Midterm Assignment: Simulation Project

Scott Girten
NetID: sgirten2

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

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

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, use 2000 simulations. For each simulation, fit a regression model of the same form used to perform the simulation.

Methods

I begin this first simulation study by setting a seed for functions that will generate randomness and reading a data file that contains data for 25 observations that will serve as the values for the predictors of both the significant and non-significant models.

```
library(tidyverse)

# Set seed
birthday = 19790221
set.seed(birthday)

# Read in data for predictors
sim1_data = read_csv('study_1.csv')
```

Simulation

The first step in obtaining the simulated data for this study is to create a function that will perform the simulation. The function will accept arguments that will allow for elements of the regression model or simulation to be modified, making subsequent portions of this study more streamlined .

```
# Create function to perform simulation - only argument to the function is sig
ma (noise)

simulation_1 = function(data, beta_0, beta_1, beta_2, beta_3, sigma, sample_si
ze, iterations, model_name){

  # create tables to hold results of study and be used for output of the funct
ion
  # Initialize columns used in results tables to 0
  f_stat    = rep(0, iterations)
  p_value   = rep(0, iterations)
  r_squared = rep(0, iterations)

  # Result table of simulation, will be used as output of the function
  results = tibble(model_type = model_name, sigma = sigma, f_stat, p_value, r_
squared)

  for(i in 1:iterations){
    # Error for model
    eps = rnorm(sample_size, mean = 0, sd = sigma)

    # Calculate y-value from known model
    sim_data = data %>%
    mutate(y = beta_0 + beta_1 * x1 + beta_2 * x2 + beta_3 * x3 + eps)

    # fit model from simulated data
    fit = lm(y ~ ., data = sim_data)

    # calculate p-value for F-statistic to store in results
    f = summary(fit)$fstatistic
    p_val = pf(f[1], f[2], f[3], lower.tail = FALSE)

    # Add results to table
    results$f_stat[i]    = summary(fit)$fstatistic[1]
    results$r_squared[i] = summary(fit)$r.squared[1]
    results$p_value[i]   = p_val

  }
  return(results)

}
```

Now the function `simulation_1` can be utilized to carry out the 6 different combinations of models for this study. First, the sample size and number of iterations are defined. Next the significant models are simulated and then finally the non-significant models are simulated. The beta parameters for both significant and non-significant models are passed into each function as arguments.

```
# Function arguments
sample_size = 25
iterations = 2000

# Significant models
b0 = 3
b1 = b2 = b3 = 1

sig_1 =  simulation_1(data = sim1_data, beta_0 = b0, beta_1 = b1, beta_2 = b2,
beta_3 = b3,
                      sigma = 1, sample_size = sample_size, iterations = itera
tions,
                      model_name = 'Significant')

sig_5 =  simulation_1(data = sim1_data, beta_0 = b0, beta_1 = b1, beta_2 = b2,
beta_3 = b3,
                      sigma = 5, sample_size = sample_size, iterations = itera
tions,
                      model_name = 'Significant')

sig_10 = simulation_1(data = sim1_data, beta_0 = b0, beta_1 = b1, beta_2 = b2,
beta_3 = b3,
                      sigma = 10, sample_size = sample_size, iterations = iter
ations,
                      model_name = 'Significant')



# Non-significant models
b0 = 3
b1 = b2 = b3 = 0

non_sig_1 =  simulation_1(data = sim1_data, beta_0 = 3, beta_1 = b1, beta_2 =
b2, beta_3 = b3,
                          sigma = 1, sample_size = sample_size, iterations = i
terations,
                          model_name = 'Non-Significant')

non_sig_5 =  simulation_1(data = sim1_data, beta_0 = 3, beta_1 = b1, beta_2 =
b2, beta_3 = b3,
                          sigma = 5, sample_size = sample_size, iterations = i
terations,
                          model_name = 'Non-Significant')

non_sig_10 = simulation_1(data = sim1_data, beta_0 = 3, beta_1 = b1, beta_2 =
b2, beta_3 = b3,
                          sigma = 10, sample_size = sample_size, iterations =
iterations,
                          model_name = 'Non-Significant')
```

After the results of the 6 different simulations produced, they are combined into one data set for analysis.

```
# Bind results of each simulation into one data frame
sim1_study = sig_1 %>%
  bind_rows(sig_5) %>%
  bind_rows(sig_10) %>%
  bind_rows(non_sig_1) %>%
  bind_rows(non_sig_5) %>%
  bind_rows(non_sig_10)
```

Visuals Non-Significant Models

The next 3 sections contain code blocks for producing visualizations for the F-Statistic distribution, p-value and R^2 for the non-significant models.

F-Statistic Distribution

```
library(patchwork)
library(latex2exp)

# plot distributions of the F-statistic for non-significant models
n = 2000
p = 4

# non-significant model and sigma  = 1
df_ns1 = sim1_study %>%
  filter(model_type == 'Non-Significant',
         sigma == 1)

p_ns1 = ggplot(df_ns1, aes(x = f_stat)) +
  geom_histogram(aes(y = ..density..),  fill = "#d95f02", color = "#d95f02", a
lpha = 0.5) +
  geom_function(fun = 'df', geom = 'line', size = 1, args = list(df1 = p - 1,
df2 = n - p)) +
  theme_bw() +
  labs(x = NULL,
       y = NULL,
       title = TeX(r'($\sigma$ = 1)'))

# non-significant model and sigma  = 5
df_ns5 = sim1_study %>%
  filter(model_type == 'Non-Significant',
         sigma == 5)

p_ns5 = ggplot(df_ns5, aes(x = f_stat)) +
  geom_histogram(aes(y = ..density..),  fill = "#d95f02", color = "#d95f02", a
lpha = 0.5) +
  geom_function(fun = 'df', geom = 'line', size = 1, args = list(df1 = p - 1,
df2 = n - p)) +
  theme_bw() +
  labs(x = NULL,
       y = NULL,
       title = TeX(r'($\sigma$ = 5)'))

# non-significant model and sigma  = 10
df_ns10 = sim1_study %>%
  filter(model_type == 'Non-Significant',
         sigma == 10)

p_ns10 = ggplot(df_ns10, aes(x = f_stat)) +
  geom_histogram(aes(y = ..density..),  fill = "#d95f02", color = "#d95f02", a
lpha = 0.5) +
  geom_function(fun = 'df', geom = 'line', size = 1, args = list(df1 = p - 1,
df2 = n - p)) +
  theme_bw() +
  labs(x = NULL,
       y = NULL,
       title = TeX(r'($\sigma$ = 10)'))


# put the 3 non-significant plots into one horizontal plot
p_ns = p_ns1 + p_ns5 + p_ns10
p_ns_fdist = p_ns + plot_annotation(title = TeX(r'(Distribution of F-Statistic
(Non-Significant Models))'))
```

p-value Distribution

```
# Non-significant p-value, sigma = 1
p_ns1_pval = ggplot(df_ns1, aes(x = p_value)) +
  geom_histogram(aes(y = ..density..),  fill = "#66c2a5", color = "#66c2a5", a
lpha = 0.5) +
  theme_bw() +
  labs(x = NULL,
       y = NULL,
       title = TeX(r'($\sigma$ = 1)'))

# Non-significant p-value, sigma = 5
p_ns5_pval = ggplot(df_ns5, aes(x = p_value)) +
  geom_histogram(aes(y = ..density..),  fill = "#66c2a5", color = "#66c2a5", a
lpha = 0.5) +
  theme_bw() +
  labs(x = NULL,
       y = NULL,
       title = TeX(r'($\sigma$ = 5)'))

# Non-significant p-value, sigma = 10
p_ns10_pval = ggplot(df_ns10, aes(x = p_value)) +
  geom_histogram(aes(y = ..density..),  fill = "#66c2a5", color = "#66c2a5", a
lpha = 0.5) +
  theme_bw() +
  labs(x = NULL,
       y = NULL,
       title = TeX(r'($\sigma$ = 10)'))

# Combine 3 individual plots into a horizontal plot
p_ns_pval = p_ns1_pval + p_ns5_pval + p_ns10_pval
p_ns_pval = p_ns_pval + plot_annotation(title=  TeX(r'(Distribution of $p$-val
ues (Non-Significant Models))'))
```

R-squared distribution

```
# Non-significant r-squared, sigma = 1
p_ns1_rsqre = ggplot(df_ns1, aes(x = r_squared)) +
  geom_histogram(aes(y = ..density..),  fill = "#e78ac3", color = "#e78ac3", a
lpha = 0.5) +
  theme_bw() +
  labs(x = NULL,
       y = NULL,
       title = TeX(r'($\sigma$ = 1)'))

# Non-significant r-squared, sigma = 5
p_ns5_rsqre = ggplot(df_ns5, aes(x = r_squared)) +
  geom_histogram(aes(y = ..density..),  fill = "#e78ac3", color = "#e78ac3", a
lpha = 0.5) +
  theme_bw() +
  labs(x = NULL,
       y = NULL,
       title = TeX(r'($\sigma$ = 5)'))

# Non-significant r-squared, sigma = 10
p_ns10_rsqre = ggplot(df_ns10, aes(x = r_squared)) +
  geom_histogram(aes(y = ..density..),  fill = "#e78ac3", color = "#e78ac3", a
lpha = 0.5) +
  theme_bw() +
  labs(x = NULL,
       y = NULL,
       title = TeX(r'($\sigma$ = 10)'))

# Combine 3 individual plots into a horizontal plot
p_ns_rsqre = p_ns1_rsqre + p_ns5_rsqre + p_ns10_rsqre
p_ns_rsqre = p_ns_rsqre + plot_annotation(TeX(r'(Distribution of $R^2$ (Non-Si
gnificant Models))'))
```

Visuals Significant Models

Similar to the 3 preceding code blocks, the following 3 sections contain code blocks for producing visualizations for the F-Statistic distribution, p-value and $R^2$ for the significant models.

F-Statistic Distribution

```
# significant model and F-statistic, sigma  = 1
df_s1 = sim1_study %>%
  filter(model_type == 'Significant',
         sigma == 1)

p_s1 = ggplot(df_s1, aes(x = f_stat)) +
  geom_histogram(aes(y = ..density..),  fill = "#d95f02", color = "#d95f02", a
lpha = 0.5) +
  geom_function(fun = 'df', geom = 'line', size = 1, args = list(df1 = p - 1,
df2 = n - p)) +
  #annotate('text', x = max(df_s1$f_stat) * 0.75, y = 0.2, label = 'Significan
t Model\nSigma = 1', size = 6) +
  scale_x_continuous(limits = c(0,152)) +
  theme_bw() +
  labs(x = NULL,
       y = NULL,
       title = TeX(r'($\sigma$ = 1)'))


# significant model and F-statistic, sigma  = 5
df_s5 = sim1_study %>%
  filter(model_type == 'Significant',
         sigma == 5)

p_s5 = ggplot(df_s5, aes(x = f_stat)) +
  geom_histogram(aes(y = ..density..),  fill = "#d95f02", color = "#d95f02", a
lpha = 0.5) +
  geom_function(fun = 'df', geom = 'line', size = 1, args = list(df1 = p - 1,
df2 = n - p)) +
  #annotate('text', x = max(df_s5$f_stat) * 0.75, y = 0.55, label = 'Significa
nt Model\nSigma = 5', size = 6) +
  #scale_x_continuous(limits = c(0,152)) +
  theme_bw() +
  labs(x = NULL,
       y = NULL,
       title = TeX(r'($\sigma$ = 5)'))

# significant model and F-statistic, sigma  = 10
df_s10 = sim1_study %>%
  filter(model_type == 'Significant',
         sigma == 10)

p_s10 = ggplot(df_s10, aes(x = f_stat)) +
  geom_histogram(aes(y = ..density..),  fill = "#d95f02", color = "#d95f02", a
lpha = 0.5) +
  geom_function(fun = 'df', geom = 'line', size = 1, args = list(df1 = p - 1,
df2 = n - p)) +
  #annotate('text', x = max(df_s10$f_stat) * 0.75, y = 0.55, label = 'Signific
ant Model\nSigma = 10', size = 6) +
  #scale_x_continuous(limits = c(0,152)) +
  theme_bw() +
  labs(x = NULL,
       y = NULL,
       title = TeX(r'($\sigma$ = 10)'))

# Combine 3 individual plots into a horizontal plot
p_s = p_s1 + p_s5 + p_s10
p_s_fdist = p_s + plot_annotation(title = 'Distribution of F-Statistic (Signif
icant Models)')
```

p-value Distribution

```
# significant model and p-value, sigma  = 1
p_s1_pval = ggplot(df_s1, aes(x = p_value)) +
  geom_histogram(aes(y = ..density..),  fill = "#66c2a5", color = "#66c2a5", a
lpha = 0.5) +
  theme_bw() +
  labs(x = NULL,
       y = NULL,
       title = TeX(r'($\sigma$ = 1)'))

# significant model and p-value, sigma  = 5
p_s5_pval = ggplot(df_s5, aes(x = p_value)) +
  geom_histogram(aes(y = ..density..),  fill = "#66c2a5", color = "#66c2a5", a
lpha = 0.5) +
  theme_bw() +
  labs(x = NULL,
       y = NULL,
       title = TeX(r'($\sigma$ = 5)'))

# significant model and p-value, sigma  = 10
p_s10_pval = ggplot(df_s10, aes(x = p_value)) +
  geom_histogram(aes(y = ..density..),  fill = "#66c2a5", color = "#66c2a5", a
lpha = 0.5) +
  theme_bw() +
  labs(x = NULL,
       y = NULL,
       title = TeX(r'($\sigma$ = 10)'))

# Combine 3 individual plots into a horizontal plot
p_s_pval = p_s1_pval + p_s5_pval + p_s10_pval
p_s_pval = p_s_pval + plot_annotation('Distribution of p-values (Significant M
odels)')
```

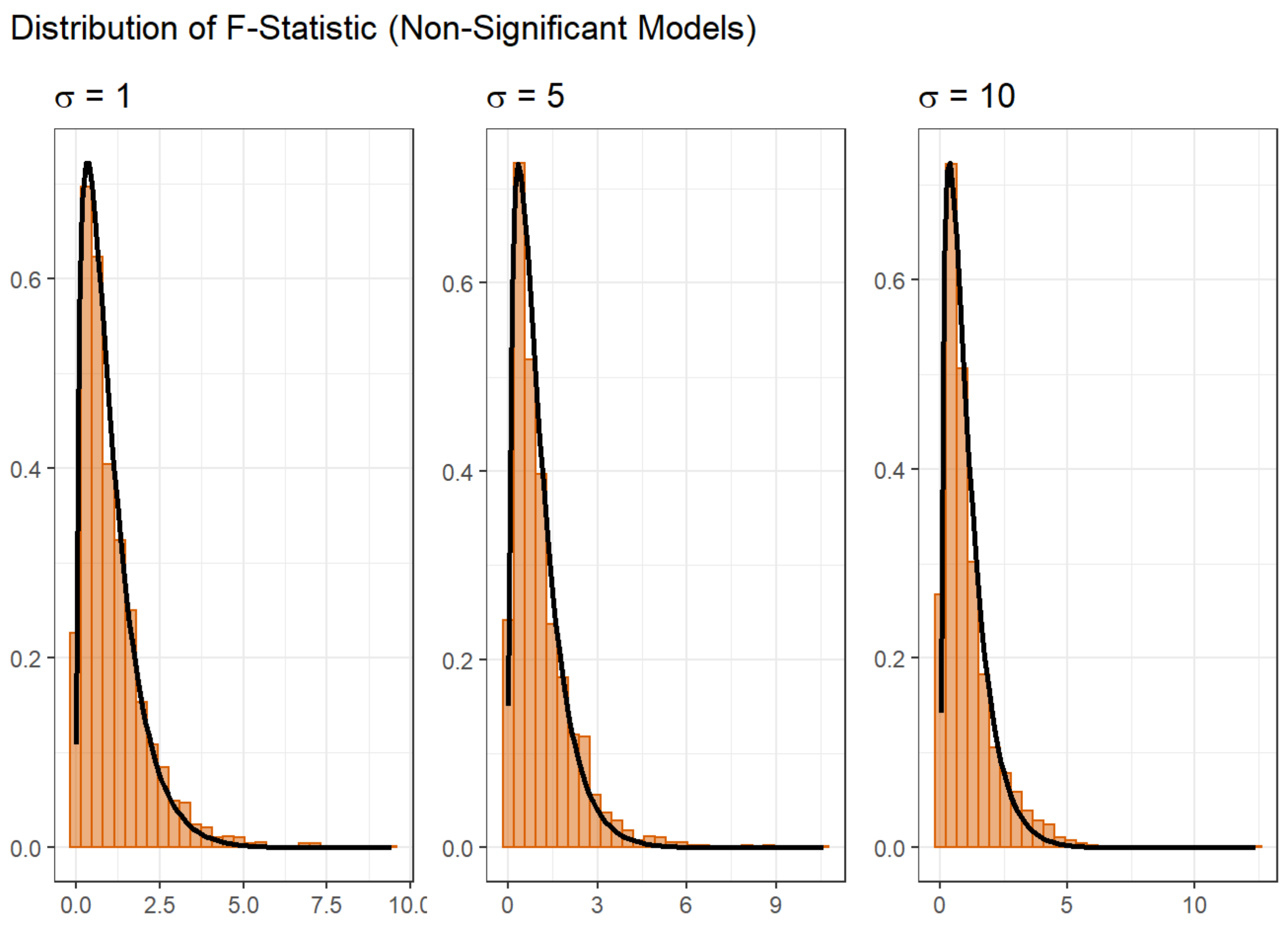
R-squared distribution

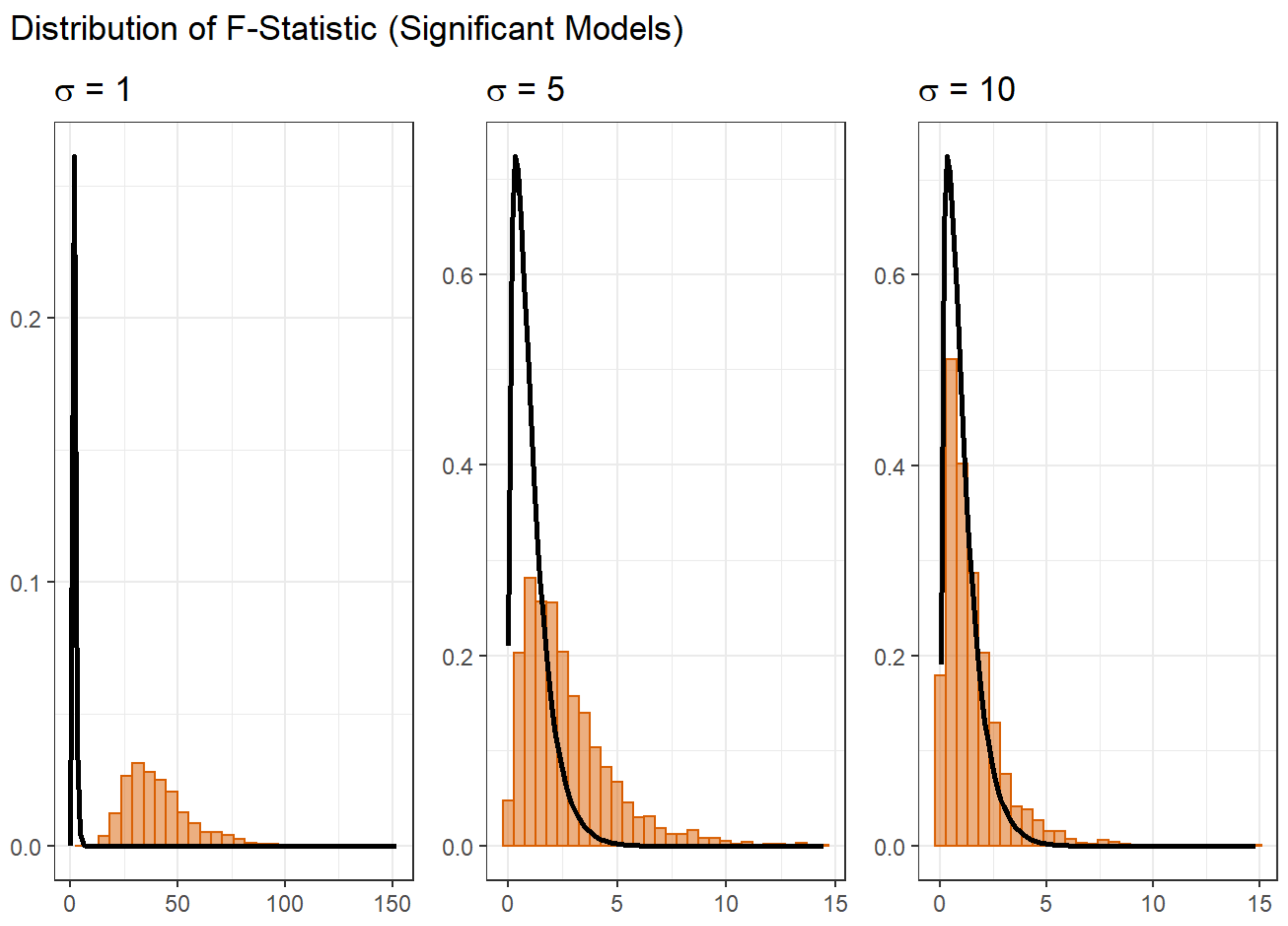
```
# Non-significant r-squared, sigma = 1
p_s1_rsqre = ggplot(df_s1, aes(x = r_squared)) +
  geom_histogram(aes(y = ..density..),  fill = "#e78ac3", color = "#e78ac3", a
lpha = 0.5) +
  theme_bw() +
  labs(x = NULL,
       y = NULL,
       title = TeX(r'($\sigma$ = 1)'))

# Non-significant r-squared, sigma = 5
p_s5_rsqre = ggplot(df_s5, aes(x = r_squared)) +
  geom_histogram(aes(y = ..density..),  fill = "#e78ac3", color = "#e78ac3", a
lpha = 0.5) +
  theme_bw() +
  labs(x = NULL,
       y = NULL,
       title = TeX(r'($\sigma$ = 5)'))

# Non-significant r-squared, sigma = 10
p_s10_rsqre = ggplot(df_s10, aes(x = r_squared)) +
  geom_histogram(aes(y = ..density..),  fill = "#e78ac3", color = "#e78ac3", a
lpha = 0.5) +
  theme_bw() +
  labs(x = NULL,
       y = NULL,
       title = TeX(r'($\sigma$ = 10)'))

# Combine 3 individual plots into a horizontal plot
p_s_rsqre = p_s1_rsqre + p_s5_rsqre + p_s10_rsqre
p_s_rsqre = p_s_rsqre + plot_annotation(title = TeX(r'(Distribution of $R^2$
(Significant Models))'))

#rm(list = ls())
```

Results

F-Distribution

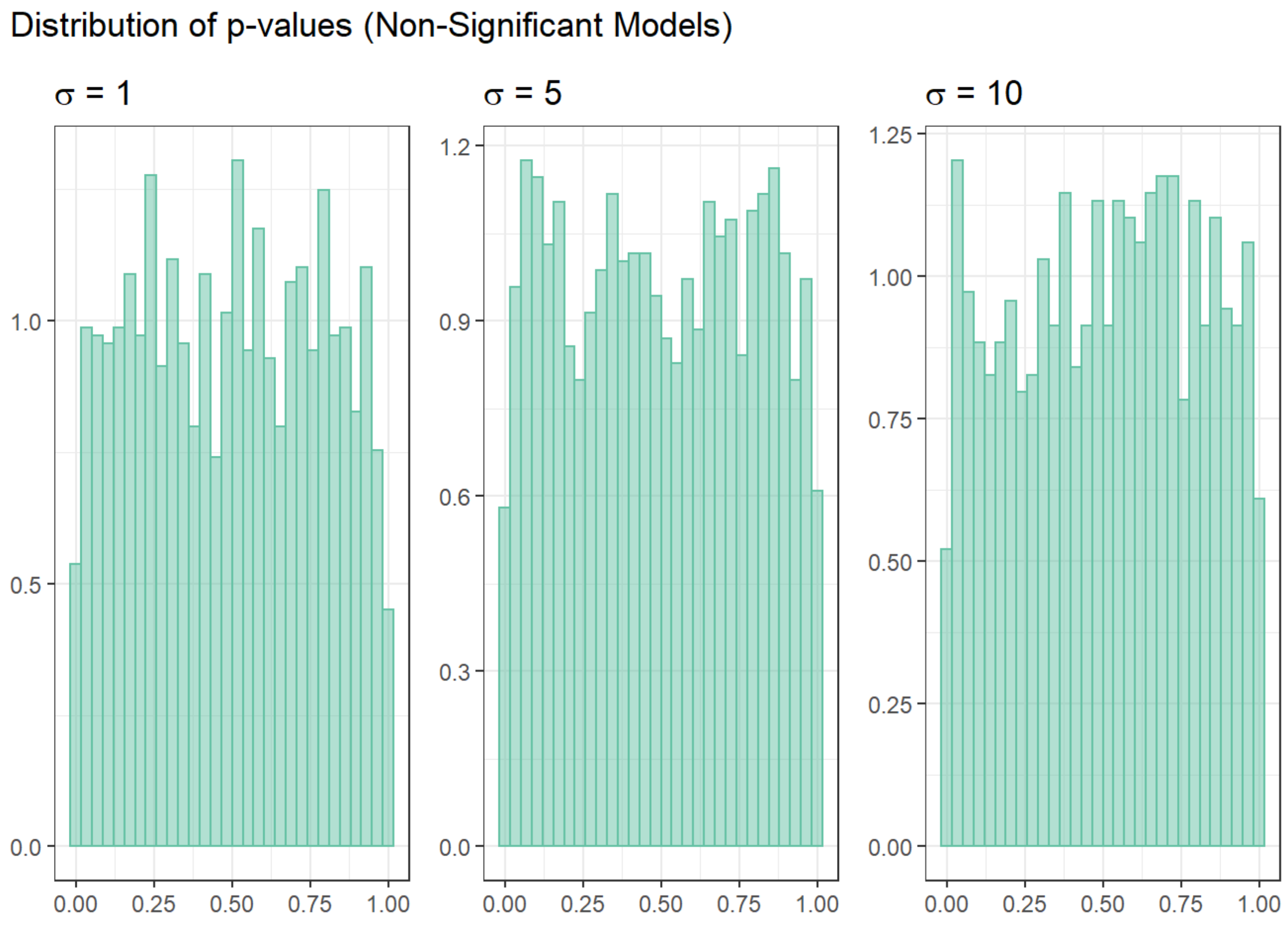
First, the distributions of the F-Statistic for both the non-significant and significant models are presented to visualize the relationship between the F-distribution and changes in the level of $\sigma$.

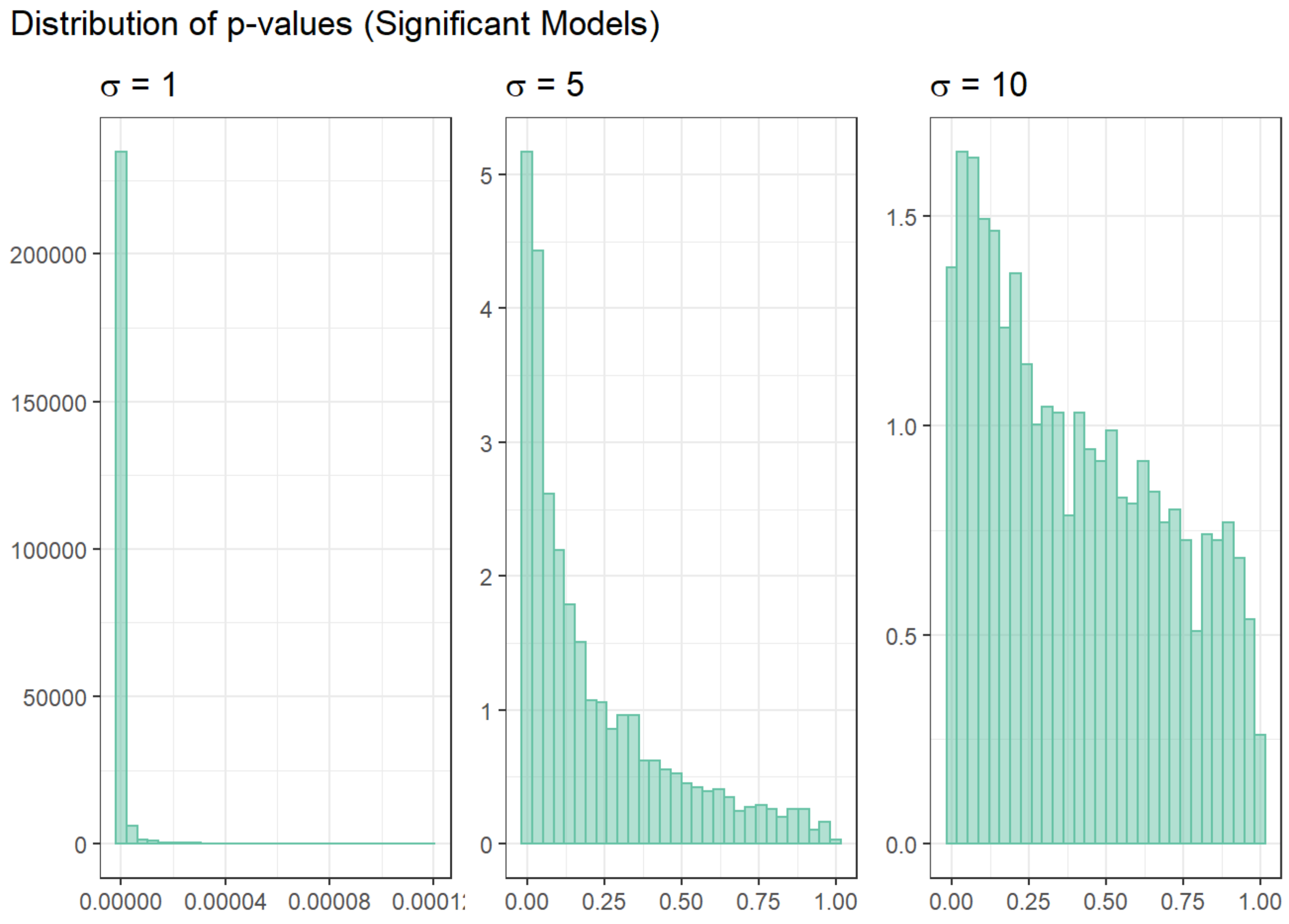

Distribution of F-Statistic (Non-Significant Models)
σ = 1
σ = 5
σ = 10
0.0
0.2
0.4
0.6
0.0
2.5
5.0
7.5
10.0
0
3
6
9
0
5
10


Distribution of F-Statistic (Significant Models)
σ = 1
σ = 5
σ = 10
0.0
0.1
0.2
0
50
100
150
0.0
0.2
0.4
0.6
0
5
10
15

p-value Distribution

Second, a visualization of the distribution of the p-value of the F-statistic and how that distribution changes as σ changes and also the difference in non-significant and significant models.

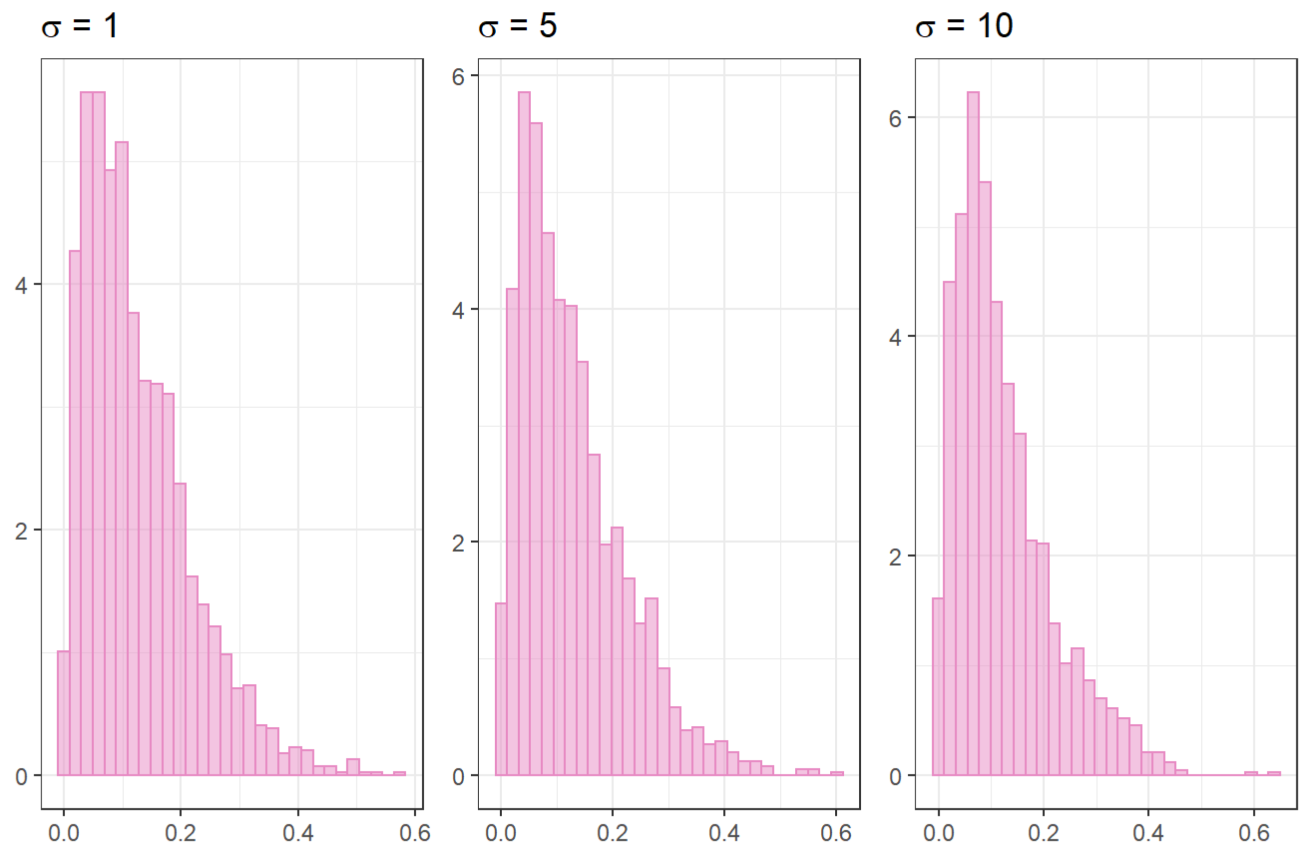

Distribution of p-values (Non-Significant Models)
σ = 1
σ = 5
σ = 10
0.0
0.5
1.0
0.3
0.6
0.9
1.2
0.00
0.25
0.50
0.75
1.00
1.25
0.00
0.25
0.50
0.75
1.00


Distribution of p-values (Significant Models)
σ = 1
σ = 5
σ = 10
0
50000
100000
150000
200000
0.00000
0.00004
0.00008
0.0001
0
1
2
3
4
5
0.0
0.5
1.0
1.5
0.00
0.25
0.50
0.75
1.00

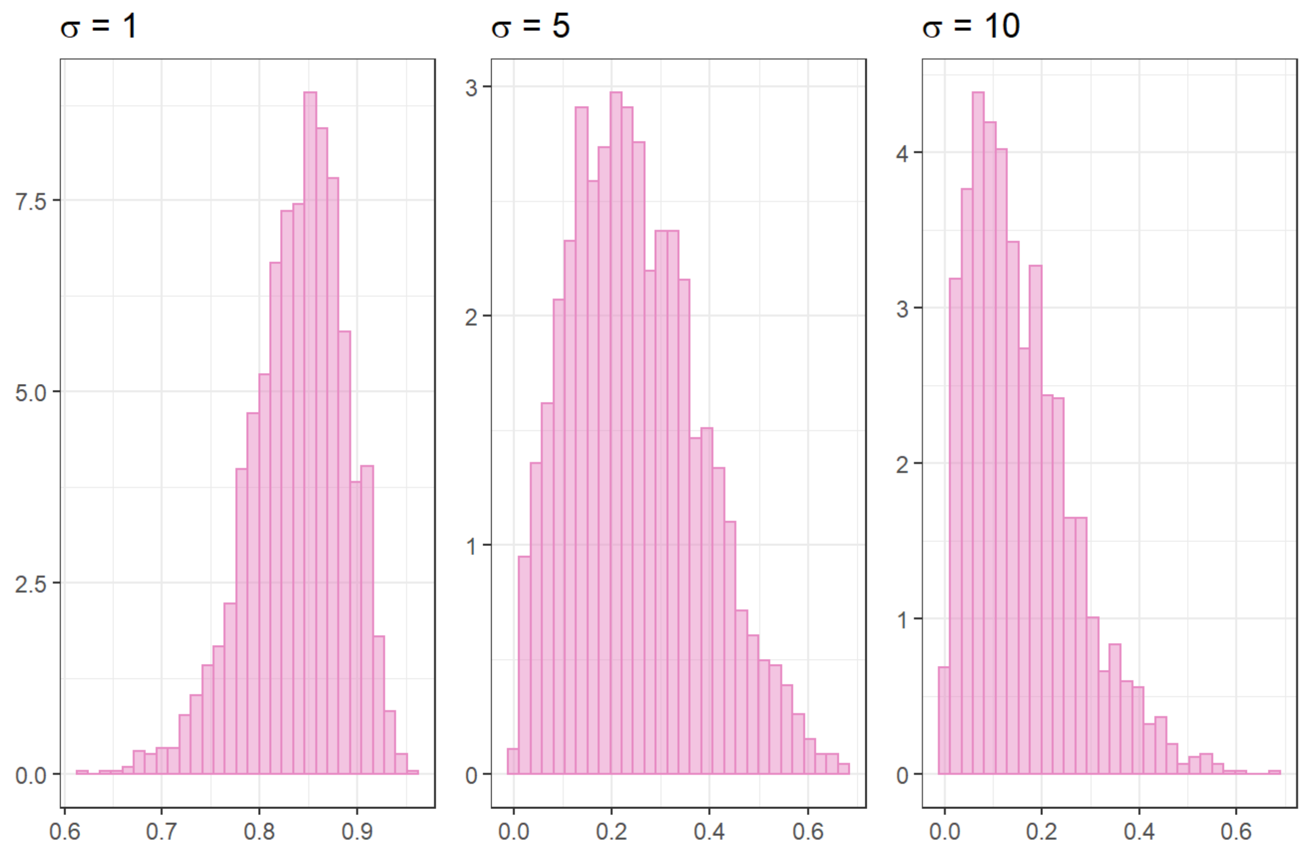
R-squared Distribution

Lastly, visuals of the distribution of R^2 as they relate to changes in σ and changes in the model type.

Distribution of R^2 (Non-Significant Models)



Distribution of R^2 (Significant Models)



Discussion

A theme among the results of this simulation is that as the value of σ becomes lower for the significant models, the performance of the significant models move away from what we would expect under the null model and towards a more significant model.

The non-significant model in this exercise can be thought of as the Null model in a formal hypothesis test since the only non-zero β is the intercept term (value of 3). Essentially the non-significant model is modeling noise that is centered around 3 and has varying values of σ^2. The distributions of the simulations for the F-distribution and the p-value of the F-distribution for the non-significant model very closely approximate the distributions of these statistics under the Null hypothesis. The F-distribution of the simulation (orange) very nearly lines up with the theoretical F-distribution (black line) that we would observe under the null hypothesis. Likewise, the p-value under the Null hypothesis has a uniform distribution and the observed distribution of the p-values for the non-significant models very closely resembles the uniform distribution. The distributions of R^2 for the non-significant models are centered around 0.1 with a right skew. The distribution of the R^2 for the non-significant model is what you would expect for a Null model that has a relatively low ability to explain the variability of the reponse variable.

The results of the significant models show the influence of the changes in σ and model performance. As the value of σ becomes lower, the performance of the model becomes more statistically significant. This visually becomes easy to see for the F-distribution where a significant model with a σ = 10 still somewhat resembles the F-distribution under the Null hypothesis but as σ changes from 10 to 5 and then finally to 1, we can see that the distribution of the F-statistic for the significant model with a σ = 1 becomes visually different than the theoretical F-distribution. A similar trend is also seen for the distribution of the p-value of the F-statistic. For a σ = 10, the significant model does somewhat resemble a slightly skewed uniform distribution. As σ becomes closer to 1, the distribution of the p-values becomes heavily skewed and the overall distribution approaches 0. Since the F-statistic and p-value distributions imply that the significant model with a σ = 1 is likely a statistically significant model, it is not surprising to see that the distribution of R^2 for this model centers around 0.85 and implies that this model has a lot of ability to account for the variance of the response variable.

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` (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.

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

Repeat this process with 1000 simulations for each of the 3 values of σ. 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.

Done correctly, you will have simulated the y vector $3 \times 1000 = 3000$ times. You will have fit $9 \times 3 \times 1000 = 27000$ models. A minimal result would use 3 plots. Additional plots may also be useful.

Potential discussions:

- Does the method **always** select the correct model? On average, does is select the correct model?
- How does the level of noise affect the results?

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.

Methods

We begin by setting a seed and reading in data which will be used for this simulation study.

```r
library(tidyverse)

# Set seed
birthday = 19790221
set.seed(birthday)

# Read in data for predictors
sim2_data = read_csv('study_2.csv')
```

Helper Functions

Next, helper functions are created to streamline repetitive steps in this simulation. Specifically, functions to calculate RMSE, split data in training and testing data sets, fit data to a linear model and finally a function to select the best performing model based on the RMSE performance.

```
# Function to calculate RMSE
rmse = function(n, y, y_hat){
  # n = number of observations
  # y = observed y-value
  # y_hat = estimate of y_hat

  output = sqrt(sum((y - y_hat)^2) / n)

  return(output)
}



# Function to perform splitting of data set in testing and training
train_test_split = function(data, split_prop = 0.50){
  # Add a row number to input data frame to use
  df = data %>% mutate(row_num = row_number())

  # Perform split into test and train
  train = slice_sample(df, prop =  split_prop)
  test = df %>% anti_join(train, by = 'row_num')

  # Remove row number
  train = train %>% select(-row_num)
  test = test %>% select(-row_num)

  # Return split data
  output = list('train' = train, 'test' = test)
  return(output)
}

# Function for fitting a model
fit_model = function(fit_data, formula, n_train, n_test, results, i){

  #Split data
  data_split = train_test_split(fit_data)

  data_train = data_split$train
  data_test = data_split$test

  # y-values for training and testing data sets, used for RMSE
  y_train = data_train$y
  y_test  = data_test$y
  #print(length(y_train))

  # Train and test model
  train_model = lm(formula, data = data_train)
  predict_y = predict(train_model, newdata = data_test)
  #print(formula)

  # Calculate train and test RMSE
  train_rmse = rmse(n_train, y_train, train_model$fitted.values)
  test_rmse = rmse(n_test, y_test, predict_y)
  #print(sqrt(mean(train_model$residuals^2)))

  results[i, 3] = train_rmse
  results[i, 4] = test_rmse
  results[i, 5] = i

  return(results)
}

# Function to indicate lowest RMSE for testing data
select_best_model = function(df){
  df_out = df %>%
  group_by(Iteration) %>%
  arrange(Test_RMSE) %>%
  mutate(Selected_Model = if_else(row_number() == 1, 1, 0)) %>%
  ungroup()

  return(df_out)
}
```

Simulation function

This is the main simulation function that utilizes some of the helper functions to perform the simulation.

```
simulation_2 = function(data, sigma = 1, split_prop = 0.5, iterations = 10){

  # Sample size of data
  sample_size = nrow(data)

  # Number of train and test observations - used for simulation
  n_train = nrow(data) * split_prop
  n_test = nrow(data) - n_train

  # Create formulas for models
  formula_1 = y ~ x1
  formula_2 = y ~ x1 + x2
  formula_3 = y ~ x1 + x2 + x3
  formula_4 = y ~ x1 + x2 + x3 + x4
  formula_5 = y ~ x1 + x2 + x3 + x4 + x5
  formula_6 = y ~ x1 + x2 + x3 + x4 + x5 + x6
  formula_7 = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7
  formula_8 = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8
  formula_9 = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9

  # Tables to hold results for each model - will be combined as the final outp
ut of the function
  results_1 = tibble(Model = 'Model 1', Sigma = sigma, Train_RMSE = rep(0, ite
rations), Test_RMSE = rep(0, iterations), Iteration = rep(0, iterations))
  results_2 = tibble(Model = 'Model 2', Sigma = sigma, Train_RMSE = rep(0, ite
rations), Test_RMSE = rep(0, iterations), Iteration = rep(0, iterations))
  results_3 = tibble(Model = 'Model 3', Sigma = sigma, Train_RMSE = rep(0, ite
rations), Test_RMSE = rep(0, iterations), Iteration = rep(0, iterations))
  results_4 = tibble(Model = 'Model 4', Sigma = sigma, Train_RMSE = rep(0, ite
rations), Test_RMSE = rep(0, iterations), Iteration = rep(0, iterations))
  results_5 = tibble(Model = 'Model 5', Sigma = sigma, Train_RMSE = rep(0, ite
rations), Test_RMSE = rep(0, iterations), Iteration = rep(0, iterations))
  results_6 = tibble(Model = 'Model 6', Sigma = sigma, Train_RMSE = rep(0, ite
rations), Test_RMSE = rep(0, iterations), Iteration = rep(0, iterations))
  results_7 = tibble(Model = 'Model 7', Sigma = sigma, Train_RMSE = rep(0, ite
rations), Test_RMSE = rep(0, iterations), Iteration = rep(0, iterations))
  results_8 = tibble(Model = 'Model 8', Sigma = sigma, Train_RMSE = rep(0, ite
rations), Test_RMSE = rep(0, iterations), Iteration = rep(0, iterations))
  results_9 = tibble(Model = 'Model 9', Sigma = sigma, Train_RMSE = rep(0, ite
rations), Test_RMSE = rep(0, iterations), Iteration = rep(0, iterations))

  # Create betas for simulation
  b0 = 0
  b1 = 3
  b2 = -4
  b3 = 1.6
  b4 = -1.1
  b5 = 0.7
  b6 = 0.5



  for(i in 1:iterations){

    # Error for model
    eps = rnorm(sample_size, mean = 0, sd = sigma)

    # Simulate Data for true model
    sim_data = data %>%
      mutate(y = b0 + b1*x1 + b2*x2 + b3*x3 + b4*x4 + b5*x5 + b6*x6 + eps)

    # Results for each of the nine models
    results_1 = fit_model(sim_data, formula_1,  n_train, n_test, results_1, i)
    results_2 = fit_model(sim_data, formula_2,  n_train, n_test, results_2, i)
    results_3 = fit_model(sim_data, formula_3,  n_train, n_test, results_3, i)
    results_4 = fit_model(sim_data, formula_4,  n_train, n_test, results_4, i)
    results_5 = fit_model(sim_data, formula_5,  n_train, n_test, results_5, i)
    results_6 = fit_model(sim_data, formula_6,  n_train, n_test, results_6, i)
    results_7 = fit_model(sim_data, formula_7,  n_train, n_test, results_7, i)
    results_8 = fit_model(sim_data, formula_8,  n_train, n_test, results_8, i)
    results_9 = fit_model(sim_data, formula_9,  n_train, n_test, results_9, i)


  }

  # Bind results together for all 9 models into one data frame
  output = results_1 %>%
    bind_rows(results_2,
              results_3,
              results_4,
              results_5,
              results_6,
              results_7,
              results_8,
              results_9)


  return(output)

}
```

Run the simulation

```
# Number of iterations for each simulation
iteration = 1000

# Perform simulation
results_sigma1 = simulation_2(data = sim2_data, sigma = 1, iterations = iterat
ion)
results_sigma2 = simulation_2(data = sim2_data, sigma = 2, iterations = iterat
ion)
results_sigma4 = simulation_2(data = sim2_data, sigma = 4, iterations = iterat
ion)

# Get the selected model for each iteration of the simulation
results_sigma1 = select_best_model(results_sigma1)
results_sigma2 = select_best_model(results_sigma2)
results_sigma4 = select_best_model(results_sigma4)

# Combine simulation results into one dataframe for analysis
results_all = results_sigma1 %>%
  bind_rows(results_sigma2,
            results_sigma4)
```

Visuals Average RMSE

```
# Data for average RMSE visual
df_avg_rmse = results_all %>%
  group_by(Sigma, Model) %>%
  summarise(Train = mean(Train_RMSE),
            Test = mean(Test_RMSE)) %>%
  mutate(sigma_display = str_c('Sigma = ', Sigma)) %>%
  pivot_longer(cols = c(Train, Test), names_to = 'RMSE Type', values_to = 'Avg
_value')

# plot
p_avg_rmse = ggplot(df_avg_rmse, aes(x = Model, y = Avg_value, group = `RMSE T
ype`, color = `RMSE Type`)) + #, color = RMSE_type
  geom_line(aes(linetype = `RMSE Type`)) +
  geom_point() +
  facet_wrap(. ~ sigma_display) +
  scale_color_manual(values = c('#fc8d62', '#8da0cb')) +
  scale_linetype_manual(values = c('dashed', 'solid')) +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 30, hjust = 1, vjust = 1),
        legend.position = "bottom") +
  labs(x = NULL,
       y = 'Average RMSE')
```

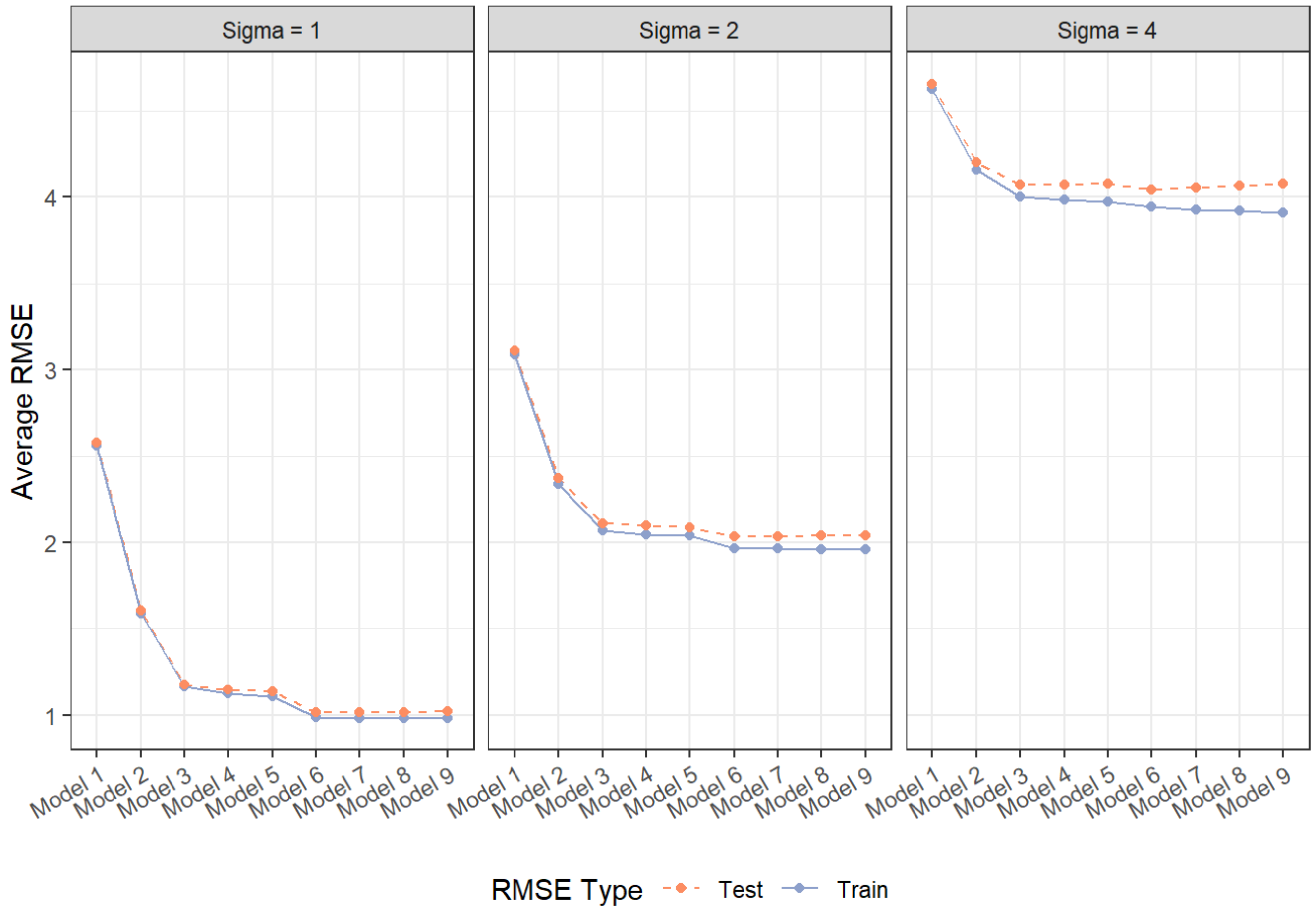
Visuals Model Selection

```
# Data for average RMSE visual
df_selected = results_all %>%
  group_by(Sigma, Model) %>%
  summarise(`Model Selected` = sum(Selected_Model)) %>%
  mutate(sigma_display = str_c('Sigma = ', Sigma)) %>%
  mutate(`True Model` = if_else(Model == 'Model 6', 'Y', 'N'))

p_selected = ggplot(df_selected, aes(x = Model, y = `Model Selected`)) +
  geom_col(aes( fill = `True Model`)) +
  facet_wrap(. ~ sigma_display) +
  scale_fill_manual(values = c('#a6d854', '#e78ac3')) +
  theme_bw() +
    theme(axis.text.x = element_text(angle = 30, hjust = 1, vjust = 1),
        legend.position = "bottom") +
  labs(x = NULL,
       y = 'Number of Times a Model Produced the \nLowest RMSE in an Iteratio
n')
```

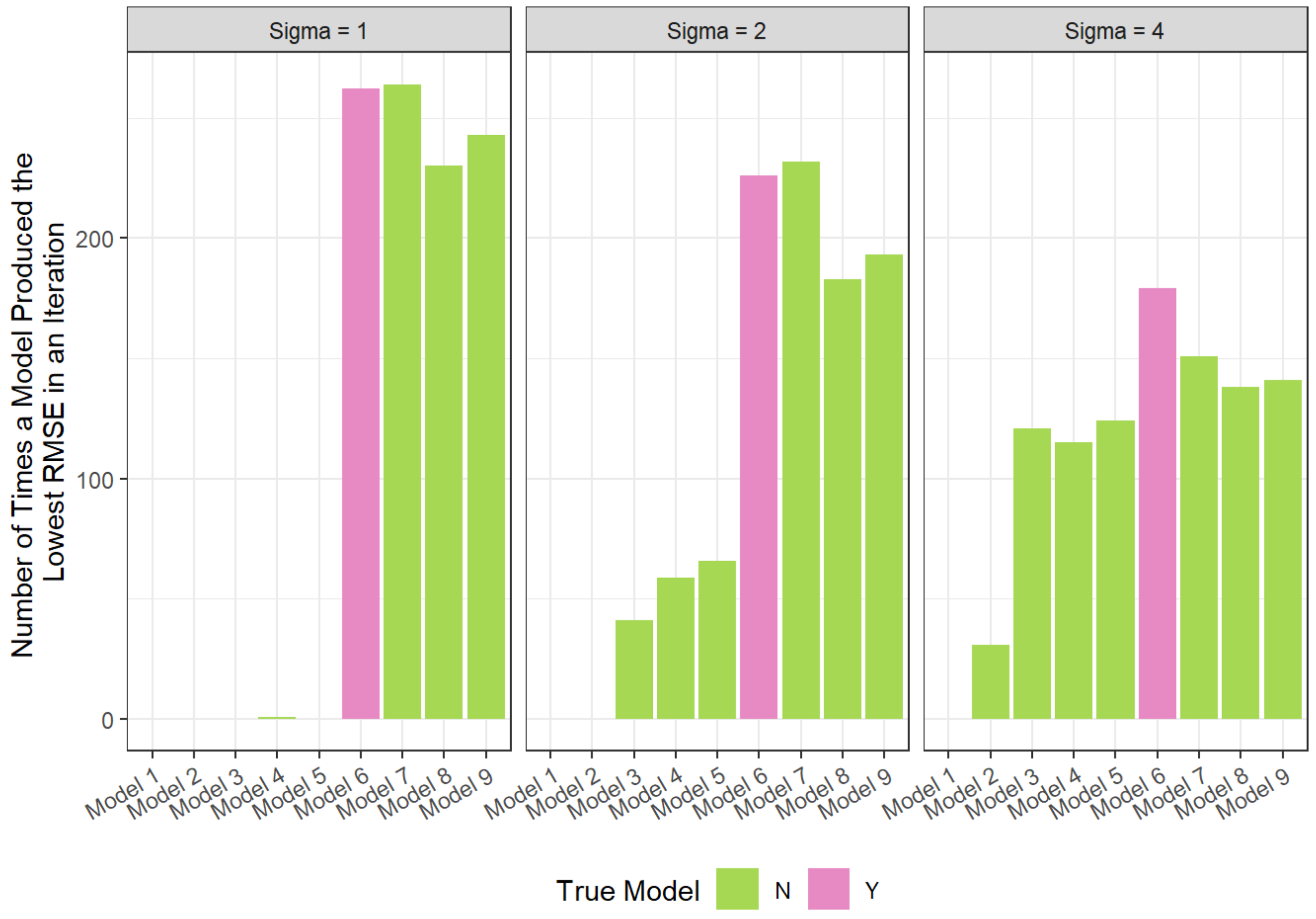
Results

The first visual shows the average RMSE performance for each model as the value of σ changes. The second visual shows the number of times a model was selected based on the lowest RMSE performance for a given iteration in the simulation.

Average RMSE



Model Selection Based on RMSE



Discussion

RMSE does not always select the correct model in this simulation (Model 6). Looking at the first set of visuals in the Results section showing the average RMSE for each model and each value of σ, for all levels of σ model 3 (predictors x1 + x2+ x3) would be the model that represents the best compromise between model performance while also minimizing the number of predictor variables. Interestingly, the value of σ did not affect the selection of the optimal model when evaluating by the average RMSE. The general trend is the same for all 3 values of σ. An increase in σ does increase the absolute amount of error in the model, but the bend in the trend line is consistently at model 3.

The second visual in the results section shows the number of times a model was selected as the best performing model based on the lowest test RMSE. This method of model selection favors models with a larger number of variables. A low level of σ seems to be more discriminating in only selecting the largest models, but as the value of σ increases the chance that a smaller model could be the best performing model also increases.

Generally, the increase in the amount of noise in the model seems to mute any patterns in the models performance. In the first visual of the trend in average RMSE, as the noise increases the decrease in RMSE becomes less substantial. I would assume for a large amount of noise in the data, this trend line would become almost flat and would be difficult to detect a difference in the performance for any of the models. Likewise for the second visual, as the noise increases the bar graph for σ = 4 almost resembles a uniform distribution. It would make sense that as σ increase this visual would become more uniform and any given model would have roughly the same likelihood to be selected as the optimal model. This increase in noise would then basically hide the signal of the model visually.

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 $\alpha = 0.05$.

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, 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

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

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

Create 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.

Potential discussions:

- How do n, β_1, and σ affect power? Consider additional plots to demonstrate these effects.
- Are 1000 simulations sufficient?

An additional tip:

- Search online for examples of power curves to give you inspiration for how you might construct your own plots here. You'll find both two-sided and one-sided power curves. Based on the way you're asked to construct the β_1 vector, you should be able to figure out which type is appropriate here.

Methods

Begin the last simulation by setting a seed

```
library(broom)
library(tidyverse)

# Set seed
birthday = 19790221
set.seed(birthday)
```

Simulation Function

`simulation_3` is the main function to carry out the simulation.

```
simulation_3 = function(sample_size, sigma, iterations){

  # Create vector of betas and store length of beta vector
  beta = seq(-2, 2, 0.1)
  beta_length = length(beta)

  # Create vector of betas with each value in beta vector repeated the length
of the sample size
  betas = rep(beta, each = sample_size)

  # Create x_value vector to hold predictor variables
  x_values = seq(0, 5, length = sample_size)

  # Create output dataframe for returning results from the loop
  output = tibble()

  # begin for loop
  for(i in 1:iterations){
  # Create vector of error term for model
  epsilon = rnorm(sample_size * beta_length, mean = 0, sd = sigma)
  iteration = i

  # Temporary data frame created from vectors above to be used in the simulati
on
  df_tmp = tibble('Betas' = betas, 'x_values' = rep(x_values, beta_length), 'e
psilon' = epsilon, 'sample_size' = sample_size,
            'sigma' = sigma, 'iteration' = iteration)

  # Calculate response variable y
  df_tmp = df_tmp %>%
  mutate(y = betas * x_values + epsilon)

  # output data frame
  df_tmp = df_tmp %>%
  group_by(sample_size, sigma, iteration, Betas) %>%
  group_modify(~ broom::tidy(lm(y ~ x_values, data = .x))) %>%
  filter(term == 'x_values')

  output = output %>%
    bind_rows(df_tmp)
  }
  return(output)
}
```

Simulation

Run the simulation using `simulation_3` from the previous block.

```
iterations = 1000

# Sample size 10, sigmas 1, 2, 4
sim_n10_s1 = simulation_3(sample_size = 10, sigma = 1, iterations = iteration
s)
sim_n10_s2 = simulation_3(sample_size = 10, sigma = 2, iterations = iteration
s)
sim_n10_s4 = simulation_3(sample_size = 10, sigma = 4, iterations = iteration
s)

# Sample size 20, sigmas 1, 2, 4
sim_n20_s1 = simulation_3(sample_size = 20, sigma = 1, iterations = iteration
s)
sim_n20_s2 = simulation_3(sample_size = 20, sigma = 2, iterations = iteration
s)
sim_n20_s4 = simulation_3(sample_size = 20, sigma = 4, iterations = iteration
s)

# Sample size 30, sigmas 1, 2, 4
sim_n30_s1 = simulation_3(sample_size = 30, sigma = 1, iterations = iteration
s)
sim_n30_s2 = simulation_3(sample_size = 30, sigma = 2, iterations = iteration
s)
sim_n30_s4 = simulation_3(sample_size = 30, sigma = 4, iterations = iteration
s)

# Create one data set for analysis
df_sim3 = sim_n10_s1 %>%
  bind_rows(sim_n10_s2,
            sim_n10_s4,
            sim_n20_s1,
            sim_n20_s2,
            sim_n20_s4,
            sim_n30_s1,
            sim_n30_s2,
            sim_n30_s4)
```

Add column indicating if the significance test was rejected for a p-value less than $\alpha = 0.05$

```
df_sim3 = df_sim3 %>%
  mutate(less_than_alpha = if_else(p.value < 0.05, 1, 0))
```

Create aggregated data frame for plotting power curve

```
df_curve = df_sim3 %>%
  group_by(sample_size, sigma, Betas) %>%
  summarise(power = mean(less_than_alpha))
```

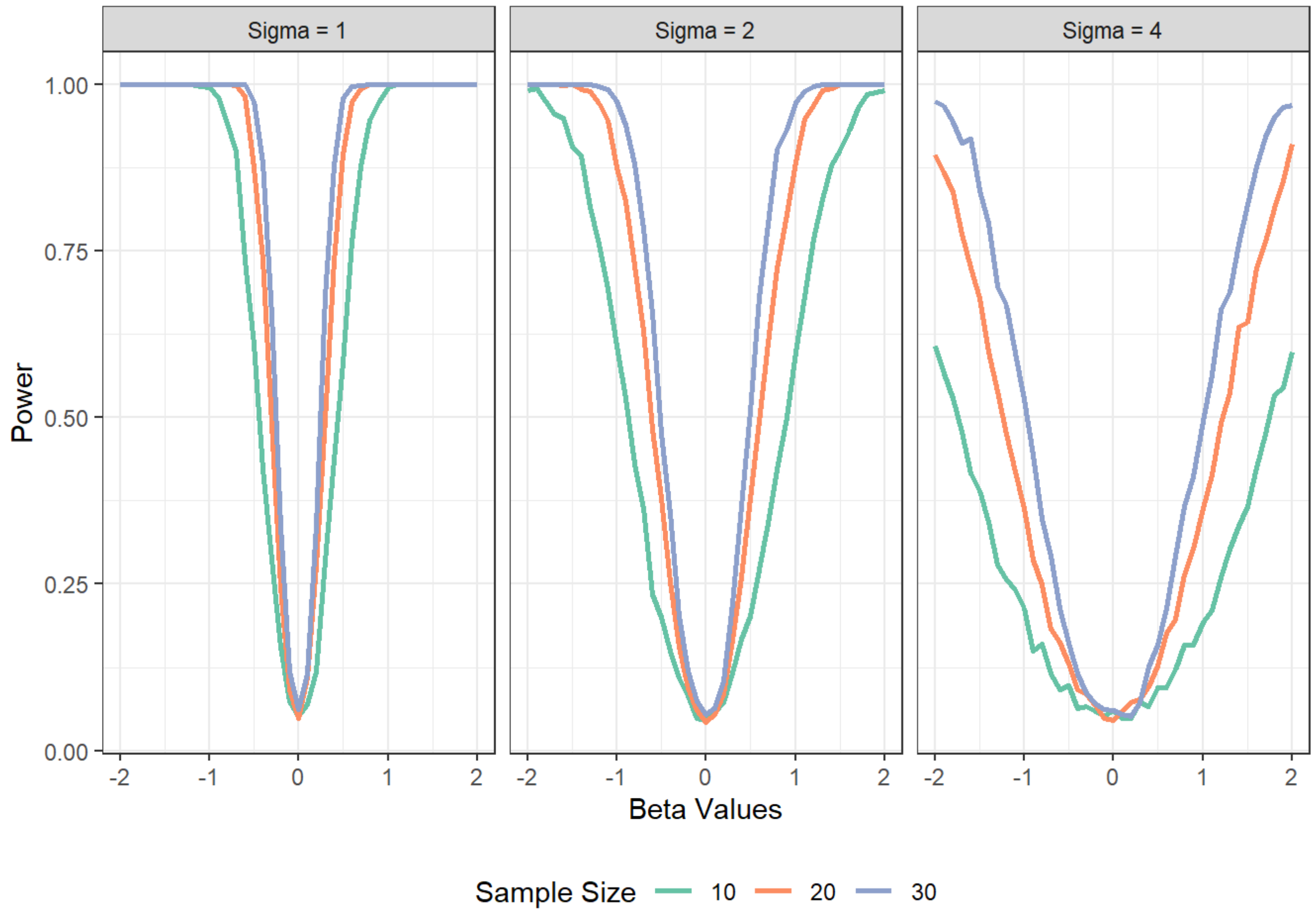
Two-sided power curve for each level of σ.

```
df_curve = df_curve %>%
  mutate(`Sample Size` = factor(sample_size),
         sigma_display = str_c('Sigma = ', sigma))



p_pcurve = ggplot(df_curve, aes(x = Betas, y = power, color = `Sample Size`, g
roup = `Sample Size`)) +
  geom_line(size = 1) +
  scale_color_manual(values = c('#66c2a5', '#fc8d62', '#8da0cb')) +
  facet_wrap(. ~ sigma_display) +
  theme_bw() +
  theme(legend.position = 'bottom') +
  labs(y = 'Power',
       x = 'Beta Values')
```

Results

Three power curves, one for each value of σ.

```
p_pcurve
```



Discussion

The factor that most affected power in this simulation was the value of σ. The increase in noise decreases the likelihood that the Null hypothesis is rejected and leads to a less powerful test. The increase in noise also makes it more difficult to detect less powerful signals. The curves for $\sigma \in (1, 2)$ are relatively steep with signals close to 0 generating larger statistical power when compared to the models where $\sigma = 4$ which is a less steep curve with a more rounded bottom. For the models where $\sigma = 4$, signals with a small sample size generate relatively less power. Larger sample sizes do produce more powerful models for all values of σ, but this is most pronounced in the noisiest model. Sample sizes of 20 and 30 generate power more quickly versus models with a sample size of 10. This could be a point to consider if one is concerned about the amount of noise in their data that if possible, increasing the sample size could be a strategy to help mitigate the impact of noise.

I do think that 1000 simulations is sufficient to calculate power to 3 decimal places. If a more precise estimate is needed to additional decimal places, then increasing the number of iterations in a simulation would be useful. I do not think that increasing the number of iterations would change the shape of the power curve for any of the values for σ. For the noisiest model with $\sigma = 4$, increasing the sample size would be a more effective strategy to increase the power of a model as opposed to increasing the iterations in the simulation. Currently, the curve in $\sigma = 4$ becomes steeper with every increase in the sample size and with a large enough sample size the power for the noisiest model would eventually approach 1.