Week 6 - Simulation Project

STAT 420, Summer 2019, Connor Segneri - segneri3

Contents

Simulation Study 1: Significance of Regression

```
birthday = 19951015
set.seed(birthday)
```

Introduction

This first simulation will be about the significance of regression. I am going to simulate two different models, one significant, and one non-significant. For both models, I'll consider 3 possible levels of noise, or 3 different standard deviations. For each model and each standard deviation, I'll run 2500 simulations. For each simulation, I'll fit a linear regression model of the same form, extracting the f-statistic, p-value, and r-squared in order to get and empirical distribution of all three.

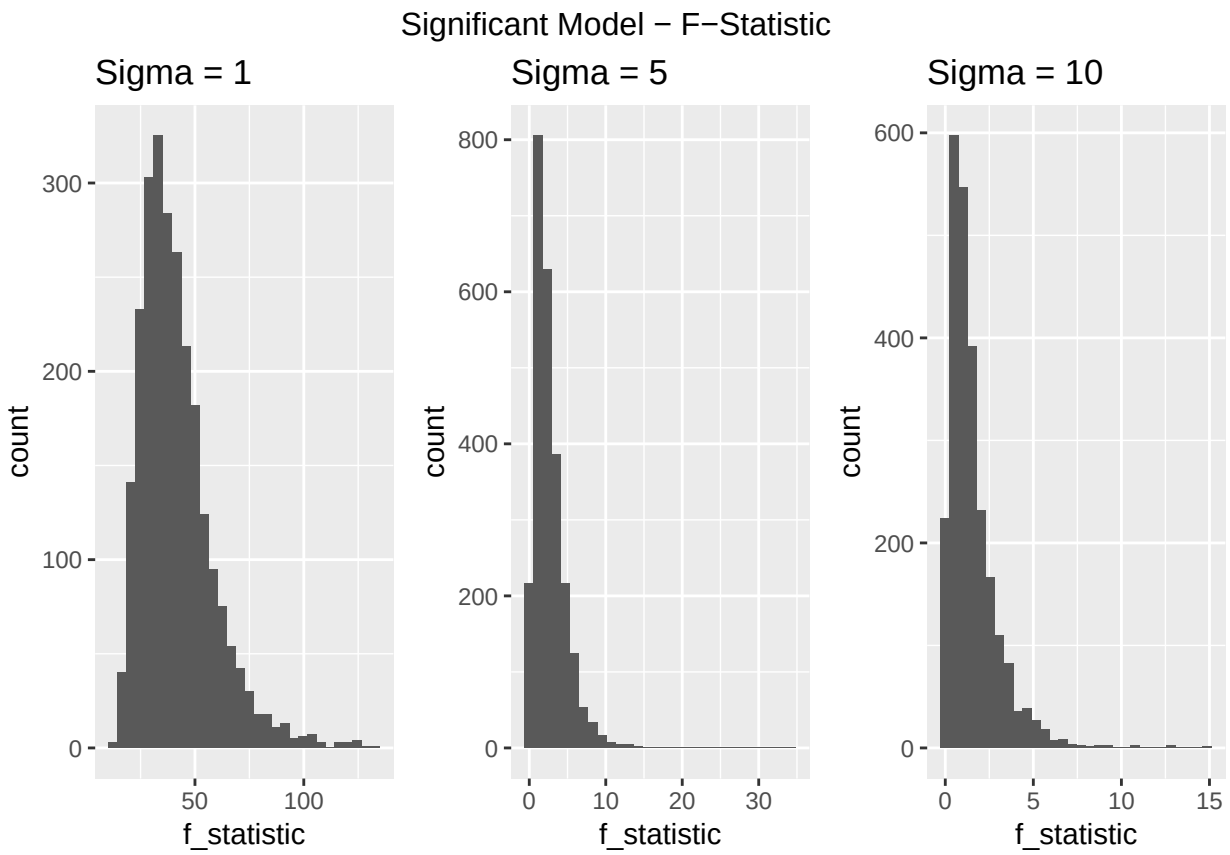
Methods

```
study1 = read.csv("study_1.csv")

MLR_simulate = function(num_sims=2500, n=25, sigma, betas, df) {
  f_statistic = rep(0, num_sims)
  p_value = rep(0, num_sims)
  r_squared = rep(0, num_sims)

  for(i in 1:num_sims) {
    eps = rnorm(n, mean=0, sd=sigma)
    df$y = betas[1] + betas[2]*df$x1 + betas[3]*df$x2 + betas[4]*df$x3 + eps
    fit = lm(y ~ x1 + x2 + x3, data = df)

    fstat = summary(fit)$fstatistic
    f_statistic[i] = fstat[1]
    p_value[i] = pf(fstat[1], fstat[2], fstat[3], lower.tail = FALSE)
    r_squared[i] = summary(fit)$r.squared
```

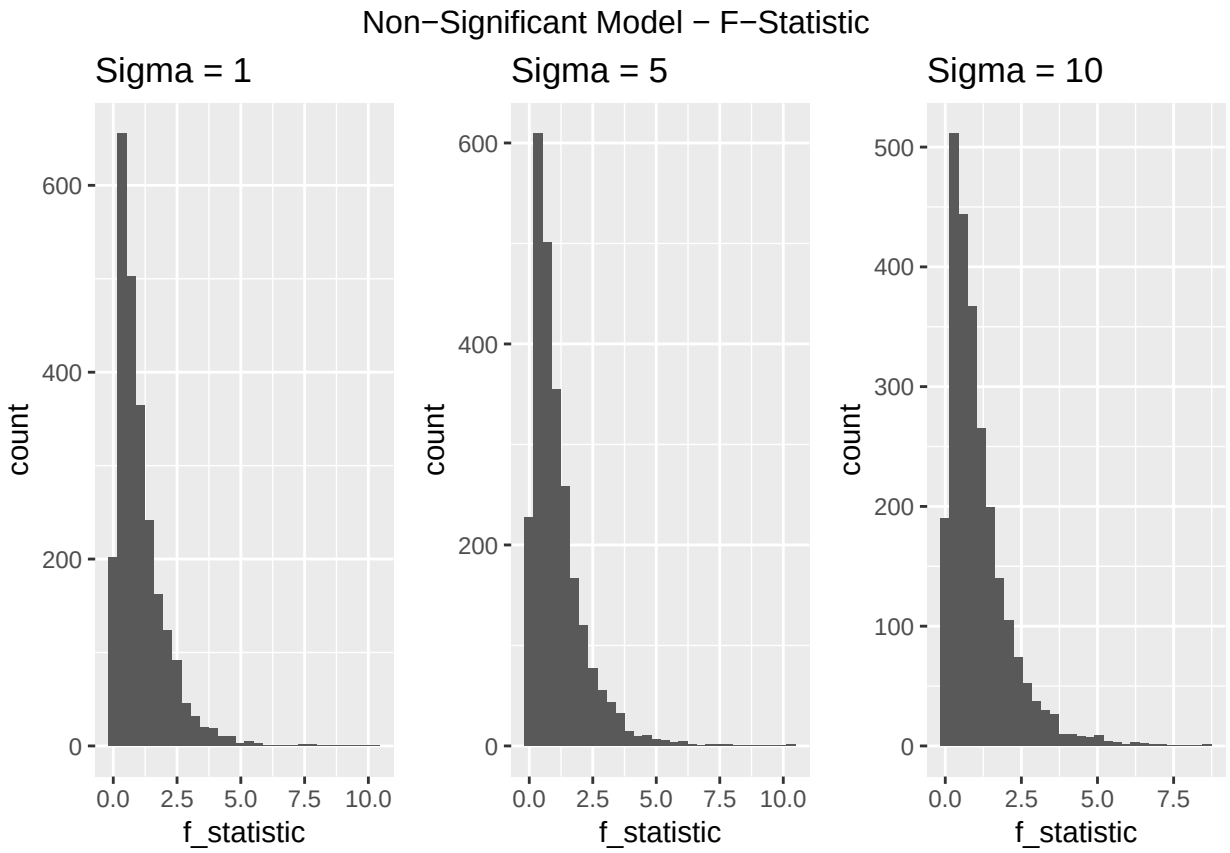
```
  }

  data.frame(
    f_statistic = f_statistic,
    p_value = p_value,
    r_squared = r_squared
  )
}

sig_betas = c(3,1,1,1)
sig_sd1 = MLR_simulate(sigma=1, betas=sig_betas, df=study1)
sig_sd5 = MLR_simulate(sigma=5, betas=sig_betas, df=study1)
sig_sd10 = MLR_simulate(sigma=10, betas=sig_betas, df=study1)

nonsig_betas = c(3,0,0,0)
nonsig_sd1 = MLR_simulate(sigma=1, betas=nonsig_betas, df=study1)
nonsig_sd5 = MLR_simulate(sigma=5, betas=nonsig_betas, df=study1)
nonsig_sd10 = MLR_simulate(sigma=10, betas=nonsig_betas, df=study1)
```

Results

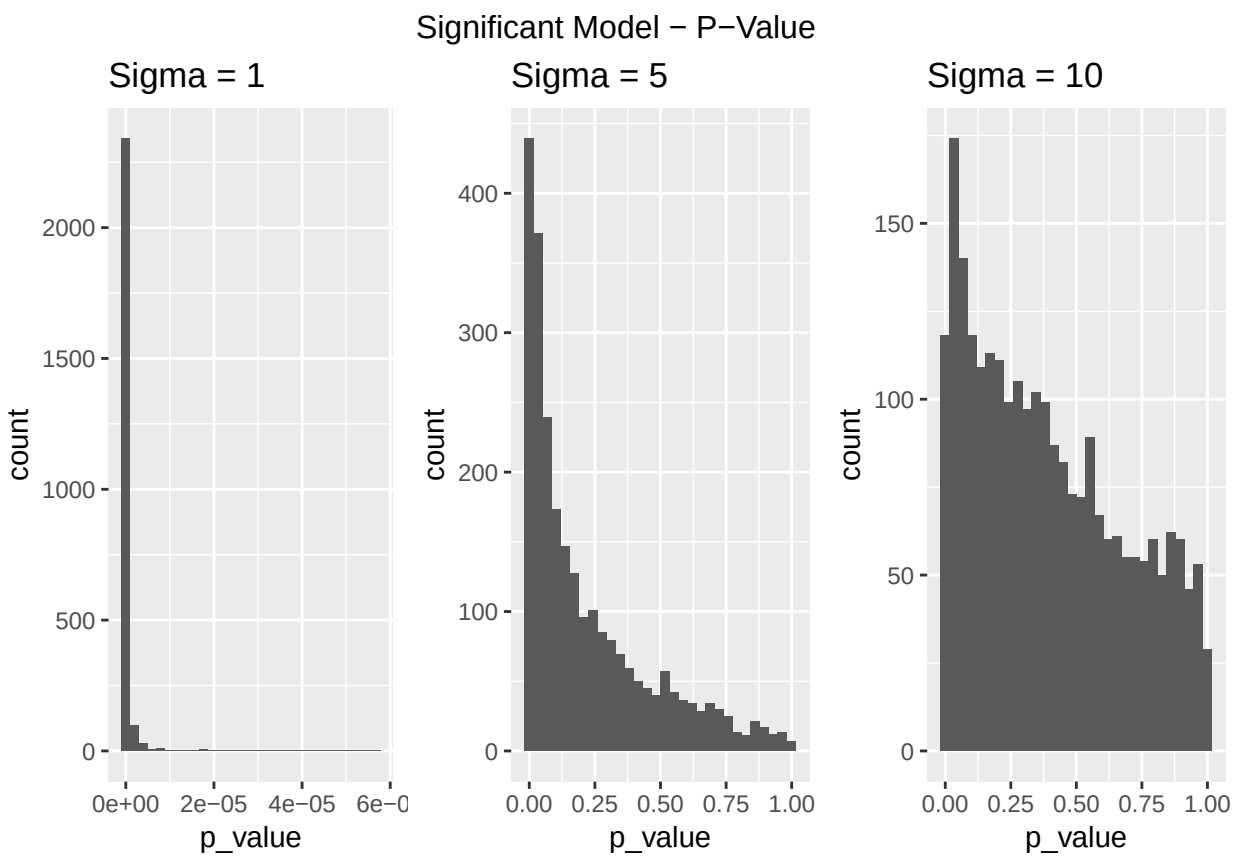
```
library(ggplot2)
library(gridExtra)

p1 = ggplot(sig_sd1, aes(x=f_statistic)) + geom_histogram() + ggtitle("Sigma = 1")
p2 = ggplot(sig_sd5, aes(x=f_statistic)) + geom_histogram() + ggtitle("Sigma = 5")
p3 = ggplot(sig_sd10, aes(x=f_statistic)) + geom_histogram() + ggtitle("Sigma = 10")
grid.arrange(p1,p2,p3, ncol = 3, nrow = 1, top = "Significant Model - F-Statistic")
```



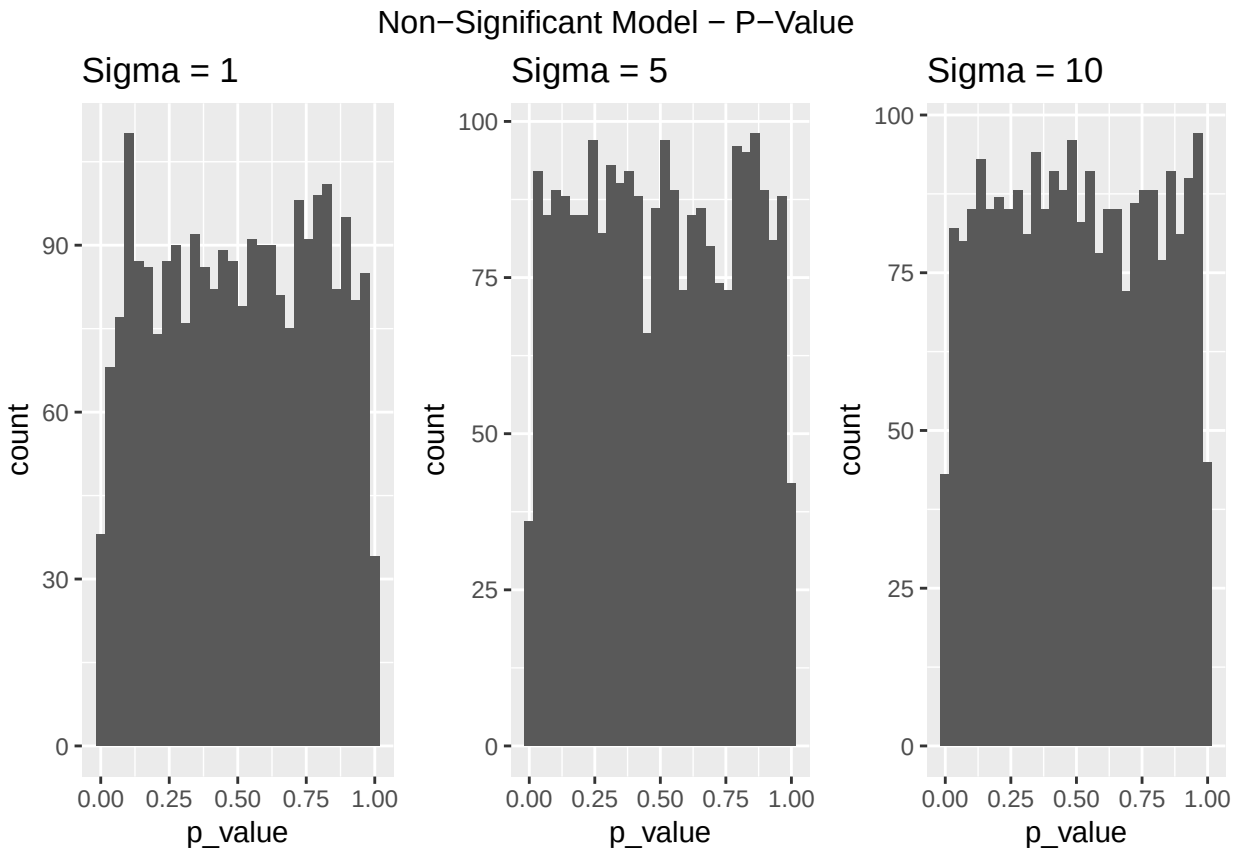

```
p1 = ggplot(nonsig_sd1, aes(x=f_statistic)) + geom_histogram() + ggtitle("Sigma = 1")
p2 = ggplot(nonsig_sd5, aes(x=f_statistic)) + geom_histogram() + ggtitle("Sigma = 5")
p3 = ggplot(nonsig_sd10, aes(x=f_statistic)) + geom_histogram() + ggtitle("Sigma = 10")
grid.arrange(p1,p2,p3, ncol = 3, nrow = 1, top = "Non-Significant Model - F-Statistic")
```



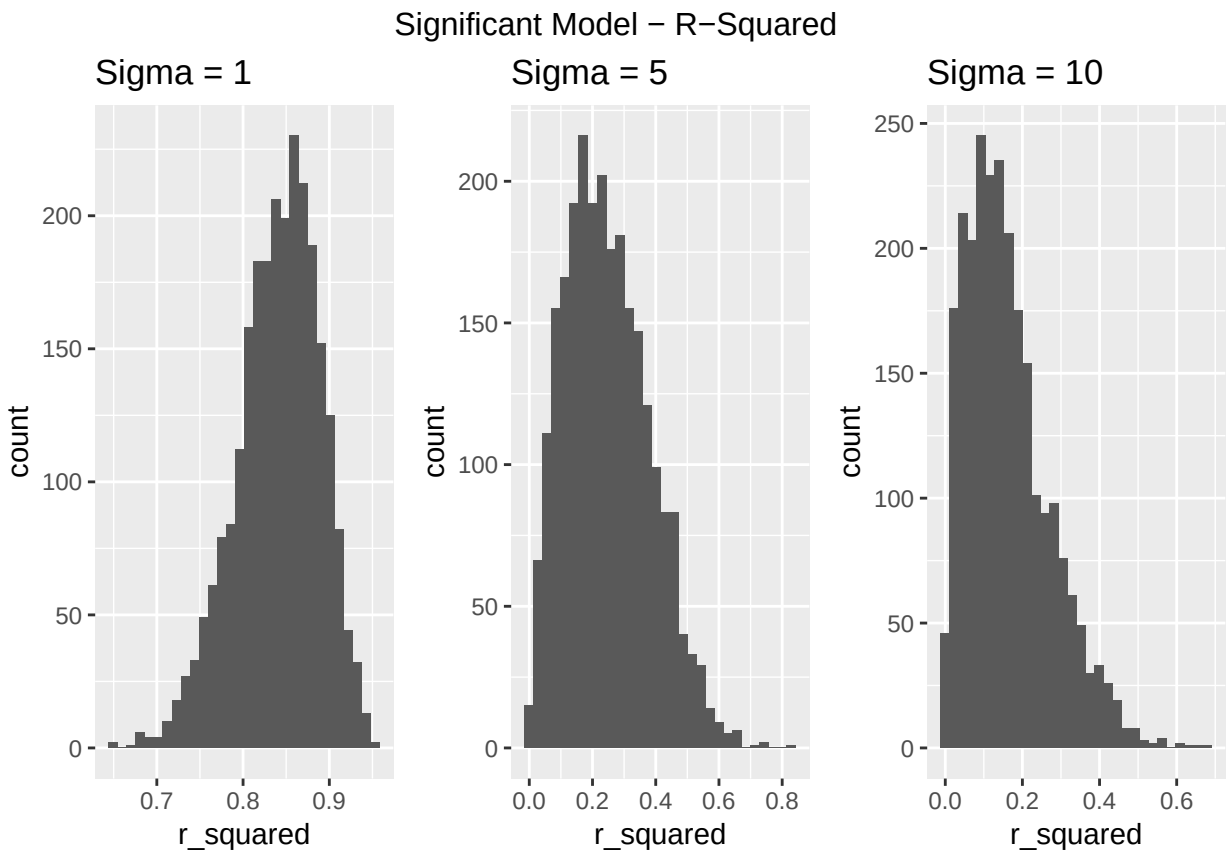

```
p1 = ggplot(sig_sd1, aes(x=p_value)) + geom_histogram() + ggtitle("Sigma = 1")
p2 = ggplot(sig_sd5, aes(x=p_value)) + geom_histogram() + ggtitle("Sigma = 5")
p3 = ggplot(sig_sd10, aes(x=p_value)) + geom_histogram() + ggtitle("Sigma = 10")
grid.arrange(p1,p2,p3, ncol = 3, nrow = 1, top = "Significant Model - P-Value")
```



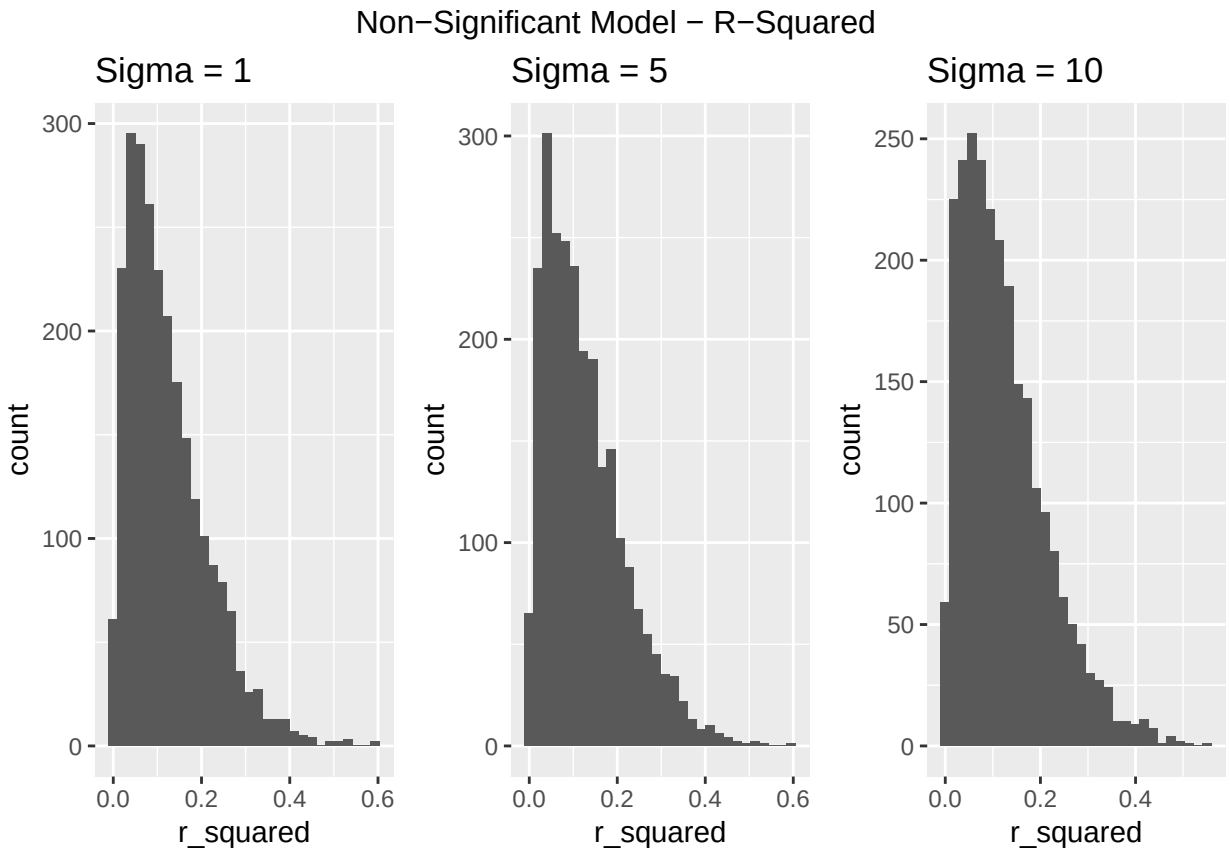

```
p1 = ggplot(nonsig_sd1, aes(x=p_value)) + geom_histogram() + ggtitle("Sigma = 1")
p2 = ggplot(nonsig_sd5, aes(x=p_value)) + geom_histogram() + ggtitle("Sigma = 5")
p3 = ggplot(nonsig_sd10, aes(x=p_value)) + geom_histogram() + ggtitle("Sigma = 10")
grid.arrange(p1,p2,p3, ncol = 3, nrow = 1, top = "Non-Significant Model - P-Value")
```




```
p1 = ggplot(sig_sd1, aes(x=r_squared)) + geom_histogram() + ggtitle("Sigma = 1")
p2 = ggplot(sig_sd5, aes(x=r_squared)) + geom_histogram() + ggtitle("Sigma = 5")
p3 = ggplot(sig_sd10, aes(x=r_squared)) + geom_histogram() + ggtitle("Sigma = 10")
grid.arrange(p1,p2,p3, ncol = 3, nrow = 1, top = "Significant Model - R-Squared")
```




```
p1 = ggplot(nonsig_sd1, aes(x=r_squared)) + geom_histogram() + ggtitle("Sigma = 1")
p2 = ggplot(nonsig_sd5, aes(x=r_squared)) + geom_histogram() + ggtitle("Sigma = 5")
p3 = ggplot(nonsig_sd10, aes(x=r_squared)) + geom_histogram() + ggtitle("Sigma = 10")
grid.arrange(p1,p2,p3, ncol = 3, nrow = 1, top = "Non-Significant Model - R-Squared")
```



Discussion

P-Value Distribution

The true distribution of p-values vary depending on which hypothesis is true in the hypothesis test for the linear regression model. If the null hypothesis is true, then all p-values between 0 and 1 are equally likely, so the distribution is rectangular. If the alternative hypothesis is true, than p-values have a distribution in which p-values near zero are more likely than p-values near 1.

We can see that this is the case in the plots above. In the significant model, when the null hypothesis is true, the p-values follow the distribution where the bulk of the values are collected around 0. In the non-significant model, when the alternative hypothesis is true, the p-values follow a rectangular distribution.

R-Squared and Sigma

It seems as though when standard deviation (sigma) increases, r-squared decreases. The mean R-Squared for the significant model as the standard deviation increases `1`, `5`, and `10`, are 0.8406959, 0.2479273, and 0.1622324. This doesn't seem to affect the non-significant model though, as the mean R-squared for the non-significant model as the standard deviation increases are 0.1227553, 0.1250004, and 0.124277. Although it would make sense for the r-squared to reflect that there is no significant relationship between the predicted variable and the predictor variables in the non-significant model no matter what the standard deviation is set to.

We can also see in the significant model plots above that as standard deviation increases, the r-squared distributions shift left, towards zero.

Simulation Study 2: Using RMSE for Selection

```
birthday = 19951015
set.seed(birthday)
```

Introduction

In this simulation, I'll be using rmse for model selection. I'll be investigating how well this procedure works. Averaged over many model fit attempts, I expect that we can use rmse to select the appropriate model. I'm going to simulate a multiple linear regression model that has 6 predictors, `x1` through `x6`. There are also true values for all of the betas. I'll consider a sample size of 500, and three possible levels of noise, where standard deviation is `1`, `2`, and `4`. Each time I simulate the data, I'll randomly split the data into train and test sets of equal sizes. For each simulation, I'll fit nine models for each both train and test data. The first model using the first predictor, the second model using the first and the second predictor, and so on.

Methods

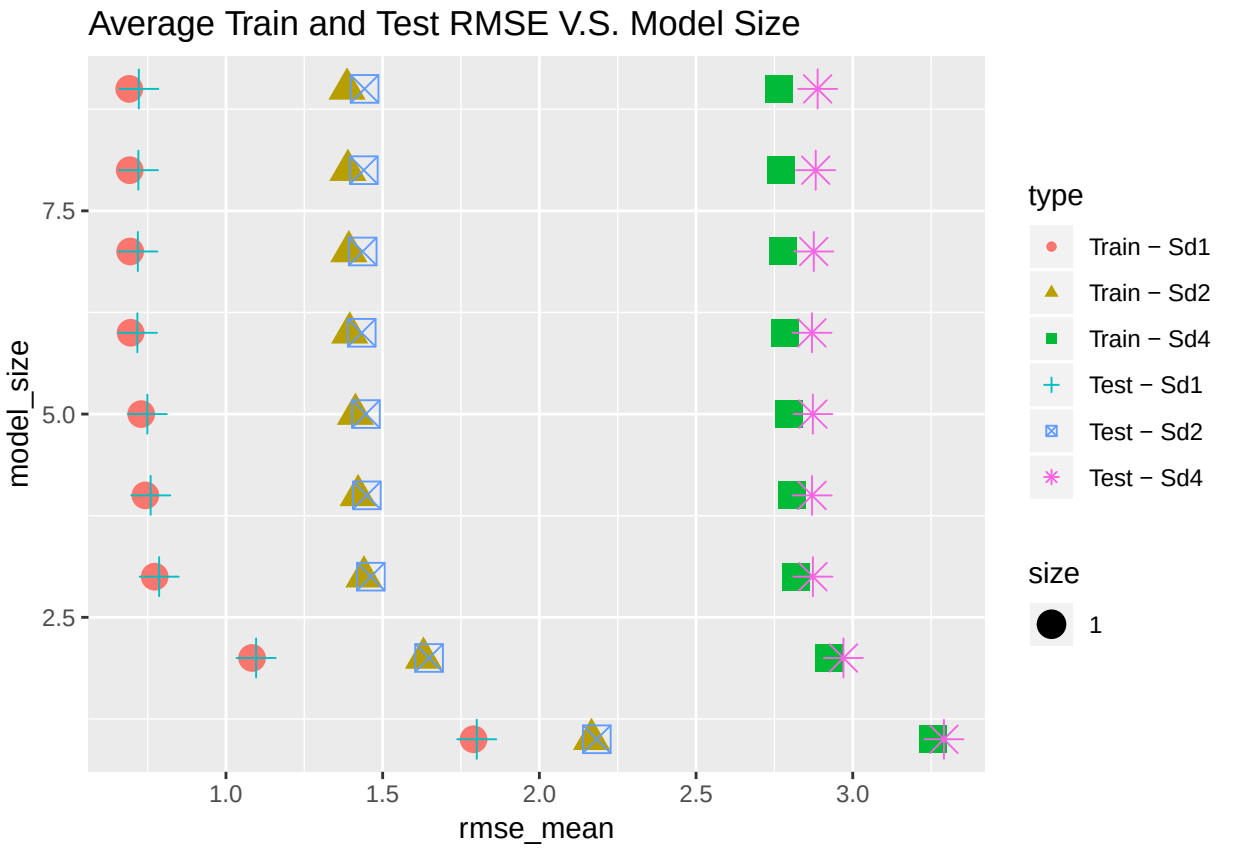
```
study2 = read.csv("study_2.csv")

MLR_simulate = function(num_sims=1000, n=500, sigma, betas, df) {
  rmse_df = data.frame(
    train_fit1 = rep(0,num_sims),
    train_fit2 = rep(0,num_sims),
    train_fit3 = rep(0,num_sims),
    train_fit4 = rep(0,num_sims),
    train_fit5 = rep(0,num_sims),
    train_fit6 = rep(0,num_sims),
    train_fit7 = rep(0,num_sims),
    train_fit8 = rep(0,num_sims),
    train_fit9 = rep(0,num_sims),
    test_fit1 = rep(0,num_sims),
    test_fit2 = rep(0,num_sims),
    test_fit3 = rep(0,num_sims),
    test_fit4 = rep(0,num_sims),
    test_fit5 = rep(0,num_sims),
    test_fit6 = rep(0,num_sims),
    test_fit7 = rep(0,num_sims),
    test_fit8 = rep(0,num_sims),
    test_fit9 = rep(0,num_sims)
  )

  for(i in 1:num_sims) {
    eps = rnorm(n, mean=0, sd=sigma)
    df$y = betas[1] + betas[2]*df$x1 + betas[3]*df$x2 + betas[4]*df$x3 +
      betas[5]*df$x4 + betas[6]*df$x5 + betas[7]*df$x6 + eps

    train = sample(seq_len(n), size=n/2)
    df_train = df[train,]
    df_test = df[-train,]
```

```
    fit1 = lm(y ~ x1, data = df_train)
    fit2 = lm(y ~ x1+x2, data = df_train)
    fit3 = lm(y ~ x1+x2+x3, data = df_train)
    fit4 = lm(y ~ x1+x2+x3+x4, data = df_train)
    fit5 = lm(y ~ x1+x2+x3+x4+x5, data = df_train)
    fit6 = lm(y ~ x1+x2+x3+x4+x5+x6, data = df_train)
    fit7 = lm(y ~ x1+x2+x3+x4+x5+x6+x7, data = df_train)
    fit8 = lm(y ~ x1+x2+x3+x4+x5+x6+x7+x8, data = df_train)
    fit9 = lm(y ~ x1+x2+x3+x4+x5+x6+x7+x8+x9, data = df_train)

    rmse_df$train_fit1[i] = sqrt((1/n) * sum((df_train$y - predict(fit1, newdata = df_train))^2))
    rmse_df$train_fit2[i] = sqrt((1/n) * sum((df_train$y - predict(fit2, newdata = df_train))^2))
    rmse_df$train_fit3[i] = sqrt((1/n) * sum((df_train$y - predict(fit3, newdata = df_train))^2))
    rmse_df$train_fit4[i] = sqrt((1/n) * sum((df_train$y - predict(fit4, newdata = df_train))^2))
    rmse_df$train_fit5[i] = sqrt((1/n) * sum((df_train$y - predict(fit5, newdata = df_train))^2))
    rmse_df$train_fit6[i] = sqrt((1/n) * sum((df_train$y - predict(fit6, newdata = df_train))^2))
    rmse_df$train_fit7[i] = sqrt((1/n) * sum((df_train$y - predict(fit7, newdata = df_train))^2))
    rmse_df$train_fit8[i] = sqrt((1/n) * sum((df_train$y - predict(fit8, newdata = df_train))^2))
    rmse_df$train_fit9[i] = sqrt((1/n) * sum((df_train$y - predict(fit9, newdata = df_train))^2))

    rmse_df$test_fit1[i] = sqrt((1/n) * sum((df_test$y - predict(fit1, newdata = df_test))^2))
    rmse_df$test_fit2[i] = sqrt((1/n) * sum((df_test$y - predict(fit2, newdata = df_test))^2))
    rmse_df$test_fit3[i] = sqrt((1/n) * sum((df_test$y - predict(fit3, newdata = df_test))^2))
    rmse_df$test_fit4[i] = sqrt((1/n) * sum((df_test$y - predict(fit4, newdata = df_test))^2))
    rmse_df$test_fit5[i] = sqrt((1/n) * sum((df_test$y - predict(fit5, newdata = df_test))^2))
    rmse_df$test_fit6[i] = sqrt((1/n) * sum((df_test$y - predict(fit6, newdata = df_test))^2))
    rmse_df$test_fit7[i] = sqrt((1/n) * sum((df_test$y - predict(fit7, newdata = df_test))^2))
    rmse_df$test_fit8[i] = sqrt((1/n) * sum((df_test$y - predict(fit8, newdata = df_test))^2))
    rmse_df$test_fit9[i] = sqrt((1/n) * sum((df_test$y - predict(fit9, newdata = df_test))^2))
  }

  rmse_df
}

betas = c(0, 5, -4, 1.6, -1.1, 0.7, 0.3)
rmse_df_1 = MLR_simulate(sigma=1, betas=betas, df=study2)
rmse_df_2 = MLR_simulate(sigma=2, betas=betas, df=study2)
rmse_df_4 = MLR_simulate(sigma=4, betas=betas, df=study2)
```

Results

```
rmse_df_1_means = as.vector(colMeans(rmse_df_1))
rmse_df_2_means = as.vector(colMeans(rmse_df_2))
rmse_df_4_means = as.vector(colMeans(rmse_df_4))

rmse_per_model_size = rbind(
  data.frame(rmse_mean = rmse_df_1_means[1:9], type = "Train - Sd1"),
  data.frame(rmse_mean = rmse_df_2_means[1:9], type = "Train - Sd2"),
  data.frame(rmse_mean = rmse_df_4_means[1:9], type = "Train - Sd4"),
  data.frame(rmse_mean = rmse_df_1_means[10:18], type = "Test - Sd1"),
  data.frame(rmse_mean = rmse_df_2_means[10:18], type = "Test - Sd2"),
```

```
  data.frame(rmse_mean = rmse_df_4_means[10:18], type = "Test - Sd4")
)

rmse_per_model_size = cbind(
  data.frame(model_size = rep(1:9,6)),
  rmse_per_model_size
)

ggplot(rmse_per_model_size, aes(x=rmse_mean, y=model_size, col=type, shape=type, size = 1)) +
  geom_point() +
  ggtitle("Average Train and Test RMSE V.S. Model Size")
```




```
train_rmse_df_1_min_models = apply(rmse_df_1[,1:9], 1, which.min)
train_rmse_df_2_min_models = apply(rmse_df_2[,1:9], 1, which.min)
train_rmse_df_4_min_models = apply(rmse_df_4[,1:9], 1, which.min)
test_rmse_df_1_min_models = apply(rmse_df_1[,10:18], 1, which.min)
test_rmse_df_2_min_models = apply(rmse_df_2[,10:18], 1, which.min)
test_rmse_df_4_min_models = apply(rmse_df_4[,10:18], 1, which.min)

train_rmse_chosen_count = data.frame(
  pred_1 = c(
    length(train_rmse_df_1_min_models[train_rmse_df_1_min_models == 1]),
    length(train_rmse_df_2_min_models[train_rmse_df_2_min_models == 1]),
    length(train_rmse_df_4_min_models[train_rmse_df_4_min_models == 1])
  ),
```

```
  pred_2 = c(
    length(train_rmse_df_1_min_models[train_rmse_df_1_min_models == 2]),
    length(train_rmse_df_2_min_models[train_rmse_df_2_min_models == 2]),
    length(train_rmse_df_4_min_models[train_rmse_df_4_min_models == 2])
  ),
  pred_3 = c(
    length(train_rmse_df_1_min_models[train_rmse_df_1_min_models == 3]),
    length(train_rmse_df_2_min_models[train_rmse_df_2_min_models == 3]),
    length(train_rmse_df_4_min_models[train_rmse_df_4_min_models == 3])
  ),
  pred_4 = c(
    length(train_rmse_df_1_min_models[train_rmse_df_1_min_models == 4]),
    length(train_rmse_df_2_min_models[train_rmse_df_2_min_models == 4]),
    length(train_rmse_df_4_min_models[train_rmse_df_4_min_models == 4])
  ),
  pred_5 = c(
    length(train_rmse_df_1_min_models[train_rmse_df_1_min_models == 5]),
    length(train_rmse_df_2_min_models[train_rmse_df_2_min_models == 5]),
    length(train_rmse_df_4_min_models[train_rmse_df_4_min_models == 5])
  ),
  pred_6 = c(
    length(train_rmse_df_1_min_models[train_rmse_df_1_min_models == 6]),
    length(train_rmse_df_2_min_models[train_rmse_df_2_min_models == 6]),
    length(train_rmse_df_4_min_models[train_rmse_df_4_min_models == 6])
  ),
  pred_7 = c(
    length(train_rmse_df_1_min_models[train_rmse_df_1_min_models == 7]),
    length(train_rmse_df_2_min_models[train_rmse_df_2_min_models == 7]),
    length(train_rmse_df_4_min_models[train_rmse_df_4_min_models == 7])
  ),
  pred_8 = c(
    length(train_rmse_df_1_min_models[train_rmse_df_1_min_models == 8]),
    length(train_rmse_df_2_min_models[train_rmse_df_2_min_models == 8]),
    length(train_rmse_df_4_min_models[train_rmse_df_4_min_models == 8])
  ),
  pred_9 = c(
    length(train_rmse_df_1_min_models[train_rmse_df_1_min_models == 9]),
    length(train_rmse_df_2_min_models[train_rmse_df_2_min_models == 9]),
    length(train_rmse_df_4_min_models[train_rmse_df_4_min_models == 9])
  )
)

test_rmse_chosen_count = data.frame(
  pred_1 = c(
    length(test_rmse_df_1_min_models[test_rmse_df_1_min_models == 1]),
    length(test_rmse_df_2_min_models[test_rmse_df_2_min_models == 1]),
    length(test_rmse_df_4_min_models[test_rmse_df_4_min_models == 1])
  ),
  pred_2 = c(
    length(test_rmse_df_1_min_models[test_rmse_df_1_min_models == 2]),
    length(test_rmse_df_2_min_models[test_rmse_df_2_min_models == 2]),
    length(test_rmse_df_4_min_models[test_rmse_df_4_min_models == 2])
  ),
```

```
  pred_3 = c(
    length(test_rmse_df_1_min_models[test_rmse_df_1_min_models == 3]),
    length(test_rmse_df_2_min_models[test_rmse_df_2_min_models == 3]),
    length(test_rmse_df_4_min_models[test_rmse_df_4_min_models == 3])
  ),
  pred_4 = c(
    length(test_rmse_df_1_min_models[test_rmse_df_1_min_models == 4]),
    length(test_rmse_df_2_min_models[test_rmse_df_2_min_models == 4]),
    length(test_rmse_df_4_min_models[test_rmse_df_4_min_models == 4])
  ),
  pred_5 = c(
    length(test_rmse_df_1_min_models[test_rmse_df_1_min_models == 5]),
    length(test_rmse_df_2_min_models[test_rmse_df_2_min_models == 5]),
    length(test_rmse_df_4_min_models[test_rmse_df_4_min_models == 5])
  ),
  pred_6 = c(
    length(test_rmse_df_1_min_models[test_rmse_df_1_min_models == 6]),
    length(test_rmse_df_2_min_models[test_rmse_df_2_min_models == 6]),
    length(test_rmse_df_4_min_models[test_rmse_df_4_min_models == 6])
  ),
  pred_7 = c(
    length(test_rmse_df_1_min_models[test_rmse_df_1_min_models == 7]),
    length(test_rmse_df_2_min_models[test_rmse_df_2_min_models == 7]),
    length(test_rmse_df_4_min_models[test_rmse_df_4_min_models == 7])
  ),
  pred_8 = c(
    length(test_rmse_df_1_min_models[test_rmse_df_1_min_models == 8]),
    length(test_rmse_df_2_min_models[test_rmse_df_2_min_models == 8]),
    length(test_rmse_df_4_min_models[test_rmse_df_4_min_models == 8])
  ),
  pred_9 = c(
    length(test_rmse_df_1_min_models[test_rmse_df_1_min_models == 9]),
    length(test_rmse_df_2_min_models[test_rmse_df_2_min_models == 9]),
    length(test_rmse_df_4_min_models[test_rmse_df_4_min_models == 9])
  )
)

rownames(train_rmse_chosen_count) = c("Sigma 1", "Sigma 2", "Sigma 4")
rownames(test_rmse_chosen_count) = c("Sigma 1", "Sigma 2", "Sigma 4")

library(knitr)
library(kableExtra)

kable(train_rmse_chosen_count) %>%
  kable_styling() %>%
  add_header_above(c("Number of Times the Model of Each Size Was Chosen for Each Value of Sigma (Traini
```

Number of Times the Model of Each Size Was Chosen for Each Value of Sigma (Training Data)									
	pred_1	pred_2	pred_3	pred_4	pred_5	pred_6	pred_7	pred_8	pred_9
Sigma 1	0	0	0	0	0	0	0	0	1000
Sigma 2	0	0	0	0	0	0	0	0	1000
Sigma 4	0	0	0	0	0	0	0	0	1000

```
kable(test_rmse_chosen_count) %>%
  kable_styling() %>%
  add_header_above(c("Number of Times the Model of Each Size Was Chosen for Each Value of Sigma (Testing
```

Number of Times the Model of Each Size Was Chosen for Each Value of Sigma (Testing Data)									
	pred_1	pred_2	pred_3	pred_4	pred_5	pred_6	pred_7	pred_8	pred_9
Sigma 1	0	0	1	7	13	517	207	116	139
Sigma 2	0	0	52	88	83	414	161	91	111
Sigma 4	0	16	242	164	111	215	108	77	67

Discussion

Out of the nine models that were fit in the simulation above, model 6 was the correct one (six predictors). Based on the tables above, we can see that model six never once had the lowest rmse (best fit) when the training data was used to fit the models. I fact, only the ninth model ever had the lowest rmse for every simulation.

This changes when the testing data was used to fit the model. In the table above where testing data was used, we can see that the 9the model is no longer the only model to have the lowest rmse in the simulations. Using a standard deviation of 1, and 2, the correct model, model six, had the largest number of lowest rmse values across 1000 simulations. When standard deviation grew to 4, the third model had the largest count, but the sixth model still wasn't far behind.

It seems that when using the testing data, we can accurately simulate the selection of the correct model, but as standard deviation grows, the other models are selected more frequently. So the more noise there is, the less likely the correct model will be selected based on the rmse value.

Simulation Study 3: Power

```
birthday = 19951015
set.seed(birthday)
```

Introduction

In this simulation I will investigate the power of the significance of regression test for simple linear regression. Power is the probability that a signal of a particular strength will be detected. I'll investigate these three values, and how they affect the power of a test:

- sample size, n
- signal strength, beta_1
- noise level, standard deviation (sigma)

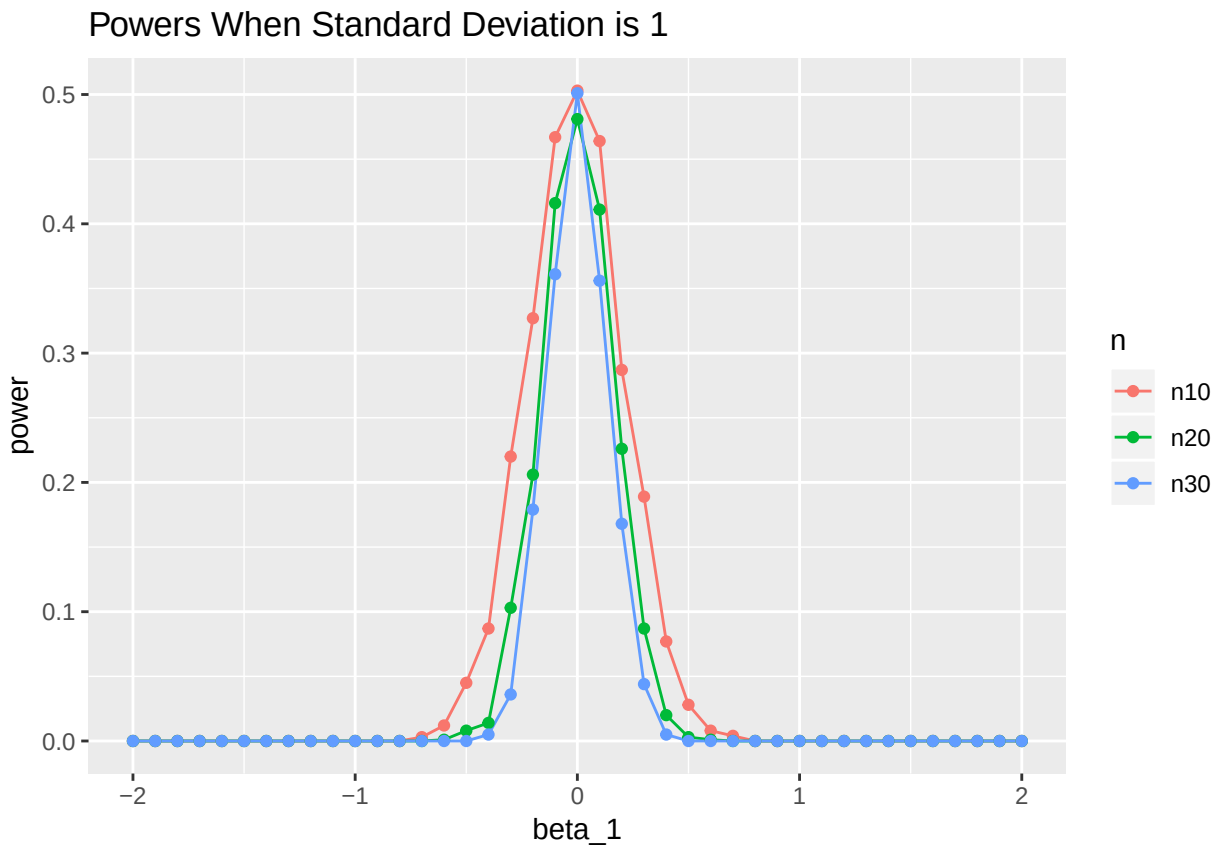
I'll be simulating a simple linear regression model with one predictor.

Methods

```r
SLR_simulate = function(num_sims=1000, n, sigma, beta_0, beta_1) {
  x_values = seq(0, 5, length = n)
  rejections = rep(0, num_sims)

  for(i in 1:num_sims) {
    eps = rnorm(n, mean=0, sd=sigma)
    y = beta_0 + beta_1*x_values + eps

    fit = lm(y ~ x_values, data = data.frame(cbind(y, x_values)))
    p_value = summary(fit)$coefficients[2,4]
    if (p_value > 0.5) {
      rejections[i] = 1
    }
  }

  sum(rejections) / num_sims
}

beta_0 = 0
beta_1 = seq(-2,2, by = 0.1)
sigma = c(1,2,4)
n = c(10,20,30)

powers = list(
  sigma1 = list(
    n10 = rep(0, length(beta_1)),
    n20 = rep(0, length(beta_1)),
    n30 = rep(0, length(beta_1))
  ),
  sigma2 = list(
    n10 = rep(0, length(beta_1)),
    n20 = rep(0, length(beta_1)),
    n30 = rep(0, length(beta_1))
  ),
  sigma4 = list(
    n10 = rep(0, length(beta_1)),
    n20 = rep(0, length(beta_1)),
    n30 = rep(0, length(beta_1))
  )
)

for (i in 1:length(sigma)) {
  for (j in 1:length(n)) {
    for (k in 1:length(beta_1)) {
      powers[[i]][[j]][k] = SLR_simulate(
        n = n[j],
        sigma = sigma[i],
        beta_0 = 0,
        beta_1 = beta_1[k]
      )
    }
  }
}
```

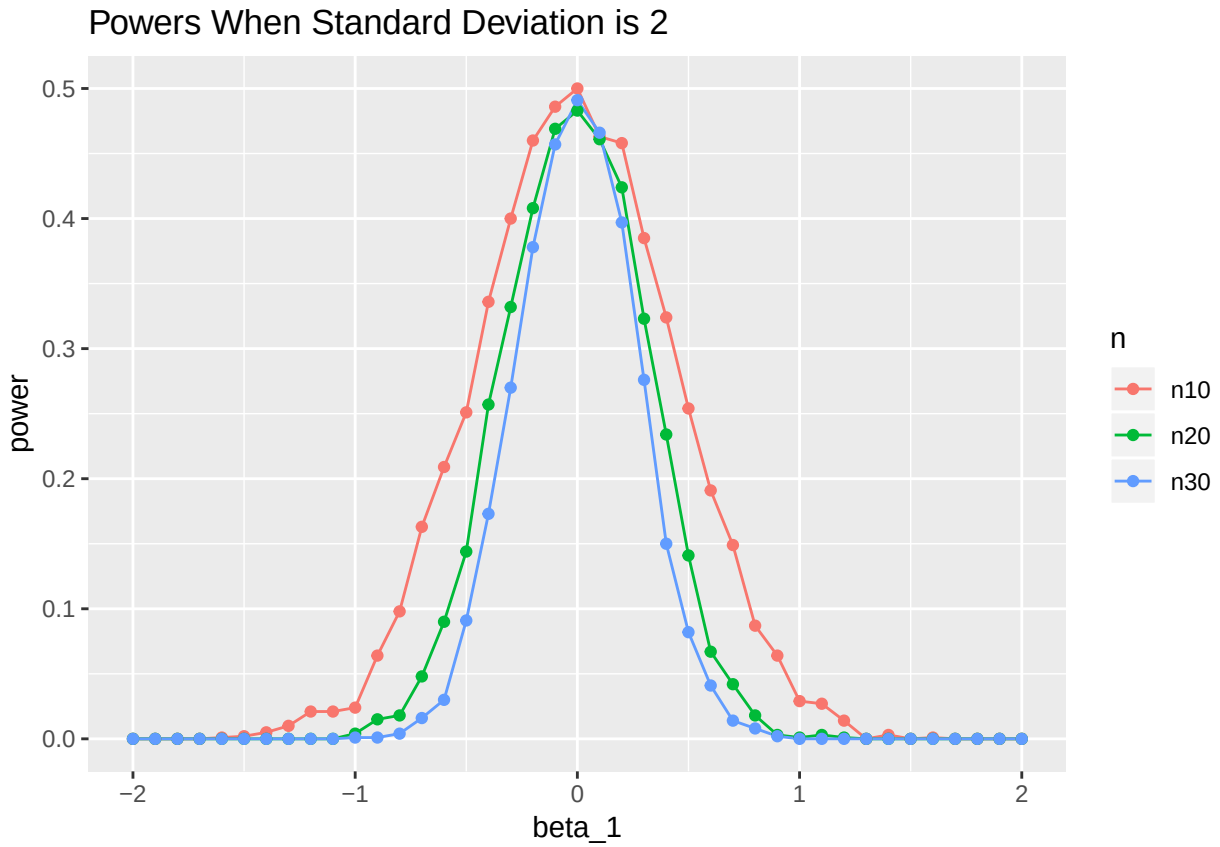
Results

```
sigma1_powers = data.frame(rbind(
  cbind(beta_1, power=powers$sigma1$n10, n=rep("n10", length(beta_1))),
  cbind(beta_1, power=powers$sigma1$n20, n=rep("n20", length(beta_1))),
  cbind(beta_1, power=powers$sigma1$n30, n=rep("n30", length(beta_1)))
))
sigma1_powers$power = as.numeric(as.character(sigma1_powers$power))
sigma1_powers$beta_1 = as.numeric(as.character(sigma1_powers$beta_1))
ggplot(sigma1_powers, aes(x = beta_1, y = power, col = n, group = n)) +
  geom_line() +
  geom_point() +
  ggtitle("Powers When Standard Deviation is 1")
```

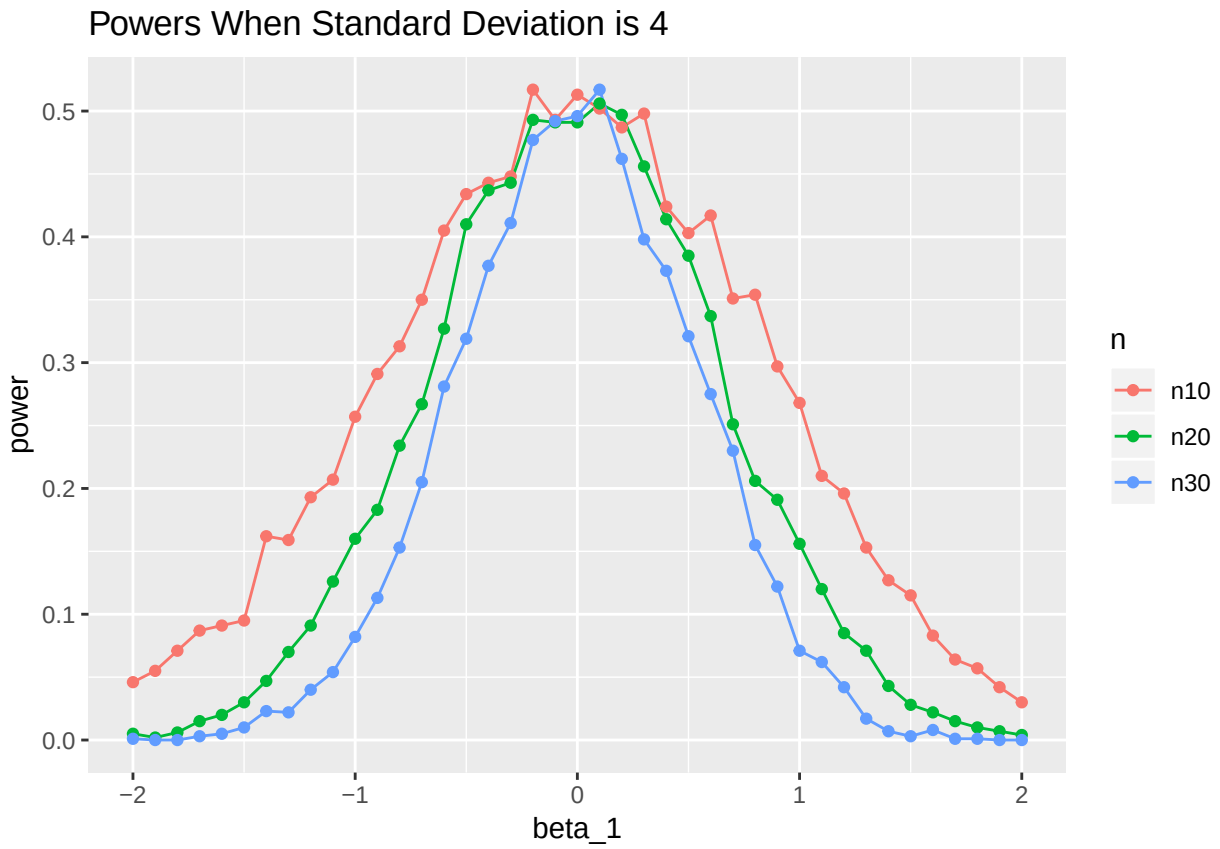



```
sigma2_powers = data.frame(rbind(
  cbind(beta_1, power=powers$sigma2$n10, n=rep("n10", length(beta_1))),
  cbind(beta_1, power=powers$sigma2$n20, n=rep("n20", length(beta_1))),
  cbind(beta_1, power=powers$sigma2$n30, n=rep("n30", length(beta_1)))
))
sigma2_powers$power = as.numeric(as.character(sigma2_powers$power))
sigma2_powers$beta_1 = as.numeric(as.character(sigma2_powers$beta_1))
ggplot(sigma2_powers, aes(x = beta_1, y = power, col = n, group = n)) +
  geom_line() +
```

```
  geom_point() +
  ggtitle("Powers When Standard Deviation is 2")
```




```
sigma4_powers = data.frame(rbind(
  cbind(beta_1, power=powers$sigma4$n10, n=rep("n10", length(beta_1))),
  cbind(beta_1, power=powers$sigma4$n20, n=rep("n20", length(beta_1))),
  cbind(beta_1, power=powers$sigma4$n30, n=rep("n30", length(beta_1)))
))
sigma4_powers$power = as.numeric(as.character(sigma4_powers$power))
sigma4_powers$beta_1 = as.numeric(as.character(sigma4_powers$beta_1))
ggplot(sigma4_powers, aes(x = beta_1, y = power, col = n, group = n)) +
  geom_line() +
  geom_point() +
  ggtitle("Powers When Standard Deviation is 4")
```



Discussion

Sample Size

Based on the plots above, we can see a relationship between power and sample size. For all three plots of varying standard deviation values, as the sample size increases, the width of the power distribution around beta_1 decreases. This suggests that as the sample size increases, the probability that a signal of a particular strength will be detected decreases.

Beta_1

Based on the plots above, we can see a relationship between beta_1 and power. The plots look as though power has a normal distribution about beta_1, as beta_1 increases from `-2` to `2`. I've used beta_1 in these simulations to control "sigma". `0` for beta_1 represents no signal, so it would make sense that the probability that a signal of a particular strength would be detected in the model would be greatest at `0` signal, and decrease in both the positive and negative direction.

Standard Deviation

Based on the plots above, we can see a relationship between standard deviation and power. As standard deviation increases, so does the width of the power distribution around beta_1. This suggests that as standard deviation increases, the probability that a signal of a particular strength will be detected increases.

Simulations

1000 simulations seemed to be sufficient in order to shape the plots above. There is not much variation in the power distributions, and it seems as though the amount of simulations was enough to gather consistent power readings.