Análisis de componentes principales y Cluster:

Alejandro Sierra Fernández

Nos disponemos en este documento a responder a las preguntas que se nos proponen en la primera parte de la práctica de Minería de Datos. Veremos, a través de distintos métodos, cómo reducir el número de variables de nuestro set de datos y encontraremos las relaciones ocultas que puedan existir entre las distintas provincias y variables. Para ello emplearemos un fichero con información de tipo socioeconómica sobre las provincias españolas. Damos por hecho que se dispone previamente de los paquetes necesarios para aplicar las funciones que vamos a emplear, por lo que no indicaremos su instalación.

Comenzamos limpiando las variables y las funciones que podamos tener almacenadas en el entorno de trabajo, cargando los datos y explorándolos. La lectura de los datos que realizamos con la función *read_excel* requiere que le especifiquemos la ruta del archivo. En este caso estamos empleando una ruta relativa, que habrá que modificar según la ubicación en la que tengamos guardado el archivo Excel con el que vamos a trabajar.

```
rm(list=ls())

datos <-
read_excel("C:/Users/aleja/OneDrive/Escritorio/Tarea/Provincias.xlsx")datos <-

as.data.frame(datos)

colnames(datos)

summary(datos)

str(datos)

tabla1 <- stat.desc(dep_datos, basic = FALSE)

knitr::kable(tabla1, digits=2, caption="Estadísticos descriptivos")

formattable(tabla1)
```

	Poblacion	Mortalidad	Natalidad	IPC	NumEmpresas	Industria	Construccion	CTH
median	6.147230e+05	9.1200000	8.9750000	102.32300000	3.800000e+04	2.516000e+03	5.070000e+03	1.548800e+04
mean	8.994489e+05	9.3790385	8.8388462	102.35013462	6.128612e+04	3.807769e+03	7.804788e+03	2.374121e+04
SE.mean	1.600722e+05	0.2940853	0.2965973	0.11362095	1.255286e+04	6.697486e+02	1.452445e+03	4.184440e+03
CI.mean.0.95	3.213584e+05	0.5904008	0.5954439	0.22810357	2.520092e+04	1.344576e+03	2.915904e+03	8.400613e+03
var	1.332402e+12	4.4972794	4.5744379	0.67130541	8.193867e+09	2.332528e+07	1.096990e+08	9.104959e+08
std.dev	1.154297e+06	2.1206790	2.1387936	0.81933230	9.051998e+04	4.829626e+03	1.047373e+04	3.017442e+04
coef.var	1.283338e+00	0.2261084	0.2419766	0.00800519	1.477006e+00	1.268361e+00	1.341962e+00	1.270972e+00

Además, generamos otro dataframe que no incluya la variable en la que se almacenan los nombres de las provincias:

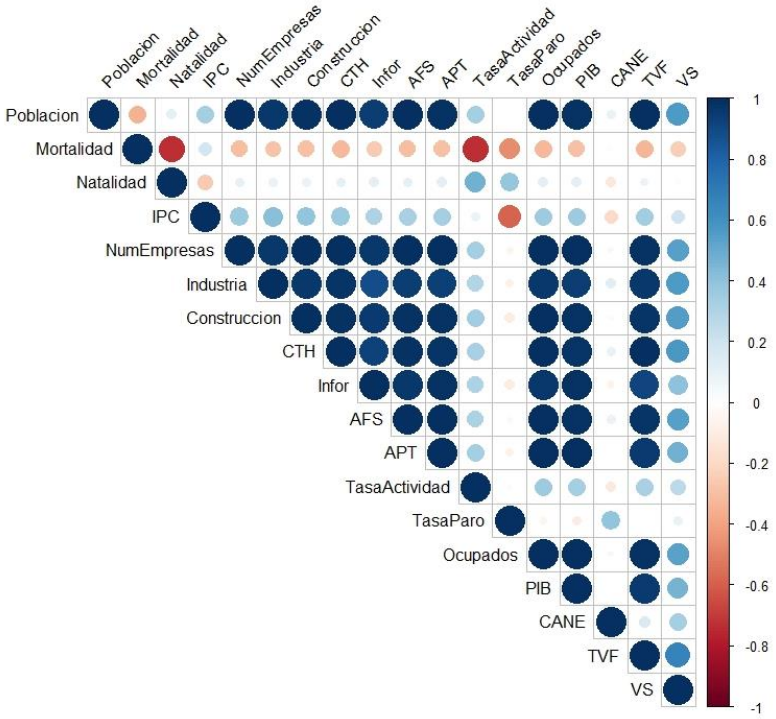
```
dep_datos <- datos[,-1]
rownames <- datos[,1]
```

Tras haber analizado cuestiones como el tipo de formato de las variables o los marcadores estadísticos más importantes, llegamos a la conclusión de que nuestro dataset no presenta elementos o variables por los que nos tengamos que preocupar. Podemos entonces pasar al análisis de la información directamente.

Cabe destacar que para presentar algunas tablas emplearemos la librería *formattable*, que ofrece una representación más limpia y la posibilidad de definir funciones que nos permitan personalizar nuestras tablas. El dataset *tabla1*, definido en la página anterior y que contiene algunas de los parámetros estadísticos de nuestros datos, lo hemos mostrado en forma de tabla utilizando este método.

Pasamos ahora a calcular la matriz de correlación de nuestros datos y a representarla gráficamente. Lo hacemos con el siguiente código:

```
correlacion <- cor(dep_datos, method="pearson")
corrplot(correlacion, type="upper", tl.col="black", tl.srt=45)
```



Empleamos de nuevo la librería *formattable* para representar la matriz de correlación. Empezamos definiendo una función a la que llamamos *colour_corr* que se encargará de colorear los elementos de la tabla dependiendo del rango de valores al que pertenezca. Esto nos permitirá a simple vista ver qué correlaciones positivas (verdes) y negativas (rojas) son las más importantes. Una vez hayamos definido la función, la aplicaremos a cada elemento de nuestro dataset dentro de la función *formattable*:

```
colour_corr <- formatter("span",

      style = x ~ style(display = "block", font.weight = "bold",color =
"black","border-radius" = "4px",

      "padding-right" = "4px",

      "background-color" = ifelse(x <= -0.8, "#A50026",

                           ifelse(x > -0.8 & x <= -0.6, "#D73027",

                           ifelse(x > -0.6 & x <= -0.4, "#F46D43",

                           ifelse(x > -0.4 & x <= -0.2, "#FDAE61",

                           ifelse(x > -0.2 & x <= 0, "#FEE08B",

                           ifelse(x > 0 & x <= 0.2, "#D9EF8B",

                           ifelse(x > 0.2 & x <= 0.4, "#A6D96A",

                           ifelse(x > 0.4 & x <= 0.6, "#66BD63",

                           ifelse(x > 0.6 & x <= 0.8, "#1A9850",

                           ifelse(x > 0.8, "#006837",NA)))))))))))
formattable(correlaciondf, list(~ colour_corr))
```

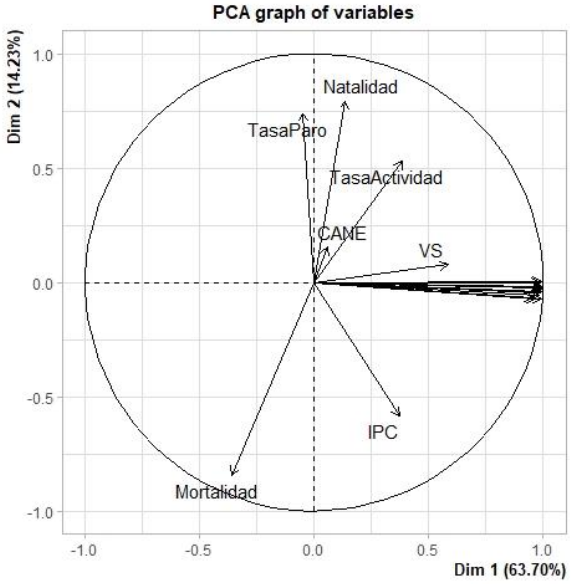
Mostramos la correlación sólo de algunas variables por facilitar su visualización:

	Poblacion	Mortalidad	Natalidad	IPC	NumEmpresas	Industria	Construccion	CTH
Poblacion	1.000000000	-0.34359937	0.11445938	0.33483095	0.99453328	0.96458267	0.98416996	0.996110046
Mortalidad	-0.343599371	1.00000000	-0.73652925	0.18846127	-0.30927237	-0.28184861	-0.29590918	-0.329333667
Natalidad	0.114459382	-0.73652925	1.00000000	-0.25391157	0.10501778	0.09274529	0.08874307	0.101444356
IPC	0.334830949	0.18846127	-0.25391157	1.00000000	0.36475679	0.41971227	0.39988432	0.362397751
NumEmpresas	0.994533276	-0.30927237	0.10501778	0.36475679	1.00000000	0.96870767	0.99458429	0.993878918
Industria	0.964582668	-0.28184861	0.09274529	0.41971227	0.96870767	1.00000000	0.96613670	0.976499325
Construccion	0.984169957	-0.29590918	0.08874307	0.39988432	0.99458429	0.96613670	1.00000000	0.985702825
CTH	0.996110046	-0.32933367	0.10144436	0.36239775	0.99387892	0.97649932	0.98570283	1.000000000
Infor	0.944538806	-0.25983697	0.11454957	0.30044315	0.96270044	0.88925300	0.95796587	0.929022359
AFS	0.992202208	-0.30518551	0.10467705	0.32267001	0.99263984	0.94716931	0.98427617	0.983338024
APT	0.980028733	-0.29572891	0.11326870	0.33261608	0.99058205	0.93414916	0.98457937	0.971043919
TasaActividad	0.334146420	-0.73348067	0.47242170	0.08970542	0.33232416	0.29177908	0.34122871	0.325934818
TasaParo	0.009850323	-0.46397242	0.38303840	-0.58223482	-0.05623228	-0.08035375	-0.10917096	-0.007188617
Ocupados	0.995500027	-0.32761124	0.11174153	0.35818832	0.99834423	0.96153241	0.99320060	0.991129933
PIB	0.980976869	-0.29560406	0.11382945	0.35539497	0.99027798	0.94271768	0.98529352	0.971695960
CANE	0.097205159	0.01643287	-0.12309106	-0.19120709	0.04483474	0.12376621	0.02974452	0.093619748
TVF	0.990717314	-0.33230307	0.08444871	0.34095210	0.98226824	0.96533176	0.97574923	0.991031374
VS	0.566539758	-0.24903743	-0.02766720	0.19117175	0.54044321	0.56716093	0.55654176	0.578205863

Estudiando la matriz de correlación observamos que las variables más correladas de forma inversa son las parejas Natalidad-Mortalidad y TasaActividad-Mortalidad, con coeficientes de correlación de aproximadamente -0.73 cada una.

A continuación, realizamos un análisis de componentes sobre la matriz de correlaciones, calculando los siete principales. Para ello empleamos la función PCA, que toma como inputs el dataframe, la opción scale.unit=TRUE que los estandariza, el número de componentes que queremos calcular y la opción que nos permite graficar la proyección de las variables originales en el plano de dos de las nuevas dimensiones. Además, tableamos los autovalores:

```
fit<-PCA(dep_datos ,scale.unit=TRUE,ncp=7,graph=TRUE)
formattable(as.data.frame(fit$eig))
```

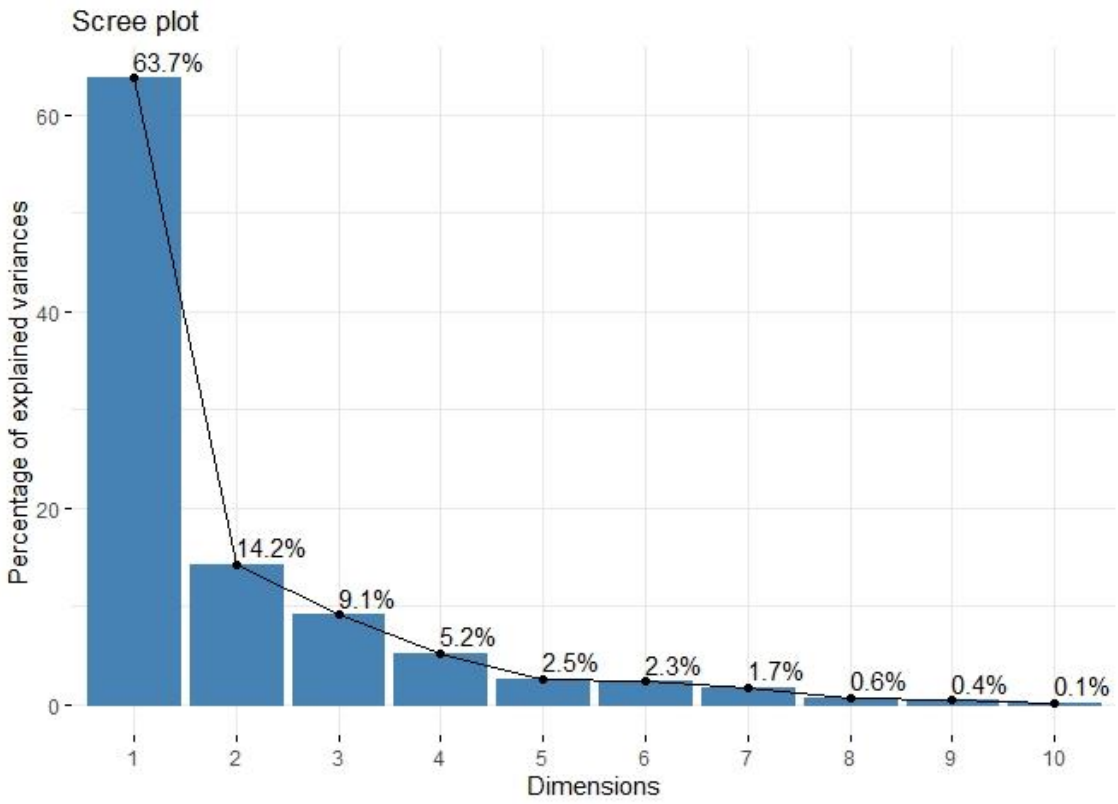


	eigenvalue	percentage of variance	cumulative percentage of variance
comp 1	11.46639894136	63.7022163409	63.70222
comp 2	2.56050798474	14.2250443597	77.92726
comp 3	1.63409670978	9.0783150543	87.00558
comp 4	0.93404883548	5.1891601971	92.19474
comp 5	0.45653201479	2.5362889711	94.73102
comp 6	0.41436300447	2.3020166915	97.03304
comp 7	0.30717431608	1.7065239782	98.73957
comp 8	0.11661535270	0.6478630706	99.38743
comp 9	0.07309913298	0.4061062943	99.79353
comp 10	0.02034495755	0.1130275419	99.90656
comp 11	0.00902756754	0.0501531530	99.95672
comp 12	0.00356611496	0.0198117498	99.97653
comp 13	0.00237272039	0.0131817799	99.98971
comp 14	0.00080139517	0.0044521954	99.99416
comp 15	0.00054727486	0.0030404159	99.99720
comp 16	0.00035127103	0.0019515057	99.99915
comp 17	0.00010289753	0.0005716530	99.99972
comp 18	0.00004950858	0.0002750477	100.00000

Analizando la tabla de autovalores vemos que el primero toma el valor 11.46 y que explica el 63.70% de la variabilidad de los datos. Si añadimos el segundo autovalor explicaríamos el 77.93% de la variabilidad, y así sucesivamente. Si nos quedamos con los cuatro primeros autovalores estaríamos explicando el 92.19% de la variabilidad, que es un porcentaje muy bueno. Sin embargo, este cuarto autovalor es menor a la unidad, por lo que el porcentaje de la varianza que explica ya es menor que el de una sola variable. Es por este motivo por el que decidimos que la mejor opción sería mantener los tres primeros para ganar en simpleza. A pesar de que perdamos en torno al 5% de la variabilidad que explicábamos con cuatro autovalores, seguimos incluyendo el 87% de esta, que es un porcentaje más que aceptable.

Podemos ver mostrar también el porcentaje de variabilidad que explica cada autovalor de forma gráfica:

```
fviz_eig(fit, geom="line")+ theme_grey()
fviz_eig(fit,addlabels=TRUE)
```



Pasamos ahora a analizar nuestras variables sobre el nuevo espacio formado por los tres componentes principales, que es el número con el que hemos decidido quedarnos. Guardamos también en la variable *var* la información de las variables que recoge el objeto *fit*:

```
fit<-PCA(dep_datos,scale.unit=TRUE, ncp=3, graph=TRUE) #Nos quedamos con 3
componentes

var<-get_pca_var(fit)
```

Mostramos los coeficientes para calcular las 3 componentes principales, que se definen como la combinación lineal de las variables estandarizadas multiplicadas por sus coeficientes correspondientes

```
formattable(as.data.frame(fit$svd$V),

  align = "c",

  list(~ formatter("span",

      style = x ~ formattable::style(display = "block",

                                     "border-radius" = "2px",

                                     "padding" = "5px",

                                     "text-align" = "center"))))
```

V1	V2	V3
0.29352698	0.002493342	0.049947340
-0.10629199	-0.527120141	0.188867032
0.04062702	0.495419403	-0.270681171
0.10991169	-0.365322291	-0.262317619
0.29418239	-0.026147510	0.007952241
0.28562283	-0.044737470	0.046369149
0.29326489	-0.045381770	-0.012082096
0.29286608	-0.010617503	0.048841149
0.28150192	-0.042018125	-0.064640257
0.29246592	-0.016454205	0.039554303
0.29072490	-0.029433207	-0.028068230
0.11433729	0.330614591	-0.362734482
-0.01399634	0.461562171	0.387356995
0.29442838	-0.016838806	0.002429338
0.29090708	-0.036157691	-0.037491740
0.01778860	0.096184477	0.656743135
0.29156666	-0.001599604	0.099761847
0.17234327	0.047821027	0.290142940

De esta forma, para calcular la primera componente a partir de las variables originales (V'), procederíamos del siguiente modo:

$$V1 = \sum(\alpha_i * V_i') = 0.29(V1') - 0.11(\text{V2}') + 0.04(\text{V3}') + 0.11(V4') + 0.29(V5') + 0.29(V6') + 0.29(V7') + 0.39(V8') + 0.28(V9') + 0.29(V10') + 0.29(V11') + 0.11(V12') - 0.01(V13') + 0.29(V14') + 0.29(V15') + 0.02(V16') + 0.29(V17') + 0.18(V18')$$

Vemos ahora la relación de correlación de las variables originales y las componentes principales:

```
formattable(as.data.frame(var$cor), align = "c", caption = "Correlaciones de
la CP con lasvariables", list(~ colour_corr))
```

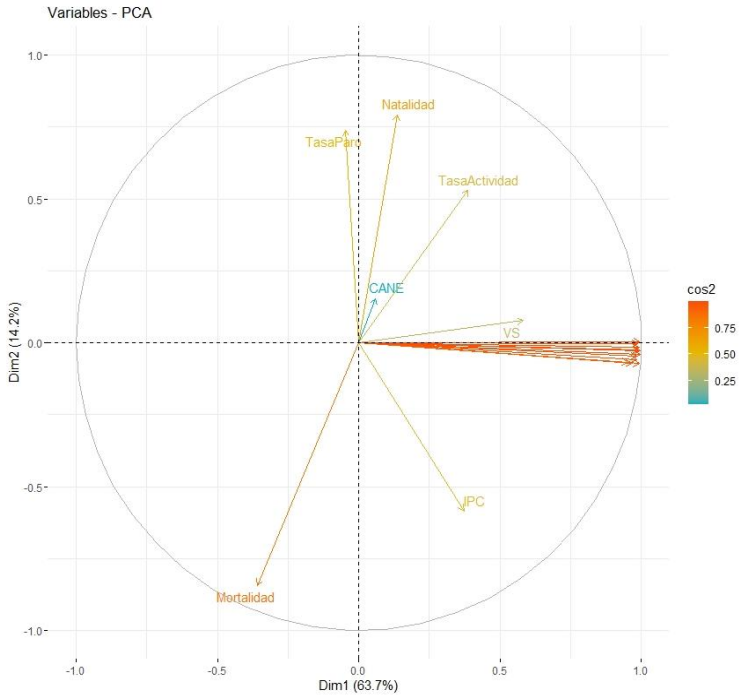
Correlaciones de la CP con lasvariables

	Dim.1	Dim.2	Dim.3
Poblacion	0.99394316	0.003989744	0.063848580
Mortalidad	-0.35992669	-0.843475898	0.241432113
Natalidad	0.13757149	0.792749686	-0.346016593
IPC	0.37218373	-0.584573656	-0.335325313
NumEmpresas	0.99616251	-0.041840166	0.010165492
Industria	0.96717807	-0.071587054	0.059274514
Construccion	0.99305566	-0.072618036	-0.015444759
CTH	0.99170521	-0.016989690	0.062434517
Infor	0.95322382	-0.067235670	-0.082630799
AFS	0.99035020	-0.026329340	0.050562974
APT	0.98445474	-0.047097804	-0.035880121
TasaActividad	0.38716977	0.529035826	-0.463689989
TasaParo	-0.04739450	0.738572741	0.495165389
Ocupados	0.99699549	-0.026944763	0.003105467
PIB	0.98507163	-0.057858046	-0.047926363
CANE	0.06023588	0.153910432	0.839526520
TVF	0.98730511	-0.002559620	0.127527357
VS	0.58359002	0.076521234	0.370894920

Se observa que la primera dimensión encuentra su mayor correlación con los ocupados, la segunda con la natalidad y la tercera con la variable CANE.

Resulta interesante ver cómo se proyectan las variables originales en el plano formado por las nuevas dimensiones. Esto lo conseguimos a través de los cosenos al cuadrado a través de la función *fviz_pca_var*:

```
fviz_pca_var(fit, col.var="cos2",gradient.cols = c("#00AFBB", "#E7B800",
"#FC4E07"),repel = TRUE)
```

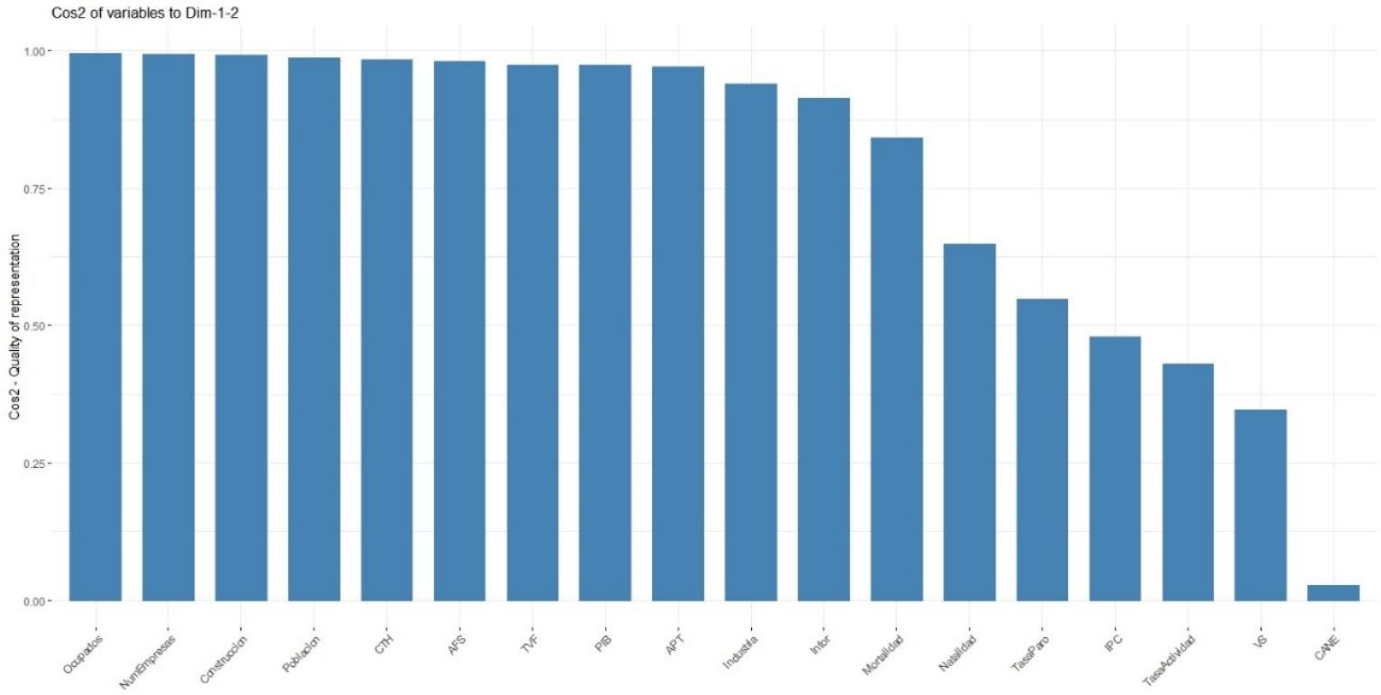


Vemos que la magnitud de los vectores viene dada por los cosenos al cuadrado, que son las correlaciones al cuadrado y que expresan la proporción de la varianza de cada variable que es explicada por cada componente.

En este caso el plano está formado por las dos primeras componentes principales, y sobre él aparecen vectores asociados a las variables originales. Dependiendo del ángulo que presenten estos vectores con respecto a los ejes, variará la correlación que existe las variables y las componentes principales. De este modo, vemos cómo la mayor parte de ellas están principalmente correladas con la segunda dimensión (eje de abcisas), mientras que variables como TasaParo o Natalidad se encuentran mucho más cercanas a la primera. Esto quiere decir que variaciones en la variable Natalidad se verán reflejadas principalmente en la primera componente principal, que la explica mucho mejor. Cabe destacar que variables como CANE o TasaActividad presentan correlaciones similares para las dos dimensiones al ser el ángulo más próximo a los 45º. Dos variables representadas a través vectores con la misma dirección, pero distinto módulo, se diferencian en la magnitud de la correlación. Es decir, que la variabilidad de ambas variables es explicada relativamente de la misma forma por las dos componentes principales, pero la de mayor módulo lo hace de manera más precisa, con mayor correlación.

Podemos graficar también el módulo de estos vectores únicamente a través de un gráfico de barras:

```
fviz_cos2(fit,choice="var",axes=1:2)
```



La tabla de los cosenos al cuadrado quedaría del siguiente modo:

```
formattable(as.data.frame(var$cos2), caption="Cosenos al cuadrado", align="c")
```

Cosenos al cuadrado

	Dim.1	Dim.2	Dim.3
Poblacion	0.987922996	0.000015918055	0.004076641160
Mortalidad	0.129547224	0.711451591311	0.058289464953
Natalidad	0.018925915	0.628452064085	0.119727482403
IPC	0.138520733	0.341726359414	0.112443065568
NumEmpresas	0.992339742	0.001750599491	0.000103337226
Industria	0.935433419	0.005124706292	0.003513468029
Construccion	0.986159537	0.005273379161	0.000238540589
CTH	0.983479227	0.000288649549	0.003898068869
Infor	0.908635650	0.004520635310	0.006827849008
AFS	0.980793513	0.000693234163	0.002556614323
APT	0.969151143	0.002218203098	0.001287383081
TasaActividad	0.149900428	0.279878905672	0.215008406111
TasaParo	0.002246239	0.545489693626	0.245188762534
Ocupados	0.994000007	0.000726020244	0.000009643925
PIB	0.970366117	0.003347553453	0.002296936278
CANE	0.003628361	0.023688420926	0.704804776949
TVF	0.974771377	0.000006551656	0.016263226870
VS	0.340577314	0.005855499233	0.137563041902

Vemos, por ejemplo, que la variación en la Población queda explicada en un 99% por la variable 1. La variable peor interpretada por los nuevos componentes es VS, cuya varianza queda explicada en un 34% por el primer componente y en un 14% por el tercero.

Gráficamente:

```
corrplot(var$cos2,is.corr=FALSE)
```

Sabiendo también que la varianza de las componentes proviene de la varianza de las variables originales, podemos ver la contribución que tiene cada variable para cada componente. Es decir, el porcentaje de la varianza de una componente que proviene de cada variable. Para ello empleamos la siguiente línea de código:

```
formattable(as.data.frame(var$contrib), caption = "Contribuciones de las
variables", align="c")
```

Contribuciones de las variables

	Dim.1	Dim.2	Dim.3
Poblacion	8.61580869	0.0006216756	0.2494736777
Mortalidad	1.12979868	27.7855642533	3.5670755962
Natalidad	0.16505544	24.5440384420	7.3268296599
IPC	1.20805785	13.3460376398	6.8810532997
NumEmpresas	8.65432772	0.0683692260	0.0063238133
Industria	8.15804006	0.2001441246	0.2150097977
Construccion	8.60042932	0.2059505064	0.0145977033
CTH	8.57705398	0.0112731361	0.2385457877
Infor	7.92433313	0.1765522833	0.4178362864
AFS	8.55363151	0.0270740871	0.1564542850
APT	8.45209684	0.0866313681	0.0787825514
TasaActividad	1.30730170	10.9306007769	13.1576304404
TasaParo	0.01958975	21.3039637789	15.0045441660
Ocupados	8.66880711	0.0283545394	0.0005901685
PIB	8.46269279	0.1307378643	0.1405630563
CANE	0.03164342	0.9251453644	43.1311545230
TVF	8.50111166	0.0002558733	0.9952426177
VS	2.97022035	0.2286850605	8.4182925697

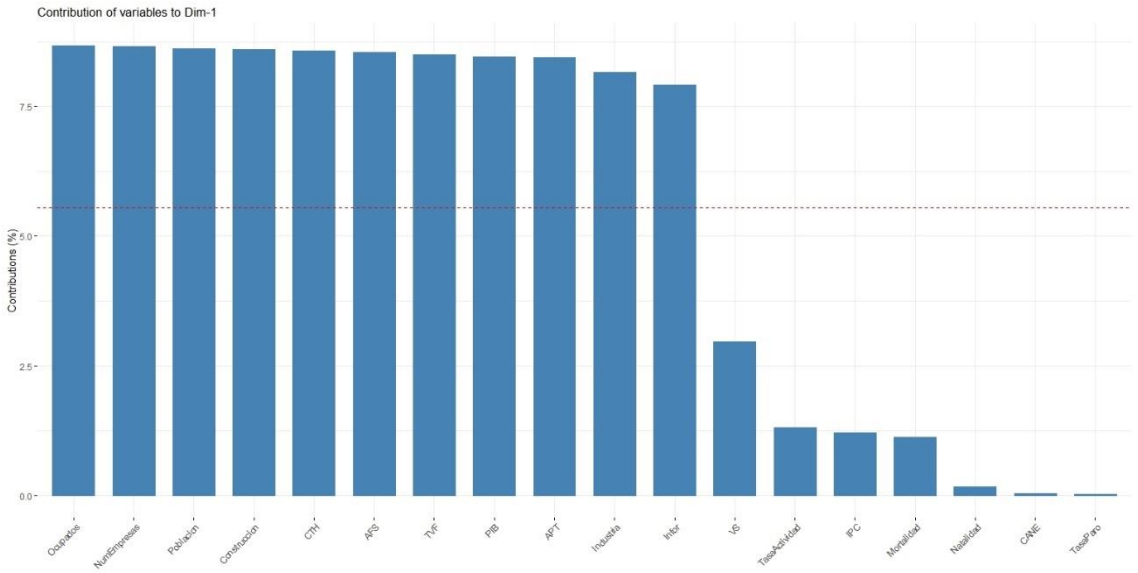
Vemos cómo el 8.62% de la varianza de la variable1 proviene de la población, el 1.13% de la mortalidad, y así sucesivamente.

Las variables que más aportan a la primera, segunda y tercera componente son *Ocupados*, *Mortalidad* y *CANE*, respectivamente.

Graficamos la contribución de las variables a las tres dimensiones, y también las 20 variables que contribuyen en mayor medida a la primera dimensión:

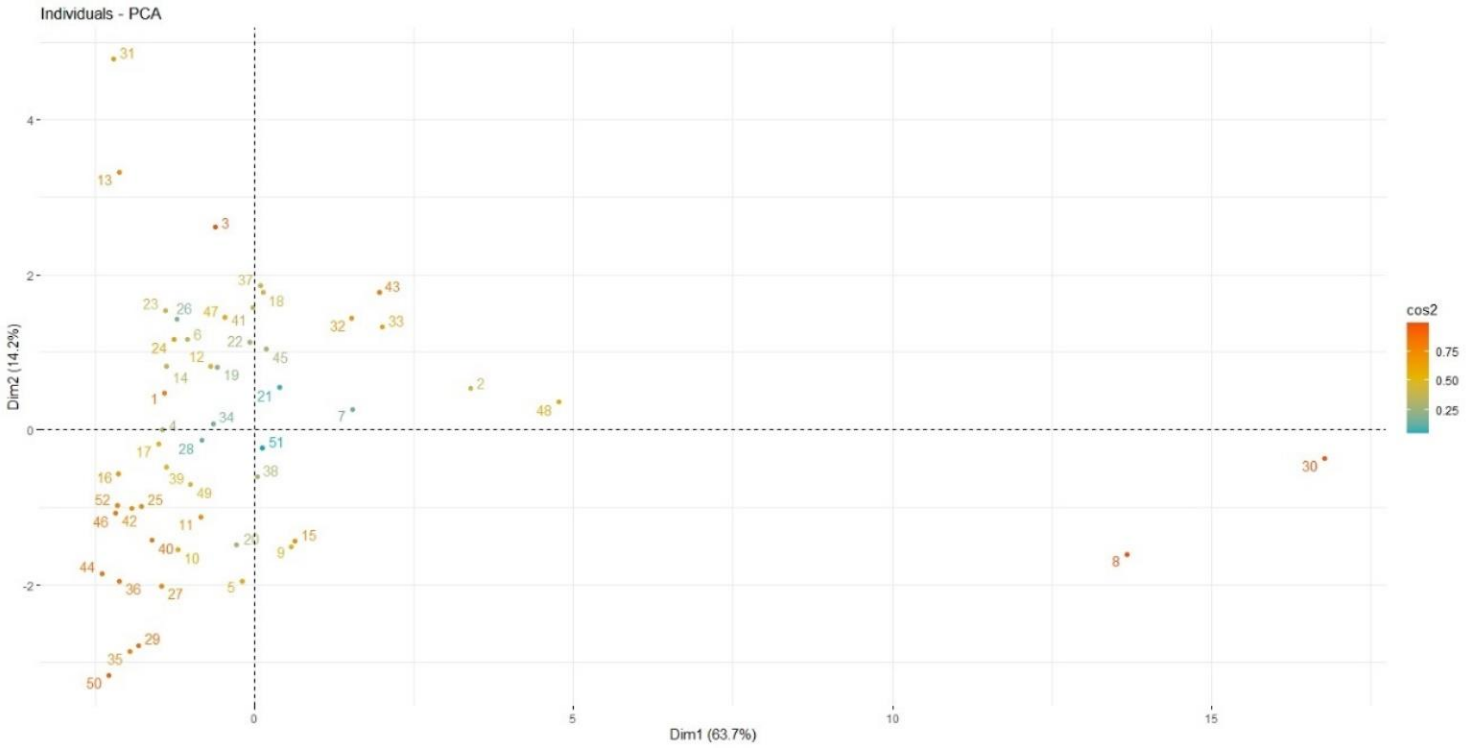
```
corrplot(var$contrib,is.corr=FALSE)

fviz_contrib(fit, choice = "var", axes = 1, top = 20)
```

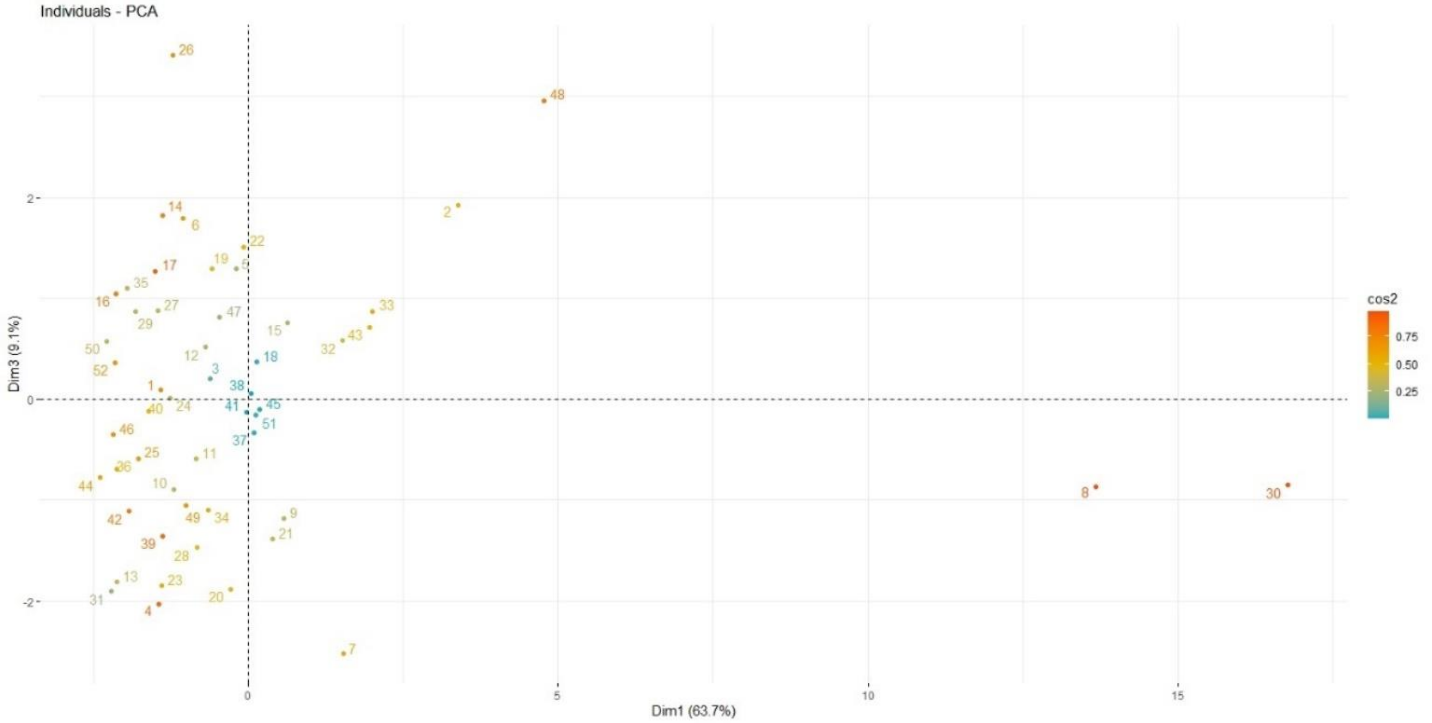

Dim.1
Dim.2
Dim.3
Poblacion
Mortalidad
Natalidad
IPC
NumEmpresas
Industria
Construccion
CTH
Infor
AFS
APT
TasaActividad
TasaParo
Ocupados
PIB
CANE
TVF
VS
43.13
38.82
34.5
30.19
25.88
21.57
17.25
12.94
8.63
4.31
0

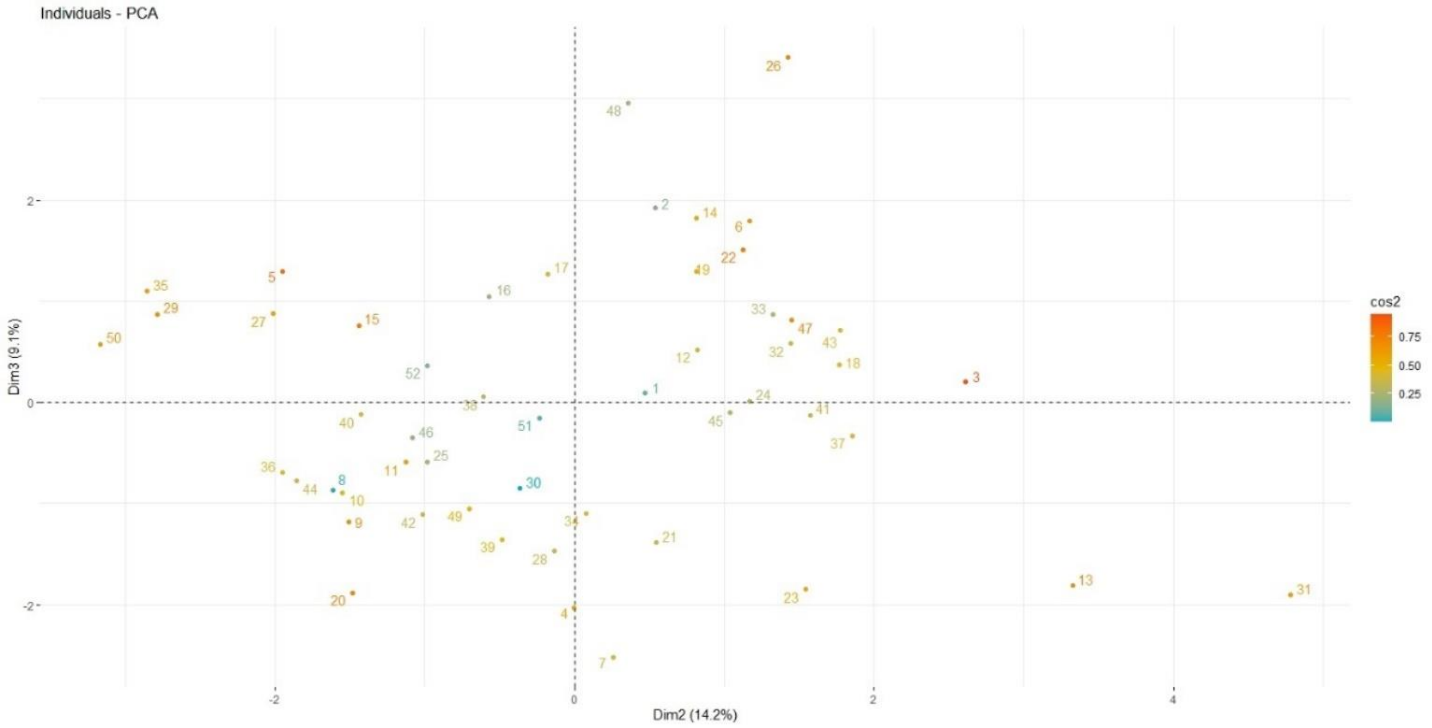

Contribution of variables to Dim-1
Contributions (%)
7.5
5.0
2.5
0.0
Ocupados
NumEmpresas
Poblacion
Construccion
CTH
AFS
TVF
PIB
APT
Industria
Infor
VS
TasaActividad
IPC
Mortalidad
Natalidad
CANE
TasaParo

A continuación, obtenemos los valores de las variables en las nuevas coordenadas y los guardamos en la variable *ind*. Hay que tener en cuenta que las CP están contraídas sobre las variables estandarizadas, por lo que los valores que toman estas variables en las CP's están alrededor de cero. Valores negativos reflejarán entonces valores que están por debajo de la media. Además, representamos los valores en los planos de las nuevas coordenadas. Cabe apuntar que, en los gráficos, colores más azulados indican valores más cercanos a la media:

```
ind<-get_pca_ind(fit)
new_data <- as.data.frame(ind$coord) #Guardamos los datos convertidos en un
nuevo df

fviz_pca_ind(fit, axes = c(1, 2), col.ind = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"), repel = TRUE)

fviz_pca_ind(fit, axes = c(1, 3), col.ind = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"), repel = TRUE)

fviz_pca_ind(fit, axes = c(2, 3), col.ind = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"), repel = TRUE)
```




Individuals - PCA
Dim3 (9.1%)
Dim1 (63.7%)
cos2
0.75
0.50
0.25

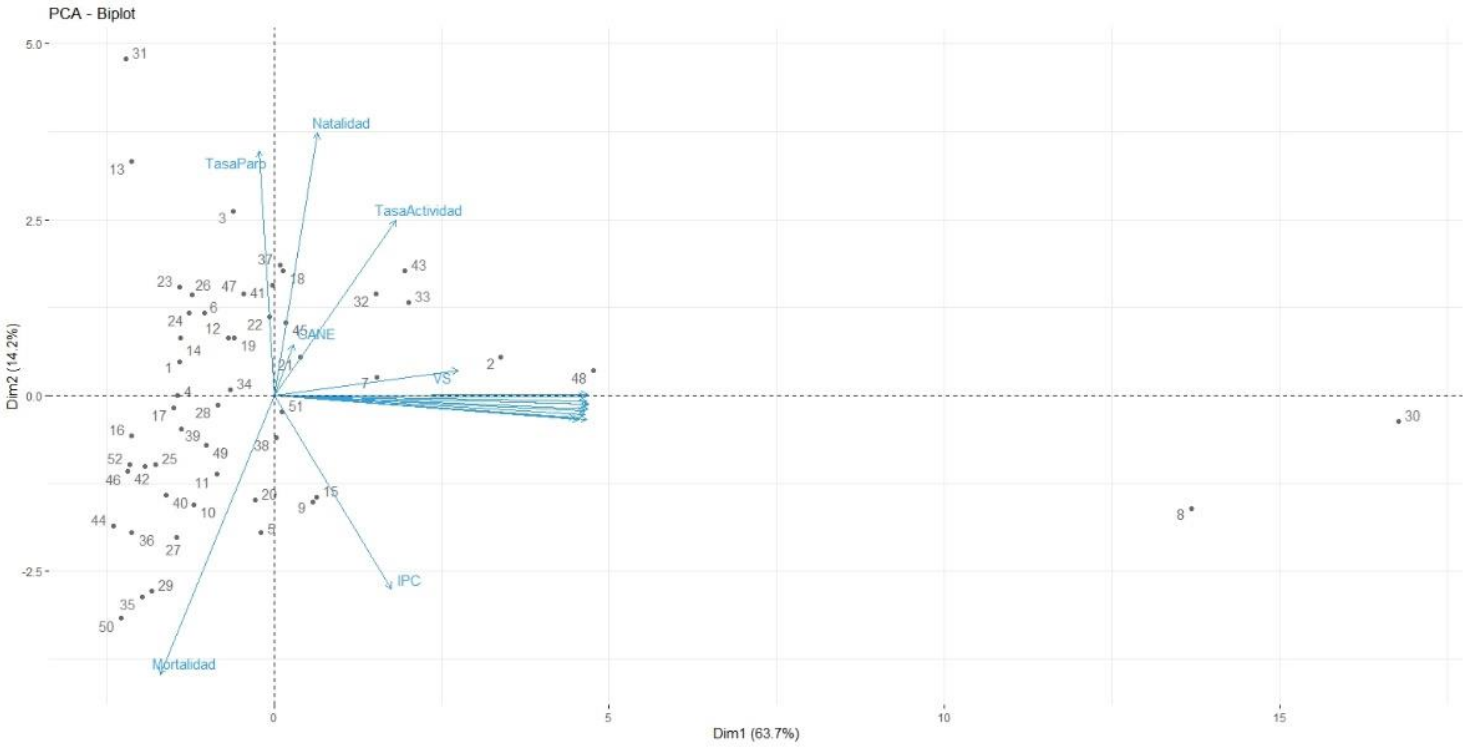

Individuals - PCA
Dim3 (9.1%)
Dim2 (14.2%)
cos2
0.75
0.50
0.25

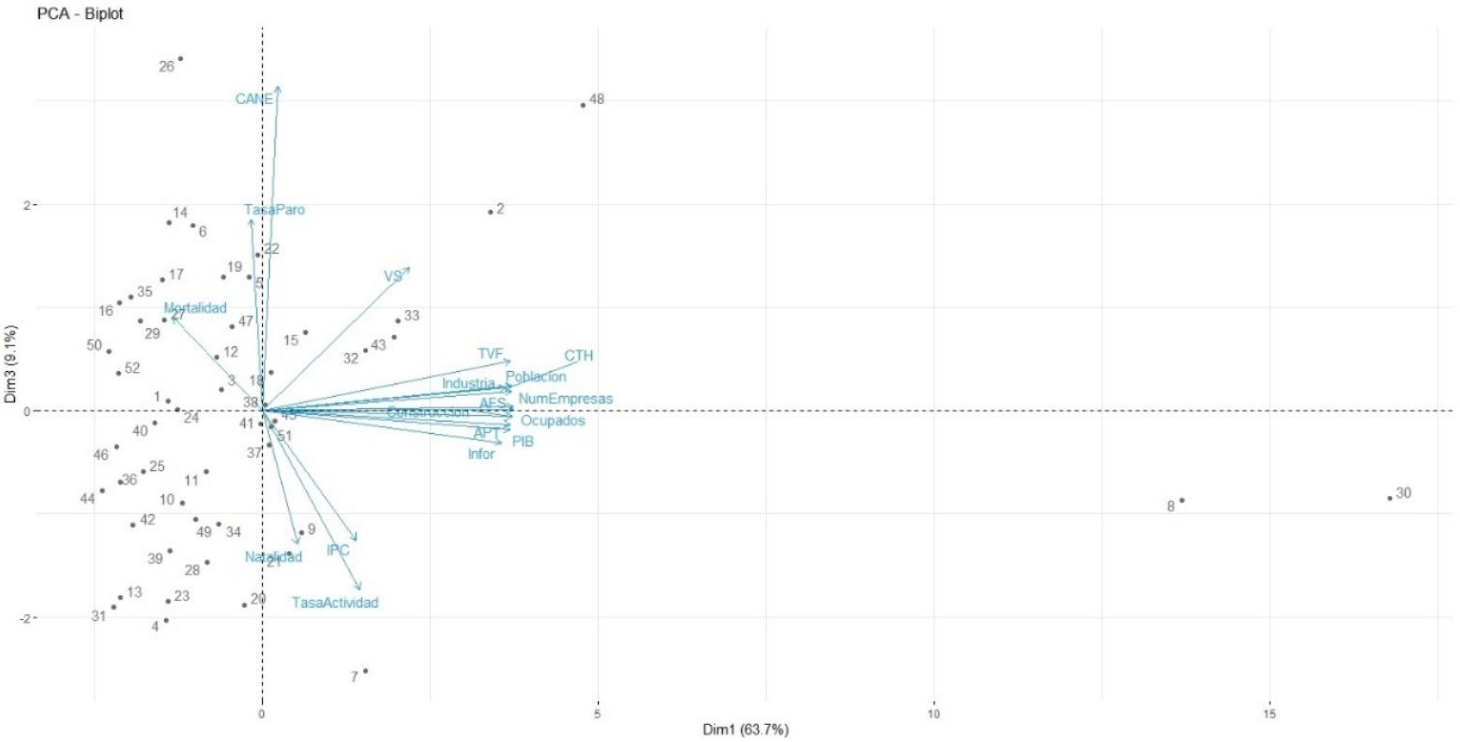
Con los gráficos *biplot* podemos representar simultáneamente las variables originales y las instancias en el nuevo plano:

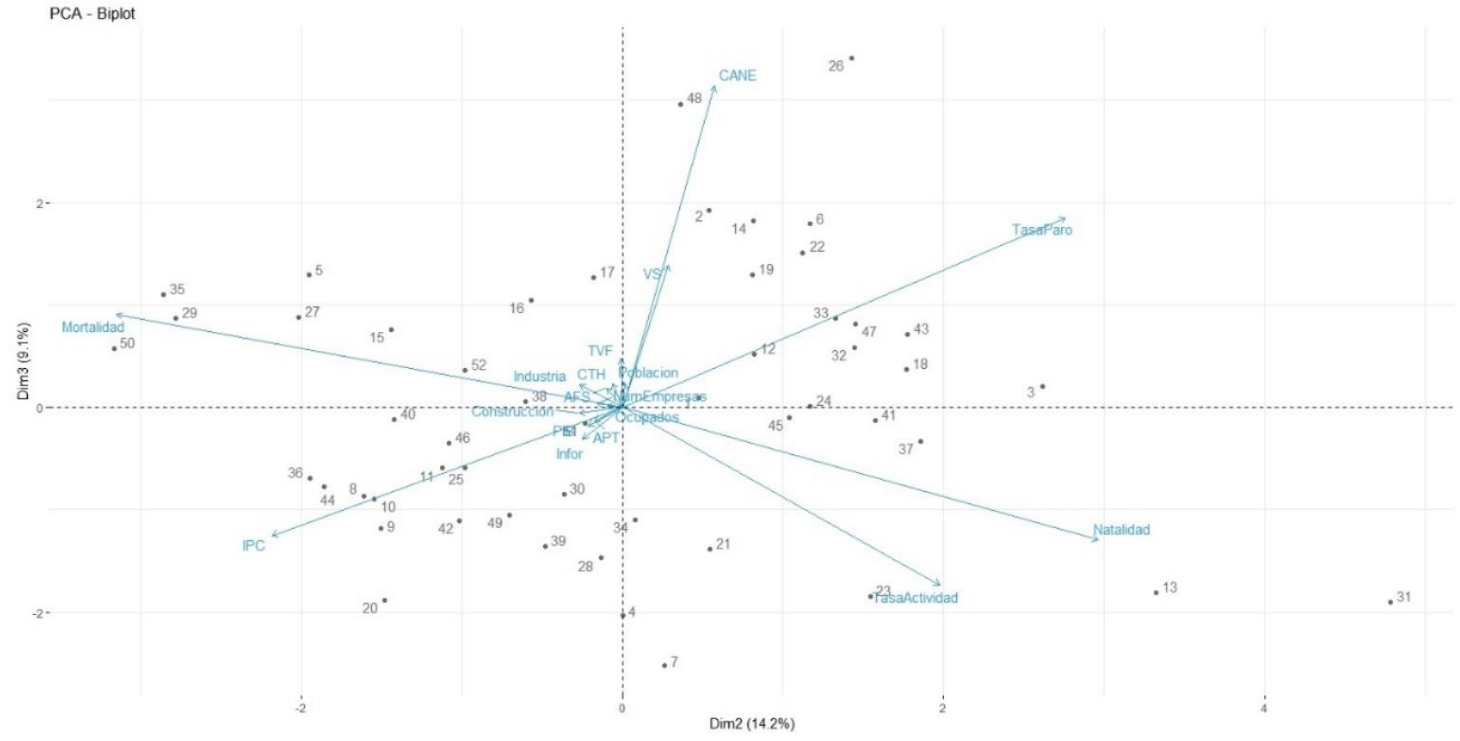
```
fviz_pca_biplot(fit, repel = TRUE, col.var = "#2E9FDF",
                col.ind = "#696969")
```

```
fviz_pca_biplot(fit, axes = c(1,3), repel = TRUE, col.var = "#2E9FDF",
                col.ind = "#696969")
```

```
fviz_pca_biplot(fit, axes = c(2,3), repel = TRUE, col.var = "#2E9FDF",
                col.ind = "#696969")
```




PCA - Biplot
CANE
TasaParo
VS
Mortalidad
TVF
CTH
Industria
Poblacion
AES
NumEmpresas
Construccion
Ocupados
APT
PIB
Infor
Natalidad
IPC
TasaActividad
Dim3 (9.1%)
Dim1 (63.7%)


PCA - Biplot
CANE
TasaParo
VS
Mortalidad
TVF
CTH
Industria
Poblacion
AES
NumEmpresas
Construccion
Ocupados
PIB
APT
Infor
Natalidad
IPC
TasaActividad
Dim3 (9.1%)
Dim2 (14.2%)

Puede llegar a resultar complicado entender por qué unas provincias se sitúan en una cierta región del plano formado por los nuevos componentes, puesto que estos son combinaciones lineales de las variables originales y es difícil recordar qué variables aportan coeficientes más altos o más bajos a esta combinación. Sin embargo, los gráficos biplot facilitan el entendimiento al representar simultáneamente las variables originales. Esto nos permite ver a simple vista qué variables son las que están "tirando" de nuestra instancia hacia un cierto eje.

En los gráficos en los que aparece la segunda dimensión, vemos como destacan las posiciones de provincias como Ceuta o Melilla (31 y 13). Lo hacen, en parte, porque son las provincias en las que la natalidad es mayor, y con una diferencia muy relevante con respecto a las demás. A su vez, vemos cómo el vector asociado a la variable *Natalidad* presenta un ángulo relativamente pequeño con respecto al eje de la segunda dimensión, lo que quiere decir que la esta variable presenta un coeficiente alto en valor absoluto en la combinación lineal que genera la segunda componente principal. En definitiva, la variable Natalidad es una variable relevante a la hora de determinar la posición de una provincia en el eje asociado a la segunda componente, aunque no debemos olvidar que no es la única. Estamos hablando específicamente de la natalidad porque destaca para las provincias de Ceuta y Melilla y sirve como ejemplo, pero hay otras que también influyen en su posición con respecto a este eje. Destacan en este sentido la tasa de paro, la tasa de actividad, el IPC o la mortalidad.

Es interesante también comentar cómo la generación de componentes principales puede cuantificar posibles nuevas variables que en un principio no teníamos definidas, indicando la presencia de aglomeraciones de provincias que en un principio se mostraban ocultas. En este sentido destaca la posición de Madrid o Barcelona con respecto a la primera componente principal, que podría estar indicando de cierta forma el nivel de desarrollo económico. Vemos además, cómo la primera dimensión tiene mucha correlación con variables como PIB, Ocupados, Población, NumEmpresas, Industria, Construcción, etc...que efectivamente nos parece que pueden ser buenos indicadores del nivel de desarrollo de una provincia.

Siguiendo esta línea de pensamiento, podríamos definir un índice que mida el desarrollo económico en función de la posición de las provincias en la primera componente principal. Los otros dos componentes decidimos descartarlos porque Madrid o Barcelona no ocupan una posición llamativa en ellos a pesar de que son provincias que podemos utilizar como referencia a la hora de medir el desarrollo económico. El valor de la primera coordenada de Madrid es de 16.778, siendo la más alta. Vamos a considerar que Madrid tiene aún margen de mejora en el aspecto económico, por lo que no vamos a tomar este valor como máximo. En cambio, nuestro máximo lo vamos a fijar en 20. Con respecto al valor más bajo de la primera coordenada, lo encontramos en la provincia de Soria (el tercer PIB más bajo) y es de -2.399. Nuevamente, vamos a considerar un valor mínimo más pequeño, por ejemplo, de -5.0. Con estos datos definimos nuestro índice de desarrollo normalizado (oscila entre 0 y 1) como:

$$Ind = \frac{Coord1 - Valor\ Mínimo}{Valor\ Máximo - Valor\ Mínimo} = \frac{Coord1 + 5}{25}$$

Teniendo en cuenta que el valor que toma la primera coordenada para la provincia de melilla es de -2.218, tenemos que:

$$Ind_{Madrid} = 0.871\ ;\ Ind_{Melilla} = 0.111$$

Recordemos que el valor de las coordenadas de las provincias en las nuevas componentes lo guardamos en la variable *ind*. Podemos tabularlas para acceder a ellas con la siguiente línea de código:

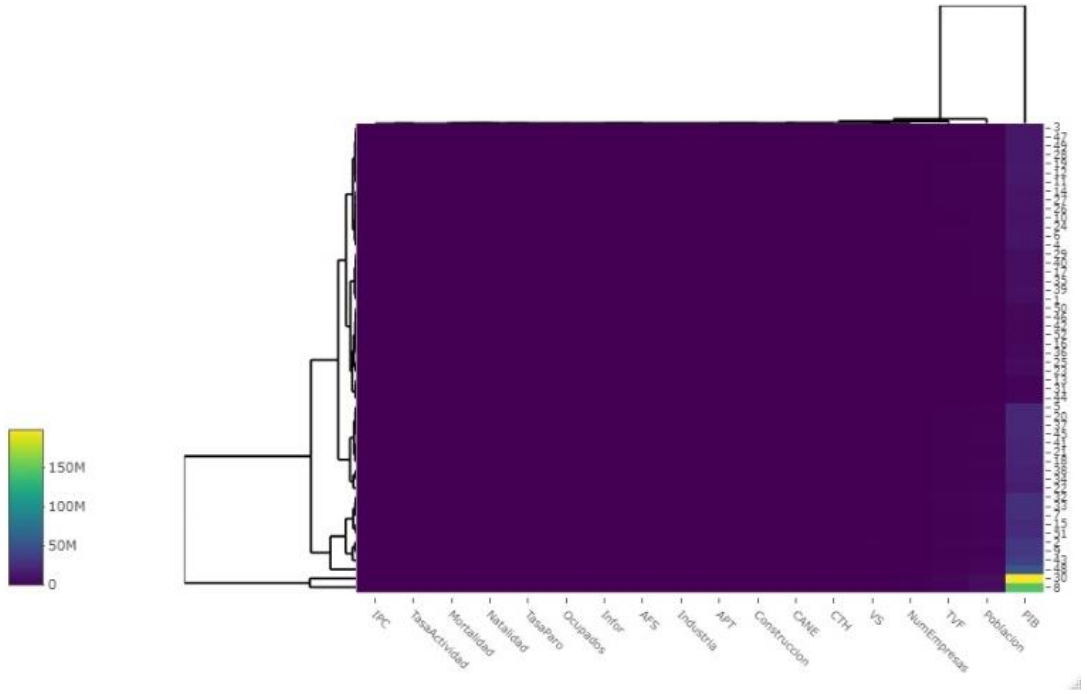
```
 knitr::kable(ind$coord, digits =3,caption = "Valores de los individuos en las
Cp", "simple")
```

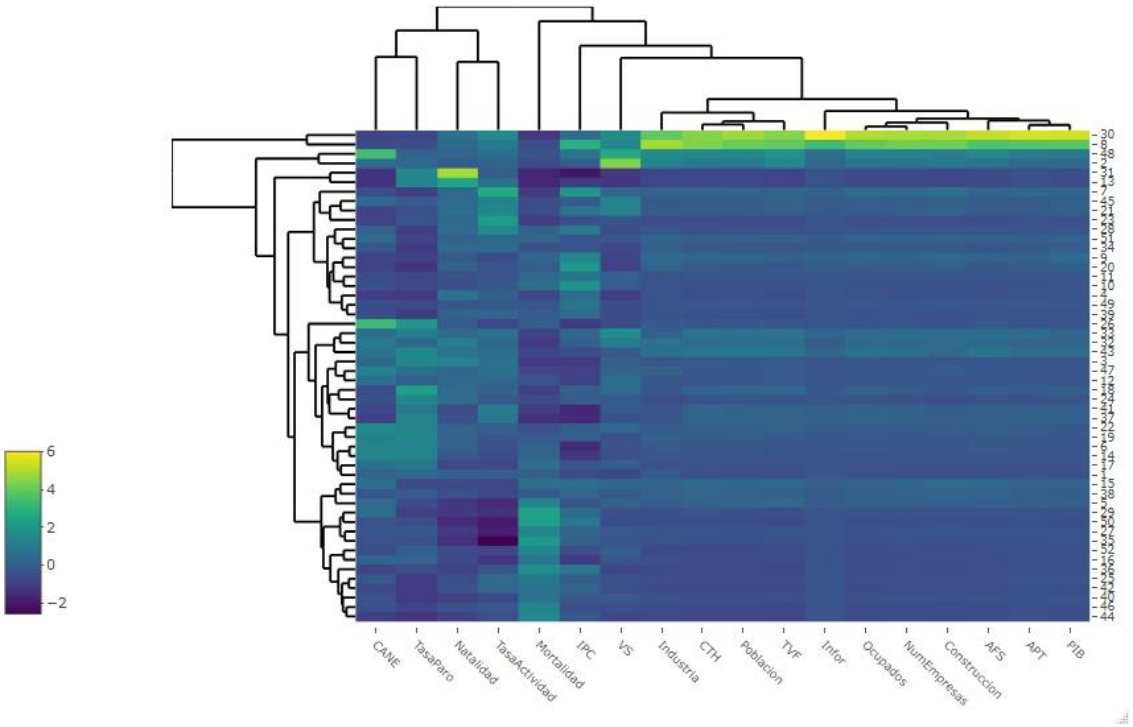
Cabe apuntar que todas las variables originales contribuyen en este índice de desarrollo económico, puesto que todas tienen un coeficiente distinto de cero a la hora de definir la primera componente principal. Eso sí, las variables con un coeficiente más alto en la combinación lineal serán variables que pesen más a la hora determinar el índice.

A continuación, representamos un mapa de calor la matriz de datos, estandarizada y sin estandarizar, para ver si detectamos grupos de provincias.

Intuimos que la mejor opción será estandarizar los datos para que la diferencia en la magnitud de variables no influya a la hora de determinar los grupos. Para ello nos serviremos de las librerías *heatmapy, Cluster y NbClust*:

```
heatmaply(dep_datos, seriate = "mean", row_dend_left = TRUE, plot_method =
"plotly")

dep_datos_st <- as.data.frame(scale(dep_datos)) #Estandarización

heatmaply(dep_datos_st, seriate = "mean", row_dend_left = TRUE, plot_method =
"plotly")
```



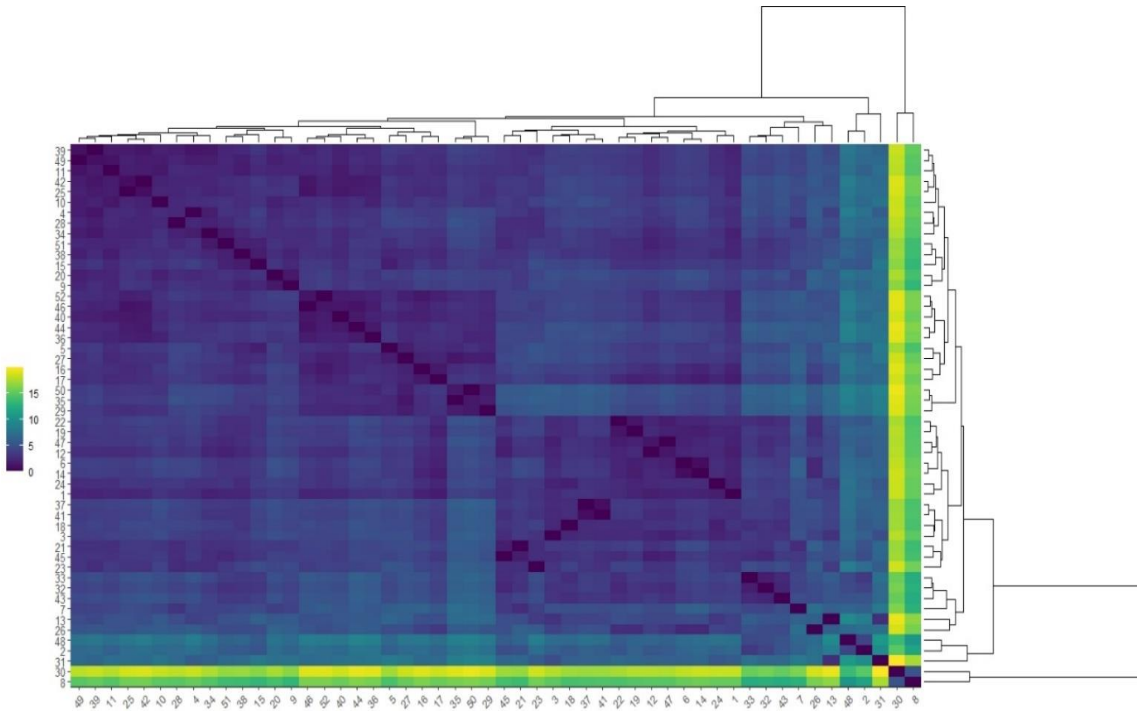


Vemos como en el premier *plot* no apreciamos nada debido a que los datos no estaban estandarizados. La variable PIB toma valores mucho mayores que el resto, luego el resto de las variables son representadas con el mismo color para todos los países, que representa un valor mucho menor. Esto no nos permite observar diferencias entre países.

Tras estandarizar podemos ver algo más. Observamos cómo Madrid y Barcelona toman valores parecidos para muchas variables, más grandes en comparación a el resto de las provincias. También observamos características diferenciadoras en provincias como Ceuta y Melilla o Murcia, Málaga y Sevilla. Esto tiene sentido, pues son provincias que comparten muchas peculiaridades entre sí en gran medida por la proximidad e historia que comparten.

Pasamos ahora a realizar un análisis jerárquico de clusters. Para ello calculamos la matriz de distancias entre cada una de las provincias y las representamos en un mapa de calor junto con el dendrograma con las agrupaciones entre provincias:

```
d <- dist(dep_datos_st, method = "euclidean")
ggheatmap(as.matrix(d), seriate="mean")
```



A la vista del dendrograma podríamos recomendar la generación de cinco grupos diferentes. En un principio tendríamos tres grupos principales. Uno formado por las provincias de Madrid y Barcelona (índices 8 y 30) y los otros dos formados a partir de la ramificación de la otra rama principal. De estos dos últimos grupos hay uno que contiene las provincias con índices 31, 2 y 48 y otro que contiene a todas las demás. Los cinco grupos diferentes los alcanzaríamos a través de una doble ramificación de este tercer grupo que contiene a la mayoría de las provincias.

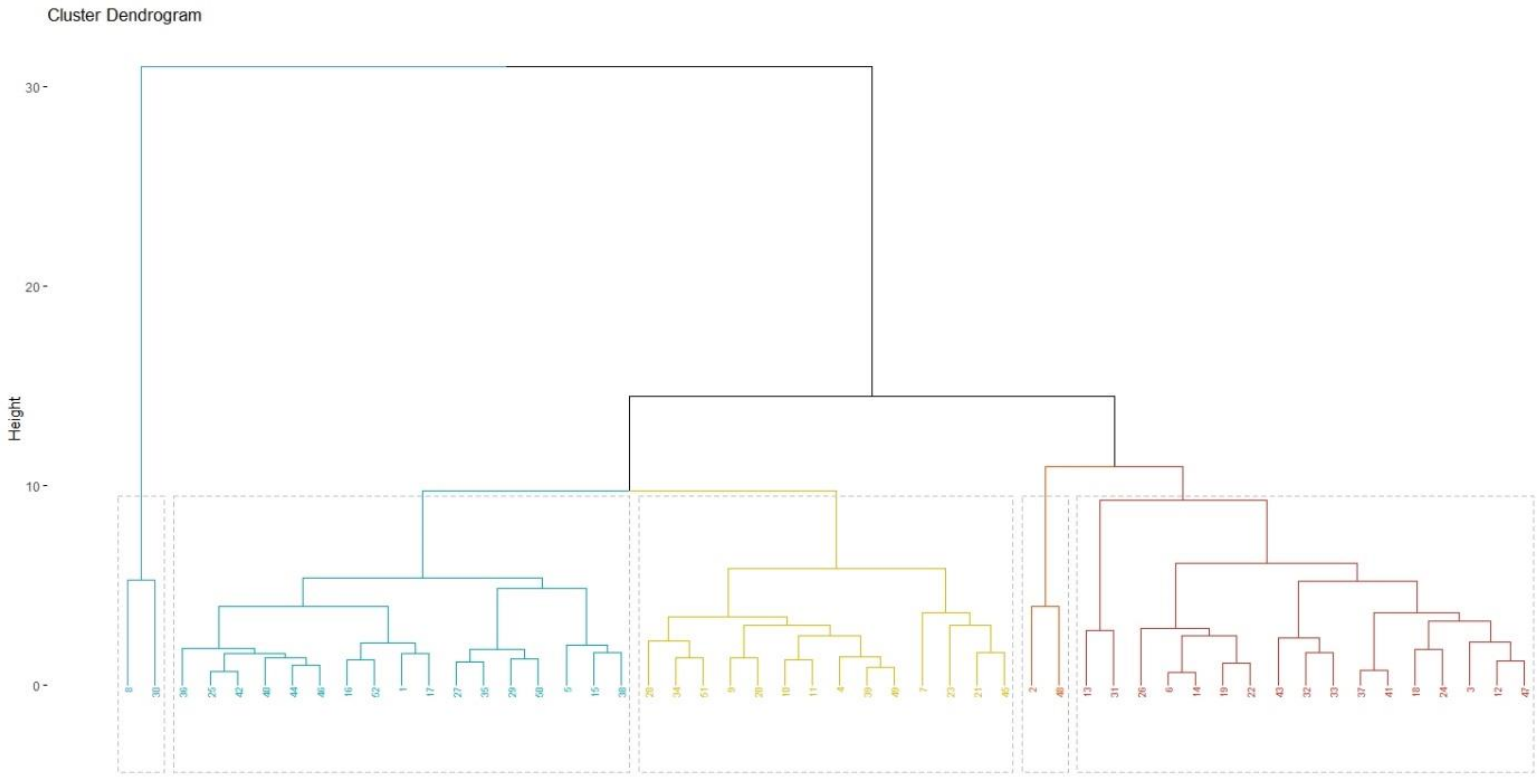
Como hemos visto, la decisión de generar cinco grupos la hemos tomado por motivos visuales, porque son fáciles de distinguir dentro del dendrograma. Además, nos parece una buena cantidad con respecto al numero de provincias que tenemos.

Cabe decir que al cambiar el criterio por el que medimos la distancia entre los clusters estamos también cambiando los dendrogramas. Por ejemplo, probemos ahora con el criterio *Ward.D2*, que calcula la distancia como la raíz de la distancia euclídea entre las medias de los clusters:

```
res.hc_st <- hclust(d, method="ward.D2")
fviz_dend(res.hc_st, cex = 0.5)
```

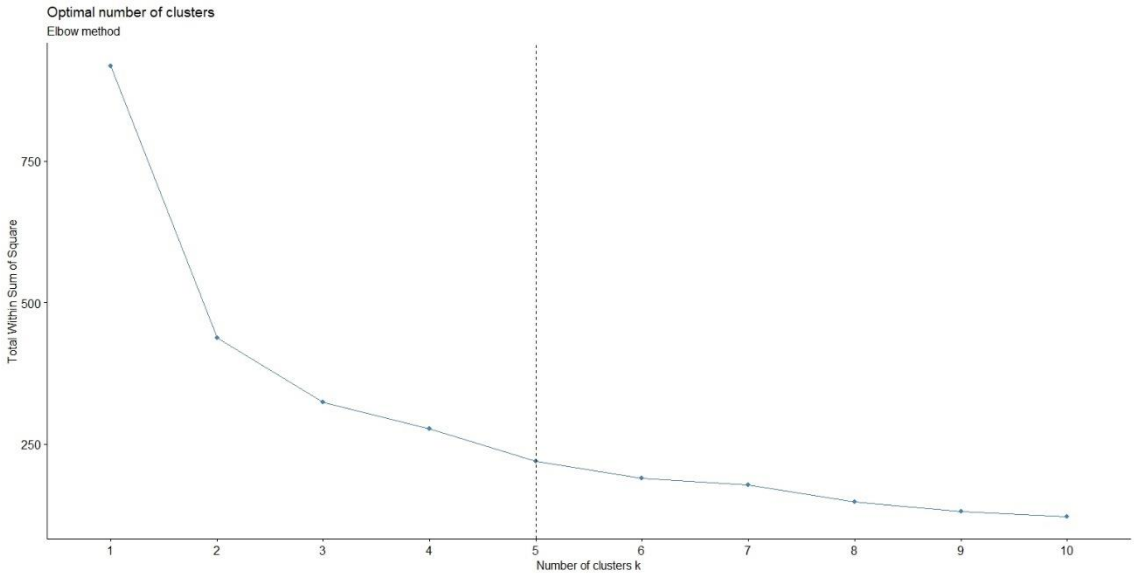
En este caso generamos un dendrograma diferente, en el que visualmente también creemos que la diferenciación en cinco grupos puede ser una buena opción. Pasamos realizarla y a graficarla, mostrando las instancias pertenecientes a cada grupo:

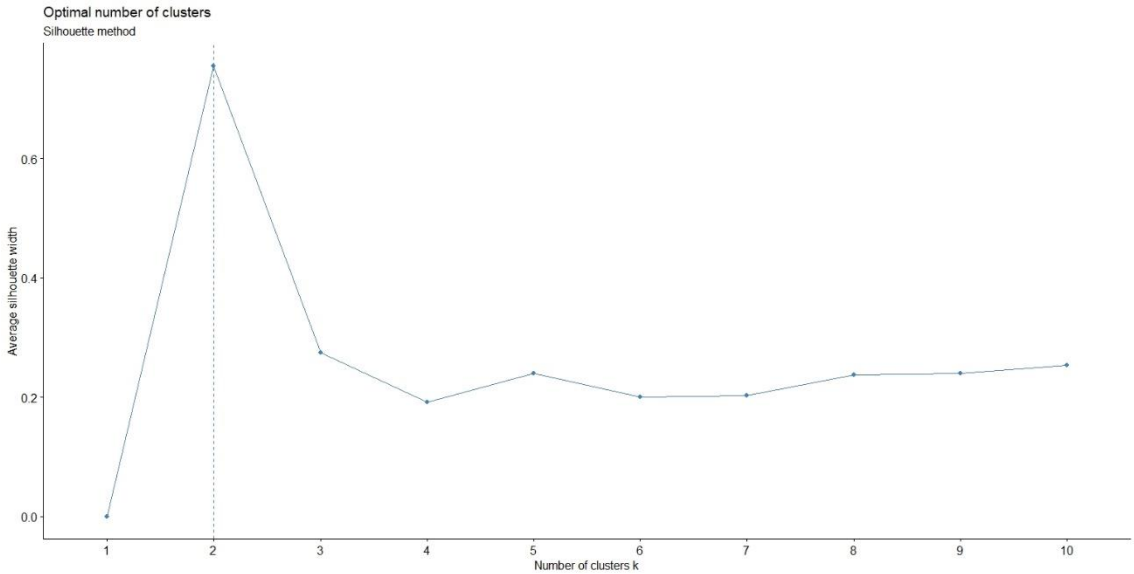
```
grp <- cutree(res.hc_st, k = 5)
fviz_dend(res.hc_st, k = 5,
          cex = 0.5, # label size
          k_colors = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07","#D73027"),
          color_labels_by_k = TRUE,
          rect = TRUE)
```



Vamos ahora a calcular qué número de clusters es el óptimo según los criterios de Elbow y Silhouette en lugar de hacerlo visualmente. Las dos primerias de código lo único que hacen es fijar la semilla, de tal manera que obtengamos siempre los mismos resultados cada vez que corramos el código. El resto del código aplica primero el criterio de Elbow y después el de Silhouette:

```
RNGkind(sample.kind ="Rejection")
set.seed(1234)

fviz_nbclust(dep_datos_st, kmeans, method = "wss") +
  geom_vline(xintercept = 5, linetype = 2)+
  labs(subtitle = "Elbow method")

fviz_nbclust(dep_datos_st, kmeans, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```





Como vemos en las imágenes, el criterio de Elbow nos recomienda agrupar en cinco clusters diferentes, mientras que el de Silhouette nos recomienda dos. Podemos pensar que quizás tenga más sentido la cifra que propone el primer criterio en base al número de provincias que tenemos, siendo el segundo criterio quizás demasiado conservador al distinguir probablemente entre Madrid y Barcelona y el resto de las provincias. Además, la cifra del criterio de Elbow coincide con la que habíamos elegido en nuestro análisis manual, lo que siempre es buena señal.

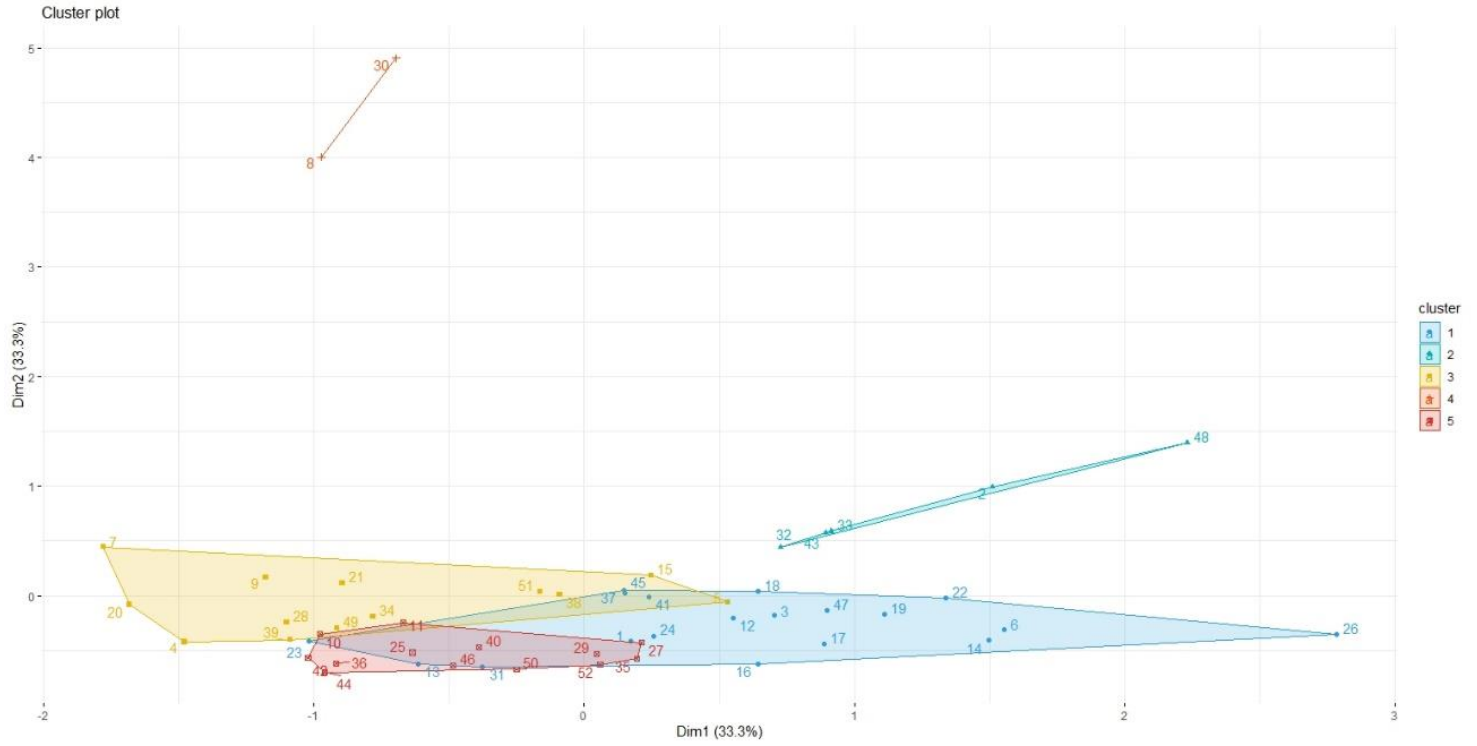
A continuación, codificamos el mismo proceso realizado anteriormente con respecto a la definición de clusters pero con el dataframe que guardamos con el nombre de *new_data*, en el que se recogían los datos transformados tras haber realizado la reducción de dimensiones para quedarnos con tres componentes principales. Además, comprobamos que el criterio de Elbow también considera que el número óptimo de agrupaciones es cinco. Esto nos permitirá posteriormente representar los clusters en los planos formados por las componentes principales:

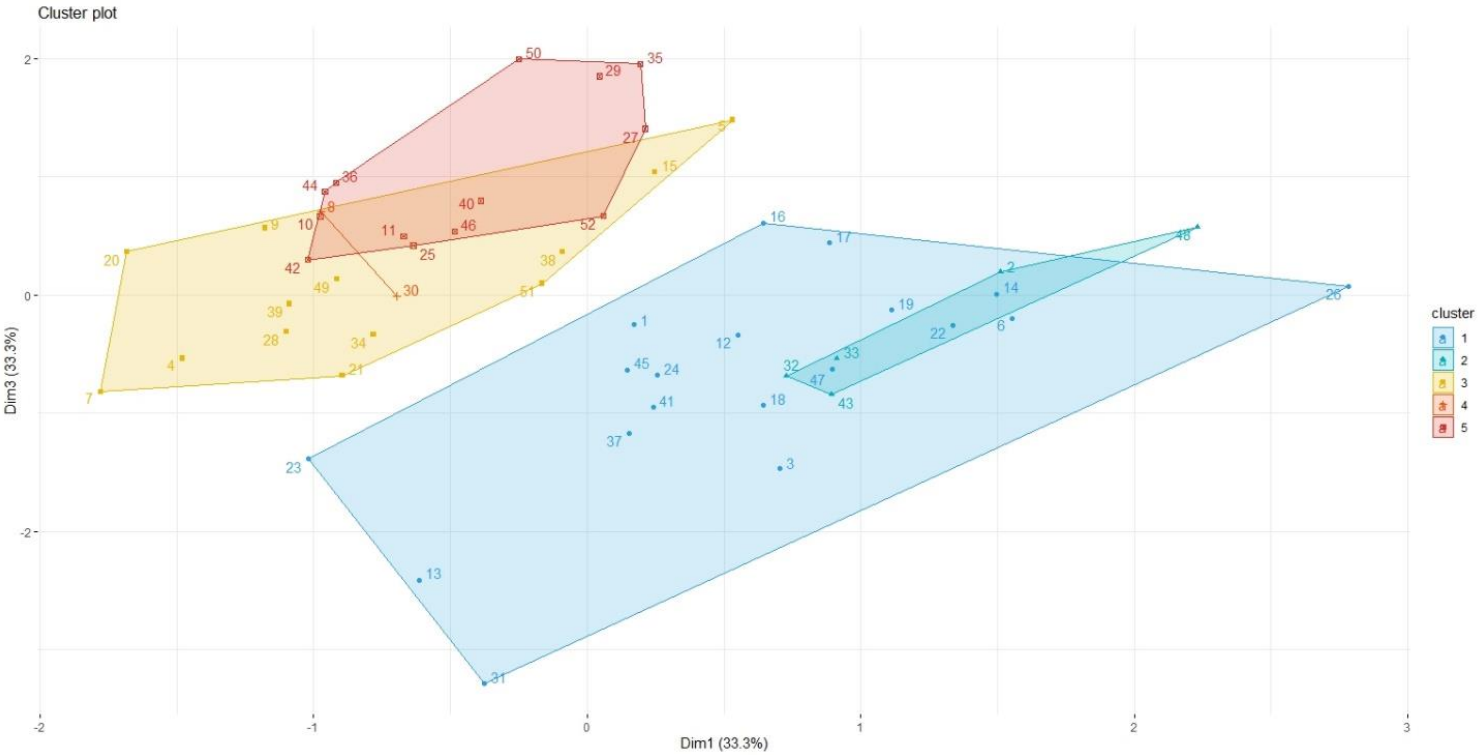
```
d2 <- dist(new_data, method = "euclidean")
res.hc_st2 <- hclust(d2, method="ward.D2")
fviz_dend(res.hc_st2, cex = 0.5)

fviz_nbclust(new_data, kmeans, method = "wss") +
  geom_vline(xintercept = 5, linetype = 2)+
  labs(subtitle = "Elbow method")

grp2 <- cutree(res.hc_st2, k = 5)
```

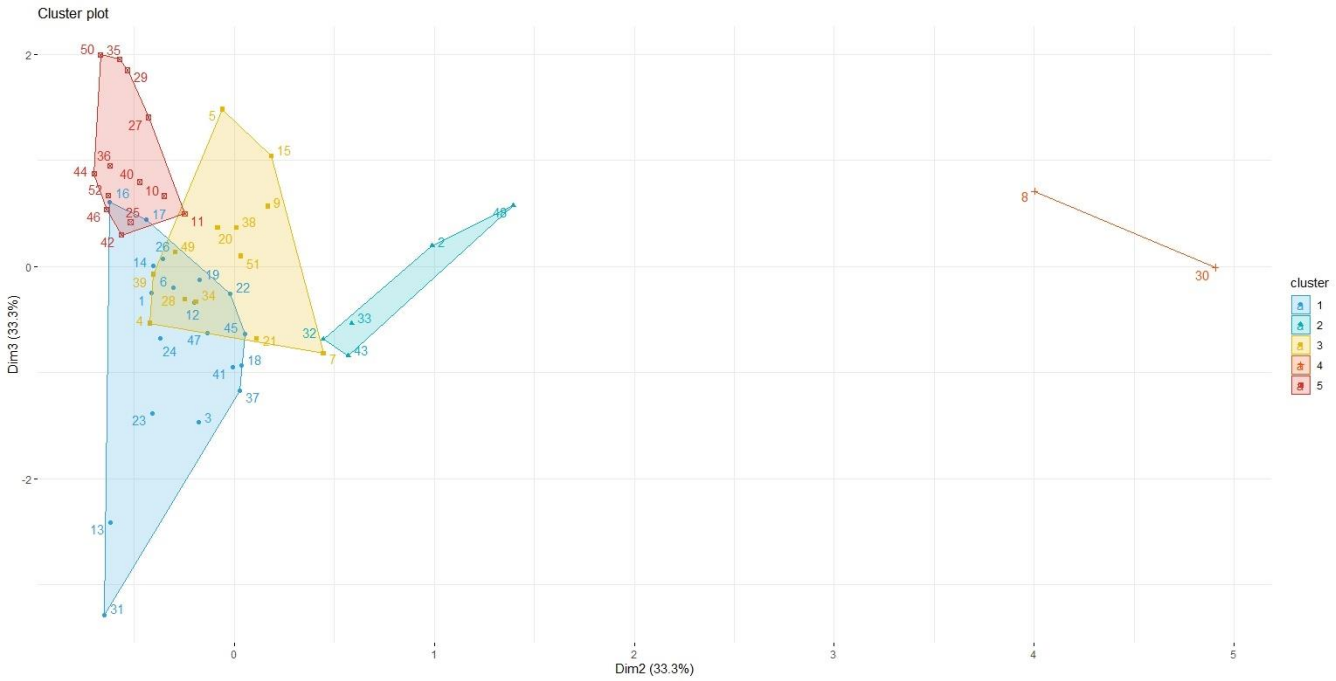
Graficamos sobre los tres planos diferentes:

```
fviz_cluster(list(data = new_data, cluster = grp2), axes=c(1,2), palette
             = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07", "#D73027"),
ellipse.type =
               "convex", repel = TRUE, show.clust.cent = FALSE, ggtheme =
               theme_minimal())

fviz_cluster(list(data = new_data, cluster = grp2), axes=c(1,3), palette
             = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07", "#D73027"),
ellipse.type =
               "convex", repel = TRUE, show.clust.cent = FALSE, ggtheme =
               theme_minimal())
```

```
fviz_cluster(list(data = new_data, cluster = grp2), axes=c(2,3), palette
             = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07", "#D73027"),
ellipse.type =
               "convex", repel = TRUE, show.clust.cent = FALSE, ggtheme =
               theme_minimal())
```



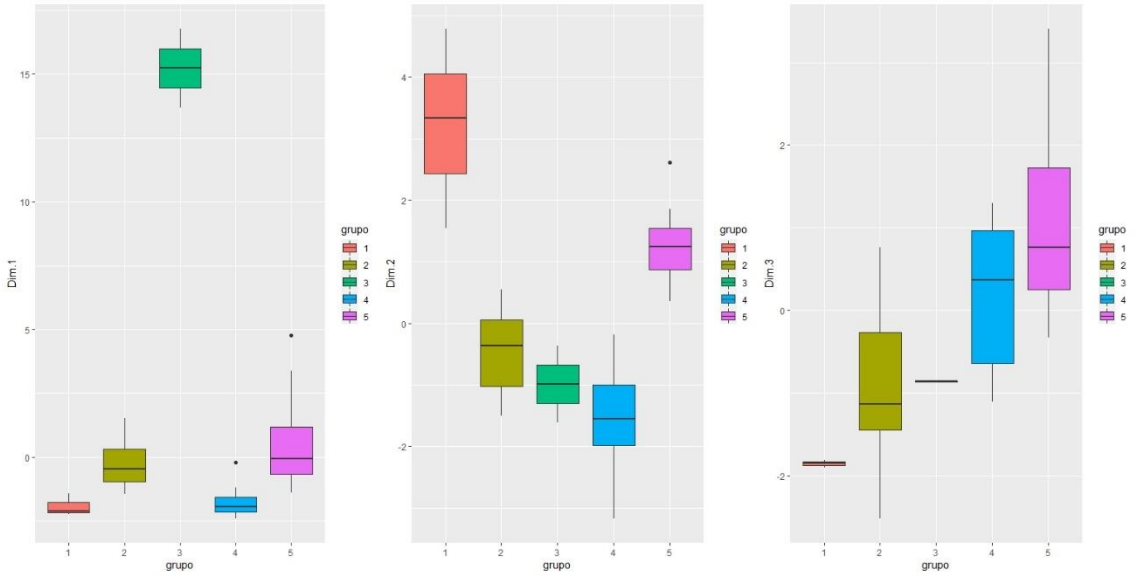




Se observa cómo la agrupación que mejor se distingue es la formada por Madrid y Barcelona, como ya habíamos analizado previamente. Otra agrupación que también destaca es la formada por las provincias de Valencia, Alicante, Sevilla, Murcia, Málaga (cluster número 2); que se distingue del resto también en el eje que habíamos decidido que podía medir el desarrollo económico. El resto de clusters se solapan algo más, distinguiéndose sobre todo en la dimensión que se encontraba más correlacionada con variables como la Tasa de paro, la natalidad o la mortalidad (dimensión 3 en el gráfico superior). De esta forma encontramos un cluster del que forman parte Ourense (35), Zamora(50), Lugo(29) o León(27) y otro que recoge provincias como Melilla(31), Ceuta(13), Guadalajara(23) o Almería(3). El primer grupo de provincias tiene una mortalidad muy alta y una natalidad muy baja. Con el segundo grupo ocurre lo contrario. Pequeños análisis como los que acabamos de realizar nos ayudan a comprender por qué los clusters agrupan a ciertas provincias y la dinámica y diferencias que existen entre ellos.

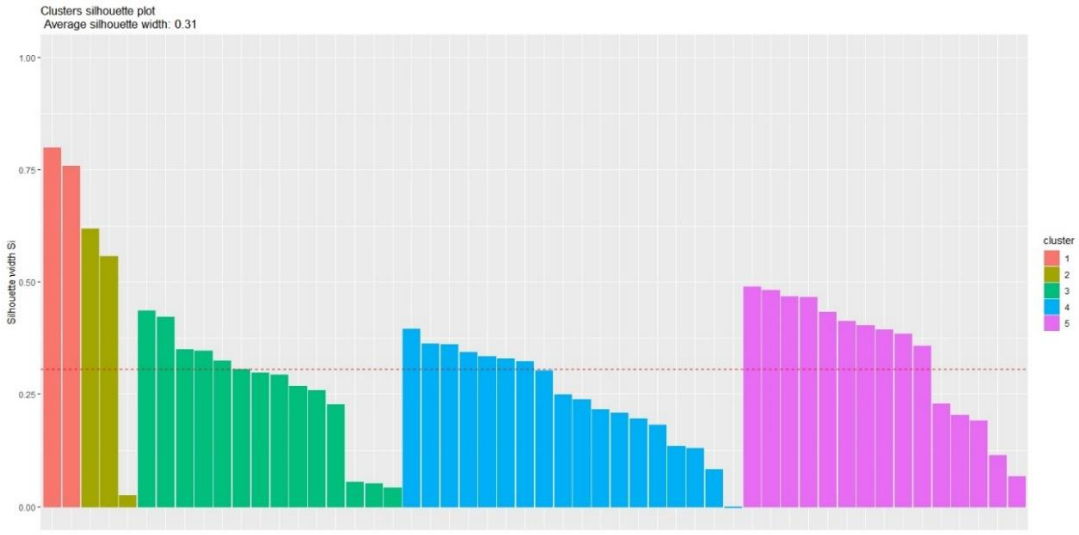
También puede resultar útil las representaciones de gráficas de las variables (en este caso las tres componentes principales) en forma de boxplot para cada cluster:

```
km.res <- kmeans(new_data, 5)
new_data$grupo<- as.factor(km.res$cluster)
g1<- ggplot(new_data, aes(x=grupo, y=Dim.1, fill=grupo)) + geom_boxplot()
g2<- ggplot(new_data, aes(x=grupo, y=Dim.2, fill=grupo)) + geom_boxplot()
g3<- ggplot(new_data, aes(x=grupo, y=Dim.3, fill=grupo)) + geom_boxplot()
gridExtra::grid.arrange(g1, g2, g3, ncol=3, nrow=1)
```



Finalmente terminamos evaluando la calidad de los clusters que hemos obtenido. Para ello empleamos el coeficiente de Silhouette, que oscila entre 0 y 1 y nos indica si una instancia está bien clasificada dentro del cluster, siendo uno si está muy bien clasificada. Lo que hacemos es graficarlo para todas las provincias, mostrando al mismo tiempo la media del coeficiente. Para ello empleamos la función *fviz_silhouette* del paquete *factoextra*:

```
sil <- silhouette(km.res$cluster, dist(new_data))

rownames(sil) <- rownames(new_data)

print(mean(sil[,"sil_width"]))

fviz_silhouette(sil)
```



Tenemos un valor medio del coeficiente de silhouette de 0.31, lo que nos asegura que existe una cierta separación entre los clusters. Esto nos asegura que nuestro modelo, lejos de ser perfecto, sí nos está ayudando a entender los datos y podría jugar un buen papel si quisiésemos implementar métodos predictivos a partir de este punto.