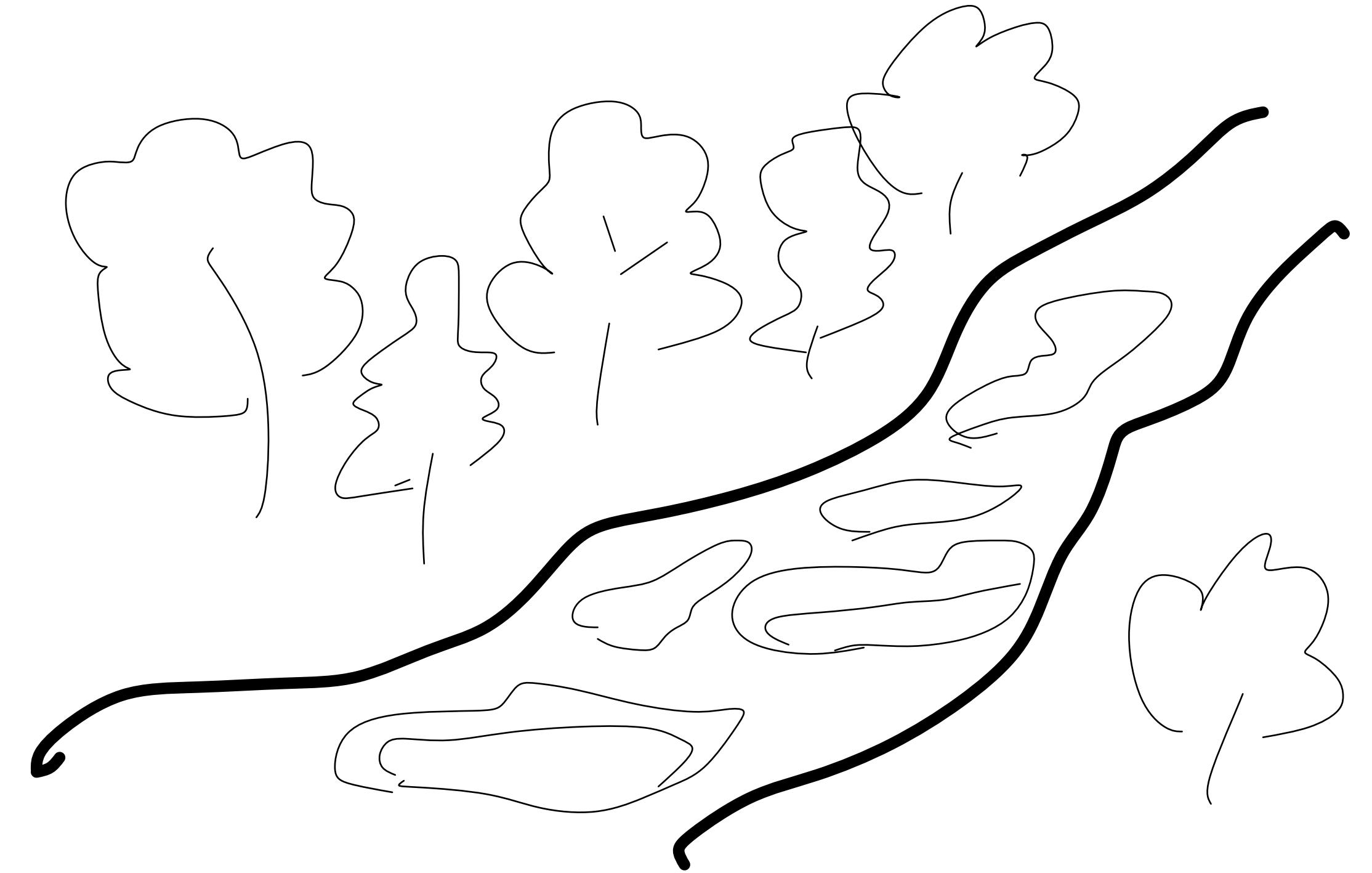
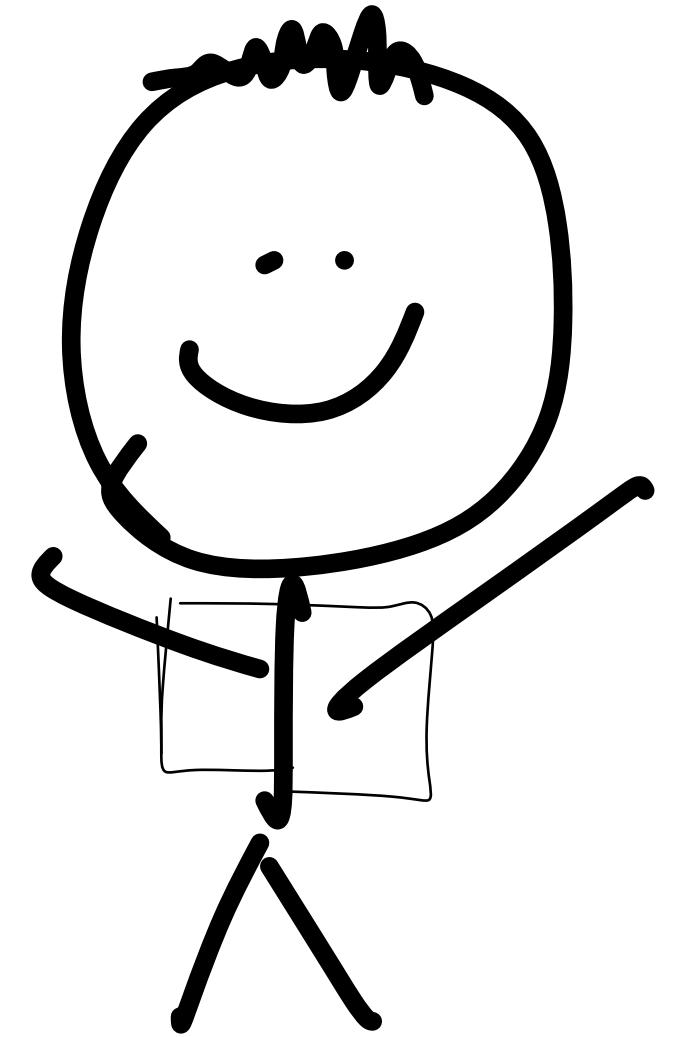
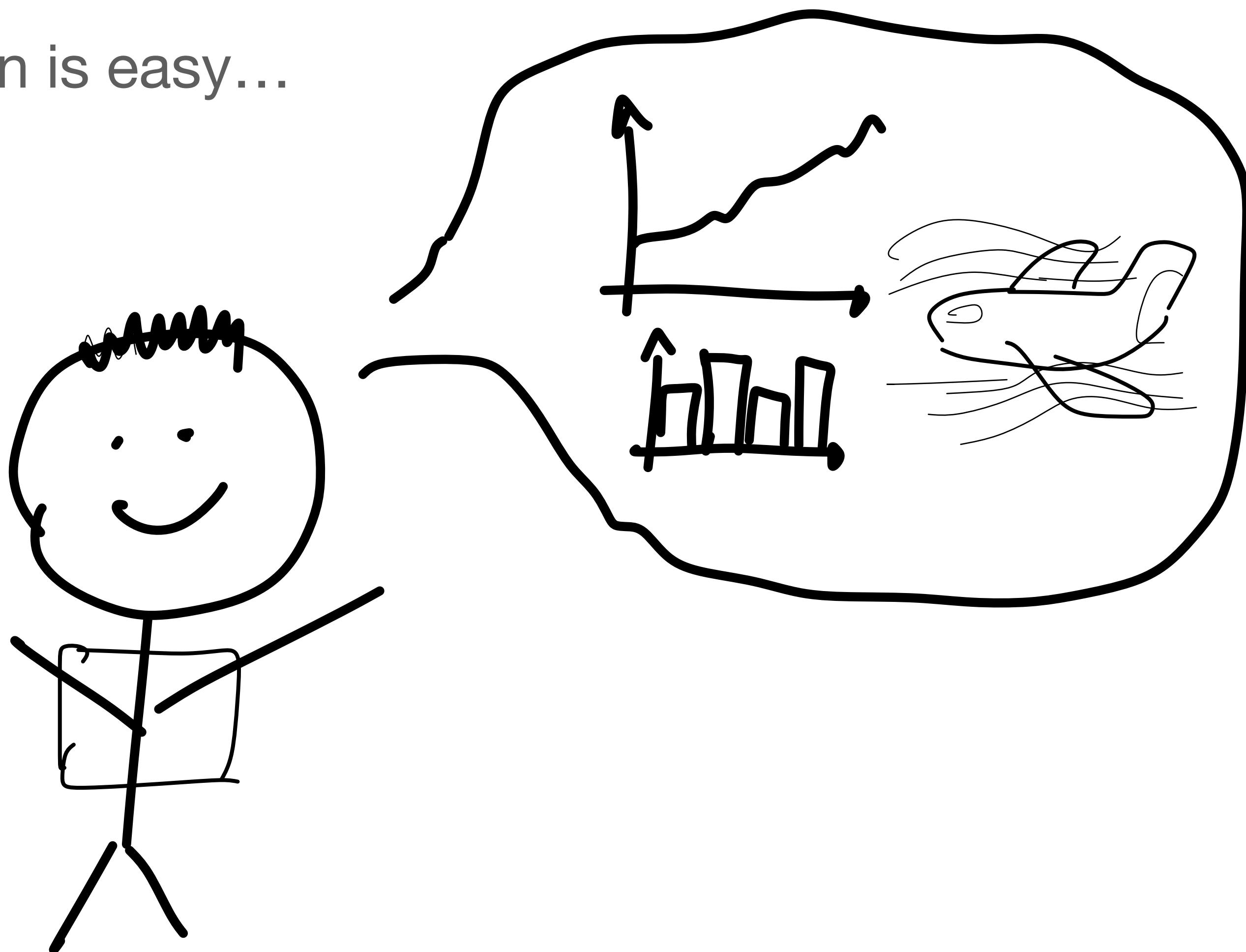


The bumps in the road: A retrospective on my data visualisation mistakes

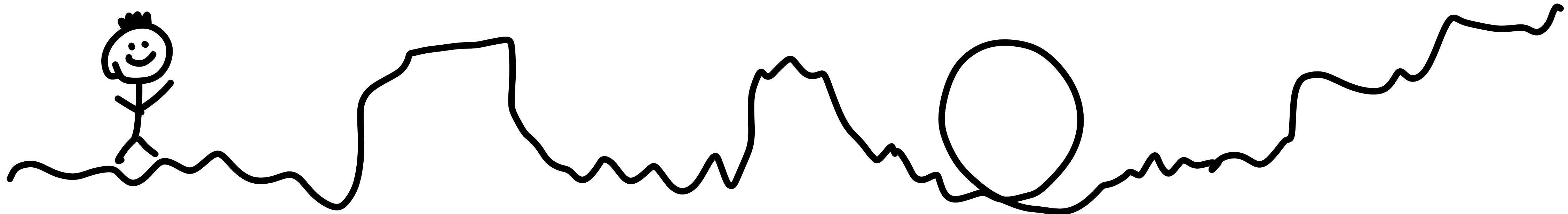
Artem Kislovskiy



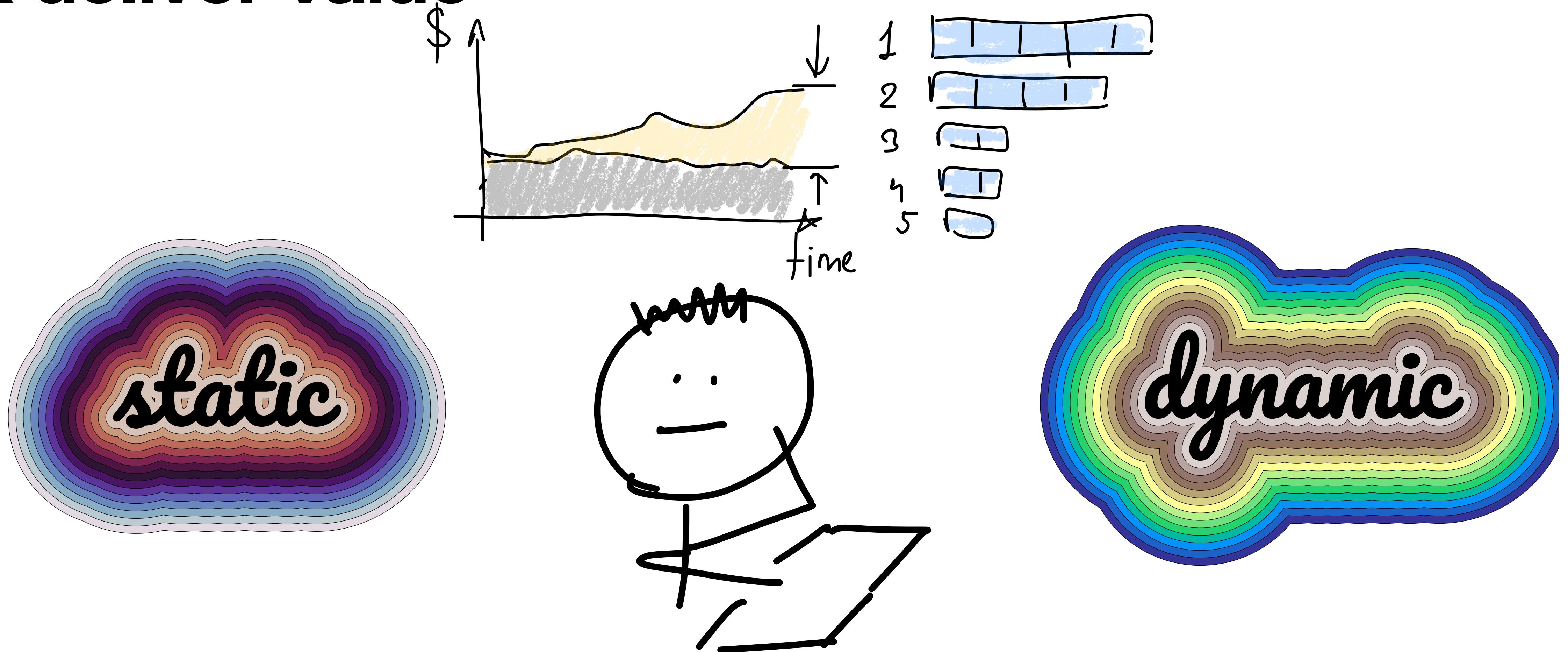
I thought that data visualisation is easy...



- 🤷 I didn't know the rules for better figures
- 🎨 I spent too much time on prettifying the graph
- 🤗 I was lost in versioning the visualisations
- 😅 I was sure that visualisations are unbreakable
- 🤦 I made a mistake and discovered it too late
- 旿 I had to reprocess the data waiting for a couple of days
- 💣 My list of Jupyter notebooks exploded

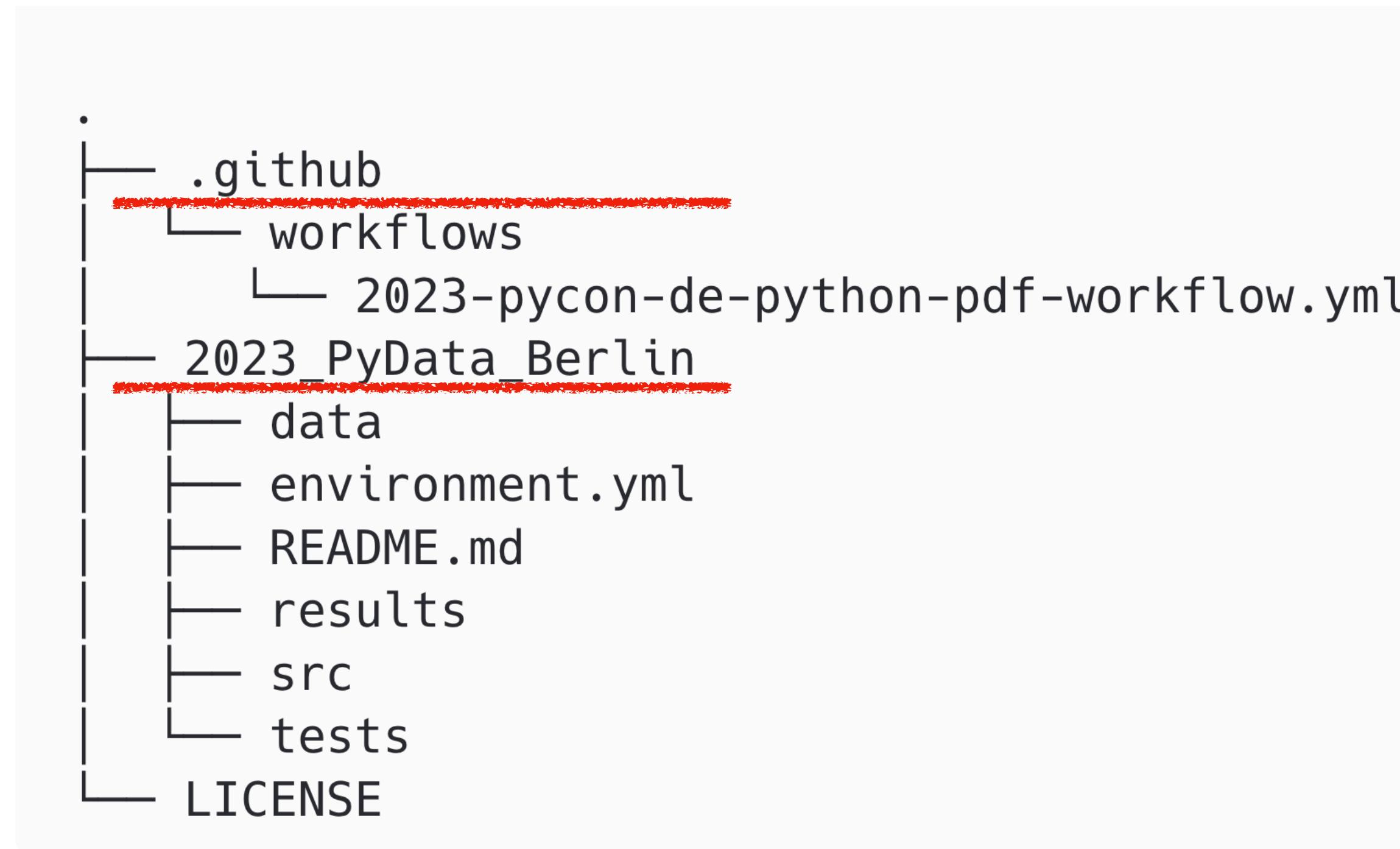
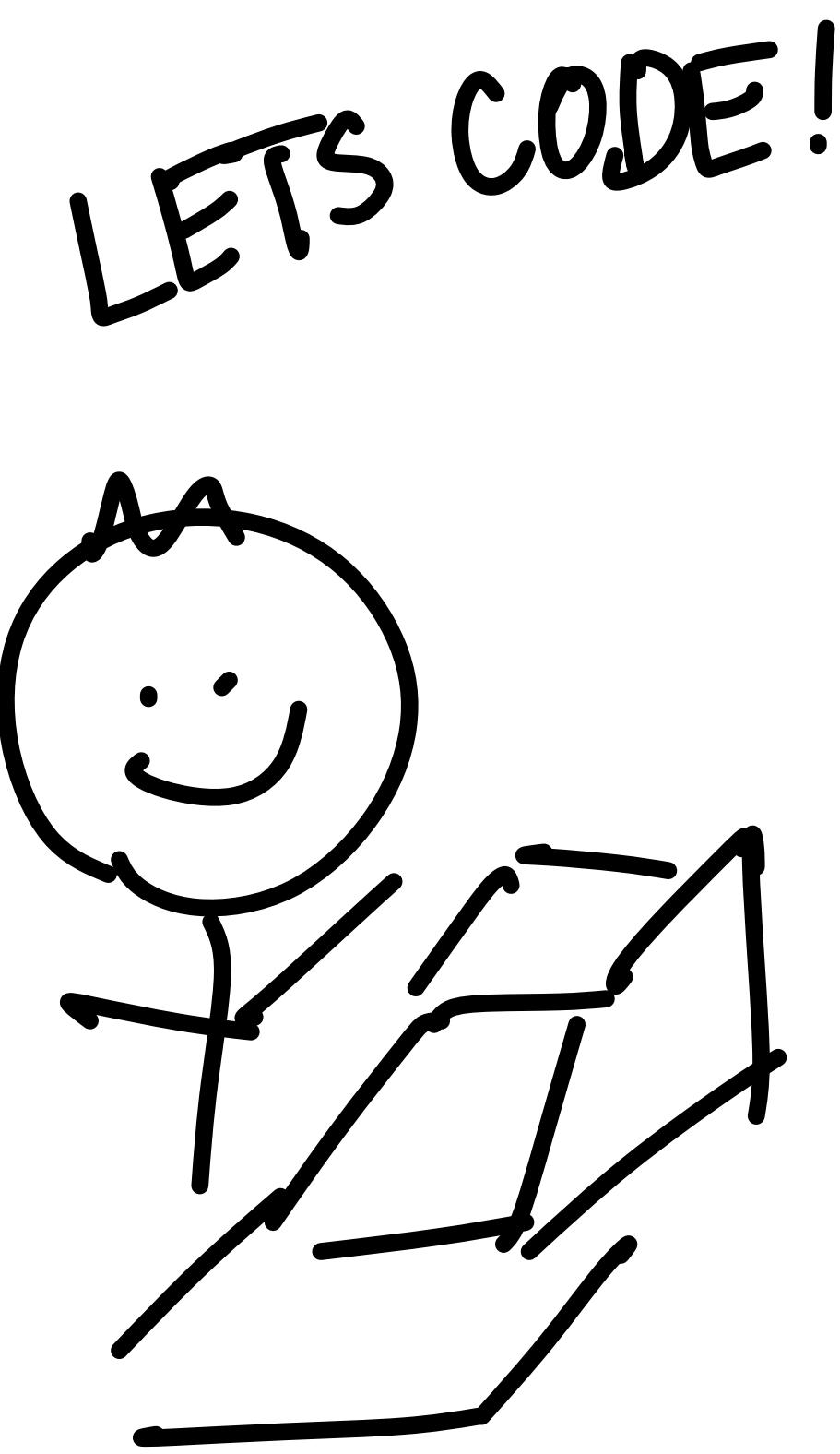


Know Your Audience & deliver value



The best visualisation software or library is the one that allows you to make the figures you need.

The structure of the project



https://github.com/Kislovskiy/talks/tree/main/2023_PyData_Berlin

Continuous integration

Kislovskiy / talks Public

< Generate PDF

✓ update README #44

Summary

Triggered via push 1 minute ago

Kislovskiy pushed → 1ffc9fd 2023-pycon-de

Status Success Total duration 1m 21s Artifacts 1

Jobs

generate-pdf

Run details

Usage

Workflow file

2023-pycon-de-python-pdf-workflow.yml

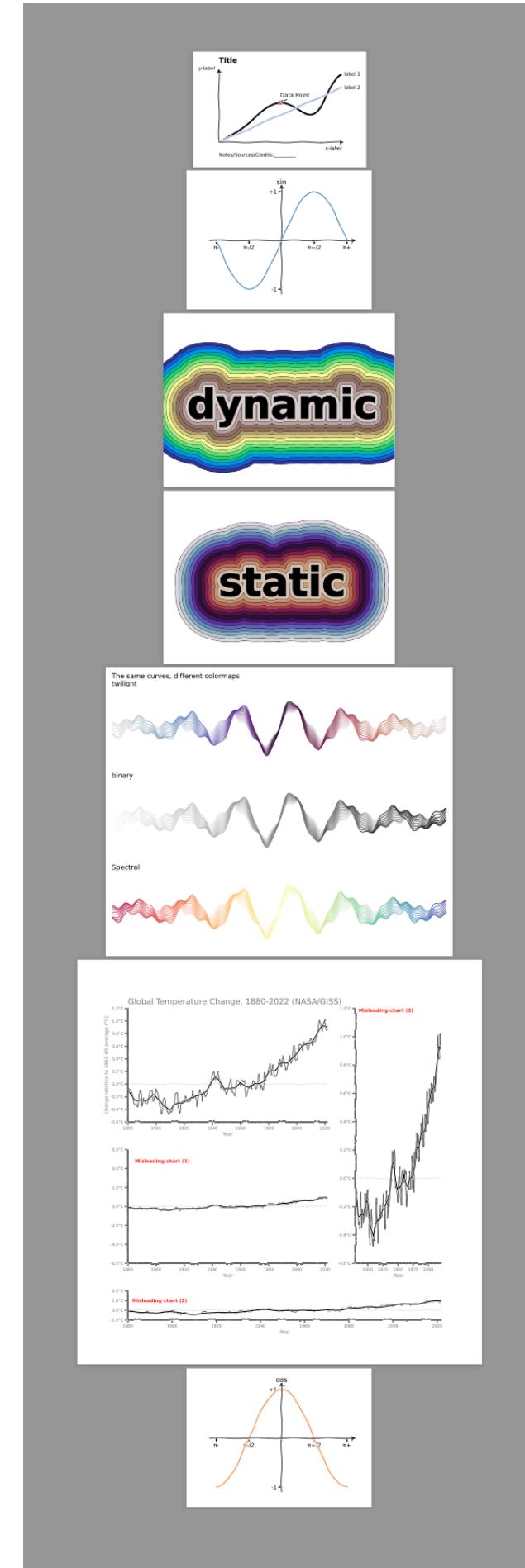
on: push

generate-pdf 1m 13s

Artifacts

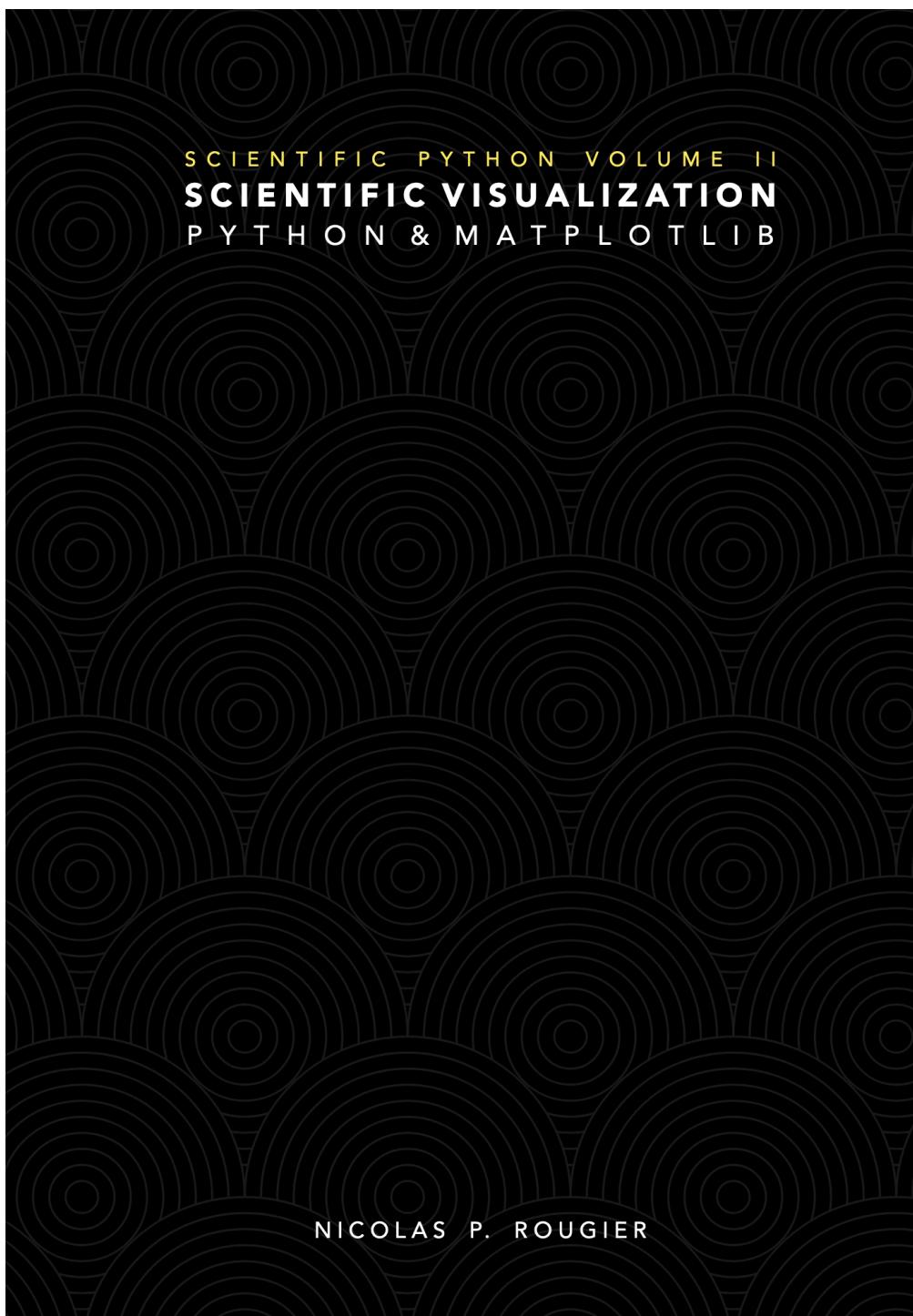
Produced during runtime

Name	Size
galery.pdf	3.13 MB



https://github.com/Kislovskiy/talks/tree/main/2023_PyData_Berlin

Design principles



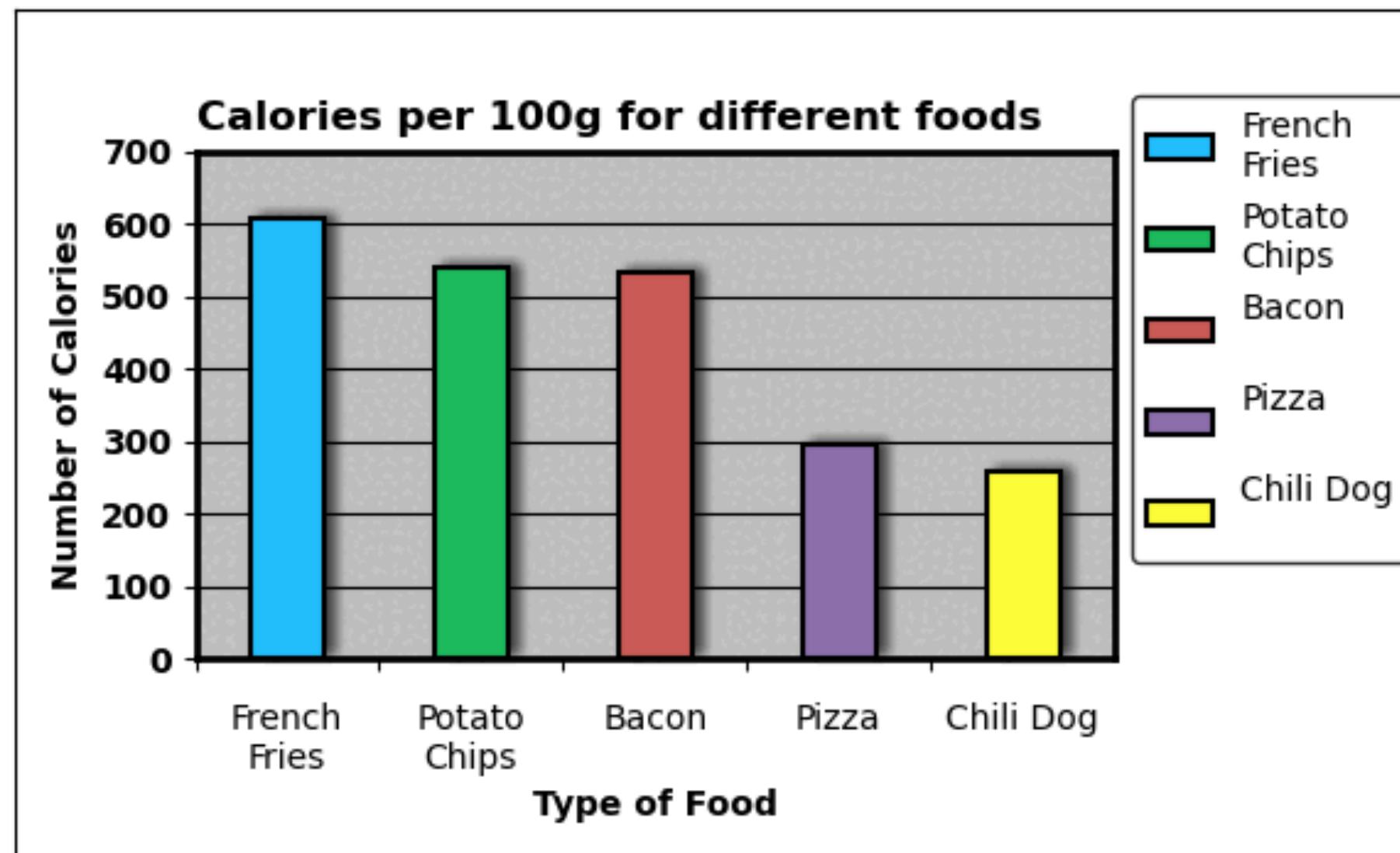
<https://github.com/rougier/scientific-visualization-book>

- Know Your Audience
- Identify Your Message
- Adapt the Figure to the Support Medium
- Captions Are Not Optional
- Do Not Trust the Defaults
- Use Color Effectively
- Do Not Mislead the Reader
- Avoid "Chartjunk"
- Message Trumps Beauty
- Get the Right Tool



Less is more

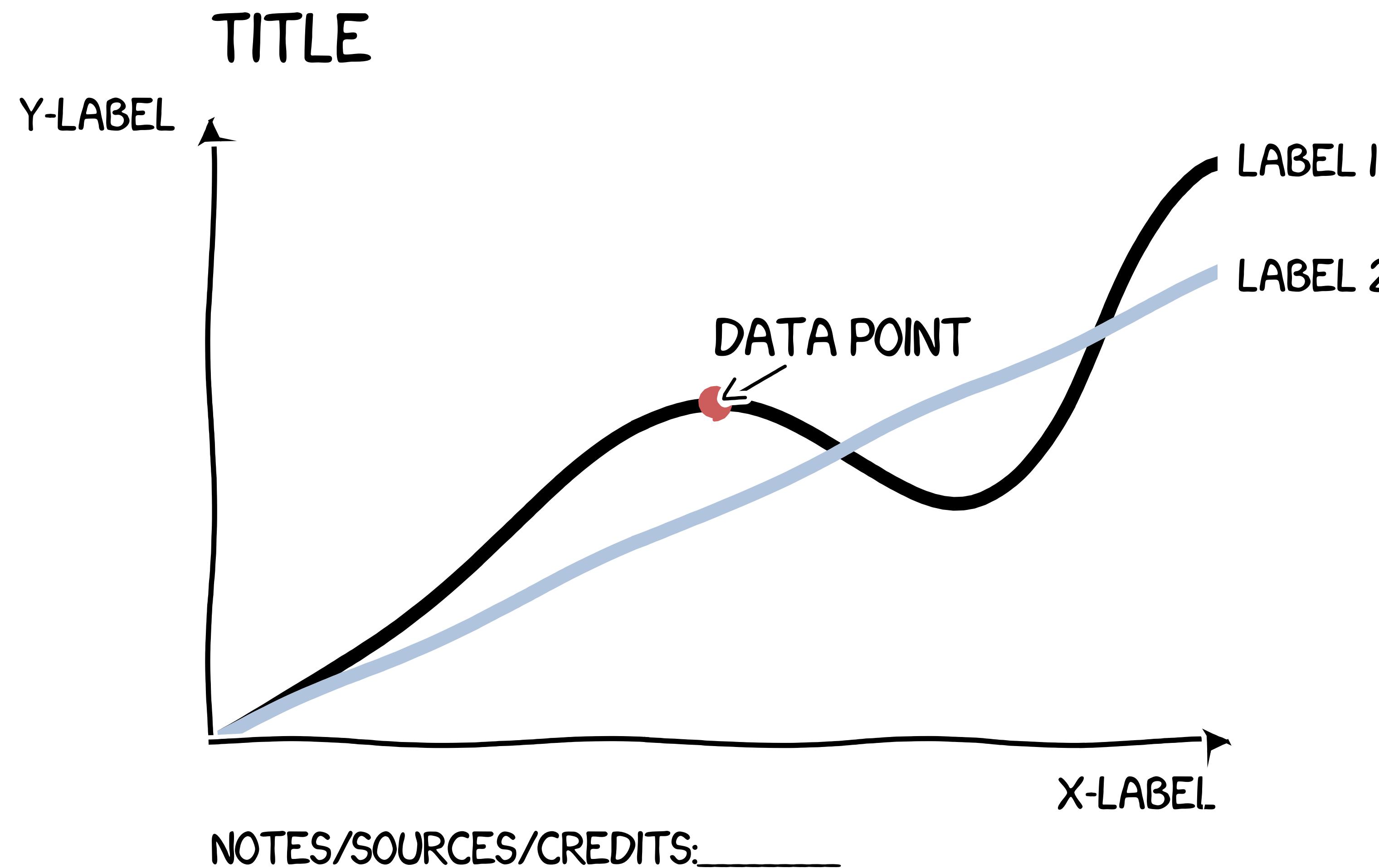
Less is More



A remake of www.darkhorseanalytics.com

Made with matplotlib

<https://github.com/rougier/scientific-visualization-book>





```
>>>from matplotlib import pyplot as plt

>>>fig, ax = plt.subplots()

>>> type(fig)
matplotlib.figure.Figure

>>> type(ax)
matplotlib.axes._axes.Axes
```



```
def plot_common_chart():
    fig, ax = plt.subplots()

    ax.set_title('...')

    ax.set_ylabel('...')
    ax.set_xlabel('...')

    ax.plot('...')
    ax.plot('...')
    ax.scatter('...')
    ax.annotate('...')

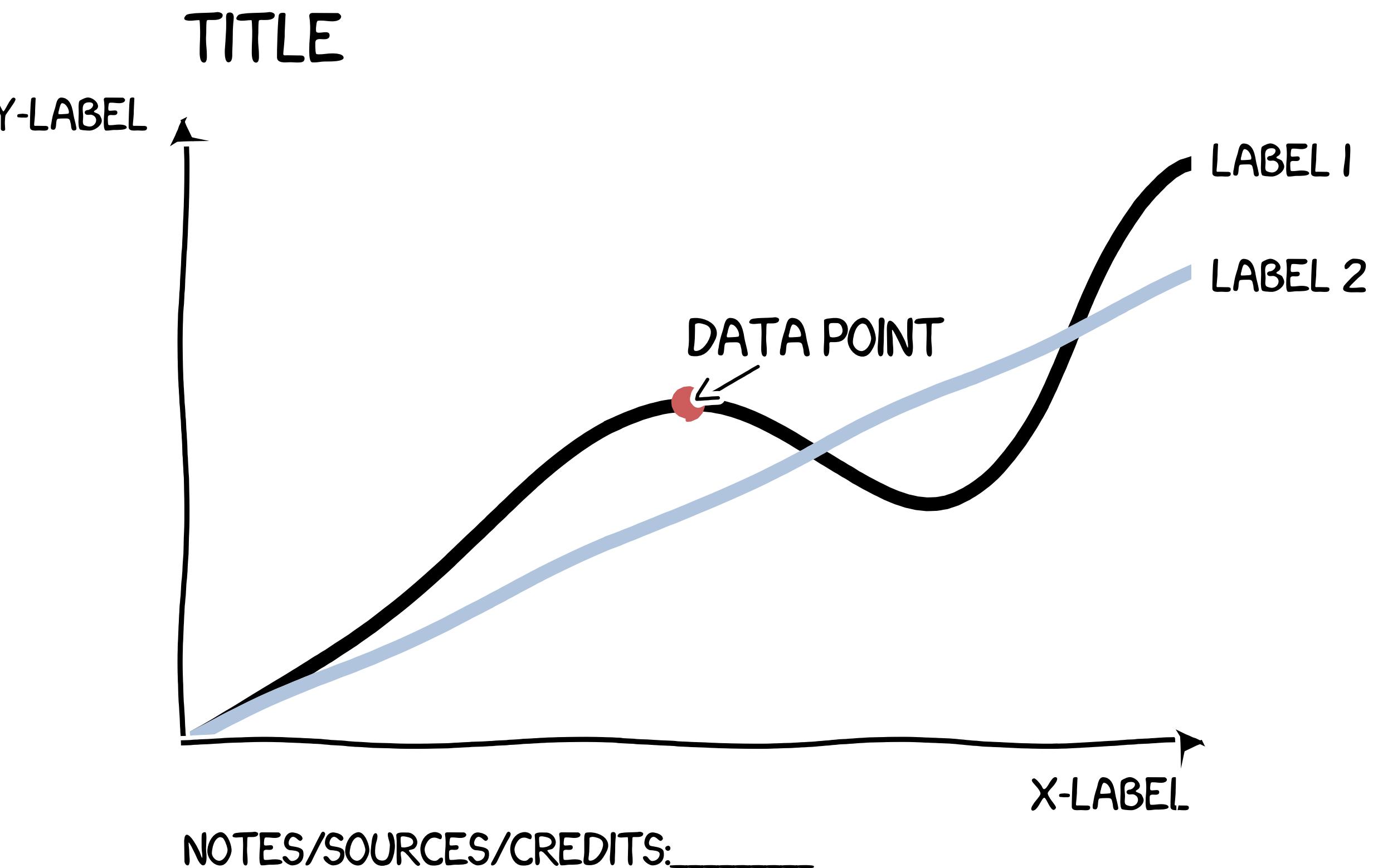
    ax.set_ylim('...')
    ax.set_xlim('...')

    add_labels_to_lines(ax, "label 1")
    add_labels_to_lines(ax, "label 2")
    despine(ax)
    add_arrows_to_axis(ax)
    remove_ticks(ax)

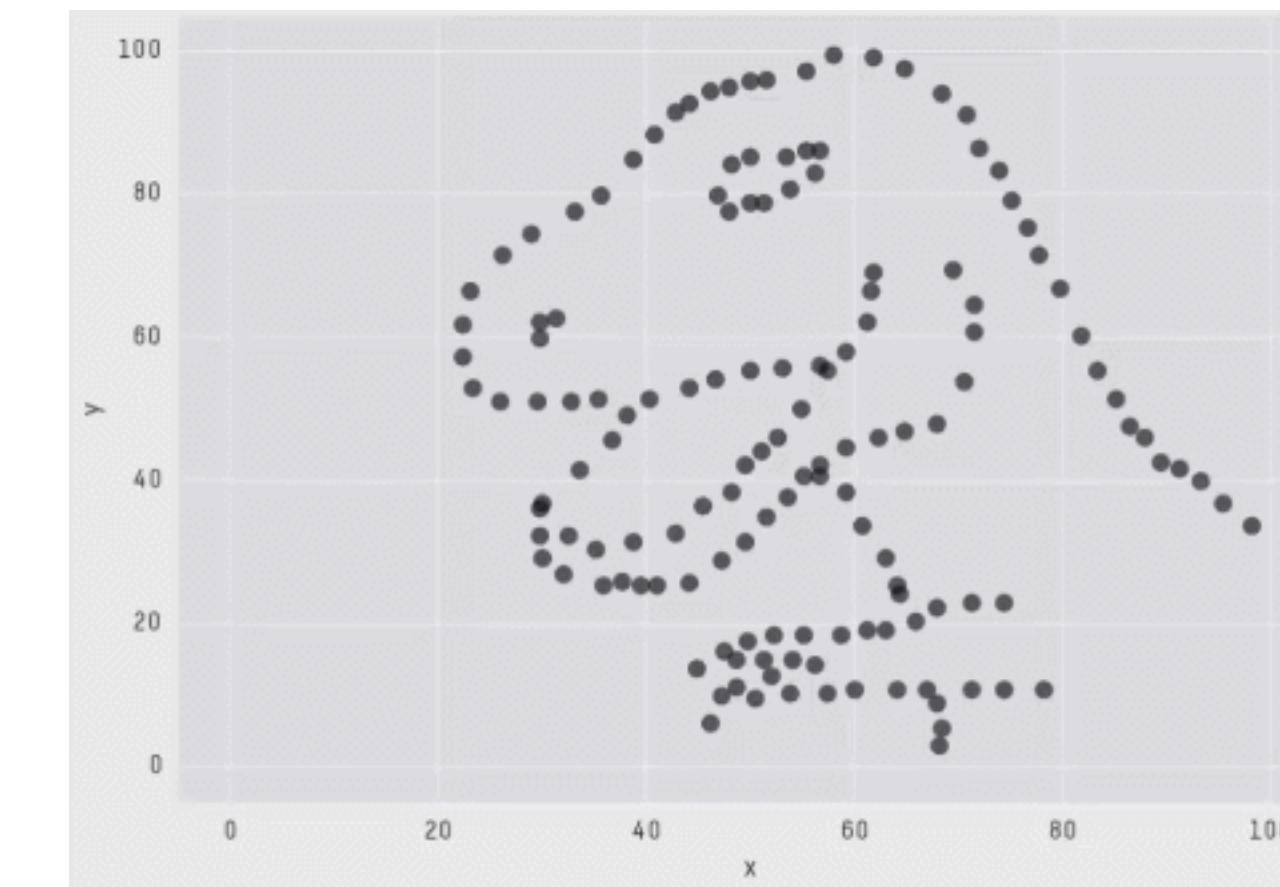
    ax.text('...')
    fig.tight_layout()

    return fig

if __name__ == "__main__":
    ...
    fig = plot_common_chart()
    ...
    fig.savefig('...')
```



Verify & Validate



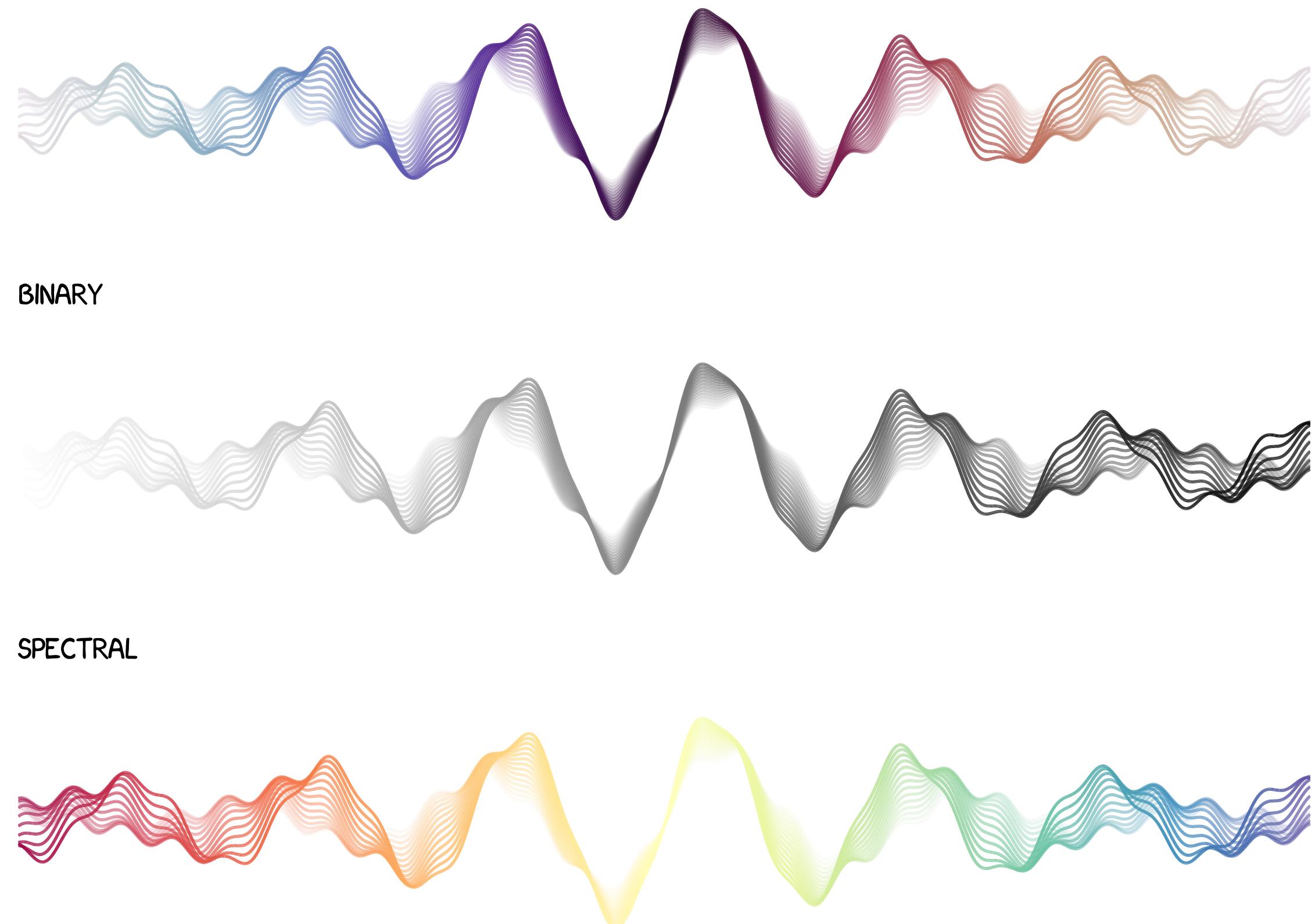
<https://www.autodesk.com/research/publications/same-stats-different-graphs>

```
• .github
  └── workflows
    └── 2023-pycon-de-python-pdf-workflow.yml
• 2023_PyData_Berlin
  └── data
  └── environment.yml
  └── README.md
  └── results
  └── src
    ├── assemble_plots.py
    ├── less-is-more.py
    ├── plot_common_chart.py
    ├── plot_cos.py
    ├── plot_data.py
    ├── plot_nothing.py
    ├── plot_sin.py
    ├── plot_text.py
    ├── plotting_utils.py
    ├── suboptimal_code
    │   ├── chart.ipynb
    │   ├── misleading.ipynb
    │   └── sin.ipynb
    └── transform_data.py
  └── tests
    ├── test_data.py
    ├── test_plot_cos.py
    └── test_transform_data.py
└── LICENSE
```

Decouple data from visualisations



THE SAME CURVES, DIFFERENT COLORMAPS
TWILIGHT





```
from matplotlib import pyplot as plt
import numpy as np
import pandas as pd
from matplotlib.collections import LineCollection
from plotting_utils import despine, remove_ticks

def visualize_lines_with_colormap(...):
    ...
    L = LineCollection(...)
    axis.add_collection(L)

def visualize_curves(...):
    ...
    for i, alpha in enumerate(...):
        visualize_lines_with_colormap(...)

    despine(...)
    remove_ticks(...)
    axis.set_xlim(...)
    axis.set_ylim(...)

    return axis

def plot_cmap_waves():
    df = pd.read_csv(...)

    fig, ax = plt.subplots(...)
    ax[0] = visualize_curves(...)
    ax[0].set_title(...)

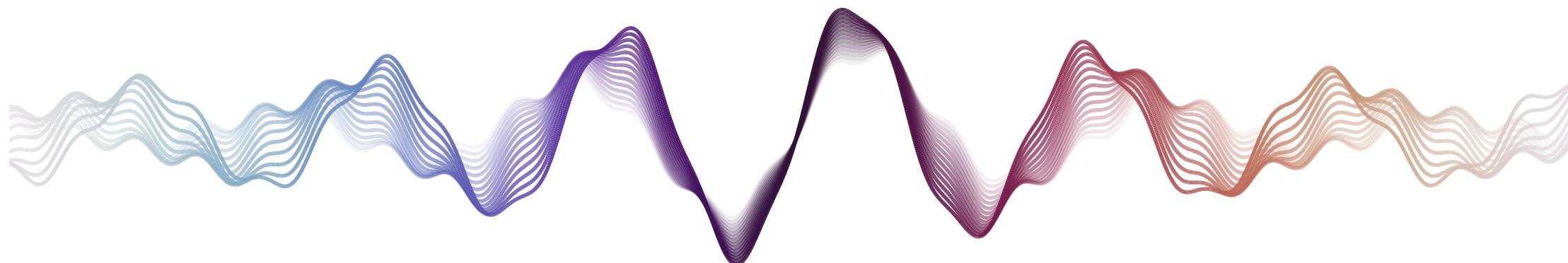
    ax[1] = visualize_curves(...)
    ax[1].set_title(...)

    ax[2] = visualize_curves(...)
    ax[2].set_title(...)

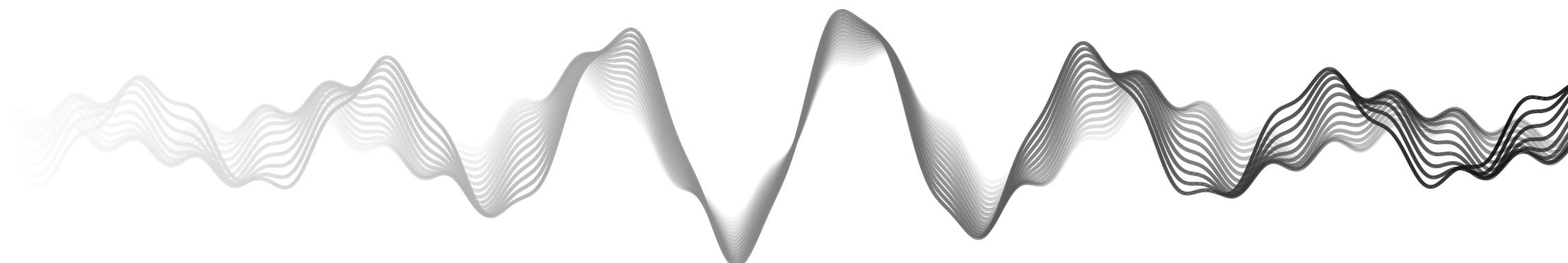
    fig.tight_layout()
    return fig

if __name__ == "__main__":
    ...
    fig = plot_cmap_waves()
    fig.savefig(...)
```

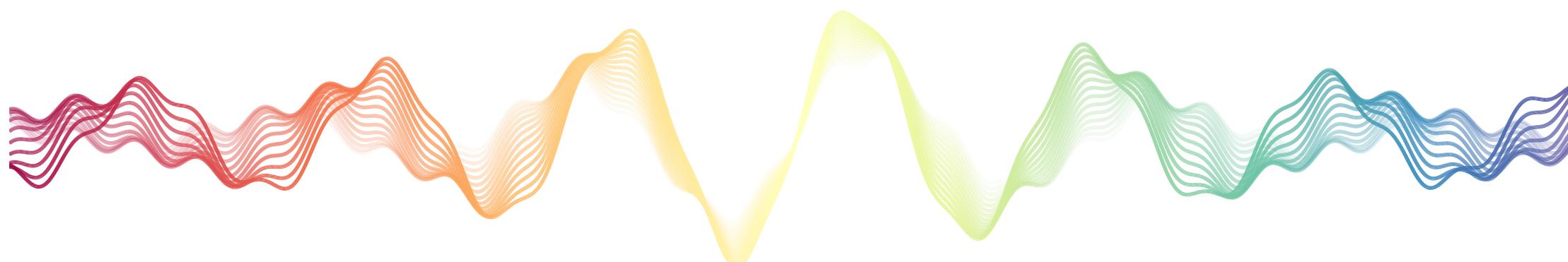
THE SAME CURVES, DIFFERENT COLORMAPS
TWILIGHT



BINARY



SPECTRAL



https://github.com/Kislovskiy/talks/tree/main/2023_PyData_Berlin

```

from matplotlib import pyplot as plt
import matplotlib.gridspec as gridspec
from plotting_utils import format_yticklabels_with_celsius, despine
import pandas as pd

def visualize_global_temp(ax):
    pd.read_csv(...).plot(...)
    despine(ax)
    ax.hlines(...)
    ax.set_xlim(...)
    ax.set_xlabel(...)
    return ax

def plot_misleading_chart():
    fig, ax = plt.subplots()
    visualize_global_temp(ax)
    ax.set_title('... ')
    ax.set_ylabel('... ')
    fig = plt.figure()
    gspec = gridspec.GridSpec(...)

    ax1 = fig.add_subplot(gspec[:5, :8])
    visualize_global_temp(ax1)
    ax1.set_title('... ')
    ax1.set_ylabel('... ')
    format_yticklabels_with_celsius(ax1)

    ax2 = fig.add_subplot(gspec[5:10, :8])
    visualize_global_temp(ax2)
    ax2.set_ylim(... )
    ax2.text(... )
    format_yticklabels_with_celsius(ax2)

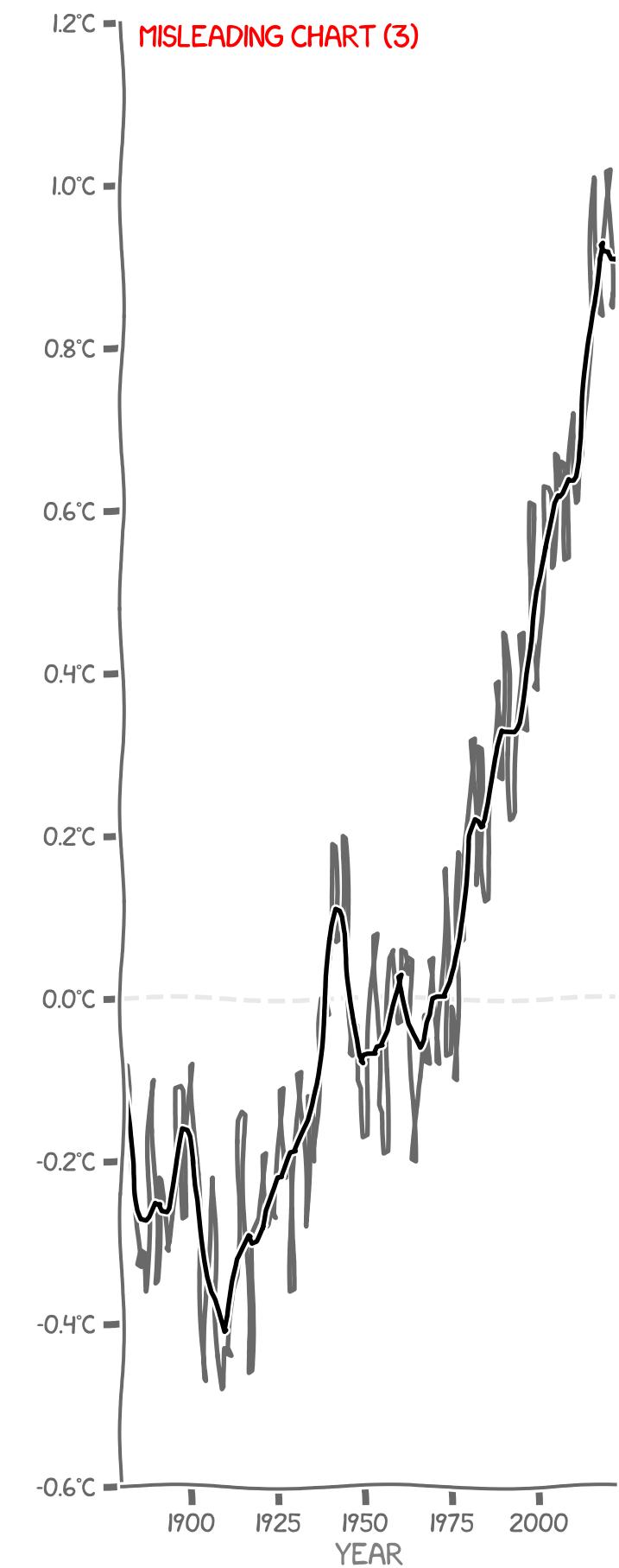
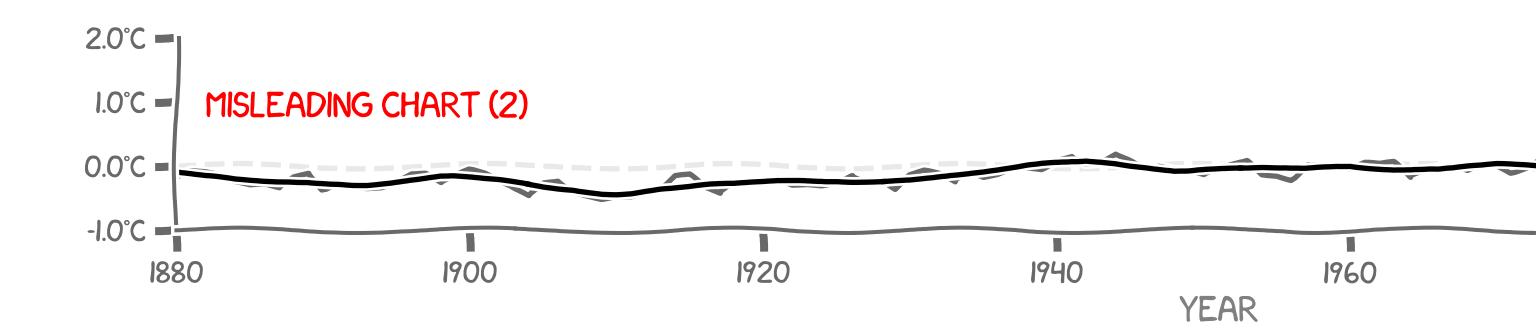
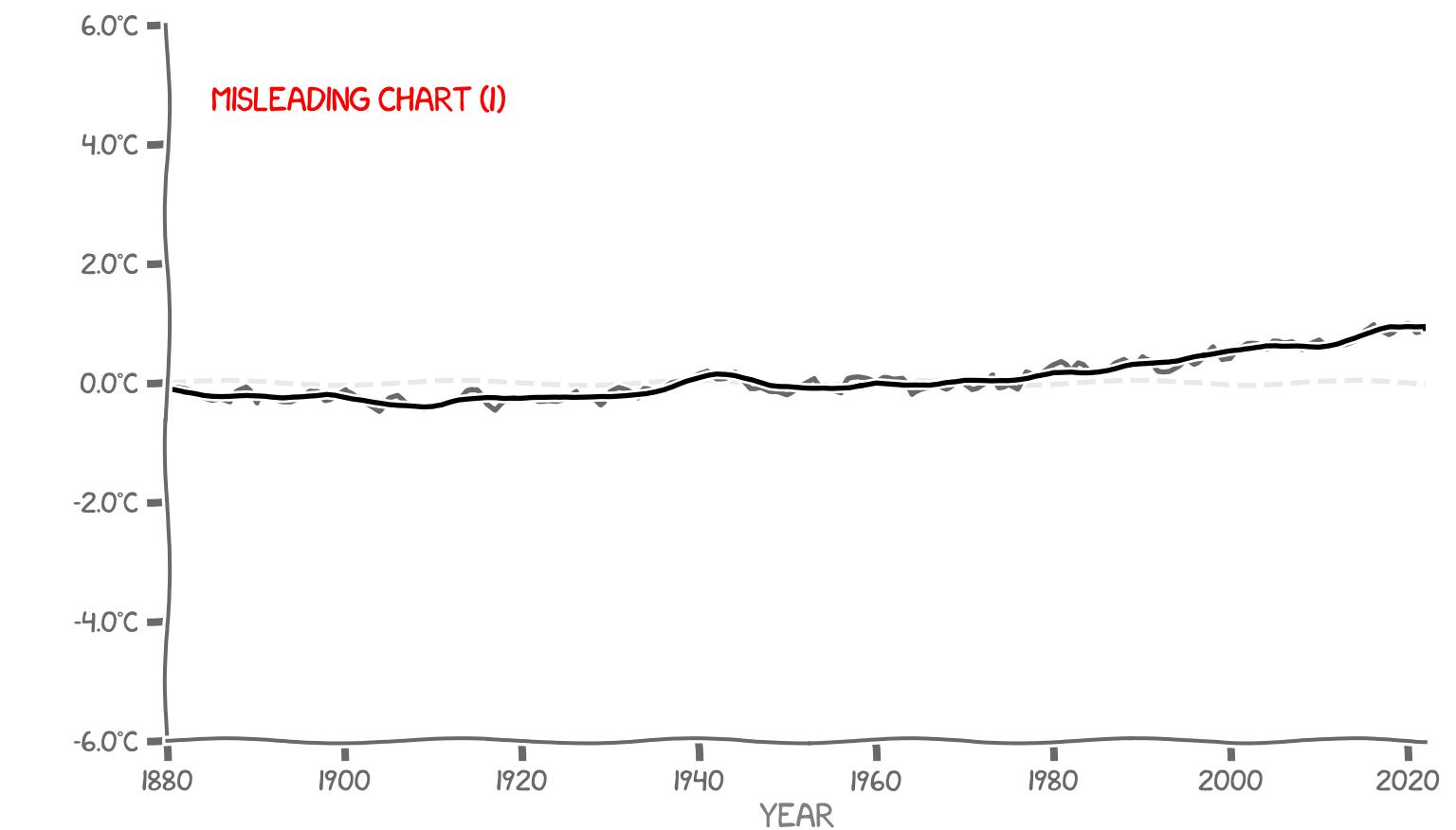
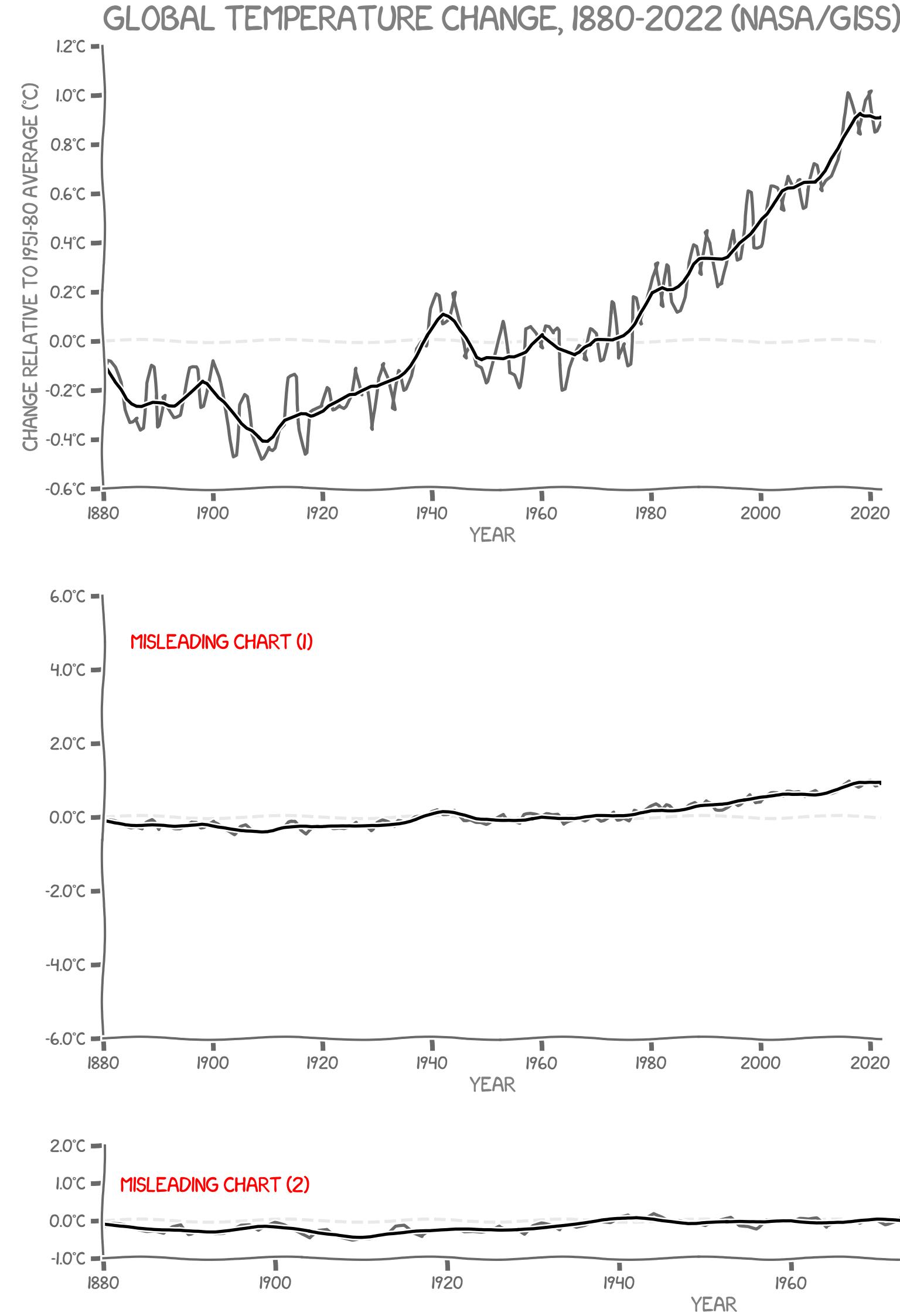
    ax3 = fig.add_subplot(gspec[10:, :])
    visualize_global_temp(ax3)
    ax3.text(... )
    format_yticklabels_with_celsius(ax3)

    ax4 = fig.add_subplot(gspec[:10, 8:])
    visualize_global_temp(ax4)
    ax4.text(... )
    format_yticklabels_with_celsius(ax4)

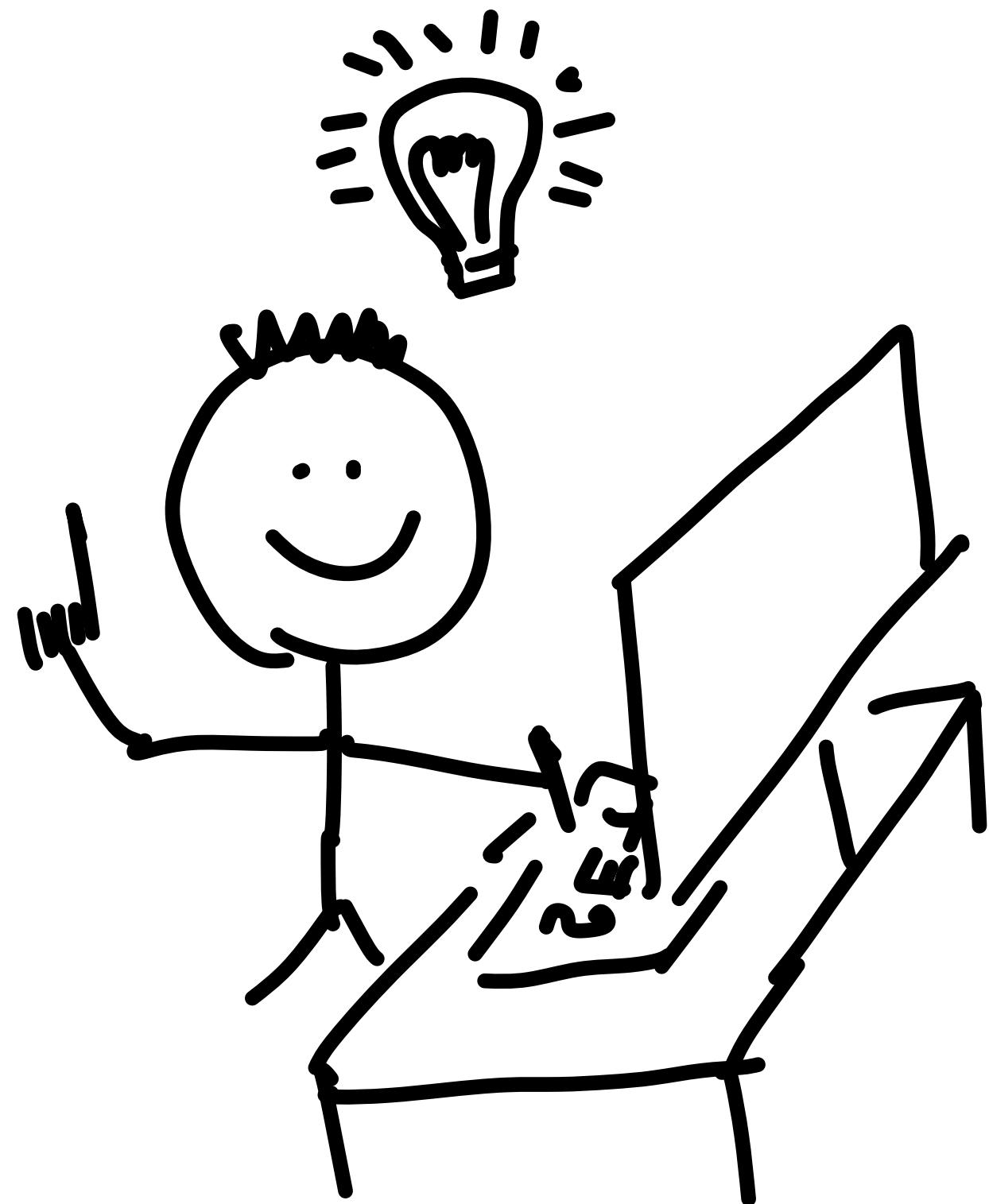
    return fig

if __name__ == "__main__":
    ...
    fig = plot_misleading_chart()
    ...
    fig.savefig(... )

```



Write tests



```
import pandas as pd\n\ndef test_csv_values():\n    ...\n    data = pd.read_csv(...)\n    assert abs(data["sin"]).all() <= 1
```

https://github.com/Kislovskiy/talks/tree/main/2023_PyData_Berlin

Version control



Kislovskiy / talks Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

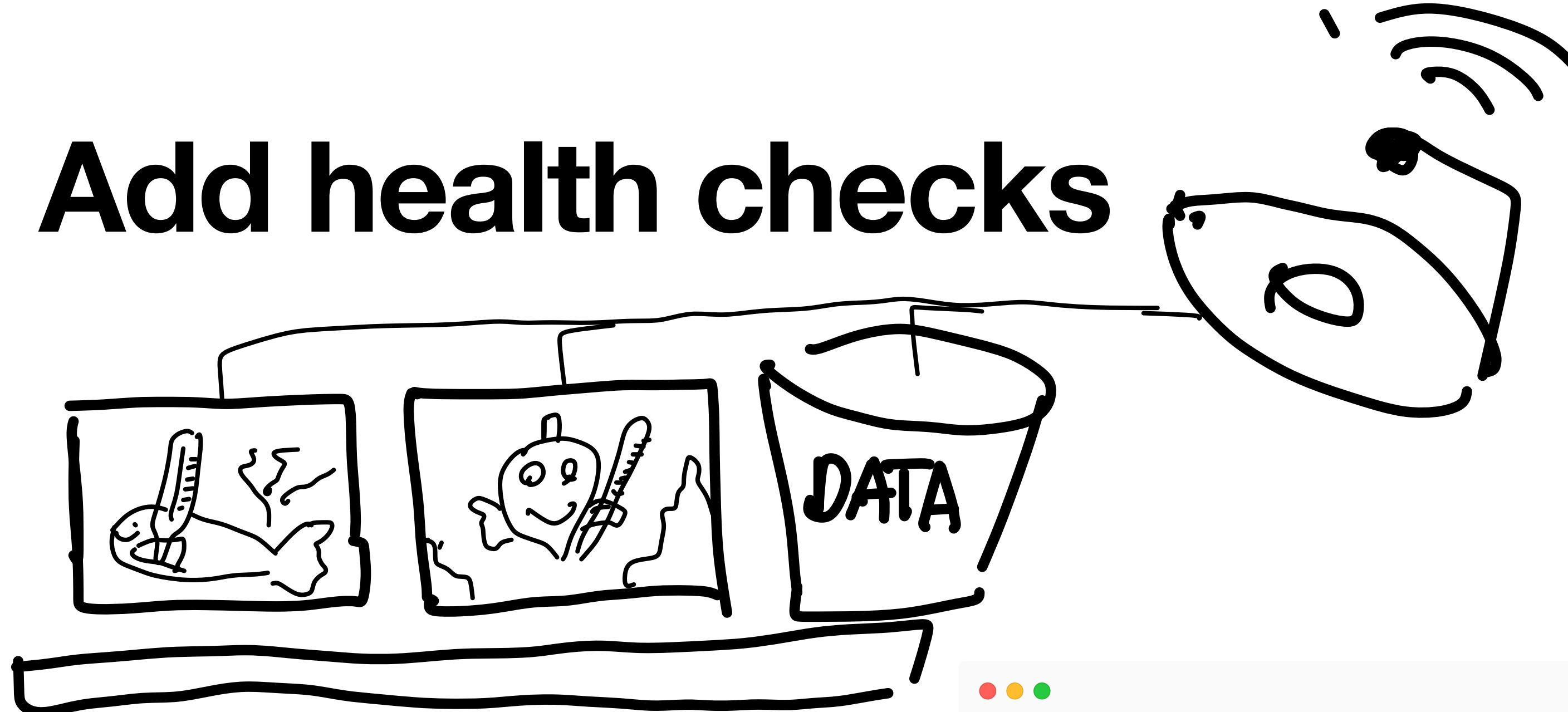
History for talks / 2023_PyData_Berlin / src / plot_cos.py

- o Commits on Apr 5, 2023
 - latest updates
Kislovskiy committed 7 minutes ago ✓ Verified d5c8475
- o Commits on Apr 4, 2023
 - add svg
Kislovskiy committed yesterday ✓ Verified c64bf1c
- o Commits on Apr 1, 2023
 - prettyfy
Kislovskiy committed 4 days ago Verified 480ac83
- o Commits on Mar 31, 2023
 - fix typos
Kislovskiy committed 5 days ago Verified 27c478d
 - kind of working version
Kislovskiy committed last week Verified 9306854
- o End of commit history for this file

Newer Older

https://github.com/Kislovskiy/talks/tree/main/2023_PyData_Berlin

Add health checks



```
name: Generate PDF
on:
  push:
    branches: [2023-pycon-de*]

jobs:
  generate-pdf:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Setup conda environment
        uses: conda-incubator/setup-miniconda@v2
        with:
          environment-file: 2023_PyData_Berlin/environment.yml
          activate-environment: 2023_PyData_Berlin

      - name: Generate PDF
        shell: bash -el {0}
        run:
          python 2023_PyData_Berlin/src/assemble_plots.py

      - name: Archive PDF
        uses: actions/upload-artifact@v3
        with:
          name: gallery.pdf
          path: 2023_PyData_Berlin/results/gallery.pdf
```

Kislovskiy / talks (Public)

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Generate PDF latest updates #26

Triggered via push 3 minutes ago Status Success Total duration 1m 17s Artifacts 1

Summary

Jobs generate-pdf

Run details Usage Workflow file

2023-pycon-de-python-pdf-workflow.yml on: push

generate-pdf 1m 1s

Artifacts Produced during runtime Name Size gallery.pdf 100 KB

Some checks haven't completed yet 1 in progress check

Generate PDF / generate-pdf (push) In progress — This check has started...

This branch has no conflicts with the base branch Merging can be performed automatically.

Merge pull request You can also open this in GitHub Desktop or view command line instructions.



```
● ● ●

from importlib.util import module_from_spec, spec_from_file_location
from inspect import getmembers, isfunction
from pathlib import Path

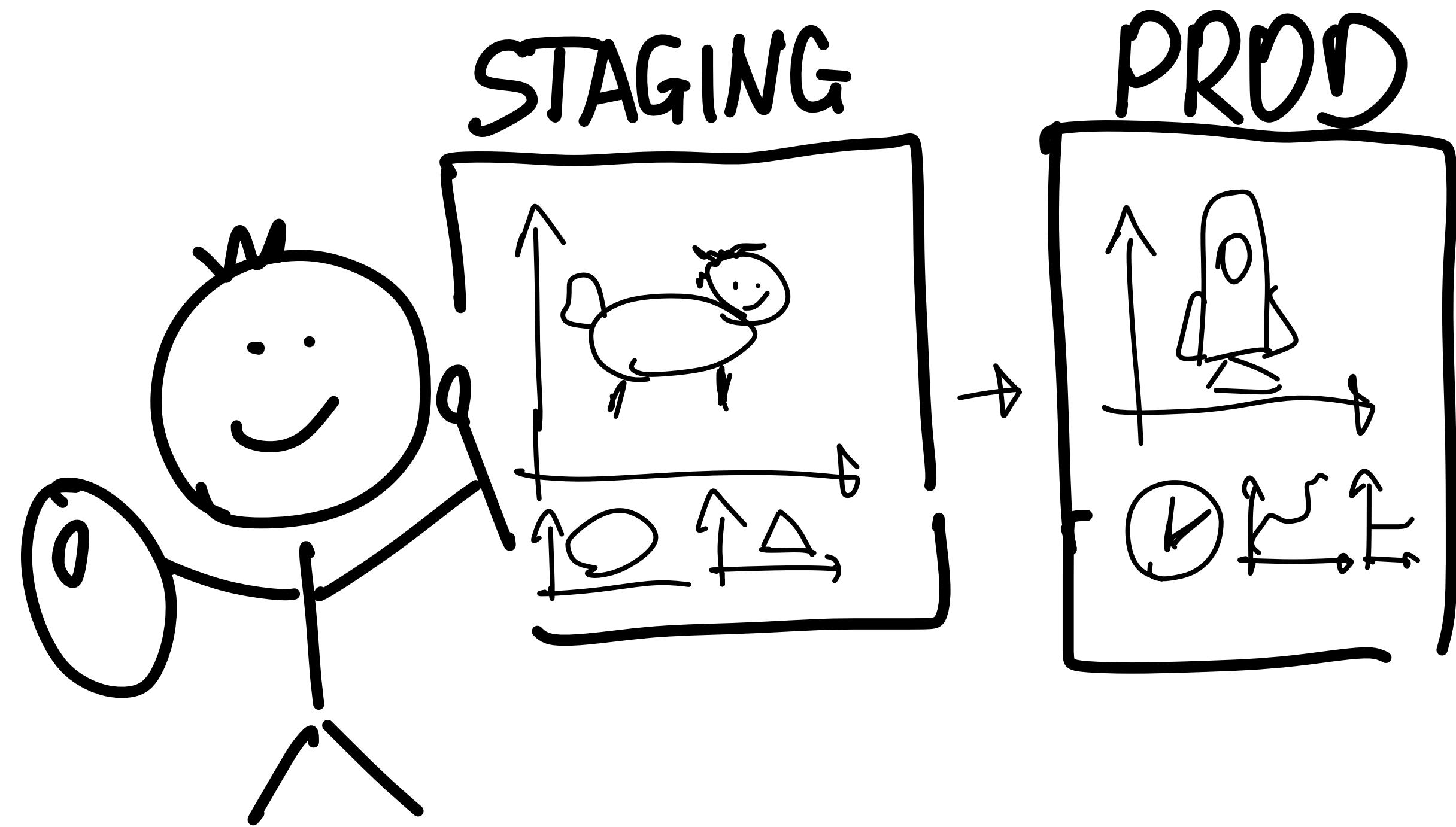
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages

def main():
    # Create a PDF file to store all the plots
    ...
    with PdfPages(...) as pdf:
        for file in Path(__file__).parent.glob("plot_*.py"):
            spec = spec_from_file_location("src", file)
            module = module_from_spec(spec)
            spec.loader.exec_module(module)

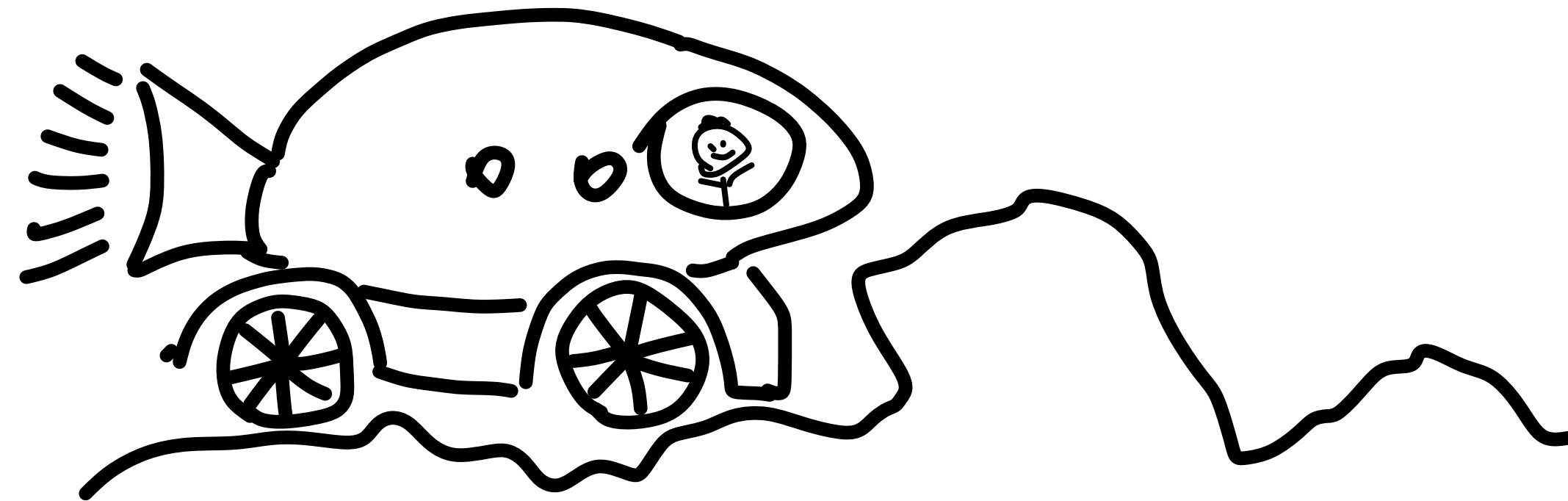
            for name, func in getmembers(module, isfunction):
                if name.startswith("plot_"):
                    fig = func()
                    if fig:
                        pdf.savefig(fig)
                        plt.close(fig)

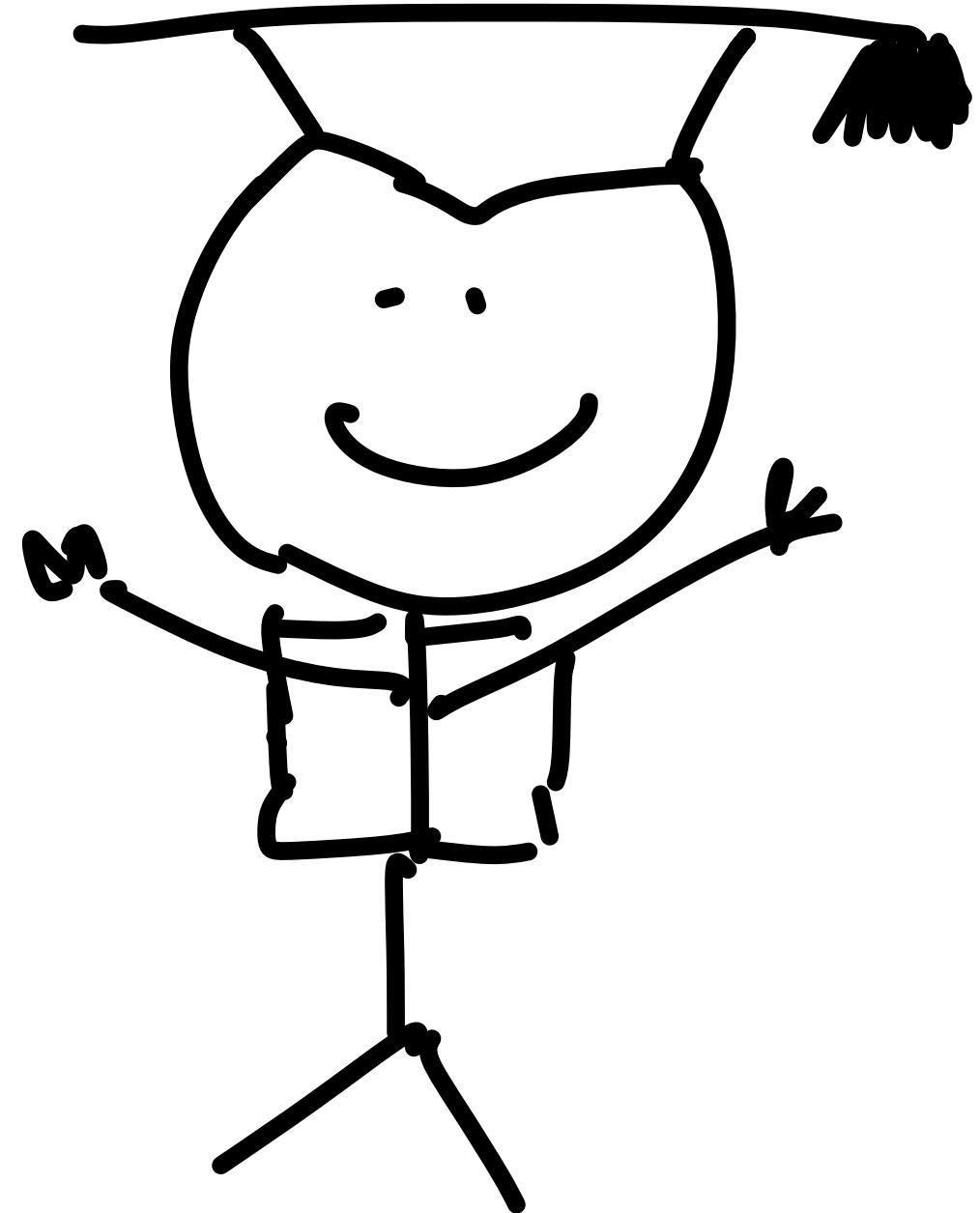
if __name__ == "__main__":
    main()
```

Create a staging environment



Know the limits





- Work collaboratively
- Aim for reproducibility
- Back up data
- Version code
- Set up monitoring
- Create a continuous delivery pipeline

https://github.com/Kislovskiy/talks/tree/main/2023_PyData_Berlin