Data Visualisation using Python

Lecture 1: Understanding Data, Why We Visualise, and Getting Python Ready

Dr. D Bhanu Prakash

Assistant Professor,
Department of Mathematics and Computer Science. SSSIHL, India.
Contact: dbhanuprakash233@gmail.com.
dbhanuprakash233.github.io.

Three Words for This Course

Data $\longrightarrow$ **Visualisation** $\longrightarrow$ **Python**

- **Data** – what we're working with, and why its *type* matters.
- **Visualisation** – why staring at numbers alone can deceive us.
- **Python** – the tool we will use, all semester, to turn data into pictures.

Today's lecture walks through all three, in that order.

What Is Data?

Data is any collected fact, measurement, or observation that can be analysed to produce information.

What Is Data?

Data is any collected fact, measurement, or observation that can be analysed to produce information.

Key idea Raw data alone doesn't tell a story – analysis (and, as we'll see today, *visualisation*) is what turns it into insight.

What Is Data?

Data is any collected fact, measurement, or observation that can be analysed to produce information.

Key idea Raw data alone doesn't tell a story – analysis (and, as we'll see today, *visualisation*) is what turns it into insight. We can classify data along

four lenses, which we'll go through one at a time:

1. Nature – qualitative or quantitative?
2. Measurement scale – nominal, ordinal, interval, ratio?
3. Structure – discrete/continuous, structured/unstructured?
4. The kind of *question* it can help answer.

Data by Nature

Qualitative (Categorical)

Descriptive, non-numeric. *e.g. colour, gender, brand*

Quantitative (Numerical)

Measurable numbers. *e.g. height, income, age*

Data by Nature

Qualitative (Categorical)

Descriptive, non-numeric. *e.g. colour, gender, brand*

Quantitative (Numerical)

Measurable numbers. *e.g. height, income, age*

This is the broadest split – everything else today is a refinement of one of these two buckets.

Data by Measurement Scale

Scale	**Property**	**Example**
Nominal	Categories, no order	nationality, eye colour
Ordinal	Ordered categories, unequal gaps	low / medium / high
Interval	Numeric, equal gaps, *no* true zero	temperature in °C
Ratio	Numeric, equal gaps, *true* zero exists	weight, income

Data by Measurement Scale

Scale	Property	Example
Nominal	Categories, no order	nationality, eye colour
Ordinal	Ordered categories, unequal gaps	low / medium / high
Interval	Numeric, equal gaps, *no* true zero	temperature in °C
Ratio	Numeric, equal gaps, *true* zero exists	weight, income

Why this matters You cannot calculate a meaningful *average* for nominal data (eye colour) – but you can for ratio data (income). The scale decides which statistics are even valid.

Quantitative Sub-types & Data Structure

Discrete vs. Continuous

Discrete: countable whole numbers *(number of students)* **Continuous:** any value in a range *(height, time)*

Structured vs. Unstructured

Structured: fits rows/columns *(spreadsheets, databases)*

Unstructured: no fixed format *(text, images, video)*

Quantitative Sub-types & Data Structure

Discrete vs. Continuous

Discrete: countable whole numbers *(number of students)* **Continuous:** any value in a range *(height, time)*

Structured vs. Unstructured

Structured: fits rows/columns *(spreadsheets, databases)*

Unstructured: no fixed format *(text, images, video)*

Both splits matter for Python: they decide which data type (int, float, string...) and which library you reach for.

Types of Questions Data Can Answer

- **Descriptive** – What happened? *(summarising past data)*

Types of Questions Data Can Answer

- **Descriptive** – What happened? *(summarising past data)*
- **Exploratory** – What patterns or relationships exist?

Types of Questions Data Can Answer

- **Descriptive** – What happened? *(summarising past data)*
- **Exploratory** – What patterns or relationships exist?
- **Inferential** – What can we conclude about a population from a sample?

Types of Questions Data Can Answer

- **Descriptive** – What happened? *(summarising past data)*
- **Exploratory** – What patterns or relationships exist?
- **Inferential** – What can we conclude about a population from a sample?
- **Predictive** – What is likely to happen next?

Types of Questions Data Can Answer

- **Descriptive** – What happened? *(summarising past data)*
- **Exploratory** – What patterns or relationships exist?
- **Inferential** – What can we conclude about a population from a sample?
- **Predictive** – What is likely to happen next?
- **Causal** – Did X actually cause Y?

Types of Questions Data Can Answer

- **Descriptive** – What happened? *(summarising past data)*
- **Exploratory** – What patterns or relationships exist?
- **Inferential** – What can we conclude about a population from a sample?
- **Predictive** – What is likely to happen next?
- **Causal** – Did X actually cause Y?
- **Mechanistic** – How does X cause Y?

Types of Questions Data Can Answer

- **Descriptive** – What happened? *(summarising past data)*
- **Exploratory** – What patterns or relationships exist?
- **Inferential** – What can we conclude about a population from a sample?
- **Predictive** – What is likely to happen next?
- **Causal** – Did X actually cause Y?
- **Mechanistic** – How does X cause Y?

Visualisation supports every one of these – but it only answers the question correctly if the underlying data type was respected.

Why It Matters

The type of data determines which statistical tests, visualisations, and analysis methods are *valid*.

Example You cannot calculate a meaningful average for nominal data (eye colour), but you can for ratio data (income).

Why It Matters

The type of data determines which statistical tests, visualisations, and analysis methods are *valid*.

Example You cannot calculate a meaningful average for nominal data (eye colour), but you can for ratio data (income). **Bridge:** Get the data type

wrong, and no chart or test built on top of it can be trusted. That's exactly where *visualisation* comes in – our next stop.

A Human Fact, and a Course Roadmap

A human fact: our brains process images $\sim$60,000$\times$ faster than raw text/numbers.

A Human Fact, and a Course Roadmap

A human fact: our brains process images $\sim$60,000$\times$ faster than raw text/numbers.

Course roadmap (this semester):

- Python environments & tooling (today)
- Matplotlib fundamentals
- Seaborn & statistical plots
- Interactive viz: Plotly, Bokeh
- Dashboards, storytelling with data

A Human Fact, and a Course Roadmap

A human fact: our brains process images $\sim$60,000$\times$ faster than raw text/numbers.

Course roadmap (this semester):

- Python environments & tooling (today)
- Matplotlib fundamentals
- Seaborn & statistical plots
- Interactive viz: Plotly, Bokeh
- Dashboards, storytelling with data

Puzzle to think about

Two datasets have the *exact same* mean, variance, correlation, and regression line. Are they the same data? *(Answer next slide.)*

The Case for Visualisation: Anscombe's Quartet (1973)

Four datasets, each with: mean(x)=9.0, mean(y)=7.5, same variance, correlation $r = 0.816$, *identical* regression line $y = 3.00 + 0.500x$.

Statistically "identical" by every summary number.

But... When plotted, they look completely different: a linear trend, a curve, a perfect line with one outlier, and a vertical cluster with one outlier.

Lesson: Summary statistics can hide structure, outliers, and non-linearity. *Always plot your data.* This is the founding argument for the entire field of data visualisation.

https://github.com/dbhanuprakash233/SSSIHL_DBP/blob/main/codes/AnscombeQuartet.ipynb

The Dinosaur Problem: Same Stats, Wildly Different Graphs

Anscombe's Quartet has a modern, more dramatic cousin: the **Datasaurus Dozen** (Matejka & Fitzmaurice, 2017, building on Alberto Cairo's original "Datasaurus").

The set-up

- 13 datasets, 142 points each
- All share, to 2 decimal places: mean(x)=54.26, mean(y)=47.83, std(x)=16.77, std(y)=26.93, $r = -0.06$

The Dinosaur Problem: Same Stats, Wildly Different Graphs

Anscombe's Quartet has a modern, more dramatic cousin: the **Datasaurus Dozen** (Matejka & Fitzmaurice, 2017, building on Alberto Cairo's original "Datasaurus").

The set-up

- 13 datasets, 142 points each
- All share, to 2 decimal places: mean(x)=54.26, mean(y)=47.83, std(x)=16.77, std(y)=26.93, $r = -0.06$

Yet... The shapes range from a **dinosaur** to a circle, an X, a star, a bullseye, and parallel lines – all with those *same* four numbers.

How Is This Even Possible? (The Main Idea)

No trick, no rounding error – it's constructed on purpose.

The core idea (in words, not code)

1. Start from a real dataset (or a target shape, like a dinosaur outline).
2. Nudge a random point by a tiny random amount.
3. Recompute mean, std, and correlation. **Keep** the nudge only if these stay within a razor-thin tolerance of the original values, *and* the point moved a little closer to the target shape.
4. Repeat this thousands of times.

How Is This Even Possible? (The Main Idea)

No trick, no rounding error – it's constructed on purpose.

The core idea (in words, not code)

1. Start from a real dataset (or a target shape, like a dinosaur outline).
2. Nudge a random point by a tiny random amount.
3. Recompute mean, std, and correlation. **Keep** the nudge only if these stay within a razor-thin tolerance of the original values, *and* the point moved a little closer to the target shape.
4. Repeat this thousands of times.

This is essentially a **simulated annealing / directed random walk**: freedom to move, but a strict statistical leash.

The Lesson – and the Punchline

Recall from earlier: x and y here are **continuous, ratio-scale** variables, so computing a mean, std, or correlation is technically *valid*.

The Lesson – and the Punchline

Recall from earlier: x and y here are **continuous, ratio-scale** variables, so computing a mean, std, or correlation is technically *valid*.

The punchline Valid statistics $\neq$ the correct picture. A perfectly legitimate summary can still hide a dinosaur.

The Lesson – and the Punchline

Recall from earlier: x and y here are **continuous, ratio-scale** variables, so computing a mean, std, or correlation is technically *valid*.

The punchline Valid statistics $\neq$ the correct picture. A perfectly legitimate summary can still hide a dinosaur.

Take-away: Visualisation isn't decoration – it's a check on your analysis, every single time.

Dataset bundled as `datasaurusdozen.csv` (columns: `dataset, x, y`). Full code in the linked notebook – we'll run it together in lab.
https://github.com/dbhanuprakash233/SSSIHL_DBP/blob/main/codes/DatasaurusDozen.ipynb

Puzzle 1 – The Truncated Y-Axis

The underlying *data* isn't lying – the numbers are accurate. The *chart* is lying, because of how humans perceive bar charts: we read bar height as proportional to value. If the axis starts at 80 instead of 0, bars going 80→90 and 80→95 *look* like a 2× difference, though the real values differ by only ∼5.5%. Tufte's "Lie Factor"

$$\text{Lie Factor} = \frac{\text{size of effect shown in graphic}}{\text{size of effect in data}}$$

A truncated bar axis routinely produces a lie factor of 2–10× – a common (often deliberate) trick in political ads and financial reports. **Rule of thumb:**

bar charts must start at zero (they encode value as *area/length*); line charts, read by *slope*, are usually fine without a zero baseline.

Puzzle 2 – Seven Similar-Sized Pie Slices

Pie charts ask the eye to compare **angles**, and humans are bad at judging angle precisely. Cleveland & McGill (1984) ranked perceptual accuracy of visual encodings: angle/area sits near the *bottom*; position along a common scale sits at the *top*.

With 7 similar-sized slices, you simply can't tell if slice 3 is bigger than slice 5 by eye.

Better alternatives

- **Sorted horizontal bar chart** – position/length is easy to compare, and sorting makes ranking immediate.
- **Dot plot** – even more precise for many categories.

General rule: pie charts work only for 2–3 clearly different slices (e.g. 70/20/10). Beyond that, switch to bars.

Ways to Get Python on Your Machine

There is no single “correct” way to install Python – only trade-offs.

Ways to Get Python on Your Machine

There is no single "correct" way to install Python – only trade-offs.

Four broad routes we will cover

1. **Official CPython** (python.org) + `pip`
2. **Anaconda / Miniconda** (scientific distribution + `conda`)
3. **IDEs / Editors that bundle or manage Python** (PyCharm, VS Code, Thonny)
4. **Cloud / zero-install notebooks** (Google Colab, Jupyter, Replit)

Ways to Get Python on Your Machine

There is no single "correct" way to install Python – only trade-offs.

Four broad routes we will cover

1. **Official CPython** (python.org) + `pip`
2. **Anaconda / Miniconda** (scientific distribution + `conda`)
3. **IDEs / Editors that bundle or manage Python** (PyCharm, VS Code, Thonny)
4. **Cloud / zero-install notebooks** (Google Colab, Jupyter, Replit)

Counter-intuitive fact Installing "Python" does not mean you installed *one* thing. On a typical CS lab machine there may be 3–5 different Python interpreters (`python2`, `python3`, a conda env, a venv, a system package) all coexisting – and `which python` may lie to you!

python.org+pip vs. Anaconda/Miniconda

python.org + pip – the "bare metal" way.

```
$ pip install matplotlib numpy
    pandas
$ python3 -m venv myenv
$ source myenv/bin/activate
```

Pros: lightweight, official.
Cons: you manage compiled/binary dependencies yourself.

Anaconda / Miniconda – "batteries included."

```
$ conda create -n dataviz python
    =3.11
$ conda activate dataviz
```

Pros: handles tricky binary deps (numpy+MKL, GDAL, CUDA).
Cons: Anaconda itself is $\sim$3 GB.

python.org+pip vs. Anaconda/Miniconda

python.org + pip – the "bare metal" way.

```
$ pip install matplotlib numpy
    pandas
$ python3 -m venv myenv
$ source myenv/bin/activate
```

Pros: lightweight, official.
Cons: you manage compiled/binary dependencies yourself.

Anaconda / Miniconda – "batteries included."

```
$ conda create -n dataviz python
    =3.11
$ conda activate dataviz
```

Pros: handles tricky binary deps (numpy+MKL, GDAL, CUDA).
Cons: Anaconda itself is ∼3 GB.

Both give you the same Python in the end – pick based on how much you want managed for you.

IDEs and Editors

An IDE does *not* install Python for you (mostly) – it lets you *use* an interpreter productively.

Tool	**Type**	**Notable feature**
VS Code	Lightweight editor	Extensions, integrated terminal
PyCharm	Full IDE	Refactoring, debugging, venv manager
Jupyter Notebook/Lab	Notebook	Cell-based, inline plots
Spyder	Scientific IDE	Variable explorer (comes with Anacond
Thonny	Beginner IDE	Bundles its own Python, ultra simple

Gotcha Your IDE's "Run" button and your terminal's `python` command can silently point to *different* interpreters – a top source of "but it works on my machine!" bugs.

Which Should *You* Use?

	python.org+pip	**Anaconda**	**Colab**
Disk space	Small	Large (∼3GB)	None (cloud)
Setup time	Fast	Slow	Instant
Offline use	Yes	Yes	No
Pre-installed sci-libs	No	Yes	Yes
Best for	Web/general dev	Data science	Quick experiments

Which Should *You* Use?

	python.org+pip	**Anaconda**	**Colab**
Disk space	Small	Large (∼3GB)	None (cloud)
Setup time	Fast	Slow	Instant
Offline use	Yes	Yes	No
Pre-installed sci-libs	No	Yes	Yes
Best for	Web/general dev	Data science	Quick experiments

This course: We recommend `python3 -m venv` *or* Miniconda + `pip install -r requirements.txt`, so everyone's environment is reproducible.

Which Should *You* Use?

	python.org+pip	**Anaconda**	**Colab**
Disk space	Small	Large (∼3GB)	None (cloud)
Setup time	Fast	Slow	Instant
Offline use	Yes	Yes	No
Pre-installed sci-libs	No	Yes	Yes
Best for	Web/general dev	Data science	Quick experiments

This course: We recommend `python3 -m venv` *or* Miniconda + `pip install -r requirements.txt`, so everyone's environment is reproducible.

Quick Question

If two students both "installed Python 3.11", can `import matplotlib` still fail for only one of them? Why?

The Python Visualisation Ecosystem (Preview)

- **Matplotlib** – the foundational, most-customisable plotting library (2003)
- **Seaborn** – statistical plots built on Matplotlib, nicer defaults
- **Plotly / Bokeh** – interactive, browser-based charts
- **Pandas .plot()** – quick plots straight from a DataFrame
- **Altair** – declarative, "grammar of graphics" style

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4],[3,5,4,8], marker='o')
plt.title("My First Plot"); plt.show()
```

Counter-intuitive idea More colours, more 3D, more chart-junk usually makes a chart *worse*, not better. Edward Tufte calls this wasted ink the "lie factor" and "chartjunk."

Before Next Class: Homework

Homework

1. Install Python via *any one* route from today (venv *or* conda).
2. Run `pip install matplotlib pandas` (or `conda install`).
3. Reproduce one Anscombe's Quartet scatter plot and bring it to class.
4. Load `datasaurusdozen.csv`, plot just *one* dataset, and compare it to the dinosaur.

Questions?

Thank You!

Dr. D Bhanu Prakash
dbhanuprakash233.github.io
Mail: dbhanuprakash233@gmail.com