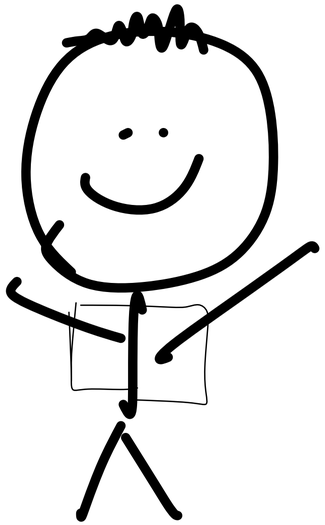
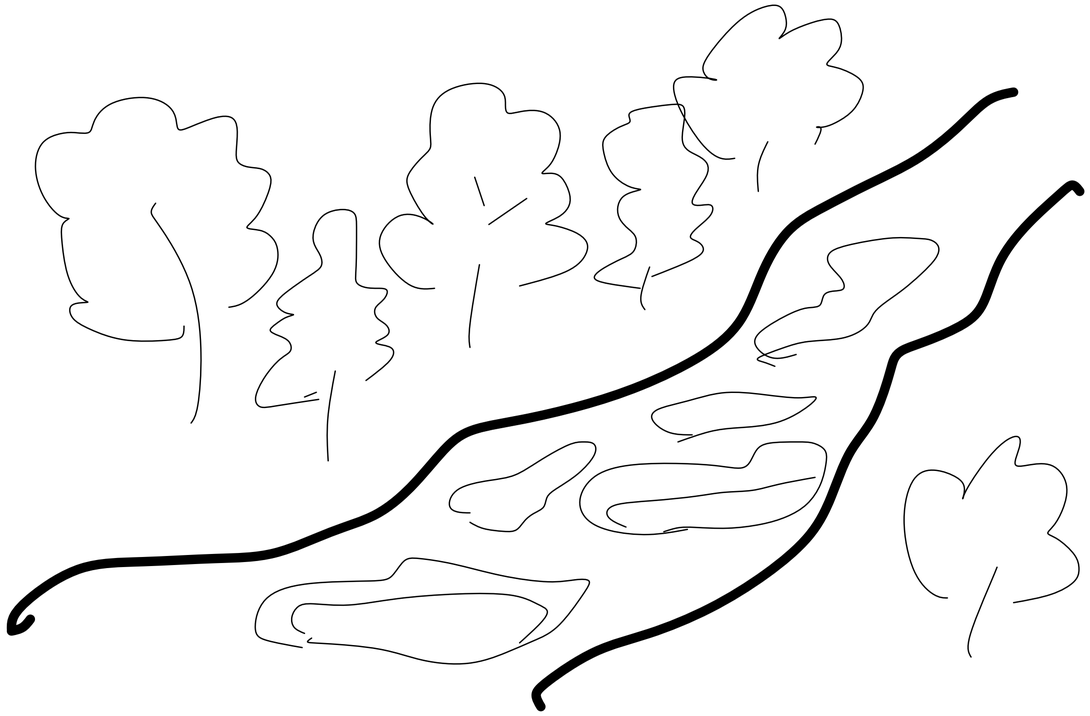
The bumps in the road:
A retrospective on my data visualisation mistakes



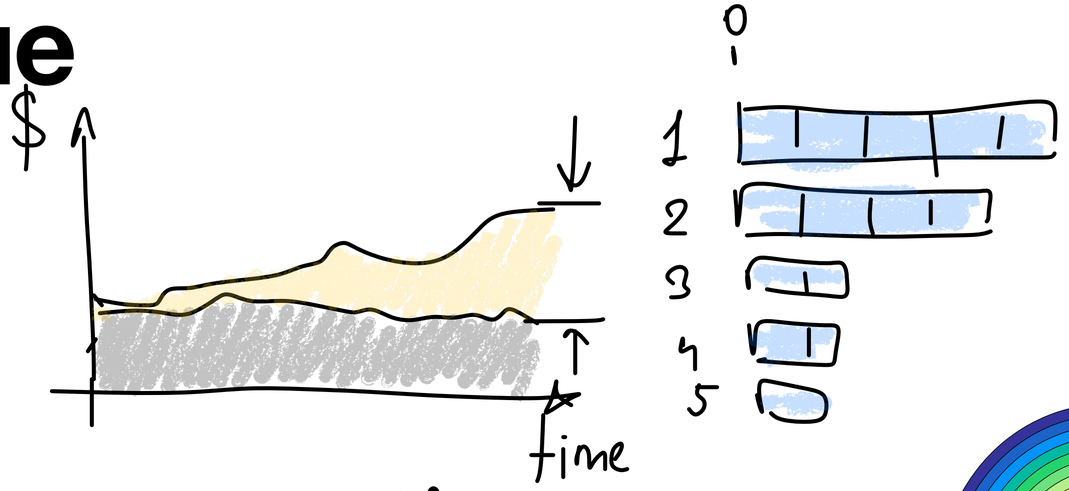


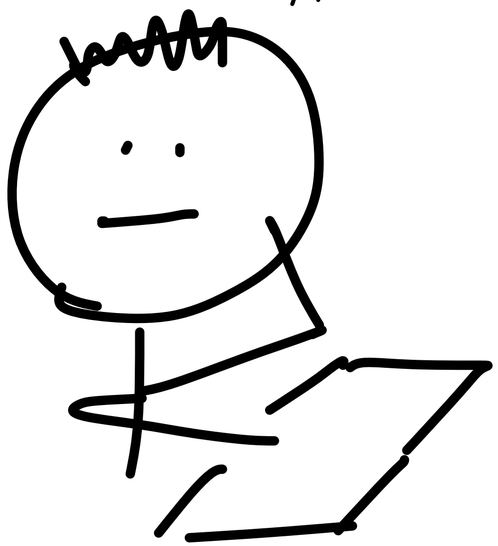
Artem Kislovskiy

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
- 😶 I had to reprocess the data waiting for a couple of days
- 💣 My list of Jupyter notebooks exploded

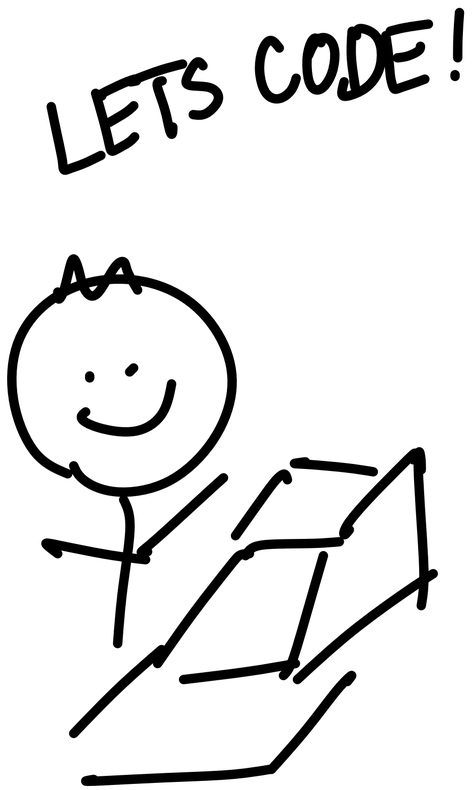
Know Your Audience & deliver value





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

The structure of the project




```
.
├── .github
│   └── workflows
│       └── 2023-pycon-de-python-pdf-workflow.yml
├── 2023_PyData_Berlin
│   ├── data
│   ├── environment.yml
│   ├── README.md
│   ├── results
│   ├── src
│   └── tests
└── LICENSE
```

https://github.com/Kislovskiy/talks/tree/2023-pycon-de/2023_PyData_Berlin

Continuous integration

Kislovskiy / talks Public

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

← Generate PDF

update README #44

Summary

Jobs

generate-pdf

Run details

Usage

Workflow file

Triggered via push 1 minute ago	Status	Total duration	Artifacts
Kislovskiy pushed 1ffc9fd 2023-pycon-de	Success	1m 21s	1

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

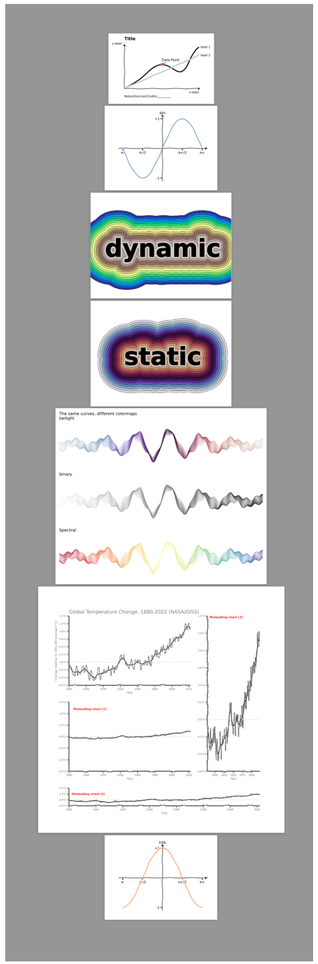
on: push

generate-pdf 1m 13s

Artifacts

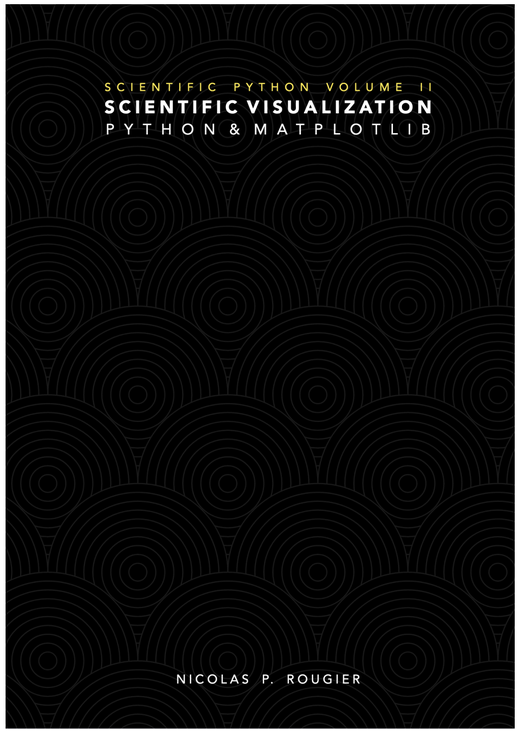
Produced during runtime

Name	Size
galery.pdf	3.13 MB



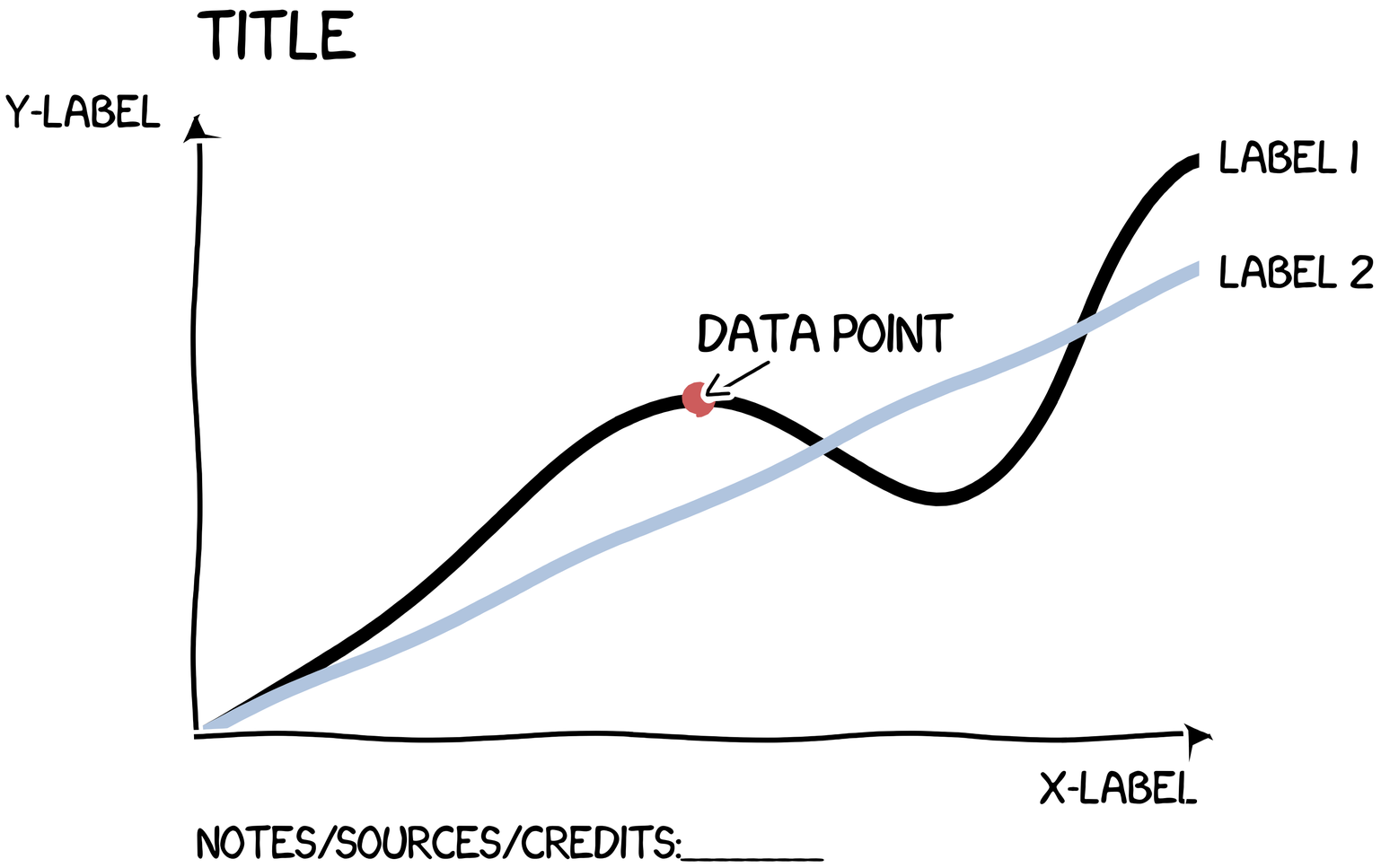
https://github.com/Kislovskiy/talks/tree/2023-pycon-de/2023_PyData_Berlin

Design principles



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

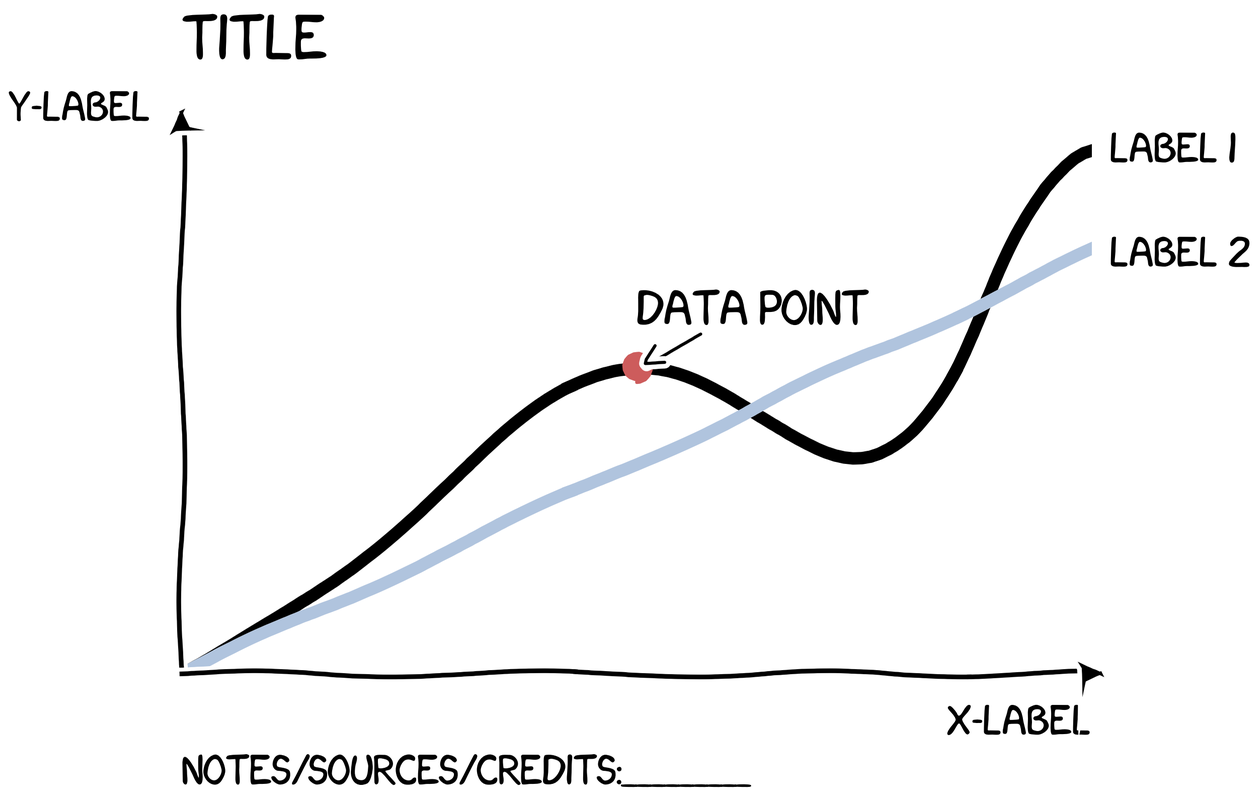
- Know Your Audience
- Identify Your Message
- Adapt the Figure to the Support Medium
- Captions Are Not Optional
- Do Not Trust the Defaults
- Use Color Effectively
- Do Not Mislead the Reader
- Avoid "Chartjunk"
- Message Trumps Beauty
- Get the Right Tool



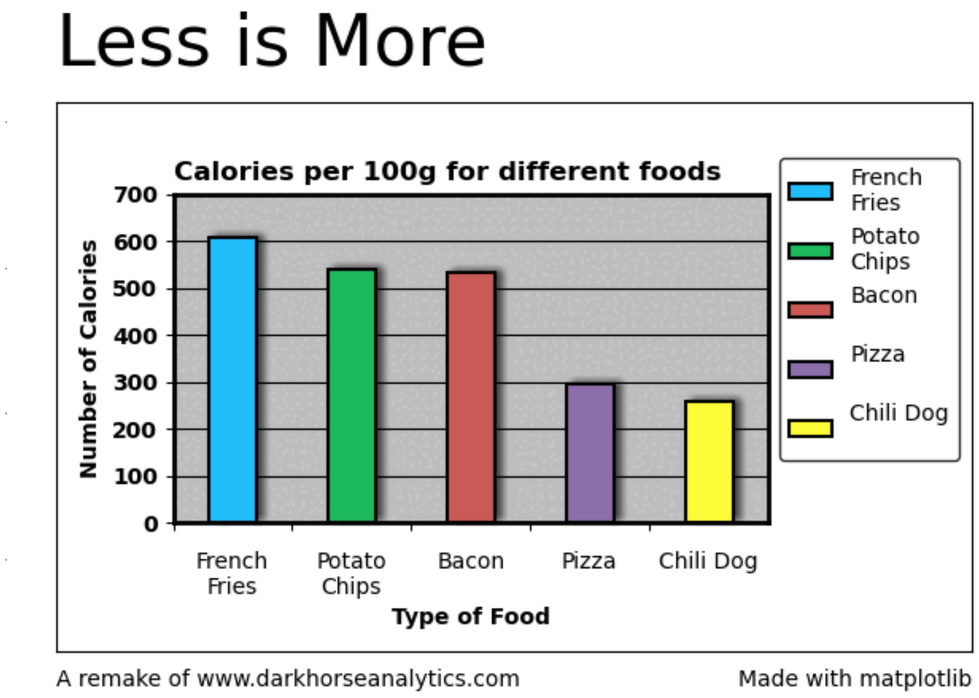
https://github.com/Kislovskiy/talks/tree/2023-pycon-de/2023_PyData_Berlin

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

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

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

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

```python
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(...)
```



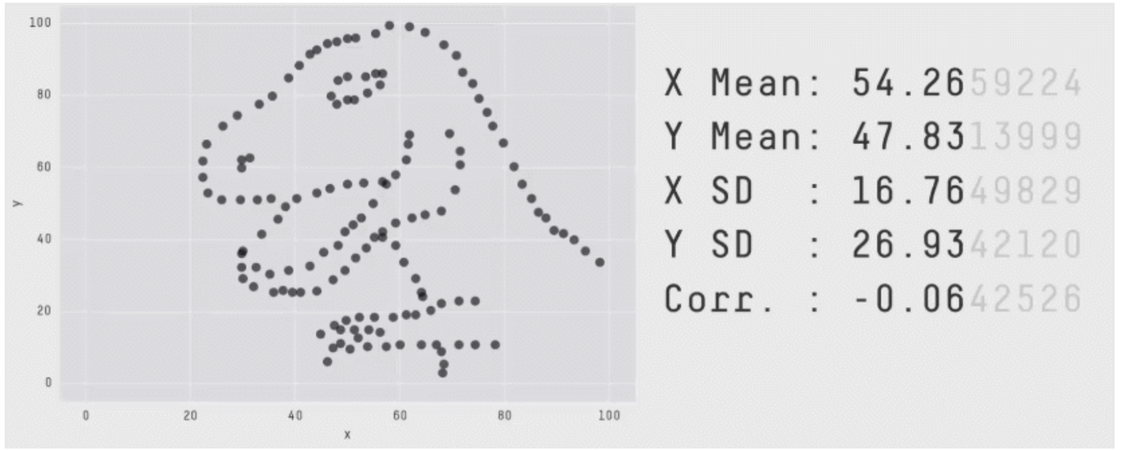
Less is more



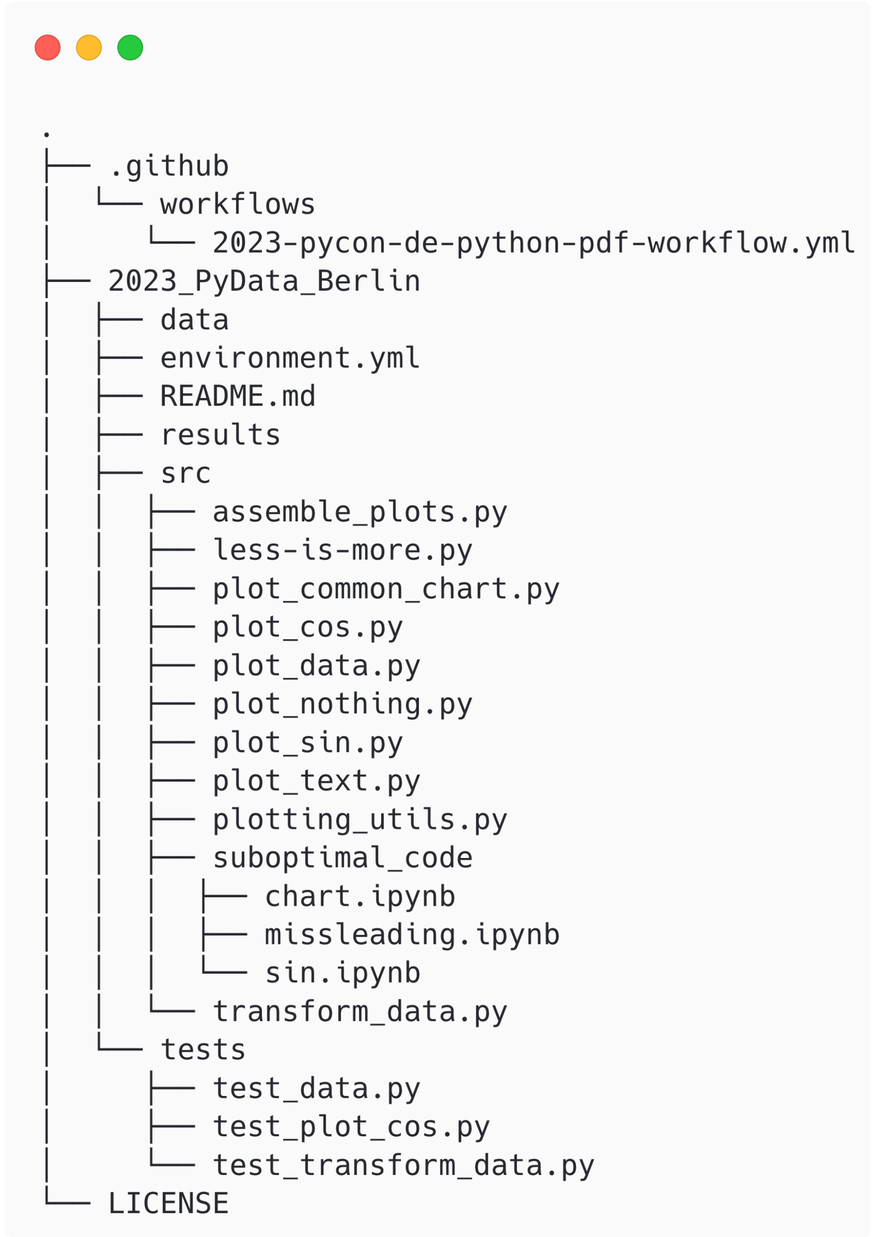
https://github.com/rougier/scientific-visualization-book

Strategies for avoiding the spaghetti graph: https://learning.oreilly.com/library/view/storytelling-with-data/9781119002253/c09.xhtml#c9_5

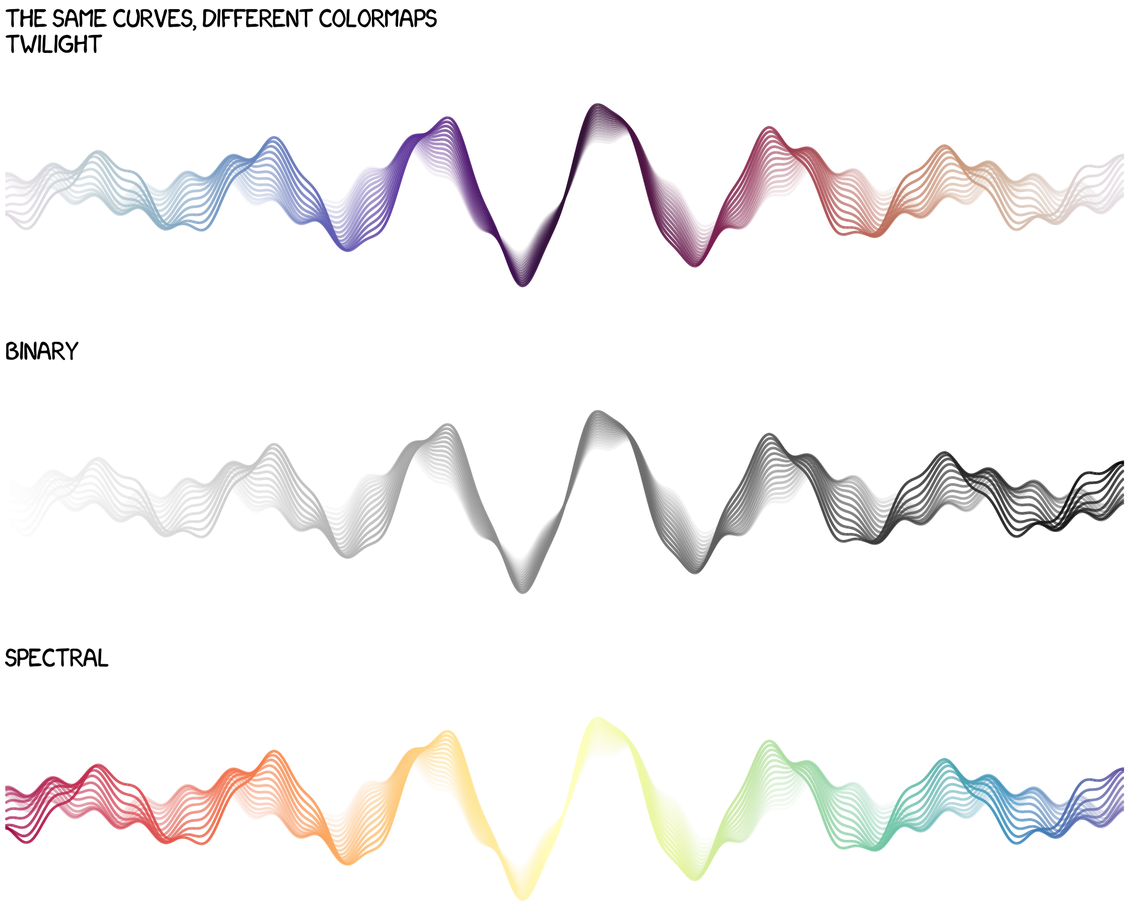
Verify & Validate



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

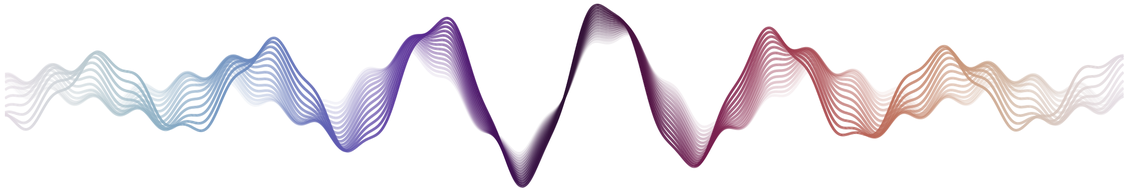

.
├── .github
│   └── workflows
│   └── 2023-pycon-de-python-pdf-workflow.yml
├── 2023_PyData_Berlin
│   ├── data
│   ├── environment.yml
│   ├── README.md
│   ├── results
│   ├── src
│   │   ├── assemble_plots.py
│   │   ├── less-is-more.py
│   │   ├── plot_common_chart.py
│   │   ├── plot_cos.py
│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

Decouple data from visualisations

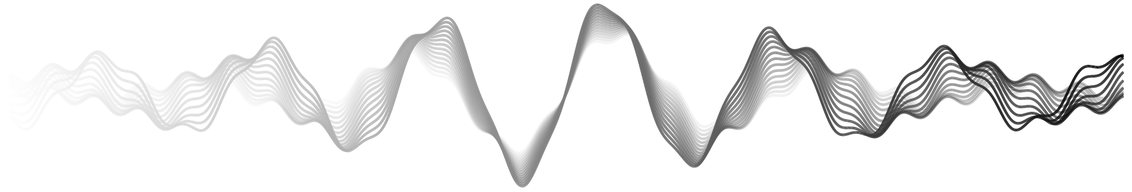


```python
from matplotlib import pyplot as plt
import numpy as np
import pandas as pd
from matplotlib.collections import LineCollection
from plotting_utils import despine, remove_ticks


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(...)
```

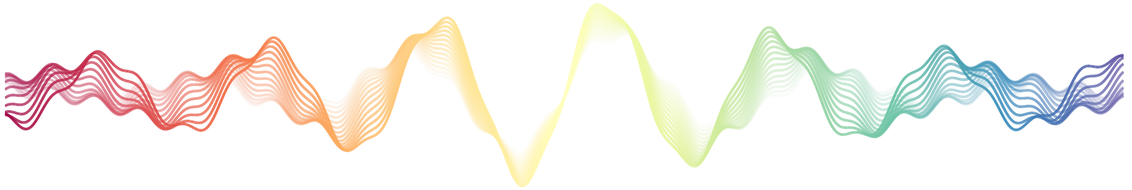
THE SAME CURVES, DIFFERENT COLORMAPS
TWILIGHT



BINARY

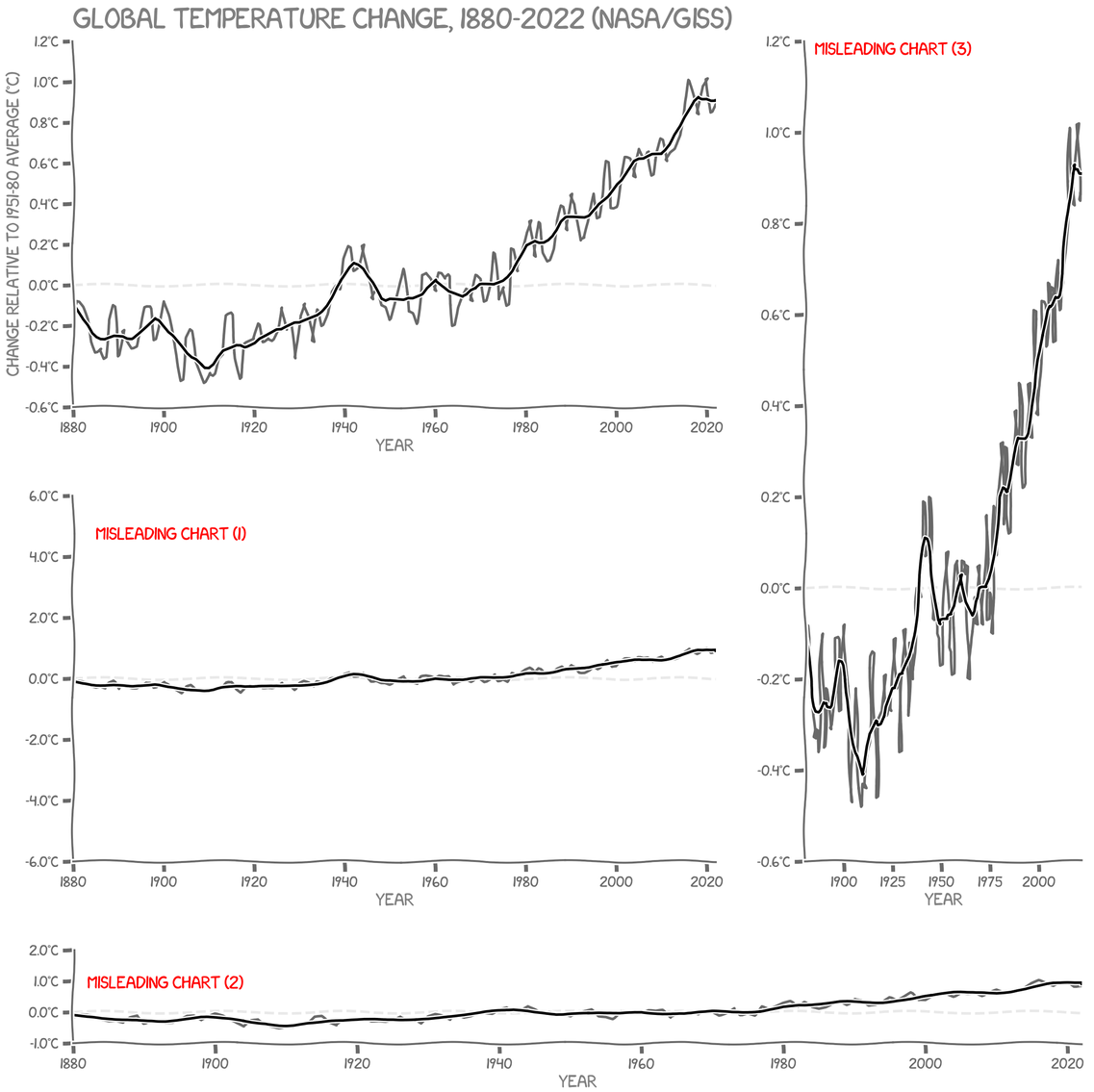


SPECTRAL



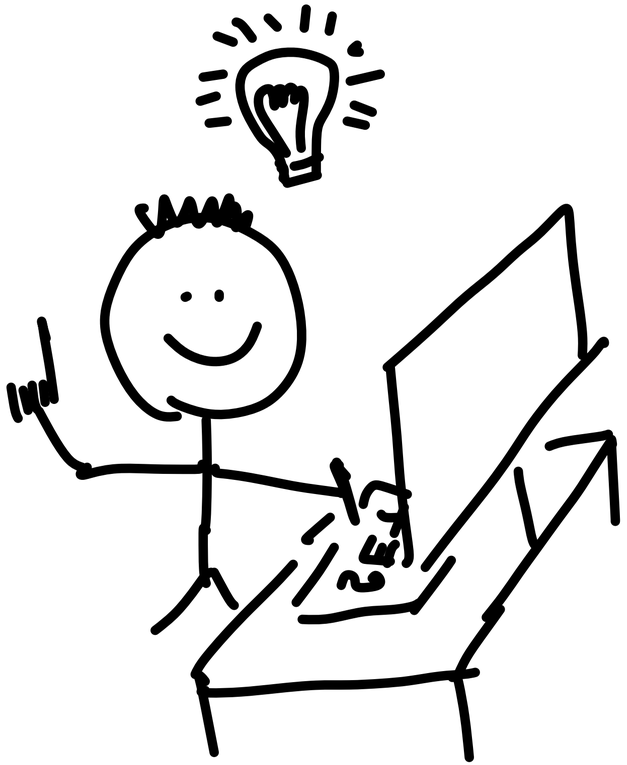
https://github.com/Kislovskiy/talks/tree/2023-pycon-de/2023_PyData_Berlin

```python
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_missleading_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_missleading_chart()
    ...
    fig.savefig(...)
```



https://github.com/Kislovskiy/talks/tree/2023-pycon-de/2023_PyData_Berlin

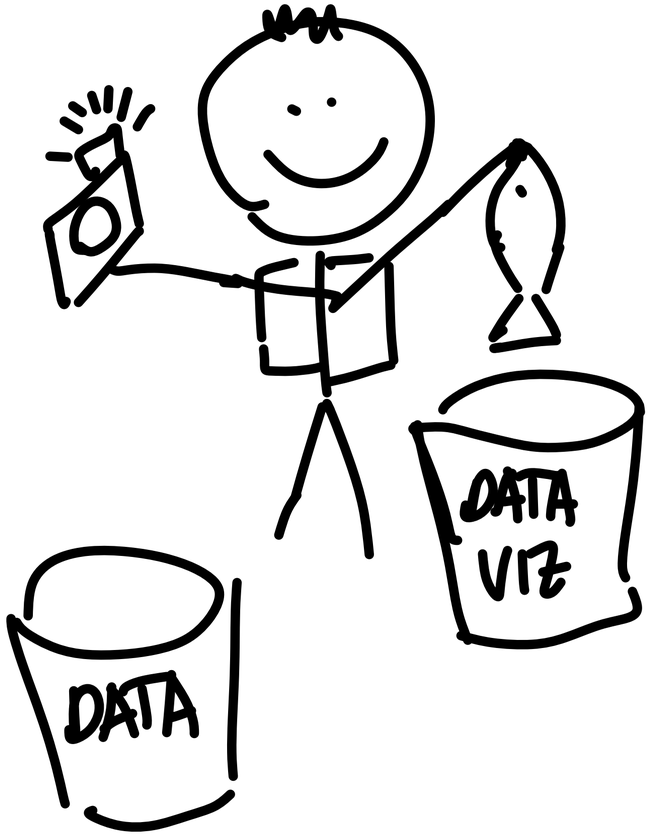
Write tests



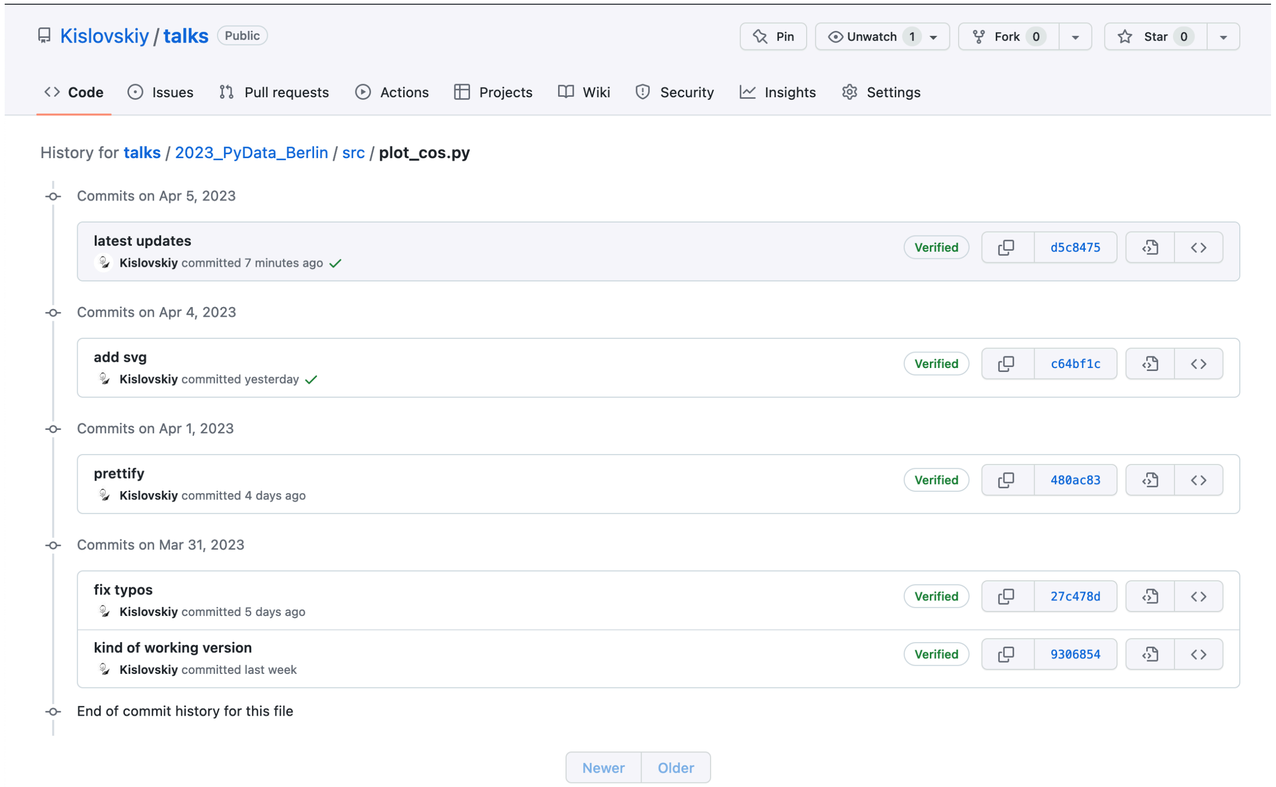
```python
import pandas as pd

def test_csv_values():
    ...
    data = pd.read_csv(...)
    assert abs(data["sin"]).all() <= 1
```

https://github.com/Kislovskiy/talks/tree/2023-pycon-de/2023_PyData_Berlin

Version control

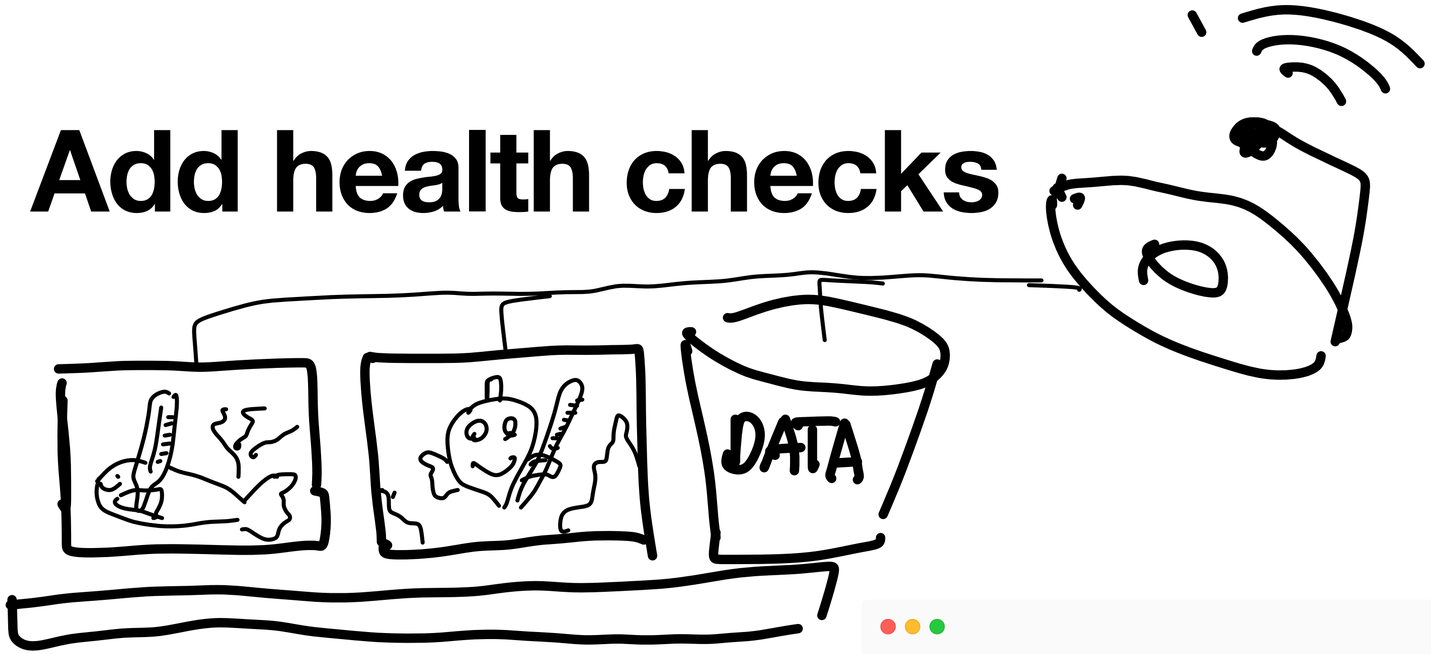


Tip: put .svg to README

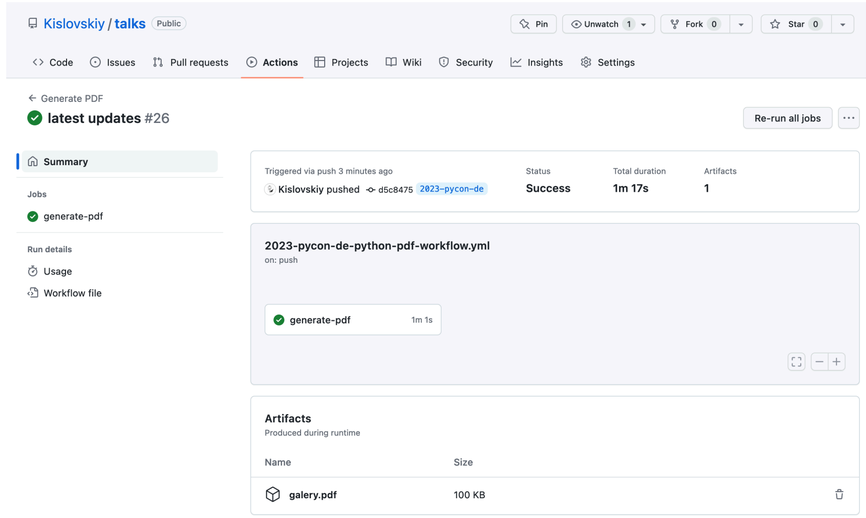


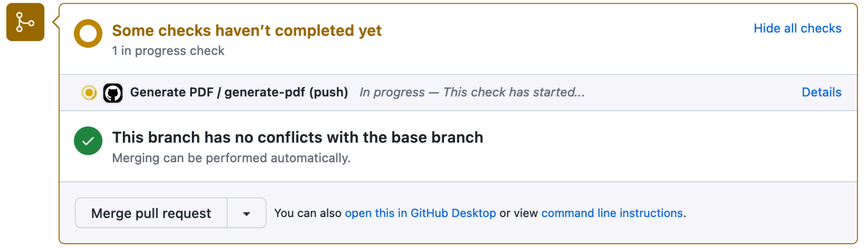
https://github.com/Kislovskiy/talks/tree/2023-pycon-de

Add health checks




```yaml
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: galery.pdf
        path: 2023_PyData_Berlin/results/galery.pdf
```

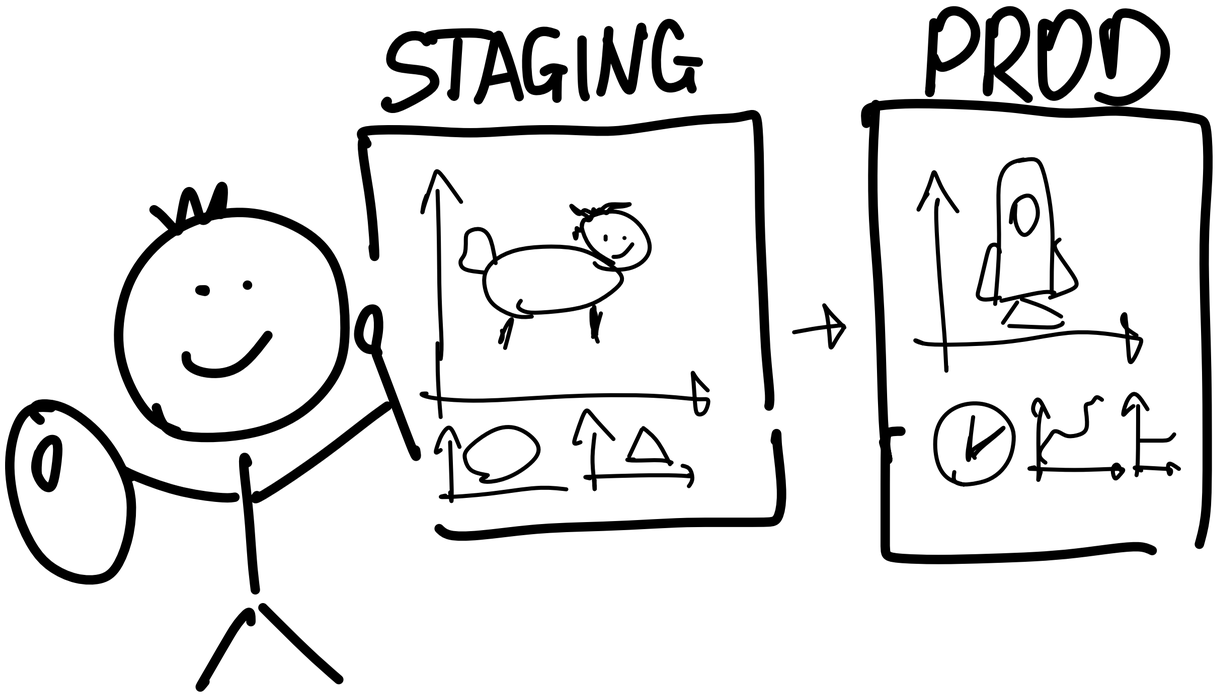




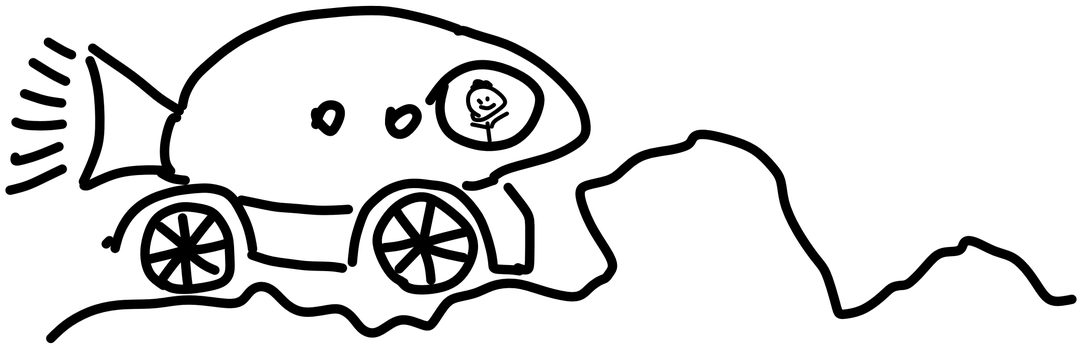
https://github.com/Kislovskiy/talks/tree/2023-pycon-de

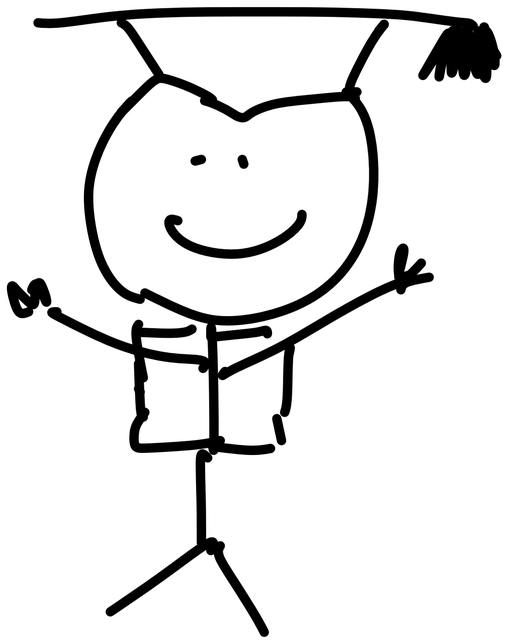
```python
from importlib.util import module_from_spec, spec_from_file_location
from inspect import getmembers, isfunction
from pathlib import Path

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/2023-pycon-de/2023_PyData_Berlin