Week 6 - Midterm Assignment: Simulation Project

STAT 420, Summer 2022, Ilya Andreev, iandre3@illinois.edu

Contents

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.

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

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
- R^2

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

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

We thus simulate the `y` vector $2(models)Ö3(sigmas)Ö2000(sims) = 12000$ times.

Methods

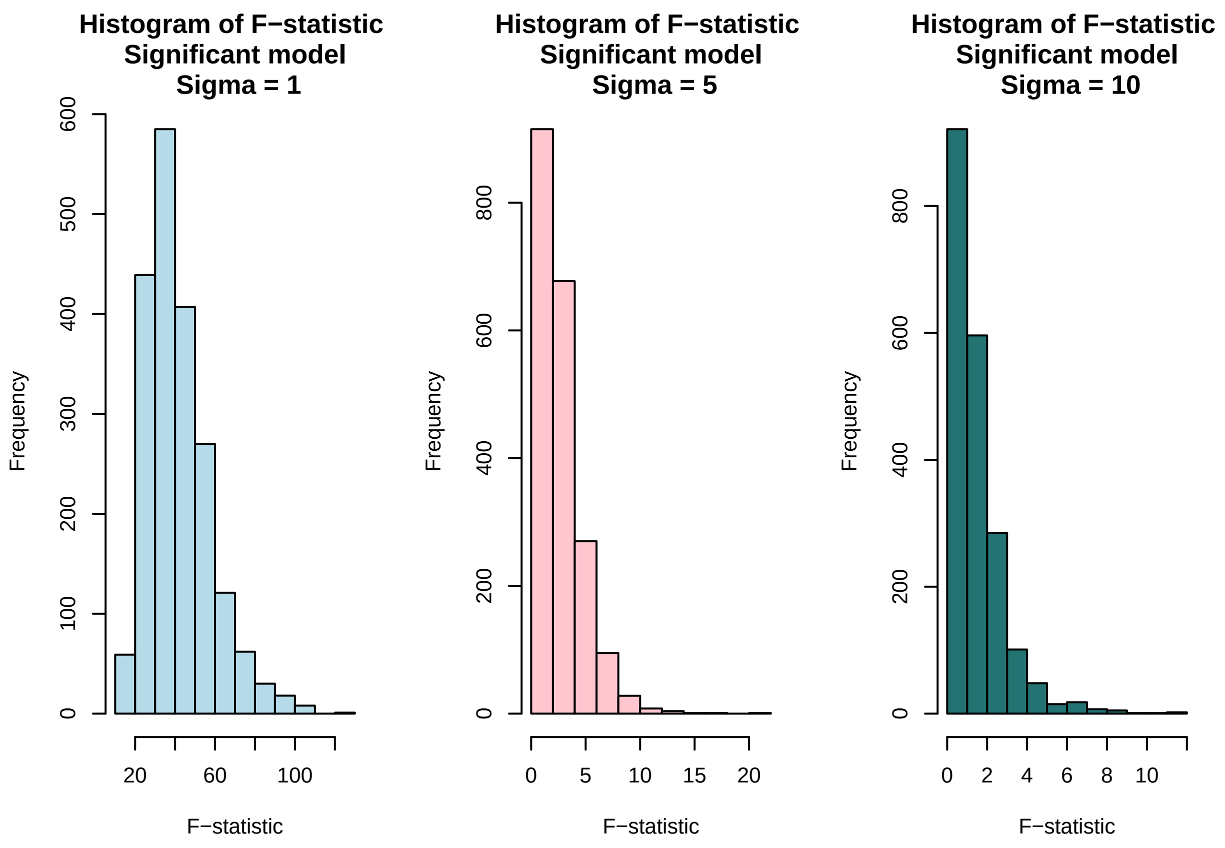
```
birthday = 19991124
set.seed(birthday)
```

```
n = 25
data = read.csv("study_1.csv")
sigmas = c(1, 5, 10)
f1 = list(rep(0, 2000), rep(0, 2000), rep(0, 2000))
p1 = list(rep(0, 2000), rep(0, 2000), rep(0, 2000))
r1 = list(rep(0, 2000), rep(0, 2000), rep(0, 2000))

f2 = list(rep(0, 2000), rep(0, 2000), rep(0, 2000))
p2 = list(rep(0, 2000), rep(0, 2000), rep(0, 2000))
r2 = list(rep(0, 2000), rep(0, 2000), rep(0, 2000))

y1 = function(x1, x2, x3, sigma) {
  3 + 1 * x1 + 1 * x2 + 1 * x3 + rnorm(25, 0, sigma)
}

y2 = function(x1, x2, x3, sigma) {
  3 + 0 * x1 + 0 * x2 + 0 * x3 + rnorm(25, 0, sigma)
}


for (i in 1:3) {
  for (j in 1:2000) {
    sigma = sigmas[i]
    y_sim= y1(data$x1, data$x2, data$x3, sigma)
    m = lm(y_sim ~ data$x1 + data$x2 + data$x3)
    s = summary(m)
    f = s$fstatistic[1]
```

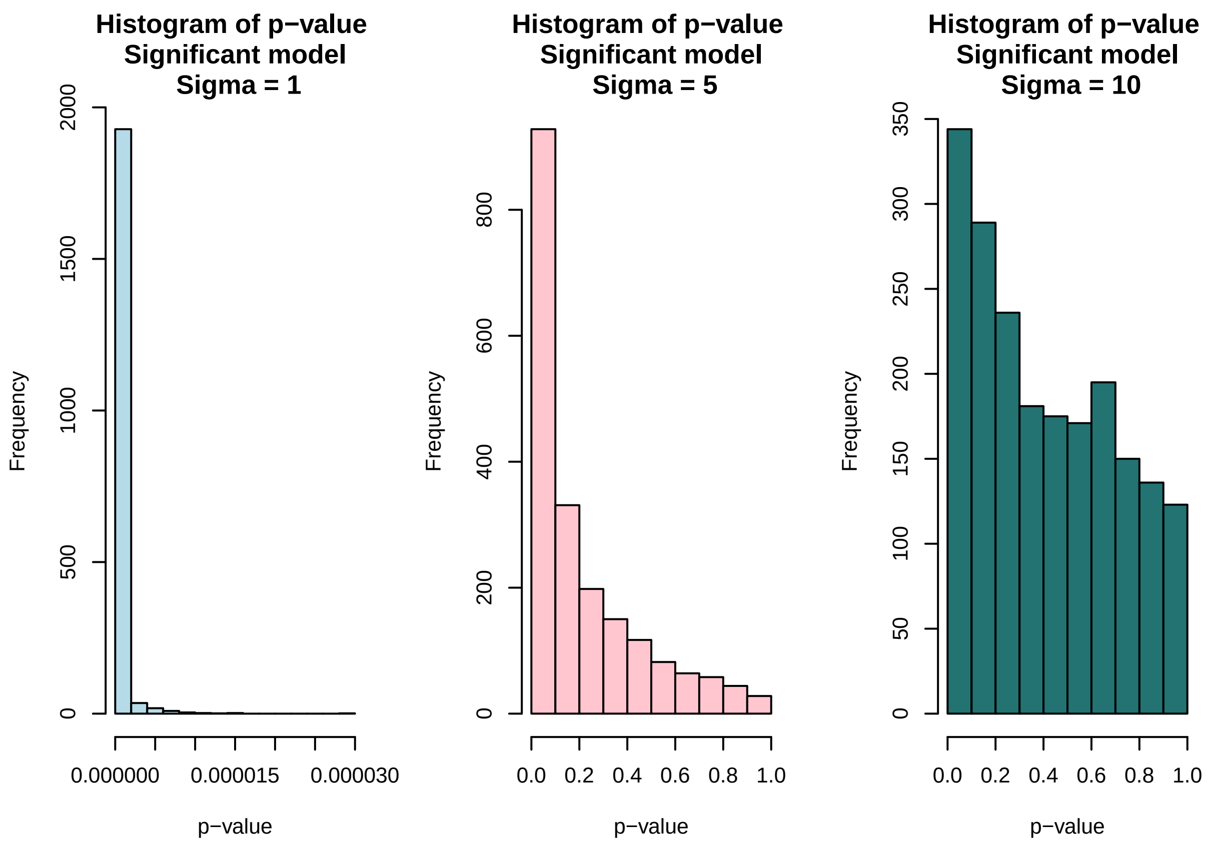
```r
    p = pf(s$fstatistic[1], s$fstatistic[2], s$fstatistic[3], lower.tail=FALSE)
    r = s$r.squared

    f1[[i]][j] = f
    p1[[i]][j] = p
    r1[[i]][j] = r
  }
}

for (i in 1:3) {
  for (j in 1:2000) {
    sigma = sigmas[i]
    y_sim= y2(data$x1, data$x2, data$x3, sigma)
    m = lm(y_sim ~ data$x1 + data$x2 + data$x3)
    s = summary(m)
    f = s$fstatistic[1]
    p = pf(s$fstatistic[1], s$fstatistic[2], s$fstatistic[3], lower.tail=FALSE)
    r = s$r.squared

    f2[[i]][j] = f
    p2[[i]][j] = p
    r2[[i]][j] = r
  }
}
```

Results

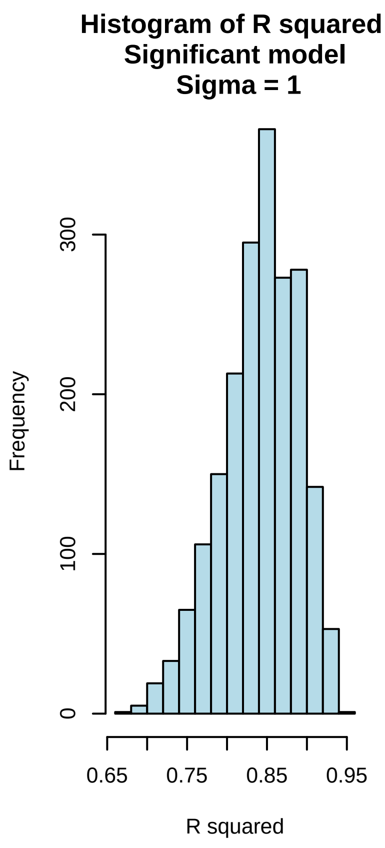
```r
par(mfrow = c(1, 3))
c1 = rgb(173, 216, 230, max = 255, alpha = 230)
c2 = rgb(255, 192, 203, max = 255, alpha = 230)
c3 = rgb(11, 100, 99, max = 255, alpha = 230)

hist(f1[[1]],
     col=c1,
     main="Histogram of F-statistic \n Significant model \n Sigma = 1",
     xlab="F-statistic")
hist(f1[[2]],
     col=c2,
     main="Histogram of F-statistic \n Significant model \n Sigma = 5",
     xlab="F-statistic")
hist(f1[[3]],
     col=c3,
     main="Histogram of F-statistic \n Significant model \n Sigma = 10",
     xlab="F-statistic")
```

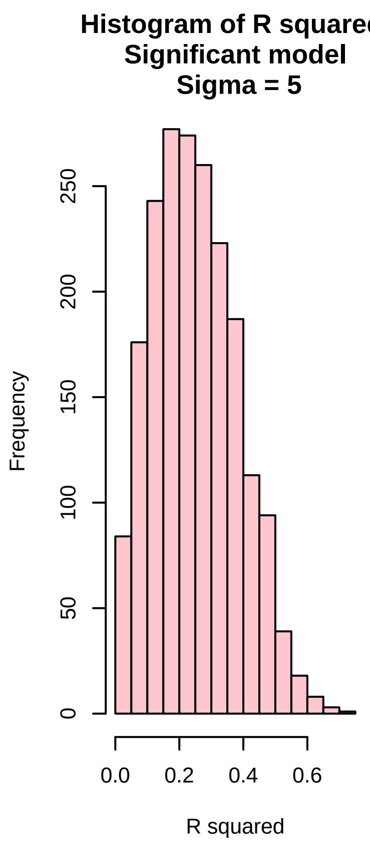


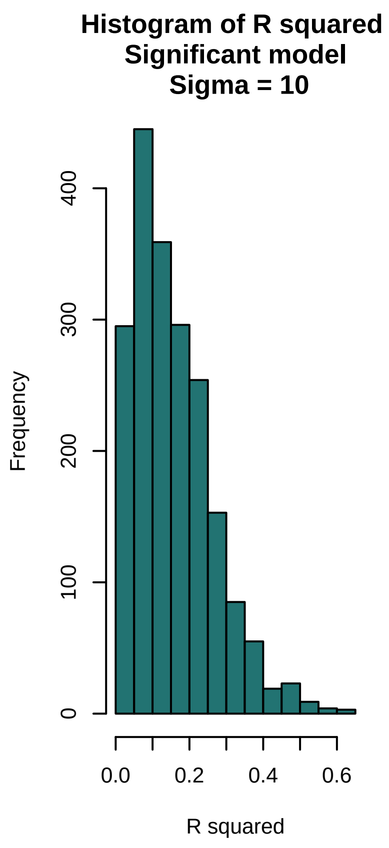
```
hist(p1[[1]],
     col=c1,
     main="Histogram of p-value \n Significant model \n Sigma = 1",
     xlab="p-value")
hist(p1[[2]],
     col=c2,
     main="Histogram of p-value \n Significant model \n Sigma = 5", xlab="p-value")
hist(p1[[3]],
     col=c3,
     main="Histogram of p-value \n Significant model \n Sigma = 10",
     xlab="p-value")
```



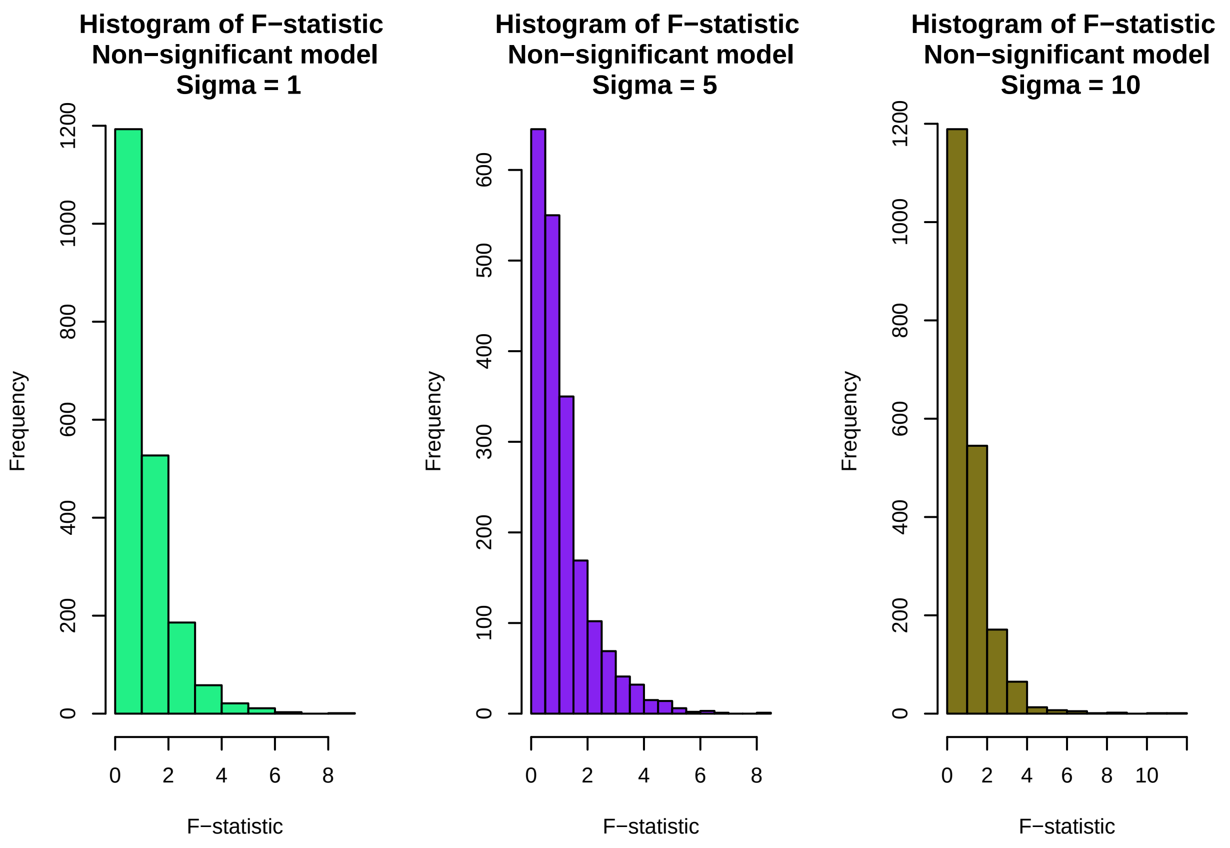
```
hist(r1[[1]],
     col=c1,
     main="Histogram of R squared \n Significant model \n Sigma = 1",
     xlab="R squared")
hist(r1[[2]],
     col=c2,
     main="Histogram of R squared \n Significant model \n Sigma = 5",
     xlab="R squared")
hist(r1[[3]],
     col=c3,
     main="Histogram of R squared \n Significant model \n Sigma = 10",
     xlab="R squared")
```



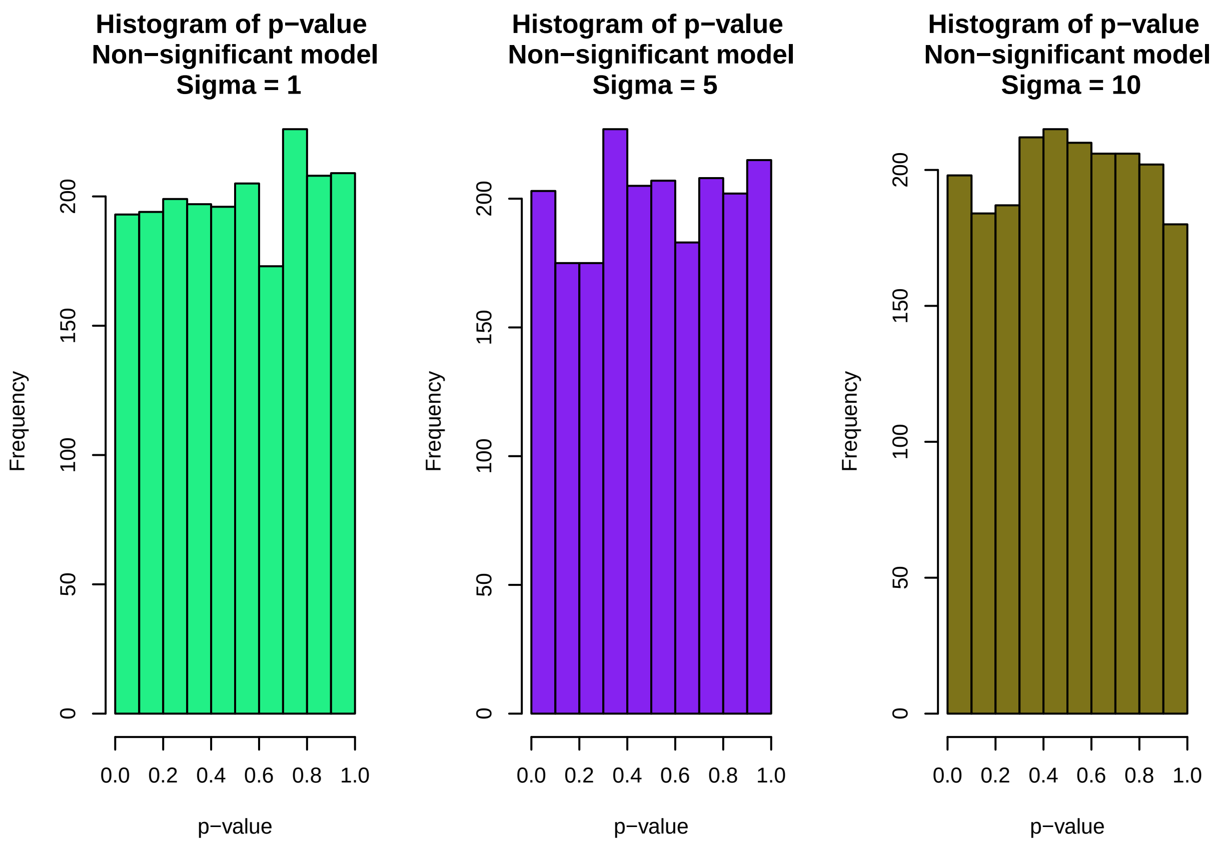




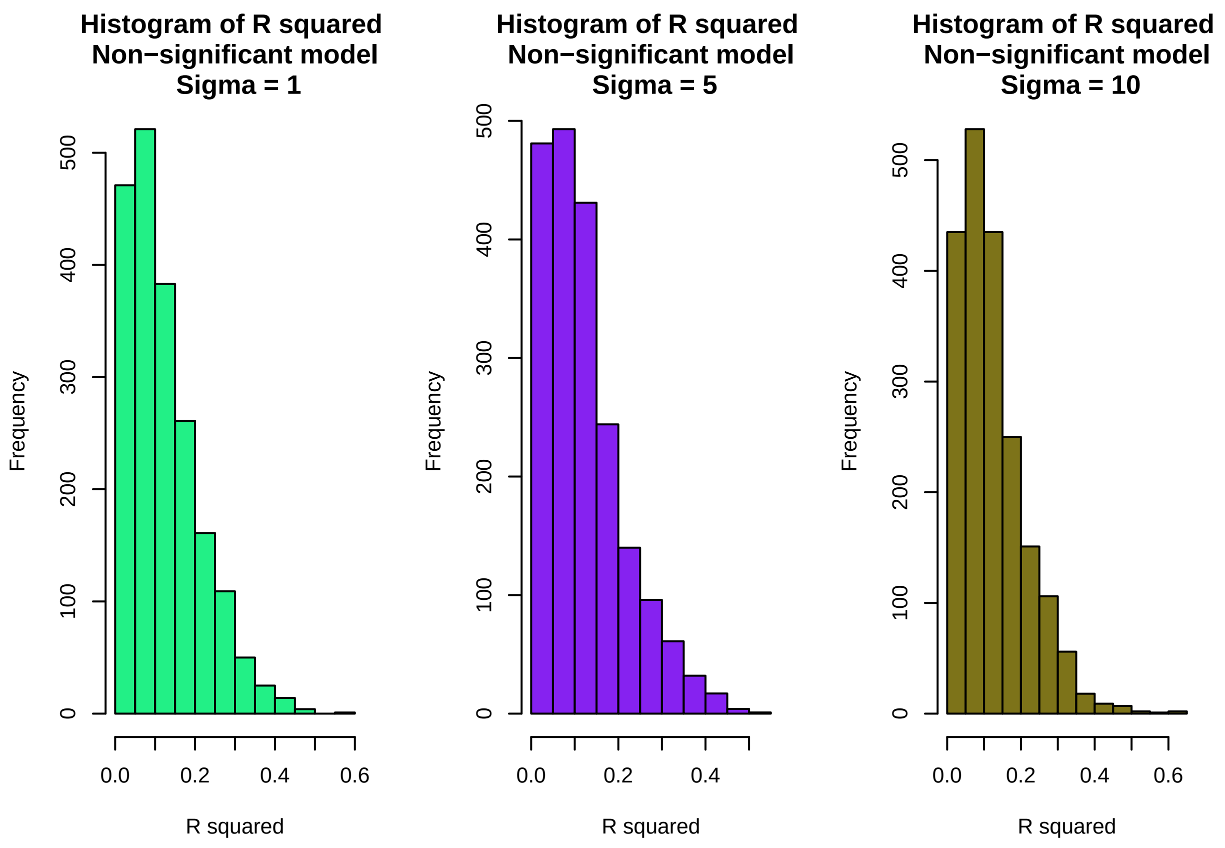
```
c1 = rgb(11, 239, 121, max = 255, alpha = 230)
c2 = rgb(121, 11, 239, max = 255, alpha = 230)
c3 = rgb(111, 100, 1, max = 255, alpha = 230)

hist(f2[[1]],
     col=c1,
     main="Histogram of F-statistic \n Non-significant model \n Sigma = 1",
     xlab="F-statistic")
hist(f2[[2]],
     col=c2,
     main="Histogram of F-statistic \n Non-significant model \n Sigma = 5",
     xlab="F-statistic")
hist(f2[[3]],
     col=c3,
     main="Histogram of F-statistic \n Non-significant model \n Sigma = 10",
     xlab="F-statistic")
```



```r
hist(p2[[1]],
     col=c1,
     main="Histogram of p-value \n Non-significant model \n Sigma = 1",
     xlab="p-value")
hist(p2[[2]],
     col=c2,
     main="Histogram of p-value \n Non-significant model \n Sigma = 5",
     xlab="p-value")
hist(p2[[3]],
     col=c3,
     main="Histogram of p-value \n Non-significant model \n Sigma = 10",
     xlab="p-value")
```



```
hist(r2[[1]],
     col=c1,
     main="Histogram of R squared \n Non-significant model \n Sigma = 1",
     xlab="R squared")
hist(r2[[2]],
     col=c2,
     main="Histogram of R squared \n Non-significant model \n Sigma = 5",
     xlab="R squared")
hist(r2[[3]],
     col=c3,
     main="Histogram of R squared \n Non-significant model \n Sigma = 10",
     xlab="R squared")
```



Discussion

In this simulation study, we observe that for the non-significant model, the distributions of values reflecting model behavior do not appear to be dependent on the value of σ. Specifically, for the non-significant model, the distributions of p-value, F-statistic, and R^2 are similar regardless of the value of σ and are all indicative of the model's poor predictive performance. The low values of F and the uniform distribution of p-values tell us that the model fails the significance of regression tests, and low right-skewed R^2 values show us that the non-significant model has low predictive capacity.

Contrast this with the significant model. Clearly, at low σ the model passes the significance of regression test at nearly any alpha, and has high predictive performance as shown by R^2. Even though we see that the model performance naturally declines at higher noise levels, it still outperforms the non-significant model in terms of R^2. Further, even at high noise levels, the significant model distribution of the p-value is most dense at low p-values, meaning that even with high noise the model often passes the significance of regression test.

As such, we have demonstrated how noise levels impact significant and non-significant model behavior.

Simulation Study 2: Using RMSE for Selection?

Introduction

In this simulation study we will investigate how well the procedure of using RMSE for model selection 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 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.

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

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

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

For each, we 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, we calculate Train and Test RMSE.

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

We repeat this process with 1000 simulations for each of the 3 values of σ. For each value of σ, we create a plot that shows how average Train RMSE and average Test RMSE changes as a function of model size. We also show the number of times the model of each size was chosen for each value of σ.

We thus simulate the y vector 3Ö1000 = 3000 times. We fit 9Ö3Ö1000 = 27000 models.

Methods

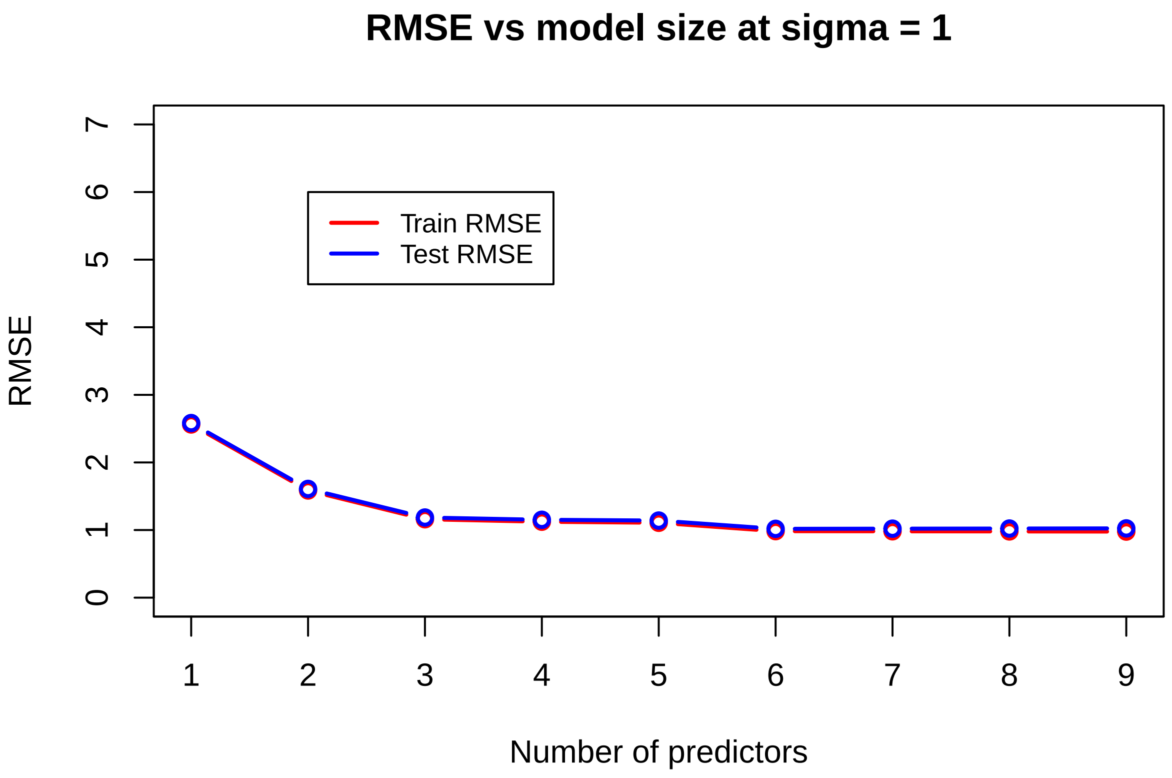
```r
birthday = 19991124
set.seed(birthday)
```

```r
n = 500
sigmas = c(1, 2, 4)
train_results = list(
  list(
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000)
  ),
  list(
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000)
  ),
  list(
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000)
  )
)

test_results = list(
  list(
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000)
  ),
  list(
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
```

```r
    rep(0, 1000),
    rep(0, 1000)
  ),
  list(
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000),
    rep(0, 1000)
  )
)

data = read.csv("study_2.csv")

y = function(x1, x2, x3, x4, x5, x6, sigma) {
  0 + 3 * x1 - 4 * x2 + 1.6 * x3 - 1.1 * x4 + 0.7 * x5 + 0.5 * x6 + rnorm(n, 0, sigma)
}

for (i in 1:3) {
  for (j in 1:1000) {
    sigma = sigmas[i]
    y_sim = y(data$x1, data$x2, data$x3, data$x4, data$x5, data$x6, sigma)
    data["y"] = y_sim

    train_idx = sample(seq_len(500), 250)
    train_data = data[train_idx, ]
    test_data = data[-train_idx, ]

    m1 = lm(y ~ x1, train_data)
    m2 = lm(y ~ x1 + x2, train_data)
    m3 = lm(y ~ x1 + x2 + x3, train_data)
    m4 = lm(y ~ x1 + x2 + x3 + x4, train_data)
    m5 = lm(y ~ x1 + x2 + x3 + x4 + x5, train_data)
    m6 = lm(y ~ x1 + x2 + x3 + x4 + x5 + x6, train_data)
    m7 = lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7, train_data)
    m8 = lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8, train_data)
    m9 = lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9, train_data)

    rmse = function(y, y_hat) {
      sqrt((1 / length(y)) * sum((y - y_hat) ** 2))
    }

    m1_res_train = rmse(train_data$y, predict(m1))
    train_results[[i]][[1]][j] = m1_res_train
    m2_res_train = rmse(train_data$y, predict(m2))
    train_results[[i]][[2]][j] = m2_res_train
    m3_res_train = rmse(train_data$y, predict(m3))
    train_results[[i]][[3]][j] = m3_res_train
    m4_res_train = rmse(train_data$y, predict(m4))
    train_results[[i]][[4]][j] = m4_res_train
    m5_res_train = rmse(train_data$y, predict(m5))
    train_results[[i]][[5]][j] = m5_res_train
    m6_res_train = rmse(train_data$y, predict(m6))
    train_results[[i]][[6]][j] = m6_res_train
    m7_res_train = rmse(train_data$y, predict(m7))
    train_results[[i]][[7]][j] = m7_res_train
    m8_res_train = rmse(train_data$y, predict(m8))
    train_results[[i]][[8]][j] = m8_res_train
    m9_res_train = rmse(train_data$y, predict(m9))
    train_results[[i]][[9]][j] = m9_res_train

    m1_res_test = rmse(test_data$y, predict(m1, newdata=test_data))
    test_results[[i]][[1]][j] = m1_res_test
    m2_res_test = rmse(test_data$y, predict(m2, newdata=test_data))
    test_results[[i]][[2]][j] = m2_res_test
    m3_res_test = rmse(test_data$y, predict(m3, newdata=test_data))
    test_results[[i]][[3]][j] = m3_res_test
    m4_res_test = rmse(test_data$y, predict(m4, newdata=test_data))
    test_results[[i]][[4]][j] = m4_res_test
    m5_res_test = rmse(test_data$y, predict(m5, newdata=test_data))
    test_results[[i]][[5]][j] = m5_res_test
    m6_res_test = rmse(test_data$y, predict(m6, newdata=test_data))
    test_results[[i]][[6]][j] = m6_res_test
    m7_res_test = rmse(test_data$y, predict(m7, newdata=test_data))
    test_results[[i]][[7]][j] = m7_res_test
    m8_res_test = rmse(test_data$y, predict(m8, newdata=test_data))
    test_results[[i]][[8]][j] = m8_res_test
    m9_res_test = rmse(test_data$y, predict(m9, newdata=test_data))
    test_results[[i]][[9]][j] = m9_res_test
  }
}
```

Results

```
x = 1:9

count_model_wins = function(
                      rmse_1,
                      rmse_2,
                      rmse_3,
                      rmse_4,
                      rmse_5,
                      rmse_6,
                      rmse_7,
                      rmse_8,
                      rmse_9) {
  res = rep(0, 9)
  for (i in 1:length(rmse_1)) {
    ind = which.min(
      c(rmse_1[i],
        rmse_2[i],
        rmse_3[i],
        rmse_4[i],
        rmse_5[i],
        rmse_6[i],
        rmse_7[i],
        rmse_8[i],
        rmse_9[i]))
    res[ind] = res[ind] + 1
  }

  res
}

plot(
  y=c(mean(train_results[[1]][[1]]),
      mean(train_results[[1]][[2]]),
      mean(train_results[[1]][[3]]),
      mean(train_results[[1]][[4]]),
      mean(train_results[[1]][[5]]),
      mean(train_results[[1]][[6]]),
      mean(train_results[[1]][[7]]),
      mean(train_results[[1]][[8]]),
      mean(train_results[[1]][[9]])),
  x=x,
  xaxp=c(0, 10, 10),
  type="b",
  col="red",
  main="RMSE vs model size at sigma = 1",
  xlab="Number of predictors",
  ylab="RMSE",
  lwd=2,
  ylim=c(0, 7),
  xlim=c(1, 9))
lines(
  y=c(mean(test_results[[1]][[1]]),
      mean(test_results[[1]][[2]]),
      mean(test_results[[1]][[3]]),
      mean(test_results[[1]][[4]]),
      mean(test_results[[1]][[5]]),
      mean(test_results[[1]][[6]]),
      mean(test_results[[1]][[7]]),
      mean(test_results[[1]][[8]]),
      mean(test_results[[1]][[9]])),
  x=x,
  type="b",
  col="blue",
  lwd=2)
legend(
  2,
  6,
  legend=c("Train RMSE", "Test RMSE"),
  col=c("red", "blue"),
  cex=0.8,
  lwd=2,
  lty=1)
```

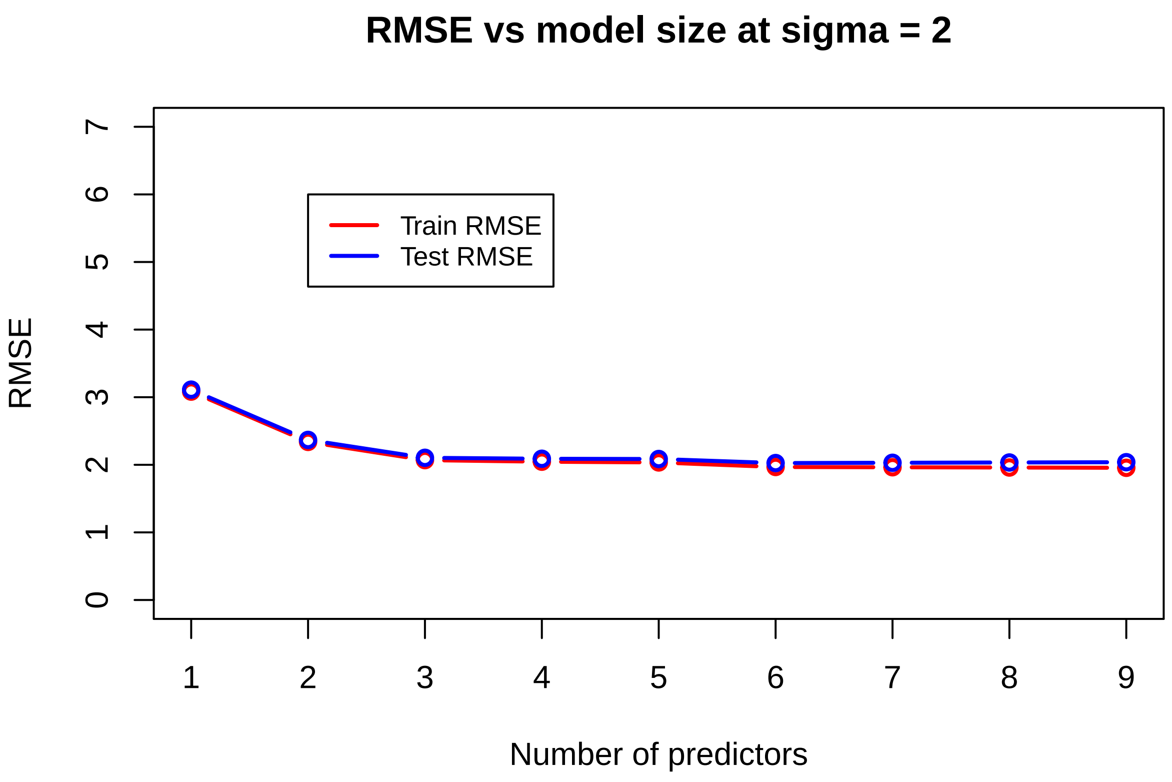


We see that with $\sigma = 1$, the model with 6 parameters is selected most often based on test RMSE.

```
res = count_model_wins(
  test_results[[1]][[1]],
  test_results[[1]][[2]],
  test_results[[1]][[3]],
  test_results[[1]][[4]],
  test_results[[1]][[5]],
  test_results[[1]][[6]],
  test_results[[1]][[7]],
  test_results[[1]][[8]],
  test_results[[1]][[9]])
d = data.frame(
  Model = c(
    "Model 1",
    "Model 2",
    "Model 3",
    "Model 4",
    "Model 5",
    "Model 6",
    "Model 7",
    "Model 8",
    "Model 9"),
  Selections = res)
knitr::kable(d)
```

Model	Selections
Model 1	0
Model 2	0
Model 3	0
Model 4	1
Model 5	1
Model 6	549
Model 7	215
Model 8	123
Model 9	111

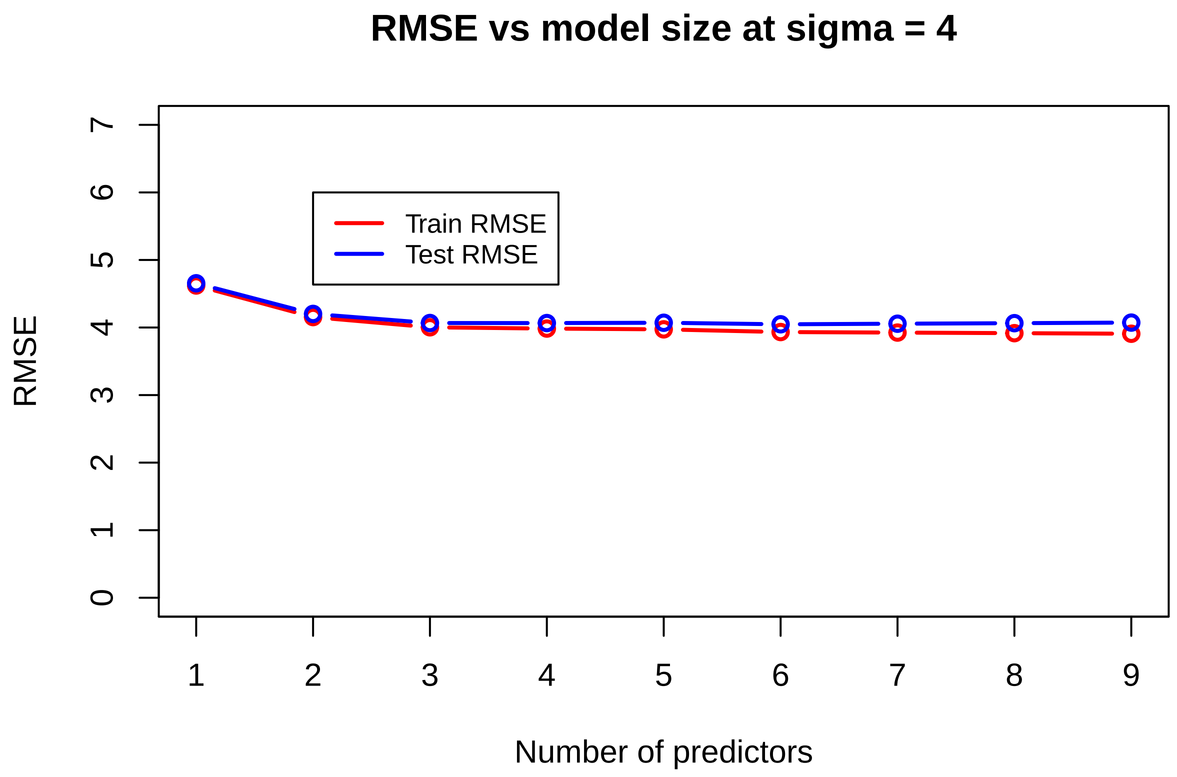
```r
x = 1:9

plot(
  y=c(mean(train_results[[2]][[1]]),
      mean(train_results[[2]][[2]]),
      mean(train_results[[2]][[3]]),
      mean(train_results[[2]][[4]]),
      mean(train_results[[2]][[5]]),
      mean(train_results[[2]][[6]]),
      mean(train_results[[2]][[7]]),
      mean(train_results[[2]][[8]]),
      mean(train_results[[2]][[9]])),
  x=x,
  xaxp=c(0, 10, 10),
  type="b",
  col="red",
  main="RMSE vs model size at sigma = 2",
  xlab="Number of predictors",
  ylab="RMSE",
  lwd=2,
  ylim=c(0, 7),
  xlim=c(1, 9))
lines(
  y=c(mean(test_results[[2]][[1]]),
      mean(test_results[[2]][[2]]),
      mean(test_results[[2]][[3]]),
      mean(test_results[[2]][[4]]),
      mean(test_results[[2]][[5]]),
      mean(test_results[[2]][[6]]),
      mean(test_results[[2]][[7]]),
      mean(test_results[[2]][[8]]),
      mean(test_results[[2]][[9]])),
  x=x,
  type="b",
  col="blue",
  lwd=2)
legend(
  2,
  6,
  legend=c("Train RMSE", "Test RMSE"),
  col=c("red", "blue"),
  cex=0.8,
  lwd=2,
  lty=1)
```



We see that with $\sigma = 2$, the model with 6 parameters is selected most often based on test RMSE.

Model	Selections
Model 1	0
Model 2	0
Model 3	22
Model 4	23
Model 5	29
Model 6	495
Model 7	203
Model 8	124
Model 9	104

```
x = 1:9

plot(
  y=c(mean(train_results[[3]][[1]]),
      mean(train_results[[3]][[2]]),
      mean(train_results[[3]][[3]]),
      mean(train_results[[3]][[4]]),
      mean(train_results[[3]][[5]]),
      mean(train_results[[3]][[6]]),
      mean(train_results[[3]][[7]]),
      mean(train_results[[3]][[8]]),
      mean(train_results[[3]][[9]])),
  x=x,
  xaxp=c(0, 10, 10),
  type="b",
  col="red",
  main="RMSE vs model size at sigma = 4",
  xlab="Number of predictors",
  ylab="RMSE",
  lwd=2,
  ylim=c(0, 7),
  xlim=c(1, 9))
lines(
  y=c(mean(test_results[[3]][[1]]),
      mean(test_results[[3]][[2]]),
      mean(test_results[[3]][[3]]),
      mean(test_results[[3]][[4]]),
      mean(test_results[[3]][[5]]),
      mean(test_results[[3]][[6]]),
      mean(test_results[[3]][[7]]),
      mean(test_results[[3]][[8]]),
      mean(test_results[[3]][[9]])),
  x=x,
  type="b",
  col="blue",
  lwd=2)
legend(
  2,
  6,
  legend=c("Train RMSE", "Test RMSE"),
  col=c("red", "blue"),
  cex=0.8,
  lwd=2,
  lty=1)
```



We see that with $\sigma = 4$, the model with 6 parameters is selected most often based on test RMSE.

```
res = count_model_wins(
  test_results[[3]][[1]],
  test_results[[3]][[2]],
  test_results[[3]][[3]],
  test_results[[3]][[4]],
  test_results[[3]][[5]],
  test_results[[3]][[6]],
  test_results[[3]][[7]],
  test_results[[3]][[8]],
  test_results[[3]][[9]])
d = data.frame(
  Model = c(
    "Model 1",
    "Model 2",
    "Model 3",
    "Model 4",
    "Model 5",
    "Model 6",
    "Model 7",
    "Model 8",
    "Model 9"),
  Selections = res)
knitr::kable(d)
```

Model	Selections
Model 1	0
Model 2	16
Model 3	171

Model	Selections
Model 4	119
Model 5	64
Model 6	342
Model 7	116
Model 8	91
Model 9	81

Discussion

We clearly see that the test RMSE method most often selects the correct model for all values of σ. However, it is evident that the performance of all models degrades with increase in noise level, and the performance of different models "evens out" with higher values of σ. Our expectation is that if we were to increase the noise level even further, all models would be selected equally often.

While not displayed in any of the data, what we also found noteworthy is that the most complex model is always selected based on train RMSE. This is evidence of the fact that overfitting occurs for the more complex models with irrelevant predictors, and it is prudent to use *test* RMSE, not *train* RMSE, for model selection.

Simulation Study 3: Power

Introduction

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

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

We had previously 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 $\alpha = 0.05$.

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

We create three plots, one for each value of σ. Within each of these plots, we add a "power curve" for each value of n that shows how power is affected by signal strength, β_1.

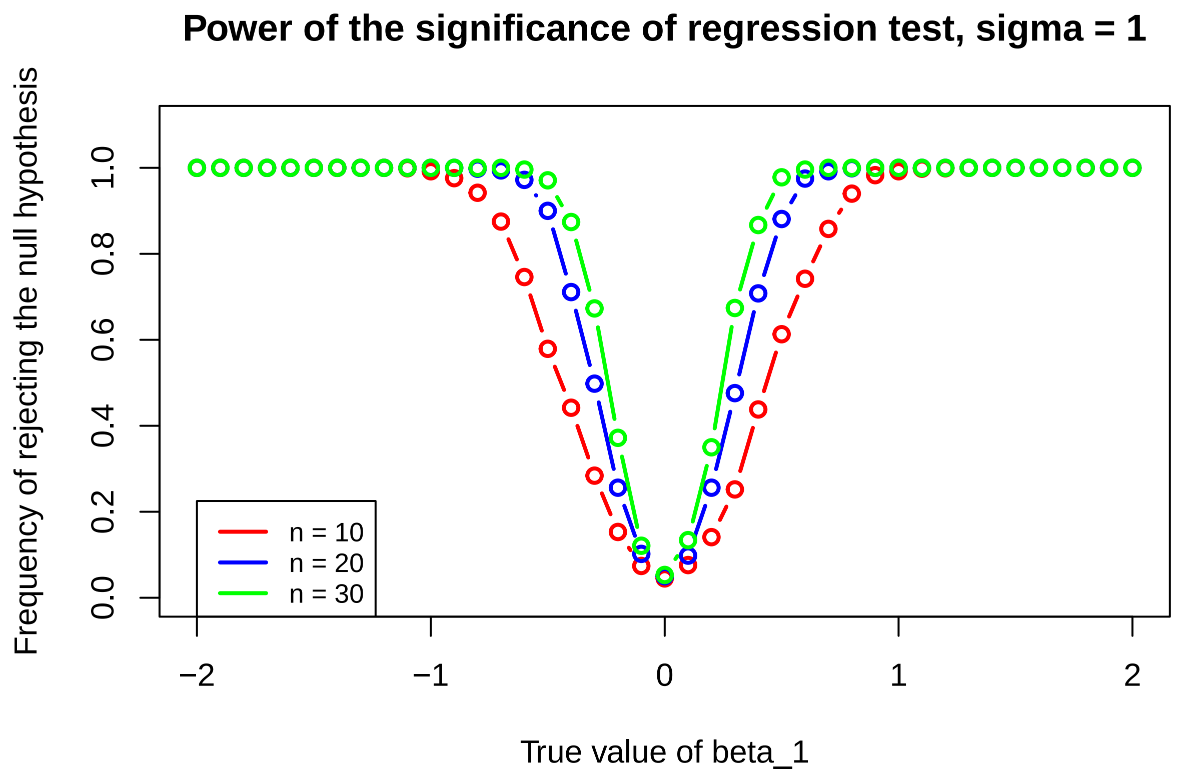
Methods

```
birthday = 19991124
set.seed(birthday)
```

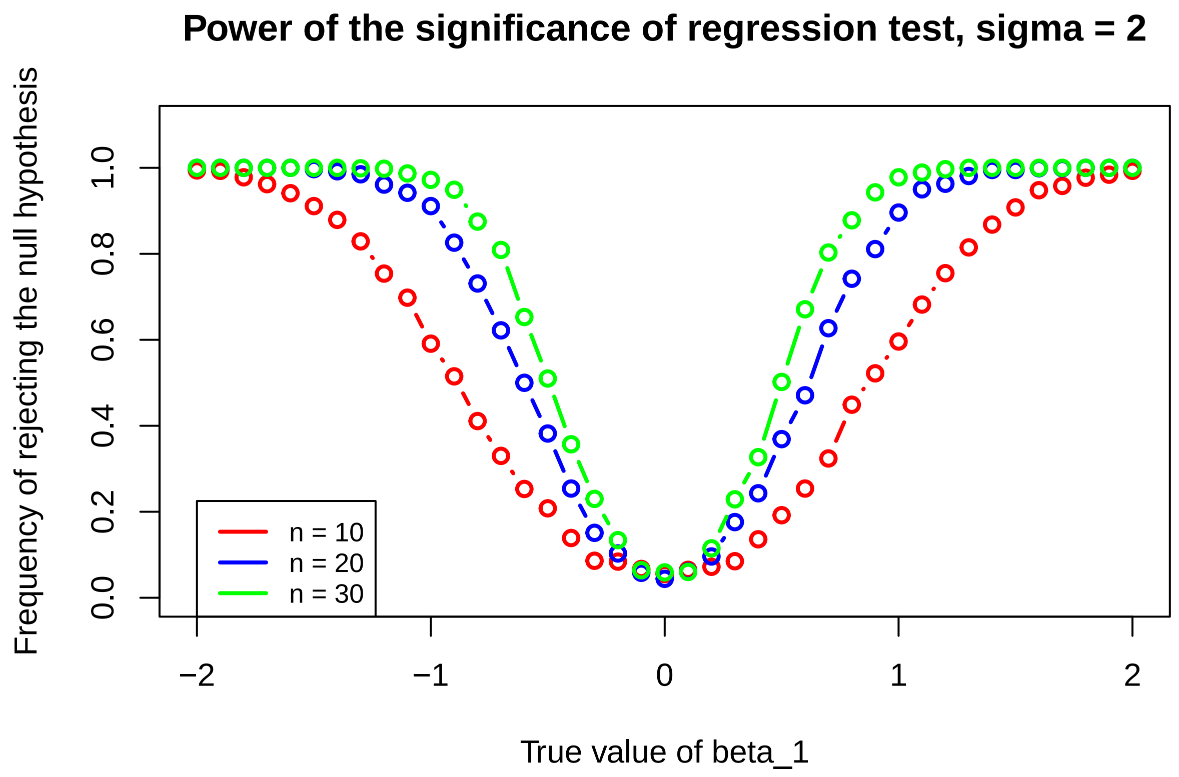
```
sigmas = c(1, 2, 4)
b1s = seq(-2, 2, 0.1)
ns = c(10, 20, 30)
alpha = 0.05
rejections_counters = list(
  list(rep(0, length(b1s)), rep(0, length(b1s)), rep(0, length(b1s))),
  list(rep(0, length(b1s)), rep(0, length(b1s)), rep(0, length(b1s))),
  list(rep(0, length(b1s)), rep(0, length(b1s)), rep(0, length(b1s)))
)

y = function(x, b1, sigma, n) {
  b1 * x + rnorm(n, 0, sigma)
}

for (i in 1:3) {
  sigma = sigmas[i]
  for (j in 1:3) {
    n = ns[j]
    x_values = seq(0, 5, length=n)
    for (k in 1:length(b1s)) {
      b1 = b1s[k]
      for (l in 1:1000) {
        y_sim = y(x_values, b1, sigma, n)
        m = lm(y_sim ~ x_values)
        s = summary(m)
        p_val = pf(s$fstatistic[1], s$fstatistic[2], s$fstatistic[3], lower.tail=FALSE)
        m = rejections_counters[[i]][[j]][k]
        rejections_counters[[i]][[j]][k] =  m + (p_val < alpha)
      }
    }
  }
}
```

Results

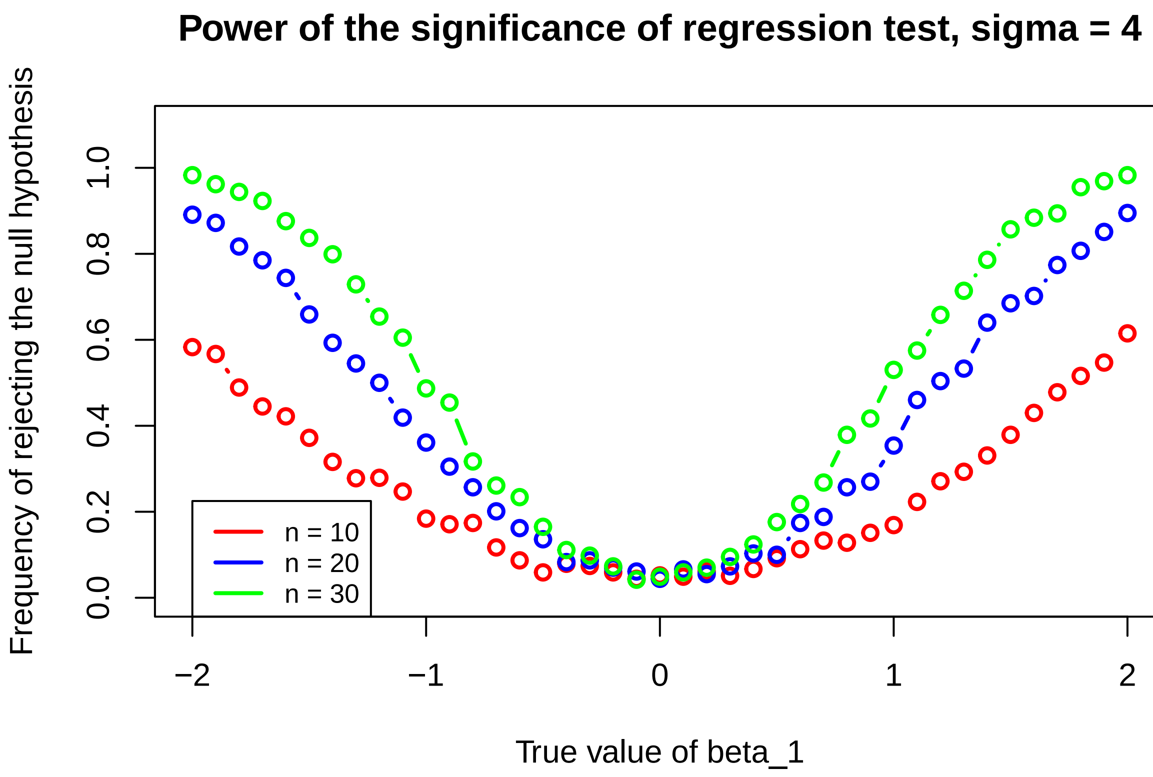
```
x = b1s
plot(y=rejections_counters[[1]][[1]] / 1000,
  x=x,
  type="b",
  col="red",
  main="Power of the significance of regression test, sigma = 1",
  xlab="True value of beta_1",
  ylab="Frequency of rejecting the null hypothesis",
  lwd=2,
  ylim=c(0, 1.1))
lines(y=rejections_counters[[1]][[2]] / 1000,
      x=x,
      type="b",
      col="blue",
      lwd=2)
lines(y=rejections_counters[[1]][[3]] / 1000,
      x=x,
      type="b",
      col="green",
      lwd=2)
legend(-2,
       0.225,
       legend=c("n = 10", "n = 20", "n = 30"),
       col=c("red", "blue", "green"),
       cex=0.8,
       lwd=2,
       lty=1)
```



```
x = b1s

plot(y=rejections_counters[[2]][[1]] / 1000,
     x=x,
     type="b",
     col="red",
     main="Power of the significance of regression test, sigma = 2",
     xlab="True value of beta_1",
     ylab="Frequency of rejecting the null hypothesis",
     lwd=2,
     ylim=c(0, 1.1))
lines(y=rejections_counters[[2]][[2]] / 1000,
      x=x,
      type="b",
      col="blue",
      lwd=2)
lines(y=rejections_counters[[2]][[3]] / 1000,
      x=x,
      type="b",
      col="green",
      lwd=2)
legend(-2,
       0.225,
       legend=c("n = 10", "n = 20", "n = 30"),
       col=c("red", "blue", "green"),
       cex=0.8,
       lwd=2,
       lty=1)
```



```r
x = b1s

plot(y=rejections_counters[[3]][[1]] / 1000,
     x=x,
     type="b",
     col="red",
     main="Power of the significance of regression test, sigma = 4",
     xlab="True value of beta_1", ylab="Frequency of rejecting the null hypothesis",
     lwd=2,
     ylim=c(0, 1.1))
lines(y=rejections_counters[[3]][[2]] / 1000,
      x=x,
      type="b",
      col="blue",
      lwd=2)
lines(y=rejections_counters[[3]][[3]] / 1000,
      x=x,
      type="b",
      col="green",
      lwd=2)
legend(-2,
       0.225,
       legend=c("n = 10", "n = 20", "n = 30"),
       col=c("red", "blue", "green"),
       cex=0.8,
       lwd=2,
       lty=1)
```



Discussion

We clearly observe that all else being equal, the sample size positively contributes to the power of the significance of regression test. Similarly, all else being equal, the amount of noise as expressed by σ negatively contributes to the power of the significance of regression test.

We also see that the closer β_1 is to 0, the smaller the chance that the null hypothesis is rejected. This is expected, since at $\beta_1 = 0$ there is no linear relationship between the predictor and the response making the null hypothesis true, meaning that us failing to reject the null hypothesis is the apt outcome.

We find it to be most curious that the absolute value of β_1 has an impact on the power of the significance of regression test. One can intuit that a more distinct linear relationship between the predictor and the response is easier to detect, and here is the demonstration of this behavior via simulation.
