Simulation - Applied Stats in R

L Wu

Simulation Study 1: Significance of Regression

Introduction

The goal of this simulation study is to examine the significance of regression using two difference models: The significant model and the non-significant model.

The significant model:

```
  Yi=β0+β1xi1+β2xi2+β3xi3+ϵi
```

where ϵi∼N(0,σ2) and

β0=3,
β1=1,
β2=1,
β3=1.

The non significnat model:

```
  Yi=β0+β1xi1+β2xi2+β3xi3+ϵi
```

where ϵi∼N(0,σ2) and

β0=3,
β1=0,
β2=0,
β3=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 σ∈(1,5,10) We 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
- R2

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.

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.1.3
```

```
library(readr)
```

```
## Warning: package 'readr' was built under R version 4.1.3
```

```
birthday = 19990101
set.seed(birthday)
study_1 = read_csv("study_1.csv")
```

```
## Rows: 25 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (4): y, x1, x2, x3
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(study_1)
```

```
## # A tibble: 6 x 4
##       y    x1    x2    x3
##   <dbl> <dbl> <dbl> <dbl>
## 1     0 -1.54   3    2.96
## 2     0 -1.61   2.9  2.71
## 3     0 -4.56   2.9  2.62
## 4     0 -2.28   2.5  2.41
## 5     0 -2.76   3    2.77
## 6     0 -3.01   2.9  2.28
```

```
s_beta_0 = 3
s_beta_1 = 1
s_beta_2 = 1
s_beta_3 = 1

ns_beta_0 = 3
ns_beta_1 = 0
ns_beta_2 = 0
ns_beta_3 = 0

n = 25
p = 3
x0 = rep(1, n)
x1 = study_1$x1
x2 = study_1$x2
x3 = study_1$x3
sigma = c(1,5,10)
simulations = 2000
```

First Simulation

```r
sim_1 = data.frame(s_fstat = rep(0,simulations),
                   s_pval = rep(0,simulations),
                   s_r2 = rep(0,simulations),
                   ns_fstat = rep(0,simulations),
                   ns_pval = rep(0,simulations),
                   ns_r2 = rep(0,simulations)
                   )

for (i in 1:2000) {
  eps = rnorm(n, mean = 0, sd = sigma[1])
  study_1$y = s_beta_0 + s_beta_1 * x1 + s_beta_2 * x2 + s_beta_3 * x3 + eps

  s_lr_model = lm(y ~., data = study_1)
  sim_1$s_fstat[i] = summary(s_lr_model)$fstatistic[[1]]
  sim_1$s_pval[i] = pf(summary(s_lr_model)$fstatistic[[1]],df1 = p, df2 = n - (p+1), lower.tail = FALSE)
  sim_1$s_r2[i] = summary(s_lr_model)$r.squared

  study_1$y = ns_beta_0 + ns_beta_1 * x1 + ns_beta_2 * x2 + ns_beta_3 * x3 + eps
  ns_lr_model = lm(y ~., data = study_1)
  sim_1$ns_fstat[i] = summary(ns_lr_model)$fstatistic[[1]]
  sim_1$ns_pval[i] = pf(summary(ns_lr_model)$fstatistic[[1]],df1 = p, df2 = n - (p+1), lower.tail = FALSE)
  sim_1$ns_r2[i] = summary(ns_lr_model)$r.squared
}
```

Second Simulation

```r
sim_2 = data.frame(s_fstat = rep(0,simulations),
                   s_pval = rep(0,simulations),
                   s_r2 = rep(0,simulations),
                   ns_fstat = rep(0,simulations),
                   ns_pval = rep(0,simulations),
                   ns_r2 = rep(0,simulations)
                   )

for (i in 1:2000) {
  eps = rnorm(n, mean = 0, sd = sigma[2])
  study_1$y = s_beta_0 + s_beta_1 * x1 + s_beta_2 * x2 + s_beta_3 * x3 + eps

  s_lr_model = lm(y ~., data = study_1)
  sim_2$s_fstat[i] = summary(s_lr_model)$fstatistic[[1]]
  sim_2$s_pval[i] = pf(summary(s_lr_model)$fstatistic[[1]],df1 = p, df2 = n - (p+1), lower.tail = FALSE)
  sim_2$s_r2[i] = summary(s_lr_model)$r.squared

  study_1$y = ns_beta_0 + ns_beta_1 * x1 + ns_beta_2 * x2 + ns_beta_3 * x3 + eps
  ns_lr_model = lm(y ~., data = study_1)
  sim_2$ns_fstat[i] = summary(ns_lr_model)$fstatistic[[1]]
  sim_2$ns_pval[i] = pf(summary(ns_lr_model)$fstatistic[[1]],df1 = p, df2 = n - (p+1), lower.tail = FALSE)
  sim_2$ns_r2[i] = summary(ns_lr_model)$r.squared
}
```

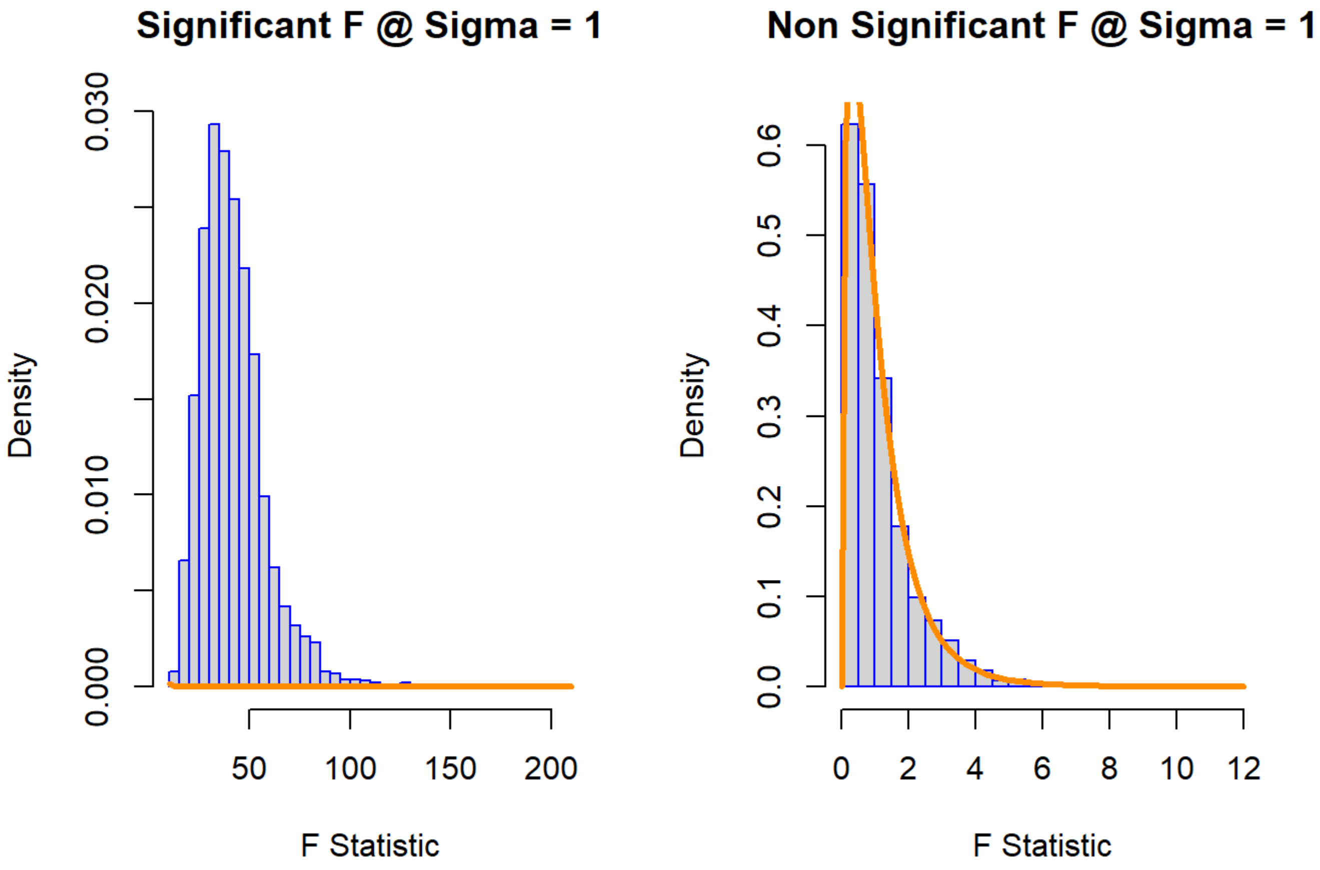
Third Simulation

```
sim_3 = data.frame(s_fstat = rep(0,simulations),
                   s_pval = rep(0,simulations),
                   s_r2 = rep(0,simulations),
                   ns_fstat = rep(0,simulations),
                   ns_pval = rep(0,simulations),
                   ns_r2 = rep(0,simulations)
                   )

for (i in 1:2000) {
  eps = rnorm(n, mean = 0, sd = sigma[3])
  study_1$y = s_beta_0 + s_beta_1 * x1 + s_beta_2 * x2 + s_beta_3 * x3 + eps

  s_lr_model = lm(y ~., data = study_1)
  sim_3$s_fstat[i] = summary(s_lr_model)$fstatistic[[1]]
  sim_3$s_pval[i] = pf(summary(s_lr_model)$fstatistic[[1]],df1 = p, df2 = n - (p+1), lower.tail = FALSE)
  sim_3$s_r2[i] = summary(s_lr_model)$r.squared

  study_1$y = ns_beta_0 + ns_beta_1 * x1 + ns_beta_2 * x2 + ns_beta_3 * x3 + eps
  ns_lr_model = lm(y ~., data = study_1)
  sim_3$ns_fstat[i] = summary(ns_lr_model)$fstatistic[[1]]
  sim_3$ns_pval[i] = pf(summary(ns_lr_model)$fstatistic[[1]],df1 = p, df2 = n - (p+1), lower.tail = FALSE)
  sim_3$ns_r2[i] = summary(ns_lr_model)$r.squared
}
```

Results

F Stats at Sigma 1,5, and 10

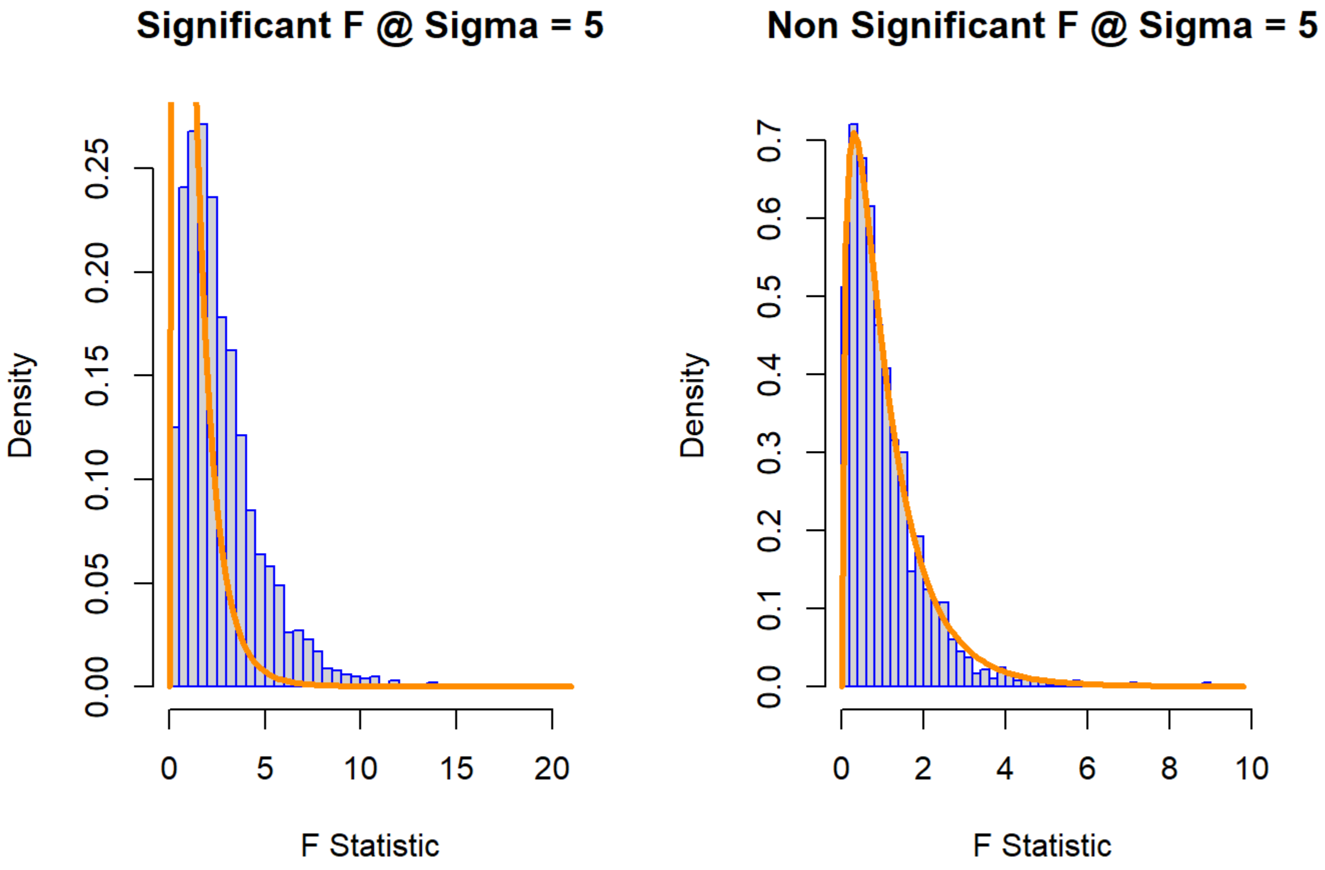
```
par(mfrow = c(1,2));
hist(sim_1$s_fstat,
     prob = TRUE,
     breaks = 40,
     main = "Significant F @ Sigma = 1",
     border = "blue",
     xlab = "F Statistic",
     );
x = sim_1$s_fstat;
curve(df(x, df1 = p, df2 = n - (p+1)), col = "darkorange", add = TRUE, lwd = 3);

hist(sim_1$ns_fstat,
     main = "Non Significant F @ Sigma = 1",
     breaks = 40,
     border = "blue",
     xlab = "F Statistic",
     prob = TRUE
     );
x = sim_1$ns_fstat;
curve(df(x, df1 = p, df2 = n - (p+1)), col = "darkorange", add = TRUE, lwd = 3)
```



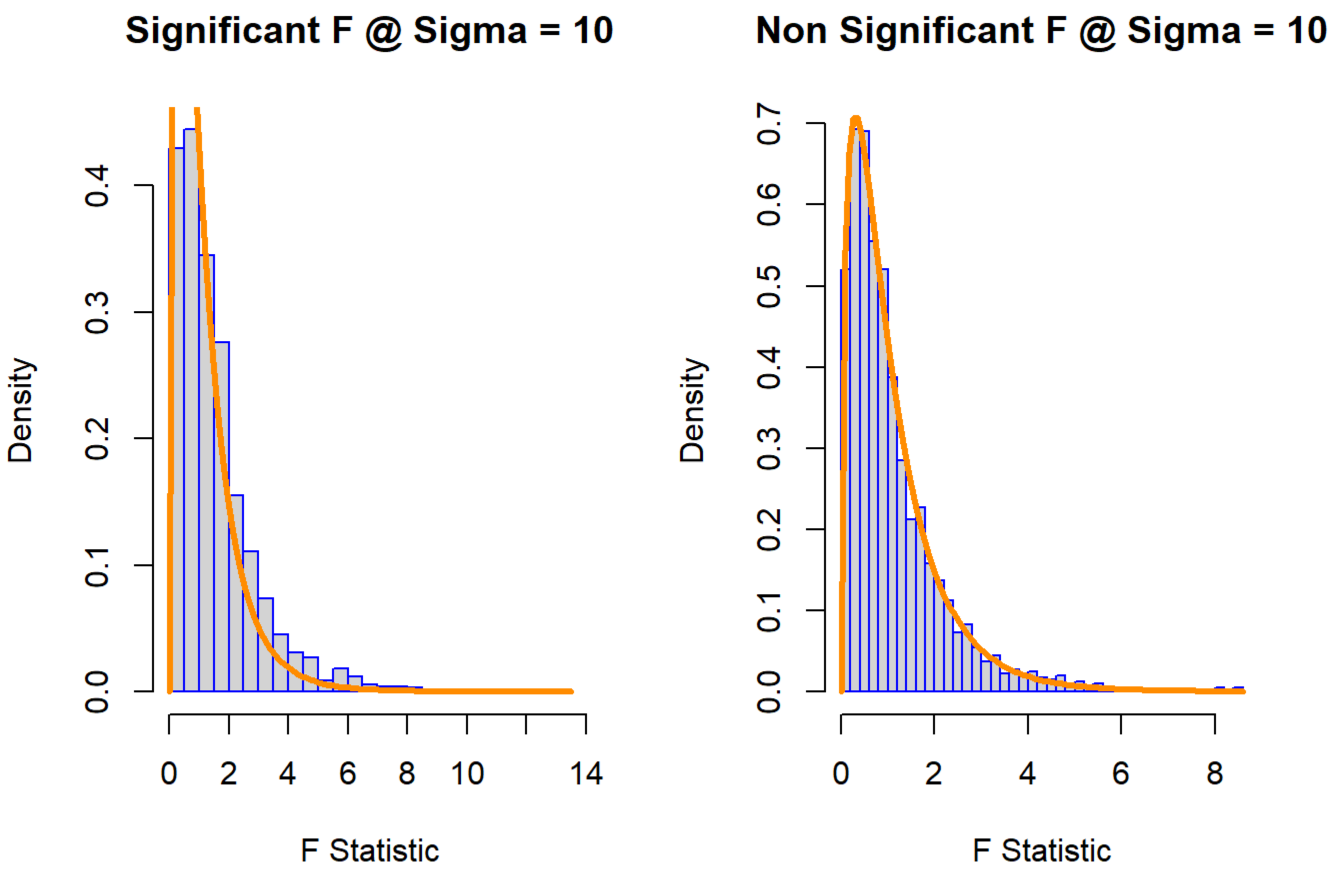
Significant F @ Sigma = 1

Non Significant F @ Sigma = 1

```
par(mfrow = c(1,2));
hist(sim_2$s_fstat,
     prob = TRUE,
     breaks = 40,
     main = "Significant F @ Sigma = 5",
     border = "blue",
     xlab = "F Statistic",
     );
x = sim_2$s_fstat;
curve(df(x, df1 = p, df2 = n - (p+1)), col = "darkorange", add = TRUE, lwd = 3);

hist(sim_2$ns_fstat,
     main = "Non Significant F @ Sigma = 5",
     breaks = 40,
     border = "blue",
     xlab = "F Statistic",
     prob = TRUE
     );
x = sim_2$ns_fstat;
curve(df(x, df1 = p, df2 = n - (p+1)), col = "darkorange", add = TRUE, lwd = 3)
```

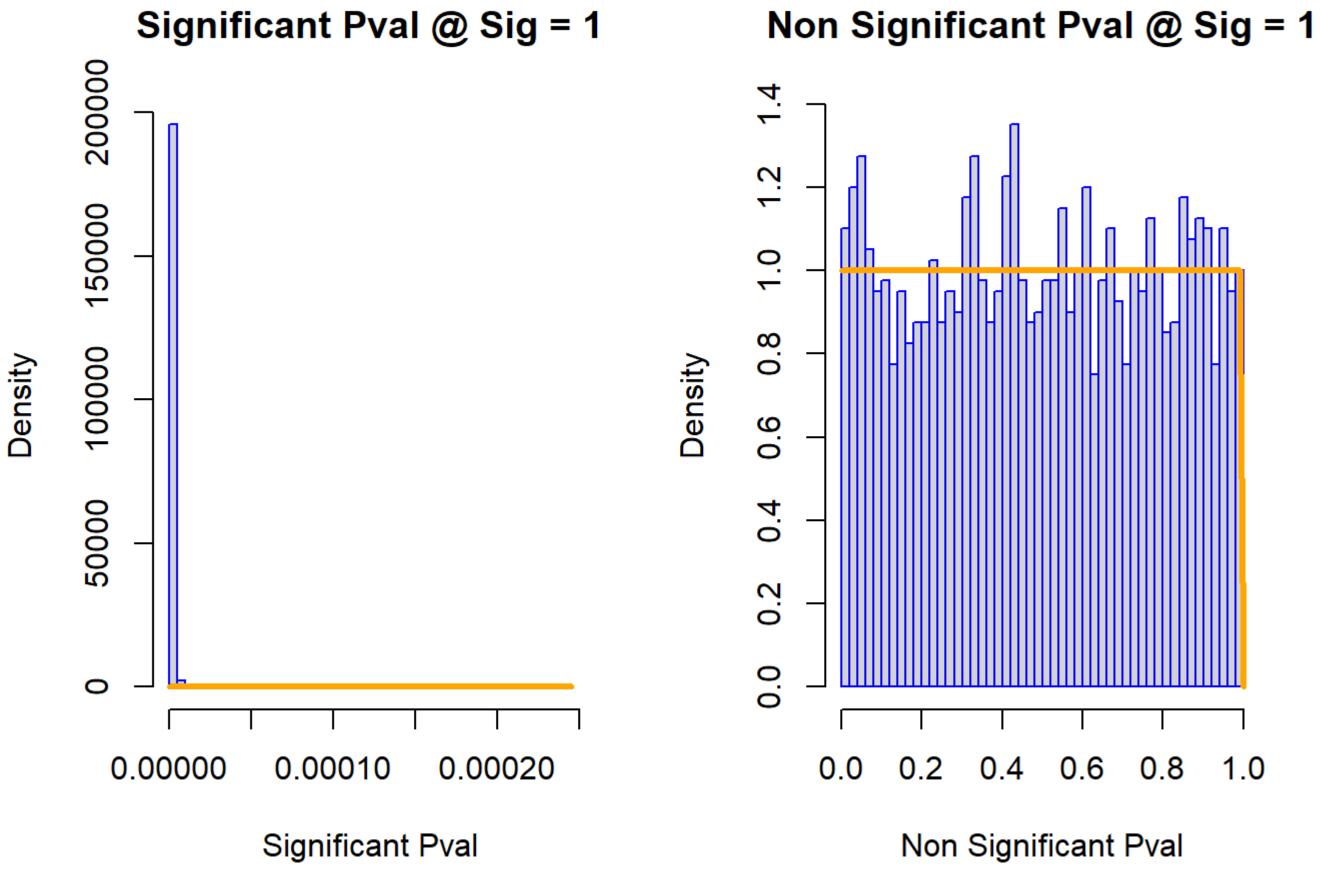


```
par(mfrow = c(1,2));
hist(sim_3$s_fstat,
     prob = TRUE,
     breaks = 40,
     main = "Significant F @ Sigma = 10",
     border = "blue",
     xlab = "F Statistic",
     );
x = sim_3$s_fstat;
curve(df(x, df1 = p, df2 = n - (p+1)), col = "darkorange", add = TRUE, lwd = 3);

hist(sim_3$ns_fstat,
     main = "Non Significant F @ Sigma = 10",
     breaks = 40,
     border = "blue",
     xlab = "F Statistic",
     prob = TRUE
     );
x = sim_3$ns_fstat;
curve(df(x, df1 = p, df2 = n - (p+1)), col = "darkorange", add = TRUE, lwd = 3)
```

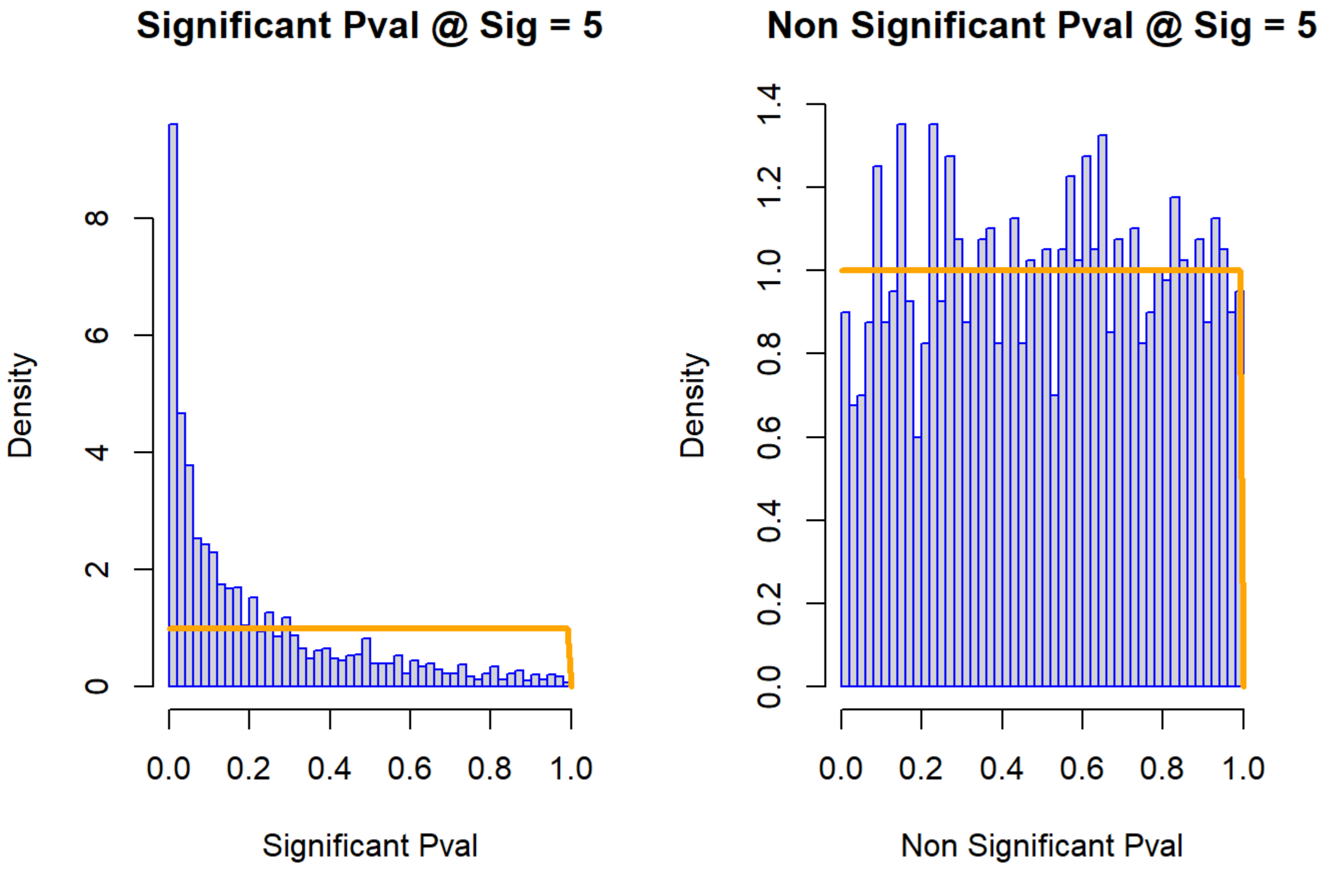


P-value at Sigma 1,5, and 10

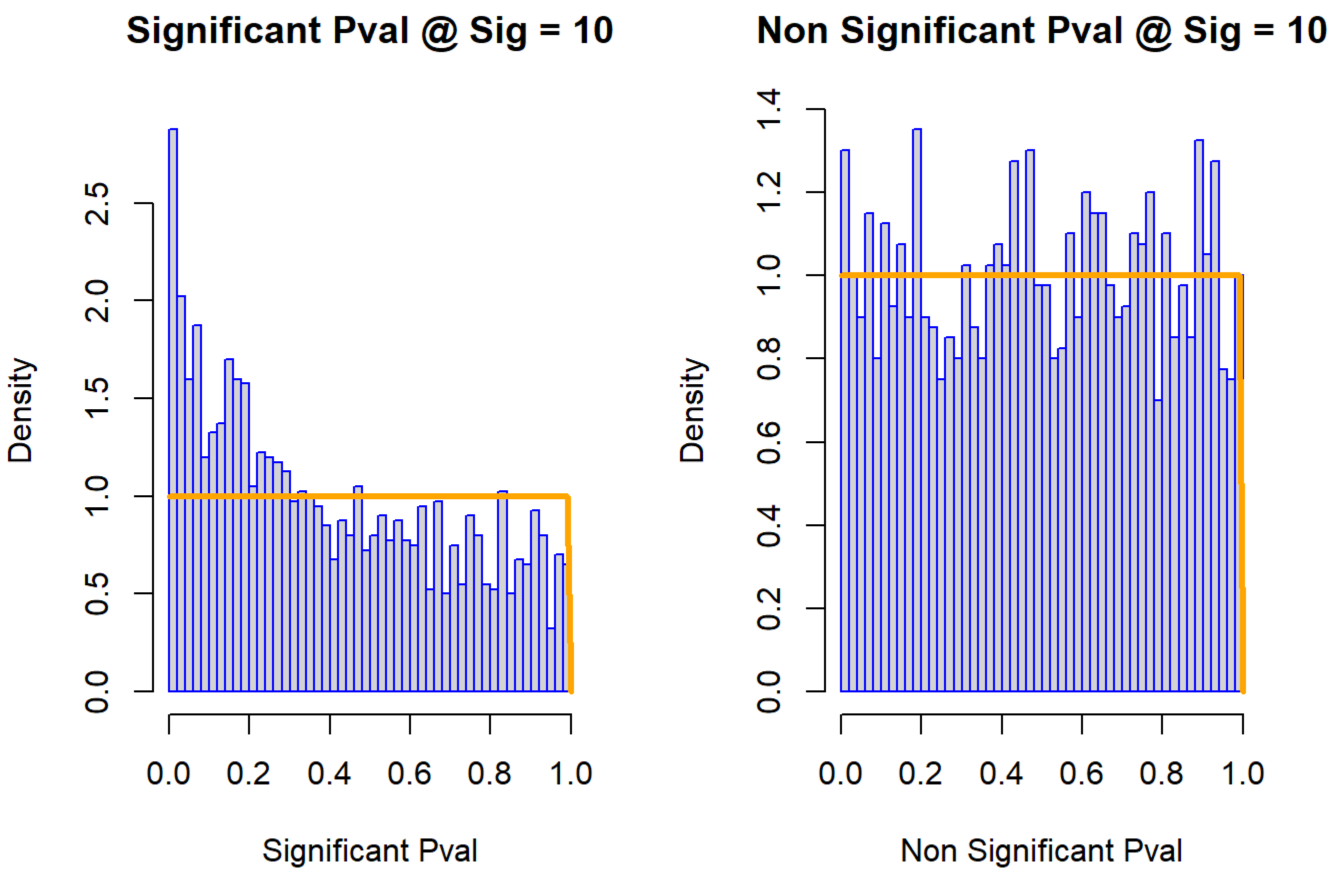
```
par(mfrow = c(1, 2))
hist(
      sim_1$s_pval,
      main = 'Significant Pval @ Sig = 1',
      border = 'blue',
      xlab = 'Significant Pval',
      prob = TRUE,
      breaks = 40
     )
x = sim_1$ns_pval
curve(dunif(x), col='orange', add=TRUE, lwd=3)
hist(
      sim_1$ns_pval,
      main = 'Non Significant Pval @ Sig = 1',
      border = 'blue',
      xlab = 'Non Significant Pval',
      prob = TRUE,
      breaks = 40
     )
x = sim_1$ns_pval
curve(dunif(x), col='orange', add=TRUE, lwd=3)
```



```
par(mfrow = c(1, 2))
hist(
      sim_2$s_pval,
      main = 'Significant Pval @ Sig = 5',
      border = 'blue',
      xlab = 'Significant Pval',
      prob = TRUE,
      breaks = 40
     )
x = sim_2$ns_pval
curve(dunif(x), col='orange', add=TRUE, lwd=3)
hist(
      sim_2$ns_pval,
      main = 'Non Significant Pval @ Sig = 5',
      border = 'blue',
      xlab = 'Non Significant Pval',
      prob = TRUE,
      breaks = 40
     )
x = sim_2$ns_pval
curve(dunif(x), col='orange', add=TRUE, lwd=3)
```

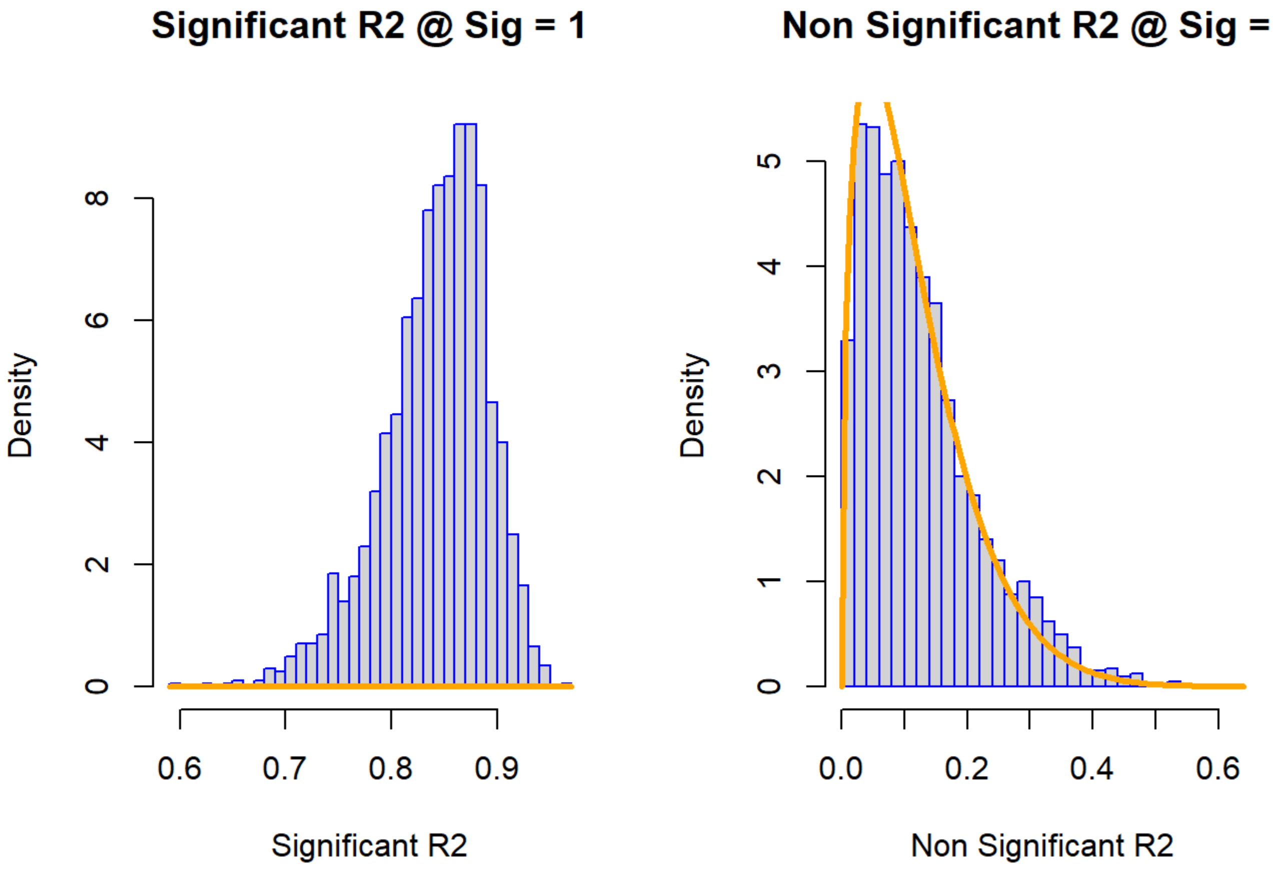


```
par(mfrow = c(1, 2))
hist(
      sim_3$s_pval,
      main = 'Significant Pval @ Sig = 10',
      border = 'blue',
      xlab = 'Significant Pval',
      prob = TRUE,
      breaks = 40
     )
x = sim_3$ns_pval
curve(dunif(x), col='orange', add=TRUE, lwd=3)
hist(
      sim_3$ns_pval,
      main = 'Non Significant Pval @ Sig = 10',
      border = 'blue',
      xlab = 'Non Significant Pval',
      prob = TRUE,
      breaks = 40
     )
x = sim_3$ns_pval
curve(dunif(x), col='orange', add=TRUE, lwd=3)
```

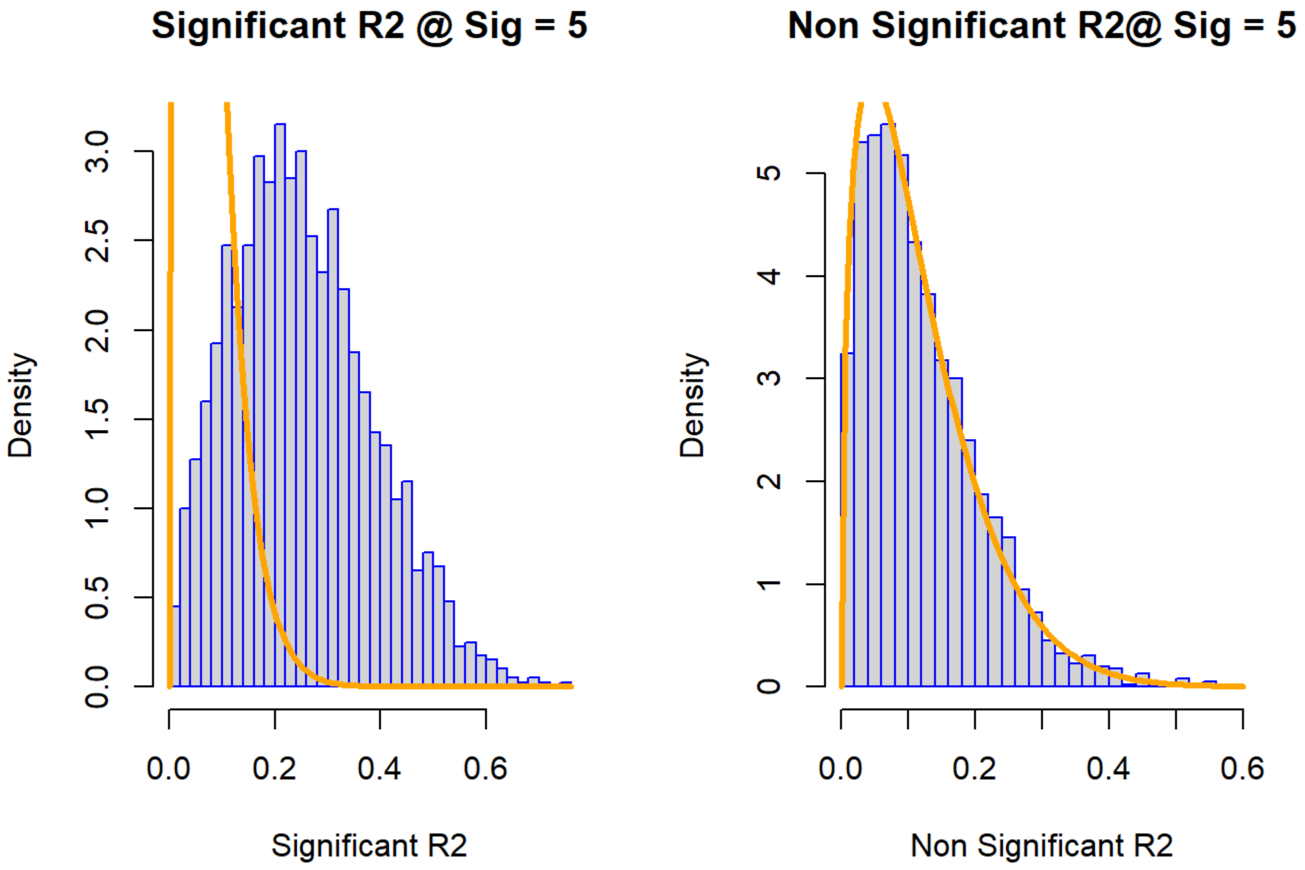


R Squared at Sigma 1,5, and 10

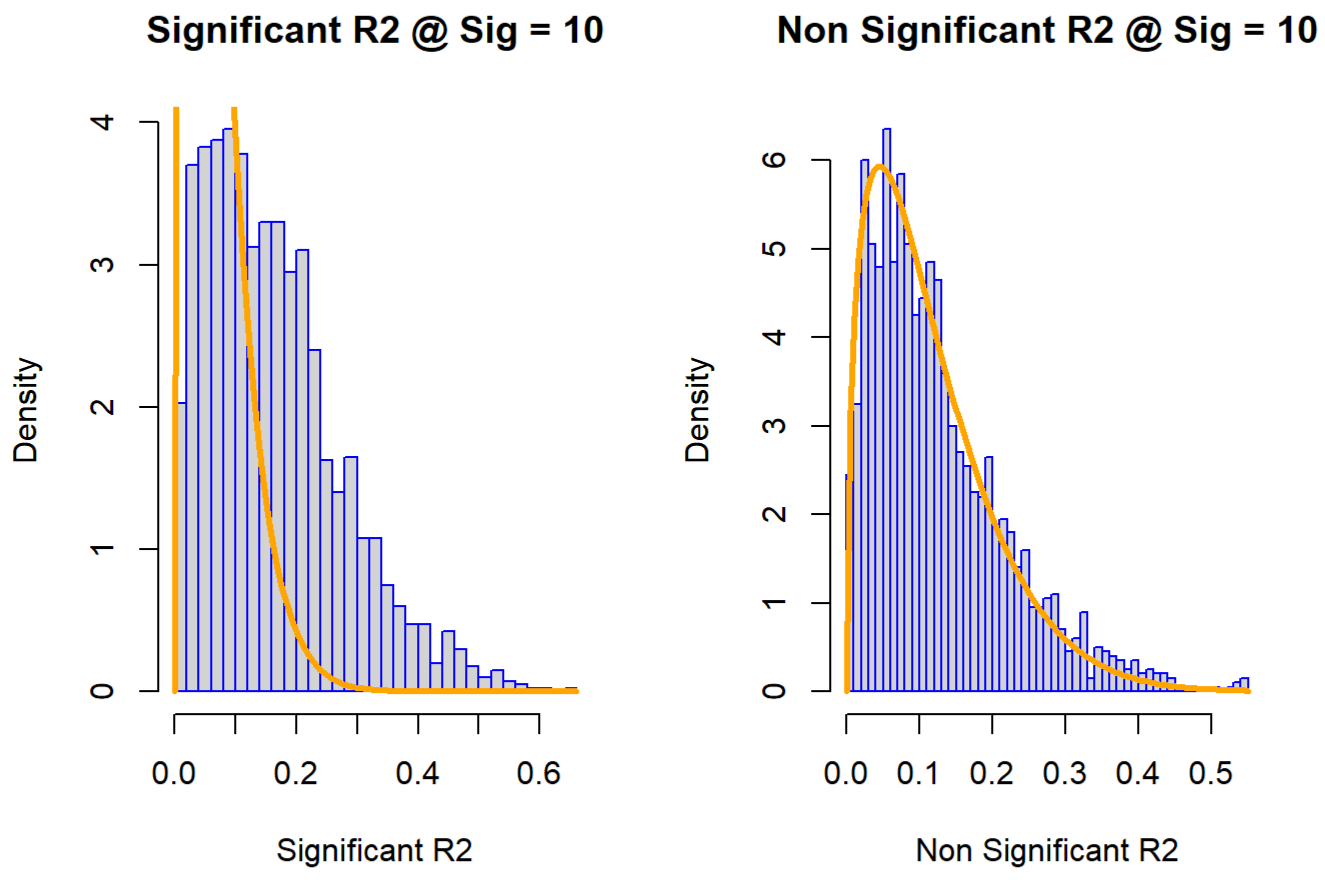
```
par(mfrow = c(1, 2))
hist(
      sim_1$s_r2,
      main = 'Significant R2 @ Sig = 1',
      border = 'blue',
      xlab = 'Significant R2',
      prob = TRUE,
      breaks = 40
     )
x = sim_1$s_r2
curve(dbeta(x,p/2,(n-(p+1)/2)), col="orange",add=TRUE,lwd=3)
hist(
      sim_1$ns_r2,
      main = 'Non Significant R2 @ Sig = 1',
      border = 'blue',
      xlab = 'Non Significant R2',
      prob = TRUE,
      breaks = 40
     )
x = sim_1$ns_r2
curve(dbeta(x,p/2,(n-p+1)/2), col="orange",add=TRUE,lwd=3)
```



```
par(mfrow = c(1, 2))
hist(
      sim_2$s_r2,
      main = 'Significant R2 @ Sig = 5',
      border = 'blue',
      xlab = 'Significant R2',
      prob = TRUE,
      breaks = 40
     )
x = sim_2$s_r2
curve(dbeta(x,p/2,(n-(p+1)/2)), col="orange",add=TRUE,lwd=3)
hist(
      sim_2$ns_r2,
      main = 'Non Significant R2@ Sig = 5',
      border = 'blue',
      xlab = 'Non Significant R2',
      prob = TRUE,
      breaks = 40
     )
x = sim_2$ns_r2
curve(dbeta(x,p/2,(n-p+1)/2), col="orange",add=TRUE,lwd=3)
```



```
par(mfrow = c(1, 2))
hist(
      sim_3$s_r2,
      main = 'Significant R2 @ Sig = 10',
      border = 'blue',
      xlab = 'Significant R2',
      prob = TRUE,
      breaks = 40
     )
x = sim_3$s_r2
curve(dbeta(x,p/2,(n-(p+1)/2)), col="orange",add=TRUE,lwd=3)
hist(
      sim_3$ns_r2,
      main = 'Non Significant R2 @ Sig = 10',
      border = 'blue',
      xlab = 'Non Significant R2',
      prob = TRUE,
      breaks = 40
     )
x = sim_3$ns_r2
curve(dbeta(x,p/2,(n-p+1)/2), col="orange",add=TRUE,lwd=3)
```



Discussion

For this Significance of Regression test, we know the true distribution of all the non significant models. This is because our null hypothesis is that all the models are insignificant. One thing that struck out to me is that for all the significant models, as sigma increases, we get closer and closer to the true distribution of our model. This is especially evident when we look at the distribution for Significant F stat at sigma 1 versus Significant F stat at sigma 1. In conclusion, all the non significant models at every sigma value follows a true distribution. However, the significant models will only get closer to the true distribution as sigma increases.

F statistics, p-value, and R2 are related to sigma. This relationship is not apparent in the non significant mode. However, it is very apparent in the significant model. We can see from the plots that as sigma increases, our model becomes less significant and holds less explanatory power.

Simulation Study 2: Using RMSE for Selection

Introduction

For this simulation study, we will investigate how well we can use RMSE to select the “best” model. To do this, we will simulate from the model:

- $Yi=\beta0+\beta1xi1+\beta2xi2+\beta3xi3+\beta4xi4+\beta5xi5+\beta6xi6+\epsilon i$

where $\epsilon i \sim N(0,\sigma 2)$ and

$\beta0=0$,
$\beta1=3$,
$\beta2=-4$,
$\beta3=1.6$,
$\beta4=-1.1$,
$\beta5=0.7$,
$\beta6=0.5$.

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

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

```
library(readr)
birthday = 19990101
set.seed(birthday)
study_2 = read_csv("study_2.csv")
```

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

```
head(study_2)
```

```
## # A tibble: 6 x 10
##       y     x1    x2     x3     x4     x5    x6     x7      x8      x9
##   <dbl>  <dbl> <dbl>  <dbl>  <dbl>  <dbl> <dbl>  <dbl>   <dbl>   <dbl>
## 1     0 -4.62   0.73 0.533  0.568  0.908    0.4 0.0806  1.42    0.0332
## 2     0  0.936  0.69 0.476  0.549  0.516    0.5 0.197   0.0788 -2.01
## 3     0 -0.109 -0.78 0.608  0.237  0.923   -0.3 0.489   0.803   0.146
## 4     0 -2.01   0.36 0.130  0.0302 0.417   -0.6 0.734   0.199   0.692
## 5     0 -4.92   0.12 0.0144 0.410  0.0416   0.5 0.110   0.0114  1.01
## 6     0 -4.29  -0.01 0.0001 0.254  0.370   -1.2 0.585  -0.742   1.19
```

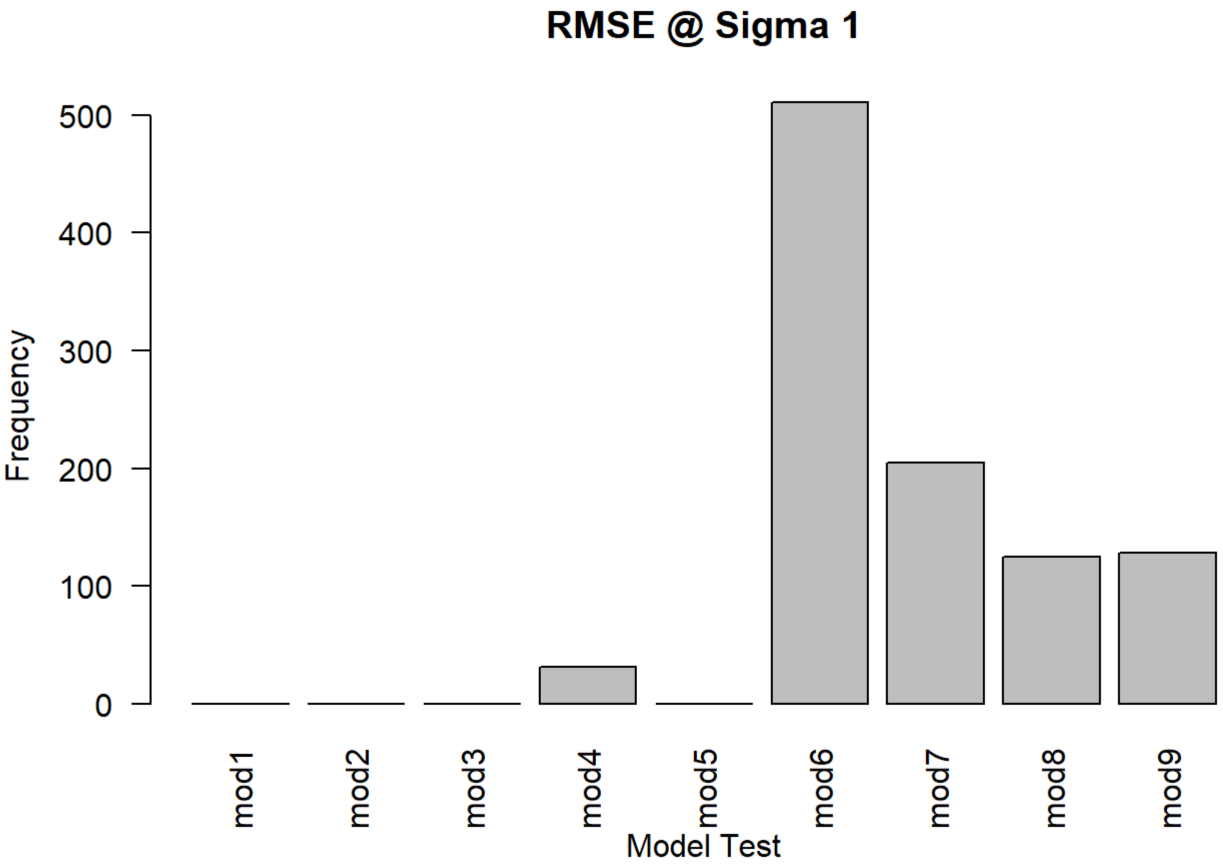
```r
beta_0 = 0
beta_1 = 3
beta_2 = -4
beta_3 = 1.6
beta_4 = -1.1
beta_5 = 0.7
beta_6 = 0.5

n = 500
sigma = c(1,2,4)
x0 = rep(1,n)
x1 = study_2$x1
x2 = study_2$x2
x3 = study_2$x3
x4 = study_2$x4
x5 = study_2$x5
x6 = study_2$x6

simulations = 1000

cal_rmse = function(actual, pred) {
  sqrt(mean((actual - pred)^2));
}
```

Running the Simulations

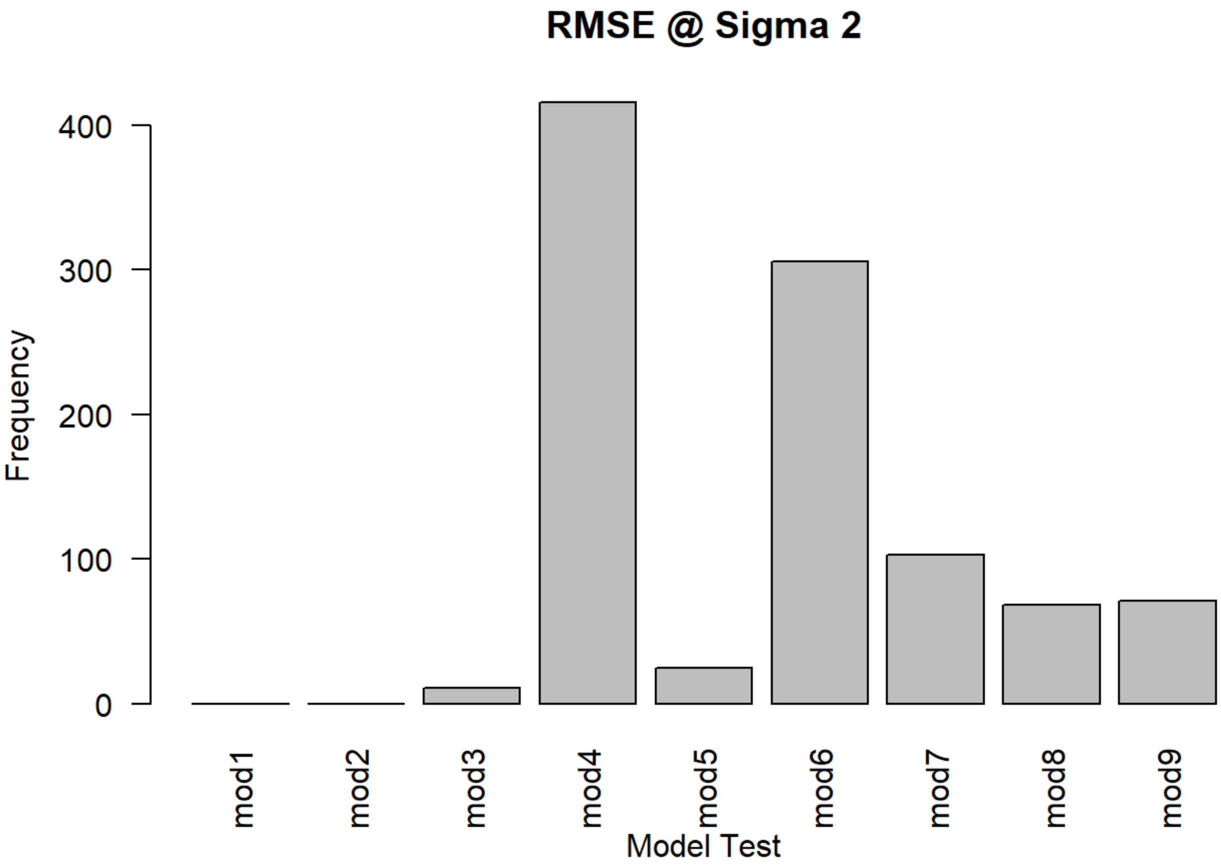
```
train_1 = data.frame(
  mod1 = rep(0,simulations),
  mod2 = rep(0,simulations),
  mod3 = rep(0,simulations),
  mod4 = rep(0,simulations),
  mod5 = rep(0,simulations),
  mod6 = rep(0,simulations),
  mod7 = rep(0,simulations),
  mod8 = rep(0,simulations),
  mod9 = rep(0,simulations)
)

train_2 = 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)
)

train_4 = 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)
)

test_1 = 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)
)

test_2 = 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)
)

test_4 = 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)
)

for(sig in sigma) {
  for(i in 1:simulations) {
    eps = rnorm(n,0,sig)
    study_2$y = beta_0 + beta_1 * x1 + beta_2 * x2 + beta_3 * x3 + beta_4 * x4 + beta_5 * x5 + beta_6 * x6
 + eps
    train_index = sample(250)
    train = study_2[train_index,]
    test = study_2[-train_index,]

    lr1 = lm(y~x1,data = train)
    lr2 = lm(y~x1 + x2,data = train)
    lr3 = lm(y~x1 + x2 + x3,data = train)
    lr4 = lm(y~x1 + x2 + x3 + x4,data = train)
    lr5 = lm(y ~ x1 + x2 + x3 + x4 + x5, data = train)
    lr6 = lm(y ~ x1 + x2 + x3 + x4 + x5 + x6, data = train)
    lr7 = lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7, data = train)
    lr8 = lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8, data = train)
    lr9 = lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9, data = train)

    if(sig == 1){
      train_1$mod1[i] = cal_rmse(train$y, predict(lr1, train))
      test_1$mod1[i] = cal_rmse(test$y, predict(lr1, test))
      train_1$mod2[i] = cal_rmse(train$y, predict(lr2, train))
      test_1$mod2[i] = cal_rmse(test$y, predict(lr2, test))
      train_1$mod3[i] = cal_rmse(train$y, predict(lr3, train))
      test_1$mod3[i] = cal_rmse(test$y, predict(lr3, test))
      train_1$mod4[i] = cal_rmse(train$y, predict(lr4, train))
      test_1$mod4[i] = cal_rmse(train$y, predict(lr4, train))
      train_1$mod5[i] = cal_rmse(train$y, predict(lr5, train))
      test_1$mod5[i] = cal_rmse(test$y, predict(lr5, test))
      train_1$mod6[i] = cal_rmse(train$y, predict(lr6, train))
      test_1$mod6[i] = cal_rmse(test$y, predict(lr6, test))
      train_1$mod7[i] = cal_rmse(train$y, predict(lr7, train))
      test_1$mod7[i] = cal_rmse(test$y, predict(lr7, test))
      train_1$mod8[i] = cal_rmse(train$y, predict(lr8, train))
      test_1$mod8[i] = cal_rmse(test$y, predict(lr8, test))
      train_1$mod9[i] = cal_rmse(train$y, predict(lr9, train))
      test_1$mod9[i] = cal_rmse(test$y, predict(lr9, test))
    } else if(sig == 2){
      train_2$mod1[i] = cal_rmse(train$y, predict(lr1, train))
      test_2$mod1[i] = cal_rmse(test$y, predict(lr1, test))
      train_2$mod2[i] = cal_rmse(train$y, predict(lr2, train))
      test_2$mod2[i] = cal_rmse(test$y, predict(lr2, test))
      train_2$mod3[i] = cal_rmse(train$y, predict(lr3, train))
      test_2$mod3[i] = cal_rmse(test$y, predict(lr3, test))
      train_2$mod4[i] = cal_rmse(train$y, predict(lr4, train))
      test_2$mod4[i] = cal_rmse(train$y, predict(lr4, train))
      train_2$mod5[i] = cal_rmse(train$y, predict(lr5, train))
      test_2$mod5[i] = cal_rmse(test$y, predict(lr5, test))
      train_2$mod6[i] = cal_rmse(train$y, predict(lr6, train))
      test_2$mod6[i] = cal_rmse(test$y, predict(lr6, test))
      train_2$mod7[i] = cal_rmse(train$y, predict(lr7, train))
      test_2$mod7[i] = cal_rmse(test$y, predict(lr7, test))
      train_2$mod8[i] = cal_rmse(train$y, predict(lr8, train))
      test_2$mod8[i] = cal_rmse(test$y, predict(lr8, test))
      train_2$mod9[i] = cal_rmse(train$y, predict(lr9, train))
      test_2$mod9[i] = cal_rmse(test$y, predict(lr9, test))
    } else {
      train_4$mod1[i] = cal_rmse(train$y, predict(lr1, train))
      test_4$mod1[i] = cal_rmse(test$y, predict(lr1, test))
      train_4$mod2[i] = cal_rmse(train$y, predict(lr2, train))
      test_4$mod2[i] = cal_rmse(test$y, predict(lr2, test))
      train_4$mod3[i] = cal_rmse(train$y, predict(lr3, train))
      test_4$mod3[i] = cal_rmse(test$y, predict(lr3, test))
      train_4$mod4[i] = cal_rmse(train$y, predict(lr4, train))
      test_4$mod4[i] = cal_rmse(train$y, predict(lr4, train))
      train_4$mod5[i] = cal_rmse(train$y, predict(lr5, train))
      test_4$mod5[i] = cal_rmse(test$y, predict(lr5, test))
      train_4$mod6[i] = cal_rmse(train$y, predict(lr6, train))
      test_4$mod6[i] = cal_rmse(test$y, predict(lr6, test))
      train_4$mod7[i] = cal_rmse(train$y, predict(lr7, train))
      test_4$mod7[i] = cal_rmse(test$y, predict(lr7, test))
      train_4$mod8[i] = cal_rmse(train$y, predict(lr8, train))
      test_4$mod8[i] = cal_rmse(test$y, predict(lr8, test))
      train_4$mod9[i] = cal_rmse(train$y, predict(lr9, train))
      test_4$mod9[i] = cal_rmse(test$y, predict(lr9, test))
    }
  }
}
```

Plotting the results

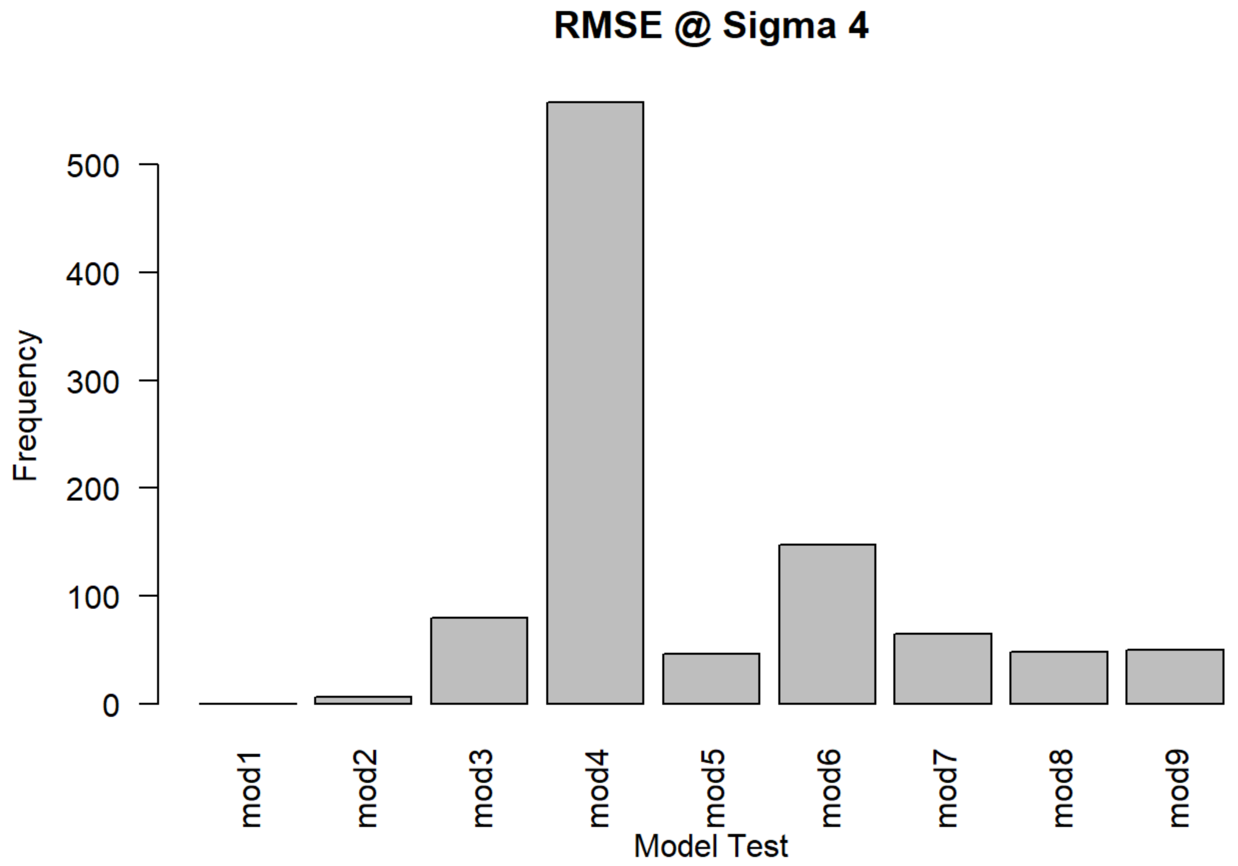
```
barplot(
        table(factor(colnames(test_1)[apply(test_1, 1, FUN = which.min)], levels = colnames(test_1))),
        main = "RMSE @ Sigma 1",
        xlab = "Model Test",
        ylab = "Frequency",
        las = 2
        )
```



```
barplot(
        table(factor(colnames(test_2)[apply(test_2, 1, FUN = which.min)], levels = colnames(test_2))),
        main = "RMSE @ Sigma 2",
        xlab = "Model Test",
        ylab = "Frequency",
        las = 2
        )
```



```
barplot(
        table(factor(colnames(test_4)[apply(test_4, 1, FUN = which.min)], levels = colnames(test_4))),
        main = "RMSE @ Sigma 4",
        xlab = "Model Test",
        ylab = "Frequency",
        las = 2
        )
```



Discussion

If Model 6 is our best model then using RMSE is one of the approach we can use to pick out the best model but it isn’t the perfect approach. As we can see from the plot, when sigma was 1, our method was able to select the best model. However, as sigma increases (2 and 4), we start seeing the frequency of model 6 being picked less and less. And thus, it’s becoming less reliable. Our method seems to gravitate towards model 4 as sigma increases.
In conclusion, level of noise affects the results greatly, as sigma increases, our model becomes less and less accurate.

Simulation Study 3: Power

Introduction

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

$H0:\beta1=0$ vs $H1:\beta1\neq0$

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

α=P[Reject H0∣H0 True]=P[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.

β=P[Fail to Reject H0∣H1 True]=P[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 β1 is non-zero.

$Power=1-\beta=P[Reject\ H0|H1\ True]$

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, $\beta1$ Noise Level, σ Significance Level, α We'll investigate the first three.

To do so we will simulate from the model

$Yi=\beta0+\beta1xi+\epsilon i$

where $\epsilon i \sim N(0,\sigma2)$.

For simplicity, we will let $\beta0=0$, thus $\beta1$ is essentially controlling the amount of "signal." We will then consider different signals, noises, and sample sizes:

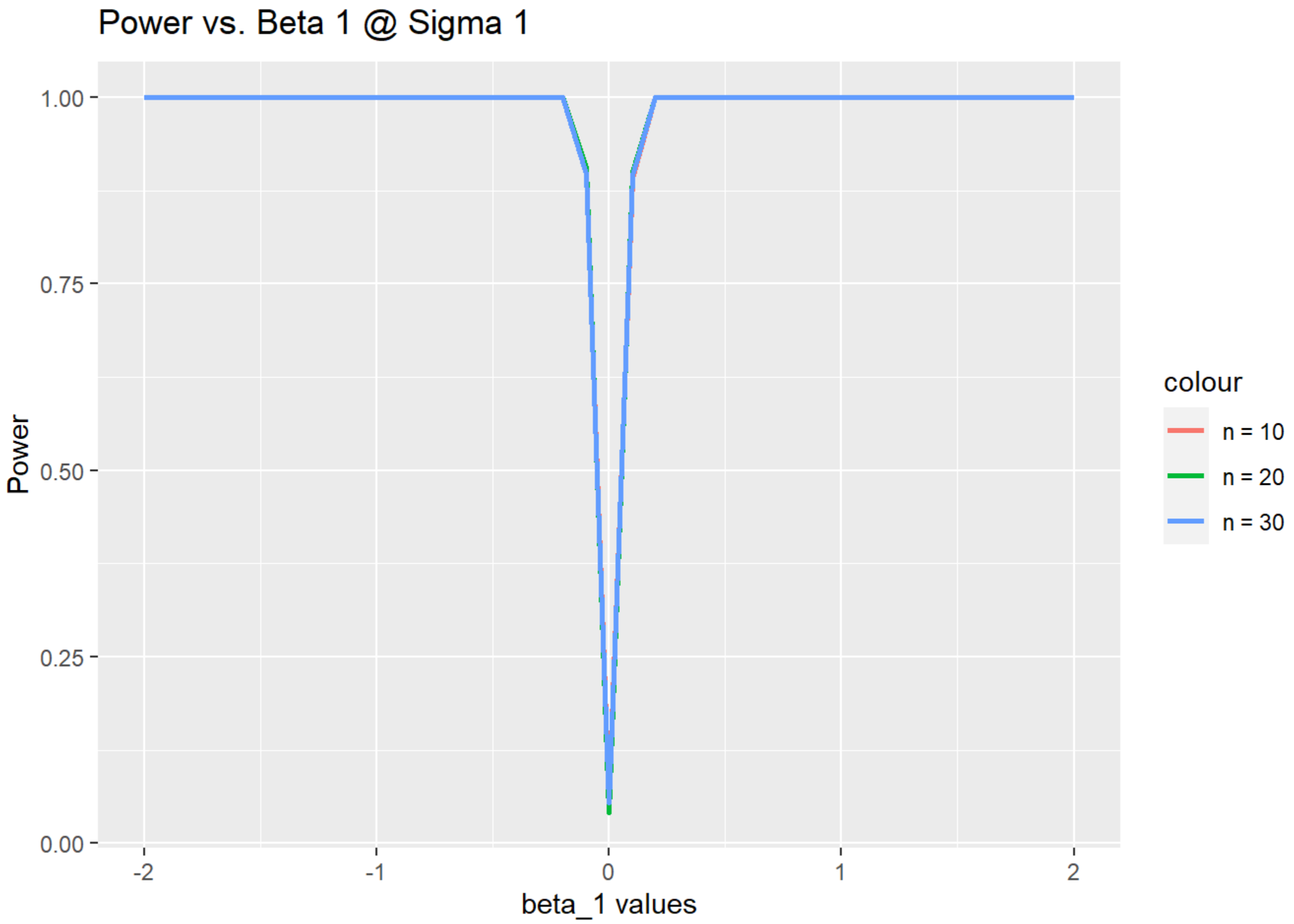
$\beta1\in(-2,-1.9,-1.8,\ldots,-0.1,0,0.1,0.2,0.3,\ldots1.9,2)$ $\sigma\in(1,2,4)$ $n\in(10,20,30)$ We will hold the significance level constant at $\alpha=0.05$.

```
birthday = 19990101
set.seed(birthday)
beta_0 = 0
beta_1s = seq(-2,2, by = .1)
sigmas = c(1,2,4)
samples = c(10,20,30)
alpha = .05
simulations = 1000
rows = length(sigmas) * length(samples)*length(beta_1s)
power_result = data.frame(sigma = rep(0,rows), s=rep(0,rows), beta_1 = rep(0,rows), power = rep(0,rows))

i = 1
for(sigma in sigmas){
  for(s in samples) {
    x_values = seq(0, 5, length = n)
    for(beta_1 in beta_1s) {
      reject_count = 0;
      for(sim in 1:simulations){
        eps = rnorm(n,0,sigma)
        y = beta_0 + beta_1 * x_values + eps
        lr = lm(y ~ x_values)
        p_val = summary(lr)$coefficients[2,4]
        if(p_val < alpha){
          reject_count = reject_count + 1
        }
      }
    power = reject_count / simulations
    power_result$sigma[i] = sigma
    power_result$s[i] = s
    power_result$beta_1[i] = beta_1
    power_result$power[i] = power
    i = i + 1
    }
  }
}
```

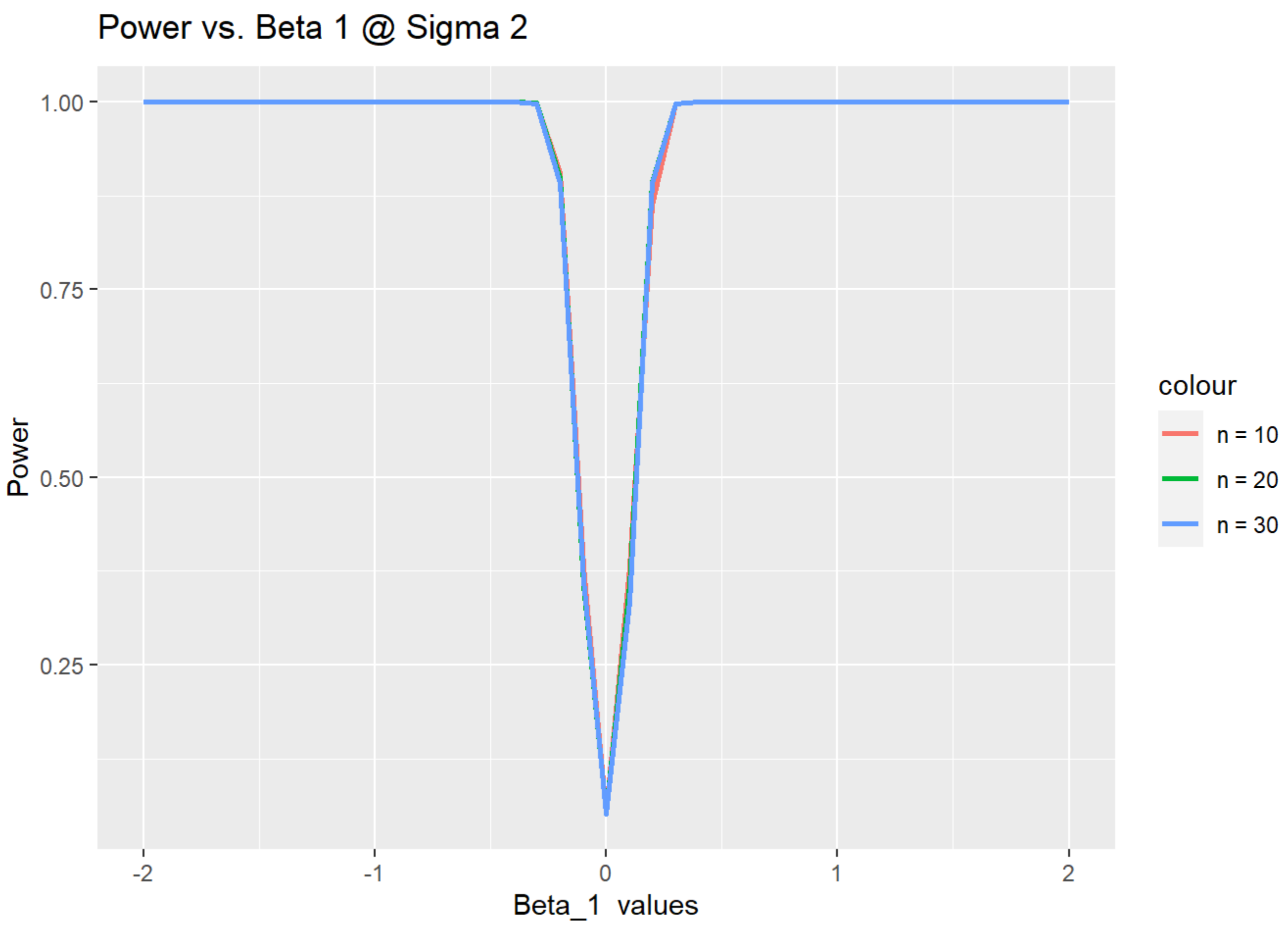
Results

Three plots

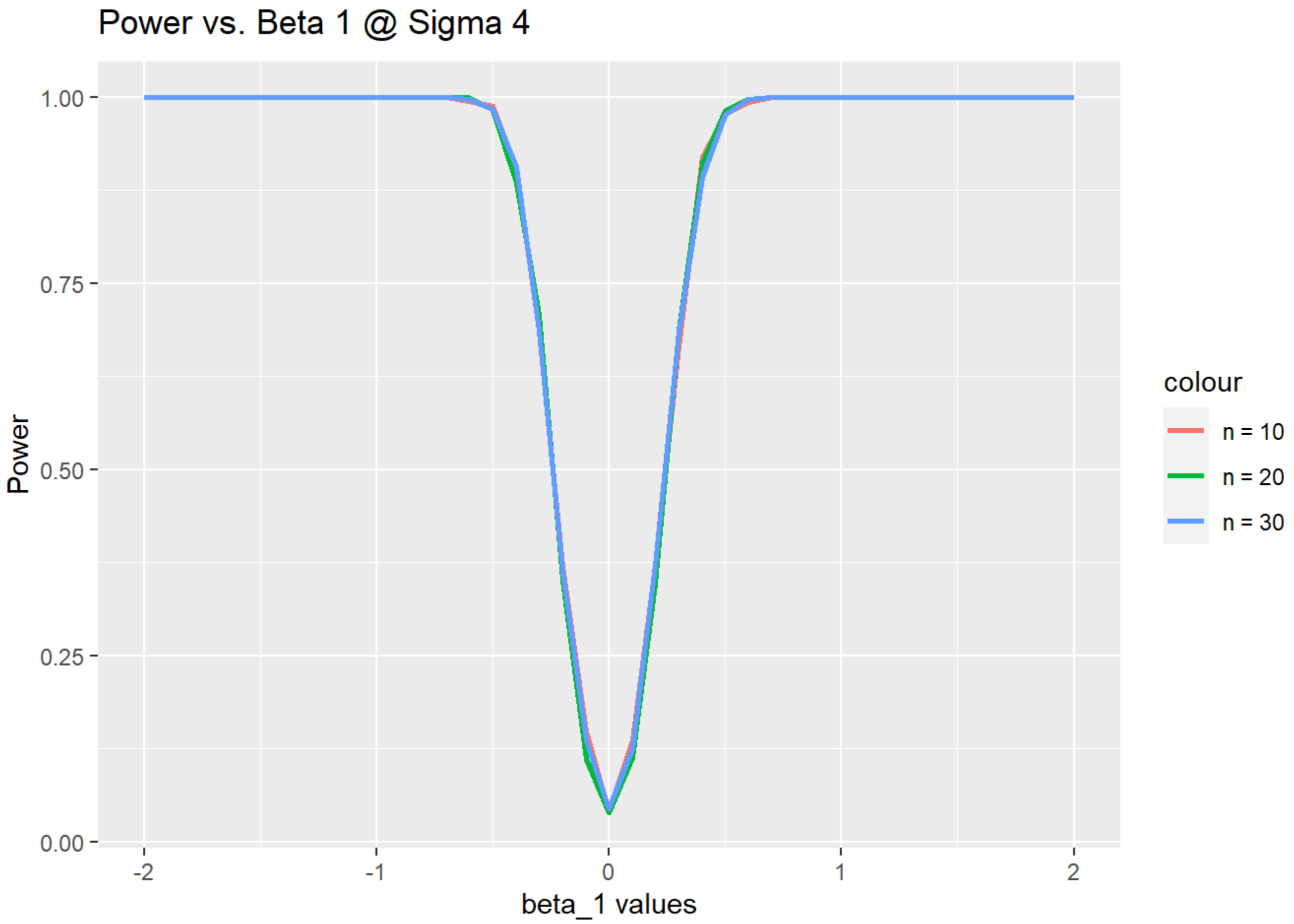
```
beta_1 = power_result[which((power_result$s == 10) & (power_result$sigma ==1)), c("beta_1")]
power1 = power_result[which((power_result$s == 10) & (power_result$sigma ==1)), c("power")]
power2 = power_result[which((power_result$s == 20) & (power_result$sigma ==1)), c("power")]
power3 = power_result[which((power_result$s == 30) & (power_result$sigma ==1)), c("power")]

power_graph = data.frame(beta_1 = beta_1, power_1 = power1, power_2 = power2, power_3 = power3)

ggplot(data = power_graph, aes(x = beta_1)) + geom_line(aes(x = beta_1, y = power_1, color = "n = 10"), siz
e=1) + geom_line(aes(x=beta_1, y = power_2, color = "n = 20"), size=1) + geom_line(aes(x=beta_1, y = power_
3, color = "n = 30"), size=1) + xlab(expression(paste("","beta_1"," values"))) + ylab("Power") + ggtitle(ex
pression(paste("Power vs. ", "Beta 1", " @ Sigma 1")))
```



```
power1 = power_result[which((power_result$s == 10) & (power_result$sigma ==2)), c("power")]
power2 = power_result[which((power_result$s == 20) & (power_result$sigma ==2)), c("power")]
power3 = power_result[which((power_result$s == 30) & (power_result$sigma ==2)), c("power")]

power_graph = data.frame(beta_1 = beta_1, power_1 = power1, power_2 = power2, power_3 = power3)

ggplot(data = power_graph, aes(x = beta_1)) + geom_line(aes(x = beta_1, y = power_1, color = "n = 10"), siz
e=1) + geom_line(aes(x=beta_1, y = power_2, color = "n = 20"), size=1) + geom_line(aes(x=beta_1, y = power_
3, color = "n = 30"), size=1) + xlab(expression(paste("","Beta_1","  values"))) + ylab("Power") + ggtitle(e
xpression(paste("Power vs. ", "Beta 1", " @ Sigma 2")))
```



```
power1 = power_result[which((power_result$s == 10) & (power_result$sigma ==4)), c("power")]
power2 = power_result[which((power_result$s == 20) & (power_result$sigma ==4)), c("power")]
power3 = power_result[which((power_result$s == 30) & (power_result$sigma ==4)), c("power")]

power_graph = data.frame(beta_1 = beta_1, power_1 = power1, power_2 = power2, power_3 = power3)

ggplot(data = power_graph, aes(x = beta_1)) + geom_line(aes(x = beta_1, y = power_1, color = "n = 10"), siz
e=1) + geom_line(aes(x=beta_1, y = power_2, color = "n = 20"), size=1) + geom_line(aes(x=beta_1, y = power_
3, color = "n = 30"), size=1) + xlab(expression(paste("","beta_1"," values"))) + ylab("Power") + ggtitle(ex
pression(paste("Power vs. ", "Beta 1", " @ Sigma 4")))
```



Discussion

How do n, β1, and σ affect power? Consider additional plots to demonstrate these effects.

From the look of the charts, it seems like n has the least effect on power. The power curve pretty much stayed the same in every level of N. B1 on the other hand has an impact on power. We can see that as values move away from b=0, power increases and plateaus as 1. Similarly, sigma also has an impact on power. We can see that as sigma increases, power curve grows wider and becomes less robust.

Are 1000 simulations sufficient?

Having a simulation of 1000 is most likely enough for what we are trying to accomplish. Think of tossing a coin: at 10 simulations, the power curve is wide, but as we increase the flips, our power curve becomes more narrow. Eventually, the number of flips (1000+) won’t have that much impact on the power curve anymore. So yes, I do think 1000 is enough.