Lab 9

Janine Lim

11:59PM May 10, 2021

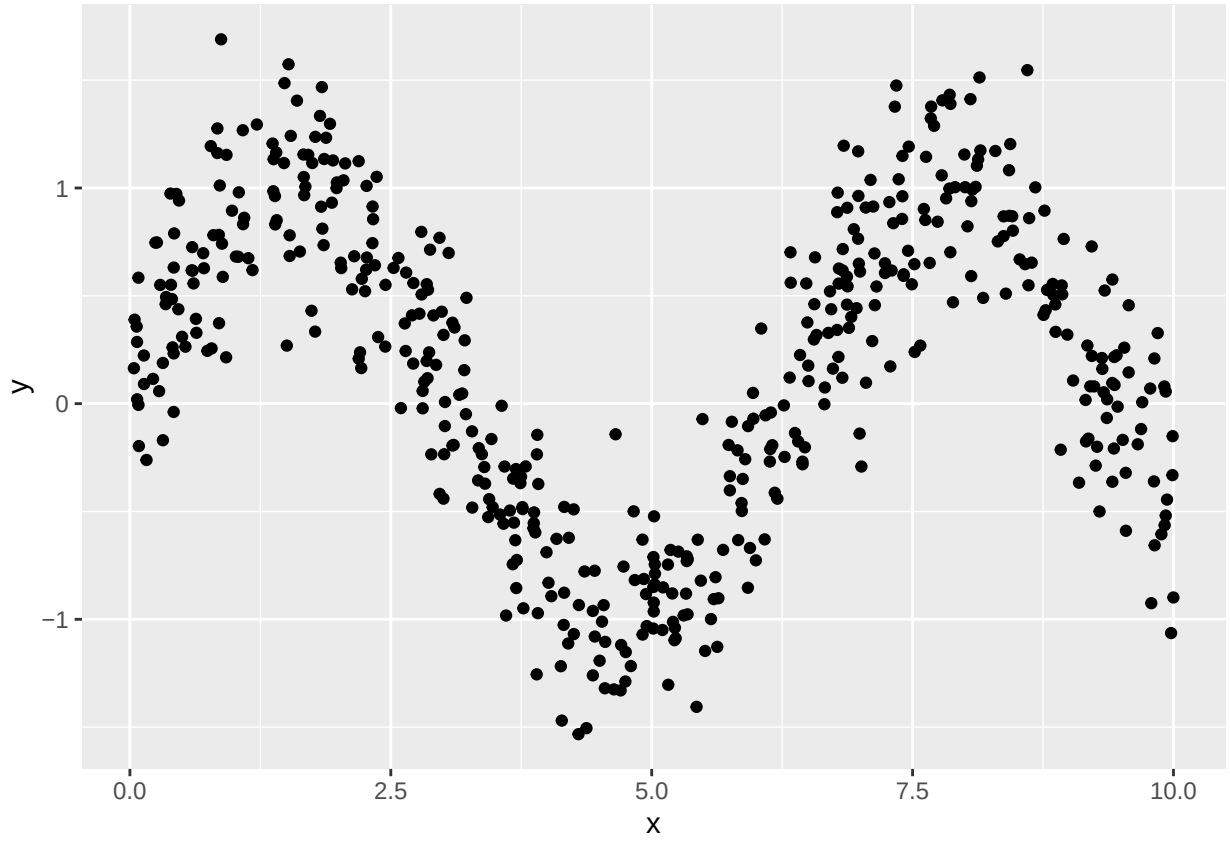
Here we will learn about trees, bagged trees and random forests. You can use the `YARF` package if it works, otherwise, use the `randomForest` package (the standard).

Let's take a look at the simulated sine curve data from practice lecture 12. Below is the code for the data generating process:

```
rm(list = ls())
n = 500
sigma = 0.3
x_min = 0
x_max = 10
f_x = function(x){sin(x)}
y_x = function(x, sigma){f_x(x) + rnorm(n, 0, sigma)}
x_train = runif(n, x_min, x_max)
y_train = y_x(x_train, sigma)
```

Plot an example dataset of size 500:

```
pacman::p_load(ggplot2)
ggplot(data.frame(x=x_train, y = y_train)) +
  geom_point(aes(x=x, y=y))
```

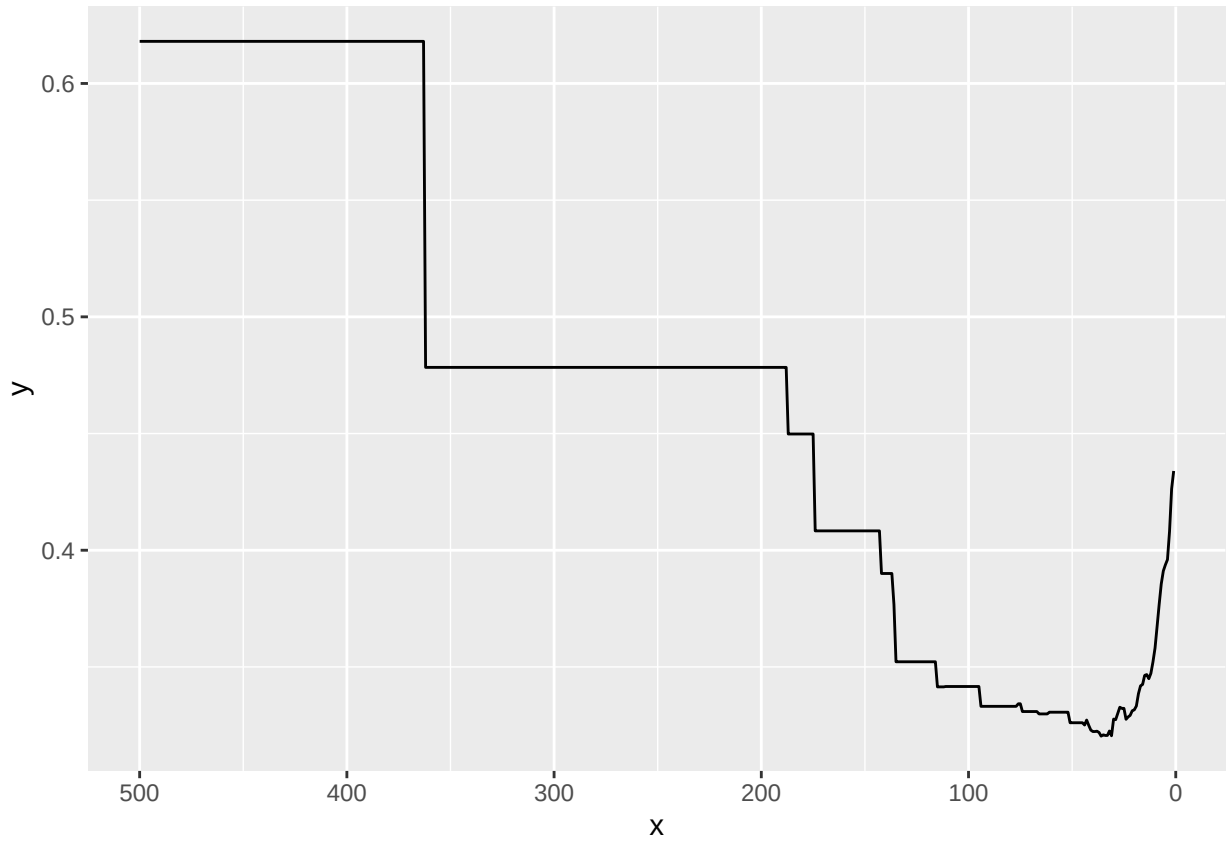


Create a test set of size 500 as well

```
x_test = runif(n, x_min, x_max)
y_test = y_x(x_test, sigma)
```

Locate the optimal node size hyperparameter for the regression tree model. I believe you can use `randomForest` here by setting `ntree = 1`, `replace = FALSE`, `sampsize = n` (`mtry` is already set to be 1 because there is only one feature) and then you can set `nodesize`. Plot oos SE by node size.

```
pacman::p_load(randomForest)

node_sizes = 1:n
se_by_node_sizes = array(NA, dim = length(node_sizes))
for (i in 1:length(node_sizes)){
  rf_mod = randomForest(x=data.frame(x=x_train), y =y_train, ntree = 1, replace = FALSE, sampsize = n, n
  yhat_test = predict(rf_mod, data.frame(x=x_test))
  se_by_node_sizes[i] = sd(y_test - yhat_test)
}

ggplot(data.frame(x=node_sizes, y = se_by_node_sizes)) +
  geom_line(aes(x=x, y=y)) +
  scale_x_reverse()
```

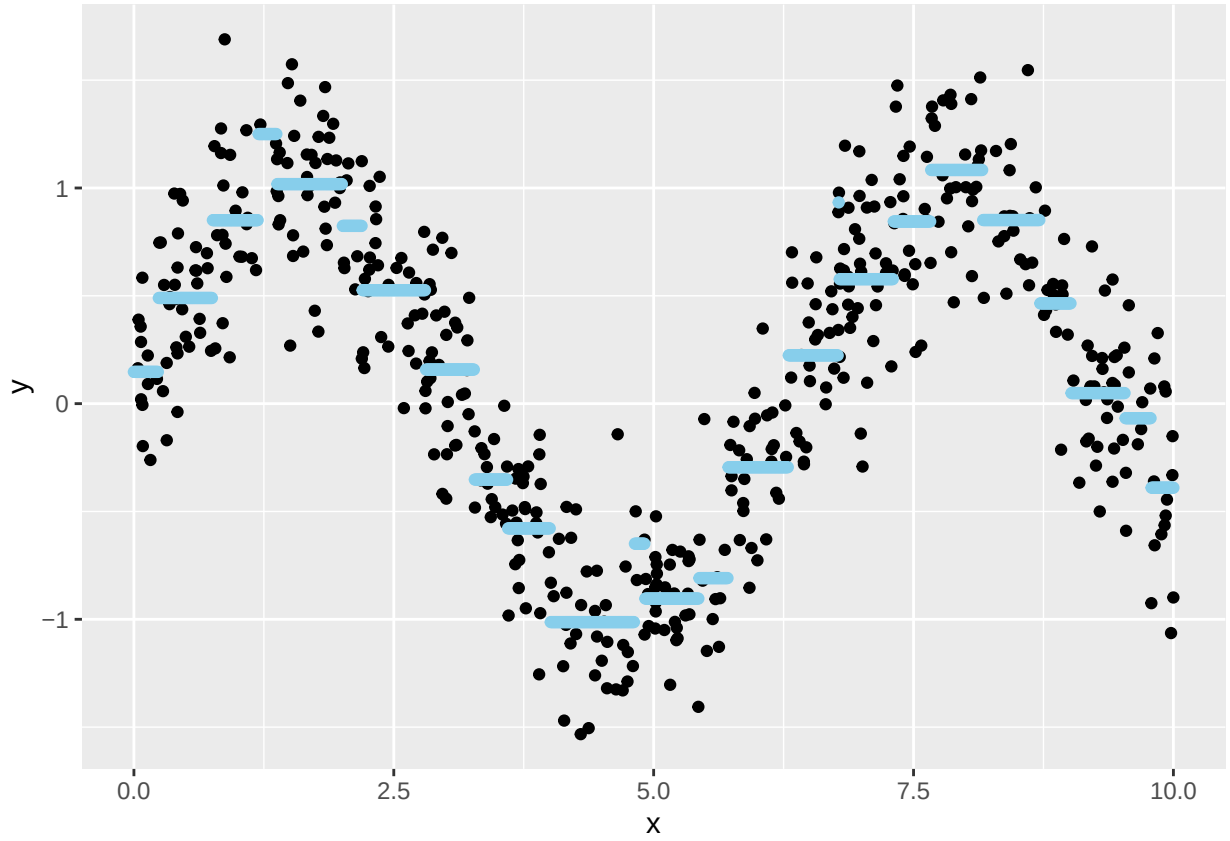



```
which.min(se_by_node_sizes)
```

```
## [1] 36
```

Plot the regression tree model with the optimal node size.

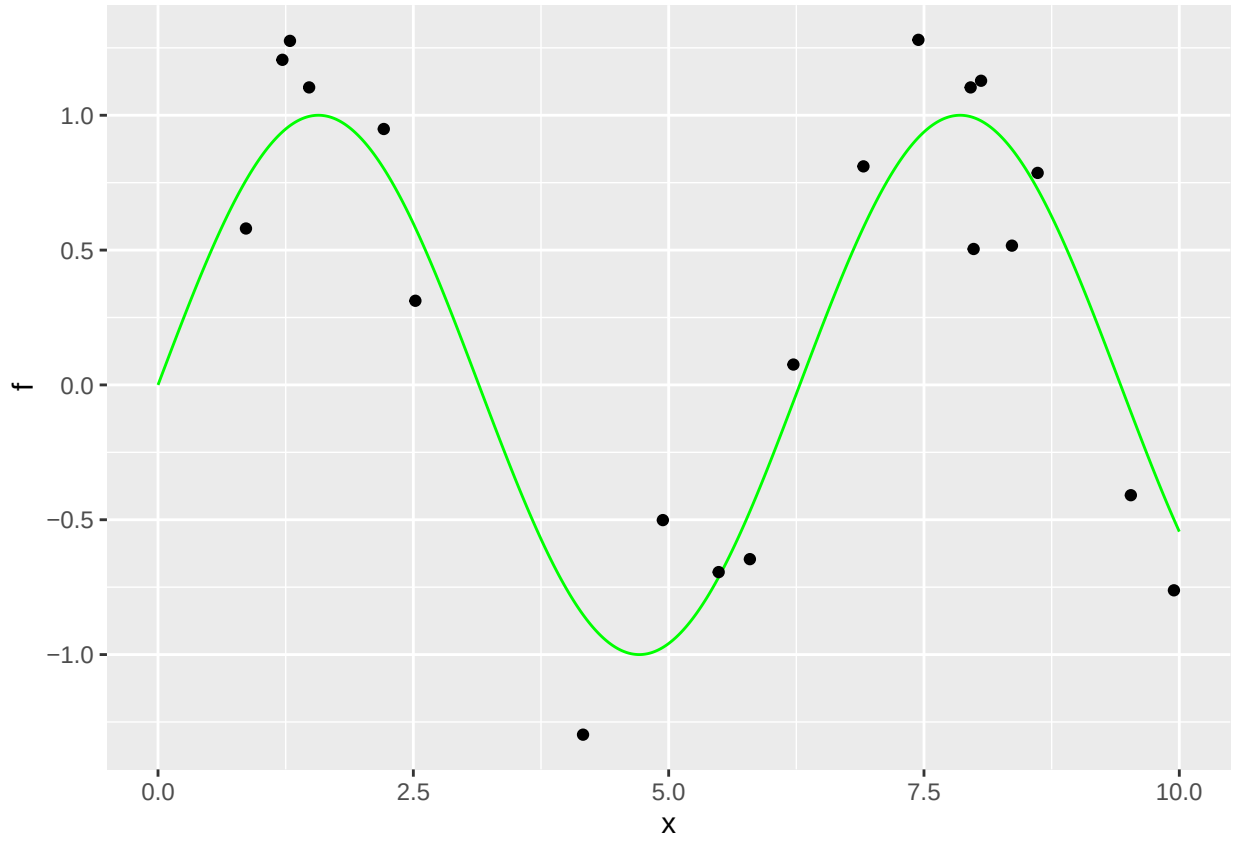
```
rf_mod = randomForest(x=data.frame(x=x_train), y =y_train, ntree = 1, replace = FALSE, sampsize = n, no
resolution = 0.01
x_grid = seq(from = x_min, to = x_max, by = resolution)
g_x = predict(rf_mod, data.frame(x=x_grid))
ggplot(data.frame(x=x_grid, y = g_x))+
  aes(x=x, y=y) +
  geom_point(data = data.frame(x=x_train, y = y_train) ) +
     geom_point(col = "skyblue")
```



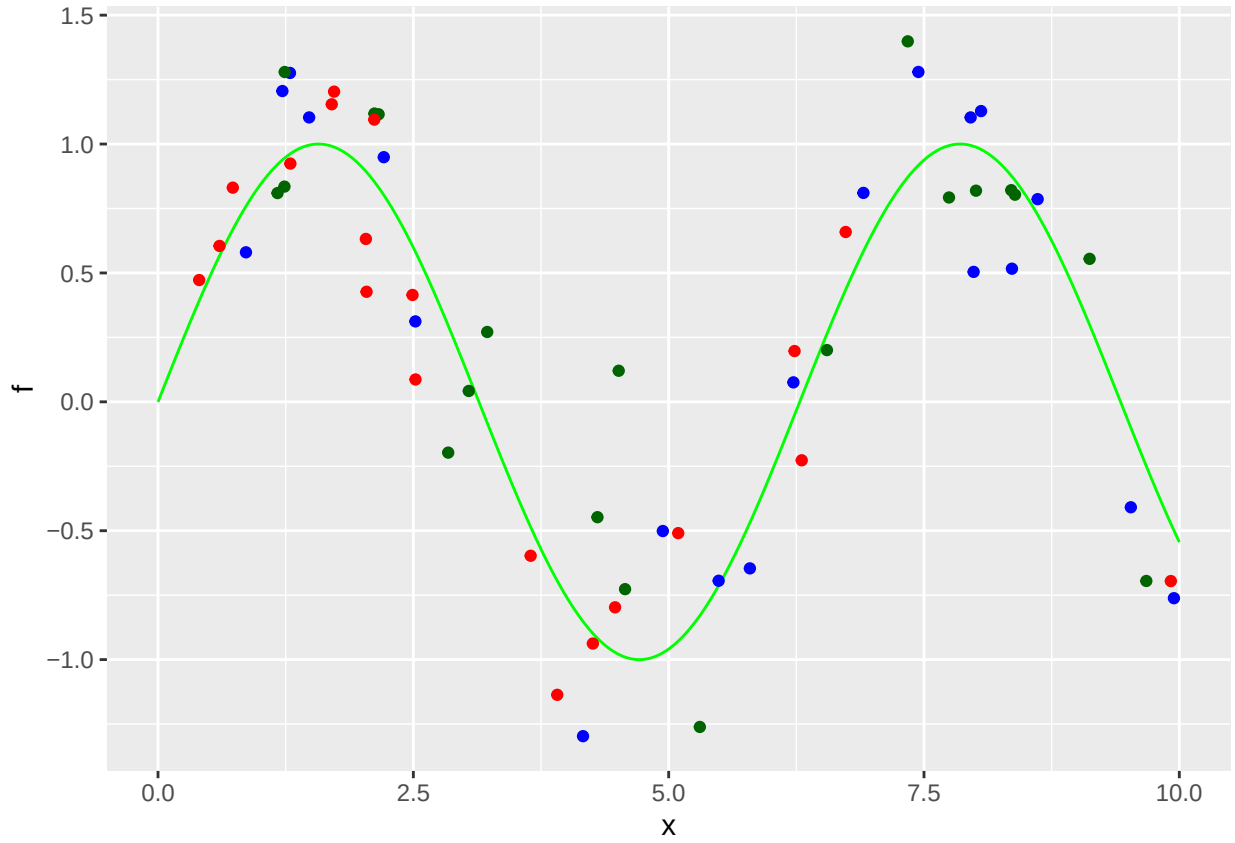
Provide the bias-variance decomposition of this DGP fit with this model. It is a lot of code, but it is in the practice lectures. If your three numbers don't add up within two significant digits, increase your resolution.

```
n=20
xmin = 0
xmax = 10
n_train = 20
n_test = 1000
sigma = 0.3
f = function(x){sin(x)}
Nsim = 1000


training_gs = matrix(NA, nrow = Nsim, ncol = 2)
x_trains = matrix(NA, nrow = Nsim, ncol = n_train)
y_trains = matrix(NA, nrow = Nsim, ncol = n_train)
all_oos_residuals = matrix(NA, nrow = Nsim, ncol = n_test)
for (nsim in 1 : Nsim){
  #simulate dataset $\mathbb{D}$
  x_train = runif(n_train, x_min, x_max)
  delta_train = rnorm(n_train, 0, sigma) #Assumption I: mean zero and Assumption II: homoskedastic
  y_train = f_x(x_train) + delta_train
  x_trains[nsim, ] = x_train
  y_trains[nsim, ] = y_train

  #fit a model g | x's, delta's and save it
```

```
  g_model = lm(y_train ~ ., data.frame(x = x_train))
  training_gs[nsim, ] = coef(g_model)

  #generate oos dataset according to the same data generating process (DGP)
  x_test = runif(n_test, x_min, x_max)
  delta_test = rnorm(n_test, 0, sigma)
  y_test = f_x(x_test) + delta_test
  #predict oos using the model and save the oos residuals
  y_hat_test = predict(g_model, data.frame(x = x_test))
  all_oos_residuals[nsim, ] = y_test - y_hat_test
}

pacman::p_load(ggplot2)
resolution = 10000
x = seq(x_min, x_max, length.out = resolution)
f_x_df = data.frame(x = x, f = f_x(x))
ggplot(f_x_df, aes(x, f)) +
  geom_line(col = "green") +
  geom_point(aes(x, y), data = data.frame(x = x_trains[1, ], y = y_trains[1, ]))
```



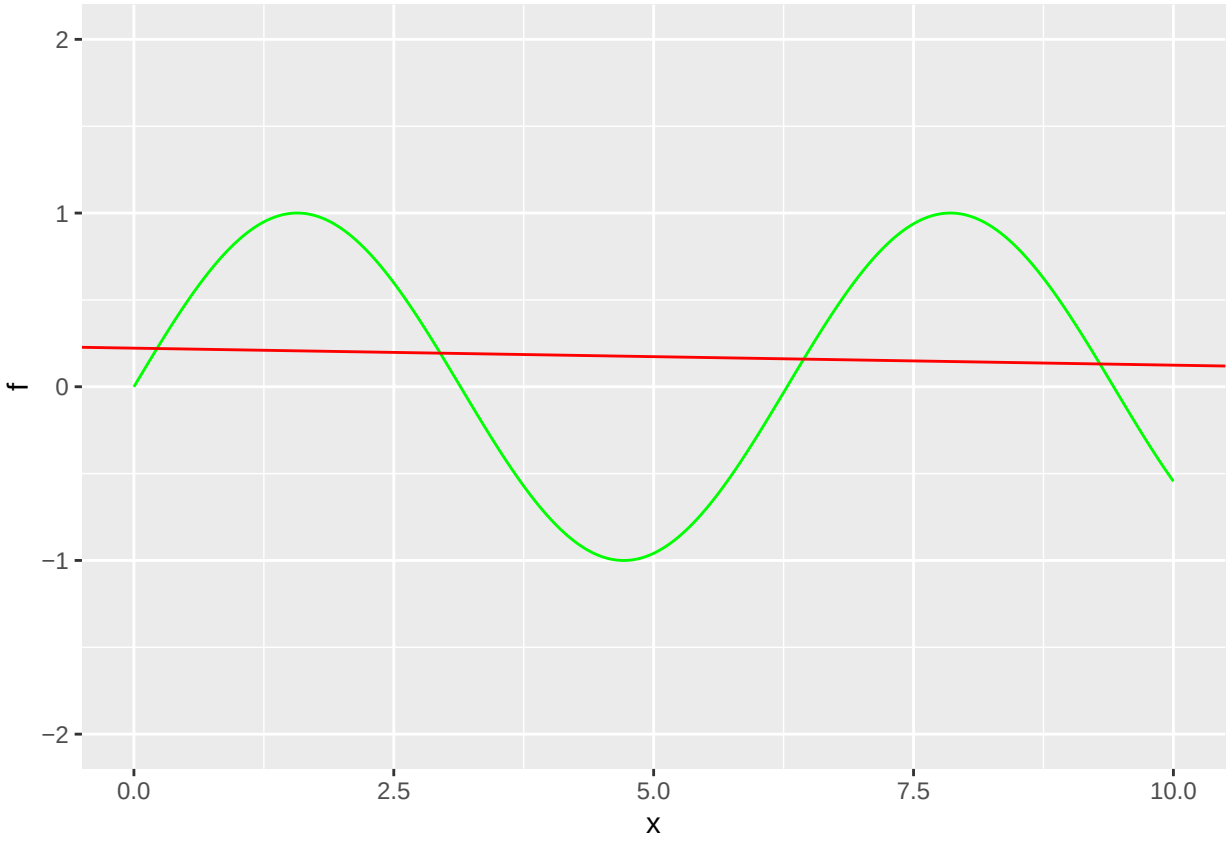

```
ggplot(f_x_df, aes(x, f)) +
  geom_line(col = "green") +
  geom_point(aes(x, y), data = data.frame(x = x_trains[1, ], y = y_trains[1, ]), col = "blue") +
  geom_point(aes(x, y), data = data.frame(x = x_trains[2, ], y = y_trains[2, ]), col = "darkgreen") +
  geom_point(aes(x, y), data = data.frame(x = x_trains[3, ], y = y_trains[3, ]), col = "red")
```




```
mse = mean(c(all_oos_residuals)^2)
mse
```

```
## [1] 0.5842197
```

```
g_average = colMeans(training_gs)
ggplot(f_x_df, aes(x, f)) +
  geom_line(col = "green") +
  geom_abline(intercept = g_average[1], slope = g_average[2], col = "red") +
  ylim(-2, 2)
```

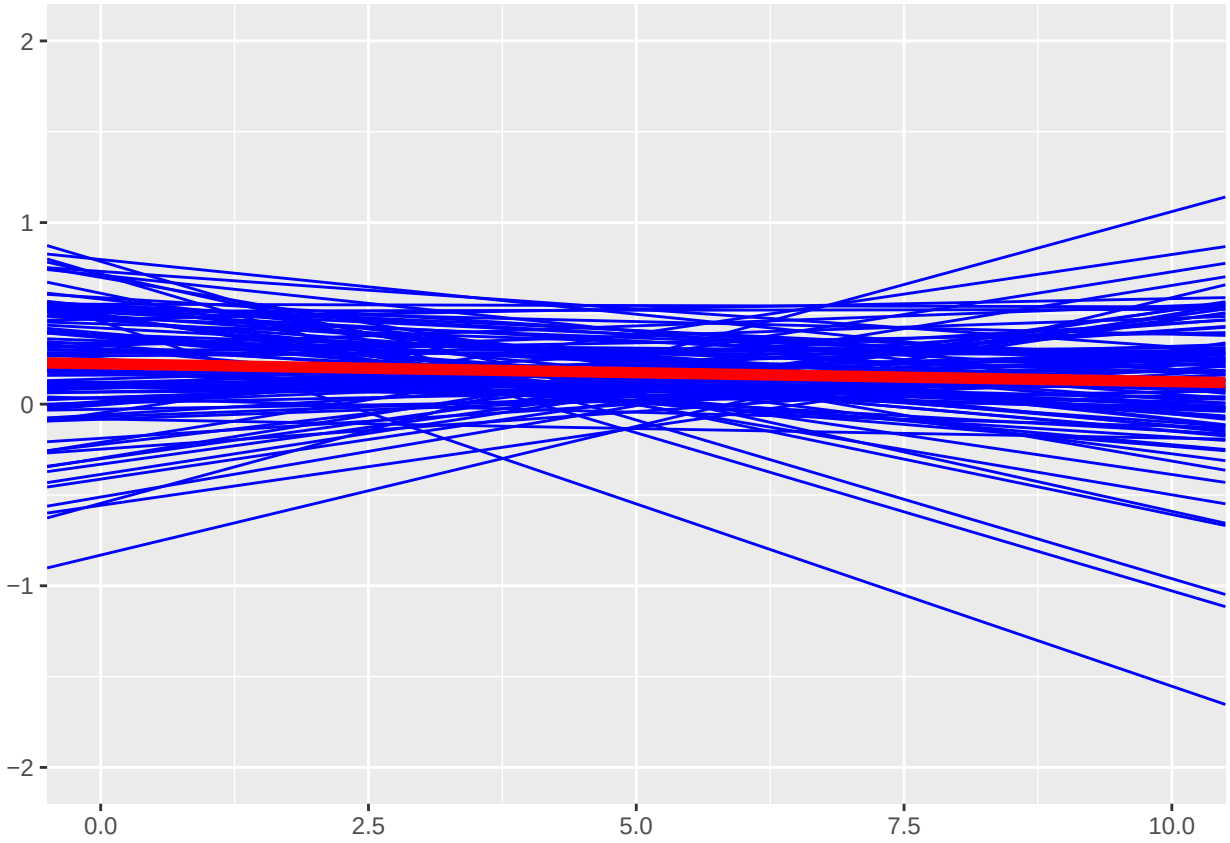



```
x = seq(x_min, x_max, length.out = resolution)
g_avg_x = g_average[1] + g_average[2] * x
f = sin(x)
biases = f - g_avg_x
expe_bias_g_sq = mean(biases^2)
expe_bias_g_sq
```

```
## [1] 0.4416051
```

```
plot_obj = ggplot() +
  xlim(x_min, x_max) + ylim(x_min^2, x_max^2)
for (nsim in 1 : min(Nsim, 100)){ #otherwise takes too long
  plot_obj = plot_obj + geom_abline(intercept = training_gs[nsim, 1], slope = training_gs[nsim, 2], col
}
plot_obj +
  geom_abline(intercept = g_average[1], slope = g_average[2], col = "red", lwd = 2) +
  ylim(-2,2)
```

```
## Scale for 'y' is already present. Adding another scale for 'y', which will
## replace the existing scale.
```




```
  # geom_line(data = f_x_df, aes(x, f), col = "green", size = 1)

x = seq(x_min, x_max, length.out = resolution)
expe_g_x = g_average[1] + g_average[2] * x
var_x_s = array(NA, Nsim)
for (nsim in 1 : Nsim){
  g_x = training_gs[nsim, 1] + training_gs[nsim, 2] * x
  var_x_s[nsim] = mean((g_x - expe_g_x)^2)
}
expe_var_g = mean(var_x_s)
expe_var_g
```

```
## [1] 0.0521022
```

```
mse
```

```
## [1] 0.5842197
```

```
sigma^2
```

```
## [1] 0.09
```

```
expe_bias_g_sq
```

```
## [1] 0.4416051
```

```
expe_var_g
```

```
## [1] 0.0521022
```

```
sigma^2 + expe_bias_g_sq + expe_var_g
```

```
## [1] 0.5837073
```

```
rm(list = ls())
```

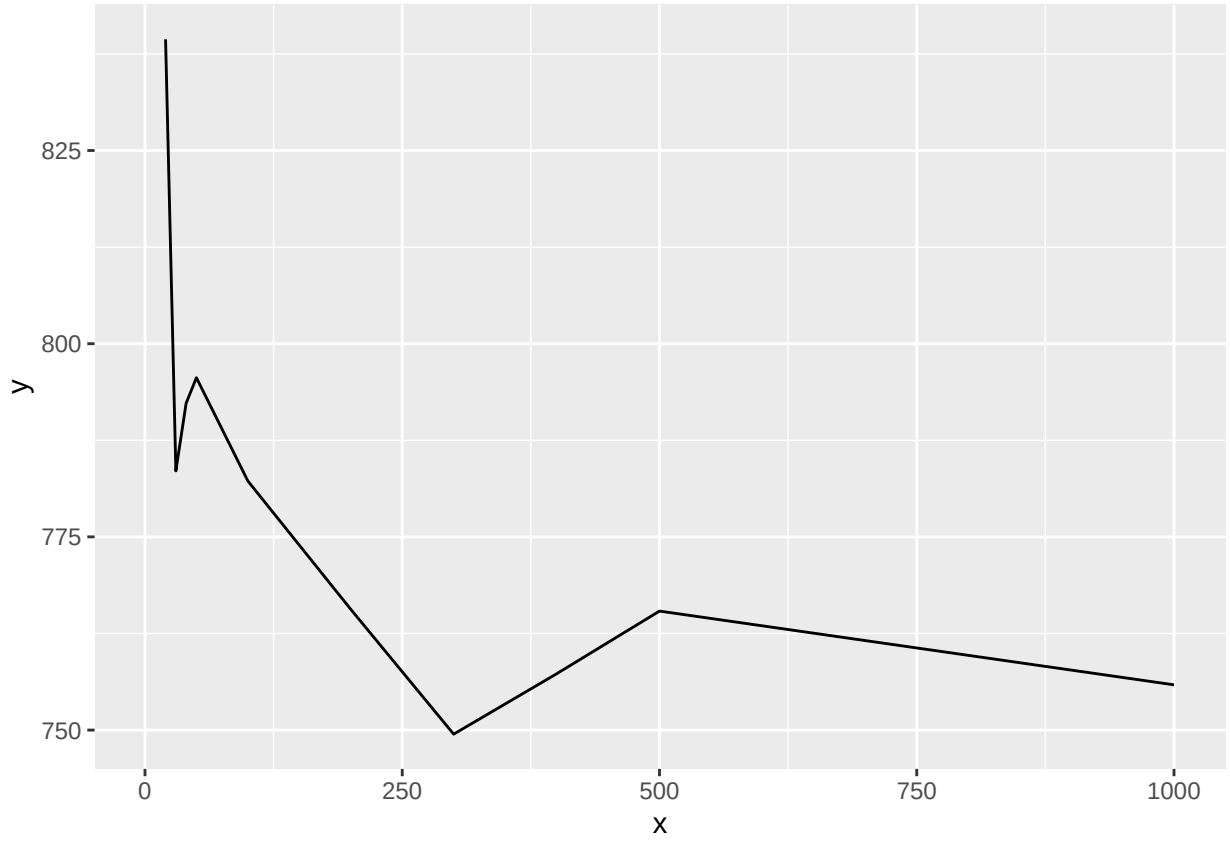
Take a sample of n = 2000 observations from the diamonds data.

```
n=2000
pacman::p_load(dplyr)

diamonds_samp = diamonds %>%
                  sample_n(n)
```

find the bootstrap s_e for a RF model using 1, 2, 5, 10, 20, 30, 40, 50, 100, 200, 300, 400, 500, 1000 trees. If you are using the `randomForest` package, you can calculate oob residuals via `e_oob = y_train - rf_mod$predicted`. Plot.

```
num_trees = c(1, 2, 5, 10, 20, 30, 40, 50, 100, 200, 300, 400, 500, 1000)
oob_se_by_num_trees = array(NA, length(num_trees))

for (i in 1:length(num_trees)){
  rf_mod = randomForest(price~., data = diamonds_samp, ntree = num_trees[i] )
  oob_se_by_num_trees[i] = sd(diamonds_samp$price - rf_mod$predicted)
}

ggplot(data.frame(x=num_trees, y = oob_se_by_num_trees)) +
  geom_line(aes(x=x, y=y))
```

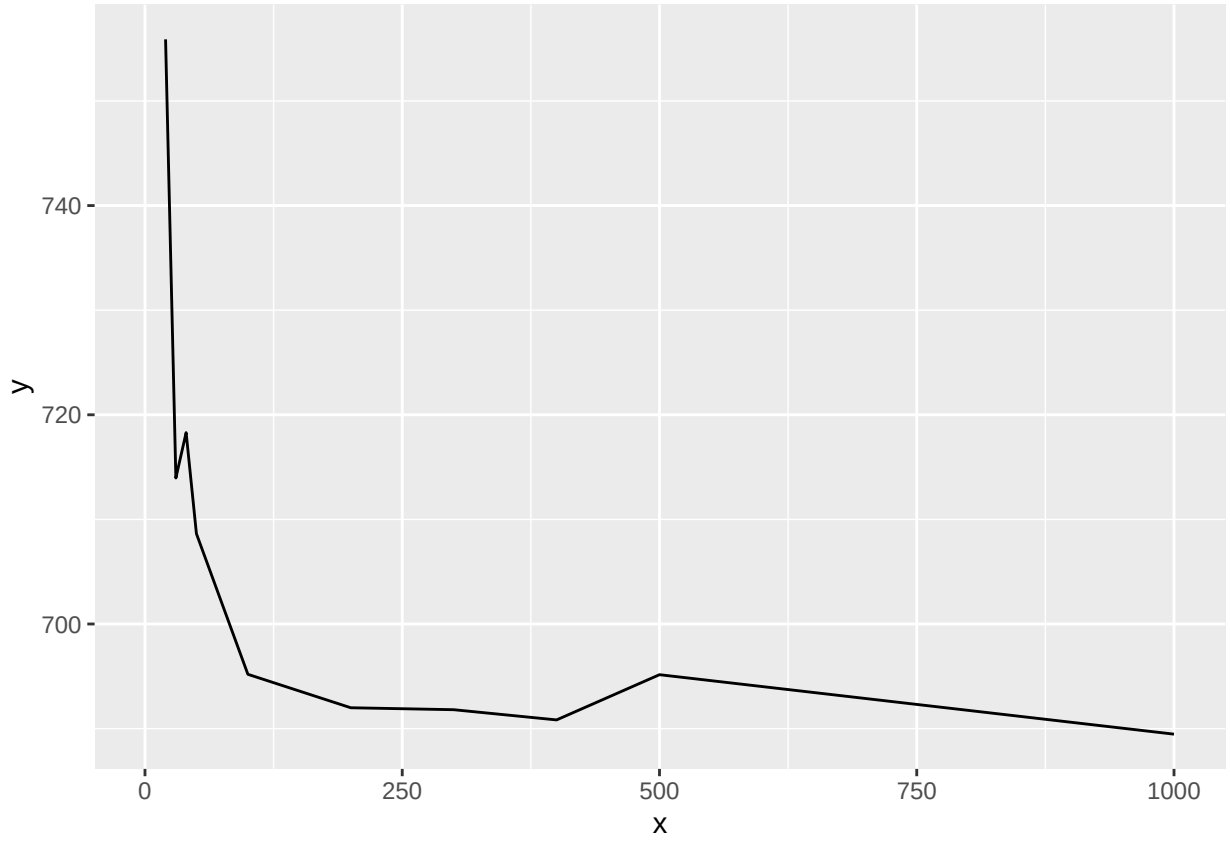
```
## Warning: Removed 4 row(s) containing missing values (geom_path).
```



Using the diamonds data, find the oob s_e for a bagged-tree model using 1, 2, 5, 10, 20, 30, 40, 50, 100, 200, 300, 400, 500, 1000 trees. If you are using the `randomForest` package, you can create the bagged tree model via setting an argument within the RF constructor function.

```
num_trees = c(1, 2, 5, 10, 20, 30, 40, 50, 100, 200, 300, 400, 500, 1000)
oob_se_by_num_trees_bag = array(NA, length(num_trees))

for (i in 1:length(num_trees)){
  rf_mod = randomForest(price~., data = diamonds_samp, ntree = num_trees[i], mtry = ncol(diamonds_samp)-
  oob_se_by_num_trees_bag[i] = sd(diamonds_samp$price - rf_mod$predicted)
}

ggplot(data.frame(x=num_trees, y = oob_se_by_num_trees_bag)) +
  geom_line(aes(x=x, y=y))
```

```
## Warning: Removed 4 row(s) containing missing values (geom_path).
```



What is the percentage gain / loss in performance of the RF model vs bagged trees model?

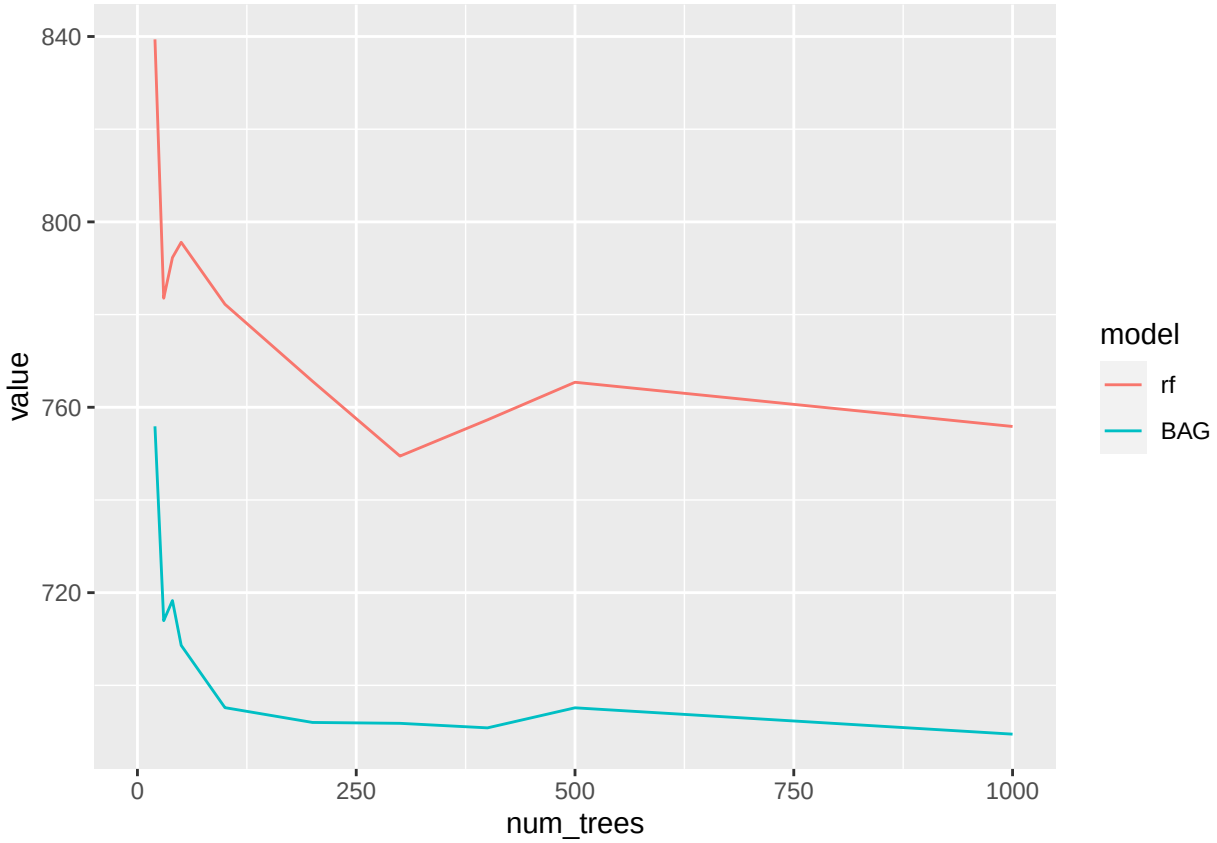
```
(oob_se_by_num_trees - oob_se_by_num_trees_bag ) / oob_se_by_num_trees_bag *100
```

```
##  [1]        NA        NA        NA        NA 11.046003  9.745314 10.302022
##  [8] 12.272495 12.519713 10.640626  8.335139  9.619090 10.104609  9.628684
```

Plot bootstrap s_e by number of trees for both RF and bagged trees.

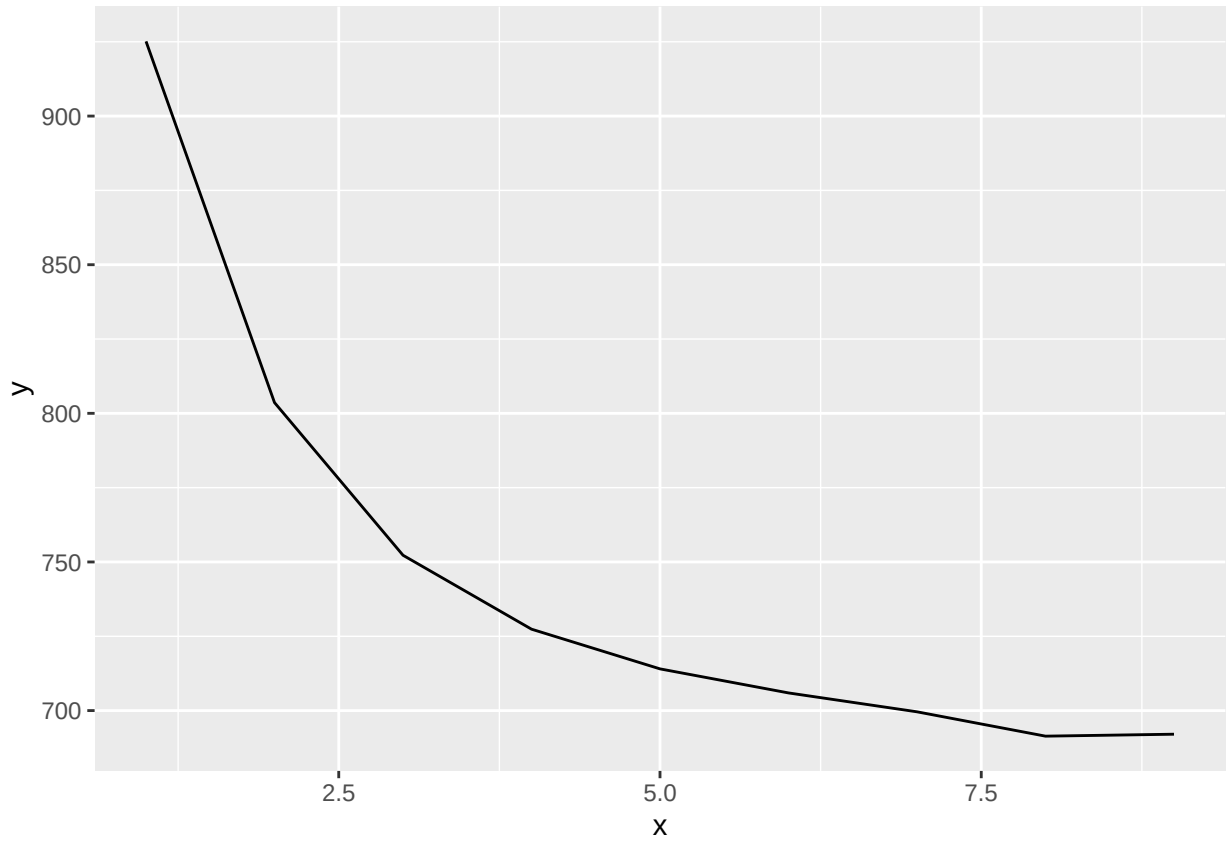
```
ggplot(rbind(data.frame(num_trees = num_trees, value = oob_se_by_num_trees, model = "rf"), data.frame(nu
  geom_line(aes(x = num_trees, y= value, col = model))
```

```
## Warning: Removed 8 row(s) containing missing values (geom_path).
```



Build RF models for 500 trees using different `mtry` values: 1, 2, ... the maximum. That maximum will be the number of features assuming that we do not binarize categorical features if you are using `randomForest` or the number of features assuming binarization of the categorical features if you are using `YARF`. Calculate oob s_e for all mtry values.

```
mtrys = 1:(ncol(diamonds_samp)-1)
oob_se_by_mtrys = array(NA, length(mtrys))

for (i in 1:length(mtrys)){
  rf_mod = randomForest(price~., data = diamonds_samp, ntree = 500, mtry = mtrys[i])
  oob_se_by_mtrys[i] = sd(diamonds_samp$price - rf_mod$predicted)
}

ggplot(data.frame(x=mtrys, y = oob_se_by_mtrys)) +
  geom_line(aes(x=x, y=y))
```




```
rm(list = ls())
```

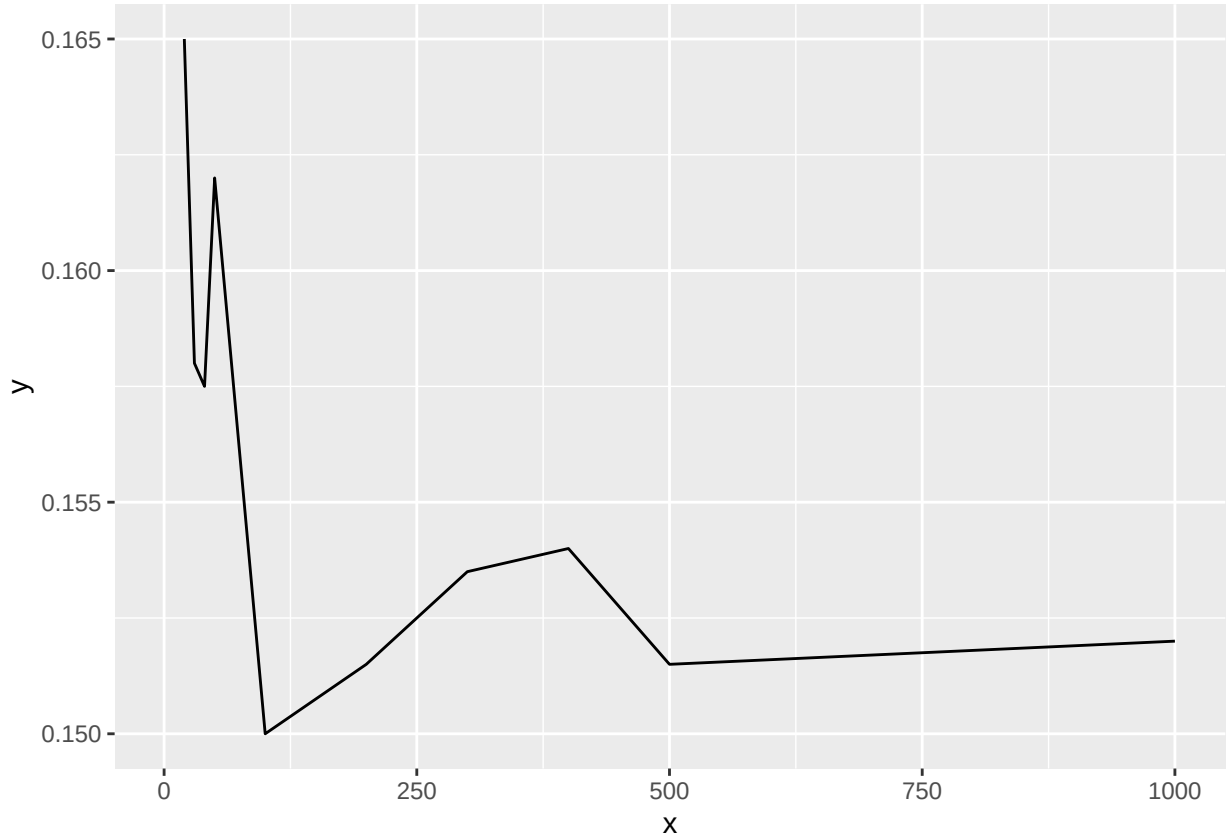
Take a sample of n = 2000 observations from the adult data.

```
pacman::p_load_gh("coatless/ucidata")
pacman::p_load(dplyr, ggplot2)
data(adult)
adult = na.omit(adult)

adult_samp = adult%>%
                  sample_n(2000)
```

Using the adult data, find the oob misclassification error for an RF model using 1, 2, 5, 10, 20, 30, 40, 50, 100, 200, 300, 400, 500, 1000 trees.

```
pacman::p_load(randomForest)
num_trees = c(1, 2, 5, 10, 20, 30, 40, 50, 100, 200, 300, 400, 500, 1000)
oob_ME_by_num_trees = array(NA, length(num_trees))

for (i in 1:length(num_trees)){
  rf_mod = randomForest(income~., data = adult_samp, ntree = num_trees[i] )
  oob_ME_by_num_trees[i] = mean(adult_samp$income != rf_mod$predicted)
}

ggplot(data.frame(x=num_trees, y = oob_ME_by_num_trees)) +
  geom_line(aes(x=x, y=y))
```

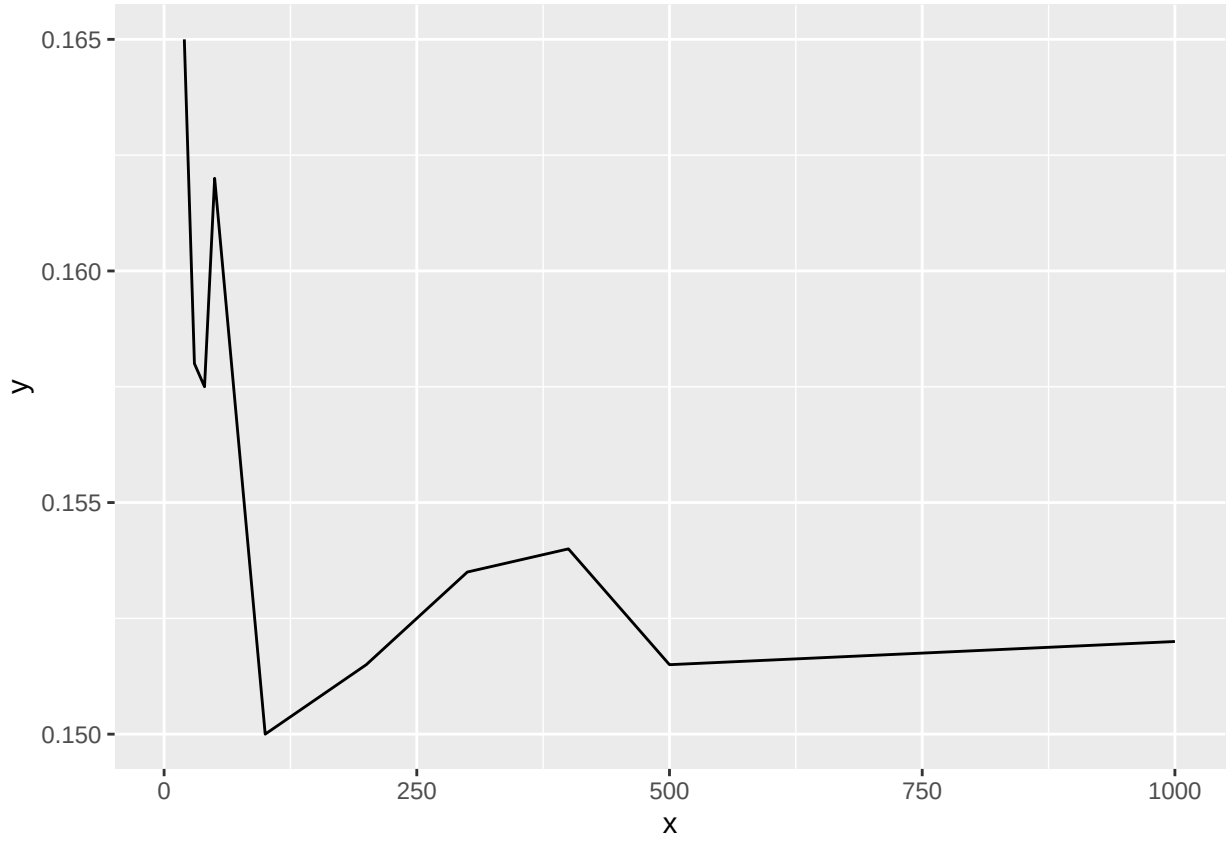
```
## Warning: Removed 4 row(s) containing missing values (geom_path).
```



Using the adult data, find the bootstrap misclassification error for a bagged-tree model using 1, 2, 5, 10, 20, 30, 40, 50, 100, 200, 300, 400, 500, 1000 trees.

```
oob_ME_by_num_trees_bag = array(NA, length(num_trees))

for (i in 1:length(num_trees)){
  rf_mod = randomForest(income~., data = adult_samp, ntree = num_trees[i], mtry = ncol(adult)-1)
  oob_ME_by_num_trees_bag[i] = mean(adult_samp$income != rf_mod$predicted)
}

ggplot(data.frame(x=num_trees, y = oob_ME_by_num_trees)) +
  geom_line(aes(x=x, y=y))
```

```
## Warning: Removed 4 row(s) containing missing values (geom_path).
```



What is the percentage gain / loss in performance of the RF model vs bagged trees model?

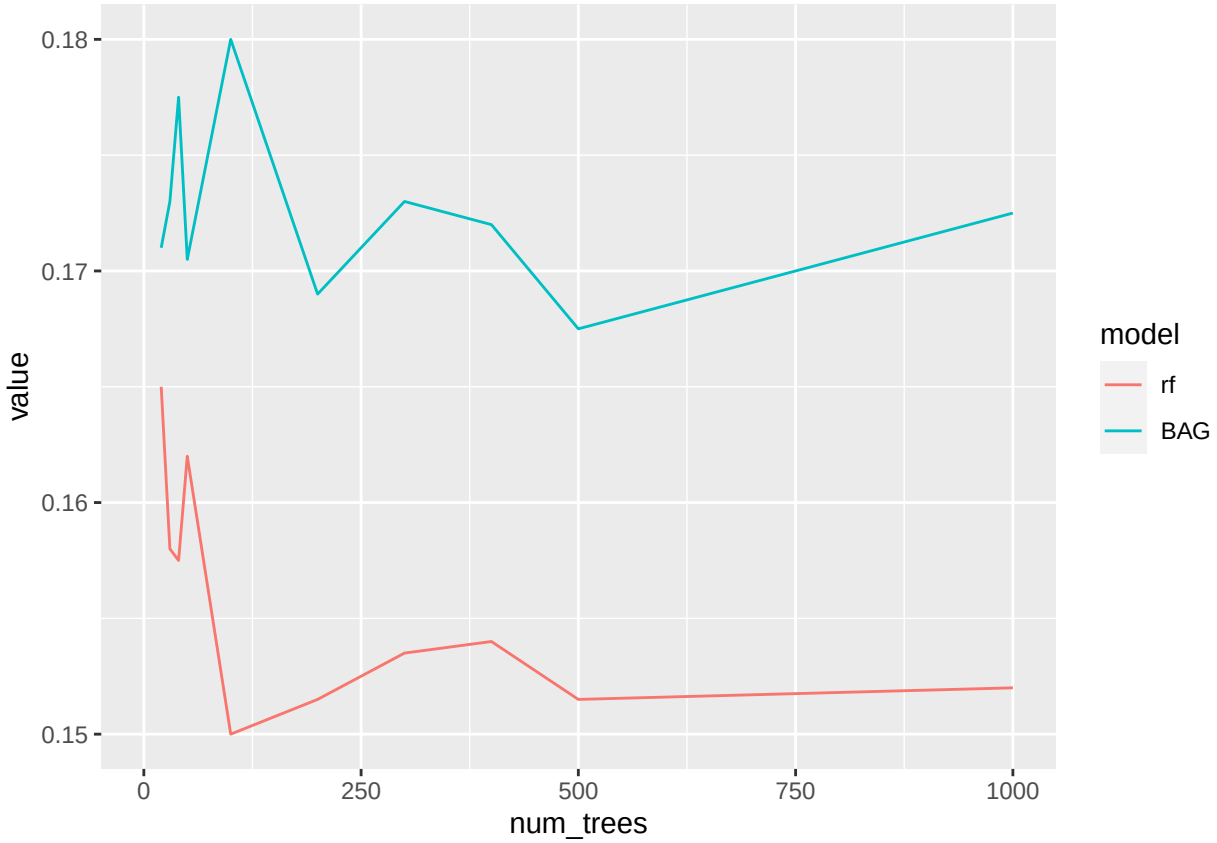
```
(oob_ME_by_num_trees - oob_ME_by_num_trees_bag ) / oob_ME_by_num_trees_bag *100
```

```
##  [1]         NA         NA         NA         NA  -3.508772  -8.670520
##  [7] -11.267606  -4.985337 -16.666667 -10.355030 -11.271676 -10.465116
## [13]  -9.552239 -11.884058
```

Plot oob bootstrap misclassification error by number of trees for both RF and bagged trees.

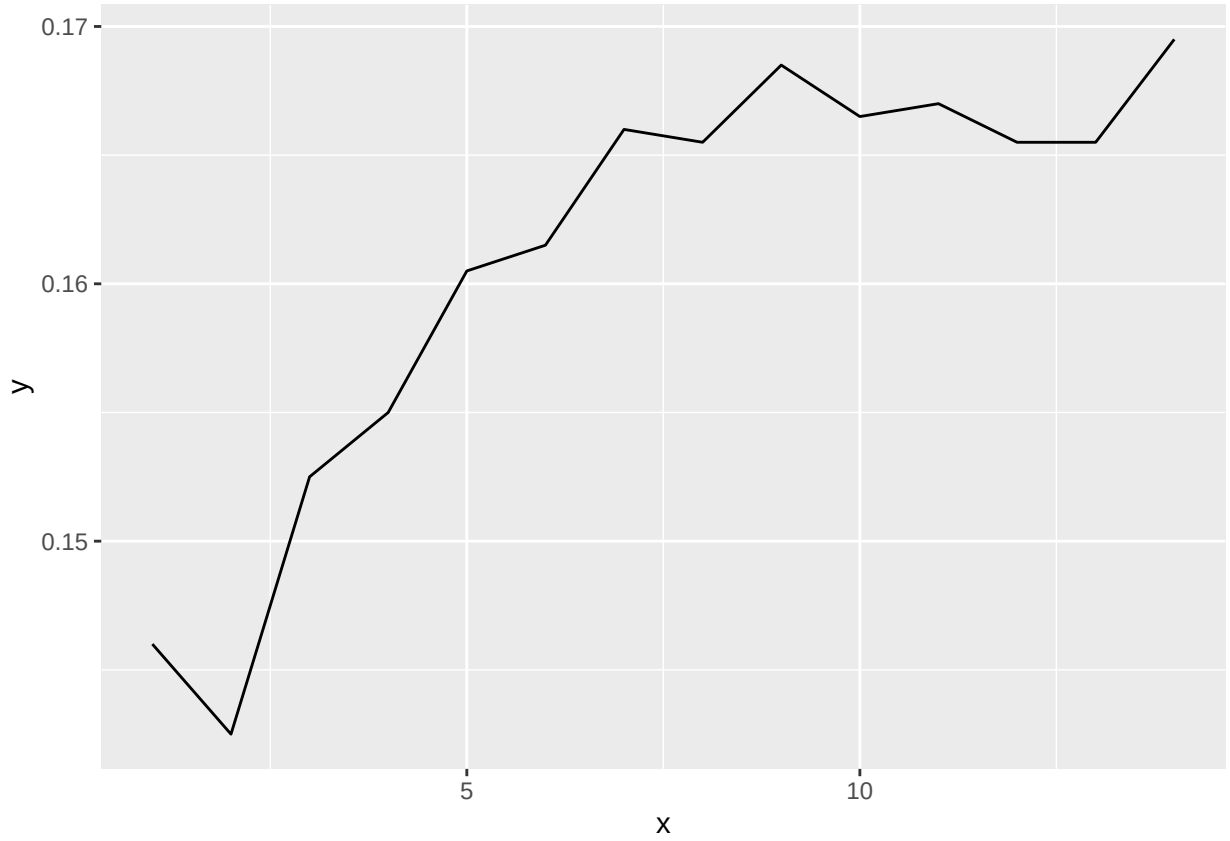
```
ggplot(rbind(data.frame(num_trees = num_trees, value = oob_ME_by_num_trees, model = "rf"), data.frame(nu
  geom_line(aes(x = num_trees, y= value, col = model))
```

```
## Warning: Removed 8 row(s) containing missing values (geom_path).
```



Build RF models for 500 trees using different `mtry` values: 1, 2, ... the maximum (see above as maximum is defined by the specific RF algorithm implementation).

```
mtrys = 1:(ncol(adult_samp)-1)
oob_ME_by_mtrys = array(NA, length(mtrys))

for (i in 1:length(mtrys)){
  rf_mod = randomForest(income~., data = adult_samp, ntree = 500, mtry = mtrys[i])
 oob_ME_by_mtrys[i] = mean(adult_samp$income != rf_mod$predicted)
}

ggplot(data.frame(x=mtrys, y = oob_ME_by_mtrys)) +
  geom_line(aes(x=x, y=y))
```




```
rm(list = ls())
```

Write a function `random_bagged_ols` which takes as its arguments `X` and `y` with further arguments `num_ols_models` defaulted to 100 and `mtry` defaulted to NULL which then gets set within the function to be 50% of available features. This argument builds an OLS on a bootstrap sample of the data and uses only `mtry < p` of the available features. The function then returns all the `lm` models as a list with size `num_ols_models`.

```
random_bagged_ols = function(X, y, num_ols_models = 100, mtry = NULL){
  p=ncol(X)
  mtry = (0.50*p)
  n = nrow(X)
  all_mods = list()

  for (i in 1:num_ols_models){

    n_train = sample(1:n, size = n, replace = TRUE)
    col_choose = sample(colnames(X), size = sample(mtry, size = 1, replace = FALSE))
    x_train = X[n_train, ]
    y_train = y[n_train]
    f = as.formula(paste("y ~", paste(col_choose[!col_choose %in% "y"], collapse = " + ")))
    all_mods[[i]] = lm(f, data = x_train)
  }
  all_mods
}
```

Load up the Boston Housing Data and separate into `X` and `y`.

```
pacman::p_load(MASS)
data(Boston)
y = Boston$medv
X = Boston[, 1:13]
X$medv= NULL
head(X)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
```

```
random_bagged_ols(X,y)
```

```
## [[1]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          dis
##     21.0240       0.3954
##
##
## [[2]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          tax           zn        indus
##  23.8701336   -0.0003248    0.0014243   -0.1109393
##
##
## [[3]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)        lstat          nox      ptratio         crim
##    20.59701     -0.10738      2.22039      0.11867     -0.03835
##
##
## [[4]]
##
## Call:
## lm(formula = f, data = x_train)
```

```
##
## Coefficients:
## (Intercept)           rm          rad        lstat          nox
##    27.65833     -1.01717      0.01899     -0.14149      5.20390
##
##
## [[5]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)           rm         chas          tax          age
##  20.8758271    0.3667412   -1.5874568    0.0001899   -0.0092580
##
##
## [[6]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)           rm         chas      ptratio         crim          age
##   25.777069    -0.152194     0.869753    -0.094356    -0.003157     0.010328
##       indus
##   -0.117611
##
##
## [[7]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          rad
##     22.8365      -0.0322
##
##
## [[8]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)        black           rm          age          dis
##    22.42563      0.00196      0.31739     -0.03465     -0.03709
##
##
## [[9]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
```

```
## (Intercept)         chas           rm      ptratio
##     29.2597       2.5184      -0.5936      -0.1730
##
##
## [[10]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)         chas
##     22.4525       0.8645
##
##
## [[11]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)         crim           zn
##    22.32977      0.01225      0.01465
##
##
## [[12]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          dis        black
##  21.5128215    0.3464670   -0.0009294
##
##
## [[13]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)      ptratio          dis        lstat        indus
##  24.2192809   -0.0456556   -0.0567931   -0.0004865   -0.0538746
##
##
## [[14]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)         crim        indus           zn
##    23.30385     -0.07676     -0.01892     -0.02772
##
##
```

```
## [[15]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)           zn           rm          nox        black
##   25.776221     0.025129     0.250175    -3.992801    -0.008397
##
##
## [[16]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          tax
##   2.249e+01    9.765e-05
##
##
## [[17]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          tax        black
##  24.4292951    0.0008065   -0.0063093
##
##
## [[18]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          rad           rm          dis           zn
##  19.0896734    0.0001411    0.3046923    0.5181997   -0.0363785
##
##
## [[19]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)        black        indus           rm         chas          nox
##   22.810515    -0.001005     0.071843    -0.132178    -0.391240     1.907738
##         age
##   -0.013512
##
##
## [[20]]
##
```

```
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          dis      ptratio           rm          nox
##  27.2220377   -0.0817762    0.0001046   -0.2748333   -4.8029553
##
##
## [[21]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          rad        indus           zn
##   20.960006     0.038673     0.099934     0.009916
##
##
## [[22]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)           zn        lstat          nox         crim          age
##   25.785316    -0.015506     0.052331    -4.499472    -0.006484    -0.018136
##
##
## [[23]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)        lstat          nox           zn
##    23.11026     -0.03055      0.02224     -0.01752
##
##
## [[24]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)      ptratio        lstat           rm          rad          dis
##    31.77037     -0.65088      0.00390      0.24407      0.06179      0.19390
##          zn
##    -0.01630
##
##
## [[25]]
##
## Call:
## lm(formula = f, data = x_train)
```

```
##
## Coefficients:
## (Intercept)         crim           rm          tax          dis           zn
##  22.2603744   -0.0751843    0.1071742   -0.0006057    0.1759093   -0.0214157
##       lstat
##  -0.0236107
##
##
## [[26]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)         chas
##      22.599       -1.116
##
##
## [[27]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)           rm
##    22.44367      0.01417
##
##
## [[28]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          age
##  22.5572498   -0.0003581
##
##
## [[29]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          rad        black
##   20.634859     0.018064     0.004848
##
##
## [[30]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
```

```
## (Intercept)        indus          tax        black
##   19.642510     0.010193     0.003926     0.003239
##
##
## [[31]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)           zn         chas          nox        indus
##   23.316248     0.007957    -0.764512    -0.792796    -0.033491
##
##
## [[32]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)           rm
##     16.9952       0.8866
##
##
## [[33]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)        black          dis      ptratio         crim
##   24.213105    -0.003161     0.091339    -0.045835    -0.015577
##
##
## [[34]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          dis        lstat           rm          tax         chas
##   34.173957     0.017831    -0.158303    -1.591672    -0.008707    -1.531326
##         nox
##    7.096138
##
##
## [[35]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)        indus        black
##   21.359197     0.070106     0.001119
```

```
##
##
## [[36]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          dis           zn        black           rm         chas
##   27.955359    -0.421649     0.028565    -0.006886    -0.262843    -0.149218
##
##
## [[37]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)           rm
##     27.4830      -0.7905
##
##
## [[38]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)        indus
##   22.574864    -0.003847
##
##
## [[39]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          tax          dis        lstat
##  23.8752329   -0.0008455   -0.1958062   -0.0202416
##
##
## [[40]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          dis          tax      ptratio
##   28.205584    -0.140631    -0.001405    -0.247571
##
##
## [[41]]
##
```

```
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)         crim        lstat
##   24.204609    -0.009566    -0.126123
##
##
## [[42]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          age          tax        lstat          rad         chas
##  22.2777061    0.0013860   -0.0007915    0.0288246    0.0267882   -1.8490831
##
##
## [[43]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)           zn        lstat
##    23.32967     -0.02279     -0.04173
##
##
## [[44]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)           rm
##     24.7999      -0.3603
##
##
## [[45]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          dis          age        lstat
##    23.99382     -0.16307     -0.03176      0.10598
##
##
## [[46]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
```

```
## (Intercept)         crim        lstat          tax         chas
##  22.8178109   -0.0583555   -0.0192074    0.0003062    1.0374725
##
##
## [[47]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          tax        lstat          nox      ptratio          rad
##   29.112666    -0.005956    -0.011727    -1.222653    -0.271244     0.172826
##
##
## [[48]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          tax        lstat           zn
##    24.63386     -0.00231     -0.08071     -0.01364
##
##
## [[49]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)        lstat          dis          age        black         chas
##   19.710445     0.122903     0.343894    -0.015614     0.002728     0.797094
##
##
## [[50]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          tax      ptratio           zn          nox
##    29.14927      0.00296     -0.26472     -0.02519     -4.77484
##
##
## [[51]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          rad          nox        black          age        indus
##   21.239584     0.024088    -4.392780     0.006571     0.007777     0.057590
##
##
```

```
## [[52]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          rad          dis         crim          age           zn
##    24.46261      0.04120     -0.10084     -0.01207     -0.03131     -0.02432
##       lstat
##     0.03933
##
##
## [[53]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)           rm         chas        lstat          tax          nox
##   26.149665    -0.728752    -0.861505    -0.084860     0.002868     3.655891
##       indus
##   -0.100289
##
##
## [[54]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          age          rad        indus         crim           rm
##    25.22979     -0.03060      0.03963      0.02686     -0.12042      0.04650
##         dis
##    -0.29653
##
##
## [[55]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          age        lstat      ptratio
##   23.524534    -0.022336     0.027124     0.009637
##
##
## [[56]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          age          nox
##    23.64539      0.04111     -7.15193
```

```
##
##
## [[57]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)        lstat      ptratio         crim          tax
##   28.766460     0.033786    -0.440092    -0.097796     0.004517
##
##
## [[58]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)         crim      ptratio          age
##    23.48891     -0.03008     -0.03557     -0.00250
##
##
## [[59]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)           zn         chas          age           rm        indus
##   22.713650     0.009023     1.180429     0.001332     0.031811    -0.054957
##
##
## [[60]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          rad           zn        lstat        indus         crim
##   25.674302    -0.071186    -0.001105    -0.093453     0.087400     0.014855
##     ptratio
##   -0.123046
##
##
## [[61]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)      ptratio           rm          tax        lstat
##   20.748437     0.036089     0.410466    -0.003157    -0.013959
##
##
```

```
## [[62]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)        lstat          age
##   23.901338    -0.134755     0.003958
##
##
## [[63]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)         chas           zn        lstat          dis      ptratio
##   26.036155     0.201699    -0.002481    -0.017986     0.135629    -0.207371
##
##
## [[64]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          nox        indus      ptratio           zn        black
##   22.614576     5.827548    -0.078696    -0.062403    -0.001458    -0.003514
##
##
## [[65]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)           rm      ptratio        lstat          tax          dis
##   25.529285    -0.740984    -0.015320    -0.055376     0.006323     0.051036
##         age
##   -0.001340
##
##
## [[66]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          nox
##     21.9929       0.9762
##
##
## [[67]]
##
```

```
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          tax          age          nox          rad        lstat
##  23.9370645    0.0008836   -0.0051838   -2.0849769   -0.1443129    0.0932909
##
##
## [[68]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          nox         crim        indus
##    20.48869      5.38433     -0.02730     -0.07216
##
##
## [[69]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          age          dis        black
##   19.768110     0.004918     0.076981     0.006130
##
##
## [[70]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          rad          tax      ptratio          dis
##   1.550e+01    5.150e-03   -5.367e-06    3.588e-01    8.147e-02
##
##
## [[71]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          nox          age
##    23.27348     -4.83494      0.02826
##
##
## [[72]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
```

```
## (Intercept)           zn          age
##   24.411353    -0.008334    -0.026653
##
##
## [[73]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)      ptratio         crim          nox           zn          rad
##   19.441642    -0.002022     0.024746     5.631055     0.003923    -0.009848
##
##
## [[74]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          rad        lstat
##    22.69551      0.07088     -0.06917
##
##
## [[75]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)        black         crim          nox          age      ptratio
##   21.112373    -0.004741     0.005039     4.167698    -0.058723     0.256259
##
##
## [[76]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)      ptratio
##    22.19760      0.01803
##
##
## [[77]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)      ptratio
##    20.73360      0.09725
##
##
```

```
## [[78]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)           zn          rad
##   22.418195     0.002969     0.007502
##
##
## [[79]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          dis
##     21.8755       0.1697
##
##
## [[80]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)        lstat          nox          rad         chas           rm
##    36.85581      0.01829     -7.55149      0.02744     -0.06321     -1.69039
##
##
## [[81]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)         chas          dis        lstat           rm           zn
##   13.627626     0.506204     0.250261     0.197107     0.939829    -0.006725
##        crim
##   -0.093518
##
##
## [[82]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)      ptratio          rad          age          dis
##    25.73004      0.05042      0.03352     -0.03725     -0.49924
##
##
## [[83]]
##
```

```
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)        lstat
##     22.8578      -0.0259
##
##
## [[84]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)         chas          rad        black        indus         crim
##   23.157841     1.592333    -0.057181    -0.004384     0.111974     0.026538
##
##
## [[85]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)        lstat         chas          dis          tax          age
##   26.141996    -0.154630    -0.558361    -0.390105    -0.001571     0.003080
##       indus
##    0.015671
##
##
## [[86]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)        lstat          nox
##    21.20903     -0.02719      3.00253
##
##
## [[87]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)        black
##    21.94092      0.00163
##
##
## [[88]]
##
## Call:
## lm(formula = f, data = x_train)
```

```
##
## Coefficients:
## (Intercept)         crim        lstat        indus          rad
##    22.96135      0.02088      0.01536      0.01028     -0.08308
##
##
## [[89]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          nox
##      24.374       -3.333
##
##
## [[90]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)        black          dis
##   23.149990    -0.004203     0.227351
##
##
## [[91]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          nox        lstat           rm          tax         chas
##   27.180165     5.306488    -0.149463    -1.184907     0.003849    -1.045902
##         dis
##    0.070429
##
##
## [[92]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)        lstat         chas          tax           zn        black
##   16.179174     0.054214     0.377188     0.005232    -0.016355     0.010985
##        crim
##   -0.053671
##
##
## [[93]]
##
## Call:
## lm(formula = f, data = x_train)
```

```
##
## Coefficients:
## (Intercept)          nox           rm      ptratio          dis
##    17.09869      4.53630      0.26473      0.02309      0.22159
##
##
## [[94]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)      ptratio         crim
##    19.83448      0.15321     -0.02934
##
##
## [[95]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)           rm        lstat          nox        black          dis
##   21.414595     0.314715     0.102520    -5.605560     0.008309    -0.544373
##
##
## [[96]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          age           rm          rad          nox
##   24.752512     0.009487     0.094547     0.019937    -6.573646
##
##
## [[97]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)           zn           rm
##    25.72866     -0.01087     -0.48587
##
##
## [[98]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)          tax          dis          rad
##   22.892957     0.002598    -0.157163    -0.085900
```

```
##
##
## [[99]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)        indus        black          tax         crim          dis
##  26.6030940   -0.3196982    0.0079276   -0.0006066    0.0039098   -0.8758299
##        chas
##   2.2209961
##
##
## [[100]]
##
## Call:
## lm(formula = f, data = x_train)
##
## Coefficients:
## (Intercept)         crim          dis          age         chas
##   22.590152     0.082544    -0.053012    -0.004689     2.965541
```

Similar to lab 1, write a function that takes a matrix and punches holes (i.e. sets entries equal to `NA`) randomly with an argument `prob_missing`.

```
make_holes = function(X, prob_missing = NULL){
  X = as.matrix(X)
  n = nrow(X)
  p = ncol(X)
  holes = matrix(nrow = n, ncol = p, sample(c(rep(0, n*p*(1-prob_missing)), rep(3, n*p*prob_missing))))
  for(i in 1:n){
    for(j in 1:p){
      if(holes[i,j]==3){
        X[i,j]=NA
    }
  }
}
X
}
```

Create a matrix `Xmiss` which is `X` but has missingness with probability of 10%.

```
X_miss = make_holes(X, 0.10)
```

```
## Warning in matrix(nrow = n, ncol = p, sample(c(rep(0, n * p * (1 -
## prob_missing)), : data length [6577] is not a sub-multiple or multiple of the
## number of rows [506]
```

```
table(is.na(X_miss))
```

```
##
## FALSE  TRUE
##  5921   657
```

Use a random forest modeling procedure to iteratively fill in the `NA`'s by predicting each feature of X using every other feature of X. You need to start by filling in the holes to use RF. So fill them in with the average of the feature.

```
##use MissForest

pacman::p_load(randomForest)
library(tidyr)

X = data.frame(X_miss)
n = nrow(X)
p = ncol(X)

for(i in 1:n){
  for(j in 1:p){
    if(is.na(X[i,j])){
      X_new = X %>%
 replace_na(as.list(colMeans(X, na.rm = TRUE)))

      mod = randomForest(X_new[,j] ~ ., data = X_new, ntree = 100)

      X[i,j] = predict(mod, X_new[i,])
    }
  }
}
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?

## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?

## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```
head(X)
```

```
##      crim       zn    indus       chas       nox       rm      age      dis rad
## 1 0.00632 18.00000 10.69901 0.00000000 0.5502501 6.575000 65.20000 4.090000   1
## 2 0.02731  0.00000  7.07000 0.00000000 0.4690000 6.323183 78.90000 4.967100   2
## 3 0.02729  0.00000  7.07000 0.00000000 0.4690000 7.185000 61.10000 4.967100   2
## 4 0.03237  0.00000  2.18000 0.07596923 0.4580000 6.998000 67.87211 6.062200   3
## 5 0.06905 12.93937  2.18000 0.00000000 0.4580000 7.147000 54.20000 6.062200   3
## 6 0.02985  0.00000  2.18000 0.00000000 0.4580000 6.313107 58.70000 3.862076   3
##       tax  ptratio    black   lstat
## 1 296.0000 15.30000 396.9000  4.9800
## 2 242.0000 18.48499 396.9000  9.1400
## 3 400.9399 17.80000 392.8300  4.0300
## 4 222.0000 18.70000 394.6300  2.9400
## 5 222.0000 18.70000 396.9000  5.3300
## 6 222.0000 18.70000 359.0642 12.2669
```

```
table(is.na(X))
```

```
##
## FALSE
##  6578
```