PR1-P2023

Alumno: Borja Villena Pardo

Práctica – Borja Villena Pardo – Intento 1

- **La resolución de la práctica (memoria técnica detallada). Importante: debéis especificar a que intento de la tabla corresponde.**
- **El código en R.**
- **Las imágenes y/o figuras que se os pidan.**

Datos:

- **id:** identificador de la sección censal.
- **rent:** renta bruta por persona.
- **inc_sal:** ingresos provenientes del salario.
- **inc_ret:** ingresos provenientes de pensiones de jubilación.
- **inc_emp:** ingresos provenientes de prestaciones del paro.
- **inc_non:** ingresos provenientes de otros tipos de prestaciones.
- **inc_oth:** otros ingresos.
- **gini:** coeficiente de Gini que mide la desigualdad.
- **dist8020:** relación de renta entre el percentil 80 (P80) y el percentil 20 (P20) − P80/P20.
- **mean_age:** edad media de la población.
- **perc_chil:** porcentaje de población menor de 18 años.
- **per_ret:** porcentaje de población mayor de 65 años.
- **home_size:** tamaño medio del hogar (m^2).

Pregunta 0. Los resultados del informe corresponden al primer intento:

Respuesta:

- V = mean_age
- C = 2
- Copiamos código R usado para conseguir la respuesta: NA

Pregunta 1. [10 %] Para empezar, generar la matriz X a partir del *dataframe* **var_df** utilizando, por ejemplo, la instrucción **as.matrix.** Aseguraros de eliminar el valor de la primera columna ('id') ya que no proporciona ninguna información relevante. Comprobar, también, que X es una matriz y no un *dataframe* utilizando la siguiente instrucción:

```
> class (X)
[1] " matrix " " array "
```

Responder: ¿cuántas secciones censales tiene la ciudad?

Respuesta:

- La ciudad tiene **61** secciones censales.
- Copiamos código R usado para conseguir la respuesta:

```
> #Abrimos archivo variables.csv y lo nombramos como var_df
> var_df <- read.csv("C:\\Users\\usuario\\Documents\\4. UOC\\1° Álgebra Lineal\\Reto 4\\variables.csv")
>
> #Imprimimos archivo para ver su contenido
> fix(var_df)
> var_df
>
> #Comprobamos que var_df es de tipo dataframe
> class(var_df)
[1] "data.frame"
>
```

```
> #Eliminamos la primera columna llamada 'id'. Para ello vamos a usar el paquete "dplyr"
> library(dplyr)
> var_df <- select(var_df, -id)
>
> #Comprobamos que realmente se ha eliminado la columna
> fix(var_df)
> var_df
>
> #Convertimos var_df en una matriz
> var_matrix <- as.matrix(var_df)
>
> #Comprobamos que ahora var_df es una matriz
> class(var_matrix)
[1] "matrix" "array"
>
> #Buscamos la dimensión que tiene la matriz, y podemos observar que nos devuelve
> # 61, 12, es decir, contamos con 61 observaciones que coinciden con el número de
> #secciones censales, y 12 columnas que coinciden con el número de variables:
> dim(var_matrix)
[1] 61 12
```

Pregunta 2. [10 %] Como alcaldables, os interesa tener una primera impresión general de las variables medidas y explorar los datos en crudo. Una de las características interesantes a estudiar es la razón (M/m) entre el valor máximo (M) y el mínimo (m) de una variable. Calcular la razón de la variable V.

Respuesta:

- La razón de la variable V es igual a **1.53**
- Copiamos código R usado para conseguir la respuesta:

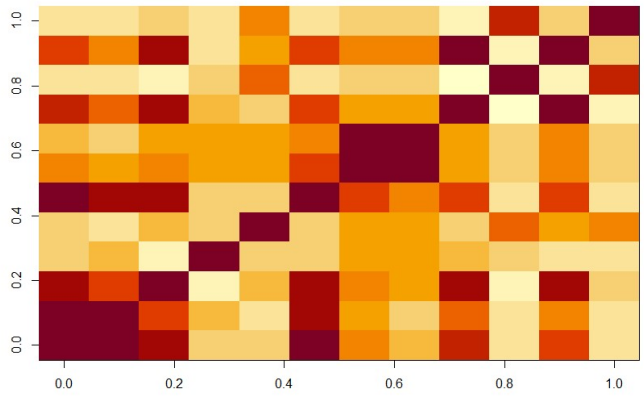
```
> #En este intento, la variable V es igual a la variable 'mean_age'. Para calcular
> #la razón de 'mean_age' vamos a detectar los valores máximo y mínimo de la variable.

> #Asignamos a V la columna 'mean_age' de la matriz var_df
> V <- var_matrix[,9]
>

> #Definimos máximo como el valor 'máximo' de V, y definimos 'mínimo'
> #como el valor mínimo de V
> maximo <- max(V)
> minimo <- min(V)
>
> fix(maximo)
> maximo
[1] 46.9
> fix(minimo)
> minimo
[1] 30.7
>

> #Calculamos la razón entre el valor Máximo y el valor Mínimo de V (M/m).
>#Guardamos valor en 'razon'
> razon <- maximo/minimo
> fix(razon)
> razon
[1] 1.527687

> #Redondeamos resultado a dos decimales
> round(razon, digits=2)
[1] 1.53
```

Pregunta 3. [15 %] Para poder realizar el análisis de componentes principales debéis, inicialmente, normalizar los datos y guardarlos a la variable ***Xs***, como se muestra a la Sección 2.1 de los apuntes del módulo. Una vez normalizados, calcular la matriz de covarianzas de los datos; guardarla en la variable ***CXs*** y adjuntarla 1 como una imagen. Finalmente, indicar cuales son el par de variables (distintas) con mayor covarianza (en valor absoluto) y el par con menor covarianza (en valor absoluto).

Respuesta:

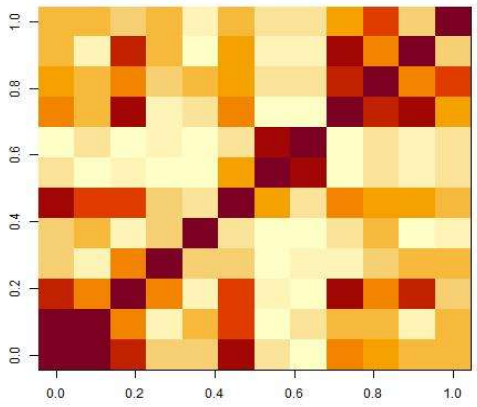
- Los pares con MAYOR varianza en valores absolutos son: {53,92}.
- Los pares con MENOR varianza en valores absolutos son: {59,76}
- Copiamos código R usado para conseguir la respuesta:

```
> #Calculamos la matriz de datos normalizada y la guardamos en la variable Xs
> Xs <- as.matrix(scale(var_matrix, center = TRUE, scale = TRUE))
> fix(Xs)
> Xs
>
> #Comprobamos que realmente el valor de la media de los datos de Xs es
> #prácticamente cero y el valor de la desviación típica prácticamente 1.
> mean(Xs)
[1] 4.598972e-19
> sd(Xs)
[1] 0.9924475
> round(mean(Xs), digits = 0)
[1] 0
> round(sd(Xs), digits = 0)
[1] 1
>
> #Ahora vamos a calcular la matriz de covarianza de los datos Xs y la vamos a
> #guardar en la variable CXs.
> CXs <- cov(Xs)
> fix(CXs)
> CXs
>
> #Visualizamos la matriz como una imagen y la guardamos en formato .jpeg
> image(CXs)
> jpeg('CXs.jpeg')
> image(CXs)
> dev.off()
```



	row.names	rent	inc_sal	inc_ret	inc_emp	inc_non	inc_oth	gini	dist8020	mean_age	perc_chil	per_ret	home_size
1	rent	1	0.9267303	0.7612249	-0.2836566	-0.3159901	0.9130952	0.1801722	-0.04145366	0.575407	-0.471699	0.4162707	-0.3950107
2	inc_sal	0.9267303	1	0.5298288	-0.1325474	-0.3995027	0.7277385	-0.03060094	-0.2353366	0.3706067	-0.3548565	0.1432096	-0.3646303
3	inc_ret	0.7612249	0.5298288	1	-0.5115812	-0.1609269	0.7283159	0.1627369	0.05805812	0.8461552	-0.5501134	0.833983	-0.2723374
4	inc_emp	-0.2836566	-0.1325474	-0.5115812	1	-0.2921928	-0.30488	0.01873346	0.09633633	-0.1687072	-0.2774858	-0.3558086	-0.3969711
5	inc_non	-0.3159901	-0.3995027	-0.1609269	-0.2921928	1	-0.2019849	0.02066379	0.02290216	-0.1966636	0.3776646	-0.00395581	0.1294757
6	inc_oth	0.9130952	0.7277385	0.7283159	-0.30488	-0.2019849	1	0.4573695	0.2185647	0.538953	-0.4420784	0.4569156	-0.3588892
7	gini	0.1801722	-0.03060094	0.1627369	0.01873346	0.02066379	0.4573695	1	0.8793883	0.07983447	-0.2432927	0.1451232	-0.2259754
8	dist8020	-0.04145366	-0.2353366	0.05805812	0.09633633	0.02290216	0.2185647	0.8793883	1	0.05947531	-0.2204091	0.1483152	-0.2113529
9	mean_age	0.575407	0.3706067	0.8461552	-0.1687072	-0.1966636	0.538953	0.07983447	0.05947531	1	-0.7753111	0.9151241	-0.482575
10	perc_chil	-0.471699	-0.3548565	-0.5501134	-0.2774858	0.3776646	-0.4420784	-0.2432927	-0.2204091	-0.7753111	1	-0.5509099	0.687172
11	per_ret	0.4162707	0.1432096	0.833983	-0.3558086	-0.00395581	0.4569156	0.1451232	0.1483152	0.9151241	-0.5509099	1	-0.3038743
12	home_size	-0.3950107	-0.3646303	-0.2723374	-0.3969711	0.1294757	-0.3588892	-0.2259754	-0.2113529	-0.482575	0.687172	-0.3038743	1

```
> #Buscamos el nombre (índice) de las variables que contienen el mayor y menor
> #valor absoluto de covarianza.

> #Primero mostramos la matriz CXs en valores absolutos
> CXs_ab <- abs(CXs)
> CXs_ab

> #Visualizamos la matriz de covarianzas definida positiva como una imagen y
> #la guardamos en formato .jpeg
> image(CXs_ab)
> jpeg('CXs_ab.jpeg')
> image(CXs_ab)
> dev.off()
```

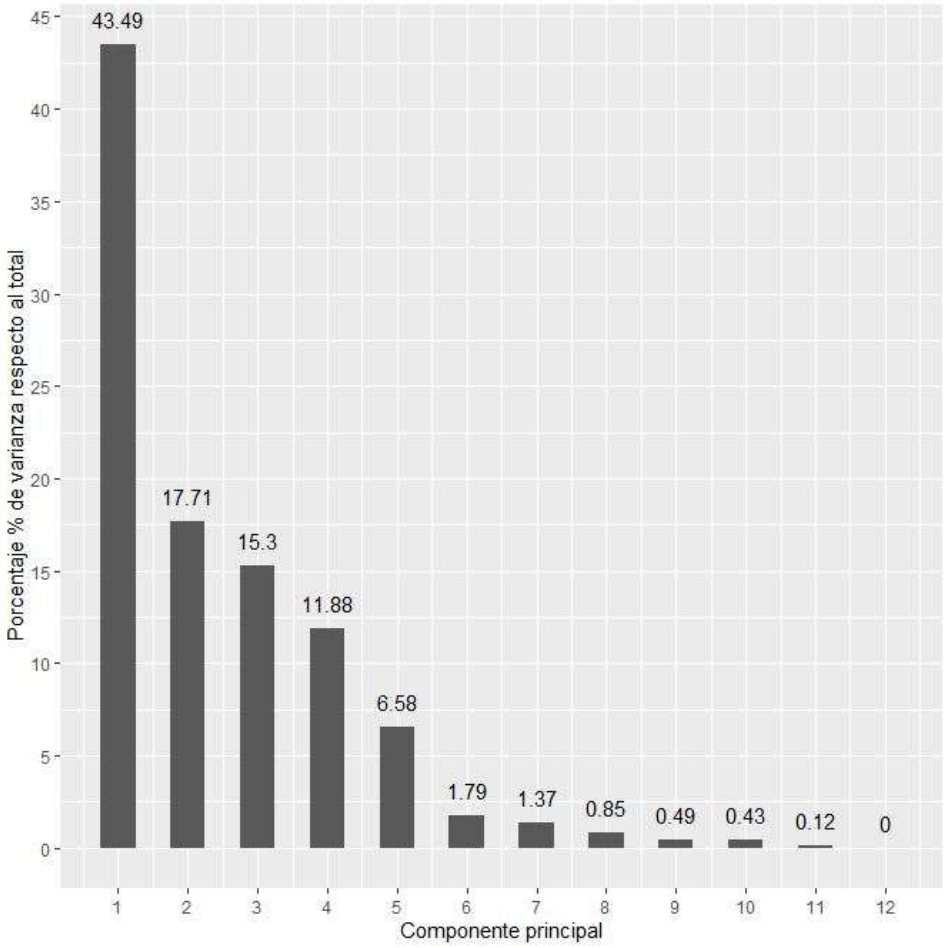



```
> #Mostramos el par de valores máximo y mínimo de los valores absolutos de la
> #matriz CXs_ab.

> par_max = tail(sort(CXs_ab), 4)
> par_min = head(sort(CXs_ab), 4)
> par_max
[1] 1 1 1 1
> par_min
[1] 0.00395581 0.00395581 0.01873346 0.01873346
>
> #Convertimos la matriz CXs_ab en una lista ordenada y localizamos el par de
> #valores mínimos y el par de valores máximos

> lista_ord = as.list(CXs_ab)
> pos_par_max = which(lista_ord == par_max)
> pos_par_min_1 = which(lista_ord == par_min[1])
> pos_par_min_2 = which(lista_ord == par_min[3])
> pos_par_max
[1] 53 92 105 118
> pos_par_min_1
[1] 59 125
> pos_par_min_2
[1] 43 76

> #Comprobamos visualmente en la lista que los datos obtenidos concuerdan

> lista_ord[53]
[[1]]
[1] 1
> lista_ord[92]
[[1]]
[1] 1
> lista_ord[59]
[[1]]
[1] 0.00395581
> lista_ord[76]
[[1]]
[1] 0.01873346
```

Pregunta 4. [5 %] Finalmente, calcular la descomposición en componentes principales de la matriz de covarianzas ***CXs***. Dibujar la distribución de la varianza explicada en porcentaje (eje de ordenadas) para cada componente principal (eje de abscisas) respecto la variancia total de los datos.

Respuesta:

```
>
> #Calculamos lo componentes principales de la matriz de datos normalizada Xs
> comp_prin <- prcomp(Xs, scale = TRUE, center = TRUE)
>

> #Comprobamos que la media es prácticamente cero y la varianza uno
> comp_prin$center

rent          inc_sal       inc_ret       inc_emp       inc_non
-2.096740e-16 1.458732e-16 -1.292227e-16 -3.150940e-16 -8.502989e-18
inc_oth       gini         dist8020      mean_age     perc_chil
9.577949e-17 -3.434184e-16 2.684556e-16 -4.436342e-16 4.555498e-16

per_ret       home_size
1.550302e-16 3.347732e-16

> comp_prin$scale

rent inc_sal inc_ret inc_emp inc_non inc_oth gini dist8020
 1       1       1       1       1       1      1     1
mean_age perc_chil per_ret home_size
    1        1        1         1

> comp_prin$sdev^2

[1] 5.218483e+00 2.125341e+00 1.836521e+00 1.425191e+00 7.895196e-01
[6] 2.145532e-01 1.647613e-01 1.015204e-01 5.855235e-02 5.127084e-02
[11] 1.428599e-02 2.020722e-07


> #Ahora dibujamos la distribución de la varianza explicada en porcentaje (eje
> #de ordenadas) para cada componente principal (eje de abscisas) respecto a la
> #varianza total de los datos.
>
> #Calculamos el % de cada valor respecto al total

> porc_varianza <- abs(comp_prin$sdev^2)*100/sum(abs(comp_prin$sdev^2))
> porc_varianza

[1] 4.348736e+01 1.771117e+01 1.530434e+01 1.187660e+01 6.579330e+00
[6] 1.787943e+00 1.373010e+00 8.460034e-01 4.879362e-01 4.272570e-01
[11] 1.190499e-01 1.683935e-06
>
> #Importamos paquete ggplot2 con el que vamos a generar la gráfica.

> library(ggplot2)
> library(crayon)
Attaching package: 'crayon'
The following object is masked from 'package:ggplot2':
%+%
> ggplot(data = data.frame(porc_varianza, pc = 1:12),
+ aes(x = pc, y = porc_varianza)) + geom_col(width = 0.5) +
+ geom_text(aes(label = round(porc_varianza, digits = 2)), vjust = -1,
+ colour = "black") + ylim(c(0, 100)) +
+ scale_x_continuous(breaks = seq(1, 12, by = 1)) +
+ scale_y_continuous(breaks = seq(0, 60, by = 5)) +
+ labs(x = "Componente principal",
+ y = "Porcentaje % de varianza respecto al total")
```



Pregunta 5. [20 %] Como habéis visto, la mayor parte de la varianza queda concentrada en unas pocas componentes principales. Por esto, podemos reducir la dimensión del subespacio, proyectar nuestros datos allí y utilizar estas representaciones para análisis posteriores. Un buen criterio para el diseño del nuevo subespacio es restringir el porcentaje total de varianza explicada por el subespacio a un cierto umbral. En esta práctica, os quedareis con las L primeras componentes principales que expliquen, al menos, un 75% de la varianza inicial. Calcular el valor mínimo de L, es decir, el mínimo número de componentes principales necesarias para explicar un 75% de la varianza de nuestros datos.

Respuesta:

- $L = 3$
- Copiamos código R usado para conseguir la respuesta:

```
> #Calculamos la variabilidad acumuladada de las variables originales y de las
> #componentes principales

> library(data.table)
> variabilidad_org = 100/12
> variabilidad_org

[1] 8.333333


> com_principales <- round(porc_varianza, digits = 2)
> variabilidad_cp = data.frame(variables = 1:12, com_principales,
+                              variabilidad_org )
> dt <- as.data.table(variabilidad_cp)
> dt[, acumulado_com.prin:= cumsum(porcentaje)]
> dt[,acumulado_original := cumsum(variabilidad_org)]
> dt
```

	variables	com_principales	variabilidad_org	acumulado_com.prin	acumulado_original
1:	1	43.49	8.333333	43.49	8.333333
2:	2	17.71	8.333333	61.20	16.666667
3:	3	15.30	8.333333	76.50	25.000000
4:	4	11.88	8.333333	88.38	33.333333
5:	5	6.58	8.333333	94.96	41.666667
6:	6	1.79	8.333333	96.75	50.000000
7:	7	1.37	8.333333	98.12	58.333333
8:	8	0.85	8.333333	98.97	66.666667
9:	9	0.49	8.333333	99.46	75.000000
10:	10	0.43	8.333333	99.89	83.333333
11:	11	0.12	8.333333	100.01	91.666667
12:	12	0.00	8.333333	100.01	100.000000

```
> #Podemos observar en la columna acumulado_com.prin que necesitamos como
> #mínimo de 3 componentes principales para explicar un 75% de la varianza
```

Pregunta 6. [10 %] Considerar la componente principal C e indicar que variables contribuyen en mayor y menor peso (en valor absoluto).

Respuesta:

- ***C*** = 2
- Variable que contribuye en mayor peso = 8
- Variable que contribuye en menor peso = 9
- Copiamos código R usado para conseguir la respuesta:

```
>#Obtenemos la carga de cada variable en la componente principal C = 2 y las

>#ordenamos en valores absolutos para detectar el valor máximo y el valor mínimo.

> loadings_cpC2 <- comp_prin$rotation[,2]
> loadings_cpC2_abs <- abs(loadings_cp2)
> loadings_cpC2_ordmax <- sort(loadings_cp2_abs, decreasing = TRUE)
> loadings_cpC2_ordmin <- sort(loadings_cp2_abs, decreasing = FALSE)
> loadings_cpC2_ordmax[0:1]
 dist8020
0.5693209
> loadings_cpC2_ordmin[0:1]
  mean_age
0.02020759

> #Comprobamos que sus posiciones son 8 y 9 respectivamente

> loadings_cpC2[8:9]
   dist8020    mean_age
 0.56932086 -0.02020759
```

Pregunta 7. [10 %] Calcular las nuevas variables proyectadas a las componentes principales. Para la componente principal C, anotar las secciones censales (relacionarlo con la variable id) con el valor máximo y mínimo.

Respuesta:

- $C = 2$
- Número de las dos regiones censales = { 1 , 3 }
- Copiamos código R usado para conseguir la respuesta:

```
# Cálculo de las nuevas variables proyectadas

> nuevas_vars <- predict(comp_prin, newdata = Xs)
> nuevas_vars

                PC1          PC2          PC3          PC4          PC5          PC6         PC7
1  -0.50794175  4.160920259 -1.358941167  1.377985127 -0.743106650  0.7389278499  0.26811024
2  -1.16093410  1.176711951  2.718741157  0.279414147  1.659629993 -0.8997296842 -0.23911466
3  -0.70326104 -0.008370836 -0.623620313 -0.120038497 -0.288280861 -0.4870146888  0.25049441
4   0.66376338  1.561242089 -0.522259943 -0.372580365 -0.171668850 -0.4806855247  0.15148039
5   7.53212491 -1.418681812  5.844886318  2.095023111 -2.229412686 -0.0219708550  0.33604441
6   2.85603905 -0.232678915  1.898477497 -1.460840712 -0.867420117 -0.0490803707 -0.47406645
7  -3.44837180 -0.793740884  0.701570102  0.853413972 -0.694867623  0.1437992747 -0.85390217
8  -1.65193465  1.106928360  0.634993608  0.976624710 -1.734932206  0.0579745415  0.25884832
9  -3.00719695 -0.435952235  1.181810561 -0.401342367 -0.388761024 -0.0369140157 -0.41558901
10 -2.92144558  0.721523613  0.864824177 -0.461269197 -0.086813947 -0.1827376379 -0.24715240
11 -2.40611561 -0.552045154  0.290695782 -0.636536564 -1.620055092 -0.0005384315 -0.33873119
12 -2.41155289  0.617334033  0.870641832 -0.361522916  0.828362048 -0.9548757813 -0.04624334
13 -1.83097026  0.275114641  0.882997097  1.244024296 -0.392906354  0.1181731956  0.20669120
14 -2.25081180 -0.272663553 -0.005282770  0.183468291 -0.297202653  0.4244824891 -0.13968576
15 -0.73101713  0.051583193 -1.122406227 -1.045611009 -0.194308554  0.4587193303  1.00883543
16  0.36727677 -1.062539284 -1.868893976 -0.210968717 -0.252516486 -0.6641057260  0.21596271
17  1.79068248  0.433502859 -1.720314740  0.888885825  0.001468144 -0.4836690379 -0.10196197
18 -3.73303788 -0.252856886  1.469450114 -1.879798360  0.130590874 -0.3663400721 -0.13079873
19 -2.77633499  1.904522730  1.155564594 -0.249036129  0.781751335 -0.5739087256 -0.76059482
20  1.36494487  0.855527711  0.177098411 -0.448515334 -0.548006405  0.5598166915 -0.05513769
21 -1.58318379 -1.891240583 -0.776358592  1.486868392  0.340187356  0.3207802734  0.02346615
22 -4.88036789 -3.110858674  1.350679044  1.022277179  1.858903435  0.1707335990  0.94569094
23 -0.67123817 -0.943954176 -0.371341678  1.055805592  1.209474840  1.0015686521 -0.59543610
24  0.22750479 -1.658635139 -2.032573575  0.486574678  0.266937814 -0.5525423189 -0.69510498
25  1.46202412 -1.918865766 -1.641496323  1.826394337 -0.224572745 -0.6704005039 -0.41786845
26 -2.96483977  0.140045063  1.462419682  2.996109375  0.308887635  0.9424639041  0.01826190
27 -0.47878183 -0.369162542  0.035019802 -0.245828459 -0.235084202 -0.4973974281  0.12658472
28  3.95185691 -0.634462876 -1.043137224  2.249050127 -0.977221690 -0.8414931022 -0.33917673
29 -2.02109901  2.663297448  0.164933992  0.698632233 -0.126895871 -0.2396369692 -0.16020433
30 -3.23961250  0.971675702 -0.627586193 -0.157155661 -0.730065366  0.3762966735 -0.20618860
31 -0.63223387  1.245865280  0.040672816  0.907578953 -0.974034315 -0.2098520340  0.58758352
32 -1.38682197 -1.903982656 -0.580892859  0.334793617 -0.404088325 -0.4196169455  0.49287760
33 -1.33446967 -2.279825003 -0.342008989  0.841062625 -0.378342989 -0.7239472662  0.30371264
34 -0.53126893 -0.051946395  1.722240959 -0.819890083 -0.208103451 -0.4316601392  0.31131555
35  1.19099293  0.935144859  0.344764697 -0.642497904  1.139335222 -0.4690341696  0.51418931
36  1.95395328  0.553139396 -1.094632848 -0.133476254  0.770276040  0.0519168927  0.44011952
37  3.02290277 -0.212658036  0.836872233 -0.482024344  1.566080140  0.1441154435 -0.10696977
38  2.38589042  0.100517509  0.046984408  0.107994163  0.365334792  0.5810676069  0.19084341
39  2.77496509  0.230245959 -0.160954095  0.007208606  1.173979284  0.2879278600 -0.30123734
40  2.53309427  0.717192172  0.223078520  0.008091281  1.754173502  0.4653467621 -0.11105555
41  2.35791401 -0.403886734  1.115558697 -1.324670774  0.816518686  0.3736704289 -0.07437692
42  3.24977327  0.520598118  0.682296342  0.693252505  1.883465964  0.0046686206  0.45269993
43  0.14844835  1.076913444  0.446207524 -1.408803103  0.988219616 -0.4609375752  0.08012771
44 -0.02298756  0.336625432 -0.005377663 -1.499240714  1.051078028  0.2284079683  0.10350660
45  2.01525020 -0.048104574 -0.922663438 -0.220875423  0.313043001 -0.1386923455  0.30697305
46  3.13031957  0.656955584 -1.000795419  0.718049973  0.843956642 -0.1627779224 -0.18113491
47  0.18877462 -1.001959781 -1.228200990  0.809786569 -0.911003694 -0.1046766557  0.30145481
48 -1.10413083 -0.278702650 -0.284343326 -0.465956112 -0.435019741  0.3498028396  0.50535751
49 -0.33350198  4.048669925 -2.542005313  1.369831763 -0.084208625  0.2073324328  0.07299176
50 -0.12202331  3.267095911  0.042918061  0.905553260 -0.610201895 -0.0791005335  0.08033796
51 -3.45157212 -2.189301067  1.133516676  0.837474387  0.411285729  0.6556821617  0.01576882
52  0.55853126  1.271048166 -0.519566610 -1.353987937 -0.567473432 -0.1989154026  0.21119984
53  1.84920961  0.216173754 -0.755716552 -0.773764234  0.066353202 -0.0002145597 -0.30442385
54 -0.15857020 -1.174274029 -0.599231247 -0.903968582  0.148464617 -0.0726752709 -0.94526472
55  1.68281255  1.304433614  1.666902707 -0.975079101 -0.152538106  0.7644755525 -0.61078482
56  0.70805968 -2.057852596 -2.707888045  0.861912207 -0.060862002  0.4055729254 -0.26630616
57  0.77465011 -2.079506926 -0.900795004 -1.272611006 -0.350789145  0.1373955709  0.43980675
58  2.18067352 -1.361775046 -0.945819452 -0.936830785 -0.957188613  0.4867769627 -0.67983350
59  1.40982317 -1.700365175 -1.370563144 -1.577320271  0.203376907  0.3333050024  0.14550188
60  0.95281684 -0.830478916 -0.075503493 -0.873745771 -0.337210687  0.4383045848  0.21545445
61 -0.82744297  0.010780128 -0.255646203 -4.407354614 -1.645970437  0.2456356030  0.21605150
            PC8          PC9          PC10          PC11          PC12
1   0.316571377  0.05233203  0.104479128 -0.0022552549  1.571576e-04
2  -0.638730802 -0.46522267 -0.341499216 -0.0282859459 -5.344653e-04
3   0.340913758  0.33230971 -0.303947486  0.0917640209 -5.879721e-05
```

```
4   0.124763511 -0.10919148 -0.124803516 -0.0006297962  1.160190e-04
5   0.467811617  0.19388971  0.305492603 -0.0679023641 -1.619465e-03
6  -0.159311208 -0.49051711 -0.022957126 -0.0111954757  9.974688e-04
7   0.426330906 -0.30768299 -0.367395819 -0.0152434712 -1.208613e-04
8  -0.732790750  0.05317348 -0.161830900 -0.0431480228  2.612461e-04
9   0.599849027  0.06478591  0.066408706 -0.0885876705  3.938604e-04
10 -0.007193477 -0.29454916 -0.140275984  0.0487716264 -3.762498e-04
11 -0.383544074 -0.05202353  0.235074024  0.2756136765 -1.042364e-04
12  0.226977125  0.02889947  0.053375773 -0.0972329028 -2.295225e-05
13 -0.124391576  0.20406990 -0.165445136 -0.0287829626  5.935276e-04
14  0.573317420  0.01869240  0.061112657 -0.0955890657  2.864816e-04
15 -0.145832896 -0.02930164  0.123485708 -0.0886265555 -2.661810e-04
16  0.123360660  0.15569162 -0.065560658 -0.0326834838 -2.448013e-04
17 -0.129769817  0.22481351  0.180824318  0.1169239880  8.939872e-05
18 -0.234935062  0.26040959  0.010414538  0.0893126198 -3.188739e-04
19 -0.041622044  0.30310279  0.513039339 -0.0554725426 -6.537855e-05
20 -0.348151954 -0.13773034  0.008618919 -0.0369510874  3.558797e-04
21 -0.163301829  0.24496521 -0.133332557  0.1135206747 -3.022725e-04
22 -0.028373983 -0.10876730  0.637521986 -0.0549815336  2.993143e-04
23 -0.416118088  0.35854228 -0.369413905  0.1079080676 -9.171161e-04
24 -0.245306597  0.06256866 -0.047995880 -0.0275953210 -8.214082e-04
25 -0.097647223  0.00658328 -0.151865830  0.0101604968  4.406103e-05
26  0.324017582 -0.17021649 -0.391617784  0.0901763095  2.447617e-04
27 -0.174697078  0.30509248 -0.099321921 -0.0002994829  2.382381e-04
28  0.157438606  0.17449732  0.178895249  0.2454278895  1.199507e-03
29 -0.075850619 -0.24630258  0.426611907  0.0616358203 -2.308706e-04
30  0.562472764 -0.04939042  0.480329814  0.0370018458  5.397045e-05
31 -0.372021328 -0.43285774 -0.096218635  0.1311230251 -6.367735e-05
32  0.181752963 -0.49968297 -0.341308879  0.0486458767 -1.212569e-04
33  0.036217901 -0.17762844  0.173138909 -0.0741739499  3.335915e-04
34 -0.574836570  0.30745647 -0.010432309 -0.1171582548  4.558822e-04
35  0.430776460  0.19380745 -0.346352529  0.0090685961  2.836349e-05
36  0.164877654 -0.02419910 -0.089934431  0.0755228205 -3.711807e-04
37  0.342449897 -0.06021424 -0.159185560  0.1020413953  4.097408e-04
38 -0.124207794  0.16502463  0.010492709  0.1015444173  3.003465e-04
39  0.475395110  0.36824858  0.073533795  0.0721964234  4.493188e-04
40 -0.302672474 -0.11255004  0.183290780  0.0066737181 -1.637943e-04
41 -0.021404937 -0.21974411  0.029003795  0.1663277633  3.769272e-04
42  0.124340983 -0.16656441 -0.111461373  0.1409567783  3.050650e-04
43  0.065424393  0.16237847  0.111847932  0.1448615167 -1.458806e-04
44  0.067541300  0.44042193  0.112601532 -0.0342018313 -2.341120e-04
45  0.138794793 -0.19562218 -0.197529620 -0.2307154062  1.459368e-04
46 -0.257581890 -0.31531431  0.190081988 -0.0331638057 -1.076899e-04
47 -0.004119639  0.14350069  0.012658523  0.0346826603  3.748274e-04
48 -0.619072503  0.38361171  0.219717713  0.1146717264 -1.726807e-04
49  0.204568208 -0.34058054  0.194869307 -0.0259086731 -8.341841e-04
50  0.011636245  0.39661053 -0.328293086 -0.3116924321  2.318727e-04
51  0.214918791 -0.02215440 -0.060346315  0.0111730651  2.132221e-04
52  0.109563729  0.14487230 -0.254061010 -0.0653914073 -2.575271e-04
53 -0.081735680 -0.39052689  0.358801766 -0.0928449174 -1.260925e-04
54  0.385525805  0.10202168 -0.003967989 -0.2203050005 -2.646952e-04
55 -0.489334269  0.12083869 -0.141083522 -0.1346276346  5.339248e-04
56 -0.421229976  0.03897256 -0.003503670 -0.1200867684 -5.603405e-04
57  0.009493622 -0.02444772  0.016762984 -0.2547576679  2.387585e-04
58 -0.184185898 -0.07878713  0.144655242  0.0271711917  3.216706e-04
59  0.242619095 -0.32654287  0.082752906 -0.0681963827 -3.954628e-04
60 -0.212072882 -0.11569446 -0.021358650 -0.2069686398  2.768668e-04
61  0.362323616 -0.04817782 -0.247593254  0.2907777006 -5.047028e-04
```

#También podríamos calcularlas así

```
comp_prin$x
```

#Calculamos valores máximos y mínimos.

```
> max <- max(abs(nuevas_vars[,2]))
> min <- min(abs(nuevas_vars[,2]))

> max
```

```
[1] 4.16092
```

```
> min
```

```
[1] 0.008370836
```

```
> ord <- sort(abs(nuevas_vars[,2]), decreasing = FALSE)
> ord
```

```
         3          61          45          15          34          38          26
0.008370836 0.010780128 0.048104574 0.051583193 0.051946395 0.100517509 0.140045063
        37          53          39           6          18          14          13
0.212658036 0.216173754 0.230245959 0.232678915 0.252856886 0.272663553 0.275114641
        48          44          27          41          17           9          42
0.278702650 0.336625432 0.369162542 0.403886734 0.433502859 0.435952235 0.520598118
        11          36          12          28          46          40          10
0.552045154 0.553139396 0.617334033 0.634462876 0.656955584 0.717192172 0.721523613
         7          60          20          35          23          30          47
0.793740884 0.830478916 0.855527711 0.935144859 0.943954176 0.971675702 1.001959781
        16          43           8          54           2          31          52
1.062539284 1.076913444 1.106928360 1.174274029 1.176711951 1.245865280 1.271048166
        55          58           5           4          24          59          21
1.304433614 1.361775046 1.418681812 1.561242089 1.658635139 1.700365175 1.891240583
        32          19          25          56          57          51          33
1.903982656 1.904522730 1.918865766 2.057852596 2.079506926 2.189301067 2.279825003
        29          22          50          49           1
2.663297448 3.110858674 3.267095911 4.048669925 4.160920259
```

```
> #comprobamos que el id 3 coincide con min y que el id 1 coincide con max
```

```
> ord[1]
          3
0.008370836
> ord[61]
       1
4.16092
```

Pregunta 8. [20 %] Cuando reducimos la dimensión del subespacio generado por los datos iniciales a L, se produce una pérdida de información. Una manera de medir el error cometido en esta aproximación es calculando el error residual, tal y como se indica en la Sección 2.5.1 de los apuntes del módulo. Considerando el valor de L calculado en el apartado 5, calcular la desviación típica del error residual cuando se consideran solo las L primeras componentes principales.

Respuesta:

- $L = 3$
- Desviación típica = 2.331166
- Copiamos código R usado para conseguir la respuesta:

```
> #Tomamos solo las tres primeras componentes principales
```

```
> L <- comp_prin$x[, 1:3]
> L
```

```
             PC1          PC2          PC3
1  -0.50794175  4.160920259 -1.358941167
2  -1.16093410  1.176711951  2.718741157
3  -0.70326104 -0.008370836 -0.623620313
4   0.66376338  1.561242089 -0.522259943
5   7.53212491 -1.418681812  5.844886318
6   2.85603905 -0.232678915  1.898477497
7  -3.44837180 -0.793740884  0.701570102
8  -1.65193465  1.106928360  0.634993608
9  -3.00719695 -0.435952235  1.181810561
10 -2.92144558  0.721523613  0.864824177
11 -2.40611561 -0.552045154  0.290695782
12 -2.41155289  0.617334033  0.870641832
13 -1.83097026  0.275114641  0.882997097
14 -2.25081180 -0.272663553 -0.005282770
15 -0.73101713  0.051583193 -1.122406227
16  0.36727677 -1.062539284 -1.868893976
17  1.79068248  0.433502859 -1.720314740
18 -3.73303788 -0.252856886  1.469450114
19 -2.77633499  1.904522730  1.155564594
20  1.36494487  0.855527711  0.177098411
21 -1.58318379 -1.891240583 -0.776358592
22 -4.88036789 -3.110858674  1.350679044
23 -0.67123817 -0.943954176 -0.371341678
```

```
24  0.22750479 -1.658635139 -2.032573575
25  1.46202412 -1.918865766 -1.641496323
26 -2.96483977  0.140045063  1.462419682
27 -0.47878183 -0.369162542  0.035019802
28  3.95185691 -0.634462876 -1.043137224
29 -2.02109901  2.663297448  0.164933992
30 -3.23961250  0.971675702 -0.627586193
31 -0.63223387  1.245865280  0.040672816
32 -1.38682197 -1.903982656 -0.580892859
33 -1.33446967 -2.279825003 -0.342008989
34 -0.53126893 -0.051946395  1.722240959
35  1.19099293  0.935144859  0.344764697
36  1.95395328  0.553139396 -1.094632848
37  3.02290277 -0.212658036  0.836872233
38  2.38589042  0.100517509  0.046984408
39  2.77496509  0.230245959 -0.160954095
40  2.53309427  0.717192172  0.223078520
41  2.35791401 -0.403886734  1.115558697
42  3.24977327  0.520598118  0.682296342
43  0.14844835  1.076913444  0.446207524
44 -0.02298756  0.336625432 -0.005377663
45  2.01525020 -0.048104574 -0.922663438
46  3.13031957  0.656955584 -1.000795419
47  0.18877462 -1.001959781 -1.228200990
48 -1.10413083 -0.278702650 -0.284343326
49 -0.33350198  4.048669925 -2.542005313
50 -0.12202331  3.267095911  0.042918061
51 -3.45157212 -2.189301067  1.133516676
52  0.55853126  1.271048166 -0.519566610
53  1.84920961  0.216173754 -0.755716552
54 -0.15857020 -1.174274029 -0.599231247
55  1.68281255  1.304433614  1.666902707
56  0.70805968 -2.057852596 -2.707888045
57  0.77465011 -2.079506926 -0.900795004
58  2.18067352 -1.361775046 -0.945819452
59  1.40982317 -1.700365175 -1.370563144
60  0.95281684 -0.830478916 -0.075503493
61 -0.82744297  0.010780128 -0.255646203
```

> #Calculamos el error residual

```
> Error_res <- Xs[, 1:3] - L
> Error_res <- as.matrix(Error_res)
> Error_res
```

```
          rent    inc_sal     inc_ret
1   0.62009272 -4.1648791  0.42780145
2   1.07620107 -1.7321341 -2.07332890
3   0.91863884  0.4496860  0.61571292
4  -1.31294948 -2.1272000 -0.18989402
5  -9.81160463 -1.2452764 -7.62068540
6  -4.39316413 -1.5814880 -2.73867629
7   5.17230458  2.0335549  0.28141489
8   2.64135677  0.0153289 -0.37662595
9   3.98312742  0.7025958  0.30680601
10  3.69939390 -0.4457306  0.16181247
11  3.47962523  1.4671918  0.81742393
12  2.83165866 -0.6531774  0.36625024
13  2.86086734  0.6602717 -0.25504550
14  3.21089742  0.8154546  0.86895326
15  0.80551697  0.1734719  1.42006037
16 -0.28650174  1.7228871  1.55760764
17 -2.13222754 -0.1737907  0.73315541
```

```
18  4.49153312  0.4817935  0.63974737
19  3.24868170 -2.1682710  0.44509120
20 -2.03295654 -1.5018900 -0.86669902
21  3.06426682  3.7299288  1.40067256
22  7.27872639  4.8943728  1.98585724
23  1.47711107  2.0102056  0.51184989
24  0.28500311  2.8133314  1.78749224
25 -0.97571512  3.2163492  0.74164028
26  4.88063639  1.2966211 -0.74134457
27  0.65070396  0.7475405  0.27791241
28 -4.50409137  0.8500912 -0.79886686
29  2.56592413 -2.5982194  0.39171992
30  4.37791343 -0.4335980  1.82155415
31  1.06457626 -0.6989154 -0.34250129
32  2.35773229  3.0866818  0.85744870
33  2.50524470  3.6347005  1.18458119
34  0.39460873 -0.1652228 -0.91822772
35 -2.30893552 -2.0509025 -0.89468003
36 -2.87783571 -1.1656764  0.04854379
37 -4.58481484 -1.4484632 -1.78911025
38 -3.26741533 -0.7754372 -0.92064946
39 -4.02437284 -1.4469250 -0.77746090
40 -3.75912696 -1.9585470 -0.98033918
41 -3.75431802 -1.2026150 -1.60945442
42 -4.68241649 -1.8786780 -1.89862682
43 -1.03044390 -2.1854624 -0.19729772
44 -0.61255002 -1.1141301  0.45071987
45 -2.87073308 -0.4840280 -0.14888908
46 -4.19900193 -1.3643143 -0.43233824
47  0.36091377  2.0460307  0.95838372
48  1.61491305  1.0103054  1.04324987
49  0.39858904 -3.9214864  1.38533209
50 -0.08225203 -3.4382405 -0.77689784
51  5.29692977  3.5829925  0.66793670
52 -1.36208697 -1.9526220 -0.07181779
53 -2.75269768 -1.1092944  0.09521704
54  0.14302111  1.0591353  0.81758489
55 -2.83558242 -2.7215691 -2.21900063
56 -0.01357135  3.5971037  2.16888562
57 -1.01234082  2.1515164  0.91398591
58 -2.78012877  1.0658650  0.14708973
59 -2.07422661  1.2120391  0.96397071
60 -1.35570185  0.5403913 -0.13683911
61 -0.06694607 -1.1592540  0.51183128
```

```
> #Calculamos la desviación típica del error residual.

> sd(Error_res)

[1] 2.331166
```