



Data Visualisation

Scientific Programming with Python

Jonas Eschle

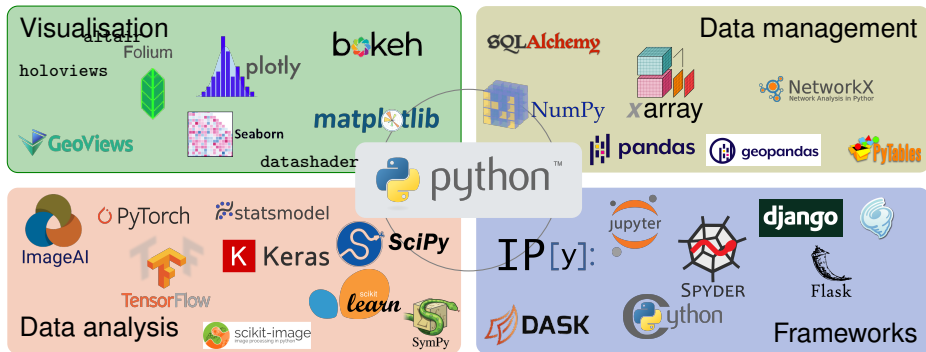


This work is licensed under the *Creative Commons Attribution-ShareAlike 4.0 License*.



Python offers a large ecosystem for scientific analytics and beyond

Domain specific modules





Visualisation is too often treated as an afterthought

80-90% of the information is consumed **visually** by humans

The human brain processes **visuals 1'000 to 10'000× faster than text**

Almost everybody will read this last! (if at all)

**You will read this first
because it is in a colored box and big!**

You will read this probably second!

And then most of the people read this!



Three simple guidelines might help to achieve an optimised way of creating visualisations

based on Jean-luc Doumont “Trees, maps, and theorems”

1. **Adapt to the audience and purpose**

Take into account the audience's pre-knowledge and how the visualisation is consumed.

2. **Maximise the signal-to-noise ratio**

Limit distraction as much as possible on your visualisation.

3. **Use effective redundancy**

Leverage multiple characteristics and channels to convey the insights and data.



Three simple guidelines might help to achieve an optimised way of creating visualisations

based on Jean-luc Doumont “Trees, maps, and theorems”

1. **Adapt to the audience and purpose**

Take into account the audience's pre-knowledge and how the visualisation is consumed.

2. **Maximise the signal-to-noise ratio**

Limit distraction as much as possible on your visualisation.

3. **Use effective redundancy**

Leverage multiple characteristics and channels to convey the insights and data.



Three simple guidelines might help to achieve an optimised way of creating visualisations

based on Jean-luc Doumont “Trees, maps, and theorems”

1. **Adapt to the audience and purpose**

Take into account the audience's pre-knowledge and how the visualisation is consumed.

2. **Maximise the signal-to-noise ratio**

Limit distraction as much as possible on your visualisation.

3. **Use effective redundancy**

Leverage multiple characteristics and channels to convey the insights and data.



Three simple guidelines might help to achieve an optimised way of creating visualisations

based on Jean-luc Doumont “Trees, maps, and theorems”

1. **Adapt to the audience and purpose**

Take into account the audience's pre-knowledge and how the visualisation is consumed.

2. **Maximise the signal-to-noise ratio**

Limit distraction as much as possible on your visualisation.

3. **Use effective redundancy**

Leverage multiple characteristics and channels to convey the insights and data.



Three simple guidelines might help to achieve an optimised way of creating visualisations

based on Jean-luc Doumont “Trees, maps, and theorems”

1. **Adapt to the audience and purpose**

Take into account the audience's pre-knowledge and how the visualisation is consumed.

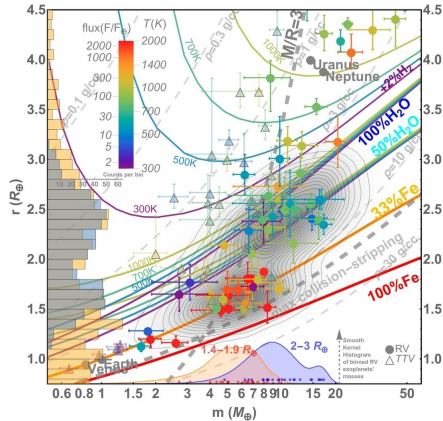
2. **Maximise the signal-to-noise ratio**

Limit distraction as much as possible on your visualisation.

3. **Use effective redundancy**

Leverage multiple characteristics and channels to convey the insights and data.

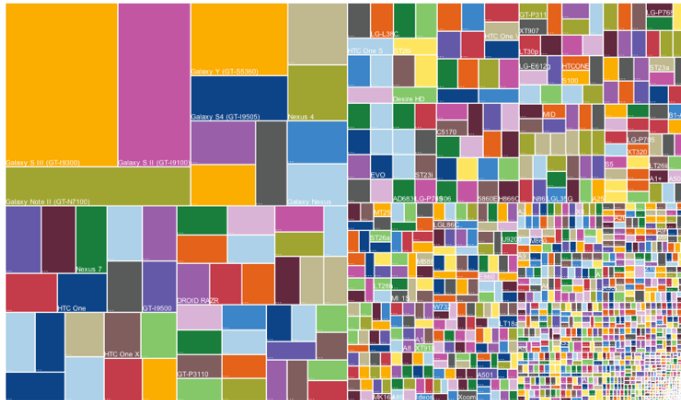
Example 1: Couldn't there be any more information in this chart?



Source: Zeng et al. *Growth model interpretation of planet size distribution*, PNAS **116** (2019), 20



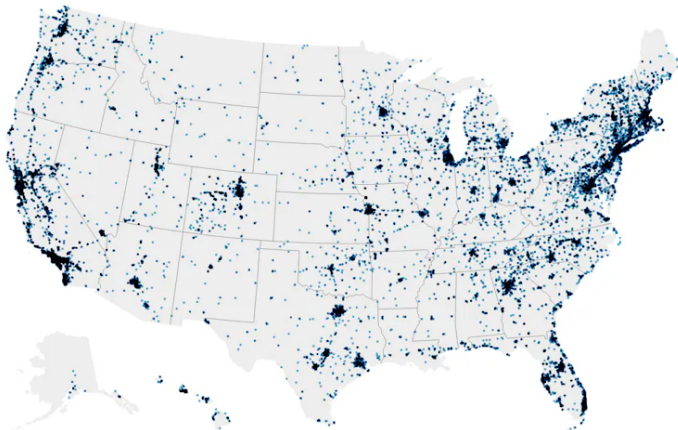
Example 2: What is the purpose?



Source: Open Signal
*Distribution of the Open
Signal App among differ-
ent smartphones*



Example 3: Be mindful of the data representation

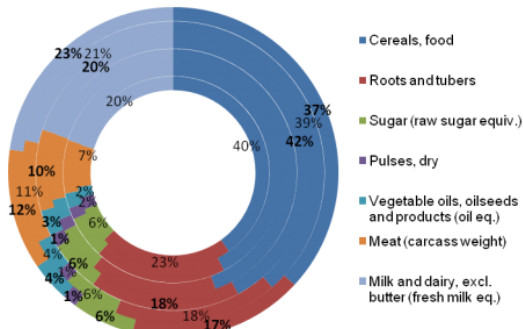


Source: US Department
of Energy

Original caption: [EV]
Charging stations as of
August 2021. Darker
areas are clusters of
many stations, primarily
in large cities.

Example 4: Be clear what message you want to convey

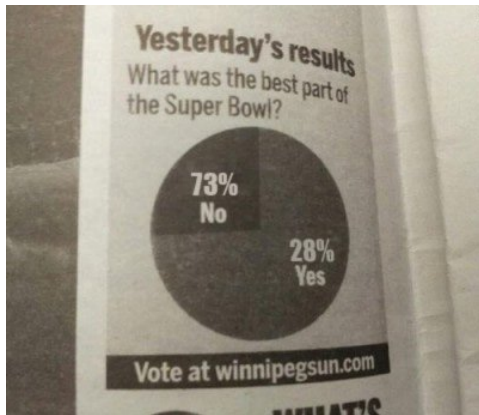
World Dietary shares: (from inside to outside) 1970,
1980, 1990, 2000, 2030, 2050



Source: Northern Ireland Assembly

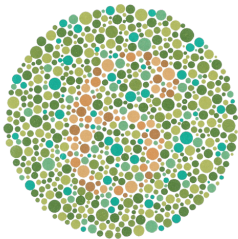
Note: figures for 1980 and 1990 shares are not shown for sake of clarity.

Example 5: Without words...

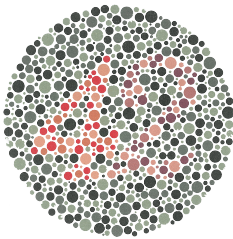


Source: Winnipeg Sun

Dedicated color schemes can be used to make colors better distinguishable for colorblind people and when read in black & white



6



42

- ▶ 8% of Caucasian, 5% of Asian, and 4% of African males are so-called "red-green" Colorblind.
- ▶ Many papers are still printed in black & white.

- ▶ Matplotlib has suitable color gradients like `viridis`, `cividis`, `magma`, `inferno`, `plasma`
- ▶ There are helpful tools like **monolens** to test the suitability of visualisation for such situations.



Different visualisation dimensions allow us to create redundancy and emphasis

Size

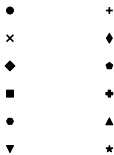


Line style



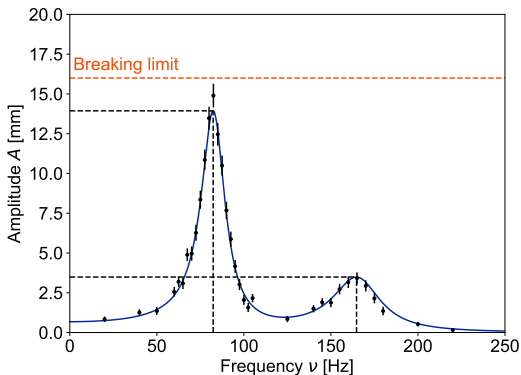
Fill style

Marker shape

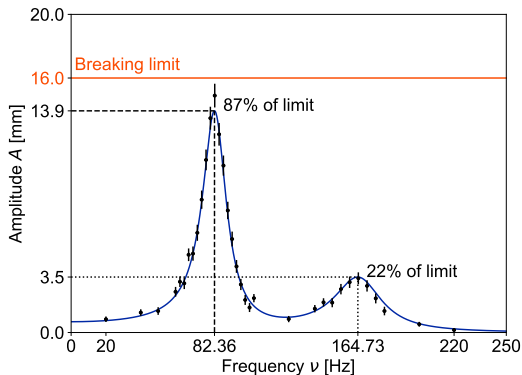


But ensure that you don't overload a visualisation as shown in the examples before!

It does not take much to make better charts – and Python libraries can help us

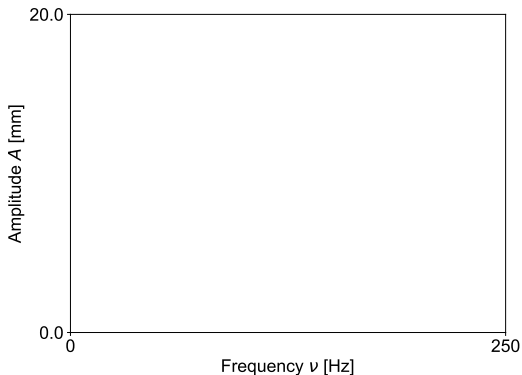


It does not take much to make better charts – and Python libraries can help us

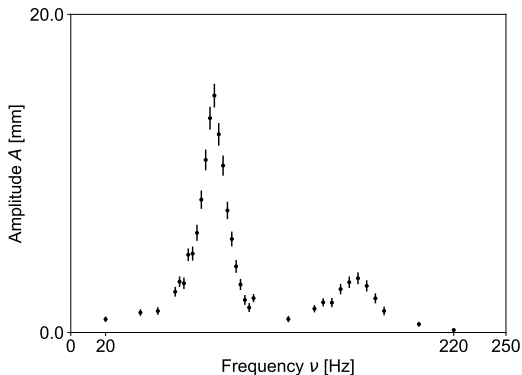




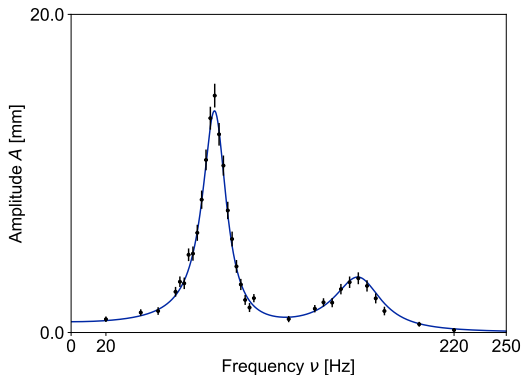
It does not take much to make better charts – and Python libraries can help us



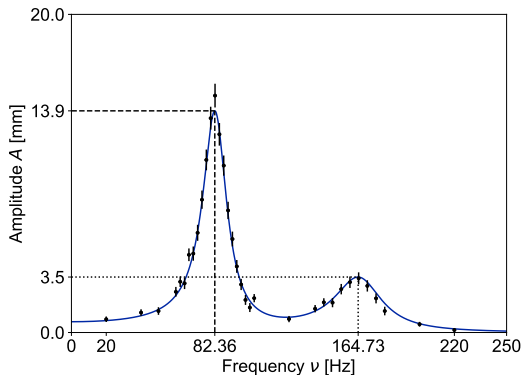
It does not take much to make better charts – and Python libraries can help us



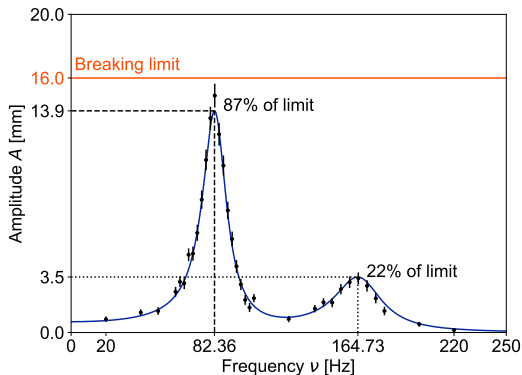
It does not take much to make better charts – and Python libraries can help us



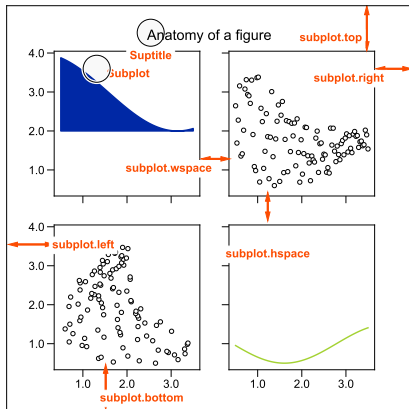
It does not take much to make better charts – and Python libraries can help us



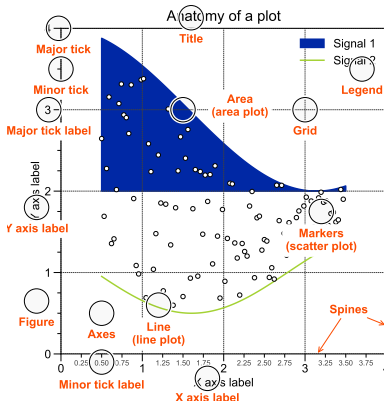
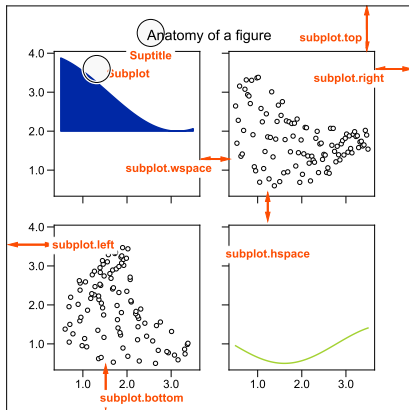
It does not take much to make better charts – and Python libraries can help us



The structure of a figure and plot in matplotlib



The structure of a figure and plot in matplotlib

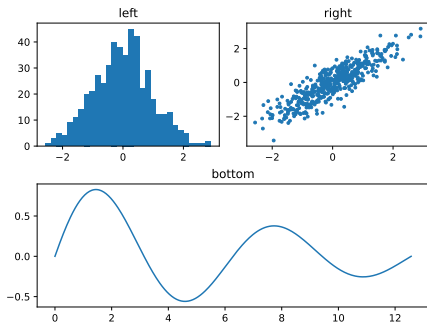




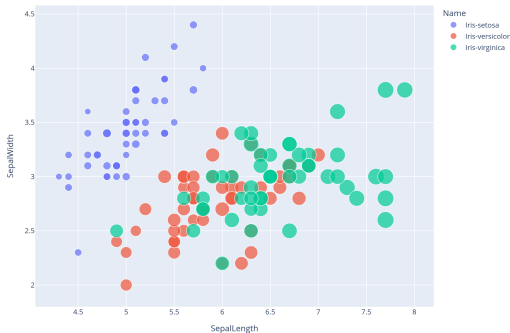
Modern matplotlib: named mosaic layouts and automatic spacing

```
fig = plt.figure(layout="constrained")
axd = fig.subplot_mosaic(
    [
        ["left", "right"],
        ["bottom", "bottom"]
    ])
axd["left"].hist(x, bins=30)
axd["right"].scatter(x, y)
axd["bottom"].plot(t, s)
```

- ▶ `subplot_mosaic`: name your axes instead of counting grid indices
- ▶ `layout="constrained"`: labels, titles and colorbars never overlap
- ▶ colormaps live in a registry:
`matplotlib.colormaps["viridis"]`; register custom maps with `matplotlib.colormaps.register(...)`



Interactive figures complement static plots for exploration



- **plotly express**: a complete interactive figure in one call
`px.scatter(iris, x=..., y=..., color=...)`
- Hover tooltips, zoom & pan come for free – ideal for data exploration
- Export as a self-contained webpage:
`fig.write_html("plot.html")`
- Static export (png/pdf) for papers and slides
- Alternatives: **bokeh**, **altair**, **holoviews**; **datashader** rasterises millions of points



References

- ▶ [matplotlib cheatsheet](#)
- ▶ Claus O. Wilke, [Fundamentals of Data Visualization](#)
- ▶ [Datavizcatalogue](#)
- ▶ [Python Graph Gallery](#)
- ▶ [seaborn objects interface tutorial](#)
- ▶ [plotly express documentation](#)