Ass2

JZHU951 120683565

5/13/2021

1. Inflows into New Zealand's major lakes

The New Zealand has a number of large hydro-electric dams that facilitate the storage of water in lakes for electricity production. The weekly inflow of water into these lakes is measured in millions of m^3, and has been recorded from 1988 to 2013 in the inflows.csv data set available on Canvas. In this question we will be performing exploratory data analysis on this data set. For this question you need to first install RStudio (or use rstudio.cloud), and download inflows.csv and locations.csv from Canvas. Remember to first load the tidyverse library, and then import inflows.csv.

```
library(s20x)
library(tidyverse)
inflows <- read_csv("inflows.csv", col_names = TRUE)
```

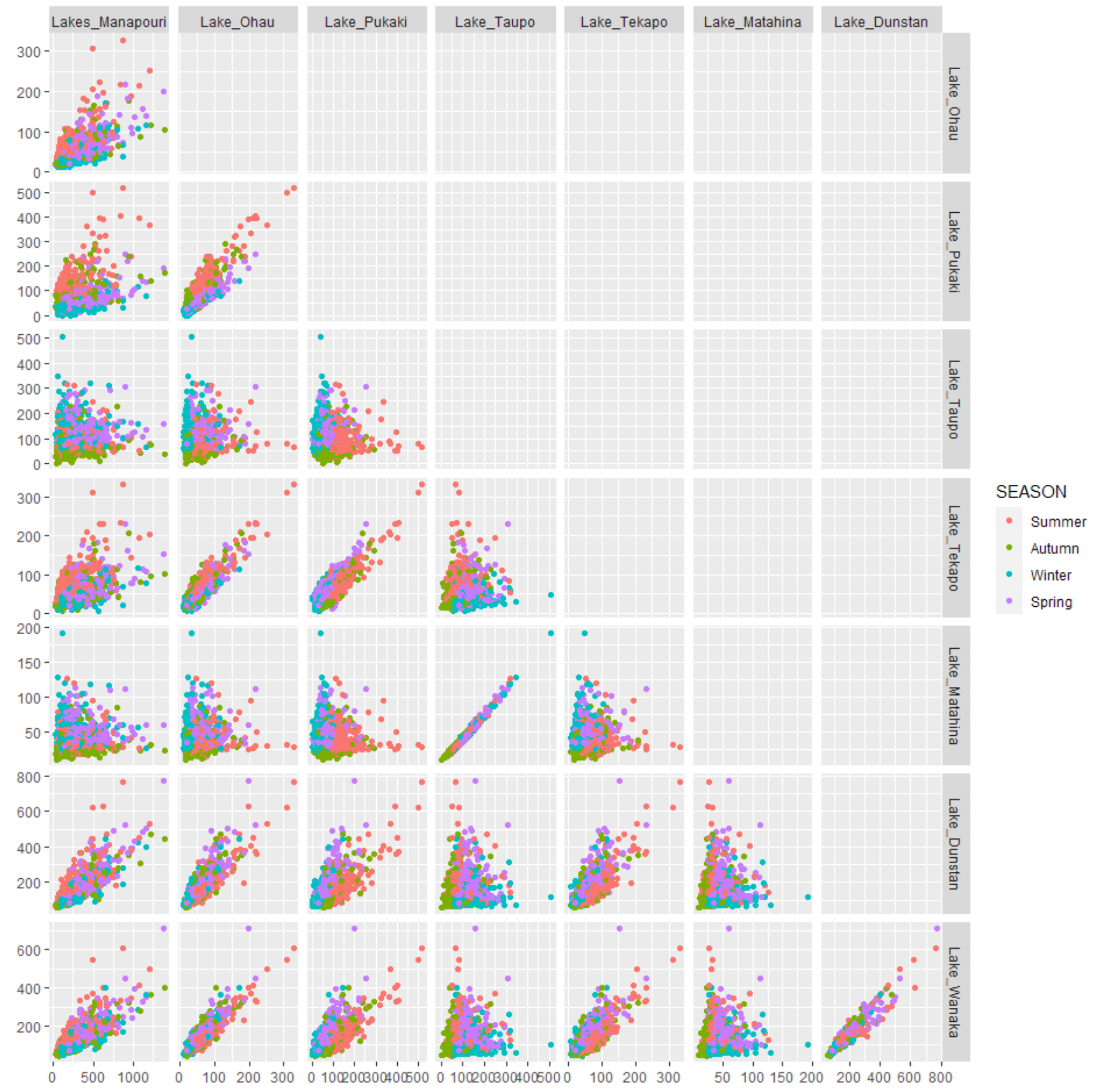
```
## -- Attaching packages --------------------------------------- tidyverse
1.3.1 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.1     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------
tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

##
## -- Column specification --------------------------------------------------
------
## cols(
##   YEAR = col_double(),
##   WEEK = col_double(),
##   Lakes_Manapouri = col_double(),
##   Lake_Ohau = col_double(),
##   Lake_Pukaki = col_double(),
##   Lake_Taupo = col_double(),
##   Lake_Tekapo = col_double(),
##   Lake_Matahina = col_double(),
##   Lake_Dunstan = col_double(),
##   Lake_Wanaka = col_double()
## )
```

A) Using the ifelse function, create a new attribute in the data set called SEASON. Weeks 10– 22 should be Autumn, weeks 23–35 should be Winter, weeks 36–48 should be Spring and the remainder should be Summer. Once the attribute values have been set, convert it to a factor using the following command: inflows$SEASON =$ $factor(inflows$SEASON,levels=c(“Summer”,“Autumn”,“Winter”,“Spring”)) (For the purposes of these questions, assume that the weeks of the year can be assigned to seasons in this way.)

```
inflows$SEASON <- ifelse((inflows$WEEK >= 23 & inflows$WEEK <=35),
"Winter","Not")
inflows$SEASON <- ifelse((inflows$WEEK >= 36 & inflows$WEEK <=48),
"Spring",inflows$SEASON)
inflows$SEASON <- ifelse((inflows$WEEK >= 10 & inflows$WEEK <=22),
"Autumn",inflows$SEASON)
inflows$SEASON <- ifelse(inflows$SEASON == "Not", "Summer",inflows$SEASON)
inflows$SEASON =
factor(inflows$SEASON,levels=c("Summer","Autumn","Winter","Spring"))
```

B) Using the ggpairs function available from Canvas, plot a pairs plot comparing the inflows for the lakes, colouring the points based on the SEASON. Give an example of a highly correlated pair of lake inflows, and a pair that are not correlated. Comment on how the SEASON affects the inflows.

```
source("ggpairs.r")
ggpairs(inflows, contains("_"), color = SEASON)

## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(indices)` instead of `indices` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(indices2)` instead of `indices2` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```



There appears to be a strong correlation between the weekly inflows of Lake Taupo and Lake Matahina, and no correlation between the inflows of Lake Taupo and Lake Dunstad. The season appears to have a moderate impact on the inflows to the lakes, however the impact varies. In some lakes we see evidence that summer and Autumn inflows tend to be higher, while this is reversed in other lakes. Although it seems is likely that season influences inflow rate, it also appears that inlow is related to other confounding variable and therefore we cannot make a conclusion regarding the exact impact.

C) Use the pivot longer function to reshape the inflows data set so that it has a single INFLOW attribute for all reservoirs, you should set the names to attribute to be LAKE.

```
inflows_lakes <- pivot_longer(inflows, c(contains("Lake")), names_to =
"LAKE", values_to = "INFLOW")
```

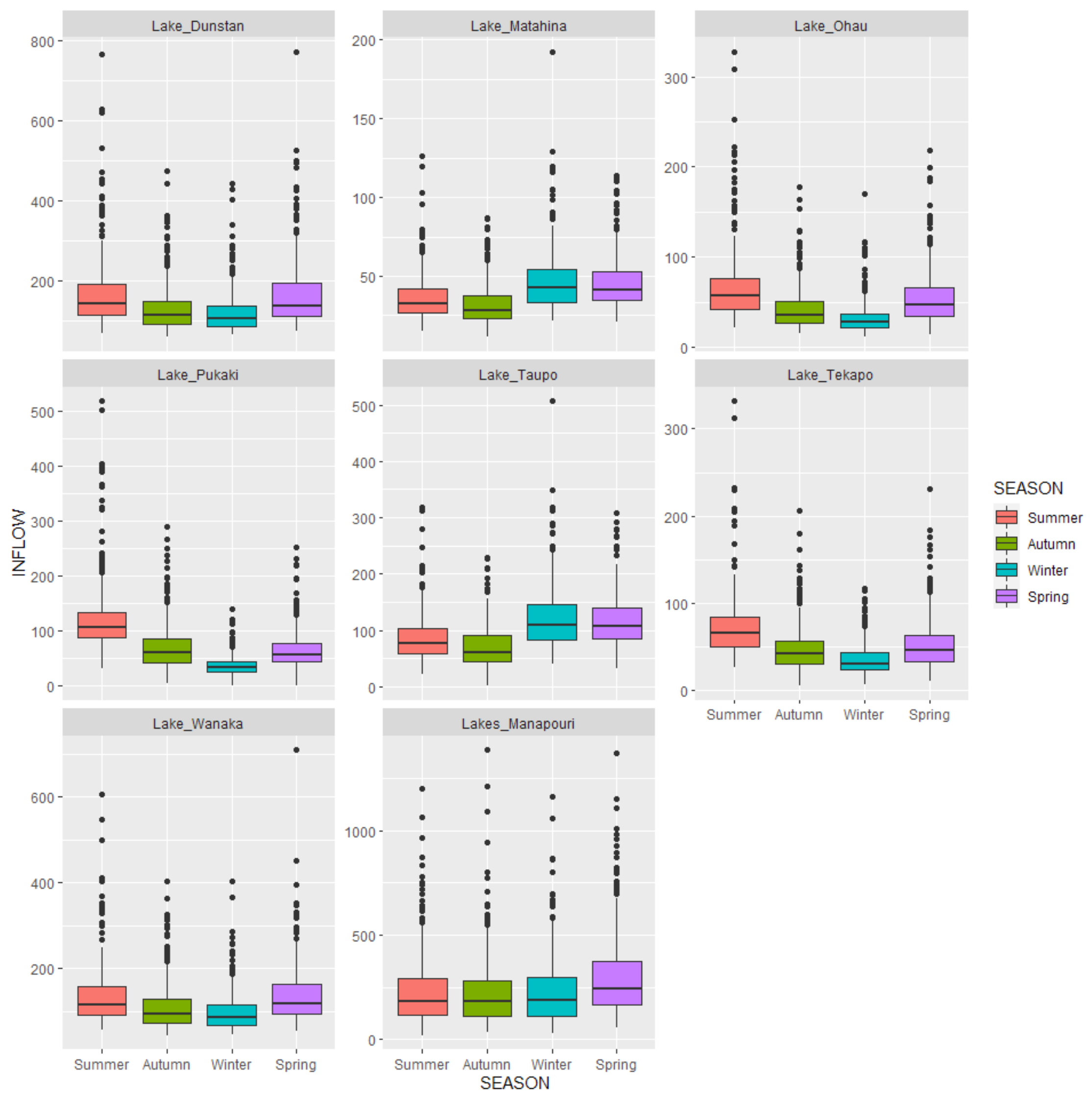
D) Use the group by(), summarise() and pivot wider() functions to display a table containing the average weekly inflows (in Mm3) LAKE and SEASON. Format the resulting data set so that the attributes are the seasons and there is a row for each LAKE.

```
inflows_lakes <- group_by(inflows_lakes,LAKE)
# summarise(inflows_lakes)
inflows_by_lakes <- inflows_lakes %>% pivot_wider(names_from = SEASON,
values_from = INFLOW) %>% group_by(LAKE) %>% summarise(SUMMER = mean(Summer,
na.rm = TRUE), AUTUMN = mean(Autumn, na.rm = TRUE), SPRING = mean(Spring,
na.rm = TRUE), WINTER = mean(Winter, na.rm = TRUE))
show(inflows_by_lakes)
```

```
## # A tibble: 8 x 5
##   LAKE             SUMMER AUTUMN SPRING WINTER
##   <chr>             <dbl>  <dbl>  <dbl>  <dbl>
## 1 Lake_Dunstan      166.   130.   168.   120.
## 2 Lake_Matahina      36.2   32.1   45.8   46.4
## 3 Lake_Ohau          66.2   42.6   54.7   33.1
## 4 Lake_Pukaki       124.    71.6   64.2   35.9
## 5 Lake_Taupo         85.2   70.4  117.   119.
## 6 Lake_Tekapo        73.5   47.7   53.0   35.6
## 7 Lake_Wanaka       136.   109.   140.    98.5
## 8 Lakes_Manapouri  234.   230.   301.   227.
```

F) Using facet wrap, create box plots for each LAKE showing the distribution of weekly hydro inflows for each season. Note that you will need to set scale="free y" due to the different scales of inflows into different lakes. Comment about the seasonal differences in inflows for the each of the lakes.

```
ggplot(inflows_lakes)+geom_boxplot(aes(x=SEASON, y = INFLOW, fill=SEASON)) +
facet_wrap(~LAKE, scale = "free_y")
```



Generally, inflows tended to decrease gradually as the seasons progressed (from summer - autumn - winter - spring), but the peak season was different for some of the lakes from the others. However, the variation in inflows between seasons seemed relatively constant for all lakes (with a few exceptions).

Lake Dunstan - the summer and spring average inflows were noticeably higher than the winter and autumn inflows.

Lake Matahina - the winter and spring average inflows were higher than the summer and autumn ones, with winter and autumn being slightly more extreme than summer and spring.

Lake Ohau - inflows tended to decrease from summer to winter, before recovering slightly in spring. Winter inflows showed markedly less variation.

Lake Pukaki - very similar to lake Ohau, inflows tended to decrease from summer through to winter before recovering in spring. Winter inflows showed less variation.

Lake Taupo - very similar to lake Matahina, with summer and autumn tending slightly lower than winter and spring. Again, winter and autumn had the highest and lowest mean.

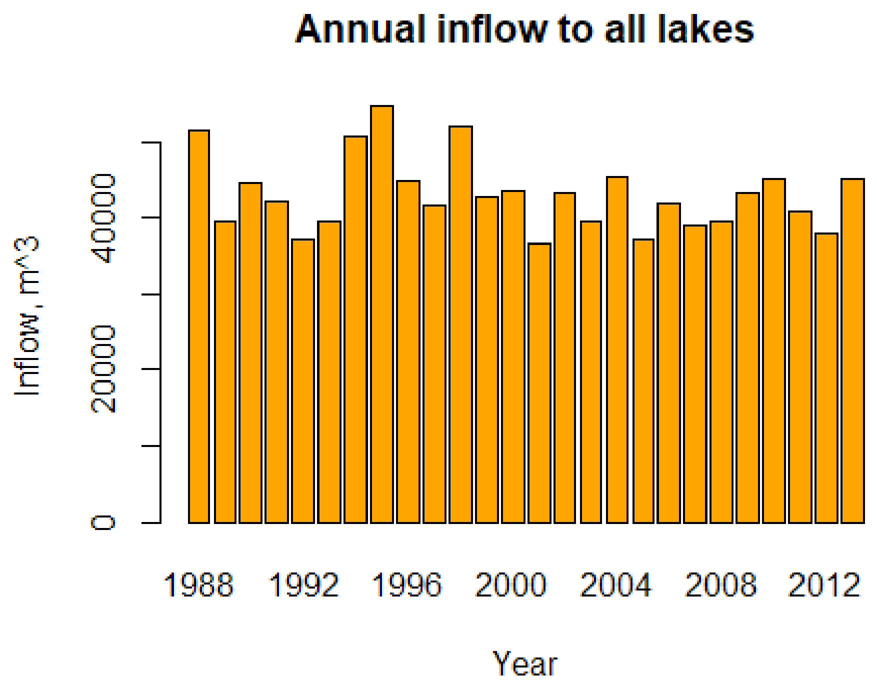
Lake Tekapo - very similar trend to lakes Ohau and Matahina Lake Wanaka - average inflow peaked in spring, before declining gradually year-round until spring.

Lake Manapouri - summer, autumn and winter had very similar inflow median and spread, while spring had slightly higher average inflow.

G) Create a column chart comparing the total annual inflows (into all lakes) for the 26 years. State two observations. The x-axis of the plot should be YEAR.

H)

```
annual_inflow <- inflows_lakes %>% group_by(YEAR) %>% summarise(TOTAL =
sum(INFLOW))
barplot(annual_inflow$TOTAL, main = "Annual inflow to all lakes", names.arg =
annual_inflow$YEAR, xlab = "Year", ylab = "Inflow, m^3", col = "orange")
```



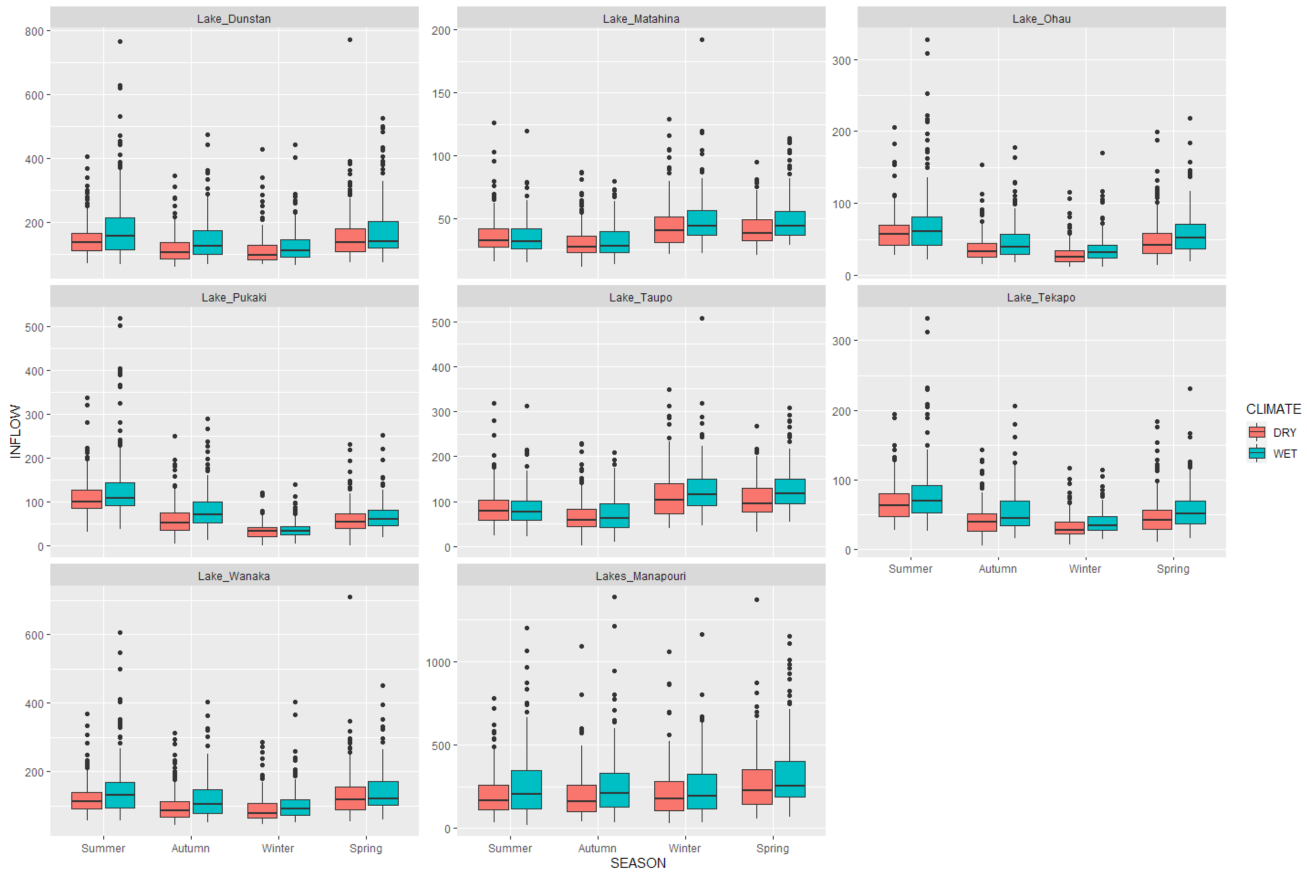
Total annual inflows average around 46,000 m^3 with fluctuations up to around 9,000. The highest annual inflow was about 55,000 in 1995 and the lowest of around 37,000 happened in 2005.

H) Using the ifelse() function, create a new attribute in the inflows data set called CLIMATE. For each (weekly) data point, set this attribute to "DRY" if the total inflows in that year are less than the mean and set it to "WET" if the total inflows are greater than (or equal to) the mean.

```
mean_total_inflow = mean(annual_inflow$TOTAL)
annual_inflow$CLIMATE = ifelse(annual_inflow$TOTAL < mean_total_inflow,
"DRY","WET")
inflows_lakes <- inner_join(annual_inflow, inflows_lakes, by = "YEAR")
```

I) Recreate the box plots from (f), as grouped box plots, by setting fill to be based on CLIMATE. State two observations.

```
ggplot(inflows_lakes)+geom_boxplot(aes(x=SEASON, y = INFLOW, fill=CLIMATE)) +
facet_wrap(~LAKE, scale = "free_y")
```



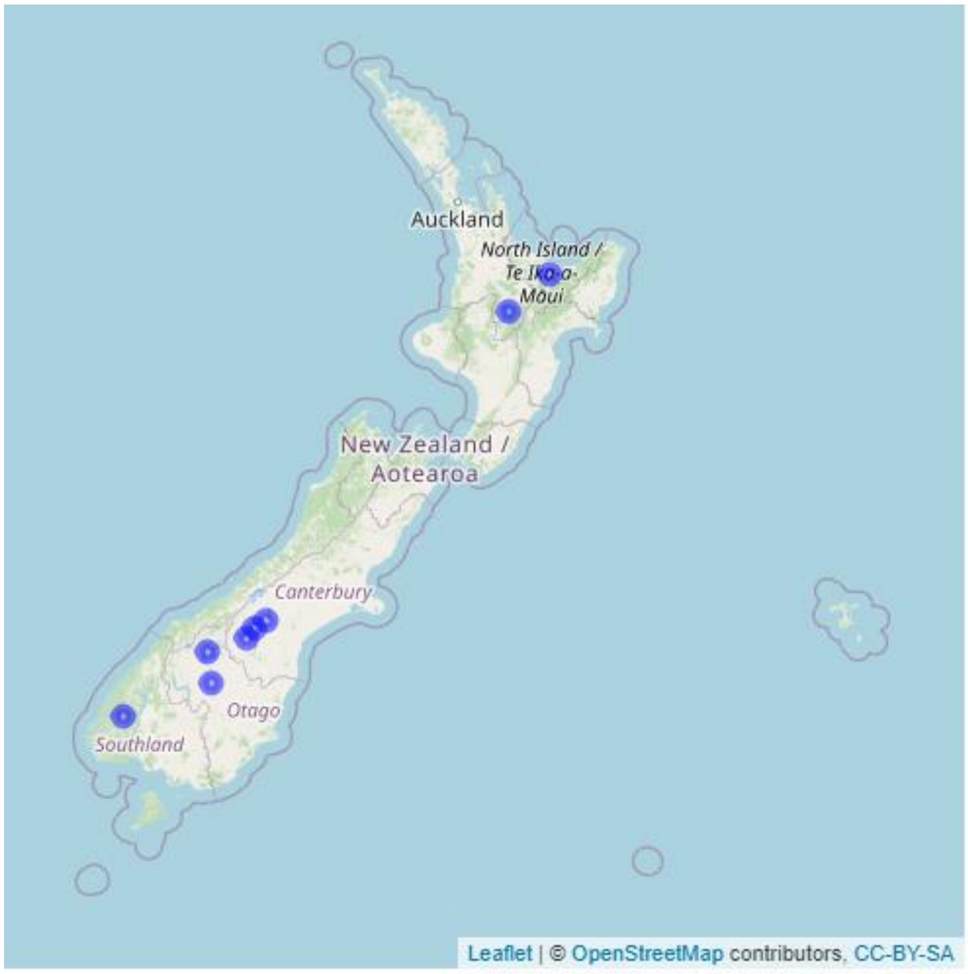
The spread of inflows during a wet climate was greater than the spread of inflows with a dry climate.

The average inflow for each lake + season during a wet climate also tended to be higher (as expected).

For the following questions, load the leaflet package, and import the locations.csv file.

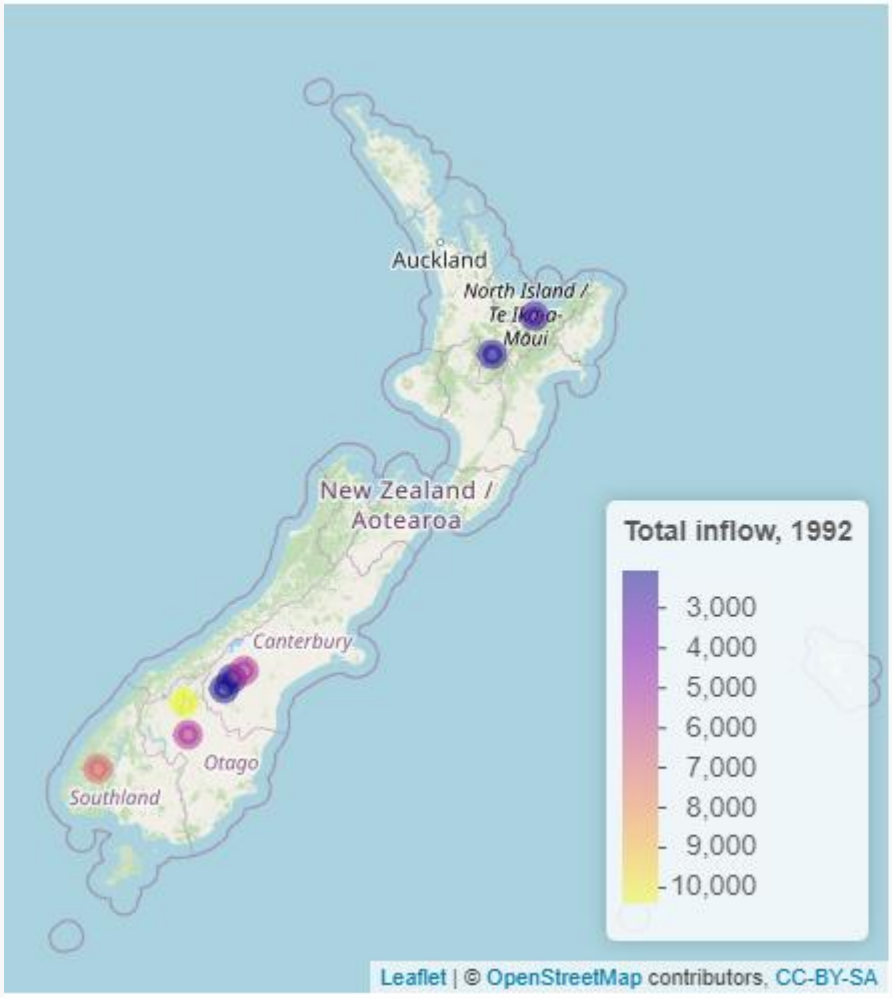
J) Using leaflet create a map of New Zealand, with a circle located at the coordinates for each lake (as given in the locations.csv file)

```
library(leaflet)
locations <- read_csv("locations.csv", col_names = TRUE)

##
## -- Column specification ---------------------------------------------------
------
## cols(
##   LAKES = col_character(),
##   LAT = col_double(),
##   LNG = col_double()
## )

map <- leaflet(locations) %>% addTiles() %>% addCircleMarkers(~LNG, ~LAT,
color = "blue", radius = 4)
show(map)
```


Auckland
North Island / Te Ika-a-Māui
New Zealand / Aotearoa
Canterbury
Otago
Southland
Leaflet | © OpenStreetMap contributors, CC-BY-SA

K) Modify the map created in (j) so that the colours of the circles are based on the total inflow into that lake in 1992; show a legend on the map.

```
lakes_1992 <- inflows_lakes %>% filter(YEAR == 1992) %>% group_by(LAKE) %>%
summarise(TOTAL = sum(INFLOW))
lake_pal <- colorNumeric( palette = "plasma", domain = lakes_1992$TOTAL)
map_proportional <- leaflet(locations) %>% addTiles() %>%
addCircleMarkers(~LNG, ~LAT, color = lake_pal(lakes_1992$TOTAL), fill = TRUE,
fillOpacity = 0.5, radius = 5) %>% addLegend(position = "bottomright", pal =
lake_pal, values = lakes_1992$TOTAL, title = "Total inflow, 1992")
show(map_proportional)
```

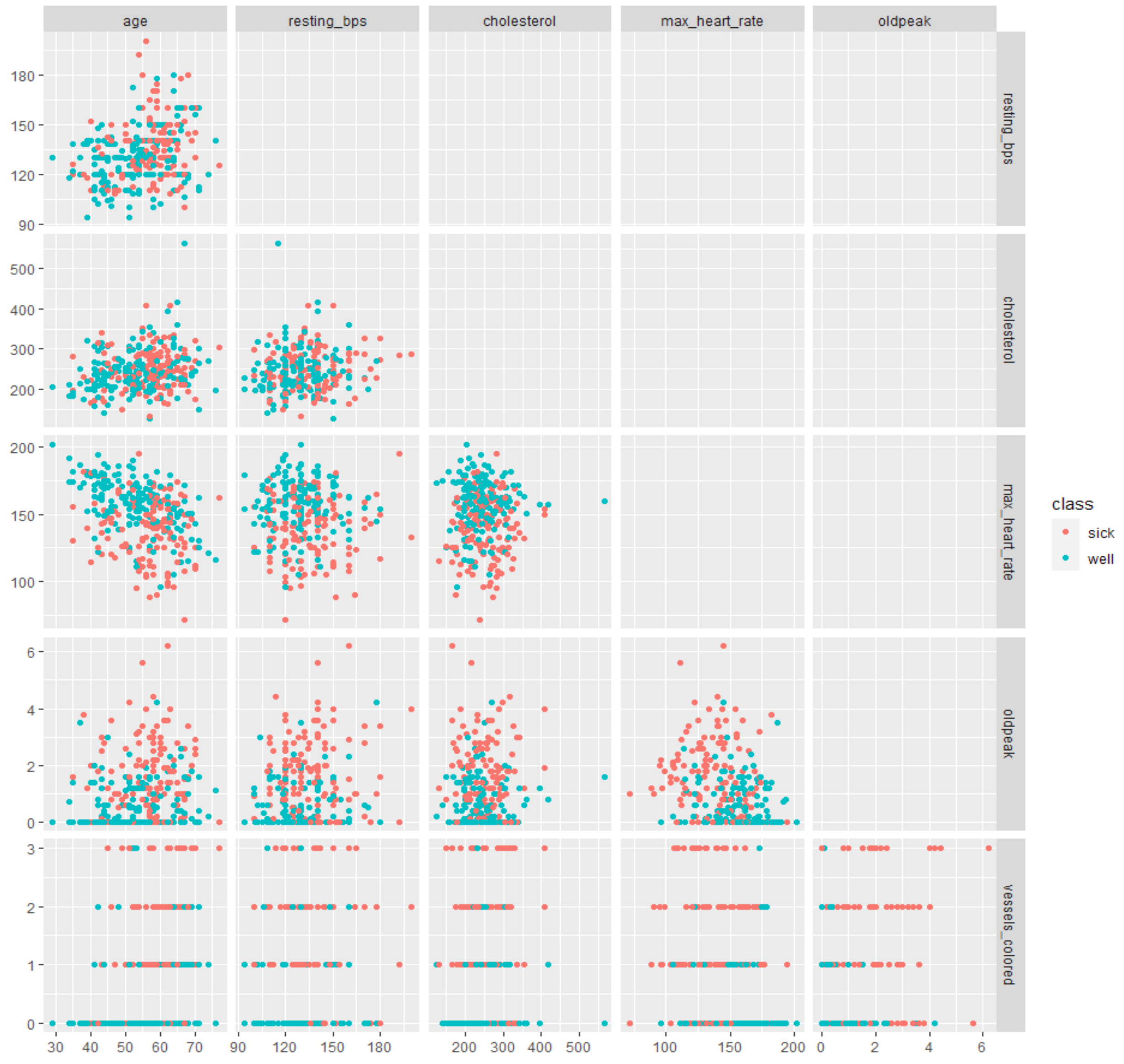


2. Disease Diagnosis

A) Create a ggpairs plots for the numeric attributes of the data set, colouring each point based on the class of the data point. Comment on any observations.

```
heart <- read_csv("heart.csv", col_names = TRUE)

##
## -- Column specification -------------------------------------------------
------
## cols(
##   age = col_double(),
##   sex = col_character(),
##   chest_pain_type = col_character(),
##   resting_bps = col_double(),
##   cholesterol = col_double(),
##   fasting_blood_sugar = col_logical(),
##   resting_ecg = col_character(),
##   max_heart_rate = col_double(),
##   exercise_induced_angina = col_logical(),
##   oldpeak = col_double(),
##   slope = col_character(),
##   vessels_colored = col_double(),
##   thal = col_character(),
##   class = col_character()
## )
```

```
ggpairs(heart, c("age","resting_bps",
"cholesterol","max_heart_rate","oldpeak","vessels_colored"), color = class)
```



The only clear observations at this stage are that those of class 'sick' tend to have higher values for vessels_colored and oldpeak, while the other variables are more mixed/random.

B) Set the seed of the random number generator to 100 (set.seed(100)), and then generate a training data set of 150 data points using the sample function (the remaining 153 data points will be the test set).

```
set.seed(120)
train = c(sample(1:303,150))
```

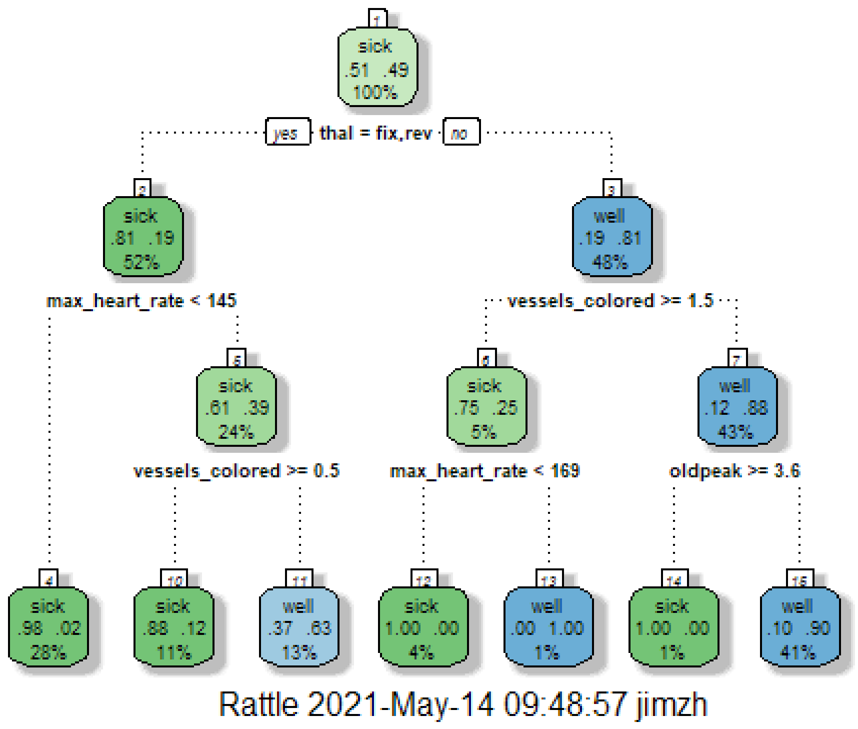
C) Using the rpart.control(...) arguments for rpart, set the termination criteria for generating the classification tree to be the max depth of the tree. (Disable any other termination criteria; i.e. set minsplit=1 and cp=0.)

i. Generate three classification trees to classify the Class as each benign or malignant, using all the other attributes as predictor variables, varying the maxdepth termination criterion through the values 3, 5 and 7.

```
library(rpart)
library(rattle)

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:rattle':
##
##     importance

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin

tree_depth3 <- rpart(class ~., data = heart, subset = train, method =
'class', control = rpart.control(minsplit = 1, cp = 0, maxdepth = 3))
tree_depth5 <- rpart(class ~., data = heart, subset = train, method =
'class', control = rpart.control(minsplit = 1, cp = 0, maxdepth = 5))
tree_depth7 <- rpart(class ~., data = heart, subset = train, method =
'class', control = rpart.control(minsplit = 1, cp = 0, maxdepth = 7))
```

ii. Visualise the tree with a maximum depth of 3.

```
fancyRpartPlot(tree_depth3)
```



iii. For each tree, give the in-sample and out-of-sample confusion matrices. –

- Depth 3

```
heart_predict3 = predict(tree_depth3, heart, type = "class")
table(Class=heart[train,]$class, "Prediction in-sample" =
heart_predict3[train])

##       Prediction in-sample
## Class  sick well
##   sick   64   13
##   well    3   70

table(Class=heart[-train,]$class, "Prediction out-sample" = heart_predict3[-
train])

##       Prediction out-sample
## Class  sick well
##   sick   38   23
##   well   14   78
```

- Depth 5

```
heart_predict5 = predict(tree_depth7, heart, type = "class")
table(Class=heart[train,]$class, "Prediction in-sample" =
heart_predict5[train])

##       Prediction in-sample
## Class  sick well
##    sick    63    9
##    well     4    74

table(Class=heart[-train,]$class, "Prediction out-sample" = heart_predict5[-
train])

##        Prediction out-sample
## Class   sick well
##    sick    53    13
##    well    16    71
```

- Depth 7

```
heart_predict7 = predict(tree_depth7, heart, type = "class")
table(Class=heart[train,]$class, "Prediction in-sample" =
heart_predict7[train])

##        Prediction in-sample
## Class   sick well
##    sick    69    3
##    well     3    75

table(Class=heart[-train,]$class, "Prediction out-sample" = heart_predict7[-
train])

##        Prediction out-sample
## Class   sick well
##    sick    56    10
##    well    21    66
```

iv. Create a table specifying the accuracy of each model, both in-sample and out-of-sample.

Accuracy:		
	IN_SAMPLE	OUT_OF_SAMPLE
Depth 3	$\frac{64+70}{150} = 0.89$	$\frac{38+78}{150} = 0.77$
Depth 5	$\frac{63+74}{150} = 0.91$	$\frac{53+71}{150} = 0.83$
Depth 7	$\frac{69+75}{150} = 0.96$	$\frac{56+66}{150} = 0.81$

V. Comment on and explain what you notice about the in-sample vs. out-of-sample accuracy.

The in-sample accuracy is higher. We used the training/in-sample data to build a model which fits it (fits the training data) and verified against the out of sample data - since the model was built with the training data as a target, it is unsurprising that it fits it better.

Interestingly, out-of-sample accuracy for depth 5 and 7 were very close and depth five was slightly more accurate – however, given the margin this is likely due to chance alone.

D) Set the termination criteria to be a max depth of 3 for the following question (i.e. set maxdepth=3, minsplit=1 and cp=0.)

i. By modifying the loss matrix, generate four classification trees (using all of the independent attributes), which range from having no false positives to no false negatives in the training data.

```
a<-100
b<-5
tree <- rpart(class ~., data = heart, subset = train, method = 'class',
control = rpart.control(minsplit = 1, cp = 0, maxdepth = 3), parms =
list(loss=matrix(c(0,a,b,0),nrow = 2)))
heart_predict = predict(tree, heart, type = "class")
table(Class=heart[train,]$class, "Prediction in-sample" =
heart_predict[train])
```

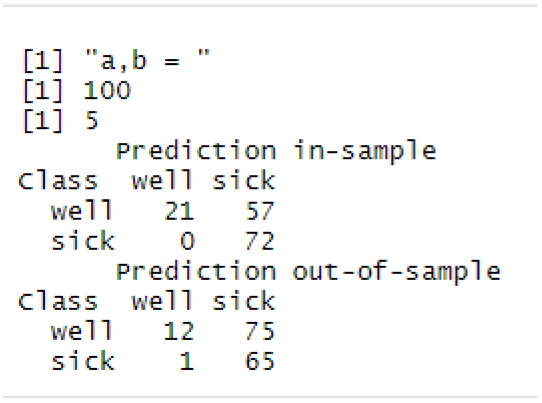
```
##       Prediction in-sample
## Class  sick well
##   sick   32   45
##   well    0   73
```

Where ‘a’ represents the penalty for a false negative and ‘b’ represents the penalty for a false positive.

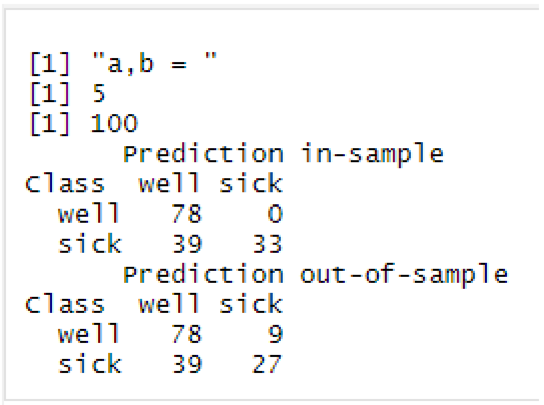
In the example above, the penalty for a false positive is much higher than the penalty for a false negative, and so the tree generates a model with no false positives. If we switched the variables around so that a = 100, b = 5, then the opposite would happen and a new tree that predicts no false negatives would be built.

If we merely wanted to weigh the model towards one end of Sensitivity vs Specificity, we could use values of a and b that land in the middle - for example, if we set a = 100, b = 30 we get a model with very few (but some) false negatives. If we set a = 30, b = 60 we get the opposite, a model with few (but some) false positives.

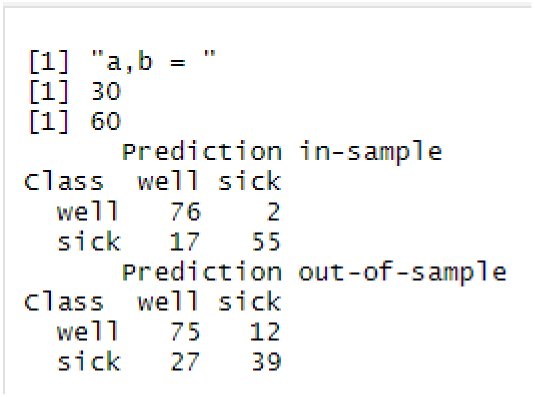
ii. For each tree, give the in-sample and out-of-sample confusion matrices.



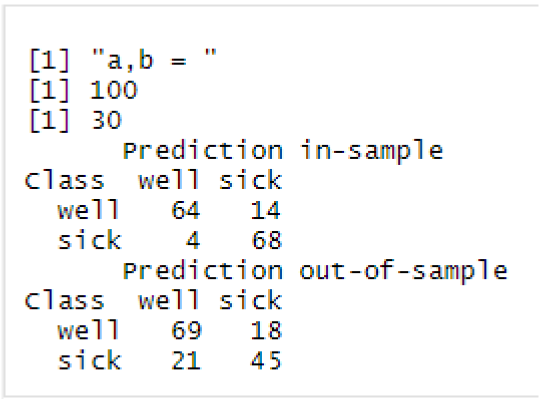
```
[1] "a,b = "
[1] 100
[1] 5
      Prediction in-sample
Class  well sick
  well   21   57
  sick    0   72
      Prediction out-of-sample
Class  well sick
  well   12   75
  sick    1   65
```



```
[1] "a,b = "
[1] 5
[1] 100
      Prediction in-sample
Class  well sick
  well   78    0
  sick   39   33
      Prediction out-of-sample
Class  well sick
  well   78    9
  sick   39   27
```



```
[1] "a,b = "
[1] 30
[1] 60
      Prediction in-sample
Class  well sick
  well   76    2
  sick   17   55
      Prediction out-of-sample
Class  well sick
  well   75   12
  sick   27   39
```

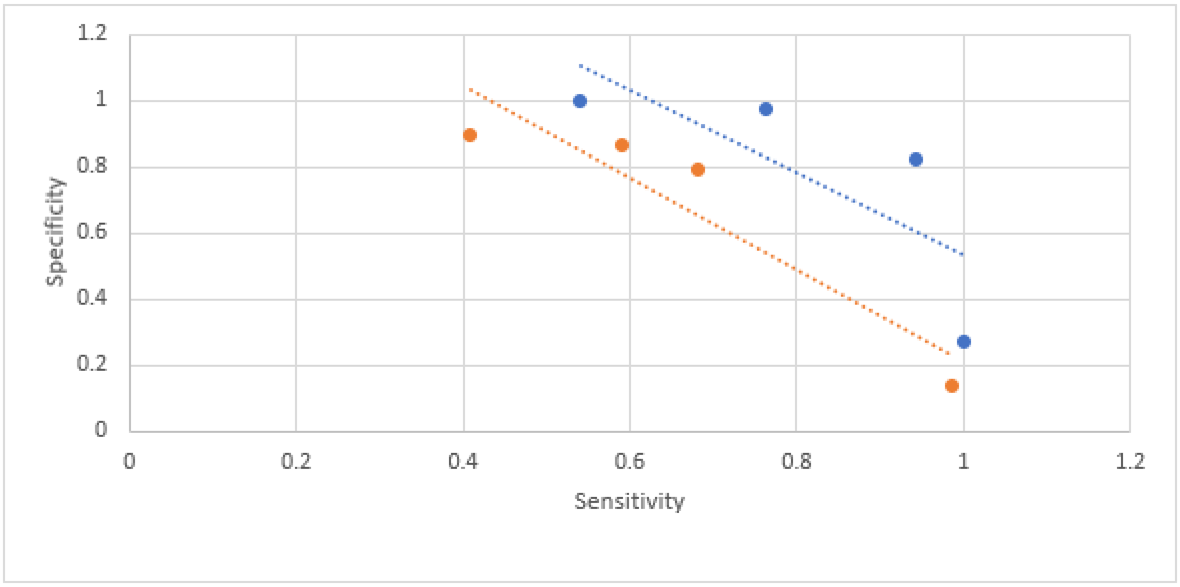


```
[1] "a,b = "
[1] 100
[1] 30
      Prediction in-sample
Class  well sick
  well   64   14
  sick    4   68
      Prediction out-of-sample
Class  well sick
  well   69   18
  sick   21   45
```

iii. On a 2D scatterplot show the sensitivity vs. specificity of each classification model, include both the in-sample and out-of-sample values in different colours. (This plot can be generated in R, Matlab or Excel.)

In our tables, a return of 'positive' is defined as identifying a student as sick.

	In-sample		Out-of-sample	
	SENSITIVITY	SPECIFICITY	SENSITIVITY	SPECIFICITY
A = 100, B = 5	1	0.27	0.985	0.138
A = 5, B = 100	0.541	1	0.409	0.897
30,60	0.764	0.974	0.591	0.862
100,30	0.944	0.821	0.682	0.793



iv. Comment on and explain any observations about the in-sample vs out-of-sample performance seen in the plot.

As expected, we see that sensitivity and specificity are roughly inversely correlated – as one increases, the other decreases.

We also see that both sensitivity and specificity tend more towards the center(0.5)/towards being equal when the model is applied to out-of-sample data. This is probably because, as explained before, out-of-sample accuracy is normally lower, meaning that the ideal results are more diluted by random chance. This random chance affects our attempt to bias the model towards one extreme or the other, skewing the model more toward the 'average' values (midway).

v. Suppose that we are much more concerned about false negatives than false positives. Explain what this means, and then recommend which of the classification models (from above) that we should choose.

This means that we are more concerned about accidentally putting a healthy student in the sick category than we are putting a sick student in the healthy category. If we are much more concerned about false negatives, we should assign the penalty for them much higher to prevent any occurrences of false negatives - accidentally classing a healthy student as sick. By assigning a harsh penalty to this mistake we teach the model to avoid it.

Of the options above, in this example I would recommend the case where a = 100, b = 5, which results in very few false negatives as desired.

E) Using the heart data set, create random forests, with ntree set to 10, 50 and 250, to predict the class attribute, given the other attributes. Create 5 random forests for each ntree value.

Note the following: • You must first convert the class attribute to a factor.

• Report the 5 OOB (out-of-bag / out-of-sample) estimates of error rate, for each random forest size (10, 50, 250).

• Comment on your results.

```
heart$class = factor(heart$class, levels =c("well","sick"))
heart$sex = factor(heart$sex, levels =c("male","fem"))
heart$chest_pain_type = factor(heart$chest_pain_type, levels
=c("angina","asympt","notang","abnang"))
heart$resting_ecg = factor(heart$resting_ecg, levels =c("abn","norm","hyp"))
heart$slope = factor(heart$slope, levels =c("down","flat","up"))
heart$thal = factor(heart$thal, levels =c("fix","norm","rev"))

# Ntree = ten, five trials
randomForest(class ~ ., data = heart, ntree = 10, subset = train)

##
## Call:
```

```
##  randomForest(formula = class ~ ., data = heart, ntree = 10, subset =
train)
##                Type of random forest: classification
##                      Number of trees: 10
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 23.49%
## Confusion matrix:
##      well sick class.error
## well   58   15   0.2054795
## sick   20   56   0.2631579
```

```
randomForest(class ~ ., data = heart, ntree = 10, subset = train)
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = heart, ntree = 10, subset =
train)
##                Type of random forest: classification
##                      Number of trees: 10
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 18.67%
## Confusion matrix:
##      well sick class.error
## well   61   12   0.1643836
## sick   16   61   0.2077922
```

```
randomForest(class ~ ., data = heart, ntree = 10, subset = train)
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = heart, ntree = 10, subset =
train)
##                Type of random forest: classification
##                      Number of trees: 10
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 26.17%
## Confusion matrix:
##      well sick class.error
## well   54   18   0.2500000
## sick   21   56   0.2727273
```

```
randomForest(class ~ ., data = heart, ntree = 10, subset = train)
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = heart, ntree = 10, subset =
train)
##                Type of random forest: classification
##                      Number of trees: 10
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 24.32%
## Confusion matrix:
##      well sick class.error
## well   56   15   0.2112676
## sick   21   56   0.2727273
```

```
randomForest(class ~ ., data = heart, ntree = 10, subset = train)
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = heart, ntree = 10, subset =
train)
##                Type of random forest: classification
##                      Number of trees: 10
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 23.49%
## Confusion matrix:
##      well sick class.error
## well   54   18   0.2500000
## sick   17   60   0.2207792
```

```
# Ntree = 50, five trials
randomForest(class ~ ., data = heart, ntree = 50, subset = train)
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = heart, ntree = 50, subset =
train)
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 19.33%
## Confusion matrix:
##      well sick class.error
## well   61   12   0.1643836
## sick   17   60   0.2207792
```

```
randomForest(class ~ ., data = heart, ntree = 50, subset = train)
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = heart, ntree = 50, subset =
train)
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 3
```

```
##
##         OOB estimate of  error rate: 15.33%
## Confusion matrix:
##      well sick class.error
## well   62   11   0.1506849
## sick   12   65   0.1558442
```

```
randomForest(class ~ ., data = heart, ntree = 50, subset = train)
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = heart, ntree = 50, subset =
train)
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 14%
## Confusion matrix:
##      well sick class.error
## well   66    7  0.09589041
## sick   14   63  0.18181818
```

```
randomForest(class ~ ., data = heart, ntree = 50, subset = train)
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = heart, ntree = 50, subset =
train)
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 14.67%
## Confusion matrix:
##      well sick class.error
## well   64    9   0.1232877
## sick   13   64   0.1688312
```

```
randomForest(class ~ ., data = heart, ntree = 50, subset = train)
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = heart, ntree = 50, subset =
train)
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 16.67%
## Confusion matrix:
```

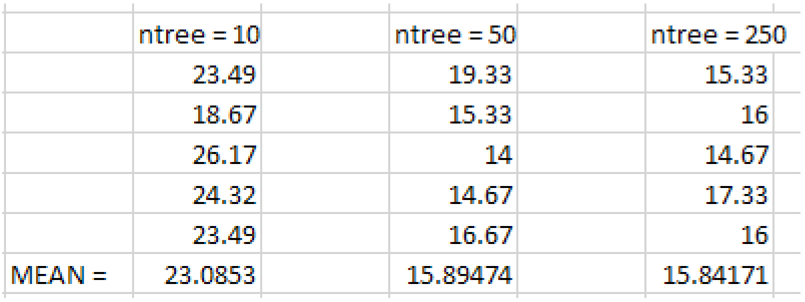
```
##      well sick class.error
## well   60   13   0.1780822
## sick   12   65   0.1558442

# Ntree = 250, five trials
randomForest(class ~ ., data = heart, ntree = 250, subset = train)

##
## Call:
##  randomForest(formula = class ~ ., data = heart, ntree = 250,      subset
= train)
##                Type of random forest: classification
##                      Number of trees: 250
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 15.33%
## Confusion matrix:
##      well sick class.error
## well   64    9   0.1232877
## sick   14   63   0.1818182

randomForest(class ~ ., data = heart, ntree = 250, subset = train)

##
## Call:
##  randomForest(formula = class ~ ., data = heart, ntree = 250,      subset
= train)
##                Type of random forest: classification
##                      Number of trees: 250
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 16%
## Confusion matrix:
##      well sick class.error
## well   63   10   0.1369863
## sick   14   63   0.1818182

randomForest(class ~ ., data = heart, ntree = 250, subset = train)

##
## Call:
##  randomForest(formula = class ~ ., data = heart, ntree = 250,      subset
= train)
##                Type of random forest: classification
##                      Number of trees: 250
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 14.67%
## Confusion matrix:
##      well sick class.error
## well   63   10   0.1369863
## sick   12   65   0.1558442
```

```r
randomForest(class ~ ., data = heart, ntree = 250, subset = train)
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = heart, ntree = 250,      subset
= train)
##                Type of random forest: classification
##                      Number of trees: 250
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 17.33%
## Confusion matrix:
##      well sick class.error
## well   62   11   0.1506849
## sick   15   62   0.1948052
```

```r
randomForest(class ~ ., data = heart, ntree = 250, subset = train)
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = heart, ntree = 250,      subset
= train)
##                Type of random forest: classification
##                      Number of trees: 250
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 16%
## Confusion matrix:
##      well sick class.error
## well   63   10   0.1369863
## sick   14   63   0.1818182
```



	ntree = 10		ntree = 50		ntree = 250
	23.49		19.33		15.33
	18.67		15.33		16
	26.17		14		14.67
	24.32		14.67		17.33
	23.49		16.67		16
MEAN =	23.0853		15.89474		15.84171

As you can see from this table of OOB error estimates, for my random seed, the OOB estimate for error rate decreases as ntree increases. This makes sense, since with more trees the error of each can be overcome by the plurality of the class predictions and the overall average accuracy rate. We also see the effects of dimishing returns between 50 and 250 trees.

We now wish to see if the Na¨ıve Bayes classifier can also be used to predict the correct diagnosis for the heart data set. However, na¨ıve Bayes requires discrete data, so we have provided a modified version of the heart data set: heart-discrete.csv, available on Canvas. (In this data set we have binned the continuous attributes into ranges.) Divide your data set into training and test data, using the same procedure (and random seed) as (c).

F) Apply the na¨ıve Bayes method to the training data set to determine the class using all the other attributes.

```
library(klaR)

## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

set.seed(120)
heart_discrete <- read_csv("heart_discrete.csv", col_names = TRUE)

##
## -- Column specification --------------------------------------------------
------
## cols(
##   age = col_character(),
##   sex = col_character(),
##   chest_pain_type = col_character(),
##   resting_bps = col_character(),
##   cholesterol = col_character(),
##   fasting_blood_sugar = col_logical(),
##   resting_ecg = col_character(),
##   max_heart_rate = col_character(),
##   exercise_induced_angina = col_logical(),
##   oldpeak = col_character(),
##   slope = col_character(),
##   vessels_colored = col_double(),
##   thal = col_character(),
##   class = col_character()
## )

train = c(sample(1:303,150))

heart_nb <- NaiveBayes(as_factor(class) ~ ., data = heart_discrete, subset =
train)
```

G) Using the predict() function, determine the in-sample and out-of-sample accuracy for this method. (R will report some warnings, but you can ignore them. However, if you wish to using R markdown, you will need to use the suppressWarnings(...) function around the predict function.)

```
heart_prednb = suppressWarnings(predict(heart_nb,
heart_discrete[1:13])$class)

table(Class = heart_discrete$class[train], "Prediction, in-sample" =
heart_prednb[train])
```

```
##       Prediction, in-sample
## Class  well sick
##   sick   18   54
##   well   65   13
```

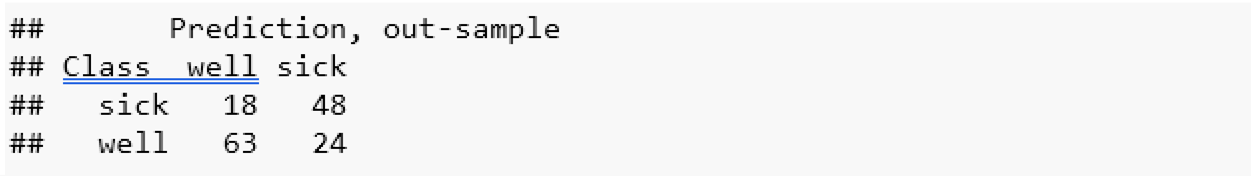
```
table(Class = heart_discrete$class[-train], "Prediction, out-sample" =
heart_prednb[-train])
```

```
##       Prediction, out-sample
## Class  well sick
##   sick   18   48
##   well   63   24
```

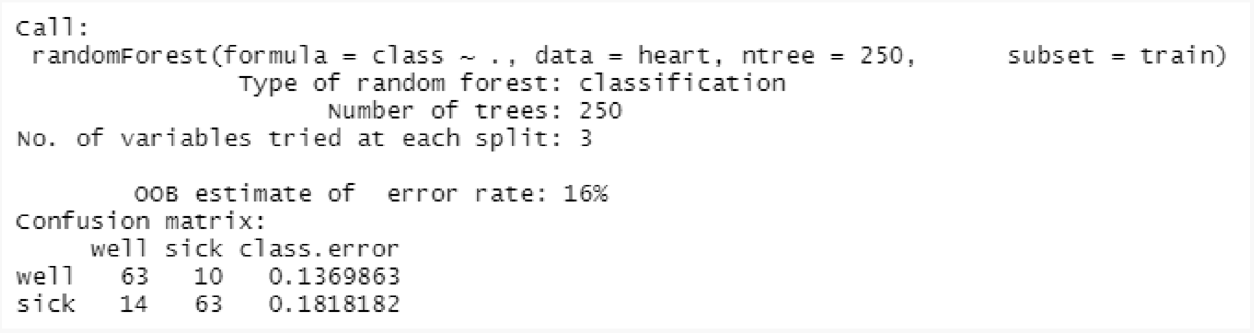
In-sample – the accuracy of the naïve bayes model was $\frac{65+54}{150} = 0.79$ or 79%.

Out-of-sample – the accuracy of the naïve bayes model was $\frac{63+48}{150} = 0.74$ or 74%.

H) Show the confusion matrix for the out-of-sample predictions, above, and discuss this in comparison to the corresponding confusion matrix for the random forest with 250 trees.



```
##       Prediction, out-sample
## Class  well sick
##   sick   18   48
##   well   63   24
```



```
Call:
 randomForest(formula = class ~ ., data = heart, ntree = 250,      subset = train)
               Type of random forest: classification
                     Number of trees: 250
No. of variables tried at each split: 3

        OOB estimate of  error rate: 16%
Confusion matrix:
     well sick class.error
well   63   10   0.1369863
sick   14   63   0.1818182
```

The confusion matrix for the naïve bayes method (above) had an accuracy rate of 74%, and therefore an error rate of 26%. Compared to the confusion table for the random forest model, with an error rate of 16%, we can see that the random forest method is in general far more accurate.

We can further compare the in-sample confusion table of the forest method with the in-sample table for the naïve bayes method to see that (14+10 vs 18+13) it is also much more accurate.