Exploración y análisis de datos

Mikel Berganza

23/1/2021

Índice

1. Descripción del data set

El dataset seleccionado es un conjunto de datos para predecir el tipo de estrellas según las diferentes características de las mismas (https://www.kaggle.com/deepu1109/star-dataset).

El conjunto de datos contiene las siguientes variables:

- Temperatura Absoluta de la estrella en grados Kelvin (K). Valor obtenido mediante la ley de desplazamiento de Wien para encontrar la temperatura de la corteza de una estrella mediante la longitud de onda de la misma.
- Luminosidad relativa (respecto al sol) de la estrella (L/Lo). Obtenido mediante la ley de Stefan-Boltzmann de radiación de cuerpos negros.
- Radio relativo (respecto al sol) de la estrella (R/Ro). Valor obtenido utilizando paralaje (técnica para obtener la distancia a una estrella en parsecs), la temperatura, la luminosidad y la magnitud absoluta de la estrella.
- Magnitud Absoluta de la estrella (Mv). Valor obtenido mediante la magnitud aparente (es una medida del brillo de una estrella u otro objeto astronómico observado desde la Tierra) y la distancia o paralaje de una estrella.
- Color de la estrella (blanca, roja, azul, amarilla, amarillo-naranja, etc.).
- Clase espectral de la estrella (O,B,A,F,G,K,M). Clasificación basada en las características espectrales de las estrellas.
- Tipo de estrella:
    1. Brown Dwarf (Enana marrón) -> Tipo de estrella = 0
    2. Red Dwarf (Enana roja) -> Tipo de estrella = 1
    3. White Dwarf (Enana blanca) -> Tipo de estrella = 2
    4. Main Sequence (Secuencia principal) -> Tipo de estrella = 3
    5. Supergiant (Supergigantes) -> Tipo de estrella = 4
    6. Hypergiant (Hipergigantes) -> Tipo de estrella = 5

Los parámetros Lo y Ro serian la media de luminosidad y de radio del sol: Lo = 3.828 x 10^26 Watts (Luminosidad media del sol) Ro = 6.9551 x 10^8 m (Radio medio del sol)

El propósito de los que han creado el conjunto de datos sería demostrar que las estrellas siguen una determinada gráfica en el espacio celeste, específicamente llamada Diagrama de Hertzsprung-Russell para que se pueda clasificar las estrellas trazando sus características basadas en esa gráfica.

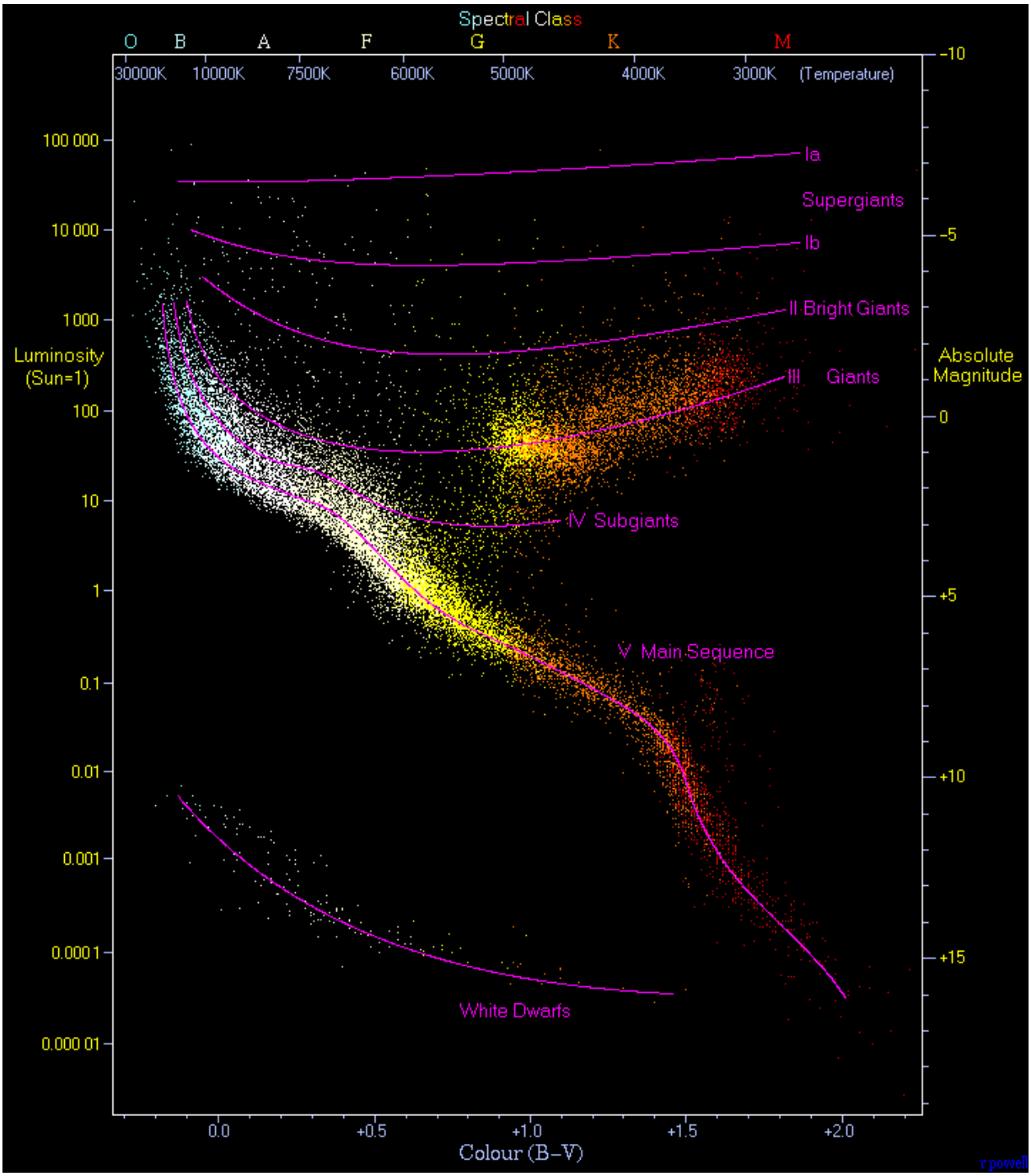


Figura 1: Imagen del Diagrama de Hertzsprung-Russell

2. Estudio descriptivo de los datos

El primer fragmento de código en R es para cargar los datos, ver los estadísticos descriptivos, cambiar los tipos de estrella de numérico a nominal (para la parte de visualización) y mirar si hay valores faltantes.

```
#Cargar los datos
data<-read.csv("stars.csv", stringsAsFactors=FALSE)
#Estadisticos de los datos por columna o variable
summary(data)
```

```
##  Temperature..K. Luminosity.L.Lo.    Radius.R.Ro.       Absolute.magnitude.Mv.
##  Min.   : 1939   Min.   :     0.0   Min.   :   0.0084   Min.   :-11.920
##  1st Qu.: 3344   1st Qu.:     0.0   1st Qu.:   0.1027   1st Qu.: -6.232
##  Median : 5776   Median :     0.1   Median :   0.7625   Median :  8.313
##  Mean   :10497   Mean   :107188.4   Mean   : 237.1578   Mean   :  4.382
##  3rd Qu.:15056   3rd Qu.:198050.0   3rd Qu.:  42.7500   3rd Qu.: 13.697
##  Max.   :40000   Max.   :849420.0   Max.   :1948.5000   Max.   : 20.060
##    Star.type    Star.color        Spectral.Class
##  Min.   :0.0   Length:240         Length:240
##  1st Qu.:1.0   Class :character   Class :character
##  Median :2.5   Mode  :character   Mode  :character
##  Mean   :2.5
##  3rd Qu.:4.0
##  Max.   :5.0
```

Como se puede observar, las cinco primeras variables del conjunto de datos son variables numéricas (incluyendo la quinta variable, clase) y las últimas dos son variables nominales. Respecto a las primeras cuatro variables se puede observar que, al ser valores obtenidos con distintas unidades de medida, son bastante diferentes entre ellas.

Al tener seis clases o tipos de estrellas diferentes parece más normal que la característica pueda variar bastante, y tener, como es en el caso de la luminosidad, una mediana de 0,1 y una media de 107188,4.

Tener una varianza tan grande en las variables podría complicar la detección de valores atípicos.

```
#6 tipos diferentes de estrellas
unique(data$Star.type)
```

```
## [1] 0 1 2 3 4 5
```

```
#Guardar orden de los numeros
starNums<-data$Star.type
#Lista de nombres del tipo de estrella
tipos<-c("Brown Dwarf", "Red Dwarf", "White Dwarf", "Main Sequence" , "SuperGiants", "HyperGiants")

#Cambiar todos los valores del tipo de estrella a nominal
for(i in 0:length(tipos)){
  #Indices que coinciden con el tipo de estrella i
  index<-which(data[,5]==i)
  #Asignar en esos indices el nombre
  data[index,5]<-tipos[i+1]
}
starsNombres<-data$Star.type
#Visualizar el encabezado de la base de datos
head(data)
```

```
##   Temperature..K. Luminosity.L.Lo. Radius.R.Ro. Absolute.magnitude.Mv.
## 1            3068         0.002400       0.1700                  16.12
## 2            3042         0.000500       0.1542                  16.60
## 3            2600         0.000300       0.1020                  18.70
## 4            2800         0.000200       0.1600                  16.65
## 5            1939         0.000138       0.1030                  20.06
## 6            2840         0.000650       0.1100                  16.98
##     Star.type Star.color Spectral.Class
## 1 Brown Dwarf        Red              M
## 2 Brown Dwarf        Red              M
## 3 Brown Dwarf        Red              M
## 4 Brown Dwarf        Red              M
## 5 Brown Dwarf        Red              M
## 6 Brown Dwarf        Red              M
```

```
#Suma de valores faltantes
missing<-sum(sapply(1:ncol(data),
                    FUN=function(r) {
                      sum(is.na(data[,r]))
                    }))
print(paste("Numero de valores faltantes:", missing))
```

```
## [1] "Numero de valores faltantes: 0"
```

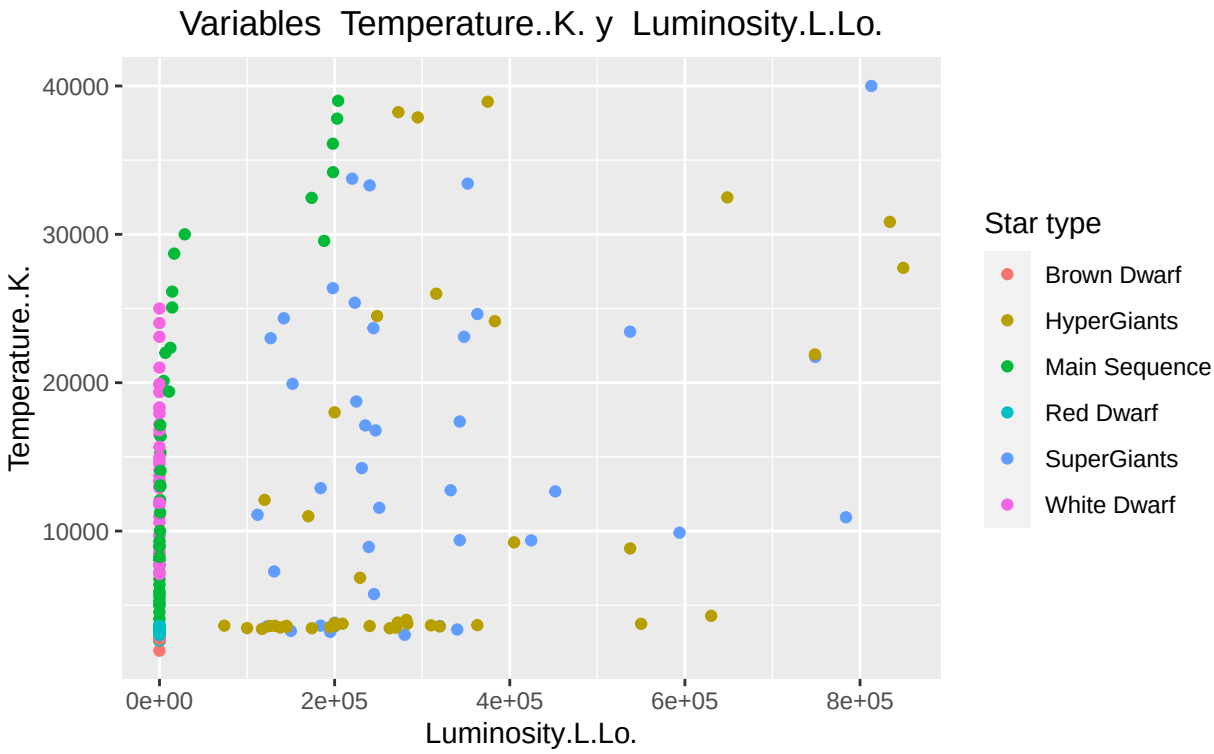
El conjunto de datos no tiene ningún valor faltante.

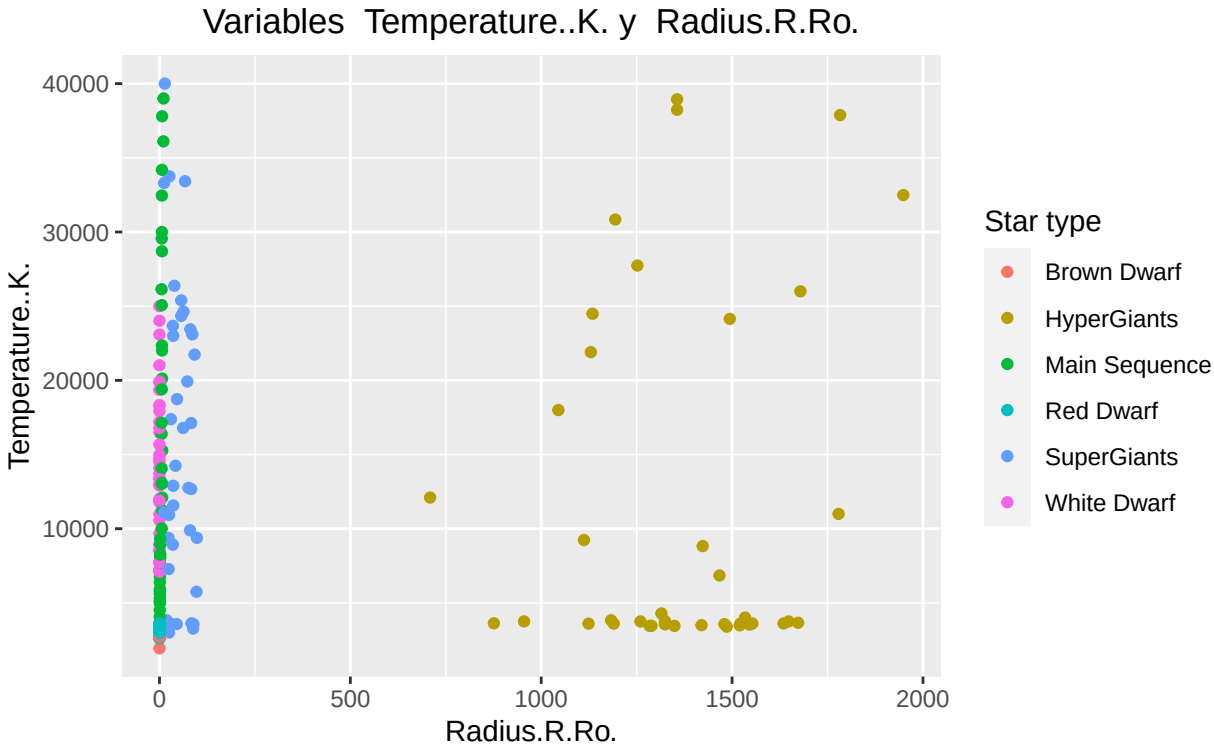
2.1. Graficas de visualización de las variables

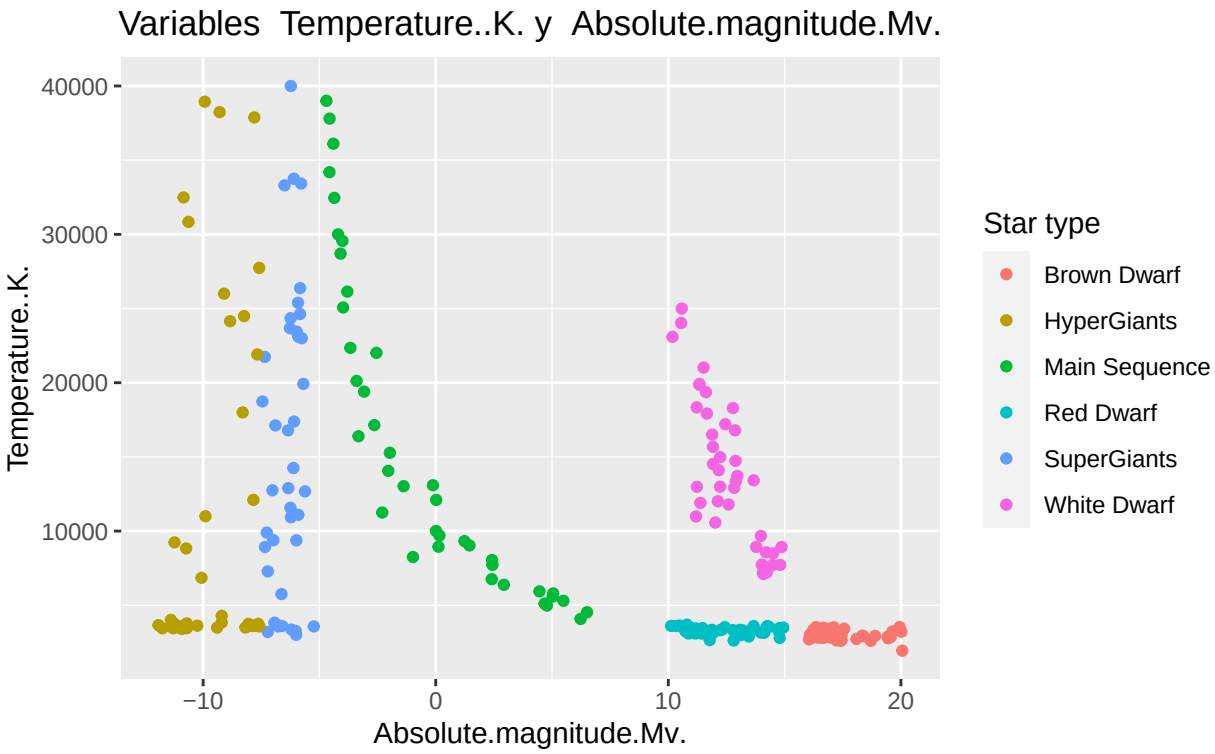
En este apartado, se va a visualizar lo siguiente:

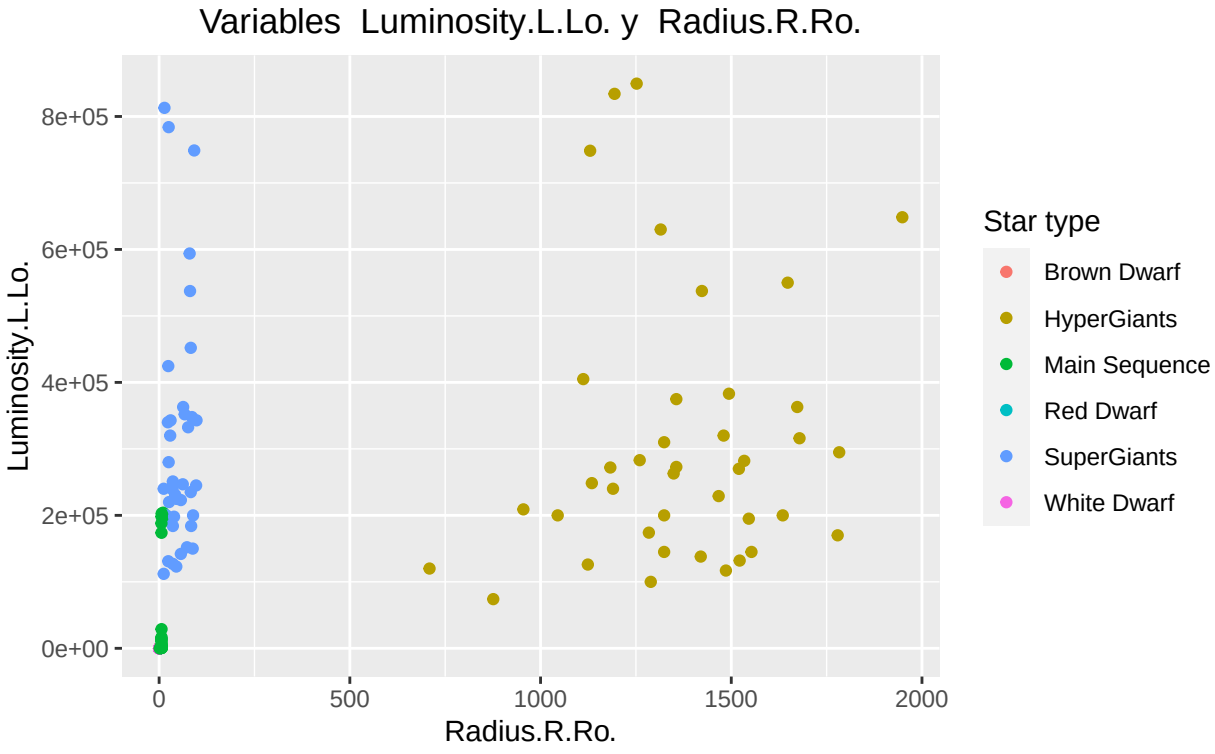
- Primero, los diferentes pares de variables numéricas. Para ver que tipo de relación puedan tener.
- Las variables numéricas respecto a la clase a predecir, para ver si hay alguna variable que pueda ser importante a la hora de clasificar los datos o predecir la clase (variable que distingue bien los tipos de estrellas).
- Gráficos de las variables nominales (la proporción de cada clase espectral y color de todas las estrella).
- Boxplots de las variables numéricas.

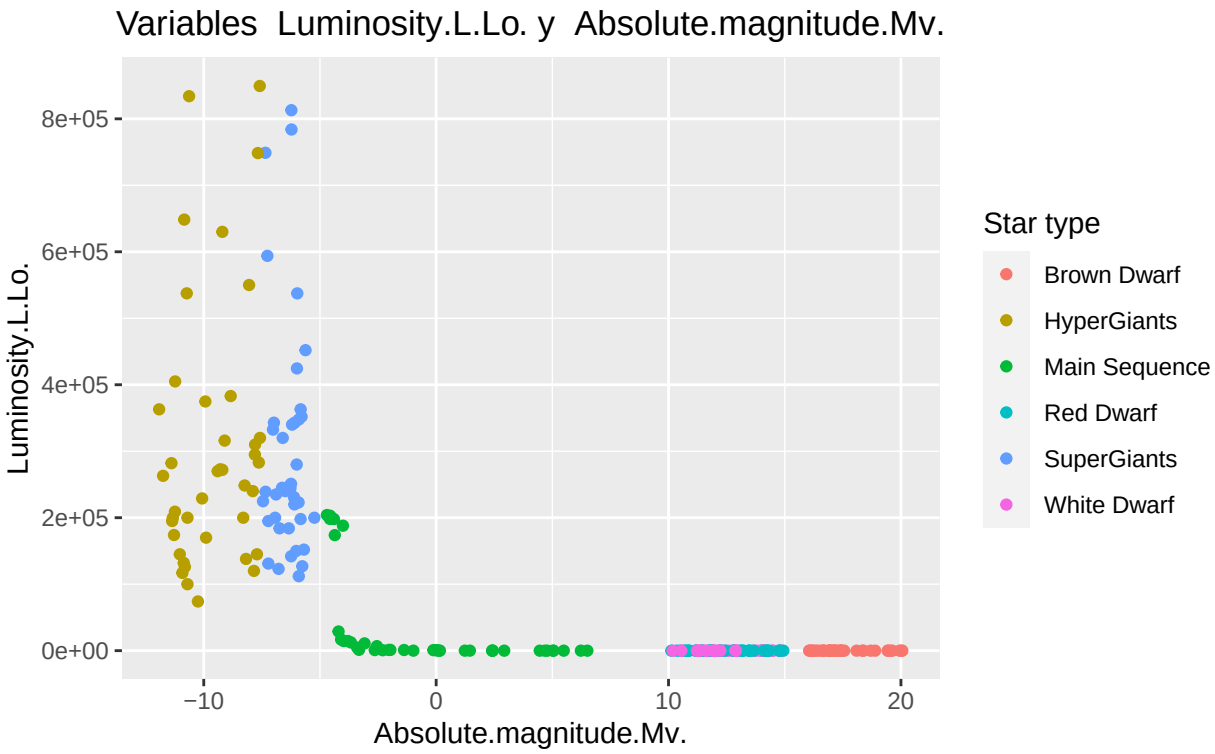
```
#Plot entre las variables numericas excepto la clase
for (i in 1:3){
  for (j in (i+1):4){
    #Pasar a caracter la columna de clase para cojer 6 colores diferentes
    #en el parametro color
    print(ggplot(data = data) +
            geom_point(mapping = aes(x = data[,j], y = data[,i], color = as.character(Star.type))) +
            xlab(colnames(data)[j]) + ylab(colnames(data)[i]) +
            labs(color="Star type") +
            ggtitle(paste("Variables ",colnames(data)[i],"y ",colnames(data)[j]))+
            theme(plot.title = element_text(hjust = 0.5)))
  }
}
```

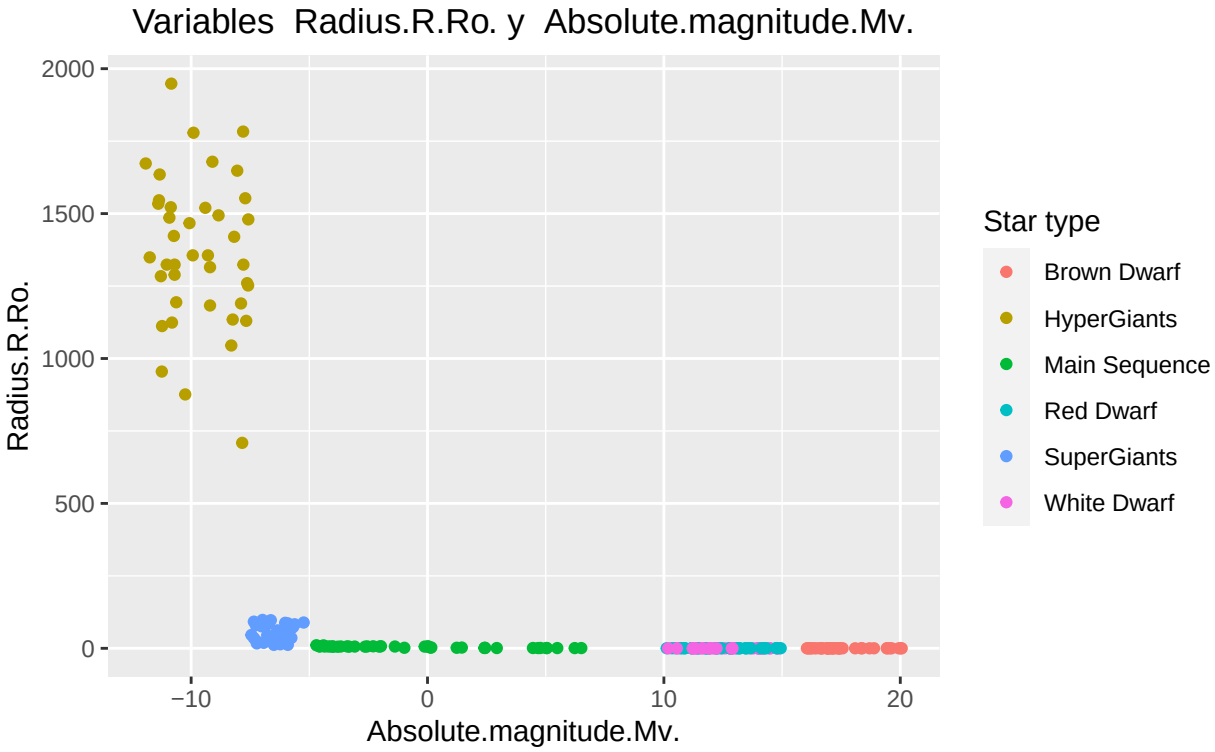

Variables Temperature..K. y Luminosity.L.Lo.
Temperature..K.
40000
30000
20000
10000
0e+00
2e+05
4e+05
6e+05
8e+05
Luminosity.L.Lo.
Star type
Brown Dwarf
HyperGiants
Main Sequence
Red Dwarf
SuperGiants
White Dwarf


Variables Temperature..K. y Radius.R.Ro.
Temperature..K.
40000
30000
20000
10000
0
500
1000
1500
2000
Radius.R.Ro.
Star type
Brown Dwarf
HyperGiants
Main Sequence
Red Dwarf
SuperGiants
White Dwarf


Variables Temperature..K. y Absolute.magnitude.Mv.
Temperature..K.
40000
30000
20000
10000
−10
0
10
20
Absolute.magnitude.Mv.
Star type
Brown Dwarf
HyperGiants
Main Sequence
Red Dwarf
SuperGiants
White Dwarf

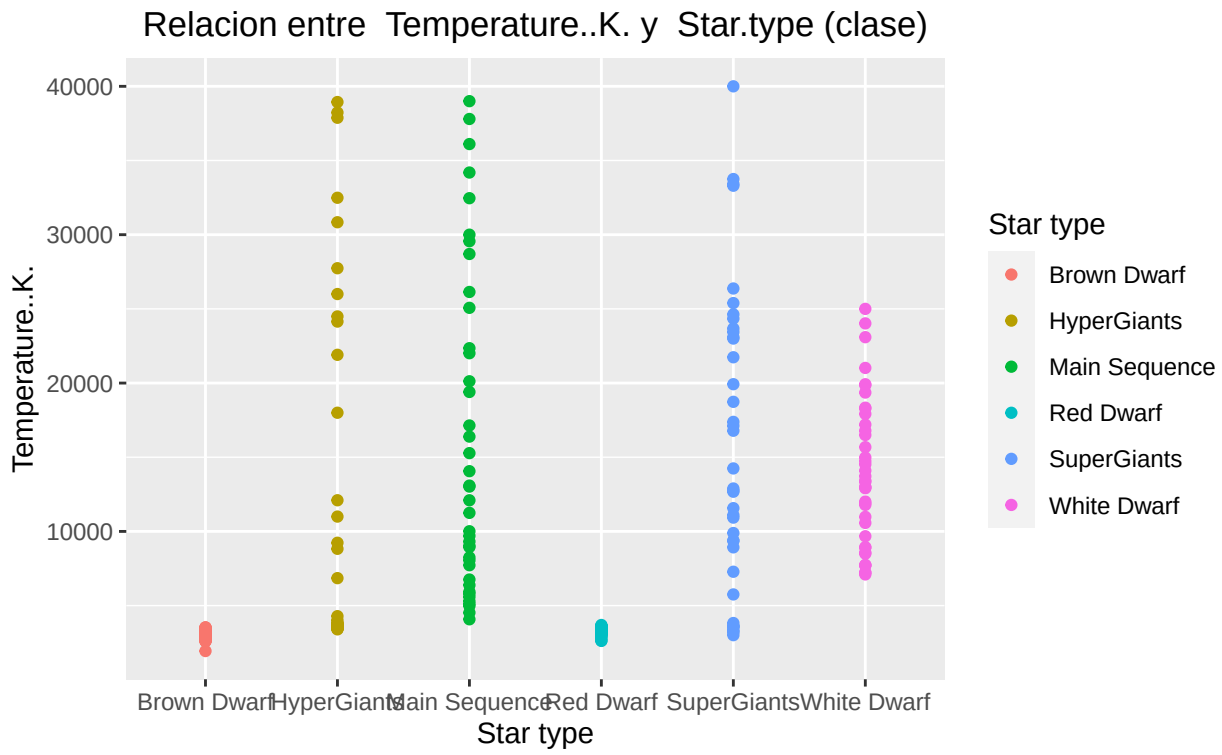

Variables Luminosity.L.Lo. y Radius.R.Ro.
Luminosity.L.Lo.
8e+05
6e+05
4e+05
2e+05
0e+00
0
500
1000
1500
2000
Radius.R.Ro.
Star type
Brown Dwarf
HyperGiants
Main Sequence
Red Dwarf
SuperGiants
White Dwarf

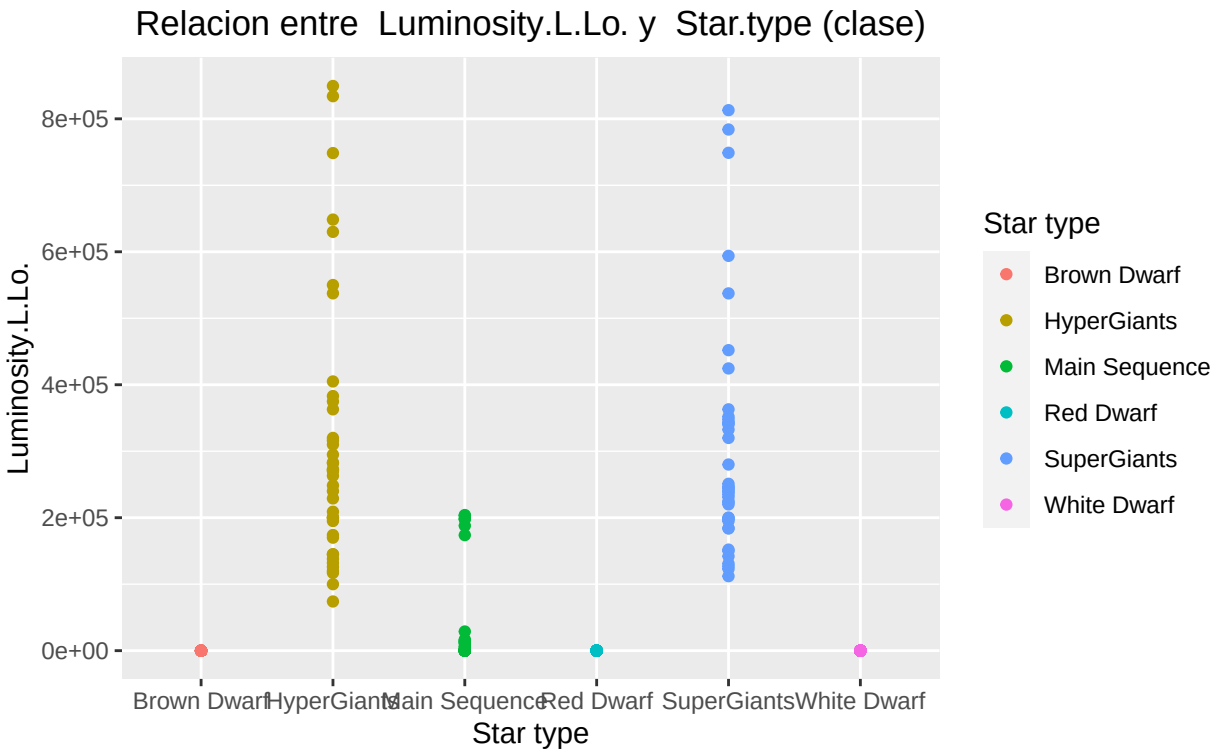


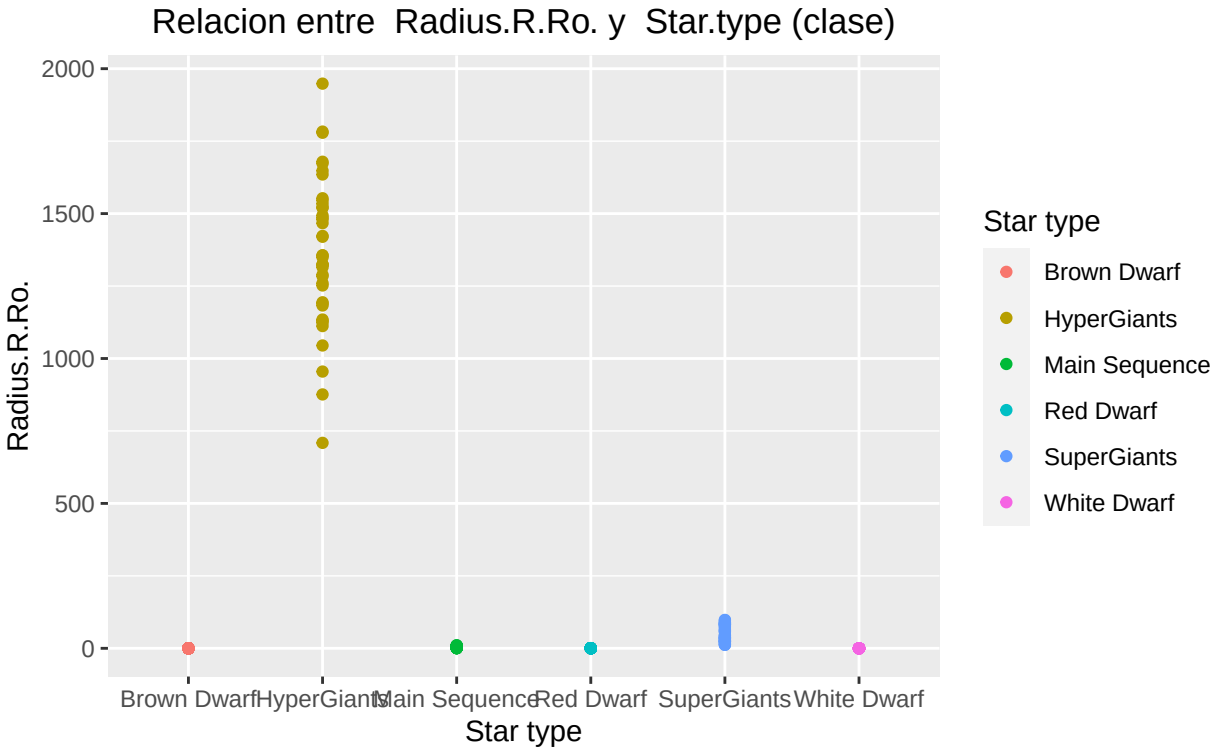


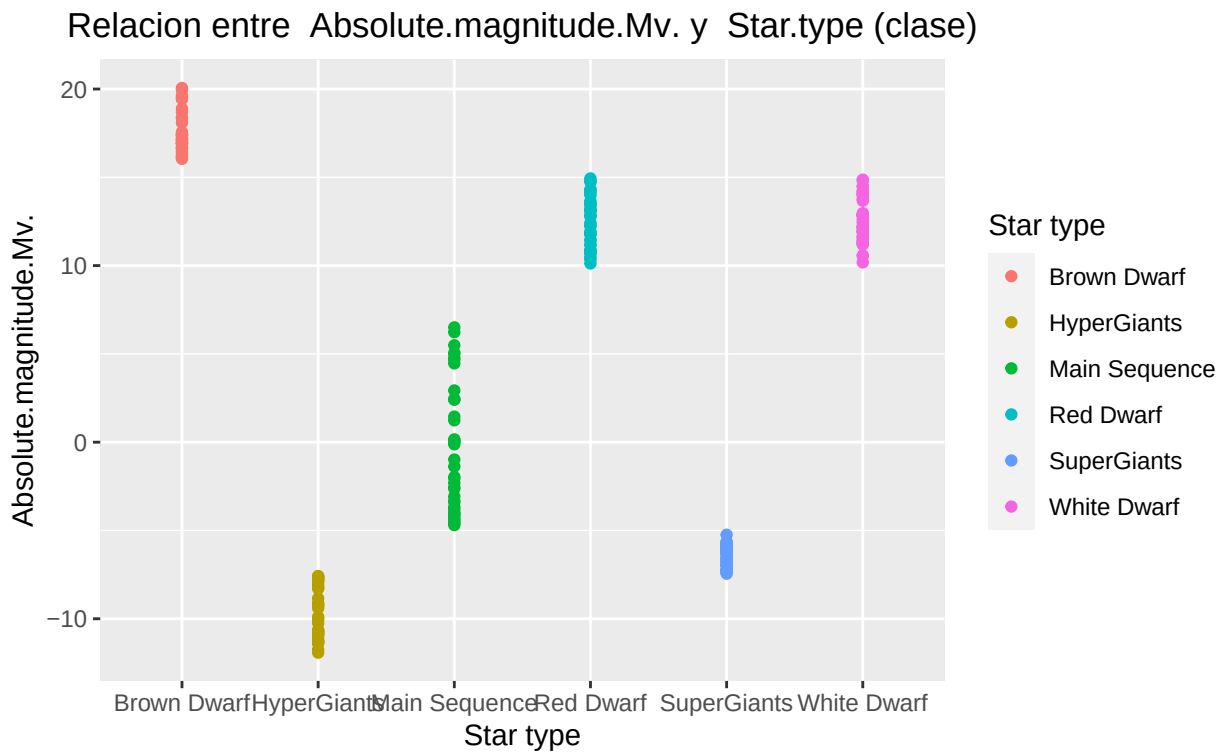
Al observar los gráficos de relaciones entre las variables numéricas, se puede ver como hay combinaciones que no aportan nada a la hora de diferenciar los tipos de estrellas, pero hay alguna que sí. Por ejemplo, las variables de temperatura y magnitud absoluta, consiguen una diferenciación perfecta de los tipos de estrellas. Las variables de luminosidad y radio en relación con la magnitud absoluta consiguen también una diferenciación casi perfecta.

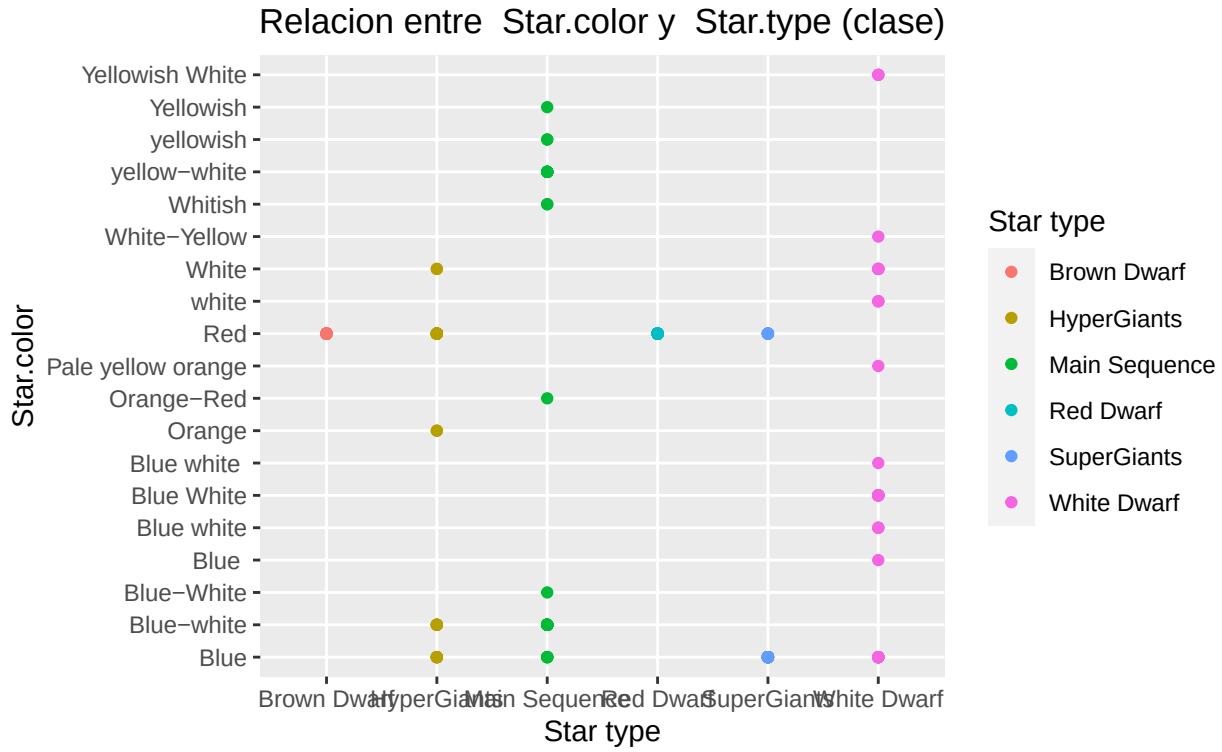
```r
#Plotear todas las variables con respecto a la clase
for (i in 1:7){
  if (i!=5){
    print(ggplot(data = data) +
            geom_point(mapping = aes(x = data[,5], y = data[,i], color = as.character(Star.type))) +
            xlab("Star type") + ylab(colnames(data)[i]) +
            labs(color="Star type") +
            ggtitle(paste("Relacion entre ",colnames(data)[i],"y  Star.type (clase)"))+
            theme(plot.title = element_text(hjust = 0.5)))
  }
}
```

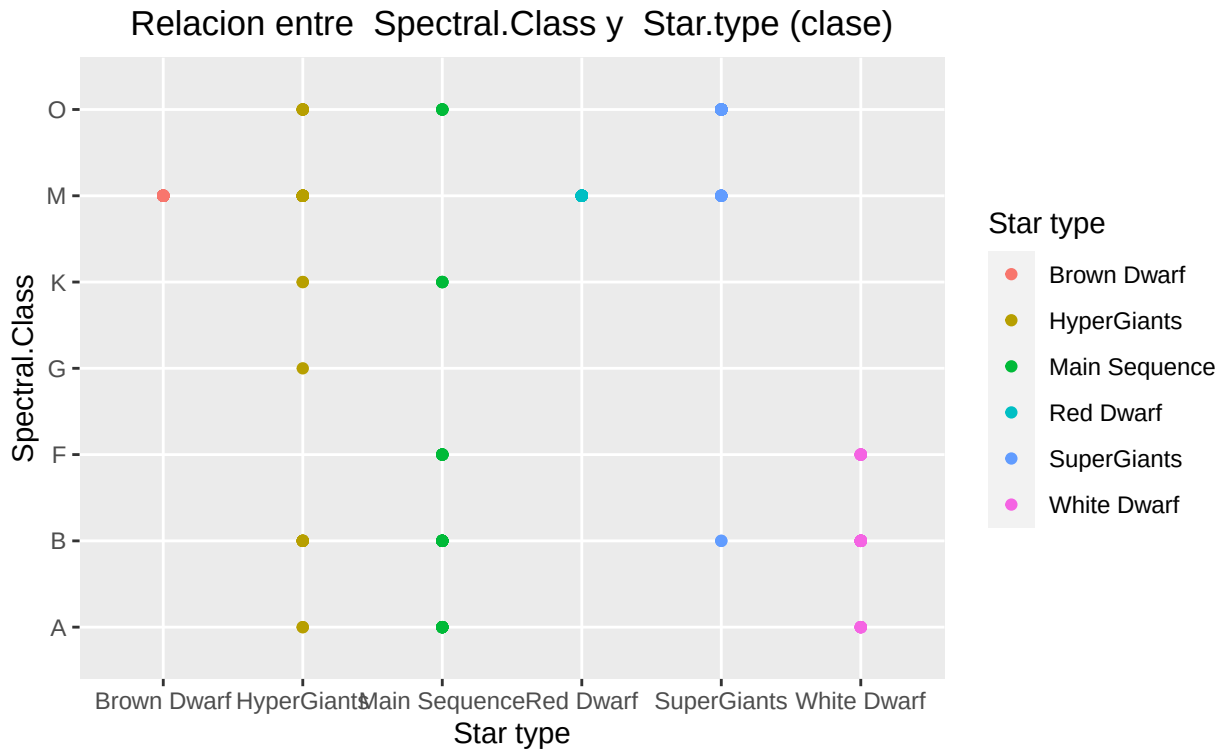



Relacion entre Luminosity.L.Lo. y Star.type (clase)
Luminosity.L.Lo.
8e+05
6e+05
4e+05
2e+05
0e+00
Brown Dwarf
HyperGiants
Main Sequence
Red Dwarf
SuperGiants
White Dwarf
Star type
Star type
Brown Dwarf
HyperGiants
Main Sequence
Red Dwarf
SuperGiants
White Dwarf


Relacion entre Radius.R.Ro. y Star.type (clase)
Radius.R.Ro.
2000
1500
1000
500
0
Brown Dwarf
HyperGiants
Main Sequence
Red Dwarf
SuperGiants
White Dwarf
Star type
Star type
Brown Dwarf
HyperGiants
Main Sequence
Red Dwarf
SuperGiants
White Dwarf

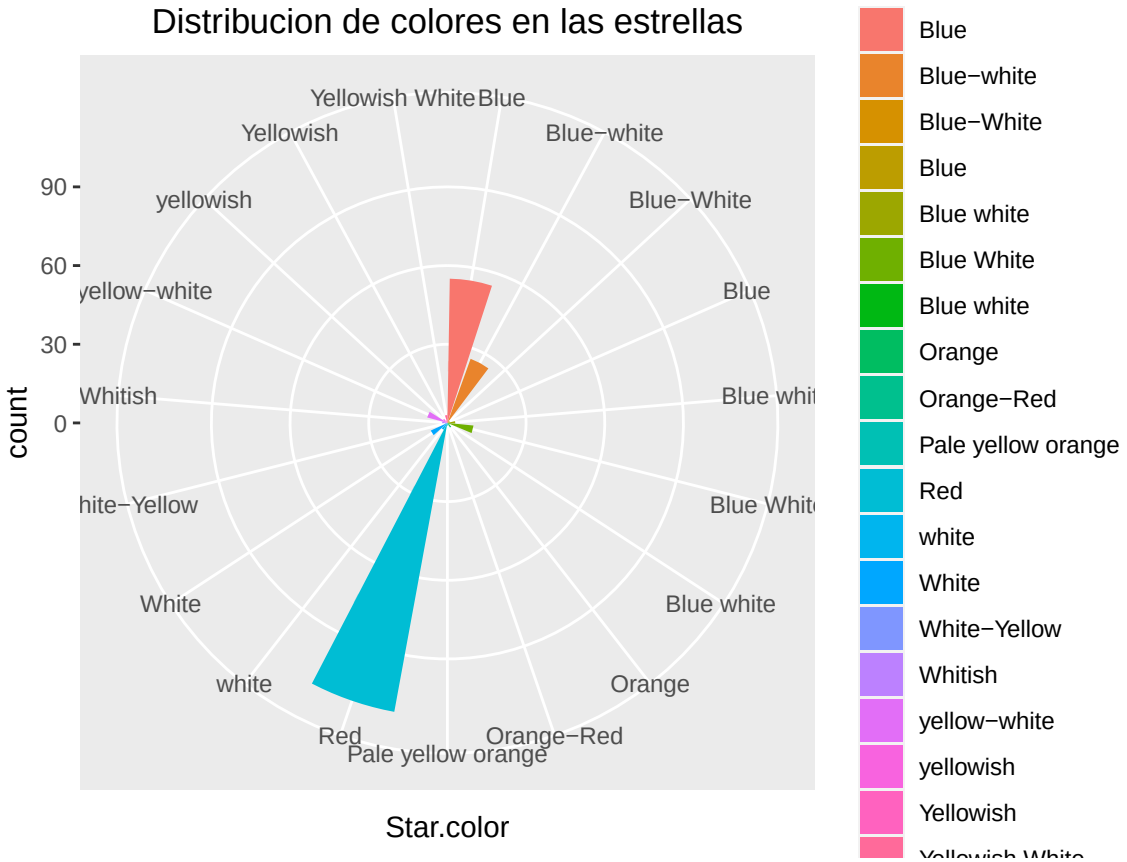

Relacion entre Absolute.magnitude.Mv. y Star.type (clase)
Absolute.magnitude.Mv.
20
10
0
−10
Brown Dwarf
HyperGiants
Main Sequence
Red Dwarf
SuperGiants
White Dwarf
Star type
Star type
Brown Dwarf
HyperGiants
Main Sequence
Red Dwarf
SuperGiants
White Dwarf


Relacion entre Star.color y Star.type (clase)
Star.color
Yellowish White
Yellowish
yellowish
yellow−white
Whitish
White−Yellow
White
white
Red
Pale yellow orange
Orange−Red
Orange
Blue white
Blue White
Blue white
Blue
Blue−White
Blue−white
Blue
Brown Dwarf
HyperGiants
Main Sequence
Red Dwarf
SuperGiants
White Dwarf
Star type
Star type
Brown Dwarf
HyperGiants
Main Sequence
Red Dwarf
SuperGiants
White Dwarf

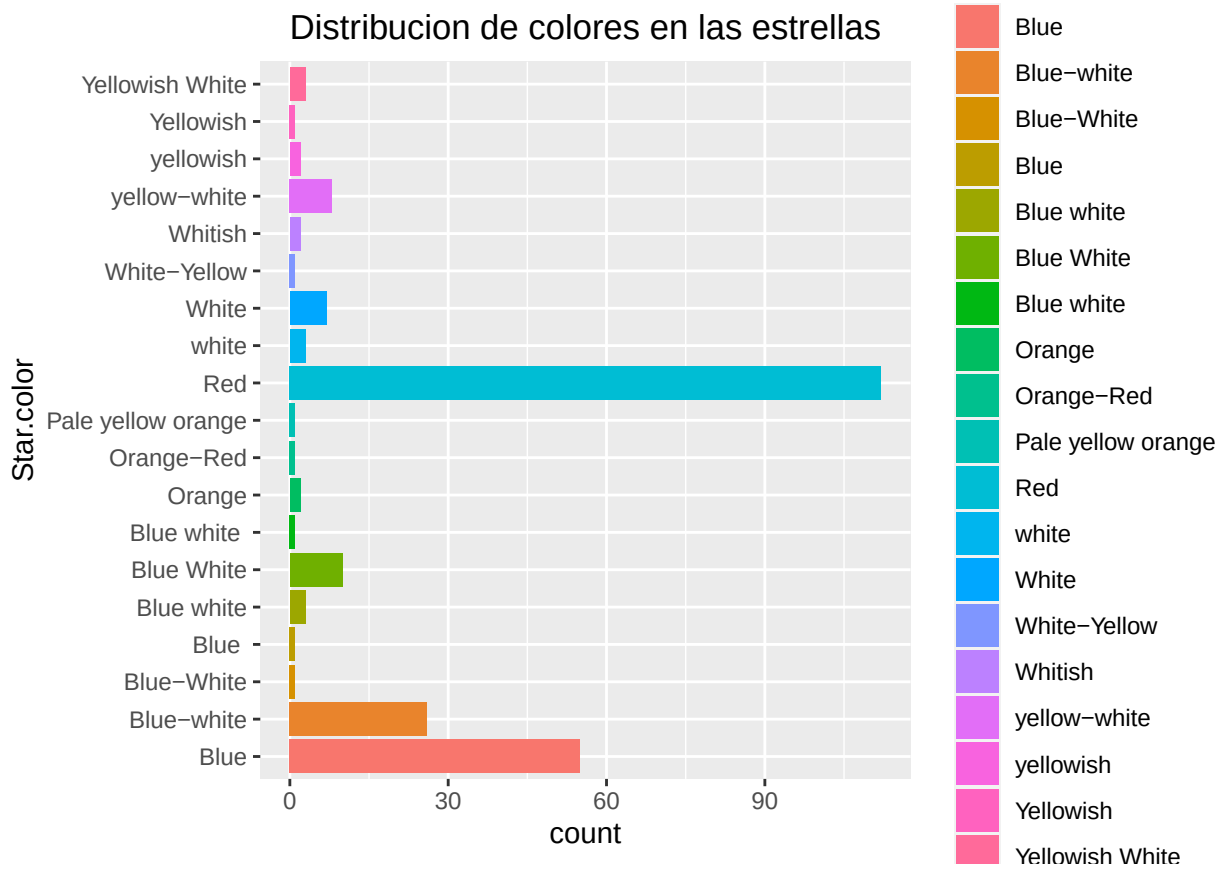


En la relación de variables respecto a la clase, se puede observar como se distribuyen los valores de las variables para cada tipo de estrella. Observando los gráficos, parece que solo la variable de magnitud absoluta parece distinguir la mayoría (5) de tipos de estrella. Aparte de eso, parece que las enanas marrones y rojas son todas de color rojo, y de magnitud M.

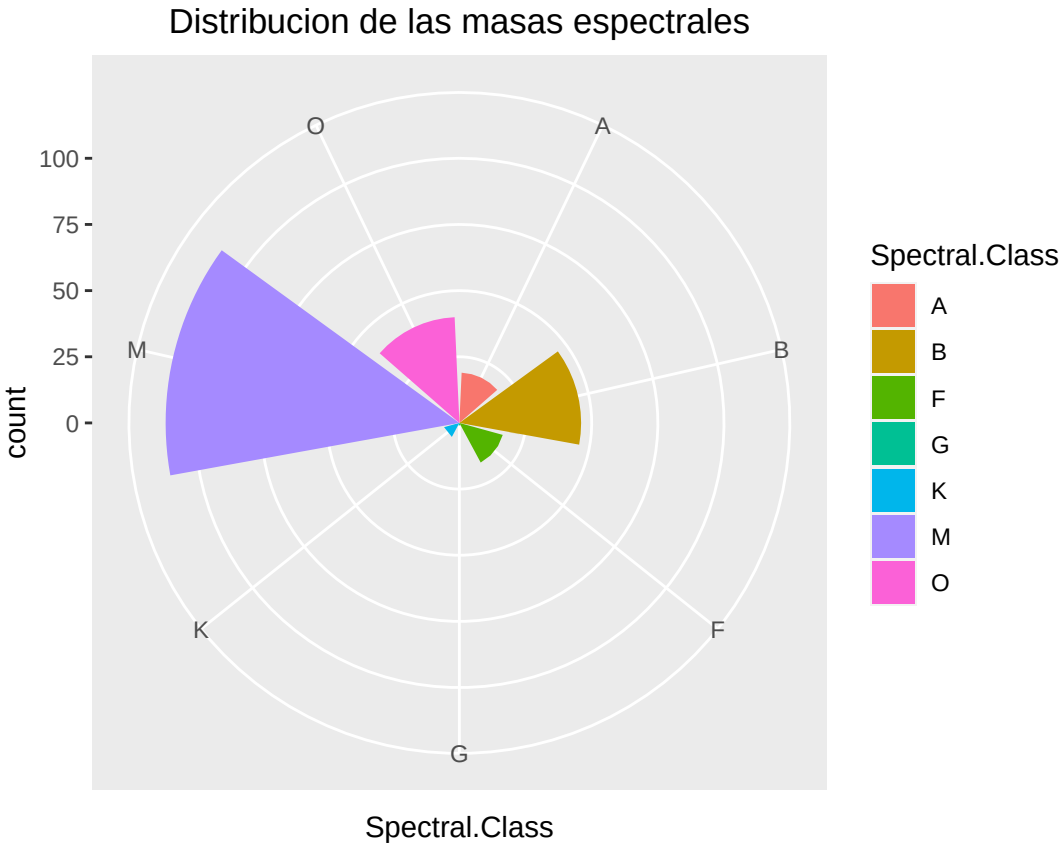
```
#Plotear las dos variables nominales
#Colores
ggplot(data = data) +
        geom_bar(mapping = aes(y = data[,6], fill = Star.color)) +
        ylab(colnames(data)[6]) +
        labs(color="Star type") +
        ggtitle(paste("Distribucion de colores en las estrellas"))+
        theme(plot.title = element_text(hjust = 0.5))+
        coord_polar("y", start=0)
```



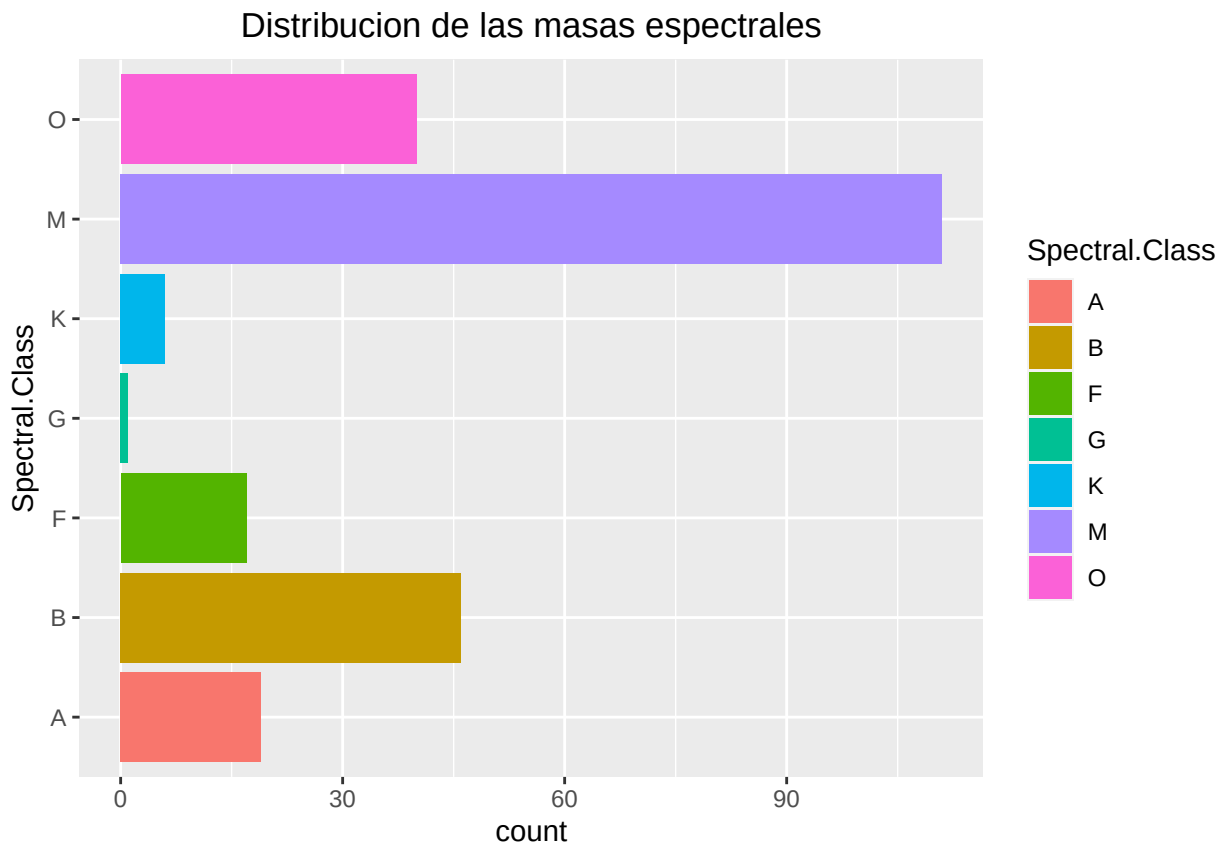

```
#Barplot de los colores
ggplot(data = data) +
  geom_bar(mapping = aes(y = data[,6], fill = Star.color)) +
  ylab(colnames(data)[6]) +
  labs(color="Star type") +
  ggtitle(paste("Distribucion de colores en las estrellas"))+
  theme(plot.title = element_text(hjust = 0.5))
```




```
#Masa espectral
ggplot(data = data) +
  geom_bar(mapping = aes(y = data[,7], fill = Spectral.Class)) +
  ylab(colnames(data)[7]) +
  labs(color="Masa Espectral") +
  ggtitle(paste("Distribucion de las masas espectrales"))+
  theme(plot.title = element_text(hjust = 0.5))+
  coord_polar("y", start=0)
```

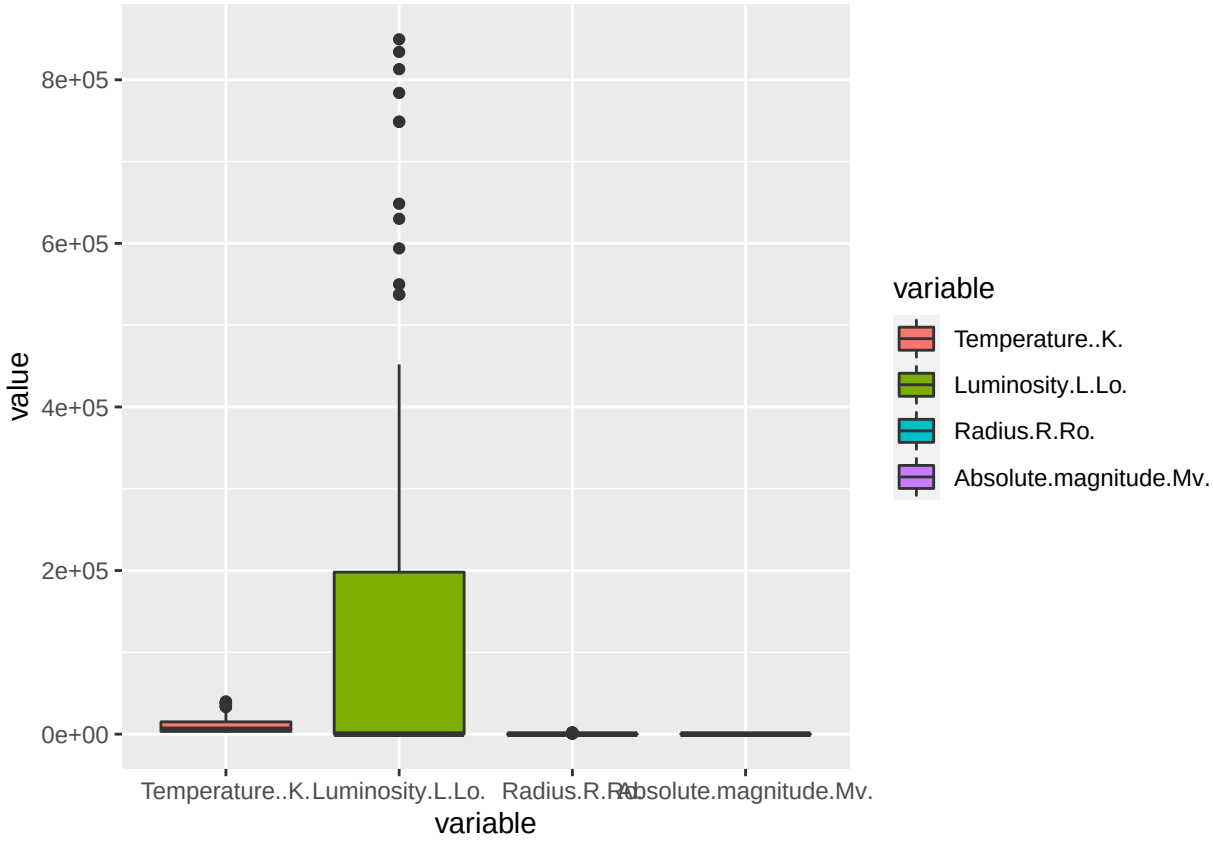



```
#Barplot de las masas espectrales
ggplot(data = data) +
  geom_bar(mapping = aes(y = data[,7], fill = Spectral.Class)) +
  ylab(colnames(data)[7]) +
  labs(color="Masa Espectral") +
  ggtitle(paste("Distribucion de las masas espectrales"))+
  theme(plot.title = element_text(hjust = 0.5))
```

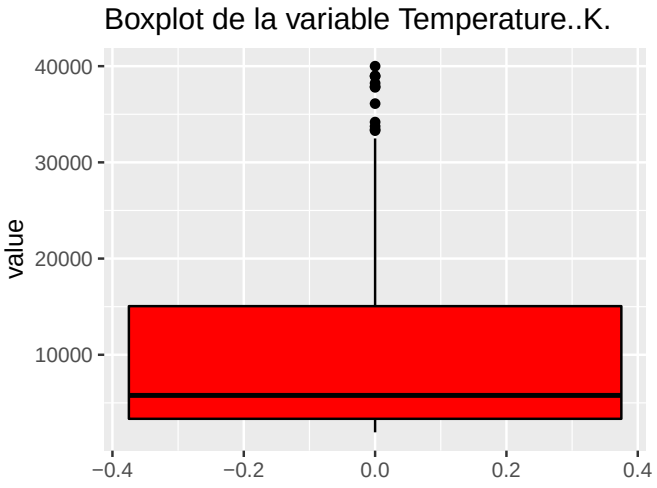


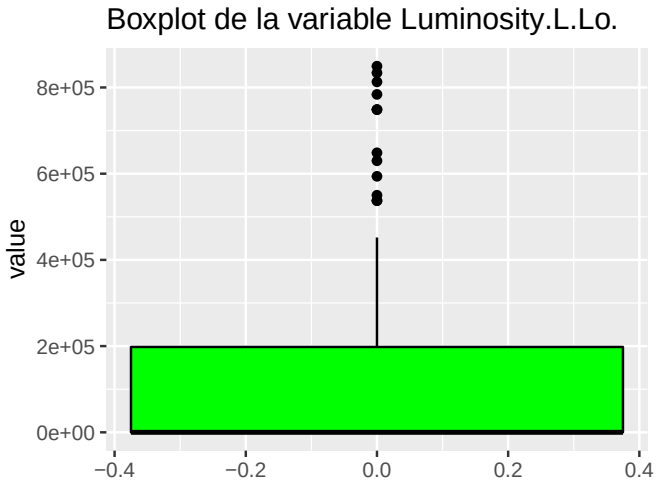
En los gráficos de barras de las variables nominales, se pueden ver la distribución de las variables nominales (color y masa espectral). Del gráfico de los colores, se puede decir que la mayoría (más de una tercera parte) de las estrellas observadas son de color rojo, y luego las seguirían las de color azul. Respecto a las masas espectrales, la mayoría parece ser de tipo M, y luego las seguirían las de tipo B.

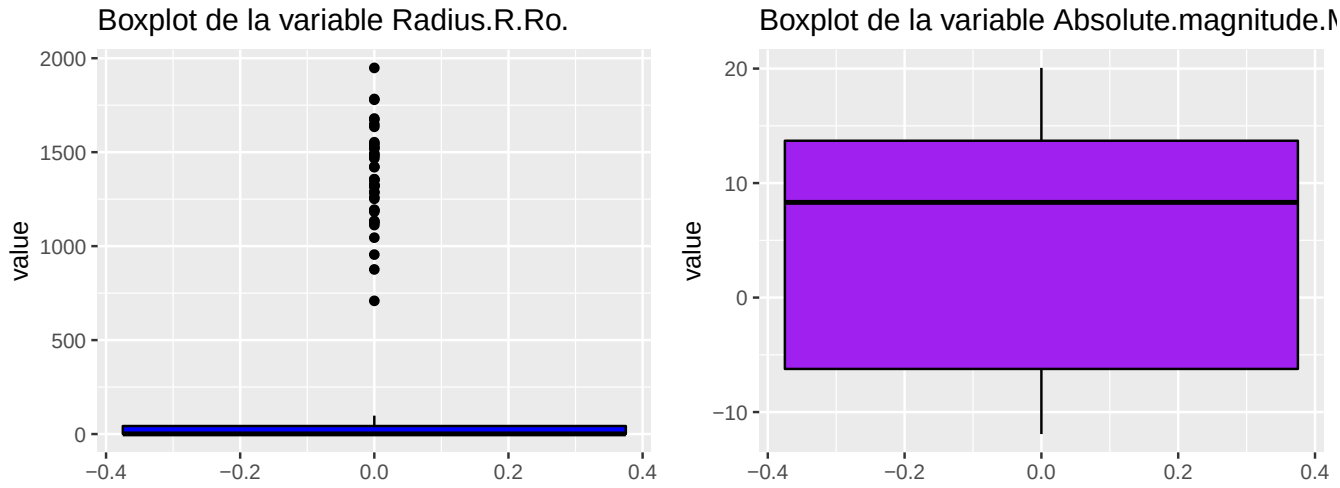
```
#Variables numericas
smallData<-data[,1:4]
#Boxplots de todas las variables
df <- melt(smallData, measure.vars=1:ncol(smallData))
ggplot(data=df) + geom_boxplot(mapping = aes(x=variable, y=value, fill=variable))
```




```
#Boxplots de las 4 variables numericas por separado
colors<-c("red", "green", "blue", "purple")
for (i in 1:4){
  print(ggplot(data=smallData) +
          geom_boxplot(mapping = aes(y=smallData[,i]), fill = colors[i], color = "black") +
          ylab("value") +
          ggtitle(paste("Boxplot de la variable",colnames(smallData)[i])))
}
```





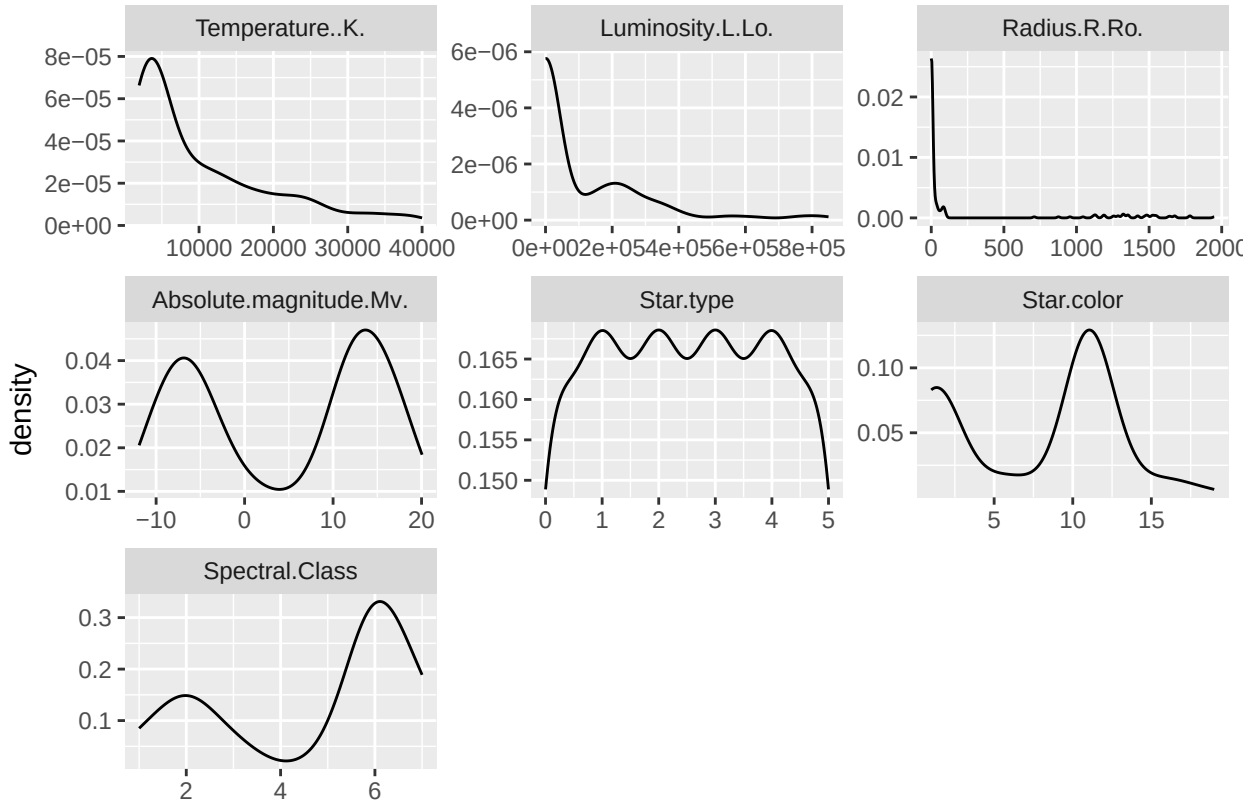


Como se ha comentado antes, al tener seis clases diferentes y una varianza tan grande en las características, parece difícil interpretar qué valor puede ser considerado outlier, por ejemplo, con un boxplot. Sin embargo, se puede ver que las variables de radio y luminosidad parecen tener una cola larga (densidad). La variable de magnitud absoluta no tiene valores atípicos, parece bastante compacta.

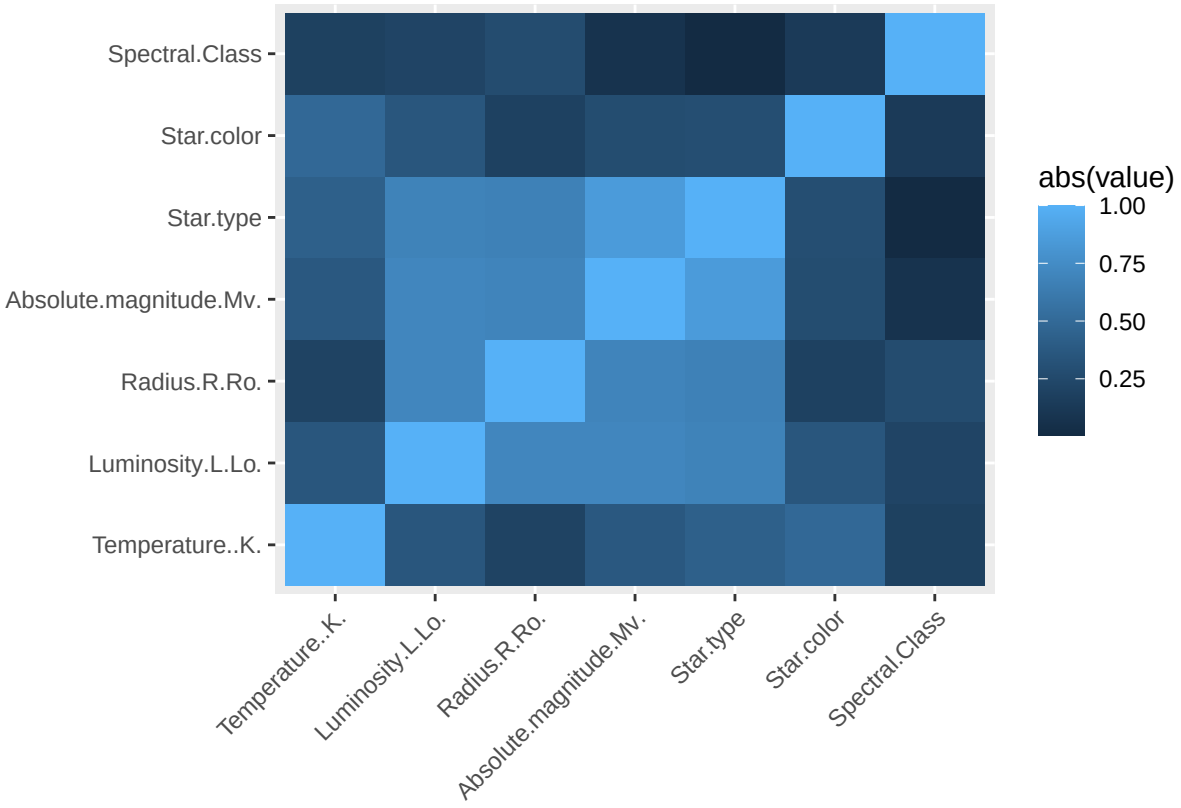
2.2. Densidad y correlación de las variables

En este apartado, se van a comprobar las densidades de las diferentes variables (habiendo pasado las variables nominales a numéricas mediante factorización) y luego la matriz de correlación entre ellas.

```
#Como las ultimas dos columnas son categoricas, pasar a numericas
#para ver la correlacion
colorsOrder<-sort(unique(data$Star.color))
spectralOrder<-sort(unique(data$Spectral.Class))

#Crear nuevo dataset, para no modificar el original
dataNum<-data
#Pasar las columnas 6 y 7 (color y masa espectral) a numericas
#con el orden de numeros creado arriba (alfabetico)
dataNum$Star.color<-as.numeric(factor(dataNum$Star.color, colorsOrder))
dataNum$Spectral.Class<-as.numeric(factor(dataNum$Spectral.Class, spectralOrder))

#Devolver a numeros para calcular densidades y correlacion entre
#variables
dataNum[,5]<-starNums

#Juntar todos los datos
df <- melt(dataNum, measure.vars=1:ncol(dataNum))
#Plotear las densidades de cada variable
ggplot(df, aes(x=value)) + geom_line(stat="density") +
  facet_wrap(~variable, scales="free") + labs(x="")
```




```
#Densidades complejas, no Gaussianas.


#use="complete.obs" si hay valores faltantes
correlation.matrix <- cor(dataNum, method="kendall")
df2 <- melt(correlation.matrix)
ggplot(df2, aes(x=Var1, y=Var2, fill=abs(value))) + geom_tile() +
  labs(x="", y="") + theme(axis.text.x=element_text(angle=45, hjust=1))
```



En el gráfico de las densidades, se puede observar que hay alguna que otra densidad bimodal (magnitud absoluta, clase espectral), pero en general parecen tener densidades complejas con colas muy largas como es en el caso de la temperatura, luminosidad y el radio.

En cuanto a las correlaciones, se puede ver que las variables más correladas con el tipo de estrella son las variables de luminosidad, radio y magnitud absoluta, siendo esta última la que mejor relación tiene (como también se ha visto en los gráficos anteriores).

Las variables nominales no parecen tener mucha correlación con las demás, al igual que la temperatura.

Para calcular las correlaciones de las variables se ha utilizado el coeficiente de Kendall, puesto que, se dice que la correlación de Kendall es más robusta y eficiente que la correlación de Spearman y se prefiere cuando hay muestras pequeñas o algunos valores atípicos, que encajaría mejor en este caso.

3. Principal Component Analysis

En este apartado se va a usar el Principal Component Analysis (PCA) como algoritmo de extracción de variables. El cual trata de reducir la dimensionalidad de las variables mediante una transformación lineal que escoge un nuevo sistema de coordenadas para el conjunto original de datos para maximizar la separación entre las clases (la varianza de mayor tamaño del conjunto de datos es capturada en el primer eje, la segunda varianza más grande es el segundo eje, y así sucesivamente).

Se ha decidido utilizar este método de reducción de dimensionalidad, puesto que parece de los métodos más utilizados y tiene una gran utilidad en el análisis de datos multivariados.

```r
#Particion de la base de datos en train y test con caTools
#80% train, 20% test
set.seed(999)
sample<-sample.split(dataNum$Temperature..K., SplitRatio = .80)
train<-subset(dataNum, sample == TRUE)
test<-subset(dataNum, sample == FALSE)

#varianza de las variables numericas
apply(data[,1:4], MARGIN = 2, FUN = var)
```

```
##        Temperature..K.       Luminosity.L.Lo.           Radius.R.Ro.
##           9.124882e+07           3.219593e+10           2.674501e+05
## Absolute.magnitude.Mv.
##           1.109338e+02
```

Aunque el análisis de componentes principales se realiza típicamente en la matriz de covarianzas, a menudo tiene un sentido más intuitivo aplicar PCA a la matriz de correlación.

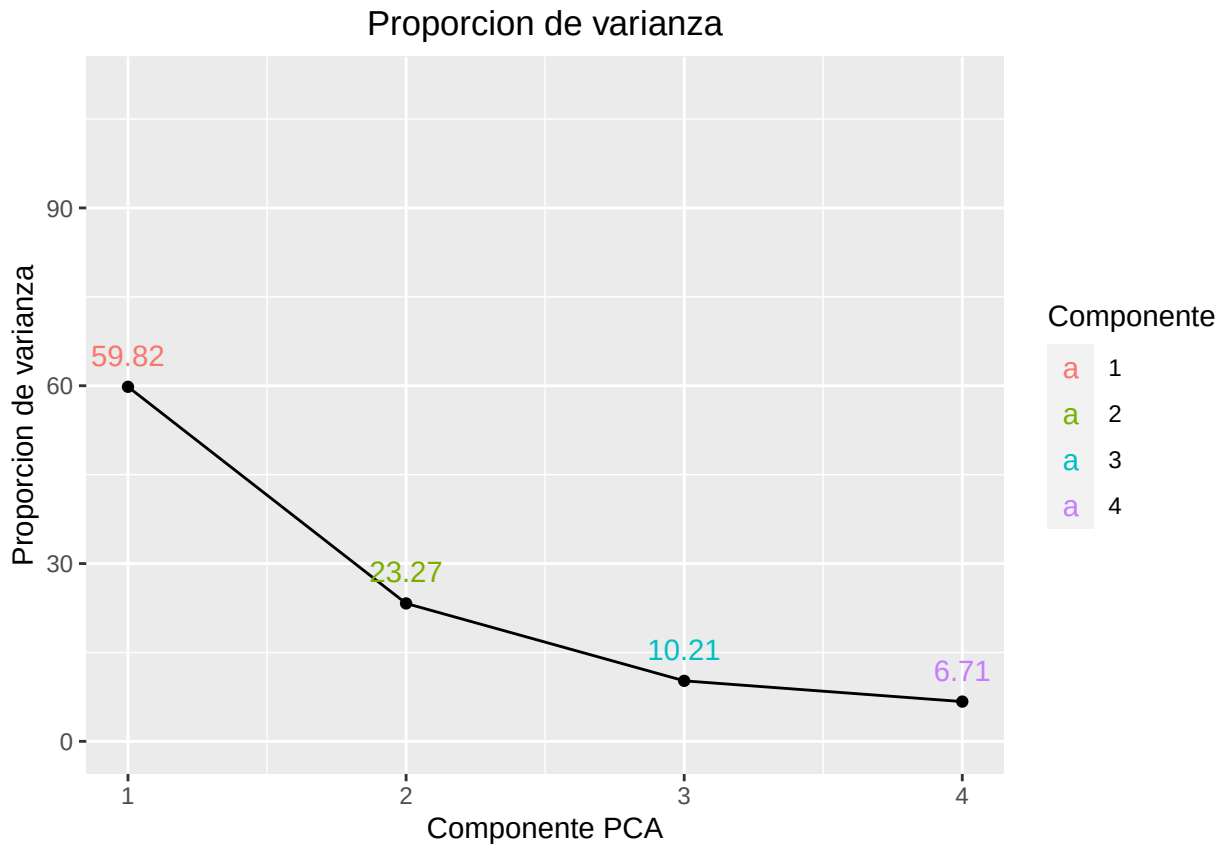
Los casos en los que el uso de la matriz de correlación puede ser preferible a la matriz de covarianzas incluyen datos que se miden en diferentes unidades o que tienen amplias varianzas, que es justo lo que ocurre en este caso.

```r
#PCA con matriz de correlaciones (scale = TRUE)
pcaT<-prcomp(train[,1:4], scale. = TRUE)
summary(pcaT)
```

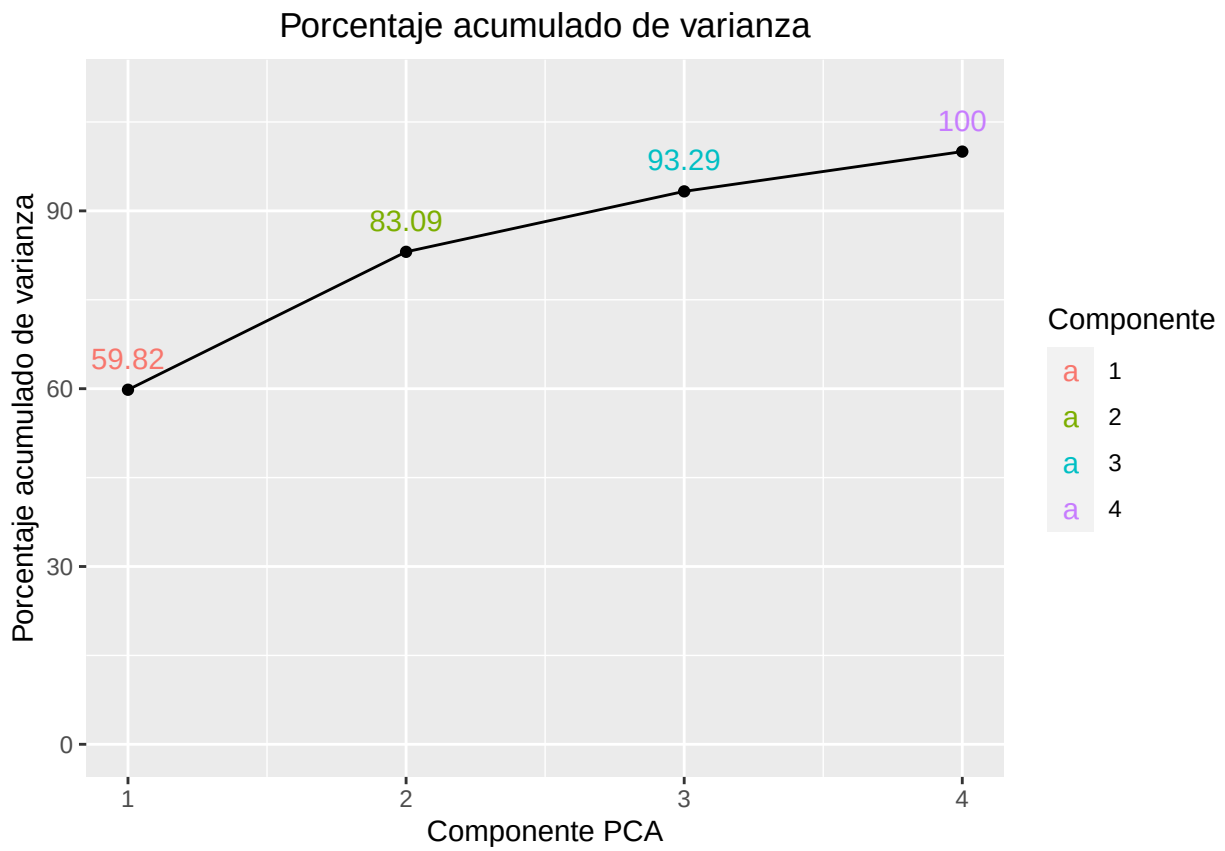
```
## Importance of components:
##                           PC1    PC2    PC3     PC4
## Standard deviation     1.5469 0.9647 0.6390 0.51789
## Proportion of Variance 0.5982 0.2327 0.1021 0.06705
## Cumulative Proportion  0.5982 0.8309 0.9329 1.00000
```

```r
#Desviacion estandar de las componentes
sdList<-pcaT$sdev
#Hacer al cuadrado para la varianza
sdList<-sdList^2

#Porcentaje de cada componente
porc<-sdList/sum(sdList)*100

#Pasar los datos a data.frame para poder dibujarlos con ggplot
xpos<-c(1,2,3,4)
porcD<-data.frame(Component=xpos,
                  Varianza=
                    as.numeric(formatC(porc,digits = 2, format = "f")),
                  CumulativePercentage=
                    as.numeric(formatC(cumsum(porc),digits = 2, format = "f")))

#Grafos de proporcion de varianza y varianza acumulada de las componentes
ggplot(data=porcD,aes(x=porcD[,1],y=porcD[,2], group=1))+
  geom_line()+geom_point()+
  geom_text(aes(x = porcD[,1],
```

```
                y = porcD[,2],
                label=porcD[,2],
                color=as.character(Component)),hjust=0.5, vjust=-1)+
  labs(color="Componente")+ ylim(0,110)+
  xlab("Componente PCA")+ ylab("Proporcion de varianza")+
  ggtitle("Proporcion de varianza")+
  theme(plot.title = element_text(hjust = 0.5))
```




```
#Porcentaje acumulado
ggplot(data=porcD,aes(x=porcD[,1],y=porcD[,3], group=1))+
  geom_line()+geom_point()+
  geom_text(aes(x = porcD[,1],
                y = porcD[,3],
                label=porcD[,3],
                color=as.character(Component)),hjust=0.5, vjust=-1)+
  labs(color="Componente")+ ylim(0,110)+
  xlab("Componente PCA")+ ylab("Porcentaje acumulado de varianza")+
  ggtitle("Porcentaje acumulado de varianza")+
  theme(plot.title = element_text(hjust = 0.5))
```



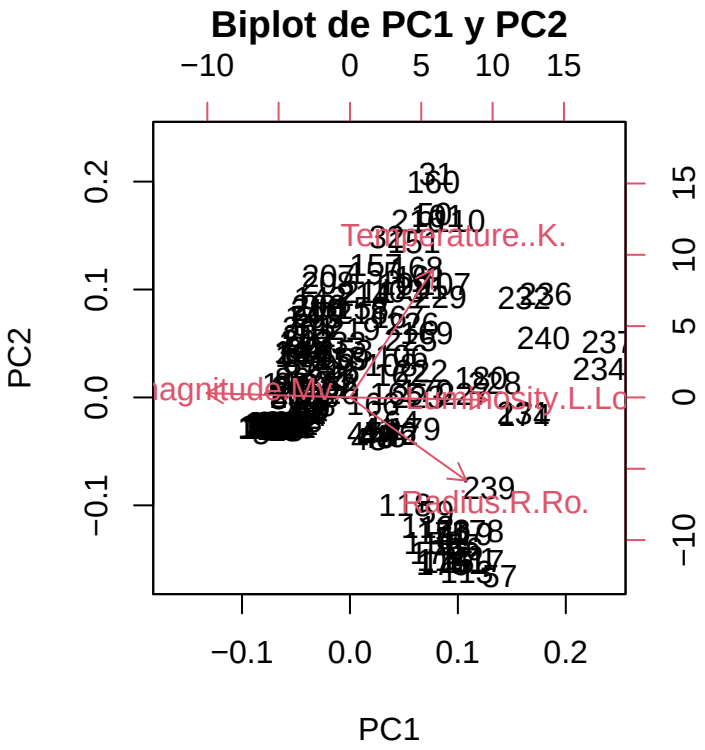
Viendo las imágenes de la proporción de varianza y varianza acumulada de las variables, se podrían coger 2 o 3 componentes. Con el método de Kaiser, solo 1 variable cumple el requisito de que el porcentaje de varianza sea mayor que 100/4=25. Sin embargo, en este caso también se podría coger la segunda componente, puesto que los resultados siguen siendo fáciles de dibujar e interpretar con dos dimensiones.

Por ello, se cogerán las dos primeras componentes.

```
#Coger las dos primeras componentes
comps<-pcaT$x[,1:2]

#Correlacion entre las variables numericas y las de PCA
cor(train[,1:4], comps)
```

```
##                                PC1         PC2
## Temperature..K.          0.5255235  0.80841948
## Luminosity.L.Lo.         0.8679890 -0.01551463
## Radius.R.Ro.             0.7355718 -0.52545042
## Absolute.magnitude.Mv. -0.9067422  0.02742827
```

```
#Plot de las dos componentes mas importantes y de las relaciones
#con las variables
biplot(pcaT, main="Biplot de PC1 y PC2")
```



Observando la imagen obtenida con el biplot, se puede decir que el componente PC1 está relacionado con el tamaño de las estrellas.

Las dos variables horizontales son la magnitud absoluta y la luminosidad. Al tener direcciones opuestas, indica que estan inversamente correlacionadas. Se puede ver que cuanto menor es la estrella (cuanto menor es el valor de PC1) mayor magnitud absoluta y menor luminosidad tiene dicha estrella.

Como el valor de luminosidad de una estrella se calcula con la temperatura y el radio de las estrellas, mirando el biplot se puede entender que esas tres variables esten más relacionadas al tener un menor angulo entre ellas.

Una estrella más caliente que otra es mas luminosa teniendo estas el mismo radio. Por otro lado, una estrella más grande, es mas luminosa teniendo estas la misma temperatura.

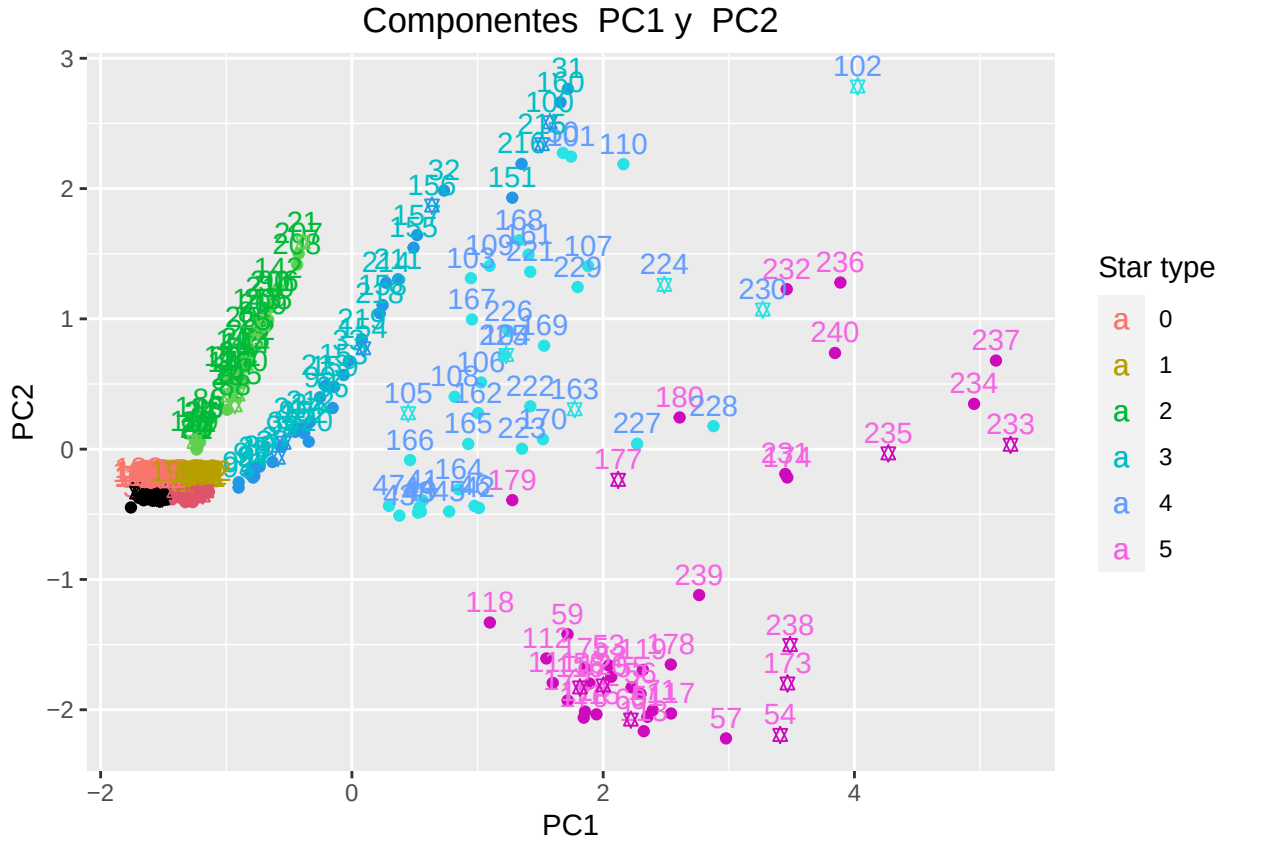
El componente PC2 está mas relacionado por lo tanto, con la temperatura primariamente, y algo menos con el radio.

Estrellas como la 234 y 237 tienen una buena proporción de temperatura y radio (temperaturas altas y gran radio), por ello la luminosidad es máxima.

La estrella 31 por ejemplo, tiene de las temperaturas más altas, pero al no tener un gran radio, no tiene mucha iluminación.

```
pcaTrain<-as.data.frame(pcaT$x)
testPrediction<-as.data.frame(predict(pcaT, test[,1:4]))

#Plot de la base de datos de train y las predichas
#de la parte de test. Los puntos son de train, y las estrellas de test.
ggplot() +
```

```
geom_point(data=train, mapping=aes(x=pcaTrain[, 1], y=pcaTrain[, 2]),
           color=as.character(train$Star.type+1), shape=19) +
xlab(colnames(pcaTrain)[1]) + ylab(colnames(pcaTrain)[2]) +
labs(color="Star type") +
geom_text(aes(x=pcaTrain[, 1], y=pcaTrain[, 2],label=row.names(train),
              color=as.character(train$Star.type)),
          hjust=0.5, vjust=-0.5)+
ggtitle(paste("Componentes ",colnames(pcaTrain)[1],"y ",colnames(pcaTrain)[2]))+
theme(plot.title = element_text(hjust = 0.5))+
#Añadiendo los puntos a predecir
geom_point(data=testPrediction, mapping=aes(x=testPrediction[, 1], y=testPrediction[, 2]),
           color=as.character(test$Star.type+1), shape=11) +
geom_text(aes(x=testPrediction[, 1], y=testPrediction[, 2],label=row.names(testPrediction),
              color=as.character(test$Star.type)),
          hjust=0.5, vjust=-0.5)
```



A la hora de realizar el PCA, el conjunto de datos se ha dividido en dos partes, entrenamiento y prueba o testeo (para predecir).

Como se puede observar en la imagen, hay dos tipos de iconos. Los puntos representan los casos de entrenamiento y las estrellas representan los valores predichos para los casos de prueba o test. Los valores de prueba parecen bastante bien representados en su propia clase.

Aparte de eso, se puede ver que se consigue una bastante buena diferenciación entre los 6 tipos de estrellas con las variables nuevas PC1 y PC2.

4. Clustering K-medoides

El segundo metodo que se va a utilizar es el agrupamiento de datos mediante el algoritmo k-medoides.

El clustering o agrupamiento de datos es la tarea de agrupar los datos o las instancias de manera que tengan algún tipo de similitud entre ellas para poder facilitar el estudio de los datos.

Se ha decidido utilizar, puesto que es un método muy útil y de gran importancia en el campo del análisis de datos. Ejemplos del uso de clustering pueden ser los sistemas de recomendación o la segmentación de imágenes.

En este caso, se van a utilizar los datos obtenidos por las componentes PC1 y PC2 para el agrupamiento de datos, puesto que se obtiene una dimensionalidad en la que las 6 clases están bastante bien representadas.

```
#Como los datos estan normalizados en el pca
#no haria falta normalizar de nuevo
dMat<-pcaT$x[,1:2]
#?dist
#?daisy
#Como para la clusterizacion se van a usar las componentes PC1 y PC2,
#para calcular las distancias se utilizara la distancia euclidea
dMat<-as.matrix(dist(dMat, method="euclidean"))
```

Una vez que se ha calculado la matriz de distancias mediante la distancia de Euclides, habría que conseguir el K o número de clusters óptimo.

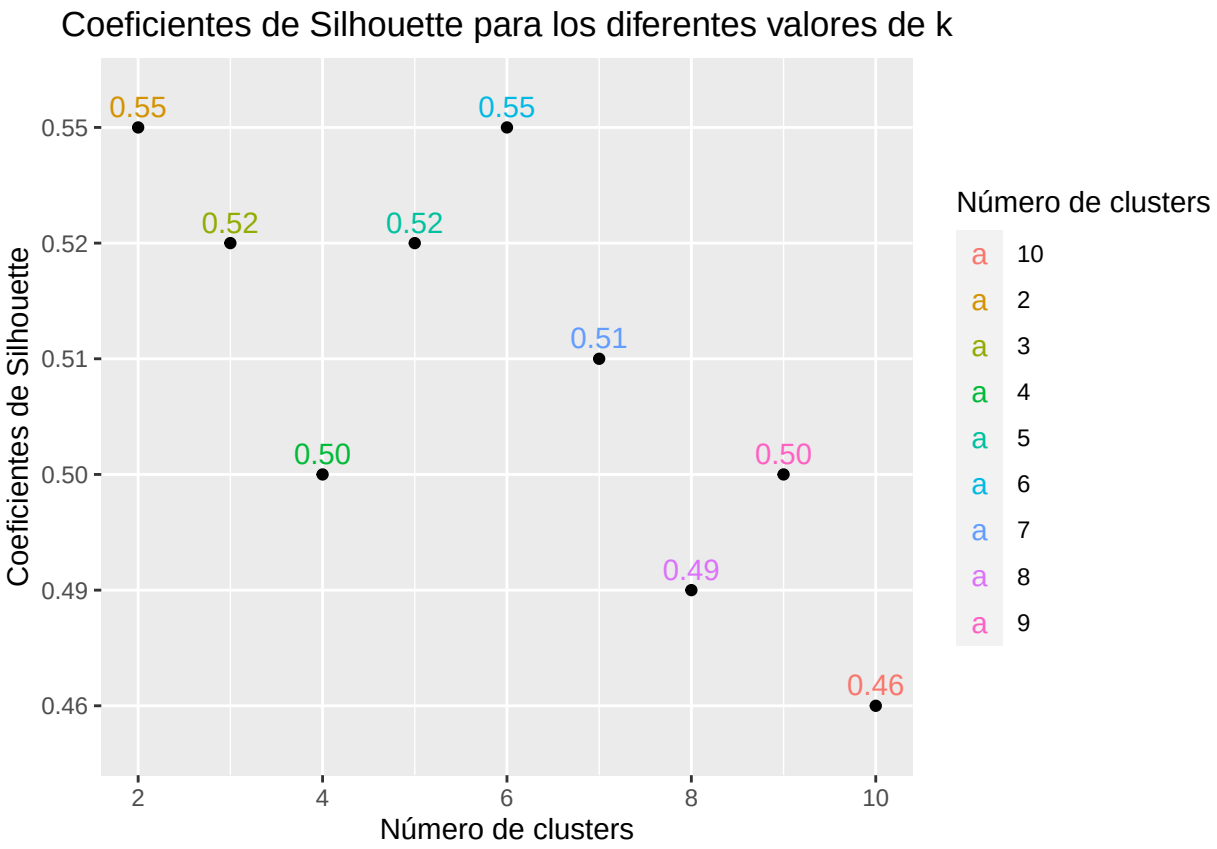
Para ello, se lleva a cabo la clusterización con diferentes valores de K, y se calcula el coeficiente de Silhouette para cada instancia.

El coeficiente de Silhouette es una métrica para evaluar la calidad del agrupamiento obtenido con algoritmos de clustering. En los algoritmos de aprendizaje no supervisado, la cantidad de grupos puede ser un parámetro de entrada del algoritmo o puede ser determinado automáticamente por el algoritmo. En el primer caso, el número óptimo de clusters se determina con alguna medida externa.

El coeficiente de silueta indica el número ideal de clusters. Un valor más alto de este índice indica una mejor partición de los datos, y por lo tanto, un mejor número de clusters.

```
#Medida de silhouette para saber cual seria el k o numero de clusters
#optimo
#?pam
#Lista con las medias de silhouette para saber que numero de clusters es mejor.
pamList<-c()
for(k in 2:10){
  #Clusterizar en k clusters y guardar el valor del cluster
  #al que pertenece cada instancia
  clusters<-pam(dMat, k)$clustering
  #Calcular el coeficiente de silhouette de los clusters obtenidos
  sil <- silhouette(clusters, dMat)
  #Guardar la media de todos los coeficientes obtenidos
  pamList<-c(pamList, mean(sil[, 3]))
}
#Pasar a data frame para visualizar en ggplot
pamList<-data.frame(kClusters=c(2:10),
                    Coeficientes=format(pamList,digits = 2, format = "f"))

ggplot(data=pamList, aes(x=pamList[,1],y=pamList[,2], group=1))+
```

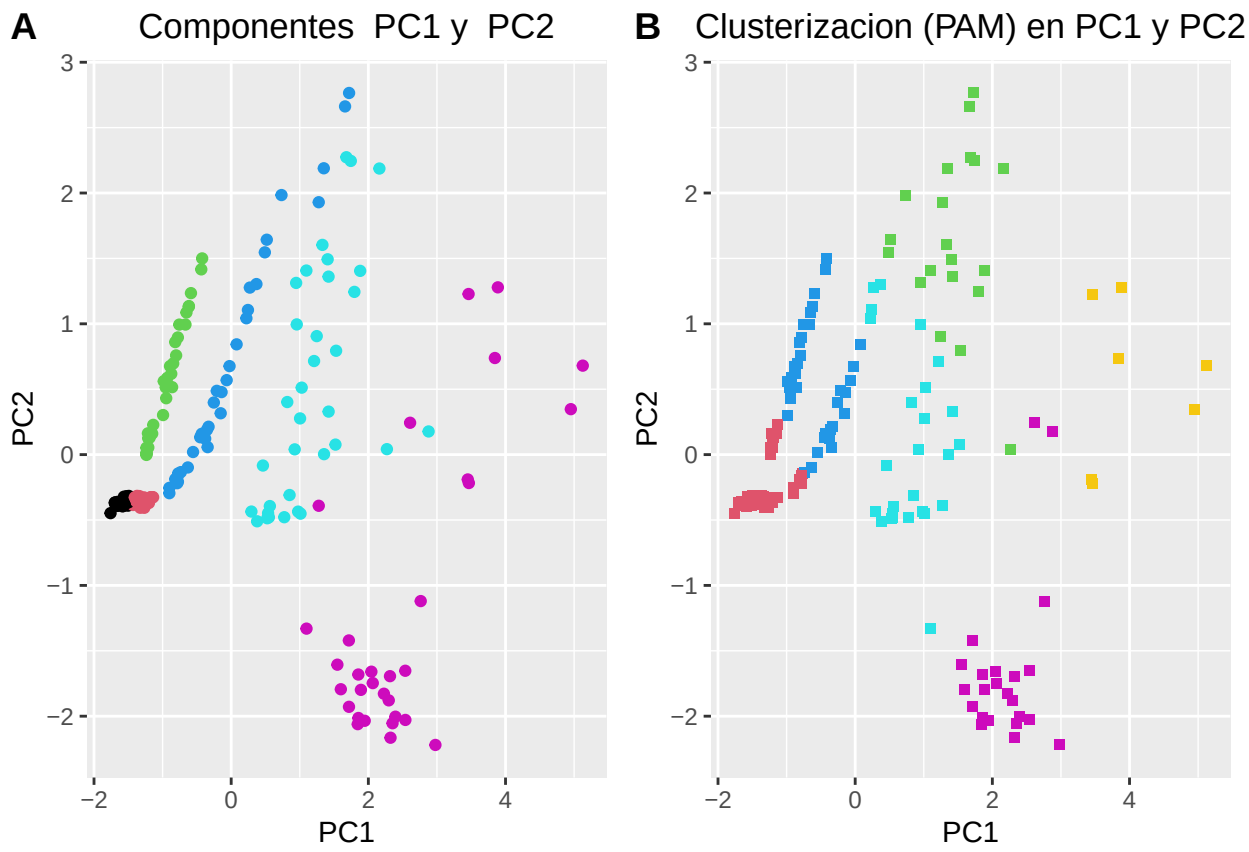
```
  geom_point()+
  geom_text(aes(x = pamList[,1],
                y = pamList[,2],
                label=pamList[,2],
                color=as.character(kClusters)),hjust=0.5, vjust=-0.5)+
  labs(color="Número de clusters")+
  xlab("Número de clusters")+ ylab("Coeficientes de Silhouette")+
  ggtitle("Coeficientes de Silhouette para los diferentes valores de k")+
  theme(plot.title = element_text(hjust = 0.5))
```



Como se puede observar en el gráfico de los coeficientes de Silhouette para los diferentes valores de K, los mejores agrupamientos se consiguen con 2 o 6 clusters En este caso, al saber previamente que los datos tienen 6 clases o tipos de estrellas diferentes, se va a utilizar el valor de 6 como número de clusters.

```
#Cluster de PCA con 6 clusters
pamC<-pam(dMat, 6)$clustering

#Unir los indices de cluster a los datos de pca para dibujar los resultados
pamDF<-as.data.frame(cbind(pcaT$x[,1:2],pamC))

#Grafico datos clusterizado
clustP<-ggplot() +
  geom_point(data=pamDF, mapping=aes(x=pamDF[, 1], y=pamDF[, 2]),
             color=as.character(pamDF[,3]+1), shape=15)+
  xlab(colnames(pcaTrain)[1]) + ylab(colnames(pcaTrain)[2]) +
  labs(color="Star type") +
```

```
  ggtitle(paste("Clusterizacion (PAM) en PC1 y PC2"))+
  theme(plot.title = element_text(hjust = 0.5))

#Grafico datos pca
pcaP<-ggplot() +
  geom_point(data=train, mapping=aes(x=pcaTrain[, 1], y=pcaTrain[, 2]),
             color=as.character(train$Star.type+1), shape=19) +
  xlab(colnames(pcaTrain)[1]) + ylab(colnames(pcaTrain)[2]) +
  labs(color="Star type") +
  ggtitle(paste("Componentes ",colnames(pcaTrain)[1],"y ",colnames(pcaTrain)[2]))+
  theme(plot.title = element_text(hjust = 0.5))

#Plots paralelos
plot_grid(pcaP, clustP, labels = "AUTO")
```




```
#Tabla de clasificación
table(pamDF[,3]-1, train[,5])
```

```
##
##      0  1  2  3  4  5
##   0 28 34 10  7  0  0
##   1  0  0  0  7 13  0
##   2  0  0 21 16  0  0
##   3  0  0  0  4 20  2
##   4  0  0  0  0  1 22
```

```
##   5  0  0  0  0  0  7
```

En las dos imágenes lado a lado se puede observar, primero (imagen A) como está la partición original en las componentes PC1 y PC2, y en la segunda imagen (B), como se han clusterizado los datos con el algoritmo de k-medoides en 6 clusters.

Como se puede observar, a veces la clusterización puede ser bastante complicada. Como es el caso de los grupos 0 y 1 (puntos negros y rojos) de las componentes PC1 y PC2. Al estar tan juntos entre ellos, a la hora de hacer la clusterización de los datos los clasifica a todos como si fuesen un único grupo. Sin embargo, hay que decir que si no se tuviese ese conocimiento previo de como están clasificados los datos realmente, probablemente se consideraría un buen resultado de agrupamiento de los datos (al menos, visualmente).

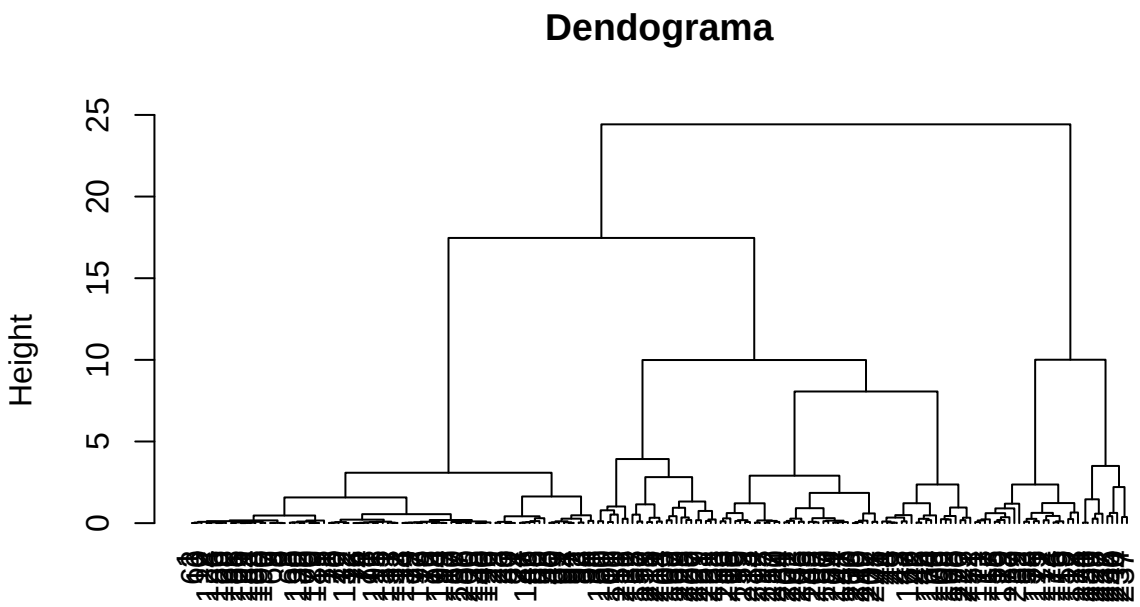
5. Clustering jerárquico

El siguiente método utilizado es el clustering jerárquico. El clustering jerárquico es un método de análisis de grupos puntuales, el cual busca construir una jerarquía de grupos.

En este caso, se ha utilizado el método de ward con la distancia euclídea (distancia más común para utilizar con el método de ward) para realizar la clusterización.

Se ha escogido el tipo de jerarquía aglomerativa (acercamiento ascendente), en el cual, cada observación comienza en su propio grupo, y los pares de grupos son mezclados mientras uno sube en la jerarquía.

```r
#Variables PC1 y PC2
dMat<-pcaT$x[,1:2]
#?agnes
#Metodo ward
hc<-agnes(dist(dMat), method="ward")

#Dendrograma del metodo ward
#?pltree
pltree(hc, hang=-1, main="Dendograma",sub = "", xlab="Estrellas")
```

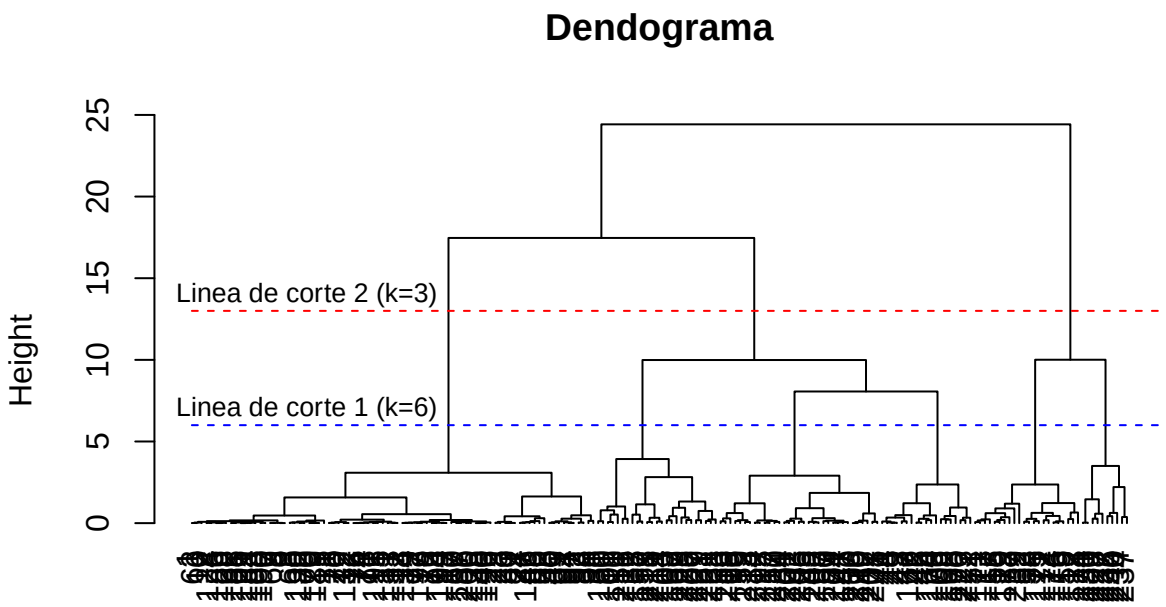


Al observar el dendograma, se puede ver que hay dos buenos puntos de corte. Cuanto mayor es la distancia entre dos niveles de la jerarquía, mayor es la distancia que se necesita para agrupar esos clusters, y por lo tanto, menos similares serán entre ellos.

El primer corte se podría realizar cuando todavía hay 6 grupos o clusters. Un segundo buen corte podría ser cuando hay 3 groups o clusters.

En la siguiente imagen se representan las dos líneas de corte.

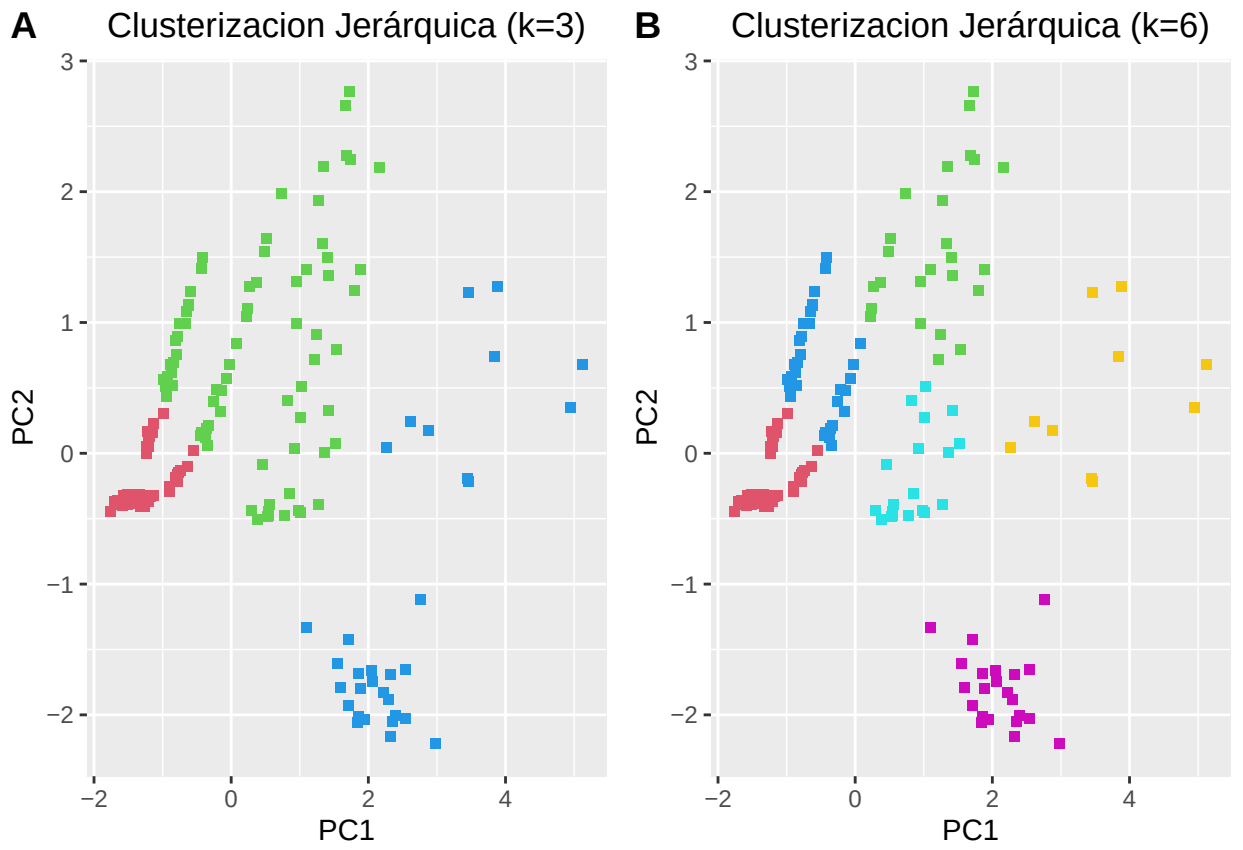
```
#Dibujar dendograma de nuevo, con las lineas de corte
pltree(hc, hang=-1, main="Dendograma", sub = "", xlab="Estrellas")
lines(x=rep(6,200), pch = 18, col = "blue", type = "l", lty = 2)
lines(x=rep(13,200), pch = 18, col = "red", type = "l", lty = 2)
#Coordenadas para los titulos de las lineas
#coords<-locator()
x1<-24.5
y1<-13.97
x2<-24.5
y2<-6.97
text(x1,y1, labels ="Linea de corte 2 (k=3)", cex=0.8)
text(x2,y2, labels ="Linea de corte 1 (k=6)", cex=0.8)
```



Para visualizar los datos, se va a realizar el clustering con 3 y con 6 para ver como quedarían los grupos dependiendo del número de clusters escogido.

```
#?cutree
#Se podria cortar por el punto de 3 o 6 clusters
wardC3<-cutree(hc, k=3)
wardC6<-cutree(hc, k=6)
#Guardar la particion de clusteres en dos data frames separados
ward3DF<-as.data.frame(cbind(pcaT$x[,1:2],wardC3))
ward6DF<-as.data.frame(cbind(pcaT$x[,1:2],wardC6))

#Grafica con 3 clusters
wardP3<-ggplot() +
  geom_point(data=ward3DF, mapping=aes(x=ward3DF[, 1], y=ward3DF[, 2]),
             color=as.character(ward3DF[,3]+1), shape=15)+
  xlab(colnames(pcaTrain)[1]) + ylab(colnames(pcaTrain)[2]) +
  labs(color="Star type") +
  ggtitle(paste("Clusterizacion Jerárquica (k=3)"))+
  theme(plot.title = element_text(hjust = 0.5))

#Grafica con 6 clusters
wardP6<-ggplot() +
  geom_point(data=ward6DF, mapping=aes(x=ward6DF[, 1], y=ward6DF[, 2]),
             color=as.character(ward6DF[,3]+1), shape=15)+
  xlab(colnames(pcaTrain)[1]) + ylab(colnames(pcaTrain)[2]) +
  labs(color="Star type") +
```

```
  ggtitle(paste("Clusterizacion Jerárquica (k=6)"))+
  theme(plot.title = element_text(hjust = 0.5))

#Plots paralelos
plot_grid(wardP3, wardP6, labels = "AUTO")
```




```
table(ward3DF[,3]-1, train[,5])
```

```
##
##       0  1  2  3  4  5
##   0 28 34 11 10  0  0
##   1  0  0 20 24 32  1
##   2  0  0  0  0  2 30
```

```
table(ward6DF[,3]-1, train[,5])
```

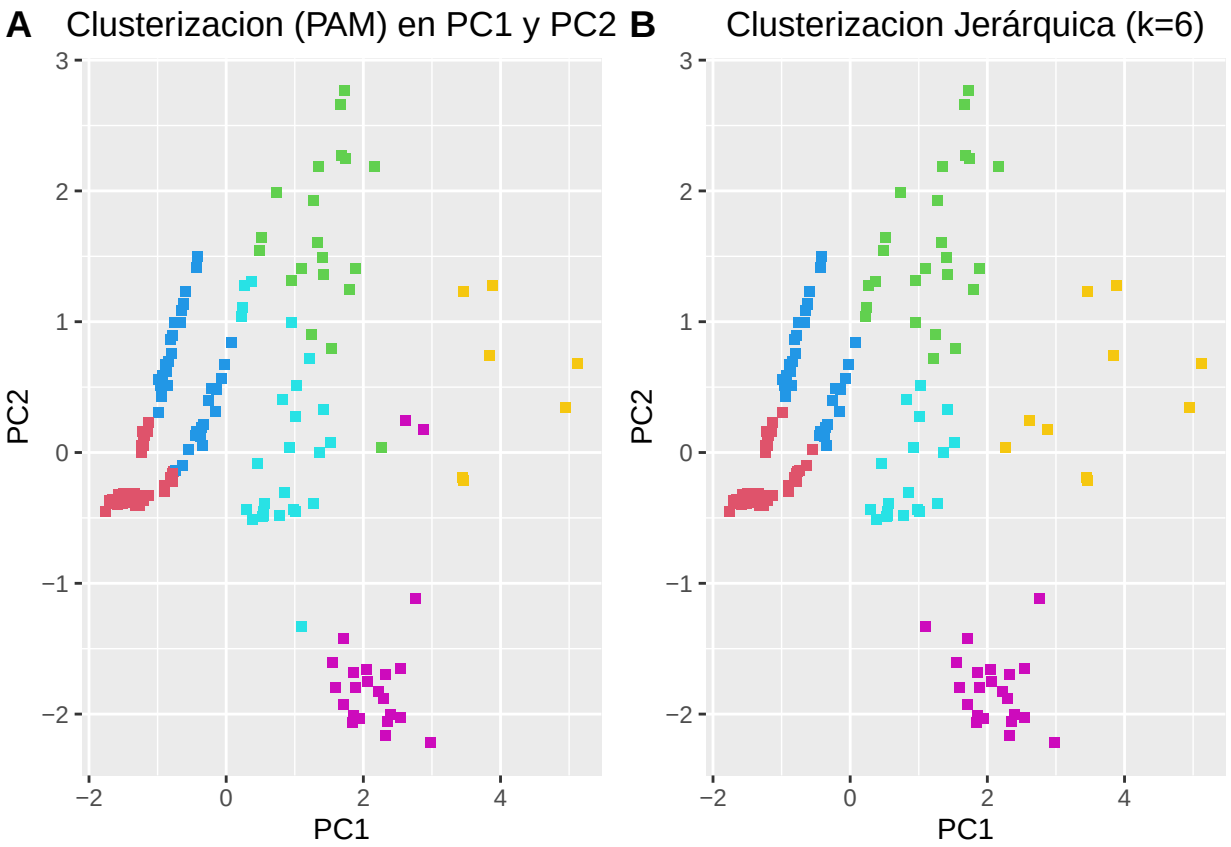
```
##
##       0  1  2  3  4  5
##   0 28 34 11 10  0  0
##   1  0  0  0 11 14  0
##   2  0  0 20 13  0  0
##   3  0  0  0  0 18  1
##   4  0  0  0  0  0 22
##   5  0  0  0  0  2  8
```

Observando los gráficos, se puede ver que cuando k tiene un valor de 6, se realiza un clustering parecido al obtenido con el algoritmo de k-medoides, pero a simple vista, parece que clasifica mejor los datos más distantes. La clase amarilla parece recoger mejor los datos Observando los gráficos, se puede ver que cuando k tiene un valor de 6, se realiza un clustering parecido al obtenido con el algoritmo de k-medoides, pero a simple vista, parece que clasifica mejor los datos mas distantes. La clase amarilla parece recoger mejor los datos mas distantes que con el método de k-medoides.

Al poner las dos imágenes lado a lado (k-medoides y jerárquica), se puede ver como se consigue una clasificación algo mas limpia con el método jerárquico (no hay puntos entre medias de otros grupos como pasa con en el algoritmo de k-medoides en las clases morada y amarilla o azul claro y verde). distantes que con el método de k-medoides.

Al poner las dos imagenes lado a lado (k-medoides y jerárquica), se puede ver como se consigue una clasificación algo mas limpia con el método jerárquico (no hay puntos entre medias de otros grupos como pasa con en el algoritmo de k-medoides en las clases morada y amarilla o azul claro y verde).

```
#Plots clusterizacion k-medoides y jerarquica
plot_grid(clustP, wardP6, labels = "AUTO")
```



Cuando k es tiene un valor de 3 con el método jerárquico, al no poder compararse con algún otro método no se pueden sacar muchas conclusiones. Lo único es que parece dividir bastante bien las estrellas en tres tamaños diferentes, pequeñas (puntos rojos: enanas marrones y rojas), medianas (puntos verdes: enanas blancas, secuencias principales y supergigantes) y grandes (puntos azules: hipergigantes).

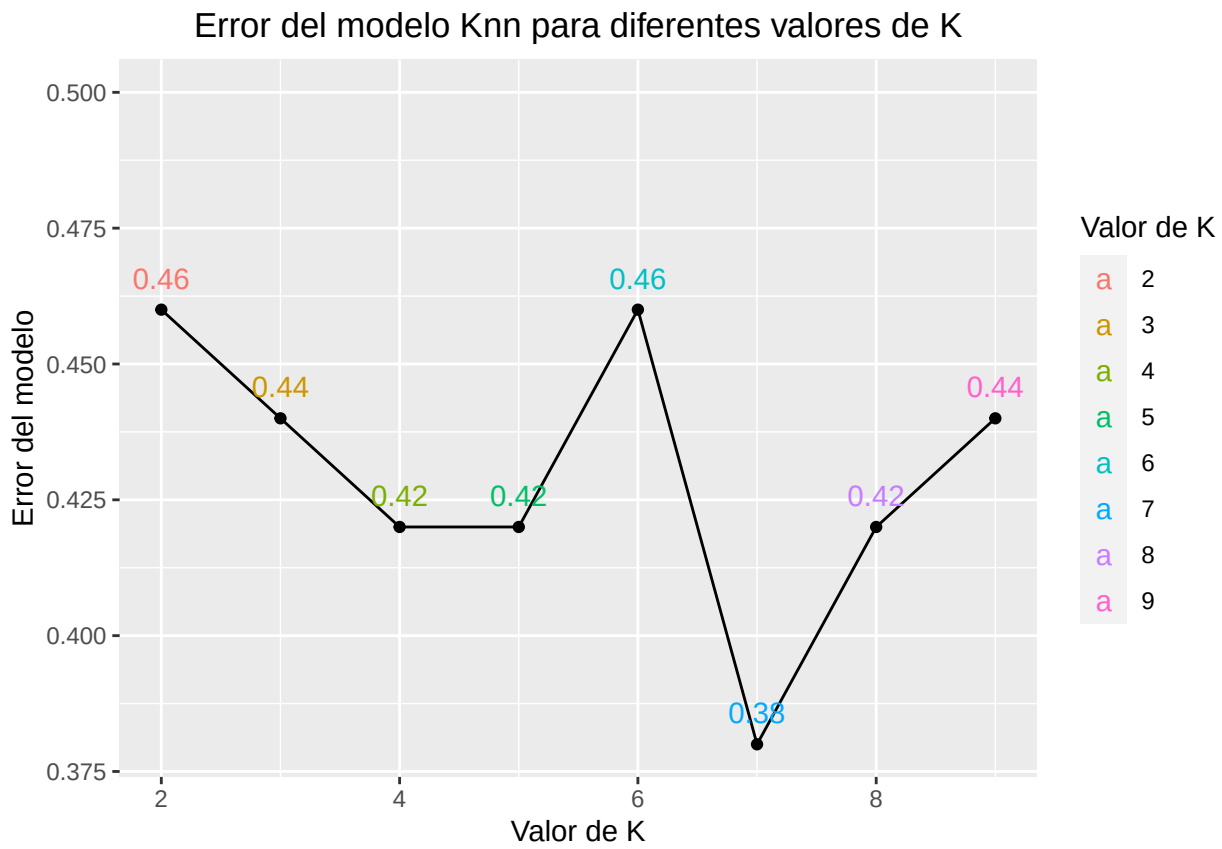
6. K-nearest neighbors

El k-nn es uno de los algoritmos de clasificación supervisada más sencillos, en el cual a un nuevo dato se le asignará la clase mayoritaria entre los K vecinos más cercanos.

El primer paso sería separar el conjunto de datos en dos grupos, entrenamiento (entrenar el modelo) y prueba (predicción). En este caso, se van a comparar las variables normales del conjunto de datos con las obtenidas mediante el PCA para ver cuál puede ayudar más a la hora de clasificar.

Una vez que la división de datos está hecha, el siguiente paso sería el de probar diferentes valores de K para poder seleccionar aquel valor que proporcione el menor error al modelo (estimación del error real).

```
#Indices de los casos de test
testIndex<-as.numeric(row.names(test))
#Vector de errores nulos
error<-c()
for (i in 2:9){
  #Clases predichas por el knn para k=i
  knnM<-knn(train[,1:4], test[,1:4], cl=train[,5], k=i)
  #Tabla de las clases (aciertos)
  (kTable<-table(dataNum[testIndex,5], knnM, dnn = c("Star type","Knn star type")))
  #Calcular el error, 1- (suma diagonal)/(suma total)
  error<-c(error,1-sum(diag(kTable))/sum(kTable))
}
#Lista de errores a data frame para visualizar en ggplot
errorD<-data.frame(ValorK=c(2:9),
                   Error=as.numeric(formatC(error,digits = 2, format = "f")))

#Grafo de errores para diferentes valores de K
ggplot(data=errorD,aes(x=errorD[,1],y=errorD[,2], group=1))+
  geom_line()+geom_point()+
  geom_text(aes(x = errorD[,1],
                y = errorD[,2],
                label=errorD[,2],
                color=as.character(ValorK)),hjust=0.5, vjust=-1)+
  labs(color="Valor de K")+ ylim(0.38,0.5)+
  xlab("Valor de K")+ ylab("Error del modelo")+
  ggtitle("Error del modelo Knn para diferentes valores de K")+
  theme(plot.title = element_text(hjust = 0.5))
```



Como se puede observar en la gráfica obtenida, el menor error (usando las variables normales) se consigue cuando K tiene un valor de 7 (error de 0,38). Por lo tanto, ese va a ser el valor seleccionado de K.

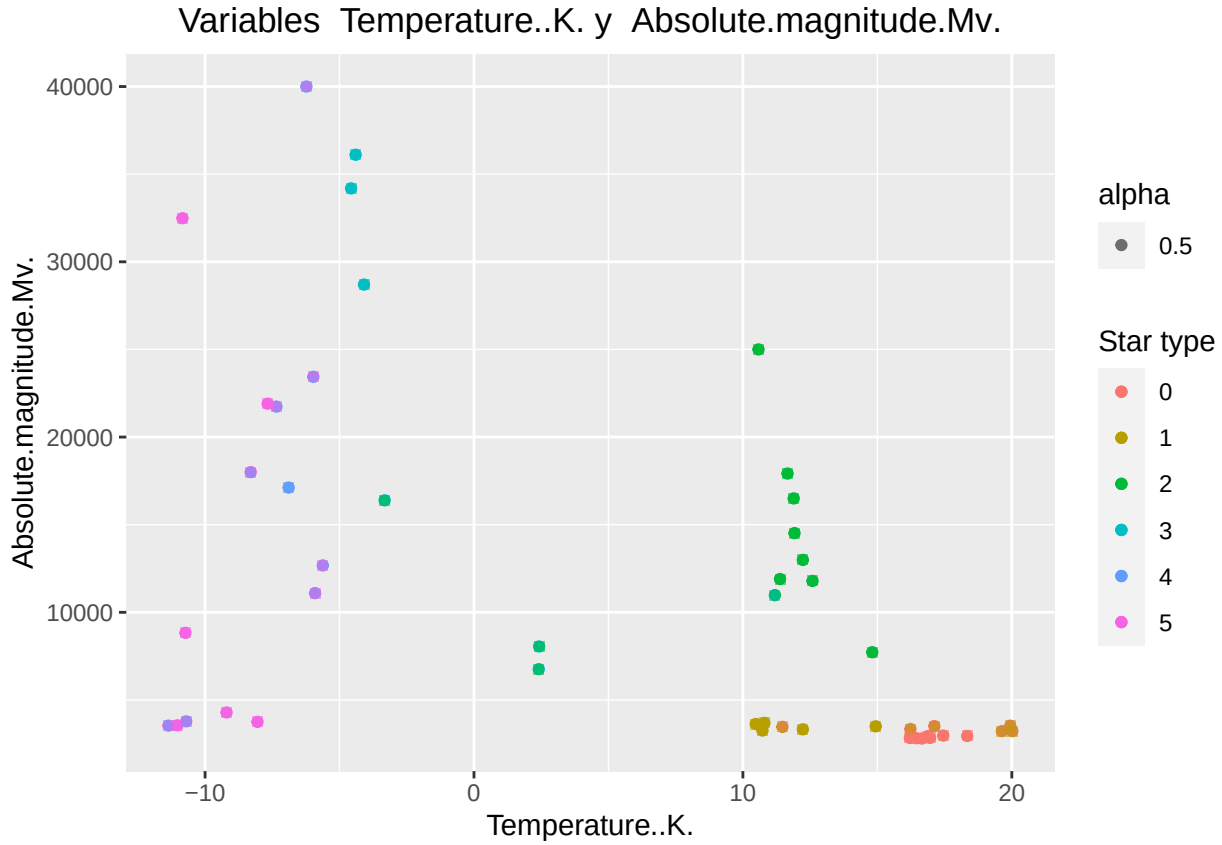
```
#K=7 el error minimo
k<-errorD[which(errorD[,2]==min(errorD[,2])),1]
#Volver a crear el modelo con k=7
knnM<-knn(train[,1:4], test[,1:4], cl=train[,5], k=k)
#Tabla de clasificados
(kTable<-table(dataNum[testIndex,5], knnM, dnn = c("Star type","Knn star type")))
```

```
##           Knn star type
## Star type 0 1 2 3 4 5
##         0 7 5 0 0 0 0
##         1 1 5 0 0 0 0
##         2 0 0 8 1 0 0
##         3 0 0 3 3 0 0
##         4 0 0 0 0 1 5
##         5 0 0 0 0 3 6
```

```
#Porcentaje de aciertos
accuracy<-sum(diag(kTable))/sum(kTable)*100
print(paste("El porcentaje de bien clasificados es: ",accuracy,"%",sep=""))
```

```
## [1] "El porcentaje de bien clasificados es: 62.5%"
```

```
ggplot(data = dataNum[testIndex,]) +
        geom_point(mapping = aes(x = dataNum[testIndex,4], y = dataNum[testIndex,1],
                                 color = as.character(Star.type))) +
        xlab(colnames(dataNum)[1]) + ylab(colnames(dataNum)[4]) +
        labs(color="Star type") +
        ggtitle(paste("Variables ",colnames(dataNum)[1],"y ",colnames(dataNum)[4]))+
        theme(plot.title = element_text(hjust = 0.5))+
        geom_point(mapping = aes(x = dataNum[testIndex,4], y = dataNum[testIndex,1],
                                 color = as.character(knnM), shape=15, alpha=0.5))+
        scale_shape_identity()
```

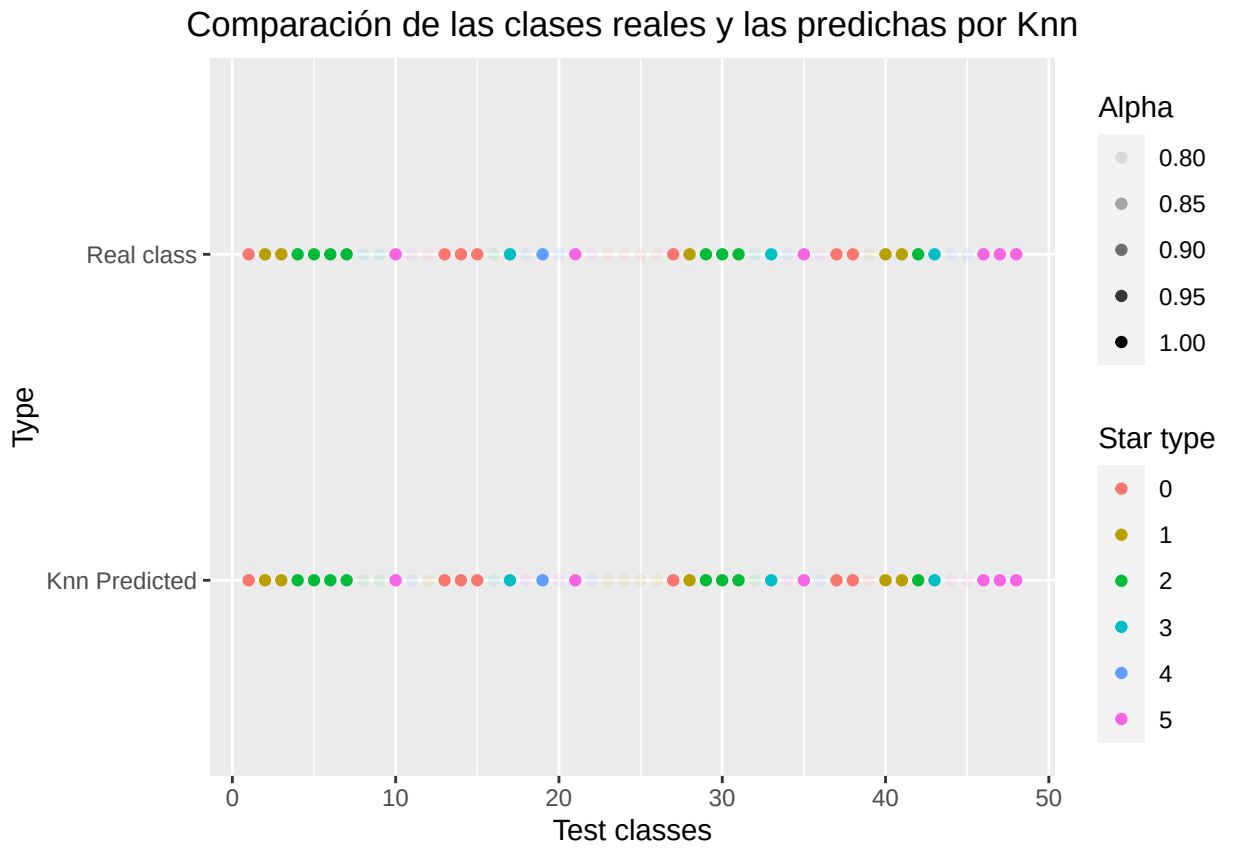


Como se puede observar en la tabla y algo menos en el gráfico (en el cual, los colores de los puntos indican las clases reales y los de los cuadrados, las clases predichas), la mayoría de los datos de las clases 0,3 y 4 están bien clasificados. Pero no se obtiene un gran porcentaje de aciertos.

En el siguiente gráfico se pueden ver las clases reales y predichas en dos niveles diferentes para facilitar las diferencias visualmente.

```
#Juntar vector de clases predichas y reales para plotear
clases<-c(as.character(knnM),as.character(test[,5]))
#Crear los dos tipos de clase, las reales y las predichas, para plotear dos lineas
knnC<-rep("Knn Predicted", length(knnM))
realC<-rep("Real class", length(knnM))
#Juntarlas
typeC<-c(knnC,realC)
#Posiciones de las instancias, de 1 a 48 (una fila arriba y otra abajo)
```

```r
position<-c(seq(1,48), seq(1,48))
#Valores alpha, si acierta (predicha==real), alpha=1
alphaC<-ifelse(knnM==test[,5],1,0.8)
predicted<-data.frame(Clases=clases, Tipo=typeC, Position=position,
                      Alpha=c(alphaC, alphaC))

#Grafico de las clases reales y predichas
ggplot(data = predicted) +
    geom_point(mapping = aes(x = Position,
                             y = Tipo,
                             color = as.character(Clases),
                             alpha = Alpha)) +
    xlab("Test classes") + ylab("Type") +
    labs(color="Star type") +
    ggtitle("Comparación de las clases reales y las predichas por Knn")+
    theme(plot.title = element_text(hjust = 0.5))
```




```r
#Resaltados los aciertos con un valor de alpha de 1 y los
#valores que han fallado, con un valor alpha de 0.8
```

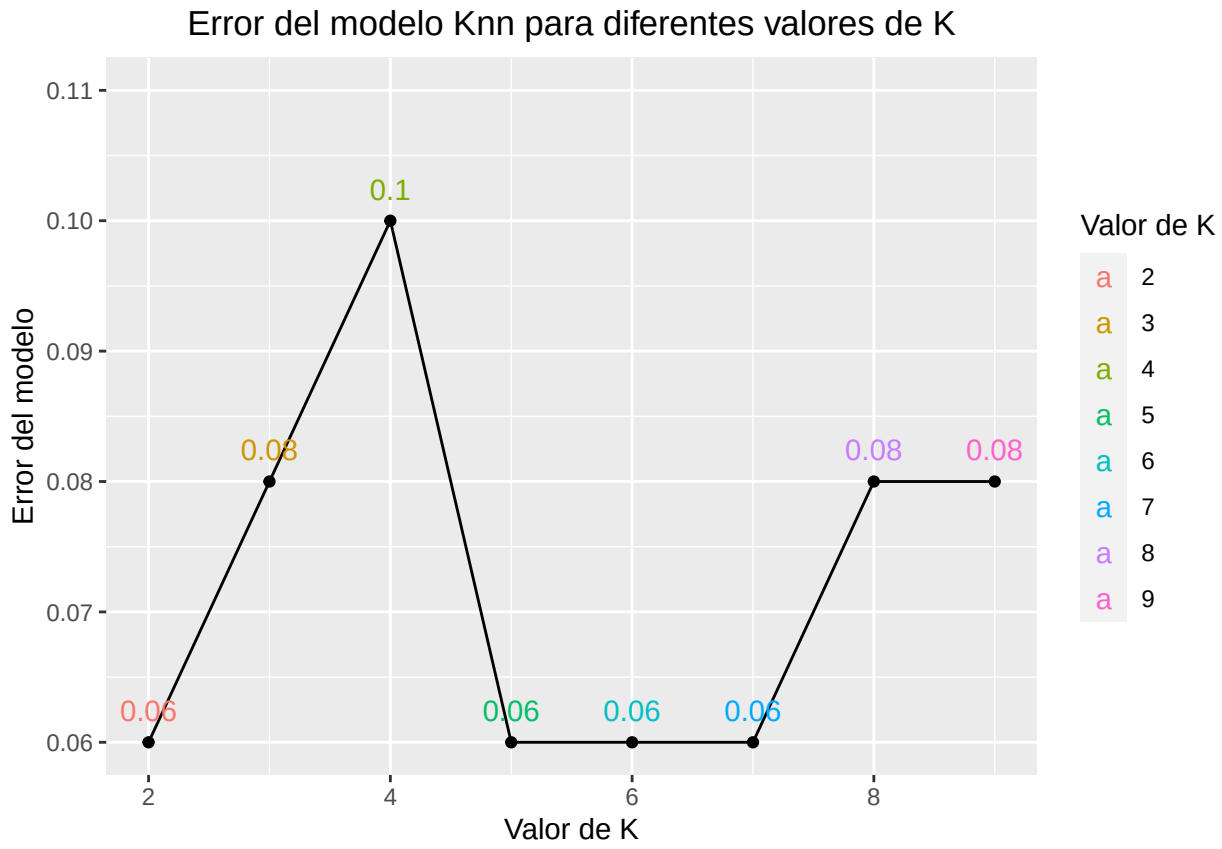
Los puntos con un valor de alfa 1 indican que han sido correctamente clasificados, mientras que los puntos con un valor de alfa menor, indican que no han sido bien clasificados.

Aunque no se consigue un gran porcentaje de aciertos mediante este algoritmo, hay que decir que el conjunto de datos es bastante reducido y que hay 6 clases para clasificar, lo cual puede dificultar al modelo. Dicho

esto, y que el k-nn es de los algoritmos de clasificación más sencillos (al ser más complejos seguramente clasifiquen mejor en la mayoréa de los casos), no se consiguen tan malos resultados (un 62,5 % de aciertos parece bastante aceptable).

Una vez terminada la clasificación del modelo entrenado con las variables normales, se va a probar con las variables obtenidas mediante el algoritmo PCA (variables PC1 y PC2) para ver cuál sería la diferencia, puesto que en las variables de PCA se pueden distinguir las clases o tipos de estrellas bastante fácilmente.

```
#Vector de errores para los casos con pca
errorpca<-c()
for (i in 2:9){
  #Clases predichas por el knn para k=i
  knnM<-knn(pcaTrain[,1:2], testPrediction[,1:2], cl=train[,5], k=i)
  #Tabla de clasificacion (aciertos)
  (kTable<-table(dataNum[testIndex,5], knnM, dnn = c("Star type","Knn star type")))
  #Calcular el error, 1- (suma diagonal)/(suma total)
  errorpca<-c(errorpca,1-sum(diag(kTable))/sum(kTable))
}

#Pasar lista de errores a data frame para visualizar
errorP<-data.frame(ValorK=c(2:9),
                   Error=as.numeric(formatC(errorpca
                                            ,digits = 2, format = "f")))

#Grafo de errores para diferentes valores de K (con componentes PCA)
ggplot(data=errorP,aes(x=errorP[,1],y=errorP[,2], group=1))+
  geom_line()+geom_point()+
  geom_text(aes(x = errorP[,1],
                y = errorP[,2],
                label=errorP[,2],
                color=as.character(ValorK)),hjust=0.5, vjust=-1)+
  labs(color="Valor de K")+ ylim(0.06,0.11)+
  xlab("Valor de K")+ ylab("Error del modelo")+
  ggtitle("Error del modelo Knn para diferentes valores de K")+
  theme(plot.title = element_text(hjust = 0.5))
```



Como se puede ver en la gráfica obtenida, los errores son mucho menores en todos los casos cuando el modelo se entrena con las variables de PCA. El error mínimo del modelo se consigue para los valores 2,5,6 y 7 de K. En este caso, se va a escoger el valor 7 para K para hacer la comparativa.

```
#K=2,5,6,7 el error minimo
#utilizar mismo k=7 para comparar con el de variables normales
k<-7
#Volver a crear el modelo con k=7
knnM<-knn(pcaTrain[,1:2], testPrediction[,1:2], cl=train[,5], k=k)
#Tabla de clasificacion
(kTable<-table(dataNum[testIndex,5], knnM, dnn = c("Star type","Knn star type")))
```

```
##           Knn star type
## Star type  0  1  2  3  4  5
##         0 12  0  0  0  0  0
##         1  0  6  0  0  0  0
##         2  0  0  9  0  0  0
##         3  0  0  0  6  0  0
##         4  0  0  0  0  4  2
##         5  0  0  0  0  1  8
```

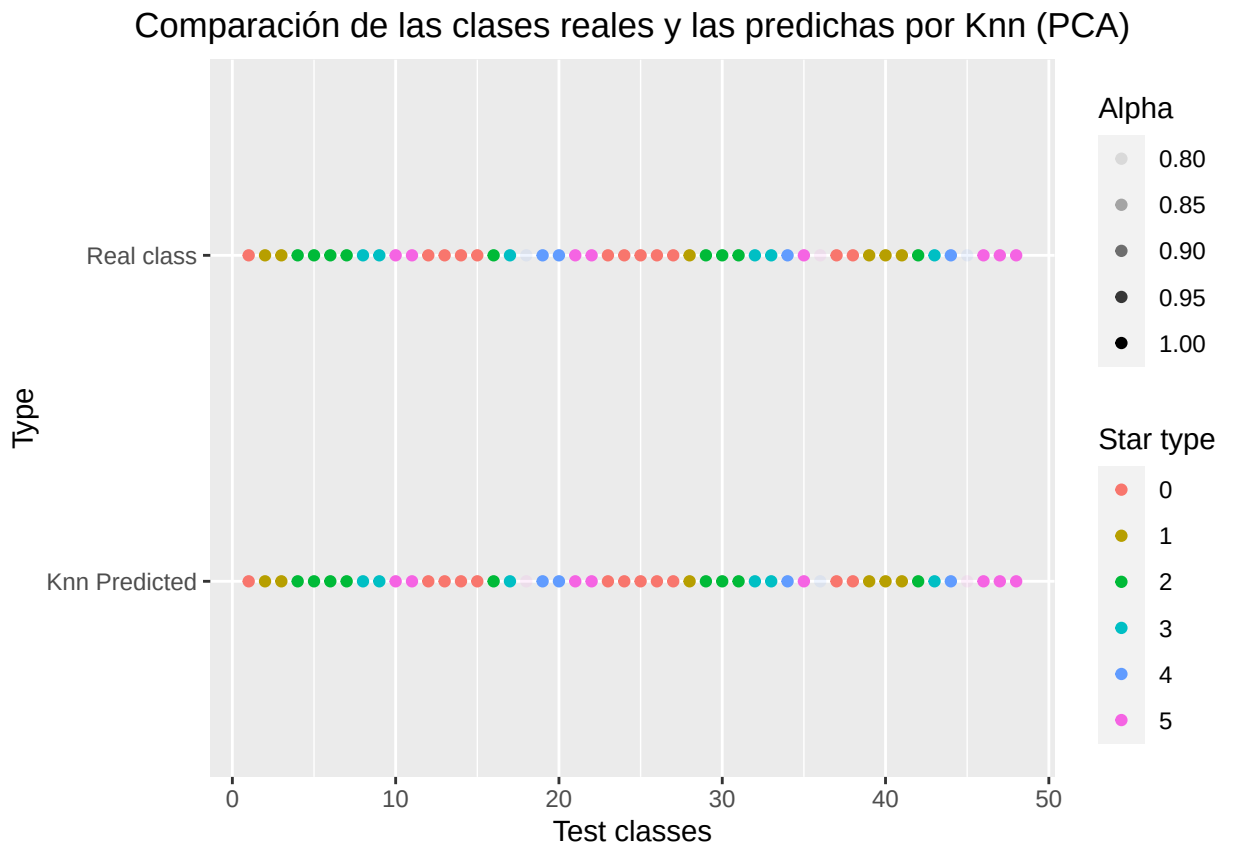
```
#Calcular porcentaje de aciertos
accuracy<-sum(diag(kTable))/sum(kTable)*100
print(paste("El porcentaje de bien clasificados es: ",accuracy,"%",sep=""))
```

```
## [1] "El porcentaje de bien clasificados es: 93.75%"
```

Como se puede ver en la tabla, se consigue un porcentaje de aciertos mucho más alto que con las variables normales (93,75 %).

En el siguiente gráfico se pueden ver como se han clasificado las clases con el modelo nuevo.

```
#Juntar las clases predichas y reales
clases<-c(as.character(knnM),as.character(test[,5]))
#Dos clases, predichas y reales
knnC<-rep("Knn Predicted", length(knnM))
realC<-rep("Real class", length(knnM))
#Juntar los dos tipos
typeC<-c(knnC,realC)
#Posiciones para las instancias, de 1 a 48 (dos lineas, segun el tipo)
position<-c(seq(1,48), seq(1,48))
#Si la clase predicha=real (acierto), alpha=1
alphaC<-ifelse(knnM==test[,5],1,0.8)
#Data frame para visualizar en ggplot
predicted<-data.frame(Clases=clases, Tipo=typeC, Position=position,
                      Alpha=c(alphaC, alphaC))

#Grafico de las clases reales y predichas por knn (variables pca)
ggplot(data = predicted) +
  geom_point(mapping = aes(x = Position,
                           y = Tipo,
                           color = as.character(Clases),
                           alpha = Alpha)) +
  xlab("Test classes") + ylab("Type") +
  labs(color="Star type") +
  ggtitle("Comparación de las clases reales y las predichas por Knn (PCA)")+
  theme(plot.title = element_text(hjust = 0.5))
```



Se puede ver como solo tres instancias han sido mal clasificadas. Se puede concluir que al tener una diferenciación de las clases más clara en las variables PC1 y PC2, se obtiene una bastante mejor clasificación que con las variables normales.

7. Local Outlier Factor (LOF)

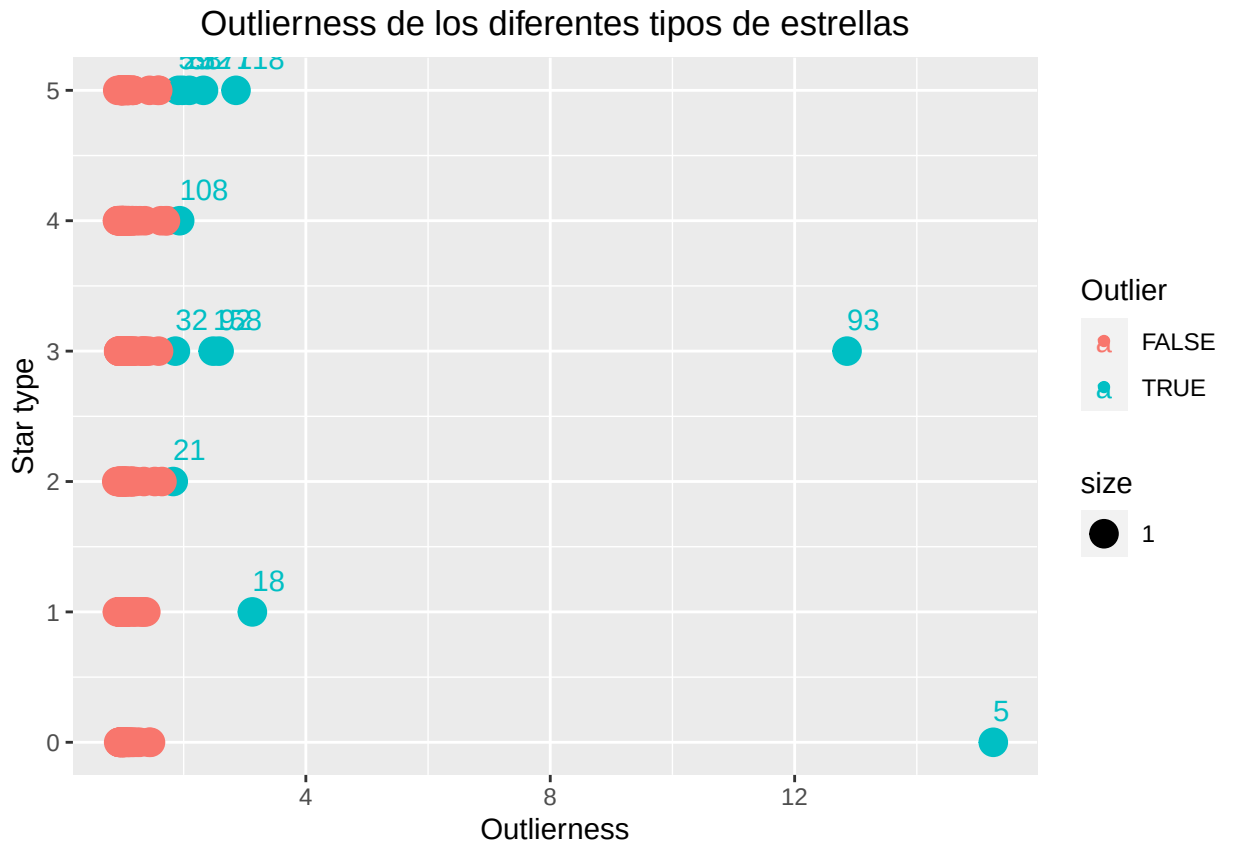
El local outlier factor es un algoritmo usado para la detección de anomalías, el cual se basa en un concepto de densidad local, donde la localidad viene dada por los K vecinos más cercanos, cuya distancia se usa para estimar la densidad. Al comparar la densidad local de un objeto con las densidades locales de sus vecinos, se pueden identificar regiones de densidad similar y puntos que tienen una densidad sustancialmente más baja que sus vecinos. Estos se consideran valores atípicos.

Para la detección de anomalías se van a usar las variables originales del conjunto de datos (por variar un poco y no implementar todos los algoritmos con las variables de PCA, aunque tendría más sentido utilizarlo con las componentes de PCA), se va a utilizar un valor de K=5, y van a ser considerados outliers aquellas instancias con un valor de "outlierness" (cuán de atípico es una instancia respecto a los K vecinos más cercanos) mayor a 1,75. Al ser un valor escogido a mano, puede variar bastante lo que se clasifica como outlier y lo que no.

```
#Guardar la matriz con las variables numericas
Stars<-dataNum
#Vector de strings nulos, para visualizar texto de los outliers solo
#si no es un outlier, asignar vector nulo de texto
fake<-rep("", dim(data)[1])
```

```r
#Pasar como parametros las columnas numericas, no la de la clase
#?LOF
Stars$outlierness<-LOF(dataNum[,1:4], k=5)
#sort(Stars$outlierness)
#print(which(Stars$outlierness>1.75))

#Crear columna nueva que dice si es o no un outlier dependiendo del
#valor de outlierness obtenido
Stars$outlier <-(Stars$outlierness>1.75)
#Asignar color, si es outlier rojo y sino negro
Stars$OutlierColor <- ifelse(Stars$outlier, "red", "black")

#Histograma de los valores de outlierness
hist(Stars$outlierness, main=paste("Histograma de outlierness de las estrellas"))
```

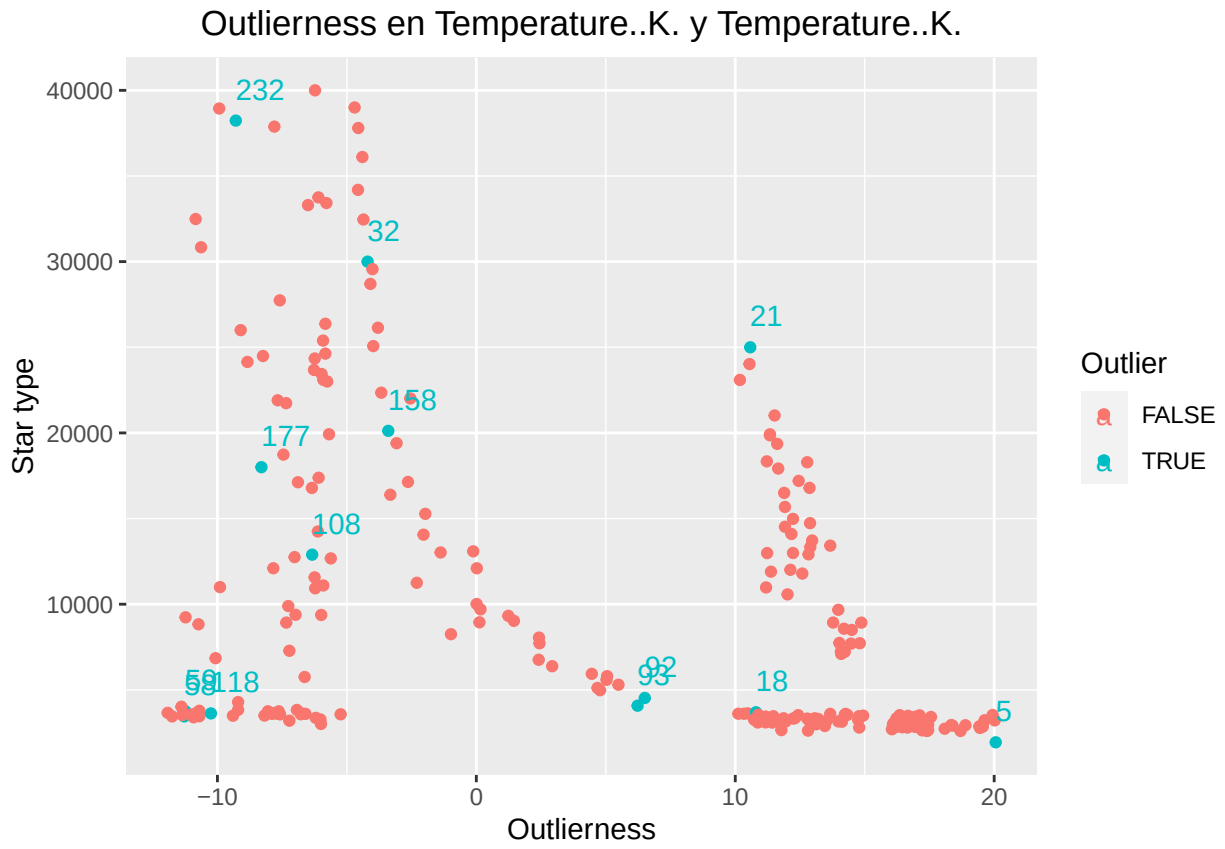

```r
#Scater plot de cada tipo de estrella y su outlierness
ggplot(data = Stars) +
        geom_point(mapping = aes(x = Stars[,8], y = Stars[,5], color = outlier, size=1)) +
        xlab("Outlierness") + ylab("Star type") +
        labs(title=paste("Outlierness de los diferentes tipos de estrellas"), color="Outlier") +
        theme(plot.title = element_text(hjust = 0.5)) +
        #Poner texto a los que sean outliers solo
        geom_text(aes(x = Stars[,8], y = Stars[,5],
                      label=ifelse(Stars[,9]==TRUE ,row.names(Stars), fake),
                      color=outlier),hjust=0, vjust=-1)
```



En el gráfico se pueden observar las instancias que han sido detectadas como outliers (outliers de color azul, datos normales de color rojo) en las diferentes clases del conjunto de datos.

Al ser un conjunto de datos con una dimensionalidad mayor a la que se puede representar gráficamente (4 dimensiones o variables en este caso), si se intentan plasmar los resultados en una dimensionalidad menor, no se pueden ver los resultados tan claramente. Como puede ser en el siguiente caso, utilizando las variables de temperatura y magnitud absoluta (las variables numéricas con mejor diferenciación entre las clases) para ver los valores anómalos, no se consigue una buena representación de estos (los puntos anómalos no se diferencian tanto de los demás puntos en estas dimensiones, muchos de esos puntos están bastante bien agrupados).

```
#Grafica de variables de temperatura y magnitud absoluta
#para visualizar los outliers
ggplot(data = Stars) +
  geom_point(mapping = aes(x = Stars[,4], y = Stars[,1], color = outlier)) +
  xlab("Outlierness") + ylab("Star type") +
  labs(title=paste("Outlierness en",colnames(Stars)[1],"y",colnames(Stars)[1]), color="Outlier") +
  theme(plot.title = element_text(hjust = 0.5)) +
  #Poner texto a los que sean outliers solo
  geom_text(aes(x = Stars[,4], y = Stars[,1],
                label=ifelse(Stars[,9]==TRUE ,row.names(Stars), fake),
                color=outlier),hjust=0, vjust=-1)
```



Las instancias que han sido detectadas como anómalas no van a ser eliminadas, puesto que el conjunto de datos ya es bastante reducido.

Se ha decidido implementar este algoritmo al final para aprender sobre la detección de anomalías, y no con la intención de facilitar los resultados de los demás algoritmos.

8. Conclusiones

Como se ha visto en el análisis llevado a cabo, hay conjuntos de variables más representativas de los tipos de estrellas que otros (como pueden ser la temperatura y magnitud absoluta).

Al tener múltiples variables, se ha decidido reducir la dimensionalidad con el algoritmo de PCA, el cual en este caso, ha facilitado la diferenciación de las seis clases, y por lo tanto, ha mejorado la clasificación del modelo de k-nn.

Sobre esas nuevas variables, se ha llevado a cabo la clusterización por K-medoides y jerárquica. Mediante estas técnicas, se ha podido observar como se han clasificado los datos de manera no supervisada.

Se ha llevado a cabo la detección de anomalías con el algoritmo LOF. Mediante este algoritmo se han podido visualizar las distintas instancias que han sido clasificadas como anómalas para cada clase. Con ello, se podrían haber eliminado las instancias que podrían dificultar la clusterización o la clasificación de datos, sin embargo, se ha decidido dejar las instancias anómalas, puesto que el conjunto de datos ya era bastante reducido.

Como se ha podido observar, gracias a las variables obtenidas mediante PCA, se ha conseguido un mejor porcentaje de clasificación que con las variables normales para el k-nn. El cual podría servir para clasificar futuras estrellas teniendo los datos de estas. Por lo tanto, se podría decir que se ha cumplido el objetivo para el cual se ha creado este conjunto de datos.

9. Lista de librerias

En el siguiente fragmento de código se pueden observar todas las librerías que se han utilizado para la realización del proyecto.

```
library(caret)
library(mlbench)
library(ggplot2)
library(reshape2)
library(CatEncoders)
library(tidyverse)
library(caTools)
library(cowplot)
library(cluster)
library(class)
library(DDoutlier)
```

10. Bibliografía

[1] C. Croux and C. Dehon. "Influence functions of the Spearman and Kendall correlation measures". In: *Statistical methods & applications* 19.4 (2010), pp. 497-515.

[2] S. M. Holland. "Principal components analysis (PCA)". In: *Department of Geology, University of Georgia, Athens, GA* (2008), pp. 30602-2501.

[3] C. Yuan and H. Yang. "Research on K-value selection method of K-means clustering algorithm". In: *Jâ€"Multidisciplinary Scientific Journal* 2.2 (2019), pp. 226-235.