Homework 1 code

Luke Rodriguez

2026-01-15

R Markdown

Question 2.1(a) using lm function:

```
# 1. Create the dataset
# 'wear' is the response (Y), 'viscosity' (x1) and 'load' (x2) are predictors
wear_data <- data.frame(
  viscosity = c(1.6, 15.5, 22.0, 43.0, 33.0, 40.0),
  load = c(851, 816, 1058, 1201, 1357, 1115),
  wear = c(193, 230, 172, 91, 113, 125)
)

# 2. Fit the linear model
# The formula 'wear ~ viscosity + load' corresponds to E(Y) = b0 + b1x1 + b2x2 [cite: 12]
model <- lm(wear ~ viscosity + load, data = wear_data)

# 3. Output the results
# This produces the Estimates, Standard Errors, t-values, and p-values [cite: 118, 141]
summary(model)
```

```
##
## Call:
## lm(formula = wear ~ viscosity + load, data = wear_data)
##
## Residuals:
##       1       2       3       4       5       6
## -24.987  24.307  11.820 -20.460  12.830  -3.511
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 350.99427   74.75307   4.695   0.0183 *
## viscosity    -1.27199    1.16914  -1.088   0.3562
## load         -0.15390    0.08953  -1.719   0.1841
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25.5 on 3 degrees of freedom
## Multiple R-squared:  0.8618, Adjusted R-squared:  0.7696
## F-statistic: 9.353 on 2 and 3 DF,  p-value: 0.05138
```

```r
# 4. (Optional) Get the ANOVA table
# This corresponds to the ANOVA test for regression relationship [cite: 132, 133]
anova(model)
```

```
## Analysis of Variance Table
##
## Response: wear
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## viscosity  1 10240.4 10240.4 15.7510 0.02859 *
## load       1  1921.2  1921.2  2.9551 0.18410
## Residuals  3  1950.4   650.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Question 2.1(a) calculating through matrices:

```
# Matrix calculation one by one
# 1. Input the Data
y_vec <- c(193, 230, 172, 91, 113, 125)
x1 <- c(1.6, 15.5, 22.0, 43.0, 33.0, 40.0)
x2 <- c(851, 816, 1058, 1201, 1357, 1115)

# 2. Construct the Design Matrix X (Section 1.1)
# We add a column of 1s for the intercept (beta0)
X <- cbind(1, x1, x2)
cat("--- Design Matrix X ---\n")
```

```
## --- Design Matrix X ---
```

```
print(X)
```

```
##          x1   x2
## [1,] 1  1.6  851
## [2,] 1 15.5  816
## [3,] 1 22.0 1058
## [4,] 1 43.0 1201
## [5,] 1 33.0 1357
## [6,] 1 40.0 1115
```

```
# 3. Calculate X' (Transpose) - Section 1.3
Xt <- t(X)
cat("\n--- X' (Transpose) ---\n")
```

```
##
## --- X’ (Transpose) ---
```

```
print(Xt)
```

```
##     [,1]  [,2] [,3] [,4] [,5] [,6]
##      1.0   1.0    1    1    1    1
## x1   1.6  15.5   22   43   33   40
## x2 851.0 816.0 1058 1201 1357 1115
```

```
# 4. Calculate X'X (Sum of Squares and Cross-products) - Section 1.4
XtX <- Xt %*% X
cat("\n--- X'X ---\n")
```

```
##
## --- X’X ---
```

```
print(XtX)
```

```
##                  x1        x2
##       6.0    155.10    6398.0
## x1  155.1   5264.81  178309.6
## x2 6398.0 178309.60 7036496.0
```

```r
# 5. Calculate (X'X)^-1 (Inverse) - Section 1.4
# Note: solve() computes the inverse in R
XtX_inv <- solve(XtX)
cat("\n--- (X'X)^-1 (Inverse) ---\n")
```

```
##
## --- (X'X)^-1 (Inverse) ---
```

```r
print(XtX_inv)
```

```
##                            x1            x2
##     8.595095903  0.0809578675 -9.866699e-03
## x1  0.080957868  0.0021024502 -1.268892e-04
## x2 -0.009866699 -0.0001268892  1.232896e-05
```

```r
# 6. Calculate X'Y - Section 1.4
XtY <- Xt %*% y_vec
cat("\n--- X'Y ---\n")
```

```
##
## --- X'Y ---
```

```r
print(XtY)
```

```
##        [,1]
##       924.0
## x1  20299.8
## x2 935906.0
```

```r
# 7. Finally, calculate Beta_hat = (X'X)^-1 * X'Y
beta_hat <- XtX_inv %*% XtY
cat("\n--- Final Beta_hat Estimates ---\n")
```

```
##
## --- Final Beta_hat Estimates ---
```

```r
print(beta_hat)
```

```
##           [,1]
##    350.9942706
## x1  -1.2719944
## x2  -0.1539042
```

Question 2.1(b) solving one by one:

```r
# 1. Total Sum of Squares (SST)
n <- length(y_vec)
SST <- sum(y_vec^2) - (sum(y_vec)^2)/n
SST
```

```
## [1] 14112
```

```r
# 2. Regression Sum of Squares (SSR)
# Formula: Beta_hat' * XtY - Correction Factor
SSR <- t(beta_hat) %*% XtY - (sum(y_vec)^2)/n
SSR
```

```
##          [,1]
## [1,] 12161.58
```

```r
# 3. Error Sum of Squares (SSE)
SSE <- SST - SSR
SSE
```

```
##          [,1]
## [1,] 1950.422
```

```r
# 4. F-Statistic
MSR <- SSR / 2
MSE <- SSE / (n - 3)
F_stat <- MSR / MSE
MSR
```

```
##          [,1]
## [1,] 6080.789
```

```r
MSE
```

```
##          [,1]
## [1,] 650.1407
```

```r
# 5. P-value
p_val <- pf(F_stat, 2, 3, lower.tail = FALSE)

cat("F-statistic:", F_stat, "\nP-value:", p_val)
```

```
## F-statistic: 9.353035
## P-value: 0.05138189
```

Question 2.1(c) solving one by one:

```r
# 1. Calculate the Variance-Covariance Matrix
# This scales your inverse matrix by the Error Mean Square
#var_beta_matrix <- MSE * XtX_inv

# 2. Extract the Diagonal elements
# These are the 'variances' of beta0, beta1, and beta2
#variances <- diag(MSE * XtX_inv)

# 3. Take the Square Root to get Standard Errors
# This is what you use in the denominator of your T-test
se_betas <- sqrt(diag(as.numeric(MSE) * XtX_inv))

# View the results
print(se_betas)
```

```
##                      x1          x2
## 74.75307396  1.16914009  0.08952967
```

```r
# T-test for Viscosity (x1)
t_val_x1 <- beta_hat[2] / se_betas[2]

# T-test for Load (x2)
t_val_x2 <- beta_hat[3] / se_betas[3]

cat("T-stat for x1:", t_val_x1, "\nT-stat for x2:", t_val_x2)
```

```
## T-stat for x1: -1.087974
## T-stat for x2: -1.71903
```

Quation 2.2(a)(b)(c) solving using 2.1 values:

```
# Calculate the interaction column
x1x2 <- x1 * x2

# Build the new 6x4 Design Matrix
X2 <- cbind(1, x1, x2, x1x2)
print(X2)
```

```
##          x1   x2    x1x2
## [1,] 1  1.6  851  1361.6
## [2,] 1 15.5  816 12648.0
## [3,] 1 22.0 1058 23276.0
## [4,] 1 43.0 1201 51643.0
## [5,] 1 33.0 1357 44781.0
## [6,] 1 40.0 1115 44600.0
```

```
# copy paste

# 3. Calculate X' (Transpose) - Section 1.3
X2t <- t(X2)
cat("\n--- X' (Transpose) ---\n")
```

```
##
## --- X’ (Transpose) ---
```

```
print(X2t)
```

```
##        [,1]    [,2]  [,3]  [,4]  [,5]  [,6]
##         1.0     1.0     1     1     1     1
## x1      1.6    15.5    22    43    33    40
## x2    851.0   816.0  1058  1201  1357  1115
## x1x2 1361.6 12648.0 23276 51643 44781 44600
```

```
# 4. Calculate X'X (Sum of Squares and Cross-products) - Section 1.4
X2tX2 <- X2t %*% X2
cat("\n--- X'X ---\n")
```

```
##
## --- X’X ---
```

```
print(X2tX2)
```

```
##                      x1          x2         x1x2
##           6.0     155.10      6398.0     178309.6
## x1      155.1    5264.81    178309.6    6192716.6
## x2     6398.0  178309.60   7036496.0  208625557.6
## x1x2 178309.6 6192716.56 208625557.6 7365095444.6
```

```r
# 5. Calculate (X'X)^-1 (Inverse) - Section 1.4
# Note: solve() computes the inverse in R
X2tX2_inv <- solve(X2tX2)
cat("\n--- (X'X)^-1 (Inverse) ---\n")
```

```
##
## --- (X'X)^-1 (Inverse) ---
```

```r
print(X2tX2_inv)
```

```
##                              x1            x2          x1x2
##      69.816949529 -2.3748471660 -7.736482e-02  2.497999e-03
## x1   -2.374847166  0.1006126717  2.580677e-03 -1.002028e-04
## x2   -0.077364822  0.0025806771  8.674678e-05 -2.754086e-06
## x1x2  0.002497999 -0.0001002028 -2.754086e-06  1.019244e-07
```

```r
# 6. Calculate X'Y - Section 1.4
X2tY <- X2t %*% y_vec
cat("\n--- X'Y ---\n")
```

```
##
## --- X'Y ---
```

```r
print(XtY)
```

```
##        [,1]
##       924.0
## x1  20299.8
## x2 935906.0
```

```r
# 7. Finally, calculate Beta_hat = (X'X)^-1 * X'Y
beta_hat_new <- X2tX2_inv %*% X2tY
cat("\n--- Final Beta_hat Estimates ---\n")
```

```
##
## --- Final Beta_hat Estimates ---
```

```r
print(beta_hat_new)
```

```
##               [,1]
##      125.865548338
## x1     7.758641128
## x2     0.094303967
## x1x2  -0.009185794
```

```r
# 1. Total Sum of Squares (SST)
n <- length(y_vec)
SST_new <- sum(y_vec^2) - (sum(y_vec)^2)/n
SST_new
```

```
## [1] 14112
```

```r
# 2. Regression Sum of Squares (SSR)
# Formula: Beta_hat' * XtY - Correction Factor
SSR_new <- t(beta_hat_new) %*% X2tY - (sum(y_vec)^2)/n
SSR_new
```

```
##          [,1]
## [1,] 12989.43
```

```r
# 3. Error Sum of Squares (SSE)
SSE_new <- SST_new - SSR_new
SSE_new
```

```
##          [,1]
## [1,] 1122.565
```

```r
# 4. F-Statistic
MSR_new <- SSR_new / 3
MSE_new <- SSE_new / (n - 4)
F_stat_new <- MSR_new / MSE_new
MSR_new
```

```
##          [,1]
## [1,] 4329.812
```

```r
MSE_new
```

```
##          [,1]
## [1,] 561.2826
```

```r
# 5. P-value
p_val_new <- pf(F_stat_new, 1, 4, lower.tail = FALSE)

cat("F-statistic:", F_stat_new, "\nP-value:", p_val_new)
```

```
## F-statistic: 7.714138
## P-value: 0.04994945
```

```r
se_betas_new <- sqrt(diag(as.numeric(MSE) * X2tX2_inv))

# View the results
print(se_betas_new)
```

```
##                        x1           x2         x1x2
## 2.130513e+02 8.087793e+00 2.374818e-01 8.140344e-03
```

```r
# T-test for Viscosity (x1)
t_val_x1 <- beta_hat_new[2] / se_betas_new[2]

# T-test for Load (x2)
```

```r
t_val_x2 <- beta_hat_new[3] / se_betas_new[3]

# T-test for interaction term (x1x2)
t_val_x1x2 <- beta_hat_new[4] / se_betas_new[4]

cat("T-stat for x1:", t_val_x1, "\nT-stat for x2:", t_val_x2, "\nT-stat for x1x2:", t_val_x1x2)
```

```
## T-stat for x1: 0.9593026
## T-stat for x2: 0.3970997
## T-stat for x1x2: -1.128428
```

```r
#partial F test
partial_F_stat <- ((SSE - SSE_new)/(1))/((SSE_new)/(n-4))
partial_F_stat
```

```
##          [,1]
## [1,] 1.474938
```

Question 2.3(b):

```r
# Define the point of interest
x0_m1 <- c(1, 25, 1000)
x0_m2 <- c(1, 25, 1000, 25000)

# Variance for Model 2.1 (using MSE = 882.8 from 2.1)
var_y0_m1 <- 650.1407 * (t(x0_m1) %*% XtX_inv %*% x0_m1)
se_y0_m1 <- sqrt(var_y0_m1)

# Variance for Model 2.2 (using MSE = 561.3 from 2.2)
var_y0_m2 <- 561.2826 * (t(x0_m2) %*% X2tX2_inv %*% x0_m2)
se_y0_m2 <- sqrt(var_y0_m2)

# 95% CI: point_estimate +/- (t_critical * se)
# For m1, df=3 (t=3.182). For m2, df=2 (t=4.303)
ci_m1 <- c(165.194 - 3.182*se_y0_m1, 165.194 + 3.182*se_y0_m1)
ci_m2 <- c(184.4906935 - 4.303*se_y0_m2, 184.4906935 + 4.303*se_y0_m2)
c(ci_m1,ci_m2)
```

```
## [1] 128.1799 202.2081 102.0833 266.8981
```

Question 2.4

```
# Response variable (Pull Strength)
Y <- c(8.0, 8.3, 8.5, 8.8, 9.0, 9.3, 9.3, 9.5, 9.8, 10.0, 10.3, 10.5, 10.8, 11.0, 11.3, 11.5, 11.8, 12.3

# Predictor variables
x2 <- c(19.6, 19.8, 19.6, 19.4, 18.6, 18.8, 20.4, 19.0, 20.8, 19.9, 18.0, 20.6, 20.2, 20.2, 19.2, 17.0,
x3 <- c(29.6, 32.4, 31.0, 32.4, 28.6, 30.6, 32.4, 32.6, 32.2, 31.8, 32.6, 33.4, 31.8, 32.4, 31.4, 33.2,
x4 <- c(94.9, 89.7, 96.2, 95.6, 86.5, 84.5, 88.8, 85.7, 93.6, 86.0, 87.1, 93.1, 83.4, 94.5, 83.4, 85.2,
x5 <- c(2.1, 2.1, 2.0, 2.2, 2.0, 2.1, 2.2, 2.1, 2.3, 2.1, 2.0, 2.1, 2.2, 2.1, 1.9, 2.1, 2.0, 2.1, 1.9)

# Create Design Matrix X (including Intercept)
X <- cbind(1, x2, x3, x4, x5)
n <- length(Y)
n
```

```
## [1] 19
```

```
p <- ncol(X)

# Solve for Beta coefficients: (X'X)^-1 * X'Y
XtX <- t(X) %*% X
XtY <- t(X) %*% Y
beta_hat <- solve(XtX) %*% XtY

# Display coefficients
print(beta_hat)
```

```
##          [,1]
##     7.4578066
## x2 -0.0297028
## x3  0.5205101
## x4 -0.1018024
## x5 -2.1605807
```

```
# ANOVA Step-by-Step
y_hat <- X %*% beta_hat
SSE <- sum((Y - y_hat)^2)
SSE
```

```
## [1] 10.90914
```

```
SST <- sum((Y - mean(Y))^2)
SSR <- SST - SSE


df_r <- 5 - 1  # p - 1 = 4
df_e <- 19 - 5 # n - p = 14

MSR <- SSR / df_r  # 5.578
MSE <- SSE / df_e  # 0.779 (Unbiased estimator s^2) [cite: 253]
MSR
```

```
## [1] 5.577979
```

```
MSE
```

```
## [1] 0.7792241
```

```
F_stat <- MSR / MSE # 7.158
F_stat
```

```
## [1] 7.158376
```

```
p_value <- 1 - pf(F_stat, df_r, df_e) # 0.0024

# New observation vector
x_new <- c(1, 20, 30, 90, 2.0)

# Prediction calculation
Y_pred <- sum(x_new * beta_hat)
print(Y_pred)
```

```
## [1] 8.995678
```

Question 2.5:

```r
# R code for the standard errors
C <- solve(t(X) %*% X)
SEs <- sqrt(MSE * diag(C))
SEs
```

```
##                   x2         x3         x4         x5
## 7.22629570 0.26326920 0.13590240 0.05339231 2.39473136
```

```r
# 4. T-test Statistics
t_stats <- beta_hat / SEs

# 5. Output Table
results <- data.frame(
  Estimate = beta_hat,
  Std_Error = SEs,
  T_Statistic = t_stats,
  P_Value = 2 * (1 - pt(abs(t_stats), n - p))
)
print(round(results, 4))
```

```
##    Estimate Std_Error T_Statistic P_Value
##      7.4578    7.2263      1.0320  0.3196
## x2  -0.0297    0.2633     -0.1128  0.9118
## x3   0.5205    0.1359      3.8300  0.0018
## x4  -0.1018    0.0534     -1.9067  0.0773
## x5  -2.1606    2.3947     -0.9022  0.3822
```

Question 2.6:

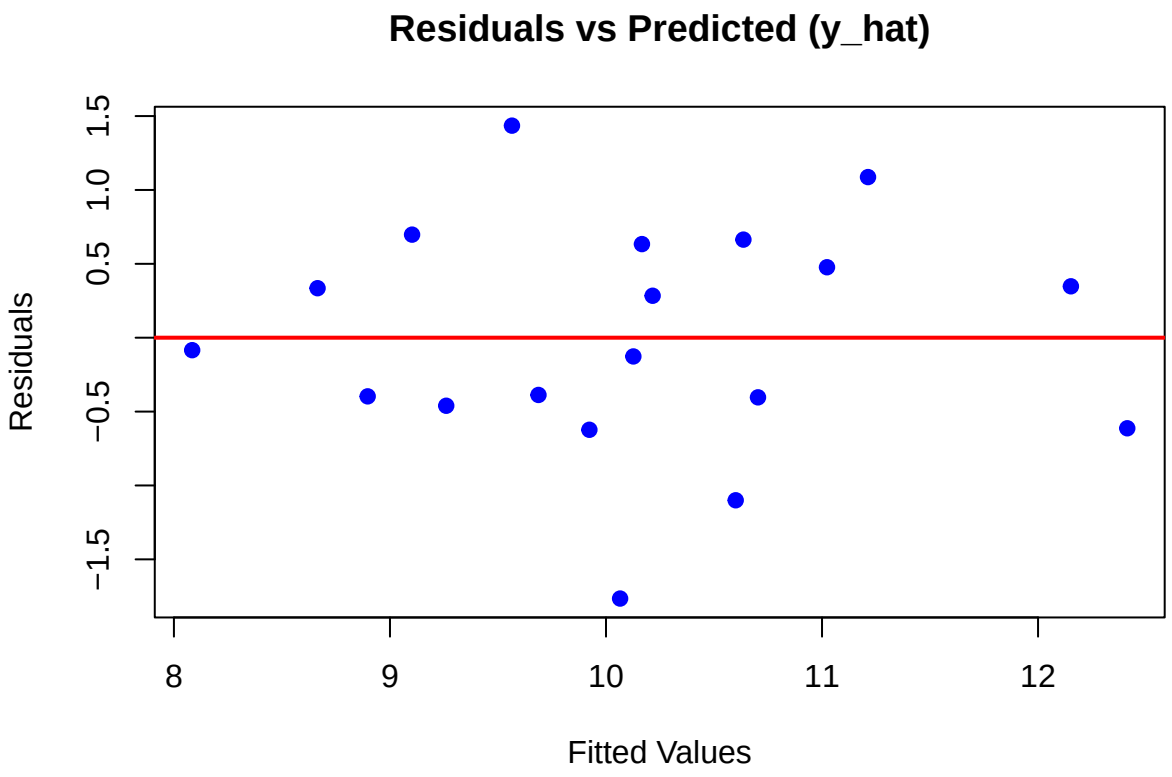
```
# 1. Manual Residual Calculation (from Section 1.10)
# y_hat = X * beta_hat
y_hat <- X %*% beta_hat
residuals <- Y - y_hat

# 2. Check lengths to avoid the error
length(y_hat)      # Should be 19
```

```
## [1] 19
```

```
length(residuals)  # Should be 19
```

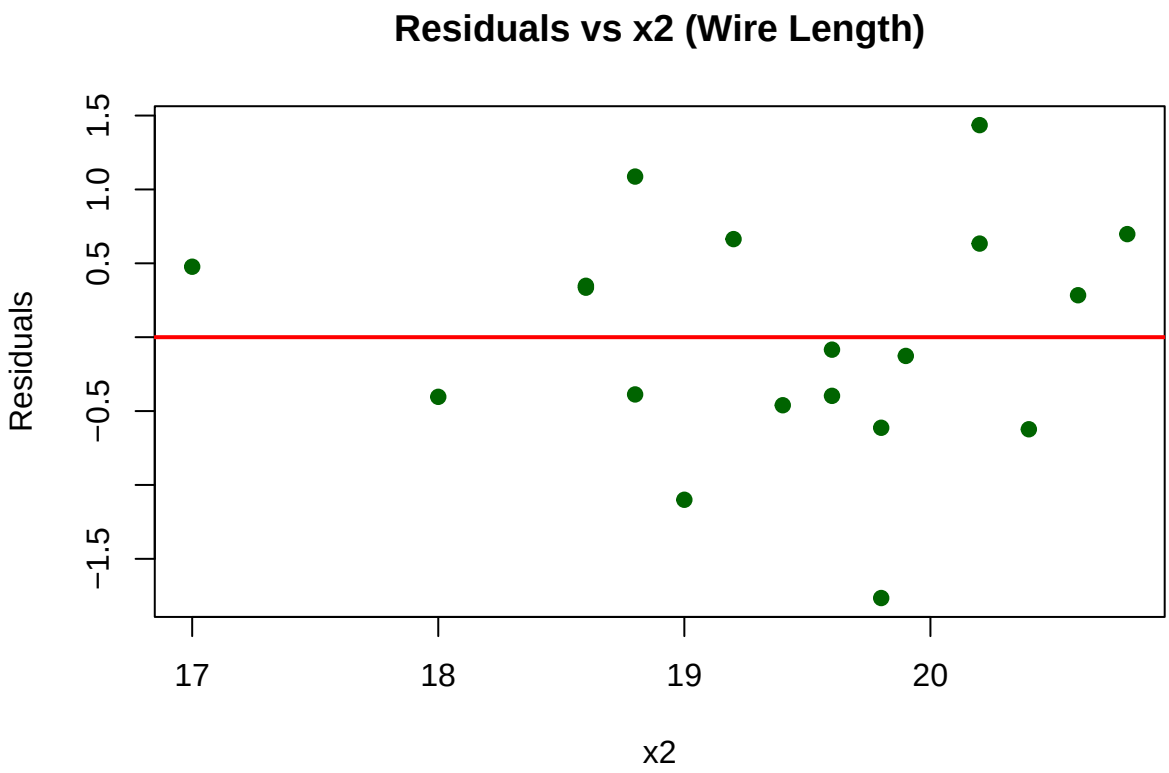
```
## [1] 19
```

```
# 2.6(a) Plot Residuals vs Fitted values
plot(y_hat, residuals,
     main="Residuals vs Predicted (y_hat)",
     xlab="Fitted Values", ylab="Residuals",
     pch=19, col="blue")
abline(h = 0, col="red", lwd=2)
```

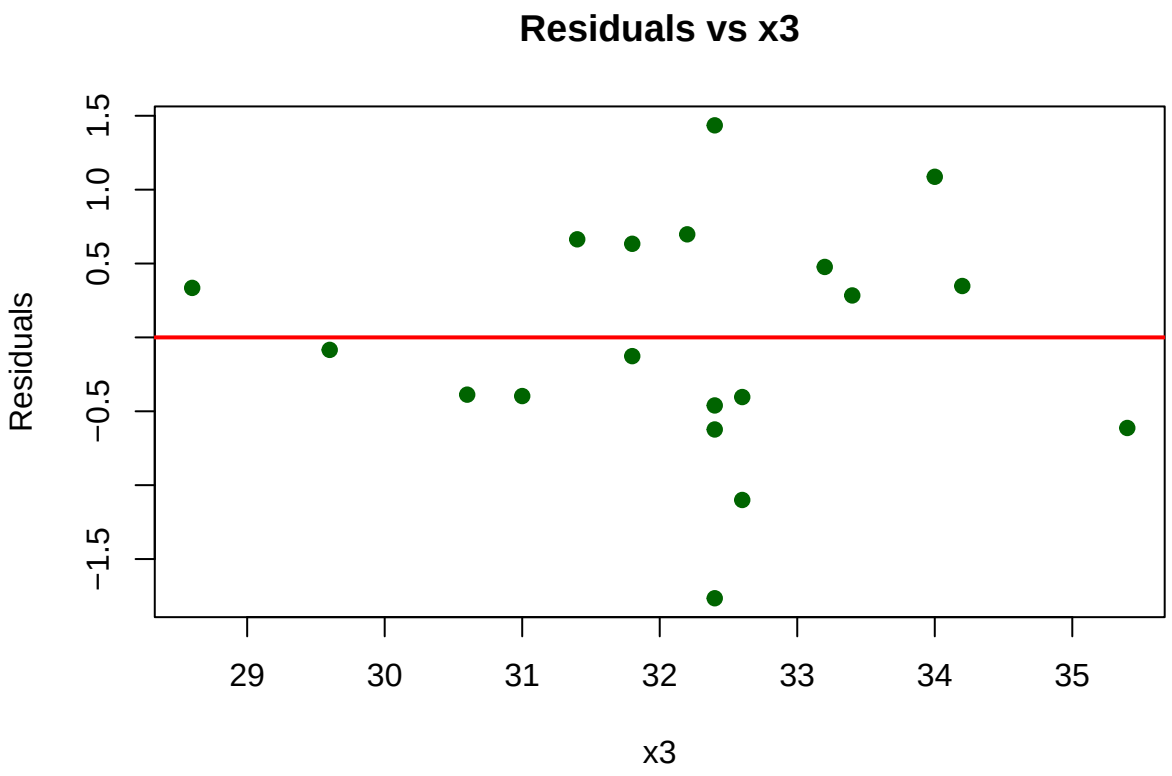



```
# 2.6(a) Plot Residuals vs Regressors (Example for x2)
# Repeat this for x3, x4, and x5
plot(x2, residuals,
     main="Residuals vs x2 (Wire Length)",
```

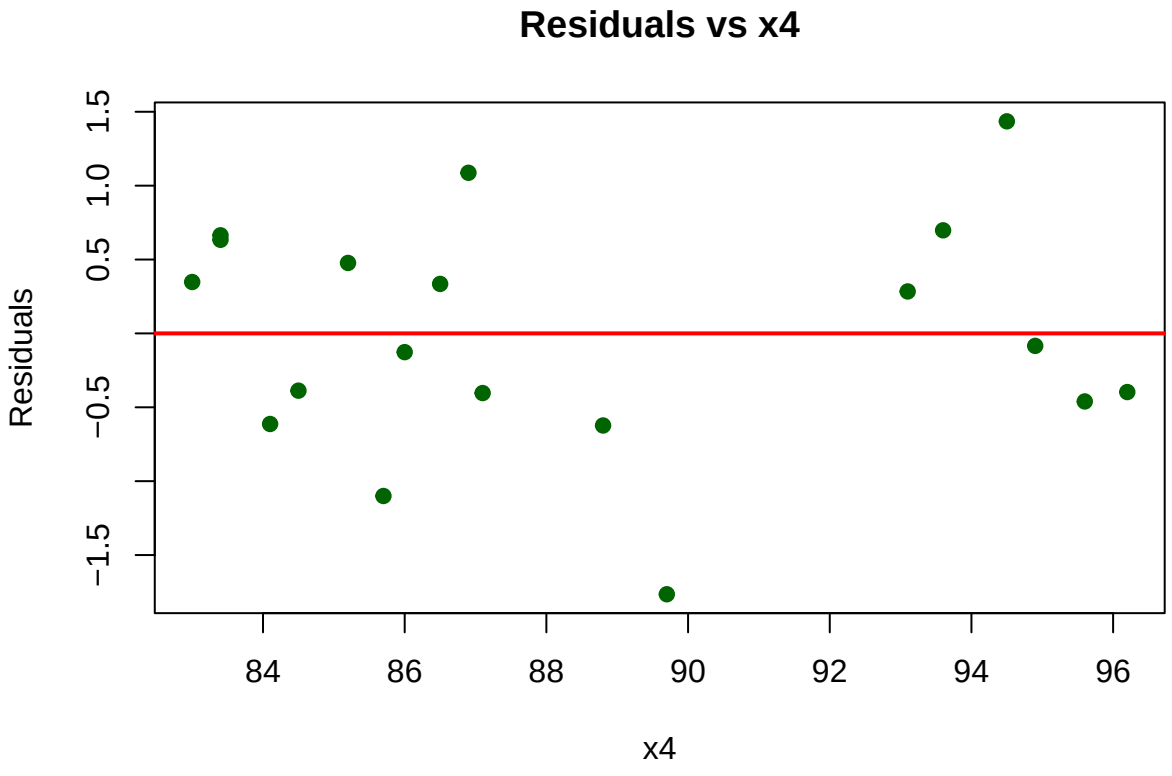
```
     xlab="x2", ylab="Residuals",
     pch=19, col="darkgreen")
abline(h = 0, col="red", lwd=2)
```



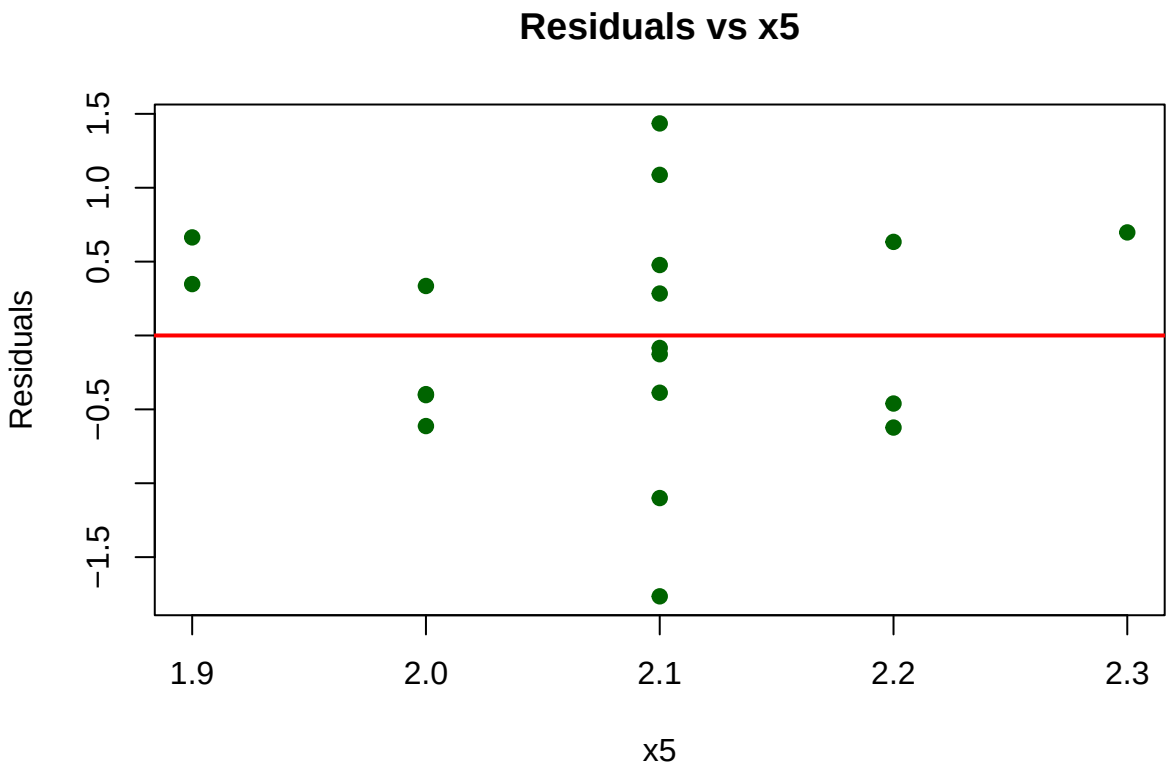

```
plot(x3, residuals,
     main="Residuals vs x3",
     xlab="x3", ylab="Residuals",
     pch=19, col="darkgreen")
abline(h = 0, col="red", lwd=2)
```



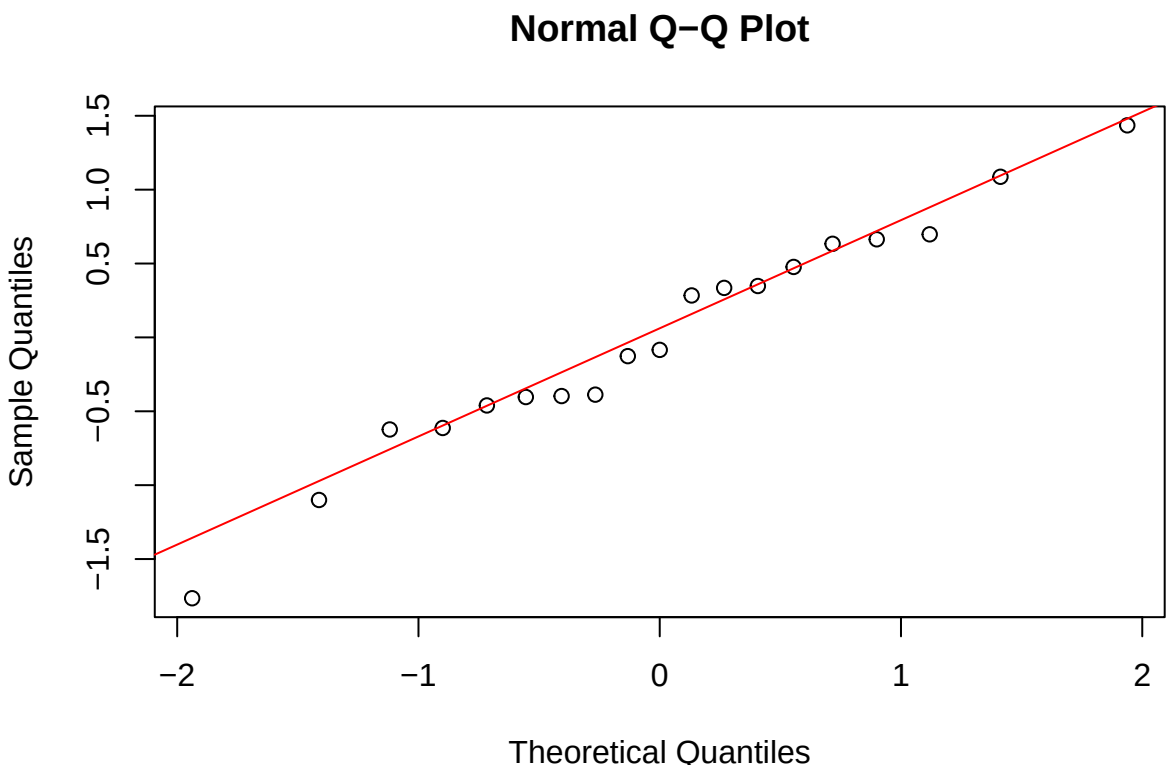

```
plot(x4, residuals,
     main="Residuals vs x4",
     xlab="x4", ylab="Residuals",
     pch=19, col="darkgreen")
abline(h = 0, col="red", lwd=2)
```



```r
plot(x5, residuals,
     main="Residuals vs x5",
     xlab="x5", ylab="Residuals",
     pch=19, col="darkgreen")
abline(h = 0, col="red", lwd=2)
```




```r
# 2.6(b) Normal Probability Plot (Manual)
# Sort residuals to create the Q-Q plot
qqnorm(residuals)
qqline(residuals, col="red")
```




```r
# 1. Calculate the Hat Matrix
# X is your matrix with the column of 1s
H <- X %*% solve(t(X) %*% X) %*% t(X)

# 2. Extract the diagonal (leverage values)
leverages <- diag(H)

# 3. Use the threshold (2p/n)
p <- ncol(X) # which is 5
n <- nrow(X) # which is 19
threshold <- 2 * p / n # 0.5263

# 4. Identify influential points
influential_points <- which(leverages > threshold)
print(influential_points)
```

```
## [1] 16
```

Question 2.7

```r
# Setup
t_crit <- qt(0.975, df = 14) # 2.144787
s <- sqrt(0.7792)            # From 2.5(a)

# Standard Errors (from diag of MSE * (X'X)^-1)
se_beta <- sqrt(diag(MSE * solve(t(X) %*% X)))

# Lower and Upper Bounds
LCL_beta <- beta_hat - t_crit * se_beta
UCL_beta <- beta_hat + t_crit * se_beta

# Results Table
data.frame(Estimate = beta_hat, Lower = LCL_beta, Upper = UCL_beta)
```

```
##      Estimate      Lower       Upper
##     7.4578066 -8.0410562 22.95666941
## x2 -0.0297028 -0.5943591  0.53495347
## x3  0.5205101  0.2290284  0.81199174
## x4 -0.1018024 -0.2163175  0.01271274
## x5 -2.1605807 -7.2967686  2.97560725
```

```r
# Define c (x0) vector
c_vec <- matrix(c(1, 20, 30, 90, 2.0), ncol = 1)

# Point Estimate: c' * beta_hat
y_hat_0 <- t(c_vec) %*% beta_hat

# Standard Error of Mean: s * sqrt(c' * (X'X)^-1 * c)
C_matrix <- solve(t(X) %*% X)
se_mean <- s * sqrt(t(c_vec) %*% C_matrix %*% c_vec)

# Confidence Interval
LCL_mean <- y_hat_0 - t_crit * se_mean
UCL_mean <- y_hat_0 + t_crit * se_mean

cat("Point Estimate:", y_hat_0, "\nInterval:", LCL_mean, "to", UCL_mean)
```

```
## Point Estimate: 8.995678
## Interval: 7.982399 to 10.00896
```

Question 2.8:

```r
# Full data including x1 and x6
y  <- c(8.0, 8.3, 8.5, 8.8, 9.0, 9.3, 9.3, 9.5, 9.8, 10.0, 10.3, 10.5, 10.8, 11.0, 11.3, 11.5, 11.8, 12
x1 <- c(5.2, 5.2, 5.8, 6.4, 5.8, 5.2, 5.6, 6.0, 5.2, 5.8, 6.4, 6.0, 6.2, 6.2, 6.2, 5.6, 6.0, 5.8, 5.6)
x2 <- c(19.6, 19.8, 19.6, 19.4, 18.6, 18.8, 20.4, 19.0, 20.8, 19.9, 18.0, 20.6, 20.2, 20.2, 19.2, 17.0,
x3 <- c(29.6, 32.4, 31.0, 32.4, 28.6, 30.6, 32.4, 32.6, 32.2, 31.8, 32.6, 33.4, 31.8, 32.4, 31.4, 33.2,
x4 <- c(94.9, 89.7, 96.2, 95.6, 86.5, 84.5, 88.8, 85.7, 93.6, 86.0, 87.1, 93.1, 83.4, 94.5, 83.4, 85.2,
x5 <- c(2.1, 2.1, 2.0, 2.2, 2.0, 2.1, 2.2, 2.1, 2.3, 2.1, 2.0, 2.1, 2.2, 2.1, 1.9, 2.1, 2.0, 2.1, 1.9)
x6 <- c(2.3, 1.8, 2.0, 2.1, 1.8, 2.1, 1.9, 1.9, 2.1, 1.8, 1.6, 2.1, 2.1, 1.9, 1.8, 2.1, 1.8, 1.8, 2.0)

# Design Matrix X (p = 7 including intercept)
X_full <- cbind(1, x1, x2, x3, x4, x5, x6)
n <- length(y)
p_full <- ncol(X_full)

# Estimates
beta_full <- solve(t(X_full) %*% X_full) %*% t(X_full) %*% y
sse_full <- sum((y - X_full %*% beta_full)^2)
mse_full <- sse_full / (n - p_full) # df = 19 - 7 = 12
```