Manual of Examples to accompany

Devore and Farnum
Applied Statistics for Engineers and Scientists
Second Edition Demonstrations Using...

A. Jonathan R. Godfrey [1]

September 30, 2015

[1]All comments or suggestions regarding this document should be sent to Jonathan Godfrey by emailing <a.j.godfrey@massey.ac.nz>.

Contents

Preface

This manual supports the second edition of Applied Statistics for Engineers and Scientists by Devore and Farnum. It uses the software known as R for as many examples from the text as possible. Many figures are also re-constructed to illustrate the differences between R and the software used by the authors to generate the figures.

Versions of R come and go. This manual was created using version 3.2.2 which was released on 14 August, 2015. All code should work on other versions equally well. Please report any faults with the code to the author.

Jonathan Godfrey <a.j.godfrey@massey.ac.nz>
September 30, 2015

Chapter 1

Data and Distributions

Before you can do any examples in this chapter, you will need to execute the command

```
> library(DevFarn2)
```

This will then give you direct access to the data for each example, and a few extra functions needed for some chapters.

Note that you must issue this command as the first act in any R session.

1.1 Populations, Samples, and Processes

Example 1.1

```
> data(e1.1)
> e1.1
```

```
 [1] 84 49 61 40 83 67 45 66 70 69 80 58 68 60 67 72 73 70 57 63 70 78 52
[24] 67 53 67 75 61 70 81 76 79 75 76 58 31
```

Note R uses the text output for the stem and leaf display and the histogram is built using slightly different boundaries for the groups. 30 to 40 etc, not 25 to 35 as in D&F

Example 1.2

```
[1] 8.141
```

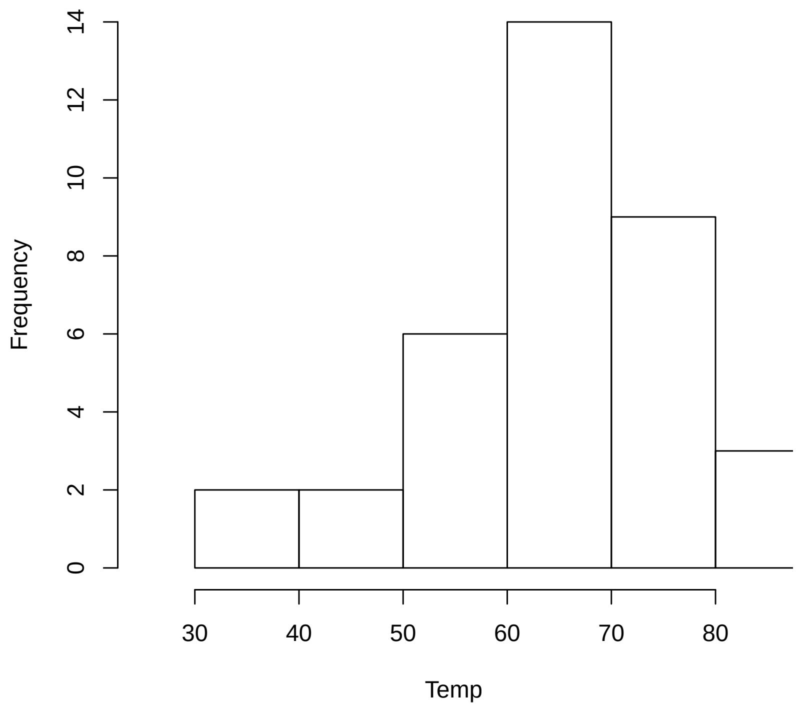
wait for the interval estimate approach in Ch 7.

Example 1.3

no analysis

Figure 1.1 on page 5 of Devore and Farnum.

```
> stem(e1.1)

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

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

> hist(e1.1,  main="",  xlab="Temp",  xlim=c(25, 85))
```



Example 1.4

no analysis

1.2 Visual Displays for Univariate Data

Example 1.5

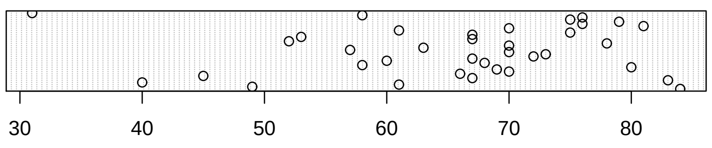
data not available

Example 1.6

Note the data for this example are from Example 1.1 — you may need to do it first.

Figure 1.5 on page 11 of Devore and Farnum.

```
> windows(7, 2.5)
> dotchart(e1.1)
```



Note that R gives added vertical jitter to the spacing of points. The horizontal spacing is correct.

Example 1.7

```
> Games = c(20, 72, 209, 527, 1048, 1457, 1988, 2256, 2403, 2256, 1967, 1509, 1230, 834, 569, 293, 253,
> Hits = rep(0:27, Games)
```

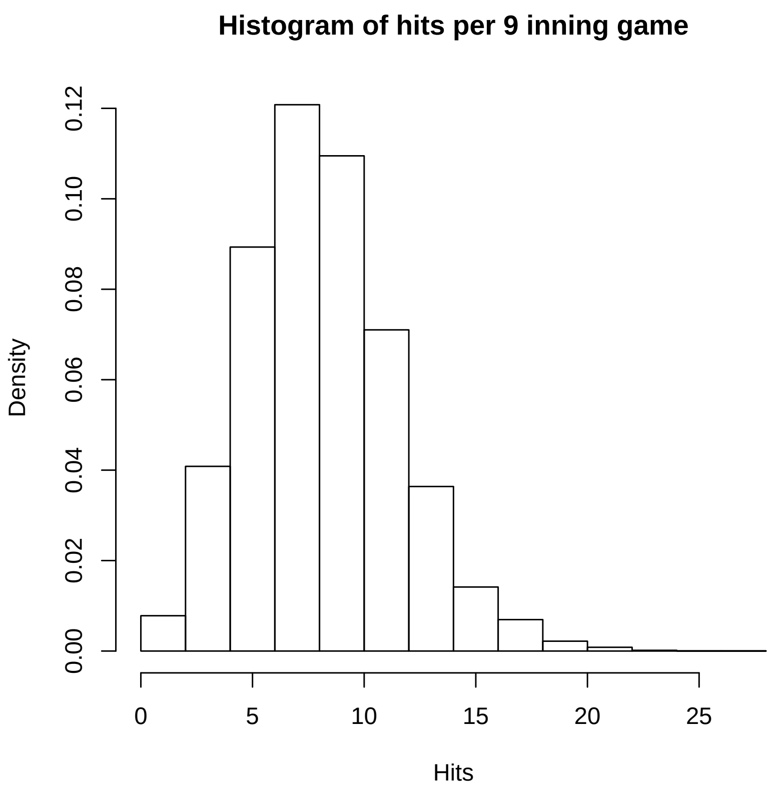
R default's to having what it thinks is a reasonable number of groups (less than D&F). The y-axis label is the default "density"

Example 1.8

```
> data(Furnace)
```

Figure 1.6 on page 13 of Devore and Farnum.

```
> hist(Hits, main="Histogram of hits per 9 inning game", freq=FALSE)
```



Example 1.9

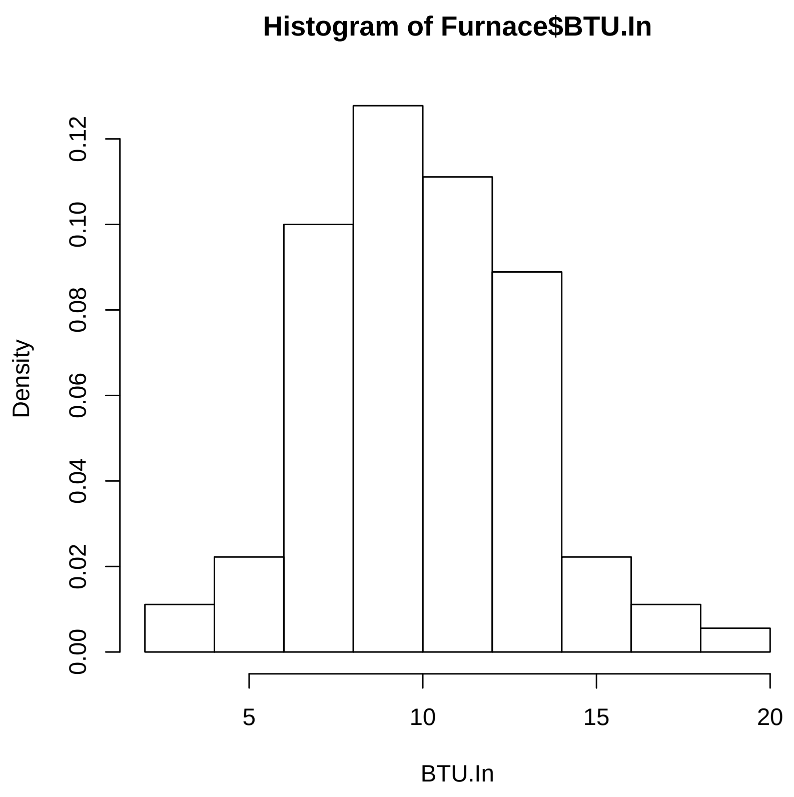
```
> data(e1.9)
```

Example 1.10

```
> data(e1.10)
> e1.10

           Type.of.Defect Frequency Relative.Frequency
1      Low copper plating       112              0.615
2 Poor electrode coverage        35              0.192
3     Lamination problems        10              0.055
4      Plating separation         8              0.044
5        Etching problems         5              0.027
6           Miscellaneous        12              0.066

> Faults = e1.10[,2]
> names(Faults) = e1.10[,1]
> Faults

     Low copper plating Poor electrode coverage     Lamination problems
                    112                      35                      10
     Plating separation        Etching problems           Miscellaneous
                      8                       5                      12
```

Figure 1.7 on page 15 of Devore and Farnum.

```
> hist(Furnace$BTU.In, xlab="BTU.In", breaks=1:10*2, freq=FALSE)
```



To make the Pareto Diagram we will need an additional package. It is called *qcc* and will need to be installed. See the instructions in Appendix **??**.

We then get access to the contents of the package by issuing the command

```
> library(qcc)
```

Note that R doesn't yet manage to get an "other" category last in the chart. Food for thought for programming...

1.3 Describing Distributions

Example 1.11

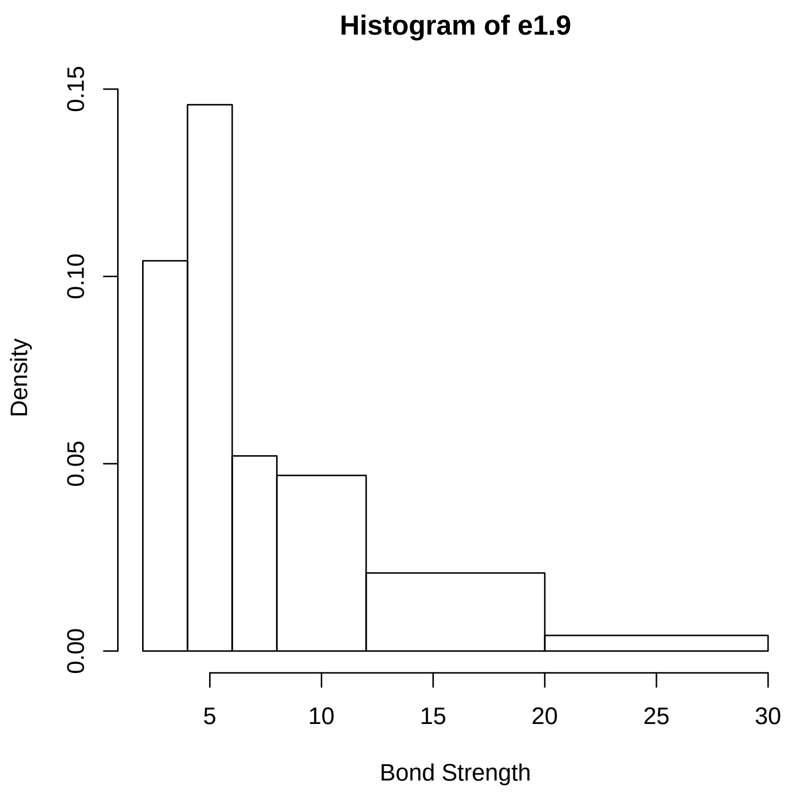
Example 1.12

```
> 1-exp(-0.2*5)
```

```
[1] 0.6321
```

Figure 1.9 on page 17 of Devore and Farnum.

```
> hist(e1.9, xlab="Bond Strength", breaks=c(2,4,6,8,12,20, 30), freq=FALSE)
```



```
> C= -log(0.1)/0.2
> C

[1] 11.51
```

Note the use of a capital C here. The lower case c is used for the name of a function.

Example 1.13

```
> factorial(4)

[1] 24

> factorial(1:8)

[1]     1     2     6    24   120   720  5040 40320
```

We actually use the binomial distribution seen later on in this chapter note choice of the third argument is important

```
> dbinom(x=0:4, size=4, prob=0.9)

[1] 0.0001 0.0036 0.0486 0.2916 0.6561
```

Figure 1.11 on page 19 of Devore and Farnum.

```
> pareto.chart(Faults)

Pareto chart analysis for Faults
                          Frequency Cum.Freq. Percentage Cum.Percent.
  Low copper plating            112       112     61.538        61.54
  Poor electrode coverage        35       147     19.231        80.77
  Miscellaneous                  12       159      6.593        87.36
  Lamination problems            10       169      5.495        92.86
  Plating separation              8       177      4.396        97.25
  Etching problems                5       182      2.747       100.00
```

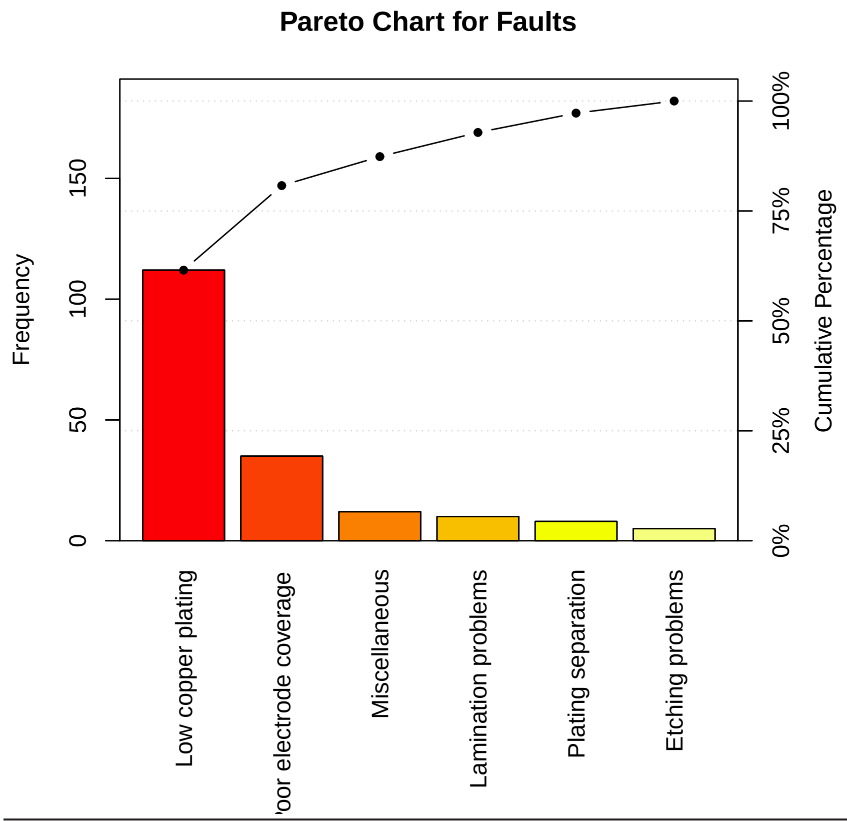
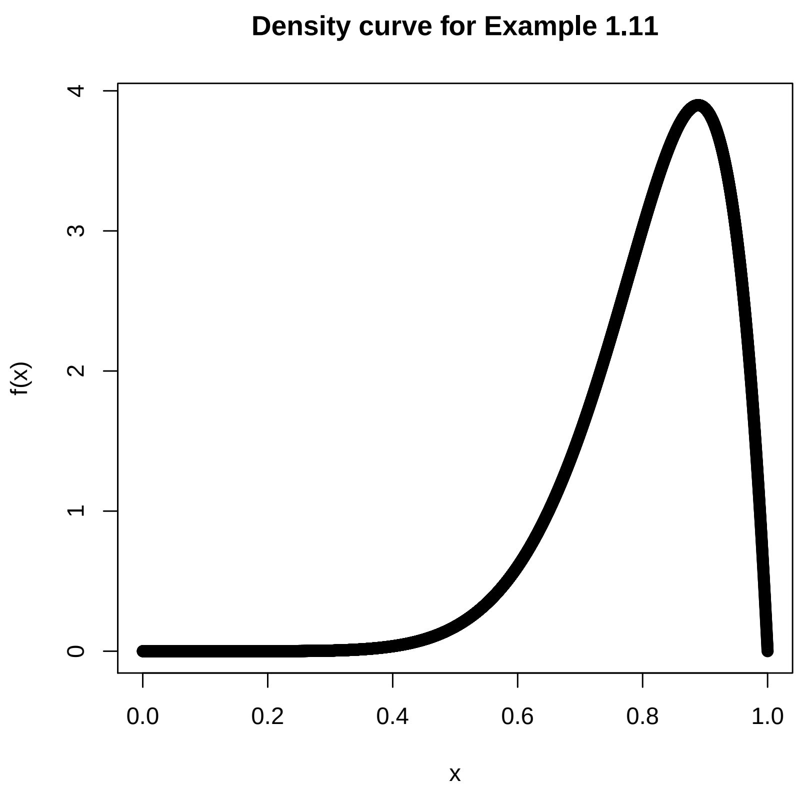


Figure 1.14 on page 26 of Devore and Farnum.

```
> x = 0:10000/10000
> # gives  a set of points from 0 to 1 in steps of 0.0001
> f = 90*x^8*(1-x)
> plot(x,f, ylab="f(x)", main="Density curve for Example 1.11")
```



can be abbreviated if parameters are put in order

```
> dbinom(0:4,4,0.9)
```

```
[1] 0.0001 0.0036 0.0486 0.2916 0.6561
```

and the complement is

```
> dbinom(0:4,4,0.1)
```

```
[1] 0.6561 0.2916 0.0486 0.0036 0.0001
```

1.4 The Normal Distribution

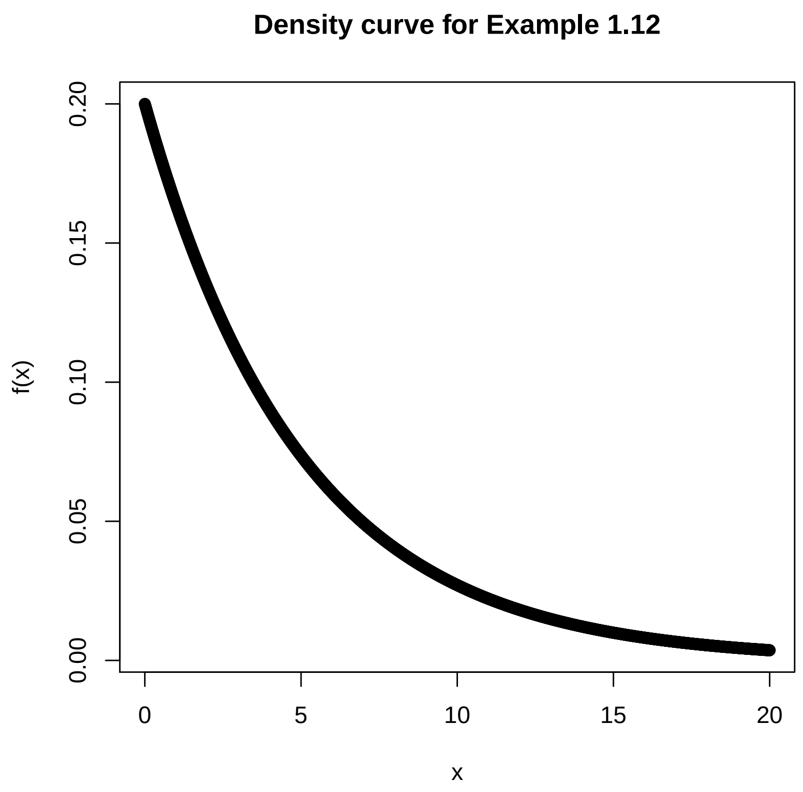
Example 1.14

```
> pnorm(1.25)
```

```
[1] 0.8944
```

Figure 1.16 on page 28 of Devore and Farnum.

```
> x = 0:2000/100
> # gives  a set of points from 0 to 20 in steps of 0.01
> # don't be tempted to try more points as this one is slow to evaluate the exponentials...
> f = 0.2*exp(-0.2*x)
> plot(x,f, ylab="f(x)", main="Density curve for Example 1.12")
```



Note that the standard normal distribution is the default so no need for parameters to be specified.

```
> pnorm(-0.38)
```

```
[1] 0.352
```

Example 1.14 cont'd

```
> pnorm(1,25) - pnorm(-0.38)
```

```
[1] -0.352
```

Example 1.15

```
> qnorm(0.67)
```

```
[1] 0.4399
```

Example 1.15 Cont'd

```
> qnorm(0.95)

[1] 1.645
```

Example 1.15 Cont'd

```
> qnorm(0.975)

[1] 1.96

> -qnorm(0.025)

[1] 1.96
```

Example 1.16

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

[1] 0.8615
```

This can be abbreviated to

```
> pnorm(1.75, 1.25, 0.46)

[1] 0.8615

> pnorm(1, 1.25, 0.46)

[1] 0.2934
```

Note the answers differ to the book as the z-value is not rounded by R.

```
> pnorm(1.75, 1.25, 0.46) - pnorm(1, 1.25, 0.46)

[1] 0.5681
```

Example 1.16 cont'd

```
> 1 - pnorm(1, 1.25, 0.46)

[1] 0.7066
```

is the same as

```
> pnorm(1, 1.25, 0.46, lower.tail=FALSE)

[1] 0.7066
```

Example 1.17

```
> qnorm(0.995, 64, 0.78)

[1] 66.01
```

1.5 Other Continuous Distributions

Example 1.18

not required in 161.100

Example 1.19

not required in 161.100

1.6 Several Useful Discrete Distributions

Example 1.20

```
> x=5:8
> Px = dbinom(x, 8, 0.5625)
> Px

[1] 0.26408 0.16977 0.06236 0.01002

> sum(Px)

[1] 0.5062
```

Note rounding differences exist An alternative is

```
> 1-pbinom(4, 8, 0.5625)

[1] 0.5062
```

this found the complement being the sum of 4 or less

to be sure of how things are working I prefer the summation approach given first.

Example 1.21

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

[1] 0.1708
```

This can be abbreviated to

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

```
[1] 0.1708

> ppois(5, 4.5)

[1] 0.7029

> x=0:12
> dpois(x, 4.5)

 [1] 0.011109 0.049990 0.112479 0.168718 0.189808 0.170827 0.128120
 [8] 0.082363 0.046329 0.023165 0.010424 0.004264 0.001599
```

Example 1.22

```
> x=0:3
> Px = dbinom(x, 100, 0.005)
> Px

[1] 0.60577 0.30441 0.07572 0.01243

> sum(Px)

[1] 0.9983
```

or

```
> pbinom(3, 100, 0.005)

[1] 0.9983
```

Chapter 2

Numerical Summary Measures

Before you can do any examples in this chapter, you will need to execute the command

```
> library(DevFarn2)
```

This will then give you direct access to the data for each example, and a few extra functions needed for some chapters.

Note that you must issue this command as the first act in any R session.

2.1 Measures of Center

Example 2.1

```
> data(e2.1)
> e2.1

      CrackLength
'x1'         16.1
'x2'          9.6
'x3'         24.9
'x4'         20.4
'x5'         12.7
'x6'         21.2
'x7'         30.2
'x8'         25.8
'x9'         18.5
'x10'        10.3
'x11'        25.3
'x12'        14.0
'x13'        27.1
'x14'        45.0
'x15'        23.3
'x16'        24.2
'x17'        14.6
'x18'         8.9
```

```
'x19'        32.4
'x20'        11.8
'x21'        28.5
```

note the way R formats the data as a data frame with only one variable not as a vector

```
> class(e2.1)

[1] "data.frame"
```

we could convert to a vector using as.vector()

```
> mean(e2.1)

[1] NA
```

Figure 2.1 on page 61 of Devore and Farnum.

```
> stem(e2.1$CrackLength)

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

  0 | 9
  1 | 00234569
  2 | 013455679
  3 | 02
  4 | 5
```

Note the difference in standard number of significant digits printed by R.

Example 2.2

```
> data(e2.2)
> e2.2

 [1]  7.6  8.3  9.3  9.4  9.4  9.7 10.4 11.5 11.9 15.2 16.2 20.4

> median(e2.2)

[1] 10.05

> mean(e2.2)

[1] 11.61
```

Example 2.3

```
> data(e2.3)
> e2.3

 [1]  612  623  666  744  883  898  964  970  983 1003 1016 1022 1029 1058
[15] 1085 1088 1122 1135 1197 1201

> mean(e2.3)

[1] 965

> median(e2.3)

[1] 1010

> mean(e2.3, trim=0.1)

[1] 979.1

> mean(e2.3, trim=0.2)

[1] 999.9
```

Example 2.4

manual working might be simpler but...

```
> x= 0:4
> Px = c(0.8, 0.14, 0.03, 0.02, 0.01)
> sum(x*Px)

[1] 0.3
```

Example 2.5

is not required in 161.100

Example 2.6

is not required in 161.100

2.2 Measures of Variability

Example 2.7

```
> data(e2.7)
> e2.7
```

```
   X.xi. X.xi.x.bar.. X.xi.x.bar.squared.
1  0.684      -0.9841              0.9685
2  2.540       0.8719              0.7602
3  0.924      -0.7441              0.5537
4  3.130       1.4619              2.1372
5  1.038      -0.6301              0.3970
6  0.598      -1.0701              1.1451
7  0.483      -1.1851              1.4045
8  3.520       1.8519              3.4295
9  1.285      -0.3831              0.1468
10 2.650       0.9819              0.9641
11 1.497      -0.1711              0.0293
```

we only want the first column

```
> x = e2.7[,1]
> x

 [1] 0.684 2.540 0.924 3.130 1.038 0.598 0.483 3.520 1.285 2.650 1.497

> var(x)

[1] 1.194

> sd(x)

[1] 1.093
```

Example 2.8

```
> data(e2.8)
> e2.8

     xi   xi2
1  15.2 231.0
2  16.8 282.2
3  12.6 158.8
4  13.2 174.2
5  12.8 163.8
6  13.8 190.4
7  16.3 265.7
8  13.0 169.0
9  12.7 161.3
10 15.8 249.6
11 19.2 368.6
12 12.7 161.3
13 15.6 243.4
14 13.5 182.2
15 12.9 166.4
```

again only the first column is relevant

```
> x = e2.8[,1]
> x
```

```
 [1] 15.2 16.8 12.6 13.2 12.8 13.8 16.3 13.0 12.7 15.8 19.2 12.7 15.6 13.5
[15] 12.9

> var(x)

[1] 3.918

> sd(x)

[1] 1.979
```

Example 2.9

manual working but...

```
> x=0:3
> Px=c(0.532, 0.389, 0.076, 0.003)
> Vx = sum(Px*x^2) - sum(x*Px)^2
> Vx

[1] 0.4175

> sqrt(Vx)

[1] 0.6461
```

Example 2.10

is not required in 161.100

2.3 More Detailed Summary Quantities

Example 2.11

```
> data(e2.11)
> e2.11

 [1]  5.9  7.2  7.3  6.3  8.1  6.8  7.0  7.6  6.8  6.5  7.0  6.3  7.9  9.0
[15]  8.2  8.7  7.8  9.7  7.4  7.7  9.7  7.8  7.7 11.6 11.3 11.8 10.7

> median(e2.11)

[1] 7.7

> fivenum(e2.11)

[1]  5.90  7.00  7.70  8.85 11.80
```

includes minimum, lower quartile, median, upper quartile and maximum

```
> IQR(e2.11)

[1] 1.85
```

Example 2.12

manual working

Example 2.13

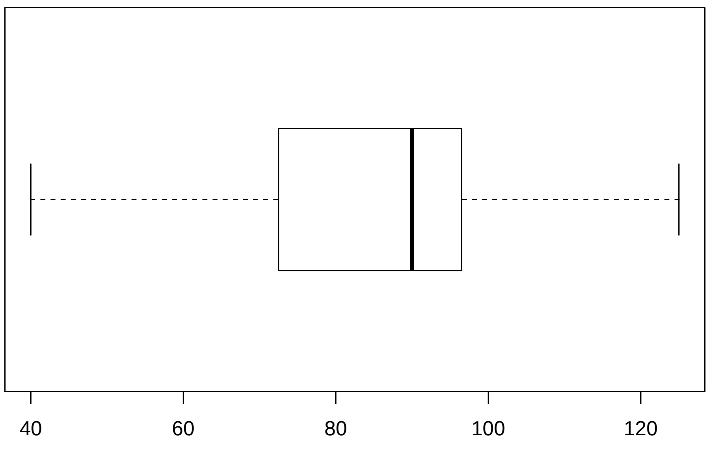
```
> qnorm(0.25)

[1] -0.6745

> qnorm(0.75)

[1] 0.6745
```

Example 2.14

```
> data(e2.14)
> e2.14

 [1]  40  52  55  60  70  75  85  85  90  90  92  94  94  95  98 100 115
[18] 125 125

> fivenum(e2.14)

[1]  40.0  72.5  90.0  96.5 125.0
```

Figure 2.8 on page 82 of Devore and Farnum.

```
> windows(7, 5)
> boxplot(e2.14, horizontal=TRUE)
```



Example 2.15

```
> data(e2.15)
> e2.15

   Type CompressionStrength
1     1               655.5
2     1               788.3
3     1               734.3
4     1               721.4
5     1               679.1
6     1               699.4
7     2               789.2
8     2               772.5
9     2               786.9
10    2               686.1
11    2               732.1
12    2               774.8
13    3               737.1
14    3               639.0
15    3               696.3
16    3               671.7
17    3               717.2
18    3               727.1
19    4               535.1
20    4               628.7
21    4               542.4
22    4               559.0
23    4               586.9
24    4               520.0
```

Example 2.16

```
> data(e2.16)
> e2.16

 [1]   5.3   8.2  13.8  74.1  85.3  88.0  90.2  91.5  92.4  92.9  93.6
[12]  94.3  94.8  94.9  95.5  95.8  95.9  96.6  96.7  98.1  99.0 101.4
[23] 103.7 106.0 113.5

> fivenum(e2.16)

[1]   5.3  90.2  94.8  96.7 113.5

> IQR(e2.16)

[1] 6.5

> sort(2.16)

[1] 2.16
```

Figure 2.9 on page 83 of Devore and Farnum.

```
> boxplot(CompressionStrength~Type, data=e2.15, xlab="Type", ylab="Compression Strength")
```

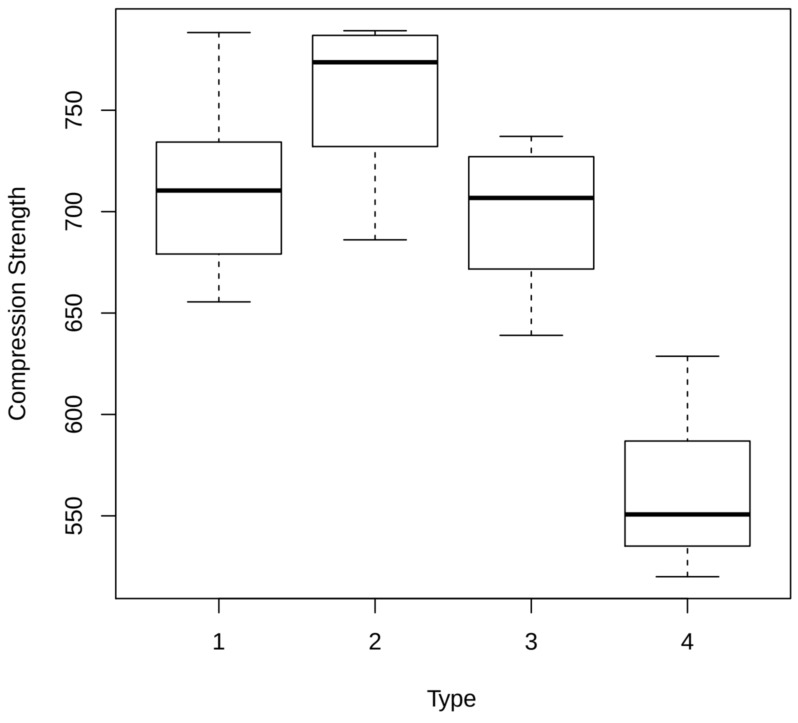
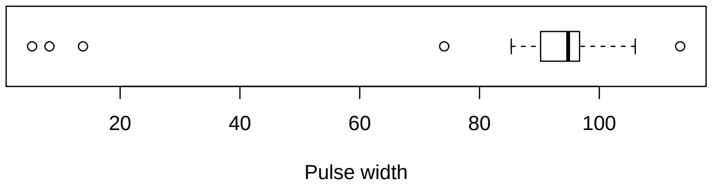


Figure 2.10 on page 85 of Devore and Farnum.

```
> windows(7, 2.5)
> boxplot(e2.16, horizontal=TRUE, xlab="Pulse width")
```



Example 2.17

```
> qnorm(0.9)

[1] 1.282

> qnorm(0.1)

[1] -1.282
```

2.4 Quantile Plots

Example 2.18

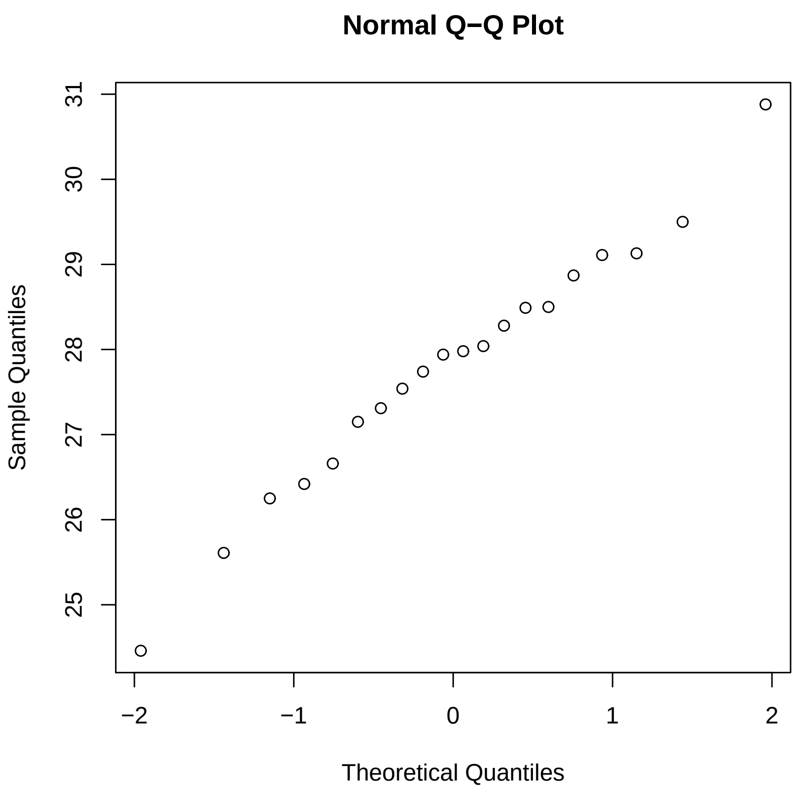
```
> 1:20 - 0.5

 [1]  0.5  1.5  2.5  3.5  4.5  5.5  6.5  7.5  8.5  9.5 10.5 11.5 12.5 13.5
[15] 14.5 15.5 16.5 17.5 18.5 19.5

> (1:20 - 0.5)/20

 [1] 0.025 0.075 0.125 0.175 0.225 0.275 0.325 0.375 0.425 0.475 0.525
[12] 0.575 0.625 0.675 0.725 0.775 0.825 0.875 0.925 0.975

> qnorm((1:20 - 0.5)/20)

 [1] -1.95996 -1.43953 -1.15035 -0.93459 -0.75542 -0.59776 -0.45376
 [8] -0.31864 -0.18912 -0.06271  0.06271  0.18912  0.31864  0.45376
[15]  0.59776  0.75542  0.93459  1.15035  1.43953  1.95996

> data(e2.18)
> e2.18

      X1 zPercentile
1  24.46       -1.96
2  25.61       -1.44
3  26.25       -1.15
4  26.42       -0.93
5  26.66       -0.76
6  27.15       -0.60
7  27.31       -0.45
8  27.54       -0.32
9  27.74       -0.19
10 27.94       -0.06
11 27.98        0.06
12 28.04        0.19
13 28.28        0.32
14 28.49        0.45
15 28.50        0.60
16 28.87        0.76
17 29.11        0.93
18 29.13        1.15
19 29.50        1.44
20 30.88        1.96
```

Example 2.19

is not required in 161.100

Figure 2.11 on page 91 of Devore and Farnum.

```
> qqnorm(e2.18[,1])
```



Chapter 3

Bivariate and Multivariate Data and Distributions

Before you can do any examples in this chapter, you will need to execute the command

```
> library(DevFarn2)
```

This will then give you direct access to the data for each example, and a few extra functions needed for some chapters.

Note that you must issue this command as the first act in any R session.

3.1 Scatter Plots

Example 3.1

```
> data(e3.1)
> e3.1

      x    y
1  0.40 1.02
2  0.42 1.21
3  0.48 0.88
4  0.51 0.98
5  0.57 1.52
6  0.60 1.83
7  0.70 1.50
8  0.75 1.80
9  0.75 1.74
10 0.78 1.63
11 0.84 2.00
12 0.95 2.80
13 0.99 2.48
14 1.03 2.47
15 1.12 3.05
```

```
16 1.15 3.18
17 1.20 3.76
18 1.25 3.68
19 1.25 3.82
20 1.28 3.21
21 1.30 4.27
22 1.34 3.12
23 1.37 3.99
24 1.40 3.75
25 1.43 4.10
26 1.46 4.18
27 1.49 3.77
28 1.55 4.34
29 1.58 4.21
30 1.60 4.92
```

R cannot add marginal plots like the actual figure.

note that the rug doesn't show the two different points that have the same value of x. Use

```
> rug(jitter(e3.1$x), side=3)
```

instead.

Example 3.2

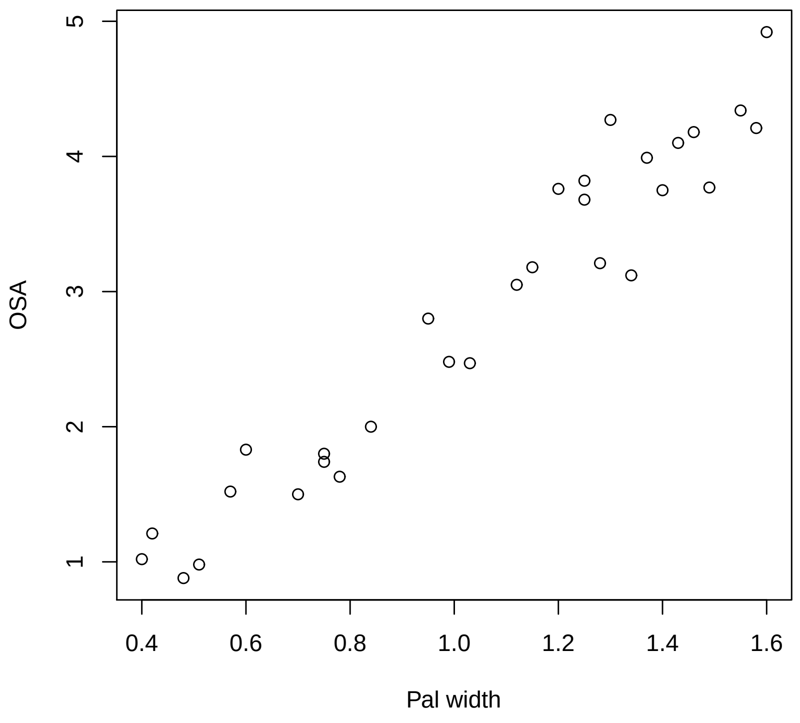
```
> data(e3.2)
> e3.2
```

```
     x    y
1  3.3  7.3
2  3.4 10.8
3  3.4 13.1
4  3.5 10.4
5  3.6  5.8
6  3.6  9.3
7  3.7 12.4
8  3.7 14.9
9  3.8 11.2
10 3.8  8.0
11 3.9  6.6
12 4.0 10.0
13 4.1  9.2
14 4.2 12.4
15 4.3  2.3
16 4.4  4.3
17 4.5  3.0
18 5.0  1.6
19 5.1  1.0
```

Note that R chooses different limits and labels for the axes than does Minitab.

Figure 3.2 on page 101 of Devore and Farnum.

```
> plot(e3.1$x, e3.1$y, xlab="Pal width", ylab="OSA")
```



```
> plot(e3.1$x, e3.1$y, xlab="Pal width", ylab="OSA")
> rug(e3.1$x, side=3)
> rug(e3.1$y, side=4)
```

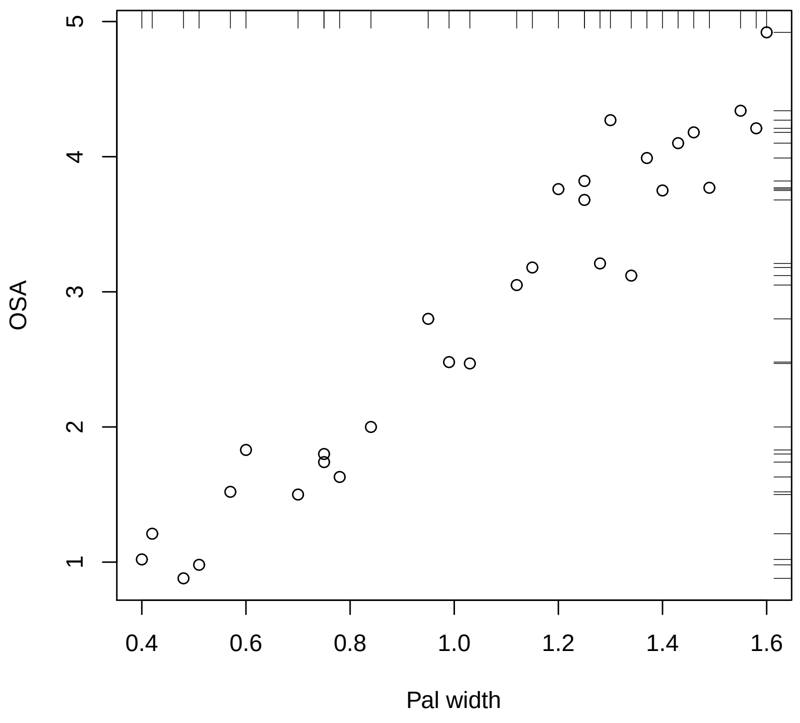
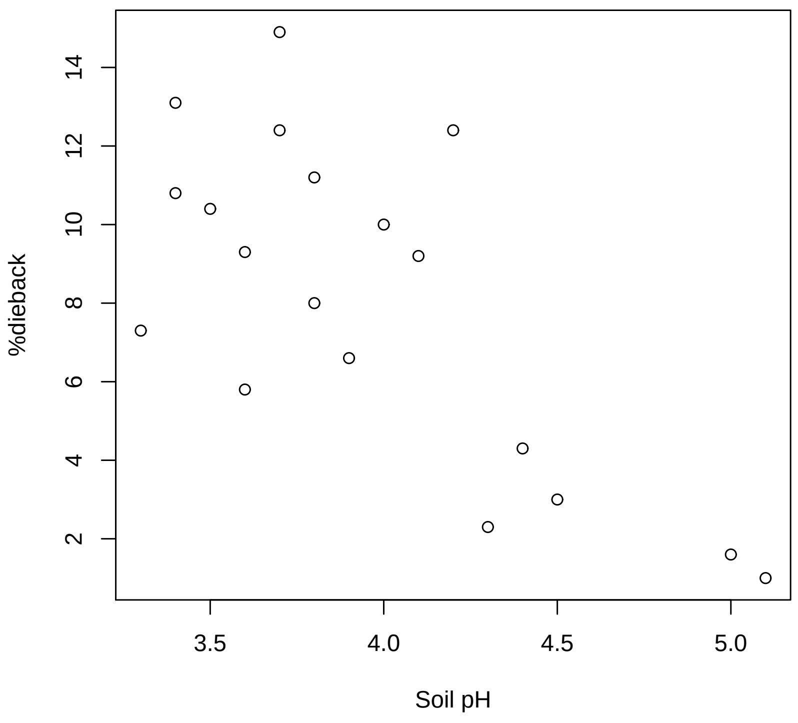
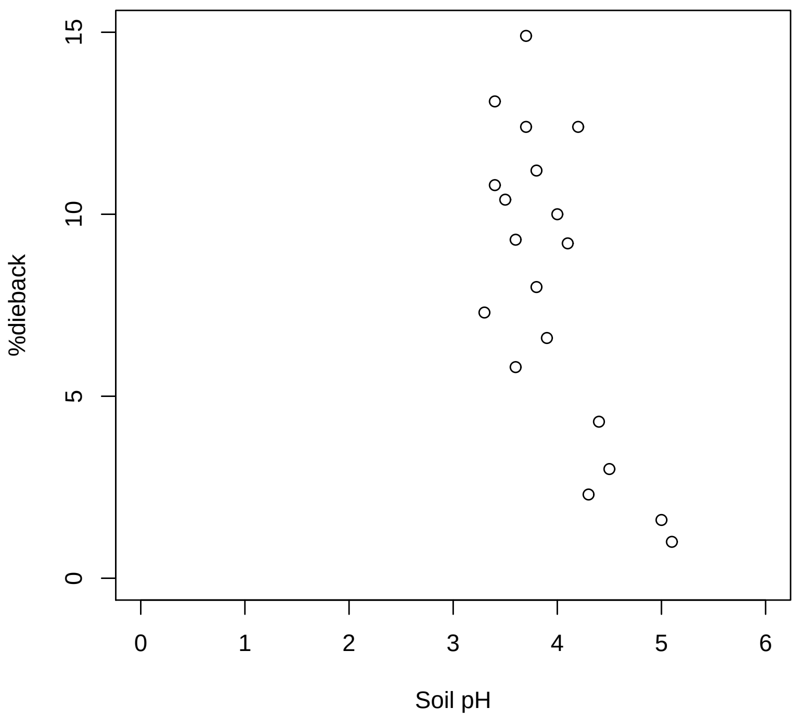


Figure 3.3 on page 103 of Devore and Farnum.

```
> plot(e3.2$x, e3.2$y, xlab= "Soil pH", ylab="%dieback")
```



```
> plot(e3.2$x, e3.2$y, xlab= "Soil pH", ylab="%dieback", xlim=c(0,6), ylim=c(0,15))
```



3.2 Correlation

Example 3.3

```
> data(e3.3)
> e3.3
```

```
     x    y
1 2.40 1.33
2 3.40 2.12
3 4.60 1.80
4 3.78 1.65
5 2.20 2.00
6 3.30 1.76
7 4.00 2.11
8 2.10 1.63
```

```
> cor(e3.3$x, e3.3$y)
```

```
[1] 0.3378
```

or

```
> cor(e3.3)
```

```
       x      y
x 1.0000 0.3378
y 0.3378 1.0000
```

Note the rounding errors created during the manual working

Example 3.4

has no working

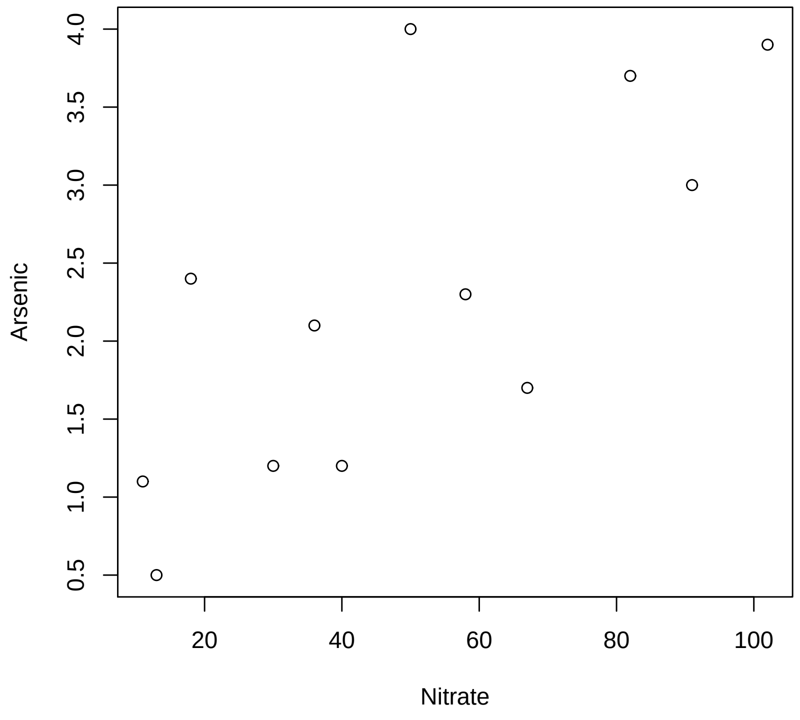
Example 3.5

```
> data(e3.5)
> e3.5
```

```
     x   y
1   11 1.1
2   13 0.5
3   18 2.4
4   30 1.2
5   36 2.1
6   40 1.2
7   50 4.0
8   58 2.3
9   67 1.7
10  82 3.7
11  91 3.0
12 102 3.9
```

Figure 3.7 on page 111 of Devore and Farnum.

```
> plot(e3.5$x, e3.5$y, xlab="Nitrate", ylab="Arsenic")
```



Example 3.6

```
> data(e3.6)
> e3.6
```

```
  x  y
1 1 74
2 2 54
3 3 52
4 4 51
5 5 52
6 6 53
7 7 58
8 8 71
```

Example 3.6 cont'd

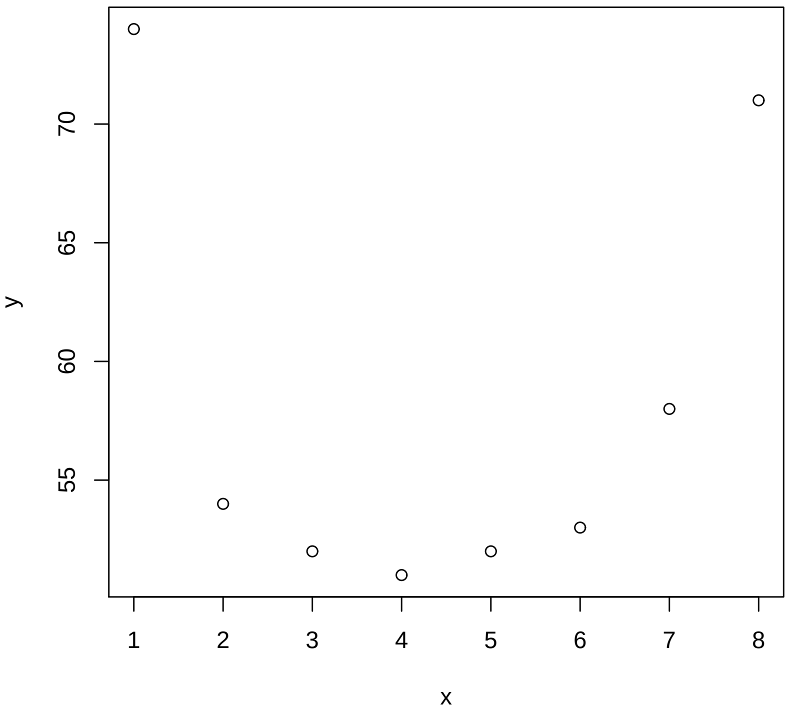
p111 manual working not required.

```
> cor(e3.6)
```

```
         x        y
x 1.000000 0.009554
y 0.009554 1.000000
```

Figure 3.8 on page 112 of Devore and Farnum.

```
> plot(e3.6$x, e3.6$y, xlab="x", ylab="y")
```



3.3 Fitting a Line to Bivariate Data

Example 3.7

```
> data(e3.7)
> e3.7

       x    y
1  125.3 77.9
2   98.2 76.8
3  201.4 81.5
4  147.3 79.8
5  145.9 78.2
6  124.7 78.3
7  112.2 77.5
8  120.2 77.0
9  161.2 80.1
10 178.9 80.2
11 159.5 79.9
12 145.8 79.0
13  75.1 76.7
14 151.4 78.2
15 144.2 79.5
16 125.0 78.1
17 198.8 81.5
```

```
18 132.5 77.0
19 159.6 79.0
20 110.7 78.6
```

```
> e3.7.lm = lm(y~x, data=e3.7)
```

This creates an object with lots of information

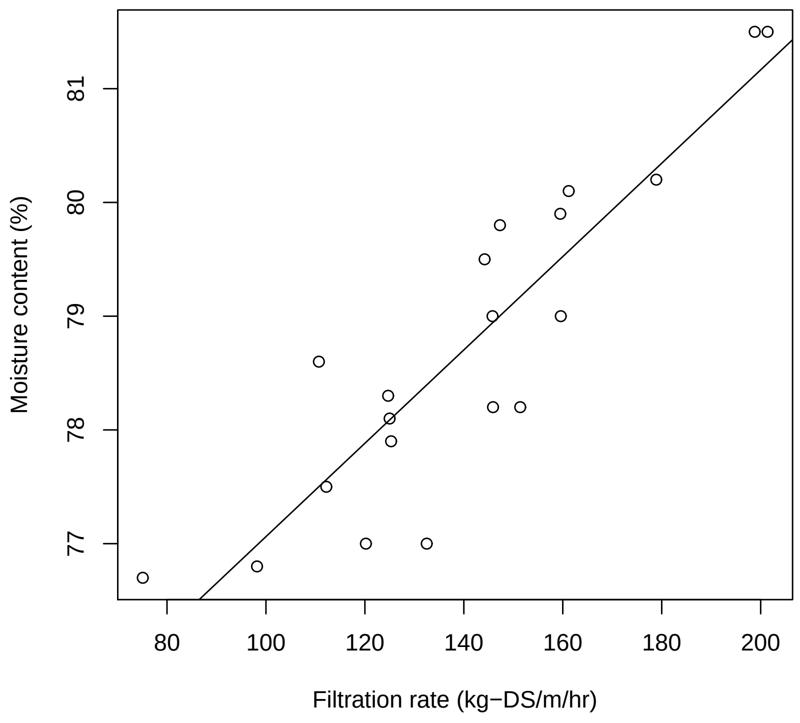
```
> coef(e3.7.lm)
```

```
(Intercept)           x
   72.95855     0.04103
```

Note that R gives us just what we want! and only if we ask for it! there are minor

Figure 3.10 on page 118 of Devore and Farnum.

```
> plot(e3.7$x, e3.7$y, xlab="Filtration rate (kg-DS/m/hr)", ylab="Moisture content (%)")
> abline(e3.7.lm)
```



but unimportant differences between the R plot and the book. also note that R does not have a ”fitted line plot” function so does not give the values of S or R-Sq in the figure.

Example 3.8

actually 3.7 cont'd

```
> summary(e3.7.lm)


Call:
lm(formula = y ~ x, data = e3.7)

Residuals:
    Min      1Q  Median      3Q     Max
-1.3955 -0.2769  0.0355  0.4291  1.0990

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 72.95855    0.69753  104.60  < 2e-16 ***
x            0.04103    0.00484    8.48  1.1e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.665 on 18 degrees of freedom
Multiple R-squared:   0.8,        Adjusted R-squared:  0.789
F-statistic:   72 on 1 and 18 DF,  p-value: 1.05e-07
```

although we can ask for

```
> summary(e3.7.lm)$r.squared


[1] 0.7999
```

and

```
> summary(e3.7.lm)$sigma


[1] 0.6653
```

Example 3.9

```
> data(e3.9)
> e3.9


  x1 y1 x2  y2 x3 y3
1 15 42  5  16  5  8
2 16 35 10  32 10 16
3 17 45 15  44 15 22
4 18 42 20  45 20 23
5 19 49 25  63 25 31
6 20 46 50 115 50 60
```

No real added value in doing this example in R.

Example 3.10

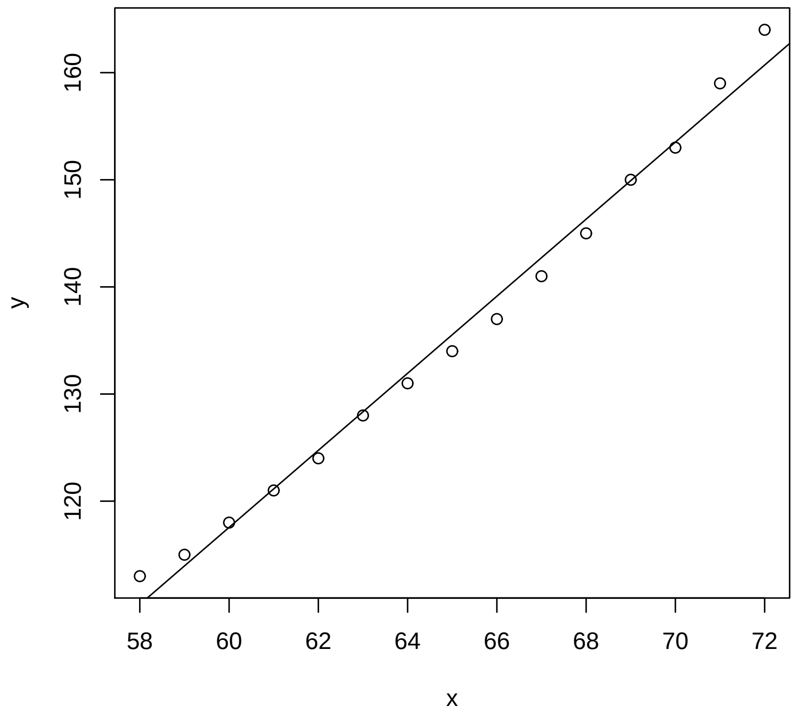
```
> data(e3.10)
> e3.10

    x   y
1  58 113
2  59 115
3  60 118
4  61 121
5  62 124
6  63 128
7  64 131
8  65 134
9  66 137
10 67 141
11 68 145
12 69 150
13 70 153
14 71 159
15 72 164

> e3.10.lm = lm(y~x, data=e3.10)
```

Example 3.11

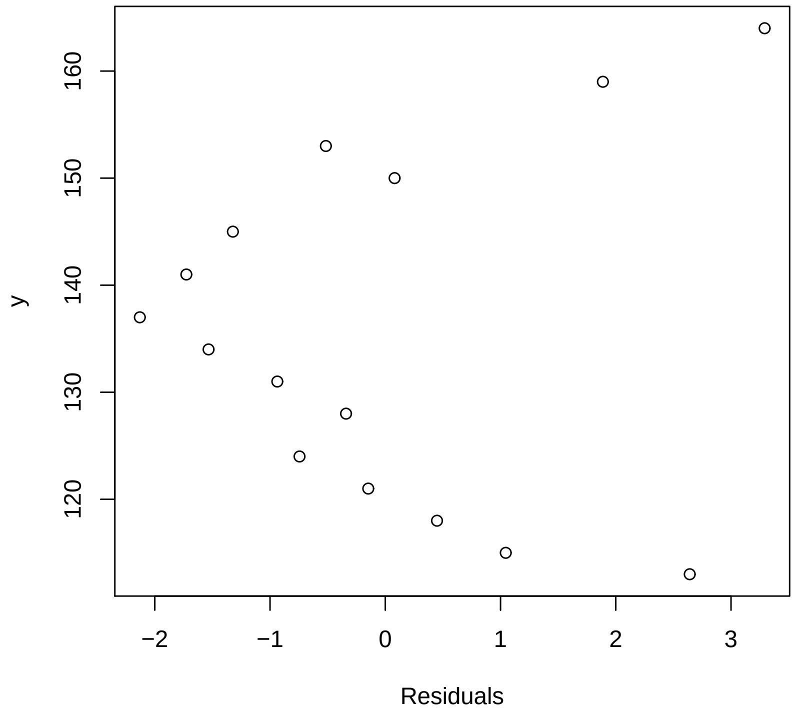
```
> data(e3.11)
> e3.11

     x   y  yFit yResid
1  119 239 225.1   13.9
2  140 262 240.3   21.7
3  150 202 247.6  -45.6
4  157 224 252.7  -28.7
5  171 255 262.8   -7.8
6  200 292 283.9    8.1
7  218 350 296.9   53.1
8  250 298 320.2  -22.2
9  272 313 336.2  -23.2
10 321 415 371.7   43.3
11 573 542 554.7  -12.7

> e3.11.lm1 = lm(y~x, data=e3.11)
> coef(e3.11.lm1)

(Intercept)           x
    138.682       0.726

> e3.11[-11,] # drops the last row

     x   y  yFit yResid
1  119 239 225.1   13.9
2  140 262 240.3   21.7
3  150 202 247.6  -45.6
4  157 224 252.7  -28.7
```

Figure 3.13 on page 123 of Devore and Farnum.

```
> plot(e3.10$x, e3.10$y, xlab="x", ylab="y")
> abline(e3.10.lm)
```



```
> plot(residuals(e3.10.lm), e3.10$y, xlab="Residuals", ylab="y")
```



```
5  171 255 262.8   -7.8
6  200 292 283.9    8.1
7  218 350 296.9   53.1
8  250 298 320.2  -22.2
9  272 313 336.2  -23.2
10 321 415 371.7   43.3

> e3.11.lm2 = lm(y~x, data=e3.11[-11,])
> coef(e3.11.lm2)

(Intercept)           x
   115.0893      0.8504
```

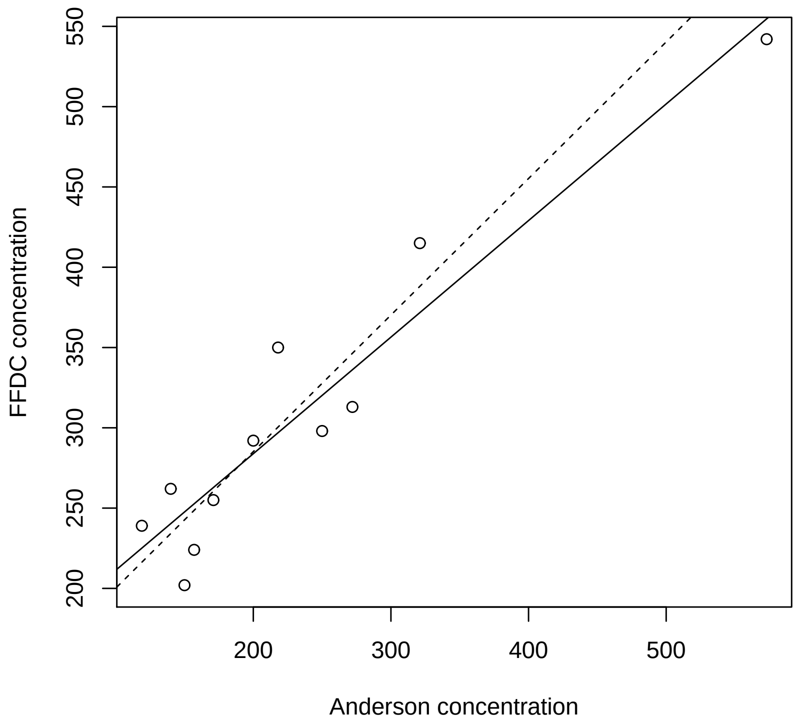
3.4 Nonlinear Relationships

Example 3.12

```
> data(e3.12)
> e3.12

   x    y
1  5 16.3
2 10  9.7
3 15  8.1
4 20  4.2
5 25  3.4
6 30  2.9
7 45  1.9
8 60  1.3

> e3.12.lm = lm(log(y)~log(x), data = e3.12)
> summary(e3.12.lm)

Call:
lm(formula = log(y) ~ log(x), data = e3.12)

Residuals:
    Min      1Q  Median      3Q     Max
-0.1586 -0.0652 -0.0213  0.0104  0.2947

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.6384     0.2110    22.0  5.8e-07 ***
log(x)       -1.0492     0.0679   -15.5  4.6e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.145 on 6 degrees of freedom
Multiple R-squared:  0.976,        Adjusted R-squared:  0.971
F-statistic:  239 on 1 and 6 DF,  p-value: 4.63e-06
```

Figure 3.14 on page 125 of Devore and Farnum.

```
> plot(e3.11$x, e3.11$y, xlab="Anderson concentration", ylab="FFDC concentration")
> abline(e3.11.lm1)
> abline(e3.11.lm2, lty=2)
```



```
> plot(fitted(e3.11.lm1), resid(e3.11.lm1), xlab="Fitted values", ylab="Residuals")
> abline(h=0)
```

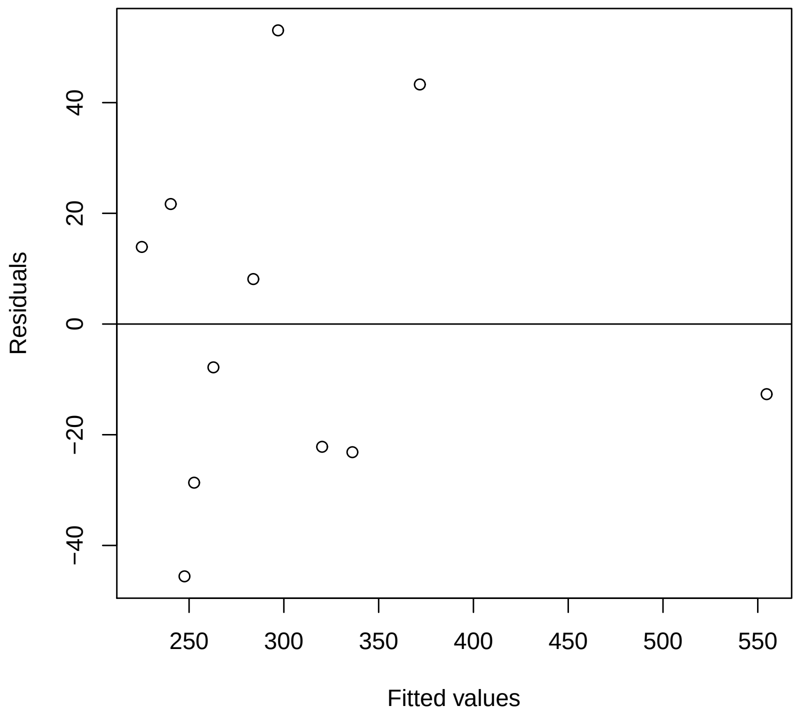
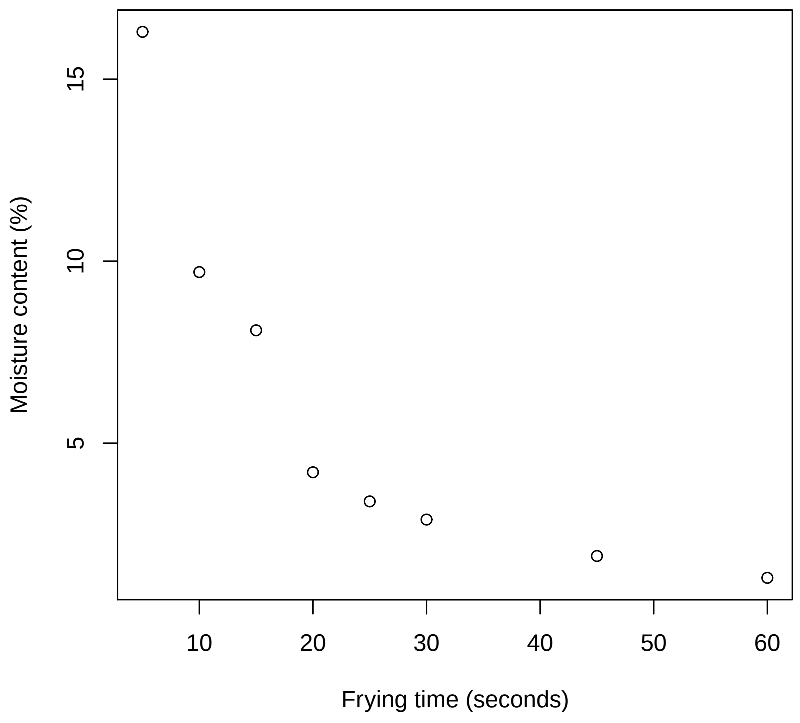
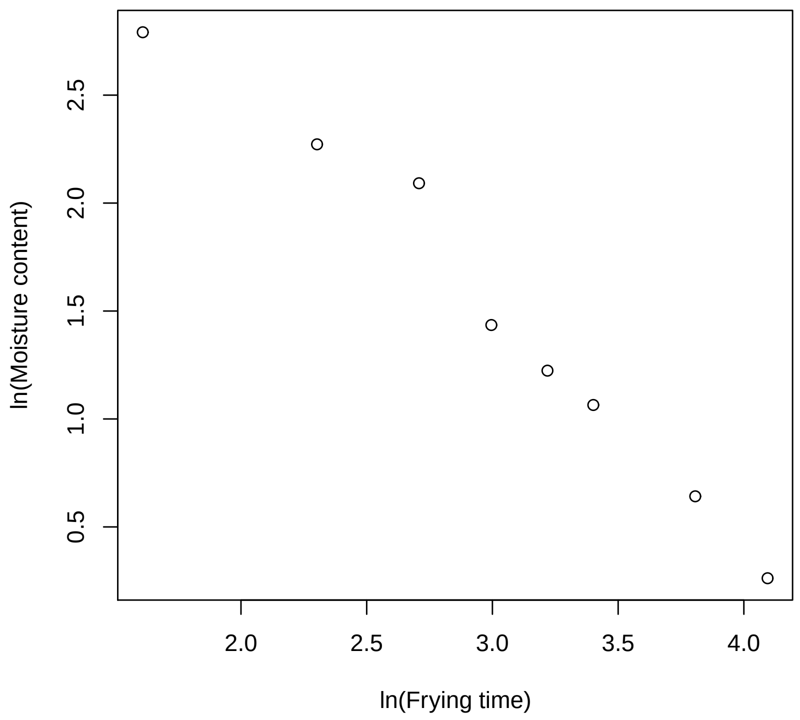


Figure 3.16 on page 131 of Devore and Farnum.

```
> plot(e3.12$x, e3.12$y, xlab="Frying time (seconds)", ylab="Moisture content (%)")
```



```
> plot(log(e3.12$x), log(e3.12$y), xlab="ln(Frying time)", ylab="ln(Moisture content)")
```



```
> plot(log(e3.12$x), resid(e3.12.lm), xlab="ln(Frying time)", ylab="Residual")
```

Example 3.13

This example uses data from Example 3.6 given above.

```
> str(e3.6)

'data.frame':        8 obs. of  2 variables:
 $ x: int  1 2 3 4 5 6 7 8
 $ y: int  74 54 52 51 52 53 58 71

> x2 = e3.6$x^2
```

This makes the new variable x^2 required for the quadratic model.

```
> e3.13.lm = lm(y~x+x2, data=e3.6)
> summary(e3.13.lm)

Call:
lm(formula = y ~ x + x2, data = e3.6)

Residuals:
     1      2      3      4      5      6      7      8
 3.625 -5.804 -0.768  1.732  2.696  0.125 -1.982  0.375

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   84.482      4.904   17.23  1.2e-05 ***
x            -15.875      2.500   -6.35   0.0014 **
x2             1.768      0.271    6.52   0.0013 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.51 on 5 degrees of freedom
Multiple R-squared:  0.895,        Adjusted R-squared:  0.853
F-statistic: 21.3 on 2 and 5 DF,  p-value: 0.00359
```

The fitted values from this model are found using

```
> fitted(e3.13.lm)

    1     2     3     4     5     6     7     8
70.37 59.80 52.77 49.27 49.30 52.88 59.98 70.62
```

Example 3.14

```
> data(Bears)
> head(Bears)

   Name ID Age Month Sex Head.L Head.W Neck.G Length Chest.G Weight Obs.No
1 Allen 39  19     7   1   10.0    5.0   15.0   45.0      23     65      1
2 Berta 41  19     7   2   11.0    6.5   20.0   47.5      24     70      1
3 Berta 41  20     8   2   12.0    6.0   17.0   57.0      27     74      2
4 Berta 41  23    11   2   12.5    5.0   20.5   59.5      38    142      3
5 Berta 41  29     5   2   12.0    6.0   18.0   62.0      31    121      4
6 Clyde 43  19     7   1   11.0    5.5   16.0   53.0      26     80      1
```

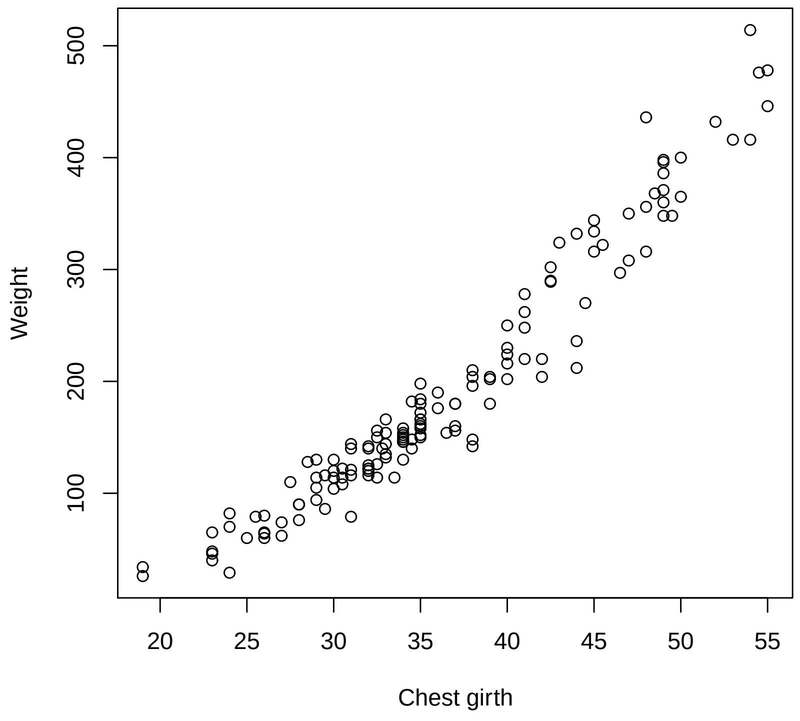
```
> str(Bears)
```

```
'data.frame':        143 obs. of  12 variables:
 $ Name   : Factor w/ 99 levels "","Abe","Adam",..: 6 9 9 9 9 18 18 24 24 72 ...
 $ ID     : int  39 41 41 41 41 43 43 45 45 48 ...
 $ Age    : int  19 19 20 23 29 19 20 55 67 81 ...
 $ Month  : int  7 7 8 11 5 7 8 7 7 9 ...
 $ Sex    : int  1 2 2 2 2 1 1 1 1 1 ...
 $ Head.L : num  10 11 12 12.5 12 11 12 16.5 16.5 15.5 ...
 $ Head.W : num  5 6.5 6 5 6 5.5 5.5 9 9 8 ...
 $ Neck.G : num  15 20 17 20.5 18 16 17 28 27 31 ...
 $ Length : num  45 47.5 57 59.5 62 53 56 67.5 78 72 ...
 $ Chest.G: num  23 24 27 38 31 26 30.5 45 49 54 ...
 $ Weight : int  65 70 74 142 121 80 108 344 371 416 ...
 $ Obs.No : int  1 1 2 3 4 1 2 1 2 1 ...
```

These commands were used because you don't want the whole data printed out!

Figure 3.17 on page 134 of Devore and Farnum.

```
> plot(Bears$Chest.G, Bears$Weight, xlab="Chest girth", ylab="Weight")
```



Note Figure 3.17b not required.

3.5 Using More Than One Predictor

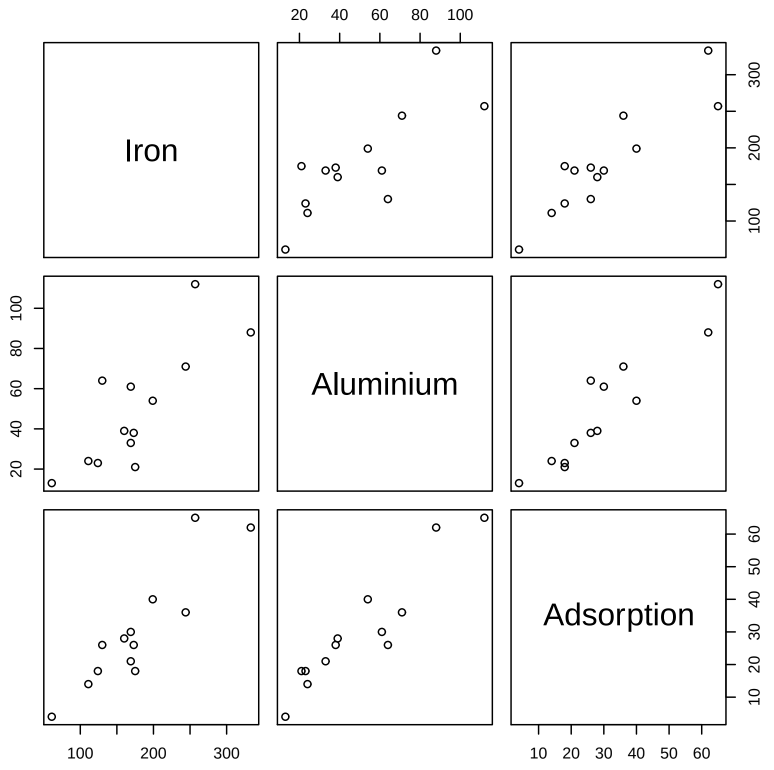
This section isn't compulsory reading. Examples are presented for completeness.

Example 3.15

```
> data(e3.15)
> e3.15

    x1  x2  y
1   61  13  4
2  175  21 18
3  111  24 14
4  124  23 18
5  130  64 26
6  173  38 26
7  169  33 21
8  169  61 30
9  160  39 28
10 244  71 36
11 257 112 65
12 333  88 62
13 199  54 40
```

Figure 3.18 on page 138 of Devore and Farnum.

```
> pairs(e3.15, labels=c("Iron", "Aluminium", "Adsorption"))
```



Example 3.16

= 3.15 cont'd

```
> e3.16.lm = lm(y~x1+x2, data=e3.15)
> summary(e3.16.lm)

Call:
lm(formula = y ~ x1 + x2, data = e3.15)

Residuals:
   Min     1Q Median     3Q    Max
-8.935 -2.218  0.461  3.345  6.071

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -7.3507     3.4847   -2.11  0.06110 .
x1            0.1127     0.0297    3.80  0.00350 **
x2            0.3490     0.0713    4.89  0.00063 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.38 on 10 degrees of freedom
Multiple R-squared:  0.948,        Adjusted R-squared:  0.938
F-statistic:   92 on 2 and 10 DF,  p-value: 3.63e-07

> predict(e3.16.lm, data.frame(x1=150, x2= 60))

   1
30.5
```

Example 3.17

```
> data(e3.17)
> e3.17

  x1   x2  x1x2 ComptStr Adsorbability
1 21 0.65 13.65    33.55          8.42
2 21 0.55 11.55    47.55          6.26
3  7 0.65  4.55    35.00          6.74
4  7 0.55  3.85    35.90          6.59
5 28 0.60 16.80    40.90          7.28
6  0 0.60  0.00    39.10          6.90
7 14 0.70  9.80    31.55         10.80
8 14 0.50  7.00    31.12          5.63
9 14 0.60  8.40    47.05          7.43

> e3.17.lm1 = lm(ComptStr~x1+x2, data=e3.17)
> summary(e3.17.lm1)

Call:
lm(formula = ComptStr ~ x1 + x2, data = e3.17)

Residuals:
   Min     1Q Median     3Q    Max
-9.189 -4.079 -0.649  3.431  9.081

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

```
(Intercept)   49.709     23.777    2.09    0.082 .
x1             0.164      0.278    0.59    0.576
x2           -23.400     38.914   -0.60    0.570
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.74 on 6 degrees of freedom
Multiple R-squared:  0.106,        Adjusted R-squared:  -0.192
F-statistic: 0.355 on 2 and 6 DF,  p-value: 0.715

> e3.17.lm2 = lm(ComptStr~x1+x2+x1x2, data=e3.17)
> summary(e3.17.lm2)

Call:
lm(formula = ComptStr ~ x1 + x2 + x1x2, data = e3.17)

Residuals:
     1      2      3      4      5      6      7      8      9
-1.124  3.986 -3.924  1.186  0.631  3.431 -4.079 -9.189  9.081

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -28.89      84.77   -0.34     0.75
x1              5.78       5.82    0.99     0.37
x2            107.60     141.08    0.76     0.48
x1x2           -9.36       9.68   -0.97     0.38

Residual standard error: 6.78 on 5 degrees of freedom
Multiple R-squared:  0.247,        Adjusted R-squared:  -0.205
F-statistic: 0.546 on 3 and 5 DF,  p-value: 0.672

> e3.17.lm3 = lm(Adsorbability~x1+x2, data=e3.17)
> summary(e3.17.lm3)

Call:
lm(formula = Adsorbability ~ x1 + x2, data = e3.17)

Residuals:
    Min      1Q  Median      3Q     Max
-1.4772 -0.2006 -0.0872  0.3994  1.3528

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -5.6628     3.1056   -1.82    0.118
x1            0.0251     0.0363    0.69    0.515
x2           21.0833     5.0827    4.15    0.006 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.88 on 6 degrees of freedom
Multiple R-squared:  0.747,        Adjusted R-squared:  0.662
F-statistic: 8.84 on 2 and 6 DF,  p-value: 0.0163

> e3.17.lm4 = lm(Adsorbability~x1+x2+x1x2, data=e3.17)
> summary(e3.17.lm4)
```

```
Call:
lm(formula = Adsorbability ~ x1 + x2 + x1x2, data = e3.17)

Residuals:
      1       2       3       4       5       6       7       8       9 
-0.6514  0.3019 -0.9747 -0.0214 -0.4106 -0.0872  1.3528  0.3994  0.0911 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    6.397     10.672    0.60     0.58
x1            -0.836      0.732   -1.14     0.31
x2             0.983     17.761    0.06     0.96
x1x2           1.436      1.219    1.18     0.29

Residual standard error: 0.853 on 5 degrees of freedom
Multiple R-squared:  0.802,	Adjusted R-squared:  0.683 
F-statistic: 6.74 on 3 and 5 DF,  p-value: 0.033 
```

not given in book but mentioned is the complete second-order relationship

```
> e3.17.lm5 = lm(Adsorbability~x1+x2+x1x2+x1^2+x2^2, data=e3.17)
> summary(e3.17.lm5)
```

```
Call:
lm(formula = Adsorbability ~ x1 + x2 + x1x2 + x1^2 + x2^2, data = e3.17)

Residuals:
      1       2       3       4       5       6       7       8       9 
-0.6514  0.3019 -0.9747 -0.0214 -0.4106 -0.0872  1.3528  0.3994  0.0911 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    6.397     10.672    0.60     0.58
x1            -0.836      0.732   -1.14     0.31
x2             0.983     17.761    0.06     0.96
x1x2           1.436      1.219    1.18     0.29

Residual standard error: 0.853 on 5 degrees of freedom
Multiple R-squared:  0.802,	Adjusted R-squared:  0.683 
F-statistic: 6.74 on 3 and 5 DF,  p-value: 0.033 
```

Example 3.18

ethanol data set from SPlus

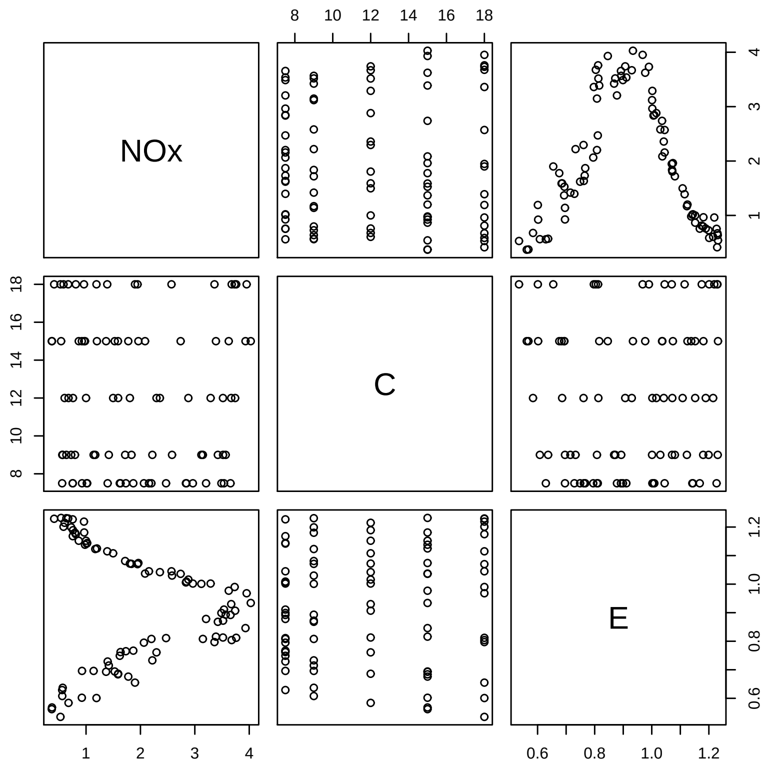
```
> data(Ethanol)
> head(Ethanol)
```

```
    NOx  C     E
1 3.741 12 0.907
2 2.295 12 0.761
3 1.498 12 1.108
4 2.881 12 1.016
5 0.760 12 1.189
6 3.120  9 1.001
```

Figure 3.22 not required in 161.100

Figure 3.21 on page 143 of Devore and Farnum.

```
> pairs(Ethanol)
```



3.6 Joint Distributions

This material is not covered in 161.100

Example 3.19

This material is not covered in 161.100

Example 3.20

This material is not covered in 161.100

Chapter 4

Obtaining Data

Before you can do any examples in this chapter, you will need to execute the command

```
> library(DevFarn2)
```

This will then give you direct access to the data for each example, and a few extra functions needed for some chapters.

Note that you must issue this command as the first act in any R session.

4.1 Operational Definitions

Example 4.1

This example has no calculations or associated figures

Example 4.2

This example has no calculations or associated figures

Example 4.3

This example has no calculations or associated figures

Example 4.4

This example has no calculations or associated figures

Example 4.5

This example has no calculations or associated figures

4.2 Data from Sampling

Example 4.6

This example has no calculations or associated figures

Example 4.7

This example has no calculations or associated figures

Example 4.8

This example has no calculations or associated figures

Example 4.9

This example has no calculations or associated figures

Example 4.10

This example has no calculations or associated figures

Example 4.11

```
> data(e4.11)
> e4.11

  Stratum Size StDev
1       A  500   0.2
2       B  300   0.2
3       C  100   0.4
4       D   50   0.4
5       E   50   0.6
6       F  200   0.8
```

This is not done simply using R In fact it is simpler to write a new function to do the work than actually do the calculations ourselves in R. This is well beyond the scope of an introductory course but should show you how users of R think.

```
> Neyman = function(Sizes, StDev, Conf=0.95, B=0.10){
+ ZScore=qnorm((1+Conf)/2)
+ BigN = sum(Sizes)
+ NISigmaI = Sizes*StDev
+ print(NISigmaI)
+ SmallN = (sum(NISigmaI))^2 /( (BigN*B/ZScore)^2 + sum(Sizes*StDev^2))
+ print(SmallN)
+ SmallN =ceiling(SmallN)
+ print(SmallN)
+ Allocation = SmallN*NISigmaI/sum(NISigmaI)
+ round(Allocation,0)
+ }
> Neyman(e4.11$Size, e4.11$StDev, Conf=0.90)

[1] 100  60  40  20  30 160
[1] 30.43
[1] 31
[1]  8  5  3  2  2 12
```

This should have worked but now we find several errors with this example exist. First there is a correction in the data file required

```
> e4.11[3,3] = 0.4
> e4.11

  Stratum Size StDev
1       A  500   0.2
2       B  300   0.2
3       C  100   0.4
4       D   50   0.4
5       E   50   0.6
6       F  200   0.8

> Neyman(e4.11$Size, e4.11$StDev, Conf=0.90)

[1] 100  60  40  20  30 160
[1] 30.43
[1] 31
[1]  8  5  3  2  2 12
```

But this did not bring the answers closer together sufficiently. Its just as well that this is not essential as this example is flawed. Forcing n=110 in the middle of our program does give the correct allocation in the end. D&F have incorrectly obtained n=110 and it has proven too hard to find the other mistakes made in preparing this example. Move on to Example 4.12 which does work for our program.

Example 4.12

```
> Neyman(c(2000, 4000, 8000, 5000, 1000), rep(0.3,5), B=0.03)

[1]  600 1200 2400 1500  300
[1] 376.9
[1] 377
[1]  38  75 151  94  19
```

We also tested Exercise 12 as it happens and are sure we are making correct statements here.

Example 4.13

This example has no calculations

4.3 Data from Experiments

Example 4.14

This example has no calculations

Example 4.15

The supplied data are not required

```
> sample(1:12)
```

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

You could repeat this to prove this is giving the randomness you expect.

Examples 4.16

This example has no calculations

4.4 Measurement Systems

Example 4.17

This example has no calculations

Example 4.18

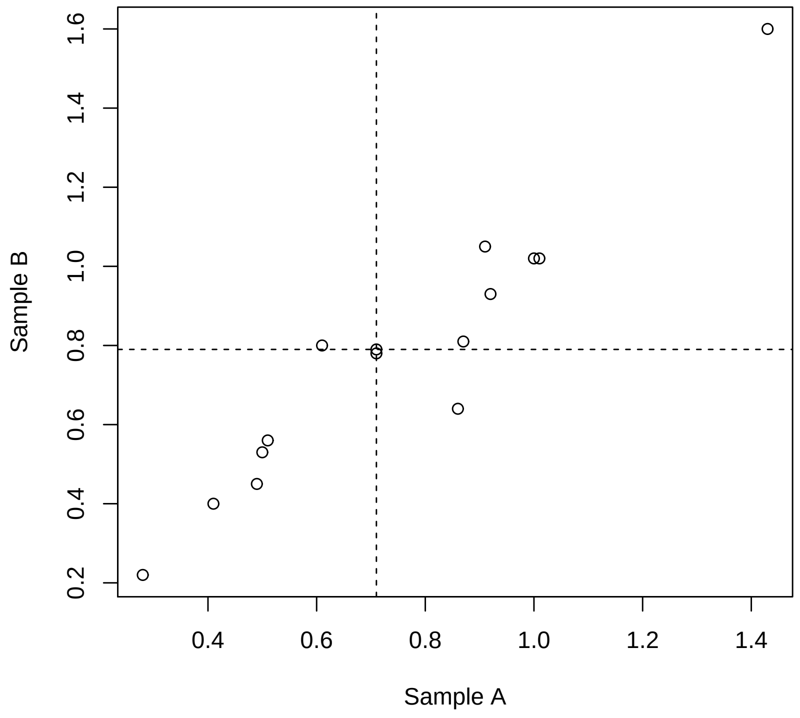
```
> data(e4.18)
> e4.18
```

```
   Sample.A Sample.B
1      0.86     0.64
2      0.61     0.80
3      0.41     0.40
4      0.92     0.93
5      0.71     0.78
6      1.00     1.02
7      0.49     0.45
8      0.50     0.53
9      0.51     0.56
10     0.87     0.81
```

```
11      1.01     1.02
12      0.71     0.79
13      1.43     1.60
14      0.28     0.22
15      0.91     1.05
```

Figure 4.6 on page 187 of Devore and Farnum.

```
> plot(e4.18[,1], e4.18[,2], xlab="Sample A", ylab="Sample B")
> abline(h=median(e4.18[,2]), lty=2)
> abline(v=median(e4.18[,1]), lty=2)
```



Chapter 5

Probability and Sampling Distributions

5.1 Chance Experiments

Example 5.1

This example has no calculations or associated figures

Example 5.2

This example has no calculations or associated figures

Example 5.3

This example has no calculations or associated figures

5.2 Probability Concepts

Example 5.4

This example has no calculations or associated figures

Example 5.5

This example has no calculations or associated figures

5.3 Conditional Probability and Independence

Example 5.6

This example has no calculations or associated figures

Example 5.7

This example has no calculations or associated figures

Example 5.8

This example has no calculations or associated figures

5.4 Random Variables

Example 5.9

```
> dbinom(0:1, 20, 0.02)

[1] 0.6676 0.2725

> pbinom(1, 20, 0.02)

[1] 0.9401
```

Example 5.10

```
> exp(-2)

[1] 0.1353
```

Example 5.11

Example 5.13

not required

5.5 Sampling Distributions

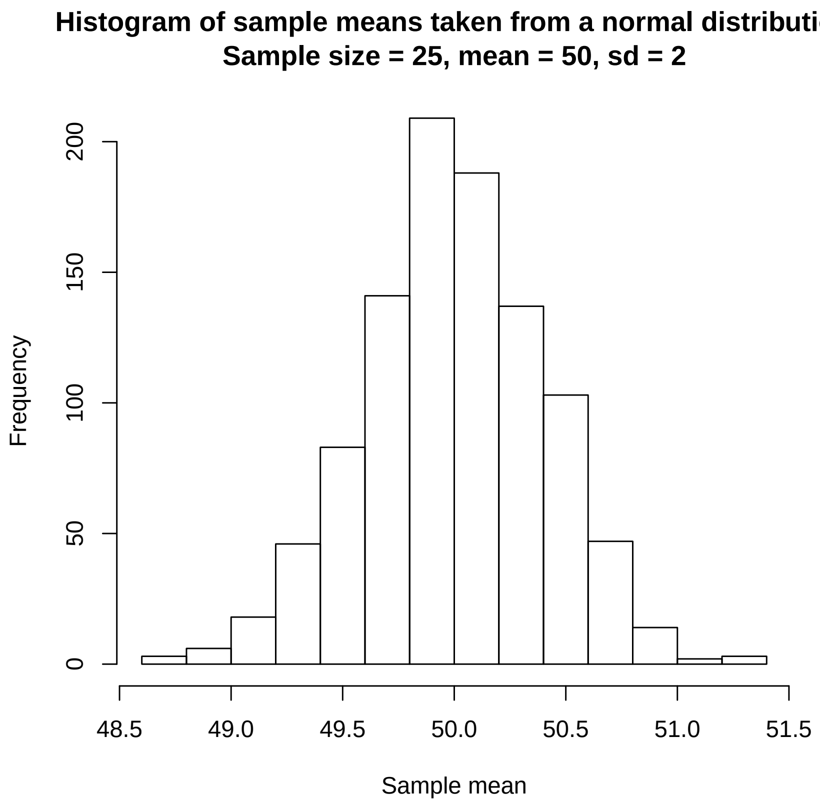
Example 5.14

```
> RandData = matrix(rnorm(25000, mean=50, sd=2), ncol=25)
> SamMeans = rowMeans(RandData)
```

This figure cannot exactly match one in the text, but certainly indicative, especially if repeated

Figure 5.14 on page 224 of Devore and Farnum.

```
> hist(SamMeans, main="Histogram of sample means taken from a normal distribution\nSample size = 25, me
```



Example 5.15

```
> mean(SamMeans)

[1] 50

> sd(SamMeans)

[1] 0.406
```

SUGGESTION: repeat these two examples to see how things vary.

5.6 Describing Sampling Distributions

Example 5.16

```
> pnorm(22, mean=20, sd=1.8/sqrt(5))

[1] 0.9935

> pnorm(18, mean=20, sd=1.8/sqrt(5))

[1] 0.006486
```

```
> round((pnorm(22, mean=20, sd=1.8/sqrt(5)) - pnorm(18, mean=20, sd=1.8/sqrt(5))), 5)

[1] 0.987
```

note the rounding errors in the text.

Example 5.17

This example has no suitable calculations for R

Example 5.18

```
> 1-pnorm(3.1, mean=3, sd=0.5/sqrt(100))

[1] 0.02275
```

Example 5.19

```
> sqrt(0.05*0.95/100)

[1] 0.02179

> (0.12-0.05)/sqrt(0.05*0.95/100)

[1] 3.212
```

Chapter 6

Quality and Reliability

Before you can do any examples in this chapter, you will need to execute the command

```
> library(DevFarn2)
```

This will then give you direct access to the data for each example, and a few extra functions needed for some chapters.

Note that you must issue this command as the first act in any R session.

Also note the qcc library must be loaded for many examples in this chapter. We would normally do this by issuing the command

```
> library(qcc)
```

but if you loaded the *DevFarn2* package, the *qcc* package is loaded automatically.

6.1 Terminology

6.2 How Control Charts Work

6.3 Control Charts for Mean and Variation

Example 6.1

```
> data(e6.1)
> rowMeans(e6.1)
```

```
      1       2       3       4       5       6       7       8       9 
0.00682 0.00760 0.00798 0.00732 0.00878 0.00670 0.00774 0.00822 0.00822 
     10      11      12      13      14      15      16      17      18 
0.00814 0.00806 0.00870 0.00816 0.00816 0.00806 0.00700 0.00838 0.00826 
     19      20 
0.00848 0.00854 
```

```
> mean(rowMeans(e6.1))

[1] 0.007966

> apply(e6.1,1,max)-apply(e6.1,1,min)

     1      2      3      4      5      6      7      8      9     10
0.0040 0.0029 0.0024 0.0024 0.0010 0.0036 0.0024 0.0023 0.0013 0.0032
    11     12     13     14     15     16     17     18     19     20
0.0036 0.0011 0.0014 0.0031 0.0017 0.0033 0.0026 0.0018 0.0010 0.0029

> mean(apply(e6.1,1,max)-apply(e6.1,1,min))

[1] 0.0024
```

Example 6.1 cont'd

Example 6.2

```
> apply(e6.1, 1, sd)

        1         2         3         4         5         6         7
0.0015881 0.0011023 0.0008955 0.0009418 0.0003768 0.0015264 0.0010831
        8         9        10        11        12        13        14
0.0009834 0.0005167 0.0012361 0.0012915 0.0004637 0.0006309 0.0012219
       15        16        17        18        19        20
0.0006656 0.0013248 0.0010941 0.0007403 0.0004207 0.0012402

> mean(apply(e6.1, 1, sd))

[1] 0.0009672
```

6.4 Process Capability Analysis

Example 6.3

```
> 1-pnorm(0.0092, mean=0.007966, sd=0.00103)

[1] 0.1154

> pnorm(0.0052, mean=0.007966, sd=0.00103)

[1] 0.003622

> 1-pnorm(0.0092, mean=0.007966, sd=0.00103) + pnorm(0.0052, mean=0.007966, sd=0.00103)

[1] 0.1191
```

Figure 6.6 on page 255 of Devore and Farnum.

```
> qcc(e6.1, type="R")

List of 11
 $ call      : language qcc(data = e6.1, type = "R")
 $ type      : chr "R"
 $ data.name : chr "e6.1"
 $ data      : num [1:20, 1:5] 0.0061 0.0088 0.008 0.0067 0.0087 0.0071 0.0078 0.0087 0.0074 0.0081 ...
  ..- attr(*, "dimnames")=List of 2
 $ statistics: Named num [1:20] 0.004 0.0029 0.0024 0.0024 0.001 0.0036 0.0024 0.0023 0.0013 0.0032 ...
  ..- attr(*, "names")= chr [1:20] "1" "2" "3" "4" ...
 $ sizes     : Named int [1:20] 5 5 5 5 5 5 5 5 5 5 ...
  ..- attr(*, "names")= chr [1:20] "1" "2" "3" "4" ...
 $ center    : num 0.0024
 $ std.dev   : num 0.00103
 $ nsigmas   : num 3
 $ limits    : num [1, 1:2] 0 0.00507
  ..- attr(*, "dimnames")=List of 2
 $ violations:List of 2
 - attr(*, "class")= chr "qcc"
```

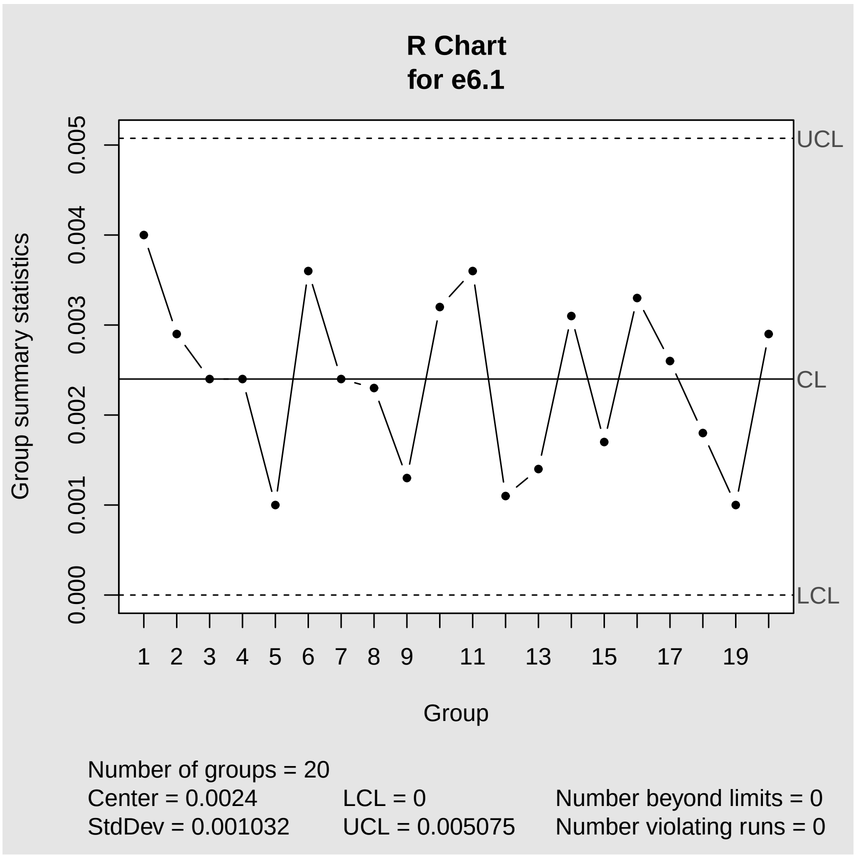


Figure 6.7 on page 255 of Devore and Farnum.

```
> qcc(e6.1, type="xbar", std.dev="UWAVE-R")

List of 11
 $ call      : language qcc(data = e6.1, type = "xbar", std.dev = "UWAVE-R")
 $ type      : chr "xbar"
 $ data.name : chr "e6.1"
 $ data      : num [1:20, 1:5] 0.0061 0.0088 0.008 0.0067 0.0087 0.0071 0.0078 0.0087 0.0074 0.0081 ...
  ..- attr(*, "dimnames")=List of 2
 $ statistics: Named num [1:20] 0.00682 0.0076 0.00798 0.00732 0.00878 0.0067 0.00774 0.00822 0.00822 0.00814
  ..- attr(*, "names")= chr [1:20] "1" "2" "3" "4" ...
 $ sizes     : Named int [1:20] 5 5 5 5 5 5 5 5 5 ...
  ..- attr(*, "names")= chr [1:20] "1" "2" "3" "4" ...
 $ center    : num 0.00797
 $ std.dev   : num 0.00103
 $ nsigmas   : num 3
 $ limits    : num [1, 1:2] 0.00658 0.00935
  ..- attr(*, "dimnames")=List of 2
 $ violations:List of 2
 - attr(*, "class")= chr "qcc"
```

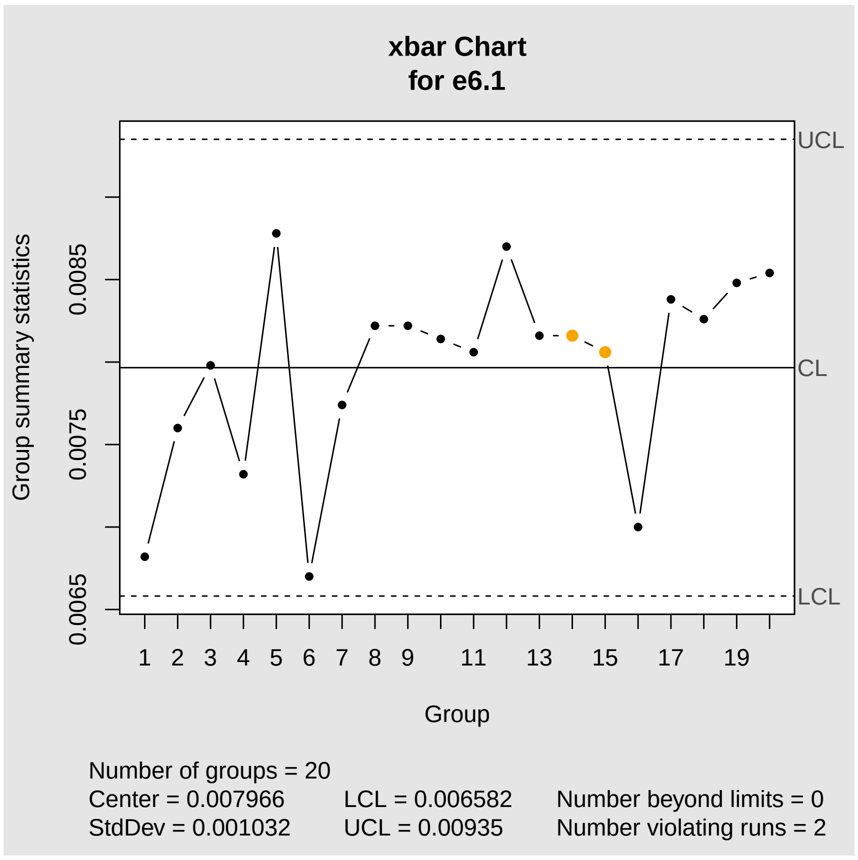


Figure 6.8 on page 257 of Devore and Farnum.

```
> qcc(e6.1, type="S")

List of 11
 $ call      : language qcc(data = e6.1, type = "S")
 $ type      : chr "S"
 $ data.name : chr "e6.1"
 $ data      : num [1:20, 1:5] 0.0061 0.0088 0.008 0.0067 0.0087 0.0071 0.0078 0.0087 0.0074 0.0081 ...
  ..- attr(*, "dimnames")=List of 2
 $ statistics: Named num [1:20] 0.001588 0.001102 0.000896 0.000942 0.000377 ...
  ..- attr(*, "names")= chr [1:20] "1" "2" "3" "4" ...
 $ sizes     : Named int [1:20] 5 5 5 5 5 5 5 5 5 5 ...
  ..- attr(*, "names")= chr [1:20] "1" "2" "3" "4" ...
 $ center    : num 0.000967
 $ std.dev   : num 0.00103
 $ nsigmas   : num 3
 $ limits    : num [1, 1:2] 0 0.00202
  ..- attr(*, "dimnames")=List of 2
 $ violations:List of 2
 - attr(*, "class")= chr "qcc"
```

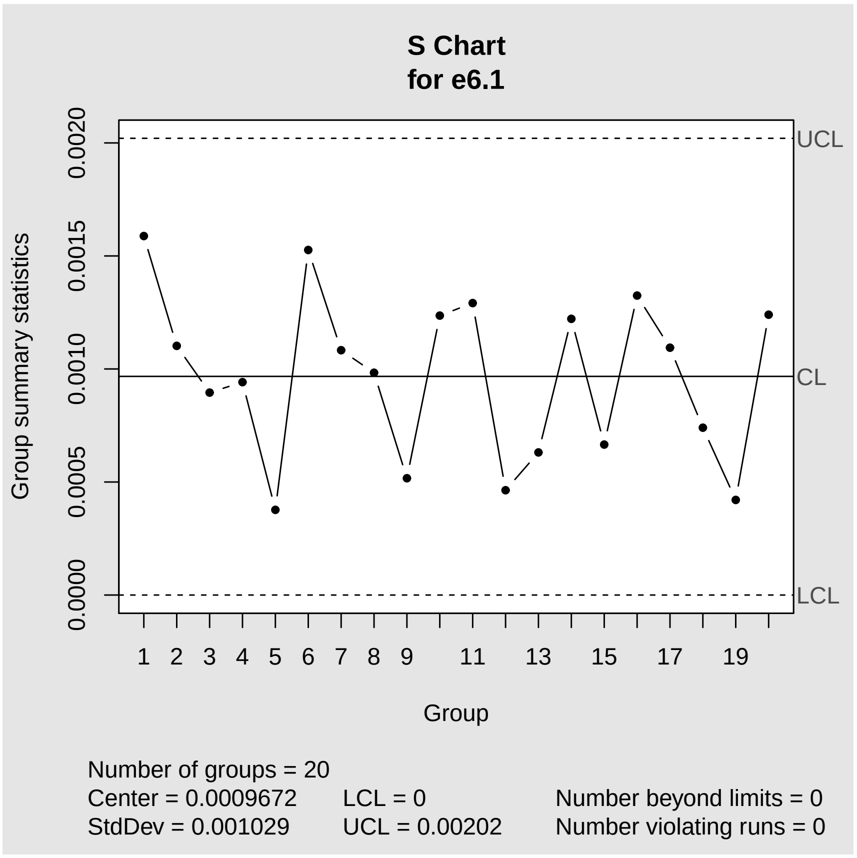
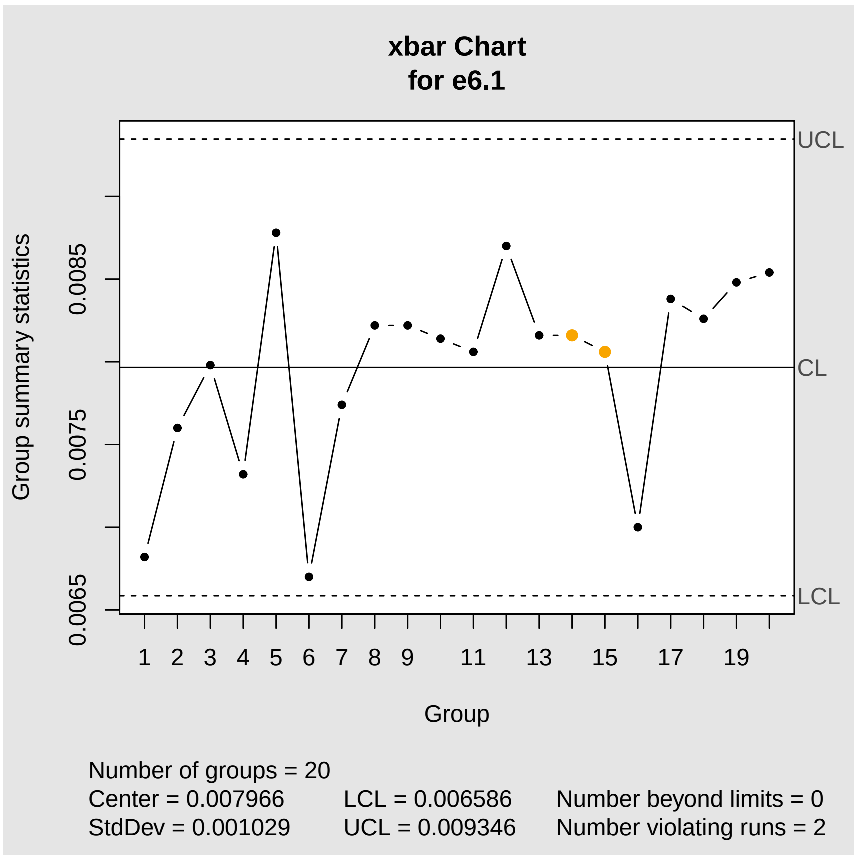


Figure 6.9 on page 258 of Devore and Farnum.

```
> qcc(e6.1, type="xbar", std.dev="UWAVE-SD")

List of 11
 $ call      : language qcc(data = e6.1, type = "xbar", std.dev = "UWAVE-SD")
 $ type      : chr "xbar"
 $ data.name : chr "e6.1"
 $ data      : num [1:20, 1:5] 0.0061 0.0088 0.008 0.0067 0.0087 0.0071 0.0078 0.0087 0.0074 0.0081 ...
  ..- attr(*, "dimnames")=List of 2
 $ statistics: Named num [1:20] 0.00682 0.0076 0.00798 0.00732 0.00878 0.0067 0.00774 0.00822 0.00822 0.00814
  ..- attr(*, "names")= chr [1:20] "1" "2" "3" "4" ...
 $ sizes     : Named int [1:20] 5 5 5 5 5 5 5 5 5 ...
  ..- attr(*, "names")= chr [1:20] "1" "2" "3" "4" ...
 $ center    : num 0.00797
 $ std.dev   : num 0.00103
 $ nsigmas   : num 3
 $ limits    : num [1, 1:2] 0.00659 0.00935
  ..- attr(*, "dimnames")=List of 2
 $ violations:List of 2
 - attr(*, "class")= chr "qcc"
```



Example 6.4

```
> abs((0.0092-0.0052) - 0.007966)/(6*0.00103)
```

```
[1] 0.6417
```

Example 6.5

```
> Cpl = (0.007966 - 0.0052) /(3*0.00103)
> Cpl
```

```
[1] 0.8951
```

```
> Cpu = (0.0092-0.007966) /(3*0.00103)
> Cpu
```

```
[1] 0.3994
```

```
>  Cpk = min(Cpl, Cpu)
> Cpk
```

```
[1] 0.3994
```

6.5 Control Charts for Attributes Data

Example 6.6

```
> data(e6.6)
> str(e6.6)
```

```
'data.frame':        30 obs. of  3 variables:
 $ Rejects: int  14 22 9 19 21 18 16 16 21 14 ...
 $ Tested : int  286 281 310 313 293 305 322 316 293 287 ...
 $ Prop   : num  0.049 0.078 0.029 0.061 0.072 0.059 0.05 0.051 0.072 0.049 ...
```

```
> BarP = sum(e6.6$Rejects) / sum(e6.6$Tested)
> BarP
```

```
[1] 0.05385
```

```
> LCL = BarP - 3* sqrt(BarP*(1-BarP)/e6.6$Tested)
> round(LCL,4)
```

```
 [1] 0.0138 0.0135 0.0154 0.0156 0.0143 0.0151 0.0161 0.0158 0.0143 0.0139
[11] 0.0152 0.0165 0.0145 0.0145 0.0158 0.0146 0.0136 0.0161 0.0158 0.0152
[21] 0.0158 0.0162 0.0150 0.0150 0.0162 0.0140 0.0147 0.0159 0.0156 0.0140
```

```
> UCL = BarP + 3* sqrt(BarP*(1-BarP)/e6.6$Tested)
> round(UCL,4)
```

```
 [1] 0.0939 0.0942 0.0923 0.0921 0.0934 0.0926 0.0916 0.0919 0.0934 0.0938
[11] 0.0925 0.0912 0.0932 0.0932 0.0919 0.0931 0.0941 0.0916 0.0919 0.0925
[21] 0.0919 0.0915 0.0927 0.0927 0.0915 0.0937 0.0930 0.0918 0.0921 0.0937
```

quantities not given by R but in the actual figure on p271

```
> mean(LCL)
```

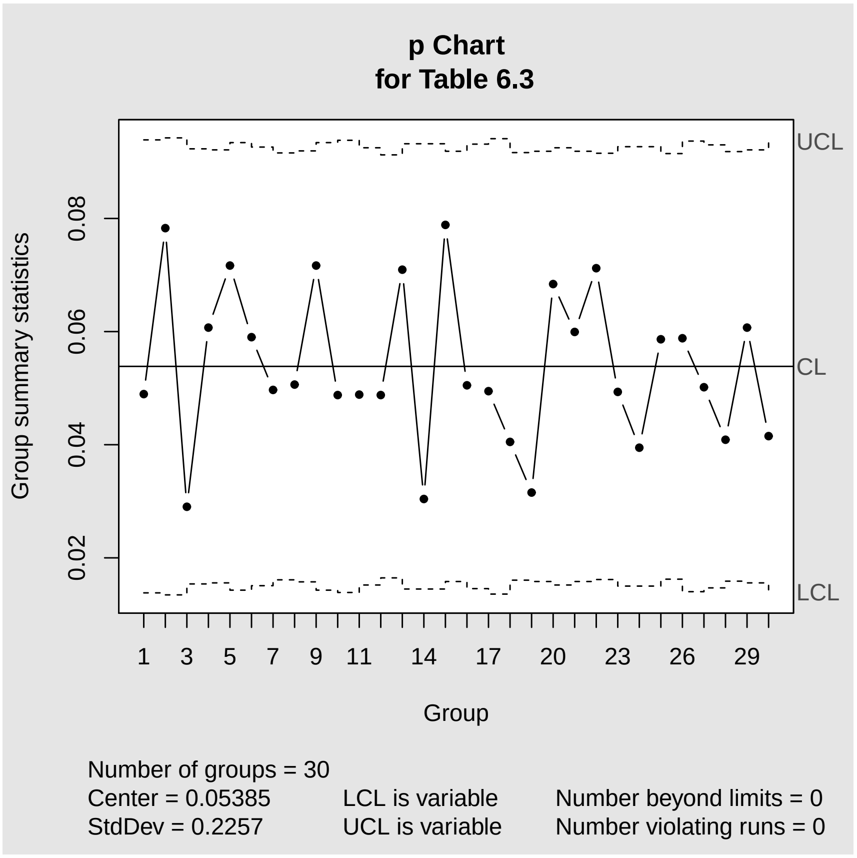
```
[1] 0.01506
```

```
> mean(UCL)
```

```
[1] 0.09264
```

Figure 6.13 on page 271 of Devore and Farnum.

```
> qcc(e6.6$Rejects, sizes=e6.6$Tested, type="p", data.name= "Table 6.3")

List of 11
 $ call      : language qcc(data = e6.6$Rejects, type = "p", sizes = e6.6$Tested, data.name = "Table 6.3")
 $ type      : chr "p"
 $ data.name : chr "Table 6.3"
 $ data      : int [1:30, 1] 14 22 9 19 21 18 16 16 21 14 ...
  ..- attr(*, "dimnames")=List of 2
 $ statistics: Named num [1:30] 0.049 0.0783 0.029 0.0607 0.0717 ...
  ..- attr(*, "names")= chr [1:30] "1" "2" "3" "4" ...
 $ sizes     : int [1:30] 286 281 310 313 293 305 322 316 293 287 ...
 $ center    : num 0.0539
 $ std.dev   : num 0.226
 $ nsigmas   : num 3
 $ limits    : num [1:30, 1:2] 0.0138 0.0135 0.0154 0.0156 0.0143 ...
  ..- attr(*, "dimnames")=List of 2
 $ violations:List of 2
 - attr(*, "class")= chr "qcc"
```

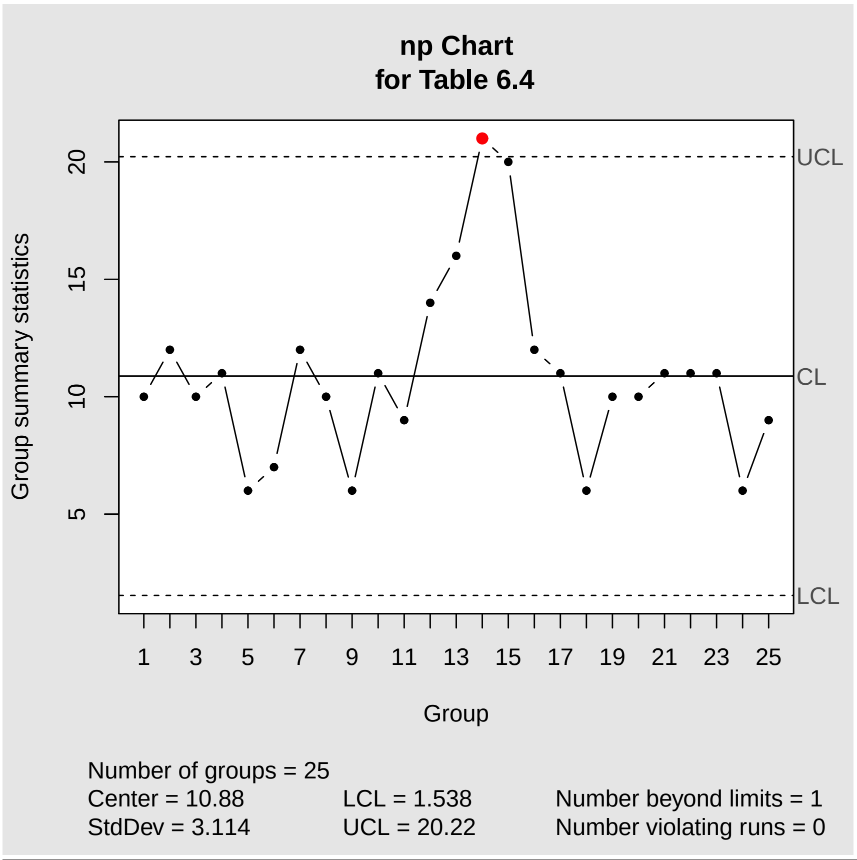


Example 6.7

```
> data(e6.7)
```

Figure 6.14 on page 273 of Devore and Farnum.

```
> qcc(e6.7$NoErrors, sizes=e6.7$SampleSize, type="np", data.name= "Table 6.4")

List of 11
 $ call      : language qcc(data = e6.7$NoErrors, type = "np", sizes = e6.7$SampleSize, data.name = "Ta
 $ type      : chr "np"
 $ data.name : chr "Table 6.4"
 $ data      : int [1:25, 1] 10 12 10 11 6 7 12 10 6 11 ...
  ..- attr(*, "dimnames")=List of 2
 $ statistics: Named int [1:25] 10 12 10 11 6 7 12 10 6 11 ...
  ..- attr(*, "names")= chr [1:25] "1" "2" "3" "4" ...
 $ sizes     : int [1:25] 100 100 100 100 100 100 100 100 100 100 ...
 $ center    : num 10.9
 $ std.dev   : num 3.11
 $ nsigmas   : num 3
 $ limits    : num [1, 1:2] 1.54 20.22
  ..- attr(*, "dimnames")=List of 2
 $ violations:List of 2
 - attr(*, "class")= chr "qcc"
```

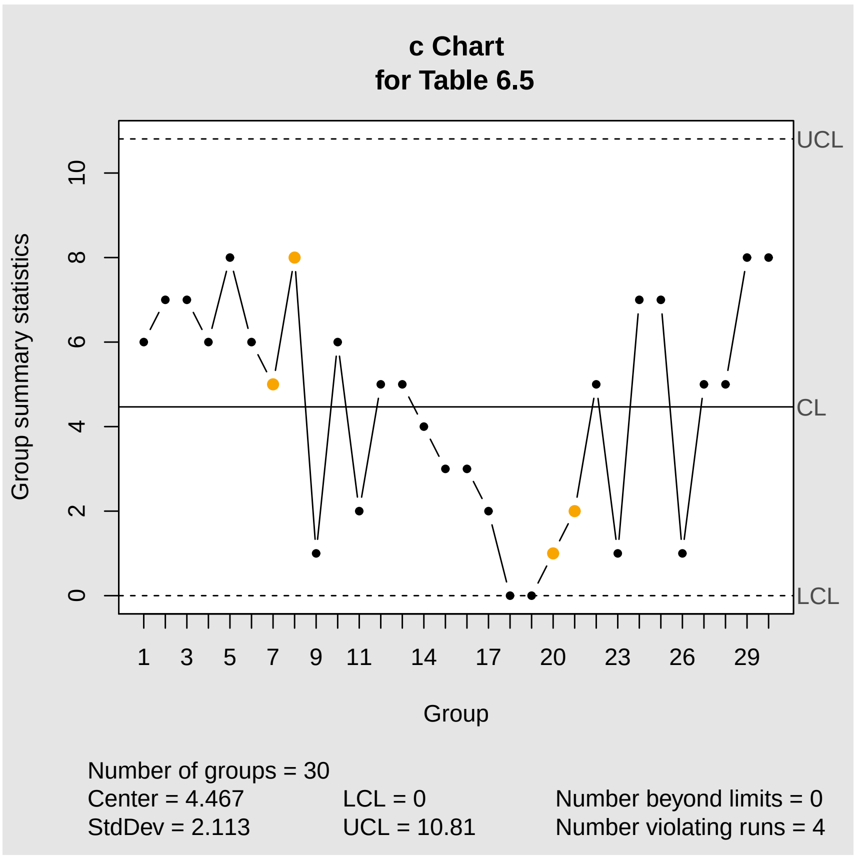


Example 6.8

```
> data(e6.8)
```

Figure 6.15 on page 275 of Devore and Farnum.

```
> qcc(e6.8$Errors, type="c", data.name= "Table 6.5")

List of 11
 $ call      : language qcc(data = e6.8$Errors, type = "c", data.name = "Table 6.5")
 $ type      : chr "c"
 $ data.name : chr "Table 6.5"
 $ data      : int [1:30, 1] 6 7 7 6 8 6 5 8 1 6 ...
  ..- attr(*, "dimnames")=List of 2
 $ statistics: Named int [1:30] 6 7 7 6 8 6 5 8 1 6 ...
  ..- attr(*, "names")= chr [1:30] "1" "2" "3" "4" ...
 $ sizes     : int [1:30] 1 1 1 1 1 1 1 1 1 1 ...
 $ center    : num 4.47
 $ std.dev   : num 2.11
 $ nsigmas   : num 3
 $ limits    : num [1, 1:2] 0 10.8
  ..- attr(*, "dimnames")=List of 2
 $ violations:List of 2
 - attr(*, "class")= chr "qcc"
```

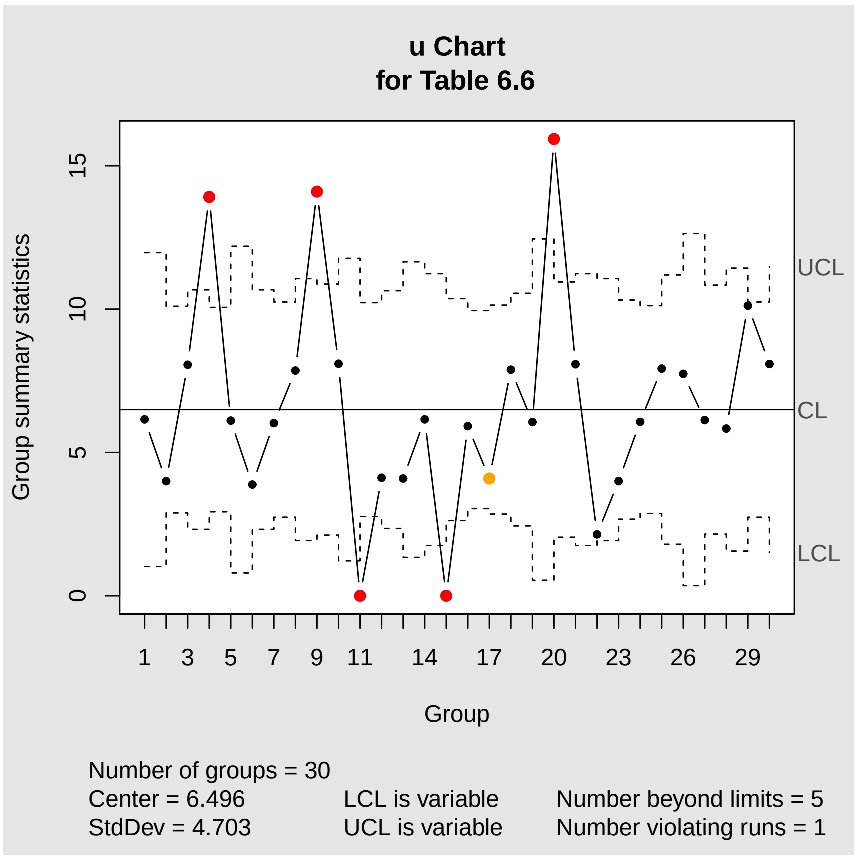


Example 6.9

```
> data(e6.9)
```

Figure 6.16 on page 277 of Devore and Farnum.

```
> qcc(e6.9$Flaws, sizes = e6.9$Units, type="u", data.name= "Table 6.6")

List of 11
 $ call      : language qcc(data = e6.9$Flaws, type = "u", sizes = e6.9$Units, data.name = "Table 6.6")
 $ type      : chr "u"
 $ data.name : chr "Table 6.6"
 $ data      : int [1:30, 1] 12 18 27 64 11 13 25 22 43 17 ...
  ..- attr(*, "dimnames")=List of 2
 $ statistics: Named num [1:30] 6.15 4 8.06 13.91 6.11 ...
  ..- attr(*, "names")= chr [1:30] "1" "2" "3" "4" ...
 $ sizes     : num [1:30] 1.95 4.5 3.35 4.6 1.8 3.35 4.15 2.8 3.05 2.1 ...
 $ center    : num 6.5
 $ std.dev   : num 4.7
 $ nsigmas   : num 3
 $ limits    : num [1:30, 1:2] 1.02 2.891 2.318 2.931 0.797 ...
  ..- attr(*, "dimnames")=List of 2
 $ violations:List of 2
 - attr(*, "class")= chr "qcc"
```



6.6 Reliability

Example 6.10

R is not required

Example 6.11

R is not required

Example 6.12

R is not required

Chapter 7

Estimation and Statistical Intervals

Before you can do any examples in this chapter, you will need to execute the command

```
> library(DevFarn2)
```

This will then give you direct access to the data for each example, and a few extra functions needed for some chapters.

Note that you must issue this command as the first act in any R session.

7.1 Point Estimation

Example 7.1

```
> floor(100*250/25)
```

```
[1] 1000
```

Example 7.2

This example does not require R.

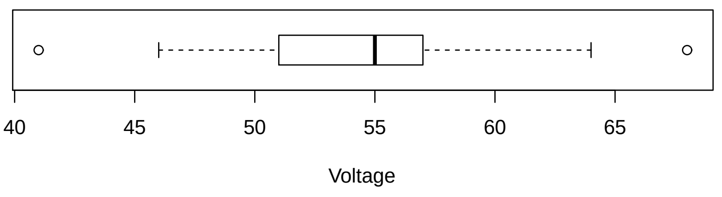
7.2 Large-Sample Confidence Intervals for a Population Mean

Example 7.3

```
> data(e7.3)
```

Figure 7.4 on page 297 of Devore and Farnum.

```
> boxplot(e7.3, horizontal=T, xlab="Voltage")
> windows(7,5)
```



```
> mean(e7.3)

[1] 54.71

> sd(e7.3)

[1] 5.231

> t.test(e7.3)

        One Sample t-test

data:  e7.3
t = 72, df = 47, p-value <2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 53.19 56.23
sample estimates:
mean of x
    54.71
```

Ignore some output here! it is explained in other examples.

Example 7.4

This example does not require R.

Example 7.5

This example does not require R.

7.3 More Large-Sample Confidence Intervals

Example 7.6

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

```
        1-sample proportions test with continuity correction

data:  16 out of 48, null probability 0.5
X-squared = 4.7, df = 1, p-value = 0.03
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.2081 0.4851
sample estimates:
     p
0.3333
```

ignore some output here! it is explained in other examples.

Example 7.7

This example does not require R.

Example 7.8

This example does not require R.

Example 7.9

No R function exists for summary data like this so we use a function from a contributed package. We've taken the function from the *BSDA* package and added it to the *DevFarn2* package for you.

```
> zsum.test(mean.x = 4.25, sigma.x = 1.3, n.x = 78, mean.y = 7.14, sigma.y = 1.68, n.y = 88, alternativ

        Two-sample z-Test

data:  Summarized x and y
z = -12, p-value <2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.344 -2.436
sample estimates:
mean of x mean of y
     4.25      7.14
```

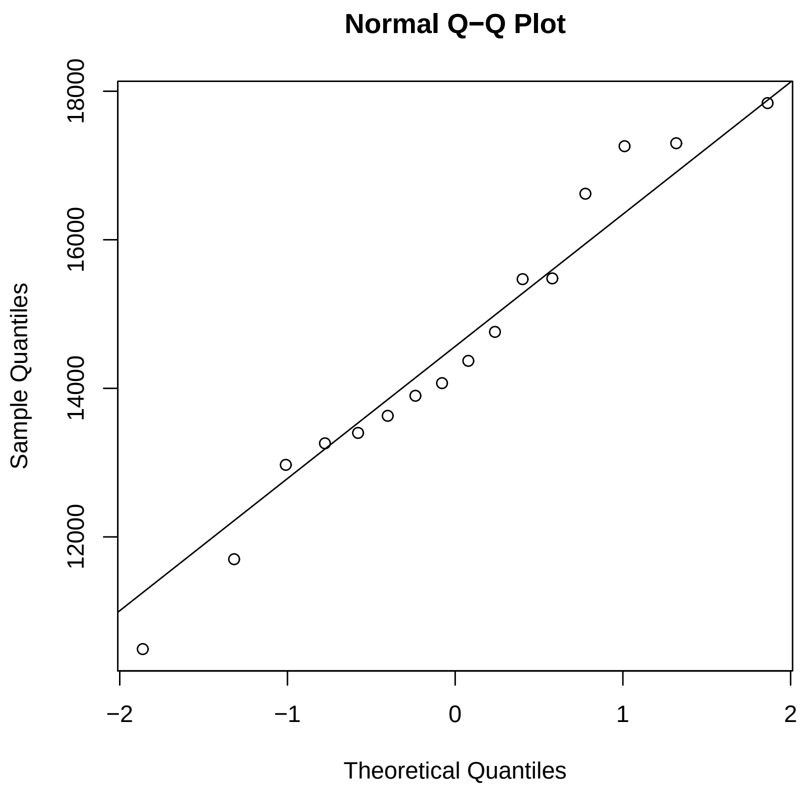
7.4 Small-Sample Intervals Based on a Normal Population Distribution

Example 7.10

```
> data(e7.10)
```

Figure 7.10 on page 316 of Devore and Farnum.

```
> qqnorm(e7.10)
> qqline(e7.10)
```



```
> e7.10adj = e7.10 -10000
> mean(e7.10adj)

[1] 4532

> sd(e7.10adj)

[1] 2056

> mean(e7.10)

[1] 14532

> sd(e7.10)

[1] 2056

> qt(c(0.025,0.975), 15)

[1] -2.131  2.131
```

Working for adjusted data is not really required when using software.

```
> t.test(e7.10)
```

```
        One Sample t-test

data:  e7.10
t = 28, df = 15, p-value = 2e-14
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 13437 15628
sample estimates:
mean of x
    14532
```

Then one-sided is found using

```
> qt(0.05, 15)

[1] -1.753

> t.test(e7.10, alternative="g")

        One Sample t-test

data:  e7.10
t = 28, df = 15, p-value = 1e-14
alternative hypothesis: true mean is greater than 0
95 percent confidence interval:
 13632   Inf
sample estimates:
mean of x
    14532
```

Example 7.11

This example is not required in 161.100

```
> mean(e7.10) + qt(c(0.025,0.975), 15)*sd(e7.10)*sqrt(1+1/16)

[1] 10016 19049
```

Example 7.12

This example is not required in 161.100

7.5 Intervals for $\mu_1 - \mu_2$ Based on Normal Population Distributions

Example 7.13

This example uses another function from the *BSDA* package. See Example 7.9 above.

```
> tsum.test(mean.x = 51.71, s.x = 0.79, n.x = 10, mean.y = 136.14, s.y = 3.59, n.y = 10, alternative = "two.si

        Welch Modified Two-Sample t-Test

data:  Summarized x and y
t = -73, df = 9.9, p-value = 8e-15
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -87.02 -81.84
sample estimates:
mean of x mean of y
    51.71    136.14
```

Note some rounding errors in manual working

Example 7.14

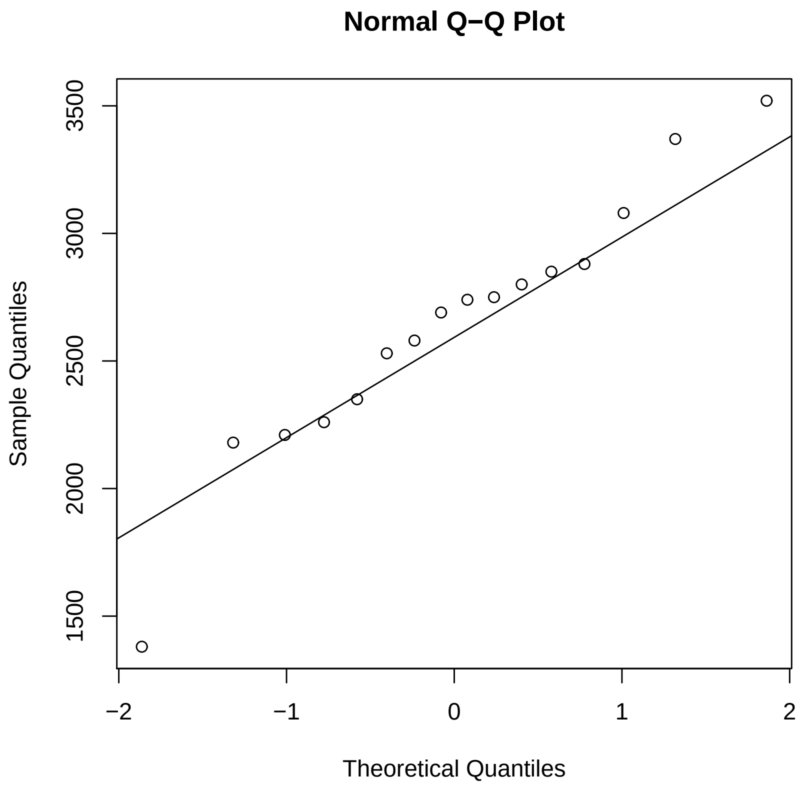
```
> data(e7.14)
> str(e7.14)

'data.frame':        16 obs. of  3 variables:
 $ Min1      : int  10490 16620 17300 15480 12970 17260 13400 13900 13630 13260 ...
 $ Weeks4    : int  9110 13250 14720 12740 10120 14570 11220 11100 11420 10910 ...
 $ Difference: int  1380 3370 2580 2740 2850 2690 2180 2800 2210 2350 ...


> qt(c(0.005,0.995),15)

[1] -2.947  2.947

> mean(e7.14$Difference)

[1] 2636

> sd(e7.14$Difference)

[1] 508.6

> t.test(e7.14$Difference, conf.level=0.99)

        One Sample t-test

data:  e7.14$Difference
t = 21, df = 15, p-value = 2e-12
alternative hypothesis: true mean is not equal to 0
99 percent confidence interval:
 2261 3010
sample estimates:
mean of x
     2636
```

Or alternatively... the incorrect way of specifying the two samples is:

```
> t.test(e7.14$Min1, e7.14$Weeks4, conf.level=0.99)
```

Figure 7.11 on page 325 of Devore and Farnum.

```
> qqnorm(e7.14$Difference)
> qqline(e7.14$Difference)
```



```
        Welch Two Sample t-test

data:  e7.14$Min1 and e7.14$Weeks4
t = 3.8, df = 30, p-value = 7e-04
alternative hypothesis: true difference in means is not equal to 0
99 percent confidence interval:
  709.5 4561.8
sample estimates:
mean of x mean of y
    14532     11897
```

the correct way is...

```
> t.test(e7.14$Min1, e7.14$Weeks4, conf.level=0.99, paired=TRUE)

        Paired t-test

data:  e7.14$Min1 and e7.14$Weeks4
t = 21, df = 15, p-value = 2e-12
alternative hypothesis: true difference in means is not equal to 0
99 percent confidence interval:
 2261 3010
sample estimates:
mean of the differences
                   2636
```

7.6 Other Topics in Estimation (Optional)

Examples 7.15

This example does not require R.

Examples 7.16

This example does not require R.

Examples 7.17

This example does not require R.

Examples 7.18

This example does not require R.

Examples 7.19

This example does not require R.

Examples 7.20

This example does not require R.

Example 7.21

This material is not covered in 161.100

Chapter 8

Testing Statistical Hypotheses

Before you can do any examples in this chapter, you will need to execute the command

```
> library(DevFarn2)
```

This will then give you direct access to the data for each example, and a few extra functions needed for some chapters.

Note that you must issue this command as the first act in any R session.

8.1 Hypotheses and Test Procedures

Example 8.1

This example does not require R.

Example 8.2

This example does not require R.

Example 8.3

This example does not require R.

Example 8.4

```
> (11.3-15)/( 6.43/sqrt(115))

[1] -6.171

> pnorm((11.3-15)/( 6.43/sqrt(115)))
```

```
[1] 3.398e-10
```

Or alternatively…

```
> pnorm(11.3, mean=15, sd=6.43/sqrt(115))
```

```
[1] 3.398e-10
```

8.2 Tests Concerning Hypotheses About Means

Example 8.5

```
> pt(1, df=4, lower.tail=F)
```

```
[1] 0.187
```

Example 8.6

```
> data(e8.6)
> e8.6
```

```
   NoFusion Fused
1      2748  3027
2      2700  3356
3      2655  3359
4      2822  3297
5      2511  3125
6      3149  2910
7      3257  2889
8      3213  2902
9      3220    NA
10     2753    NA
```

note that the second set of numbers is shorter than the first. we want access to the two sets separately

```
> attach(e8.6)
> t.test(NoFusion, Fused)
```

```
        Welch Two Sample t-test

data:  NoFusion and Fused
t = -1.8, df = 16, p-value = 0.09
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -446.97   36.32
sample estimates:
mean of x mean of y
     2903      3108
```

```
> detach(e8.6)
```

The detach command undoes the attach just above. - not essential but good practice.

Example 8.7

```
> data(e8.7)
> attach(e8.7)
> t.test(Before, After, paired=TRUE)

        Paired t-test

data:  Before and After
t = 3.3, df = 15, p-value = 0.005
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  2.362 11.138
sample estimates:
mean of the differences
                   6.75

> t.test(Before, After, mu=5, alternative="g", paired=TRUE)

        Paired t-test

data:  Before and After
t = 0.85, df = 15, p-value = 0.2
alternative hypothesis: true difference in means is greater than 5
95 percent confidence interval:
 3.141   Inf
sample estimates:
mean of the differences
                   6.75

> detach(e8.7)
```

8.3 Tests Concerning Hypotheses About a Categorical Population

Example 8.8

```
> Obsvd = c(11, 24, 69, 96)
> Obsvd

[1] 11 24 69 96

> Exptd = 200 * c(15, 46, 120, 184)/365
> Exptd

[1]   8.219  25.205  65.753 100.822

> (Obsvd-Exptd)^2 / Exptd

[1] 0.94084 0.05765 0.16030 0.23061

> sum((Obsvd-Exptd)^2 / Exptd)

[1] 1.389

> pchisq(sum((Obsvd-Exptd)^2 / Exptd),3, lower.tail=F)

[1] 0.708
```

Example 8.9

```
> data(e8.9)
> chisq.test(e8.9)

        Pearson's Chi-squared test

data:  e8.9
X-squared = 14, df = 8, p-value = 0.08
```

8.4 Testing the Form of a Distribution

Example 8.10

This test was not in version 2.9.1 or earlier versions of R and must be substituted by another test - many exist!

```
> data(e2.18)
> e2.18$X1

 [1] 24.46 25.61 26.25 26.42 26.66 27.15 27.31 27.54 27.74 27.94 27.98
[12] 28.04 28.28 28.49 28.50 28.87 29.11 29.13 29.50 30.88

> shapiro.test(e2.18$X1)

        Shapiro-Wilk normality test

data:  e2.18$X1
W = 0.99, p-value = 1
```

but should be used in conjunction with the plot similar to

Example 8.11

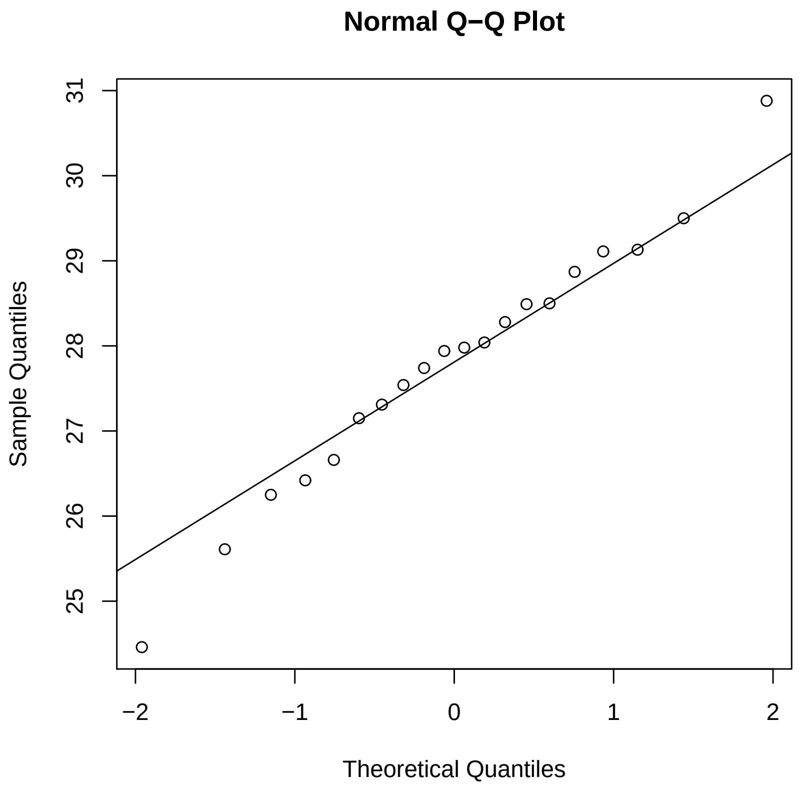
```
> Observed = c(9,9,10,14,6)
> Lambda=2.1
> PoissonProbs = exp(-Lambda)*Lambda^(0:3) / factorial(0:3)
```

Note this is only the first four categories fifth category is the left overs.

```
> Expected = 48* c(PoissonProbs, (1-sum(PoissonProbs)))
> Expected

[1]  5.878 12.344 12.961  9.073  7.745

> (Observed-Expected)^2/Expected

[1] 1.6583 0.9057 0.6764 2.6762 0.3932

> sum((Observed-Expected)^2/Expected)

[1] 6.31

> pchisq(sum((Observed-Expected)^2/Expected), df=3, lower.tail=FALSE)

[1] 0.09747
```

Figure 8.10 on page 384 of Devore and Farnum.

```
> qqnorm(e2.18$X1)
> qqline(e2.18$X1)
```



8.5 Further Aspects of Hypothesis Testing

Example 8.12

The working is not particulary relevant to the purpose of the example

Example 8.13

This example is not required for 161.100

Chapter 9

The Analysis of Variance

9.1 Terminology and Concepts

This section has no examples.

9.2 Single-Factor ANOVA

Example 9.1

This example does not need R.

9.3 Interpreting ANOVA Results

Example 9.2

This example does not need R.

Example 9.3

This example does not need R.

Example 9.4

This example does not need R.

Example 9.5

```
> WaveTimes = c(55,53,54,26,37,32,78,91,85,92,100,96,49,51,50,80,85,83)
> Rail = rep(1:6, each=3)
```

to chekc that the variables line up

```
> Rail
```

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

for illustration only.

```
> aov(WaveTimes~Rail)
```

```
Call:
   aov(formula = WaveTimes ~ Rail)

Terms:
                Rail Residuals
Sum of Squares  1884      7620
Deg. of Freedom    1        16

Residual standard error: 21.82
Estimated effects may be unbalanced
```

better to do

```
> summary(aov(WaveTimes~Rail))
```

```
            Df Sum Sq Mean Sq F value Pr(>F)
Rail         1   1884    1884    3.96  0.064 .
Residuals   16   7620     476
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

but NOTE the df for Rail are wrong. At present Rail is not a factor!!!

```
> Rail = as.factor(paste("Rail", Rail))
> Rail
```

```
 [1] Rail 1 Rail 1 Rail 1 Rail 2 Rail 2 Rail 2 Rail 3 Rail 3 Rail 3 Rail 4
[11] Rail 4 Rail 4 Rail 5 Rail 5 Rail 5 Rail 6 Rail 6 Rail 6
Levels: Rail 1 Rail 2 Rail 3 Rail 4 Rail 5 Rail 6
```

```
> summary(aov(WaveTimes~Rail))
```

```
            Df Sum Sq Mean Sq F value Pr(>F)
Rail         5   9310    1862     115  1e-09 ***
Residuals   12    194      16
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

9.4 Randomized Block Experiments

Example 9.6

```
> Yield=c(11,12,10,10,10,9,10,10,10,12,12,12,10,10,10,9,10,10,10,12,14,15,12,13,12,12,13,13,14,16,12,13,10,10
> Block = paste("Plot", rep(1:10,4))
> Block

 [1] "Plot 1"  "Plot 2"  "Plot 3"  "Plot 4"  "Plot 5"  "Plot 6"  "Plot 7"
 [8] "Plot 8"  "Plot 9"  "Plot 10" "Plot 1"  "Plot 2"  "Plot 3"  "Plot 4"
[15] "Plot 5"  "Plot 6"  "Plot 7"  "Plot 8"  "Plot 9"  "Plot 10" "Plot 1"
[22] "Plot 2"  "Plot 3"  "Plot 4"  "Plot 5"  "Plot 6"  "Plot 7"  "Plot 8"
[29] "Plot 9"  "Plot 10" "Plot 1"  "Plot 2"  "Plot 3"  "Plot 4"  "Plot 5"
[36] "Plot 6"  "Plot 7"  "Plot 8"  "Plot 9"  "Plot 10"

> Variety = paste("V", rep(1:4, each=10))
> Variety

 [1] "V 1" "V 1" "V 1" "V 1" "V 1" "V 1" "V 1" "V 1" "V 1" "V 1" "V 2"
[12] "V 2" "V 2" "V 2" "V 2" "V 2" "V 2" "V 2" "V 2" "V 2" "V 3" "V 3"
[23] "V 3" "V 3" "V 3" "V 3" "V 3" "V 3" "V 3" "V 3" "V 4" "V 4" "V 4"
[34] "V 4" "V 4" "V 4" "V 4" "V 4" "V 4" "V 4"

> summary(aov(Yield~Block+Variety))

            Df Sum Sq Mean Sq F value  Pr(>F)
Block        9   49.1    5.46    23.4 1.6e-10 ***
Variety      3   58.2   19.40    83.1 9.3e-14 ***
Residuals   27    6.3    0.23
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Chapter 10

Experimental Design

Before you can do any examples in this chapter, you will need to execute the command

```
> library(DevFarn2)
```

This will then give you direct access to the data for each example, and a few extra functions needed for some chapters.

Note that you must issue this command as the first act in any R session.

10.1 Terminology and Concepts

There are no examples in this section.

10.2 Two-Factor Designs

Example 10.1

```
> data(e10.1)
> str(e10.1)
```

```
'data.frame':	30 obs. of  3 variables:
 $ Brand    : Factor w/ 5 levels "1","2","3","4",..: 1 1 1 1 1 1 2 2 2 2 ...
 $ Material : Factor w/ 3 levels "1","2","3": 1 1 2 2 3 3 1 1 2 2 ...
 $ Vibration: num  13.1 13.2 15 14.8 14 14.3 16.3 15.8 15.7 16.4 ...
```

We need to check the type of variable that R thinks Brand and Material are. If R had thought they were numeric rather than factors, we would get the wrong model!!!

```
> summary(aov(Vibration~Brand*Material, data=e10.1))
```

```
               Df Sum Sq Mean Sq F value  Pr(>F)
Brand           4   36.7    9.17   82.35 5.1e-10 ***
Material        2    0.7    0.35    3.16   0.071 .
Brand:Material  8   11.6    1.45   13.03 1.8e-05 ***
Residuals      15    1.7    0.11
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

If the degrees of freedom were 1 for a term in the model then you might try code like the following, which forces Brand and Material variables to be converted into factors.

```
> e10.1$Brand = as.factor(e10.1$Brand)
> e10.1$Material = as.factor(e10.1$Material)
```

and then run the model to get

```
> summary(aov(Vibration~Brand*Material, data=e10.1))
```

```
               Df Sum Sq Mean Sq F value  Pr(>F)
Brand           4   36.7    9.17   82.35 5.1e-10 ***
Material        2    0.7    0.35    3.16   0.071 .
Brand:Material  8   11.6    1.45   13.03 1.8e-05 ***
Residuals      15    1.7    0.11
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In this instance, the data was imported correctly, but always look out for this problem to arise. In fact, it happens so frequently, that I prefer to make the conversion as part of my standard practice before I fit the first model.

10.3 Multifactor Designs

Example 10.2

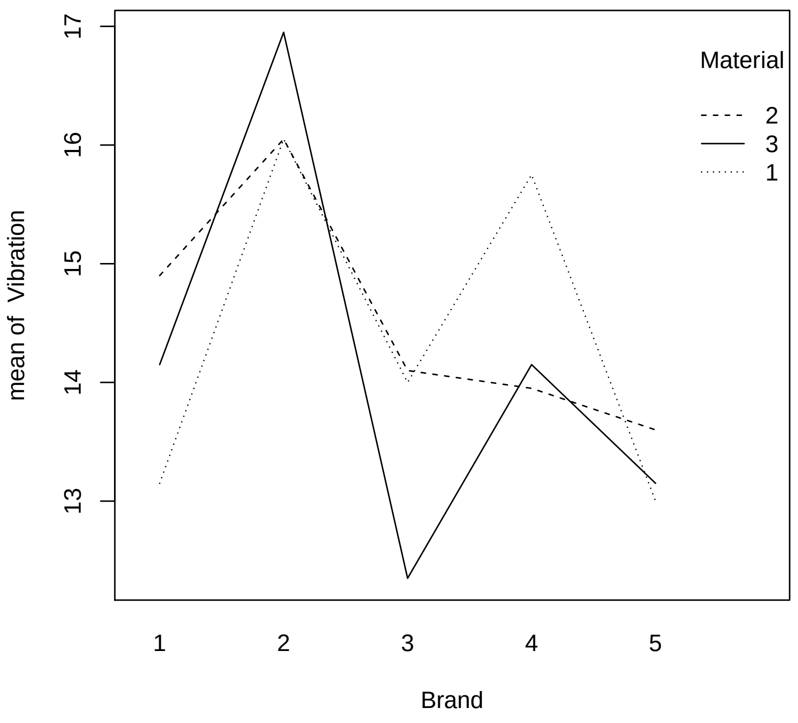
```
> data(e10.2)
> str(e10.2)
```

```
'data.frame':        54 obs. of  4 variables:
 $ Temp        : Factor w/ 3 levels "8","50","75": 1 1 1 1 1 1 2 2 2 2 ...
 $ Denier      : Factor w/ 3 levels "420","630","840": 1 1 1 1 1 1 1 1 1 1 ...
 $ Pressure    : Factor w/ 3 levels "17.2","34.4",..: 1 1 2 2 3 3 1 1 2 2 ...
 $ Permeability: int  73 80 157 155 332 322 52 51 125 118 ...
```

```
> e10.2$Temp = as.factor(e10.2$Temp)
> e10.2$Denier = as.factor(e10.2$Denier)
> e10.2$Pressure = as.factor(e10.2$Pressure)
> str(e10.2)
```

Figure 10.14 on page 449 of Devore and Farnum.

```
> attach(e10.1)
> interaction.plot(Vibration, x.factor=Brand, trace.factor=Material)
> detach(e10.1)
```



```
'data.frame':        54 obs. of  4 variables:
 $ Temp        : Factor w/ 3 levels "8","50","75": 1 1 1 1 1 1 2 2 2 2 ...
 $ Denier      : Factor w/ 3 levels "420","630","840": 1 1 1 1 1 1 1 1 1 1 ...
 $ Pressure    : Factor w/ 3 levels "17.2","34.4",..: 1 1 2 2 3 3 1 1 2 2 ...
 $ Permeability: int  73 80 157 155 332 322 52 51 125 118 ...
```

```
> summary(aov(Permeability~Temp*Pressure*Denier, data=e10.2))
```

```
                     Df Sum Sq Mean Sq F value  Pr(>F)
Temp                  2  33494   16747   455.8 < 2e-16 ***
Pressure              2 517207  258604  7038.6 < 2e-16 ***
Denier                2 107502   53751  1463.0 < 2e-16 ***
Temp:Pressure         4  11472    2868    78.1 1.9e-14 ***
Temp:Denier           4   7044    1761    47.9 7.0e-12 ***
Pressure:Denier       4   9654    2414    65.7 1.6e-13 ***
Temp:Pressure:Denier  8   6673     834    22.7 4.3e-10 ***
Residuals            27    992      37
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

this example carries on further than is required for 161.100

Example 10.3

This example does not need R

10.4 2^k Designs

Example 10.4

This example does not need R

Example 10.5

This example does not need R

Example 10.6

```
> data(e10.6)
> str(e10.6)

'data.frame':        16 obs. of  8 variables:
 $ X  : int  1 2 3 4 5 6 7 8 9 10 ...
 $ A  : int  -1 1 -1 1 -1 1 -1 1 -1 1 ...
 $ B  : int  -1 -1 1 1 -1 -1 1 1 -1 -1 ...
 $ C  : int  -1 -1 -1 -1 1 1 1 1 -1 -1 ...
 $ D  : int  -1 -1 -1 -1 -1 -1 -1 -1 1 1 ...
 $ BC : int  1 1 -1 -1 -1 -1 1 1 1 1 ...
 $ ABC: int  -1 1 1 -1 1 -1 -1 1 -1 1 ...
 $ y  : num  27.2 25.2 23.2 18.9 25.3 ...
```

the effects we want can be easily obtained via

```
> Effects = (2*coef(lm(y~A*B*C*D, data=e10.6)))[-1]
> Effects

      A       B       C       D     A:B     A:C     B:C     A:D     B:D
-4.2300 -0.8675  0.4975 18.8675 -1.7250 -0.5950  3.0775 -0.3200  1.9275
    C:D   A:B:C   A:B:D   A:C:D   B:C:D A:B:C:D
 0.4075 -0.1500 -0.1850  0.1500  0.7475  0.2550
```

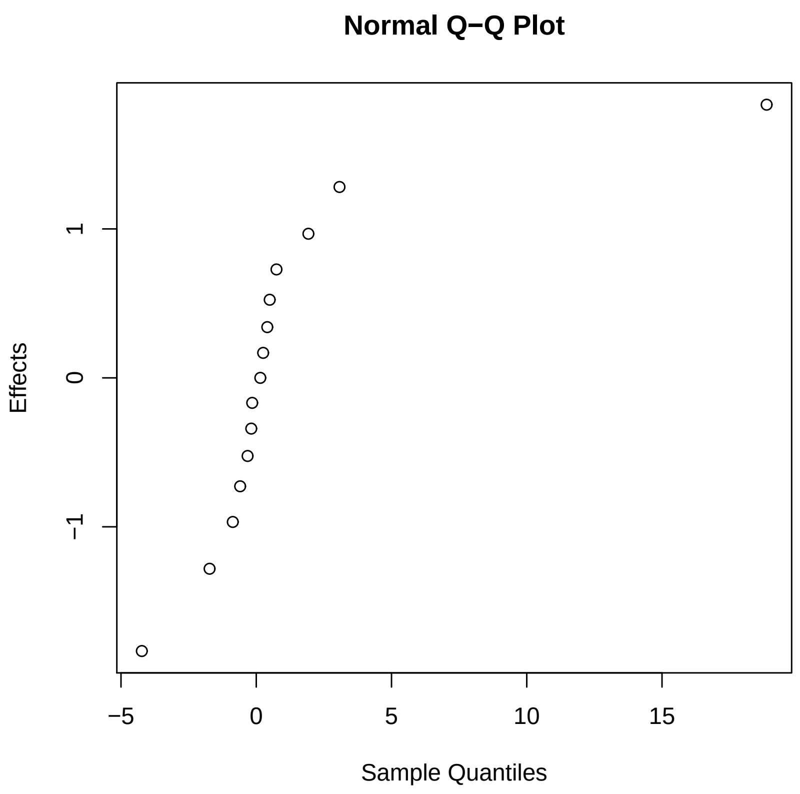
Example 10.7

getting text labels on the plot is difficult and need not be attempted. The following command will sort out the order in which the effects are plotted.

```
> sort(Effects)

      A     A:B       B     A:C     A:D   A:B:D   A:B:C   A:C:D A:B:C:D
-4.2300 -1.7250 -0.8675 -0.5950 -0.3200 -0.1850 -0.1500  0.1500  0.2550
    C:D       C   B:C:D     B:D     B:C       D
 0.4075  0.4975  0.7475  1.9275  3.0775 18.8675
```

Figure 10.21 on page 466 of Devore and Farnum.

```
> qqnorm(Effects, datax=TRUE, xlab="Effects")
```



Example 10.8

This example is not required in 161.100

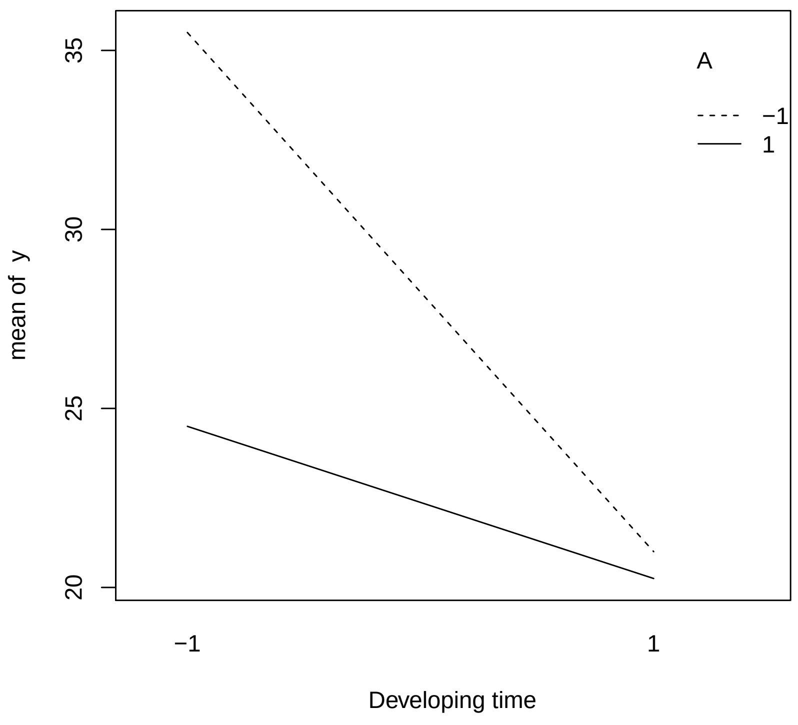
Example 10.9

```
> data(e10.9)
> str(e10.9)

'data.frame':        16 obs. of  5 variables:
 $ Run: int  1 2 3 4 5 6 7 8 1 2 ...
 $ A  : int  -1 1 -1 1 -1 1 -1 1 -1 1 ...
 $ B  : int  -1 -1 1 1 -1 -1 1 1 -1 -1 ...
 $ C  : int  -1 -1 -1 -1 1 1 1 1 -1 -1 ...
 $ y  : int  34 26 33 21 24 23 19 18 40 29 ...

> Effects = (2*coef(lm(y~A*B*C, data=e10.9)))[-1]
> Effects

     A      B      C    A:B    A:C    B:C  A:B:C
-5.875 -4.625 -9.375 -0.625  5.125 -0.125  0.875

> anova(lm(y~A*B*C, data=e10.9))
```

```
Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value  Pr(>F)
A          1    138     138   41.68 0.00020 ***
B          1     86      86   25.83 0.00095 ***
C          1    352     352  106.13 6.8e-06 ***
A:B        1      2       2    0.47 0.51162
A:C        1    105     105   31.72 0.00049 ***
B:C        1      0       0    0.02 0.89414
A:B:C      1      3       3    0.92 0.36445
Residuals  8     26       3
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 10.22 on page 470 of Devore and Farnum.

```
> attach(e10.9)
> interaction.plot(y, x.factor=C, trace.factor=A, xlab="Developing time")
> detach(e10.9)
```



note that the data used in our working and for creating the figure are the Yates coding values not the actual variable settings used in the experiment. It is of course the contents of the graph that are important.

we now see that the figure and description in the text are wrong. Both A and C should be set high according to the sign of their effects (Remember we want to minimize here) but the interaction limits the gain of having both set at the same position.

Example 10.10

using the variables as supplied in the dataset

```
> summary(lm(y~A+B+C+A:C, data=e10.9))

Call:
lm(formula = y ~ A + B + C + A:C, data = e10.9)

Residuals:
   Min     1Q Median     3Q    Max 
-3.812 -0.594 -0.062  0.500  2.188 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   25.312      0.421   60.13  3.3e-15 ***
A             -2.938      0.421   -6.98  2.3e-05 ***
B             -2.313      0.421   -5.49  0.00019 ***
C             -4.688      0.421  -11.14  2.5e-07 ***
A:C            2.562      0.421    6.09  7.9e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Residual standard error: 1.68 on 11 degrees of freedom
Multiple R-squared:  0.956,	Adjusted R-squared:  0.94 
F-statistic:   60 on 4 and 11 DF,  p-value: 2.12e-07
```

to use indicator variables (discussed in Chapter11 more thoroughly) we will use calculations although other approaches do exist.

```
> IndA = (e10.9$A + 1)/2
> IndB = (e10.9$B + 1)/2
> IndC = (e10.9$C + 1)/2
> IndAC = (e10.9$A*e10.9$C + 1)/2
```

in these cases the -1 became a 0 and the 1 stayed a 0 for all variables. Note the interaction is showing that A and C are set the same.

then the model is found using

```
> summary(lm(e10.9$y~IndA+IndB+IndC+IndAC))

Call:
lm(formula = e10.9$y ~ IndA + IndB + IndC + IndAC)

Residuals:
   Min     1Q Median     3Q    Max 
-3.812 -0.594 -0.062  0.500  2.188 

Coefficients:
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   32.688      0.941   34.73  1.4e-12 ***
IndA          -5.875      0.842   -6.98  2.3e-05 ***
IndB          -4.625      0.842   -5.49  0.00019 ***
IndC          -9.375      0.842  -11.14  2.5e-07 ***
IndAC          5.125      0.842    6.09  7.9e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.68 on 11 degrees of freedom
Multiple R-squared:  0.956,	Adjusted R-squared:  0.94
F-statistic:   60 on 4 and 11 DF,  p-value: 2.12e-07
```

This approach gives us the effects as desired in Example 10.9! It could be extended to get the other 4 effects.

Example 10.11

This example does not require R.

10.5 Fractional Factorial Designs

Example 10.12

This example is not required for 161.100.

Example 10.13

This example is not required for 161.100.

Example 10.14

This example is not required for 161.100.

Example 10.15

This example is not required for 161.100.

Chapter 11

Inferential Methods in Regression and Correlation

Before you can do any examples in this chapter, you will need to execute the command

```
> library(DevFarn2)
```

This will then give you direct access to the data for each example, and a few extra functions needed for some chapters.

Note that you must issue this command as the first act in any R session.

11.1 Regression Models Involving a Single Independent Variable

Example 11.1

This example does not use software

Example 11.2

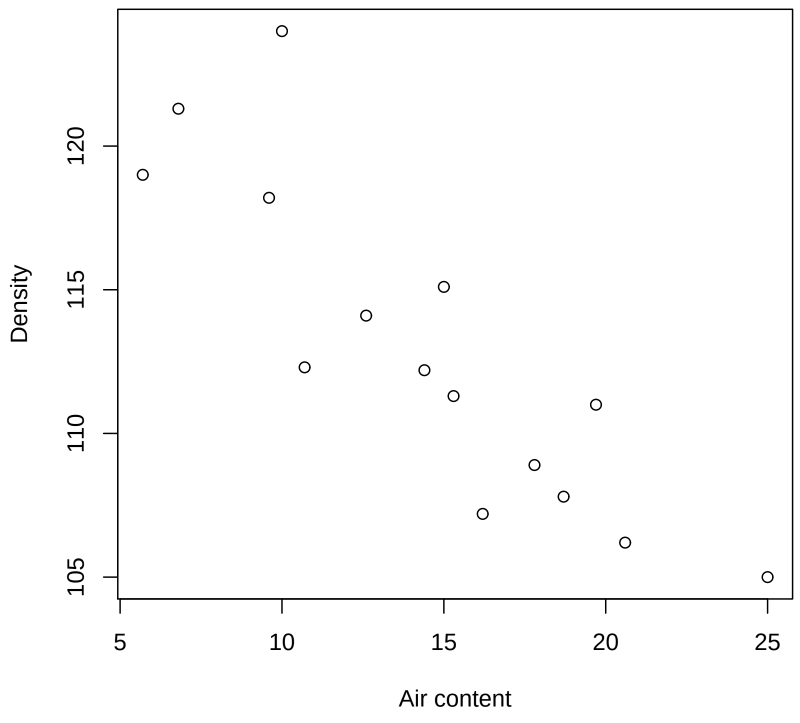
```
> data(e11.2)
> str(e11.2)

'data.frame':	15 obs. of  2 variables:
 $ x: num  5.7 6.8 9.6 10 10.7 12.6 14.4 15 15.3 16.2 ...
 $ y: num  119 121 118 124 112 ...
```

The model coefficients

```
> Model11.2 = lm(y~x, data=e11.2)
> Model11.2
```

Figure 11.5 on page 494 of Devore and Farnum.

```
> plot(e11.2, xlab="Air content", ylab="Density")
```



```
Call:
lm(formula = y ~ x, data = e11.2)

Coefficients:
(Intercept)            x
    126.249       -0.918
```

Example 11.3

We normally want the summary

```
> summary(Model11.2)

Call:
lm(formula = y ~ x, data = e11.2)

Residuals:
   Min     1Q Median     3Q    Max
-4.183 -1.218 -0.835  1.491  6.927

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  126.249      2.254   56.00  < 2e-16 ***
x             -0.918      0.146   -6.29  2.8e-05 ***
```

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

Residual standard error: 2.94 on 13 degrees of freedom
Multiple R-squared:  0.752,       Adjusted R-squared:  0.733
F-statistic: 39.5 on 1 and 13 DF,  p-value: 2.81e-05
```

but the sums of squares aren't given in full so we need

```
> anova(Model11.2)

Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value  Pr(>F)
x          1    342     342    39.5 2.8e-05 ***
Residuals 13    112       9
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

To get the residuals

```
> resid(Model11.2)

      1       2       3       4       5       6       7       8       9
-2.0184  1.2909  0.7603  6.9273 -4.1303 -0.5869 -0.8351  2.6154 -0.9093
     10      11      12      13      14      15
-4.1834 -1.0152 -1.2894  2.8283 -1.1459  1.6917
```

11.2 Inferences About the Slope Coefficient β

Example 11.4

to get the slope coefficient

```
> coef(Model11.2)[2]

      x
-0.9176
```

and to get the right t value

```
> qt(c(0.025,0.975), 13)

[1] -2.16  2.16
```

To get the standard error using commands is possible, but most easily done by typing it in.

```
> summary(Model11.2)$coefficients
```

```
            Estimate Std. Error t value  Pr(>|t|)
(Intercept) 126.2489      2.254  56.001 6.916e-17
x            -0.9176      0.146  -6.286 2.808e-05
```

and to get the exact value required...

```
> summary(Model11.2)$coefficients[2,2]

[1] 0.146
```

So the confidence interval is

```
> coef(Model11.2)[2] + qt(c(0.025,0.975), 13)*summary(Model11.2)$coefficients[2,2]

[1] -1.2330 -0.6022
```

Example 11.5

```
> data(e11.5)
> str(e11.5)

'data.frame':        17 obs. of  2 variables:
 $ x: num  4.6 17 17.4 18 18.5 22.4 36.5 30 34 38.8 ...
 $ y: num  0.66 0.92 1.45 1.03 0.7 0.73 1.2 0.8 0.91 1.19 ...

> Model11.5 = lm(y~x, data=e11.5)
> summary(Model11.5)

Call:
lm(formula = y ~ x, data = e11.5)

Residuals:
    Min      1Q  Median      3Q     Max
-0.2174 -0.1484  0.0011  0.1145  0.5416

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.77249    0.09478    8.15  6.8e-07 ***
x            0.00781    0.00190    4.11  0.00094 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.199 on 15 degrees of freedom
Multiple R-squared:  0.529,       Adjusted R-squared:  0.498
F-statistic: 16.9 on 1 and 15 DF,  p-value: 0.000936
```

Example 11.6

does not use R nor is it covered in 161.100

11.3 Inferences Based on the Estimated Regression Line

Example 11.7

```
> data(e11.7)
> str(e11.7)

'data.frame':       18 obs. of  2 variables:
 $ x: num  8 15 16.5 20 20 27.5 30 30 35 38 ...
 $ y: num  22.8 27.2 23.7 17.1 21.5 18.6 16.1 23.4 13.4 19.5 ...

> Model11.7 = lm(y~x, data=e11.7)
> summary(Model11.7)

Call:
lm(formula = y ~ x, data = e11.7)

Residuals:
   Min     1Q Median     3Q    Max 
-4.132 -2.004 -0.749  2.137  5.144 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  27.1829     1.6513   16.46  1.9e-11 ***
x            -0.2976     0.0412   -7.23  2.0e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Residual standard error: 2.86 on 16 degrees of freedom
Multiple R-squared:  0.766,       Adjusted R-squared:  0.751 
F-statistic: 52.3 on 1 and 16 DF,  p-value: 2.01e-06 

> predict(Model11.7, data.frame(x=45), interval="confidence")

    fit   lwr  upr
1 13.79 12.19 15.4
```

to do this for more observations...

```
> predict(Model11.7, data.frame(x=c(45,35)), interval="confidence")

    fit   lwr   upr
1 13.79 12.19 15.40
2 16.77 15.33 18.21
```

Example 11.8

```
> predict(Model11.7, data.frame(x=45), interval="prediction")

    fit   lwr   upr
1 13.79 7.512 20.07
```

11.4 Multiple Regression Models

Example 11.9

to 11.11 do not use software

11.5 Inferences in Multple Regression

Example 11.12

```
> data(e11.12)
> str(e11.12)
```

```
'data.frame':        30 obs. of  5 variables:
 $ Force      : int  30 40 30 40 30 40 30 40 30 40 ...
 $ Power      : int  60 60 90 90 60 60 90 90 60 60 ...
 $ Temperature: int  175 175 175 175 225 225 225 225 175 175 ...
 $ Time       : int  15 15 15 15 15 15 15 15 25 25 ...
 $ Strength   : num  26.2 26.3 39.8 39.7 38.6 35.5 48.8 37.8 26.6 23.4 ...
```

```
> Model11.12 = lm(Strength ~ Force + Power + Temperature + Time, data=e11.12)
> summary(Model11.12)
```

```
Call:
lm(formula = Strength ~ Force + Power + Temperature + Time, data = e11.12)

Residuals:
    Min      1Q  Median      3Q     Max 
-11.090  -1.761  -0.307   2.439   7.593 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -37.4767    13.0996   -2.86   0.0084 ** 
Force         0.2117     0.2106    1.01   0.3244    
Power         0.4983     0.0702    7.10  1.9e-07 ***
Temperature   0.1297     0.0421    3.08   0.0050 ** 
Time          0.2583     0.2106    1.23   0.2313    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.16 on 25 degrees of freedom
Multiple R-squared:  0.714,	Adjusted R-squared:  0.668 
F-statistic: 15.6 on 4 and 25 DF,  p-value: 1.59e-06
```

```
> predict(Model11.12, data.frame(Force= 35, Power=75, Temperature=200, Time=20))
```

```
    1 
38.41 
```

Example 11.13

See Example 11.12 for some output

```
> anova(Model11.12)

Analysis of Variance Table

Response: Strength
            Df Sum Sq Mean Sq F value  Pr(>F)
Force        1     27      27    1.01   0.324
Power        1   1341    1341   50.41 1.9e-07 ***
Temperature  1    252     252    9.48   0.005 **
Time         1     40      40    1.51   0.231
Residuals   25    665      27
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Example 11.14

```
> coef(Model11.12)[3] + qt(c(0.025,0.975), 25)*summary(Model11.12)$coefficients[3,2]

[1] 0.3538 0.6429
```

Example 11.15

```
> data(e3.15)
> str(e3.15)

'data.frame':        13 obs. of  3 variables:
 $ x1: int  61 175 111 124 130 173 169 169 160 244 ...
 $ x2: int  13 21 24 23 64 38 33 61 39 71 ...
 $ y : int  4 18 14 18 26 26 21 30 28 36 ...

> Model11.15 = lm(y~x1*x2, data=e3.15)
> summary(Model11.15)

Call:
lm(formula = y ~ x1 * x2, data = e3.15)

Residuals:
   Min     1Q Median     3Q    Max
-8.441 -2.069 -0.314  2.939  6.938

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.367840   7.178907   -0.33     0.75
x1           0.082788   0.048176    1.72     0.12
x2           0.245971   0.148064    1.66     0.13
x1:x2        0.000528   0.000661    0.80     0.45

Residual standard error: 4.46 on 9 degrees of freedom
Multiple R-squared:  0.952,        Adjusted R-squared:  0.936
F-statistic: 59.3 on 3 and 9 DF,  p-value: 2.98e-06
```

Example 11.16

```
> Model11.16 = lm(y~x1+x2, data=e3.15)
> summary(Model11.16)

Call:
lm(formula = y ~ x1 + x2, data = e3.15)

Residuals:
   Min     1Q Median     3Q    Max
-8.935 -2.218  0.461  3.345  6.071

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -7.3507     3.4847   -2.11  0.06110 .
x1            0.1127     0.0297    3.80  0.00350 **
x2            0.3490     0.0713    4.89  0.00063 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.38 on 10 degrees of freedom
Multiple R-squared:  0.948,        Adjusted R-squared:  0.938
F-statistic:   92 on 2 and 10 DF,  p-value: 3.63e-07

> predict(Model11.16, data.frame(x1=200, x2=40), interval="confidence")

    fit   lwr   upr
1 29.16 25.25 33.07

> predict(Model11.16, data.frame(x1=200, x2=40), interval="prediction")

    fit   lwr   upr
1 29.16 18.64 39.67
```

Example 11.17

not required for 161.100 but useful to know

```
> attach(e11.12)
> Force2=Force^2
> Power2=Power^2
> Temperature2=Temperature^2
> Time2=Time^2
> Model11.17 = lm(Strength ~ Force + Power + Temperature + Time + Force2 + Power2 + Temperature2 + Time2 + For
> summary(Model11.17)

Call:
lm(formula = Strength ~ Force + Power + Temperature + Time +
    Force2 + Power2 + Temperature2 + Time2 + Force:Power + Force:Temperature +
    Force:Time + Power:Temperature + Power:Time + Temperature:Time)

Residuals:
    Min      1Q  Median      3Q     Max
-10.283  -2.177   0.079   2.723   7.917
```

```
Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)       -5.46e-01   1.56e+02    0.00     1.00
Force             -2.30e+00   3.96e+00   -0.58     0.57
Power             -7.61e-02   1.22e+00   -0.06     0.95
Temperature        8.37e-01   8.26e-01    1.01     0.33
Time              -3.99e+00   3.54e+00   -1.13     0.28
Force2             1.52e-02   4.07e-02    0.37     0.71
Power2             1.30e-03   4.53e-03    0.29     0.78
Temperature2      -1.13e-04   1.63e-03   -0.07     0.95
Time2             -7.83e-03   4.07e-02   -0.19     0.85
Force:Power        2.40e-02   1.78e-02    1.35     0.20
Force:Temperature -9.30e-03   1.07e-02   -0.87     0.40
Force:Time         7.55e-02   5.34e-02    1.42     0.18
Power:Temperature -4.67e-03   3.56e-03   -1.31     0.21
Power:Time         2.37e-02   1.78e-02    1.33     0.20
Temperature:Time   7.00e-04   1.07e-02    0.07     0.95

Residual standard error: 5.33 on 15 degrees of freedom
Multiple R-squared:  0.816,       Adjusted R-squared:  0.645
F-statistic: 4.76 on 14 and 15 DF,  p-value: 0.0024
```

We can compare two models when one model completely contains all terms in the other using...

```
> anova(Model11.12, Model11.17)
```

```
Analysis of Variance Table

Model 1: Strength ~ Force + Power + Temperature + Time
Model 2: Strength ~ Force + Power + Temperature + Time + Force2 + Power2 +
    Temperature2 + Time2 + Force:Power + Force:Temperature +
    Force:Time + Power:Temperature + Power:Time + Temperature:Time
  Res.Df RSS Df Sum of Sq    F Pr(>F)
1     25 665
2     15 427 10       238 0.84    0.6
```

```
> detach(e11.12)
```

11.6 Further Aspects of Regression Analysis

Example 11.18

does not require R

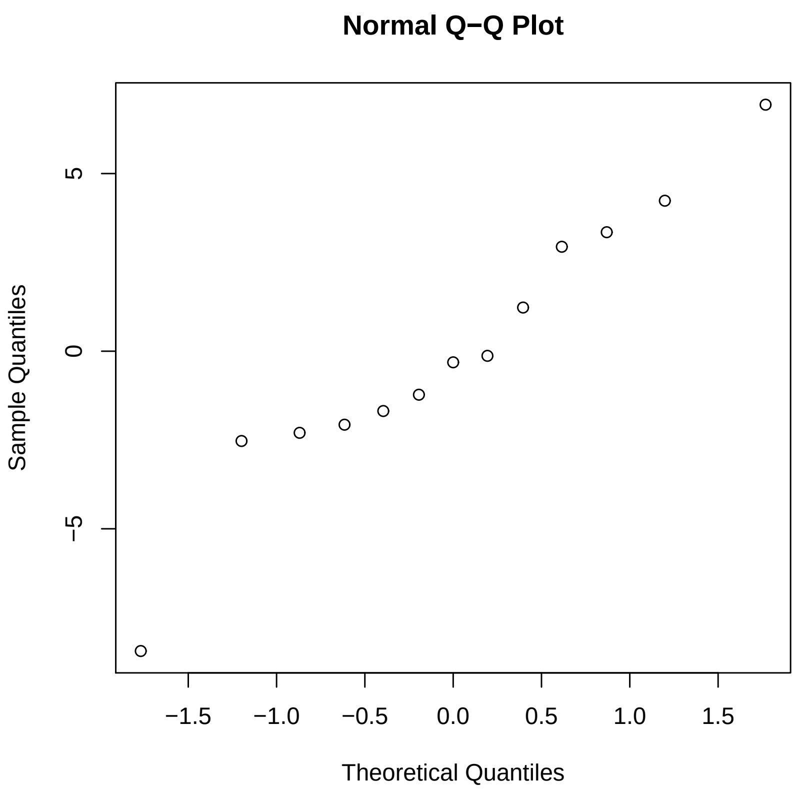
Example 11.19

Note there is actually no need to standardize the residuals for the normality test.

Example 11.20

Figure 11.19 on page 540 of Devore and Farnum.

```
> qqnorm(resid(Model11.15))
```



Example 11.21

This example is not covered in 161.100

Example 11.22

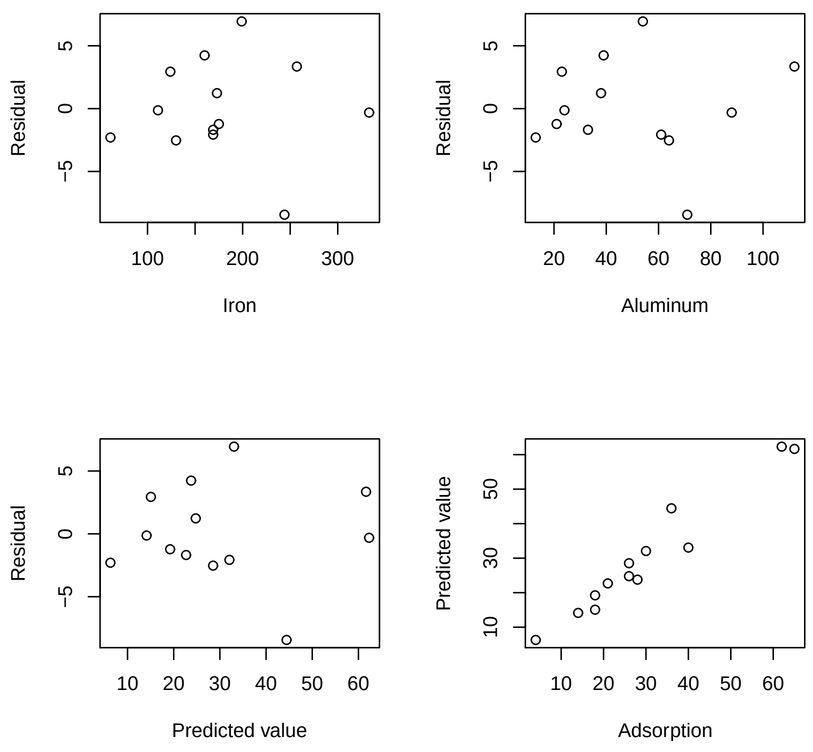
This example is not covered in 161.100

Example 11.23

This example is not covered in 161.100

Figure 11.20 on page 541 of Devore and Farnum.

```
> attach(e3.15)
> par(mfrow=c(2,2))
> plot(x1, resid(Model11.15), ylab="Residual", xlab="Iron")
> plot(x2, resid(Model11.15), ylab="Residual", xlab="Aluminum")
> plot(fitted(Model11.15), resid(Model11.15), ylab="Residual", xlab="Predicted value")
> plot(y, fitted(Model11.15), ylab="Predicted value", xlab="Adsorption")
> detach(e3.15)
```



Appendix A

Installing R

You will need to install R from a downloaded installation file. Go to

http://www.r-project.org

Choose which of the various file servers is best for your location — probably Australia or New Zealand; select your operating system, and then download the base R system. If you are a Windows user, this will be a file with a name like `R-2.12.0-win.exe`

This file is an installer for R. For the easiest introduction to R, you should use the default settings offered during the guided installation process. The development team is trying to ensure that all operating systems can be catered for using the minimum number of installation files. The Windows installer for version 2.12.0, for example, now allows for either a 32 or 64 bit operating system.

Appendix B

Installing the necessary DevFarn2 package

This manual assumes you have installed the add-on package we have developed. called *DevFarn2*. If you are a Windows user then you will find the package in a file called something like `DevFarn2-0.7.zip` while users of other operating systems will use the file `DevFarn2-0.7.tar.gz` instead. These files should be found in the same location where you obtained this manual.

Download the appropriate file to your local machine. Then open R and go to the 'Packages' menu. The menu item you want is the last one, "Install package(s) from local zip file..." — well that's where it is for Windows users.

You should now be able to direct R to the zip file you saved on your machine. If the package is installed successfully, you will see a small series of messages indicating success.

Appendix C

Installing Additional Packages

The examples in Chapter 6 need an additional package that is not built into the base distribution of R. This will probably not automatically be installed when you install the *DevFarn2* package. You'll need an internet connection for this task to work properly.

In the packages menu you will see the item for installing a package. There are literally thousands of additional packages to choose among but do not be tempted to waste time looking at packages you don't need.

Search for the *qcc* package. After you download it, R will install it and if any additional packages were required for it to work, these would have been downloaded as well.

To gain access to the functions and data in a package we will use the `library()` command See Chapter 6 for more on how to use the *qcc* package, and note that you will need to use the `library()` command every time you want to work through any examples in this manual.

Appendix D

GettingHelp

If you need help with the syntax of particular commands, such as the `mean()` command, you might type `?mean` or whatever command is of interest. It is probably a much better idea to emulate the code given in an example from the text. That's why this manual exists.

Other useful documents for R do exist. Unfortunately, some less than useful ones exist as well. You will need to be careful when choosing a suitable resource. At this stage, it is probably a good idea to stick to what resources were made available by the staff responsible for your course.

If all else fails, you should contact your course staff for help.

Index