

A tour of data viz in Python



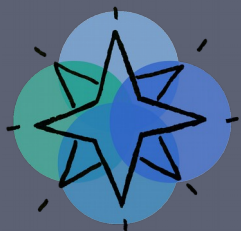
Éléonore Mayola



Developer + Data scientist

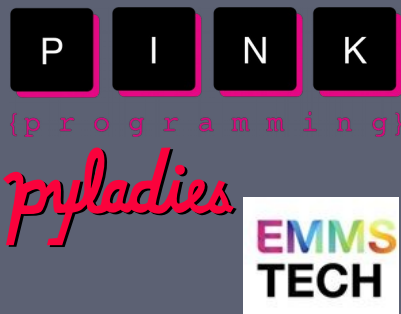
Python, Clojure, JS, HTML/CSS

Freelance work



web + data

Community volunteering



Endeavour

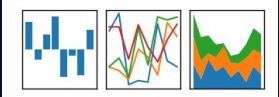
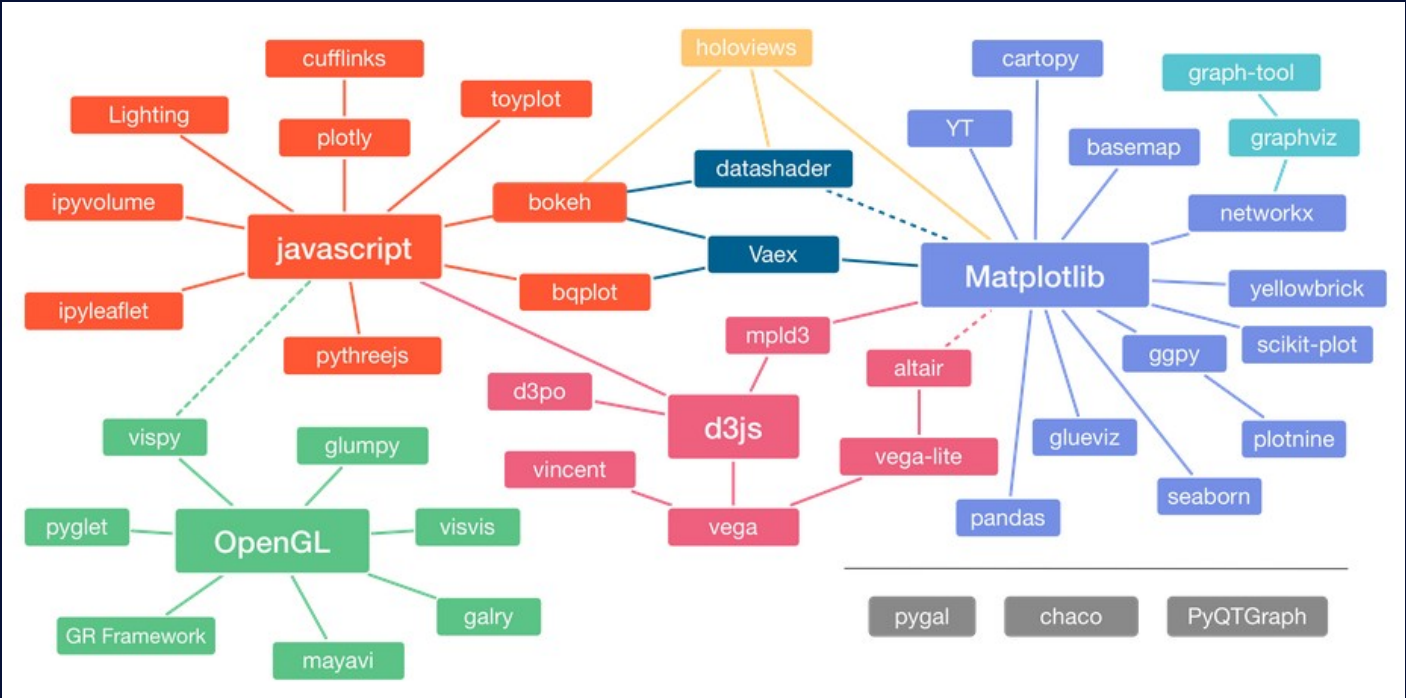


art + tech + data



Different needs for different use cases

- * yourself
- * your colleagues
- * your manager
- * your clients
- * exploring a dataset
- * writing an internal report
- * writing a client report
- * writing a research article

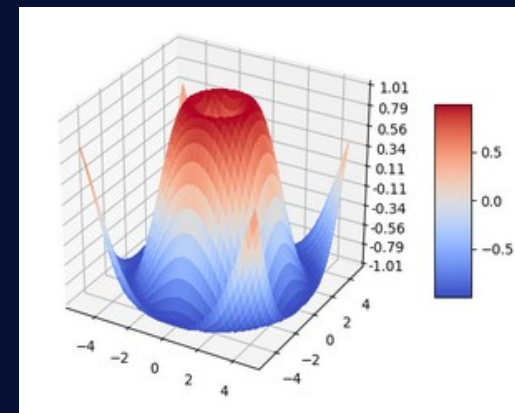
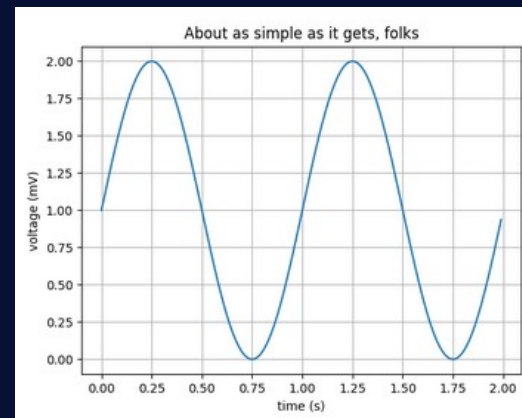


Adaptation of Jake VanderPlas graphic about the Python visualization landscape, by Nicolas P. Rougier
Source: <https://pyviz.org/overviews/index.html>



matplotlib.org

“Matplotlib tries to make
easy things easy and
hard things possible.”





The **pyplot** module provides a MATLAB-like interface

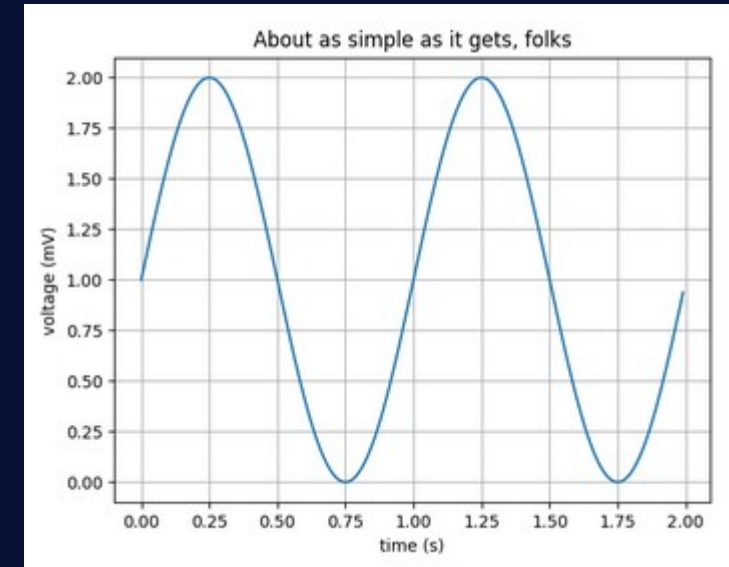
```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

# Data for plotting
t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2 * np.pi * t)

fig, ax = plt.subplots()
ax.plot(t, s)

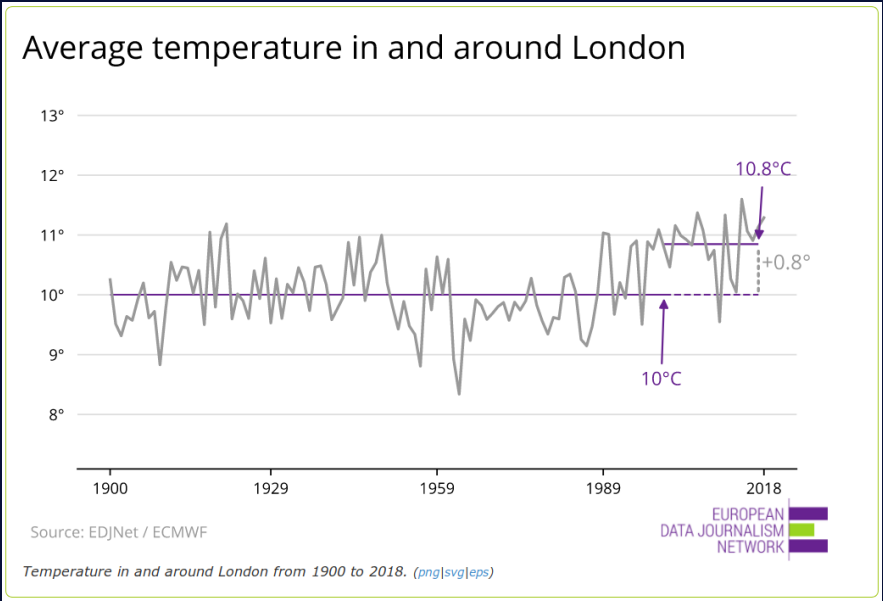
ax.set(xlabel='time (s)', ylabel='voltage (mV)',
       title='About as simple as it gets, folks')
ax.grid()

fig.savefig("test.png")
plt.show()
```





onedegreewarmer.eu



Customisable

Example: add labels, lines and annotations

```
matplotlib.axes.Axes.axhline
ax.axhline(y=10yrs_average, color='#6f2c91',
            linewidth=1.5)

matplotlib.pyplot.annotate
ax.annotate(f'{temperature}°C',
            xy=(3, 1), xycoords='data',
            xytext=(0.8, 0.95),
            arrowprops=dict(facecolor='#6f2c91'),
            horizontalalignment='right',
            verticalalignment='top')
```

Look up the rich gallery of examples: matplotlib.org/gallery/index.html



Best for

Complex or
customised plots

Challenge

Syntax can
become tricky



pandas.pydata.org

“[...] high-performance, easy-to-use data structures and data analysis tools for the Python programming language.”

Uses Matplotlib for plotting

```
import matplotlib.pyplot as plt
```



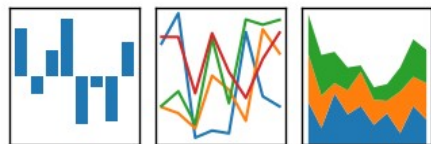
`pandas.DataFrame.plot`

```
DataFrame.plot(x=None, y=None, kind='line', ax=None, subplots=False,  
                sharex=None, sharey=False, layout=None, figsize=None,  
                use_index=True, title=None, grid=None, legend=True, style=None,  
                logx=False, logy=False, loglog=False, xticks=None, yticks=None,  
                xlim=None, ylim=None, rot=None, fontsize=None, colormap=None,  
                table=False, yerr=None, xerr=None, secondary_y=False,  
                sort_columns=False, **kwds)
```

<https://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.plot.html>

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



```
In [1]: %matplotlib inline
```

```
In [2]: import matplotlib
import matplotlib.pyplot as plt
matplotlib.style.use('seaborn')
import numpy as np
import pandas as pd
```

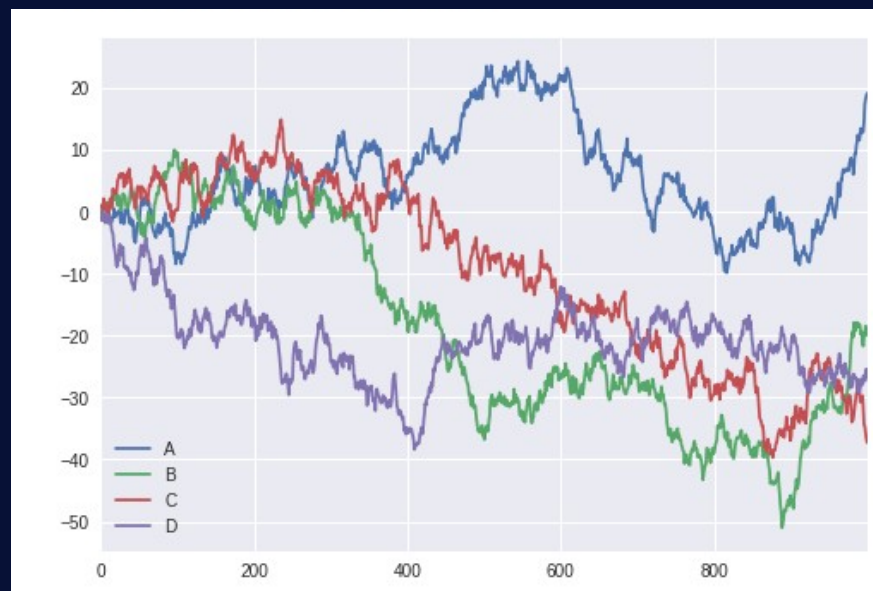
```
In [3]: df = pd.DataFrame(np.random.randn(1000, 4),
                           columns=list('ABCD'))

df = df.cumsum()

plt.figure()

df.plot()
plt.show()
```

<Figure size 576x396 with 0 Axes>





Best for

Simple plots,
plotting during data
analysis

Challenge

Not the most
aesthetic
→ test styles or try
Seaborn



plot.ly

An Open Source Company

Plotly's team maintains the fastest growing open-source visualization libraries for R, Python, and JavaScript.

These libraries seamlessly interface with our enterprise-ready Deployment servers for easy collaboration, code-free editing, and deploying of production-ready dashboards and apps.

DEMO DASH

WELCOME TO THE

Plotly Open Source Graphing Libraries

D3.js and WebGL charts and maps for Python, MATLAB, R, and more.



Plotly Python Open
Source Graphing Library

Star 5,359



Plotly MATLAB Open
Source Graphing Library

Star 209



Plotly R Open Source
Graphing Library

Star 1,528



Plotly JavaScript Open
Source Graphing Library

Star 10,610



Plotly Python Open Source Graphing Library

Plotly's Python graphing library makes interactive, publication-quality graphs. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, polar charts, and bubble charts.

plot.ly/python

plotly.py

High-level, declarative charting library with over 30 chart types, including scientific charts, 3D graphs, statistical charts, SVG maps, financial charts, and more.





plotly.py

github.com/plotly/plotly.py

```
pip install plotly==4.1.0
```

```
import plotly.graph_objects as go

Gapminder =
go.data.gapminder().query("continent=='Oceania'")

fig =
go.Figure(data=go.Scatter(x=Gapminder["year"],
y=Gapminder["lifeExp"], mode='lines',
name='country'))

fig.show()
```

Plotly Express

github.com/plotly/plotly_express

```
pip install plotly_express==0.4.1
```

```
import plotly.express as px

Gapminder =
px.data.gapminder().query("continent=='Oceania'")

fig = px.line(Gapminder, x="year", y="lifeExp",
color='country')

fig.show()
```





Best for

Notebook or
generate html
output

Challenge

Actively maintained
(by a company),
consultancy



bokeh.org

“Bokeh is an **interactive** visualization library for Python that enables beautiful and meaningful visual presentation of data in modern web browsers.”



github.com/bokeh/bokeh



Interactive notebook tutorial

```
In [1]: from bokeh.io import output_notebook, show
        from bokeh.plotting import figure

        # Install bokeh sample data (the first time you run it)
        # import bokeh
        # bokeh.sampledata.download()

        # Glucose sample data requires Pandas to be installed
        from bokeh.sampledata.glucose import data

In [2]: output_notebook()

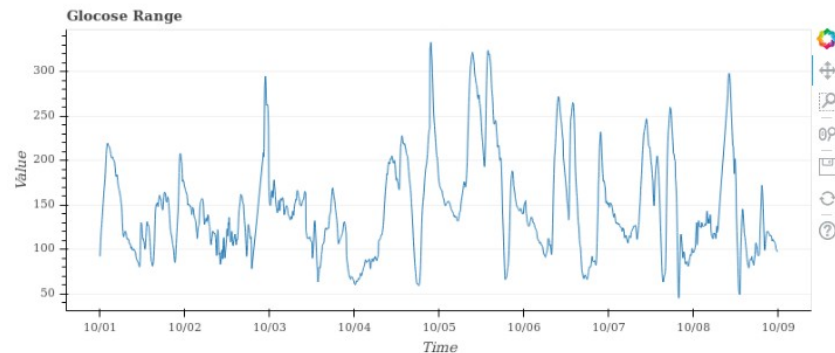
        BokehJS 1.3.4 successfully loaded.

In [3]: # reduce data size to one week
        week = data.loc['2010-10-01':'2010-10-08']

        p = figure(x_axis_type="datetime", title="Glucose Range", plot_height=350, plot_width=800)
        p.xgrid.grid_line_color=None
        p.ygrid.grid_line_alpha=0.5
        p.xaxis.axis_label = 'Time'
        p.yaxis.axis_label = 'Value'

        p.line(week.index, week.glucose)

        show(p)
```





Best for

Interactive plots,
dashboard apps
(Bokeh Server)

Challenge

Creating dashboard
apps can get tricky
(dev skills)



altair-viz.github.io

vega.github.io/vega



“Altair is a declarative statistical visualization library for Python, based on Vega and Vega-Lite.”

“Vega is a visualization grammar, a declarative language for creating, saving, and sharing **interactive** visualization designs.”

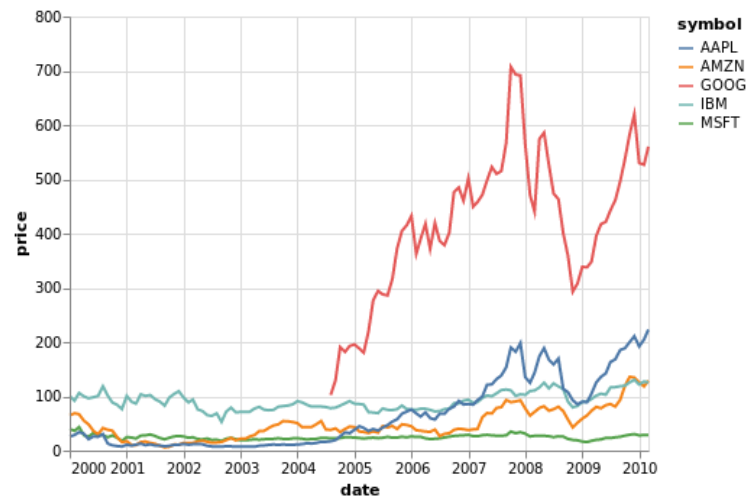


```
In [1]: import altair as alt
alt.renderers.enable('notebook')

from vega_datasets import data
```

```
In [2]: source = data.stocks()

alt.Chart(source).mark_line().encode(
    x='date',
    y='price',
    color='symbol'
)
```



Out[2]:








Best for

Interactive plots &
maps, data
transformation

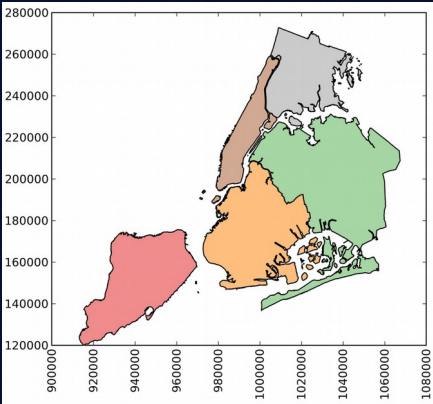
Challenge

One main
maintainer (slower
development)

	Best for	Challenges
	Complex or customised plots	Syntax can get tricky
	Simple plots, plotting during data analysis	Not the most aesthetic → test styles or try Seaborn
	Notebook or generate html output	Actively maintained (by a company), consultancy
	Interactive plots, dashboard apps (Bokeh Server)	Creating dashboard apps can get tricky (dev skills)
	Interactive plots & maps, data transformation	One main maintainer (slower development)

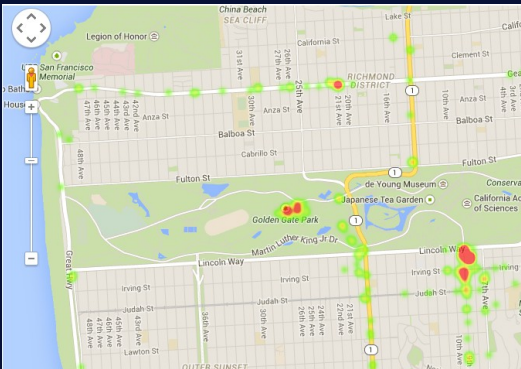
GeoPandas

<https://github.com/geopandas/geopandas>



gmplot

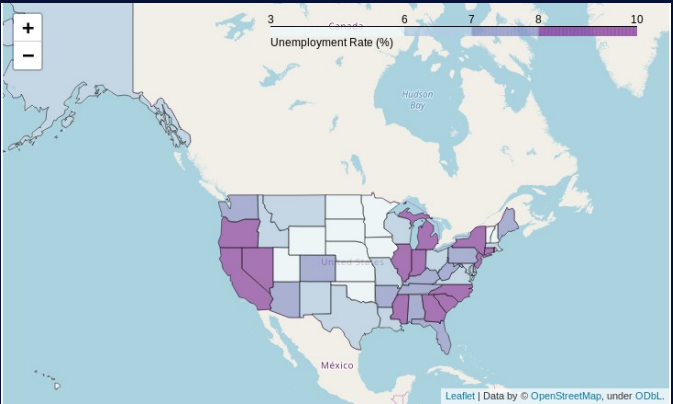
<https://github.com/vgm64/gmplot>



Folium

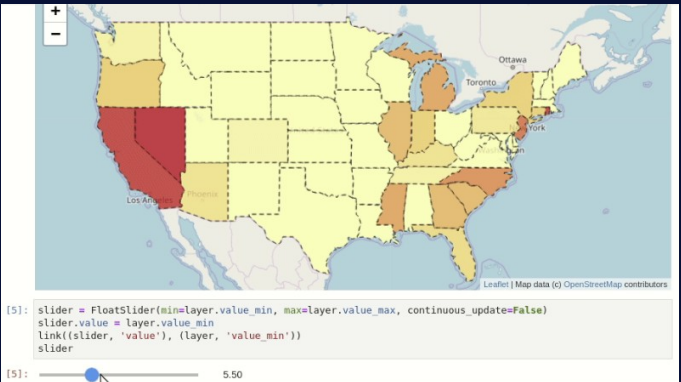


<https://github.com/python-visualization/folium>



ipyleaflet

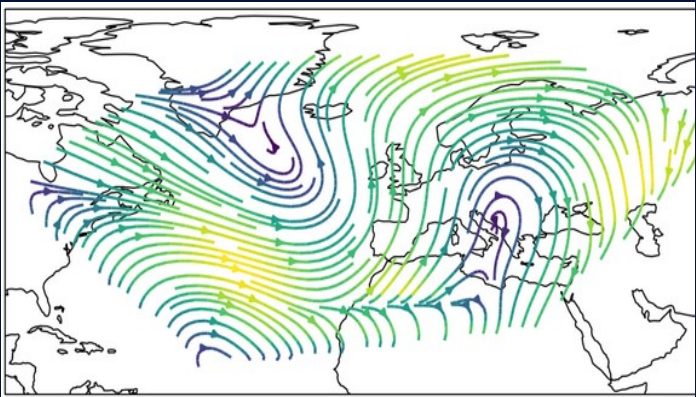
<https://github.com/jupyter-widgets/ipyleaflet>



CartoPy

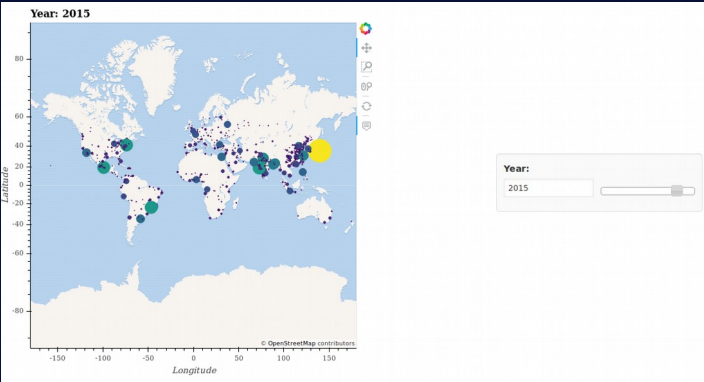


<https://github.com/SciTools/cartopy>



geoviews

<https://github.com/pyviz/geoviews>



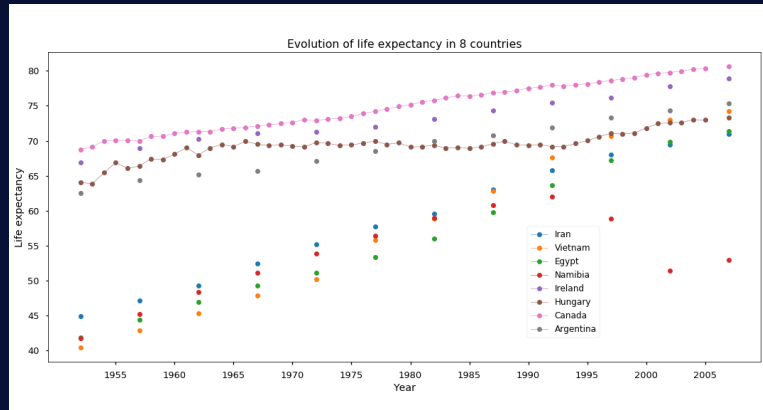
Example project: Explore the evolution of life expectancy throughout the world

- data from the Gapminder dataset
- compare libraries while exploring data
- further example with a notebook dashboard



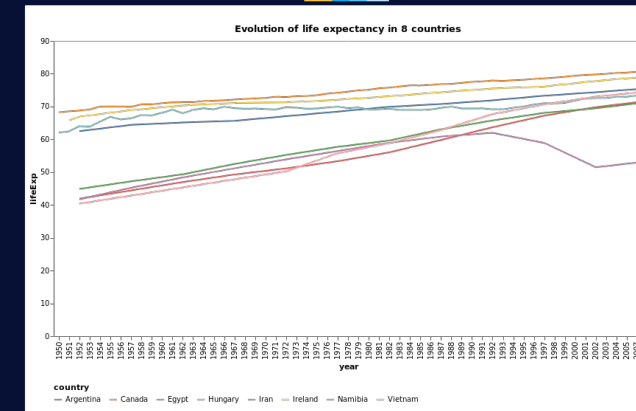
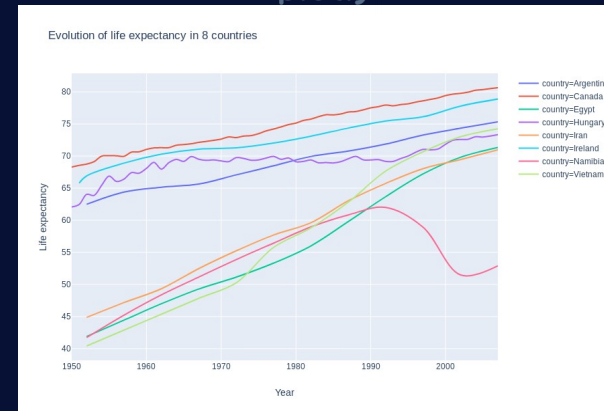
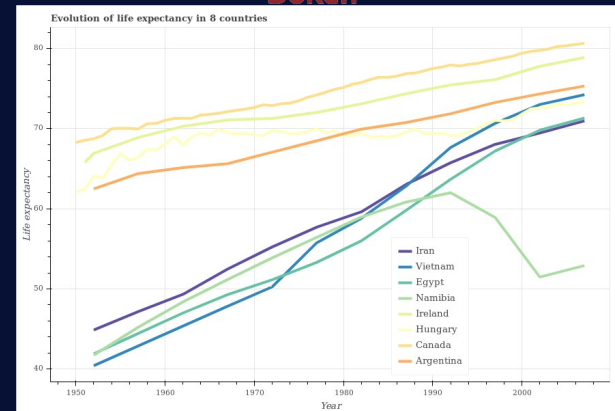
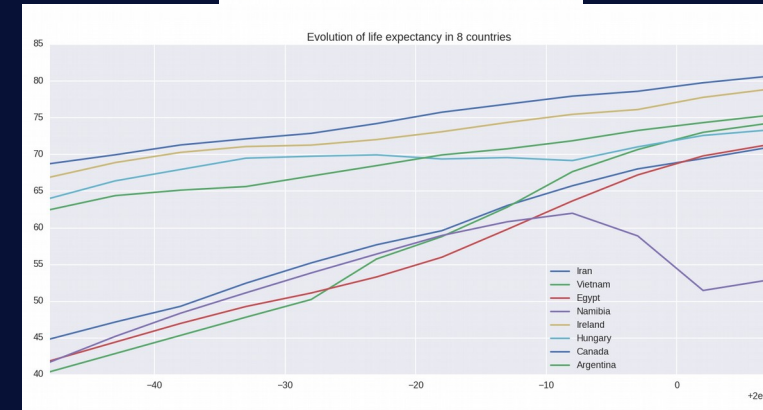
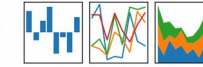
www.gapminder.org

Gapminder is a fact tank, not a think tank. Gapminder fights devastating misconceptions about global development.



pandas

$$y_{it} = \beta'x_{it} + \mu_i + \epsilon_{it}$$





```
jupyter try_matplotlib Last Checkpoint: 18 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [3]: import csv

country = []
continent = []
year = []
life_exp = []

countries_to_filter = ["Iran", "Vietnam", "Egypt", "Namibia",
                      "Ireland", "Hungary", "Canada", "Argentina"]

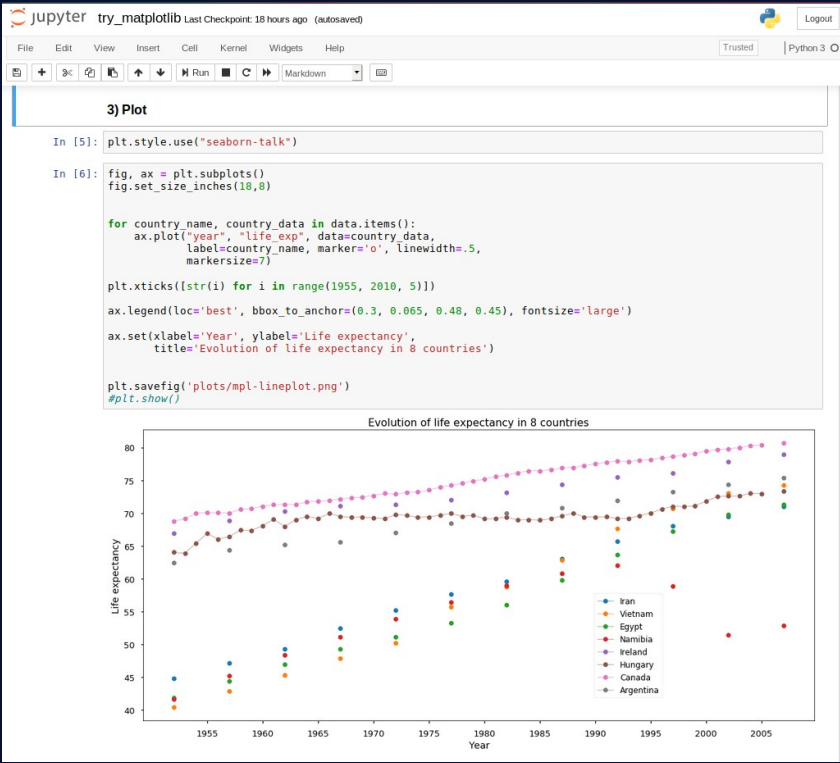
with open('data/gapminder.csv', 'r') as csvfile:
    plots = csv.DictReader(csvfile, delimiter=',')
    for row in plots:
        if row["country"] in countries_to_filter:
            country.append(row["country"])
            continent.append(row["continent"])
            year.append(row["year"])
            life_exp.append(row["lifeExp"])

life_exp = [float(l) for l in life_exp]

2) Reshape data

In [4]: data = {}
all_years = [str(i) for i in range(1952, 2008)]

for c in countries_to_filter:
    indices = [i for i, x in enumerate(country) if x == c]
    years = year[indices[0]:indices[-1]+1]
    life_exp = [life_exp[indices.index(yr)] if yr in years else np.nan for yr in all_years]
    data[c] = {"year": all_years, "life_exp": life_exp}
```





```
jupyter try_pandas Last Checkpoint: 9 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Exploring a Gapminder dataset

1) Read CSV data

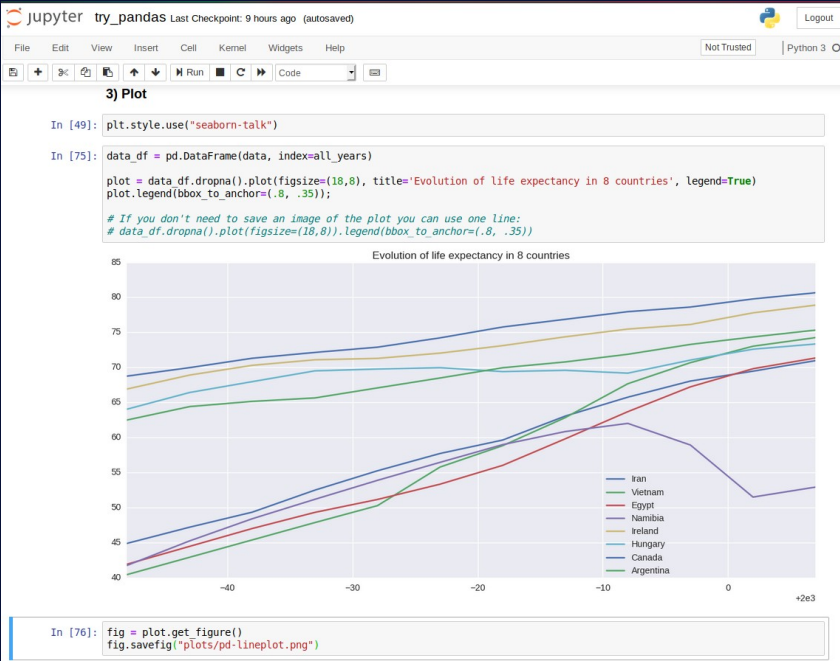
In [3]: gapminder_df = pd.read_csv('data/gapminder.csv')

2) Reshape data

In [4]: countries_to_filter = ["Iran", "Vietnam", "Egypt", "Namibia",
                              "Ireland", "Hungary", "Canada", "Argentina"]

In [74]: data = {}

all_years = range(1952, 2008)
for c in countries_to_filter:
    data[c] = []
    df = gapminder_df[gapminder_df.country == c]
    for yr in all_years:
        if yr not in list(df.year):
            data[c].append(np.NaN)
        else:
            year_idx = df[df.year==yr].index[0]
            life_exp = df.loc[year_idx, "lifeExp"]
            data[c].append(life_exp)
```





```
jupyter try_bokeh Last Checkpoint: 26 minutes ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
+ - - - - - Run Code
Exploring a Gapminder dataset

1) Read CSV data

In [3]: import pandas as pd

In [4]: gapminder_df = pd.read_csv('data/gapminder.csv')

2) Plot

In [5]: countries_to_filter = ["Iran", "Vietnam", "Egypt", "Namibia",
                               "Ireland", "Hungary", "Canada", "Argentina"]

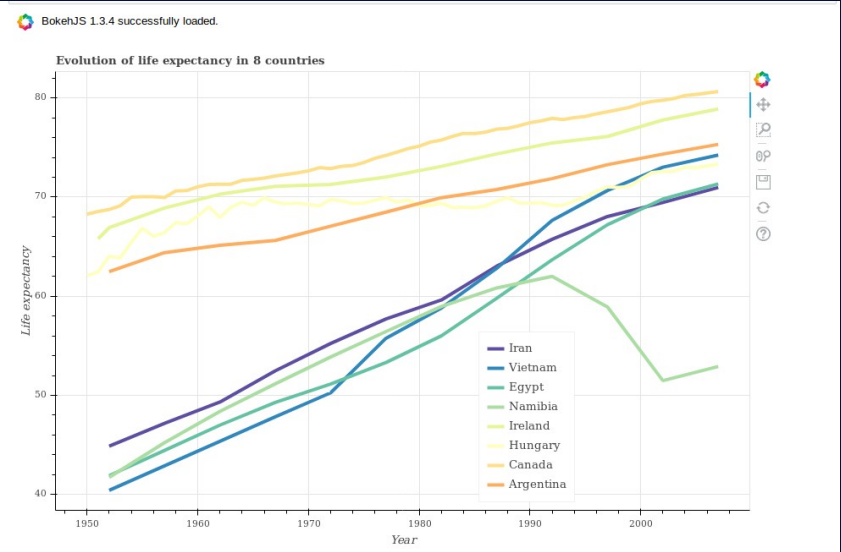
In [7]: p = figure(title = "Evolution of life expectancy in 8 countries", plot_width=900, plot_height=600)
p.xaxis.axis_label = 'Year'
p.yaxis.axis_label = 'Life expectancy'
plot_legend = []

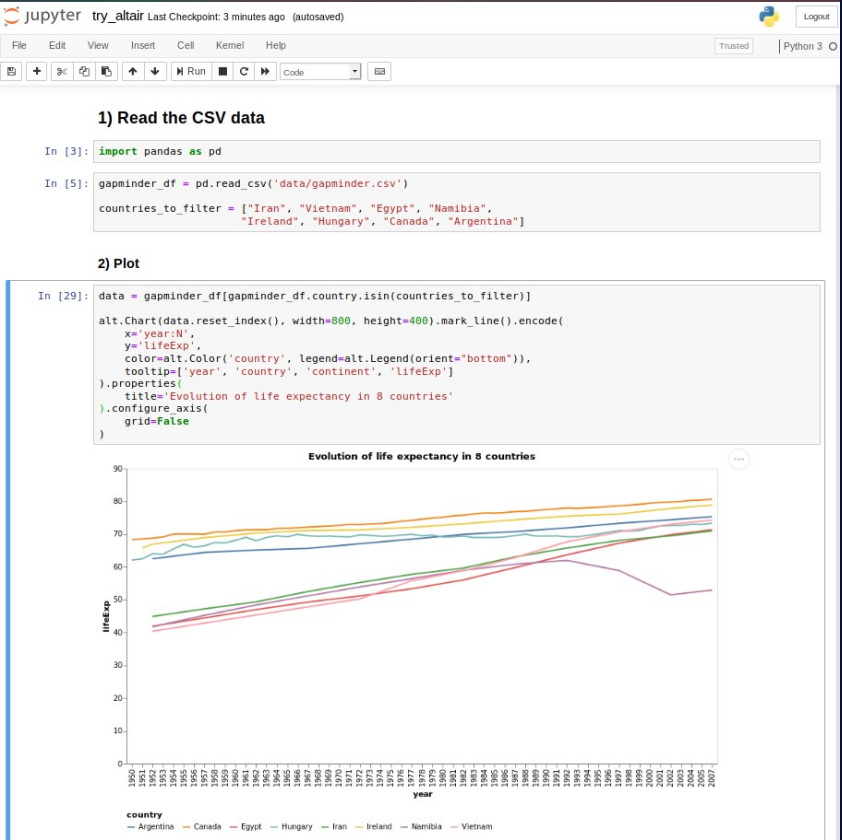
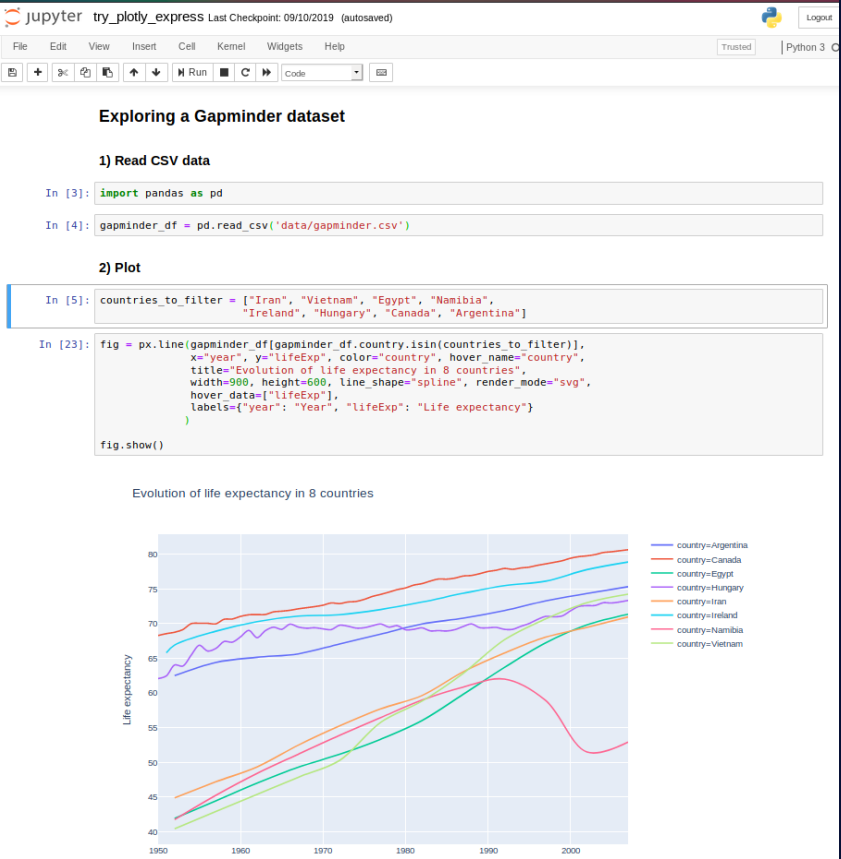
for c in countries_to_filter:
    line_plot = p.line(gapminder_df[gapminder_df.country == c][["year",
                                                                "lifeExp"]],
                       line_color=Spectral1[countries_to_filter.index(c)],
                       line_width=4)
    plot_legend.append((c, [line_plot]))

legend = Legend(items=plot_legend, location=(500, 10))
p.add_layout(legend)
p.legend.click_policy="hide"

output_notebook()
show(p)

BokehJS 1.3.4 successfully loaded.
```



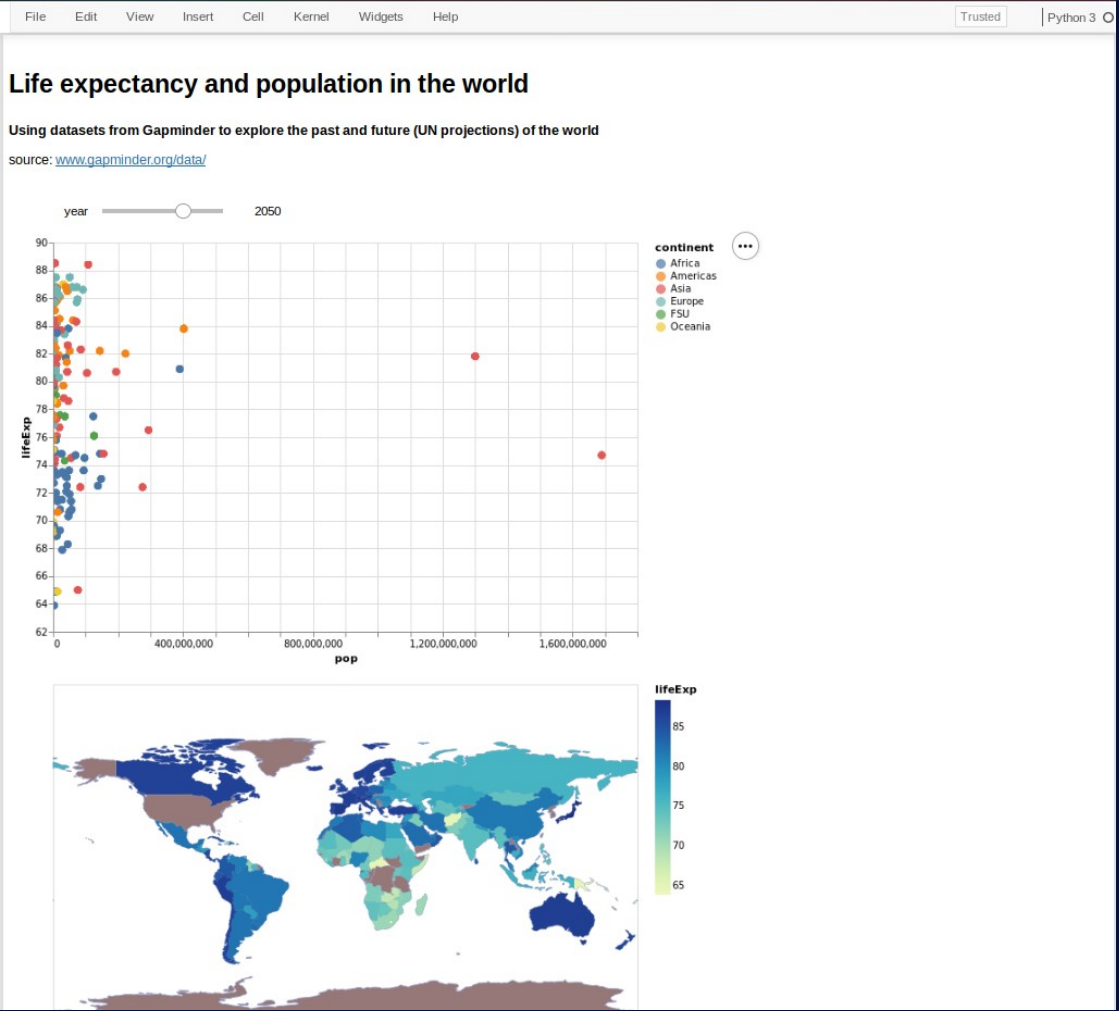


Example project: Explore the evolution of life expectancy throughout the world



Jupyter notebook
+ Altair plot and map
+ ipywidgets





Data exploration | Development

Jupyter notebooks



Production

Generate plot images



Dashboard applications



Voilà dashboards



Thank you



github.com/Eleonore9/tour_dataviz_python

* Exhaustive list of Python tools for data viz: pyviz.org/tools.html

* Libraries mentioned:

matplotlib.org

pandas.pydata.org

plot.ly

bokeh.org

altair-viz.github.io

matplotlib



plotly



Bokeh