Week 6 - Midterm Assignment: Simulation Project

STAT 420, Summer 2023, Yogi Patel (ypatel55)

Contents

Simulation Study 1: Significance of Regression

Introduction

In this simulation study, I will investigate the significance of regression test. I 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 of these models, I wil consider a sample size of 25 and three levels of noise, or σ.

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

Then I will use 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, I will do 2000 simulations. For each simulation, I will also fit a regression model of the same form used to perform the simulation.

I will be using the data found in `study_1.csv` for the values of the predictors. These will be kept constant for the entirety of this study. The `y` values in this data are a blank placeholder.

Methods

First I will read in the data and set the seed.

```
library(readr)
study1_data = read_csv("study_1.csv")
birthday = 20020527
set.seed(birthday)
```

Now I will set up the variables for both models that will be used later in this simulation study.

```
mod1_beta_0 = 3
mod1_beta_1 = 1
mod1_beta_2 = 1
mod1_beta_3 = 1

mod2_beta_0 = 3
mod2_beta_1 = 0
mod2_beta_2 = 0
mod2_beta_3 = 0

sample_size = 25
simulations = 2000
sigma = c(1,5,10)

x0 = rep(1, sample_size)
x1 = study1_data$x1
x2 = study1_data$x2
x3 = study1_data$x3

p = 3 + 1

# creating empty dataframes to store simulation results for each value of sigma,
#containing values for the F statistic, P value, and R^2 value for each model
simulation_1_results = data.frame(mod1_F = rep(0, simulations),
                                  mod1_P = rep(0,simulations),
```

```
                                    mod1_R2 = rep(0,simulations),
                                    mod2_F = rep(0, simulations),
                                    mod2_P = rep(0,simulations),
                                    mod2_R2 = rep(0, simulations))

simulation_2_results = data.frame(mod1_F = rep(0, simulations),
                                    mod1_P = rep(0,simulations),
                                    mod1_R2 = rep(0,simulations),
                                    mod2_F = rep(0, simulations),
                                    mod2_P = rep(0,simulations),
                                    mod2_R2 = rep(0, simulations))

simulation_3_results = data.frame(mod1_F = rep(0, simulations),
                                    mod1_P = rep(0,simulations),
                                    mod1_R2 = rep(0,simulations),
                                    mod2_F = rep(0, simulations),
                                    mod2_P = rep(0,simulations),
                                    mod2_R2 = rep(0, simulations))
```

Here I am creating a function called "sim_slr" that will be used to run the simulations. The function takes the beta and sigma values as inputs and returns the y value.

```
sim_slr = function(beta_0, beta_1, beta_2, beta_3, sigma) {
  epsilon = rnorm(n = sample_size, mean = 0, sd = sigma)
  y = beta_0*x0 + beta_1*x1 + beta_2*x2 + beta_3*x3 + epsilon
}
```

Below, I begin by doing simulations for the first sigma value of 1. I use the dataframe called simulation_1_results to store the F **statistic**, the **p-value**, and the R^2 value for the significant model and the insignificant model. Then I call the sim_slr function 2000 times for each model and extract the values needed.

```
# simulations for sigma = 1

for (i in 1:simulations) {
  # For the first significant model
  study1_data$y = sim_slr(mod1_beta_0, mod1_beta_1, mod1_beta_2, mod1_beta_3, sigma[1])
  mod1 = lm(y ~ ., data=study1_data)
  simulation_1_results$mod1_F[i] = summary(mod1)$fstatistic[[1]]
  simulation_1_results$mod1_P[i] = pf(summary(mod1)$fstatistic[[1]],
                                      df1 = p - 1, df2 = sample_size - p, lower.tail = FALSE)
  simulation_1_results$mod1_R2[i] = summary(mod1)$r.squared

  # For the second non significant model
  study1_data$y = sim_slr(mod2_beta_0, mod2_beta_1, mod2_beta_2, mod2_beta_3, sigma[1])
  mod2 = lm(y ~ ., data=study1_data)
  simulation_1_results$mod2_F[i] = summary(mod2)$fstatistic[[1]]
  simulation_1_results$mod2_P[i] = pf(summary(mod2)$fstatistic[[1]],
                                      df1 = p - 1, df2 = sample_size - p, lower.tail = FALSE)
  simulation_1_results$mod2_R2[i] = summary(mod2)$r.squared
}
```

Now, I run simulations for the second sigma value of 5. I use the dataframe called simulation_2_results to store the F **statistic**, the **p-value**, and the R^2 value for the significant model and the insignificant model. Then I call the sim_slr function 2000 times for each model and extract the values needed.

```r
# simulations for sigma = 5

for (i in 1:simulations) {
  # For the first significant model
  study1_data$y = sim_slr(mod1_beta_0, mod1_beta_1, mod1_beta_2, mod1_beta_3, sigma[2])
  mod1 = lm(y ~ ., data=study1_data)
  simulation_2_results$mod1_F[i] = summary(mod1)$fstatistic[[1]]
  simulation_2_results$mod1_P[i] = pf(summary(mod1)$fstatistic[[1]],
                                      df1 = p - 1, df2 = sample_size - p, lower.tail = FALSE)
  simulation_2_results$mod1_R2[i] = summary(mod1)$r.squared

  # For the second non significant model
  study1_data$y = sim_slr(mod2_beta_0, mod2_beta_1, mod2_beta_2, mod2_beta_3, sigma[2])
  mod2 = lm(y ~ ., data=study1_data)
  simulation_2_results$mod2_F[i] = summary(mod2)$fstatistic[[1]]
  simulation_2_results$mod2_P[i] = pf(summary(mod2)$fstatistic[[1]],
                                      df1 = p - 1, df2 = sample_size - p, lower.tail = FALSE)
  simulation_2_results$mod2_R2[i] = summary(mod2)$r.squared
}
```

Finally, I run simulations for the third sigma value of 10. I use the dataframe called simulation_3_results to store the F **statistic**, the **p-value**, and the R^2 value for the significant model and the insignificant model. Then I call the sim_slr function 2000 times for each model and extract the values needed.

```r
# simulations for sigma = 10

for (i in 1:simulations) {
  # For the first significant model
  study1_data$y = sim_slr(mod1_beta_0, mod1_beta_1, mod1_beta_2, mod1_beta_3, sigma[3])
  mod1 = lm(y ~ ., data=study1_data)
  simulation_3_results$mod1_F[i] = summary(mod1)$fstatistic[[1]]
  simulation_3_results$mod1_P[i] = pf(summary(mod1)$fstatistic[[1]],
                                      df1 = p - 1, df2 = sample_size - p, lower.tail = FALSE)
  simulation_3_results$mod1_R2[i] = summary(mod1)$r.squared

  # For the second non significant model
  study1_data$y = sim_slr(mod2_beta_0, mod2_beta_1, mod2_beta_2, mod2_beta_3, sigma[3])
  mod2 = lm(y ~ ., data=study1_data)
  simulation_3_results$mod2_F[i] = summary(mod2)$fstatistic[[1]]
  simulation_3_results$mod2_P[i] = pf(summary(mod2)$fstatistic[[1]],
                                      df1 = p - 1, df2 = sample_size - p, lower.tail = FALSE)
  simulation_3_results$mod2_R2[i] = summary(mod2)$r.squared
}
```

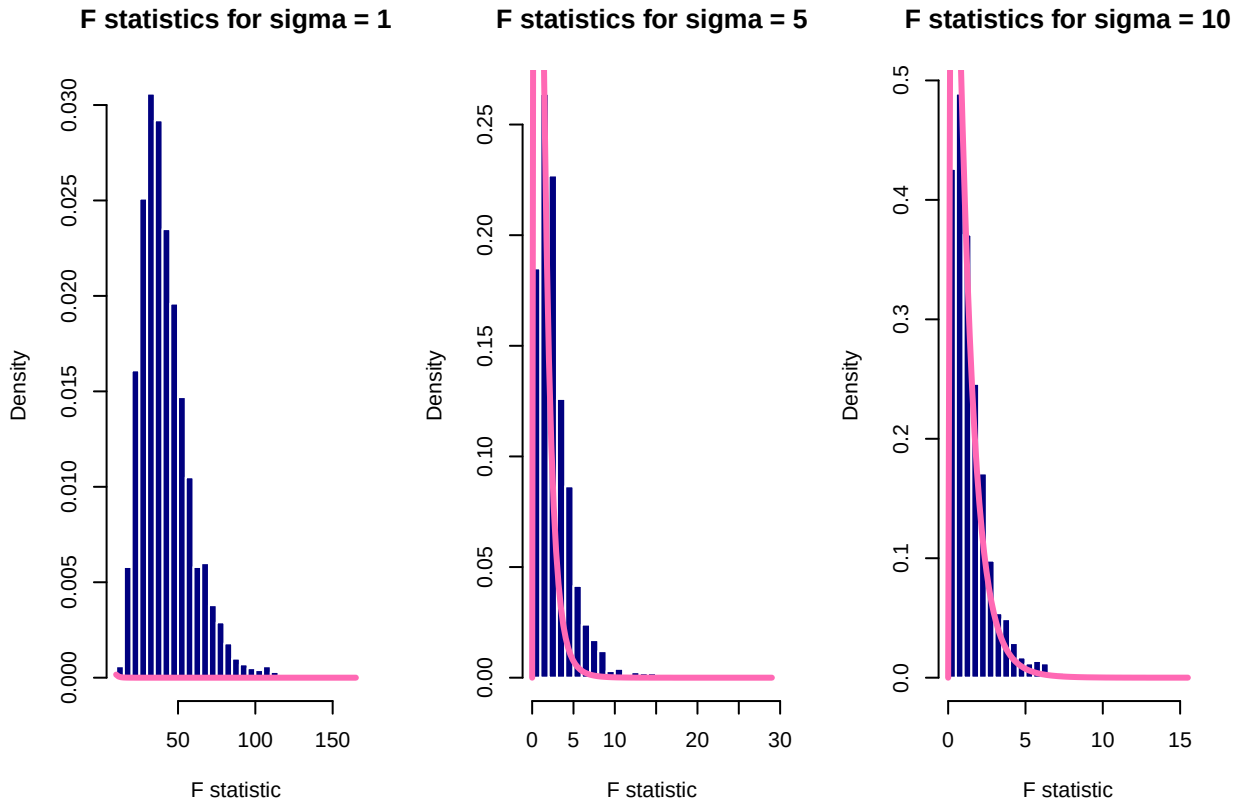
Results

Graphs for significant model

Below I display a 1 x 3 row of F statistic plots as σ changes, then a 1 x 3 row of p-value plots as σ changes, followed by a similar row for the R^2 values for the significant model.

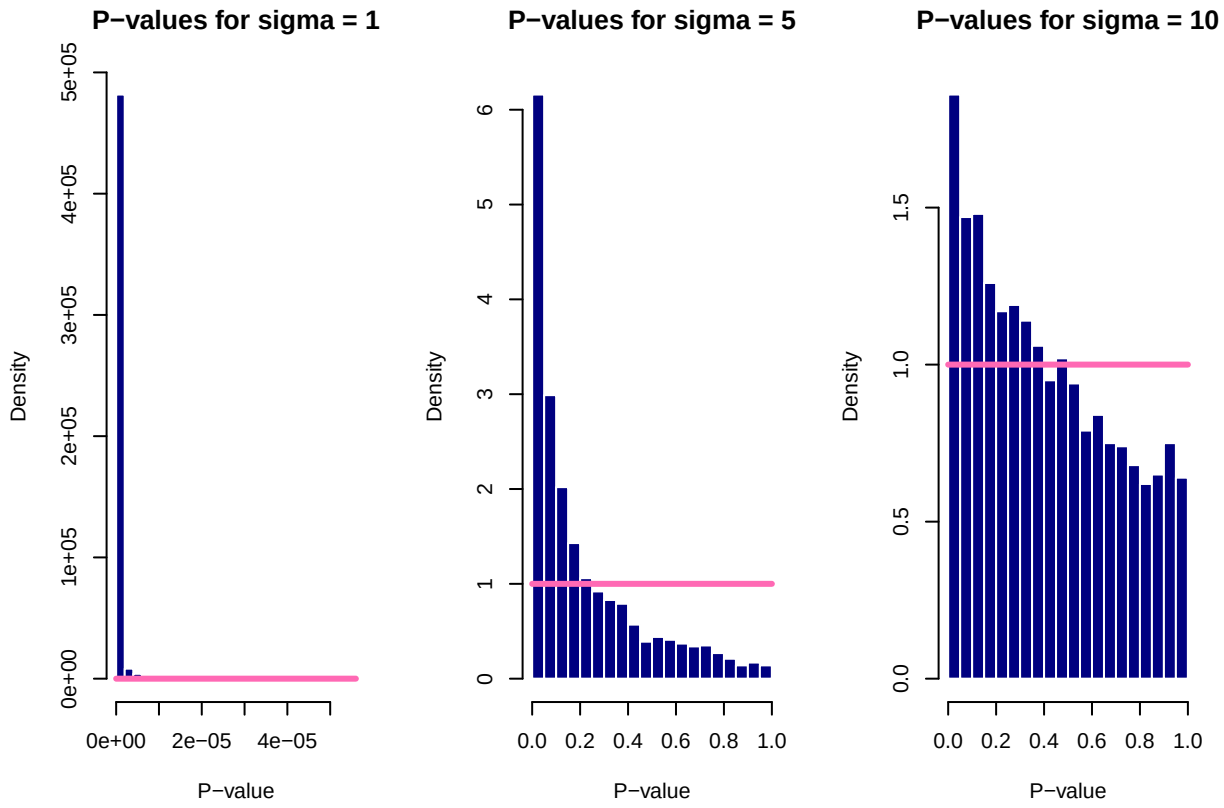
```r
# Significant model graphs

par(mfrow=c(1,3))
```

```r
# graphs for F statistics for each sigma of significant model
x = simulation_1_results$mod1_F
hist(x, prob=TRUE, breaks = 30, main = "F statistics for sigma = 1",
     xlab = "F statistic", col='navy', border='white')
curve(df(x, df1 = p - 1, df2 = sample_size - p), col = "hotpink", add = TRUE, lwd = 3)

x = simulation_2_results$mod1_F
hist(x, prob=TRUE, breaks = 30, main = "F statistics for sigma = 5",
     xlab = "F statistic", col='navy', border='white')
curve(df(x, df1 = p - 1, df2 = sample_size - p), col = "hotpink", add = TRUE, lwd = 3)

x = simulation_3_results$mod1_F
hist(x, prob=TRUE, breaks = 30, main = "F statistics for sigma = 10",
     xlab = "F statistic", col='navy', border='white')
curve(df(x, df1 = p - 1, df2 = sample_size - p), col = "hotpink", add = TRUE, lwd = 3)
```

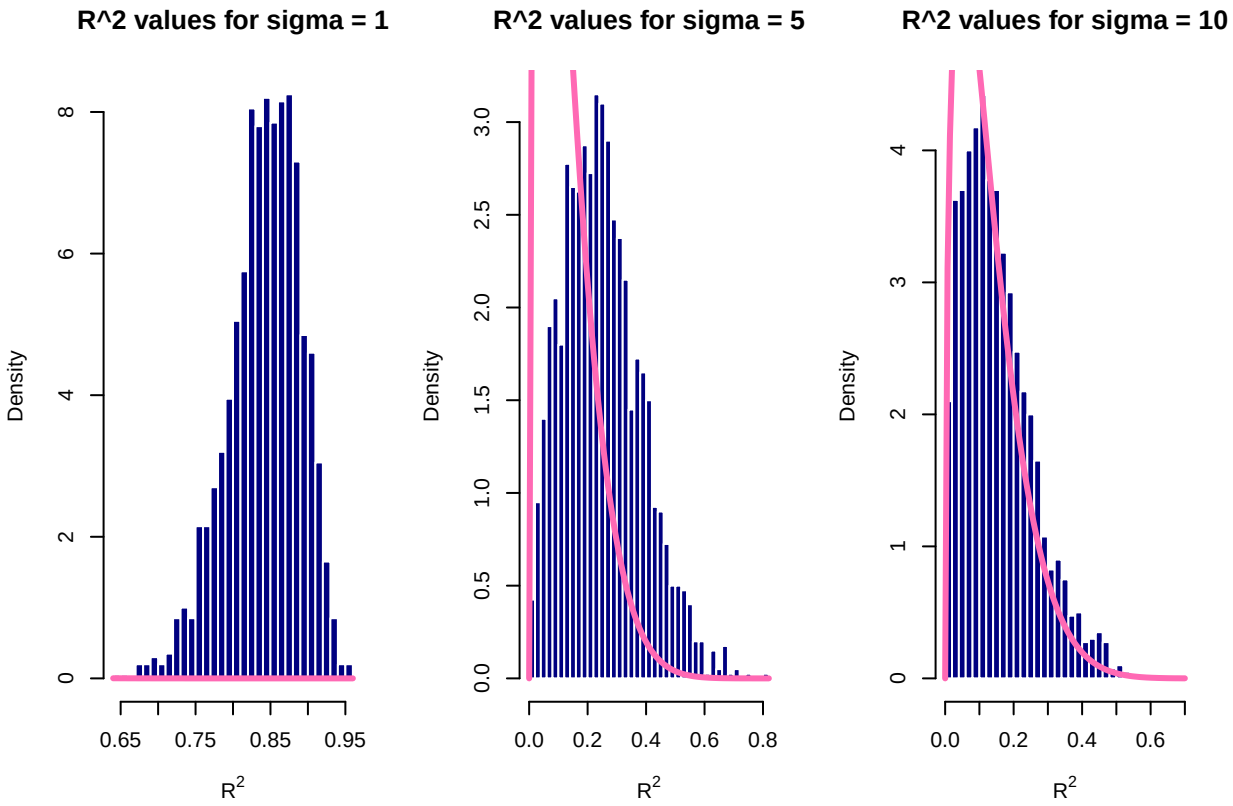



```r
# graphs for p values for each sigma of significant model
x = simulation_1_results$mod1_P
hist(x, prob=TRUE, breaks = 30, main = "P-values for sigma = 1",
     xlab = "P-value", col='navy', border='white')
curve(dunif(x), col='hotpink', add=TRUE, lwd=3)

x = simulation_2_results$mod1_P
hist(x, prob=TRUE, breaks = 30, main = "P-values for sigma = 5",
     xlab = "P-value", col='navy', border='white')
curve(dunif(x), col='hotpink', add=TRUE, lwd=3)

x = simulation_3_results$mod1_P
```

```r
hist(x, prob=TRUE, breaks = 30, main = "P-values for sigma = 10",
     xlab = "P-value", col='navy', border='white')
curve(dunif(x), col='hotpink', add=TRUE, lwd=3)
```



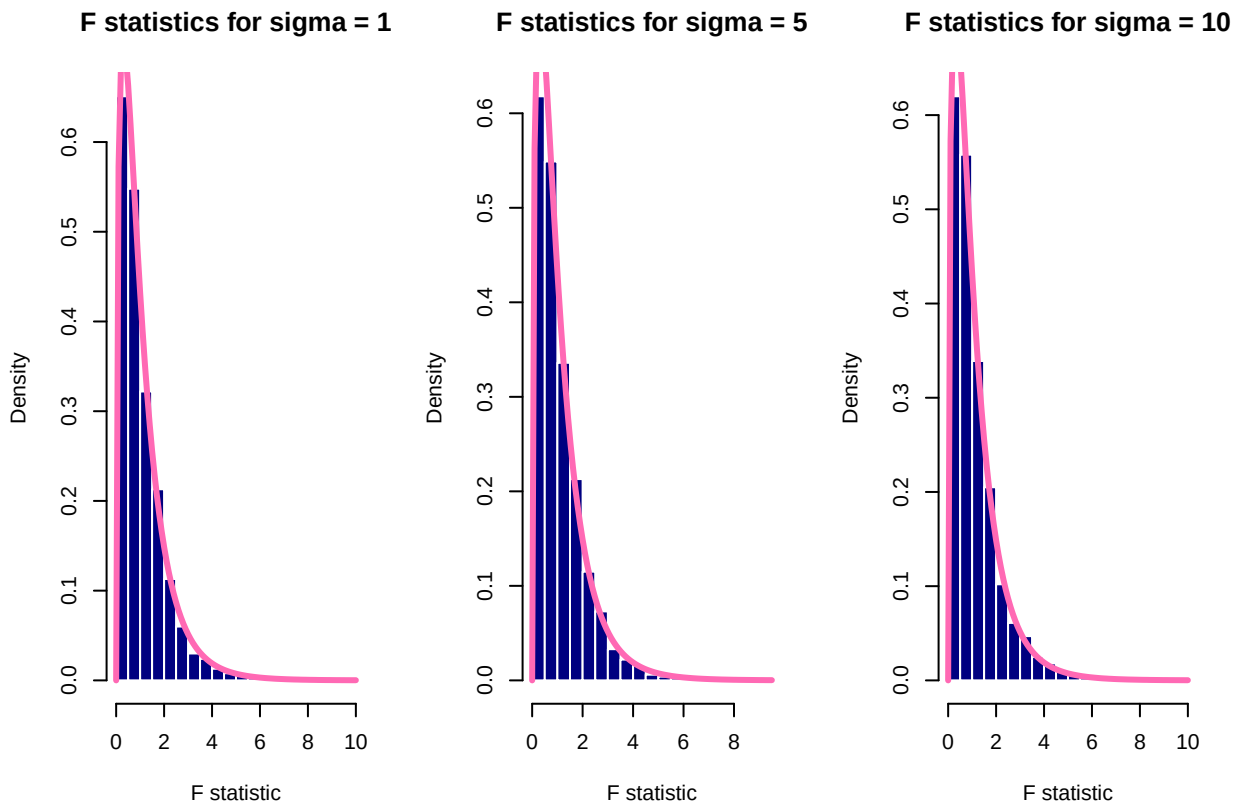

```r
# graphs for R^2 values for each sigma of significant model
x = simulation_1_results$mod1_R2
hist(x, prob=TRUE, breaks = 30, main = "R^2 values for sigma = 1",
     xlab = expression(R^2), col='navy', border='white')
curve(dbeta(x, (p-1)/2,(sample_size-p)/2), col="hotpink", add=TRUE, lwd=3)

x = simulation_2_results$mod1_R2
hist(x, prob=TRUE, breaks = 30, main = "R^2 values for sigma = 5",
     xlab = expression(R^2), col='navy', border='white')
curve(dbeta(x, (p-1)/2,(sample_size-p)/2), col="hotpink", add=TRUE, lwd=3)

x = simulation_3_results$mod1_R2
hist(x, prob=TRUE, breaks = 30, main = "R^2 values for sigma = 10",
     xlab = expression(R^2), col='navy', border='white')
curve(dbeta(x, (p-1)/2,(sample_size-p)/2), col="hotpink", add=TRUE, lwd=3)
```



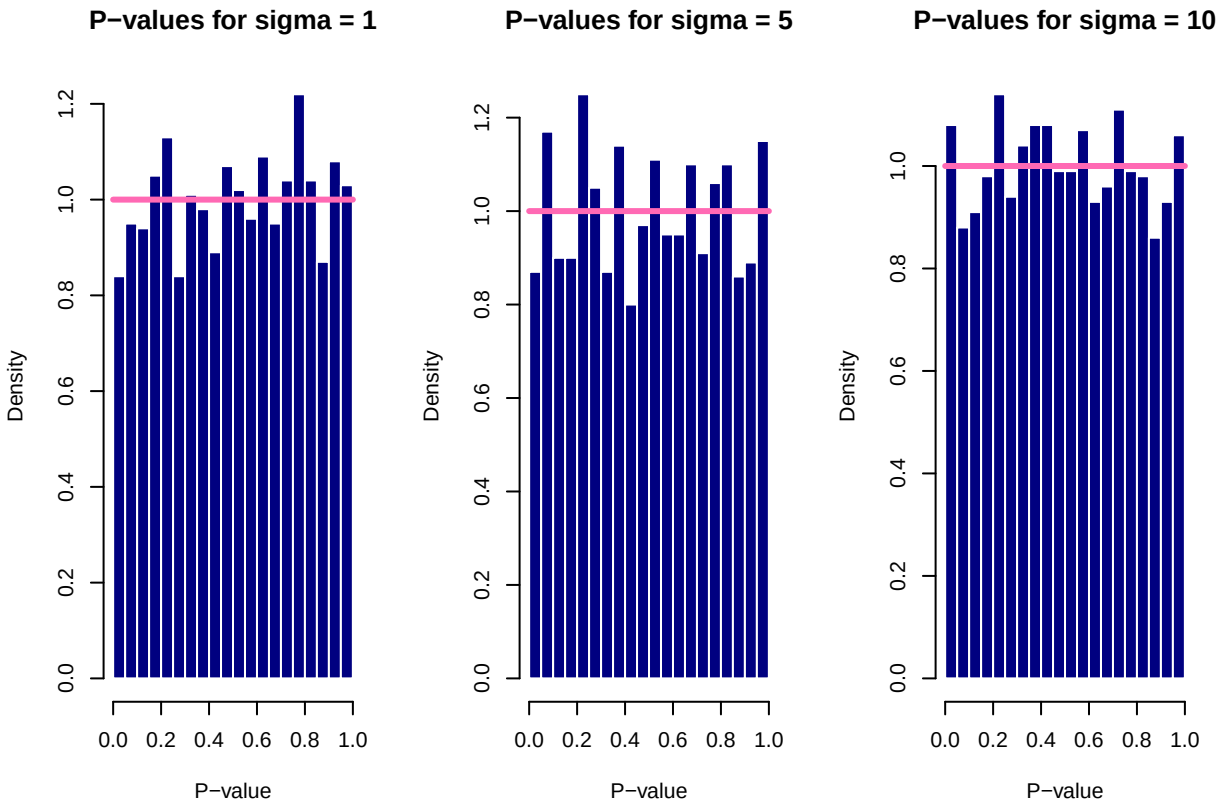
Graphs for non-significant model

Below I display a 1 x 3 row of F statistic plots as σ changes, then a 1 $\times$ 3 row of p-value plots as σ changes, followed by a similar row for the R^2 values for the non-significant model.

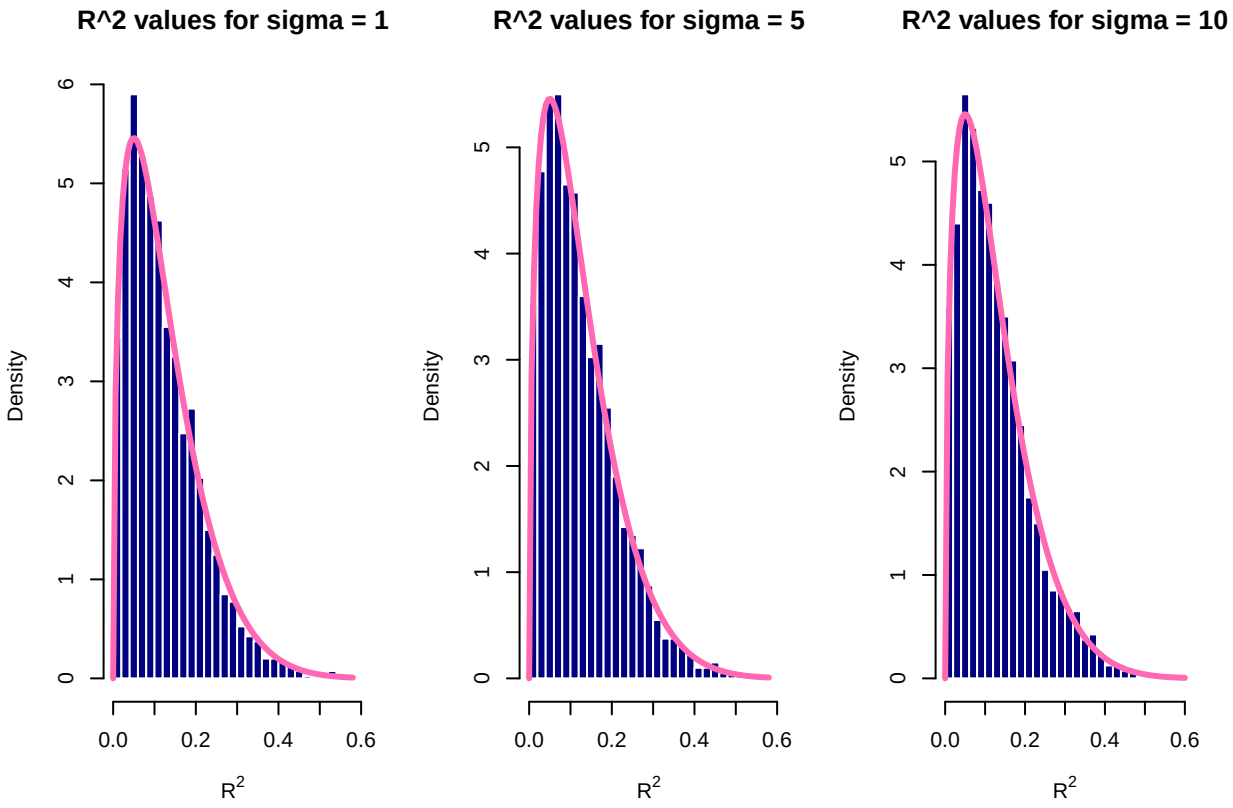
```
# Non-significant model graphs

par(mfrow=c(1,3))

# graphs for F statistics for each sigma of non significant model
x = simulation_1_results$mod2_F
hist(x, prob=TRUE, breaks = 30, main = "F statistics for sigma = 1",
     xlab = "F statistic", col='navy', border='white')
curve(df(x, df1 = p - 1, df2 = sample_size - p), col = "hotpink", add = TRUE, lwd = 3)

x = simulation_2_results$mod2_F
hist(x, prob=TRUE, breaks = 30, main = "F statistics for sigma = 5",
     xlab = "F statistic", col='navy', border='white')
curve(df(x, df1 = p - 1, df2 = sample_size - p), col = "hotpink", add = TRUE, lwd = 3)

x = simulation_3_results$mod2_F
hist(x, prob=TRUE, breaks = 30, main = "F statistics for sigma = 10",
     xlab = "F statistic", col='navy', border='white')
curve(df(x, df1 = p - 1, df2 = sample_size - p), col = "hotpink", add = TRUE, lwd = 3)
```




```
# graphs for p values for each sigma of non significant model
x = simulation_1_results$mod2_P
hist(x, prob=TRUE, breaks = 30, main = "P-values for sigma = 1",
     xlab = "P-value", col='navy', border='white')
curve(dunif(x), col='hotpink', add=TRUE, lwd=3)

x = simulation_2_results$mod2_P
hist(x, prob=TRUE, breaks = 30, main = "P-values for sigma = 5",
     xlab = "P-value", col='navy', border='white')
curve(dunif(x), col='hotpink', add=TRUE, lwd=3)

x = simulation_3_results$mod2_P
hist(x, prob=TRUE, breaks = 30, main = "P-values for sigma = 10",
     xlab = "P-value", col='navy', border='white')
curve(dunif(x), col='hotpink', add=TRUE, lwd=3)
```




```
# graphs for R^2 values for each sigma of non significant model
x = simulation_1_results$mod2_R2
hist(x, prob=TRUE, breaks = 30, main = "R^2 values for sigma = 1",
     xlab = expression(R^2), col='navy', border='white')
curve(dbeta(x, (p-1)/2,(sample_size-p)/2), col="hotpink", add=TRUE, lwd=3)

x = simulation_2_results$mod2_R2
hist(x, prob=TRUE, breaks = 30, main = "R^2 values for sigma = 5",
     xlab = expression(R^2), col='navy', border='white')
curve(dbeta(x, (p-1)/2,(sample_size-p)/2), col="hotpink", add=TRUE, lwd=3)

x = simulation_3_results$mod2_R2
hist(x, prob=TRUE, breaks = 30, main = "R^2 values for sigma = 10",
     xlab = expression(R^2), col='navy', border='white')
curve(dbeta(x, (p-1)/2,(sample_size-p)/2), col="hotpink", add=TRUE, lwd=3)
```



Discussion

The significance of regression test, tells us whether our linear regression model is a better fit to a dataset than a model with no predictor variables. In this simulation study the model we are testing is the significant model vs the non-significant model.

For both of these models, we do know the true distribution of the values.

Starting with the F **statistic**, we know that the result should follow a standard F distribution with p-q and n-p degrees of freedom. So we know that p - q = (4 - 1) = 3 and n - p = (2000 - 4 parameters) = 1996. The non-significant model is nested in the significant model. We can refer to the significant model as the full model and the non-significant model as the null model. In the graphs from the results section, I have added curves to show these true distributions. In these graphs, we can see that the F statistic graphs for the models and all three sigma values roughly follow the true distribution except for the significant full model, especially seen at $\sigma = 1$. This proves that the significant model is actually significant. Because the F values do not follow the F distribution, we can conclude that the F statistic is larger than expected and the model will reject the null hypothesis for the full model. The empirical distribution compared to the true distribution varies for full model as seen, and as σ increases, the F statistics seem to follow the true distribution more. This overall means that as σ increases, we are more likely to not reject the null hypothesis as the F value gets smaller.

Now, looking at the **p-value**, we see that the non significant model sees larger p values than the significant one, proving again that our significant model is truly significant as smaller p-values will allow us to reject the null hypothesis. We know that when the null hypothesis is true, the p-values follow a uniform distribution. In our null model graphs, we see this uniform distribution clearly, but in our full model graphs, we see that the empirical distribution is no where near uniform. In these significant model p value graphs, we also see that as σ increases, the graphs follow the true uniform distribution more. We again can say that as σ increases, we are more likely to not reject the null hypothesis as p values get larger.

We expect that the true distribution of the R^2 values follow a Beta distribution. The non-significant model is

the null model and in its graphs we can see that this model follows this true distribution, with k-1/2 and n-k/2 where k is the number of parameters and n is the number of observations. The significant model does not follow this beta distribution. Also as σ increases, the R^2 values tend to decrease. This is because increasing σ would cause more noise. Again we see that the increase in σ also causes the empirical distributions to follow the true distribution more. Proving that our first model is significant and showing that we are more likely to not reject the null hypothesis as R^2 values decrease.

In all the graphs above, compared to the pink curves of true distributions, we can see that the empirical distributions tend to follow the true ones in the non-significant model but not so much in the significant model. This means our significance of regression test is correct in saying the first model is signficant. As σ increases in the significant model, the F statistic and p-value gets larger, while the R^2 gets smaller. In the non-significant model, as σ increases, we tend to not see much change in the empirical distributions for any of the values as the values already follow their true distributions due the model being insignificant.

Simulation Study 2: Using RMSE for Selection?

Introduction

In this simulation study, I will investigate how test RMSE can be used for selection of the "best" model when it is averaged over many attempts.

I 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

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

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

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

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

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

For each, I will 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`

Then for each model, I will calculate Train and Test RMSE.

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

This process will be repeated with 1000 simulations for each of the 3 values of σ.

Methods

First I will read in the data and set the seed.

```
# reading in data and setting the seed
library(readr)
study2_data = read_csv("study_2.csv")
birthday = 20020527
set.seed(birthday)
```

Now I will set up the variables that will be used later in this simulation study.

```
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

sample_size = 500
simulations = 1000
sigma = c(1,2,4)

x0 = rep(1, sample_size)
x1 = study2_data$x1
x2 = study2_data$x2
x3 = study2_data$x3
x4 = study2_data$x4
x5 = study2_data$x5
x6 = study2_data$x6


# creating empty data frames to store simulation train and test results for each level of
# sigma, containing the rmse for the nine models mentioned above
simulation1_RMSE_train_data = data.frame(mod1 = rep(0, simulations),
                                         mod2 = rep(0, simulations),
                                         mod3 = rep(0, simulations),
                                         mod4 = rep(0, simulations),
                                         mod5 = rep(0,simulations),
                                         mod6 = rep(0, simulations),
                                         mod7 = rep(0, simulations),
                                         mod8 = rep(0, simulations),
                                         mod9 = rep(0, simulations))
simulation1_RMSE_test_data = data.frame(mod1 = rep(0, simulations),
                                        mod2 = rep(0, simulations),
                                        mod3 = rep(0, simulations),
```

```
                                        mod4 = rep(0, simulations),
                                        mod5 = rep(0, simulations),
                                        mod6 = rep(0, simulations),
                                        mod7 = rep(0, simulations),
                                        mod8 = rep(0, simulations),
                                        mod9 = rep(0, simulations))

simulation2_RMSE_train_data = data.frame(mod1 = rep(0, simulations),
                                         mod2 = rep(0, simulations),
                                         mod3 = rep(0, simulations),
                                         mod4 = rep(0, simulations),
                                         mod5 = rep(0,simulations),
                                         mod6 = rep(0, simulations),
                                         mod7 = rep(0, simulations),
                                         mod8 = rep(0, simulations),
                                         mod9 = rep(0, simulations))
simulation2_RMSE_test_data = data.frame(mod1 = rep(0, simulations),
                                        mod2 = rep(0, simulations),
                                        mod3 = rep(0, simulations),
                                        mod4 = rep(0, simulations),
                                        mod5 = rep(0, simulations),
                                        mod6 = rep(0, simulations),
                                        mod7 = rep(0, simulations),
                                        mod8 = rep(0, simulations),
                                        mod9 = rep(0, simulations))

simulation3_RMSE_train_data = data.frame(mod1 = rep(0, simulations),
                                         mod2 = rep(0, simulations),
                                         mod3 = rep(0, simulations),
                                         mod4 = rep(0, simulations),
                                         mod5 = rep(0,simulations),
                                         mod6 = rep(0, simulations),
                                         mod7 = rep(0, simulations),
                                         mod8 = rep(0, simulations),
                                         mod9 = rep(0, simulations))
simulation3_RMSE_test_data = data.frame(mod1 = rep(0, simulations),
                                        mod2 = rep(0, simulations),
                                        mod3 = rep(0, simulations),
                                        mod4 = rep(0, simulations),
                                        mod5 = rep(0, simulations),
                                        mod6 = rep(0, simulations),
                                        mod7 = rep(0, simulations),
                                        mod8 = rep(0, simulations),
                                        mod9 = rep(0, simulations))
```

Here I am creating a function called “rmse” that will be used to calculate the rmse when it is called. The function takes the actual and prediction as inputs and returns the rmse value which is the average difference between the values predicted by the respective model and the actual value.

```
# Function to calculate RMSE
rmse = function(actual, predicted) {
  return (sqrt(mean(( actual - predicted )^2 )))
}
```

Here I am creating a function called “sim_slr” that will be used to run the simulations. The function takes

the beta and sigma values as inputs and returns the y value.

```
# Function to run simulations
sim_slr = function(beta_0, beta_1, beta_2, beta_3, beta_4, beta_5, beta_6, sigma) {
  epsilon = rnorm(n = sample_size, mean = 0, sd = sigma)
  y = beta_0*x0 + beta_1*x1 + beta_2*x2 + beta_3*x3 + beta_4*x4 + beta_5*x5 + beta_6*x6 + epsilon
}
```

Below, I begin by doing simulations for the first sigma value of 1. I use the dataframes called simulation1_RMSE_train_data and simulation1_RMSE_test_data to store the rmse values for the train and test data for each model. I also assign half of the data to train_data and half of the data to test_data.

```
# simulations for sigma = 1

for (i in 1:simulations) {
  study2_data$y = sim_slr(beta_0, beta_1, beta_2, beta_3, beta_4, beta_5, beta_6, sigma[1])

  # creating an index for half of the data, keeping everything up to this index for train data
  # then removing the train data and keeping the rest for the test data
  train_idx = sample(sample_size/2)
  train_data = study2_data[train_idx,]
  test_data = study2_data[-train_idx,]

  mod1 = lm(y ~ x1, data = train_data)
  mod2 = lm(y ~ x1 + x2, data = train_data)
  mod3 = lm(y ~ x1 + x2 + x3, data = train_data)
  mod4 = lm(y ~ x1 + x2 + x3 + x4, data = train_data)
  mod5 = lm(y ~ x1 + x2 + x3 + x4 + x5, data = train_data)
  mod6 = lm(y ~ x1 + x2 + x3 + x4 + x5 + x6, data = train_data)
  mod7 = lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7, data = train_data)
  mod8 = lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8, data = train_data)
  mod9 = lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9, data = train_data)

  simulation1_RMSE_train_data$mod1[i] = rmse(train_data$y, predict(mod1, train_data))
  simulation1_RMSE_train_data$mod2[i] = rmse(train_data$y, predict(mod2, train_data))
  simulation1_RMSE_train_data$mod3[i] = rmse(train_data$y, predict(mod3, train_data))
  simulation1_RMSE_train_data$mod4[i] = rmse(train_data$y, predict(mod4, train_data))
  simulation1_RMSE_train_data$mod5[i] = rmse(train_data$y, predict(mod5, train_data))
  simulation1_RMSE_train_data$mod6[i] = rmse(train_data$y, predict(mod6, train_data))
  simulation1_RMSE_train_data$mod7[i] = rmse(train_data$y, predict(mod7, train_data))
  simulation1_RMSE_train_data$mod8[i] = rmse(train_data$y, predict(mod8, train_data))
  simulation1_RMSE_train_data$mod9[i] = rmse(train_data$y, predict(mod9, train_data))

  simulation1_RMSE_test_data$mod1[i] = rmse(test_data$y, predict(mod1, test_data))
  simulation1_RMSE_test_data$mod2[i] = rmse(test_data$y, predict(mod2, test_data))
  simulation1_RMSE_test_data$mod3[i] = rmse(test_data$y, predict(mod3, test_data))
  simulation1_RMSE_test_data$mod4[i] = rmse(test_data$y, predict(mod4, test_data))
  simulation1_RMSE_test_data$mod5[i] = rmse(test_data$y, predict(mod5, test_data))
  simulation1_RMSE_test_data$mod6[i] = rmse(test_data$y, predict(mod6, test_data))
  simulation1_RMSE_test_data$mod7[i] = rmse(test_data$y, predict(mod7, test_data))
  simulation1_RMSE_test_data$mod8[i] = rmse(test_data$y, predict(mod8, test_data))
  simulation1_RMSE_test_data$mod9[i] = rmse(test_data$y, predict(mod9, test_data))

}
```

Now, I run simulations for the second sigma value of 2. I use the dataframes called simula-

tion2_RMSE_train_data and simulation2_RMSE_test_data to store the rmse values for the train and test data for each model. I also assign half of the data to train_data and half of the data to test_data again.

```
# simulations for sigma = 2

for (i in 1:simulations) {
  study2_data$y = sim_slr(beta_0, beta_1, beta_2, beta_3, beta_4, beta_5, beta_6, sigma[2])

  # creating an index for half of the data, keeping everything up to this index for train data
  # then removing the train data and keeping the rest for the test data
  train_idx = sample(sample_size/2)
  train_data = study2_data[train_idx,]
  test_data = study2_data[-train_idx,]

  mod1 = lm(y ~ x1, data = train_data)
  mod2 = lm(y ~ x1 + x2, data = train_data)
  mod3 = lm(y ~ x1 + x2 + x3, data = train_data)
  mod4 = lm(y ~ x1 + x2 + x3 + x4, data = train_data)
  mod5 = lm(y ~ x1 + x2 + x3 + x4 + x5, data = train_data)
  mod6 = lm(y ~ x1 + x2 + x3 + x4 + x5 + x6, data = train_data)
  mod7 = lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7, data = train_data)
  mod8 = lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8, data = train_data)
  mod9 = lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9, data = train_data)

  simulation2_RMSE_train_data$mod1[i] = rmse(train_data$y, predict(mod1, train_data))
  simulation2_RMSE_train_data$mod2[i] = rmse(train_data$y, predict(mod2, train_data))
  simulation2_RMSE_train_data$mod3[i] = rmse(train_data$y, predict(mod3, train_data))
  simulation2_RMSE_train_data$mod4[i] = rmse(train_data$y, predict(mod4, train_data))
  simulation2_RMSE_train_data$mod5[i] = rmse(train_data$y, predict(mod5, train_data))
  simulation2_RMSE_train_data$mod6[i] = rmse(train_data$y, predict(mod6, train_data))
  simulation2_RMSE_train_data$mod7[i] = rmse(train_data$y, predict(mod7, train_data))
  simulation2_RMSE_train_data$mod8[i] = rmse(train_data$y, predict(mod8, train_data))
  simulation2_RMSE_train_data$mod9[i] = rmse(train_data$y, predict(mod9, train_data))

  simulation2_RMSE_test_data$mod1[i] = rmse(test_data$y, predict(mod1, test_data))
  simulation2_RMSE_test_data$mod2[i] = rmse(test_data$y, predict(mod2, test_data))
  simulation2_RMSE_test_data$mod3[i] = rmse(test_data$y, predict(mod3, test_data))
  simulation2_RMSE_test_data$mod4[i] = rmse(test_data$y, predict(mod4, test_data))
  simulation2_RMSE_test_data$mod5[i] = rmse(test_data$y, predict(mod5, test_data))
  simulation2_RMSE_test_data$mod6[i] = rmse(test_data$y, predict(mod6, test_data))
  simulation2_RMSE_test_data$mod7[i] = rmse(test_data$y, predict(mod7, test_data))
  simulation2_RMSE_test_data$mod8[i] = rmse(test_data$y, predict(mod8, test_data))
  simulation2_RMSE_test_data$mod9[i] = rmse(test_data$y, predict(mod9, test_data))

}
```

Finally, I run simulations for the second sigma value of 4. I use the dataframes called simulation3_RMSE_train_data and simulation3_RMSE_test_data to store the rmse values for the train and test data for each model. I also assign half of the data to train_data and half of the data to test_data again.

```
# simulations for sigma = 4

for (i in 1:simulations) {
  study2_data$y = sim_slr(beta_0, beta_1, beta_2, beta_3, beta_4, beta_5, beta_6, sigma[3])
```

```
  # creating an index for half of the data, keeping everything up to this index for train data
  # then removing the train data and keeping the rest for the test data
  train_idx = sample(sample_size/2)
  train_data = study2_data[train_idx,]
  test_data = study2_data[-train_idx,]

  mod1 = lm(y ~ x1, data = train_data)
  mod2 = lm(y ~ x1 + x2, data = train_data)
  mod3 = lm(y ~ x1 + x2 + x3, data = train_data)
  mod4 = lm(y ~ x1 + x2 + x3 + x4, data = train_data)
  mod5 = lm(y ~ x1 + x2 + x3 + x4 + x5, data = train_data)
  mod6 = lm(y ~ x1 + x2 + x3 + x4 + x5 + x6, data = train_data)
  mod7 = lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7, data = train_data)
  mod8 = lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8, data = train_data)
  mod9 = lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9, data = train_data)

  simulation3_RMSE_train_data$mod1[i] = rmse(train_data$y, predict(mod1, train_data))
  simulation3_RMSE_train_data$mod2[i] = rmse(train_data$y, predict(mod2, train_data))
  simulation3_RMSE_train_data$mod3[i] = rmse(train_data$y, predict(mod3, train_data))
  simulation3_RMSE_train_data$mod4[i] = rmse(train_data$y, predict(mod4, train_data))
  simulation3_RMSE_train_data$mod5[i] = rmse(train_data$y, predict(mod5, train_data))
  simulation3_RMSE_train_data$mod6[i] = rmse(train_data$y, predict(mod6, train_data))
  simulation3_RMSE_train_data$mod7[i] = rmse(train_data$y, predict(mod7, train_data))
  simulation3_RMSE_train_data$mod8[i] = rmse(train_data$y, predict(mod8, train_data))
  simulation3_RMSE_train_data$mod9[i] = rmse(train_data$y, predict(mod9, train_data))

  simulation3_RMSE_test_data$mod1[i] = rmse(test_data$y, predict(mod1, test_data))
  simulation3_RMSE_test_data$mod2[i] = rmse(test_data$y, predict(mod2, test_data))
  simulation3_RMSE_test_data$mod3[i] = rmse(test_data$y, predict(mod3, test_data))
  simulation3_RMSE_test_data$mod4[i] = rmse(test_data$y, predict(mod4, test_data))
  simulation3_RMSE_test_data$mod5[i] = rmse(test_data$y, predict(mod5, test_data))
  simulation3_RMSE_test_data$mod6[i] = rmse(test_data$y, predict(mod6, test_data))
  simulation3_RMSE_test_data$mod7[i] = rmse(test_data$y, predict(mod7, test_data))
  simulation3_RMSE_test_data$mod8[i] = rmse(test_data$y, predict(mod8, test_data))
  simulation3_RMSE_test_data$mod9[i] = rmse(test_data$y, predict(mod9, test_data))

}
```

Results

For each value of σ, create 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.

An additional tip:

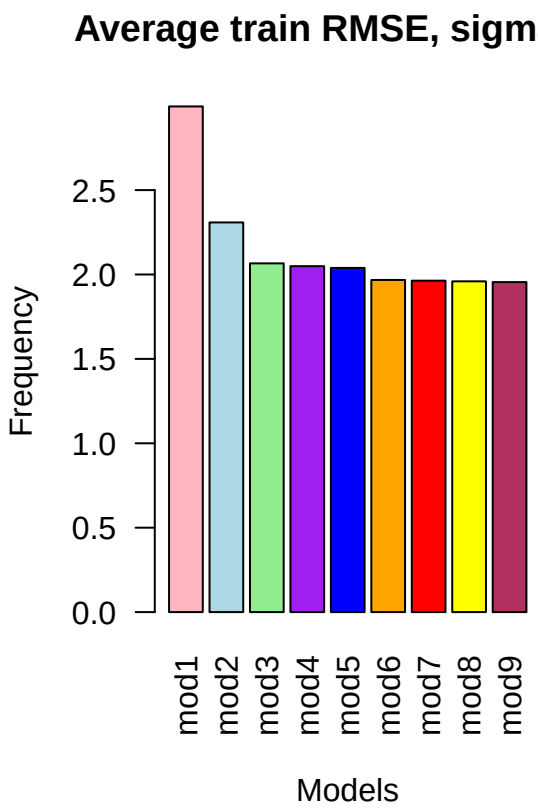
- To address the second discussion topic, consider making a line graph for the RMSE values at each level of σ. Within a single plot for a given σ, one line could correspond to the training data and the other to the test data.

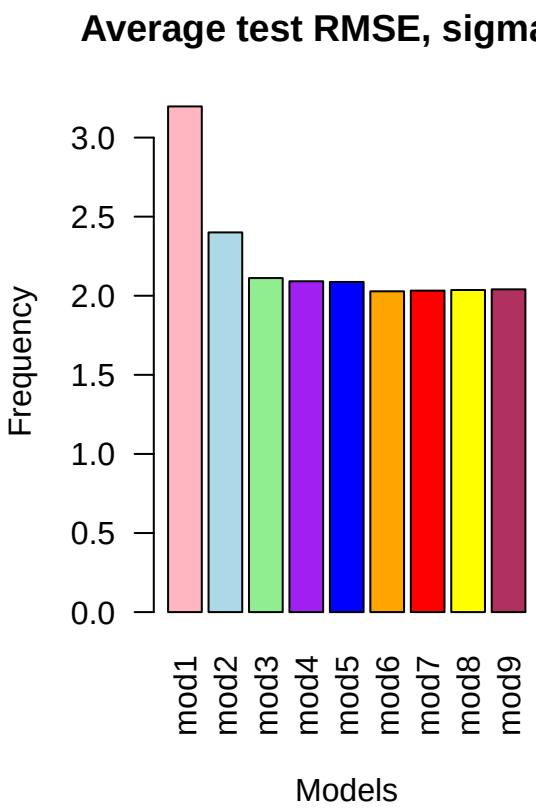
Average Train and Average Test RMSE graphs

Below I display plots that show how average Train RMSE and average Test RMSE change as a function of model size, for each value of σ.

```r
par(mfrow=c(1,2))

# sigma = 1
means_train1 <- apply(simulation1_RMSE_train_data,2,mean)
means_test1 <- apply(simulation1_RMSE_test_data,2,mean)

barplot(means_train1, main = "Average train RMSE, sigma = 1", sub = "Models",
        ylab = "Frequency",col = c("lightpink", "lightblue","lightgreen",
        "purple","blue","orange","red","yellow","maroon"),las = 2)
barplot(means_test1, main = "Average test RMSE, sigma = 1", sub = "Models",
        ylab = "Frequency",col = c("lightpink","lightblue","lightgreen",
        "purple","blue","orange","red","yellow","maroon"),las = 2)
```


```r
#sigma = 2
means_train2 <- apply(simulation2_RMSE_train_data,2,mean)
means_test2 <- apply(simulation2_RMSE_test_data,2,mean)

barplot(means_train2, main = "Average train RMSE, sigma = 2", sub = "Models",
        ylab = "Frequency",col = c("lightpink", "lightblue","lightgreen",
        "purple","blue","orange","red","yellow","maroon"),las = 2)
barplot(means_test2, main = "Average test RMSE, sigma = 2", sub = "Models",
        ylab = "Frequency",col = c("lightpink","lightblue","lightgreen",
        "purple","blue","orange","red","yellow","maroon"),las = 2)
```






```r
#sigma = 4
means_train3 <- apply(simulation3_RMSE_train_data,2,mean)
means_test3 <- apply(simulation3_RMSE_test_data,2,mean)

barplot(means_train3, main = "Average train RMSE, sigma = 4", sub = "Models",
        ylab = "Frequency",col = c("lightpink", "lightblue","lightgreen",
        "purple","blue","orange","red","yellow","maroon"),las = 2)
barplot(means_test3, main = "Average test RMSE, sigma = 4", sub = "Models",
        ylab = "Frequency",col = c("lightpink","lightblue","lightgreen",
        "purple","blue","orange","red","yellow","maroon"),las = 2)
```

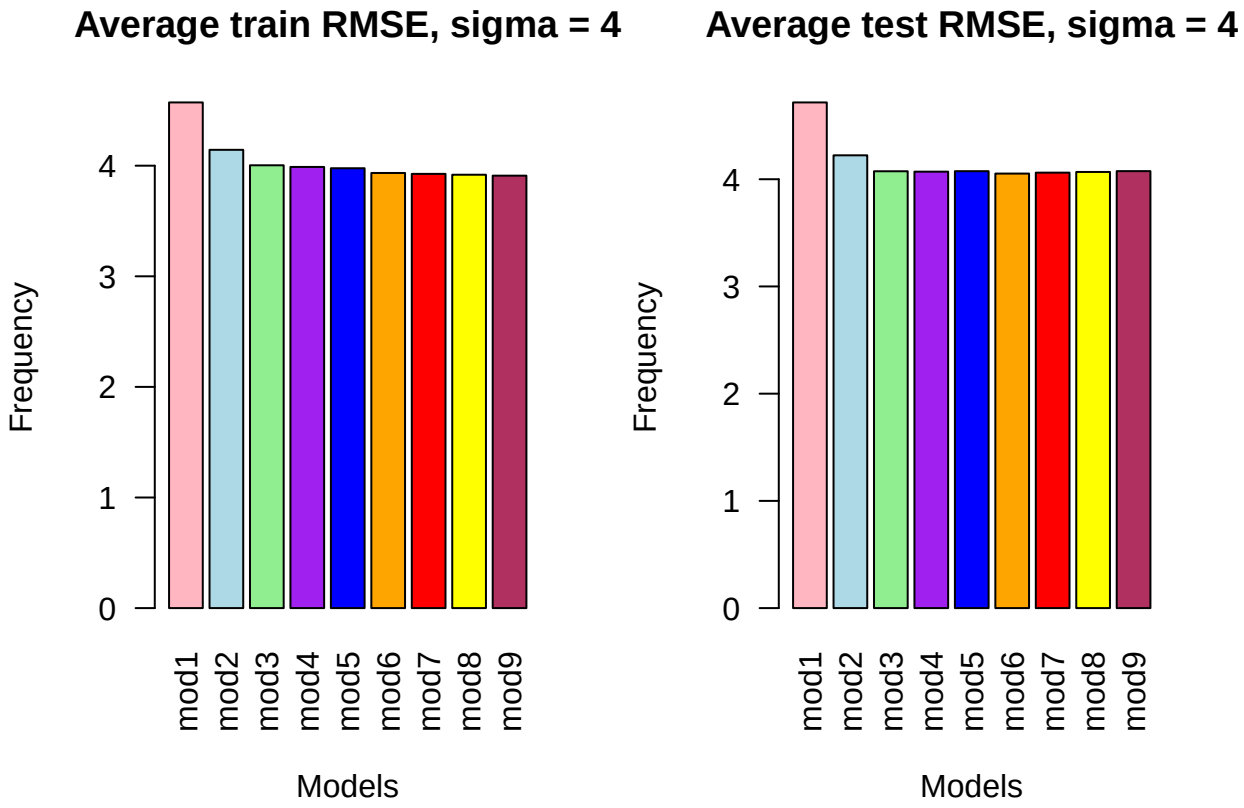


Table of Model Selection Frequency Here I am creating tables that shows the number of times the model of each size was chosen for each value of σ.

```
# Finding the lowest value of RMSE for each row and counting it as the model chosen
sigma_1 = table(factor(colnames(simulation1_RMSE_test_data)[apply(simulation1_RMSE_test_data,
          1, FUN = which.min)], levels = colnames(simulation1_RMSE_test_data)))
sigma_2 = table(factor(colnames(simulation2_RMSE_test_data)[apply(simulation2_RMSE_test_data,
          1, FUN = which.min)], levels = colnames(simulation2_RMSE_test_data)))
sigma_4 = table(factor(colnames(simulation3_RMSE_test_data)[apply(simulation3_RMSE_test_data,
          1, FUN = which.min)], levels = colnames(simulation3_RMSE_test_data)))
mod_names = c('mod1', 'mod2', 'mod3', 'mod4', 'mod5', 'mod6', 'mod7', 'mod8', 'mod9')

knitr::kable(t(cbind(sigma_1,sigma_2,sigma_4)),col.names=mod_names)
```

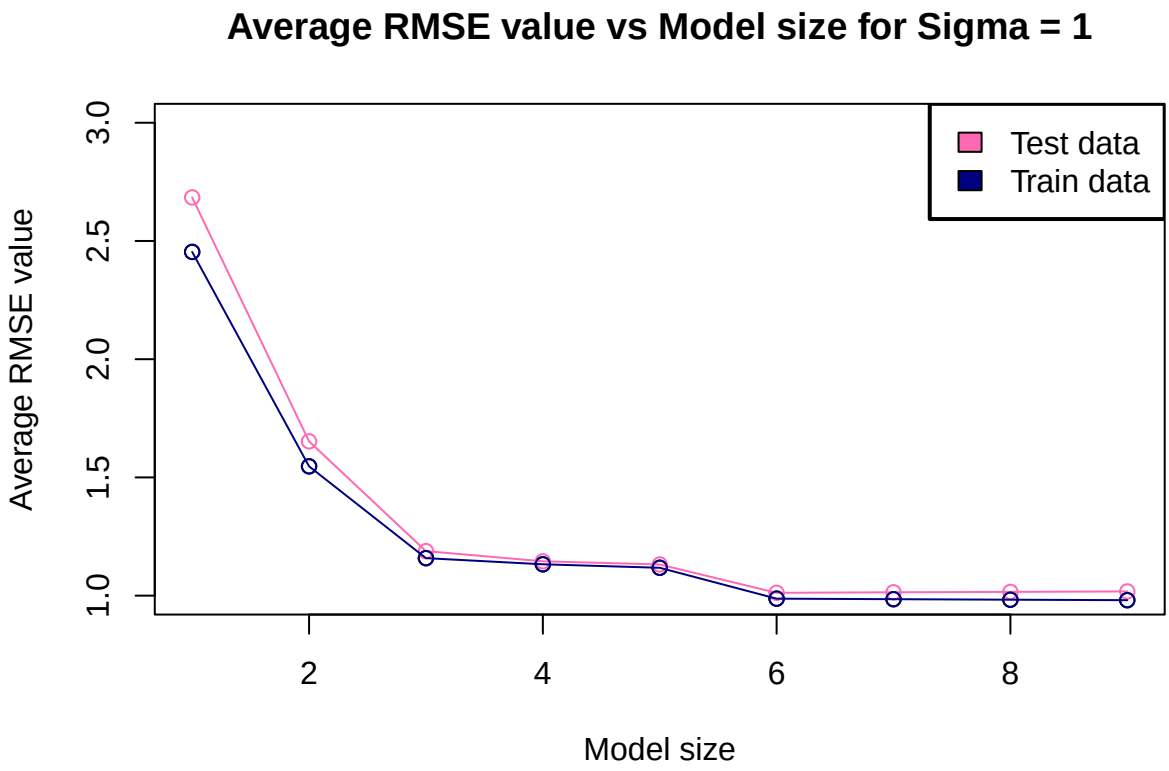
	mod1	mod2	mod3	mod4	mod5	mod6	mod7	mod8	mod9
sigma_1	0	0	0	0	0	537	189	158	116
sigma_2	0	0	13	17	20	499	195	132	124
sigma_4	0	8	156	119	63	336	137	89	92

RMSE values at each level of sigma

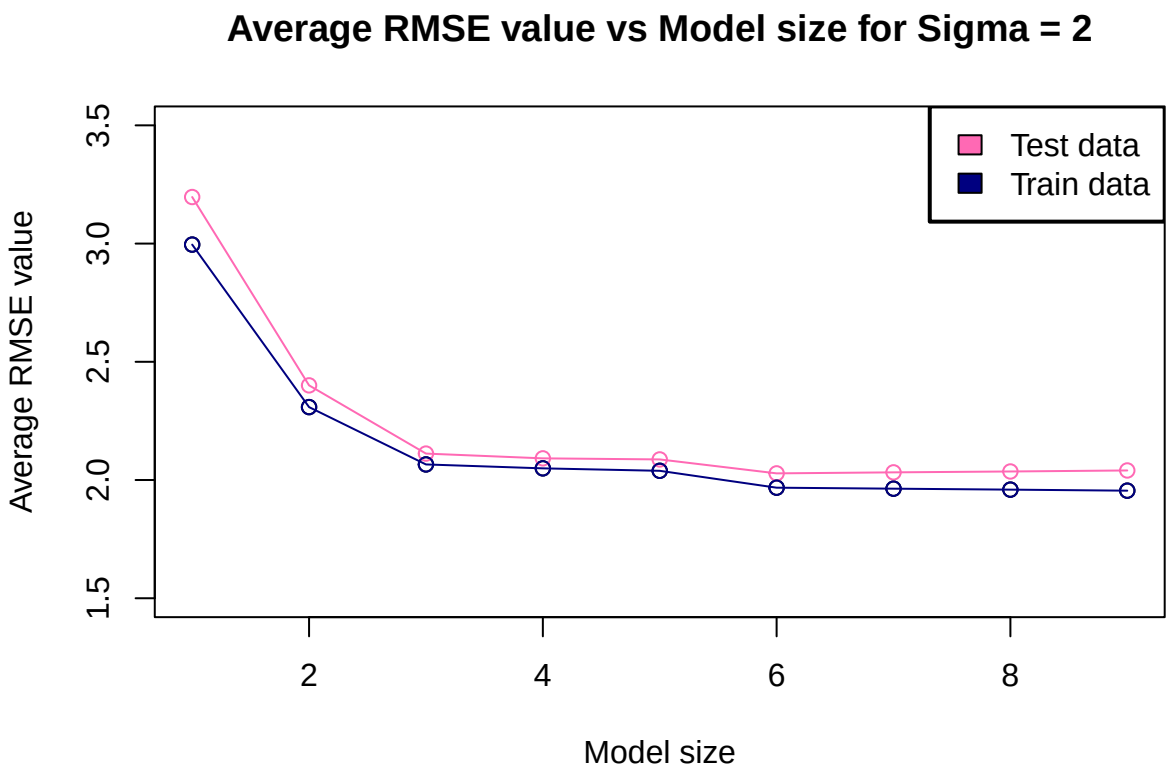
Here I am creating a line graph for the average RMSE values at each level of σ. The graphs display two lines which correspond to the training data and testing data, respectfully.

```
#sigma = 1
plot(means_train1, xlab="Model size", ylab="Average RMSE value",
     main="Average RMSE value vs Model size for Sigma = 1", ylim=c(1,3))
legend(x = "topright", box.lwd = 2, legend=c("Test data", "Train data"),
       fill = c("hotpink","navy"))
```

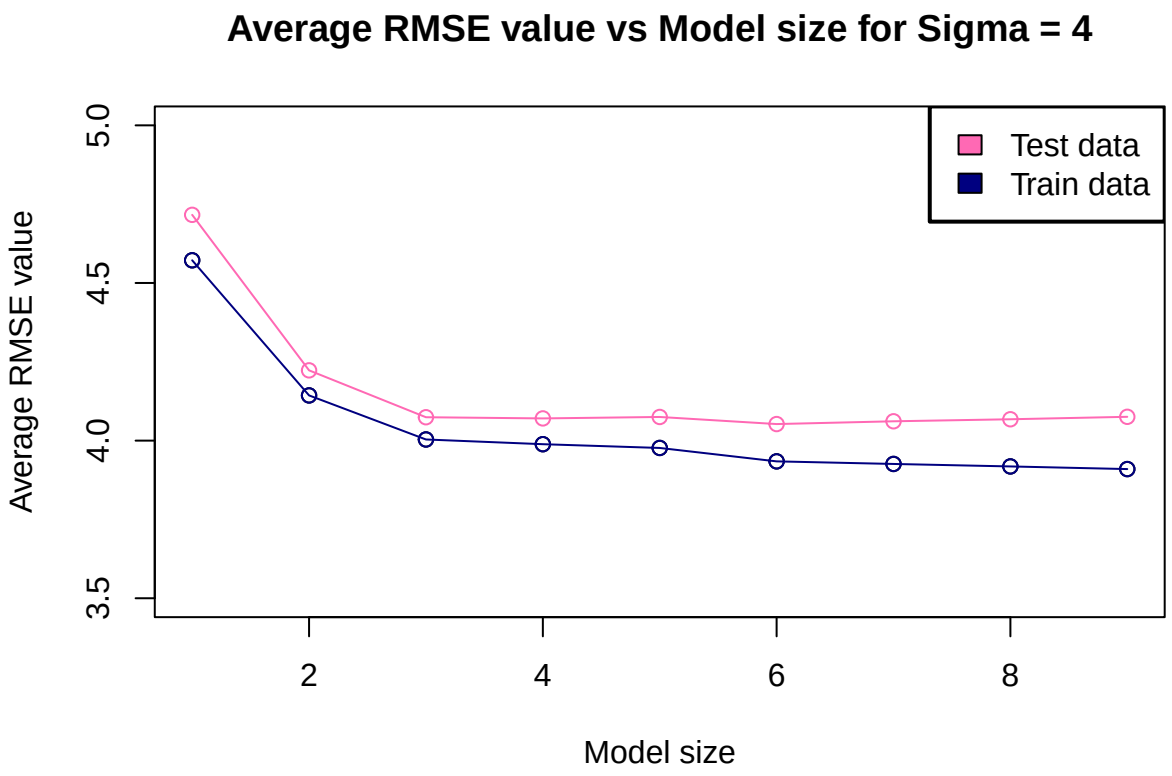
```
lines(means_test1, type ="o", col="hotpink")
lines(means_train1,type ="o", col="navy")
```




```
#sigma = 2
plot(means_train2, xlab="Model size", ylab="Average RMSE value",
     main="Average RMSE value vs Model size for Sigma = 2",ylim=c(1.5,3.5))
legend(x = "topright", box.lwd = 2, legend=c("Test data", "Train data"),
       fill = c("hotpink","navy"))
lines(means_test2, type ="o", col="hotpink")
lines(means_train2,type ="o", col="navy")
```




```
#sigma = 4
plot(means_train3, xlab="Model size", ylab="Average RMSE value",
     main="Average RMSE value vs Model size for Sigma = 4", ylim=c(3.5, 5))
legend(x = "topright", box.lwd = 2, legend=c("Test data", "Train data"),
       fill = c("hotpink","navy"))
lines(means_test3, type ="o", col="hotpink")
lines(means_train3,type ="o", col="navy")
```



Discussion

RMSE refers to the average difference between values predicted by a model and the actual values. It is a estimation on how accurate the model is. In the above simulation, I displayed how RMSE can be used to select the "best" model when it is averaged over many attempts, however this method doesn't always select the correct model. From the "Table of Model Selection Frequency" above, we can see that the majority of the time, model 6 was chosen for all three values of sigma. Even though it was chosen the most, the other models were also picked often times, proving that this selection method is not always accurate. The table also shows us how for all levels of sigma, model 1 with only one predictor was never chosen, telling us this is the worst model. Averaging the RMSE values for the test and train data, we also saw in the barplot and line graphs above, that the average RMSE was often lowest for a model size of 6, but was sometimes also lower for model sizes above 6. We can also note that as model size increases, RMSE mostly decreases. Even though the RMSE might often times be lower for other models, on average, the method selects the true model with 6 parameters that we mentioned in the introduction.

Looking at the level of noise, we see in the graphs above that as we increase σ, the average RMSE values for both train and test data increases as well. Also, with an increase in σ, we see in the table that the variation in the model chosen also increases. As σ got bigger, more model sizes were chosen than before. For example, when $\sigma = 1$, models 6,7,8, and 9 were the only ones that were chosen. When we increased σ to 2, models 4 and 5 were also chosen a few times and when we increased σ to 4, models 2 and 3 were chosen a few times as well. Overall, more noise causes the average RMSE to increase and a higher level of σ means that the correct model 6 is chosen less times than before. This increase in the level of noise makes it harder for the model to predict the values correctly, and makes the chosen model vary.

From this simulation study, we showed that on average, model 6 has the lowest average RMSE value for each level of σ and it is also chosen the most for each level as well. On average, test RMSE is also larger than train RMSE. This difference may be due to the fact that the train data was used to actually build all nine models and the test data is data that the models have not seen before. We can conclude that using RMSE for selection is a valid method, if it is averaged over a large number of simulations.

Simulation Study 3: Power

Introduction

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

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

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

I'll investigate the first three.

To do so I 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, I will let $\beta_0 = 0$, thus β_1 is essentially controlling the amount of "signal." I will then consider different signals, noises, and sample sizes:

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

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

I will also 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, I will simulate from the true model at least 1000 times. Each time, I will perform the significance of the regression test. To estimate the power with these simulations, and some α, I will use

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

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

Methods

```
#setting the seed
birthday = 20020527
set.seed(birthday)
```

Now I will set up the variables that will be used later in this simulation study.

```
sigma = c(1, 2, 4)
sample_size = c(10, 20, 30)
alpha = 0.05

beta_0 = 0
beta_1s = seq(from = -2, to = 2, by = .1)
simulations = 1000

#the number of power iterations should be equal to the num signals * noises * sample sizes
power_iterations = length(beta_1s) * 3 * 3

#creating empty data frame to store power from simulations for each signal, noise, sample size
power_simulations_results = data.frame(sigma = rep(0, power_iterations),
                                       sample_size = rep(0, power_iterations),
                                       beta_1 = rep(0, power_iterations),
                                       power = rep(0, power_iterations))
```

Here I am creating a function called "sim_slr" that will be used to run the simulations. The function takes the beta and sigma values as inputs and returns the y value.

```
# Function to run simulations
sim_slr = function(beta_0, beta_1, x_values, sigma, size) {
  epsilon = rnorm(n = size, mean = 0, sd = sigma)
  y = beta_0 + beta_1*x_values + epsilon
}
```

Now I am running the simulations to find power. I loop through each sigma, sample size and then signal and run 1000 simulations for each one as noted above. I store the results in the data frame created previously and use my sim_slr function accordingly. I test the p value using the alpha value of 0.05 and store the power as the count of rejections over the total number of simulations.

```
count = 1

for (s in sigma) {
  for (n in sample_size) {
    x_values = seq(0, 5, length = n)
    for (beta_1 in beta_1s) {
      power_simulations_results$sigma[count] = s
      power_simulations_results$sample_size[count] = n
      power_simulations_results$beta_1[count] = beta_1
      reject_count = 0
      for (i in 1:simulations) {
```

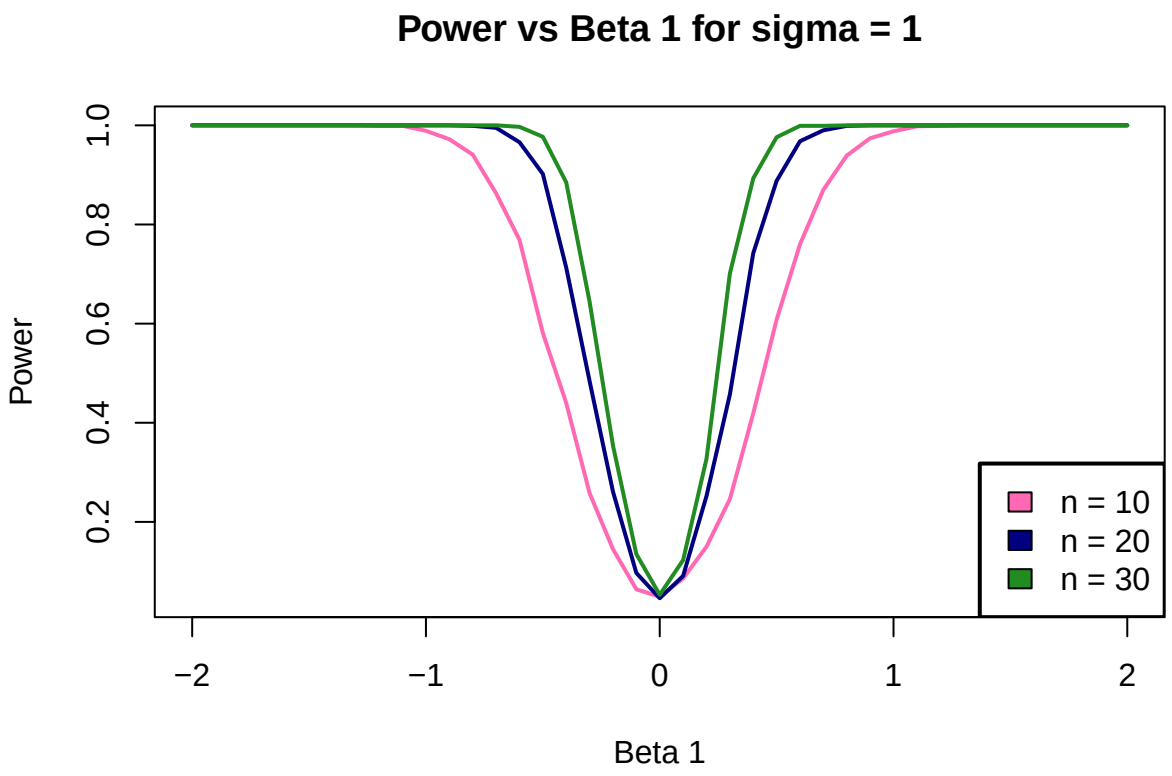
```
        y = sim_slr(beta_0, beta_1, x_values, s, n)
        mod = lm(y ~ x_values)
        p = summary(mod)$coefficients[2,4]
        if (p < alpha) {
          reject_count = reject_count + 1
        }
      }
      power_simulations_results$power[count] = reject_count / simulations
      count = count + 1
    }
  }
}
```

Results

Sigma = 1 graph

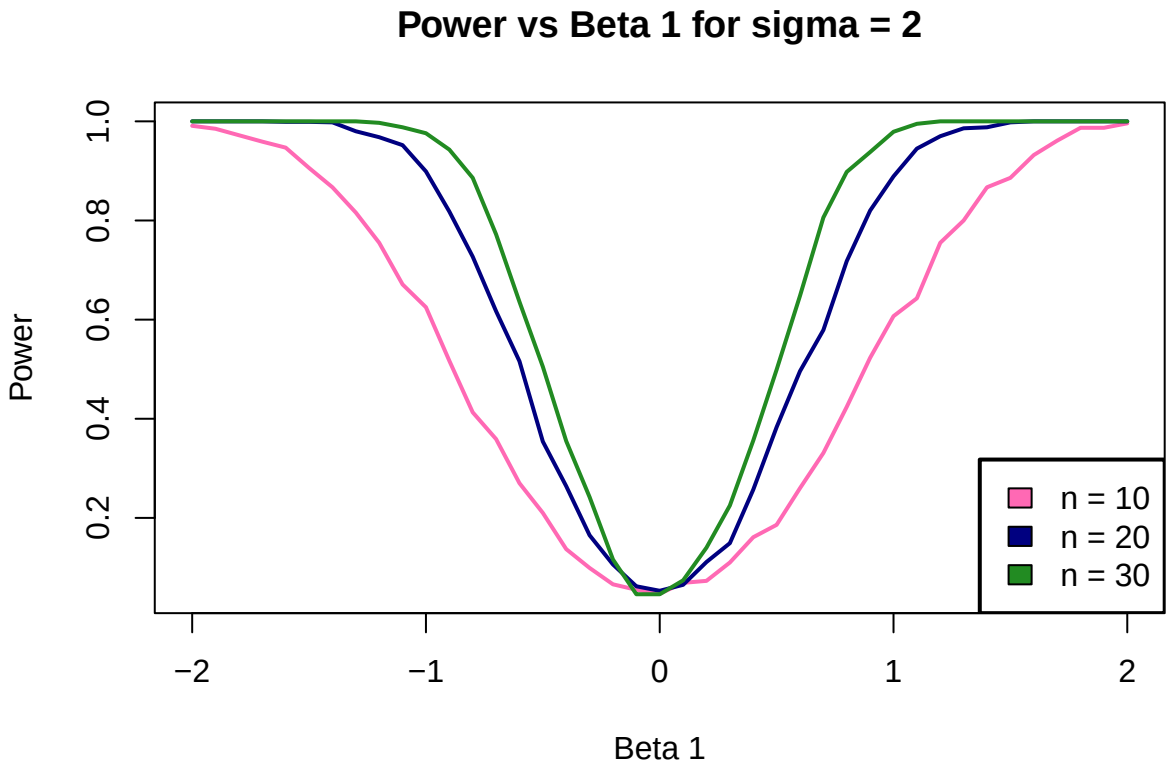
Below I will display a graph for each value of σ, also showing a power curve for each value of n, to show how power is affected by signal strength or β_1 values. Since our β_1 vector contains values in the range from -2 to 2, our power curves will be two-sided.

```
# sigma = 1 graph
sigma1 = power_simulations_results[power_simulations_results$sigma == 1, ]
n10 = sigma1[sigma1$sample_size == 10, ]
n20 = sigma1[sigma1$sample_size == 20, ]
n30 = sigma1[sigma1$sample_size == 30, ]
plot(sigma1$power ~ sigma1$beta_1, type="n", ylab="Power", xlab="Beta 1",
     main="Power vs Beta 1 for sigma = 1")
lines(n10$power ~ n10$beta_1, col='hotpink', lwd=2)
lines(n20$power ~ n20$beta_1, col='navy', lwd=2)
lines(n30$power ~ n30$beta_1, col='forestgreen', lwd=2)
legend(x = "bottomright", box.lwd = 2, legend=c("n = 10", "n = 20", "n = 30"),
       fill = c("hotpink","navy", "forestgreen"))
```



Sigma = 2 graph

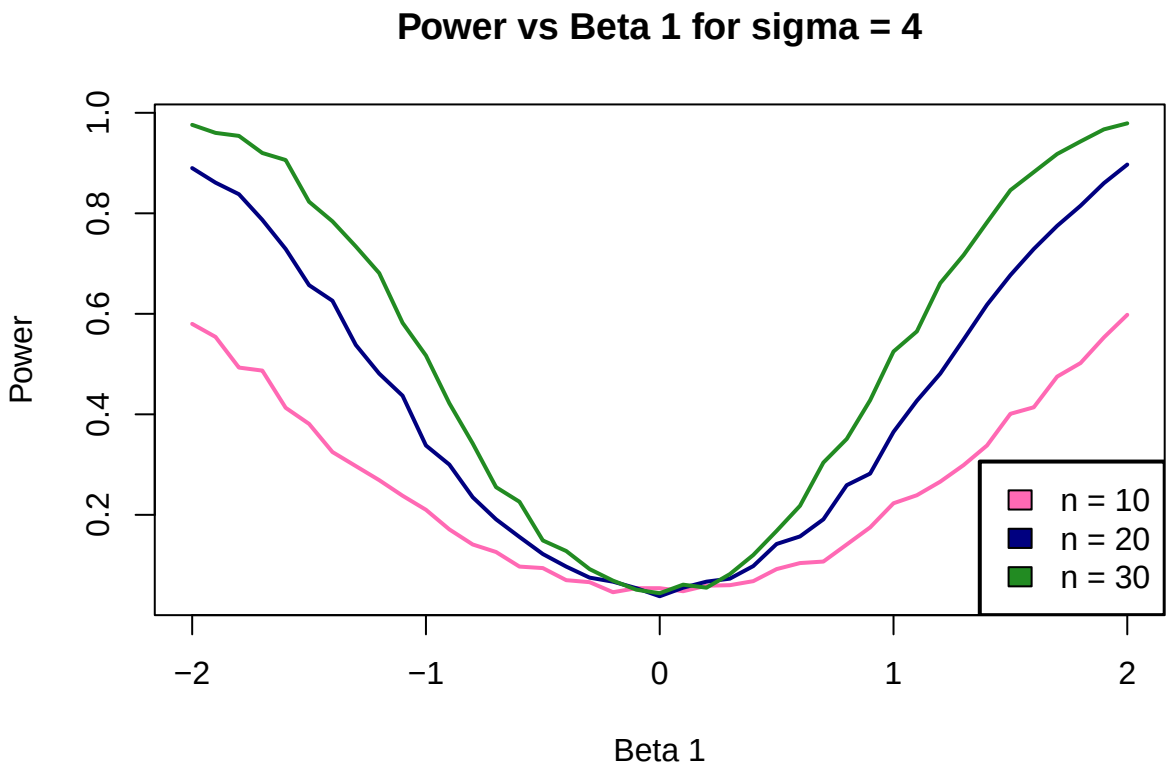
```
# sigma = 2 graph
sigma2 = power_simulations_results[power_simulations_results$sigma == 2, ]
n10 = sigma2[sigma2$sample_size == 10, ]
n20 = sigma2[sigma2$sample_size == 20, ]
n30 = sigma2[sigma2$sample_size == 30, ]
plot(sigma2$power ~ sigma2$beta_1, type="n", ylab="Power", xlab="Beta 1",
     main="Power vs Beta 1 for sigma = 2")
lines(n10$power ~ n10$beta_1, col='hotpink', lwd=2)
lines(n20$power ~ n20$beta_1, col='navy', lwd=2)
lines(n30$power ~ n30$beta_1, col='forestgreen', lwd=2)
legend(x = "bottomright", box.lwd = 2, legend=c("n = 10", "n = 20", "n = 30"),
       fill = c("hotpink","navy", "forestgreen"))
```



###

Sigma = 4 graph

```
# sigma = 4 graph
sigma4 = power_simulations_results[power_simulations_results$sigma == 4, ]
n10 = sigma4[sigma4$sample_size == 10, ]
n20 = sigma4[sigma4$sample_size == 20, ]
n30 = sigma4[sigma4$sample_size == 30, ]
plot(sigma4$power ~ sigma4$beta_1, type="n", ylab="Power", xlab="Beta 1",
     main="Power vs Beta 1 for sigma = 4")
lines(n10$power ~ n10$beta_1, col='hotpink', lwd=2)
lines(n20$power ~ n20$beta_1, col='navy', lwd=2)
lines(n30$power ~ n30$beta_1, col='forestgreen', lwd=2)
legend(x = "bottomright", box.lwd = 2, legend=c("n = 10", "n = 20", "n = 30"),
       fill = c("hotpink","navy", "forestgreen"))
```



Discussion

In this simulation study, we examined how power refers to probability that a signal of a particular strength will be detected. We know that aspects like sample size, signal strength, noise level, and significance level affect the power of a test. Since our alpha was fixed constant at level 0.05, we will discuss how the other three values affect power. Also we can note that the graphs above level out around a value of power = 1 due to power being a probability value that will range from 0 to 1.

Starting with n, we can see from all 3 graphs in the result section that as we increase n, power also increases. Looking at each graph, as we increase n, the power at a given β_1 value will be larger for a larger n. This means that a increase in sample size will cause a increase in the "probability that a signal of a particular strength will be detected", proving that there is a positive relationship between n and power. Next, looking at β_1, we can again use the 3 graphs from the result section to see that power seems to increase as β_1 get farther from zero. At $\beta_1 = 0$, power is nearly 0 but as we move to the right or left towards our range from -2 to 2, power increases. This aligns with our null hypothesis of $\beta_1 = 0$ as power increases when $\beta_1 \neq 0$. Finally, we can see from each separate graph how σ affects the power. As σ is increased, the power at a given β_1 value will be lower. For example the power value at -1 or 1 in the above graphs, will be largest in the first graph of $\sigma = 1$ vs the third graph of $\sigma = 4$. Furthermore, in the smaller value of sigma, we also see the power curve reach the limit value of 1 whereas in the larger values of sigma, power never reaches this limit in the range -2 to 2.

Overall, we can conclude that larger n values, β_1 values that are not zero, and smaller σ values will cause the power to be larger. We analyzed this with the 1000 simulations done above, but in order to see if this number of simulations is sufficient, we can also run more simulations. When I tested these graphs with 2000 simulations, I noticed that the power curves did not change much besides being smoother. The graphs and curves still showed the same results as we discussed. Even though more simulations would definitely cause the power curves to smoothen out, a larger number of simulations would not be efficient in terms of running time. We can therefore say that 1000 simulations is sufficient as increasing the value to 2000 simulations, did not create a difference in our final conclusions in the relationships.