Plotly graphs and figures

BUILDING DASHBOARDS WITH DASH AND PLOTLY

Alex Scriven
Data Scientist

datacamp

What is Dash?

A Python library for creating interactive, modern, functional web applications easily.

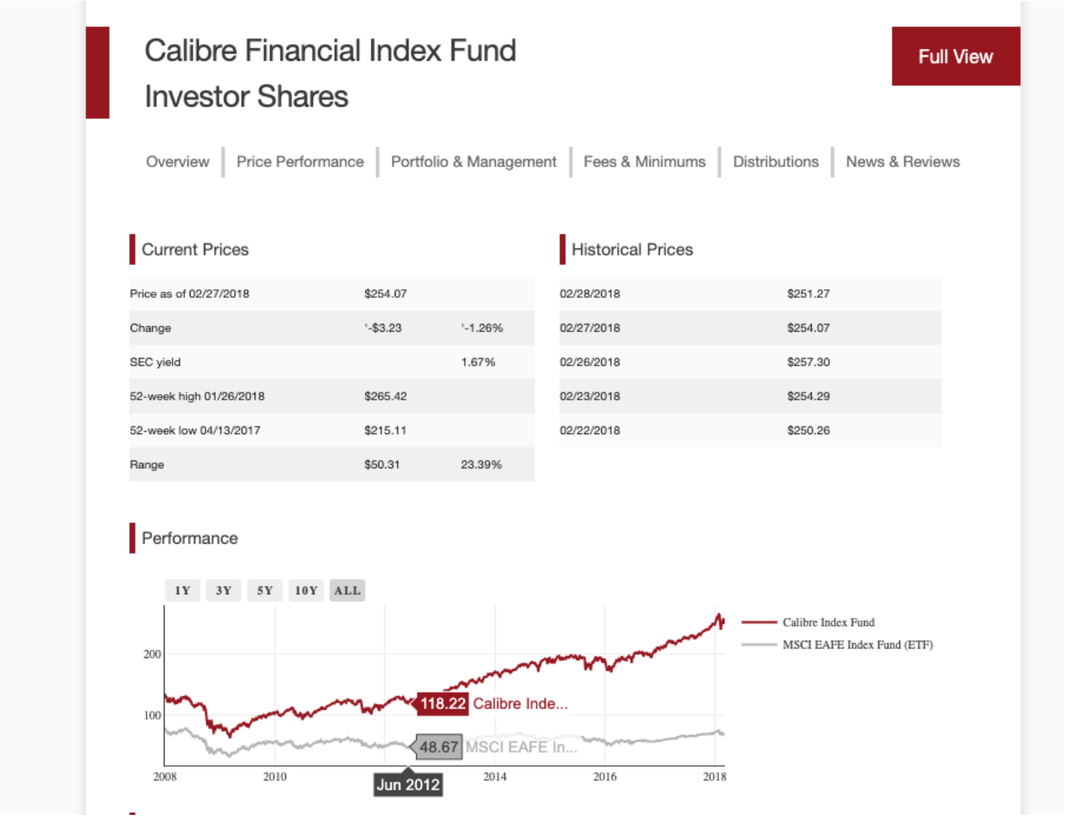
Advantages:

- Free! Unlike Tableau and PowerBI etc.
- Harness JavaScript with only Python
- Less code than web application frameworks like Django

Plotly and Dash

Plotly and Dash work together (same company creator)

- Dash: Interactive dashboards with multiple Plotly graphs
- See this **example**
  - Images, text and Plotly graphs
  - Check out the **source code** (search `go.scatter`)



What is Plotly?

- Revise Plotly, focus on Dash
- A Python library for creating modern, interactive graphs
  - Wraps JavaScript but code in Python
- `plotly.express` for graphs

Our e-commerce data

- Dataset of e-commerce sales
- Details:
  - Item category (Major, Minor) + description
  - Unit price, quantity (+ OrderValue)
  - Country
  - Year-Month of sale

Line charts with plotly.express

Monthly sales using our e-commerce data (`ecom_sales`).

```
import plotly.express as px
line_graph = px.line(
  data_frame=ecom_sales,
  x='Year-Month',
  y='Total Sales ($)',
  title='Total Sales by Month')
line_graph.show()
```

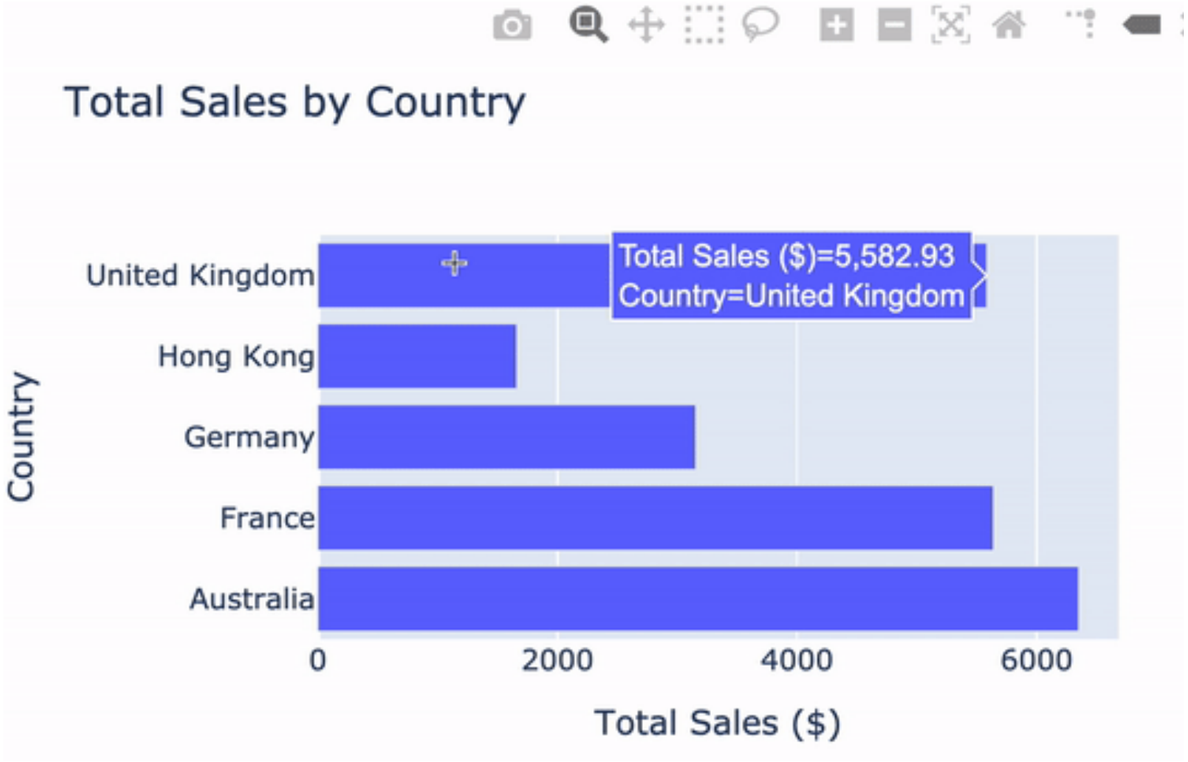
Bar charts with plotly.express

Other `plotly.express` plots are created similarly

A bar chart of the total sales by country:

```
bar_fig = px.bar(
  data_frame=ecom_sales,
  x='Total Sales ($)',
  y='Country',
  title='Total Sales by Country',
  orientation='h')
bar_fig.show()
```

We get an interactive bar chart!



Customizing Plotly graphs

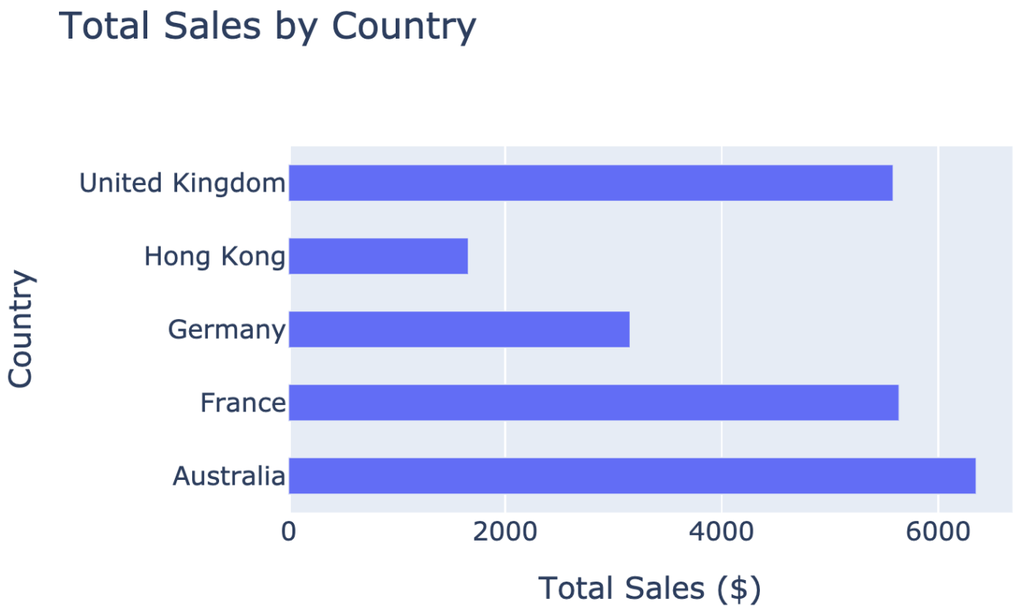
Plotly graph properties can be updated later with `update_layout()` (important for Dash apps!).

Changing the bar width of our bar graph:

```
bar_fig.update_layout({'bargap': 0.5})
bar_fig.show()
```

Check out the **Plotly documentation** for specific arguments for each plot.

Notice the larger gaps between bars?



Let's practice!

BUILDING DASHBOARDS WITH DASH AND PLOTLY

From Plotly to Dash

BUILDING DASHBOARDS WITH DASH AND PLOTLY

Alex Scriven
Data Scientist

datacamp

A first Dash App

A complete Dash app:

```
import dash
from dash import dcc
app = dash.Dash()
app.layout = dcc.Graph(id='example-graph', figure=bar_fig)
if __name__ == '__main__':
    app.run_server(debug=True)
```

- Python functionality possible
  - e.g., String interpolation `print("f{my_variable}")`

The main Dash imports

```
import dash
from dash import dcc
```

- `dash` is the main library that creates the app itself
- `dcc` ('dash core components') contains the different building blocks to create the app
  - Two components in our app
  - More components throughout the course (e.g., user inputs!)

The app layout

```
app = dash.Dash()
app.layout = dcc.Graph(
  id='example-graph',
  figure=bar_fig)
```

- Create an app object using `dash.Dash()`
- Set the `app.layout`
  - Here, a single graph
  - Using `dcc.Graph()`
    - `figure` = The Plotly figure to render
    - `id` = Important for callbacks later

Running the app

```
if __name__ == '__main__':
    app.run_server(debug=True)
```

- Lastly, running the server
- Script is run from command-line (not read into a notebook)
  - i.e., `python my_app.py` in the command-line
- `debug` for helpful feedback when testing

Our app

Script is run via the command-line (`python3 script.py`), served on a local server

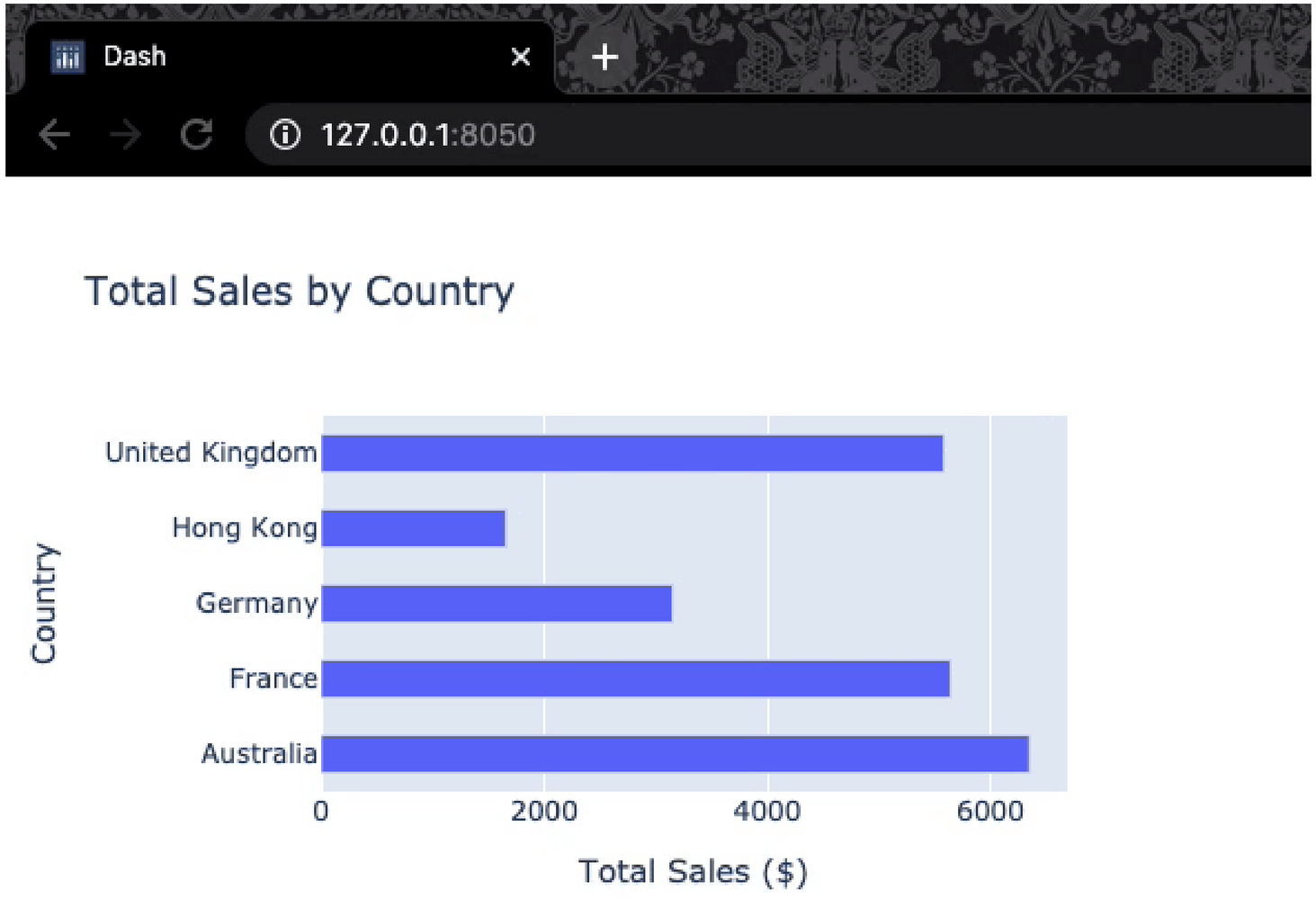
Access via a web browser such as Google Chrome

While served, update and save `.py` file to see live updates in browser

```
Dash is running on http://127.0.0.1:8050/

 * Serving Flask app "simple_app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
```

Our app in the browser



Dash in DataCamp

- Some differences to other DataCamp exercises:
  - All code inside the panel (Pre-exercise, dataset etc.)
  - All executed at once (not line-by-line)
  - (Much) longer code
  - `dash.Dash(__name__)` (The `__name__` not needed locally)
- Fully-functional dashboards (expand window to see!)

Let's practice!

BUILDING DASHBOARDS WITH DASH AND PLOTLY

Positioning Dash components

BUILDING DASHBOARDS WITH DASH AND PLOTLY

Alex Scriven
Data Scientist

datacamp

HTML and the web

HTML: language for structuring websites

- HTML: wooden structure of a house
  - Set placement of objects
- CSS: paint color of a room
  - Style (e.g., background color) of objects
- JavaScript: Smart home clap-on lights!
  - Interactivity e.g., clickable actions

Div and H tags

Dash uses dash html components (`dash.html`) to interface between HTML and Python.

Two important HTML structures ('tags'):

- Div tags: important for structuring websites
  - Can have many different-sized divs with different things inside
- H tags: different sized titles (H1 > H6)

Using Div and H tags

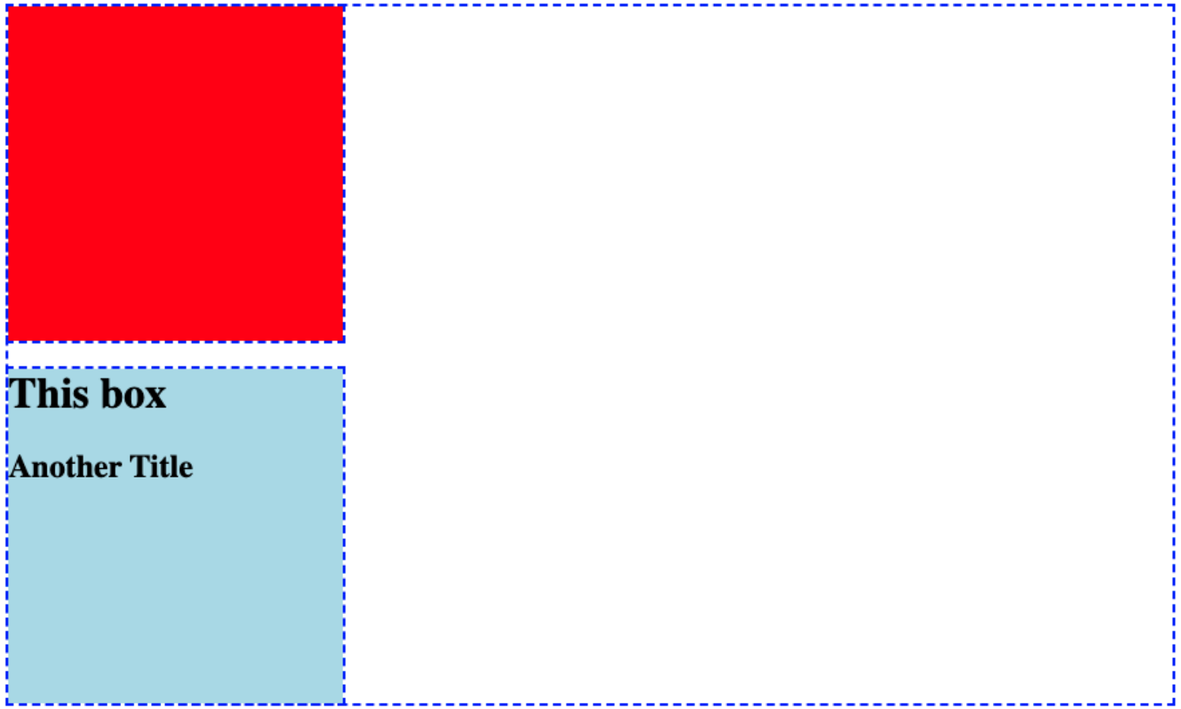
Some HTML code with:

- Overall div (everything inside)
- Div inside: red background
- Div with blue background
  - H tags inside
- Ignore the `style` part - more on 'CSS' later!

```
<div>
  <div style="background-color: red;
              width:250; height:250;">
  </div>
  <div style="background-color: lightblue;
              width:250; height:250;">
    <h1>This box</h1>
    <h2>Another Title</h2>
  </div>
</div>
```

Our example displayed

Our example



Take note:

- Red background div
- Blue background div with H tags

The div tag can nest; lots of possibilities when structuring our web app.

Our example in Dash

Recreating HTML example with Dash

```
import dash
from dash import dcc, html
app = dash.Dash()
app.layout = html.Div(children=[
    html.Div(style={'height':250, 'width':250, 'background-color':'red'}),
    html.Div(children=[
        html.H1("This box"),
        html.H2("Another Title")],
        style={'background-color':'lightblue'})
    ])
```

Breaking down the layout

- HTML tags align to Dash `html.()`
  - `html.Div()` = `<div>`
  - `html.H1()` = `<h1>`
- The overall div and the last div have a `children` argument
  - A list of components go inside
  - Second Div doesn't have this (single sub-element)
- We can put `dcc.Graph()` components inside too!

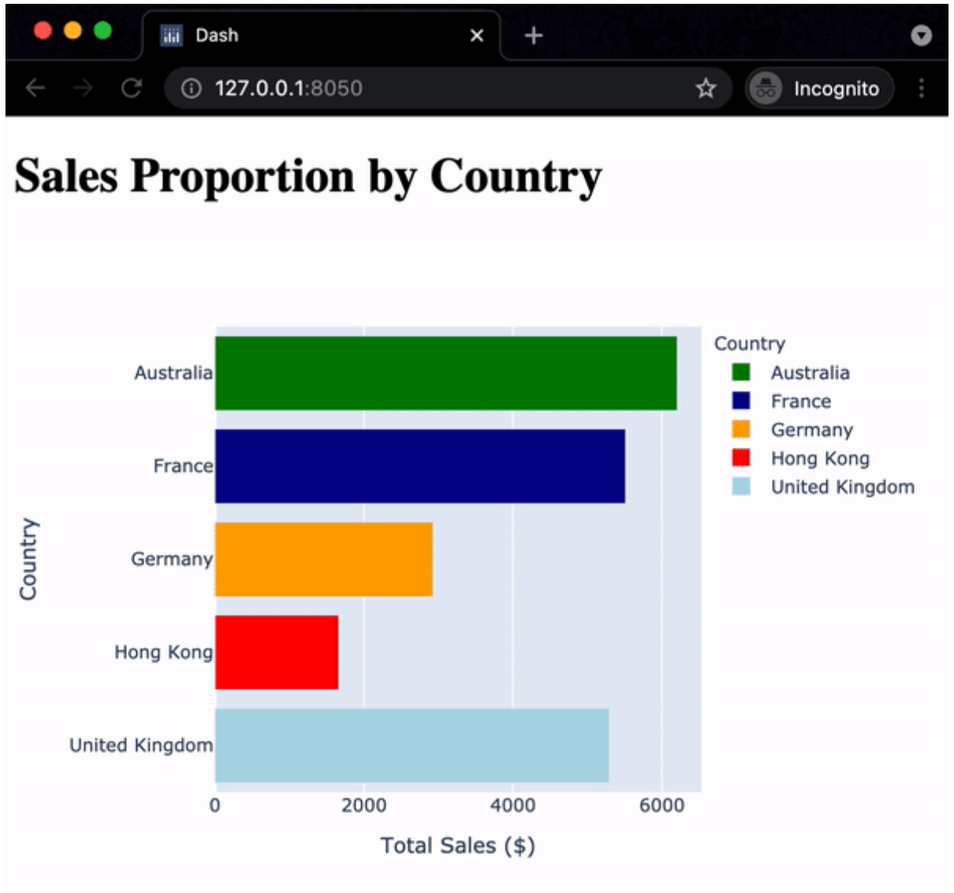
```
import dash
from dash import html
app.layout = html.Div(
  children=[
  html.Div(
    style={'background-color':'red',
           'height':250, 'width':250}),
  html.Div(
  children=[
      html.H1("This box"),
      html.H2("Another Title")]
    ,style={'background-color':'lightblue',
            'height':250, 'width':250})
```

Graphs in the layout

Graphs can be added inside the `children` list of a `html.Div()`

```
bar_fig_country = px.bar(ecom_sales,
 x='Total Sales ($)', y='Country',
 color='Country', color_discrete_map=
 {'United Kingdom':'lightblue',
 'Germany':'orange', 'France':'darkblue',
 'Australia':'green', 'Hong Kong':'red'})
app = dash.Dash()
app.layout = html.Div(
  children=[
  html.H1("Sales Proportion by Country"),
  dcc.Graph(id='bar_graph',
            figure=bar_fig_country)])
```

Produces:

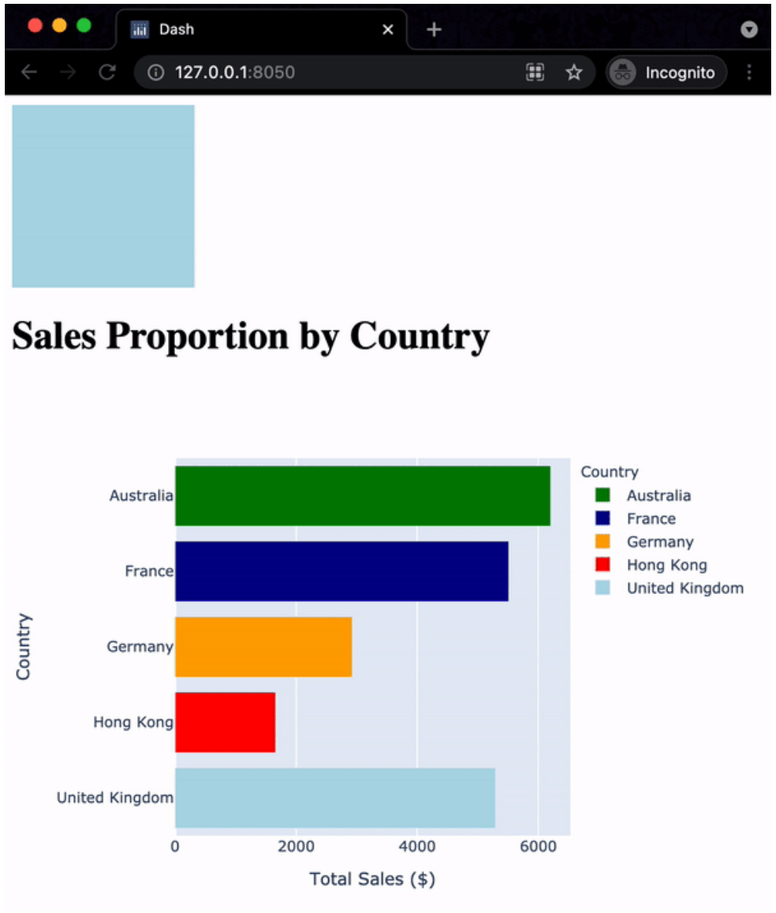


Adding more structure

Let's add another `html.Div()`. What happens?

```
app.layout = html.Div(
  children=[
  html.Div( style={
    'width':150,'height':150,
    'background-color':'lightblue'}),
  html.H1("Sales Proportion by Country"),
  dcc.Graph(id='bar_graph',
            figure=bar_fig_country)])
```

Our new dashboard:



Let's practice!

BUILDING DASHBOARDS WITH DASH AND PLOTLY