

TAREA I

15 de febrero, 2024

Licenciatura en Actuaría

Estadística Multivariada

Cañas Taboada Andrea

Maya Crotos Angel

Rodríguez Herrera Dante

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

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

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.

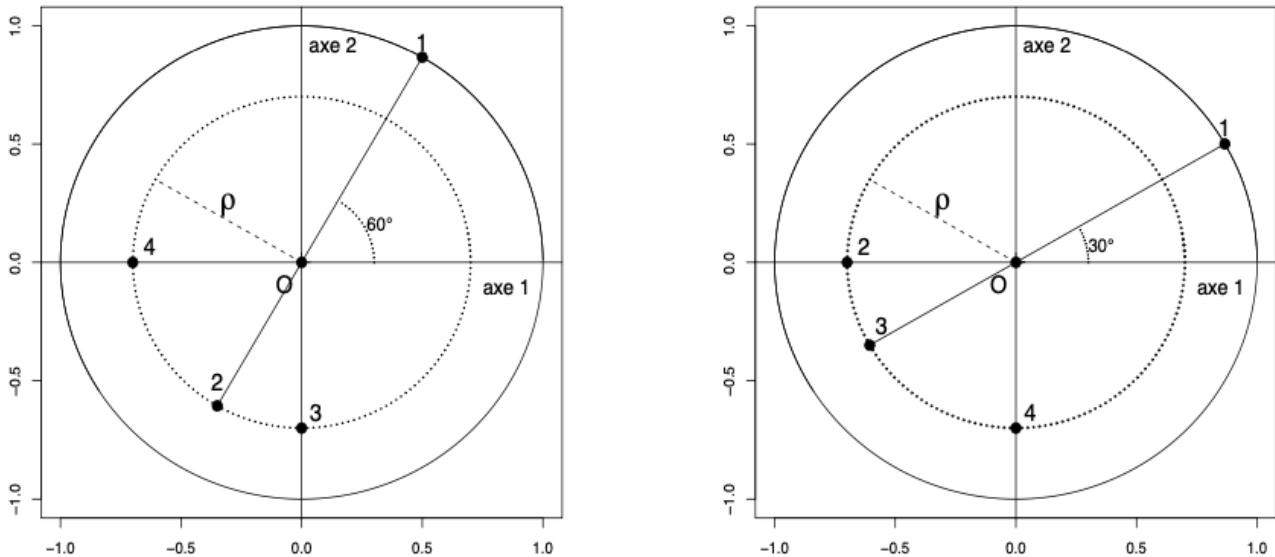


FIGURE 1.

- (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 $X = (X_1, \dots, X_n)$.

Demostración . . .

Para demostrar que el segundo problema de optimización para $\text{Var}(\xi_2)$ tiene solución cuando el multiplicador de Lagrange λ_2 es un eigenvalor de la matriz de covarianza de

$$X = (X_1, \dots, X_n)$$

Tenemos que

$$\begin{array}{ll} \max & \text{Var}(\xi_2) \\ \text{subject to} & \lambda_2 = u' \Sigma u \quad \dots \text{u: vector de coeficientes} \end{array}$$

dado que λ_2 es un eigenvalor de la matriz de covarianza Σ_{xx} , existe un vector propio (no nulo) "u" tal que

$$\sum u = \lambda_2 u$$

entonces

$$\begin{array}{ll} \max & \text{Var}(\xi_2) \\ \text{subject to} & u' \Sigma u = \lambda_2 \end{array}$$

Usamos el método de multiplicador de Lagrange para resolver el problema. La función objetivo es

$$\begin{array}{ll} \max & \text{Var}(\xi_2) = u' \Sigma u \\ \text{subject to} & u' \Sigma u - \lambda_2 = 0 \end{array}$$

Al calcular el gradiente de la función objetivo y de la restricción obtenemos:

$$\begin{aligned} \nabla(u' \Sigma u) &= 2 \sum u \\ \nabla(u' \Sigma u - \lambda_2) &= 2 \sum u - 0 \\ &= 2 \sum u \end{aligned}$$

Igualando el gradiente de la función objetivo al gradiente de la restricción multiplicado por el multiplicador de Lagrange, obtenemos:

$$\begin{aligned} 2 \sum u &= 2 \lambda_2 u \\ \sum u &= \lambda_2 u \quad \dots \text{dividimos entre 2} \end{aligned}$$

Entonces, tenemos que u es un vector propio de la matriz correspondiente al eigenvalor λ_2 . Por lo tanto $\max \text{Var}(\xi_2)$ está sujeta a que λ_2 es un eigenvalor de la matriz de covarianza de X \therefore el problema tiene solución.

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.

```
[ ] #Importamos las librerías necesarias
import numpy as np

#Creamos el arreglo
x=np.array([[-1,0,1],[0,-1,1]])

print(x)
[[[-1  0  1]
 [ 0 -1  1]]]

[ ] x.ndim #indica la dimensión de la matriz
2

[ ] x.shape #[filas, columnas]
(2, 3)

[ ] np.mean(x)
0.0

[ ] np.mean(x, axis=0) #saca la media del vector, i.e valores aleatorios
array([-0.5, -0.5,  1.])
```

```
[ ] np.mean(x, axis=1)
array([0., 0.])

▶ #Media de los vectores
x_media = x - np.mean(x, axis = 0)
x_media
array([[-0.5,  0.5,  0. ],
 [ 0.5, -0.5,  0. ]])

[ ] #Sacamos la covarianza
S = np.cov(x_media, rowvar = False)
print(S)
[[ 0.5 -0.5  0. ]
 [-0.5  0.5  0. ]
 [ 0.   0.   0. ]]

[ ] eigen_val, eigen_vec = np.linalg.eigh(S)
eigen_val
```

```
▶ eigen_val[::-1]
array([1., 0., 0.])

▶ eigen_val[::-1]
array([1., 0., 0.])

[ ] eigen_val = eigen_val[np.argsort(eigen_val)[::-1]]
eigen_val
array([1., 0., 0.])

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

```
[ ] n_components = 2
eigenvector_2 = eigen_vec[:,0:n_components]

[ ] x_red = np.dot(eigenvector_2.transpose(),x_media.transpose()).transpose()

x_red
array([[ 0.        ,  0.70710678],
 [ 0.        , -0.70710678]])
```

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.

```
import numpy as np
import pandas as pd
import urllib.request

x = np.random.randint(10,50,100)
x

array([46, 16, 33, 42, 44, 27, 15, 32, 28, 36, 31, 39, 38, 25, 35, 35, 42,
       39, 20, 44, 17, 38, 15, 28, 17, 18, 24, 29, 42, 27, 45, 49, 14, 18,
       24, 15, 18, 36, 15, 24, 25, 17, 35, 33, 40, 40, 14, 35, 37, 30, 48,
       11, 37, 48, 13, 14, 21, 44, 34, 19, 21, 45, 35, 24, 20, 17, 11, 11,
       36, 39, 38, 49, 31, 11, 31, 32, 45, 38, 32, 25, 12, 39, 49, 15, 15,
       49, 25, 19, 45, 25, 39, 22, 14, 29, 10, 42, 35, 48, 18, 33])

x.ndim

1

[3] x.shape

(100,)

[4] np.mean(x)

29.49

# URL del conjunto de datos
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data'

# Descargar el archivo
urllib.request.urlretrieve(url, 'wine.data')

# Leer el archivo descargado en un DataFrame de pandas
Y= pd.read_csv('wine.data', header=None)

print(Y)

   0    1    2    3    4    5    6    7    8    9    10   11 \
0  14.23  1.71  2.43  15.6  127  2.80  3.06  0.28  2.29  5.64  1.04
1  13.20  1.78  2.14  11.2  100  2.65  2.76  0.26  1.28  4.38  1.05
2  13.16  2.36  2.67  18.6  101  2.80  3.24  0.30  2.81  5.68  1.03
3  14.37  1.95  2.50  16.8  113  3.85  3.49  0.24  2.18  7.80  0.86
4  13.24  2.59  2.87  21.0  118  2.80  2.69  0.39  1.82  4.32  1.04
..  ...
173 3.13  7.1  5.65  2.45  20.5  95  1.68  0.61  0.52  1.06  7.70  0.64
174 3.13  4.0  10.65  2.48  23.0  102  1.80  0.75  0.43  1.41  7.30  0.70
175 3.13  2.27  4.28  2.26  20.0  120  1.59  0.69  0.43  1.35  10.20  0.59
176 3.13  1.17  2.59  2.37  20.0  120  1.65  0.68  0.53  1.46  9.30  0.60
177 3.14  1.13  4.10  2.74  24.5  96  2.05  0.76  0.56  1.35  9.20  0.61

   12    13
0  3.92  1065
1  3.40  1050
2  3.17  1185
3  3.45  1480
4  2.93  735

[5]      12    13
0    3.92  1065
1    3.40  1050
2    3.17  1185
3    3.45  1480
4    2.93  735
..  ...
173 1.74  740
174 1.56  750
175 1.56  835
176 1.62  840
177 1.60  560

[178 rows x 14 columns]

[5] np.mean(Y, axis = 0)

  0    1.938202
  1   13.000618
  2    2.336348
  3    2.366517
  4   19.494944
  5   99.741573
  6   2.295112
  7   2.029270
  8   0.361854
  9   1.590899
 10   5.058090
 11   0.957449
 12   2.611685
 13  746.893258
dtype: float64
```

```
[7] np.mean(Y, axis = 1)
0    89.000000
1    85.364286
2    95.915714
3   117.963571
4   64.977857
...
173   63.875714
174   65.124286
175   72.444286
176   72.640714
177   51.471429
Length: 178, dtype: float64
```

Y_media = Y - np.mean(Y, axis = 0)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	-0.938202	1.229382	-0.626348	0.063483	-3.894944	27.258427	0.504888	1.03073	-0.081854	0.699101	0.58191	0.082551	1.308315	318.106742
1	-0.938202	0.199382	-0.556348	-0.226517	-8.294944	0.258427	0.354888	0.73073	-0.101854	-0.310899	-0.67809	0.092551	0.788315	303.106742
2	-0.938202	0.159382	0.023652	0.303483	-0.894944	1.258427	0.504888	1.21073	-0.061854	1.219101	0.62191	0.072551	0.558315	438.106742
3	-0.938202	1.369382	-0.386348	0.133483	-2.694944	13.258427	1.554888	1.46073	-0.121854	0.589101	2.74191	-0.097449	0.838315	733.106742
4	-0.938202	0.239382	0.253652	0.503483	1.505056	18.258427	0.504888	0.66073	0.028146	0.229101	-0.73809	0.082551	0.318315	-11.893258
...

[8] ...
173 1.061798 0.709382 3.313652 0.083483 1.005056 -4.741573 -0.615112 -1.41927 0.158146 -0.530899 2.64191 -0.317449 -0.871685 -6.893258
174 1.061798 0.399382 1.573652 0.113483 3.505056 2.258427 -0.495112 -1.27927 0.068146 -0.180899 2.24191 -0.257449 -1.051685 3.106742
175 1.061798 0.269382 1.943652 -0.106517 0.505056 20.258427 -0.705112 -1.33927 0.068146 -0.240899 5.14191 -0.367449 -1.051685 88.106742
176 1.061798 0.169382 0.253652 0.003483 0.505056 20.258427 -0.645112 -1.34927 0.168146 -0.130899 4.24191 -0.357449 -0.991685 93.106742
177 1.061798 1.129382 1.763652 0.373483 5.005056 -3.741573 -0.245112 -1.26927 0.198146 -0.240899 4.14191 -0.347449 -1.011685 -186.893258

178 rows × 14 columns

Next steps: [Generate code with Y_media](#) [View recommended plots](#)

S = np.cov(Y_media, rowvar = False)

```
print(S)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13			
0	6.00679236e-01	-2.06515267e-01	3.79038596e-01	-1.05554498e-02	1.34036374e+00	-2.31549546e+00	-3.48834825e-01	-6.56099586e-01	4.71773630e-02	-2.21413064e-01	4.77338914e-01	-1.09367549e-01	-4.33737066e-01	-1.54667651e+02			
1	-2.06515267e-01	6.59062328e-01	8.56113090e-02	4.71151590e-02	-8.41092903e-01	3.13987812e+00	1.46887218e-01	1.92033222e-01	-8.57542595e-02	6.35175205e-02	1.02828254e+00	-1.33134432e-02	4.16978226e-02	1.64567185e+02			
2	3.79038596e-01	8.56113090e-02	1.24801540e+00	5.02770393e-02	1.07633171e+00	-8.70779534e-01	-2.34337723e-01	-4.58630366e-01	1.07633171e+00	-8.70779534e-01	1.93283248e+00	6.62052061e+00	1.80851266e-01	-2.92447483e-01	-6.75488666e+01		
3	6.00679236e-01	1.1293652e+00	2.21455913e-02	3.15347299e-02	6.35847140e-03	1.51557799e-03	1.64654327e-01	-4.68215451e-03	7.61835841e-04	1.93197391e+01	[...]	[...]	[...]	[...]	[...]		
4	-2.06515267e-01	-8.41092903e-01	-3.97476036e+00	-6.71149146e-01	-1.17208282e+00	1.50421856e-01	-3.77176220e-01	1.45024186e-01	-6.56234368e-01	-4.63355345e+02	[...]	[...]	[...]	[...]	[...]		
5	3.13987812e+00	-8.70779534e-01	1.12293658e+00	-3.97476036e+00	2.03989335e+02	1.91646988e+00	2.79308703e+00	-4.55563385e-01	1.93283248e+00	6.62052061e+00	1.80851266e-01	6.69308068e-01	1.76915870e+03	[...]	[...]	[...]	
6	-8.70779534e-01	-2.34337723e-01	-2.34337723e-01	-4.58630366e-01	3.11021278e-01	9.81710573e+01	-4.6834825e-01	1.46887218e-01	-6.71149146e-01	1.91646988e+00	3.91689535e-01	5.40470422e-01	-3.50451247e-02	2.19373345e-01	-7.99975192e-02	6.20388758e-02	
7	-6.56099586e-01	1.92033222e-01	-4.58630366e-01	3.15347299e-02	-1.17208281e+00	2.79308703e+00	5.40470422e-01	9.97718673e-01	-6.68669999e-02	3.73147553e-01	-3.99168626e-01	1.24081969e-01	5.58262255e-01	1.55447492e+02	[...]	[...]	[...]
8	4.71773630e-02	-1.57542595e-02	4.07333619e-02	6.35847140e-03	1.50421856e-01	-4.55563385e-01	-3.50451247e-02	-6.68669999e-02	1.54886339e-02	-2.60598680e-02	4.01205097e-02	-7.47117692e-03	-4.44692440e-02	-1.22035863e+01	[...]	[...]	[...]
9	-2.21455913e-02	6.35175205e-02	-1.41146982e-01	1.51557799e-03	-3.77176220e-01	1.93283248e+00	2.19373345e-01	3.73147553e-01	-2.60598680e-02	3.27594668e-01	-3.35039177e-02	3.86645655e-02	2.10932940e-01	5.9554338e+01	[...]	[...]	[...]
10	1.50421856e-01	6.62052061e+00	-7.99975192e-02	-3.99168626e-01	4.01205097e-02	5.37444938e+00	-2.76505801e-01	-7.05812576e-01	2.30767480e+02	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]
11	-1.09367549e-01	-1.33134432e-02	-1.43325638e-01	-4.68215451e-03	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]
12	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]
13	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]

0s completed at 6:01PM

```
✓ 0s [12] array([ 9.92020307e+04, 1.72536596e+02, 9.53119601e+00, 5.1008096e+00,
   1.28578826e+00, 8.68166571e-01, 2.87006972e-01, 1.55278963e-01,
   1.13733317e-01, 8.63784262e-02, 4.62026188e-02, 3.49273038e-02,
   2.07626732e-02, 8.09265892e-03])
```

```
✓ 0s ⏎ eigen_vec = eigen_vec[:,np.argsort(eigen_val)[::-1]]
```

```
eigen_vec
array([[ -4.71580781e-02, -9.59876464e-02, -3.66354280e-01,
   6.23529116e-01, -5.10993277e-01, -1.65827952e-01,
   1.90995227e-01, 1.78800746e-01, 1.68685378e-01,
   -2.19197410e-01, 1.47645845e-01, -1.00919962e-01,
   -2.77861762e-03, 1.55934284e-03],
[ -1.34219856e-02, 4.24032584e-03, -3.47767055e-02,
   8.90749178e-02, -5.75902207e-02, -1.04297167e-01,
   3.49028029e-01, -8.90116287e-01, -2.02080449e-01,
   4.89326076e-02, 1.36988601e-01, -1.83035443e-02,
   -1.20426744e-03, -1.65926191e-03],
[ 1.20548686e-02, -6.32356961e-02, 3.57362843e-02,
   -7.74361990e-03, 3.36100827e-02, -1.02255162e-02,
   -7.55207359e-02, 1.46858065e-01, -8.03839930e-01,
   -5.28930958e-01, 1.56739376e-01, -1.24559463e-01,
   -2.15953126e-03, 6.81019630e-04],
[ 1.62344580e-01, 8.64880935e-02, -8.91486857e-01,
   -3.53255783e-01, 1.07161337e-01, -6.88814741e-02,
   -1.50528090e-01, -5.03492983e-02, -4.16939040e-02,
   2.58456588e-02, -1.20471220e-02, -5.12928854e-02,
   -4.59399561e-03, -1.94905197e-04],
[ 6.48326979e-05, -2.69485827e-04, 5.93105853e-02,
   2.65861782e-03, 9.98035827e-03, 2.44128319e-03,
```

```
✓ 0s [14] n_components = 2
eigenvector_2 = eigen_vec[:,0:n_components]
```

```
✓ 0s ⏎ # Descomposición espectral
```

```
Y_red = np.dot(eigenvector_2.transpose(), Y_media.transpose()).transpose()
```

```
Y_red
```

```
array([[ -8.93553535e-02, 1.64297340e-02],
[ -1.78962892e-02, 4.48448723e-02],
[ 3.94205943e-02, 5.03578282e-02],
[-2.11603900e-02, 1.57084512e-01],
[ 1.14053955e-02, 5.65396027e-02],
[-8.25335663e-02, 1.78086584e-02],
[-6.841403227e-03, 6.94861455e-02],
[-4.48826615e-02, 1.00621722e-02],
[-1.64323448e-02, 2.98326256e-02],
[ 2.93237575e-02, 1.83961783e-02],
[ 3.47534238e-02, -1.65062235e-01],
[ 6.62723118e-02, -3.25793862e-02],
[ 5.77346183e-02, -3.90429173e-02],
[-1.08423533e-01, -1.19858358e-01],
[-1.19562539e-02, -1.46962162e-01],
[ 6.94012704e-02, -2.32391994e-01],
[-1.93959449e-02, 2.31094702e-02],
[-8.51455290e-02, -5.17581980e-02],
[-7.00116423e-02, -2.14358889e-01],
[ 1.04915243e-01, 5.98928362e-02],
```

```
✓ 0s ⏎ # Información total
```

```
eigen_val_total = sum(eigen_val)
varianza_explicada = [(i/ eigen_val_total )*100 for i in eigen_val ]
varianza_explicada = np.round(varianza_explicada, 2)
varianza_explicada_acumulada = np.cumsum(varianza_explicada)
print("Varianza explicada: {}".format(varianza_explicada))
print("Varianza explicada acumulada: {}".format(varianza_explicada_acumulada))
```

```
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]
Varianza explicada acumulada: [ 99.81 99.98 99.99 100. 100. 100. 100. 100. 100. 100. 100. 100. ]
```

```
✓ 0s [17] # Escribimos lo anterior como función
def PCA(X , num_componentes):
```

```
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]
```

```
[16] varianza_explicada_acumulada = np.cumsum(varianza_explicada)
print("Varianza explicada: {}".format(varianza_explicada))
print("Varianza explicada acumulada: {}".format(varianza_explicada_acumulada))

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.   100. ]
```

```
# Escribimos lo anterior como función
def PCA(X , num_componentes):

    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:num_componentes]

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

    return X_red
```

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.

```
[1] #Importamos las librerías necesarias
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sb

[2]
#Definimos la media
mean=[0,0,0]

#Definimos la matriz de covarianza
cov=[[1,0,0],[0,1,0],[0,0,1]]

#Generamos la muestra de tamaño 100
X=np.random.multivariate_normal(mean,cov,100)

[3]
def PCA(X, num_componentes):
    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[:,0:num_componentes]
    eigenvector_=sorted_eigenvec[:,0:num_componentes]

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



```
[4] mat_red=PCA(X,2)
[5] principal_df=pd.DataFrame(mat_red,columns=['PC1','PC2'])
[6]
principal_df=pd.concat([principal_df],axis=1)
principal_df
```

	PC1	PC2
0	0.040860	-0.252587
1	1.432637	1.170940
2	1.113698	-1.650798
3	-0.083308	-0.445220
4	-0.517488	-1.169287
...
95	0.256625	-0.820305
96	1.574706	2.198303
97	-0.653296	0.091154
98	-0.326205	0.787352
99	0.297593	-1.226972

100 rows × 2 columns


```
plt.figure(figsize=(6,6))
sb.scatterplot(data=principal_df,x='PC1',y='PC2',s=60,palette='icefire')
```

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).

```

import matplotlib.pyplot as plt
x=E[:,0]
y=E[:,1]
plt.plot(0.969, -0.102,marker='o',linestyle='none')
plt.plot(0.907, -0.392,marker='o',linestyle='none')
plt.plot(0.971, 0.156,marker='o',linestyle='none')
plt.plot(0.943, 0.252,marker='o',linestyle='none')
plt.grid()
plt.xlabel('eje 1')
plt.ylabel('eje 2')
plt.title('Proyección de componentes principales')
plt.legend(['1','2','3','4'],loc='upper left')
plt.show()

import numpy as np
S=np.array([[140.017, 107.881, 139.068, 109.095],[ 107.881,106.038,110.0439, 82.627],[ 139.068,110.0439,168.752,125.136],[109.095,82.627,125.136,108.960]])

E=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(S)

sorted_index = np.argsort(eigen_val)[::-1]
sorted_eigenval = eigen_val[sorted_index]
sorted_eigenvec = eigen_vec[:,sorted_index]
eigenvector_ = sorted_eigenvec[:,0:4]

def bt(i):
    bt=eigenvector_[:,i]
    return bt
def var(n):
    var=np.matmul(np.matmul(bt(n-1).T,S),bt(n-1))
    return var

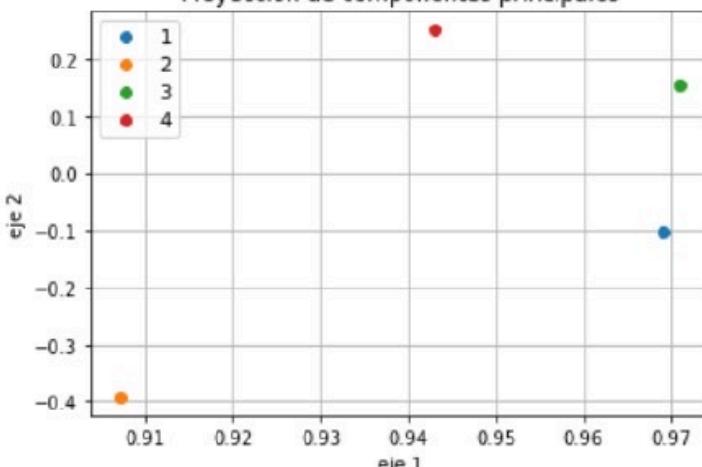
print("las varianzas empíricas son:",var(1),var(2),var(3),var(4))

varexp1=(eigenvector_[0,2]**2*sorted_eigenval[2])/np.sqrt(S[0,0])+(eigenvector_[0,3]**2*sorted_eigenval[3])/np.sqrt(S[0,0])
varexp2=(eigenvector_[1,0]**2*sorted_eigenval[0])/np.sqrt(S[1,1])+(eigenvector_[1,1]**2*sorted_eigenval[1])/np.sqrt(S[1,1])
print("las varianzas explicadas son:",varexp1,varexp2)

```

las varianzas empíricas son: 474.44435803491024 29.052324882714338 11.37657544558559 8.893741636789787
 las varianzas explicadas son: 0.5988730539901584 10.057786140903675

Proyección de 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 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.

```
datos = pd.read_csv('/content/tortugas.csv')

[ ] datos.fillna(datos.mean(), inplace = True)

<ipython-input-14-f6549c69270f>:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future versi
    datos.fillna(datos.mean(), inplace = True)

[ ] cols = [col for col in datos.columns if col !='sex']

cols_numericas = datos.select_dtypes(include=[np.number]).columns.tolist()

datos_log = np.log(datos[cols_numericas])

[ ] #Datos de las tortugas macho, tomando logaritmo

datos_machos = datos_log.iloc[:24, :]

print(datos_machos)
```

	length	width	height
0	4.584967	4.394449	3.637586
1	4.634729	4.430817	3.637586
2	4.634729	4.454347	3.737670
3	4.653960	4.454347	3.688879
4	4.691348	4.477337	3.784190
5	4.812184	4.521789	3.912023
6	4.812184	4.553877	3.828641
7	4.890349	4.595120	3.931826
8	4.890349	4.624973	3.931826
9	4.890349	4.624973	3.931826
10	4.897840	4.605170	3.871201
11	4.912655	4.624973	3.891820
12	4.919981	4.584967	3.931826
13	4.927254	4.595120	3.931826
14	4.948760	4.653960	3.970292
15	4.990433	4.682131	4.043051
16	5.003946	4.672829	4.007333
17	5.030438	4.672829	4.025352
18	5.043425	4.744932	4.143135
19	5.043425	4.762174	4.094345
20	5.062595	4.744932	4.127134
21	5.068904	4.770685	4.143135
22	5.087596	4.820282	4.110874
23	5.176150	4.882802	4.204693

```
[ ] vector_medio_machos = datos_machos.mean()
cov_machos = datos_machos.cov()
print("El vector medio es:",vector_medio_machos)

print("La matriz de covarianza es:",cov_machos)
```

```
El vector medio es: length      4.900356
width        4.622909
height       3.938253
dtype: float64
La matriz de covarianza es:           length      width      height
length    0.026391  0.020124  0.025443
width     0.020124  0.016190  0.019782
height    0.025443  0.019782  0.025899
```

```
[ ] vector_medio_hembras = datos_hembras.mean()
cov_hembras = datos_hembras.cov()

print("El vector medio es:", vector_medio_hembras)
print("La matriz de covarianza es:", cov_hembras)
```

```
El vector medio es: length      4.725444
width        4.477574
height       3.703186
dtype: float64
La matriz de covarianza es:           length      width      height
length    0.011072  0.008019  0.008160
width     0.008019  0.006417  0.006005
height    0.008160  0.006005  0.006773
```

```
[ ] val_machos, vec_machos = np.linalg.eig(cov_machos)
print(val_machos)
print(vec_machos)

[0.06719961 0.00053   0.00075045]
[[-0.62226438 -0.63686634 -0.45517943]
 [-0.48407139  0.77002303 -0.41561932]
 [-0.6151926   0.03828576  0.78744668]]
```

```
[ ] val_hembras, vec_hembras = np.linalg.eig(cov_hembras)
print(val_hembras)
print(vec_hembras)

[0.02330335 0.00035984 0.0005983 ]
[[ 0.68310233  0.71269743 -0.15947908]
 [ 0.51021953 -0.62195341 -0.59401177]
 [ 0.52253923 -0.32440148  0.78848997]]
```

```
[ ] pca_machos = PCA().fit(datos_machos)
varianza_explicada_machos = pca_machos.explained_variance_ratio_
print(varianza_explicada_machos)

[0.9813018 0.0109587 0.0077395]
```

```
[ ] pca_hembras = PCA().fit(datos_hembras)
varianza_explicada_hembras = pca_hembras.explained_variance_ratio_
print(varianza_explicada_hembras)

[0.96050774 0.02466069 0.01483157]
```

```
[ ] volumen_machos = np.exp(vector_medio_machos.prod()*(4/3))
volumen_hembras = np.exp(vector_medio_hembras.prod()*(4/3))
print(volumen_machos)
print(volumen_hembras)
```

```
4.5899963960073627e+51
2.352763875489634e+45
```

```
[ ] if volumen_machos > volumen_hembras:
    print("El volumen de los caparazones de machos es mayor.")
elif volumen_hembras > volumen_machos:
    print("El volumen de los caparazones de hembras es mayor.")
else:
    print("Los volumenes de los caparazones de machos y hembras son iguales.")
```

```
El volumen de los caparazones de machos es mayor.
```