Fachhochschule Kiel
University of Applied Sciences

Data Visualization and Visual Analytics
Streamlit

Study Program Data Science
Prof. Dr. Tillmann Schwörer

Lecture Roadmap

Data Domains

Comparing Categories | Relationships | Geospatial | Time |
Part-to-whole | Distributions | Uncertainty | …

Storytelling

Perception + Visualization Design

Python + Tools

Interactive Visualization

Streamlit Overview

"**A faster way to build and share data apps**: Streamlit turns data scripts into shareable web apps in minutes. All in pure Python. No front-end experience required."

- **Easy-to-use**: no web development knowledge required
- **User interaction**: app updates in response to user input
- **Integration** : Pandas, Plotly, Matplotlib, Altair, Markdown, etc.
- **Deployment**: host and share via Streamlit Community Cloud

Streamlit Frontend

x-axis

sepal_length

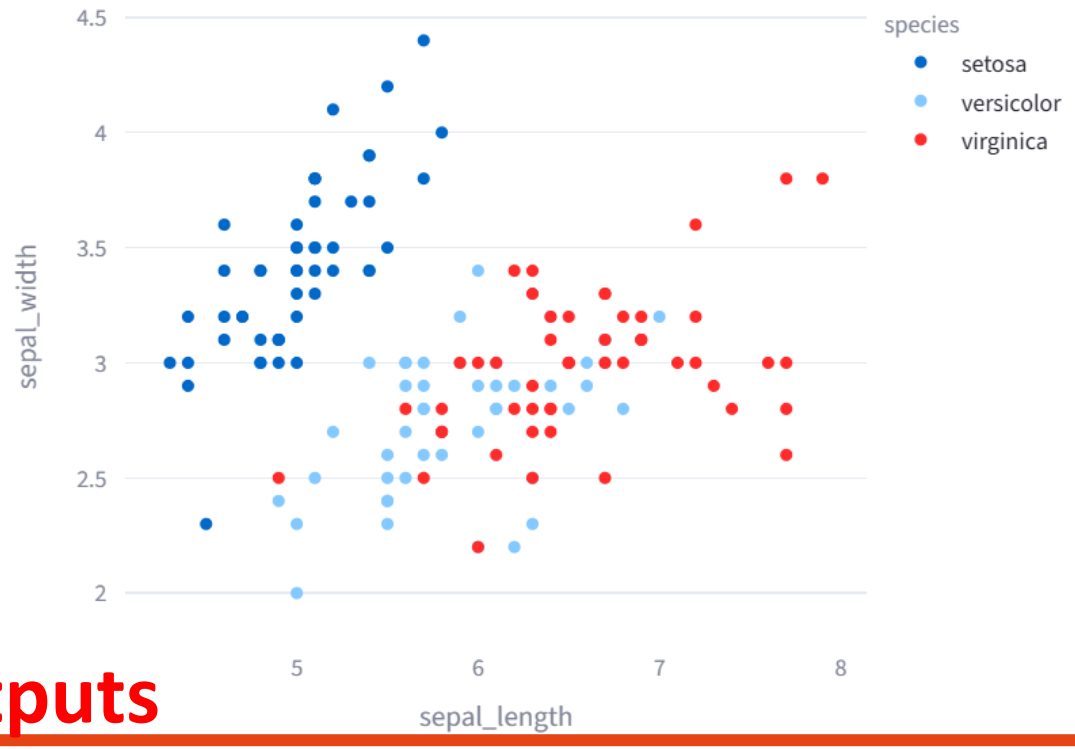
y-axis

sepal_width

Inputs

Layout

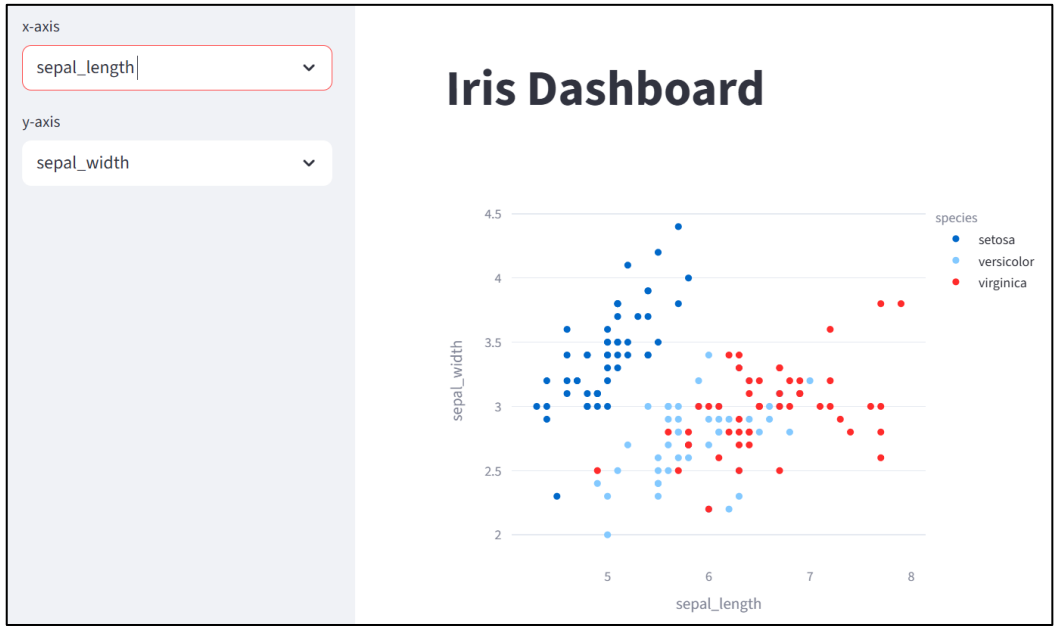
Iris Dashboard



Outputs

Streamlit Python Script

Browser Frontend



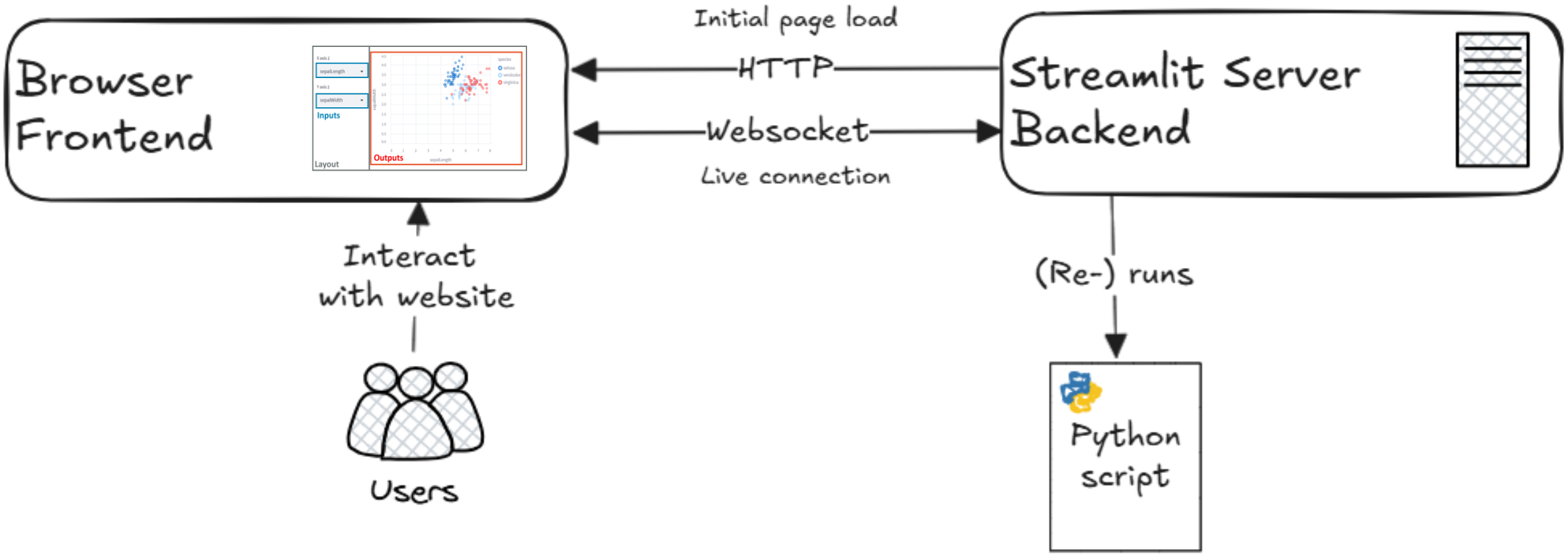
Backend

app.py

```
import streamlit as st
import pandas as pd
import plotly.express as px

df = px.data.iris()

st.title('Iris Dashboard')

cols = ["sepal_length","sepal_width","petal_length","petal_width"]
x = st.sidebar.selectbox(label="x-axis", options=cols, index=0)
y = st.sidebar.selectbox(label="y-axis", options=cols, index=1)

fig = px.scatter(df, x=x, y=y, color='species')
st.plotly_chart(fig)
```

streamlit run app.py

Architecture



App Model

- **Scripting:** Streamlit apps are Python scripts running from top to bottom
- **Inputs:** user interacts with input widget → rerun full (!) script
- **Outputs:** visualizations, images, text, data, …
- **Caching:** Scripts can cache results to avoid repeting expensive operations

Inputs

Radio

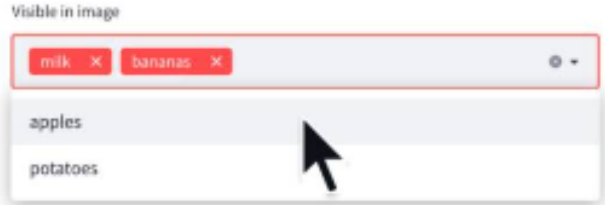
Display a radio button widget.

```
choice = st.radio("Pick one",
```

Selectbox

Display a select widget.

```
choice = st.selectbox("Pick or
```



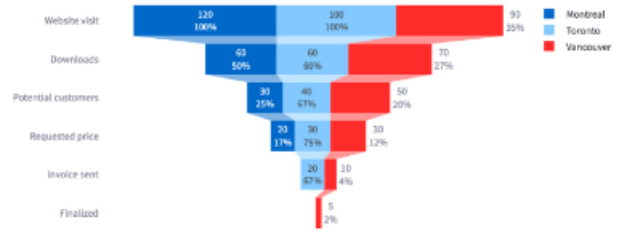
Multiselect

Display a multiselect widget. The multiselect widget starts as empty.

```
choices = st.multiselect("Buy'
```

More inputs: https://docs.streamlit.io/develop/api-reference/widgets

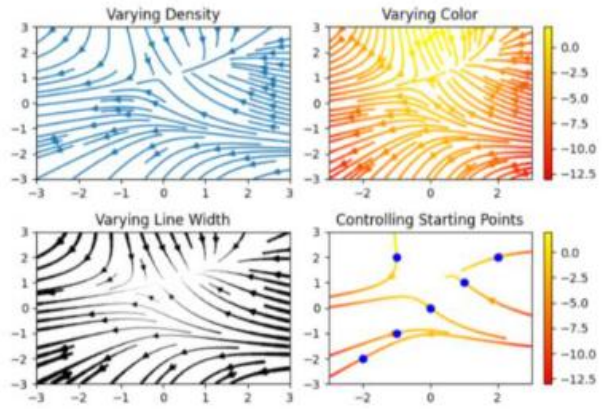
Charts outputs



Plotly

Display an interactive Plotly chart.

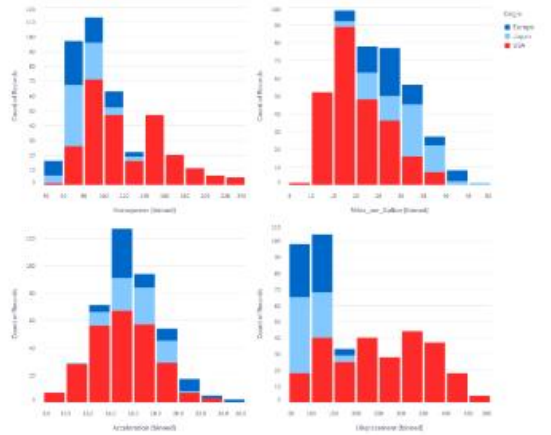
```
st.plotly_chart(my_plotly_chart)
```



Matplotlib

Display a matplotlib.pyplot figure.

```
st.pyplot(my_mpl_figure)
```



Altair

Display a chart using the Altair library.

```
st.altair_chart(my_altair_chart)
```

More chart outputs: https://docs.streamlit.io/develop/api-reference/charts

Data outputs

```
st.dataframe(iris)
st.json({'Name':'Till','City':'Kiel'})
st.metric('Temperature', '25°', 2)
```

	sepalLength	sepalWidth	petalLength	petalWidth	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5	3.6	1.4	0.2	setosa


```
{
  "Name" : "Till"
  "City" : "Kiel"
}
```


Temperature

25°

↑ 2

More data outputs: https://docs.streamlit.io/develop/api-reference/data

Text outputs

```
st.text('Fixed width text')
st.markdown('_Markdown_') # see *
st.latex(r''' e^{i\pi} + 1 = 0 ''')
st.write('Most objects') # df, err, func, keras!
st.write(['st', 'is <', 3]) # see *
st.title('My title')
st.header('My header')
st.subheader('My sub')
st.code('for i in range(8): foo()')
* optional kwarg unsafe_allow_html = True
```

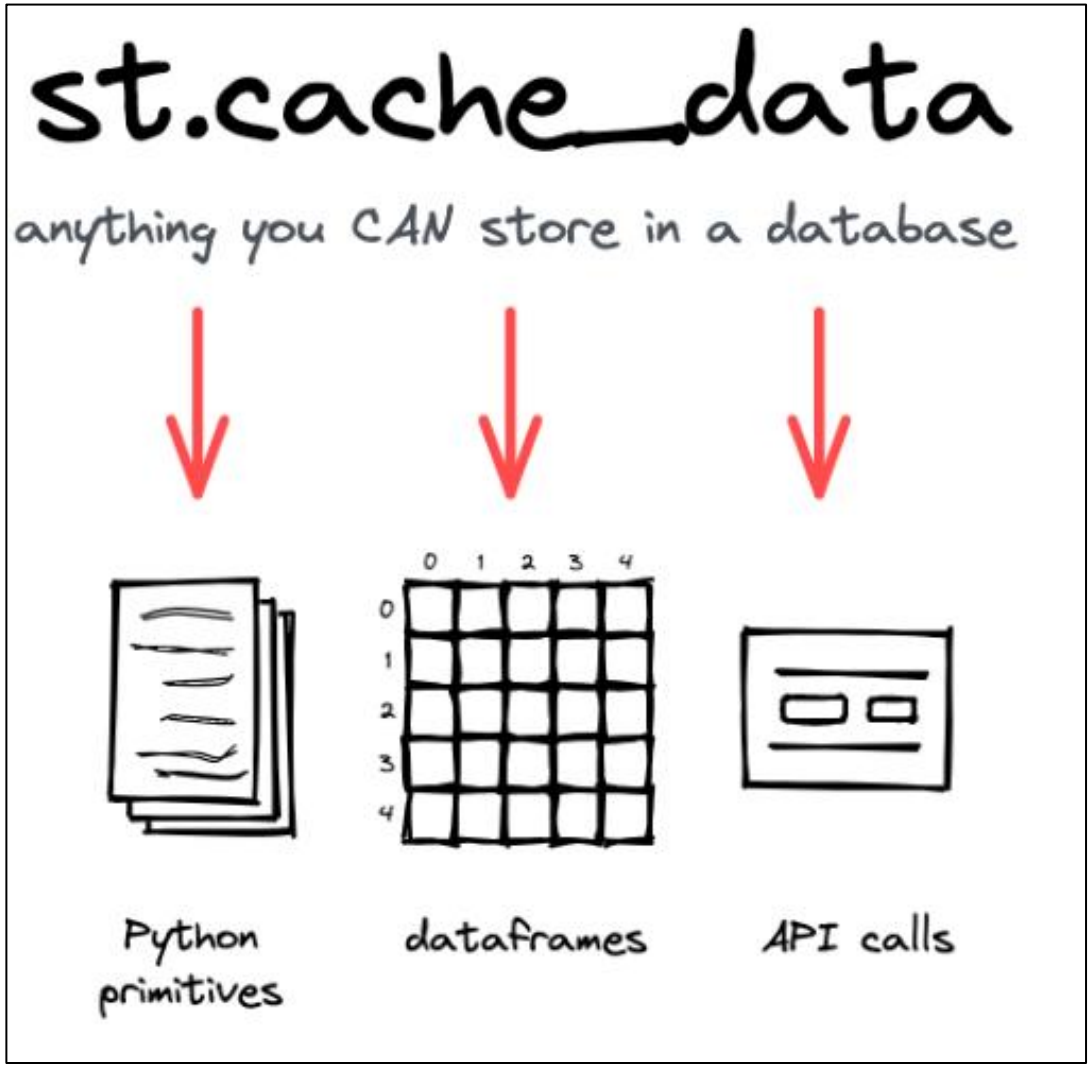
More text outputs: https://docs.streamlit.io/develop/api-reference/text

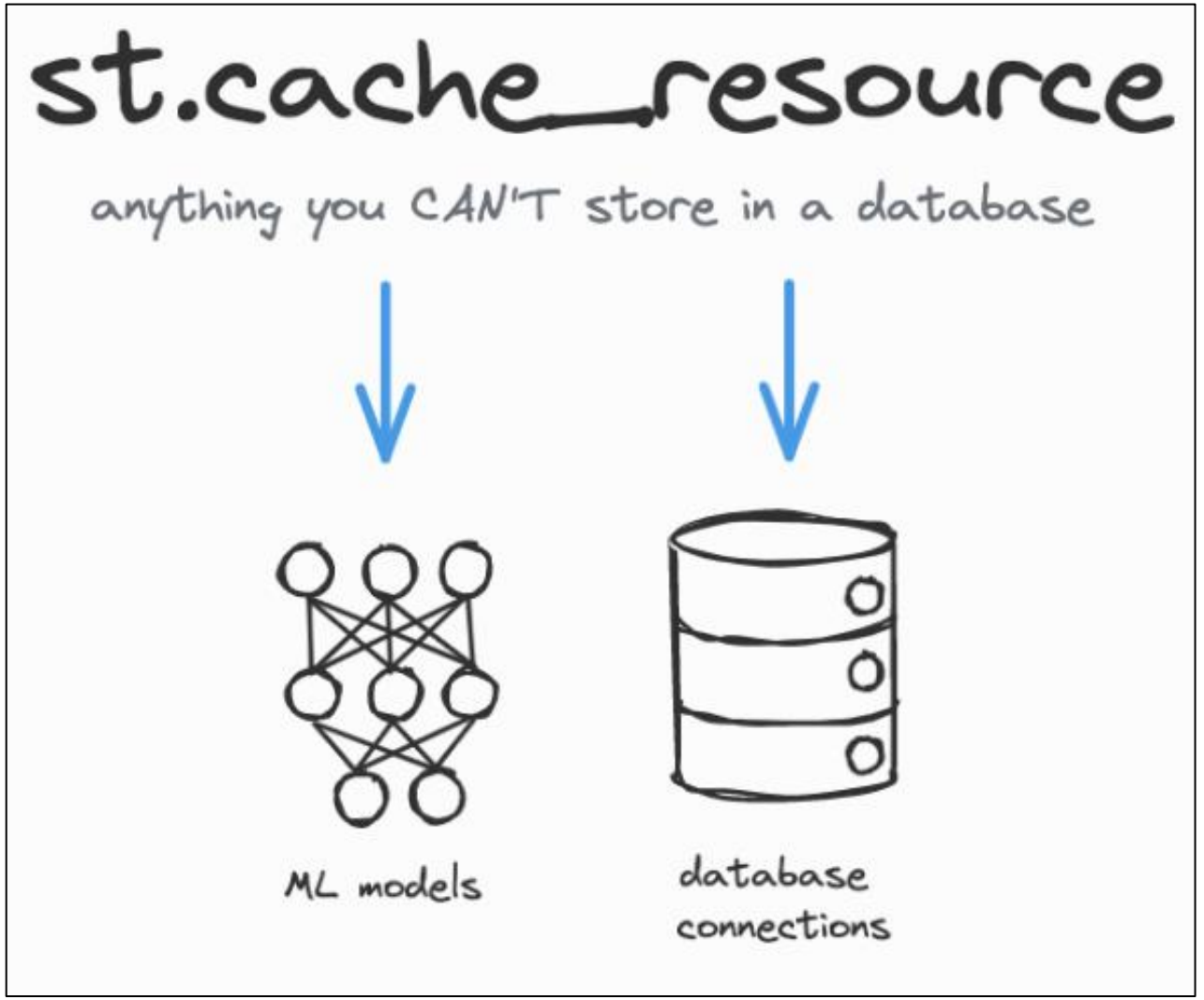
Caching

- Purpose: Avoid long running re-computations of static objects
- How:
  - Wrap long-running operation into a function
  - Preprend cache decorator

```
@st.cache_data
def long_running_function(param1, param2):
    return …
```

Caching


st.cache_data
anything you CAN store in a database
0 1 2 3 4
0
1
2
3
4
Python primitives
dataframes
API calls


st.cache_resource
anything you CAN'T store in a database
ML models
database connections

Layout

Sidebar

Display items in a sidebar.

```
st.sidebar.write("This lives in the
st.sidebar.button("Click me!")
```

Columns

Insert containers laid out as side-by-side columns.

```
col1, col2 = st.columns(2)
col1.write("this is column 1")
col2.write("this is column 2")
```

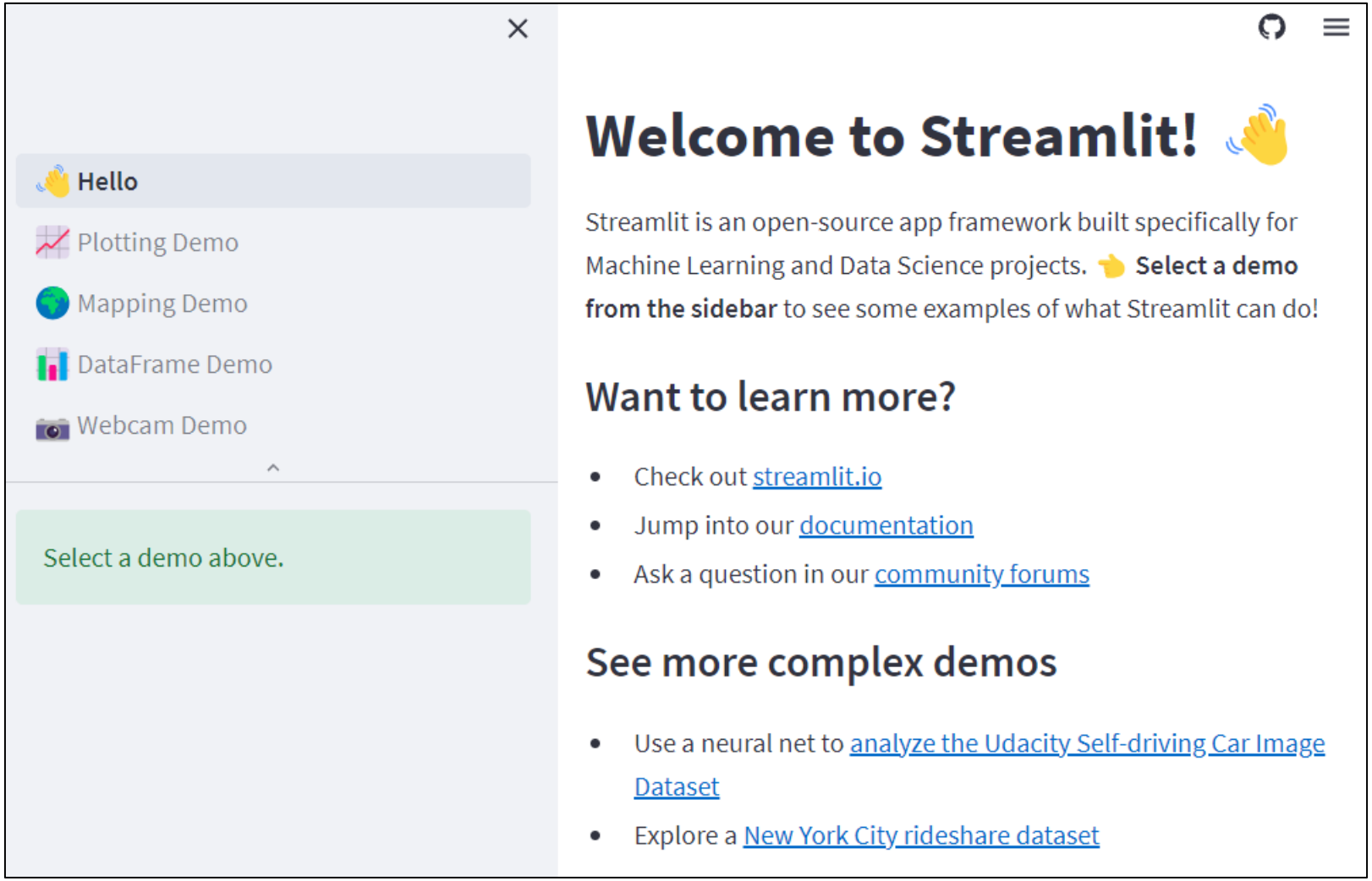
Tabs

Insert containers separated into tabs.

```
tab1, tab2 = st.tabs(["Tab 1", "Tab
tab1.write("this is tab 1")
tab2.write("this is tab 2")
```

Multipage Apps

- Entrypoint file: hello.py
- Each page is represented as a separate script and placed in the pages subdirecotry

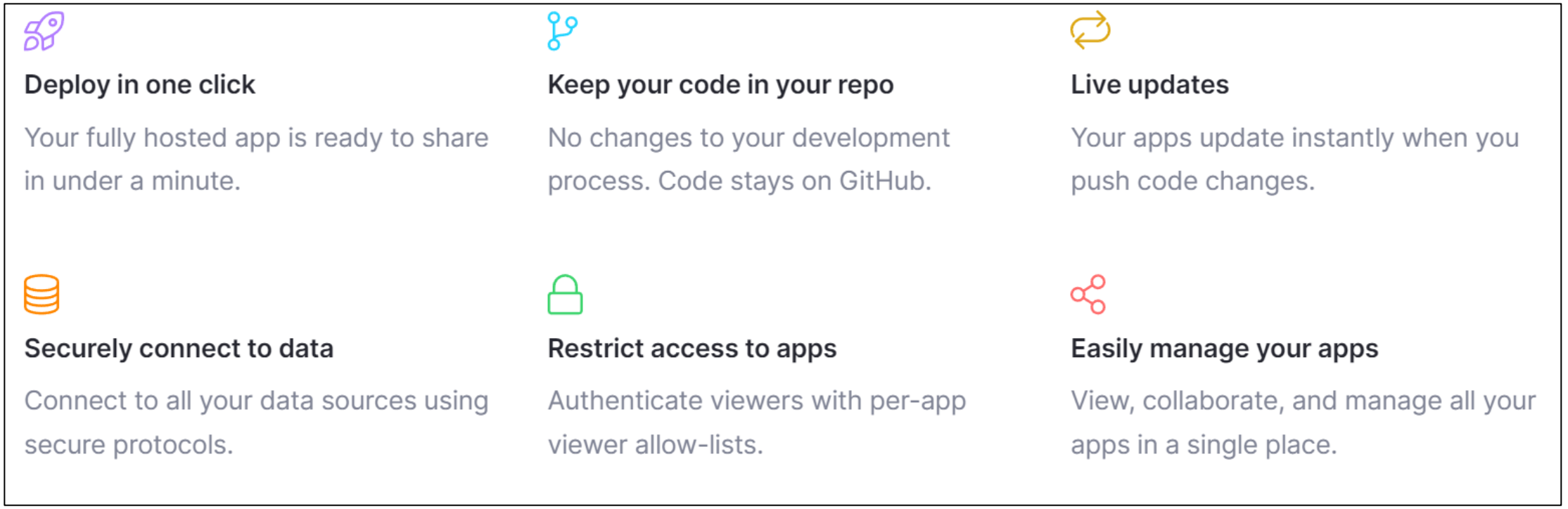


Page Configuration

```
st.set_page_config(
    page_title="Ex-stream-ly Cool App",
    page_icon="🧊",
    layout="wide",
    initial_sidebar_state="expanded",
    menu_items={
        'Get Help': 'https://www.extremelycoolapp.com/help',
        'Report a bug': "https://www.extremelycoolapp.com/bug",
        'About': "# This is a header. This is an *extremely* cool app!"
    }
)
```

App Deployment

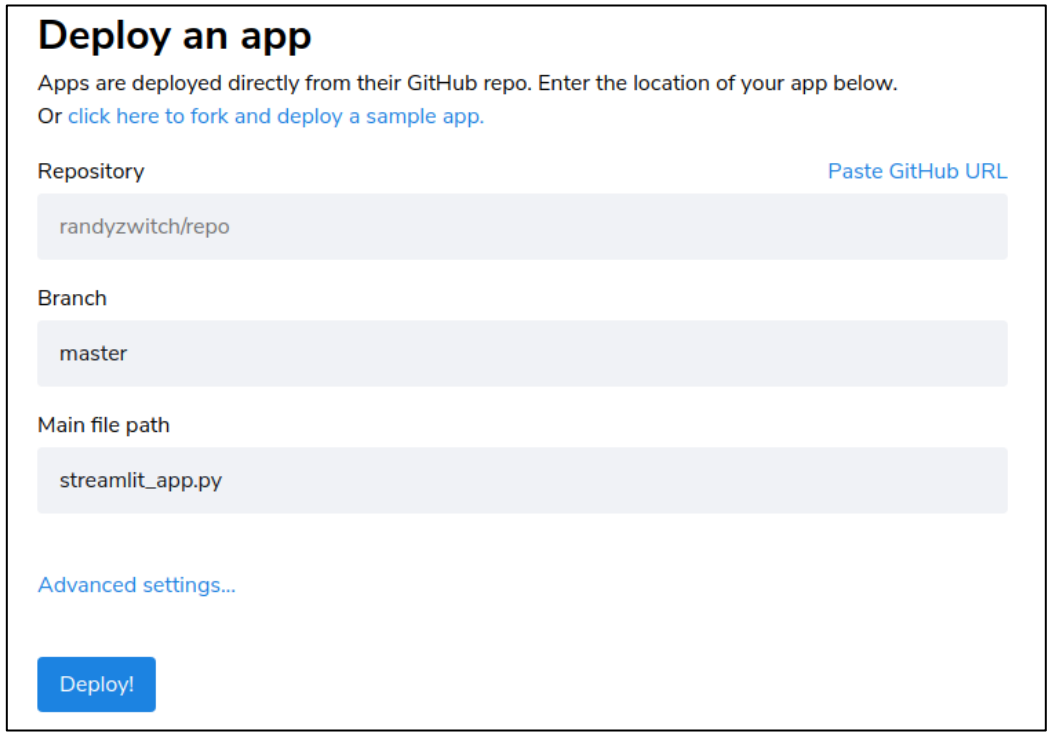
Apps can be easily hosted and shared via Streamlit Community Cloud



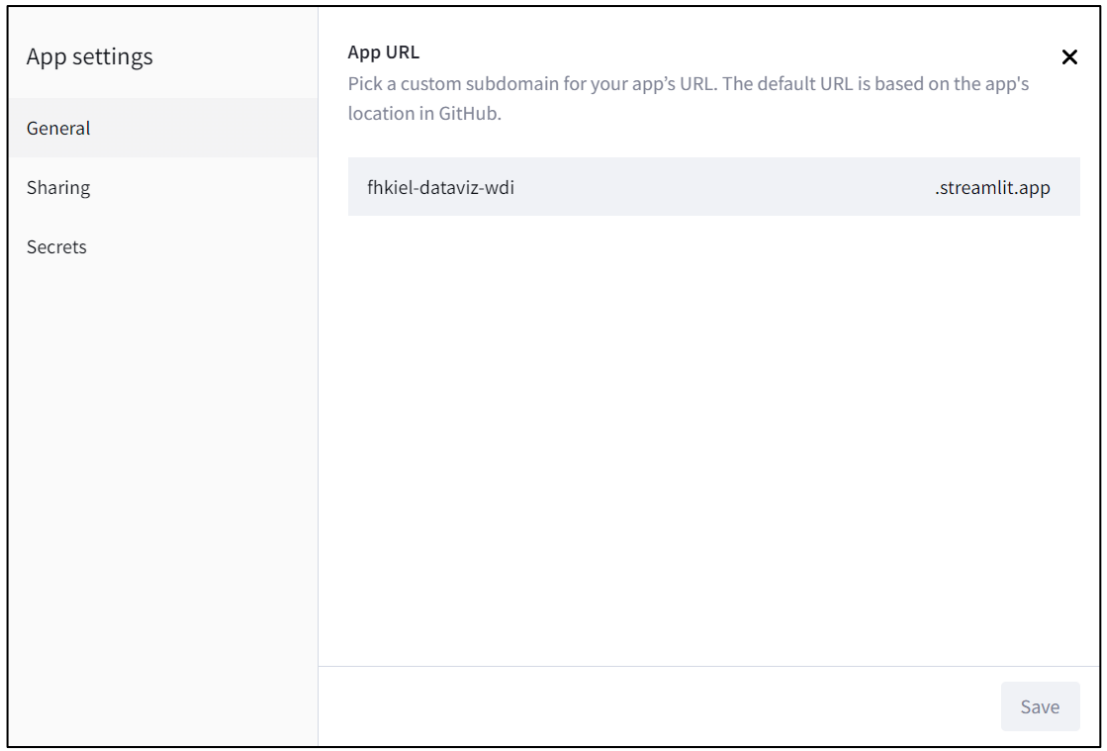
Source: https://streamlit.io/cloud

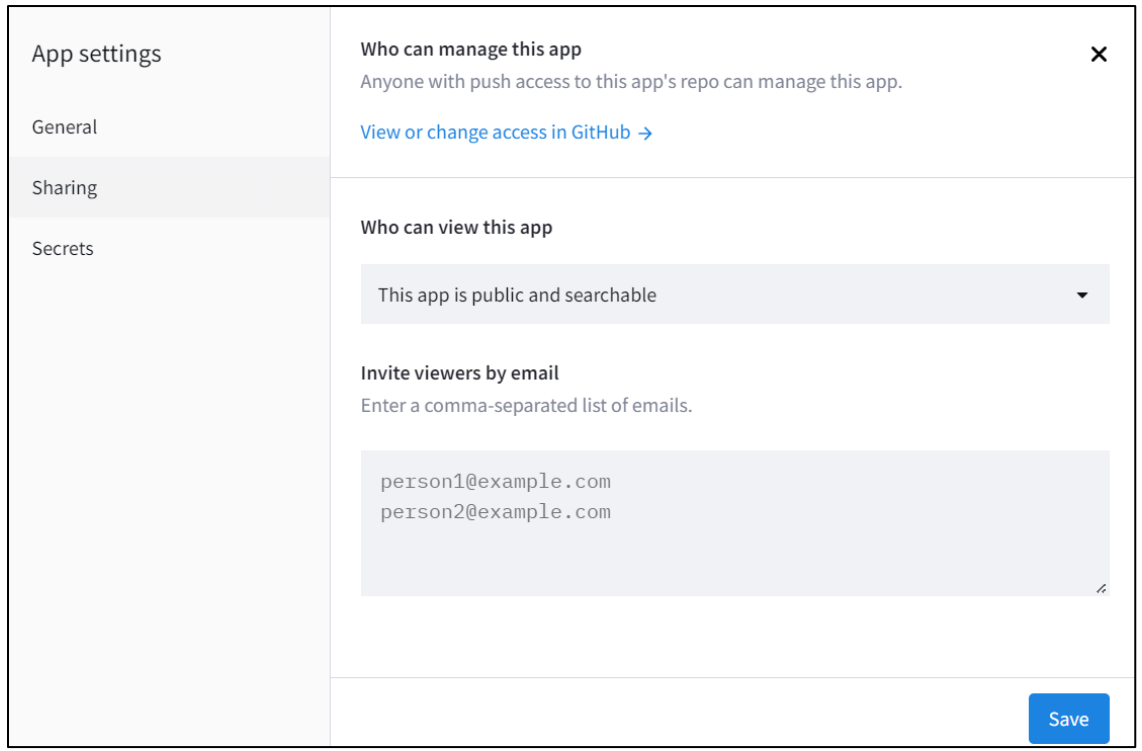
App Deployment Workflow

- **GitHub Repo:** Streamlit Community Cloud launches apps directly from your GitHub repo, so your app code and dependencies need to be on GitHub before you try to deploy the app.
- **Requirements**: Include a requirements.txt file for Python package dependencies
- **Deploy** on Streamlit Community Cloud



App Settings





Streamlit Resources

- **Cheat Sheet**: https://cheat-sheet.streamlit.app/
- **Gallery**: https://streamlit.io/gallery
- **Explore Streamlit Apps**: https://share.streamlit.io/explore
- **Third Party Extensions**: https://streamlit.io/components

Challenges of interactive visualization

- Risk of **overwhelming** the user with too much information
- **Little control** over what the viewer sees and interprets
- Bad user experience due to
  - Long **loading times**
  - **No data available** due to user choices
  - **Bad axis representation** due to user choices
  - …

Recommendations

- Focus on **user needs**
- **Use interactivity** sparingly and with purpose
- Optimize **performance**
- Ask **users** to test the application