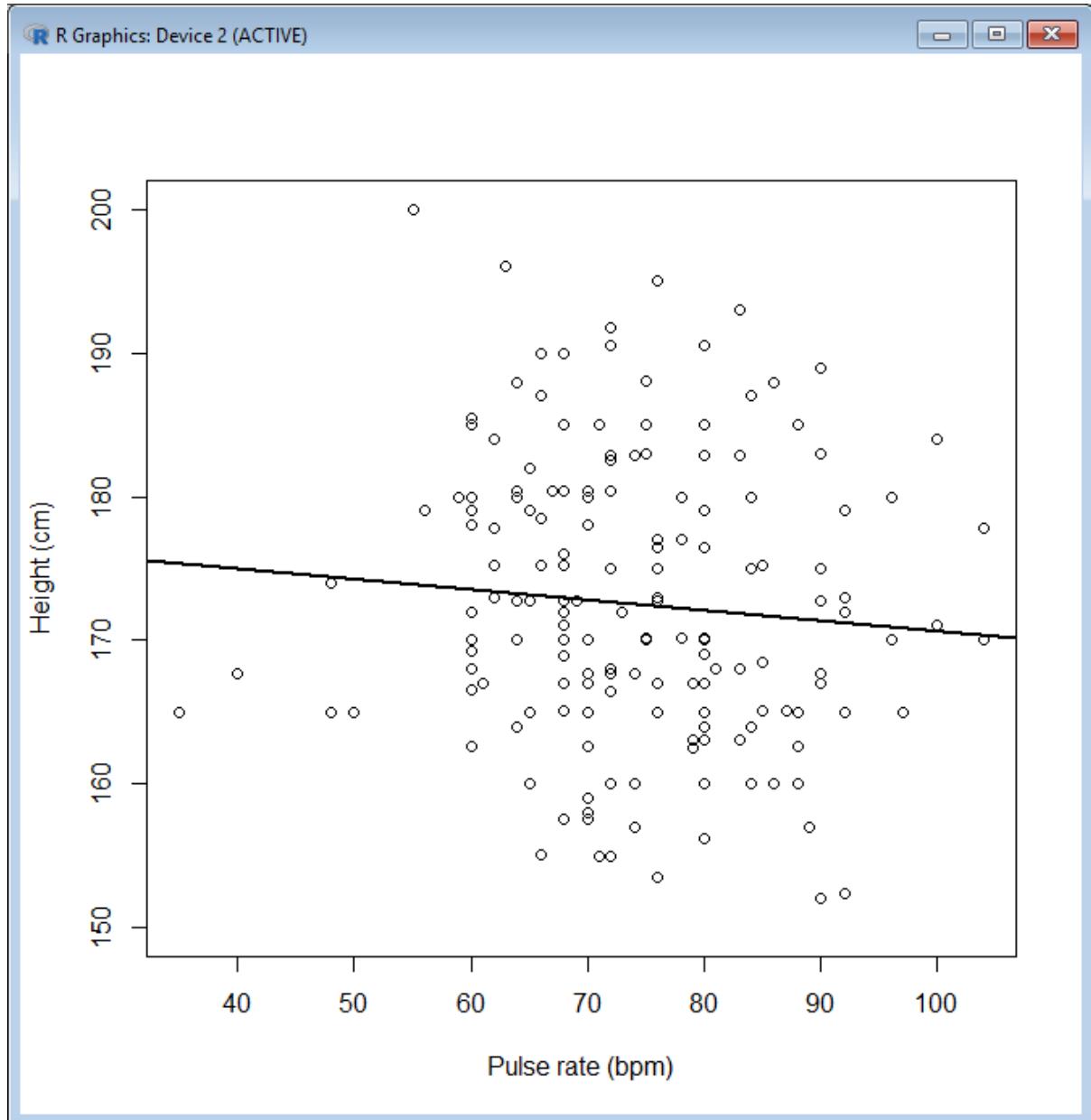
Call:

```
lm(formula = Height ~ Pulse, data = survey)
```

Coefficients:

(Intercept)	Pulse
177.85708	-0.07225

```
> abline(survfit,lwd=2)
```



```
> ##(ii)
```

```
> summary(survfit) # For each additional bpm, the mean student height is estimated to decrease by  
0.072cm; p-value for slope is 0.275. No evidence to reject H0. Insufficient evidence to conclude  
that pulse rate affects mean student height.
```

Call:

```
lm(formula = Height ~ Pulse, data = survey)
```

Residuals:

Min	1Q	Median	3Q	Max
-----	----	--------	----	-----

-19.3543 -7.2019 -0.9439 7.2622 26.1168

Coefficients:

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

(Intercept) 177.85708 4.93485 36.041 <2e-16 \*\*\*

Pulse -0.07225 0.06598 -1.095 0.275

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9.884 on 169 degrees of freedom

(66 observations deleted due to missingness)

Multiple R-squared: 0.007046, Adjusted R-squared: 0.001171

F-statistic: 1.199 on 1 and 169 DF, p-value: 0.275

> confint(survfit,level=0.9)

5 % 95 %

(Intercept) 169.6952304 186.01892281

Pulse -0.1813757 0.03687061

> ##(iii)

> xseq <- data.frame(Pulse=seq(30,110,length=100))

> survfit.ci <- predict(survfit,newdata=xseq,interval="confidence",level=0.9)

> survfit.pi <- predict(survfit,newdata=xseq,interval="prediction",level=0.9)

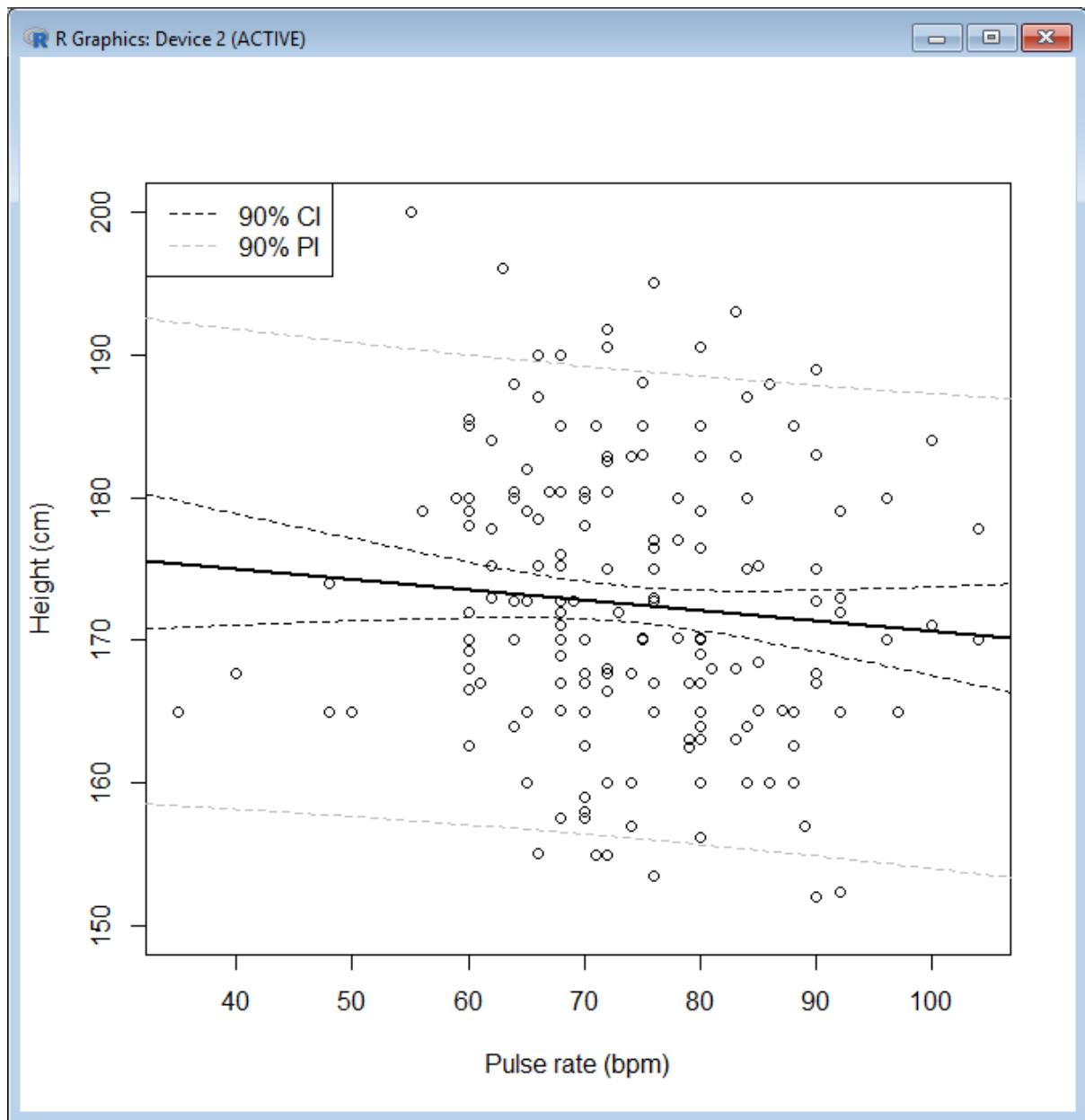
> lines(xseq[,1],survfit.ci[,2],lty=2)

> lines(xseq[,1],survfit.ci[,3],lty=2)

> lines(xseq[,1],survfit.pi[,2],lty=2,col="grey")

> lines(xseq[,1],survfit.pi[,3],lty=2,col="grey")

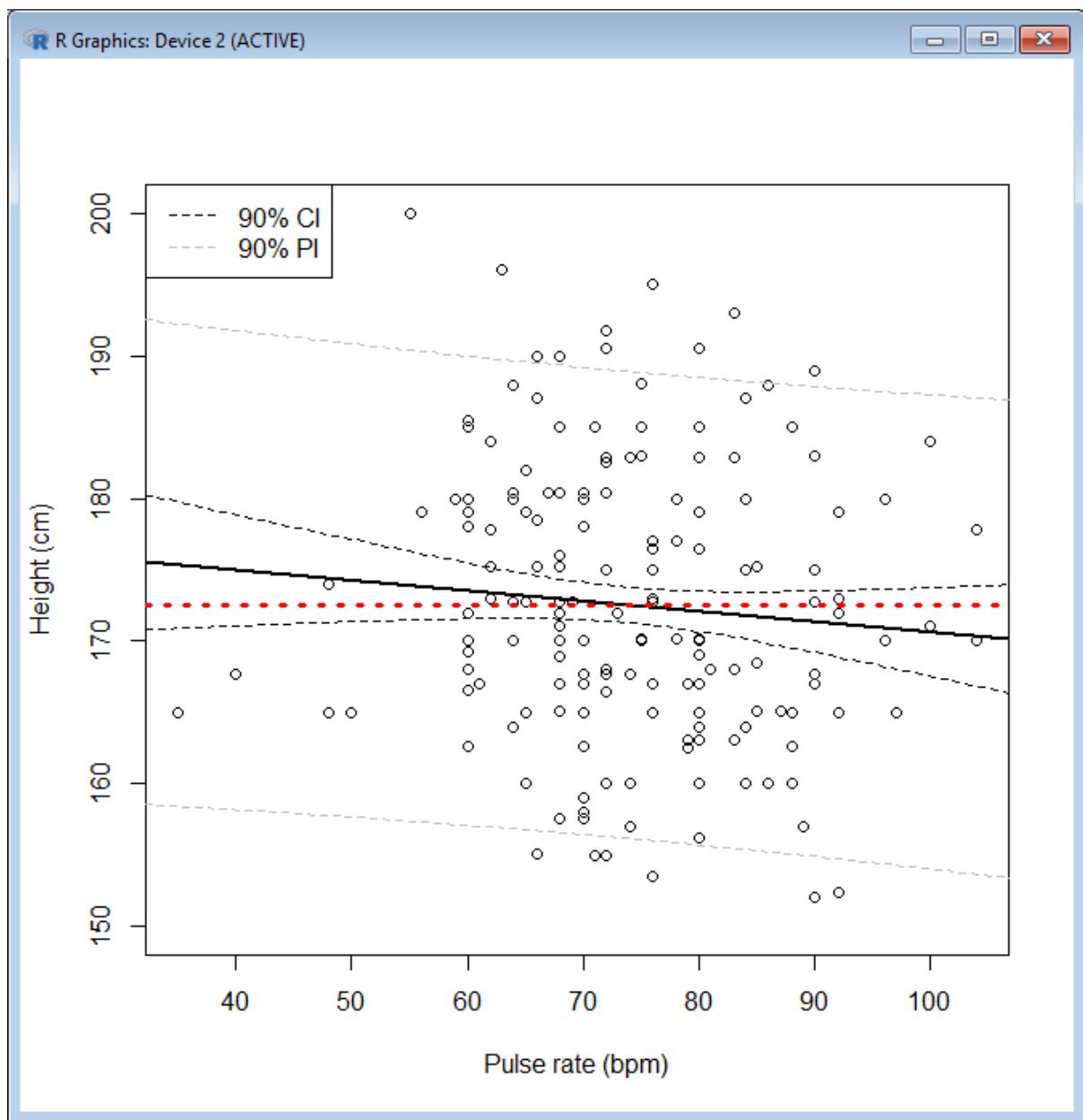
> legend("topleft",legend=c("90% CI","90% PI"),lty=2,col=c("black","grey"))



```

> ##(iv)
> incomplete.obs <- which(is.na(survey$Height)|is.na(survey$Pulse))
> abline(h=mean(survey$Height[-incomplete.obs]),col=2,lty=3,lwd=3) # The line sits in the
  middle of the CI bands without breaching them. This supports the conclusion that pulse rate is not
  significantly related to mean student height.

```



```
> #(d)
```

```
> ?mtcars
```

mtcars {datasets}

R Documentation

## Motor Trend Car Road Tests

### Description

The data was extracted from the 1974 *Motor Trend* US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models).

### Usage

```
mtcars
```

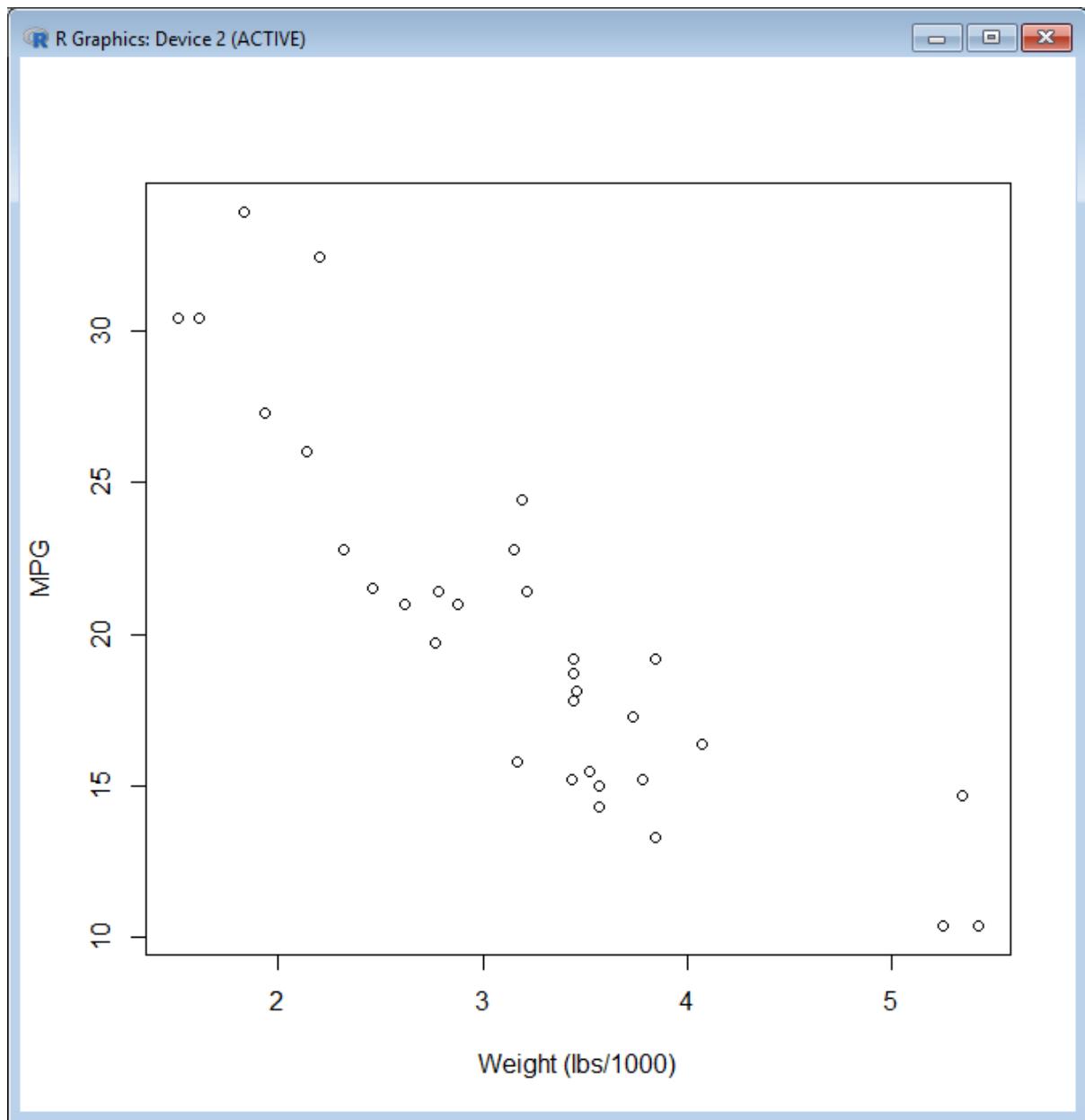
### Format

A data frame with 32 observations on 11 variables.

```
[, 1] mpg Miles/(US) gallon  
[, 2] cyl  Number of cylinders  
[, 3] disp Displacement (cu.in.)  
[, 4] hp   Gross horsepower  
[, 5] drat Rear axle ratio  
[, 6] wt   Weight (1000 lbs)  
[, 7] qsec 1/4 mile time  
[, 8] vs    V/S  
[, 9] am    Transmission (0 = automatic, 1 = manual)  
[,10] gear  Number of forward gears  
[,11] carb  Number of carburetors
```

### Source

```
> plot(mtcars$mpg~mtcars$wt,xlab="Weight (lbs/1000)",ylab="MPG")
```



```
> #(e)  
> carfit <- lm(mpg~wt,data=mtcars)  
> carfit
```

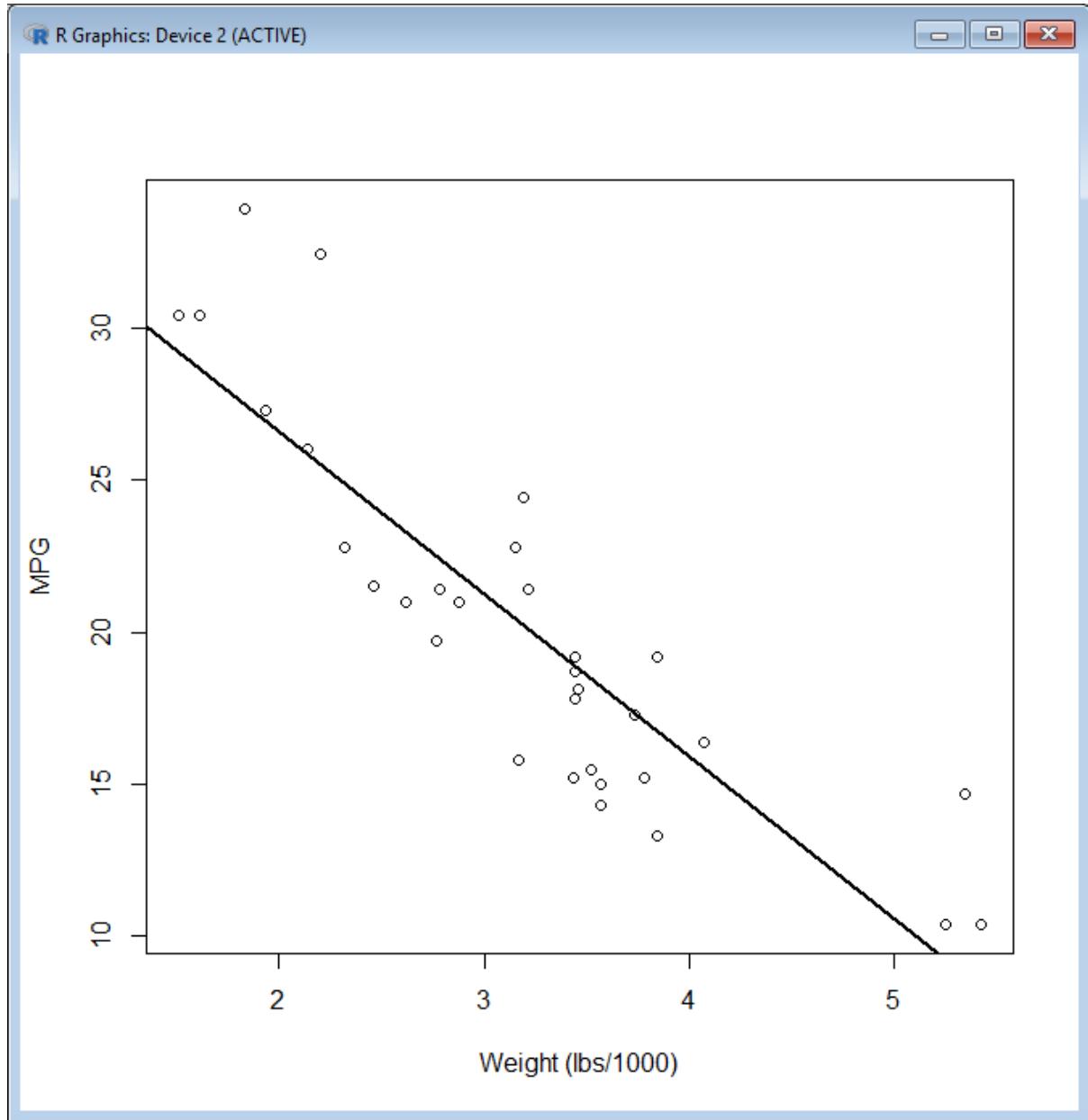
Call:

```
lm(formula = mpg ~ wt, data = mtcars)
```

Coefficients:

(Intercept)	wt
37.285	-5.344

```
> abline(carfit,lwd=2)
```



```
> #(f)
```

```
> summary(carfit) # mean MPG = 37.28 - 5.34*weight # For each extra 1000lbs of weight, the  
mean MPG decreases by 5.34; p-value for slope is very small---result is statistically significant---  
strong evidence to suggest that mean MPG changes according to weight (for cars of this era).
```

Call:

```
lm(formula = mpg ~ wt, data = mtcars)
```

Residuals:

Min	1Q	Median	3Q	Max
-4.5432	-2.3647	-0.1252	1.4096	6.8727

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	37.2851	1.8776	19.858	< 2e-16 ***
wt	-5.3445	0.5591	-9.559	1.29e-10 ***
---				

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.046 on 30 degrees of freedom

Multiple R-squared: 0.7528, Adjusted R-squared: 0.7446

F-statistic: 91.38 on 1 and 30 DF, p-value: 1.294e-10

> #(g)

> predict(carfit,newdata=data.frame(wt=6),interval="prediction",level=0.95) # Predicting at 6000lbs seems untrustworthy. Extrapolation is far enough outside the range of the observed data such that the associated PI has a lower limit that is negative, which makes no sense in terms of the response variable of MPG.

fit	lwr	upr
1 5.218297	-1.85279	12.28938

## Exercise 20.2

Continue using the *survey* data frame from the package *MASS* for the next few exercises.

- a. The *survey* data set has a variable named *Exer*, a factor with  $k = 3$  levels describing the amount of physical exercise time each student gets: none, some, or frequent. Obtain a count of the number of students in each category and produce side-by-side boxplots of student height split by exercise.
- b. Assuming independence of the observations and normality as usual, fit a linear regression model with height as the response variable and exercise as the explanatory variable (dummy coding). What's the default reference level of the predictor?  
Produce a model summary.
- c. Draw a conclusion based on the fitted model from (b)—does it appear that exercise frequency has any impact on mean height?  
What is the nature of the estimated effect?
- d. Predict the mean heights of one individual in each of the three exercise categories, accompanied by 95 percent prediction intervals.
- e. Do you arrive at the same result and interpretation for the height-by-exercise model if you construct an ANOVA table using *aov*?
- f. Is there any change to the outcome of (e) if you alter the model so that the reference level of the exercise variable is “none”?

Would you expect there to be?

Now, turn back to the ready-to-use *mtcars* data set. One of the variables in this data frame is *qsec*, described as the time in seconds it takes to race a quarter mile; another is *gear*, the number of forward gears (cars in this data set have either 3, 4, or 5 gears).

- g. Using the vectors straight from the data frame, fit a simple linear regression model with *qsec* as the response variable and *gear* as the explanatory variable and interpret the model summary.
- h. Explicitly convert *gear* to a factor vector and refit the model. Compare the model summary with that from (g). What do you find?
- i. Explain, with the aid of a relevant plot in the same style as the right image of Figure 20-6, why you think there is a difference between the two models (g) and (h).

## Solution 20.2

```
> library("MASS")
```

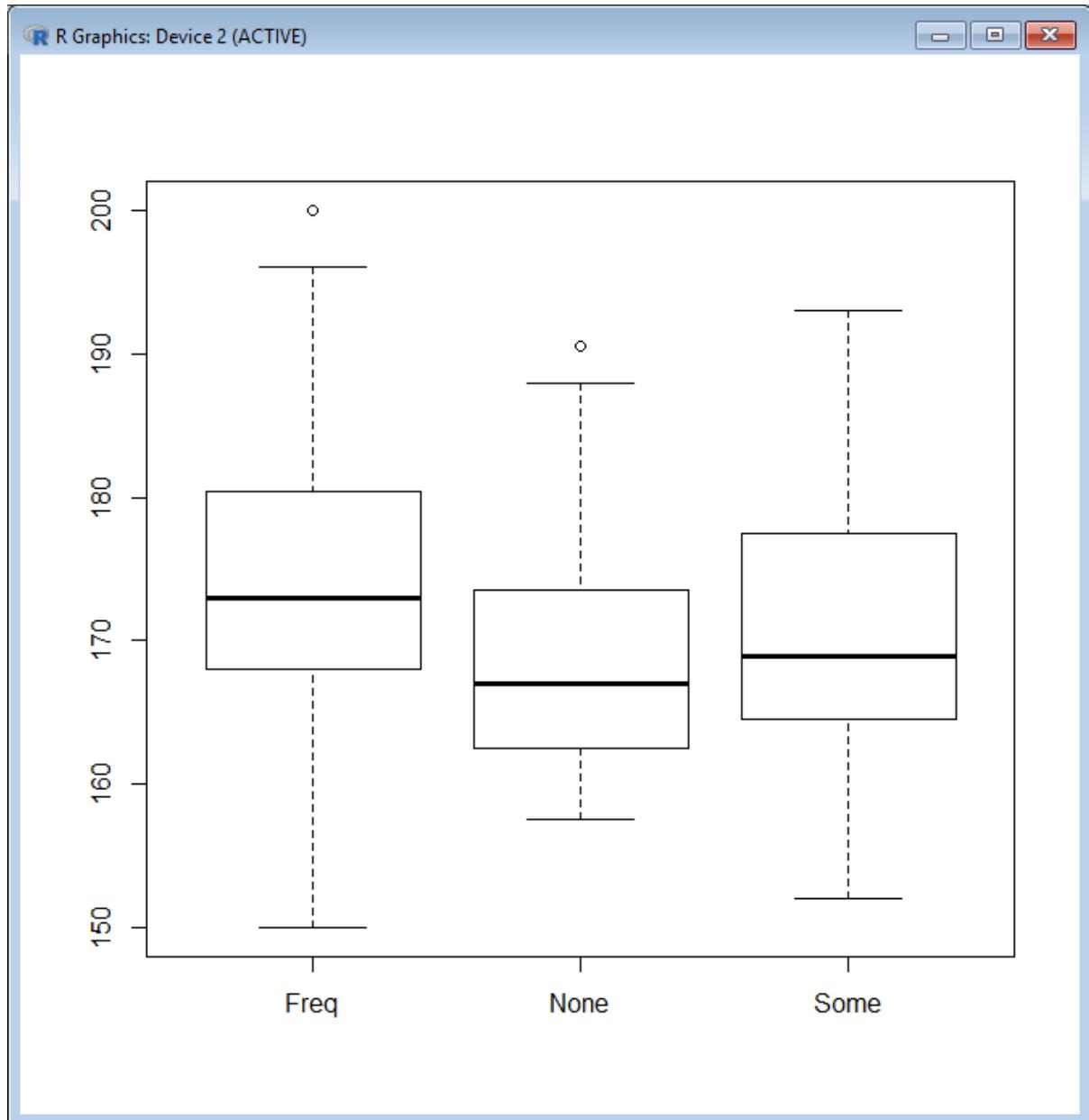
```
> #(a)
```

```
> table(survey$Exer)
```

Freq	None	Some
------	------	------

115	24	98
-----	----	----

```
> boxplot(survey$Height~survey$Exer)
```



```
> #(b)
```

```
> survfit <- lm(Height~Exer,data=survey)

> summary(survfit) # The reference level of the predictor defaults to the first level of the factor,
which in this case (as is the default in R) is alphabetically arranged to be 'Freq'.
```

Call:

```
lm(formula = Height ~ Exer, data = survey)
```

Residuals:

Min	1Q	Median	3Q	Max
-24.607	-6.397	-1.607	6.103	25.393

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	174.6067	0.9396	185.836	< 2e-16 ***
ExerNone	-5.5787	2.3489	-2.375	0.01847 *
ExerSome	-4.2098	1.4094	-2.987	0.00316 **
---				
Signif. codes:	0 ‘***’	0.001 ‘**’	0.01 ‘*’	0.05 ‘.’
	0.1 ‘ ’	1		

Residual standard error: 9.628 on 206 degrees of freedom

(28 observations deleted due to missingness)

Multiple R-squared: 0.05333, Adjusted R-squared: 0.04414

F-statistic: 5.802 on 2 and 206 DF, p-value: 0.003536

```
> #(c)
```

```
> # We observe that both of the levels for which we've obtained coefficient estimates yield p-values that suggest evidence the coefficients are different to zero.
```

```
> # The coefficient corresponding to 'some' has the smallest p-value of the two additive dummy levels.
```

```
> # The negative point estimates of both estimates tell us that the model predicts the effect on height of being in either the 'none' or the 'some' categories, when compared to the 'frequent' category, is one of a decrease. In other words, it appears those who exercise less than 'frequent' are shorter on average.
```

```

> # The shortest mean height is reserved for those in the 'none' exercise category; the estimated
  coefficient (-5.58) is more extreme than the coefficient for 'some' (-4.21).

> # Overall statistical significance of the predictor (in terms of the effect of exercise on height) is
  supported by the global (omnibus F-test) P-value of 0.0035.

> #(d)

> one.of.each <- factor(levels(survey$Exer))

> one.of.each

[1] Freq None Some

Levels: Freq None Some

> predict(survfit,newdata=data.frame(Exer=one.of.each),interval="prediction")

  fit    lwr    upr
1 174.6067 155.5349 193.6784
2 169.0280 149.5777 188.4783
3 170.3969 151.3027 189.4911

> #(e)

> summary(aov(Height~Exer,data=survey)) # Same 'global' P-value as the lm model summary.
There is evidence to suggest mean student height differs according to exercise frequency.

  Df Sum Sq Mean Sq F value Pr(>F)
Exer      2  1076   537.8   5.802 0.00354 ***
Residuals 206 19095   92.7
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

28 observations deleted due to missingness

> #(f)

> ExerReordered <- relevel(survey$Exer,ref="None")

> levels(ExerReordered)

[1] "None" "Freq" "Some"

> summary(aov(Height~ExerReordered,data=survey)) # There is no change to the omnibus F-test
if we reorder the the reference level of exercise to be 'none', and nor should we expect there to be.
The global test of a difference between the means doesn't depend on what we set the baseline
value of the response to be.

  Df Sum Sq Mean Sq F value Pr(>F)
ExerReordered 2  1076   537.8   5.802 0.00354 **

```

```
Residuals 206 19095 92.7
```

```
---
```

```
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
28 observations deleted due to missingness
```

```
> #(g)
```

```
> carfit <- lm(qsec~gear,data=mtcars)
```

```
> summary(carfit) # The effect of 'gear' when treated as a continuous variable is interpreted as a decrease in quarter-mile time of around 0.5 seconds, on average, for each additional forward gear. However, there is no evidence that this effect is different to zero, with a high P-value of 0.243 (and a similar global p-value)
```

Call:

```
lm(formula = qsec ~ gear, data = mtcars)
```

Residuals:

Min	1Q	Median	3Q	Max
-2.7929	-1.1604	-0.3278	1.2122	5.2122

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	19.7482	1.6239	12.161	4e-13 ***
gear	-0.5151	0.4321	-1.192	0.243

```
---
```

```
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Residual standard error: 1.775 on 30 degrees of freedom

Multiple R-squared: 0.04523, Adjusted R-squared: 0.01341

F-statistic: 1.421 on 1 and 30 DF, p-value: 0.2425

```
> #(h)
```

```
> carfit2 <- lm(qsec~factor(gear),data=mtcars)
```

```
> summary(carfit2) # The effect of 'gear' when treated as a categorical variable now appears to be statistically significant. Having 4 gears (as opposed to the reference level of 3) seems to increase the mean quarter-mile time by 1.27 seconds, having 5 gears appears to decrease the mean quarter-mile time by 2.05 seconds. The global (omnibus F-test) p-value is also quite small, yielding evidence in support of an effect of 'gear' on 'qsec'.
```

Call:

```
lm(formula = qsec ~ factor(gear), data = mtcars)
```

Residuals:

Min	1Q	Median	3Q	Max
-2.5050	-0.6667	-0.2060	0.6125	3.9350

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )		
(Intercept)	17.6920	0.3691	47.928	< 2e-16 ***		
factor(gear)4	1.2730	0.5537	2.299	0.02890 *		
factor(gear)5	-2.0520	0.7383	-2.779	0.00946 **		
---						
Signif. codes:	0 ‘***’	0.001 ‘**’	0.01 ‘*’	0.05 ‘.’	0.1 ‘ ’	1

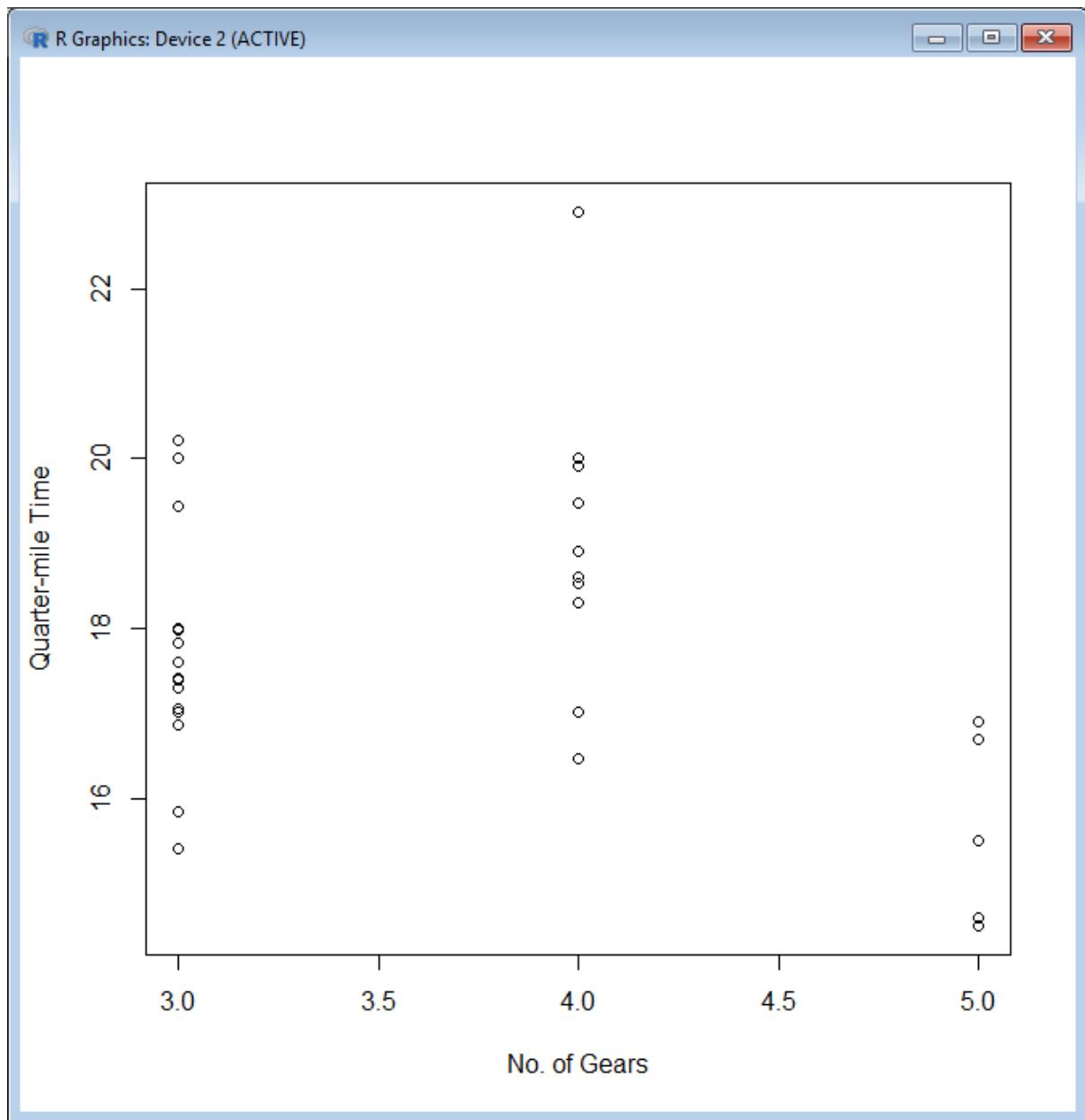
Residual standard error: 1.43 on 29 degrees of freedom

Multiple R-squared: 0.4012, Adjusted R-squared: 0.3599

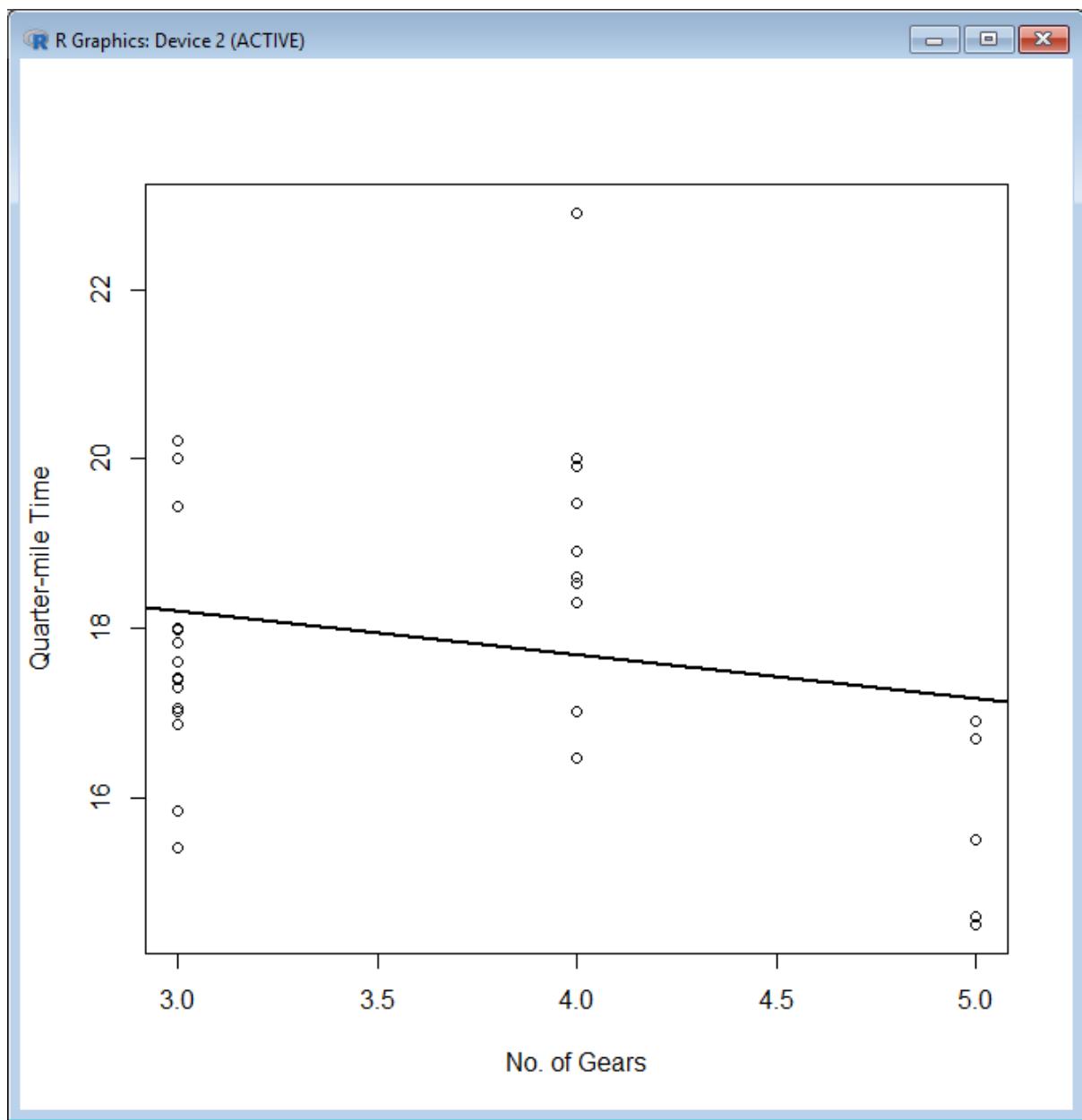
F-statistic: 9.715 on 2 and 29 DF, p-value: 0.0005897

```
> #(i)
```

```
> plot(mtcars$qsec~mtcars$gear,xlab="No. of Gears",ylab="Quarter-mile Time")
```



> abline(carfit,lwd=2) # The plot indicates clearly that the difference between the two models is due to the fact that the relationship cannot be well explained by a continuous straight line. The first model therefore is incapable of realistically capturing the effect of changing categories in 'gear' on 'qsec'.



>

# Chapter 21: Multiple Linear Regression

Estimated time to complete: 30 minutes

## Exercise 21.1

In the *MASS* package, you'll find the data frame *cats*, which provides data on sex, body weight (in kilograms), and heart weight (in grams) for 144 household cats (see Venables and Ripley, 2002, for further details); you can read the documentation with a call to `?cats`. Load the *MASS* package with a call to `library("MASS")`, and access the object directly by entering *cats* at the console prompt.

- a. Plot heart weight on the vertical axis and body weight on the horizontal axis, using different colors or point characters to distinguish between male and female cats. Annotate your plot with a legend and appropriate axis labels.
- b. Fit a least-squares multiple linear regression model using heart weight as the response variable and the other two variables as predictors, and view a model summary.
  - i. Write down the equation for the fitted model and interpret the estimated regression coefficients for body weight and sex. Are both statistically significant? What does this say
  - ii. about the relationship between the response and predictors?
  - iii. Report and interpret the coefficient of determination and the outcome of the omnibus  $F$ -test.
- c. Tilman's cat, Sigma, is a 3.4 kg female. Use your model to estimate her mean heart weight and provide a 95 percent prediction interval.
- d. Use `predict` to superimpose continuous lines based on the fitted linear model on your plot from (a), one for male cats and one for female. What do you notice? Does this reflect the statistical significance (or lack thereof) of the parameter estimates? The *boot* package (Davison and Hinkley, 1997; Canty and Ripley, 2015) is another library of R code that's included with the standard installation but isn't automatically loaded. Load *boot* with a call to `library("boot")`. You'll find a data frame called *nuclear*, which contains data on the construction of nuclear power plants in the United States in the late 1960s (Cox and Snell, 1981).
- e. Access the documentation by entering `?nuclear` at the prompt and examine the details of the variables. (Note there is a mistake for *date*, which provides the date that the construction permits were issued—it should read “measured in years since January 1 **1900** to the nearest month.”) Use `pairs` to produce a quick scatterplot matrix of the data.
- f. One of the original objectives was to predict the cost of further construction of these power plants. Create a fit and summary of a linear regression model that aims to model *cost* by *t1* and *t2*, two variables that describe different elapsed times associated with the application for and issue of various permits. Take note of the estimated regression coefficients and their significance in the fitted model.
- g. Refit the model, but this time also include an effect for the date the construction permit was issued. Contrast the output for this new model against the previous one. What do you notice, and what does this information suggest about the relationships in the data with respect to these predictors?
- h. Fit a third model for power plant cost, using the predictors for “date of permit issue,” “power plant capacity,” and the binary variable describing whether the plant was sited in the northeastern United States. Write down the fitted model equation and provide 95 percent confidence intervals for each estimated coefficient.

The following table gives an excerpt of a historical data set compiled between 1961 and 1973. It concerns the annual murder rate in Detroit, Michigan; the data were originally presented and analyzed by Fisher (1976) and are reproduced here from Harraway (1995).

In the data set you'll find the number of murders, police officers, and gun licenses issued per 100,000 population, as well as the overall unemployment rate as a percentage of the overall population.

Murders	Police	Unemployment	Guns
8.60	260.35	11.0	178.15
8.90	269.80	7.0	156.41
8.52	272.04	5.2	198.02
8.89	272.96	4.3	222.10
13.07	272.51	3.5	301.92
14.57	261.34	3.2	391.22
21.36	268.89	4.1	665.56
28.03	295.99	3.9	1131.21
31.49	319.87	3.6	837.60
37.39	341.43	7.1	794.90
46.26	356.59	8.4	817.74
47.24	376.69	7.7	583.17
52.33	390.19	6.3	709.59

- i. Create your own data frame in your R workspace and produce a scatterplot matrix. Which of the variables appears to be most strongly related to the murder rate?
- j. Fit a multiple linear regression model using the number of murders as the response and all other variables as predictors.  
Write down the model equation and interpret the coefficients. Is it reasonable to state that all relationships between the response and the predictors are causal?
- k. Identify the amount of variation in the response attributed to the joint effect of the three explanatory variables. Then refit the model excluding the predictor associated with the largest (in other words, “most nonsignificant”) p-value. Compare the new coefficient of determination with that of the previous model. Is there much difference?
- l. Use your model from (k) to predict the mean number of murders per 100,000 residents, with 300 police officers and 500 issued gun licenses. Compare this to the mean response if there were no gun licenses issued and provide 99 percent confidence intervals for both predictions.

## Solution 21.1

```
> library("MASS")
```

```
> ?cats
```

The screenshot shows a web browser window displaying the R Documentation for the `cats` dataset. The URL in the address bar is `127.0.0.1:15361/library/MASS/html`. The page title is "Anatomical Data from Domestic Cats". The content includes sections for **Description**, **Usage**, **Format**, and **Source**. The **Description** section states: "The heart and body weights of samples of male and female cats used for *digitalis* experiments. The cats were all adult, over 2 kg body weight." The **Usage** section shows the command `cats`. The **Format** section describes the data frame columns: **Sex** (factor levels "F" and "M"), **Bwt** (body weight in kg), and **Hwt** (heart weight in g). The **Source** section cites R. A. Fisher (1947) for the analysis of covariance method.

cats {MASS}

R Documentation

Anatomical Data from Domestic Cats

**Description**

The heart and body weights of samples of male and female cats used for *digitalis* experiments. The cats were all adult, over 2 kg body weight.

**Usage**

```
cats
```

**Format**

This data frame contains the following columns:

**Sex**  
sex: Factor with levels "F" and "M".

**Bwt**  
body weight in kg.

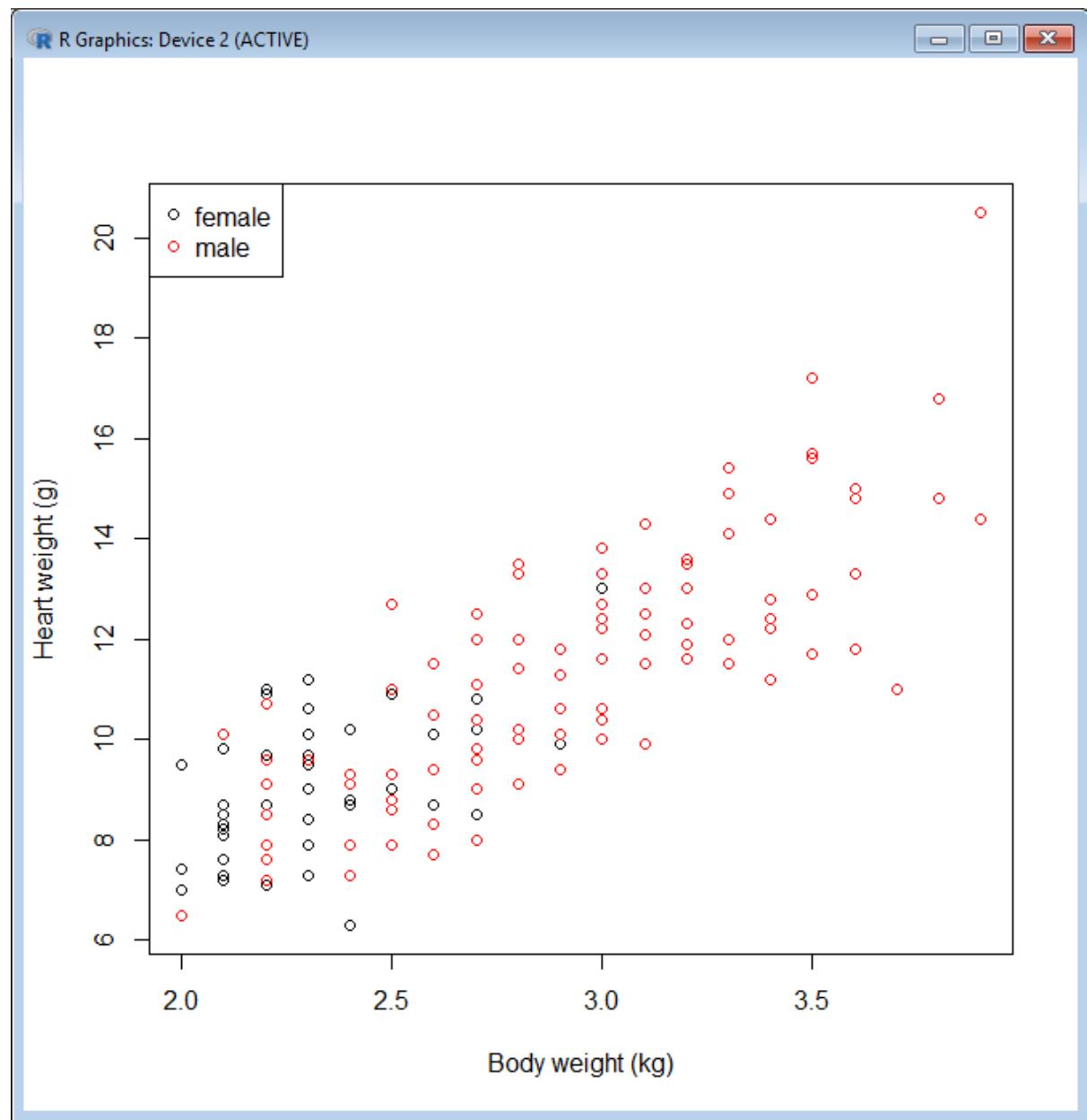
**Hwt**  
heart weight in g.

**Source**

R. A. Fisher (1947) The analysis of covariance method for the relation between a part and the whole, *Biometrics* 3, 65–68.

```
> #(a)  
> plot(cats$Bwt,cats$Hwt,col=cats$Sex,xlab="Body weight (kg)",ylab="Heart weight (g)")
```

```
> legend("topleft",legend=c("female","male"),col=c(1,2),pch=c(1,1)) # Females are black, since  
the levels of the factor vector cats$Sex are in the alphabetical order of "F" "M", which R interprets  
as 1 and 2 when this factor vector is passed to the 'col' argument of 'plot'.
```



```
> #(b)  
> cats.fit <- lm(Hwt~Bwt+Sex,data=cats)  
> summary(cats.fit)
```

Call:

```
lm(formula = Hwt ~ Bwt + Sex, data = cats)
```

Residuals:

Min	1Q	Median	3Q	Max
-3.5833	-0.9700	-0.0948	1.0432	5.1016

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )		
(Intercept)	-0.4149	0.7273	-0.571	0.569		
Bwt	4.0758	0.2948	13.826	<2e-16 ***		
SexM	-0.0821	0.3040	-0.270	0.788		
---						
Signif. codes:	0 ‘***’	0.001 ‘**’	0.01 ‘*’	0.05 ‘.’	0.1 ‘ ’	1

Residual standard error: 1.457 on 141 degrees of freedom

Multiple R-squared: 0.6468, Adjusted R-squared: 0.6418

F-statistic: 129.1 on 2 and 141 DF, p-value: < 2.2e-16

```
> ##(i)
```

```
> # "Mean heart weight" = -0.415 + 4.076*"Body weight" - 0.082*"is male"
```

```
> # For cats of the same sex, the effect of an additional kg of body weight is, on average, an extra  
4.076 grams of heart weight. For cats of the same body weight, the heart weight of a male is, on  
average, 0.082 grams lighter than that of a female.
```

```
> # The model states the effect of body weight is highly statistically significant -- there is evidence  
to believe body weight does indeed affect heart weight. However, the effect of sex is not  
significant. There is no statistical evidence to suggest the coefficient for "is male" is any different  
to zero (when also adjusting for body weight).
```

```
> # The above notes imply that the inclusion of "sex" as a predictor is statistically unnecessary  
when it comes to modeling the response for these data.
```

```
> ##(ii)
```

```
> names(summary(cats.fit))
```

```

[1] "call"      "terms"      "residuals"   "coefficients" "aliased"      "sigma"       "df"
"r.squared"  "adj.r.squared" "fstatistic"  "cov.unscaled"

> summary(cats.fit)$r.squared
[1] 0.6468035

> # The coefficient of determination, 'R-squared', shows that for your fitted model, 64.5% of the variation in heart weight is able to be captured by the regression.

> summary(cats.fit)$fstatistic

  value  numdf  dendf
129.1056 2.0000 141.0000

> 1-pf(129.1056,df1=2,df2=141)
[1] 0

> # Reading from the summary output, or running the line above, the result of the omnibus F-test is a tiny p-value; effectively zero. This suggests very strong evidence against the null hypothesis ( $H_0$  is that modeling heart weight isn't improved by taking body weight and sex into account).

> #(c)

> predict(cats.fit,newdata=data.frame(Bwt=3.4,Sex="F"),interval="prediction",level=0.95)

  fit    lwr    upr
1 13.44266 10.46904 16.41628

> #(d)

> plot(cats$Bwt,cats$Hwt,col=cats$Sex,xlab="Body weight (kg)",ylab="Heart weight (g)")

> legend("topleft",legend=c("female","male"),col=c(1,2),pch=c(1,1))

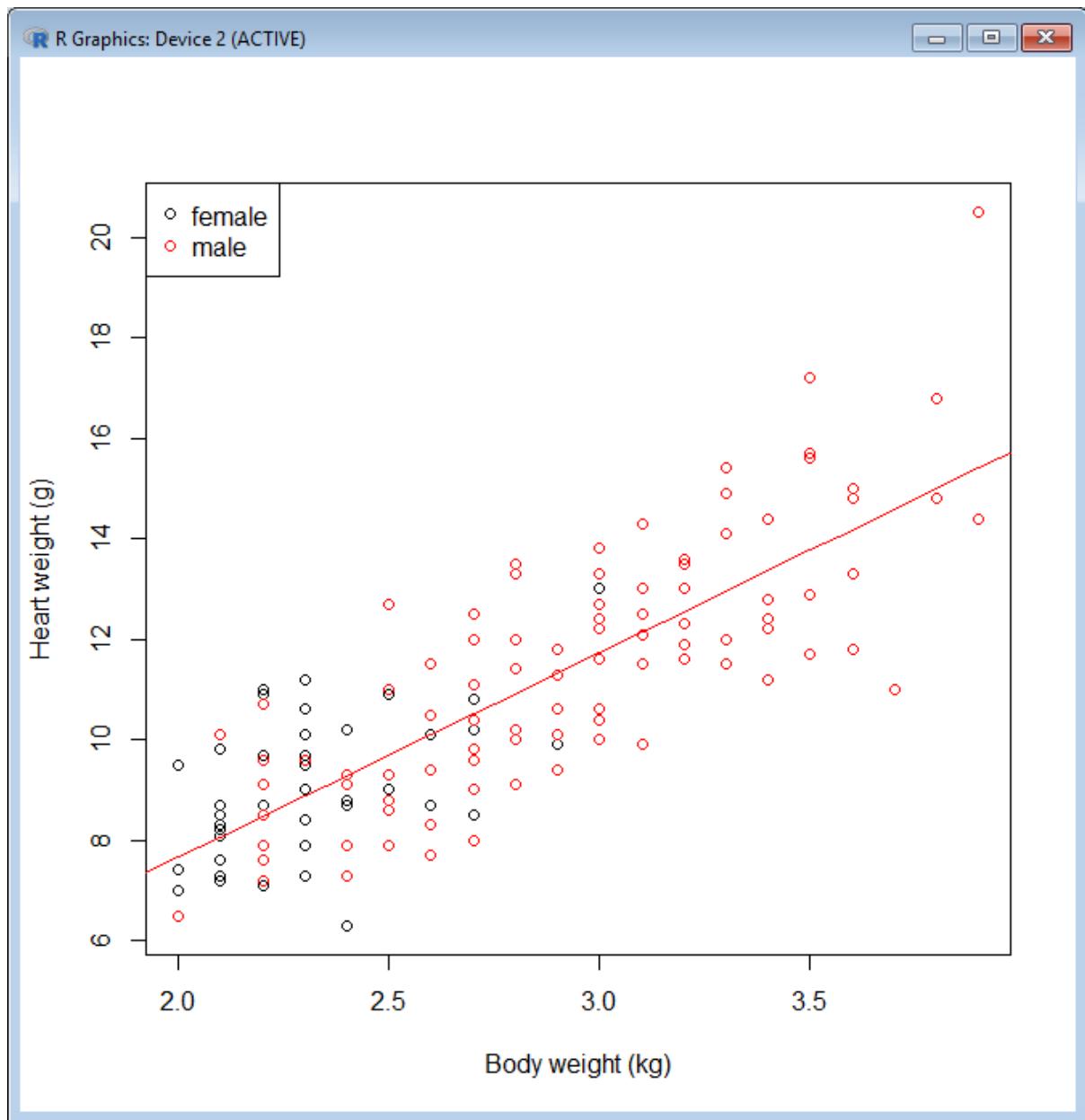
> Bwt.seq <- seq(min(cats$Bwt)-0.5,max(cats$Bwt)+0.5,length=30)

> n <- length(Bwt.seq)

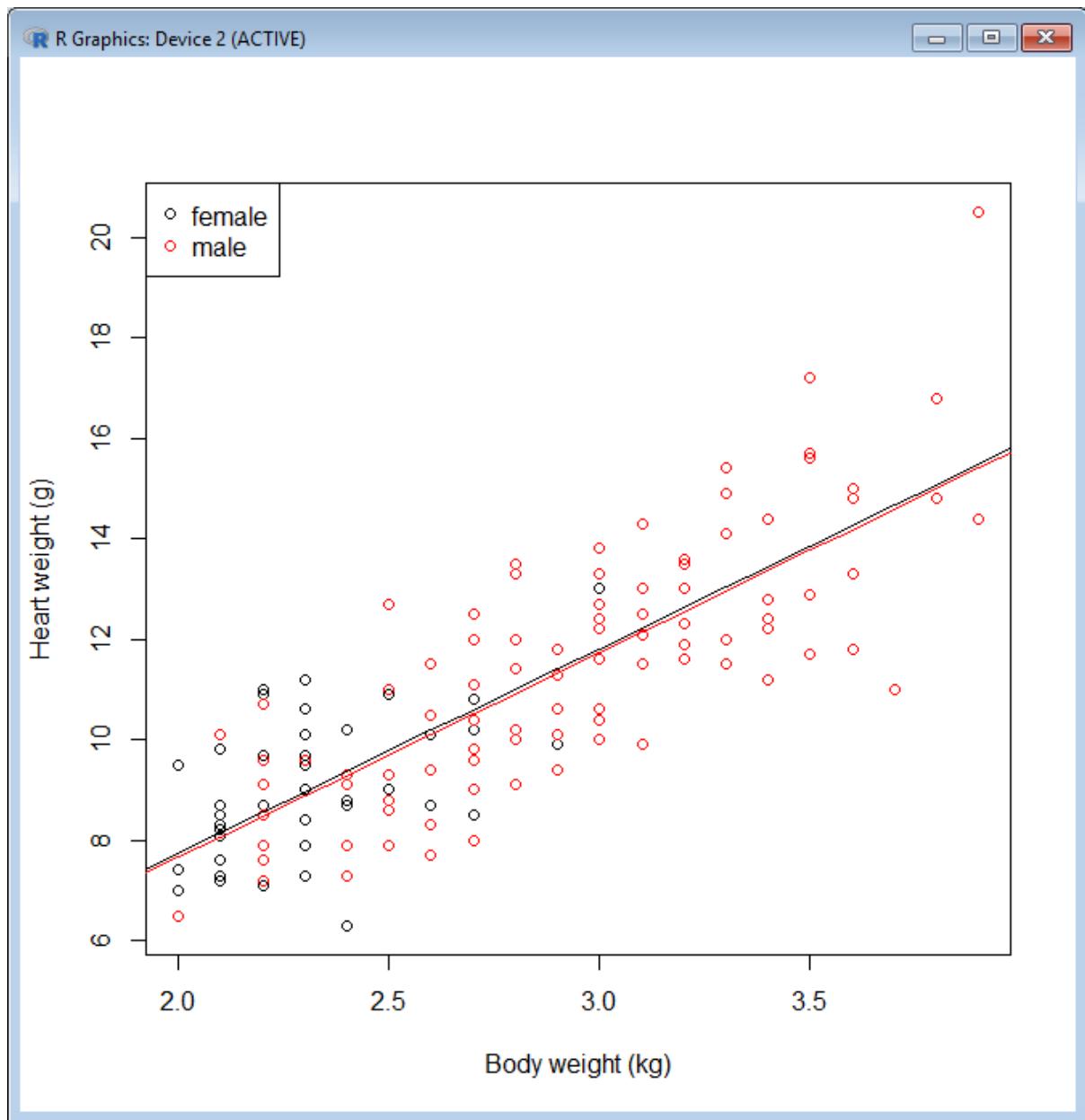
> cats.pred <-
predict(cats.fit,newdata=data.frame(Bwt=rep(Bwt.seq,2),Sex=rep(c("M","F"),each=n)))

> lines(Bwt.seq,cats.pred[1:n],col=2)

```



```
> lines(Bwt.seq,cats.pred[(n+1):(2*n)])
```



> # The two superimposed lines are positively sloped according to the coefficient for "Bwt", but extremely close together, mirroring the minimal impact (and lack of statistical significance) of "Sex".

```
> #(e)  
> library("boot")
```

```
> ?nuclear
```

The screenshot shows a web browser displaying the R Documentation for the 'nuclear' dataset. The URL in the address bar is 127.0.0.1:15361/library/boot/html/. The page title is 'nuclear {boot}'. The main content is titled 'Nuclear Power Station Construction Data'. It includes sections for 'Description' and 'Usage'. The 'Description' section states that the data frame has 32 rows and 11 columns, relating to the construction of 32 light water reactor (LWR) plants in the U.S.A. between the late 1960's and early 1970's. The 'Usage' section shows the command 'nuclear'. The 'Format' section describes the data frame with columns: 'cost' (capital cost in millions of dollars adjusted to 1976 base), 'date' (date of construction permit issuance), and 't1' (time between application and issue). The browser interface includes standard navigation buttons (back, forward, search, etc.) and a toolbar with icons for document, star, and other functions.

nuclear {boot}

R Documentation

## Nuclear Power Station Construction Data

### Description

The `nuclear` data frame has 32 rows and 11 columns.

The data relate to the construction of 32 light water reactor (LWR) plants constructed in the U.S.A in the late 1960's and early 1970's. The data was collected with the aim of predicting the cost of construction of further LWR plants. 6 of the power plants had partial turnkey guarantees and it is possible that, for these plants, some manufacturers' subsidies may be hidden in the quoted capital costs.

### Usage

```
nuclear
```

### Format

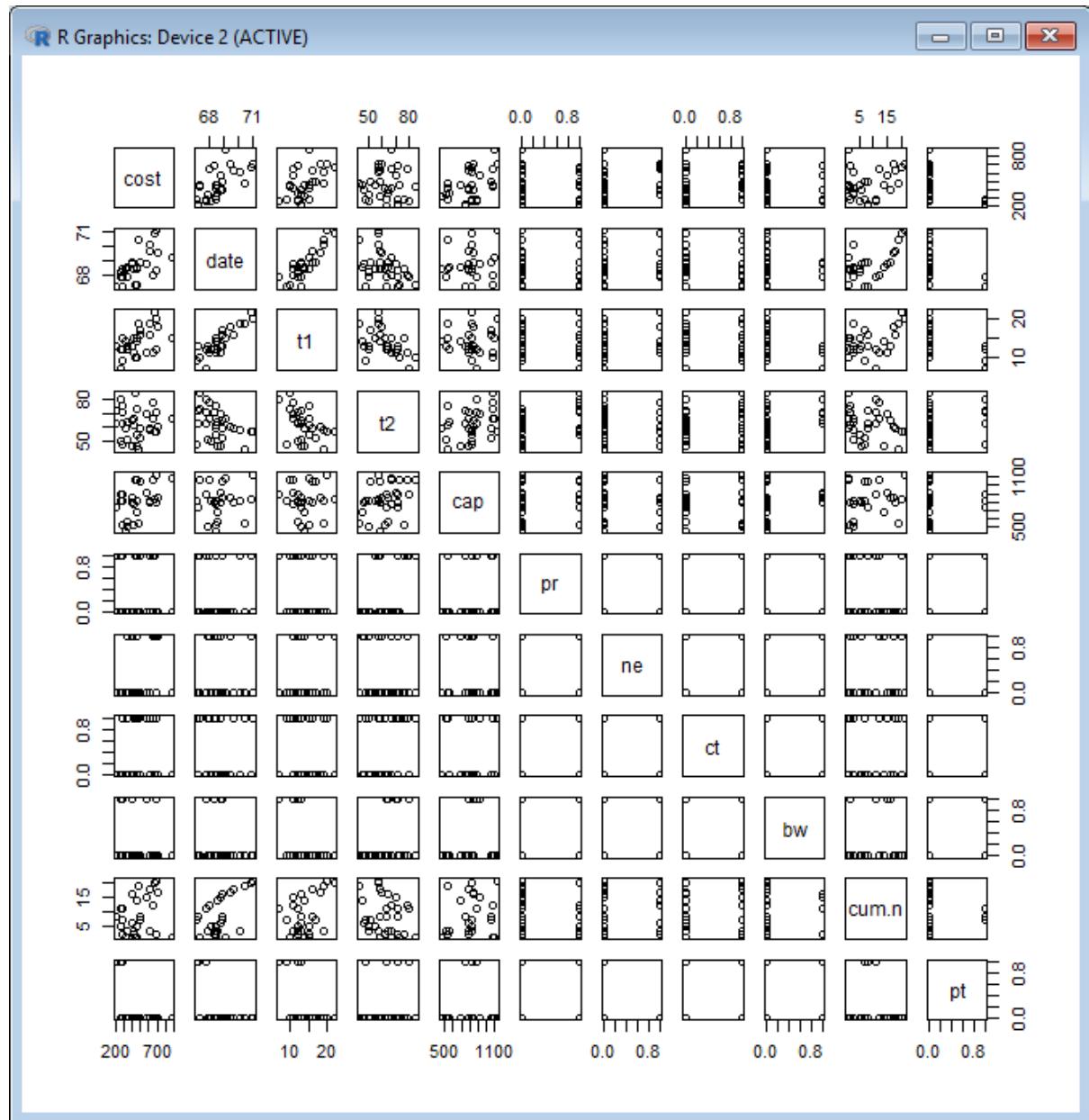
This data frame contains the following columns:

**cost**  
The capital cost of construction in millions of dollars adjusted to 1976 base.

**date**  
The date on which the construction permit was issued. The data are measured in years since January 1 1990 to the nearest month.

**t1**  
The time between application for and issue of the construction permit.

```
> pairs(nuclear)
```



```
> #(f)
```

```
> nuc.fit1 <- lm(cost~t1+t2,data=nuclear)
```

```
> summary(nuc.fit1)
```

Call:

```
lm(formula = cost ~ t1 + t2, data = nuclear)
```

Residuals:

Min	1Q	Median	3Q	Max
-----	----	--------	----	-----

-273.17 -73.42 -13.40 69.31 360.61

Coefficients:

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

(Intercept) -242.146 268.020 -0.903 0.37373

t1 29.908 9.086 3.292 0.00262 \*\*

t2 4.689 2.945 1.592 0.12224

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 150.1 on 29 degrees of freedom

Multiple R-squared: 0.272, Adjusted R-squared: 0.2218

F-statistic: 5.418 on 2 and 29 DF, p-value: 0.01001

```
> #(g)
```

```
> nuc.fit2 <- lm(cost~t1+t2+date,data=nuclear)
```

```
> summary(nuc.fit2)
```

Call:

lm(formula = cost ~ t1 + t2 + date, data = nuclear)

Residuals:

Min 1Q Median 3Q Max

-208.63 -90.74 -12.07 59.78 324.19

Coefficients:

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

(Intercept) -9232.833 2974.432 -3.104 0.00434 \*\*

t1 -5.918 14.281 -0.414 0.68176

t2 4.639 2.601 1.784 0.08535 .

date 138.324 45.617 3.032 0.00519 \*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 132.5 on 28 degrees of freedom

Multiple R-squared: 0.452, Adjusted R-squared: 0.3933

F-statistic: 7.698 on 3 and 28 DF, p-value: 0.000667

> # By including "date" in the linear model, this completely removes the statistical significance of "t1" as seen in the previous model. In fact, the coefficient for "t1" changes signs! What this implies is the previous positive, significant relationship between "t1" and "cost" is actually explained more by "date", owing to the positive correlation of "t1" with "date", and it is "date" that should probably be used in a fitted model. The "t2" predictor remains non-significant, albeit with a reduced p-value in this latest model.

> #(h)

> nuc.fit3 <- lm(cost~date+cap+ne,data=nuclear)

> summary(nuc.fit3) # Fitted model is "cost" = -6458 + 95.4\*"date of permit issue" + 0.42\*"capacity" + 126.1\*"constructed in north-east"

Call:

lm(formula = cost ~ date + cap + ne, data = nuclear)

Residuals:

Min	1Q	Median	3Q	Max
-154.966	-68.202	-3.614	45.919	285.014

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-6.458e+03	1.216e+03	-5.310	1.19e-05 ***
date	9.544e+01	1.773e+01	5.382	9.77e-06 ***
cap	4.157e-01	9.463e-02	4.393	0.000145 ***
ne	1.261e+02	4.092e+01	3.083	0.004575 **

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 99.74 on 28 degrees of freedom

Multiple R-squared: 0.6895, Adjusted R-squared: 0.6562

F-statistic: 20.73 on 3 and 28 DF, p-value: 2.827e-07

```
> confint(nuc.fit3) # All intervals exclude null value of zero, reflecting their significance in the model summary.
```

2.5 % 97.5 %

(Intercept) -8949.8032112 -3966.9745900

date 59.1134640 131.7636535

cap 0.2218791 0.6095524

ne 42.3145363 209.9430014

```
> #(i)
```

```
> detroit <-
```

```
data.frame(Murder=c(8.6,8.9,8.52,8.89,13.07,14.57,21.36,28.03,31.49,37.39,46.26,47.24,52.33),Police=c(260.35,269.8,272.04,272.96,272.51,261.34,268.89,295.99,319.87,341.43,356.59,376.69,390.19),Unemploy=c(11.7,5.2,4.3,3.5,3.2,4.1,3.9,3.6,7.1,8.4,7.7,6.3),Guns=c(178.15,156.41,198.02,222.1,301.92,391.22,665.56,1131.21,837.6,794.9,817.74,583.17,709.59))
```

```
> detroit
```

	Murder	Police	Unemploy	Guns
1	8.60	260.35	11.0	178.15
2	8.90	269.80	7.0	156.41
3	8.52	272.04	5.2	198.02
4	8.89	272.96	4.3	222.10
5	13.07	272.51	3.5	301.92
6	14.57	261.34	3.2	391.22
7	21.36	268.89	4.1	665.56
8	28.03	295.99	3.9	1131.21
9	31.49	319.87	3.6	837.60
10	37.39	341.43	7.1	794.90
11	46.26	356.59	8.4	817.74
12	47.24	376.69	7.7	583.17
13	52.33	390.19	6.3	709.59

```

> pairs(detroit)

> # The number of police seems to be the single most telling variable for prediction of murder
numbers.

> #(j)

> murd.fit <- lm(Murder~Police+Unemploy+Guns,data=detroit)

> summary(murd.fit)

```

Call:

`lm(formula = Murder ~ Police + Unemploy + Guns, data = detroit)`

Residuals:

Min	1Q	Median	3Q	Max
-2.8422	-1.9451	0.2012	0.9172	4.6694

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-68.852509	5.862631	-11.744	9.25e-07 ***
Police	0.280799	0.024657	11.388	1.20e-06 ***
Unemploy	0.147248	0.408235	0.361	0.72665
Guns	0.014177	0.003538	4.007	0.00308 **
---				
Signif. codes:	0 ‘***’	0.001 ‘**’	0.01 ‘*’	0.05 ‘.’
	0.1 ‘ ’	1		

Residual standard error: 2.89 on 9 degrees of freedom

Multiple R-squared: 0.9767, Adjusted R-squared: 0.9689

F-statistic: 125.6 on 3 and 9 DF, p-value: 1.158e-07

```
> summary(murd.fit)$r.squared
```

```
[1] 0.9766753
```

```
> # "Mean murders" = -68.85 + 0.281*"no. of police" + 0.147*"unemployment" + 0.014*"no. of
gun licenses".
```

```

> # After adjusting for "no. of gun licenses", and "unemployment", each additional police per
100000 population is related to a mean increase of 0.28 murders per 100000 population.

> # After adjusting for "no. of police", and "unemployment", each additional gun license per
100000 population results in a mean increase of 0.014 murders per 100000 population.

> # After adjusting for "no. of gun licenses", and "no. of police", each additional percentage of
unemployment results in a mean increase of 0.147 murders per 100000 population.

> # No, it doesn't make sense to claim that *any* of the relationships are causal, particularly based
only one data set and analysis. Causality is extremely difficult to prove in general. In this case, the
idea that having a larger police force 'causes' more murders, for example, is rather difficult to
justify.

> #(k)

> summary(murd.fit)$r.squared

[1] 0.9766753

> # Approx. 97.67% of the variability in mean murder numbers is explained by the three-predictor
model (this can also be seen from the model summary).

> murd.fit2 <- lm(Murder~Police+Guns,data=detroit)

> summary(murd.fit2)

```

Call:

```
lm(formula = Murder ~ Police + Guns, data = detroit)
```

Residuals:

Min	1Q	Median	3Q	Max
-2.9424	-2.1068	0.2775	0.9614	4.6408

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-69.002919	5.587647	-12.349	2.23e-07 ***
Police	0.285048	0.020697	13.772	7.92e-08 ***
Guns	0.013636	0.003062	4.453	0.00123 **
---				
Signif. codes:	0 ‘***’	0.001 ‘**’	0.01 ‘*’	0.05 ‘.’
	0.1 ‘ ’	1		

Residual standard error: 2.761 on 10 degrees of freedom  
Multiple R-squared: 0.9763, Adjusted R-squared: 0.9716  
F-statistic: 206.3 on 2 and 10 DF, p-value: 7.417e-09

```
> summary(murd.fit2)$r.squared
```

```
[1] 0.9763381
```

```
> ## The coefficient of determination has barely changed from before; approx. 97.63% of the  
variability in mean murder numbers is now explained by the two-predictor model. Removing the  
unemployment predictor variable has had little discernable impact on the model's ability to explain  
the variation in the murder numbers. This mirrors the non-significant nature of "unemployment  
rate" when it is included in the model---non-significance implies there is no evidence to suggest  
varying the unemployment rate will change the mean response.
```

```
> #(1)
```

```
>  
predict(murd.fit2,newdata=data.frame(Police=c(300,300),Guns=c(500,0)),interval="confidence",l  
evel=0.99)
```

	fit	lwr	upr
1	23.32948	20.88251	25.77645
2	16.51159	10.90530	22.11787

## Exercise 21.2

The following table presents data collected in one of Galileo’s famous “ball” experiments, in which he rolled a ball down a ramp of different heights and measured how far it traveled from the base of the ramp. For more on this and other interesting examples, look at “Teaching Statistics with Data of Historic Significance” by Dickey and Arnold (1995).

Initial height	Distance
1000	573
800	534
600	495
450	451
300	395
200	337
100	253

- a. Create a data frame in R based on this table and plot the data points with distance on the y-axis.
- b. Galileo believed there was a quadratic relationship between initial height and the distance traveled.
  - i. Fit an order 2 polynomial in height, with distance as the response.
  - ii. Fit a cubic (order 3) and a quartic (order 4) model for these data. What do they tell you about the nature of the relationship?
- c. Based on your models from (b), choose the one that you think best represents the data and plot the fitted line on the raw data. Add 90 percent confidence bands for mean distance traveled to the plot.

The contributed R package *faraway* contains a large number of data sets that accompany a linear regression textbook by Faraway (2005).

Install the package and then call `library("faraway")` to load it. One of the data sets is *trees*, which provides data on the dimensions of felled trees of a certain type (see, for example, Atkinson, 1985).

- d. Access the data object at the prompt and plot volume against girth (the latter along the x-axis).
- e. Fit two models with *Volume* as the response: one quadratic model in *Girth* and the other based on log transformations of both *Volume* and *Girth*. Write down the model equations for each and comment on the similarity (or difference) of the fits in terms of the coefficient of determination and the omnibus F-test.
- f. Use *predict* to add lines to the plot from (d) for each of the two models from (e). Use different line types; add a corresponding legend. Also include 95 percent prediction intervals, with line types matching those of the fitted values (note that for the model that involves log transformation of the response and the predictor, any returned values from *predict* will themselves be on the log scale; you have to back-transform these to the original scale using *exp* before the lines for that model can be superimposed).

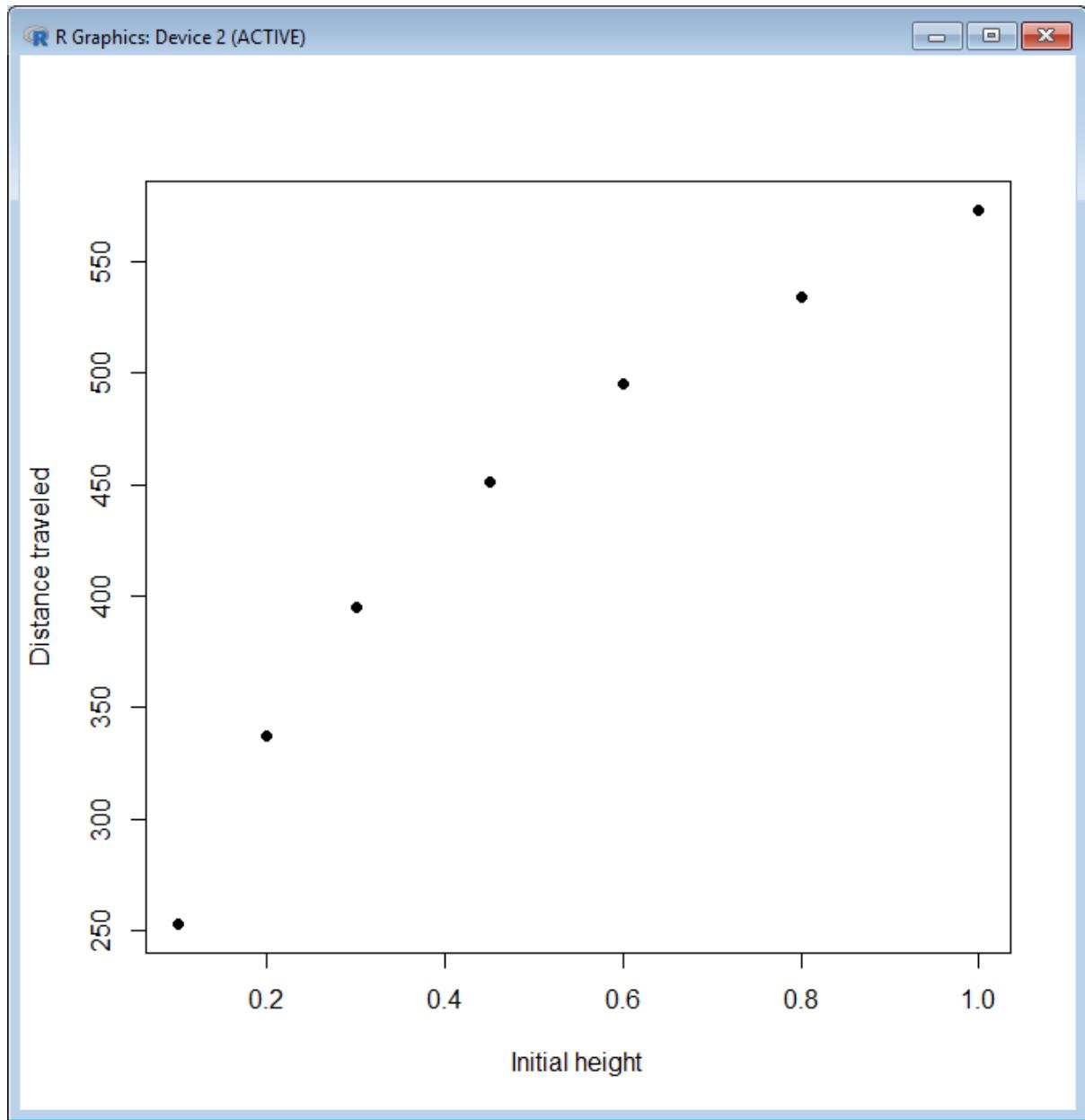
Comment on the respective fits and their estimated prediction intervals.

Lastly, turn your attention back to the mtcars data frame.

- g. Fit and summarize a multiple linear regression model to determine mean MPG from horsepower, weight, and displacement.
- h. In the spirit of Henderson and Velleman (1981), use *I* to refit the model in (g) in terms of GPM = 1/MPG. Which model explains a greater amount of variation in the response?

## Solution 21.2

```
> #(a)  
> gal <- data.frame(d=c(573,534,495,451,395,337,253),h=c(1,0.8,0.6,0.45,0.3,0.2,0.1))  
> plot(gal$d~gal$h,pch=19,xlab="Initial height",ylab="Distance traveled")
```



```
> #(b)  
> ##(i)  
> gal.fit.order2 <- lm(d~h+I(h^2),data=gal)  
> summary(gal.fit.order2)
```

Call:

```
lm(formula = d ~ h + I(h^2), data = gal)
```

Residuals:

1	2	3	4	5	6	7
8.458	-12.607	-6.177	1.940	13.523	9.170	-14.308

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )		
(Intercept)	199.91	16.76	11.928	0.000283 ***		
h	708.32	74.82	9.467	0.000695 ***		
I(h^2)	-343.69	66.78	-5.147	0.006760 **		
---						
Signif. codes:	0 ‘***’	0.001 ‘**’	0.01 ‘*’	0.05 ‘.’	0.1 ‘ ’	1

Residual standard error: 13.64 on 4 degrees of freedom

Multiple R-squared: 0.9903, Adjusted R-squared: 0.9855

F-statistic: 205 on 2 and 4 DF, p-value: 9.333e-05

```
> ##(ii)  
> gal.fit.order3 <- lm(d~h+I(h^2)+I(h^3),data=gal)  
> summary(gal.fit.order3)
```

Call:

```
lm(formula = d ~ h + I(h^2) + I(h^3), data = gal)
```

Residuals:

1	2	3	4	5	6	7
-0.84138	2.32159	-0.08044	-4.46885	1.89175	3.58091	-2.40359

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
--	----------	------------	---------	----------

```

(Intercept) 155.776   8.326 18.710 0.000333 ***
h          1115.298  65.671 16.983 0.000445 ***
I(h^2)     -1244.943 138.425 -8.994 0.002902 **
I(h^3)     547.710   83.273  6.577 0.007150 **

---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

```

Residual standard error: 4.011 on 3 degrees of freedom

Multiple R-squared: 0.9994, Adjusted R-squared: 0.9987  
F-statistic: 1595 on 3 and 3 DF, p-value: 2.662e-05

```

> gal.fit.order4 <- lm(d~h+I(h^2)+I(h^3)+I(h^4),data=gal)
> summary(gal.fit.order4)

```

Call:

`lm(formula = d ~ h + I(h^2) + I(h^3) + I(h^4), data = gal)`

Residuals:

1	2	3	4	5	6	7
0.1708	-0.9279	2.3183	-2.3092	0.2576	0.9338	-0.4433

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	138.295	9.066	15.254	0.00427 **
h	1346.071	106.071	12.690	0.00615 **
I(h^2)	-2116.913	379.271	-5.582	0.03063 *
I(h^3)	1766.391	518.566	3.406	0.07644 .
I(h^4)	-561.014	237.498	-2.362	0.14201
---				
Signif. codes:	0 ‘***’	0.001 ‘**’	0.01 ‘*’	0.05 ‘.’
	0.1 ‘ ’	1		

Residual standard error: 2.523 on 2 degrees of freedom

Multiple R-squared: 0.9998, Adjusted R-squared: 0.9995

F-statistic: 3024 on 4 and 2 DF, p-value: 0.0003306

> # These models reveal that the order 3 model is significant in its highest-order term, and the fit is improved in terms of the coefficient of determination. The same cannot be said for the order 4 model.

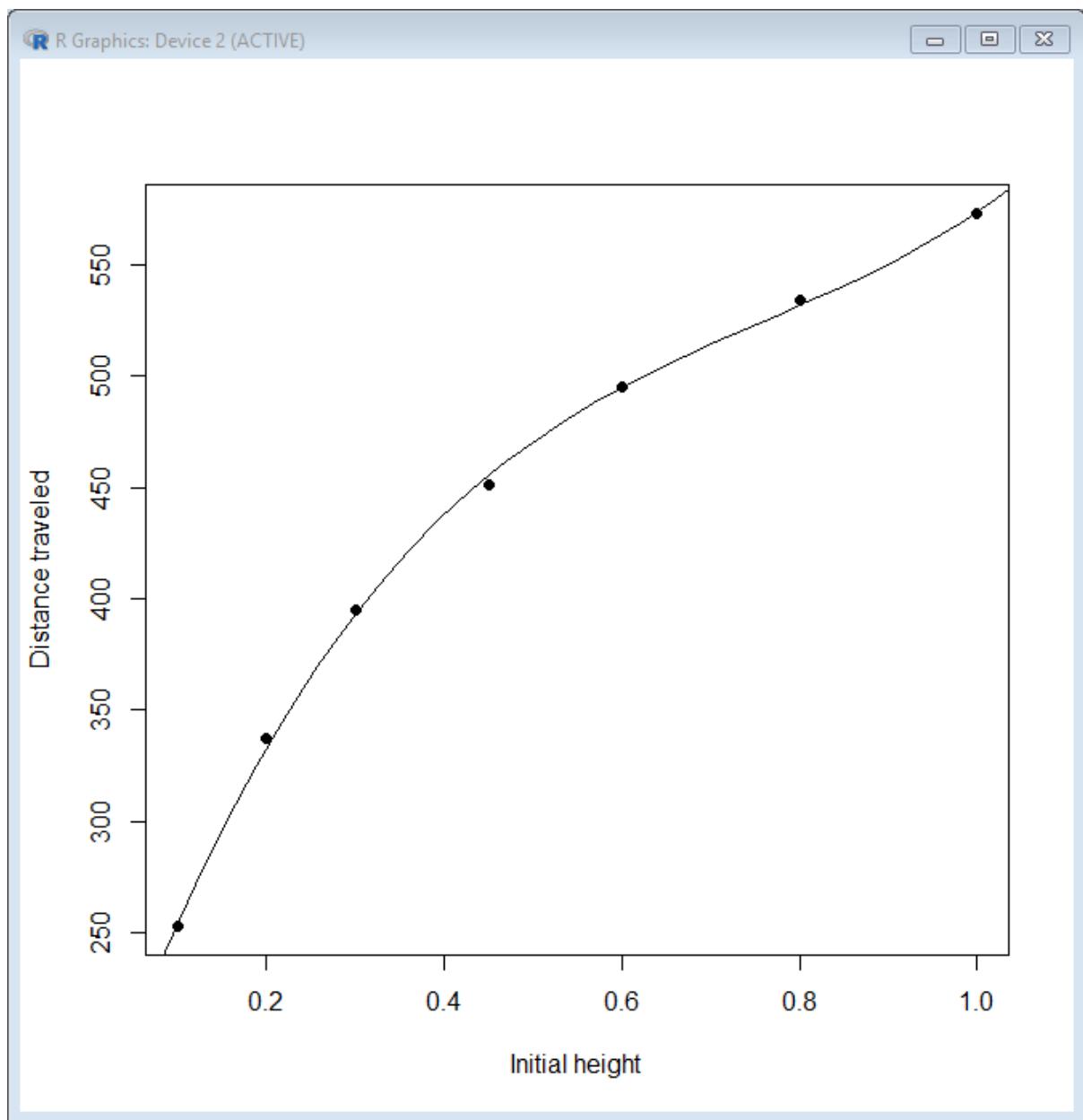
> #(c)

> # Based on the above, out of the three fitted models, the cubic function in height seems preferable -- in other words, the relationship between "distance traveled" and "initial height" therefore appears cubic -- the quadratic model seems too simple, and the quartic model seems unnecessarily complex.

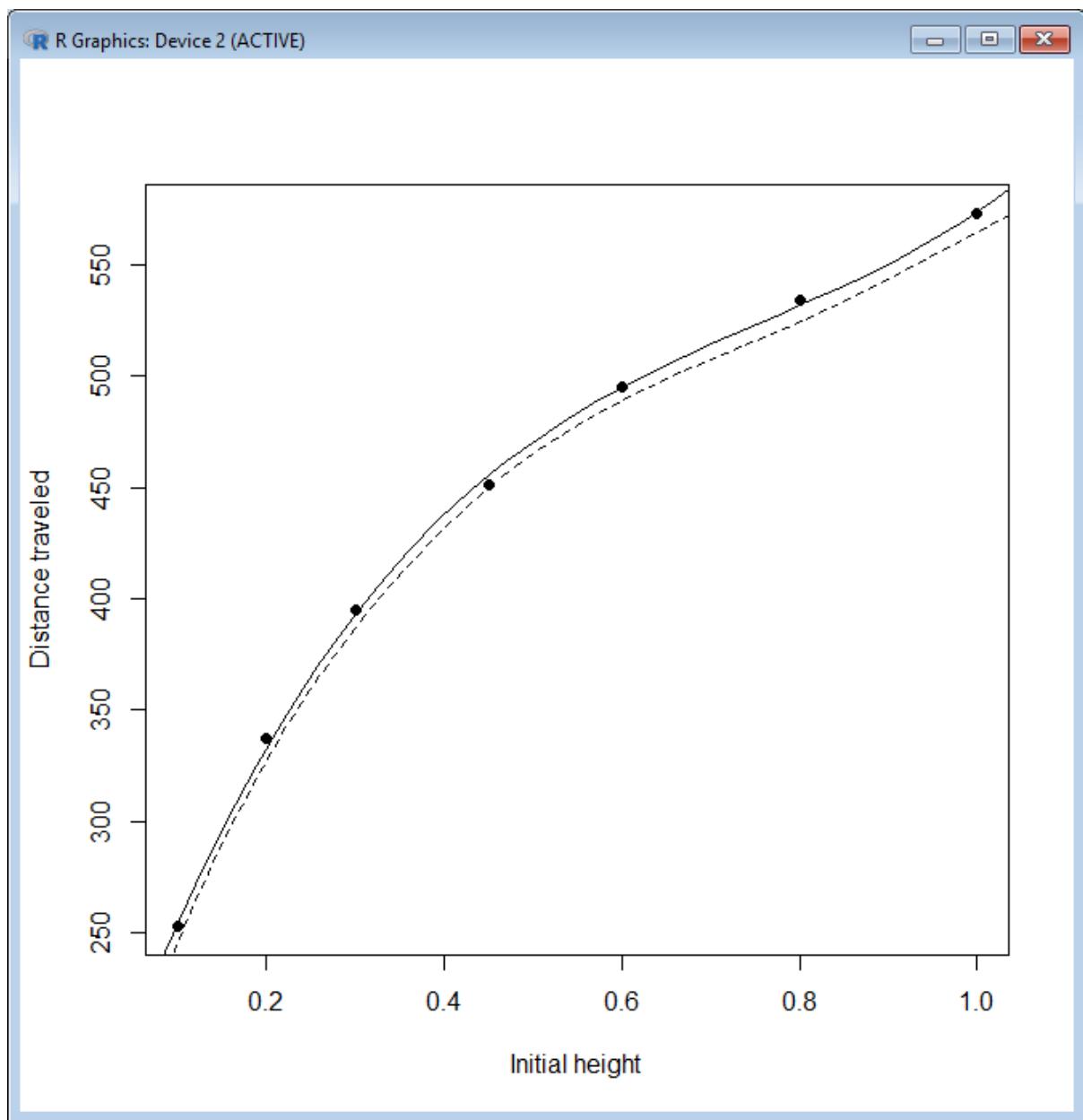
> hseq <- seq(0.05,1.05,length=30)

> gal.pred <- predict(gal.fit.order3,newdata=data.frame(h=hseq),interval="confidence",level=0.9)

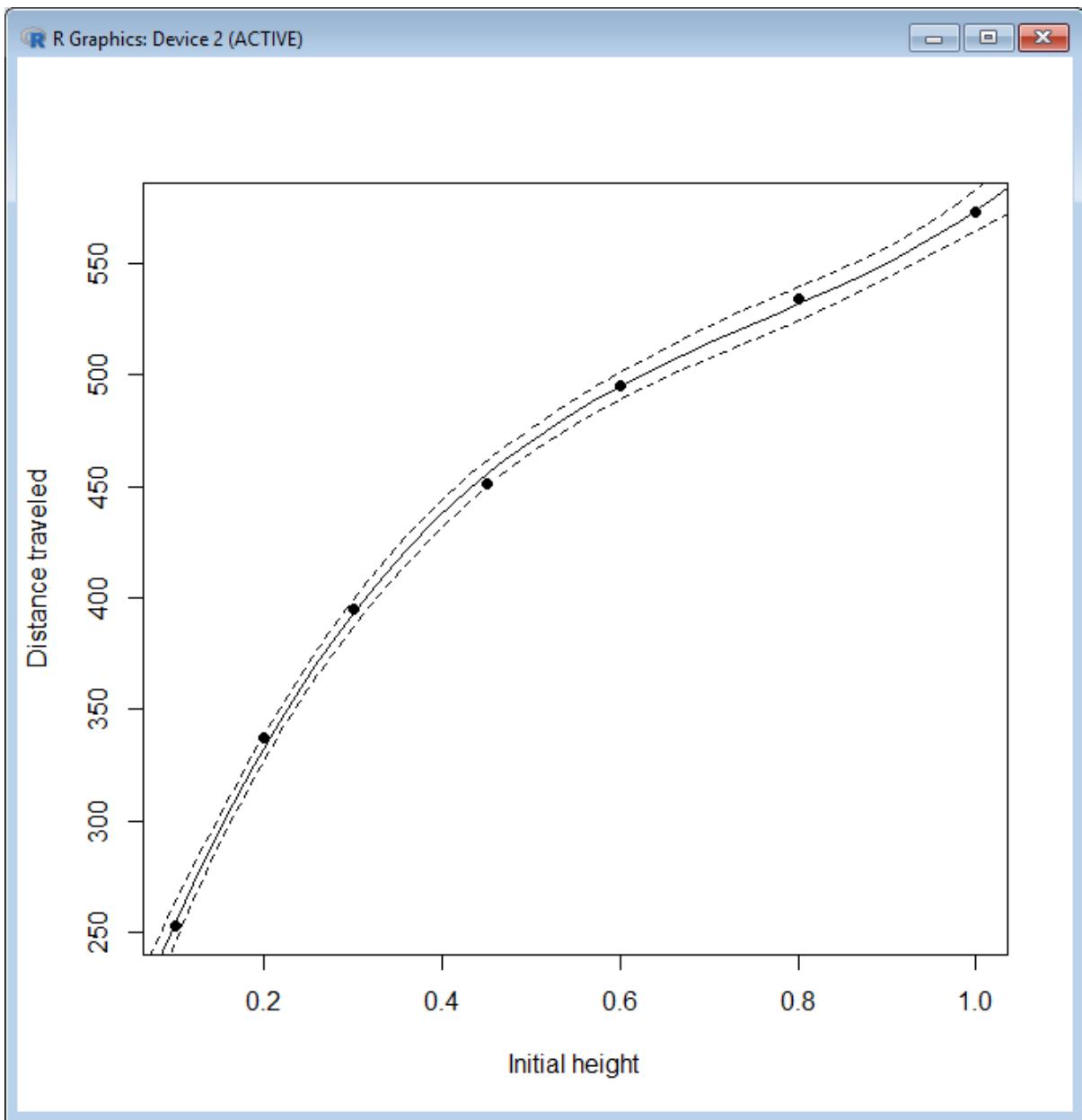
> lines(hseq,gal.pred[,1])



```
> lines(hseq,gal.pred[,2],lty=2)
```



```
> lines(hseq,gal.pred[,3],lty=2)
```



```
> #(d)  
> library("faraway")  
> ?trees
```

## Girth, Height and Volume for Black Cherry Trees

### Description

This data set provides measurements of the girth, height and volume of timber in 31 felled black cherry trees. Note that girth is the diameter of the tree (in inches) measured at 4 ft 6 in above the ground.

### Usage

```
trees
```

### Format

A data frame with 31 observations on 3 variables.

```
[,1] Girth  numeric Tree diameter in inches  
[,2] Height numeric Height in ft  
[,3] Volume numeric Volume of timber in cubic ft
```

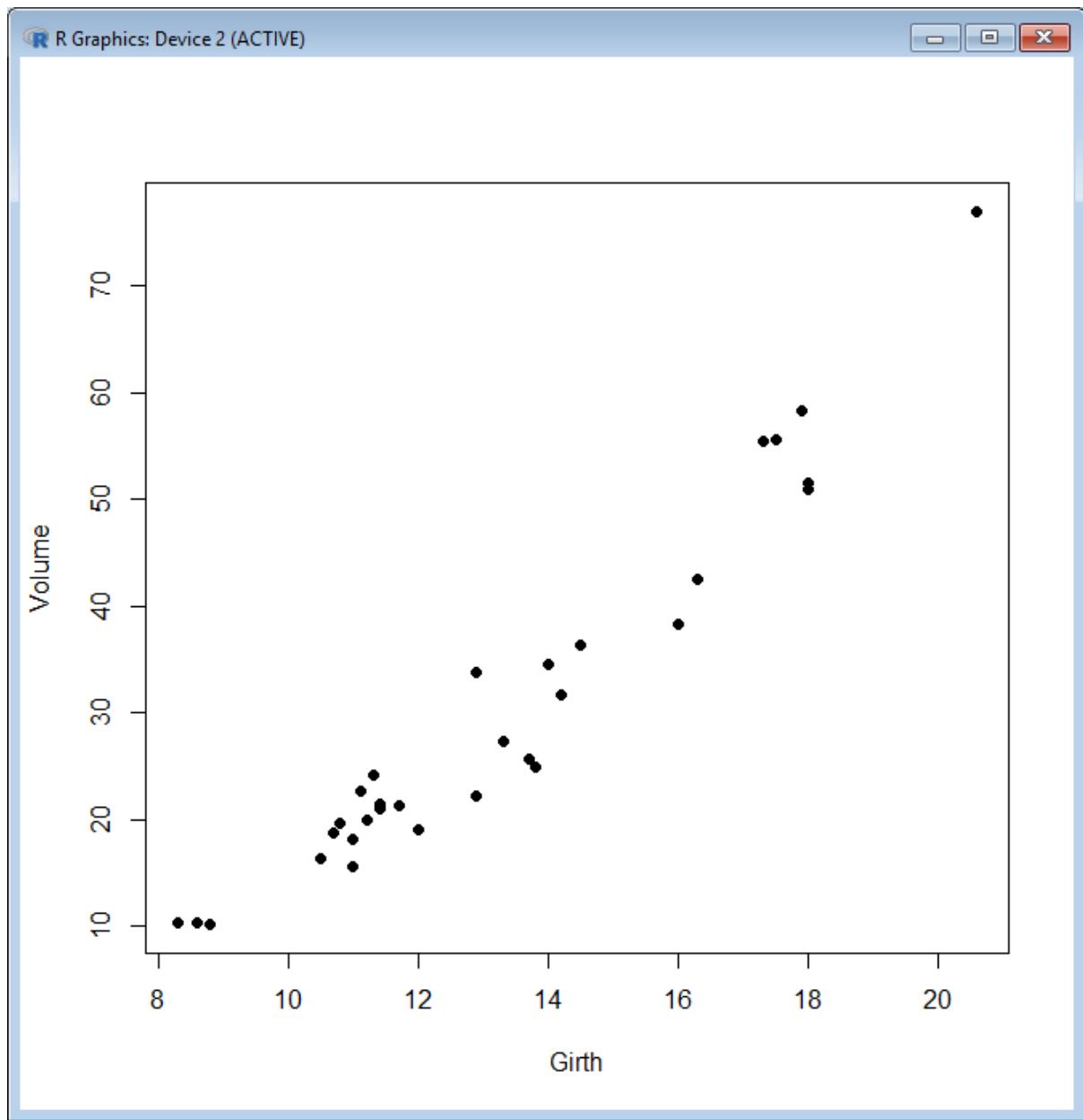
### Source

Ryan, T. A., Joiner, B. L. and Ryan, B. F. (1976) *The Minitab Student Handbook*. Duxbury Press.

### References

Atkinson, A. C. (1985) *Plots, Transformations and Regression*. Oxford University Press.

```
> plot(trees$Volume~trees$Girth,pch=19,xlab="Girth",ylab="Volume")
```



```
> #(e)
> tree.fit1 <- lm(Volume~Girth+I(Girth^2),trees)
> summary(tree.fit1) ## "Mean volume" = 10.79 - 2.09*"girth" + 0.254*"girth^2"
```

Call:

```
lm(formula = Volume ~ Girth + I(Girth^2), data = trees)
```

Residuals:

Min	1Q	Median	3Q	Max
-5.4889	-2.4293	-0.3718	2.0764	7.6447

Coefficients:

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

(Intercept) 10.78627 11.22282 0.961 0.344728

Girth -2.09214 1.64734 -1.270 0.214534

I(Girth^2) 0.25454 0.05817 4.376 0.000152 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.335 on 28 degrees of freedom

Multiple R-squared: 0.9616, Adjusted R-squared: 0.9588

F-statistic: 350.5 on 2 and 28 DF, p-value: < 2.2e-16

```
> tree.fit2 <- lm(log(Volume)~log(Girth),trees)
```

```
> summary(tree.fit2) ## "Mean log(volume)" = -2.35 + 2.20*log(girth)"
```

Call:

```
lm(formula = log(Volume) ~ log(Girth), data = trees)
```

Residuals:

Min 1Q Median 3Q Max

-0.205999 -0.068702 0.001011 0.072585 0.247963

Coefficients:

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

(Intercept) -2.35332 0.23066 -10.20 4.18e-11 \*\*\*

log(Girth) 2.19997 0.08983 24.49 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

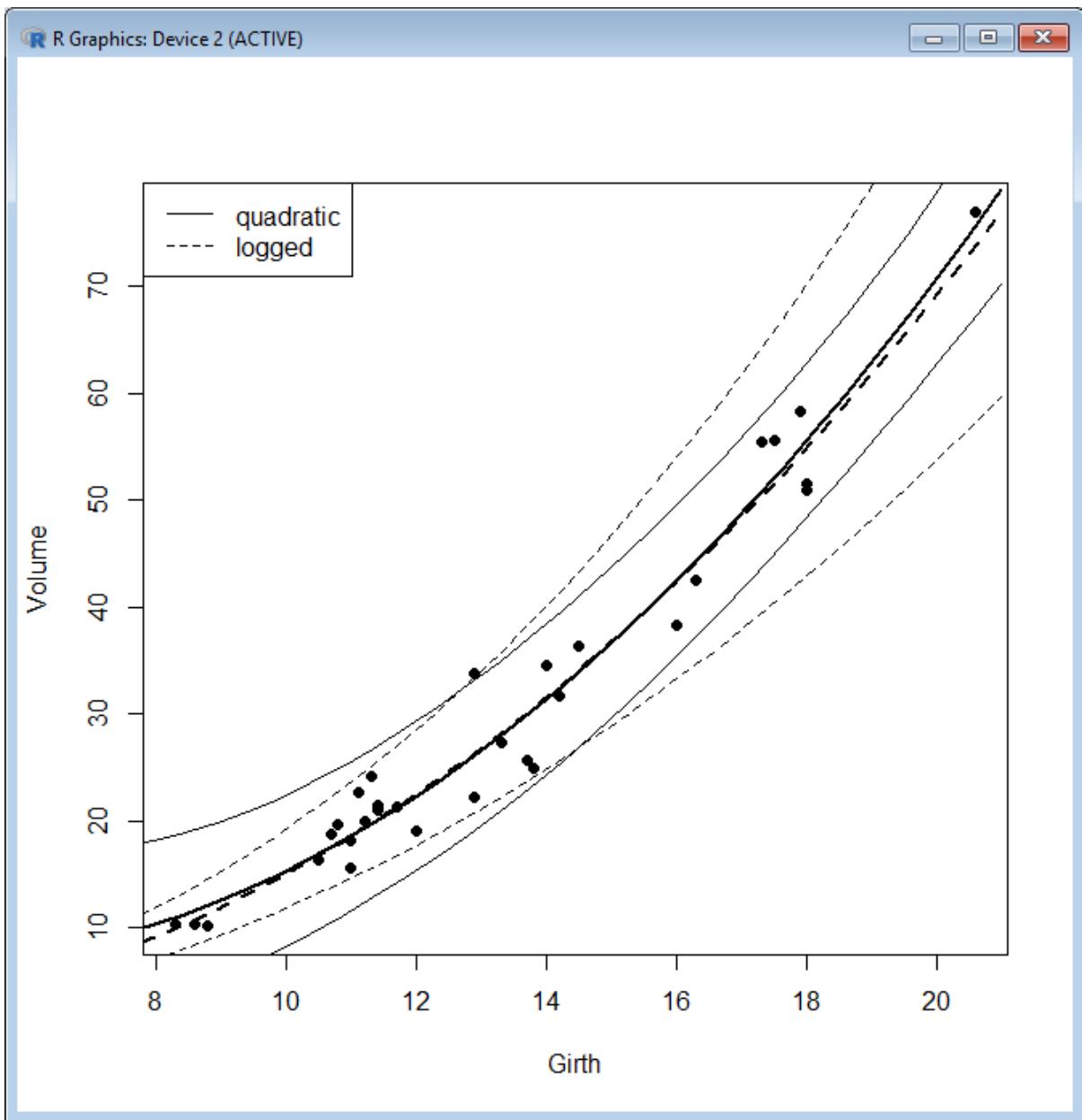
Residual standard error: 0.115 on 29 degrees of freedom

Multiple R-squared: 0.9539, Adjusted R-squared: 0.9523

F-statistic: 599.7 on 1 and 29 DF, p-value: < 2.2e-16

> # Coefficients of determination are similar; the quadratic model is slightly higher. Both indicate a statistically significant positive effect of girth on volume. Based on the F-test, both models are clearly better than fitting an intercept alone (i.e. girth does appear to be able to explain (mean) volume very well).

```
> #(f)
> gseq <- seq(7,21,length=30)
> tree.pred1 <- predict(tree.fit1,newdata=data.frame(Girth=gseq),interval="prediction")
> tree.pred2 <- predict(tree.fit2,newdata=data.frame(Girth=gseq),interval="prediction")
> plot(trees$Volume~trees$Girth,pch=19,xlab="Girth",ylab="Volume")
> lines(gseq,tree.pred1[,1],lwd=2)
> lines(gseq,exp(tree.pred1[,2]))
> lines(gseq,exp(tree.pred1[,3]))
> lines(gseq,exp(tree.pred2[,1]),lwd=2,lty=2)
> lines(gseq,exp(tree.pred2[,2]),lty=2)
> lines(gseq,exp(tree.pred2[,3]),lty=2)
> legend("topleft",legend=c("quadratic","logged"),lty=1:2)
```



> # The fitted values of the models themselves are extremely similar. However, the prediction intervals tell a different story. Notably, the quadratic model has far wider limits for small girth values than for larger ones. On the other hand, the limits for the logged model are substantially wider than those of the quadratic model at larger girth values. Which model is 'better'...? It's very difficult to answer that without further information...

```

> #(g)
> library("MASS")
> car.fit <- lm(mpg~wt+hp+disp,data=mtcars)
> summary(car.fit)

```

Call:

`lm(formula = mpg ~ wt + hp + disp, data = mtcars)`

Residuals:

Min	1Q	Median	3Q	Max
-3.891	-1.640	-0.172	1.061	5.861

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )		
(Intercept)	37.105505	2.110815	17.579	< 2e-16 ***		
wt	-3.800891	1.066191	-3.565	0.00133 **		
hp	-0.031157	0.011436	-2.724	0.01097 *		
disp	-0.000937	0.010350	-0.091	0.92851		
---						
Signif. codes:	0 ‘***’	0.001 ‘**’	0.01 ‘*’	0.05 ‘.’	0.1 ‘ ’	1

Residual standard error: 2.639 on 28 degrees of freedom

Multiple R-squared: 0.8268, Adjusted R-squared: 0.8083

F-statistic: 44.57 on 3 and 28 DF, p-value: 8.65e-11

```
> #(h)
> car.fit <- lm(I(1/mpg)~wt+hp+disp,data=mtcars)
> summary(car.fit)
```

Call:

lm(formula = I(1/mpg) ~ wt + hp + disp, data = mtcars)

Residuals:

Min	1Q	Median	3Q	Max
-0.0163719	-0.0043511	0.0008672	0.0032544	0.0133345

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
--	----------	------------	---------	----------

```
(Intercept) 9.496e-03 5.322e-03 1.784 0.08521 .
wt         9.469e-03 2.688e-03 3.522 0.00149 **
hp         5.864e-05 2.883e-05 2.034 0.05155 .
disp       2.456e-05 2.609e-05 0.941 0.35472
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Residual standard error: 0.006653 on 28 degrees of freedom

Multiple R-squared: 0.8518, Adjusted R-squared: 0.8359

F-statistic: 53.63 on 3 and 28 DF, p-value: 9.94e-12

> # Both fits to the mtcars data here provide similar levels of significance for the three predictors; though there is a mild yet noticeable improvement in the coefficient of determination for the latter model based on a response of GPM = 1/MPG.

>

## Exercise 21.3

Return your attention to the *cats* data frame in package *MASS*. In the first few problems in Exercise 21.1, you fitted the main-effect-only model to predict the heart weights of domestic cats by total body weight and sex.

- a. Fit the model again, and this time include an interaction between the two predictors. Inspect the model summary.

What do you notice in terms of the parameter estimates and their significance when compared to the earlier main-effect-only version?

- b. Produce a scatterplot of heart weight on body weight, using different point characters or colors to distinguish the observations according to sex. Use *abline* to add two lines denoting the fitted model. How does this plot differ from the one in Exercise 21.1 (d)?
- c. Predict the heart weight of Tilman's cat using the new model (remember that Sigma is a 3.4 kg female) accompanied by a 95 percent prediction interval. Compare it to the main-effects-only model from the earlier exercise.

In Exercise 21.2, you accessed the *trees* data frame in the contributed *faraway* package. After loading the package, access the *?trees* help file; you'll find the volume and girth measurements you used earlier, as well as data on the height of each tree.

- d. Without using any transformations of the data, fit and inspect a main-effects-only model for predicting volume from girth and height. Then, fit and inspect a second version of this model including an interaction.

- e. Repeat (d), but this time use the log transformation of all variables.

What do you notice about the significance of the interaction between the untransformed and transformed models? What does this suggest about the relationships in the data?

Turn back to the *mtcars* data set and remind yourself of the variables in the help file *?mtcars*.

- f. Fit a linear model for *mpg* based on a two-way interaction between *hp* and *factor(cyl)* and their main effects, as well as a main effect for *wt*. Produce a summary of the fit.
- g. Interpret the estimated coefficients for the interaction between horsepower and the (categorical) number of cylinders.
- h. Suppose you're keen on purchasing a 1970s performance car. Your mother advises you to purchase a "practical and economical" car that's capable of an average MPG value of at least 25. You see three vehicles advertised: car 1 is a four-cylinder, 100 horsepower car that weighs 2100 lbs; car 2 is an eight-cylinder, 210 horsepower car that weighs 3900 lbs; and car 3 is a six-cylinder, 200 horsepower car that weighs 2900 lbs.
  - i. Use your model to predict the mean MPG for each of the three cars; provide 95 percent confidence intervals. Based on your point estimates only, which car would you propose to your mother?
  - ii. You still want the most gas-guzzling car you can own with your mother's blessing, so you decide to be sneaky and base your decision on what the confidence intervals tell you instead. Does this change your choice of vehicle?

## Solution 21.3

```
> library("MASS")
```

```
> #(a)
```

```
> cat.fit <- lm(Hwt~Bwt*Sex,data=cats)
```

```
> summary(cat.fit)
```

Call:

```
lm(formula = Hwt ~ Bwt * Sex, data = cats)
```

Residuals:

Min	1Q	Median	3Q	Max
-3.7728	-1.0118	-0.1196	0.9272	4.8646

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )		
(Intercept)	2.9813	1.8428	1.618	0.107960		
Bwt	2.6364	0.7759	3.398	0.000885 ***		
SexM	-4.1654	2.0618	-2.020	0.045258 *		
Bwt:SexM	1.6763	0.8373	2.002	0.047225 *		
---						
Signif. codes:	0 ‘***’	0.001 ‘**’	0.01 ‘*’	0.05 ‘.’	0.1 ‘ ’	1

Residual standard error: 1.442 on 140 degrees of freedom

Multiple R-squared: 0.6566, Adjusted R-squared: 0.6493

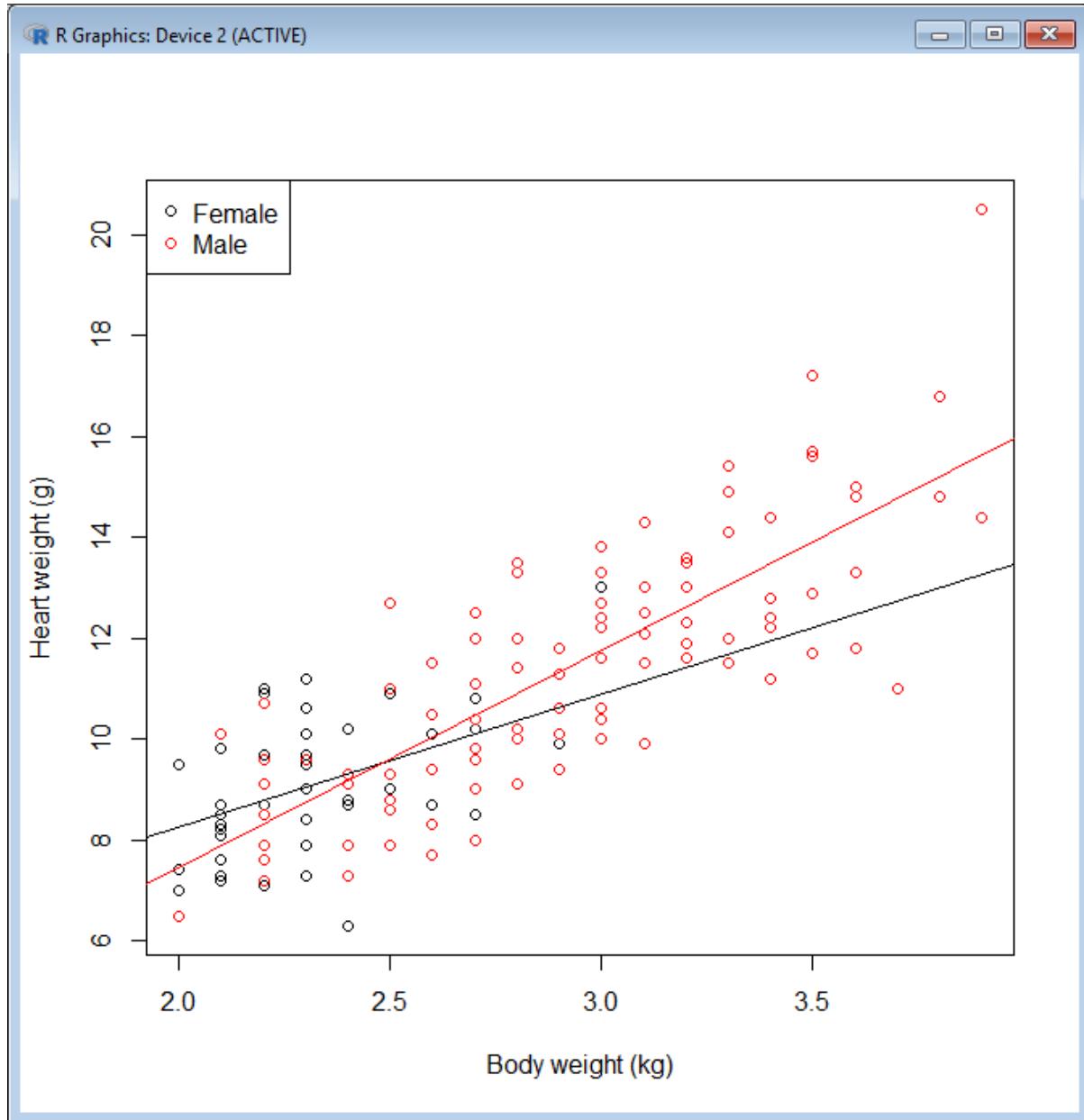
F-statistic: 89.24 on 3 and 140 DF, p-value: < 2.2e-16

> # The main-effects-only version of the model had a mild negative effect of "sex male", and it was not significant. In this version, the effect of being male is more extreme, and the p-value is far smaller, now providing weak evidence of significance. The interactive term for the slope of 'Bwt' for a male is reduced somewhat compared to females (estimated parameter is negative).

```

> #(b)
> plot(cats$Bwt,cats$Hwt,col=cats$Sex,ylab="Heart weight (g)",xlab="Body weight (kg)")
> legend("topleft",legend=c("Female","Male"),col=1:2,pch=1)
> cat.coefs <- coef(cat.fit)
> abline(coef=cat.coefs[1:2])
> abline(coef=c(sum(cat.coefs[c(1,3)]),sum(cat.coefs[c(2,4)])),col=2)

```



```

> # Lines of the fitted model are no longer parallel; the effect of the weakly significant interaction
is apparent.

```

```

> #(c)

> predict(cat.fit,newdata=data.frame(Bwt=3.4,Sex="F"),interval="prediction",level=0.95)
    fit     lwr     upr
1 11.94512 8.651786 15.23845

```

> # Sigma's heart weight predicted from the new model is around 1.5 grams lighter than predicted from the main-effects-only model in the earlier exercise. The prediction interval is set accordingly lower as well, but is also wider than the interval from earlier.

```

> #(d)

> library("faraway")

> tree.fit1 <- lm(Volume~Girth+Height,data=trees)

> summary(tree.fit1)

```

Call:

```
lm(formula = Volume ~ Girth + Height, data = trees)
```

Residuals:

Min	1Q	Median	3Q	Max
-6.4065	-2.6493	-0.2876	2.2003	8.4847

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-57.9877	8.6382	-6.713	2.75e-07 ***
Girth	4.7082	0.2643	17.816	< 2e-16 ***
Height	0.3393	0.1302	2.607	0.0145 *
	---			
Signif. codes:	0 ‘***’	0.001 ‘**’	0.01 ‘*’	0.05 ‘.’
	0.1 ‘ ’	1		

Residual standard error: 3.882 on 28 degrees of freedom

Multiple R-squared: 0.948, Adjusted R-squared: 0.9442

F-statistic: 255 on 2 and 28 DF, p-value: < 2.2e-16

```
> tree.fit2 <- lm(Volume~Girth*Height,data=trees)
```

```
> summary(tree.fit2)
```

Call:

```
lm(formula = Volume ~ Girth * Height, data = trees)
```

Residuals:

Min	1Q	Median	3Q	Max
-6.5821	-1.0673	0.3026	1.5641	4.6649

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	69.39632	23.83575	2.911	0.00713 **
Girth	-5.85585	1.92134	-3.048	0.00511 **
Height	-1.29708	0.30984	-4.186	0.00027 ***
Girth:Height	0.13465	0.02438	5.524	7.48e-06 ***
---				
Signif. codes:	0 ‘***’	0.001 ‘**’	0.01 ‘*’	0.05 ‘.’
	0.1 ‘ ’	1		

Residual standard error: 2.709 on 27 degrees of freedom

Multiple R-squared: 0.9756, Adjusted R-squared: 0.9728

F-statistic: 359.3 on 3 and 27 DF, p-value: < 2.2e-16

```
> #(e)
```

```
> tree.fit3 <- lm(log(Volume)~log(Girth)+log(Height),data=trees)
```

```
> summary(tree.fit3)
```

Call:

```
lm(formula = log(Volume) ~ log(Girth) + log(Height), data = trees)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.168561	-0.048488	0.002431	0.063637	0.129223

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-6.63162	0.79979	-8.292	5.06e-09 ***
log(Girth)	1.98265	0.07501	26.432	< 2e-16 ***
log(Height)	1.11712	0.20444	5.464	7.81e-06 ***
---				
Signif. codes:	0 ‘***’	0.001 ‘**’	0.01 ‘*’	0.05 ‘.’
	0.1 ‘ ’	1		

Residual standard error: 0.08139 on 28 degrees of freedom

Multiple R-squared: 0.9777, Adjusted R-squared: 0.9761

F-statistic: 613.2 on 2 and 28 DF, p-value: < 2.2e-16

```
> tree.fit4 <- lm(log(Volume)~log(Girth)*log(Height),data=trees)
> summary(tree.fit4)
```

Call:

lm(formula = log(Volume) ~ log(Girth) \* log(Height), data = trees)

Residuals:

Min	1Q	Median	3Q	Max
-0.165941	-0.048613	0.006384	0.062204	0.132295

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-3.6869	7.6996	-0.479	0.636
log(Girth)	0.7942	3.0910	0.257	0.799
log(Height)	0.4377	1.7788	0.246	0.808
log(Girth):log(Height)	0.2740	0.7124	0.385	0.704

Residual standard error: 0.08265 on 27 degrees of freedom

Multiple R-squared: 0.9778, Adjusted R-squared: 0.9753

F-statistic: 396.4 on 3 and 27 DF, p-value: < 2.2e-16

> # The interactive effect is highly significant in the untransformed model from (d), but completely non-significant after log-transformation of all present variables. This suggests that 'straight-line' relationships are not the best way to model these data (you can experiment with plots if you wish), and that we must account for curvature in the response surface by either working with transformed data or including the two-way interaction between the two untransformed continuous predictors. Once more, it is difficult to decide on which approach ought to be preferred -- we need to know more about the nature of the data themselves in context, as well as the ultimate purpose of the fitted model.

> #(f)

```
> car.fit <- lm(mpg~factor(cyl)*hp+wt,data=mtcars)
> summary(car.fit)
```

Call:

lm(formula = mpg ~ factor(cyl) \* hp + wt, data = mtcars)

Residuals:

Min	1Q	Median	3Q	Max
-3.1864	-1.4098	-0.4022	1.0186	4.3920

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	41.87732	3.23293	12.953	1.37e-12 ***
factor(cyl)6	-9.98213	5.76950	-1.730	0.095931 .
factor(cyl)8	-11.72793	4.22507	-2.776	0.010276 *
hp	-0.09947	0.03487	-2.853	0.008576 **
wt	-3.05994	0.68275	-4.482	0.000143 ***
factor(cyl)6:hp	0.07809	0.05236	1.492	0.148335
factor(cyl)8:hp	0.08602	0.03703	2.323	0.028601 *

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.3 on 25 degrees of freedom

Multiple R-squared: 0.8826, Adjusted R-squared: 0.8544

F-statistic: 31.32 on 6 and 25 DF, p-value: 1.831e-10

```
> #(g)
```

```
> coef(car.fit)
```

(Intercept)	factor(cyl)6	factor(cyl)8	hp	wt	factor(cyl)6:hp	factor(cyl)8:hp
41.87732085	-9.98213264	-11.72792959	-0.09946598	-3.05993524	0.07808919	
0.08602496						

> # The interactive effect is between a continuous (hp) and a categorical (factor(cyl)) predictor. As such, each of the two estimated coefficients can be interpreted as the change in the slope of hp for each of the non-reference levels of factor(cyl).

> coef(car.fit)[4] # When the car has 4 cylinders (reference level), the slope for hp is -0.0995 (to 4 decimal places)

```
hp
```

```
-0.09946598
```

> coef(car.fit)[4] + coef(car.fit)[6] # When the car has 6 cylinders, the slope for hp is -0.0995 + 0.0781 = -0.0214 (to 4 decimal places)

```
hp
```

```
-0.02137679
```

> coef(car.fit)[4] + coef(car.fit)[7] # When the car has 8 cylinders, the slope for hp is -0.0995 + 0.0860 = -0.0135 (to 4 decimal places)

```
hp
```

```
-0.01344103
```

> # This model suggests that as hp increases, mean MPG decreases (for a fixed wt). However, in comparison to 4-cylinder cars, mean MPG is estimated to decrease at a slower rate with an increasing hp for both 6- and 8-cylinder cars (since the positive additive terms to the baseline slope of -0.0995 still provide negative slopes in hp, but ones that are closer to zero).

```
> #(h)
```

```
> ##(i)
```

```

>
predict(car.fit,newdata=data.frame(wt=c(2.1,3.9,2.9),hp=c(100,210,200),cyl=c(4,8,6)),interval="confidence",level=0.95)

  fit    lwr    upr
1 25.50486 23.57668 27.43304
2 15.39303 14.11928 16.66678
3 18.74602 12.29560 25.19644

> # The first car is the only car that has a point estimate of mean MPG that is higher than your
  mother's demand of 25, so this would be the initial choice.

> ##(ii)

> # Although the point estimate for Car 3 is much less than 25, looking at the confidence intervals
  you can see that the interval for Car 3 includes 25. So, you could argue to your mother that you're
  95% confident that the true mean MPG of a car like Car 3 lies somewhere in that interval; in
  particular, your model suggests no evidence against the hypothesis that the true mean MPG of
  such a car is equal to 25. Of course, the interval also includes possible true values that are far
  worse than 25... but you don't need to tell your mother that.

```

# Chapter 22: Linear Model Selection and Diagnostics

Estimated time to complete: **30 minutes**

## Exercise 22.1

In Sections 22.2.2 and 22.2.3, you used forward and backward selection approaches to find a model for predicting the cost of the construction of nuclear power plants (based on the *nuclear* data frame in the *boot* package).

- a. Using the same fullest model (in other words, main effects of all present predictors only), use stepwise AIC selection to find a suitable model for the data.
- b. Does the final model found in (a) match either of the models resulting from the earlier use of forward and backward selection?

How does it differ?

Exercise 21.2 on page 512 detailed Galileo’s ball data. Enter these as a data frame in your current R workspace if you haven’t already.

- c. Fit five linear models to these data with distance as the response—an intercept-only model and four separate polynomial models of increasing order 1 to 4 in height.
- d. Construct a table of partial F-tests to identify your favored model for distance traveled. Is your selection consistent with Exercise 21.2 (b) and (c)?

You first encountered the *diabetes* data frame in the contributed *faraway* package in Section 21.5.2, where you modeled the mean total cholesterol. Load the package and inspect the documentation in *?diabetes* to refresh your memory of the data set.

- e. There are some missing values in *diabetes* that might interfere with model selection algorithms. Define a new version of the *diabetes* data frame that deletes all rows with a missing value in any of the following variables: *chol*, *age*, *gender*, *height*, *weight*, *frame*, *waist*, *hip*, *location*. Hint: Use *na.omit* or your knowledge of record extraction or deletion for a data frame. You can create the required vector of row numbers to be extracted or deleted using *which* and *is.na*, or you can try using the *complete.cases* function to obtain a logical flag vector—inspect its help file for details.
- f. Use your data frame from (e) to fit two linear models with *chol* as the response. The null model object, named *dia.null*, should be an intercept-only model. The full model object, named *dia.full*, should be the overly complex model with a four-way interaction (and all lower-order terms) among *age*, *gender*, *weight*, and *frame*; a three-way interaction (and all lower-order terms) among *waist*, *height*, and *hip*; and a main effect for *location*.
- g. Starting from *dia.null* and using the same terms as in *dia.full* for *scope*, implement stepwise selection by AIC to choose a model for mean total cholesterol and then summarize.
- h. Use forward selection based on partial F-tests with a conventional significance level of  $\alpha = 0.05$  to choose a model, again starting from *dia.null*. Is the result here the same as the model arrived at in (g)?
  - i. Stepwise selection doesn’t have to start from the simplest model.
  - ii. Repeat (g), but this time, set *dia.full* to be the starting model (you don’t need to supply anything to *scope* if you’re starting from the most complex model). What is the final model selected via AIC if you start from *dia.full*? Is it different than the final model from (g)? Why is this or is this not the case, do you think?

Revisit the ubiquitous *mtcars* data frame from the *MASS* package.

j. In Section 22.2.4, you used stepwise AIC selection to model mean MPG. The selected model included a main effect for *qsec*. Rerun the same AIC selection process, but this time, do it in terms of GPM=1/MPG. Does this change the complexity of the final model?

## Solution 22.1

```
> #(a)  
> library("boot")  
> nuc.null <- lm(cost~1,data=nuclear)  
> nuc.step <- step(nuc.null,scope=~.+date+t1+t2+cap+pr+ne+ct+bw+cum.n+pt,direction="both")  
Start: AIC=329.72  
cost ~ 1
```

	Df	Sum of Sq	RSS	AIC
+ date	1	334335	562837	316.80
+ pt	1	305334	591839	318.41
+ cap	1	199673	697499	323.66
+ t1	1	186984	710189	324.24
+ ne	1	128641	768531	326.77
+ cum.n	1	67938	829234	329.20
<none>		897172		329.72
+ ct	1	43042	854130	330.15
+ bw	1	16205	880967	331.14
+ pr	1	9037	888136	331.40
+ t2	1	27	897145	331.72

Step: AIC=316.8

cost ~ date

	Df	Sum of Sq	RSS	AIC
+ cap	1	189732	373105	305.64
+ ne	1	92256	470581	313.07

```

+ pt   1  90587 472250 313.18
+ t2   1  68161 494676 314.67
+ ct   1  54794 508043 315.52
<none>          562837 316.80
+ t1   1  15322 547515 317.92
+ cum.n 1  4658 558179 318.53
+ pr   1  4027 558810 318.57
+ bw   1  1240 561597 318.73
- date 1  334335 897172 329.72

```

Step: AIC=305.64

cost ~ date + cap

	Df	Sum of Sq	RSS	AIC
+ pt	1	95917	277189	298.13
+ ne	1	94537	278569	298.29
+ ct	1	48957	324148	303.14
+ cum.n	1	28062	345044	305.14
<none>		373105	305.64	
+ pr	1	18453	354652	306.02
+ t2	1	13356	359750	306.48
+ bw	1	7505	365601	306.99
+ t1	1	957	372148	307.56
- cap	1	189732	562837	316.80
- date	1	324394	697499	323.66

Step: AIC=298.13

cost ~ date + cap + pt

	Df	Sum of Sq	RSS	AIC
+ ne	1	54572	222617	293.12

	Df	Sum of Sq	RSS	AIC
+ ct	1	18119	259069	297.97
<none>		277189	298.13	
+ t2	1	10429	266760	298.91
+ pr	1	6017	271171	299.43
+ bw	1	654	276535	300.06
+ cum.n	1	463	276726	300.08
+ t1	1	12	277177	300.13
- pt	1	95917	373105	305.64
- date	1	111397	388586	306.94
- cap	1	195062	472250	313.19

Step: AIC=293.12

cost ~ date + cap + pt + ne

	Df	Sum of Sq	RSS	AIC
+ t2	1	19230	203387	292.23
+ ct	1	15765	206852	292.77
+ cum.n	1	13820	208797	293.07
<none>		222617	293.12	
+ pr	1	5231	217386	294.36
+ bw	1	448	222169	295.06
+ t1	1	107	222510	295.10
- ne	1	54572	277189	298.13
- pt	1	55952	278569	298.29
- date	1	119333	341950	304.85
- cap	1	195756	418373	311.31

Step: AIC=292.23

cost ~ date + cap + pt + ne + t2

	Df	Sum of Sq	RSS	AIC
--	----	-----------	-----	-----

	Df	Sum of Sq	RSS	AIC
+ pr	1	23244	180143	290.35
+ cum.n	1	12951	190436	292.12
<none>		203387	292.23	
+ ct	1	10212	193175	292.58
- t2	1	19230	222617	293.12
+ bw	1	860	202527	294.09
+ t1	1	368	203019	294.17
- pt	1	50299	253686	297.30
- ne	1	63373	266760	298.91
- cap	1	132791	336178	306.31
- date	1	138352	341739	306.83

Step: AIC=290.34

cost ~ date + cap + pt + ne + t2 + pr

	Df	Sum of Sq	RSS	AIC
+ cum.n	1	20987	159156	288.38
<none>		180143	290.35	
+ t1	1	6495	173648	291.17
+ bw	1	5898	174245	291.28
+ ct	1	4828	175316	291.48
- pr	1	23244	203387	292.23
- pt	1	32754	212898	293.69
- t2	1	37243	217386	294.36
- ne	1	67232	247375	298.49
- cap	1	131615	311758	305.90
- date	1	158588	338731	308.55

Step: AIC=288.38

cost ~ date + cap + pt + ne + t2 + pr + cum.n

	Df	Sum of Sq	RSS	AIC
- pt	1	4066	163222	287.19
+ ct	1	11794	147362	287.92
<none>		159156	288.38	
+ t1	1	3103	156053	289.75
+ bw	1	2806	156350	289.81
- cum.n	1	20987	180143	290.35
- pr	1	31280	190436	292.12
- t2	1	40640	199796	293.66
- ne	1	86979	246135	300.33
- date	1	150115	309271	307.64
- cap	1	150213	309369	307.65

Step: AIC=287.19

cost ~ date + cap + ne + t2 + pr + cum.n

	Df	Sum of Sq	RSS	AIC
+ ct	1	15200	148021	286.06
<none>		163222	287.19	
+ bw	1	4884	158337	288.22
+ pt	1	4066	159156	288.38
+ t1	1	2452	160770	288.70
- pr	1	42270	205492	292.56
- t2	1	47075	210297	293.30
- cum.n	1	49676	212898	293.69
- ne	1	127488	290709	303.66
- cap	1	157450	320672	306.80
- date	1	374577	537799	323.34

Step: AIC=286.06

cost ~ date + cap + ne + t2 + pr + cum.n + ct

	Df	Sum of Sq	RSS	AIC
<none>		148021	286.06	
- ct	1	15200	163222	287.19
+ t1	1	2853	145168	287.44
+ bw	1	1661	146360	287.70
+ pt	1	660	147362	287.92
- pr	1	27209	175230	289.46
- t2	1	28539	176560	289.70
- cum.n	1	53288	201310	293.90
- ne	1	114114	262135	302.35
- cap	1	162846	310867	307.80
- date	1	369019	517040	324.08
> summary(nuc.step)				

Call:

```
lm(formula = cost ~ date + cap + ne + t2 + pr + cum.n + ct, data = nuclear)
```

Residuals:

Min	1Q	Median	3Q	Max
-137.833	-46.897	-3.834	38.100	178.174

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-9.606e+03	1.259e+03	-7.627	7.27e-08 ***
date	1.386e+02	1.792e+01	7.735	5.70e-08 ***
cap	4.215e-01	8.203e-02	5.138	2.93e-05 ***
ne	1.434e+02	3.333e+01	4.301	0.000245 ***
t2	4.011e+00	1.865e+00	2.151	0.041748 *
pr	-7.372e+01	3.510e+01	-2.100	0.046386 *
cum.n	-8.222e+00	2.797e+00	-2.939	0.007164 **

```

ct      4.757e+01 3.030e+01 1.570 0.129534
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

```

Residual standard error: 78.53 on 24 degrees of freedom

Multiple R-squared: 0.835, Adjusted R-squared: 0.7869

F-statistic: 17.35 on 7 and 24 DF, p-value: 5.663e-08

> #(b) The model in (a) is again different to that selected by either forward or backward elimination. It's most similar to the model chosen via backward selection, differing only by an additional, non-significant, effect for 'ct'.

> #(c)

```

> gal <- data.frame(d=c(573,534,495,451,395,337,253),h=c(1,0.8,0.6,0.45,0.3,0.2,0.1))
> gal.mod0 <- lm(d~1,data=gal)
> gal.mod1 <- lm(d~h,data=gal)
> gal.mod2 <- lm(d~h+I(h^2),data=gal)
> gal.mod3 <- lm(d~h+I(h^2)+I(h^3),data=gal)
> gal.mod4 <- lm(d~h+I(h^2)+I(h^3)+I(h^4),data=gal)

```

> #(d)

```
> anova(gal.mod0,gal.mod1,gal.mod2,gal.mod3,gal.mod4)
```

Analysis of Variance Table

Model 1: d ~ 1

Model 2: d ~ h

Model 3: d ~ h + I(h^2)

Model 4: d ~ h + I(h^2) + I(h^3)

Model 5: d ~ h + I(h^2) + I(h^3) + I(h^4)

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	6	77022				
2	5	5671	1	71351	11208.1079	8.921e-05 ***
3	4	744	1	4927	773.9758	0.001290 **
4	3	48	1	696	109.3033	0.009025 **

```
5   2 13 1    36  5.5799 0.142011
```

```
---
```

```
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
> # Yes, the model chosen via nested partial F-tests matches the cubic model you would have likely identified in Exercise 21.2. The improvement to goodness-of-fit is minimal (and not statistically significant) if moving from the order 3 to the order 4 model.
```

```
> #(e)
```

```
> library("faraway")
```

```
> diab <-
```

```
na.omit(diabetes[,c("chol","age","gender","height","weight","frame","waist","hip","location")])
```

```
> #(f)
```

```
> dia.null <- lm(chol~1,data=diab)
```

```
> dia.full <- lm(chol~age*gender*weight*frame+waist*height*hip+location,data=diab)
```

```
> #(g)
```

```
> dia.step <- step(dia.null,scope=..+age*gender*weight*frame+waist*height*hip+location)
```

```
Start: AIC=2892.63
```

```
chol ~ 1
```

	Df	Sum of Sq	RSS	AIC
+ age	1	36863	689181	2874.7
+ frame	2	20262	705781	2885.8
+ waist	1	13442	712601	2887.5
+ hip	1	5567	720476	2891.7
+ weight	1	4954	721090	2892.0
<none>		726044	2892.6	
+ location	1	2511	723532	2893.3
+ height	1	1891	724153	2893.6
+ gender	1	494	725550	2894.4

```
Step: AIC=2874.67
```

```
chol ~ age
```

	Df	Sum of Sq	RSS	AIC
+ frame	2	17740	671441	2868.7
+ waist	1	7079	682102	2872.7
+ weight	1	6147	683034	2873.2
+ hip	1	4774	684408	2874.0
<none>		689181	2874.7	
+ location	1	2676	686505	2875.2
+ gender	1	1089	688092	2876.1
+ height	1	660	688521	2876.3
- age	1	36863	726044	2892.6

Step: AIC=2868.68

chol ~ age + frame

	Df	Sum of Sq	RSS	AIC
+ age:frame	2	9896	661545	2867.0
+ waist	1	5820	665621	2867.3
+ weight	1	5007	666434	2867.8
<none>		671441	2868.7	
+ hip	1	2780	668661	2869.1
+ location	1	1911	669530	2869.6
+ gender	1	736	670705	2870.3
+ height	1	282	671159	2870.5
- frame	2	17740	689181	2874.7
- age	1	34341	705781	2885.8

Step: AIC=2867

chol ~ age + frame + age:frame

	Df	Sum of Sq	RSS	AIC
+ waist	1	4468.8	657076	2866.4

	Df	Sum of Sq	RSS	AIC
<none>		661545	2867.0	
+ weight	1	3207.1	658338	2867.1
+ hip	1	2022.8	659522	2867.8
+ location	1	1660.4	659884	2868.0
+ gender	1	623.4	660921	2868.6
- age:frame	2	9896.1	671441	2868.7
+ height	1	355.2	661189	2868.8

Step: AIC=2866.4

chol ~ age + frame + waist + age:frame

	Df	Sum of Sq	RSS	AIC
<none>		657076	2866.4	
- waist	1	4468.8	661545	2867.0
- age:frame	2	8545.1	665621	2867.3
+ location	1	1585.1	655491	2867.5
+ height	1	442.5	656633	2868.1
+ hip	1	262.7	656813	2868.2
+ gender	1	225.4	656850	2868.3
+ weight	1	3.6	657072	2868.4

>

```
> summary(dia.step)
```

Call:

```
lm(formula = chol ~ age + frame + waist + age:frame, data = diab)
```

Residuals:

Min	1Q	Median	3Q	Max
-126.228	-26.008	-5.256	21.263	221.888

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	131.3260	18.3762	7.147	4.66e-12 ***
age	0.9939	0.2662	3.734	0.000218 ***
framemedium	28.8524	15.4951	1.862	0.063377 .
framelarge	39.1384	19.4922	2.008	0.045369 *
waist	0.6875	0.4299	1.599	0.110633
age:framemedium	-0.3788	0.3336	-1.136	0.256864
age:framelarge	-0.8286	0.3754	-2.208	0.027880 *
---				

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 41.8 on 376 degrees of freedom

Multiple R-squared: 0.09499, Adjusted R-squared: 0.08055

F-statistic: 6.578 on 6 and 376 DF, p-value: 1.271e-06

> #(h)

```
> add1(dia.null,scope=~.+age*gender*weight*frame+waist*height*hip+location,test="F")  
Single term additions
```

Model:

chol ~ 1

	Df	Sum of Sq	RSS	AIC	F value	Pr(>F)
<none>		726044	2892.6			
age	1	36863	689181	2874.7	20.3787	8.48e-06 ***
gender	1	494	725550	2894.4	0.2593	0.610892
weight	1	4954	721090	2892.0	2.6175	0.106518
frame	2	20262	705781	2885.8	5.4547	0.004618 **
waist	1	13442	712601	2887.5	7.1871	0.007662 **
height	1	1891	724153	2893.6	0.9949	0.319187
hip	1	5567	720476	2891.7	2.9441	0.087004 .
location	1	2511	723532	2893.3	1.3224	0.250874
---						

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

```
> dia.1 <- update(dia.null,~.+age)
```

```
> add1(dia.1,scope=~.+age*gender*weight*frame+waist*height*hip+location,test="F")
```

Single term additions

Model:

chol ~ age

	Df	Sum of Sq	RSS	AIC	F value	Pr(>F)
<none>		689181	2874.7			
gender	1	1089.1	688092	2876.1	0.6015	0.438497
weight	1	6147.4	683034	2873.2	3.4200	0.065186 .
frame	2	17740.3	671441	2868.7	5.0068	0.007142 **
waist	1	7079.1	682102	2872.7	3.9438	0.047763 *
height	1	659.6	688521	2876.3	0.3640	0.546629
hip	1	4773.6	684408	2874.0	2.6504	0.104353
location	1	2676.4	686505	2875.2	1.4815	0.224302

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> dia.2 <- update(dia.1,.~.+frame)

> add1(dia.2,scope=.~.+age\*gender\*weight\*frame+waist\*height\*hip+location,test="F")

Single term additions

Model:

chol ~ age + frame

	Df	Sum of Sq	RSS	AIC	F value	Pr(>F)
<none>		671441	2868.7			
gender	1	735.5	670705	2870.3	0.4145	0.52007
weight	1	5007.3	666434	2867.8	2.8401	0.09276 .
waist	1	5819.8	665621	2867.3	3.3050	0.06986 .
height	1	281.6	671159	2870.5	0.1586	0.69068
hip	1	2779.6	668661	2869.1	1.5714	0.21078
location	1	1910.8	669530	2869.6	1.0788	0.29964
age:frame	2	9896.1	661545	2867.0	2.8198	0.06088 .

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> summary(dia.2)

Call:

lm(formula = chol ~ age + frame, data = diab)

Residuals:

Min	1Q	Median	3Q	Max
-138.53	-26.66	-3.69	22.58	225.41

Coefficients:

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

(Intercept) 171.0427 7.1807 23.820 < 2e-16 \*\*\*  
age 0.6077 0.1380 4.403 1.39e-05 \*\*\*  
framemedium 15.5544 5.2444 2.966 0.00321 \*\*  
framelarge 4.7671 6.1191 0.779 0.43643  
---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 42.09 on 379 degrees of freedom

Multiple R-squared: 0.07521, Adjusted R-squared: 0.06789

F-statistic: 10.27 on 3 and 379 DF, p-value: 1.619e-06 > # The two models in (g) and (h) are quite different. Step-wise AIC has chosen a model that includes the two-way interaction between 'age' and 'frame', as well as a main effect for 'waist'. Performing forward selection halts after main effects for 'age' and 'frame' are added only, with no further additions offering a statistically significant improvement to goodness-of-fit.

> #(i)

> dia.step2 <- step(dia.full)

Start: AIC=2876.45

chol ~ age \* gender \* weight \* frame + waist \* height \* hip +  
location

	Df	Sum of Sq	RSS	AIC
- age:gender:weight:frame	2	1570.6	593562	2873.5
- location	1	1704.8	593696	2875.6
<none>		591991	2876.4	
- waist:height:hip	1	4221.1	596212	2877.2

Step: AIC=2873.47

chol ~ age + gender + weight + frame + waist + height + hip +  
location + age:gender + age:weight + gender:weight + age:frame +  
gender:frame + weight:frame + waist:height + waist:hip +

height:hip + age:gender:weight + age:gender:frame + age:weight:frame +  
 gender:weight:frame + waist:height:hip

	Df	Sum of Sq	RSS	AIC
- age:gender:frame	2	1473.5	595035	2870.4
- age:weight:frame	2	4008.6	597570	2872.0
- gender:weight:frame	2	4114.0	597676	2872.1
- location	1	1448.8	595011	2872.4
- age:gender:weight	1	1883.2	595445	2872.7
<none>		593562	2873.5	
- waist:height:hip	1	4131.1	597693	2874.1

Step: AIC=2870.41

chol ~ age + gender + weight + frame + waist + height + hip +  
 location + age:gender + age:weight + gender:weight + age:frame +  
 gender:frame + weight:frame + waist:height + waist:hip +  
 height:hip + age:gender:weight + age:weight:frame + gender:weight:frame +  
 waist:height:hip

	Df	Sum of Sq	RSS	AIC
- age:weight:frame	2	4209.1	599244	2869.1
- location	1	1482.2	596517	2869.4
- gender:weight:frame	2	5363.1	600398	2869.8
- age:gender:weight	1	2751.8	597787	2870.2
<none>		595035	2870.4	
- waist:height:hip	1	3989.8	599025	2871.0

Step: AIC=2869.11

chol ~ age + gender + weight + frame + waist + height + hip +  
 location + age:gender + age:weight + gender:weight + age:frame +  
 gender:frame + weight:frame + waist:height + waist:hip +

height:hip + age:gender:weight + gender:weight:frame + waist:height:hip

	Df	Sum of Sq	RSS	AIC
- age:frame	2	386.7	599631	2865.4
- location	1	1258.2	600503	2867.9
- gender:weight:frame	2	5609.4	604854	2868.7
- age:gender:weight	1	2882.2	602127	2868.9
<none>		599244	2869.1	
- waist:height:hip	1	3980.4	603225	2869.7

Step: AIC=2865.36

chol ~ age + gender + weight + frame + waist + height + hip +  
location + age:gender + age:weight + gender:weight + gender:frame +  
weight:frame + waist:height + waist:hip + height:hip + age:gender:weight +  
gender:weight:frame + waist:height:hip

	Df	Sum of Sq	RSS	AIC
- location	1	1284.0	600915	2864.2
- gender:weight:frame	2	5824.2	605455	2865.1
- age:gender:weight	1	3030.1	602661	2865.3
<none>		599631	2865.4	
- waist:height:hip	1	4146.7	603778	2866.0

Step: AIC=2864.18

chol ~ age + gender + weight + frame + waist + height + hip +  
age:gender + age:weight + gender:weight + gender:frame +  
weight:frame + waist:height + waist:hip + height:hip + age:gender:weight +  
gender:weight:frame + waist:height:hip

	Df	Sum of Sq	RSS	AIC
- gender:weight:frame	2	5701.2	606616	2863.8

```

- age:gender:weight  1  2977.2 603892 2864.1
<none>                  600915 2864.2
- waist:height:hip  1  3945.4 604860 2864.7

```

Step: AIC=2863.8

chol ~ age + gender + weight + frame + waist + height + hip +  
 age:gender + age:weight + gender:weight + gender:frame +  
 weight:frame + waist:height + waist:hip + height:hip + age:gender:weight +  
 waist:height:hip

	Df	Sum of Sq	RSS	AIC
- weight:frame	2	3371.5	609988	2861.9
- gender:frame	2	4023.3	610640	2862.3
- waist:height:hip	1	1577.2	608193	2862.8
<none>		606616	2863.8	
- age:gender:weight	1	3460.2	610077	2864.0

Step: AIC=2861.92

chol ~ age + gender + weight + frame + waist + height + hip +  
 age:gender + age:weight + gender:weight + gender:frame +  
 waist:height + waist:hip + height:hip + age:gender:weight +  
 waist:height:hip

	Df	Sum of Sq	RSS	AIC
- gender:frame	2	3209.7	613197	2859.9
- waist:height:hip	1	1617.6	611605	2860.9
<none>		609988	2861.9	
- age:gender:weight	1	4315.1	614303	2862.6

Step: AIC=2859.93

chol ~ age + gender + weight + frame + waist + height + hip +

age:gender + age:weight + gender:weight + waist:height +  
 waist:hip + height:hip + age:gender:weight + waist:height:hip

	Df	Sum of Sq	RSS	AIC
- waist:height:hip	1	1841.4	615039	2859.1
<none>		613197	2859.9	
- age:gender:weight	1	4169.7	617367	2860.5
- frame	2	7774.5	620972	2860.8

Step: AIC=2859.08

chol ~ age + gender + weight + frame + waist + height + hip +  
 age:gender + age:weight + gender:weight + waist:height +  
 waist:hip + height:hip + age:gender:weight

	Df	Sum of Sq	RSS	AIC
- waist:height	1	119.3	615158	2857.2
- height:hip	1	196.6	615235	2857.2
<none>		615039	2859.1	
- age:gender:weight	1	4835.4	619874	2860.1
- frame	2	8260.9	623300	2860.2
- waist:hip	1	8874.0	623913	2862.6

Step: AIC=2857.15

chol ~ age + gender + weight + frame + waist + height + hip +  
 age:gender + age:weight + gender:weight + waist:hip + height:hip +  
 age:gender:weight

	Df	Sum of Sq	RSS	AIC
- height:hip	1	1248.5	616407	2855.9
<none>		615158	2857.2	
- age:gender:weight	1	4828.3	619986	2858.2

- frame	2	8801.8	623960	2858.6
- waist:hip	1	8950.0	624108	2860.7

Step: AIC=2855.93

chol ~ age + gender + weight + frame + waist + height + hip +  
 age:gender + age:weight + gender:weight + waist:hip + age:gender:weight

	Df	Sum of Sq	RSS	AIC
- height	1	727.1	617134	2854.4
<none>			616407	2855.9
- age:gender:weight	1	4713.5	621120	2856.8
- frame	2	9157.6	625564	2857.6
- waist:hip	1	8105.1	624512	2858.9

Step: AIC=2854.38

chol ~ age + gender + weight + frame + waist + hip + age:gender +  
 age:weight + gender:weight + waist:hip + age:gender:weight

	Df	Sum of Sq	RSS	AIC
<none>			617134	2854.4
- age:gender:weight	1	4969.6	622103	2855.4
- frame	2	9423.4	626557	2856.2
- waist:hip	1	7822.5	624956	2857.2

```
> summary(dia.step2)
```

Call:

```
lm(formula = chol ~ age + gender + weight + frame + waist + hip +  
    age:gender + age:weight + gender:weight + waist:hip + age:gender:weight,  
    data = diab)
```

Residuals:

Min	1Q	Median	3Q	Max
-129.934	-24.257	-3.982	22.115	212.604

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-90.024818	111.236018	-0.809	0.4189
age	1.851810	1.056141	1.753	0.0804 .
genderfemale	32.626630	60.751227	0.537	0.5916
weight	0.549625	0.290229	1.894	0.0590 .
framemedium	10.484266	5.542926	1.891	0.0593 .
framelarge	0.309343	7.117136	0.043	0.9654
waist	6.213040	2.521085	2.464	0.0142 *
hip	3.788484	2.441865	1.551	0.1216
age:genderfemale	-0.980576	1.292190	-0.759	0.4484
age:weight	-0.011434	0.005813	-1.967	0.0499 *
genderfemale:weight	-0.474328	0.335521	-1.414	0.1583
waist:hip	-0.119344	0.055108	-2.166	0.0310 *
age:genderfemale:weight	0.012477	0.007228	1.726	0.0852 .
---				
Signif. codes:	0 ***	0.001 **	0.01 *	0.05 .
	1			

Residual standard error: 40.84 on 370 degrees of freedom

Multiple R-squared: 0.15, Adjusted R-squared: 0.1224

F-statistic: 5.441 on 12 and 370 DF, p-value: 1.525e-08> # The model chosen via step-wise AIC when starting from 'dia.full' is far, far more complex than the AIC-selected model in (g). The model contains a three-way interaction between 'age', 'gender' and 'weight', and all lower-order effects, as well as main effects and a two-way interaction between 'waist' and 'hip'. By starting from the very complex model, it seems we've accessed potential candidate models which have AIC values that are lower than those accessible by starting the selection algorithm at the null (intercept-only) model. For this reason, the final model reached in this way refuses to move to the simpler model selected in (g) since it's already found a model with a lower value for that criterion. Whether or not we'd be willing to settle on this rather complex model for predictive purposes is a more difficult question to answer...

```
> #(j)
> library("MASS")
> car.null <- lm(I(1/mpg)~1,data=mtcars)
> car.step <-
step(car.null,scope=..+wt*hp*factor(cyl)*disp+am+factor(gear)+drat+vs+qsec+carb)
```

Start: AIC=-261.99

I(1/mpg) ~ 1

	Df	Sum of Sq	RSS	AIC
+ wt	1	0.0066224	0.0017402	-310.22
+ disp	1	0.0064733	0.0018892	-307.60
+ factor(cyl)	2	0.0055711	0.0027914	-293.10
+ hp	1	0.0048677	0.0034948	-287.91
+ vs	1	0.0034243	0.0049382	-276.85
+ drat	1	0.0034045	0.0049580	-276.72
+ factor(gear)	2	0.0034781	0.0048844	-275.20
+ am	1	0.0024375	0.0059251	-271.02
+ carb	1	0.0023167	0.0060458	-270.37
+ qsec	1	0.0012450	0.0071175	-265.15
<none>		0.0083625	-261.99	

Step: AIC=-310.22

I(1/mpg) ~ wt

Df	Sum of Sq	RSS	AIC
----	-----------	-----	-----

```

+ hp      1 0.0004614 0.0012787 -318.08
+ qsec    1 0.0004578 0.0012824 -317.99
+ disp    1 0.0003175 0.0014226 -314.67
+ vs      1 0.0002579 0.0014823 -313.36
+ factor(cyl) 2 0.0003238 0.0014163 -312.81
+ carb    1 0.0002176 0.0015226 -312.50
+ factor(gear) 2 0.0002346 0.0015055 -310.86
<none>          0.0017402 -310.22
+ am      1 0.0000937 0.0016465 -310.00
+ drat    1 0.0000003 0.0017399 -308.23
- wt     1 0.0066224 0.0083625 -261.99

```

Step: AIC=-318.08

$I(1/\text{mpg}) \sim \text{wt} + \text{hp}$

	Df	Sum of Sq	RSS	AIC
<none>		0.0012787	-318.08	
+ qsec	1	0.00004906	0.0012297	-317.34
+ disp	1	0.00003920	0.0012395	-317.08
+ vs	1	0.00002130	0.0012574	-316.62
+ wt:hp	1	0.00000850	0.0012702	-316.30
+ drat	1	0.00000187	0.0012769	-316.13
+ am	1	0.00000185	0.0012769	-316.13
+ carb	1	0.00000000	0.0012787	-316.08
+ factor(gear)	2	0.00003001	0.0012487	-314.84
+ factor(cyl)	2	0.00001654	0.0012622	-314.50
- hp	1	0.00046144	0.0017402	-310.22
- wt	1	0.00221606	0.0034948	-287.91

> summary(car.step)

Call:

```
lm(formula = I(1/mpg) ~ wt + hp, data = mtcars)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.0168690	-0.0049601	0.0007663	0.0040027	0.0142186

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )		
(Intercept)	6.305e-03	4.094e-03	1.540	0.13435		
wt	1.149e-02	1.620e-03	7.089	8.45e-08 ***		
hp	7.479e-05	2.312e-05	3.235	0.00303 **		
---						
Signif. codes:	0 ‘***’	0.001 ‘**’	0.01 ‘*’	0.05 ‘.’	0.1 ‘ ’	1

Residual standard error: 0.00664 on 29 degrees of freedom

Multiple R-squared: 0.8471, Adjusted R-squared: 0.8365

F-statistic: 80.33 on 2 and 29 DF, p-value: 1.494e-12

```
> # Modeling the response as GPM instead of MPG, step-wise AIC selection offers up a far simpler model, with main effects for 'wt' and 'hp' only. It would seem that this transformation of the response greatly simplifies the relationships in the data.
```

## Exercise 22.2

In Section 22.2.2, you used the *nuclear* data frame in the *boot* package to illustrate forward selection, where a model was selected for *cost* as a function of main effects of *date*, *cap*, *pt*, and *ne*.

- a. Access the data frame; fit and summarize the model described earlier.
- b. Inspect the raw residuals versus fitted values and a normal QQ plot of the residuals and comment on your interpretations—do the assumptions underpinning the error component of the linear model appear satisfied in this case?
- c. Determine the rule-of-thumb cutoff for influential observations based on the Cook's distances. Produce a plot of the Cook's distances and add a horizontal line corresponding to the cutoff.

Comment on your findings.

- d. Produce a combination diagnostic plot of the standardized residuals against leverage. Set the Cook's distance contours to include the cutoff value from (c) as well as the default contours.

Interpret the plot—how are any individually influential points characterized?

- e. Based on (c) and (d), you should be able to identify the record in *nuclear* exerting the largest influence on the fitted model. For the sake of argument, let's assume the observation was recorded incorrectly. Refit the model from (a), this time omitting the offending row from the data frame. Summarize the model—which coefficients have changed the most? Produce the diagnostic plots from (b) for the new model and compare them to the ones from earlier.

Load the *faraway* package and access the *diabetes* data frame. In Exercise 22.1 (g), you used stepwise AIC selection to choose a model for *chol*.

- f. Using *diabetes*, fit the multiple linear model identified in the earlier exercise, that is, with main effects and a two-way interaction between age and frame and a main effect for waist. By summarizing the fit, determine the number of records that contained missing values in *diabetes* that were deleted from the estimation.
- g. Produce the raw residuals versus fitted and QQ diagnostic plots for the model in (f). Comment on the validity of the error assumptions.
- h. Investigate influential points. Make use of the familiar ruleof-thumb cutoff (note you'll need to subtract the number of missing values from the total size of the data frame to get the effective sample size for your model). In the combination plot of the standardized residuals against leverage, use one, three, and five times the cutoff as the Cook's distance contours.

Recall the discussion of reading in web-based files in Section 8.2.3.

There, you called in a data frame containing data on the prices of 308 diamonds (in Singapore dollars), as well as weight (in carats—continuous), color (categorical—six levels from *D*, the least yellow and the reference level, to *I*, the most yellow), clarity (categorical—five levels with *IF*, essentially flawless and the reference level, *VVS1*, *VVS2*, *VS1*, and *VS2*, with the last being the least clear), and certification (categorical—three levels for different diamond certification bodies with levels *GIA* as the reference, *HRD* and *IGI*). Seek out the freely available article by Chu (2001) for more information on these data.

With an Internet connection, run the following lines, which will read in the data as the object *diamonds* and name each variable column appropriately.

```
R> dia.url <- "http://www.amstat.org/publications/jse/v9n2/4cdatal.txt"
R> diamonds <- read.table(dia.url)
R> names(diamonds) <- c("Carat", "Color", "Clarity", "Cert", "Price")
```

- i. Using either base R graphics or *ggplot2*, to get a feel for the data, produce a scatterplot of the price on the y-axis and carat weight on the x-axis. Experiment with using plotting color to split the points according to the following:

– Diamond clarity

- Diamond color
  - Diamond certification
- j. Fit a multiple linear model with *Price* as the response and main effects for the other variables as the predictors. Summarize the model and produce the three diagnostic plots that tell you about the assumptions surrounding the error term. Comment on the plots—are you satisfied that this is an appropriate model for the diamond prices? Why or why not?
- k. Repeat (j) but use the log transformation of *Price*. Again, inspect and comment on the validity of the error assumptions.
- l. Repeat (k), but in modeling the log-price, this time include an additional quadratic term for *Carat* (refer to Section 21.4.1 for details on polynomial transformations). How do the residual diagnostics look now?

## Solution 22.2

```
> #(a)

> library("boot")

> nuc.fit <- lm(cost~date+cap+pt+ne,data=nuclear)

> summary(nuc.fit)
```

Call:

`lm(formula = cost ~ date + cap + pt + ne, data = nuclear)`

Residuals:

Min	1Q	Median	3Q	Max
-157.894	-38.424	-2.493	35.363	267.445

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-4.756e+03	1.286e+03	-3.699	0.000975 ***
date	7.102e+01	1.867e+01	3.804	0.000741 ***
cap	4.198e-01	8.616e-02	4.873	4.29e-05 ***
pt	-1.289e+02	4.950e+01	-2.605	0.014761 *
ne	9.940e+01	3.864e+01	2.573	0.015908 *
---				
Signif. codes:	0 ‘***’	0.001 ‘**’	0.01 ‘*’	0.05 ‘.’
	0.1 ‘ ’	1		

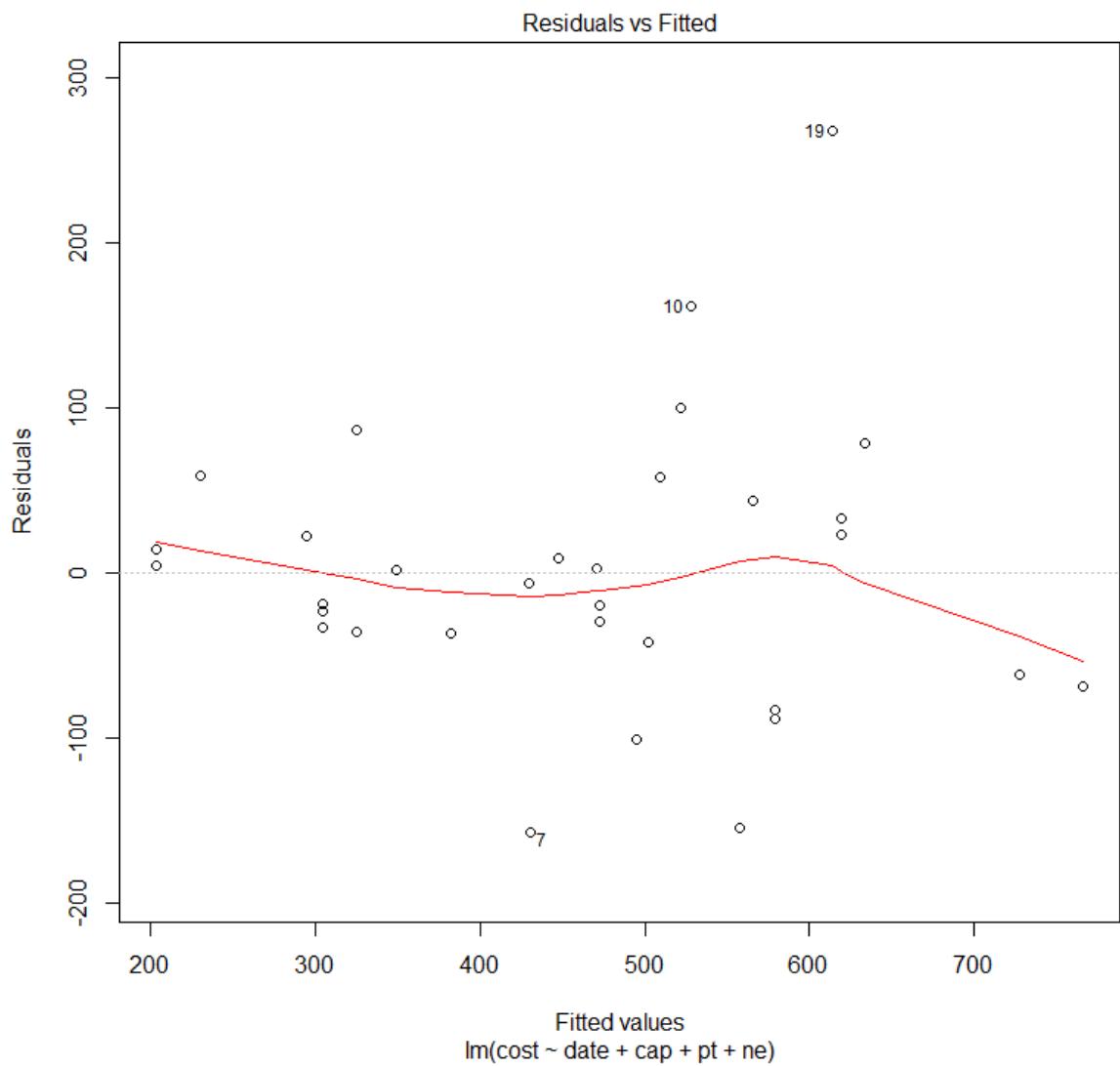
Residual standard error: 90.8 on 27 degrees of freedom

Multiple R-squared: 0.7519, Adjusted R-squared: 0.7151

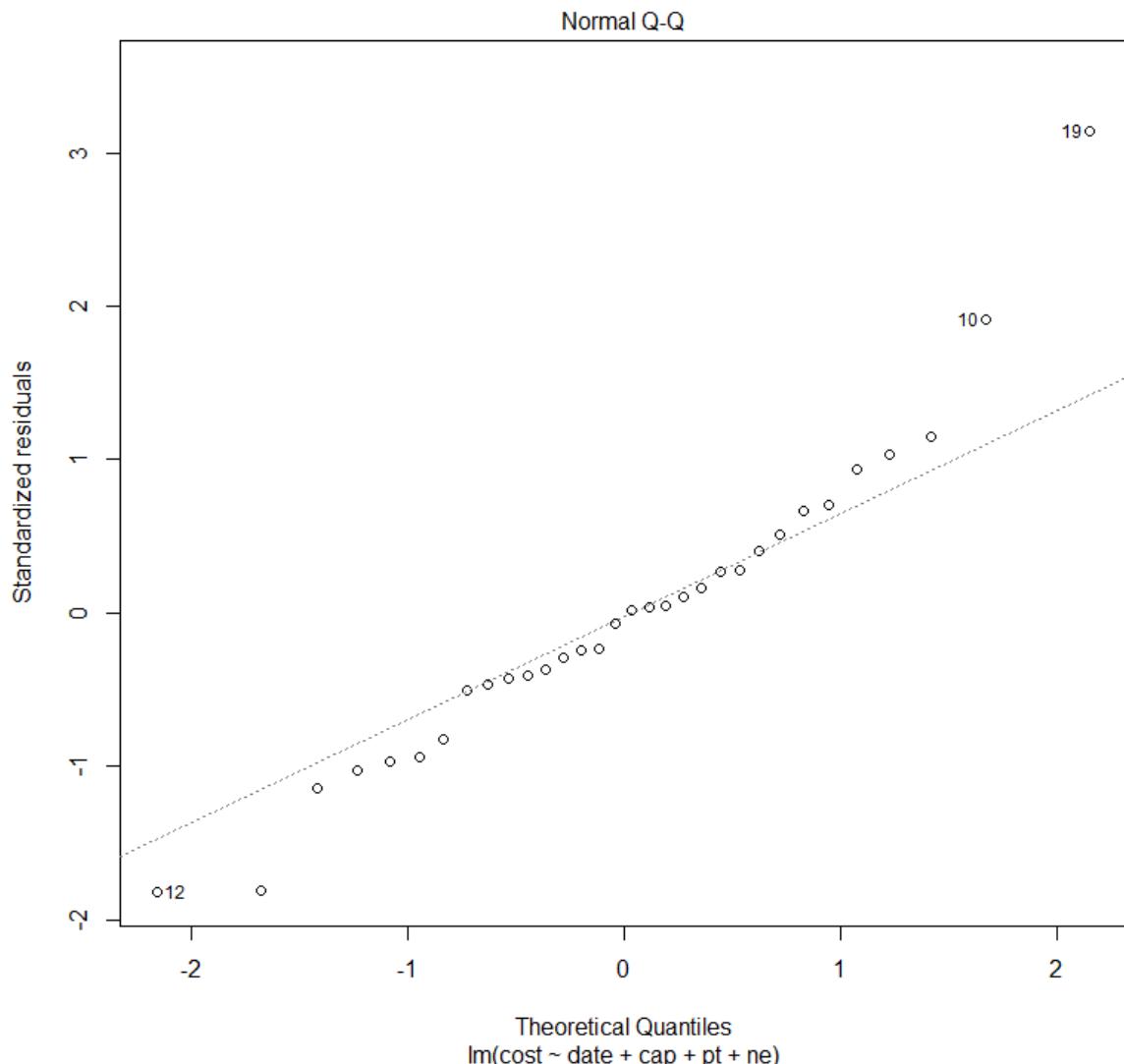
F-statistic: 20.45 on 4 and 27 DF, p-value: 7.507e-08

> #(b)

> plot(nuc.fit,which=1)

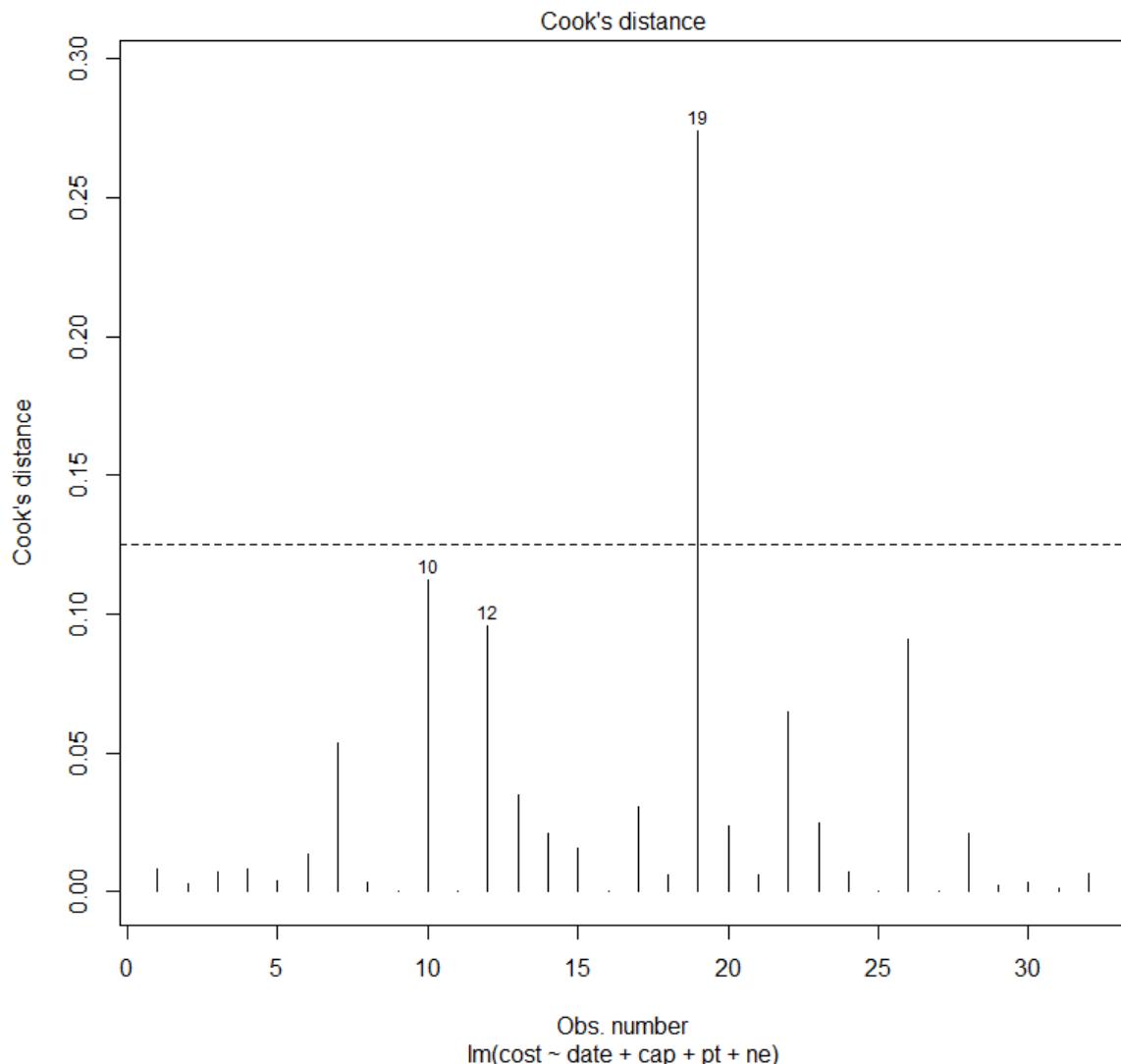


```
> plot(nuc.fit,which=2)
```



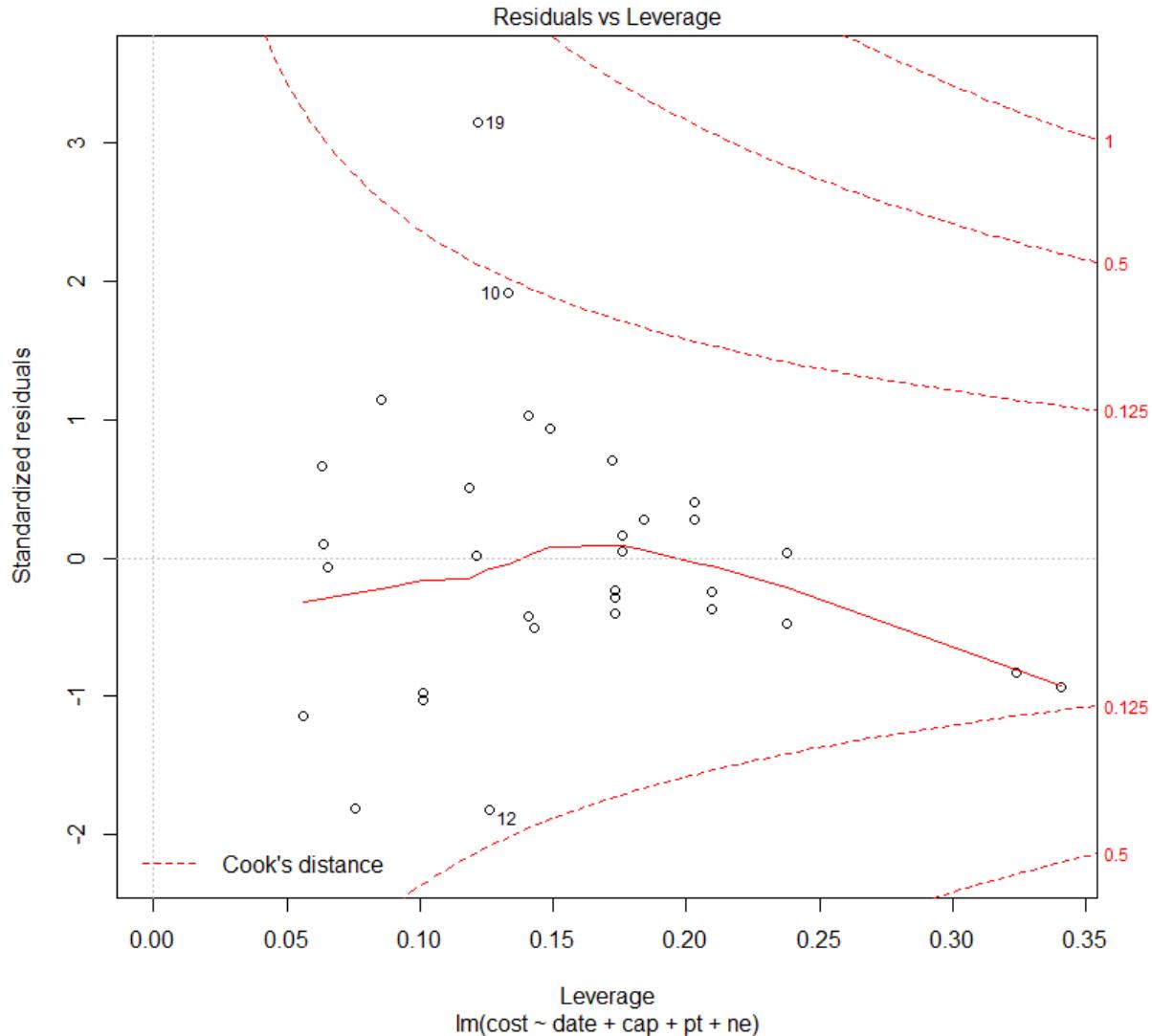
```
> plot(nuc.fit,which=4)
```

```
> abline(h=cutoff,lty=2)
```



> # The Cook's distances demonstrate what you might already have expected: that observation 19 is highly influential. It breaches the rule-of-thumb cut-off of 0.125 by a considerable amount. Observation 10 is the next most influential, albeit one that registers a Cook's distance less than 0.125.

```
> #(d)
> plot(nuc.fit,which=5,cook.levels=c(cutoff,0.5,1))
```



> # Observation 19 sits in a low-to-medium leverage position; it's the large residual for that point that pushes it past the rule-of-thumb contour. The same characterization can be applied to the other two labeled points that have more or less the same leverage (observations 10 and 12), though their smaller residuals mean a correspondingly lower influence.

```
> #(e)
> nuc.fit2 <- lm(cost ~ date + cap + pt + ne, data=nuclear[-19,])
> summary(nuc.fit2)
```

Call:

`lm(formula = cost ~ date + cap + pt + ne, data = nuclear[-19,`

] )

Residuals:

Min	1Q	Median	3Q	Max
-157.226	-46.025	-0.714	42.392	157.996

Coefficients:

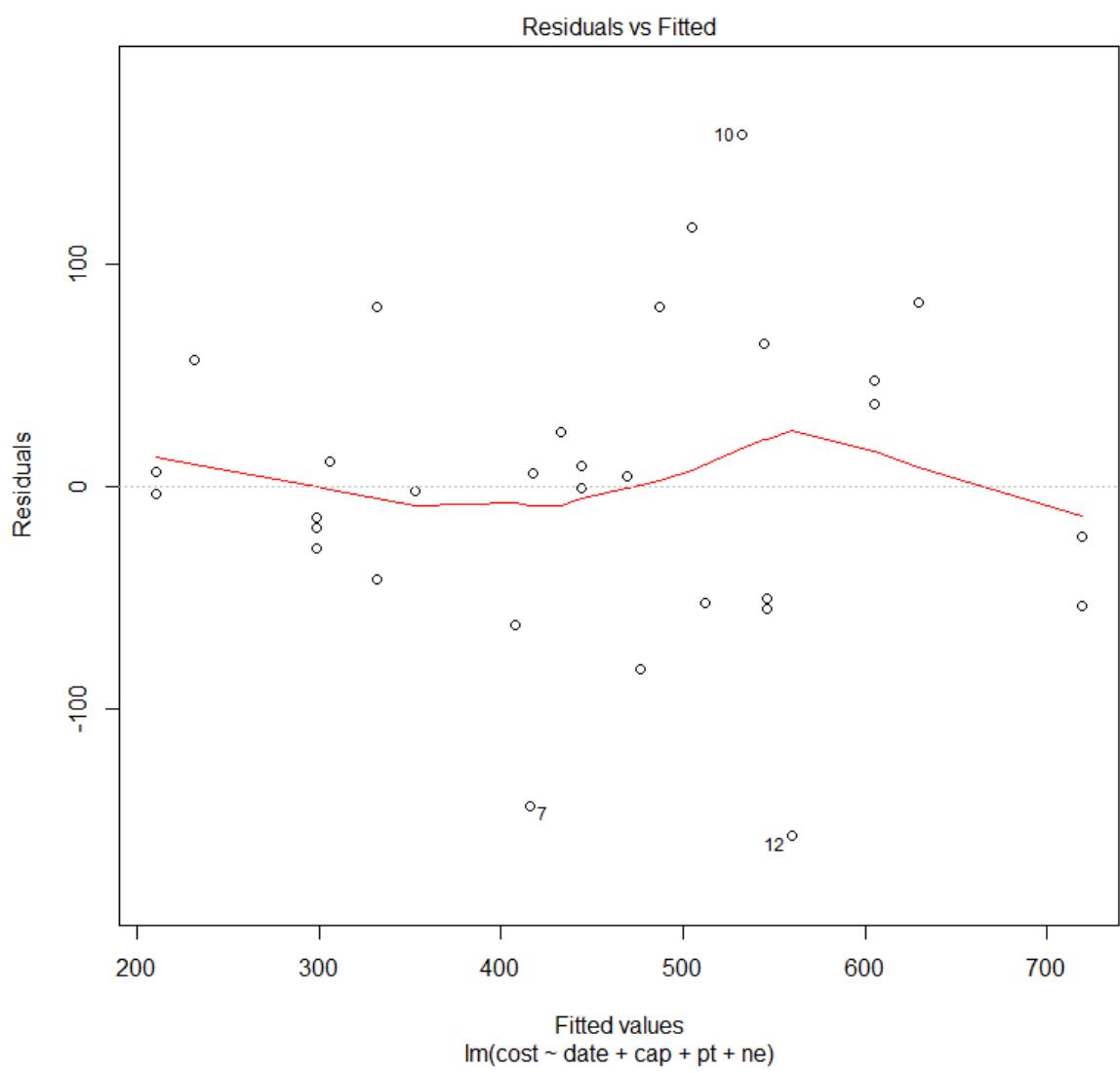
	Estimate	Std. Error	t value	Pr(> t )		
(Intercept)	-4.466e+03	1.046e+03	-4.270	0.000231 ***		
date	6.742e+01	1.518e+01	4.442	0.000146 ***		
cap	3.478e-01	7.235e-02	4.807	5.6e-05 ***		
pt	-1.166e+02	4.029e+01	-2.894	0.007598 **		
ne	1.158e+02	3.164e+01	3.660	0.001128 **		
---						
Signif. codes:	0 ‘***’	0.001 ‘**’	0.01 ‘*’	0.05 ‘.’	0.1 ‘ ’	1

Residual standard error: 73.68 on 26 degrees of freedom

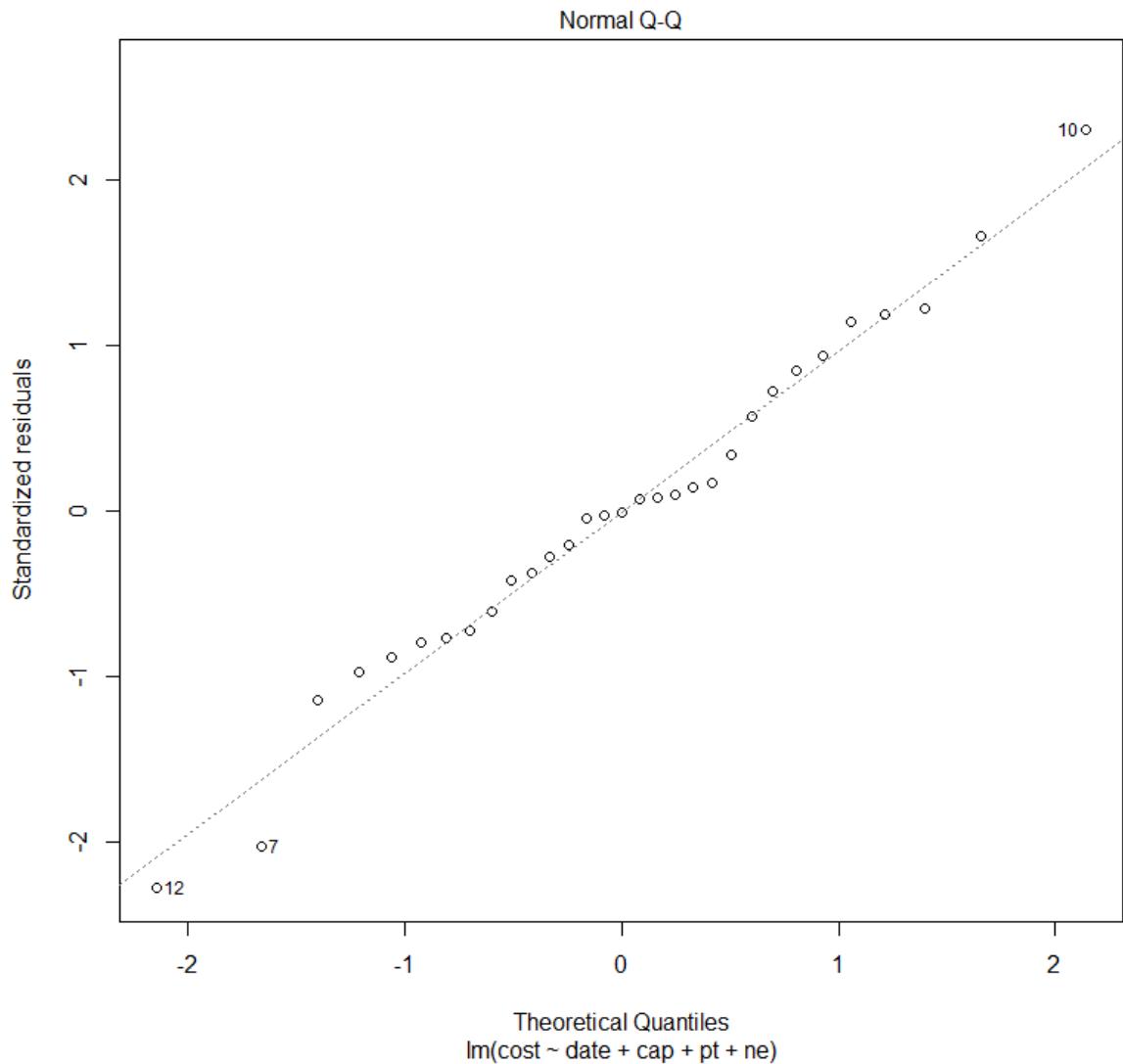
Multiple R-squared: 0.8027, Adjusted R-squared: 0.7723

F-statistic: 26.44 on 4 and 26 DF, p-value: 7.868e-09

> plot(nuc.fit2,which=1)



```
> plot(nuc.fit2,which=2)
```



> # Assuming there's a good reason to remove observation 19, once done so, the plots of the residuals are improved in terms of satisfaction of the assumptions of the error component. Independence and homogeneity are illustrated with randomness around zero, and normality also appears reasonable. Points 10 and 12 are now the top two extreme points, joined by observation 7.

```

> #(f)
> library("faraway")
> dia.fit <- lm(chol~age*frame+waist,data=diabetes)
> summary(dia.fit)

```

Call:

$\text{lm}(\text{formula} = \text{chol} \sim \text{age} * \text{frame} + \text{waist}, \text{data} = \text{diabetes})$

Residuals:

	Min	1Q	Median	3Q	Max
	-125.156	-25.290	-4.328	21.261	221.876

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	128.7526	18.5098	6.956	1.53e-11 ***
age	0.9555	0.2684	3.560	0.000418 ***
framemedium	26.9255	15.5990	1.726	0.085139 .
framelarge	35.6819	19.5654	1.824	0.068977 .
waist	0.8343	0.4311	1.935	0.053706 .
age:framemedium	-0.3757	0.3364	-1.117	0.264839
age:framelarge	-0.7885	0.3784	-2.084	0.037844 *
---				
Signif. codes:	0 ‘***’	0.001 ‘**’	0.01 ‘*’	0.05 ‘.’
	0.1 ‘ ’	1		

Residual standard error: 42.22 on 381 degrees of freedom

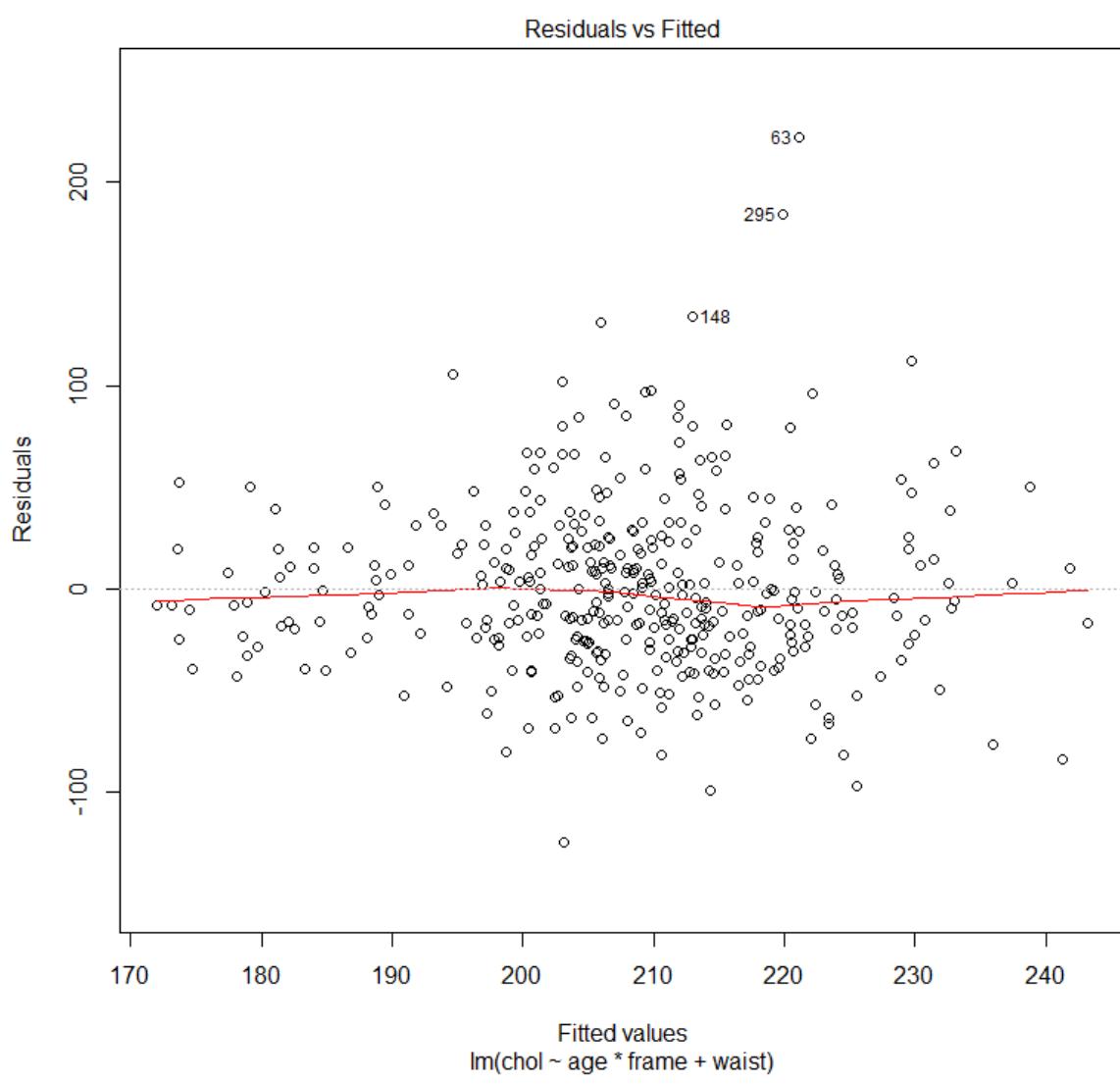
(15 observations deleted due to missingness)

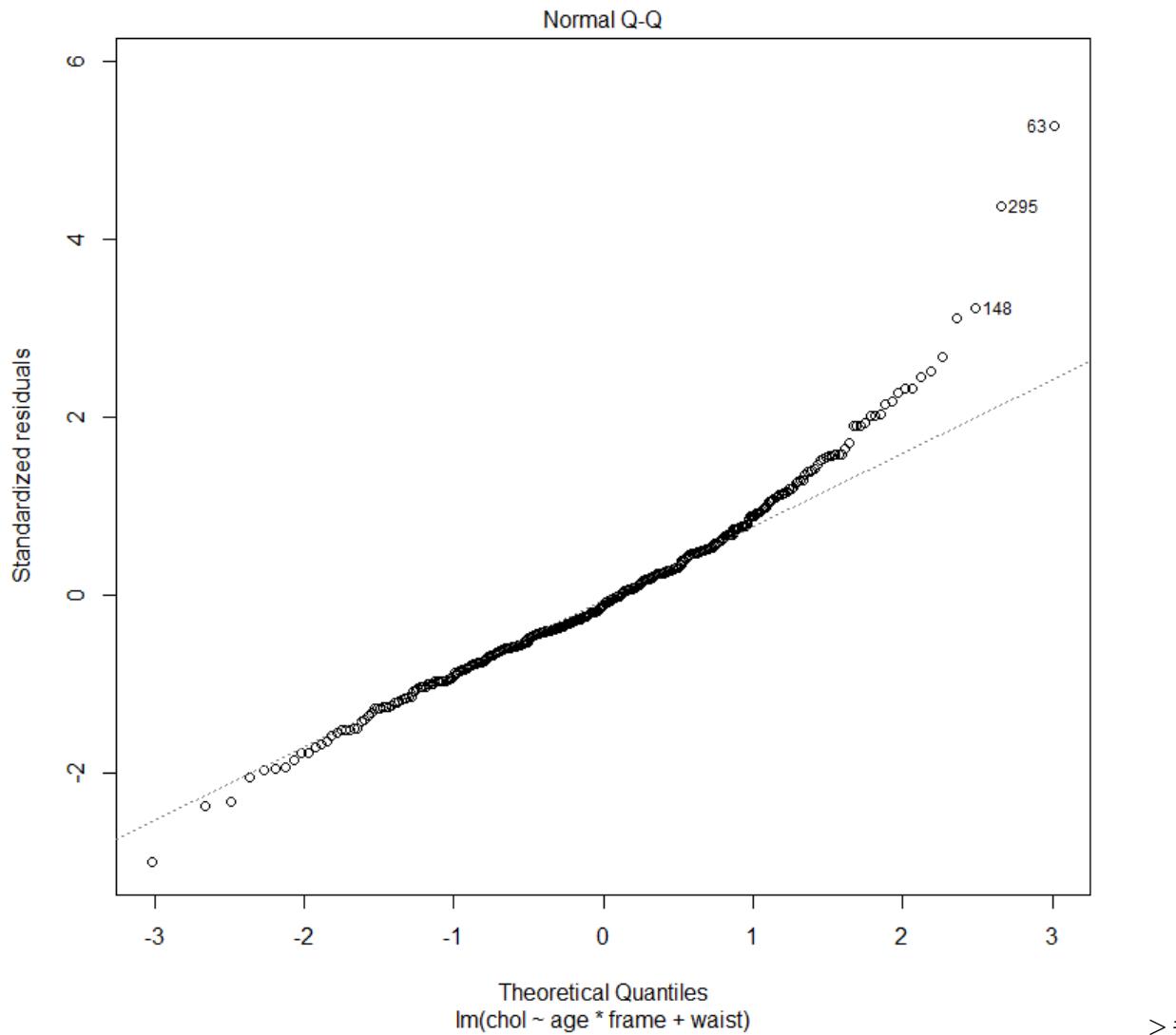
Multiple R-squared: 0.08764, Adjusted R-squared: 0.07328

F-statistic: 6.1 on 6 and 381 DF, p-value: 4.068e-06

> #(g)

> plot(dia.fit,which=1)



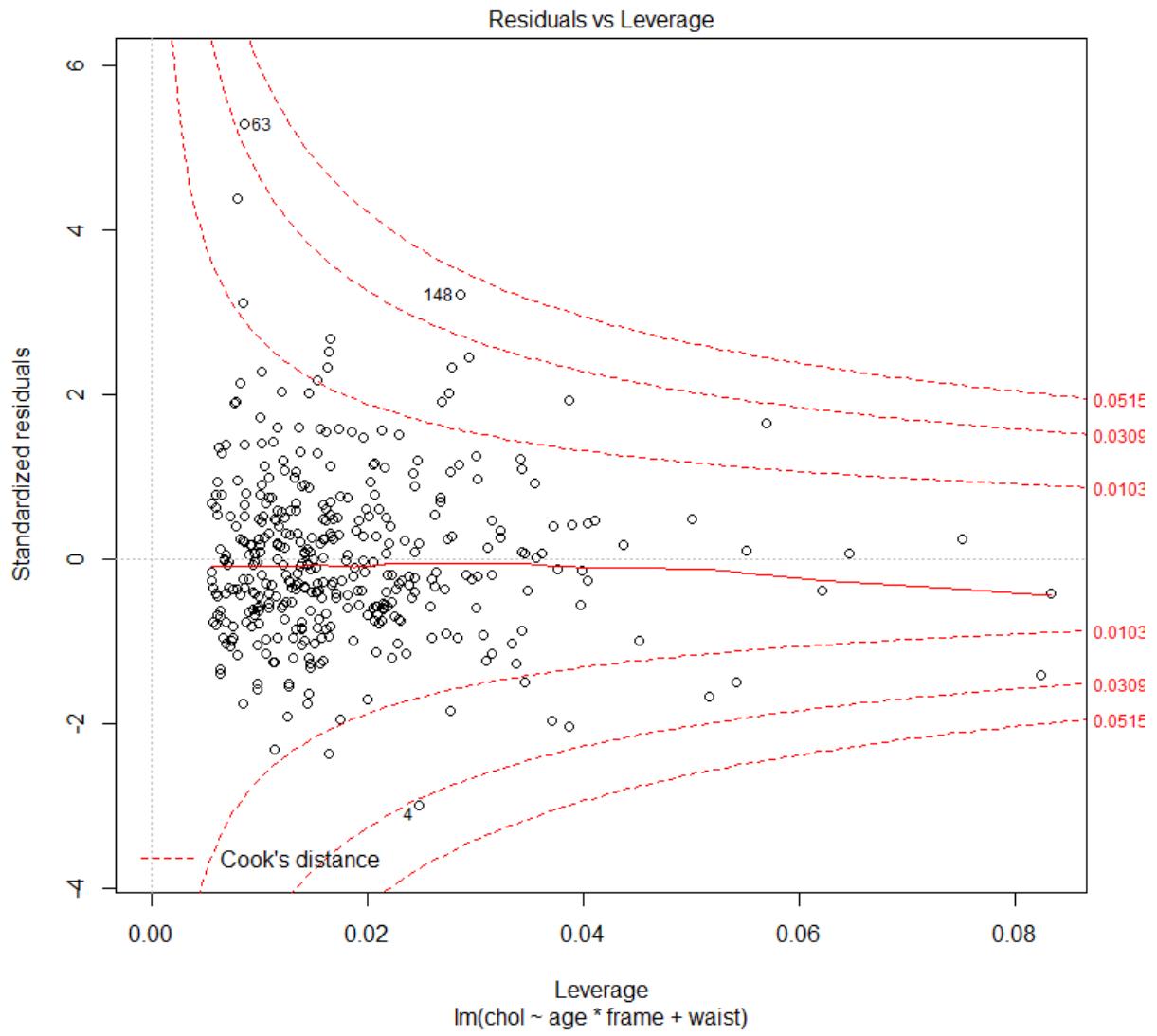


Very similar to the interpretation of the plots in the earlier exercise (b), the error assumptions seem satisfied here, with randomness around zero and homoscedasticity seemingly valid. There are some extreme positive residuals, namely point 63. The QQ plot does appear to exhibit some departure from normality in the upper tail, with the three extreme points 63, 295 and 148 not helping in that respect. Estimation will remain valid if normality is questionable, though the estimates might not be 'optimal' in a theoretical sense.

```

> #(h)
> nrow(diabetes)-15
[1] 388
> cutoff <- 4/(nrow(diabetes)-15)
> cutoff
[1] 0.125
> plot(dia.fit,which=5,cook.levels=c(1,3,5)*cutoff)

```



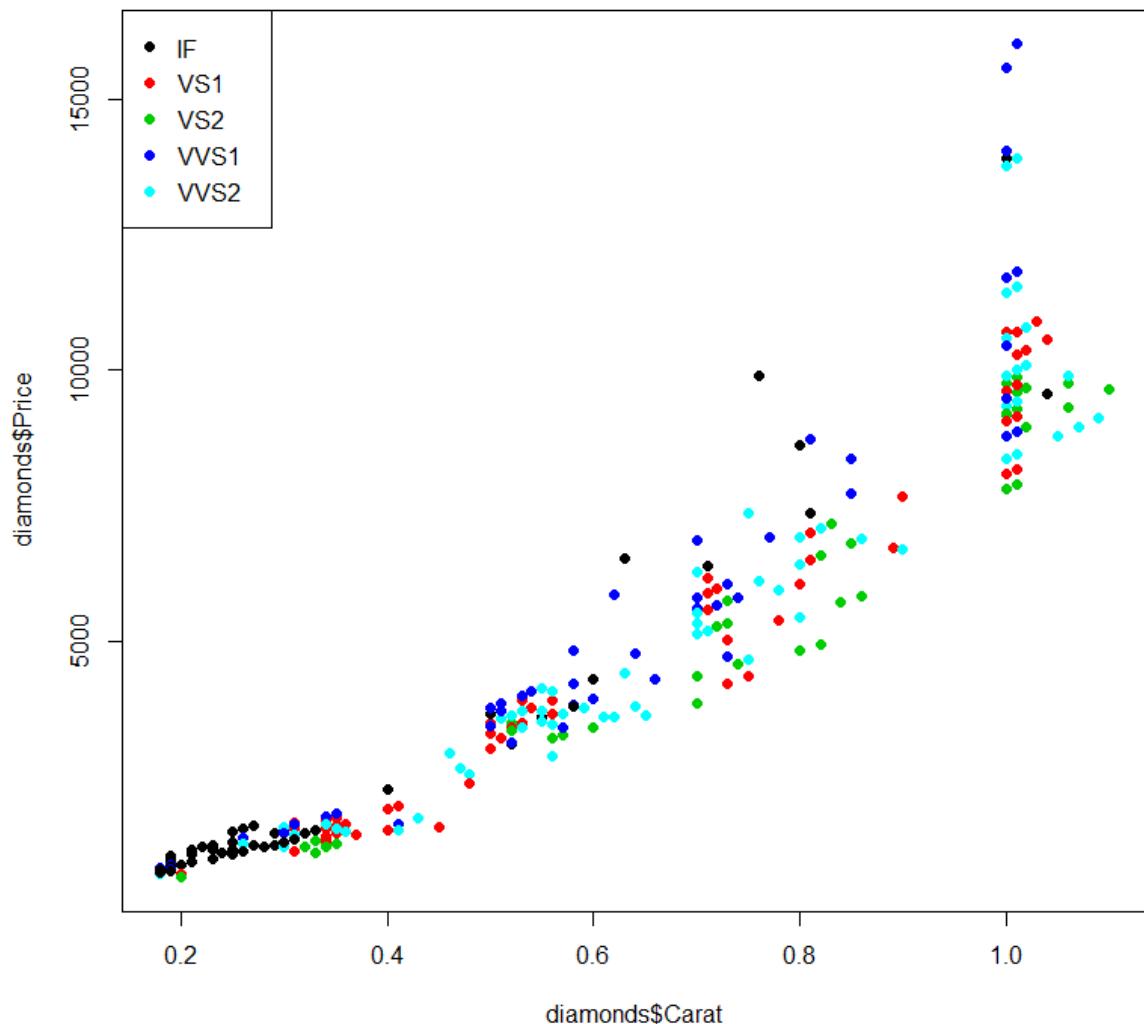
> # The size of the data set results in a relatively small rule-of-thumb influence cut-off of around 0.01. A number of points breach this mark, though the overall pattern of the standardized residuals vs. leverage appears consistent with the contours, so we shouldn't be especially concerned here. There are three points that breach the '3 times cut-off' contour. Point 63 is in a very low leverage position, but its extreme residual leads it to be highly influential. Points 4 and 148 have much smaller residuals, but their position in much higher leverage areas when compared with 63 which means they too are considered highly influential here.

```

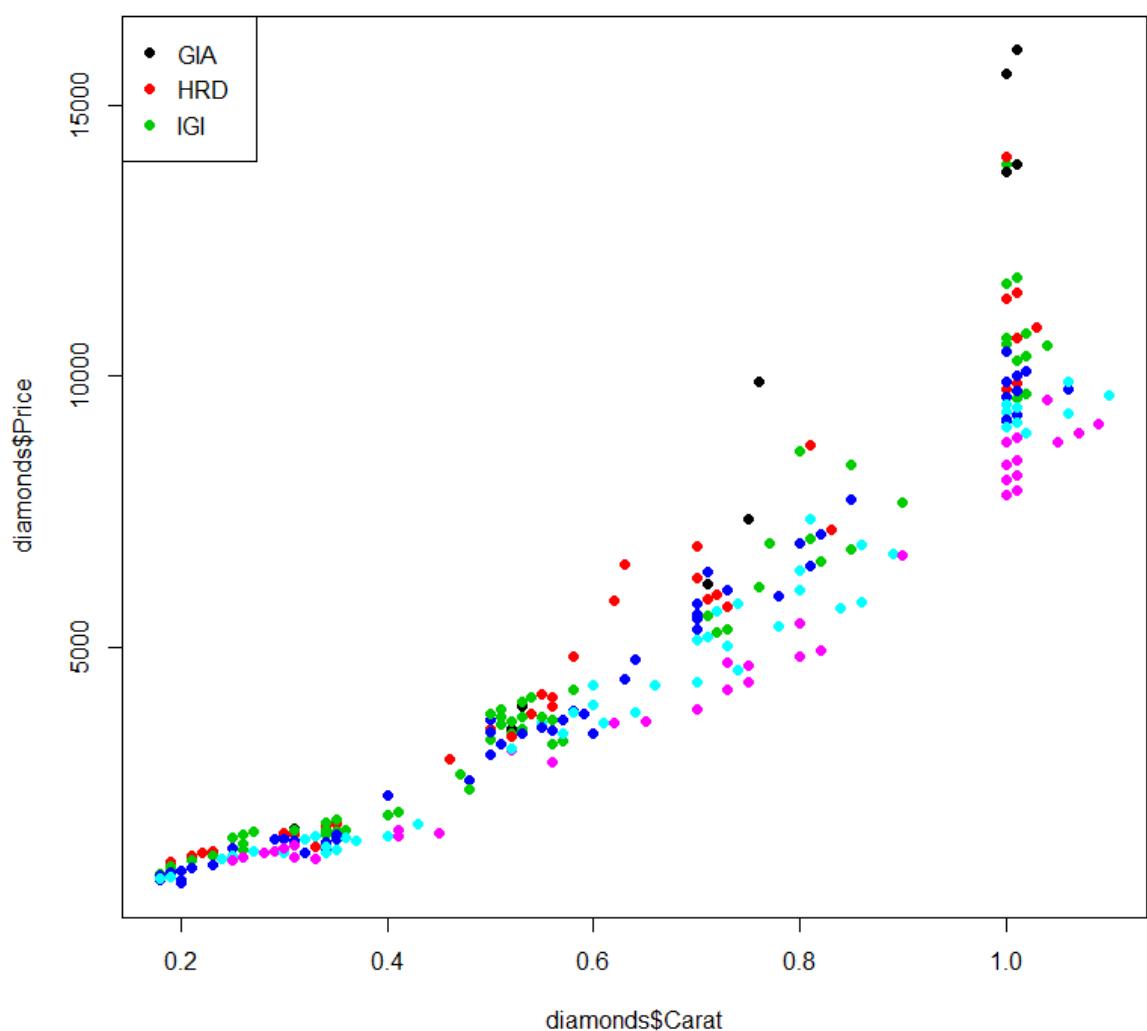
> #(i)
> dia.url <- "http://www.amstat.org/publications/jse/v9n2/4cdata.txt"
> diamonds <- read.table(dia.url)
> names(diamonds) <- c("Carat", "Color", "Clarity", "Cert", "Price")
> plot(diamonds$Carat, diamonds$Price, pch=19, col=as.numeric(diamonds$Clarity))

```

```
>  
legend("topleft",legend=levels(diamonds$Clarity),col=1:length(levels(diamonds$Clarity)),pch=19  
)
```



```
> plot(diamonds$Carat,diamonds$Price,pch=19,col=as.numeric(diamonds$Color))  
>  
legend("topleft",legend=levels(diamonds$Color),col=1:length(levels(diamonds$Color)),pch=19)  
> plot(diamonds$Carat,diamonds$Price,pch=19,col=as.numeric(diamonds$Cert))  
> legend("topleft",legend=levels(diamonds$Cert),col=1:length(levels(diamonds$Cert)),pch=19)
```



```

> #(j)
> sparkly.fit <- lm(Price~Carat+Color+Clarity+Cert,data=diamonds)
> summary(sparkly.fit)

```

Call:

`lm(formula = Price ~ Carat + Color + Clarity + Cert, data = diamonds)`

Residuals:

Min	1Q	Median	3Q	Max
-1740.0	-428.8	-128.3	314.3	3634.1

Coefficients:

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

(Intercept)	169.18	255.02	0.663	0.508
Carat	12766.40	190.02	67.183	< 2e-16 ***
ColorE	-1439.09	207.98	-6.919	2.83e-11 ***
ColorF	-1841.69	195.23	-9.433	< 2e-16 ***
ColorG	-2176.67	200.39	-10.862	< 2e-16 ***
ColorH	-2747.15	202.91	-13.538	< 2e-16 ***
ColorI	-3313.10	212.71	-15.575	< 2e-16 ***
ClarityVS1	-1474.57	159.67	-9.235	< 2e-16 ***
ClarityVS2	-1792.01	171.19	-10.468	< 2e-16 ***
ClarityVVS1	-689.29	159.93	-4.310	2.23e-05 ***
ClarityVVS2	-1191.16	148.76	-8.007	2.73e-14 ***
CertHRD	15.23	107.25	0.142	0.887
CertIGI	141.26	128.26	1.101	0.272

---

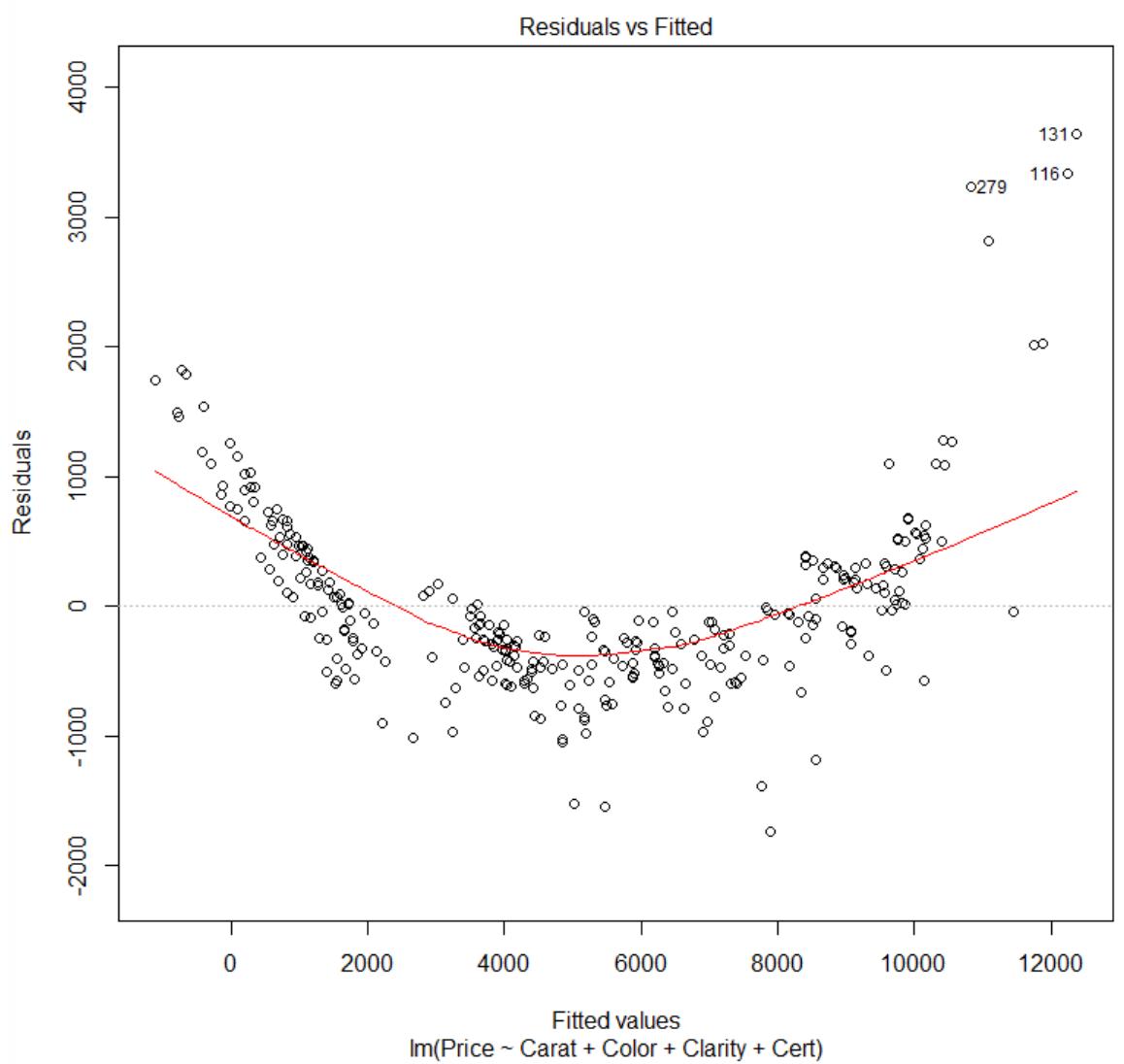
Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 710.4 on 295 degrees of freedom

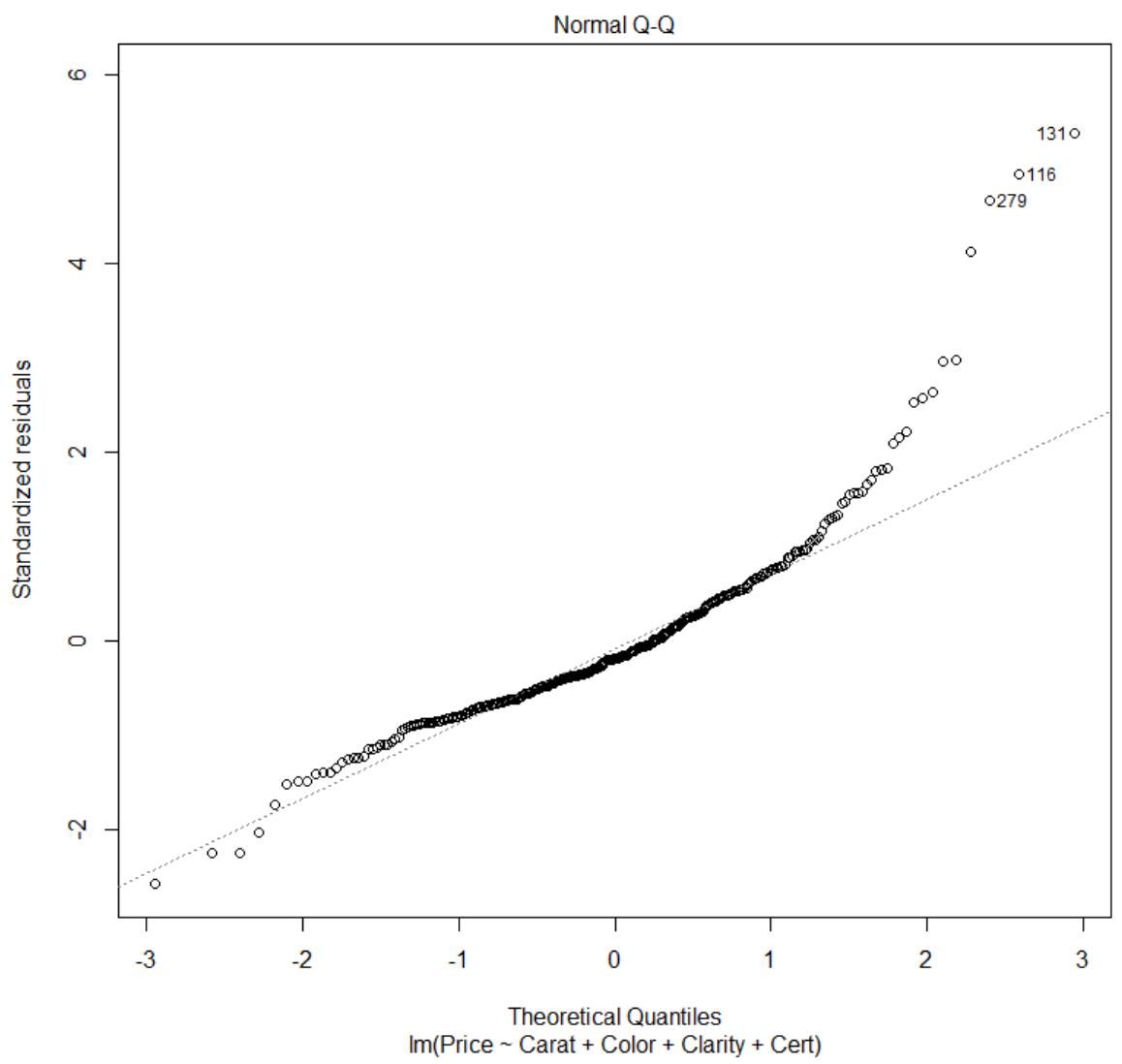
Multiple R-squared: 0.9581, Adjusted R-squared: 0.9564

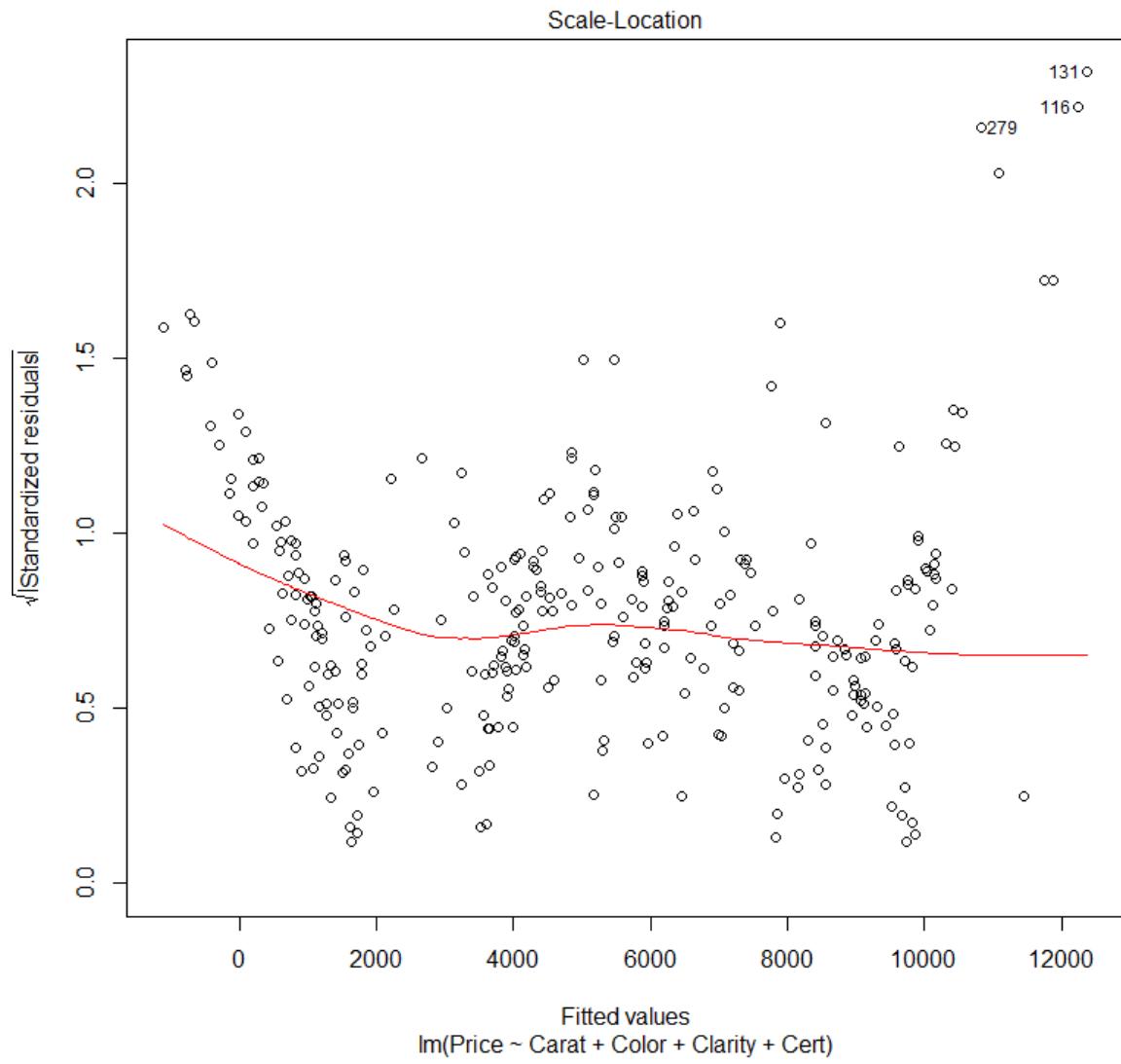
F-statistic: 562.5 on 12 and 295 DF, p-value: < 2.2e-16

> plot(sparkly.fit,which=1)



```
> plot(sparkly.fit,which=2)
```





> # There seems to be clear, systematic non-linearity in the estimated residuals. This violates the linearity assumption of the trends in our data, and suggests our current model is inadequate in terms of representation of the data at hand. Based on the plots from (i), should we try modeling a log-transformation of the response?

> # Normality is also affected---there is rather obvious deviation, from the distribution, particularly in the upper tail.

> # Barring three extreme points, though, plots 1 and 3 don't show that the assumptions of homoscedasticity is violated---the variability of the residuals remains more or less constant.

> #(k)

> sparkly.fit2 <- lm(log(Price)~Carat+Color+Clarity+Cert,data=diamonds)

> summary(sparkly.fit2)

Call:

```
lm(formula = log(Price) ~ Carat + Color + Clarity + Cert, data = diamonds)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.31236	-0.11520	0.01613	0.10833	0.36339

Coefficients:

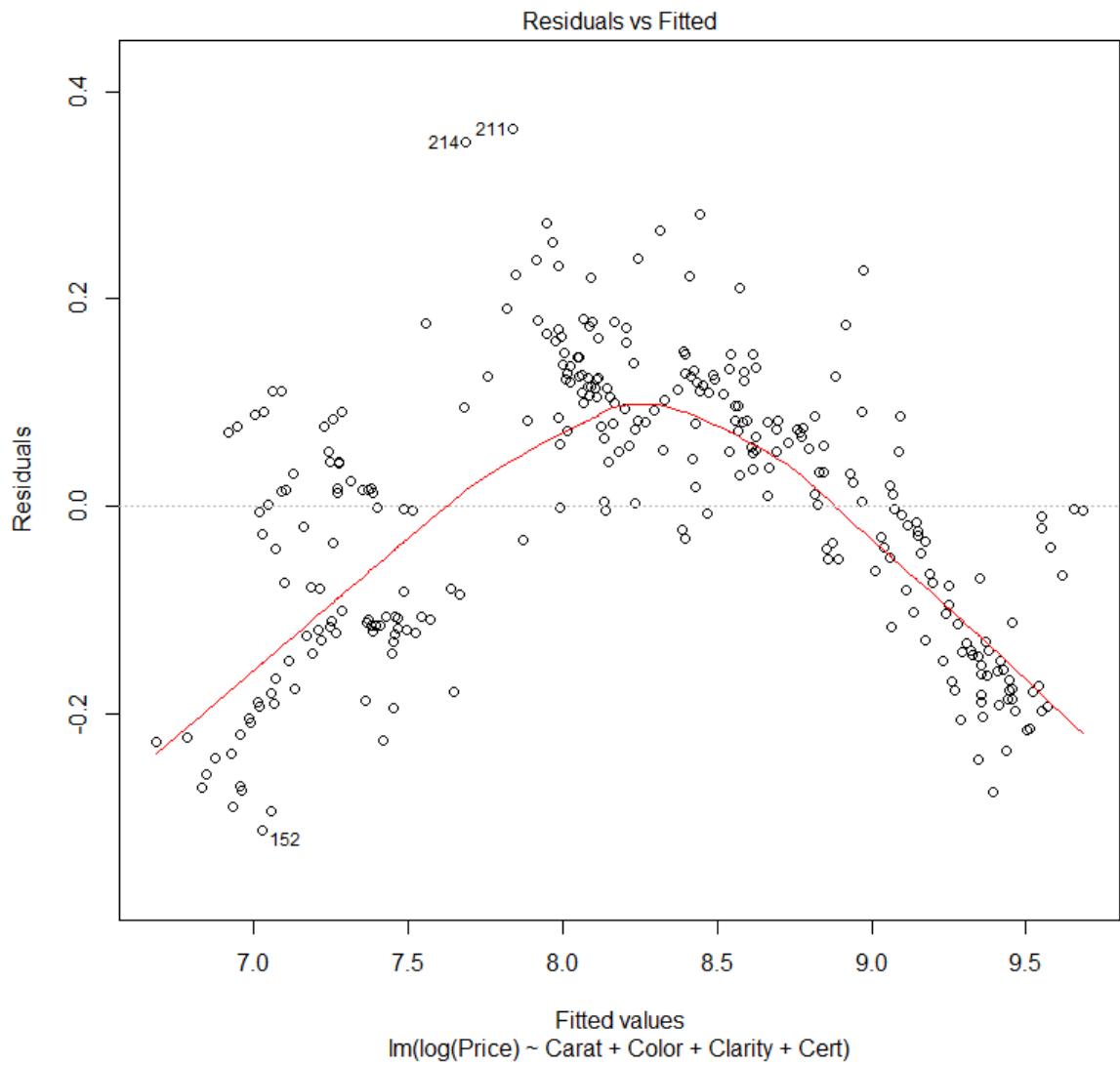
	Estimate	Std. Error	t value	Pr(> t )		
(Intercept)	6.8011915	0.0496111	137.090	< 2e-16 ***		
Carat	2.8550130	0.0369676	77.230	< 2e-16 ***		
ColorE	-0.0295093	0.0404611	-0.729	0.46638		
ColorF	-0.1063582	0.0379807	-2.800	0.00544 **		
ColorG	-0.2063494	0.0389848	-5.293	2.35e-07 ***		
ColorH	-0.2878756	0.0394752	-7.293	2.81e-12 ***		
ColorI	-0.4165565	0.0413818	-10.066	< 2e-16 ***		
ClarityVS1	-0.2019313	0.0310634	-6.501	3.41e-10 ***		
ClarityVS2	-0.2985406	0.0333027	-8.964	< 2e-16 ***		
ClarityVVS1	-0.0007058	0.0311121	-0.023	0.98192		
ClarityVVS2	-0.0966174	0.0289396	-3.339	0.00095 ***		
CertHRD	-0.0088557	0.0208641	-0.424	0.67155		
CertIGI	-0.1827107	0.0249516	-7.323	2.33e-12 ***		
---						
Signif. codes:	0 ‘***’	0.001 ‘**’	0.01 ‘*’	0.05 ‘.’	0.1 ‘ ’	1

Residual standard error: 0.1382 on 295 degrees of freedom

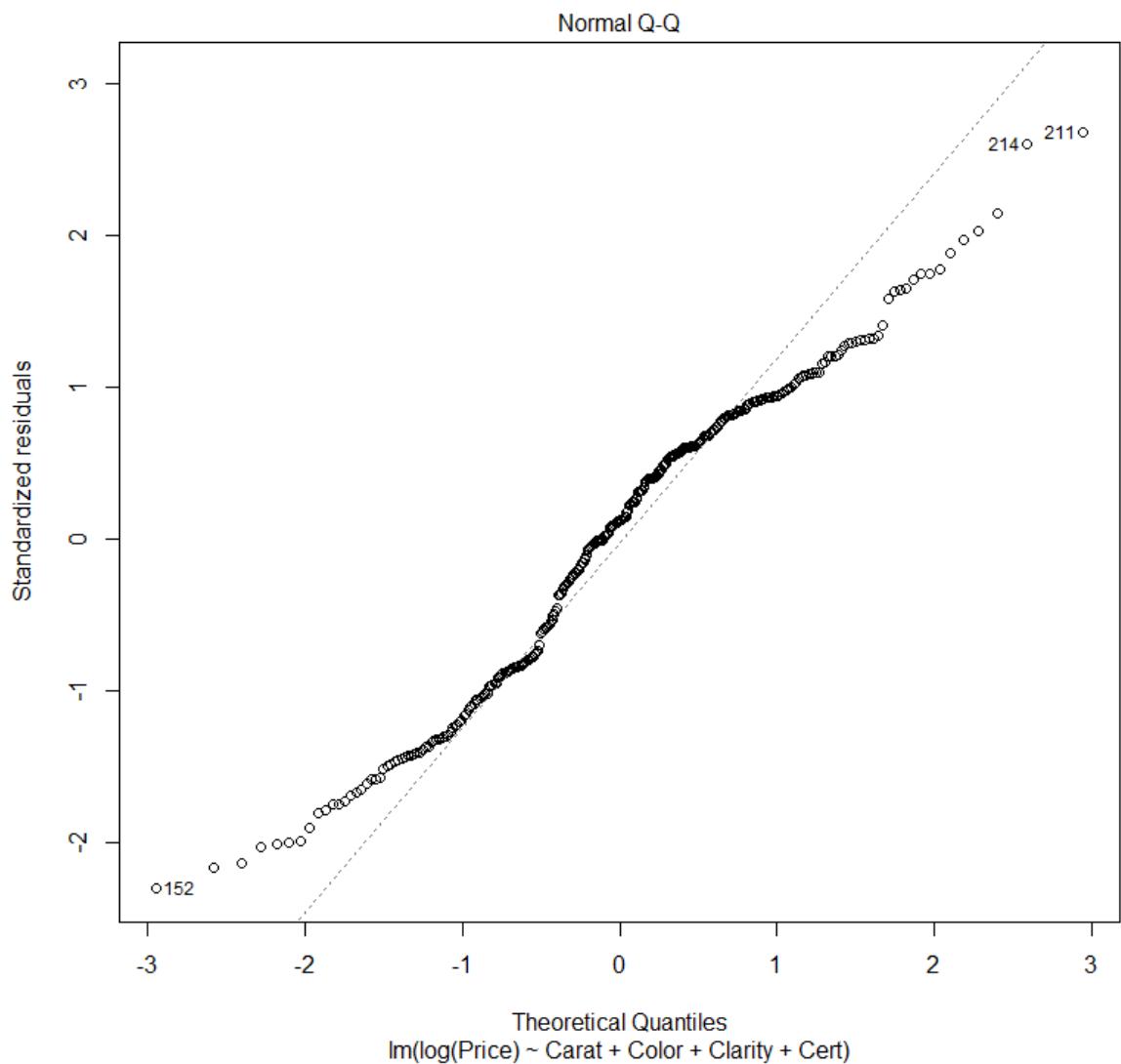
Multiple R-squared: 0.9723, Adjusted R-squared: 0.9712

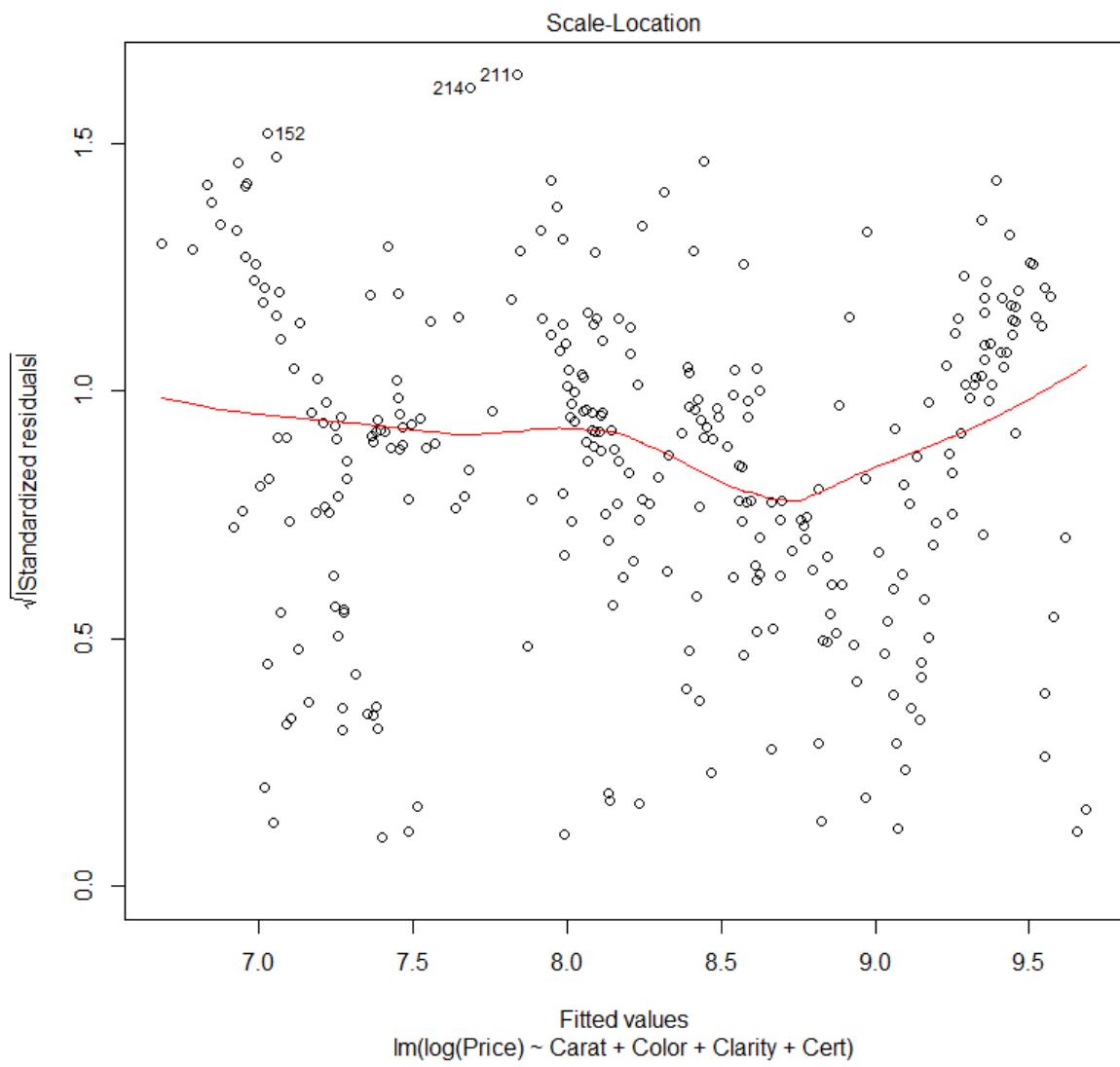
F-statistic: 863.6 on 12 and 295 DF, p-value: < 2.2e-16

```
> plot(sparkly.fit2,which=1)
```



```
> plot(sparkly.fit2,which=2)
```





> # Log-transformation of Price has done nothing to curb the clear, systematic, non-linear appearance of the residuals. A non-linear trend still appears very prominent.

> # However, the log-transformation has reignited in the extreme points and the severity of the non-normality to a certain extent.

> #(1)

> sparkly.fit3 <- lm(log(Price) ~ Carat + I(Carat^2) + Color + Clarity + Cert, data = diamonds)

> summary(sparkly.fit3)

Call:

$\text{lm}(\text{formula} = \log(\text{Price}) \sim \text{Carat} + \text{I}(\text{Carat}^2) + \text{Color} + \text{Clarity} + \text{Cert}, \text{data} = \text{diamonds})$

Residuals:

Min	1Q	Median	3Q	Max
-0.15411	-0.04120	-0.00911	0.04543	0.14158

Coefficients:

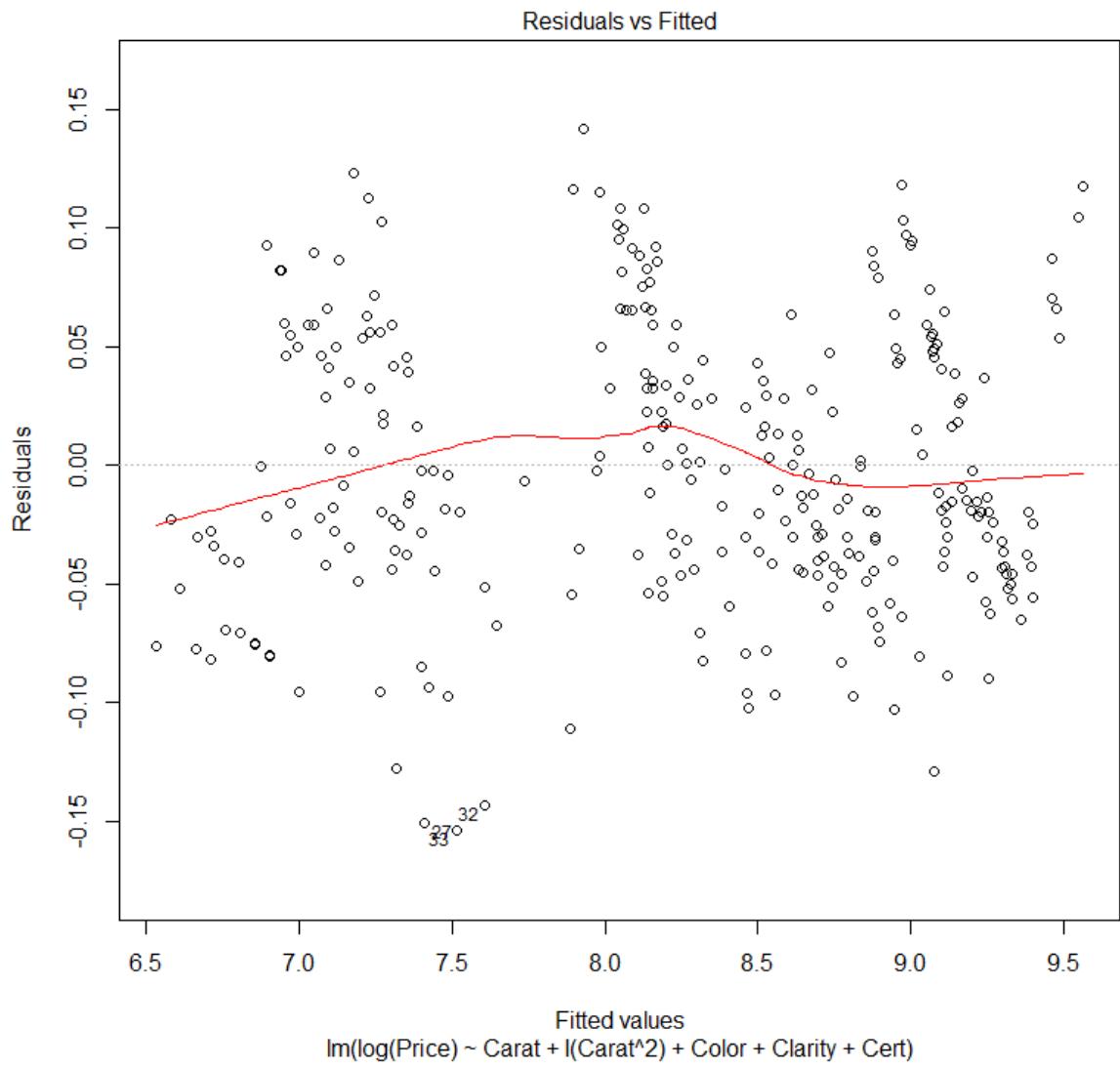
	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	6.075351	0.029202	208.049	< 2e-16 ***
Carat	5.670616	0.079284	71.523	< 2e-16 ***
I(Carat^2)	-2.102922	0.058022	-36.243	< 2e-16 ***
ColorE	-0.079247	0.017387	-4.558	7.59e-06 ***
ColorF	-0.155991	0.016328	-9.554	< 2e-16 ***
ColorG	-0.245033	0.016734	-14.643	< 2e-16 ***
ColorH	-0.339098	0.016969	-19.983	< 2e-16 ***
ColorI	-0.442606	0.017742	-24.947	< 2e-16 ***
ClarityVS1	-0.244470	0.013358	-18.301	< 2e-16 ***
ClarityVS2	-0.320183	0.014279	-22.424	< 2e-16 ***
ClarityVVS1	-0.094008	0.013574	-6.926	2.74e-11 ***
ClarityVVS2	-0.176701	0.012592	-14.032	< 2e-16 ***
CertHRD	-0.006223	0.008938	-0.696	0.4868
CertIGI	-0.025413	0.011536	-2.203	0.0284 *
---				
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1				

Residual standard error: 0.0592 on 294 degrees of freedom

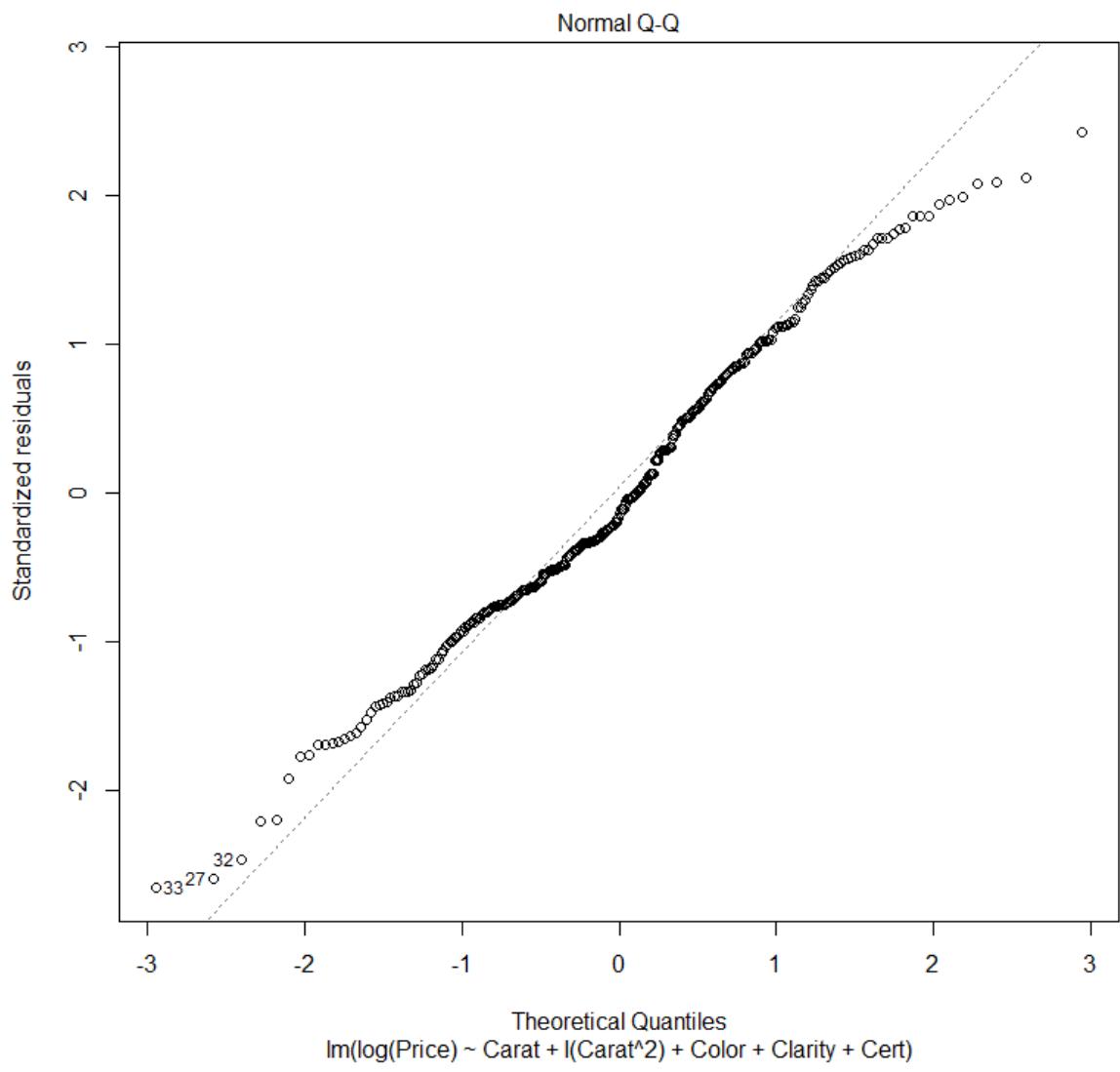
Multiple R-squared: 0.9949, Adjusted R-squared: 0.9947

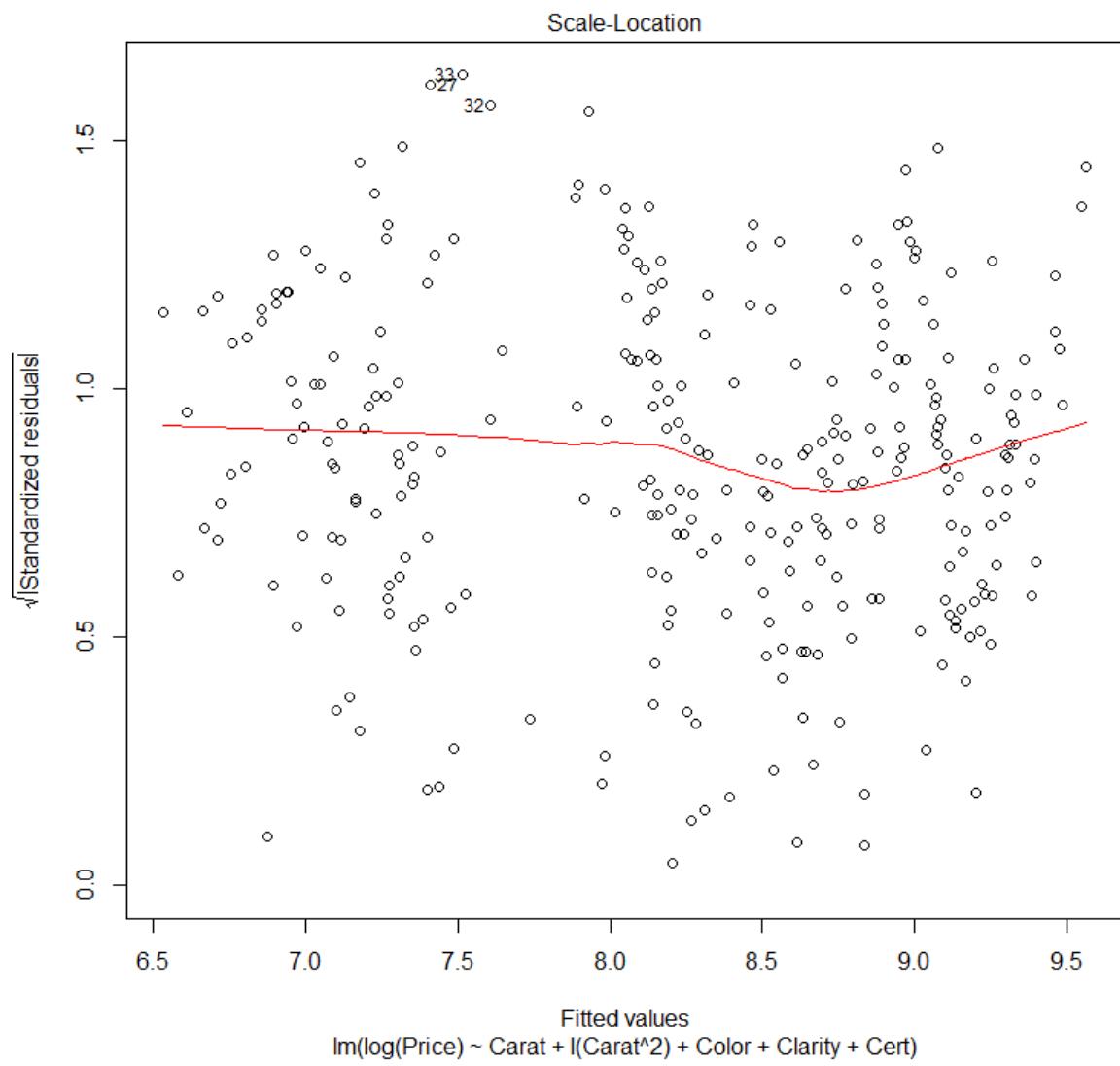
F-statistic: 4445 on 13 and 294 DF, p-value: < 2.2e-16

```
> plot(sparkly.fit3,which=1)
```



```
> plot(sparkly.fit3,which=2)
```





Fitted values  
 $\text{lm}(\log(\text{Price}) \sim \text{Carat} + \text{I}(\text{Carat}^2) + \text{Color} + \text{Clarity} + \text{Cert})$

> # Including an order 2 polynomial term in Carat has eliminated much of the concern of the non-linear curvature.

> # Normality is more apparent now, and there's little if any concern of heteroscedasticity.

> # This third model is by far the most appropriate of the three (with respect to the residual diagnostics) when it comes to modeling the cost of diamonds in light of the available data.

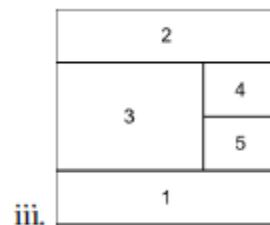
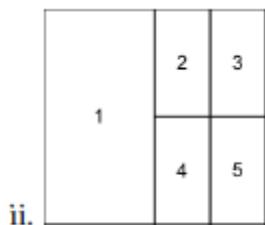
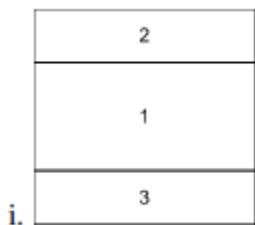
## PART 5

# Chapter 23: Advanced Plot Customization

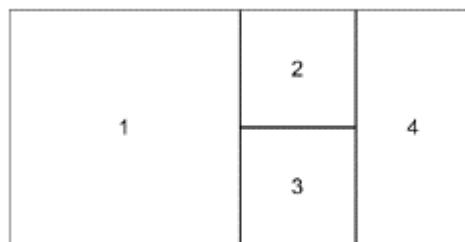
Estimated time to complete: **30 minutes**

## Exercise 23.1

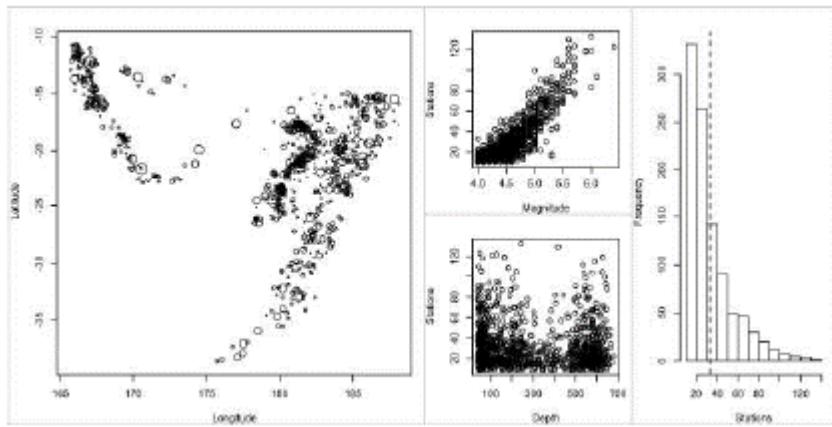
- In Section 20.5.4 (page 478), I gave you code showing a simple linear model fitted with a categorical predictor being treated as continuous (the *mtcars* data with *mpg* as the response and *cyl* as the explanatory variable). Reproduce the side-by-side boxplots and the scatterplot (with fitted line) from Figure 20-6, but this time, use *mfrow* to present the two plots as a vertical column in one device.
- Create the appropriate layout matrices to reproduce the following three plots (as they appear in a square device):



- By opening a new device of dimension  $9 \times 4.5$  inches, set the following layout:



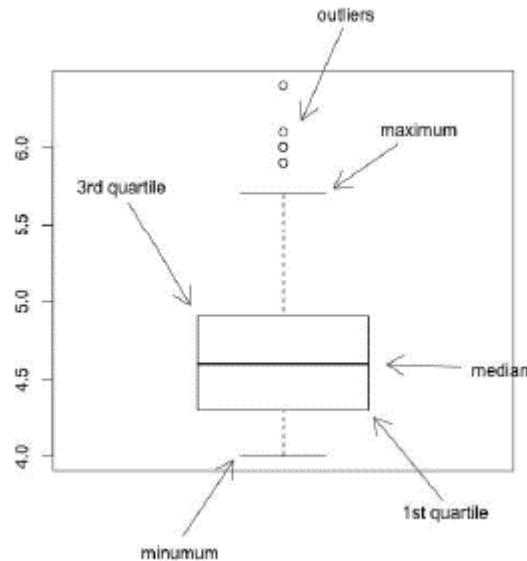
Then, produce the following combined set of plots exactly:



To achieve this, note the following:

- After you open the device and setting the layout, the plot margins should be reset to four lines, four lines, two lines, and one line of space on the bottom, left, top, and right, respectively.
- After each plot, add a gray box corresponding to the figure region to achieve the visible partitions.
- Plots 1 and 4 are the same as the two plots shown in Figures 23-1 and 23-2.
- Plots 2 and 3 are scatterplots showing the number of detecting stations on the y-axis, with magnitude and depth on the x-axis, respectively.
- Do not place main titles on any plots, and ensure the axis titles are neat (that is, compared to their defaults).
  - d. Write a little R function named *interactive.arrow*. The purpose of this function is to superimpose an arrow upon any base R plot using two clicks of your mouse. The details are as follows:
    - The crux of your function will be the use of *locator* to read exactly two mouse clicks. You may assume a suitable active graphics device is already open whenever the function is called. The first click should mean the beginning of the arrow, and the second click should mean the tip of the arrow (where it's pointing to).
    - In the function, the coordinates returned by *locator* should be passed to *arrows* to do the actual drawing.
    - The function should take an ellipsis as its first argument, intended to hold additional arguments to be passed directly to *arrows*.
    - The function should take an optional logical argument *label*, which defaults to *NA* but should be intended to have an optional character string. If *label* is not *NA*, then *locator* should be invoked once more (separately, after drawing the arrow) to select exactly one coordinate. That point will be passed to *text* so that the user can additionally place the character string given to *label* as an annotation (intended to be for the interactively placed arrow). The call to *text* should consistently allow completely relaxed clipping (in other words, any text added in this fashion will still be visible in the figure region and outer margins, if there are any).

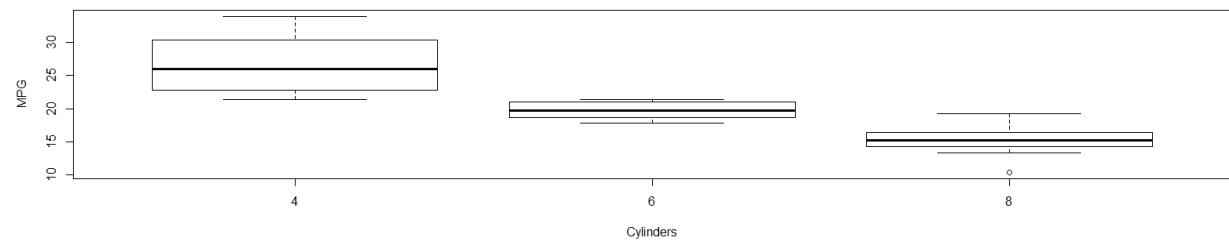
Take another look at the rightmost plot of Figure 14-6 on page 298, a stand-alone boxplot of the magnitude data from the *quakes* data frame. Arrows and labels were superimposed externally pointing out the various statistics summarized by a boxplot. Create the same boxplot and use *interactive.arrow* to annotate the same features to your own satisfaction (you'll likely have to use the ellipsis to relax the clipping associated with each arrow). My result is given here:



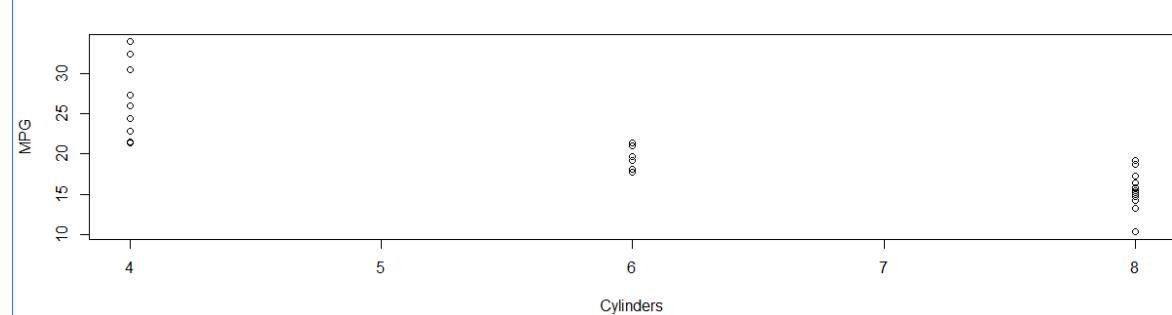
## Solution 23.1

#(a)

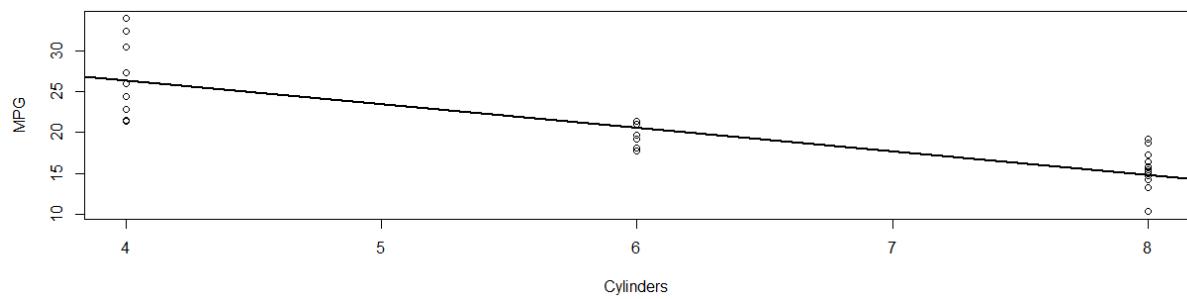
```
>par(mfrow=c(2,1))
>boxplot(mtcars$mpg~mtcars$cyl,xlab="Cylinders",ylab="MPG")
```



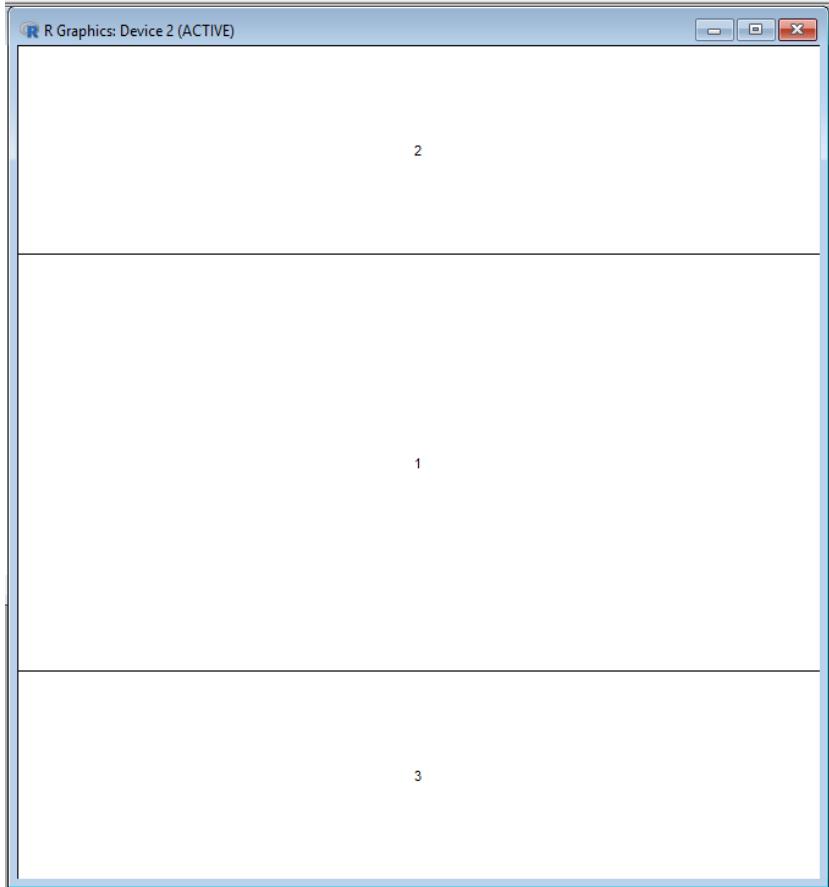
```
>carfit <- lm(mpg~cyl,data=mtcars)
>plot(mtcars$mpg~mtcars$cyl,xlab="Cylinders",ylab="MPG")
```



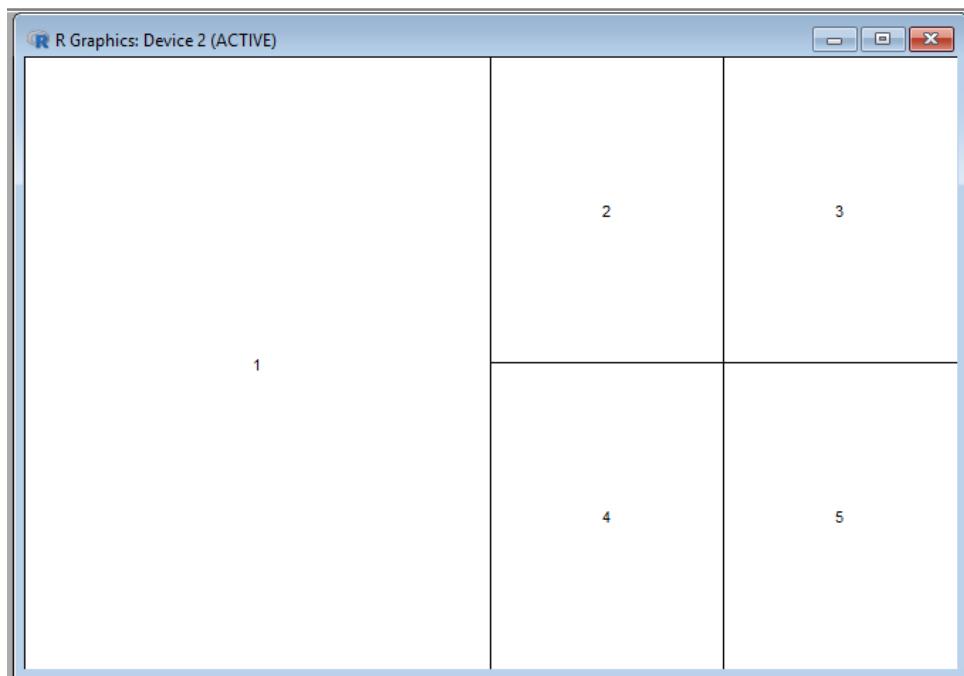
```
>abline(carfit,lwd=2)
```



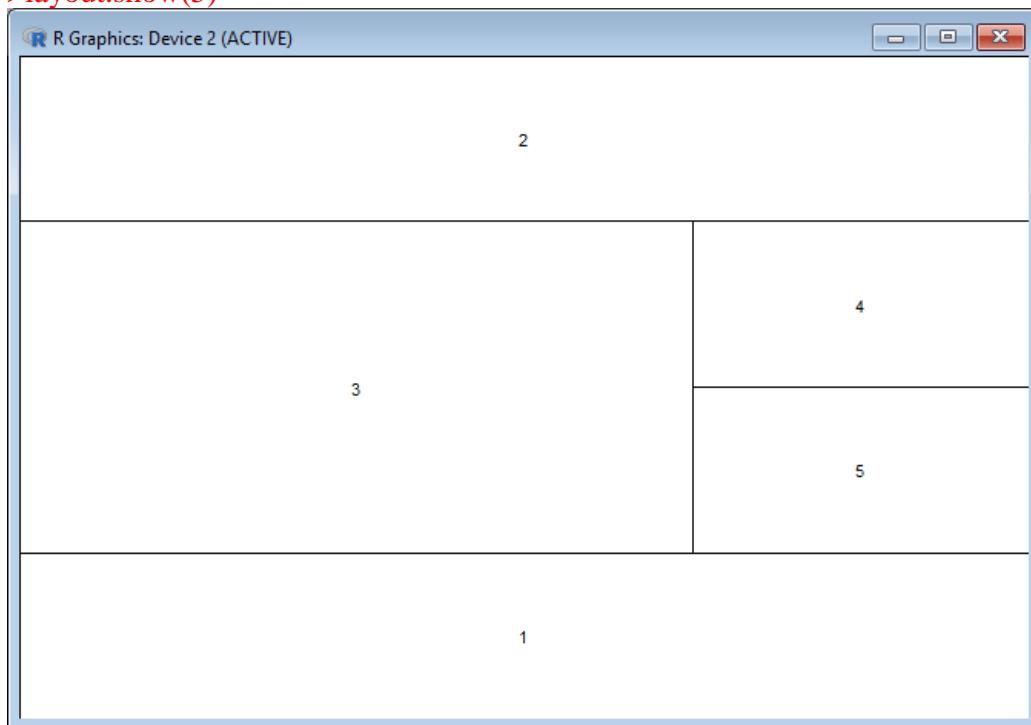
```
##(i)
>lay.mat <- cbind(c(2,1,1,3),c(2,1,1,3))
>lay.mat
 [,1] [,2]
[1,] 2 2
[2,] 1 1
[3,] 1 1
[4,] 3 3
>layout(lay.mat)
>layout.show(3)
```



```
##(ii)
>lay.mat <- rbind(c(1,1,2,3),c(1,1,4,5))
>lay.mat
>layout(lay.mat)
 [,1] [,2] [,3] [,4]
[1,] 1 1 2 3
[2,] 1 1 4 5
>layout.show(5)
```

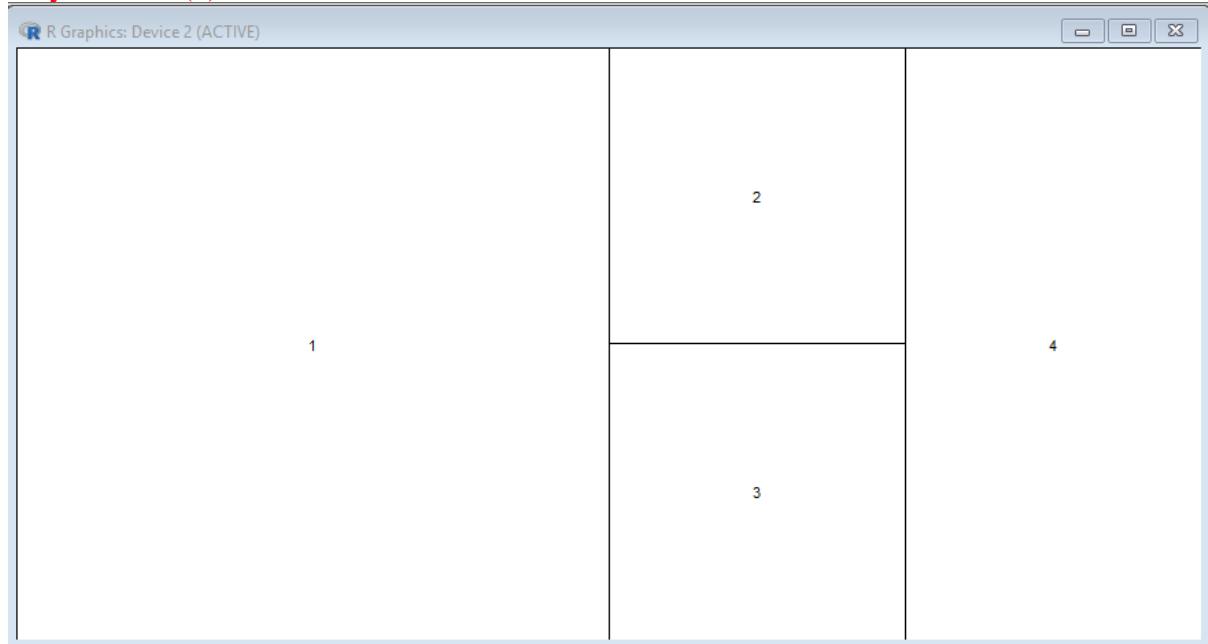


```
##(iii)
>lay.mat <- cbind(c(2,3,3,1),c(2,3,3,1),c(2,4,5,1))
>lay.mat
>layout(lay.mat)
 [,1] [,2] [,3]
[1,]  2   2   2
[2,]  3   3   4
[3,]  3   3   5
[4,]  1   1   1
>layout.show(5)
```

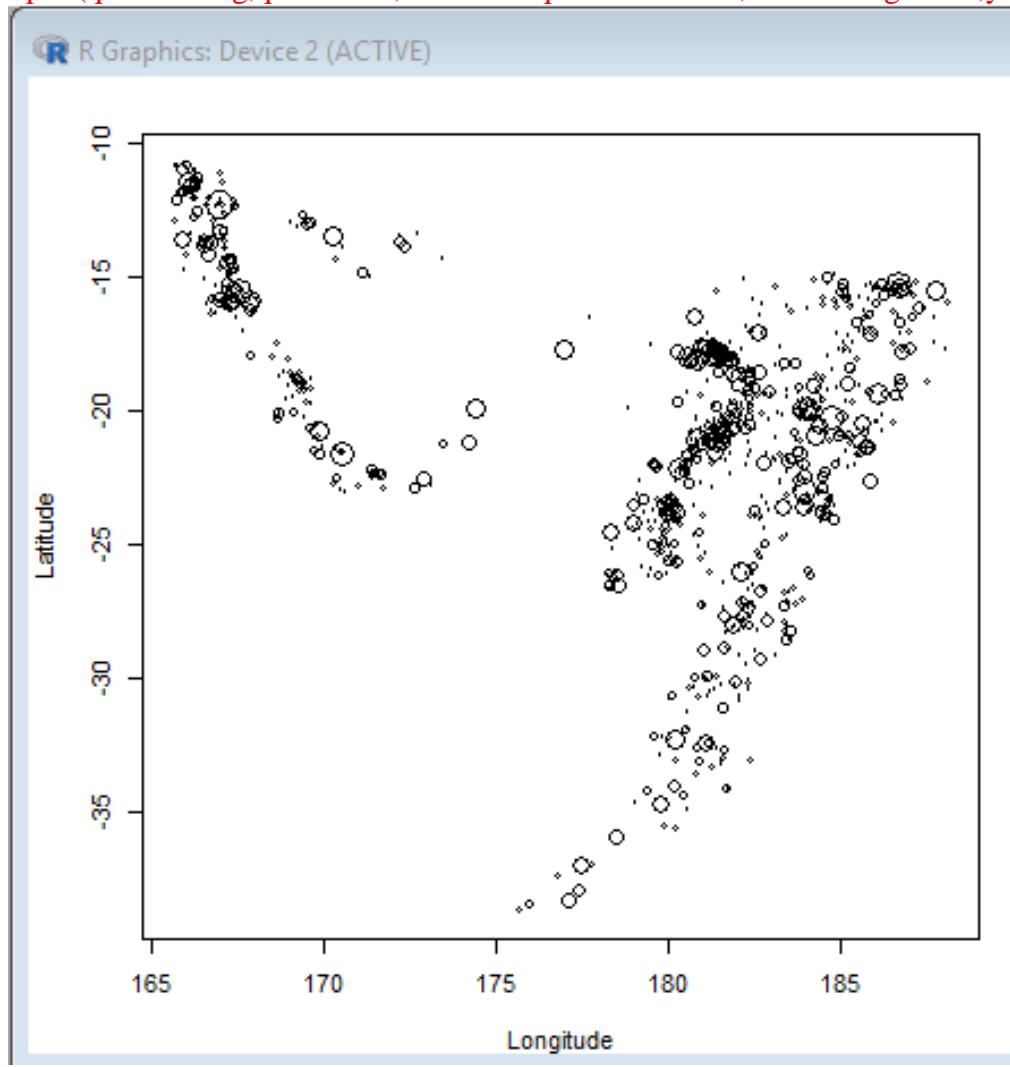


```
#(c)
>dev.new(width=9,height=4.5)
>lay.mat <- rbind(c(1,1,2,4),c(1,1,3,4))
```

```
>layout(lay.mat)
>layout.show(4)
```

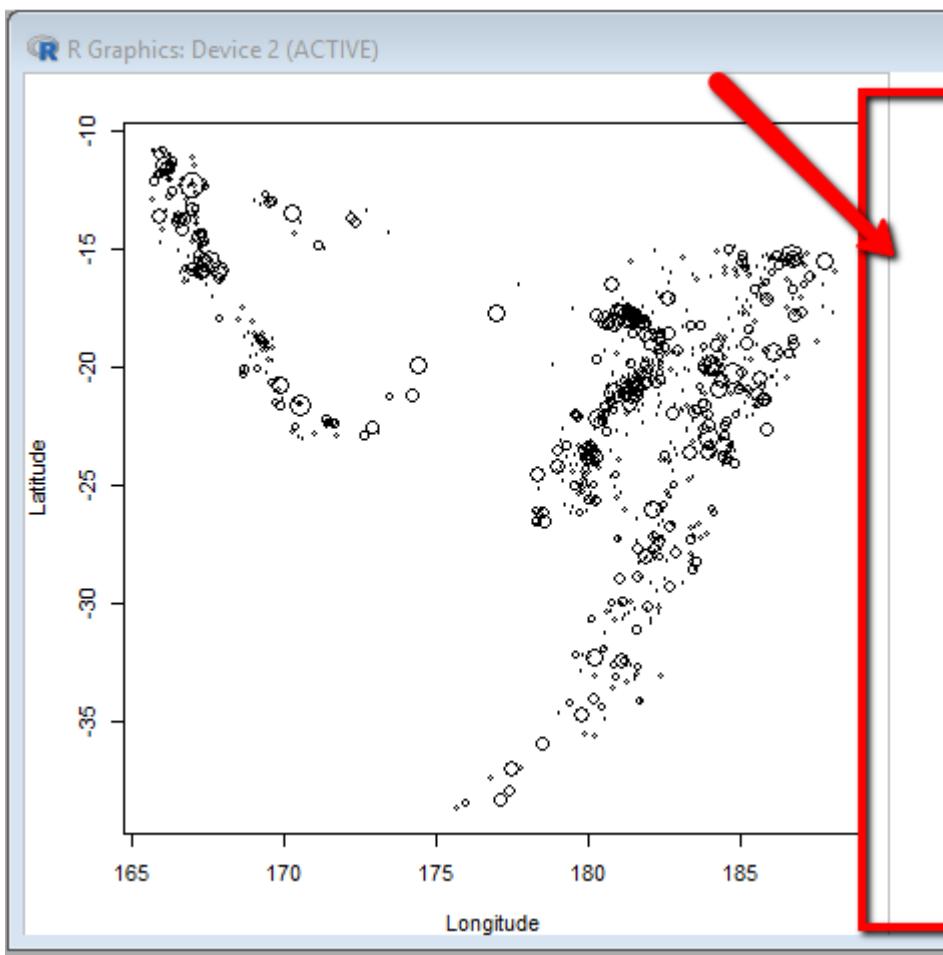


```
>par(mar=c(4,4,2,1))
>plot(quakes$long,quakes$lat,cex=0.02*quakes$stations,xlab="Longitude",ylab="Latitude")
```

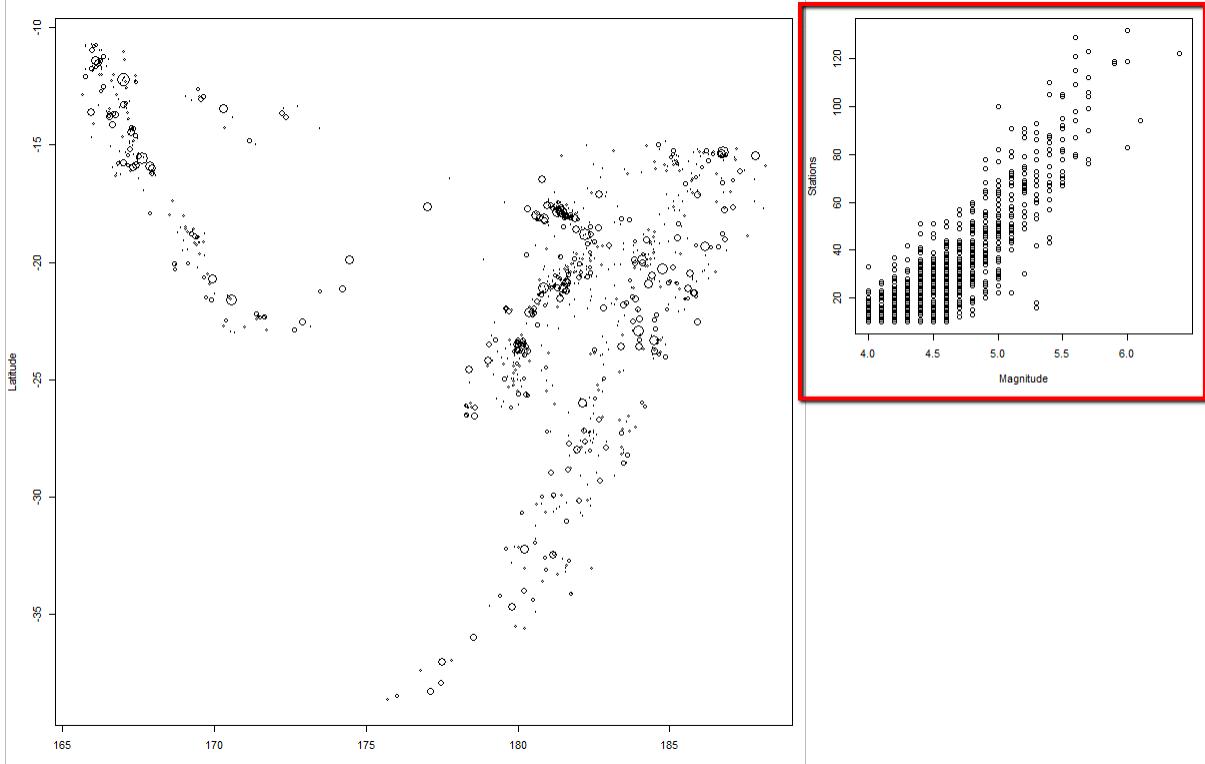


```
>box(which="figure",col="gray")
```

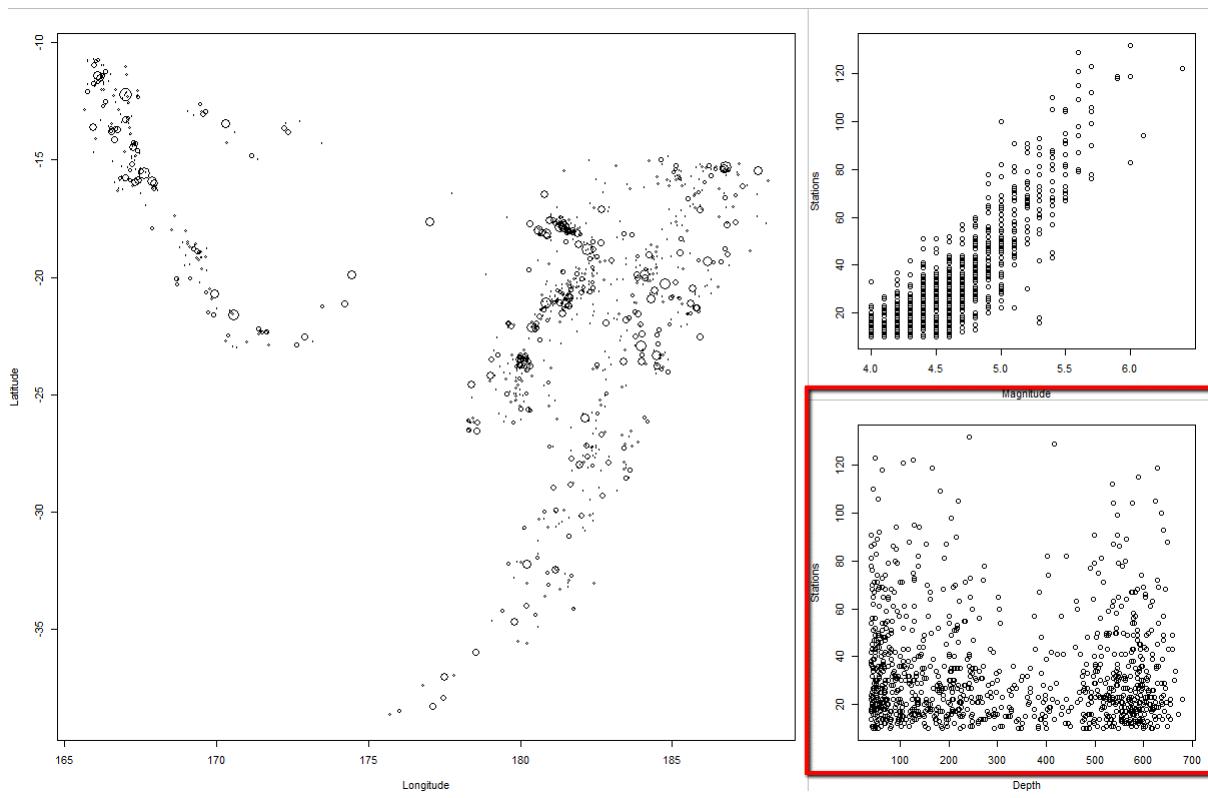




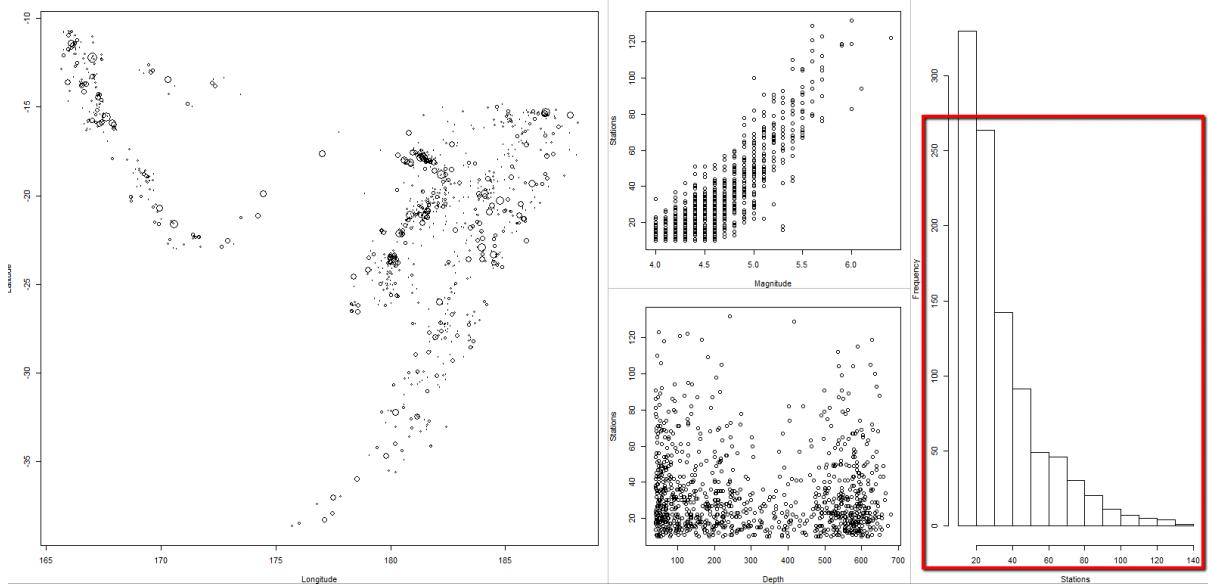
```
>plot(quakes$mag,quakes$stations,xlab="Magnitude",ylab="Stations")
```



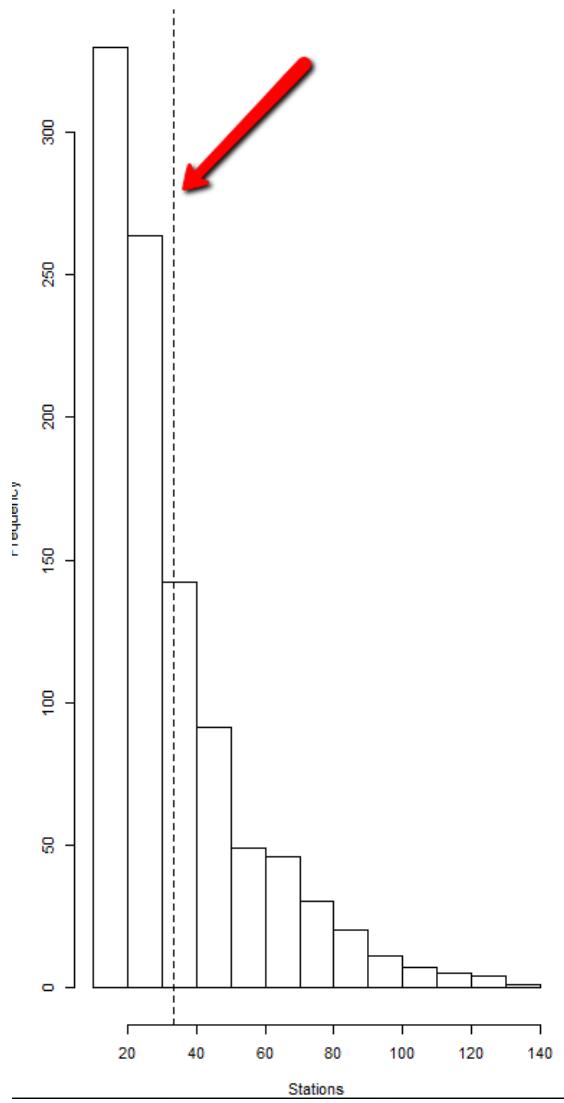
```
>box(which="figure",col="gray")
>plot(quakes$depth,quakes$stations,xlab="Depth",ylab="Stations")
```



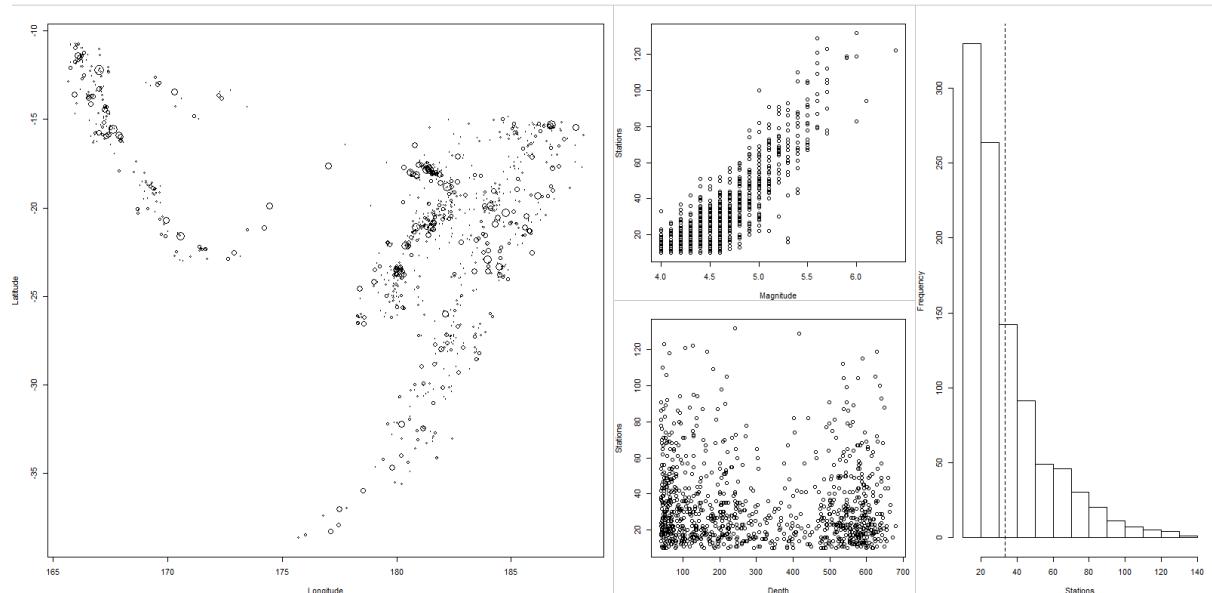
```
>box(which="figure",col="gray")
>hist(quakes$stations,main="",xlab="Stations")
```



```
>abline(v=mean(quakes$stations),lty=2)
```



```
>box(which="figure",col="gray")
```



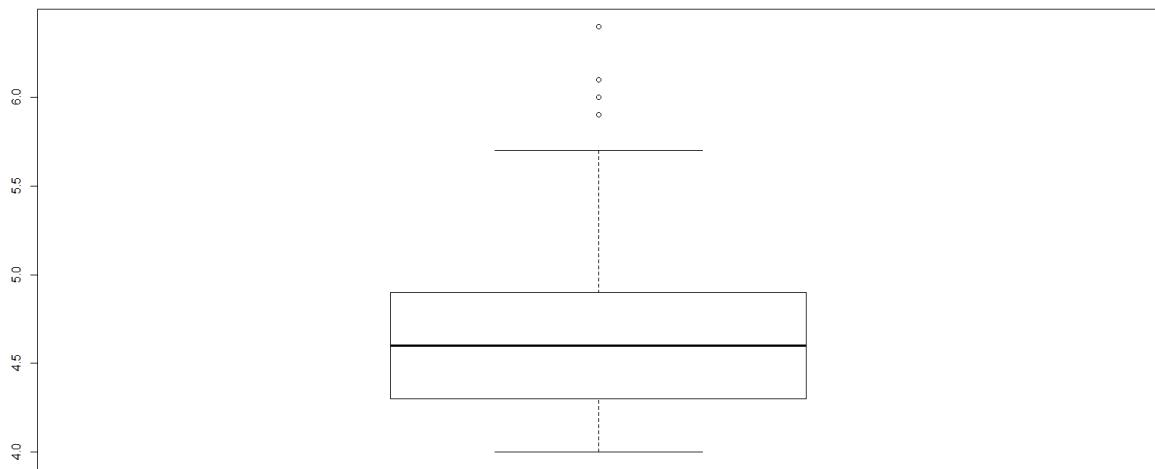
#(d)

```
>interactive.arrow <- function(...,label=NA){
```

```

arr.pts <- locator(2)
arrows(x0=arr.pts$x[1],y0=arr.pts$y[1],x1=arr.pts$x[2],y1=arr.pts$y[2],...)
if(!is.na(label)){
  lab.pt <- text(locator(1),label=label,xpd=NA)
}
}
boxplot(quakes$mag)

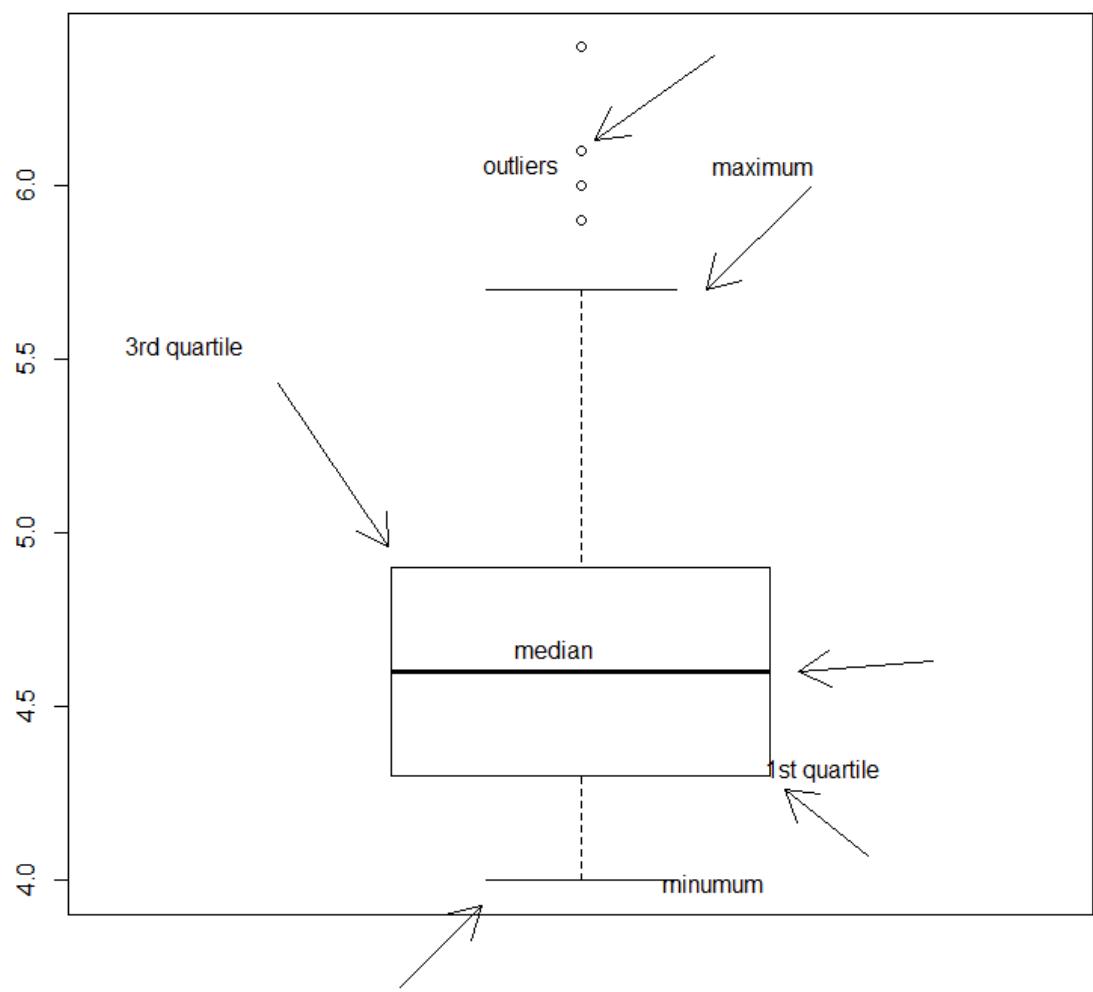
```



```

>interactive.arrow(xpd=TRUE,label="minimum")
>interactive.arrow(xpd=TRUE,label="1st quartile")
>interactive.arrow(xpd=TRUE,label="median")
>interactive.arrow(xpd=TRUE,label="3rd quartile")
>interactive.arrow(xpd=TRUE,label="maximum")
>interactive.arrow(xpd=TRUE,label="outliers")

```



## Exercise 23.2

For the following tasks, you'll work with the diamond-pricing data as analyzed by Chu (2001). You'll need an Internet connection for this.

Read the data in and name the columns as you've done previously with the following:

```
R> dia.url <- "http://www.amstat.org/publications/jse/v9n2/4cdata.txt"
```

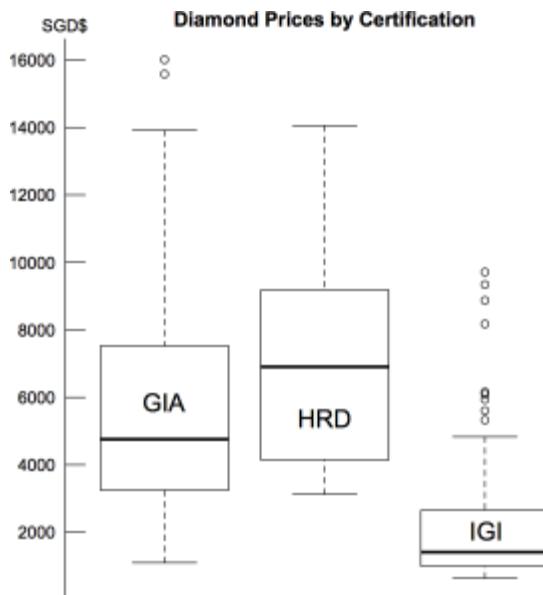
```
R> diamonds <- read.table(dia.url)
```

```
R> names(diamonds) <- c("Carat", "Color", "Clarity", "Cert", "Price")
```

a. Open a new graphics device of  $6 \times 6$  inches. Initialize the margin spacing to be zero, four, two, and zero lines on the bottom, left, top, and right of the plot region, respectively. Then, complete the following:

- i. Produce side-by-side boxplots of the diamond prices in Singapore dollars (SGD\$) split by certification. Suppress all axes and the surrounding box—note that the *boxplot* command requires you to set *frame=FALSE* for suppressing the box (as opposed to *bty="n"* in *plot*). Use the same command to provide an appropriate title.
- ii. Next, insert a vertical axis. The axis should have tick marks ranging from SGD\$0 to SGD\$18000, progressing in steps of SGD\$2000. However, the axis should be clipped to the plotting region. The axis tick marks should point inward and be one line in length. The axis labels should sit only half a line away from the axis and should be horizontally readable.
- iii. Finally, use *locator* in conjunction with *text* to add an appropriate title sitting at the top of the y-axis; note that clipping will need to be relaxed. Use the same approach to add text, sitting inside each boxplot, denoting the corresponding certification (GIA, HRD, or IGI).

My version of the plot looks like this:



b. Now, open a new graphics device of  $8 \times 7$  inches. Set the figure margins to be two, five, three, and five lines on the bottom, left, top, and right, respectively. Also allow one line of outer margin space on each side other than the bottom, which should get two lines of outer margin.

- i. Produce a scatterplot of diamond price on the vertical axis and carat weight on the horizontal axis. Use the colors red, green, and blue to distinguish the points according to certification. Suppress all axes, boxes, labels, and titles in the initial plot, but then add a U-shaped box.
- ii. Add the horizontal axis. Use *axis* to place tick marks at an evenly spaced sequence of carat values between 0.2 and 1.1, in steps of 0.1. Use a bold, italic, sans-style font for the

labels and adjust the labels to be only half a line from the axis. Then add smaller, outward-facing tick marks between the existing ones. To do this, make a second call to the `axis` function and place the ticks at a sequence of values from 0.15 to 1.05 at steps of 0.1. Set these secondary tick marks to have a length of one-quarter of a line and suppress the axis labels.

iii. Add the vertical axes. On the left, ticks should appear at SGD\$1000–17000. Labels should be horizontally readable and in the same font style as the horizontal axis. On the right, axis ticks should be made in the equivalent of US dollars (USD\$) at the sequence USD\$1000–11000 in steps

of USD\$1000 and should be labeled as such. To do this, use the conversion  $\text{USD\$} = 1.37 \times \text{SGD\$}$ . Label orientation and font should match the other axes.

iv. Fit a linear model of price on a quadratic polynomial of carat weight for the data. Provide a prediction of the model for a sequence of carat values spanning the range of the observed values; include estimation of a 95 percent prediction interval.

Use this information to superimpose a gray solid line for the fitted values and gray dashed lines for the prediction interval upon the scatterplot.

v. Set up expression objects for labeling the approximate US dollar conversion and the regression equation. Name the conversion `expr1`; it should look something like  $\text{USD\$} \approx 1.37 \times \text{SGD\$}$ . The regression equation should look similar to  $\text{Price} = \beta_0 + \beta_1 \text{Carat} + \beta_2 \text{Carat}^2$ ; name it `expr2`.

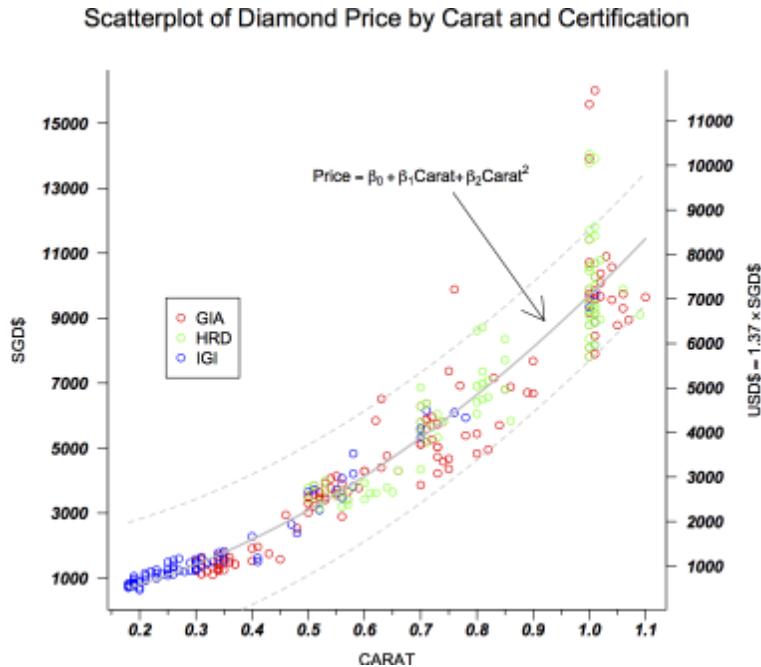
vi. Use `mtext` to add an appropriate main title and titles for all three individual axes. You may need to experiment a little with line depth for each one, as well as whether to write in the outer margin or the figure margin, depending on your own spacing preference. The rightmost axis title should make use of `expr1`.

vii. Either via trial and error to find appropriate coordinates or by using the `interactive.arrow` function from Exercise 23.1,

place an arrow pointing to the fitted polynomial regression line and label it with `expr2`.

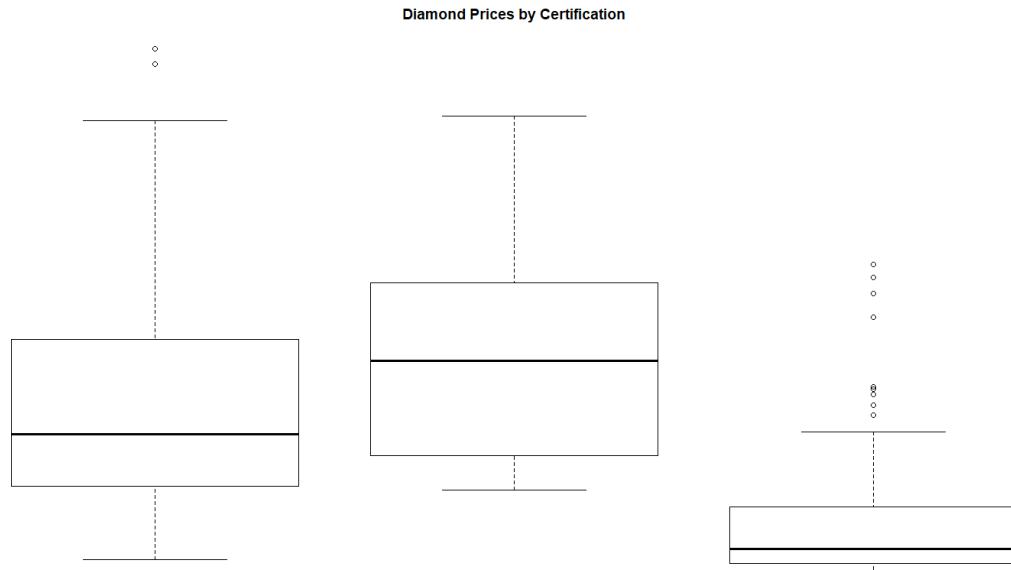
viii. Finally, use a call to `locator` to place a legend in any appropriate location, referencing the color of the points according to the appropriate certification.

My version of the plot looks like this:



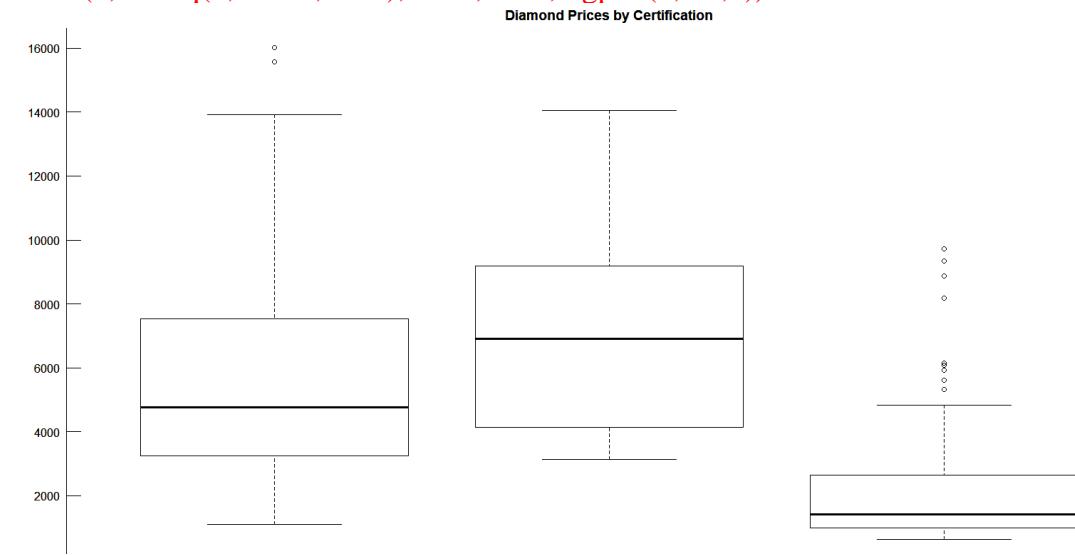
## Solution 23.2

```
>dia.url <- "http://www.amstat.org/publications/jse/v9n2/4cdatal.txt"
>diamonds <- read.table(dia.url)
>names(diamonds) <- c("Carat", "Color", "Clarity", "Cert", "Price")
#(a)
>dev.new(width=6,height=6)
>par(mar=c(0,4,2,0))
##(i)
>boxplot(diamonds$Price~diamonds$Cert,axes=FALSE,frame=FALSE,main="Diamond Prices by Certification")
```



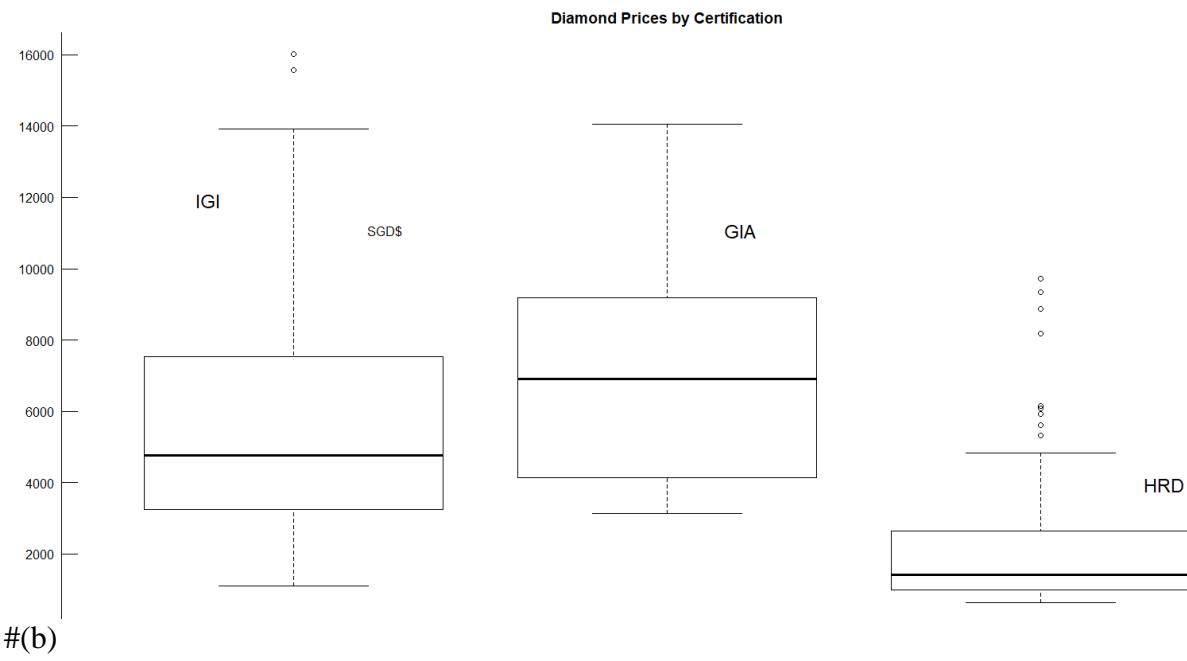
##(ii)

```
>axis(2,at=seq(0,18000,2000),las=1,tcl=1,mgp=c(3,0.5,0))
```



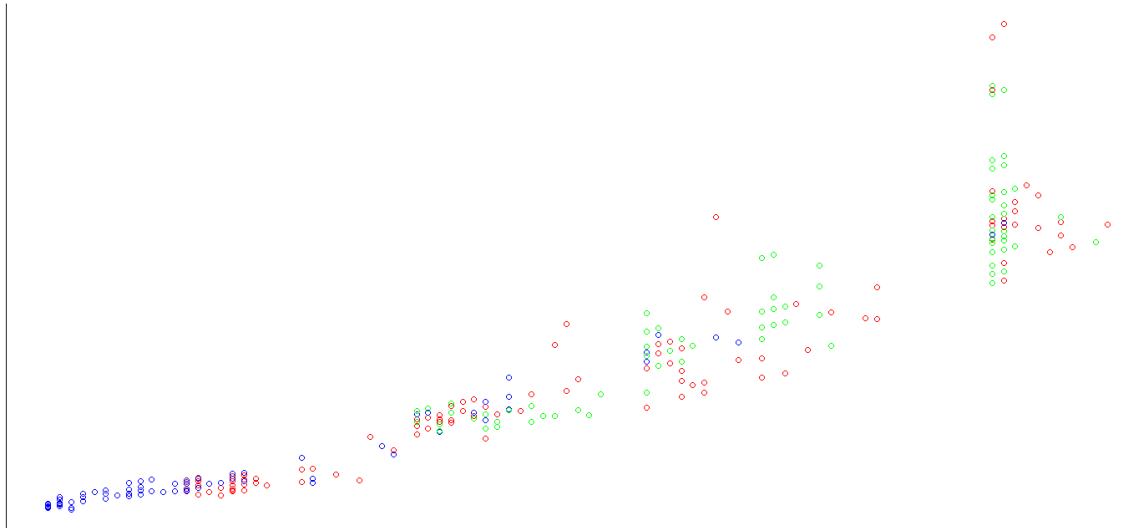
##(iii)

```
>text(locator(1),"SGD$",xpd=TRUE)
>text(locator(1),"GIA",cex=1.5)
>text(locator(1),"HRD",cex=1.5)
>text(locator(1),"IGI",cex=1.5)
```



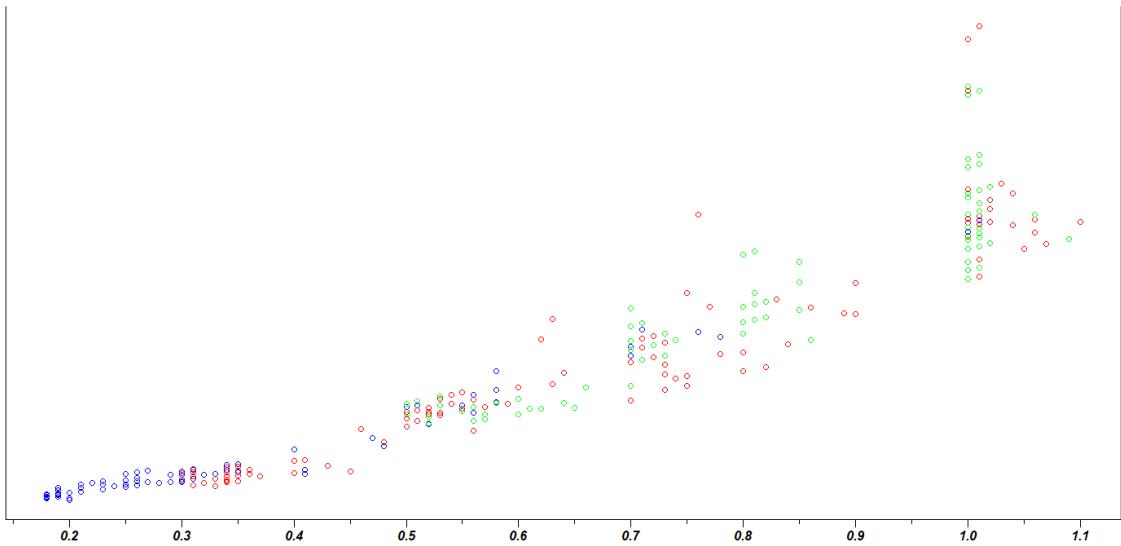
#(b)  
1

```
>dev.new(width=8,height=7)
>par(mar=c(2,5,3,5),oma=c(2,rep(1,3)))
##(i)
>plot(diamonds$Price~diamonds$Carat,col=c("red","green","blue")[as.numeric(diamonds$Cert)],
axes=FALSE,ann=FALSE)
>box(bty="u")
```



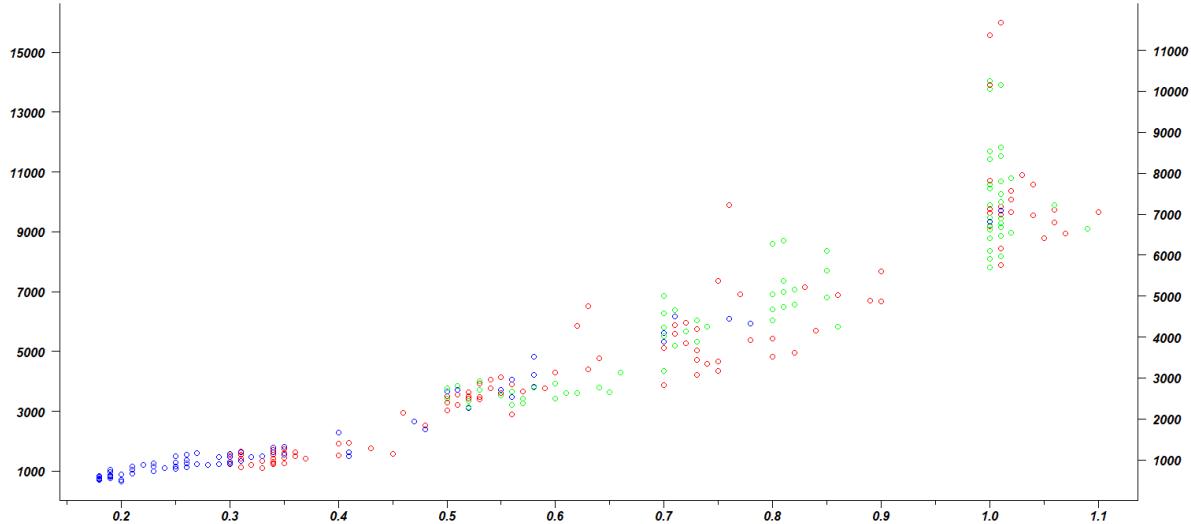
##(ii)

```
>axis(1,at=seq(0.2,1.1,0.1),font=4,mgp=c(3,0.5,0))  
>axis(1,at=seq(0.15,1.05,0.1),tcl=-0.25,labels=FALSE)
```



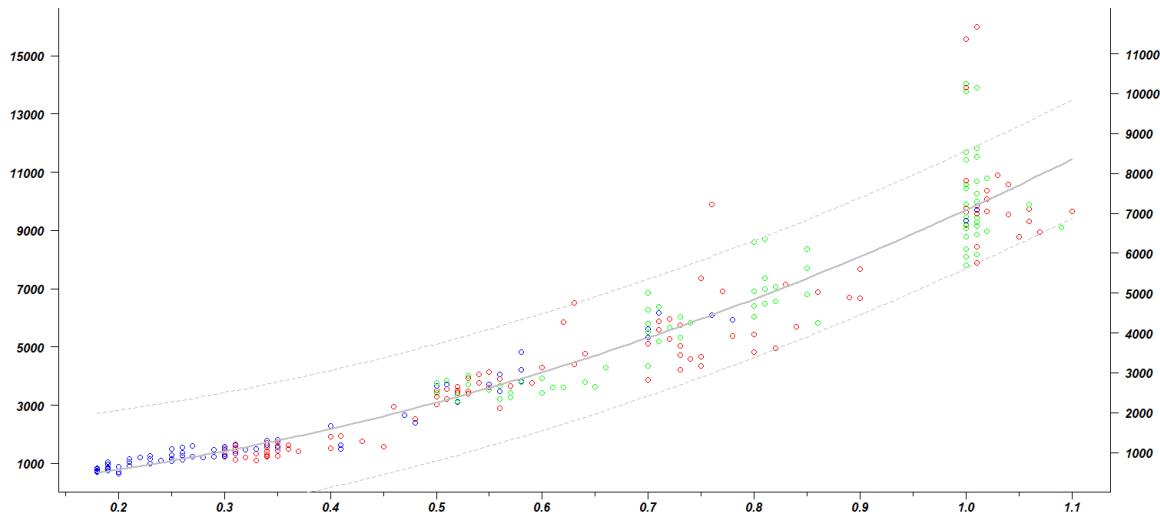
##(iii)

```
>axis(2,at=seq(1000,17000,2000),las=1,font=4)
>axis(4,at=seq(1000,11000,1000)*1.37,labels=seq(1000,11000,1000),las=1,font=4)
```



##(iv)

```
>dia.fit <- lm(Price~Carat+I(Carat^2),data=diamonds)
>carat.seq <- seq(min(diamonds$Carat),max(diamonds$Carat),length=100)
>dia.pred <- predict(dia.fit,newdata=data.frame(Carat=carat.seq),interval="prediction")
>lines(carat.seq,dia.pred[,1],col="gray",lwd=2)
>lines(carat.seq,dia.pred[,2],col="gray",lty=2)
>lines(carat.seq,dia.pred[,3],col="gray",lty=2)
```

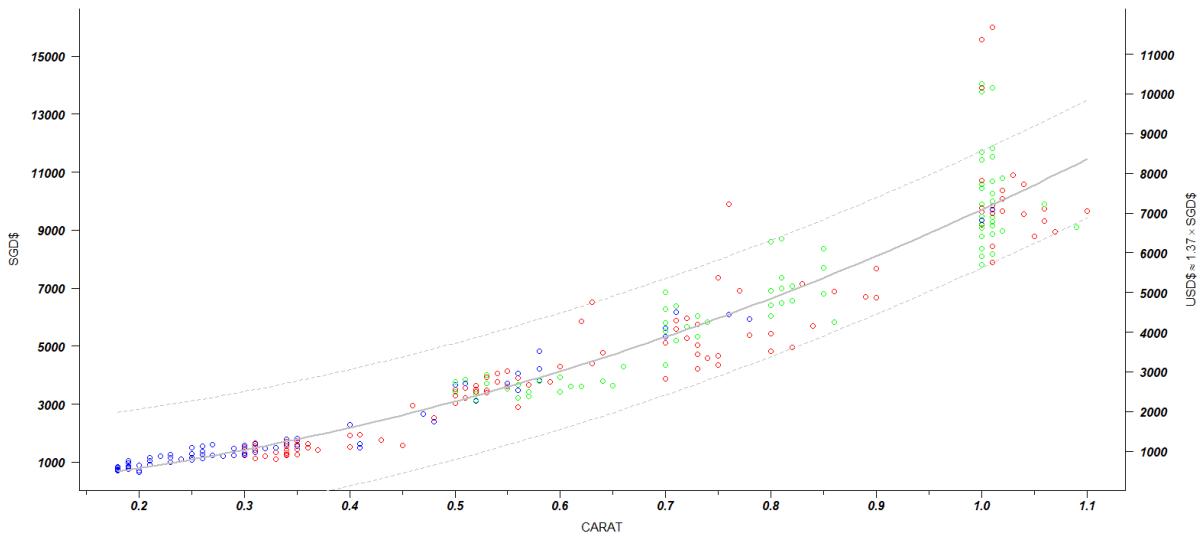


```

##(v)
>expr1 <- expression("USD$"%~~% 1.37% *% "SGD$")
>expr2 <- >expression(paste("Price"==beta[0]+beta[1],"Carat",+beta[2],"Carat"^-2))
##(vi)
>mtext("CARAT",side=1,line=0,outer=TRUE)
>mtext("SGD$",side=2,line=4)
>mtext("Scatterplot of Diamond Price by Carat and Certification",side=3,line=2,cex=1.5)
>mtext(expr1,side=4,line=4)

```

Scatterplot of Diamond Price by Carat and Certification

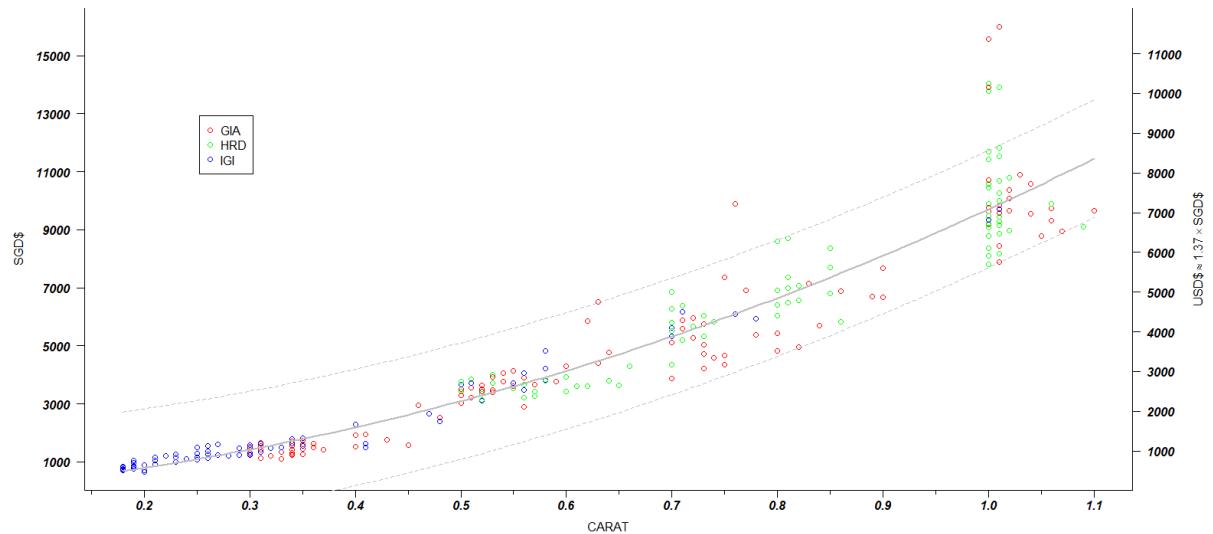


```

##(vii)
>interactive.arrow(label=expr2)
##(viii)
>legend(locator(1),legend=levels(diamonds$Cert),col=c("red","green","blue"),pch=1)

```

Scatterplot of Diamond Price by Carat and Certification



# Chapter 24: Going Further with the Grammar of Graphics

Estimated time to complete: **60 minutes**

## Exercise 24.1

Load the *MASS* package and inspect the help file for the *UScereal* data. This data frame provides nutritional and other information concerning breakfast cereals for sale in the United States in the early 1990s.

- a. Create a copy of the data frame; name it *cereal*. To ease plotting, collapse the *mfr* column (manufacturer) of *cereal* to be a factor with only three levels, with the corresponding labels "*General Mills*", "*Kellogg's*", and "*Other*". Also, convert the *shelf* variable (shelf number from floor) to a factor.
- b. Using *cereal*, construct and store two *ggplot* objects.
  - i. A scatterplot of calories on protein. Points should be colored according to shelf position and shaped according to manufacturer. Include simple linear regression lines for calories on protein, split according to shelf position. Ensure tidy axis and legend titles.
  - ii. A set of kernel estimates of calories, with filled color differentiating shelf positions. Use 50 percent opaque fills, and again ensure tidy axis and legend titles.
- c. Arrange the two plots in (b) on a single device.
- d. Produce a faceted graphic of calories on protein, with each panel corresponding to a manufacturer as defined in your *cereal* object. A LOESS smoother with a 90 percent span should be superimposed upon each scatterplot. In addition, the points should be colored according to sugar content, sized according to sodium content, and shaped according to shelf position.

Load the *car* package (downloading and installing it first if you haven't already) and consider the *Salaries* object—a data frame detailing the salaries (in US dollars) of 397 academics working in the United States during the 2008–2009 school year (Fox and Weisberg, 2011). An inspection of the help file *?Salaries* informs you of the present variables, which, in addition to the salary figure, include each academic's rank, sex, and research discipline (as factors) as well as the number of years of service.

- e. Produce a *ggplot* object, named *gg1*, of a scatterplot of salary on the vertical axis against years of service on the horizontal axis.

Color should be used to distinguish between males and females, along with sex-specific LOESS trends, and ensure axis and legend titles are understandable. View your plot.

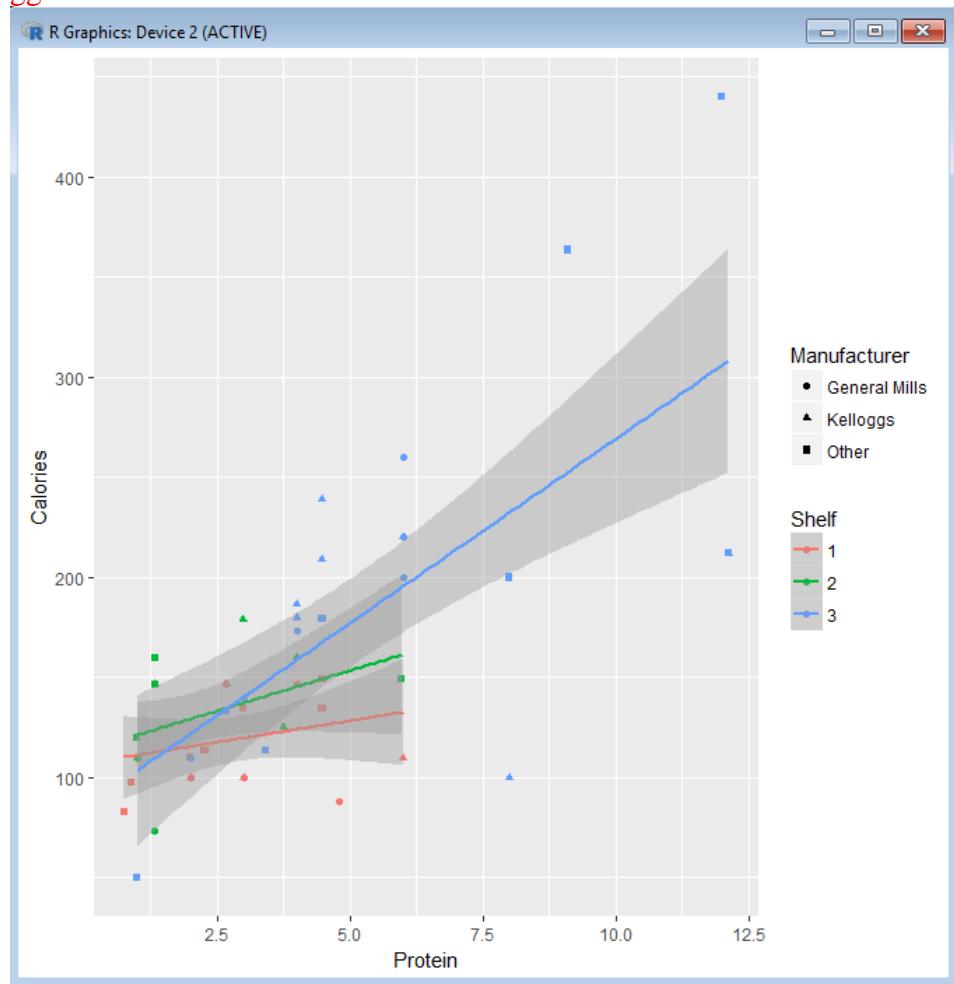
- f. Create the following three additional plot objects, again ensuring tidy axis and legend titles. Name the following *gg2*, *gg3*, and *gg4*, respectively:

- i. Side-by-side boxplots of salary, split by rank. Each boxplot should be further split up according to sex (this can be done simply in the default aesthetic mapping—try assigning the
  - ii. sex variable to either *col* or *fill*).
  - iii. Side-by-side boxplots of salary, split by discipline, with each discipline split further by sex using color or fill.
  - iv. Kernel density estimates of salary, using 30 percent opaque fills to distinguish rank.
- g. Display your four plot objects (*gg1*, *gg2*, *gg3*, and *gg4*) from (e) and (f) in a single device.
  - h. Finally, plot the following:
    - i. A series of kernel density estimates of salary using 70 percent opaque fills to distinguish between males and females, faceted by academic rank.

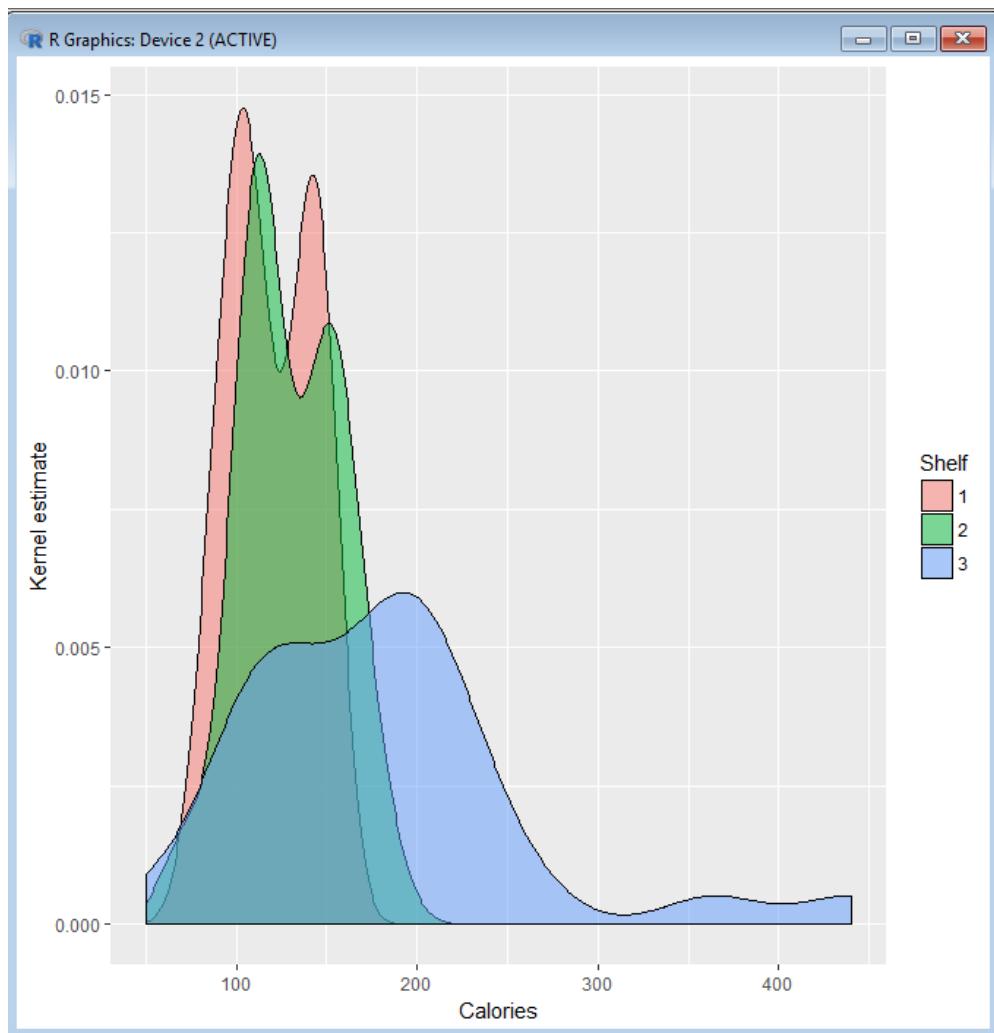
- ii. Scatterplots of salary on years of service, using color to distinguish between males and females, faceted by discipline as rows and by academic rank as columns. Each scatterplot
- iii. should have a sex-specific simple linear regression line with confidence band superimposed and have free horizontal scales.

### Solution 24.1

```
library("MASS")
?UScereal
library("ggplot2")
#(a)
cereal <- UScereal
new.mfr <- as.numeric(UScereal$mfr)
new.mfr[new.mfr>2] <- 3
cereal$mfr <- factor(new.mfr,labels=c("General Mills","Kellogg's","Other"))
cereal$shelf <- factor(cereal$shelf)
#(b)
##(i)
gg1 <- ggplot(cereal,aes(x=protein,y=calories,col=shelf)) + geom_point(aes(shape=mfr)) +
  geom_smooth(method="lm") +
  labs(x="Protein",y="Calories",col="Shelf",size="Carbs",shape="Manufacturer")
gg1
```

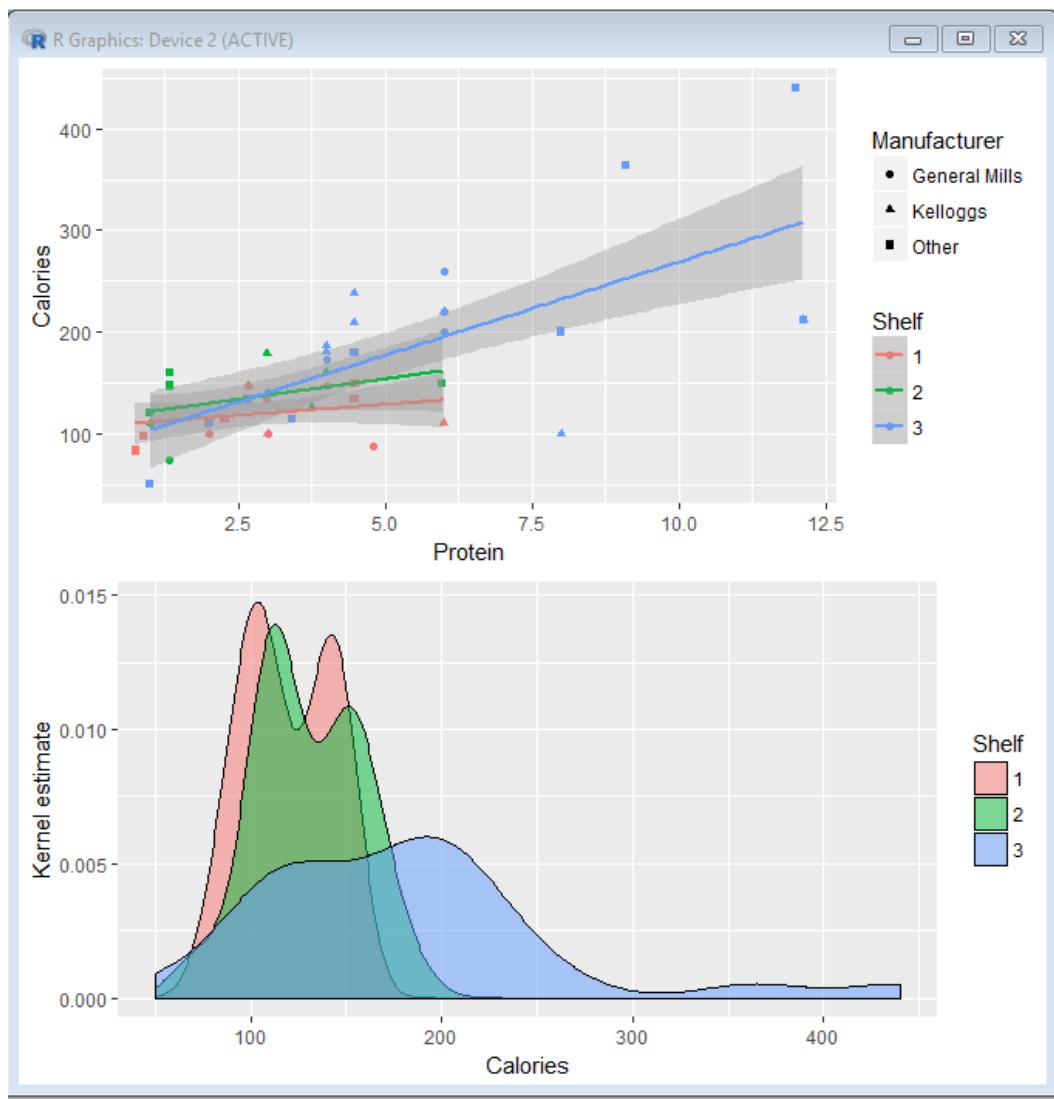


```
##(ii)
gg2 <- ggplot(cereal,aes(x=calories,fill=shelf)) + geom_density(alpha=0.5) +
  labs(x="Calories",y="Kernel estimate",fill="Shelf")
gg2
```



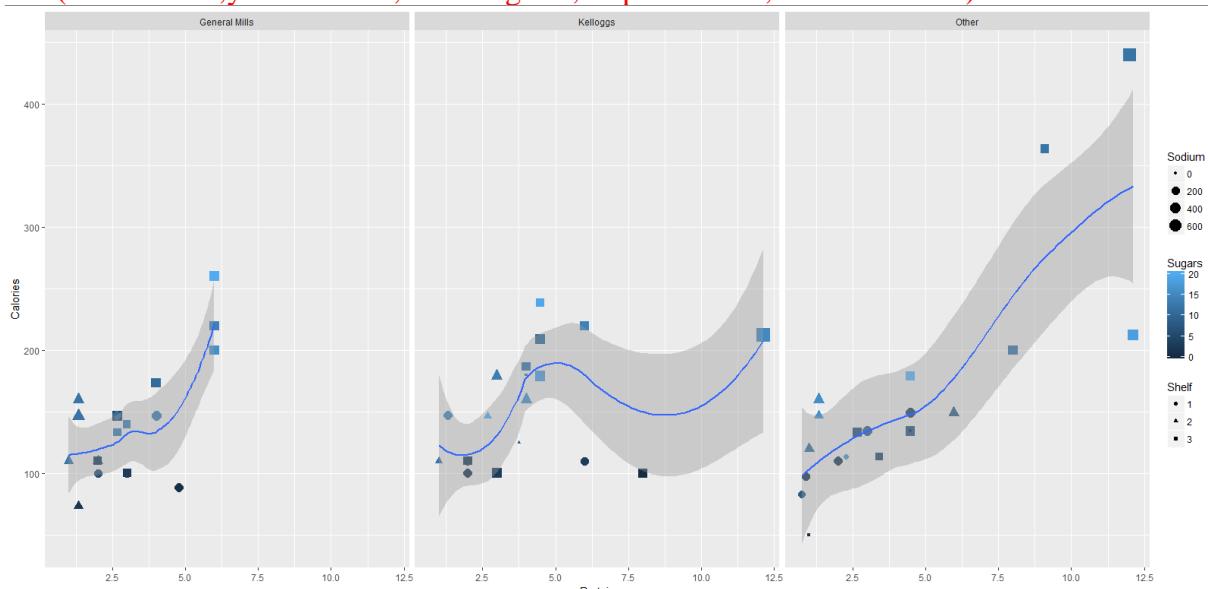
#(c)

```
library("gridExtra")
grid.arrange(gg1,gg2)
```

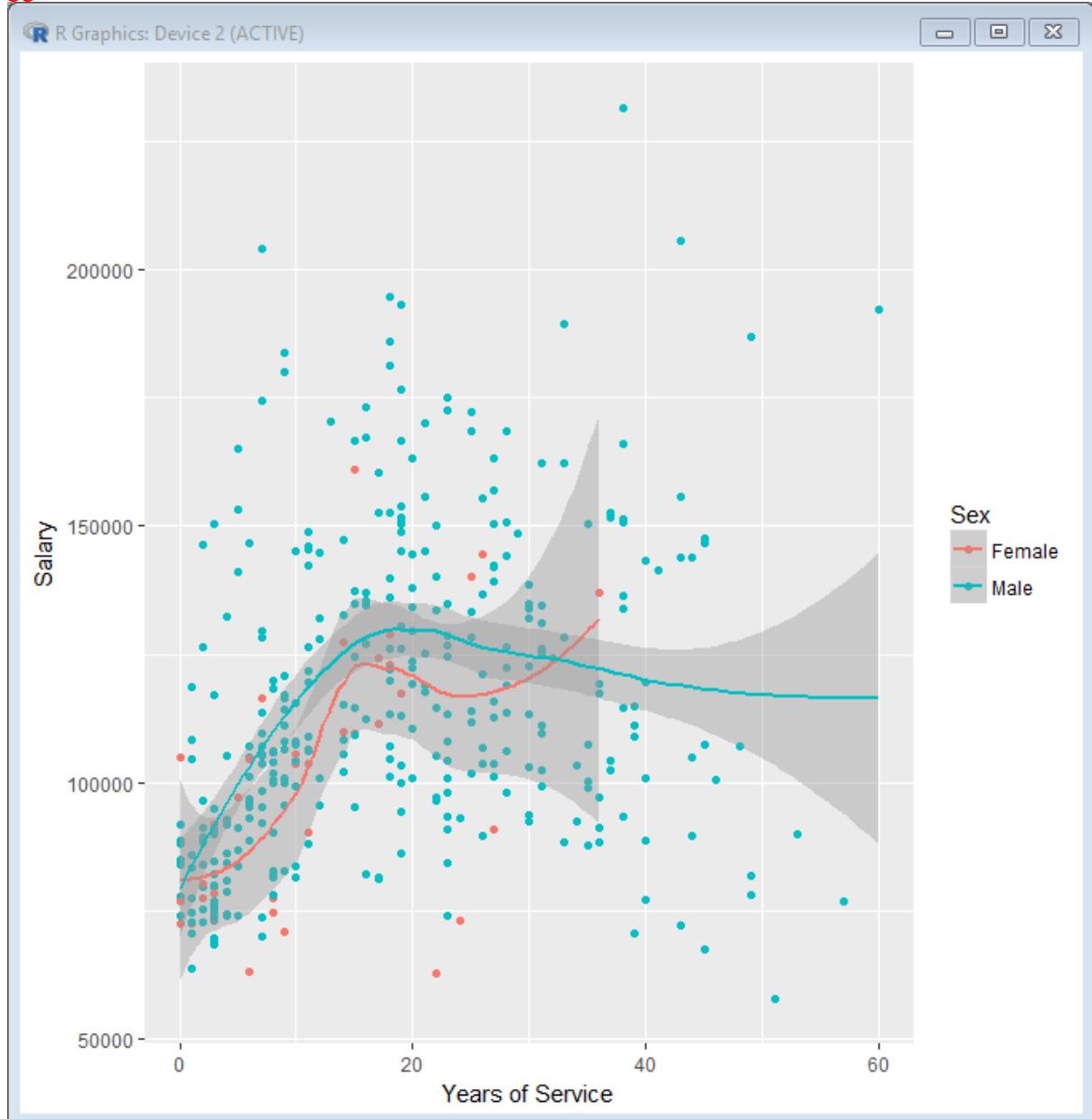


#(d)

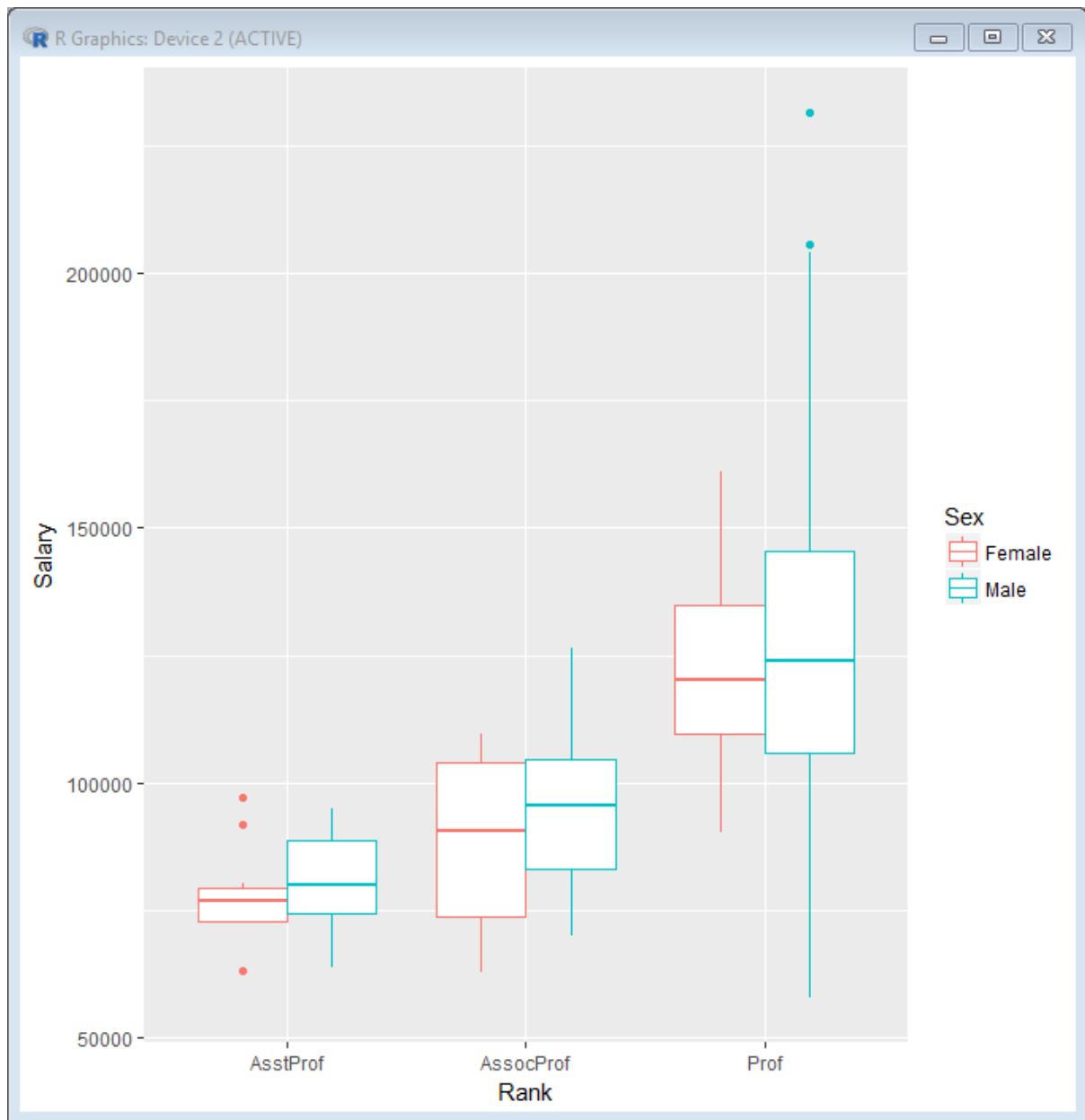
```
ggplot(cereal,aes(x=protein,y=calories)) + geom_point(aes(col=sugars,size=sodium,shape=shelf)) + geom_smooth(method="loess",span=0.9) + facet_wrap(~mfr) + labs(x="Protein",y="Calories",col="Sugars",shape="Shelf",size="Sodium")
```



```
#(e)
library("car")
gg1 <- ggplot(Salaries,aes(x=yrs.service,y=salary,col=sex)) + geom_point() +
geom_smooth(method="loess") + labs(x="Years of Service",y="Salary",col="Sex")
gg1
```



```
#(f)
##(i)
gg2 <- ggplot(Salaries,aes(x=rank,y=salary,col=sex)) + geom_boxplot() +
labs(x="Rank",y="Salary",col="Sex")
gg2
```



##(ii)

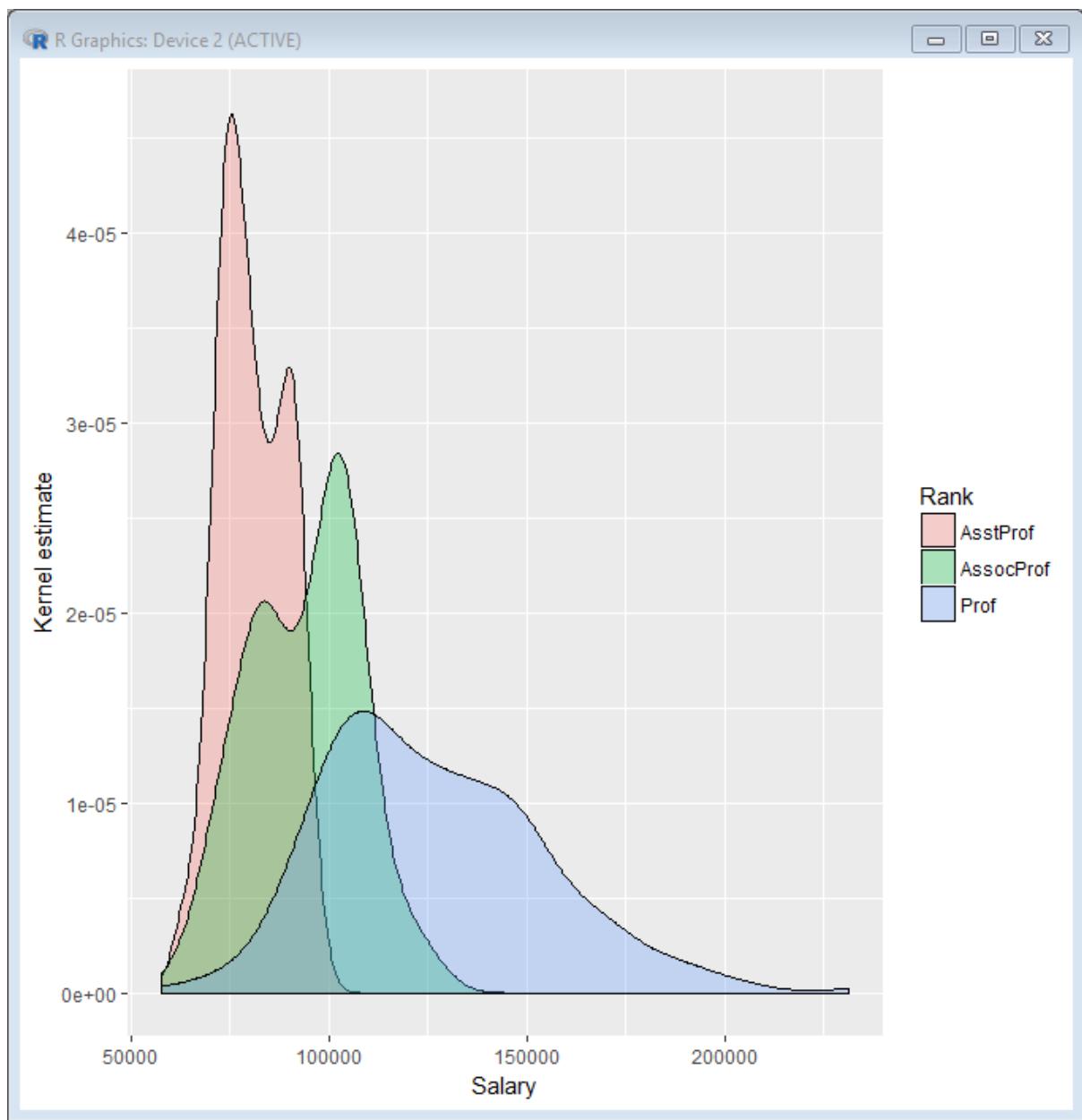
```
gg3 <- ggplot(Salaries,aes(x=discipline,y=salary,fill=sex)) + geom_boxplot() +
  labs(x="Discipline",y="Salary",fill="Sex")
```

gg3

##(iii)

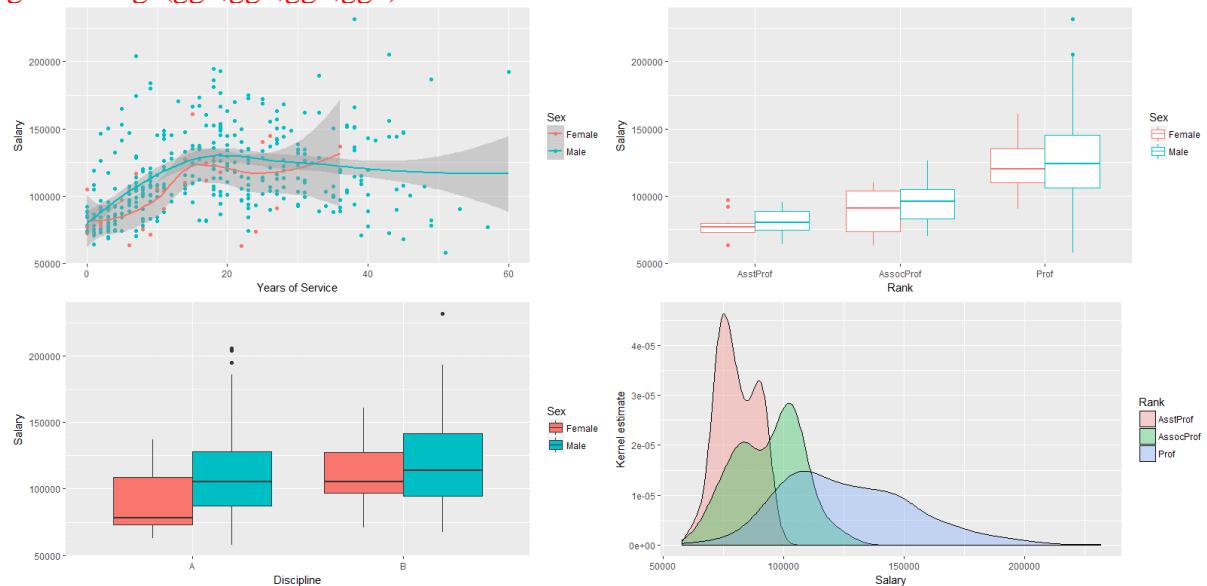
```
gg4 <- ggplot(Salaries,aes(x=salary,fill=rank)) + geom_density(alpha=0.3) +
  labs(x="Salary",y="Kernel estimate",fill="Rank")
```

gg4

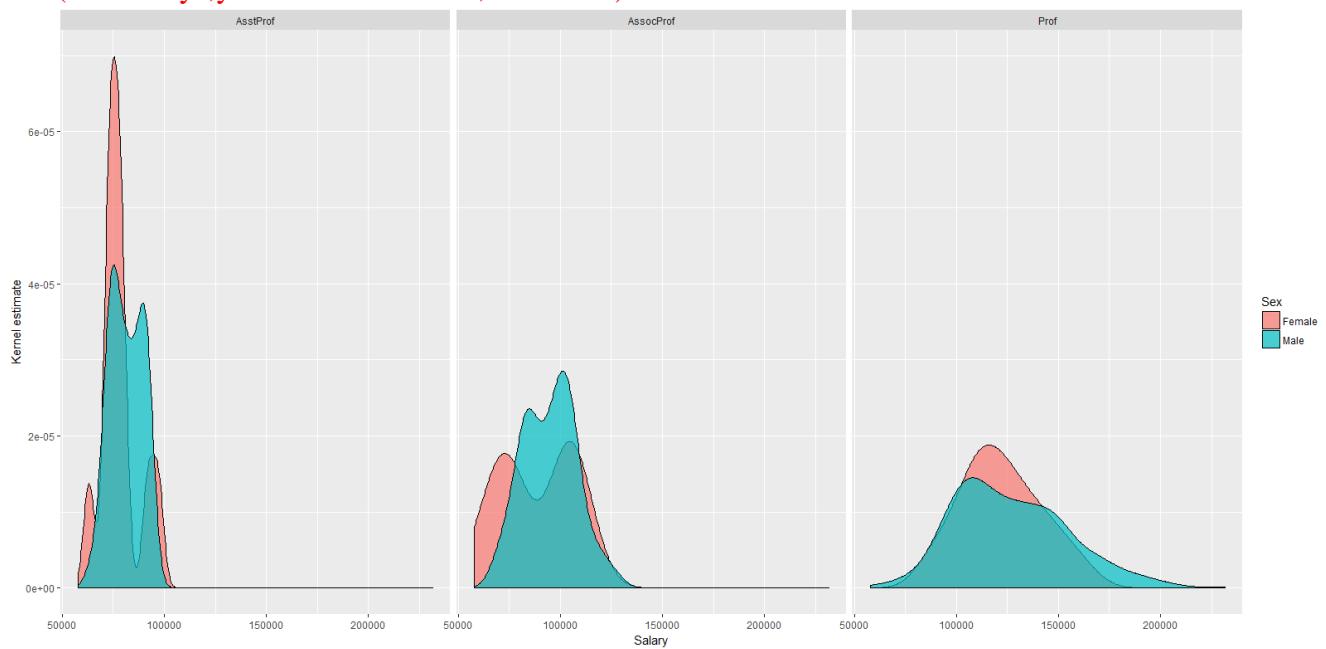


#(g)

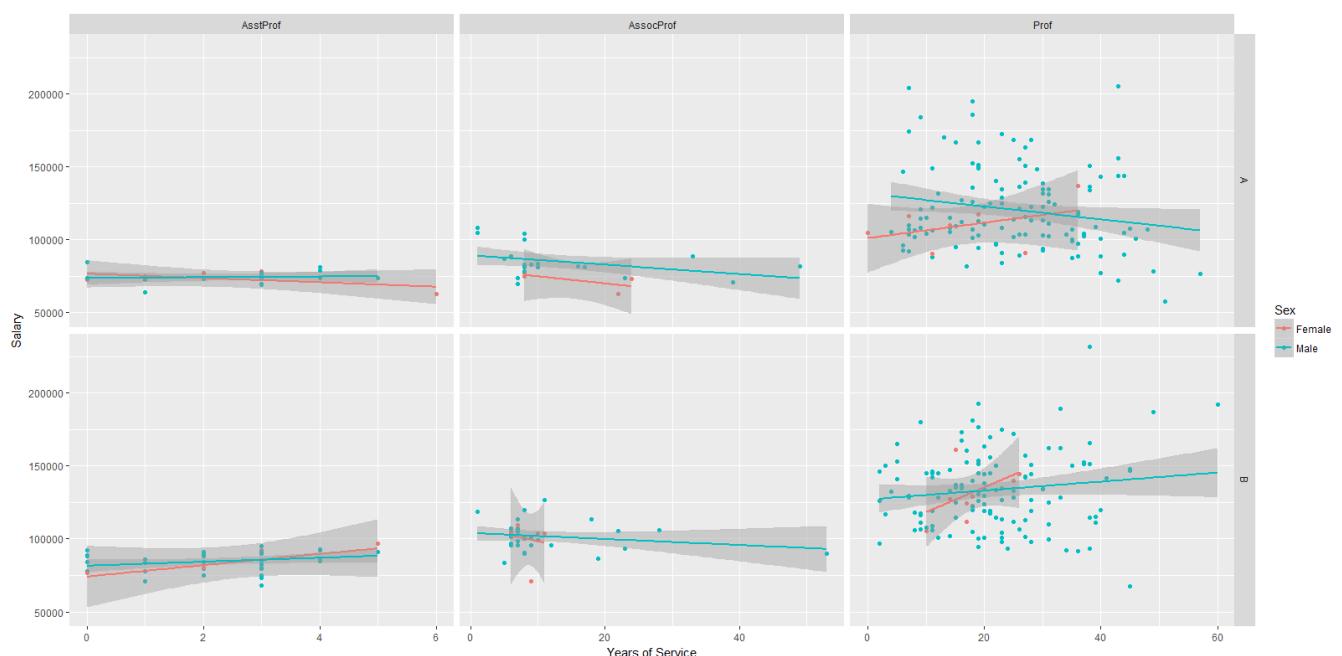
grid.arrange(gg1,gg2,gg3,gg4)



```
#(h)
##(i)
ggplot(Salaries,aes(x=salary,fill=sex)) + geom_density(alpha=0.7) + facet_wrap(~rank) +
labs(x="Salary",y="Kernel estimate",fill="Sex")
```



```
##(ii)
ggplot(Salaries,aes(x=yrs.service,y=salary,col=sex)) + geom_point() +
geom_smooth(method="lm") + facet_grid(discipline~rank,scales="free_x") + labs(x="Years of Service",y="Salary",col="Sex")
```



## Exercise 24.2

Ensure the `car` and `ggvis` packages are loaded. Revisit the `Salaries` data frame you looked at in Exercise 24.1; inspect the help file `?Salaries` to remind yourself of the present variables.

- a. Produce an interactive scatterplot of salary on the vertical axis and the years of service on the horizontal axis. Employ radio buttons to color points according to either academic rank, research discipline, or sex. Use pipes to `add_legend` and `add_axis` to omit a legend title and to tidy up the axis titles, respectively.
- b. A pipe to `layer_densities` (which you've not yet met) is used to produce kernel density estimates, similar to those appearing in Figure 24-5.
  - i. Use `ggvis` to create a static plot of kernel density estimates of salary distributions, split up according to academic rank.  
To do this, assign the salary variable to `x` and the rank variable to `fill`, followed by a pipe to `group_by` to explicitly instruct grouping by the rank variable. Lastly, piping to `layer_densities` (just use all default argument values in this instance) will generate the graphic. Your result should resemble the `gg4` object from Exercise 24.1.
  - ii. Just like the `width` argument to `layer_histograms` is used to control the appearance of a histogram, the `adjust` argument in `layer_densities` is used to control the degree of smoothness of the kernel estimates. Reproduce the rank-specific kernel estimates from the previous plot, but this time, the graphic should be interactive—implement a slider button with a range of 0.2 to 2 and a label of "Smoothness" to control the smoothing adjustment. At your discretion, either suppress or clarify the axis and legend titles of the result.

Ensure you have the `MASS` package loaded, once more gaining access to the `UScereal` data frame. If you haven't already done so, inspect the help file `?UScereal` and re-create the `cereal` object exactly as specified in Exercise 24.1 (a). Then do the following:

- c. Set up an object for radio buttons to choose among the manufacturer, the shelf, and the vitamins variables. Make sure the labels for each radio button are clear, and set up an appropriate title label for what will form the collection of options to color the points. Name the object `filler`.
- d. Borrowing the `sizer` and `opacityyer` objects created in Section 24.4 and using the object you just created in (c) to control `fill`, create an interactive scatterplot of calories on protein. Tidy up the axis titles and suppress the legend title for the point color fill. The result should essentially be the same, in terms of functionality, as the graphic appearing as the topmost screenshot in Figure 24-8.
- e. Create a new object for the same radio buttons as specified in (c) that will control the shape of the points (in other words, the characters used to plot points). Modify the title label accordingly.

Name this object `shaper`.

- f. Finally, re-create the interactive scatterplot of calories on protein exactly as in (d), but this time additionally assigning `shaper` from (e) to the `shape` modifier in your call to `ggvis`. To prevent the legends for the two sets of radio buttons from overlapping each other, you need to add the following pipes to your code:

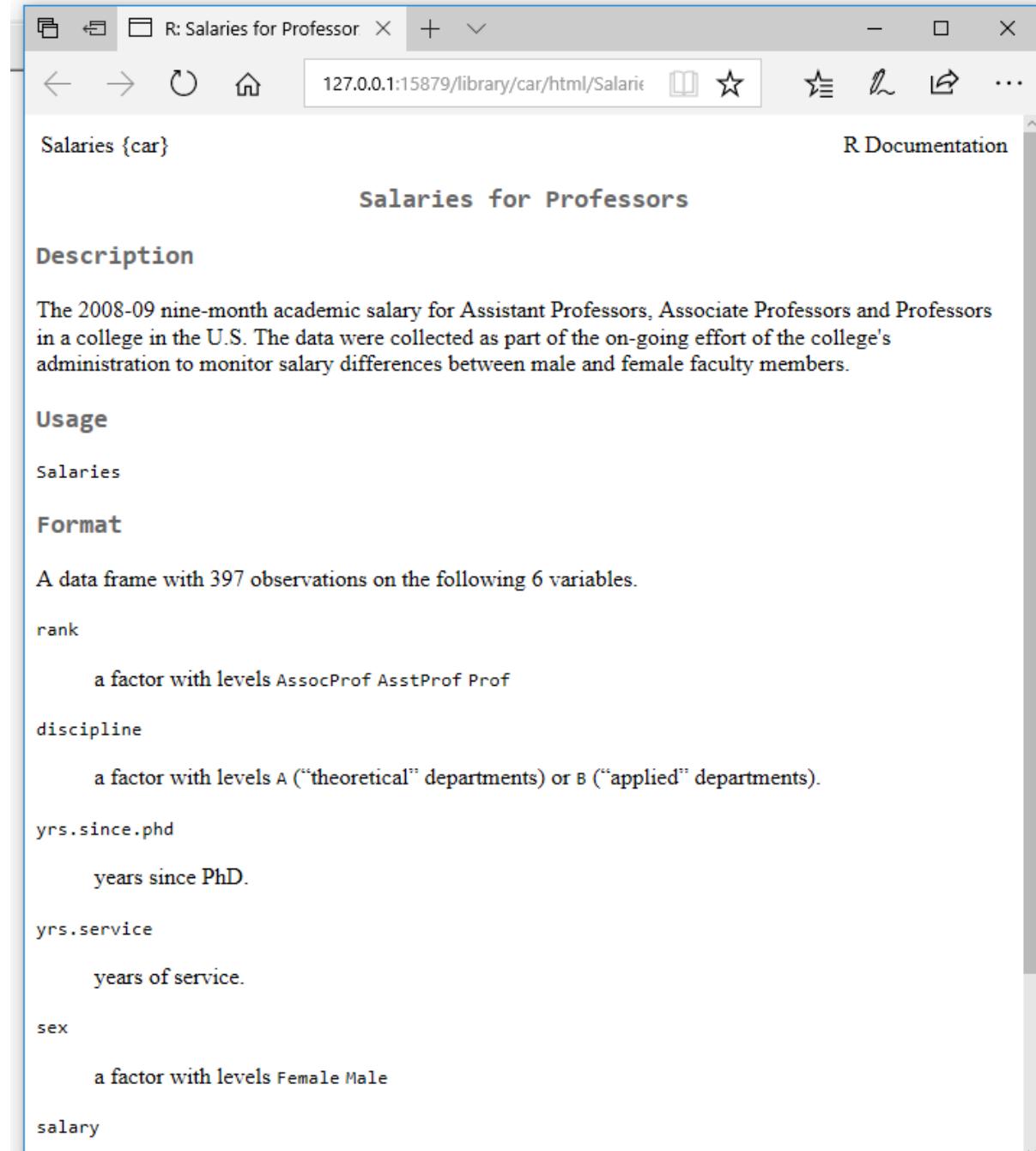
```
add_legend("shape",title="",
properties=legend_props(legend=list(y=100)))
and
set_options(duration=0)
```

The first simply moves the legend for the *shape* modifier vertically downward, and the second eliminates the slight “animation delay” that occurs by default when switching between options in the interactive graphic. Once more, use additional calls to *add\_axis* and *add\_legend* to clarify or suppress axis and legend titles.

## Solution 24.2

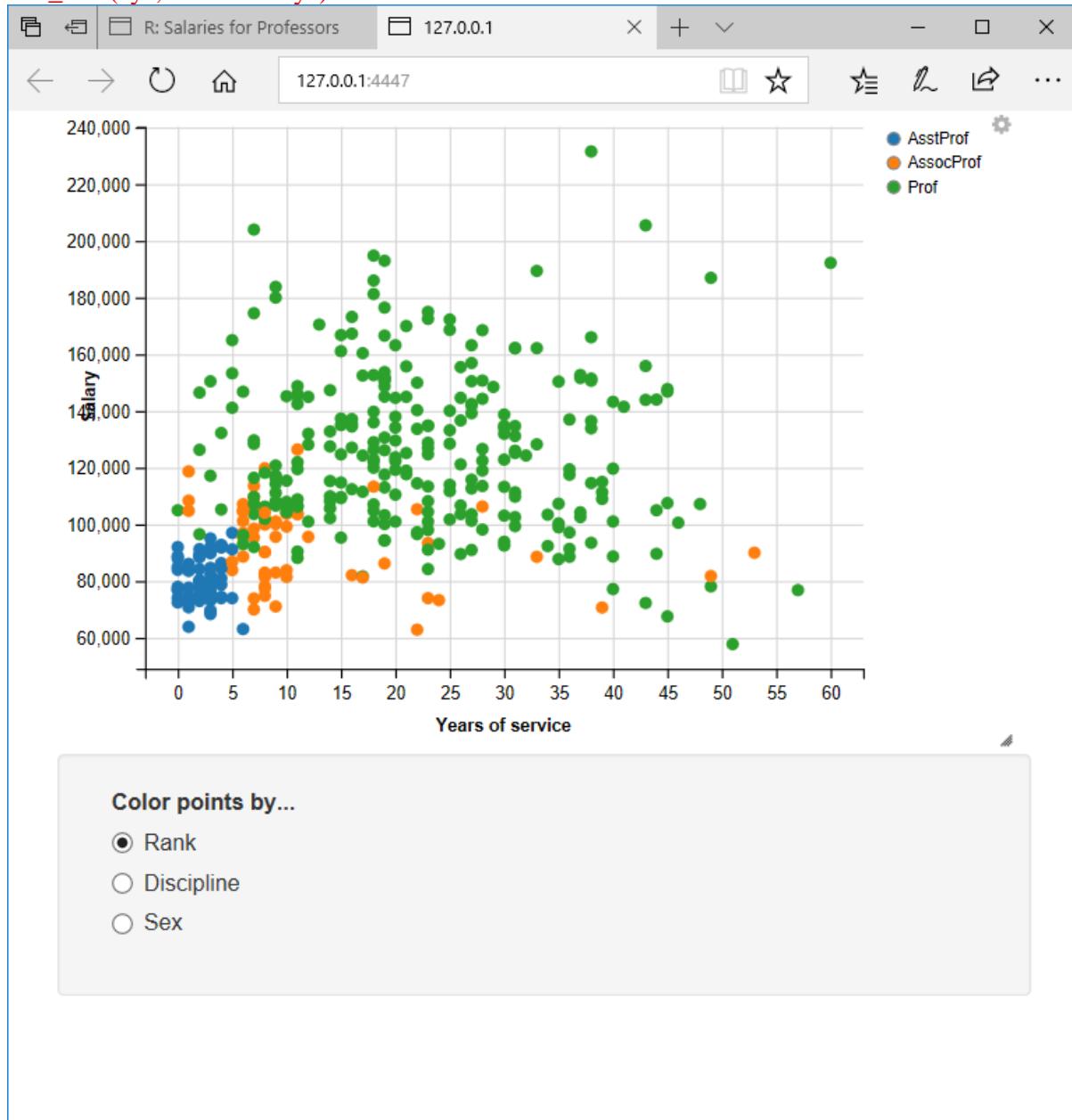
```
library("ggvis")
library("car")
?Salaries
```

starting httpd help server ... done



```
#(a)
>salfill <-
input_radiobuttons(c("Rank"="rank","Discipline"="discipline","Sex"="sex"),map=as.name,label=
"Color points by...")
```

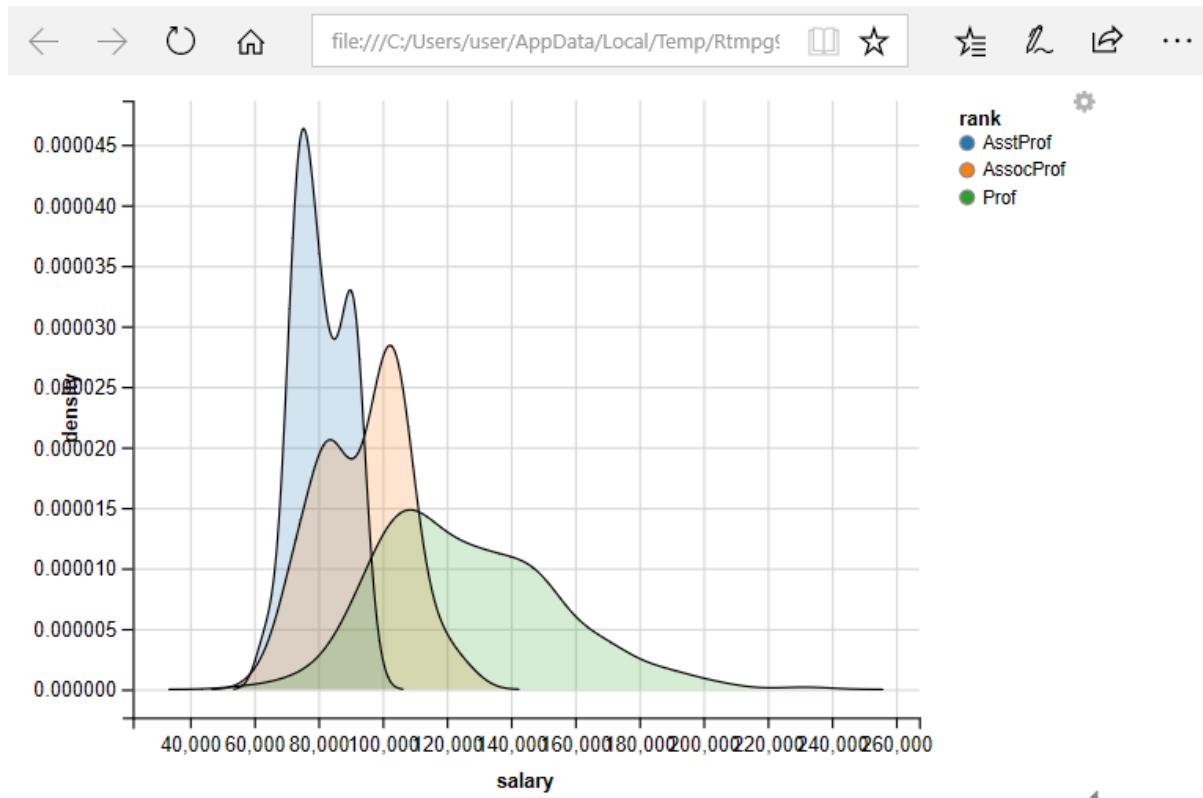
```
>Salaries %>% ggvis(x=~yrs.service,y=~salary,fill=salfill) %>% layer_points() %>%  
add_legend("fill",title "") %>% add_axis("x",title="Years of service") %>%  
add_axis("y",title="Salary")
```



#(b)

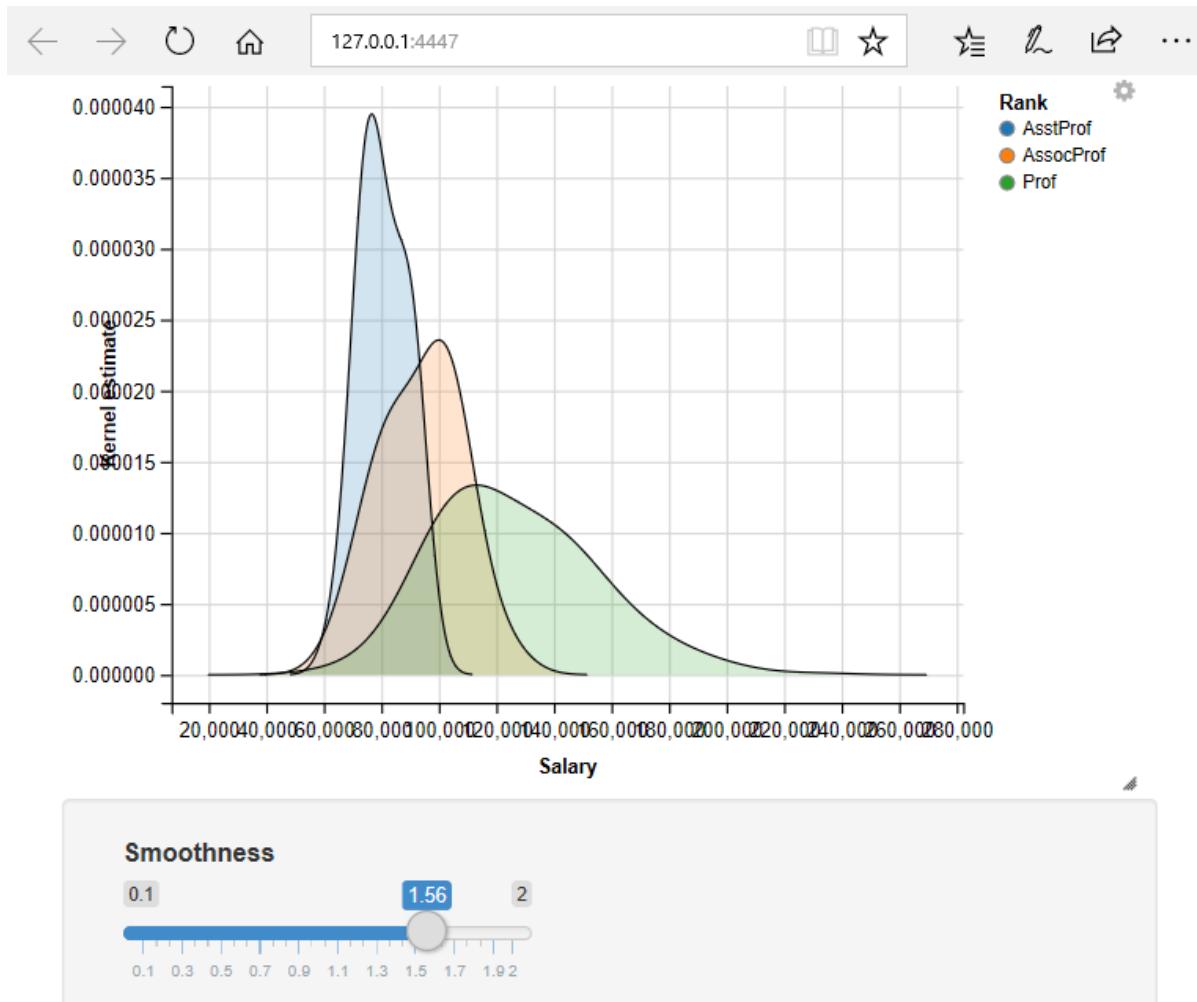
##(i)

```
Salaries %>% ggvis(x=~salary,fill=~rank) %>% group_by(rank) %>% layer_densities()
```



##(ii)

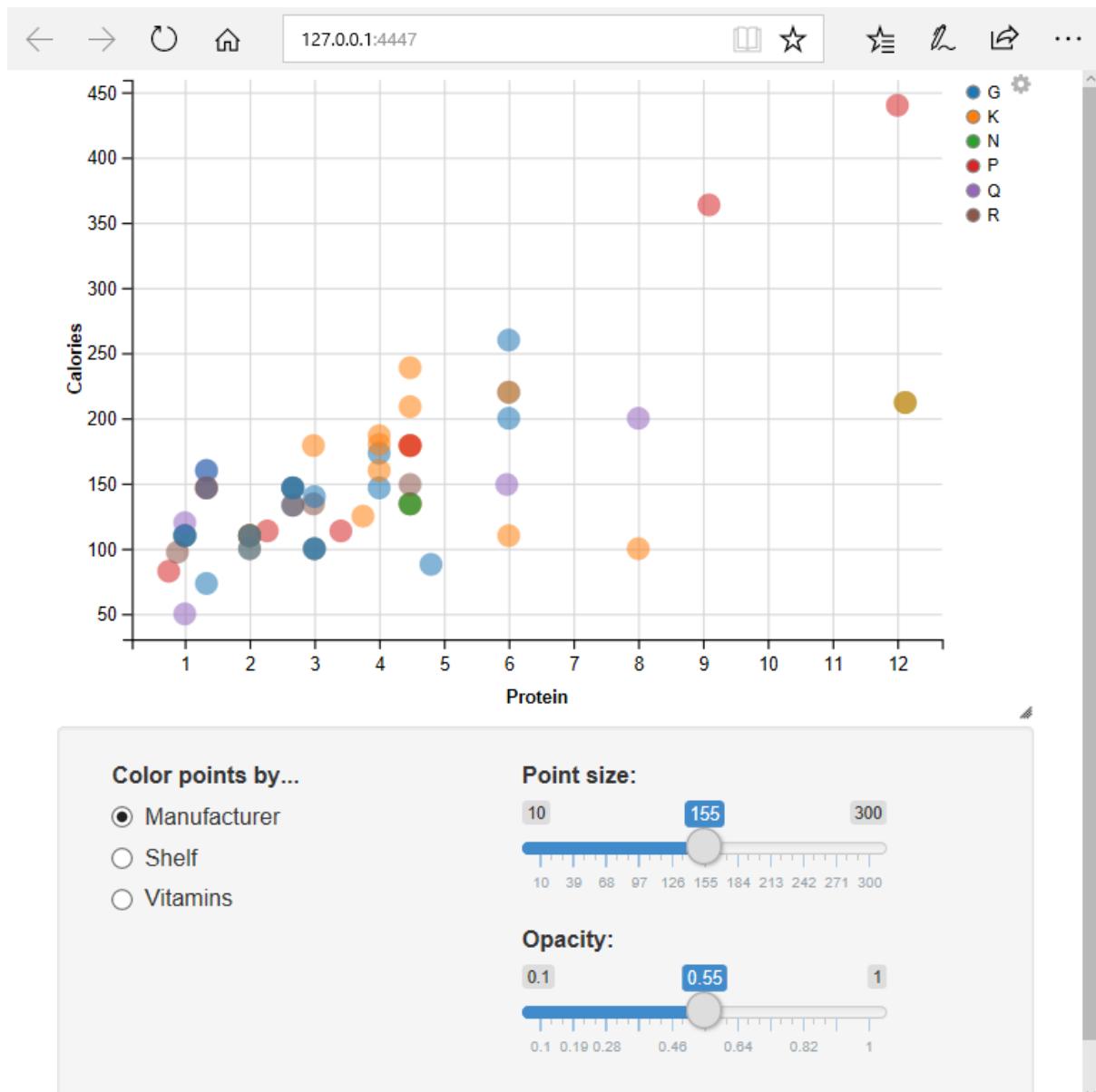
```
Salaries %>% ggyvis(x=~salary,fill=~rank) %>% group_by(rank) %>%
layer_densities(adjust=input_slider(0.1,2,label="Smoothness")) %>% add_axis("x",title="Salary")
%>% add_axis("y",title="Kernel estimate") %>% add_legend("fill",title="Rank")
```

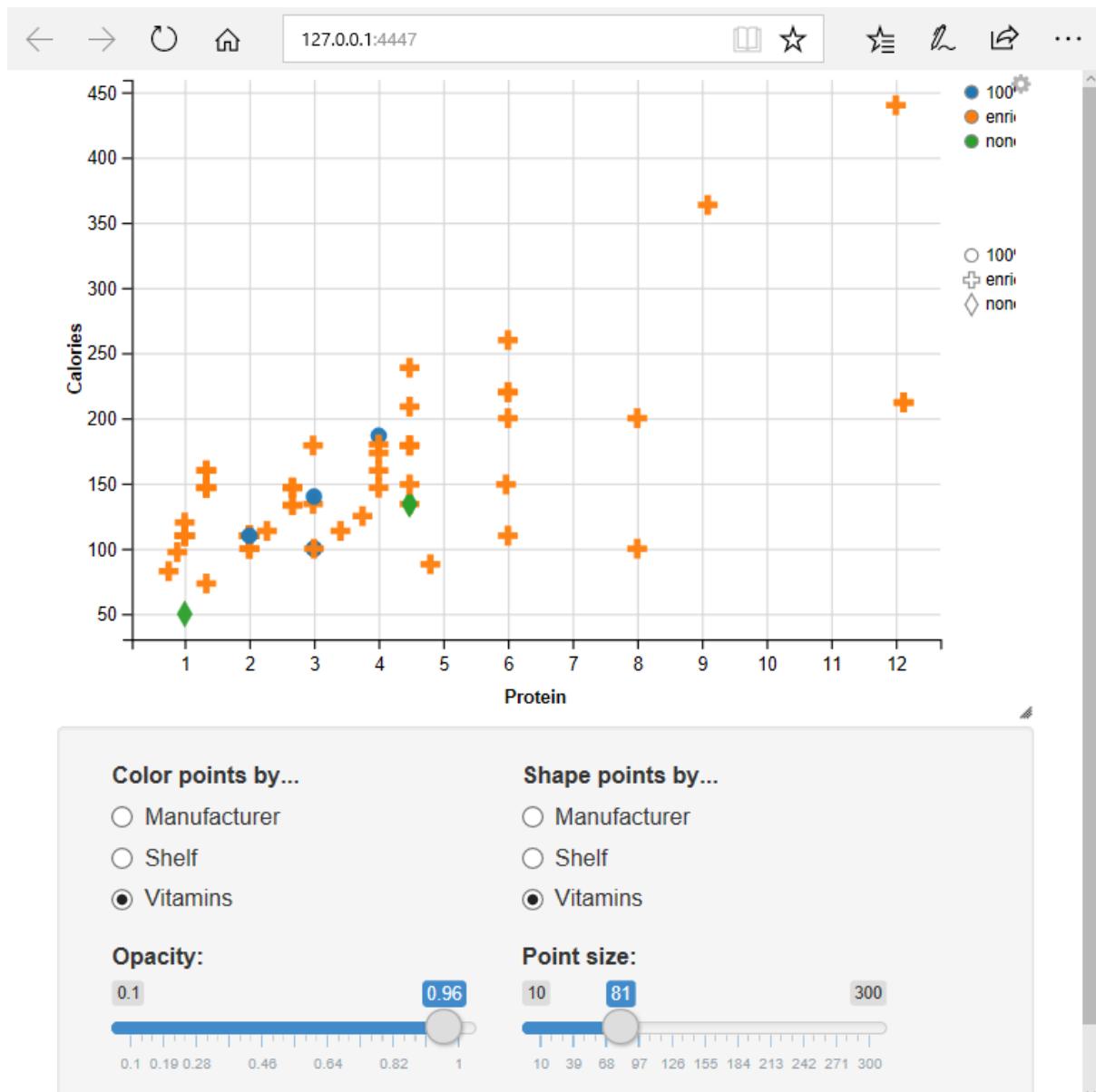


```

#(c)
library("MASS")
cereal <- UScereal
new.mfr <- as.numeric(UScereal$mfr)
new.mfr[new.mfr>2] <- 3
cereal$mfr <- factor(new.mfr,labels=c("General Mills","Kelloggs","Other"))
cereal$shelf <- factor(cereal$shelf)
filler <-
input_radiobuttons(c("Manufacturer"="mfr","Shelf"="shelf","Vitamins"="vitamins"),map=as.name,
e,label="Color points by...")
sizer <- input_slider(10,300,label="Point size:")
opacityyer <- input_slider(0.1,1,label="Opacity:")
#(d)
cereal %>% ggvis(x=~protein,y=~calories,fill=filler,size:=sizer,opacity:=opacityyer) %>%
layer_points() %>% add_axis("x",title="Protein") %>% add_axis("y",title="Calories") %>%
add_legend("fill",title="")
#(e)
shaper <-
input_radiobuttons(c("Manufacturer"="mfr","Shelf"="shelf","Vitamins"="vitamins"),map=as.name,
e,label="Shape points by...")

```





# Chapter 25:Defining Colors and Plotting in Higher Dimensions

Estimated time to complete: **90 minutes**

## Exercise 25.1

Ensure the *car* package is loaded. Revisit the *Salaries* data frame you looked at in Exercises 24.1 (page 622) and 24.2 (page 628) and take a look at the help file *?Salaries* to remind yourself of the variables.

Your task is to use color, point size, opacity, and point character type to reflect “years since Ph.D.,” “sex,” and “rank” in a scatterplot of “salary” against “years of service,” by completing the following steps:

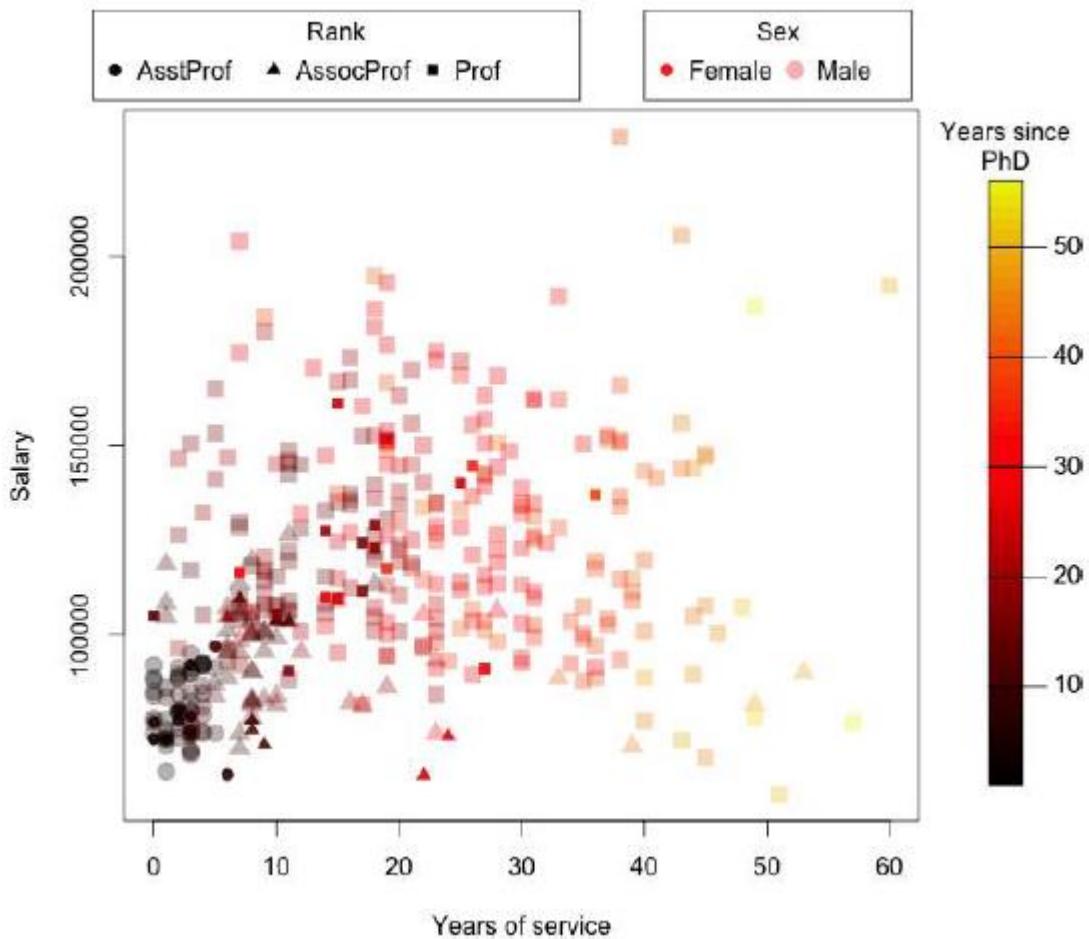
- a. Set up a custom color palette that goes from “*black*” to “*red*” to “*yellow2*”. Create two versions of this palette—one that expects a number of colors and one that expects a vector of normalized values between 0 and 1.
- b. Create two vectors that will control point character and character expansion following the guidelines in (i) and (ii). Each of these can be achieved in a single line by vector subsetting/repetition based on a numeric coercion of the corresponding factor vector in the data frame.
  - i. Use the point characters *19*, *17*, and *15* to reference the three increasing academic ranks in that order.
  - ii. Use a character expansion of *1* for females and a character expansion of *1.5* for males.
- c. Use the *normalize* function defined in Section 25.1.4 to create a [0,1] normalized version of the range of values of the “years since Ph.D.” variable. Then use the appropriate palette from (a) along with *rgb* to convert these to the required hex codes.
- d. Modify the vector of colors you just created in (c), adjusting opacity. Colors in the vector that correspond to females should be reduced to 90 percent opacity; colors that correspond to males should be reduced to 30 percent opacity.
- e. Now, start the plot; alter the default figure margins to be 5, 4, 4, and 6 lines wide on the bottom, left, top, and right, respectively.

Plot salary on the *y*-axis against years of service on the *x*-axis. Set the corresponding point colors according to your vector from (d) and the point characters and character expansion according to your vectors from (b). Tidy up the *x*-axis and *y*-axis titles.

- f. Incorporate two separate legends following the guidelines in (i) and (ii). Both legends should be horizontal, and you should relax clipping to allow their placement in figure margins (refer to Section 23.2.3).
  - i. Place the first legend at the user coordinate given by *x*=-5 and *y*=265000. It should use the levels of the “rank” factor vector as the referencing text and pair these with the corresponding *pch* symbols as assigned. Include an appropriate title for the legend.
  - ii. The second legend should be placed next to the first, using an *x*-coordinate of 40 and the same *y*-coordinate value. This legend should show two points, both red and of type *19*, but reference the two levels of sex by altering the character expansion and opacity of these to reference points as assigned.
- g. Lastly, ensure the *shape* package is loaded and use the *colorlegend* function along with 50 colors generated from the appropriate palette from (a) to reference “years since Ph.D.” You can leave the horizontal and vertical placements of the legend at their default values. The *zlim* range should simply be set to match the range of the observed data, and the tick mark values set via *zval* should be a sequence between 10 and 50, increasing in steps of

10. Include an appropriate title for the color legend.

After all this, my version of this plot is given here:



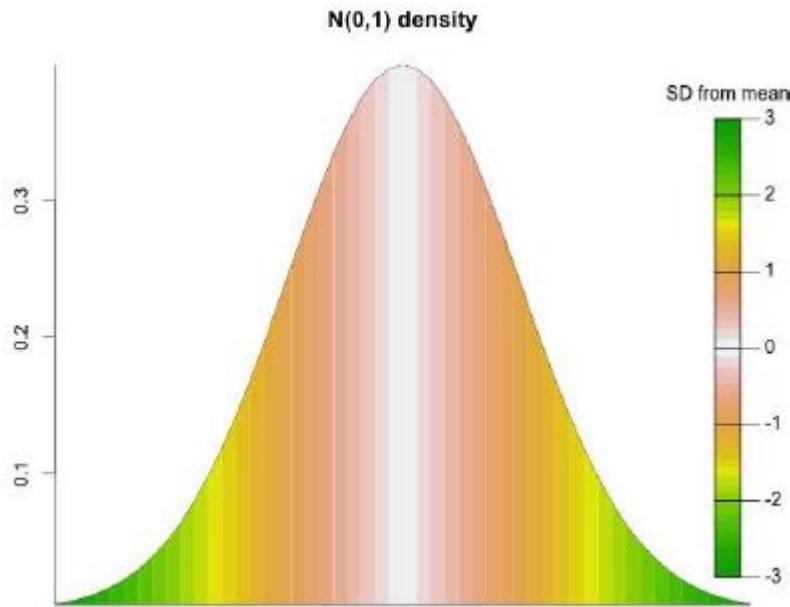
Your next task is a little different. The goal is to plot a standard normal probability density function but use color to shade in polygons underneath the curve to denote “distance from mean.” To achieve this, complete the following:

- h. Generate a vector of exactly 25 colors from the built-in palette `terrain.colors` and name it `tcols`. Then, using a reversed version of it obtained via `tcols[25:1]`, append the two vectors together to form a new vector of length 50 containing the first 25 colors shading one way and then the same 25 shading the opposite way.
- i. Next, create and store an evenly spaced sequence of exactly 51 values between  $-3$  and  $3$  inclusive; name it `vals`. Use `dnorm` to calculate and store the corresponding 51 values of the standard normal density curve; name it `normvals`.
- j. Draw the normal density curve by plotting the values in (i) as a line (recall `type="l"`). In the same call to `plot`, use knowledge from Chapter 23 to set both the `x`-axis and `y`-axis styles to be of type "`i`"; suppress both axis titles with empty strings; change the surrounding box to be an `L` shape; and suppress the drawing of the `x`-axis. Give the plot a suitable main title.
- k. To shade the different colors underneath the curve, use a `for` loop, iterating through the integers 1 to 50. At each iteration, the loop should call `polygon` (refer to Section 15.2.3).

Assuming your indexer is  $i$ , the vertices of each polygon should be formed by the vectors  $\text{vals}[\text{rep}(c(i,i+1),\text{each}=2)]$  and  $c(0,\text{normvals}[c(i,i+1)],0)$ . Each polygon should suppress its border and be colored according to the relevant  $i$ th entry in your color vector of length 50 created in (h).

1. Lastly, ensure the *shape* package has been loaded and use your length 50 color vector to produce a color legend with default placement to reference “distance from mean.” You can easily set the *zlim* and *zval* arguments in the call to *colorlegend* using *vals*.

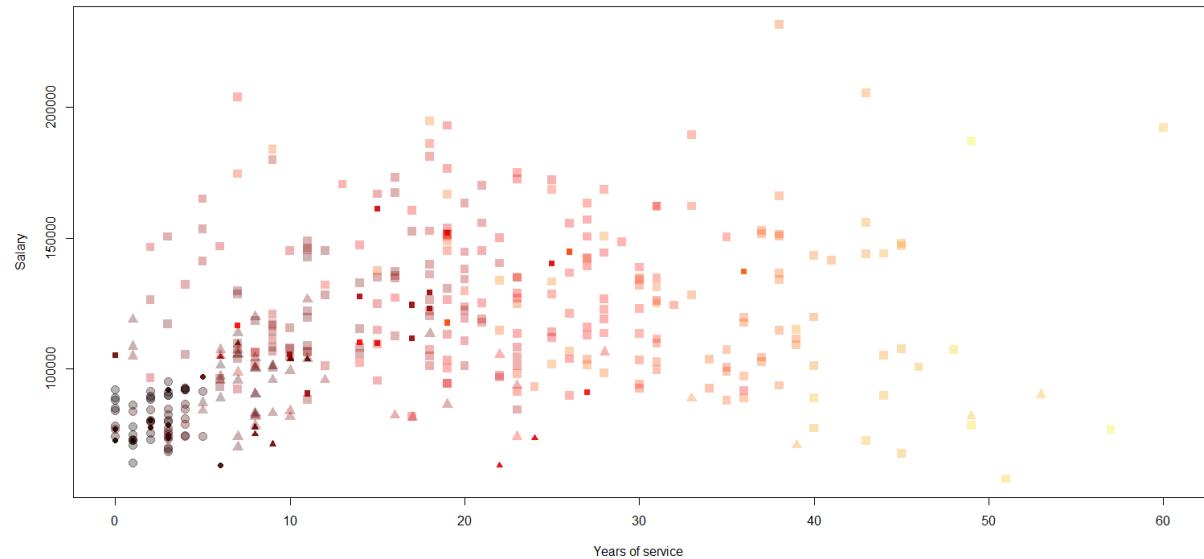
Include an appropriate title for the legend. For reference, my result is given here:



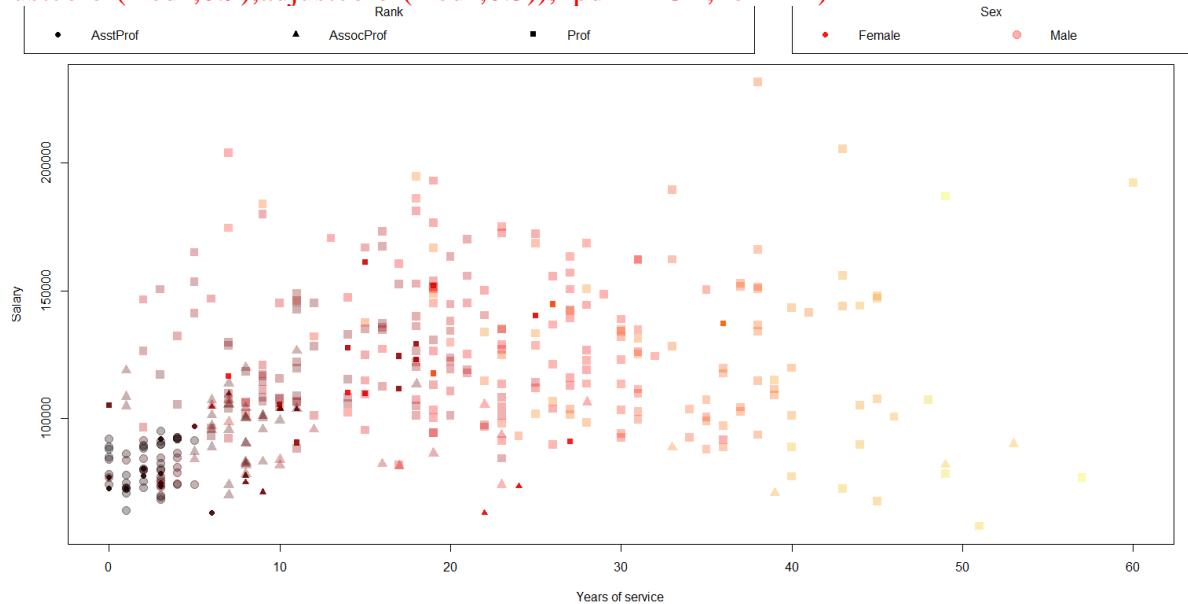
## Solution 25.1

```
library("car")
#(a)
cols1 <- colorRampPalette(c("black","red","yellow2"))
cols2 <- colorRamp(c("black","red","yellow2"))
#(b)
##(i)
rank.pchs <- c(19,17,15)[as.numeric(Salaries$rank)]
##(ii)
sex.cexs <- c(1,1.5)[as.numeric(Salaries$sex)]
#(c)
normalize <- function(datavec){
  lo <- min(datavec,na.rm=TRUE)
  up <- max(datavec,na.rm=TRUE)
  datanorm <- (datavec-lo)/(up-lo)
  return(datanorm)
}
phd.norm <- normalize(Salaries$yrs.since.phd)
phd.cols <- rgb(cols2(phd.norm),maxColorValue=255)
#(d)
phd.cols[Salaries$sex=="Female"] <- adjustcolor(phd.cols[Salaries$sex=="Female"],alpha=0.9)
phd.cols[Salaries$sex=="Male"] <- adjustcolor(phd.cols[Salaries$sex=="Male"],alpha=0.3)
```

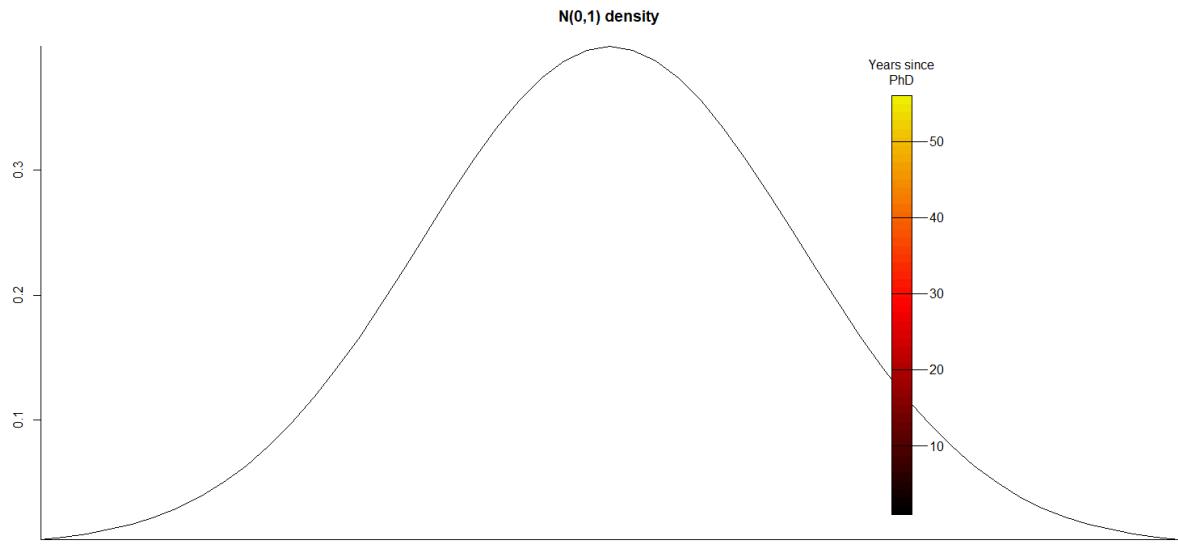
```
#(e)
par(mar=c(5,4,4,6))
plot(Salaries$salary~Salaries$yrs.service,col=phd.cols,pch=rank.pchs,cex=sex.cexs,xlab="Years
of service",ylab="Salary")
```



```
#(f)
legend(-
5,265000,legend=levels(Salaries$rank),title="Rank",pch=c(19,17,15),xpd=TRUE,horiz=TRUE)
legend(40,265000,legend=levels(Salaries$sex),title="Sex",pch=c(19,19),pt.cex=c(1,1.5),col=c(adj
ustcolor("red",0.9),adjustcolor("red",0.3)),xpd=TRUE,horiz=T)
```



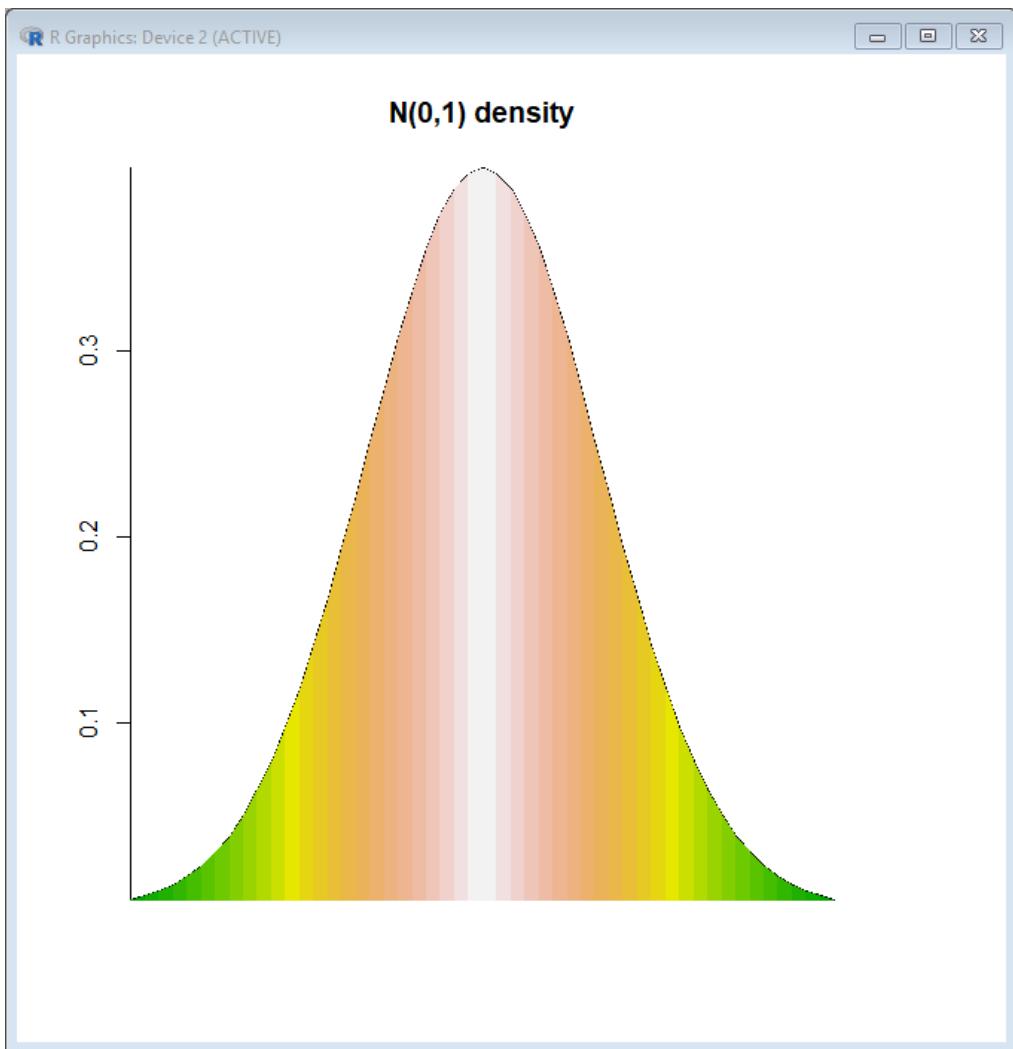
```
#(g)
library("shape")
colorlegend(col=cols1(50),zlim=range(Salaries$yrs.since.phd),zval=seq(10,50,10),main="Years
since\nPhD")
```



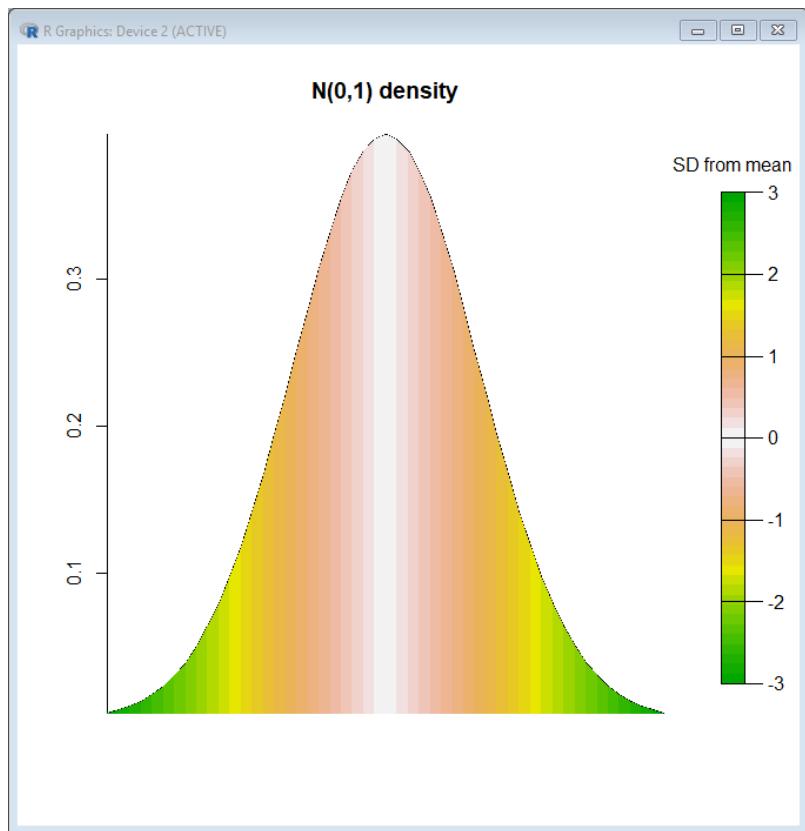
```

#(h)
tcols <- terrain.colors(25)
fillcols <- c(tcols,tcols[25:1])
#(i)
nvals <- 51
vals <- seq(-3,3,length=nvals)
normvals <- dnorm(vals)
#(j)
plot(vals,normvals,type="l",xaxs="i",yaxs="i",xlab="",ylab="",bty="L",xaxt="n",main="N(0,1)
density")
#(k)
for(i in 1:(nvals-1)){
  polygon(x=rep(vals[c(i,i+1)],each=2),y=c(0,normvals[c(i,i+1)],0),border=NA,col=fillcols[i])
}

```



```
#(1)
library("shape")
colorlegend(fillcols,zlim=range(vals),zval=-3:3,main="SD from mean")
```



## Exercise 25.2

Ensure the *scatterplot3d* library has been loaded in your current R session.

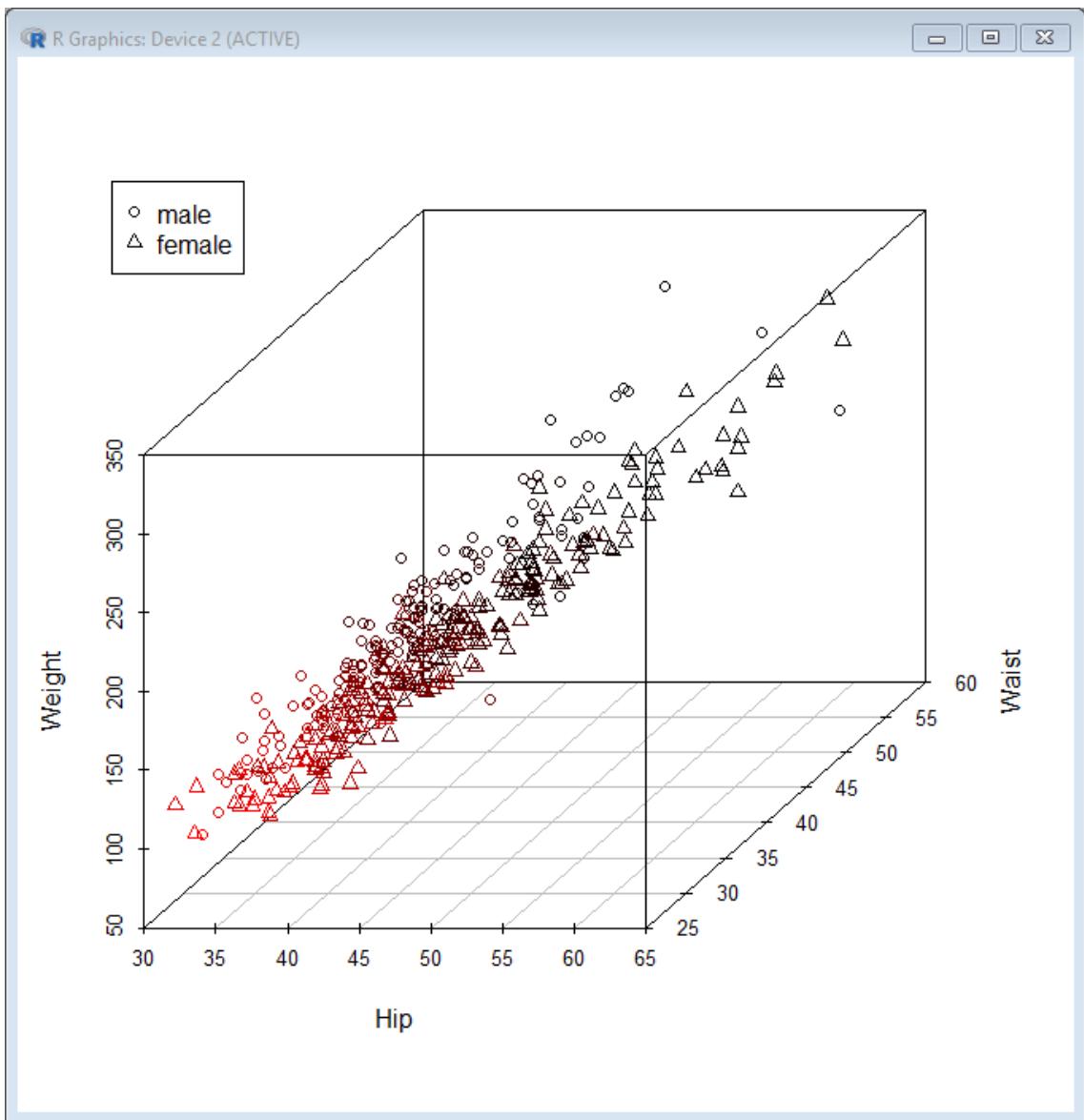
- a. Turn your attention back to the *diabetes* data frame found in the *faraway* package (you first looked at these data in Section 21.5.2).

Your goal is to produce a *scatterplot3d* plot of weight, hip, and waist measurements as per the following guidelines:

- Hip, waist, and weight variables should correspond to the *x*-axis, *y*-axis, and *z*-axis, respectively; provide neat axis titles.
  - Use built-in functionality to ensure the 3D depth is highlighted by color.
  - Choose two different point characters to reflect gender.
  - Place a simple legend referencing these two point characters and gender in the blank space in the upper-left area.
- b. Create a 3D scatterplot of the built-in *airquality* data, which you first met in Section 24.2.2, according to the following guidelines:
- Create a copy of the data frame using *na.omit* to remove all rows that contain missing values and work with this copy.
  - Plot wind speed and solar radiation against the *x*- and *y*-axes, respectively, using the *z*-axis to plot temperature.
  - Apply vertical dotted lines reaching up from the *x*-*y* plane to each observation.
  - The data in *airquality* are comprised of measurements taken over five months, from May to September. Each plotted point should take on the corresponding *pch* value from 1 to 5 respective to the order of these five months.
  - With a vector of 50 colors generated from the built-in *topo.colors* palette, use the categorization approach to ensure each plotted point is colored according to its ozone value.
  - Set a legend to reference the five point types according to month.
  - Set a color legend (using functionality from the *shape* package) to reference the ozone value accordingly.
  - Ensure the plot has neat axis, main, and legend titles.

## Solution 25.2

```
library("scatterplot3d")
#(a)
library("faraway")
scatterplot3d(x=diabetes$hip,y=diabetes$waist,z=diabetes$weight,highlight.3d=TRUE,pch=c(1,2)
[as.numeric(diabetes$gender)],xlab="Hip",ylab="Waist",zlab="Weight")
legend("topleft",legend=levels(diabetes$gender),pch=1:2)
```



#(b)

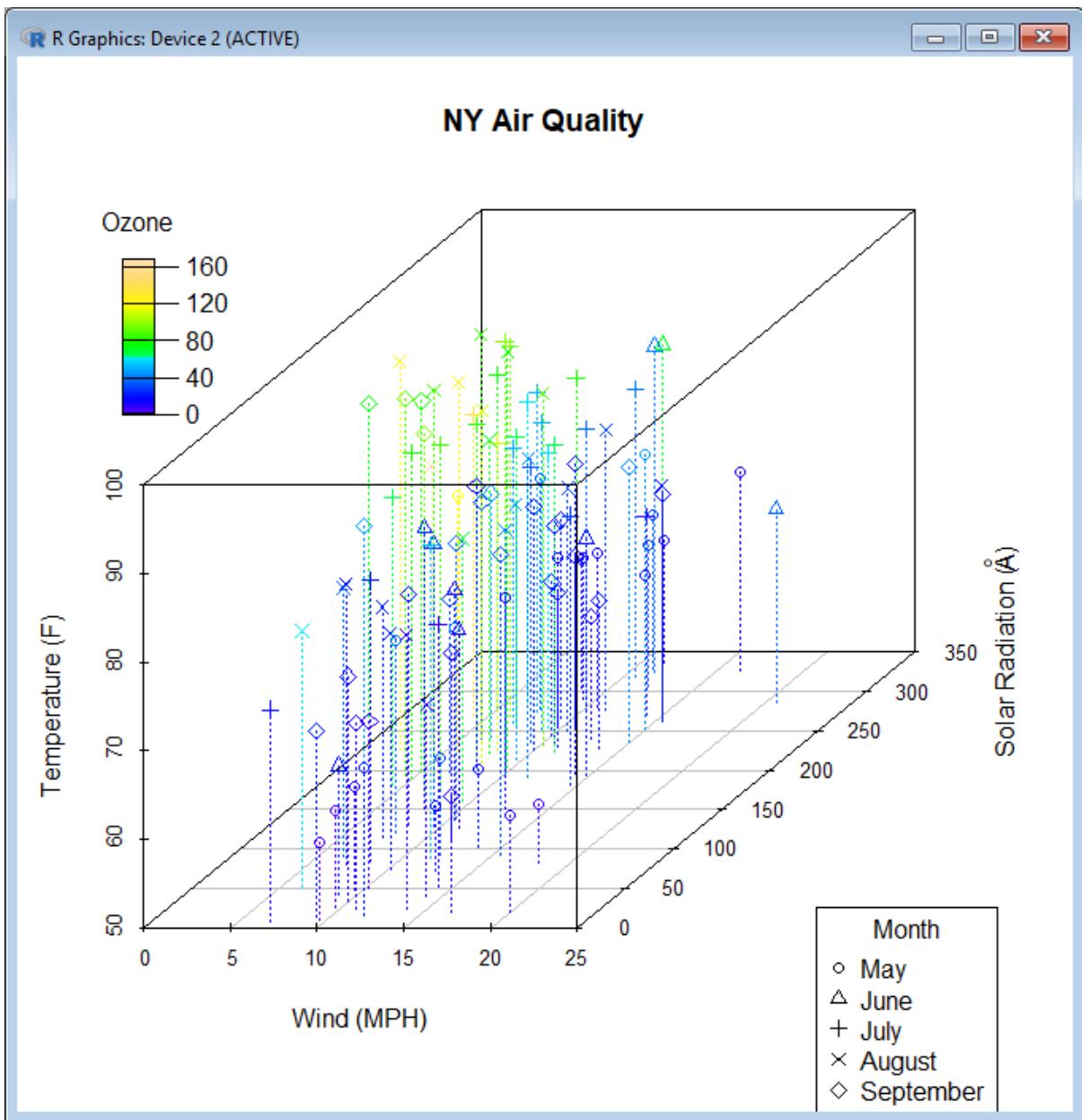
```

aq <- na.omit(airquality)
ozcols <- topo.colors(50)
colindex <-
cut(aq$Ozone,breaks=seq(min(aq$Ozone),max(aq$Ozone),length=51),include.lowest=TRUE)
scatterplot3d(x=aq$Wind,y=aq$Solar.R,z=aq$Temp,type="h",lty.hplot=3,color=ozcols[colindex],
pch=aq$Month-4,main="NY Air Quality",xlab="Wind (MPH)",ylab=expression(paste("Solar
Radiation (",ring(A),""))),zlab="Temperature (F)")

legend(locator(1),legend=c("May","June","July","August","September"),pch=1:5,title="Month",x
pd=TRUE)

library("shape")
colorlegend(ozcols,zlim=range(aq$Ozone),main="Ozone",zval=seq(0,160,40),posx=c(0.1,0.13),p
osy=c(0.7,0.9))

```



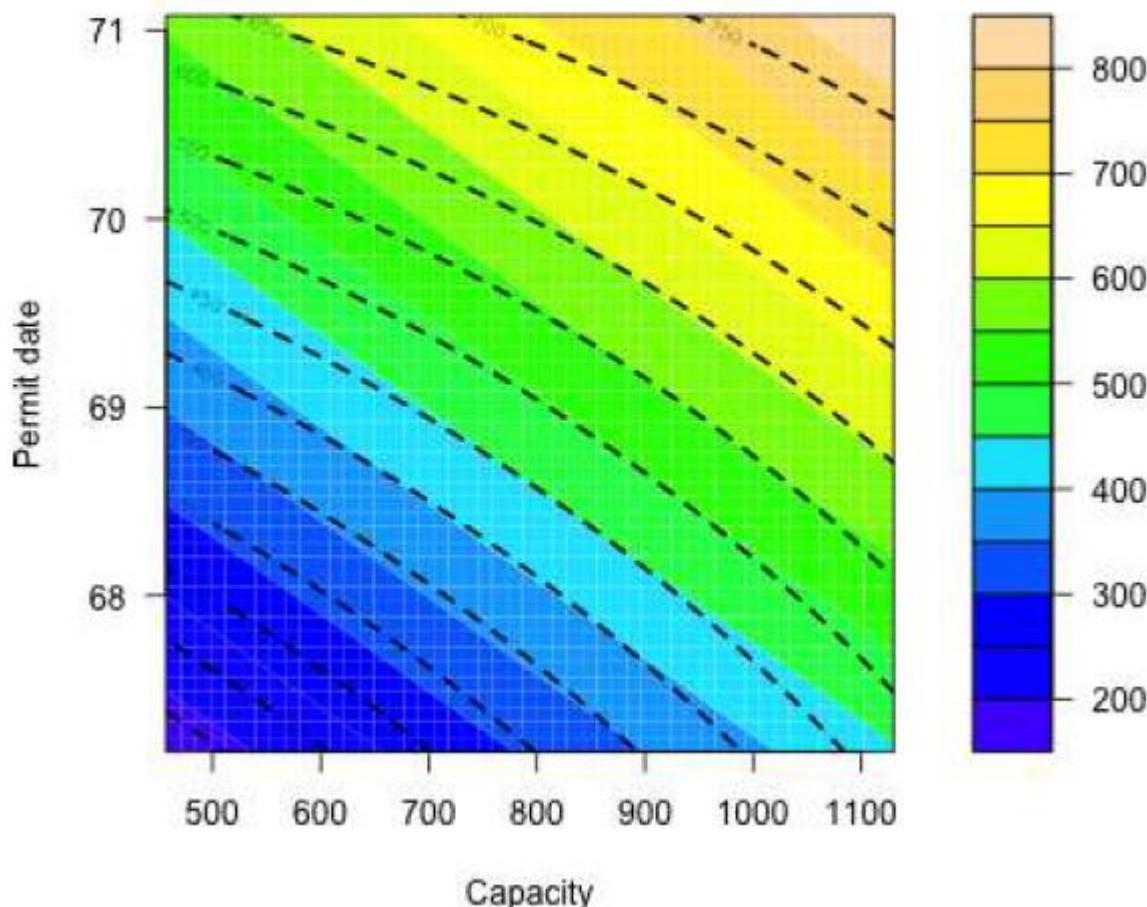
## Exercise 25.3

Remember that you inspected various multiple linear regression models of the cost of nuclear power plant construction in Chapters 21 and 22. The goal now will be to visually assess the impact of including/excluding an interactive term between two continuous predictors using contours. Revisit the *nuclear* data set, available when you load the *boot* package, and bring up the help file to refresh your memory of the variables present.

- a. Fit and summarize two linear models with construction cost as the response variable according to the following guidelines:
  - i. The first should account for main effects of the two predictors concerning the date of issue of the construction permit and plant capacity.
  - ii. The second, in addition to the two main effects, should include an interaction between permit issue date and capacity.
- b. Set up appropriate *z*-matrices for plotting each of these response surfaces. Each one should be based on a  $50 \times 50$  evaluation grid constructed using evenly spaced sequences in the capacity and date variables.
- c. Specify *mfrow* in *par* so that you can display default contour plots for the two response surfaces from (a)(i) and (a)(ii) next to one another. Do they appear similar? Does this tie in with the statistical significance (or lack thereof) of the interaction term in (a)(ii)?
- d. To directly compare the two surfaces, use your choice of built-in color palette to produce a filled contour plot of the main-effects-only model and superimpose the contour lines of the interactive model on it. Take note of the following:
  - This plot is achieved in a single call to *filled.contour*. Recall the special way you use *plot.axes* to draw additional features on an existing color-filled contour plot.
  - The contour lines of the interactive model can be added with an appropriate call to *contour*. Recall the use of the optional argument *add*.
  - The superimposed contours should be dashed lines of double thickness.
  - The *x*- and *y*-axes should be included and given tidy titles.
  - Add some brief text describing the filled contours versus the contour lines, with reference to the two versions of the construction cost model and with an additional call to *text* that makes use of a single mouse-clicked location from *locator* (see Section 23.3). Note that this call will need to fully relax clipping for the text to be visible in any of the margins.

My result is shown here.

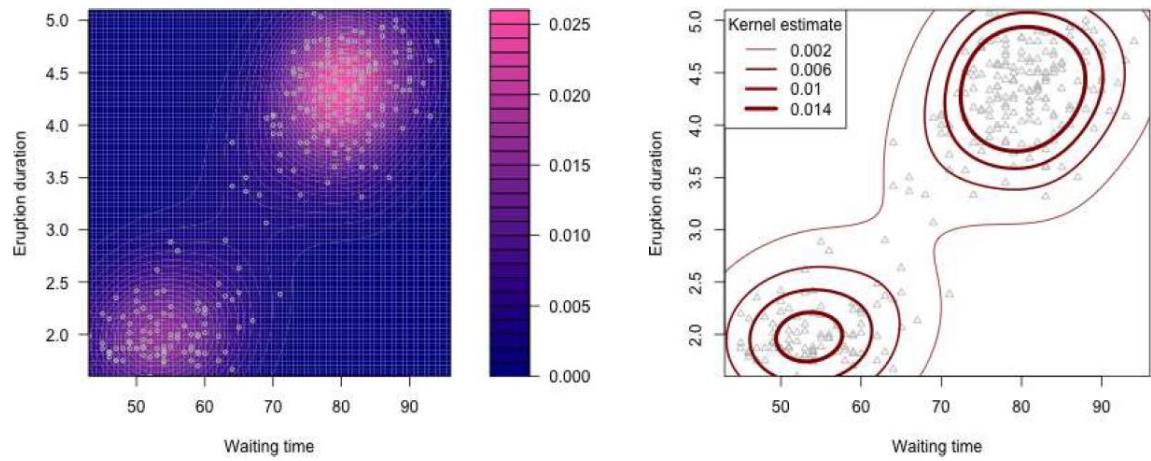
Color fill: Main effects only.  
Lines: Interaction included.



- e. Another built-in data frame in R, *faithful*, contains observations of waiting times and durations of eruptions of the Old Faithful geyser in Yellowstone National Park, Wyoming. See the documentation in `?faithful` for details. Plot the data with duration on the y-axis and waiting time on the x-axis.
- f. Estimate the bivariate density of these data via KDE using a  $100 \times 100$  evaluation grid and produce a default contour plot thereof.
- g. Create a filled contour plot of the kernel estimate using a custom palette that ranges from "darkblue" to "hotpink"; include the raw data as half-size gray points. Label the axes and titles appropriately.
- h. Replot the raw data as gray, 3/4-sized, type 2 point characters; set the style of the axes to restrict to exactly the ranges of the observed data; and ensure tidy axis titles and a main title. To this plot, add the contour lines of the density estimate at the specific levels obtained in a sequence from 0.002 to 0.014 in steps of 0.004. Suppress the labeling of the contours. The contour lines should be dark red and increase in line width thickness for higher levels of the density. Add a legend referencing the density level at each of these lines.

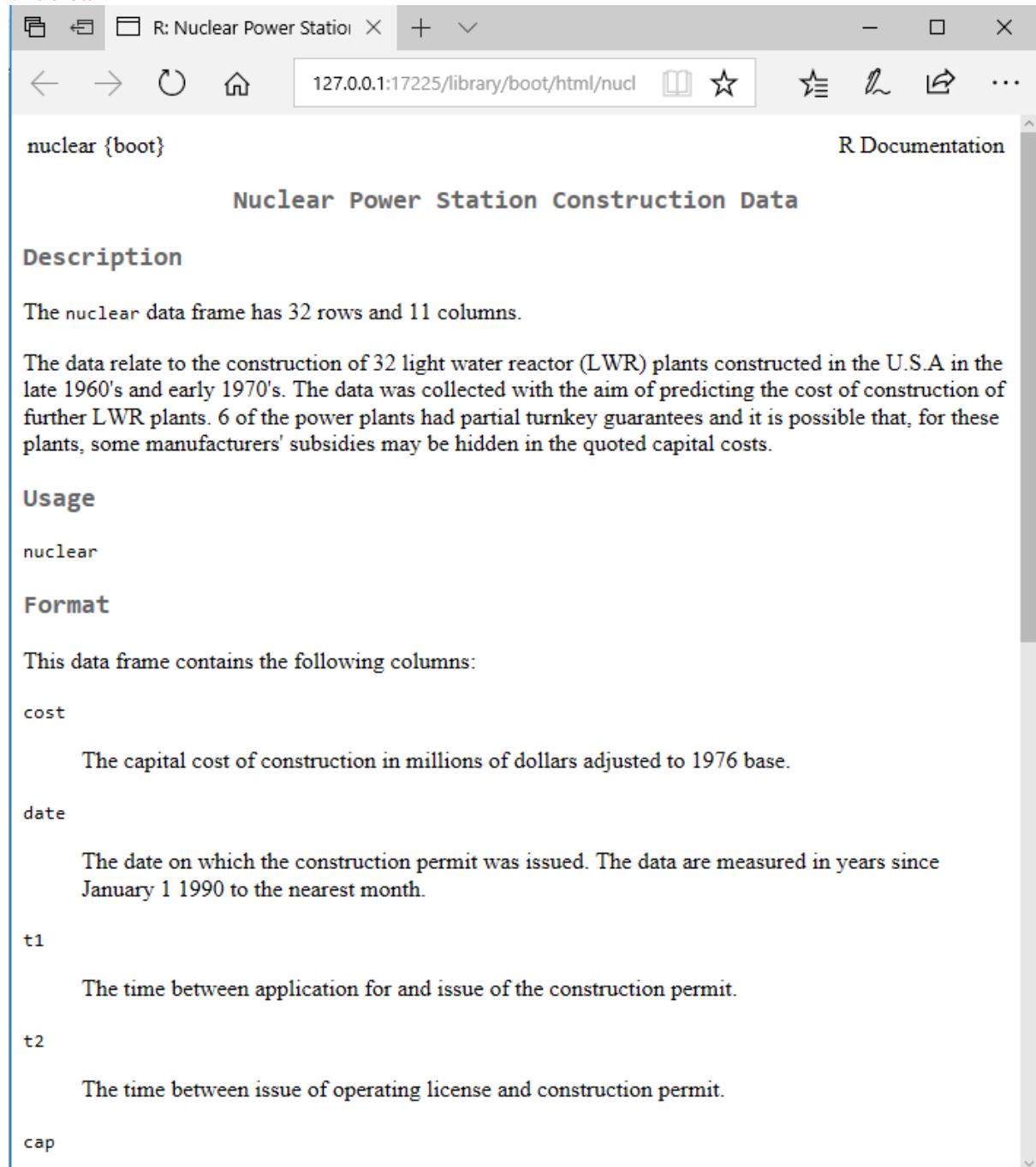
My plots for (g) and (h) are shown here.

### Old Faithful Geyser



## Solution 25.3

```
library("boot")
?nuclear
```



The screenshot shows a web browser window displaying the R Documentation for the 'nuclear' dataset. The URL in the address bar is 127.0.0.1:17225/library/boot/html/nuclear. The page title is 'Nuclear Power Station Construction Data'. The content includes a 'Description' section stating that the dataset has 32 rows and 11 columns, and a detailed description of the data relating to the construction of 32 LWR plants in the U.S.A. from the late 1960's to early 1970's. It also includes sections for 'Usage', 'Format', and descriptions of individual variables: cost, date, t1, t2, and cap.

nuclear {boot}

Nuclear Power Station Construction Data

**Description**

The `nuclear` data frame has 32 rows and 11 columns.

The data relate to the construction of 32 light water reactor (LWR) plants constructed in the U.S.A. in the late 1960's and early 1970's. The data was collected with the aim of predicting the cost of construction of further LWR plants. 6 of the power plants had partial turnkey guarantees and it is possible that, for these plants, some manufacturers' subsidies may be hidden in the quoted capital costs.

**Usage**

```
nuclear
```

**Format**

This data frame contains the following columns:

**cost**  
The capital cost of construction in millions of dollars adjusted to 1976 base.

**date**  
The date on which the construction permit was issued. The data are measured in years since January 1 1990 to the nearest month.

**t1**  
The time between application for and issue of the construction permit.

**t2**  
The time between issue of operating license and construction permit.

**cap**

```
#(a)
##(i)
nuc.fit1 <- lm(cost~cap+date,data=nuclear)
summary(nuc.fit1)
```

Call:  
`lm(formula = cost ~ cap + date, data = nuclear)`

Residuals:

Min	1Q	Median	3Q	Max
-146.351	-88.244	-2.202	58.457	267.741

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-6790.8792	1377.6683	-4.929	3.09e-05 ***
cap	0.4132	0.1076	3.840	0.000616 ***
date	100.7764	20.0696	5.021	2.39e-05 ***
---				

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 113.4 on 29 degrees of freedom  
Multiple R-squared: 0.5841, Adjusted R-squared: 0.5555  
F-statistic: 20.37 on 2 and 29 DF, p-value: 2.984e-06

```
##(ii)
nuc.fit2 <- lm(cost~cap*date,data=nuclear)
summary(nuc.fit2)
```

Call:

lm(formula = cost ~ cap \* date, data = nuclear)

Residuals:

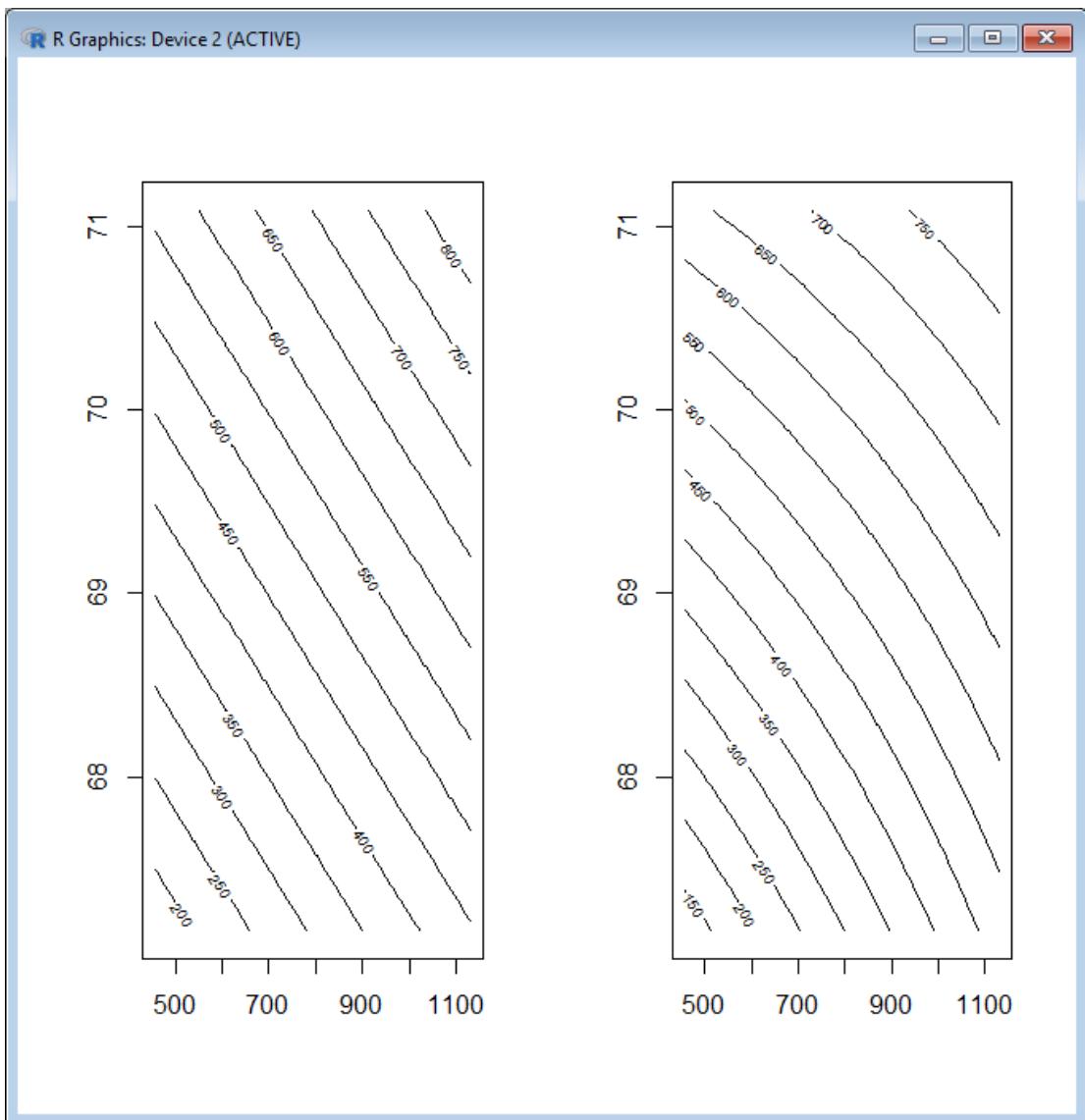
Min	1Q	Median	3Q	Max
-144.990	-91.625	-9.374	57.218	269.239

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-1.119e+04	6.698e+03	-1.670	0.106
cap	5.436e+00	7.487e+00	0.726	0.474
date	1.648e+02	9.754e+01	1.690	0.102
cap:date	-7.314e-02	1.090e-01	-0.671	0.508

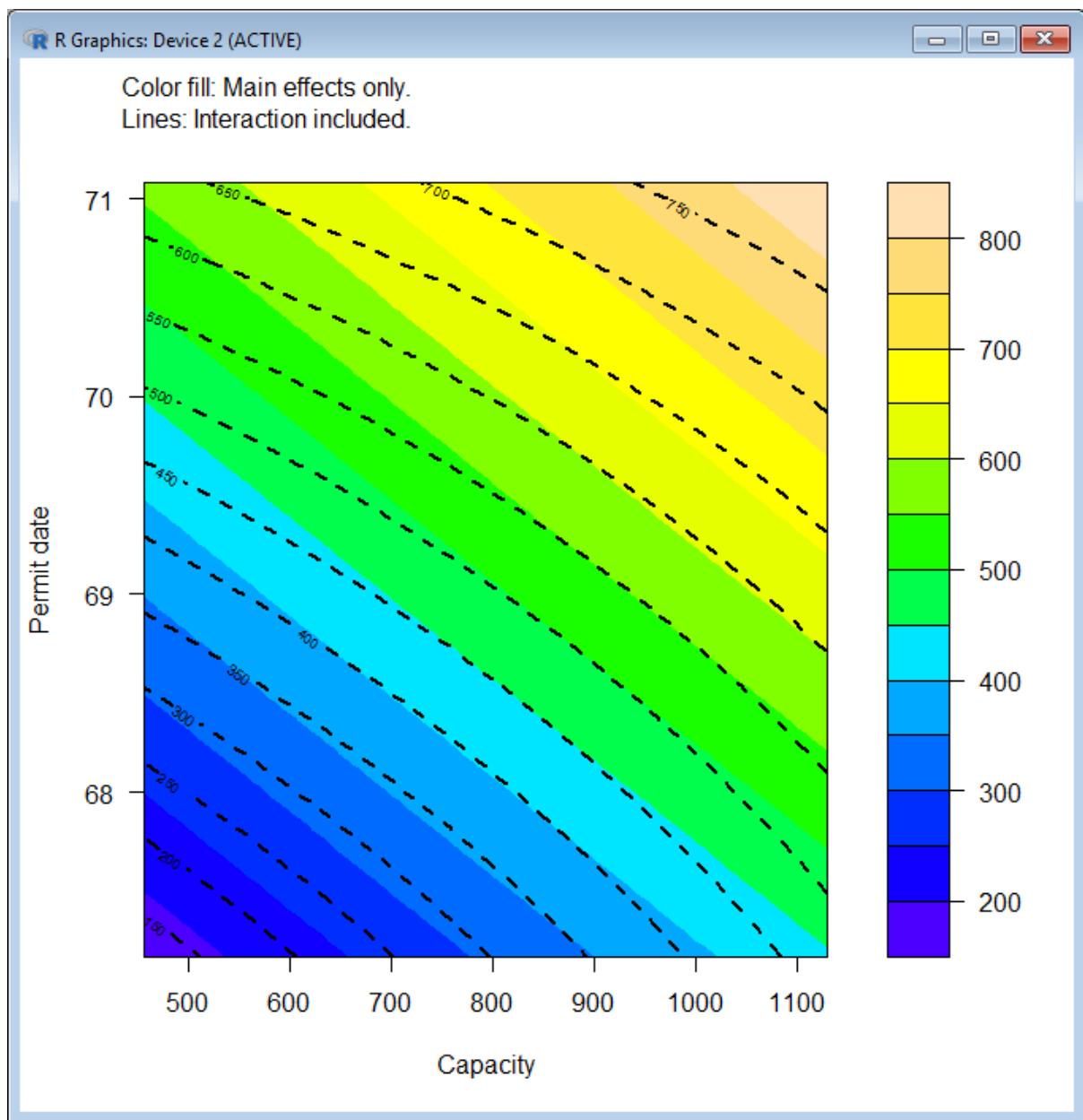
Residual standard error: 114.5 on 28 degrees of freedom  
Multiple R-squared: 0.5907, Adjusted R-squared: 0.5469  
F-statistic: 13.47 on 3 and 28 DF, p-value: 1.261e-05

```
#(b)
capseq <- seq(min(nuclear$cap),max(nuclear$cap),length=50)
datseq <- seq(min(nuclear$date),max(nuclear$date),length=50)
capdat <- expand.grid(cap=capseq,date=datseq)
nuc.pred1 <- matrix(predict(nuc.fit1,newdata=capdat),50,50)
nuc.pred2 <- matrix(predict(nuc.fit2,newdata=capdat),50,50)
#(c)
par(mfrow=c(1,2))
contour(x=capseq,y=datseq,z=nuc.pred1)
contour(x=capseq,y=datseq,z=nuc.pred2)
```



# The two response surfaces appear extremely similar, with only a slight curvature appearing in the surface associated with the interactive model. This similarity is to be expected based on the model summaries from (a), which don't provide any evidence supporting the need to include the interaction.

```
#(d)
filled.contour(x=capseq,y=datseq,z=nuc.pred1,xlab="Capacity",ylab="Permit
date",color.palette=topo.colors,plot.axes={axis(1);axis(2);contour(capseq,datseq,nuc.pred2,add=T
RUE,lwd=2,lty=2)})
text(locator(1),label=c("Color fill: Main effects only.\nLines: Interaction included."),xpd=NA)
```



#(e)

?faithful

```
plot(faithful$eruptions~faithful$waiting,xlab="Waiting time",ylab="Eruption duration",main="Old Faithful Geyser")
```

R: Nuclear Power Station C R: Old Faithful Geyser I 127.0.0.1:17225/library/datasets/html/f

## faithful {datasets}

### Old Faithful Geyser Data

#### Description

Waiting time between eruptions and the duration of the eruption for the Old Faithful geyser in Yellowstone National Park, Wyoming, USA.

#### Usage

```
faithful
```

#### Format

A data frame with 272 observations on 2 variables.

```
[.1] eruptions numeric Eruption time in mins  
[.2] waiting numeric Waiting time to next eruption (in mins)
```

#### Details

A closer look at `faithful$eruptions` reveals that these are heavily rounded times originally in seconds, where multiples of 5 are more frequent than expected under non-human measurement. For a better version of the eruption times, see the example below.

There are many versions of this dataset around: Azzalini and Bowman (1990) use a more complete version.

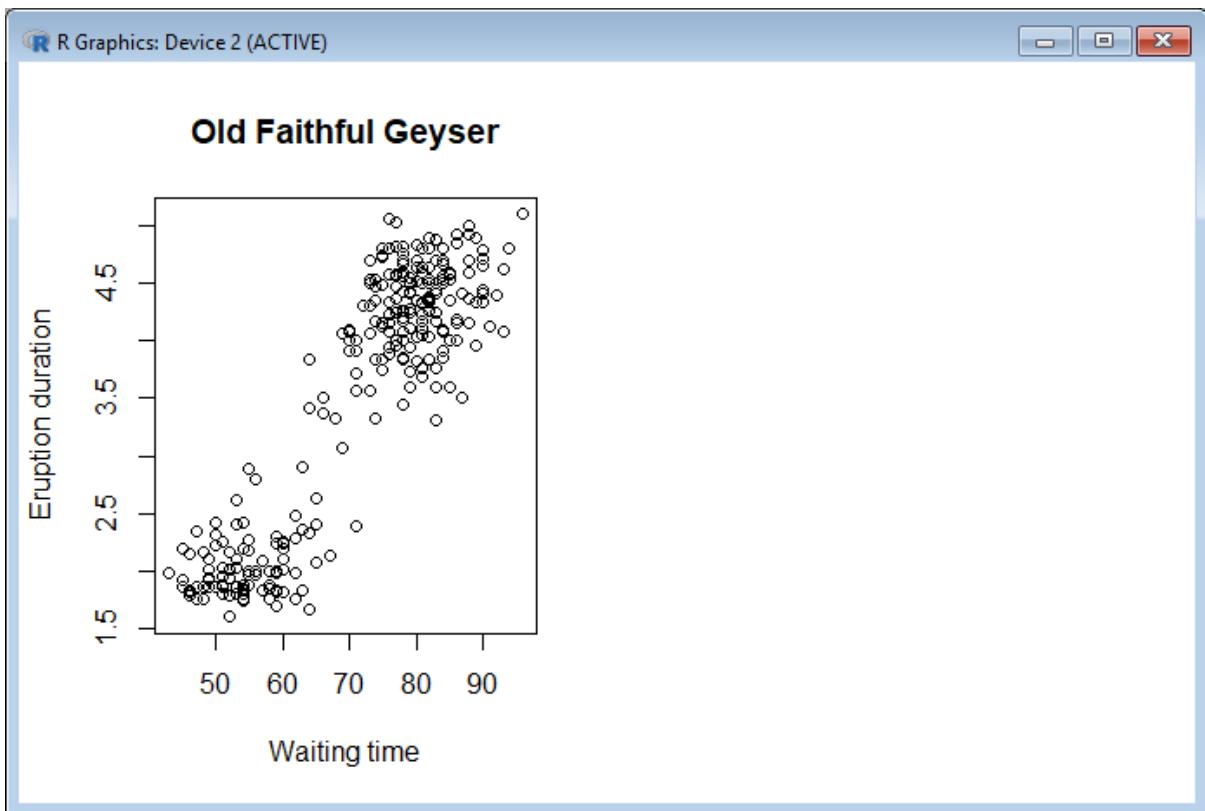
#### Source

W. Härdle.

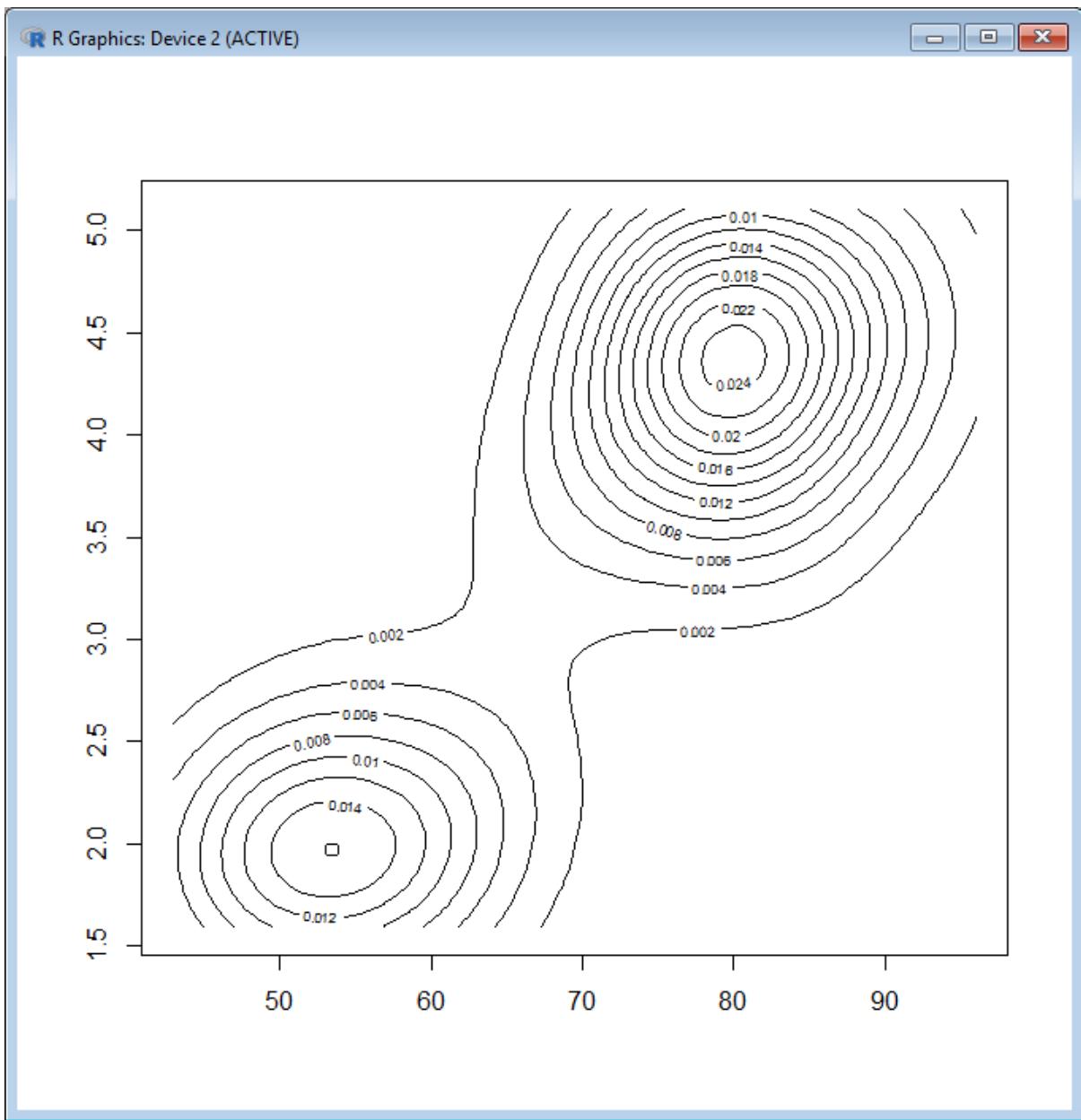
#### References

Härdle, W. (1991) *Smoothing Techniques with Implementation in S*. New York: Springer.

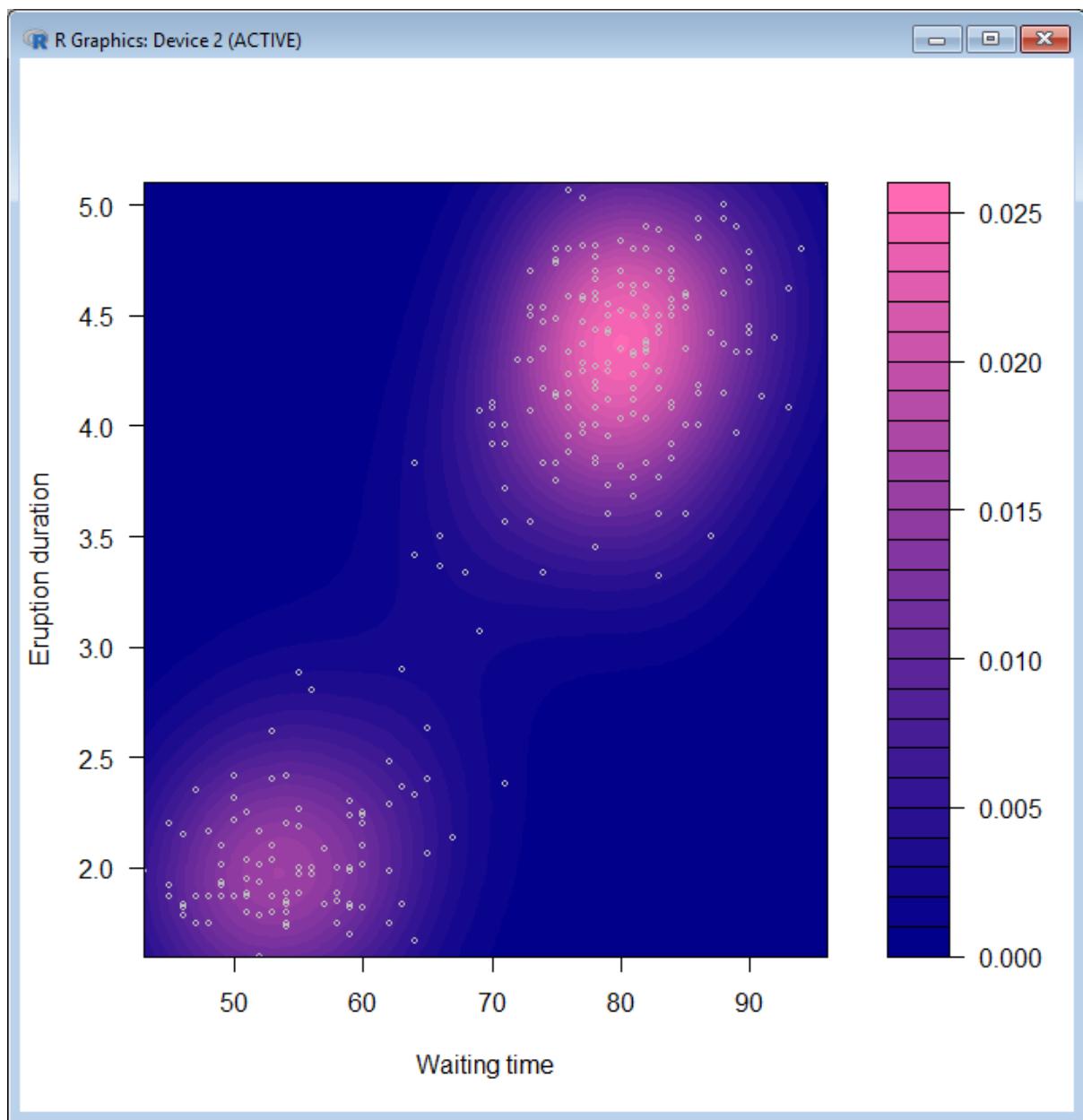
Azzalini, A. and Bowman, A. W. (1990). A look at some data on the Old Faithful geyser. *Applied*



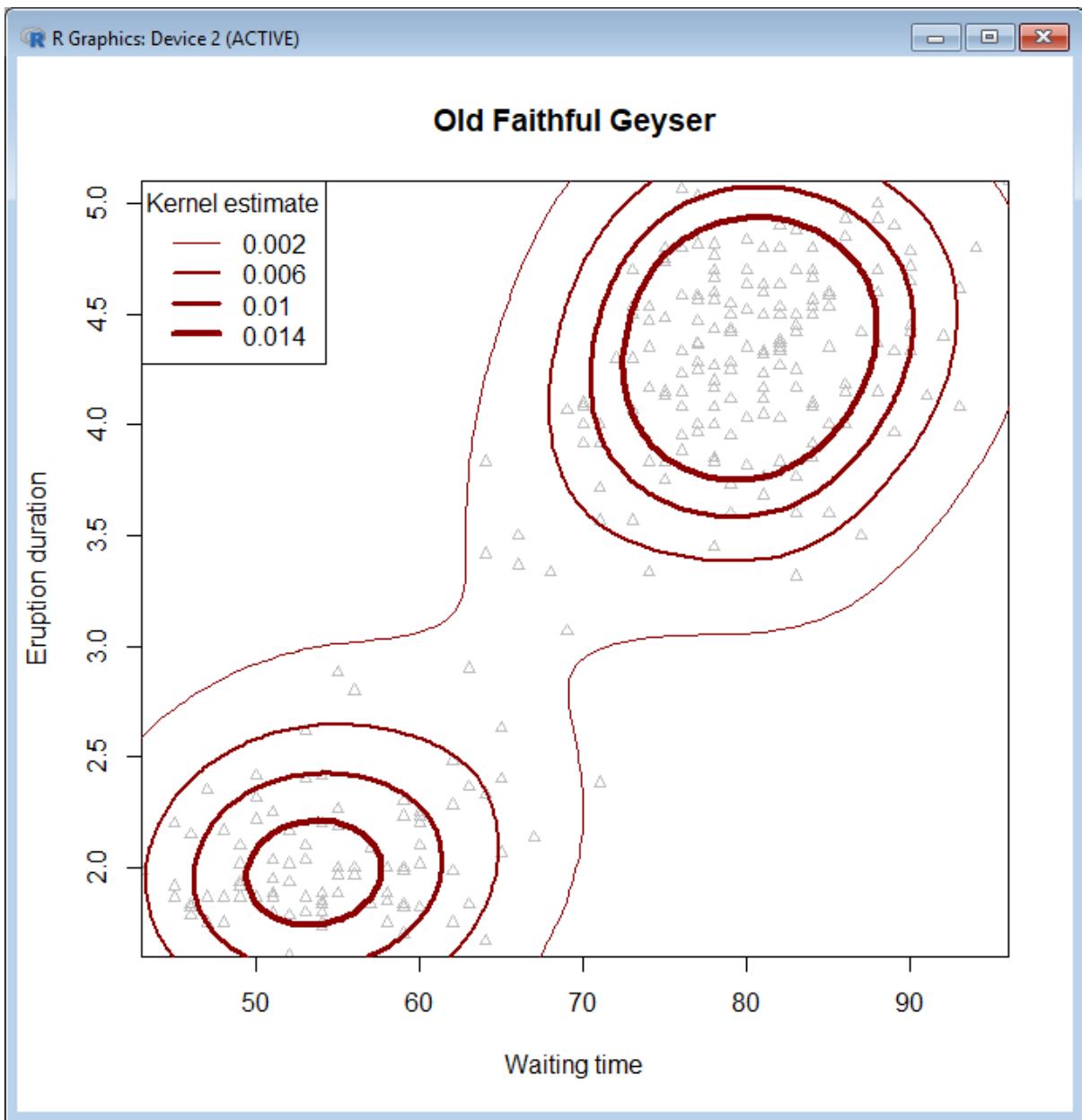
```
#(f)
library("MASS")
faith.dens <- kde2d(x=faithful$waiting,y=faithful$eruptions,n=100)
contour(faith.dens$x,faith.dens$y,faith.dens$z)
```



```
#(g)
faith.cols <- colorRampPalette(c("darkblue","hotpink"))
filled.contour(faith.dens$x,faith.dens$y,faith.dens$z,color.palette=faith.cols,xlab="Waiting
time",ylab="Eruption
duration",plot.axes={axis(1);axis(2);points(faithful$waiting,faithful$eruption,cex=0.5,col="gray")
})
```



```
#(h)
plot(faithful$eruptions~faithful$waiting,col="gray",pch=2,cex=0.75,xaxs="i",yaxs="i",xlab="Waiting time",ylab="Eruption duration",main="Old Faithful Geyser")
contour(faith.dens$x,faith.dens$y,faith.dens$z,levels=seq(0.002,0.014,0.004),col="red4",add=TRUE,drawlabels=FALSE,lwd=1:4)
legend("topleft",legend=seq(0.002,0.014,0.004),lwd=1:4,col="red4",title="Kernel estimate")
```

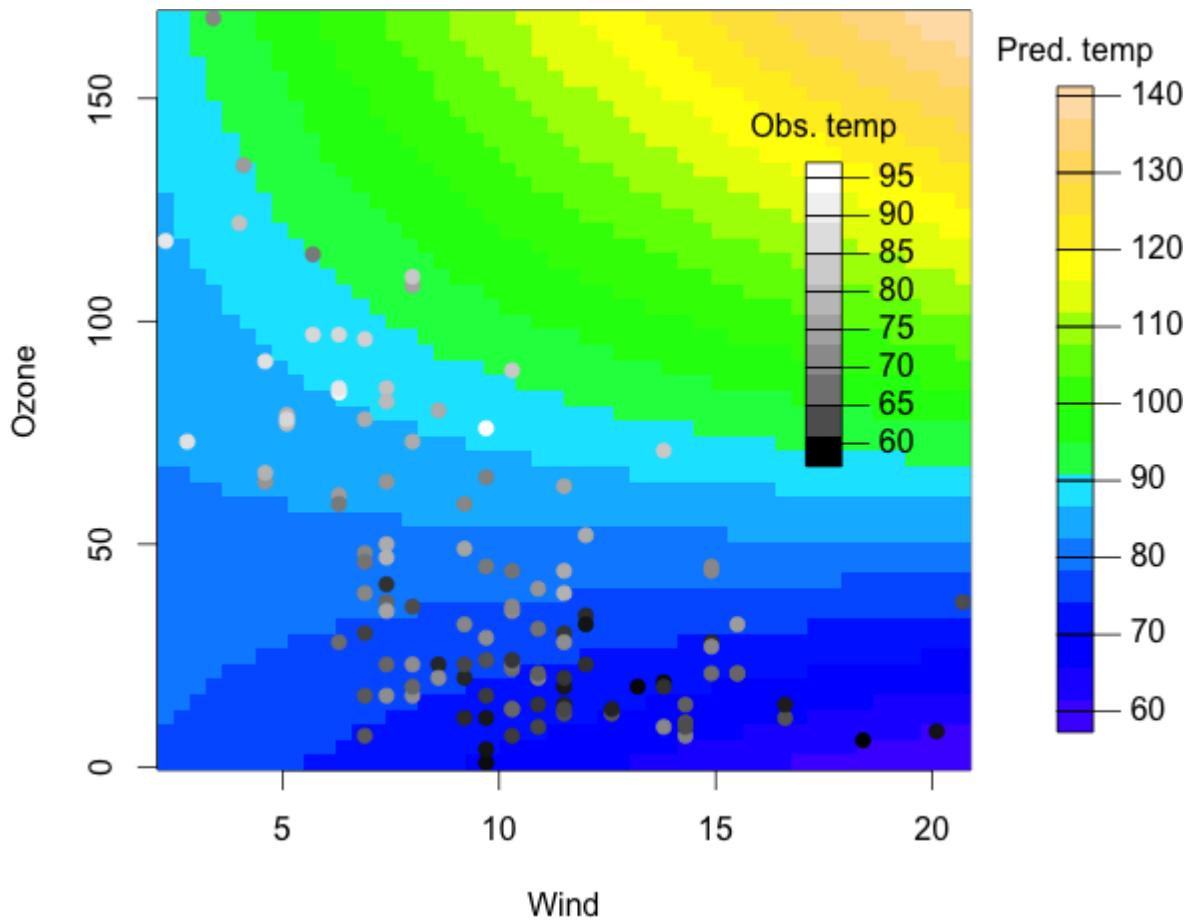


## Exercise 25.4

Revisit the built-in *airquality* data set and take a look at the help file to refresh your memory of the variables present. Create a copy of the data frame: select the columns pertaining to daily temperature, wind speed, and ozone level and use *na.omit* to remove any records with missing values.

- a. From your explorations of these data in Chapter 24, there appears to be an association among daily temperature, wind speed, and ozone level. Fit a multiple linear regression model that aims to predict mean temperature based on the wind speed and ozone level, including an interactive effect. Summarize the resulting object.
- b. Using the model from (a), construct a *z*-matrix of predicted mean daily temperature based on a  $50 \times 50$  evaluation grid in both wind speed and ozone.
- c. Create a pixel image of the response surface, superimposing the raw observations as per the following:
  - A graphics device should be initialized based on bottom, left, top, and right margin lines of 5, 4, 4, and 6, respectively.
  - 20 colors from the built-in *topo.colors* palette should be used to produce the image; include tidy axis titles.
  - Revisit the *normalize* function defined in Section 25.1.4 and use the built-in function *gray* to generate a vector of gray colors (refer to Section 25.1.2) based on the normalized raw temperature observations. Superimpose the raw observations based on wind speed and ozone onto the pixel image, using the gray color vector to indicate the corresponding temperature observations.
  - Two separate calls should then be made to *colorlegend* of the *shape* package. The first should appear in the space on the right margin, referencing the surface itself. The second should use the built-in *gray.colors* function, setting the optional arguments *start=0* and *end=1*, to generate 10 shades of gray for use in the legend that references the raw temperature observations of the superimposed points. This legend should reside on top of the pixel image itself, in the upper-right quadrant where there are no raw observations.
  - Both legends should have appropriate titles, and you may need to experiment a little with the *posx* and *posy* arguments to find satisfactory placement.

My result of this plotting exercise appears here.



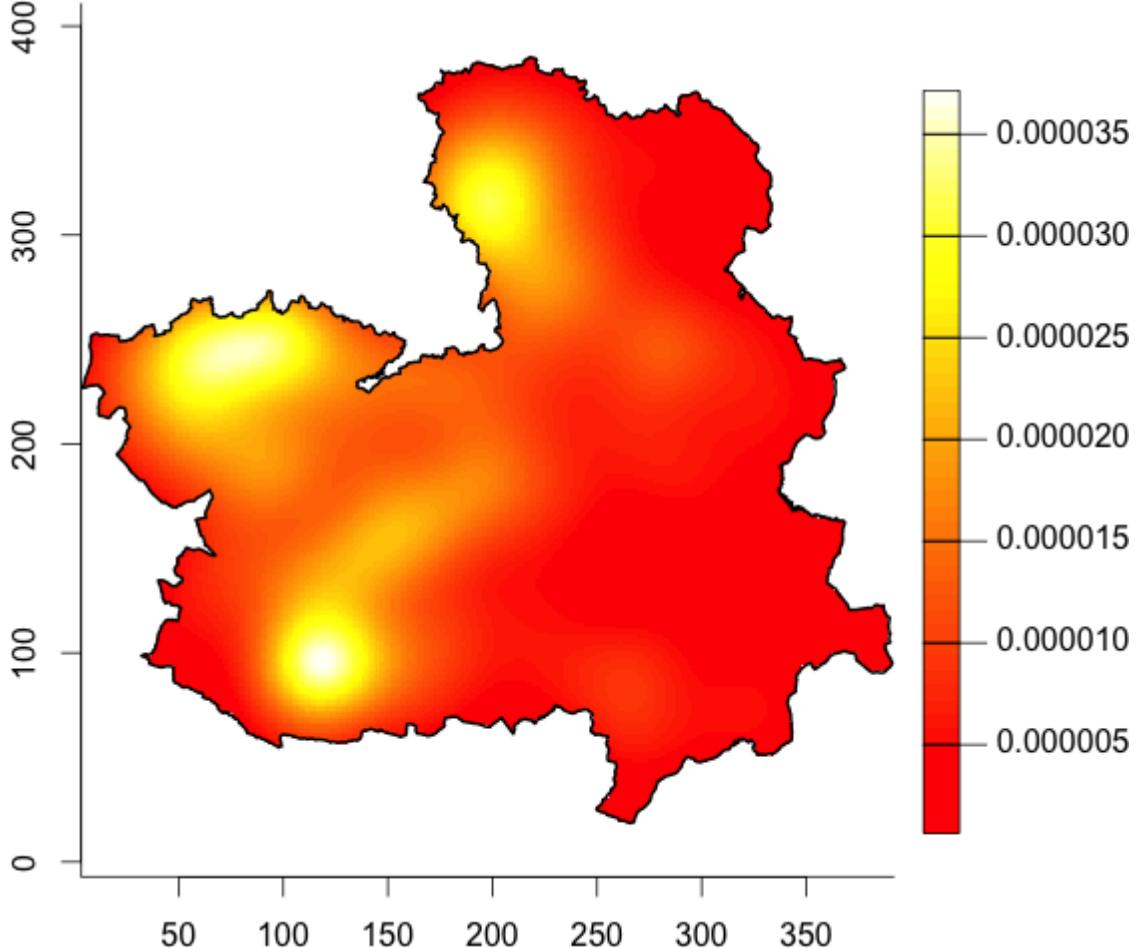
In Section 25.5.2, you used the *chorley* data set in creating a pixel image truncated to a subset of the overall rectangular evaluation grid. Ensure *spatstat* is loaded in your current R session and execute the following two lines:

```
R> fire <- split(clmfires)$intentional
R> firewin <- clmfires>window
```

This extracts the 1,786 locations of fires recorded as intentionally lit in a particular region of Spain. The spatial coordinates can be extracted as the *\$x* and *\$y* members of *fire*, and the geographical region itself is stored as a polygon in *firewin* (of the same class as the *chorley\$window* object you looked at earlier). See the documentation obtained with *?clmfires* for further details.

- d. Using the total *x*- and *y*-range of the study region, use *kde2d* from the *MASS* package to calculate a bivariate kernel estimate of the probability density function of the spatial dispersion of intentionally lit fires. The KDE surface should be calculated based on a  $256 \times 256$  evaluation grid.
- e. Identify all points on the rectangular revaluation grid that fall outside the geographical region using *expand.grid* in conjunction with *inside.owin*. Set all corresponding pixels of the density surface to *NA*.
- f. Construct a pixel image of the truncated density, as per the following:
  - The graphics device should have three lines of space on the bottom, left, and top of the plot region and should have seven lines on the right.
  - In producing the image itself, you should use 50 colors generated from the built-in *heat.colors* palette. A one-to-one aspect ratio should be maintained, the axis titles should be suppressed, and the box type set to be an *L* shape.

- The geographical study region should be superimposed onto the image using a double-width line.
- Using `colorlegend` from `shape`, a color legend referencing the density with an appropriate title should be placed to the right of the image. You’ll need to experiment with the `posx` argument for placement. Label the legend at a sequence from  $5e-6$  to  $35e-6$  in steps of  $5e-6$  (refer to Section 2.1.3 for an explanation of e-notation); also, ensure these labels are able to display up to six decimal places of precision.



## Solution 25.4

```
aq <- na.omit(airquality[,c("Temp","Wind","Ozone")])
#(a)
aq.fit <- lm(Temp~Wind*Ozone,aq)
summary(aq.fit)
```

Call:

```
lm(formula = Temp ~ Wind * Ozone, data = aq)
```

Residuals:

Min	1Q	Median	3Q	Max
-15.2008	-4.1492	0.5976	4.6656	13.4738

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	80.268679	3.141135	25.554	< 2e-16 ***
Wind	-1.133278	0.275753	-4.110	7.57e-05 ***
Ozone	0.015800	0.044790	0.353	0.725

```

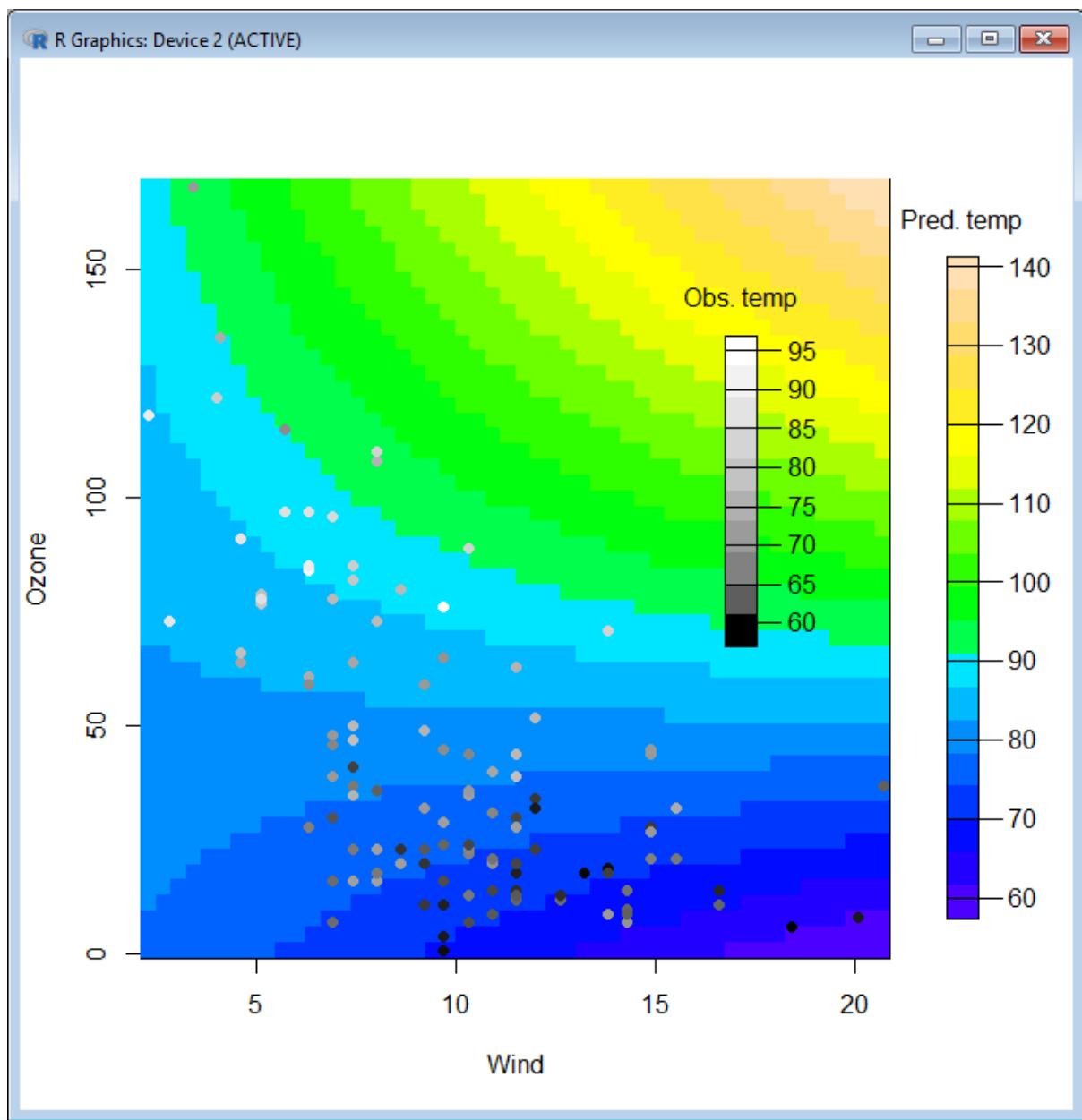
Wind:Ozone 0.023505 0.005687 4.133 6.93e-05 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.326 on 112 degrees of freedom
Multiple R-squared: 0.5668, Adjusted R-squared: 0.5552
F-statistic: 48.84 on 3 and 112 DF, p-value: < 2.2e-16

#(b)
res <- 50
winseq <- seq(min(aq$Wind),max(aq$Wind),length=res)
ozoseq <- seq(min(aq$Ozone),max(aq$Ozone),length=res)
winozo <- expand.grid(Wind=winseq,Ozone=ozoseq)
aq.pred <- matrix(predict(aq.fit,newdata=winozo),res,res)

#(c)
normalize <- function(datavec){
  lo <- min(datavec,na.rm=TRUE)
  up <- max(datavec,na.rm=TRUE)
  datanorm <- (datavec-lo)/(up-lo)
  return(datanorm)
}
library("shape")
par(mar=c(5,4,4,6))
image(x=winseq,y=ozoseq,z=aq.pred,col=topo.colors(20),xlab="Wind",ylab="Ozone")
points(aq$Wind,aq$Ozone,col=gray(normalize(aq$Temp)),pch=19)
colorlegend(col=topo.colors(20),zlim=range(aq.pred),zval=seq(60,140,10),posx=c(0.88,0.91),main="Pred. temp")
colorlegend(col=gray.colors(10,start=0,end=1),zlim=range(aq$Temp),zval=seq(60,95,5),posx=c(0.67,0.7),posy=c(0.4,0.8),main="Obs. temp")

```



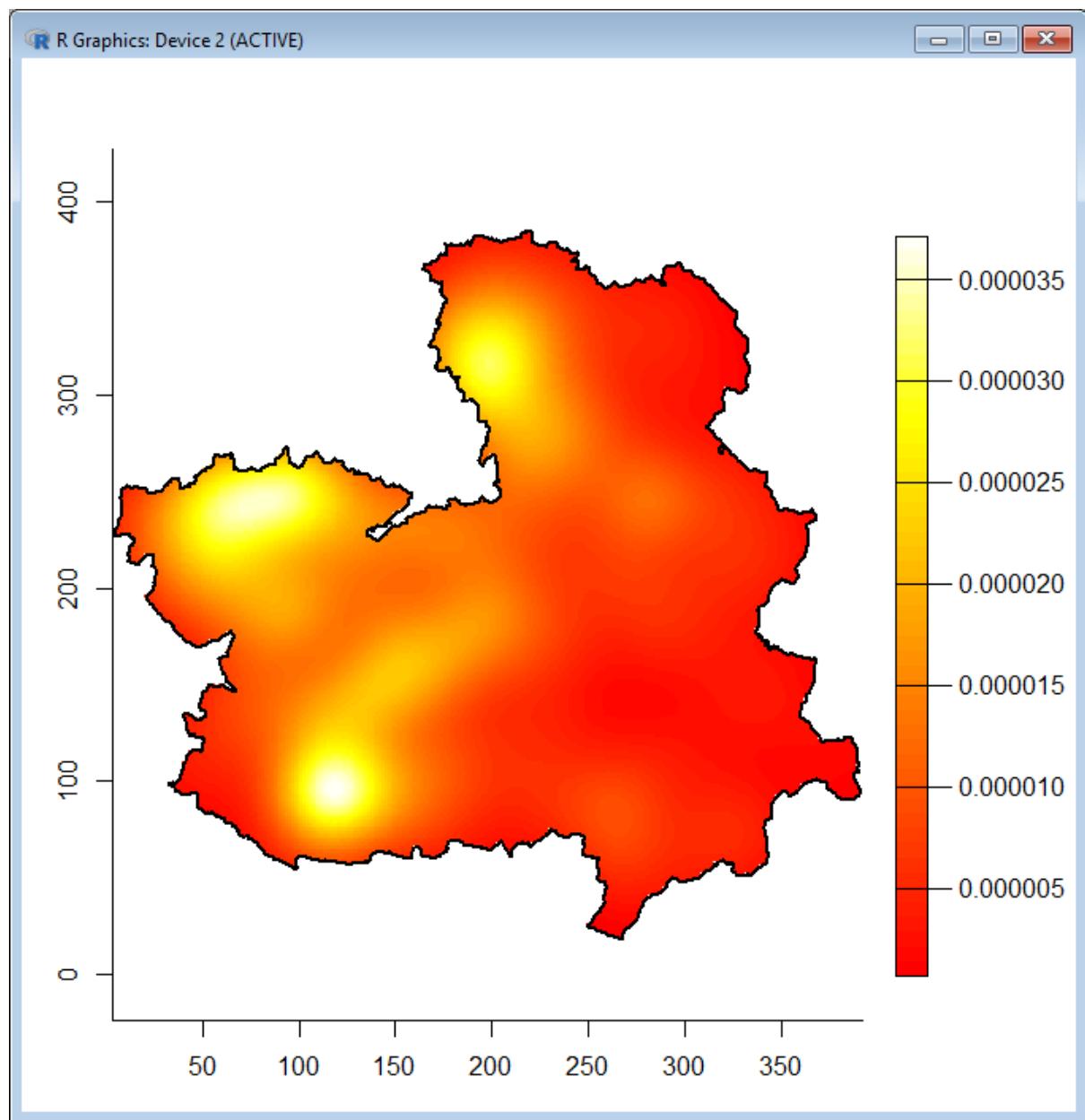
```
#(d)
library("spatstat")
?clmfires
fire <- split(clmfires)$intentional
firewin <- clmfires$window
firewin.xr <- firewin$xrange
firewin.yr <- firewin$yrange
fire.dens <- kde2d(x=fire$x,fire$y,n=256,lims=c(firewin.xr,firewin.yr))
```

```

#(e)
fire.xy <- expand.grid(fire.dens$x,fire.dens$y)
fire.outside <- !inside.owin(x=fire.xy[,1],y=fire.xy[,2],w=firewin)
fire.dens$z[fire.outside] <- NA

#(f)
library("shape")
par(mar=c(3,3,3,7))
image(fire.dens$x,fire.dens$y,fire.dens$z,col=heat.colors(50),asp=1,bty="l",ann=FALSE)
plot(firewin,lwd=2,add=TRUE)
colorlegend(col=heat.colors(50),posx=c(0.83,0.86),zlim=range(fire.dens$z,na.rm=TRUE),zval=seq(5e-6,35e-6,5e-6),digit=6)

```



## Exercise 25.5

In Exercise 25.3 (a), you revisited the *nuclear* data set from the *boot* package and fitted two multiple linear regression models aiming to model mean construction cost by permit date issue and plant capacity—one with main effects only and the other including an interaction term between the two continuous predictors.

- a. Refit the two versions of the model and produce perspective plots of the response surfaces based again on a  $50 \times 50$  evaluation grid, taking the following into account:

Use *mfrow* in a call to *par* to display the two perspective plots next to each other. In the same call to *par*, override the default figure margins to have only one line of space on each side (*par* is explored in this role in Chapter 23).

Use *zlim* to ensure both plots are displayed on the same scale of vertical axis, spin each one horizontally 25 degrees, and ensure detailed axis markings and tidy titles.

Is there any visual indication that the presence of the interaction term has had any meaningful impact on modeling the response?

- b. Start a fresh plot. To get a better idea of the difference between the two surfaces, produce a perspective plot of the *z*-matrix obtained as the elementwise difference between the two individual *z*-matrices for the two fitted models in (a). What, in general, is the effect of including the interaction term?

Turn your attention back to the topographical information on the Auckland volcano, as the built-in R object *volcano*: an  $87 \times 61$  matrix of elevation values (in meters). You first looked at this in Section 25.4.1 as a contour plot.

- c. Produce the most basic, default perspective plot of the volcano, using simple integer sequences for the *x*- and *y*-coordinates.

- d. The plot in (c) is decidedly unappealing for a number of reasons.

Produce a more realistic depiction of the volcano as per the following:

- Use a new graphics device with the margin widths reset to one, one, one, and four lines on the bottom, left, top, and right, respectively.
- The help file *?volcano* reveals the *x*- and *y*-coordinates to which the volcano *z*-matrix corresponds is in 10-meter units.

Using *scale* and altering *expand*, replot the surface with the correct aspect ratio in all three axes.

- Suppress all axis tick marks and notation using *axes*.
- The facets should be colored according to 50 colors generated from the built-in *terrain.colors* palette, and the facet border lines should be suppressed.
- Find your choice of visually appealing viewing angle.
- Use *colorlegend* from the *shape* package to place a color legend referencing elevation in meters in the space to the right of the plot. Experiment with the arguments to find appropriate placement and tick mark labels.

Here's my version of the improved plot:

In Exercise 25.4, you looked at the spatial distribution of intentionally lit fires in a region of Spain. Ensure the *spatstat* package is loaded, and then rerun the following lines to obtain the relevant data objects:

```
R> fire <- split(clmfires)$intentional
```

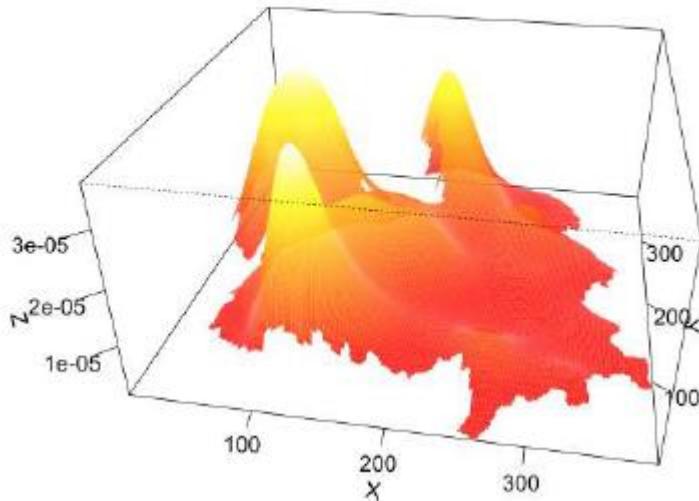
```
R> firewin <- clmfires$window
```

- e. Borrow the code from Exercise 25.4 (d) and (e) to reproduce the kernel density estimate of this dispersion of observations, based on a  $256 \times 256$  evaluation grid, truncated to the study region. Then, display it as a perspective plot according to the following:

- Just as with the pixel image, use 50 colors from the built-in *heat.colors* palette to color the facets by *z* value. Note the truncated *z*-matrix for this function contains *NA* values.

- Border lines on the surface should be suppressed, and you should find your preferred choice of viewing angle.
- Use *scale* to ensure the correct spatial aspect ratio. In doing so, you’ll also need to adjust the *z*-axis expansion by a factor of around 5,000,000 for the density surface to be visible along the vertical, given the natural scaling of the density estimate on the specified evaluation grid.
- Employ detailed axis labeling and simply entitle the axes “*X*”, “*Y*”, and “*Z*” as appropriate.

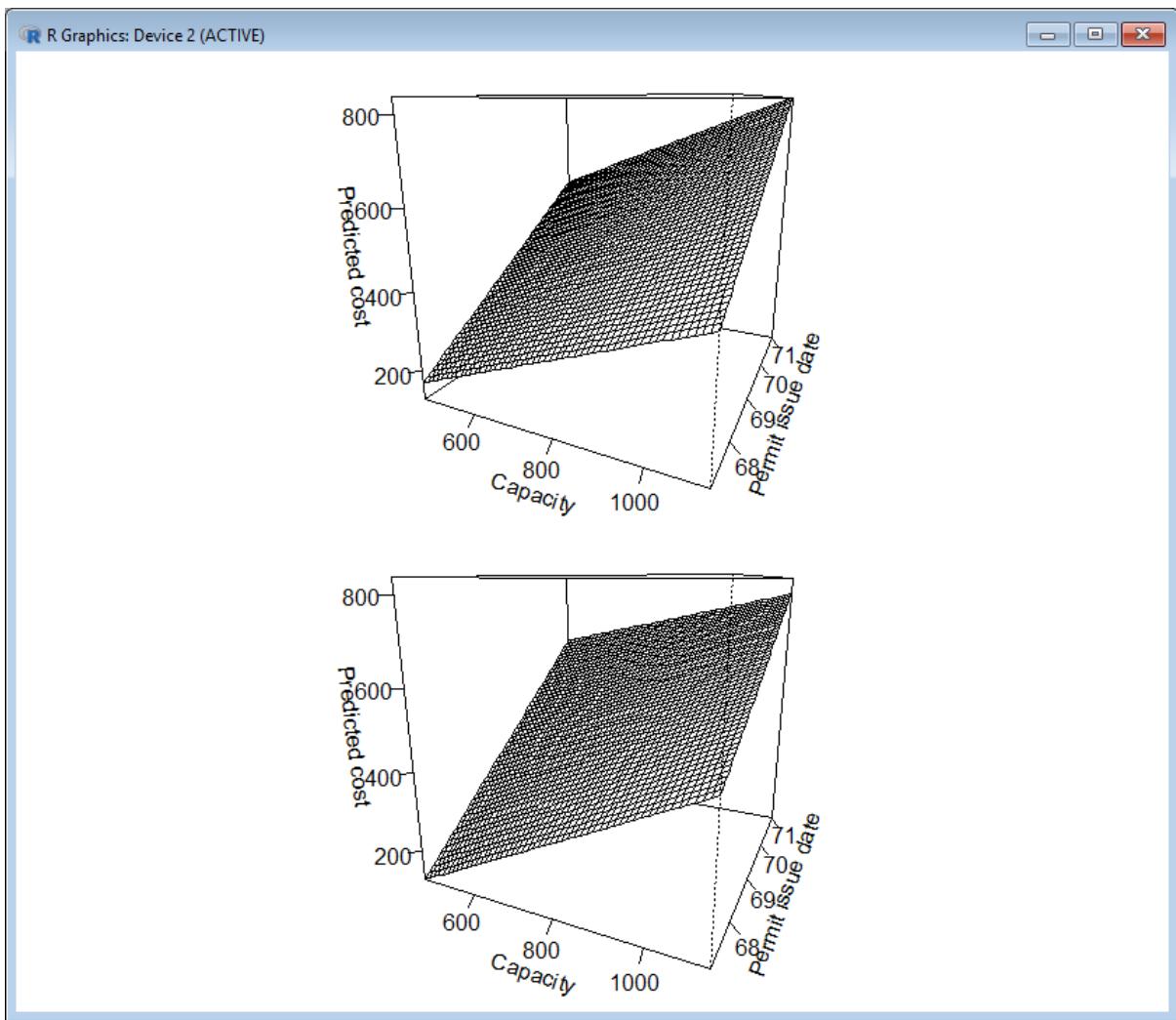
My product is given here.



- f. Use the *persprot* function defined in Section 25.6.3 to view the surface from (e), setting *skip*=10.

## Solution 25.5

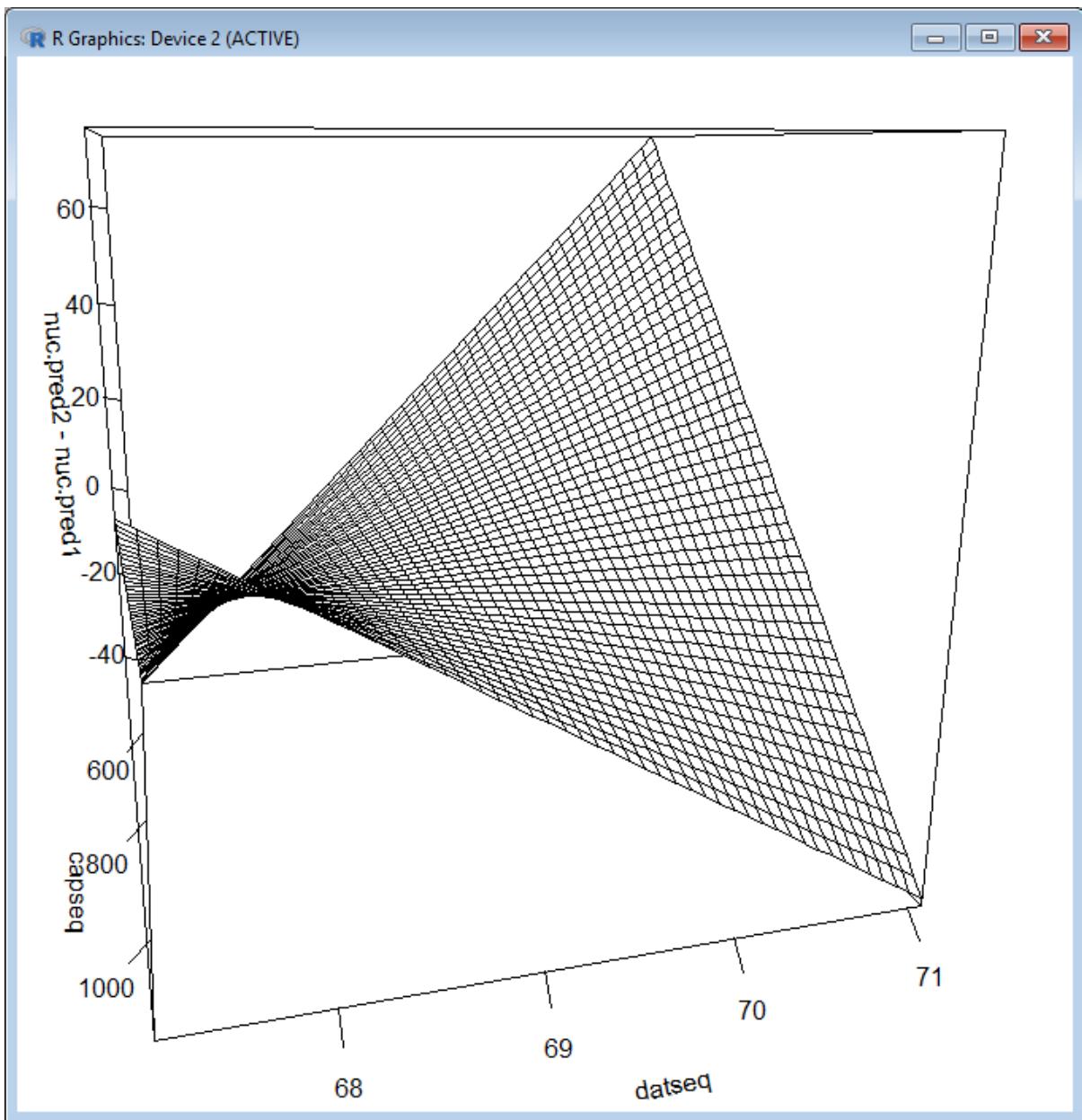
```
library("boot")
#(a)
nuc.fit1 <- lm(cost~cap+date,data=nuclear)
nuc.fit2 <- lm(cost~cap*date,data=nuclear)
capseq <- seq(min(nuclear$cap),max(nuclear$cap),length=50)
datseq <- seq(min(nuclear$date),max(nuclear$date),length=50)
capdat <- expand.grid(cap=capseq,date=datseq)
nuc.pred1 <- matrix(predict(nuc.fit1,newdata=capdat),50,50)
nuc.pred2 <- matrix(predict(nuc.fit2,newdata=capdat),50,50)
par(mfrow=c(2,1),mar=rep(1,4))
persp(x=capseq,y=datseq,z=nuc.pred1,theta=25,zlim=range(c(nuc.pred1,nuc.pred2)),ticktype="detailed",xlab="Capacity",ylab="Permit issue date",zlab="Predicted cost")
persp(x=capseq,y=datseq,z=nuc.pred2,theta=25,zlim=range(c(nuc.pred1,nuc.pred2)),ticktype="detailed",xlab="Capacity",ylab="Permit issue date",zlab="Predicted cost")
```



# As expected based on earlier results, there's barely any visual differences between the two surfaces. Inclusion of the interaction term, as evidenced by extremely large p-values, has no tangible effect on modeling the response.

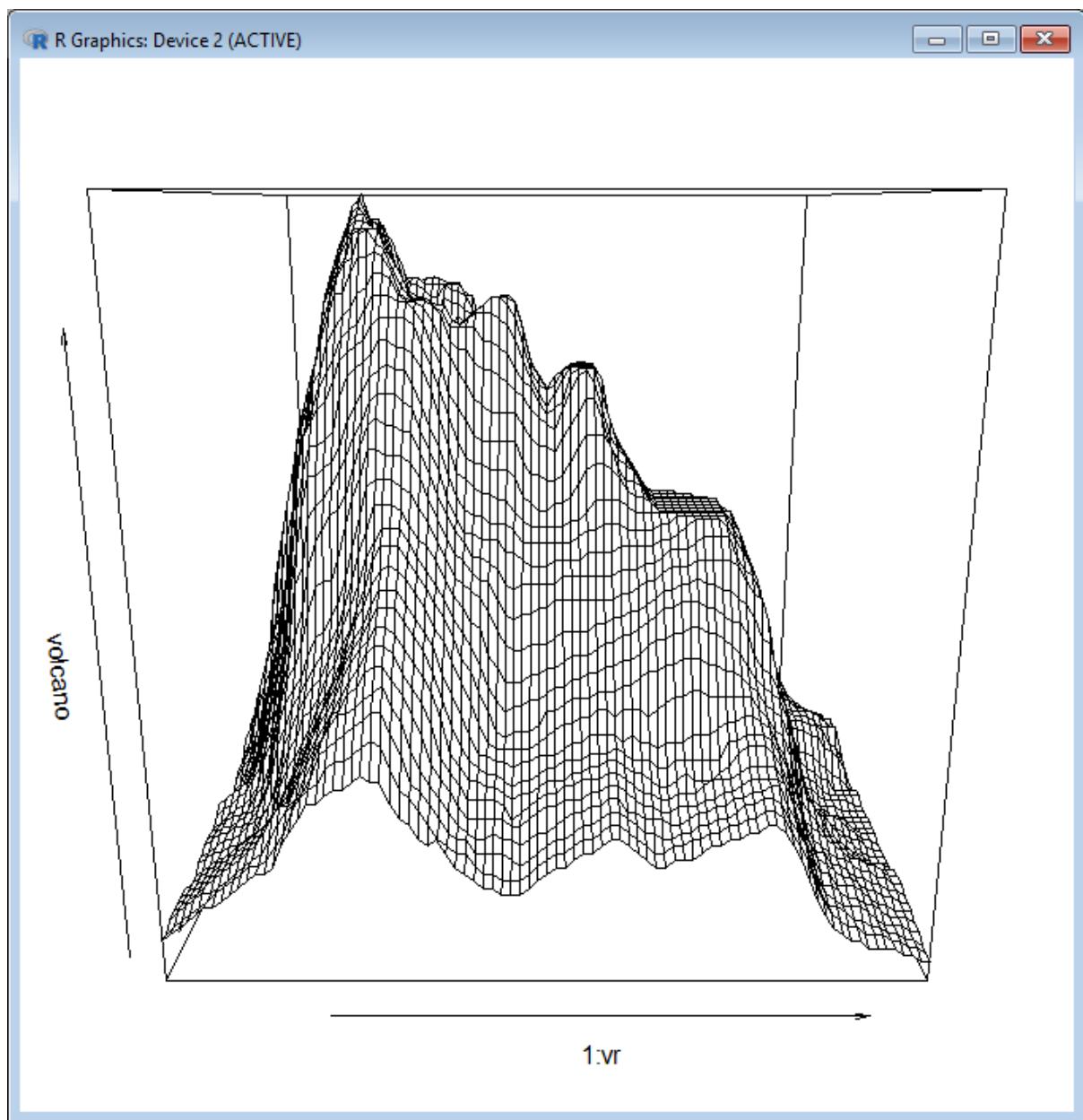
#(b)

```
par(mfrow=c(1,1),mar=rep(1,4)) #(reset to default)
persp(x=capseq,y=datseq,z=nuc.pred2-nuc.pred1,theta=75,ticktype="detailed")
```

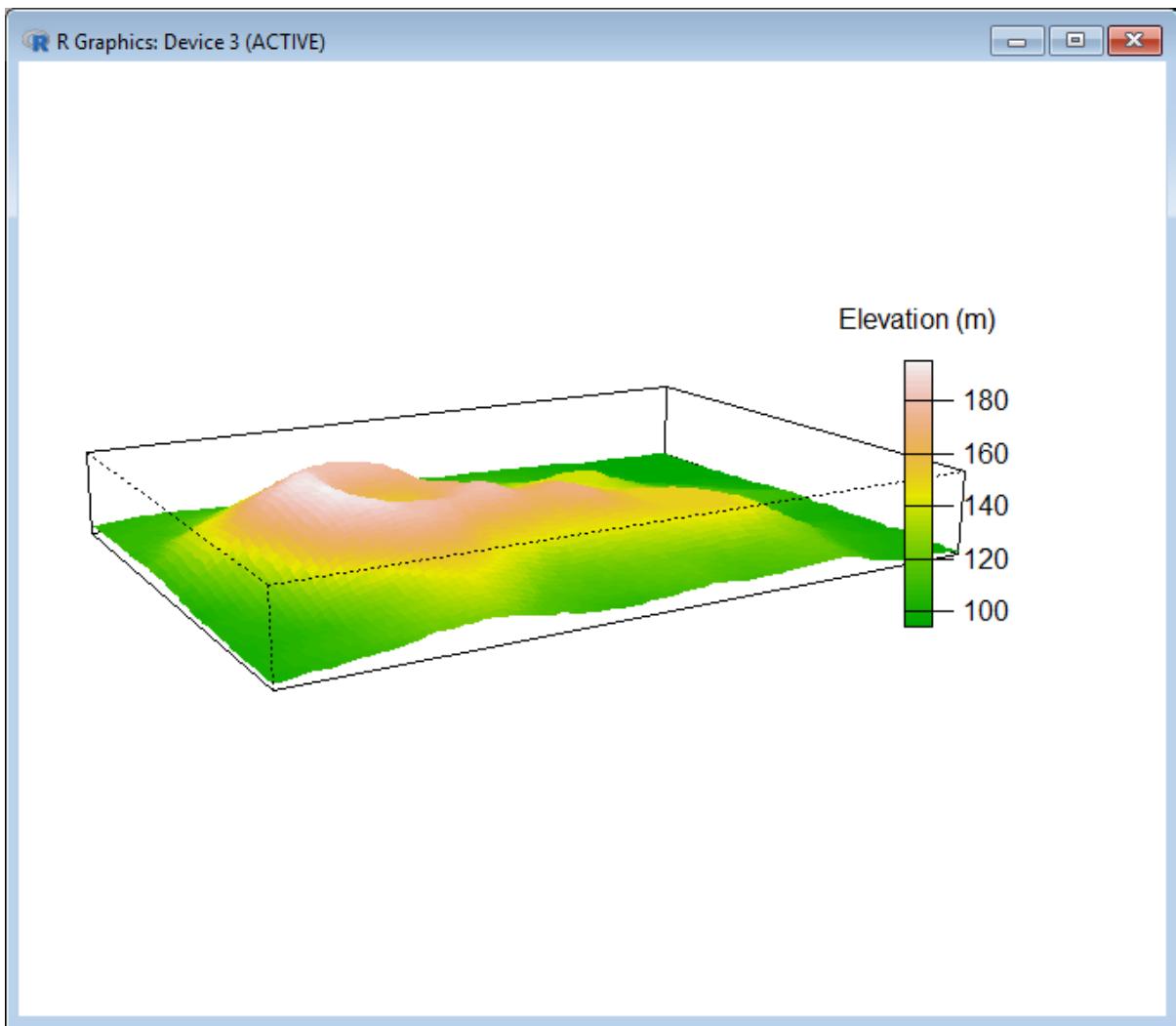


# The main impact of including the interaction term seems to be a discrepancy (when compared with the main-effect-only model) at the later dates (approx. 1970-71) as you increase plant capacity.

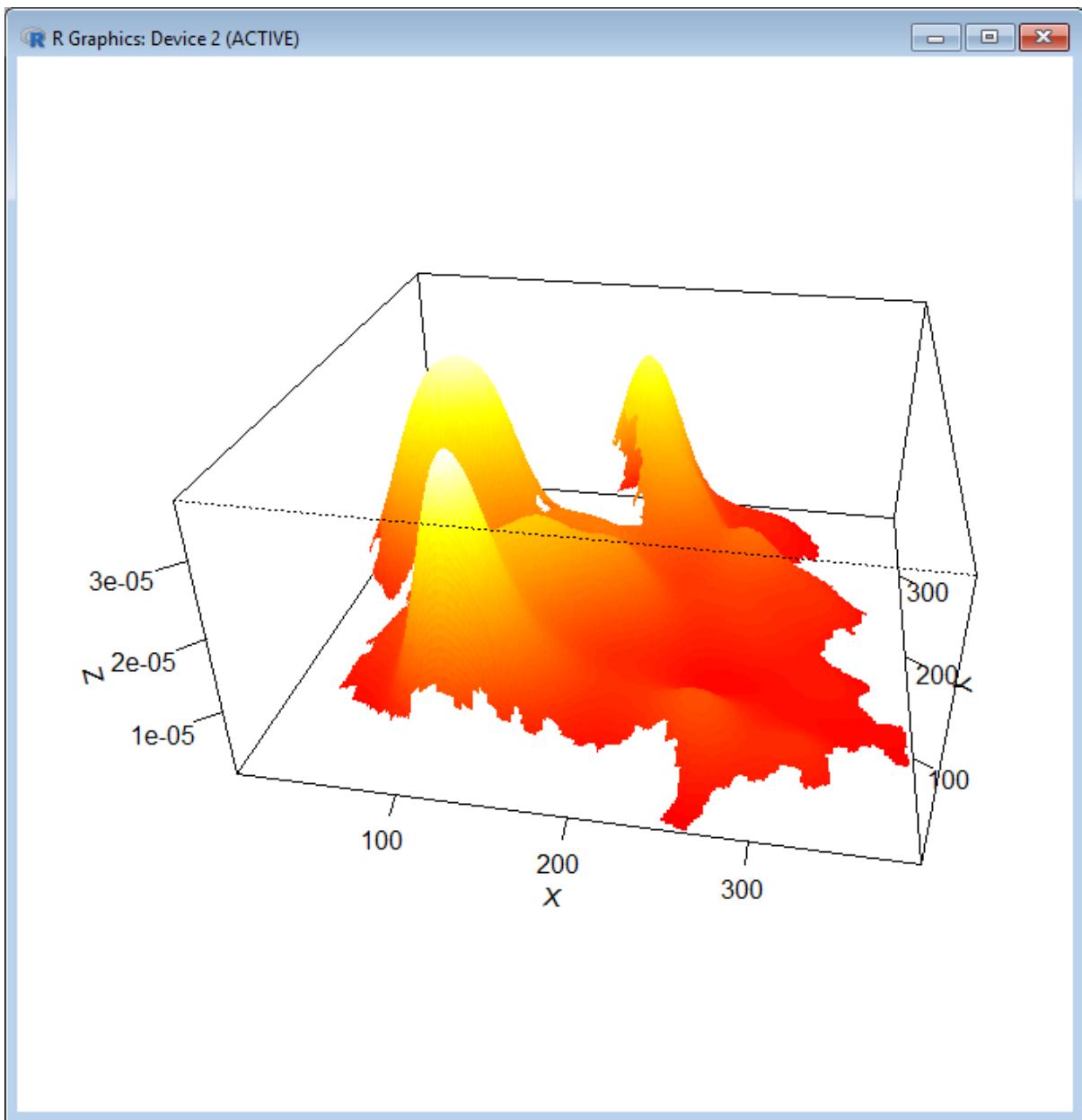
```
#(c)
vr <- nrow(volcano)
vc <- ncol(volcano)
persp(x=1:vr,y=1:vc,z=volcano)
```



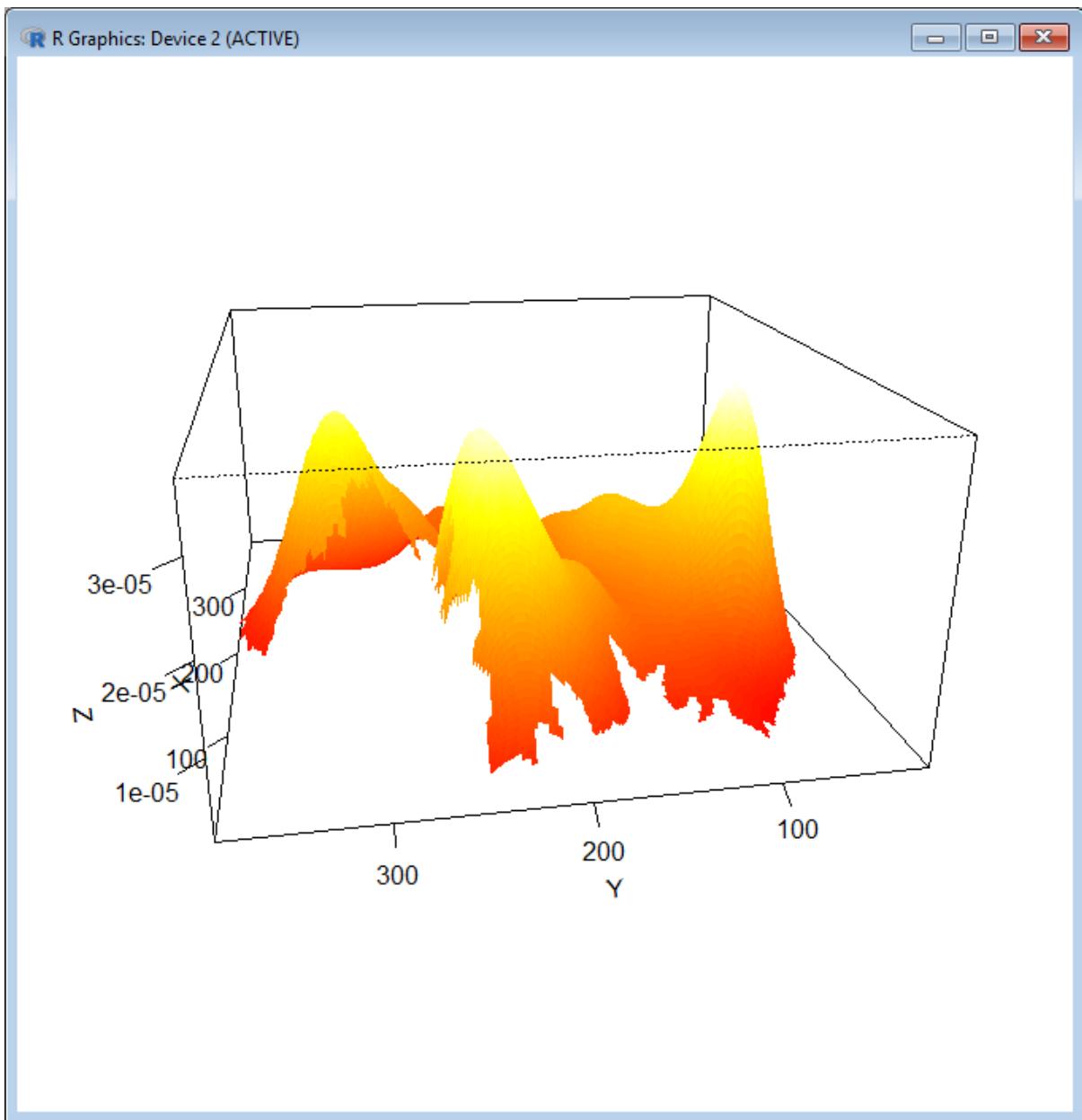
```
#(d)
dev.new()
par(mar=c(1,1,1,5))
vm <- (volcano[-1,-1]+volcano[-1,-vc]+volcano[-vr,-1]+volcano[-vr,-vc])/4
tcols <- terrain.colors(50)
volcol <-
tcols[cut(vm,breaks=seq(min(volcano),max(volcano),length=51),include.lowest=TRUE)]
persp(1:vr,1:vc,volcano,col=volcol,border=NA,theta=-
30,phi=15,scale=FALSE,expand=0.1,axes=FALSE)
library("shape")
colorlegend(tcols,zlim=range(volcano),zval=seq(100,180,20),posx=c(0.83,0.86),posy=c(0.4,0.7),
main="Elevation (m)")
```



```
#(e)
library("spatstat")
library("MASS")
fire <- split(clmfires)$intentional
firewin <- clmfires$window
firewin.xr <- firewin$xrange
firewin.yr <- firewin$yrange
fire.dens <- kde2d(x=fire$x,fire$y,n=256,lims=c(firewin.xr,firewin.yr))
fire.xy <- expand.grid(fire.dens$x,fire.dens$y)
fire.outside <- !inside.owin(x=fire.xy[,1],y=fire.xy[,2],w=firewin)
fire.dens$z[fire.outside] <- NA
fm <- (fire.dens$z[-1,-1]+fire.dens$z[-1,-256]+fire.dens$z[-256,-1]+fire.dens$z[-256,-256])/4
hcols <- heat.colors(50)
firecols <-
hcols[cut(fm,breaks=seq(min(fm,na.rm=TRUE),max(fm,na.rm=TRUE),length=51),include.lowest
=TRUE)]
persp(fire.dens$x,fire.dens$y,fire.dens$z,col=firecols,theta=10,phi=30,border=NA,scale=FALSE,
expand=5e+6,ticktype="detailed",xlab="X",ylab="Y",zlab="Z")
```



```
#(f)
persprot <- function(...,skip=1){
  for(i in seq(90,20,by=-skip)){
    persp(phi=i,theta=0,...)
  }
  for(i in seq(0,360,by=skip)){
    persp(phi=20,theta=i,...)
  }
}
persprot(skip=10,fire.dens$x,fire.dens$y,fire.dens$z,col=firecols,border=NA,scale=FALSE,expand=5e+6,ticktype="detailed",xlab="X",ylab="Y",zlab="Z")
```



# Chapter 26: Interactive 3D Plots

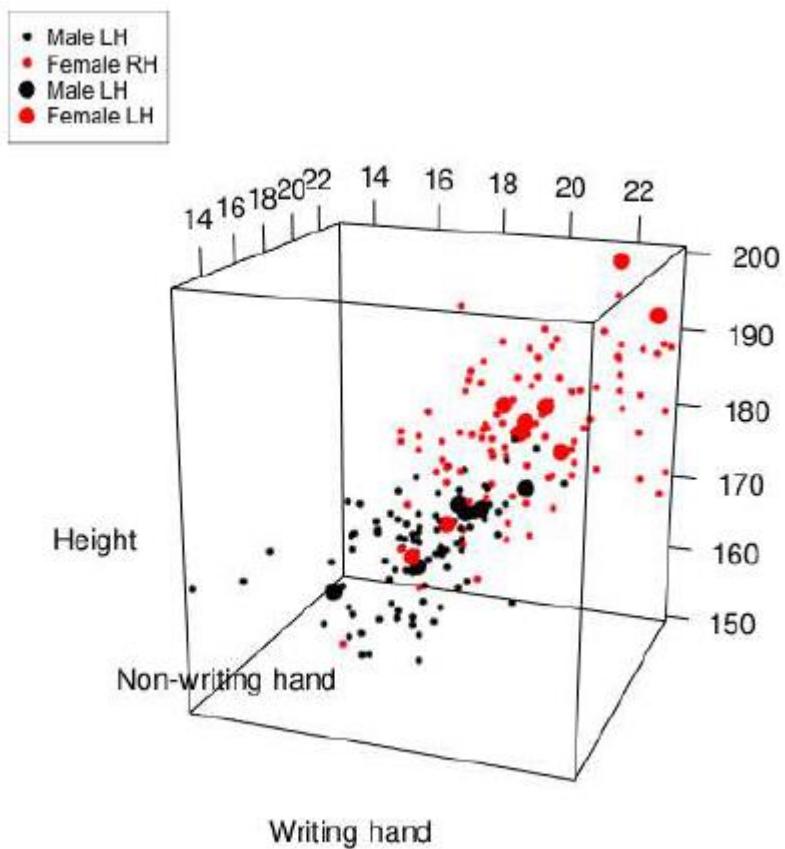
Estimated time to complete: **30 minutes**

## Exercise 26.1

Turn back to the *survey* data frame in the *MASS* package, checking the description of the present variables in the help file `?survey` if you need to. Create a copy of *survey* containing only the writing handspan, nonwriting handspan, left- or right-handedness, sex, and height columns. Then use `na.omit` to remove any rows of this subsetted data frame that contain missing values.

- a. Produce a basic interactive 3D point cloud of student height on the  $z$ -axis, writing handspan on the  $x$ -axis, and nonwriting handspan on the  $y$ -axis.
- b. Create a more informative version of the scatterplot in (a) that uses color to distinguish between sexes and uses point size to distinguish between left- and right-handed individuals, following these guidelines:
  - Start by plotting only those points that correspond to righthanded individuals. Set the color via vector indexing using the numeric version of sex for right-handed individuals—females should be black, males red.
  - Set the plotted point size as 4 for the right-handed individuals and ensure tidy axis labels.
  - Using `points3d`, add the points for left-handed individuals to the existing plot. Colors are to be assigned according to sex in the same way as for the right-handed students, but this time, the point size should be set at 10.
  - Resize the RGL device to your liking and add a legend to the top-left corner that references the four types of points:  
*"Male RH"*, *"Female RH"*, *"Male LH"*, and *"Female LH"*. In setting the legend, use a `pch` value of 19 and use `pt.cex` values of 0.8 and 1.5 for right- and left-handed individuals, respectively.

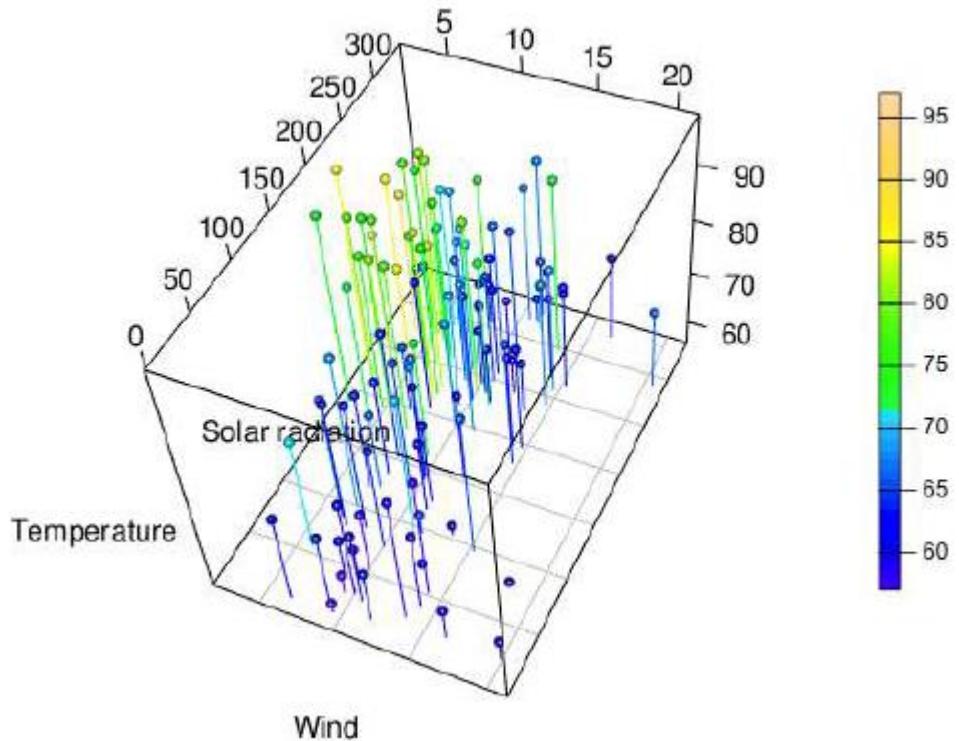
For reference, my version of the rotatable 3D scatterplot is shown here:



In Exercise 25.2 on page 652, you looked at a static 3D scatterplot of the built-in *airquality* data. Again, create a copy of the data frame, omitting any rows with *NA* entries.

- c. Create a similar version of the plot from the earlier exercise using *rgl* functionality, displaying wind speed, solar radiation, and temperature on the *x*-, *y*-, and *z*-axes, respectively, according to the following guidelines:
  - Set up 50 colors from the built-in *topo.colors* palette. Set up the appropriate color index vector for the ozone values, based on the categorization approach.
  - Plot the observations as size 1 spheres, colored as before, and modify the aspect ratio so that the *y*-axis is 1.5 times the length of the other two axes. Provide neat axis titles.
  - Add correspondingly colored lines, one for each observation, stretching vertically upward from the *x*-*y* plane to meet the plotted spheres. Also, place a grid on the lower *x*-*y* plane.
  - Modify the background of the RGL device to include a color legend referencing the ozone level; use a sequence of values between 60 and 95, in steps of 5, to label it.

Here's my result:

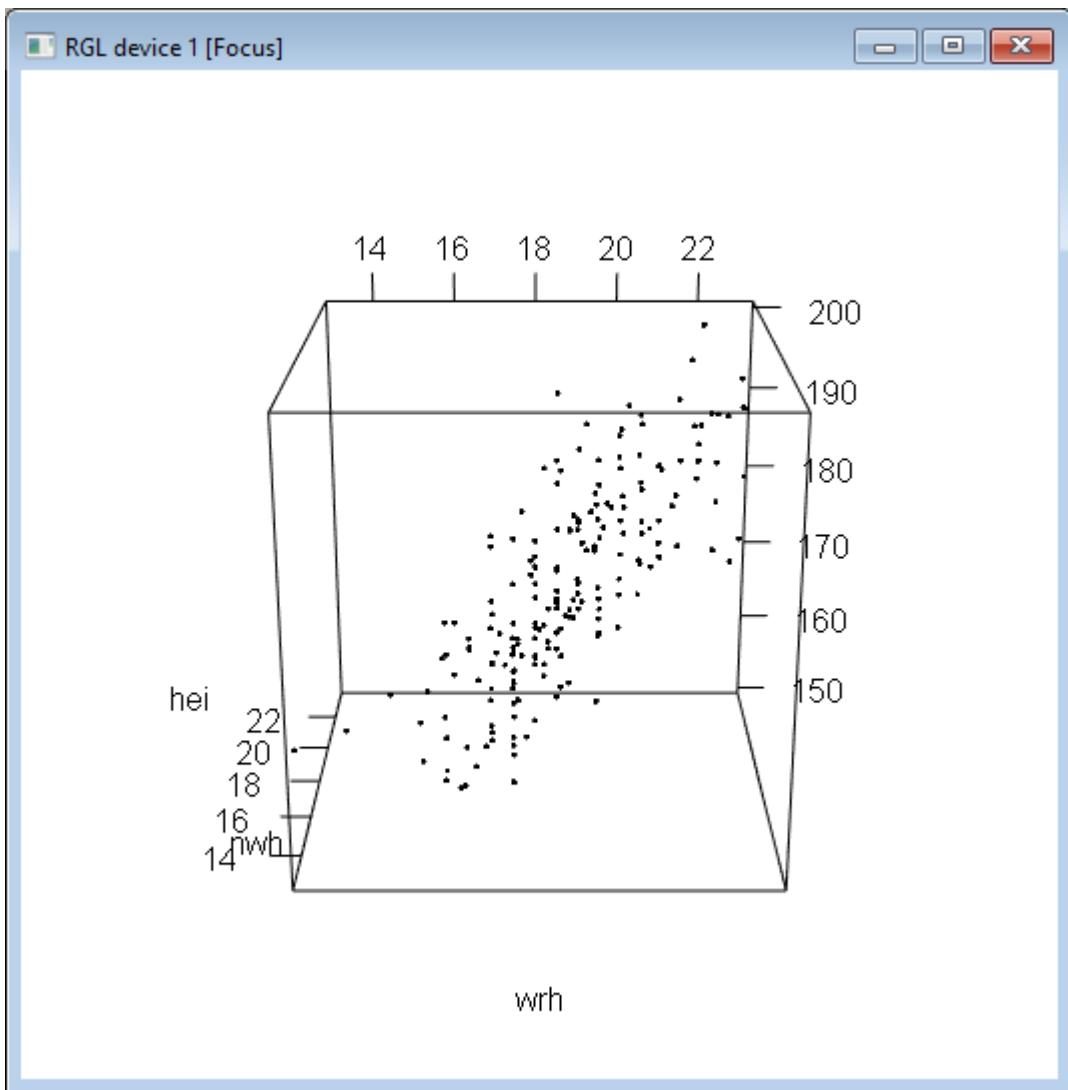


## Solution 26.1

```

library("rgl")
library("MASS")
#(a)
surv <- na.omit(survey[,c("Wr.Hnd","NW.Hnd","W.Hnd","Sex","Height")])
wrh <- surv$Wr.Hnd
nwh <- surv$NW.Hnd
wh <- surv$W.Hnd
hei <- surv$Height
sex <- surv$Sex
plot3d(x=wrh,y=nwh,z=hei)

```

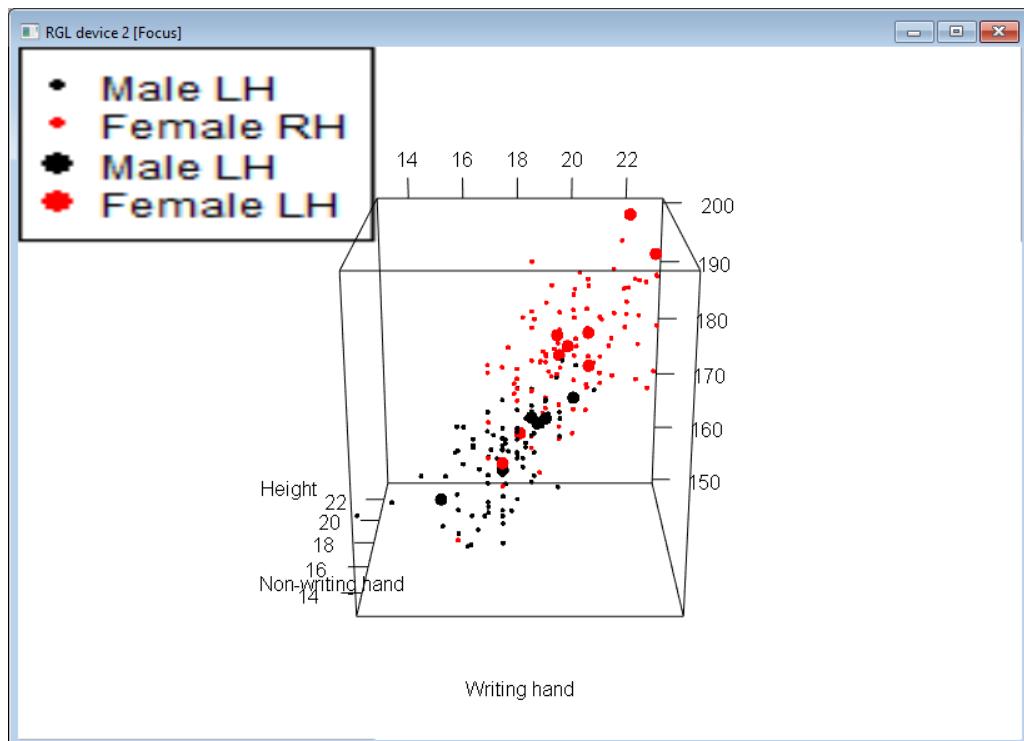


#(b)

```

plot3d(x=wrh[wh=="Right"],y=nwh[wh=="Right"],z=hei[wh=="Right"],col=c("black","red")[as.numeric(sex[wh=="Right"])],size=4,xlab="Writing hand",ylab="Non-writing hand",zlab="Height")
points3d(x=wrh[wh=="Left"],y=nwh[wh=="Left"],z=hei[wh=="Left"],col=c("black","red")[as.numeric(sex[wh=="Left"])],size=10)
legend3d("topleft",legend=c("Male LH","Female RH","Male LH","Female LH"),pch=19,pt.cex=c(0.8,0.8,1.5,1.5),col=c("black","red","black","red"))

```

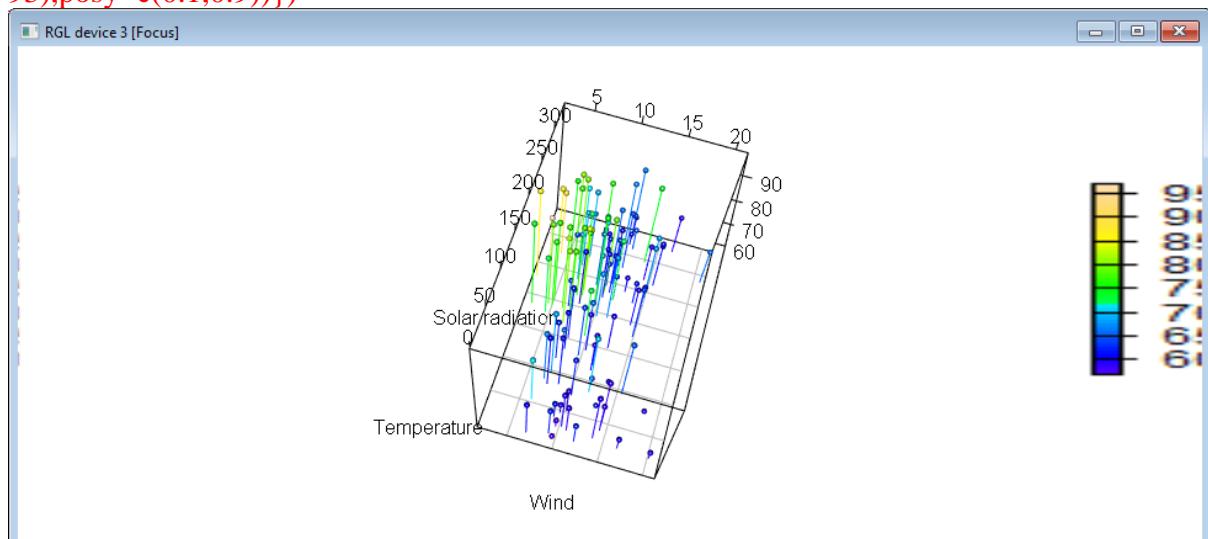


#(c)

```

aq <- na.omit(airquality)
ozcols <- topo.colors(50)
colindex <-
cut(aq$Ozone,breaks=seq(min(aq$Ozone),max(aq$Ozone),length=51),include.lowest=TRUE)
plot3d(aq$Wind,aq$Solar.R,aq$Temp,type="s",col=ozcols[colindex],size=1,aspect=c(1,1.5,1),xla
b="Wind",ylab="Solar radiation",zlab="Temperature")
xfromto <- rep(aq$Wind,each=2)
yfromto <- rep(aq$Solar.R,each=2)
zfromto <- rep(min(aq$Temp),times=2*nrow(aq))
zfromto[seq(2,length(zfromto),2)] <- aq$Temp
segments3d(x=xfromto,y=yfromto,z=zfromto,col=rep(ozcols[colindex],each=2))
grid3d(side="z-")
library("shape")
bgplot3d({plot.new();colorlegend(ozcols,zlim=range(aq$Temp),zval=seq(60,95,5),posx=c(0.91,0.
93),posy=c(0.1,0.9))})

```



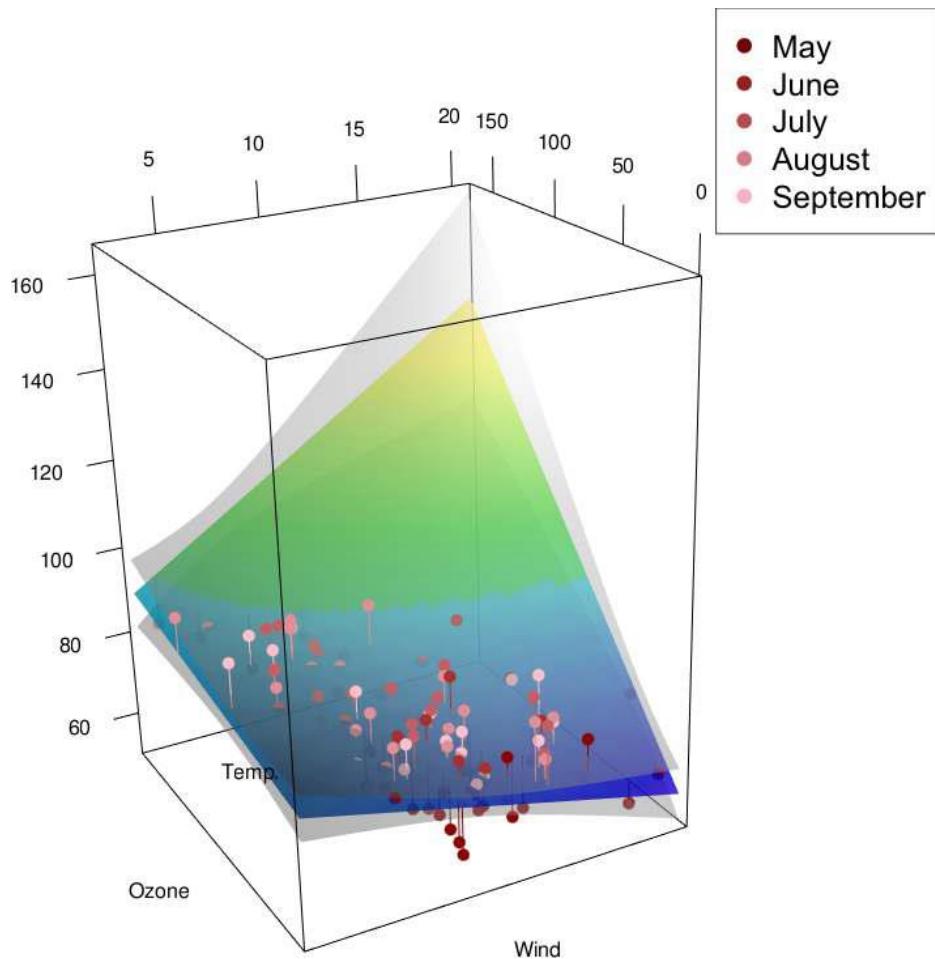
## Exercise 26.2

Return to the measurements in the built-in *airquality* data frame.

Create a copy of the data frame containing the variables pertaining to temperature, wind speed, ozone level, and month; delete all rows with any missing values. You're now going to experiment with an *rgl* visualization of an earlier regression model for mean temperature.

- a. Refit the multiple linear model from Exercise 25.4 on page 676, which regressed temperature against the main effects and an interactive effect of wind speed and ozone. Use *expand.grid* and *predict* to construct a *z*-matrix of the response surface; include the estimation of a 95 percent confidence interval for the fitted mean. Then, use *rgl* functionality to produce an interactive 3D plot of the response surface and color it yellow.
- b. Using the built-in *topo.colors* palette, replot the response surface, assigning the colors according to the *z* value and setting the opacity at 80 percent. Tidy up the axis titles, resize the RGL device, and leave the plot open.
- c. Enhance the plot from (b) as follows:
  - i. Generate exactly five colors from a custom palette that goes from "red4" to "pink" and add the raw wind, ozone, and temperature observations as points to the plot of the response surface. The points should be colored according to month (May through September) using these five colors in order. Set the size of the added points to 10.
  - ii. Add vertical lines that denote the residuals of the fitted model to the plot; in other words, each observation should have a vertical line connecting it to the corresponding fitted value of the response surface. These added lines should use the custom palette from earlier to match the color of each data point.
  - iii. Add the upper and lower 95 percent confidence limits you stored as part of the model prediction in (a). The added surfaces should both be gray with 50 percent opacity.
  - iv. Add a legend to the top-right corner of the interactive plot referencing the five colors of points/lines according to month. Use a *pch* value of 19 and a *cex* value of 2.

The result should look like this:



Next, load the *spatstat* package and revisit the *clmfires* data set. Execute the following lines to restrict attention to only the intentionally lit fires and to obtain the geographical study region:

```
R> fire <- split(clmfires)$intentional
```

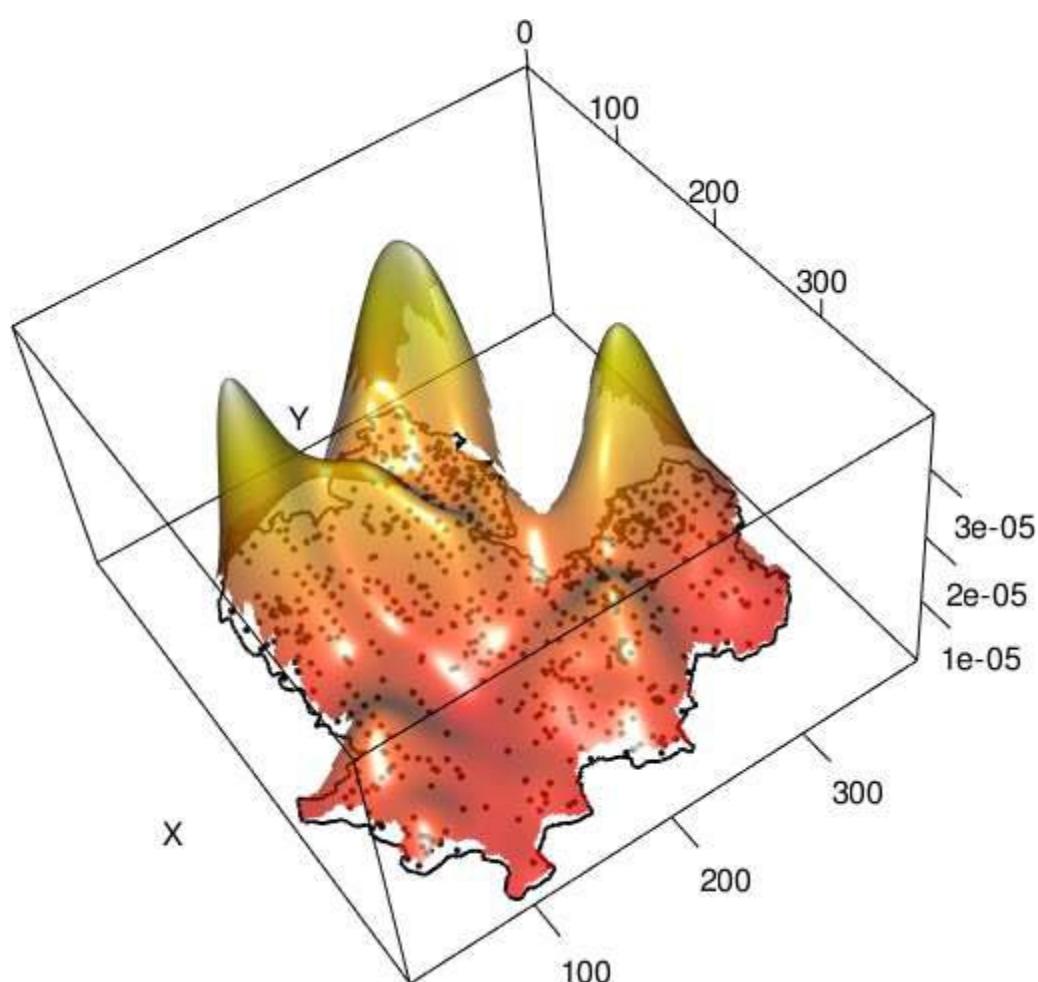
```
R> firewin <- clmfires$window
```

- d. Reproduce the static perspective plot from Exercise 25.5 (e) on page 689 as an interactive perspective plot, based on the following guidelines. Then keep the plot open.
  - Calculate the KDE surface of the *x*- and *y*-coordinates of *fire*, truncated to the study region in *firewin*, using a  $256 \times 256$  evaluation grid.
  - Use the built-in color palette *heat.colors* to color the surface according to the *z* value. Set the opacity to 70 percent.
  - Ensure the *x*-*y* axes have the correct ratio. Then reduce the vertical aspect ratio to be 0.6 relative to the *y*-axis.
  - Suppress the *z*-axis title but add neat "X" and "Y" titles to the other two axes.
- e. Make the following enhancements to the plot:
  - i. Add the raw observations so they lie underneath the surface itself. To do this, set a constant *z* value for each data point as the minimum (and non-NA) value of the *z*-matrix.

- ii. You can obtain the vectors of the  $x$ - and  $y$ -coordinates of the irregular polygon that forms the study region by using the *vertices* function of *spatstat* as follows:

```
R> firepoly <- vertices(firewin)
R> fwx <- firepoly$x
R> fwy <- firepoly$y
```

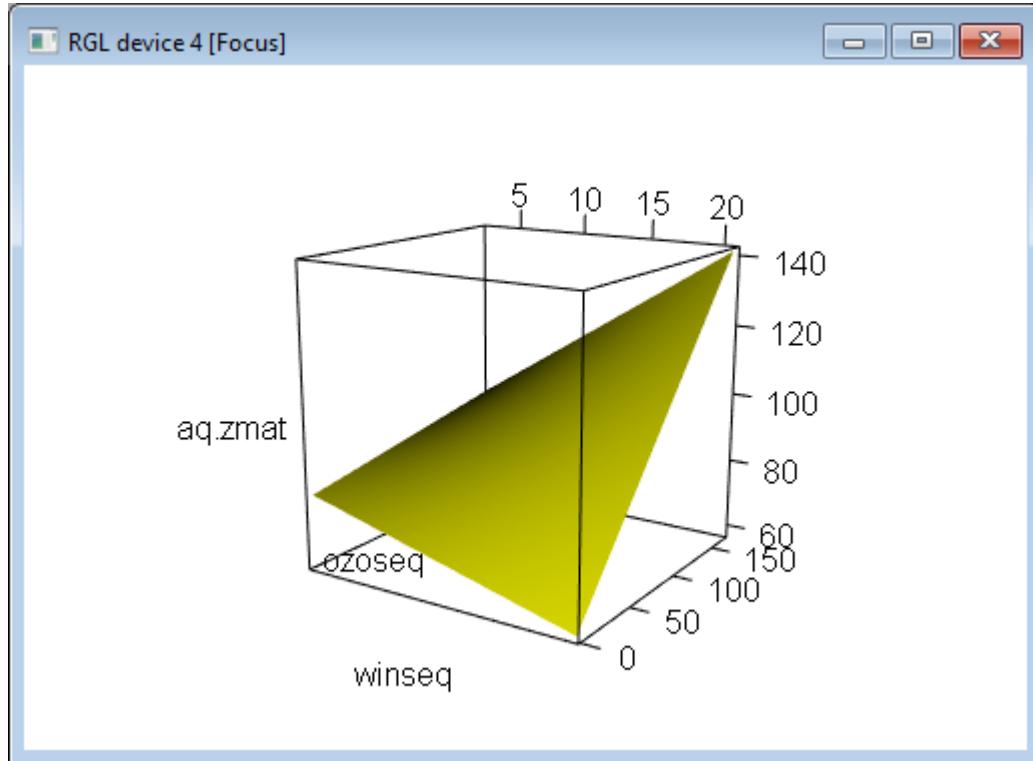
By supplying these two vectors to the appropriate  $x$  and  $y$  arguments of the *lines3d* function, add the study region to surround the superimposed observations lying flush to the  $x$ - $y$  plane underneath the plotted surface. Again, you'll need to specify the  $z$  value as the minimum  $z$ -matrix value for all drawn lines. Set *lwd*=2 for a slightly thicker line than the one drawn by default. Your production should look something like this:



## Exercise 26.2

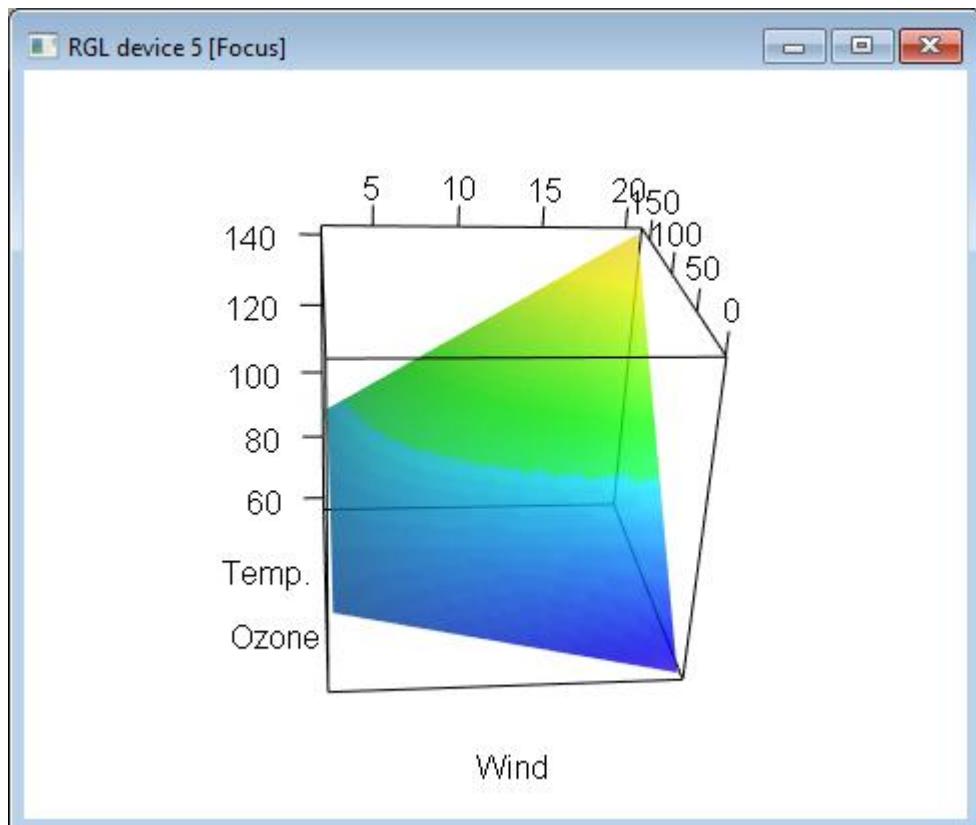
```
library("rgl")
#(a)
aq <- na.omit(airquality[,c("Temp","Wind","Ozone","Month")])
aq.fit <- lm(Temp~Wind*Ozone,aq)
summary(aq.fit)
res <- 50
winseq <- seq(min(aq$Wind),max(aq$Wind),length=res)
ozoseq <- seq(min(aq$Ozone),max(aq$Ozone),length=res)
```

```
winozo <- expand.grid(Wind=winseq,Ozone=ozoseq)
aq.pred <- predict(aq.fit,newdata=winozo,interval="confidence",level=0.95)
aq.zmat <- matrix(aq.pred[,1],res,res)
persp3d(x=winseq,y=ozoseq,z=aq.zmat,col="yellow")
```



#(b)

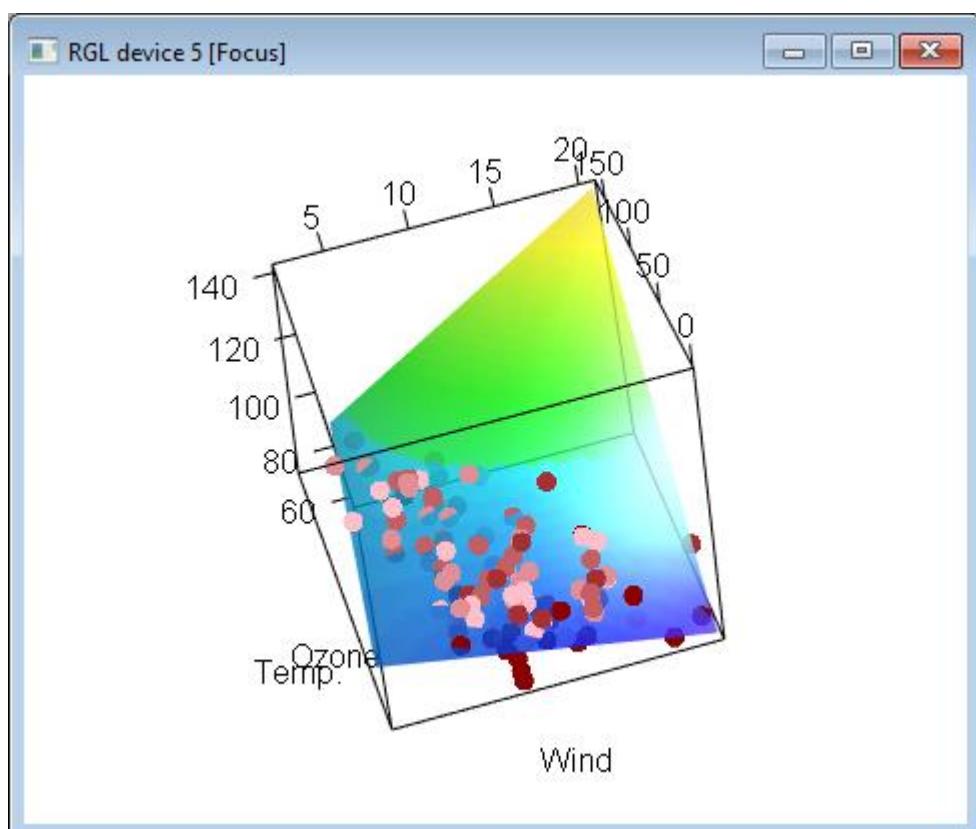
```
tcols <- topo.colors(50)
persp3d(x=winseq,y=ozoseq,z=aq.zmat,col=tcols[cut(aq.pred,breaks=seq(min(aq.pred),max(aq.pred),length=51),include.lowest=TRUE)],alpha=0.8,xlab="Wind",ylab="Ozone",zlab="Temp.")
```



#(c)

##(i)

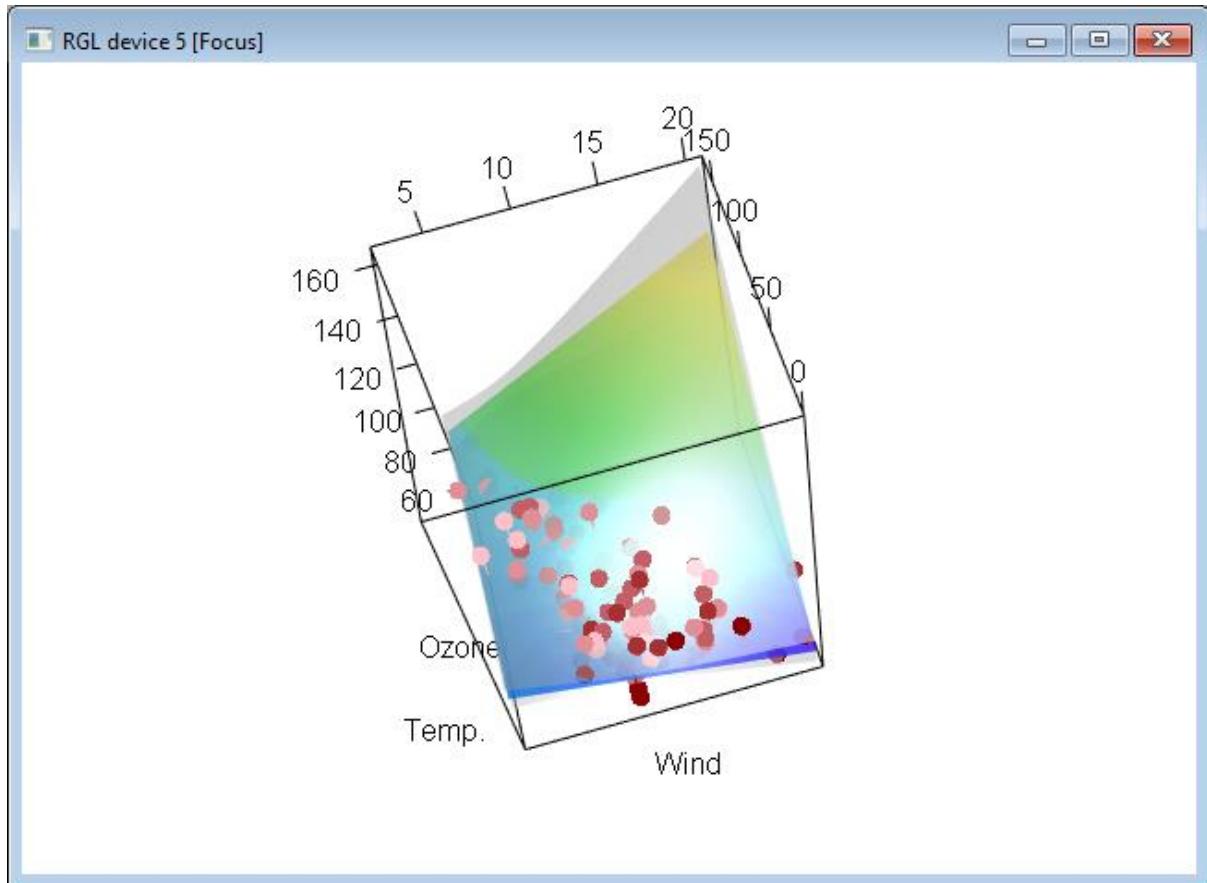
```
reds <- colorRampPalette(c("red4","pink"))
rcols <- reds(5)
points3d(aq$Wind,aq$Ozone,aq$Temp,col=rcols[aq$Month-4],size=10)
```



```

##(ii)
xfromto <- rep(aq$Wind,each=2)
yfromto <- rep(aq$Ozone,each=2)
zfromto <- rep(aq.fit$fitted.values,each=2)
zfromto[seq(2,2*nrow(aq),2)] <- aq$Temp
segments3d(x=xfromto,y=yfromto,z=zfromto,col=rep(rcols[aq$Month-4],each=2))
##(iii)
persp3d(x=winseq,y=ozoseq,z=matrix(aq.pred[,2],res,res),col="gray",alpha=0.5,add=TRUE)
persp3d(x=winseq,y=ozoseq,z=matrix(aq.pred[,3],res,res),col="gray",alpha=0.5,add=TRUE)

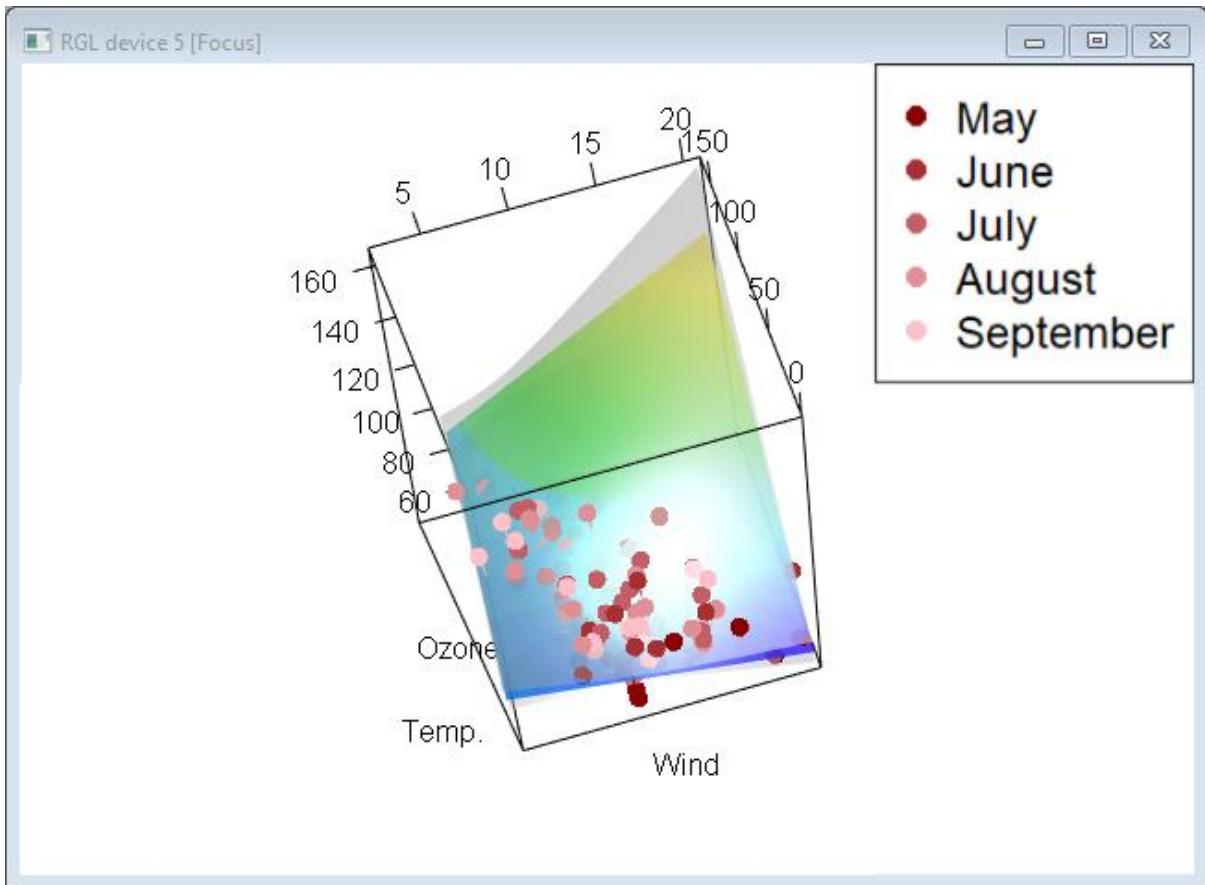
```



```

##(iv)
legend3d("topright",legend=c("May","June","July","August","September"),pch=19,cex=2,col=rcols)

```

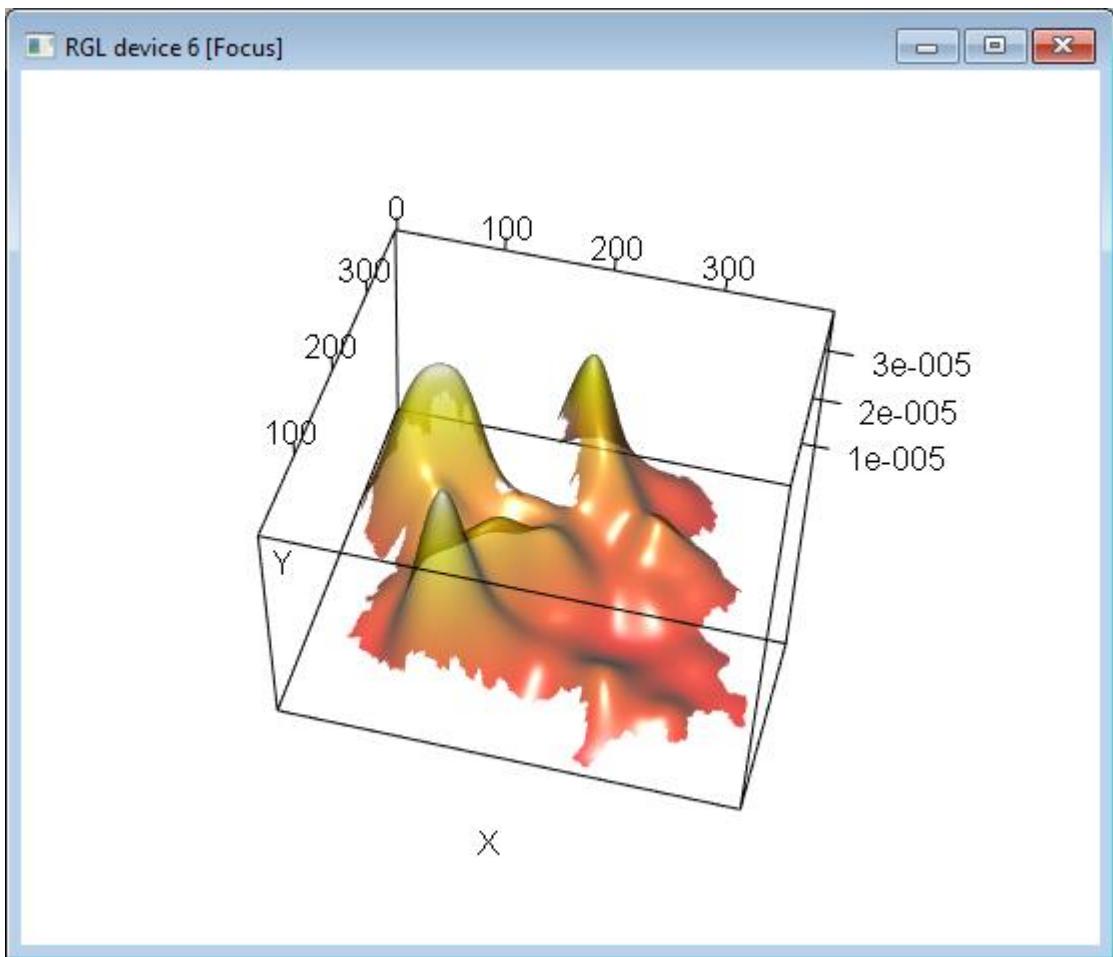


#(d)

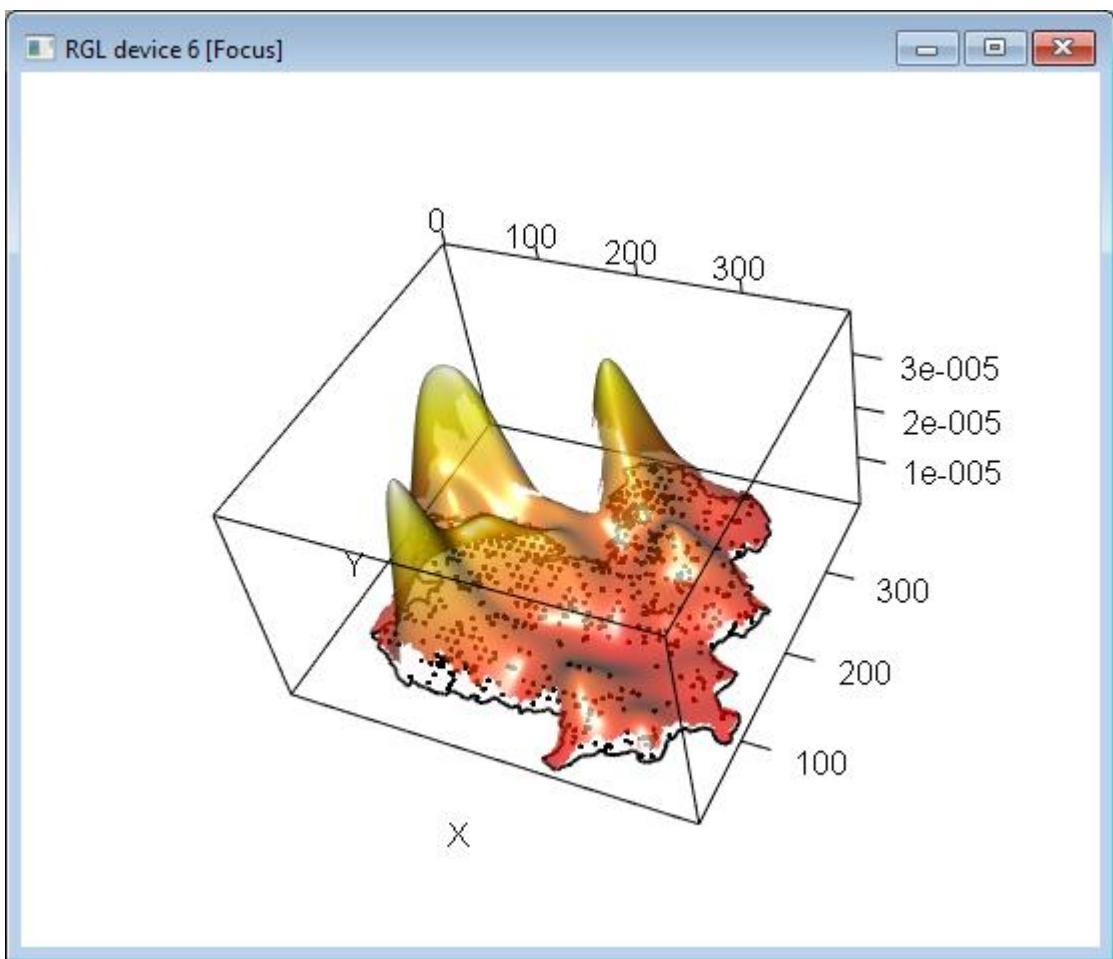
```

library("spatstat")
library("MASS")
fire <- split(clmfires)$intentional
firewin <- clmfires$window
firewin.xr <- firewin$xrange
firewin.yr <- firewin$yrange
fire.dens <- kde2d(x=fire$x,fire$y,n=256,lims=c(firewin.xr,firewin.yr))
fire.xy <- expand.grid(fire.dens$x,fire.dens$y)
fire.outside <- !inside.owin(x=fire.xy[,1],y=fire.xy[,2],w=firewin)
fire.dens$z[fire.outside] <- NA
hcold <- heat.colors(50)
firecols <-
hcold[cut(fire.dens$z,breaks=seq(min(fire.dens$z,na.rm=TRUE),max(fire.dens$z,na.rm=TRUE),length=51),include.lowest=TRUE)]
xr <- firewin$xrange[2]-firewin$xrange[1]
yr <- firewin$yrange[2]-firewin$yrange[1]
persp3d(x=fire.dens$x,y=fire.dens$y,z=fire.dens$z,col=firecols,aspect=c(xr/yr,1,0.6),alpha=0.7,xl
ab="X",ylab="Y",zlab="")

```



```
#(e)
##(i)
points3d(x=fire$x,y=fire$y,z=min(fire.dens$z,na.rm=TRUE))
##(ii)
firepoly <- vertices(firewin)
fwx <- firepoly$x
fwy <- firepoly$y
lines3d(x=fwx,y=fwy,z=min(fire.dens$z,na.rm=TRUE),lwd=2)
```



## Exercise 26.3

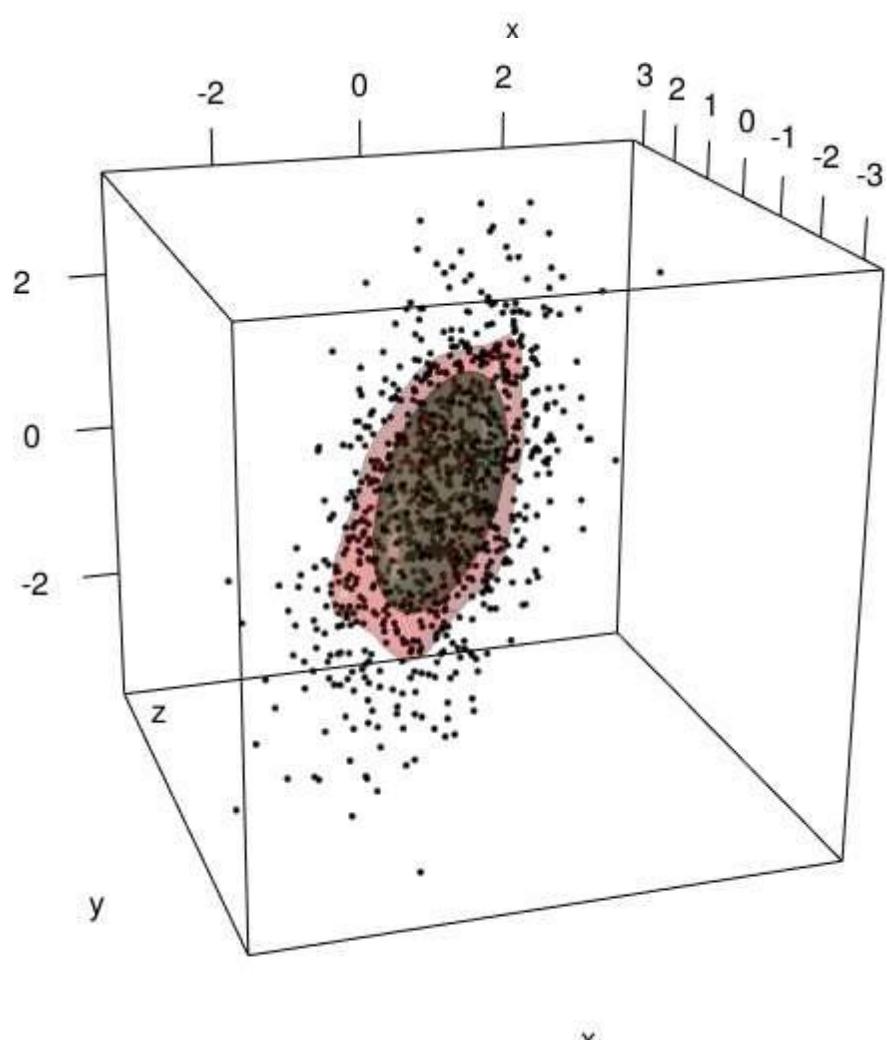
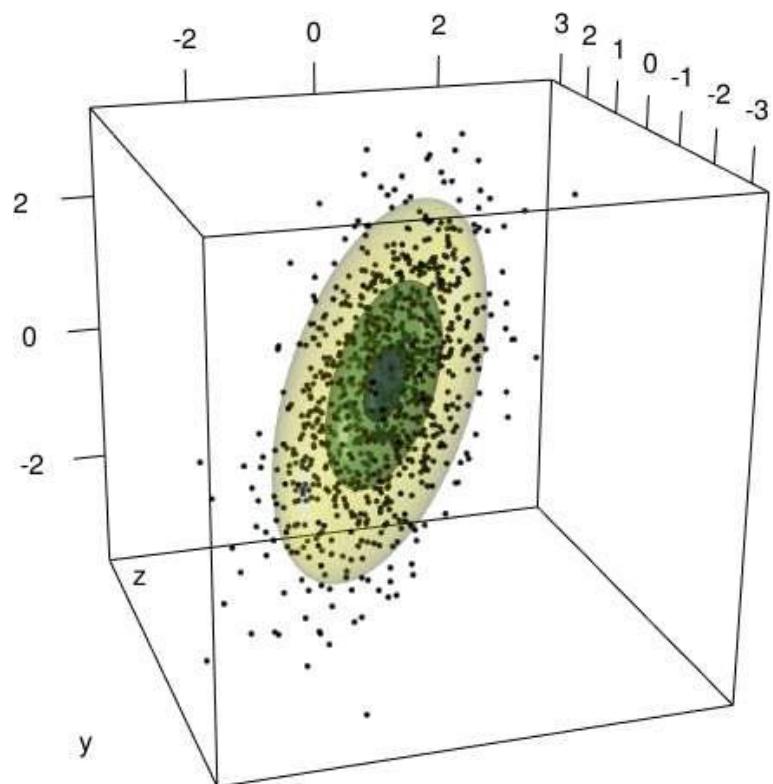
Ensure you have the functionality of the *mvtnorm*, *rgl*, *misc3d*, and *ks* packages available in your current R session. By specifying different *covariance matrices*, you can control how the different components of a multivariate normal random variable relate to one another, which affects the appearance of the distribution itself. In the standard trivariate normal density, for example, the three elements (x, y, z) are independent of one another. Executing the following code will generate 1000 observations from a nonstandard trivariate normal distribution where the three elements are related to one another in a specific way:

```
R> covmat <- matrix(c(1,0.8,0.4,0.8,1,0.6,0.4,0.6,1),3,3)
R> rand3d.norm <- rmvnorm(1000,mean=c(0,0,0),sigma=covmat)
```

Note that the covariance matrix *covmat* is supplied to the optional *sigma* and that the mean of this collection of points remains centered at (0,0,0).

- a. Plot the generated data as an interactive 3D point cloud, with simple axis titles "x", "y", and "z". You should see how the points form an elliptical shape, in contrast to the spherical shape in the standard trivariate normal in Figures 26-11 and 26-12. Keep the plot open.
- b. Using a  $50 \times 50 \times 50$  evaluation lattice between -3 and 3 in each of the three axes, calculate this particular trivariate normal density function using *dmvnorm* and store it as an appropriately sized array. Note that you'll also need to set *sigma=covmat* in any use of *dmvnorm*. Calculate the maximum value of the density and use this to superimpose upon the point cloud isosurfaces at three specific  $\alpha$  levels—0.1, 0.5, and 0.9. Color the three isosurfaces "yellow", "seagreen4", and "navyblue" and set them at 20 percent, 40 percent, and 60 percent opacity, respectively.
- c. Now, use *ks* functionality to calculate a 3D kernel estimate of the density based on the 1000 generated observations. Ensure that the returned object contains the vector of 99 sensible contour levels. Replot the point cloud of (a) in a new RGL device and then make two separate calls to *contour3d* as follows.
  - i. The first should superimpose the theoretical contour at an  $\alpha$  level of 0.5 from (b) only. Use the same color and opacity as you did in (b).
  - ii. The second should draw the isosurface at the 50th percentile as estimated from the point-specific KDE surface. Make it red and reduce the opacity to 20 percent.

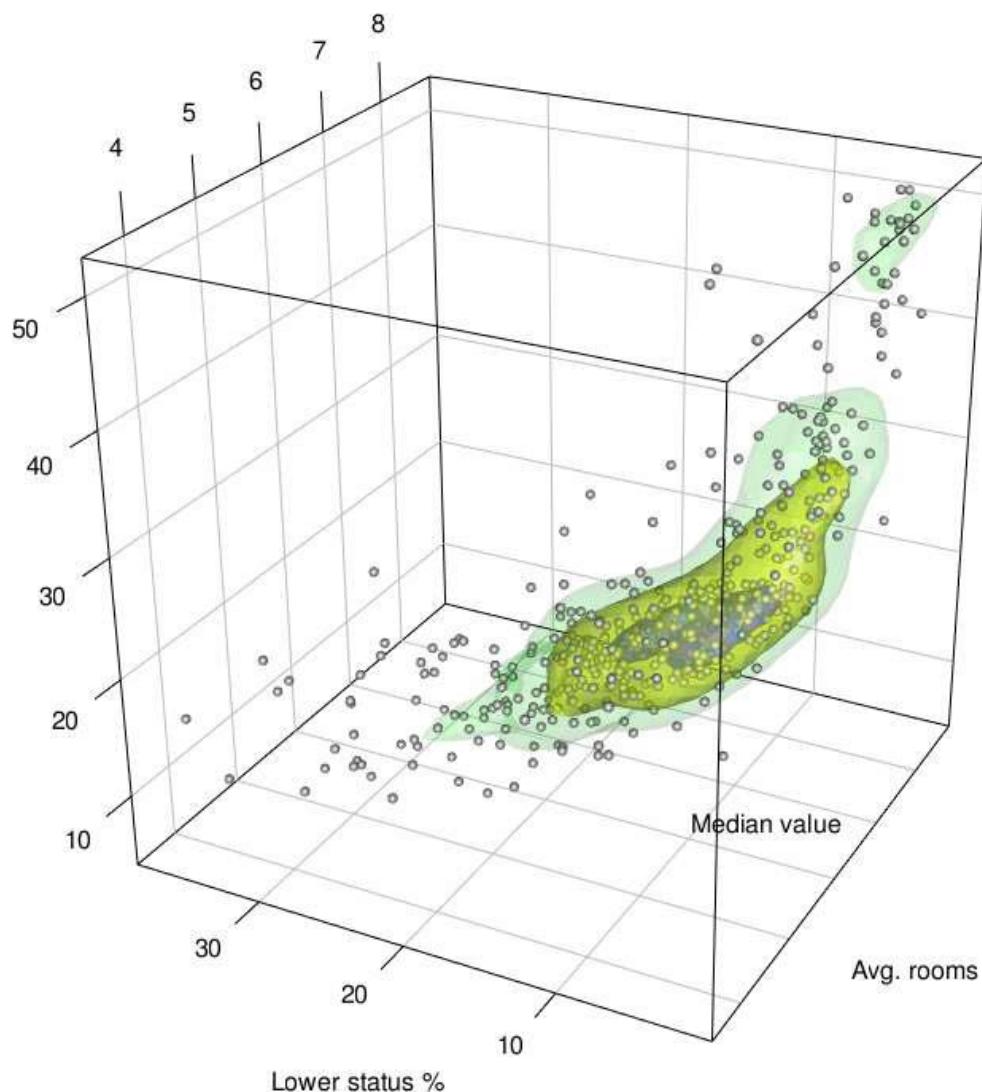
Here are my results from (b) on the left and (c) on the right. Note that the appearance of your KDE isosurface will vary because of the random generation of the 1000 data points, which dictate the final estimate.



The *MASS* package has another data set you've not yet met. The *Boston* data frame object contains a number of descriptive observations concerning house prices in suburbs of Boston, Massachusetts, in the 1970s (Harrison and Rubinfeld, 1978). Load the *MASS* package and inspect the help file *?Boston* to learn about the present variables.

- d. Focus on the variables for average number of rooms, percentage of lower-socioeconomic-status dwellings, and median value—you're going to experiment with the visualization of a 3D scatterplot as follows:
- i. Use *rgl* functionality to plot the three variables, with rooms, status, and value on the *x*-, *y*-, and *z*-axes, respectively; supply tidy axis titles. The data points should be plotted as gray spheres, with a size of 0.5. Keep the plot open.
  - ii. Use *ks* functionality to estimate the trivariate density function of these data. Base the estimate on a  $64 \times 64 \times 64$  evaluation lattice; ensure the 99 integer percentiles of the observation specific density levels are returned. Superimpose isosurface contours delineating the 75 percent, 50 percent, and 10 percent “most dense” observations using green, yellow, and blue. Set the opacity at 10 percent, 40 percent, and 50 percent, respectively.
  - iii. Finally, add reference grids along the three planes identified at the lower *z*-, upper *x*-, and upper *y*-axis locations.

- e. Interpret the final plot from (d). For example, what values of the present variables tend to characterize the most common types of houses in Boston's suburbs?  
 Here's the result:



The *umbilic torus* is another interesting 3D shape in mathematics and can be defined by the following parametric equations:

$$x = \sin(\theta)F(\theta, \varphi)$$

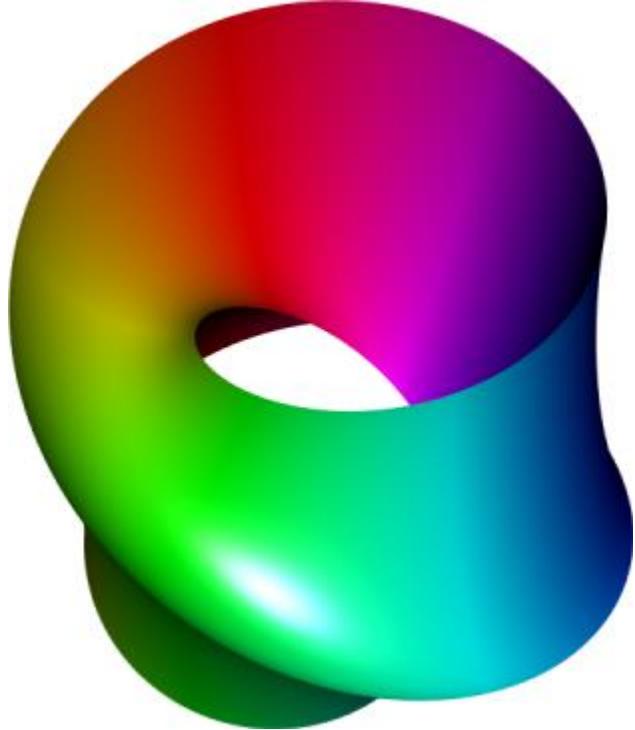
$$y = \cos(\theta)F(\theta, \varphi)$$

$$z = \sin(\theta/3 - 2\varphi) + 2 \sin(\theta/3 + \varphi)$$

In these equations,  $F(\theta, \varphi) = 7 + \cos(\theta/3 - 2\varphi) + 2 \cos(\theta/3 + \varphi)$ , and you allow for both  $-\pi \leq \theta \leq \pi$  and  $-\pi \leq \varphi \leq \pi$ .

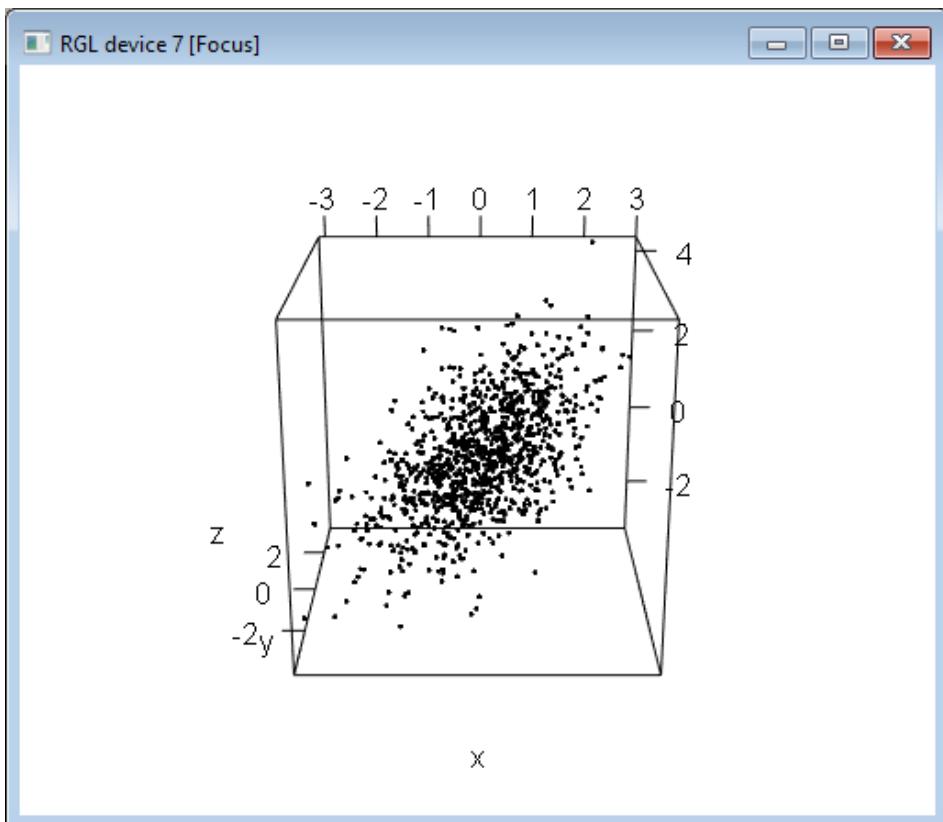
f. Using a sequence of length 1000 for both  $\theta$  and  $\varphi$ , as well as 1000 colors generated from the built-in *rainbow* palette assigned directly to the *col* argument, produce an interactive 3D plot of the umbilic torus. Suppress the box, axes, and axis titles. In viewing the object from different perspectives, note that, much like the Möbius strip you plotted earlier, this shape has only one edge.

Here's the result:



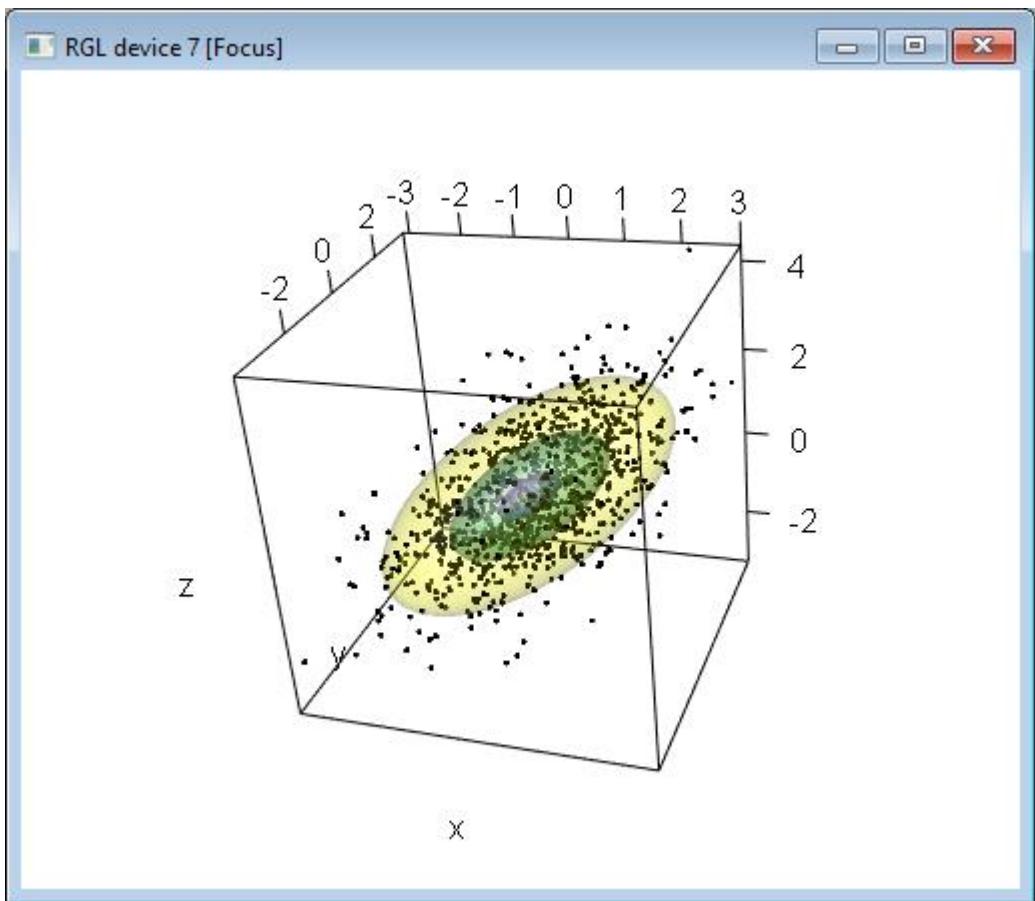
### Solution 26.3

```
library("mvtnorm")
library("rgl")
library("misc3d")
library("ks")
covmat <- matrix(c(1,0.8,0.4,0.8,1,0.6,0.4,0.6,1),3,3)
rand3d.norm <- rmvnorm(1000,mean=c(0,0,0),sigma=covmat)
#(a)
plot3d(rand3d.norm,xlab="x",ylab="y",zlab="z")
```

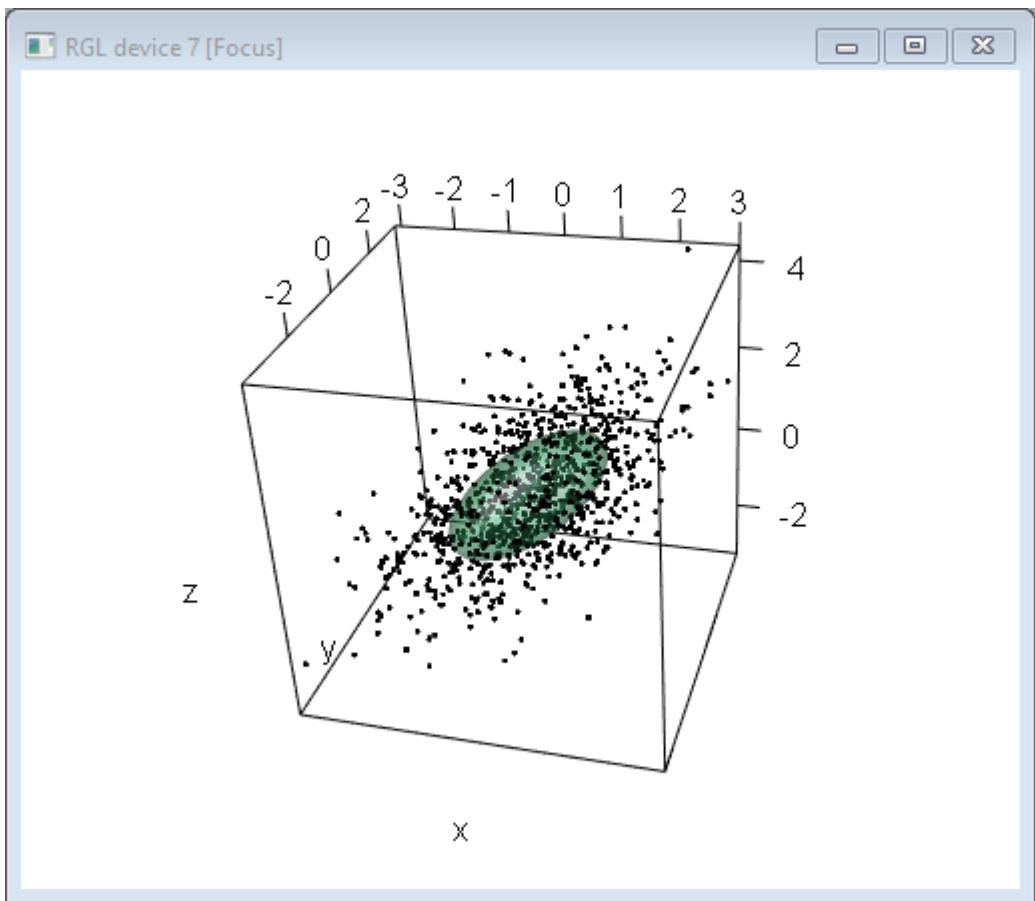


#(b)

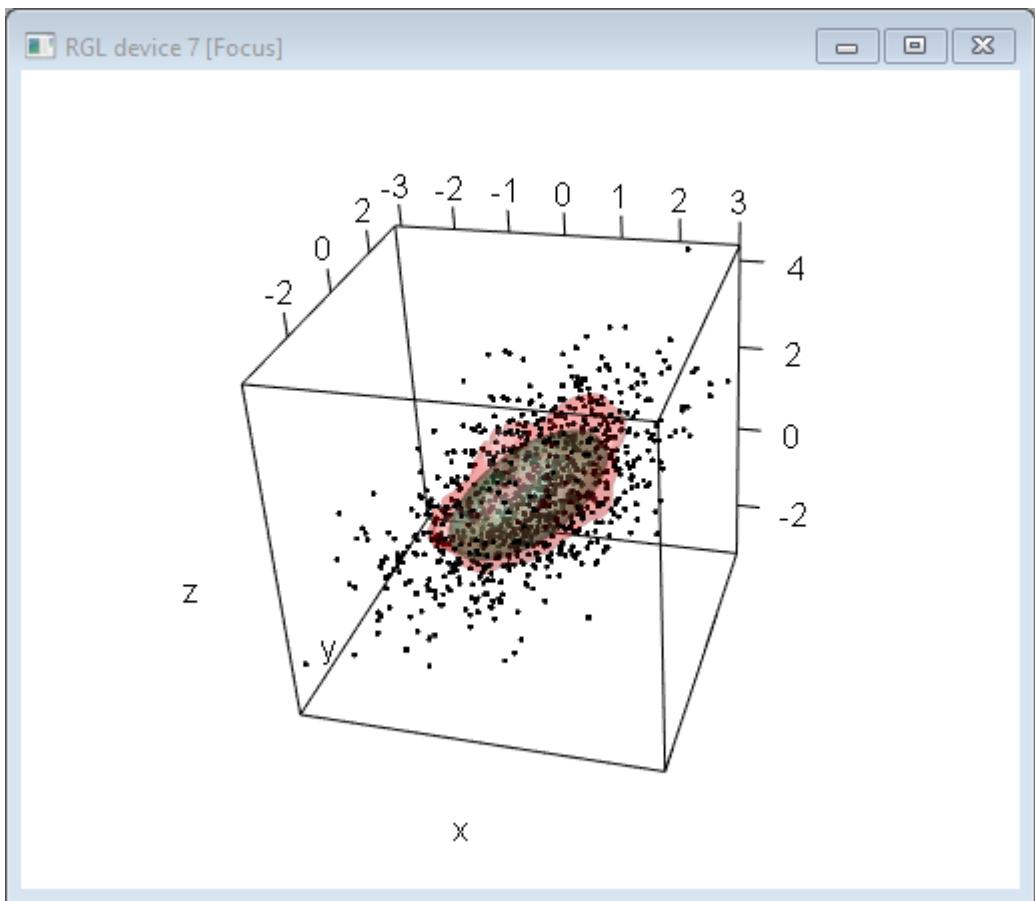
```
vals <- seq(-3,3,length=50)
xyz <- expand.grid(vals,vals,vals)
w <- array(dmvnorm(xyz,mean=c(0,0,0),sigma=covmat),c(50,50,50))
max3d.norm <- dmvnorm(c(0,0,0),mean=c(0,0,0),sigma=covmat)
contour3d(x=vals,y=vals,z=vals,f=w,level=c(0.1,0.5,0.9)*max3d.norm,color=c("yellow","seagreen4","navyblue"),alpha=c(0.2,0.4,0.6),add=TRUE)
```



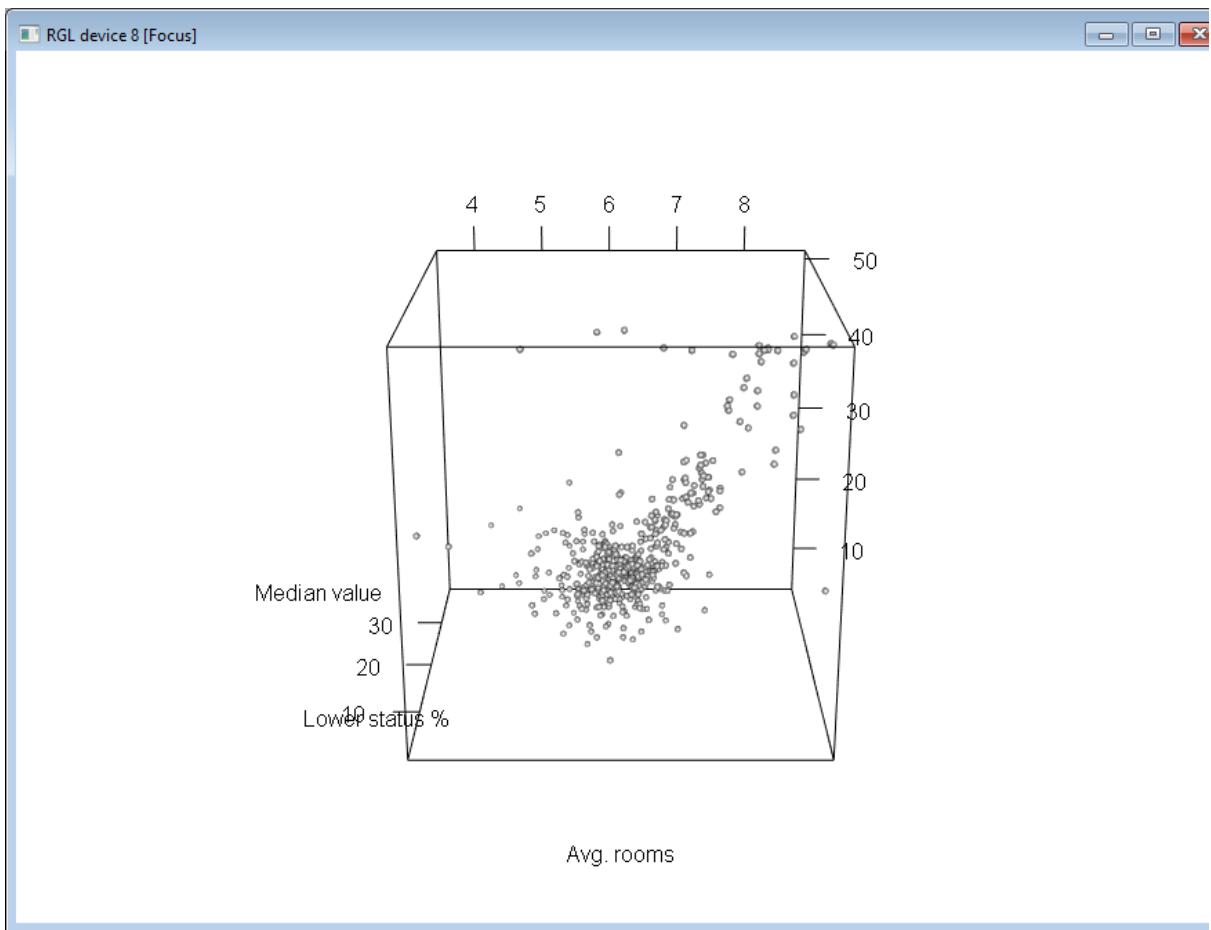
```
#(c)
kde3d.norm <- kde(rand3d.norm,compute.cont=TRUE)
plot3d(rand3d.norm,xlab="x",ylab="y",zlab="z")
##(i)
contour3d(x=vals,y=vals,z=vals,f=w,level=0.5*max3d.norm,color="seagreen4",alpha=0.4,add=TRUE)
```



```
##(ii)  
contour3d(x=kde3d.norm$eval.points[[1]],y=kde3d.norm$eval.points[[2]],z=kde3d.norm$eval.poi  
nts[[3]],f=kde3d.norm$estimate,level=kde3d.norm$cont[50],color="red",alpha=0.2,add=TRUE)
```

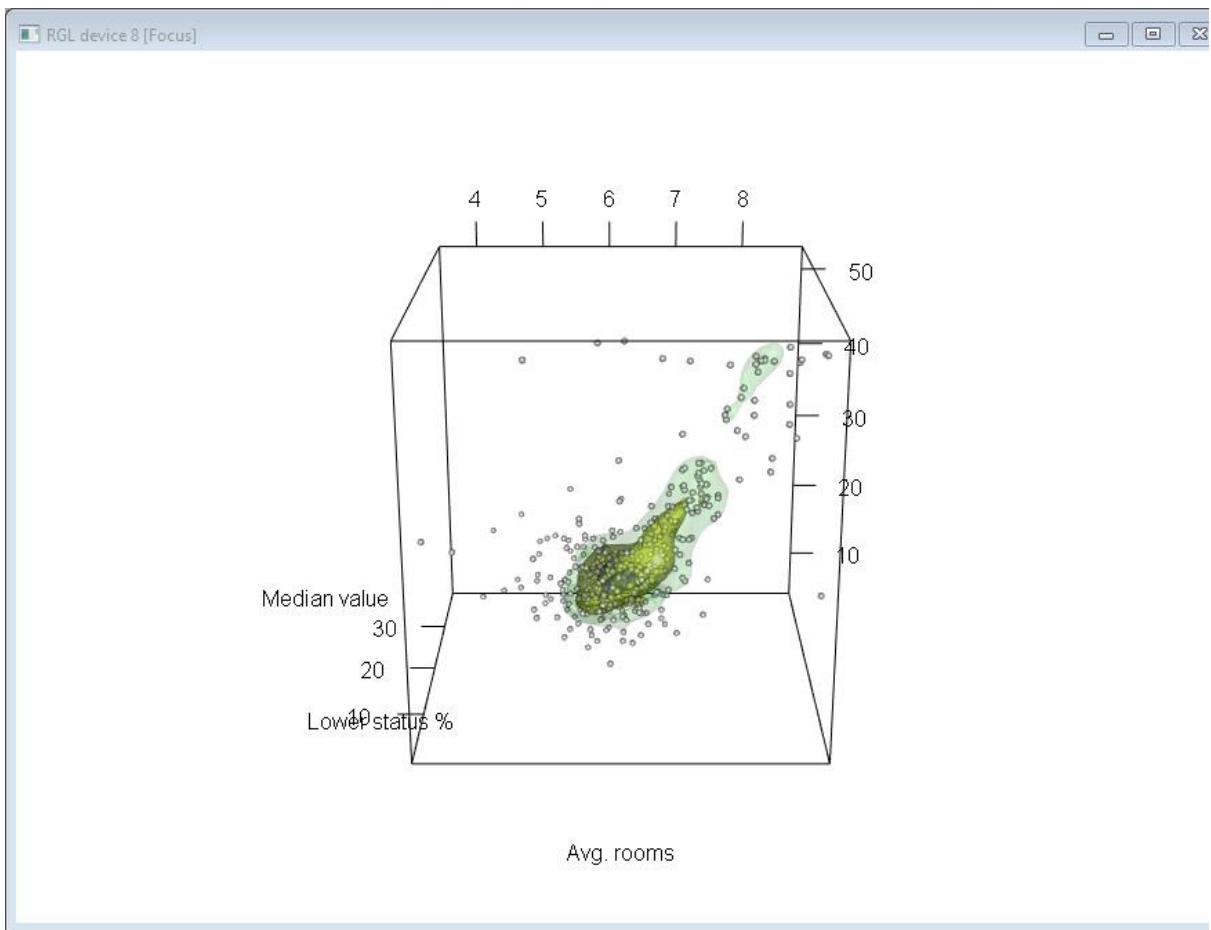


```
#(d)
library("MASS")
?Boston
bos <- Boston[,c("rm","lstat","medv")]
##(i)
plot3d(bos,col="gray",type="s",size=0.5,xlab="Avg. rooms",ylab="Lower status %",zlab="Median value")
```

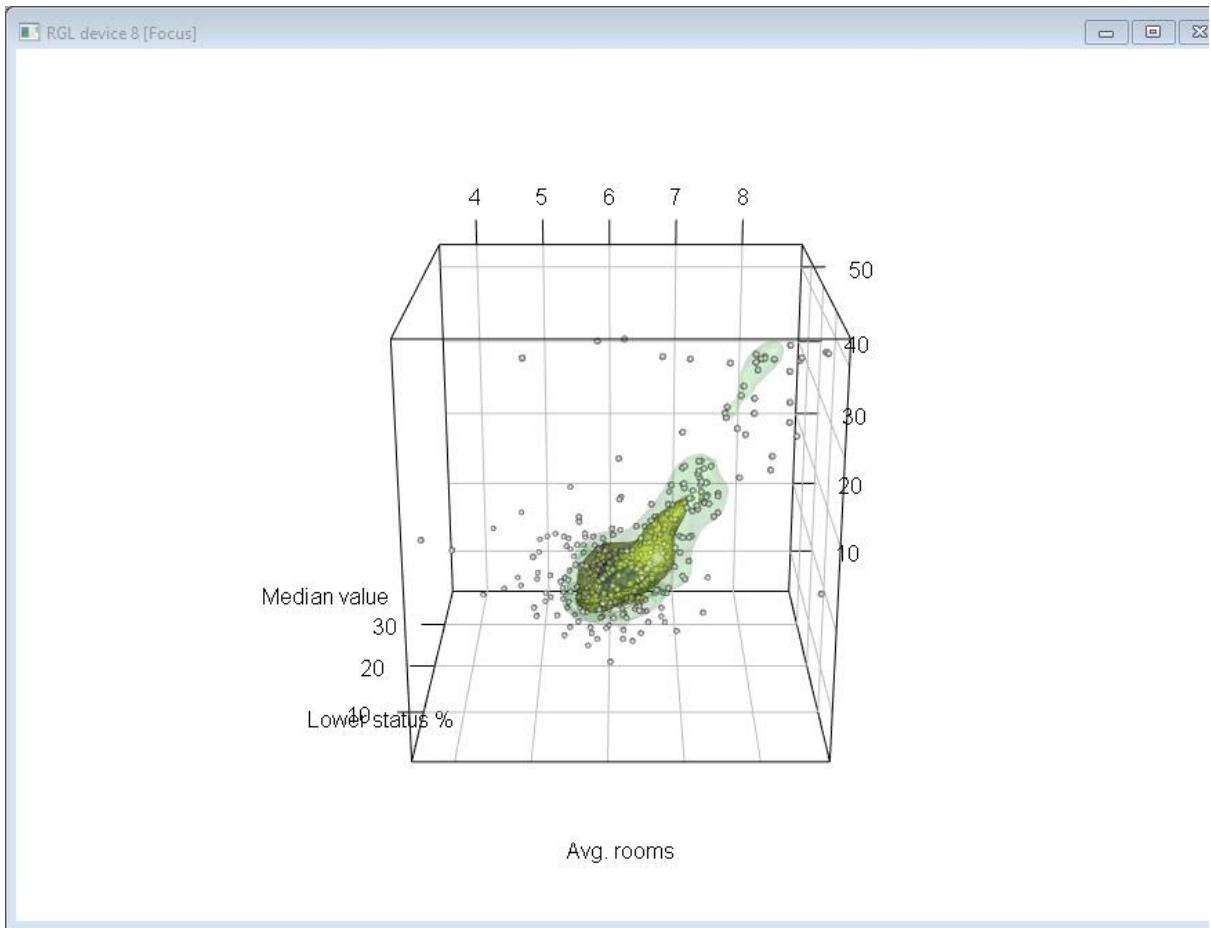


##(ii)

```
bos.dens <- kde(bos,gridsize=c(64,64,64),compute.cont=TRUE)
contour3d(x=bos.dens$eval.points[[1]],y=bos.dens$eval.points[[2]],z=bos.dens$eval.points[[3]],
=bos.dens$estimate,level=bos.dens$cont[c(75,50,10)],color=c("green","yellow","blue"),alpha=c(0
.1,0.4,0.5),add=TRUE)
```



##(iii)  
grid3d(side=c("z-","x+","y+"))



#(e)

# The most common houses tend to have an average of around 6 rooms, are associated with lower-status areas of around 5 to 15%, and have median values of around \$20,000 to \$30,000. There seems to be an additional pocket of observations that represent relatively high-value suburb homes with a large average number of rooms in areas with a very small lower-status percentage.

#(f)

```
vals <- seq(-pi,pi,length=1000)
theta.phi <- expand.grid(vals,vals)
xv <- outer(vals,vals,function(theta,phi) sin(theta)*(7+cos(theta/3-2*phi)+2*cos(theta/3+phi)))
yv <- outer(vals,vals,function(theta,phi) cos(theta)*(7+cos(theta/3-2*phi)+2*cos(theta/3+phi)))
zv <- outer(vals,vals,function(theta,phi) sin(theta/3-2*phi)+2*sin(theta/3+phi))
persp3d(x=xv,y=yv,z=zv,col=rainbow(1000),axes=FALSE,xlab="",ylab="",zlab="")
```

