DSS5201 Data Visualization

Weeks 12

Yuting Huang

NUS DSDS

2024-11-04

Week 12

- Graphs makeover.
- Interactive visualization and dashboards.

Week 13

- No class on Monday.
- 10-min Video & slides are due on **Wednesday 11:59pm (extended deadline)**.
- Notebook & rendered HTML are due on **Friday 11:59pm**.
- Only one submission is needed for each group.

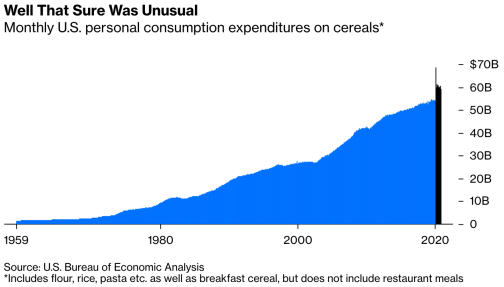
Graphs makeover (continued)

Create and improve the original visualization (Week 11 lecture notes).

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

Here is the original plot, from Bloomberg.

- The quality of the chart is quite good.
- We will re-create it and explore alternative ways to visualize the information.

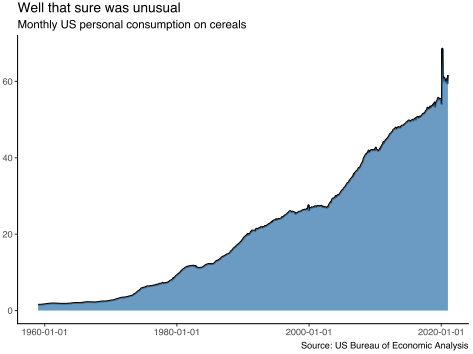


The data are available in **wk11_cereals.xlsx**.

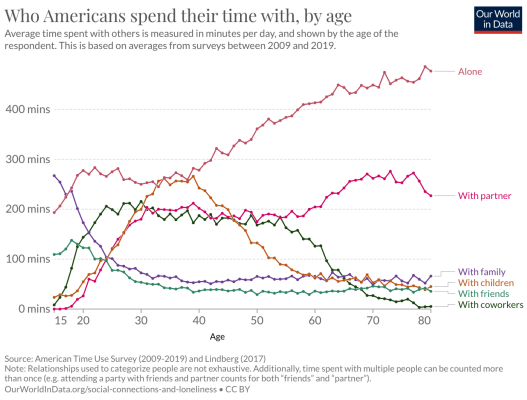
```
df3 = pd.read_excel("../data/wk11_cereals.xlsx")
df3 = df3.rename(columns = {df3.columns[3]: "exp"})
df3["exp"] = df3["exp"]/1000
df3["Month"] = pd.to_datetime(df3["Month"])
df3.head(3)
```

```
                      Category Sub-Category      Month    exp
0  Cereals and bakery products      Cereals 1976-01-01  6.494
1  Cereals and bakery products      Cereals 1976-02-01  6.411
2  Cereals and bakery products      Cereals 1976-03-01  6.437
```

```
(ggplot(df3, aes(x = "Month", y = "exp")) +
     geom_line(size = 1) +
     geom_area(fill = "steelblue", alpha = 0.8) +
     labs(title = "Well that sure was unusual", x = "", y = "",
          subtitle = "Monthly US personal consumption on cereals",
          caption = "Source: US Bureau of Economic Analysis") +
    theme_classic())
```



Here is the original plot, from Our World Data.



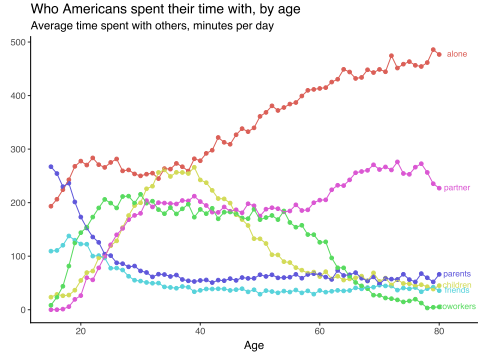
```
df4 = pd.read_csv("../data/wk11_time-spent.csv")
df4.columns = ["Entity", "Code", "Age", "alone", "friends",
               "children", "parents", "partner", "coworkers"]
df4 = df4.query("Age <= 80")
df4.head()
```

```
          Entity Code  Age  ...    parents    partner   coworkers
0  United States  USA   15  ...  267.12091   0.000000    8.342007
1  United States  USA   16  ...  254.33810   0.030691   23.529137
2  United States  USA   17  ...  229.81561   1.122841   43.809685
3  United States  USA   18  ...  236.35201   5.697267   81.633575
4  United States  USA   19  ...  201.27660  19.341291  124.850520

[5 rows x 9 columns]
```

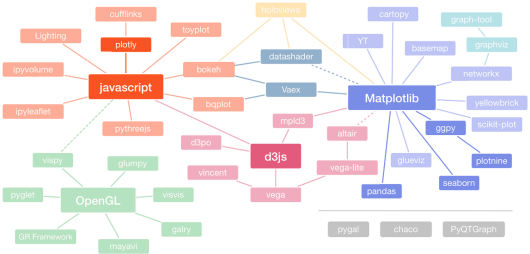
```
df4 = pd.melt(df4, id_vars = ["Age"],
              value_vars = ["alone", "friends", "children",
                            "parents", "partner", "coworkers"],
              var_name = "Category", value_name = "Time")
df4_text = df4.query("Age == 80")
df4_text.head()
```

	Age	Category	Time
65	80	alone	476.767360
131	80	friends	35.589745
197	80	children	45.010258
263	80	parents	66.100365
329	80	partner	227.046340

```
(ggplot(df4, aes(x = "Age", y = "Time", color = "Category")) +
     geom_point() + geom_line() +
     geom_text(df4_text, aes(label = "Category"), nudge_x = 3, size = 8) +
     labs(x = "Age", y = "",
          title = "Who Americans spent their time with, by age",
          subtitle = "Average time spent with others, minutes per day") +
     theme_classic() + theme(legend_position = "none"))
```



Interactive visualization with plotly



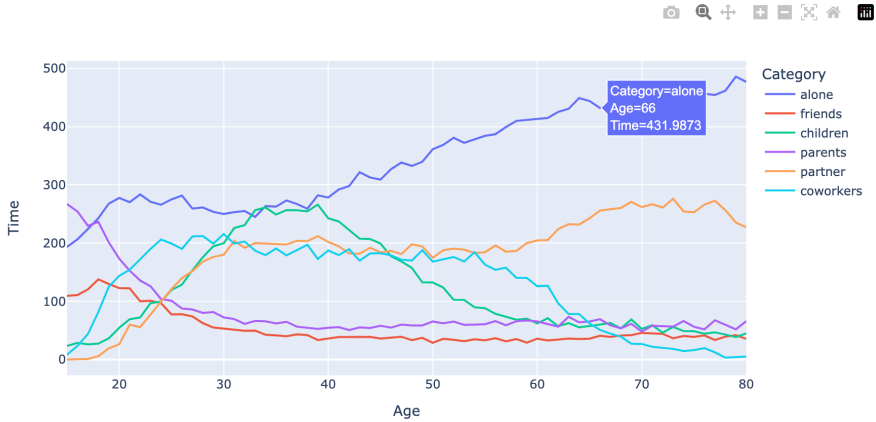
`Plotly` is a another popular library for **interactive graphs** with Python.

- Based on the famous d3.js javascript library.
- Install it via `pip install plotly` .
- We will be using the `plotly.express` module in this lecture.

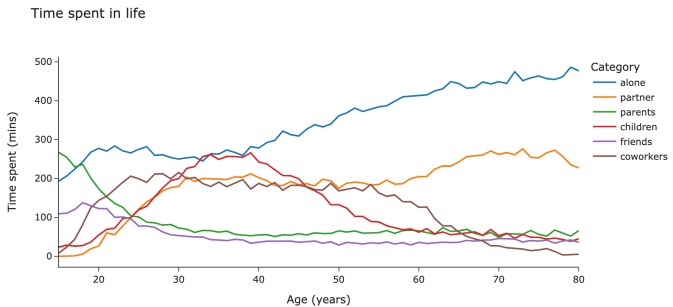
```
import plotly.express as px
```

Line plot with plotly

```
fig = px.line(df4, x = "Age", y = "Time", color = "Category",
              title = "Time spent in life")
fig.show()
```



```
fig = px.line(df4, x = "Age", y = "Time", color = "Category",
              title = "Time spent in life",
              labels={"Age": "Age (years)", "Time": "Time spent (mins)"},
              category_orders = {
                  "Category": ["alone", "partner", "parents",
                               "children", "friends", "coworkers"]},
              template = "simple_white")
fig.show()
```

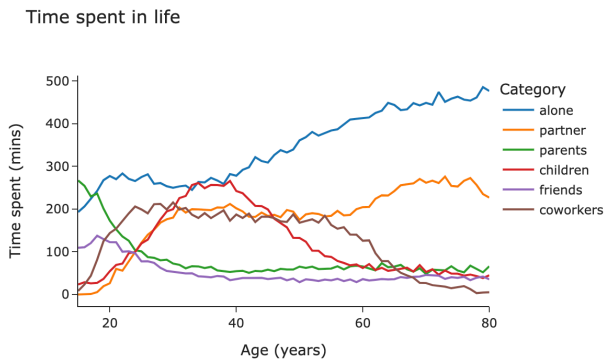


We can configure the styling options with `plotly.express.defaults`.

- The values we set will be used for the rest of the notebook.

```
px.defaults.template = "simple_white"
px.defaults.color_continuous_scale = px.colors.sequential.Blues
px.defaults.color_discrete_sequence = px.colors.qualitative.D3
px.defaults.width = 600
px.defaults.height = 400
```

```
fig = px.line(df4, x = "Age", y = "Time", color = "Category",
              title = "Time spent in life",
              labels={"Age": "Age (years)", "Time": "Time spent (mins)"},
              category_orders = {
                  "Category": ["alone", "partner", "parents",
                               "children", "friends", "coworkers"]})
fig.show()
```



Next, we introduce how to visualize geo-spatial data with `plotly`.

We need two components:

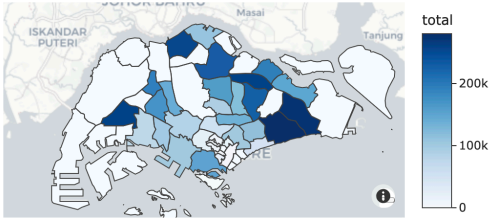
- A `geoJSON` file with geometry information for each location.
- A data frame with values to be superimposed across locations.

```
import geopandas as gpd
sg = gpd.read_file("../data/wk11_MasterPlan2019.geojson")
sg_pop = pd.read_csv("../data/wk11_sg_pop.csv")
sg_pop = sg_pop[sg_pop["Time"] == 2023]
```

Distribution of population in 2023

```
px.choropleth_mapbox(
    geojson = sg, data_frame = sg_pop, color = "total",
    featureidkey = "properties.town", locations = "PA",
    mapbox_style = "carto-positron",
    center = {"lat": 1.35, "lon": 103.82}, zoom = 9,
    title = "Singapore population distribution by region")
```

Singapore population distribution by region



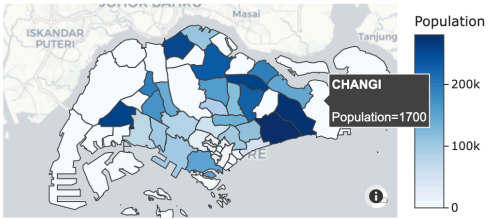
	town	geometry
0	BEDOK	MULTIPOLYGON (((103.93208 1.30555, 103.93208 1...
1	BOON LAY	MULTIPOLYGON (((103.72042 1.32824, 103.72003 1...
2	BUKIT BATOK	MULTIPOLYGON (((103.76408 1.37001, 103.76444 1...
3	BUKIT MERAH	MULTIPOLYGON (((103.82361 1.26018, 103.82362 1...
4	BUKIT PANJANG	MULTIPOLYGON (((103.77445 1.39029, 103.77499 1...

	PA	Time	total	change
22	ANG MO KIO	2023	161910	10.0
45	BEDOK	2023	280230	1260.0
68	BISHAN	2023	88340	2400.0
91	BOON LAY	2023	0	0.0
114	BUKIT BATOK	2023	169360	4080.0

- `featureidkey = "properties.town"` specifies the key column in the GeoJson file that identifies each region. This is then matched to another column in `locations`.
- `locations = "PA"` allows `plotly` to match the values in `town` based on `PA`.
- `mapbox_style` specifies the style of the map.

```
px.choropleth_mapbox(
    geojson = sg, data_frame = sg_pop, color = "total",
    featureidkey = "properties.town", locations = "PA",
    mapbox_style = "carto-positron",
    center = {"lat": 1.35, "lon": 103.82}, zoom = 9,
    title = "Singapore population distribution by region",
    hover_name = "PA", hover_data = {"PA": False},
    labels = {"total": "Population"})
```

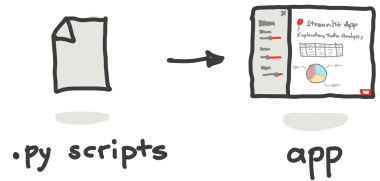
Singapore population distribution by region



So far, we've covered static and interactive visualizations using python.

- `Matplotlib`: The foundation of almost all visualizations in python.
- `Seaborn`: Built on `matplotlib` and simplifies the syntax.
- `Plotnine`: Inspired by the grammar of graphics. Intuitive for creating layered plots and useful for those familiar with `ggplot2` syntax in `R`.
- `Lets-plot`: Adopts a `plotnine`-style syntax and allows us to build interactive and layered plots.
- `Plotly`: A popular library for interactive graph. Allows flexible and advanced customization options.

Web applications (dashboards)



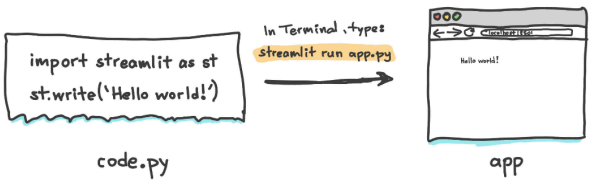
In this week, we will learn the basics about **web applications**.

There are a few tools for dashboarding in Python.

- `streamlit`: The simplest and easiest dashboard option. Does not require front-end experience.
- `Dash`: Comes with a steeper learning curve than `streamlit`, but offers more customizations in return.

There are several templates and applications created by the community:
https://streamlit.io/gallery?category=data-visualization

- Backgroud removal
- Covid-19 situation report
- 30 days of streamlit.



1. Build an app with a few lines of code.
   - Start of as a Pythons cript file (`.py`).
2. Add widgets (a button, a drop-down list, etc).
   - Input and output widgets.
3. Share, manage, and deploy the apps, directly from `streamlit`. For free!

Build a web application with `streamlit`

- After installing the package with pip install streamlit, we shall get start with a simple app
- Create a new **Python script** (`wk12_dashboard.py`) and import necessary libraries.

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

Next, we will define the default page configuration:

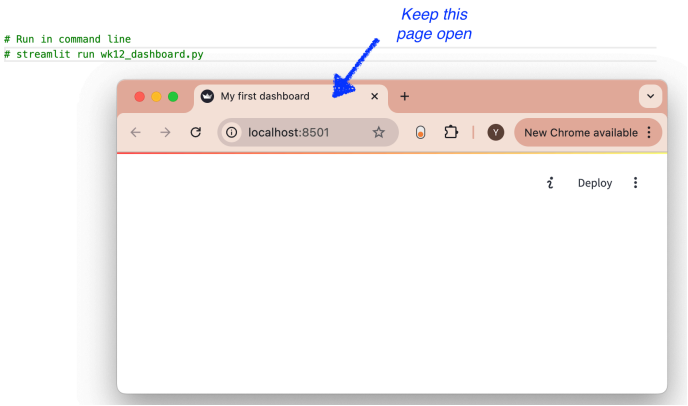
- A page title to be displayed on the browser.
- Define a page layout that fits the page's width.

```
st.set_page_config(
    page_title = "My first dashboard",
    layout = "wide")
```

- Note that we should only define the configuration once.

Now we can run the app by the following command in the Terminal:

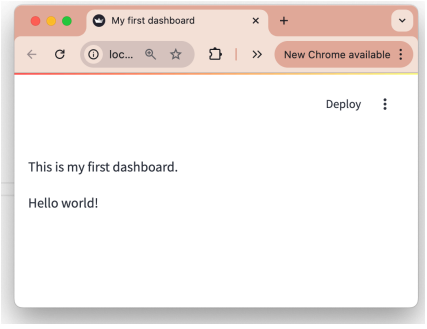
```
streamlit run wk12_dashboard.py
```



We can use `st.write()` to add anything to a app. Here we add two string statements.

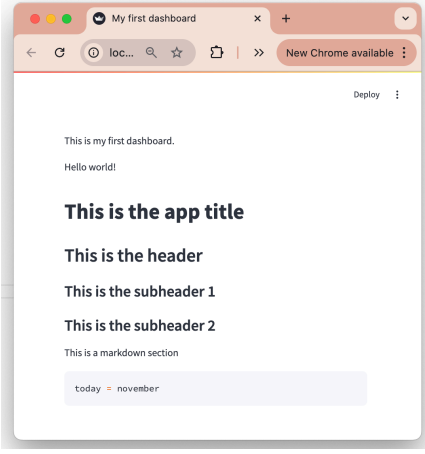
```
st.write("This is my first dashboard.")
st.write("Hello world!")
```

- Save the python script and go back to the app.
- The top panel will show *Source file changed*.
- Click *Rerun*.



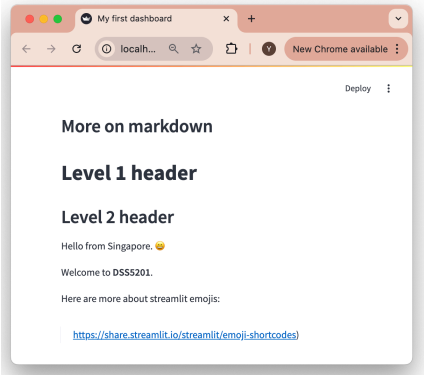
There are other functions for displaying texts:

```
st.title("This is the app title")
st.header("This is the header")
st.subheader("This is the subheader 1")
st.subheader("This is the subheader 2")
st.markdown("This is a markdown section")
st.code("today = november")
```



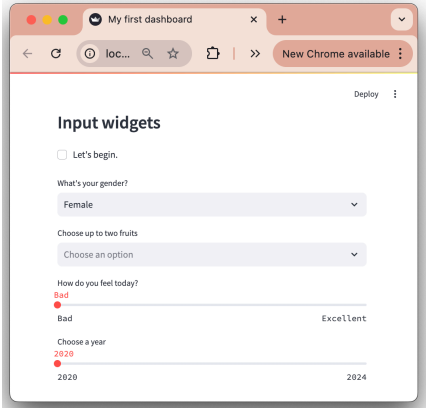
We can insert texts formatted according to the **markdown** language.

```
st.header("More on markdown")

st.markdown("# Level 1 header")
st.markdown("## Level 2 header")
st.markdown("Hello from Singapore :smile:")
st.markdown("Welcome to **DSS5201**.")


st.markdown("Here are more about streamlit emojis:")
st.markdown("> https://share.streamlit.io/streamlit/emoji-shortcodes)")
```



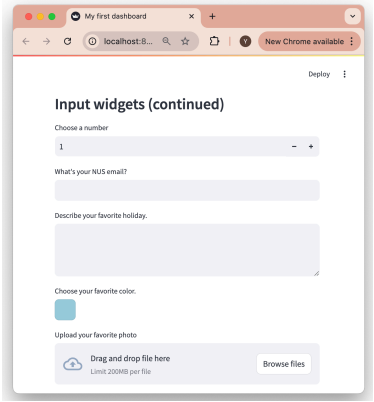
Widgets are one of the most important user interface components.

```
st.header("Input widgets")

st.checkbox("Let's begin.")
st.selectbox("What's your gender?",
             ["Female", "Male",
              "Prefer not to say"])
st.multiselect("Choose up to two fruits",
               ["Apple", "Melon", "Orange"],
               max_selections = 2)
st.select_slider("How do you feel today?",
              ["Bad", "Okay", "Excellent"])
st.slider("Choose a year", 2020, 2024)
```



Widgets are one of the most important user interface components.

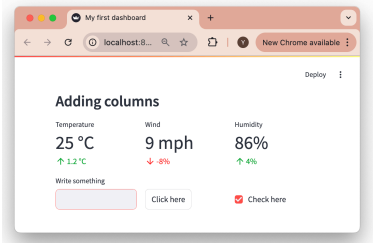
```
st.header("Input widgets (continued)")

st.number_input("Choose a number", 0, 9)
st.text_input("What's your NUS email?")
st.text_area("Describe your favorite holiday."
st.color_picker("Choose your favorite color.")
st.file_uploader("Upload your favorite photo")
```



Adding columns

We can create side-by-side columns with `st.columns()`.

```
st.header("Adding columns")

col1, col2, col3 = st.columns(3)
col1.metric("Temperature", "25°C", "1.2°C")
col2.metric("Wind", "9 mph", "-8%")
col3.metric("Humidity", "86%", "4%")

col1, col2, col3 = st.columns(3, vertical_alignment = "bottom")
col1.text_input("Write something here")
col2.button("Click here")
col3.checkbox("Check here")
```

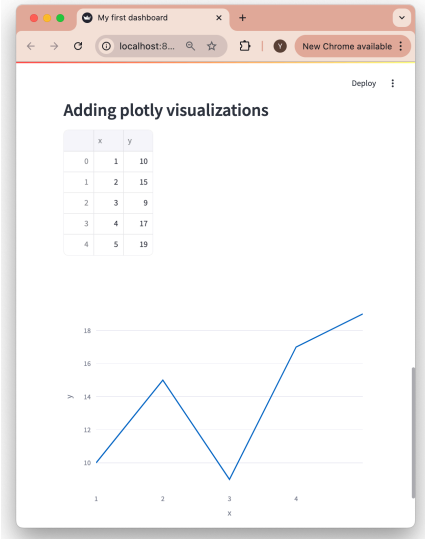


Displaying graphs

Let's continue to include visualizations.

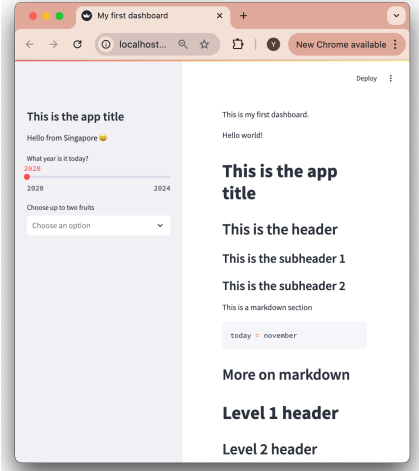
- `Streamlit` uses `st.dataframe()` to display data frames.
- `st.plotly_chart()` can embed `plotly` visualizations into an app.

```
st.header("Adding plotly visualizations")

df = pd.DataFrame({
    "x": [1, 2, 3, 4, 5],
    "y": [10, 15, 9, 17, 19]})
st.dataframe(df)
fig = px.line(df, x = "x", y = "y")
st.plotly_chart(fig)
```



We can organize the widgets into a sidebar, using the `with` notation.

```
with st.sidebar:
  st.title("This is the app title")
  st.markdown("Hello from Singapore :smile:")
  st.slider("What year is it today?",
            2020, 2024)
  st.multiselect("Choose up to two fruits",
                 ["Apple", "Melon", "Orange"]
                 max_selections = 2))
```



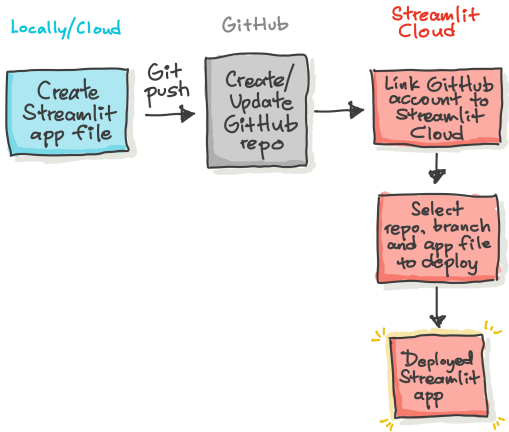
The true value of a dashboard can be unlocked if other people can access it too.

- ... host the dashboard on a server, for others to access it via a specific URL.
- Deployment solutions: `Streamlit Cloud`, Amazon Web Services, Google App Engine, Azure, ...

Typically, the process with `Streamlit Cloud` involves

1. Create an app with `streamlit`
2. Create an account with `streamlit`;
3. Create a GitHub repository.
4. Upload the application source code on GitHub.
5. Link `streamlit` to the GitHub repository and deploy the app.

For a video walk through, check out this video tutorial.



Source of image: Chanin Nantasenamat.

Example: A database query app

Now, let's use what we've learned to build a database query app.

- Read in data from file: `wk11_sg_pop.csv`.
- Add widgets to allow user interaction.
- Produce and display a data frame based on user input.
- Build a simple visualization based on user input.

Deploy

Querying population data 🇸🇬

User inputs

Select up to 10 towns

BISHAN × BUKIT MERAH × BUKIT PANJANG × CLEMENTI × JURONG EAST × JURONG WEST × ANG MO KIO × CHOA CHU KANG × HOUGANG × NEWTON ×

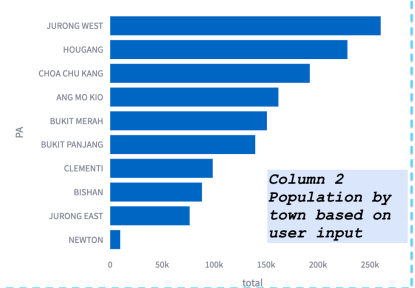
Select a year to display

2001 — 2023

2023

Column 1 Data frame based on user input

	PA	Time	total	change
597	NEWTON	2,023	9,650	570
390	JURONG EAST	2,023	76,490	-310
68	BISHAN	2,023	88,340	2,400
298	CLEMENTI	2,023	98,710	4,380
160	BUKIT PANJANG	2,023	139,580	1,850
137	BUKIT MERAH	2,023	150,840	210
22	ANG MO KIO	2,023	161,910	10
275	CHOA CHU KANG	2,023	192,110	1,330
367	HOUGANG	2,023	228,340	730
413	JURONG WEST	2,023	260,340	1,640



Column 2 Population by town based on user input

Your turn: SG population dashboard

SG Population Dashboard

Select a year

2023

2001 — 2023

Select up to 5 towns

ANG MO KIO × BISHAN × BUKIT MERAH ×

Instructions

- Choose a year to view the **geographic distribution**.
- Choose up to five towns to display the **population trends**.
- The information on top gains/losses will update based on the selected year.

About

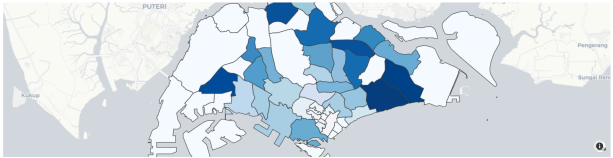
- **Data Source:** Singapore Department of Statistics.
- The figures have been rounded to the nearest 10.

Sidebar: App title + Input widgets

Deploy

Geographic distribution in 2023

Visualization Geographical distribution across towns in selected year



Top Gains/Losses

TAMPINES
274770
↑ 8920

JURONG EAST
76490
↓ -310

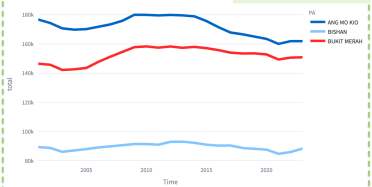
TOTAL
4162530
↑ 76090

Column 1: Metrics Top inbound/ outbound in selected year

Gains/Losses: Regions with high inbound/ outbound residents for the selected year.

Trend in selected towns over the years

Column 2: Visualization Population trend in selected town(s)



Non-script based visualization tools

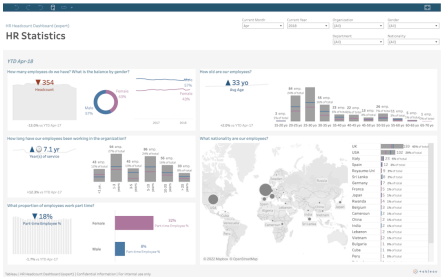
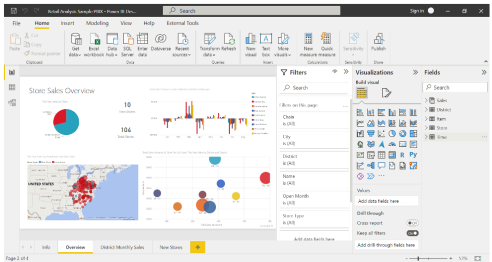


Tableau is an interactive data visualization software, widely used as for business intelligence.

- Pros: Allows a click-drag-drop method to build data visualizations quickly.
- Cons: Limited reproducibility, as workflows are not inherently script-based.



Power BI is another powerful business intelligence tool for data visualization.

- Pros: Click-drag-drop method; Supports scripting with both Python and R for more advanced analytics and customization.
- Cons: Slightly better reproducibility with Power Query and DAX, but still not fully script-based. Primarily based on Windows.

Tableau

- Tableau Public (free).
- Tableau Desktop: A paid version that supports larger data sets, collaboration, and publishing features.

Power BI

- Power BI Desktop (free).
- Power BI Pro: A paid version that allows collaboration with teams, publishing, and sharing reports.

1. Getting started with Tableau.

   https://www.tableau.com/learn/get-started

2. Getting started with Power BI Desktop, by Microsoft.

   https://learn.microsoft.com/en-us/power-bi/fundamentals/desktop-getting-started

3. Courses on DataCamp (our license is valid till Feb 7, 2025).