Lesson 5 R Activity

Rick Davila

5/2/2020

Lesson 4 - Install packages

```
knitr::opts_chunk$set(echo = TRUE)

library(e1071)
library(xtable)
library("xlsx") # Needed to read data
```

```
## Warning: package 'xlsx' was built under R version 4.0.3
```

```
library(car) # Needed for alternative scatterplot matrix to default
```

```
## Loading required package: carData
```

```
library(scatterplot3d) # Needed for 3D scatterplot
```

```
## Warning: package 'scatterplot3d' was built under R version 4.0.3
```

```
library(matlib) # Needed for Invers() function
```

```
## Warning: package 'matlib' was built under R version 4.0.4
```

```
library(MASS) # Needed for ginv() function
library(standardize) # Needed for unit normal scaling in Example 3.14
```

```
## Warning: package 'standardize' was built under R version 4.0.4
```

```
## Registered S3 methods overwritten by 'lme4':
##   method                          from
##   cooks.distance.influence.merMod car
##   influence.merMod                car
##   dfbeta.influence.merMod         car
##   dfbetas.influence.merMod        car
```

```
##
## **********************************************************
##          Loading standardize package version 0.2.2
##     Call standardize.news() to see new features/changes
## **********************************************************
```

```
rm(list = ls())
```

Lesson 4 - Read data file (data-ex-3-1.xlsx)

```
ex3_1 <- read.xlsx("data-ex-3-1.xlsx",
                   sheetIndex = 1,
                   colIndex = c(2,3,4),
                   as.data.frame = TRUE,
                   header = TRUE)
```

Lesson 4 - Assign labels to data columns using names() and attach() commands

```
names(ex3_1) <- c("Delivery_Time", "Num_Cases", "Distance")
attach(ex3_1)
```

Lesson 4 - Output data to make sure it reads properly

```
# Output data to make sure it reads properly
xtable(ex3_1)
```

Delivery_Time <dbl>	Num_Cases <dbl>	Distance <dbl>
16.68	7	560
11.50	3	220
12.03	3	340
14.88	4	80
13.75	6	150
18.11	7	330
8.00	2	110
17.83	7	210

	Delivery_Time	Num_Cases	Distance
	<dbl>	<dbl>	<dbl>
	79.24	30	1460
	21.50	5	605

1-10 of 25 rows Previous **1** 2 3 Next

Lesson 4 - Output data structure and dimensions

```
# output dataframe structure
str(ex3_1)
```

```
## 'data.frame':	25 obs. of  3 variables:
##  $ Delivery_Time: num  16.7 11.5 12 14.9 13.8 ...
##  $ Num_Cases    : num  7 3 3 4 6 7 2 7 30 5 ...
##  $ Distance     : num  560 220 340 80 150 330 110 210 1460 605 ...
```

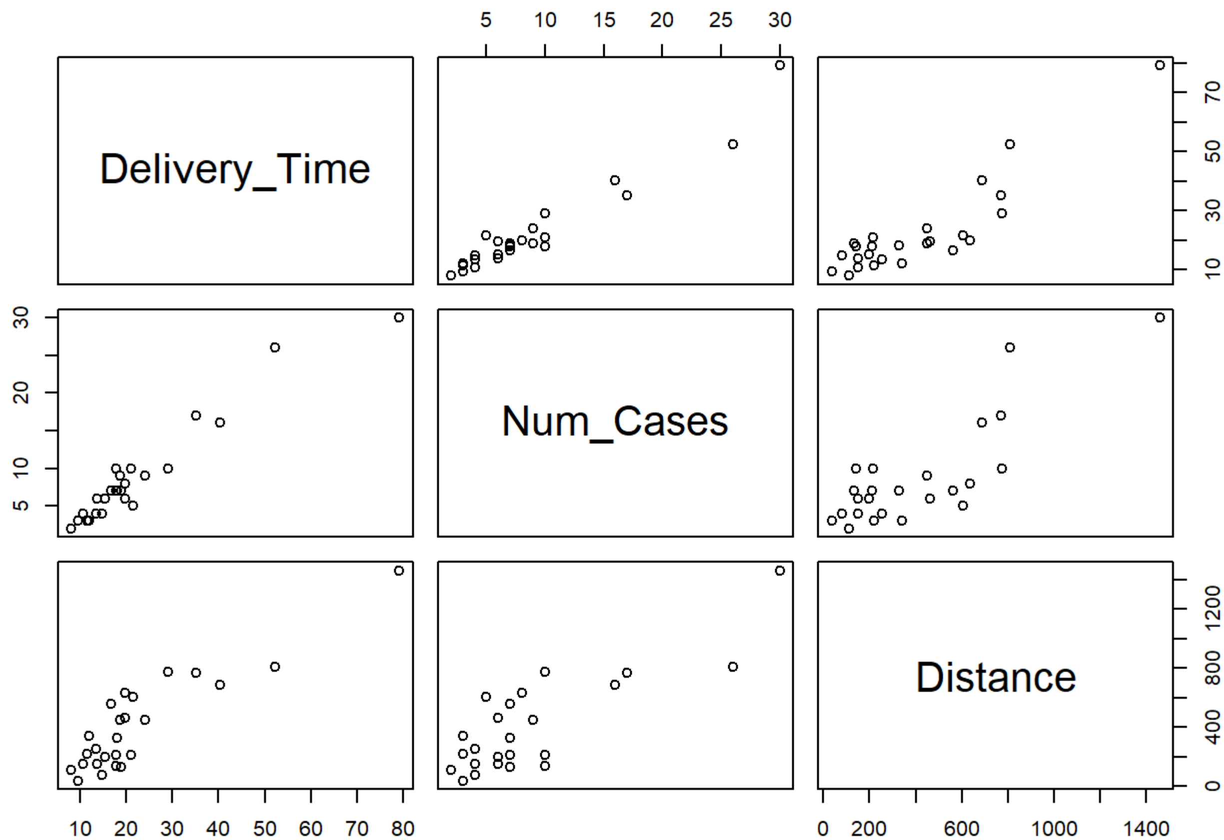
```
# dim of data 'matrix'
dim(ex3_1)
```

```
## [1] 25  3
```

Lesson 4 - Example 3.1 (p. 75-77)

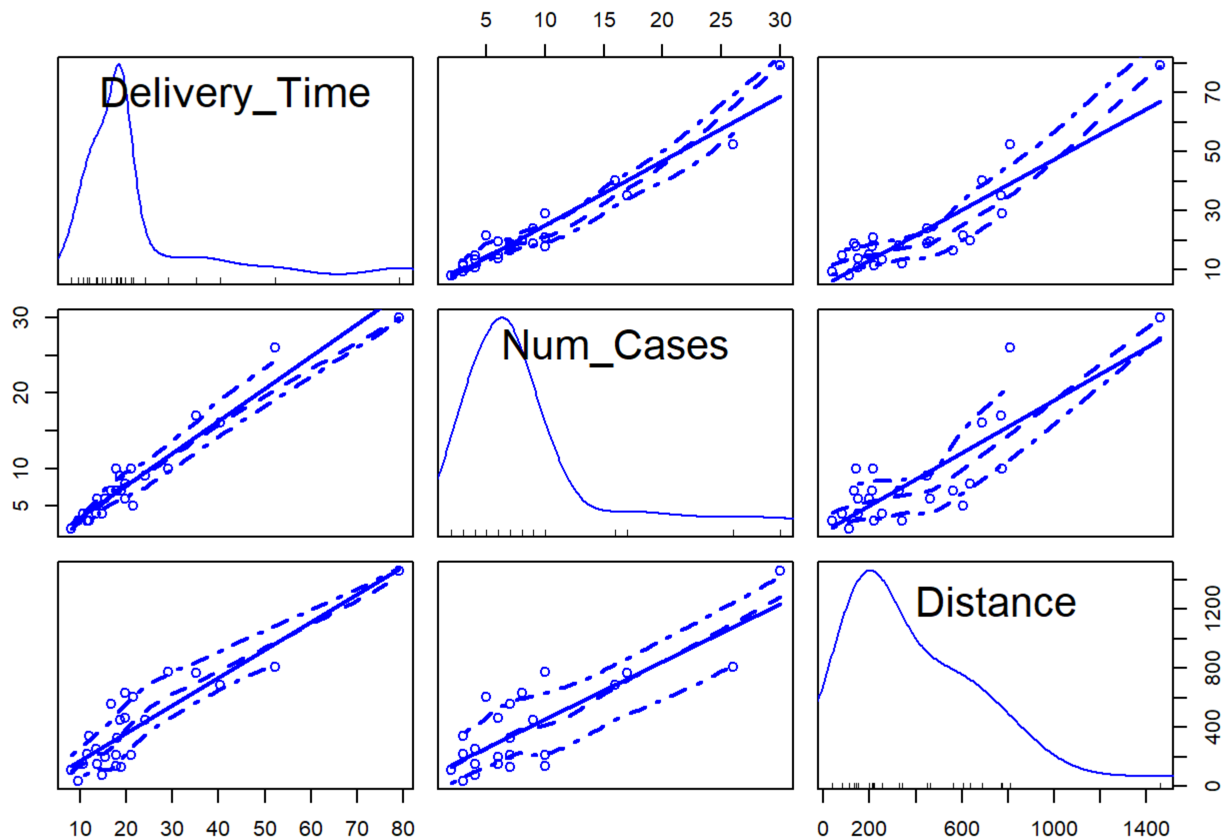
Ex 3.1. Create pairwise scatterplots using pairs() command

```
# pairs() - The pairs function returns a plot matrix, consisting of scatterplots for each variab
le-combination of a data frame.

pairs(ex3_1)
```



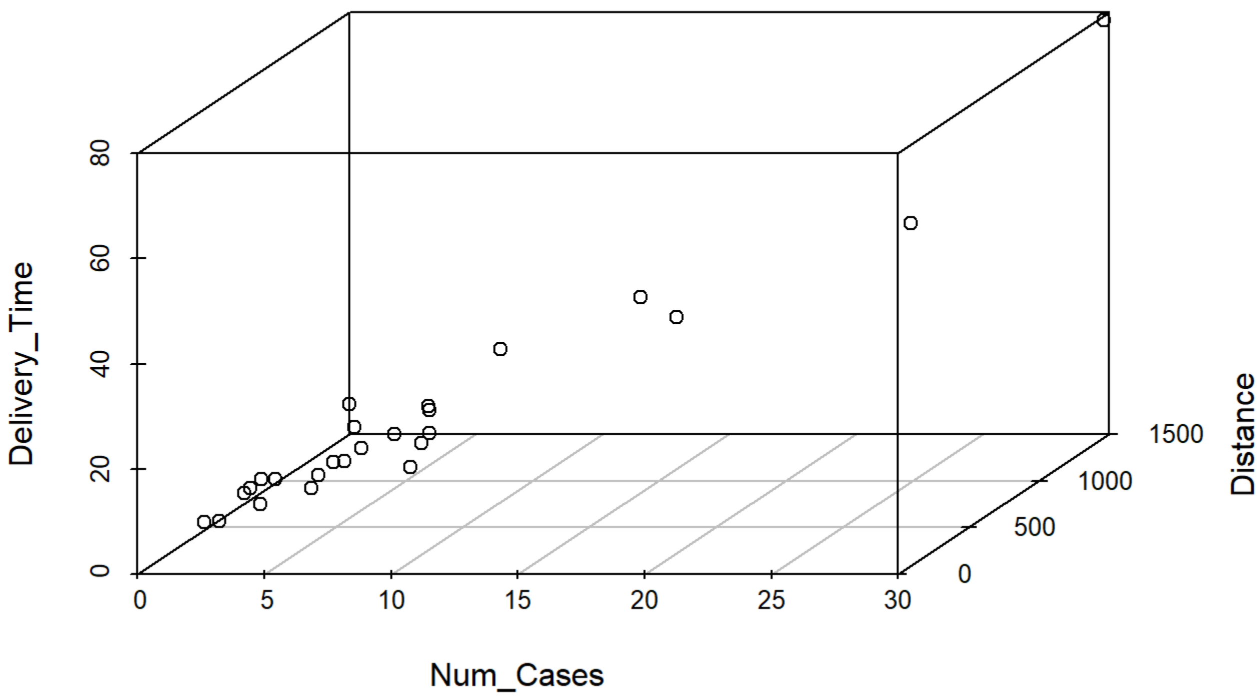
Ex 3.1. Create pairwise scatterplots using the scatterplotMatrix() command from the “car” package

```
# The scatterplotMatrix function provides a convenient interface to the pairs function to produc
e enhanced scatterplot matrices, including univariate displays on the diagonal and a variety of
 fitted lines, smoothers, variance functions, and concentration ellipsoids.

scatterplotMatrix(ex3_1, use = c("pairwise.complete.obs"))
```



Ex 3.1. Create 3D scatterplot using scatterplot3d() command from “scatterplot3d” package

```
# Plots a three dimensional (3D) point cloud
scatterplot3d(Num_Cases, Distance, Delivery_Time)
```



Ex 3.1. Obtain regression estimators using matrix algebra (“by hand”)

Define X matrix of regressor observations (don’t forget to include column for intercept)

```
X <- cbind(matrix(1,length(Distance),1),as.matrix(Num_Cases),as.matrix(Distance))
y <- as.matrix(Delivery_Time)
```

Ex 3.1. Display X to make sure it is correct; compare to p. 75)

```
# display X matrix
X
```

```
##       [,1] [,2] [,3]
##  [1,]    1    7  560
##  [2,]    1    3  220
##  [3,]    1    3  340
##  [4,]    1    4   80
##  [5,]    1    6  150
##  [6,]    1    7  330
##  [7,]    1    2  110
##  [8,]    1    7  210
##  [9,]    1   30 1460
## [10,]    1    5  605
## [11,]    1   16  688
## [12,]    1   10  215
## [13,]    1    4  255
## [14,]    1    6  462
## [15,]    1    9  448
## [16,]    1   10  776
## [17,]    1    6  200
## [18,]    1    7  132
## [19,]    1    3   36
## [20,]    1   17  770
## [21,]    1   10  140
## [22,]    1   26  810
## [23,]    1    9  450
## [24,]    1    8  635
## [25,]    1    4  150
```

Ex 3.1. Define the matrix product of X_Transpose and X and display output to make sure it is correct (compare to p. 76)

```
# X'X matrix
xTx <- t(X) %*% X
xTx
```

```
##       [,1]   [,2]    [,3]
## [1,]    25    219   10232
## [2,]   219   3055  133899
## [3,] 10232 133899 6725688
```

Ex 3.1. Take the inverse of xTx (“x-Transpose time x”) using three different approaches: Inverse(), ginv(), and inv() and display output to make sure it is correct (compare to p. 77)

```
print("Inverse(xTx)")
```

```
## [1] "Inverse(xTx)"
```

```
Inverse(xTx)
```

```
##             [,1]        [,2]       [,3]
## [1,]  0.11321519 -0.00444859 -8.367e-05
## [2,] -0.00444859  0.00274378 -4.786e-05
## [3,] -0.00008367 -0.00004786  1.230e-06
```

```
print("ginv(xTx)")
```

```
## [1] "ginv(xTx)"
```

```
ginv(xTx)
```

```
##               [,1]          [,2]          [,3]
## [1,]  1.132152e-01 -4.448593e-03 -8.367257e-05
## [2,] -4.448593e-03  2.743783e-03 -4.785709e-05
## [3,] -8.367257e-05 -4.785709e-05  1.228745e-06
```

```
print("inv(xTx)")
```

```
## [1] "inv(xTx)"
```

```
inv(xTx)
```

```
##             [,1]        [,2]       [,3]
## [1,]  0.11321519 -0.00444859 -8.367e-05
## [2,] -0.00444859  0.00274378 -4.786e-05
## [3,] -0.00008367 -0.00004786  1.230e-06
```

Ex 3.1. Illustration - the above matrix inverse calculations are possibly bad. How to tell? Take the inverse of the inverse using each approach and see…

```
print("Inverse(Inverse(xTx))")
```

```
## [1] "Inverse(Inverse(xTx))"
```

```
Inverse(Inverse(xTx))
```

```
##             [,1]        [,2]       [,3]
## [1,]    24.88136    217.4561   10153.88
## [2,]   217.45605   3034.9085  132882.33
## [3,] 10153.87809 132882.3328 6674246.69
```

```
print("ginv(ginv(xTx))")
```

```
## [1] "ginv(ginv(xTx))"
```

```
ginv(ginv(xTx))
```

```
##       [,1]   [,2]    [,3]
## [1,]    25    219   10232
## [2,]   219   3055  133899
## [3,] 10232 133899 6725688
```

```
print("inv(inv(xTx))")
```

```
## [1] "inv(inv(xTx))"
```

```
inv(inv(xTx))
```

```
##             [,1]        [,2]       [,3]
## [1,]    24.88136    217.4561   10153.88
## [2,]   217.45605   3034.9085  132882.33
## [3,] 10153.87809 132882.3328 6674246.69
```

ginv(ginv(xTx)) returns the original matrix

Ex 3.1. Continuing on… use ginv() for matrix inverse calculations from here on out. Define / calculate X-transpose * y, where y is the vector of dependent variable observations. Compare to p. 77

```
print("X-transpose * y")
```

```
## [1] "X-transpose * y"
```

```
t(X) %*% y
```

```
##           [,1]
## [1,]    559.60
## [2,]   7375.44
## [3,] 337071.69
```

Ex 3.1. Calculate Beta coefficients. Compare to values on p. 77.

```
# The least-squares estimator of beta_coeffs

print("Beta Coefficients")
```

```
## [1] "Beta Coefficients"
```

```
beta_hat <- ginv(xTx) %*% t(X) %*% y
beta_hat
```

```
##            [,1]
## [1,] 2.34123115
## [2,] 1.61590721
## [3,] 0.01438483
```

So we have ˆy = 2.3412311 + 1.6159072x1 + 0.0143848x2

The equation is

$$\hat{y} = (2.3412311) + (1.6159072)x_1 + (0.0143848)x_2$$

Ex 3.1. Obtain regression estimates using lm command

```
model <- lm(Delivery_Time ~ Num_Cases + Distance)
xtable(model)
```

	Estimate <dbl>	Std. Error <dbl>	t value <dbl>	Pr(>|t|) <dbl>
(Intercept)	2.34123115	1.096730168	2.134738	4.417012e-02
Num_Cases	1.61590721	0.170734918	9.464421	3.254932e-09
Distance	0.01438483	0.003613086	3.981313	6.312469e-04

3 rows

```
# aov table
xtable(summary(aov(model)))
```

	Df <dbl>	Sum Sq <dbl>	Mean Sq <dbl>	F value <dbl>	Pr(>F) <dbl>
Num_Cases	1	5382.4088	5382.40880	506.61936	1.112549e-16
Distance	1	168.4021	168.40213	15.85085	6.312469e-04
Residuals	22	233.7317	10.62417	NA	NA

3 rows

```
# R coefficient
res <- data.frame(model$residuals)
colnames(res) <- "Residuals"
out <- as.data.frame(c(sigma(model), summary(model)$r.squared, summary(model)$adj.r.squared))
names(out) <- ""
rownames(out) <- c("$$S$$", "$$R^2$$", "$$R^{2}_{adj}$$")
xtable(out, digits=6)
```

	<dbl>
S	3.2594734
R^2	0.9595937
R^2_{adj}	0.9559205

3 rows

```
#xtable(res)
```

Lesson 4 - Example 3.2 (p. 81)

Calculate SS_Residual (by hand) using the definition of a residual on p.80

$$SS_{res} = \sum_{i=1}^{n} (y_i - \hat{y_i})^2$$

```
y_hat <- beta_hat[1,1] + beta_hat[2,1]*Num_Cases + beta_hat[3,1]*Distance
SS_res1 <- sum((Delivery_Time - y_hat)^2)
SS_res1
```

```
## [1] 233.7317
```

$$SS_{res} = \sum_{i=1}^{n} (y_i - \hat{y_i})^2 = 233.7316774$$

Ex 3.2. Calculate SS_Residual using matrix algebra formula on p. 81.

```
yprime_y <- t(Delivery_Time) %*% Delivery_Time
beta_x_y <- t(beta_hat)%*%t(X)%*% Delivery_Time
SS_res2 <- yprime_y - beta_x_y
SS_res2
```

```
##          [,1]
## [1,] 233.7317
```

The residual sum of squares using using the matrix algebra formula is:

$$SS_{res} = \mathbf{y}'\mathbf{y} - \hat{\beta}'\mathbf{X}'\mathbf{y} = 233.7316774$$

Ex 3.2. Learning point: Different packages use different 'behind the curtain' approaches to calculate the matrix inverse. For some reason, inv() and Inverse() use rounding that results in over half of the SS_Residual being lost. Using ginv(), from the MASS package, gives results that match the textbook and the results obtained from the lm() command

```
# The least-squares estimator of beta_coeffs using inv() and Inverse()

print("Beta Coefficients - using inv()")
```

```
## [1] "Beta Coefficients - using inv()"
```

```
beta_hat_inv <- inv(xTx) %*% t(X) %*% y

# yprime_y calculated in Ex 3.1
beta_x_y_inv <- t(beta_hat_inv)%*%t(X)%*% Delivery_Time
SS_res_inv <- yprime_y - beta_x_y_inv

print("Beta Coefficients - using Inverse()")
```

```
## [1] "Beta Coefficients - using Inverse()"
```

```
beta_hat_Inverse = Inverse(xTx) %*% t(X) %*% y

beta_x_y_Inverse <- t(beta_hat_Inverse)%*%t(X)%*% Delivery_Time
SS_res_Inverse <- yprime_y - beta_x_y_Inverse
```

The residual sum of squares using using the matrix algebra formula and inv() function is:

$$SS_{res} = \mathbf{y}'\mathbf{y} - \hat{\beta}'\mathbf{X}'\mathbf{y} = 104.7801627$$

The residual sum of squares using using the matrix algebra formula and Inverse() function is:

$$SS_{res} = \mathbf{y}'\mathbf{y} - \hat{\beta}'\mathbf{X}'\mathbf{y} = 104.7801627$$

Ex 3.2. Note also the discrepancy between using the definition of a residual and using the matrix algebra in Equation 3.16

```
# Using the inv() function
y_hat <- beta_hat_inv[1,1] + beta_hat_inv[2,1]*Num_Cases + beta_hat_inv[3,1]*Distance
SS_res_inv2 <- sum((Delivery_Time - y_hat)^2)

# Using the inv() function
y_hat <- beta_hat_Inverse[1,1] + beta_hat_Inverse[2,1]*Num_Cases + beta_hat_Inverse[3,1]*Distanc
e
SS_res_Inverse2 <- sum((Delivery_Time - y_hat)^2)
```

The residual sum of squares resulting from using inv()

$$SS_{res} = \sum_{i=1}^{n} (y_i - \hat{y_i})^2 = 234.725971$$

The residual sum of squares resulting from using Inverse()

$$SS_{res} = \sum_{i=1}^{n} (y_i - \hat{y_i})^2 = 234.725971$$

Ex 3.2. Calculate SST and SS_Regression (by hand). Compare to ANOVA table on p. 78. Values match.

```
SS_T <- t(Delivery_Time)%*%Delivery_Time-(sum(Delivery_Time))^2/length(Delivery_Time)
SS_R <- t(beta_hat)%*%t(X)%*%Delivery_Time-(sum(Delivery_Time))^2/length(Delivery_Time)
```

$$SS_T = \mathbf{y}'\mathbf{y} - \frac{\left(\sum_{i=1}^{n} y_i\right)^2}{n} = 5784.5426$$

$$SS_R = \hat{\beta}'\mathbf{X}'\mathbf{y} - \frac{\left(\sum_{i=1}^{n} y_i\right)^2}{n} = 5550.8109226$$

The SS_T and SS_R calculated values are consistent with the values reported in the book's ANOVA table (5784.5 for SS_T and 5550.8 for SS_R).

Ex 3.2. Calculate ANOVA table using anova() and display the output

```
# anova table, model calculated using lm() in Ex 3.1 abov
anova(model)
```

	Df <int>	Sum Sq <dbl>	Mean Sq <dbl>	F value <dbl>	Pr(>F) <dbl>
Num_Cases	1	5382.4088	5382.40880	506.61936	1.112549e-16
Distance	1	168.4021	168.40213	15.85085	6.312469e-04
Residuals	22	233.7317	10.62417	NA	NA

3 rows

```
summary(anova(model))
```

```
##        Df          Sum Sq          Mean Sq          F value
##  Min.   : 1.0   Min.   : 168.4   Min.   :  10.62   Min.   : 15.85
##  1st Qu.: 1.0   1st Qu.: 201.1   1st Qu.:  89.51   1st Qu.:138.54
##  Median : 1.0   Median : 233.7   Median : 168.40   Median :261.24
##  Mean   : 8.0   Mean   :1928.2   Mean   :1853.81   Mean   :261.24
##  3rd Qu.:11.5   3rd Qu.:2808.1   3rd Qu.:2775.41   3rd Qu.:383.93
##  Max.   :22.0   Max.   :5382.4   Max.   :5382.41   Max.   :506.62
##                                                     NA's   :1
##      Pr(>F)
##  Min.   :0.0000000
##  1st Qu.:0.0001578
##  Median :0.0003156
##  Mean   :0.0003156
##  3rd Qu.:0.0004734
##  Max.   :0.0006312
##  NA's   :1
```

Lesson 4 - Example 3.3 (p. 87)

Test for significance of regression using F-test (by hand)

Calculate degrees of freedom. Output to make sure it is correct; compare to the ANOVA table on p. 87. The test for significance is a test to determine if there is a linear relationship between the response and any of the regressor variables.

```
# degrees of freedom
k <- 2                     # number of beta regressor parameters
n <- length(Delivery_Time) # number of obervations

SS_reg_df <- k
SS_res_df <- n-k-1
SS_T_df <- n - 1
```

$$SS_R df = 2$$

$$SS_{res} df = 22$$

$$SS_T df = 24$$

Ex 3.3. Calculate Mean-Square for Regression and Residual

```
# n = length(Delivery_Time); calculated in previous chunk

SS_T <- t(Delivery_Time)%*%Delivery_Time - (sum(Delivery_Time))^2/n
SS_R <- t(beta_hat) %*% t(X) %*% Delivery_Time - (sum(Delivery_Time))^2/n
SS_res <- SS_T - SS_R
```

$$SS_R = 5550.8109226$$

$$SS_{res} = 233.7316774$$

Ex 3.3. Calculate F-test algebraically and F-critical using qf() command

```
# recall, k = 2 (above)

MS_R <- SS_R/k
MS_res <- SS_res/(n-k-1)
F_0 <- MS_R/MS_res

# F critical using the qf() command at the 0.01 significance
siglevel <- 0.01

Fcritical <- qf(1-siglevel, SS_reg_df, SS_res_df)
```

The F_0 statistic is

$$F_0 = \frac{MS_R}{MS_{res}} = \frac{2775.4054613}{10.6241672} = 261.2351087$$

The F critical value is

$$F_0 = 5.7190219$$

Lesson 4 - Example 3.4 (p. 88-89)

Test the significance of the individual regression coefficients (by hand)

Define C matrix for use in computing se(B_j)

```
# xTx matrix calculated in the begining of this lesson R activity
C_matrix <- ginv(xTx)
C_matrix
```

```
##               [,1]          [,2]          [,3]
## [1,]  1.132152e-01 -4.448593e-03 -8.367257e-05
## [2,] -4.448593e-03  2.743783e-03 -4.785709e-05
## [3,] -8.367257e-05 -4.785709e-05  1.228745e-06
```

Ex 3.4. (p. 88-89) Compute the t-test algebraically and t-critical using qt() command. Calculate p-value using pt() command.

```
# n is the number of observations and p is the number of beta parameters
p <- length(beta_hat)
sigma_hat_sq = SS_res/(n - p)

# calculate t statistic for regressor parameters beta_1 and beta_2
t0_beta2 = beta_hat[2,1]/sqrt(sigma_hat_sq * C_matrix[2,2])
t0_beta3 = beta_hat[3,1]/sqrt(sigma_hat_sq * C_matrix[3,3])

# t-critical and p-value calculations
# for alpha = 0.05
alpha = 0.05

tcritical_value = abs(qt(alpha/2, df = SS_res_df))
p_value_beta2 = 2*pt(-abs(t0_beta2), df = SS_res_df)
p_value_beta3 = 2*pt(-abs(t0_beta3), df = SS_res_df)
```

The test statistics for β_1 and β_2 are 9.4644214 and 3.9813131, respectively. The t critical value is

$$t_{0.025,22} = 2.0738731$$

so we conclude that each regressor individually, number of cases and distance, contribute significantly to the model. The P values are

$$P(\beta 1) = 3.2549316 \times 10^{-9}$$

and

$$P(\beta 2) = 6.3124686 \times 10^{-4}$$

Lesson 4 - Example 3.5 (p. 92-93)

Perform partial F-test on the significance of the contribution of Distance to the full model

Ex 3.5. Create full model using lm() command

```
y_bar = sum(Delivery_Time)/length(Delivery_Time)
model_full <- lm(Delivery_Time ~ Num_Cases + Distance)

beta_0 <- model_full$coefficients[1]
beta_1 <- model_full$coefficients[2]
beta_2 <- model_full$coefficients[3]

y_hat <- beta_0 + beta_1*Num_Cases + beta_2*Distance
SS_R_mf <- sum((y_hat - y_bar)^2)
```

Ex 3.5. Create reduced model using lm() command

```
# excludes the Distance parameter

model_reduced <- lm(Delivery_Time ~ Num_Cases)
beta_0 <- model_reduced$coefficients[1]
beta_1 <- model_reduced$coefficients[2]

y_hat <- beta_0 + beta_1*Num_Cases
SS_R_mr <- sum((y_hat - y_bar)^2)
```

Ex 3.5. Create F-test algebraically and F-critical using qf() command

```
r <- length(model_full$coefficients) - length(model_reduced$coefficients)
p <- length(model_full$coefficients)
n <- length(Delivery_Time)

SS_R_beta2contrib = SS_R_mf - SS_R_mr

F_0 <- (SS_R_beta2contrib/r)/MS_res

alpha = 0.05
F_critical <- qf(1-alpha,r,n-p)
```

$$F_0 = \frac{\beta_2|\beta_1, \beta_0/1}{MS_{Res}} = \frac{168.4021256}{10.6241672} = 15.8508543$$

Since $F_{0.05,1,22} = 4.3009495$, we conclude that Distance (x2) contributes significantly to the model.

Ex 3.5. Faster way with R - Use anova() command

```
# use anova() function on the ful model
F_0anova <- anova(model_full)$F[2]

#1Jun2020 -- added more info
model.full <- lm(Delivery_Time ~ Num_Cases + Distance)
model.reduced <- lm(Delivery_Time ~ Num_Cases)

summary(model.full)
```

```
##
## Call:
## lm(formula = Delivery_Time ~ Num_Cases + Distance)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7880 -0.6629  0.4364  1.1566  7.4197
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.341231   1.096730   2.135 0.044170 *
## Num_Cases   1.615907   0.170735   9.464 3.25e-09 ***
## Distance    0.014385   0.003613   3.981 0.000631 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.259 on 22 degrees of freedom
## Multiple R-squared:  0.9596, Adjusted R-squared:  0.9559
## F-statistic: 261.2 on 2 and 22 DF,  p-value: 4.687e-16
```

```
anova(model.full)
```

	Df <int>	Sum Sq <dbl>	Mean Sq <dbl>	F value <dbl>	Pr(>F) <dbl>
Num_Cases	1	5382.4088	5382.40880	506.61936	1.112549e-16
Distance	1	168.4021	168.40213	15.85085	6.312469e-04
Residuals	22	233.7317	10.62417	NA	NA

3 rows

```
results <- anova(model.reduced, model.full)
results
```

	Res.Df <dbl>	RSS <dbl>	Df <dbl>	Sum of Sq <dbl>	F <dbl>	Pr(>F) <dbl>
1	23	402.1338	NA	NA	NA	NA
2	22	233.7317	1	168.4021	15.85085	0.0006312469

2 rows

The F_0 test statistic from running anova on the full model is

$$F_0 > 15.8508543$$

Lesson 5 - Example 3.8 (p. 98)

Calculate 95% confidence intervals for the regression parameters (by hand)

A $100(1-\alpha)$ percent confidence interval (CI) for the regression coefficient $\beta_j, j = 0, 1, \ldots, k$, as

$$\hat{\beta}_1 - t_{\alpha/2,n-p}\sqrt{\hat{\sigma}^2 C_{jj}} \leq \hat{\beta}_j \leq \hat{\beta}_1 + t_{\alpha/2,n-p}\sqrt{\hat{\sigma}^2 C_{jj}}$$

```
# get the regression coefficients using the lm command
# re-running this to ensure we get the correct regression parameters
model <- lm(Delivery_Time ~ Num_Cases + Distance)

beta_0 <- model$coefficients[1]
beta_1 <- model$coefficients[2]
beta_2 <- model$coefficients[3]

leftwingb1CI <- beta_1-tcritical_value*sqrt(sigma_hat_sq*C_matrix[2,2])
rightwingb1CI <- beta_1+tcritical_value*sqrt(sigma_hat_sq*C_matrix[2,2])

leftwingb2CI <- beta_2-tcritical_value*sqrt(sigma_hat_sq*C_matrix[3,3])
rightwingb2CI <- beta_2+tcritical_value*sqrt(sigma_hat_sq*C_matrix[3,3])
```

$\underline{\textbf{Ans:}}$ For $\hat{\beta}_1 = 1.6159072$, $C_{1,1} = 0.0027438$, $\hat{\sigma^2} = 10.6241672$ and $t_{0.025,22} = 2.0738731$, the 95% CI is:

$$1.2618247 \leq \beta_1 = 1.6159072 \leq 1.9699898$$

for $\hat{\beta}_2 = 0.0143848$, $C_{2,2} = 1.2287447 \times 10^{-6}$, $\hat{\sigma^2} = 10.6241672$ and $t_{0.025,22} = 2.0738731$, the 95% CI is:

$$0.0068917 \leq \beta_2 = 0.0143848 \leq 0.0218779$$

Lesson 5 - Example 3.9 (p.99)

Calculate confidence interval for mean response about some point x_0

```
# enter the x_0 data
data.x_0 <- c(1,8, 275)
x_0 <- matrix(data.x_0, nrow = 3, ncol = 1, byrow = TRUE)

# The fitted value at this point x_0 is found from Eq. (3.47)
data.beta <- c(beta_0, beta_1, beta_2)
beta_matrix <- matrix(data.beta, nrow = 3, ncol = 1, byrow = TRUE)
y_hat <- t(x_0) %*% beta_matrix

# The variance of ŷ_0 is estimated by
var_y_hat <- sigma_hat_sq*t(x_0) %*%  ginv(t(X) %*% X) %*% x_0
left_wingCI = y_hat - tcritical_value*sqrt(var_y_hat)
right_wingCI = y_hat + tcritical_value*sqrt(var_y_hat)
```

$\underline{\textbf{Ans:}}$ A 95% CI on the mean response at this point $x_0 = 1, 8, 275$ is found using eqn. (3.49) in the e-book. A $100(1-\alpha)$ percent confidence interval on the mean response at the point $x_{01}, x_{02}, \ldots, x_{0k}$ is

$$\hat{y_0} - t_{\alpha/2,n-p}\sqrt{\hat{\sigma}^2 \mathbf{x_0'(X'X)^{-1}x_0}} \leq E(y|x_0) \leq \hat{y_0} + t_{\alpha/2,n-p}\sqrt{\hat{\sigma}^2 \mathbf{x_0'(X'X)^{-1}x_0}}$$

for this case,

$$19.2243161 - 2.0738731\sqrt{0.5734134} \leq E(y|x_0) \leq 19.2243161 + 2.0738731\sqrt{0.5734134}$$

finally,

$$17.653895 \leq E(y|x_0) \leq 20.7947371$$

Lesson 5 - Example 3.12 (p. 104)

Create confidence interval on a new observation. Use the same x_0

A point estimate of the future observation y_0 at the point $x_{01}, x_{02}, \ldots, x_0, x_{0k}$ is

$$\hat{y_0} = \mathbf{x_0'}\hat{\beta}$$

A $100(1-\alpha)$ percent prediction interval for the future observation is

$$\hat{y_0} - t_{\alpha/2,n-p}\sqrt{\hat{\sigma}^2(\mathbf{1 + x_0'(X'X)^{-1}x_0})} \leq y_0 \leq \hat{y_0} + t_{\alpha/2,n-p}\sqrt{\hat{\sigma}^2(\mathbf{1 + x_0'(X'X)^{-1}x_0})}$$

substituing the values in the equation,

```
sqrt_value <- sigma_hat_sq*(1+ t(x_0) %*%  ginv(t(X) %*% X) %*% x_0)
left_wingPI = y_hat - tcritical_value*sqrt(sqrt_value)
right_wingPI = y_hat + tcritical_value*sqrt(sqrt_value)
```

$\underline{\mathbf{Ans:}}$

$$19.2243161 - 2.0738731\sqrt{11.1975806} \leq y_0 \leq 19.2243161 + 2.0738731\sqrt{11.1975806}$$

finally,

$$12.284559 \leq y_0 \leq 26.1640731$$

Lesson 5 - Example 3.14 (p.115-116)

Create a model with standardized regression coefficients. Scale using the scale() function

```
# follows the unit normal scalling methodology outlined in section 3.9

# create 2 dim array containing only regressors
X <- cbind(as.matrix(Num_Cases),as.matrix(Distance))

# scale regressors using scale function -- essentially eqn 3.55
Z <- scale(X)
y <- as.matrix(Delivery_Time)

# scale responses using scale function -- essentially eqn 3.56
y_star <- scale(y)

# from eqn 3.57
b_hat = ginv(t(Z) %*% Z) %*% t(Z) %*% y_star
```

Ans: The fitted model is

$$\hat{y}^0 = 0.7162722z_1 + 0.3013078z_2$$