R Code Supplement for Basic Engineering Data Collection and Analysis by Vardeman and Jobe

Spencer Wadsworth
Iowa State University

2023-07-28

Contents

Introduction

This book contains supplemental R code for *Basic Engineering Data Collection and Analysis* by Stephen B. Vardeman and J. Marcus Jobe. Throughout that textbook, output from MINITAB code is shown in various "Printouts." The R code herein is written to produce similar output for each of these printouts. Included also is code for selected exercises from chapters 4, 7, and 9.

The code provided here is not sophisticated, and the intention is only to give examples of how one might reproduce the MINITAB examples in the book with R. It is not intended to be a set of lessons on how to program in R, though there are some short informational comments that may help clarify what certain lines of code do. Free R software is available at https://www.r-project.org/. For more help with programming in R, manuals including *An Introduction to R* are found at https://cran.r-project.org/manuals.html.

Chapter 2 Example

Printout 1: Random Selection of 25 Objects from a Population of 80 objects (p. 36)

Note: Since the 25 objects are sampled using a random number generator, the selected objects will not match what is selected in the MINITAB example.

```
#set a seed (setting the same seed before randomly sampling objects from the same
#set will result in the same sampled objects)
set.seed(29)

#create vector with object labels 1-80
C1 <- 1:80

#sample 25 objects from the 80
C2 <- sample(C1, 25)

#display sampled labels
C2
```

```
##  [1]  5 44 66  3 71 56 48 12 34 17 40 26 28 52 15 70 51  7 64 59 72 58  1 29 57
```

Chapter 3 Example

Printout 1: Descriptive Statistics for the Percent Waste Data of Table 3.13 (p. 103)

```
#create an R function for calculating and displaying summary statistics
desc_stat <- function(x, tr = .1) {
  N <- length(x)
  mean <- mean(x)
  median <- median(x)
  trmean <- mean(x, trim = tr)
  sd <- sd(x)
  se <- sd(x)/sqrt(N)
  min <- min(x)
  max <- max(x)

  #type = 6 gives the same quantile values as in the MINITAB output
  #other types may not
  q1 <- as.numeric(quantile(x, .25, type = 6))
  q3 <- as.numeric(quantile(x, .75, type = 6))

  stats <- c(N = N, mean = mean, median = median, trmean = trmean, sd = sd,
             se = se, min = min, max = max, q1 = q1, q3 = q3)

  #round to the 4th decimal
  stats <- round(stats, 4)
  return(stats)
}
```

```
#save data to vectors
supply_1 <- c(.37, .52, .65, .92, 2.89, 3.62)
supply_2 <- c(.89, .99, 1.45, 1.47, 1.58, 2.27, 2.63, 6.54)

#call the previously made function over both sets of data to display the
#descriptive statistics
rbind(desc_stat(supply_1), desc_stat(supply_2))
```

```
##      N   mean median trmean     sd     se  min  max     q1     q3
## [1,] 6 1.4950  0.785 1.4950 1.3945 0.5693 0.37 3.62 0.4825 3.0725
## [2,] 8 2.2275  1.525 2.2275 1.8392 0.6503 0.89 6.54 1.1050 2.5400
```

Chapter 4 Examples

Note: Some examples in this section as well as in examples from chapters 7, 8, and 9 contain ANOVA tables. Unlike in MINITAB, the default ANOVA tables in R do not directly give overall Regression df and SS entries. Rather they give sub entries which sum to the overall Regression df and SS.

Printout 1: Fitting the Least Squares Line to the Pressure/Density Data (p. 138-39)

```
pressure <- rep(c(2000, 4000, 6000, 8000, 10000), each = 3)
density <- c(2.486, 2.479, 2.472, 2.558, 2.570, 2.580,
             2.646, 2.657, 2.653, 2.724, 2.774, 2.808,
             2.861, 2.879, 2.858)

#use the lm function for fitting the least squares line
ls_mod <- lm(density ~ pressure)

#display the fitted model including fitted coefficients and R-squared value
summary(ls_mod)
```

```
##
## Call:
## lm(formula = density ~ pressure)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.040333 -0.010833 -0.000333  0.010000  0.043667
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.375e+00  1.206e-02  197.01  < 2e-16 ***
## pressure    4.867e-05  1.817e-06   26.78 9.31e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01991 on 13 degrees of freedom
## Multiple R-squared:  0.9822, Adjusted R-squared:  0.9808
## F-statistic: 717.1 on 1 and 13 DF,  p-value: 9.307e-13
```

```
#display the ANOVA table
anova(ls_mod)
```

```
## Analysis of Variance Table
##
## Response: density
##           Df   Sum Sq  Mean Sq F value    Pr(>F)
## pressure   1 0.284213 0.284213  717.06 9.307e-13 ***
## Residuals 13 0.005153 0.000396
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#save model fits and residuals
ls_preds <- predict(ls_mod, se.fit = TRUE)
ls_fits <- ls_preds$fit
ls_resids <- resid(ls_mod)

#get the standardized residuals
ls_stresid <- rstandard(ls_mod)

#get the fitted standard errors
ls_sdfit <- ls_preds$se.fit

#display the data along with fitted values, standard deviations of fit,
#residuals, and standardized residuals
cbind(pressure = pressure, density = density, Fit = ls_fits,
      `StDev Fit` = ls_sdfit, Residual = ls_resids,
      `St Resid` = ls_stresid)
```

```
##    pressure density      Fit   StDev Fit      Residual    St Resid
## 1      2000   2.486 2.472333 0.008903471  0.0136666667  0.76749095
## 2      2000   2.479 2.472333 0.008903471  0.0066666667  0.37438583
## 3      2000   2.472 2.472333 0.008903471 -0.0003333333 -0.01871929
## 4      4000   2.558 2.569667 0.006295705 -0.0116666667 -0.61770510
## 5      4000   2.570 2.569667 0.006295705  0.0003333333  0.01764872
## 6      4000   2.580 2.569667 0.006295705  0.0103333333  0.54711023
## 7      6000   2.646 2.667000 0.005140421 -0.0210000000 -1.09183386
## 8      6000   2.657 2.667000 0.005140421 -0.0100000000 -0.51992089
## 9      6000   2.653 2.667000 0.005140421 -0.0140000000 -0.72788924
## 10     8000   2.724 2.764333 0.006295705 -0.0403333333 -2.13549477
## 11     8000   2.774 2.764333 0.006295705  0.0096666667  0.51181280
## 12     8000   2.808 2.764333 0.006295705  0.0436666667  2.31198195
## 13    10000   2.861 2.861667 0.008903471 -0.0006666667 -0.03743858
## 14    10000   2.879 2.861667 0.008903471  0.0173333333  0.97340315
## 15    10000   2.858 2.861667 0.008903471 -0.0036666667 -0.20591220
```

Printout 2: Quadratic Fit to the Fly Ash Data (p. 142)

```
y <- c(1221, 1207, 1187, 1555, 1562, 1575, 1827, 1839,
              1802, 1609, 1627, 1642, 1451, 1472, 1465, 1321,
              1289, 1292)
x <- rep(0:5, each = 3)

#square x for the quadratic predictor variable
x_2 <- x^2

#fit the model
quad_mod <- lm(y ~ x + x_2)

#display the model summary
summary(quad_mod)
```

```
##
## Call:
## lm(formula = y ~ x + x_2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -95.983 -70.193  -7.895  51.548 137.419
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1242.893     42.982  28.917 1.43e-14 ***
## x            382.665     40.430   9.465 1.03e-07 ***
## x_2          -76.661      7.762  -9.877 5.88e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 82.14 on 15 degrees of freedom
## Multiple R-squared:  0.8667, Adjusted R-squared:  0.849
## F-statistic: 48.78 on 2 and 15 DF,  p-value: 2.725e-07
```

```
#display the ANOVA table
anova(quad_mod)
```

```
## Analysis of Variance Table
##
## Response: y
##           Df Sum Sq Mean Sq F value    Pr(>F)
## x          1     21      21  0.0032    0.9559
## x_2        1 658209  658209 97.5546 5.879e-08 ***
## Residuals 15 101206    6747
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Printout 3: Cubic Fit to the Fly Ash Data (p. 144)

```
#cube x for the cubic predictor variable
x_3 <- x^3

#fit the model
cube_mod <- lm(y ~ x + x_2 + x_3)

#display the model summary
summary(cube_mod)
```

```
##
## Call:
## lm(formula = y ~ x + x_2 + x_3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -70.677 -27.353  -3.874  24.579  93.545
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1188.050     28.786  41.272 5.03e-16 ***
## x            633.113     55.913  11.323 1.96e-08 ***
## x_2         -213.767     27.787  -7.693 2.15e-06 ***
## x_3           18.281      3.649   5.010 0.000191 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 50.88 on 14 degrees of freedom
## Multiple R-squared:  0.9523, Adjusted R-squared:  0.9421
## F-statistic: 93.13 on 3 and 14 DF,  p-value: 1.733e-09
```

```
#display the ANOVA table
anova(cube_mod)
```

```
## Analysis of Variance Table
##
## Response: y
##           Df Sum Sq Mean Sq  F value   Pr(>F)
## x          1     21      21   0.0083 0.928881
## x_2        1 658209  658209 254.2771 2.26e-10 ***
## x_3        1  64967   64967  25.0977 0.000191 ***
## Residuals 14  36240    2589
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Printout 4: Multiple Regression for the Stack Loss Data (p. 151)

```
air <- c(80, rep(62, 4), rep(58, 6), rep(50, 5), 56)
water <- c(27, 22, 23, 24, 24, 23, 18, 18, 17, 18, 19, 18, 18,
                19, 19, 20, 20)
acid <- c(88, 87, 87, 93, 93, 87, 80, 89, 88, 82, 93, 89, 86,
                      72, 79, 80, 82)
stack <- c(37, 18, 18, 19, 20, 15, 14, 14, 13, 11, 12, 8, 7, 8,
                8, 9, 15)

#use the vectors to make a data frame to operate on
stack_loss <- data.frame(air, water, acid, stack)

#fit model
mult_reg_mod <- lm(stack ~ air + water + acid, data = stack_loss)

#print model summary
summary(mult_reg_mod)
```

```
##
## Call:
## lm(formula = stack ~ air + water + acid, data = stack_loss)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5065 -0.6713  0.1432  0.5862  1.9342
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -37.65246    4.73205  -7.957 2.37e-06 ***
## air           0.79769    0.06744  11.828 2.48e-08 ***
## water         0.57734    0.16597   3.479  0.00408 **
## acid         -0.06706    0.06160  -1.089  0.29611
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.253 on 13 degrees of freedom
## Multiple R-squared:  0.975,  Adjusted R-squared:  0.9692
## F-statistic:   169 on 3 and 13 DF,  p-value: 1.159e-10
```

```
#print ANOVA table
anova(mult_reg_mod)
```

```
## Analysis of Variance Table
##
## Response: stack
##           Df Sum Sq Mean Sq F value   Pr(>F)
## air        1 775.48  775.48 494.160 9.97e-12 ***
## water      1  18.49   18.49  11.784 0.004453 **
## acid       1   1.86    1.86   1.185 0.296107
## Residuals 13  20.40    1.57
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#get predictions, standard errors, and residuals
mr_preds <- predict(mult_reg_mod, se.fit = TRUE)
mr_fits <- mr_preds$fit
mr_resids <- resid(mult_reg_mod)
mr_stresid <- rstandard(mult_reg_mod)
mr_sdfit <- mr_preds$se.fit

mult_reg_obs <- cbind(air = air, stack = stack, Fit = mr_fits,
             `StDev Fit` = mr_sdfit, Residual = mr_resids,
             `St Resid` = mr_stresid)

#show the "unusual" observation from MINITAB output ("unusualness" is judged in
#terms of the magnitudes of the residuals)
mult_reg_obs[10,]
```

```
##        air      stack        Fit  StDev Fit   Residual   St Resid
## 58.0000000 11.0000000 13.5064973  0.5519432 -2.5064973 -2.2288554
```

Printout 5: Multiple Regression for the Lift/Drag Ratio Data (p. 157)

```
x1 <- rep(c(-1.2, 0, 1.2), each = 3)
x2 <- rep(c(-1.2, 0, 1.2), 3)
y <- c(.858, 3.156, 3.644, 4.281, 3.481, 3.918, 4.136, 3.364, 4.018)

int_mod <- lm(y ~ x1 + x2 + x1*x2)

summary(int_mod)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x1 * x2)
##
```

```
## Residuals:
##        1        2        3        4        5        6        7        8
## -0.81694  0.37089 -0.25128  1.23672  0.05256  0.10539 -0.27761 -0.70778
##        9
##  0.28806
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.4284     0.2613  13.121 4.59e-05 ***
## x1            0.5361     0.2667   2.010    0.101
## x2            0.3201     0.2667   1.200    0.284
## x1:x2        -0.5042     0.2722  -1.852    0.123
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7839 on 5 degrees of freedom
## Multiple R-squared:  0.6406, Adjusted R-squared:  0.425
## F-statistic: 2.971 on 3 and 5 DF,  p-value: 0.1358
```

```
anova(int_mod)
```

```
## Analysis of Variance Table
##
## Response: y
##           Df Sum Sq Mean Sq F value Pr(>F)
## x1         1 2.4833 2.48327  4.0412 0.1006
## x2         1 0.8855 0.88550  1.4410 0.2837
## x1:x2      1 2.1083 2.10830  3.4310 0.1232
## Residuals  5 3.0724 0.61449
```

Printout 6: Computation for the Joint Strength Data (p. 170-71)

```
#using the function factor() makes the data categorical instead of numerical
joint <- factor(c("beveled", "butt", "beveled", "butt", "beveled", "beveled",
           "lap", "beveled", "butt", "lap", "lap", "lap", "butt",
           "lap", "butt", "beveled"))

wood <- factor(c("oak", "pine", "walnut", "oak", "oak", "pine", "walnut",
          "walnut", "walnut", "oak", "oak", "pine", "pine", "pine",
          "walnut", "pine"))

y <- c(1518, 829, 2571, 1169, 1927, 1348, 1489, 2443, 1263, 1295,
       1561, 1000, 596, 859, 1029, 1207)
```

```r
#setting this option makes the "constraints" for representing linear models
#match those in MINITAB
options(contrasts=c("contr.sum", "contr.sum"))

js_mod <- lm(y ~ joint*wood)

anova(js_mod)
```

```
## Analysis of Variance Table
##
## Response: y
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## joint       2 2153879 1076940 37.3911 0.0001835 ***
## wood        2 1641095  820548 28.4893 0.0004332 ***
## joint:wood  4  468408  117102  4.0658 0.0515147 .
## Residuals   7  201614   28802
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
summary(js_mod)
```

```
##
## Call:
## lm(formula = y ~ joint * wood)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -204.5  -82.0    0.0   82.0  204.5
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1375.67      44.22  31.107 9.16e-09 ***
## joint1         460.00      59.63   7.714 0.000115 ***
## joint2        -366.50      63.95  -5.731 0.000712 ***
## wood1           64.17      63.95   1.003 0.349066
## wood2         -402.50      59.63  -6.750 0.000265 ***
## joint1:wood1  -177.33      85.38  -2.077 0.076418 .
## joint2:wood1    95.67      97.07   0.986 0.357196
## joint1:wood2  -155.67      82.20  -1.894 0.100104
## joint2:wood2   105.83      85.38   1.240 0.255071
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 169.7 on 7 degrees of freedom
## Multiple R-squared:  0.9548, Adjusted R-squared:  0.9032
## F-statistic:  18.5 on 8 and 7 DF,  p-value: 0.0004713
```

```r
js_preds <- predict(js_mod, se.fit = TRUE)
js_fits <- js_preds$fit
js_resids <- resid(js_mod)
js_stresid <- rstandard(js_mod)
```

```
js_sdfit <- js_preds$se.fit

js_obs <- cbind(strength = y, Fit = js_fits,
             `StDev Fit` = js_sdfit, Residual = js_resids,
             `St Resid` = js_stresid)

#observations with large influence as shown in the MINITAB output
js_obs[c(4, 7),]
```

```
##   strength  Fit StDev Fit      Residual St Resid
## 4     1169 1169  169.7115  9.237056e-14      NaN
## 7     1489 1489  169.7115 -1.421085e-14      NaN
```

```
#call the library that contains the lsmeans() function for obtaining the
#least squares means
#if the library is not installed use "install.packages('lsmeans')" to install it
library(lsmeans)
```

```
## Loading required package: emmeans
```

```
## The 'lsmeans' package is now basically a front end for 'emmeans'.
## Users are encouraged to switch the rest of the way.
## See help('transition') for more information, including how to
## convert old 'lsmeans' objects and scripts to work with 'emmeans'.
```

```
#least squares means for joint main effects
lsmeans(js_mod, specs = "joint")
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

```
##  joint   lsmean   SE df lower.CL upper.CL
##  beveled   1836 69.3  7     1672     1999
##  butt      1009 80.0  7      820     1198
##  lap       1282 80.0  7     1093     1471
##
## Results are averaged over the levels of: wood
## Confidence level used: 0.95
```

```
#least squares means for wood main effects
lsmeans(js_mod, specs = "wood")
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

```
##  wood   lsmean   SE df lower.CL upper.CL
##  oak      1440 80.0  7     1251     1629
##  pine      973 69.3  7      809     1137
##  walnut   1714 80.0  7     1525     1903
##
## Results are averaged over the levels of: joint
## Confidence level used: 0.95
```

```
#least squares means for interaction terms
lsmeans(js_mod, specs = c("joint", "wood"))
```

```
##  joint   wood   lsmean  SE df lower.CL upper.CL
##  beveled oak      1722 120  7     1439     2006
##  butt    oak      1169 170  7      768     1570
##  lap     oak      1428 120  7     1144     1712
##  beveled pine     1278 120  7      994     1561
##  butt    pine      712 120  7      429      996
##  lap     pine      930 120  7      646     1213
##  beveled walnut   2507 120  7     2223     2791
##  butt    walnut   1146 120  7      862     1430
##  lap     walnut   1489 170  7     1088     1890
##
## Confidence level used: 0.95
```

Printout 7: Calculation of Fitted Effects for the Airplane Experiment (p. 187)

```
#using the function factor() makes the data categorical instead of numerical
design <- factor(rep(1:2, each = 4))
size <- factor(rep(rep(1:2, each = 2), 2))
paper <- factor(rep(1:2, 4))
distance <- c(15.8, 22.5, 18.4, 21.6, 26.0, 18.3, 24.0, 14.0)


exp_mod <- lm(distance ~ design*size*paper)
anova(exp_mod)
```

```
## Warning in anova.lm(exp_mod): ANOVA F-tests on an essentially perfect fit are
## unreliable
```

```
## Analysis of Variance Table
##
## Response: distance
##                   Df Sum Sq Mean Sq F value Pr(>F)
## design             1  2.000   2.000     NaN    NaN
## size               1  2.645   2.645     NaN    NaN
## paper              1  7.605   7.605     NaN    NaN
```

```
## design:size        1  8.000   8.000     NaN    NaN
## design:paper       1 95.220  95.220     NaN    NaN
## size:paper         1  4.205   4.205     NaN    NaN
## design:size:paper  1  0.180   0.180     NaN    NaN
## Residuals          0  0.000     NaN
```

```
#display the model summary (NaN values are a result of the denominator of the
#F-test statistic being zero as stated in the MINITAB output)
summary(exp_mod)
```

```
##
## Call:
## lm(formula = distance ~ design * size * paper)
##
## Residuals:
## ALL 8 residuals are 0: no residual degrees of freedom!
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)             20.075        NaN     NaN      NaN
## design1                 -0.500        NaN     NaN      NaN
## size1                    0.575        NaN     NaN      NaN
## paper1                   0.975        NaN     NaN      NaN
## design1:size1           -1.000        NaN     NaN      NaN
## design1:paper1          -3.450        NaN     NaN      NaN
## size1:paper1            -0.725        NaN     NaN      NaN
## design1:size1:paper1    -0.150        NaN     NaN      NaN
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 7 and 0 DF,  p-value: NA
```

Chapter 5 Example

Printout 1: Simulation of Solar Collector Efficiency (p. 306-07)

```
#create a function to calculate efficiency from (5.52) in chapter 5 example 22
Efficiency <- function(C, G, A, Mi, Mo, Ta, Ti, To) {
  return(C/(G*A) * (Mo*To - Mi*Ti - (Mo - Mi)*Ta))
}

#measured values from Table 5.23
meanC <- 1003.8
meanG <- 1121.4
meanA <- 1.58
meanMi <- .0234
meanMo <- .0247
meanTa <- -13.22
meanTi <- -6.08
meanTo <- 24.72

#uncertainties (standard deviations) from Table 5.23
sdC <- 1.004
sdG <- 33.6
sdA <- .005
sdMi <- .00035
sdMo <- .00037
sdTa <- .5
sdTi <- .5
sdTo <- .5

#sample size
N <- 100

#simulate from normal distributions with measured values as the means and
#uncertainties as the standard deviations
simC <- rnorm(N, meanC, sdC)
simG <- rnorm(N, meanG, sdG)
simA <- rnorm(N, meanA, sdA)
simMi <- rnorm(N, meanMi, sdMi)
simMo <- rnorm(N, meanMo, sdMo)
simTa <- rnorm(N, meanTa, sdTa)
simTi <- rnorm(N, meanTi, sdTi)
simTo <- rnorm(N, meanTo, sdTo)
```

```
#create a vector to be filled with simulated efficiencies
effsim <- rep(0, N)

#loop that calculates efficiency for each simulated sample
for (i in 1:N) {
  effsim[i] <- Efficiency(simC[i], simG[i], simA[i], simMi[i], simMo[i],
                          simTa[i], simTi[i], simTo[i])
}


#call a global option so that the results of the desc_stat() function  are
#not displayed in scientific notation
options(scipen = 999)
```

Note: Since the following R output is generated by simulation, it doesn't exactly match the simulation outcomes derived using MINITAB.

```
#display the descriptive statistics of 100 simulated efficiency values
desc_stat(effsim)
```

```
##         N      mean    median    trmean        sd        se       min       max
## 100.0000    0.4369    0.4380    0.4366    0.0207    0.0021    0.3983    0.4912
##        q1        q3
##    0.4210    0.4522
```

```
#display a stem and leaf plot
stem(effsim)
```

```
##
##   The decimal point is 2 digit(s) to the left of the |
##
##   39 | 89
##   40 | 145556778
##   41 | 000114557889
##   42 | 012334445577889
##   43 | 001244555668889
##   44 | 0011222335567778899
##   45 | 111334556699999
##   46 | 00112566
##   47 | 489
##   48 | 1
##   49 | 1
```

Chapter 7 Example

Printout 1: ANOVA Table for a One-Way Analysis of the Concrete Strength Data (p. 486)

```
formula <- factor(rep(1:8, each = 3))
strength <- c(5800, 4598, 6508, 5659, 6225, 5376, 5093, 4386, 4103, 3395,
              3820, 3112, 3820, 2829, 2122, 2971, 3678, 3325, 2122, 1372,
              1160, 2051, 2631, 2490)

concrete_mod <- lm(strength ~ formula)

#display the ANOVA table for the fitted model
anova(concrete_mod)
```

```
## Analysis of Variance Table
##
## Response: strength
##           Df   Sum Sq Mean Sq F value       Pr(>F)
## formula    7 47360781 6765826  20.005 0.0000008551 ***
## Residuals 16  5411409  338213
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#load the library for least squares means
library(lsmeans)

#display the least squares means
#the SE column is the pooled standard error (based on the pooled standard deviation)
lsmeans(concrete_mod, specs = "formula")
```

```
##  formula lsmean  SE df lower.CL upper.CL
##  1         5635 336 16     4924     6347
##  2         5753 336 16     5042     6465
##  3         4527 336 16     3816     5239
##  4         3442 336 16     2731     4154
##  5         2924 336 16     2212     3635
##  6         3325 336 16     2613     4036
##  7         1551 336 16      840     2263
##  8         2391 336 16     1679     3102
##
## Confidence level used: 0.95
```

```
#pooled standard deviation
summary(concrete_mod)$sigma
```

```
## [1] 581.5609
```

Chapter 8 Example

Printout 1: Estimated Standard Deviations of the Joint Strength Fitted Effects (p. 562)

```
#see code for Printout 6 in chapter 4 for fitting the model

#display the ANOVA table
anova(js_mod)
```

```
## Analysis of Variance Table
##
## Response: y
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## joint       2 2153879 1076940 37.3911 0.0001835 ***
## wood        2 1641095  820548 28.4893 0.0004332 ***
## joint:wood  4  468408  117102  4.0658 0.0515147 .
## Residuals   7  201614   28802
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#display the fitted model summary
summary(js_mod)
```

```
##
## Call:
## lm(formula = y ~ joint * wood)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -204.5  -82.0    0.0   82.0  204.5
##
## Coefficients:
##              Estimate Std. Error t value      Pr(>|t|)
## (Intercept)   1375.67      44.22  31.107 0.00000000916 ***
## joint1         460.00      59.63   7.714      0.000115 ***
## joint2        -366.50      63.95  -5.731      0.000712 ***
## wood1           64.17      63.95   1.003      0.349066
## wood2         -402.50      59.63  -6.750      0.000265 ***
## joint1:wood1  -177.33      85.38  -2.077      0.076418 .
## joint2:wood1    95.67      97.07   0.986      0.357196
## joint1:wood2  -155.67      82.20  -1.894      0.100104
## joint2:wood2   105.83      85.38   1.240      0.255071
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 169.7 on 7 degrees of freedom
## Multiple R-squared:  0.9548, Adjusted R-squared:  0.9032
## F-statistic:  18.5 on 8 and 7 DF,  p-value: 0.0004713
```

Chapter 9 Examples

Printout 1: Simple Linear Regression for the Pressure/Density Data (p. 672-73)

```
#see the code for Printout 1 of Chapter 4 for fitting the model
summary(ls_mod)
```

```
##
## Call:
## lm(formula = density ~ pressure)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.040333 -0.010833 -0.000333  0.010000  0.043667
##
## Coefficients:
##                Estimate  Std. Error t value             Pr(>|t|)
## (Intercept) 2.375000000 0.012055357  197.01 < 0.0000000000000002 ***
## pressure    0.000048667 0.000001817   26.78    0.000000000000931 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01991 on 13 degrees of freedom
## Multiple R-squared:  0.9822, Adjusted R-squared:  0.9808
## F-statistic: 717.1 on 1 and 13 DF,  p-value: 0.0000000000009307
```

```
#ANOVA table of the model
anova(ls_mod)
```

```
## Analysis of Variance Table
##
## Response: density
##           Df   Sum Sq  Mean Sq F value             Pr(>F)
## pressure   1 0.284213 0.284213  717.06 0.0000000000009307 ***
## Residuals 13 0.005153 0.000396
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#get fits, residuals, and standardized residuals
ls_preds <- predict(ls_mod, se.fit = TRUE)
ls_fits <- ls_preds$fit
ls_resids <- resid(ls_mod)
ls_stresid <- rstandard(ls_mod)
```

```
#get predictions
ls_sdfit <- ls_preds$se.fit

#print the table
cbind(pressure = pressure, density = density, Fit = ls_fits,
      `StDev Fit` = ls_sdfit, Residual = ls_resids,
      `St Resid` = ls_stresid)
```

```
##    pressure density      Fit   StDev Fit      Residual    St Resid
## 1      2000   2.486 2.472333 0.008903471  0.0136666667  0.76749095
## 2      2000   2.479 2.472333 0.008903471  0.0066666667  0.37438583
## 3      2000   2.472 2.472333 0.008903471 -0.0003333333 -0.01871929
## 4      4000   2.558 2.569667 0.006295705 -0.0116666667 -0.61770510
## 5      4000   2.570 2.569667 0.006295705  0.0003333333  0.01764872
## 6      4000   2.580 2.569667 0.006295705  0.0103333333  0.54711023
## 7      6000   2.646 2.667000 0.005140421 -0.0210000000 -1.09183386
## 8      6000   2.657 2.667000 0.005140421 -0.0100000000 -0.51992089
## 9      6000   2.653 2.667000 0.005140421 -0.0140000000 -0.72788924
## 10     8000   2.724 2.764333 0.006295705 -0.0403333333 -2.13549477
## 11     8000   2.774 2.764333 0.006295705  0.0096666667  0.51181280
## 12     8000   2.808 2.764333 0.006295705  0.0436666667  2.31198195
## 13    10000   2.861 2.861667 0.008903471 -0.0006666667 -0.03743858
## 14    10000   2.879 2.861667 0.008903471  0.0173333333  0.97340315
## 15    10000   2.858 2.861667 0.008903471 -0.0036666667 -0.20591220
```

```
#get prediction for x = 5,000 along with the standard deviation of the fit
#and the 95% confidence interval
pred5000c <- predict(ls_mod, newdata = data.frame(pressure = 5000),
                     se.fit = TRUE, interval = "confidence")

#fit with the confidence interval
pred5000c$fit
```

```
##        fit      lwr      upr
## 1 2.618333 2.606554 2.630112
```

```
#standard deviation of the fit
pred5000c$se.fit
```

```
## [1] 0.00545224
```

```
#fit prediction for x = 5,000 along with the 95% prediction interval
pred5000p <- predict(ls_mod, newdata = data.frame(pressure = 5000),
                     se.fit = TRUE, interval = "prediction")



#fit with the prediction interval
pred5000p$fit
```

```
##        fit      lwr      upr
## 1 2.618333 2.573739 2.662927
```

Printout 2: Multiple Linear Regression for the Stack Loss Data (p. 679)

```
#air and water data are the same as used in Printout 4 in Chapter 4
#to compare the model with the MINITAB output (x1 = air and x2 = water)

#make the x1**2 variable
x1_2 <- air^2

mlr_mod <- lm(stack ~ air + water + x1_2)

#s_SF is found as the residual standard error in the summary
summary(mlr_mod)
```

```
##
## Call:
## lm(formula = stack ~ air + water + x1_2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0177 -0.6530 -0.1252  0.5101  2.3429
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -15.409290  12.602668  -1.223  0.24315
## air          -0.069142   0.398419  -0.174  0.86490
## water         0.527804   0.150079   3.517  0.00379 **
## x1_2          0.006818   0.003178   2.145  0.05139 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.125 on 13 degrees of freedom
## Multiple R-squared:  0.9799, Adjusted R-squared:  0.9752
## F-statistic: 210.8 on 3 and 13 DF,  p-value: 0.00000000002854
```

```
anova(mlr_mod)
```

```
## Analysis of Variance Table
##
## Response: stack
##           Df Sum Sq Mean Sq  F value             Pr(>F)
## air        1 775.48  775.48 613.2127 0.000000000002526 ***
## water      1  18.49   18.49  14.6231           0.002109 **
## x1_2       1   5.82    5.82   4.6024           0.051393 .
## Residuals 13  16.44    1.26
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
mlr_preds <- predict(mlr_mod, se.fit = TRUE)
mlr_fits <- mlr_preds$fit
mlr_resids <- resid(mlr_mod)
```

```
mlr_stresid <- rstandard(mlr_mod)
mlr_sdfit <- mlr_preds$se.fit

mlr_obs <- cbind(x1 = air, y = stack, Fit = mlr_fits,
             `StDev Fit` = mlr_sdfit, Residual = mlr_resids,
             `St Resid` = mlr_stresid)

mlr_obs
```

```
##    x1  y       Fit StDev Fit    Residual   St Resid
## 1  80 37 36.947239 1.1207801  0.05276127  0.5731607
## 2  62 18 18.125178 0.4074059 -0.12517752 -0.1194258
## 3  62 18 18.652982 0.4620239 -0.65298193 -0.6368947
## 4  62 19 19.180786 0.5531725 -0.18078634 -0.1846469
## 5  62 20 19.180786 0.5531725  0.81921366  0.8367072
## 6  58 15 15.656762 0.5128107 -0.65676213 -0.6562220
## 7  58 14 13.017740 0.4752072  0.98225992  0.9637413
## 8  58 14 13.017740 0.4752072  0.98225992  0.9637413
## 9  58 13 12.489936 0.5945868  0.51006433  0.5343732
## 10 58 11 13.017740 0.4752072 -2.01774008 -1.9796994
## 11 58 12 13.545544 0.3783619 -1.54554449 -1.4594486
## 12 50  8  7.679858 0.4934499  0.32014195  0.3168125
## 13 50  7  7.679858 0.4934499 -0.67985805 -0.6727875
## 14 50  8  8.207662 0.4991449 -0.20766246 -0.2060740
## 15 50  8  8.207662 0.4991449 -0.20766246 -0.2060740
## 16 50  9  8.735467 0.5475818  0.26453313  0.2693189
## 17 56 15 12.657059 0.2977241  2.34294131  2.1605336
```

```
#predict and make the intervals for air = 60 and water = 20
pred60_20c <- predict(mlr_mod, newdata = data.frame(air = 60, water = 20,
                                                    x1_2 = 60^2),
                      se.fit = TRUE, interval = "confidence")

pred60_20p <- predict(mlr_mod, newdata = data.frame(air = 60, water = 20,
                                                    x1_2 = 60^2),
                      se.fit = TRUE, interval = "prediction")

#prediction and confidence interval
pred60_20c$fit
```

```
##        fit      lwr      upr
## 1 15.54419 14.71685 16.37152
```

```
#standard deviation of the fit
pred60_20c$se.fit
```

```
## [1] 0.3829601
```

```
#prediction and prediction interval
pred60_20p$fit
```

```
##        fit      lwr      upr
## 1 15.54419 12.97773 18.11065
```

Printout 3: Analysis of the Fitted Quadratic for the Bread Wrapper Data (p. 704)

```
#make M1 matrix
M1 <- matrix(data = c(-.7596, -.175, -.250,
                      -.175, -1.042, .075,
                      -.250, .075, -1.148), nrow = 3)

#make M2 vector
M2 <- c(-1.104, .0872, 1.020)

#eigen decomposition
C1 <- eigen(M1)

#print the eigenvalues (the eigen() function always lists eigenvalues in descending order)
C1$values
```

```
## [1] -0.561900 -1.116802 -1.270898
```

```
#calculate the M5 matrix as seen in the MINITAB output
M3 <- solve(M1)
M4 <- M2%*%M3
M5 <- M4*-.5
M5
```

```
##           [,1]      [,2]      [,3]
## [1,] -1.011039 0.2606919 0.6814561
```

Printout 4: Multiple Regression Version of the With-Interactions Factorial Analysis of Joint Strength (p. 711-12)

```
xa1 <- rep(c(1, 0, -1), times = c(3, 4, 4))
xa2 <- rep(c(0, 1, -1), times = c(3, 4, 4))
xb1 <- c(1, 1, -1, 1, 1, -1, -1, 1, 1, -1, -1)
y <- c(829, 596, 1169, 1348, 1207, 1518, 1927, 1000, 859, 1295, 1561)

js_data <- data.frame(xa1, xa2, xb1, y)
print(js_data)
```

```
##    xa1 xa2 xb1    y
## 1    1   0   1  829
## 2    1   0   1  596
## 3    1   0  -1 1169
## 4    0   1   1 1348
## 5    0   1   1 1207
## 6    0   1  -1 1518
## 7    0   1  -1 1927
## 8   -1  -1   1 1000
## 9   -1  -1   1  859
## 10  -1  -1  -1 1295
## 11  -1  -1  -1 1561
```

```
jsi_mod <- lm(y ~ xa1*xb1 + xa2*xb1, data = js_data)

summary(jsi_mod)
```

```
##
## Call:
## lm(formula = y ~ xa1 * xb1 + xa2 * xb1, data = js_data)
##
## Residuals:
##                       1                       2                       3
##  116.49999999999974420 -116.49999999999977263   -0.00000000000002842
##                       4                       5                       6
##   70.50000000000002842  -70.50000000000002842 -204.49999999999997158
##                       7                       8                       9
##  204.50000000000002842   70.50000000000001421  -70.50000000000000000
##                      10                      11
## -133.00000000000000000  133.00000000000000000
##
## Coefficients:
##             Estimate Std. Error t value   Pr(>|t|)
## (Intercept) 1206.500     56.821  21.233 0.00000429 ***
## xa1         -265.750     85.905  -3.094     0.0271 *
## xb1         -233.333     56.821  -4.106     0.0093 **
## xa2          293.500     77.434   3.790     0.0128 *
## xa1:xb1        5.083     85.905   0.059     0.9551
## xb1:xa2       10.833     77.434   0.140     0.8942
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 182.2 on 5 degrees of freedom
## Multiple R-squared:  0.8855, Adjusted R-squared:  0.7709
## F-statistic:  7.73 on 5 and 5 DF,  p-value: 0.02123
```

```
anova(jsi_mod)
```

```
## Analysis of Variance Table
##
## Response: y
##           Df Sum Sq Mean Sq F value   Pr(>F)
## xa1        1 120144  120144  3.6178 0.115557
## xb1        1 677489  677489 20.4009 0.006302 **
## xa2        1 484346  484346 14.5849 0.012386 *
## xa1:xb1    1    897     897  0.0270 0.875865
## xb1:xa2    1    650     650  0.0196 0.894197
## Residuals  5 166044   33209
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
jsi_preds <- predict(jsi_mod, se.fit = TRUE)
jsi_fit <- jsi_preds$fit
jsi_sdf <- jsi_preds$se.fit
jsi_resid <- resid(jsi_mod)
jsi_sresid <- rstandard(jsi_mod)

cbind(xa1, y, Fit = jsi_fit, `StDev Fit` = jsi_sdf, Residual = jsi_resid,
      `St Resid` = jsi_sresid)
```

```
##    xa1    y    Fit StDev Fit                   Residual   St Resid
## 1    1  829  712.5  128.8581  116.49999999999974420462  0.9040956
## 2    1  596  712.5  128.8581 -116.49999999999977262632 -0.9040956
## 3    1 1169 1169.0  182.2328   -0.00000000000002842171        NaN
## 4    0 1348 1277.5  128.8581   70.50000000000002842171  0.5471136
## 5    0 1207 1277.5  128.8581  -70.50000000000002842171 -0.5471136
## 6    0 1518 1722.5  128.8581 -204.49999999999997157829 -1.5870175
## 7    0 1927 1722.5  128.8581  204.50000000000002842171  1.5870175
## 8   -1 1000  929.5  128.8581   70.50000000000001421085  0.5471136
## 9   -1  859  929.5  128.8581  -70.50000000000000000000 -0.5471136
## 10  -1 1295 1428.0  128.8581 -133.00000000000000000000 -1.0321434
## 11  -1 1561 1428.0  128.8581  133.00000000000000000000  1.0321434
```

Printout 5: Multiple Regression Version of the No-Interactions Factorial Analysis of Joint Strength (p. 712-13)

```
jsn_mod <- lm(y ~ xa1 + xa2 + xb1, data = js_data)

summary(jsn_mod)
```

```
##
## Call:
## lm(formula = y ~ xa1 + xa2 + xb1, data = js_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -215.969  -99.234   -7.625  101.141  193.031
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept)  1207.14      47.38  25.480 0.0000000366 ***
## xa1          -264.48      70.62  -3.745      0.00722 **
## xa2           292.86      65.11   4.498      0.00281 **
## xb1          -233.97      47.38  -4.939      0.00168 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 154.7 on 7 degrees of freedom
## Multiple R-squared:  0.8844, Adjusted R-squared:  0.8348
## F-statistic: 17.85 on 3 and 7 DF,  p-value: 0.001167
```

```
anova(jsn_mod)
```

```
## Analysis of Variance Table
##
## Response: y
##           Df Sum Sq Mean Sq F value   Pr(>F)
## xa1        1 120144  120144  5.0182 0.060066 .
## xa2        1 577927  577927 24.1390 0.001727 **
## xb1        1 583908  583908 24.3888 0.001678 **
## Residuals  7 167591   23942
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
jsn_preds <- predict(jsn_mod, se.fit = TRUE)
jsn_fit <- jsn_preds$fit
jsn_sdf <- jsn_preds$se.fit
jsn_resid <- resid(jsn_mod)
jsn_sresid <- rstandard(jsn_mod)

cbind(xa1, y, Fit = jsn_fit, `StDev Fit` = jsn_sdf, Residual = jsn_resid,
      `St Resid` = jsn_sresid)
```

```
##    xa1    y       Fit StDev Fit   Residual    St Resid
## 1    1  829  708.6875  94.75289  120.31250  0.98354422
## 2    1  596  708.6875  94.75289 -112.68750 -0.92121051
## 3    1 1169 1176.6250 109.41122   -7.62500 -0.06969121
## 4    0 1348 1266.0313  90.71899   81.96875  0.65393834
## 5    0 1207 1266.0313  90.71899  -59.03125 -0.47094530
## 6    0 1518 1733.9688  90.71899 -215.96875 -1.72297670
## 7    0 1927 1733.9688  90.71899  193.03125  1.53998366
## 8   -1 1000  944.7812  90.71899   55.21875  0.44052957
## 9   -1  859  944.7812  90.71899  -85.78125 -0.68435408
## 10  -1 1295 1412.7188  90.71899 -117.71875 -0.93914820
## 11  -1 1561 1412.7188  90.71899  148.28125  1.18297271
```

Printout 6: Multiple Regression Version of the With-Interactions Factorial Analysis of Power Requirement (p. 717-18)

```
xa2 <- c(rep(c(-1, 1), times = c(3, 4)), rep(rep(c(-1, 1), each = 4), 3))
xb2 <- c(rep(c(-1, 1), times = c(7, 8)), rep(c(-1, 1), each = 8))
xc2 <- rep(c(-1, 1), times = c(15, 16))
y <- c(26.5, 30.5, 27.0, 28.0, 28.5, 28.0, 25.0, 28.5, 28.5, 30.0, 32.5,
       29.5, 32.0, 29.0, 28.0, 28.0, 25.0, 26.5, 26.5, 24.5, 25.0, 28.0,
       26.0, 27.0, 29.0, 27.5, 27.5, 27.5, 28.0, 27.0, 26.0)

dum_data <- data.frame(xa2, xb2, xc2)
dumint_mod <- lm(y ~ xa2*xb2*xc2)

summary(dumint_mod)
```

```
##
## Call:
## lm(formula = y ~ xa2 * xb2 * xc2)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -2.375 -1.062 -0.125  0.750  2.625
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept) 27.76563    0.27307 101.678 < 0.0000000000000002 ***
## xa2         -0.26563    0.27307  -0.973              0.34081
## xb2          0.82812    0.27307   3.033              0.00592 **
## xc2         -0.95313    0.27307  -3.490              0.00197 **
```

```
## xa2:xb2      0.04688    0.27307   0.172              0.86521
## xa2:xc2     -0.04687    0.27307  -0.172              0.86521
## xb2:xc2     -0.20312    0.27307  -0.744              0.46450
## xa2:xb2:xc2 -0.04688    0.27307  -0.172              0.86521
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.514 on 23 degrees of freedom
## Multiple R-squared:  0.5096, Adjusted R-squared:  0.3603
## F-statistic: 3.414 on 7 and 23 DF,  p-value: 0.01192
```

```
anova(dumint_mod)
```

```
## Analysis of Variance Table
##
## Response: y
##             Df Sum Sq Mean Sq F value   Pr(>F)
## xa2          1  2.202  2.2022  0.9613 0.337057
## xb2          1 22.645 22.6450  9.8853 0.004546 **
## xc2          1 28.398 28.3975 12.3965 0.001832 **
## xa2:xb2      1  0.091  0.0913  0.0399 0.843508
## xa2:xc2      1  0.051  0.0515  0.0225 0.882165
## xb2:xc2      1  1.293  1.2931  0.5645 0.460082
## xa2:xb2:xc2  1  0.068  0.0675  0.0295 0.865208
## Residuals   23 52.687  2.2908
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
dumint_preds <- predict(dumint_mod, se.fit = TRUE)
dumint_fit <- dumint_preds$fit
dumint_sdf <- dumint_preds$se.fit
dumint_resid <- resid(dumint_mod)
dumint_sresid <- rstandard(dumint_mod)

cbind(xa2, y, Fit = dumint_fit, `StDev Fit` = dumint_sdf, Residual = dumint_resid,
      `St Resid` = dumint_sresid)
```

```
##    xa2    y    Fit StDev Fit                   Residual    St Resid
## 1   -1 26.5 28.000 0.8738346 -1.4999999999999864552791 -1.21379966
## 2   -1 30.5 28.000 0.8738346  2.4999999999999955591079  2.02299943
## 3   -1 27.0 28.000 0.8738346 -1.0000000000000053290705 -0.80919977
## 4    1 28.0 27.375 0.7567630  0.6250000000000034416914  0.47682554
## 5    1 28.5 27.375 0.7567630  1.1249999999999984456878  0.85828597
## 6    1 28.0 27.375 0.7567630  0.6249999999999994448885  0.47682554
## 7    1 25.0 27.375 0.7567630 -2.3750000000000013322676 -1.81193705
## 8   -1 28.5 29.875 0.7567630 -1.3750000000000011102230 -1.04901618
## 9   -1 28.5 29.875 0.7567630 -1.3750000000000006661338 -1.04901618
## 10  -1 30.0 29.875 0.7567630  0.1249999999999999722444  0.09536511
## 11  -1 32.5 29.875 0.7567630  2.6250000000000000000000  2.00266726
## 12   1 29.5 29.625 0.7567630 -0.1250000000000000277556 -0.09536511
## 13   1 32.0 29.625 0.7567630  2.3750000000000008881784  1.81193705
## 14   1 29.0 29.625 0.7567630 -0.6250000000000001110223 -0.47682554
```

```
## 15   1 28.0 29.625 0.7567630 -1.624999999999997779554 -1.23974640
## 16  -1 28.0 26.500 0.7567630  1.499999999999997779554  1.14438129
## 17  -1 25.0 26.500 0.7567630 -1.499999999999997779554 -1.14438129
## 18  -1 26.5 26.500 0.7567630  0.000000000000001110223  0.00000000
## 19  -1 26.5 26.500 0.7567630  0.000000000000001110223  0.00000000
## 20   1 24.5 25.875 0.7567630 -1.375000000000002220446 -1.04901618
## 21   1 25.0 25.875 0.7567630 -0.875000000000001110223 -0.66755575
## 22   1 28.0 25.875 0.7567630  2.125000000000000000000  1.62120683
## 23   1 26.0 25.875 0.7567630  0.124999999999998612221  0.09536511
## 24  -1 27.0 27.750 0.7567630 -0.749999999999998889777 -0.57219065
## 25  -1 29.0 27.750 0.7567630  1.249999999999997779554  0.95365108
## 26  -1 27.5 27.750 0.7567630 -0.250000000000000000000 -0.19073022
## 27  -1 27.5 27.750 0.7567630 -0.250000000000000000000 -0.19073022
## 28   1 27.5 27.125 0.7567630  0.374999999999996669331  0.28609532
## 29   1 28.0 27.125 0.7567630  0.874999999999998889777  0.66755575
## 30   1 27.0 27.125 0.7567630 -0.125000000000003608225 -0.09536511
## 31   1 26.0 27.125 0.7567630 -1.125000000000006661338 -0.85828597
```

Printout 7: Multiple Regression Version of a "B and C Main Effects Only" Analysis of Power Requirement (p. 718-19)

```
dumbc_mod <- lm(y ~ xb2 + xc2)
summary(dumbc_mod)
```

```
##
## Call:
## lm(formula = y ~ xb2 + xc2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8793 -0.9806 -0.1444  0.5194  2.9569
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)  27.7619     0.2553 108.732 < 0.0000000000000002 ***
## xb2           0.8319     0.2553   3.258              0.00294 **
## xc2          -0.9494     0.2553  -3.718              0.00089 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.42 on 28 degrees of freedom
## Multiple R-squared:  0.4744, Adjusted R-squared:  0.4369
## F-statistic: 12.64 on 2 and 28 DF,  p-value: 0.0001226
```

```
anova(dumbc_mod)
```

```
## Analysis of Variance Table
##
## Response: y
##           Df Sum Sq Mean Sq F value    Pr(>F)
## xb2        1 23.093 23.0928  11.452 0.0021284 **
## xc2        1 27.879 27.8793  13.825 0.0008897 ***
## Residuals 28 56.463  2.0165
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
dumbc_preds <- predict(dumbc_mod, se.fit = TRUE)
dumbc_fit <- dumbc_preds$fit
dumbc_sdf <- dumbc_preds$se.fit
dumbc_resid <- resid(dumbc_mod)
dumbc_sresid <- rstandard(dumbc_mod)
cbind(xb2, y, Fit = dumbc_fit, `StDev Fit` = dumbc_sdf, Residual = dumbc_resid,
      `St Resid` = dumbc_sresid)
```

```
##    xb2    y      Fit StDev Fit    Residual    St Resid
## 1   -1 26.5 27.87931 0.4567368 -1.37931034 -1.02581718
## 2   -1 30.5 27.87931 0.4567368  2.62068966  1.94905264
## 3   -1 27.0 27.87931 0.4567368 -0.87931034 -0.65395845
## 4   -1 28.0 27.87931 0.4567368  0.12068966  0.08975900
## 5   -1 28.5 27.87931 0.4567368  0.62068966  0.46161773
## 6   -1 28.0 27.87931 0.4567368  0.12068966  0.08975900
## 7   -1 25.0 27.87931 0.4567368 -2.87931034 -2.14139336
## 8    1 28.5 29.54310 0.4372922 -1.04310345 -0.77207124
## 9    1 28.5 29.54310 0.4372922 -1.04310345 -0.77207124
## 10   1 30.0 29.54310 0.4372922  0.45689655  0.33817996
## 11   1 32.5 29.54310 0.4372922  2.95689655  2.18859864
## 12   1 29.5 29.54310 0.4372922 -0.04310345 -0.03190377
## 13   1 32.0 29.54310 0.4372922  2.45689655  1.81851490
## 14   1 29.0 29.54310 0.4372922 -0.54310345 -0.40198750
## 15   1 28.0 29.54310 0.4372922 -1.54310345 -1.14215497
## 16  -1 28.0 25.98060 0.4372922  2.01939655  1.49469163
## 17  -1 25.0 25.98060 0.4372922 -0.98060345 -0.72581077
## 18  -1 26.5 25.98060 0.4372922  0.51939655  0.38444043
## 19  -1 26.5 25.98060 0.4372922  0.51939655  0.38444043
## 20  -1 24.5 25.98060 0.4372922 -1.48060345 -1.09589451
## 21  -1 25.0 25.98060 0.4372922 -0.98060345 -0.72581077
## 22  -1 28.0 25.98060 0.4372922  2.01939655  1.49469163
## 23  -1 26.0 25.98060 0.4372922  0.01939655  0.01435670
## 24   1 27.0 27.64440 0.4372922 -0.64439655 -0.47696136
## 25   1 29.0 27.64440 0.4372922  1.35560345  1.00337357
## 26   1 27.5 27.64440 0.4372922 -0.14439655 -0.10687763
## 27   1 27.5 27.64440 0.4372922 -0.14439655 -0.10687763
## 28   1 27.5 27.64440 0.4372922 -0.14439655 -0.10687763
## 29   1 28.0 27.64440 0.4372922  0.35560345  0.26320610
## 30   1 27.0 27.64440 0.4372922 -0.64439655 -0.47696136
## 31   1 26.0 27.64440 0.4372922 -1.64439655 -1.21712883
```

Chapter 4 Selected Questions

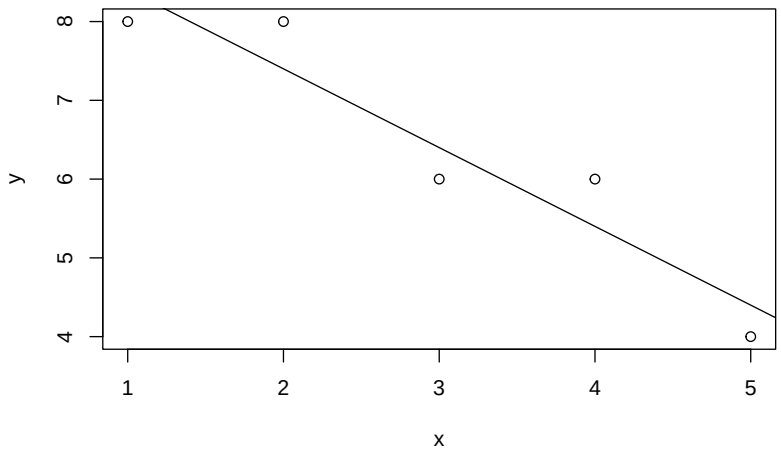
Question 4.1.2 (p. 139-40)

(a)

```
x <- 1:5
y <- c(8, 8, 6, 6, 4)

mod4.2 <- lm(y ~ x)

#print the coefficients
coef(mod4.2)
```

```
## (Intercept)           x
##         9.4        -1.0
```

```
#plot the points and the line of best fit
plot(x, y)
abline(mod4.2)
```



(b)

```
#calculate the correlation
cor(x, y)
```

```
## [1] -0.9449112
```

(c)

```
#calculate y_hat and the correlation of y and y_hat
y_hat <- predict(mod4.2)
cor(y, y_hat)
```

```
## [1] 0.9449112
```

(d)

```
#get R^2 from the fitted model
mod4.2_R2 <- summary(mod4.2)$r.squared
mod4.2_R2
```

```
## [1] 0.8928571
```

```
#see that the squares of the previous correlations are the same as R^2
cor(x, y)^2
```

```
## [1] 0.8928571
```

```
cor(y, y_hat)^2
```

```
## [1] 0.8928571
```

(e)

```
#calculate the residuals
resid(mod4.2)
```

```
##    1    2    3    4    5
## -0.4  0.6 -0.4  0.6 -0.4
```

Question 4.1.3 (p. 140)

```
#save data and fit the model
temp <- c(165, 176, 188, 205, 220, 235, 250, 260)
weight <- c(808, 940, 1183, 1545, 2012, 2362, 2742, 2935)

wt_mod <- lm(weight ~ temp)
```

(a)

```
#get R^2
wt_mod_R2 <- summary(wt_mod)$r.squared
wt_mod_R2
```

```
## [1] 0.9942725
```

(b)

```
#both fitted coefficients
coef(wt_mod)
```

```
## (Intercept)        temp
## -3174.56922    23.49827
```

```
#fitted coefficient for temperature
coef(wt_mod)[2]
```

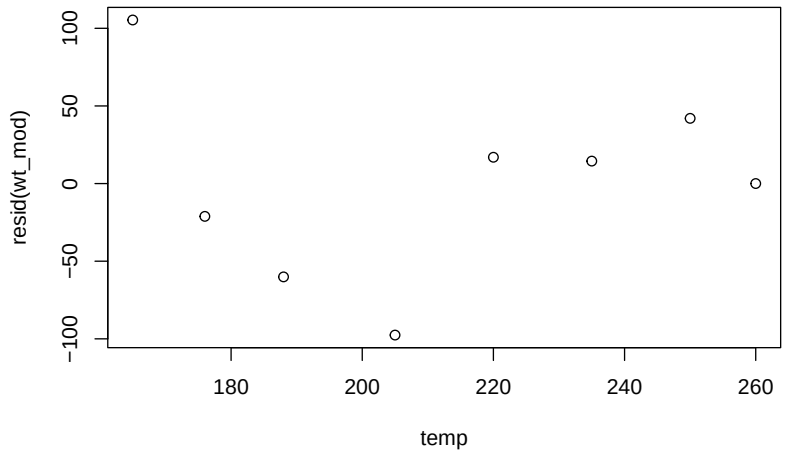
```
##     temp
## 23.49827
```

(c)

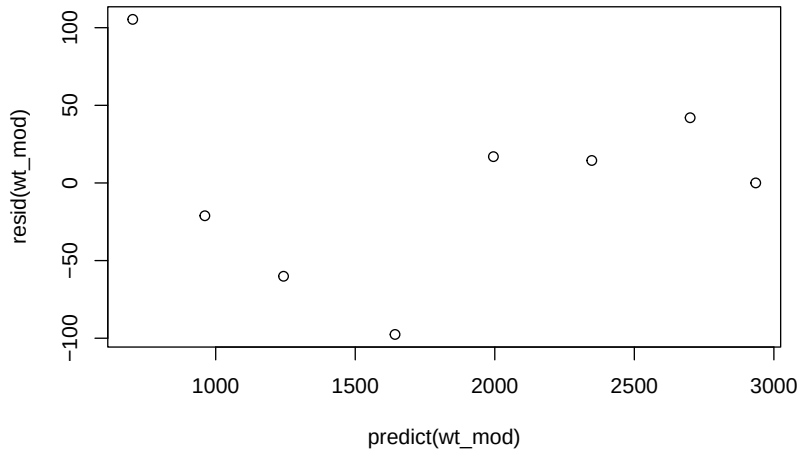
```
#obtain the residuals
resid(wt_mod)
```

```
##            1            2            3            4            5            6
## 105.35534763 -21.12557741 -60.10476837 -97.57528889  16.95072241  14.47673372
##            7            8
##  42.00274502   0.02008589
```

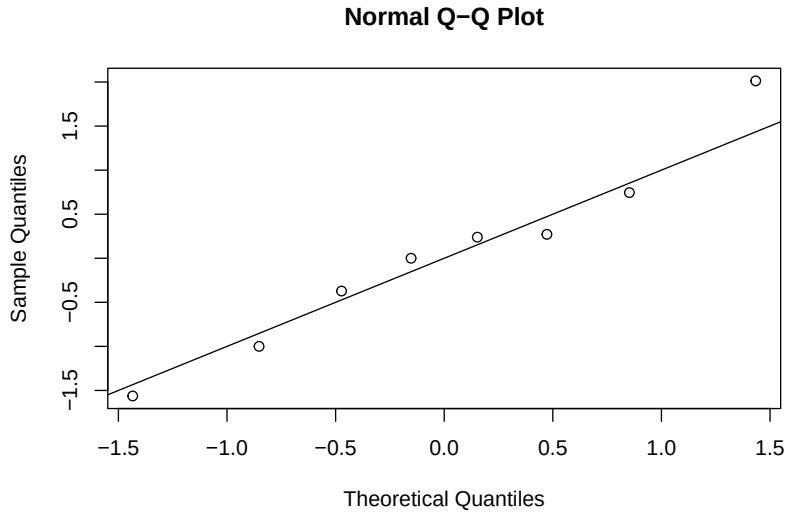
```
#plot the residuals vs. x
plot(temp, resid(wt_mod))
```




```
#plot the residuals vs. y_hat
plot(predict(wt_mod), resid(wt_mod))
```




```
#normal plot of the standardized residuals
qqnorm(rstandard(wt_mod))
abline(0, 1)
```



(e)

```
#predict for temperature = 188 degrees
predict(wt_mod, newdata = list(temp = 188))
```

```
##        1
## 1243.105
```

```
#predict for temperature = 200 degrees
predict(wt_mod, newdata = list(temp = 200))
```

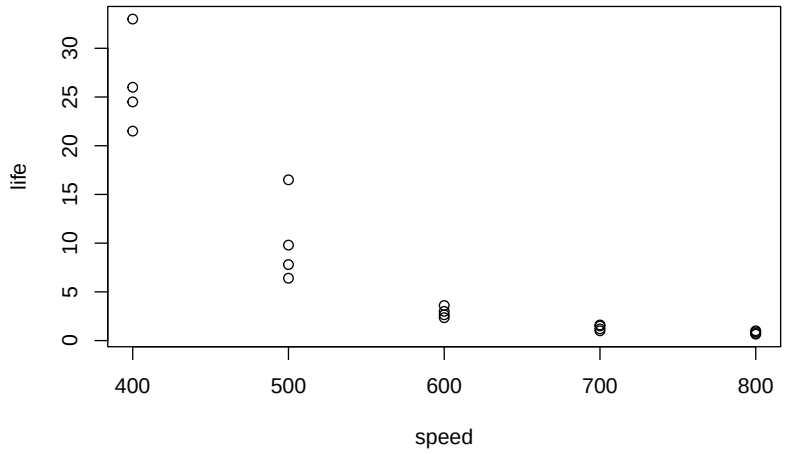
```
##        1
## 1525.084
```

Question 4.1.4 (p. 140)

```
speed <- rep(8:4*100, each = 4)
life <- c(1.00, 0.90, 0.74, 0.66, 1.00, 1.20, 1.50, 1.60, 2.35, 2.65, 3.00,
          3.60, 6.40, 7.80, 9.80, 16.50, 21.50, 24.50, 26.00, 33.00)

lf_mod <- lm(life ~ speed)
```

(a)

```
plot(speed, life)
```



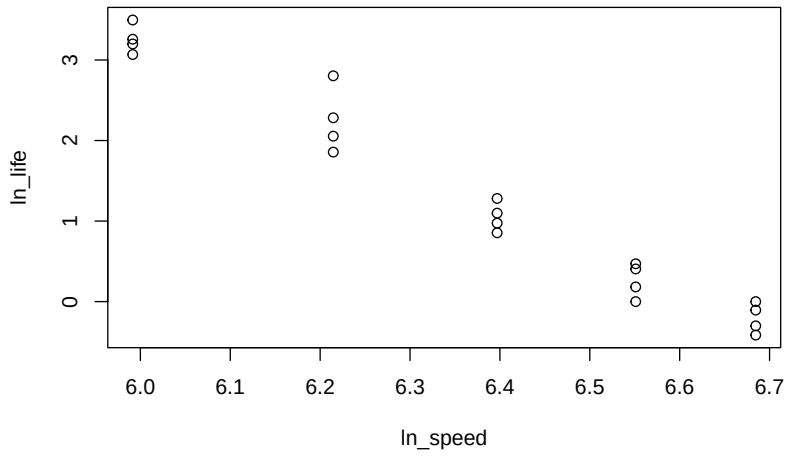

```
#R^2
lf_mod_R2 <- summary(lf_mod)$r.squared
lf_mod_R2
```

```
## [1] 0.7227132
```

(b)

```
#take logs of variables
ln_speed <- log(speed)
ln_life <- log(life)
```

```
#model with transformed variables
lnlf_mod <- lm(ln_life ~ ln_speed)

plot(ln_speed, ln_life)
```




```
#R^2
lnlf_mod_R2 <- summary(lnlf_mod)$r.squared
lnlf_mod_R2
```

```
## [1] 0.9650491
```

(c)

```
ln_pred <- predict(lnlf_mod, newdata = list(ln_speed = log(550)))

#exponentiate to get the prediction
exp(ln_pred)
```

```
##        1
## 5.067917
```

Question 4.2.1 (p. 161)

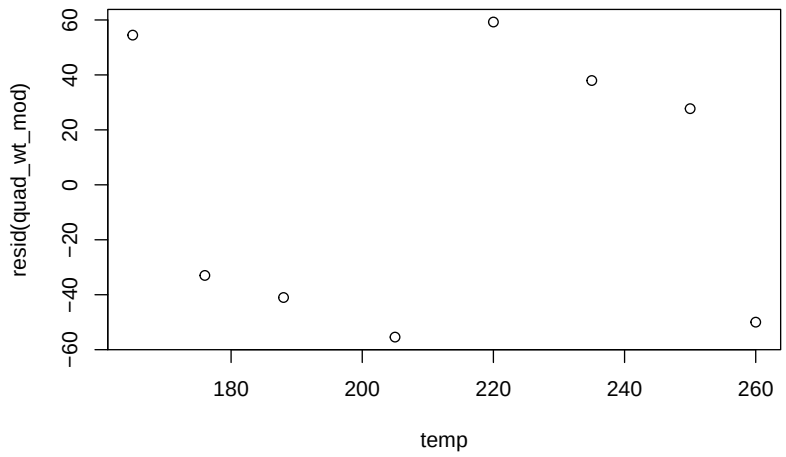
```
#weight and temp vectors were saved earlier
temp_2 <- temp^2

#quadratic model
quad_wt_mod <- lm(weight ~ temp + temp_2)
summary(quad_wt_mod)
```

```
##
## Call:
## lm(formula = weight ~ temp + temp_2)
##
## Residuals:
##      1      2      3      4      5      6      7      8
##  54.48 -32.98 -41.02 -55.42  59.25  37.96  27.71 -49.98
##
## Coefficients:
##                Estimate  Std. Error t value Pr(>|t|)
## (Intercept) -1315.25426  1098.14130  -1.198    0.285
## temp            5.58831    10.51533   0.531    0.618
## temp_2          0.04212     0.02468   1.706    0.149
##
## Residual standard error: 58.35 on 5 degrees of freedom
## Multiple R-squared:  0.9964, Adjusted R-squared:  0.9949
## F-statistic: 688.2 on 2 and 5 DF,  p-value: 0.0000007883
```

```
#R^2
quad_wt_mod_R2 <- summary(quad_wt_mod)$r.squared
quad_wt_mod_R2
```

```
## [1] 0.9963803
```

```
plot(temp, resid(quad_wt_mod))
```




```
#predict for temperature = 200 degrees
predict(quad_wt_mod, newdata = list(temp = 200, temp_2 = 200^2))
```

```
##        1
## 1487.185
```

Question 4.2.2 (p. 161-62)

```
x1 <- rep(c(3, 9, 15), each = 3) #%NaOH
x2 <- rep(c(30, 60, 90), 3) #Time
s_area <- c(5.95, 5.60, 5.44, 6.22, 5.85, 5.61, 8.36, 7.30, 6.43) #surface area
```

(a)

```
area_mod <- lm(s_area ~ x1 + x2)

#model summary
summary(area_mod)
```

```
##
## Call:
## lm(formula = s_area ~ x1 + x2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5950 -0.2183 -0.0150  0.1433  0.6950
##
## Coefficients:
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  6.048333   0.520812  11.613 0.0000245 ***
## x1           0.141667   0.033005   4.292   0.00514 **
## x2          -0.016944   0.006601  -2.567   0.04251 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4851 on 6 degrees of freedom
## Multiple R-squared:  0.8065, Adjusted R-squared:  0.742
## F-statistic: 12.51 on 2 and 6 DF,  p-value: 0.007242
```

```
#R^2
area_mod_R2 <- summary(area_mod)$r.squared
area_mod_R2
```

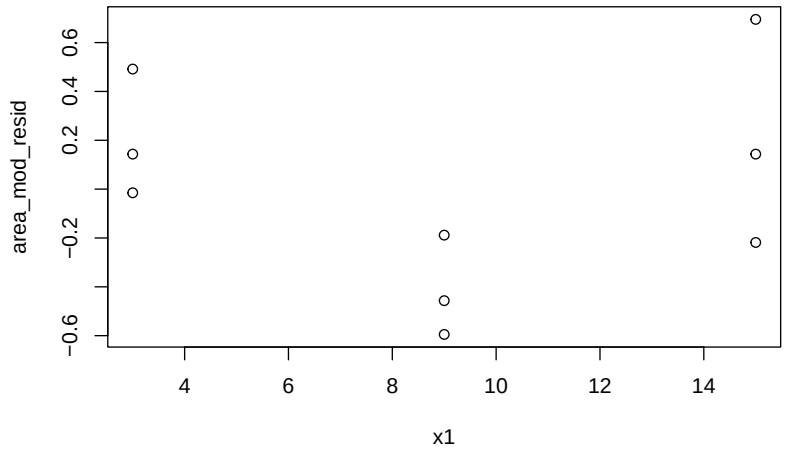
```
## [1] 0.8065308
```

(b)

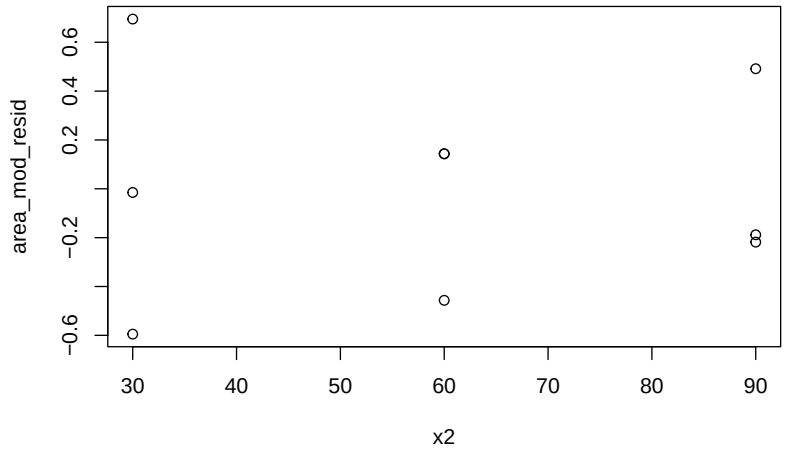
```
area_mod_resid <- resid(area_mod)
area_mod_resid
```

```
##          1          2          3          4          5          6          7
## -0.0150000  0.1433333  0.4916667 -0.5950000 -0.4566667 -0.1883333  0.6950000
##          8          9
##  0.1433333 -0.2183333
```

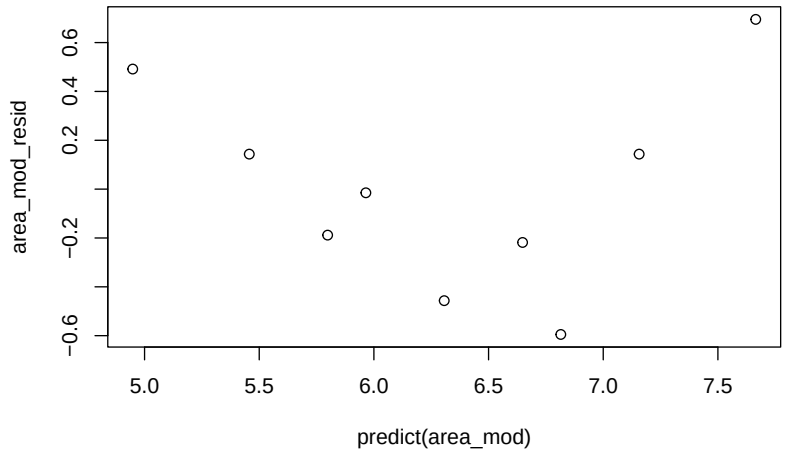
```
#plot the residuals vs. x1
plot(x1, area_mod_resid)
```



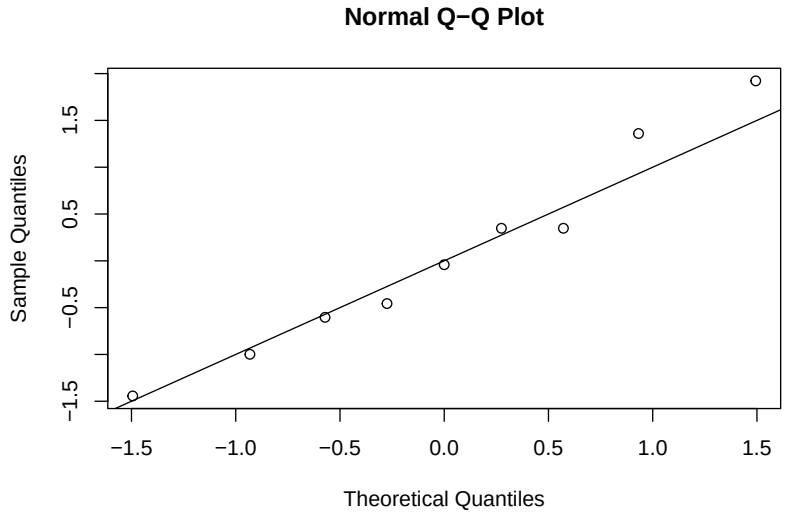

```
#plot the residuals vs. x2
plot(x2, area_mod_resid)
```




```
#plot the residuals vs. y_hat
plot(predict(area_mod), area_mod_resid)
```



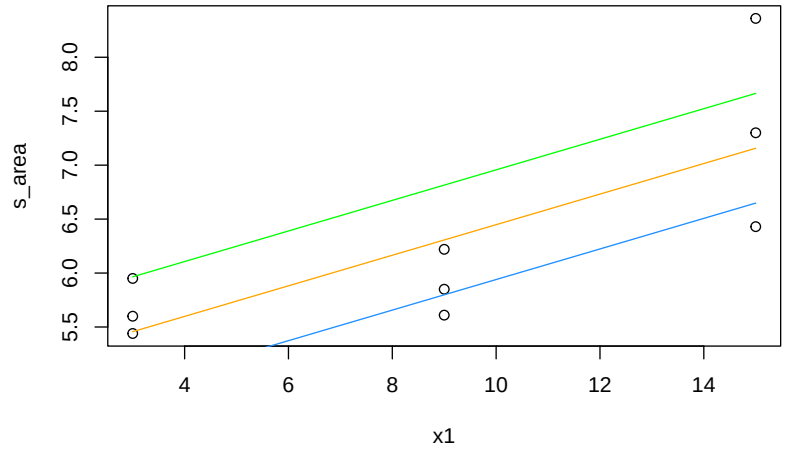

```
#normal plot of the standardized residuals
qqnorm(rstandard(area_mod))
abline(0, 1)
```



(c)

```
#long vector used for plotting a line
x1_range <- seq(3, 15, length.out = 1001)
x2_30 <- 30
x2_60 <- 60
x2_90 <- 90
```

```
#predicted values for x2 = 30, 60, and 90
l1 <- coef(area_mod)[1] + coef(area_mod)[2]*x1_range + coef(area_mod)[3]*x2_30
l2 <- coef(area_mod)[1] + coef(area_mod)[2]*x1_range + coef(area_mod)[3]*x2_60
l3 <- coef(area_mod)[1] + coef(area_mod)[2]*x1_range + coef(area_mod)[3]*x2_90

plot(x1, s_area)
lines(x1_range, l1, col = "green")
lines(x1_range, l2, col = "orange")
lines(x1_range, l3, col = "dodgerblue")
```



(d)

```
#prediction for x1 = 10 and x2 = 70
predict(area_mod, newdata = list(x1 = 10, x2 = 70))
```

```
##        1
## 6.278889
```

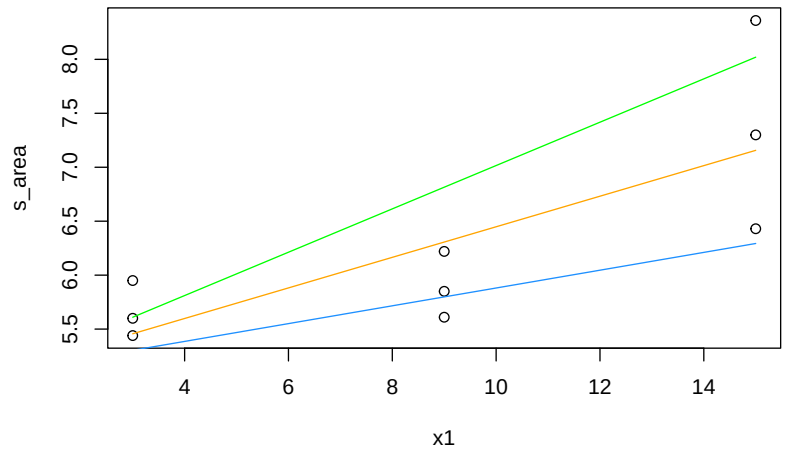
(e)

```
#interaction model
area_int_mod <- lm(s_area ~ x1*x2)
summary(area_int_mod)$r.squared
```

```
## [1] 0.8756121
```

(f)

```
b0 <- coef(area_int_mod)[1]
b1 <- coef(area_int_mod)[2]
b2 <- coef(area_int_mod)[3]
b3 <- coef(area_int_mod)[4]

nl1 <- b0 + b1*x1_range + b2*x2_30 + b3*x1_range*x2_30
nl2 <- b0 + b1*x1_range + b2*x2_60 + b3*x1_range*x2_60
nl3 <- b0 + b1*x1_range + b2*x2_90 + b3*x1_range*x2_90

plot(x1, s_area)
lines(x1_range, nl1, col = "green")
lines(x1_range, nl2, col = "orange")
lines(x1_range, nl3, col = "dodgerblue")
```



(h)

```
area_mod_x1 <- lm(s_area ~ x1)
area_mod_x2 <- lm(s_area ~ x2)

summary(area_mod_x1)
```

```
##
## Call:
## lm(formula = s_area ~ x1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.72667 -0.45667 -0.01667  0.14333  1.20333
##
```

```
## Coefficients:
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  5.03167    0.45355  11.094 0.0000107 ***
## x1           0.14167    0.04426   3.201    0.0151 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6505 on 7 degrees of freedom
## Multiple R-squared:  0.5941, Adjusted R-squared:  0.5361
## F-statistic: 10.24 on 1 and 7 DF,  p-value: 0.01505
```

```
summary(area_mod_x2)
```

```
##
## Call:
## lm(formula = s_area ~ x2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8650 -0.5950 -0.3583  0.6317  1.5450
##
## Coefficients:
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  7.32333    0.79908   9.165 0.0000379 ***
## x2          -0.01694    0.01233  -1.374     0.212
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9061 on 7 degrees of freedom
## Multiple R-squared:  0.2125, Adjusted R-squared:  0.09996
## F-statistic: 1.889 on 1 and 7 DF,  p-value: 0.2118
```

Chapter 7 Selected Questions

Question 7.1.1 (p. 460)

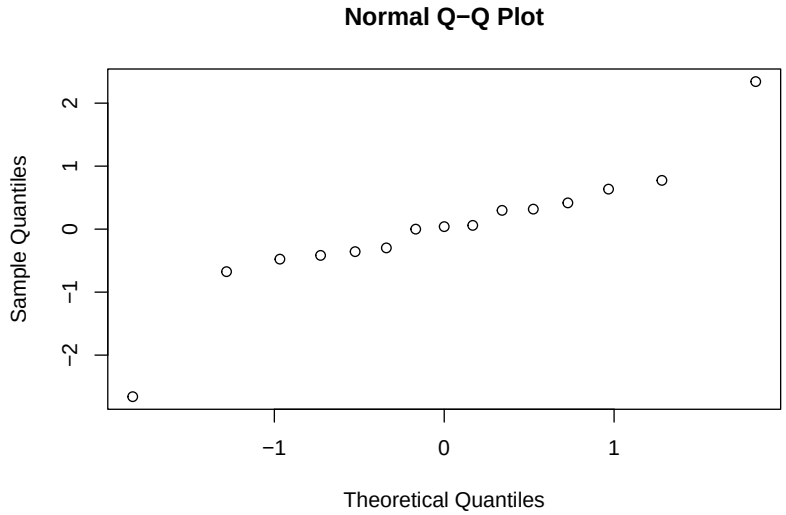
```
pressure <- rep(c(2000, 4000, 6000, 8000, 10000), each = 3)
density <- c(2.486, 2.479, 2.472, 2.558, 2.570, 2.580, 2.646, 2.657,
             2.653, 2.724, 2.774, 2.808, 2.861, 2.879, 2.858)
```

(a)

```
den_mod <- lm(density ~ factor(pressure))
den_mod_stresids <- rstandard(den_mod)
den_mod_stresids
```

```
##                          1                          2                          3
##  0.41677825300621462734  0.00000000000005148757 -0.41677825300626597516
##                          4                          5                          6
## -0.67478383820068188470  0.03969316695297527009  0.63509067124770679502
##                          7                          8                          9
## -0.35723850257683459342  0.29769875214736052360  0.05953975042947377838
##                         10                         11                         12
## -2.65944218584972125186  0.31754533562384845702  2.34189685022587301688
##                         13                         14                         15
## -0.29769875214735180835  0.77401675558312543757 -0.47631800343577340717
```

```
#normal plot of the standardized residuals
qqnorm(den_mod_stresids)
```



(b)

```
#get s_P
den_mod_s_P <- summary(den_mod)$sigma
den_mod_s_P
```

```
## [1] 0.0205702
```

```
#lower 95% confidence bound for sigma
den_mod_s_P_low <- sqrt(sum(den_mod$residuals^2)/qchisq(0.95,df=10))

#upper 95% confidence bound for sigma
den_mod_s_P_upp <- sqrt(sum(den_mod$residuals^2)/qchisq(0.05,df=10))

#90% confidence interval for sigma
c(den_mod_s_P_low, den_mod_s_P_upp)
```

```
## [1] 0.01520301 0.03276982
```

Question 7.1.2 (p. 460)

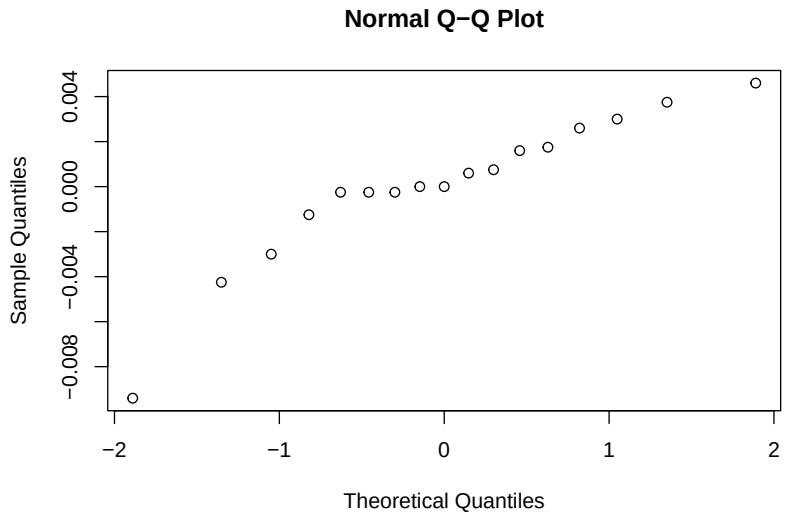
```
vans <- factor(c(rep(1:2, each = 4), rep(3, 5), rep(4, 4)))
tilt <- c(1.096, 1.090, 1.093, 1.093, .962, .967, .970, .966, 1.010,
          1.021, 1.022, 1.024, 1.020, 1.002, 1.002, 1.001, 1.004)
```

(a)

```
tilt_mod <- lm(tilt ~ vans)

#save and print the residuals
tilt_mod_resid <- resid(tilt_mod)
tilt_mod_resid
```

```
##                            1                            2
##  0.00299999999999971686884 -0.00299999999999983699844
##                            3                            4
##  0.00000000000000007285839  0.00000000000000004683753
##                            5                            6
## -0.00424999999999999857059  0.00074999999999999871457
##                            7                            8
##  0.00375000000000000072858 -0.00025000000000000217361
##                            9                           10
## -0.00939999999999998475664  0.00159999999999991290682
##                           11                           12
##  0.00260000000000002416678  0.00460000000000002594314
##                           13                           14
##  0.00060000000000002293252 -0.00024999999999997224963
##                           15                           16
## -0.00024999999999997224963 -0.00125000000000008416011
##                           17
##  0.00175000000000002974357
```

```
#normal plot of the residuals
qqnorm(tilt_mod_resid)
```

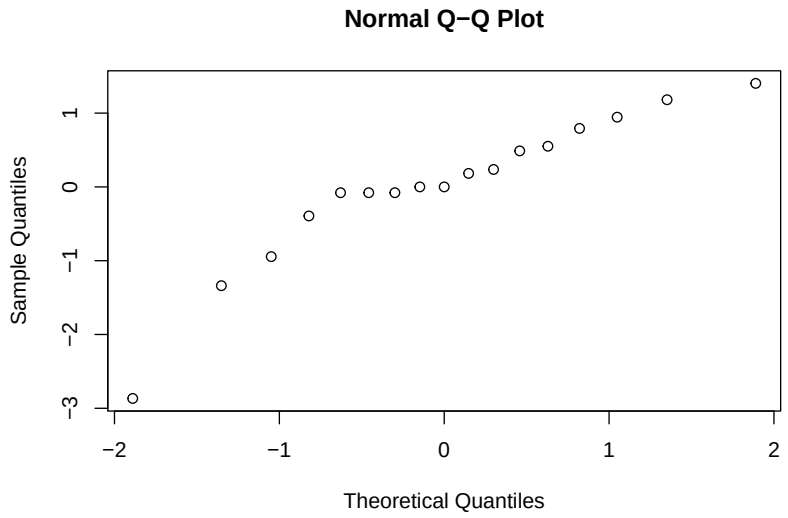


(b)

```
tilt_mod_stresids <- rstandard(tilt_mod)
tilt_mod_stresids
```

```
##                         1                         2                         3
##  0.94496526862145457937 -0.94496526862149221593  0.00000000000002294955
##                         4                         5                         6
##  0.00000000000001475328 -1.33870079721385315530  0.23624131715538551624
##                         7                         8                         9
##  1.18120658577692982938 -0.07874710571846266216 -2.86687055269059332119
##                        10                        11                        12
##  0.48797796641539437079  0.79296419542506635381  1.40293665344434304032
##                        13                        14                        15
##  0.18299173740578983383 -0.07874710571845323914 -0.07874710571845323914
##                        16                        17
## -0.39373552859233640344  0.55122974002924318704
```

```
#normal plot of the standardized residuals
qqnorm(tilt_mod_stresids)
```



(c)

```
#get s_P
tilt_mod_s_P <- summary(tilt_mod)$sigma
tilt_mod_s_P
```

```
## [1] 0.003665851
```

```
#lower 97.5% confidence bound for sigma
tilt_mod_s_P_low <- sqrt(sum(tilt_mod$residuals^2)/qchisq(0.975,df=13))

#upper 97.5% confidence bound for sigma
tilt_mod_s_P_upp <- sqrt(sum(tilt_mod$residuals^2)/qchisq(0.025,df=13))

#95% confidence interval for sigma
c(tilt_mod_s_P_low, tilt_mod_s_P_upp)
```

```
## [1] 0.002657573 0.005905841
```

Question 7.2.1 (p. 470-71)

(a)

```
#calculate delta
den_mod_s_P/sqrt(3)*qt(.975, 10)
```

```
## [1] 0.02646185
```

(b)

```
#calculate delta
den_mod_s_P*sqrt(2/3)*qt(.975, 10)
```

```
## [1] 0.03742271
```

(c)

```
#95% confidence interval for the contrast
cont <- c(0, -2, 1, 0, 0)
cont%*%coef(den_mod) +
  c(-1, 1)*qt(.975, 10)*sqrt(cont%*%vcov(den_mod)%*%cont)
```

```
## [1] 0.2203583 0.3363083
```

```
#alternatively, use the glmglrt package to calculate the
#95% contrast confidence interval
library(glmglrt)
confint_contrast(den_mod, contrast = cont)
```

```
##     lower     upper
## 0.2203583 0.3363083
```

Question 7.2.2 (p. 471)

(a)

```
#parameterize the model to not have an intercept
r_tilt_mod <- lm(tilt ~ -1 + vans)

#99% confidence intervals
confint(r_tilt_mod, level = .99)
```

```
##           0.5 %    99.5 %
## vans1 1.0874787 1.0985213
## vans2 0.9607287 0.9717713
## vans3 1.0144616 1.0243384
## vans4 0.9967287 1.0077713
```

(b)

```
#get s_P
r_tilt_mod_s_P <- summary(r_tilt_mod)$sigma

#delta for two samples of size 4
r_tilt_mod_s_P*sqrt(2/4)*qt(.995, 13)
```

```
## [1] 0.007808265
```

```
#delta for sample sizes 4 and 5
r_tilt_mod_s_P*sqrt((1/4) + (1/5))*qt(.995, 13)
```

```
## [1] 0.00740757
```

(c)

```
#make a contrast
cont <- c(1/2, 1/2, -1/2, -1/2)

#99% confidence interval
cont%*%coef(r_tilt_mod) +
  c(-1, 1)*qt(.995, 13)*sqrt(cont%*%vcov(r_tilt_mod)%*%cont)
```

```
## [1] 0.01341853 0.02418147
```

```
#alternative method to calculate the 99% confidence interval
confint_contrast(r_tilt_mod, contrast = cont, level = .99)
```

```
##      lower      upper
## 0.01341853 0.02418147
```

Chapter 9 Selected Questions

Question 9.1.1 (p. 674)

(a)

```
#get s_LF from the model from 4.1.3
wt_s_LF <- summary(wt_mod)$sigma
wt_s_LF
```

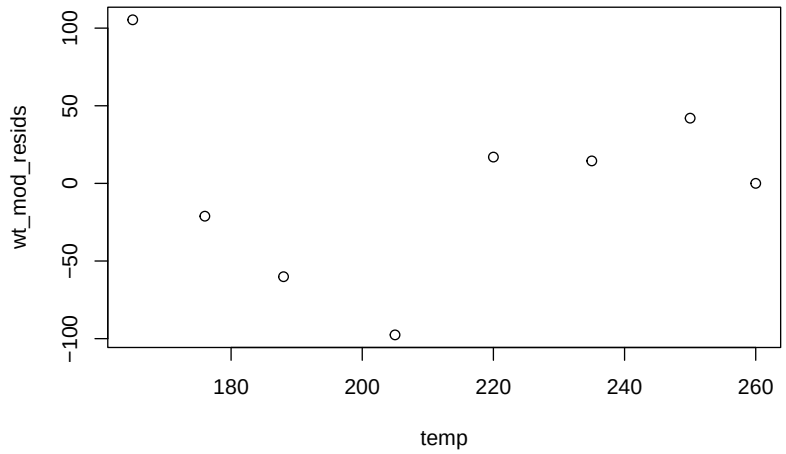
```
## [1] 67.00832
```

(b)

```
wt_mod_resids <- resid(wt_mod)
wt_mod_resids
```

```
##           1           2           3           4           5           6
## 105.35534763 -21.12557741 -60.10476837 -97.57528889  16.95072241  14.47673372
##           7           8
##  42.00274502   0.02008589
```

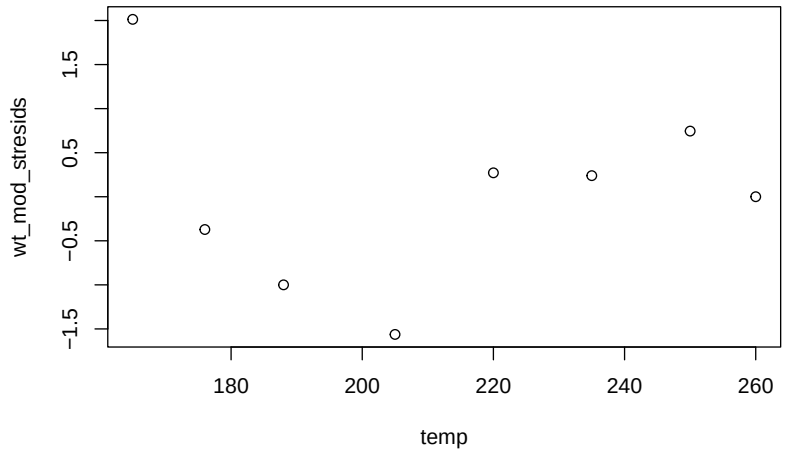
```
#plot the residuals vs. x
plot(temp, wt_mod_resids)
```



```
wt_mod_stresids <- rstandard(wt_mod)
wt_mod_stresids
```

```
##             1             2             3             4             5
##  2.0130640925 -0.3718613214 -0.9998204669 -1.5624516563  0.2714973975
##             6             7             8
##  0.2393753283  0.7450316322  0.0003846738
```

```
#plot the standardized residuals vs. x
plot(temp, wt_mod_stresids)
```



(c)

```
#90% confidence intervals for the temperature parameter
confint(wt_mod, parm = "temp", level = .9)
```

```
##           5 %     95 %
## temp 22.08344 24.91309
```

(d)

```
#fitted values and 90% confidence intervals
predict(wt_mod, newdata = list(temp = c(212, 250)), interval = "confidence",
        level = .9)
```

```
##        fit      lwr      upr
## 1 1807.063 1761.024 1853.102
## 2 2699.997 2629.619 2770.375
```

(f)

```
#use level = .8  to  get  the  90%  lower  prediction bound, lwr below
predict(wt_mod, newdata = list(temp = c(212, 250)), interval = "prediction",
        level = .8)
```

```
##        fit      lwr      upr
## 1 1807.063 1704.735 1909.392
## 2 2699.997 2590.331 2809.663
```

(h)

```
#ANOVA table
anova(wt_mod)
```

```
## Analysis of Variance Table
##
## Response: weight
##           Df  Sum Sq Mean Sq F value        Pr(>F)
## temp       1 4676798 4676798  1041.6 0.00000005884 ***
## Residuals  6   26941    4490
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Question 9.1.2 (p. 674)

```
ratio <- rep(c(.45, .50, .55), each = 3)
strength <- c(2954, 2913, 2923, 2743, 2779, 2739, 2652, 2607, 2583)
```

```
#model with ratio as a numeric variable
st_lmod <- lm(strength ~ ratio)
summary(st_lmod)$sigma
```

```
## [1] 26.75521
```

(a)

```
#get coefficients
coef(st_lmod)
```

```
## (Intercept)       ratio
##    4345.889   -3160.000
```

```
#model ratio as a categorical instead of a numeric variable
st_gmod <- lm(strength ~ factor(ratio))

#compare sigma and s_P
st_s_LF <- summary(st_lmod)$sigma
st_s_P <- summary(st_gmod)$sigma
```

```
st_s_LF
```

```
## [1] 26.75521
```

```
st_s_P
```

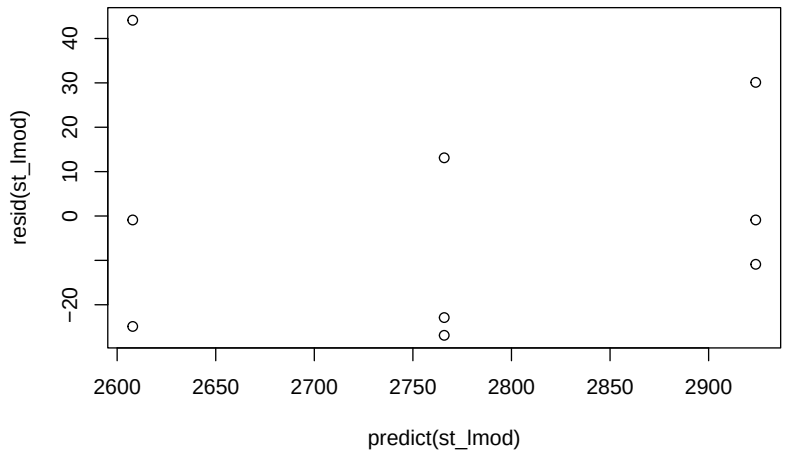
```
## [1] 26.89073
```

(b)

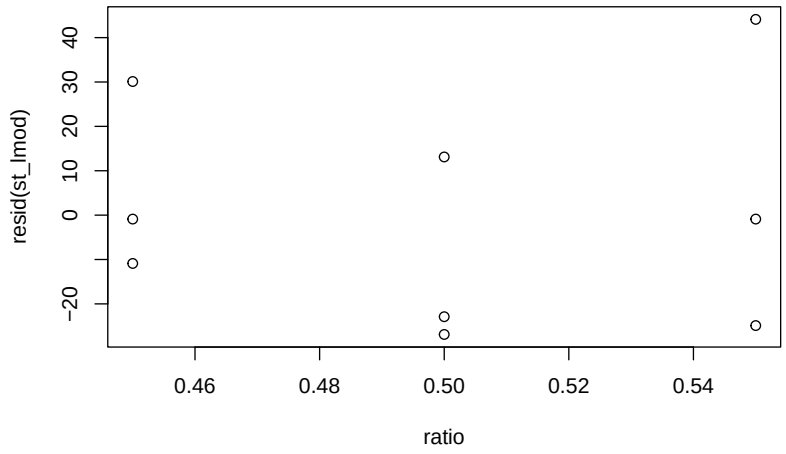
```
#residuals
resid(st_lmod)
```

```
##           1           2           3           4           5           6
##  30.1111111 -10.8888889  -0.8888889 -22.8888889  13.1111111 -26.8888889
##           7           8           9
##  44.1111111  -0.8888889 -24.8888889
```

```
#plot the residuals vs. y_hat
plot(predict(st_lmod), resid(st_lmod))
```



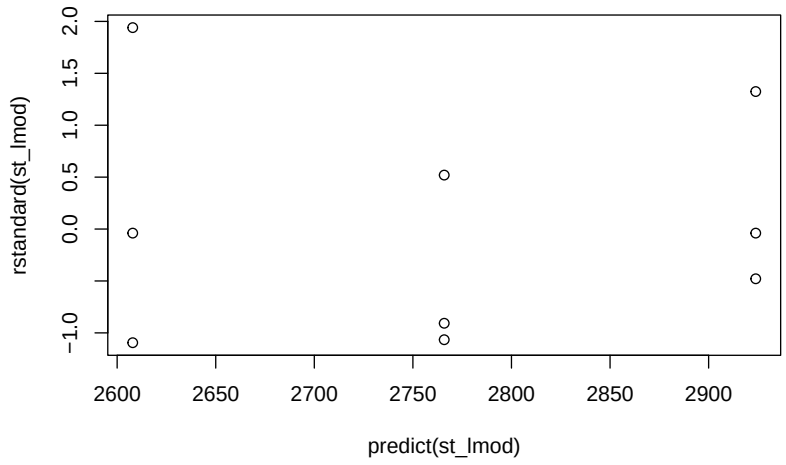

```
#plot the residuals vs. x
plot(ratio, resid(st_lmod))
```



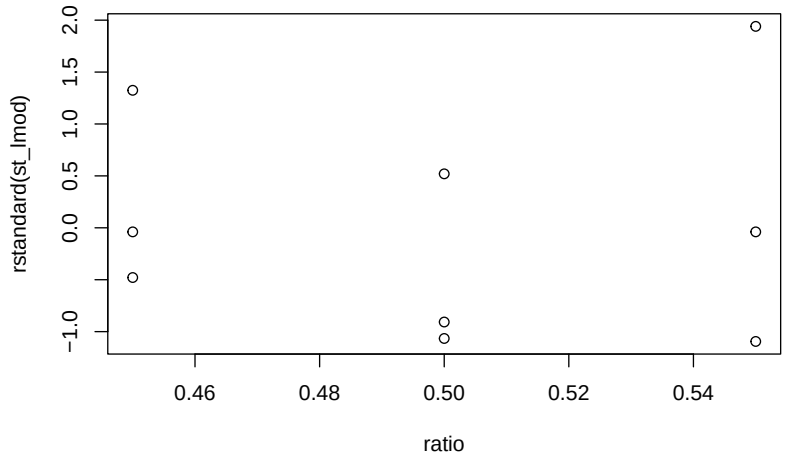

```
#standardized residuals
rstandard(st_lmod)
```

```
##           1           2           3           4           5           6
##  1.32428967 -0.47889442 -0.03909342 -0.90738711  0.51976543 -1.06595962
##           7           8           9
##  1.94001107 -0.03909342 -1.09461582
```

```
#plot the standardized residuals vs. y_hat
plot(predict(st_lmod), rstandard(st_lmod))
```




```
#plot the standardized residuals vs. x
plot(ratio, rstandard(st_lmod))
```



(c)

```
#divide by 10 to have the correct units
confint(st_lmod, parm = "ratio", level = .9)/10
```

```
##             5 %      95 %
## ratio -357.3881 -274.6119
```

(d)

```
#check the model summary to do the hypothesis test
summary(st_lmod)
```

```
##
## Call:
## lm(formula = strength ~ ratio)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -26.889 -22.889  -0.889  13.111  44.111
##
## Coefficients:
##             Estimate Std. Error t value      Pr(>|t|)
## (Intercept)   4345.9      109.6   39.66 0.00000000169 ***
## ratio        -3160.0      218.5  -14.46 0.00000179899 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.76 on 7 degrees of freedom
## Multiple R-squared:  0.9676, Adjusted R-squared:  0.963
## F-statistic: 209.2 on 1 and 7 DF,  p-value: 0.000001799
```

```
#check the ANOVA table to also do the hypothesis test
anova(st_lmod)
```

```
## Analysis of Variance Table
##
## Response: strength
##           Df Sum Sq Mean Sq F value      Pr(>F)
## ratio      1 149784  149784  209.24 0.000001799 ***
## Residuals  7   5011     716
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

(e)

```
#fit and 95% confidence interval
#(no need to specify the confidence level since .95 is the default)
predict(st_lmod, newdata = list(ratio = .5), interval = "confidence")
```

```
##        fit    lwr      upr
## 1 2765.889 2744.8 2786.978
```

(f)

```
#fit and 95% prediction interval
predict(st_lmod, newdata = list(ratio = .5), interval = "prediction")
```

```
##        fit      lwr      upr
## 1 2765.889 2699.201 2832.577
```

Question 9.2.1 (p. 697)

```
size <- c(66, 68, 70, 72, 74, 76)
size2 <- size^2
time <- c(14.90, 14.67, 14.50, 14.53, 14.79, 15.02)

time_mod <- lm(time ~ size + size2)
```

(a)

```
#get s_SF
time_s_SF <- summary(time_mod)$sigma
time_s_SF
```

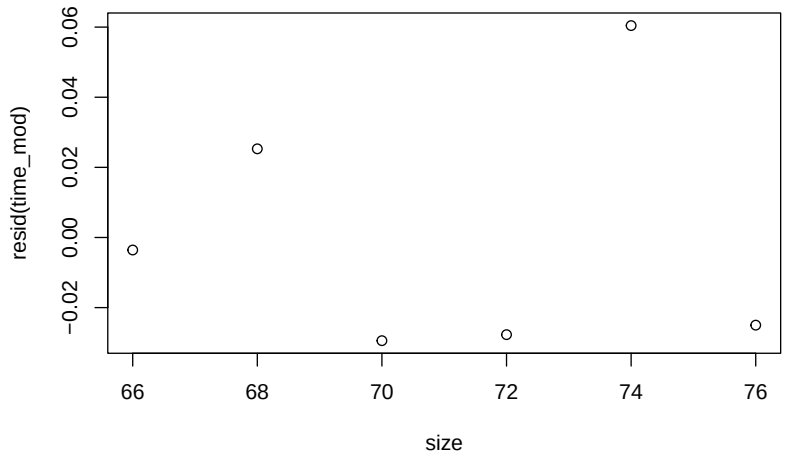
```
## [1] 0.04677199
```

(b)

```
resid(time_mod)
```

```
##           1           2           3           4           5           6
## -0.003571429  0.025285714 -0.029428571 -0.027714286  0.060428571 -0.025000000
```

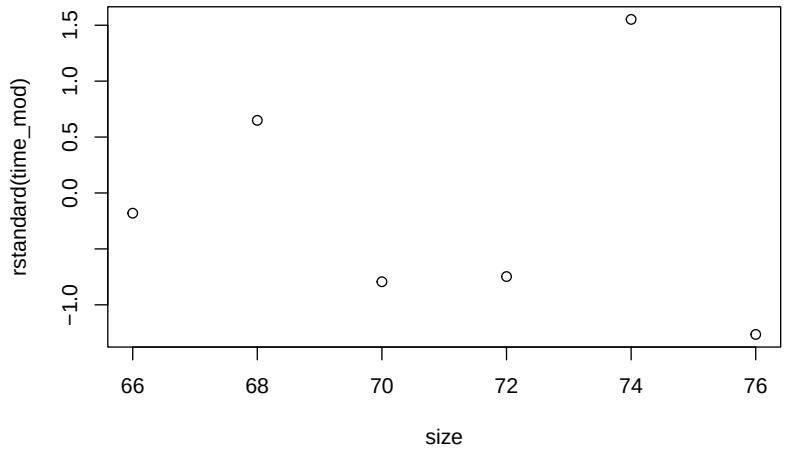
```
#plot the residuals vs. x
plot(size, resid(time_mod))
```




```
rstandard(time_mod)
```

```
##           1           2           3           4           5           6
## -0.1806967  0.6494826 -0.7936080 -0.7473784  1.5521534 -1.2648766
```

```
#plot the standardized residuals vs. x
plot(size, rstandard(time_mod))
```



(c)

```
#90% confidence intervals
confint(time_mod, level = .9)
```

```
##                      5 %        95 %
## (Intercept) 81.31967885 126.6591783
## size        -3.17390632  -1.8945937
## size2        0.01344276   0.0224501
```

(d)

```
#predicted value and 90% confidence interval for size = 70
predict(time_mod, newdata = list(size = 70, size2 = 70^2),
        interval = "confidence", level = .9)
```

```
##        fit      lwr      upr
## 1 14.52943 14.46235 14.59651
```

```
#predicted value and 90% confidence interval for size = 76
predict(time_mod, newdata = list(size = 76, size2 = 76^2),
        interval = "confidence", level = .9)
```

```
##      fit      lwr      upr
## 1 15.045 14.94524 15.14476
```

(f)

```
#lower  bound at  level = .8  gives  90%  lower  prediction
  #bound, lwr below, for size = 70
predict(time_mod, newdata = list(size = 70, size2 = 70^2),
        interval = "prediction", level = .8)
```

```
##        fit      lwr      upr
## 1 14.52943 14.43972 14.61913
```

```
#lower  90%  prediction bound, lwr, for size = 76
predict(time_mod, newdata = list(size = 76, size2 = 76^2),
        interval = "prediction", level = .8)
```

```
##      fit      lwr      upr
## 1 15.045 14.94162 15.14838
```

(h)

```
#use the ANOVA table for hypothesis testing
time_mod_anova <- anova(time_mod)
time_mod_anova
```

```
## Analysis of Variance Table
##
## Response: time
##           Df   Sum Sq  Mean Sq F value   Pr(>F)
## size       1 0.014001 0.014001  6.4003 0.085433 .
## size2      1 0.192386 0.192386 87.9430 0.002568 **
## Residuals  3 0.006563 0.002188
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#add the sums of squares and df for size and size2 to get SSR and MSR
time_mod_SSR <- sum(time_mod_anova$`Sum Sq`[1:2])
time_mod_SSR
```

```
## [1] 0.2063871
```

```
time_mod_df <- sum(time_mod_anova$Df[1:2])
time_mod_df
```

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

```
#divide SSR by degrees of freedom to calculate MSR
time_mod_MSR <- time_mod_SSR/time_mod_df
time_mod_MSR
```

```
## [1] 0.1031936
```

```
#find MSE and df and use them to calculate f
time_mod_MSE <- time_mod_anova$`Mean Sq`[3]
time_mod_MSE
```

```
## [1] 0.002187619
```

```
time_mod_df_mse <- time_mod_anova$Df[3]
time_mod_df_mse
```

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

```
f <- time_mod_MSR/time_mod_MSE
f
```

```
## [1] 47.17164
```

```
#p-value
1 - pf(f, time_mod_df, time_mod_df_mse)
```

```
## [1] 0.00541032
```

(i)

```
#use the model summary to test the hypothesis
summary(time_mod)
```

```
##
## Call:
## lm(formula = time ~ size + size2)
##
## Residuals:
##         1         2         3         4         5         6
## -0.003571  0.025286 -0.029429 -0.027714  0.060429 -0.025000
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 103.989429   9.632915  10.795  0.00170 **
## size         -2.534250   0.271805  -9.324  0.00261 **
## size2         0.017946   0.001914   9.378  0.00257 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04677 on 3 degrees of freedom
## Multiple R-squared:  0.9692, Adjusted R-squared:  0.9486
## F-statistic: 47.17 on 2 and 3 DF,  p-value: 0.00541
```

Question 9.2.2 (p. 697)

(a)

```
#get s_LF from the model from 4.2.2
area_s_LF <- summary(area_mod)$sigma
area_s_LF
```

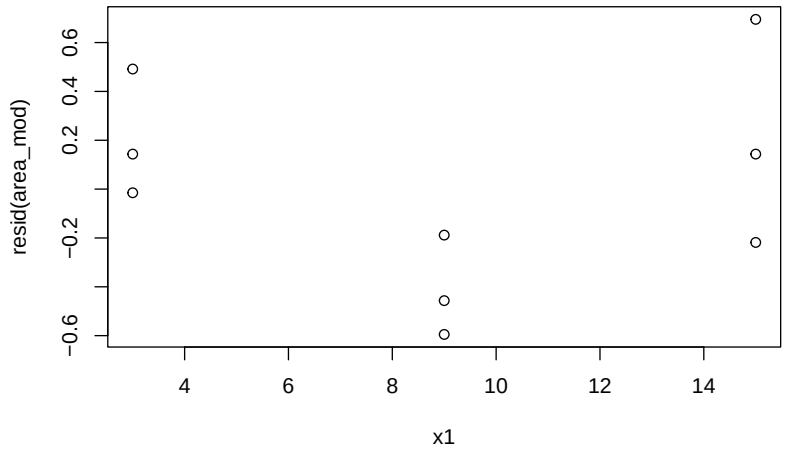
```
## [1] 0.4850745
```

(b)

```
resid(area_mod)
```

```
##          1          2          3          4          5          6          7
## -0.0150000  0.1433333  0.4916667 -0.5950000 -0.4566667 -0.1883333  0.6950000
##          8          9
##  0.1433333 -0.2183333
```

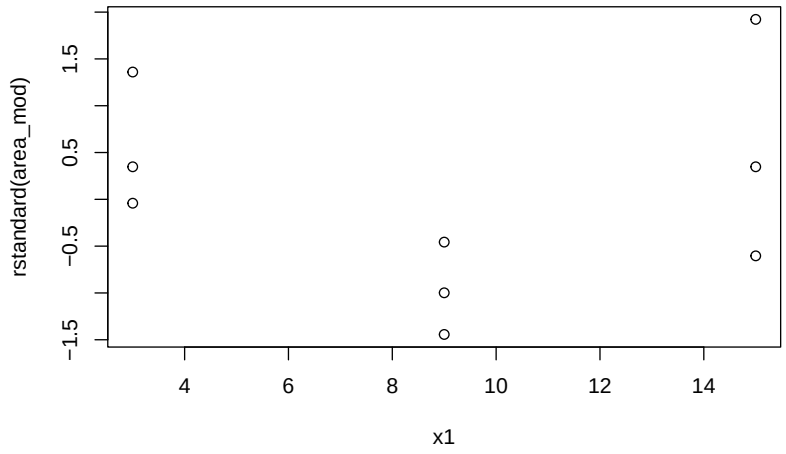
```
#plot the residuals vs. x1
plot(x1, resid(area_mod))
```



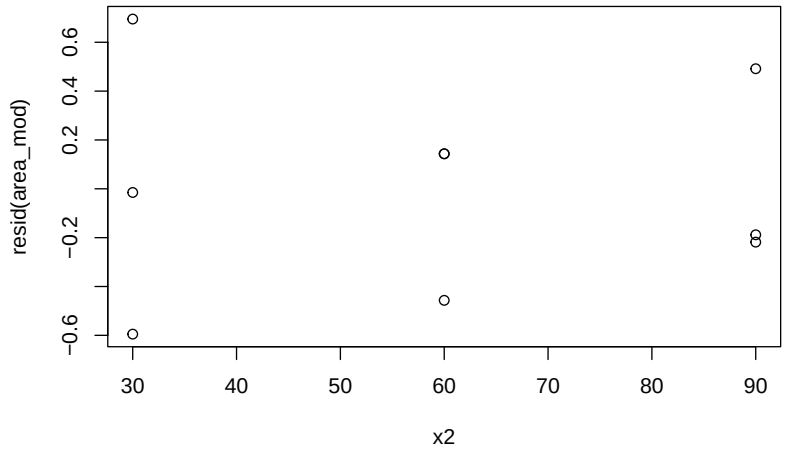

```
rstandard(area_mod)
```

```
##          1          2          3          4          5          6
## -0.04148768  0.34769895  1.35987384 -1.44335494 -0.99854392 -0.45686025
##          7          8          9
##  1.92226234  0.34769895 -0.60387618
```

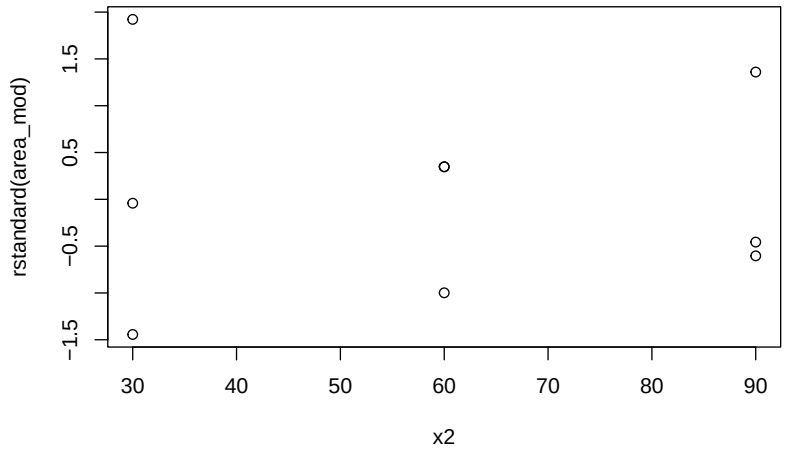
```
#plot the standardized residuals vs. x1
plot(x1, rstandard(area_mod))
```



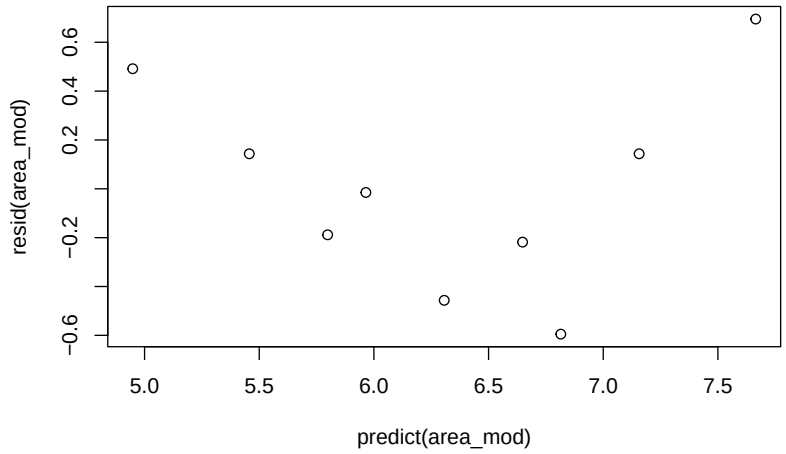

```r
#plot the residuals vs. x2
plot(x2, resid(area_mod))
```



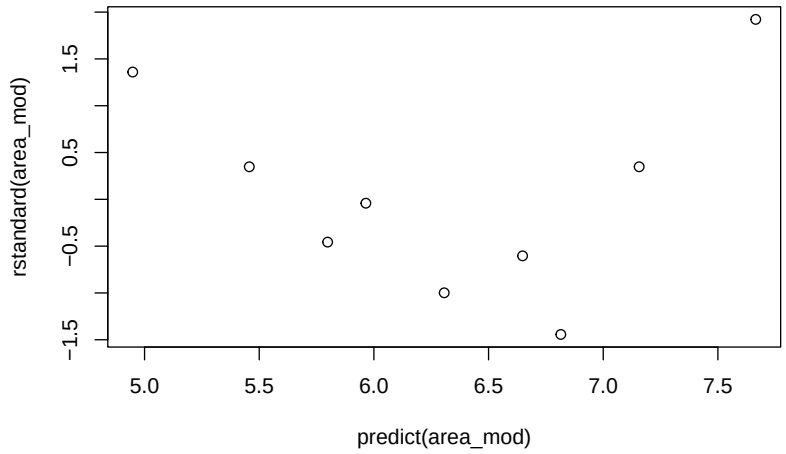

```r
#plot the standardized residuals vs. x2
plot(x2, rstandard(area_mod))
```




```
#plot the residuals vs. y_hat
plot(predict(area_mod), resid(area_mod))
```




```
#plot the standardized residuals vs. y_hat
plot(predict(area_mod), rstandard(area_mod))
```



(c)

```
#90% confidence intervals
confint(area_mod, level = .9)
```

```
##                      5 %         95 %
## (Intercept)  5.03630122  7.060365445
## x1           0.07753174  0.205801596
## x2          -0.02977143 -0.004117459
```

(d)

```
#fits and 90% confidence intervals
predict(area_mod, newdata = list(x1 = c(9, 10), x2 = c(60, 70)),
        interval = "confidence",
        level = .9)
```

```
##        fit      lwr      upr
## 1 6.306667 5.992471 6.620862
## 2 6.278889 5.933512 6.624266
```

(f)

```
#lower  bound at  level = .8  gives  the  90%  lower prediction bound, lwr below
predict(area_mod, newdata = list(x1 = c(9, 10), x2 = c(60, 70)),
        interval = "prediction",
        level = .8)
```

```
##        fit      lwr      upr
## 1 6.306667 5.570500 7.042833
## 2 6.278889 5.535094 7.022684
```

(h)

```
#use the ANOVA table for hypothesis testing
area_mod_anova <- anova(area_mod)
area_mod_anova
```

```
## Analysis of Variance Table
##
## Response: s_area
##           Df Sum Sq Mean Sq F value   Pr(>F)
## x1         1 4.3350  4.3350 18.4235 0.005136 **
## x2         1 1.5504  1.5504  6.5892 0.042510 *
## Residuals  6 1.4118  0.2353
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#add thesums of squares and df for size and size2 to get SSR and MSR
area_mod_SSR <- sum(area_mod_anova$`Sum Sq`[1:2])
area_mod_SSR
```

```
## [1] 5.885417
```

```
area_mod_df <- sum(area_mod_anova$Df[1:2])
area_mod_df
```

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

```
#divide SSR by degrees of freedom to calculate MSR
area_mod_MSR <- area_mod_SSR/area_mod_df
area_mod_MSR
```

```
## [1] 2.942708
```

```
#find MSE and df and use them to calculate f
area_mod_MSE <- area_mod_anova$`Mean Sq`[3]
area_mod_MSE
```

```
## [1] 0.2352972
```

```
area_mod_df_mse <- area_mod_anova$Df[3]
area_mod_df_mse
```

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

```
f <- area_mod_MSR/area_mod_MSE
f
```

```
## [1] 12.50635
```

```
#p-value
1 - pf(f, area_mod_df, area_mod_df_mse)
```

```
## [1] 0.007241615
```