



Tarea 1

Equipo:

Isabel Ruiz Mijangos
Alondra Vega Lázaro
Alejandro Zamora Reyes
Nadia Paola Jiménez Flores

Estadística multivariada

15/febrero/2024

UNIVERSIDAD La Salle
Facultad de Negocios
Estadística Multivariada
Examen

Fecha de entrega: jueves 15 de febrero de 2024

1. PCA

Ejercicio 1. Demuestra que el segundo problema de optimización para $\text{Var}(\xi_2)$ tiene solución cuando el multiplicador de lagrange λ_2 es eigenvalor de la matriz de covarianza de $X = (X_1, \dots, X_n)$.

Ejercicio 2. Considera la matriz

$$X = \begin{pmatrix} -1 & 0 & 1 \\ 0 & -1 & 1 \end{pmatrix}$$

Realiza un PCA realizando la descomposición espectral de la matriz de covarianza de X . Calcula la varianza total.

Ejercicio 3. Considera los datos `wine.data`¹. Realiza un PCA usando el código visto en clase (sin sklearn). Recuerda preprocessar tus datos. Comenta tus resultados.

Ejercicio 4. Genera una muestra aleatoria de tamaño 100 a partir de una distribución gaussiana 3-dimensional en dónde una de las variables tiene más alta varianza que las otras. Realiza un PCA. En cada caso, encuentra los eigenvalores y eigenvectores y dibuja la gráfica scree.

Ejercicio 5. Se requiere efectuar un PCA de

$$X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

(1) Verifica que obtener los eigenvalores de la matriz

$$A = \begin{pmatrix} 3 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 3 \end{pmatrix} \rightsquigarrow \text{simétrica}$$

implica obtener los eigenvalores de X .

- (2) Determina al segundo eje principal del PCA de X .
- (3) Provee las coordenadas sobre los segundos ejes por renglones y columnas del PCA de X .

Ejercicio 6. En 28 años, se han observado 4 estados metereológicos:

- ξ_1 = precipitación en julio (en mm),
- ξ_2 = temperatura media en julio (en C),
- ξ_3 = velocidad media del viento en julio (en km/h),
- ξ_4 = precipitación en septiembre (en mm).

¹<https://archive.ics.uci.edu/dataset/109/wine>

A partir de tales dator, se tiene la matriz de covarianzas:

$$S = \begin{pmatrix} 140.017 & 107.881 & 139.068 & 109.095 \\ & 106.038 & 110.0439 & 82.627 \\ & & 168.752 & 125.136 \\ & & & 108.960 \end{pmatrix}$$

y las correlaciones empícas

$$\begin{pmatrix} 0.969 & -0.102 & 0.194 & 0.116 \\ 0.907 & -0.392 & -0.106 & -0.111 \\ 0.971 & 0.156 & -0.157 & 0.092 \\ 0.943 & 0.252 & 0.092 & -0.196 \end{pmatrix}$$

- (1) Calcula las varianzas empíricas de las componentes principales.
- (2) Calcular la parte de la varianza ξ_1 explicada por las dos últimas componentes principales, y la parte de la varianza de ξ_2 que explicada por las dos primeras componentes principales.
- (3) Realiza la proyección de variables sobre los dos primeros ejes principales y comenta tus resultados.

Ejercicio 7. Sea X un vector aleatorio en dimensión 4 de media μ y matriz de covarianza $\Sigma = (\sigma_{ij})$. Supongamos que $\sigma_{ii} = 1$ para todo i .

- (1) Sea $0 < \rho < 1$. En la **Figura 1** se muestran dos gráficas, una de ellas presenta la proyección de variables sobre las dos primeros ejes principales. ¿Cuál es? Justifica tu respuesta.

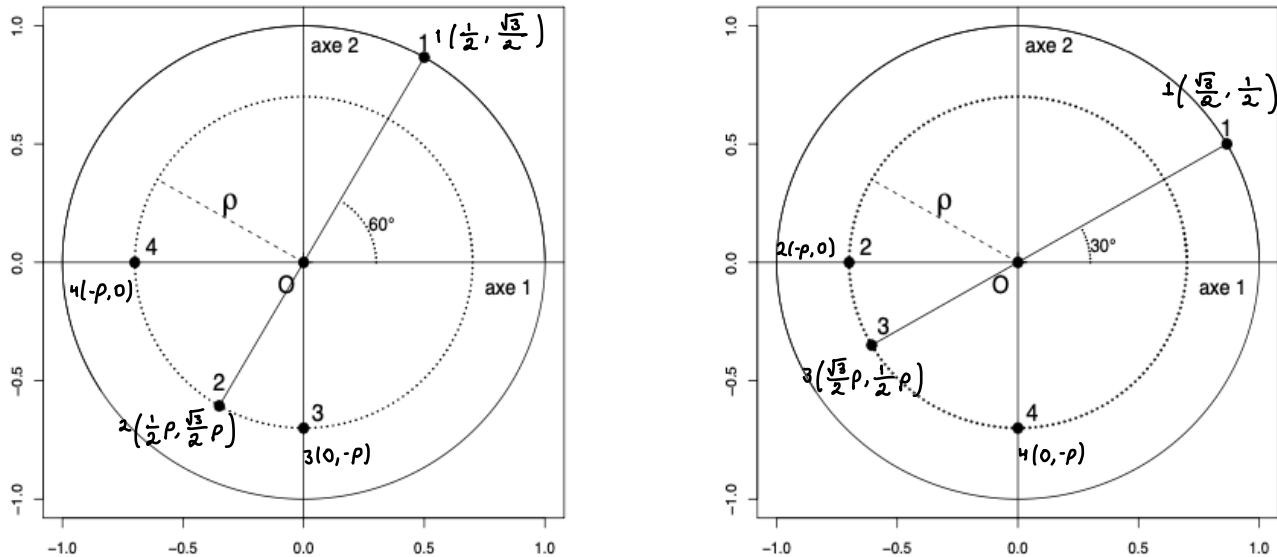


FIGURE 1.

- (2) Interpreta las correlaciones existentes entre las variables y las componentes principales.
- (3) Calcula la parte de la varianza total explicada por las dos primeras componentes principales.

Ejercicio 8. En el archivo **tortugas.txt**² se encuentran tres variables, largo, ancho y alto, de los caparazones de 48 tortugas pintadas, 24 hembras y 24 machos. Toma logaritmos de las tres variables. Estime el vector medio y la matriz de covarianza de las tortugas macho y de las tortugas hembra por separado. Encuentre los valores propios y los vectores propios de cada matriz de covarianza estimada y realice un PCA de cada conjunto de datos. Encuentra una expresión para el volumen del caparazón de una tortuga para

²"<https://web.stanford.edu/class/bios221/data/PaintedTurtles.txt>"

machos y hembras. (Sugerencia: use el hecho de que las variables son logaritmos de las medidas originales). Compare los volúmenes de los caparazones de machos y hembras.

Ejercicio 1. Demuestra que el segundo problema de optimización para $\text{Var}(\xi_2)$ tiene solución cuando el multiplicador de lagrange λ_2 es eigenvalor de la matriz de covarianza de $X = (X_1, \dots, X_n)$.

El segundo problema de optimización asociado al PCA.

$$\text{Maximizar} \quad \text{Var}(\xi_2) = b_2 \sum_{x,x} b_2^{tr}$$

$$\text{s.a} \quad b_2 \cdot b_2^{tr} = 1$$

1er. comp. priro → $b_1 \cdot b_2^{tr} = 0$

definimos,

$$S(b_2, \lambda_1, \lambda_2) = b_2 \sum_{x,x} b_2^{tr} - \lambda_1 (b_2 \cdot b_2^{tr} - 1) - \lambda_2 (b_1 \cdot b_2^{tr})$$

obteniendo las derivadas parciales

$$\frac{\partial S(b_2, \lambda_1, \lambda_2)}{\partial b_2} = 2 \left(\sum_{x,x} - \lambda_1 \right) b_2 - \lambda_2 b_1 = 0$$

$$\frac{\partial S(b_2, \lambda_1, \lambda_2)}{\partial \lambda_1} = b_2 \cdot b_2^{tr} - 1 = 0$$

$$\frac{\partial S(b_2, \lambda_1, \lambda_2)}{\partial \lambda_2} = b_1 \cdot b_2^{tr} = 0$$

de la primera ecuación, tenemos que

$$\left(\sum_{x,x} - \lambda_1 I_n \right) b_2 = \lambda_2 b_1$$

pero sabemos que b_1 y b_2 son ortogonales, por lo que $\lambda_2 = 0$, obtenemos

$$\left(\sum_{x,x} - \lambda_1 I_n \right) b_2 = 0$$

con $b_2 \neq 0$

esto significa que λ_1 es eigenvalor de la matriz de covarianza.

$$\sum_{x,x} b_2 = \lambda_1 b_2$$

Ejercicio 2. Considera la matriz

$$X = \begin{pmatrix} -1 & 0 & 1 \\ 0 & -1 & 1 \end{pmatrix}$$

Realiza un PCA realizando la descomposición espectral de la matriz de covarianza de X . Calcula la varianza total.

```
'> [6] import numpy as np
      X=np.array([[ -1, 0, 1],
                  [ 0, -1, 1]])
      X
      array([[ -1,  0,  1],
             [ 0, -1,  1]])

'> [7] # Matriz de covarianza.
      S = np.cov(X, rowvar = False)
      S
      array([[ 0.5, -0.5,  0. ],
             [-0.5,  0.5,  0. ],
             [ 0. ,  0. ,  0. ]])

'> [9] # Vamos a calcular eigenvalores y eigenvectores
      eigen_val, eigen_vec = np.linalg.eigh(S)
      eigen_val
      array([0., 0., 1.])

'> [11] eigen_val[::-1]
      eigen_vec = eigen_vec[:,np.argsort(eigen_val)[::-1]]
      eigen_vec
      array([[ -0.70710678,  0.          , -0.70710678],
             [-0.70710678,  0.          ,  0.70710678],
             [-0.          ,  1.          ,  0.          ]])

      ⏎ n_components = 2
      eigenvector_2 = eigen_vec[:,0:n_components]
      eigenvector_2
      array([[ -0.70710678,  0.          ],
             [-0.70710678,  0.          ],
             [-0.          ,  1.          ]])

[14] # Descomposición espectral
      #estos tienen la info de las componentes principales
      Y_red = np.dot(eigenvector_2.transpose(),X.transpose()).transpose()
      Y_red
      array([[ 0.70710678,  1.          ],
             [ 0.70710678,  1.          ]])

[15] eigen_val_total = sum(eigen_val) #es la traza de la matriz
      varianza_explicada = [(i/eigen_val_total)*100 for i in eigen_val ]
      varianza_explicada_acumulada = np.cumsum(varianza_explicada)
      print("Varianza explicada acumulada: {}".format(varianza_explicada_acumulada))

      Varianza explicada acumulada: [ 0.  0. 100.]
```

Ejercicio 3. Considera los datos `wine.data`¹. Realiza un PCA usando el código visto en clase (sin sklearn).

Recuerda preprocessar tus datos. Comenta tus resultados.

[2] import pandas as pd
import numpy as np

Datos
url = "wine.data"
data = pd.read_csv(url)
data

	1	14.23	1.71	2.43	15.6	127	2.8	3.06	.28	2.29	5.64	1.04	3.92	1065
0	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
1	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
2	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
3	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735
4	1	14.20	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.75	1.05	2.85	1450
...
172	3	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	1.06	7.70	0.64	1.74	740
173	3	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	1.41	7.30	0.70	1.56	750
174	3	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	1.35	10.20	0.59	1.56	835
175	3	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	1.46	9.30	0.60	1.62	840
176	3	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	1.35	9.20	0.61	1.60	560

```
np.mean(data)  
#Media por columnas  
np.mean(data, axis = 0)  
#media por renglones  
np.mean(data, axis = 1)  
#Centramos  
dta_media = data - np.mean(data, axis = 0)  
#Matriz covarianza  
S = np.cov(dta_media, rowvar = False)  
print(S)  
#Eigenvalores y eigenvectores  
eigen_val, eigen_vec = np.linalg.eigh(S)  
eigen_val[::-1]  
#Eigenvalores decreciente  
eigen_val = eigen_val[np.argsort(eigen_val)[::-1]]  
eigen_val  
  
eigen_vec = eigen_vec[:,np.argsort(eigen_val)[::-1]]  
eigen_vec  
n_components = 2  
eigenvector_2 = eigen_vec[:,0:n_components]  
  
#descomposicion espectral  
dta_red = np.dot(eigenvector_2.transpose(),dta_media.transpose()).transpose()  
dta_red  
  
# Info total  
eigen_val_total = sum(eigen_val)  
var_expl = [(i/ eigen_val_total )*100 for i in eigen_val ]  
var_expl = np.round(var_expl, 2)  
var_expl_ac = np.cumsum(var_expl)  
print("Varianza explicada: {}".format(var_expl))  
print("Varianza explicada acumulada: {}".format(var_expl_ac))
```

[[5.99062661e-01	-2.01098164e-01	3.77834489e-01	-1.02751027e-02
1.32709938e+00	-2.18252440e+00	-3.48110234e-01	-6.54292822e-01	
4.70066127e-02	-2.18923344e-01	4.83170577e-01	-1.09546418e-01	
-4.29187853e-01	-1.53841134e+02]			
[-	6.0198164e-01	6.54171097e-01	9.04975764e-02	4.69369158e-02
-8.18511460e-01	2.96623909e+00	1.44151717e-01	1.85883863e-01	
-1.52687821e-02	5.89675173e-02	1.03003738e+00	-1.39689715e-02	
3.27443695e-02	1.63267657e+02]			
[3.77834489e-01	9.04975764e-02	1.25286476e+00	5.07899043e-02
1.06850761e+00	-7.78171867e-01	-2.33862243e-01	-4.57547336e-01	
4.06718541e-02	-1.39446938e-01	6.50584636e-01	-1.43844547e-01	
-2.89426785e-01	-6.67941936e+01]			
[-	0.2751027e-02	4.69369158e-02	5.07899043e-02	7.56692476e-02
4.09929122e-01	1.11942925e+00	2.20882768e-02	3.13400199e-02	
6.44249058e-03	1.27059900e-03	1.65378782e-01	-4.73870185e-03	
2.91589625e-04	1.93141211e+01]			
[1.32709938e+00	-8.18511460e-01	1.06850761e+00	4.09929122e-01
1.11293702e+01	-3.39069723e+00	-6.63726021e-01	-1.15580306e+00	
1.49454834e-01	-3.63760497e-01	1.58798825e-01	-2.08469029e-01	
-6.30845917e-01	-4.58908455e+02]			
[-	2.18252440e+00	2.96623909e+00	-7.78171867e-01	1.11942925e+00
-3.39069723e+00	2.00902799e+02	1.84872143e+00	2.64841808e+00	
-4.45402863e-01	1.83492777e+00	6.56750329e+00	1.69021379e-01	
4.69337763e-01	1.72966484e+03]			
[-	3.48110234e-01	1.44175177e-01	-2.33862243e-01	2.20882768e-02
-6.63726021e-01	1.84872143e+00	3.92458500e-01	5.49567736e-01	
-3.50081054e-02	2.18602956e-01	-8.21307955e-02	6.21532203e-02	
3.09614105e-01	9.78111454e+01]			
[-	6.54292822e-01	1.85883863e-01	-4.57547336e-01	3.13400199e-02
-1.15580306e+00	2.64841808e+00	5.40567736e-01	9.97317033e-01	
-6.6764866e-02	3.71150347e-01	-4.04863795e-01	1.24300797e-01	
5.53728868e-01	1.54457224e+02]			
[4.70066127e-02	-1.52687821e-02	4.06718541e-02	6.44249058e-03
1.49454834e-01	-4.45402863e-01	-3.50081054e-02	-6.67648466e-02	
1.55383539e-02	-2.58809611e-02	4.06206304e-02	-7.47501733e-03	
-4.41100026e-02	-1.21241442e+01]			
[-	2.18923344e-01	5.89675173e-02	-1.39446938e-01	1.27059900e-03
-3.63760497e-01	1.83492777e+00	2.18602956e-01	3.71150347e-01	
-2.58809611e-02	3.26663367e-01	-3.60187831e-02	3.85544935e-02	
2.06905220e-01	5.86219992e+01]			
[4.83170577e-01	1.03003738e+00	6.50584636e-01	1.65378782e-01
1.58798825e-01	6.56750329e+00	-8.21307955e-02	-4.04863795e-01	
4.06206304e-02	-3.60187831e-02	5.40305118e+00	-2.78351336e-01	
-7.14173004e-01	2.31020958e+02]			
[-	1.09546418e-01	-1.39689715e-02	-1.43844547e-01	-4.73870185e-03
-2.08469029e-01	1.69021379e-01	6.21532203e-02	1.24300797e-01	
-7.47501733e-03	3.85544935e-02	-2.78351336e-01	5.25028690e-02	
9.16705277e-02	1.69467687e+01]			
[-	4.29187853e-01	3.27443695e-02	-2.89426785e-01	2.91589625e-04
-6.30845917e-01	4.69337763e-01	3.09014105e-01	5.53728868e-01	
-4.41100026e-02	2.06905220e-01	-7.14173004e-01	9.16705277e-02	
4.97170095e-01	6.79468012e+01]			
[-	1.53841134e+02	1.63267657e+02	-6.67941936e+01	1.93141211e+01
-4.58908455e+02	1.72966484e+03	9.78111454e+01	1.54457224e+02	
-1.21241442e+01	5.86219992e+01	2.31020958e+02	1.69467687e+01	
6.79468012e+01	9.91519623e+04]]			

```
Varianza explicada: [9.981e+01 1.700e-01 1.000e-02 1.000e-02 0.000e+00 0.000e+00 0.000e+00  
0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00]  
Varianza explicada acumulada: [ 99.81 99.98 99.99 100. 100. 100. 100. 100.  
100. 100. 100. ]  
/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py:3502: FutureWarning: In a future version, DataFrame.mean(axis=None) will  
return mean(axis=axis, dtype=dtype, out=out, **kwargs)
```

#Analisis de resultados

```
#El primer componente principal explica la gran mayoría de la varianza en los datos  
#aproximadamente el 99.81%). Esto sugiere que una gran cantidad de información en los  
#datos originales se puede capturar utilizando solo este componente principal.'
```

Ejercicio 4. Genera una muestra aleatoria de tamaño 100 a partir de una distribución gaussiana 3-dimensional en dónde una de las variables tiene más alta varianza que las otras. Realiza un PCA. En cada caso, encuentra los eigenvalores y eigenvectores y dibuja la gráfica scree.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

# Generar una muestra aleatoria de tamaño 100 de una distribución gaussiana 3-dimensional
# con una varianza más alta para una de las variables.
np.random.seed(0)
mean = [0, 0, 0] # Media de las tres dimensiones
cov = [[3, 0, 0], # Matriz de covarianza con una varianza más alta en una variable
       [0, 1, 0],
       [0, 0, 1]]
X = np.random.multivariate_normal(mean, cov, 100)

# Realizar PCA
pca = PCA(n_components=3)
pca.fit(X)

# Eigenvalores y eigenvectores
eigenvalues = pca.explained_variance_
eigenvectors = pca.components_

# Gráfica scree
plt.figure(figsize=(8, 5))
plt.plot(range(1, 4), eigenvalues, 'o-', linewidth=2, color='blue')
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Eigenvalue')
plt.grid(True)
plt.xticks(range(1, 4))
plt.tight_layout()

eigenvalues, eigenvectors, plt.show()
```



Ejercicio 5. Se requiere efectuar un PCA de

$$X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

- (1) Verifica que obtener los eigenvalores de la matriz

$$A = \begin{pmatrix} 3 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 3 \end{pmatrix}$$

implica obtener los eigenvalores de X .

Podemos notar que en la entrada $a_{2,2}$ tenemos la suma de la traza de X , en a_{12}, a_{21}, a_{23} tenemos la suma que da 2, en a_{11}, a_{33} tenemos la suma de 3 y en a_{13} y a_{31} tenemos la suma de 1.

- (2) Determina el segundo eje principal del PCA de X .

```
import numpy as np

X=np.array([[1,0,0,0],[1,1,0,0],[1,1,1,0],[1,1,1,1]])

# Matriz de covarianza.
S = np.cov(X, rowvar = False)

# Vamos a calcular eigenvalores y eigenvectores
eigen_val, eigen_vec = np.linalg.eigh(S)
eigen_val[::-1]
eigen_vec = eigen_vec[:,np.argsort(eigen_val)[::-1]]
n_components = 2
eigenvector_2 = eigen_vec[:,0:n_components]
```

```

# Descomposición espectral
#estas tienen la info de las componentes principales
Y_red = np.dot(eigenvector_2.transpose(), X.transpose()).transpose()

eigen_val_total = sum(eigen_val) #es la traza de la matriz
varianza_explicada = [(i / eigen_val_total) * 100 for i in eigen_val]
varianza_explicada_acumulada = np.cumsum(varianza_explicada)

X_media = X - np.mean(X, axis=0)

cov_mat = np.cov(X_media, rowvar=False)

eigen_val, eigen_vec = np.linalg.eigh(cov_mat)

sorted_index = np.argsort(eigen_val)[::-1]
sorted_eigenval = eigen_val[sorted_index]
sorted_eigenvec = eigen_vec[:, sorted_index]

eigenvector_ = sorted_eigenvec[:, 0:n_components]

X_red = np.dot(eigenvector_.transpose(), X_media.transpose()).transpose()

```

```

[[ 0.85355339  0.35355339]
 [ 0.35355339 -0.35355339]
 [-0.35355339 -0.35355339]
 [-0.85355339  0.35355339]]

```

- (3) Provee las coordenadas sobre los segundos ejes por renglones y columnas del PCA de X .

```

# Para conseguir las coordenadas
proyeccion = np.dot(X, sorted_eigenvec[:, :2])

print("Las coordenadas son:", proyeccion)

```

```

Las coordenadas son: [[ 0.00000000e+00  0.00000000e+00]
 [-5.00000000e-01 -7.07106781e-01]
 [-1.20710678e+00 -7.07106781e-01]
 [-1.70710678e+00  2.22044605e-16]]

```

Ejercicio 6. En 28 años, se han observado 4 estados metereológicos:

- ξ_1 = precipitación en julio (en mm),
- ξ_2 = temperatura media en julio (en C),
- ξ_3 = velocidad media del viento en julio (en km/h),
- ξ_4 = precipitación en septiembre (en mm).

A partir de tales dator, se tiene la matriz de covarianzas:

$$S = \begin{pmatrix} 140.017 & 107.881 & 139.068 & 109.095 \\ & 106.038 & 110.0439 & 82.627 \\ & & 168.752 & 125.136 \\ & & & 108.960 \end{pmatrix}$$

y las correlaciones empíricas

$$\begin{pmatrix} 0.969 & -0.102 & 0.194 & 0.116 \\ 0.907 & -0.392 & -0.106 & -0.111 \\ 0.971 & 0.156 & -0.157 & 0.092 \\ 0.943 & 0.252 & 0.092 & -0.196 \end{pmatrix}$$

- (1) Calcula las varianzas empíricas de las componentes principales.
- (2) Calcular la parte de la varianza ξ_1 explicada por las dos últimas componentes principales, y la parte de la varianza de ξ_2 que explicada por las dos primeras componentes principales.
- (3) Realiza la proyección de variables sobre los dos primeros ejes principales y comenta tus resultados.

```
import numpy as np

covm = np.array([[140.017, 107.881, 139.068, 109.095], [0, 106.038, 110.0439, 82.627], [0, 0, 168.752, 125.136], [0, 0, 0, 108.960]])

corm = np.array([[0.969, -0.102, 0.194, 0.116], [0.907, -0.392, -0.106, -0.111], [0.971, 0.156, -0.157, 0.092], [0.943, 0.252, 0.092, -0.196]])

eigen_val, eigen_vec = np.linalg.eigh(covm)
eigen_val[::-1]
eigen_vec = eigen_vec[:,np.argsort(eigen_val)[::-1]]
n_components = 2
eigenvector_2 = eigen_vec[:,0:n_components]

X_media = corm - np.mean(corm, axis = 0)
eigen_val = eigen_val[np.argsort(eigen_val)[::-1]]

varemp = np.diag(covm)
print("Las varianzas empíricas son:", varemp)
print()

eigen_vec = eigen_vec[:,np.argsort(eigen_val)[::-1]]
n_components = 2
eigenvector_2 = eigen_vec[:,0:n_components]

ve1 = eigen_val[2:1].sum() / eigen_val.sum()
ve2 = eigen_val[:2].sum() / eigen_val.sum()

print("La parte de la varianza  $\xi_1$  explicada por las dos ultimas componentes principales ", ve1)
print()
print("La parte de la varianza de  $\xi_2$  que explicada por las dos primeras componentes principales ", ve2)
print()

eigen_val, eigen_vec = np.linalg.eigh(covm)
```

```
sorted_index = np.argsort(eigen_val)[::-1]
sorted_eigenval = eigen_val[sorted_index]
sorted_eigenvec = eigen_vec[:,sorted_index]

eigenvector_ = sorted_eigenvec[:,0:n_components]

X_red = np.dot(eigenvector_.transpose(), X_media.transpose()).transpose()
print(X_red)
print()
# Para conseguir la coordenadas
proyeccion = np.dot(corm, sorted_eigenvec[:, :2])

print("Las coordenadas de la proyección son:", proyeccion)
```

Las varianzas empíricas son: [140.017 106.038 168.752 108.96]

La parte de la varianza ξ_1 explicada por las dos ultimas componentes principales 0.4104840511143314

La parte de la varianza de ξ_2 que explicada por las dos primeras componentes principales 0.5895159488856687

```
[[ 0.18825  0.0215 ]
 [-0.11175 -0.0405 ]
 [-0.16275  0.0235 ]
 [ 0.08625 -0.0045 ]]
```

Las coordenadas de la proyección son: [[0.194 0.969]

```
[-0.106  0.907]
[-0.157  0.971]
[ 0.092  0.943]]
```

Ejercicio 7. Sea X un vector aleatorio en dimensión 4 de media μ y matriz de covarianza $\Sigma = (\sigma_{ij})$. Supongamos que $\sigma_{ii} = 1$ para todo i .

- (1) Sea $0 < \rho < 1$. En la **Figura 1** se muestran dos gráficas, una de ellas presenta la proyección de variables sobre los dos primeros ejes principales. ¿Cuál es? Justifica tu respuesta.

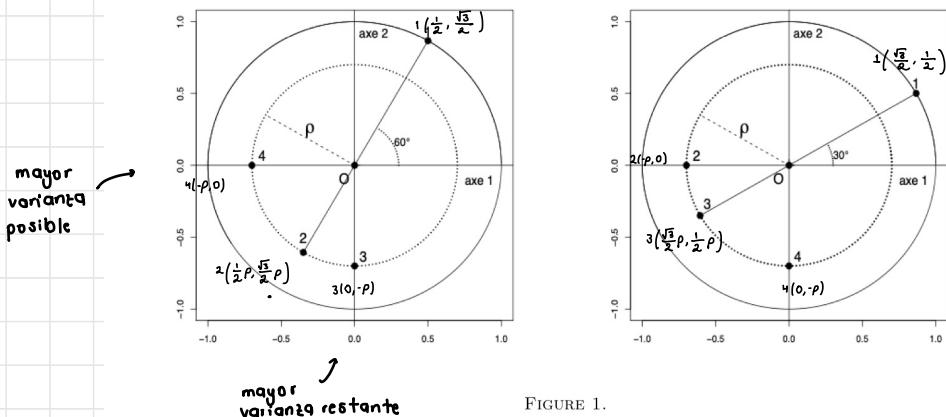


FIGURE 1.

- (2) Interpreta las correlaciones existentes entre las variables y las componentes principales.
 (3) Calcula la parte de la varianza total explicada por las dos primeras componentes principales.

1) Calculando los eigenvalores de la gráfica de lo derecho

$$\lambda_1 = \sum_{i=1}^4 r_{ij}^2 s_{ii} = \sum_{i=1}^4 r_{ij}^2 = \frac{\sqrt{3}}{2}^2 + \left(\frac{\sqrt{3}}{2}\right)^2 \rho^2 + \rho^2 = \frac{3}{4} + \frac{3}{4} \rho^2 + \rho^2$$

$$\lambda_2 = \sum_{j=1}^4 r_{ij}^2 s_{ii} = \sum_{j=1}^4 r_{ij}^2 = \frac{1}{4} + \frac{1}{4} \rho^2 + \rho^2$$

$\lambda_1 > \lambda_2$, por lo tanto pertenece a la cámara de Weyl

$$\lambda_3 = \sum_{i=1}^4 r_{ij}^2 s_{ii} = \sum_{j=1}^4 r_{ij}^2 = \frac{1}{4} + \frac{1}{4} \rho^2 + \rho^2$$

$$\lambda_4 = \sum_{i=1}^4 r_{ij}^2 s_{ii} = \sum_{i=1}^4 r_{ij}^2 = \frac{\sqrt{3}}{2}^2 + \left(\frac{\sqrt{3}}{2}\right)^2 \rho^2 + \rho^2 = \frac{3}{4} + \frac{3}{4} \rho^2 + \rho^2$$

$$\text{Tr} \left(\sum_{i=1}^4 \lambda_i \right) = \frac{3}{4} + \frac{3}{4} \rho^2 + \rho^2 + \frac{1}{4} + \frac{1}{4} \rho^2 + \rho^2 + \frac{1}{4} + \frac{1}{4} \rho^2 + \rho^2 + \frac{3}{4} + \frac{3}{4} \rho^2 + \rho^2 = 2 + 6\rho^2$$

2)

Podemos observar que la variable 1 respecto al primer componente principal esté altamente correlacionado, al igual que la variable 2, mientras la variable 4, esté altamente correlacionado con la segunda componente principal.

3)

$$\frac{\lambda_1 + \lambda_2}{\text{tr}(\sum_{i=1}^n \lambda_i)} = \frac{1 + 3\rho^2}{2 + 6\rho^2}$$

Ejercicio 8. En el archivo **tortugas.txt**² se encuentran tres variables, largo, ancho y alto, de los caparazones de 48 tortugas pintadas, 24 hembras y 24 machos. Toma logaritmos de las tres variables. Estime el vector medio y la matriz de covarianza de las tortugas macho y de las tortugas hembra por separado. Encuentre los valores propios y los vectores propios de cada matriz de covarianza estimada y realice un PCA de cada conjunto de datos. Encuentra una expresión para el volumen del caparazón de una tortuga para machos y hembras. (Sugerencia: use el hecho de que las variables son logaritmos de las medidas originales). Compare los volúmenes de los caparazones de machos y hembras.

```
1s ➜ import pandas as pd
import numpy as np

# Datos

url = "https://web.stanford.edu/class/bios221/data/PaintedTurtles.txt"
datos = pd.read_csv(url,delimiter="\t")

0s ➜ #Aplicamos logaritmo

ln_len=datos['length'].apply(np.log)
ln_w=datos['width'].apply(np.log)
ln_h=datos['height'].apply(np.log)

#Tabla con datos

datos_ln=pd.concat([ datos['sex'],ln_len , ln_w, ln_h] , axis = 1)
```

[36] datos_ln

	sex	length	width	height
0	f	4.584967	4.394449	3.637586
1	f	4.634729	4.430817	3.637586
2	f	4.634729	4.454347	3.737670
3	f	4.653960	4.454347	3.688879
4	f	4.691348	4.477337	3.784190
5	f	4.812184	4.521789	3.912023
6	f	4.812184	4.553877	3.828641
7	f	4.890349	4.595120	3.931826
8	f	4.890349	4.624973	3.931826
9	f	4.890349	4.624973	3.931826
10	f	4.897840	4.605170	3.871201
11	f	4.912655	4.624973	3.891820
12	f	4.919981	4.584967	3.931826
13	f	4.927254	4.595120	3.931826
14	f	4.948760	4.653960	3.970292
15	f	4.990433	4.682131	4.043051

```

0s ⏪ #Separamos los datos en masculinos y femeninos
      datos_fem= datos_ln.loc[datos_ln['sex '] == 'f ']
      datos_mas= datos_ln.loc[datos_ln['sex '] == 'm ']

      datos_f1=pd.DataFrame(datos_fem [[ 'length ','width ','height'])]
      datos_m1=pd.DataFrame(datos_mas [[ 'length ','width ','height'])]

0s ⏪ datos_fem = datos_f1.to_numpy()
      datos_mas = datos_m1.to_numpy()

0s ⏪ #Vector de medias por tabla
      media_fem = datos_fem-np.mean(datos_fem, axis = 0)
      media_mas = datos_mas-np.mean(datos_mas, axis = 0)

      #Obtenemos matrices de covarianzas

      S_fem = np.cov(media_fem, rowvar=False)
      S_mas = np.cov(media_mas, rowvar=False)

0s ⏪ #Eigenvalores y eigenvectores
      eigen_val_fem, eigen_vec_fem = np.linalg.eigh(S_fem)
      eigen_val_mas, eigen_vec_mas = np.linalg.eigh(S_mas)

0s ⏪ #Cámera de Weyl
      eigen_val_fem_weyl=eigen_val_fem[::-1]
      eigen_val_mas_weyl=eigen_val_mas[::-1]

0s ⏪ eigen_vec_fem_weyl=eigen_vec_fem[:,np.argsort(eigen_val_fem)[::-1]]
      eigen_vec_mas_weyl=eigen_vec_mas[:,np.argsort(eigen_val_mas)[::-1]]

0s ⏪ #Ocupamos 2 componentes
      n_components = 2
      eigenvector_fem2 = eigen_vec_fem_weyl[:,0:n_components]
      eigenvector_mas2 = eigen_vec_mas_weyl[:,0:n_components]

0s ⏪ #Hacemos descomposición espectral
      datos_fem_red = np.dot(eigenvector_fem2.transpose(),media_fem.transpose()).transpose()
      datos_mas_red = np.dot(eigenvector_mas2.transpose(),media_mas.transpose()).transpose()

```

Tortugas hembra :

```

0s ⏪ f_red = pd.DataFrame(datos_fem_red , columns = [ 'PC1','PC2'])
      f_red

      PC1      PC2
      0  0.491814  0.001752
      1  0.443245 -0.036014
      2  0.370284  0.033017
      3  0.388332 -0.014156
      4  0.295304  0.034322
      5  0.119952  0.061507
      6  0.155715 -0.017488
      7  0.023633  0.011044
      8  0.009182 -0.001364
      9  0.009182 -0.001364
      10 0.051403 -0.044282
      11 0.019913 -0.043019
      12 0.010109  0.001775
      13 0.000669 -0.005755
      14 -0.064861 -0.009709

```

tortugas macho :

```
0s ⏎ m_red = pd.DataFrame(datos_mas_red , columns = ['PC1','PC2'])  
m_red
```

	PC1	PC2
0	0.268473	0.061069
1	0.263345	-0.015724
2	0.236046	-0.034121
3	0.119923	0.014125
4	0.120728	-0.014957
5	0.152593	-0.008908
6	0.106039	0.016571
7	0.093027	0.013533
8	0.106371	-0.001245
9	0.006575	-0.016744
10	0.006268	-0.011450
11	0.011980	0.000801
12	-0.060887	0.028047
13	-0.041863	-0.010877

Calculamos volumen de los caparazones con : $\text{lnlargo} + \text{lanchos} + \text{lnalto}$

```
0s ⏎ volumen_fem = pd.DataFrame(datos_f1.sum(numeric_only=True, axis=1) , columns = ['Volumen Hembras'])  
volumen_fem
```

	Volumen Hembras
0	12.617003
1	12.703132
2	12.826746
3	12.797187
4	12.952874
5	13.245996
6	13.194703
7	13.417295
8	13.447148
9	13.447148
10	13.374211
11	13.429448
12	13.436774
13	13.454199
14	13.573012
15	13.715615

```
volumen_mas = pd.DataFrame(datos_m1.sum(numeric_only=True, axis=1) ,columns = ['Volumen Machos'])
```



Volumen Machos	
24	12.447582
25	12.455352
26	12.501723
27	12.709499
28	12.705210
29	12.640096
30	12.726793
31	12.745841
32	12.717134
33	12.896015
34	12.893604
35	12.879425
36	13.014600
37	12.975556
38	12.986606