HW3

2025-02-07

Setup

Suppose $X_1, X_2, ..., X_n \sim N(\mu_x, \sigma^2)$ and $Y_1, Y_2, ..., Y_n \sim N(\mu_y, \sigma^2)$ are independent random samples. We want to test the following hypothesis

$$H_0 : \mu_x = \mu_y \quad \text{vs.} \quad H_1 : \mu_x \neq \mu_y$$

Using the standard notations, the pooled t-statistic is given by

$$T = \frac{(\bar{X} - \bar{Y}) - (\mu_x - \mu_y)}{S_p \sqrt{\left(\frac{1}{n} + \frac{1}{m}\right)}},$$

where $S_p^2 = \frac{(n-1)S_x^2 + (m-1)S_y^2}{n+m-2}$. Under H_0 the distribution of T is given by $T \sim t_{n+m-2}$. We reject H_0 at level α if observed value of $|T| > t_{\alpha/2, n+m-2}$, where $t_{\alpha/2, n+m-2}$ is the upper $\alpha/2$ point of the t-distribution with $n + m - 2$ degrees of freedom.

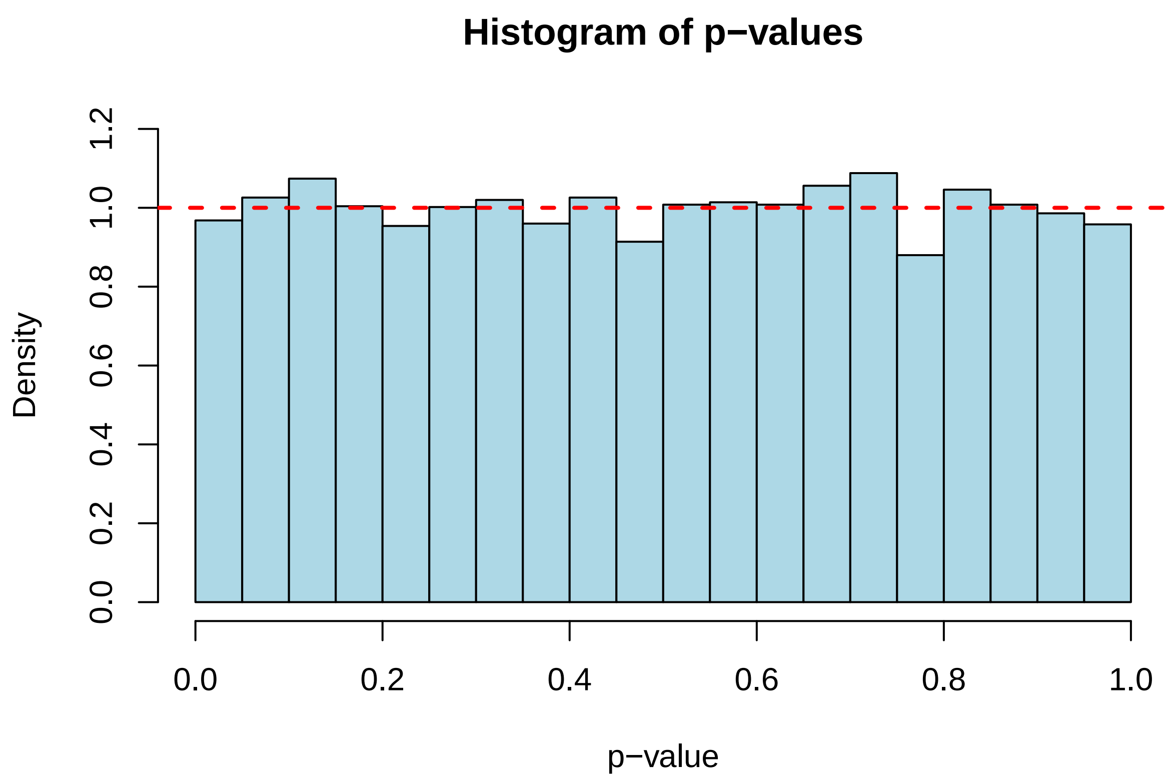
Question

Take suitable values of sample sizes and other parameters. Repeatedly generate data from the null distribution, and show that the test properly maintains the Type I error, i.e., equivalently show that the distribution of the p-value is Uniform(0,1).

Write your code in R. You can utilize a built-in function, such as t.test, to calculate the p-values. However, it would be beneficial to practice by writing your own code based on the definition of the test statistic.

```r
# Set seed for reproducibility
set.seed(42)

# Parameters
n <- 30  # Sample size for X
m <- 30  # Sample size for Y
mu_x <- 0  # Mean under H0
mu_y <- 0  # Same as mu_x under H0
sigma <- 1 # Standard deviation
num_simulations <- 10000  # Number of simulations

# Store p-values
p_values <- numeric(num_simulations)

for (i in 1:num_simulations) {
```

```r
  # Generate data under H0
  X <- rnorm(n, mean = mu_x, sd = sigma)
  Y <- rnorm(m, mean = mu_y, sd = sigma)

  # Compute sample statistics
  x_bar <- mean(X)
  y_bar <- mean(Y)
  s_x2 <- var(X)
  s_y2 <- var(Y)

  # Compute pooled variance
  sp2 <- ((n-1)*s_x2 + (m-1)*s_y2) / (n + m - 2)
  sp <- sqrt(sp2)

  # Compute t-statistic
  T_stat <- (x_bar - y_bar) / (sp * sqrt(1/n + 1/m))

  # Compute p-value from t-distribution
  df <- n + m - 2
  p_values[i] <- 2 * pt(-abs(T_stat), df)  # Two-tailed test
}

# Plot histogram of p-values
hist(p_values, breaks = 30, probability = TRUE, col = "lightblue",
     main = "Histogram of p-values",
     xlab = "p-value", ylim = c(0, 1.2))
abline(h = 1, col = "red", lwd = 2, lty = 2) # Expected uniform density
```



Setup

Consider a simple linear regression (SLR) model:

$$Y = \beta_0 + \beta_1 x + \epsilon$$

All notations carry their usual meaning.

Question

Write R codes for the following questions:

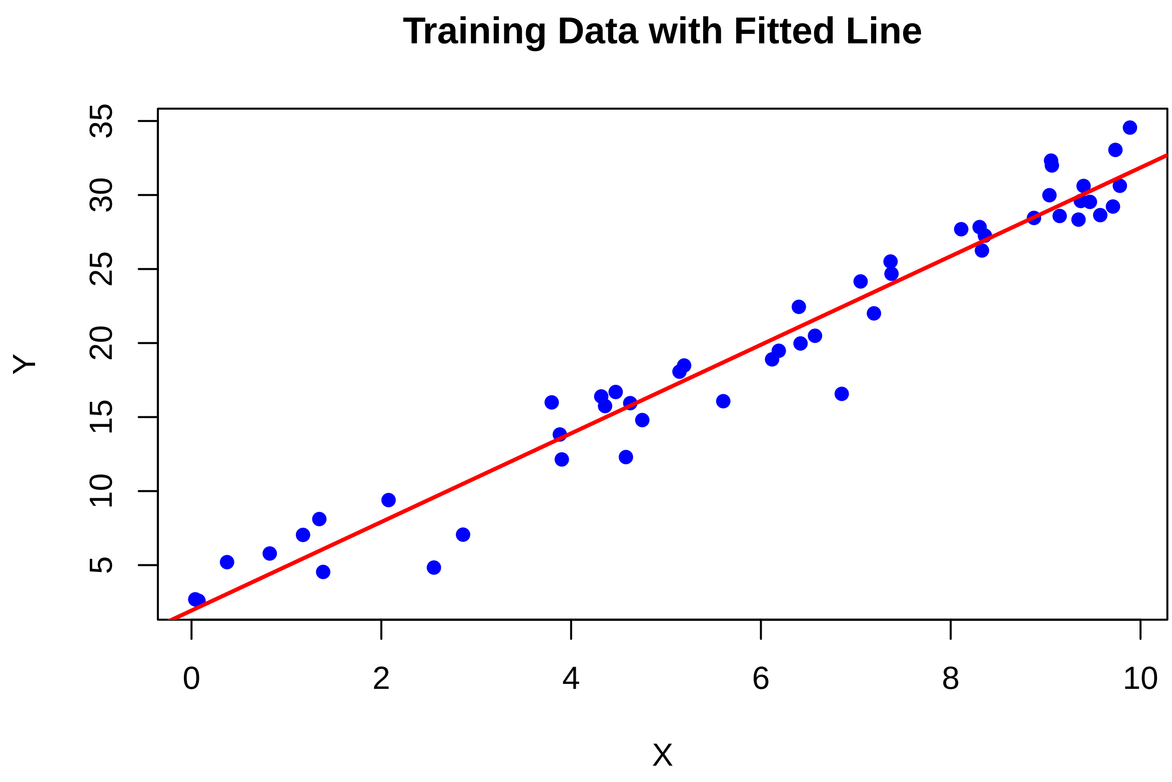
1. Generate training data, fit the SLR model, and calculate training RMSE. Take suitable values for the sample size and parameters.

2. Draw the scatter plot and overlay the fitted line.

3. Generate test data and evaluate prediction performance. Keep in mind that, on average, the test RMSE is higher than the training RMSE, but it may vary in different sample sets.

```r
# Set seed for reproducibility
set.seed(42)

# 1. Generate Training Data
```

```r
n_train <- 50  # Sample size for training data
beta_0 <- 2    # Intercept
beta_1 <- 3    # Slope
sigma <- 2     # Noise standard deviation

x_train <- runif(n_train, min = 0, max = 10)  # Generate x values
epsilon_train <- rnorm(n_train, mean = 0, sd = sigma)  # Noise
y_train <- beta_0 + beta_1 * x_train + epsilon_train  # Generate y values

# Fit Simple Linear Regression Model
slr_model <- lm(y_train ~ x_train)

# Training RMSE Calculation
y_train_pred <- predict(slr_model)
train_rmse <- sqrt(mean((y_train - y_train_pred)^2))

# 2. Scatter Plot with Fitted Line
plot(x_train, y_train, main = "Training Data with Fitted Line",
     xlab = "X", ylab = "Y", pch = 16, col = "blue")
abline(slr_model, col = "red", lwd = 2)  # Overlay fitted line
```




```r
# 3. Generate Test Data
n_test <- 30  # Sample size for test data
x_test <- runif(n_test, min = 0, max = 10)  # Generate new x values
epsilon_test <- rnorm(n_test, mean = 0, sd = sigma)  # Noise
```

```r
y_test <- beta_0 + beta_1 * x_test + epsilon_test  # Generate test y values

# Predict on Test Data
y_test_pred <- predict(slr_model, newdata = data.frame(x = x_test))
```

```
## Warning: 'newdata' had 30 rows but variables found have 50 rows
```

```r
# Test RMSE Calculation
test_rmse <- sqrt(mean((y_test - y_test_pred)^2))
```

```
## Warning in y_test - y_test_pred: longer object length is not a multiple of
## shorter object length
```

```r
# Display RMSE values
cat("Training RMSE:", round(train_rmse, 3), "\n")
```

```
## Training RMSE: 1.922
```

```r
cat("Test RMSE:", round(test_rmse, 3), "\n")
```

```
## Test RMSE: 12.604
```