Proyecto de análisis + clasificación (k-NN, LDA y QDA) en R

José María Manzano Ortega

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
library(skimr)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(GGally)
```

```
## Warning: package 'GGally' was built under R version 4.3.3
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.3
```

```
## Warning: package 'tidyr' was built under R version 4.3.3
```

```
## Warning: package 'readr' was built under R version 4.3.3
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(caret)
```

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

```
## Warning: package 'lattice' was built under R version 4.3.3
```

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

```r
library(dplyr)

#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------

#Trabajo EDA (bupa)

#-------------------------------------------------------------------------------
#Importamos el dataset
bupa <- read.csv("bupa.dat", comment.char="@", header = FALSE)

#Asignamos manualmente los nombres a las distintas variables
names(bupa) <- c("Mcv", "Alkphos", "Sgpt",
                 "Sgot", "Gammagt", "Drinks", "Selector")
#Vemos las primeras filas del dataset para ver si se ha importado correctamente
head(bupa)
```

```
##   Mcv Alkphos Sgpt Sgot Gammagt Drinks Selector
## 1  85      92   45   27      31    0.0        1
## 2  85      64   59   32      23    0.0        2
## 3  86      54   33   16      54    0.0        2
## 4  91      78   34   24      36    0.0        2
## 5  98      55   13   17      17    0.0        2
## 6  88      62   20   17       9    0.5        1
```

```r
#Observamos los atributos y el número de instancias y de características del dataset
str(bupa)
```

```
## 'data.frame':    345 obs. of  7 variables:
##  $ Mcv     : num  85 85 86 91 98 88 88 92 90 89 ...
##  $ Alkphos : num  92 64 54 78 55 62 67 54 60 52 ...
##  $ Sgpt    : num  45 59 33 34 13 20 21 22 25 13 ...
##  $ Sgot    : num  27 32 16 24 17 17 11 20 19 24 ...
##  $ Gammagt : num  31 23 54 36 17 9 11 7 5 15 ...
##  $ Drinks  : num  0 0 0 0 0 0.5 0.5 0.5 0.5 0.5 ...
##  $ Selector: int  1 2 2 2 2 1 1 1 1 1 ...
```

```r
dim(bupa)
```

```
## [1] 345   7
```

```r
#Comprobamos los tipos de datos atómicos
clase_variables <- sapply(bupa, class)
clase_variables
```

```
##       Mcv   Alkphos      Sgpt      Sgot   Gammagt    Drinks  Selector
## "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "integer"
```

```r
#Podemos pasar a enteras varias variables de este dataset, además hace falta pasar
#a categórica la variable dependiente
bupa_proc <- bupa
bupa_proc[,1:5] <- sapply(bupa_proc[,1:5], as.integer)
bupa_proc$Selector <- factor(bupa_proc$Selector, levels = c(1, 2), labels = c(1, 0))
```

```r
#Generamos un resumen estadístico de las distintas variables
#Aprovechando esto, utilizamos la función skim del paquete skimr
#para ver si hay missing values y obtener más información
```

```r
summary(bupa_proc)
```

```
##       Mcv            Alkphos           Sgpt            Sgot
##  Min.   : 65.00   Min.   : 23.00   Min.   :  4.00   Min.   : 5.00
##  1st Qu.: 87.00   1st Qu.: 57.00   1st Qu.: 19.00   1st Qu.:19.00
##  Median : 90.00   Median : 67.00   Median : 26.00   Median :23.00
##  Mean   : 90.16   Mean   : 69.87   Mean   : 30.41   Mean   :24.64
##  3rd Qu.: 93.00   3rd Qu.: 80.00   3rd Qu.: 34.00   3rd Qu.:27.00
##  Max.   :103.00   Max.   :138.00   Max.   :155.00   Max.   :82.00
##     Gammagt          Drinks       Selector
##  Min.   :  5.00   Min.   : 0.000   1:145
##  1st Qu.: 15.00   1st Qu.: 0.500   0:200
##  Median : 25.00   Median : 3.000
##  Mean   : 38.28   Mean   : 3.455
##  3rd Qu.: 46.00   3rd Qu.: 6.000
##  Max.   :297.00   Max.   :20.000
```

```r
skim(bupa_proc)
```

Table 1: Data summary

Name	bupa_proc
Number of rows	345
Number of columns	7
Column type frequency:	
factor	1
numeric	6
Group variables	None

Variable type: factor

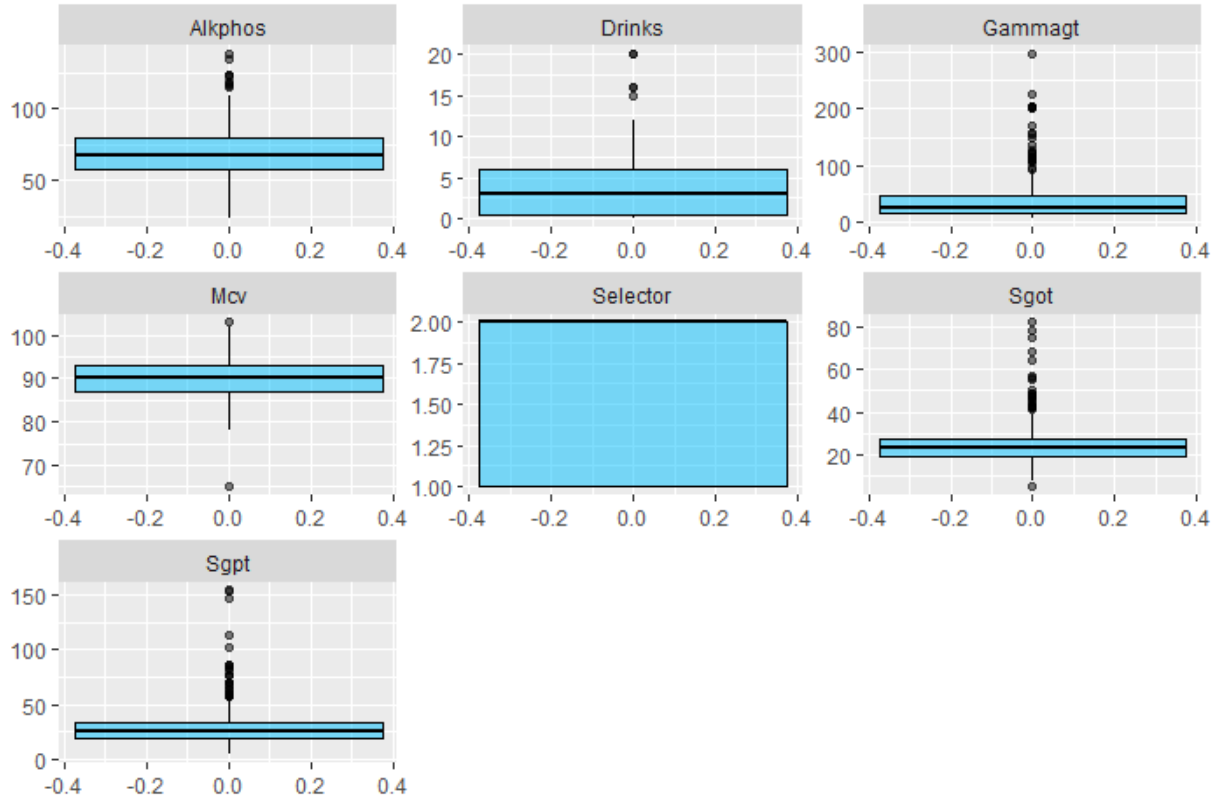
skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
Selector	0	1	FALSE	2	0: 200, 1: 145

Variable type: numeric

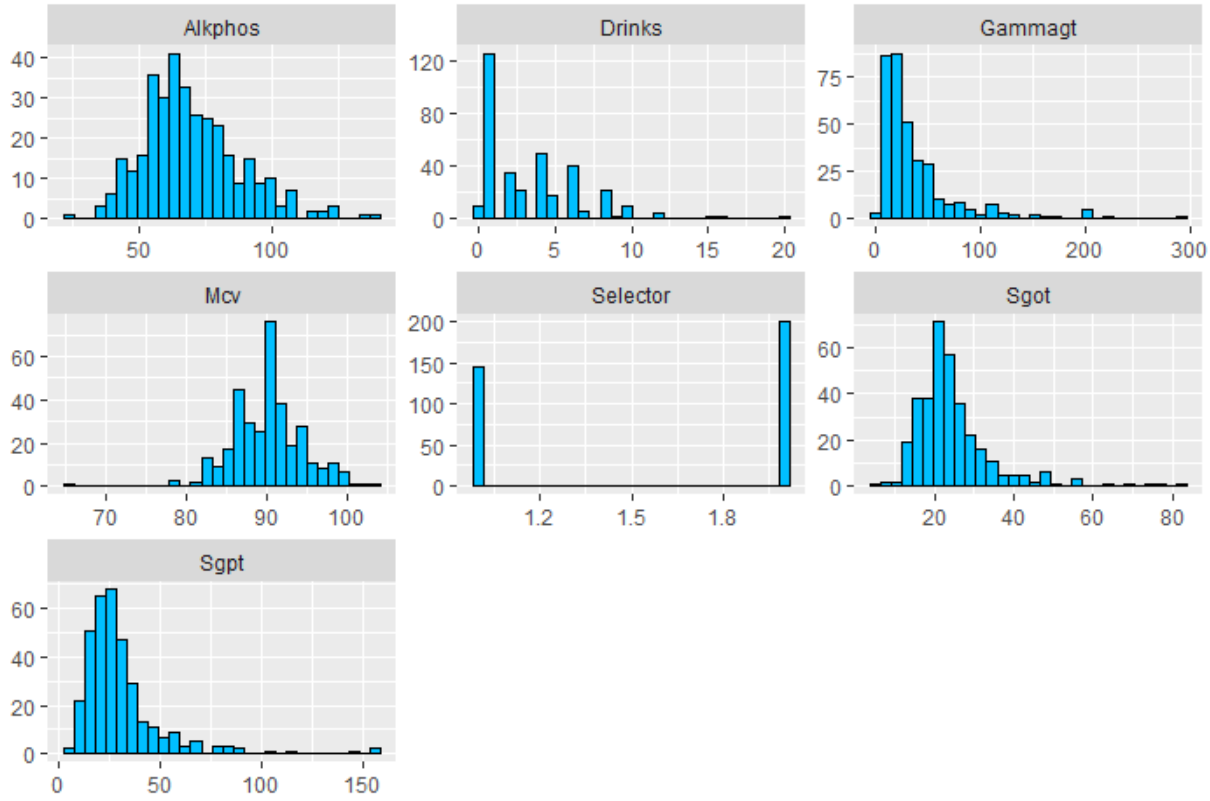
skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Mcv	0	1	90.16	4.45	65	87.0	90	93	103	
Alkphos	0	1	69.87	18.35	23	57.0	67	80	138	
Sgpt	0	1	30.41	19.51	4	19.0	26	34	155	
Sgot	0	1	24.64	10.06	5	19.0	23	27	82	
Gammagt	0	1	38.28	39.25	5	15.0	25	46	297	
Drinks	0	1	3.46	3.34	0	0.5	3	6	20	

```r
#-------------------------------------------------------------------------

#Outliers
```

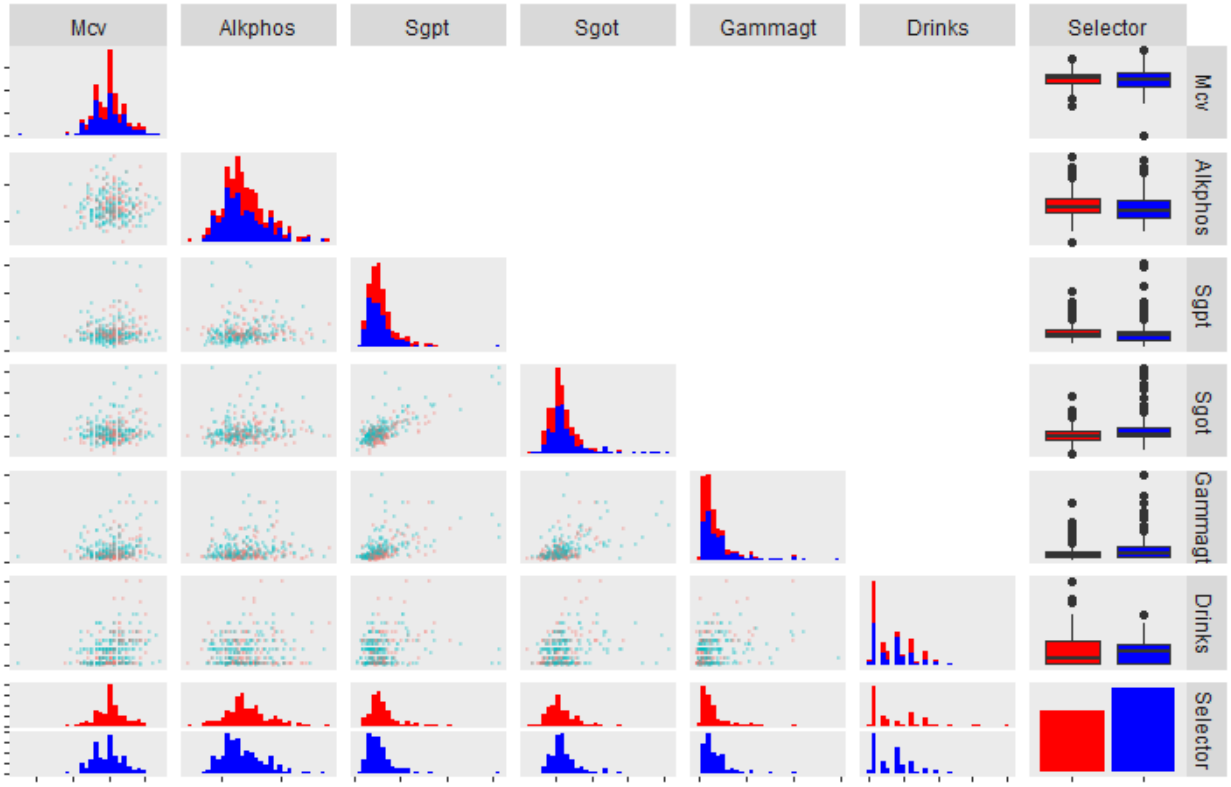
```r
bupa %>% gather(key = "variable", value = "valor") %>%
  ggplot(aes(y = valor)) +
  geom_boxplot(fill = "deepskyblue", color = "black", alpha = 0.5) +
  labs(
    x = "",
    y = ""
  ) +
  facet_wrap(~ variable, scales = "free")
```



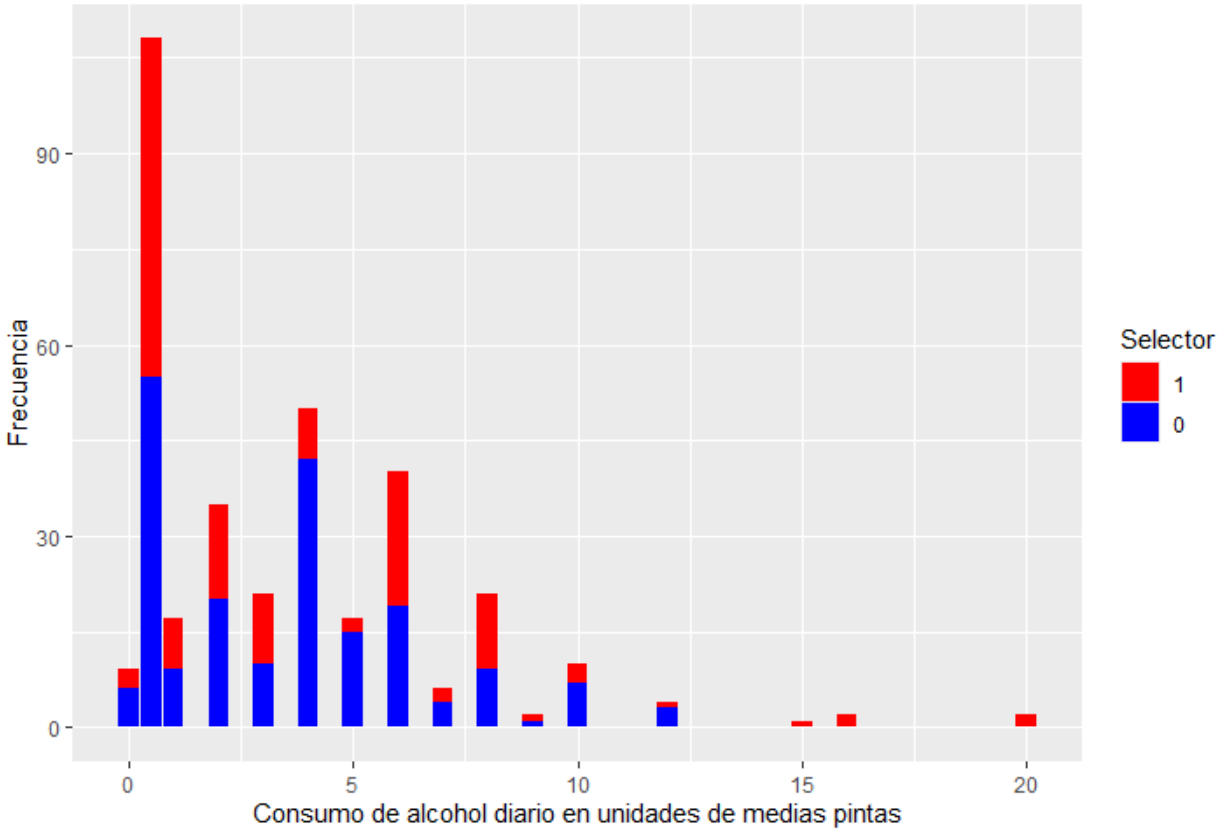

```r
#-------------------------------------------------------------------------------

#Plots univariables

bupa %>% gather(key = "variable", value = "valor") %>%
  ggplot(aes(x = valor, fill = Selector)) +
  geom_histogram(bins = 30, fill = "deepskyblue", color = "black") +
  labs(
    x = "",
    y = ""
  ) +
  facet_wrap(~ variable, scales = "free")
```



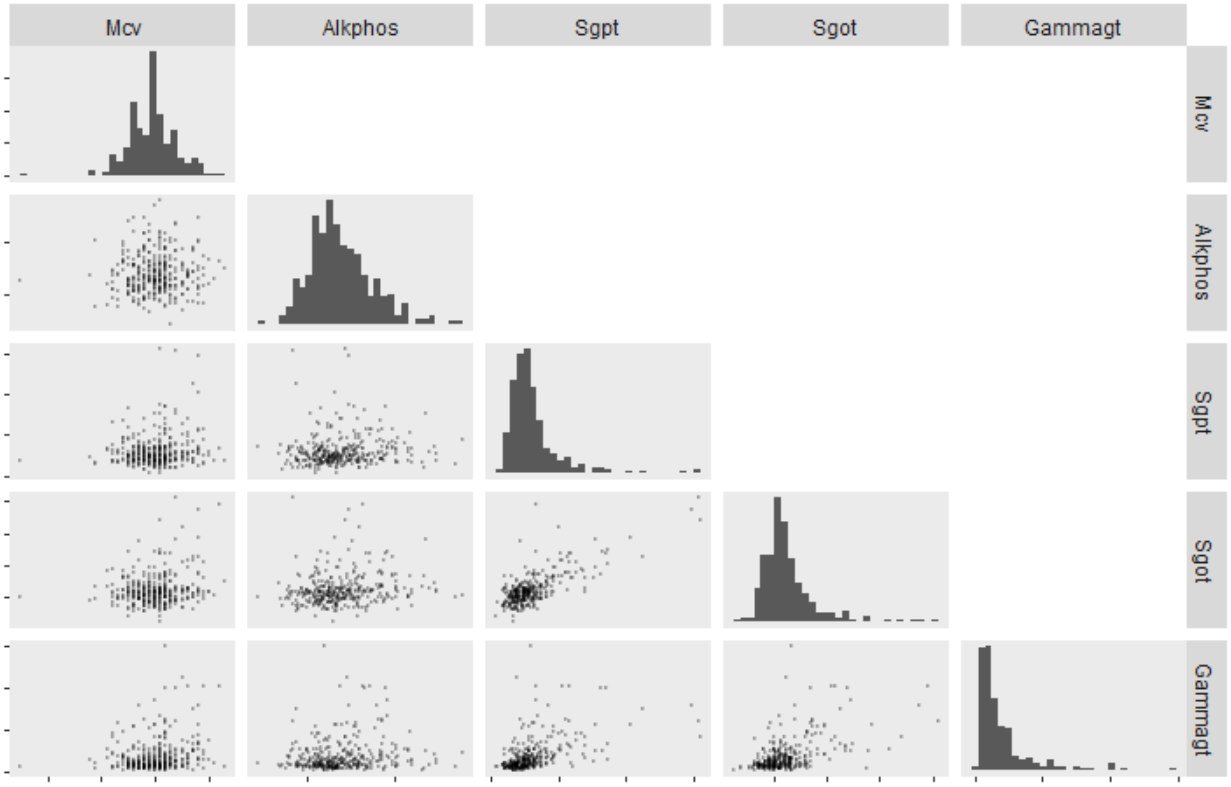

```
#-------------------------------------------------------------------------------

#Plots bivariables

ggpairs(bupa_proc, mapping = aes(color = Selector),
        lower = list(continuous = wrap("points", size = 0.07, alpha = 0.2)),
        diag = list(continuous = wrap("barDiag")),
        upper = list(continuous = "blank")) +
  scale_fill_manual(values = c("0" = "blue", "1" = "red")) +
  theme(axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        panel.grid = element_blank()) +
  ggtitle("")
```



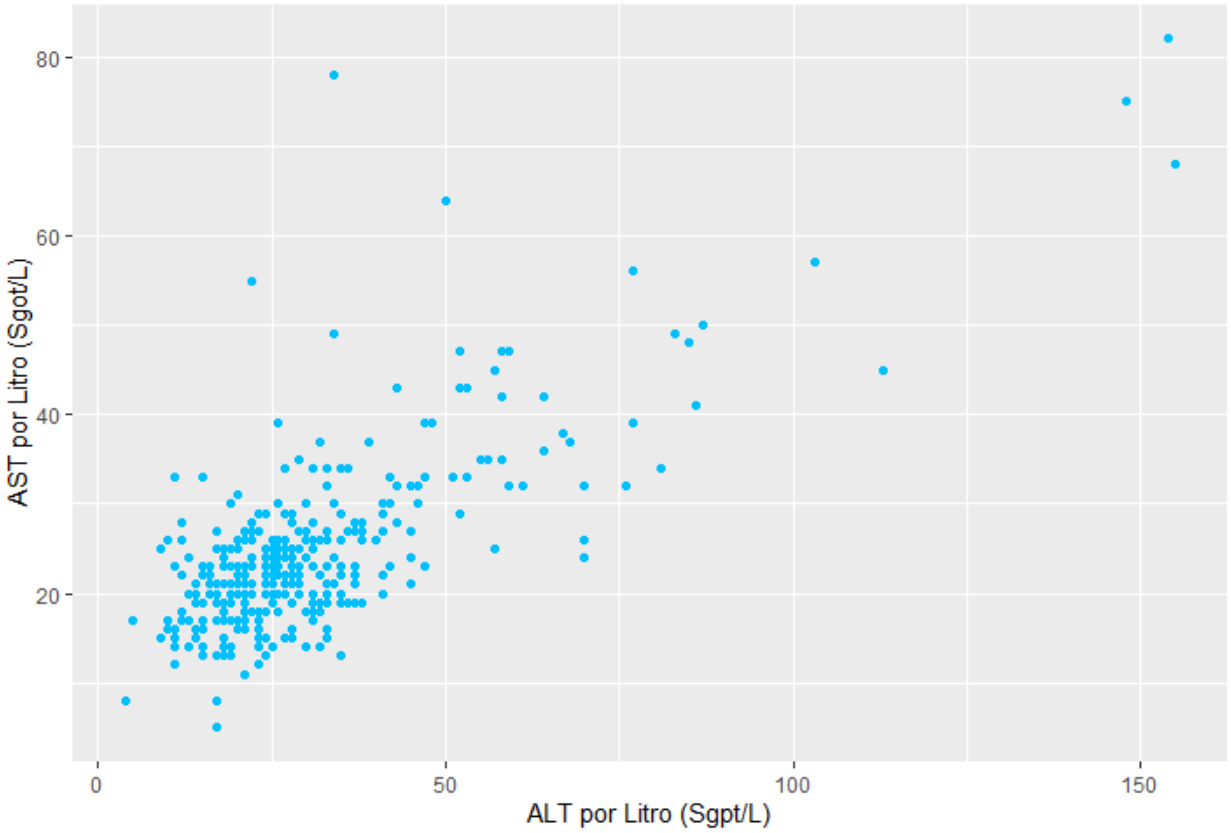

```
#-------------------------------------------------------------------------------
#Plots para verificar hipótesis

#H1
bupa_proc %>% ggplot(aes(x = Drinks)) +
  geom_bar(aes(fill = Selector)) +
  scale_fill_manual(values = c("0" = "blue", "1" = "red")) +
  labs(
    x = "Consumo de alcohol diario en unidades de medias pintas",
    y = "Frecuencia"
  )
```



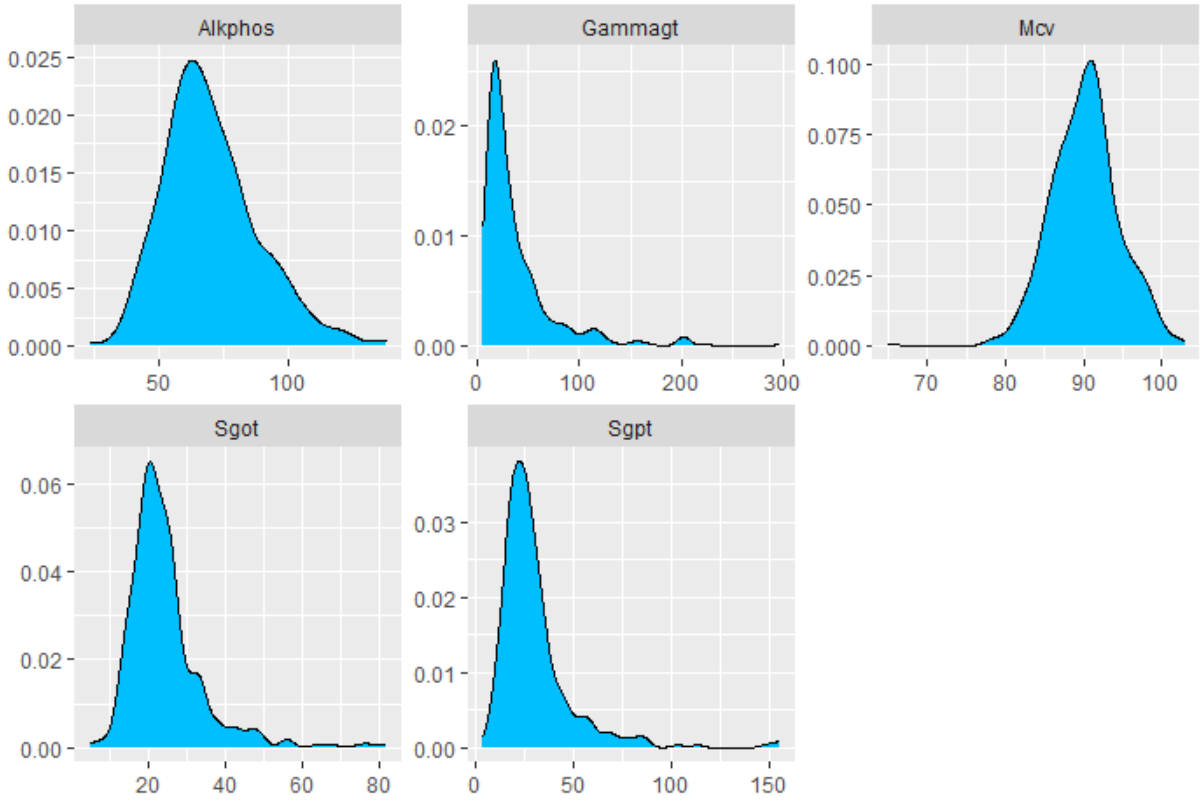

```
#H2
ggpairs(bupa_proc[, 1:5],
        lower = list(continuous = wrap("points", size = 0.07, alpha = 0.2)),
        diag = list(continuous = wrap("barDiag")),
        upper = list(continuous = "blank")) +
  theme(axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        panel.grid = element_blank()) +
  ggtitle("")
```



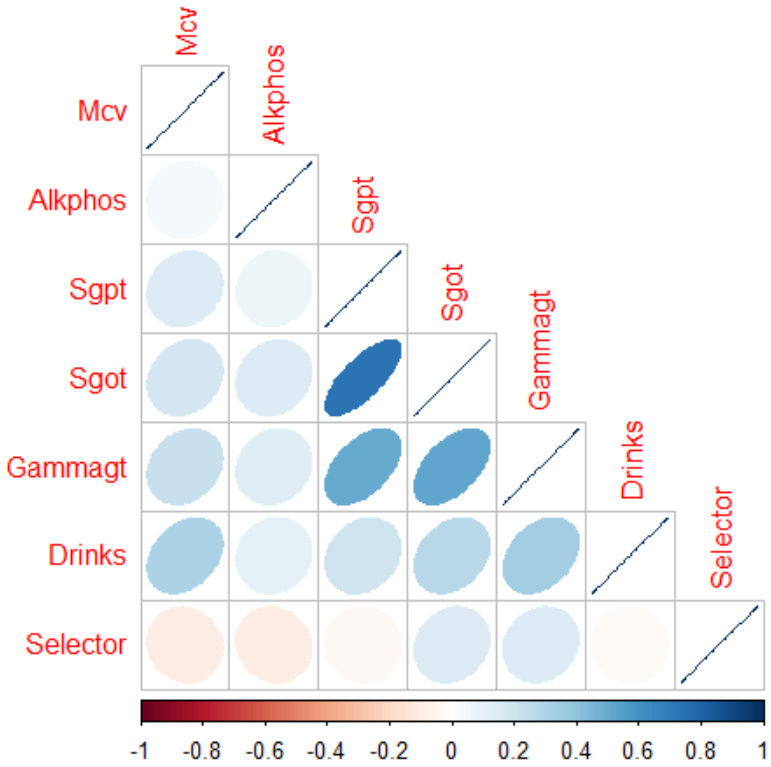

```
bupa_proc %>% ggplot(aes(x = Sgpt, y = Sgot)) +
  geom_point(color = "deepskyblue") +
  labs(
    x = "ALT por Litro (Sgpt/L)",
    y = "AST por Litro (Sgot/L)"
  )
```




```
#H3
bupa_bio <- bupa_proc[,1:5]
bupa_bio %>% gather(key = "variable", value = "valor") %>%
  ggplot(aes(x = valor, fill = Selector)) +
  geom_density(fill = "deepskyblue", color = "black") +
  labs(
    x = "",
    y = ""
  ) +
  facet_wrap(~ variable, scales = "free")
```




```
#-------------------------------------------------------------------------------
#Correlación

y <- bupa %>%
  cor()

corrplot(y,, type ="lower", method = "ellipse")
```



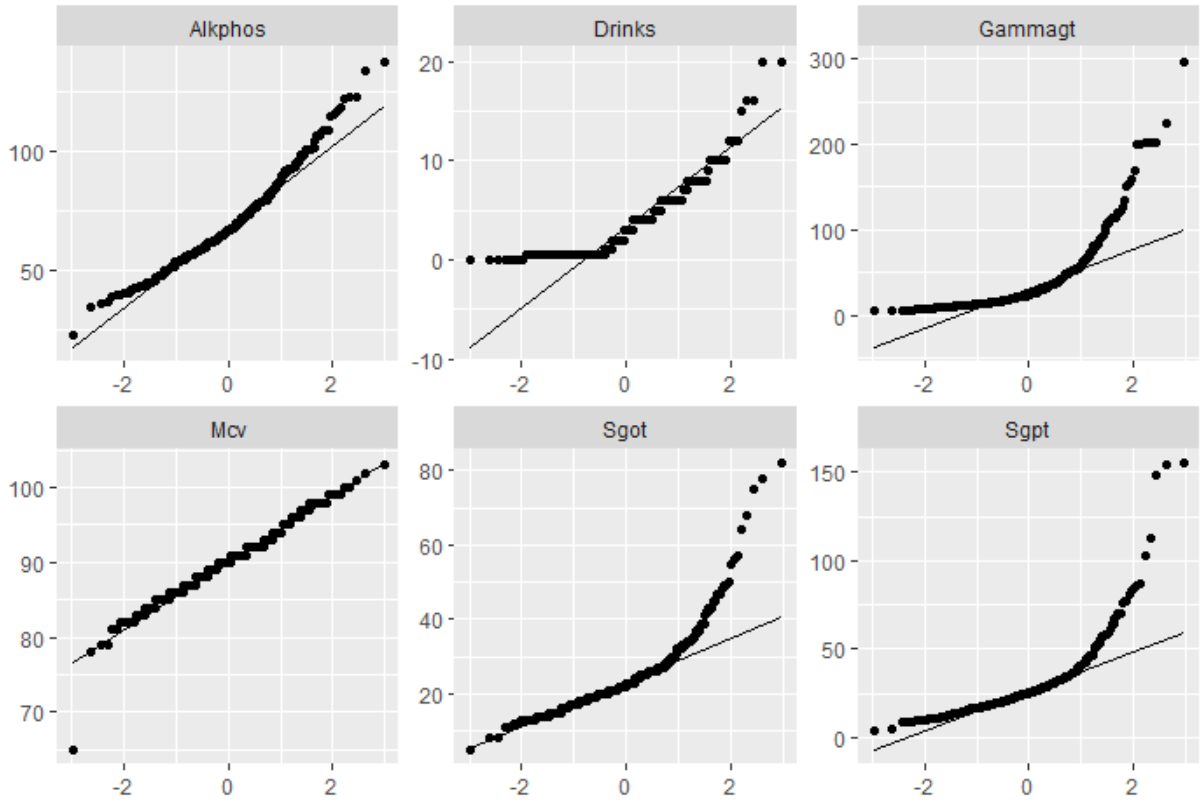

```
#Reducción de dimensionalidad

#Recuento de categoría Selector cuando AST/ALT es menor y mayor a 1

AST_ALT_menor1 <- bupa_proc %>%
  mutate(AST_ALT = (Sgot/Sgpt)) %>% group_by(Selector) %>% filter(AST_ALT<1) %>%
  summarise(N_menor = n())

AST_ALT_mayor1 <- bupa_proc %>%
  mutate(AST_ALT = (Sgot/Sgpt)) %>% group_by(Selector) %>% filter(AST_ALT>1) %>%
  summarise(N_mayor = n())

AST_ALT_menor1
```

```
## # A tibble: 2 x 2
##   Selector N_menor
##   <fct>      <int>
## 1 1            113
## 2 0            105
```

```
AST_ALT_mayor1
```

```
## # A tibble: 2 x 2
##   Selector N_mayor
##   <fct>      <int>
## 1 1             25
## 2 0             86
```

```r
#Creación nueva variable categórica AST_ALT
bupa_new <- bupa_proc %>%
  mutate(AST_ALT = factor(ifelse(Sgot/Sgpt < 1, 1, 0))) %>%
  dplyr::select(-Sgpt, -Sgot)

#Modificación variable Drinks
bupa_new2 <- bupa_new %>%
  mutate(Drinks = factor(ifelse(Drinks >= 15, 1, 0)))
#-------------------------------------------------------------------------------

#Normalidad

bupa[,1:6] %>% gather(key = "variable", value = "valor") %>%
  ggplot(aes(sample = valor)) +
  stat_qq() +
  stat_qq_line()+
  labs(
    x = "",
    y = ""
  ) +
  facet_wrap(~ variable, scales = "free")
```

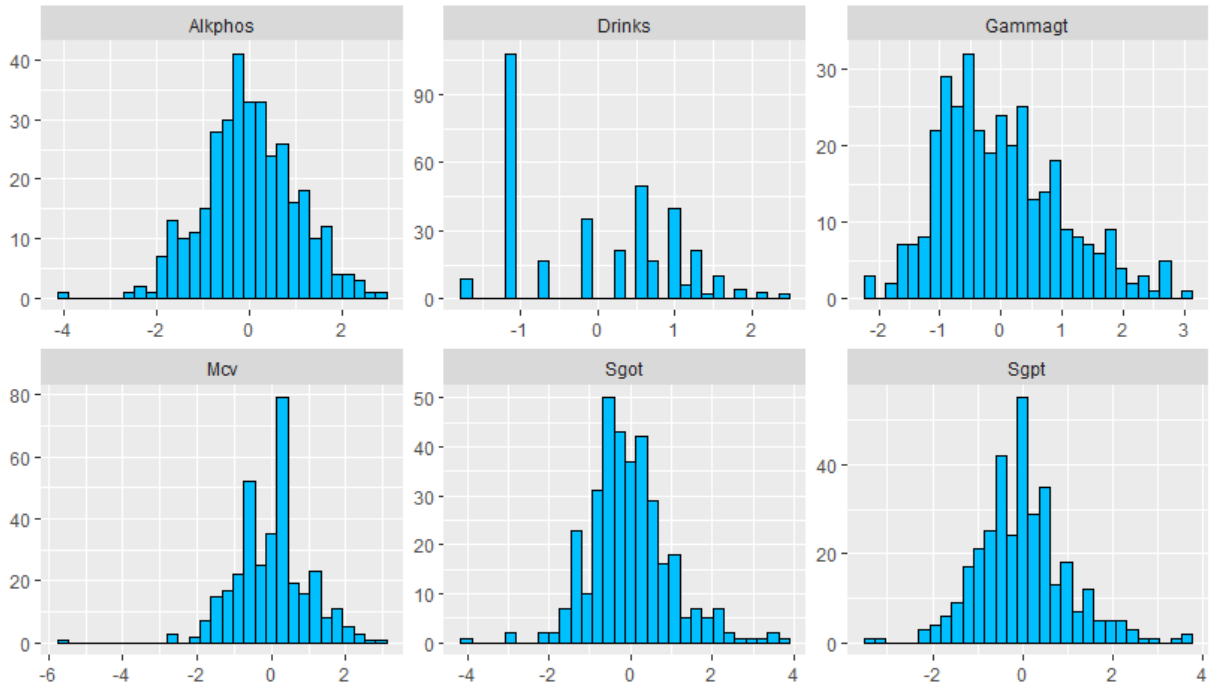



```r
shapiro_test <- function(x) {
  shapiro.test(x)$p.value
}
```

```r
shapiro_results <- apply(bupa[,1:6], 2, shapiro_test)

print(shapiro_results)
```

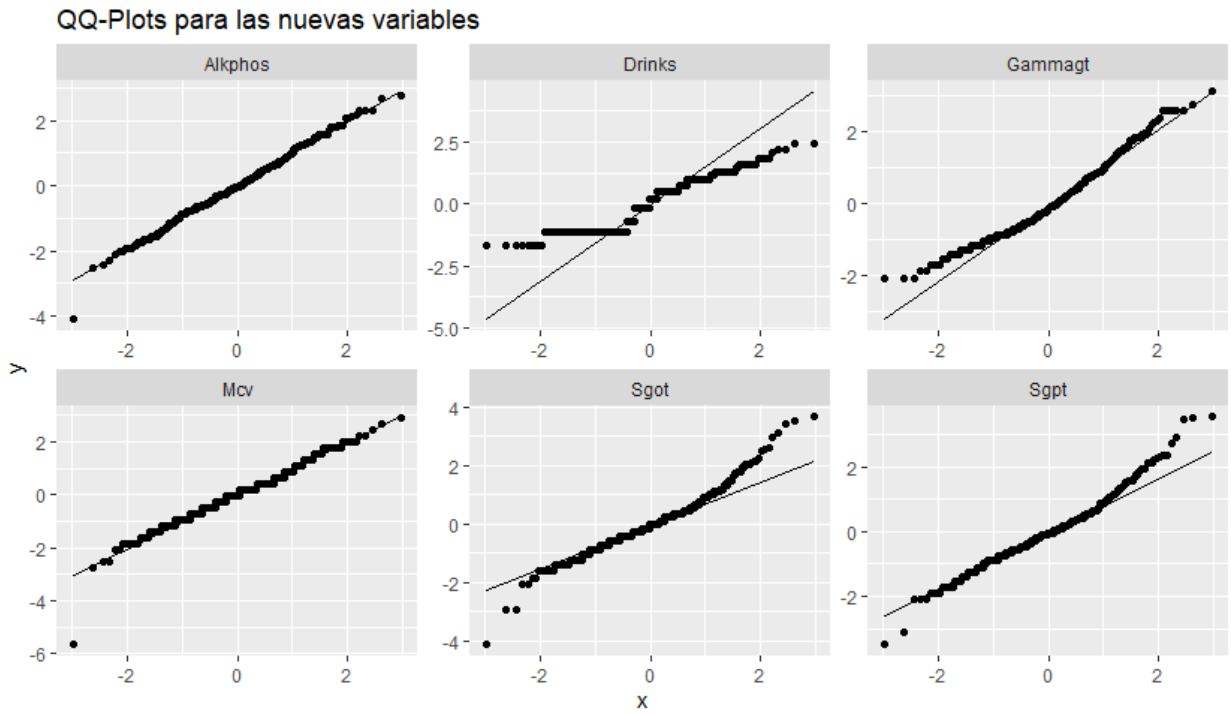
```
##          Mcv      Alkphos         Sgpt         Sgot      Gammagt       Drinks
## 3.340830e-06 3.604551e-07 2.579879e-23 1.402884e-19 6.480735e-25 1.686482e-18
```

```r
#-------------------------------------------------------------------------------

#Transformaciones

bupa_log_transformed <- bupa_proc %>%
  mutate_at(c("Alkphos", "Gammagt", "Sgpt", "Sgot", "Drinks"), log1p) %>%
  mutate_at(c("Mcv", "Alkphos", "Gammagt", "Sgpt", "Sgot", "Drinks"), scale)

bupa_log_transformed[,1:6] %>% gather(key = "variable", value = "valor") %>%
  ggplot(aes(x = valor)) +
  geom_histogram(bins = 30, fill = "deepskyblue", color = "black") +
  labs(
    x = "",
    y = ""
  ) +
  facet_wrap(~ variable, scales = "free")
```




```r
#-------------------------------------------------------------------------------

#Normalidad de las transformaciones (qqplot y shapiro)

bupa_log_transformed[,1:6] %>% gather(key = "variable", value = "valor") %>%
  ggplot(aes(sample = valor)) +
  stat_qq() +
```

```
  stat_qq_line()+
  labs(
    title = "QQ-Plots para las nuevas variables",
  ) +
  facet_wrap(~ variable, scales = "free")
```




```
# Verificar la normalidad de las nuevas variables transformadas

shapiro_log <- bupa_log_transformed[,1:6] %>%
  summarise_all(shapiro_test)

shapiro_log
```

```
##           Mcv    Alkphos          Sgpt          Sgot      Gammagt       Drinks
## 1 3.34083e-06 0.2542805 1.529513e-05 2.752713e-07 3.762563e-06 8.657008e-14
```

```
#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------

#Clasificación


#-------------------------------------------------------------------------------
#C-1: Utilizar el algoritmo k-NN probando con diferentes valores de k.
#Elegir el que considere más adecuado para su conjunto de datos.
#Analice qué ocurre en los valores de precisión en training y test con los
#diferentes valores de k.

library(class)
```

```
## Warning: package 'class' was built under R version 4.3.3
```

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

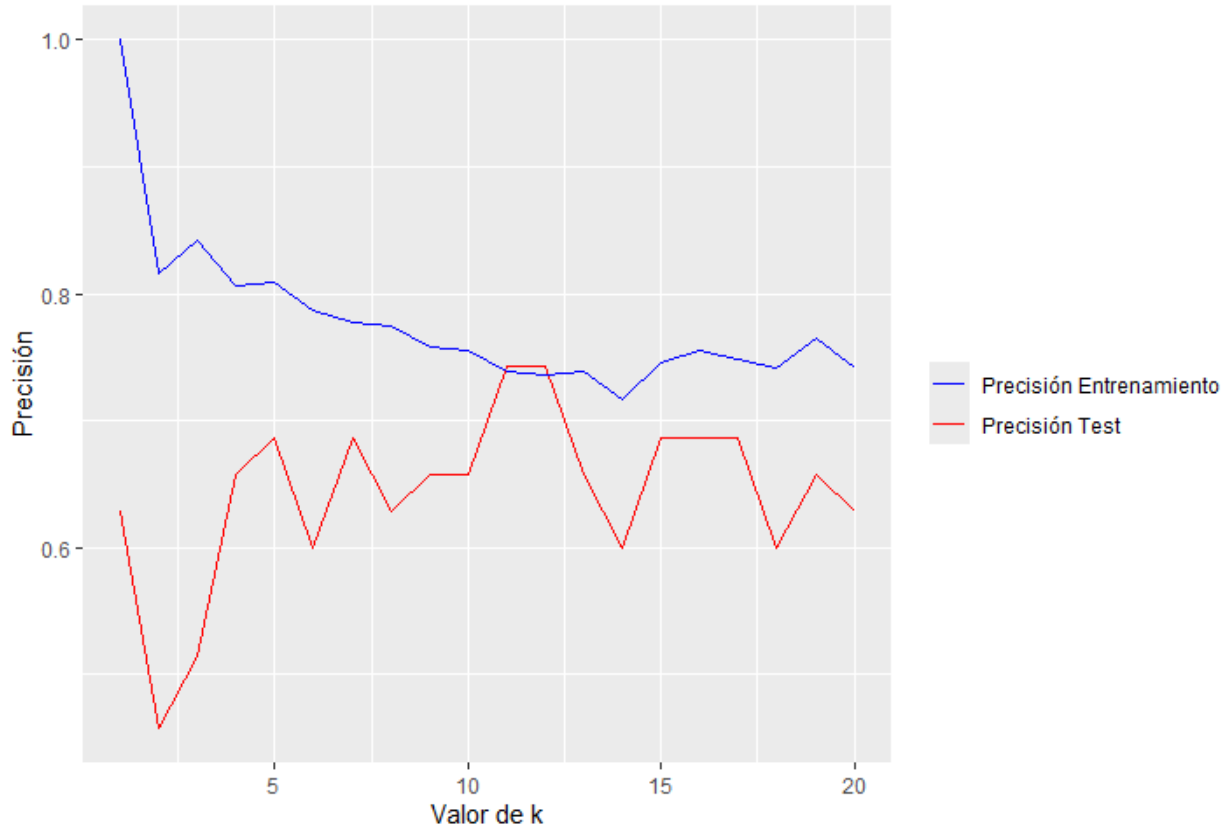
```r
library(biotools)
```

```
## Warning: package 'biotools' was built under R version 4.3.3
```

```
## ---
## biotools version 4.2
```

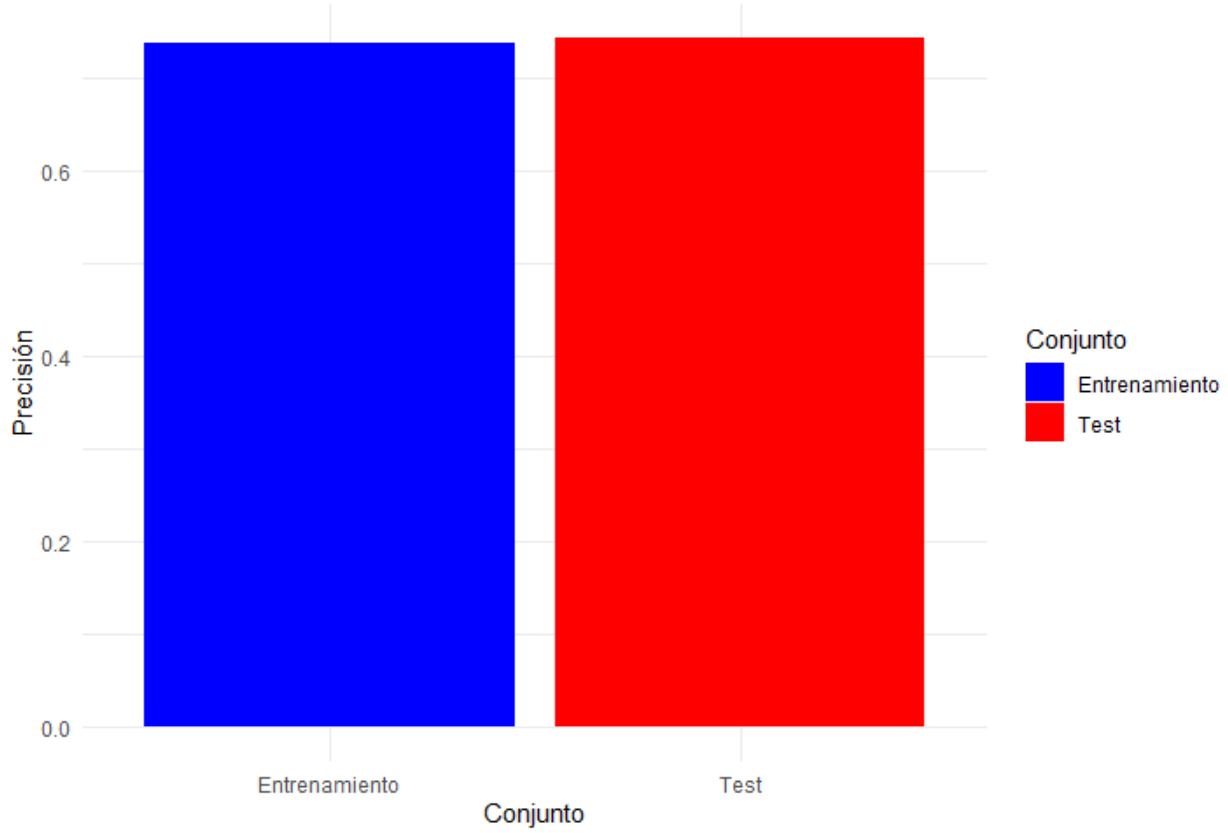
```r
#función para aplicar k-NN a los 10-folds de datos
knn_fold <- function(i, x, k_value) {
  file <- paste(x, "-10-", i, "tra.dat", sep="")
  x_tra <- read.csv(file, comment.char="@" , header=FALSE )
  file <- paste(x, "-10-", i, "tst.dat", sep="")
  x_tst <- read.csv(file, comment.char="@" , header=FALSE )
  names(x_tra) <- c("Mcv", "Alkphos", "Sgpt",
                    "Sgot", "Gammagt", "Drinks", "Selector")
  names(x_tst) <- c("Mcv", "Alkphos", "Sgpt",
                    "Sgot", "Gammagt", "Drinks", "Selector")
  #Preprocesamiento
  x_tra_log_transformed <- x_tra %>%
    mutate(Selector = factor(Selector, levels = c(1, 2), labels = c(1, 0))) %>%
    mutate_at(c("Alkphos", "Gammagt", "Sgpt", "Sgot", "Drinks"), log1p) %>%
    mutate_at(c("Mcv", "Alkphos", "Gammagt", "Sgpt", "Sgot", "Drinks"), scale)

  x_tst_log_transformed <- x_tst %>%
    mutate(Selector = factor(Selector, levels = c(1, 2), labels = c(1, 0))) %>%
    mutate_at(c("Alkphos", "Gammagt", "Sgpt", "Sgot", "Drinks"), log1p) %>%
    mutate_at(c("Mcv", "Alkphos", "Gammagt", "Sgpt", "Sgot", "Drinks"), scale)

  #Predicción
  knn.pred.tst <- knn(x_tra_log_transformed[,-7], x_tst_log_transformed[,-7], x_tra_log_transformed$Sel
  knn.pred.tra <- knn(x_tra_log_transformed[,-7], x_tra_log_transformed[,-7], x_tra_log_transformed$Sel

  #Medidas de calidad
  t_tst <- table(knn.pred.tst, x_tst_log_transformed$Selector)
  val_acc_rate_tst <- sum(diag(t_tst)) / nrow(x_tst_log_transformed)
  t_tra <- table(knn.pred.tra, x_tra_log_transformed$Selector)
  val_acc_rate_tra <- sum(diag(t_tra)) / nrow(x_tra_log_transformed)

  c(val_acc_rate_tst, val_acc_rate_tra)
}

#Valores de k a probar

k_valores <- 1:20

#Aplicar función a cada valor de k distinto
```

```r
nombre <- "bupa"
resultados <- mapply(function(k) {
  r_tst <- mean(sapply(1:10, knn_fold, nombre, k)[1])
  r_tra <- mean(sapply(1:10, knn_fold, nombre, k)[2])
  c(r_tst, r_tra)
}, k_valores)

#Pasar vector de resultados a dataframe
resultados_knn_df <- data.frame(
  k = k_valores,
  acc_test = resultados[1, ],
  acc_train = resultados[2, ]
)

#Gráfico para visualizar rendimiento
ggplot(resultados_knn_df, aes(x = k)) +
  geom_line(aes(y = acc_test, color = "Precisión Test")) +
  geom_line(aes(y = acc_train, color = "Precisión Entrenamiento")) +
  labs(x = "Valor de k", y = "Precisión") +
  scale_color_manual("", values = c("Precisión Test" = "red", "Precisión Entrenamiento" = "blue"))
```

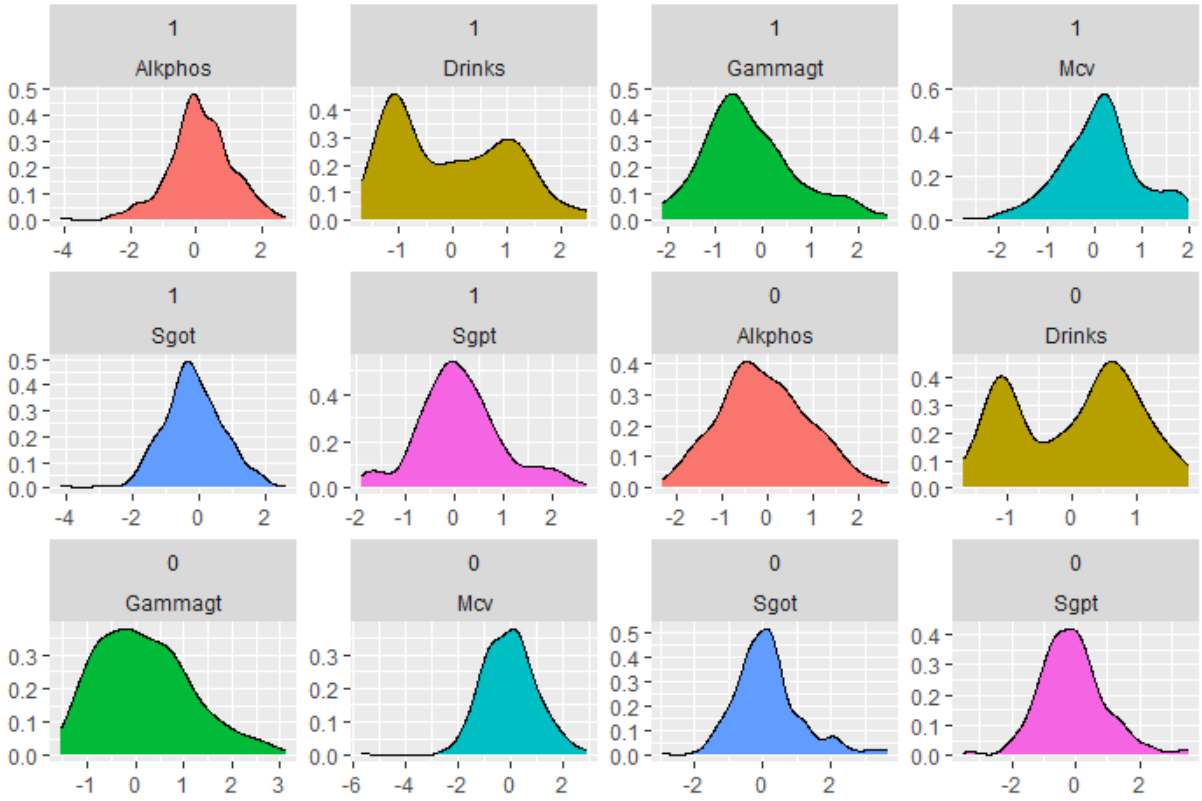



```r
#Para el conjunto de datos, el valor de k que proporciona mejor rendimiento sería 11 o 12.
knn_fold_best <- function(i, x) {
  file <- paste(x, "-10-", i, "tra.dat", sep="")
  x_tra <- read.csv(file, comment.char="@" , header=FALSE )
  file <- paste(x, "-10-", i, "tst.dat", sep="")
  x_tst <- read.csv(file, comment.char="@" , header=FALSE )
```

```r
  names(x_tra) <- c("Mcv", "Alkphos", "Sgpt",
                    "Sgot", "Gammagt", "Drinks", "Selector")
  names(x_tst) <- c("Mcv", "Alkphos", "Sgpt",
                    "Sgot", "Gammagt", "Drinks", "Selector")
  #Preprocesamiento
  x_tra_log_transformed <- x_tra %>%
    mutate(Selector = factor(Selector, levels = c(1, 2), labels = c(1, 0))) %>%
    mutate_at(c("Alkphos", "Gammagt", "Sgpt", "Sgot", "Drinks"), log1p) %>%
    mutate_at(c("Mcv", "Alkphos", "Gammagt", "Sgpt", "Sgot", "Drinks"), scale)

  x_tst_log_transformed <- x_tst %>%
    mutate(Selector = factor(Selector, levels = c(1, 2), labels = c(1, 0))) %>%
    mutate_at(c("Alkphos", "Gammagt", "Sgpt", "Sgot", "Drinks"), log1p) %>%
    mutate_at(c("Mcv", "Alkphos", "Gammagt", "Sgpt", "Sgot", "Drinks"), scale)

  #Predicción
  knn.pred.tst <- knn(x_tra_log_transformed[,-7], x_tst_log_transformed[,-7], x_tra_log_transformed$Sel
  knn.pred.tra <- knn(x_tra_log_transformed[,-7], x_tra_log_transformed[,-7], x_tra_log_transformed$Sel

  #Medidas de calidad
  t_tst <- table(knn.pred.tst, x_tst_log_transformed$Selector)
  val_acc_rate_tst <- sum(diag(t_tst)) / nrow(x_tst_log_transformed)
  t_tra <- table(knn.pred.tra, x_tra_log_transformed$Selector)
  val_acc_rate_tra <- sum(diag(t_tra)) / nrow(x_tra_log_transformed)

  c(val_acc_rate_tst, val_acc_rate_tra)
}

r_knn_tst <- mean(sapply(1:10, knn_fold_best, nombre)[1])
r_knn_tra <- mean(sapply(1:10, knn_fold_best, nombre)[2])

resultados_knn_best_df <- data.frame(
  Conjunto = c("Test", "Entrenamiento"),
  Precision = c(r_knn_tst, r_knn_tra)
)

#Gráfico para visualizar rendimiento
ggplot(resultados_knn_best_df, aes(x = Conjunto, y = Precision, fill = Conjunto)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Conjunto", y = "Precisión") +
  scale_fill_manual(values = c("Test" = "red", "Entrenamiento" = "blue"))
```



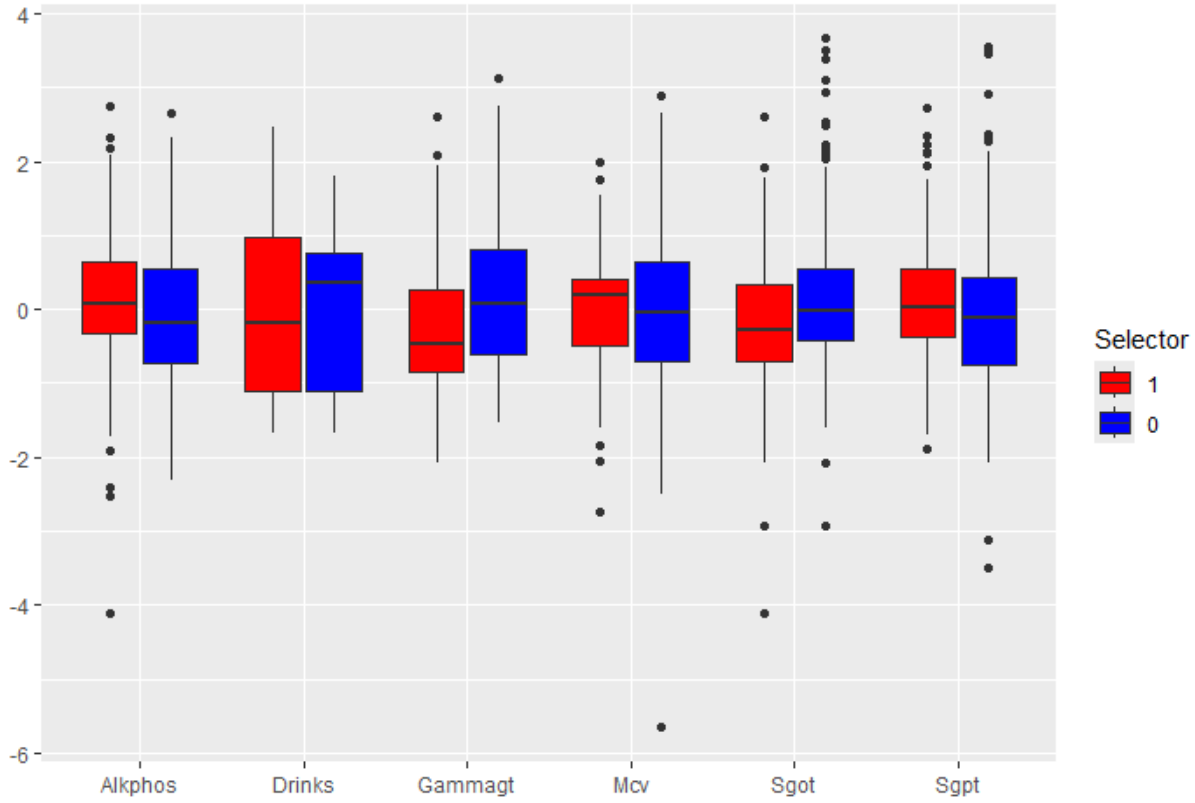

```
#-------------------------------------------------------------------------------
#C-2: Utilizar el algoritmo LDA para clasificar. No olvide comprobar las asunciones.

#Comprobamos las asunciones de partida, para ello se mostrarán gráficos y se realizarán
#los tests pertinentes.

#Normalidad en las variables predictoras (tras las transformaciones, log + scale)
#(Ver parte de EDA para mas información)

#Gráfico
plot_data_C2 <- bupa_log_transformed %>% pivot_longer(-Selector)

ggplot(plot_data_C2) +
  geom_density(aes(x = value, fill = name)) +
  labs(
    x = "",
    y = ""
  ) +
  facet_wrap(~ Selector + name, scales = "free")+
  theme(legend.position = "none")
```




```
#Test de Shapiro-Wilk's
my_shapiro <- function(a) {
  shapiro.test(a)$statistic
}

bupa_log_transformed %>%
  group_by(Selector) %>%
  summarise(across(Mcv:Drinks, my_shapiro))
```

```
## # A tibble: 2 x 7
##   Selector   Mcv Alkphos  Sgpt  Sgot Gammagt Drinks
##   <fct>    <dbl>   <dbl> <dbl> <dbl>   <dbl>  <dbl>
## 1 1        0.978   0.976 0.975 0.977   0.960  0.881
## 2 0        0.966   0.991 0.968 0.949   0.968  0.906
```

```
#Homogeneidad de las matrices de varianza y covarianza

#Gráfico
ggplot(plot_data_C2) +
  geom_boxplot(aes(x=name, y=value, fill=Selector)) +
  labs(
    x = "",
    y = ""
  )+
  scale_fill_manual(values = c("1" = "red", "0" = "blue"))
```




```
#Test de Bartlett
my_bartlett <- function(variable, clase) {
  bartlett.test(variable ~ clase)$p.value
}

bupa_log_transformed %>%
  summarise(across(Mcv:Drinks, ~ my_bartlett(.x, bupa_log_transformed$Selector)))
```

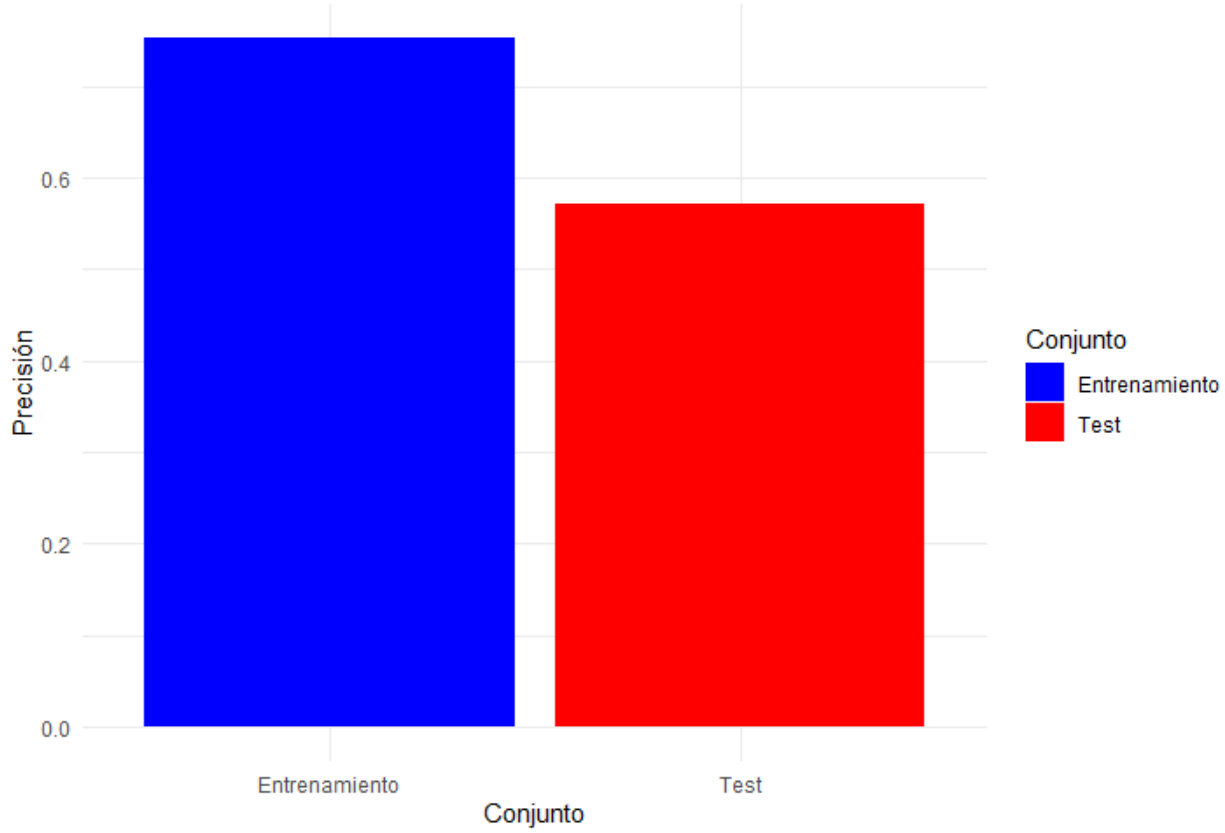
```
##           Mcv   Alkphos        Sgpt      Sgot   Gammagt    Drinks
## 1 0.005894088 0.6140872 0.006819117 0.3807296 0.8545676 0.1202088
```

```
#Test boxM

boxM(bupa_log_transformed[, 1:6], bupa_log_transformed$Selector)
```

```
##
##  Box's M-test for Homogeneity of Covariance Matrices
##
## data:  bupa_log_transformed[, 1:6]
## Chi-Sq (approx.) = 42.203, df = 21, p-value = 0.003967
```

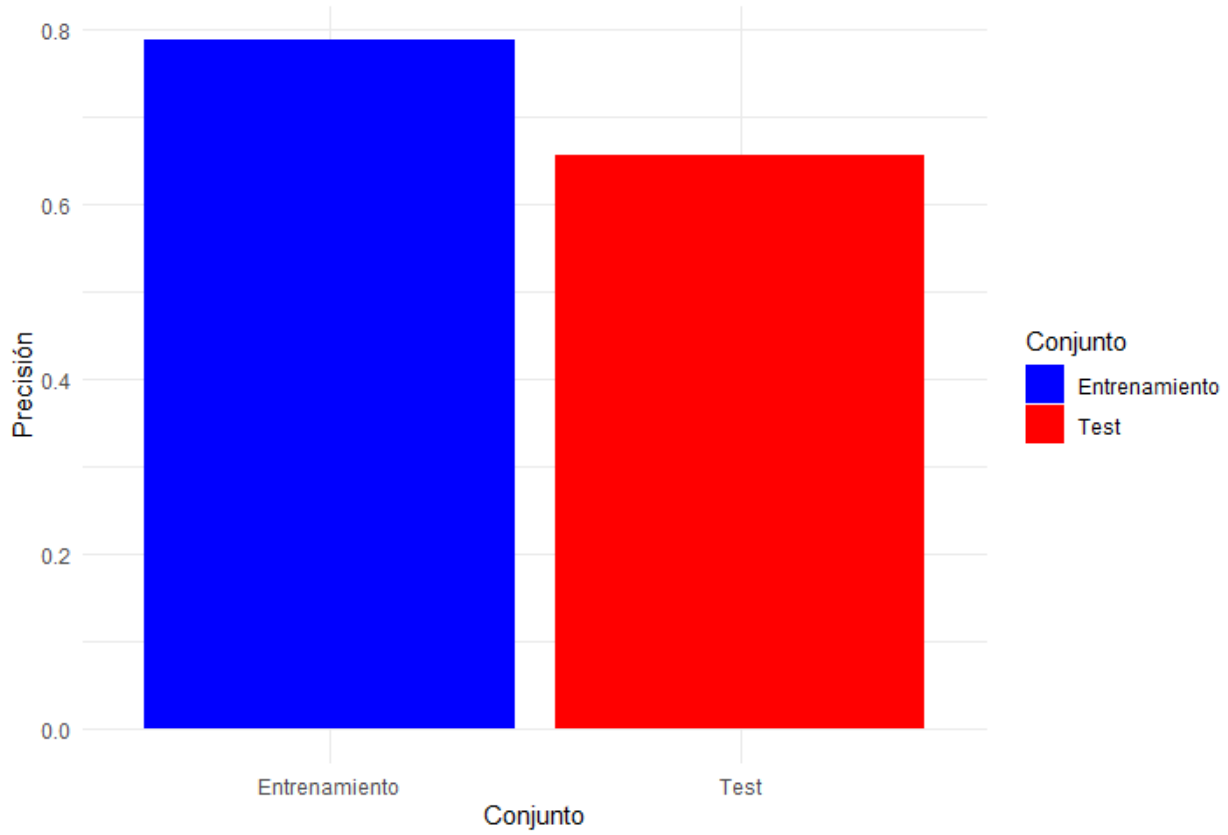
```
#Aplicar LDA
lda_fold <- function(i, x) {
  file <- paste(x, "-10-", i, "tra.dat", sep="")
  x_tra <- read.csv(file, comment.char="@" , header=FALSE )
  file <- paste(x, "-10-", i, "tst.dat", sep="")
  x_tst <- read.csv(file, comment.char="@" , header=FALSE )
  names(x_tra) <- c("Mcv", "Alkphos", "Sgpt",
```

```
                    "Sgot", "Gammagt", "Drinks", "Selector")
  names(x_tst) <- c("Mcv", "Alkphos", "Sgpt",
                    "Sgot", "Gammagt", "Drinks", "Selector")
  #Preprocesamiento
  x_tra_log_transformed <- x_tra %>%
    mutate(Selector = factor(Selector, levels = c(1, 2), labels = c(1, 0))) %>%
    mutate_at(c("Alkphos", "Gammagt", "Sgpt", "Sgot", "Drinks"), log1p) %>%
    mutate_at(c("Mcv", "Alkphos", "Gammagt", "Sgpt", "Sgot", "Drinks"), scale)

  x_tst_log_transformed <- x_tst %>%
    mutate(Selector = factor(Selector, levels = c(1, 2), labels = c(1, 0))) %>%
    mutate_at(c("Alkphos", "Gammagt", "Sgpt", "Sgot", "Drinks"), log1p) %>%
    mutate_at(c("Mcv", "Alkphos", "Gammagt", "Sgpt", "Sgot", "Drinks"), scale)

  #Predicción
  model.lda <- lda(Selector ~ ., data = x_tra_log_transformed)

  lda.pred.tra <- predict(model.lda, x_tra_log_transformed)
  lda.pred.tst <- predict(model.lda, x_tst_log_transformed)

  #Medidas de calidad

  t_tst <- table(lda.pred.tst$class, x_tst_log_transformed$Selector)
  val_acc_rate_tst <- sum(diag(t_tst)) / nrow(x_tst_log_transformed)
  t_tra <- table(lda.pred.tra$class, x_tra_log_transformed$Selector)
  val_acc_rate_tra <- sum(diag(t_tra)) / nrow(x_tra_log_transformed)

  c(val_acc_rate_tst, val_acc_rate_tra)
}

r_lda_tst <- mean(sapply(1:10, lda_fold, nombre)[1])
r_lda_tra <- mean(sapply(1:10, lda_fold, nombre)[2])

#Pasar vector de resultados a dataframe
resultados_lda_df <- data.frame(
  Conjunto = c("Test", "Entrenamiento"),
  Precision = c(r_lda_tst, r_lda_tra)
)

#Gráfico para visualizar rendimiento
ggplot(resultados_lda_df, aes(x = Conjunto, y = Precision, fill = Conjunto)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Conjunto", y = "Precisión") +
  scale_fill_manual(values = c("Test" = "red", "Entrenamiento" = "blue"))
```

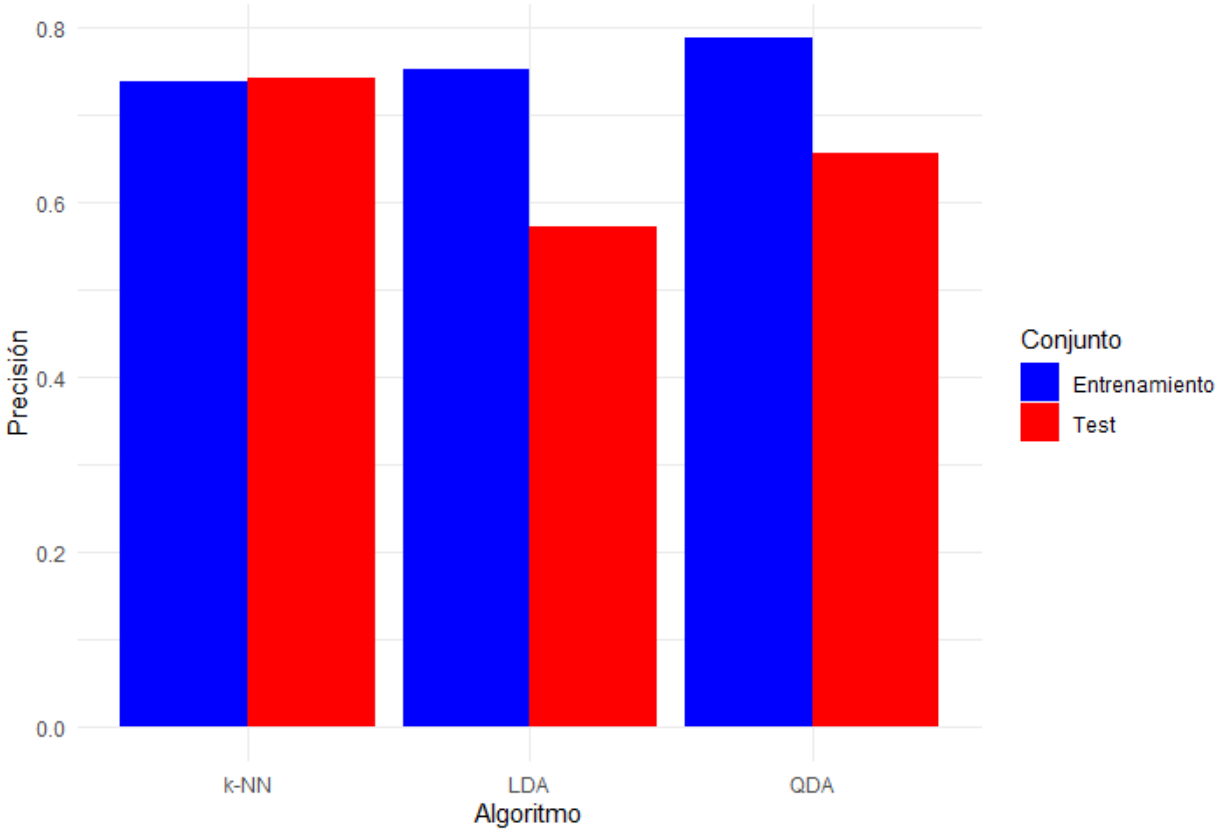



```
#-------------------------------------------------------------------------------
#C-3: Utilizar el algoritmo QDA para clasificar. No olvide comprobar las asunciones.

#Asuncion de normalidad estudiadas en C2

#Aplicar QDA
qda_fold <- function(i, x) {
  file <- paste(x, "-10-", i, "tra.dat", sep="")
  x_tra <- read.csv(file, comment.char="@" , header=FALSE )
  file <- paste(x, "-10-", i, "tst.dat", sep="")
  x_tst <- read.csv(file, comment.char="@" , header=FALSE )
  names(x_tra) <- c("Mcv", "Alkphos", "Sgpt",
                    "Sgot", "Gammagt", "Drinks", "Selector")
  names(x_tst) <- c("Mcv", "Alkphos", "Sgpt",
                    "Sgot", "Gammagt", "Drinks", "Selector")
  #Preprocesamiento
  x_tra_log_transformed <- x_tra %>%
    mutate(Selector = factor(Selector, levels = c(1, 2), labels = c(1, 0))) %>%
    mutate_at(c("Alkphos", "Gammagt", "Sgpt", "Sgot", "Drinks"), log1p) %>%
    mutate_at(c("Mcv", "Alkphos", "Gammagt", "Sgpt", "Sgot", "Drinks"), scale)

  x_tst_log_transformed <- x_tst %>%
    mutate(Selector = factor(Selector, levels = c(1, 2), labels = c(1, 0))) %>%
    mutate_at(c("Alkphos", "Gammagt", "Sgpt", "Sgot", "Drinks"), log1p) %>%
    mutate_at(c("Mcv", "Alkphos", "Gammagt", "Sgpt", "Sgot", "Drinks"), scale)

  #Predicción
```

```r
  model.qda <- qda(Selector ~ ., data = x_tra_log_transformed)

  qda.pred.tra <- predict(model.qda, x_tra_log_transformed)
  qda.pred.tst <- predict(model.qda, x_tst_log_transformed)

  #Medidas de calidad

  t_tst <- table(qda.pred.tst$class, x_tst_log_transformed$Selector)
  val_acc_rate_tst <- sum(diag(t_tst)) / nrow(x_tst_log_transformed)
  t_tra <- table(qda.pred.tra$class, x_tra_log_transformed$Selector)
  val_acc_rate_tra <- sum(diag(t_tra)) / nrow(x_tra_log_transformed)

  c(val_acc_rate_tst, val_acc_rate_tra)
}

r_qda_tst <- mean(sapply(1:10, qda_fold, nombre)[1])
r_qda_tra <- mean(sapply(1:10, qda_fold, nombre)[2])

#Pasar vector de resultados a dataframe
resultados_qda_df <- data.frame(
  Conjunto = c("Test", "Entrenamiento"),
  Precision = c(r_qda_tst, r_qda_tra)
)

#Gráfico para visualizar rendimiento
ggplot(resultados_qda_df, aes(x = Conjunto, y = Precision, fill = Conjunto)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Conjunto", y = "Precisión") +
  scale_fill_manual(values = c("Test" = "red", "Entrenamiento" = "blue"))
```




```r
#-------------------------------------------------------------------------------
#C-4: Comparar los resultados de los tres algoritmos.

#Comparamos mediante rendimiento por precisión

resultados_modelos <- rbind(resultados_knn_best_df, resultados_lda_df,
                            resultados_qda_df)

resultados_modelos <- cbind (resultados_modelos, Modelo = rep(c("k-NN", "LDA", "QDA"), each = 2))

ggplot(resultados_modelos, aes(x = Modelo, y = Precision, fill = Conjunto)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Algoritmo", y = "Precisión") +
  scale_fill_manual(values = c("Test" = "red", "Entrenamiento" = "blue"))
```


0.8
0.6
0.4
0.2
0.0
Precisión
k-NN
LDA
QDA
Algoritmo
Conjunto
Entrenamiento
Test