Coffee

Sean Cerulean Johnson

2022-04-03

Library

```r
library(Thematic)
```

```
## Warning: replacing previous import 'magrittr::set_names' by 'rlang::set_names'
## when loading 'Thematic'
```

```r
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.1.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.1.3
```

Data : Importing and Cleaning

From TidyTuesday URL:https://github.com/rfordatascience/tidytuesday/tree/master/data/2020/2020-07-07

> Note: within the above link, there was already some pre-processing done to the data with the column and value names.

Quick Overview

```
summary<-summary(coffee_ratings)
tibble(summary)%>%
  tabGT()
```

Quite a few NA's. Numerical Columns: 1 within quakers, and 230 in Altitude low/high/mean. Next, need to check what is happening in the rest of the data set, the character type. # Data Wrangling ## NA's per coloumn

```
apply(X=is.na(coffee_ratings), MARGIN = 2, FUN = sum)
```

```
##      total_cup_points               species                 owner
##                     0                     0                     7
##     country_of_origin             farm_name            lot_number
##                     1                   359                  1063
##                  mill            ico_number               company
##                   315                   151                   209
##              altitude                region              producer
##                   226                    59                   231
##        number_of_bags            bag_weight    in_country_partner
##                     0                     0                     0
##          harvest_year          grading_date               owner_1
##                    47                     0                     7
##               variety     processing_method                 aroma
##                   226                   170                     0
##                flavor            aftertaste               acidity
##                     0                     0                     0
##                  body               balance            uniformity
##                     0                     0                     0
##             clean_cup             sweetness         cupper_points
##                     0                     0                     0
##              moisture  category_one_defects               quakers
##                     0                     0                     1
##                 color  category_two_defects            expiration
##                   218                     0                     0
##    certification_body certification_address certification_contact
##                     0                     0                     0
##   unit_of_measurement    altitude_low_meters   altitude_high_meters
##                     0                   230                   230
##  altitude_mean_meters
##                   230
```

There a quite a few missing values overall and many columns have many. I will be just removing some of the columns with too many missing values, for instance lot_number and farm_name. Additionally, I there will be removal of columns that do not heavily influence the goals of this project. While there are a few methods for replacing values within the dataset, my first approach is to understand the set in the instance if all values were correct. After that analysis, to evaluate if an how to replace missing values.

Removal of columns

```r
coffee = coffee_ratings%>%
  select(-farm_name,-lot_number,-mill,-ico_number,-altitude,
         -altitude_low_meters,-altitude_high_meters,-producer,-company,
         -expiration,-certification_address,-owner_1,-grading_date,
         -certification_contact,-unit_of_measurement)
```

Removal of Remaining NA's

```r
coffee = na.omit(coffee)
```

Alter Units Notations

Bag Weight

```r
#selecting only items with lbs pattern within column to see how many
#Nathan F reminded me to the use of grep
coffee[grep("lbs",coffee$bag_weight),]
```

```
## # A tibble: 18 x 28
##    total_~1 species owner count~2 region numbe~3 bag_w~4 in_co~5 harve~6 variety
##       <dbl> <chr>   <chr> <chr>   <chr>    <dbl> <chr>   <chr>   <chr>   <chr>
##  1     87.2 Arabica the ~ Costa ~ san r~     250 3 lbs   Specia~ 2014    Caturra
##  2     86.3 Arabica fran~ Costa ~ west ~     250 2 lbs   Specia~ 2015/2~ Caturra
##  3     85.3 Arabica the ~ Costa ~ west ~     250 3 lbs   Specia~ 2014    Caturra
##  4     85.3 Arabica the ~ Costa ~ san r~     250 3 lbs   Specia~ 2014    Caturra
##  5     84.7 Arabica fabi~ Costa ~ tarra~      50 1 lbs   Specia~ 2014    Caturra
##  6     84.5 Arabica fabi~ Costa ~ tarra~     250 1 lbs   Specia~ 2014    Caturra
##  7     83.8 Arabica germ~ United~ yauco~      18 5 lbs   Specia~ 2013    Other
##  8     83.8 Arabica the ~ Guatem~ quetz~     250 3 lbs   Specia~ 2012    Caturra
##  9     83.3 Arabica the ~ Costa ~ san r~     250 3 lbs   Specia~ 2014    Caturra
## 10     83.3 Arabica itia~ Haiti   thiot~       2 4 lbs   Specia~ 2012    Typica
## 11     83   Arabica germ~ United~ yauco~      17 5 lbs   Specia~ 2013    Other
## 12     81.5 Arabica myri~ Haiti   dondo~     300 4 lbs   Specia~ 2013    Blue M~
## 13     81.2 Arabica esse~ Guatem~ huehu~      36 55 lbs  Blosso~ 2014    Pacama~
## 14     81.1 Arabica germ~ United~ yauco~      18 5 lbs   Specia~ 2013    Other
## 15     80.9 Arabica chri~ Nicara~ matag~     275 1 lbs   Specia~ 2013    Caturra
## 16     80.8 Arabica the ~ Costa ~ san r~     250 3 lbs   Specia~ 2014    Caturra
## 17     79.3 Arabica the ~ Colomb~ perei~     250 3 lbs   Specia~ 2013    Caturra
## 18     79.1 Arabica germ~ United~ yauco~      18 5 lbs   Specia~ 2013    Other
## # ... with 18 more variables: processing_method <chr>, aroma <dbl>,
## #   flavor <dbl>, aftertaste <dbl>, acidity <dbl>, body <dbl>, balance <dbl>,
## #   uniformity <dbl>, clean_cup <dbl>, sweetness <dbl>, cupper_points <dbl>,
## #   moisture <dbl>, category_one_defects <dbl>, quakers <dbl>, color <chr>,
## #   category_two_defects <dbl>, certification_body <chr>,
## #   altitude_mean_meters <dbl>, and abbreviated variable names
## #   1: total_cup_points, 2: country_of_origin, 3: number_of_bags, ...
```

```r
#separating out the columns based on the value and units associated with it
coffee = tidyr::separate(data = coffee, col = bag_weight, into = c("weight", "type"), sep = " ")
```

```r
#converted string to numeric
coffee$weight = as.numeric(coffee$weight)
```

```r
#simple loop to change units
for(i in 1:length(coffee)){
  if(coffee[i,8]=="kg"){
  coffee[i,7] = round(coffee[i,7] * 2.20462,0)
  coffee[i,8] = "lbs"
  }
}
```

```r
#remove type column as the weight col is uniform for unit type
coffee = coffee%>%
  select(-type)
```

Altitude

```r
#Note: If reshape lib is on, this will break
coffee = coffee%>%rename(avg_altitude=altitude_mean_meters)
coffee$avg_altitude = round(coffee$avg_altitude * 3.28084,0)
```

Years

Here there were years in the form of Year1/Year2, the following will be changing year to the initial year (Year1)

```r
coffee$harvest_year = substr(coffee$harvest_year,1,4)
coffee$harvest_year = as.numeric(coffee$harvest_year)
```

The above chunk was done do to the initial inception of that batch of coffee.

#Numerical Summary

```r
summary(coffee[,c(9,12:24,26,28)])
```

```
##   harvest_year      aroma          flavor        aftertaste       acidity
##  Min.   :2011   Min.   :5.080   Min.   :6.170   Min.   :6.170   Min.   :5.250
##  1st Qu.:2012   1st Qu.:7.420   1st Qu.:7.330   1st Qu.:7.170   1st Qu.:7.330
##  Median :2014   Median :7.580   Median :7.500   Median :7.420   Median :7.500
##  Mean   :2014   Mean   :7.559   Mean   :7.504   Mean   :7.374   Mean   :7.515
##  3rd Qu.:2015   3rd Qu.:7.750   3rd Qu.:7.670   3rd Qu.:7.580   3rd Qu.:7.670
##  Max.   :2018   Max.   :8.750   Max.   :8.670   Max.   :8.500   Max.   :8.580
##       body          balance        uniformity       clean_cup
##  Min.   :6.330   Min.   :6.080   Min.   : 6.000   Min.   : 0.000
##  1st Qu.:7.330   1st Qu.:7.330   1st Qu.:10.000   1st Qu.:10.000
##  Median :7.500   Median :7.500   Median :10.000   Median :10.000
```

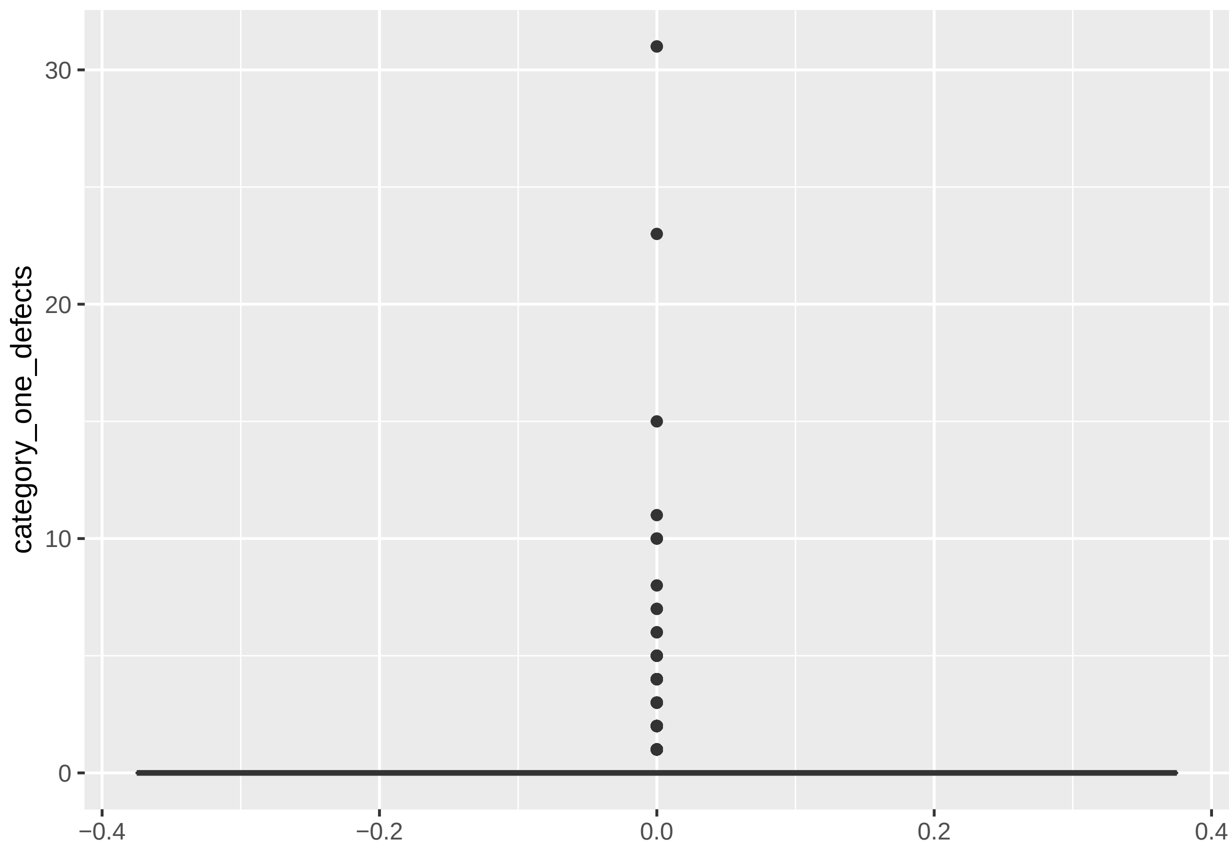
```
##  Mean   :7.494   Mean   :7.488   Mean   : 9.871   Mean   : 9.849  
##  3rd Qu.:7.670   3rd Qu.:7.670   3rd Qu.:10.000   3rd Qu.:10.000  
##  Max.   :8.420   Max.   :8.580   Max.   :10.000   Max.   :10.000  
##    sweetness     cupper_points      moisture       category_one_defects
##  Min.   : 1.33   Min.   :5.170   Min.   :0.00000   Min.   : 0.0000     
##  1st Qu.:10.00   1st Qu.:7.250   1st Qu.:0.10000   1st Qu.: 0.0000     
##  Median :10.00   Median :7.500   Median :0.11000   Median : 0.0000     
##  Mean   : 9.93   Mean   :7.459   Mean   :0.09737   Mean   : 0.4262     
##  3rd Qu.:10.00   3rd Qu.:7.670   3rd Qu.:0.12000   3rd Qu.: 0.0000     
##  Max.   :10.00   Max.   :8.580   Max.   :0.17000   Max.   :31.0000     
##     quakers        category_two_defects  avg_altitude   
##  Min.   : 0.0000   Min.   : 0.000       Min.   :     3  
##  1st Qu.: 0.0000   1st Qu.: 0.000       1st Qu.:  3609  
##  Median : 0.0000   Median : 2.000       Median :  4300  
##  Mean   : 0.1521   Mean   : 3.822       Mean   :  6145  
##  3rd Qu.: 0.0000   3rd Qu.: 5.000       3rd Qu.:  5249  
##  Max.   :11.0000   Max.   :47.000       Max.   :623898  
```

The parameters for defects, quakers, and average altitude seem to have quite a range for values. Additionally, it can be seen for these fields that the max points are quite a ways away from the mean.

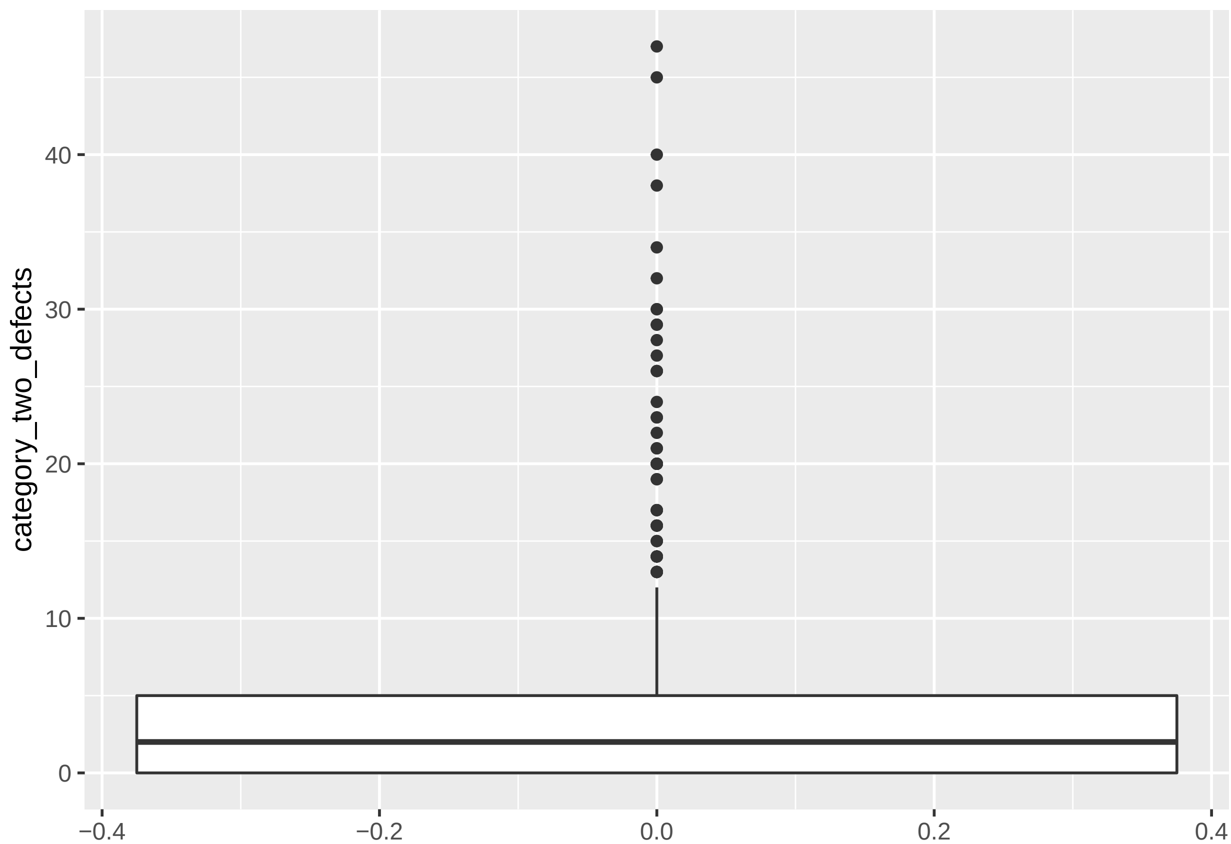
EDA / Visuals

Outliers Check

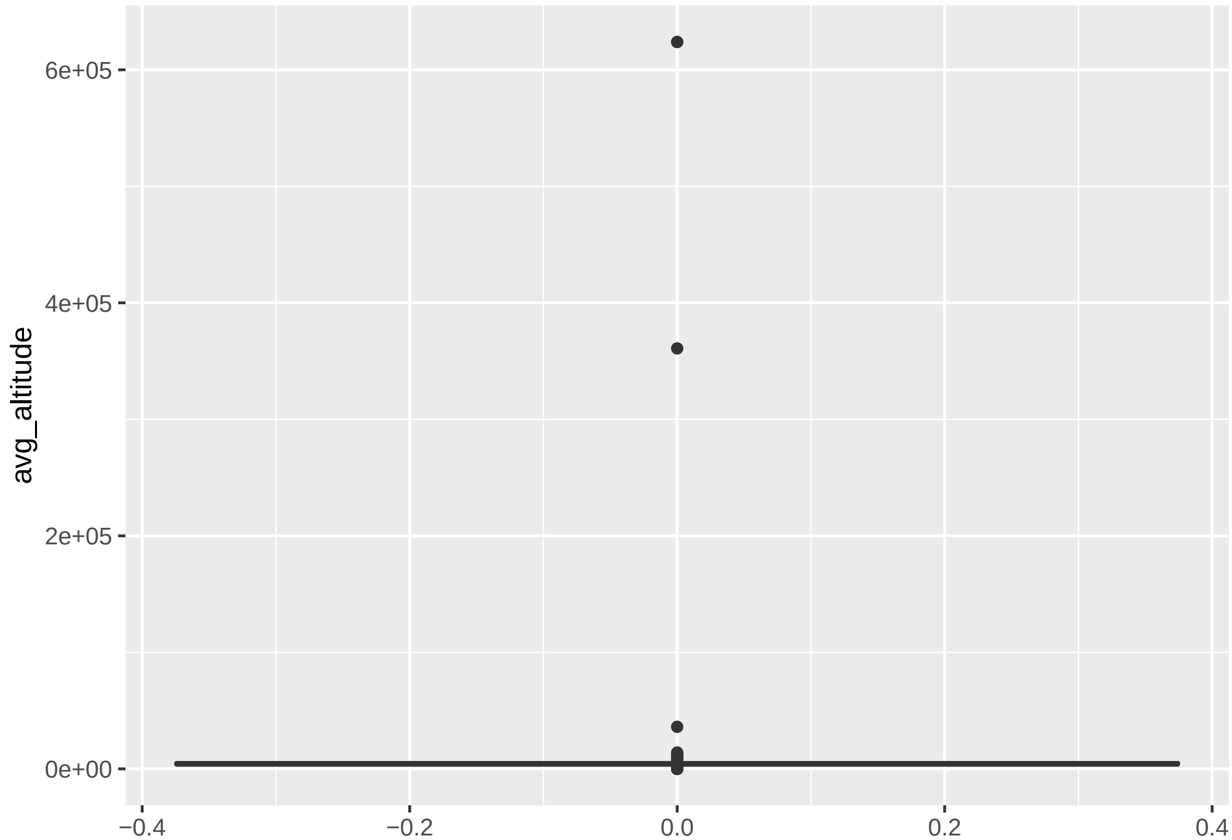
```
defect1_plt = ggplot(coffee, aes(y=category_one_defects)) +
              geom_boxplot()
defect2_plt = ggplot(coffee, aes(y=category_two_defects)) +
              geom_boxplot()
alt_plt = ggplot(coffee, aes(y=avg_altitude)) +
              geom_boxplot()
quakers = ggplot(coffee, aes(y=quakers)) +
              geom_boxplot()

defect1_plt
```



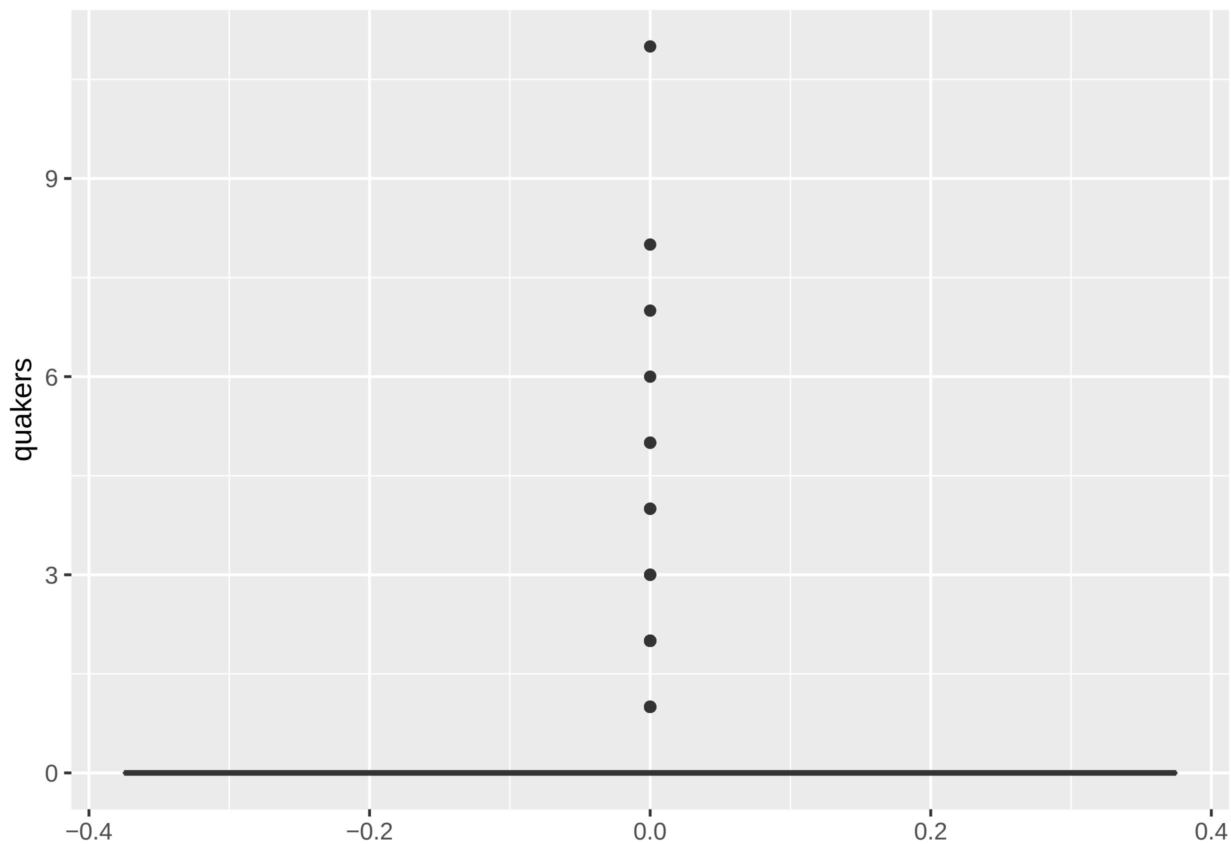

```
defect2_plt
```




```
alt_plt
```




```
quakers
```



There are some outliers, but not that many that would result in a concern at this time. These fields may be removed from the current analysis due to the outliers and lack of variance within the data. As the majority of these values are 0. This will be removed in the upcoming data chunks. Additionally, as this project is to have more focus in analysis, there will be additional removal of fields. Specifically, the ownership items and their location details.

Redefine DF for Visuals

```
c = coffee[,c(1:2,4,10:26,28)]
```

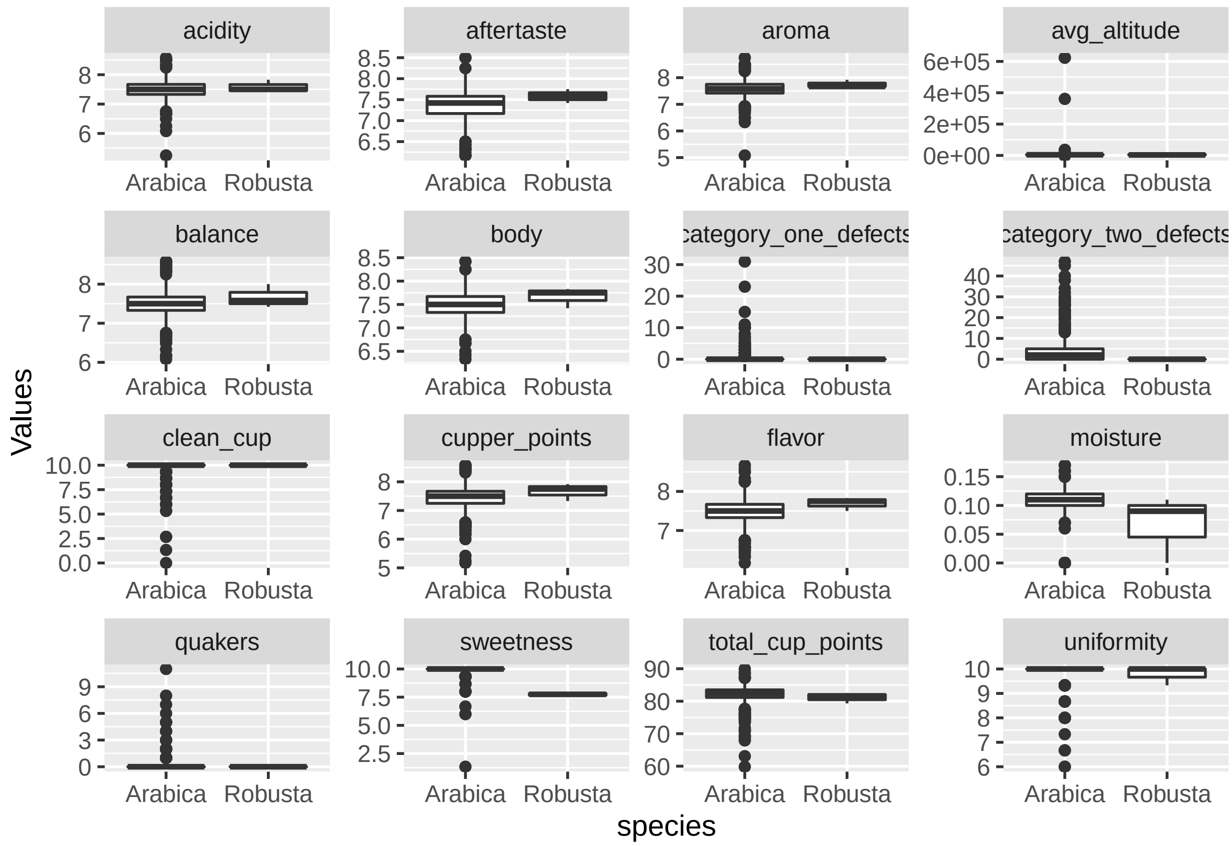
Condense the data

```
c.v1 = c%>%tidyr::pivot_longer(
  cols = !c(species, country_of_origin,variety,processing_method,color),
  names_to = "Variables",
  values_to = "Values")
```

Since, this data set will be re-used for other visuals. Otherwise the following code chunk could be used to generate a specific visual.

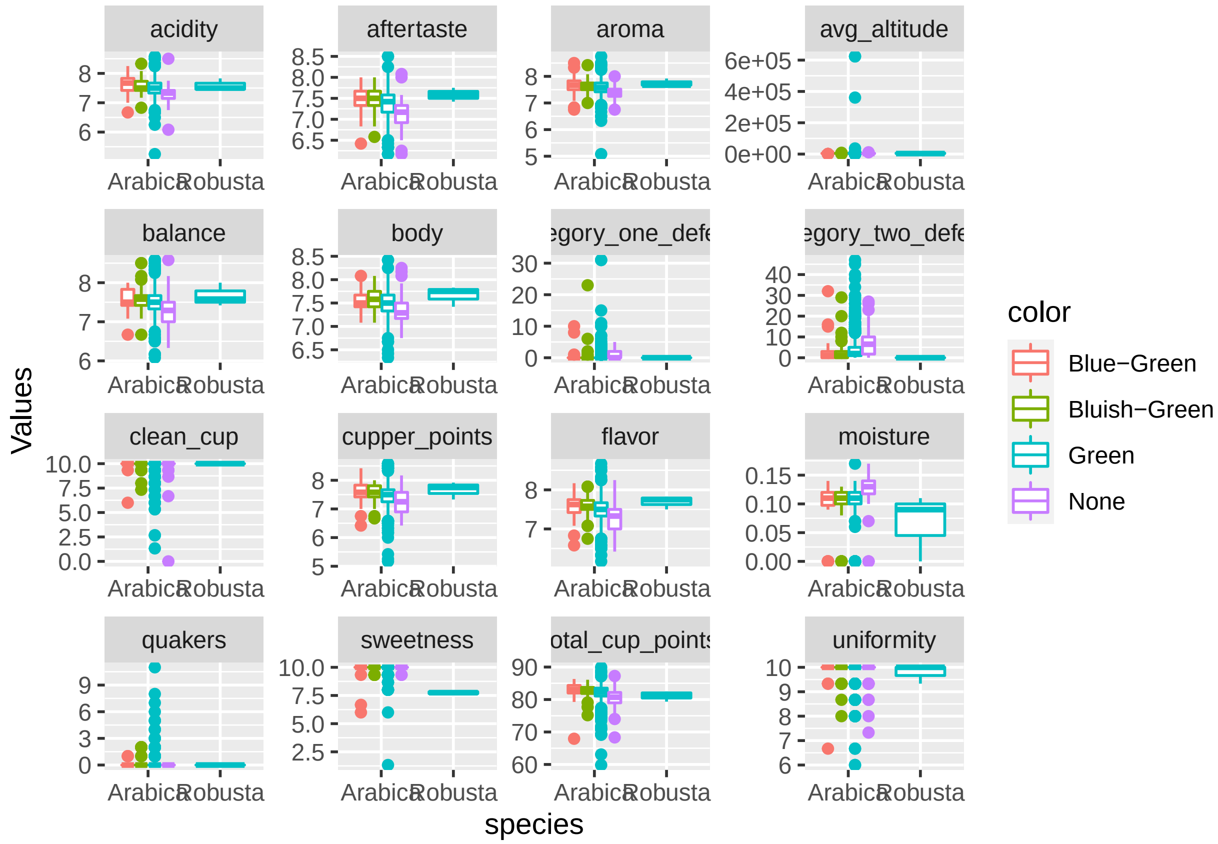
Overall Bbhavior

```
ggplot(c.v1,aes(x=species,y=Values))+
  geom_boxplot()+
  facet_wrap(~Variables,scales = "free")
```



Overall behavior: Coffee Color

```
ggplot(c.v1,aes(x=species,y=Values,color=color))+
  geom_boxplot()+
  facet_wrap(~Variables,scales = "free")
```



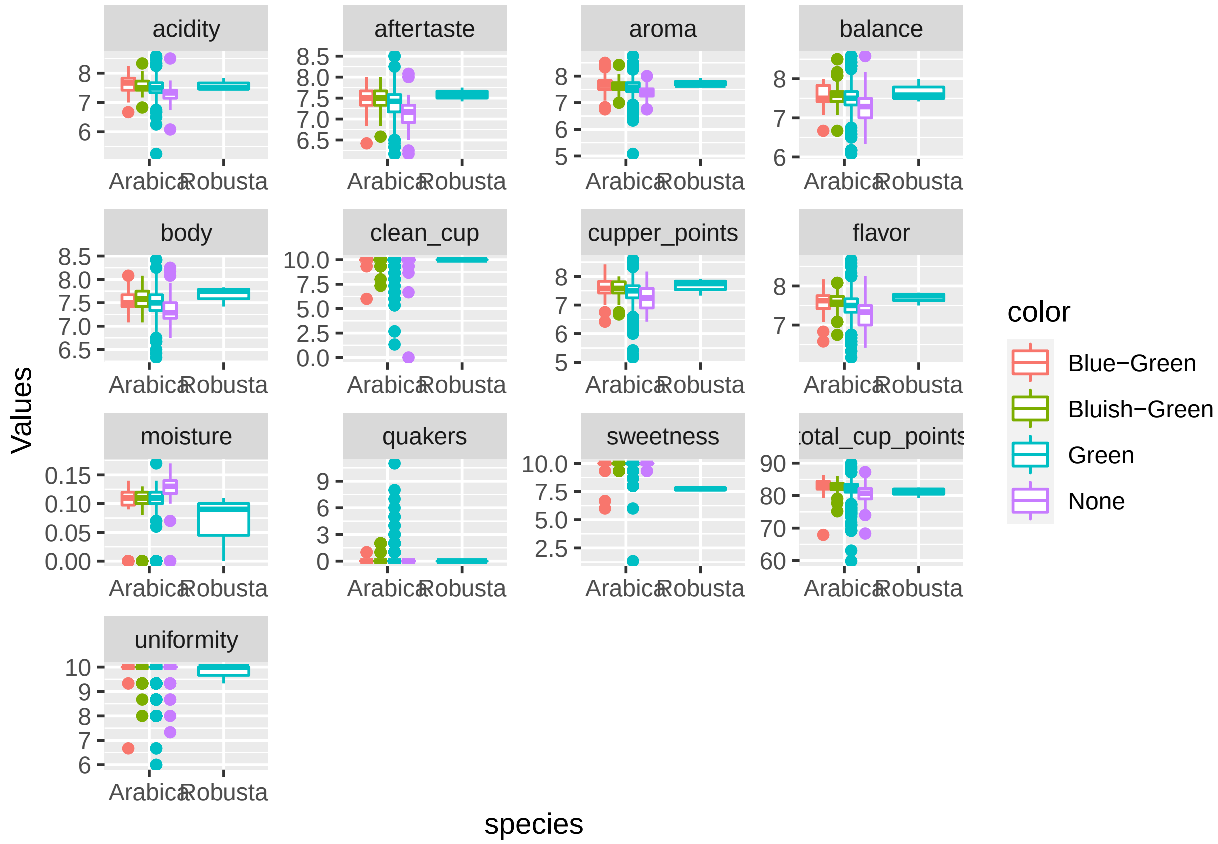
Filter out variables that many outliers

This is an area that could be re-visited and have filters placed on data but chose to remove for initial analysis

```
c.v2 = c.v1 %>%
  filter(Variables != 'avg_altitude' & Variables != 'category_one_defects'& Variables != 'category_two_
```

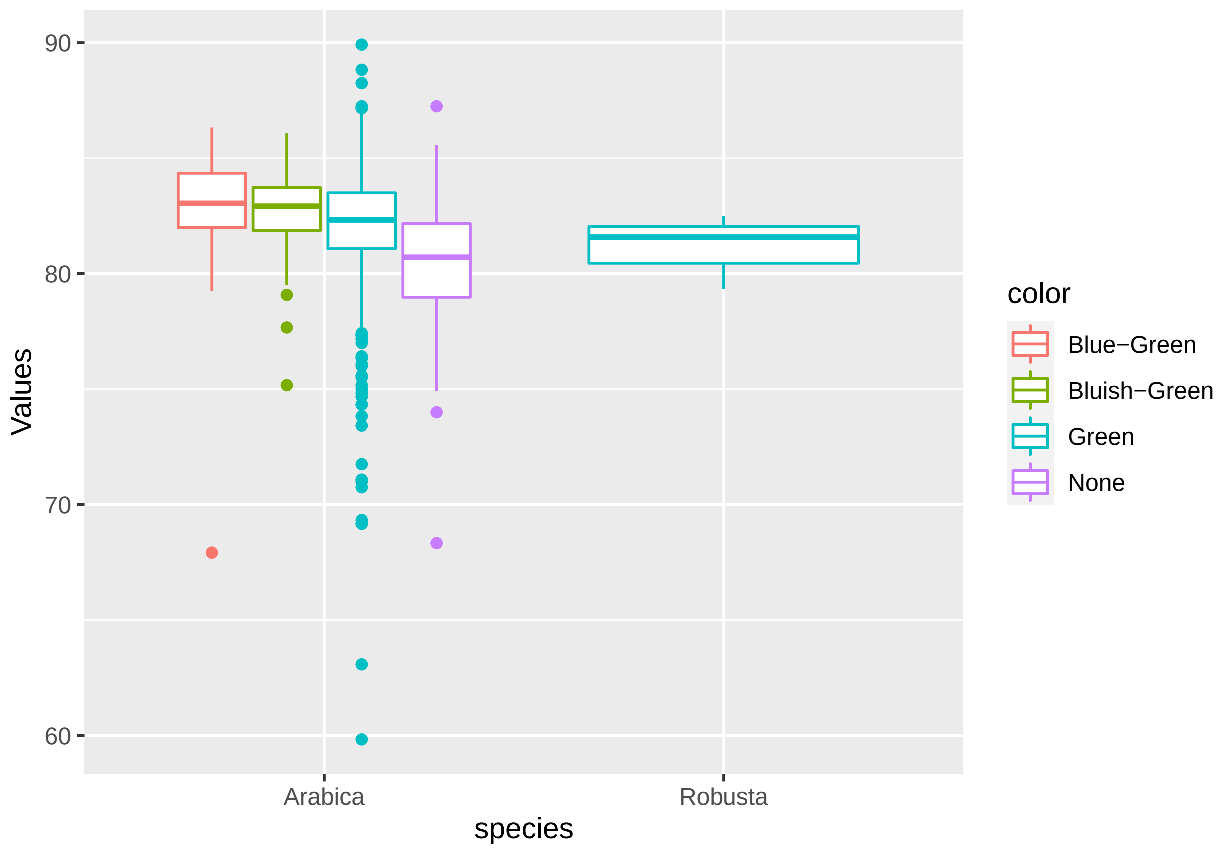
Re-run plot

```
ggplot(c.v2,aes(x=species,y=Values,color=color))+
  geom_boxplot()+
  facet_wrap(~Variables,scales = "free")
```



Cup points distribution

```
c.v2 %>%
  filter(Variables == 'total_cup_points')%>%
  ggplot(aes(x=species,y=Values,color=color))+
  geom_boxplot()
```



Cup points distribution: Coffee Color and Processing Method

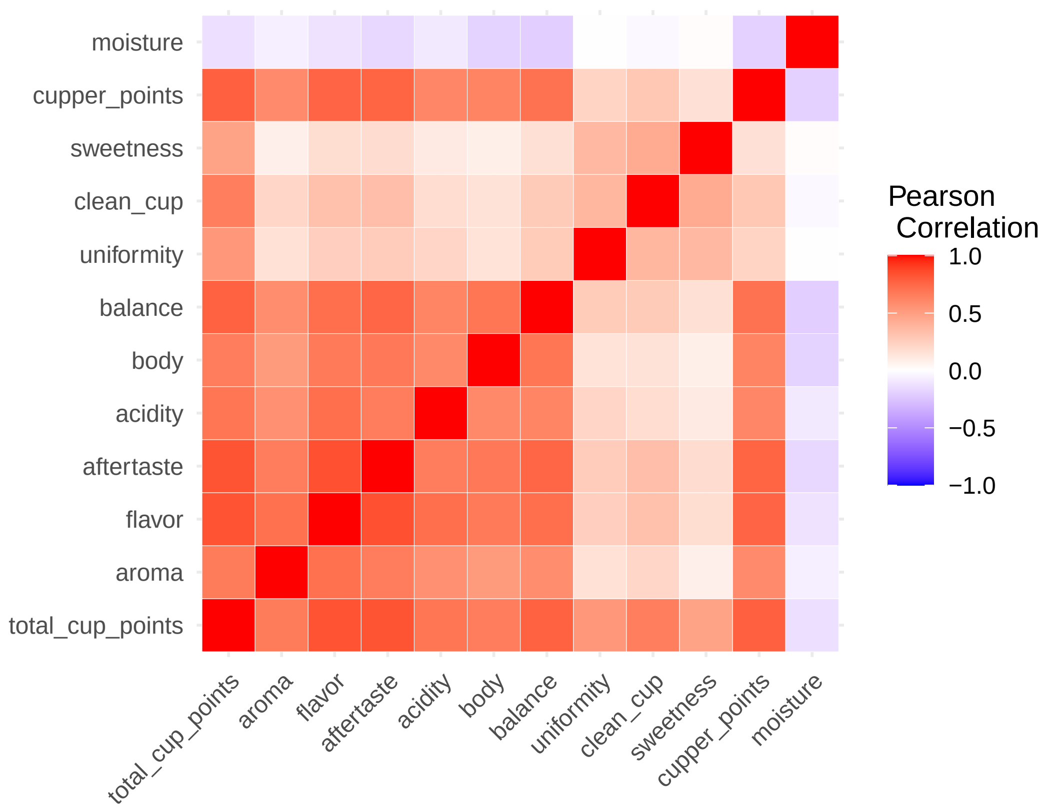
```
c.v2 %>%
  filter(Variables == 'total_cup_points')%>%
  ggplot(aes(x=processing_method,y=Values,color=color))+
  geom_boxplot()+
    facet_wrap(~processing_method,scales = "free")
```

Heatmap of Correlations

```
c = c[,c(1,6:16)]
cormat = cor(c)
melted = reshape::melt(cormat,varnames = c("ParameterX", "ParameterY"))
```

Heatmap

```
ggplot(data = melted, aes(x=ParameterX, y=ParameterY, fill=value)) +
  geom_tile(color = "white")+
 scale_fill_gradient2(low = "blue", high = "red", mid = "white",
   midpoint = 0, limit = c(-1,1), space = "Lab",
   name="Pearson \n Correlation") +
  labs(x = "", y = "")+
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5), axis.text.x = element_text(angle = 45, vjust = 1, size =
 coord_fixed()
```



Function: Calculating Frequency

```
freqq = function(df,col_i,col_j){
  a = df %>%
  group_by({{col_i}},{{col_j}}) %>%
  summarise(count = n()) %>%
  mutate(freq = formattable::percent(count / sum(count)))
  return(a)
}
```

#Overall Frequency all Countries

```
freqq(c.v1,Variables,Values)%>%
  tabDT()
```

Overall Frequency for Brazil

```
freqq(c.v1%>%filter(country_of_origin=="Brazil"),Variables,Values)%>%
  tabDT()
```

Analysis Preparation

Cup Points to Categorical

```
coffee$tcp = coffee$total_cup_points
```

Bins for the Cup Points

```
for(i in 1:894){
  if(coffee[i,29] >= 80){
    coffee[i,29] = 80
  }
  else if(coffee[i,29] >= 70 & coffee[i,29] < 80){
    coffee[i,29] = 70
  }
  else if(coffee[i,29] >= 60 & coffee[i,29] < 70){
    coffee[i,29] = 60
  }
  else{
    coffee[i,29] = 50
  }
}
coffee$tcp = round(coffee$tcp,0)
```

While the bins could be more specific and look at every 2 or 5 points, it made more sense to use broader bins. This is due to trying to understand what makes a coffee from a specific bean have higher or lower overall cup points (i.e., what is the difference between 70s and 80s cup of coffee).

Accuracy table for Model Comparison

```
table_accuracy = matrix(nrow=4,ncol=1)
colnames(table_accuracy) = c('Accuracy')
rownames(table_accuracy) = c('DTree','NB','ANN','KNN')
table_accuracy
```

```
##       Accuracy
## DTree       NA
## NB          NA
## ANN         NA
## KNN         NA
```

This is to help determining which model or models is better than the others. If there are many with similar accuracy, then the model that is the easiest to interpret and explain to a general audience.

Set seed for Reproducibilty

```
set.seed(1)
```

Additional Analysis setup

```
df = coffee[,c(9:22,25,29)]
for(i in 4 : 13){
  df[,i]=round(df[,i],2)
}
```

If the data was processing a bit slowly for initial predicting, as it was too granular so this step was helpful to making the ML run quicker.

Formatting Data

```
df$processing_method= as.factor(df$processing_method)
df$variety = as.factor(df$variety)
df = df[,c(1:16)]
df$tcp = as.factor(df$tcp)
df$moisture = round(df$moisture,1)
```

This was missed earlier in the summary, but the fields that are characters, need to be changed to type factor for the analysis.

Simple k-fold cross validation(cv)

```
n = nrow(df)
folds = 10
tail = n%/%folds

rnd = runif(n)
rank = rank(rnd)

#block/chunk from cv
blk = (rank-1)%/%tail+1
blk = as.factor(blk)

#to see formation of folds
print(summary(blk))
```

```
##  1  2  3  4  5  6  7  8  9 10 11
## 89 89 89 89 89 89 89 89 89 89  4
```

Could turn the above into a more personalized cross validation method than one of the packages in an R library.

Predicitve Analysis

Decision Tree (rpart)

```
set.seed(1)

all.acc = numeric(0)
for(i in 1:folds){
  tree = rpart::rpart(tcp~.,df[blk != i,],method="class")
  pred = predict(tree,df[blk==i,],type="class")
  confMat = table(pred,df$tcp[blk==i])
  acc = (confMat[1,1]+confMat[2,2]+confMat[3,3]+confMat[4,4])/sum(confMat)
  all.acc = rbind(all.acc,acc)
}

print(mean(all.acc))
```

```
## [1] 0.9516854
```

```
table_accuracy[1,1] = mean(all.acc)
```

A 95% overall accuracy is really good! This indicates if following this tree, with details on a bean one could reasonable figure out what its overall score will be prior to evaluation. It also indicates what are the more important parameters are for a coffee scoring.

Example of a table matrix

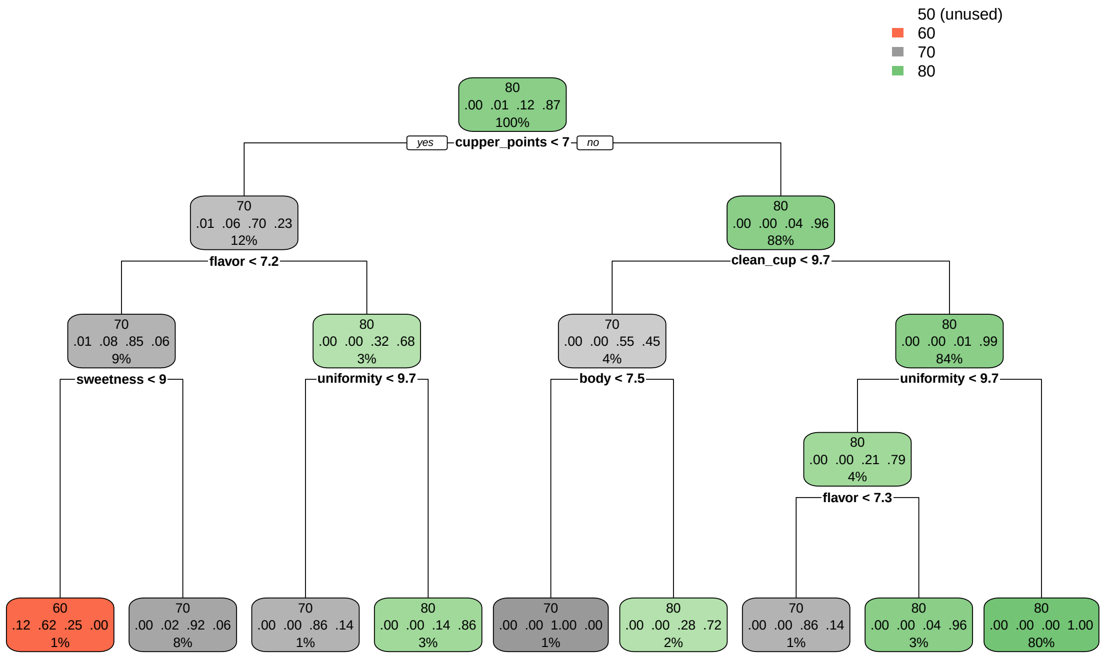
```
confMat
```

```
##
## pred 50 60 70 80
##   50  0  0  0  0
##   60  0  0  0  0
##   70  0  0 13  0
##   80  0  0  3 73
```

This indicates, for the given run, there were 3 miss classifications. Where the tree suggested that the bean should have been in the 80s, but was actually in the 70s.

Visual of Decision Tree

```
rpart.plot::rpart.plot(tree)
```



From this plot, I could just bin 50s with the 60sw group. This will help with future evaluations where re-binning the classifier would be a potential option to get more granular information.

Naive Bayes (e1071)

```
set.seed(1)

all.acc = numeric(0)
for(i in 1:folds){
  model = e1071::naiveBayes(tcp~.,df[blk != i,],method="class")
  pred = predict(model,df[blk==i,],type="class")
  confMat = table(pred,df$tcp[blk==i])
  acc = (confMat[1,1]+confMat[2,2]+confMat[3,3]+confMat[4,4])/sum(confMat)
  all.acc = rbind(all.acc,acc)
}

print(mean(all.acc))
```

```
## [1] 0.9550562
```

```
table_accuracy[2,1] = mean(all.acc)
```

Wierd R Issue

```r
#switch the classifier to numerical
df$tcp = round(as.numeric(df$tcp),0)
#them switch it back to a factor
df$tcp = as.factor(df$tcp)
```

This was a very weird issue. I knew that this was a factor was needed for the classifier. However, it was throwing a NaN for an accuracy value and just by switching the format back and forth corrected it.

Neural Network (nnet)

```r
set.seed(1)

all.acc = numeric(0)
for(i in 1:folds){
  model = nnet::nnet(tcp~.,df[blk != i,], size = 11, trace=FALSE, rang=.06, decay=.006,maxit=500)
  pred = predict(model, df[blk==i,],type="class")
  confMat = table(factor(pred,levels=1:4),factor(df$tcp[blk==i],levels=1:4))
  acc = (confMat[1,1]+confMat[2,2]+confMat[3,3]+confMat[4,4])/sum(confMat)
  all.acc = rbind(all.acc,acc)
}
print(mean(all.acc))
```

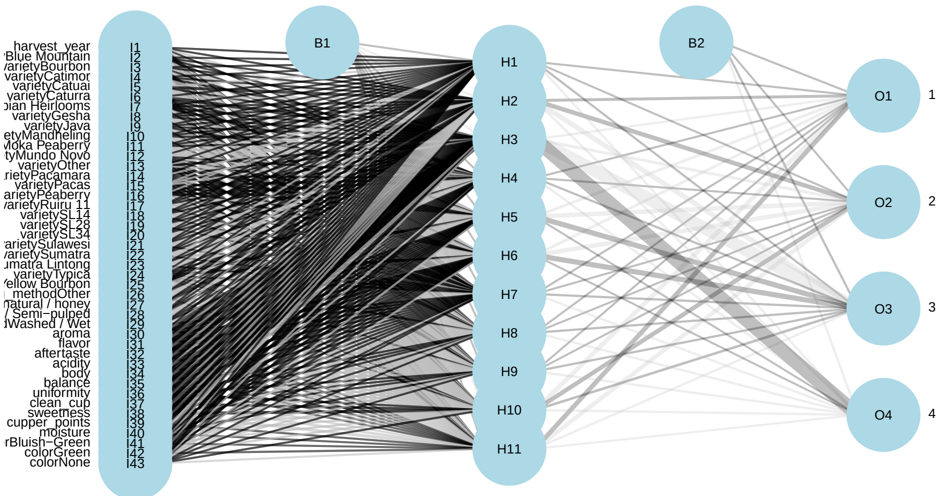
```
## [1] 0.8932584
```

```r
table_accuracy[3,1] = mean(all.acc)
```

Not the best not the worst NN that I have seen. If there was more time, I would have liked to increased the classifiers and used a different library that allowed for more hidden layers.

Neuarl Network Visual (NeuralNetTools)

```r
NeuralNetTools::plotnet(model,circle_cex=5,cex_val=.4,max_sp=TRUE,alpha_val=.25,skip=TRUE)
```



Note

An issue I ran in to:

I re-formatted the label/target field and went from a binary (good [>74]/bad[<75]) classifier to what is it currentlty; 50s,60s,70s, and 80s. However, when running running the all of the PAs prior to neural network there were no strange issues. When running the NN I recieved an output accuracy of 0.003 an knew there was an issue.

There was an (un)interesting issue with NN table (well, all tables), as it was dropping the first two rows as it was not forward feeding into those nodes. The following is the work around to resolve this issue.

Before

```
set.seed(1)
i=1
  model = nnet::nnet(tcp~.,df[blk != i,], size = 10, trace=FALSE, wgts=.05)
  pred = predict(model, df[blk==i,],type="class")
  confMat = table(pred,df$tcp[blk==i])
  confMat
```

```
##
## pred  1  2  3  4
##    3  1  0 16 72
```

After

```
set.seed(1)
i=1
  model = nnet::nnet(tcp~.,df[blk != i,], size = 10, trace=FALSE, wgts=.05)
  pred = predict(model, df[blk==i,],type="class")
  confMat = table(factor(pred,levels=1:4),factor(df$tcp[blk==i],levels=1:4))
  confMat
```

```
##
##      1  2  3  4
##   1  0  0  0  0
##   2  0  0  0  0
##   3  1  0 16 72
##   4  0  0  0  0
```

This was then applied to all of the PAs.

K-Nearest Neighbor Preparation

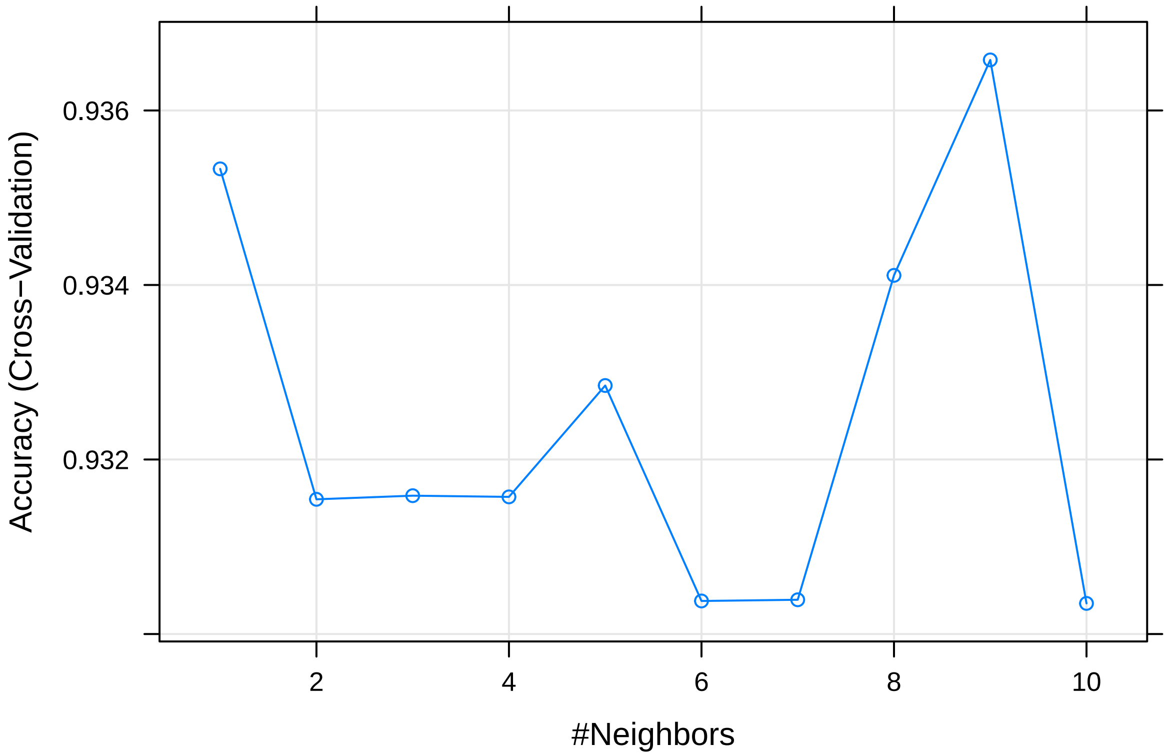
```
set.seed(1)
df$tcp = as.factor(df$tcp)
trControl <- caret::trainControl(method  = "cv", number  = 10)
knn = df[,]
```

KNN

```
set.seed(1)
model <- caret::train(tcp ~ .,
             method     = "knn",
             tuneGrid   = expand.grid(k = 1:10),
             trControl  = trControl,
             data       = knn)
```

```
## Loading required package: lattice
```

```
acc = mean(model$results$Accuracy)
table_accuracy[4,1] = acc

plot(model)
```



This is a visual to see how many neighbors the KNN will be running. From this visual it could possibly run at 9 groups due to the accuracy level.

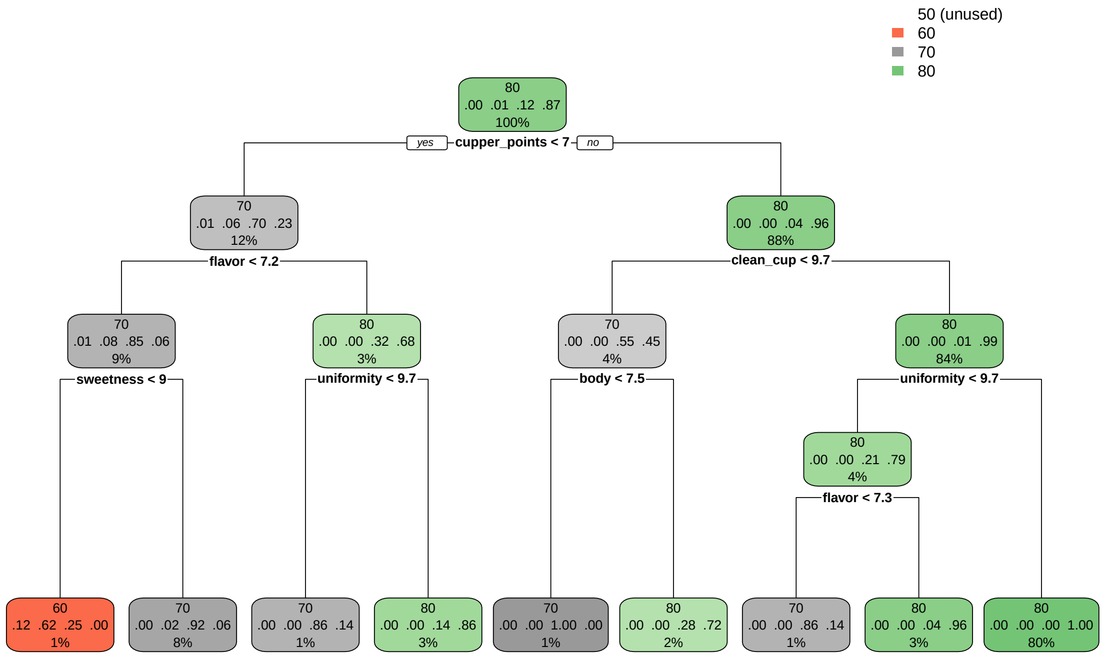
View Accuracy Table

```
tab = round(table_accuracy,4)
tab
```

```
##       Accuracy
## DTree   0.9517
## NB      0.9551
## ANN     0.8933
## KNN     0.9325
```

Preferred Model

```
rpart.plot::rpart.plot(tree)
```



Top 3 parameters for understanding a coffee's score.

~Cupper points are the most informative parameter in deciding if a coffee is to be in the 80s or below this.

~If place coffee is <7 cupper points, the next deciding factor is how good is the flavor of the coffee.

~ If coffee is >7 cupper points, the next deciding factor is how clean the coffee leaves the cup.

For further analysis

```
df2 = coffee[,c(4,5,9:22,25,29)]
for(i in 6 : 16){
  df2[,i]=round(df2[,i],2)

df2$processing_method= as.factor(df2$processing_method)
df2$variety = as.factor(df2$variety)
df2$tcp = as.factor(df2$tcp)
df2$moisture = round(df2$moisture,1)
df2$color = as.factor(df2$color)
df2$country_of_origin = as.factor(df2$country_of_origin)
df2$region = as.factor(df2$region)
df3 = df2[,c(1,3:18)]
}
```

```r
set.seed(1)
n = nrow(df3)
folds = 10
tail = n%/%folds

rnd = runif(n)
rank = rank(rnd)

#block/chunk from cv
blk = (rank-1)%/%tail+1
blk = as.factor(blk)

#to see formation of folds
print(summary(blk))
```

```
##  1  2  3  4  5  6  7  8  9 10 11
## 89 89 89 89 89 89 89 89 89 89  4
```

```r
set.seed(1)

all.acc = numeric(0)
for(i in 1:folds){
  tree = rpart::rpart(tcp~.,df3[blk != i,],method="class")
  pred = predict(tree,df3[blk==i,],type="class")
  confMat = table(pred,df3$tcp[blk==i])
  acc = (confMat[1,1]+confMat[2,2]+confMat[3,3]+confMat[4,4])/sum(confMat)
  all.acc = rbind(all.acc,acc)
}

print(mean(all.acc))
```

Interestingly, adding countries lowers the accuracy.

```r
rpart.plot::rpart.plot(tree)
```

From the visual, it appears that Central and South America do not produce good coffee.