TFM\_E2\_Grupo02

2023-06-08

library(dplyr)

##   
## 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

library(factoextra)

## Loading required package: ggplot2

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

library(ggplot2)  
library(cluster)  
library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ forcats 1.0.0 ✔ stringr 1.5.0  
## ✔ lubridate 1.9.2 ✔ tibble 3.2.1  
## ✔ purrr 1.0.1 ✔ tidyr 1.3.0  
## ✔ readr 2.1.4

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(ggcorrplot)  
library(writexl)

## Warning: package 'writexl' was built under R version 4.3.1

library(openxlsx)

## Warning: package 'openxlsx' was built under R version 4.3.1

library(PerformanceAnalytics)

## Warning: package 'PerformanceAnalytics' was built under R version 4.3.1

## Loading required package: xts  
## Loading required package: zoo  
##   
## Attaching package: 'zoo'  
##   
## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric  
##   
##   
## ######################### Warning from 'xts' package ##########################  
## # #  
## # The dplyr lag() function breaks how base R's lag() function is supposed to #  
## # work, which breaks lag(my\_xts). Calls to lag(my\_xts) that you type or #  
## # source() into this session won't work correctly. #  
## # #  
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #  
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop #  
## # dplyr from breaking base R's lag() function. #  
## # #  
## # Code in packages is not affected. It's protected by R's namespace mechanism #  
## # Set `options(xts.warn\_dplyr\_breaks\_lag = FALSE)` to suppress this warning. #  
## # #  
## ###############################################################################  
##   
## Attaching package: 'xts'  
##   
## The following objects are masked from 'package:dplyr':  
##   
## first, last  
##   
##   
## Attaching package: 'PerformanceAnalytics'  
##   
## The following object is masked from 'package:graphics':  
##   
## legend

library(corrplot)

## corrplot 0.92 loaded

library(ggplot2)  
library(car)

## Loading required package: carData  
##   
## Attaching package: 'car'  
##   
## The following object is masked from 'package:purrr':  
##   
## some  
##   
## The following object is masked from 'package:dplyr':  
##   
## recode

library(simstudy)

## Warning: package 'simstudy' was built under R version 4.3.1

library(data.table)

## Warning: package 'data.table' was built under R version 4.3.1

##   
## Attaching package: 'data.table'  
##   
## The following objects are masked from 'package:xts':  
##   
## first, last  
##   
## The following objects are masked from 'package:lubridate':  
##   
## hour, isoweek, mday, minute, month, quarter, second, wday, week,  
## yday, year  
##   
## The following object is masked from 'package:purrr':  
##   
## transpose  
##   
## The following objects are masked from 'package:dplyr':  
##   
## between, first, last

library(corrr)

## Warning: package 'corrr' was built under R version 4.3.1

#install.packages("dbscan")  
library(dbscan)

## Warning: package 'dbscan' was built under R version 4.3.1

##   
## Attaching package: 'dbscan'  
##   
## The following object is masked from 'package:stats':  
##   
## as.dendrogram

library(DMwR2)

## Warning: package 'DMwR2' was built under R version 4.3.1

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo   
##   
## Attaching package: 'DMwR2'  
##   
## The following object is masked from 'package:dbscan':  
##   
## kNN

library(FactoMineR)  
library(factoextra)  
library(ggthemes)

## Warning: package 'ggthemes' was built under R version 4.3.1

#cargamos los datos con las variables que nos interesan  
#df1 = data.frame(read.csv2('C:/Users/linfante/OneDrive/Documentos/MasterBigData/TFM/tfm\_master\_ds/1 - BackEnd/Datos/alimentos v3.csv', sep=','))  
#df2 = data.frame(read.csv2('C:/Users/linfante/OneDrive/Documentos/MasterBigData/TFM/tfm\_master\_ds/1 - BackEnd/Datos/alimentos\_vegan.csv', sep=','))  
df = data.frame(read.csv2('C:/Users/linfante/OneDrive/Documentos/MasterBigData/TFM/tfm\_master\_ds/1 - BackEnd/Datos/alimentos v7.csv', sep=','))

# Selecciona columnas a investigar y Elimina registros duplicados

df <- df[,c("ISO3","PRODUCT\_NAME","ENERGY\_100G","FAT\_100G","CARBOHYDRATES\_100G","PROTEINS\_100G")]  
#df <- df[,c("PRODUCT\_NAME","PROTEINS\_100G","CARBOHYDRATES\_100G","FAT\_100G")]  
# Elimina registros duplicados  
df <- subset(df, !duplicated(df))

PRODUCT\_NAME <- df[,"PRODUCT\_NAME"]  
ISO3 <- df[,"ISO3"]  
 #Convertir a numerico todo  
df\_nuevo <- df %>%  
 mutate(across(where(is.character), type.convert, as.is = TRUE)) %>%  
 select\_if(is.numeric)

## Warning: There was 1 warning in `mutate()`.  
## ℹ In argument: `across(where(is.character), type.convert, as.is = TRUE)`.  
## Caused by warning:  
## ! The `...` argument of `across()` is deprecated as of dplyr 1.1.0.  
## Supply arguments directly to `.fns` through an anonymous function instead.  
##   
## # Previously  
## across(a:b, mean, na.rm = TRUE)  
##   
## # Now  
## across(a:b, \(x) mean(x, na.rm = TRUE))

PRODUCT\_NAME <- df[,"PRODUCT\_NAME"]  
df\_nuevo <- cbind(ISO3,PRODUCT\_NAME,df\_nuevo)

# Eliminar vectores fila nulos

# Sumar las columnas del dataframe  
suma\_columnas <- rowSums(df\_nuevo[,3:ncol(df\_nuevo)])  
  
# Agregar la variable de suma al dataframe  
df\_nuevo <- cbind(df\_nuevo, suma = suma\_columnas)  
  
df\_nuevo <- subset(df\_nuevo, suma != 0)  
df\_nuevo <- df\_nuevo[, -which(names(df\_nuevo) == "suma")]  
  
cantidad\_ceros\_por\_columna <- colSums(df\_nuevo == 0)  
  
print(cantidad\_ceros\_por\_columna)

## ISO3 PRODUCT\_NAME ENERGY\_100G FAT\_100G   
## 0 0 38 314   
## CARBOHYDRATES\_100G PROTEINS\_100G   
## 222 166

# ver <- df\_nuevo[,"PRODUCT\_NAME"]  
# ver2<-as.data.frame(table(ver))  
# duplicados <- duplicated(ver)  
#   
# # Filtrar el dataframe original para mostrar solo las filas duplicadas  
# dataframe\_duplicados <- df\_nuevo[duplicados, ]  
# dataframe\_duplicados <- dataframe\_duplicados[,"PRODUCT\_NAME"]  
#   
# VER3 <- as.data.frame(subset(dataframe\_duplicados, !duplicated(dataframe\_duplicados)))

# Aplicar filtros negativos

# Crear un vector con las palabras a buscar  
palabras\_a\_eliminar <- c("SALSA DE SOJA","SAUCE SOJA", "SAUCE SOJA","BEBIDA DE SOJA", "YAOURT SOJA","YOGURT DE SOJA","CHOCOLAT","ARROZ Y SOJA","LECHE","MOUSSE","MILK","ACEITE","DESSERT","PAN SOJA","BOISSON SOJA","GLACE","ML","SAUCE DE SOJA","SAUCE","SALSA","BIBEDA DE SOJA","LAIT SOJA","VIVESOY SOJA","LAIT DE SOJA","YAOURT","POSTRE","MUESLI","YOGUR","BEBIDA","MARGARINA","VINAIGRETTE","DRINK","SAUCE","BATIDO","LAIT","MAYONNAISE","CAFE","VANILLE","NATA","YOGURT","LACTOVISOY","ALIMENTO DE SOYA","ALIMENTO LIQUIDO DE SOYA","BEVANDA","SWEET SOY","VANILLE","VANILLA","UNSWEETENED","SWEETENED","STRAWBERRY","PROTEIN ISOLATE","SOYA CALCIUM","BLUEBERRY","BEVERAGE","BARISTA","SOYA A TARTINER","JUGO","PROTEIN POWDER","PROTEIN ISOLATE","CACAO","CALCIUM","CALCIO","NUTRI SOJA","SOIA BIANCO CREMOSO","LECITINA","LECITHINE","LATTE","HARICOTS","GERME","GELATO","GATEAU","FROMAGE FRAIS","INFUSION","SOYBEAN PASTE","MIXBEANS","MISO SOUP","SOYBEANS","PATE DE SOJA","PLAIN SOYA","PETIT’SOIF","SOYBEANS","FRAMBOISE","MANGO","DOUCEUR","SOJABOHNENKEIMLINGE","KEIMLINGE","MANGUE","PECHE","MANZANA","VICHYSSOISE","CHOCOAVENA","AVENA")  
  
# Crear una función que verifica si alguna de las palabras está presente en el texto  
verificar\_palabras <- function(texto, palabras) {  
 sapply(palabras, function(palabra) grepl(palabra, texto))  
}  
# Identificar las filas que contienen alguna de las palabras a eliminar  
filas\_a\_eliminar <- apply(df\_nuevo, 1, function(row) any(verificar\_palabras(row["PRODUCT\_NAME"], palabras\_a\_eliminar)))  
  
# Eliminar las filas identificadas del dataframe original  
df\_nuevo1 <- df\_nuevo[!filas\_a\_eliminar, ]  
#df\_nuevo2 <- df\_nuevo1[,-1]  
  
#Elimino los faltantes  
df\_nuevo1 <- df\_nuevo1[complete.cases(df\_nuevo1), ]

df\_soja <- subset(df\_nuevo1, grepl("soja", PRODUCT\_NAME, ignore.case = TRUE)) ## Registros que contienen la palabra "soja"  
df\_soja <- na.omit(df\_soja)  
  
df\_seitan <- subset(df\_nuevo1, grepl("seitan", PRODUCT\_NAME, ignore.case = TRUE)) ## Registros que contienen la palabra "seitan"  
df\_seitan <- na.omit(df\_seitan)  
  
df\_tofu <- subset(df\_nuevo1, grepl("tofu", PRODUCT\_NAME, ignore.case = TRUE)) ## Registros que contienen la palabra "tofu"  
df\_tofu <- na.omit(df\_tofu)

# Frecuencia de productos por país de orígen

print(table(df\_seitan$ISO3))

##   
## AND AUT BEL CAN CHE CZE DEU DZA ESP FIN FRA GBR HRV IRL ISR ITA JPN LUX MLT NLD   
## 1 2 39 10 26 4 57 1 144 2 144 7 2 2 2 122 1 2 1 5   
## PRT ROU SWE USA   
## 3 1 2 44

print(table(df\_tofu$ISO3))

##   
## AND ARG AUS AUT BEL BGR BRA CAN CHE CHL CUB CZE DEU DNK ESP FIN FRA GBR HKG   
## 1 2 2 62 28 65 2 2 156 148 1 1 43 224 2 326 13 632 76 2   
## HRV HUN IND IRL ISR ITA JPN KEN LUX MEX MLT MUS NCL NLD NZL POL PRT PYF REU ROU   
## 3 4 2 42 2 179 10 1 2 5 1 1 2 13 2 14 10 6 4 11   
## SGP SVK SVN SWE THA UKR URY USA VIR   
## 7 6 1 13 1 1 1 390 1

print(table(df\_soja$ISO3))

##   
## AND ARG AUS AUT BEL BOL BRA BRN CAN CHE CHL CIV CZE DEU DNK   
## 3 2 16 3 27 84 1 26 1 5 55 1 1 2 213 3   
## DZA ESP FIN FRA GBR HRV HUN IRL ITA JPN KHM LUX MDG NLD NOR NZL   
## 1 335 4 1119 8 1 1 32 28 5 1 9 1 14 1 1   
## POL PRT PYF REU ROU SAU SEN SLV SWE THA URY USA   
## 4 23 1 1 1 1 1 1 6 1 4 18

# SEITAN

# Aplicar algoritmo Local Outliers Factor para eliminar atípicos  
  
# 1.- First, we need to convert the incorrect type of variables.  
  
seitan\_clean <- df\_seitan %>%  
 mutate(PRODUCT\_NAME = as.factor(PRODUCT\_NAME))  
str(seitan\_clean)

## 'data.frame': 624 obs. of 6 variables:  
## $ ISO3 : chr "USA" "USA" "USA" "USA" ...  
## $ PRODUCT\_NAME : Factor w/ 451 levels "2 CRISP CRUMBED SEITAN SLICES",..: 407 407 407 67 67 375 197 197 156 415 ...  
## $ ENERGY\_100G : num 587 519 640 628 738 732 551 551 661 398 ...  
## $ FAT\_100G : num 1.75 1.77 3.53 2.21 2.94 ...  
## $ CARBOHYDRATES\_100G: num 6.14 4.42 7.06 7.96 10.59 ...  
## $ PROTEINS\_100G : num 24.6 24.8 25.9 26.6 27.1 ...

seitan\_clean <- as.data.frame(scale(df\_seitan[,-1:-2]))  
seitan\_lof <- lof(seitan\_clean, minPts = 7)  
head(seitan\_clean)

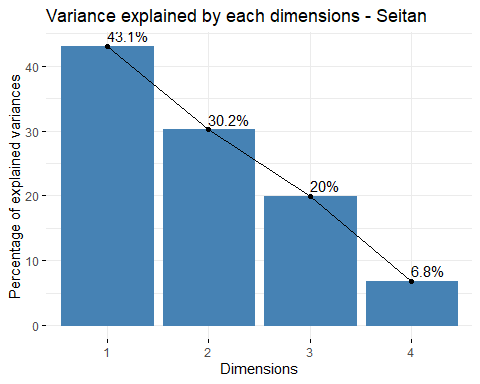
## ENERGY\_100G FAT\_100G CARBOHYDRATES\_100G PROTEINS\_100G  
## 78 -0.5521054 -0.7131746 -0.50641464 0.2729566  
## 83 -0.7550831 -0.7106072 -0.73501417 0.2922927  
## 84 -0.3939022 -0.4210450 -0.38438150 0.3898047  
## 85 -0.4297218 -0.6381919 -0.26463383 0.4488568  
## 86 -0.1013754 -0.5178510 0.08459036 0.4938623  
## 87 -0.1192852 -0.2794073 0.19245894 0.2728328

#PCA and use the first two dimensions of the PCA  
library(FactoMineR)  
library(factoextra)  
  
seitan\_pca <- PCA(seitan\_clean, scale.unit = F, ncp = 6, graph = F)

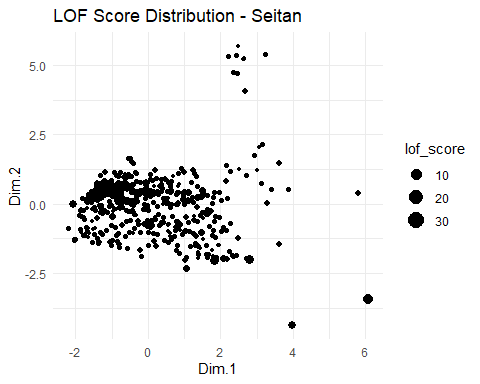
summary(seitan\_pca)

##   
## Call:  
## PCA(X = seitan\_clean, scale.unit = F, ncp = 6, graph = F)   
##   
##   
## Eigenvalues  
## Dim.1 Dim.2 Dim.3 Dim.4  
## Variance 1.719 1.205 0.797 0.272  
## % of var. 43.050 30.175 19.962 6.813  
## Cumulative % of var. 43.050 73.225 93.187 100.000  
##   
## Individuals (the 10 first)  
## Dist Dim.1 ctr cos2 Dim.2 ctr cos2   
## 78 | 1.070 | -0.922 0.079 0.743 | 0.516 0.035 0.233 |  
## 83 | 1.304 | -1.161 0.126 0.793 | 0.578 0.044 0.197 |  
## 84 | 0.795 | -0.580 0.031 0.533 | 0.515 0.035 0.420 |  
## 85 | 0.929 | -0.649 0.039 0.488 | 0.571 0.043 0.377 |  
## 86 | 0.728 | -0.190 0.003 0.069 | 0.506 0.034 0.484 |  
## 87 | 0.451 | -0.078 0.001 0.030 | 0.204 0.006 0.205 |  
## 392 | 1.053 | -0.976 0.089 0.859 | 0.315 0.013 0.089 |  
## 393 | 1.053 | -0.976 0.089 0.859 | 0.315 0.013 0.089 |  
## 394 | 1.054 | -0.649 0.039 0.379 | 0.778 0.081 0.546 |  
## 395 | 1.653 | -1.619 0.244 0.960 | 0.256 0.009 0.024 |  
## Dim.3 ctr cos2   
## 78 0.154 0.005 0.021 |  
## 83 -0.012 0.000 0.000 |  
## 84 0.050 0.001 0.004 |  
## 85 0.292 0.017 0.099 |  
## 86 0.466 0.044 0.411 |  
## 87 0.354 0.025 0.617 |  
## 392 0.220 0.010 0.044 |  
## 393 0.220 0.010 0.044 |  
## 394 0.271 0.015 0.066 |  
## 395 0.182 0.007 0.012 |  
##   
## Variables  
## Dim.1 ctr cos2 Dim.2 ctr cos2 Dim.3  
## ENERGY\_100G | 0.897 46.846 0.807 | 0.249 5.157 0.062 | 0.020  
## FAT\_100G | 0.680 26.915 0.463 | -0.320 8.480 0.102 | -0.625  
## CARBOHYDRATES\_100G | 0.607 21.420 0.369 | -0.446 16.476 0.199 | 0.631  
## PROTEINS\_100G | 0.288 4.819 0.083 | 0.918 69.887 0.844 | 0.083  
## ctr cos2   
## ENERGY\_100G 0.053 0.000 |  
## FAT\_100G 49.071 0.392 |  
## CARBOHYDRATES\_100G 50.009 0.399 |  
## PROTEINS\_100G 0.867 0.007 |

fviz\_eig(seitan\_pca, ncp = 6, addlabels = T, main = "Variance explained by each dimensions - Seitan")



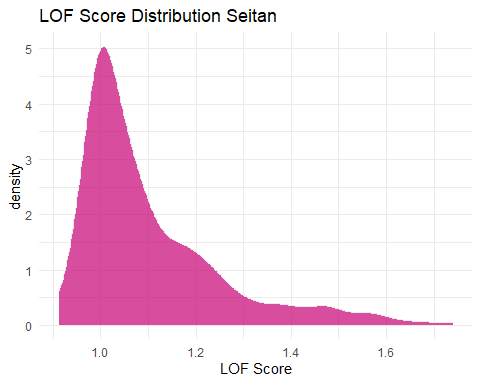
library(ggthemes)  
seitan\_a <- data.frame(seitan\_pca$ind$coord[,1:3])  
seitan\_b <- cbind(seitan\_a, lof\_score = seitan\_lof)  
#seitan\_b <- cbind(seitan\_a, fraud = seitan\_clean$, lof\_score = seitan\_clean$lof)  
  
seitan\_lof\_visual <- ggplot(seitan\_b, aes(x=Dim.1 ,y=Dim.2)) +   
 geom\_point(aes(size=lof\_score)) +  
 ggtitle("LOF Score Distribution - Seitan")+  
 theme\_minimal()  
  
seitan\_lof\_visual



summary(seitan\_b)

## Dim.1 Dim.2 Dim.3 lof\_score   
## Min. :-2.2051 Min. :-4.37419 Min. :-8.2367 Min. :0.9126   
## 1st Qu.:-0.9834 1st Qu.:-0.68423 1st Qu.:-0.3688 1st Qu.:1.0014   
## Median :-0.3252 Median : 0.06254 Median :-0.0602 Median :1.0707   
## Mean : 0.0000 Mean : 0.00000 Mean : 0.0000 Mean : Inf   
## 3rd Qu.: 0.8360 3rd Qu.: 0.55616 3rd Qu.: 0.3935 3rd Qu.:1.2404   
## Max. : 6.0740 Max. : 5.68950 Max. : 5.4533 Max. : Inf

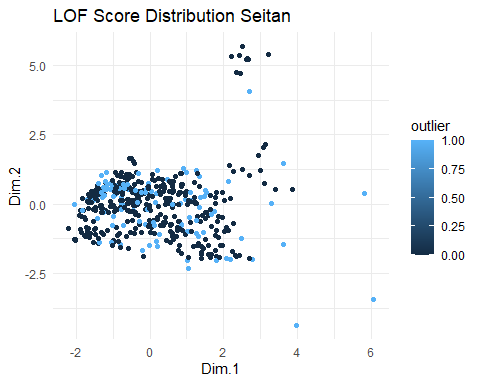
seitan\_b %>%  
 filter(lof\_score <= 1.75) %>%   
 ggplot( aes(x=lof\_score)) +  
 geom\_density( color="#e9ecef", fill = "#c90076", alpha=0.7) +  
 scale\_fill\_manual(values="#8fce00") +  
 xlab("LOF Score")+  
 ggtitle("LOF Score Distribution Seitan")+  
 theme\_minimal() +  
 labs(fill="")



quantile(seitan\_b$lof\_score, probs = c(0, 0.8))

## 0% 80%   
## 0.9125573 1.3319969

seitan\_b <- seitan\_b %>%   
 mutate(outlier = ifelse(lof\_score > 1.3319969, 1, 0))  
  
seitan\_lof\_visual\_b <- ggplot(seitan\_b, aes(x=Dim.1 ,y=Dim.2, color=outlier)) +   
 geom\_point() +  
 ggtitle("LOF Score Distribution Seitan")+  
 theme\_minimal()  
  
seitan\_lof\_visual\_b



outliers <- seitan\_b[seitan\_b$outlier==1,]  
nooutliers <- seitan\_b[seitan\_b$outlier==0,]

df\_seitan <- df\_seitan[seitan\_lof < 1.3319969,]  
df\_seitan\_outliers <- df\_seitan[seitan\_lof >= 1.3319969,]

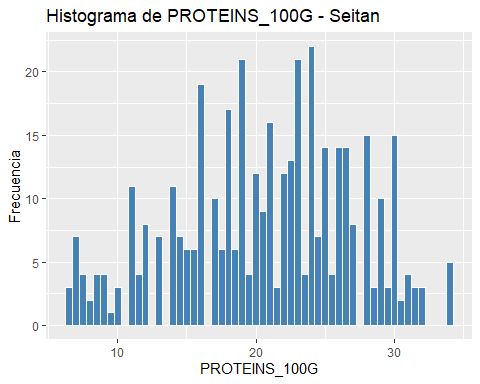
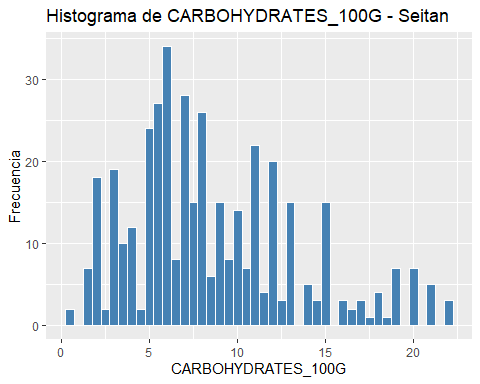
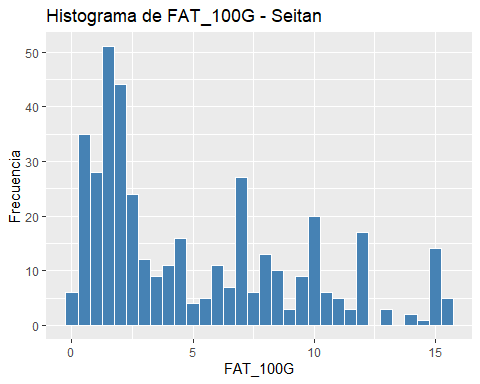
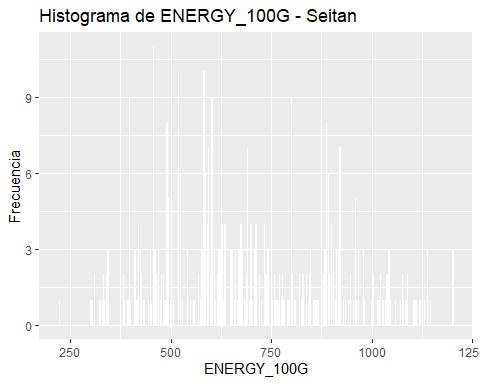
# ELIMINAR ATIPICOS MULTIVARIANTES CON DBSCAN

# ELIMINAR ATIPICOS MULTIVARIANTES  
# lof\_scores <- lof(df\_seitan[c("ENERGY\_100G","PROTEINS\_100G", "CARBOHYDRATES\_100G", "FAT\_100G")], minPts = 6)  
# df\_seitan <- df\_seitan[lof\_scores <= 1,]

# Eliminar valores atipicos univariantes

remove\_outliers <- function(data, column, sd\_threshold = 2) {  
 data[abs(scale(data[[column]])) < sd\_threshold, ]  
}  
  
columns\_to\_check <- 3:ncol(df\_seitan)  
for (column in columns\_to\_check) {  
 df\_seitan <- remove\_outliers(df\_seitan, column)  
}

# Obtener las columnas cuantitativas del dataframe  
columnas\_cuantitativas <- sapply(df\_seitan, is.numeric)  
  
# Crear un histograma para cada columna cuantitativa  
for (columna in names(df\_seitan[columnas\_cuantitativas])) {  
 plot\_data <- df\_seitan[, columna]  
 p <- ggplot(data.frame(x = plot\_data), aes(x)) +  
 geom\_histogram(binwidth = 0.5, fill = "steelblue", color = "white") +  
 labs(title = paste("Histograma de", columna,"- Seitan"),  
 x = columna,  
 y = "Frecuencia")  
   
 print(p)  
}

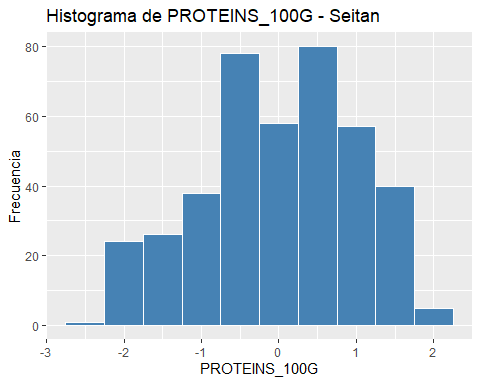
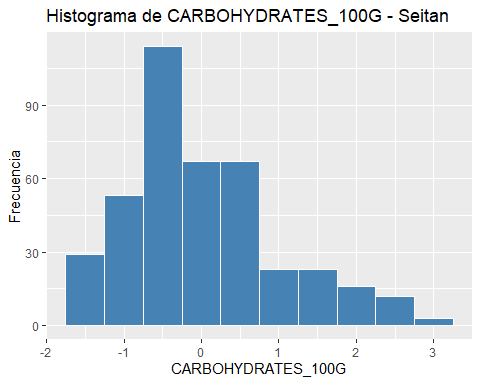
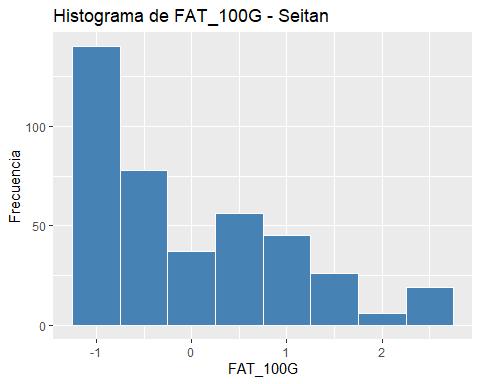
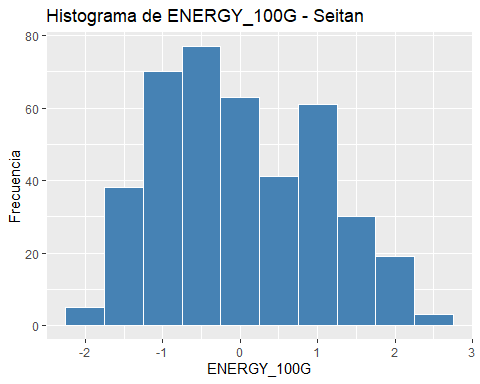


summary(as.data.frame(scale(df\_seitan[,-1:-2])))

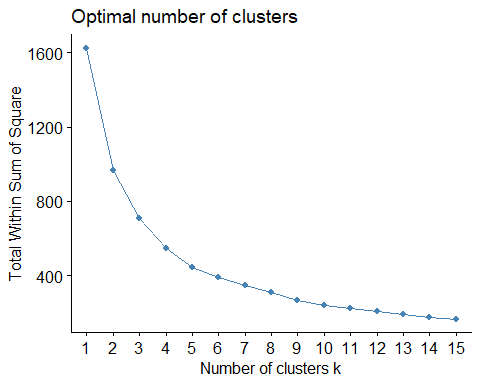
## ENERGY\_100G FAT\_100G CARBOHYDRATES\_100G PROTEINS\_100G   
## Min. :-2.1889 Min. :-1.1914 Min. :-1.7133 Min. :-2.2511   
## 1st Qu.:-0.8173 1st Qu.:-0.8407 1st Qu.:-0.6554 1st Qu.:-0.7415   
## Median :-0.1246 Median :-0.3731 Median :-0.2111 Median : 0.0448   
## Mean : 0.0000 Mean : 0.0000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.8775 3rd Qu.: 0.7257 3rd Qu.: 0.5284 3rd Qu.: 0.7446   
## Max. : 2.3322 Max. : 2.4557 Max. : 2.8357 Max. : 2.0891

#Datos normalizados para el Seitán

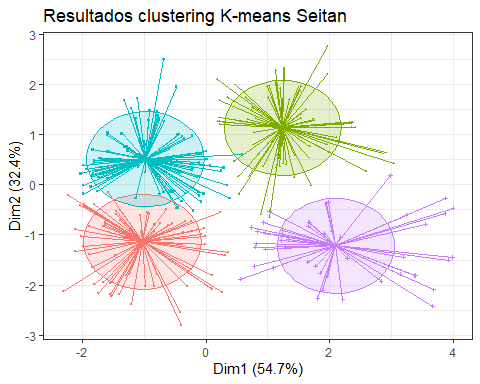
s\_seitan <- as.data.frame(scale(df\_seitan[,-1:-2]))  
# Obtener las columnas cuantitativas del dataframe  
columnas\_cuantitativas <- sapply(s\_seitan, is.numeric)  
  
# Crear un histograma para cada columna cuantitativa  
for (columna in names(s\_seitan[columnas\_cuantitativas])) {  
 plot\_data <- s\_seitan[, columna]  
 p <- ggplot(data.frame(x = plot\_data), aes(x)) +  
 geom\_histogram(binwidth = 0.5, fill = "steelblue", color = "white") +  
 labs(title = paste("Histograma de", columna,"- Seitan"),  
 x = columna,  
 y = "Frecuencia")  
   
 print(p)  
}



s\_seitan <- scale(df\_seitan[,-1:-2])  
  
# total de cluster óptimos  
elbow <- fviz\_nbclust(x = s\_seitan, FUNcluster = kmeans, method = "wss", k.max = 15,   
 diss = get\_dist(s\_seitan, method = "euclidean"), nstart = 25)  
print(elbow)

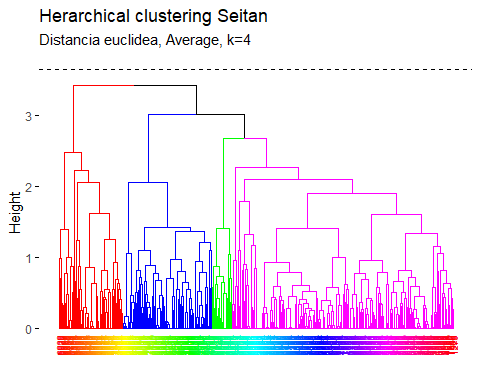


# set.seed(123)  
# km\_clusters <- kmeans(x=s\_seitan,centers=4,nstart=50)  
# fviz\_cluster(object=km\_clusters,data=s\_seitan,show.clust.cent = TRUE,  
# ellipse.type="euclid",star.plot=TRUE,repel=TRUE,  
# pointsize=0.5,outlier.color="darkred") +  
# labs(title ="Resultados clustering K-means") +  
# theme\_bw() +  
# theme(legend.position = "none")  
  
set.seed(123)  
km\_clusters <- kmeans(x = s\_seitan, centers = 4, nstart = 50)  
  
fviz\_cluster(object = km\_clusters, data = s\_seitan, show.clust.cent = TRUE,  
 ellipse.type = "euclid", star.plot = TRUE, repel = TRUE,  
 pointsize = 0.5, outlier.color = "darkred", geom = "point") +  
 theme\_bw() +  
 theme(legend.position = "none") +  
 ggtitle("Resultados clustering K-means Seitan")

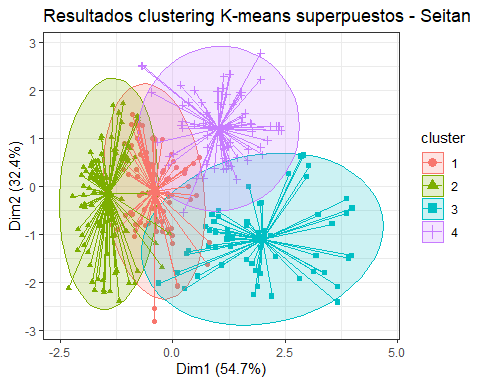


set.seed(101)  
  
hc\_euclidea\_av <- hclust(d = dist(x = s\_seitan, method = "euclidean"),  
 method = "average")  
fviz\_dend(x = hc\_euclidea\_av, k = 4, cex = 0.5,  
 k\_colors = c("red","blue","green","magenta"),color\_labels\_by\_k = T,  
 lwd = 0.2,type = "r",label\_cols = rainbow(nrow(df\_seitan)),  
 rect\_lty = "lightblue") +  
 geom\_hline(yintercept = 3.65, linetype = "dashed") +  
 labs(title = "Herarchical clustering Seitan",  
 subtitle = "Distancia euclidea, Average, k=4")

## Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as  
## of ggplot2 3.3.4.  
## ℹ The deprecated feature was likely used in the factoextra package.  
## Please report the issue at <https://github.com/kassambara/factoextra/issues>.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

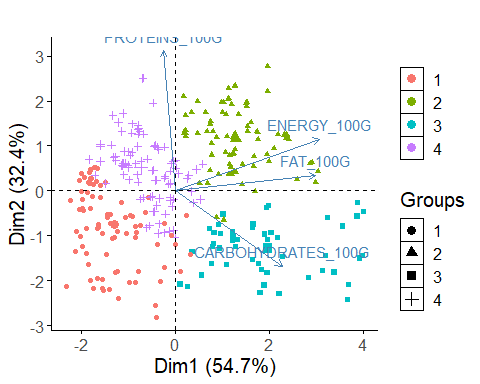


pam.res <- pam(s\_seitan, 4)  
# Visualización  
fviz\_cluster(pam.res, geom = "point", ellipse.type = "norm",  
 show.clust.cent = TRUE,star.plot = TRUE)+  
 labs(title = "Resultados clustering K-means superpuestos - Seitan")+ theme\_bw()



# Biplot PCA y K-Means para medir representatividad seitan

# PCA  
pca <- prcomp(df\_seitan[,-1:-2], scale=TRUE)  
df\_seitan.pca <- pca$x  
# Cluster over the three first PCA dimensions  
kc <- kmeans(df\_seitan.pca[,1:3], 4)  
fviz\_pca\_biplot(pca, label="var", habillage=as.factor(kc$cluster)) +  
 labs(color=NULL) + ggtitle("") +  
 theme(text = element\_text(size = 15),  
 panel.background = element\_blank(),   
 panel.grid.major = element\_blank(),  
 panel.grid.minor = element\_blank(),  
 axis.line = element\_line(colour = "black"),  
 legend.key = element\_rect(fill = "white"))



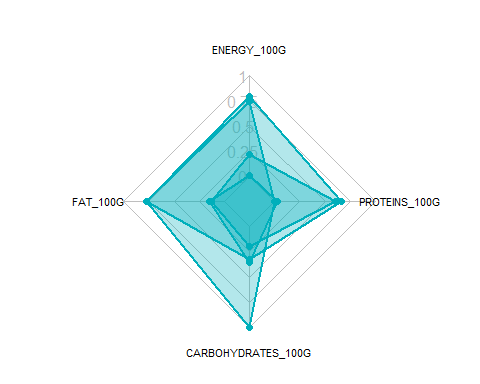
library(ggplot2)  
  
# Obtener los centroides de cada clúster  
centroids <- as.data.frame(km\_clusters$centers)  
  
  
# Obtener el valor mínimo y máximo en el dataframe  
min\_value <- min(centroids, na.rm = TRUE)  
max\_value <- max(centroids, na.rm = TRUE)  
  
# Escalar los datos entre 0 y 1  
scaled\_df <- (centroids - min\_value) / (max\_value - min\_value)  
  
library(fmsb)

## Warning: package 'fmsb' was built under R version 4.3.1

# Define the variable ranges: maximum and minimum  
max\_min <- data.frame(  
 ENERGY\_100G = c(1, 0), FAT\_100G = c(1, 0), CARBOHYDRATES\_100G = c(1, 0),PROTEINS\_100G = c(1, 0)  
)  
rownames(max\_min) <- c("Max", "Min")  
  
# Bind the variable ranges to the data  
df <- rbind(max\_min, scaled\_df)

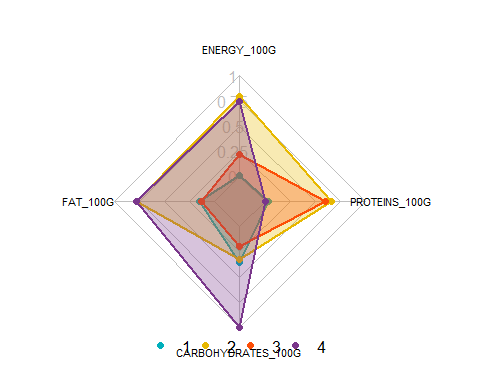
create\_beautiful\_radarchart <- function(data, color = "#00AFBB",   
 vlabels = colnames(data), vlcex = 0.7,  
 caxislabels = NULL, title = NULL, ...){  
 radarchart(  
 data, axistype = 1,  
 # Customize the polygon  
 pcol = color, pfcol = scales::alpha(color, 0.3), plwd = 2, plty = 1,  
 # Customize the grid  
 cglcol = "grey", cglty = 1, cglwd = 0.8,  
 # Customize the axis  
 axislabcol = "grey",   
 # Variable labels  
 vlcex = vlcex, vlabels = vlabels,  
 caxislabels = caxislabels, title = title, ...  
 )  
}

# Reduce plot margin using par()  
op <- par(mar = c(1, 2, 2, 1))  
create\_beautiful\_radarchart(df, caxislabels = c(0, .25, .5, .75, 1))



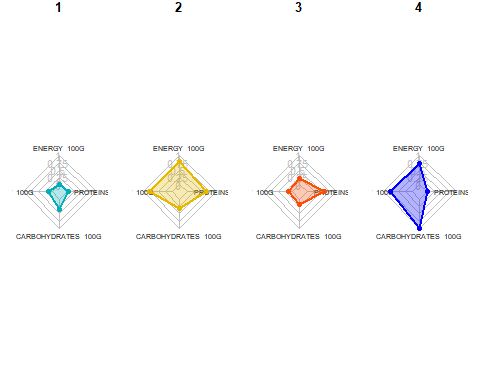
par(op)

# Reduce plot margin using par()  
op <- par(mar = c(1, 2, 2, 2))  
# Create the radar charts  
create\_beautiful\_radarchart(  
 data = df, caxislabels = c(0, .25, .5, .75, 1),  
 color = c("#00AFBB", "#E7B800", "#FC4E07","#7A378B")  
)  
# Add an horizontal legend  
legend(  
 x = "bottom", legend = rownames(df[-c(1,2),]), horiz = TRUE,  
 bty = "n", pch = 20 , col = c("#00AFBB", "#E7B800", "#FC4E07","#7A378B"),  
 text.col = "black", cex = 1, pt.cex = 1.5  
 )



par(op)

# Define colors and titles  
colors <- c("#00AFBB", "#E7B800", "#FC4E07","blue")  
titles <- c("1", "2", "3","4")  
  
# Reduce plot margin using par()  
# Split the screen in 3 parts  
op <- par(mar = c(1, 1, 1, 1))  
  
par(mfrow = c(1,4))  
  
# Create the radar chart  
for(i in 1:4){  
 create\_beautiful\_radarchart(  
 data = df[c(1, 2, i+2), ], caxislabels = c(0, .25, .5, .75, 1),  
 color = colors[i], title = titles[i]  
 )  
}



par(op)

# Visualizar los productos más cercanos al centroide que representan cada cluster top10

# Realizar clustering en el dataframe  
set.seed(123)  
df\_top <- df\_seitan  
km\_clusters <- kmeans(x = df\_top[, c("ENERGY\_100G", "FAT\_100G", "CARBOHYDRATES\_100G", "PROTEINS\_100G")], centers = 4, nstart = 50)  
  
# Obtener las asignaciones de clúster  
cluster\_assignments <- km\_clusters$cluster  
  
# Agregar las asignaciones de clúster al dataframe  
df\_top$cluster <- cluster\_assignments  
  
# Inicializar una lista para almacenar los productos más cercanos a cada centroide  
closest\_products <- vector("list", max(cluster\_assignments))  
  
# Encontrar los productos más cercanos a cada centroide  
for (cluster in 1:max(cluster\_assignments)) {  
 cluster\_center <- km\_clusters$centers[cluster, ]  
 distances <- apply(df\_top[, c("ENERGY\_100G", "FAT\_100G", "CARBOHYDRATES\_100G", "PROTEINS\_100G")], 1, function(row) {  
 sum((row - cluster\_center)^2)  
 })  
 closest\_products[[cluster]] <- head(order(distances), 10)  
}  
  
# Imprimir los productos más cercanos a cada centroide  
for (cluster in 1:max(cluster\_assignments)) {  
 cat("Cluster", cluster, ":\n")  
 print(df\_top[closest\_products[[cluster]], c("PRODUCT\_NAME", "ENERGY\_100G", "FAT\_100G", "CARBOHYDRATES\_100G", "PROTEINS\_100G")])  
 cat("\n")  
}

## Cluster 1 :  
## PRODUCT\_NAME ENERGY\_100G FAT\_100G  
## 7673 CROCCHETTA DI SEITAN 649 2.0  
## 2116 ARROSTO DI SEITAN 653 1.4  
## 5972 SEITAN VEGGIE-HACHE 653 1.5  
## 5977 SEITAN MET MARROKAANSE KRUIDEN 653 1.5  
## 8523 BIO BURGUER VEGETAL SEITAN AL CURRY 649 6.9  
## 8538 BIO MAXI BURGUER VEGETAL, SEITAN AL CURRY 649 6.9  
## 5708 SEITANGEHAKT 636 0.9  
## 1538 SEITAN 636 1.0  
## 4363 SEITAN, DURCHSCHNITT 636 1.0  
## 6031 VEGAN BURGER SEITAN WITH CHEESE FLAVOR 657 5.0  
## CARBOHYDRATES\_100G PROTEINS\_100G  
## 7673 14.0 19  
## 2116 12.0 23  
## 5972 13.0 21  
## 5977 13.0 21  
## 8523 9.8 11  
## 8538 9.8 11  
## 5708 7.8 28  
## 1538 5.6 30  
## 4363 5.6 30  
## 6031 13.0 14  
##   
## Cluster 2 :  
## PRODUCT\_NAME ENERGY\_100G FAT\_100G  
## 8547 SALCHIVEGGIE SEITAN Y TOFU 1071 12.0  
## 4784 BISTECCA DI SEITAN BIO ALLA MEDITERRANEA 1059 14.0  
## 7869 AFFETTATO VEGETALE DI SEITAN E TOFU 1075 14.7  
## 7864 AFFETTATO VEGETALE DI SEITAN E TOFU 1075 15.0  
## 1976 SEITAN TOFU 1079 14.8  
## 8535 BIO SEITAN Y TOFU TRADICIONAL 1084 15.0  
## 8385 SEITAN CON TOFU 1086 15.0  
## 8536 SEITAN Y TOFU TRADICIONAL 1088 15.0  
## 8795 SEITAN BIO 1088 15.0  
## 7651 CHARCUTERIE DE SEITAN 1042 12.0  
## CARBOHYDRATES\_100G PROTEINS\_100G  
## 8547 10.0 26.0  
## 4784 11.0 19.0  
## 7869 7.5 23.0  
## 7864 7.6 23.0  
## 1976 6.8 23.1  
## 8535 6.8 23.0  
## 8385 6.9 23.0  
## 8536 6.8 23.0  
## 8795 6.8 23.0  
## 7651 9.9 24.0  
##   
## Cluster 3 :  
## PRODUCT\_NAME ENERGY\_100G FAT\_100G CARBOHYDRATES\_100G  
## 8787 SEITAN NATURAL VEGANO 444 1.00 4.200000  
## 792 CUBED SEITAN WHEAT PROTEIN 443 0.00 4.705882  
## 7870 SEITAN INCREDIBILE 448 1.10 8.300000  
## 7515 SEITAN BIOLOGICO AL NATURALE 448 1.10 8.300000  
## 453 SEITAN 448 0.77 9.520000  
## 6033 SEITAN NATURAL 452 1.30 1.900000  
## 2239 SEITAN GOURMET 456 1.50 6.000000  
## 2235 SEITAN GOUTMET 456 1.50 6.000000  
## 2236 SEITAN GOURMET ORIGINAL 456 1.50 6.000000  
## 2241 SEITAN GOURMET 456 1.50 6.000000  
## PROTEINS\_100G  
## 8787 20.00000  
## 792 21.17647  
## 7870 16.00000  
## 7515 16.00000  
## 453 17.86000  
## 6033 21.50000  
## 2239 17.50000  
## 2235 18.00000  
## 2236 18.00000  
## 2241 18.00000  
##   
## Cluster 4 :  
## PRODUCT\_NAME ENERGY\_100G FAT\_100G CARBOHYDRATES\_100G  
## 2164 SEITAN ALLA PIASTRA 879 7.7000 11.7000  
## 1493 SEITAN 865 8.6667 10.0000  
## 4913 SEITAN AUFSCHNITT SPICY BEANS 864 6.8000 9.3000  
## 2183 COTOLETTA DI SEITAN DORATA 879 4.0000 17.0000  
## 4458 SEITAN 883 9.4000 8.6000  
## 4910 SEITAN GESCHNETZELTES 883 6.8000 8.0000  
## 4921 SEITAN HACK 883 6.8000 8.0000  
## 4456 MEDAILLONS DE SEITAN 866 6.6000 4.1000  
## 1494 SEITAN VEGE-POULET 886 8.2353 10.5882  
## 4908 SEITAN GERAUCHERT 862 7.1000 5.0000  
## PROTEINS\_100G  
## 2164 23.3000  
## 1493 22.0000  
## 4913 25.2000  
## 2183 18.0000  
## 4458 22.4000  
## 4910 28.0000  
## 4921 28.0000  
## 4456 31.4000  
## 1494 23.5294  
## 4908 29.3000

##### TOfu

# Aplicar algoritmo Local Outliers Factor para eliminar atípicos  
  
# 1.- First, we need to convert the incorrect type of variables.  
  
tofu\_clean <- df\_tofu %>%  
 mutate(PRODUCT\_NAME = as.factor(PRODUCT\_NAME))  
str(tofu\_clean)

## 'data.frame': 2523 obs. of 6 variables:  
## $ ISO3 : chr "FRA" "USA" "USA" "USA" ...  
## $ PRODUCT\_NAME : Factor w/ 1744 levels "1KG TOFU NATURE",..: 624 1413 1300 1283 1232 636 1433 879 36 38 ...  
## $ ENERGY\_100G : num 2686 540 644 372 1339 ...  
## $ FAT\_100G : num 62.7 7.09 9 3.8 28 ...  
## $ CARBOHYDRATES\_100G: num 12.2 0 2.1 2.53 6 ...  
## $ PROTEINS\_100G : num 6.3 13.73 16 8.86 18 ...

tofu\_clean <- as.data.frame(scale(df\_tofu[,-1:-2]))  
tofu\_lof <- lof(tofu\_clean, minPts = 7)  
head(tofu\_clean)

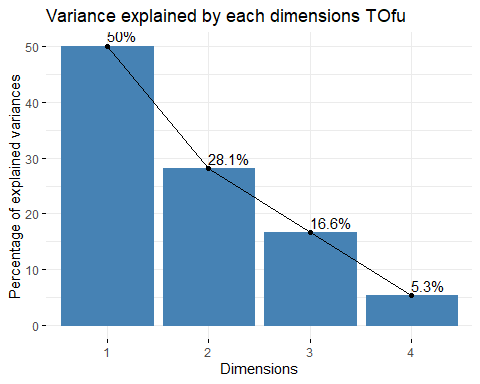
## ENERGY\_100G FAT\_100G CARBOHYDRATES\_100G PROTEINS\_100G  
## 1 4.9440915 7.97448441 0.58461773 -1.0218141  
## 4 -0.3460313 -0.31074089 -0.69333810 0.2750375  
## 5 -0.0896600 -0.02617374 -0.47336209 0.6712494  
## 6 -0.7601695 -0.80091159 -0.42831939 -0.5749850  
## 21 1.6235904 2.80459916 -0.06483523 1.0203346  
## 25 -0.9401224 -0.83497026 -0.44392721 -0.8746921

#PCA and use the first two dimensions of the PCA  
library(FactoMineR)  
library(factoextra)  
  
tofu\_pca <- PCA(tofu\_clean, scale.unit = F, ncp = 6, graph = F)

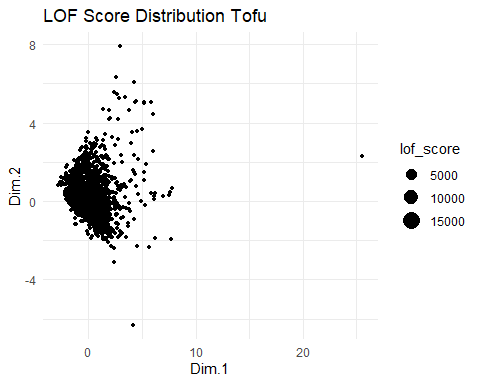
summary(tofu\_pca)

##   
## Call:  
## PCA(X = tofu\_clean, scale.unit = F, ncp = 6, graph = F)   
##   
##   
## Eigenvalues  
## Dim.1 Dim.2 Dim.3 Dim.4  
## Variance 1.999 1.123 0.664 0.212  
## % of var. 49.997 28.075 16.616 5.312  
## Cumulative % of var. 49.997 78.072 94.688 100.000  
##   
## Individuals (the 10 first)  
## Dist Dim.1 ctr cos2 Dim.2 ctr cos2   
## 1 | 9.456 | 7.777 1.199 0.676 | 0.650 0.015 0.005 |  
## 4 | 0.879 | -0.529 0.006 0.362 | -0.695 0.017 0.625 |  
## 5 | 0.827 | 0.015 0.000 0.000 | -0.793 0.022 0.920 |  
## 6 | 1.317 | -1.307 0.034 0.985 | 0.069 0.000 0.003 |  
## 21 | 3.398 | 3.059 0.186 0.810 | -0.872 0.027 0.066 |  
## 25 | 1.595 | -1.555 0.048 0.951 | 0.238 0.002 0.022 |  
## 31 | 2.581 | -0.261 0.001 0.010 | 2.492 0.219 0.932 |  
## 32 | 2.084 | -1.978 0.078 0.901 | 0.507 0.009 0.059 |  
## 33 | 0.911 | 0.478 0.005 0.275 | -0.429 0.006 0.221 |  
## 34 | 0.519 | -0.225 0.001 0.187 | -0.251 0.002 0.235 |  
## Dim.3 ctr cos2   
## 1 -5.304 1.678 0.315 |  
## 4 0.048 0.000 0.003 |  
## 5 0.232 0.003 0.079 |  
## 6 -0.073 0.000 0.003 |  
## 21 -1.091 0.071 0.103 |  
## 25 -0.248 0.004 0.024 |  
## 31 0.617 0.023 0.057 |  
## 32 -0.409 0.010 0.039 |  
## 33 0.642 0.025 0.497 |  
## 34 0.392 0.009 0.571 |  
##   
## Variables  
## Dim.1 ctr cos2 Dim.2 ctr cos2 Dim.3  
## ENERGY\_100G | 0.934 43.624 0.872 | 0.093 0.776 0.009 | -0.033  
## FAT\_100G | 0.831 34.579 0.692 | -0.117 1.224 0.014 | -0.482  
## CARBOHYDRATES\_100G | 0.436 9.511 0.190 | 0.798 56.680 0.637 | 0.386  
## PROTEINS\_100G | 0.496 12.286 0.246 | -0.681 41.320 0.464 | 0.531  
## ctr cos2   
## ENERGY\_100G 0.159 0.001 |  
## FAT\_100G 35.012 0.233 |  
## CARBOHYDRATES\_100G 22.434 0.149 |  
## PROTEINS\_100G 42.394 0.282 |

fviz\_eig(tofu\_pca, ncp = 6, addlabels = T, main = "Variance explained by each dimensions TOfu")



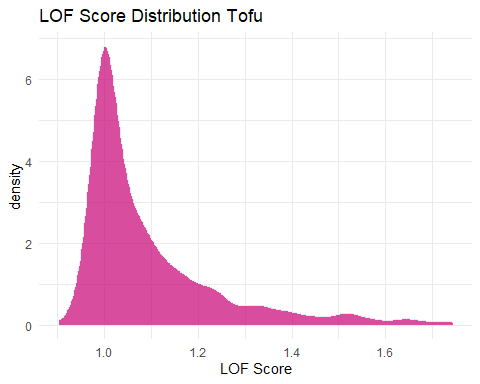
library(ggthemes)  
tofu\_a <- data.frame(tofu\_pca$ind$coord[,1:3])  
tofu\_b <- cbind(tofu\_a, lof\_score = tofu\_lof)  
#tofu\_b <- cbind(tofu\_a, fraud = tofu\_clean$, lof\_score = tofu\_clean$lof)  
  
tofu\_lof\_visual <- ggplot(tofu\_b, aes(x=Dim.1 ,y=Dim.2)) +   
 geom\_point(aes(size=lof\_score)) +  
 ggtitle("LOF Score Distribution Tofu")+  
 theme\_minimal()  
  
tofu\_lof\_visual



summary(tofu\_b)

## Dim.1 Dim.2 Dim.3 lof\_score   
## Min. :-2.8170 Min. :-6.3239 Min. :-6.57541 Min. :0.9036   
## 1st Qu.:-0.8214 1st Qu.:-0.6949 1st Qu.:-0.25797 1st Qu.:1.0000   
## Median :-0.1787 Median :-0.2655 Median : 0.01589 Median :1.0532   
## Mean : 0.0000 Mean : 0.0000 Mean : 0.00000 Mean : Inf   
## 3rd Qu.: 0.6344 3rd Qu.: 0.4581 3rd Qu.: 0.30434 3rd Qu.:1.2100   
## Max. :25.5570 Max. : 7.8974 Max. : 6.35320 Max. : Inf

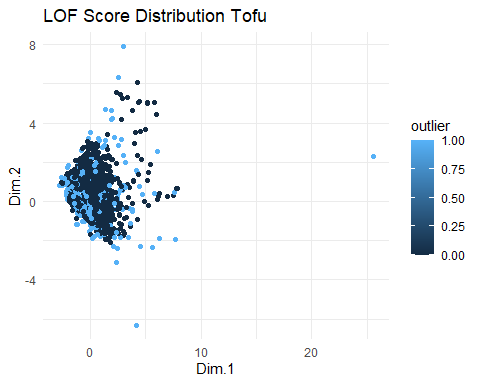
tofu\_b %>%  
 filter(lof\_score <= 1.75) %>%   
 ggplot( aes(x=lof\_score)) +  
 geom\_density( color="#e9ecef", fill = "#c90076", alpha=0.7) +  
 scale\_fill\_manual(values="#8fce00") +  
 xlab("LOF Score")+  
 ggtitle("LOF Score Distribution Tofu")+  
 theme\_minimal() +  
 labs(fill="")



quantile(tofu\_b$lof\_score, probs = c(0, 0.8))

## 0% 80%   
## 0.9036087 1.2815631

tofu\_b <- tofu\_b %>%   
 mutate(outlier = ifelse(lof\_score > 1.2815631, 1, 0))  
  
tofu\_lof\_visual\_b <- ggplot(tofu\_b, aes(x=Dim.1 ,y=Dim.2, color=outlier)) +   
 geom\_point() +  
 ggtitle("LOF Score Distribution Tofu")+  
 theme\_minimal()  
  
tofu\_lof\_visual\_b



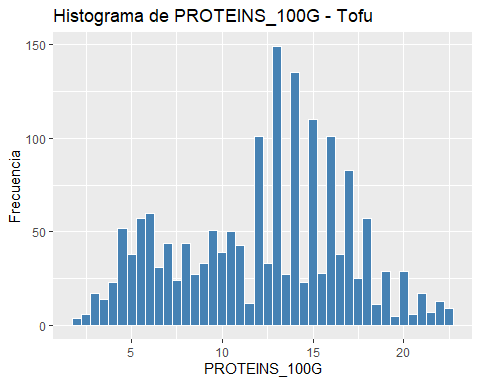
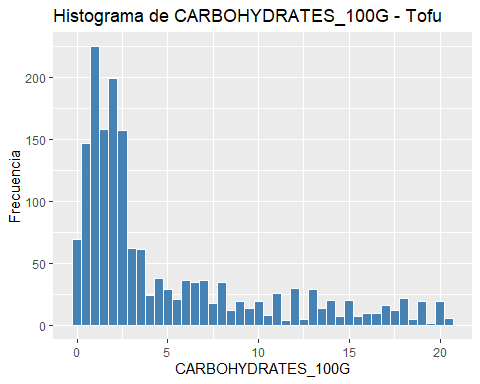
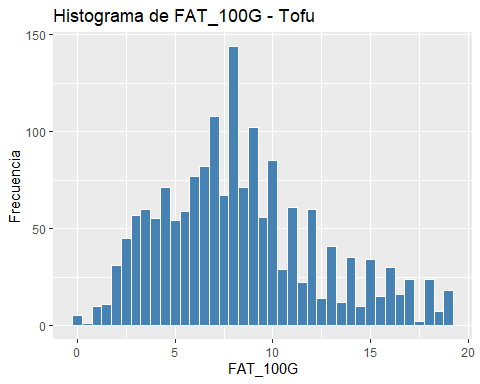
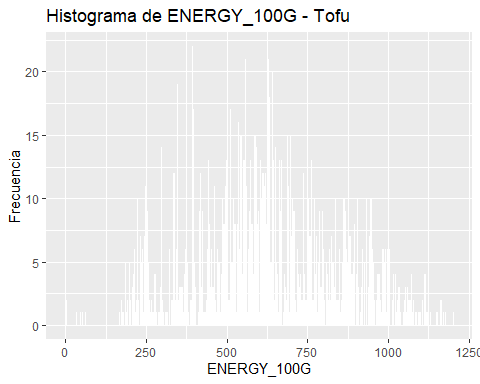
outliers <- tofu\_b[tofu\_b$outlier==1,]  
nooutliers <- tofu\_b[tofu\_b$outlier==0,]

df\_tofu <- df\_tofu[tofu\_lof < 1.2822731,]  
df\_tofu\_outliers <- df\_tofu[tofu\_lof >= 1.2822731,]

# Eliminar valores atipicos univariantes

remove\_outliers <- function(data, column, sd\_threshold = 2) {  
 data[abs(scale(data[[column]])) < sd\_threshold, ]  
}  
  
columns\_to\_check <- 3:ncol(df\_tofu)  
for (column in columns\_to\_check) {  
 df\_tofu <- remove\_outliers(df\_tofu, column)  
}

# Obtener las columnas cuantitativas del dataframe  
columnas\_cuantitativas <- sapply(df\_tofu, is.numeric)  
  
# Crear un histograma para cada columna cuantitativa  
for (columna in names(df\_tofu[columnas\_cuantitativas])) {  
 plot\_data <- df\_tofu[, columna]  
 p <- ggplot(data.frame(x = plot\_data), aes(x)) +  
 geom\_histogram(binwidth = 0.5, fill = "steelblue", color = "white") +  
 labs(title = paste("Histograma de", columna,"- Tofu"),  
 x = columna,  
 y = "Frecuencia")  
   
 print(p)  
}

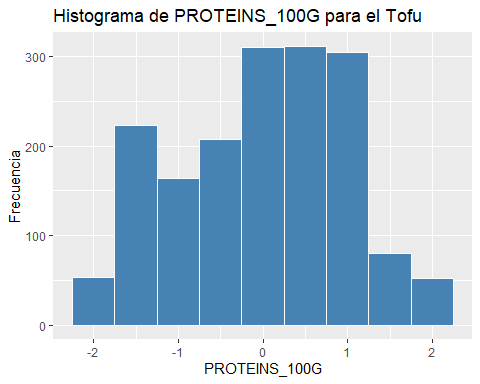
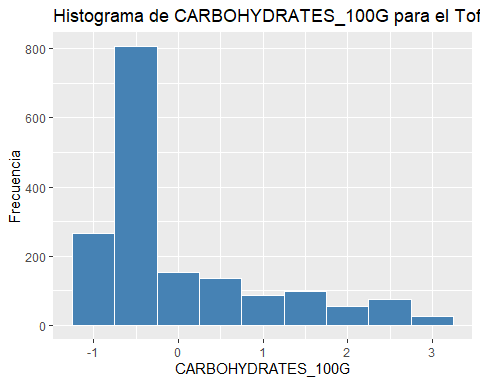
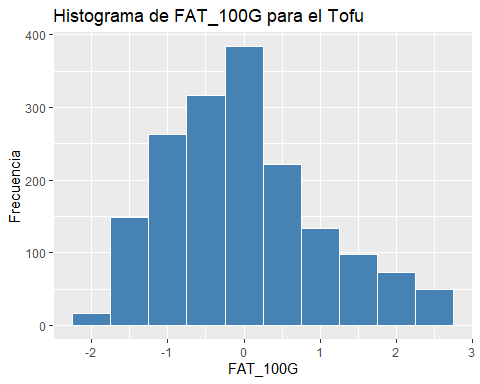
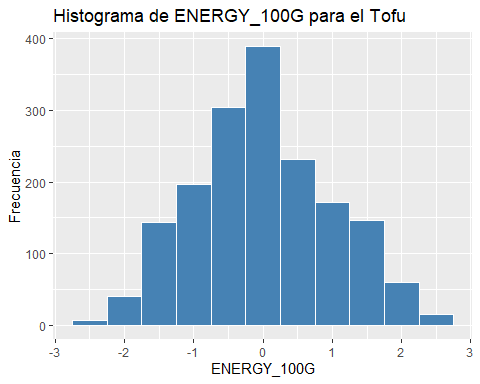


summary(as.data.frame(scale(df\_tofu[,-1:-2])))

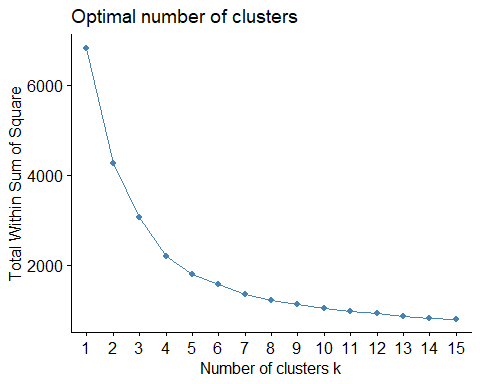
## ENERGY\_100G FAT\_100G CARBOHYDRATES\_100G PROTEINS\_100G   
## Min. :-2.74021 Min. :-2.05837 Min. :-0.9301 Min. :-2.1914   
## 1st Qu.:-0.67723 1st Qu.:-0.75421 1st Qu.:-0.7025 1st Qu.:-0.8305   
## Median :-0.05834 Median :-0.09722 Median :-0.4839 Median : 0.1723   
## Mean : 0.00000 Mean : 0.00000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.67267 3rd Qu.: 0.56711 3rd Qu.: 0.3973 3rd Qu.: 0.7740   
## Max. : 2.64594 Max. : 2.59936 Max. : 2.9574 Max. : 2.2136

#Datos normalizados para el tofu

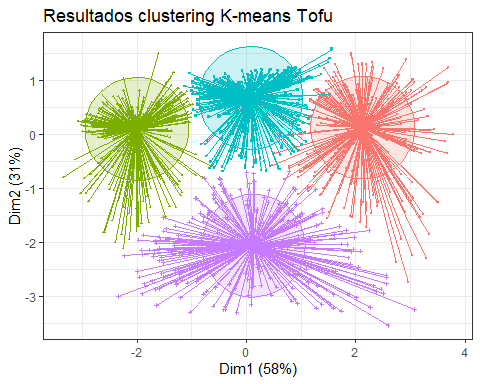
s\_tofu <- as.data.frame(scale(df\_tofu[,-1:-2]))  
# Obtener las columnas cuantitativas del dataframe  
columnas\_cuantitativas <- sapply(s\_tofu, is.numeric)  
  
# Crear un histograma para cada columna cuantitativa  
for (columna in names(s\_tofu[columnas\_cuantitativas])) {  
 plot\_data <- s\_tofu[, columna]  
 p <- ggplot(data.frame(x = plot\_data), aes(x)) +  
 geom\_histogram(binwidth = 0.5, fill = "steelblue", color = "white") +  
 labs(title = paste("Histograma de", columna,"para el Tofu"),  
 x = columna,  
 y = "Frecuencia")  
   
 print(p)  
}



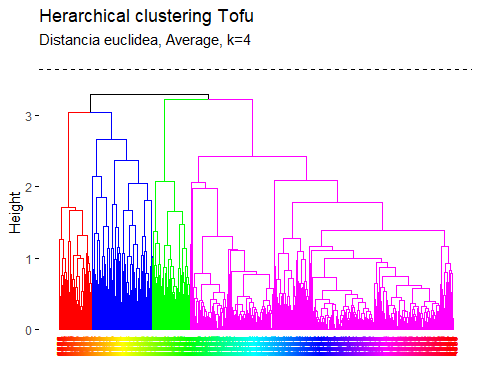
s\_tofu <- scale(df\_tofu[,-1:-2])  
  
# total de cluster óptimos  
elbow <- fviz\_nbclust(x = s\_tofu, FUNcluster = kmeans, method = "wss", k.max = 15,   
 diss = get\_dist(s\_tofu, method = "euclidean"), nstart = 25)  
print(elbow)



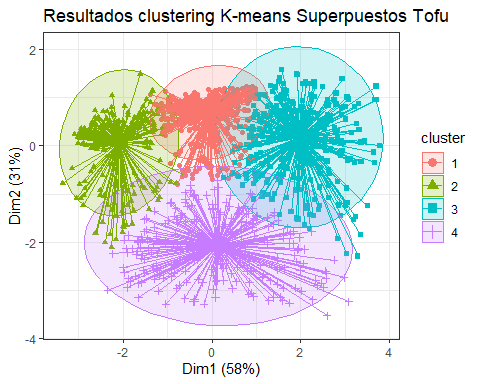
# set.seed(123)  
# km\_clusters <- kmeans(x=s\_tofu,centers=4,nstart=25)  
# fviz\_cluster(object=km\_clusters,data=s\_tofu,show.clust.cent = TRUE,  
# ellipse.type="euclid",star.plot=TRUE,repel=TRUE,  
# pointsize=0.5,outlier.color="darkred") +  
# labs(title ="Resultados clustering K-means") +  
# theme\_bw() +  
# theme(legend.position = "none")  
  
set.seed(123)  
km\_clusters <- kmeans(x = s\_tofu, centers = 4, nstart = 25)  
  
fviz\_cluster(object = km\_clusters, data = s\_tofu, show.clust.cent = TRUE,  
 ellipse.type = "euclid", star.plot = TRUE, repel = TRUE,  
 pointsize = 0.5, outlier.color = "darkred", geom = "point") +  
 theme\_bw() +  
 theme(legend.position = "none") +  
 ggtitle("Resultados clustering K-means Tofu")



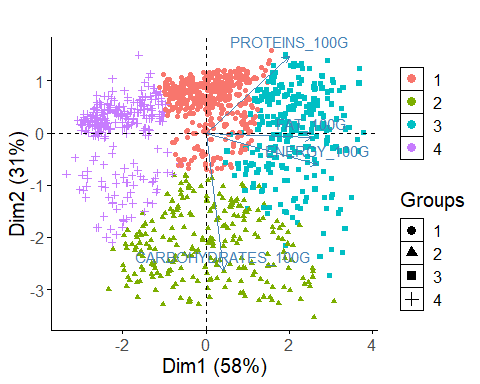
set.seed(101)  
hc\_euclidea\_av <- hclust(d = dist(x = s\_tofu, method = "euclidean"),  
 method = "average")  
fviz\_dend(x = hc\_euclidea\_av, k = 4, cex = 0.5,  
 k\_colors = c("red","blue","green","magenta"),color\_labels\_by\_k = T,  
 lwd = 0.2,type = "r",label\_cols = rainbow(nrow(df\_tofu)),  
 rect\_lty = "lightblue") +  
 geom\_hline(yintercept = 3.65, linetype = "dashed") +  
 labs(title = "Herarchical clustering Tofu",  
 subtitle = "Distancia euclidea, Average, k=4")



pam.res <- pam(s\_tofu, 4)  
# Visualización  
fviz\_cluster(pam.res, geom = "point", ellipse.type = "norm",  
 show.clust.cent = TRUE,star.plot = TRUE)+  
 labs(title = "Resultados clustering K-means Superpuestos Tofu")+ theme\_bw()

 # Biplot PCA y K-Means para medir representatividad tofu

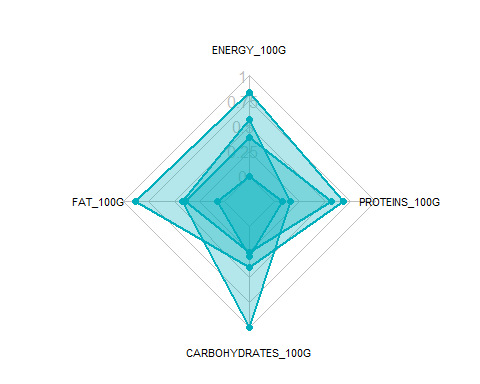
# PCA  
pca <- prcomp(df\_tofu[,-1:-2], scale=TRUE)  
df\_tofu.pca <- pca$x  
# Cluster over the three first PCA dimensions  
kc <- kmeans(df\_tofu.pca[,1:3], 4)  
fviz\_pca\_biplot(pca, label="var", habillage=as.factor(kc$cluster)) +  
 labs(color=NULL) + ggtitle("") +  
 theme(text = element\_text(size = 15),  
 panel.background = element\_blank(),   
 panel.grid.major = element\_blank(),  
 panel.grid.minor = element\_blank(),  
 axis.line = element\_line(colour = "black"),  
 legend.key = element\_rect(fill = "white"))



library(ggplot2)  
  
# Obtener los centroides de cada clúster  
centroids <- as.data.frame(km\_clusters$centers)  
  
  
# Obtener el valor mínimo y máximo en el dataframe  
min\_value <- min(centroids, na.rm = TRUE)  
max\_value <- max(centroids, na.rm = TRUE)  
  
# Escalar los datos entre 0 y 1  
scaled\_df <- (centroids - min\_value) / (max\_value - min\_value)  
  
library(fmsb)  
  
  
# Define the variable ranges: maximum and minimum  
max\_min <- data.frame(  
 ENERGY\_100G = c(1, 0), FAT\_100G = c(1, 0), CARBOHYDRATES\_100G = c(1, 0),PROTEINS\_100G = c(1, 0)  
)  
rownames(max\_min) <- c("Max", "Min")  
  
# Bind the variable ranges to the data  
df <- rbind(max\_min, scaled\_df)

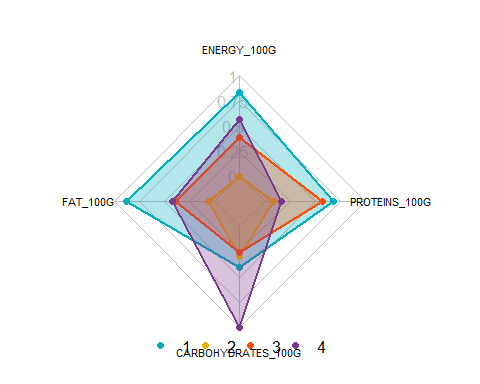
create\_beautiful\_radarchart <- function(data, color = "#00AFBB",   
 vlabels = colnames(data), vlcex = 0.7,  
 caxislabels = NULL, title = NULL, ...){  
 radarchart(  
 data, axistype = 1,  
 # Customize the polygon  
 pcol = color, pfcol = scales::alpha(color, 0.3), plwd = 2, plty = 1,  
 # Customize the grid  
 cglcol = "grey", cglty = 1, cglwd = 0.8,  
 # Customize the axis  
 axislabcol = "grey",   
 # Variable labels  
 vlcex = vlcex, vlabels = vlabels,  
 caxislabels = caxislabels, title = title, ...  
 )  
}

# Reduce plot margin using par()  
op <- par(mar = c(1, 2, 2, 1))  
create\_beautiful\_radarchart(df, caxislabels = c(0, .25, .5, .75, 1))



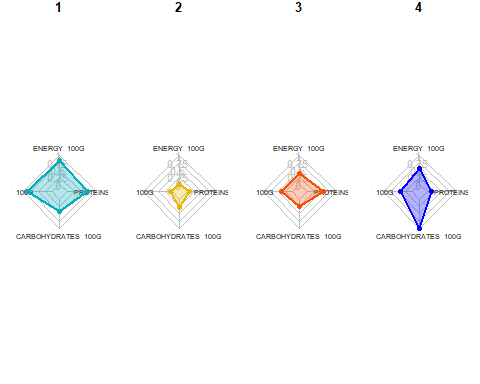
par(op)

# Reduce plot margin using par()  
op <- par(mar = c(1, 2, 2, 2))  
# Create the radar charts  
create\_beautiful\_radarchart(  
 data = df, caxislabels = c(0, .25, .5, .75, 1),  
 color = c("#00AFBB", "#E7B800", "#FC4E07","#7A378B")  
)  
# Add an horizontal legend  
legend(  
 x = "bottom", legend = rownames(df[-c(1,2),]), horiz = TRUE,  
 bty = "n", pch = 20 , col = c("#00AFBB", "#E7B800", "#FC4E07","#7A378B"),  
 text.col = "black", cex = 1, pt.cex = 1.5  
 )



par(op)

# Define colors and titles  
colors <- c("#00AFBB", "#E7B800", "#FC4E07","blue")  
titles <- c("1", "2", "3","4")  
  
# Reduce plot margin using par()  
# Split the screen in 3 parts  
op <- par(mar = c(1, 1, 1, 1))  
  
par(mfrow = c(1,4))  
  
# Create the radar chart  
for(i in 1:4){  
 create\_beautiful\_radarchart(  
 data = df[c(1, 2, i+2), ], caxislabels = c(0, .25, .5, .75, 1),  
 color = colors[i], title = titles[i]  
 )  
}



par(op)

# Visualizar los productos más cercanos al centroide que representan cada cluster top10

# Realizar clustering en el dataframe  
set.seed(123)  
df\_top <- df\_tofu  
km\_clusters <- kmeans(x = df\_top[, c("ENERGY\_100G", "FAT\_100G", "CARBOHYDRATES\_100G", "PROTEINS\_100G")], centers = 4, nstart = 50)  
  
# Obtener las asignaciones de clúster  
cluster\_assignments <- km\_clusters$cluster  
  
# Agregar las asignaciones de clúster al dataframe  
df\_top$cluster <- cluster\_assignments  
  
# Inicializar una lista para almacenar los productos más cercanos a cada centroide  
closest\_products <- vector("list", max(cluster\_assignments))  
  
# Encontrar los productos más cercanos a cada centroide  
for (cluster in 1:max(cluster\_assignments)) {  
 cluster\_center <- km\_clusters$centers[cluster, ]  
 distances <- apply(df\_top[, c("ENERGY\_100G", "FAT\_100G", "CARBOHYDRATES\_100G", "PROTEINS\_100G")], 1, function(row) {  
 sum((row - cluster\_center)^2)  
 })  
 closest\_products[[cluster]] <- head(order(distances), 10)  
}  
  
# Imprimir los productos más cercanos a cada centroide  
for (cluster in 1:max(cluster\_assignments)) {  
 cat("Cluster", cluster, ":\n")  
 print(df\_top[closest\_products[[cluster]], c("PRODUCT\_NAME", "ENERGY\_100G", "FAT\_100G", "CARBOHYDRATES\_100G", "PROTEINS\_100G")])  
 cat("\n")  
}

## Cluster 1 :  
## PRODUCT\_NAME ENERGY\_100G FAT\_100G CARBOHYDRATES\_100G  
## 385 MEDIUM FIRM TOFU 295 3.529412 2.352941  
## 723 TOFU 295 3.529412 2.352941  
## 730 HOUSE FOODS TOFU REGULAR 295 3.529412 2.352941  
## 735 TOFU 295 3.529412 2.352941  
## 749 TOFU MEDIUM FIRM 295 3.529412 2.352941  
## 1429 ORGANIC SILKEN TOFU 295 3.529412 2.352941  
## 1432 PULMUONE TOFU 295 3.529412 2.352941  
## 736 SOFT TOFU 295 3.529412 2.352941  
## 1577 FIRM TOFU 295 3.529412 2.352941  
## 1796 TOFU 295 3.529412 2.352941  
## PROTEINS\_100G  
## 385 7.058824  
## 723 7.058824  
## 730 7.058824  
## 735 7.058824  
## 749 7.058824  
## 1429 7.058824  
## 1432 7.058824  
## 736 5.882353  
## 1577 8.235294  
## 1796 8.235294  
##   
## Cluster 2 :  
## PRODUCT\_NAME ENERGY\_100G FAT\_100G CARBOHYDRATES\_100G  
## 6719 TOFU FUME 707 10.0000 3.20  
## 2014 BIO-TOFU 707 9.5000 2.00  
## 2346 BIO TOFU GERAUCHERT 707 9.5000 2.00  
## 5122 RAUCHER-TOFU 707 9.5000 2.00  
## 5084 BIO-TOFU GERAUCHERT 708 9.7143 2.00  
## 1800 TOFU YU, TERIYAKI TOFU SUSHI 703 10.1800 7.96  
## 4832 BIO TOFU GERAUCHERT 704 9.5000 2.00  
## 5070 RAUCHER TOFU SCHNITTFEST 704 9.5000 2.00  
## 5100 RAUCHER-TOFU KRAFTIG 704 9.5000 2.00  
## 2618 TOFU NATURE 705 9.8000 0.70  
## PROTEINS\_100G  
## 6719 16.0000  
## 2014 18.0000  
## 2346 18.0000  
## 5122 18.0000  
## 5084 18.2857  
## 1800 15.4900  
## 4832 18.0000  
## 5070 18.0000  
## 5100 18.0000  
## 2618 18.0000  
##   
## Cluster 3 :  
## PRODUCT\_NAME ENERGY\_100G FAT\_100G CARBOHYDRATES\_100G  
## 7389 COTOLETTA DI TOFU 950 14.00 12.00  
## 7571 COTOLETTA DI TOFU E SEITAN 946 11.79 10.80  
## 7562 COTOLETTA DI TOFU E SEITAN 946 11.70 10.80  
## 8407 SALCHICHAS TOFU FRANKFURT 950 13.47 5.44  
## 8698 HAPPYBURGER TOFU CHAMPIGNON 954 14.00 13.00  
## 6751 QUINOA TOFU BALLS 950 14.00 12.00  
## 8143 VEGEBURGER DE TOFU Y CHAMPINONES 954 13.70 13.10  
## 8161 VEGEBURGER DE TOFU Y CHAMPINONES 954 13.70 13.10  
## 8686 VEGBURGUER TOFU Y CHAMPINONES 954 13.70 13.10  
## 5446 FISH TOFU 946 15.40 11.20  
## PROTEINS\_100G  
## 7389 12.00  
## 7571 15.60  
## 7562 15.60  
## 8407 20.61  
## 8698 11.00  
## 6751 9.00  
## 8143 10.70  
## 8161 10.70  
## 8686 10.70  
## 5446 10.60  
##   
## Cluster 4 :  
## PRODUCT\_NAME ENERGY\_100G FAT\_100G  
## 8662 TOFU CON SEMILLA DE AMAPOLA 527 6.80  
## 695 ORGANIC FIRM TOFU 531 6.33  
## 807 O ORGANICS, ORGANIC EXTRA FIRM CUBED TOFU 531 6.33  
## 8399 ORGANIC TOFU JAPONES 527 8.00  
## 4530 TOFU TERIYAKI 531 7.00  
## 4313 FETO TOFU FERMENTE AUX HERBES 527 6.40  
## 4608 BIO TOFU 527 7.90  
## 4611 BIO TOFU NATURE 527 7.90  
## 4697 TOFU NATUR 527 7.90  
## 6437 BIO TOFU AL NATURALE 527 7.90  
## CARBOHYDRATES\_100G PROTEINS\_100G  
## 8662 3.3 12.20  
## 695 3.8 12.66  
## 807 3.8 12.66  
## 8399 0.7 12.26  
## 4530 3.0 13.00  
## 4313 1.9 15.00  
## 4608 0.0 13.00  
## 4611 0.0 13.00  
## 4697 0.0 13.00  
## 6437 0.0 13.00

#### SOja

# Aplicar algoritmo Local Outliers Factor para eliminar atípicos  
  
# 1.- First, we need to convert the incorrect type of variables.  
  
soja\_clean <- df\_soja %>%  
 mutate(PRODUCT\_NAME = as.factor(PRODUCT\_NAME))  
str(soja\_clean)

## 'data.frame': 2066 obs. of 6 variables:  
## $ ISO3 : chr "DEU" "FRA" "FRA" "FRA" ...  
## $ PRODUCT\_NAME : Factor w/ 1611 levels "100% SOJA PROTEIN HASELNUSS",..: 1 864 974 348 347 732 340 575 2 1094 ...  
## $ ENERGY\_100G : num 1590 1506 238 992 992 ...  
## $ FAT\_100G : num 0.5 1.3 2.2 11 11 7.2 12 11 0.5 12.4 ...  
## $ CARBOHYDRATES\_100G: num 1.1 6.1 5 18 18 10 16 18 1.1 4.7 ...  
## $ PROTEINS\_100G : num 88.5 80 4.1 14 14 8.3 16 14 88.5 10.2 ...

soja\_clean <- as.data.frame(scale(df\_soja[,-1:-2]))  
soja\_lof <- lof(soja\_clean, minPts = 7)  
head(soja\_clean)

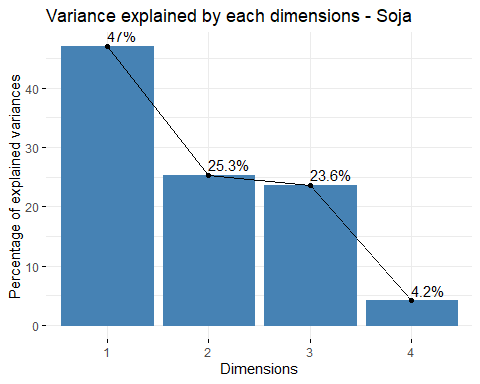
## ENERGY\_100G FAT\_100G CARBOHYDRATES\_100G PROTEINS\_100G  
## 3 1.0741605 -0.74192648 -0.6873522 3.9649794  
## 7 0.9447013 -0.66400852 -0.4500815 3.4940833  
## 14 -1.0095166 -0.57635082 -0.5022811 -0.7107426  
## 15 0.1525341 0.28074672 0.1146227 -0.1622871  
## 16 0.1525341 0.28074672 0.1146227 -0.1622871  
## 17 -0.4284913 -0.08936358 -0.2650104 -0.4780645

#PCA and use the first two dimensions of the PCA  
library(FactoMineR)  
library(factoextra)  
  
soja\_pca <- PCA(soja\_clean, scale.unit = F, ncp = 6, graph = F)

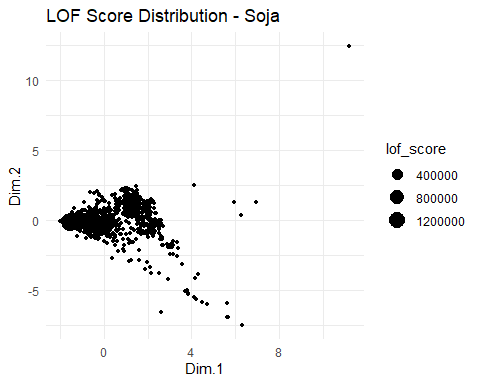
summary(soja\_pca)

##   
## Call:  
## PCA(X = soja\_clean, scale.unit = F, ncp = 6, graph = F)   
##   
##   
## Eigenvalues  
## Dim.1 Dim.2 Dim.3 Dim.4  
## Variance 1.879 1.011 0.942 0.166  
## % of var. 47.003 25.283 23.563 4.151  
## Cumulative % of var. 47.003 72.285 95.849 100.000  
##   
## Individuals (the 10 first)  
## Dist Dim.1 ctr cos2 Dim.2 ctr cos2   
## 3 | 4.231 | 2.026 0.106 0.229 | 1.362 0.089 0.104 |  
## 7 | 3.707 | 1.832 0.086 0.244 | 1.300 0.081 0.123 |  
## 14 | 1.452 | -1.450 0.054 0.998 | -0.064 0.000 0.002 |  
## 15 | 0.376 | 0.192 0.001 0.260 | -0.196 0.002 0.272 |  
## 16 | 0.376 | 0.192 0.001 0.260 | -0.196 0.002 0.272 |  
## 17 | 0.700 | -0.647 0.011 0.853 | -0.228 0.002 0.106 |  
## 18 | 0.439 | 0.298 0.002 0.460 | -0.292 0.004 0.441 |  
## 19 | 0.376 | 0.192 0.001 0.260 | -0.196 0.002 0.272 |  
## 23 | 4.231 | 2.026 0.106 0.229 | 1.362 0.089 0.104 |  
## 24 | 0.786 | -0.300 0.002 0.146 | -0.727 0.025 0.854 |  
## Dim.3 ctr cos2   
## 3 -3.429 0.604 0.657 |  
## 7 -2.920 0.438 0.620 |  
## 14 0.030 0.000 0.000 |  
## 15 0.257 0.003 0.465 |  
## 16 0.257 0.003 0.465 |  
## 17 0.139 0.001 0.039 |  
## 18 0.138 0.001 0.099 |  
## 19 0.257 0.003 0.465 |  
## 23 -3.429 0.604 0.657 |  
## 24 0.014 0.000 0.000 |  
##   
## Variables  
## Dim.1 ctr cos2 Dim.2 ctr cos2 Dim.3  
## ENERGY\_100G | 0.956 48.618 0.914 | -0.027 0.072 0.001 | 0.000  
## FAT\_100G | 0.596 18.908 0.355 | -0.748 55.304 0.559 | 0.227  
## CARBOHYDRATES\_100G | 0.458 11.185 0.210 | 0.593 34.775 0.352 | 0.650  
## PROTEINS\_100G | 0.632 21.288 0.400 | 0.316 9.849 0.100 | -0.684  
## ctr cos2   
## ENERGY\_100G 0.000 0.000 |  
## FAT\_100G 5.458 0.051 |  
## CARBOHYDRATES\_100G 44.854 0.423 |  
## PROTEINS\_100G 49.687 0.468 |

fviz\_eig(soja\_pca, ncp = 6, addlabels = T, main = "Variance explained by each dimensions - Soja")



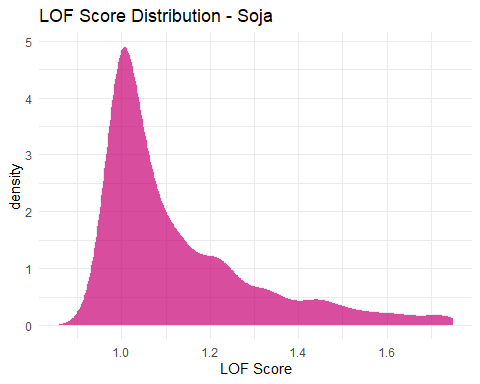
library(ggthemes)  
soja\_a <- data.frame(soja\_pca$ind$coord[,1:3])  
soja\_b <- cbind(soja\_a, lof\_score = soja\_lof)  
#soja\_b <- cbind(soja\_a, fraud = soja\_clean$, lof\_score = soja\_clean$lof)  
  
soja\_lof\_visual <- ggplot(soja\_b, aes(x=Dim.1 ,y=Dim.2)) +   
 geom\_point(aes(size=lof\_score)) +  
 ggtitle("LOF Score Distribution - Soja")+  
 theme\_minimal()  
  
soja\_lof\_visual



summary(soja\_b)

## Dim.1 Dim.2 Dim.3 lof\_score   
## Min. :-1.9743 Min. :-7.50403 Min. :-3.59918 Min. :0.8579   
## 1st Qu.:-1.2351 1st Qu.:-0.34531 1st Qu.:-0.35626 1st Qu.:1.0058   
## Median :-0.3107 Median :-0.03605 Median : 0.01204 Median :1.0928   
## Mean : 0.0000 Mean : 0.00000 Mean : 0.00000 Mean : Inf   
## 3rd Qu.: 1.1951 3rd Qu.: 0.38722 3rd Qu.: 0.32017 3rd Qu.:1.3530   
## Max. :11.2007 Max. :12.42671 Max. :18.73336 Max. : Inf

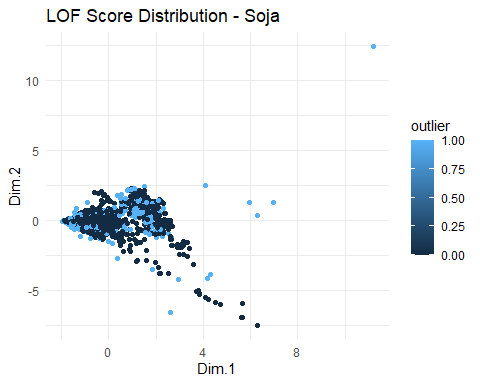
soja\_b %>%  
 filter(lof\_score <= 1.75) %>%   
 ggplot( aes(x=lof\_score)) +  
 geom\_density( color="#e9ecef", fill = "#c90076", alpha=0.7) +  
 scale\_fill\_manual(values="#8fce00") +  
 xlab("LOF Score")+  
 ggtitle("LOF Score Distribution - Soja")+  
 theme\_minimal() +  
 labs(fill="")



quantile(soja\_b$lof\_score, probs = c(0, 0.8))

## 0% 80%   
## 0.857934 1.485425

soja\_b <- soja\_b %>%   
 mutate(outlier = ifelse(lof\_score > 1.485425, 1, 0))  
  
soja\_lof\_visual\_b <- ggplot(soja\_b, aes(x=Dim.1 ,y=Dim.2, color=outlier)) +   
 geom\_point() +  
 ggtitle("LOF Score Distribution - Soja")+  
 theme\_minimal()  
  
soja\_lof\_visual\_b



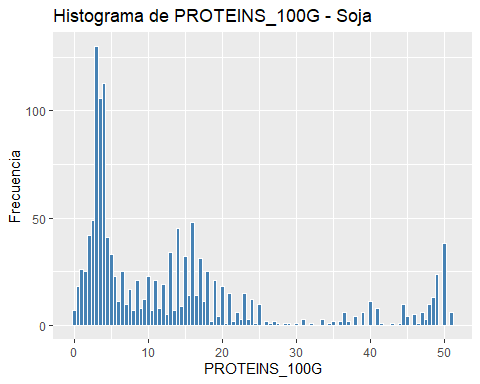
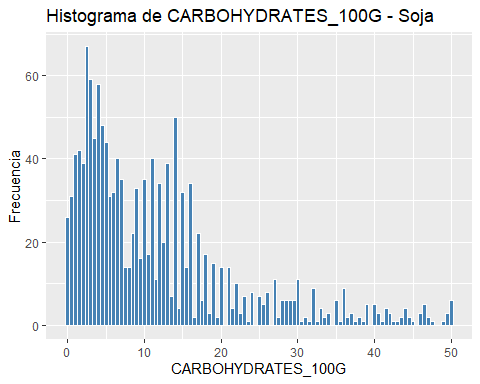
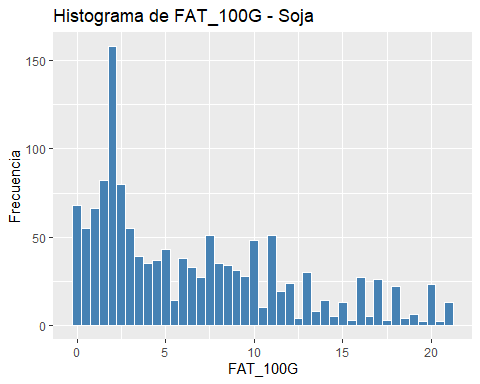
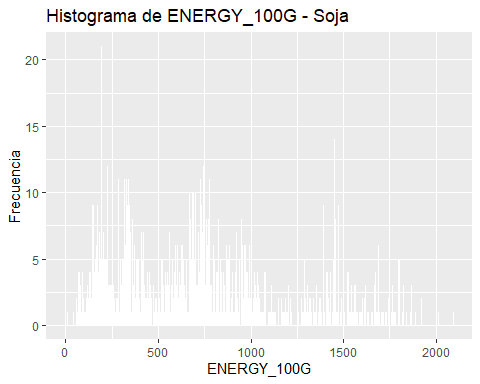
outliers <- soja\_b[soja\_b$outlier==1,]  
nooutliers <- soja\_b[soja\_b$outlier==0,]

df\_soja <- df\_soja[soja\_lof < 1.485425,]  
df\_soja\_outliers <- df\_soja[soja\_lof >= 1.485425,]

# Eliminar valores atipicos univariantes

remove\_outliers <- function(data, column, sd\_threshold = 2) {  
 data[abs(scale(data[[column]])) < sd\_threshold, ]  
}  
  
columns\_to\_check <- 3:ncol(df\_soja)  
for (column in columns\_to\_check) {  
 df\_soja <- remove\_outliers(df\_soja, column)  
}

# Obtener las columnas cuantitativas del dataframe  
columnas\_cuantitativas <- sapply(df\_soja, is.numeric)  
  
# Crear un histograma para cada columna cuantitativa  
for (columna in names(df\_soja[columnas\_cuantitativas])) {  
 plot\_data <- df\_soja[, columna]  
 p <- ggplot(data.frame(x = plot\_data), aes(x)) +  
 geom\_histogram(binwidth = 0.5, fill = "steelblue", color = "white") +  
 labs(title = paste("Histograma de", columna,"- Soja"),  
 x = columna,  
 y = "Frecuencia")  
   
 print(p)  
}

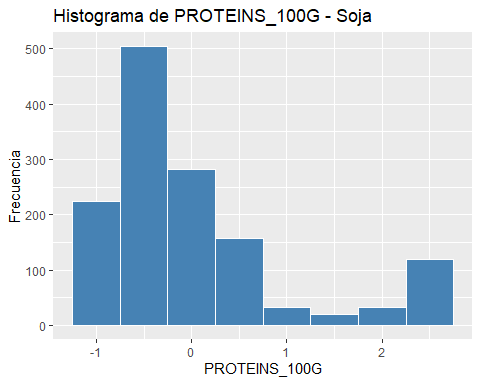
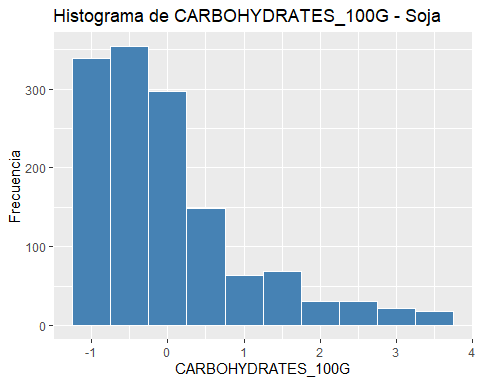
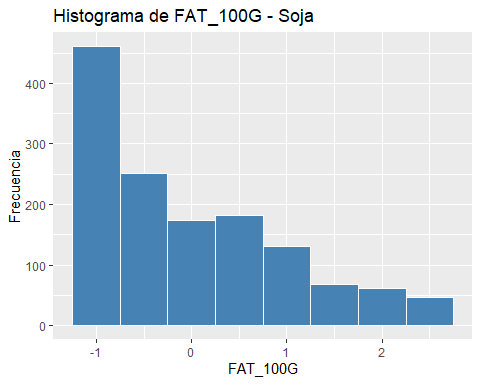
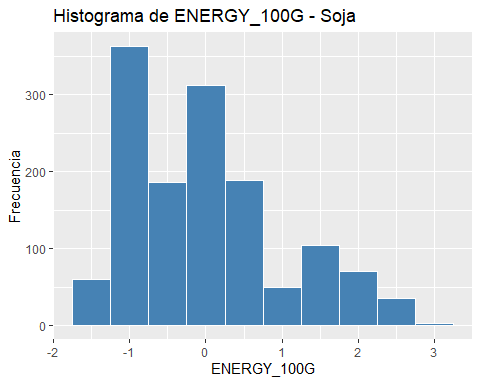


summary(as.data.frame(scale(df\_soja[,-1:-2])))

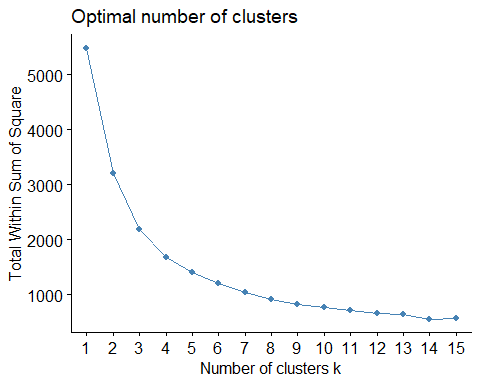
## ENERGY\_100G FAT\_100G CARBOHYDRATES\_100G PROTEINS\_100G   
## Min. :-1.53946 Min. :-1.1866 Min. :-1.0813 Min. :-0.9730   
## 1st Qu.:-0.82857 1st Qu.:-0.8203 1st Qu.:-0.7347 1st Qu.:-0.7118   
## Median :-0.08933 Median :-0.2708 Median :-0.2663 Median :-0.3708   
## Mean : 0.00000 Mean : 0.0000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.54087 3rd Qu.: 0.6450 3rd Qu.: 0.3613 3rd Qu.: 0.2605   
## Max. : 3.02243 Max. : 2.6596 Max. : 3.6024 Max. : 2.7275

#Datos normalizados para el soja

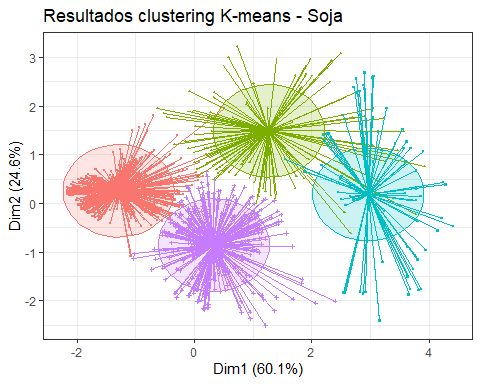
s\_soja <- as.data.frame(scale(df\_soja[,-1:-2]))  
# Obtener las columnas cuantitativas del dataframe  
columnas\_cuantitativas <- sapply(s\_soja, is.numeric)  
  
# Crear un histograma para cada columna cuantitativa  
for (columna in names(s\_soja[columnas\_cuantitativas])) {  
 plot\_data <- s\_soja[, columna]  
 p <- ggplot(data.frame(x = plot\_data), aes(x)) +  
 geom\_histogram(binwidth = 0.5, fill = "steelblue", color = "white") +  
 labs(title = paste("Histograma de", columna,"- Soja"),  
 x = columna,  
 y = "Frecuencia")  
   
 print(p)  
}



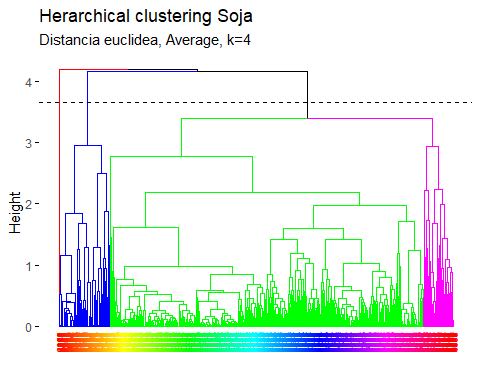
s\_soja <- scale(df\_soja[,-1:-2])  
  
# total de cluster óptimos  
elbow <- fviz\_nbclust(x = s\_soja, FUNcluster = kmeans, method = "wss", k.max = 15,   
 diss = get\_dist(s\_soja, method = "euclidean"), nstart = 25)  
print(elbow)



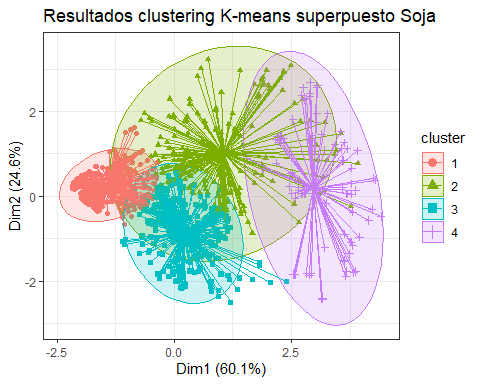
# set.seed(123)  
# km\_clusters <- kmeans(x=s\_soja,centers=4,nstart=25)  
# fviz\_cluster(object=km\_clusters,data=s\_soja,show.clust.cent = TRUE,  
# ellipse.type="euclid",star.plot=TRUE,repel=TRUE,  
# pointsize=0.5,outlier.color="darkred") +  
# labs(title ="Resultados clustering K-means") +  
# theme\_bw() +  
# theme(legend.position = "none")  
  
set.seed(123)  
km\_clusters <- kmeans(x = s\_soja, centers = 4, nstart = 50)  
  
fviz\_cluster(object = km\_clusters, data = s\_soja, show.clust.cent = TRUE,  
 ellipse.type = "euclid", star.plot = TRUE, repel = TRUE,  
 pointsize = 0.5, outlier.color = "darkred", geom = "point") +  
 theme\_bw() +  
 theme(legend.position = "none") +  
 ggtitle("Resultados clustering K-means - Soja")



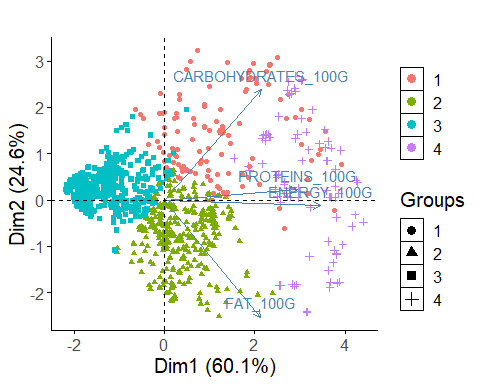
set.seed(101)  
hc\_euclidea\_av <- hclust(d = dist(x = s\_soja, method = "euclidean"),  
 method = "average")  
fviz\_dend(x = hc\_euclidea\_av, k = 4, cex = 0.5,  
 k\_colors = c("red","blue","green","magenta"),color\_labels\_by\_k = T,  
 lwd = 0.2,type = "r",label\_cols = rainbow(nrow(df\_soja)),  
 rect\_lty = "lightblue") +  
 geom\_hline(yintercept = 3.65, linetype = "dashed") +  
 labs(title = "Herarchical clustering Soja",  
 subtitle = "Distancia euclidea, Average, k=4")



pam.res <- pam(s\_soja, 4)  
# Visualización  
fviz\_cluster(pam.res, geom = "point", ellipse.type = "norm",  
 show.clust.cent = TRUE,star.plot = TRUE)+  
 labs(title = "Resultados clustering K-means superpuesto Soja")+ theme\_bw()

 # Biplot PCA y K-Means para medir representatividad soja

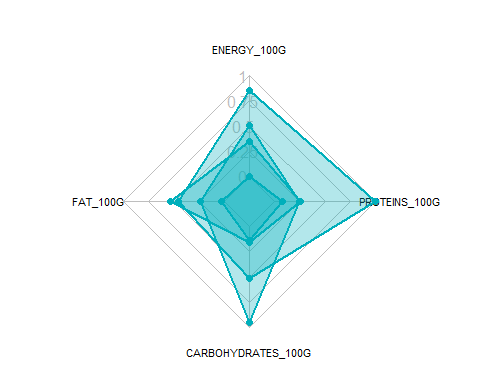
# PCA  
pca <- prcomp(df\_soja[,-1:-2], scale=TRUE)  
df\_soja.pca <- pca$x  
# Cluster over the three first PCA dimensions  
kc <- kmeans(df\_soja.pca[,1:3], 4)  
fviz\_pca\_biplot(pca, label="var", habillage=as.factor(kc$cluster)) +  
 labs(color=NULL) + ggtitle("") +  
 theme(text = element\_text(size = 15),  
 panel.background = element\_blank(),   
 panel.grid.major = element\_blank(),  
 panel.grid.minor = element\_blank(),  
 axis.line = element\_line(colour = "black"),  
 legend.key = element\_rect(fill = "white"))



library(ggplot2)  
  
# Obtener los centroides de cada clúster  
centroids <- as.data.frame(km\_clusters$centers)  
  
  
# Obtener el valor mínimo y máximo en el dataframe  
min\_value <- min(centroids, na.rm = TRUE)  
max\_value <- max(centroids, na.rm = TRUE)  
  
# Escalar los datos entre 0 y 1  
scaled\_df <- (centroids - min\_value) / (max\_value - min\_value)  
  
library(fmsb)  
  
  
# Define the variable ranges: maximum and minimum  
max\_min <- data.frame(  
 ENERGY\_100G = c(1, 0), FAT\_100G = c(1, 0), CARBOHYDRATES\_100G = c(1, 0),PROTEINS\_100G = c(1, 0)  
)  
rownames(max\_min) <- c("Max", "Min")  
  
# Bind the variable ranges to the data  
df <- rbind(max\_min, scaled\_df)

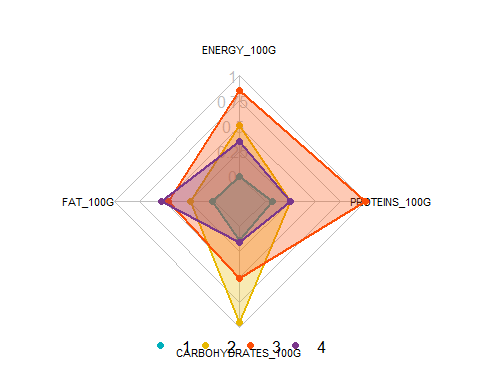
create\_beautiful\_radarchart <- function(data, color = "#00AFBB",   
 vlabels = colnames(data), vlcex = 0.7,  
 caxislabels = NULL, title = NULL, ...){  
 radarchart(  
 data, axistype = 1,  
 # Customize the polygon  
 pcol = color, pfcol = scales::alpha(color, 0.3), plwd = 2, plty = 1,  
 # Customize the grid  
 cglcol = "grey", cglty = 1, cglwd = 0.8,  
 # Customize the axis  
 axislabcol = "grey",   
 # Variable labels  
 vlcex = vlcex, vlabels = vlabels,  
 caxislabels = caxislabels, title = title, ...  
 )  
}

# Reduce plot margin using par()  
op <- par(mar = c(1, 2, 2, 1))  
create\_beautiful\_radarchart(df, caxislabels = c(0, .25, .5, .75, 1))



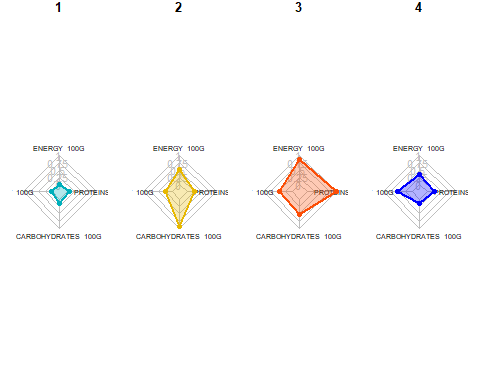
par(op)

# Reduce plot margin using par()  
op <- par(mar = c(1, 2, 2, 2))  
# Create the radar charts  
create\_beautiful\_radarchart(  
 data = df, caxislabels = c(0, .25, .5, .75, 1),  
 color = c("#00AFBB", "#E7B800", "#FC4E07","#7A378B")  
)  
# Add an horizontal legend  
legend(  
 x = "bottom", legend = rownames(df[-c(1,2),]), horiz = TRUE,  
 bty = "n", pch = 20 , col = c("#00AFBB", "#E7B800", "#FC4E07","#7A378B"),  
 text.col = "black", cex = 1, pt.cex = 1.5  
 )



par(op)

# Define colors and titles  
colors <- c("#00AFBB", "#E7B800", "#FC4E07","blue")  
titles <- c("1", "2", "3","4")  
  
# Reduce plot margin using par()  
# Split the screen in 3 parts  
op <- par(mar = c(1, 1, 1, 1))  
  
par(mfrow = c(1,4))  
  
# Create the radar chart  
for(i in 1:4){  
 create\_beautiful\_radarchart(  
 data = df[c(1, 2, i+2), ], caxislabels = c(0, .25, .5, .75, 1),  
 color = colors[i], title = titles[i]  
 )  
}



par(op)

# Visualizar los productos más cercanos al centroide que representan cada cluster top10

# Realizar clustering en el dataframe  
set.seed(123)  
df\_top <- df\_soja  
km\_clusters <- kmeans(x = df\_top[, c("ENERGY\_100G", "FAT\_100G", "CARBOHYDRATES\_100G", "PROTEINS\_100G")], centers = 4, nstart = 50)  
  
# Obtener las asignaciones de clúster  
cluster\_assignments <- km\_clusters$cluster  
  
# Agregar las asignaciones de clúster al dataframe  
df\_top$cluster <- cluster\_assignments  
  
# Inicializar una lista para almacenar los productos más cercanos a cada centroide  
closest\_products <- vector("list", max(cluster\_assignments))  
  
# Encontrar los productos más cercanos a cada centroide  
for (cluster in 1:max(cluster\_assignments)) {  
 cluster\_center <- km\_clusters$centers[cluster, ]  
 distances <- apply(df\_top[, c("ENERGY\_100G", "FAT\_100G", "CARBOHYDRATES\_100G", "PROTEINS\_100G")], 1, function(row) {  
 sum((row - cluster\_center)^2)  
 })  
 closest\_products[[cluster]] <- head(order(distances), 10)  
}  
  
# Imprimir los productos más cercanos a cada centroide  
for (cluster in 1:max(cluster\_assignments)) {  
 cat("Cluster", cluster, ":\n")  
 print(df\_top[closest\_products[[cluster]], c("PRODUCT\_NAME", "ENERGY\_100G", "FAT\_100G", "CARBOHYDRATES\_100G", "PROTEINS\_100G")])  
 cat("\n")  
}

## Cluster 1 :  
## PRODUCT\_NAME ENERGY\_100G FAT\_100G CARBOHYDRATES\_100G  
## 3165 SOJA FERMENTADA 264 2.3 6.4  
## 2895 BOULETTES DE SOJA ET TOMATES 262 1.5 6.3  
## 6816 SOJA 264 0.0 7.0  
## 4977 16G PROTEIN OHNE SOJA 268 0.3 8.8  
## 3062 SO SOJA ! BANANE-PASSION 259 2.0 7.0  
## 4772 GULASCH VEGAN MIT SOJA 259 1.1 6.5  
## 4338 SHOYU - SOJASOSSE JAPANISCHE ART 267 0.2 5.2  
## 4467 SHOYU SOJASOSSE AUS JAPAN 259 0.1 6.2  
## 2894 BOUCHEES SOJA ET LEGUMES 272 2.0 5.2  
## 2896 BOUCHEES DE SOJA ET LEGUMES 272 2.0 5.2  
## PROTEINS\_100G  
## 3165 4.1  
## 2895 4.0  
## 6816 8.7  
## 4977 6.4  
## 3062 3.6  
## 4772 4.0  
## 4338 10.0  
## 4467 8.8  
## 2894 4.5  
## 2896 4.5  
##   
## Cluster 2 :  
## PRODUCT\_NAME ENERGY\_100G  
## 5624 LA PANEE SOJA ET BLE 981  
## 6864 LA PANEE SOJA ET BLE 981  
## 6689 GARDEN GOURMET LA PANEE SOJA ET BLE 180G 984  
## 6701 GARDEN GOURMET LA PANEE SOJA ET BLE FORMAT FAMILIAL 360G 984  
## 3231 BURGER VEGETARIEN STEAK SOJA ET MOZZARELLA 980  
## 2496 GRILL VEGETAL - NUGGETS SOJA & BLE 975  
## 6441 NUGGETS SOJA & BLE 975  
## 5718 BURGERS DE LEGUMES AU SOJA BELGE 983  
## 3390 NUGGETS SOJA ET BLE 983  
## 15 ESCALOPES PANEES DE SOJA 992  
## FAT\_100G CARBOHYDRATES\_100G PROTEINS\_100G  
## 5624 11.66667 17.55556 12.22222  
## 6864 11.66667 17.55556 12.22222  
## 6689 11.70000 17.60000 12.30000  
## 6701 11.70000 17.60000 12.30000  
## 3231 10.00000 21.00000 13.00000  
## 2496 11.00000 14.00000 17.00000  
## 6441 11.00000 14.00000 17.00000  
## 5718 14.00000 9.20000 15.00000  
## 3390 8.00000 25.00000 13.00000  
## 15 11.00000 18.00000 14.00000  
##   
## Cluster 3 :  
## PRODUCT\_NAME ENERGY\_100G FAT\_100G  
## 2949 PROTEINES DE SOJA GROSSES 1544 8.0  
## 3254 PROTEINES DE SOJA GROS MORCEAUX 1540 8.0  
## 3256 PROTEINE DE SOJA PETITS MORCEAUX 1540 8.0  
## 4057 PROTEINE DE SOJA 1540 8.0  
## 8051 SOJA FINA TEXTURIZADA 1531 6.0  
## 4624 SOJASCHNETZEL 1561 9.0  
## 4868 PLANET NATURE SOJA FLEISCHSCHNETZEL ART 1561 9.0  
## 9251 BIO SOJA GRANULAT 1561 9.0  
## 3364 SOJA VERT 1536 18.1  
## 8560 FIDEOS DE SOJA BIOLOGICOS 1527 5.0  
## CARBOHYDRATES\_100G PROTEINS\_100G  
## 2949 14.0 50.0  
## 3254 14.0 50.0  
## 3256 14.0 50.0  
## 4057 14.0 50.0  
## 8051 32.0 46.0  
## 4624 16.0 50.0  
## 4868 16.0 50.0  
## 9251 16.0 50.0  
## 3364 6.3 33.7  
## 8560 19.9 26.8  
##   
## Cluster 4 :  
## PRODUCT\_NAME ENERGY\_100G FAT\_100G  
## 2513 CROQ' SOJA, POMME DE TERRE & EMMENTAL 674 8.1  
## 3584 GALETTES DE SOJA RECETTE MEDITERRANEENNE 678 7.6  
## 4040 BOULE DE SOJA ENROBEE CARAMEL CACAHUETE 678 6.8  
## 3619 BURGERS BLE, SOJA 674 5.9  
## 3022 STEAK DE SOJA TOMATE MOZZARELLA 674 6.5  
## 3583 GALETTES DE SOJA AUX 5 LEGUMES 682 8.0  
## 3020 STEAKS DE SOJA A LA PROVENÇALE 678 8.5  
## 3025 STEAKS DE SOJA A LA MEDITERRANEENNE 678 8.5  
## 6757 CHARCUTERIE A BASE DE SOJA ET POIS 682 11.0  
## 2500 HACHE VEGETAL (SOJA OIGNONS ET PERSIL) 674 4.5  
## CARBOHYDRATES\_100G PROTEINS\_100G  
## 2513 7.2 13.0  
## 3584 8.6 13.0  
## 4040 10.0 15.0  
## 3619 8.5 16.0  
## 3022 6.5 16.7  
## 3583 7.6 13.0  
## 3020 3.5 15.5  
## 3025 3.5 15.5  
## 6757 6.3 8.4  
## 2500 13.0 16.0