Applied Statistics - Notes - v0.2.0

260236

April 2025

Preface

Every theory section in these notes has been taken from two sources:

- The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition. [2]
- An Introduction to Statistical Learning: with Applications in Python. [3]
- Applied Multivariate Statistical Analysis. [4]
- Course slides. [5]

About:

GitHub repository

These notes are an unofficial resource and shouldn't replace the course material or any other book on applied statistics. It is not made for commercial purposes. I've made the following notes to help me improve my knowledge and maybe it can be helpful for everyone.

As I have highlighted, a student should choose the teacher's material or a book on the topic. These notes can only be a helpful material.

Contents

1 Business Data Analytics

1.1 Multivariate Descriptive Statistics

When we move from **analyzing** a single variable (univariate analysis) to **multiple variables at once**, we enter the realm of **Multivariate (MV) analysis**. A natural question arises: *Is multivariate analysis just a replication of univariate analysis across several variables?*

The answer is no, multivariate analysis introduces new and fundamental questions that cannot be answered by simply analyzing variables individually. The **core focus** shifts to **understanding how these variables interact with each other**. Specifically, we are concerned with the **dependence** and **correlation between variables**.

Covariance: Measuring Joint Variability

To capture how two variables vary together, we use **Covariance**. The **Sample Covariance** between variables x_j and x_k is calculated as:

$$\mathrm{cov}_{jk} = \mathrm{Cov}\,(x_j, x_k) = s_{jk} = \frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k) \tag{1}$$

- $s_{jk} = 0 \Rightarrow$ implies that there is **no linear relationship** between the two variables.
- $s_{jk} > 0 \Rightarrow$ suggests that **as one variable increases**, the **other tends to increase**.
- $s_{jk} < 0 \Rightarrow$ one **variable** tends to **decrease when the other increases**.

Covariance Is Not Standardized

The **value of covariance is not standardized**, it **depends on the units of measurement**, which makes comparisons difficult. For example:

- Suppose we're measuring
  - Height in centimeters
  - Weight in kilograms
- The covariance between height and weight will be expressed in *centimeter-kilogram* units.

Now imagine we convert height to meters. The covariance value changes, because now you're multiplying meters × kilograms instead of centimeters × kilograms. Even though the **relationship between height and weight hasn't changed**, the **numerical value of covariance does change due to this unit change**. **Because of unit dependency, it's hard to compare covariances between different variable pairs**. Finally, it is **hard to interpret the magnitude of covariance in any absolute sense** (e.g., is 50 a large covariance or small? It depends on the units!).

Correlation: Standardized Covariance

To standardize covariance and **measure the strength of a linear relationship** on a scale between -1 and 1, we use the **Correlation** coefficient, defined as:

$$\mathrm{cor}_{jk} = r_{jk} = \frac{s_{jk}}{\sqrt{s_{jj}}\sqrt{s_{kk}}} = \frac{\sum_{i=1}^{n}(x_{ij}-\bar{x}_j)(x_{ik}-\bar{x}_k)}{\sqrt{\sum_{i=1}^{n}(x_{ij}-\bar{x}_j)^2}\sqrt{\sum_{i=1}^{n}(x_{ik}-\bar{x}_k)^2}} \tag{2}$$

This formula divides the covariance by the product of the standard deviations of the two variables, giving a **unitless value**:

- $r = 0$: No linear correlation
- $r > 0$: Positive correlation (**both variables increase or decrease together**)
- $r < 0$: Negative correlation (**one increases while the other decreases**)
- $|r| = 1$: Perfect correlation (**exact linear dependence**)

Thus, correlation not only **reveals the direction of the relationship** but also its **strength**.

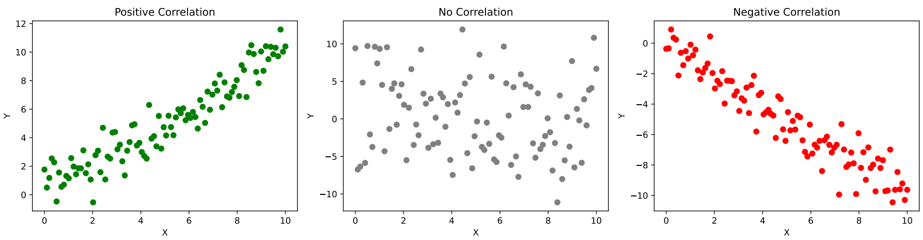

Figure 1: Direction of correlation: positive (left), none (center), negative (right).

Describing MV Data: Vector and Matrices

When analyzing multivariate data:

- We compute the **vector of Sample Means**:

$$\bar{\mathbf{X}} = [\bar{X}_1, \bar{X}_2, \dots, \bar{X}_p] \tag{3}$$

- And the **symmetric and positive semidefinite** (eigenvalues are non-negative) **Variance-Covariance matrix S**, which **summarizes the covariances between all pairs of variables**:

$$S = \begin{bmatrix} s_{11} & \cdots & s_{1p} \\ \vdots & \ddots & \vdots \\ s_{p1} & \cdots & s_{pp} \end{bmatrix} \tag{4}$$

- Alternatively, we can use the **Correlation matrix R**, where **all diagonal elements are 1** (because each variable is perfectly correlated with itself) and **off-diagonal elements are correlation coefficients**:

$$R = \begin{bmatrix} 1 & \cdots & r_{1p} \\ \vdots & \ddots & \vdots \\ r_{p1} & \cdots & 1 \end{bmatrix} \tag{5}$$

Scatterplots - Visualizing Variable Pairs

One of the most intuitive and widely used tools in multivariate analysis is the **2D Scatterplot**. Each plot shows how two variables relate to each other:

- ✔ Clusters or linear trends can indicate correlation or dependence.
- ✔ Scatterplots are **ideal for spotting positive, negative, or no correlation** visually.

However, scatterplots have a **limitation**: they **only** allow us to **analyze two variables at a time**. When dealing with many variables, the **number of possible pairings becomes large**, making it **difficult to read or interpret** the scatterplots individually. This is where quantitative measures (like correlation matrices) and higher-dimensional graphics come into play.

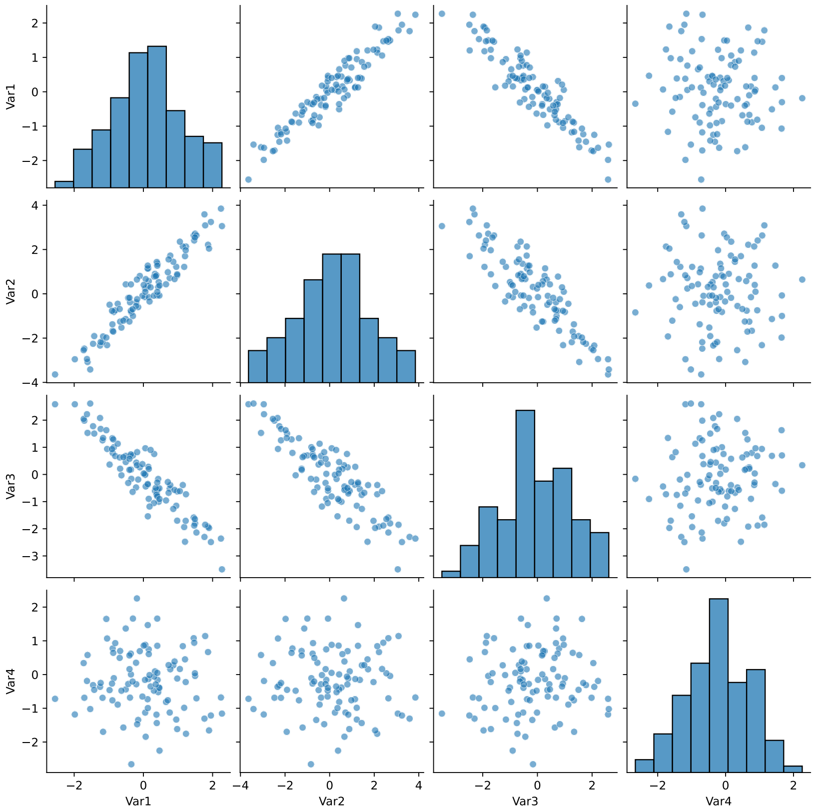

Figure 2: Scatterplot matrix of four variables. This scatterplot matrix displays all pairwise relationships among four variables:

- Diagonal plots (top-left to bottom-right): Histograms showing the distribution of each variable.
- Off-diagonal plots: 2D scatterplots illustrating the relationship between each pair of variables.

Rotated Plots in 3D - Capturing Complexity

When dealing with **three variables**, we can **extend scatterplots into 3D space** using **Rotated plots**. These visualizations allow us to:

- **Explore interdependencies** among three variables at once.
- **Gain spatial insight** into how data points spread in three-dimensional space.
- Observe **complex patterns that are invisible in 2D**.

Yet again, as we move **beyond three variables**, visualizing becomes **impractical**, our brains cannot easily comprehend 4D or higher dimensions. Hence, dimensionality reduction techniques like PCA are often used alongside visualizations to make high-dimensional data "digestible".

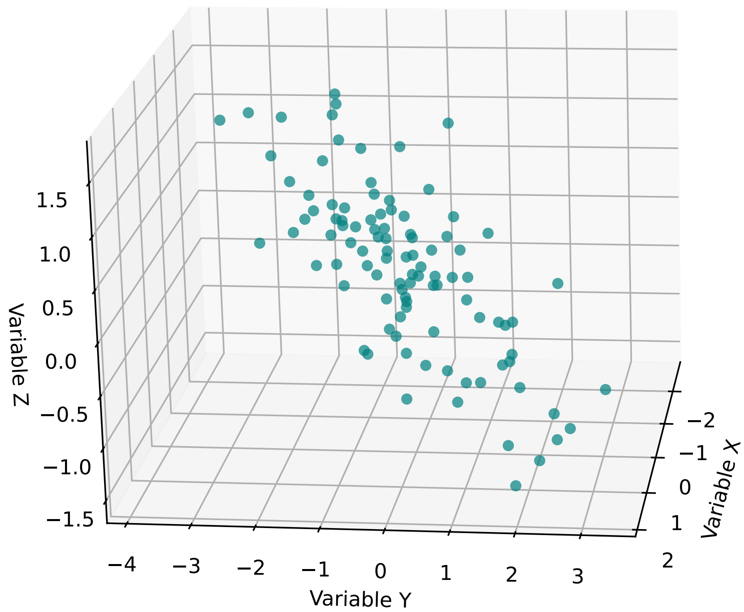

Figure 3: A simple rotated plots in 3D.

Star Plots - Shape-Based Comparison

Star plots offer a creative way to **represent multivariate data**:

- **Each variable is represented as a ray** (spoke) starting from a central point.
- The **length of each ray** corresponds to the **value** of that variable.
- When **rays are connected**, they form a "star-like shape" **unique** to each observation.

This method is **excellent for comparing patterns** between observations:

✔ **Similar shapes** suggest **similar data profiles**.

✔ **Differences** in shape can **quickly highlight outliers** or clusters.

However, star plots have limitations:

✘ They **do not quantify correlation**.

✘ The **direction** and **magnitude** of relationships between variables are **not explicit**.

✘ They are better for **visual pattern recognition** than for precise statistical analysis.

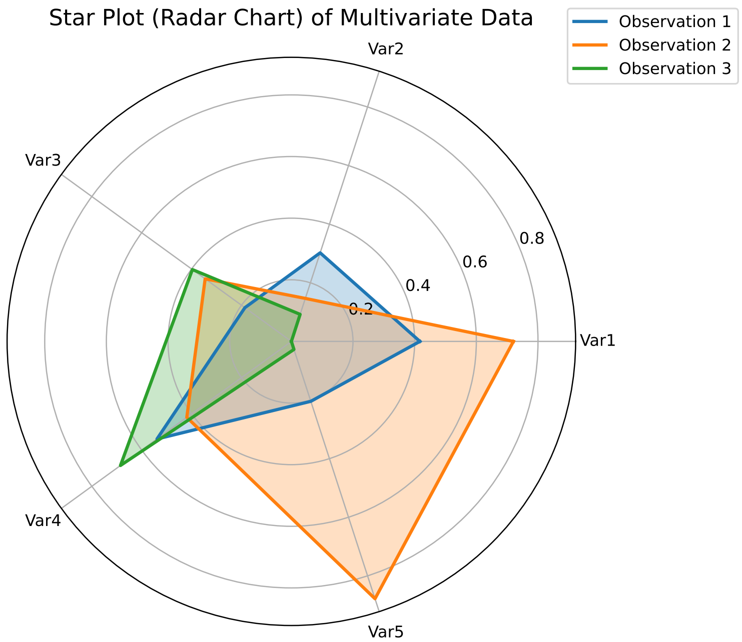

Figure 4: Star Plot (Radar Chart) of Multivariate Data.

Chernoff Faces - Human-Centric Visualization

Chernoff faces [1] are an **innovative visualization method** where **multivariate data is represented as a human face**:

- Each **variable controls a facial feature** (e.g., mouth curvature, eye size, nose length).
- **People are naturally attuned to recognizing faces** and subtle differences in expressions.
- Hence, Chernoff faces **leverage human perception** for quickly comparing **multivariate observations**.

Despite being engaging, Chernoff faces also have **drawbacks**:

✗ They **do not provide numerical precision**.

✗ The **mapping of variables to facial features** can be **arbitrary**.

✗ They work best as a **qualitative summary tool** rather than for deep statistical inference.

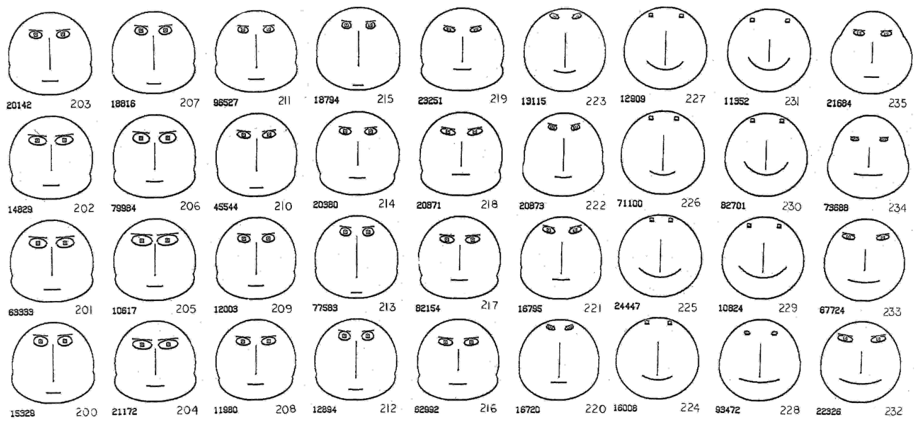

Figure 5: Some Chernoff faces. [1]

Graphic Type	Strengths	Limitations
Scatterplots	Clear view of pairwise relationships	Hard to scale beyond 2 variables
3D Plots	Visualizes 3-variable interaction	Limited to 3 dimensions, requires rotation
Star Plots	Quick shape-based comparison across variables	No quantification, poor at showing correlations
Chernoff Faces	Leverages facial perception for comparison	Subjective, lacks precision

Table 1: When and why to use graphics.

1.2 Dimensionality Reduction

The Challenge: Data in High Dimensions

In many real-world problems, we **collect multiple variables for each observation**. For example, in a medical study, a patient might be described by age, weight, blood pressure, and dozens of test results. This leads to **high-dimensional data**, where **each observation is a point in a complex, multi-dimensional space** (formally, a Euclidean space of dimension p).

The problem? As the **number of variables** (p) **increases**, the **data becomes harder to visualize, interpret, and model**:

- Some **variables** might be **redundant** or **highly correlated**.
- **Computations** become more **expensive**.
- **Patterns become obscured** by the complexity.

Goal of Dimensionality Reduction

We want to **summarize the data using fewer variables**, say k derived variables (with $k < p$), that still **retain most of the information**. This process is a balancing act:

- **Clarity**: fewer variables make data easier to understand and visualize.
- **Risk of oversimplification**: reducing dimensions too much can cause loss of important information.

The key concept here is that in statistics, **information is variability**. The more variability we retain from the original data, the more information we preserve.

Example 1: Blood Cells

Imagine we measure thickness and diameter for a set of red blood cells:

- Each cell = one observation with two variables.
- We can represent this as a table (numbers) or as a 2D scatterplot.

Now we ask ourselves: *Can we describe these cells using only one feature instead of two?*
If we choose only diameter or only thickness, we'll lose detail:

- Some cells will appear more similar than they really are.
- We miss variability that distinguishes them.

So, we seek a better single feature, one that captures the most variation possible from both thickness and diameter combined.

The Statistical Insight: Maximize Variability

Rather than randomly picking a feature, we **analyze the directions along which the data varies the most**.

1. First, we find the **direction of maximum spread**.
2. Then, the **second most spread direction**, orthogonal to the first.

This is the essence of **Principal Component Analysis (PCA)**, a **dimensionality reduction technique that finds the best directions** (linear combinations of variables) **to project the data**, while **maximizing retained variability**.

Dimensionality Reduction in Practice

Let's formalize the idea:

- We start with a **data matrix** X of shape $n \times p$ (n observations, p variables).
- The goal is to **obtain a new matrix** M of shape $n \times k$, with $k < p$, that **captures most of the variability**.
- The **difference** between X and M is **residual variation**, the **information lost**.

In summary, **Dimensionality Reduction** is about simplifying complexity: **transforming a large set of variables into a smaller**, more interpretable set **without losing the essence of the data**. It's central to data exploration, preprocessing, and modeling, especially when working with high-dimensional datasets.

1.3 Principal Component Analysis

Why PCA?

Imagine we have a data set with **many variables**, such as measurements of people, products, or cities. Some **variables might be closely related** (like height and weight), **while others might carry similar information**. This creates two challenges:

1. It becomes **hard to analyze and interpret** the data.
2. **Redundancy** can lead to inefficiency and confusion.

Principal Component Analysis (PCA) helps solve this by providing a **simplified version of the dataset**, where we focus only on the **essential information**.

The Main Idea

PCA works by **creating new variables**, called **principal components**, that are:

- **Combinations** of the original variables.
- **Ordered** so that the **first component captures the most variability** in the data.
- Each **subsequent component captures the next most variability**, but only from what's left over, and each **new component is uncorrelated with the previous** ones.

Imagine this like finding better "angles" or "perspectives" from which to view our data, ones that maximize how much we can see (i.e., variability) with as few perspectives as possible.

Variability is Information

In statistics, **Variability**, **how much values change from one observation to the next**, is considered **information**. The more variability a component captures, the **more useful it is** in understanding the data. So, PCA's job is to **find the directions in which the data varies the most**, and to use those directions to summarize the dataset.

How PCA Changes the Dataset

Let's say we start with p variables (like height, weight, age, income...). **PCA gives us p principal components**, but the *magic* is that **we usually only need the first few** to understand most of what's going on.

So instead of working with the full, original dataset, we now have a **simpler version**:

- We still have the **same number of observations**.
- But each observation is now described by **fewer, more informative variables** (components).

This is called a **low-dimensional representation** of the data.

Scores and Loadings - The Ingredients and the Result

In this new system:

- The **Scores** tell us where **each observation lies along the new axes** (the principal components).
- The **Loadings** tell us **how the original variables contribute to each component**.

Think of it like cooking:

- **Original variables** = ingredients
- **Loadings** = recipe instructions
- **Principal components** = final dishes
- **Scores** = ratings of the dishes for each person (observation)

Matrix Representation

Let's denote the data matrix as X, with n rows (observations) and p columns (variables). PCA is essentially about **transforming this matrix X into a new matrix**, where:

- The data is now described by **principal components** instead of the original variables.
- The **goal is to rotate and simplify** the data in a way that **emphasizes the most important directions** (maximum variability).

Two key matrices in PCA:

- **Loadings Matrix (V)**
  - This matrix contains the **weights** used to build the principal components.

  - Each **column** in V corresponds to a **principal component**.
  - Each **value** in V shows **how much each original variable contributes** to the component.

  We can think of V as the recipe book: it tells us how to combine original variables to create the new components.

- **Scores Matrix (U)**
  - This matrix contains the **projections of the data** onto the principal components.
  - Each **row** in U represents an **observation in the new PCA space**.
  - These are called scores, they tell us **where each observation lands along the new axes** (PC1, PC2...).

  So U is the result: it shows **how our data looks in the new, simplified system**.

PCA can be **computed using Singular Value Decomposition (SVD)**:

$$X = U \cdot S \cdot V^T \tag{6}$$

PCA **factorizes the data matrix** X. Here's what each matrix means:

- X: Original data ($n \times p$)
- U: Scores matrix ($n \times p$), orthogonal (columns are independent)
- S: Diagonal matrix of **singular values** (related to the variance explained)
- V^T: Transposed loadings matrix ($p \times p$), also orthogonal.

But for understanding, focus on this **simpler form**:

$$\textbf{PCA result} = \text{Scores} = X \cdot V$$

We **multiply the data X by the loadings matrix V** to obtain the **scores**.

However, both U and V are **orthogonal matrices**, meaning:

- Their columns are perpendicular (**no redundancy**).
- Principal **components are uncorrelated**, each new component captures new, non-overlapping information.

This is mathematically elegant and practically useful because it **removes multicollinearity** and makes downstream **analysis simpler and more robust**.

❓ How Much Information Do We Keep?

Each principal component has a **percentage of variance explained**, this tells us how much of the original data's information it retains. Often, **the first 1 or 2 components explain so much that we can ignore the rest**.

In conclusion, **PCA helps us focus on what matters** in our data. It's like cleaning our glasses: everything becomes sharper, simpler, and more meaningful. We go from being overwhelmed by numbers to seeing clear patterns. It's a tool used everywhere, from finance to biology, marketing to engineering, whenever people need to make sense of complex data.

1.4 PCA Reference System

Why Do We Talk About Projections in PCA?

When we say that PCA projects data, we mean it **transforms the data points by placing them onto new axes**, the principal components, that better represent the structure of variability.

But why are projections so important? Because PCA has **two goals**:

1. **Maximize variance**: We want the **data to spread out as much as possible along the new axis**. This spread means we're capturing differences between observations.

2. **Minimize residuals**: We want to **minimize the error** when we approximate each point using the new axes. This is done by projecting each point perpendicularly onto the new axes, just like the shortest path between a point and a line.

This is why we "keep talking about projections": they allow us to **retain the most information** with **the least distortion**.

Generalizing to More Dimensions

In real datasets, we often have **more than two variables**. PCA scales to p dimensions, and the idea of projection still applies:

- PC1: The **direction in p-dimensional space** where data varies most.
- PC2: The next direction, **perpendicular to the first**, that captures remaining variability.
- This continues until we have p principal components, each orthogonal to the others.

So **PCA rotates the entire dataset into a new reference system** where the data structure is easier to understand.

A New Reference System

After PCA, our data now lives in a **new coordinate system**:

- The **original variables** (e.g., height, weight) are no longer the axes.
- Instead, the axes are **principal components**, which are **combinations of the original variables**.

This new reference system has two main features:

1. **PCA results are rotation invariant**

   What does this means? It means that if we rotate our dataset (e.g., by changing the coordinate system), PCA still finds the same underlying structure, the same principal components (relative to the data).

- ***Why?*** PCA depends only on the relationships between the data points, specifically: the variances and covariances between variables. These are not affected by rotations of the data in space. Mathematically, PCA extracts eigenvectors of the covariance matrix that are invariant under rotation with respect to relative directions.

2. **Principal Components are independent (uncorrelated)**

- ***What does this means?*** The principal components (PC1, PC2, ...) are uncorrelated with each other. Knowing the value of one PC tells us nothing about the value of the next.
- ***Why?*** Each new component is orthogonal to the previous one. And since the correlation is equal to the cosine of the angle between the variables ($\cos(90) = 0$), PAC ensures that the correlation between the PCs is zero.
- ✓ In high dimensions, **collinearity** often indicates redundant information between variables and can cause problems in the analysis. **PCA solves** this problem by giving us independent, uncorrelated variables.

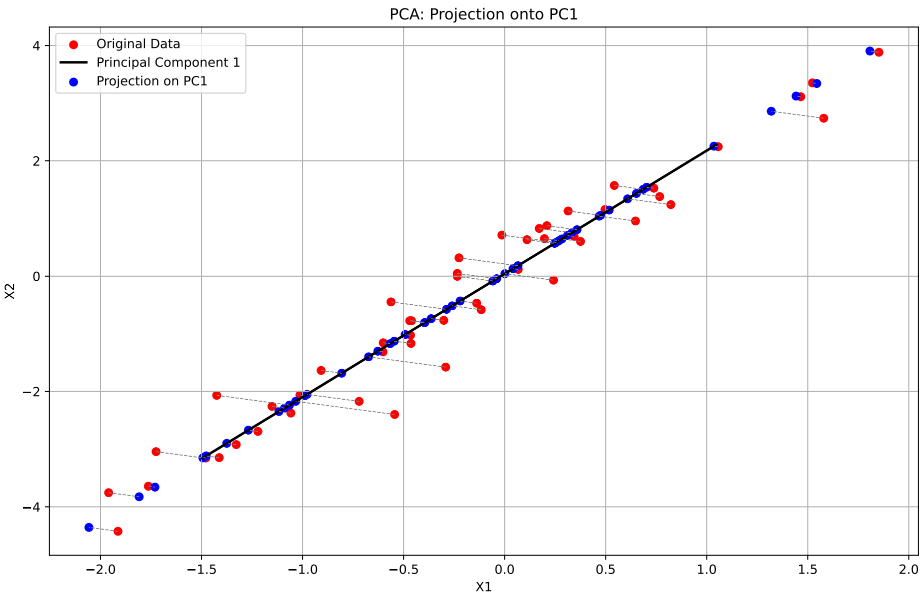

Figure 6: We have a bunch of scattered red dots. We have drawn a black line through the middle of the cloud of dots in the direction where the dots are most scattered. This line is our first principal component (PC1). For each red point, we drop a perpendicular line onto the black line. Where it lands, we place a blue point. This is called projection. Furthermore, the longer the black line, the more red points are taken into account (thus maximizing variance); on the other hand, the blue points must be close to the red points (shorter the distance) to minimize residuals and lose less information.

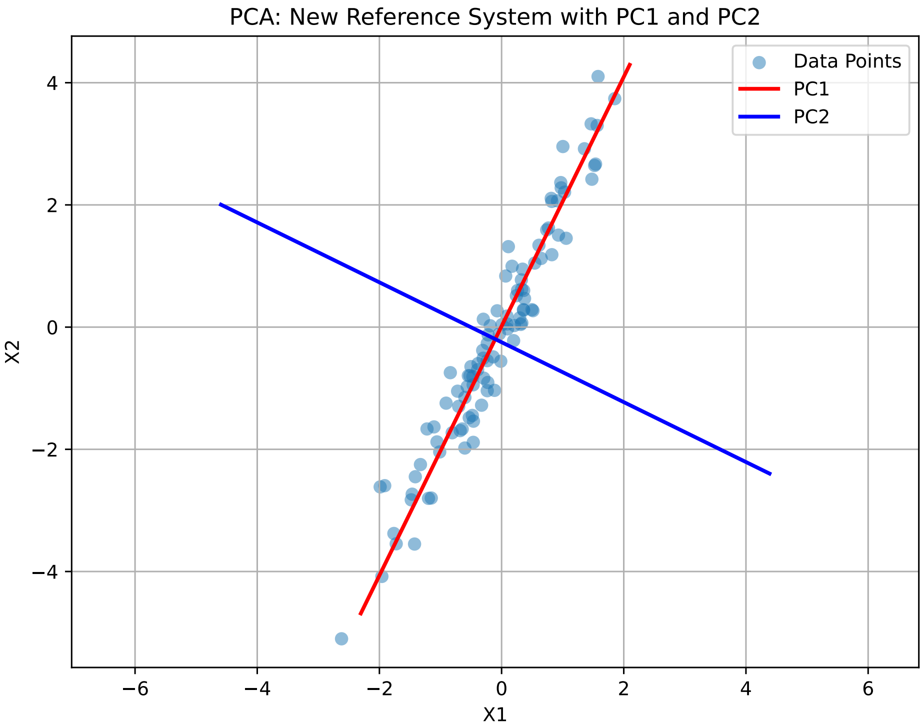

Figure 7: By drawing two lines, PC1 and PC2, we obtain a new reference system. It is a rotated system. The axes are principal components, which are combinations of the original variables.

Principal Components: Linear Combinations

Let's now talk about **how principal components are constructed**. Each principal component is a **linear combination of the original variables**. This means:

$$\mathrm{PC}_1 = \phi_{11} \cdot x_1 + \phi_{21} \cdot x_2 + \cdots + \phi_{p1} \cdot x_p$$

But in general:

$$\mathrm{PC}_j = \phi_{1j} \cdot x_1 + \phi_{2j} \cdot x_2 + \cdots + \phi_{pj} \cdot x_p \tag{7}$$

Where:

- The ϕ-values (phi) are called **Loadings**.
- They are the **weights assigned to each original variable**.
- The **higher the loading**, the **more that variable contributes** to that principal component.

For **each observation** (e.g., a person, product), we now **compute their position in the new PCA space**:

$$z_{ij} = \phi_{1j} \cdot x_{i1} + \phi_{2j} \cdot x_{i2} + \cdots + \phi_{pj} \cdot x_{ip} \tag{8}$$

These values z_{ij} are called **Scores**. They tell us:

- Where each observation lies **along a principal component axis**.
- How much that **component contributes** to describing this observation.

So now, instead of describing each person with p original variables, we describe them with k scores, where $k < p$, but we still capture the essence of their data.

1.5 PCA as Optimization Problem

◎ What Is PCA Trying to Achieve?

PCA aims to find new variables (**principal components**) that are:

- **Linear combinations** of the original variables.
- Chosen so that the **first principal component** (PC1) captures the **maximum possible variance** in the data.
- **Second and further PCs** capture the **remaining variance**, while being uncorrelated with all previous ones.

In other words, PCA tries to find a new axis (direction) along which the data varies the most. This is the first principal component. Once this direction is found, each data point can be projected onto it to create a simplified representation of the data.

To find this direction (**PCA1**), **PCA solves a mathematical optimization problem**. This is because the **goal is to find the direction that maximizes the variance** of the data projected onto that direction (we want to maximize the variance because it captures the most variability, information, in the data).

❓ How Is This Formulated as an Optimization Problem? (PC1)

1. **Define a New Variable Z_1**. Let's construct a new variable Z_1 which is a **linear combination** of the original variables:

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \cdots + \phi_{p1}X_p$$

   Where:

   - X_j is the j-th variable (all observations for that variable)
   - ϕ_{j1} is the j-th weight (loading) assigned to the variable X_j.

   The subscript 1 indicates that they belong to the first principal component.

2. **Objective - Maximize Variance of Z_1**. We want to maximize the variance to capture the most variability (information) in the data:

$$\text{Maximize} \quad \text{Var}\left(Z_1\right) = \text{Var}\left(\phi_1^T X\right)$$

   Where:

   - X is the **data matrix** $n \times p$
   - ϕ_1 is the vector $p \times 1$ of **weights** (**loadings**) used to build Z_1. It is transposed to allow multiplication with the data matrix X.

   More specifically:

$$\text{Var}\left(Z_1\right) = \frac{1}{n}\sum_{i=1}^{n} z_{i1}^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{j1}\cdot x_{ij}\right)^2$$

Where z_{i1} is the **score** (see page 18) of the i-th observation on the **first principal component** (e.g., z_{i2} score of observation i on PC2). In our case it is:

$$z_{i1} = \sum_{j=1}^{p} \phi_{j1} \cdot x_{ij}$$

3. **Constraint - Normalize the Loadings**. We need a **constraint** because we could scale the weights (loadings) infinitely. Therefore, we need **to fix the length** (norm) **of the loadings vector**.

$$\sum_{j=1}^{p} \phi_{j1}^2 = 1$$

Or, in vector notation:

$$||\phi_1||^2 = 1$$

The constraint says that the total energy or length of the loadings vector is fixed to 1 and PCA can only choose the direction of ϕ_1, not its size.

Now we can write the **full optimization problem for PC1**:

$$\text{Maximize} \quad \text{Var}\,(Z_1) \quad \text{subject to} \quad ||\phi_1||^2 = 1 \tag{9}$$

Specifically, the extended form:

$$\max_{\phi_{11},\ldots,\phi_{p1}} \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j1} \cdot x_{ij} \right)^2 \quad \text{subject to} \quad \sum_{j=1}^{p} \phi_{j1}^2 = 1 \tag{10}$$

❓ Okay, but how does PCA solve the optimization problem?

In previous steps, we posed PCA as:

- Maximize the variance of $Z_1 = \phi_1^T X$
- Subject to $||\phi_1||^2 = 1$

The solution to this problem: the **optimal loadings vector** ϕ_1 (for PC1) **is the first eigenvector of the sample covariance matrix S**. Let's analyze this sentence to understand how and why.

❓ **Why Eigenvectors Help Solve This?** We need eigenvectors because PCA's optimization problem is mathematically equivalent to a linear algebra problem whose solution requires eigenvectors.

The **mathematical problem can also be written in quadratic form**[1]

$$\text{Var}(Z_1) = \phi_1^T S \phi_1$$

Where S is the covariance matrix (page 6). It follows the exact pattern of the quadratic form, because ϕ_1 is a vector of loadings, S is the covariance matrix (square), and the whole expression evaluates to a scalar: the variance of Z_1.

In linear algebra, the **quadratic form problem is solved by eigenvectors**. Using the **Rayleigh Quotient Theorem**, we can obtain the goal of PCA. Given a symmetric matrix S (like the covariance matrix in PCA), the *maximum value* of the quadratic form:

$$\phi_1^T S \phi_1 \quad \text{subject to} \quad ||\phi_1||^2 = 1$$

Is achieved when ϕ_1 is the eigenvector corresponding to the largest eigenvalue of S. Therefore, the **solution must be the first eigenvector** (by the Rayleigh Quotient Theorem).

[1] A **Quadratic Form** is any expression that involves a vector, a matrix, and the same vector transposed. The general structure is:

$$Q(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x} \tag{11}$$

Where $\mathbf{x}$ is a vector, $\mathbf{A}$ is a square matrix, and the result is a scalar.

❓ Optimization Problem for the Second Principal Component (PC2)

The following steps allow to obtain the PC2. We avoid a long explanation here, because we made the same logical steps of the optimization problem for PC1.

1. **Define a New Variable Z_2.** Just like with PC1, we define PC2 as a **linear combination** of the original variables:

$$Z_2 = \phi_{12}X_1 + \phi_{22}X_2 + \cdots + \phi_{p2}X_p$$

   For a specific observation i:

$$z_{i2} = \sum_{j=1}^{p} \phi_{j2} \cdot x_{ij}$$

   Where: $\phi_2 = [\phi_{12}, \ldots, \phi_{p2}]^T$ is loadings vector for PC2. Here the **goal is to find the best ϕ_2 to create PC2**.

2. **Objective - Maximize Variance of Z_2.** Just like PC1, we want to **maximize variance**:

$$\max_{\phi_2} \quad \mathrm{Var}(Z_2) = \frac{1}{n}\sum_{i=1}^{n} z_{i2}^2$$

3. **Constraints (there are two now)**

   (a) **First Constraint: Normalize the Loadings.** We must prevent arbitrary scaling of the weights:

$$\sum_{j=1}^{p} \phi_{j2}^2 = 1 \quad \text{or} \quad \|\boldsymbol{\phi}_2\|^2 = 1$$

   (b) **Second Constraint: Ensure PC2 is Uncorrelated with PC1.** This is new compared to PC1. Here we want the covariance between PC1 and PC2 to be zero; in other words, we're saying ***we don't want these two results to be correlated***.

$$\mathrm{Cov}(Z_1, Z_2) = 0$$

   Mathematically, this is equivalent to saying:

$$\phi_1^T\phi_2 = 0 \qquad \text{(orthogonality)}$$

   If **two directions are orthogonal**, their **projections** (scores) are **uncorrelated**.

Finally, we can write the **full optimization problem for PC2**:

$$\max_{\phi_{12},\ldots,\phi_{p2}} \quad \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{j2}x_{ij}\right)^2 \tag{12}$$

Subject to:

$$\sum_{j=1}^{p} \phi_{j2}^2 = 1 \quad \text{and} \quad \sum_{j=1}^{p} \phi_{j1} \cdot \phi_{j2} = 0 \tag{13}$$

❓ **And how to solve the PC2 optimization problem?** The solution to this constrained optimization problem is similar to the PC1:

$$\phi_2 = \mathbf{e}_2 \tag{14}$$

Where:

- $\mathbf{e}_2$ equal to the **second eigenvector** of the **covariance matrix** S.
- This eigenvector corresponds to the **second largest eigenvalue** λ_2.

1.6 Proportion of Variance Explained (PVE)

In the previous sections, we explained how to compute PC1, PC2, etc. as new variables. Each PC is a linear combination of the original variables, and each maximizes the variance under orthogonality constraints. But now the question becomes: "*okay, we computed these new components... but how useful are they?*".

In PCA, variance equals information. So now our **goal** is: "how much **information (variance) is explained by PC1**? And by **PC2**? And by **PC3**? And so on". This is where **Proportion of Variance Explained (PVE)** comes in.

We know that each principal component explains some variance:

- PC1 captures the **largest possible amount** of variance.
- PC2 captures the **next largest amount**, and so on.
- All PCs **together** explain **100% of the variance** in the data (no information is lost if we keep all of them).

If the data matrix X is centered (i.e., variables have mean 0), then the **total variance in the data** is:

$$\mathrm{Var}_{\mathrm{TOT}} = \sum_{j=1}^{p} \mathrm{Var}(X_j) = \sum_{j=1}^{p} \left(\frac{1}{n} \sum_{i=1}^{n} x_{ij}^2 \right) \tag{15}$$

This is just the **sum of the variances** of the original variables.

To know the **variance** at the step k, so for the **k-th principal component**, we generalize:

$$\mathrm{Var}_k = \frac{1}{n} \sum_{i=1}^{n} z_{ik}^2 \tag{16}$$

Where z_{ij} is the **score of observation** i on component k. In other words, it is the **spread of data along PCk**.

Since the **PVE** is a proportion, the equation is pretty obvious:

$$\mathrm{PVE}_k = \frac{\mathrm{Var}_k}{\mathrm{Var}_{\mathrm{TOT}}} \tag{17}$$

It is the **percentage of total variance** explained by PCk. This tells us how informative PCk is, and whether it's worth keeping or can be discarded.

In practice, we **often keep only the first few PCs**. For example, if we find that $PC1 + PC2$ explain 90% of the variance, we can keep only those two (dimensionality reduction).

1.7 Covariance vs. Correlation in PCA

In the previous sections, we discussed what PCA is and how we can use it. We know that PCA is based on the covariance matrix, that each principal component explains a portion of the total variance, and that PCA depends on the variance of the data. But "**what happens when the variables are on completely different scales?**" (Variable 1 is height, variable 2 is euros, variable 3 is just a count). If we apply PCA to the covariance matrix, the **variable with the largest variance will dominate the result**, even if it's not more important!

More formally, **PCA is based on the spread** (variance) **and co-movement** (covariance) **between variables**. But this depends heavily on the units of measurement and scales of the variables. And the problem is, should **we use the covariance matrix or the correlation matrix** to perform PCA? The answer depends on the scales of our variables.

⚠ Covariance Matrix: use with caution

The **covariance matrix measures how variables change together**, in absolute units.

- ✓ This is fine only if **all variables are measured on comparable scales** (e.g., all in centimeters, or all in euros).
- ✗ If one variable has **much larger variance**, it will **dominate the PCA**, even if it's not the most important feature.

For example, suppose height (meters) has a variance of 0.01 and income (euros) has a variance of $10'000$. Income will control the direction of PC1 simply because its variance is greater, not because it's more meaningful.

✓ Solution: Standardize the Data

To prevent PCA from being biased toward large-scale variables, we **standardize** each variable:

$$X'_{ij} = \frac{X_{ij} - \bar{X}_j}{\mathrm{SD}_j} \tag{18}$$

Where:

- $\bar{x}_j$ is the mean of variable j.
- SD_j is the standard deviation of variable j.

This standardization ensures that the **mean is zero** and the **standard deviation is one**. Now **all variables are on the same scale** and PCA treats them equally.

✅ Alternative: use the correlation matrix instead of the covariance matrix

Another way to achieve this is to run PCA on the **correlation matrix**, which is simply the **covariance matrix of the standardized variables**. This work because:

$$\mathrm{Corr}(X_j, X_k) = \frac{\mathrm{Cov}(X_j, X_k)}{\mathrm{SD}\,(X_j) \cdot \mathrm{SD}\,(X_k)} \tag{19}$$

So if we **standardize the variables first, the covariance becomes correlation**.

Case	**Matrix to use**	**Why**
All variables are on the same scale	Covariance matrix	Keeps units and variances as they are
Variables have different units/scales	Correlation matrix	Avoids bias toward large-variance variables
You standardized the variables	Covariance = correlation	Because standardization equalizes them

Table 2: When to use Covariance or Correlation.

1.8 PCA via SVD - Computational Aspects

Instead of computing PCA by:

1. Building the covariance matrix $S = \frac{1}{n-1} X^T X$
2. Diagonalizing S to get eigenvectors and eigenvalues

We can instead **perform PCA directly using Singular Value Decomposition (SVD)** on the **centered data matrix** X itself.

Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) method is a factorization of a matrix into three other matrices. If X is the centered data matrix (size $n \times p$), then:

$$X = U \Sigma V^T \tag{20}$$

Where:

- **U**: matrix of **left singular vectors** (related to scores). It is orthogonal and $n \times n$.
- **Σ**: diagonal matrix of **singular values** $\Sigma_1, \Sigma_2, \ldots$
  - Non-negative real numbers on the diagonal (singular values)
  - Singular values sorted in descending order (from largest to smallest)
  - Number of eigenvalues guaranteed by the minimum between number of columns and number of rows: $\min(n, p)$.
- **V**: matrix of **right singular vectors**. It is orthogonal and $p \times p$. They are the main directions, so the loadings of PCA.

The eigenvalues of the covariance matrix can be calculated from the singular values Σ_i:

$$\lambda_i = \frac{\Sigma_i^2}{n-1}$$

This is because if we plug the SVD of X into this formula, the eigen decomposition of S naturally follows.

Concept	**Meaning/Role in PCA**
$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$	SVD decomposition of centered data matrix
$\mathbf{V}$	Principal directions (same as eigenvectors of covariance)
Σ_i	Singular values
$\lambda_i = \frac{\Sigma_i^2}{n-1}$	Variance explained by each principal component

Table 3: Summary table of SVD for PCA.

Advantages

- **Numerically more stable**: works better when the data matrix is large or ill-conditioned.
- **No need to compute and store the covariance matrix**: we can work directly on X.
- **More efficient** when p is large (many variables), especially in high-dimensional problems.

2 Clustering Methods

2.1 Introduction

In the previous section, we talked about PCA for compression or visualization. However, this topic falls under a type of technique called unsupervised learning.

- **Supervised Learning** is a type of machine learning where the algorithm is trained on a labeled dataset, meaning **each** training **example includes both the input data and the correct output**.

  The **goal** is to learn a function that maps inputs to outputs, in order to make predictions on new, unseen data. Typical tasks include:

  - Classification (e.g., spam detection, image recognition)
  - Regression (e.g., predicting house prices or credit scores)

- **Unsupervised Learning** is a type of machine learning where the algorithm is given **only input data without any labeled output**.

  The **objective** is to identify patterns, structures or groupings within the data. Common applications include:

  - Clustering (this section)
  - Dimensionality Reduction (e.g., PCA for compression or visualization, section 1.2, page 11)

In essence, while supervised learning is about *prediction*, unsupervised learning is about *discovery*.

What is Clustering?

Clustering is the process of grouping a set of objects in such a way that **objects within the same group (cluster)** are **more similar** to each other than to those in other groups.

- No "true" labels are provided (unsupervised learning), the objective is to **uncover structure**.
- Often used in exploratory data analysis.
- The notion of "similarity" is fundamental and context-dependent (commonly based on **distance measures**).

The basic idea of clustering is:

1. **Minimize intra-cluster distances**: member of the **same cluster** should be **close to each other**.
2. **Maximize inter-cluster distances**: **different clusters** should be **well separated**.

Example 1: Energy Consumption in Milan

A practical example:

- Consider n users, each described by their energy consumption across p time slots.
- The task is to identify **groups of users with similar consumption patterns**, potentially revealing roles like *residences vs offices* or *daytime vs nighttime* usage.
- Since there is no predefined label, this is a classic unsupervised learning task.

This example encapsulates the **essence of clustering**, discovering structure in data that wasn't explicitly labeled.

How Many Clusters?

A fundamental **challenge** in clustering is **deciding the number of clusters** (k). Visualizations show that data can often be reasonable clustered in 2, 4, or 6 groups, depending on the chosen definition or similarity.

The **clustering result depends on the similarity measure** (e.g., Euclidean distance), and the notion of a "true" cluster is often ambiguous. This ambiguity is a central theme in unsupervised learning: since there's no ground truth, **evaluating the "correctness" of a clustering is inherently difficult**.

Types of Clustering

There are two main paradigms:

1. **Hierarchical Clustering**: builds a tree of clusters. Two main strategies:
   - **Agglomerative**: start from individual points and merge them.
   - **Divisive**: start from the whole dataset and split it recursively/
2. **Partitional Clustering** (also called flat clustering): divides data into k non-overlapping groups. Each data point belongs to exactly one cluster.

Hierarchical methods are more interpretable, while partitional methods (e.g., k-means) are often faster and scalable.

2.2 Defining Similarity in Clustering

At the heart of clustering lies a simple yet crucial idea: "objects in the same cluster should be **more similar** to each other than to objects in other clusters". But how do we define "*similar*"? Clustering methods don't work in a vacuum, they **need a distance** (or similarity) **function to determine how close two data points are**. This choice directly affects the clustering result.

Types of Distance Metrics

Given two data points $x = (x_1, x_2, \ldots, x_p)$ and $y = (y_1, y_2, \ldots, y_p)$, the distance between them can be computed in different ways, depending on the nature of the data and the goals of the analysis.

- **Euclidean Distance**

$$d_E(x, y) = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2} \tag{21}$$

  - Measures straight-line distance.
  - Works well when **features are on the same scale**.
  - Variants: squared euclidean and standardized euclidean (normalize variables before computing).

- **Manhattan Distance**

$$d_M(x, y) = \sum_{i=1}^{p} |x_i - y_i| \tag{22}$$

  Also called **L1 norm**, measures **grid-like distance** (think of navigating city blocks).

- **Chebyshev Distance**

$$d_C(x, y) = \max_i |x_i - y_i| \tag{23}$$

  Takes the **maximum absolute difference** across dimensions. Sensitive to the **worst-case** coordinate difference.

- **Cosine Similarity (Angle-based)**

$$\text{similarity}(x, y) = \cos(\theta) \tag{24}$$

  Based on the **angle between vectors**. The smaller the angle, the more similar the vectors. Common in **text mining** and **high-dimensional sparse data**.

- **Correlation-Based Distance**

$$d_R(x, y) = 1 - \text{corr}(x, y) \tag{25}$$

  Measures **shape similarity** and ignores magnitude; focuses on patter of variation.

- **Mahalanobis Distance**

$$d_M(x, y) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)} \tag{26}$$

  Accounts for **correlation between variables**. Useful when features have **different variances** or **covariances**.

- **Minkowski Distance**

$$d_p(x, y) = \left(\sum_{i=1}^{p} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{27}$$

  General formula that generalizes Euclidean and Manhattan. If $p = 1$ is Manhattan, if $p = 2$ is Euclidean.

❓ Why Distance Matters

Clustering doesn't "know" what matter, we tell it via distance.

- The **choice of distance metric** can lead to **entirely different clusters**.
- Different applications require different definitions of similarity. For example, in customer segmentation, cosine similarity may be more appropriate than Euclidean distance, because we care about *purchasing trends*, not *absolute values*.

So instead of asking "*what's the best distance?*", ask: "***what kind of similarity is meaningful in our application?***"

2.3 Clustering Validity

Clustering can produce a result, a partition of our data, but **how do we know if it's good**? Unlike supervised learning, clustering lacks ground truth, so evaluating the quality of clustering is challenging. There are three main families of evaluation metrics, each answering a different kind of question: External Metrics, Internal Metrics, Relative Metrics.

2.3.1 External Metrics

External metrics assess the quality of a clustering by **comparing it to known class labels**. They answer the question: "how similar is the clustering result to the actual classification?". This is **possible only when ground truth labels[2] are available**, which is often the case in benchmarking or simulated data. However, these metrics are **not available in real-world applications where true labels are unknown**.

We usually **compare clusters** to classes using a **contingency table**:

	Class 1	Class 2	Class 3	Total
Cluster 1	m_{11}	m_{12}	m_{13}	m_1
Cluster 2	m_{21}	m_{22}	m_{23}	m_2
Cluster 3	m_{31}	m_{32}	m_{33}	m_3
Total	c_1	c_2	c_3	n

- m_{ij} number of points from class j assigned to cluster i
- m_i total points in cluster i
- c_j total points in class j
- n total number of points

From this, we compute **frequencies**:

$$p_{ij} = \frac{m_{ij}}{m_i} \tag{28}$$

[2]**Ground Truth Labels** are the true, correct categories or values assigned to data points. They represent the known answer. In clustering, ground truth refers to the real classification or grouping of our data, which is often manually annotated, observed from reality, or known from the context.

Main External Metrics

- **Entropy Metrics**. Measures **class diversity within each cluster**.
  - A **pure cluster** (all elements from the same class) has entropy 0.
  - A **mixed cluster** hsa higher entropy (maximum when classes are equally mixed).

  The formula for **cluster** i:

$$e_i = -\sum_{j=1}^{L} p_{ij} \log(p_{ij}) \tag{29}$$

  Overall clustering entropy:

$$e = \sum_{i=1}^{K} \frac{m_i}{n} e_i \tag{30}$$

  Entropy decreases with better alignment. The **ideal value is zero**.

- **Purity Metrics**. Measures the **proportion of the dominant class in each cluster**. Like a "best guess" correctness. The formula for **cluster** i:

$$p_i = \max_j (p_{ij}) \tag{31}$$

  Overall clustering purity:

$$\text{purity} = \sum_{i=1}^{K} \frac{m_i}{n} p_i \tag{32}$$

  Purity increases with better alignment. The **ideal value is one**.

- **Precision Metrics** and **Recall Metrics**. These are adapted from classification metrics:
  - Precision (for cluster i, class j):

$$\text{Prec}(i,j) = \frac{m_{ij}}{m_i} \tag{33}$$

    Among the points in cluster i, **how many truly belong to class** j?
  - Recall (for cluster i, class j):

$$\text{Rec}(i,j) = \frac{m_{ij}}{c_j} \tag{34}$$

    Among all points from class j, **how many are captured by cluster** i?

- **F-Measure Metrics**. Combines precision and recall into a **single score** using the harmonic mean.

$$F(i,j) = \frac{2 \cdot \text{Prec}(i,j) \cdot \text{Rec}(i,j)}{\text{Prec}(i,j) + \text{Rec}(i,j)} \tag{35}$$

  We can aggregate these across all clusters to get a **global F-score** for the clustering.

Metric	Ideal	Interprets as...
Entropy	0	Purity of cluster (lower is better)
Purity	1	Dominance of single class (higher = better)
Precision	1	Cluster doesn't mix in wrong classes
Recall	1	Class is fully captured by a cluster
F-measure	1	Balanced precision and recall

Table 4: Summary External Metrics.

2.3.2 Internal Metrics

When class labels (ground truth) are not available, which is the most common case in unsupervised learning, we need **Internal Metrics** to assess:

- How **cohesive** each cluster is (tightness of points within clusters).
- How **well-separated the clusters are from each other**.

These metrics measure **structural quality** based on distances between two points.

- **Sum of Squared Errors (SSE)**. Also called **Within-Cluster Sum of Squares (WCSS)**.

  What is measures

  * **How far** the **points** in each cluster are from their **cluster center** (or medoid)
  * **Lower SSE means tighter clusters**[3]

  Formula

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} ||x - c_i||^2 \tag{36}$$

  Where:

  * C_i is cluster i
  * c_i is the center of cluster i

  Used to evaluate compactness: a good clustering should have a small SSE.

[3] A tight cluster means that the data points inside the cluster are close to each other. They are packed together, not scattered.

- **Elbow Method**
  - **What is measures**
    - Used with SSE to find the **optimal number of clusters** K.
  - **Procedure**
    - Run clustering for various values of K (e.g., 1 to 30)
    - Plot $SSE(K)$ vs K.
    - Find the "elbow" point: the value of K where the SSE stops decreasing sharply.

  In other words, adding more clusters beyond the elbow gives **diminishing returns**.

- **Silhouette Coefficient**
  - **What is measures**
    - **Combines cohesion and separation** into a single score for each point.
    - For each data point:
      - a: average distance to points in **same cluster**
      - b: average distance to points in **nearest other cluster**
  - **Formula**

$$s = \frac{b - a}{\max(a, b)} \qquad s \in [-1, 1] \tag{37}$$

  - ✓ $+1$ well clustered
  - 0 on the border
  - ✗ -1 misclassified

  The **average silhouette score** across all points is **used to evaluate the whole clustering**.

- **Between-cluster Sum of Squares (BSS)**. Measures **cluster separation**:

$$BSS = \sum_i m_i \left|\left|c_i - \bar{c}\right|\right|^2 \tag{38}$$

  Where:

  - m_i: size of cluster i
  - c_i: center of cluster i
  - $\bar{c}$: global centroid

  A **good clustering** has **low WCSS** (or SSE) and **high BSS**.

⚠ Limitation of Internal Metrics

Most clustering algorithms **don't explicitly optimize internal metrics**. As a result, the **best-looking clustering under one metric may not be optimal in another**. For example, K-Means minimizes WCSS but may produce poor separation between clusters.

We can **design custom clustering algorithms that directly optimize an internal metric**, but this is not always practical or generalizable. A famous quote from Jain and Dubes (1988): "The validation of clustering structures is the most difficult and frustrating part of cluster analysis. Without a strong effort in this direction, clustering remains a black art accessible only to those true believers who have experience and great courage.". This reflects **how tricky and interpretation-heavy clustering evaluation can be**.

References

[1] Herman Chernoff. The use of faces to represent points in k-dimensional space graphically. *Journal of the American statistical Association*, 68(342):361–368, 1973.

[2] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics. Springer New York, 2009.

[3] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: with Applications in Python*. Springer Texts in Statistics. Springer New York, 2013.

[4] R.A. Johnson and D.W. Wichern. *Applied Multivariate Statistical Analysis*. Applied Multivariate Statistical Analysis. Pearson Prentice Hall, 2007.

[5] Beraha Mario. Applied statistics. Slides from the HPC-E master's degree course on Politecnico di Milano, 2024-2025.

Index

Q

R

S

U

V

W