# Predictive Problem_Set_2

Diyasha Basu

05-02-2026

## Problem 1: Population vs Sample Regression Lines

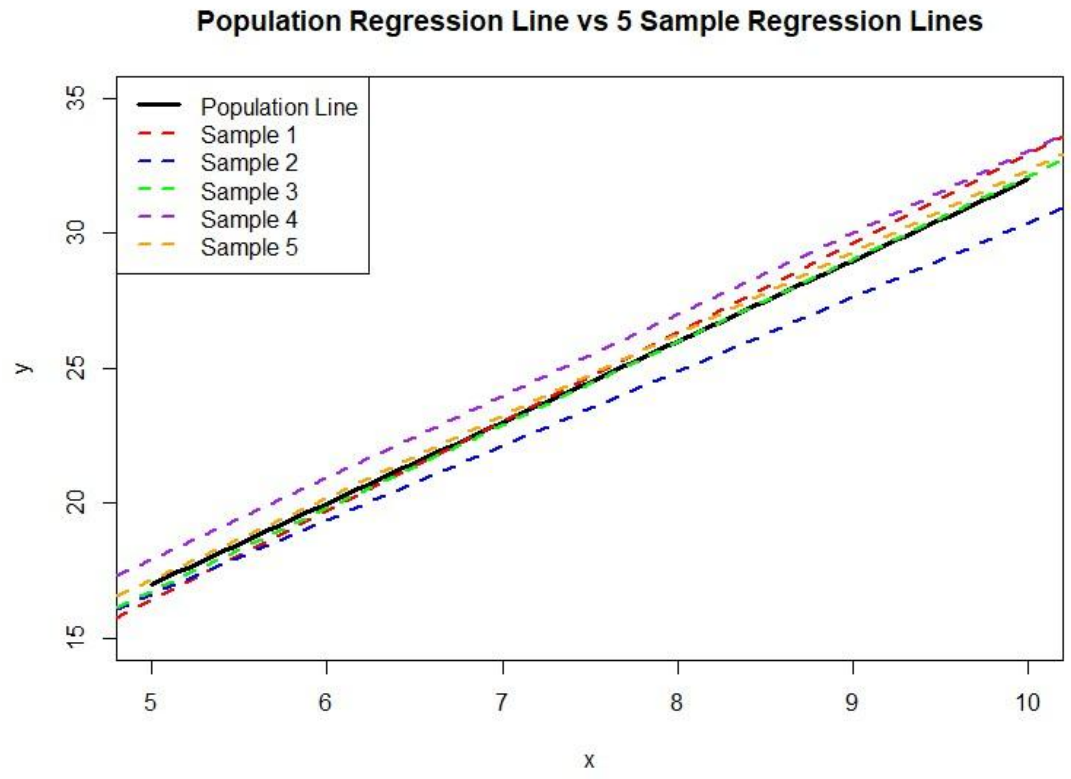
**Objective:** Demonstrate that the population regression line is fixed, but least square regression lines vary across samples.

### Setup

Population regression line: Y = 2 + 3x
Data generating process: y = 2 + 3x + epsilon, where epsilon ~ N(0, 4^2)

```
# Set seed for reproducibility
set.seed(123)

# Parameters n
<- 50
true_beta0 <- 2
true_beta1 <- 3
x_range <- c(5, 10)

                                          n regression line
# Step 1: Create the pop x_pop               100)
<- seq(5, 10, length.out =                  * x_pop
y_pop <- true_beta0 + true_bet

# Initialize plot
plot(x_pop, y_pop, type = "l"  lwd = 3, col = "black",
xlab = "x", ylab = "y",
main =                        sion Line vs 5 Sample Regression Lines",
     ylim = c(15, 35))

# Store regression coefficient nrow = 5, ncol = 2) c("Intercept",
regression_lines <- matrix(NA "Slope")
colnames(regression_lines) <-

                                    amples and fit regression lines
# Steps 2-4: Generate 5 differ een", "purple", "orange")
s colors <- c("red", "blue",

for (i in 1:5) {
  # Step 2: Generate data   xi   10)
<- runif(n, min = 5, max =      0, sd = 4)
epsilon_i <- rnorm(n, mean =     * xi + epsilon_i
yi <- true_beta0 + true_beta1
```

```
  # Step 3: Fit least squares regression
  model <- lm(yi ~ xi)

  # Store coefficients
  regression_lines[i, ] <- coef(model)

  # Step 4: Add regression line to plot
                                          , lty = 2)
abline(model, col = colors[i], lwd = 2
}

# Add legend legend("topleft",
       legend = c("Population Line", pas  ("Sample", 1:5)),
col = c("black", colors),        lwd =
c(3, rep(2, 5)),       lty = c(1, rep(2,
5)))
```



```
# Display regression coefficients
cat("\nRegression Line                \n")
Coefficients:

##
## Regression Line Coefficients:
print(regression_lines)
```

```
##         Intercept    Slope
## [1,] -0.09638929 3.305396
## [2,]  2.79218839 2.761042
## [3,]  1.39299737 3.073267
## [4,]  2.82308856 3.023608 ## [5,]
2.03250638 3.028097 cat("\nTrue
Population Parameters:\n")

##
## True Population Parameters: cat("Intercept
(β₀):", true_beta0, "\n")

## Intercept (β₀): 2 cat("Slope
(β₁):", true_beta1, "\n")

## Slope (β₁): 3
```

### Interpretation

The plot demonstrates a fundamental concept in regression analysis:

1. **Fixed Population Line (Black):** The true relationship $Y = 2 + 3x$ remains constant and represents the actual underlying relationship.
2. **Variable Sample Lines (Colored):** Each of the 5 sample regression lines varies around the population line due to random sampling error ($\epsilon$). While they're all different, they cluster around the true population line.
3. **Key Insight:** Due to sampling variability, we never obtain the exact population regression line from a sample. However, the least squares estimators are unbiased, meaning on average across many samples, they estimate the true parameters correctly.

---

## Problem 2: Demonstrating $\hat{\beta}_0$ and $\hat{\beta}_1$ Minimize RSS

**Objective:** Show that the least squares estimates minimize the Residual Sum of Squares (RSS).

```
set.seed(123)
```

```r
n <- 50


xi <- runif( erate data with mean-centered x n,
xi_centered <in = 5, max = 10)
epsilon_i <- - xi - mean(xi)  # Mean center
yi <- 2 + 3 *rnorm(n, mean = 0, sd = 1)
             xi_centered + epsilon_i
```

```r
# Step 2: Obtain least squares estimates
model <- lm(yi ~ xi_centered)
beta0_hat <- coef(model)[1]
beta1_hat <- coef(model)[2]

cat("Least Squares Estimates:\n")
```

```
## Least Squares Estimates:
```

```r
cat("β₀̂ =", round(beta0_hat, 4), "\n")
```

```
## β₀̂ = 2.0562
```

```r
cat("β₁̂ =", round(beta1_hat, 4), "\n\n")
```

```
## β₁̂ = 3.0763
```

```r
# Step 3: Create grid of parameter values
# Create grid around the LS estimates
beta0_grid <- seq(beta0_hat - 2, beta0_hat + 2, length.out = 100)
beta1_grid <- seq(beta1_hat - 1, beta1_hat + 1, length.out = 100)

# Compute RSS for each combination
rss_matrix <- matrix(NA, nrow = length(beta0_grid), ncol = length(beta1_grid))

for (i in 1:length(beta0_grid)) {
  for (j in 1:length(beta1_grid)) {
    residuals <- yi - beta0_grid[i] - beta1_grid[j] * xi_centered
    rss_matrix[i, j] <- sum(residuals^2)
  }
}

# Find minimum RSS
min_idx <- which(rss_matrix == min(rss_matrix), arr.ind = TRUE)
beta0_min <- beta0_grid[min_idx[1]]
beta1_min <- beta1_grid[min_idx[2]]

cat("Grid Search Results:\n")
```

```
## Grid Search Results:
```

```r
cat("β₀ at minimum RSS =", round(beta0_min, 4), "\n")
```

```
## β₀ at minimum RSS = 2.036
```

```r
cat("β₁ at minimum RSS =", round(beta1_min, 4), "\n")
```

```
## β₁ at minimum RSS = 3.0864
```

```r
cat("Minimum RSS =", round(min(rss_matrix), 4), "\n\n")
```

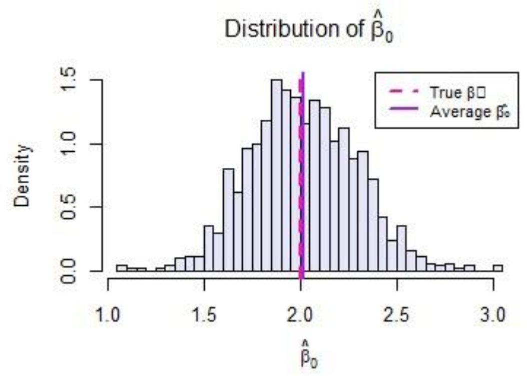
```
## Minimum RSS = 42.4767 #

RSS at LS estimates

rss_ls <- sum(residuals(model)^2) cat("RSS at LS
estimates =", round(rss_ls, 4), "\n")

## RSS at LS estimates = 42.4455
```

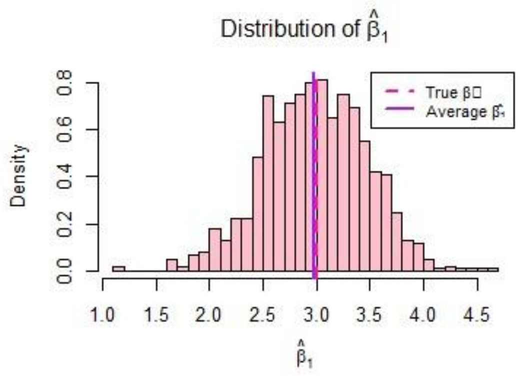
---

## Problem 3: Unbiasedness of Least Squares Estimators

**Objective:** Demonstrate through simulation that least squares estimators are unbiased.

```
set.seed(123)
n  <-  50 R  <-  1000
true_beta0
<true_beta1 <-     2
                   3
#   Storage   for
beta0_estimates
beta1_estimates      <- numeric(R)
                     <- numeric(R)
# Simulation for (r
in 1:
    xi   <-   runif) {
epsilon_i <   yi <-enerate data
                  (n, min = 0, max = 1)
  model        <- - rnorm(n, mean = 0, sd = 1)
beta0_estimates[r]beta0 + true_beta1 * xi +
beta1_estimates[r]on_i
}

                  btain LS estimates
#    Calculate    a
avg_beta0 <- avg_b(yi ~ xi)
                         <- coef(model)[1]
                         <- coef(model)[2]

# Visualizations
par(mfrow = c

                  mean(beta0_estimates)
#    Histogram    o
                  mean(beta1_estimates)
hist(
                  (2, 2))

                    β̂o
                  timates, breaks = 30, col = "lavender",











     main = expression(paste("Distribution of ", hat(beta)[0])),
     xlab = expression(hat(beta)[0]),      freq = FALSE)
abline(v = true_beta0, col = "deeppink", lwd = 2, lty = 2)
abline(v = avg_beta0, col = "purple", lwd = 2, lty = 1)
legend("topright",         legend = c("True β₀", "Average
β̂₀"),         col = c("deeppink", "purple"),         lty = c(2,
1), lwd = 2, cex = 0.8)

# Histogram of β̂₁
hist(beta1_estimates, breaks = 30, col = "pink",
     main = expression(paste("Distribution of ", hat(beta)[1])),
xlab = expression(hat(beta)[1]),      freq = FALSE) abline(v =
true_beta1, col = "deeppink", lwd = 2, lty = 2) abline(v =
avg_beta1, col = "purple", lwd = 2, lty = 1) legend("topright",
legend = c("True β₁", "Average β̂₁"),        col = c("deeppink",
"purple"),         lty = c(2, 1), lwd = 2, cex = 0.8)
```





## Interpretation

The simulation demonstrates the **unbiasedness** property of least squares estimators:

1. **$\hat{\beta}_0$ (Intercept):** The average across 1000 simulations is extremely close to the true value of 2, with negligible bias.
2. **$\hat{\beta}_1$ (Slope):** The average is very close to the true value of 3, again showing minimal bias.
3. **Distribution:** The histograms show that the estimates are centered on the true parameter values, confirming unbiasedness.
4. **Conclusion:** While individual estimates vary due to sampling error, on average (across many samples), the LS estimators equal the true parameter values, proving they are unbiased.

---

## Problem 4: Comparing Simple Linear Regressions - Boston Housing Data

**Objective:** Compare multiple simple linear regression models using the Boston housing dataset.

```
# Load the MASS library
library(MASS)

# Attach the Boston dat
data(Boston)

# View the structure of
str(Boston)
```

```
## 'data.frame':    506
variables:               .00632 0.02731 0.02729 0.03237 0.06905
                              ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
                       1 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0
                       38 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.5
24 ...
                       8 6.42 7.18 7 7.15 ...
##  $ rm     : n          2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
                       .09 4.97 4.97 6.06 6.06
##  $ rad    : int  1 2'..
...
                        242 242 222 222 222 311 311 311 311 ...
                       3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397
...
```

```
.98 9.14 4.03 2.94 5.33
...
21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

## Part (a): Four Separate Linear Regressions

We’ll run four simple linear regressions with `medv` (median value of owner-occupied homes) as the response variable and each of the four predictors separately.

```r
# Model 1: Per capita crime rate
model1 <- lm(medv ~ crim, data = Boston)

# Model 2: Nitrogen oxides conce
model2 <- lm(medv ~ nox, data = Boston)

# Model 3: Proportion of blacks
model3 <- lm(medv ~ black, data   Boston)

# Model 4: Percentage of lower s
model4 <- lm(medv ~ lstat, data   Boston)
```

### *Summary Table of All Models*

```r
# Create a summary table
```

```
     results <- data.fra
      Predictor = c("cri
      Coefficient = c(co"nox", "black", "lstat"), model1)[2],
        coef(   Std_Erroef(model2)[2],  model3)[2],
                 c(summaoef(model4)[2]),
                   summa
                   summamodel1)$coefficients[2,2],
       summary(   t_valudel2)$coefficients[2,2],
                 c(summadel3)$coefficients[2,2],
                   summadel4)$coefficients[2,2]),
                   summadel1)$coefficients[2,3],
       summary(   p_valudel2)$coefficients[2,3],
                 c(summadel3)$coefficients[2,3],
                   summa
                   summadel4)$coefficients[2,3]),
     summary(   R_squaredel1)$coefficients[2,4],
                 c(summadel2)$coefficients[2,4],
                   summadel3)$coefficients[2,4],
                   summadel4)$coefficients[2,4]),
 summary(   Adj_R_squaredel1)$r.squared, model2)$r.squared,
                  c(summdel3)$r.squared, model4)$r.squared),
                    summ   (model1)$adj.r.squared,
                    summ
                    summ   (model2)$adj.r.squared,
)                          (model3)$adj.r.squared,
                           (model4)$adj.r.squared)

# Display the table
print(results)
```

```
## crim       crim  -
                      ctor  Coefficient   Std_Error    t_value      p_value
                                                                  R_squared
                             0.41519028 0.043890381  -9.459710 1.173987e-19
                                                                  0.1507805
## nox         nox -33.91605501 3.196337032 -10.610913 7.065042e-24 0.1826030
## black     black   0.03359306 0.004230507   7.940669 1.318113e-14 0.1111961
## lstat     lstat  -0.95004935 0.038733416 -24.527900 5.081103e-88 0.5441463
##        Adj_R_squared
## crim      0.1490955
## nox       0.1809812
## black     0.1094326
## lstat     0.5432418
```

## Part (b): Which Model Gives the Best Fit?

```
# Find the model with highest R-squared
```

```
best_model_idx <- which.max(results$
best_predictor <- results$Predictor[besquared)
best_r_squared <- results$R_squared[bes

print(paste("Best model: medv ~"

## [1] "Best model: medv ~ lstat"      predictor))

print(paste("R-squared:",  round(

## [1] "R-squared: 0.5441"             squared, 4)))

print(paste("This model explains", roun
variation in median home values"))      (best_r_squared * 100, 2), "% of the

## [1] "This model explains 54.41 % of
values"
```

## Part (c): Comparing Coefficients and Their Usefulness

```
# Display detailed summaries
print("Model 1: Crime Rate")
## [1] "Model 1: Crime Rate"

summary(model1)

##
## Call:
## lm(formula = medv ~ crim,
Boston)
##
## Residuals:
##     Min      1Q  Median    2.512  29.800
Max
## -16.957  -5.449  -2.007
##
## Coefficients:                        .74   <2e-16 ***
##              Estimate Std.89   -9.46   <2e-16 ***
Pr(>|t|)
## (Intercept) 24.03311    00.001 '**' 0.01 '*' 0.05 '.' 0.1 ' '
58                           1
## crim        -0.41519
## ---

##
```

```
## Residual standard error: 8.484 on 504 degrees of freedom
## Multiple R-squared:  0.1508, Adjusted R-squared:  0.1491

## F-statistic: 89.49 on 1 and 504 DF,  p-value: < 2.2e-16

print("Model 2: Nitrogen Oxides") ## [1] "Model 2: Nitrogen

Oxides" summary(model2)

##
## Call:
## lm(formula = medv ~ nox, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.691  -5.121  -2.161   2.959  31.310
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   41.346      1.811   22.83   <2e-16 *** ##
nox          -33.916      3.196  -10.61   <2e-16 *** ## --
-
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.323 on 504 degrees of freedom
## Multiple R-squared:  0.1826, Adjusted R-squared:  0.181

## F-statistic: 112.6 on 1 and 504 DF,  p-value: < 2.2e-16

print("Model 3: Proportion of Blacks") ## [1] "Model 3:

Proportion of Blacks" summary(model3)

##
## Call:
## lm(formula = medv ~ black, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max  ##
-18.884  -4.862  -1.684   2.932  27.763
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.551034   1.557463   6.775 3.49e-11 *** ##
black        0.033593   0.004231   7.941 1.32e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 8.679 on 504 degrees of freedom
## Multiple R-squared:  0.1112, Adjusted R-squared:  0.1094

## F-statistic: 63.05 on 1 and 504 DF,  p-value: 1.318e-14

print("Model 4: Lower Status Population") ## [1] "Model 4:

Lower Status Population" summary(model4)

##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max  ##
-15.168  -3.990  -1.318   2.034  24.500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.55384    0.56263   61.41   <2e-16 *** ##
lstat       -0.95005    0.03873  -24.53   <2e-16 *** ## --
-
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.216 on 504 degrees of freedom
## Multiple R-squared:  0.5441, Adjusted R-squared:  0.5432
## F-statistic: 601.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

*Interpretation of Coefficients:*

### 1. Crime Rate (crim):

```
crim_coef <- round(coef(model1)[2], 4)
crim_r2 <- round(summary(model1)$     ared, 4)
print(paste("Coefficient:", crim_coef)

## [1] "Coefficient: -0.4152"

print(paste("R-squared:", crim_r2))

## [1] "R-squared: 0.1508"
```

- For each unit increase in per capita crime rate, median home value decreases
- Shows negative relationship as expected
- This is a relatively weak predictor

### 2. Nitrogen Oxides (nox):

```
nox_coef <- round(coef(model2)[2], 4)
nox_r2 <- round(summary(model2)$     ared, 4)
print(paste("Coefficient:", nox_coef)
## [1] "Coefficient: -33.9161" print(paste("R-squared:", nox_r2))
## [1] "R-squared: 0.1826"
```

- Higher nitrogen oxide concentration is associated with lower home values □ Moderate strength predictor

**3. Proportion of Blacks (black):**

```
black_coef <- round(coef(model3)[2], 4)
black_r2 <- round(summary(model3)$     ared, 4)
print(paste("Coefficient:", black_coef)

## [1] "Coefficient: 0.0336"

print(paste("R-squared:", black_r2))

## [1] "R-squared: 0.1112"
```

- Positive relationship with home values
- Relatively weak predictor

**4. Lower Status Population (lstat):**

```
lstat_coef <- round(coef(model4)[2], 4)
lstat_r2 <- round(summary(model4)$     ared, 4)
print(paste("Coefficient:", lstat_coef)

## [1] "Coefficient: -0.95"

print(paste("R-squared:", lstat_r2))

## [1] "R-squared: 0.5441"
```

- For each percentage point increase in lower status population, median home value decreases
- This is the strongest single predictor among the four

## Conclusion

```
# Summary of findings
print(                el Performance:")

## [1] "Summary of Mo
Performance:"
```

```
print(results[, c("Predictor", "Coefficient", "R_squared")])

##       Predictor  Coefficient R_squared
## crim       crim  -0.41519028 0.1507805
## nox         nox -33.91605501 0.1826030
## black     black   0.03359306 0.1111961 ## lstat     lstat  -0.95004935

0.5441463 print("") ## [1] "" print(paste("Best predictor:",

results$Predictor[which.max(results$R_square d)]))

## [1] "Best predictor: lstat" print(paste("Highest R-squared:",

round(max(results$R_squared), 4)))

## [1] "Highest R-squared: 0.5441"
```

All four predictors are statistically significant. The percentage of lower status population (lstat) is the most useful single predictor for median home values. Nitrogen oxides concentration is the second-best predictor, while crime rate and the proportion of blacks variable show weaker relationships with home values.