

# Lab 9

Christian Guaraca

11:59PM May 10, 2021

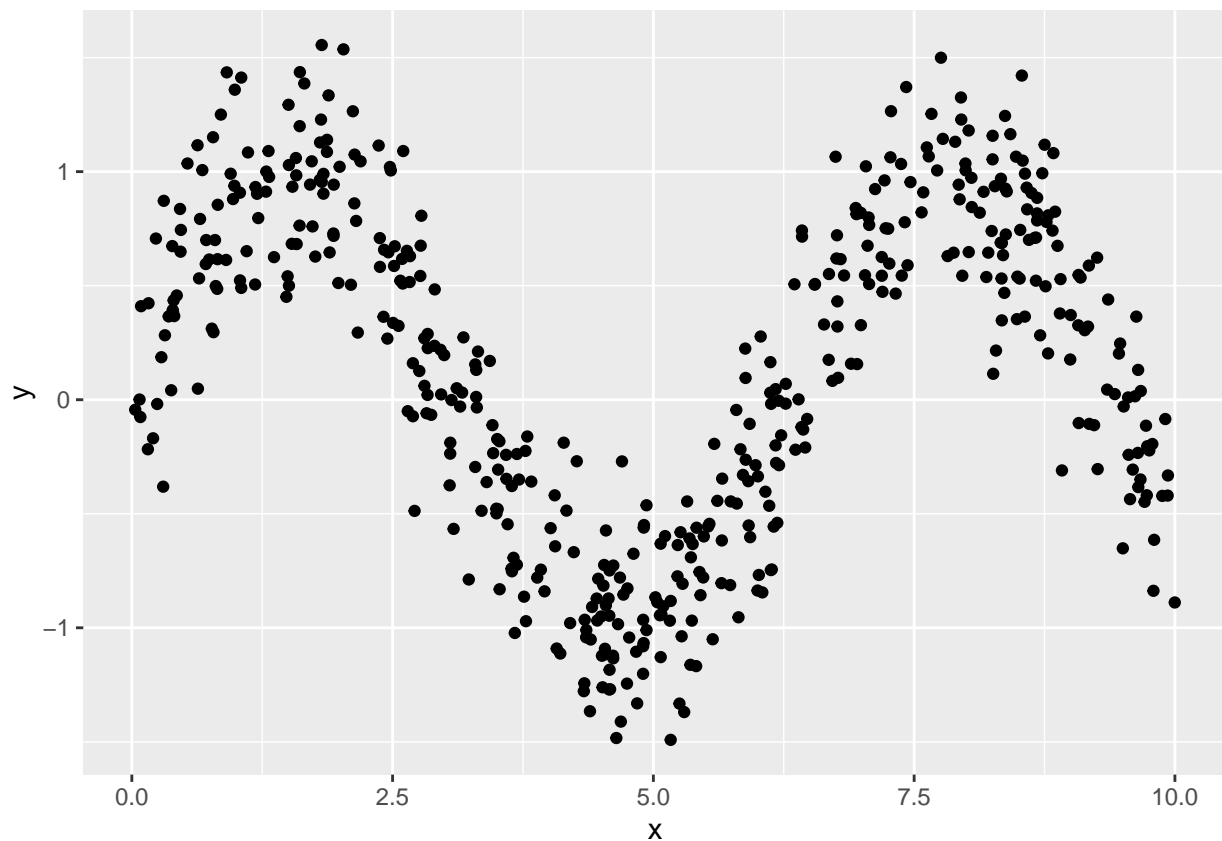
Here we will learn about trees, bagged trees and random forests. You can use the **YARF** package if it works, otherwise, use the **randomForest** package (the standard).

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

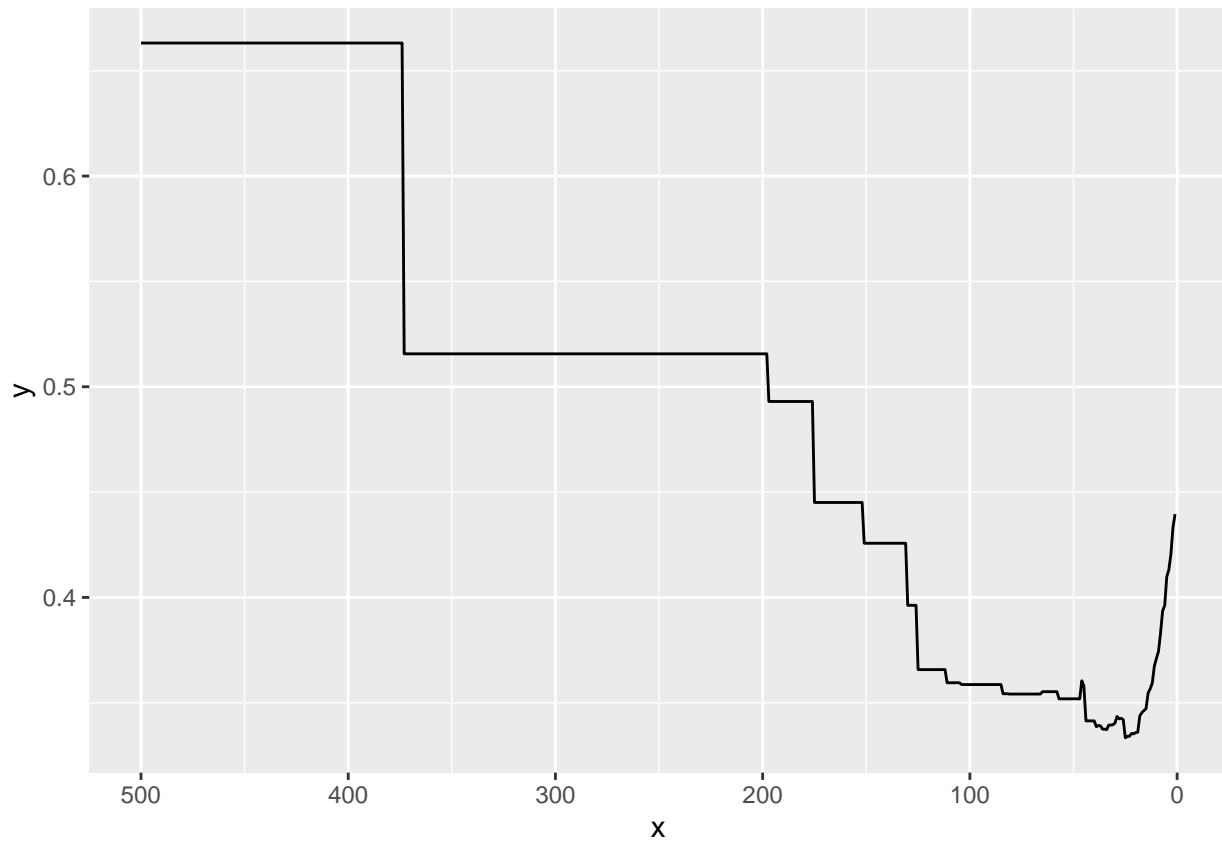
```
x_test = runif(n, x_min, x_max)
y_test = y_x(x_test, sigma)
```

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

```
pacman::p_load(randomForest)
```

```
node_sizes = 1:n
se_by_node_sizes = array(NA, 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, sampszie = n)
  y_hat_test = predict(rf_mod, data.frame(x = x_test))
  se_by_node_sizes[i] = sd(y_test - y_hat_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] 25
```

Plot the regression tree model  $g(x)$  with the optimal node size.

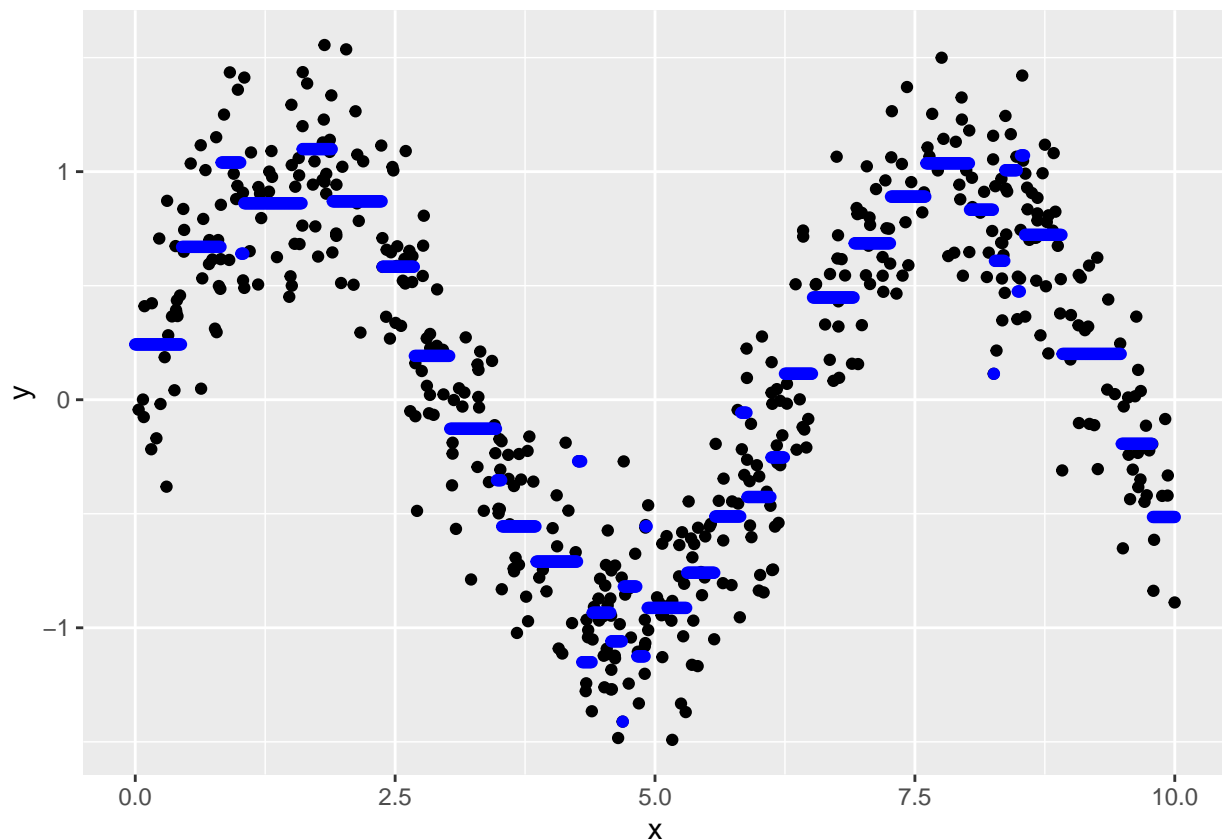
```
rf_mod = randomForest(x = data.frame(x = x_train), y = y_train, ntree = 1, replace = FALSE, sampsize = 1000)
```

```
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(aes(x = x, y = y), color = 'blue')
```



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.

*#TO-DO*

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

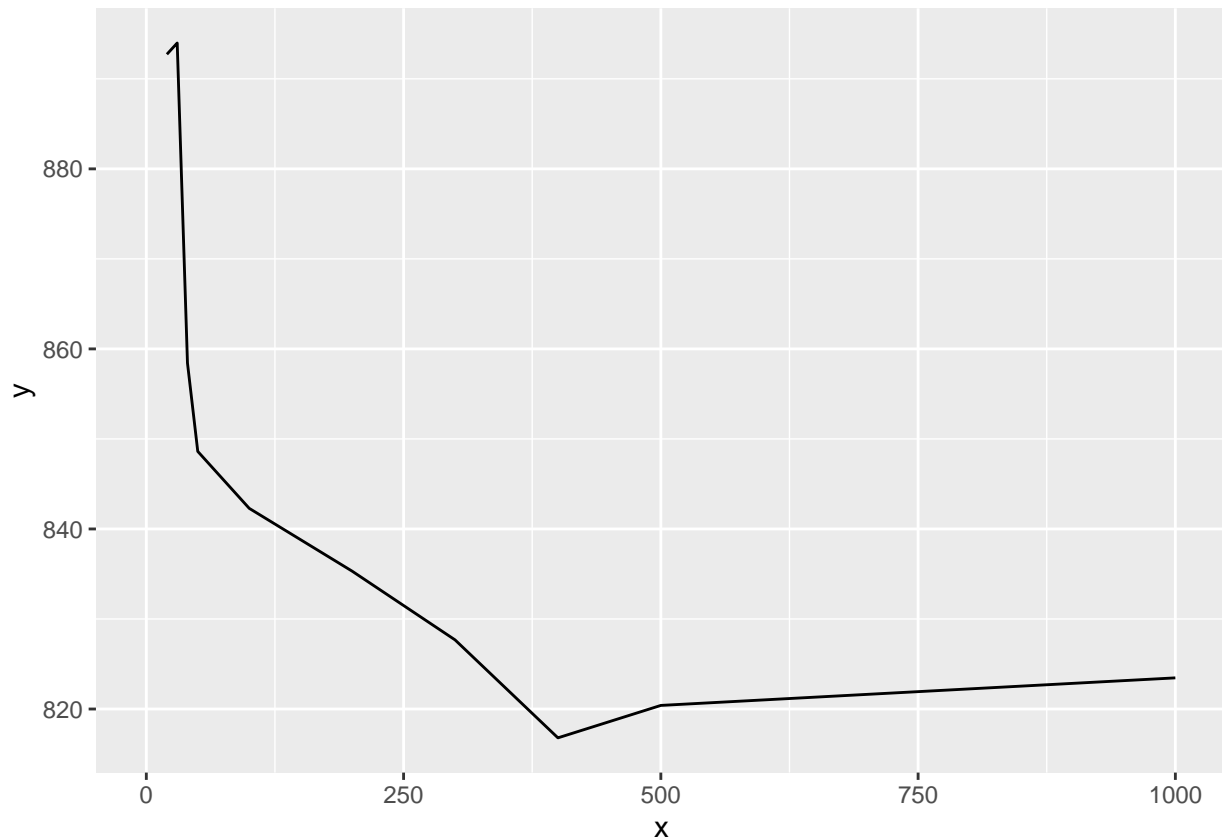
Take a sample of  $n = 2000$  observations from the diamonds data.

```
pacman::p_load(dplyr)
diamond_sample = diamonds%>%
  sample_n(2000)
```

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 = diamond_sample, ntree = num_trees[i])
  oob_se_by_num_trees[i] = sd(diamond_sample$price - rf_mod$predicted)
}
ggplot(data.frame(x = num_trees, y = oob_se_by_num_trees)) +
  geom_line(aes(x=x, y=y))
```

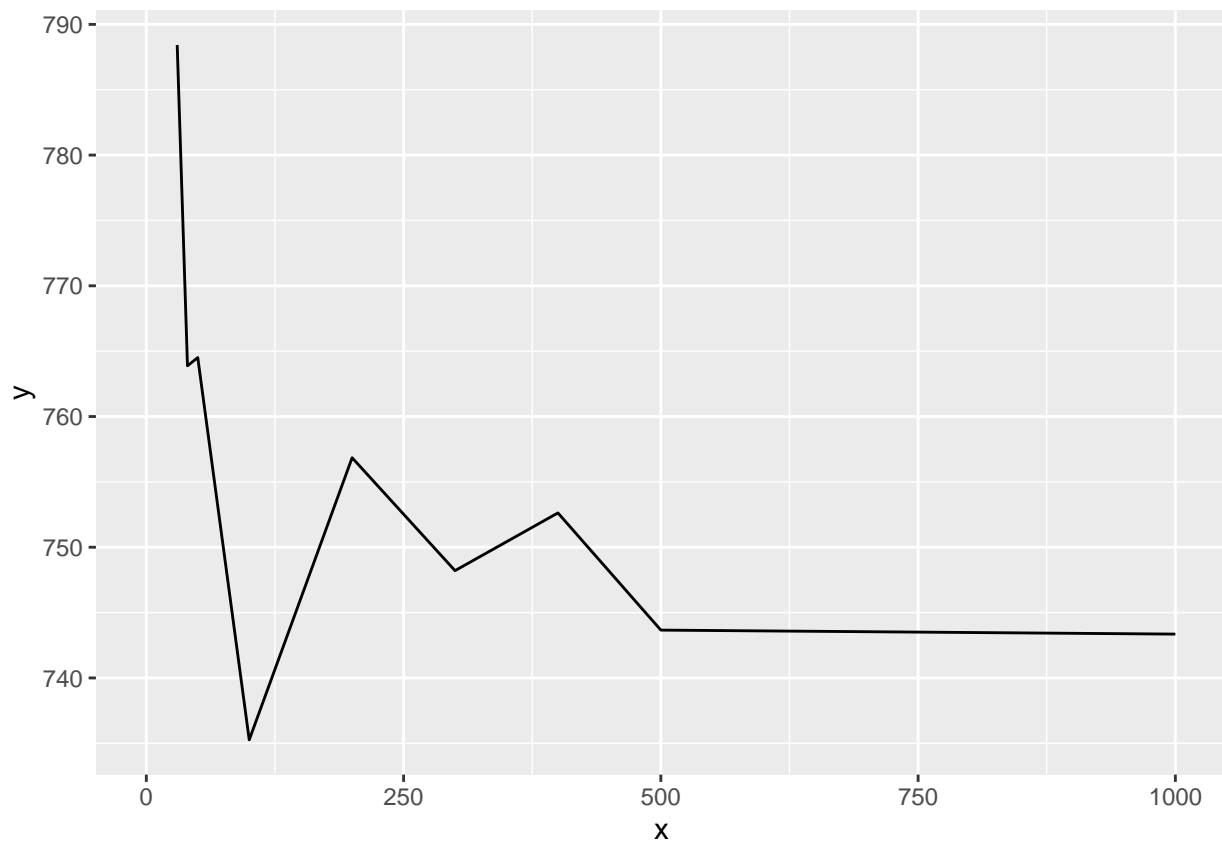
## Warning: Removed 4 row(s) containing missing values (geom\_path).



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. Plot.

```
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 = diamond_sample, ntree = num_trees[i], mtry = ncol(diamond_samp
  oob_se_by_num_trees_bag[i] = sd(diamond_sample$price - rf_mod$predicted)
}
ggplot(data.frame(x = num_trees, y = oob_se_by_num_trees_bag)) +
  geom_line(aes(x=x, y=y))
```

## Warning: Removed 5 row(s) containing missing values (geom\_path).



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

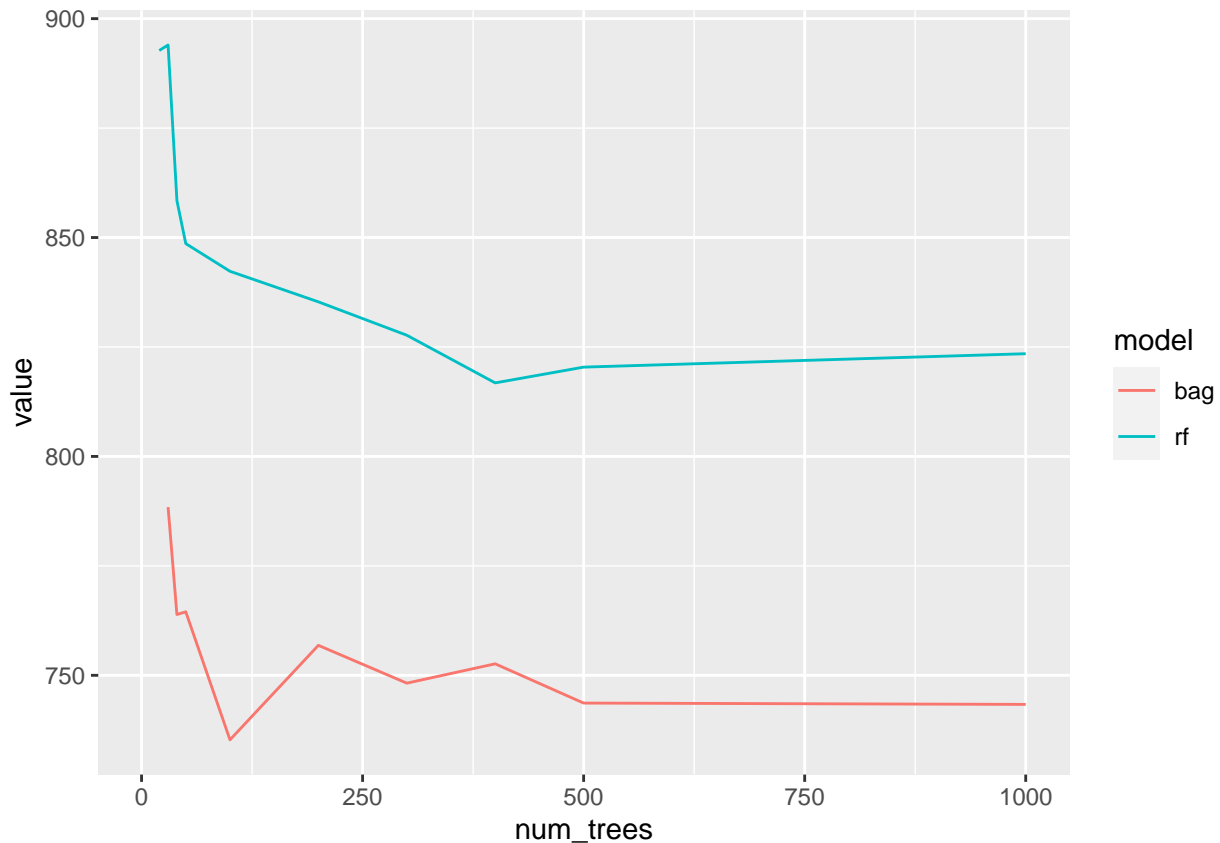
```
(oob_se_by_num_trees - oob_se_by_num_trees_bag) / oob_se_by_num_trees_bag * 100
```

```
## [1]      NA      NA      NA      NA      NA 13.388450 12.377586
## [8] 11.001152 14.555965 10.367502 10.621626  8.526678 10.319072 10.776191
```

Plot oob s\_e by number of trees for both RF and bagged trees using a long data frame.

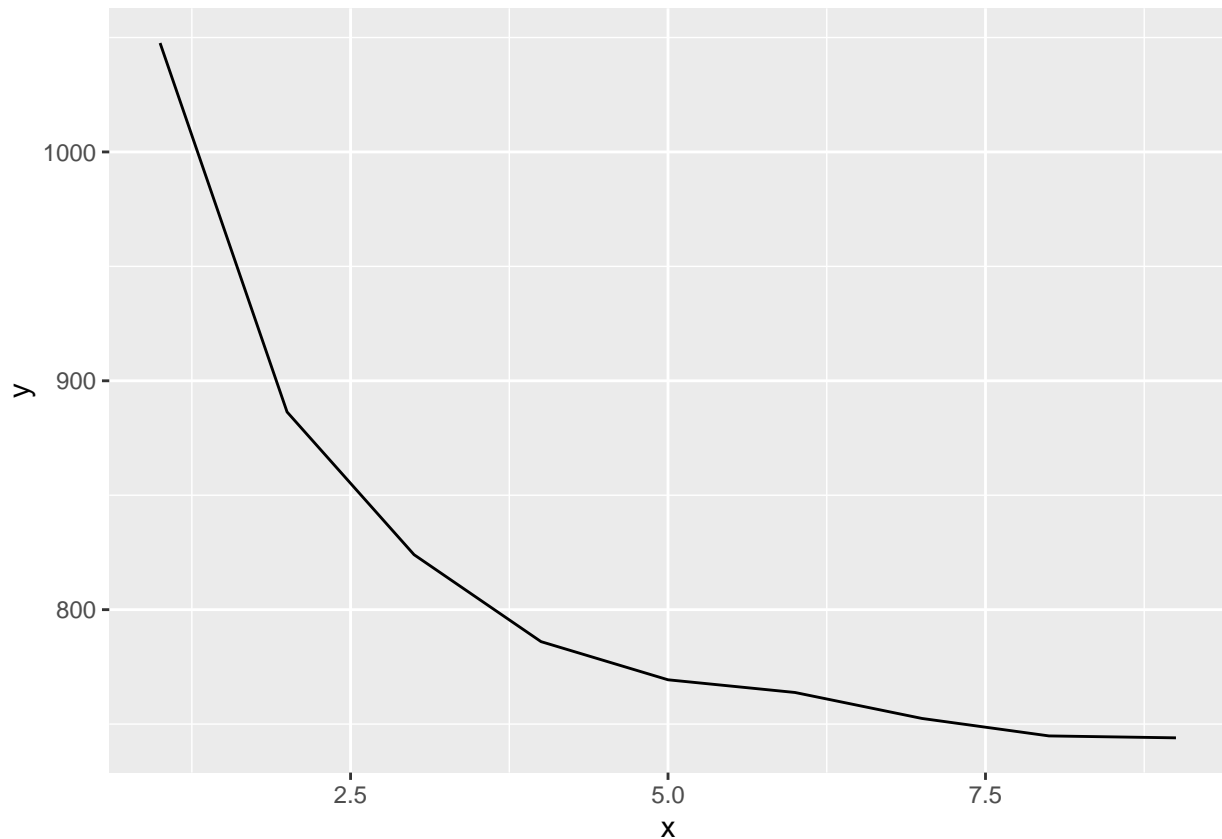
```
ggplot(rbind(data.frame(num_trees = num_trees, value = oob_se_by_num_trees, model = 'rf'), data.frame(n
  geom_line(aes(x = num_trees, y = value, color = model))
```

```
## Warning: Removed 9 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. Plot.

```
m_trys = 1:(ncol(diamond_sample)-1)
oob_m_trys = array(NA, length(m_trys))
for(i in 1:length(m_trys)){
  rf_mod = randomForest(price ~., data = diamond_sample, mtry = m_trys[i])
  oob_m_trys[i] = sd(diamond_sample$price - rf_mod$predicted)
}
ggplot(data.frame(x = m_trys, y = oob_m_trys)) +
  geom_line(aes(x=x, y=y))
```



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

```
pacman::p_load_gh('coatless/ucidata')
data(adult)
adult = na.omit(adult)

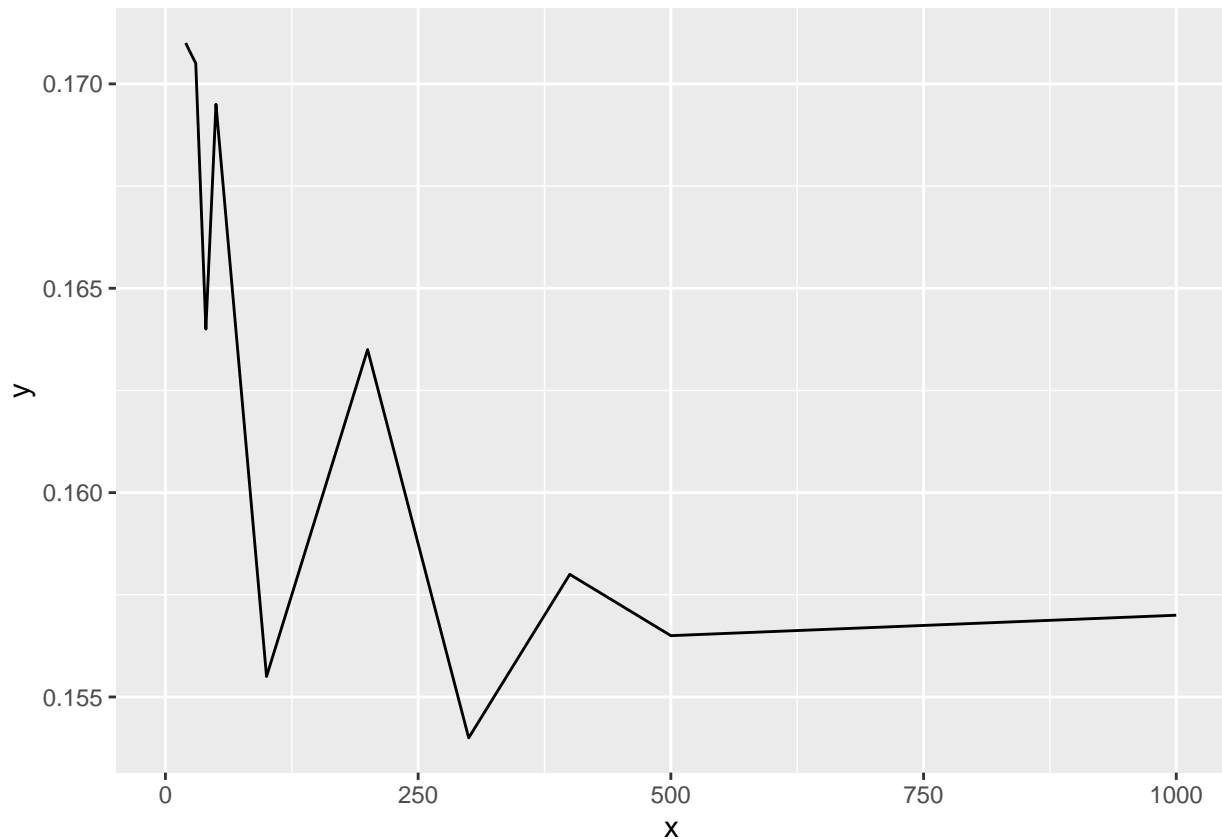
adult_samp = adult %>%
  sample_n(2000)
```

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

```
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).
```

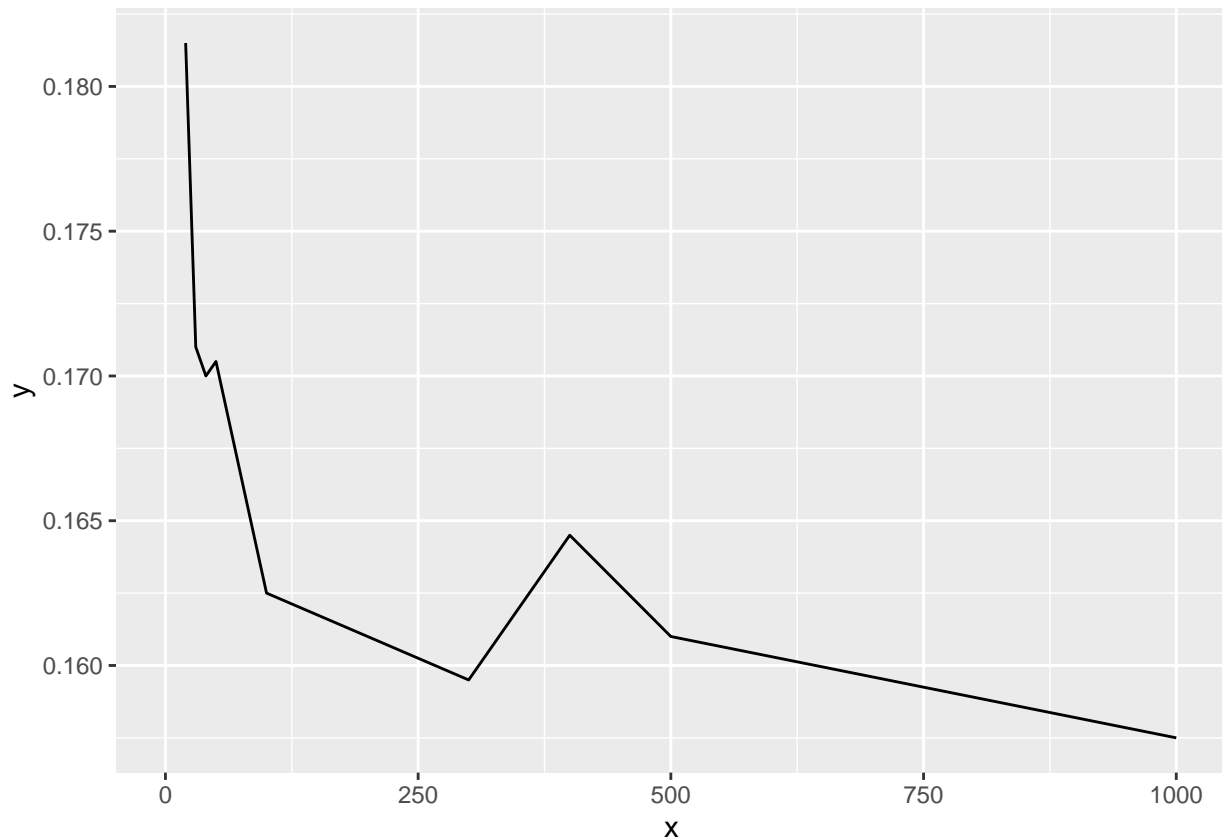




Using the adult data, find the bootstrap misclassification error for a bagged-tree model using 1, 2, 5, 10, 20, 30, 40, 50, 100, 200, 300, 400, 500, 1000 trees. Plot.

```
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_samp) - 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_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?

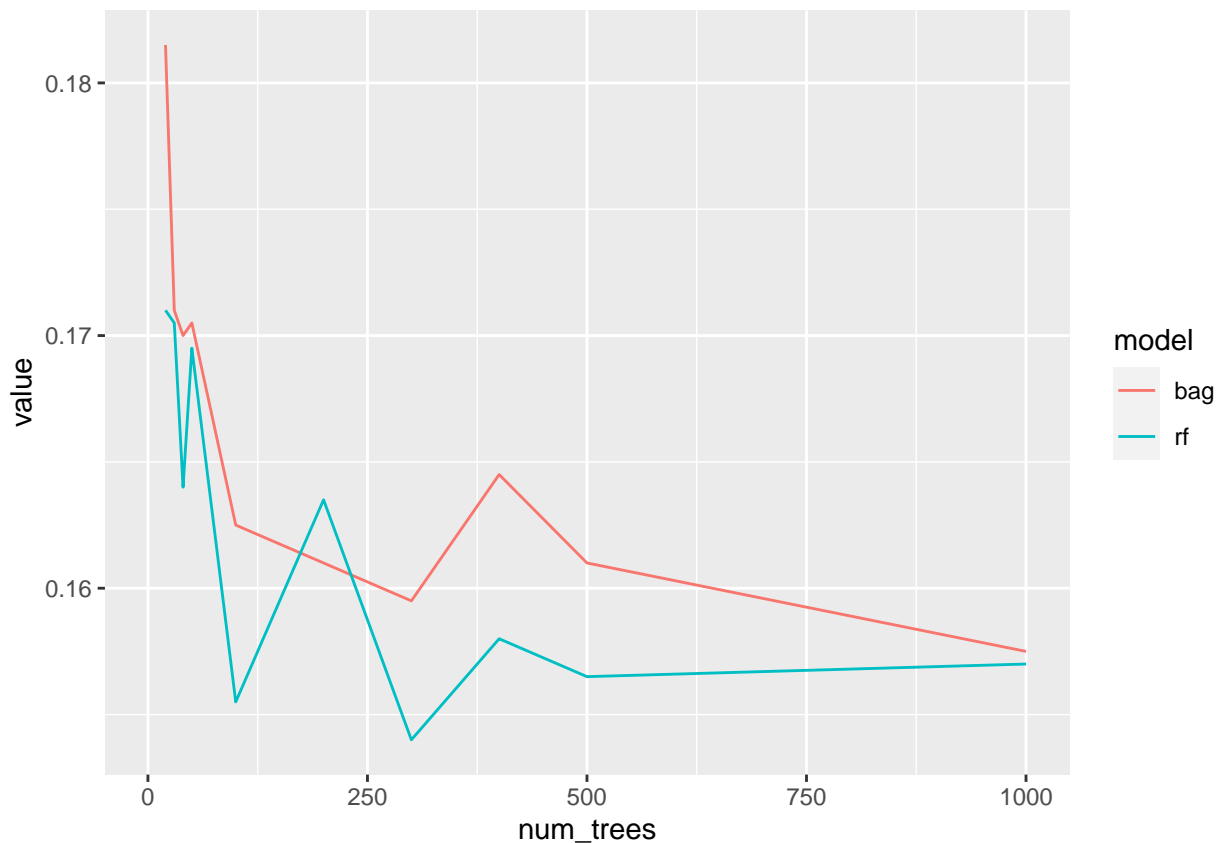
```
(oob_me_by_num_trees - oob_me_by_num_trees_bag) / oob_me_by_num_trees_bag * 100
```

```
## [1]      NA      NA      NA      NA -5.7851240 -0.2923977
## [7] -3.5294118 -0.5865103 -4.3076923  1.5527950 -3.4482759 -3.9513678
## [13] -2.7950311 -0.3174603
```

Plot bootstrap misclassification error by number of trees for both RF and bagged trees using a long data frame.

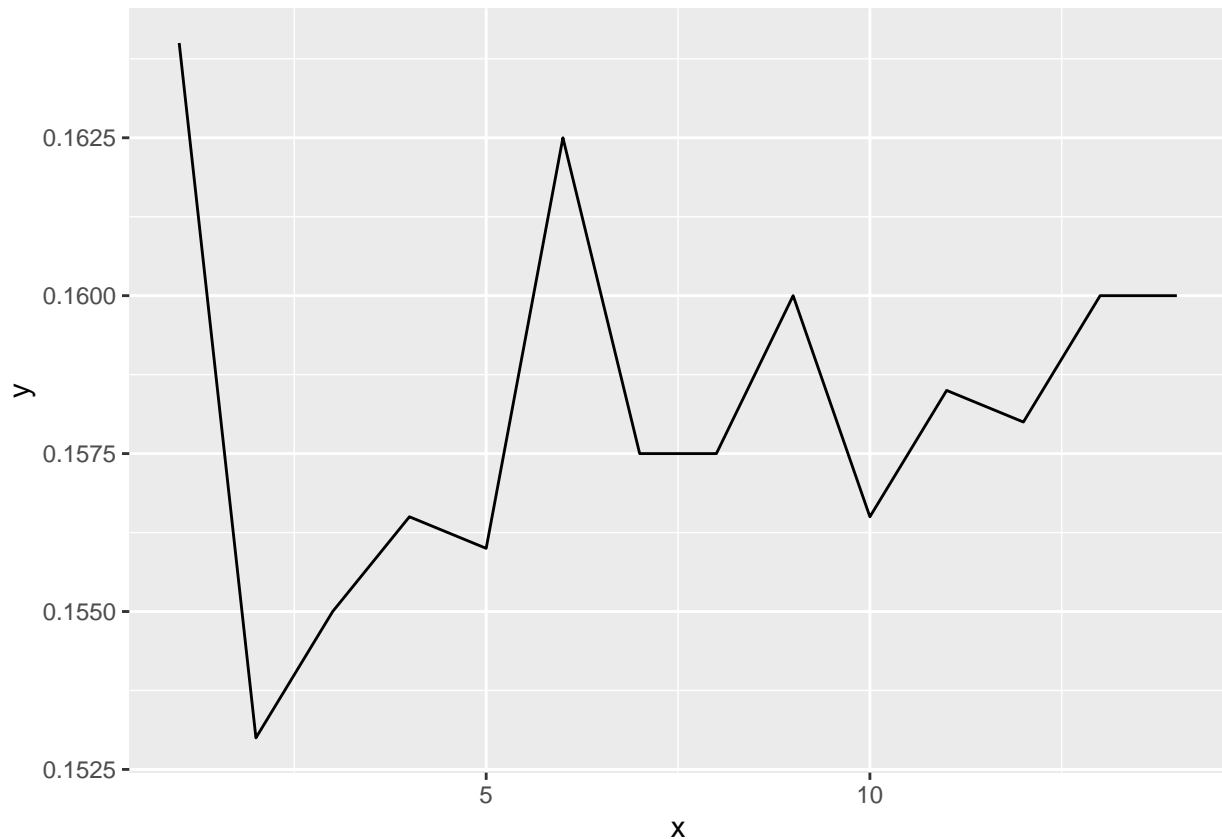
```
ggplot(rbind(data.frame(num_trees = num_trees, value = oob_me_by_num_trees, model = 'rf'), data.frame(n
  geom_line(aes(x = num_trees, y = value, color = model))
```

```
## 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). Plot.

```
m_trys = 1:(ncol(adult_samp)-1)
oob_me_m_trys = array(NA, length(m_trys))
for(i in 1:length(m_trys)){
  rf_mod = randomForest(income ~., data = adult_samp, mtry = m_trys[i])
  oob_me_m_trys[i] = mean(adult_samp$income != rf_mod$predicted)
}
ggplot(data.frame(x = m_trys, y = oob_me_m_trys)) +
  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){
  bagged_list = list()
  col_name = colnames(X)
  for(i in 1:num_ols_models){
    mtry = sample(col_name, floor((ncol(X) - 1) / 2))
    model = lm(y~., data = X[,mtry])
    bagged_list[[i]] = model
  }
  bagged_list
}
```

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

```
pacman::p_load(MASS)
data(Boston)
X = Boston
y = Boston$medv
X$medv = NULL
```

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`.

```
punch_holes = function(X, prob_missing){  
  n = nrow(X)  
  p = ncol(X)  
  M = matrix(rbinom(n * p, 1, prob_missing), nrow = n, ncol = p)  
  X[M == 1] = NA  
  X  
}
```

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

```
Xmiss = punch_holes(X, .10)
```