Week 6 - Midterm Assignment: Simulation Project

STAT 420, Summer 2023, Adriane Yi - adriane3

2023-06-26

Contents

Simulation Study 1: Significance of Regression

Introduction

In this simulation study we will investigate the significance of regression test. We will simulate from two different models:

1. The **"significant"** model

$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \epsilon_i$$

where $\epsilon_i \sim N(0, \sigma^2)$ and

- $\beta_0 = 3$,
- $\beta_1 = 1$,
- $\beta_2 = 1$,
- $\beta_3 = 1$.

2. The **"non-significant"** model

$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \epsilon_i$$

where $\epsilon_i \sim N(0, \sigma^2)$ and

- $\beta_0 = 3$,
- $\beta_1 = 0$,
- $\beta_2 = 0$,
- $\beta_3 = 0$.

For both, we consider a sample size of 25 and three possible levels of noise. That is, three values of σ.

- $n = 25$
- $\sigma \in (1, 5, 10)$

We used simulation to obtain an empirical distribution for each of the following values, for each of the three values of σ, for both models.

- The F **statistic** for the significance of regression test.
- The **p-value** for the significance of regression test
- R^2

For each model and σ combination, we used 2000 simulations. For each simulation, we fit a regression model of the same form used to perform the simulation.

We used dataset from an external file in `study_1.csv` for the values of the predictors. These were kept constant for the entirety of this study

Methods

Seed set up for reproducibility

```
birthday = 19870503
set.seed(birthday)
```

Read in from external file to study_1

```r
study_1 = read.csv('study_1.csv')
#str(study_1)
```

Variable setup

```r
n = 25                          # sample size
num_sim = 2000                  # number of simulation
beta_0_sig = 3
beta_1_sig = 1
beta_2_sig = 1
beta_3_sig = 1

beta_0_insig = 3
beta_1_insig = 0
beta_2_insig = 0
beta_3_insig = 0

sigma = c(1,5,10)
```

Simulation

```r
simulate = function(data, sigma, beta_0, beta_1, beta_2, beta_3) {
  y = data$y
  x1 = data$x1
  x2 = data$x2
  x3 = data$x3
  eps = rnorm(length(y), mean = 0, sd = sigma)
  # calculate y
  y = beta_0 + beta_1 * x1 + beta_2 * x2 + beta_3 * x3 + eps

  # Fit the regression model
  model = lm(y ~ x1 + x2 + x3)

  f_stat = summary(model)$fstatistic[1]  # Get the F statistic
  p_val = summary(model)$coefficient[4,4]  # Get the p-value
  r_squared = summary(model)$r.squared  # Get the R-squared value
  df = cbind(F_Stat = f_stat, P_Value = p_val, R_Squared = r_squared)
  return(df)
}

# Create empty vectors to store the results
f_stat_sig = matrix(0, nrow = num_sim, ncol = length(sigma))
f_stat_insig = matrix(0, nrow = num_sim, ncol = length(sigma))
p_values_sig = matrix(0, nrow = num_sim, ncol = length(sigma))
p_values_insig = matrix(0, nrow = num_sim, ncol = length(sigma))
r_squared_sig = matrix(0, nrow = num_sim, ncol = length(sigma))
r_squared_insig = matrix(0, nrow = num_sim, ncol = length(sigma))

# output matrix setup
```

```
result =  data.frame(matrix(0, nrow = length(sigma)*2, ncol = 3))
rownames(result) = c(paste("Significant Model_", sigma[1]),
                     paste("Significant Model_", sigma[2]),
                     paste("Significant Model_", sigma[3]),
                     paste("Non-Significant Model_", sigma[1]),
                     paste("Non-Significant Model_", sigma[2]),
                     paste("Non-Significant Model_", sigma[3])
                     )
colnames(result) = c("F-statistics", "P-values", "R-sqaured")

# iterate
for(j in 1:length(sigma)){
  for(i in 1:num_sim) {
    #simulate and fit regression models
    sig_model_result = simulate(study_1, sigma[j], beta_0_sig, beta_1_sig, beta_2_sig, beta_3_sig)

    insig_model_result = simulate(study_1, sigma[j], beta_0_insig, beta_1_insig, beta_2_insig, beta_3_i

    # Store the results in the list
    f_stat_sig[i,j] = sig_model_result[1]
    p_values_sig[i,j] = sig_model_result[2]
    r_squared_sig[i,j] = sig_model_result[3]
    f_stat_insig[i,j] = insig_model_result[1]
    p_values_insig[i,j] = insig_model_result[2]
    r_squared_insig[i,j] = insig_model_result[3]
  }
}
```

Results

The non-significant model is the model for Null hypothesis H_0.
The significant model is the model for Alternative hypothesis H_1.
Results show these in plot:

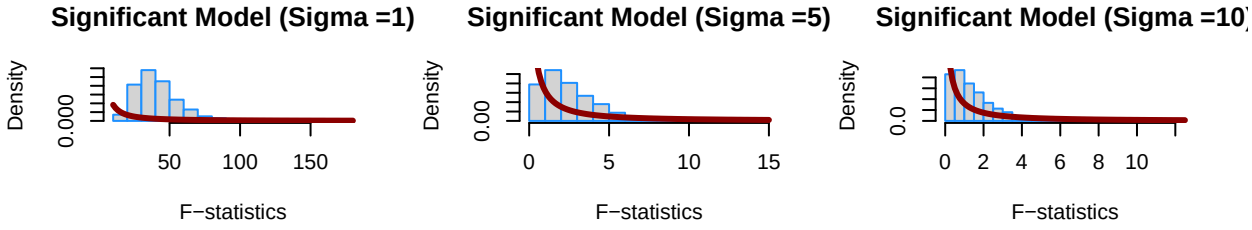
- The F **statistic** for the significance of regression test.
- The **p-value** for the significance of regression test
- R^2

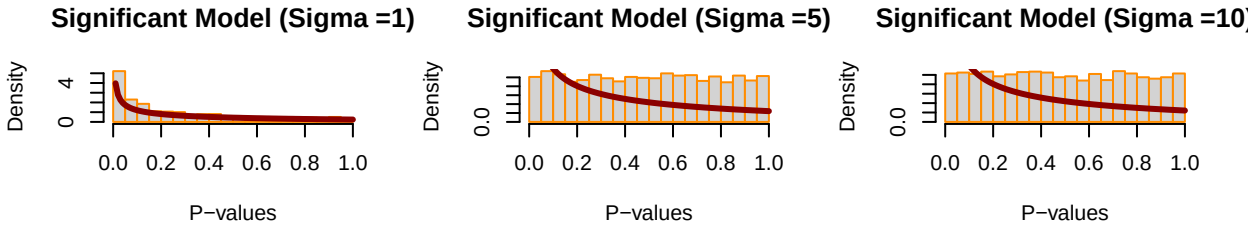
Plotting

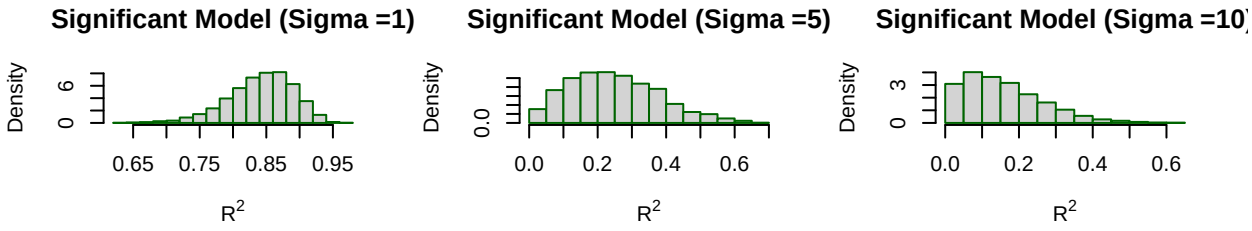
Plots are organized in a grid for easy comparison between significant and non-significant models and F-statistic, P-value, and R^2 values per sigma value changes.

```
#### Significant model plot empirical distributions for F-statistics, P-values, and R-squared
par(mfrow = c(3, 3))
for(j in 1:length(sigma)){
  hist(f_stat_sig[, j], prob = TRUE, breaks = 20,
       main = paste0("Significant Model (Sigma =", sigma[j],")"),
       xlab = "F-statistics",
       border = "dodgerblue"
  )
```

```
  x = f_stat_sig[, j]
  curve(df(x, df1 = 1, df2 = 1), col = "darkred", add = TRUE,
        lwd = 3)
}
for(j in 1:length(sigma)){
  hist(p_values_sig[, j], prob = TRUE, breaks = 20,
      main = paste0("Significant Model (Sigma =", sigma[j],")"),
      xlab = "P-values",
      border = "darkorange"
  )
  x = p_values_sig[, j]
  curve(df(x, df1 = 1, df2 = num_sim), col = "darkred", add = TRUE, lwd = 3)
}
for(j in 1:length(sigma)){
  hist(r_squared_sig[, j], prob = TRUE, breaks = 20,
       main = paste0("Significant Model (Sigma =", sigma[j],")"),
       xlab = expression(R^2),
       border = "darkgreen"
  )
}
```

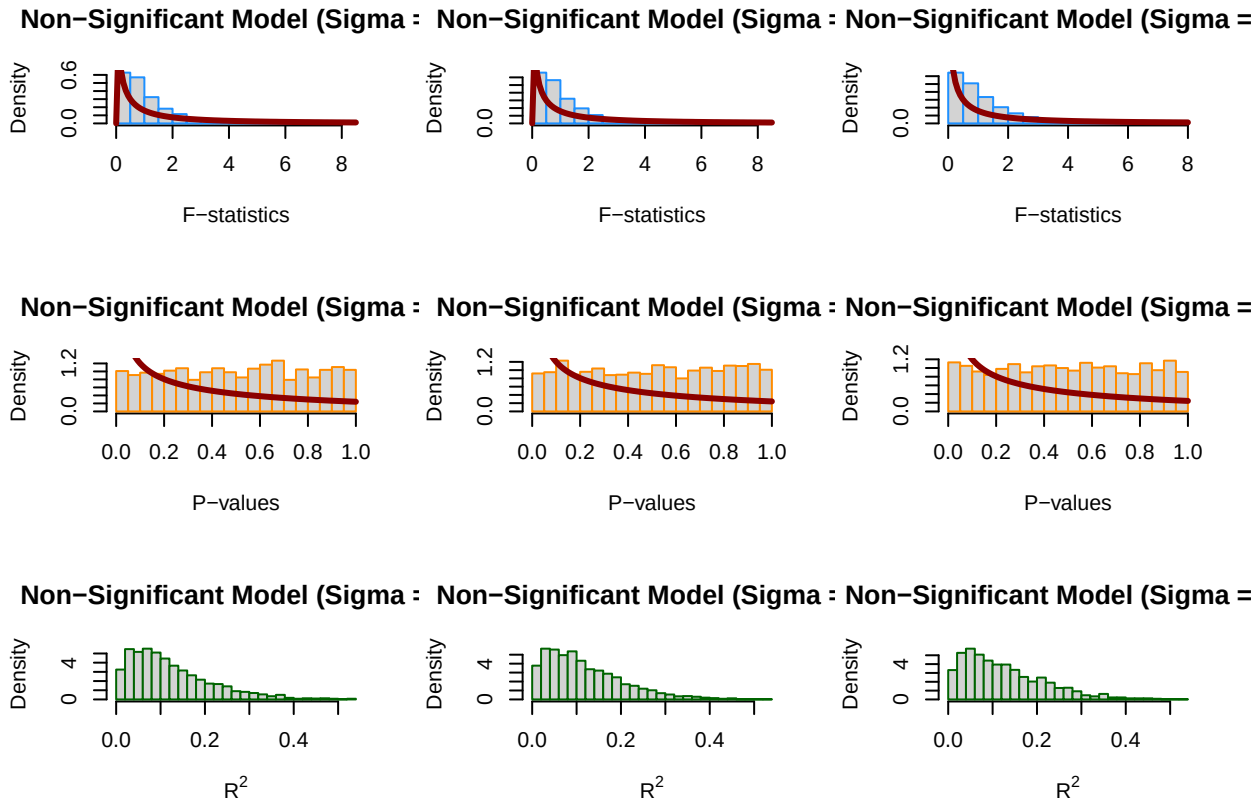







```
#### Non-significant model Plot
for(j in 1:length(sigma)){
  hist(f_stat_insig[, j], prob = TRUE, breaks = 20,
       main = paste0("Non-Significant Model (Sigma =", sigma[j],")"),
       xlab = "F-statistics",
       border = "dodgerblue"
  )
```

```r
  x = f_stat_insig[, j]
  curve(df(x, df1 = 1, df2 = 1), col = "darkred", add = TRUE,
        lwd = 3)

}
for(j in 1:length(sigma)){
  hist(p_values_insig[, j], prob = TRUE, breaks = 20,
       main = paste0("Non-Significant Model (Sigma =", sigma[j],")"),
       xlab = "P-values",
       border = "darkorange"
  )
  x = p_values_insig[, j]
  curve(df(x, df1 = 1, df2 = num_sim), col = "darkred", add = TRUE, lwd = 3)
}
for(j in 1:length(sigma)){
  hist(r_squared_insig[, j], prob = TRUE, breaks = 20,
       main = paste0("Non-Significant Model (Sigma =", sigma[j],")"),
       xlab = expression(R^2),
       border = "darkgreen"
  )
}
```




```r
par(mfrow = c(1, 1))
```

Discussion

Comparison between significant and non-significant models

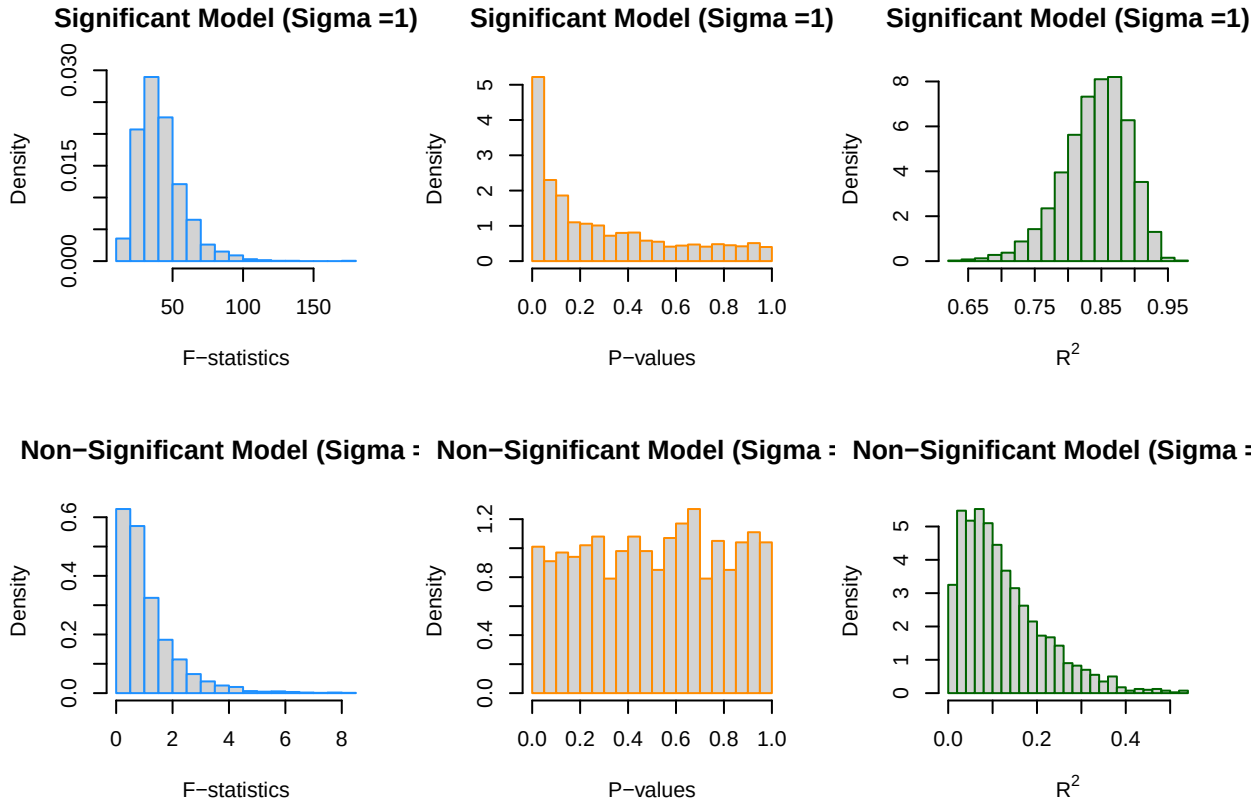
In this simulation study, we have generated empirical distributions through simulations for 3 sigma values for both Siginificant model and Non-Significant model. From the plots of the significant model, we see that F-statistics is significantly higher when sigma = 1 compared to sigma = 5 or sigma = 10(higher noise level). This suggests the significance of the model when the noise level is lower(sigma = 1) As F-statistic is a measure of the overall significance of the regression model, higher F-statstic indicates that the proportion of SSR, more portion of total variation is explained by the model.

Comparing R^2 distributions obtained, for different sigma levels, we see that the R^2 values are significantly higher when sigma = 1 compared to sigma = 5 or sigma = 10. This also suggests that the best fit of the model at sigma = 1.

If we focus on the Significant model and Non-Significant model at sigma = 1 :

```
# Significant model plot empirical distributions for F-statistics, P-values, and R-squared
par(mfrow = c(2, 3))
hist(f_stat_sig[, 1], prob = TRUE, breaks = 20,
     main = paste0("Significant Model (Sigma =", sigma[1],")"),
     xlab = "F-statistics",
     border = "dodgerblue"
)

hist(p_values_sig[, 1], prob = TRUE, breaks = 20,
    main = paste0("Significant Model (Sigma =", sigma[1],")"),
    xlab = "P-values",
    border = "darkorange"
)

hist(r_squared_sig[, 1], prob = TRUE, breaks = 20,
     main = paste0("Significant Model (Sigma =", sigma[1],")"),
     xlab = expression(R^2),
     border = "darkgreen"
)

#### Non-significant model Plot

hist(f_stat_insig[, 1], prob = TRUE, breaks = 20,
     main = paste0("Non-Significant Model (Sigma =", sigma[1],")"),
     xlab = "F-statistics",
     border = "dodgerblue"
)



hist(p_values_insig[, 1], prob = TRUE, breaks = 20,
     main = paste0("Non-Significant Model (Sigma =", sigma[1],")"),
     xlab = "P-values",
     border = "darkorange"
)


hist(r_squared_insig[, 1], prob = TRUE, breaks = 20,
```

```
     main = paste0("Non-Significant Model (Sigma =", sigma[1],")"),
     xlab = expression(R^2),
     border = "darkgreen"
)
```




```
par(mfrow = c(1, 1))
```

We see that most of the P-values are very small in the Significant model throughout the different noise levels. R^2 values for the Significant model are > 0.70. R^2 values for Non-Significant model are very small. Higher R^2 value indicates that the model explains a larger portion of the variability and also better fit for the data. This support better fit of Significant model compared to the Non-Significant model.
Looking at the above plot between Significant model and Non-Significant model, we can see that the higher F-statistics values with lower p-values and higher R^2 value for the Significant model which suggest that the significance of the Significant model and we can reject the Non-Significant model.

Simulation Study 2: Using RMSE for Selection?

Introduction

In homework we saw how Test RMSE can be used to select the "best" model. In this simulation study we will investigate how well this procedure works. Since splitting the data is random, we don't expect it to work correctly each time. We could get unlucky. But averaged over many attempts, we should expect it to select the appropriate model.

We will simulate from the model

$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \beta_4 x_{i4} + \beta_5 x_{i5} + \beta_6 x_{i6} + \epsilon_i$$

where $\epsilon_i \sim N(0, \sigma^2)$ and

- $\beta_0 = 0$,
- $\beta_1 = 3$,
- $\beta_2 = -4$,
- $\beta_3 = 1.6$,
- $\beta_4 = -1.1$,
- $\beta_5 = 0.7$,
- $\beta_6 = 0.5$.

We will consider a sample size of 500 and three possible levels of noise. That is, three values of σ.

- $n = 500$
- $\sigma \in (1, 2, 4)$

Use the data found in `study_2.csv` for the values of the predictors. These should be kept constant for the entirety of this study. The `y` values in this data are a blank placeholder.

Each time you simulate the data, randomly split the data into train and test sets of equal sizes (250 observations for training, 250 observations for testing).

For each, fit **nine** models, with forms:

- `y ~ x1`
- `y ~ x1 + x2`
- `y ~ x1 + x2 + x3`
- `y ~ x1 + x2 + x3 + x4`
- `y ~ x1 + x2 + x3 + x4 + x5`
- `y ~ x1 + x2 + x3 + x4 + x5 + x6`, the correct form of the model as noted above
- `y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7`
- `y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8`
- `y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9`

For each model, calculate Train and Test RMSE.

$$\text{RMSE(model, data)} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

We repeat this process with 1000 simulations for each of the 3 values of σ. For each value of σ, we created a plot that shows how average Train RMSE and average Test RMSE changes as a function of model size. Also show the number of times the model of each size was chosen for each value of σ.

We have simulated the y vector $3\ddot{O}1000 = 3000$ times and have fit $9\ddot{O}3\ddot{O}1000 = 27000$ models.

Methods

Seed set up for reproducibility

```r
birthday = 19870503
set.seed(birthday)
```

Read in from external file to study_1

```r
library(readr)
study_2 = read_csv("study_2.csv")
```

```
## Rows: 500 Columns: 10
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (10): y, x1, x2, x3, x4, x5, x6, x7, x8, x9
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
#str(study_2)
```

Variable setup

```r
# Setup
beta_0 = 0
beta_1 = 3
beta_2 = -4
beta_3 = 1.6
beta_4 = -1.1
beta_5 = 0.7
beta_6 = 0.5

n = 500
sigma = c(1,2,4)
num_sim = 1000

rmse = function(actual, predicted) {
  sqrt(sum((actual - predicted)^2) / length(actual))
}

# Set up Output
rmse_train_sig1 = data.frame(matrix(0, nrow = num_sim, ncol = 9))
rmse_test_sig1 = data.frame(matrix(0, nrow = num_sim, ncol = 9))
rmse_train_sig2 = data.frame(matrix(0, nrow = num_sim, ncol = 9))
rmse_test_sig2 = data.frame(matrix(0, nrow = num_sim, ncol = 9))
rmse_train_sig3 = data.frame(matrix(0, nrow = num_sim, ncol = 9))
rmse_test_sig3 = data.frame(matrix(0, nrow = num_sim, ncol = 9))
colnames(rmse_train_sig1) = c("Model1", "Model2", "Model3", "Model4", "Model5",
                              "Model6", "Model7", "Model8", "Model9")
colnames(rmse_test_sig1) = c("Model1", "Model2", "Model3", "Model4", "Model5",
                             "Model6", "Model7", "Model8", "Model9")
```

```r
colnames(rmse_train_sig2) = c("Model1", "Model2", "Model3", "Model4", "Model5",
                              "Model6", "Model7", "Model8", "Model9")
colnames(rmse_test_sig2) = c("Model1", "Model2", "Model3", "Model4", "Model5",
                             "Model6", "Model7", "Model8", "Model9")
colnames(rmse_train_sig3) = c("Model1", "Model2", "Model3", "Model4", "Model5",
                              "Model6", "Model7", "Model8", "Model9")
colnames(rmse_test_sig3) = c("Model1", "Model2", "Model3", "Model4", "Model5",
                             "Model6", "Model7", "Model8", "Model9")

models = list(
  model_1 = y ~ x1,
  model_2 = y ~ x1 + x2,
  model_3 = y ~ x1 + x2 + x3,
  model_4 = y ~ x1 + x2 + x3 + x4,
  model_5 = y ~ x1 + x2 + x3 + x4 + x5,
  model_6 = y ~ x1 + x2 + x3 + x4 + x5 + x6,          # True model
  model_7 = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7,
  model_8 = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8,
  model_9 = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9
)
```

Simulation

```r
set.seed(birthday)
for (sig in sigma) {
  for (i in 1:num_sim) {
    epsilon = rnorm(n, 0, sig)
    # Generate error terms
    study_2$y = beta_0 + beta_1 * study_2$x1 + beta_2 * study_2$x2 + beta_3 * study_2$x3 +
      beta_4 * study_2$x4 + beta_5 * study_2$x5 + beta_6 * study_2$x6 + epsilon

    train_idx = sample(n/2)
    train = study_2[train_idx, ]
    test = study_2[-train_idx, ]

    # Calculate RMSE for train and test data on each model
    if(sig == 1){
      for (m in 1:9) {
        model = lm(models[[m]], data = train)
        rmse_train_sig1[i, m] = rmse(train$y, predict(model, train))
        rmse_test_sig1[i, m] = rmse(test$y, predict(model, test))
      }
    }else if(sig == 2){
      for (m in 1:9) {
        model = lm(models[[m]], data = train)
        rmse_train_sig2[i, m] = rmse(train$y, predict(model, train))
        rmse_test_sig2[i, m] = rmse(test$y, predict(model, test))
      }
    }else{
      for (m in 1:9) {
        model = lm(models[[m]], data = train)
        rmse_train_sig3[i, m] = rmse(train$y, predict(model, train))
```

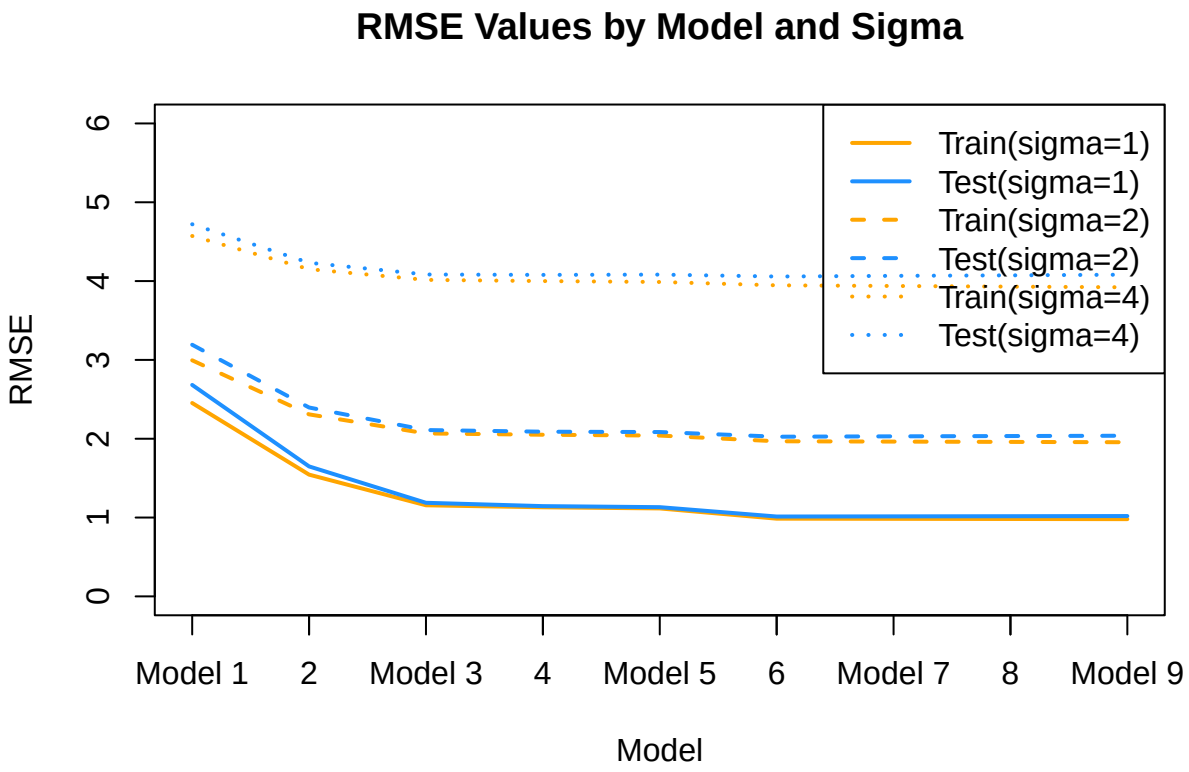
```
        rmse_test_sig3[i, m] = rmse(test$y, predict(model, test))
      }
    }
  }
}
```

Results

The **results** section should contain numerical or graphical summaries of your results as they pertain to the goal of each study. [For the Midterm Assignment, while the code chunks appear in Methods, the actual plots, numeric tables, etc. would appear in Results.] ### Summary

```
# Average RMSE values
avg_rmse_train_sig1 = colMeans(rmse_train_sig1)
avg_rmse_test_sig1 = colMeans(rmse_test_sig1)
avg_rmse_train_sig2 = colMeans(rmse_train_sig2)
avg_rmse_test_sig2 = colMeans(rmse_test_sig2)
avg_rmse_train_sig3 = colMeans(rmse_train_sig3)
avg_rmse_test_sig3 = colMeans(rmse_test_sig3)

plot_data = data.frame(avg_rmse_train_sig1 = avg_rmse_train_sig1,
                       avg_rmse_test_sig1 = avg_rmse_test_sig1,
                       avg_rmse_train_sig2 = avg_rmse_train_sig2,
                       avg_rmse_test_sig2 = avg_rmse_test_sig2,
                       avg_rmse_train_sig3 = avg_rmse_train_sig3,
                       avg_rmse_test_sig3 = avg_rmse_test_sig3
                       )
# Line Graph
plot(NULL,
     xlim = c(1, 9),
     ylim = c(0, 6),
     xlab = "Model",
     ylab = "RMSE",
     main = "RMSE Values by Model and Sigma"
     )

# Lines for Sigma = 1
lines(1:9, avg_rmse_train_sig1, col = "orange", lty = 1, lwd = 2)
lines(1:9, avg_rmse_test_sig1, col = "dodgerblue", lty = 1, lwd = 2)

# Lines for Sigma = 2
lines(1:9, avg_rmse_train_sig2, col = "orange", lty = 2, lwd = 2)
lines(1:9, avg_rmse_test_sig2, col = "dodgerblue", lty = 2, lwd = 2)

# Lines for Sigma = 3
lines(1:9, avg_rmse_train_sig3, col = "orange", lty = 3, lwd = 2)
lines(1:9, avg_rmse_test_sig3, col = "dodgerblue", lty = 3, lwd = 2)

# Legend
legend("topright", legend = c("Train(sigma=1)", "Test(sigma=1)",
                              "Train(sigma=2)", "Test(sigma=2)",
                              "Train(sigma=4)", "Test(sigma=4)"),
```

```
       col = c("orange", "dodgerblue"), lty = c(1, 1, 2, 2, 3, 3), lwd = 2)
axis(1, at = 1:9, labels = paste0("Model ", 1:9))
```



Table

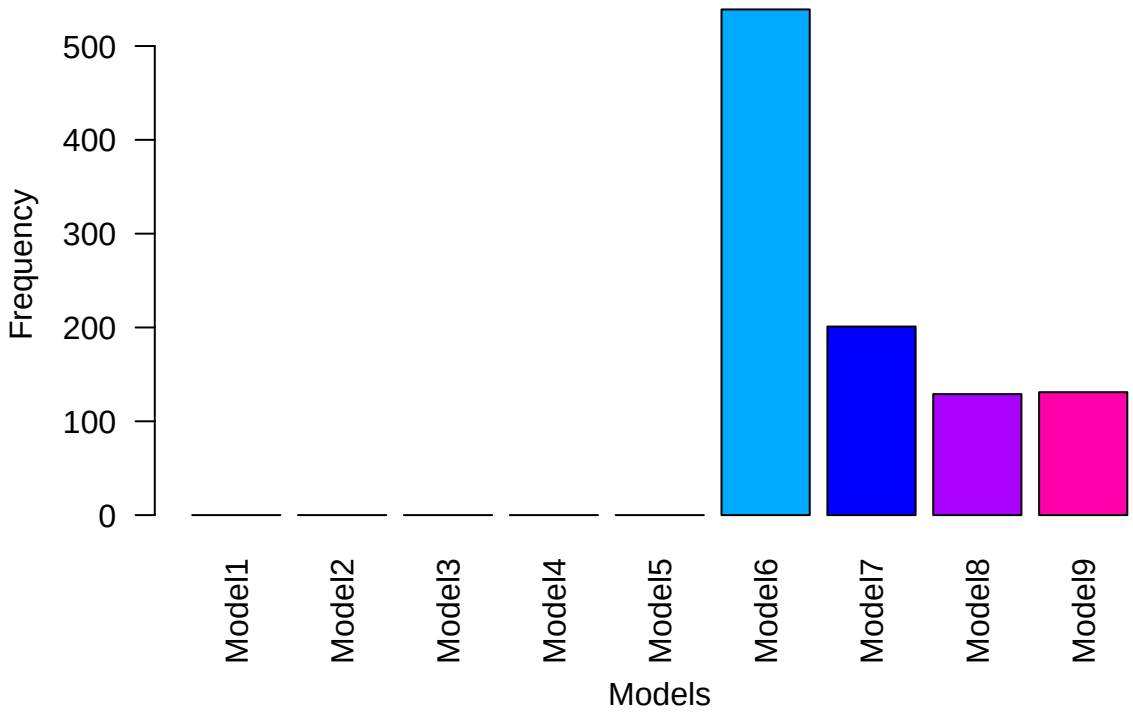
```
library(knitr)
# RMSE Frequency (Sigma = 1)
table_sig1 <- table(
  factor(
    colnames(rmse_test_sig1)[apply(rmse_test_sig1, 1, FUN = which.min)],
    levels = colnames(rmse_test_sig1)
  )
)

# RMSE Frequency (Sigma = 2)
table_sig2 <- table(
  factor(
    colnames(rmse_test_sig2)[apply(rmse_test_sig2, 1, FUN = which.min)],
    levels = colnames(rmse_test_sig2)
  )
)

# RMSE Frequency (Sigma = 4)
table_sig3 <- table(
  factor(
    colnames(rmse_test_sig3)[apply(rmse_test_sig3, 1, FUN = which.min)],
    levels = colnames(rmse_test_sig3)
  )
)
```

```r
full_table = cbind(table_sig1, table_sig2,table_sig3)
colnames(full_table) = c("min RMSE Frequency - Sigma = 1",
                         "min RMSE Frequency - Sigma = 2",
                         "min RMSE Frequency - Sigma = 4")
kable_columns = c("Model 1", "Model 2", "Model 3", "Model 4", "Model 5", "Model 6",
                  "Model 7", "Model 8", "Model 9")
kable(t(full_table), col.names = kable_columns)
```

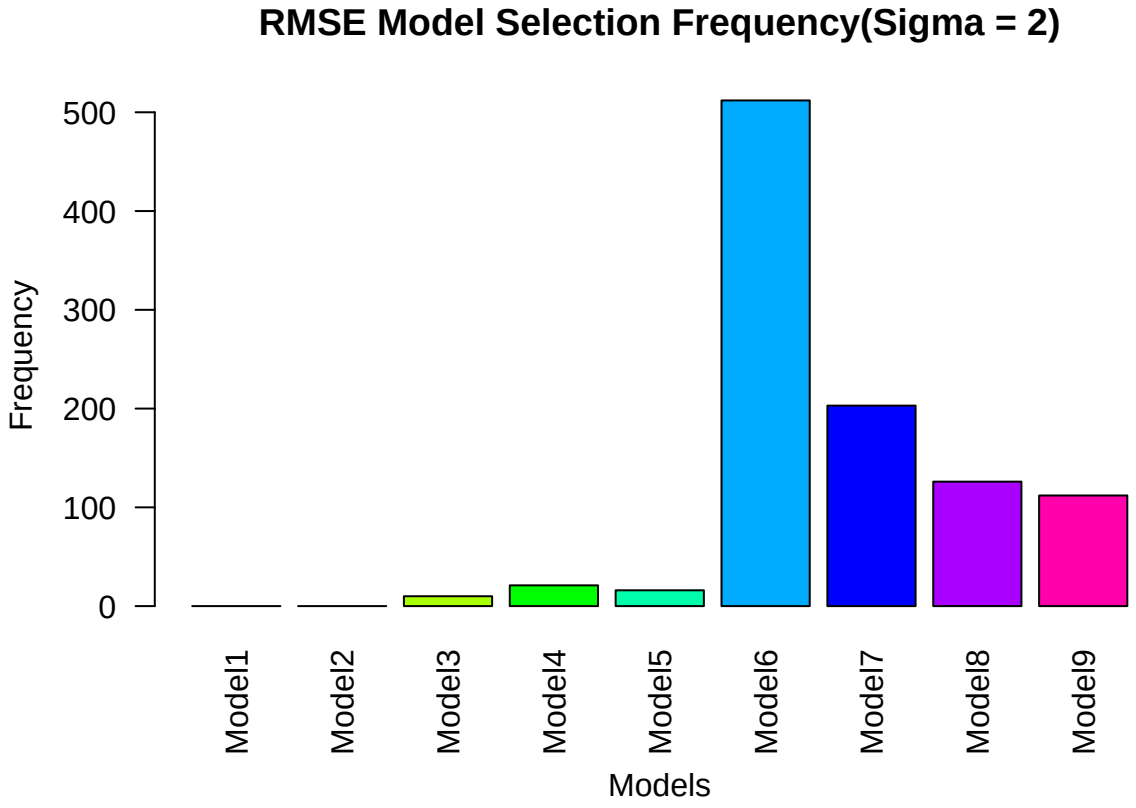
	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8	Model 9
min RMSE Frequency - Sigma = 1	0	0	0	0	0	539	201	129	131
min RMSE Frequency - Sigma = 2	0	0	10	21	16	512	203	126	112
min RMSE Frequency - Sigma = 4	0	14	145	111	60	354	130	98	88

Plotting

```r
# Plot
barplot(
  table(
    factor(
      colnames(rmse_test_sig1)[apply(rmse_test_sig1, 1, FUN = which.min)],
      levels = colnames(rmse_test_sig1)
    )
  ),
  main = "RMSE Model Selection Frequency(Sigma = 1)",
  sub = "Models",
  ylab = "Frequency",
  col = rainbow(9),
  las = 2
)
```




```r
# Plotting the minimum RMSE frequency for sigma = 2
barplot(
  table(
    factor(
      colnames(rmse_test_sig2)[apply(rmse_test_sig2, 1, FUN = which.min)],
      levels = colnames(rmse_test_sig2)
    )
  ),
  main = "RMSE Model Selection Frequency(Sigma = 2)",
  sub = "Models",
  ylab = "Frequency",
  col = rainbow(9),
  las = 2
)
```




```r
# Plotting the minimum RMSE frequency for sigma = 4
barplot(
  table(
    factor(
      colnames(rmse_test_sig3)[apply(rmse_test_sig3, 1, FUN = which.min)],
      levels = colnames(rmse_test_sig3)
    )
  ),
  main = "RMSE Model Selection Frequency(Sigma = 4)",
  sub = "Models",
  ylab = "Frequency",
  col = rainbow(9),
  las = 2
)
```

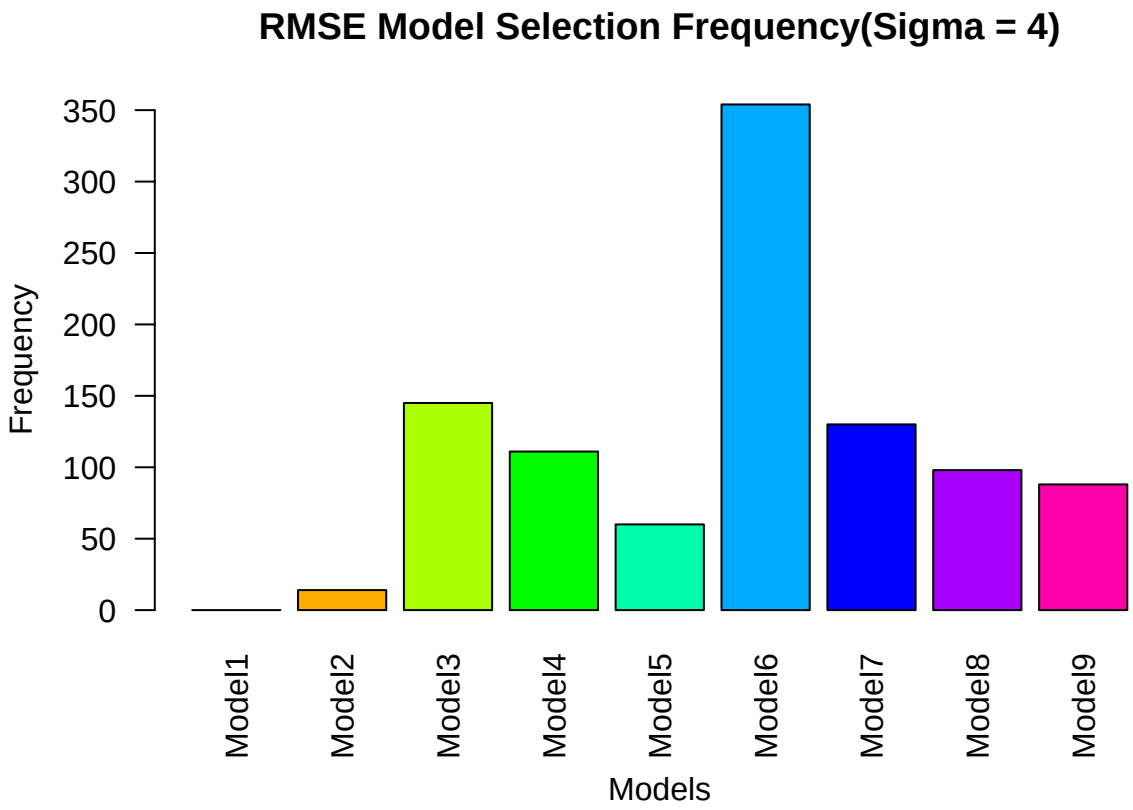


Table below summarizes the frequency of RMSE of test data for each model per sigma

```
library(knitr)
# RMSE Frequency (Sigma = 1)
table_sig1 <- table(
  factor(
    colnames(rmse_test_sig1)[apply(rmse_test_sig1, 1, FUN = which.min)],
    levels = colnames(rmse_test_sig1)
  )
)

# RMSE Frequency (Sigma = 2)
table_sig2 <- table(
  factor(
    colnames(rmse_test_sig2)[apply(rmse_test_sig2, 1, FUN = which.min)],
    levels = colnames(rmse_test_sig2)
  )
)

# RMSE Frequency (Sigma = 4)
table_sig3 <- table(
  factor(
    colnames(rmse_test_sig3)[apply(rmse_test_sig3, 1, FUN = which.min)],
    levels = colnames(rmse_test_sig3)
  )
)

full_table = cbind(table_sig1, table_sig2,table_sig3)
colnames(full_table) = c("Model Selection Frequency(Sigma = 1)",
```

```
                        "Model Selection Frequency(Sigma = 2)",
                        "Model Selection Frequency(Sigma = 4)"
                        )
kable_columns = c("Model 1", "Model 2", "Model 3", "Model 4", "Model 5",
                  "Model 6", "Model 7", "Model 8", "Model 9")
kable(t(full_table), col.names = kable_columns)
```

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8	Model 9
Model Selection Frequency(Sigma = 1)	0	0	0	0	0	539	201	129	131
Model Selection Frequency(Sigma = 2)	0	0	10	21	16	512	203	126	112
Model Selection Frequency(Sigma = 4)	0	14	145	111	60	354	130	98	88

Discussion

As we can see from the plots from **Result** section, this method don't always select the correct model. However, as shown from the across the different sigma levels, Model 6 which is the correct model, got the most frequency. We can expect RMSE method to choose correct model on average.

The level of noise affect the selection and we can see from the plots and the summary table that the frequency of Model 6 decreases as the level of noise increases.

As shown in the summary line plot in the **Result** section plots the average RMSE values from the 3 different sigma levels and by the models. As sigma level decreases, the average RMSE value decreases suggesting better fit of model with lower sigma which makes sense.
It shows that both train and test model at sigma 1 show the lowest average RMSE values across the sigma levels.

Simulation Study 3: Power

Introduction

In this simulation study we will investigate the **power** of the significance of regression test for simple linear regression.

$$H_0 : \beta_1 = 0 \text{ vs } H_1 : \beta_1 \neq 0$$

Recall, we had defined the *significance* level, α, to be the probability of a Type I error.

$$\alpha = P[\text{Reject } H_0 \mid H_0 \text{ True}] = P[\text{Type I Error}]$$

Similarly, the probability of a Type II error is often denoted using β; however, this should not be confused with a regression parameter.

$$\beta = P[\text{Fail to Reject } H_0 \mid H_1 \text{ True}] = P[\text{Type II Error}]$$

Power is the probability of rejecting the null hypothesis when the null is not true, that is, the alternative is true and β_1 is non-zero.

$$\text{Power} = 1 - \beta = P[\text{Reject } H_0 \mid H_1 \text{ True}]$$

Essentially, power is the probability that a signal of a particular strength will be detected. Many things affect the power of a test. In this case, some of those are:

- Sample Size, n
- Signal Strength, β_1
- Noise Level, σ
- Significance Level, α

We'll investigate the first three.

To do so we will simulate from the model

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

where $\epsilon_i \sim N(0, \sigma^2)$.

For simplicity, we will let $\beta_0 = 0$, thus β_1 is essentially controlling the amount of "signal." We will then consider different signals, noises, and sample sizes:

- $\beta_1 \in (-2, -1.9, -1.8, \ldots, -0.1, 0, 0.1, 0.2, 0.3, \ldots 1.9, 2)$
- $\sigma \in (1, 2, 4)$
- $n \in (10, 20, 30)$

We will hold the significance level constant at $\alpha = 0.05$.

We use the following code to generate the predictor values, `x`: values for different sample sizes.

```
x_values = seq(0, 5, length = n)
```

For each possible β_1 and σ combination, we simulate from the true model at least 1000 times. Each time, perform the significance of the regression test. To estimate the power with these simulations, and some α, use

$$\hat{\text{Power}} = \hat{P}[\text{Reject } H_0 \mid H_1 \text{ True}] = \frac{\text{number of Tests Rejected}}{\text{number of Simulations}}$$

It is *possible* to derive an expression for power mathematically, but often this is difficult, so instead, we rely on simulation.

Methods

Seed set up for reproducibility

```
birthday = 19870503
set.seed(birthday)
```

Variable Setup

```
beta_0 = 0
beta_1s = seq(-2, 2, by = 0.1)
sigmas = c(1,2,4)
ns = c(10,20,30)
alpha = 0.05
num_sims = 1000


results = data.frame()

for(sig in sigmas){
  for(n in ns){
    for(beta_1 in beta_1s){
      reject_count = 0
      for(i in 1:num_sims){
        # x values
        x_values = seq(0, 5, length = n)
        eps = rnorm(n, 0, sig)
        y = beta_0 + beta_1 * x_values + eps

        model = lm(y ~ x_values)
        p_value = summary(model)$coef[2,4]
        if(p_value < alpha)
        {
          reject_count = reject_count + 1
        }
      }
      power = reject_count / num_sims
      result = data.frame(Sigma = sig, n = n, Beta_1 = beta_1, Power = power)
      results = rbind(results, result)
    }
  }
}
```

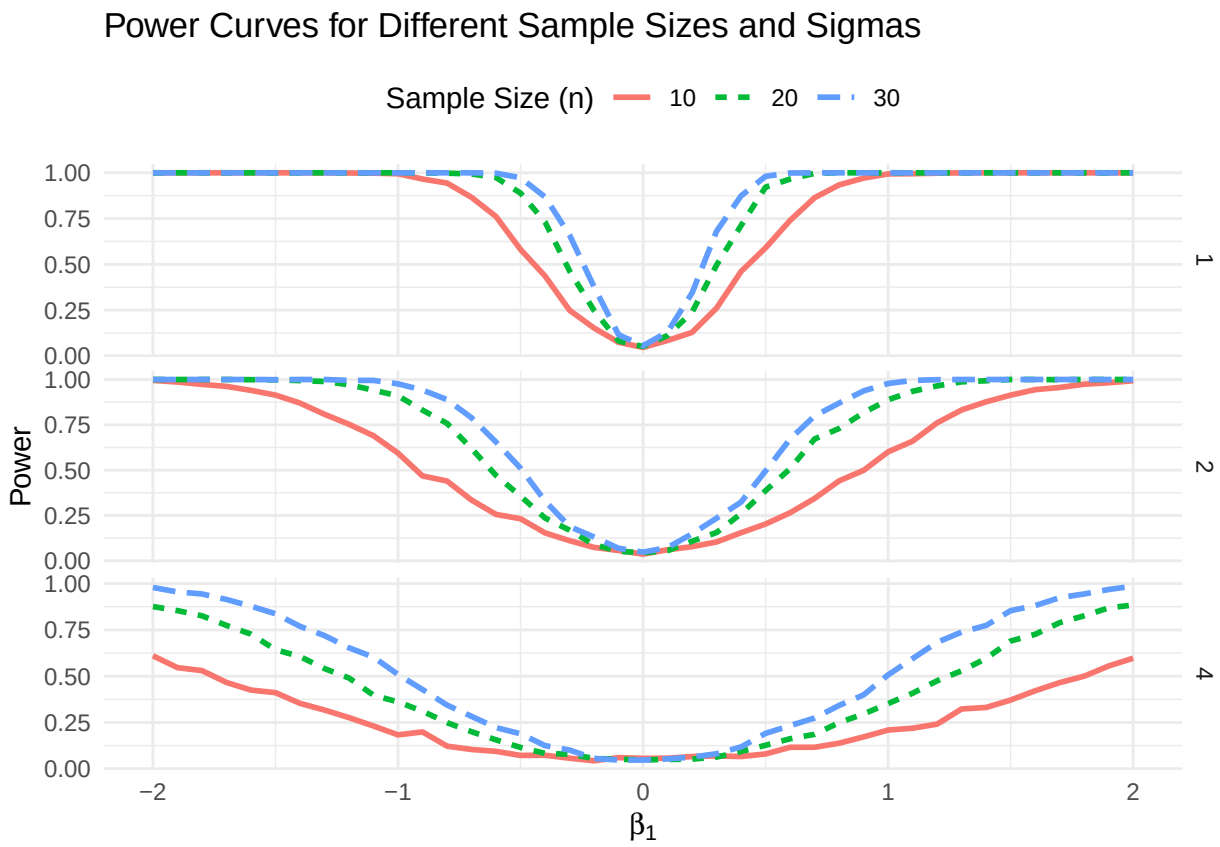
Results

Plot

We created three plots, one for each value of σ. Within each of these plots, add a "power curve" for each value of n that shows how power is affected by signal strength, β_1.

```
library(ggplot2)
ggplot(results, aes(x = Beta_1, y = Power, color = factor(n), linetype = factor(n))) +
  geom_line(size = 1) +
  scale_color_discrete(name = "Sample Size (n)") +
  scale_linetype_discrete(name = "Sample Size (n)") +
  facet_grid(Sigma ~ ., scales = "free") +
  labs(x = expression(beta[1]), y = "Power", title = "Power Curves for Different Sample Sizes and Sigmas
  theme_minimal() +
  theme(legend.position = "top")
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```



Discussion

Looking at the power curve plots, first we see that as the number of sample size increases, the power of the significance of regression test increase. This suggests that larger number of sample size affect the power of the test. Second, we can see that as the $|Beta_1|$ value increase, power value increases. $Beta_1$ values closer to 0 have the least power as expected since we set $B_0 = 0$ and $Beta_1$ essentially is controling the amount of "signal" in our model.(signal strength) $Y_i = \beta_0 + \beta_1 x_1 + \epsilon_i$ If we recall t-statistics essentially is equal to (Estimated - Hypothesized value) / SE, the greater the difference between the parameter and hypothesized value, the greater t statistic value suggests stronger evidence against null hypothesis. Third, we see that as *sigma* level increases, power decreases. Generally as the noise level increase, we can see the power curve gets steeper around $beta_1$ value 0 suggesting the inverse relationship between the power and the level of noise.

We set simulation size of 1000 for this simulation study. From observing the results of this simulation and considering the size, we feel the number of simulation is reasonably sufficient to observe the effects of $beta_1$, *sigma*, and sample size(n) based on this study.
