Project

SeanJ- Volcaetus

4/3/2022

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)
```

```
##  total_cup_points   species              owner            country_of_origin
##  Min.   : 0.00    Length:1339        Length:1339        Length:1339
##  1st Qu.:81.08    Class :character   Class :character   Class :character
##  Median :82.50    Mode  :character   Mode  :character   Mode  :character
##  Mean   :82.09
##  3rd Qu.:83.67
##  Max.   :90.58
##
##   farm_name          lot_number            mill             ico_number
##  Length:1339        Length:1339        Length:1339        Length:1339
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##    company            altitude            region            producer
##  Length:1339        Length:1339        Length:1339        Length:1339
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  number_of_bags    bag_weight         in_country_partner harvest_year
##  Min.   :   0.0   Length:1339        Length:1339        Length:1339
##  1st Qu.:  14.0   Class :character   Class :character   Class :character
##  Median : 175.0   Mode  :character   Mode  :character   Mode  :character
##  Mean   : 154.2
##  3rd Qu.: 275.0
##  Max.   :1062.0
```

```
##
##  grading_date        owner_1           variety          processing_method
##  Length:1339        Length:1339        Length:1339        Length:1339
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##      aroma           flavor        aftertaste       acidity            body
##  Min.   :0.000   Min.   :0.00   Min.   :0.000   Min.   :0.000   Min.   :0.000
##  1st Qu.:7.420   1st Qu.:7.33   1st Qu.:7.250   1st Qu.:7.330   1st Qu.:7.330
##  Median :7.580   Median :7.58   Median :7.420   Median :7.580   Median :7.500
##  Mean   :7.567   Mean   :7.52   Mean   :7.401   Mean   :7.536   Mean   :7.517
##  3rd Qu.:7.750   3rd Qu.:7.75   3rd Qu.:7.580   3rd Qu.:7.750   3rd Qu.:7.670
##  Max.   :8.750   Max.   :8.83   Max.   :8.670   Max.   :8.750   Max.   :8.580
##
##     balance        uniformity       clean_cup        sweetness
##  Min.   :0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.:7.330   1st Qu.:10.000   1st Qu.:10.000   1st Qu.:10.000
##  Median :7.500   Median :10.000   Median :10.000   Median :10.000
##  Mean   :7.518   Mean   : 9.835   Mean   : 9.835   Mean   : 9.857
##  3rd Qu.:7.750   3rd Qu.:10.000   3rd Qu.:10.000   3rd Qu.:10.000
##  Max.   :8.750   Max.   :10.000   Max.   :10.000   Max.   :10.000
##
##  cupper_points       moisture       category_one_defects    quakers
##  Min.   : 0.000   Min.   :0.00000   Min.   : 0.0000      Min.   : 0.0000
##  1st Qu.: 7.250   1st Qu.:0.09000   1st Qu.: 0.0000      1st Qu.: 0.0000
##  Median : 7.500   Median :0.11000   Median : 0.0000      Median : 0.0000
##  Mean   : 7.503   Mean   :0.08838   Mean   : 0.4795      Mean   : 0.1734
##  3rd Qu.: 7.750   3rd Qu.:0.12000   3rd Qu.: 0.0000      3rd Qu.: 0.0000
##  Max.   :10.000   Max.   :0.28000   Max.   :63.0000      Max.   :11.0000
##                                                          NA's   :1
##     color           category_two_defects  expiration        certification_body
##  Length:1339        Min.   : 0.000       Length:1339        Length:1339
##  Class :character   1st Qu.: 0.000       Class :character   Class :character
##  Mode  :character   Median : 2.000       Mode  :character   Mode  :character
##                     Mean   : 3.556
##                     3rd Qu.: 4.000
##                     Max.   :55.000
##
##  certification_address certification_contact unit_of_measurement
##  Length:1339           Length:1339           Length:1339
##  Class :character      Class :character      Class :character
##  Mode  :character      Mode  :character      Mode  :character
##
##
##
##
##  altitude_low_meters altitude_high_meters altitude_mean_meters
##  Min.   :     1      Min.   :     1       Min.   :     1
##  1st Qu.:  1100      1st Qu.:  1100       1st Qu.:  1100
##  Median :  1311      Median :  1350       Median :  1311
##  Mean   :  1751      Mean   :  1799       Mean   :  1775
```

```
##  3rd Qu.:  1600      3rd Qu.:  1650       3rd Qu.:  1600
##  Max.   :190164      Max.   :190164       Max.   :190164
##  NA's   :230         NA's   :230          NA's   :230
```

Quite a few NA's.

Numerical Columns: 1 within quakers, and 230 in Altitude low/high/mean.

Next, nee to check what is happening in the rest of the data set, the character type.

Count of NA's per coloumn

```
#type of data, col = 2, type of function applied
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
```

I had to use a quick google search to figure if margin had to be 1 or 2. Site: https://www.guru99.com/r-apply-sapply-tapply.html

There a quite a few missing values 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.

```
library(tidyverse)
```

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

```
## v ggplot2 3.3.4     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.7
## 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()
```

Removal of columns

```
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)
```

#Remove Rows Containing Missing Values#

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

#Changing Mass to all Imperial Units of Measurements#

```
#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_cup_points species owner     country_of_origin  region   number_of_bags
##               <dbl> <chr>   <chr>     <chr>              <chr>             <dbl>
##  1             87.2 Arabica the coff~ Costa Rica         san ram~            250
##  2             86.3 Arabica francisc~ Costa Rica         west an~            250
##  3             85.3 Arabica the coff~ Costa Rica         west va~            250
##  4             85.3 Arabica the coff~ Costa Rica         san ram~            250
##  5             84.7 Arabica fabian c~ Costa Rica         tarrazu              50
##  6             84.5 Arabica fabian c~ Costa Rica         tarrazu             250
##  7             83.8 Arabica german n~ United States (Pu~ yauco r~             18
##  8             83.8 Arabica the coff~ Guatemala          quetzal~            250
##  9             83.3 Arabica the coff~ Costa Rica         san ram~            250
## 10             83.3 Arabica itiah co~ Haiti              thiotte~              2
## 11             83   Arabica german n~ United States (Pu~ yauco r~             17
## 12             81.5 Arabica myriam k~ Haiti              dondon,~            300
## 13             81.2 Arabica essencec~ Guatemala          huehuet~             36
## 14             81.1 Arabica german n~ United States (Pu~ yauco r~             18
## 15             80.9 Arabica chris fi~ Nicaragua          matagal~            275
## 16             80.8 Arabica the coff~ Costa Rica         san ram~            250
## 17             79.3 Arabica the coff~ Colombia           pereira             250
## 18             79.1 Arabica german n~ United States (Pu~ yauco r~             18
## # ... with 22 more variables: bag_weight <chr>, in_country_partner <chr>,
## #   harvest_year <chr>, variety <chr>, 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>
```

```r
#separating out the columns based on the value and units associated with it
coffee = 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)
```

#Changing Length to all Imperial Units of Measurements#

```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)
```

#Altering rows with years with form Year1/Year2 to the intial 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 to see the data for potentail outliers#

```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
```

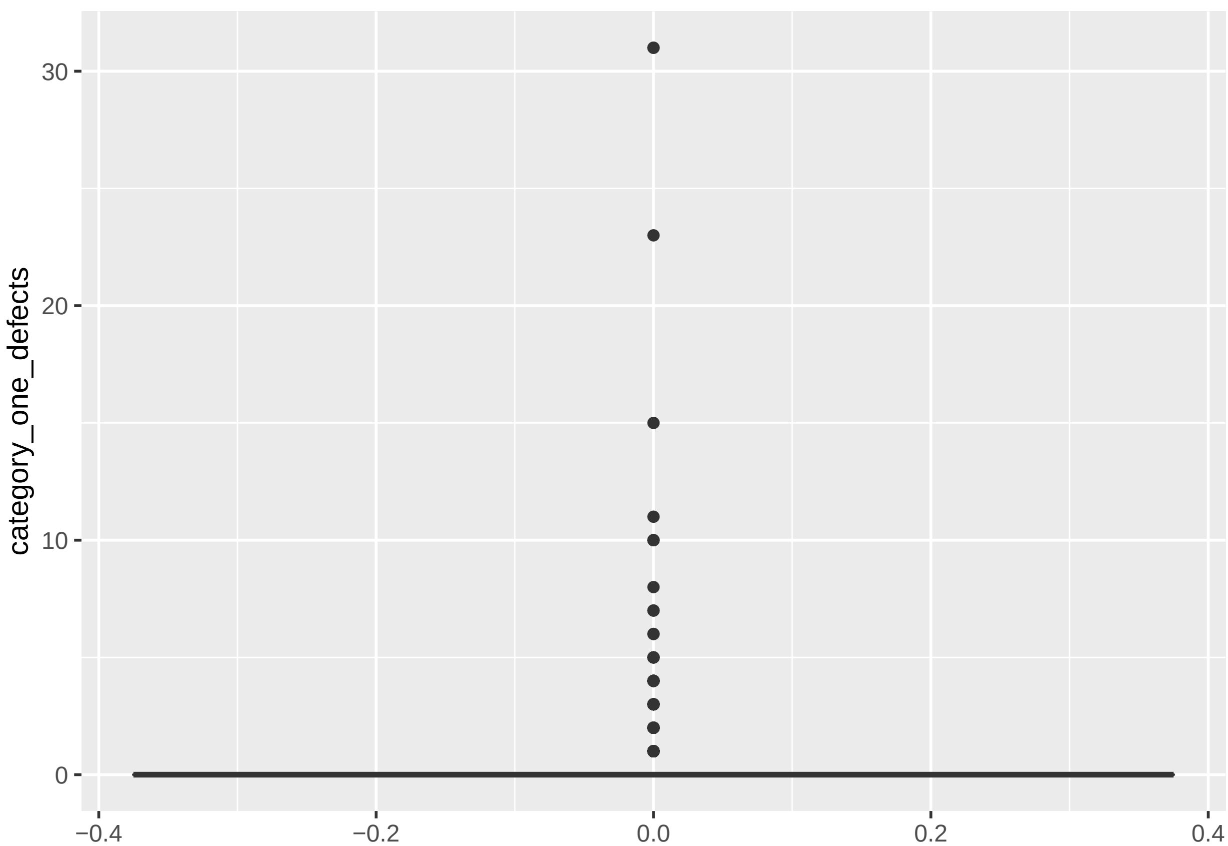
The parameters for defects, quakers, and average altitude seem to have quite a range for values. Additionally, it can be seen for these fields that the max points are quite a ways away from the mean.

EDA / Visuals

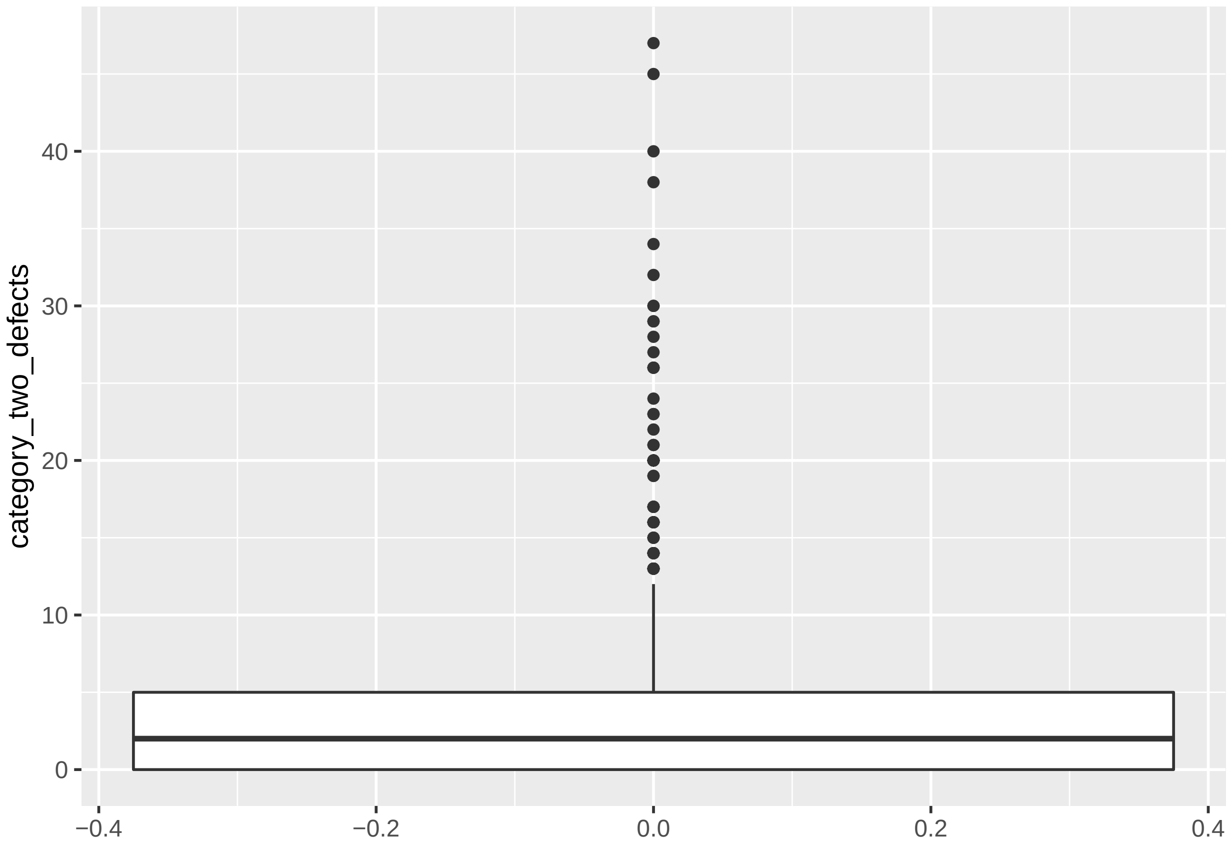
```
library(ggplot2)
```

#Check for outliers in some of the fields#

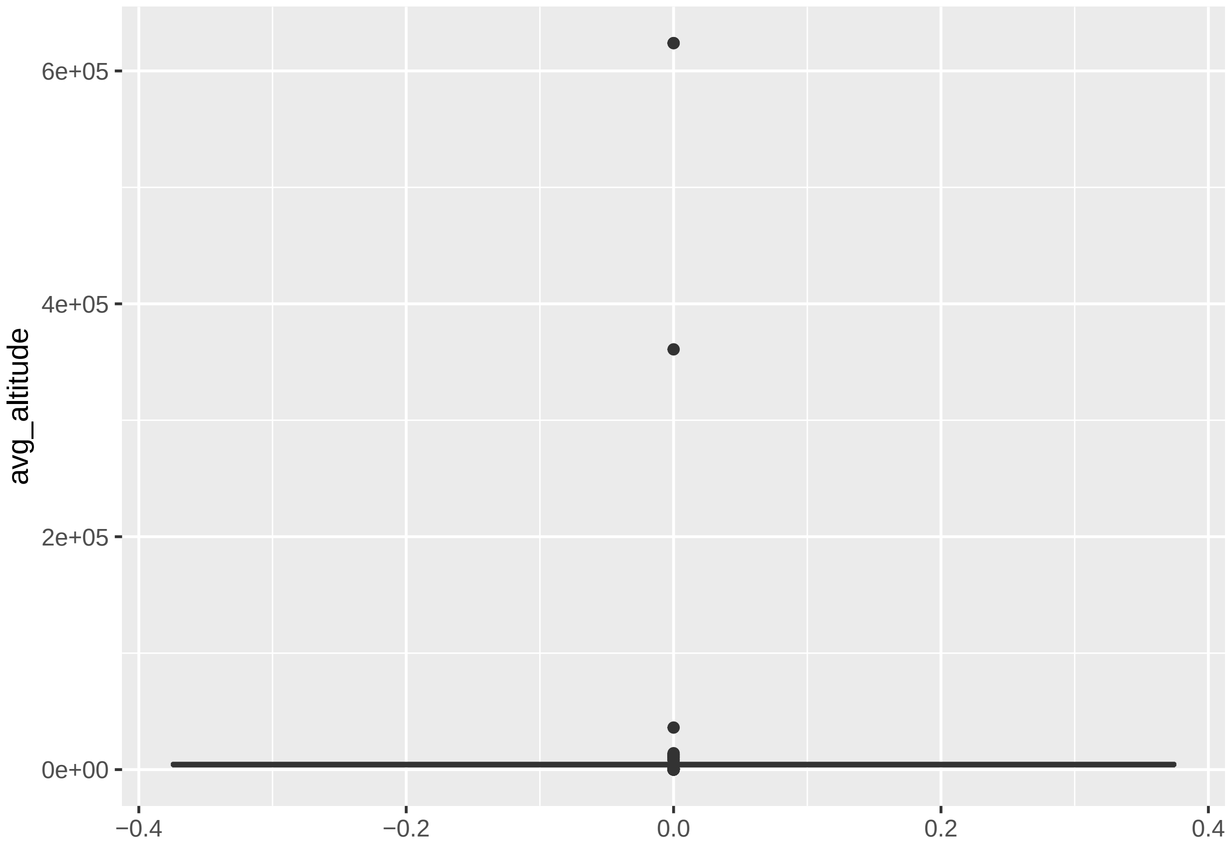
```
defect1_plt = ggplot(coffee, aes(y=category_one_defects)) +
              geom_boxplot()
defect2_plt = ggplot(coffee, aes(y=category_two_defects)) +
              geom_boxplot()
alt_plt = ggplot(coffee, aes(y=avg_altitude)) +
              geom_boxplot()
quakers = ggplot(coffee, aes(y=quakers)) +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

defect1_plt
```



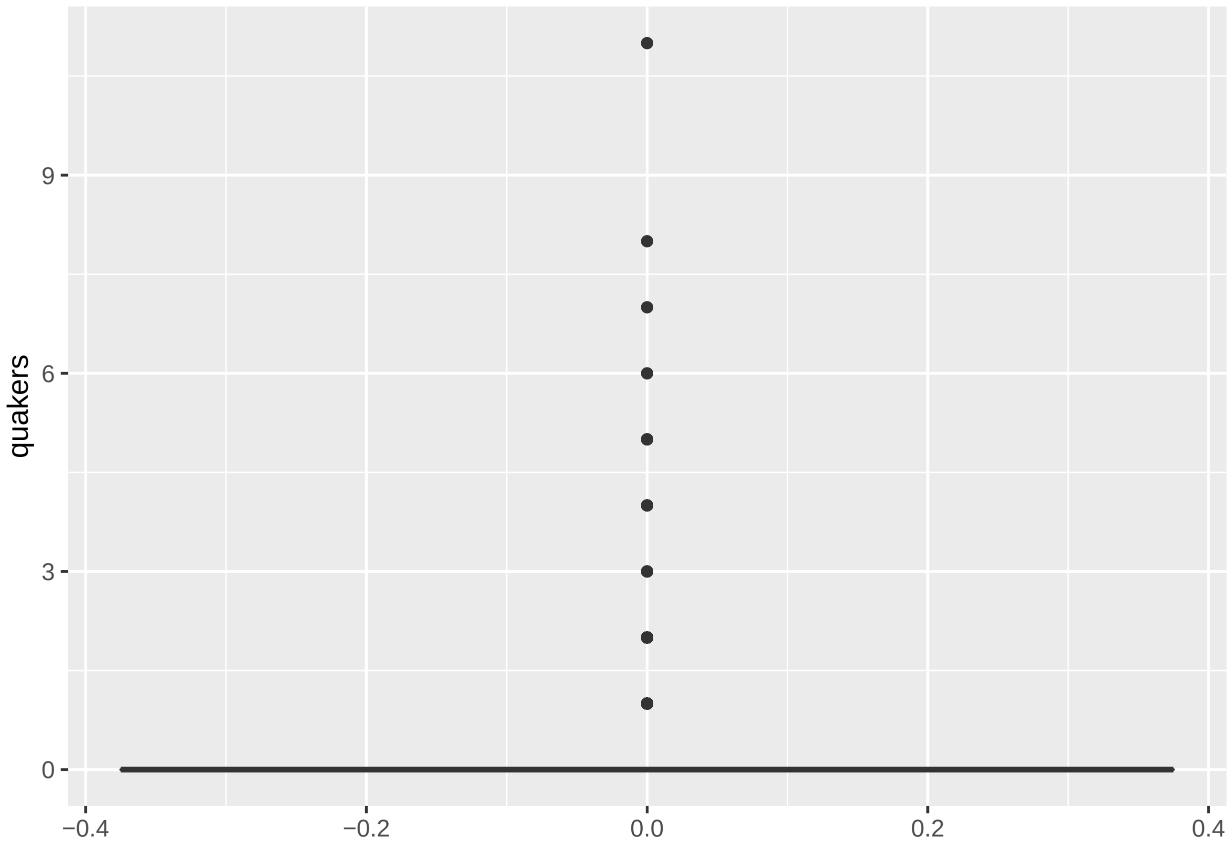

```
defect2_plt
```




```
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.

#Redfine the Dataset#

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

#Condense the data#

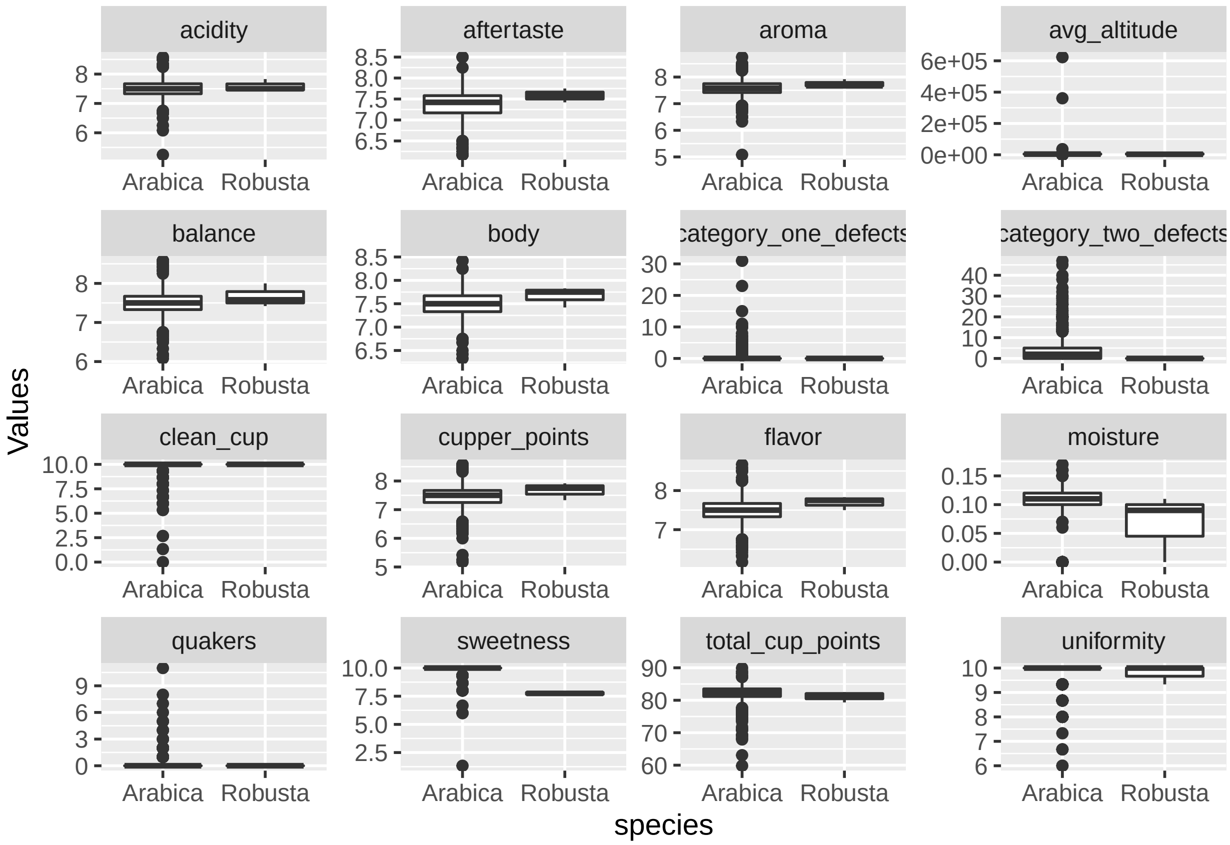
```
c.v1 = c%>%pivot_longer(
  cols = !c(species, country_of_origin,variety,processing_method,color),
  names_to = "Variables",
  values_to = "Values")
```

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

```
c%>%pivot_longer(
  cols = !c(species, country_of_origin,variety,processing_method,color),
  names_to = "Variables",
  values_to = "Values")%>%
  ggplot(aes(x=species,y=Values,color=color))+
  geom_boxplot()+
  facet_wrap(~Variables,scales = "free")
```

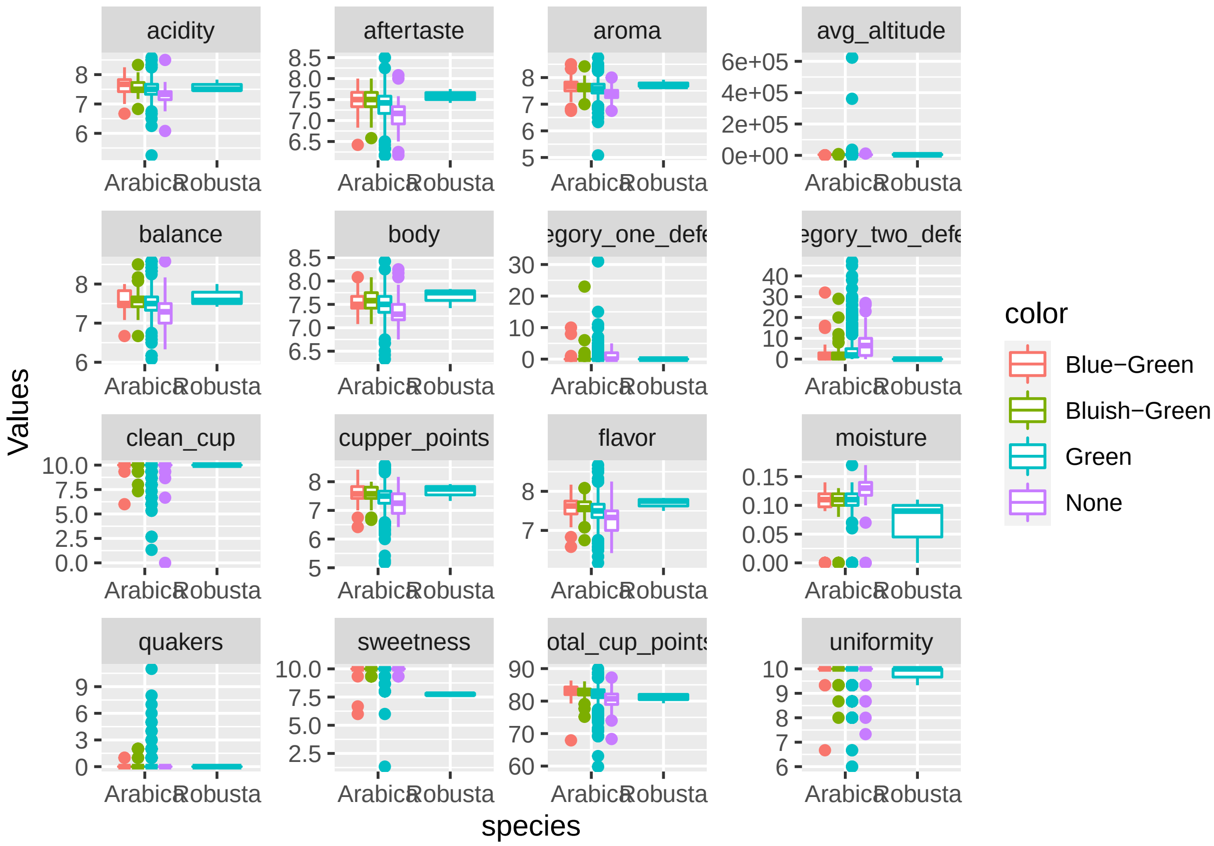
#Plot the data to see overall behavior#

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



#Plot the data to see overall behavior for specific field Coffee Color#

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

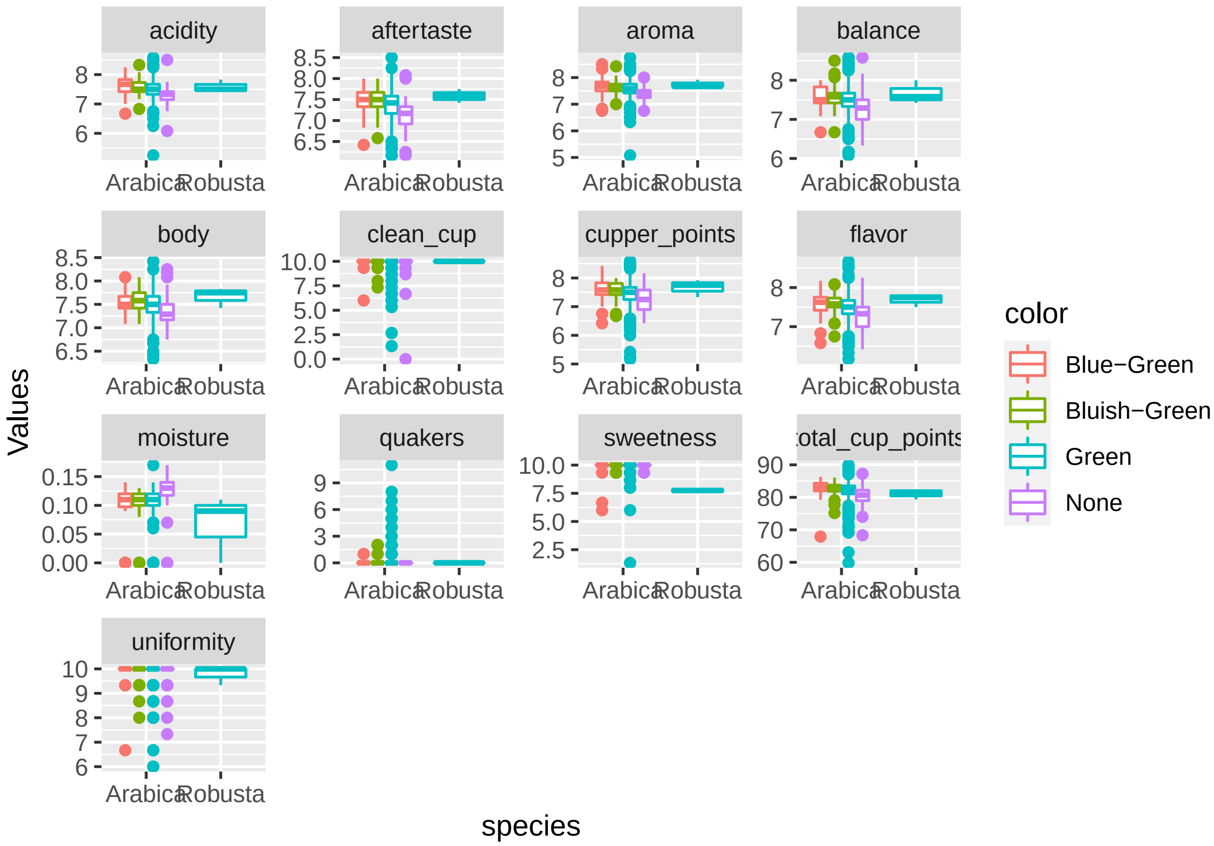


#Filter out the items that have known outliers#

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

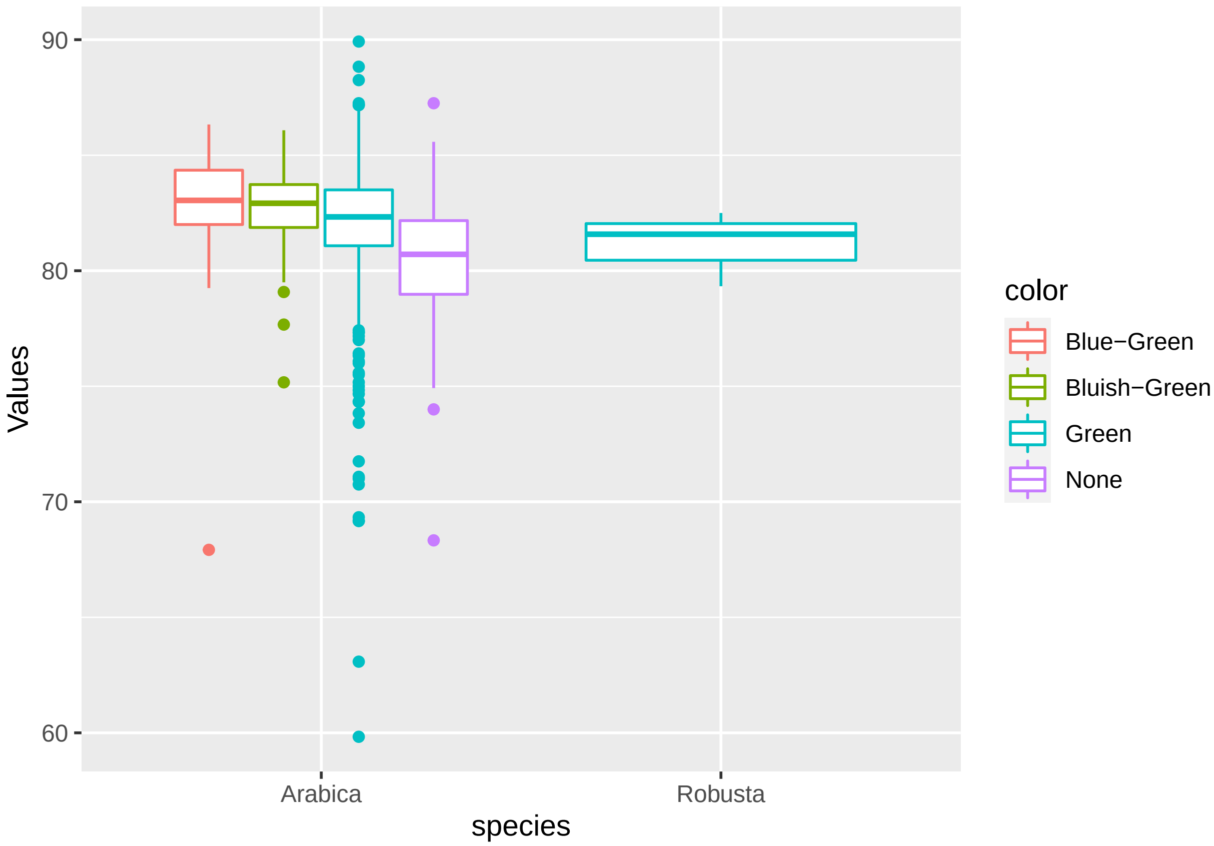
#Re-run plot#

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

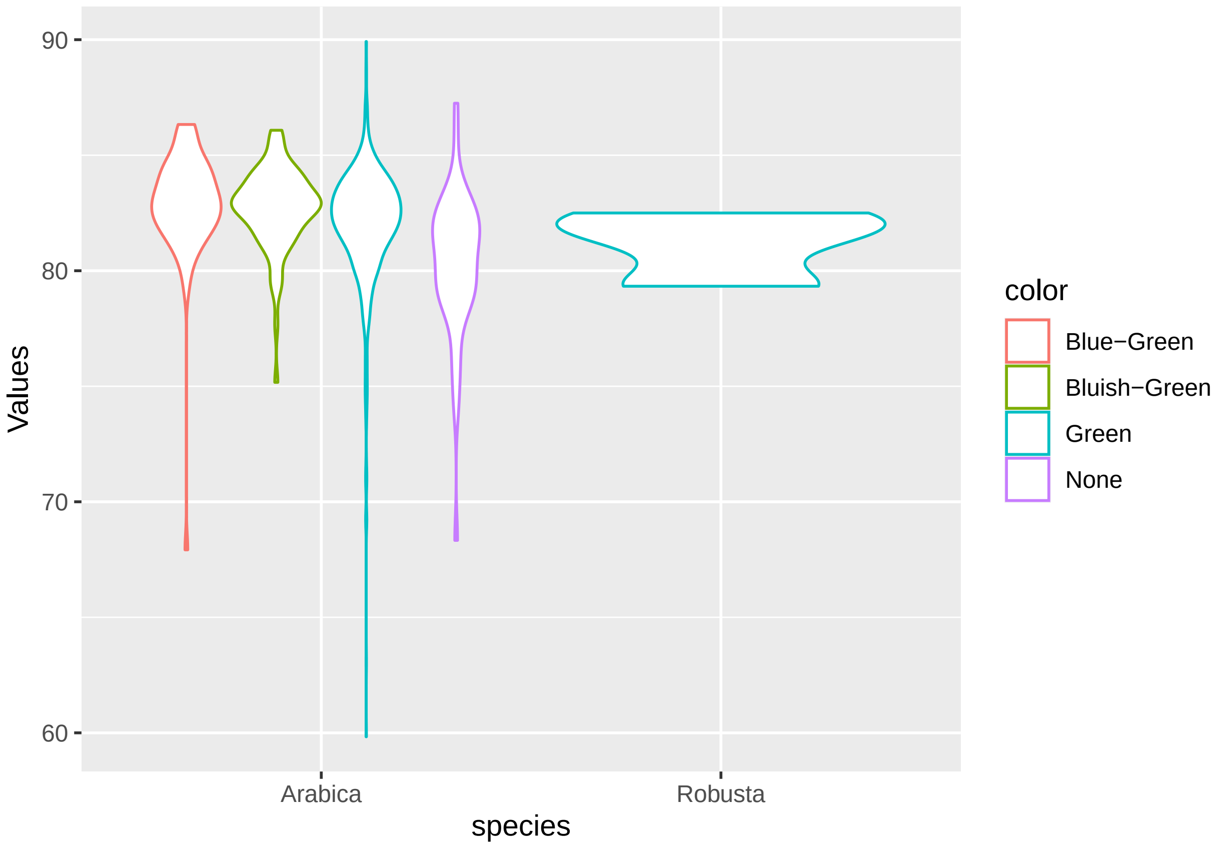


#How are the cup points distributed and where the 'weight' it is at by the Species and Coffee Color#

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

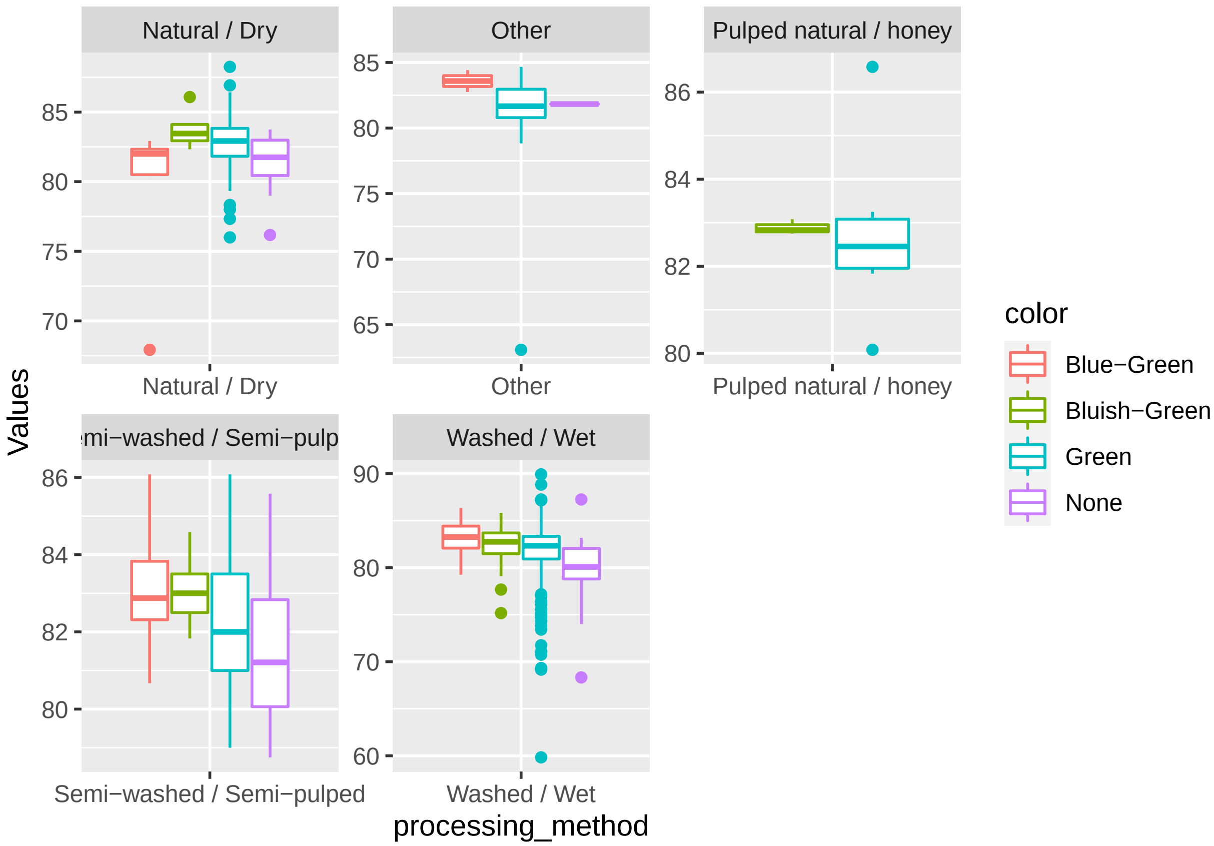



```
c.v2 %>%
  filter(Variables == 'total_cup_points')%>%
  ggplot(aes(x=species,y=Values,color=color))+geom_violin()
```

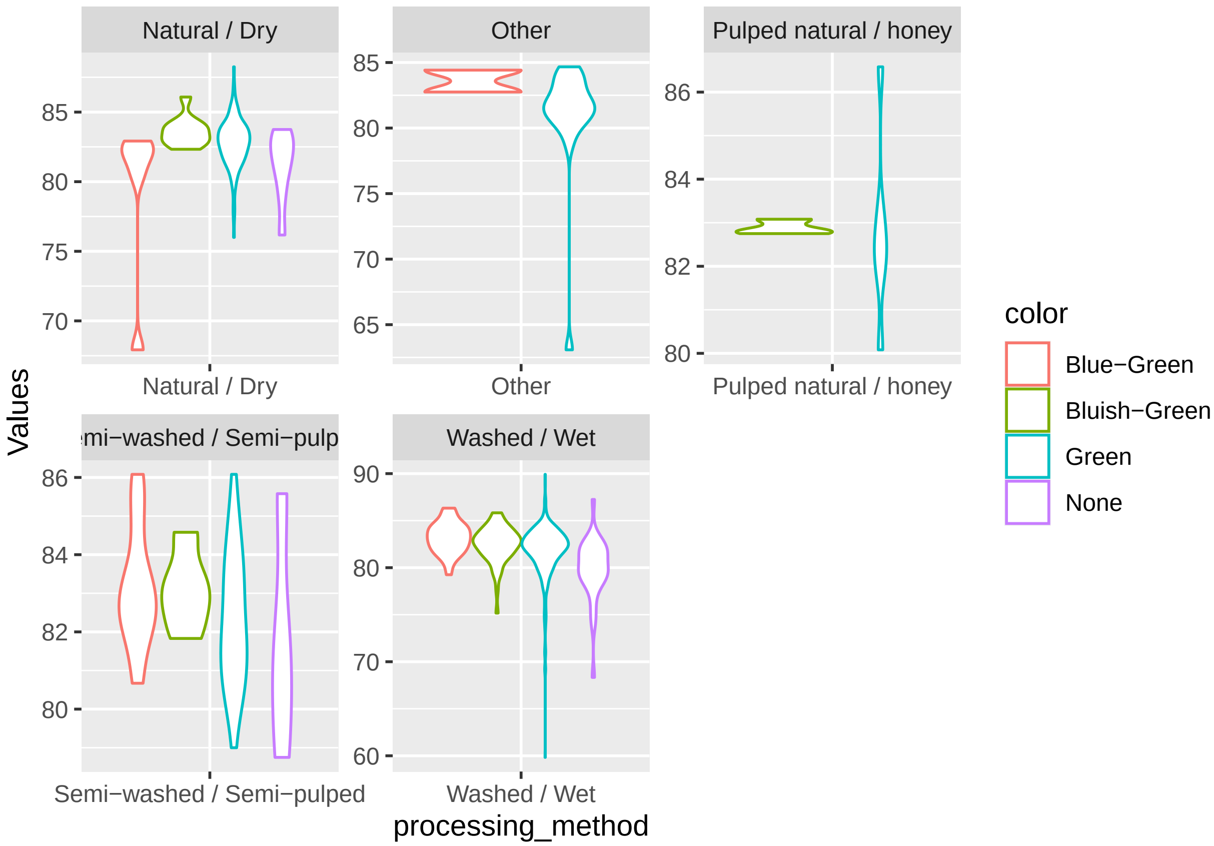


#How are the cup points distributed and where the 'weight' it is at by the Coffee Color and Processing Method#

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




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



#Heatmap of Correlations#

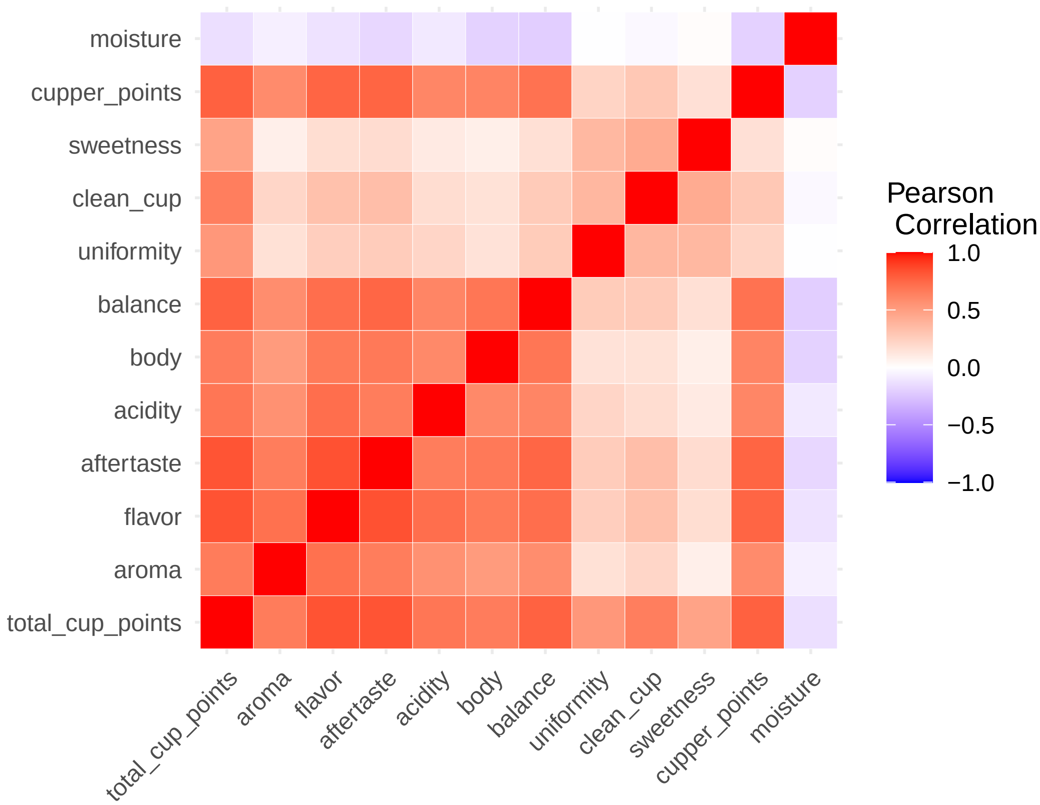
```
library(reshape)
```

```
##
## Attaching package: 'reshape'

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

## The following objects are masked from 'package:tidyr':
##
##     expand, smiths
```

```
c = c[,c(1,6:16)]
cormat = cor(c)
melted = 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()
```



This took quite a bit of using ggplot2 to aid in creating this visual. I used quite a few site for reference. ~1 https://ggplot2.tidyverse.org/reference/geom_tile.html ~2 https://r-charts.com/correlation/heat-map-ggplot2/ ~3 https://stackoverflow.com/questions/1330989/rotating-and-spacing-axis-labels-in-ggplot2

```
detach('package:reshape')
```

#See the data make-up in a numerical summary#

```
library(formattable)
```

#Function for 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)
```

```
## `summarise()` has grouped output by 'Variables'. You can override using the `.groups` argument.
```

```
## # A tibble: 611 x 4
## # Groups:   Variables [16]
##    Variables Values count freq
##    <chr>      <dbl> <int> <formttbl>
##  1 acidity     5.25     1 0.11%
##  2 acidity     6.08     1 0.11%
```

```
##  3 acidity     6.25     1 0.11%
##  4 acidity     6.5      1 0.11%
##  5 acidity     6.67     3 0.34%
##  6 acidity     6.75     2 0.22%
##  7 acidity     6.83     6 0.67%
##  8 acidity     6.92     7 0.78%
##  9 acidity     7       23 2.57%
## 10 acidity     7.08    25 2.80%
## # ... with 601 more rows
```

#Overall Frequency for Brazil#

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

```
## `summarise()` has grouped output by 'Variables'. You can override using the `.groups` argument.
```

```
## # A tibble: 216 x 4
## # Groups:   Variables [16]
##    Variables Values count freq
##    <chr>      <dbl> <int> <formttbl>
##  1 acidity     6.92     1 1.05%
##  2 acidity     7        1 1.05%
##  3 acidity     7.08     3 3.16%
##  4 acidity     7.17     4 4.21%
##  5 acidity     7.25     5 5.26%
##  6 acidity     7.33     8 8.42%
##  7 acidity     7.42     7 7.37%
##  8 acidity     7.5     26 27.37%
##  9 acidity     7.58     9 9.47%
## 10 acidity     7.67    13 13.68%
## # ... with 206 more rows
```

##Analysis Preparation##

#Format new label (total_cup_points) to be categorical#

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

#Creating Bins for the Cup Points#

```r
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 comparison between models#

```
table_accuracy = matrix(nrow=6,ncol=1)
colnames(table_accuracy) = c('Accuracy')
rownames(table_accuracy) = c('DTree','NB','SVM-Linerar','SVM-Polynomial','ANN','KNN')
table_accuracy
```

```
##                Accuracy
## DTree                NA
## NB                   NA
## SVM-Linerar          NA
## SVM-Polynomial       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 so analysis is repeatable#

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

For analysis

```
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.

#Fix issue with the 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)

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

```r
#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#

```r
library(rpart)
set.seed(1)

all.acc = numeric(0)
for(i in 1:folds){
  tree = 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
```

```r
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 of predicted(rows) and actual(columns)

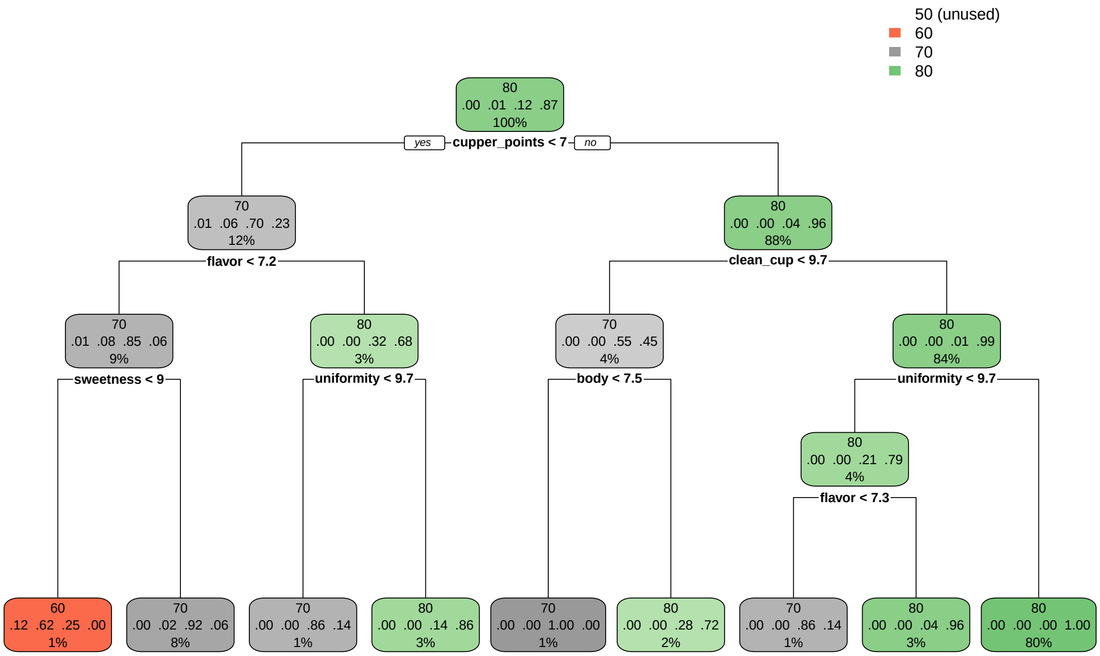
```r
confMat
```

```
##
## 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

```r
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

```
library(e1071)
set.seed(1)

all.acc = numeric(0)
for(i in 1:folds){
  model = 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)
```

Another nice and high accuracy for this PA!

#Lineat Support Vector Machine (SVM)#

```
set.seed(1)

all.acc = numeric(0)
for(i in 1:folds){
  model = svm(tcp~. ,df[blk != i,],kernel="linear",type="C")
  pred = predict(model,df[blk==i,],type="class")
  confMat = table(pred,df$tcp[blk==i])
```

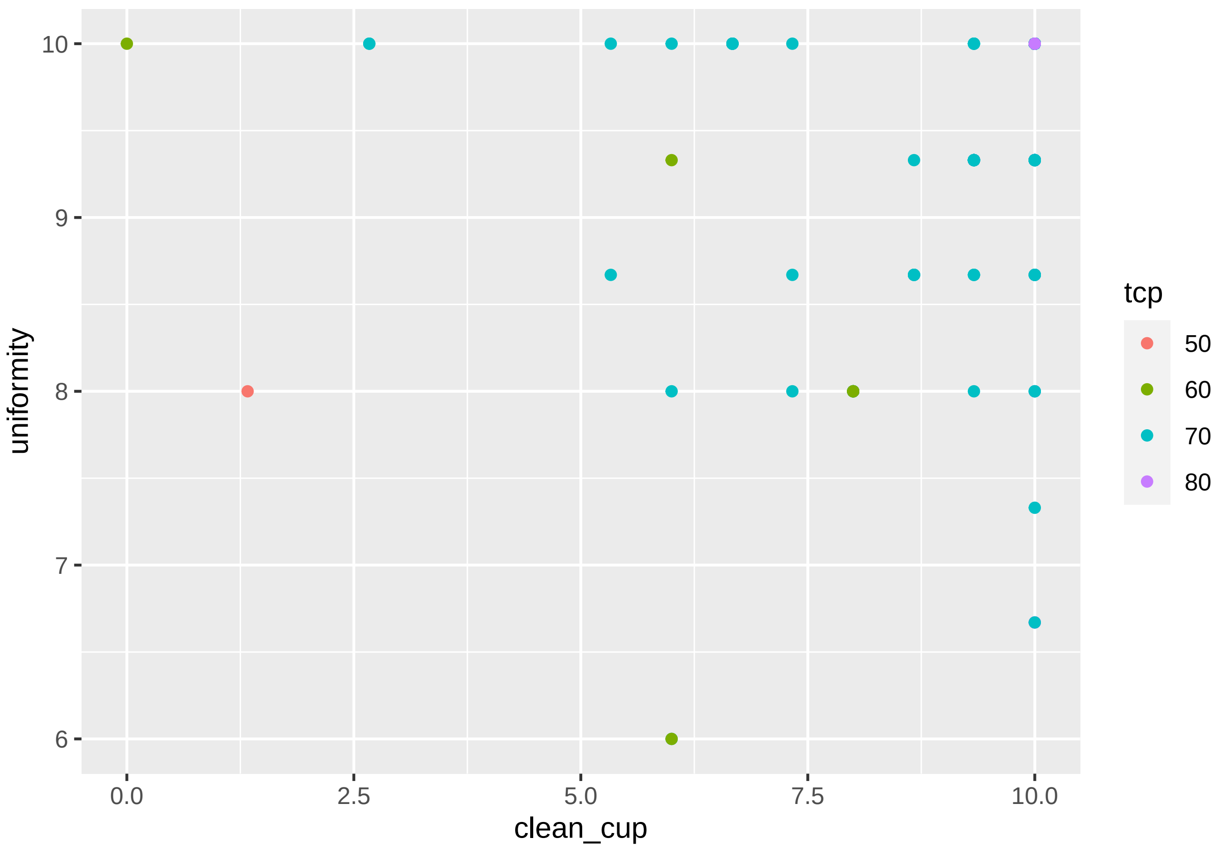
```
  acc = (confMat[1,1]+confMat[2,2]+confMat[3,3]+confMat[4,4])/sum(confMat)
  all.acc = rbind(all.acc,acc)
}

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

```
## [1] 0.9865169
```

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

This makes sense as the data has many types of fields, and not all of the fields are continuous.

```
ggplot(df,aes(x=clean_cup,y=uniformity,color=tcp))+geom_point()
```




```
  #+ facet_wrap(~processing_method,scales = "free")
```

From the decision tree, taking the top two parameters and placing them in a scatter plot and giving color to the points based on their classifier. It is very easy to see where the lines could be to separate the 70s and 80s bins.

Polynomial SVM

```
set.seed(1)
all.acc = numeric(0)
for(i in 1:folds){
  model = svm(tcp~.,df[blk != i,],kernel="polynomial",type="C")
  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] 0.9404494
```

```
table_accuracy[4,1] = mean(all.acc)
```

Wierd R Issue

```
#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

```
library(nnet)
set.seed(1)

all.acc = numeric(0)
for(i in 1:folds){
  model = 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))
```

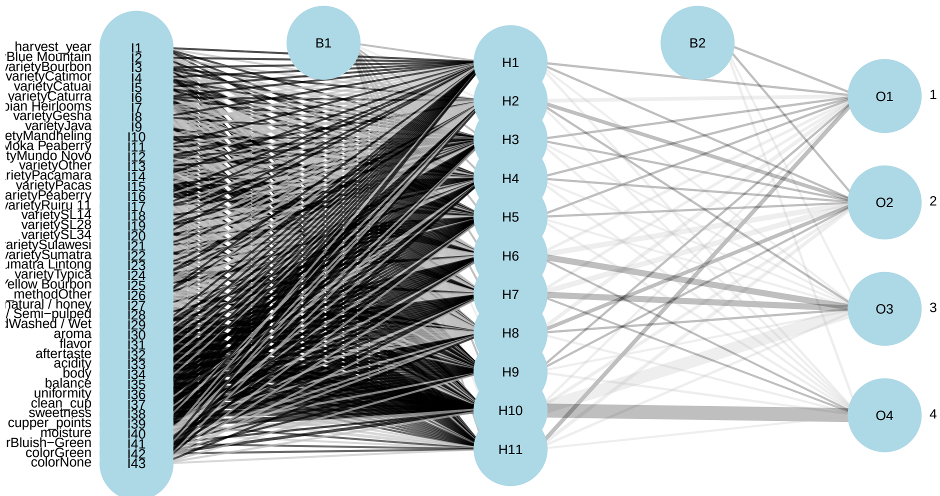
```
## [1] 0.8876404
```

```
table_accuracy[5,1] = 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

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



The above code was applied from then following link:

chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/viewer.html?pdfurl=https%3A%2F%2Fcran.r-project.org%2Fweb%2Fpackages%2FNeuralNetTools%2FNeur alNetTools.pdf&clen=142691&chunk=true

to be able to visualize a neural network.

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(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(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

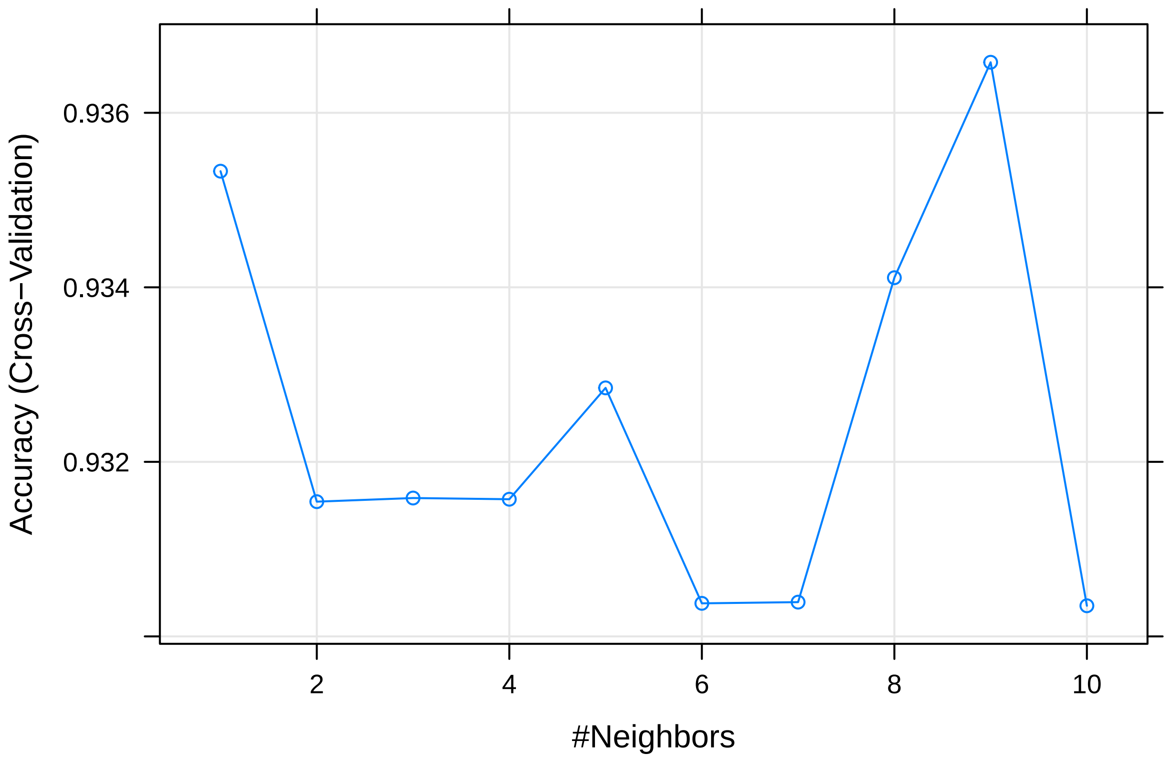
```r
set.seed(1)
df$tcp = as.factor(df$tcp)
library (caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
trControl <- trainControl(method  = "cv", number  = 10)
knn = df[,]
```

KNN

```r
set.seed(1)
model <- train(tcp ~ .,
             method     = "knn",
             tuneGrid   = expand.grid(k = 1:10),
             trControl  = trControl,
             data       = knn)
acc = mean(model$results$Accuracy)
table_accuracy[6,1] = 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#

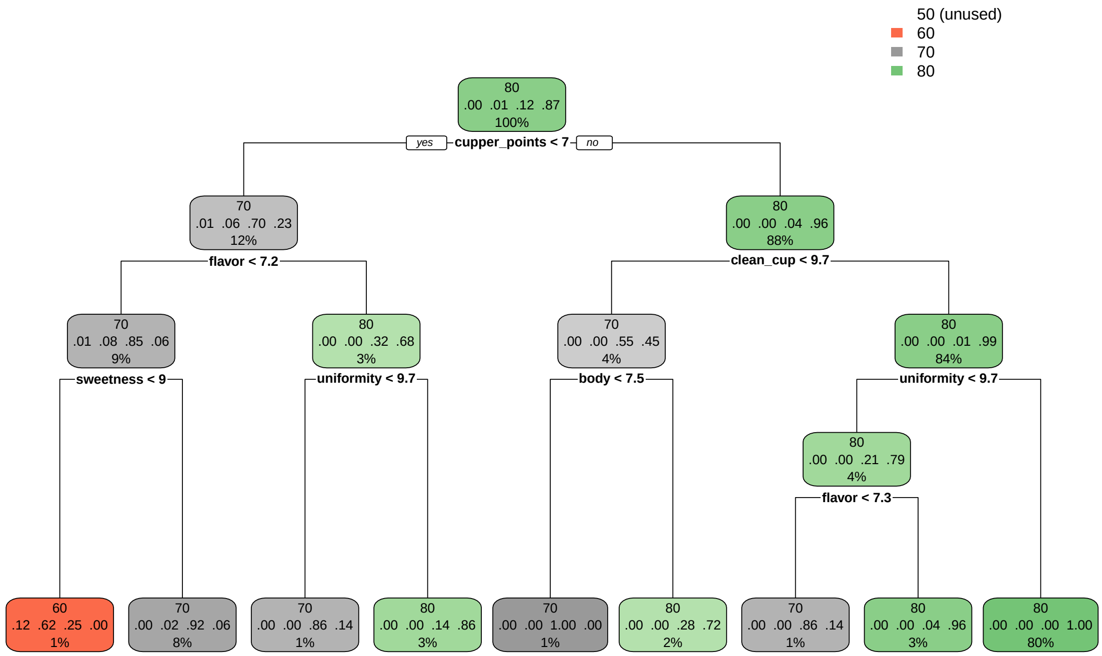
```
tab = round(table_accuracy,4)
tab
```

```
##                 Accuracy
## DTree             0.9517
## NB                0.9551
## SVM-Linerar       0.9865
## SVM-Polynomial    0.9404
## ANN               0.8876
## KNN               0.9325
```

Most of these predictive techniques did really well! Linear SVM having the highest accuracy, but due to the nature of the data, I do not believe it will be the most informative to an audience. This would suggest Decision Tree and Naive Bayes to be the next best options based on accuracy. As, decision trees are much easier to visualize and conceptually understand the flow of the diagram, this will be the preferred method for any further analysis and discussion.

##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)]
}
```

```
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)
```

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

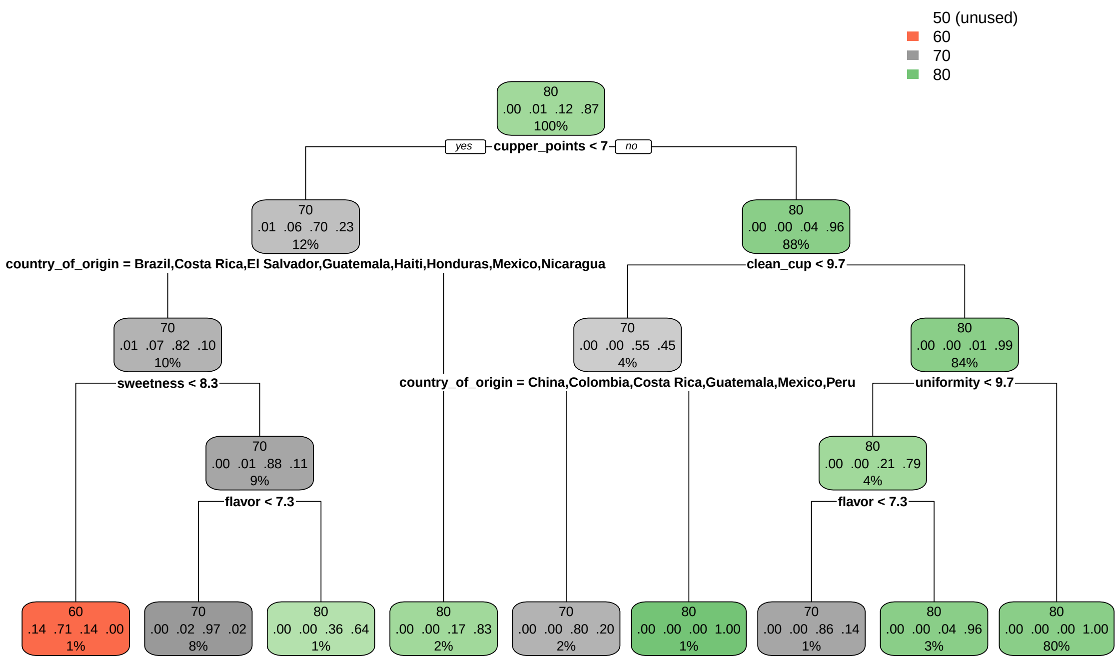
```
##  1  2  3  4  5  6  7  8  9 10 11
## 89 89 89 89 89 89 89 89 89 89  4
```

```r
set.seed(1)

all.acc = numeric(0)
for(i in 1:folds){
  tree = rpart(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))
```

```
## [1] 0.947191
```

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.

#Export Data to be used in Interactive Visuals#

```r
#write.csv(coffee,"coffee.csv", row.names = FALSE)
```