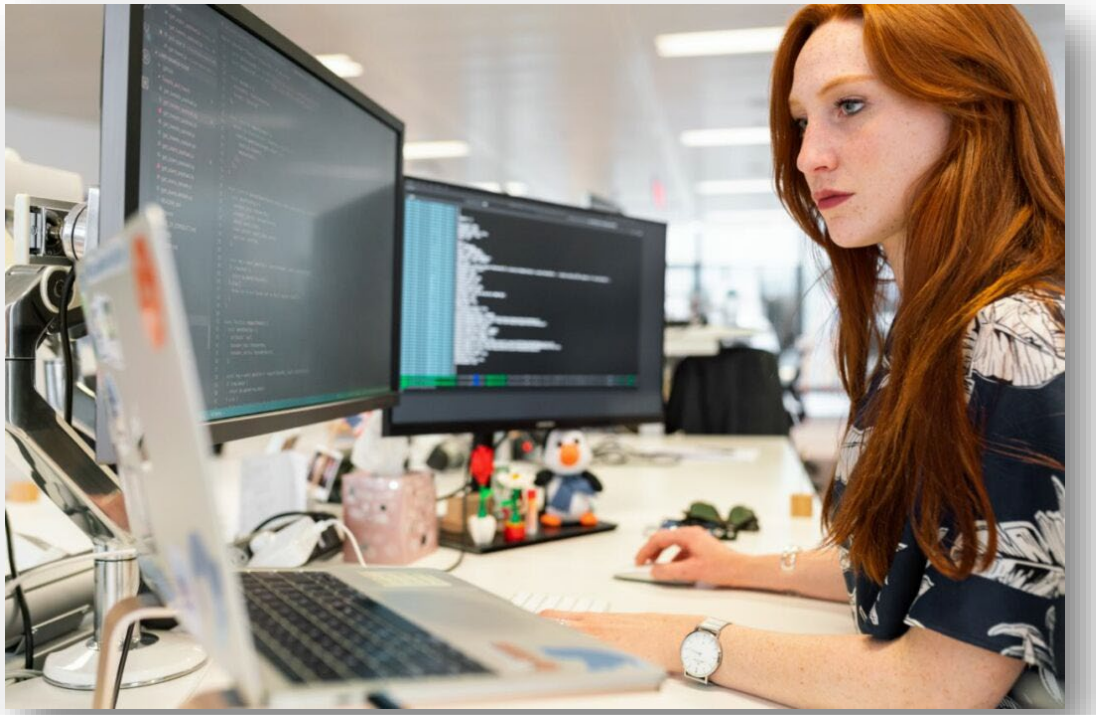
Źródła danych dla inżyniera

Narzędzia programistyczne



jupyter

INSTALL PROJECT DOCUMENTATION BLOG DONATE

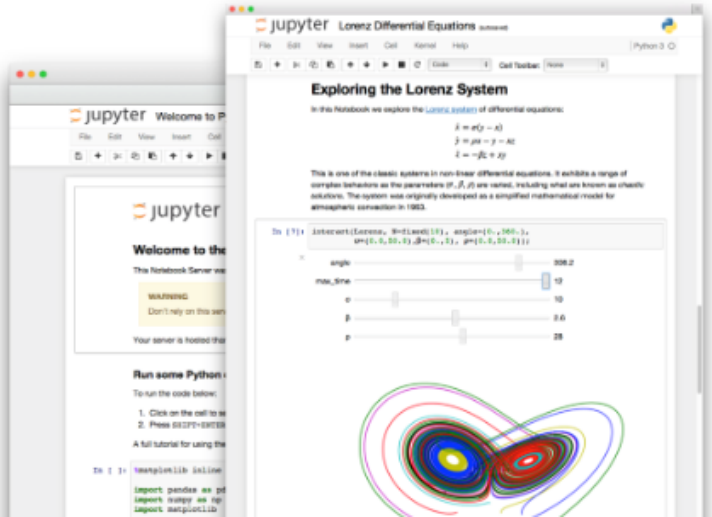
ANACONDA.

https://www.anaconda.com/

https://colab.research.google.com/

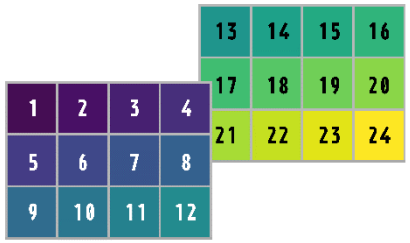
Jupyter Notebook

The Jupyter Notebook is a web-based interactive computing platform that allows users to author data- and code-driven narratives that combine live code, equations, narrative text, visualizations, interactive dashboards and other media.

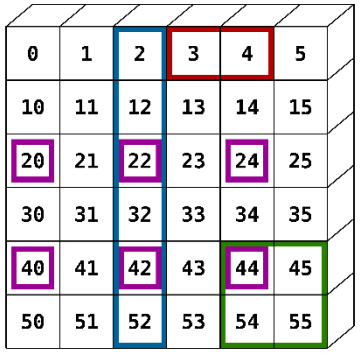


https://www.jetbrains.com/dataspell/

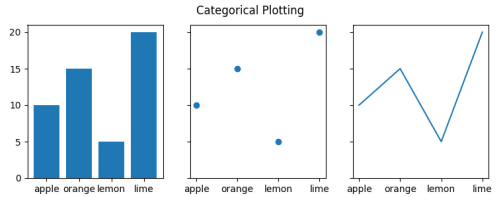
NumPy

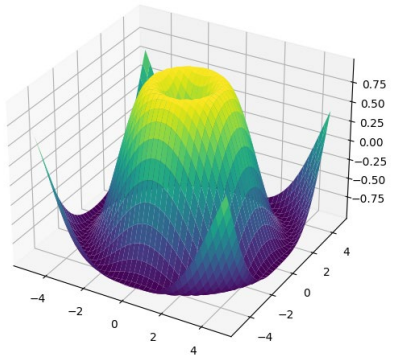


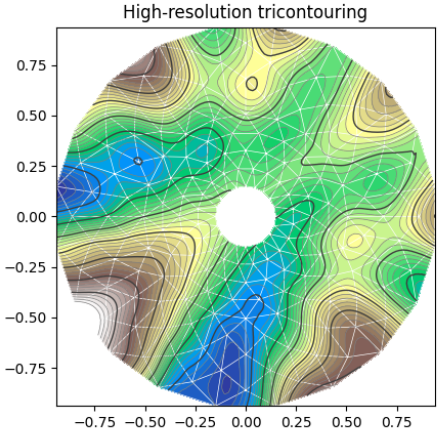

```
>>> a[0, 3:5]
array([3, 4])

>>> a[4:, 4:]
array([[44, 55],
       [54, 55]])

>>> a[:, 2]
a([2, 12, 22, 32, 42, 52])

>>> a[2::2, ::2]
array([[20, 22, 24],
       [40, 42, 44]])
```

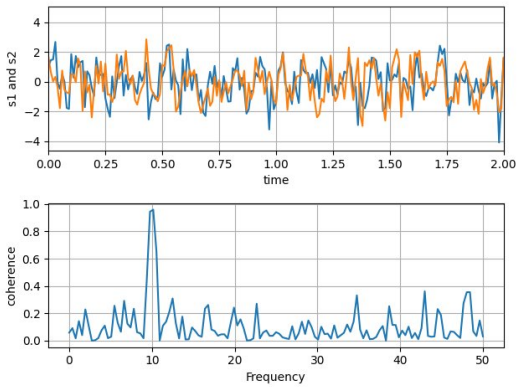


matplotlib



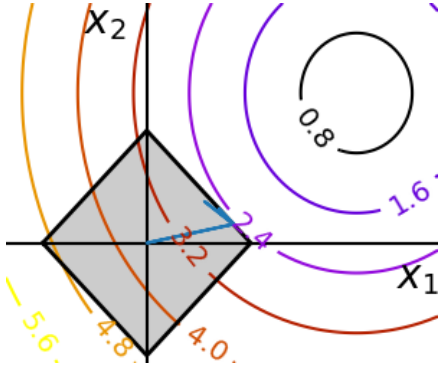


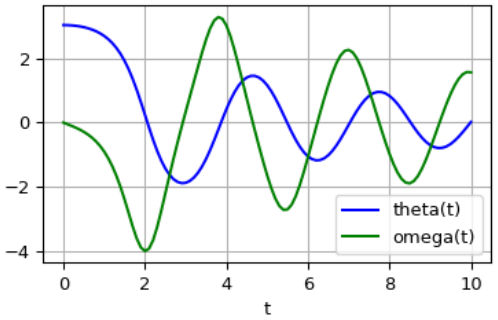


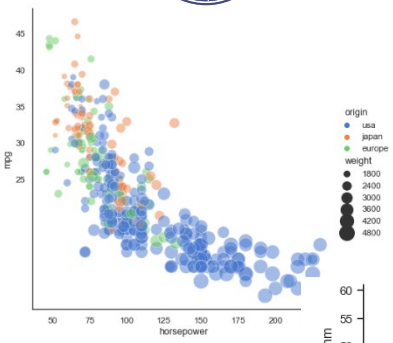


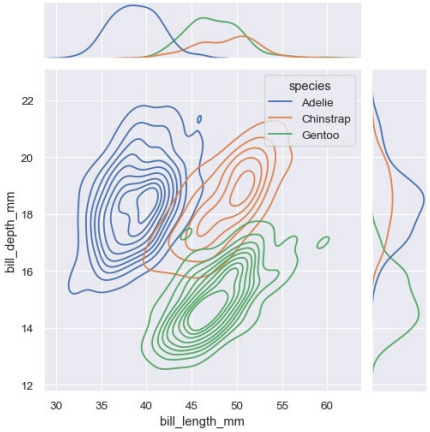
SciPy.org

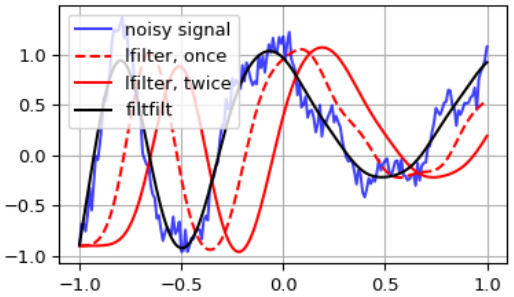
seaborn

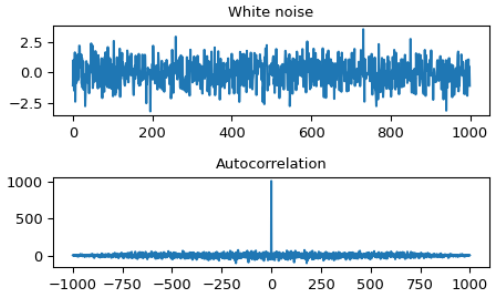


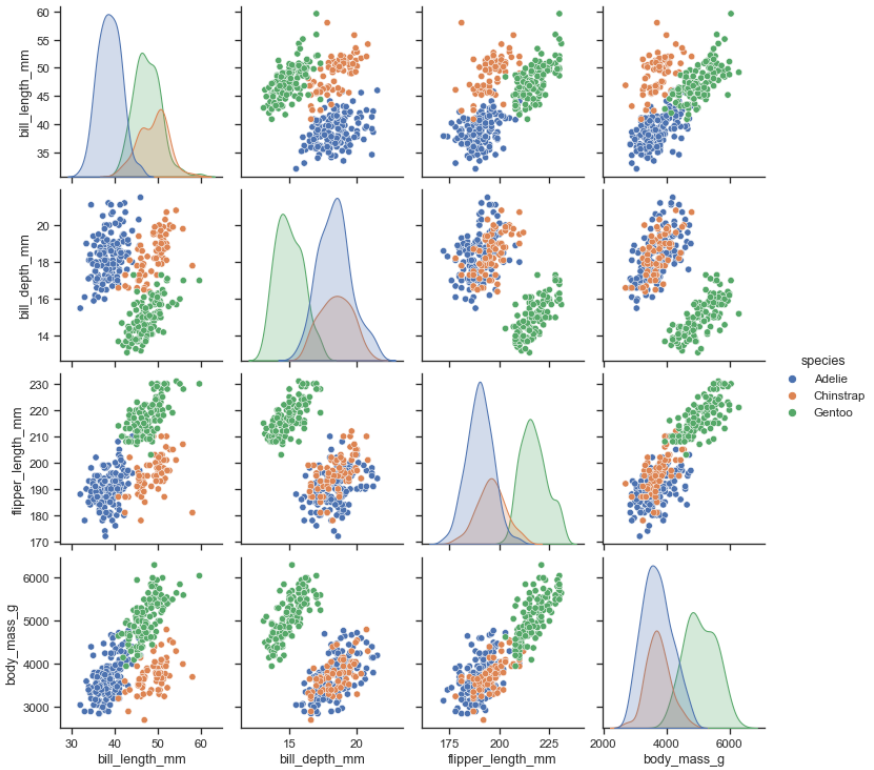










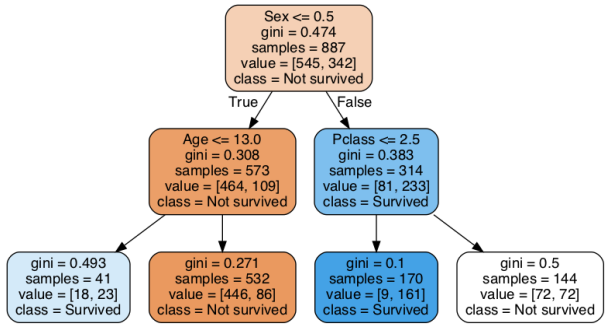


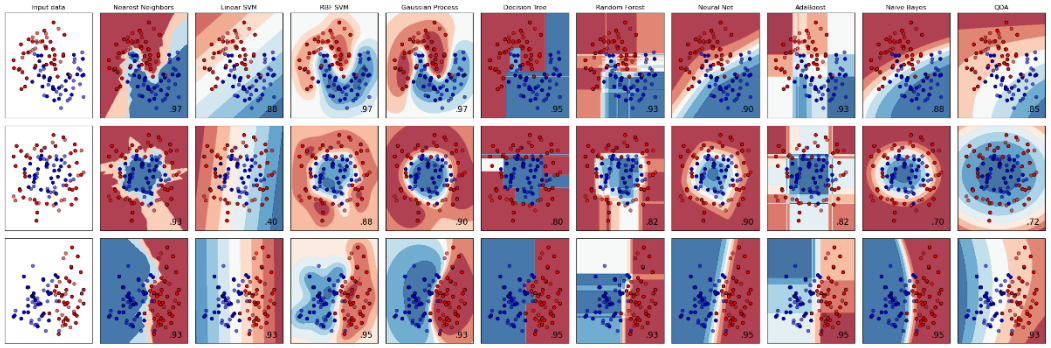
[6]:

date	total_cases	new_cases	total_deaths	new_deaths
2020-03-04	1.0	1.0	0.0	0.0
2020-03-07	5.0	4.0	0.0	0.0
2020-03-08	6.0	1.0	0.0	0.0
2020-03-09	11.0	5.0	0.0	0.0
2020-03-10	17.0	6.0	0.0	0.0
2020-03-11	22.0	5.0		
2020-03-12	31.0	9.0		
2020-03-13	49.0	18.0		
2020-03-14	68.0	19.0		
2020-03-15	104.0	36.0		
2020-03-16	125.0	21.0		
2020-03-17	177.0	52.0		
2020-03-18	238.0	61.0		
2020-03-19	287.0	49.0		
2020-03-20	355.0	68.0		
2020-03-21	425.0	70.0		

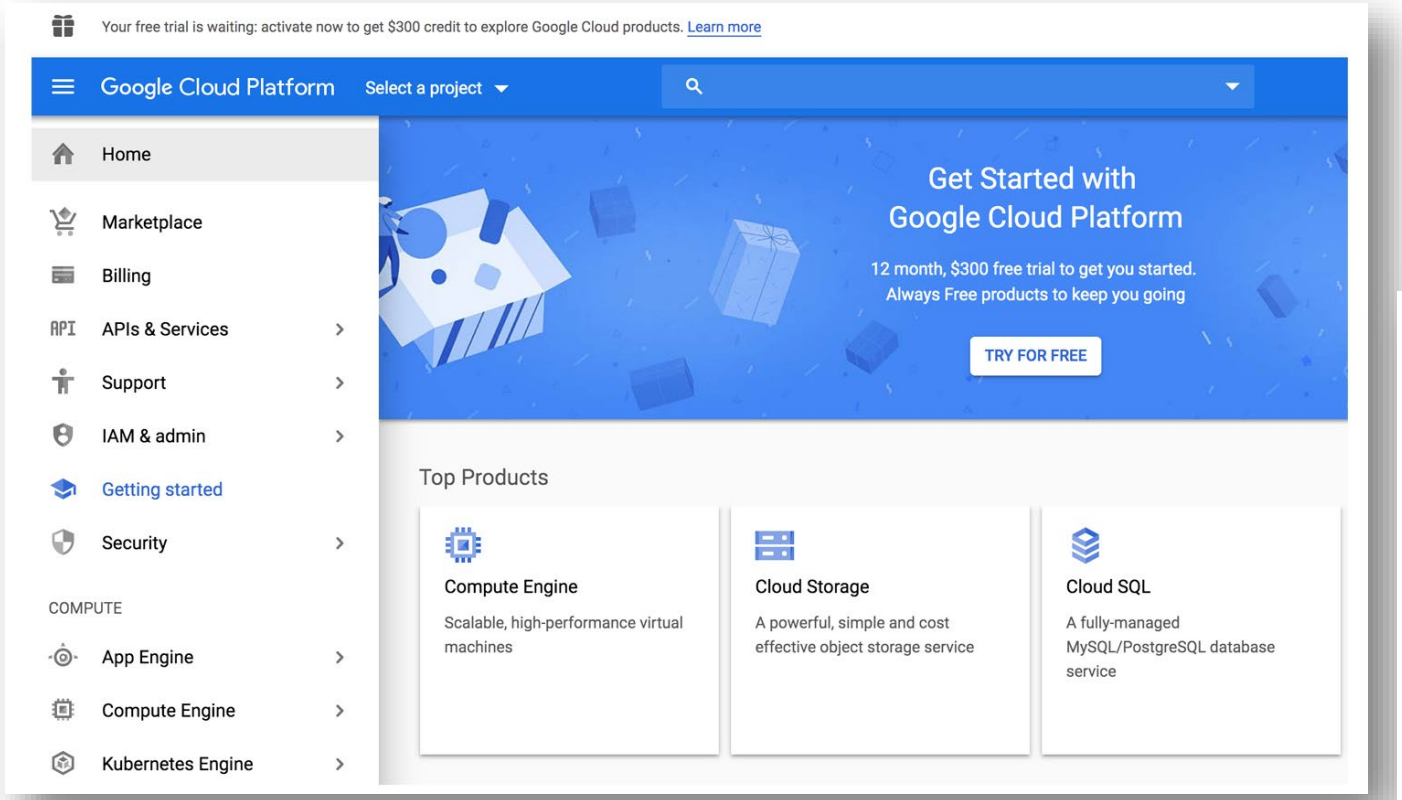
Rat 1 - day 08\10.08.2020\

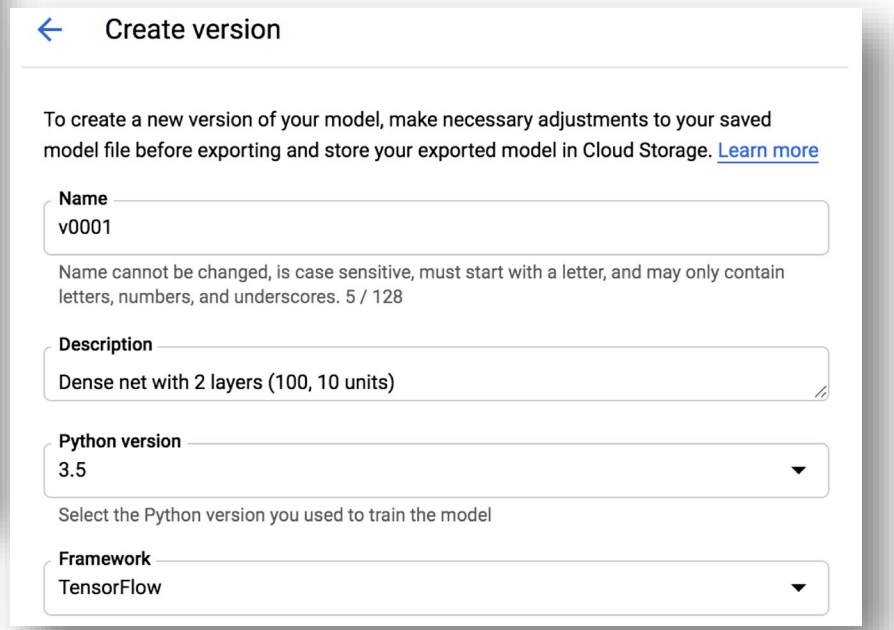
	Left stimulus	Right stimulus	Click	Choice	Payout	Next decision	Next click
0	triangle.png	circleIN.png	Left	triangle.png	Lose	Stay	Stay
1	triangle.png	circleIN.png	Left	triangle.png	Lose	Stay	Shift
2	circleIN.png	triangle.png	Right	triangle.png	Lose	Shift	Stay
3	triangle.png	circleIN.png	Right	circleIN.png	Win	Stay	Shift
4	circleIN.png	triangle.png	Left	circleIN.png	Win	Stay	Stay
5	circleIN.png	triangle.png	Left	circleIN.png	Win	Stay	Shift
6	triangle.png	circleIN.png	Right	circleIN.png	Win	Stay	Shift
7	circleIN.png	triangle.png	Left	circleIN.png	Win	Stay	Stay
8	circleIN.png	triangle.png	Left	circleIN.png	Win	Stay	Stay
9	circleIN.png	triangle.png	Left	circleIN.png	Win	Shift	Stay
10	triangle.png	circleIN.png	Left	triangle.png	Lose	Shift	Stay
11	circleIN.png	triangle.png	Left	circleIN.png	Win	Shift	Stay
12	triangle.png	circleIN.png	Left	triangle.png	Lose	Stay	Shift
13	circleIN.png	triangle.png	Right	triangle.png	Lose	Shift	Shift
14	circleIN.png	triangle.png	Left	circleIN.png	Win	Shift	Stay
15	triangle.png	circleIN.png	Left	triangle.png	Lose	Stay	Shift



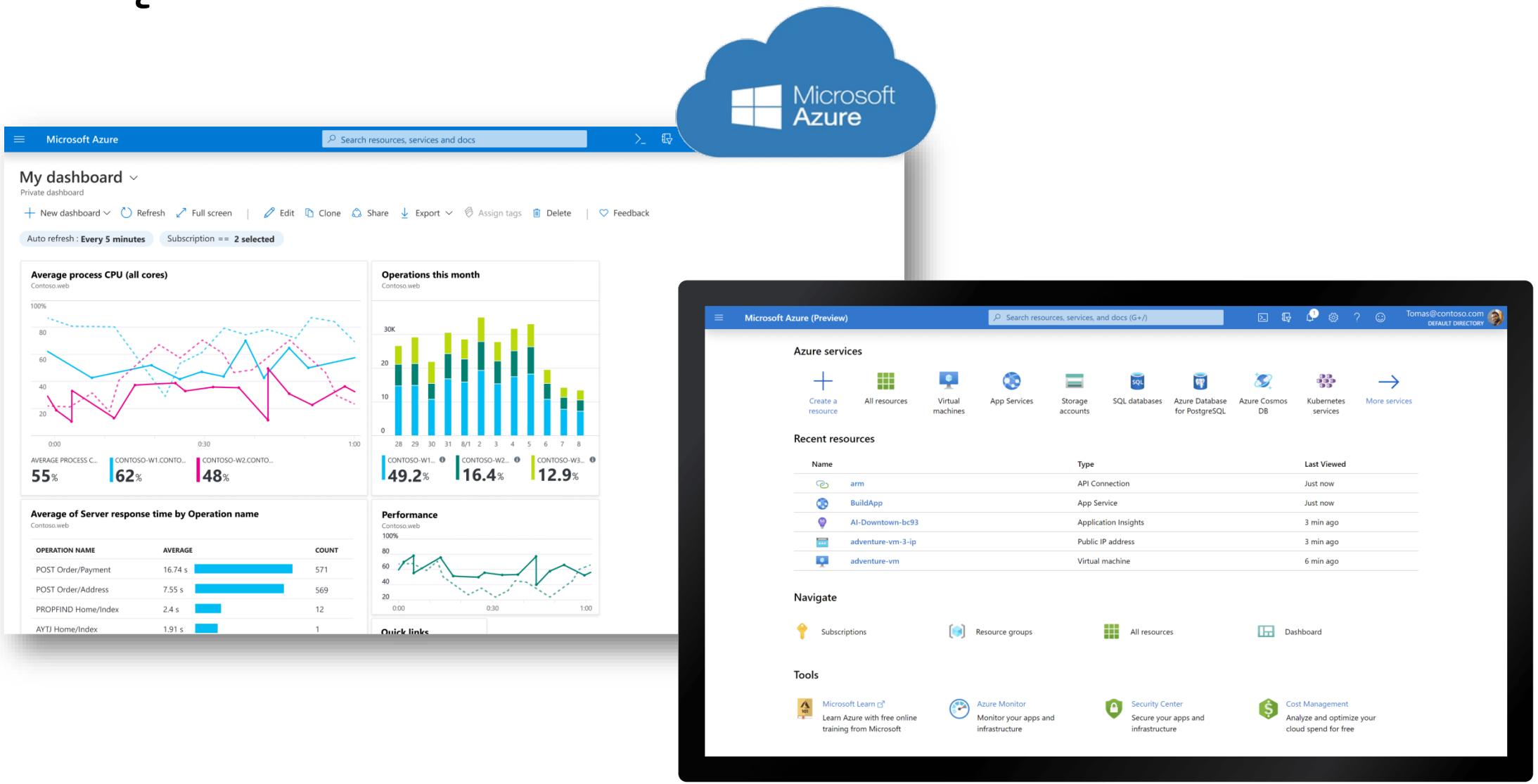


Narzędzia chmurowe





Narzędzia chmurowe

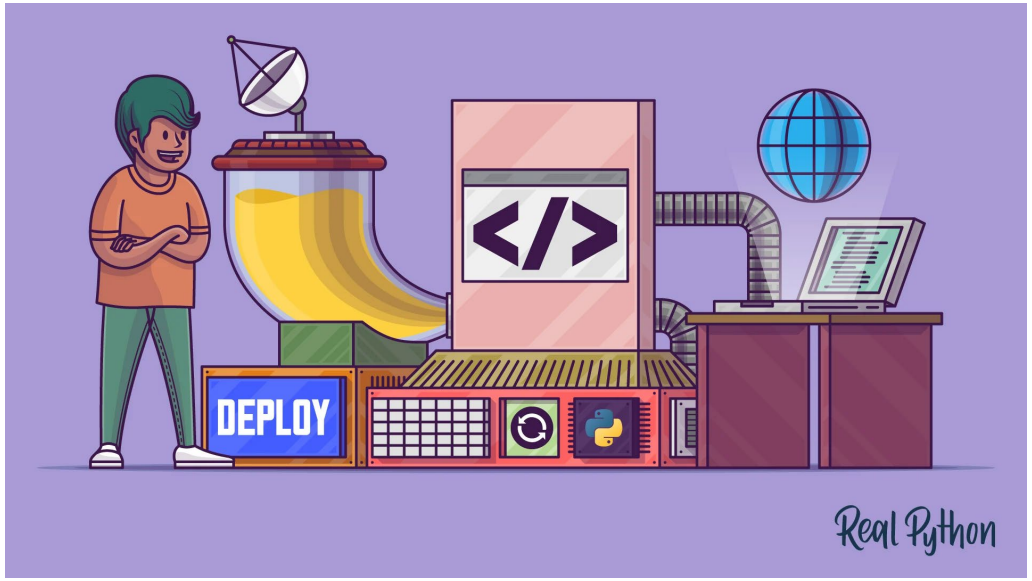


Narzędzia chmurowe

amazon web services™ | EC2

Udostępnianie własnych rozwiązań

https://flask.palletsprojects.com/en/2.0.x/



https://realpython.com/tutorials/flask/

Zbiory danych

Pliki własnej produkcji

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
```

```
import urllib.request
import os
```

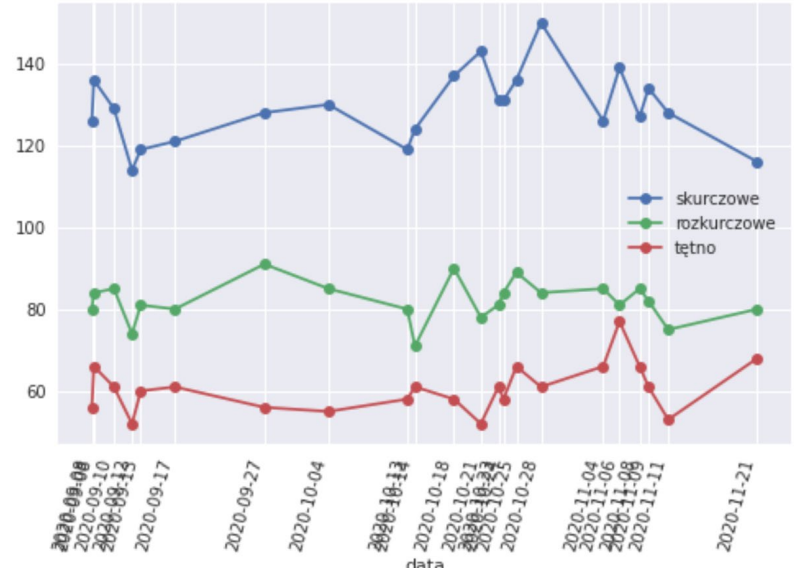
```
filename = 'cisnienie.xlsx'
url = "https://byes.pl/wp-content/uploads/datasets/" + filename
if not os.path.isfile(filename):
    print(f'Downloading {url} ...')
    urllib.request.urlretrieve(url, filename)
```

```
df_raw = pd.read_excel(filename, index_col=0)
df_raw.head()
```

```
df = df_raw.set_index(pd.to_datetime(df_raw.index, format='%d_%m_%Y %H_%M'))
df.head()
```

data	skurczowe	rozkurczowe	tętno
2020-09-08 12:20:00	126	80	56
2020-09-08 17:42:00	136	84	66
2020-09-10 23:30:00	129	85	61
2020-09-12 23:57:00	114	74	52
2020-09-13 23:29:00	119	81	60

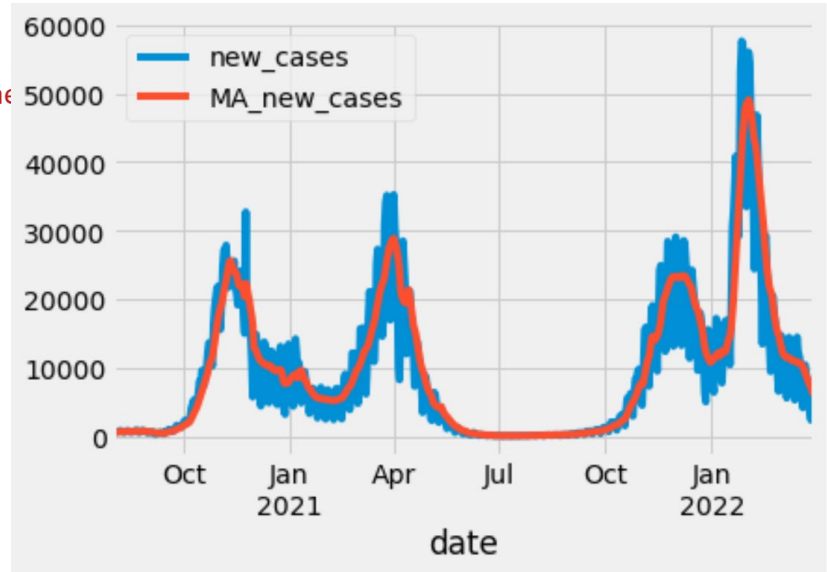
```
with plt.style.context('seaborn'):
    df.plot(marker='o', xticks=df.index, rot=75)
plt.savefig('obr.png', dpi=300, bbox_inches = 'tight', pad_inches = 0.1)
```

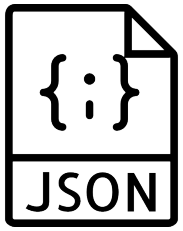


Pliki czyjejś produkcji


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime

filename = 'owid-covid-data.csv'
url = 'https://covid.ourworldindata.org/data/' + filename
if not os.path.isfile(filename):
    print(f'Downloading {url} ...')
    urllib.request.urlretrieve(url, filename)
```

```
df_raw = pd.read_csv(filename, index_col=3)
df_raw.index = pd.DatetimeIndex(df_raw.index)
df_raw.head()

features = ['location', 'total_cases', 'new_cases', 'total_deaths', 'new_deaths', 'ne
df_features = df_raw[features]

df_Pol = df_features[df_features['location'] == 'Poland'].copy()
df_Ger = df_features[df_features['location'] == 'Germany'].copy()

feature = 'new_cases'
df_Pol['MA_'+feature] = df_Pol[feature].rolling(window=7).mean()
df_Ger['MA_'+feature] = df_Ger[feature].rolling(window=7).mean()

with plt.style.context('fivethirtyeight'):
    df_Pol[[feature, 'MA_'+feature]]['2020-08-01':].plot()
```






```python
import json

dane = {"one": [1],
        "two": [1,2],
        "three": [1,2,3]}
plik = open("dane.json",'w')
dane_json = json.dump(dane, plik)

plik = open("dane.json",'r')
dane_z_pliku = json.load(plik)
```


```python
with open("plik.txt",'r', encoding="utf-8") as plik:
    for linia in plik:
        print(linia.strip())
```


```python
filename = 'cardiac_patients.xlsx'
URL = "https://raw.githubusercontent.com/jdrapala/datasets/main/" + filename

df = pd.read_excel(URL)
df['BMI'] = df['BMI'].astype(int)
```

	Height	Weight	BMI	Cholesterol LDL	Cholesterol HDL	Gender	Age
306	170	115	39	145	25	Male	70
371	179	108	33	54	29	Male	64
383	160	59	23	183	38	Male	68
611	194	82	21	76	62	Male	63
543	170	99	34	62	30	Male	58
135	156	77	31	99	56	Female	69
644	185	80	23	68	36	Male	68

Ekstrakcja danych ze stron internetowych

```
import requests
from bs4 import BeautifulSoup

URL = "https://pl.wikipedia.org/wiki/Dane_statystyczne_o_miastach_w_Polsce"
resp = requests.get(URL)
if resp.status_code == 200:
    soup = BeautifulSoup(resp.content,'html.parser')
    tab = soup.find('table',{'class':"wikitable"})
df = pd.read_html(str(tab))[0]
df.head()
```

	Miasto	Województwo	Powierzchnia	Ludność	Gęstość zaludnienia
849	Warszawa	mazowieckie	51724	1790658	3462
327	Kraków	małopolskie	32685	779115	2384
411	Łódź	łódzkie	29325	679941	2319
893	Wrocław	dolnośląskie	29282	642869	2195
601	Poznań	wielkopolskie	26191	534813	2042
165	Gdańsk	pomorskie	26196	470907	1798
770	Szczecin	zachodniopomorskie	30060	401907	1337
72	Bydgoszcz	kujawsko-pomorskie	17598	348190	1979
378	Lublin	lubelskie	14747	339784	2304

```
tabela = df.drop(["Powiat"], axis=1)
tabela = tabela.rename(columns={"Powierzchnia [ha] (01.01.2020)": "Powierzchnia",
                                "Liczba ludności (01.01.2020)": "Ludność",
                                "Gęstość zaludnienia [osoby/km²] (01.01.2020)": "Gęstość zaludnienia"})
tabela
```

```
tabela.sort_values(by='Ludność', ascending=False).head(20)
```

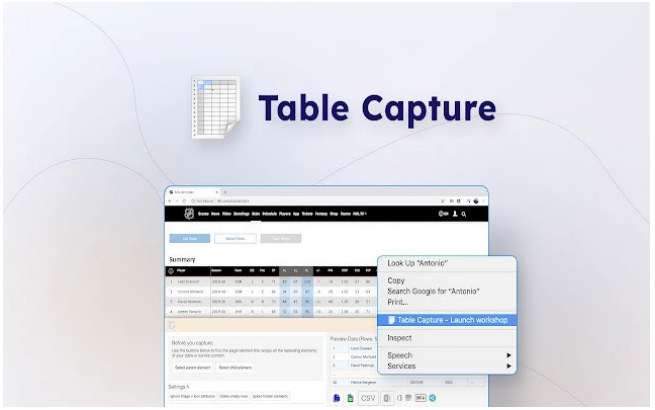
Web Scraping With BeautifulSoup

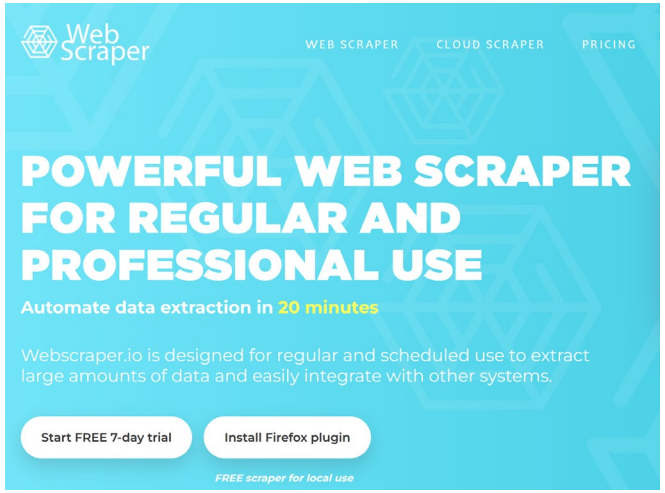
Ekstrakcja danych ze stron internetowych

How to Use Selenium to Web-Scrape with Example

Scraping NBA Player Names and Salaries from Hoopshype.com Using Selenium

Bryan Pfalzgraf · Follow
Published in Towards Data Science · 5 min read · Apr 3, 2020






```python
import sqlite3

# jeżeli baza nie istnieje, zostanie utworzona
conn = sqlite3.connect('mydata.sqlite')

query = """
CREATE TABLE test(city VARCHAR(20), area REAL, population INTEGER);
"""
conn.execute(query)
# zatwierdza transakcję
conn.commit()

data = [('Wrocław', 326.5, 800000),
        ('Kraków', 372.1, 920000),
        ('Sieniawa', 25, 4000)]

mquery = "INSERT INTO test VALUES(?, ?, ?)"

conn.executemany(mquery, data)
conn.commit()
conn.close()
```

```python
conn = sqlite3.connect('mydata.sqlite')

query = "SELECT * FROM test"
cursor = conn.execute(query)

rows = cursor.fetchall()
print(rows)
```

```python
import pandas as pd
pd.DataFrame(rows, columns=[elem[0] for elem in cursor.description])
```

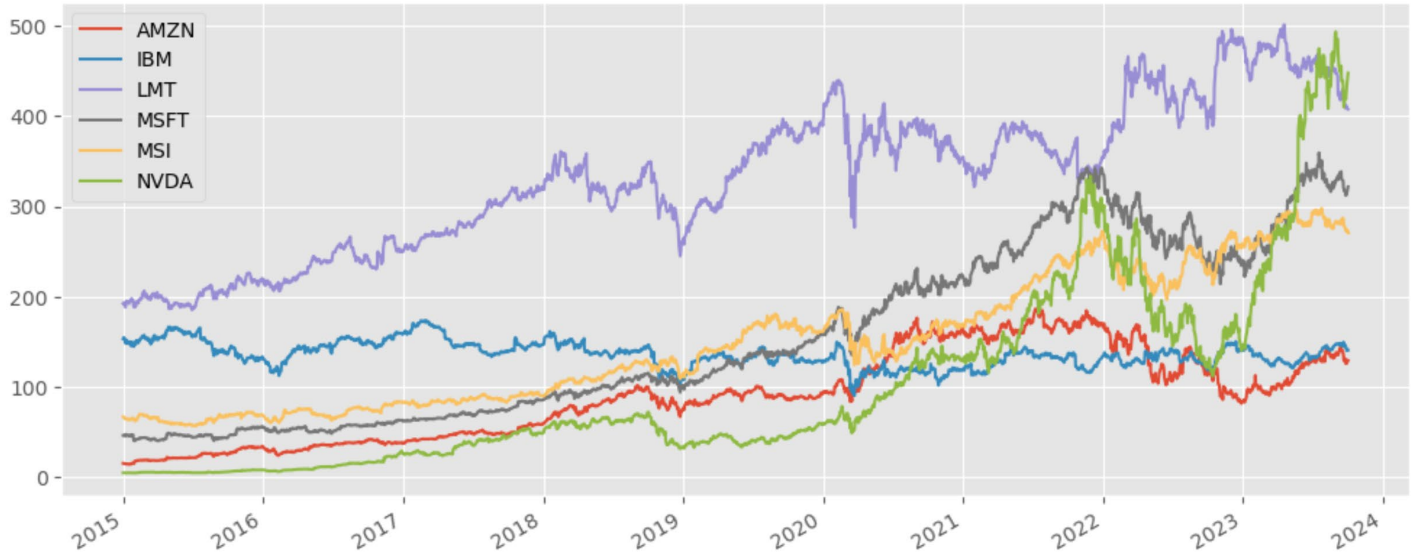
```python
import sqlalchemy as sqla

database = sqla.create_engine('sqlite:///mydata.sqlite')

query = "SELECT * FROM test"
pd.read_sql(query, database)
```

Google Trends

```
from pytrends.request import TrendReq

pytrend = TrendReq()


keyword = 'Geralt'
pytrend.build_payload(kw_list=[keyword])


df = pytrend.interest_by_region()
df.reset_index().plot(x='geoName', y=keyword, figsize=(120, 10), kind ='bar')


df = pytrend.trending_searches(pn='poland')
df.head()


df = pytrend.today_searches(pn='PL')
df
```


```python
from datetime import datetime as dt

import pytz   # strefy czasowe
tz = pytz.timezone("Europe/Warsaw")

import yfinance as yf

df = yf.download(
    tickers=['AMZN', 'MSFT', 'IBM', 'LMT','NVDA', 'MSI'],
    start=tz.localize(dt(2015,1,1)),
    end=tz.localize(dt.today()),
)

df['Close']
```

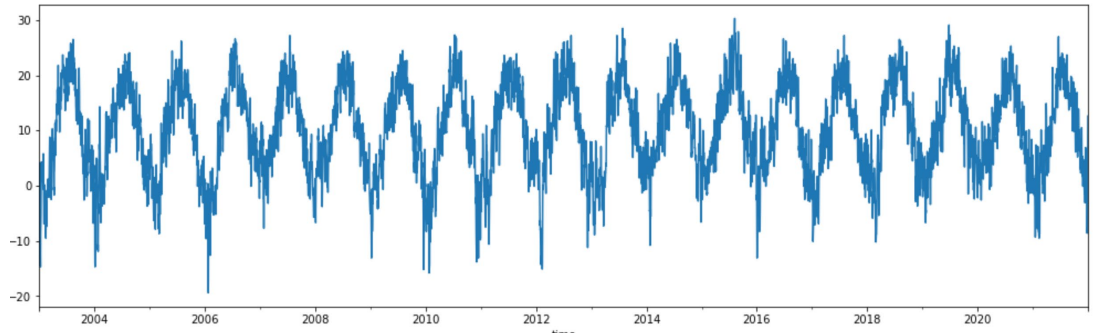
Date	AMZN	IBM	LMT	MSFT	MSI	NVDA
2014-12-31	15.517500	153.384323	192.570007	46.450001	67.080002	5.012500
2015-01-02	15.426000	154.933075	193.309998	46.759998	66.510002	5.032500
2015-01-05	15.109500	152.495224	189.289993	46.330002	65.059998	4.947500
2015-01-06	14.764500	149.206497	188.399994	45.650002	64.510002	4.797500
2015-01-07	14.921000	148.231354	190.830002	46.230000	64.430000	4.785000
...	...	...	...	...	...	...
2023-09-27	125.980003	143.169998	408.720001	312.790009	272.980011	424.679993
2023-09-28	125.980003	141.580002	410.959991	313.640015	274.190002	430.890015
2023-09-29	127.120003	140.300003	408.959991	315.750000	272.239990	434.989990
2023-10-02	129.460007	140.800003	407.820007	321.799988	272.920013	447.820007



meteostat

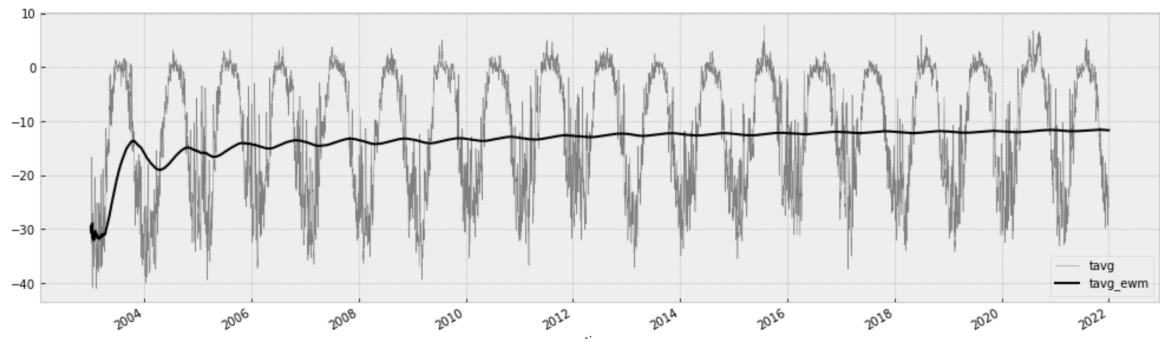
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

from datetime import datetime

import meteostat
```

```
start, end = datetime(2003, 1, 1), datetime(2021, 12, 31)

stacje = meteostat.Stations()
wroclaw = stacje.nearby(51,17)
wroclaw = wroclaw.fetch(1)

pomiary = meteostat.Daily(wroclaw, start, end)
pomiary = pomiary.fetch()
pomiary['tavg'].plot(figsize=(17,5))
```



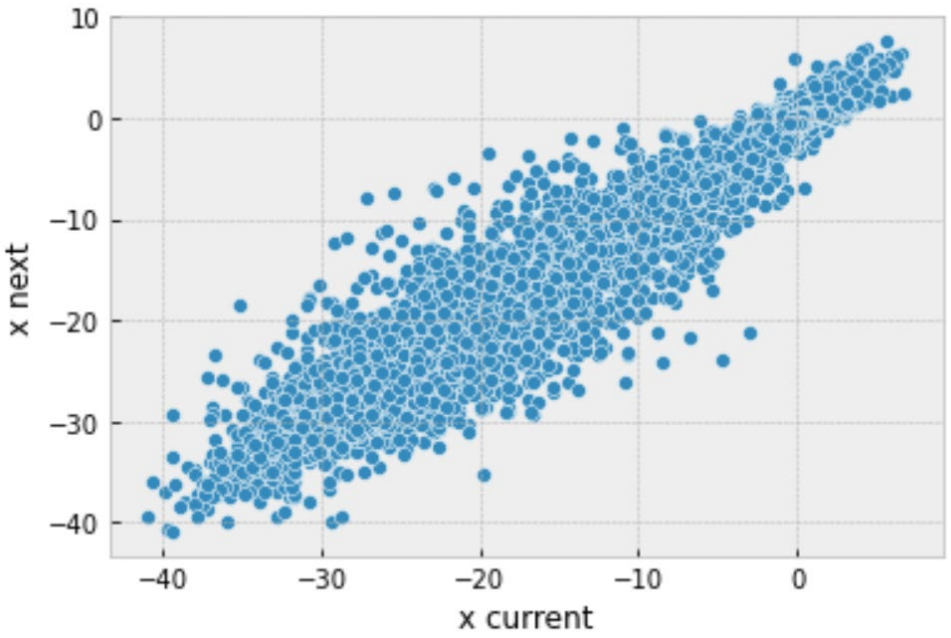

```
df = pomiary[['tavg']]
df.insert(0, 'tavg_interp', df['tavg'].interpolate())
df.insert(0, 'tavg_ewm', df['tavg'].ewm(alpha = .000055).mean())
```

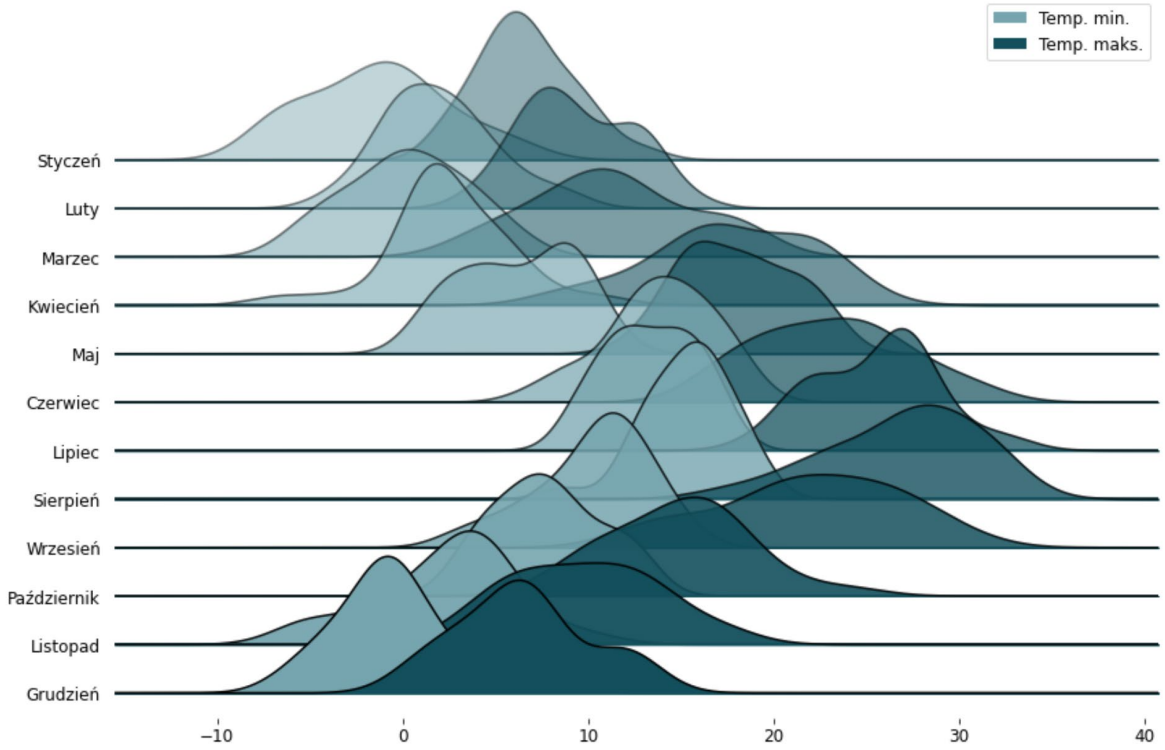
```
with plt.style.context('bmh'):
    fig, ax = plt.subplots(figsize=(17,5))
    df['tavg'].plot(ax=ax, color='gray', linewidth=0.5)
    df['tavg_ewm'].plot(ax=ax, color='black')
    plt.legend()
    plt.show()
```



meteostat

```
df_pom = pd.DataFrame({'x current': df['tavg'].values[:-1], 'x next': df['tavg'].values[1:]})
with plt.style.context('bmh'):
    sns.scatterplot(data=df_pom, x="x current", y="x next")
    plt.show()
```





Independent Statistics & Analysis

U.S. Energy Information Administration

```
!pip install EIA-python
```

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import networkx as nx
import eia
```

```
klucz="6ddcschoeringwielguscec9"
API = eia.API(klucz)
```

```
API.data_by_series("EMISS.CO2-TOTV-EC-NG-AR.A")
```

```
SERIES_ID_list = ["INTL.57-1-ARG-TBPD.M",
                  "INTL.57-1-WP11-TBPD.M",
                  "INTL.57-1-AUT-TBPD.M",
                  "INTL.57-1-AZE-TBPD.M",
                  "INTL.57-1-BGD-TBPD.M",
                  "INTL.57-1-BLR-TBPD.M",
                  "INTL.57-1-BRA-TBPD.M",
```

```
lista_ramek = [pd.DataFrame(API.data_by_series(ID)) for ID in SERIES_ID_list]
```

```
dane = pd.concat(lista_ramek, axis=1)
```

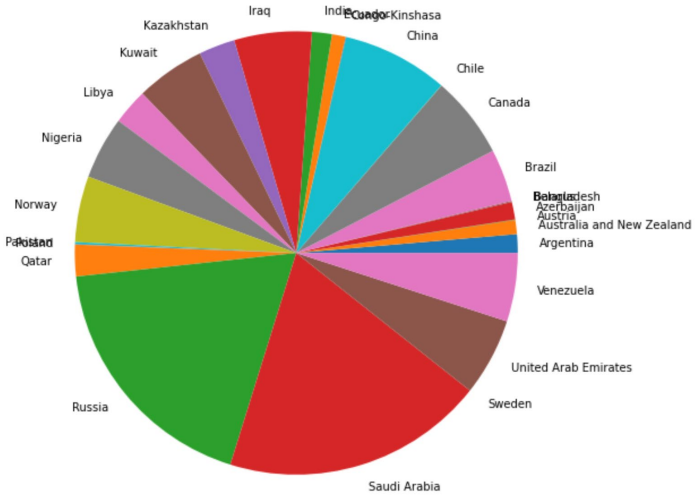
```
for col in dane.columns:
    dane.rename(columns={col: col.replace('Crude oil including lease condensate production,','').
                        replace(', Monthly (thousand barrels per day)','').
                        strip()},
                inplace=True)
```

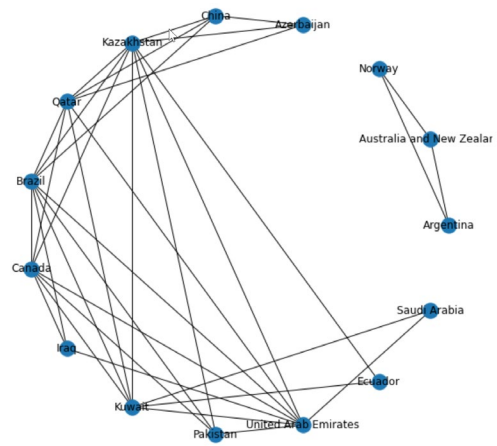
```
dane = dane.dropna()
```

```
dane.mean().sort_values(ascending=False)
```

```
plt.figure(figsize=(9,9))
plt.pie(dane.mean(), labels=dane.columns)
plt.show()
```

Saudi Arabia	9091.621386
Russia	8772.642723
China	3671.588523
Canada	2847.754127
United Arab Emirates	2689.643094
Iraq	2683.660082
Kuwait	2400.514293
Venezuela	2369.314420
Norway	2281.583281
Nigeria	2171.176036
Brazil	1824.856285






```
from github import Github
```

Go to Settings -> Developer settings -> Personal access tokens

```
USER='jdrapala'
TOKEN='ghp_AschoeringwielguscSVZ'
git=Github(USER, TOKEN)
```

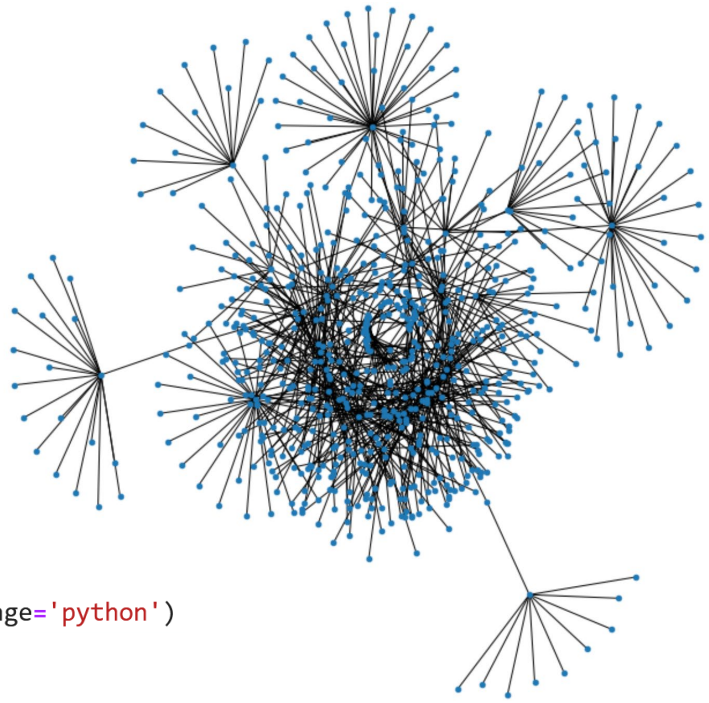
```
list(git.get_user().get_repos())
```

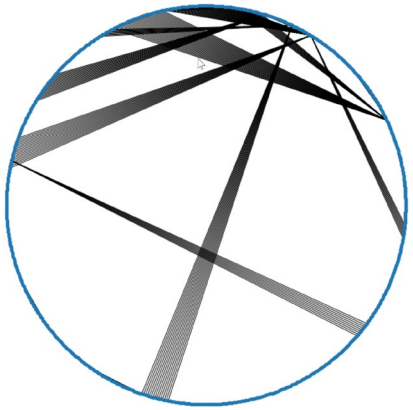
```
repozytoria = git.search_repositories(query='covid data analysis', language='python')
```

```
for repo in repozytoria:
    print(repo.full_name)
    print(set([komit.commit.author.name for komit in repo.get_commits()]))
```

```
repos_size = {repo: len(commiters) for repo, commiters in commiters_dict.items()}
df = pd.DataFrame(repos_size.items(), columns=['repo', 'No. of commiters'])
```

```
import itertools
connections_list = []
for commiters in commiters_dict.values():
    connections_list.extend([conn for conn in itertools.combinations(commiters,2)])
```





repo	No. of commiters
spectralpython/spectral	13
PyImageSearch/imutils	20
dask/dask-image	18
Image-Py/imagepy	27
ankitaggarwal011/PyCNN	14

World Bank Open Data

Free and open access to global development data

Search data e.g. GDP, population, Indonesia

Browse by Country or Indicator

DataBank

This page is i

Log in Now

- DataBank Home
- Databases
- Create Report
- Saved Reports
- Saved Datasets
- Metadata Glossary

WHAT'S NEW

Environment Social and Governance (ESG) Data was updated on October 2, 2023

Worldwide Governance Indicators was updated on September 29, 2023

Health Nutrition and Population Statistics

Explore. Create. Share: Development Data

DataBank is an analysis and visualisation tool that contains collections of time series data on a variety c queries; generate tables, charts, and maps; and easily save, embed, and share them. Enjoy using Dat think!

FAQs Feedback

Explore databases

Type keywords to filter database names

Filter by: Topic Source

Sort by: Most Used | Alphabetical | Last Updated | View all databases

Database preview: ON OFF

World Development Indicators Public

Our World in Data

Browse by topic | Latest | Resources | About

Search for a topic, chart or article...

Research and data to make progress against the world's largest problems

3704 charts across 297 topics

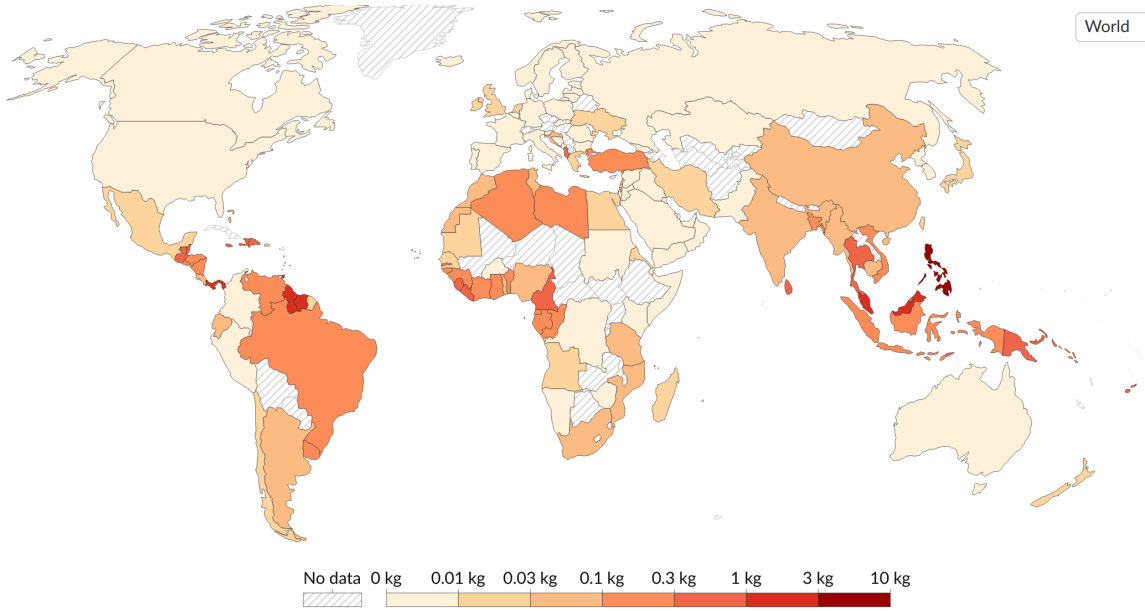
All free: open access and open source

Plastic waste emitted to the ocean per capita, 2019

This is an annual estimate of plastic emissions. A country's total does not include the waste that is exported overseas and that may be at higher risk of entering the ocean.

Our World in Data

World



Source: Meijer et al. (2021). More than 1000 rivers account for 80% of global riverine plastic emissions into the ocean. Science Advances.
OurWorldInData.org/plastic-pollution • CC BY

CHART | MAP | TABLE | SOURCES | DOWNLOAD | Sub

DATA

- DATABASE
  - Information
  - Statistical themes
  - Stats finder A-Z
  - Experimental statistics
  - Data visualisations
  - Education corner
  - Bulk download
  - Web services
  - GISCO:Geographical Information and maps
  - Microdata
  - Quality
  - Metadata
  - SDMX InfoSpace
  - Data validation
  - Evaluation

DATABASE

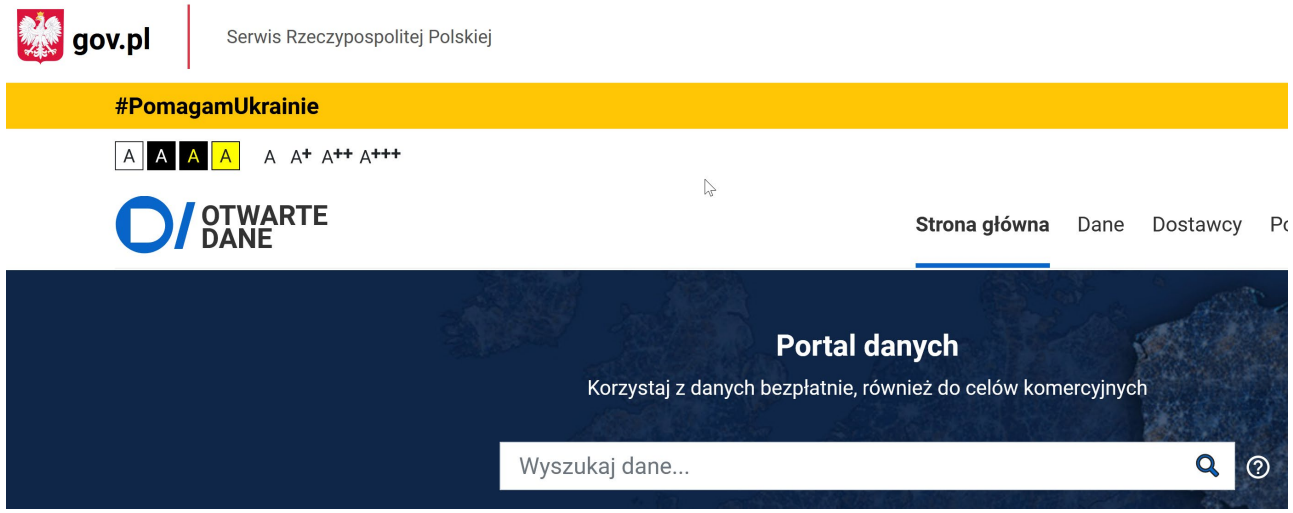
- Data navigation tree
  - Detailed datasets
    - General and regional statistics
    - Economy and finance
    - Population and social conditions
    - Industry, trade and services
    - Agriculture, forestry and fisheries
    - International trade
    - Transport
    - Environment and energy
    - Science, technology, digital society
  - Selected datasets
    - General and regional statistics

Komisja Europejska

Portal Otwartych Danych UE

Otwarte dane instytucji, agencji i innych organów UE

Dostęp do otwartych danych UE



Wybierz kategorię danych

Edukacja, kultura i sport
Liczba zbiorów danych: 147

Energia
Liczba zbiorów danych: 91

Gospodarka i finanse
Liczba zbiorów danych: 323

Kwestie międzynarodowe
Liczba zbiorów danych: 51

Ludność i społeczeństwo
Liczba zbiorów danych: 165

Nauka i technologia
Liczba zbiorów danych: 293

Regiony i miasta
Liczba zbiorów danych: 488

Rolnictwo, rybołówstwo, leśnictwo i żywność
Liczba zbiorów danych: 132

Rząd i sektor publiczny
Liczba zbiorów danych: 620

Sprawiedliwość, ustrój sądów i bezpieczeństwo publiczne
Liczba zbiorów danych: 91

Środowisko
Liczba zbiorów danych: 223

Transport
Liczba zbiorów danych: 98

Ukraina
Liczba zbiorów danych: 17

Zdrowie
Liczba zbiorów danych: 213

Stan zasilenia danych

Dane dla roku 2023 (dane krótkookresowe)
Dane dla roku 2022
Dane dla roku 2021
Komunikaty

Ankieta

Zapraszamy do wypełnienia ankiety. Państwa opinia jest dla nas bardzo ważna!

Ostatnio aktualizowane

Popularne podgrupy

Pobrane podgrupy

Linki

Dane według dziedzin

Dane roczne i krótkookresowe z wybranej dziedziny tematycznej dla wielu jednostek terytorialnych (administracyjnych i statystycznych). Informacje można prezentować w tablicach i na wykresach

Dane dla jednostki terytorialnej

Dane dla wybranej jednostki administracyjnej (miejscowości, gminy, powiatu, województwa, kraju) lub statystycznej (makroregionu, regionu, podregionu) z wielu dziedzin tematycznych.

Interfejs API

Dostęp do danych Banku poprzez interfejs REST API w formacie Json i Xml. API do BDL zrealizowano w ramach partnerskiego projektu „Otwarte dane – dostęp, standard, edukacja", którego Liderem jest Ministerstwo Cyfryzacji. Projekt jest współfinansowany ze środków Unii Europejskiej z Programu Operacyjnego Polska Cyfrowa.

BANK DANYCH LOKALNYCH jest największą w Polsce bazą danych o gospodarce, społeczeństwie i środowisku. BDL oferuje ponad 40 tys. cech statystycznych pogrupowanych tematycznie. Pierwsze dane pochodzą z 1995 roku.

- Dane i wskaźniki opisują miejscowości statystyczne, gminy, powiaty, województwa i Polskę, a także jednostki zgodne z nomenklaturą NUTS: makroregiony, regiony i podregiony.

BANKDANYCH.IO

Przeglądarka danych statystycznych GUS

REJESTR.IO

Wyszukiwarka danych o spółkach, fundacjach i stowarzyszeniach

TENDERS.GURU

Portal monitorujący zamówienia publiczne

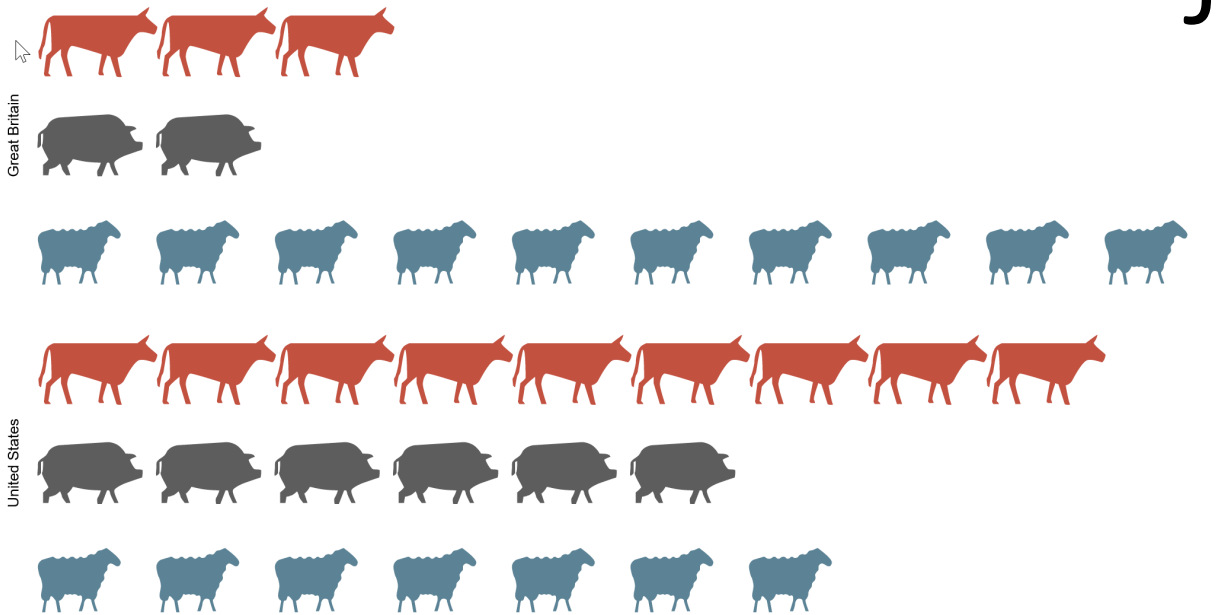
KTOILE.PL

Wykaz instytucji, którym przyznawana jest pomoc publiczna

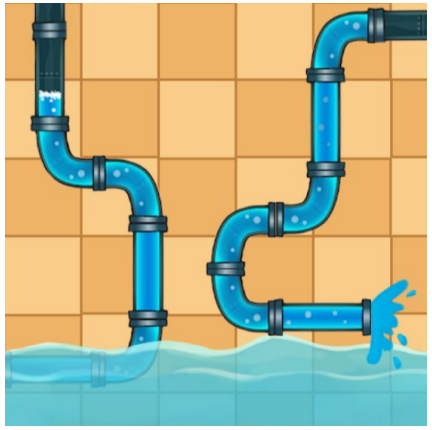

Prześwietl.pl
NASZ PRODUKT
CENNIK
O NAS
ZALOGUJ
ZAŁÓŻ KONTO
Wywiad Gospodarczy Online
Sprawdź z kim robisz biznes
Wyszukiwarka firm, stowarzyszeń, fundacji i osób, powiązania kapitałowo-osobowe dzięki informacjom z KRS, historia podmiotów, raporty handlowe, najnowsze odpisy z rejestru KRS, statusy VAT i informacje z Białej Listy. Miliony danych z różnych źródeł w jednym miejscu.
WYPRÓBUJ ZA DARMO
3 raporty handlowe PREMIUM za darmo i 7 dni testów bez zobowiązań

i wiele innych …

Oporządzanie danych i wizualizacja



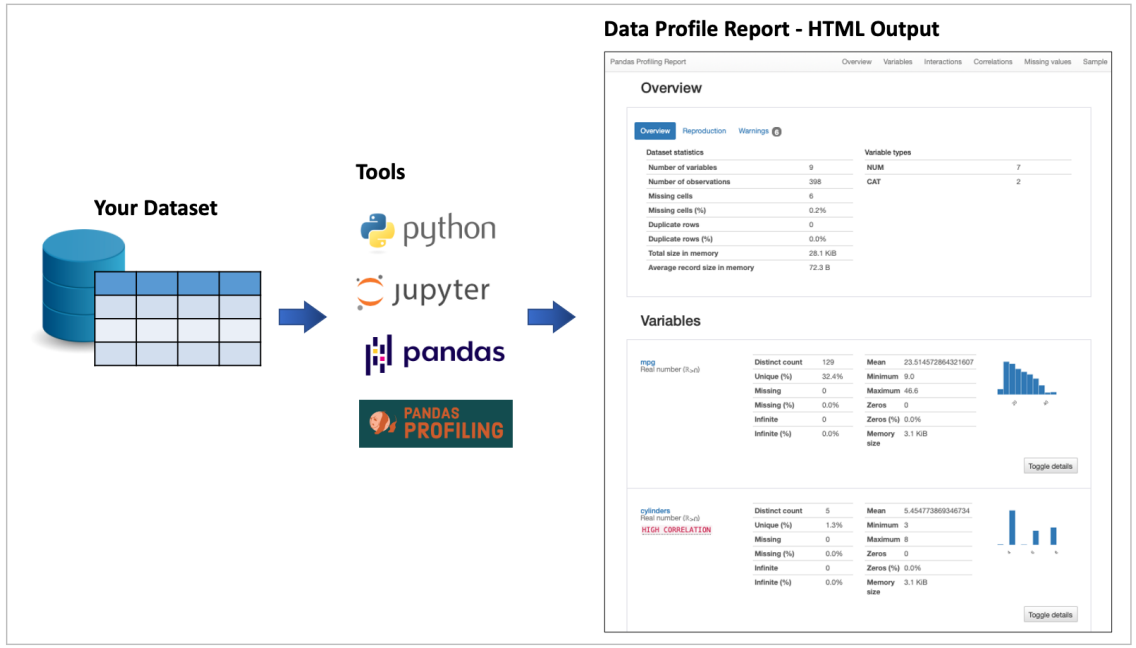
pandas

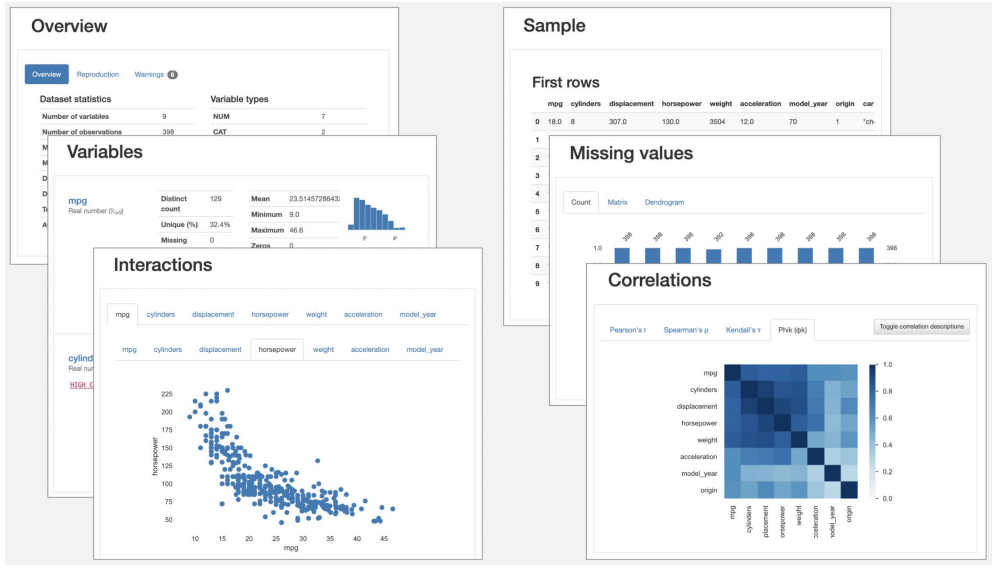


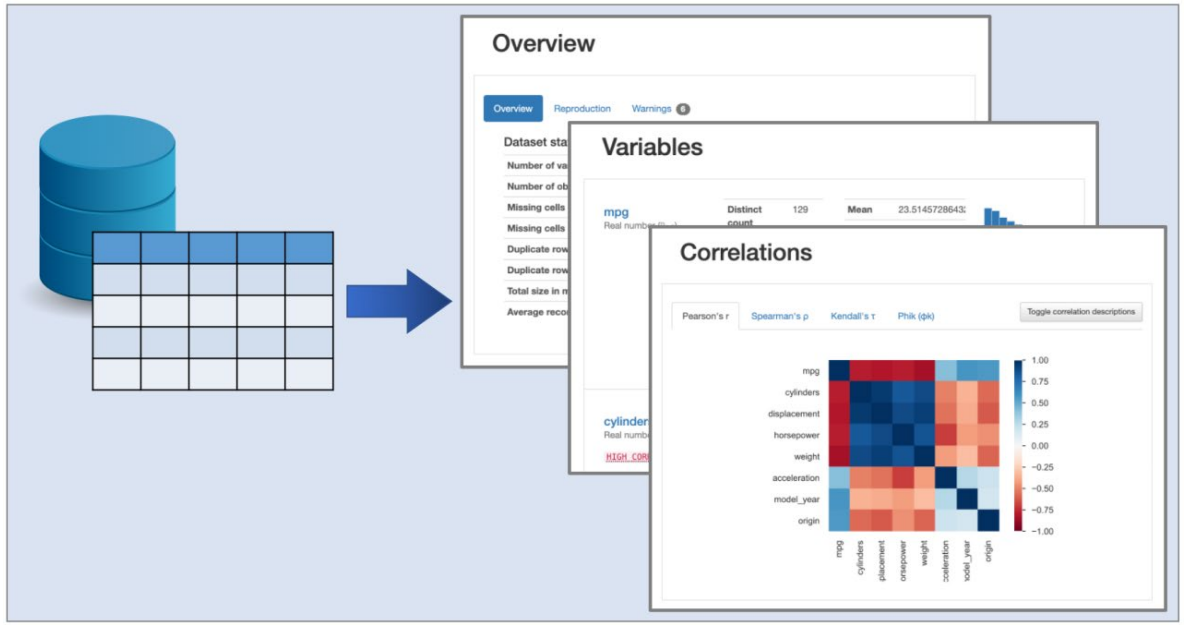
```python
def remove_high(df, column, value):
    return df[abs(df[column]) <= value]

def add_sign_column(df, column):
    df_copy = df.copy()
    df_copy[column+'_sign'] = np.sign(df[column])
    return df_copy

(df_ext.
 pipe(remove_high, column='Z', value=1).
 pipe(add_sign_column, column='Z')
)
```

	Height	Weight	BMI	Cholesterol LDL	Cholesterol HDL	Gender	Age
306	170	115	39	145	25	Male	70
371	179	108	33	54	29	Male	64
383	160	59	23	183	38	Male	68
611	194	82	21	76	62	Male	63
543	170	99	34	62	30	Male	58
135	156	77	31	99	56	Female	69
644	185	80	23	68	36	Male	68

PANDAS PROFILING



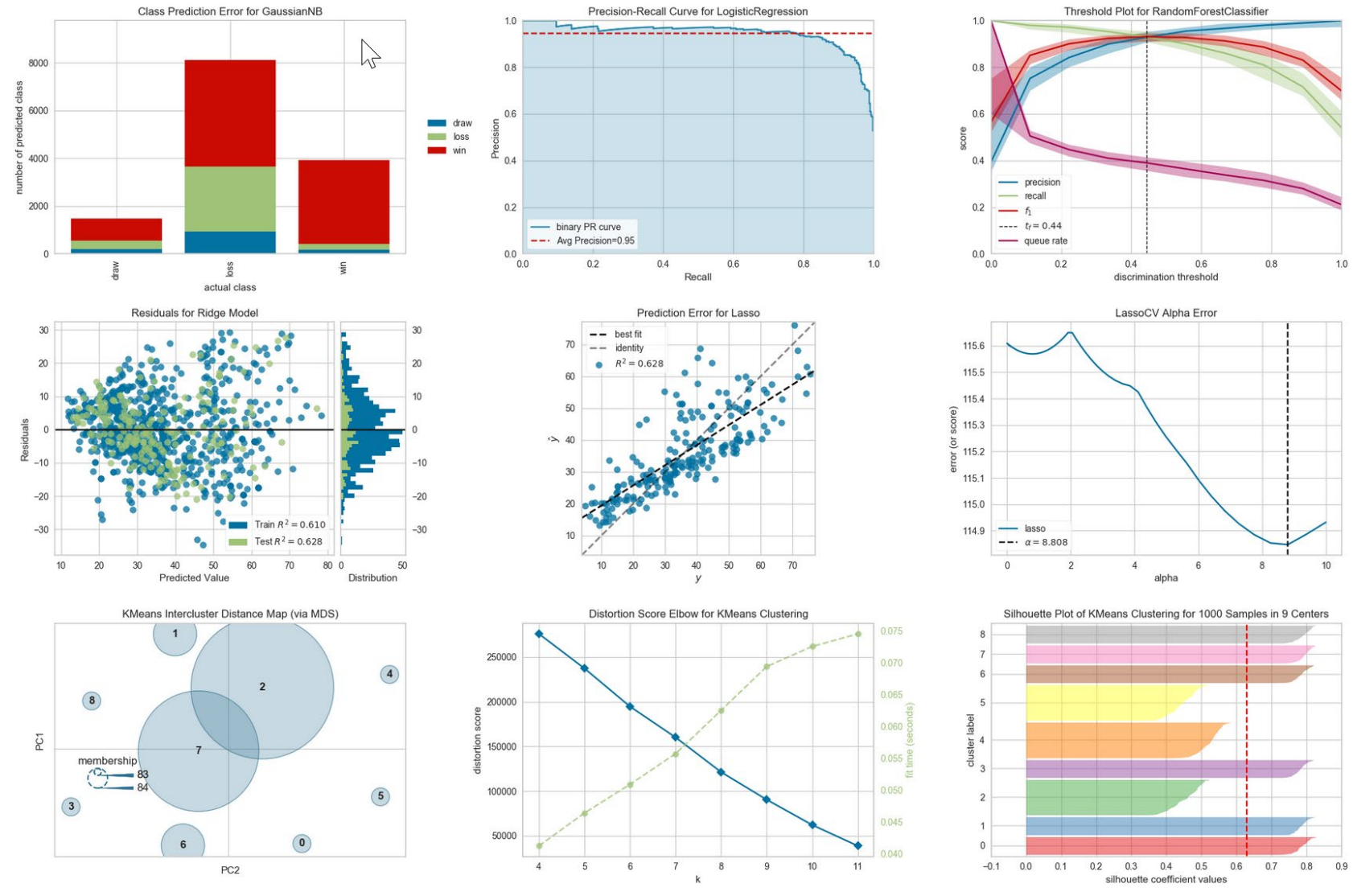




SweetViz


SweetViz
DataFrame
Compared
TV
Radio
Newspaper
Sales
NUMERICAL ASSOCIATIONS
CATEGORICAL ASSOCIATIONS
MOST FREQUENT VALUES
SMALLEST VALUES
LARGEST VALUES

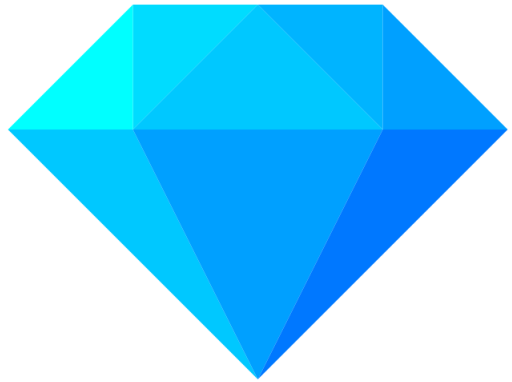
Yellowbrick: Machine Learning Visualization



OpenRefine

OpenRefine is a powerful free, open source tool for working with messy data: cleaning it; transforming it from one format into another; and extending it with web services and external data.

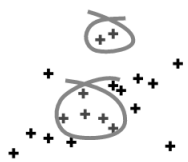
Download



Main features

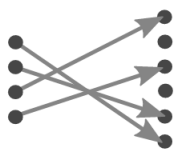
Faceting

Drill through large datasets using facets and apply operations on filtered views of your dataset.



Clustering

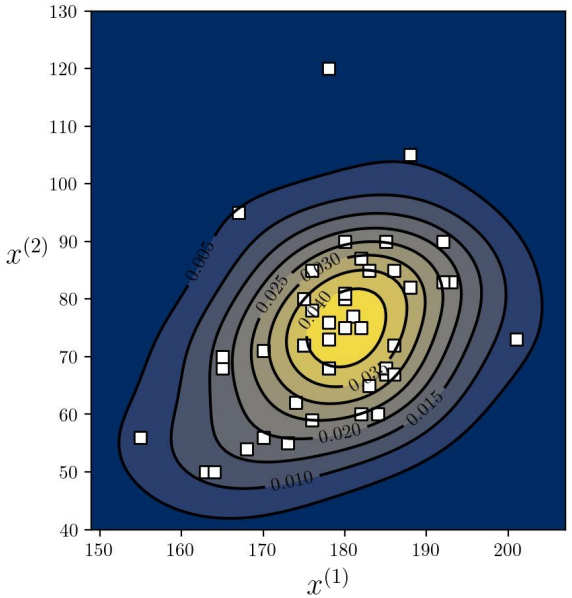
Fix inconsistencies by merging similar values thanks to powerful heuristics.

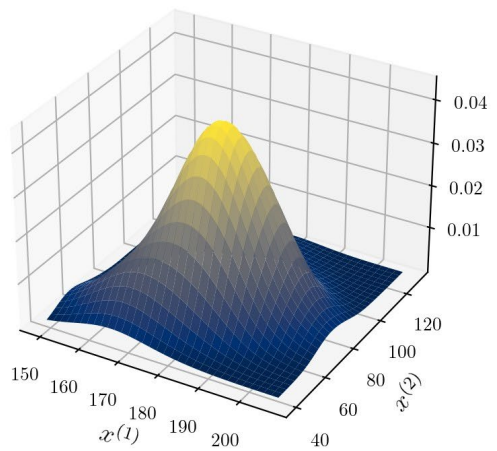


Reconciliation

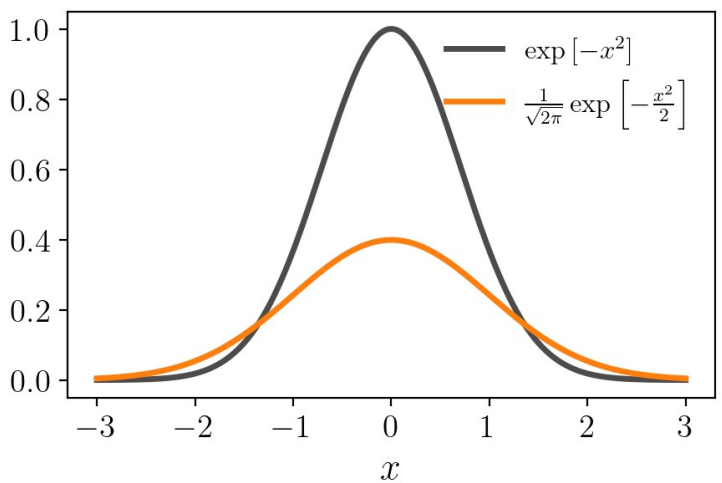
Match your dataset to external databases via reconciliation services.

matplotlib



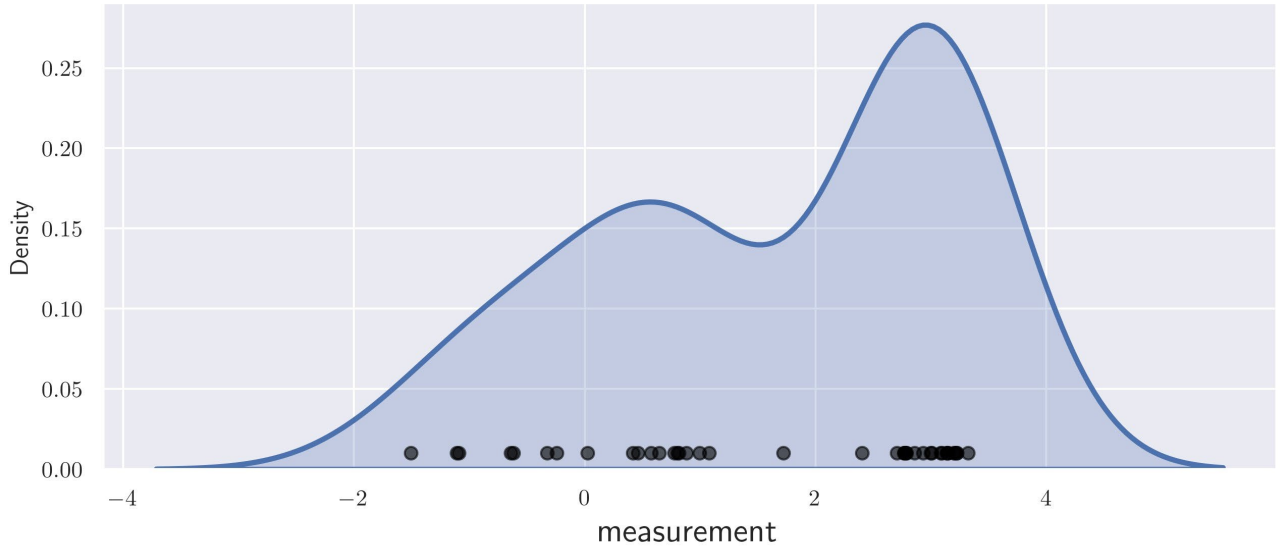



```
import matplotlib.pyplot as plt
import numpy as np

plt.rcParams.update({
    "text.usetex": True,
    "font.family": "sans-serif",
    "font.sans-serif": ["Helvetica"]
})

xx = np.linspace(start=-3, stop=3, num=300)
yy = np.exp(-xx**2)
yy_rescaled = np.exp(-xx**2/2)/np.sqrt(2*np.pi)

plt.figure(figsize=(5,3))
plt.plot(xx, yy, color='black',
         linewidth=2.5, alpha=0.7,
         label=r'$\exp\left[-x^2\right]$')
plt.plot(xx, yy_rescaled, color='tab:orange',
         linewidth=2.5, alpha=1,
         label=r'$\frac{1}{\sqrt{2\pi}}\exp\left[-\frac{x^2}{2}\right]$')
plt.xlabel('$x$', fontsize=16)

plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.legend(fontsize=12,frameon=False)
plt.savefig('filename.png', dpi=200, bbox_inches = 'tight', pad_inches = 0.1)
plt.show()
```

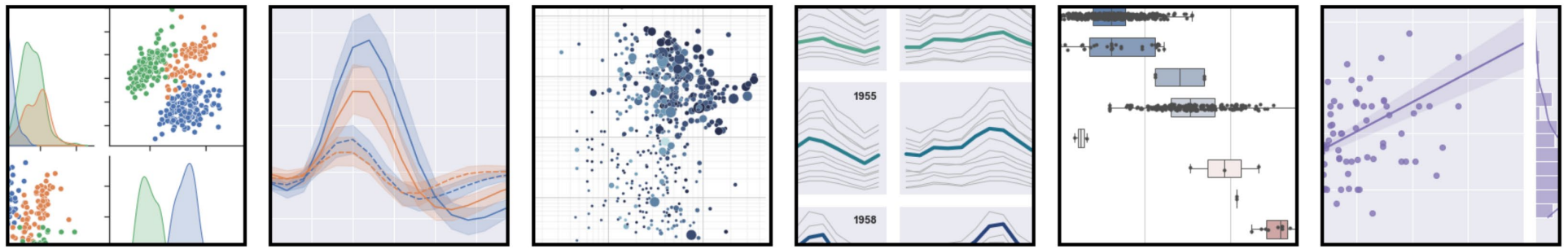


seaborn

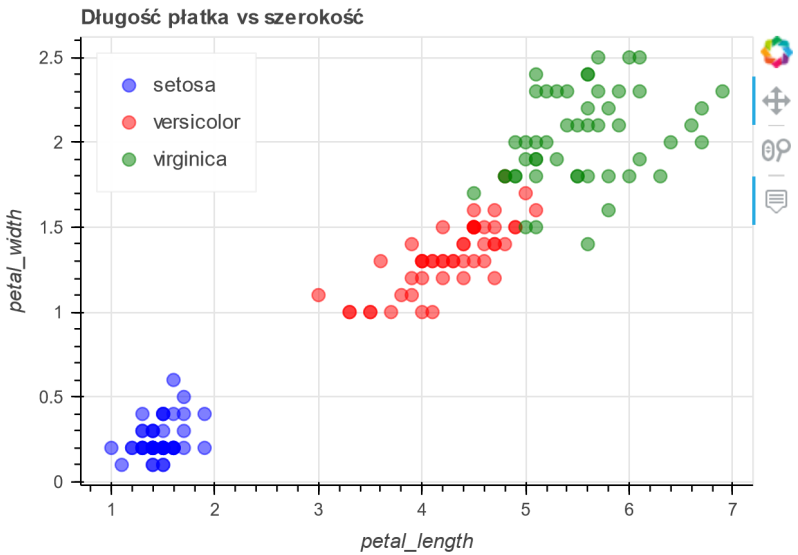
```
import matplotlib.pyplot as plt
import numpy as np

import seaborn as sns
sns.set()

X = np.concatenate([np.random.normal(0,1, size=20),
                    np.random.normal(3,0.3, size=20)])

fig, ax = plt.subplots(figsize=(10,4))
sns.kdeplot(ax=ax, data=X,
            # bw_adjust=0.3,
            linewidth=2.5, fill=True)

ax.plot(X, np.zeros_like(X) + 0.01, 'o',
        markersize=6, color='black', alpha=0.6)
ax.set_xlabel('measurement', fontsize=16)
plt.show()
```

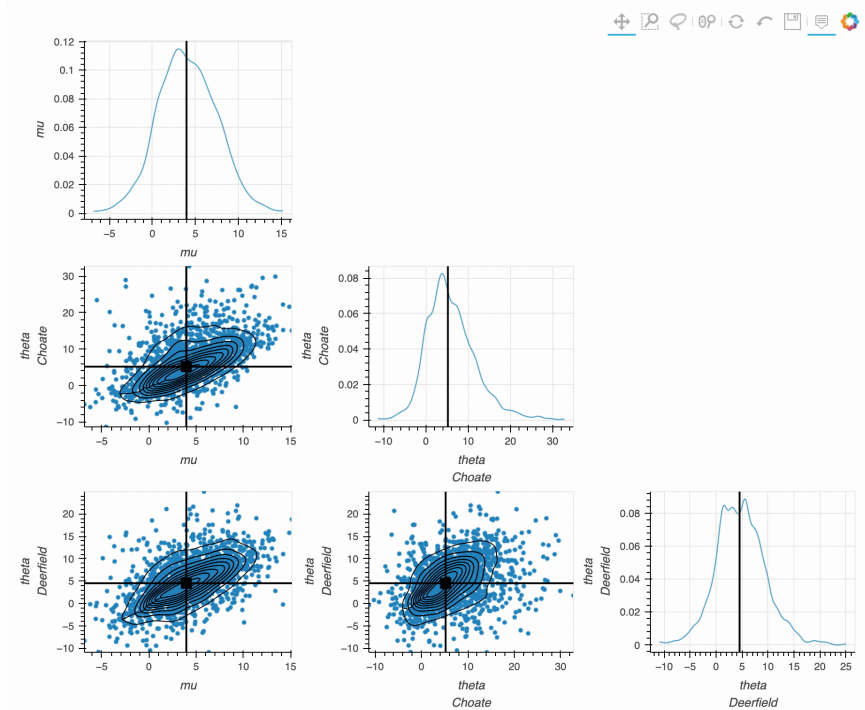


seaborn: statistical data visualization

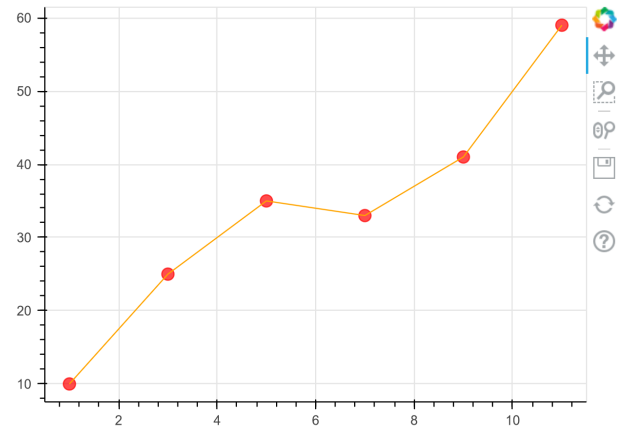


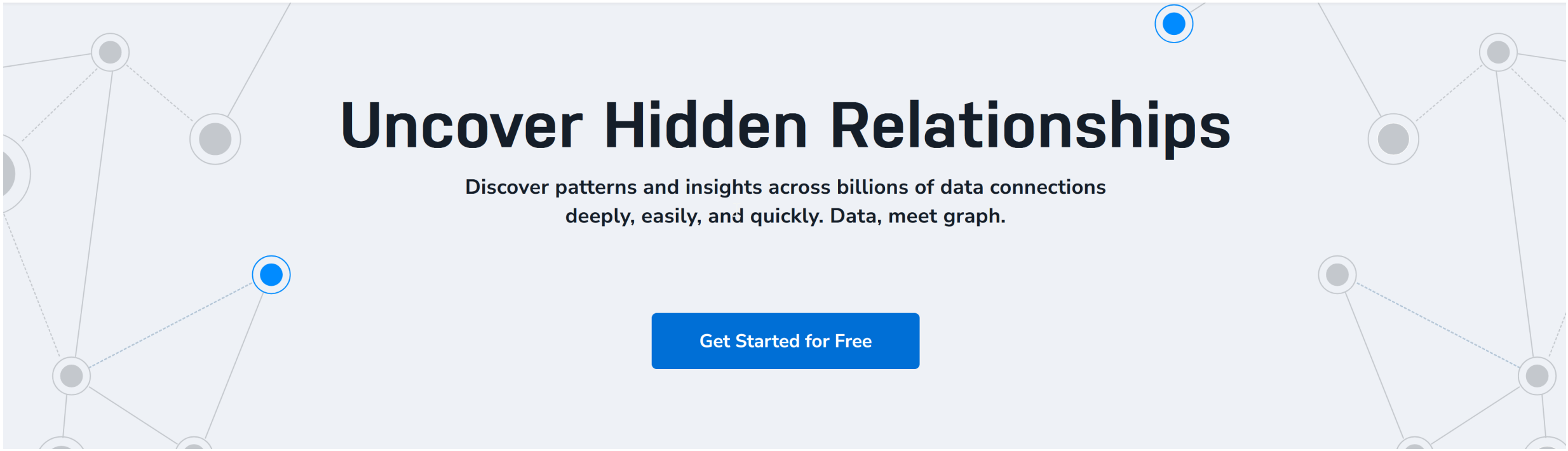
bokeh






```
from bokeh.plotting import figure, output_notebook, show

x = [1, 3, 5, 7, 9, 11]
y = [10, 25, 35, 33, 41, 59]

output_notebook()

fig = figure(plot_width=500, plot_height=350)
fig.circle(x,y, size=10, color='red', alpha=0.7)
fig.line(x,y, line_width=1, color='orange')
show(fig)
```




Partners
Company
Support
neo4j
Products
Solutions
Learn
Developers
Data Scientists
Pricing
GenAI
Contact Us
Get Started Free
Uncover Hidden Relationships
Discover patterns and insights across billions of data connections deeply, easily, and quickly. Data, meet graph.
Get Started for Free

Gephi
makes graphs handy

Download | Blog | Wiki | Ask a question | Support | Bug tracker

Home | Features | Learn | Develop | Plugins

The Open Graph Viz Platform

Gephi is the leading visualization and exploration software for all kinds of graphs and networks. Gephi is open-source and free.

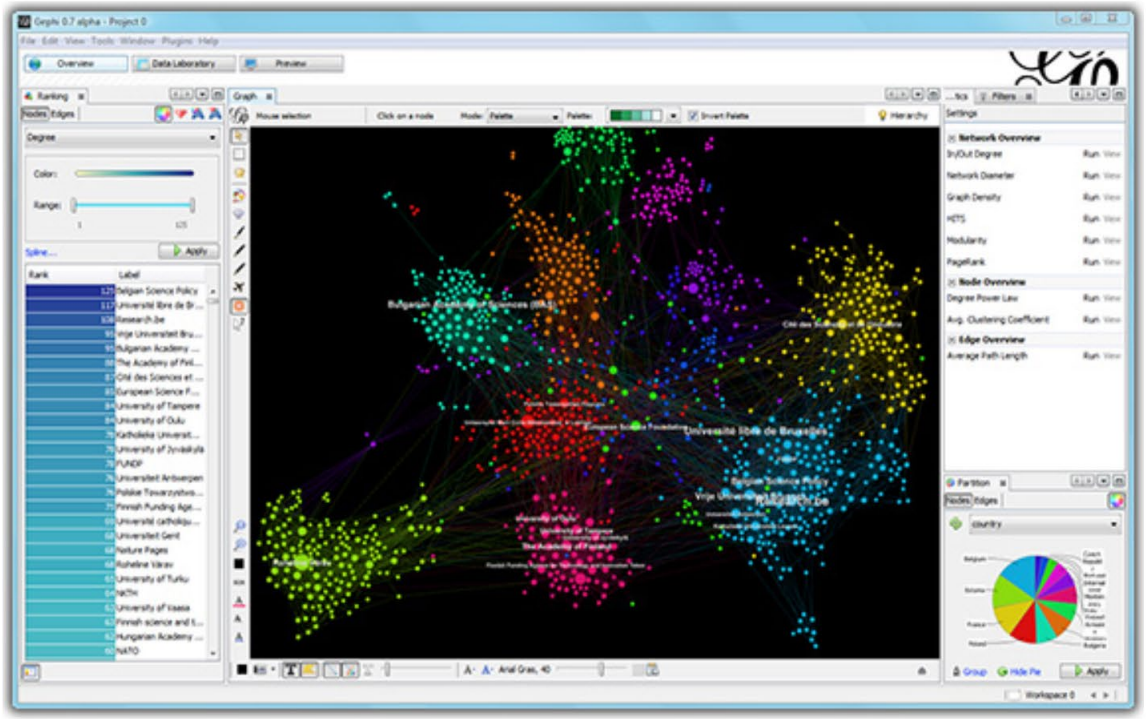
Runs on Windows, Mac OS X and Linux.

Learn More on Gephi Platform »

Download FREE
Gephi 0.10

Release Notes | System Requirements

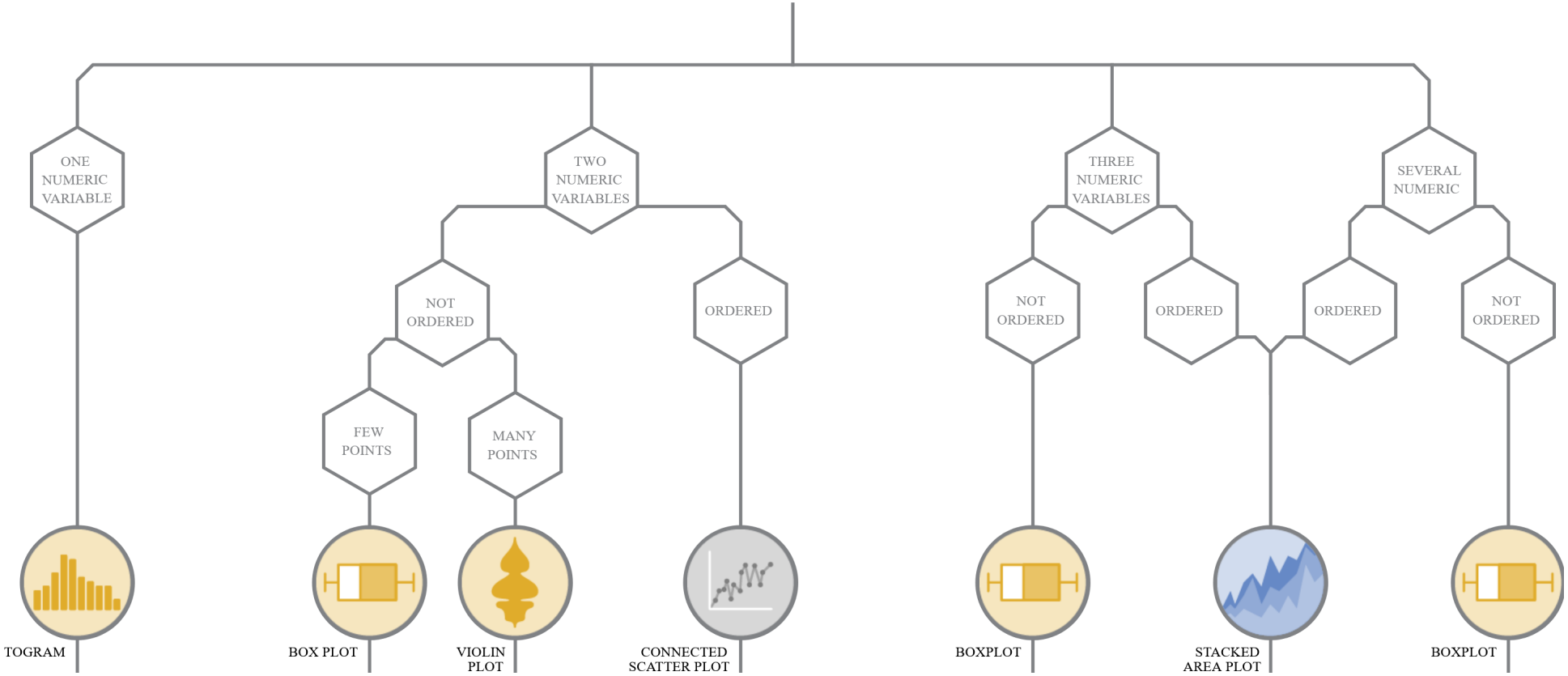
- **Features**
- **Quick start**
- **Screenshots**
- **Videos**



from Data to Viz

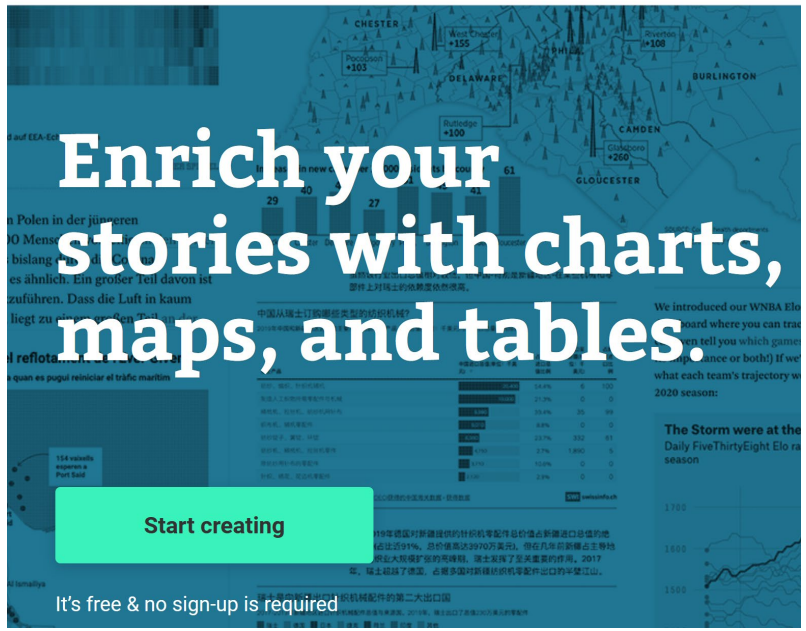
What kind of data do you have? Pick the main type using the buttons below. Then let the decision tree guide you toward your graphic possibilities.

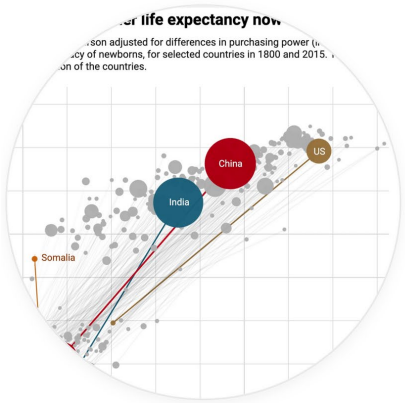
Numeric | Categoric | Num & Cat | Maps | Network | Time series



Datawrapper

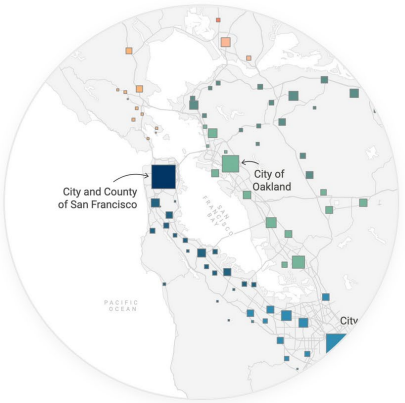
Product





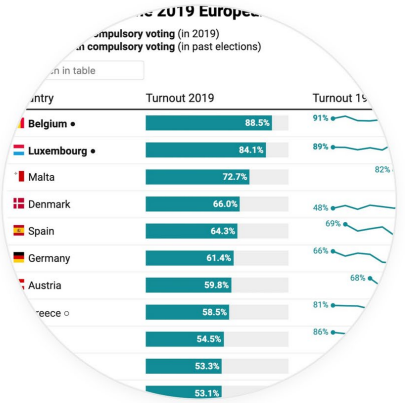
Charts

Use one of **19 interactive and responsive chart types** ranging from simple bars and lines to arrow, range, and scatter plots.



Maps

Three interactive and responsive map types let you create anything from locator maps to thematic choropleth and symbol maps.



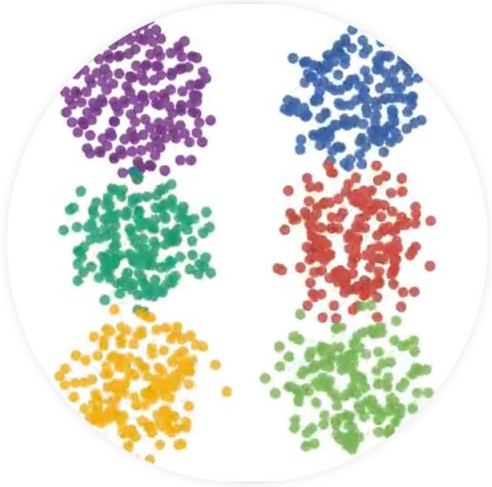
Tables

Responsive data tables allow for lots of styling options and let you include bar, column & line charts, heatmaps, images, a search bar and pagination.

Flourish

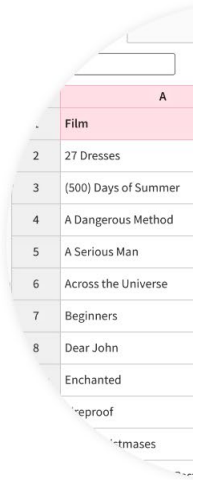
Beautiful and easy **data visualization and storytelling**

Easily turn your data into stunning charts, maps and interactive stories. View examples.

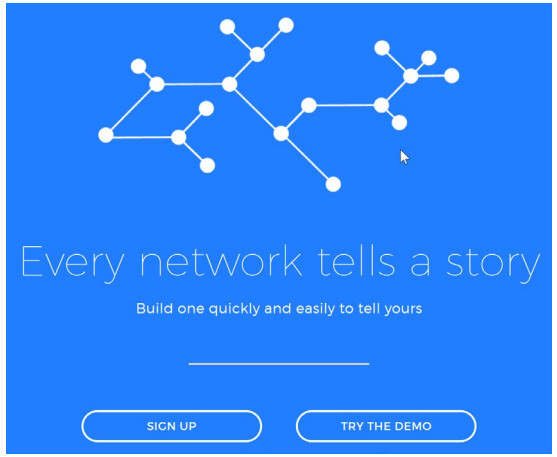


Engage your audience

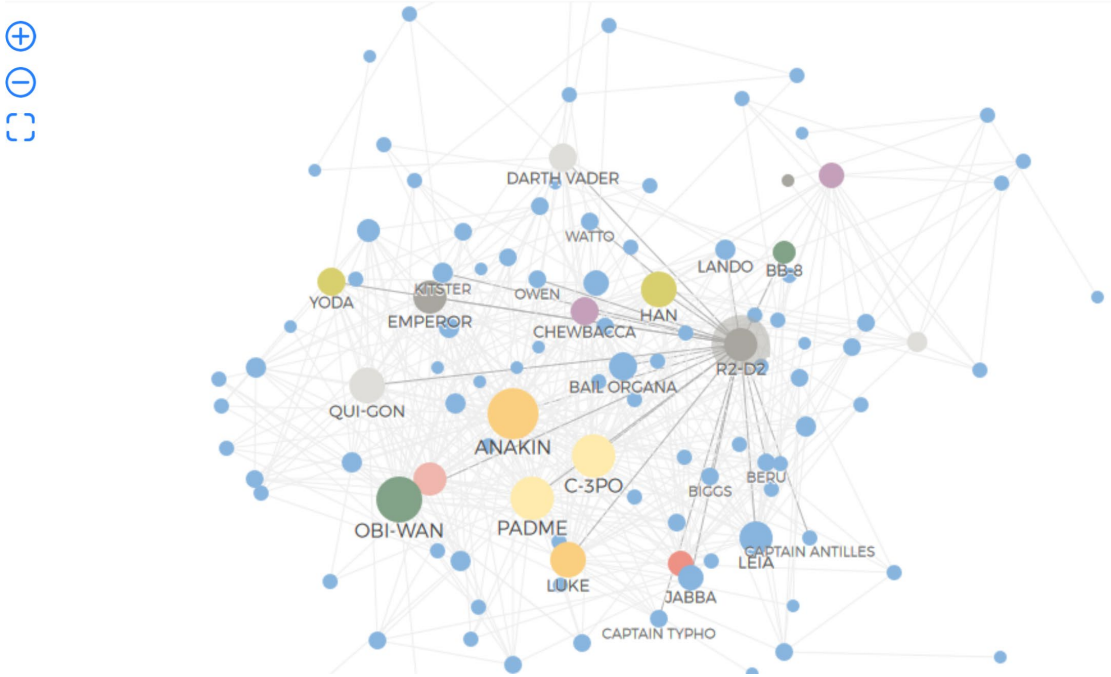
Create agency-quality data graphics and animated stories that bring your data to life.



ONODO



Star Wars Social Network



R2-D2

TYPE

MENTIONS

+ a b | e a u®

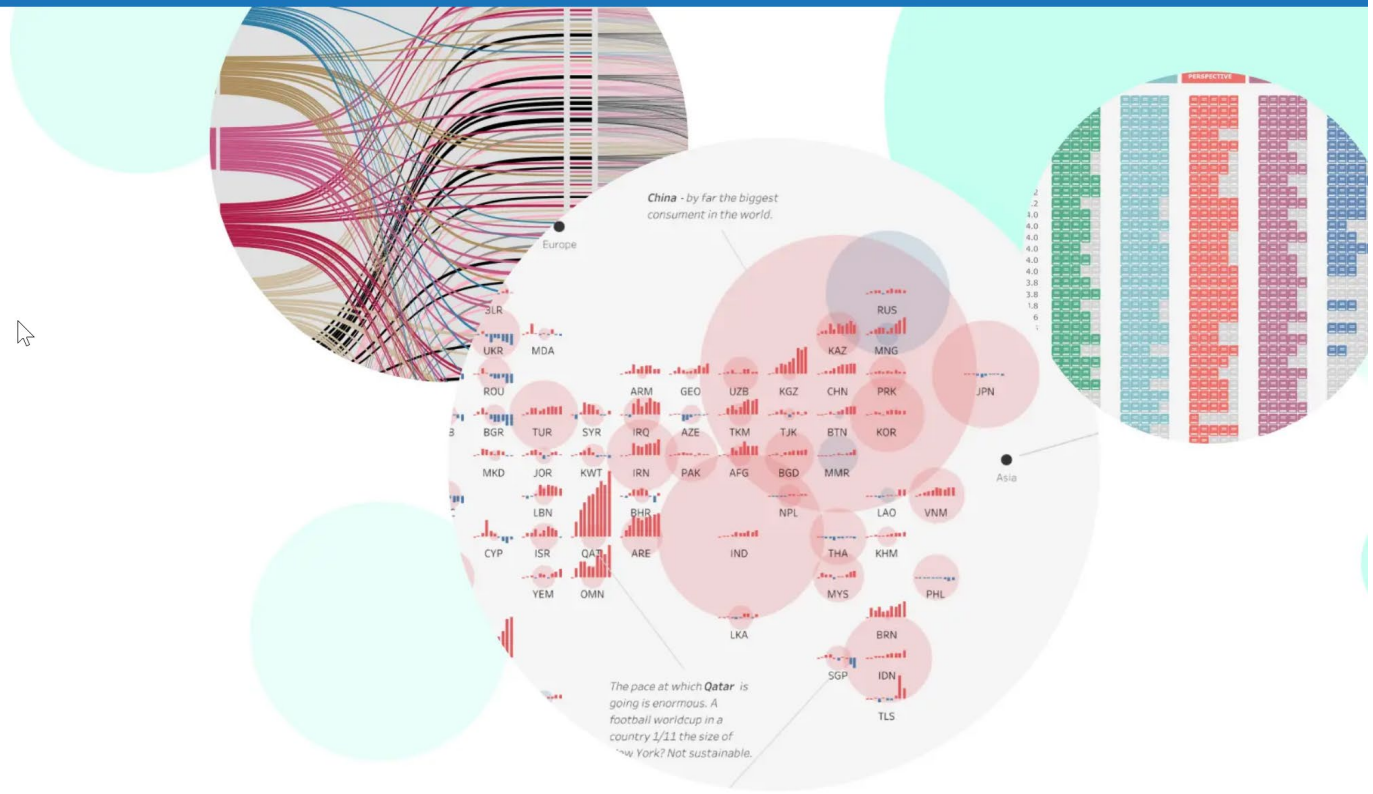
+ableau public Create Learn

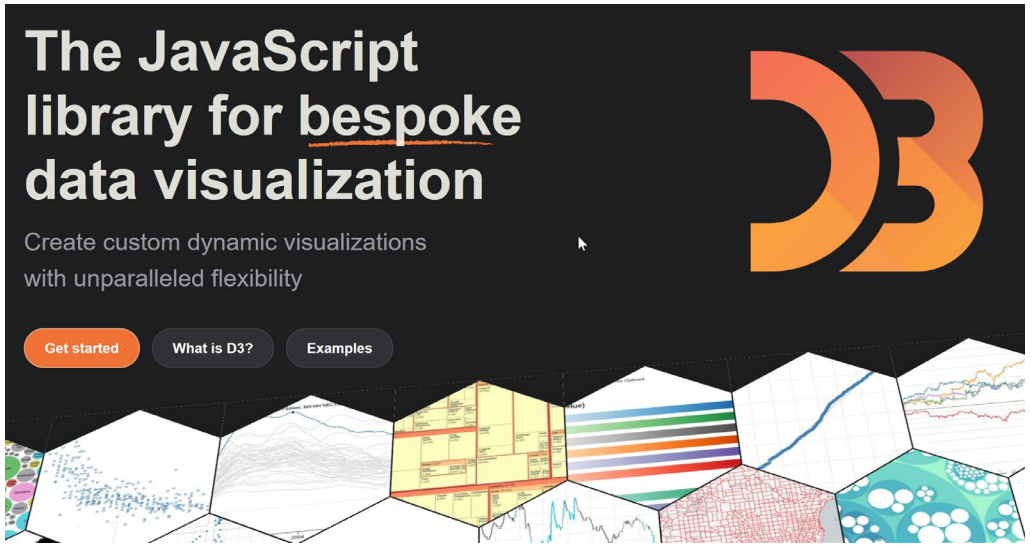
Want to take your data skills to the next level? Connect with the Tableau Community to accelerate your learning. **Show me →**

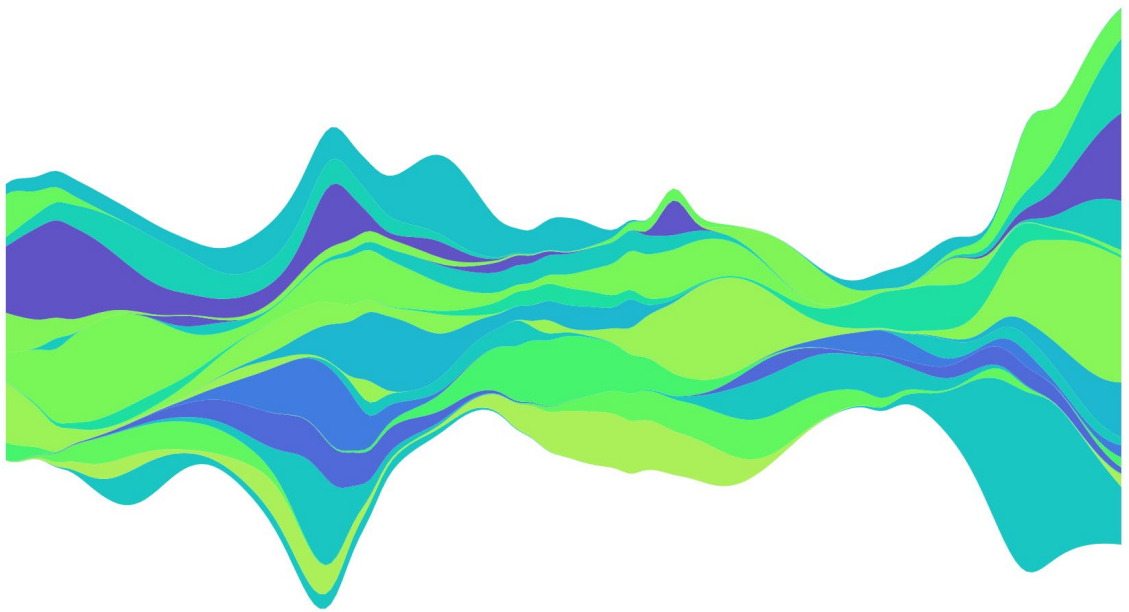
Welcome to Tableau Public

A free platform to explore, create, and publicly share data visualizations online.

Sign Up for Tableau Public Learn More







```
chart = {
  const width = 928;
  const height = 500;

  const x = d3.scaleLinear([0, m - 1], [0, width]);
  const y = d3.scaleLinear([0, 1], [height, 0]);
  const z = d3.interpolateCool;

  const area = d3.area()
    .x((d, i) => x(i))
```

Vega-Lite – A Grammar of Interactive Graphics