Data Analysis and Decision Making

4th Year - Data Analysis - 2025

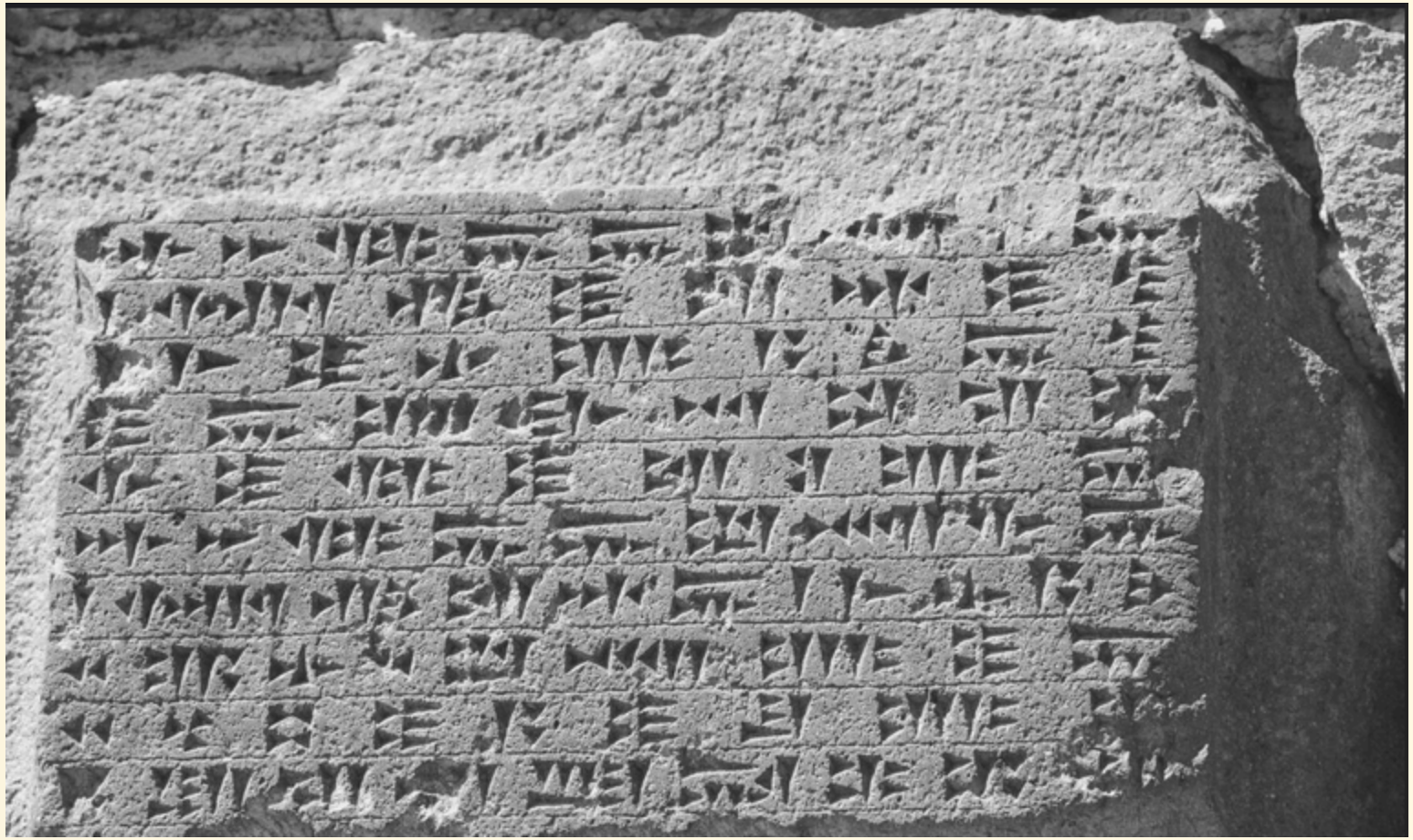
Alexandre Gazagnes

1. Introduction to -- *REAL* -- Data Analysis

1.1 A Brief History of the Data Revolution

1.1.1 From Writing to the Internet

1. **Writing Invention - Mesopotamia**: The earliest form of data recording—cuneiform tablets used to track trade, taxes, and inventories.



2. **Gutenberg's Printing Press**: Made books accessible, empowering the spread of knowledge and enhancing religious and political influence.

3. **Telegraph / Phone**: Instant communication across distances revolutionized commerce and personal connectivity.

4. **The Internet**: The advent of websites, email, and social media made data ubiquitous, driving modern global communication and business.

1.1.2 Data as a Source of Power and Wealth

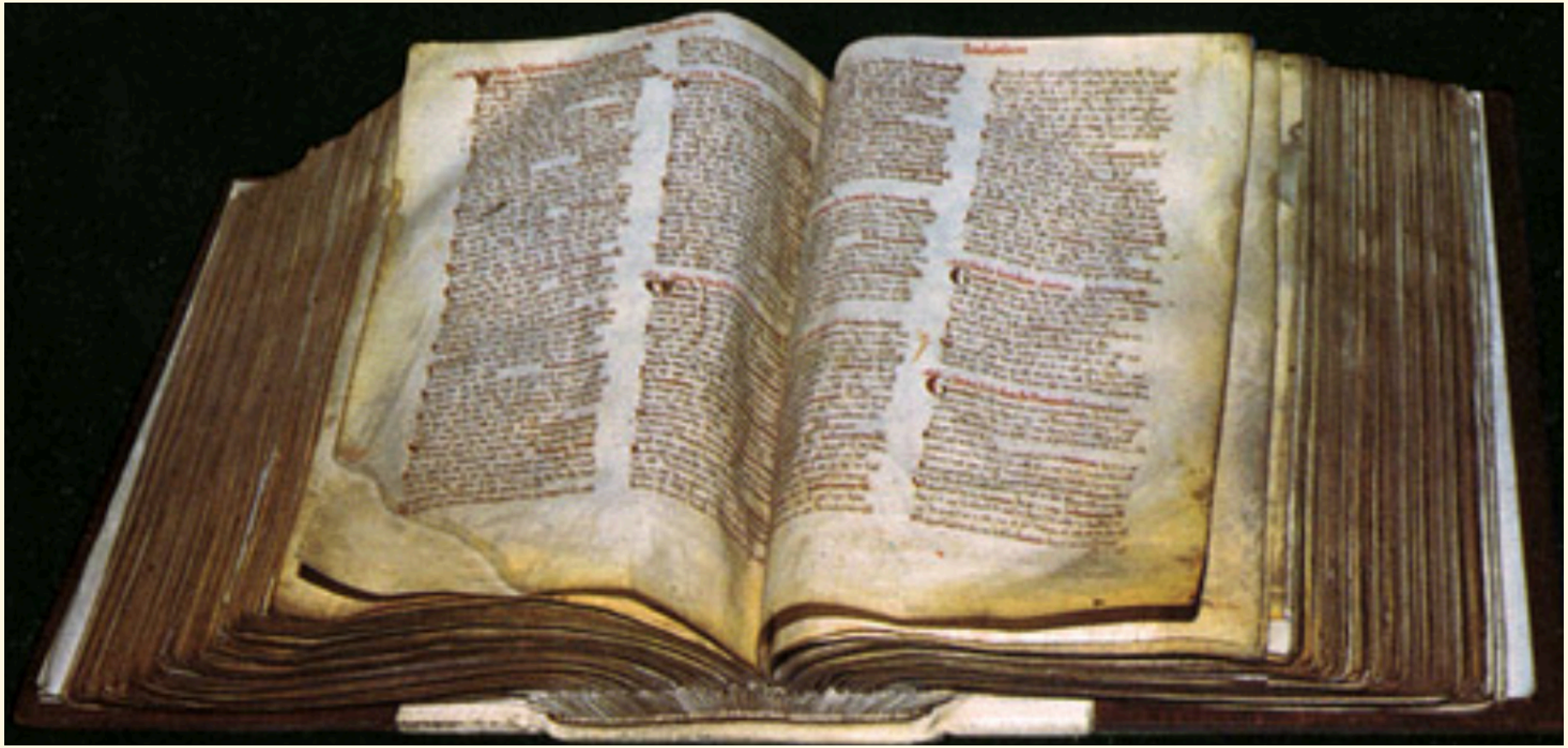
- **Data → Insight → Decision → Value**:
  - Analyze situations and options.
  - Make informed decisions.
  - Drive actions that generate power and wealth.

- **The Double-Edged Sword**:
  - Wrong data can lead to successful outcomes by luck.
  - Good data and analysis might fail due to unforeseen circumstances.

1.1.3 Short Stories from Data History

1. **Mesopotamian Trade Records**: Early accounting systems to track wealth.

2. **Domesday Book (England)**: Commissioned by William the Conqueror to assess land and taxes.

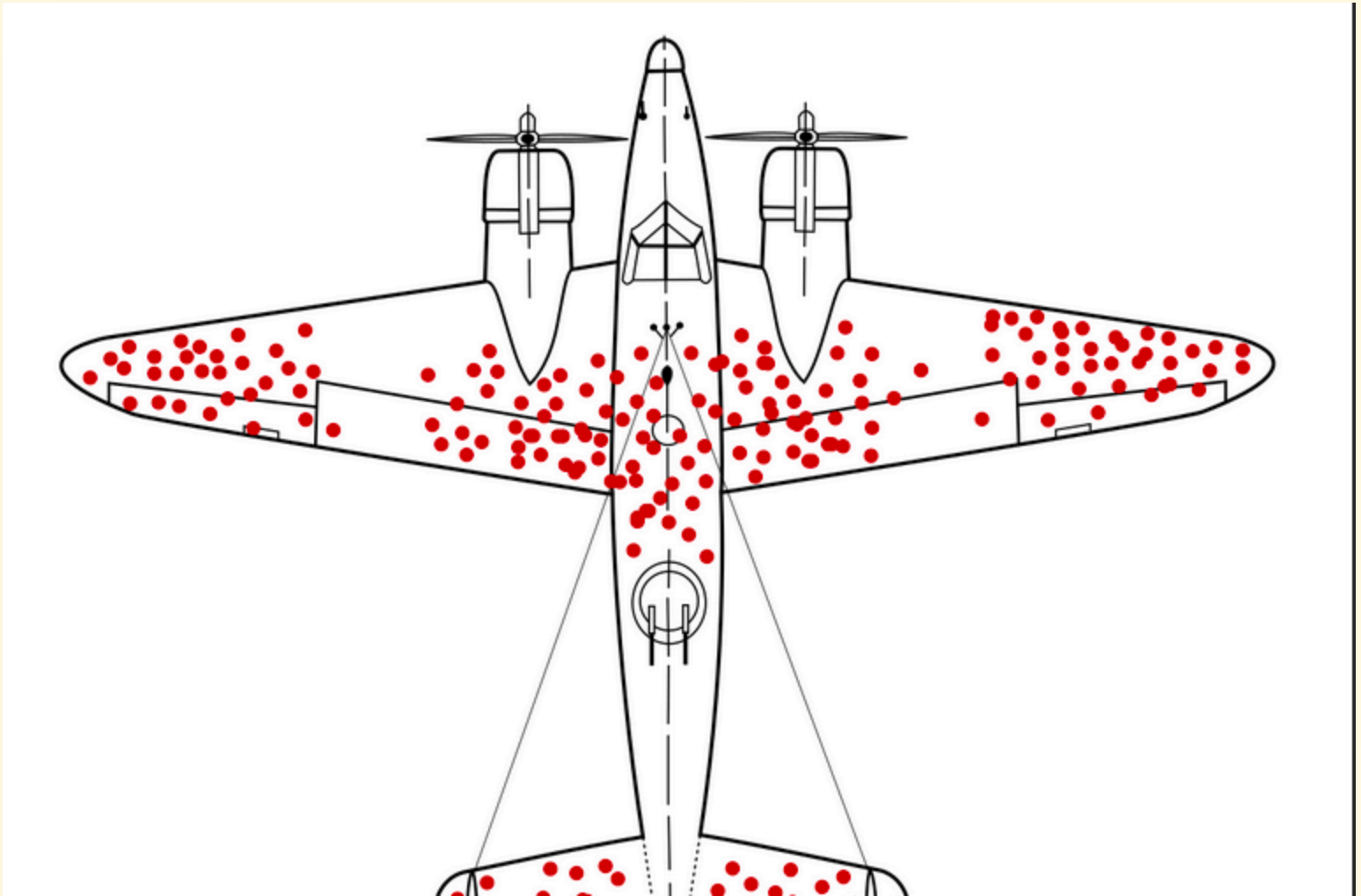


3. **Nurse in Wartime**: Florence Nightingale's use of data visualization to highlight sanitary conditions over battlefield injuries.

4. **London Cholera Epidemic**: John Snow's mapping of cholera clusters linked to contaminated water sources.

1.2 Misinterpretations of Data

1.2.1 WWII Plane Armor



- Observing bullet holes on returning planes led to the assumption armor should be placed there.
- **The Insight**: Focus on parts where planes didn't sustain damage but were still crucial to survival—the non-returned planes provided key missing data.

1.2.2 Being Fooled by Data:

- Common cognitive biases:
  - **Confirmation Bias**: Seeking data that confirms existing beliefs.
  - **Availability Heuristic**: Overvaluing easily accessible information.
  - **Survivorship Bias**: Focusing on successful outcomes while ignoring failures.
  - **Loss Aversion**: Overvaluing potential losses over gains.
  - **Sunk Cost Fallacy**: Justifying continued investment in a failing endeavor.
  - **Anchoring**: Relying too heavily on the first piece of information.

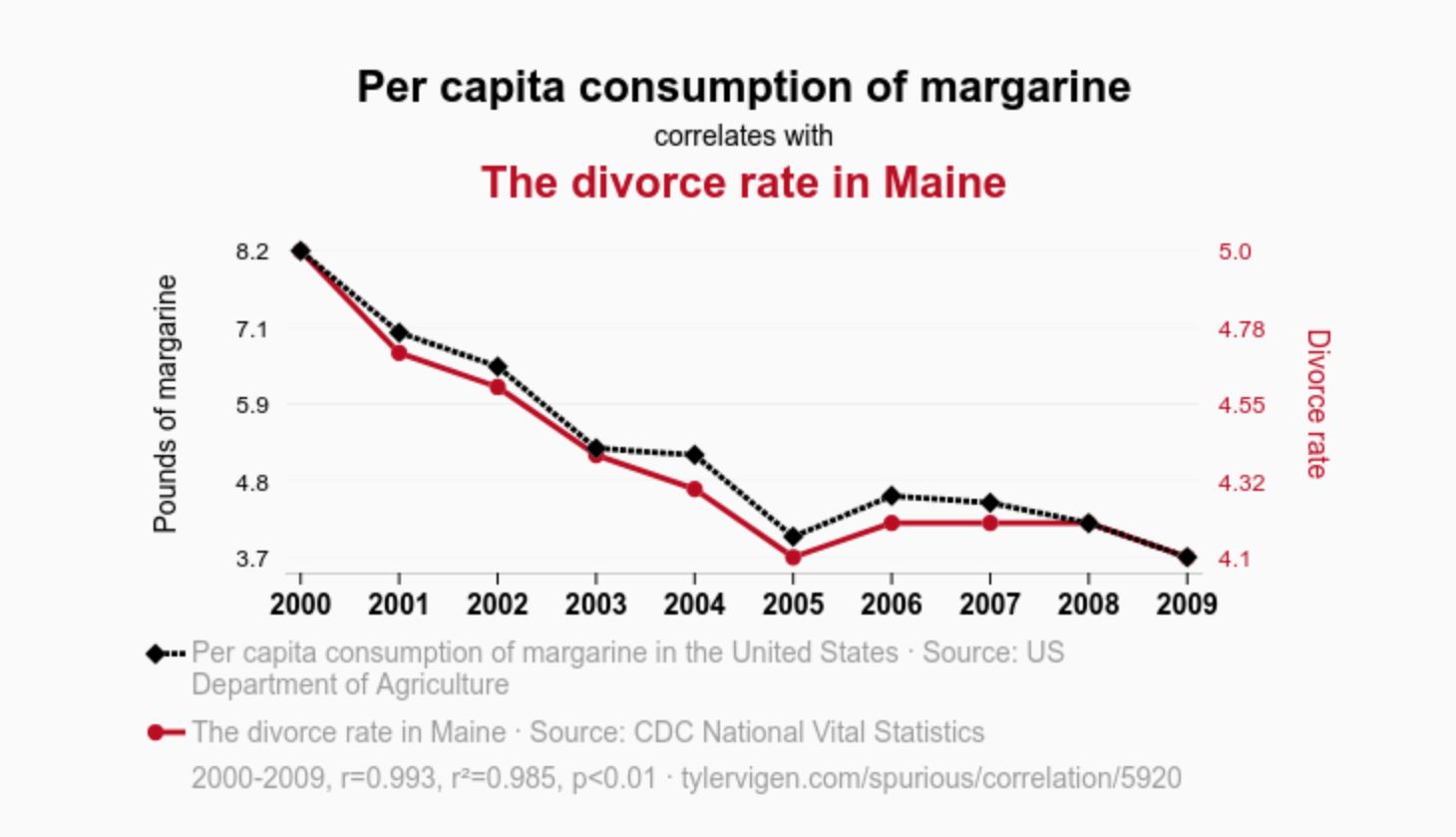
1.2.3 Occam's Razor:

To Apply Occam's Razor:

1. Determine how many assumptions and conditions are necessary for each explanation to be correct.
2. If an explanation requires extra assumptions or conditions, demand evidence commensurate with the strength of each claim.
3. Extraordinary claims require extraordinary evidence.

- "The simplest explanation is usually the best." MISSUNDERSTOOD
- Be cautious of convoluted interpretations.
- Simplest theory are always easiest to refute

1.2.4 Spurious Correlations and Hidden Variables



https://www.tylervigen.com/spurious-correlations

Correlation but no causation

Correlation but no causation

Correlation but no causation

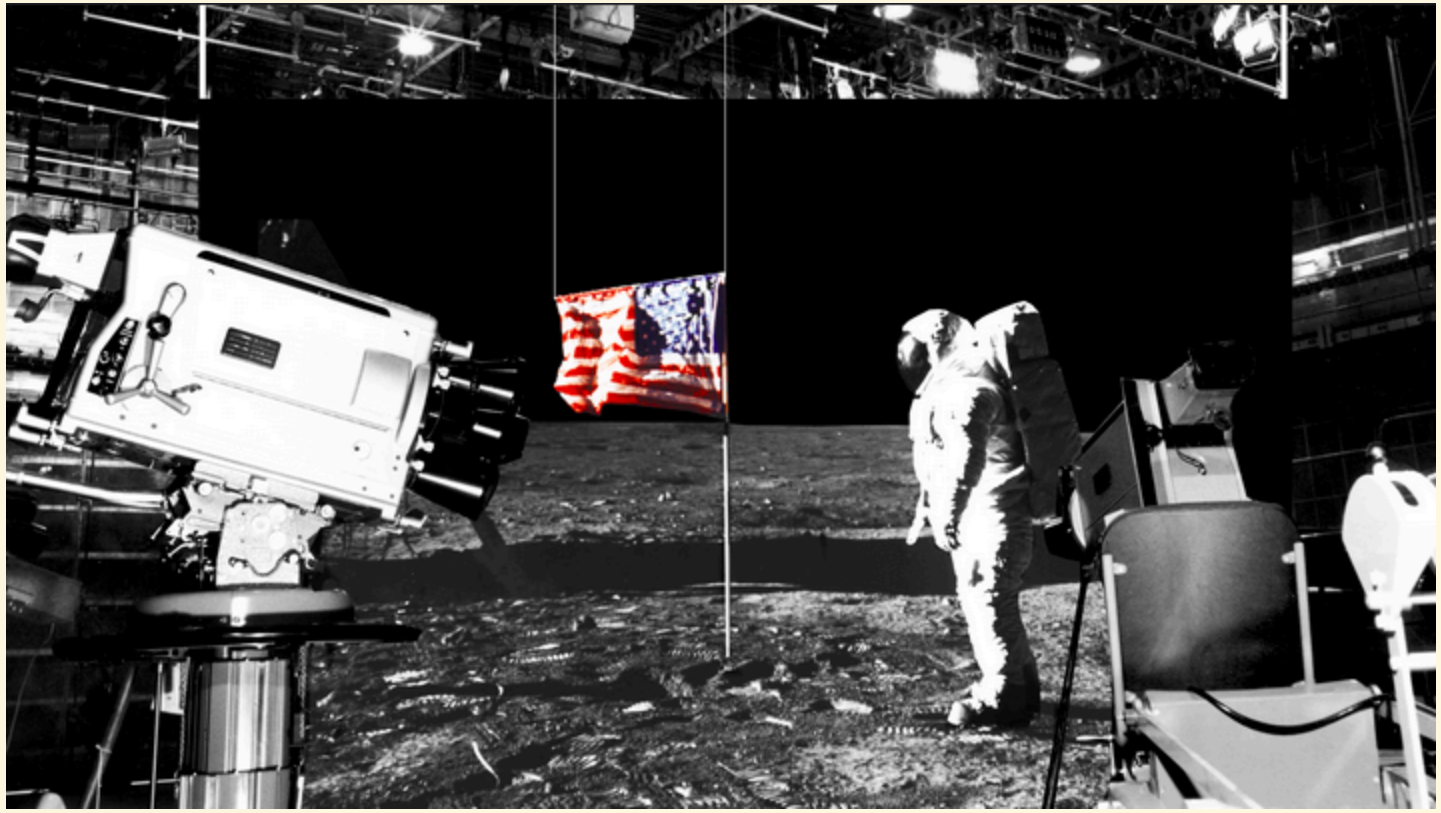
1.2.5 Famous Data Misunderstanding

- A shark attack in a small town.
- Mayor worried about the business.
- Major operation to kill **THE** shark, Shark was killed!
- Mayor says "We're done with **THE** shark"
- Scientist says "We're not done with **THE** shark, We're done with **ONE** shark"
- Mayor says "Come on, fool, stop with your science, we killed **THE** shark."
- Scientist says, let me show you **THE** data, we're not done with **THE** shark, we're done with **ONE** shark"

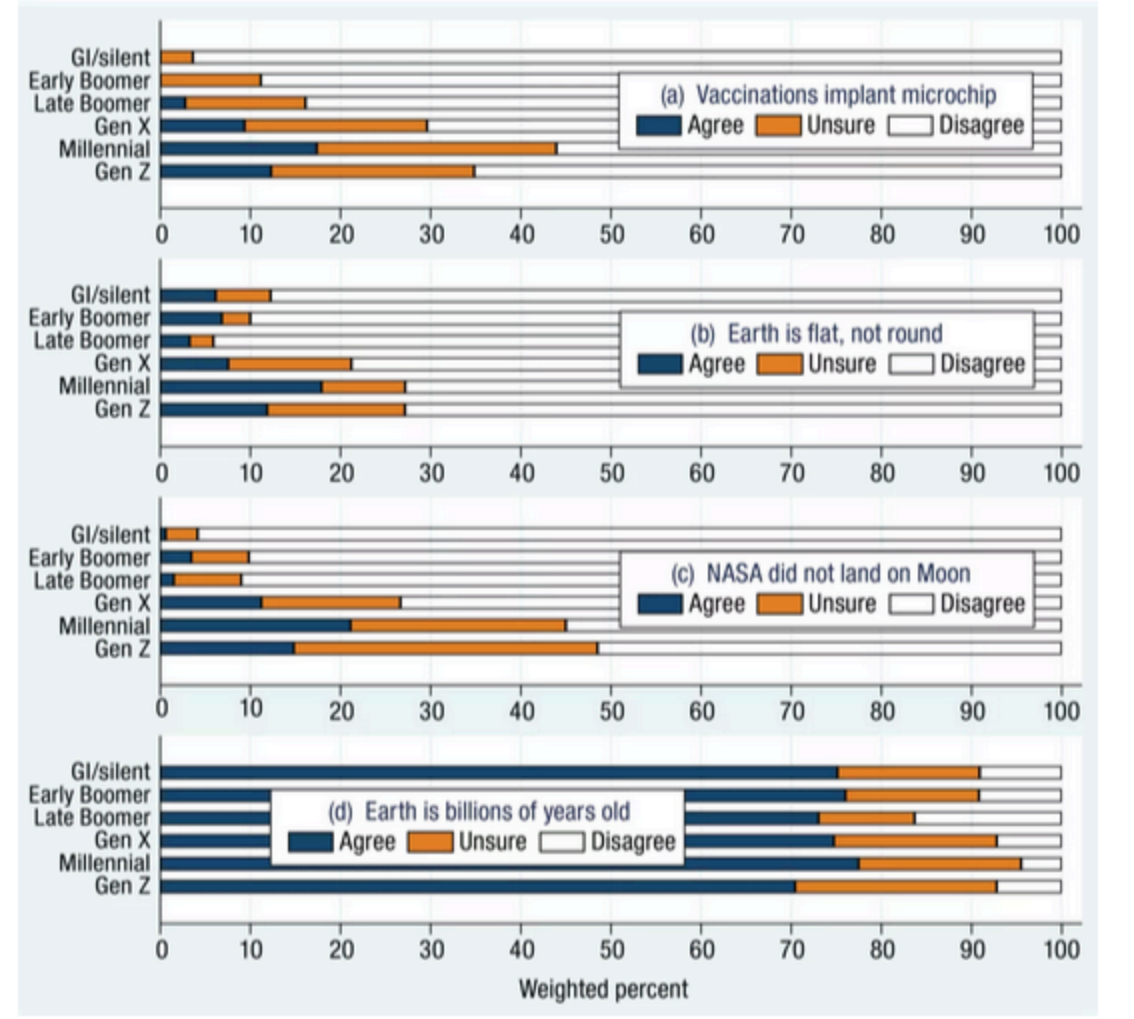
1.3 Living in a Post-Truth World

1.3.1 Somes stories false but not critical

1.3.2 Somes stories false and critical



50% of the population at least "not" sure in US Gen Z



1.3.3 Somes "conspiracies" migh be true

Colin Powell, 2003, UN, "blufing" like pro poker player :

1.3.4 New challenges for everyone

- Where is THE truth ?
- AI-generated videos and audio clips can manipulate reality.
- Tons of short videos, generated by IA with huge click rate
- Origanized farms to generate fake news and manipulate public opinion

1.4 Conclusion

By understanding the power of data, historical lessons, and the risks of misinterpretation, we can develop a more informed, critical approach to decision-making in a data-driven world.

2. Not so Basic Reminder about Not so Basic Data Analysis

2.1 Data is not about mean, IQ or stat test?

No

No

No

No

Data → Insight → Decision → Value

Think in reverse:

- What is my purpose?
  - Optimizing **n**, finding solutions.
  - Business, money, medical goals.
- What is my problem?
- **What do I know about my problem?**

2.2 Getting Some Relevant Data

- What data might/should I want?
- Where to find it?
- What variables could I find (internet/open data/in my company)?
- Is it relevant? Too old? Too many? Trustable? Related?

2.3 Being Critical with Data

- How many variables are there?
- Can I trust them?

Examples:

- User database: someone listed as 198 years old?
- Many users 1750m tall? (Maybe data is in cm, in mm?)
- dataset with 95% of missing values?

2.4 First Look at the Data

2.4.1 Reading the Data

- **Display the Data**:
  - Top 10 rows
  - Last 10 rows
  - Random 10 rows
- **Shape of the Data**

Take the time to look at the data

Let your brain make some inferences (it’s pretty good at this)

2.4.2 Data Types and First Insights

- **Data Types**:
  - Numerical:
    - Continuous
    - Discrete
  - Categorical:
    - Ordinal
    - Cardinal

Tip: Convert ordinal categorical data to numerical if needed.

2.5 Missing Values and Outliers

- **Identify Missing Values**:
  - Count NaN values.
  - Missing values (%) by row/column.
- **Examine Outliers**:
  - Min / Max values.
  - Use summary stats and visualizations:
    - Mean / Standard Deviation / Density Function
    - Median / IQR / Box Plots

2.6 Preparing / cleaning the data

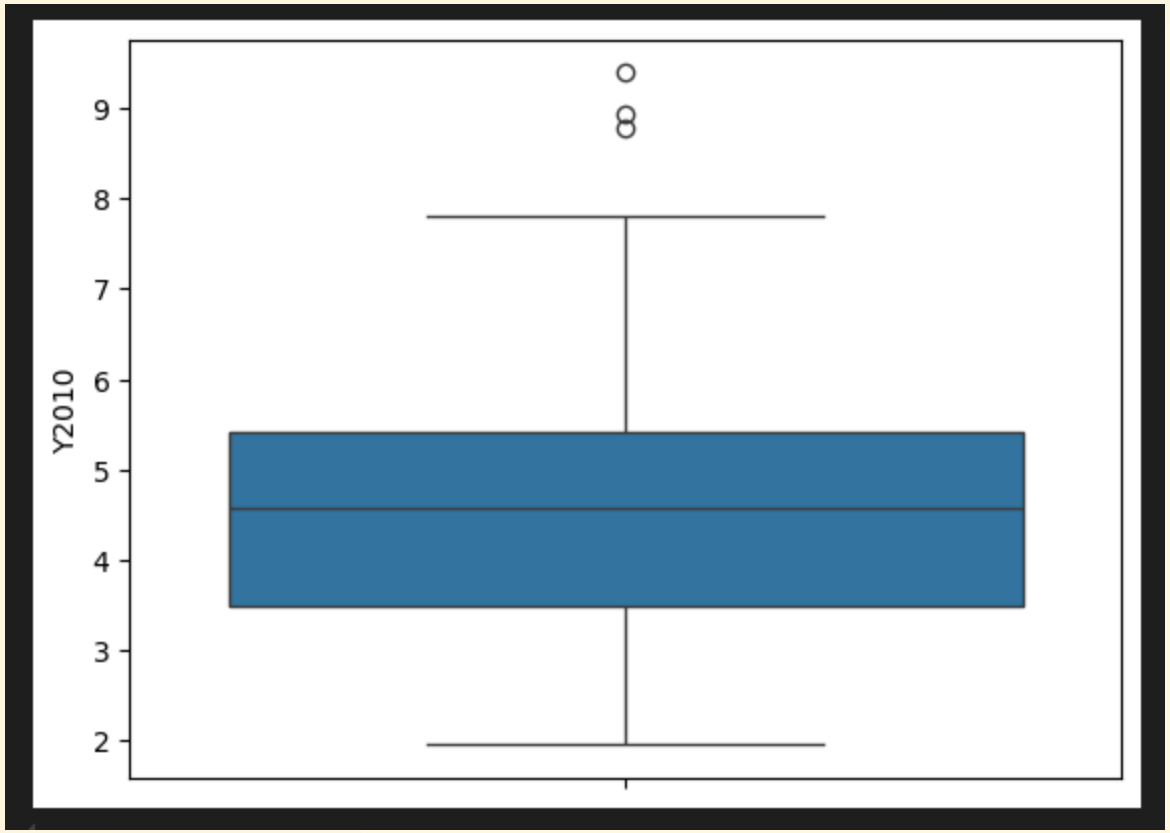
- Drop columns / lines with too many missing values
- Select subset of columns / lines with relevant data for your analysis
- Fill missing values with mean, median, mode, or other values if relevant
- Transform categorical ordinal data into numerical data if relevant
- Apply log, square root, or other transformations to numerical data if relevant
- Remove outliers if relevant
- Create new columns if relevant

2.7 Univariate Analysis

2.7.1 Numerical Data

- Descriptive statistics.
  - Mean / Standard Deviation / Density Function
  - Median / IQR / Box Plots

Difference between mean and median?



- Graphs:
  - Box Plots
  - Density Distributions

2.7.2 Categorical Data

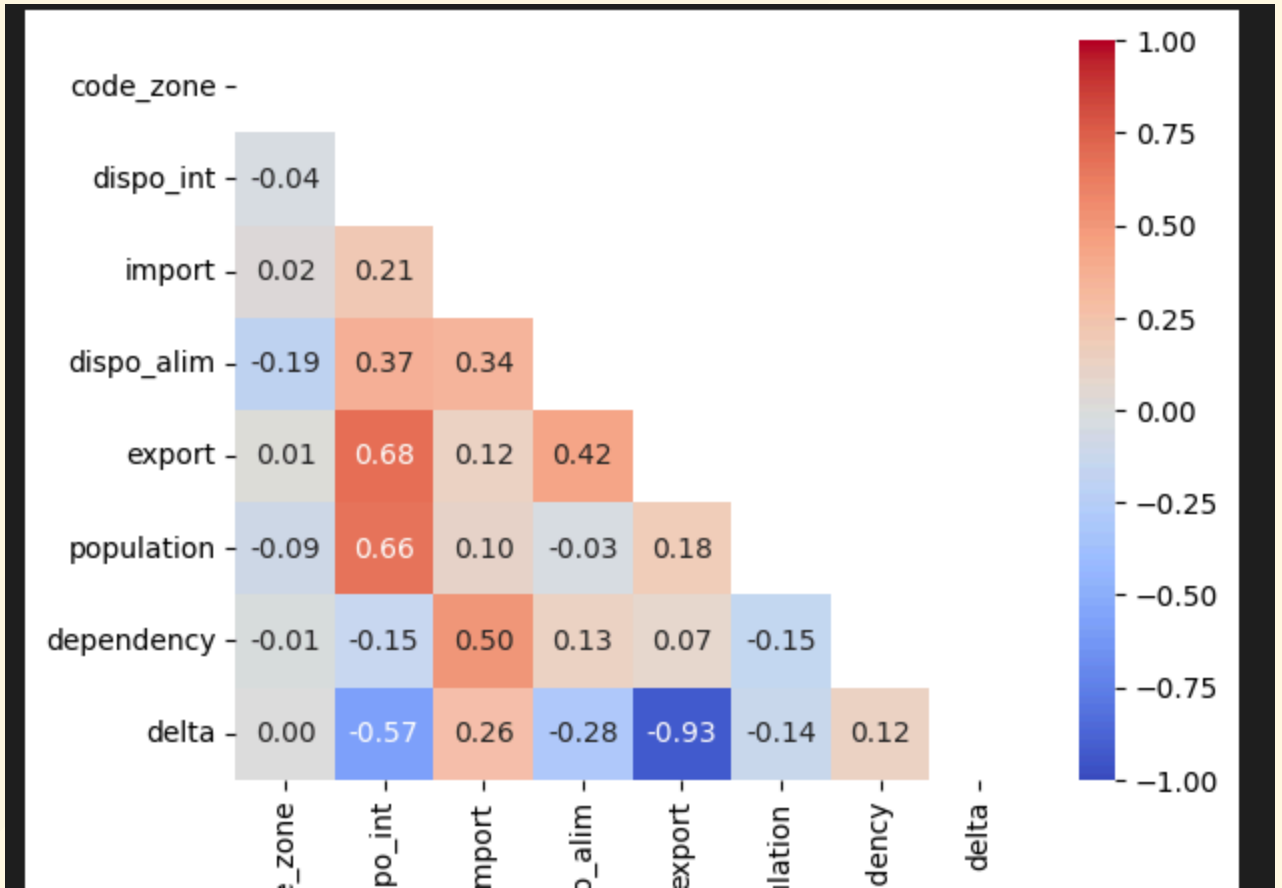
- Descriptive statistics.
  - Count of categories.
  - Percentages per category.
- Graphs:
  - Bar Plot
  - Pie Chart

2.8 Bivariate/Mulivariate Analysis

2.8.1 Bivariate Analysis

- **Numerical vs. Numerical**:
  - Scatter Plots
  - Correlation Coefficient
- **Categorical vs. Numerical**:
  - Box Plots
  - Violin Plots
  - ANOVA

- **Categorical vs. Categorical**:
  - Contingency Tables
  - Chi-Square Test



2.8.2 Multivariate analysis

Correlation matrix :

- Heatmap
- Scatter plot matrix

Question to ask :

- What are the most correlated variables ?
- What does that mean to have 2 variables very highly correlated ?
- What not so easy to interpret variables do we have ?

3. Grouping Data & dimensions

3.1 Curse of dimentionality

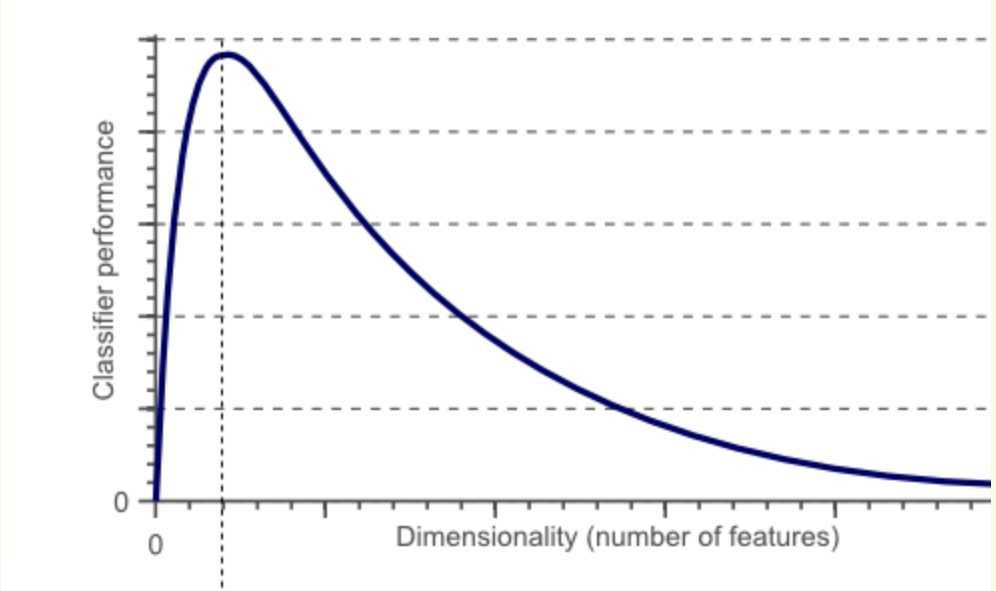
In data :

- More columns = more data = more information
- More columns = more data = more noise

The more columns you have, the more difficult it is to find the signal in the noise

Data selection is key to avoid the curse of dimentionality

Having 100 or 1000 columns is a garantee of having a good dataset ?



Do you really need all of them ?

3.2 Main ideas behind Grouping Data & dimensions

1. **Reduces Complexity**

   High-dimensional data can be difficult to analyze due to the "curse of dimensionality." Reducing dimensions simplifies data, making it easier to understand and process.

2. **Enhances Visualization**

   Dimensionality reduction techniques like PCA (Principal Component Analysis) or t-SNE reduce data to 2D or 3D, enabling visualization of complex datasets.

3. **Facilitates Interpretability**

   A dataset with fewer dimensions is easier to interpret and analyze. It highlights the most important variables contributing to the underlying patterns.

4. **Improves Performance**

   Algorithms like clustering, regression, or classification perform faster on lower-dimensional data. Computational efficiency increases, especially for machine learning models.

Grouping Data & dimensions : 2 main techniques

- Reducing the number of columns ==> ACP /PCA : Principal Component Analysis

- Reducing the number of lines ==> clustering : K-means / DBSCAN / Hierarchical clustering

G3.3 "Grouping the columns" with Principal Component Analysis

- **ACP /PCA : Principal Component Analysis**
  - **Goal**: Reduce the number of dimensions while retaining the most important information.
  - **Method**: Linear transformation to create new uncorrelated variables (principal components).
  - **Applications**: Data visualization, noise reduction, feature extraction, and data compression.

Process

- Prepare the data by standardizing it.
- Computing Explain variance ratio.
- Selecting the number of components.
- Computing new synthetic variables.
- Interpreting the new synthetic variables.
- Transforming the data.
- Projetcion of the data in the new synthetic variables.
- Analysing the results.

Raw data :

	zone	dispo_int	import	dispo_alim	population	dependence	delta
0	Afghanistan	57.0	29.0	5.0	36296000	0.508772	29.0
1	Algérie	277.0	2.0	22.0	41389000	0.007220	2.0
2	Angola	319.0	277.0	35.0	29816000	0.868339	277.0
3	Argentine	1962.0	8.0	182.0	43937000	0.004077	-199.0
4	Australie	1171.0	16.0	192.0	24584000	0.013664	-26.0
...	...	...	...	...	...	...	...
104	Yémen	246.0	78.0	30.0	27834000	0.317073	78.0
105	Zambie	60.0	12.0	11.0	16853000	0.200000	11.0
106	Belgique	152.0	338.0	44.0	11419000	2.223684	-318.0
107	Serbie	90.0	12.0	35.0	8829000	0.133333	5.0
108	Soudan	69.0	2.0	5.0	40813000	0.028986	2.0

109 rows × 7 columns

Standardized data :

```
1  X_scaled = pd.DataFrame(X_scaled, columns=X.columns)
2  X_scaled.head()
```

[20] ✓ 0.0s

	dispo_int	import	dispo_alim	population	dependence	delta
0	-0.374105	-0.444496	-1.095340	-0.159395	0.466402	0.106225
1	-0.291244	-0.565846	-0.783851	-0.132269	-0.622814	0.059870
2	-0.275425	0.670125	-0.545653	-0.193908	1.247270	0.532009
3	0.343397	-0.538879	2.147814	-0.118699	-0.629639	-0.285221
4	0.045473	-0.502924	2.331044	-0.221774	-0.608821	0.011798

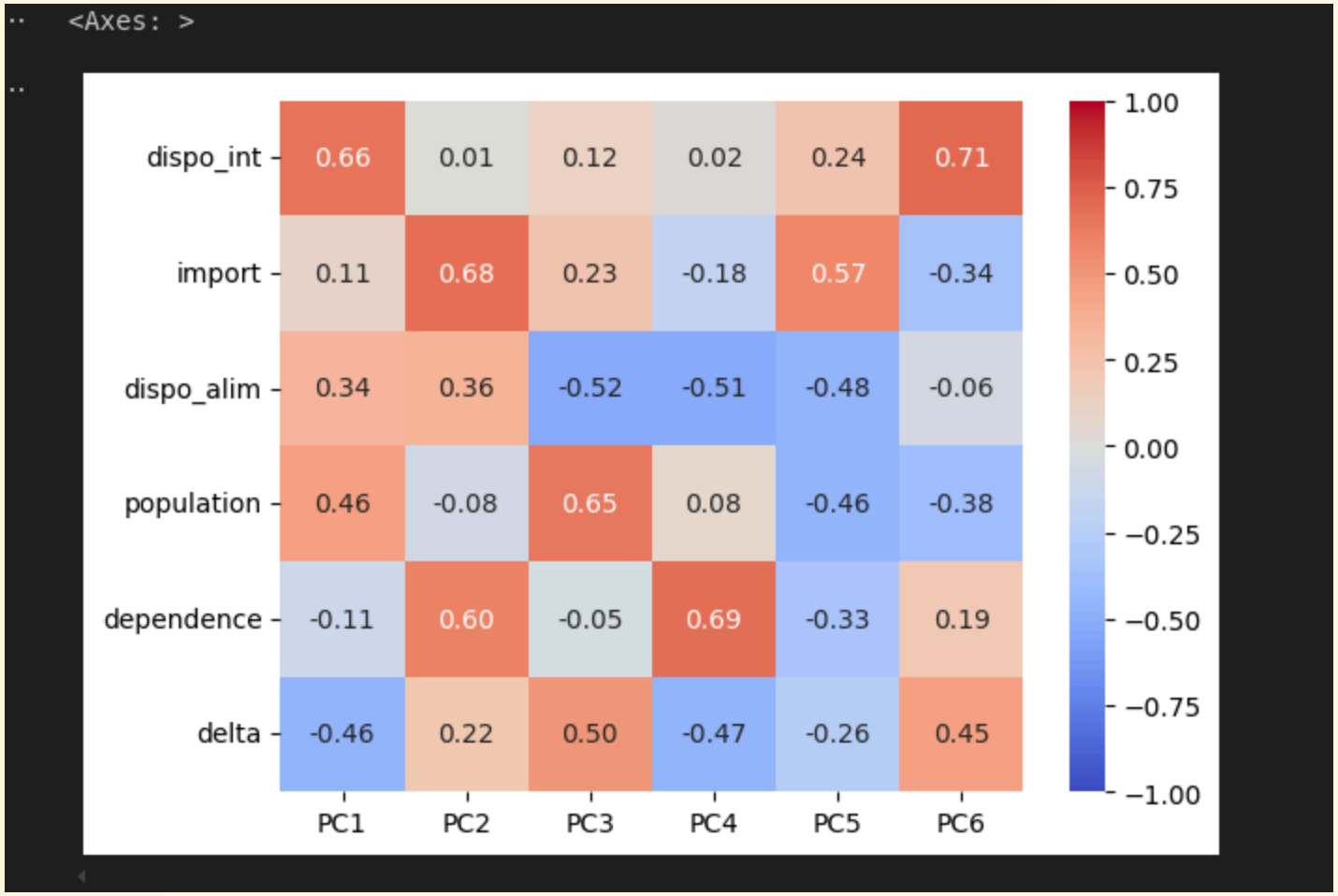
Check that data were scaled:

```
1  X_scaled.describe().round(2)
```

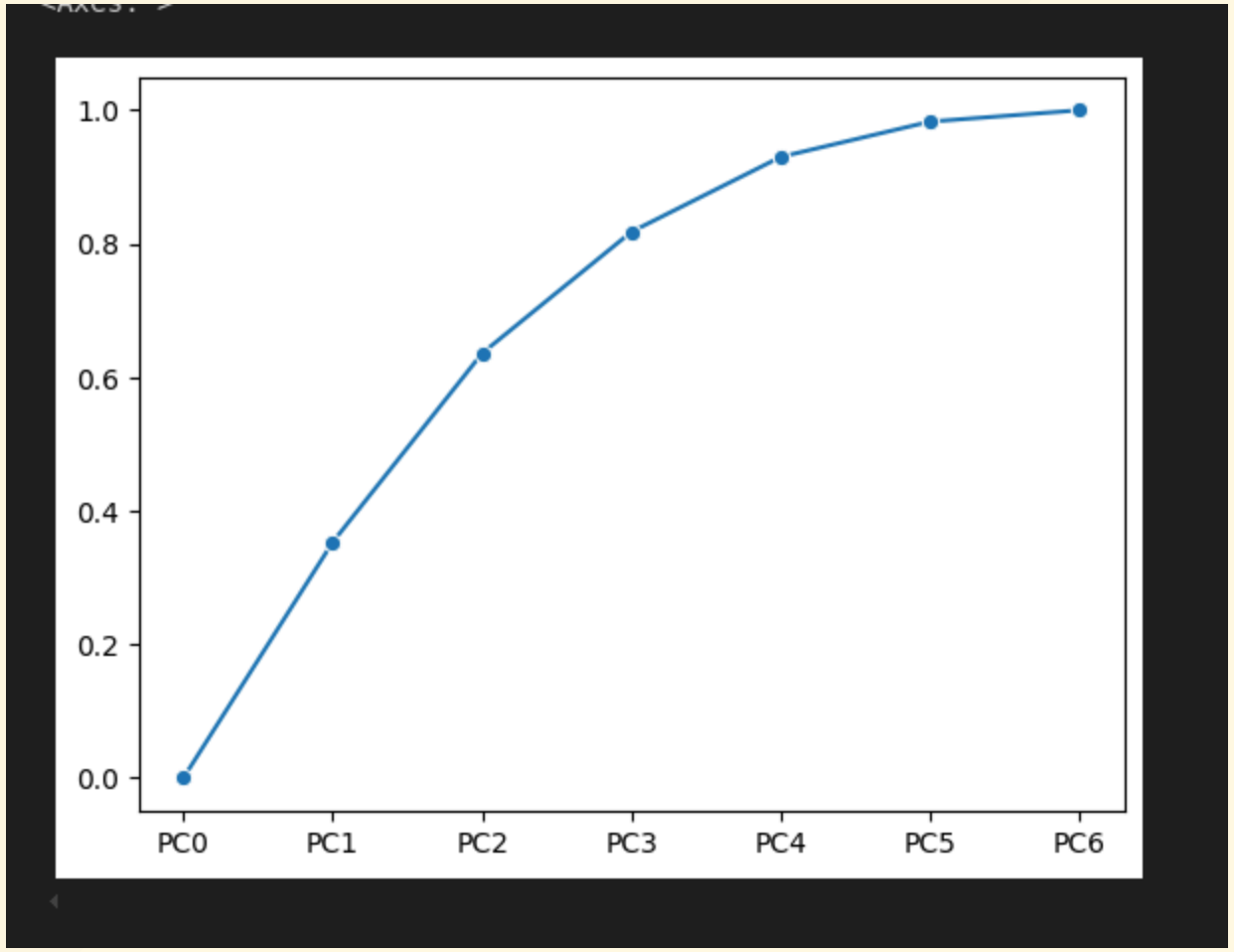
[21] ✓ 0.0s

	dispo_int	import	dispo_alim	population	dependence	delta
count	109.00	109.00	109.00	109.00	109.00	109.00
mean	-0.00	-0.00	-0.00	-0.00	0.00	0.00
std	1.00	1.00	1.00	1.00	1.00	1.00
min	-0.39	-0.57	-1.19	-0.33	-0.64	-7.19
25%	-0.36	-0.57	-0.89	-0.30	-0.62	0.06
50%	-0.30	-0.45	-0.11	-0.24	-0.35	0.07

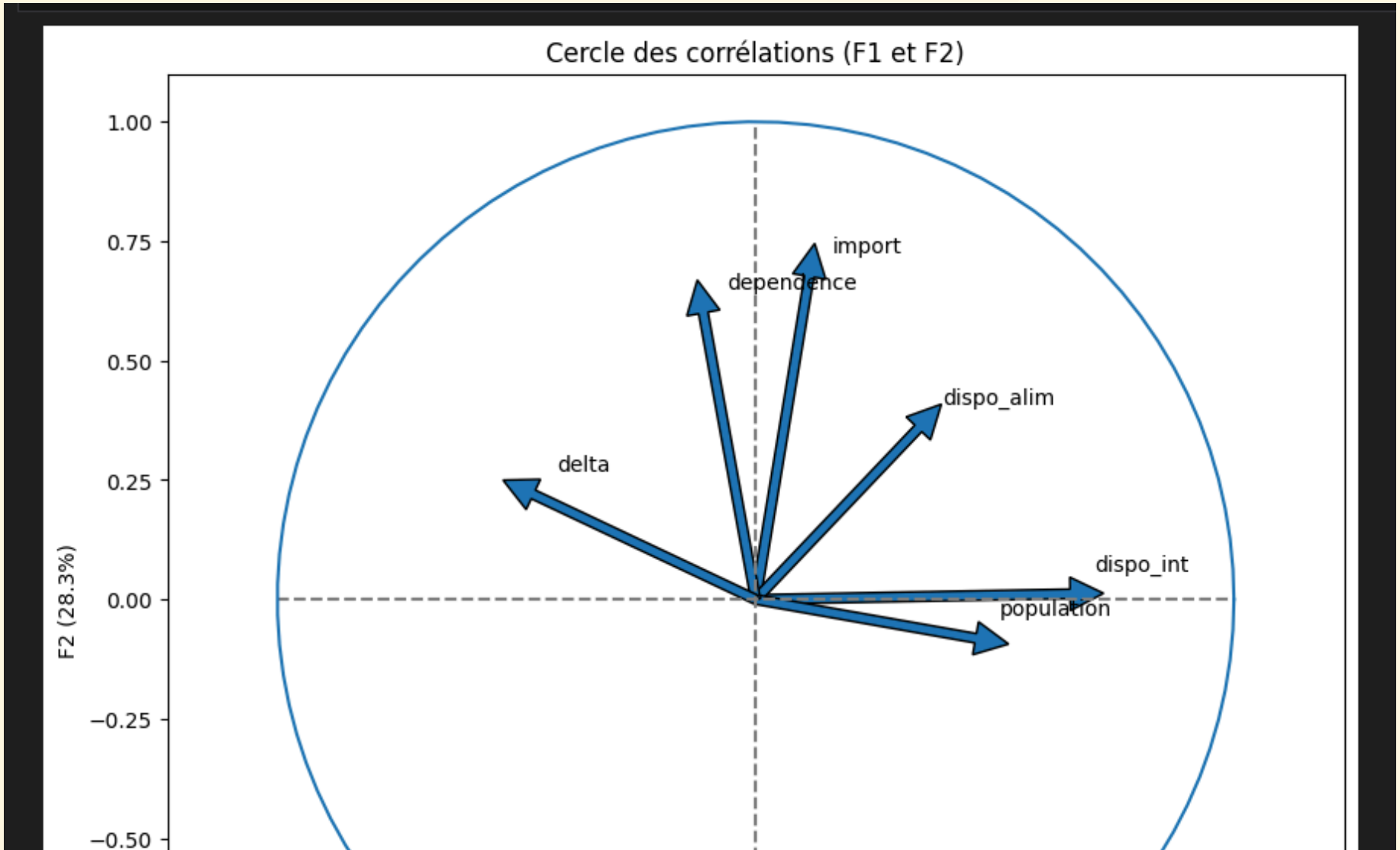
New dimensions :



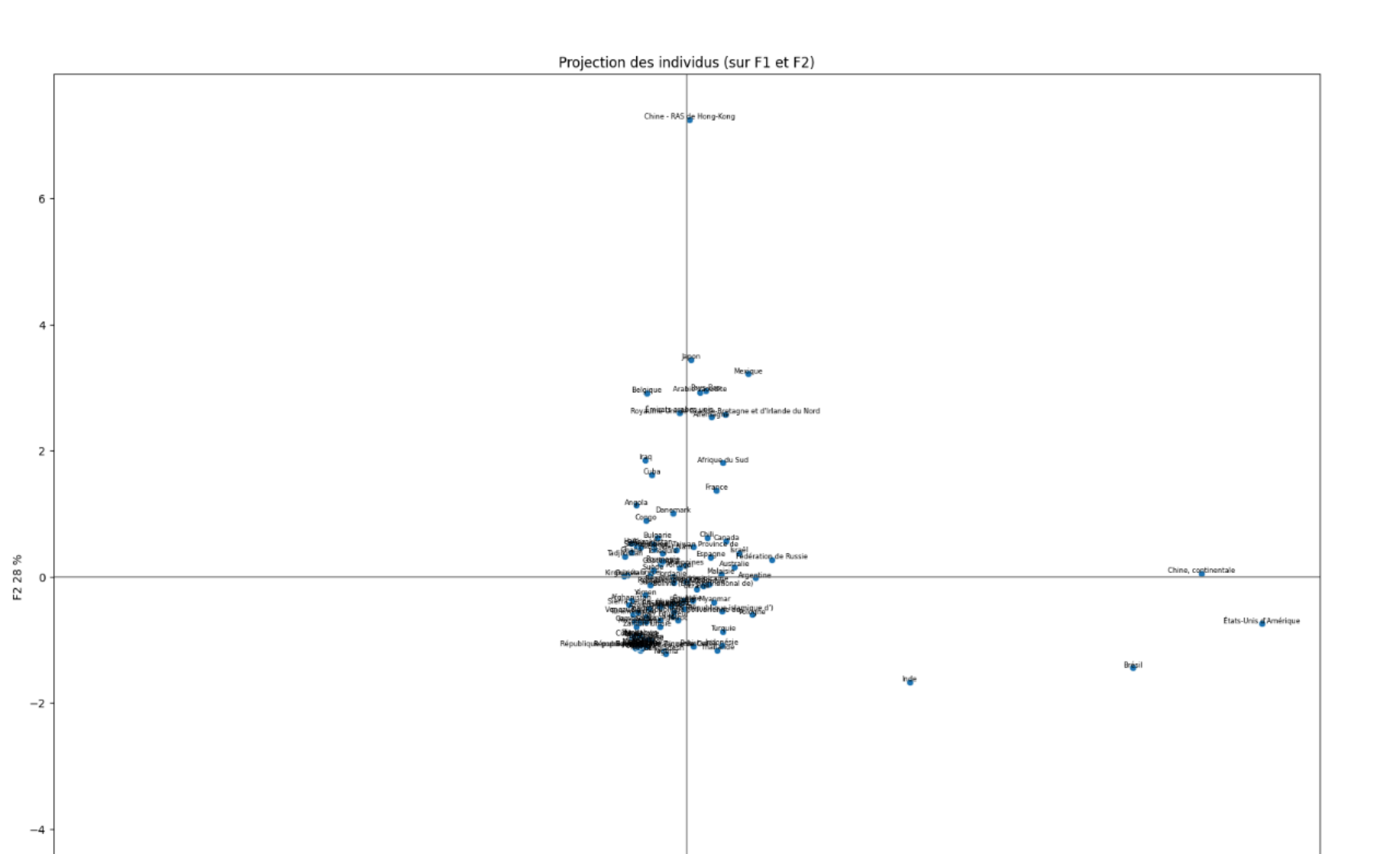
Explained variance ratio :



Correlation between the new dimensions and the old ones :



Projection of the data in the new dimensions :



3.4 "Grouping the lines" with clustering : K-Means

- **K-means**
  - **Goal** : Group data points into clusters based on similarity, minimizing the variance within each cluster.
  - **Method** : Iteratively assigns data points to the nearest cluster centroid and updates centroids to minimize within-cluster variance.
  - **Applications** : Customer segmentation, anomaly detection, pattern recognition, and document clustering.

Process

- Prepare the data by normalizing or scaling it.
- Choose the number of clusters (k) based on prior knowledge or methods like the Elbow Method.
- Initialize the centroids randomly or using optimized techniques.
- Assign data points to the nearest centroid using a distance metric.
- Process data until convergence (when centroids no longer change significantly).
- Analyze the results: Evaluate cluster quality, Plot, Interpret clusters in the context of the problem domain.

Running the Kmeans :

Fit the Kmeans model to the data:

```
1  kmeans.fit(X_scaled)
2  kmeans
```

48] ✓ 0.0s

```
KMeans
KMeans(n_clusters=6, random_state=42)
```

```
1  labels = kmeans.predict(X_scaled)
2  labels
```

49] ✓ 0.0s

```
array([4, 1, 4, 1, 1, 4, 1, 1, 2, 4, 1, 1, 1, 1, 1, 1, 5, 1, 4, 4, 1, 4,
       4, 1, 1, 1, 1, 1, 1, 3, 3, 4, 1, 1, 4, 4, 1, 3, 1, 0, 1, 1, 3, 1,
       1, 1, 4, 3, 1, 4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 3, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 4, 4, 3, 1, 4, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 3, 1, 3, 1, 2, 1, 1, 1, 1, 1, 1, 4, 1, 1],
      dtype=int32)
```

Assigning letters to the clusters :

Use alphanumerical labels for the clusters:

```
labels_values = {
    0: "A",
    1: "B",
    2: "C",
    3: "D",
    4: "E",
    5: "F",
    6: "G",
    7: "H",
    8: "I",
    9: "J",
}

labels = [labels_values[l] for l in labels]
labels[:10]
```

[50] ✓ 0.0s

```
['E', 'B', 'E', 'B', 'B', 'E', 'B', 'B', 'C', 'E']
```

Displaying the clusters :

```
1  _df = df.copy()
2  _df["labels"] = labels
3  _df
```

✓ 0.0s

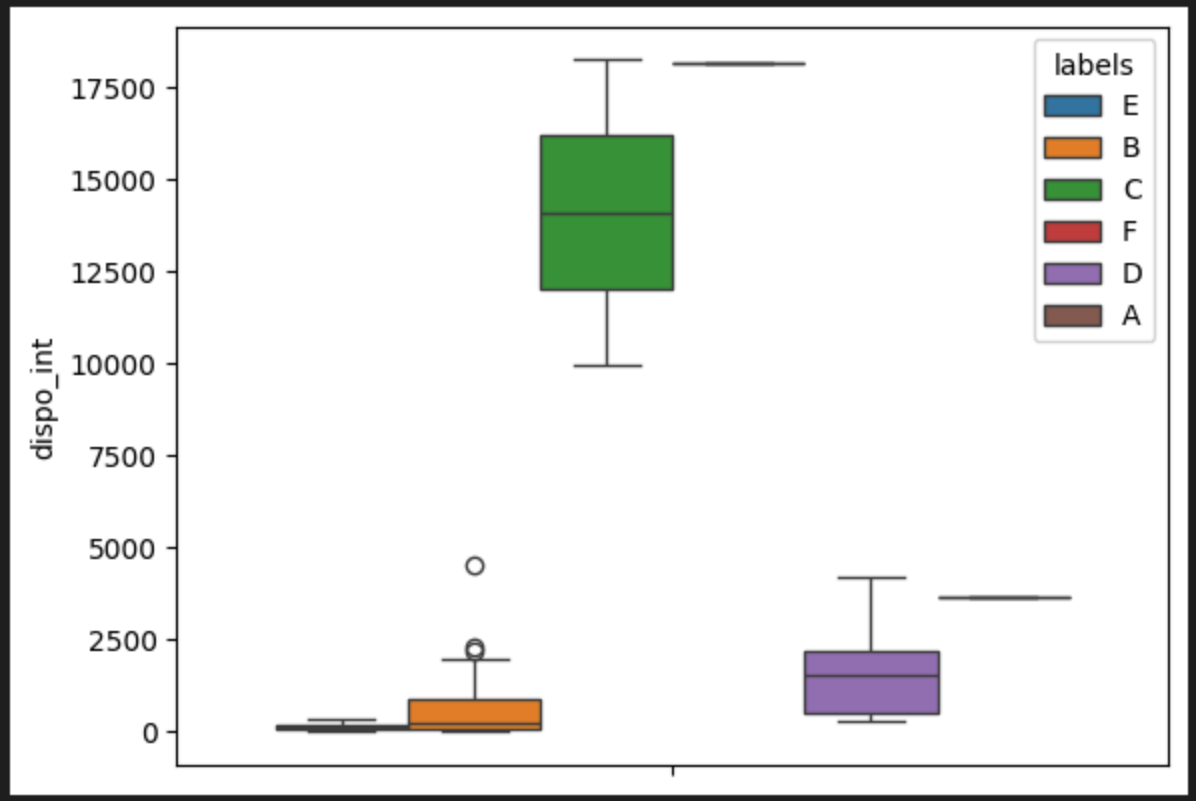
	zone	dispo_int	import	dispo_alim	population	dependence	delta	labels
0	Afghanistan	57.0	29.0	5.0	36296000	0.508772	29.0	E
1	Algérie	277.0	2.0	22.0	41389000	0.007220	2.0	B
2	Angola	319.0	277.0	35.0	29816000	0.868339	277.0	E
3	Argentine	1962.0	8.0	182.0	43937000	0.004077	-199.0	B
4	Australie	1171.0	16.0	192.0	24584000	0.013664	-26.0	B
...	...	...	...	...	...	...	...	...
104	Yémen	246.0	78.0	30.0	27834000	0.317073	78.0	B
105	Zambie	60.0	12.0	11.0	16853000	0.200000	11.0	B
106	Belgique	152.0	338.0	44.0	11419000	2.223684	-318.0	E
107	Serbie	90.0	12.0	35.0	8829000	0.133333	5.0	B
108	Soudan	69.0	2.0	5.0	40813000	0.028986	2.0	B

109 rows × 8 columns

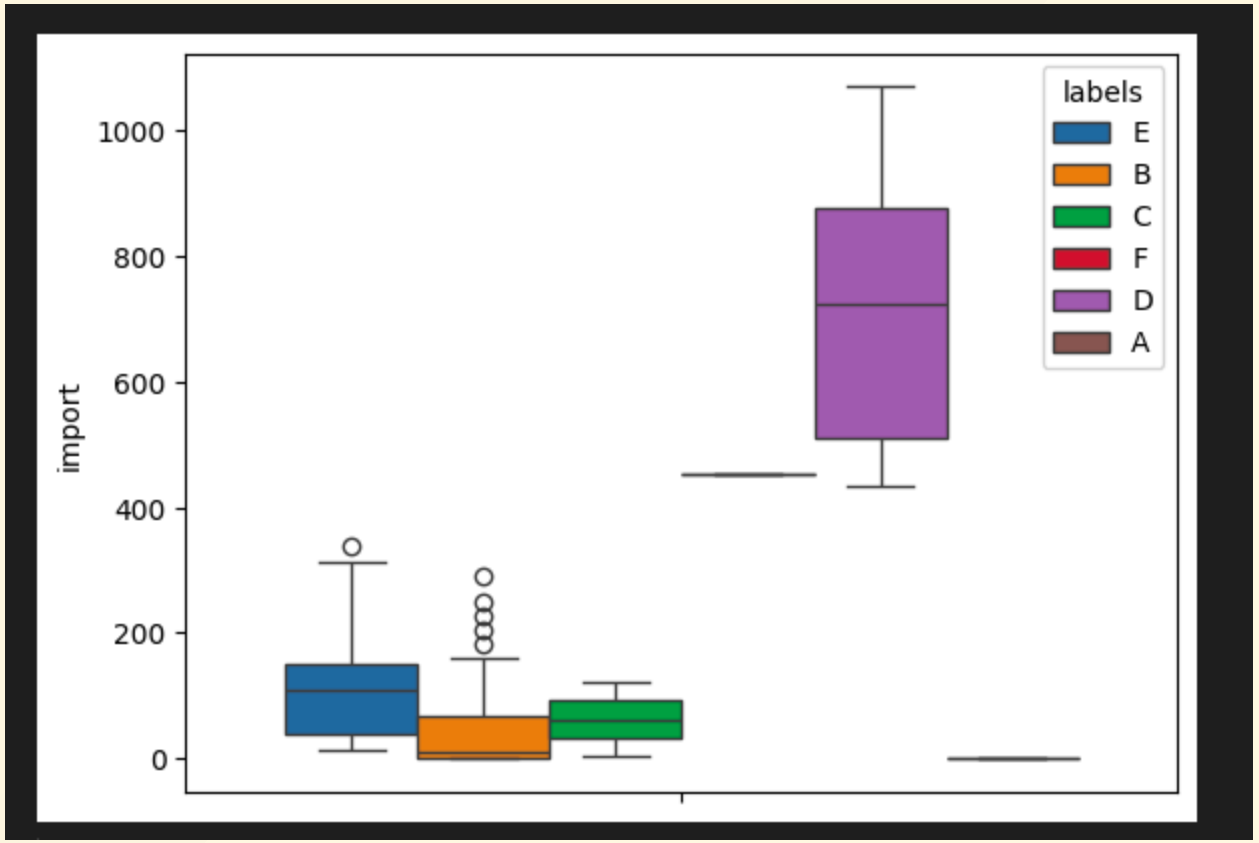
Using Box plot to evaluate the quality of the clusters :

```
for col in _df.select_dtypes(include="number").columns:
    sns.boxplot(data=_df, y=col, hue="labels")
    plt.show()
```

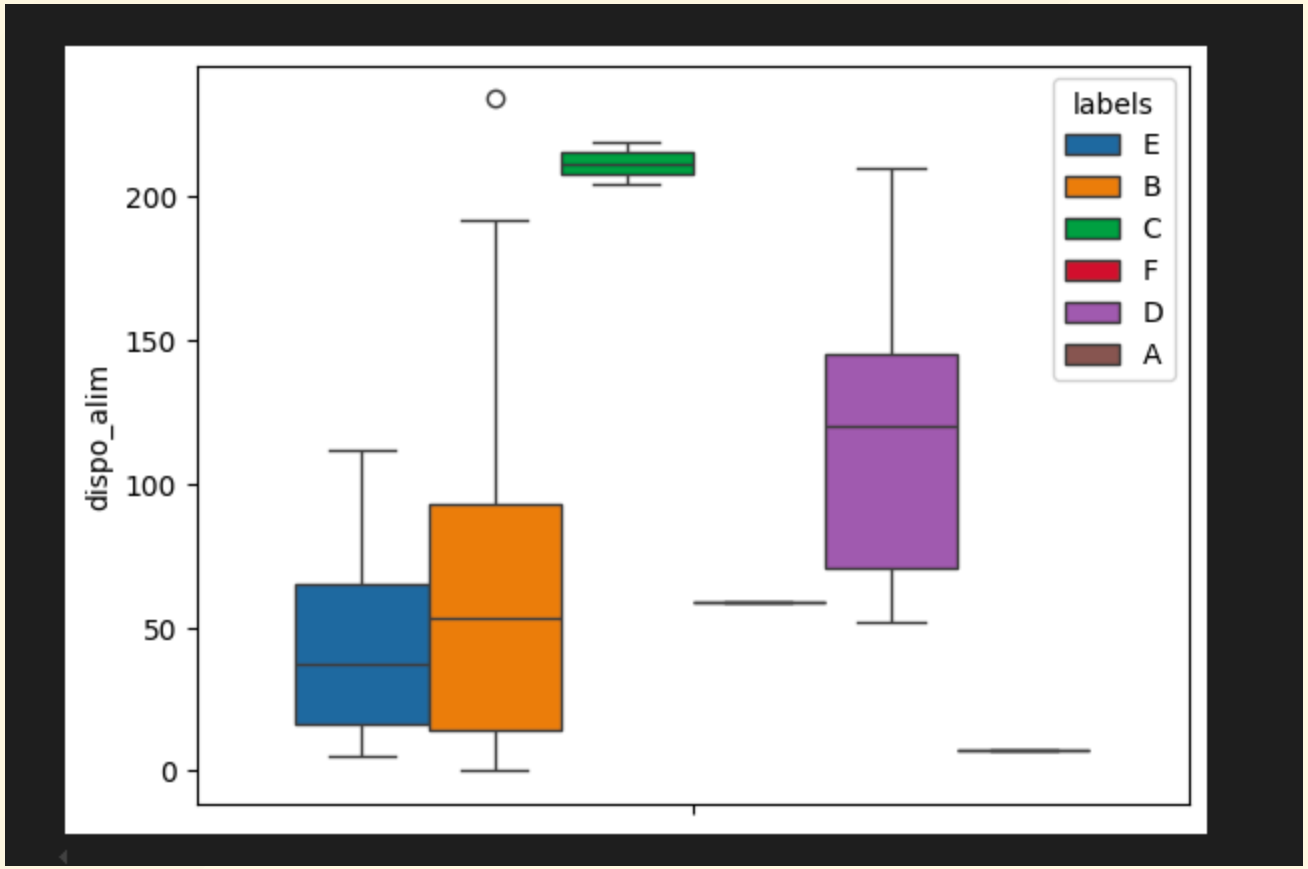
[58] ✓ 2.9s



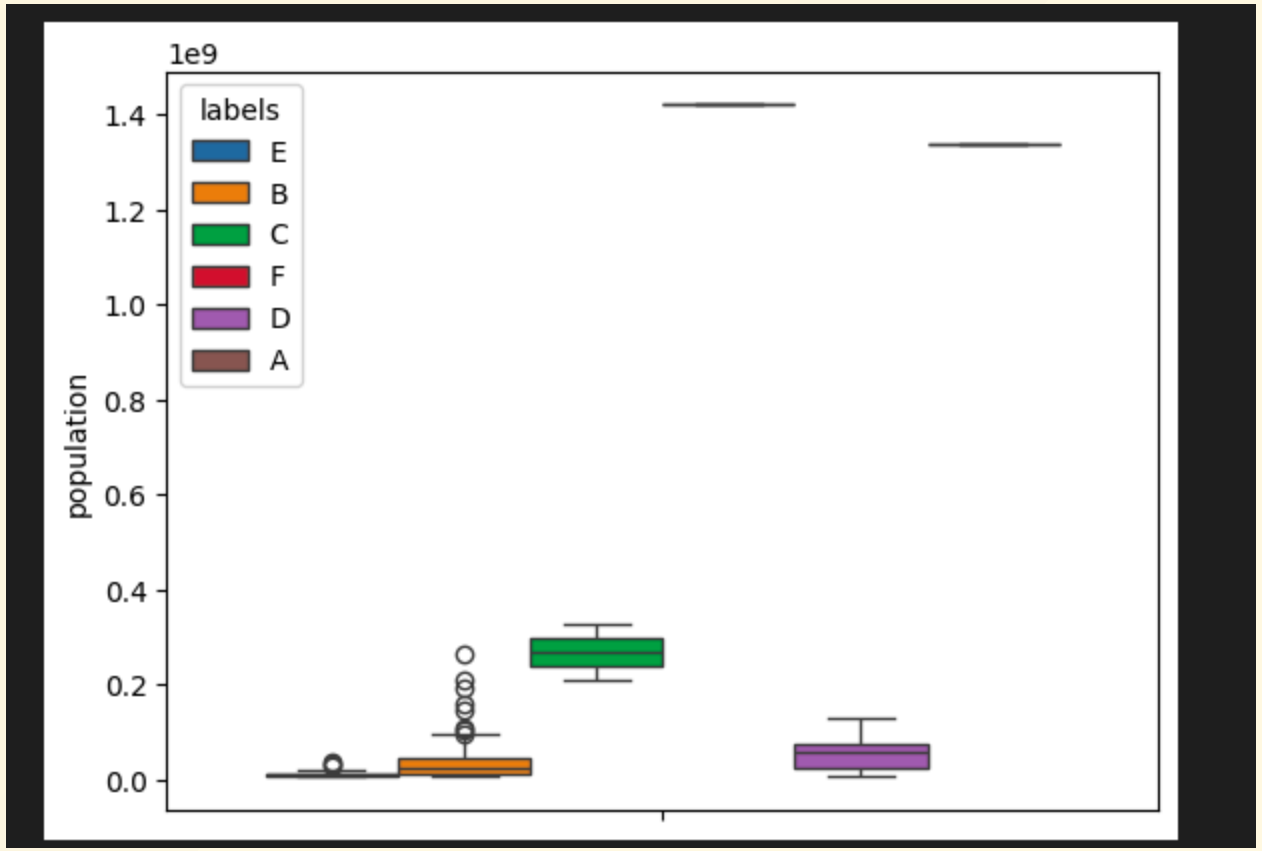
Using Box plot to evaluate the quality of the clusters :



Using Box plot to evaluate the quality of the clusters :



Using Box plot to evaluate the quality of the clusters :



About our groups :

```
[52]  1  _df.loc[_df.labels == "A"]
```

Python

	zone	dispo_int	import	dispo_alim	population	dependence	delta	labels
39	Inde	3661.0	0.0	7.0	1338676000	0.0	-4.0	A

What about B cluster?

```
[53]  1  _df.loc[_df.labels == "B"]
```

Python

	zone	dispo_int	import	dispo_alim	population	dependence	delta	labels
1	Algérie	277.0	2.0	22.0	41389000	0.007220	2.0	B
3	Argentine	1962.0	8.0	182.0	43937000	0.004077	-199.0	B
4	Australie	1171.0	16.0	192.0	24584000	0.013664	-26.0	B
6	Bangladesh	250.0	0.0	7.0	159685000	0.000000	0.0	B
7	Bolivie (État plurinational de)	429.0	1.0	155.0	11192000	0.002331	0.0	B
...	...	...	...	...	...	...	...	...
103	Éthiopie	14.0	1.0	0.0	106399000	0.071429	1.0	B
104	Yémen	246.0	78.0	30.0	27834000	0.317073	78.0	B

About our groups :

```
[54]  1  _df.loc[_df.labels == "C"]
```

Python

	zone	dispo_int	import	dispo_alim	population	dependence	delta	labels
8	Brésil	9982.0	3.0	204.0	207833000	0.000301	-4220.0	C
99	États-Unis d'Amérique	18266.0	123.0	219.0	325084000	0.006734	-3569.0	C

```
[55]  1  _df.loc[_df.labels == "D"]
```

Python

	zone	dispo_int	import	dispo_alim	population	dependence	delta	labels
29	France	1573.0	506.0	92.0	64842000	0.321678	5.0	D
30	Allemagne	1739.0	842.0	71.0	82658000	0.484186	196.0	D
37	Chine - RAS de Hong-Kong	280.0	907.0	210.0	7306000	3.239286	244.0	D
42	Iraq	566.0	470.0	52.0	37552000	0.830389	470.0	D
47	Japon	2415.0	1069.0	67.0	127502000	0.442650	1059.0	D
59	Mexique	4219.0	972.0	123.0	124777000	0.230386	963.0	D
63	Pays-Bas	372.0	608.0	70.0	17021000	1.634409	-810.0	D
79	Arabie saoudite	1435.0	722.0	151.0	33101000	0.503136	712.0	D
83	Afrique du Sud	2118.0	514.0	143.0	57009000	0.242682	451.0	D
95	Émirats arabes unis	412.0	433.0	147.0	9487000	1.050971	339.0	D

About our groups :

Generate + Code + Markdown

```
[57]  1  _df.loc[_df.labels == "F"]
```

Python

	zone	dispo_int	import	dispo_alim	population	dependence	delta	labels
16	Chine, continentale	18161.0	452.0	59.0	1421021000	0.024888	-124.0	F

```
[56]  1  _df.loc[_df.labels == "E"]
```

Python

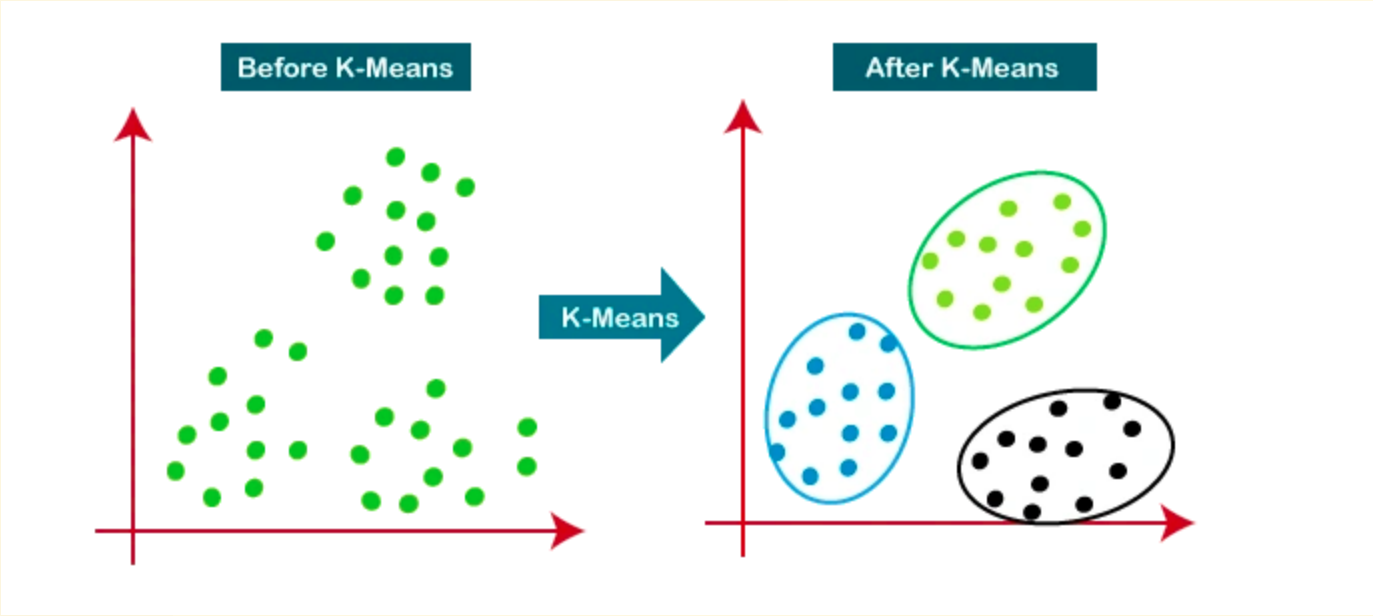
	zone	dispo_int	import	dispo_alim	population	dependence	delta	labels
0	Afghanistan	57.0	29.0	5.0	36296000	0.508772	29.0	E
2	Angola	319.0	277.0	35.0	29816000	0.868339	277.0	E
5	Autriche	173.0	110.0	65.0	8819000	0.635838	32.0	E
9	Bulgarie	157.0	108.0	81.0	7102000	0.687898	63.0	E
18	Congo	110.0	104.0	72.0	5110000	0.945455	104.0	E
19	Cuba	342.0	312.0	82.0	11339000	0.912281	312.0	E
21	Bénin	161.0	123.0	37.0	11175000	0.763975	123.0	E
22	Danemark	167.0	133.0	112.0	5732000	0.796407	-6.0	E
31	Ghana	211.0	151.0	16.0	29121000	0.715640	151.0	E

Conclusion

Feature	Cluster A	Cluster C	Cluster x
dispo int	=		?
import	--	- (-)	?
dispo alim	--	+	?
pop	++	=	?
dependance	--	--	?
delta	++	++	?
dispo alim (modified)		+ (+)	?

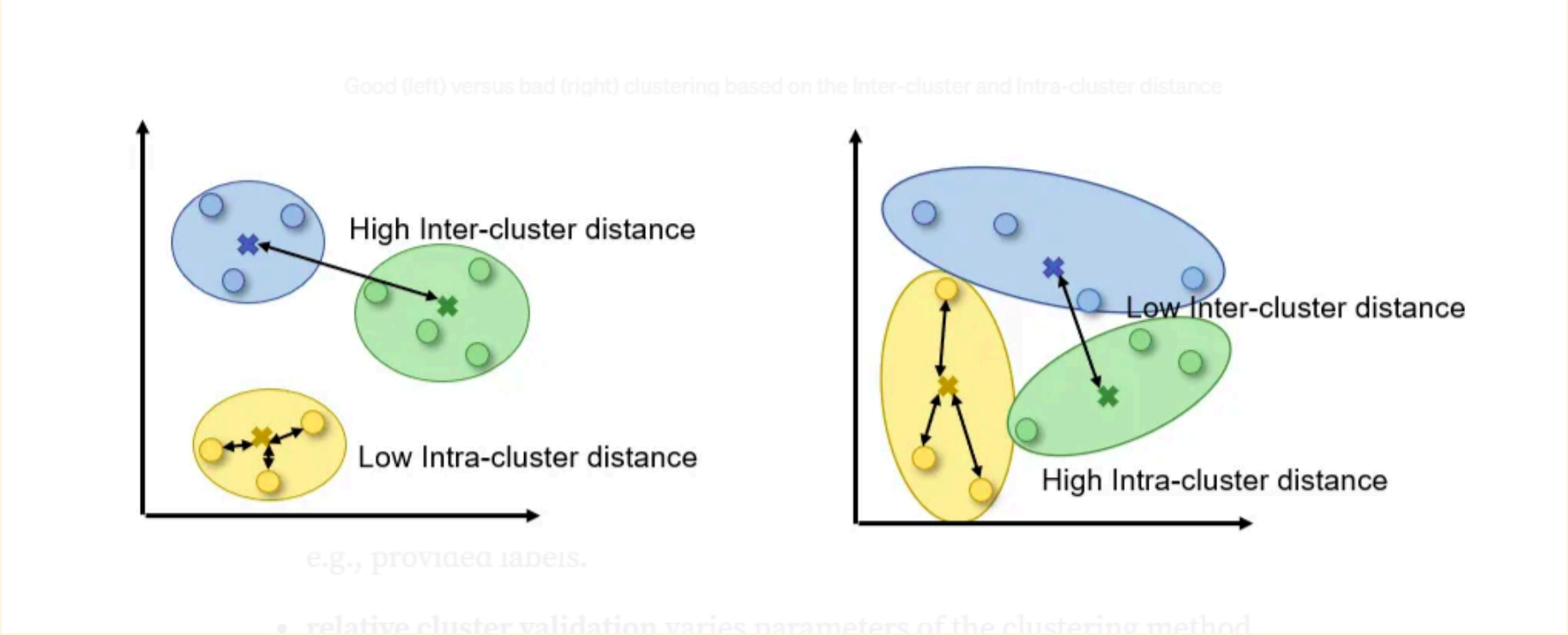
Clustering is about making consistent groups of data points.

Simple example in a 2d space :



How to define a good clustering ?

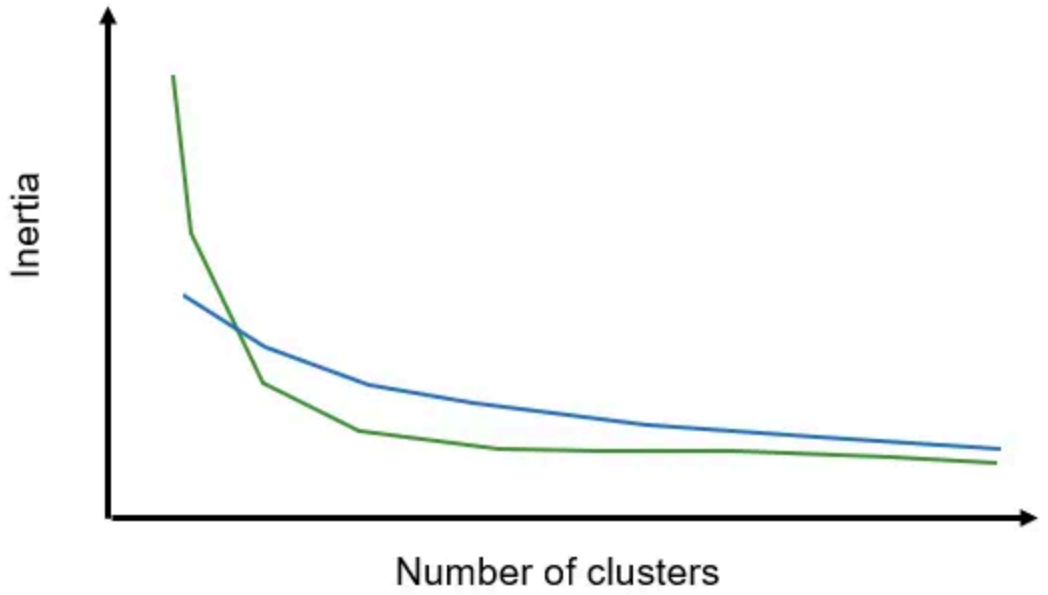
- Maximising extra-cluster variance
- Minimising intra-cluster variance



How defining a good number of clusters ?

About The elbow method :

- Finding the elbow in the curve of the intra-cluster variance



Example of an Elbow curve which is relatively easy to interpret (green line) and one that is hard to interpret (blue line).

Computing Within-cluster variance for different number of clusters :

Compute the inertia for different number of clusters:

Generate + Code + Markdown

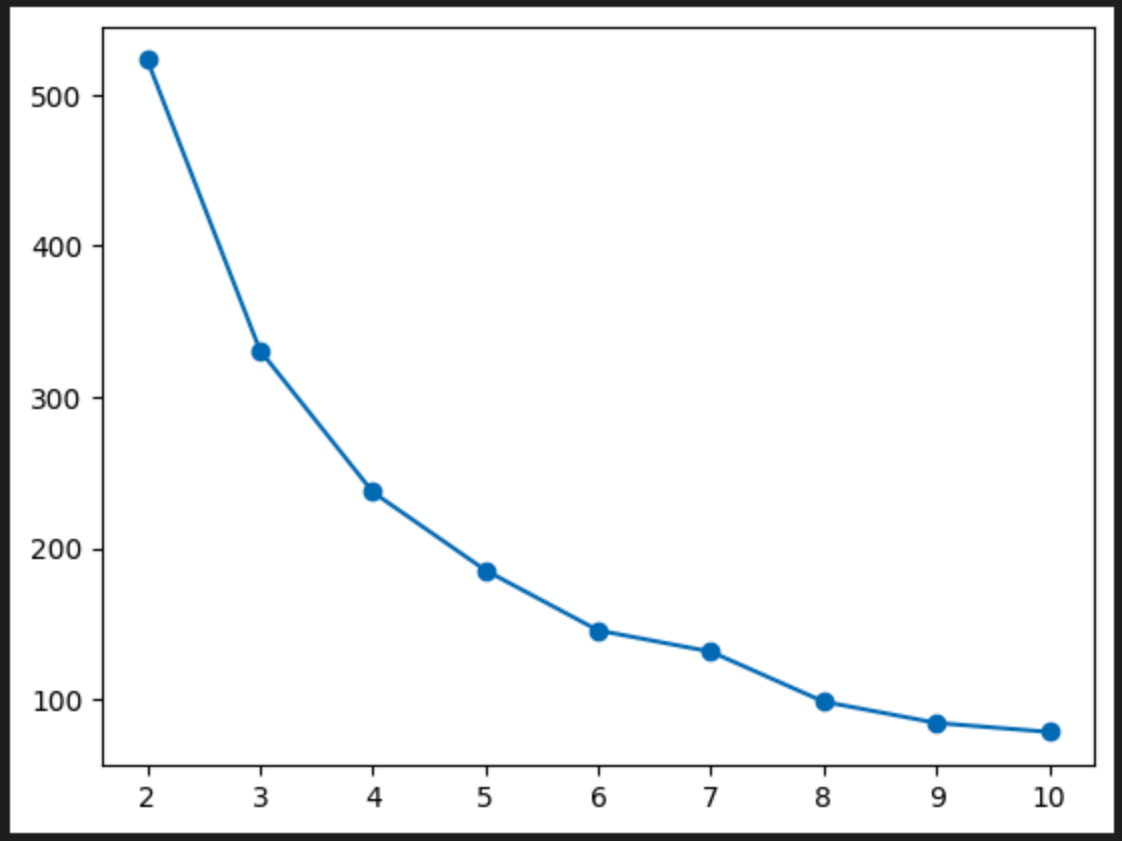
```python
# WCSS : Within-Cluster-Sum-of-Squares

inertia_list = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(X_scaled)
    print(k, kmeans.inertia_)
    inertia_list.append(kmeans.inertia_)
```

✓ 0.1s Python

```
2 522.8838177827712
3 330.79066667109197
4 237.146988322484
5 185.37249127267006
6 145.60692318021646
7 131.52790044646872
8 98.48988352252336
9 84.41440666399866
10 78.44564475545795
```

Displaying the elbow :

```
[62]  1  plt.plot(range(2, 11), inertia_list, marker="o")
      ✓  0.2s
...   [<matplotlib.lines.Line2D at 0x757b80779210>]
```



4 Python and Data Analysis

4.1 Why Python for Data Analysis?

Python is the best candidate :

- Easy to learn
- Huge community
- Many libraries
- Many tools
- Used in many fields IRL

4.2 Using Python for Data Analysis

- Online:
  - **Use Google Colab https://colab.research.google.com/** (you have to be connected to your google account)
  - Use Jupyter online https://jupyter.org/try-jupyter (**Warning** ⚠️: External packages cannot be installed)

4.3 Material

All the material for this course could be found here.

- https://github.com/AlexandreGazagnes/Unilassalle-Public-Ressources/tree/main/4a-data-analysis

External ressources :

- On Youtube :
  - https://www.youtube.com/watch?v=8KeJZBZGtYo
  - https://www.youtube.com/watch?v=JJYZ3OE_lGo
  - https://www.youtube.com/watch?v=tCVXoTV12dE

- On Youtube :
  - https://www.youtube.com/watch?v=ovlID7gefzE
  - https://www.youtube.com/watch?v=IMrxB8Mq5KU
  - https://www.youtube.com/watch?v=Ou-7G9VQugg
  - https://www.youtube.com/watch?v=5pf0_bpNbkw

4.4 Context of the pratical work

You are a new employee of the NPO named "NPO".

You are in charged of data analysis.

First project is about GHG emissions, more precisely regarding Bovine Meat.

4.5 Data

After a quick look on the internet, you find a very interesting dataset on the FAO website. It contains a list of various indicators. You decide to use this dataset to identify segments of countries.

- Find relevant data :
  - https://www.kaggle.com/datasets/unitednations/global-food-agriculture-statistics
  - https://www.kaggle.com/datasets/dorbicycle/world-foodfeed-production
  - https://www.fao.org/faostat/en/
  - https://fr-en.openfoodfacts.org/

You can use a preprocessed version of the dataset here. (Best option)

4.6 Mission

Our job is to :

- Prepare notebook environment
- Load data / Explore data
- Clean data ==> Select relevant data
- Clean data ==> Handle missing values
- Clean data ==> Handle duplicates ? / Handle outliers ?
- Perform some basic analysis and data inspection
- Perform some basic visualisation
- Export our data

4.7 Time To practice !

- Let's Go !