ACP

DANIEL MARTÍN GARCÍA (versión de Pablo Flores Vidal)

Dic 2024

1 Introducción

Cuando se recoge la información de una muestra de datos, lo más frecuente es tomar el mayor número posible de variables. Sin embargo, disponer de un amplio número de variables puede ocasionar diferentes problemas. En este contexto complejo y multidimensional, surge la necesidad de técnicas que puedan simplificar y resumir la información de manera eficiente. Aquí es donde el Análisis de Componentes Principales (ACP) juega un papel fundamental.

El ACP es una técnica estadística (fuertemente anclada en las matemáticas) que busca reducir la dimensionalidad de un conjunto de datos, manteniendo al mismo tiempo la mayor cantidad de información ($\approx$ variabilidad) posible. En un mundo donde los datos masivos son comunes, la posibilidad de simplificar la información sin perder sus características esenciales se convierte en una herramienta poderosa.

Uno de los principales problemas que el ACP ayuda a resolver es la multicolinealidad, es decir, la correlación alta entre diversas variables. Esto puede complicar la interpretación de los datos y afectar la eficacia de otros métodos de análisis. Transformando los datos en componentes principales, el ACP elimina este problema, facilitando la interpretación y el análisis.

Además, el ACP permite una visualización más clara de los datos. Al reducir la cantidad de variables a unos pocos componentes principales, se pueden representar gráficamente los datos en un espacio de baja dimensión. Esto es invaluable para la comprensión y el descubrimiento de patrones y relaciones en conjuntos de datos complejos.

La versatilidad del ACP es otra de sus fortalezas. No se limita a un campo de estudio específico, sino que se aplica en a cualquier disciplina donde quepa la ciencia de Datos, como la biología, economía, psicología, y demás. Su capacidad para detectar estructuras en los datos ha llevado a avances significativos en campos tan diversos como la genómica o las finanzas.

El ACP también contribuye a la mejora de la calidad de los datos. Identificar y eliminar las variables redundantes mejora la precisión de los modelos, permitiendo un análisis más ajustado a la realidad. Además, a mayor dimensionalidad de los datos, mayor es la capacidad computacional que se precisa. Por lo tanto, reducir la dimensionalidad mediante ACP no solo optimiza los recursos computacionales sino que también facilita el manejo y procesamiento de grandes conjuntos de datos.

En conclusión, el Análisis de Componentes Principales es más que una simple técnica de reducción de dimensionalidad. Es una herramienta multifacética que responde a los desafíos de la era del Big Data, permitiendo una interpretación de los datos más sencilla, eliminando la multicolinealidad, facilitando la visualización, y contribuyendo a la calidad y eficiencia de los análisis. Su aplicabilidad en diversos campos y su capacidad para desentrañar patrones ocultos en datos complejos subrayan su importancia y justifican su uso continuo en el análisis de datos moderno.

2 Análisis de Componentes Principales

2.1 ¿En qué consiste el ACP?

El Análisis de Componentes Principales es una técnica estadística que se aplica a un conjunto de datos compuesto por variables que están potencialmente correlacionadas entre sí. El objetivo central del ACP es transformar estas variables originales en un nuevo conjunto de variables, que son linealmente independientes y, por lo tanto, no correlacionadas entre sí, si la estructura de los datos lo permite. Estas nuevas variables se denominan componentes principales.

La transformación se lleva a cabo mediante una combinación lineal de las variables originales, donde se seleccionan los coeficientes de manera que el primer componente principal capture la mayor parte de la variabilidad presente en los datos originales. Esto significa que esta primera componente resume, en la medida de lo posible, la información contenida en las variables originales.

El segundo componente principal se calcula de manera similar, pero está restringido a ser ortogonal al primer componente, es decir, no correlacionado con él. Este segundo componente explica la siguiente mayor parte de la variabilidad que no ha sido capturada por el primer componente.

Este proceso continúa de manera secuencial, con cada componente principal subsiguiente explicando una proporción decreciente de la variabilidad restante, y siempre siendo ortogonal a todos los componentes principales anteriores. La transformación se realiza de tal manera que se puede representar la estructura compleja y multidimensional de los datos originales en un número reducido de dimensiones, sin perder significativamente la información esencial.

El resultado es una representación simplificada y estructurada de los datos originales, donde los componentes principales capturan y resumen la información crítica, facilitando así la interpretación y el análisis posterior. La eficacia de esta transformación depende de la naturaleza de las relaciones entre las variables originales y de la variabilidad inherente en los datos, y el cumplimiento de los requisitos que mencionaremos a continuación garantiza una aplicación más precisa y significativa del ACP.

A continuación, en la Figura 1, se muestra (a la izquierda) una nube de puntos que representa la relación entre las variables originales (X, Y), altamente correlacionadas. Por otro lado, a la derecha se presenta el resultado del ACP: dos variables ortogonales que han sido derivadas de las variables originales (X, Y). Aunque la dimensionalidad no se ha reducido (sigue siendo un espacio bidimensional), lo notable aquí es que la correlación entre las variables originales ha sido eliminada. Las nuevas variables, ahora ortogonales entre sí, ofrecen una vista descorrelacionada de los datos,

preservando la estructura subyacente pero en un marco donde las variables no están linealmente relacionadas.

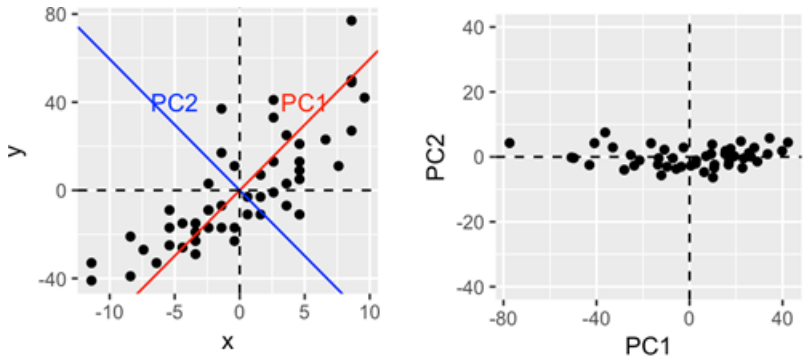

Figure 1: Representación de las Componentes.

2.2 Diferencias con el Análisis Factorial

El Análisis de Componentes Principales (ACP) y el Análisis Factorial (AF) son técnicas de reducción de la dimensionalidad, pero tienen propósitos y enfoques distintos, lo que puede generar confusión.

1. **Objetivo principal**
   ACP: Busca transformar un conjunto de variables originales en un nuevo conjunto de variables no correlacionadas llamadas componentes principales. Estas componentes explican la máxima varianza posible del conjunto de datos. Su meta es descriptiva, resumir los datos de forma más manejable.
   AF: Intenta identificar factores latentes (no observables) que explican las correlaciones entre las variables originales. Su meta es explicar las relaciones subyacentes en los datos, idealmente en términos de constructos teóricos.

2. **Modelo matemático**
   ACP: No asume un modelo subyacente. Es puramente algorítmico y se basa en la diagonalización de la matriz de covarianzas o correlaciones. Se busca una combinación lineal de las variables que maximice la varianza explicada.
   AF: Utiliza un modelo subyacente que supone que las variables observadas están influenciadas por factores comunes (latentes) y factores únicos (error). Se basa en la ecuación:

   $$\mathbf{X} = \Lambda F + \epsilon$$

   donde: $\mathbf{X}$: Variables observadas, Λ: Matriz de cargas factoriales (relación entre factores y variables), F: Factores latentes, ϵ: Error específico o varianza única.

3. **Varianza explicada**
   ACP: Explica toda la varianza de las variables originales (tanto común como única). Por esto, los componentes principales suelen ser útiles para compresión y visualización de datos.
   AF: Solo intenta explicar la varianza común entre las variables, ignorando la varianza única. Esto hace que sea más útil para identificar patrones o constructos subyacentes.

4. **Método de extracción**
   ACP: Se basa en la descomposición en valores propios (eigenvalues) de la matriz de correlación o covarianza.
   AF: Emplea métodos como máxima verosimilitud, mínimos cuadrados ponderados o métodos iterativos para estimar las cargas factoriales y los factores comunes.

5. **Interpretación**
   ACP: Los componentes principales son combinaciones lineales de las variables originales, y no tienen una interpretación teórica más allá de representar la dirección de mayor varianza en los datos.
   AF: Los factores latentes tienen una interpretación más teórica y suelen representar constructos subyacentes (por ejemplo, "inteligencia general" en psicometría o "satisfacción del cliente" en estudios de mercado).

Resumen:
ACP: Ideal para resumir datos y reducir la dimensionalidad, explicando varianza sin enfocarse en factores latentes.
AF: Diseñado para descubrir relaciones subyacentes y latentes en los datos, basado en un modelo teórico.

2.3 ¿Qué requisitos precisan los datos originales?

Para que un conjunto de datos sea susceptible de análisis mediante el ACP, es vital cumplir con ciertos requisitos:

- **Escala de las Variables**: Las variables de entrada deben estar en la misma escala o ser estandarizadas, para que cada una contribuya de manera equitativa al análisis.
- **Ausencia de Valores Atípicos**: Los datos extremos pueden dificultar el análisis, y su manejo adecuado es fundamental.
- **Datos perdidos**: La ausencia de datos puede complicar el análisis, por lo que pueden ser necesarias técnicas de imputación.
- **Distribución de las Variables Originales**: Aunque no es necesario que las variables originales tengan una distribución normal, esto puede mejorar la interpretabilidad de los resultados.

Nota: Es fundamental destacar que el ACP presupone correlaciones entre las variables originales. Aunque técnicamente es posible llevar a cabo el análisis en ausencia de correlaciones, el resultado sería trivial y no proporcionaría ninguna comprensión nueva o valiosa acerca de la estructura de los datos. En otras palabras, sin correlaciones entre las variables, el ACP no obtendría ningún resultado útil.

2.4 Pasos del Proceso

Supongamos que tenemos un conjunto de datos con n observaciones y p variables donde X_{ij}, es el valor de la observacion i-esima, en la variable j-esima, y queremos reducirlo a k componentes principales.

1. **Estandarización de los Datos:** Si las escalas de las variables originales difieren, deben ser estandarizadas:
$$Z_{ij} = \frac{X_{ij} - \bar{X}_j}{s_j}, i \in n,\, j \in p,$$
donde $\bar{X}_j$ es la media de la variable j-esima, y s_j es su desviación estándar.

2. **Cálculo de la Matriz de Covarianzas Σ:** Si los datos originales están en la misma escala, (poco habitual), podremos utilizar la matriz de covarianzas, la cual, es una matriz cuadrada $p \times p$ que contiene las covarianzas entre cada par de variables originales.

   La entrada en la i-ésima fila y j-ésima columna de la matriz de covarianzas Σ es la covarianza entre X_i y X_j, denotada como $\text{Cov}(X_i, X_j)$. La covarianza entre dos variables se calcula mediante la fórmula:

$$\text{Cov}(X_i, X_j) = \frac{1}{n-1} \sum_{k=1}^{n} (X_{i,k} - \bar{X}_i)(X_{j,k} - \bar{X}_j)$$

   Donde:

   - n es el número de observaciones.
   - $X_{i,k}$ es la k-ésima observación de la variable X_i.
   - $\bar{X}_i$ es la media de la variable X_i.

   La matriz de covarianzas Σ es una matriz simétrica de $p \times p$, donde cada elemento Σ_{ij} representa la covarianza entre X_i y X_j.

   En forma matricial, la matriz de covarianza Σ se expresa como:

$$\Sigma = \begin{bmatrix} \text{Cov}(X_1, X_1) & \text{Cov}(X_1, X_2) & \dots & \text{Cov}(X_1, X_p) \\ \text{Cov}(X_2, X_1) & \text{Cov}(X_2, X_2) & \dots & \text{Cov}(X_2, X_p) \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}(X_p, X_1) & \text{Cov}(X_p, X_2) & \dots & \text{Cov}(X_p, X_p) \end{bmatrix}$$

   Es importante señalar que al trabajar con datos muestrales, se utiliza el factor $\frac{1}{n-1}$ para calcular la matriz de covarianza muestral. Si se tiene toda la población, se usaría $\frac{1}{n}$ en su lugar.

3. **Cálculo de la Matriz de Correlaciones:** Si las variables originales no están medidas en la misma escala, entonces, utilizaremos la matriz de correlaciones, la cual, se calcula a partir de los datos estandarizados y vuelve a ser una matriz cuadrada $p \times p$ que contiene las correlaciones entre cada par de variables estandarizadas. Se calcula como:

$$R = \begin{bmatrix} \rho(Z_1, Z_1) & \rho(Z_1, Z_2) & \dots & \rho(Z_1, Z_p) \\ \rho(Z_2, Z_1) & \rho(Z_2, Z_2) & \dots & \rho(Z_2, Z_p) \\ \vdots & \vdots & \ddots & \vdots \\ \rho(Z_p, Z_1) & \rho(Z_p, Z_2) & \dots & \rho(Z_p, Z_p) \end{bmatrix}$$

   En esta notación:

- $\rho(Z_i, Z_j)$ representa la correlación entre Z_i y Z_j.
- R es la matriz de correlaciones.
- La matriz es simétrica y cada elemento R_{ij} representa la correlación entre Z_i y Z_j.

Ten en cuenta que el coeficiente de correlación se indica comúnmente con la letra griega ρ (rho).

4. **Determinación de los Valores Propios:** A partir de la matriz de covarianza (o de correlaciones), se calculan los valores propios, resolviendo la ecuación característica $\det(\Sigma - \lambda I) = 0$, donde Σ es la matriz de covarianzas o la matriz de correlaciones, I es la matriz identidad, matriz de dimensión $p \times p$ con unos en la diagonal y ceros en todas las demás posiciones; y λ es el vector que contiene el conjunto de valores propios

5. **Determinación de los k primeros Valores Propios:** Los autovalores se ordenan en orden descendente, y se eligen los primeros k, que representarán los componentes principales. Para elegir estos k primeros, hay que tener en cuenta que el valor de cada uno de los autovalores λ_i para $i = 1, ..., p$, es proporcional a la variabilidad explicada por las componentes asociadas. Por ejemplo, estando los autovalores ordenados, tal que $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_p$, la componente asociada al primer autovalor λ_1, explica un $\frac{\lambda_1}{\sum_{i=1}^{p} \lambda_i} * 100$ por ciento de la variabilidad total de las variables originales.

6. **Determinación de los vectores propios asociados:** Una vez que tenemos los valores propios λ, podemos encontrar los autovectores asociados, resolviendo la siguiente ecuación lineal para cada λ: $(\Sigma - \lambda I)w = 0$. Donde w es el vector propio que estamos buscando.

7. **Construcción de la Matriz de Transformación:** Los primeros k vectores propios forman la matriz de transformación. Es decir, $W = [w_1, w_2, \ldots, w_k]$, donde w_i, para $i = 1, ..., k$ son los primeros k vectores propios.

8. **Cálculo de los Componentes Principales:** Una vez que se ha obtenido la matriz de transformación W, las s componentes principales CP se pueden calcular como la proyección de los datos originales X o los datos estandarizados Z en el espacio de las nuevas dimensiones (componentes principales). Esto se realiza mediante una multiplicación matricial entre los datos y la matriz de transformación:

$$CP = XW \quad \text{ó} \quad CP = ZW$$

donde CP es una matriz de dimensiones $n \times k$ que contiene los componentes principales de los datos. Cada columna de CP representa un componente principal, y cada fila de CP representa los componentes principales para una observación individual en el conjunto de datos original. Este paso produce las coordenadas de las observaciones en el nuevo espacio definido por los componentes principales.

Nota: Incluso aunque las variables se encuentren en la misma escala, siempre es aconsejable estandarizar los datos, por los siguientes motivos:

- **Influencia de Variables:** Si no estandarizas, las variables con mayor variabilidad tendrán más peso en la formación de los componentes principales. Esto puede ser problemático si la variabilidad es alta debido a ruido o errores de medición, y no porque la variable sea particularmente importante para el fenómeno estudiado.
- **Comparabilidad:** Si planeas comparar los resultados de tu ACP con otros estudios que también utilizan ACP pero en diferentes escalas, la estandarización podría ser esencial para hacer que los resultados sean comparables.

2.5 Selección del Número de Componentes

Una de las decisiones fundamentales en el Análisis de Componentes Principales es la selección del número de componentes principales, denotado como k, que se deben retener. Las componentes principales resultantes son linealmente independientes y ortogonales entre sí, y proporcionan un resumen compacto y eficiente de la variabilidad en los datos originales.

La elección de k es esencial para garantizar que el ACP capture una cantidad significativa de la variabilidad presente en los datos originales sin incluir componentes innecesarios. Una técnica común para determinar k es utilizar el *método del codo*, que se basa en el gráfico de la varianza explicada.

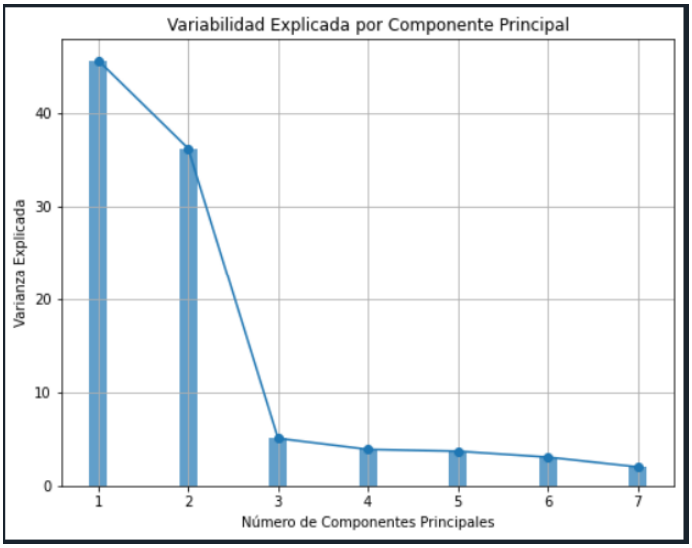

Figure 2: Representación de la variabilidad explicada

En este gráfico, el eje x representa el número de componentes y el eje y representa la proporción de varianza explicada. Buscamos el punto en el que agregar componentes adicionales ya no proporciona una ganancia significativa en la varianza explicada. Este punto se asemeja a la forma de un 'codo' en la curva, y marca el número óptimo de componentes a retener. En otras palabras, seleccionamos k de manera que capturemos la mayor parte de la variabilidad en los datos originales sin introducir redundancias innecesarias.

Este proceso de selección de k es crucial para obtener un modelo de ACP efectivo que reduzca la dimensionalidad de los datos de manera informativa y eficiente.

2.6 Relación entre las Componentes Principales y las variables

En el contexto del Análisis de Componentes Principales, es fundamental comprender la relación entre las componentes principales y las variables originales. Esta relación se basa en los siguientes conceptos clave:

- λ_i: Representa el autovalor asociado a la i-ésima componente principal (CP_i) obtenido durante el ACP. Los autovalores indican cuánta varianza (variabilidad) se explica por cada CP_i. Un autovalor alto (λ_i grande) implica que CP_i retiene una gran cantidad de información o variabilidad de los datos originales, mientras que un autovalor bajo (λ_i pequeño) indica que CP_i retiene menos información.

- e_{ij}: Representa el coeficiente ubicado en la posición ij del vector propio asociado a la CP_i. Este coeficiente indica la contribución relativa de la variable original X_j en la construcción de la CP_i. En otras palabras, e_{ij} muestra cuánto peso tiene la variable X_j en la definición de los valores de CP_i.

- $Cov(CP_i, X_j)$. La covarianza entre la i-ésima componente principal (CP_i) y la j-ésima variable original (X_j) se calcula como $\lambda_i \cdot e_{ij}$. Esta covarianza nos proporciona información sobre la relación lineal entre CP_i y X_j. Si el valor resultante es positivo, significa que ambas variables tienden a aumentar o disminuir juntas (correlación positiva). Si es negativo, indica que una variable tiende a aumentar cuando la otra disminuye (correlación negativa). Un valor cercano a cero sugiere que no hay una fuerte relación lineal entre CP_i y X_j.

- $Corr(CP_i, X_j)$ La correlación entre una componente principal CP_i y una variable X_j se calcula como

$$Corr(CP_i, X_j) = \frac{Cov(CP_i, X_j)}{\sqrt{Var(CP_i)Var(X_j)}} = \frac{\lambda_i e_{ij}}{\sqrt{\lambda_i s_j^2}} = e_{ij}\frac{\sqrt{\lambda_i}}{s_j},$$

2.6.1 Contribuciones de las Componentes Principales a la Variabilidad Explicada de las Variables Originales

Para evaluar si las nuevas variables generadas por el ACP capturan eficazmente la información contenida en las variables originales, y para determinar qué variables se explican mejor (o peor) mediante estas nuevas componentes, utilizamos una medida conocida como 'cosenos al cuadrado', que se calculan como:

$$Cos_{ij}^2 = Corr^2(CP_i, X_j).$$

Los cosenos al cuadrado, a pesar de su nombre, no están relacionados directamente con la función coseno, sino que se derivan de la geometría subyacente del ACP. Estos valores representan las correlaciones al cuadrado entre cada variable original y las componentes principales. Expresan la proporción de la varianza de cada variable original que se explica mediante cada componente principal.

En otras palabras, los cosenos al cuadrado nos indican cuánto de la variabilidad de una variable original está contenida en cada componente principal. Un valor cercano a 1 implica que la variable se explica en gran medida por la componente, mientras que un valor cercano a 0 indica que la componente tiene poco poder para explicar la variabilidad de esa variable.

El uso de los cosenos al cuadrado es esencial para evaluar la capacidad del ACP para resumir y retener información relevante de las variables originales, lo que nos permite identificar qué variables son más influyentes en la definición de cada componente principal.

2.6.2 Contribuciones de las Variables a las Componentes Principales

Para comprender cómo cada variable original contribuye a la construcción de las CP en el Análisis de Componentes Principales, podemos calcular su contribución específica. La contribución de una variable en la creación de una CP se calcula mediante la fórmula:

$$Contribucion_{ji} = Cos^2_{ij}\sqrt{\lambda_i} \text{ para todo i=1,...,k; j=1,...,p.}$$

Esta fórmula nos permite cuantificar la influencia de cada variable en la definición de una CP. Al evaluar las contribuciones de las variables, podemos identificar cuáles son las más relevantes para cada CP y comprender mejor cómo se estructuran las CP en función de las variables originales.

Estas contribuciones estan en terminos relativos, por lo que se acostumbra a normalizar estas contribuciones entre $[0, 1]$ o entre $[0, 100]$ para que sea más fácilmente comparable la contribución de una variable con respecto al resto en la construcción de una CP.

3 Ejemplo con Python

Para comprender mejor cómo funciona el Análisis de Componentes Principales (ACP) en la práctica, consideremos un ejemplo concreto que involucra las notas de 120 alumnos en siete asignaturas diferentes: Matemáticas, Física, Lengua, Inglés, Historia, Literatura y Economía. Lo que crea un conjunto de datos multidimensional. A priori, es claro que estas notas pueden mostrar correlación entre sí. Nuestro objetivo es utilizar el ACP para simplificar esta información compleja (reduciendo la dimensionalidad) y obtener una visión más clara de las tendencias subyacentes en el desempeño académico de los estudiantes (creando las componentes principales).

A medida que avanzamos en este ejemplo, exploraremos paso a paso cómo aplicar el ACP a estos datos utilizando Python, desde la selección del número adecuado de componentes principales hasta la interpretación de los resultados. A lo largo del proceso, veremos cómo el ACP nos permite reducir la dimensionalidad de nuestros datos y facilitar la toma de decisiones informadas en el entorno educativo.

El fichero con el que vamos a trabajar se llama 'NOTAS.xlsx' y a continuación, en la Figura 3 se muestran los primeros 15 registros.

	A	B	C	D	E	F	G	H	I
1	Alumno	Matematicas	Fisica	Lengua	Ingles	Historia	Literatura	Economia	EXTRA_ESC
2	1	8,21	6,58	6,50	7,42	9,27	8,04	5,80	Musi-Dep
3	2	5,37	3,93	6,04	6,07	6,19	7,32	4,32	Deporte
4	3	4,38	2,99	6,43	6,25	5,40	6,74	6,24	Deporte
5	4	7,64	5,81	8,02	8,07	7,28	9,03	6,34	Musi-Dep
6	5	5,39	5,76	5,21	5,68	5,44	5,78	6,47	Deporte
7	6	3,67	4,44	5,57	6,77	6,12	5,04	1,81	Deporte
8	7	4,89	4,82	6,50	6,37	7,69	6,79	5,36	Deporte
9	8	3,32	3,34	5,93	4,49	6,19	5,75	3,80	Deporte
10	9	4,21	4,43	6,06	8,17	6,53	8,44	3,24	Deporte
11	10	5,09	4,50	4,97	6,50	4,25	5,82	6,79	Deporte
12	11	6,02	5,36	4,67	5,78	6,00	6,69	6,99	Música
13	12	4,01	2,67	7,00	8,08	6,52	6,58	4,02	Deporte
14	13	6,60	6,35	5,49	6,98	7,28	6,06	5,27	Música
15	14	7,75	5,69	4,36	6,54	4,41	2,40	6,62	Musi-Dep

Figure 3: Calificaciones de los 15 primeros alumnos.

Lo primero que vamos a necesitar para empezar a trabajar, es tener todas las librerías necesarias cargadas en el entorno "prompt" de Anaconda, mediante el código "pip install nombre_librería" como se muestra a continuación.

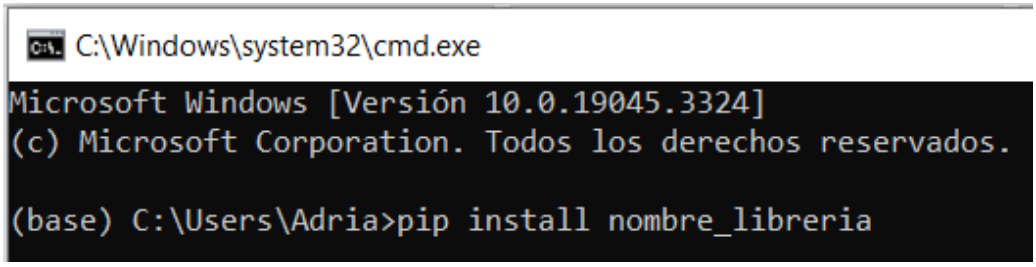

Figure 4: Instalar librerías en entorno pompt.

Una vez están cargadas las librerías en el sistema, debemos llamarlas en nuestro script. A continuación se muestran todas las librerías utilizadas para realizar un ACP completo junto con una breve descripción de su utilidad.

```
import os # Proporciona funciones para interactuar con el sistema operativo.
import pandas as pd # Manipulación y análisis de datos tabulares (filas y columnas).
import numpy as np # Operaciones numéricas y matriciales.
import seaborn as sns # Visualización estadística de datos.
import matplotlib.pyplot as plt # Creación de gráficos y visualizaciones.

# Matplotlib es una herramienta versátil para crear gráficos desde cero,
# mientras que Seaborn simplifica la creación de gráficos estadísticos.

from sklearn.decomposition import PCA # Implementación del Análisis de Componentes Principales (PCA).
from sklearn.preprocessing import StandardScaler # Estandarización de datos para análisis estadísticos.
```

Listing 1: Importación de librerías en Python.

Definimos nuestro entorno de trabajo utilizando la función "chdir" de la librería "os".

```
os.chdir('G:/Mi unidad/Apuntes titulo propio1/Datos') #Definimos nuestro
    entorno de trabajo.
```

Listing 2: Deinición entorno de trabajo.

Una vez cargadas las librerías y definido el entorno de trabajo, procedemos a la lectura en formato DataFrame de la base de datos, la cual, esta en formato tabla o tabulado. Para ello hacemos uso de la función *read_excel* de la librería *pandas*. Le asignamos el nombre de *notas*. Indicamos que la primera variable (*Alumno*) es el índice del DataFrame y posteriormente eliminamos dicha variable junto con la última ($EXTRA\_ESC$). Quedándonos así únicamente con las variables numéricas objeto del ACP.

```
# Cargar un archivo Excel llamado 'notas.xlsx' en un DataFrame llamado
    notas.
notas = pd.read_excel('notas.xlsx')

# Establecer la columna 'Alumno' como índice del DataFrame notas y
    eliminarla.
notas = notas.set_index('Alumno', drop=True)

# Quita 'EXTRA_ESC' y crea versión con sólo 2 decimales del data frame
notas = notas.drop('EXTRA_ESC', axis=1)
notas_2 = notas.round(decimals = 2)
notas_2.head()
```

Listing 3: Carga y preparación de los datos.

Antes de empezar con el ACP es conveniente realizar un análisis descriptivo de las variables, para observar si hay alguna anomalía en los datos, como datos fuera de rango, atípicos o perdidos.

```

# Cálculo de los estadísticos descriptivos.
# Genera una lista con los nombres de las variables.
variables = list(notas)

# Calcula los estadísticas descriptivas para cada variable y crea un
    DataFrame con los resultados.
estadisticos = pd.DataFrame({
    'Mínimo': notas[variables].min(),
    'Percentil 25': notas[variables].quantile(0.25),
    'Mediana': notas[variables].median(),
    'Percentil 75': notas[variables].quantile(0.75),
    'Media': notas[variables].mean(),
    'Máximo': notas[variables].max(),
    'Desviación Estándar': notas[variables].std(),
    'Varianza': notas[variables].var(),
```

```
    'Coeficiente de Variación': (notas[variables].std() / notas[variables].
        mean()),
    'Datos Perdidos': notas[variables].isna().sum()  # Cuenta los valores
        NaN por variable.
})
```

Listing 4: Estudio descriptivo de los datos.

Como podemos observar en la Figura 5 los datos no muestran ningún indicio de anomalías, contamos con todos los datos, y estos, están en los rangos esperados.

Índice	Mínimo	Percentil 25	Mediana	Percentil 75	Media	Máximo	Desviación Estándar	Varianza	Coeficiente de Variación	Datos Perdidos
Matematicas	1.65588	4.33087	5.26635	6.2365	5.30219	8.85618	1.46979	2.16028	0.277204	0
Fisica	0.211836	3.55401	4.50995	5.64319	4.63173	8.3716	1.51536	2.2963	0.327168	0
Lengua	2.96587	5.14468	6.04806	6.62505	5.97155	8.57129	1.27483	1.6252	0.213484	0
Ingles	3.27163	5.77339	6.7487	8.03896	6.80625	10.6239	1.47323	2.17041	0.216452	0
Historia	2.32689	5.55147	6.26843	7.13108	6.35473	9.41208	1.37323	1.88575	0.216095	0
Literatura	2.4029	5.61109	6.58599	7.5491	6.61477	10.4562	1.45661	2.1217	0.220205	0
Economia	1.64023	4.04546	5.07997	6.14481	5.1205	8.77921	1.50773	2.27324	0.294449	0

Figure 5: Descriptivos de las variables originales.

También es importante analizar las relaciones entre las variables, ya que si estas mostrasen fuerte independencia, no tendría sentido realizar un ACP. Para ello calculamos la matriz de correlaciones.

```
# Calcula la matriz de correlación entre las variables del DataFrame 'notas'.
R = notas.corr()

# Crea una nueva figura de dimensiones 10x8 para el gráfico.
plt.figure(figsize=(10, 8))

# Genera un mapa de calor (heatmap) de la matriz de correlación 'R' utilizando Seaborn.
# 'annot=True' agrega los valores de correlación en las celdas.
# 'cmap' establece el esquema de colores (en este caso, 'coolwarm' para colores fríos y cálidos).
# 'fmt' controla el formato de los números en las celdas ('.2f' para dos decimales).
# 'linewidths' establece el ancho de las líneas que separan las celdas.
sns.heatmap(R, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
```

Listing 5: Matriz de correlaciones y representación.

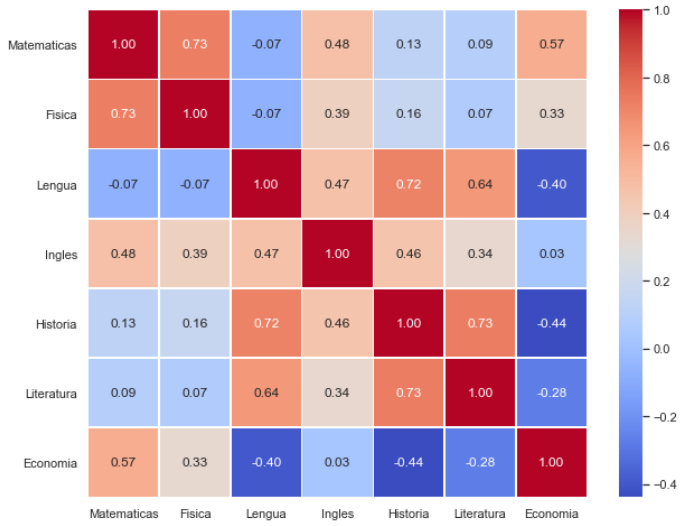

Figure 6: Gráfico de correlaciones de las variables originales.

Observemos que las variables con mayor correlación positiva son Lengua e Historia, Matemáticas y Física y también Historia y Literatura. Por otro lado, se observa una destacable correlación negativa entre las notas de Economía con las asignaturas de Lengua e Historia.

Ahora si, una vez cargados y validado los datos, y observado las *dependencias* entre las variables, estamos en disposición de empezar el ACP. Para ello, lo primero, estandarizamos los datos, que pese a estar en la misma escala y no ser imprescindible para realizar el análisis, siempre es aconsejable.

```
    # Estandarizamos los datos:
notas_std = StandardScaler().fit_transform(notas)
```

Listing 6: Estandarización de los datos.

Para llevar a cabo nuestro ACP vamos a hacer uso del objeto PCA de la librería scikit-learn. Este objeto proporciona una variedad de resultados y métodos que permiten explorar y comprender los componentes principales de un conjunto de datos, así como reducir su dimensionalidad. A continuación, se presenta una lista de las opciones de configuración, resultados y métodos disponibles en un objeto PCA, junto con una breve descripción de cada uno de ellos. Estos recursos son esenciales para realizar análisis y transformaciones de datos en el espacio de los componentes principales.

Opciones de Configuración:

- `n_components`: Número de componentes principales a calcular y mantener.
- `whiten`: Indica si queremos que la matriz de componentes principales tenga varianza unitaria.
  - `whiten=False`: (valor predeterminado), las varianzas de las componentes no se estandarizan, lo que significa que pueden tener varianzas diferentes.
  - `whiten=True`: Esto puede ser útil para simplificar la interpretación de las componentes y cuando se utilizan en algoritmos de clasificación o regresión.

- `svd_solver`: El algoritmo que PCA utilizará para calcular los componentes principales. La elección del `svd_solver` dependerá de la cantidad de datos y componentes que estés manejando.
    1. `'auto'` (Opción por defecto): Seleccionará automáticamente el método más adecuado en función del tamaño de los datos y `n_components`.
    2. `'randomized'`: Suele ser una buena opción cuando tienes un conjunto de datos grande y solo necesitas un subconjunto de componentes principales.
    3. `'full'`: Es adecuado para conjuntos de datos más pequeños o cuando necesitas calcular todos los componentes principales.
    4. `'arpack'`: Resulta útil cuando necesitas un número específico de componentes principales y deseas un equilibrio entre eficiencia computacional y precisión.

Resultados:

- `components_`: Devuelve los vectores de los componentes principales.
- `explained_variance_`: Devuelve los autovalores asociados a la matriz de correlaciones o covarianzas, los cuales, están ordenados en orden descendente.
- `explained_variance_ratio_`: Devuelve la proporción de la varianza total explicada por cada componente principal.
- `mean_`: Devuelve la media de cada variable original antes de la estandarización (si no se realizó).
- `n_components_`: Devuelve el número de componentes principales que se mantuvieron.
- `noise_variance_`: Proporciona una medida de cuánta información se ha perdido al reducir la dimensionalidad de los datos con PCA (solo disponible si se especifica `whiten=True`).
- `n_features_`: Devuelve el número de variables originales en los datos de entrada.

Métodos:

- `fit(X)`: Ajusta el modelo PCA a los datos de entrada `X`.
- `fit_transform(X)`: Ajusta el modelo PCA a los datos de entrada `X` y luego transforma `X` en el espacio de los componentes principales.
- `transform(X)`: Transforma los datos de entrada `X` en el espacio de los componentes principales, utilizando el modelo previamente ajustado.
- `inverse_transform(X_reduced)`: Realiza la inversa de la transformación PCA para llevar los datos desde el espacio de los componentes principales de nuevo al espacio original.

Procedemos a realizar el ACP a nuestro ejemplo mediante el siguiente código:

```
    # Crea una instancia de Análisis de Componentes Principales (ACP):
# - Utilizamos PCA(n_components=7) para crear un objeto PCA que realizará un análisis de componentes principales.
# - Establecemos n_components en 7 para retener el maximo de las componentes principales (maximo= numero de variables).
pca = PCA(n_components=7)

# Aplicar el Análisis de Componentes Principales (ACP) a los datos estandarizados:
# - Usamos pca.fit(notas_estandarizadas) para ajustar el modelo de ACP a los datos estandarizados.
fit = pca.fit(notas_estandarizadas)
```

Listing 7: Aplicación del ACP.

A continuación, procedemos a calcular los autovalores asociados a las componentes principales (valores λ que resuelven la ecuación $\det(\Sigma - \lambda I) = 0$), la varianza explicada y la varianza acumulada de las mismas. Así como su representación en una tabla.

```
    # Obtener los autovalores asociados a cada componente principal.
autovalores = fit.explained_variance_

# Obtener la varianza explicada por cada componente principal como un porcentaje de la varianza total.
var_explicada = fit.explained_variance_ratio_

# Calcular la varianza explicada acumulada a medida que se agregan cada componente principal.
var_acumulada = np.cumsum(var_explicada)

# Crear un DataFrame de pandas con los datos anteriores y establecer índice.
data = {'Autovalores': autovalores, 'Variabilidad Explicada': var_explicada, 'Variabilidad Acumulada': var_acumulada}
tabla = pd.DataFrame(data, index=['Componente {}'.format(i) for i in range(1, fit.n_components_+1)])
```

Listing 8: Cálculo de los autovalores/variabilidad explicada.

Como podemos observar en la Figura 7, usando las dos primeras componentes conseguiremos explicar el 74.9% de la variabilidad total de las variables originales. La decisión de cuántas componentes seleccionar es subjetiva y se debe decidir teniendo en cuenta dos factores, por un lado, la cantidad de variabilidad que deseamos explicar y por otro, la dimensión que queremos conseguir. En este ejemplo, consideramos que añadiendo una tercera componente no se consigue *demasiado* beneficio en cuanto a la cantidad de variabilidad explicada. No obstante, esta decisión podría ser diferente, si por ejemplo, quisiéramos explicar un mínimo superior de variabilidad.

Índice	Autovalores	Variabilidad Explicada	Variabilidad Acumulada
Componente 1	2.94116	41.6664	41.6664
Componente 2	2.34795	33.2626	74.929
Componente 3	0.574517	8.13899	83.068
Componente 4	0.561622	7.95632	91.0243
Componente 5	0.265562	3.76214	94.7865
Componente 6	0.220849	3.1287	97.9152
Componente 7	0.147164	2.08482	100

Figure 7: Gráfico de correlaciones de las variables originales.

Realizamos la representación de la variabilidad explicada, para observar gráficamente como esta se estabiliza una vez seleccionadas las dos primeras componentes.

```
# Representacion de la variabilidad explicada:
  #Creamos la función plot_varianza_explicada
    #Args:
      #var_explicada (array): Un array que contiene el porcentaje de
          varianza explicada
        #por cada componente principal. Generalmente calculado como
        #var_explicada = fit.explained_variance_ratio_ * 100.
      #n_components (int): El número total de componentes principales.
        #Generalmente calculado como fit.n_components.

def plot_varianza_explicada(var_explicada, n_components):
    # Crear un rango de números de componentes principales de 1 a
       n_components
    num_componentes_range = np.arange(1, n_components + 1)

    # Crear una figura de dimensiones 8x6
    plt.figure(figsize=(8, 6))

    # Trazar la varianza explicada en función del número de componentes
       principales
    plt.plot(num_componentes_range, var_explicada, marker='o')

    # Etiquetas de los ejes x e y
    plt.xlabel('Número de Componentes Principales')
    plt.ylabel('Varianza Explicada')

    # Título del gráfico
    plt.title('Variabilidad Explicada por Componente Principal')

    # Establecer las marcas en el eje x para que coincidan con el número de
       componentes
    plt.xticks(num_componentes_range)

    # Mostrar una cuadrícula en el gráfico
```

```
    plt.grid(True)

    # Agregar barras debajo de cada punto para representar el porcentaje de
        variabilidad explicada
    # - 'width': Ancho de las barras de la barra. En este caso, se
        establece en 0.2 unidades.
    # - 'align': Alineación de las barras con respecto a los puntos en el
        eje x.
    #   'center' significa que las barras estarán centradas debajo de los
        puntos.
    # - 'alpha': Transparencia de las barras. Un valor de 0.7 significa que
        las barras son 70% transparentes.
    plt.bar(num_componentes_range, var_explicada, width=0.2, align='center'
        , alpha=0.7)

    # Mostrar el gráfico
    plt.show()

plot_varianza_explicada(var_explicada, fit.n_components_)
```

Listing 9: Representación de la variabilidad explicada por Componentes Principales.

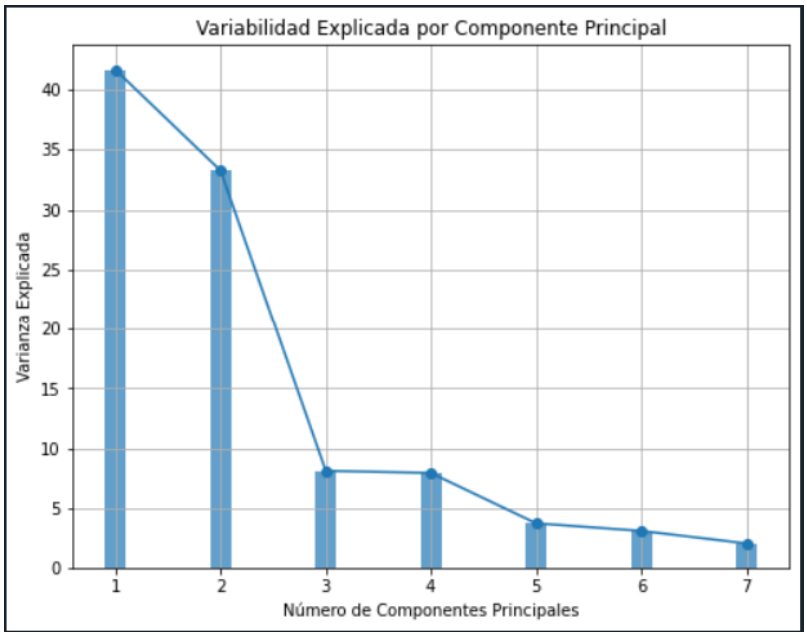

Figure 8: Varianza explicada.

Podemos observar que la variabilidad explicada por cada componente, decae y se estabiliza a partir de la tercera. Por lo tanto, optamos por retener únicamente dos de las siete componentes principales, las cuales explican conjuntamente el 74.91% de la varianza de las variables originales. Los

dos autovalores correspondientes a estas componentes seleccionadas son 2.94 y 2.35, representando el 41.7% y el 33.3% de la variabilidad, respectivamente.

Una vez hemos decidido quedarnos con las dos primeras componentes, procedemos a calcularlas. Para ello necesitamos conocer la matriz de transformación formada por los autovectores o vectores propios asociados a los autovalores seleccionados.

```
# Crea una instancia de ACP con las dos primeras componentes que nos
    interesan y aplicar a los datos.
pca = PCA(n_components=2)
fit = pca.fit(notas_estandarizadas)

# Obtener los autovectores asociados a cada componente principal y
    transponerlos.
autovectores = pd.DataFrame(pca.components_.T,
                            columns = ['Autovector {}'.format(i) for i in
                                range(1, fit.n_components_+1)],
                            index = ['{}_z'.format(variable) for variable
                                in variables])
```

Listing 10: Cálculo de los autovalores/variabilidad explicada.

Transponemos los autovectores para tenerlos por columnas e indicamos que las filas están representadas por las variables originales tipificadas Z.

Índice	Autovector 1	Autovector 2
Matematicas_z	-0.117466	-0.602636
Fisica_z	-0.128305	-0.53538
Lengua_z	-0.496457	0.162045
Ingles_z	-0.388664	-0.293545
Historia_z	-0.534553	0.0568696
Literatura_z	-0.482633	0.0657072
Economia_z	0.231363	-0.479813

Figure 9: Matriz de transformación $W = [w_1, w_2]$.

Una vez tenemos nuestros datos tipificados Z y nuestra matriz de transformación W procedemos a calcular las componentes principales CP tales que $CP = ZW$, es decir,

$$CP_1 = -0.117Mat_z - 0.128Fis_z - 0.496Len_z - 0.388Ing_z - 0.534His_z - 0.482Lit_z + 0.231Eco_z$$

$$CP_2 = -0.602Mat_z - 0.535Fis_z + 0.162Len_z - 0.293Ing_z + 0.056His_z + 0.065Lit_z - 0.479Eco_z.$$

```
# Crea una instancia de ACP con las dos primeras componentes que nos
    interesan y aplicar a los datos.
pca = PCA(n_components=2)
fit = pca.fit(notas_estandarizadas)
```

```
4
5  # Obtener los autovectores asociados a cada componente principal y
       transponerlos.
6  autovectores = pd.DataFrame(pca.components_.T,
7                              columns = ['Autovector {}'.format(i) for i in
                                   range(1, fit.n_components_+1)],
8                              index = ['{}_z'.format(variable) for variable
                                   in variables]
9
10 resultados_pca = pd.DataFrame(fit.transform(notas_estandarizadas),
11                                columns=['Componente {}'.format(i) for i in
                                      range(1, fit.n_components_+1)],
12                                index=notas_estandarizadas.index)
13
14
15 notas_z_cp = pd.concat([notas_estandarizadas, resultados_pca], axis=1)
16 variables_cp = notas_z_cp.columns
```

Listing 11: Cálculo de las componentes principales.

Se muestra a continuación, en la Figura 10, la base de datos estandarizadas (5 primeras observaciones) con las dos componentes seleccionadas indexadas.

Alumno	Matematicas_z	Fisica_z	Lengua_z	Ingles_z	Historia_z	Literatura_z	Economia_z	Componente 1	Componente 2
1	1.98479	1.2881	0.412892	0.4168	2.13099	0.979363	0.454951	-2.27193	-1.97392
2	0.0463853	-0.462778	0.0538108	-0.501058	-0.119002	0.483532	-0.531895	-0.0708595	0.655825
3	-0.627619	-1.08649	0.357885	-0.381331	-0.699891	0.0874642	0.743655	0.687632	0.738973
4	1.59566	0.782447	1.61198	0.862701	0.675615	1.6637	0.809081	-2.40033	-1.61301
5	0.0571131	0.747611	-0.598737	-0.764955	-0.665394	-0.578819	0.901758	1.33561	-0.815696

Figure 10: Primeras 5 observaciones con las Componentes Principales seleccionadas.

Este es el momento en el que podríamos *eliminar* nuestras variables originales y trabajar con las componentes principales calculadas, por ejemplo, en objetivos más *avanzados* en el análisis de datos como es el caso de la *predicción*. Hemos entonces, reducido la dimensionalidad de nuestra base de datos original, sustituyendo un total de 7 variables por tan solo dos componentes, las cuales, combinadas, explican aproximadamente un 75% de la variabilidad original.

Vamos ahora a estudiar cómo de fuerte (o débil) es la relación entre las componentes principales seleccionadas y cada una de las variables originales. Para ello calculamos la correlación entre las variables y las componentes retenidas.

```
1
2  # Cálculo de las correlaciones entre las variables originales y las
       componentes seleccionadas.
3  # Guardamos el nombre de las variables del archivo conjunto (variables y
       componentes).
4  variables_cp = notas_z_cp.columns
5
```

```
# Calculamos las correlaciones y seleccionamos las que nos interesan (
    variables contra componentes).
correlacion = pd.DataFrame(np.corrcoef(notas_estandarizadas.T,
    resultados_pca.T),
                           index = variables_cp, columns = variables_cp)

n_variables = fit.n_features_
correlaciones_notas_con_cp = correlacion.iloc[:fit.n_features_, fit.
    n_features_:]
```

Listing 12: Correlación entre variables y componentes.

Como podemos observar en la Figura 11 la primera componente principal (CP_1) muestra una correlación negativa significativa con las variables Lengua, Historia y Literatura. Esto sugiere que estas variables están fuertemente influenciadas por CP_1 y, por lo tanto, están representadas de manera directa por esta componente. Por otro lado, la Componente Principal 2 (CP_2) presenta una correlación notable solo con Matemáticas, Física y Economía, lo que indica que estas variables están predominantemente influenciadas por CP_2. Además, es interesante observar que la nota en inglés está correlacionada con ambas componentes principales, lo que sugiere cierta influencia mixta de ambas CP_1 y CP_2 en esta variable.

Índice	Componente 1	Componente 2
Matematicas_z	-0.200611	-0.919564
Fisica_z	-0.219122	-0.816938
Lengua_z	-0.847859	0.247265
Ingles_z	-0.663768	-0.447922
Historia_z	-0.912921	0.0867775
Literatura_z	-0.82425	0.100263
Economia_z	0.395127	-0.732148

Figure 11: Correlaciones entre variables y componentes.

Podemos comprobar que la correlación entre cada par *(Componente, variable)* se puede calcular como

$$Corr(CP_i, x_j) = w_{i,j}\frac{\sqrt{\lambda_i}}{s_j}, \text{ para i=1,2 y para j=1,...,7.}$$

Por ejemplo, la correlación entre la asignatura de Física y la Componente 1 sería:

$$Corr(CP_1, x_2) = w_{1,2}\frac{\sqrt{\lambda_1}}{s_2} = -0.128\frac{\sqrt{2.941}}{1} = -0.219.$$

Si multiplicamos estas correlaciones al cuadrado, conseguimos los valores conocidos como *cosenos al cuadrado* que representan la proporción de variabilidad que cada componente explica de cada variable.

```
cos2 = correlaciones_notas_con_cp **2
```

Listing 13: cos2.

En la figura 12 podemos observar, por ejemplo, que la variable Matemáticas es la que más explicada esta por el conjunto de las dos componentes, mientras que ingles es la que menos.

$$Corr(CP_1, Mat)^2 + Corr(CP_2, Mat)^2 = 0.04 + 0.84 = 0.885,$$

$$Corr(CP_1, Ing)^2 + Corr(CP_2, Ing)^2 = 0.44 + 0.20 = 0.64.$$

Índice	Componente 1	Componente 2
Matematicas_z	0.0402447	0.845598
Fisica_z	0.0480146	0.667387
Lengua_z	0.718864	0.0611399
Ingles_z	0.440588	0.200634
Historia_z	0.833424	0.00753034
Literatura_z	0.679388	0.0100527
Economia_z	0.156125	0.536041

Figure 12: Cos^2

Aunque para comparar la variabilidad explicada por las componentes entre el conjunto de variables, es más *visual* el siguiente gráfico de barras, donde se representa la suma de los cos^2 para cada variable.

```


def plot_cos2_bars(cos2):
    """
    Genera un gráfico de barras para representar la varianza explicada de
        cada variable utilizando los cuadrados de las cargas (cos^2).

    Args:
        cos2 (pd.DataFrame): DataFrame que contiene los cuadrados de las
            cargas de las variables en las componentes principales.

    Returns:
```

```
        None
    """
    # Crea una figura de dimensiones 8x6 pulgadas para el gráfico
    plt.figure(figsize=(8, 6))

    # Crea un gráfico de barras para representar la varianza explicada por
        cada variable
    sns.barplot(x=cos2.sum(axis=1), y=cos2.index, color="blue")

    # Etiqueta los ejes
    plt.xlabel('Suma de los cos^2')
    plt.ylabel('Variables')

    # Establece el título del gráfico
    plt.title('Varianza Explicada de cada Variable por las Componentes
        Principales')

    # Muestra el gráfico
    plt.show()


plot_cos2_bars(cos2)
```

Listing 14: Gráfico de barras para los $\cos^2$.

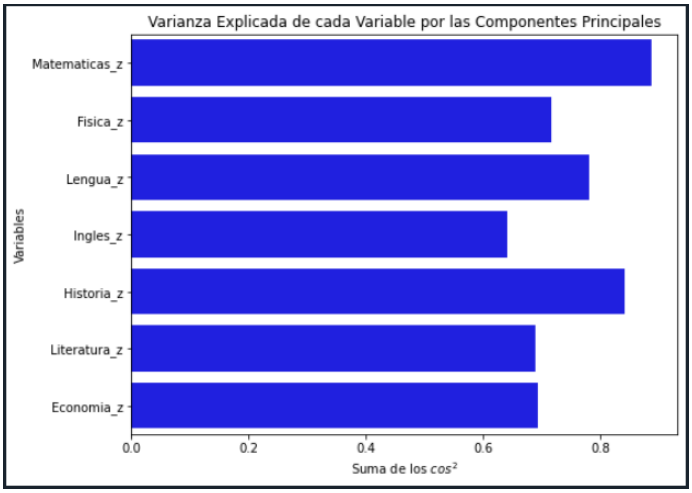

Figure 13: Gráfico de barras para la suma de los cos^2.

Con la siguiente función *plot_corr_cos* generamos diferentes gráficos (en este caso particular solo 1), donde se representan las correlaciones entre cada variable y cada par de componentes principales (ejes). El color de cada vector en los gráficos indica el valor de los cosenos al cuadrado.

```

```

```
def plot_corr_cos(n_components, correlaciones_notas_con_cp):

        """
    Genera un graficos en los que se representa un vector por cada variable, usando como ejes las compoenntes, la orientación y la longitud del vector representa la correlación
    entre cada variable y dos de las componentes. El color representa el valor de la suma de los cosenos al cuadrado
    Args:
        n_components (int): Número entero que representa el número de componentes principales seleccionadas.
        correlaciones_notas_con_cp (DataFrame): DataFrame que contiene la matriz de coorrelaciones entre variables y componentes

    """
    # Definir un mapa de color (cmap) sensible a las diferencias numéricas
    cmap = plt.get_cmap('coolwarm')  # Puedes ajustar el cmap según tus preferencias

    for i in range(n_components):
        for j in range(i + 1, n_components):  # Evitar pares duplicados
            # Calcular la suma de los cosenos al cuadrado
            sum_cos2 = correlaciones_notas_con_cp.iloc[:, i] ** 2 + correlaciones_notas_con_cp.iloc[:, j] ** 2

            # Crear un nuevo gráfico para cada par de componentes principales
            plt.figure(figsize=(10, 10))

            # Dibujar un círculo de radio 1
            circle = plt.Circle((0, 0), 1, fill=False, color='b', linestyle='dotted')

            plt.gca().add_patch(circle)

            # Dibujar vectores para cada variable con colores basados en la suma de los cosenos al cuadrado
            for k, var_name in enumerate(correlaciones_notas_con_cp.index):
                x = correlaciones_notas_con_cp.iloc[k, i]  # Correlación en la primera dimensión
                y = correlaciones_notas_con_cp.iloc[k, j]  # Correlación en la segunda dimensión

                # Seleccionar un color de acuerdo a la suma de los cosenos al cuadrado
                color = cmap(sum_cos2[k])

                # Dibujar el vector con el color seleccionado
                plt.quiver(0, 0, x, y, angles='xy', scale_units='xy', scale=1, color=color)
```

```
38
39                  # Agregar el nombre de la variable junto a la flecha con el
                        mismo color
40                  plt.text(x, y, var_name, color=color, fontsize=12, ha='
                        right', va='bottom')
41
42              # Dibujar líneas discontinuas que representen los ejes
43              plt.axhline(0, color='black', linestyle='--', linewidth=0.8)
44              plt.axvline(0, color='black', linestyle='--', linewidth=0.8)
45
46              # Etiquetar los ejes
47              plt.xlabel(f'Componente Principal {i + 1}')
48              plt.ylabel(f'Componente Principal {j + 1}')
49
50              # Establecer los límites del gráfico
51              plt.xlim(-1.1, 1.1)
52              plt.ylim(-1.1, 1.1)
53
54              # Agregar un mapa de color (colorbar) y su leyenda
55              sm = plt.cm.ScalarMappable(cmap=cmap)
56              sm.set_array([])  # Evita errores de escala
57              plt.colorbar(sm, orientation='vertical', label='cos^2')  #
                    Agrega la leyenda
58
59              # Mostrar el gráfico
60              plt.grid()
61              plt.show()
62
63  plot_corr_cos(fit.n_components, correlaciones_notas_con_cp)
```

Listing 15: Grafico de las correlaciones entre variables y componentes.

Como se enunciaba anteriormente, en el gráfico de la Figura 14 se representa un vector por cada asignatura, siendo los ejes cada una de las componentes seleccionadas, te tal forma que que longitud del vector en cada uno de los ejes representa la correlación en dicha componente, el color del vector es la suma de estas correlaciones al cuadrado (cos^2). En este gráfico podemos visualizar lo ofrecido por la Figura 11, como las asignaturas de *'letras'* están fuertemente relacionadas con la segunda componente, las de *'ciencias'* con la primera e Ingles medianamente con ambas. Pero además podemos observar que las variables con el color *rojo* más intenso son Matemáticas e Historia, lo cual indica que son las variables mejor explicadas por el conjunto de componentes.

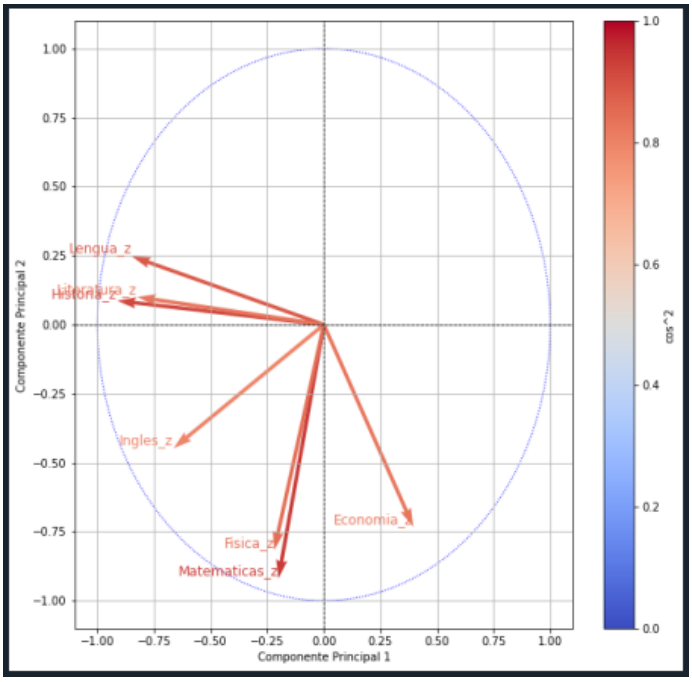

Figure 14: Correlaciones entre Variables y Componentes Principales.

Procedemos a continuación a calcular las contribuciones normalizadas de cada variable en la construcción de cada una de las componentes. Representamos los resultados mediante un *mapa de calor* donde el valor de las contribuciones dependerá de la intensidad del color azul utilizado.

```
1
2  def plot_contribuciones_proporcionales(cos2, autovalores, n_components):
3      """
4      Cacula las contribuciones de cada variable a las componentes
           principales y
5      Genera un gráfico de mapa de calor con los datos
6      Args:
7          cos2 (DataFrame): DataFrame de los cuadrados de las cargas (cos^2).
8          autovalores (array): Array de los autovalores asociados a las
               componentes principales.
9          n_components (int): Número de componentes principales seleccionadas
               .
10     """
11     # Calcula las contribuciones multiplicando cos2 por la raíz cuadrada de
           los autovalores
12     contribuciones = cos2 * np.sqrt(autovalores)
13
14     # Inicializa una lista para las sumas de contribuciones
15     sumas_contribuciones = []
16
```

```
    # Calcula la suma de las contribuciones para cada componente principal
    for i in range(n_components):
        nombre_componente = f'Componente {i + 1}'
        suma_contribucion = np.sum(contribuciones[nombre_componente])
        sumas_contribuciones.append(suma_contribucion)

    # Calcula las contribuciones proporcionales dividiendo por las sumas de contribuciones
    contribuciones_proporcionales = contribuciones.div(sumas_contribuciones, axis=1) * 100

    # Crea una figura de dimensiones 8x8 pulgadas para el gráfico
    plt.figure(figsize=(8, 8))

    # Utiliza un mapa de calor (heatmap) para visualizar las contribuciones proporcionales
    sns.heatmap(contribuciones_proporcionales, cmap='Blues', linewidths=0.5, annot=False)

    # Etiqueta los ejes (puedes personalizar los nombres de las filas y columnas si es necesario)
    plt.xlabel('Componentes Principales')
    plt.ylabel('Variables')

    # Establece el título del gráfico
    plt.title('Contribuciones Proporcionales de las Variables en las Componentes Principales')

    # Muestra el gráfico
    plt.show()

    # Devuelve los DataFrames de contribuciones y contribuciones proporcionales
    return contribuciones_proporcionales

contribuciones_proporcionales = plot_contribuciones_proporcionales(cos2, autovalores,fit.n_components)
```

Listing 16: Contribuciones de las variables a la construcción de las CP.

En la Figura 15 volvemos a corroborar lo ya observado anteriormente, las variables de *letras*, participan fuertemente en la construcción de la primera Componente, las de *ciencias* en la segunda, e ingles medianamente en ambas.

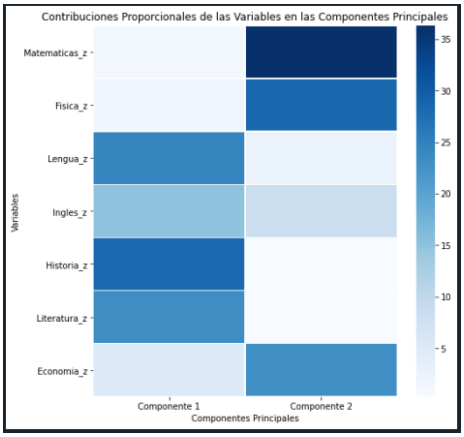

Figure 15: Contribuciones de las Variables en las Componentes Principales.

3.1 Coordenadas de los individuos en los nuevos ejes

Por último, vamos a calcular las nuevas coordenadas (o valores) de cada uno de los individuos en los nuevos ejes (Componentes Principales), para ello hacemos uso de nuestra matriz X o Z donde tenemos nuestras variables o variables tipificadas y de la matriz de transformación W. De tal manera que el valor de la $i-esima$ observación en la $\alpha-esima$ componente, se calcula como:

$$CP_{\alpha,i} = \sum_{j=1}^{p} z_{i,j} w_{\alpha,j} = \sum_{j=1}^{p} \frac{x_{i,j} - \bar{x}_j}{S_j} w_{\alpha,j}.$$

Como las variables están estandarizadas, es sencillo comprobar que la suma de los valores de los n individuos sobre cada Componente es nula,

$$\sum_{i=1}^{n} CP_{\alpha,i} = \sum_{i=1}^{n} \sum_{j=1}^{p} \frac{x_{i,j} - \bar{x}_j}{S_j} w_{\alpha,j} = \sum_{i=1}^{n} w_{\alpha,j} \sum_{j=1}^{p} \frac{x_{i,j} - \bar{x}_j}{S_j} = 0,$$

por lo que la media de las Componentes es cero y su varianza λ_α .

Continuando con el ejemplo, tenemos en la siguiente figura, los valores de los 10 primeros alumnos en las Componentes Principales.

```

# Indexamos las componentes principales a la base de datos original.
notas_cp = pd.concat([notas, extra, resultados_pca], axis=1)
```

Listing 17: Base de datos original más CP.

Recordamos que los valores de las medias y desviaciones estándar de las variables originales son:

Alumno	Matematicas	Fisica	Lengua	Ingles	Historia	Literatura	Economia	EXTRA_ESC	Componente 1	Componente 2
1	8.20723	6.57551	6.49572	7.41773	9.26885	8.03536	5.80358	Musi-Dep	-2.27193	-1.97392
2	5.37009	3.93339	6.03986	6.07116	6.192	7.31615	4.32189	Deporte	-0.0708595	0.655825
3	4.38358	2.99218	6.42589	6.24681	5.39764	6.74164	6.23704	Deporte	0.687632	0.738973
4	7.63769	5.81247	8.01798	8.0719	7.27863	9.02802	6.33528	Musi-Dep	-2.40033	-1.61301
5	5.38579	5.7599	5.21145	5.684	5.44481	5.77518	6.47443	Deporte	1.33561	-0.815696
6	3.67495	4.4407	5.57038	6.76995	6.11794	5.03942	1.81466	Deporte	0.420651	1.66906
7	4.88547	4.8183	6.49907	6.37287	7.69025	6.79006	5.36362	Deporte	-0.616813	0.245222
8	3.3211	3.33974	5.93082	4.49104	6.1948	5.74971	3.79899	Deporte	1.04484	2.10859
9	4.21363	4.42519	6.05626	8.16895	6.52674	8.4445	3.2434	Deporte	-1.25451	0.949534
10	5.08802	4.49552	4.97355	6.50225	4.24672	5.8174	6.79072	Deporte	1.8463	-0.587592

Figure 16: Contribuciones de las Variables en las Componentes Principales.

Luego, el valor que toma el primer alumno en la Primera Componente, viene dado por:

$$CP_{11} = -\frac{8.2-5.30}{1.47}0.117 - \frac{6.57-4.63}{1.52}0.128 - \frac{6.49-5.97}{1.27}0.496 - \frac{7.41-6.81}{1.47}0.389$$
$$-\frac{9.26-6.35}{1.37}0.534 - \frac{8.03-6.61}{1.46}0.483 + \frac{5.80-5.12}{1.46}0.231 = -2.27.$$

A continuación vamos a representar a los individuos utilizando como ejes cada par de Componentes principales (en este ejemplo solo 1).

```
1
2  def plot_pca_scatter(pca, datos_estandarizados, n_components):
3      """
4      Genera gráficos de dispersión de observaciones en pares de componentes
           principales seleccionados.
5
6      Args:
7          pca (PCA): Objeto PCA previamente ajustado.
8          datos_estandarizados (pd.DataFrame): DataFrame de datos
               estandarizados.
9          n_components (int): Número de componentes principales seleccionadas
               .
10     """
11     # Representamos las observaciones en cada par de componentes
           seleccionadas
12     componentes_principales = pca.transform(datos_estandarizados)
13
14     for i in range(n_components):
15         for j in range(i + 1, n_components):  # Evitar pares duplicados
16             # Calcular la suma de los valores al cuadrado para cada
                   variable
17             # Crea un gráfico de dispersión de las observaciones en las dos
                    primeras componentes principales
18             plt.figure(figsize=(8, 6))  # Ajusta la dimension de la figura
                   si es necesario
```

```
            plt.scatter(componentes_principales[:, i],
                componentes_principales[:, j])

            # Agrega etiquetas a las observaciones
            etiquetas_de_observaciones = list(datos_estandarizados.index)

            for k, label in enumerate(etiquetas_de_observaciones):
                plt.annotate(label, (componentes_principales[k, i],
                    componentes_principales[k, j]))

            # Dibujar líneas discontinuas que representen los ejes
            plt.axhline(0, color='black', linestyle='--', linewidth=0.8)
            plt.axvline(0, color='black', linestyle='--', linewidth=0.8)

            # Etiquetar los ejes
            plt.xlabel(f'Componente Principal {i + 1}')
            plt.ylabel(f'Componente Principal {j + 1}')

            # Establece el título del gráfico
            plt.title('Gráfico de Dispersión de Observaciones en PCA')

            plt.show()

plot_pca_scatter(pca, notas_estandarizadas, fit.n_components)
```

Listing 18: Gráfico de dispersión de los individuos por cada par de CPs.

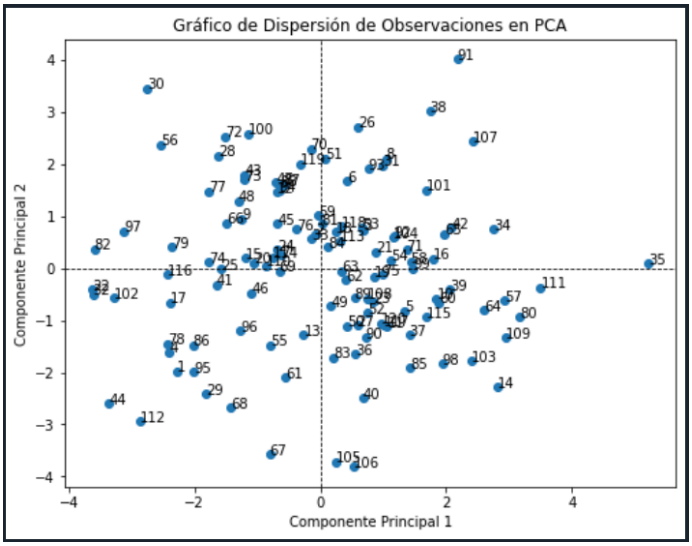

Figure 17: Dispersión de los individuos en los nuevos ejes.

Podemos modificar el código anterior para que el gráfico nos inlcuya un vector que represente

para cada una de las variables la correlación escalada con las componentes (ejes) del gráfico.

```
1
2
3  def plot_pca_scatter_with_vectors(pca, datos_estandarizados, n_components,
       components_):
4      """
5      Genera gráficos de dispersión de observaciones en pares de componentes
           principales seleccionados
6      con vectores de las correlaciones escaladas entre variables y
           componentes
7
8      Args:
9          pca (PCA): Objeto PCA previamente ajustado.
10         datos_estandarizados (pd.DataFrame): DataFrame de datos
               estandarizados.
11         n_components (int): Número de componentes principales seleccionadas
               .
12         components_: Array con las componentes.
13     """
14     # Representamos las observaciones en cada par de componentes
           seleccionadas
15     componentes_principales = pca.transform(datos_estandarizados)
16
17     for i in range(n_components):
18         for j in range(i + 1, n_components):  # Evitar pares duplicados
19             # Calcular la suma de los valores al cuadrado para cada
                   variable
20             # Crea un gráfico de dispersión de las observaciones en las dos
                    primeras componentes principales
21             plt.figure(figsize=(8, 6))  # Ajusta el tama o de la figura si
                    es necesario
22             plt.scatter(componentes_principales[:, i],
                   componentes_principales[:, j])
23
24             # A ade etiquetas a las observaciones
25             etiquetas_de_observaciones = list(datos_estandarizados.index)
26
27             for k, label in enumerate(etiquetas_de_observaciones):
28                 plt.annotate(label, (componentes_principales[k, i],
                       componentes_principales[k, j]))
29
30             # Dibujar líneas discontinuas que representen los ejes
31             plt.axhline(0, color='black', linestyle='--', linewidth=0.8)
32             plt.axvline(0, color='black', linestyle='--', linewidth=0.8)
33
34             # Etiquetar los ejes
35             plt.xlabel(f'Componente Principal {i + 1}')
36             plt.ylabel(f'Componente Principal {j + 1}')
37
38             # Establece el título del gráfico
```

```
            plt.title('Gráfico de Dispersión de Observaciones y variables
                en PCA')


            # A adimos vectores que representen las correlaciones
                escaladas entre variables y componentes
            coeff = np.transpose(fit.components_)
            scaled_coeff = 8 * coeff  #8 = escalado utilizado, ajustar en
                función del ejemplo
            for var_idx in range(scaled_coeff.shape[0]):
                plt.arrow(0, 0, scaled_coeff[var_idx, i], scaled_coeff[
                    var_idx, j], color='red', alpha=0.5)
                plt.text(scaled_coeff[var_idx, i], scaled_coeff[var_idx, j
                    ],
                      notas_estandarizadas.columns[var_idx], color='red', ha
                          ='center', va='center')


            plt.show()

plot_pca_scatter_with_vectors(pca, notas_estandarizadas, fit.n_components,
    fit.components_)
```

Listing 19: Gráfico de dispersión de los individuos y variables por cada par de CPs.

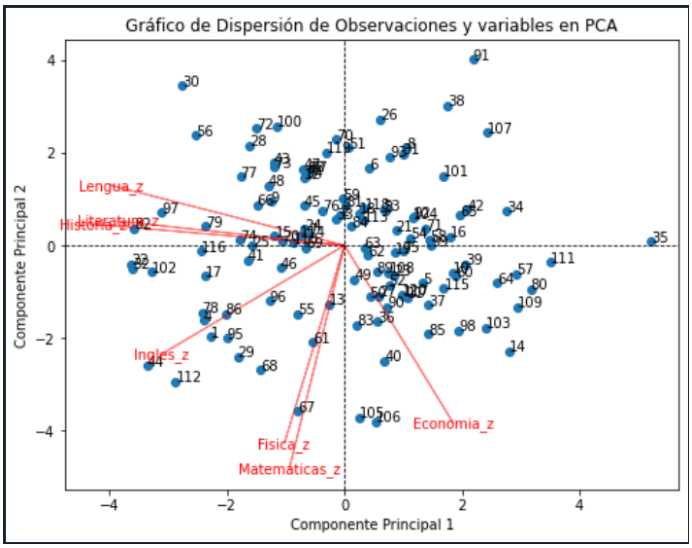

Figure 18: Dispersión de los individuos en los nuevos ejes.

Observemos, por ejemplo, que los alumnos 67 y 105 deben tener buenas notas en las asignaturas de *ciencias* mientras que en letras tienen notas sobre la media (su coordenada en la CP_1 es casi cero). Destacamos también al alumno 35, que debe tener muy malas notas en las signaturas de *letras*.

3.2 Variables e individuos suplementarios

El ACP a menudo se utiliza para resumir y comprender las relaciones entre variables en un conjunto de datos. Sin embargo, en situaciones del mundo real, es común que surjan nuevos elementos o variables que no estaban presentes en el conjunto de datos original. Estos nuevos elementos se conocen como 'individuos suplementarios' cuando se refieren a observaciones, y 'variables suplementarias' cuando se refieren a características adicionales.

En nuestro ejemplo, en el que hemos realizado un ACP para comprender las notas escolares de los estudiantes, en un momento posterior podría incorporarse un nuevo estudiante, o decidir medir o incorporar una nueva variable que no estaba presente inicialmente en nuestro conjunto de datos. Estos nuevos elementos o variables deben ser incorporados en nuestro análisis de ACP existente para evaluar su relación con las componentes principales ya calculadas.

La inclusión de individuos y variables suplementarias en el ACP nos permite evaluar cómo se sitúan en relación con la estructura existente de las componentes principales y cómo contribuyen a la variabilidad general. Esto es esencial para mantener nuestro análisis actualizado y relevante en entornos en constante cambio.

Para aplicar esto a nuestro ejemplo particular vamos a utilizar los datos de 4 nuevos alumnos, los cuales están en el archivo *'NOTAS_S.xlsx'*

	A	B	C	D	E	F	G	H	I
1	Alumno	Matematicas	Fisica	Lengua	Ingles	Historia	Literatura	Economia	EXTRA_ESC
2	121	6,8	5,7	5,6	7,6	8,3	7,7	5,8	Deporte
3	122	7,5	6,5	4,5	6,5	3,6	4,8	7,3	Música
4	123	4,7	3,2	3,5	5,4	4,6	5,6	5,3	Deporte
5	124	8,5	7,8	6,7	8,9	6,7	8,8	9,5	Musi-Dep

Figure 19: Individuos suplementarios.

Para tratar a estos nuevos estudiantes, debemos estandarizar sus variables, utilizando la media y la desviación estándar del conjunto de datos original. Posteriormente, los añadimos a nuestra base de datos estandarizada.

```
# Cargar un archivo Excel llamado 'notas.xlsx' en un DataFrame llamado
    notas.
notas_S = pd.read_excel('notas_S.xlsx')

# Establecer la columna 'Alumno' como índice del DataFrame notas y
    eliminarla.
notas_S = notas_S.set_index('Alumno', drop=True)

# Guarda la variable el indice y 'EXTRA_ESC' en un dataframe
extra_S = notas_S.iloc[:, [7]]

# Elimina la variable 'EXTRA_ESC' del DataFrame 'notas'.
notas_S = notas_S.drop(notas_S.columns[7], axis=1)

# Calcular la media y la desviación estándar de 'notas'
media_notas = notas.mean()
desviacion_estandar_notas = notas.std()

```

```
# Estandarizar 'notas_S' utilizando la media y la desviación estándar de 'notas'
notas_S_estandarizadas = pd.DataFrame(((notas_S - media_notas) / desviacion_estandar_notas))

notas_S_estandarizadas.columns = ['{}_z'.format(variable) for variable in variables]

# Agregar las observaciones estandarizadas a 'notas'
notas_sup = pd.concat([notas_estandarizadas, notas_S_estandarizadas])
```

Listing 20: Carga y tratamiento de observaciones complementarias.

Una vez tenemos las observaciones complementarias estandarizadas y agregadas a nuestra base de datos, procedemos a calcular sus CP.

```

# Calcular las componentes principales para el conjunto de datos combinado
componentes_principales_sup = pca.transform(notas_sup)

# Calcular las componentes principales para el conjunto de datos combinado
# y renombra las componentes
resultados_pca_sup = pd.DataFrame(fit.transform(notas_sup),
                                  columns=['Componente {}'.format(i) for i in range(1, fit.n_components_+1)],
                                  index=notas_sup.index)
```

Listing 21: Cálculo de las CP para las observaciones complementarias.

Podríamos representar a estos nuevos alumnos en los nuevos ejes, de la misma manera que lo hicimos sin ellos.

```

# Representacion observaciones + suplementarios
plot_pca_scatter(pca, notas_sup, fit.n_components)
```

Listing 22: Representacion observaciones + suplementarios.

A partir de los observado en la Figura 20 podemos intuir, por ejemplo, que la observación 124 ha sacado notas (en general) por encima de la media, tanto en *letras* como en *ciencias*. Todo lo contrario esperaríamos de la observación o alumno 123.

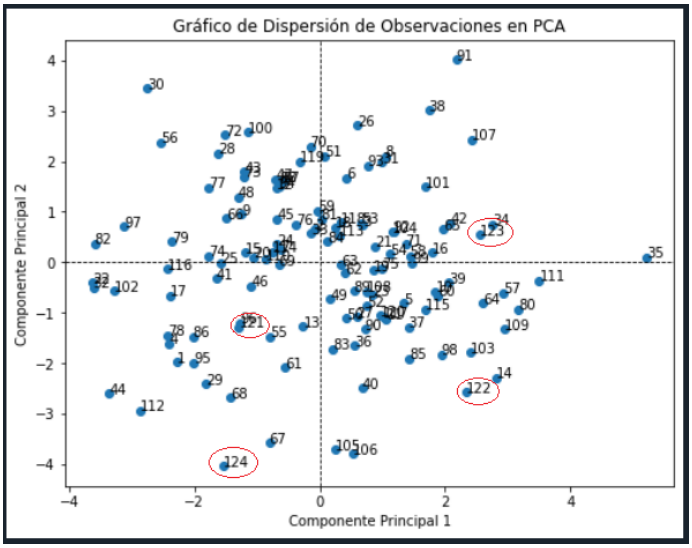

Figure 20: Dispersión observaciones y suplementarios.

Por último vamos vamos a considerar la posibilidad de añadir una nueva variable categórica al análisis, como sabemos, las variables categóricas no peuden pertenecer al ACP, pero, ¿Como se relacionaría con dicho analisis?

Cuando incorporamos una variable categórica a un ACP previamente realizado, estamos enriqueciendo nuestra comprensión de los datos al considerar la influencia de categorías adicionales en la estructura de los datos. En este contexto, representar los centroides de estas categorías en los nuevos ejes de las componentes principales se convierte en una estrategia esencial. Los centroides actúan como puntos de referencia que resumen la tendencia central de cada categoría en el espacio de las CP. Este enfoque nos permite visualizar cómo las distintas categorías se relacionan entre sí y con las CP, lo que puede revelar patrones y relaciones más sutiles en los datos que antes no eran evidentes. En esencia, al considerar variables categóricas y representar sus centroides en el ACP, estamos agregando una dimensión adicional a nuestra comprensión de la estructura subyacente de los datos.

Volviendo a nuestro ejemplo, vamos a retomar la variable categórica *EXTRA_ESC* para ver si hay algún patrón visible entre la actividad extraescolar de los alumnos y las CP.

```
1
2 # Incorporamos la variable categórica "EXTRA_ESC" en los datos
3 notas_componentes_sup= pd.concat([notas_sup, resultados_pca_sup], axis=1)
4
5 extra_sup = pd.concat([extra, extra_S], axis=0)
6 notas_componentes_sup_extra= pd.concat([notas_componentes_sup,
7                                          extra_sup], axis=1)
```

Listing 23: Representacion observaciones + suplementarios.

Para ver como se relacionan las categorías de la nueva variable, vamos a representar los centroides de dichas categorías calculados por cada par de ejes o CP (en nuestro ejemplo, solo dos).

```python


def plot_pca_scatter_with_categories(datos_componentes_sup_var, componentes_principales_sup, n_components):
    """
    Genera gráficos de dispersión de observaciones en pares de componentes principales seleccionados con categorías.

    Args:
        datos_componentes_sup_var (pd.DataFrame): DataFrame que contiene las categorías.
        componentes_principales_sup (np.ndarray): Matriz de componentes principales.
        n_components (int): Número de componentes principales seleccionadas.
    """
    # Obtener las categorías únicas
    categorias = datos_componentes_sup_var["EXTRA_ESC"].unique() #Modificar por el nombre de la variable categórica

    for i in range(n_components):
        for j in range(i + 1, n_components):  # Evitar pares duplicados
            # Crear un gráfico de dispersión de las observaciones en las dos primeras componentes principales
            plt.figure(figsize=(8, 6))  # Ajustar la dimensió de la figura si es necesario
            plt.scatter(componentes_principales_sup[:, i], componentes_principales_sup[:, j])

            for categoria in categorias:
                # Filtrar las observaciones por categoría
                observaciones_categoria = componentes_principales_sup[datos_componentes_sup_var["EXTRA_ESC"] == categoria]
                # Calcular el centroide de la categoría
                centroide = np.mean(observaciones_categoria, axis=0)
                plt.scatter(centroide[i], centroide[j], label=categoria, s=100, marker='o')

            # Mostrar etiquetas a las observaciones
            etiquetas_de_observaciones = list(datos_componentes_sup_var.index)

            for k, label in enumerate(etiquetas_de_observaciones):
                plt.annotate(label, (componentes_principales_sup[k, i], componentes_principales_sup[k, j]))
                # Dibujar líneas discontinuas que representen los ejes

            # Dibujar líneas discontinuas que representen los ejes
            plt.axhline(0, color='black', linestyle='--', linewidth=0.8)
```

```
            plt.axvline(0, color='black', linestyle='--', linewidth=0.8)

            # Etiquetar los ejes
            plt.xlabel(f'Componente Principal {i + 1}')
            plt.ylabel(f'Componente Principal {j + 1}')

            # Establecer el título del gráfico
            plt.title('Gráfico de Dispersión de Observaciones en PCA')

            # Mostrar la leyenda para las categorías
            plt.legend()
            plt.show()

plot_pca_scatter_with_categories(notas_componentes_sup_extra,
    componentes_principales_sup, fit.n_components)
```

Listing 24: Representacion observaciones + centroides.

Podemos intuir a partir de la Figura 21 que los estudiantes que practican tanto música como deporte en las actividades extraescolares tienen en media valores más altos tanto en las asignaturas de *letras* como en *ciencias*, por otro lado, los estudiantes que solo practican música tienen, en media, valores próximos al promedio en las variables de *ciencias* pero algo superiores en las de *letras*. Misma situación para los que solo realizan deporte, pero con notas en proimedio inferiores a la media en las asignaturas de *letras*.

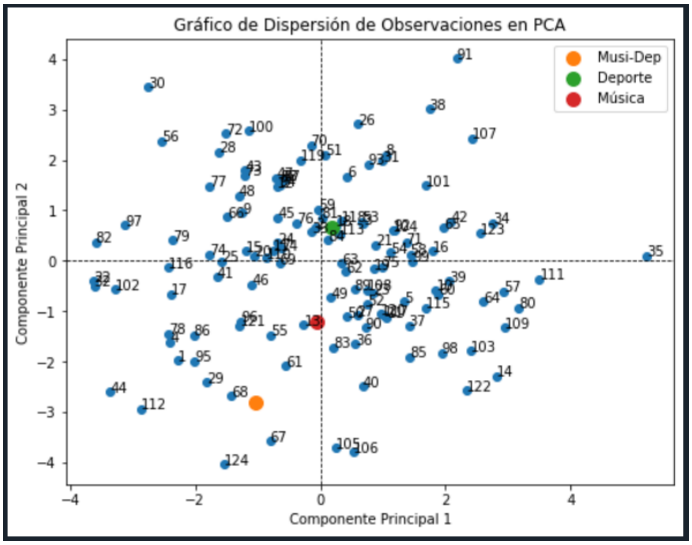

Figure 21: Dispersión centroides.