Tarea-Examen 2: Aprendizaje No Supervisado

Curso Avanzado de Estadística. Profa. Guillermina Eslava Gómez.

Aldo Sayeg Pasos Trejo.

Posgrado en Ciencias Matemáticas. Universidad Nacional Autónoma de México.

11 de marzo de 2020

1. Ejercicio 8

1.1. Incisos a) y b)

El ejercicio, en ambos incisos, nos piden aplicar un análisis de componentes principales y calcular la proporción de la varianza explicada (PVE, por sus siglas en inglés), primero utilizando algún paquete de Software y después utilizando la siguiente fórmula para el PVE de la m-ésima componente

$$\text{PVE}_m = \frac{\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{jm}x_{ij}\right)^2}{\sum_{j=1}^{p}\sum_{i=1}^{n}x_{ij}^2} \tag{1}$$

Dónde x_{ij} es la variable j de la observación i y ϕ_{jm} es el coeficiente de X_j para la componente principal m.

Invitamos al lector a consultar el anexo 2 del presente trabajo para encontrar el código con el que se realizaron dichos cálculos. Nosotros presentamos solo los siguientes resultados. Los datos tienen la siguiente forma

	Murder	Assault	UrbanPop	Rape
Alabama	13.2	236	58	21.2
Alaska	10.0	263	48	44.5
Arizona	8.1	294	80	31.0
Arkansas	8.8	190	50	19.5
California	9.0	276	91	40.6

Tabla 1: Primeras observaciones de los datos

Al estandarizar los datos para que tengan $\sigma^2 = 1$ y $\mu = 0$ y realizar el análisis de componentes principales, obtenemos los siguientes resultados para los coeficientes

	Murder	Assault	UrbanPop	Rape
PC0	0.5359	0.5832	0.2782	0.5434
PC1	0.4182	0.1880	-0.8728	-0.1673
PC2	-0.3412	-0.2681	-0.3780	0.8178
PC3	0.6492	-0.7434	0.1339	0.0890

Tabla 2: Coeficientes de la transformación a componentes principales

Las tablas de la PVE se presentan a continuación

	PC0	PC1	PC2	PC3
PVE	0.6201	0.2474	0.0891	0.0434

Tabla 3: PVE calculado por el la librería `sklean` en Python 3

	PC0	PC1	PC2	PC3
PVE	0.6201	0.2474	0.0891	0.0434

Tabla 4: PVE calculado manualmente con la ecuación 1

2. Ejercicio 9

Trabajando con los mismos datos del inciso anterior, lo primero que debemos hacer es realizar clustering jerárquico, utilizando la métrica euclideana para medir la distancia entre puntos y distintos tipos de enlaces para medir la distancia entre clusters. Los dendrogramas resultantes del clustering pueden consultarse en las figuras 2, 3, 4, 5, 6, 7, 8, 9 y 10 del Anexo 2 a este trabajo.

Para analizar la diferencia entre los nueve clusterings realizados y poder observarlos de manera más sencilla que en los dendogramas, la tabla 6 del Anexo 1 muestra las etiquetas obtenidas entre los distintos clusters. En la tabla podemos ver claramente, que, para un tipo de enlace dado, los resultados entre los datos estandarizados y con desviación estándar 1 son los mismos, mientras que para los datos originales se obtienen distintos clusters.

Para responder a la pregunta de si los datos deben de estandarizarse antes de calcular las distancias entre observaciones, la respuesta es que si se debe de hacer, debido a que las variables de los datos pueden estar en distintas escalas. Es decir, es posible que una variable esté en un intervalo más grande que la otra por lo que, al calcular la distancia entre dos observaciones, dicha variable tenga mucho más influencia sobre el valor final de la distancia.

Para los datos que presentamos, por ejemplo, es claro que la escala de la variable “Assault” es mucho menor que la de la variable “Murder”, lo que implica que puntos con un valor parecido de “Murder” pero muy distinto de “Assault” pueden considerarse muy similares. Este efecto, en principio, no es deseado, pues generalmente usamos como hipótesis que todas las variables deben de ser significativas.

En general, recomendamos estandarizar los datos antes de realizar un proceso de clustering.

3. Ejercicio 10

3.1. Inciso a)

Para este ejercicio, debemos de realizar generar 60 observaciones en 50 variables, que se puedan diferenciar en tres grupos de 20 observaciones. Generamos las variables $X_i = (x_{i1}, \ldots, x_{i50})$ tales que $x_{ij} \sim U(0,1)$ y realizamos les añadimos una media para diferenciar a los tres grupos de la siguiente manera:

$$Z_i = \begin{cases} X_i + \hat{\boldsymbol{e}}_{\mathbf{1}} & \text{Si } 1 \leq i \leq 20 \\ X_i + \hat{\boldsymbol{e}}_{\mathbf{2}} & \text{Si } 21 \leq i \leq 41 \\ X_i + \hat{\boldsymbol{e}}_{\mathbf{3}} & \text{Si } 41 \leq i \leq 60 \end{cases} \tag{2}$$

Con $\hat{\boldsymbol{e}}_{\boldsymbol{i}} \in \mathbb{R}^{50}$ el i-ésimo vector de la base canónica de $\mathbb{R}^{50}$.

3.2. Inciso b)

Realizamos ahora PCA sobre las observaciones y graficamos las dos componentes principales, como muestra la figura 1

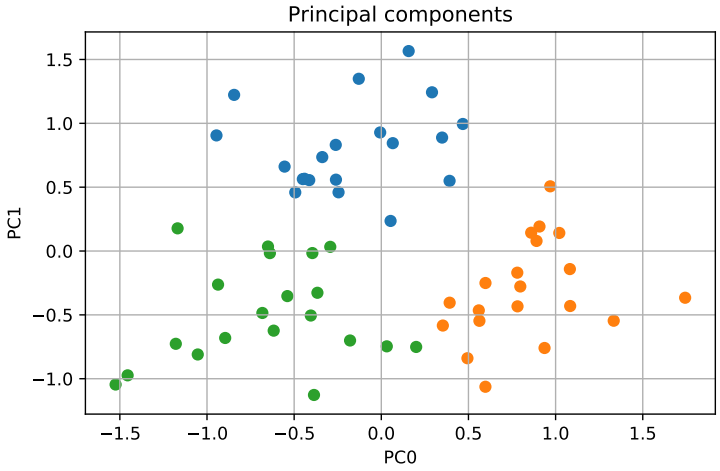

Figura 1: Primeras dos componentes principales para las 60 observaciones generadas. Cada color representa que son de un grupo distinto

En la figura podemos observar que las observaciones se separan en los tres grupos de manera muy clara en el espacio de las componentes principales.

3.3. Incisos c), d), e) y g)

Pretendemos ahora usar el algoritmo de K-means para realizar clustering sobre las observaciones originales, estandarizadas y con desviación estándar 1 para $K = 3, 2, 4$. Presentamos los resultados de dicho clustering graficando las tres primeras componentes de los vectores Z_i y marcando clusters distintos con colores distintos.

La tabla 8 muestra que con los datos originales la separación en grupos originales siempre se logra, a excepción de uno o dos puntos, para $K = 3$. Para $K = 2$, dos de estos grupos se fusionan, mientras que para $K = 4$ un grupo se separa en dos y dos de los originales se mantienen relativamente bien.

Un comportamiento parecido se obtiene cuando hacemos que los datos tengan $\sigma^2 = 1$, aunque con menor calidad en los resultados. Sin embargo, si estandarizamos los datos, el algoritmo deja de distinguir los grupos. La explicación a esto es clara, ya que por la construcción de los datos, lo que diferencía a cada grupo es una constante que se suma, que se quita de todos cuando le quitamos el valor promedio, por lo que los datos vuelven a su distribución aleatoria y no existe una distinción clara entre ellos.

3.4. Inciso f)

Las gráficas 21, 20 y 22 muestran los resultados de aplicar el clustering sobre las dos primeras componentes principales para $K = 3, 4, 2$, mientrasl que la tabla 10. Para $K = 3$ la separación se logra de manera normal y óptima, mientras que para $K = 2$ dos grupos se fusionan en uno y para $K = 4$ los grupos uno y dos se dividen en tres grupos.

4. Ejercicio 11

Dato un conjunto de 40 tejidos que son representados por 1000 genes, sabemos a priori que los 20 primeros tejidos son muestras sanas mientras que los últimos 20 genes son de tejido enfermo.

4.1. Inciso b)

Queremos aplicar clustering jerárquico a los datos usando distancia basada en correlación para separarlos en dos grupos. El dendrograma resultante para las 9 combinaciones de los datos originales, estandarizados y con $\sigma^2 = 1$ y para los tres tipos de enlace entre clusters se muestran en las figuras 23, 24, 25, 26, 27, 28, 29, 30 y 31.

Para la tabla 12 muestra el etiquetado obtenido a partir de estos algoritmos. Como podemos ver, los únicos datos en los cuales el clutering jerárquico si logra dividirlos en los dos grupos conocidos es para los datos estandarizados. Para los tipos de datos, el resultado varia según el tipo de enlace pero en general podemos afirmar que no se separan en los grupos buscados o en un caso (enlace promedio y datos con $\sigma^2 = 1$) no hay dos grupos pues un tejido termina siendo su propio cluster.

Notemos que, para los datos estandarizados, la separación en los dos grupos conocidos se cumple para cualquier enlace entre clusters, es decir, es independiente del enlace. Podemos concluir que, para este conjunto de datos, la estandarización si resulta necesaria para obtener resultados satisfactorios.

4.2. Inciso c)

Queremos ahora determinar cual es el conjunto de genes que difieren más entre ambos grupos. Claramente no hay una manera axiomática de resolver este problema, por lo presentamos tres métodos utilizados para abordar dicho poblema.

4.2.1. Método 1: maximizar la norma de la matriz de diferencias

Para cada gen k, definimos la matriz de diferencias $A^{(k)}$ de la siguiente forma

$$A_{ij}^{(k)} = x_{ik} - x_{jk} \tag{3}$$

Es decir, $A_{ij}^{(k)}$ es la diferencia del valor del gen k entre los tejidos i, j. Recordando que la norma de Frobenius de una matriz D de $n \times m$ es

$$|D| = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} |D_{ij}|^2} \tag{4}$$

Es claro que la norma de Frobenius de la matriz $A^{(k)}$ cuantifica cuanto varían las observaciones entre ambos grupos. Así, podemos decir entonces que los genes con mayor variación son aquellos cuya matriz de variación es máxima. La tabla 14 en el Anexo 1 presenta cuales son los 20 genes que maximizan dicho valor.

4.2.2. Método 2: quitar un subconjunto de genes y ver si el clustering se logra

En principio, el conjunto de variables que varía más es el conjunto de genes tales que al removerlos de los datos de los tejidos e intentar hacer clustering jerárquico sobre los genes restantes, este no divide a los tejidos en los dos grupos conocidos.

Sabemos que hay $\frac{1000!}{n!(1000-n)!}$ subconjuntos de n genes, por lo que es muy complicado ir probando con todos los subconjuntos de distintos tamaños para utilizar este método. Sin embargo, una manera más simple sería empezar simplemente remover un gen del conjunto de datos y ver si el clustering se logra.

Se implementó dicho procedimiento en el código y lo que se obtuvo, en principio, es que solo removiendo un gen de los datos, de igual manera se logra hacer el clustering jerárquico para los tres tipos de enlace, por lo que este método no nos resulta satisfactorio quitando solo un gen.

Por el tiempo de cómputo, no se intentó realizar la prueba quitando subconjuntos de más de un gen.

4.2.3. Método 3: intentar hacer clustering con solo un subconjunto de genes

Pensando el problema de manera inversa al método anterior, podríamos pensar que si logramos realizar el clustering en los dos grupos conocidos utilizando un subconjunto mínimo de genes, esos son los que permiten distinguirlos en ambos grupos y los que varían más. Nuevamente, por el número de subconjuntos, esto no es práctico.

Nos limitamos a analizar el clustering utilizando solamente un gen y ver si era posible diferenciarlos en los grupos buscados. No resultó posible hacer dicha separación con un solo gen de manera perfecta. Sin embargo, podemos cuantificar, con la ecuación 5, la razón de errores de cada clasificación con un solo gen:

$$\text{error} = \frac{N_{\text{aciertos}}}{N_{tejidos}} \tag{5}$$

La tabla 16 en el Anexo 1 presenta cuales son los 20 genes que maximizan dicho valor para distintos tipos de enlace

4.2.4. Conclusiones

Al comprar los genes obtenidos por los métodos 1 y 3, es claro que el conjunto de genes contiene varios en común. En particular, el gen 501 aparece como el más variante en ambos métodos.

Debido a esta coincidencia en resultados, consideramos que los resultados arrojados por ambos métodos son satisfactorios.

Además, notemos que el haber tomado 20 genes fue arbitario. En general, podríamos usarlos para ordenar los genes según su variación.

Anexo 1: Tablas relevantes

Resultados del clustering para el ejercicio 9

	Cluster complete original	Cluster complete standardized	Cluster complete with-mean	Cluster single original	Cluster single standardized	Cluster single with-mean	Cluster average original	Cluster average standardized	Cluster average with-mean
Alabama	0	0	0	0	0	0	0	0	0
Alaska	0	0	0	0	1	1	0	2	2
Arizona	0	1	1	0	0	0	0	0	0
Arkansa	2	2	2	0	0	0	2	1	1
Califor	0	1	1	0	0	0	0	0	0
Colorad	2	1	1	0	0	0	2	0	0
Connect	1	2	2	0	0	0	1	1	1
Delawar	0	2	2	0	0	0	0	1	1
Florida	0	1	1	2	2	2	0	0	0
Georgia	2	0	0	0	0	0	2	0	0
Hawaii	1	2	2	0	0	0	1	1	1
Idaho	1	2	2	0	0	0	1	1	1
Illinoi	0	1	1	0	0	0	0	0	0
Indiana	1	2	2	0	0	0	1	1	1
Iowa	1	2	2	0	0	0	1	1	1
Kansas	1	2	2	0	0	0	1	1	1
Kentuck	1	2	2	0	0	0	1	1	1
Louisia	0	0	0	0	0	0	0	0	0
Maine	1	2	2	0	0	0	1	1	1
Marylan	0	1	1	0	0	0	0	0	0
Massach	2	2	2	0	0	0	2	1	1
Michiga	0	1	1	0	0	0	0	0	0
Minneso	1	2	2	0	0	0	1	1	1
Mississ	0	0	0	0	0	0	0	0	0
Missour	2	2	2	0	0	0	2	0	0
Montana	1	2	2	0	0	0	1	1	1
Nebrask	1	2	2	0	0	0	1	1	1
Nevada	0	1	1	0	0	0	0	0	0
New Ham	1	2	2	0	0	0	1	1	1
New Jer	2	2	2	0	0	0	2	1	1
New Mex	0	1	1	0	0	0	0	0	0
New Yor	0	1	1	0	0	0	0	0	0
North C	0	0	0	1	0	0	0	0	0
North D	1	2	2	0	0	0	1	1	1
Ohio	1	2	2	0	0	0	1	1	1
Oklahom	2	2	2	0	0	0	2	1	1
Oregon	2	2	2	0	0	0	2	1	1

Continued on next page

	Cluster complete original	Cluster complete standarized	Cluster complete with-mean	Cluster single original	Cluster single standarized	Cluster single with-mean	Cluster average original	Cluster average standarized	Cluster average with-mean
Pennsyl	1	2	2	0	0	0	1	1	1
Rhode I	2	2	2	0	0	0	2	1	1
South C	0	0	0	0	0	0	0	0	0
South D	1	2	2	0	0	0	1	1	1
Tenness	2	0	0	0	0	0	2	0	0
Texas	2	1	1	0	0	0	2	0	0
Utah	1	2	2	0	0	0	1	1	1
Vermont	1	2	2	0	0	0	1	1	1
Virgini	2	2	2	0	0	0	2	1	1
Washing	2	2	2	0	0	0	2	1	1
West Vi	1	2	2	0	0	0	1	1	1
Wiscons	1	2	2	0	0	0	1	1	1
Wyoming	2	2	2	0	0	0	2	1	1

Tabla 6: Resultados de clustering con distintas escalas en los datos y distintos tipos de enlace

Resultados del clustering para el ejercicio 10

	K: 3 , data: original	K: 2 , data: original	K: 4 , data: original	K: 3 , data: standarized	K: 2 , data: standarized	K: 4 , data: standarized	K: 3 , data: with-mean	K: 2 , data: with-mean	K: 4 , data: with-mean
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	2	0	0
2	0	0	0	1	0	0	0	1	0
3	0	0	0	1	0	0	1	1	1
4	0	0	0	0	0	0	1	0	1
5	0	0	0	2	1	3	0	1	2
6	0	0	2	0	1	3	2	1	2
7	0	0	2	2	1	3	0	1	2
8	0	0	0	0	0	0	1	0	1
9	0	0	3	0	0	2	2	0	2
10	0	0	0	1	0	1	1	0	3
11	0	0	0	1	0	1	1	0	3
12	0	0	0	0	0	2	1	0	1
13	0	0	0	1	1	0	0	0	0
14	0	0	2	2	0	3	1	1	2
15	0	0	0	0	1	1	2	0	3
16	0	0	0	1	0	3	1	1	2
17	0	0	0	1	0	2	0	1	3

Continued on next page

	K: 3 , data: original	K: 2 , data: original	K: 4 , data: original	K: 3 , data: standarized	K: 2 , data: standarized	K: 4 , data: standarized	K: 3 , data: with-mean	K: 2 , data: with-mean	K: 4 , data: with-mean
18	0	0	0	0	0	2	2	0	0
19	0	0	0	1	0	0	1	0	1
20	2	1	1	0	0	0	1	0	1
21	2	1	1	1	0	1	1	0	3
22	2	1	1	2	1	3	2	1	2
23	2	1	1	0	0	0	2	0	1
24	2	1	1	2	0	0	1	0	1
25	2	1	1	0	0	2	1	0	1
26	2	1	1	0	0	2	1	0	3
27	2	1	1	0	0	2	1	0	1
28	2	1	1	1	1	1	0	1	3
29	2	1	1	1	0	1	1	0	3
30	2	1	1	1	0	1	1	0	3
31	2	1	1	0	0	0	2	0	1
32	2	1	1	1	0	1	1	0	3
33	2	1	1	2	0	3	2	1	2
34	2	1	2	2	1	3	2	1	2
35	2	1	1	2	1	3	2	1	2
36	2	1	1	2	1	3	2	1	2
37	2	1	1	0	0	0	1	0	1
38	2	1	1	1	0	0	1	0	1
39	2	1	1	1	1	3	0	0	0
40	1	0	2	2	0	3	2	1	2
41	2	1	3	0	0	0	1	0	1
42	1	0	3	0	0	3	2	0	2
43	1	0	2	2	1	3	0	1	2
44	1	0	3	1	0	3	2	1	2
45	1	0	3	2	1	3	2	1	2
46	1	0	3	1	1	3	0	1	2
47	1	0	2	2	1	3	2	1	2
48	1	0	3	2	0	3	2	1	2
49	2	1	3	0	0	0	1	0	1
50	1	0	3	2	0	3	0	1	2
51	1	0	3	0	0	1	2	0	3
52	1	0	3	2	1	3	2	1	2
53	1	0	3	1	0	1	2	0	3
54	1	1	3	2	1	3	2	1	2
55	1	0	3	0	0	0	2	0	1
56	1	0	3	2	1	3	0	1	2
57	1	0	2	2	1	3	0	1	2
58	1	0	3	0	1	3	2	0	2
59	1	0	3	0	0	2	2	0	2

Tabla 8: Resultados de clustering con K-means para datos originales, estandarizados y con $\sigma^2 = 1$ para distintos K

	data: pca 2 vars, K=3	data: pca 2 vars, K=2	data: pca 2 vars, K=4
0	0	0	0
1	0	0	3
2	0	0	0
3	0	0	0
4	0	0	3
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	3
9	0	0	0
10	0	0	3
11	0	0	3
12	0	0	3
13	0	0	0
14	0	0	0
15	0	0	0
16	0	0	0
17	0	0	3
18	0	0	0
19	0	0	3
20	1	1	1
21	1	1	1
22	1	1	1
23	1	1	1
24	1	1	1
25	1	1	1
26	1	1	1
27	1	1	1
28	1	1	1
29	1	1	1
30	1	1	1
31	1	1	1
32	1	1	1
33	1	1	1
34	1	1	1
35	1	1	1
36	1	1	1
37	1	1	1
38	1	1	1

Continued on next page

	data: pca 2 vars, K=3	data: pca 2 vars, K=2	data: pca 2 vars, K=4
39	1	1	1
40	2	0	0
41	1	1	1
42	2	0	2
43	2	0	2
44	2	0	0
45	2	0	2
46	2	0	0
47	2	0	2
48	2	0	2
49	2	1	2
50	2	0	2
51	2	0	2
52	2	0	2
53	2	0	0
54	2	1	2
55	2	1	2
56	2	0	2
57	2	0	2
58	2	0	2
59	2	0	0

Tabla 10: Resultados de clustering con K-means para las dos primeras componentes principales de los datos, estandarizados y con $\sigma^2 = 1$ para distintos K

Resultados del clustering para el ejercicio 11

	Cluster complete original	Cluster complete standardized	Cluster complete with-mean	Cluster single original	Cluster single standardized	Cluster single with-mean	Cluster average original	Cluster average standardized	Cluster average with-mean
0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	1	0	0	1
2	0	0	1	0	0	1	0	0	1
3	0	0	1	0	0	1	0	0	1
4	1	0	0	0	0	1	1	0	1
5	0	0	0	0	0	1	1	0	1
6	1	0	1	0	0	1	0	0	1
7	0	0	0	0	0	1	1	0	1
8	1	0	1	0	0	1	0	0	1
9	1	0	0	0	0	1	1	0	1

Continued on next page

	Cluster complete original	Cluster complete standarized	Cluster complete withmean	Cluster single original	Cluster single standarized	Cluster single withmean	Cluster average original	Cluster average standarized	Cluster average withmean
10	0	0	0	0	0	1	1	0	1
11	0	0	0	0	0	1	1	0	1
12	1	0	0	0	0	1	1	0	1
13	0	0	0	0	0	1	1	0	1
14	0	0	0	0	0	1	1	0	1
15	1	0	1	0	0	1	0	0	1
16	1	0	0	0	0	1	1	0	1
17	1	0	0	0	0	1	1	0	1
18	1	0	0	1	0	1	0	0	1
19	0	0	1	0	0	1	0	0	1
20	1	1	0	0	1	1	1	1	1
21	1	1	0	0	1	1	1	1	1
22	1	1	0	0	1	1	1	1	1
23	1	1	0	0	1	1	1	1	1
24	1	1	0	0	1	1	1	1	1
25	1	1	0	0	1	1	1	1	1
26	1	1	0	0	1	1	1	1	1
27	1	1	0	0	1	1	1	1	1
28	1	1	0	0	1	1	1	1	1
29	1	1	0	0	1	1	1	1	1
30	1	1	0	0	1	1	1	1	1
31	1	1	0	0	1	1	1	1	1
32	1	1	0	0	1	1	1	1	1
33	1	1	0	0	1	1	1	1	1
34	1	1	0	0	1	1	1	1	1
35	1	1	0	0	1	1	1	1	1
36	1	1	0	0	1	1	1	1	1
37	1	1	0	0	1	1	1	1	1
38	1	1	0	0	1	1	1	1	1
39	1	1	0	0	1	1	1	1	1

Tabla 12: Resultados de clustering con distintas escalas en los datos y distintos tipos de enlace

Método 3 del inciso c) del ejercicio 11

	original	original variance distance	standarized	standarized variance distance	withmean	withmean variance distance
0	522	54.29	10	35.67	10	35.67

Continued on next page

	original	original variance distance	standarized	standarized variance distance	withmean	withmean variance distance
1	592	54.73	591	35.68	591	35.68
2	537	54.88	554	35.68	554	35.68
3	501	55.25	569	35.69	569	35.69
4	564	55.36	539	35.70	539	35.70
5	545	55.79	510	35.72	510	35.72
6	508	55.85	534	35.77	534	35.77
7	560	55.90	553	35.78	553	35.78
8	598	56.09	507	35.81	507	35.81
9	553	56.25	563	35.86	563	35.86
10	11	56.34	568	35.89	568	35.89
11	512	56.64	537	35.92	537	35.92
12	561	57.82	583	36.00	583	36.00
13	539	58.98	592	36.07	592	36.07
14	583	59.48	550	36.09	550	36.09
15	528	59.99	564	36.46	564	36.46
16	567	60.07	589	36.66	589	36.66
17	599	60.33	599	36.98	599	36.98
18	581	60.39	588	37.15	588	37.15
19	548	60.57	501	37.28	501	37.28

Tabla 14: Genes que minimizan la razón de errores

Método 3 del inciso c) del ejercicio 11

	complete	complete error ratio	single	single error ratio	average	average error ratio
0	501	0.050	501	0.050	501	0.050
1	550	0.075	564	0.100	589	0.075
2	589	0.075	568	0.125	550	0.075
3	565	0.100	19	0.150	553	0.100
4	569	0.100	589	0.225	564	0.100
5	592	0.100	569	0.275	569	0.100
6	548	0.100	552	0.350	574	0.100
7	12	0.100	97	0.350	546	0.125
8	529	0.100	591	0.350	561	0.125
9	546	0.125	164	0.350	513	0.125
10	553	0.125	12	0.375	510	0.125
11	544	0.125	509	0.375	507	0.125
12	561	0.125	10	0.375	562	0.125
13	535	0.125	557	0.400	567	0.125

Continued on next page

	complete	complete error ratio	single	single error ratio	average	average error ratio
14	587	0.125	898	0.400	522	0.125
15	586	0.125	640	0.400	568	0.125
16	507	0.125	587	0.400	587	0.125
17	567	0.125	599	0.400	592	0.125
18	568	0.125	562	0.400	528	0.150
19	513	0.125	749	0.400	512	0.150

Tabla 16: Genes que minimizan la razón de errores

Anexo 2: Figuras relevantes

Dendrogramas para el ejercicio 10

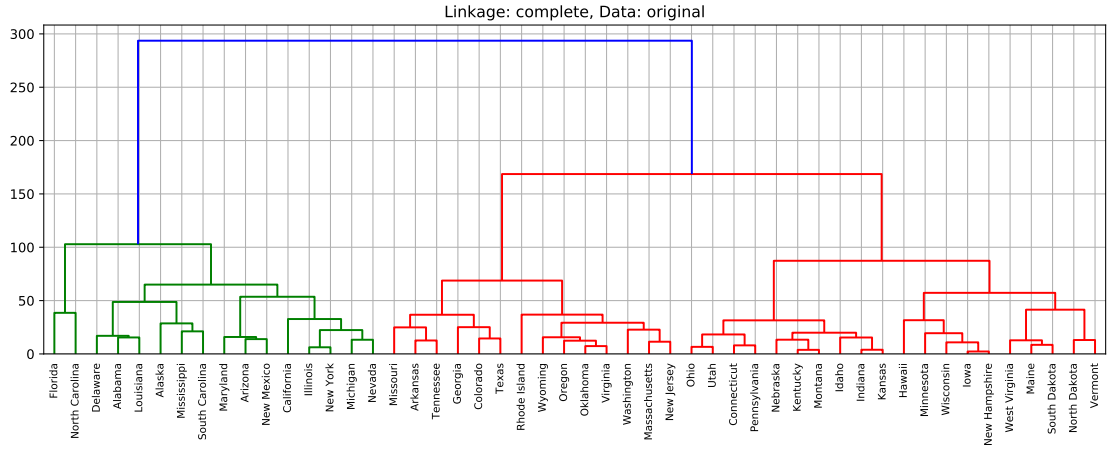

Figura 2: Clustering para datos originales con enlace completo

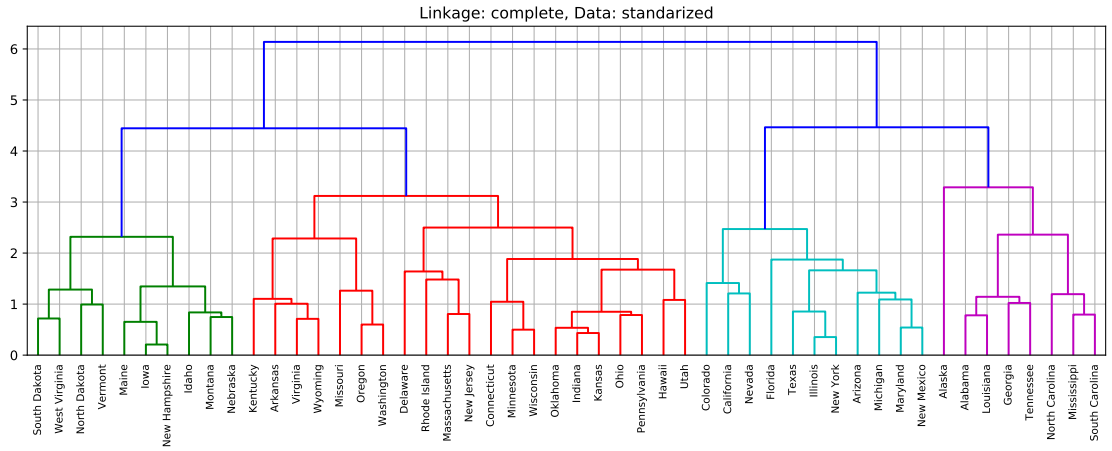

Figura 3: Clustering para datos estandarizados con enlace completo

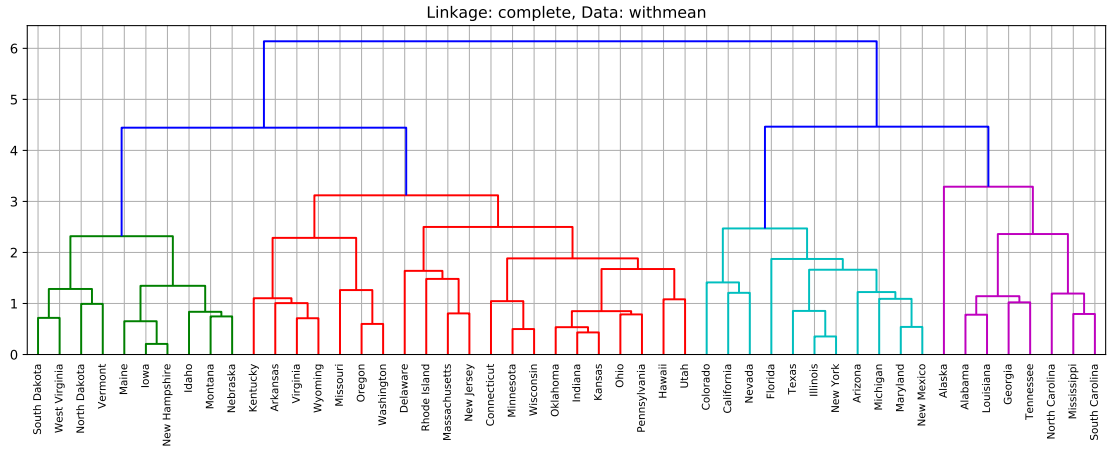

Figura 4: Clustering para datos con $\sigma^2 = 1$ con enlace completo

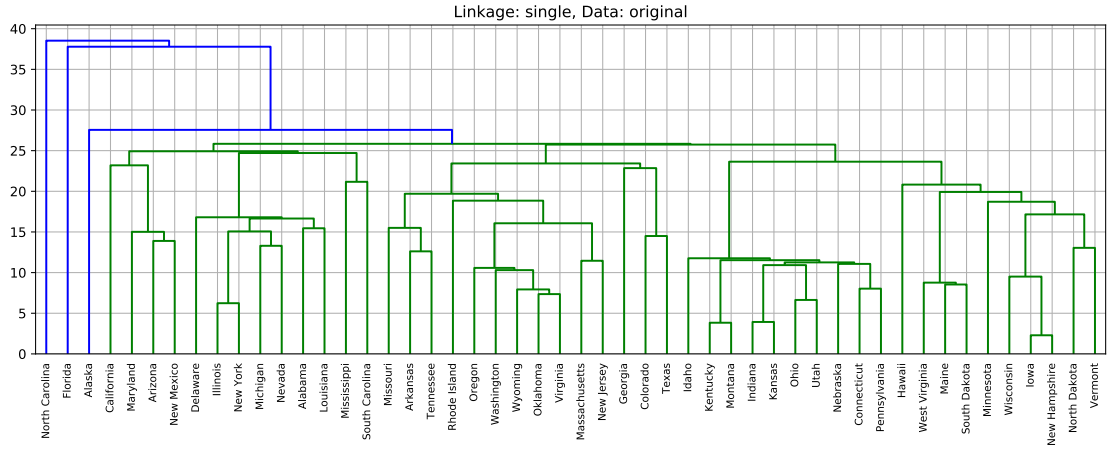

Figura 5: Clustering para datos originales con enlace completo

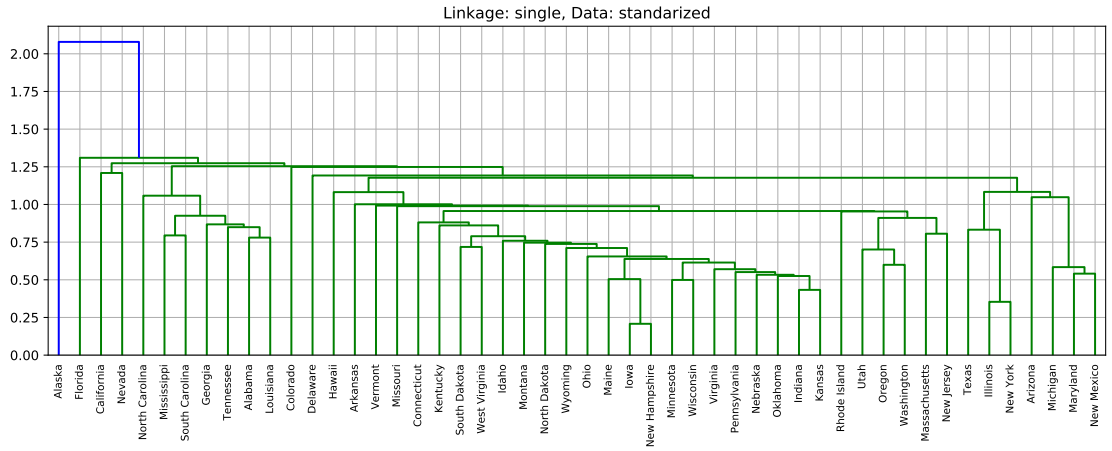

Figura 6: Clustering para datos estandarizados con enlace completo

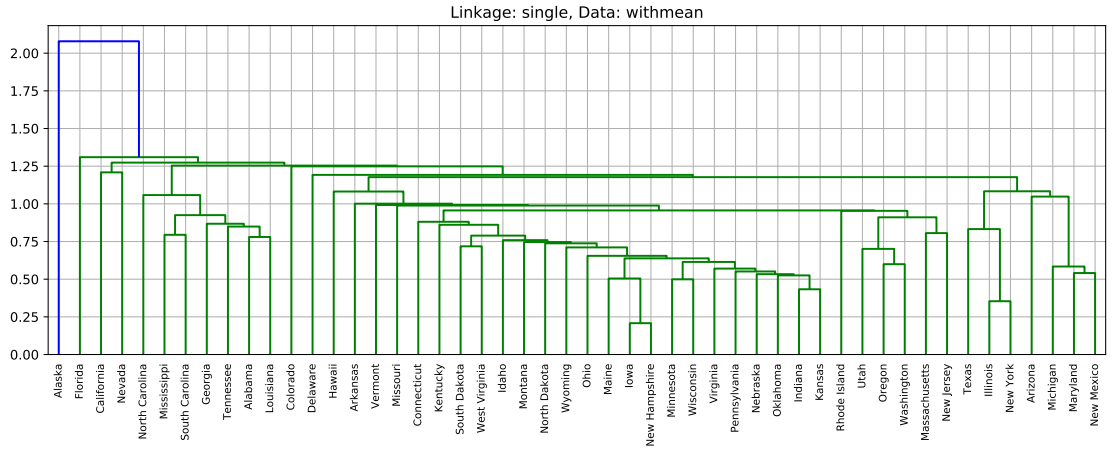

Figura 7: Clustering para datos con $\sigma^2 = 1$ con enlace completo

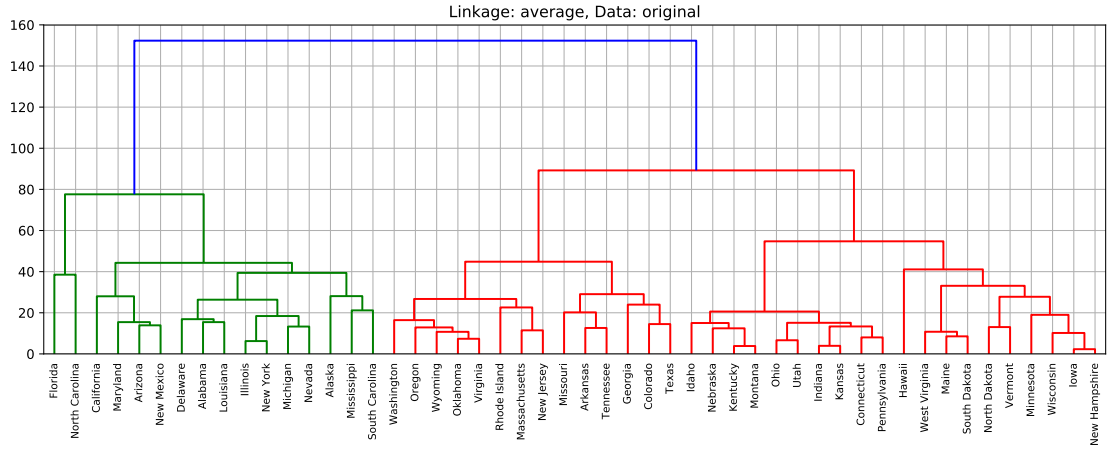

Figura 8: Clustering para datos originales con enlace completo

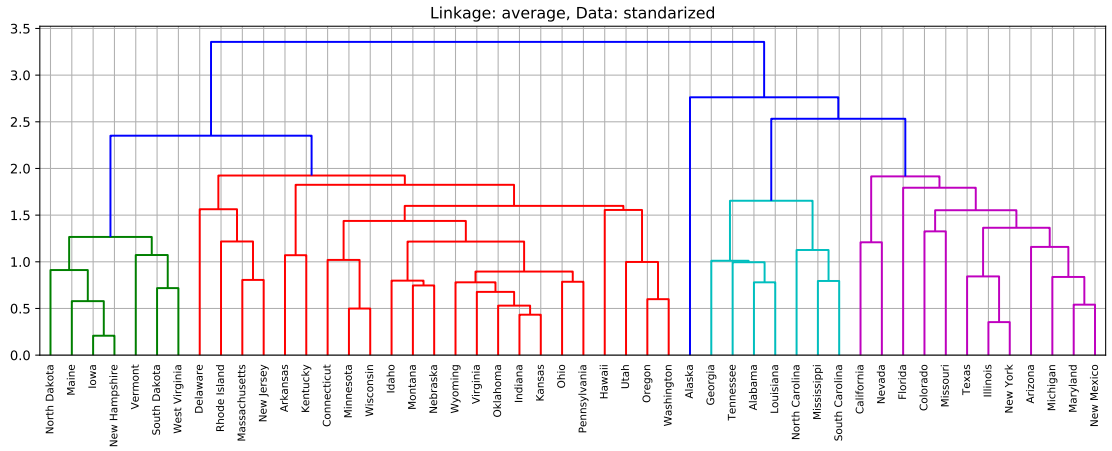

Figura 9: Clustering para datos estandarizados con enlace completo

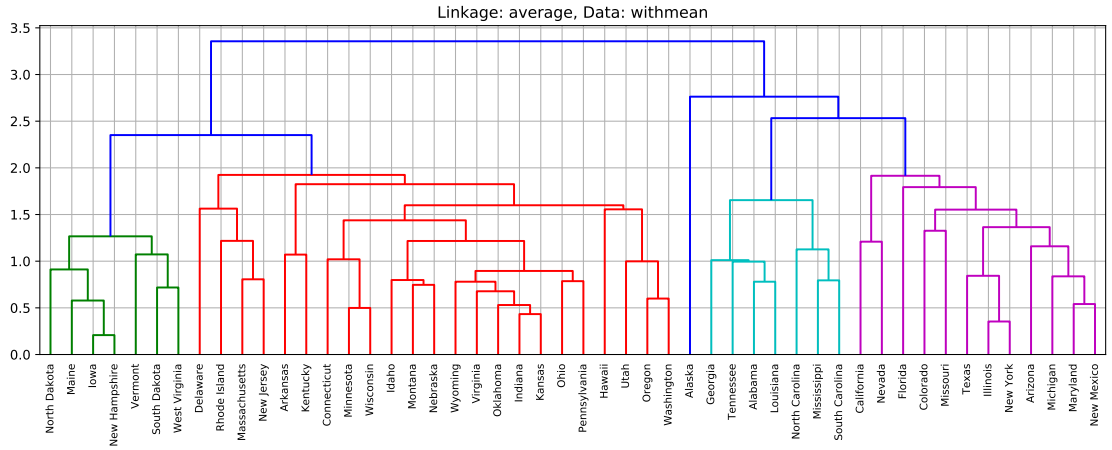

Figura 10: Clustering para datos con $\sigma^2 = 1$ con enlace completo

Resultados del clustering con K-means para el ejercicio 10

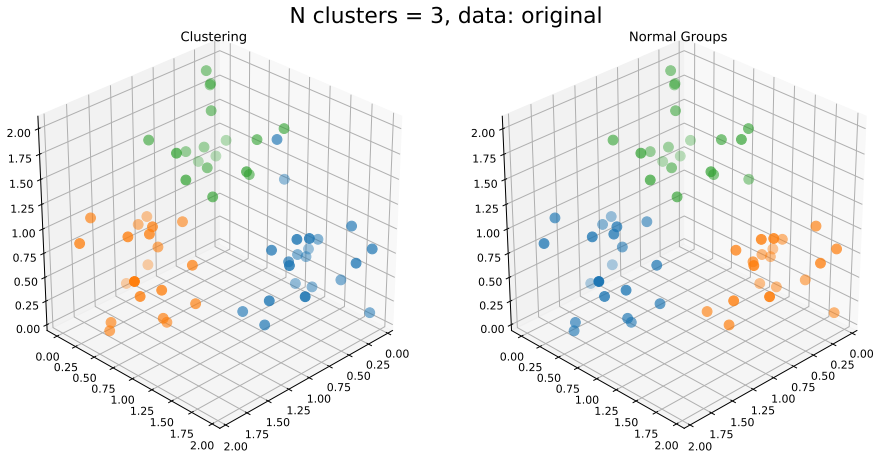

Figura 11: Clustering para datos originales con $K = 3$

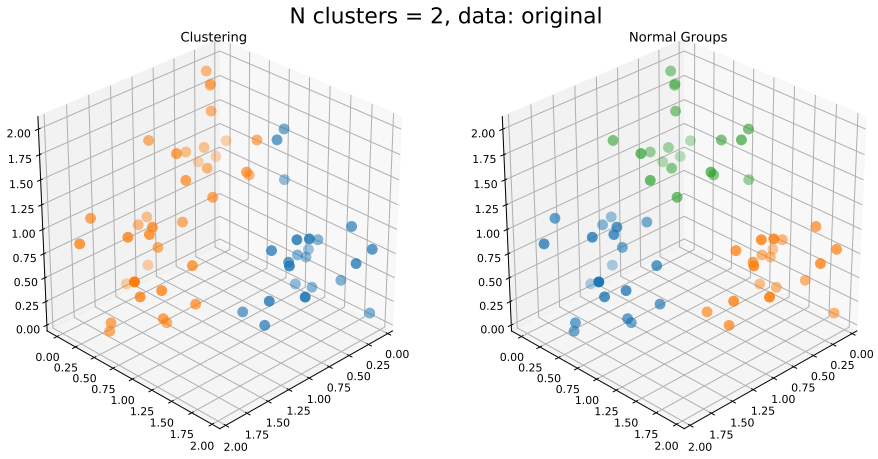

Figura 12: Clustering para datos originales con $K = 2$

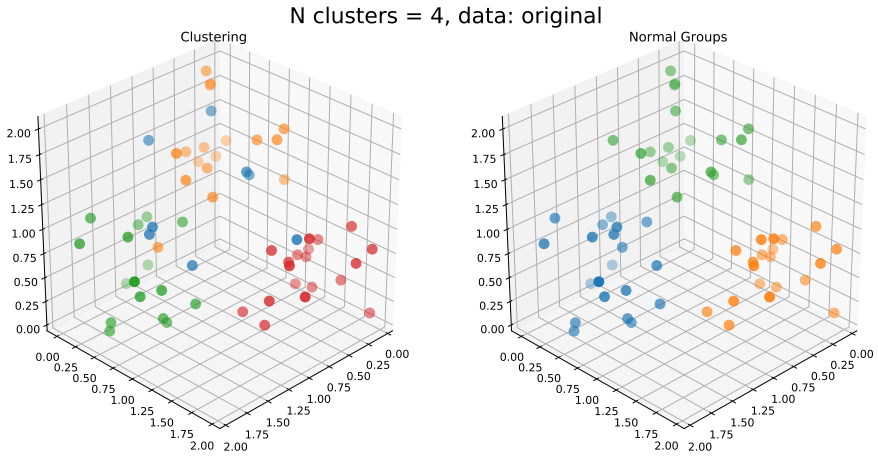

Figura 13: Clustering para datos originales con $K = 4$

Figura 14: Clustering para datos estandarizados con $K = 3$

Figura 15: Clustering para datos estandarizados con $K = 2$

Figura 16: Clustering para datos estandarizados con $K = 4$

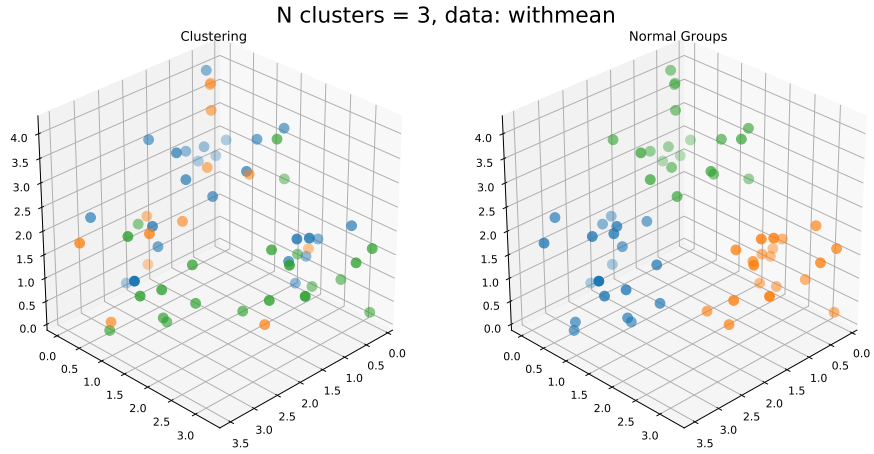

Figura 17: Clustering para datos con $\sigma^2 = 1$ con $K = 3$

Figura 18: Clustering para datos con $\sigma^2 = 1$ con $K = 2$

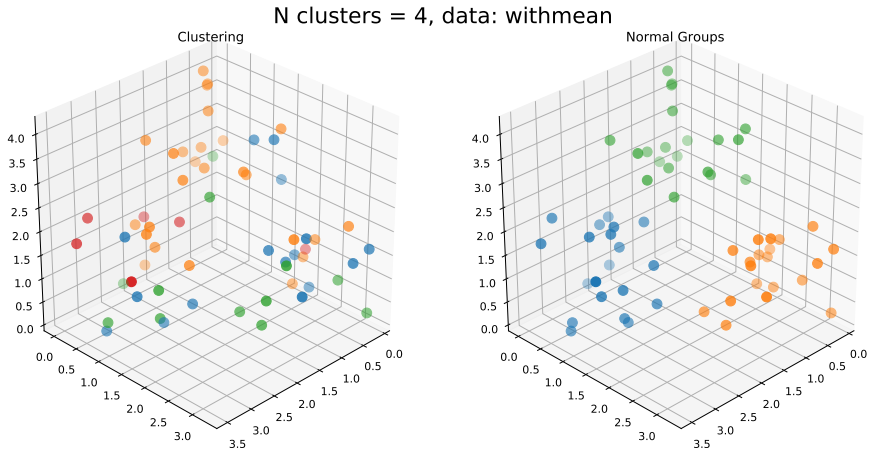

Figura 19: Clustering para datos con $\sigma^2 = 1$ con $K = 4$

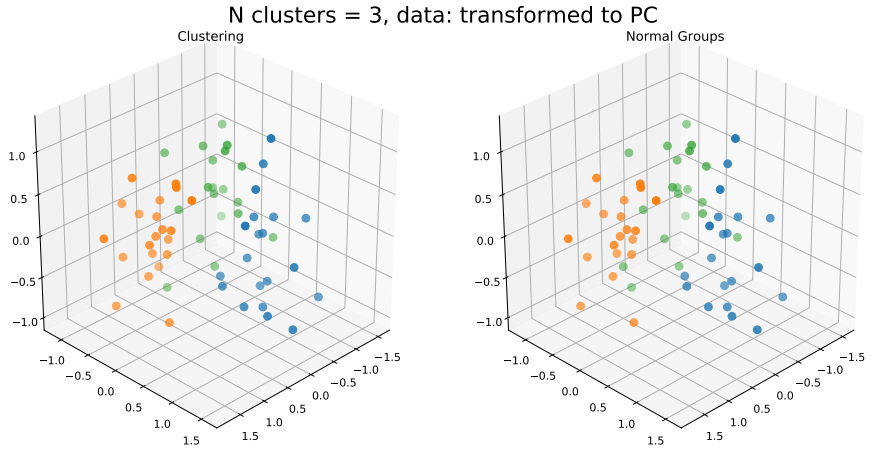

Figura 20: Clustering sobre las primeras dos componentes principales con $K = 3$

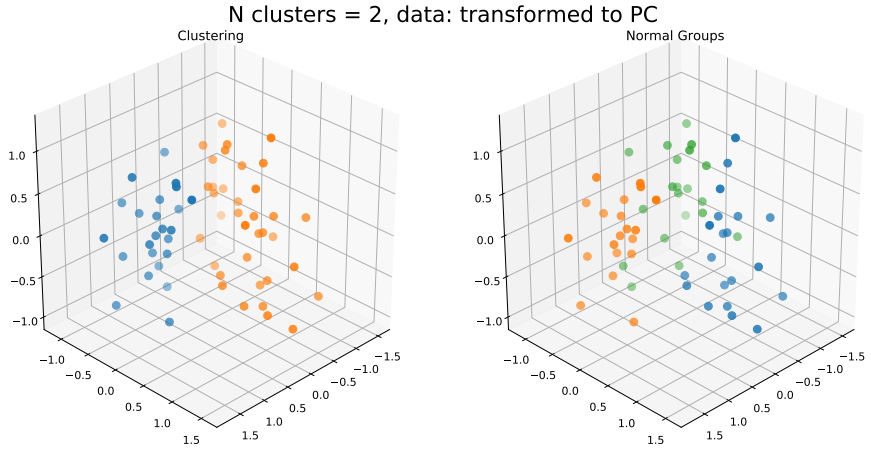

Figura 21: Clustering sobre las primeras dos componentes principales con $K = 2$

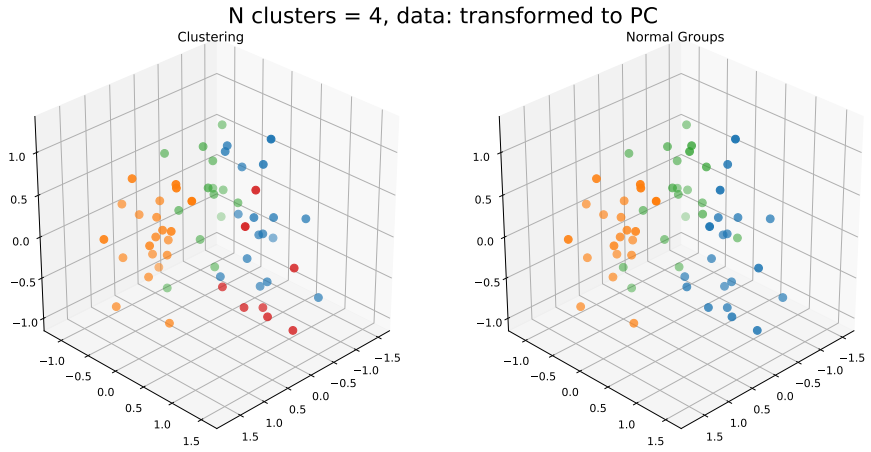

Figura 22: Clustering sobre las primeras dos componentes principales con $K = 4$

Dendogramas para el ejercicio 11

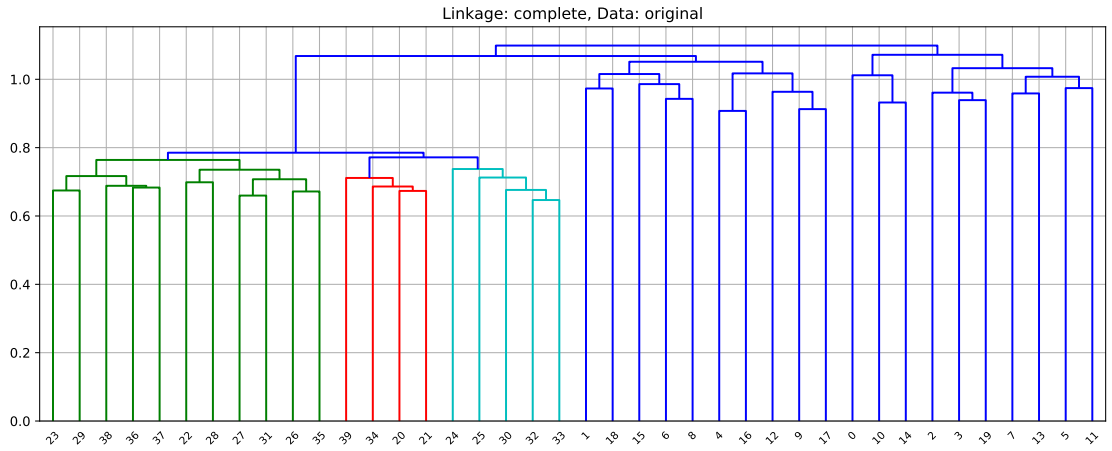

Figura 23: Clustering para datos originales con enlace completo

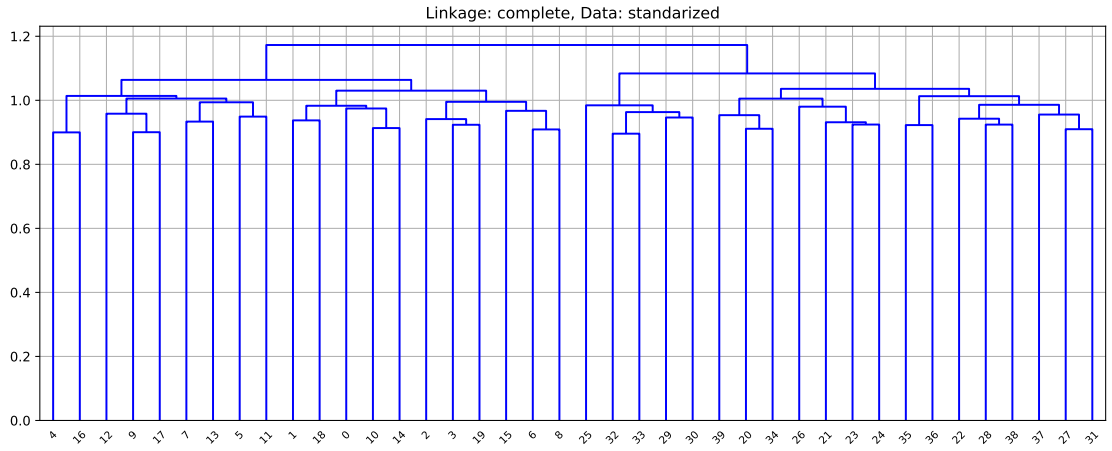

Figura 24: Clustering para datos estandarizados con enlace completo

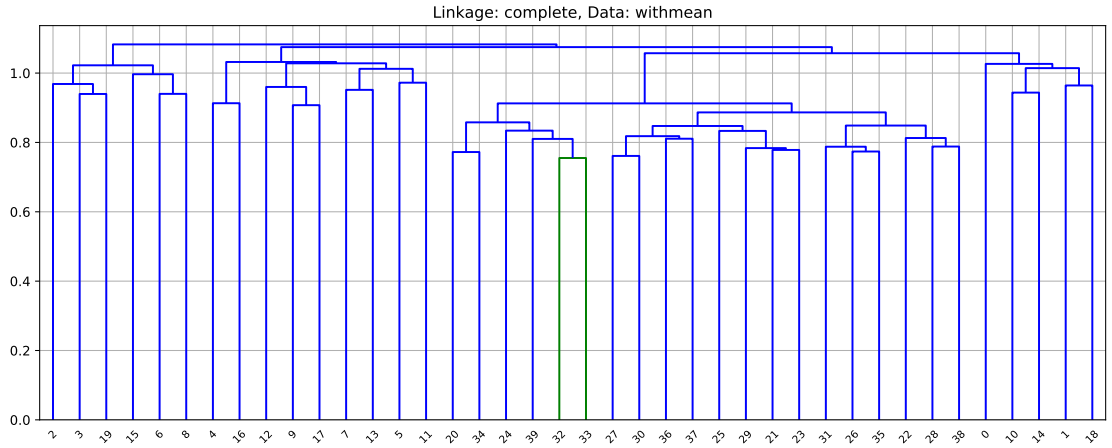

Figura 25: Clustering para datos con $\sigma^2 = 1$ con enlace completo

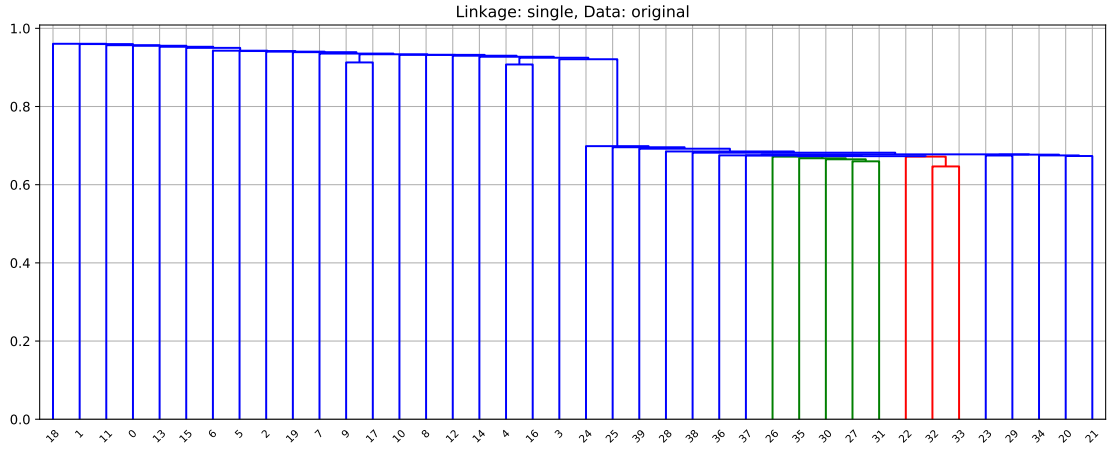

Figura 26: Clustering para datos originales con enlace completo

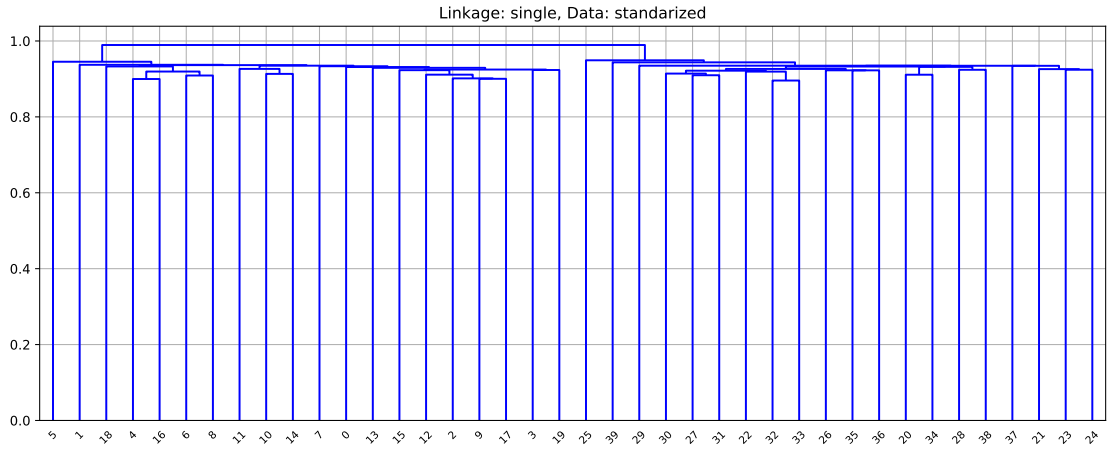

Figura 27: Clustering para datos estandarizados con enlace completo

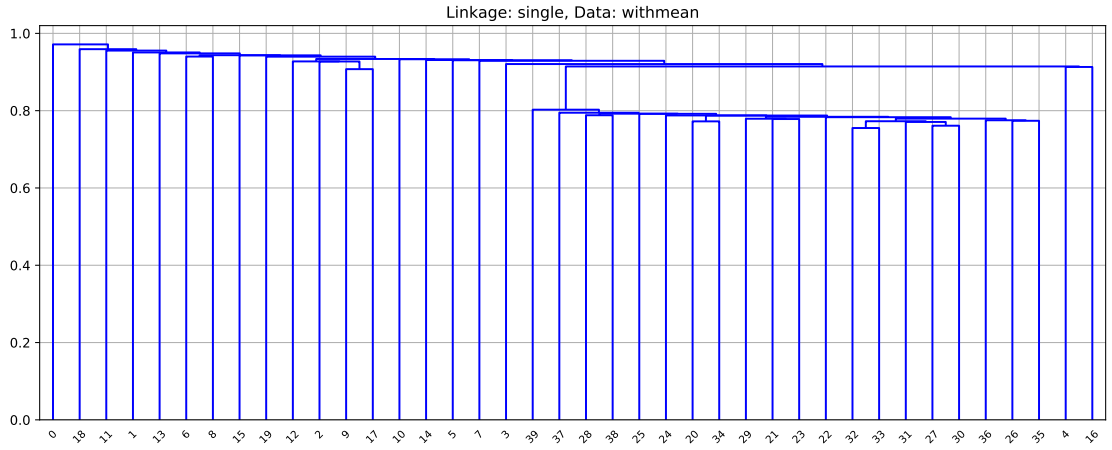

Figura 28: Clustering para datos con $\sigma^2 = 1$ con enlace completo

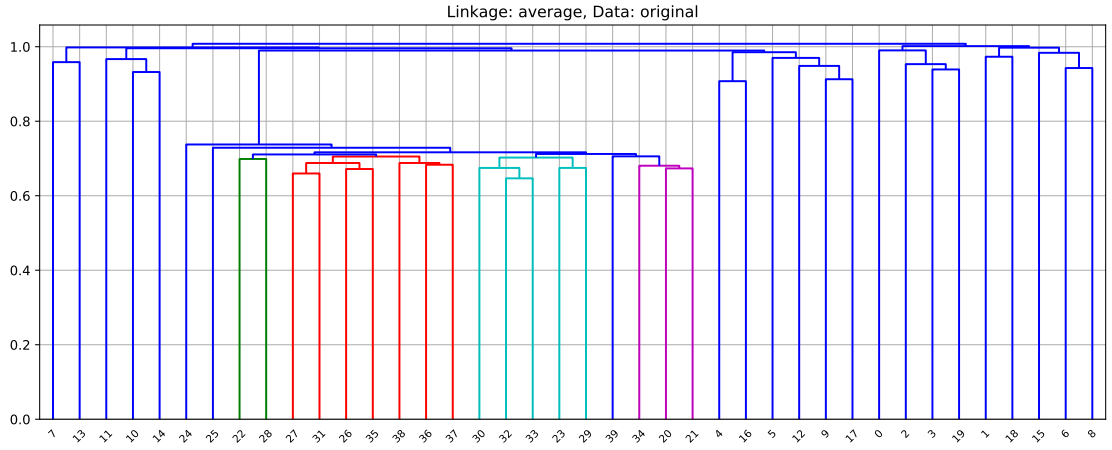

Figura 29: Clustering para datos originales con enlace completo

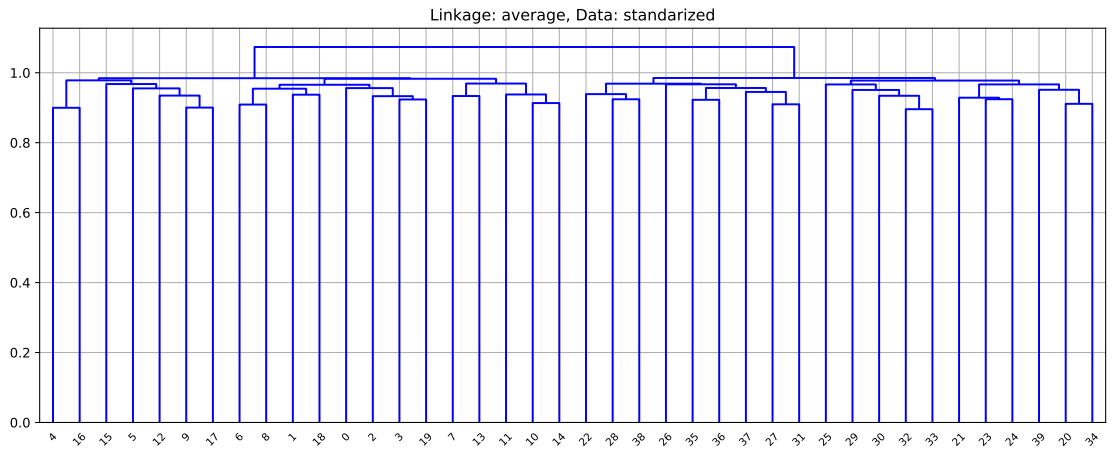

Figura 30: Clustering para datos estandarizados con enlace completo

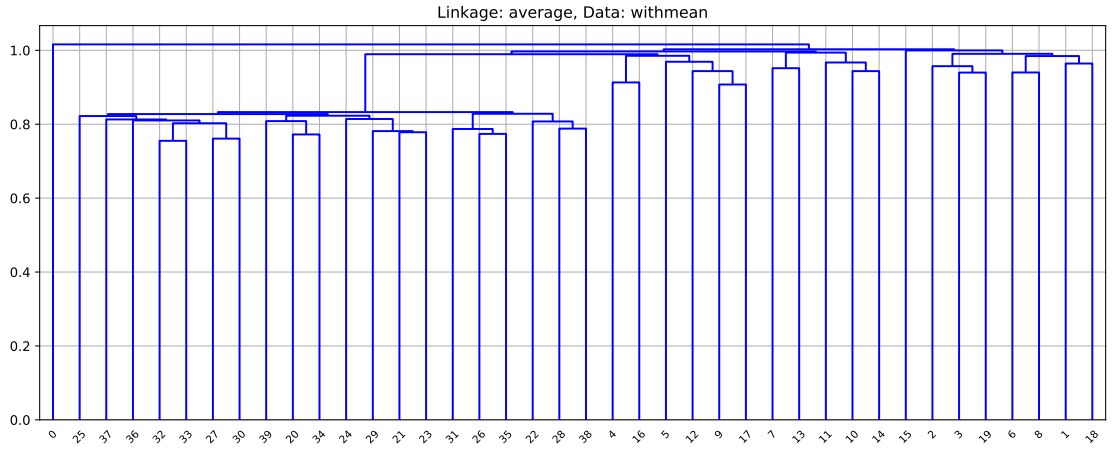

Figura 31: Clustering para datos con $\sigma^2 = 1$ con enlace completo

Anexo 3: código en Python de los problemas

```
import sklearn.cluster as clus
import sklearn.decomposition as dc
import sklearn.preprocessing as pr
from scipy.cluster.hierarchy import dendrogram
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import pandas as pd
import os
if not(os.path.isdir("tarea")):
    os.mkdir("tarea")
# function to scale data acording to different
def dataScaled(data):
    # Creating dictionary to store the different data frames
    data = {"original":df}
    # Standarizing data to have mean 0 and variance 1
    scaler = pr.StandardScaler()
    scaler.fit(df)
    data["standarized"] = pd.DataFrame(scaler.transform(df),index=df.index
    ,columns=df.columns)
    # Centering data to have variance 1
    scaler = pr.StandardScaler(with_mean=False)
    scaler.fit(df)
    data["withmean"] = pd.DataFrame(scaler.transform(df),index=df.index,
    columns=df.columns)
    return data
# function to export the PCA coefficient table
def compCoefTable(model,data):
    p = len(data.columns)
    c = int(np.floor(np.log10(p)+1))
    names = ["PC"+"{0}".format(i).zfill(c) for i in range(len(data.columns
    ))]
    return pd.DataFrame(model.components_, index = names,columns = data.
    columns)

# function to export the singular value table
def singvalTable(model,data):
    p = len(data.columns)
    c = int(np.floor(np.log10(p)+1))
    names = ["PC"+"{0}".format(i).zfill(c) for i in range(len(data.columns
    ))]
    return pd.DataFrame(model.singular_values_,index =names).transpose()

# function to create the PVE table using the scikit-learn package
def pveTable(model,data):
    p = len(data.columns)
    c = int(np.floor(np.log10(p)+1))
    names = ["PC"+"{0}".format(i).zfill(c) for i in range(len(data.columns
    ))]
    return pd.DataFrame(model.explained_variance_ratio_,index=names,
    columns = ["PVE"]).transpose()

```

```
#function to calculate the PVE of the m-th principal component
def pve(m,model,data):
    n,p = df.shape
    norm = 0
    s = 0
    for i in range(n):
        temp = 0
        for j in range(p):
            #print("s = {0}".format(s))
            #print("norm = {0}".format(norm))
            temp = temp + model.components_[m][j]*data.iloc[i][j]
            norm = norm + data.iloc[i][j]**2
        s = s + temp**2
    return s/norm

#function to create the PVE table using the PVE(model,data) function
def pveFromFormulaTable(model,data):
    p = len(data.columns)
    c = int(np.floor(np.log10(p)+1))
    names = ["PC"+"{0}".format(i).zfill(c) for i in range(len(data.columns
    ))]
    vals = [pve(i,model,data) for i in range(p)]
    return pd.DataFrame(vals,index=names,columns = ["PVE"]).transpose()

# Auxiliar function taken from https://scikit-learn.org/stable/
    auto_examples/cluster/plot_agglomerative_dendrogram.html
# for plotting dendogram of agglomerative clustering
def plotDendogram(model, **kwargs):
    # Create linkage matrix and then plot the dendrogram

    # create the counts of samples under each node
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1  # leaf node
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

    linkage_matrix = np.column_stack([model.children_, model.distances_,
                                      counts]).astype(float)

    # Plot the corresponding dendrogram
    dendrogram(linkage_matrix, **kwargs)

# function for returning lists with all clusters
def clusters(fit,data):
    clu = [[] for i in range(fit.n_clusters_)]
    for i in range(fit.n_leaves_):
        clu[fit.labels_[i]].append(data.index[i])
    return clu
```

```
98
99  def plotClusteringResults(fit,df,s=20):
100     #labels = (fit.labels_-fit.labels_[0]) % nclus
101     labels = fit.labels_
102     clusColors = ["C{0}".format(i) for i in labels]
103     normColors = ["C{0}".format(i) for i in df["initG"]]
104
105     ax1 = fig.add_subplot(121, projection='3d')
106     ax2 = fig.add_subplot(122, projection='3d')
107
108     ax1.scatter(df.iloc[:,0],df.iloc[:,1],df.iloc[:,2],s=s,c=clusColors)
109     ax1.view_init(30, 45)
110     ax1.set_title("Clustering")
111
112     ax2.scatter(df.iloc[:,0],df.iloc[:,1],df.iloc[:,2],s=s,c=normColors)
113     ax2.view_init(30, 45)
114     ax2.set_title("Normal Groups")
115
116 # Exercise 8
117
118 # Reading data
119 df = pd.read_csv("USArrests.csv",header=0,index_col=0)
120 print(df.head())
121 df.head().to_latex(buf="tarea/1-head.tex")
122 # Making PCA over standarized data for effecive PCA
123
124 data = dataScaled(df)
125 print(data["standarized"].head())
126 model = dc.PCA()
127 model.fit(data["standarized"])
128 x = compCoefTable(model,data["standarized"])
129 print(x)
130 x.to_latex(buf="tarea/1-coeffTable.tex",float_format = "{:0.4f}".format)
131 print(singvalTable(model,data["standarized"]))
132 # calculating PVE table from package
133 x = pveTable(model,data["standarized"])
134 print(x)
135 x.to_latex(buf="tarea/1-pve.tex",float_format = "{:0.4f}".format)
136 # calculating PVE table using formula
137 x = pveFromFormulaTable(model,data["standarized"])
138 print(x)
139 x.to_latex(buf="tarea/1-pveForm.tex",float_format = "{:0.4f}".format)
140
141 # Exercise 9
142
143 print(data["withmean"].head())
144 # Performinc hierarchical clustering over different linkages
145 res = pd.DataFrame(index=data["original"].index)
146 for l in ["complete","single","average"]:
147     for name, df in data.items():
148         model = clus.AgglomerativeClustering(linkage=l,affinity = "
    euclidean",distance_threshold=0.0,n_clusters=None)
149         fit  = model.fit(X=df)
150         plt.figure(figsize=(12,5))
```

```
151         plotDendogram(fit,labels=df.index)
152         plt.grid()
153         plt.title("Linkage: {0}, Data: {1}".format(l,name))
154         plt.tight_layout()
155         plt.savefig("tarea/2-{0}{1}.pdf".format(l,name))
156         plt.show()
157         model = clus.AgglomerativeClustering(linkage=l,affinity = "
    euclidean",n_clusters=3)
158         fit  = model.fit(X=df)
159         # making first observation be in cluster 0
160         res["Cluster {0} {1}".format(l,name)] = (fit.labels_ - fit.labels_
    [0]) % 3
161 tabs = "c"
162 for i in range(0,res.shape[1]):
163     tabs = tabs + "p{1.5cm}"
164 res.index = [a[0:7] for a in res.index]
165 res.to_latex(buf = "tarea/2-clusresults.tex",column_format=tabs,longtable=
    True)
166
167 # Exercise 10
168
169 n=50
170 ng = 3
171 gobs = 20
172 g = []
173 df = pd.DataFrame()
174 for i in range(ng):
175     mean = np.zeros(n)
176     mean[i] = 1.0
177     obs = np.array([np.random.rand(n) + mean for i in range(gobs)])
178     aux = pd.DataFrame(obs)
179     aux["initG"] = i
180     df = df.append(aux,ignore_index=True)
181 print(df.head())
182 data = dataScaled(df.drop(columns="initG"))
183 for name,df1 in data.items():
184     df1["initG"] = df["initG"]
185 model = clus.KMeans(n_clusters=3)
186 fit = model.fit(X=data["original"].drop(columns="initG"))
187 fig = plt.figure(figsize=(12,6))
188 plotClusteringResults(fit,data["original"],s=80)
189 plt.tight_layout()
190 plt.savefig("tarea/3-initClus.pdf")
191 plt.show()
192 model = dc.PCA()
193 model.fit(data["original"].drop(columns="initG"))
194 compCoefTable(model,data["original"].drop(columns="initG"));
195 trans = model.transform(data["original"].drop(columns="initG"))
196 trans = pd.DataFrame(trans,columns = pveTable(model,data["original"].drop(
    columns="initG")).columns)
197 trans["initG"] = data["original"]["initG"]
198 normColors = ["C{0}".format(i) for i in data["original"]["initG"]]
199 plt.figure()
200 plt.title("Principal components")
```

```
plt.xlabel("PC0")
plt.ylabel("PC1")
plt.scatter(trans["PC00"],trans["PC01"], color = normColors)
plt.grid()
plt.tight_layout()
plt.savefig("tarea/3-pca.pdf")
plt.show()
print(pveTable(model,data["original"].drop(columns="initG")))
res = pd.DataFrame(index = data["original"].index)
for name,df in data.items():
    for j in [3,2,4]:
        model = clus.KMeans(n_clusters=j)
        fit = model.fit(X=df.drop(columns="initG"))
        labels = (fit.labels_ - fit.labels_[0] )% j
        res["K: {0} , data: {1}".format(j,name)] = labels
        fig = plt.figure(figsize=(12,6))
        plotClusteringResults(fit,df,s=80)
        plt.suptitle("N clusters = {0}, data: {1}".format(j,name),fontsize
    =20)
        plt.tight_layout()
        plt.savefig("tarea/3-{1}-{0}.pdf".format(j,name))
        plt.show()
tabs = "c"
for i in range(0,res.shape[1]):
    tabs = tabs + "p{1.5cm}"
res.to_latex(buf = "tarea/3-clusresults.tex",column_format=tabs,longtable=
    True)
for j in [3,2,4]:
    model = clus.KMeans(n_clusters=j)
    fit = model.fit(X=trans.drop(columns="initG"))
    fig = plt.figure(figsize=(12,6))
    plotClusteringResults(fit,trans,s=50)
    plt.suptitle("N clusters = {0}, data: transformed to PC".format(j,name
    ),fontsize=20)
    plt.tight_layout()
    plt.savefig("tarea/3-fullpcs-{0}.pdf".format(j))
    plt.show()
res = pd.DataFrame(index=data["original"].index)
for j in [3,2,4]:
    model = clus.KMeans(n_clusters=j)
    fit = model.fit(X=trans.iloc[:,[0,1]])
    labels = (fit.labels_ - fit.labels_[0]) % j
    res["data: pca 2 vars, K={0}".format(j)] = labels
    fig = plt.figure(figsize=(12,6))
    plotClusteringResults(fit,trans,s=50)
    plt.suptitle("N clusters = {0}, data: transformed to PC".format(j,name
    ),fontsize=20)
    plt.tight_layout()
    plt.savefig("tarea/3-2pcs-{0}.pdf".format(j))
    plt.show()
tabs = "c"
for i in range(0,res.shape[1]):
    tabs = tabs + "p{3cm}"

```

```
res.to_latex(buf = "tarea/3-clusresults-pca.tex",column_format=tabs,
    longtable=True)

# Exercise 11

# Reading Data
df = pd.read_csv("Ch10Ex11.csv",header=None)
df = df.transpose()
data = dataScaled(df)
print(df.head().iloc[:,0:5])
df.head().iloc[:,0:5].to_latex("tarea/4-initData.tex",float_format="{:0.4f
    }".format)
res = pd.DataFrame(index=data["original"].index)
for l in ["complete","single","average"]:
    for name, df in data.items():
        model = clus.AgglomerativeClustering(linkage=l,affinity = "
    correlation",distance_threshold=0.0,n_clusters=None)
        fit  = model.fit(X=df)
        plt.figure(figsize=(12,5))
        plotDendogram(fit,labels=df.index)
        plt.grid()
        plt.title("Linkage: {0}, Data: {1}".format(l,name))
        plt.tight_layout()
        plt.savefig("tarea/4-{0}{1}.pdf".format(l,name))
        plt.show()
        model = clus.AgglomerativeClustering(linkage=l,affinity = "
    correlation",n_clusters=2)
        fit  = model.fit(X=df)
        # making first observation be in cluster 0
        res["Cluster {0} {1}".format(l,name)] = (fit.labels_ - fit.labels_
    [0]) % 2
tabs = "c"
for i in range(0,res.shape[1]):
    tabs = tabs + "p{1.5cm}"
res.to_latex(buf="tarea/4-clusres.tex",longtable=True,column_format=tabs)
# method 1: calculating variance and getting gen that maximizes it
df = data["standarized"]
genDif = {}
for name,df in data.items():
    health = df.iloc[0:20,:]
    sick = df.iloc[20:,:]
    norms = []
    for i in data["original"].columns:
        diffvec = [health.loc[n,i] - sick.loc[m,i] for n in health.index
    for m in sick.index]
        norms.append(np.linalg.norm(diffvec))
    genDif[name] = norms.index(max(norms))
print(genDif)
for name,gen in genDif.items():
    plt.figure()
    plt.title("Gen that maximizes |A_k|: {0} \n Linkage: {1}".format(gen,
    name))
    plt.hist(health.loc[:,gen],alpha=0.7)
    plt.hist(sick.loc[:,gen],alpha=0.7)
```

```
    plt.grid()
    plt.tight_layout()
    plt.savefig("tarea/4-Ak-{0}.pdf".format(name))
    plt.show()
# Method 1: calculating variance and getting 20 gens that maximize it
genDif = {}
dists = {}
n=20
for name,df in data.items():
    health = df.iloc[0:20,:]
    sick = df.iloc[20:,:]
    norms = []
    for i in data["original"].columns:
        diffvec = [health.loc[n,i] - sick.loc[m,i] for n in health.index
   for m in sick.index]
        norms.append(np.linalg.norm(diffvec))
    genDif[name] = list(np.argsort(norms))[-n:]
    dists[name] = list(sorted(norms))[-n:]
print(genDif)
res = pd.DataFrame()
for name,lis in genDif.items():
    res[name] = lis
    res[name + " variance distance"] = dists[name]
tabs = "c"
for i in range(res.shape[1]):
    tabs += "p{2cm}"
res.to_latex("tarea/4-variance-distance.tex",column_format=tabs,longtable=
   True,float_format="{:0.2f}".format)
# Method 2: looking for a gen which removing it causes clustering fail
df = data["standarized"]
for l in ["complete","single","average"]:
    for i in df.columns:
        aux = df.drop(columns=i)
        model = clus.AgglomerativeClustering(linkage=l,affinity = "
   correlation",n_clusters=2)
        fit  = model.fit(X=aux)
        cluslabels = [0 for i in range(20)] + [1 for i in range(20)]
        labels = list((fit.labels_ - fit.labels_[0] ) % 2)
        if cluslabels != labels:
            print("Whitout this gen, clustering fails")
            print(cluslabels)
            print(labels)
            print(i)
# Looking for the gene that makes posible the clustering with only with
   him as variable
genDif = {}
for l in ["complete","single","average"]:
    for i in df.columns:
        aux = pd.DataFrame(df[i])
        # distance has to be euclidean in order to avoid problems
        model = clus.AgglomerativeClustering(linkage=l,affinity = "
   euclidean",n_clusters=2)
        fit = model.fit(X = aux)
        cluslabels = [0 for i in range(20)] + [1 for i in range(20)]
```

```
        labels = list((fit.labels_ - fit.labels_[0] ) % 2)
        if cluslabels == labels:
            print("With this gene, clustering works ")
            print(i)
# Method 3: Looking for the gene that minimizes the classification error
genDif = {}
for l in ["complete","single","average"]:
    vec = []
    for i in df.columns:
        aux = pd.DataFrame(df[i])
        # distance has to be euclidean in order to avoid problems
        model = clus.AgglomerativeClustering(linkage=l,affinity = "
    euclidean",n_clusters=2)
        fit = model.fit(X = aux)
        cluslabels = [0 for i in range(20)] + [1 for i in range(20)]
        labels = list((fit.labels_ - fit.labels_[0] ) % 2)
        errors = 0
        for i,lab in enumerate(cluslabels):
            if labels[i] != lab:
                errors = errors + 1
        vec.append(errors/df.shape[0])
    mingens = [x[0] for x in enumerate(vec) if vec[x[0]]==min(vec)]
    if len(mingens) == 1:
        genDif[l] = mingens[0]
    else:
        print("These gens minimize error:")
        print(mingens)
print(genDif)
for name,gen in genDif.items():
    plt.figure()
    plt.title("Gen with minimal classification error: {0} \n Linkage: {1}"
    .format(gen,name))
    plt.hist(health.loc[:,gen],alpha=0.7)
    plt.hist(sick.loc[:,gen],alpha=0.7)
    plt.grid()
    plt.tight_layout()
    plt.savefig("tarea/4-minimal-{0}.pdf".format(name))
    plt.show()
# Looking for the 20 genes that minimizes the classification error
genDif = {}
errs = {}
n = 20
for l in ["complete","single","average"]:
    vec = []
    for i in df.columns:
        aux = pd.DataFrame(df[i])
        # distance has to be euclidean in order to avoid problems
        model = clus.AgglomerativeClustering(linkage=l,affinity = "
    euclidean",n_clusters=2)
        fit = model.fit(X = aux)
        cluslabels = [0 for i in range(20)] + [1 for i in range(20)]
        labels = list((fit.labels_ - fit.labels_[0] ) % 2)
        errors = 0
        for i,lab in enumerate(cluslabels):
```

```
            if labels[i] != lab:
                errors = errors + 1
        vec.append(errors/df.shape[0])
    mingens = list(np.argsort(vec)[0:n])
    genDif[l] = mingens
    errs[l] = sorted(vec)[0:n]
print(genDif)
res = pd.DataFrame()
for name,lis in genDif.items():
    res[name] = lis
    res[name + " error ratio"] = errs[name]
tabs = "c"
for i in range(res.shape[1]):
    tabs += "p{2cm}"
res.to_latex("tarea/4-errors-ratio.tex",column_format=tabs,longtable=True)
res = pd.DataFrame(index = data["original"].index)
```

Referencias

[1] Hastie et al. *An Introduction to Statitistical Learning*. Editorial Springer. Séptima edición. 2013.