DS 6030 HW01 Supervised Learning

Ben Wilson

‘09/06/2022

Contents

R Setup

```
knitr::opts_chunk$set(echo = TRUE)
data.dir = 'https://mdporter.github.io/DS6030/data/' # data directory
library(R6030)      # functions for DS-6030
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.1
## v readr   2.1.2     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(FNN)
library(ggplot2)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

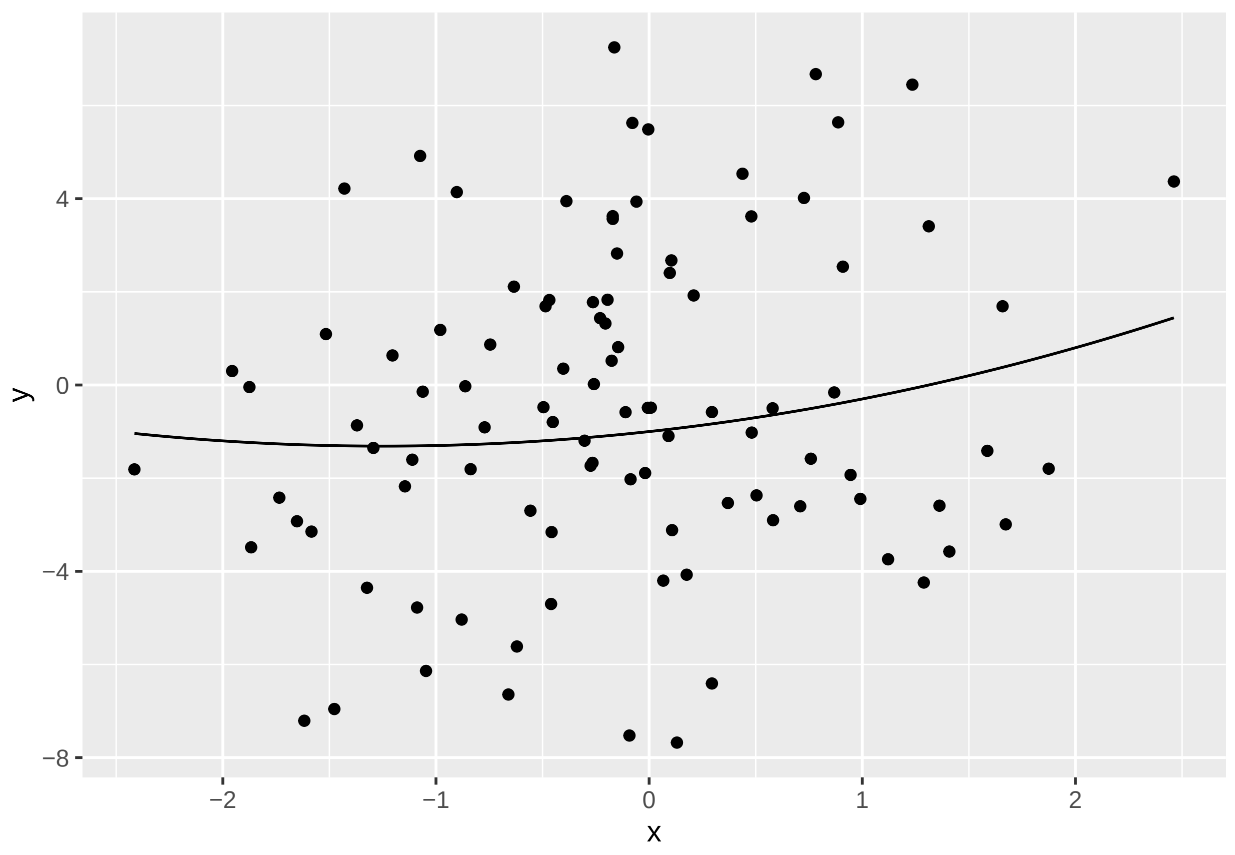
Problem 1: Evaluating a Regression Model

a. Create a set of functions to generate data from the following distributions:

```
# create functions
# create x [0,1]
sim_x <- function(n) rnorm(n)

#set function f
f <- function(x) -1 + 0.5*x + 0.2*x^2

#create y
sim_y <- function(x, sd){
  n = length(x)
  f(x) + rnorm(n, sd=sd)
}
```

b. Simulate n=100 realizations from these distributions using sd=3. Produce a scatterplot and draw the true regression line. Use set.seed(611) prior to generating the data

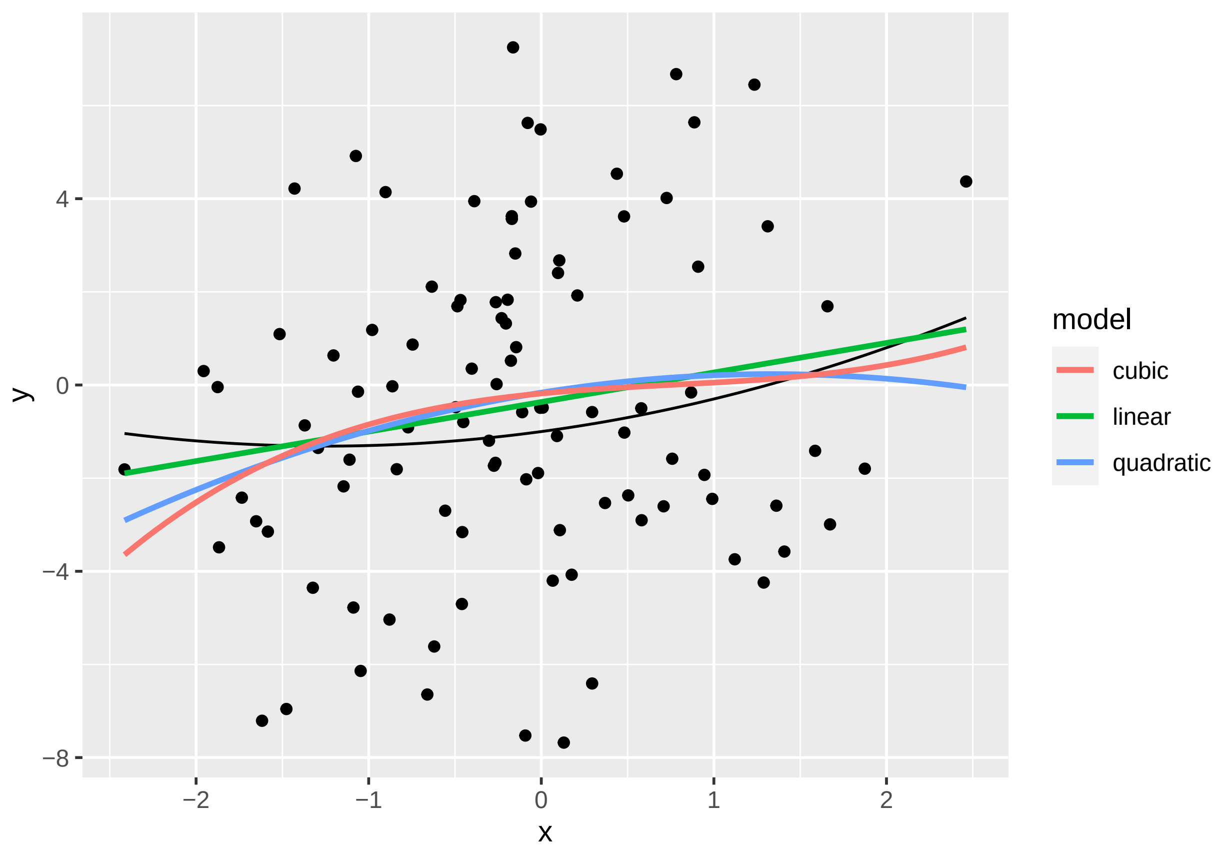
```r
#observations
n = 100
# standard deviations
sd = 3

#create data
#set seed for reproducibility
set.seed(611)
#generate x values
x = sim_x(n)
#generate y values
y = sim_y(x, sd=sd)
#training tibble (df)
data_train = tibble(x, y)

#generate scatter
gg_example = ggplot(data_train, aes(x,y)) +
  geom_point()

#generate updated scatter plot
gg_example = ggplot(data_train, aes(x,y)) +
  geom_point()+
  geom_function(fun=f)

#view scatter plot
gg_example
```



c. Fit three polynomial regression models using least squares: linear, quadratic, and cubic. Produce another scatterplot, add the fitted lines and true population line f(x) using different colors, and add a legend that maps the line color to a model. Note: The true model is quadratic, but we are also fitting linear (less complex) and cubic (more complex) models.

```
#generate linear
m1 = lm(y~x, data=data_train)

#generate quadratic line
m2 = lm(y~poly(x, degree=2), data=data_train)

#generate cubic line
m3 = lm(y~poly(x, degree=3), data=data_train)

#generate updated scatter plot
gg_example +
  geom_smooth(method="lm", se=FALSE, aes(color="linear")) +
  geom_smooth(method="lm", formula="y~poly(x,2)", se=FALSE, aes(color="quadratic")) +
  geom_smooth(method="lm", formula="y~poly(x,3)", se=FALSE, aes(color="cubic")) +
  scale_color_discrete(name="model")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



d. Simulate a test data set of 10,000 observations from the same distributions. Use set.seed(612) prior to generating the test data. Calculate the estimated mean squared error (MSE) for each model. Are the results as expected?

```
#observations
n_test = 10000
# standard deviations
sd = 3

#set seed for reproducibility
set.seed(612)

#tible test data
x_test = sim_x(n_test)
y_test = sim_y(x_test, sd=sd)
data_test = tibble(x = x_test, y = y_test)

# generating yhat/predictions for each model
yhat1 = predict(m1, newdata=tibble(x = x_test))
yhat2 = predict(m2, newdata=tibble(x = x_test))
yhat3 = predict(m3, newdata=tibble(x = x_test))

# calculating MSEs
mse_m1 = mean((y_test-yhat1)^2)
```

```r
mse_m2 = mean((y_test-yhat2)^2)
mse_m3 = mean((y_test-yhat3)^2)
```

e. What is the best achievable MSE? That is, what is the MSE if the true f(x) was used to evaluate the test set? How close does the best method come to achieving the optimum?

```r
#create true form assessment
MSE_true <- f(x_test)
MSE_result <- mean((data_test$y - MSE_true)^2)
MSE_result
```

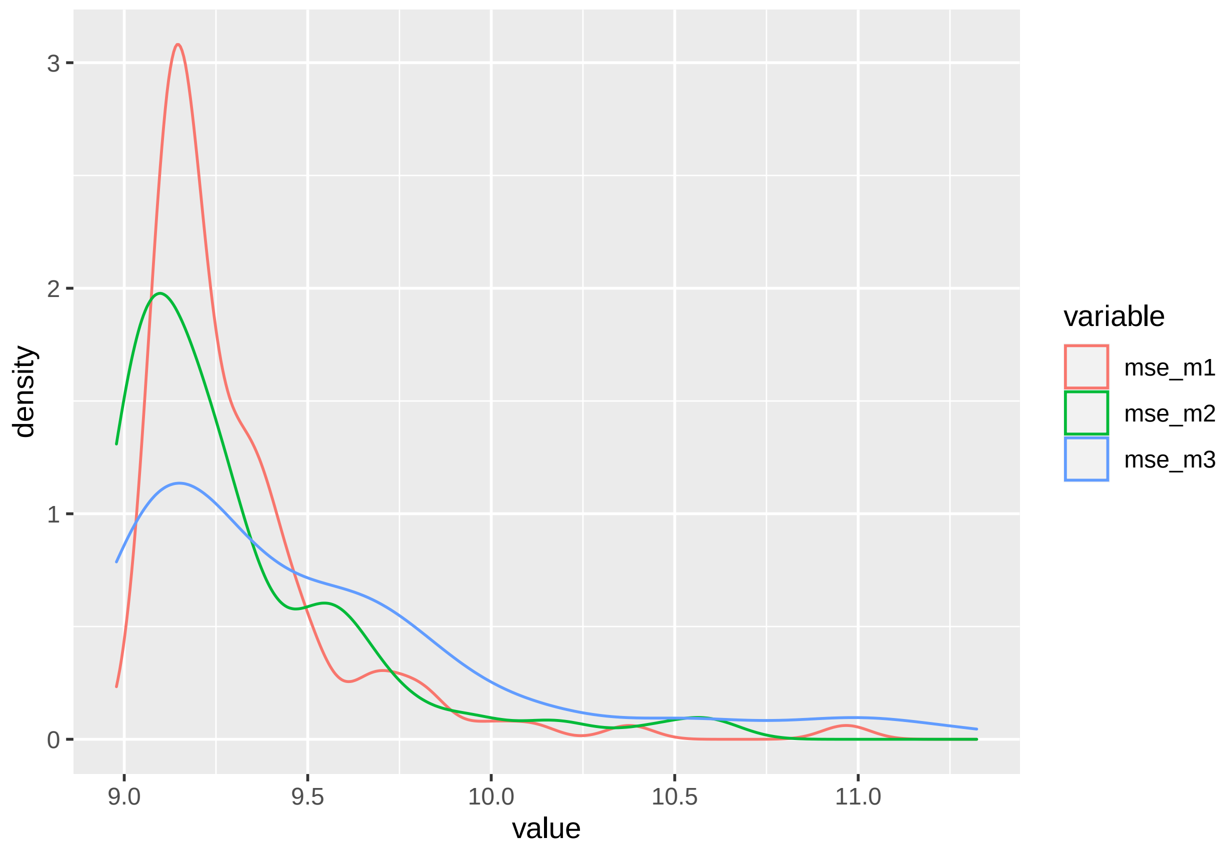
```
## [1] 8.972119
```

f. The MSE scores obtained in part d came from one realization of training data. Here will we explore how much variation there is in the MSE scores by replicating the simulation many times. Re-run parts b. and c. (i.e., generate training data and fit models) 100 times. Calculate the MSE for all simulations. Create kernel density or histogram plots of the resulting MSE values for each model. Use set.seed(613) prior to running the simulation and do not set the seed in any other places. Use the same test data from part d. (This question is only about the variability that comes from the training data).

```r
set.seed(613)

# define vectors to capture results
mse_m1 <- NULL
mse_m2 <- NULL
mse_m3 <- NULL

n_test = 10000
n <- 100
sd <- 3
for (i in 1:100) {
  #tible test data
  x = sim_x(n)
  y = sim_y(x, sd=sd)
  data_train = tibble(x = x, y = y)

  #generate linear
  m1 = lm(y~x, data=data_train)
  m2 = lm(y~poly(x, degree=2), data=data_train)
  m3 = lm(y~poly(x, degree=3), data=data_train)

  # generating yhat/predictions for each model
  yhat1 = predict(m1, newdata=tibble(x = x_test))
  yhat2 = predict(m2, newdata=tibble(x = x_test))
  yhat3 = predict(m3, newdata=tibble(x = x_test))

  # calculating MSEs
  mse_m1[i] = mean((y_test-yhat1)^2)
  mse_m2[i] = mean((y_test-yhat2)^2)
  mse_m3[i] = mean((y_test-yhat3)^2)
}

mse_mat <- data.frame(mse_m1, mse_m2, mse_m3)

m_mses.plot <- melt(mse_mat)
```

```
## No id variables; using all as measure variables
```

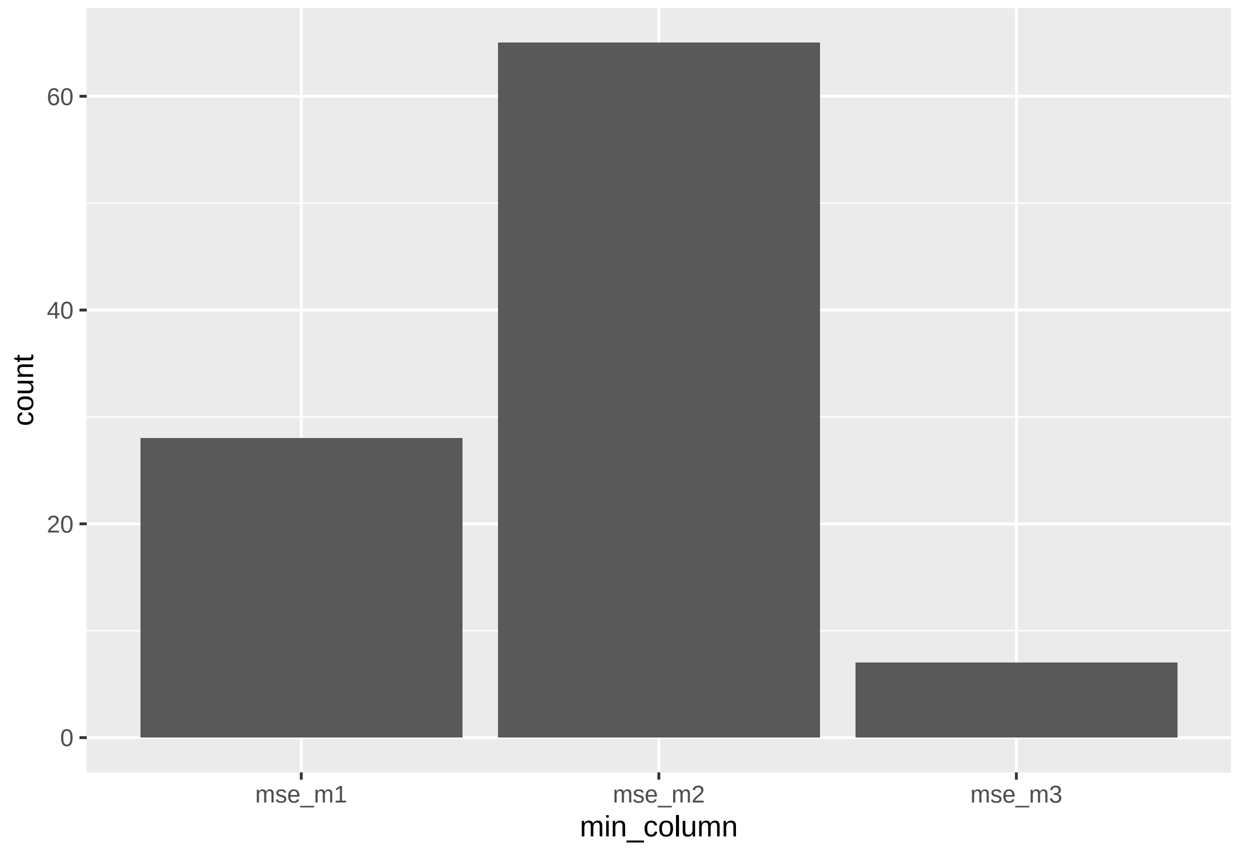
```
ggplot(aes(x=value, colour = variable), data = m_mses.plot)+
  geom_density()
```



g. Show a count of how many times each model was the best. That is, out of the 100 simulations, count how many times each model had the lowest MSE.

```
#identify min row of 3 columns
mse_mat$min_column <- colnames(mse_mat)[apply(mse_mat, 1, which.min)]
```

```
#summarize results of min
ggplot(mse_mat, aes(x = min_column))+
  geom_bar()
```



h. Write a function that implements the simulation in part f. The function should have arguments for i) the size of the training data n, and ii) the standard deviation of the random error sd. Use the same set.seed(613).

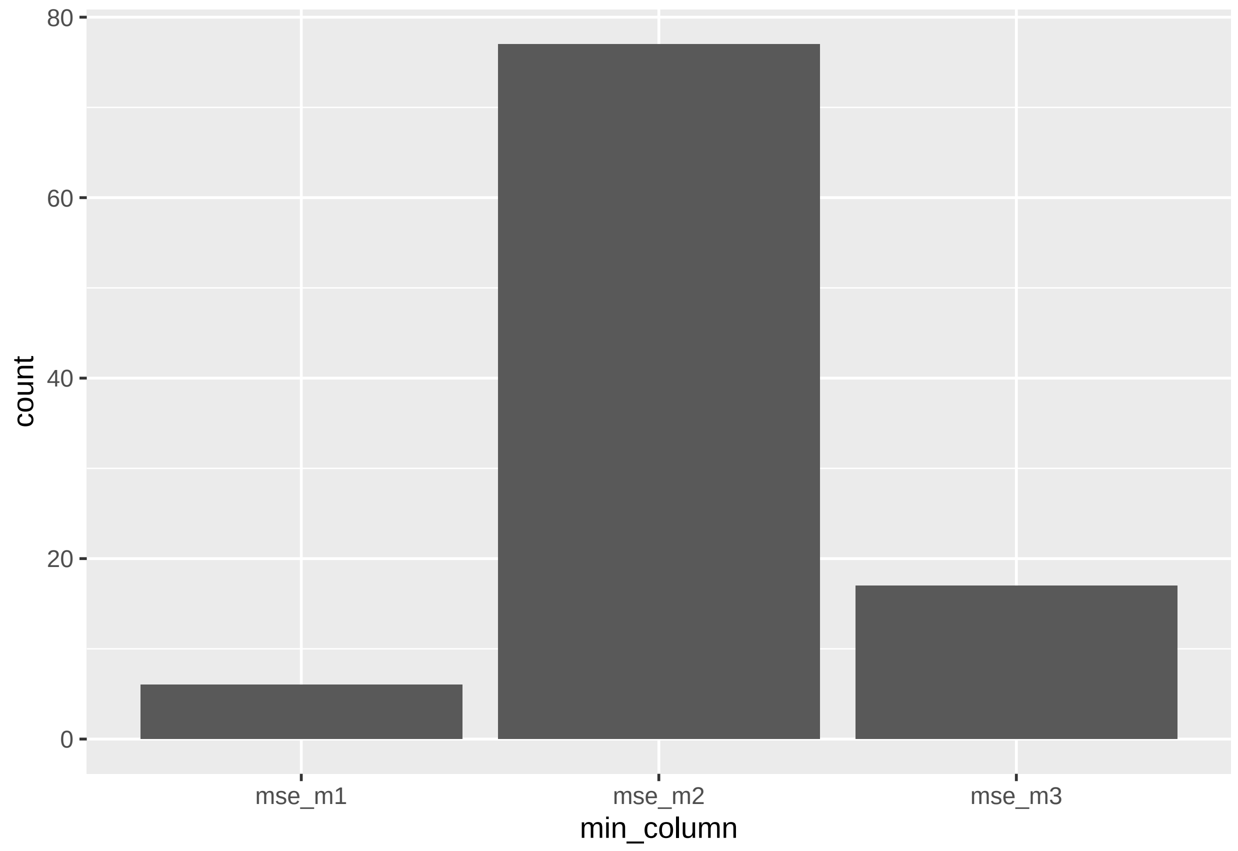
```
#set seed
set.seed(613)

#create function

sim_report <- function(n_train_input,sd_input) {

  m1_mses <- NULL
  m2_mses <- NULL
  m3_mses <- NULL

  n <- n_train_input
  sd <- sd_input
  for (i in 1:n_train_input) {
    #tible test data
    x = sim_x(n)
    y = sim_y(x, sd=sd)
    data_train = tibble(x,y)

    #generate linear
    m1 = lm(y~x, data=data_train)
    m2 = lm(y~poly(x, degree=2), data=data_train)
    m3 = lm(y~poly(x, degree=3), data=data_train)

    # generating yhat/predictions for each model
    yhat1 = predict(m1, newdata=tibble(x = x_test))
    yhat2 = predict(m2, newdata=tibble(x = x_test))
    yhat3 = predict(m3, newdata=tibble(x = x_test))

    # calculating MSEs
    mse_m1[i] = mean((y_test-yhat1)^2)
    mse_m2[i] = mean((y_test-yhat2)^2)
    mse_m3[i] = mean((y_test-yhat3)^2)
  }

  mse_mat <- tibble(mse_m1,mse_m2,mse_m3)
  return(mse_mat)

}

#simulate function
simulate_Result_100 <- sim_report(n_train_input = 100,sd_input = 2)
print(simulate_Result_100)
```

```
## # A tibble: 100 x 3
##    mse_m1 mse_m2 mse_m3
##     <dbl>  <dbl>  <dbl>
##  1   9.28   9.06   9.13
##  2   9.16   9.15   9.28
##  3   9.14   9.12   9.11
##  4   9.71   9.54   9.47
##  5   9.12   9.03   9.13
##  6   9.12   9.00   9.00
##  7   9.12   9.08   9.36
##  8   9.12   9.03   9.05
##  9   9.18   9.21   9.22
## 10   9.12   8.99   9.00
## # ... with 90 more rows
```

i. Use your function to repeat the simulation in part f, but use sd=2. Report the number of times each model was best (you do not need to produce any plots).

```
#resimulate results with 100n, 2sd
simulate_Result_100 <- sim_report(100,2)
```

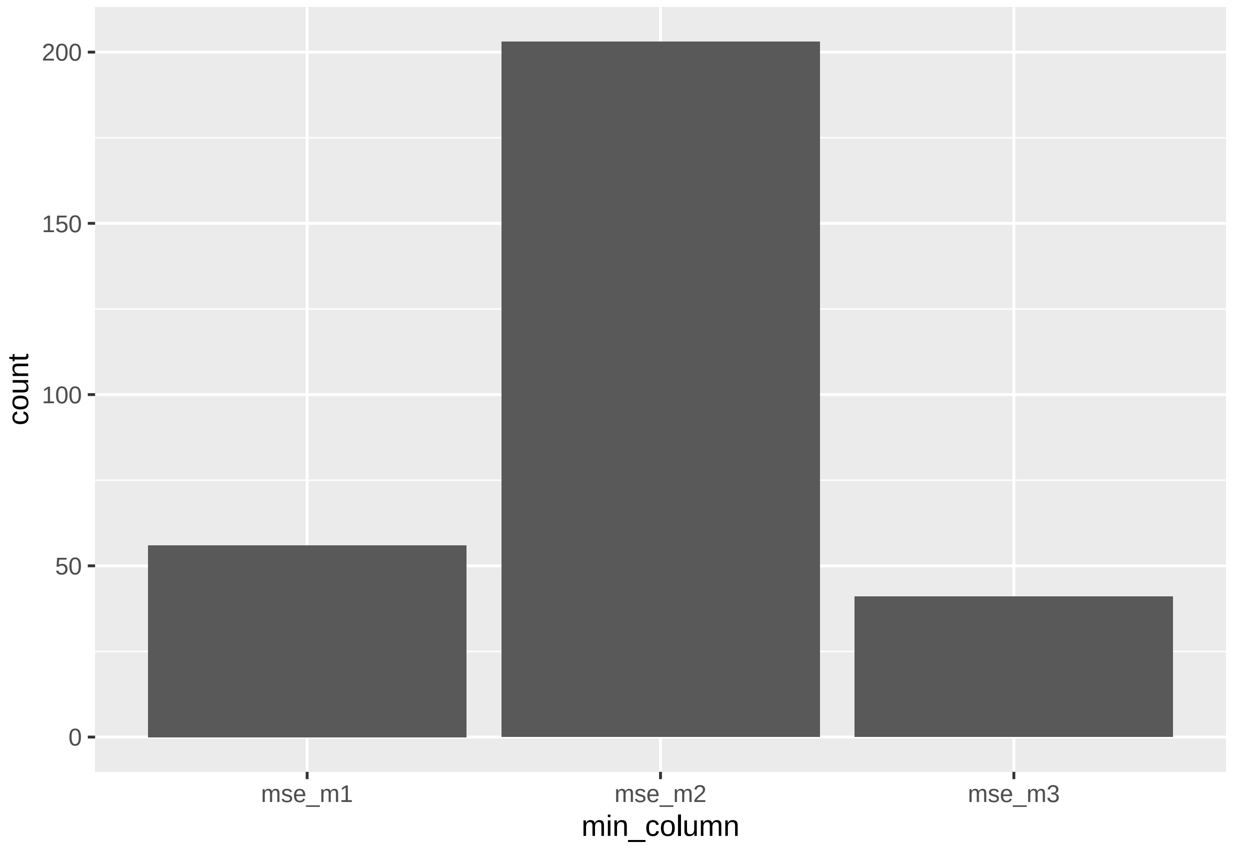
```
#id min row of 3 columns
simulate_Result_100$min_column <- colnames(simulate_Result_100)[apply(simulate_Result_100, 1, which.min

#summarize results of min
ggplot(simulate_Result_100, aes(x = min_column))+
  geom_bar()
```



```
## j. Repeat i, but now use sd=4 and n=300.

#resimulate results with 300n, 4sd
simulate_Result_300 <- sim_report(300,4)
print(simulate_Result_300)
```

```
## # A tibble: 300 x 3
##    mse_m1 mse_m2 mse_m3
##     <dbl>  <dbl>  <dbl>
##  1   9.25   9.11   9.13
##  2   9.16   9.23   9.21
##  3   9.12   9.05   9.18
##  4   9.10   8.99   8.99
##  5   9.17   9.07   9.14
##  6   9.13   9.01   9.01
##  7   9.17   9.05   9.08
##  8   9.17   9.17   9.37
##  9   9.15   9.04   9.04
## 10   9.14   9.00   9.14
## # ... with 290 more rows
```

```
#identify min row of 3 columns
simulate_Result_300$min_column <- colnames(simulate_Result_300)[apply(simulate_Result_300, 1, which.min

#summarize results of min
ggplot(simulate_Result_300, aes(x = min_column))+
  geom_bar()
```



k. Describe the effects sd and n has on selection of the best model? Why is the true model form (i.e., quadratic) not always the best model to use when prediction is the goal?

The extremes between MSES become more evident, with quadratic model having more of the lowest MSES results. The true model may not be the best since it will overfit and the outlier data points will then fail to fall close to the true line.