Apuntes Componentes Principales

Consoli Quintana

2022-10-24

Tema 2. Componentes principales:

Vamos a borrar todo lo que tengamos en el Environment:

```
rm(list = ls())
```

Para importar los datos, ejecutaremos el código:

```
library(readxl)

## Warning: package 'readxl' was built under R version 4.2.1

eolicos <- read_excel("eolica_60.xlsx", sheet = "Datos")
```

En el *Environment*, ya aparece un objeto: - una estructura de datos tipo data frame. - Nobmre: "eolicos". Contiene 12 columnas, una por cada una de las variables almacenadas en el archivo Excel. - Si desplegamos `eolicos` se compone de 3 `chr`(caracteres, atributos o factores).

Pregunta: ¿A qué variables corresponden?.

Al igual que en la sesión anterior: R considera la *primera columna (nombre)* como *VARIABLE*.

En realidad, no es una variable, sino el nombre de los casos (empresas). Para evitar que R tome los nombres de los individuos como una variable, podemos redefinir nuestro data frame diciéndole que tome esa primera columna como los nombres de los individuos o casos (filas):

```
eolicos <- data.frame(eolicos, row.names = 1)
```

Vemos que: - Ya no aparece *NOMBRE* como variable, y en el Environment ya aparece el data frame "eolicos" con *100 observaciones*, pero con *11 variables* (una menos).

El análisis de componentes principales (ACP) solo se puede efectuar entre variables en escala métrica, por lo que, para seleccionar solo aquellas con tal naturaleza (todas, salvo MATRIZ y DIMENSION, que son las dos últimas columnas del data frame "eolicos"), crearemos un nuevo data frame llamado "originales":

Script de MAT:

```
library(dplyr)

## Warning: package 'dplyr' was built under R version 4.2.1
```

```
##
## 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

originales <- eolicos %>% select(-MATRIZ,-DIMENSION)
summary (originales)

##       RES              ACTIVO            FPIOS              RENECO
##  Min.   :-5661   Min.   : 24944   Min.   :-77533   Min.   :-2.813
##  1st Qu.:  496   1st Qu.: 33101   1st Qu.:  2323   1st Qu.: 1.363
##  Median : 1939   Median : 41995   Median :  9727   Median : 4.237
##  Mean   : 2699   Mean   : 74180   Mean   : 18801   Mean   : 5.909
##  3rd Qu.: 3903   3rd Qu.: 68628   3rd Qu.: 26493   3rd Qu.: 8.620
##  Max.   :17026   Max.   :303904   Max.   :177707   Max.   :35.262
##  NA's   :1       NA's   :1        NA's   :1        NA's   :1
##      RENFIN            LIQUIDEZ            MARGEN           SOLVENCIA
##  Min.   :-359.77   Min.   :  0.0290   Min.   :-302.03   Min.   :-40.74
##  1st Qu.:   2.91   1st Qu.:  0.5500   1st Qu.:  11.70   1st Qu.:  5.94
##  Median :  14.14   Median :  0.9405   Median :  27.64   Median : 16.88
##  Mean   :  11.90   Mean   :  3.6208   Mean   :  30.63   Mean   : 28.12
##  3rd Qu.:  38.07   3rd Qu.:  1.5733   3rd Qu.:  39.59   3rd Qu.: 51.38
##  Max.   : 150.42   Max.   :128.4330   Max.   : 400.90   Max.   : 99.08
##                                       NA's   :2         NA's   :1
##     APALANCA
##  Min.   :-7770.00
##  1st Qu.:   12.28
##  Median :  134.90
##  Mean   :  333.21
##  3rd Qu.:  651.93
##  Max.   : 8049.39
##
```

Otra posibilidad:

```
originales2 <- eolicos %>% select(where(is.numeric))
```

Lo comprobamos:

```
summary(originales2)

##       RES              ACTIVO            FPIOS              RENECO
##  Min.   :-5661   Min.   : 24944   Min.   :-77533   Min.   :-2.813
##  1st Qu.:  496   1st Qu.: 33101   1st Qu.:  2323   1st Qu.: 1.363
##  Median : 1939   Median : 41995   Median :  9727   Median : 4.237
##  Mean   : 2699   Mean   : 74180   Mean   : 18801   Mean   : 5.909
```

```
##  3rd Qu.: 3903   3rd Qu.: 68628   3rd Qu.: 26493   3rd Qu.: 8.620
##  Max.   :17026   Max.   :303904   Max.   :177707   Max.   :35.262
##  NA's   :1       NA's   :1        NA's   :1        NA's   :1
##      RENFIN            LIQUIDEZ            MARGEN          SOLVENCIA
##  Min.   :-359.77   Min.   :  0.0290   Min.   :-302.03   Min.   :-40.74
##  1st Qu.:   2.91   1st Qu.:  0.5500   1st Qu.:  11.70   1st Qu.:  5.94
##  Median :  14.14   Median :  0.9405   Median :  27.64   Median : 16.88
##  Mean   :  11.90   Mean   :  3.6208   Mean   :  30.63   Mean   : 28.12
##  3rd Qu.:  38.07   3rd Qu.:  1.5733   3rd Qu.:  39.59   3rd Qu.: 51.38
##  Max.   : 150.42   Max.   :128.4330   Max.   : 400.90   Max.   : 99.08
##                                       NA's   :2         NA's   :1
##     APALANCA
##  Min.   :-7770.00
##  1st Qu.:   12.28
##  Median :  134.90
##  Mean   :  333.21
##  3rd Qu.:  651.93
##  Max.   : 8049.39
##
```

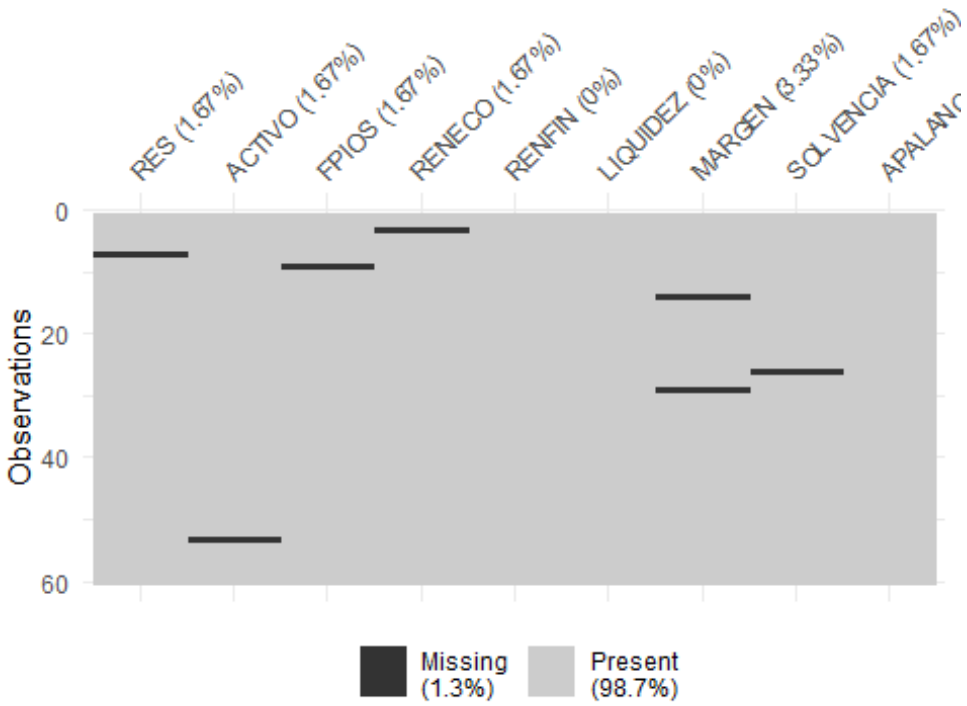
##1. Preparación de los datos:

1.1. Identificación y eliminación de los *missing values*:

Para obtener componentes principales es necesario que todos los casos posean dato para todas las variables del análisis.

Recordamos la sesión pasada: idea general de la localización de los *missing values* de las variables. Cálculo de porcentaje de casos con respecto al total de observaciones: usando el paquete *visdat* y la función `vis_miss()`:

```
library(visdat)

## Warning: package 'visdat' was built under R version 4.2.1

vis_miss(originales)

## Warning: `gather_()` was deprecated in tidyr 1.2.0.
## Please use `gather()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning
was generated.
```



Del gráfico anterior se desprende que existen 7 missing values repartidos en 6 de las 9 variables del estudio. Para localizarlos, filtramos con la función *filter*:

```
originales %>% filter(is.na(RES) | is.na(ACTIVO) | is.na(FPIOS) |
is.na(RENECO) | is.na(MARGEN) | is.na(SOLVENCIA)) %>%
               select(RES, ACTIVO, FPIOS, RENECO, MARGEN, SOLVENCIA)

##                                       RES    ACTIVO       FPIOS
RENECO
## Viesgo Renovables SL.            4609.00000 269730.00 177707.0000
NA
## Bivent Energia SA                     NA 183899.00  70033.0000
4.551
## Eolica La Janda SL               9880.09100 153429.44          NA
8.586
## Parc Eolic Sant Antoni SL         668.00000  69654.00   9727.0000
1.361
## WPD Parque Eolico El Poleo SL.    -30.63754  43997.60    520.6033 -
0.092
## Eolica La Brujula SA             2306.06200  42146.98  21694.7910
7.295
## Energias Naturales La Calzada SL  754.00000        NA  -1983.0000
3.446
##                                   MARGEN SOLVENCIA
## Viesgo Renovables SL.             11.818    65.883
## Bivent Energia SA                 22.792    38.082
## Eolica La Janda SL                38.256    16.428
```

```
## Parc Eolic Sant Antoni SL              NA    13.964
## WPD Parque Eolico El Poleo SL.    -11.121        NA
## Eolica La Brujula SA                   NA    51.474
## Energias Naturales La Calzada SL   15.561    -6.834
```

Como ya vimos, se puede proceder de varias formas con respecto a los *missing values*: obtener la información por otros medios, recurrir a estimaciones o bien, eliminarlos. Optamos por la *eliminación* usando la funciónn *filter* como sigue:

```
originales <- originales %>%
  filter(! is.na(RES) & ! is.na(ACTIVO) & ! is.na(FPIOS) & !
is.na(RENECO) & ! is.na(MARGEN) & ! is.na(SOLVENCIA))
```

Podemos verificar en el Environment que el data frame "originales" ha pasado a tener 53 casos (ver la diferencia con el objeto `originales2`).

1.2. Outliers:

La técnica de componentes principales es muy sensible a la existencia de outliers.

Procedemos a la *identificación y eliminación, en su caso, de valores atípicos*:

En nuestro análisis contamos con 9 variables:

1º "Resumiremos" el valor que toman dichas variables para cada caso, mediante el cálculo de la *distancia de Mahalanobis*.

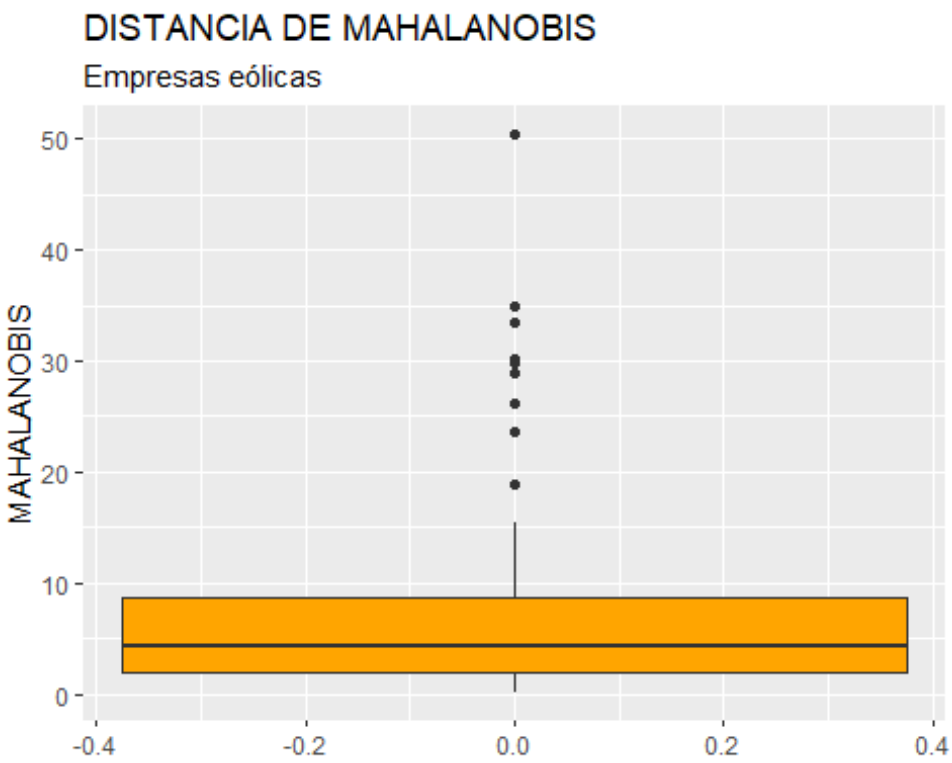
Usamos directamente el *objeto* (vector) al que llamaremos `MAHALANOBIS` y lo añadiremos como una nueva columna (variable) al *data frame* `originales_maha".

Usamos para ello, la función `cbind()`.

```
MAHALANOBIS <- mahalanobis(originales[],
                 center = colMeans(originales[]),
                 cov=cov(originales[]))
originales_maha <- cbind(originales, MAHALANOBIS)
```

Construimos un *diagrama de caja* para *MAHALANOBIS*, mediante la función *ggplot()* del paquete `ggplot2()`:

```
library (ggplot2)
ggplot(data = originales_maha, map = (aes(y = MAHALANOBIS))) +
  geom_boxplot(fill = "orange") +
  ggtitle("DISTANCIA DE MAHALANOBIS", subtitle = "Empresas eólicas") +
  ylab("MAHALANOBIS")
```



En el gráfico se observa que existen, por encima de la caja, varios outliers.

Identificamos los valores concretos:

```
Q1M <- quantile (originales_maha$MAHALANOBIS, c(0.25))
Q3M <- quantile (originales_maha$MAHALANOBIS, c(0.75))
originales_maha %>% filter(MAHALANOBIS > Q3M + 1.5*IQR(MAHALANOBIS) |
MAHALANOBIS < Q1M - 1.5*IQR(MAHALANOBIS)) %>% select(MAHALANOBIS)

##                                       MAHALANOBIS
## Parque Eolico La Boga SL.                 23.68380
## Guzman Energia SL                         28.92177
## Parque Eolico Santa Catalina SL           34.96254
## WPD Wind Investment SL.                   33.43117
## Molinos Del Ebro SA                       30.13740
## Tarraco Eolica SA                         29.79795
## WPD Parque Eolico Las Panaderas SL.       18.83613
## Eolica Navarra SL                         50.37161
## Parque Eolico El Moral SL                 26.16517

muestra <- originales_maha %>% filter(MAHALANOBIS <= Q3M +
1.5*IQR(MAHALANOBIS) & MAHALANOBIS >= Q1M - 1.5*IQR(MAHALANOBIS))
muestra <- muestra %>% select(-MAHALANOBIS)
```

Tras ejecutar el código anterior, se obtiene el listado de casos que se comportan, según su distancia de Mahalanobis observada, como outliers.

Hemos creado un nuevo *data frame* llamado `muestra`para recoger todos los casos que nos outliers y, además, no contienen valores faltantes.

*El *data frame* `muestra` va a ser con la que trabajaremos para el cálculo de las componentes principales.

2. Correlaciones y cálculo de componentes:

2.1. Comprobación de correlaciones:

El ACP solo tiene sentido si las variables comparten "información redundante" sobre los casos, que en nuestro caso son empresas eólicas.

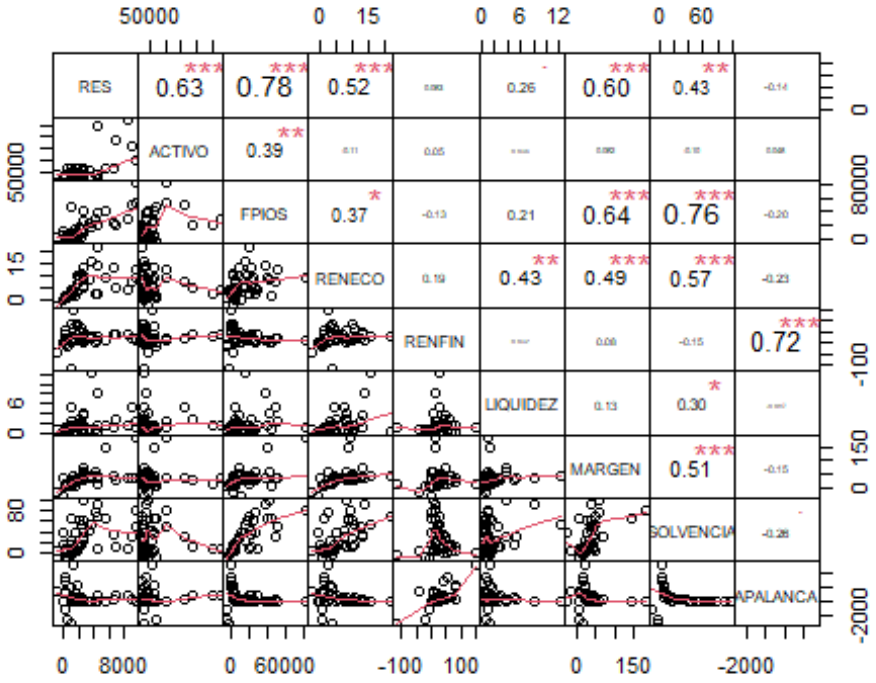
Para ello, un modo de comprobarlo es calcular las correlaciones entre las variables del estudio. Usamos la función `chart.Correlation()`del paquete `PerformanceAnalytics`.

Cargamos el paquete y usamos la función:

```
library(PerformanceAnalytics)

## Warning: package 'PerformanceAnalytics' was built under R version
4.2.1

## Loading required package: xts

## Warning: package 'xts' was built under R version 4.2.1

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 4.2.1

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

##
## 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

chart.Correlation(muestra, histogram = F, pch = 18)
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
```



¿Qué se aprecia en el gráfico?:

Altas correlaciones (en valor absoluto) entre algunas variables. Por tanto,*tiene sentido hacer un ACP*, ya que hay variables que *parecen compartir información*.

2.2. Obtención de componentes:

La obtención de las componentes se va a realizar mediante la función `prcomp()`.

Es conveniente que activemos el argumento `scale` con *"T" (true)* para que las variables originales sean consideradas en sus *versiones tipificadas*. Vamos a asignar los resultados a un objeto de nombre: `componentes`.

```
componentes <- prcomp (muestra, scale=T)
summary (componentes)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6
PC7
## Standard deviation     1.8892 1.3304 1.1952 0.9655 0.77718 0.63115
0.39059
## Proportion of Variance 0.3966 0.1967 0.1587 0.1036 0.06711 0.04426
0.01695
## Cumulative Proportion  0.3966 0.5932 0.7520 0.8555 0.92266 0.96692
0.98387
##                            PC8     PC9
## Standard deviation     0.32696 0.19562
## Proportion of Variance 0.01188 0.00425
## Cumulative Proportion  0.99575 1.00000
```

`Standard deviation` es la raíz cuadrada de los autovalores asociados a cada componente.

`Proportion of Variance` es la proporción de la suma de varianzas de las variables originales (comunalidad) recogida por cada componente.

`Cumulative Proportion`: la proporción de la suma de varianzas de las variables originales acumulada.

Las componentes aparecen ordenadas de más a menos importantes en función de la cantidad de varianza que capturan.

Las cuatro primeras componentes acumulan más del 85% de la varianza (comportamiento) de las variables originales.

2.2.1. Obtención de coeficientes o cargas:

Los coeficientes o cargas de cada componente se obtienen pidiendo a nuestro objeto `componentes` el elemento `rotation`:

```
cargas <- componentes$rotation
round(cargas,4)
```

```
##                PC1     PC2     PC3     PC4     PC5     PC6     PC7
PC8
## RES       -0.4506  0.2151 -0.2741  0.0937 -0.2092  0.0963 -0.2840
0.4696
## ACTIVO    -0.1613  0.2935 -0.6644  0.2925 -0.0500  0.1207  0.1577 -
0.5535
## FPIOS     -0.4695  0.0191 -0.1986 -0.1873  0.3863  0.0409  0.0653
0.4048
## RENECO    -0.3688  0.0340  0.4372  0.1261 -0.4644  0.4159 -0.3209 -
0.2228
## RENFIN     0.0369  0.6678  0.2867 -0.0694 -0.1542  0.1195  0.6289
0.1784
## LIQUIDEZ  -0.2222  0.0194  0.3062  0.7923  0.2652 -0.3904  0.0690
0.0363
## MARGEN    -0.4013  0.0772  0.0728 -0.4020 -0.2567 -0.7236  0.0164 -
0.2757
## SOLVENCIA -0.4145 -0.1613  0.2364 -0.2111  0.4965  0.3289  0.2142 -
0.3568
## APALANCA   0.1760  0.6227  0.1252 -0.1266  0.4270 -0.0553 -0.5838 -
0.1524
##                PC9
## RES       -0.5591
## ACTIVO     0.1122
## FPIOS      0.6210
## RENECO     0.3385
## RENFIN    -0.0162
## LIQUIDEZ  -0.0019
## MARGEN    -0.0184
## SOLVENCIA -0.4169
## APALANCA   0.0145
```

Le pedimos que nos lo redondee a 4 decimales con la segunda línea del código.

Si observamos el resultado:

En columnas: aparecen las componentes.

En las filas: las variables originales.

En las intersecciones: los coeficientes (o cargas) calculados.

Aclaración:

Por ejemplo, la combinación lineal que define a la primera componente:

$$yi1 = -0.4506 \cdot RESi - 0.1613 \cdot ACTIVOi - \cdots + 0.1760 \cdot APALANCAi$$

Analizamos las cargas para la primera y segunda componente:

Primera componente: las mayores cargas (en términos absolutos) son las correspondientes a: FPIOS, RES, SOLVENCIA y MARGEN.

Segunda componente: las cargas correspondientes a RENFIN y APALANCA son las que tienen un mayor valor absoluto.

IMPORTANTE: a menudo, encontrar un significado económico para las componentes no es una tarea sencilla, puesto que en realidad son elementos puramente matemáticos.

3. Retención de componentes mediante el criterio del Autovalor mayor que 1:

Queremos menos componentes que variables originales, porque así simplificaremos la información suministrada por dichas variables –> a las que llamaremos *componentes principales*.

Hay varios métodos de selección de componentes principales.

Nosotros usaremos aquel que selecciona las varianzas de las componentes (o autovalores) mayores a uno.

Procedimiento:

1º Calculamos los autovalores y los mostramos.

2ª Comprobación de cuántos autovalores son mayores que 1.

Los autovalores son el *cuadrado del elemento Standard deviation* `sdev` del objeto `componentes` que hemos creado a partir de la función `prcomp()`.

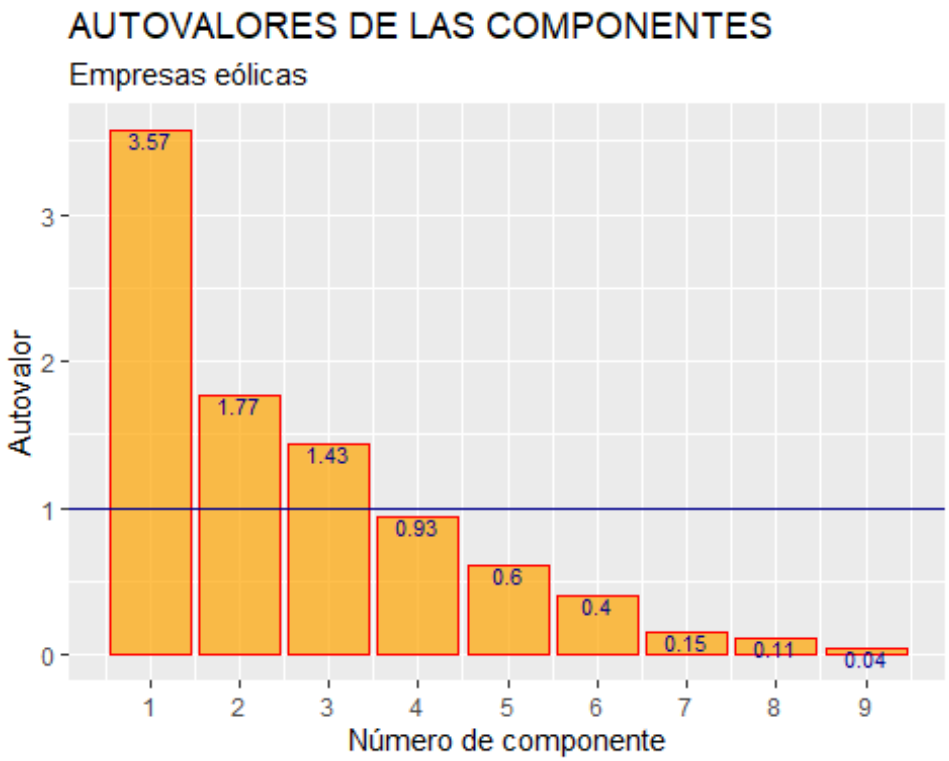
```
orden <- c(1:ncol(muestra))
autovalor <- componentes$sdev^2
autovalores <- data.frame(orden, autovalor)
```

Hemos creado un *data frame* con estos autovalores calculados (y su orden, al que hemos llamado variable o columna `orden`: es un vector de números enteros consecutivos que va desde uno hasta número de variables originales o de componentes)

Lo representamos en un *gráfico de barras* y lo llamamos:

```
autograph <- ggplot(data = autovalores, map = (aes(x = orden, y =
autovalor))) +
             geom_col(colour = "red", fill = "orange", alpha = 0.7) +
             scale_x_continuous(breaks=c(1:nrow(autovalores)))+
             geom_hline(yintercept = 1, colour = "dark blue") +
             geom_text(aes(label = round(autovalor,2)), vjust = 1, colour
= "dark blue", size = 3) +
             ggtitle("AUTOVALORES DE LAS COMPONENTES", subtitle =
"Empresas eólicas") +
             xlab ("Número de componente") +
             ylab("Autovalor")
```

```
autograph
```



En el gráfico de barras, si no se quieren representar las frecuencias sino los valores que toma una variable (`autovalor`) para cada valor de la otra variable (en este caso, `orden`), en el geom habrá que añadir el argumento `stat` con el valor `identity`.

Se utiliza el elemento `scale_x_continuous()` para pesonalizar la escala del eje x, y que se divida dicho eje en tantos tramos como componentes hay.

Del gráfico se desprende: que los 3 primeros autovalores son mayores que 1, sobrepasan la línea horizontal que hemos dibujado con la función `geom_hline()`, por lo que se retendrán las 3 primeras componentes, que serán las componentes principales.

Las tres primeras componentes (las componentes principales) según este criterio de los autovalores superiores a 1, recogen el 75,20% de la varianza total de las variables originales o *comunalidad*.

También se puede observar lo anterior, no solo a través del *summary* sino, gráficamente con el siguiente código:

```
autovalores <- autovalores %>% mutate(variacum =
100*(cumsum((autovalor/nrow(autovalores)))))
checkcp <- c(1:nrow(autovalores))
            for (i in 1: nrow(autovalores)) {
              if (autovalores$autovalor[i] >= 1) {
```

```
                checkcp[i] <- c("CP")
              } else {
                checkcp[i] <- c("NCP")}
            }
checkcp

## [1] "CP"  "CP"  "CP"  "NCP" "NCP" "NCP" "NCP" "NCP" "NCP"
```

Desciframos el código:

1. Añadimos a nuestro *data frame* `autovalores` una nueva columna (o variable) que es la *suma acumulada del porcentaje de comunalidad recogido por las sucesivas componentes, que están ordenadas de mayor a menor autovalor*. Esta porcentaje se calcula con R, con la función `cumsum()`.

Recordamos que como las variables fueron tipificadas para calcular las componentes, la comunalidad, que coincide con la suma de las varianzas de las componentes (autovalores), es igual al número de variables o componentes (valor que toma la función `nrow()`).

2. Creamos un *vector* que contiene tantos elementos como variables o componentes hay en el análisis al que llamamos `checkcp`.

Estos elementos se han determinado a partir de un *bucle* , y con una *estructura condicional*.

Construcción del bucle:

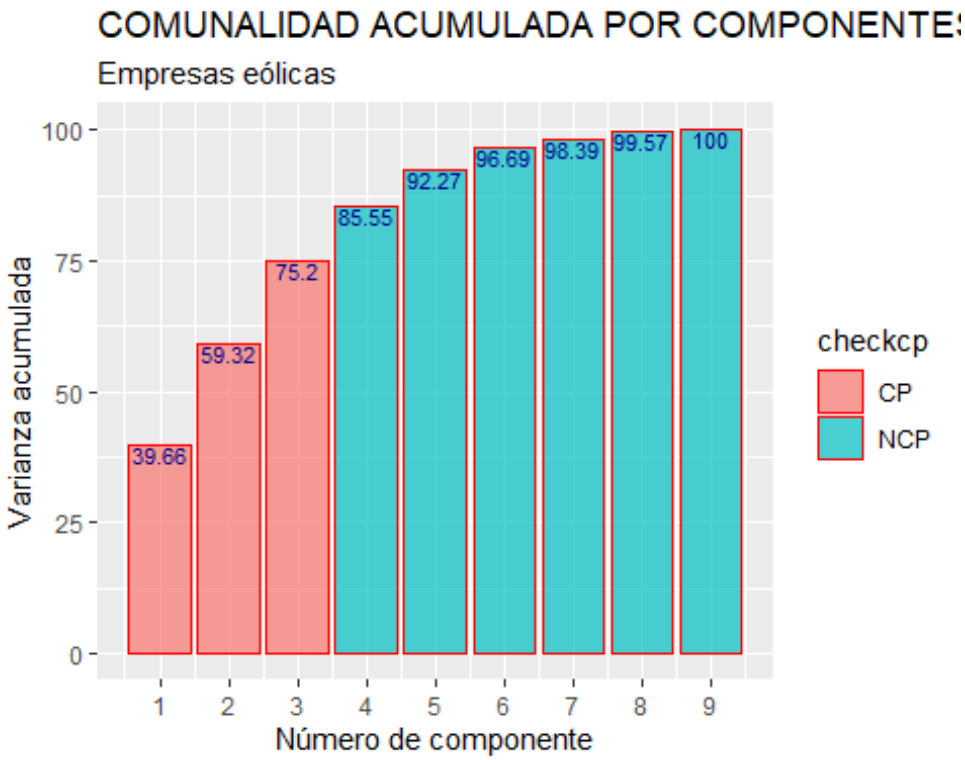
El bucle comienza con la instrucción `for(){}`. Dentro del paréntesis que sigue a la instrucción: - Establecemos un valor *contador* (valor "i", por ejemplo). Valor que va a ir aumentando de uno en uno, desde 1 hasta llegar al número de variables, componentes o autovalores (9 en nuestro ejercicio). - Cuando "i" toma un valor: se realiza lo que está dentro de las llaves del bucle "{ }". Luego vuelve a comenzar, tomando "i" el siguiente valor (1, 2, 3,…,9), hasta que termina.

- Lo que hay *dentro del bucle* es una *estructura condicional* *`if (){}/ else{}`*.
- Dentro del paréntesis de `if()` establecemos una condición:
- que el autovalor de la componente de orden "i" sea mayor o igual que 1.
- Si se cumple: el código a ejecutar será el que se encuentra entre las llaves inmediatamente posteriores al paréntesis de la condición (que el elemento "i" del vector `checkcp` tome valor `CP` (componente principal)).

Si no se cumple: se ejecutará el código situado dentro de las llaves localizadas después de la instrucción else (que el elemento "i" del vector `checkcp` tome valor `NCP` (no componente principal)).

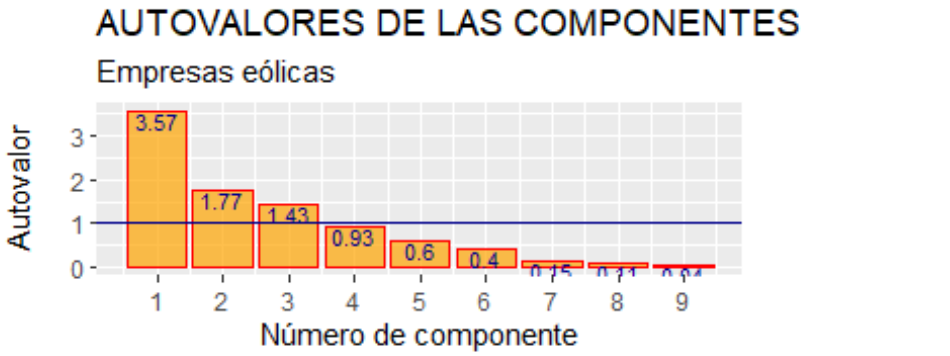
Podemos plasmar esta información gráficamente en un gráfico de barras y el vector `checkcp`será capaz de distinguir entre las CP y las componentes que no serán retenidas:

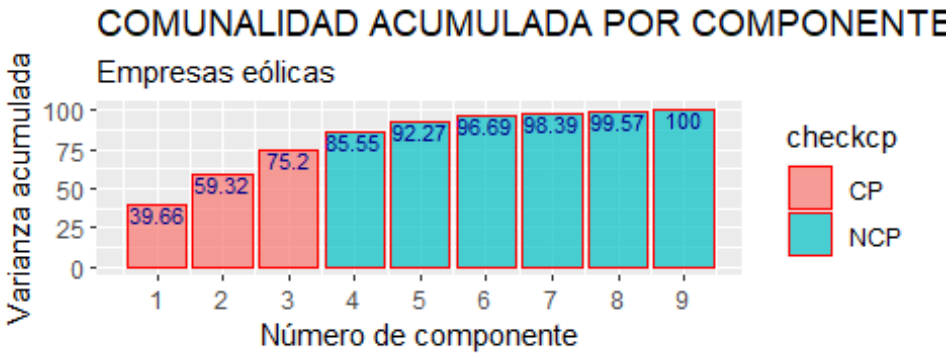
```
vacumgraph <- ggplot(data = autovalores, map = (aes(x = orden, y =
variacum, fill = checkcp))) +
              geom_bar(stat = "identity", colour = "red", alpha = 0.7) +
              scale_x_continuous(breaks=c(1:nrow(autovalores)))+
              geom_text(aes(label = round(variacum,2)), vjust = 1, colour
= "dark blue", size = 3) +
              ggtitle("COMUNALIDAD ACUMULADA POR COMPONENTES", subtitle =
"Empresas eólicas") +
              xlab ("Número de componente") +
              ylab("Varianza acumulada")
vacumgraph
```



Usamos el paquete `patchwork()`, ya visto en el tema anterior, para unnir los dos gráficos anteriores, uno debajo del otro:

```
library (patchwork)

## Warning: package 'patchwork' was built under R version 4.2.1

autograph / vacumgraph
```





En el gráfico inferior se observa cómo las tres primeras componentes (CP) acumulan el 75,2% de la varianza total o *comunalidad*.

Así pues, para caracterizar el comportamiento de las empresas eólicas estudiadas en otros análisis (regresión, clúster, etc.) podrían emplearse como variables las puntuaciones para cada empresa de las tres primeras componentes (que serían las componentes principales), en lugar de utilizar los valores de las 9 variables originales.

4. Cálculo de las puntuacions de los casos (o *scores*):

Para obtener las puntuaciones de cada caso (empresa) en cada CP (las tres primeras componentes)–> estas puntuaciones están guardadas en la matriz "x" del objeto `prcomp()`.

1. Renombramos las 3 primeras columnas (componentes) de esta matriz como `scores` y vamos a ver las puntuaciones de las primeras empresas, redondeadas a 4 decimales:

```
scores <- componentes$x[,1:3]
round(head(scores), 4)
```

```
##                                     PC1    PC2     PC3
## Naturgy Wind, S.L.                -1.4408 2.0744 -3.5595
## Al-Andalus Wind Power SL          -0.0498 1.4615 -3.0214
## Acciona Eolica Del Levante SL     -0.8477 1.5180 -1.9849
## Esquilvent SL                     -2.7713 0.9430 -1.5715
## Eolia Gregal De Inversiones SA.   -5.6488 0.4244 -1.1871
## CYL Energia Eolica SL             -3.1226 0.4955 -0.7791
```

Se obtienen, para las 6 primeras empresas de la muestra (excluidas empresas con missing values y outliers).

Alternativa: mediante cálculo matricial, ya que es el resultado de multiplicar la matriz de variables originales (tipificadas) por la matriz compuesta por las 3 primeras componentes, que son las componentes principales. En R se realizaría como sigue:

```
x <-data.matrix(scale(muestra))
puntuaciones <- x %*% cargas[,1:3]
round(head(puntuaciones), 4)
```

```
##                                     PC1    PC2     PC3
## Naturgy Wind, S.L.               -1.4408 2.0744 -3.5595
## Al-Andalus Wind Power SL         -0.0498 1.4615 -3.0214
## Acciona Eolica Del Levante SL    -0.8477 1.5180 -1.9849
## Esquilvent SL                    -2.7713 0.9430 -1.5715
## Eolia Gregal De Inversiones SA. -5.6488 0.4244 -1.1871
## CYL Energia Eolica SL            -3.1226 0.4955 -0.7791
```

1. Tipificamos los valores del *data frame* `muestra`mediante el argumento `scale`.
2. El resultado lo traspasamos a un formato de *matriz* con la función `data.matrix()`.
3. Lo anterior lo asignamos a un objeto que denominamos `x` para operar algebraicamente.
4. Obtenemos la *matriz* `puntuaciones`como multiplicación de la matriz `putuacionesy` la `matriz de componentes principales`(que se compone de las 3 primeras columnas de las cargas calculadas en la obtención de las componentes).

#¿Para qué realizamos este cálculo de las componentes principales?:

Las puntuaciones de los casos para cada una de las tres CP pueden integrarse en un data frame y ser utilizadas como cualquier otra variable en un análisis multivariante.

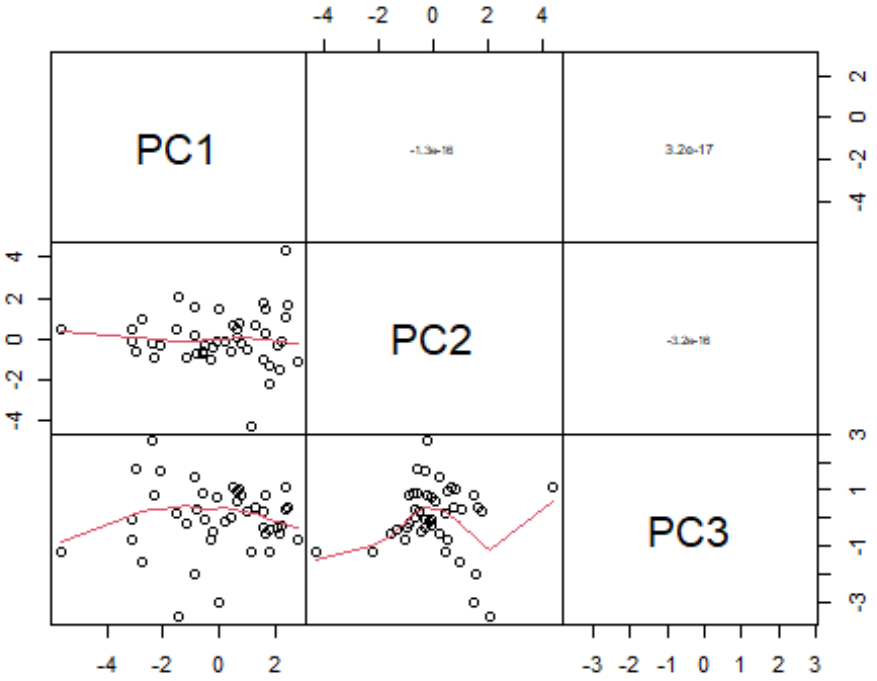
Esto es así pues contienen gran parte de la información que, como caracterización de las distintas empresas, contenían las 9 variables originales.

5. Comprobación de correlaciones entre CP:

si se calcula la matriz de correlaciones existentes entre las CP se obtiene que estas correlaciones son prácticamente nulas, lo que nos permitirá trabajar con ellas.

```
chart.Correlation(puntuaciones, histogram = F, pch = 18)
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter
```



FIN.

¡Muchas gracias por vuestra atención!