PEC 4: PCA con R

José Manuel Alés Granados

2025-01-02

Enunciado de la práctica PEC3 de la asignatura de Álgebra Lineal de la UOC

En la televisión pública de vuestro país quieren relevar al meteorólogo de cabecera para calcular y presentar la predicción meteorológica en prime time. Después de un duro proceso de selección os acaban seleccionando y hoy es el primer día de trabajo.

Como especialistas en ciencia de datos, lo primero que queréis hacer es analizar períodos históricos temporales para observar los diferentes patrones y ver si se replican en el tiempo. Estáis interesados en conocer cuáles han sido las variables más relevantes para la predicción a lo largo del tiempo. Para realizar esta tarea, os proporcionan las observaciones diarias (a las 12 del mediodía) de diferentes variables relacionadas con la meteorología durante un período de 3 años (2006-2008).

- *weather label* : el tiempo del día (nublado: 0, lluvioso: 1, soleado: 2).
- *temperature*: temperatura (en grados centígrados).
- *temp app*: sensación térmica (en grados centígrados).
- *humidity*: humedad relativa (en tanto por uno [0-1]).
- *vind vel* : velocidad del viento (en kilómetros por hora).
- *vind dir* : dirección del viento (en grados).
- *visibility*: visibilidad (en kilómetros).
- *atm pres*: presión atmosférica (en milibares).

Una librería que os puede ser útil para realizar la práctica es`fields` , como veréis más adelante. Recordar que debéis instalarla una sola vez y luego importarla en el código.

Antes de empezar, debéis abrir la **"Tabla resumen de la Práctica 1"** del Moodle. Allí,encontraréis el valor de los parámetros (T , E) para poder realizar la práctica. Recordar, también, que debéis indicar los valores utilizados al inicio de la memoria, así como el intento de la Tabla correspondiente (primero o segundo).

La práctica se corresponde con el *segundo intento* y tiene como T y E los siguientes valores:

```r
# Valores práctica
intento = "2º intento"
T = 2
E = 80
cat(paste("La práctica se corresponde con el ", intento, ", con los valores de T = ",
          T, " y E = ", E))
```

```
## La práctica se corresponde con el  2º intento , con los valores de T =  2  y E =  80
```

Paso 1:

Primeramente, leer el fichero de datos correspondiente al período 2006-2008. `> data _ df <- read . csv ( ’/ home / data _ 0608. csv ’)` De la tabla resultante, guardar la primera columna (*weather label*) al vector y y las otras columnas a la matriz de características X. Responder: ¿qué dimensión tiene la matriz X?

```r
# Lectura del fichero
data_df <- read.csv('datasets/data_0608.csv')
y <- data_df %>%
  select(weather_label)

X <- data_df %>%
  select(-weather_label)

dimensiones_X <- dim(X)
cat(paste("La Matriz X tiene ", dimensiones_X[1], " filas y ",
            dimensiones_X[2], "columnas"))
```

```
## La Matriz X tiene  1096  filas y  7 columnas
```

Paso 2

Antes de realizar cualquier tipo de análisis, es importante hacer una exploración estadística (cuantitativa y cualitativa) de los datos. Para este propósito, observar el número de días con tiempo T y calcular su temperatura media (sólo de los días correspondientes a T)

```r
dias_T <- data_df %>%
  filter(weather_label == T)

# Número de días con T == 2
num_dias_T <- nrow(dias_T)
cat(paste("El número de días con tiempo T = ", T, " es ", num_dias_T , "\n"))
```

```
## El número de días con tiempo T =  2  es  93
```

```r
# Cálculo de la temperatura de los días con T == 2
temp_media_T <- mean(dias_T$temperature)
cat(paste("La temperatura media de los días con T = ", T, " es ", round(temp_media_T, 2)))
```

```
## La temperatura media de los días con T =  2  es  22.93
```

Paso 3

Para poder aplicar la descomposición en componentes principales, debéis normalizar la matriz de datos X siguiendo los criterios de la Sección 2.1 de los apuntes del módulo. Para hacerlo, debéis calcular la media y la desviación típica de los datos; guardar ambas en las variables m_X y s_X, respectivamente, ya que las necesitaréis más adelante. Nombrar a la nueva matriz de datos normalizada Xs. Una vez hecho, indicar la temperatura media de todo el período 2006-2008 de los datos normalizados.

```r
# Cálculo de la media y de la desviación estándar
m_X <- colMeans(X)
s_X <- apply(X, 2, sd)

# Normalización de la matriz X
Xs <- sweep(X, 2, m_X, "-")
Xs <- sweep(Xs, 2, s_X, "/")

temperatura_media_Xs <- mean(Xs[["temperature"]])

# Mostramos los resultados
cat("--Media de cada columna--\n")
```

```
## --Media de cada columna--
```

```r
cat(m_X)
```

```
## 15.48427 14.47927 0.597281 13.64112 187.6624 10.01661 996.2525
```

```r
cat("\n--Desviación típica de cada columna--\n")
```

```
##
## --Desviación típica de cada columna--
```

```r
cat(s_X)
```

```
## 10.66402 11.91016 0.206462 7.805106 107.2027 2.757732 146.1272
```

```r
cat(paste("\nLa temperatura media de la matriz normalizada es : ",
            temperatura_media_Xs))
```

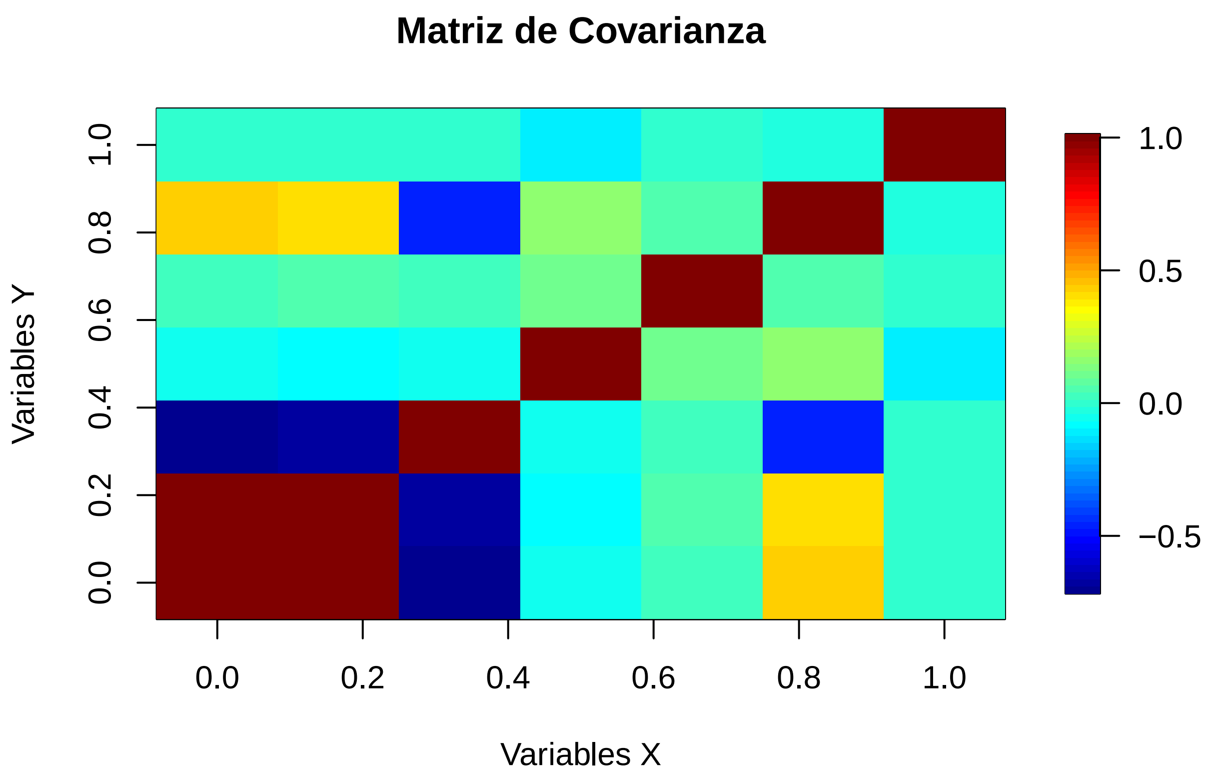
```
##
## La temperatura media de la matriz normalizada es :  6.38599036536197e-17
```

La temperatura media de los datos normalizados es 0.00.

Paso 4

Para ver la relación cruzada entre las distintas variables, observar la matriz de covarianza `CXs`. Dibujarla mediante la instrucción `image.plot()` de la librería `fields` y contestar: - ¿Qué variable está más asociada a la visibilidad en valor absoluto (y sin que sea ella misma)? - ¿Qué variable está menos asociada a la visibilidad en valor absoluto?

```r
# Cálculo de la matriz de covarianza
CXs <- cov(Xs)

# Impresión de la matriz de covarianza
image.plot(CXs, main = "Matriz de Covarianza", xlab = "Variables X",
           ylab = "Variables Y")
```




```r
# Obtenemos el índice de la columna visibilidad
index_visibility <- which(colnames(Xs) == "visibility")

# Guardamos el nombre de las columnas sin visibility para mostrar luego los
# nombres correctamente
colnames_ <- colnames(Xs)[-index_visibility]

# Obtenemos las asociaciones con 'visibility' excluyendo esta columna
visibility_asoc <- abs(CXs[index_visibility, -index_visibility])

# Comprobamos cuáles son las variables más y menos asociadas a la visibility
max_asoc <- which.max(visibility_asoc)
min_asoc <- which.min(visibility_asoc)

# Mostramos las variables más y menos asociadas
cat(paste("La variable más asociada con la visibilidad es ", colnames_[max_asoc]))
```

```
## La variable más asociada con la visibilidad es  humidity
```

```r
cat(paste("La variable menos asociada con la visibilidad es ", colnames_[min_asoc]))
```

```
## La variable menos asociada con la visibilidad es  atm_pres
```

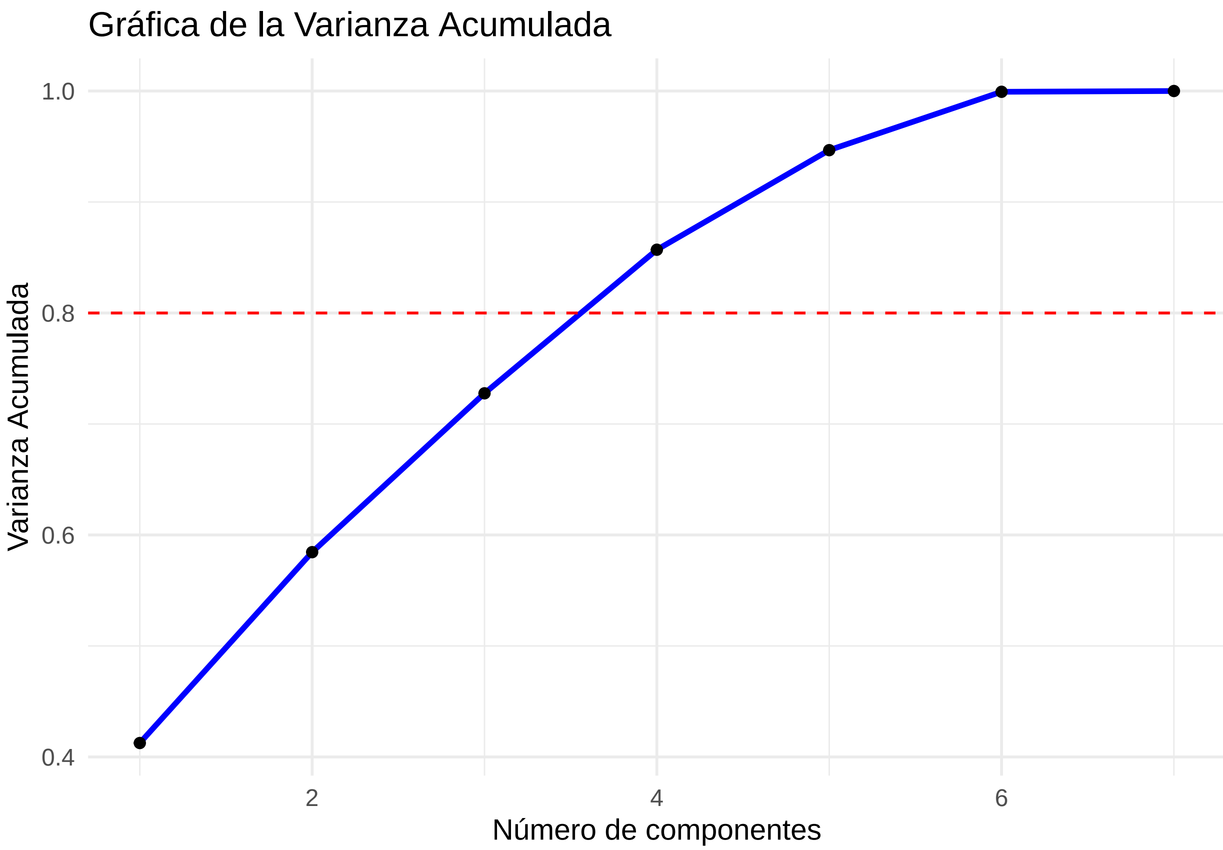
Respuesta 4:

- La variable más asociada a la visibilidad (en valor absoluto y sin ser ella misma) es **humidity**.
- La variable menos asociada a la visibilidad (en valor absoluto) es **atm_pres**.

Paso 5

Seguidamente, calcular la descomposición en componentes principales de la matriz de covarianza `CXs`. **Utilizar la instrucción** `eigen` y consultar la documentación si lo necesitáis. Esta función proporciona los valores y vectores propios (componentes principales) de forma ordenada de mayor a menor varianza explicada de los datos originales. Así, la primera componente corresponde a la dirección de máxima varianza mientras que la última componente corresponde a la dirección de mínima varianza. **Dibujar** la distribución de la **varianza acumulada** (eje de ordenadas) para cada componente principal (eje de abscisas) respecto a la varianza total de los datos. Seguidamente, **indicar** el número mínimo de componentes necesarios P para explicar un $E\%$ de la varianza inicial de los datos.

```
# Cálculo de los VAPS y los VEPS de la matriz de covarianza
CXs_VPS <- eigen(CXs)
valores_propios <- CXs_VPS$values

# Cálculo de la varianza total
varianza_total <- sum(valores_propios)

# Cálculo de la varianza explicada por componente
varianza_explicada <- valores_propios/varianza_total

# Cálculo de la varianza acumulada
varianza_acumulada <- cumsum(varianza_explicada)

# Creación de un dataframe para graficar la varianza
varianza_df <- data.frame(
  NComponente = 1:length(varianza_acumulada),
  VarianzaAcumulada = varianza_acumulada
)

# Gráfica de la varianza acumulada
ggplot(varianza_df, aes(x = NComponente, y = VarianzaAcumulada)) +
  geom_line(color="blue", linewidth=1) +
  geom_point(color="black", size=1.5) +
  geom_hline(yintercept = 0.8, linetype="dashed", color="red") +
  labs (
    title = "Gráfica de la Varianza Acumulada",
    x = "Número de componentes",
    y = "Varianza Acumulada"
  ) +
  theme_minimal()
```



Respuesta 5: Observando el gráfico podemos comprobar que nececesitamos al menos **4 componentes** para explicar el 80% de los datos

Paso 6

Con el tiempo, se os encarga un nuevo estudio, ahora durante el período 2009-2011. Como suposición inicial, considerar una distribución estacionaria de los datos, eso es, que sus propiedades estadísticas son constantes en el tiempo. Esto os permite utilizar la media y desviación típica anteriormente calculadas así como las componentes principales del período 2006-2008.

Empezar leyendo los datos del nuevo período y normalizarlos usando la media y desviación típica previamente calculadas (apartado 3); guardar los datos normalizados a la matriz `Xs_test`. Una vez hecho, **contestar**: ¿cuántos días de T habéis observado en este segundo período?

```
# Carga de los datos del período 2009-2011
data2_df <- read.csv('datasets/data_0911.csv')
y2 <- data2_df %>%
  select(weather_label)

X2 <- data2_df %>%
  select(-weather_label)

# Normalización de los datos de testeo con la media y desviación típica de los
# datos del período anterior
Xs_test <- sweep(X2, 2, m_X, "-")
Xs_test <- sweep(Xs_test, 2, s_X, "/")

# Cuando días T = 2 hay en el segundo período
```

```r
dias2_T <- data2_df %>%
  filter(weather_label==T)

# Mostramos el número de días por consola
print(paste("El número de días con T = 2 en el período 2009-2011 es ", nrow(dias2_T)))
```

```
## [1] "El número de días con T = 2 en el período 2009-2011 es  56"
```

Respuesta 6 El número de días con T = 2 en el periodo 2009-2011 es 56

Paso 7

Utilizando las P primeras componentes principales calculadas anteriormente, proyectar los datos del período 2009-2011 (normalizados) al nuevo subespacio. Guardar dicha proyección a la variable Xproj_test. Recordar que este subespacio tiene dimensión P . **Responder**: ¿qué proporción (en porcentaje) de la varianza inicial de los datos explican los datos del subespacio, Xproj_test? ¿Es mayor o menor que la obtenida en el período 2006-2008?

```r
# Número de componentes calculados previamente
P <- 4

# Extración de P vectores propios
VEPS_P <- CXs_VPS$vectors[, 1:P]

# Proyección de los datos 2009-2011 normalizado al subespacio de dimensión P
Xproy_test <- as.matrix(Xs_test) %*% VEPS_P

# Proporción de la varianza inicial explicada por las P componentes
varianza_explicada1 <- sum(valores_propios[1:P])/varianza_total * 100

# Mostramos la varianza explicada
print(paste("La varianza explicada por ", P,
            " componentes de los datos 2009-2011 es, ",
            round(varianza_explicada1, 2), "%"))
```

```
## [1] "La varianza explicada por  4  componentes de los datos 2009-2011 es,  85.7 %"
```

```r
# Calculamos las varianza explicada de los datos 2006-2009 con P componentes
varianza_explicada2 <- varianza_acumulada[P]*100

print(paste("La varianza explicada por ", P,
            " componentes de los datos 2006-2009 es, ",
            round(varianza_explicada2, 2), "%"))
```

```
## [1] "La varianza explicada por  4  componentes de los datos 2006-2009 es,  85.7 %"
```

```r
# Realizamos la comprobación de la varianza explicada de ambos periodos
dif <- abs(varianza_explicada1 - varianza_explicada2)

if (varianza_explicada1 > varianza_explicada2) {
  print(paste("El periodo 2009-2011 explica un ", dif, " de varianza más"))
} else if (varianza_explicada1 < varianza_explicada2) {
  print(paste("El periodo 2006-2009 explica un ", dif, " de varianza más"))
} else {
  print(paste("Ambos periodos explican el ", round(varianza_explicada2, 2),
```

```
             "% de la varianza por lo que son iguales"))
}
```

```
## [1] "Ambos periodos explican el  85.7 % de la varianza por lo que son iguales"
```

Paso 8

Finalmente, a partir de la proyección Xproj_test queréis recuperar los datos observados tal y como se indica a la Sección 2.5 y 2.5.1 de los apuntes del módulo. **Calcular** el error de reconstrucción y **responder**: ¿cuál es la desviación típica del error de reconstrucción de la temperatura? ¿Creéis que la suposición de una distribución estacionaria de los datos es correcta?

```
# Reconstrucción de la matriz de los datos normalizados desde su proyección
Xs_rec <- Xproy_test %*% t(VEPS_P)

# Desnormalización de los datos recuperados
X_rec <- sweep(Xs_rec, 2 , s_X, "*")
X_rec <- sweep(X_rec, 2, m_X, "+")

# Error
E <- as.matrix(X2) - X_rec

# Desviación estándar de la temperatura
err_temp <- E[, "temperature"]
sd_err_temp <- sd(err_temp)

# Representación por consola de la desviación estándar
print(paste("La desviación estándar de la temperatura es ", round(sd_err_temp, 2)))
```

```
## [1] "La desviación estándar de la temperatura es  3.29"
```

```
# Comprobación de la suposición inicial
if (sd_err_temp < s_X["temperature"]) {
  print("La suposición inicial de una distribución estacionaría parece razonable")
} else {
  print("La suposición inicial de una distribución estacionaría no " +
        "parece razonable")
}
```

```
## [1] "La suposición inicial de una distribución estacionaría parece razonable"
```

Realización de la PEC4 en Python

Para comprobar los resultados obtenidos en `R` he realizado la práctica en `Python`:

Paso 0: Importaciones

```
# Importaciones necesarias
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Paso 1:

```python
# Lectura del fichero

# Ruta relativa al conjunto de datos
file_path = "./datasets/data_0608.csv"

# Carga del archivo como dataframe
df = pd.read_csv(file_path)

# Mostramos las primeras filas del dataframe
print(df)
```

Carga de los datos del primer periodo

```
##       weather_label  temperature   temp_app  ...  wind_dir  visibility  atm_pres
## 0                 1    16.061111  16.061111  ...       212      9.9015   1024.37
## 1                 0    13.816667  13.816667  ...       282     11.4954   1017.78
## 2                 0    12.288889  12.288889  ...       236      9.9820   1020.96
## 3                 0    21.111111  21.111111  ...       299      9.9820   1009.09
## 4                 0    12.172222  12.172222  ...       329     11.2056      0.00
## ...             ...          ...        ...  ...       ...         ...       ...
## 1091              0    23.861111  23.861111  ...       110     16.1000   1019.89
## 1092              0    22.838889  22.838889  ...        39     16.1000   1017.60
## 1093              2    23.861111  23.861111  ...        31     16.1000   1020.81
## 1094              0    26.038889  26.038889  ...       351     16.1000   1022.90
## 1095              1    21.855556  21.855556  ...       298      9.9820   1017.60
##
## [1096 rows x 8 columns]
```

```python
def division_datos(datos):
    """
    Dividimos los datos en las dos matrices que nos pide el ejercicio: Es decir
    separamos la primera columna en
    una nueva matriz y con el resto de columnas creamos una nueva matriz.
    ---
    Args:
        datos(Dataframe): conjuntos de datos que vamos a dividir.
    Return:
        y: Matriz con los datos de la primera columna
        X: Matriz con el resto de los datos
    """
    # Separación de la primera columna en el vector y el resto de las columnas
    # en la matriz X
    y = df.iloc[:, 0] # Primera columna
    X = df.iloc[:, 1:] # Todas las columnas excepto la primera

    return y, X

# División en dos tablas
y, X = division_datos(df)

# Mostramos las dimensiones de X
print(f"Las dimensiones de X son : {X.shape}")
```

División de los datos

```
## Las dimensiones de X son : (1096, 7)
```

Paso 2: Análisis previo

```python
def analisis_T(T, y, X):
  """
  Función que mostrara el número de días cuyo 'weather_label' es igual a T, y la
  temperatura media de esos días.
  ---
  Args:
    T (int): Valor del tipo de día que se quiere analizar.
    y (dataframe): Conjunto que almacena la columna 'weather_label'.
    X (dataframe): Conjunto de datos que almacena el resto de columnas de las
    observaciones.
  """
  # Obtenemos los indices de los días T
  indices_T = y[y==T].index

  # Obtenemos las filas correspondientes en X
  X_T = X.loc[indices_T]

  # Calculamos la temperatura media de los días T
  temperatura_media = X_T['temperature'].mean()

  # Mostramos los resultados
  # Número de días en los que 'weather_label' = T
  print(f"El número de días lluviosos (T=1) es: {len(indices_T)}")

  # Temperatura media en los días T
  print(f"La temperatura media en los días que T es igual a {T} es " \
        f"{round(temperatura_media, 2)}")

T = 2

analisis_T(T, y, X)
```

```
## El número de días lluviosos (T=1) es: 93
## La temperatura media en los días que T es igual a 2 es 22.93
```

Paso 3: Primeras operaciones con el PCA

```python
# Calculamos la media y la desviación estándar de cada columna
m_X = X.mean(axis=0) # Media
s_X = X.std(axis=0, ddof=1) # Desviación estándar (ddof=0 para dividir por n-1)

# Normalizamos la matriz X
Xs = (X - m_X) / s_X

# Calculamos la temperatura media de los datos normalizados
temperatura_media_Xs = Xs['temperature'].mean()

# Mostramos los resultados
```

```python
print(f"Media de las columnas (m_X) : \n{m_X}")
```

```
## Media de las columnas (m_X) :
## temperature     15.484271
## temp_app        14.479268
## humidity         0.597281
## wind_vel        13.641122
## wind_dir       187.662409
## visibility      10.016609
## atm_pres       996.252546
## dtype: float64
```

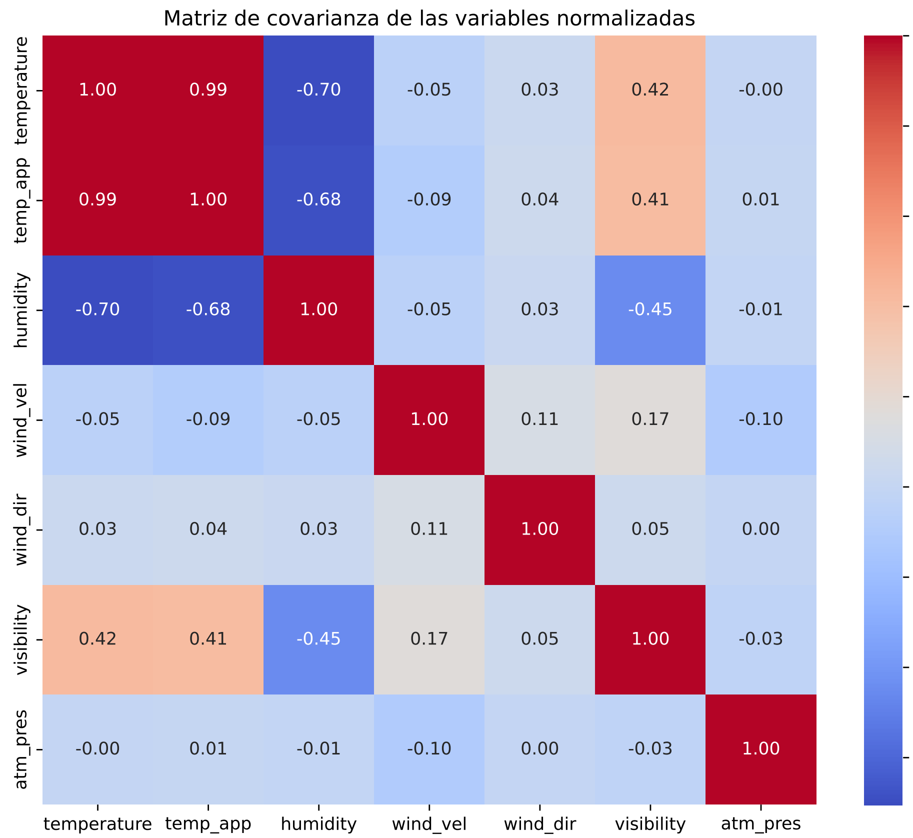
```python
print(f"Desviación estándar de las columnas (s_X) : \n{s_X}")
```

```
## Desviación estándar de las columnas (s_X) :
## temperature     10.664018
## temp_app        11.910165
## humidity         0.206462
## wind_vel         7.805106
## wind_dir       107.202710
## visibility       2.757732
## atm_pres       146.127179
## dtype: float64
```

```python
print(f"Temperatura media de la matriz normalizada : \n{temperatura_media_Xs}")
```

```
## Temperatura media de la matriz normalizada :
## -9.724581237592612e-17
```

```python
# Calculamos la matriz de covarianza
# rowvar=False asegura que las columnas sean las variables
cov_matriz = np.cov(Xs, rowvar=False)

# Convertimos la matriz de covarianza en un Dataframe para etiquetas claras
cov_df = pd.DataFrame(cov_matriz, columns=Xs.columns, index=Xs.columns)

# Dibujamos el mapa de calor
plt.figure(figsize=(10, 8))
sns.heatmap(cov_df, annot=True, cmap="coolwarm", fmt=".2f", cbar=True)
plt.title("Matriz de covarianza de las variables normalizadas")
plt.show()
```



Paso 4 : Matriz de covarianza

```
# Análisis específico de la visibilidad
# Excluimos visibilidad
visibilidad_corr = cov_df['visibility'].drop('visibility', errors='ignore')
max_corr = visibilidad_corr.abs().idxmax() # Variable más asociada
min_corr = visibilidad_corr.abs().idxmin() # Variable menos asociada

# Mostramos los resultados
print(f"Variable más asociada a la visibilidad: {max_corr} (valor absoluto = "\
      f"{visibilidad_corr[max_corr]:2f})")
```

```
## Variable más asociada a la visibilidad: humidity (valor absoluto = -0.449831)
```

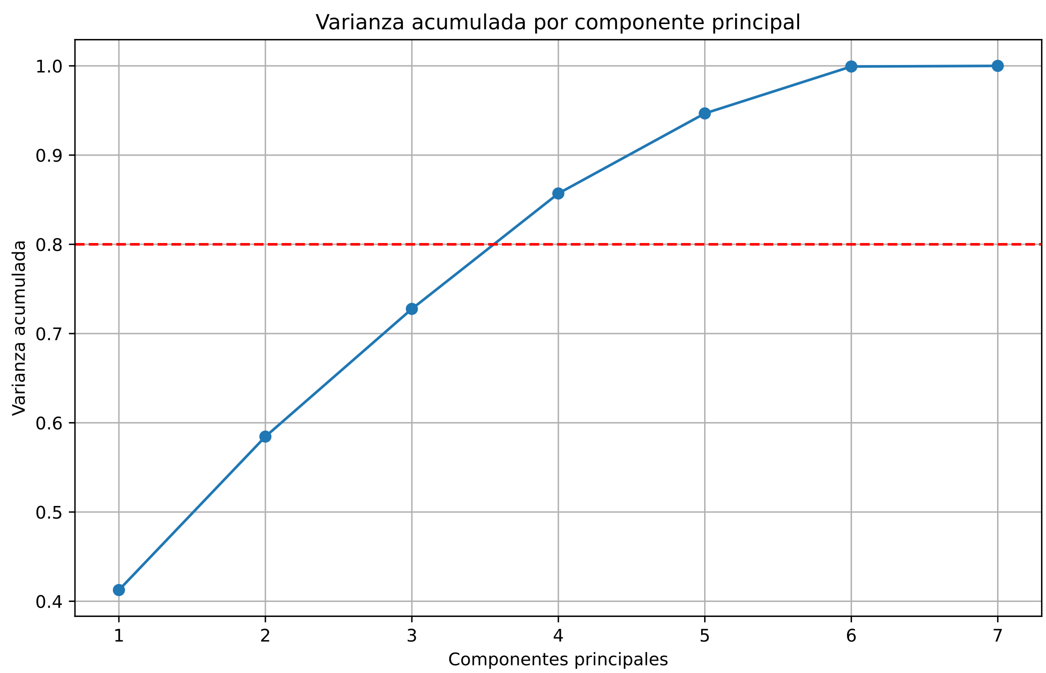
```
print(f"Variable menos asociada a la visibilidad: {min_corr} (valor  "\
      f"absoluto = {visibilidad_corr[min_corr]:2f})")
```

```
## Variable menos asociada a la visibilidad: atm_pres (valor  absoluto = -0.025329)
```

Paso 5

```
# Calculamos los valores y vectores propios
eigen_values, eigen_vectors = np.linalg.eig(cov_matriz)

# Ordenamos los indices de los valores de mayor a menor y ordenamos los VAPS y
```

```python
# VEPS
sorted_indices = np.argsort(eigen_values)[::-1]
eigen_values = eigen_values[sorted_indices]
eigen_vectors = eigen_vectors[:, sorted_indices]

# Calculamos la varianza total explicada y acumulada
total_varianza = np.sum(eigen_values)
explicada_varianza = eigen_values / total_varianza
acumulada_varianza = np.cumsum(explicada_varianza)

# Dibujamos la distribución de la varianza acumulada
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(eigen_values) + 1), acumulada_varianza, marker = "o",
          linestyle="-")
plt.axhline(y=0.8, color="red", linestyle="--", label="Umbral 80%")
plt.xlabel("Componentes principales")
plt.ylabel("Varianza acumulada")
plt.title("Varianza acumulada por componente principal")
plt.grid()
plt.show()
```




```python
# Calculamos el mínimo número de componentes principales para explicar E%
E = 0.8
componentes_necesarios = np.argmax(acumulada_varianza >= E) + 1
print(f"Para explicar al menos el {E * 100}% de la varianza,se necesita al "\
      f"menos {componentes_necesarios} componentes principales")
```

```
## Para explicar al menos el 80.0% de la varianza,se necesita al menos 4 componentes principales
```

Paso 6

```
# Comprobar porque los resultados son muy raros.
# Cargar los datos del período 2009-2011
file_path2 = "./datasets/data_0911.csv"
data_test = pd.read_csv(file_path2)

# Repetimos la división
y_test, X_test = division_datos(data_test)

# Normalizamos los datos del nuevo período usando
# la media y desviación del anterior dataset
Xs_test = (X_test - m_X)/s_X

# Mostramos los datos normalizados
print("Primera filas de la matriz de testeo normalizada")
```

```
## Primera filas de la matriz de testeo normalizada
```

```
print(Xs_test.head())
```

```
##    temperature  temp_app  humidity  wind_vel  wind_dir  visibility  atm_pres
## 0     0.054092  0.132815 -0.083701 -1.135080  0.227024   -0.041741  0.192418
## 1    -0.156377 -0.055633 -1.391447  0.913233  0.879993    0.536234  0.147320
## 2    -0.299641 -0.183908 -0.858662 -0.736969  0.450899   -0.012550  0.169082
## 3     0.527647  0.556822 -1.052402  1.241210  1.038571   -0.012550  0.087851
## 4    -0.310582 -0.193704 -1.343012  2.169449  1.318414    0.431148 -6.817709
```

```
# Días cuando T = 2: primera salida
analisis_T(T, y_test, X_test)
```

```
## El número de días lluviosos (T=1) es: 93
## La temperatura media en los días que T es igual a 2 es 22.93
```