stat420_simproj

Ian Vetter (ivette2)

2023-06-21

Simulation Study 1

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.

Use the data found in `study_1.csv` (study_1.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.

Done correctly, you will have simulated the `y` vector $2(models) \times 3(sigmas) \times 2000(sims) = 12000$ times.

Potential discussions:

- Do we know the true distribution of any of these values?
- How do the empirical distributions from the simulations compare to the true distributions? (You could consider adding a curve for the true distributions if you know them.)
- How are each of the F statistic, the p-value, and R^2 related to σ? Are any of those relationships the same for the significant and non-significant models?

An additional tip:

- Organize the plots in a grid for easy comparison. For example, a 1×3 row of F statistic plots as σ changes, then a 1×3 row of p-value plots as σ changes, followed by a similar row for the R^2 values. Consider a similar setup for the values attributed to the significant model and then again for the nonsignificant model.

Methods

```
birthday = 20010606
set.seed(birthday)
```

We can begin by defining the predictor coefficients of the two models:

The “significant” model

```
beta_0 = 3
beta_1 = 1
beta_2 = 1
beta_3 = 1
```

The “non-significant” model

```
beta_0_ = 3
beta_1_ = 0
beta_2_ = 0
beta_3_ = 0
```

Sample size and possible values of noise

```
n = 25
sigma = c(1, 5, 10)
```

Now, we’ll load in the predictor values from the provided csv file:

```
X = read.csv("study_1.csv")
summary(X)
```

```
##        y           x1                x2             x3
##  Min.   :0   Min.   :-4.850   Min.   :2.10   Min.   :2.064
##  1st Qu.:0   1st Qu.:-3.530   1st Qu.:2.30   1st Qu.:2.280
##  Median :0   Median :-2.280   Median :2.60   Median :2.551
##  Mean   :0   Mean   :-1.926   Mean   :2.56   Mean   :2.537
##  3rd Qu.:0   3rd Qu.: 0.090   3rd Qu.:2.80   3rd Qu.:2.774
##  Max.   :0   Max.   : 0.960   Max.   :3.00   Max.   :3.000
```

Now for our simulation. We'll first create some vectors to store the F-statistics, p-values and R^2 values for the significant and non-significant models

```
F_sig_1 = rep(0, 2000)
F_sig_2 = rep(0, 2000)
F_sig_3 = rep(0, 2000)

F_nonsig_1 = rep(0, 2000)
F_nonsig_2 = rep(0, 2000)
F_nonsig_3 = rep(0, 2000)

p_sig_1 = rep(0, 2000)
p_sig_2 = rep(0, 2000)
p_sig_3 = rep(0, 2000)

p_nonsig_1 = rep(0, 2000)
p_nonsig_2 = rep(0, 2000)
p_nonsig_3 = rep(0, 2000)

R2_sig_1 = rep(0, 2000)
R2_sig_2 = rep(0, 2000)
R2_sig_3 = rep(0, 2000)

R2_nonsig_1 = rep(0, 2000)
R2_nonsig_2 = rep(0, 2000)
R2_nonsig_3 = rep(0, 2000)

for (s in c(1, 5, 10)) {

  for (i in 1:2000) {

    eps = rnorm(n, mean = 0 , sd = s)

    X$y = beta_0 + beta_1 * X$x1 + beta_2 * X$x2 + beta_3 * X$x3 + eps
    sig_fit = lm(y ~ x1 + x2 + x3, data=X)

    X$y = beta_0_ + beta_1_ * X$x1 + beta_2_ * X$x2 + beta_3_ * X$x3 + eps
    nonsig_fit = lm(y ~ x1 + x2 + x3, data=X)

    ss = summary(sig_fit)
    nss = summary(nonsig_fit)

    if (s == 1) {

      F_sig_1[i] = ss$fstatistic[1]
      F_nonsig_1[i] = nss$fstatistic[1]

      p_sig_1[i] = pf(ss$fstatistic[1],ss$fstatistic[2],ss$fstatistic[3],lower.tail=FALSE)
      p_nonsig_1[i] = pf(nss$fstatistic[1],nss$fstatistic[2],nss$fstatistic[3],lower.tail=FALSE)

      R2_sig_1[i] = ss$r.squared
      R2_nonsig_1[i] = nss$r.squared

    } else if (s == 5) {
```

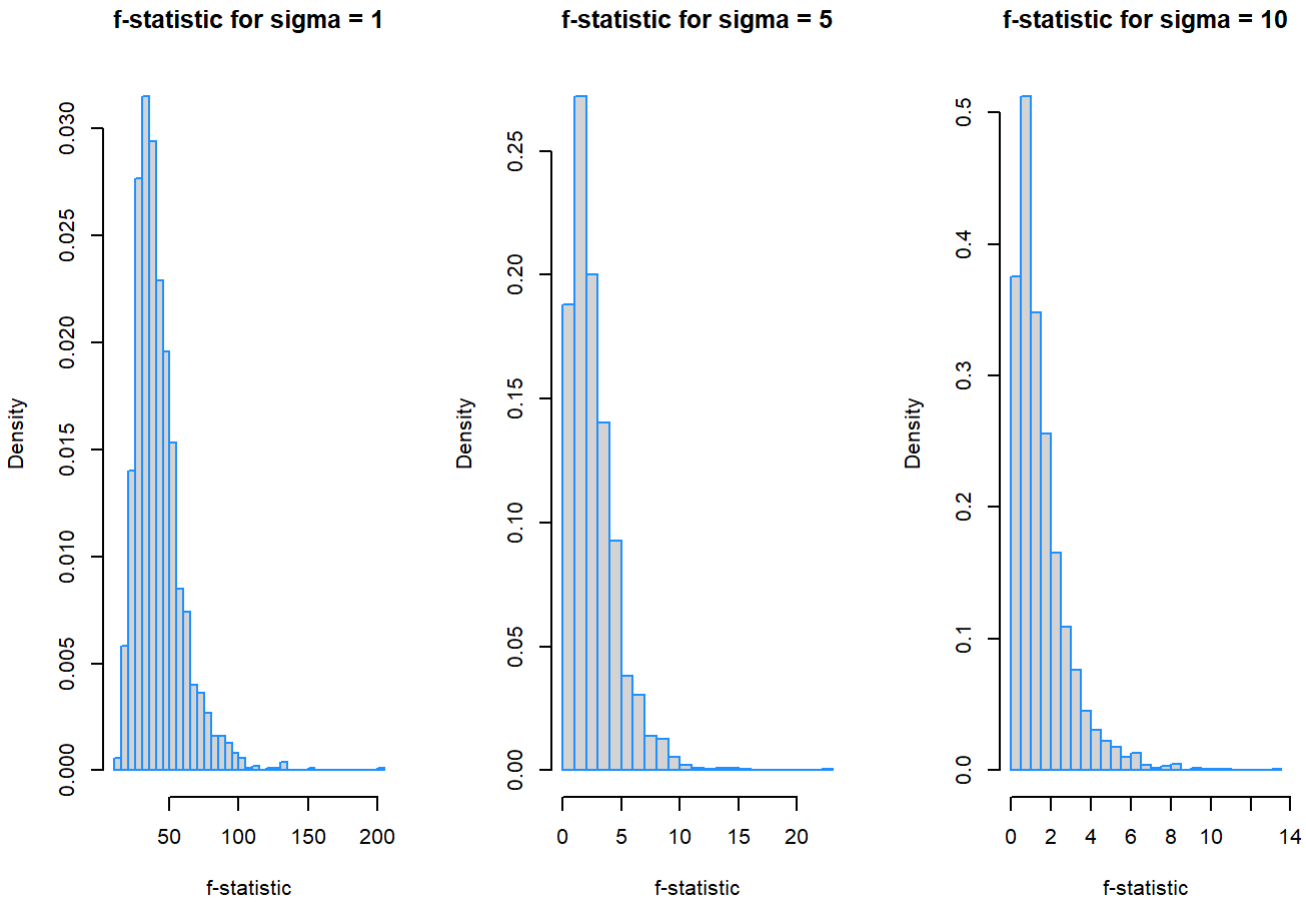
```
      F_sig_2[i] = ss$fstatistic[1]
      F_nonsig_2[i] = nss$fstatistic[1]

      p_sig_2[i] = pf(ss$fstatistic[1],ss$fstatistic[2],ss$fstatistic[3],lower.tail=FALSE)
      p_nonsig_2[i] = pf(nss$fstatistic[1],nss$fstatistic[2],nss$fstatistic[3],lower.tail=FALSE)

      R2_sig_2[i] = ss$r.squared
      R2_nonsig_2[i] = nss$r.squared

    } else if (s == 10) {

      F_sig_3[i] = ss$fstatistic[1]
      F_nonsig_3[i] = nss$fstatistic[1]

      p_sig_3[i] = pf(ss$fstatistic[1],ss$fstatistic[2],ss$fstatistic[3],lower.tail=FALSE)
      p_nonsig_3[i] = pf(nss$fstatistic[1],nss$fstatistic[2],nss$fstatistic[3],lower.tail=FALSE)

      R2_sig_3[i] = ss$r.squared
      R2_nonsig_3[i] = nss$r.squared
    }
  }
}
```

Results

Now let's visualize the results of the simulation.

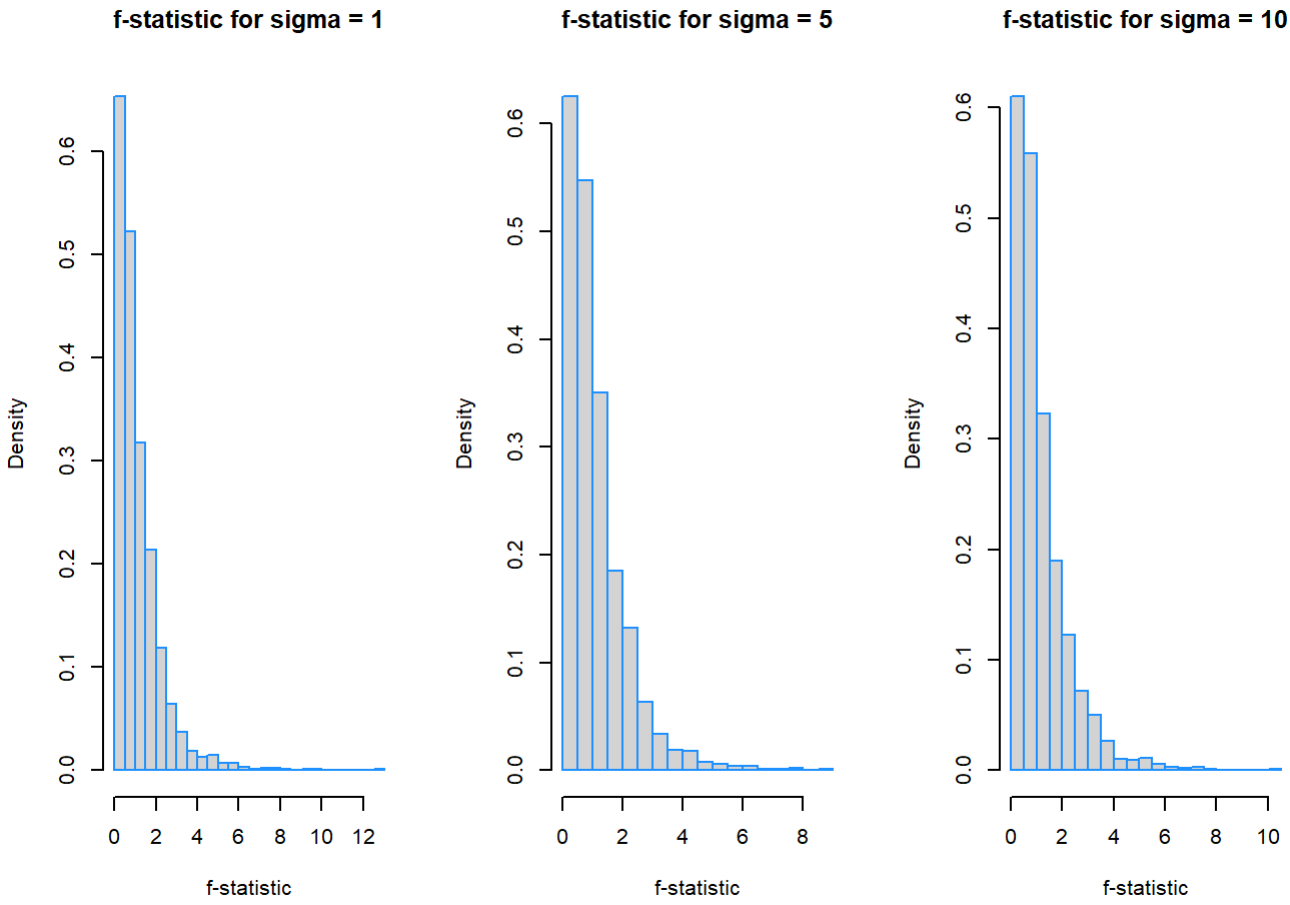
The F-statistics for the significant model:

```
par(mfrow=c(1,3))
hist(F_sig_1, prob = TRUE, breaks = 30,
     xlab = "f-statistic", main = "f-statistic for sigma = 1", border = "dodgerblue")
hist(F_sig_2, prob = TRUE, breaks = 30,
     xlab = "f-statistic", main = "f-statistic for sigma = 5", border = "dodgerblue")
hist(F_sig_3, prob = TRUE, breaks = 30,
     xlab = "f-statistic", main = "f-statistic for sigma = 10", border = "dodgerblue")
```



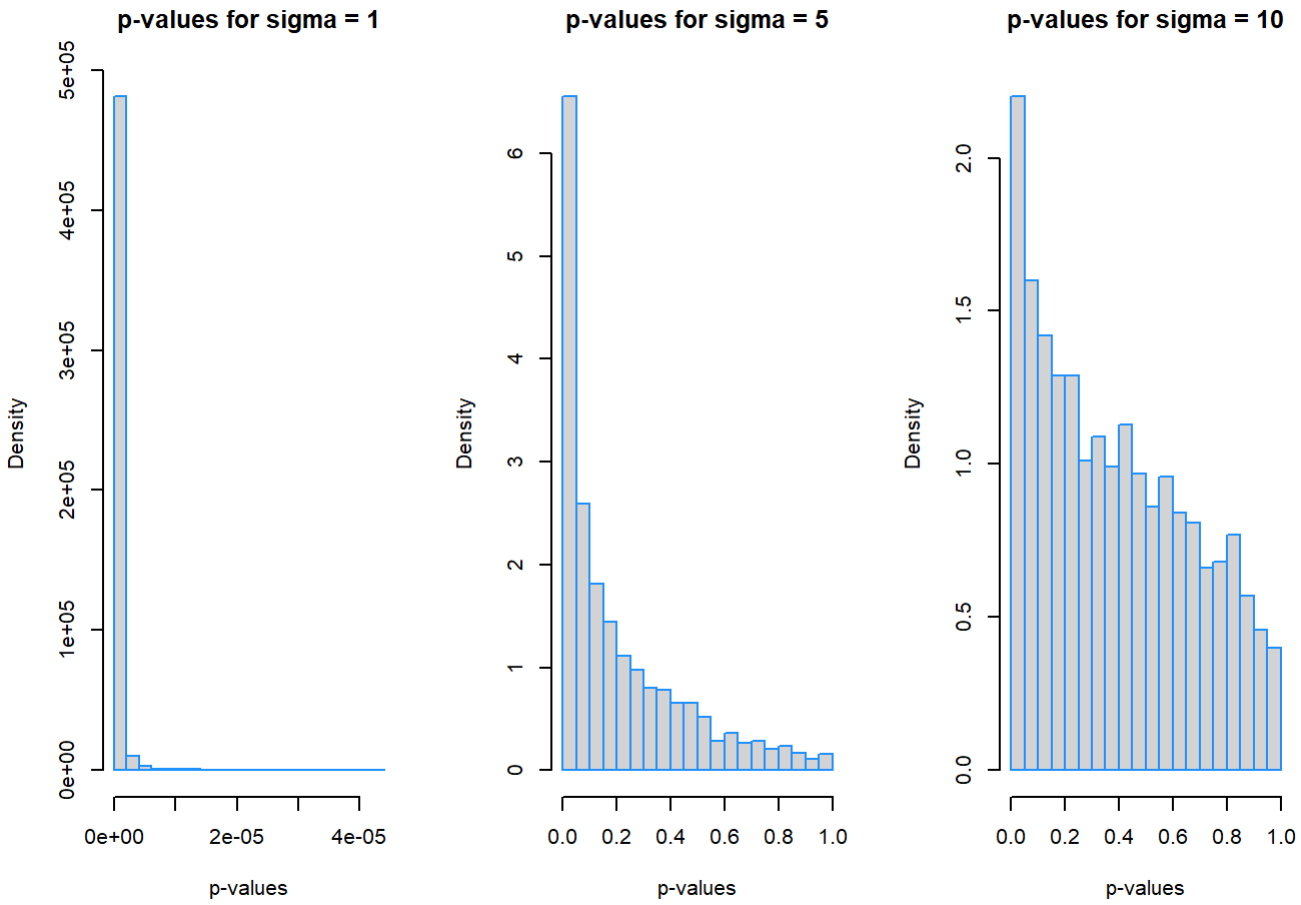
F-statistics for the non-significant model:

```
par(mfrow=c(1,3))
hist(F_nonsig_1, prob = TRUE, breaks = 30,
     xlab = "f-statistic", main = "f-statistic for sigma = 1", border = "dodgerblue")
hist(F_nonsig_2, prob = TRUE, breaks = 30,
     xlab = "f-statistic", main = "f-statistic for sigma = 5", border = "dodgerblue")
hist(F_nonsig_3, prob = TRUE, breaks = 30,
     xlab = "f-statistic", main = "f-statistic for sigma = 10", border = "dodgerblue")
```



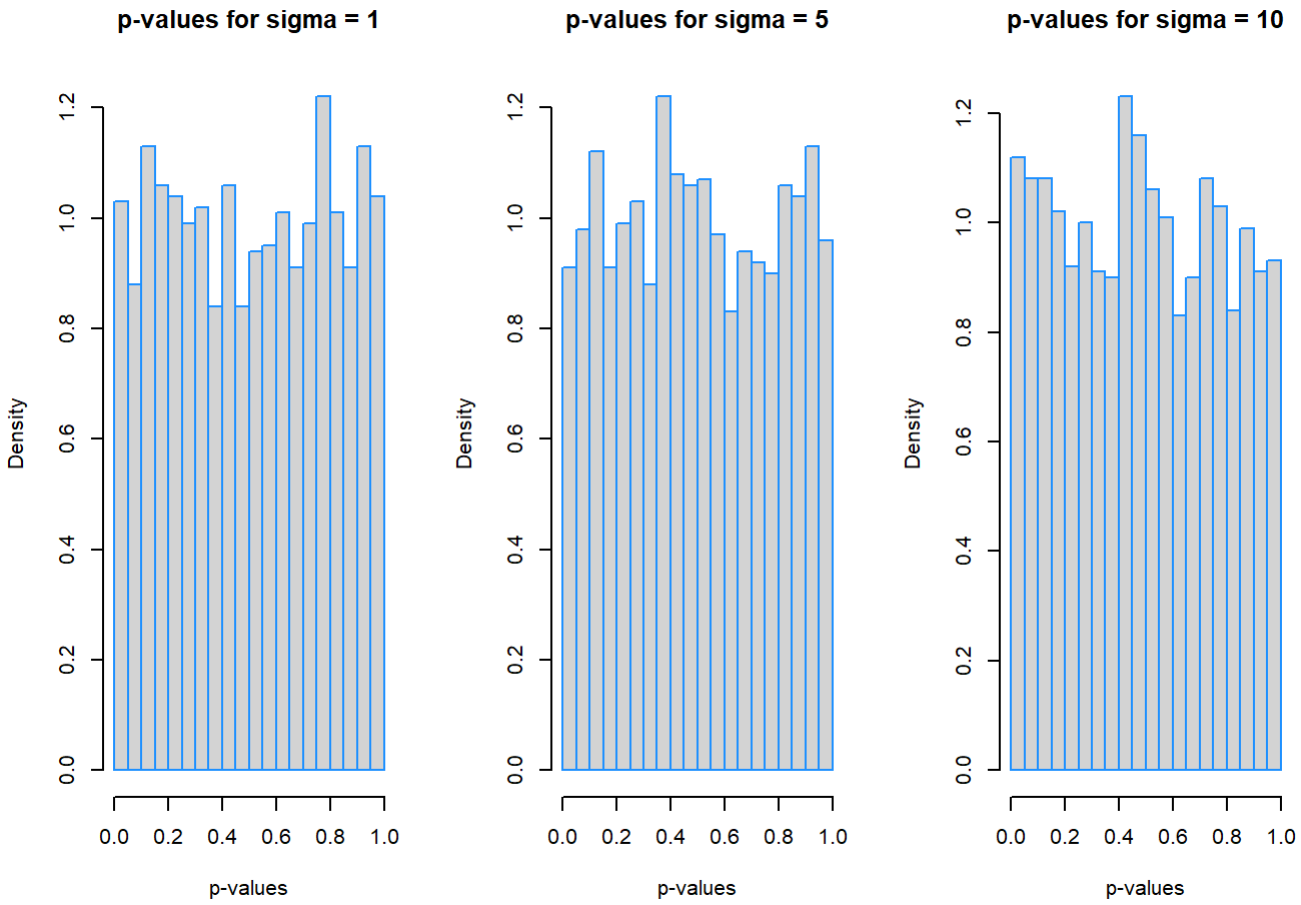
p-values for the significant model:

```
par(mfrow=c(1,3))
hist(p_sig_1, prob = TRUE, breaks = 30,
     xlab = "p-values", main = "p-values for sigma = 1", border = "dodgerblue")
hist(p_sig_2, prob = TRUE, breaks = 30,
     xlab = "p-values", main = "p-values for sigma = 5", border = "dodgerblue")
hist(p_sig_3, prob = TRUE, breaks = 30,
     xlab = "p-values", main = "p-values for sigma = 10", border = "dodgerblue")
```



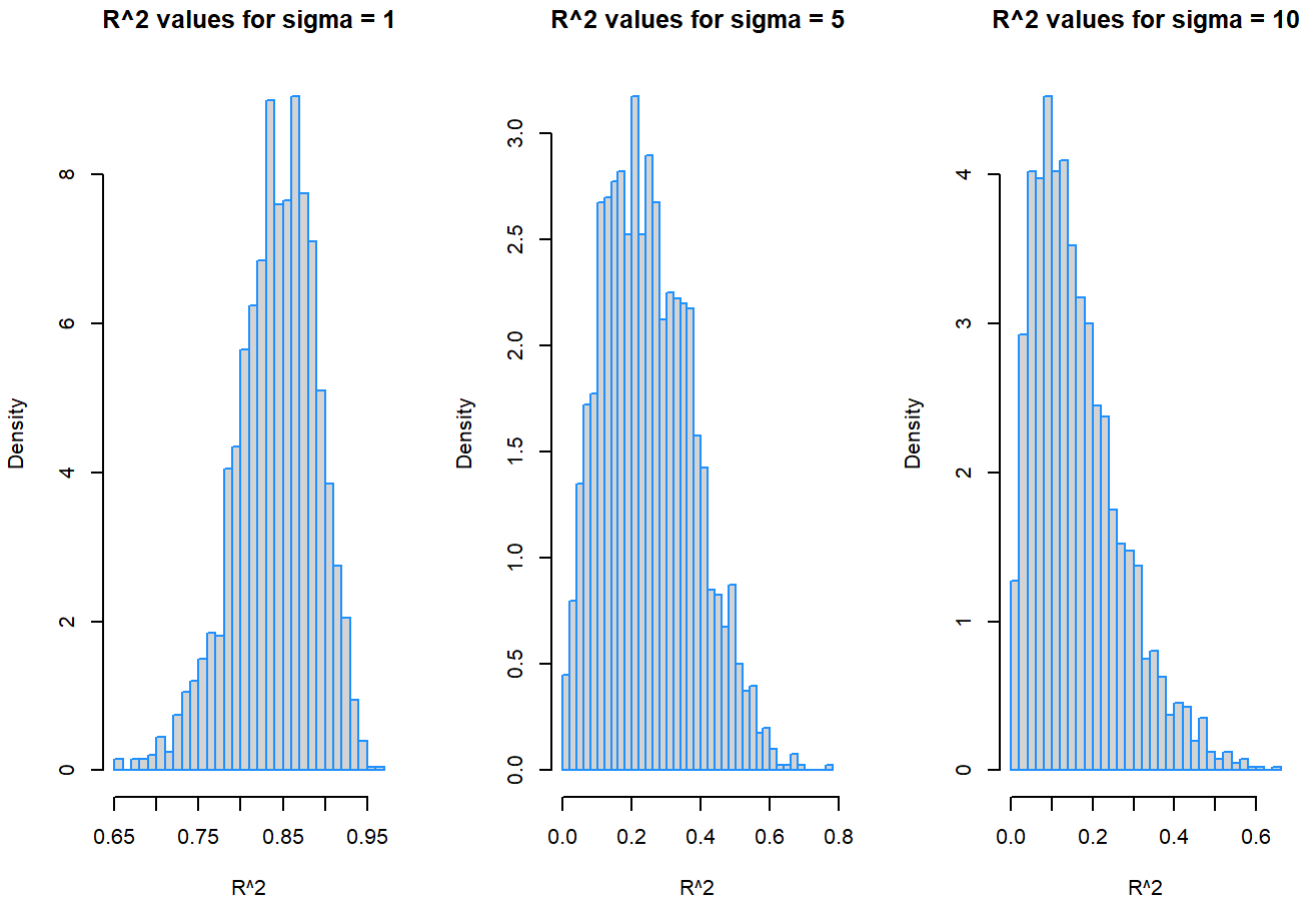
p-values for the non-significant model:

```
par(mfrow=c(1,3))
hist(p_nonsig_1, prob = TRUE, breaks = 30,
     xlab = "p-values", main = "p-values for sigma = 1", border = "dodgerblue")
hist(p_nonsig_2, prob = TRUE, breaks = 30,
     xlab = "p-values", main = "p-values for sigma = 5", border = "dodgerblue")
hist(p_nonsig_3, prob = TRUE, breaks = 30,
     xlab = "p-values", main = "p-values for sigma = 10", border = "dodgerblue")
```



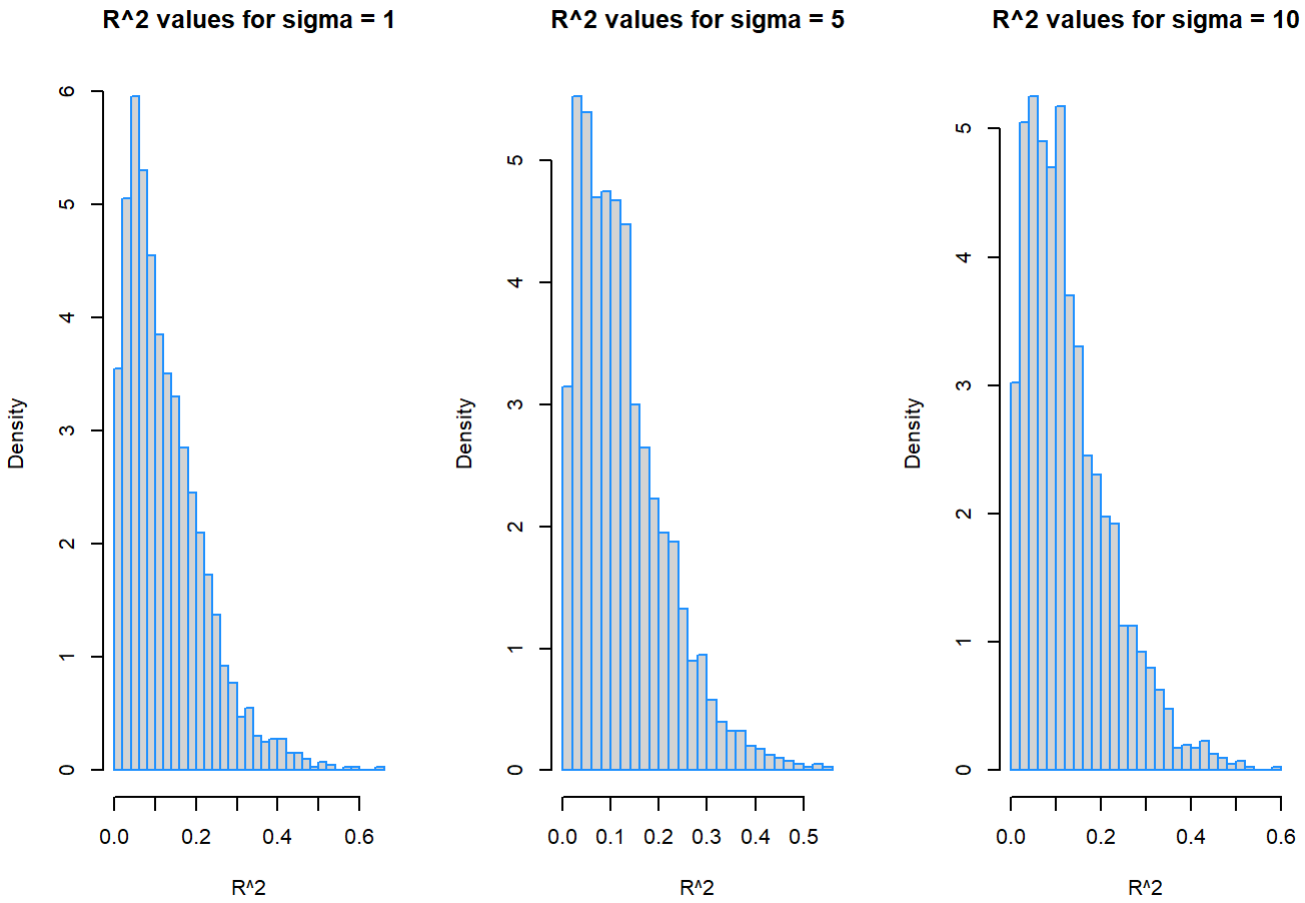
R2 values for the significant model:

```
par(mfrow=c(1,3))
hist(R2_sig_1, prob = TRUE, breaks = 30,
     xlab = "R^2", main = "R^2 values for sigma = 1", border = "dodgerblue")
hist(R2_sig_2, prob = TRUE, breaks = 30,
     xlab = "R^2", main = "R^2 values for sigma = 5", border = "dodgerblue")
hist(R2_sig_3, prob = TRUE, breaks = 30,
     xlab = "R^2", main = "R^2 values for sigma = 10", border = "dodgerblue")
```



R2 values for the non-significant model:

```
par(mfrow=c(1,3))
hist(R2_nonsig_1, prob = TRUE, breaks = 30,
     xlab = "R^2", main = "R^2 values for sigma = 1", border = "dodgerblue")
hist(R2_nonsig_2, prob = TRUE, breaks = 30,
     xlab = "R^2", main = "R^2 values for sigma = 5", border = "dodgerblue")
hist(R2_nonsig_3, prob = TRUE, breaks = 30,
     xlab = "R^2", main = "R^2 values for sigma = 10", border = "dodgerblue")
```



(Discussion)

Do we know the true distribution of any of these values?

We do not know the true distribution for any of the given values, but we can make inferences upon how they relate dependent on each other to our Y. One of the main assumptions we make in linear regression is that we know how the data is distributed, but we inherently can't know the true distribution so we assume that it follows some distribution. We are able to know the true distribution of the null model as this assumes each of our predictors does not influence Y which allows for us to know the distribution of the F statistic, P-value, and R^2.

How are R^2 and σ related?

R^2 and σ have an inverse relationship here. The R^2 value measures the proportion of the variation in the dependent variable (y) that can be explained by the model, and σ ultimately defines the error of the model. As we can confirm from the histograms, greater values of σ lead to greater model variability, which in turn makes the model less significant and decreases its expected R^2 value.

Simulation Study 2

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.

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

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

```
birthday = 20010606
set.seed(birthday)
```

Six parameter model for simulation:

```
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
```

Load in the data:

```
X = read.csv("study_2.csv")
summary(X)
```

```
##        y           x1                 x2                x3          
##  Min.   :0   Min.   :-4.9589   Min.   :-1.0000   Min.   :0.0001  
##  1st Qu.:0   1st Qu.:-3.4924   1st Qu.:-0.3100   1st Qu.:0.1369  
##  Median :0   Median :-2.0590   Median : 0.4350   Median :0.5112  
##  Mean   :0   Mean   :-2.0171   Mean   : 0.4901   Mean   :1.0042  
##  3rd Qu.:0   3rd Qu.:-0.5494   3rd Qu.: 1.2300   3rd Qu.:1.5129  
##  Max.   :0   Max.   : 0.9980   Max.   : 2.0000   Max.   :4.0000  
##        x4                  x5                  x6               x7          
##  Min.   :0.0009879   Min.   :0.0006568   Min.   :-3.200   Min.   :0.002189  
##  1st Qu.:0.2468956   1st Qu.:0.2688126   1st Qu.:-0.700   1st Qu.:0.222560  
##  Median :0.5071822   Median :0.5280717   Median : 0.000   Median :0.462155  
##  Mean   :0.5055345   Mean   :0.5167349   Mean   :-0.026   Mean   :0.472884  
##  3rd Qu.:0.7750806   3rd Qu.:0.7739698   3rd Qu.: 0.600   3rd Qu.:0.727191  
##  Max.   :0.9993016   Max.   :0.9999392   Max.   : 3.500   Max.   :0.999086  
##        x8                  x9          
##  Min.   :-2.6056337   Min.   :-2.87997  
##  1st Qu.:-0.6877945   1st Qu.:-0.63199  
##  Median : 0.0294629   Median : 0.02893  
##  Mean   : 0.0005055   Mean   : 0.02550  
##  3rd Qu.: 0.7162151   3rd Qu.: 0.75458  
##  Max.   : 2.9979299   Max.   : 2.99684
```

Define an RMSE function to use for the simulation:

```
RMSE = function(y, yhat) {
  return(sqrt((1/length(y)) * sum((y-yhat)^2)))
}
```

We will store train and test RMSE values for each of the three σ values: 1, 2 and 4

```
sig1_tr = matrix(0, 9, 1000)
sig1_te = matrix(0, 9, 1000)

sig2_tr = matrix(0, 9, 1000)
sig2_te = matrix(0, 9, 1000)

sig3_tr = matrix(0, 9, 1000)
sig3_te = matrix(0, 9, 1000)

n = 500

for (s in c(1, 2, 4)) {

  for (i in 1:1000) {

    eps = rnorm(n, mean = 0, sd = s)
    X$y = beta_0 + X$x1 * beta_1 + X$x2 * beta_2 + X$x3 * beta_3 + X$x4 * beta_4 + X$x5 * beta_
5 + X$x6 * beta_6 + eps

    idx = sample(1:nrow(X),250)
    train_data = X[idx,]
    test_data = X[-idx,]


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


    if (s == 1) {

      sig1_tr[1, i] = RMSE(train_data$y, predict(mod1, train_data))
      sig1_te[1, i] = RMSE(test_data$y, predict(mod1, test_data))

      sig1_tr[2, i] = RMSE(train_data$y, predict(mod2, train_data))
      sig1_te[2, i] = RMSE(test_data$y, predict(mod2, test_data))

      sig1_tr[3, i] = RMSE(train_data$y, predict(mod3, train_data))
      sig1_te[3, i] = RMSE(test_data$y, predict(mod3, test_data))

      sig1_tr[4, i] = RMSE(train_data$y, predict(mod4, train_data))
      sig1_te[4, i] = RMSE(test_data$y, predict(mod4, test_data))

      sig1_tr[5, i] = RMSE(train_data$y, predict(mod5, train_data))
      sig1_te[5, i] = RMSE(test_data$y, predict(mod5, test_data))
```

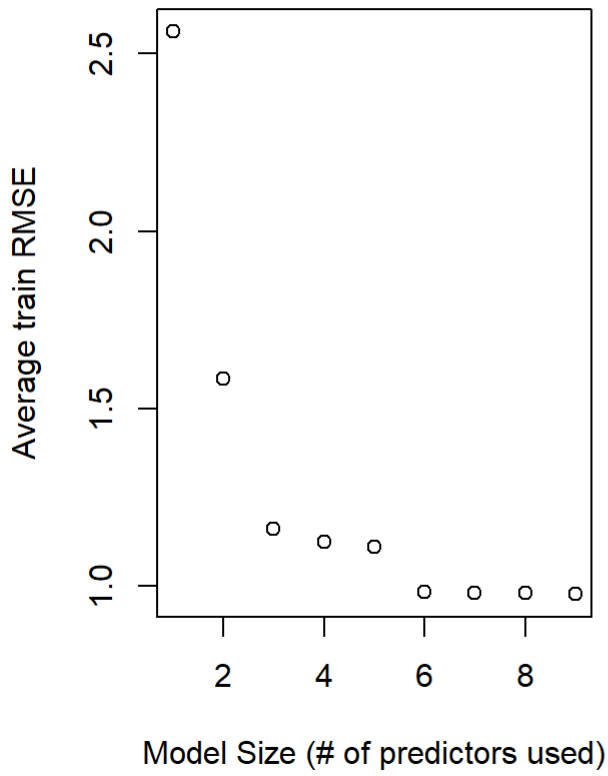
```
    sig1_tr[6, i] = RMSE(train_data$y, predict(mod6, train_data))
    sig1_te[6, i] = RMSE(test_data$y, predict(mod6, test_data))

    sig1_tr[7, i] = RMSE(train_data$y, predict(mod7, train_data))
    sig1_te[7, i] = RMSE(test_data$y, predict(mod7, test_data))

    sig1_tr[8, i] = RMSE(train_data$y, predict(mod8, train_data))
    sig1_te[8, i] = RMSE(test_data$y, predict(mod8, test_data))

    sig1_tr[9, i] = RMSE(train_data$y, predict(mod9, train_data))
    sig1_te[9, i] = RMSE(test_data$y, predict(mod9, test_data))

  } else if (s == 2) {

    sig2_tr[1, i] = RMSE(train_data$y, predict(mod1, train_data))
    sig2_te[1, i] = RMSE(test_data$y, predict(mod1, test_data))

    sig2_tr[2, i] = RMSE(train_data$y, predict(mod2, train_data))
    sig2_te[2, i] = RMSE(test_data$y, predict(mod2, test_data))

    sig2_tr[3, i] = RMSE(train_data$y, predict(mod3, train_data))
    sig2_te[3, i] = RMSE(test_data$y, predict(mod3, test_data))

    sig2_tr[4, i] = RMSE(train_data$y, predict(mod4, train_data))
    sig2_te[4, i] = RMSE(test_data$y, predict(mod4, test_data))

    sig2_tr[5, i] = RMSE(train_data$y, predict(mod5, train_data))
    sig2_te[5, i] = RMSE(test_data$y, predict(mod5, test_data))

    sig2_tr[6, i] = RMSE(train_data$y, predict(mod6, train_data))
    sig2_te[6, i] = RMSE(test_data$y, predict(mod6, test_data))

    sig2_tr[7, i] = RMSE(train_data$y, predict(mod7, train_data))
    sig2_te[7, i] = RMSE(test_data$y, predict(mod7, test_data))

    sig2_tr[8, i] = RMSE(train_data$y, predict(mod8, train_data))
    sig2_te[8, i] = RMSE(test_data$y, predict(mod8, test_data))

    sig2_tr[9, i] = RMSE(train_data$y, predict(mod9, train_data))
    sig2_te[9, i] = RMSE(test_data$y, predict(mod9, test_data))

  } else {

    sig3_tr[1, i] = RMSE(train_data$y, predict(mod1, train_data))
    sig3_te[1, i] = RMSE(test_data$y, predict(mod1, test_data))

    sig3_tr[2, i] = RMSE(train_data$y, predict(mod2, train_data))
    sig3_te[2, i] = RMSE(test_data$y, predict(mod2, test_data))

    sig3_tr[3, i] = RMSE(train_data$y, predict(mod3, train_data))
    sig3_te[3, i] = RMSE(test_data$y, predict(mod3, test_data))
```

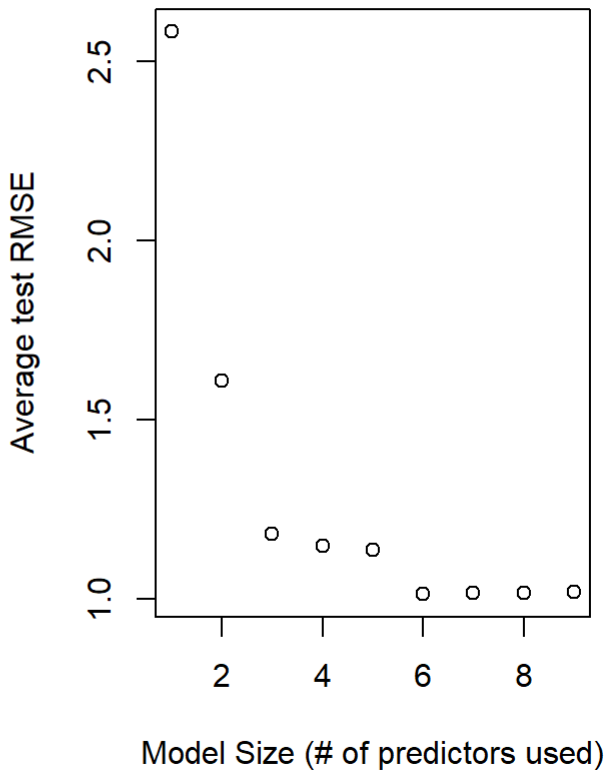
```
      sig3_tr[4, i] = RMSE(train_data$y, predict(mod4, train_data))
      sig3_te[4, i] = RMSE(test_data$y, predict(mod4, test_data))

      sig3_tr[5, i] = RMSE(train_data$y, predict(mod5, train_data))
      sig3_te[5, i] = RMSE(test_data$y, predict(mod5, test_data))

      sig3_tr[6, i] = RMSE(train_data$y, predict(mod6, train_data))
      sig3_te[6, i] = RMSE(test_data$y, predict(mod6, test_data))

      sig3_tr[7, i] = RMSE(train_data$y, predict(mod7, train_data))
      sig3_te[7, i] = RMSE(test_data$y, predict(mod7, test_data))

      sig3_tr[8, i] = RMSE(train_data$y, predict(mod8, train_data))
      sig3_te[8, i] = RMSE(test_data$y, predict(mod8, test_data))

      sig3_tr[9, i] = RMSE(train_data$y, predict(mod9, train_data))
      sig3_te[9, i] = RMSE(test_data$y, predict(mod9, test_data))
    }


  }
}
```

Results

Using our simulation data, we can begin by plotting the average train and test RMSE vs. model size for $\sigma = 1$:

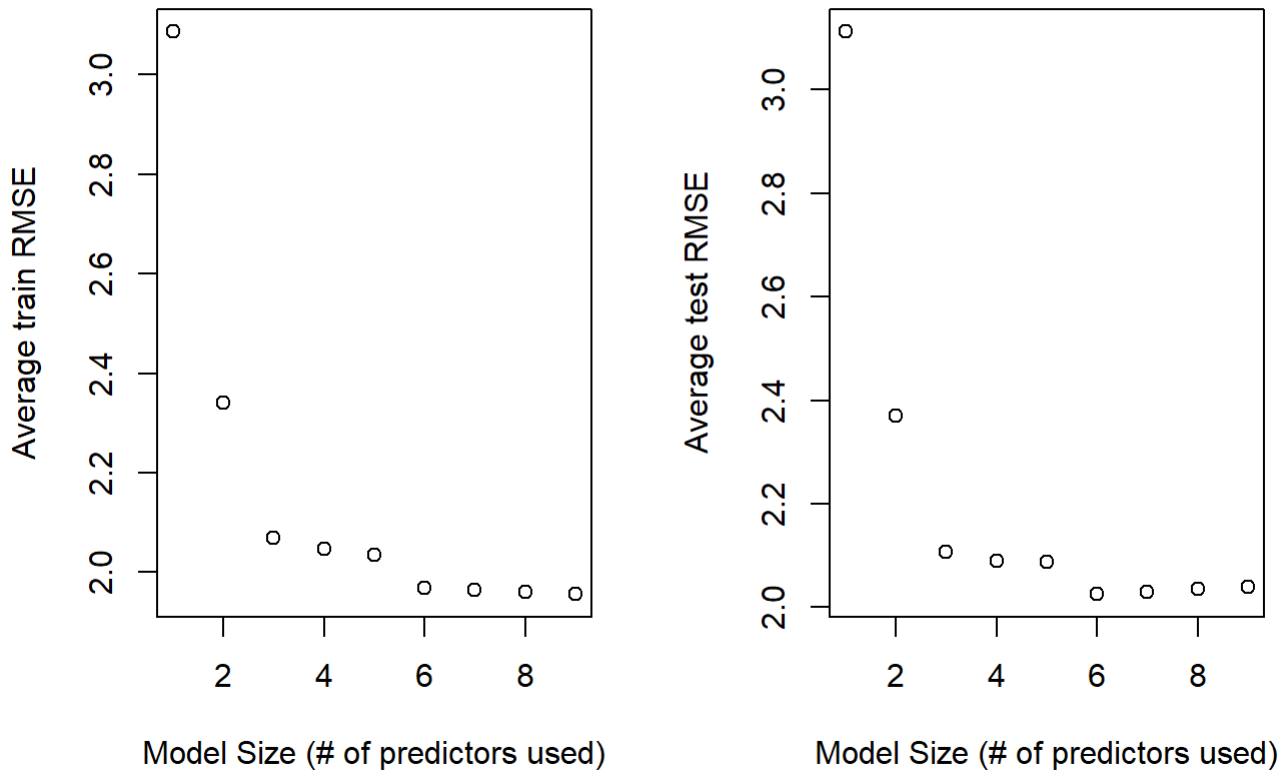
```
sig1_tr_means = apply(sig1_tr, 1, mean)
sig1_te_means = apply(sig1_te, 1, mean)

par(mfrow=c(1,2))

plot(1:9, sig1_tr_means, xlab="Model Size (# of predictors used)", ylab = "Average train RMSE")
plot(1:9, sig1_te_means, xlab="Model Size (# of predictors used)", ylab = "Average test RMSE")
```





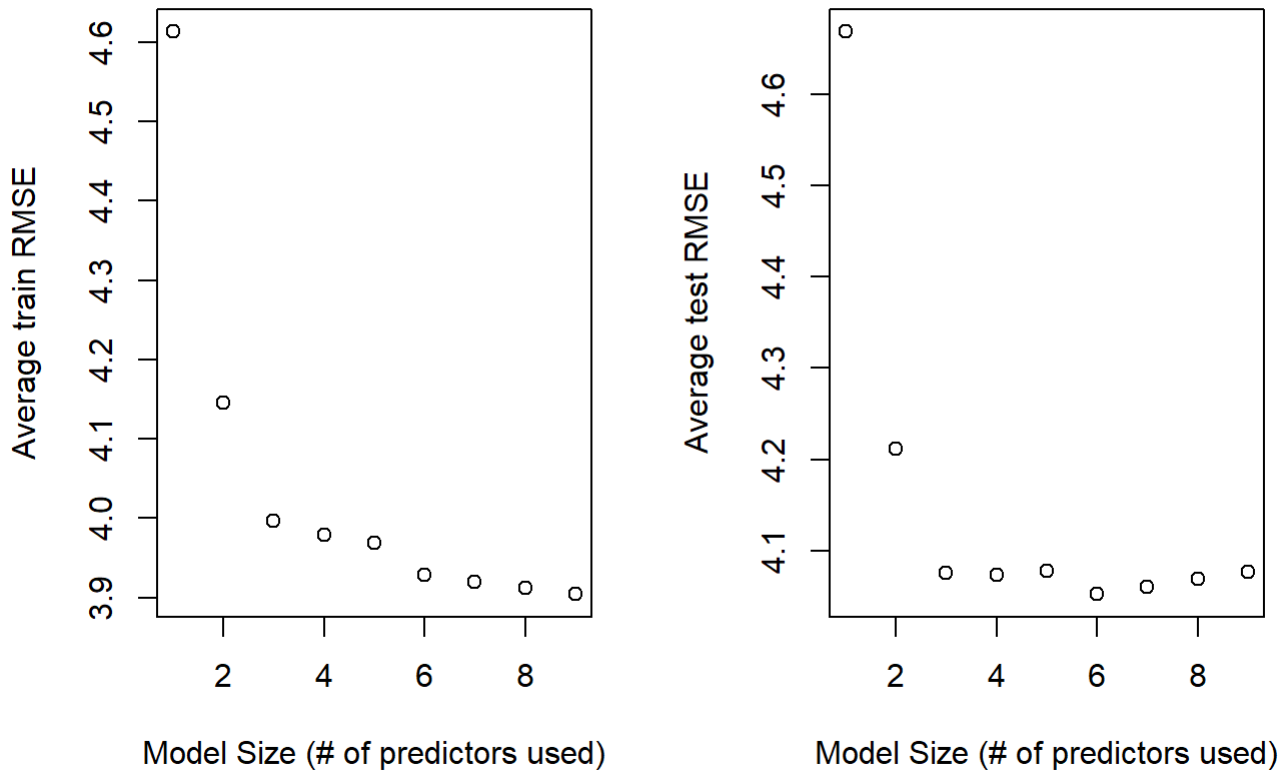
Now for $\sigma = 2$:

```
sig2_tr_means = apply(sig2_tr, 1, mean)
sig2_te_means = apply(sig2_te, 1, mean)

par(mfrow=c(1,2))

plot(1:9, sig2_tr_means, xlab="Model Size (# of predictors used)", ylab = "Average train RMSE")
plot(1:9, sig2_te_means, xlab="Model Size (# of predictors used)", ylab = "Average test RMSE")
```



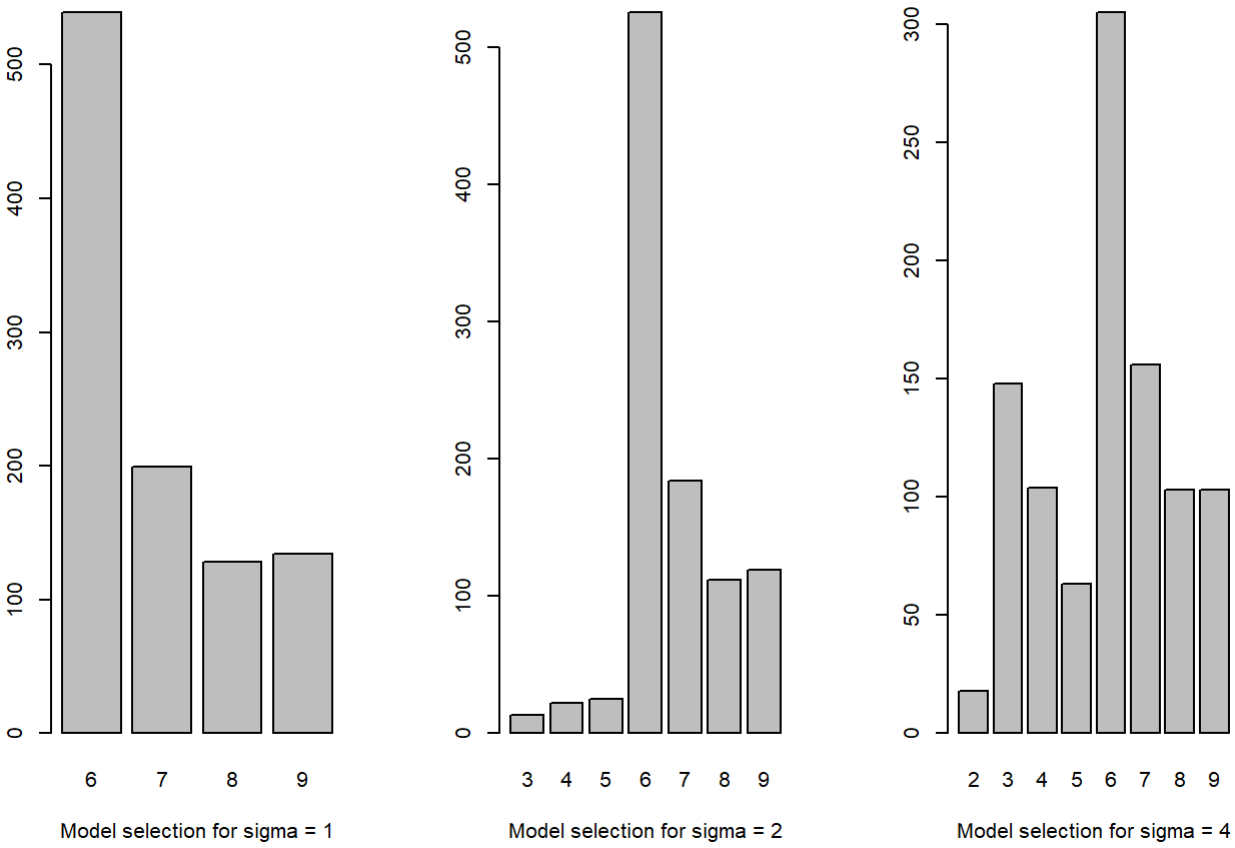
And now $\sigma = 3$:

```
sig3_tr_means = apply(sig3_tr, 1, mean)
sig3_te_means = apply(sig3_te, 1, mean)

par(mfrow=c(1,2))

plot(1:9, sig3_tr_means, xlab="Model Size (# of predictors used)", ylab = "Average train RMSE")
plot(1:9, sig3_te_means, xlab="Model Size (# of predictors used)", ylab = "Average test RMSE")
```



Now let's plot the RMSE selected model for each simulation iteration:

```
selection1 = apply(sig1_te, 2, which.min)
selection2 = apply(sig2_te, 2, which.min)
selection3 = apply(sig3_te, 2, which.min)

par(mfrow=c(1,3))
barplot(table(selection1), xlab = "Model selection for sigma = 1")
barplot(table(selection2), xlab = "Model selection for sigma = 2")
barplot(table(selection3), xlab = "Model selection for sigma = 4")
```



Discussion

Does RMSE Always select the correct model? On average, does it select the correct model?

As we can see from our simulations, RMSE will not always select the correct model. With any amount of noise in the model, RMSE will select a variety of model options with sufficient iterations, but on average it does select the correct model. We can see in our simulation that RMSE selected some incorrect models of 7, 8 and 9 parameters at $\sigma = 1$, and we see an even wider spread of model selections as we increase σ.

How does the level of noise affect the results?

As we can expect, the level of noise (σ value) greatly affects the variability of the selected model by RMSE. For the highest noise simulation with $\sigma = 4$, we see that RMSE selected nearly every type of the nine models. If we took many more iterations of the simulation, we could expect that RMSE would select each available model with some probability that depends on the noise level. Regardless, this method still selects the correct model on average, assuming that the noise isn't high enough to completely overpower the model.

Simulation Study 3

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

```
birthday = 20010606
set.seed(birthday)
```

We'll first define the various β_1, σ and n parameters that we will simulate with.

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

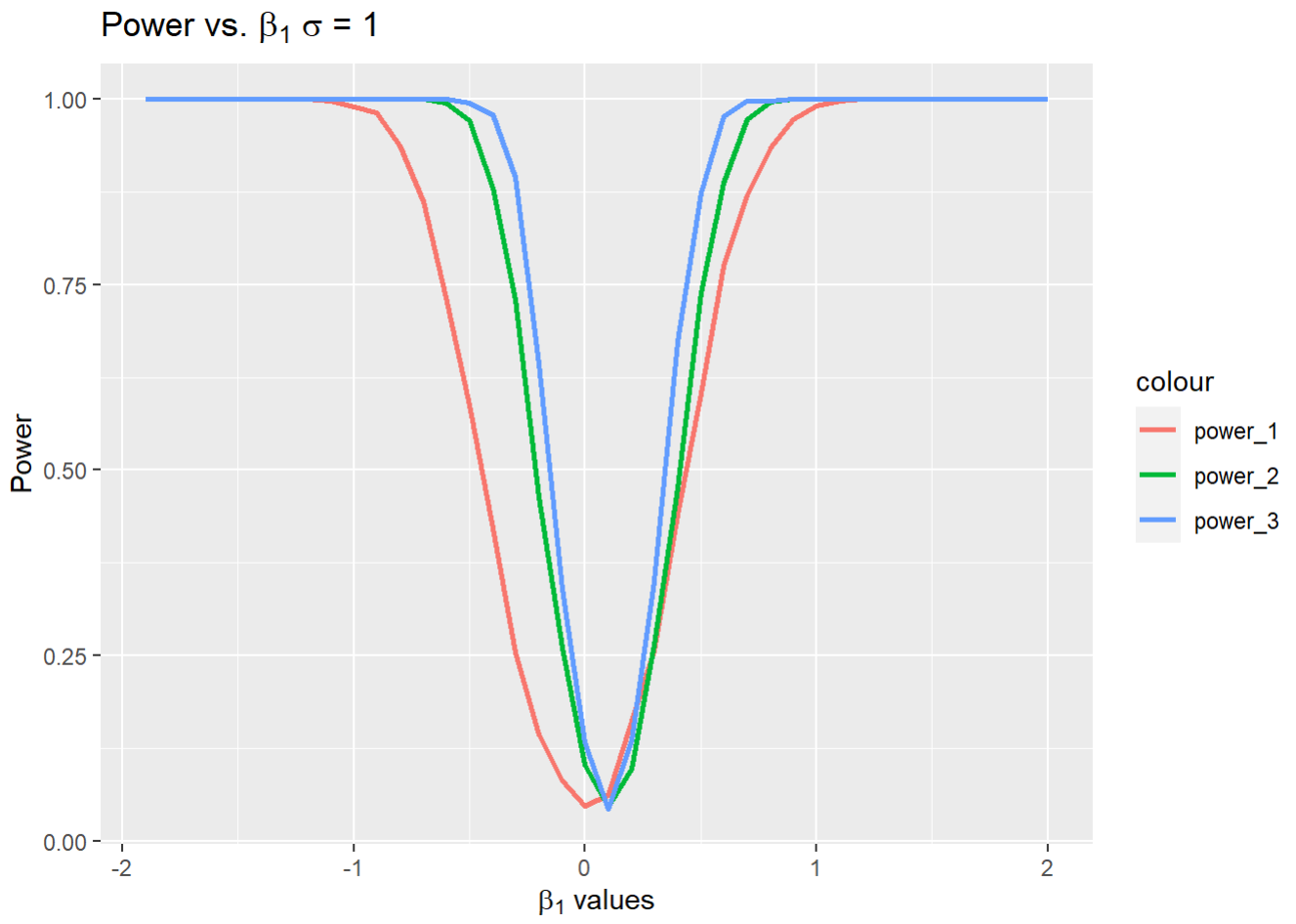
We can store the power data in a dataframe to easily retrieve the results later.

```
number_iterations = length(sigmas) * length(ns) * length(beta_1s)

power_data = data.frame(sigma = rep(0, number_iterations), n = rep(0, number_iterations),
                        beta_1 = rep(0, number_iterations), power = rep(0, number_iterations))
i=0
for (n in ns) {
  for (s in sigmas) {
    for (beta1 in beta_1s) {
      rejected = 0

      for (j in 1:1000) {

        eps = rnorm(n, mean = 0, sd = s)
        x_values = seq(0, 5, length.out = n)

        ys = beta_0 + beta1 * x_values + eps
        model = lm(ys ~ x_values)

        p = summary(model)$coefficients[2,4]

        if (p < alpha) {
          rejected = rejected + 1
        }

      }
      power = rejected / 1000
      power_data$sigma[i] = s
      power_data$n[i] = n
      power_data$beta_1[i] = beta1
      power_data$power[i] = power
      i = i + 1

    }
  }
}
```

Results

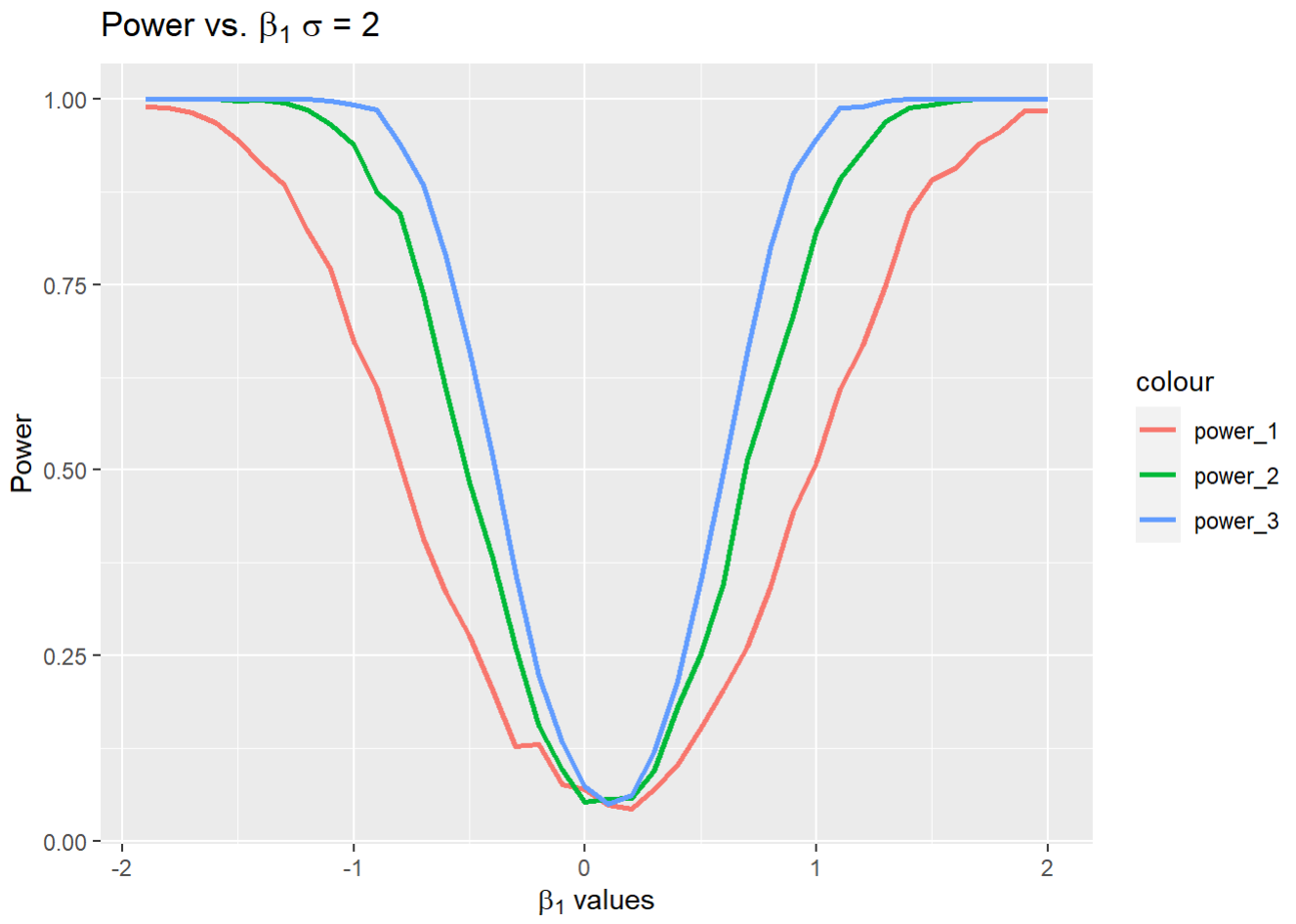
We can begin by plotting power vs β_1 at $\sigma = 1$:

```
library(ggplot2)
beta_1 = power_data[which((power_data$n == 10) & (power_data$sigma == 1)), c("beta_1")]
power1 = power_data[which((power_data$n == 10) & (power_data$sigma == 1)), c("power")]
power2 = power_data[which((power_data$n == 20) & (power_data$sigma == 1)), c("power")]
power3 = power_data[which((power_data$n == 30) & (power_data$sigma == 1)), c("power")]

power1 = power1[1:length(beta_1)]
power2 = power2[1:length(beta_1)]
power3 = power3[1:length(beta_1)]

power_graph_data = data.frame(beta_1 = beta_1, power_1 = power1, power_2 = power2, power_3 = pow
er3)
ggplot(data=power_graph_data, aes(x = beta_1)) + geom_line(aes(x = beta_1, y = power_1, color =
"power_1"), linewidth=1) +   geom_line(aes(x = beta_1, y = power_2, color = "power_2"), linewidt
h=1) +
  geom_line(aes(x = beta_1, y = power_3, color = "power_3"), linewidth=1) +
  xlab(expression(paste("", beta[1], " values"))) +
  ylab("Power") +
  ggtitle(expression(paste("Power vs. ", beta[1], " ", sigma, " = 1")))
```



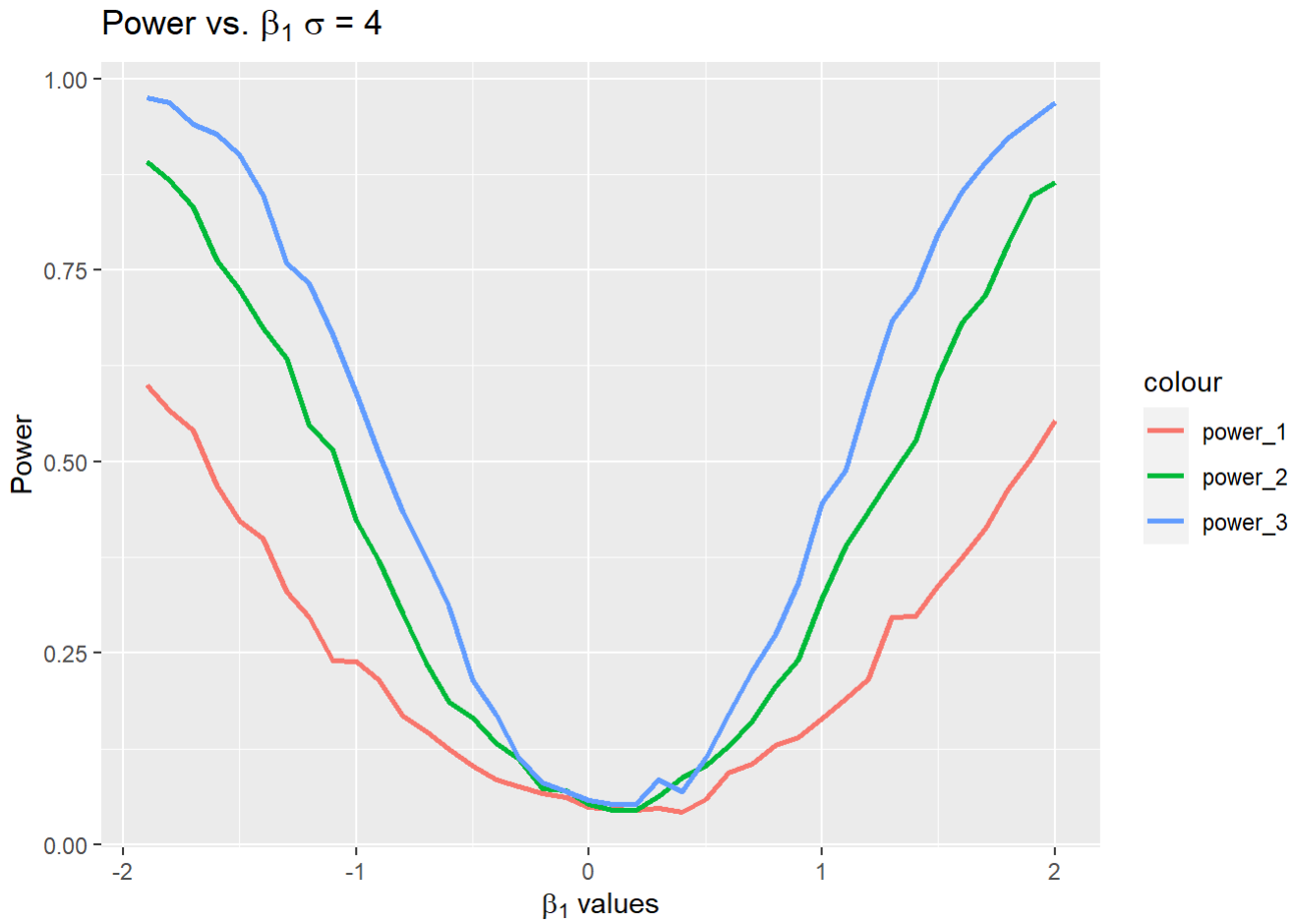
Now for $\sigma = 2$:

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

power1 = power1[1:length(beta_1)]
power2 = power2[1:length(beta_1)]
power3 = power3[1:length(beta_1)]


power_graph_data = data.frame(beta_1 = beta_1, power_1 = power1, power_2 = power2, power_3 = pow
er3)

ggplot(data=power_graph_data, aes(x = beta_1)) + geom_line(aes(x = beta_1, y = power_1, color =
"power_1"), linewidth=1) +   geom_line(aes(x = beta_1, y = power_2, color = "power_2"), linewidt
h=1) +
  geom_line(aes(x = beta_1, y = power_3, color = "power_3"), linewidth=1) +
  xlab(expression(paste(beta[1], " values"))) +
  ylab("Power") +
  ggtitle(expression(paste("Power vs. ", beta[1], " ", sigma, " = 2")))
```



And finally for $\sigma = 4$:

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

power1 = power1[1:length(beta_1)]
power2 = power2[1:length(beta_1)]
power3 = power3[1:length(beta_1)]

power_graph_data = data.frame(beta_1 = beta_1, power_1 = power1, power_2 = power2, power_3 = pow
er3)

ggplot(data=power_graph_data, aes(x = beta_1)) + geom_line(aes(x = beta_1, y = power_1, color =
"power_1"), linewidth=1) +   geom_line(aes(x = beta_1, y = power_2, color = "power_2"), linewidt
h=1) +
  geom_line(aes(x = beta_1, y = power_3, color = "power_3"), linewidth=1) +
  xlab(expression(paste(beta[1], " values"))) +
  ylab("Power") +
  ggtitle(expression(paste("Power vs. ", beta[1], " ", sigma, " = 4")))
```



Discussion

How do n, β_1, and σ affect power?

First looking at n, we can see that increasing sample size results in a much sharper power curve, such that power changes much quicker with respect to β_1. The histograms show that average power increases with greater sample size n. This relationship between n and power makes intuitive sense, as increasing sample size should in turn

increase the model's resilience to outlier data points and lead to greater explanatory power.

For β_1, we can see that power follows an expected curve that nears zero power at $\beta_1 = 0$ and power of around 1 a β_1 approaches -1 or +1. Furthermore, as we move away from our null hypothesis that $\beta_1 = 0$ we are able to see that power quickly increases and gives us more confidence to reject the null hypothesis given that the alternative hypothesis is correct.

Finally, as σ increases we can see that that the power curves begin to significantly flatten out. Since higher σ values lead to increased variance in the epsilon error terms, and thus greater variation in the models overall, the models tend to lose significance and explanatory power. The σ values will directly impact the β_1 coefficient, which in turn affects the conclusion of the significance of regression test and thus power.

Are 1000 simulations sufficient?

I believe that 1000 simulations is sufficient for our purposes. The above plots from our simulations clearly depict the expected relationships between n, β_1 and σ across the different values of each. Additional simulations or additional values of the different parameters would likely lead to an even more clear analysis, but considering the trade-offs between simulation/model size and computing time, I consider 1000 simulations to be effective.