Linear Regression Fundamentals Problem Set - Solutions

Solution Key

2025-09-04

Table of contents

1 Problem 1 Solution **1**

2 Problem 2 Solution **4**

3 Problem 3 Solution **5**

4 Problem 4 Solution **8**

5 Problem 5 Solution **10**

6 Problem 6 Solution **11**


```
library(ggplot2)
library(dplyr)
```

1 Problem 1 Solution

Data: | Observation | x | x | y | |————-|—-|—-|—-| | 1 | 2 | 1 | 5.2| | 2 | 3 | 4 | 8.1| | 3 | 1 | 2 | 4.8| | 4 | 4 | 3 | 9.5|

Part A: Design matrix $\mathbf{X}$ for multiple regression with intercept:

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 3 & 4 \\ 1 & 1 & 2 \\ 1 & 4 & 3 \end{bmatrix}$$

Part B: Calculate $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{y}$:

First, $\mathbf{X}^T$:

$$\mathbf{X}^T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 3 & 1 & 4 \\ 1 & 4 & 2 & 3 \end{bmatrix}$$

$\mathbf{X}^T\mathbf{X}$:

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 3 & 1 & 4 \\ 1 & 4 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \\ 1 & 3 & 4 \\ 1 & 1 & 2 \\ 1 & 4 & 3 \end{bmatrix}$$

Computing each element: - (1,1): $1\times1+1\times1+1\times1+1\times1 = 4$ - (1,2): $1\times2+1\times3+1\times1+1\times4 = 10$ - (1,3): $1\times1+1\times4+1\times2+1\times3 = 10$ - (2,2): $2\times2+3\times3+1\times1+4\times4 = 4+9+1+16 = 30$ - (2,3): $2\times1+3\times4+1\times2+4\times3 = 2+12+2+12 = 28$ - (3,3): $1\times1+4\times4+2\times2+3\times3 = 1+16+4+9 = 30$

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 4 & 10 & 10 \\ 10 & 30 & 28 \\ 10 & 28 & 30 \end{bmatrix}$$

$\mathbf{X}^T\mathbf{y}$:

$$\mathbf{X}^T\mathbf{y} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 3 & 1 & 4 \\ 1 & 4 & 2 & 3 \end{bmatrix} \begin{bmatrix} 5.2 \\ 8.1 \\ 4.8 \\ 9.5 \end{bmatrix} = \begin{bmatrix} 27.6 \\ 78.1 \\ 77.3 \end{bmatrix}$$

Part C: Normal equations: $\mathbf{X}^T\mathbf{X}\widehat{\beta} = \mathbf{X}^T\mathbf{y}$

$$\begin{bmatrix} 4 & 10 & 10 \\ 10 & 30 & 28 \\ 10 & 28 & 30 \end{bmatrix} \begin{bmatrix} \widehat{\beta}_0 \\ \widehat{\beta}_1 \\ \widehat{\beta}_2 \end{bmatrix} = \begin{bmatrix} 27.6 \\ 78.1 \\ 77.3 \end{bmatrix}$$

To solve: $\widehat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

Part D: Verification in R:

```r
# Data setup
X <- matrix(c(1, 2, 1,
              1, 3, 4,
              1, 1, 2,
              1, 4, 3), byrow = TRUE, ncol = 3)
y <- c(5.2, 8.1, 4.8, 9.5)

# Manual calculation
XtX <- t(X) %*% X
Xty <- t(X) %*% y
beta_hat <- solve(XtX) %*% Xty

print("Manual calculation:")
```

```
[1] "Manual calculation:"
```

```r
print(beta_hat)
```

```
     [,1]
[1,] 2.15
[2,] 1.40
[3,] 0.50
```

```r
# Using lm()
data_df <- data.frame(x1 = c(2, 3, 1, 4), x2 = c(1, 4, 2, 3), y = y)
lm_result <- lm(y ~ x1 + x2, data = data_df)
print("Using lm():")
```

```
[1] "Using lm():"
```

```r
print(coefficients(lm_result))
```

```
(Intercept)          x1          x2
       2.15        1.40        0.50
```

---

2 Problem 2 Solution

Using $\hat{\beta}$ from Problem 1:

Part A: Predicted values $\hat{y}_i$:

For observation i: $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{1i} + \hat{\beta}_2 x_{2i}$

```
# Calculate predicted values
y_hat <- X %*% beta_hat
print("Predicted values:")
```

```
[1] "Predicted values:"
```

```
for(i in 1:4) {
  cat(sprintf("ŷ_%d = %.3f\n", i, y_hat[i]))
}
```

```
ŷ_1 = 5.450
ŷ_2 = 8.350
ŷ_3 = 4.550
ŷ_4 = 9.250
```

Part B: Residuals $\hat{\varepsilon}_i = y_i - \hat{y}_i$:

```
# Calculate residuals
residuals <- y - y_hat
print("Residuals:")
```

```
[1] "Residuals:"
```

```
for(i in 1:4) {
  cat(sprintf("ê_%d = %.3f\n", i, residuals[i]))
}
```

```
ê_1 = -0.250
ê_2 = -0.250
ê_3 = 0.250
ê_4 = 0.250
```

Part C: Sum of squared errors (SSE):

Individual terms: $\text{SSE} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

Matrix form: $\text{SSE} = (\mathbf{y} - \mathbf{X}\hat{\beta})^T(\mathbf{y} - \mathbf{X}\hat{\beta})$

```
# Individual terms
SSE_individual <- sum(residuals^2)
print(paste("SSE (individual terms):", round(SSE_individual, 6)))
```

```
[1] "SSE (individual terms): 0.25"
```

```
# Matrix form
SSE_matrix <- t(residuals) %*% residuals
print(paste("SSE (matrix form):", round(SSE_matrix, 6)))
```

```
[1] "SSE (matrix form): 0.25"
```

Part D: Sum of residuals and geometric interpretation:

```
print(paste("Sum of residuals:", round(sum(residuals), 10)))
```

```
[1] "Sum of residuals: 0"
```

The residuals sum to approximately zero because $\mathbf{X}^T\hat{\varepsilon} = \mathbf{0}$. The first row of this equation represents the constraint that $\sum \hat{\varepsilon}_i = 0$. Geometrically, this occurs because the residual vector is orthogonal to the column space of $\mathbf{X}$, which includes the vector of ones (intercept column).

---

3 Problem 3 Solution

Part A: Hat matrix $\mathbf{H}$ for:

$$\mathbf{X} = \begin{bmatrix} 1 & 2 \\ 1 & 4 \\ 1 & 6 \end{bmatrix}$$

First, calculate $\mathbf{X}^T\mathbf{X}$:

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 4 & 6 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 1 & 4 \\ 1 & 6 \end{bmatrix} = \begin{bmatrix} 3 & 12 \\ 12 & 56 \end{bmatrix}$$

Next, $(\mathbf{X}^T\mathbf{X})^{-1}$:

$$(\mathbf{X}^T\mathbf{X})^{-1} = \frac{1}{3 \times 56 - 12 \times 12} \begin{bmatrix} 56 & -12 \\ -12 & 3 \end{bmatrix} = \frac{1}{24} \begin{bmatrix} 56 & -12 \\ -12 & 3 \end{bmatrix}$$

$$= \begin{bmatrix} 7/3 & -1/2 \\ -1/2 & 1/8 \end{bmatrix}$$

Now calculate $\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$:

```
# Problem 3 calculations
X3 <- matrix(c(1, 2,
               1, 4,
               1, 6), byrow = TRUE, ncol = 2)

# Calculate hat matrix
XtX3 <- t(X3) %*% X3
XtX_inv3 <- solve(XtX3)
H <- X3 %*% XtX_inv3 %*% t(X3)

print("Hat matrix H:")
```

```
[1] "Hat matrix H:"
```

```
print(H)
```

```
           [,1]      [,2]       [,3]
[1,]  0.8333333 0.3333333 -0.1666667
[2,]  0.3333333 0.3333333  0.3333333
[3,] -0.1666667 0.3333333  0.8333333
```

Part B: Verify $\mathbf{H}^2 = \mathbf{H}$ (idempotent property):

```
# Check idempotent property
H_squared <- H %*% H
print("H^2:")
```

```
[1] "H^2:"
```

```
print(H_squared)
```

```
           [,1]      [,2]       [,3]
[1,]  0.8333333 0.3333333 -0.1666667
[2,]  0.3333333 0.3333333  0.3333333
[3,] -0.1666667 0.3333333  0.8333333
```

```
print("H^2 = H?")
```

```
[1] "H^2 = H?"
```

```
print(all.equal(H, H_squared))
```

```
[1] TRUE
```

Part C: Verify $\mathbf{H}^T = \mathbf{H}$ (symmetric property):

```
# Check symmetric property
print("H^T:")
```

```
[1] "H^T:"
```

```
print(t(H))
```

```
           [,1]      [,2]       [,3]
[1,]  0.8333333 0.3333333 -0.1666667
[2,]  0.3333333 0.3333333  0.3333333
[3,] -0.1666667 0.3333333  0.8333333
```

```
print("H^T = H?")
```

```
[1] "H^T = H?"
```

```r
print(all.equal(H, t(H)))
```

```
[1] TRUE
```

Geometric interpretation: Symmetry means that the projection operation works the same way regardless of the order of operations. If we project vector **a** onto the column space and then take the dot product with vector **b**, we get the same result as taking the dot product of **a** with the projection of **b**.

---

4 Problem 4 Solution

Using the design matrix from Problem 3: $\mathbf{X} = \begin{bmatrix} 1 & 2 \\ 1 & 4 \\ 1 & 6 \end{bmatrix}$

Part A: Two vectors in the column space of $\mathbf{X}$:

Any vector in the column space can be written as $\beta_0 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \beta_1 \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}$

Example 1: Choose $\beta_0 = 2, \beta_1 = 1$:

$$\mathbf{v_1} = 2 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + 1 \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \\ 8 \end{bmatrix}$$

Example 2: Choose $\beta_0 = 0, \beta_1 = 0.5$:

$$\mathbf{v_2} = 0 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + 0.5 \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

These represent predicted values from the regression model with different coefficient values.

Part B: Why $\mathbf{y} = \begin{bmatrix} 3 \\ 7 \\ 12 \end{bmatrix}$ likely doesn't lie in the column space:

```r
# Check if y is in column space
y4 <- c(3, 7, 12)

# If y were in column space, we could solve X*beta = y exactly
# This would mean residuals = 0
beta_exact <- solve(t(X3) %*% X3) %*% t(X3) %*% y4
y_pred <- X3 %*% beta_exact
residuals4 <- y4 - y_pred

print("If we fit the model:")
```

```
[1] "If we fit the model:"
```

```r
print(paste("Residuals:", paste(round(residuals4, 6), collapse = ", ")))
```

```
[1] "Residuals: 0.166667, -0.333333, 0.166667"
```

```r
print(paste("Sum of squared residuals:", sum(residuals4^2)))
```

```
[1] "Sum of squared residuals: 0.166666666666667"
```

Since we have residuals 0, the observed **y** doesn't lie exactly in the column space. Practically, this means our linear model cannot perfectly predict the observed data - there is unexplained variation (error).

Part C: Projection of **y** onto column space:

```r
# Calculate projection
y_hat4 <- H %*% y4
residuals_final <- y4 - y_hat4

print("Projection (predicted values):")
```

```
[1] "Projection (predicted values):"
```

```r
print(y_hat4)
```

```
          [,1]
[1,]  2.833333
[2,]  7.333333
[3,] 11.833333
```

```
print("Residuals:")
```

```
[1] "Residuals:"
```

```
print(residuals_final)
```

```
           [,1]
[1,]  0.1666667
[2,] -0.3333333
[3,]  0.1666667
```

```
print("X^T * residuals (should be ~0):")
```

```
[1] "X^T * residuals (should be ~0):"
```

```
print(t(X3) %*% residuals_final)
```

```
              [,1]
[1,] -8.881784e-15
[2,] -3.375078e-14
```

The fact that $\mathbf{X}^T\hat{\varepsilon} \approx \mathbf{0}$ confirms orthogonality.

---

5 Problem 5 Solution

Part A: Derivation of normal equations from orthogonality:

Starting with the geometric principle: $\mathbf{X}^T\hat{\varepsilon} = \mathbf{0}$

Since $\hat{\varepsilon} = \mathbf{y} - \mathbf{X}\hat{\beta}$, substitute:

$$\mathbf{X}^T(\mathbf{y} - \mathbf{X}\hat{\beta}) = \mathbf{0}$$

Distribute $\mathbf{X}^T$:

$$\mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\hat{\beta} = \mathbf{0}$$

Rearrange to get the normal equations:

$$\mathbf{X}^T\mathbf{X}\hat{\beta} = \mathbf{X}^T\mathbf{y}$$

Part B: Why we multiply by $(\mathbf{X}^T\mathbf{X})^{-1}$:

To isolate $\hat{\beta}$, we multiply both sides by $(\mathbf{X}^T\mathbf{X})^{-1}$:

$$(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

The inverse $(\mathbf{X}^T\mathbf{X})^{-1}$ exists when $\mathbf{X}^T\mathbf{X}$ is non-singular, which requires: - The columns of $\mathbf{X}$ are linearly independent - We have at least as many observations as parameters ($n \geq p$) - No perfect multicollinearity among predictors

Part C: Verification that this minimizes SSE:

The least squares estimator satisfies the orthogonality condition by construction. Since orthogonal projection gives the minimum distance from a point to a subspace, our solution minimizes $||\mathbf{y} - \mathbf{X}\hat{\beta}||^2 = \text{SSE}$.

---

6 Problem 6 Solution

Data:

```
# Problem 6 data
x6 <- 1:10
y6 <- c(4.8, 6.2, 8.1, 9.9, 11.5, 13.8, 15.2, 17.1, 18.9, 21.3)
print(data.frame(x = x6, y = y6))
```

```
    x    y
1   1  4.8
2   2  6.2
3   3  8.1
4   4  9.9
5   5 11.5
6   6 13.8
7   7 15.2
8   8 17.1
9   9 18.9
10 10 21.3
```

Part A: Matrix implementation:

```r
# Create design matrix
X6 <- cbind(1, x6)
print("Design matrix X (first 5 rows):")
```

```
[1] "Design matrix X (first 5 rows):"
```

```r
print(head(X6, 5))
```

```
      x6
[1,] 1  1
[2,] 1  2
[3,] 1  3
[4,] 1  4
[5,] 1  5
```

```r
# Calculate coefficients using matrix operations
XtX6 <- t(X6) %*% X6
Xty6 <- t(X6) %*% y6
beta_hat6 <- solve(XtX6) %*% Xty6

print("Coefficients (matrix method):")
```

```
[1] "Coefficients (matrix method):"
```

```
print(beta_hat6)
```

```
       [,1]
   2.660000
x6 1.821818
```

Part B: Comparison with lm():

```
# Compare with lm()
lm6 <- lm(y6 ~ x6)
print("Coefficients (lm method):")
```

```
[1] "Coefficients (lm method):"
```

```
print(coefficients(lm6))
```

```
(Intercept)          x6
   2.660000    1.821818
```

```
print("Difference:")
```

```
[1] "Difference:"
```

```
print(abs(beta_hat6 - coefficients(lm6)))
```

```
           [,1]
   2.220446e-15
x6 6.661338e-16
```

Part C: Hand-drawing instructions:

Using the provided grid, plot: 1. **Data points:** (1, 4.8), (2, 6.2), (3, 8.1), (4, 9.9), (5, 11.5), (6, 13.8), (7, 15.2), (8, 17.1), (9, 18.9), (10, 21.3)

2. **True regression line (dashed):** $y = 3.5 + 1.8x$
   - At x=0: y = 3.5
   - At x=10: y = 3.5 + 18 = 21.5
3. **Fitted regression line (solid):** $y = \hat{\beta}_0 + \hat{\beta}_1 x$

```r
print(sprintf("Fitted line equation: y = %.3f + %.3f*x", beta_hat6[1], beta_hat6[2]))
```

```
[1] "Fitted line equation: y = 2.660 + 1.822*x"
```

```r
print(sprintf("At x=0: y = %.3f", beta_hat6[1]))
```

```
[1] "At x=0: y = 2.660"
```

```r
print(sprintf("At x=10: y = %.3f", beta_hat6[1] + beta_hat6[2]*10))
```

```
[1] "At x=10: y = 20.878"
```

Part D: Commentary on fit:

```r
# Calculate R-squared for assessment
y_pred6 <- X6 %*% beta_hat6
SSE6 <- sum((y6 - y_pred6)^2)
SST6 <- sum((y6 - mean(y6))^2)
R_squared <- 1 - SSE6/SST6

print(sprintf("R-squared: %.4f", R_squared))
```

```
[1] "R-squared: 0.9982"
```

```r
print(sprintf("Root Mean Square Error: %.4f", sqrt(SSE6/length(y6))))
```

```
[1] "Root Mean Square Error: 0.2229"
```

The fitted line should closely approximate the true relationship, demonstrating that least squares estimation is highly effective when the linear model assumptions are met. The high R^2 value indicates that the linear model explains most of the variation in the data, confirming the effectiveness of the least squares method for this dataset.