Project_cacao

Khasanova Dina and Olshanova Sasha

2023-04-12

Essential packages:

```
#install.packages("corrplot")
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
#install.packages(c('maptools', 'wordcloud'))
library(maptools)
```

```
## Loading required package: sp
```

```
## Checking rgeos availability: FALSE
## Please note that 'maptools' will be retired during 2023,
## plan transition at your earliest convenience;
## some functionality will be moved to 'sp'.
##      Note: when rgeos is not available, polygon geometry     computations in maptools depend on gpcli
##      which has a restricted licence. It is disabled by default;
##      to enable gpclib, type gpclibPermit()
```

```
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
#install.packages(c('rworldmap', 'corrplot'))
library(corrplot)
library(rworldmap)
```

```
## ### Welcome to rworldmap ###
```

```
## For a short introduction type :   vignette('rworldmap')
```

In this project we are going to perform exploratory data analysis by using "Chocolate Bar Rating Dataset". First of all, we read the data and investigate general information about it.

```
raw_chocolate_data <- read.csv("flavors_of_cacao.csv", header = TRUE)
str(raw_chocolate_data)
```

```
## 'data.frame':    1795 obs. of  9 variables:
##  $ Company...Maker.if.known.       : chr  "A. Morin" "A. Morin" "A. Morin" "A. Morin" ...
##  $ Specific.Bean.Origin.or.Bar.Name: chr  "Agua Grande" "Kpime" "Atsane" "Akata" ...
##  $ REF                             : int  1876 1676 1676 1680 1704 1315 1315 1315 1319 1319 ...
##  $ Review.Date                     : int  2016 2015 2015 2015 2015 2014 2014 2014 2014 2014 ...
##  $ Cocoa.Percent                   : chr  "63%" "70%" "70%" "70%" ...
##  $ Company.Location                : chr  "France" "France" "France" "France" ...
##  $ Rating                          : num  3.75 2.75 3 3.5 3.5 2.75 3.5 3.5 3.75 4 ...
##  $ Bean.Type                       : chr  " " " " " " " " ...
##  $ Broad.Bean.Origin               : chr  "Sao Tome" "Togo" "Togo" "Togo" ...
```

```
summary(raw_chocolate_data)
```

```
##  Company...Maker.if.known. Specific.Bean.Origin.or.Bar.Name      REF
##  Length:1795               Length:1795                      Min.   :   5
##  Class :character          Class :character                 1st Qu.: 576
##  Mode  :character          Mode  :character                 Median :1069
##                                                             Mean   :1036
##                                                             3rd Qu.:1502
##                                                             Max.   :1952
##   Review.Date   Cocoa.Percent      Company.Location       Rating
##  Min.   :2006   Length:1795        Length:1795        Min.   :1.000
##  1st Qu.:2010   Class :character   Class :character   1st Qu.:2.875
##  Median :2013   Mode  :character   Mode  :character   Median :3.250
##  Mean   :2012                                         Mean   :3.186
##  3rd Qu.:2015                                         3rd Qu.:3.500
##  Max.   :2017                                         Max.   :5.000
##   Bean.Type         Broad.Bean.Origin
##  Length:1795        Length:1795
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

By looking at general information about the data, we can sea that there are some issues with "Bean.Type", particularly there are no values. Also, type of "Cocoa.Percent" is not correct.

For now, let's explore our data.

```
#raw_chocolate_data

class(raw_chocolate_data)
```

```
## [1] "data.frame"
```

Tibble is a special type of table where the data is considered "tidy".However, in our case it's Data.frame, which means that our dataset has to be cleaned.

Some basic characteristics of our data:

```
# get dimensions, number of rows and columns
dim(raw_chocolate_data)
```

```
## [1] 1795    9
```

```
nrow(raw_chocolate_data)
```

```
## [1] 1795
```

```
ncol(raw_chocolate_data)
```

```
## [1] 9
```

```
# column names
names(raw_chocolate_data)
```

```
## [1] "Company...Maker.if.known."        "Specific.Bean.Origin.or.Bar.Name"
## [3] "REF"                              "Review.Date"
## [5] "Cocoa.Percent"                    "Company.Location"
## [7] "Rating"                           "Bean.Type"
## [9] "Broad.Bean.Origin"
```

DUBLICATES

Each data set has to be explored on the existence of the duplicates. Otherwise, it can lead to irrelevant results later.

```
raw_chocolate_data_no_dupl <- raw_chocolate_data %>% distinct()
nrow(raw_chocolate_data_no_dupl)
```

```
## [1] 1795
```

It means that there are no duplicates and it's possible to continue working with raw_chocolate_data. Moreover, we noticed that the data type for "Cocoa.Percent" column is "character", which is inappropriate to work with.In order to fix this we need to remove the percent sign and then convert the data type to a number (the code is illustrated below).

```
#removing sign of percent
raw_chocolate_data$Cocoa.Percent<-gsub("%","",as.character(raw_chocolate_data$Cocoa.Percent))
#head(raw_chocolate_data)
```

```
#changing data type
raw_chocolate_data <- transform(raw_chocolate_data, Cocoa.Percent = as.numeric(Cocoa.Percent))
#head(raw_chocolate_data)
```

MISSING VALUES

Let's see how much of our data is missing. We found out that the "Bean.Type" column has some empty cells, so let's see if we can figure out how many empty rows there are. Also, all other columns have to be checked.

```
sum(is.na(raw_chocolate_data))
```

```
## [1] 0
```

```
apply(raw_chocolate_data, 2, function(x) any(is.na(x)))
```

```
##        Company...Maker.if.known. Specific.Bean.Origin.or.Bar.Name
##                            FALSE                            FALSE
##                              REF                      Review.Date
##                            FALSE                            FALSE
##                    Cocoa.Percent                 Company.Location
##                            FALSE                            FALSE
##                           Rating                        Bean.Type
##                            FALSE                            FALSE
##                Broad.Bean.Origin
##                            FALSE
```

It seems like we do not have missing values, but we have already seen by observing our data that there are no values for each row in the "Bean.Type" column. By checking info about our data, it was found out that most of the rows in the "Bean.Type" column has just space: " ". Lets see how many rows are empty (contain space:" ").

```
Missing_Beans <- raw_chocolate_data %>% count(Bean.Type) %>% arrange(desc(n))
#print(Missing_Beans)
```

So, 887 items have just space (" ") value for the"Bean.Type column", and one item is (""). We can not just delete rows containing space (" ") value because it is 50 % of our data set.

The space (" ") value is not just a string, it's an object. By trying to find rows containing this value and using Google it was found out that this value is Unicode Character 'NO-BREAK SPACE'. Let's try to find which columns contain the same value, and how many items are there.

```
for (name in names(raw_chocolate_data)) {
  strange_values <- nrow(raw_chocolate_data[raw_chocolate_data[[name]] == "\u00A0", ])
  print(paste(name, strange_values))
}
```

```
## [1] "Company...Maker.if.known. 0"
## [1] "Specific.Bean.Origin.or.Bar.Name 0"
## [1] "REF 0"
## [1] "Review.Date 0"
## [1] "Cocoa.Percent 0"
## [1] "Company.Location 0"
## [1] "Rating 0"
## [1] "Bean.Type 887"
## [1] "Broad.Bean.Origin 73"
```

Eventually, two columns "Bean.Type" and "Broad.Bean.Origin" contain Unicode Character 'NO-BREAK SPACE' value. "Bean.Type" column has 887 items, and "Broad.Bean.Origin" has 73 items. Later, the plan is to find some correlations of "Bean.Type" and "Broad.Bean.Origin" features with other features, in order to substitute the Unicode Character 'NO-BREAK SPACE' value with something reasonable and not to lose big part of the data

“Broad.Bean.Origin” column

73 items in “Broad.Bean.Origin” are possible to delete as they don’t influence on the whole data set (1795 items). Also, we tried to find out the way to replace this values with relevant ones from the same column. But the thing is that there are no any strong correlations of this column with other ones, so we decided to delete them.

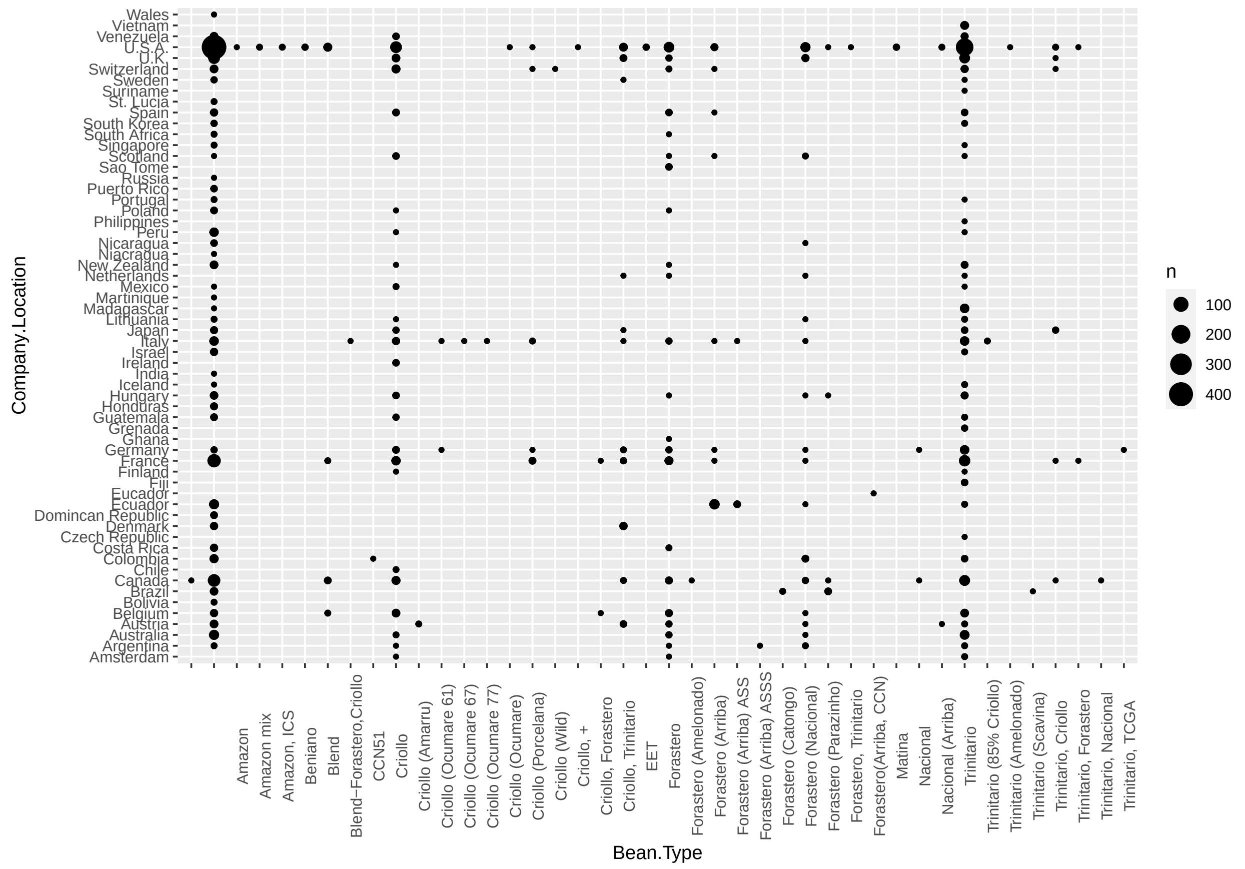
```
raw_chocolate_data <- raw_chocolate_data[raw_chocolate_data$Broad.Bean.Origin != "\u00A0", ]
#raw_chocolate_data
```

Result: We decreased number of rows by 73, so for now there 1722 rows to work with in raw_chocolate_data.

Bean.Type column

Almost a half of our data set has the missing value for the Bean.Type column. We had an idea that “Bean.Type” somehow depends on”Company.Location”. In order to explore this idea ggplot was used.

```
plot_bean_type <- ggplot(data = raw_chocolate_data) +
  geom_count(mapping = aes(x = Bean.Type, y = Company.Location))

plot_bean_type + theme(axis.text.x = element_text(angle = 90))
```

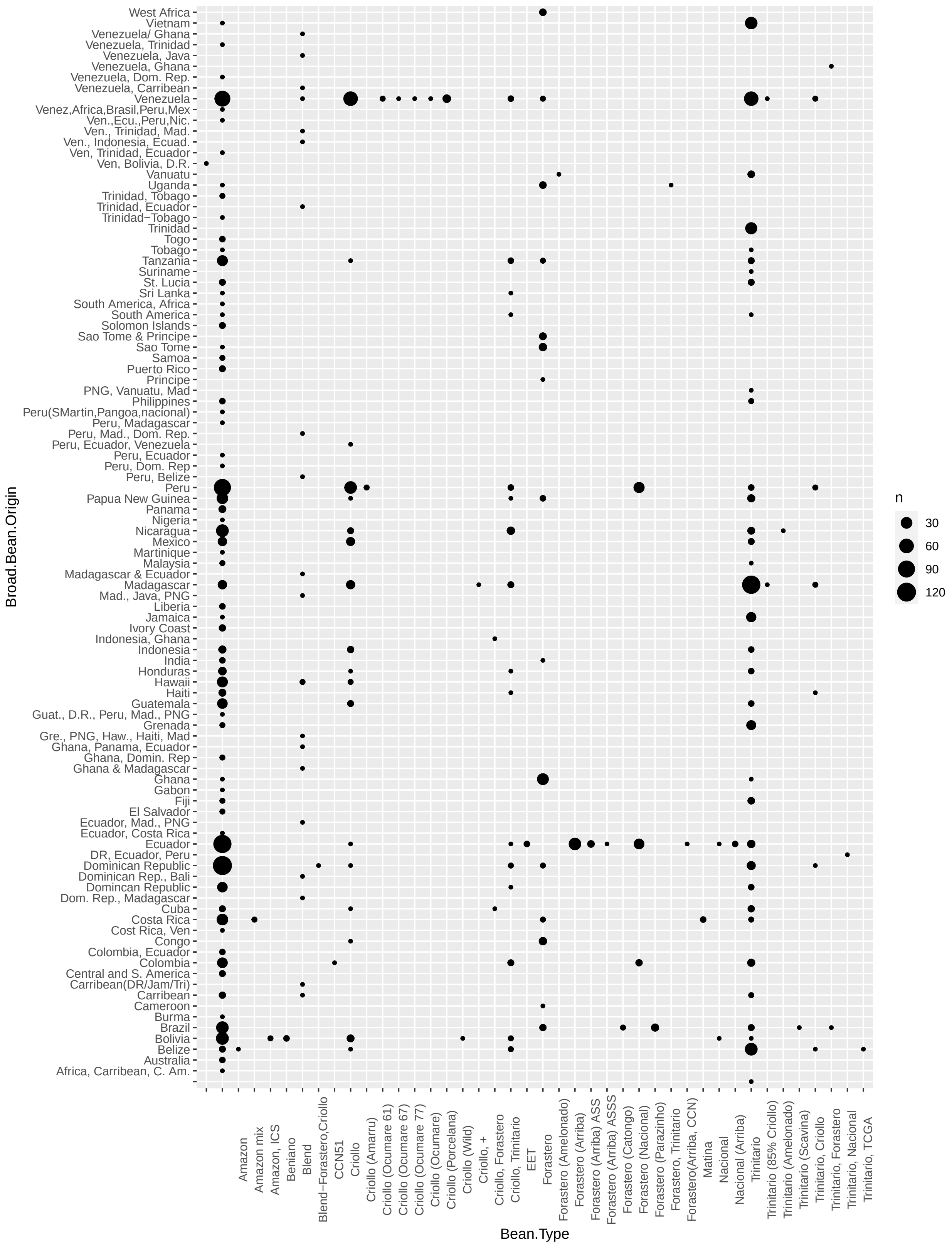


Result: Each country has several types of beans, so the hypothesis that each country has its own bean type did not work in this case.

By looking on the data set generated for one country, it was notices that “Bean.Type” could somehow depend on “Broad.Bean.Origin”. So, new ggplot was generated to find the correlation between these features.

```r
bean_type <- raw_chocolate_data[raw_chocolate_data$Company.Location == "France", ]
#bean_type
```

```r
plot_bean_origin <- ggplot(data = raw_chocolate_data) +
  geom_count(mapping = aes(x = Bean.Type, y = Broad.Bean.Origin))

plot_bean_origin + theme(axis.text.x = element_text(angle = 90))
```



It seems that there is a correlation between "Bean.Type" and "Broad.Bean.Origin". We can find out the most common origin for each bean type, and then fill empty spaces in "Bean.Type" column based on their origin.

First, it is essential to define all existent bean types (empty_list).

```
bean_type_list <- unique(raw_chocolate_data$Bean.Type)
#bean_type_list
```

```
empty_list <- list()
item <- list()

for (i in bean_type_list){
  if (i == "\u00A0" | i == "") print("no bean types")
  else
    item <- c(i)
    empty_list <- append(empty_list, item)
}
```

```
## [1] "no bean types"
## [1] "no bean types"
```

Now, let's create a DataFrame, which has "Bean.Type" column and corresponding "Broad.Bean.Origin" (bean_type_origin DataFrame).

```
columns = c("Bean.Type","Broad.Bean.Origin")
bean_type_origin = data.frame(matrix(nrow = 0, ncol = length(columns)))
colnames(bean_type_origin) = columns

for (i in empty_list){

  Forastero <- raw_chocolate_data[raw_chocolate_data$Bean.Type == i, ]

  grp_tbl <- Forastero %>% group_by(Broad.Bean.Origin)%>%
  summarise(total_count=n(),
            .groups = 'drop')%>%
  as.data.frame()

  newdata <- grp_tbl[order(grp_tbl$total_count, decreasing=TRUE), ]

  bean_origin = newdata$Broad.Bean.Origin[1]

  bean_type_origin[nrow(bean_type_origin) + 1,] <- c(i,bean_origin)

  #print(paste(i,bean_origin))
}

head(bean_type_origin)
```

```
##              Bean.Type Broad.Bean.Origin
## 1              Criollo         Venezuela
## 2           Trinitario        Madagascar
## 3   Forastero (Arriba)           Ecuador
## 4            Forastero             Ghana
## 5 Forastero (Nacional)              Peru
## 6  Criollo, Trinitario         Nicaragua
```

Here, we will substitute Unicode Character 'NO-BREAK SPACE' and " " values in "Bean.Type" column with corresponding values from previous DataFrame(bean_type_origin DataFrame).

```
P <- nrow(raw_chocolate_data)
number = 0
for(i in 1: P){
  bean_origin = raw_chocolate_data$Broad.Bean.Origin[i]
  bean_type_r = raw_chocolate_data$Bean.Type[i]
  #print(bean_type_r)
  if (bean_type_r == "\u00A0" | bean_type_r == "")
  number = number + 1
  table = bean_type_origin[bean_type_origin$Broad.Bean.Origin == bean_origin, ]
  value = table$Bean.Type[1]
  #print(value)
  raw_chocolate_data$Bean.Type[i] <- value
}

print(number)
```

```
## [1] 837
```

```
sum(is.na(raw_chocolate_data$Bean.Type))
```

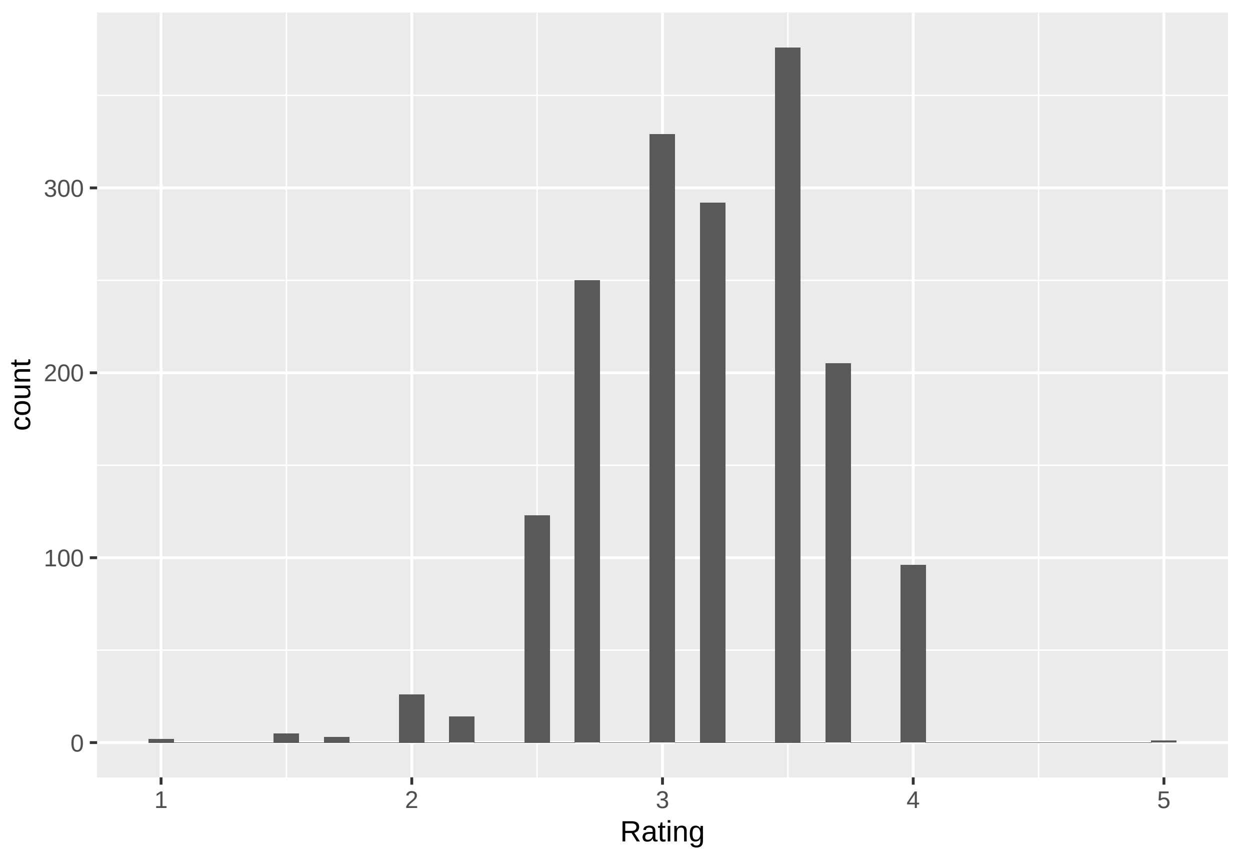
```
## [1] 474
```

Eventually, 837 Unicode Character 'NO-BREAK SPACE' and " " values were found. And 474 left as Nan values. Now only 474 Nan values in "Bean.Type" column in raw_chocolate_data. For all graphs where "Bean.Type" column is not essential, we will use raw_chocolate_data. For graphs where "Bean.Type" column is vital, the following DataFrame will be used(Bean_type_without_NAN).

```
Bean_type_without_NAN <- raw_chocolate_data[!is.na(raw_chocolate_data$Bean.Type), ]
#Bean_type_without_NAN
```

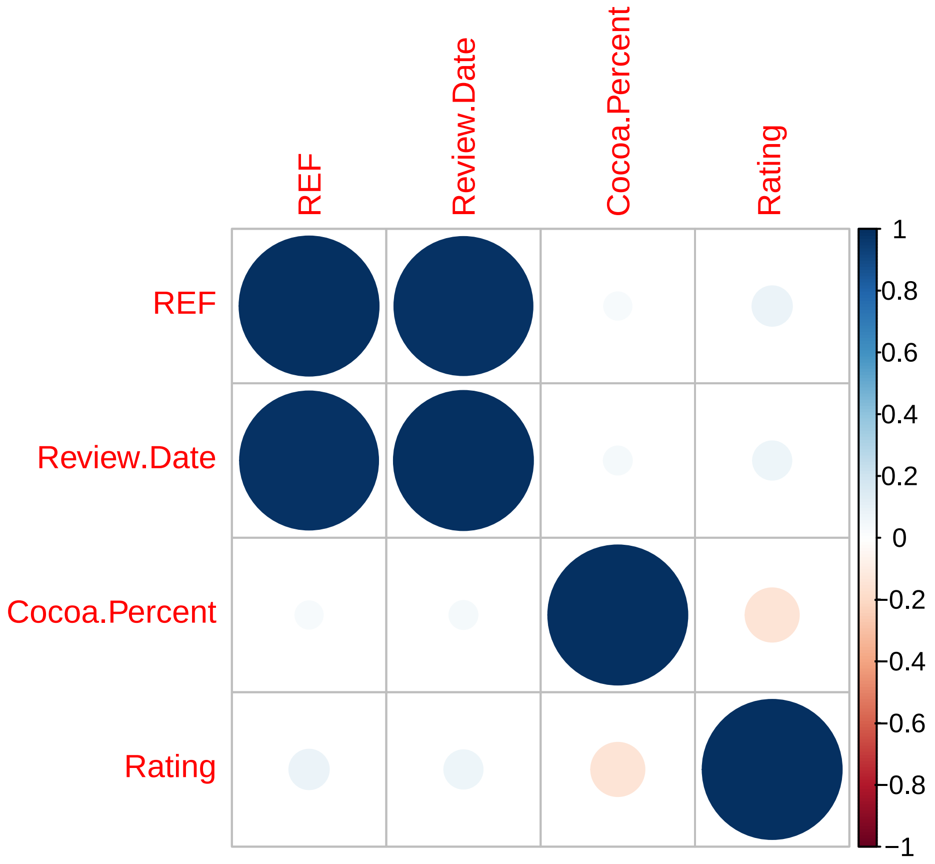
In this work we decided to focus on rating of chocolate. So, let's explore the relationship of this feature with other properties. First of all, it is essential to understand the distribution of the values in the "Rating" column.

```
ggplot(data = raw_chocolate_data) +
  geom_histogram(mapping = aes(x = Rating), binwidth = 0.1)
```



Second, we decided to investigate the correlation between numeric value's columns. It seems that there is only one interesting correlation for us related to "Rating" column. It is "Cocoa.Percent" column.

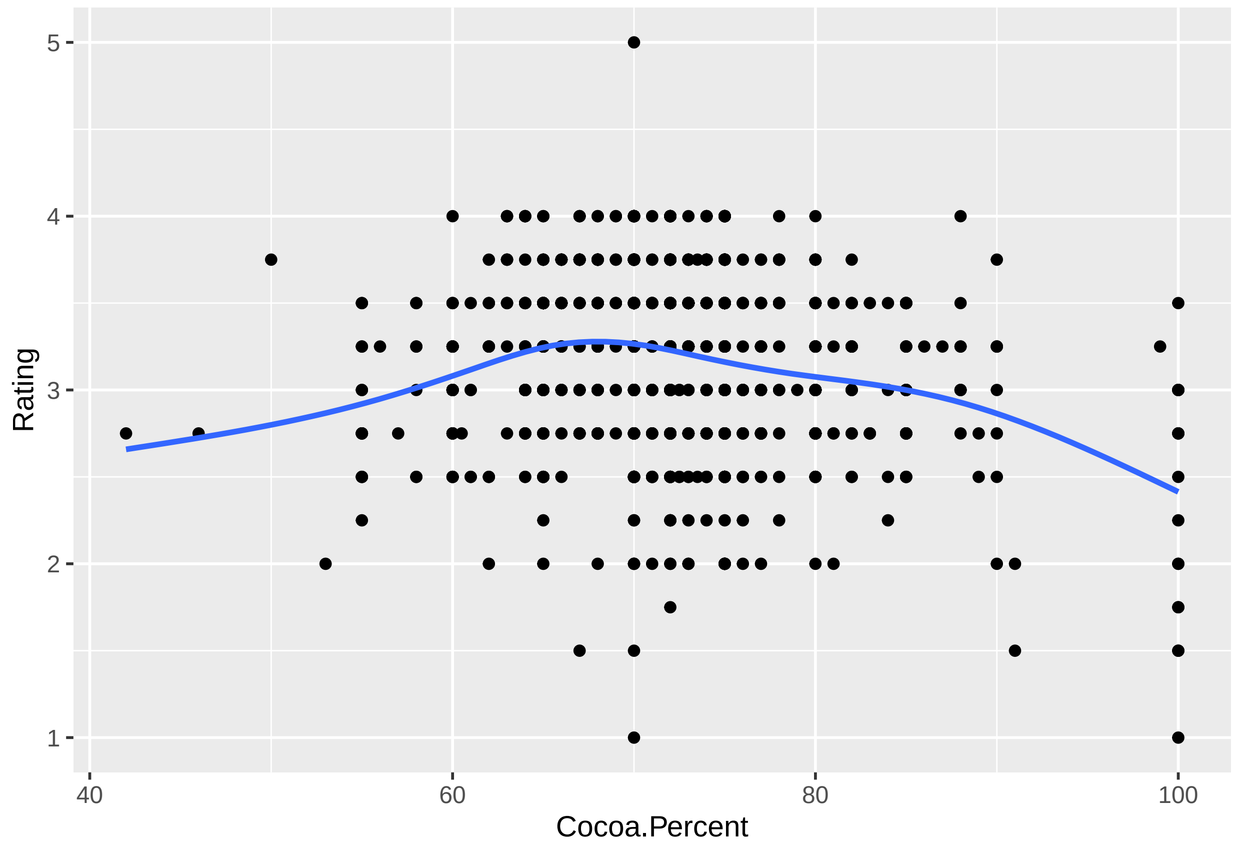
```
chololate_data_for_corr <- raw_chocolate_data[, c("REF", "Review.Date", "Cocoa.Percent", "Rating")]


mydata.cor = cor(chololate_data_for_corr)
corrplot(mydata.cor)
```



In order to investigate the relationship between "Rating" column and "Cocoa.Percent" ggplot was built.

```
# plotting a scatter plot (with smoothing curve)
ggplot(raw_chocolate_data, aes(x=Cocoa.Percent, y=Rating)) + geom_point() + geom_smooth(se=FALSE)
```

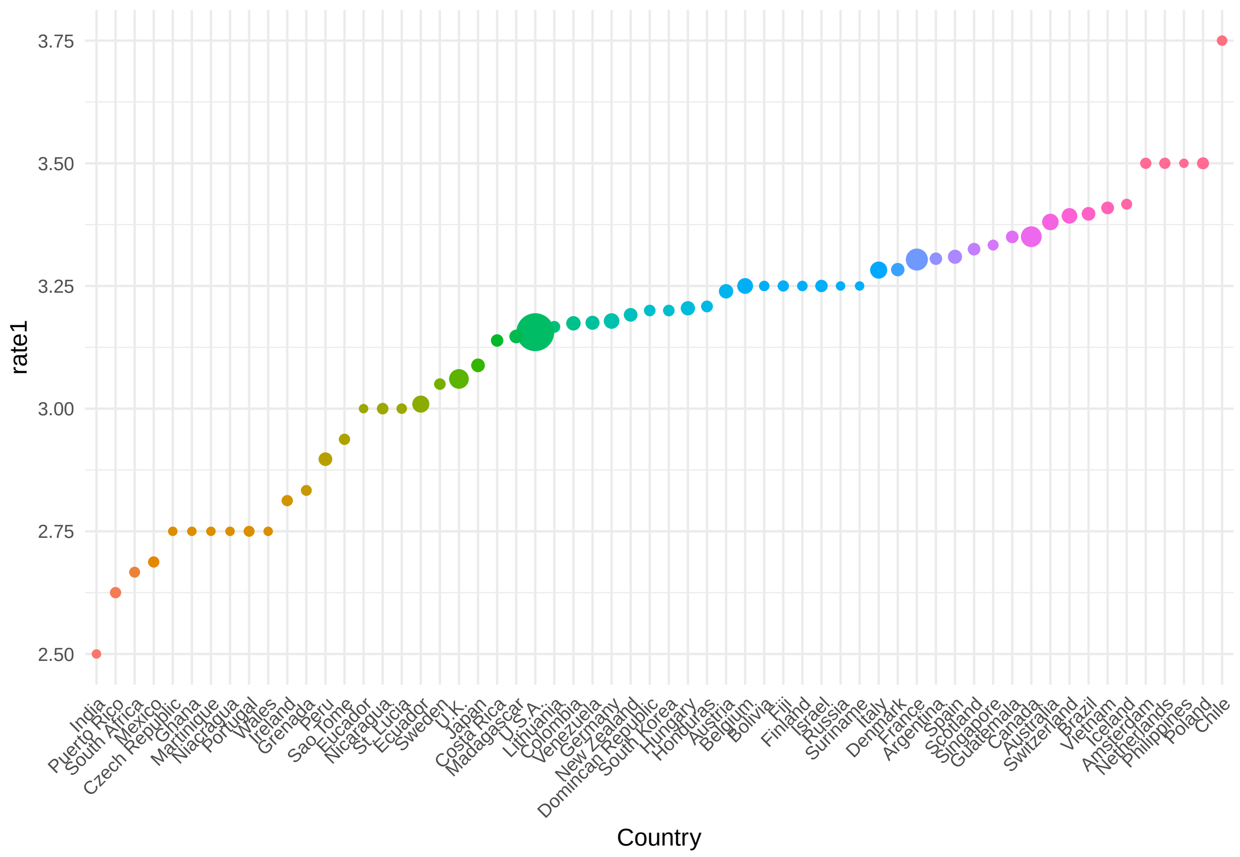
```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



It seems that there is no linear correlation between percent of cocoa in the chocolate and the rating. The Best rating as 5 the chocolate with 70% of cocoa has and the one with 100% of chocolate has a rating as 1.

Numerical values do not influence a lot on our Rating. So, for now we decided to observe the influence of the company location.

```
loca <- group_by(raw_chocolate_data, Company.Location)
good <- summarise(loca,  count=n(),
                  rate1= mean(Rating))

good1<- arrange(good, desc(rate1))

ggplot(good1,aes(x=reorder(Company.Location,rate1), y=rate1)) +geom_point(aes(size=count, colour=factor
  theme(axis.text.x = element_text(angle = 45, hjust = 1) , legend.position="none") +
  labs(x="Country", "Chocolate Rating", "Chocolate Rating vs Country")
```



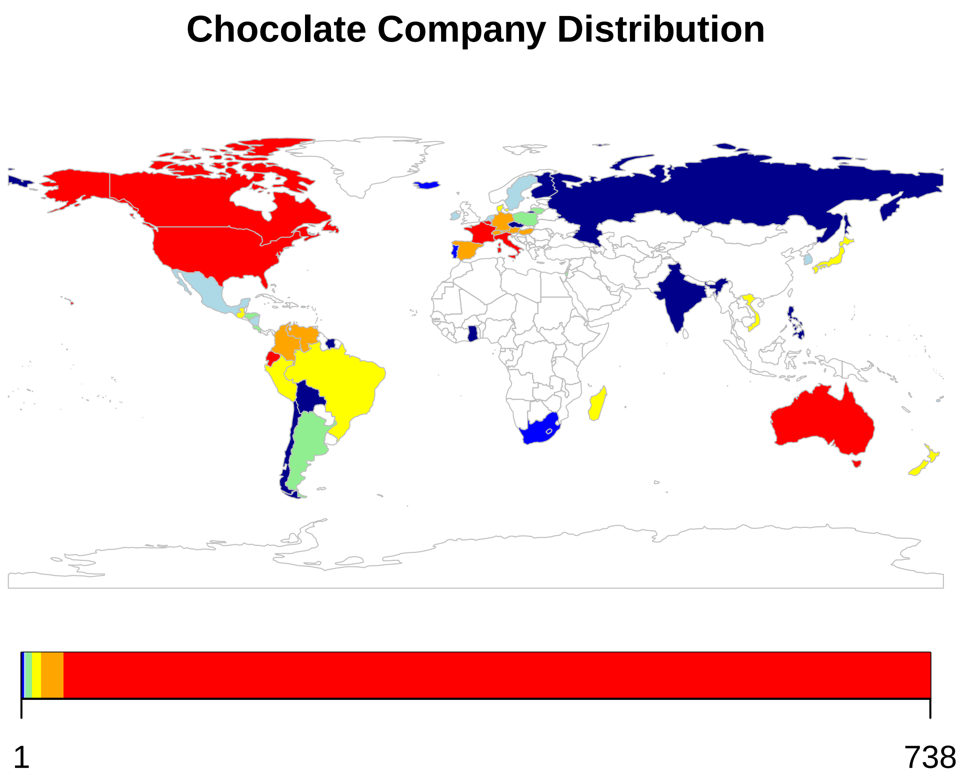
Using this plot we identify which country in average has the highest rating. It is obvious that there is a strong dependence of rating the the country.

Let's see how many companies in each country.Now we can check where the highest rated company located.

```
commap <- group_by(raw_chocolate_data, Company.Location)

commap1 <- summarise(commap,  count=n())

map1 <- joinCountryData2Map(commap1, joinCode="NAME", nameJoinColumn="Company.Location")
```

```
## 51 codes from your data successfully matched countries in the map
## 9 codes from your data failed to match with a country code in the map
## 192 codes from the map weren't represented in your data
```

```
mapCountryData(map1, nameColumnToPlot="count", mapTitle="Chocolate Company Distribution" , colourPalett
```



By this map we can clearly see that there are a lot of companies are located in America, Canada and Australia. In the flip side in Russia and India the amount of companies is small.

To be honest, we were interested where we can find the best chocolate in the world. If we take into the account the previous plot, we can see that the well-rated chocolate we are able to find in Chile (based on average rating score along all countries). Does it meat that the best rated chocolate company is located in Chile? Let's have a look.

```
raw_chocolate_data_rat <- raw_chocolate_data[order(raw_chocolate_data$Rating, decreasing=TRUE), ]
raw_chocolate_data_rat[1, ]
```

```
##    Company...Maker.if.known. Specific.Bean.Origin.or.Bar.Name REF Review.Date
## 79                    Amedei                            Chuao 111        2007
##    Cocoa.Percent Company.Location Rating Bean.Type Broad.Bean.Origin
## 79            70            Italy      5   Criollo         Venezuela
```

Finally, if we want to buy the best rated chocolate, we need to go to Italy, but search for this particular company. Or another option is to find less rated chocolate in other country, but the probability to find it among all companies in another country will be higher compared with Italy.

Classification

Rating classification was made to categorize rating value for chocolate.
We'll call some the "bad" ones, "good" and some the "excellent" ones. In order to do this, we have created new column "Class_rating", and filled it with categories discussed above.

```
#new column
raw_chocolate_data<- raw_chocolate_data %>%
  mutate(Class_rating = "1")
```

```
#filling the column with categories
number_ratting_rows <- nrow(raw_chocolate_data)

for(i in 1: number_ratting_rows){
rating <- raw_chocolate_data$Rating[i]
if (rating <= 2.5){
  raw_chocolate_data$Class_rating[i] <- "bad"
}else if(2.5 < rating &rating < 3.5){
  raw_chocolate_data$Class_rating[i] <- "good"
}else{
  raw_chocolate_data$Class_rating[i] <- "exellent"
}}
```

For future PCA analysis we created a subset of the original data set containing only numeric value's columns, as PCA can be performed only by using them. We have chosen columns "REF", "Review.Date", "Cocoa.Percent" and the "Class_rating" column which is important for classification task.

chololate_data_numeric is a new dataframe which contains only columns with numeric values, and "Class_rating".

```
numeric_vars <- c("REF", "Review.Date", "Cocoa.Percent", "Class_rating")

chololate_data_numeric <- raw_chocolate_data[, numeric_vars]

head(chololate_data_numeric)
```

```
##    REF Review.Date Cocoa.Percent Class_rating
## 1 1876        2016            63     exellent
## 2 1676        2015            70         good
## 3 1676        2015            70         good
## 4 1680        2015            70     exellent
## 5 1704        2015            70     exellent
## 6 1315        2014            70         good
```

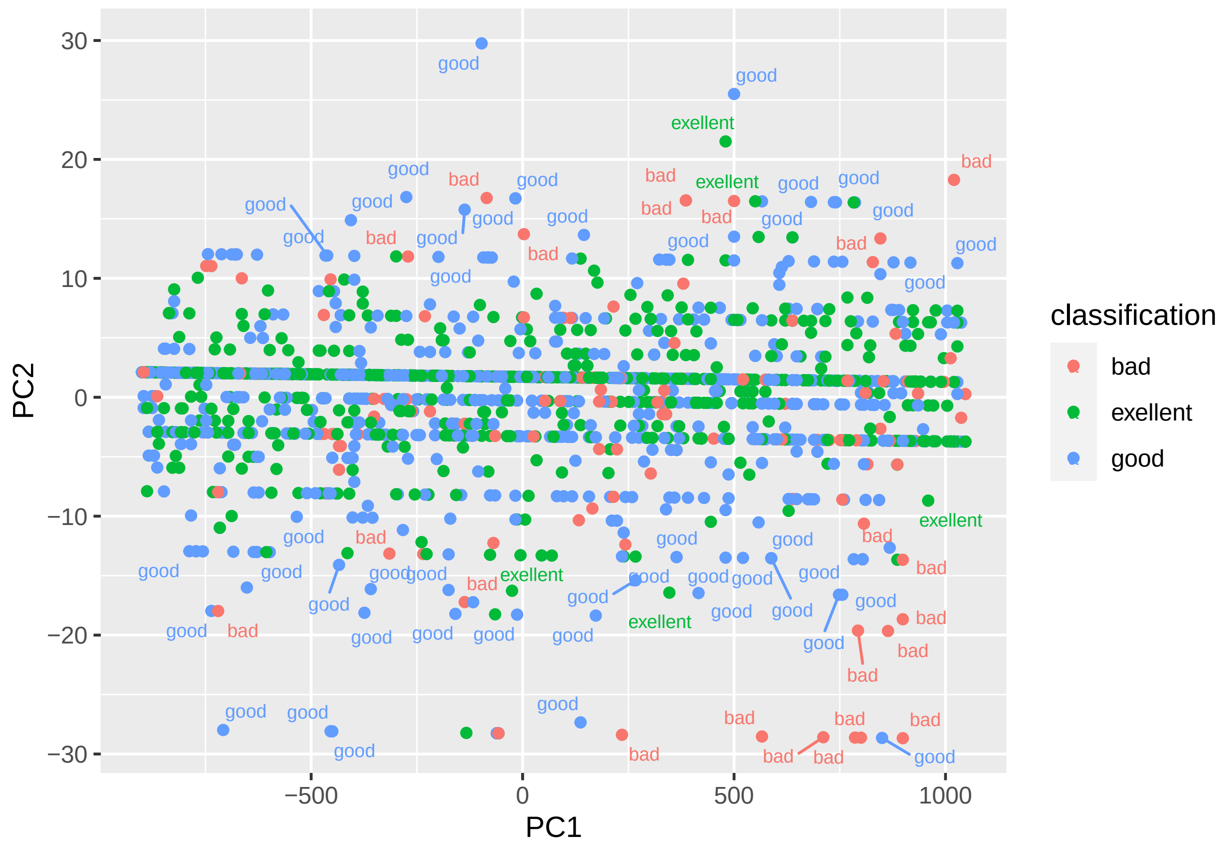
PCA

We performed PCA and t-SNE analysis by using chololate_data_numeric, but they couldn't give us reasonable result as 3 numeric columns are not enough description of the data. ggplots of these two method are illustrated below.

```
# Preform PCA on our data
prcomp_choco_num <- prcomp(chololate_data_numeric[, 1:3])
pca_choco_num <- data.frame(
  PC1 = prcomp_choco_num$x[, 1],
  PC2 = prcomp_choco_num$x[, 2],
  classification = chololate_data_numeric$Class_rating,
  label = chololate_data_numeric$Class_rating
)
```

```r
ggplot(pca_choco_num, aes(x = PC1, y = PC2, label = label, col = classification)) +
  geom_point() +
  ggrepel::geom_text_repel(cex = 2.5)
```

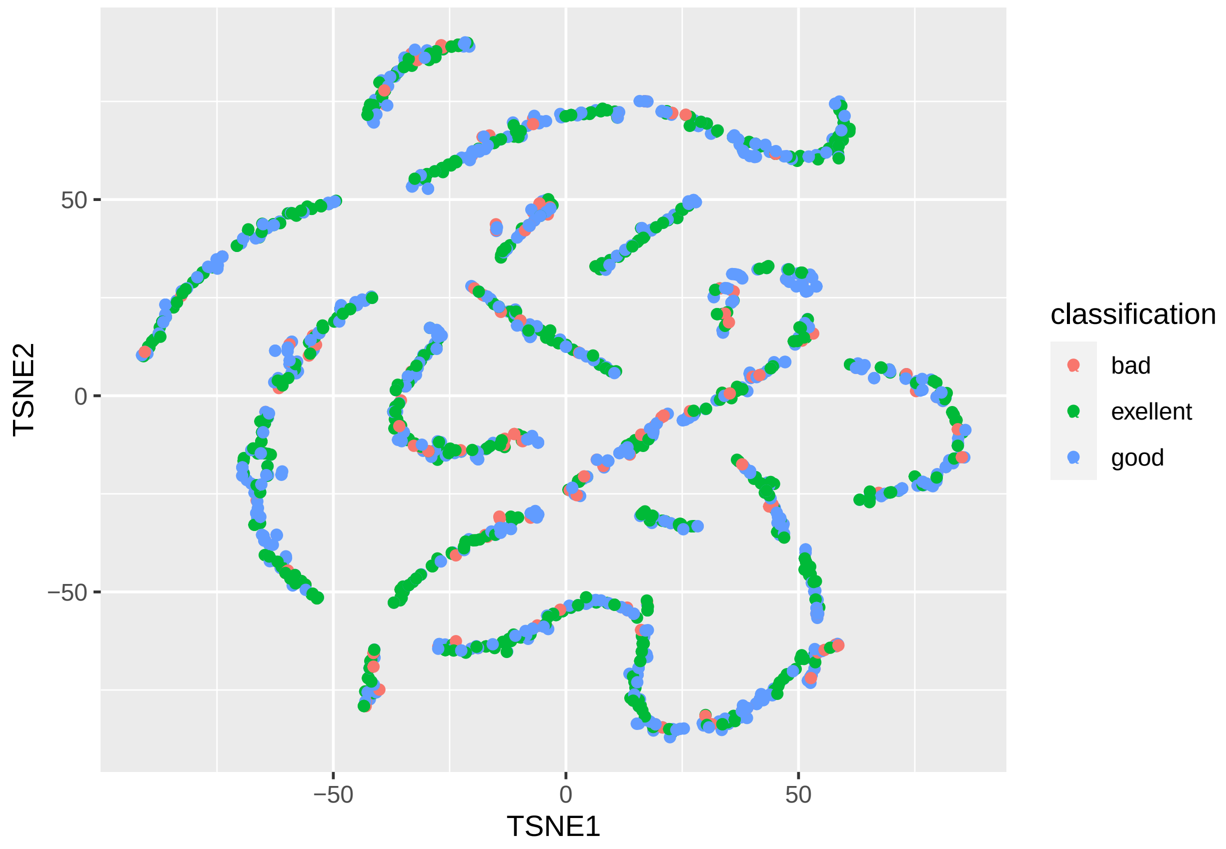
```
## Warning: ggrepel: 1650 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



```r
# Preform TSNE on our data

library(Rtsne)

#prcomp_choco_num <- prcomp(chololate_data_numeric[, 1:3])

tsne_chololate_num <- Rtsne(chololate_data_numeric[, 1:3],
  pca = FALSE, perplexity = 10,
  theta = 0.0, check_duplicates = FALSE
)

tsne_chololate_num <- data.frame(
  TSNE1 = tsne_chololate_num$Y[, 1],
  TSNE2 = tsne_chololate_num$Y[, 2],
  label = chololate_data_numeric$Class_rating,
  classification = chololate_data_numeric$Class_rating
)
```

```
ggplot(tsne_chololate_num, aes(
  x = TSNE1, y = TSNE2,
  label = label, col = classification
)) +
  geom_point() +
  ggrepel::geom_text_repel(cex = 2.5)
```

```
## Warning: ggrepel: 1722 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



So, we decided to use other features (object columns). By implementing dummyVars function, we translated text data into numerical data for PCA and t-SNE analysis.

df_transformed is a new dataframe after applying dummyVars function.

```
library(caret)
```

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

```
##
## Attaching package: 'caret'
```

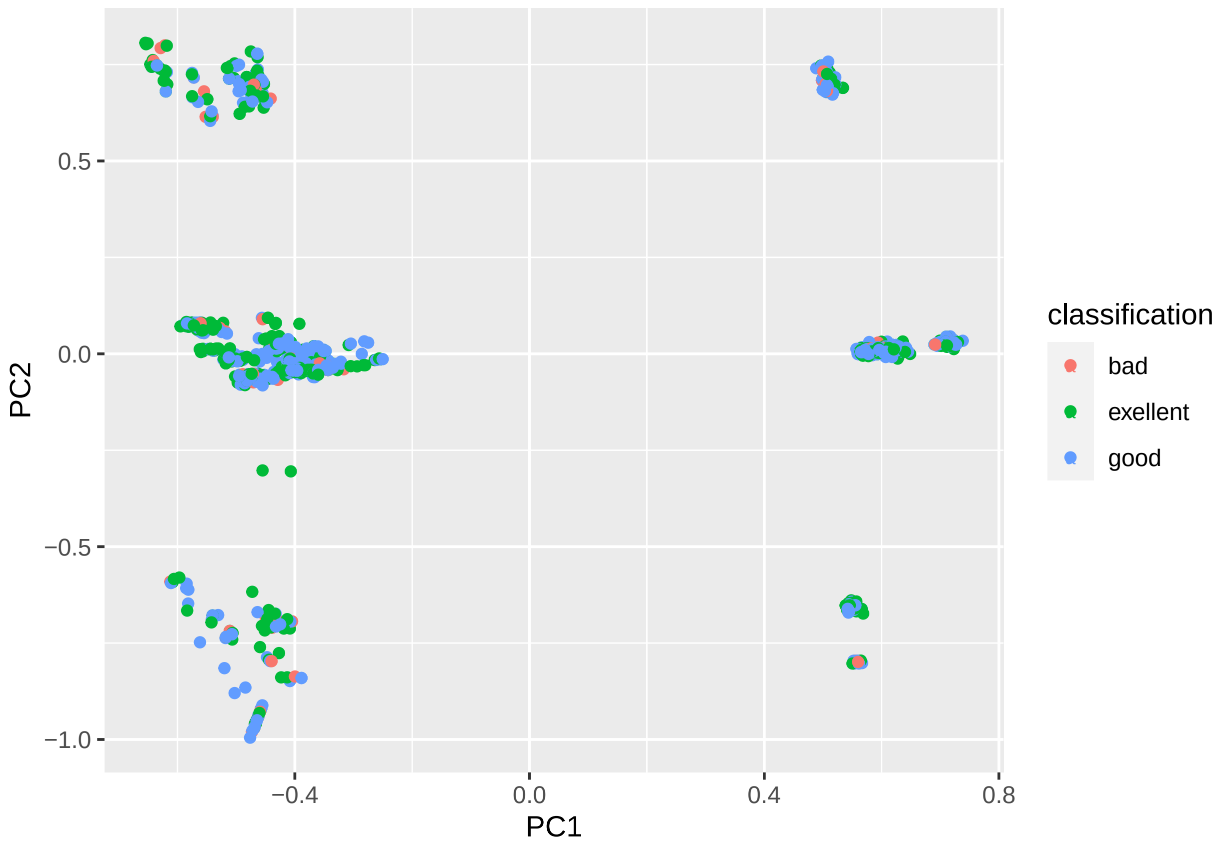
```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
df_dummies <- dummyVars(~ Company...Maker.if.known. + Company.Location + Specific.Bean.Origin.or.Bar.Na

df_transformed <- data.frame(predict(df_dummies, newdata = raw_chocolate_data))
```

We performed PCA and t-SNE analysis by using df_transformed. In this case we got better results, and we could clearly see that our data can be divided into different groups. ggplots of these two method are illustrated below.

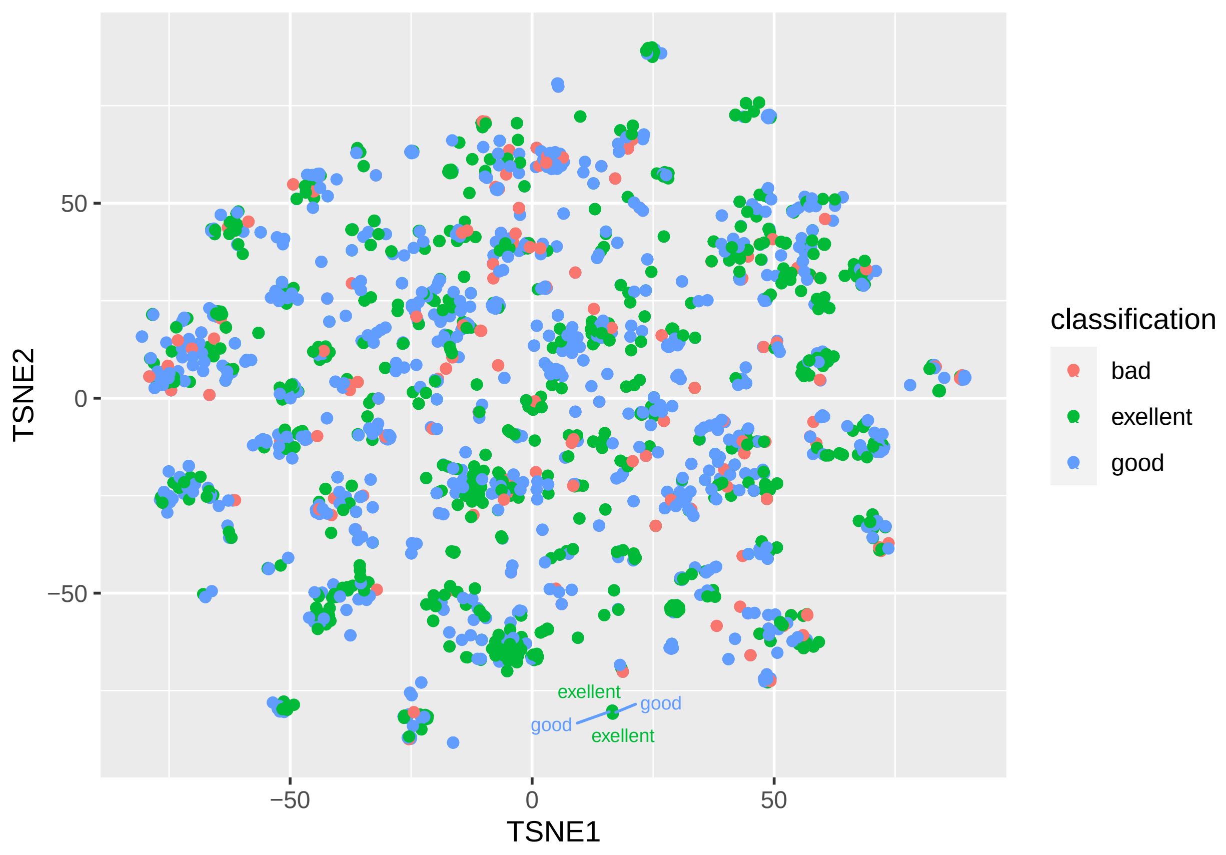
```
# Preform PCA on our data
prcomp_choco_num <- prcomp(df_transformed)
pca_choco_num <- data.frame(
  PC1 = prcomp_choco_num$x[, 1],
  PC2 = prcomp_choco_num$x[, 2],
  classification = chololate_data_numeric$Class_rating,
  label = chololate_data_numeric$Class_rating
)

ggplot(pca_choco_num, aes(x = PC1, y = PC2, label = label, col = classification)) +
  geom_point() +
  ggrepel::geom_text_repel(cex = 2.5)
```

```
## Warning: ggrepel: 1722 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



```r
# Preform t-SNE on our data

library(Rtsne)


tsne_chololate_num <- Rtsne(df_transformed,
  pca = FALSE, perplexity = 10,
  theta = 0.0, check_duplicates = FALSE
)

tsne_chololate_num <- data.frame(
  TSNE1 = tsne_chololate_num$Y[, 1],
  TSNE2 = tsne_chololate_num$Y[, 2],
  label = chololate_data_numeric$Class_rating,
  classification = chololate_data_numeric$Class_rating
)

ggplot(tsne_chololate_num, aes(
  x = TSNE1, y = TSNE2,
  label = label, col = classification
)) +
  geom_point() +
  ggrepel::geom_text_repel(cex = 2.5)
```

```
## Warning: ggrepel: 1718 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



In order to make PCA and t-SNE on initial data set, we merged df_transformed (data set after dummyVars function) and chololate_data_numeric (only numeric columns from initial data set) based on their index.

```
# creating the same column with index in both dataframe for merging

df_transformed <-df_transformed %>%
  mutate(index = "1")

chololate_data_numeric <- chololate_data_numeric %>%
  mutate(index = "1")
```

```
# filling index column with indexes

numb_rows <- nrow(df_transformed)


for(i in 1: numb_rows) {
  df_transformed$index[i] <- i
  chololate_data_numeric$index[i] <- i

}
head(chololate_data_numeric)
```

```
##    REF Review.Date Cocoa.Percent Class_rating index
## 1 1876        2016            63     exellent     1
## 2 1676        2015            70         good     2
## 3 1676         2015              70          good     3
## 4 1680         2015              70      exellent     4
## 5 1704         2015              70      exellent     5
## 6 1315         2014              70          good     6
```

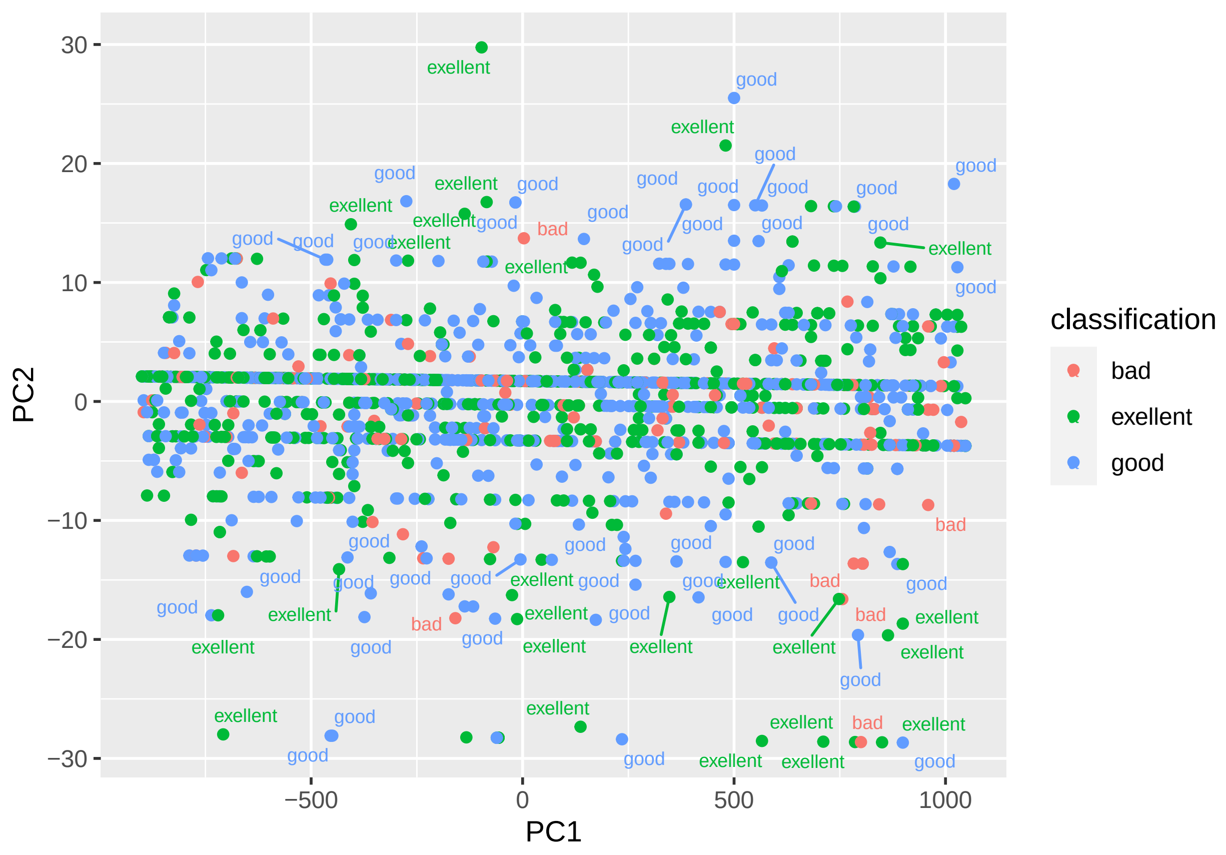
```
#merge

jointdataset <- merge(chololate_data_numeric, df_transformed, by = 'index')
#head(jointdataset)
```

```
# deleting columns we don't need to use for PCA and t-SNE
join_new <- jointdataset[ , !names(jointdataset) %in%
    c("Class_rating","index")]
```

So, we performed PCA and t-SNE on final data, and we got almost the same result, as using only numerical columns. It means that they influence a lot, and this columns have to be scaled to work with them later.

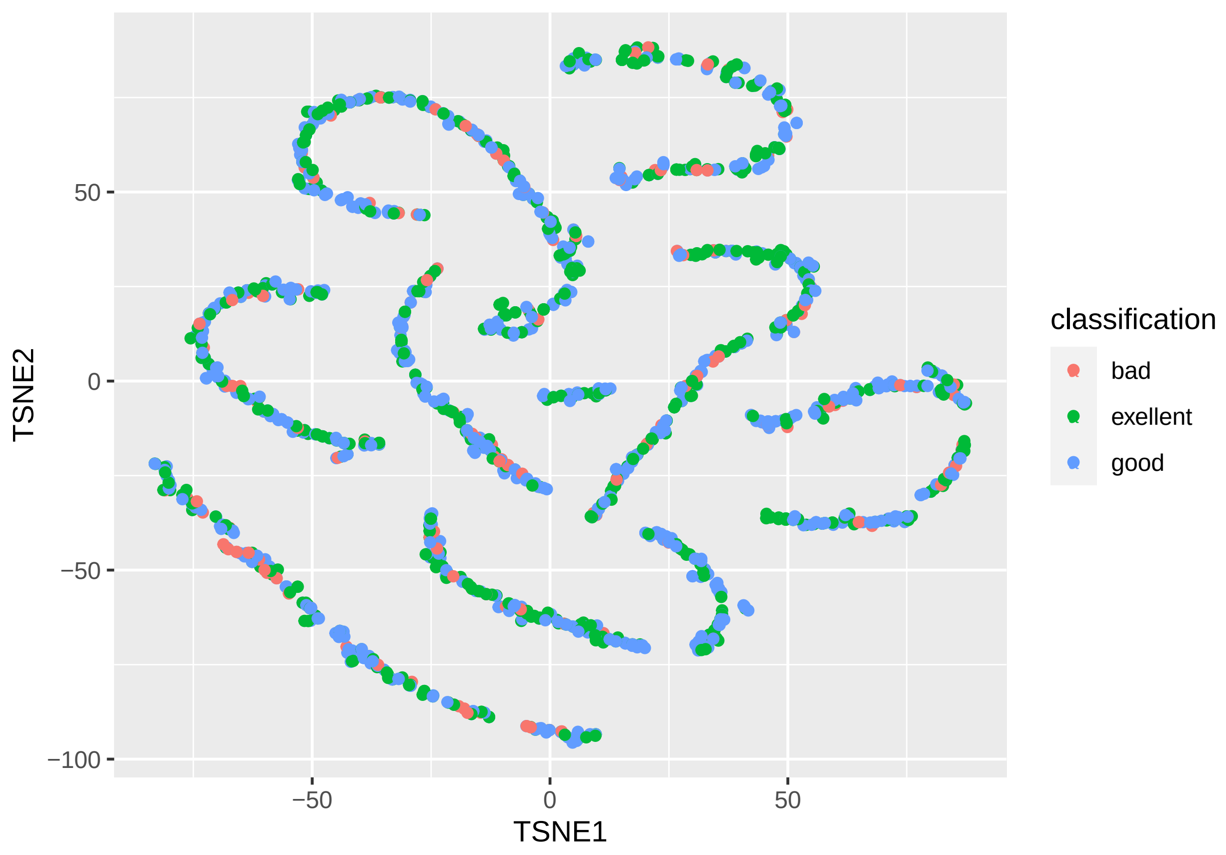
```
# Preform PCA on our data
prcomp_choco_num <- prcomp(join_new)
pca_choco_num <- data.frame(
  PC1 = prcomp_choco_num$x[, 1],
  PC2 = prcomp_choco_num$x[, 2],
  classification = chololate_data_numeric$Class_rating,
  label = chololate_data_numeric$Class_rating
)

ggplot(pca_choco_num, aes(x = PC1, y = PC2, label = label, col = classification)) +
  geom_point() +
  ggrepel::geom_text_repel(cex = 2.5)
```

```
## Warning: ggrepel: 1651 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



```
# Preform t-SNE on our data

library(Rtsne)


tsne_chololate_num <- Rtsne(join_new,
  pca = FALSE, perplexity = 10,
  theta = 0.0, check_duplicates = FALSE
)

tsne_chololate_num <- data.frame(
  TSNE1 = tsne_chololate_num$Y[, 1],
  TSNE2 = tsne_chololate_num$Y[, 2],
  label = chololate_data_numeric$Class_rating,
  classification = chololate_data_numeric$Class_rating
)

ggplot(tsne_chololate_num, aes(
  x = TSNE1, y = TSNE2,
  label = label, col = classification
)) +
  geom_point() +
  ggrepel::geom_text_repel(cex = 2.5)
```

```
## Warning: ggrepel: 1722 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```


TSNE2
50
0
−50
−100
−50
0
50
TSNE1
classification
bad
exellent
good