Toulouse Data Engineering

Construire et déployer une application de visualisation des données COVID-19.

Partie 1: Présentation Altair, une librairie python de visualisation déclarative

Altair: Declarative Visualization in Python

Altair is a declarative statistical visualization library for Python, based on Vega and Vega-Lite, and the source is available on GitHub.

With Altair, you can spend more time understanding your data and its meaning. Altair's API is simple, friendly and consistent and built on top of the powerful Vega-Lite visualization grammar. This elegant simplicity produces beautiful and effective visualizations with a minimal amount of code.

altair-viz / altair

Used by 1.9k | Watch 150 | Star 5k | Fork 490

Code | Issues 145 | Pull requests 11 | Actions | Projects 0 | Wiki | Security | Insights

Declarative statistical visualization library for Python https://altair-viz.github.io/

2,947 commits | 33 branches | 0 packages | 22 releases | 96 contributors | BSD-3-Clause

Jake Vanderplas
jakevdp

Unfollow

Python, Astronomy, Data Science
Google
Seattle WA
http://www.vanderplas.com

Block or report user

Organizations

Overview | Repositories 205 | Projects 0 | Stars 45 | Followers 10.4k | Following 6

Pinned

PythonDataScienceHandbook
Python Data Science Handbook: full text in Jupyter Notebooks
Jupyter Notebook ★ 22.5k 9.6k

altair-viz/altair
Declarative statistical visualization library for Python
Python ★ 5k 490

WhirlwindTourOfPython
The Jupyter Notebooks behind my OReilly report, "A Whirlwind Tour of Python"
Jupyter Notebook ★ 2.3k 1k

sklearn_tutorial
Materials for my scikit-learn tutorial
Jupyter Notebook ★ 1.2k 729

1,532 contributions in the last year

Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar
Mon
Wed
Fri

Learn how we count contributions.

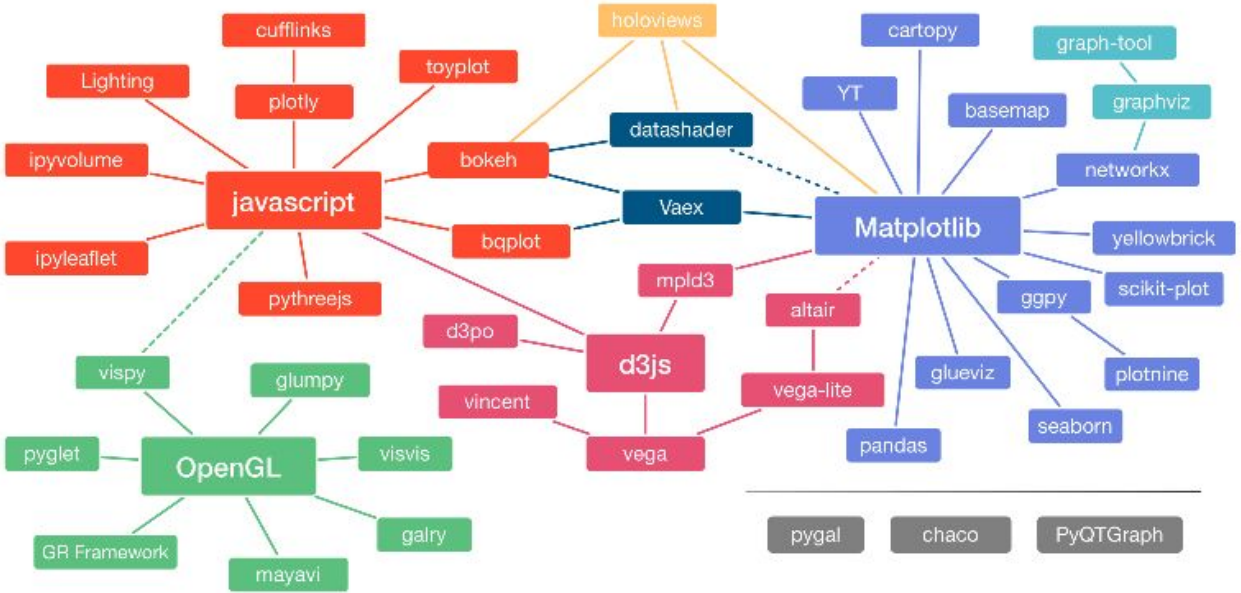
Less More

PyViz

Navigation

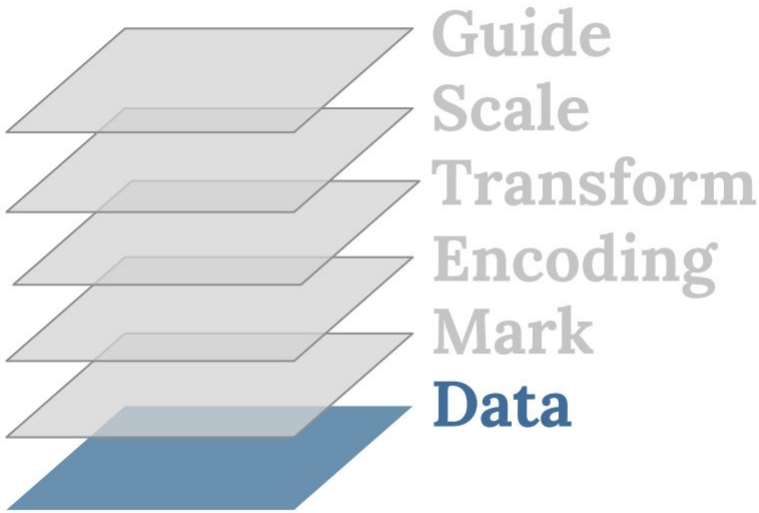
Overviews

The Python visualization landscape can seem daunting at first. These overviews attempt to shine light on common patterns and use cases, c
cussed in the overviews are no longer maintained, so be sure to check the list of dormant projects before choosing that library.

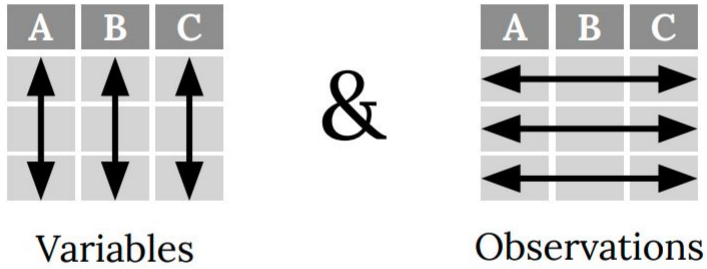


Visualization Grammar

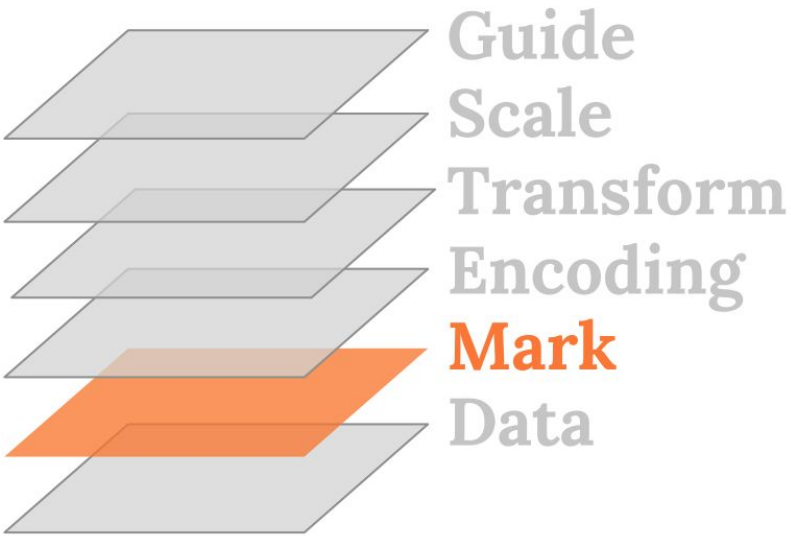
https://eitanlees.com/Visualization-Grammar.pdf



Tabular Data

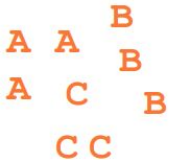


https://eitanlees.com/Visualization-Grammar.pdf

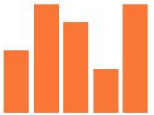


Line

Circle

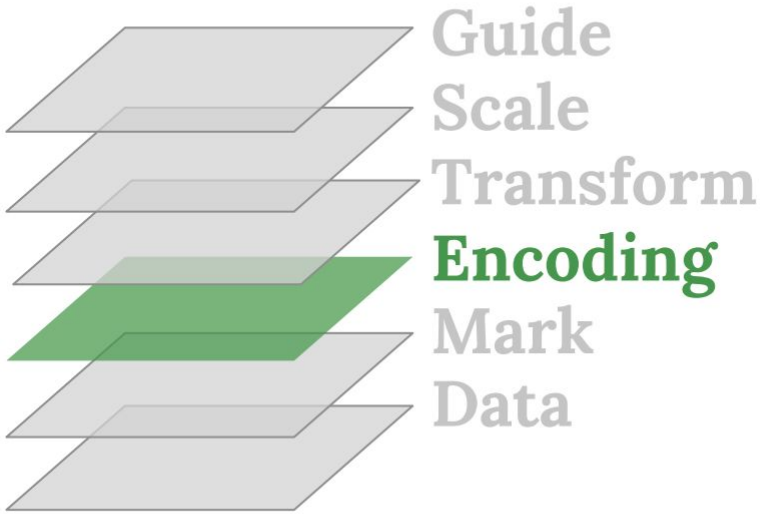


Text



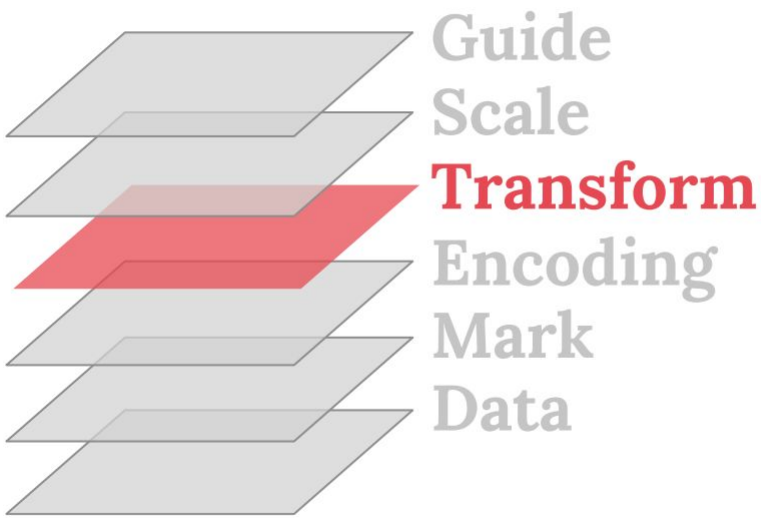
Bar

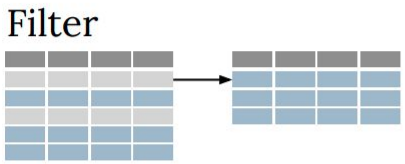
https://eitanlees.com/Visualization-Grammar.pdf

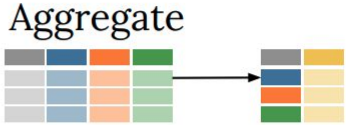


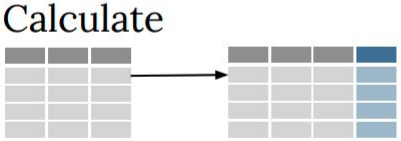
Channel	Variable
X Position	A
Y Position	B
Size	C
Color	D
⋮	⋮

https://eitanlees.com/Visualization-Grammar.pdf



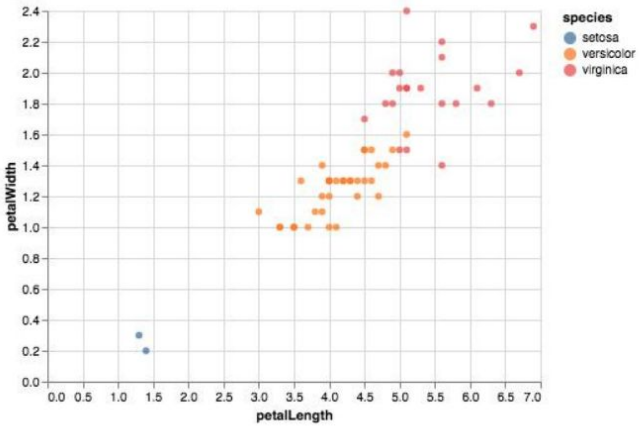






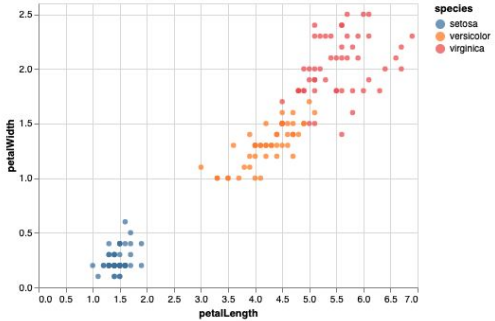
https://eitanlees.com/Visualization-Grammar.pdf


```
import altair as alt
from vega_datasets import data

iris = data.iris()
alt.Chart(iris).mark_circle().encode(
    alt.X('petalLength'),
    alt.Y('petalWidth'),
    alt.Color('species')
).transform_filter(
    alt.datum.sepalWidth < 3
)
```



https://eitanlees.com/Visualization-Grammar.pdf

Vega




```
{
  "config": {"view": {"continuousWidth": 400, "continuousHeight": 300}},
  "data": {"name": "data-a264acbd6e539a8b3afc0cb5f240fb57"},
  "mark": "circle",
  "encoding": {
    "color": {"type": "nominal", "field": "species"},
    "x": {"type": "quantitative", "field": "petalLength"},
    "y": {"type": "quantitative", "field": "petalWidth"}
  },
  "$schema": "https://vega.github.io/schema/vega-lite/v4.0.2.json",
  "datasets": {
    "data-a264acbd6e539a8b3afc0cb5f240fb57": [
      {
        "sepalLength": 5.1,
        "sepalWidth": 3.5,
        "petalLength": 1.4,
        "petalWidth": 0.2,
        "species": "setosa"
      },
      {
        "sepalLength": 4.9,
        "sepalWidth": 3,
        "petalLength": 1.4,
        "petalWidth": 0.2,
        "species": "setosa"
      },
```


```
import altair as alt
from vega_datasets import data

iris = data.iris()
alt.Chart(iris).mark_circle().encode(
    alt.X('petalLength'),
    alt.Y('petalWidth'),
    alt.Color('species')
).transform_filter(
    alt.datum.sepalWidth < 3
)
```

link

HTML, SVG, and CSS

JSON

Python

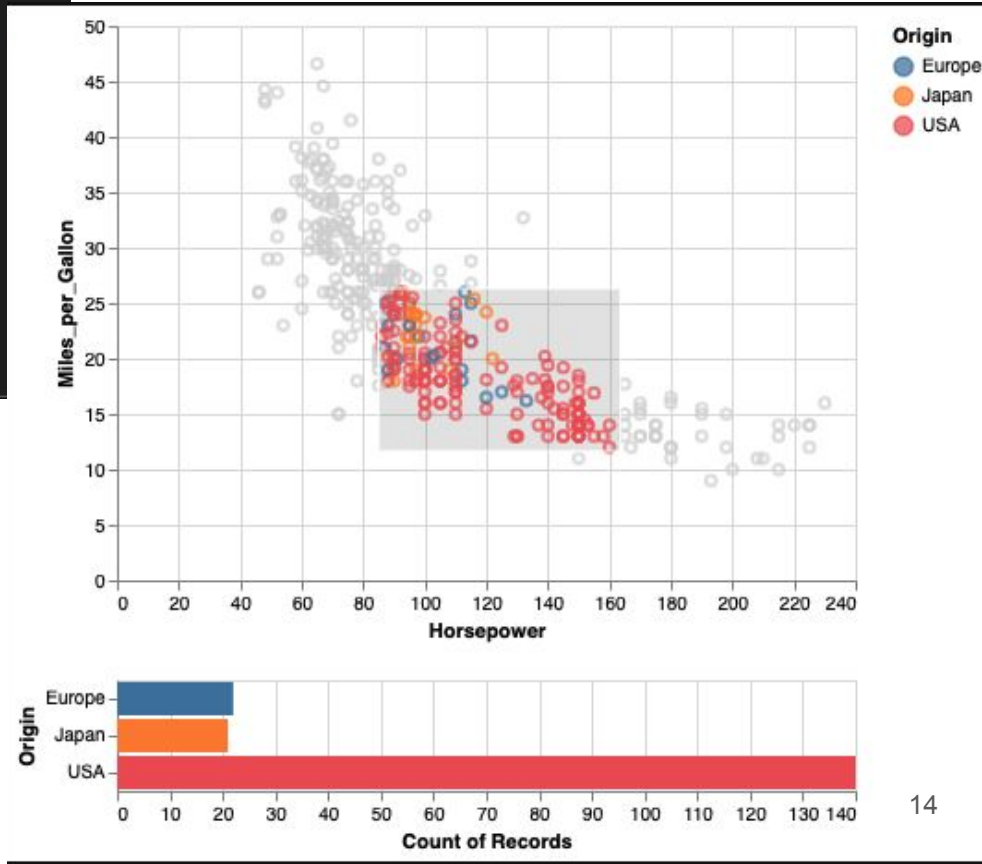
[1]:
```
import altair as alt
from vega_datasets import data
source = data.cars()
source.head()
```

[1]:

	Name	Miles_per_Gallon	Cylinders	Displacement	Horsepower	Weight_in_lbs	Acceleration	Year	Origin
0	chevrolet chevelle malibu	18.0	8	307.0	130.0	3504	12.0	1970-01-01	USA
1	buick skylark 320	15.0	8	350.0	165.0	3693	11.5	1970-01-01	USA
2	plymouth satellite	18.0	8	318.0	150.0	3436	11.0	1970-01-01	USA
3	amc rebel sst	16.0	8	304.0	150.0	3433	12.0	1970-01-01	USA
4	ford torino	17.0	8	302.0	140.0	3449	10.5	1970-01-01	USA

[2]:
```
brush = alt.selection(type='interval')

points = alt.Chart(source).mark_point().encode(
    x='Horsepower',
    y='Miles_per_Gallon',
    color=alt.condition(brush, 'Origin', alt.value('lightgray')),
    tooltip='Name:N'
).add_selection(
    brush
)

bars = alt.Chart(source).mark_bar().encode(
    y='Origin',
    color='Origin',
    x='count(Origin)'
).transform_filter(
    brush
)

points & bars
```

link



Partie 2: Présentation Streamlit, un app framework pour constuire des data app

Streamlit.

The fastest way to build custom ML tools

Streamlit is an open-source app framework for Machine Learning and Data Science teams. Create beautiful data apps in hours, not weeks. All in pure Python. All for free.

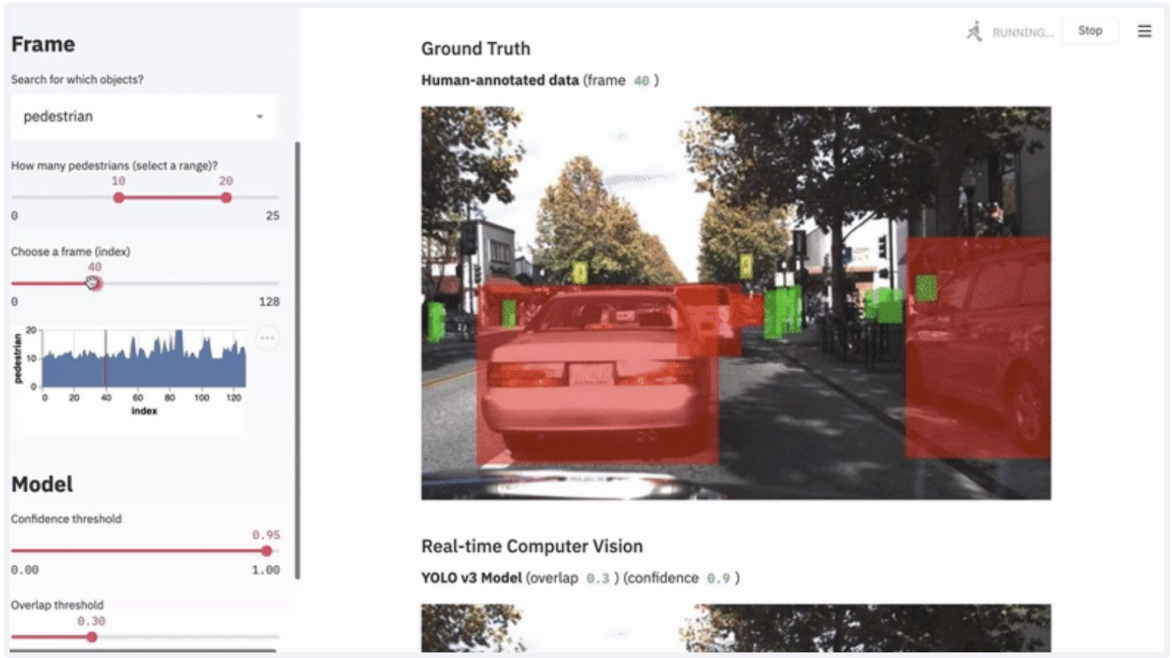
streamlit

Streamlit — The fastest way to build custom ML tools

python data-science machine-learning deep-learning

data-visualization data-analysis streamlit

Python Apache-2.0 647 7,316 392 (7 issues need help) 14 Updated 10 hours ago



Featured

Real time object detection

An image browser for the Udacity self-driving-car dataset with real-time object detection.

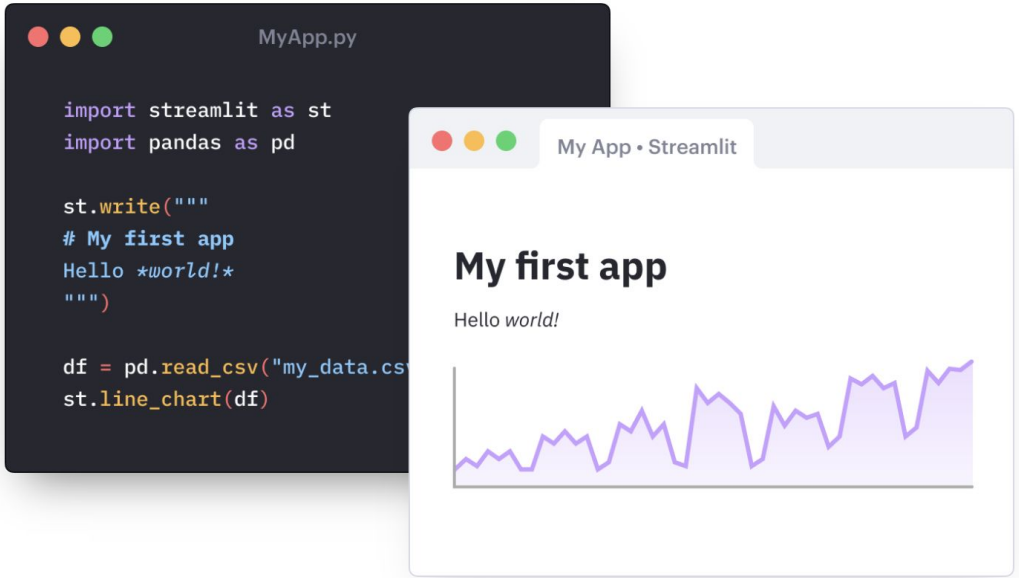
View source on GitHub

Next >

Streamlit combines three simple ideas

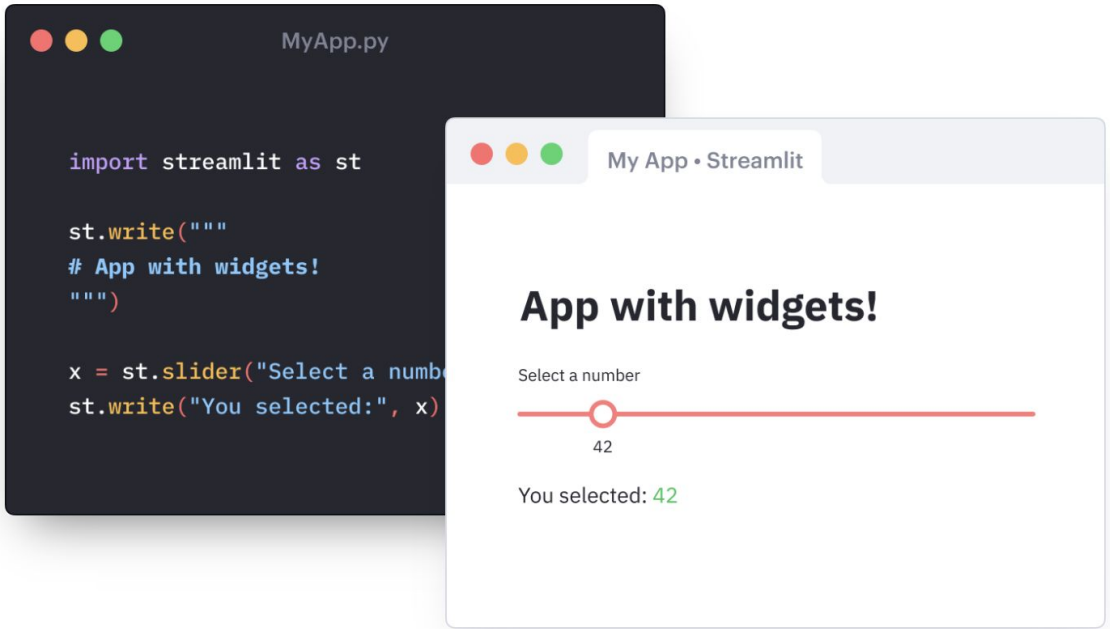
Embrace Python scripting

Build an app in a few lines of code with our magically simple API. Then see it automatically update as you save the source file.



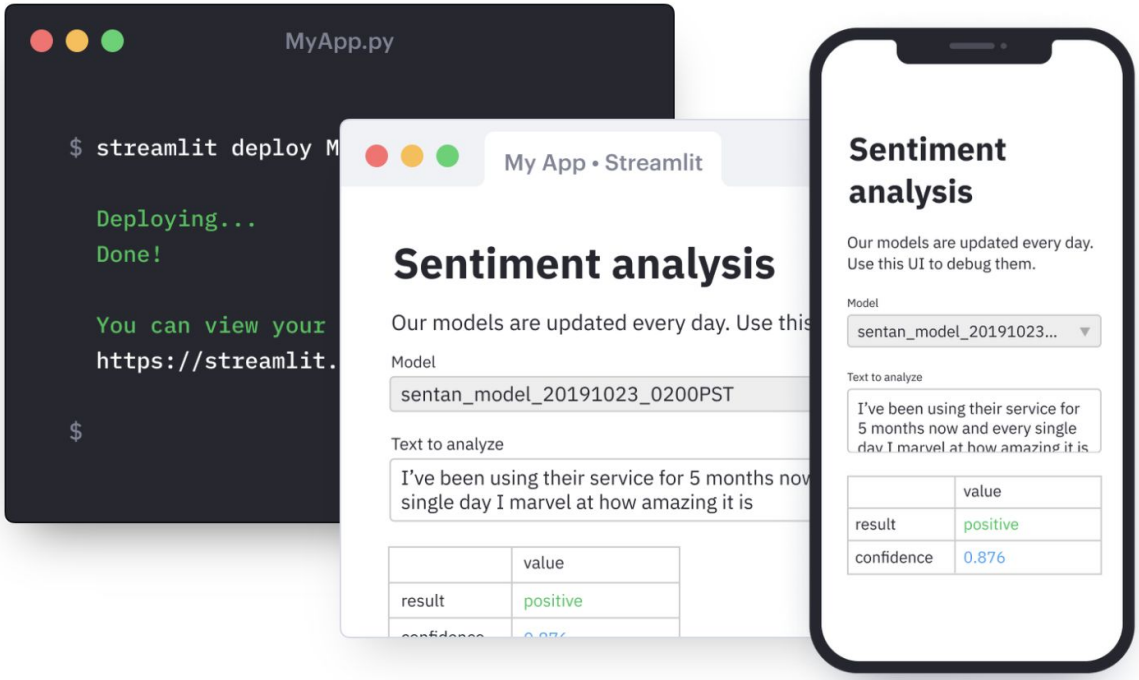
Weave in interaction

Adding a widget is the same as declaring a variable. No need to write a backend, define routes, handle HTTP requests, etc.



Deploy instantly

Host it yourself or use **Streamlit For Teams** to effortlessly deploy, manage, and collaborate on apps.
[Join the beta now!]

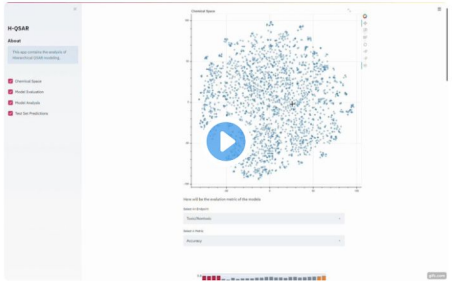


Compatible with
major libraries & frameworks

Xinhao Li
@XinhaoLi1

Replying to @XinhaoLi1 and 4 others

An interactive app to analyze the chemical space, model performances, and molecular predictions was made using @streamlit.

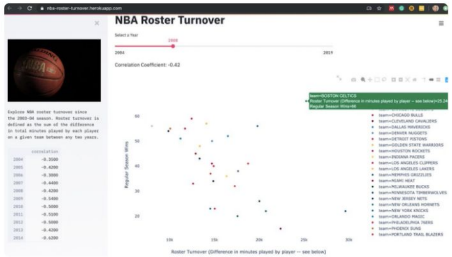


♡ 17 3:52 PM - Feb 6, 2020

See Xinhao Li's other Tweets

Kevin Arvai
@arvkevi

The @streamlit hype is real, this app went from zero to deployed in one night! #python #DataScience nba-roster-turnover.herokuapp.com

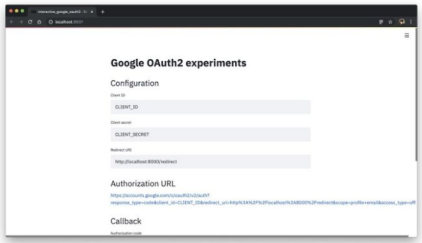


♡ 28 4:45 PM - Oct 18, 2019

16 people are talking about this

François Voron
@fvoron

Just discovered @streamlit! Amazing tool that can go beyond ML: I wrote an interactive #OAuth2 flow in minutes 👍
gist.github.com/frankie567/63d…

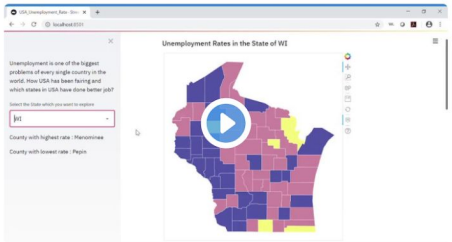


♡ 20 5:26 PM - Dec 12, 2019

See François Voron's other Tweets

Umesh
@dataasana

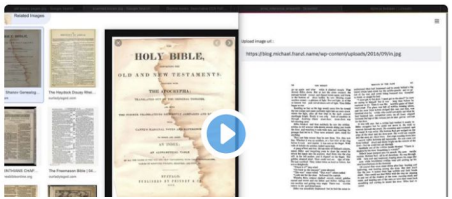
One of the best part of @streamlit is the ease with which we can build the interactive data apps.



♡ 14 5:57 PM - Feb 18, 2020

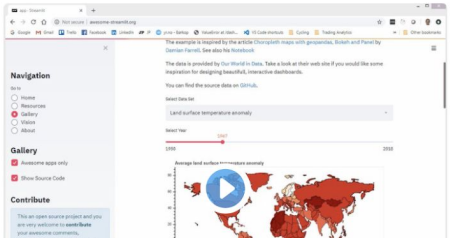
ajinkya bobade
@BobadeAjinkya

I have developed a Deep Learning app with @streamlit
to convert damaged documents to PDF with highest accuracy. The algorithm gives excellent results on par with existing technology. ** email me for public url of the demo: ajinkyabobade93@gmail.com



Marc Skov Madsen
@MarcSkovMadsen

I've added a Choropleth map to the gallery at awesome-streamlit.org and upgraded to Streamlit v. 0.52.2.#streamlit #data #models #analytics #python #machinelearning #datascience #apps



Part 3: Création d’une app de data viz en moins de 1h et 150 lignes de code...

https://github.com/opencovid19-fr/data

README.md

English

Données nationales concernant l'épidémie de COVID19

L'information officielle sur la progression de l'épidémie en France est assez fragmentée, et n'est presque jamais structurée sous forme de données.

L'objectif de ce dépôt est de consolider l'information officielle, et de la rendre disponible dans des formats ouverts et aisément réutilisables (JSON, CSV...).

Inutile de perdre du temps à écrire des scrappers, à ce stade il est plus efficace de r
source.

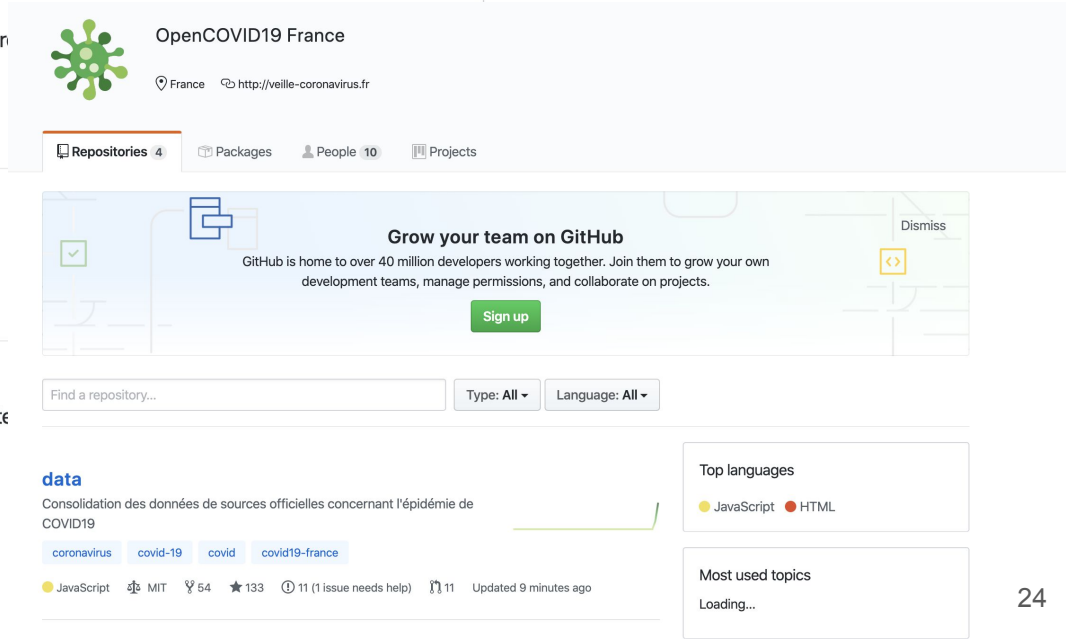
Données résultantes

- chiffres-cles.json
- chiffres-cles.csv

Sources utilisées

- Santé publique France
- Agences Régionales de Santé - Merci de prendre les issues ouvertes pour traite
fait en double.
- Préfectures
- Vidéos du Ministère des Solidarités et de la Santé

Fichiers sources



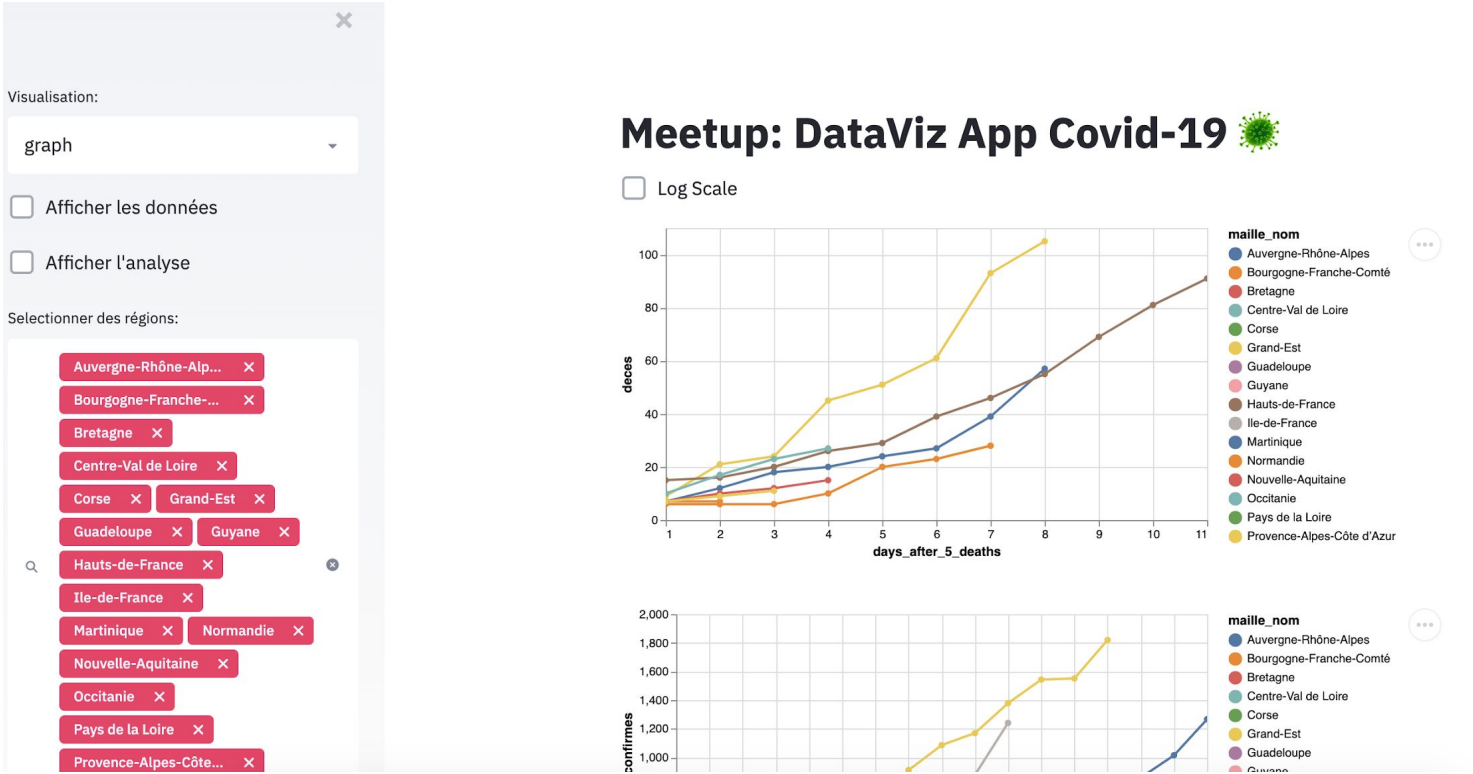
Bloc YAML par région ou département

Voici un exemple de bloc YAML pour une région ou un département:

```yaml
nom: region-ou-departement-exemple
code: Exemple
casConfirmes: 500
gueris: 40 # valeur copiée du fichier YAML précédent
deces: 10
hospitalises: 10
reanimation: 5
victimes:
  - age: 85
    date: 2020-03-10
    sexe: homme
  - sexe: femme
    date: 2020-03-10
  - date: 2020-03-10
```

Les champs `casConfirmes`, `gueris` et `deces` comptabilisent le total par catégorie depuis le début de la crise Covid-19. Par contre, les champs `hospitalises` et `reanimation` donnent le nombre de patient par catégorie à l'instant de l'édition du bulletin d'information, ces 2 chiffres peuvent bien sûr évoluer à la hausse ou à la baisse.

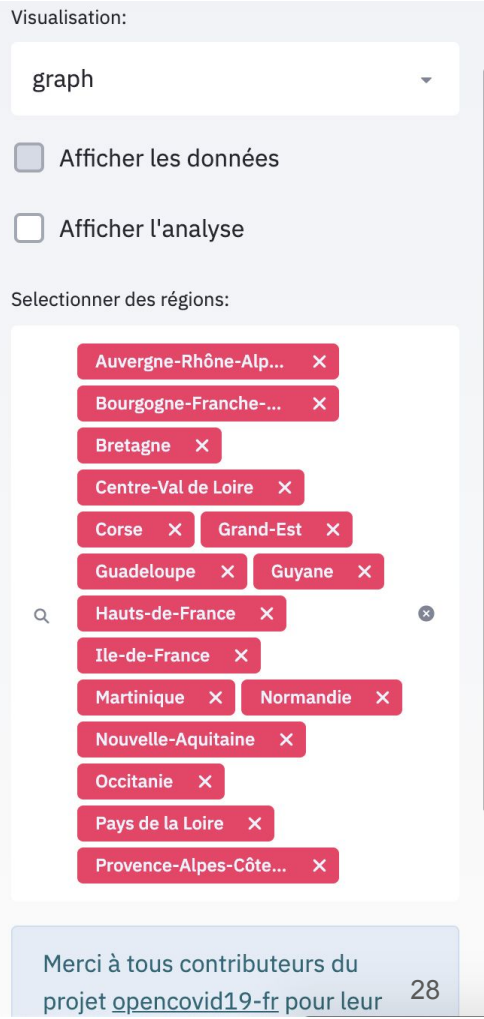
Live Démo



moins de 100 lignes de code ...

```python
import pandas as pd
from altair import datum
import streamlit as st
import altair as alt

st.title('Meetup: DataViz App Covid-19 🦠')

@st.cache
def get_data():
    url = 'https://raw.githubusercontent.com/opencovid19-fr/data/master/dist/chiffres-cles.csv'
    data = pd.read_csv(url)

    df_fr = data[(data.granularite == 'pays') & (data.source_type == 'sante-publique-france')]
    df_fr = pd.melt(df_fr, id_vars=['date'], value_vars=['cas_confirmes', 'deces', 'reanimation'], var_name='type',
                    value_name='nombre')

    df_fr = df_fr.fillna(value=0)

    df = data[(data.granularite == 'region') & (data.source_type == 'agences-regionales-sante')]
    df = df[["date", "maille_nom", "cas_confirmes", "deces"]]
    df = df.sort_values(by=['maille_nom', 'date'])
    df["delta_deces"] = df.groupby('maille_nom')['deces'].diff()
    df["delta_cas_confirmes"] = df.groupby('maille_nom')['cas_confirmes'].diff()
    df['fatality_rate'] = (df['deces'] / df['cas_confirmes'])
    df['days_after_5_deaths'] = df[df.deces > 5].groupby("maille_nom")['deces'].rank(method="first", ascending=True)
    df['days_after_50_confirmed'] = df[df.cas_confirmes > 50].groupby("maille_nom")['cas_confirmes'].rank(
        method="first", ascending=True)
    df.reset_index(drop=True)
    return df, df_fr


df, df_fr = get_data()
```

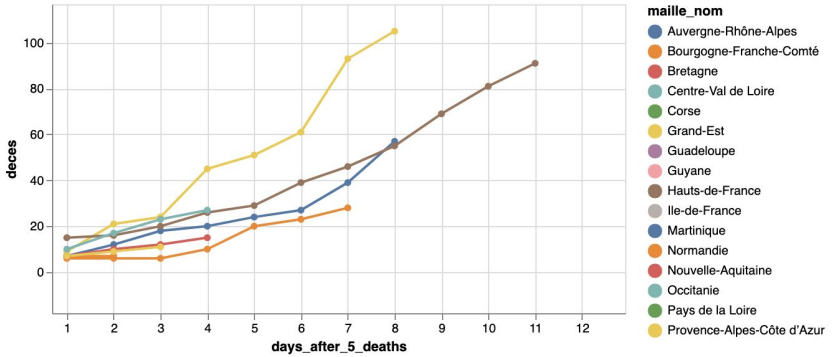
```
regions = list(df.maille_nom.unique())
option = st.sidebar.selectbox('Visualisation: ',('graph', 'heatmap', 'histo'))
check_box_table = st.sidebar.checkbox("Afficher les données")
check_box_analyse = st.sidebar.checkbox("Afficher l'analyse")

multiselection = st.sidebar.multiselect("Selectionner des régions:", regions, default=regions)
st.sidebar.info('Merci à tous contributeurs du projet [opencovid19-fr](https://github.com/opencovid1
```

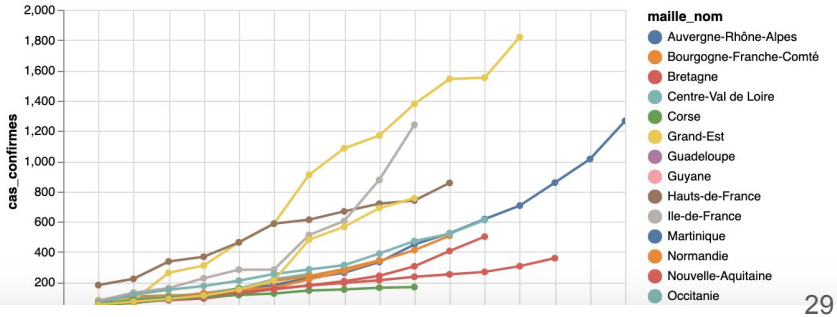


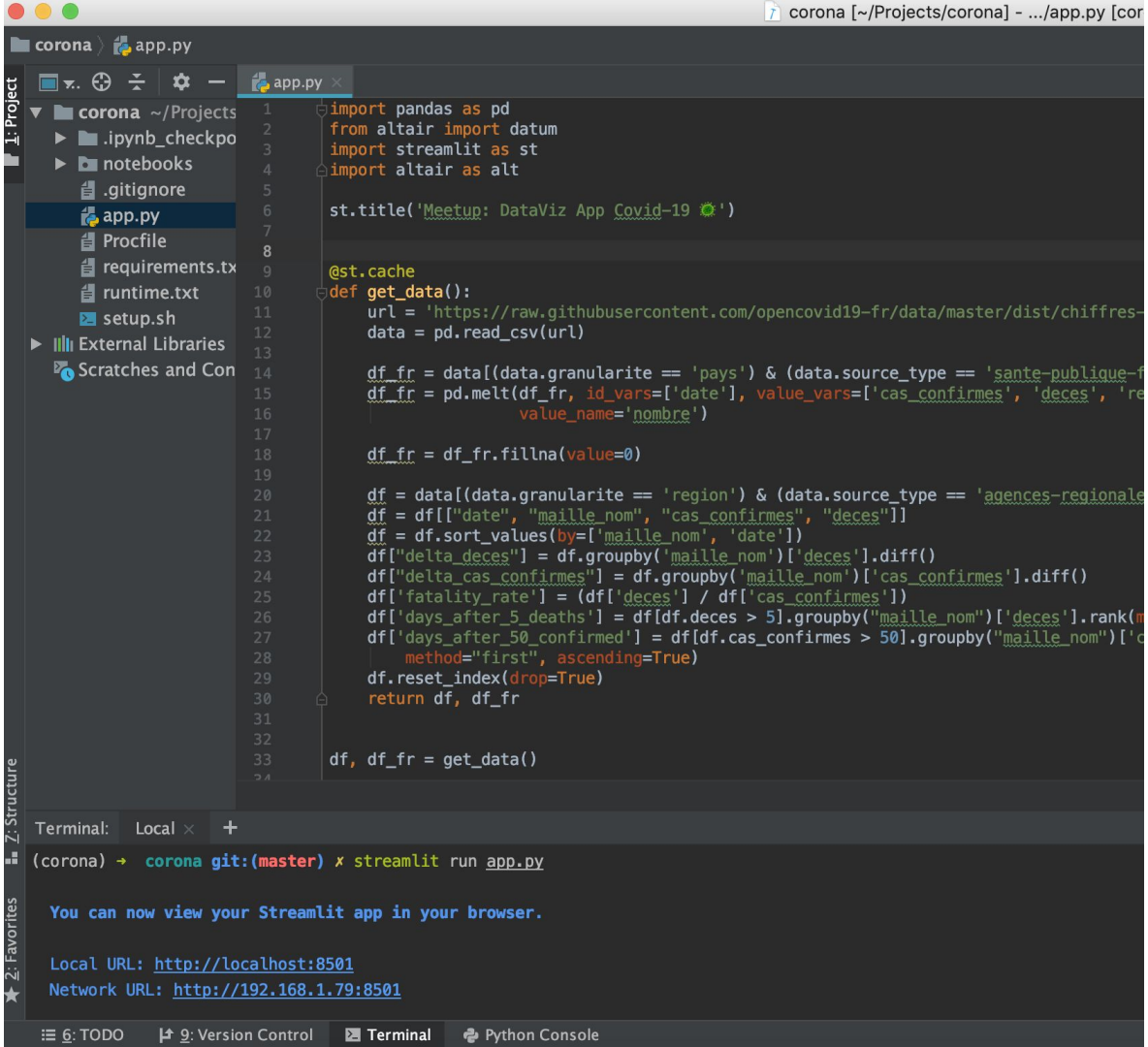
```python
if option =='graph':

    if st.checkbox("Log Scale"):
        scale = alt.Scale(type='log', domain=[10, 5000], clamp=True)
    else:
        scale = alt.Scale(type='linear')

    if check_box_analyse:
        st.info("[03/22] Les régions Grand-Est, Ile-de-France et Haut-de-France
                "Par ailleurs l'affiche en échelle Log, nous montre que l'ensemb

    c_deces = alt.Chart(df).mark_line(point=True).encode(
        alt.X('days_after_5_deaths'),
        alt.Y('deces', scale=scale),
        alt.Color('maille_nom'),
        tooltip=['days_after_5_deaths', 'deces', 'maille_nom']
    ).interactive()

    c_confirmed = alt.Chart(df).mark_line(point=True).encode(
        alt.X('days_after_50_confirmed'),
        alt.Y('cas_confirmes', scale=scale),
        alt.Color('maille_nom'),
        tooltip=['days_after_5_deaths', 'deces', 'maille_nom']
    ).interactive()

    st.altair_chart(c_deces, use_container_width=True)
    st.altair_chart(c_confirmed, use_container_width=True)
```

Meetup: DataViz App Covid-19

Log Scale






corona [~/Projects/corona] - .../app.py [cor
corona
app.py
1: Project
app.py
corona ~/Projects
.ipynb_checkpo
notebooks
.gitignore
app.py
Procfile
requirements.tx
runtime.txt
setup.sh
External Libraries
Scratches and Con
import pandas as pd
from altair import datum
import streamlit as st
import altair as alt
st.title('Meetup: DataViz App Covid-19 ')
@st.cache
def get_data():
url = 'https://raw.githubusercontent.com/opencovid19-fr/data/master/dist/chiffres-
data = pd.read_csv(url)
df_fr = data[(data.granularite == 'pays') & (data.source_type == 'sante-publique-f
df_fr = pd.melt(df_fr, id_vars=['date'], value_vars=['cas_confirmes', 'deces', 're
value_name='nombre')
df_fr = df_fr.fillna(value=0)
df = data[(data.granularite == 'region') & (data.source_type == 'agences-regionale
df = df[["date", "maille_nom", "cas_confirmes", "deces"]]
df = df.sort_values(by=['maille_nom', 'date'])
df["delta_deces"] = df.groupby('maille_nom')['deces'].diff()
df["delta_cas_confirmes"] = df.groupby('maille_nom')['cas_confirmes'].diff()
df['fatality_rate'] = (df['deces'] / df['cas_confirmes'])
df['days_after_5_deaths'] = df[df.deces > 5].groupby("maille_nom")['deces'].rank(m
df['days_after_50_confirmed'] = df[df.cas_confirmes > 50].groupby("maille_nom")['c
method="first", ascending=True)
df.reset_index(drop=True)
return df, df_fr
df, df_fr = get_data()
Terminal:
Local
(corona) → corona git:(master) ✗ streamlit run app.py
You can now view your Streamlit app in your browser.
Local URL: http://localhost:8501
Network URL: http://192.168.1.79:8501
7: Structure
2: Favorites
6: TODO
9: Version Control
Terminal
Python Console

Merci Pour Votre Écoute

Slide + Code: https://github.com/TlseDataEngineering/data-app-covid19