Chocolate\_Analysis.R

skhim

2023-04-10

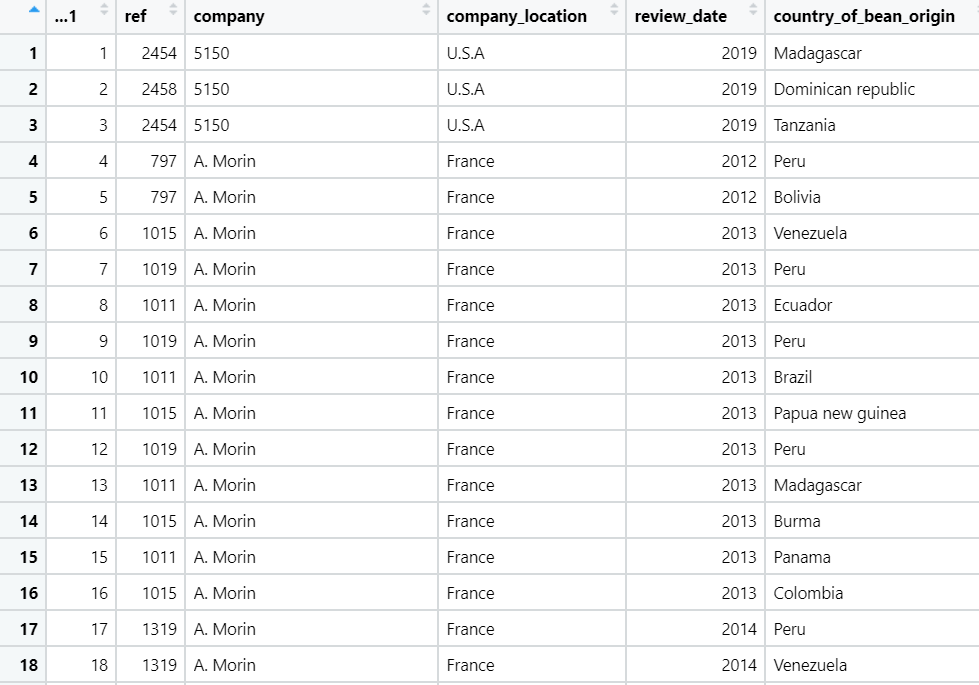
#Creating data frame  
library(readr)

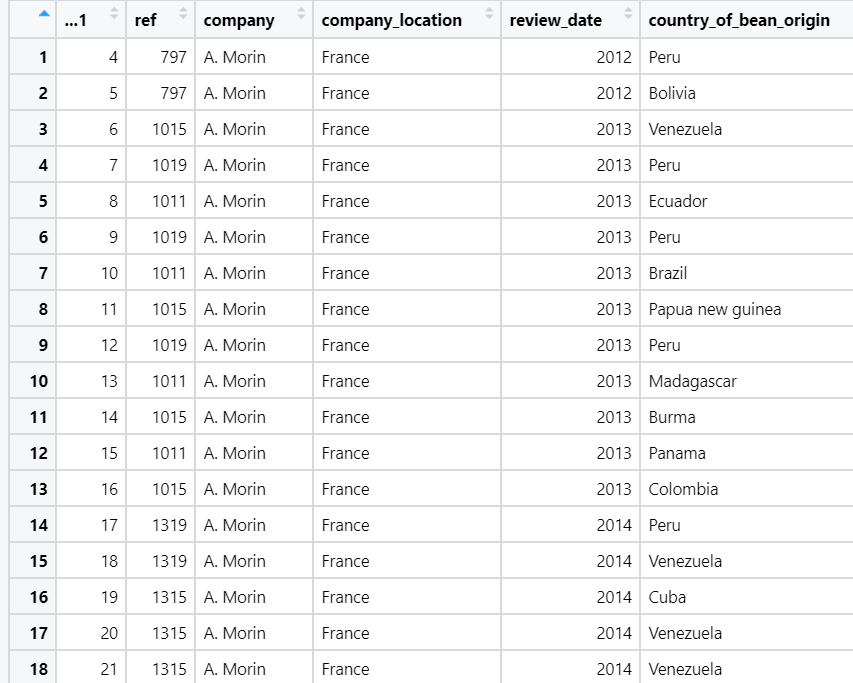
## Warning: package 'readr' was built under R version 4.2.3

chocolate <- read\_csv("data/chocolate.csv")

## New names:  
## Rows: 2224 Columns: 19  
## ── Column specification  
## ──────────────────────────────────────────────────────── Delimiter: "," chr  
## (6): company, company\_location, country\_of\_bean\_origin, first\_taste, se... dbl  
## (13): ...1, ref, review\_date, cocoa\_percent, rating, counts\_of\_ingredien...  
## ℹ Use `spec()` to retrieve the full column specification for this data. ℹ  
## Specify the column types or set `show\_col\_types = FALSE` to quiet this message.  
## • `` -> `...1`

View(chocolate)



chocolate <- data.frame(chocolate)  
  
#Questions  
#Selecting top 10 companies  
freq\_Origin = as.data.frame(table(chocolate$company))  
colnames(freq\_Origin) = c("Company", "Frequency")  
  
# The top 10 chocolate bar producers  
top10s = dplyr::arrange(freq\_Origin, desc(Frequency))[1:10,]  
onlytops = dplyr::filter(chocolate, company %in% top10s[,1])  
  
View(onlytops)  


#1. Find most common first taste and compare rating with other tastes  
  
# find most common first taste  
common\_taste = names(which.max(table(chocolate$first\_taste)))  
  
# find the avg ratings  
average\_rating = 0  
average\_count = nrow(chocolate["rating"])  
common\_taste\_rating = 0  
common\_taste\_rating\_count = 0  
  
taste = chocolate["first\_taste"]  
rating = chocolate["rating"]  
  
for (index in 1:nrow(chocolate["rating"])) {  
 if (taste[[1]][[index]] == common\_taste) {  
 common\_taste\_rating = common\_taste\_rating + rating[[1]][[index]]  
 common\_taste\_rating\_count = common\_taste\_rating\_count + 1  
 }  
 average\_rating = average\_rating + rating[[1]][[index]]  
}  
  
message("Average rating of chocolates is: ", average\_rating / average\_count)

## Average rating of chocolates is: 3.19856115107914

message("Rating for chocolates with most popular taste is: ", common\_taste\_rating / common\_taste\_rating\_count)

## Rating for chocolates with most popular taste is: 3.484375

#2. Multiple regression (model rating in terms of cocoa percent and count of ingredients)  
library(dplyr)

## Warning: package 'dplyr' was built under R version 4.2.3

##   
## 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

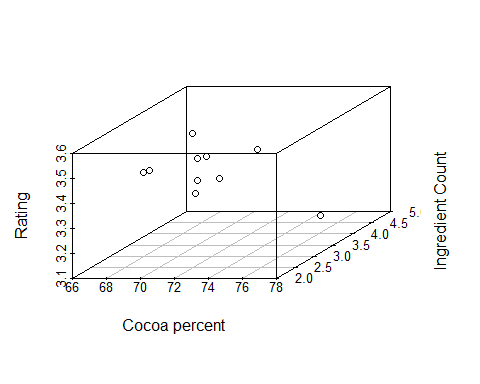
Rating\_data <- select(onlytops, c(company, rating, cocoa\_percent, counts\_of\_ingredients))  
Rating\_data = Rating\_data %>% group\_by(company) %>%  
 summarise(avg\_rating = mean(rating),  
 avg\_cocoa\_percent = mean(cocoa\_percent),  
 avg\_ingredient\_count = round(mean(counts\_of\_ingredients)),  
 .groups = 'drop')  
  
RegModel = lm(Rating\_data$avg\_rating ~ Rating\_data$avg\_cocoa\_percent + Rating\_data$avg\_ingredient\_count)  
RegModel

##   
## Call:  
## lm(formula = Rating\_data$avg\_rating ~ Rating\_data$avg\_cocoa\_percent +   
## Rating\_data$avg\_ingredient\_count)  
##   
## Coefficients:  
## (Intercept) Rating\_data$avg\_cocoa\_percent   
## 4.61467 -0.01119   
## Rating\_data$avg\_ingredient\_count   
## -0.12244

summary(RegModel)

##   
## Call:  
## lm(formula = Rating\_data$avg\_rating ~ Rating\_data$avg\_cocoa\_percent +   
## Rating\_data$avg\_ingredient\_count)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.09964 -0.05898 -0.03310 0.06209 0.13657   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 4.61467 0.87982 5.245 0.00119 \*\*  
## Rating\_data$avg\_cocoa\_percent -0.01119 0.01175 -0.953 0.37255   
## Rating\_data$avg\_ingredient\_count -0.12244 0.03222 -3.800 0.00671 \*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.09634 on 7 degrees of freedom  
## Multiple R-squared: 0.6781, Adjusted R-squared: 0.5862   
## F-statistic: 7.374 on 2 and 7 DF, p-value: 0.01892

library(scatterplot3d)  
scatterplot3d(Rating\_data$avg\_rating ~ Rating\_data$avg\_cocoa\_percent + Rating\_data$avg\_ingredient\_count,  
 xlab = "Cocoa percent",  
 ylab = "Ingredient Count",  
 zlab = "Rating")



#3. Hypothesis testing (take top 10 companies producing chocolates as a sample and see if it's representative of the entire population) mean of the rating  
sample\_mean=0  
sample\_size=0  
temp = data.frame(sort(table(chocolate$company), decreasing=TRUE)[1:10])  
temp

## Var1 Freq  
## 1 Soma 52  
## 2 Arete 32  
## 3 Fresco 31  
## 4 Bonnat 28  
## 5 Pralus 26  
## 6 A. Morin 25  
## 7 Domori 22  
## 8 Guittard 22  
## 9 Valrhona 22  
## 10 Zotter 21

soma = data.frame(chocolate[chocolate$company=='Soma' , ])  
tempmean=mean(soma$rating)  
n=nrow(soma)  
sample\_size=sample\_size+n  
sample\_mean=sample\_mean+tempmean  
  
arete = data.frame(chocolate[chocolate$company=='Arete' , ])  
tempmean=mean(arete$rating)  
n=nrow(arete)  
sample\_size=sample\_size+n  
sample\_mean=sample\_mean+tempmean  
  
fresco = data.frame(chocolate[chocolate$company=='Fresco' , ])  
tempmean=mean(fresco$rating)  
n=nrow(fresco)  
sample\_size=sample\_size+n  
sample\_mean=sample\_mean+tempmean  
  
bonnat = data.frame(chocolate[chocolate$company=='Bonnat' , ])  
tempmean=mean(bonnat$rating)  
n=nrow(bonnat)  
sample\_size=sample\_size+n  
sample\_mean=sample\_mean+tempmean  
  
pralus = data.frame(chocolate[chocolate$company=='Pralus' , ])  
tempmean=mean(pralus$rating)  
n=nrow(pralus)  
sample\_size=sample\_size+n  
sample\_mean=sample\_mean+tempmean  
  
amorin = data.frame(chocolate[chocolate$company=='A. Morin' , ])  
tempmean=mean(amorin$rating)  
n=nrow(amorin)  
sample\_size=sample\_size+n  
sample\_mean=sample\_mean+tempmean  
  
domori = data.frame(chocolate[chocolate$company=='Domori' , ])  
tempmean=mean(domori$rating)  
n=nrow(domori)  
sample\_size=sample\_size+n  
sample\_mean=sample\_mean+tempmean  
  
guittard = data.frame(chocolate[chocolate$company=='Guittard' , ])  
tempmean=mean(guittard$rating)  
n=nrow(guittard)  
sample\_size=sample\_size+n  
sample\_mean=sample\_mean+tempmean  
  
valrhona = data.frame(chocolate[chocolate$company=='Valrhona' , ])  
tempmean=mean(valrhona$rating)  
n=nrow(valrhona)  
sample\_size=sample\_size+n  
sample\_mean=sample\_mean+tempmean  
  
zotter = data.frame(chocolate[chocolate$company=='Zotter' , ])  
tempmean=mean(zotter$rating)  
n=nrow(zotter)  
sample\_size=sample\_size+n  
sample\_mean=sample\_mean+tempmean  
  
sample\_mean=sample\_mean/10  
population\_mean=mean(chocolate$rating)  
sigma=sd(chocolate$rating)  
  
z=(sample\_mean-population\_mean)/(sigma/sqrt(sample\_size))  
  
alpha=0.05  
zhalfalpha=qnorm(1-(alpha/2))  
c(-zhalfalpha,zhalfalpha)

## [1] -1.959964 1.959964

pval=2\*pnorm(z)  
  
if(pval>alpha){print("Accept Null hypothesis")} else{print("Reject Null hypothesis")}

## [1] "Accept Null hypothesis"

#4. Hypothesis testing (take 2 countries, compare the average reviews) - check of one country produces objectively better chocolates   
ratingmeans=aggregate(onlytops$rating, list(onlytops$company), FUN=mean)  
ratingmeans

## Group.1 x  
## 1 A. Morin 3.400000  
## 2 Arete 3.531250  
## 3 Bonnat 3.526786  
## 4 Domori 3.522727  
## 5 Fresco 3.403226  
## 6 Guittard 3.170455  
## 7 Pralus 3.173077  
## 8 Soma 3.591346  
## 9 Valrhona 3.318182  
## 10 Zotter 3.321429

P1=(ratingmeans$Group.1[1])  
m1=ratingmeans$x[1]  
P2=(ratingmeans$Group.1[2])  
m2=ratingmeans$x[2]  
n1=sum(onlytops$company==P1)  
n2=sum(onlytops$company==P2)  
n2

## [1] 32

ratingsd=aggregate(onlytops$rating, list(onlytops$company), FUN=sd)  
sd1=ratingsd$x[1]  
sd2=ratingsd$x[2]  
  
#t test for 2 means  
#H0 : Both companies have the same quality/popularity as the other  
#H1 : The companies are not on the same level of quality/popularity  
# testing at 5% level of significance  
t= (m1-m2)/sqrt(((sd1\*sd1)/n1)+((sd2\*sd2)/n2))  
t

## [1] -1.318355

cv=qt(0.975,(n1+n2-2))  
cv

## [1] 2.004045

if(cv <=t){print("Accept Ho")} else{print("Reject Ho")}

## [1] "Reject Ho"