howest
hogeschool

Plotly

Introduction to Plotly

What?

- (Python) Library for creating interactive visualisations
- Works with Jupyter Notebooks, Python scripts and web applications

Why?

- Interactive by default
- Multiple chart types: line, bar, scatter, pie, 3D, maps …
- Easy to create full dashboards with **Dash** or integrated in Gradio
- Works with large datasets

Getting started

- Installation

```
pip install plotly
```

- Importing the library, two different ways

```
import plotly.express as px

import plotly.graph_objects as go
```

- Works well with Pandas Dataframes in Python

Plotly Express vs Graph Objects

Plotly Express

- Easy for quick access, without much additional formatting options
- Only shows one **Trace**
- Basic animation options
- High-level API
- DataFrame-based data

Graph Objects

- More customisation options
  - Layout, axes …
- Multiple Traces in one Graph
- Advanced **Animation** options
- 3D plots
  - Scatter3D, Surface, Map
- Full control
- Dashboard-style

Plotly terminology

- **Trace**
  - A single series of data in a chart
  - Multiple traces for complexer visualisations
- **Figure**
  - Main object to hold traces and layout configuration
- **Layouts**
  - Control appearance, axes, titles, gridlines, backgrounds …
- **Annotations & Shapes**
  - Custom tekst, lines, highlights …
- **Subplots**
- **Dash components --> Building dashboards, more later**

Basic chart options

- Line
- Scatter
- Bar
- Histogram
- Box plot

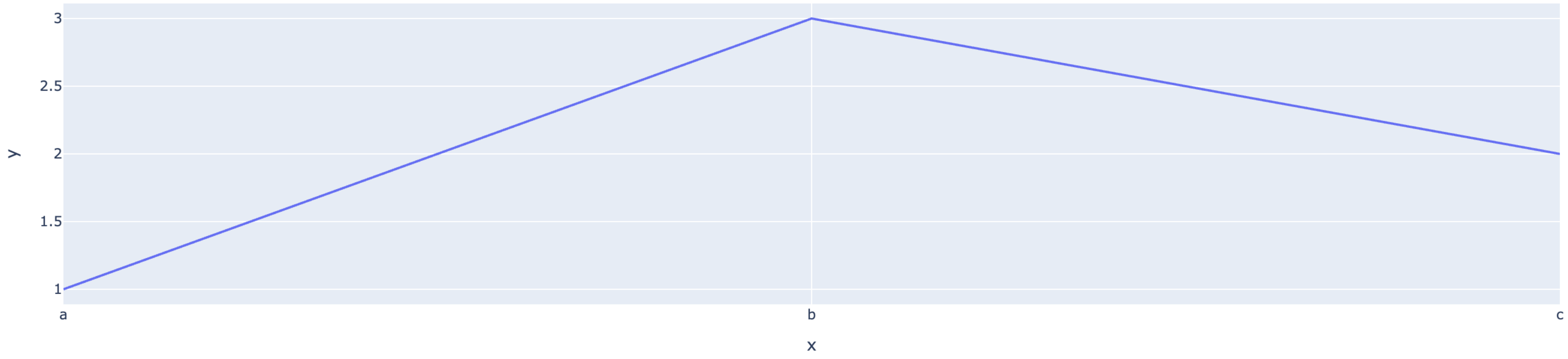
Plotly advantages over **Matplotlib, Bokeh, Seaborn …**

- Interactivity: Zooming, hovering, animations …

Plotly Express: Line plots

- Automatically draws straight lines between points

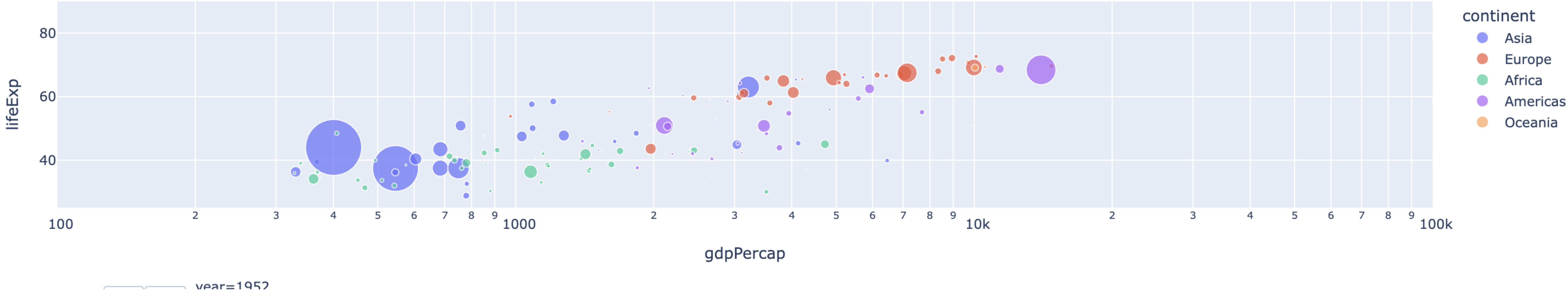
```python
px.line(x=["a", "b", "c"], y=[1, 3, 2])
```



Plotly Express: Scatter plots

- Coloured by specific column
- Possible: Logarithmic axis values

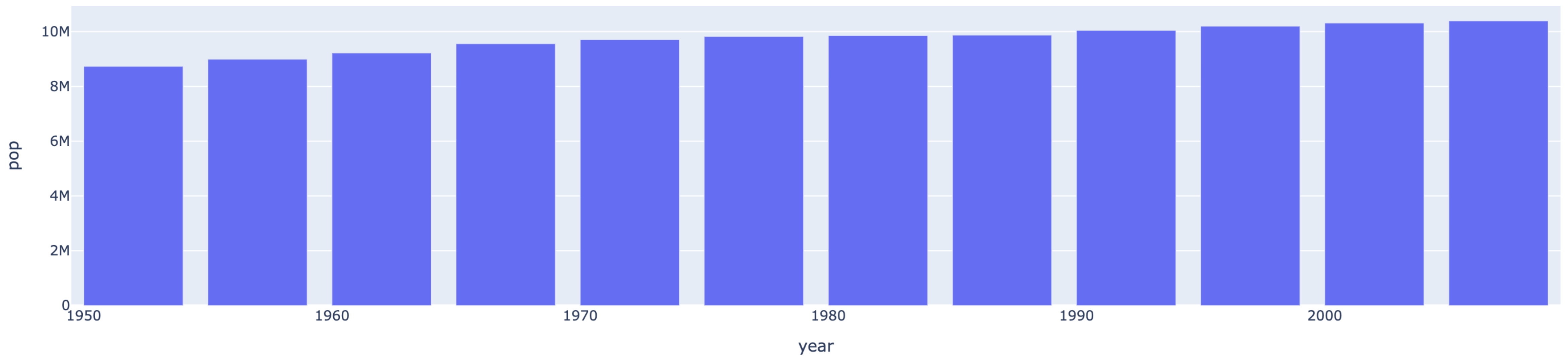
```
px.scatter(df, x="gdpPercap", y="lifeExp", animation_frame="year",              animation_group="country", size="pop",
color="continent", hover_name="country",
        log_x=True, size_max=55, range_x=[100,100000], range_y=[25,90])
```



Bar plots

- Categorical data
- **Discrete comparisons** between different categories
- **Example**: Population of different countries per year

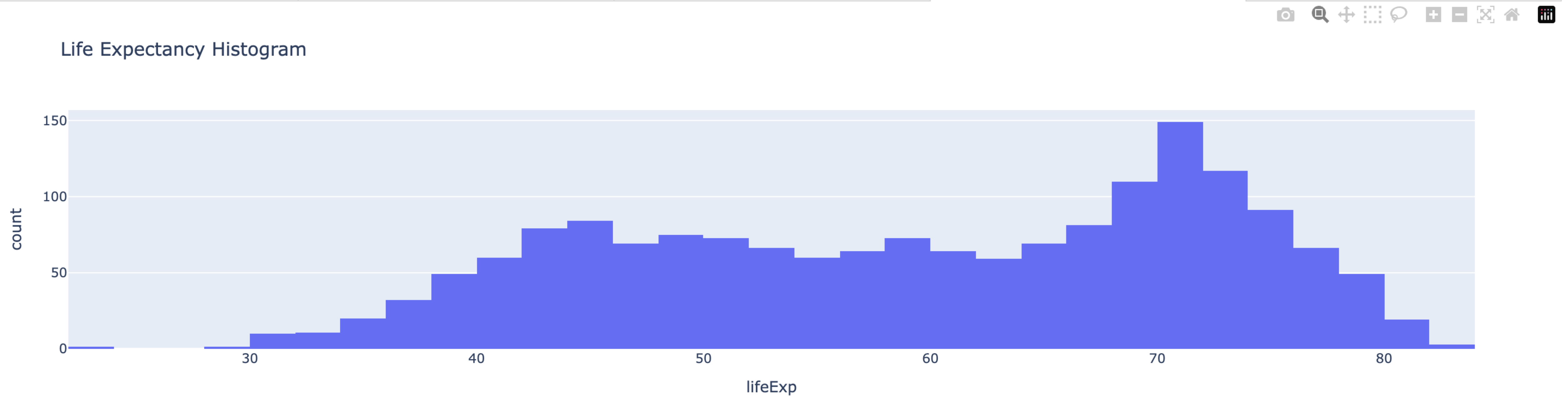
```
px.bar(df[df["country"] == "Belgium"], x='year', y='pop', height=400)
```



Plotly Express: Histogram plot

- Continuous numerical data
- Shows **distribution** of values over intervals (bins)
- **Example:** Distribution of life expectancy across all countries

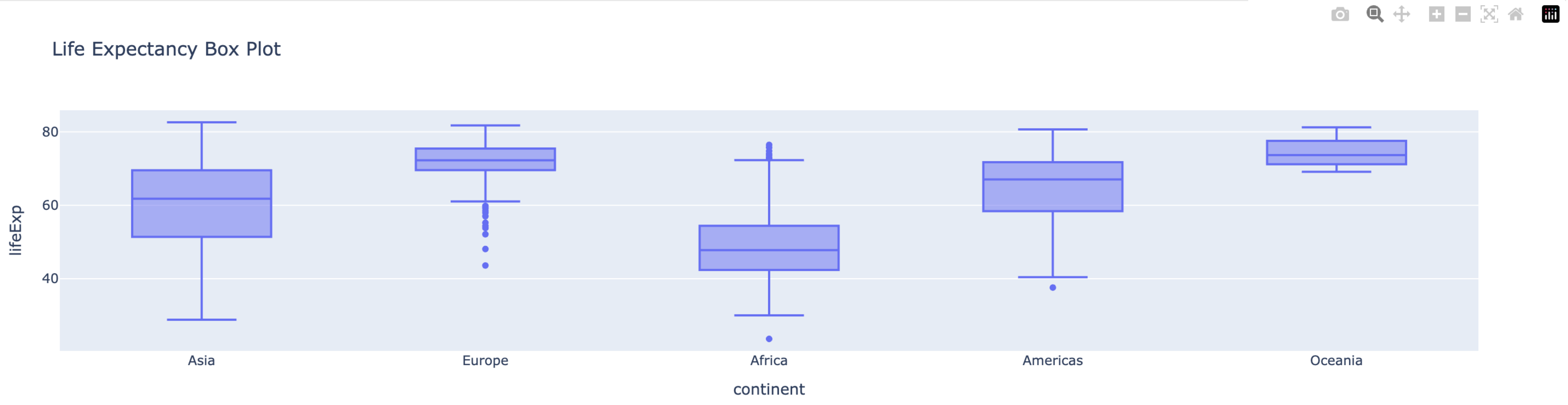
```
px.histogram(df, x='lifeExp', nbins=30, title='Life Expectancy Histogram')
```



Plotly Express: Box plot

- Finding mean values
- Spotting outliers

```
px.box(df, x='continent', y='lifeExp', title='Life Expectancy Box Plot')
```



Advanced charts

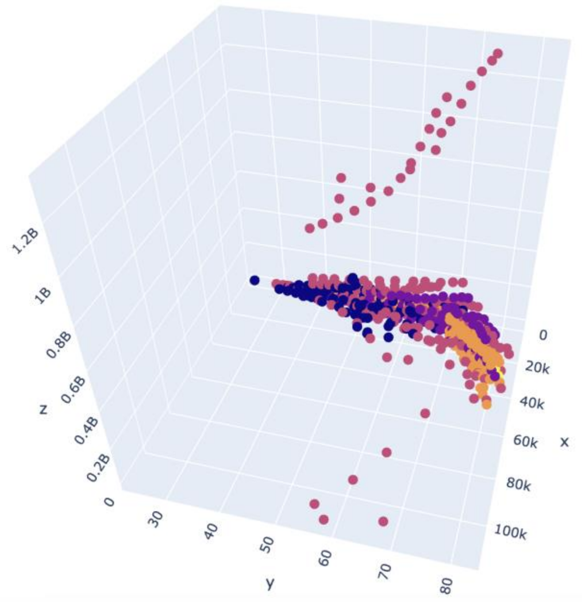
Some of these are often created using Graph objects if Plotly Express is not sufficient.

- Scatter3D
- Map
- Treemaps
- Heatmaps
- Radar plots
- …

3D Scatter Plots

- Used when you have three dimensional data
- Plotted on x, y, z
- Colours indicate a fourth dimension or classes

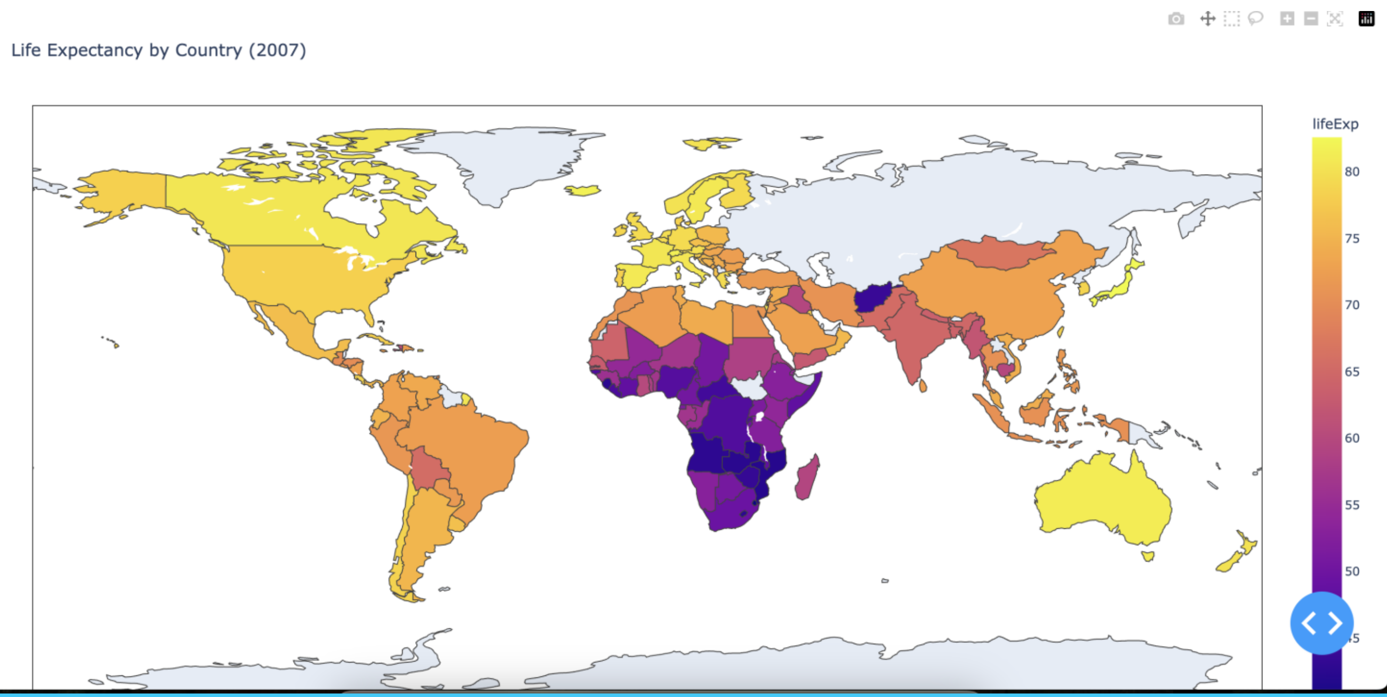
3D GDP, Life Expectancy, Population

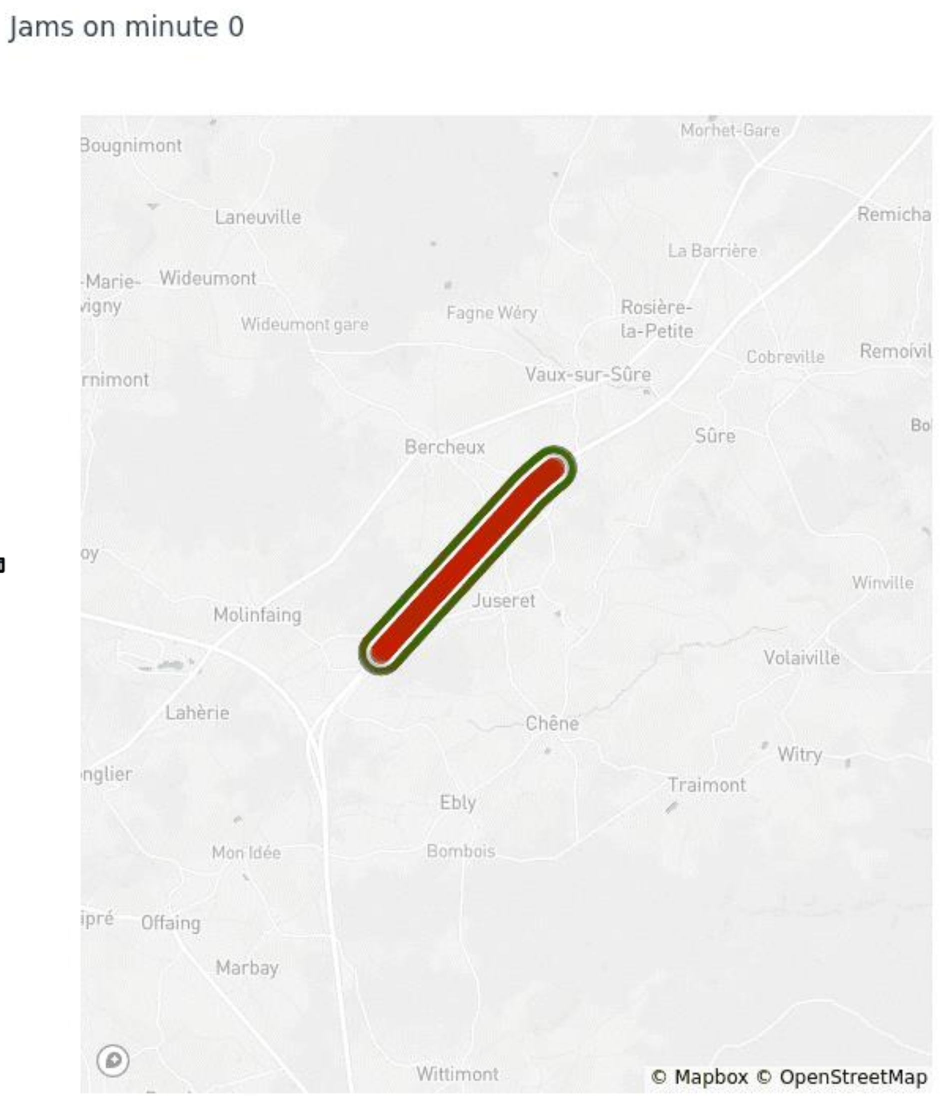


```
fig = go.Figure(
data=[
        go.Scatter3d(
                x=df['gdpPercap’],
                y=df['lifeExp’],
                z=df['pop’],
                mode='markers’,
                marker=dict(
                        size=5,
                        color=df['continent’]
                .astype('category').cat.codes))
        ])
fig.update_layout(
        title='3D GDP, Life Expectancy,       Population’,
        height=800
)
```

Map plots

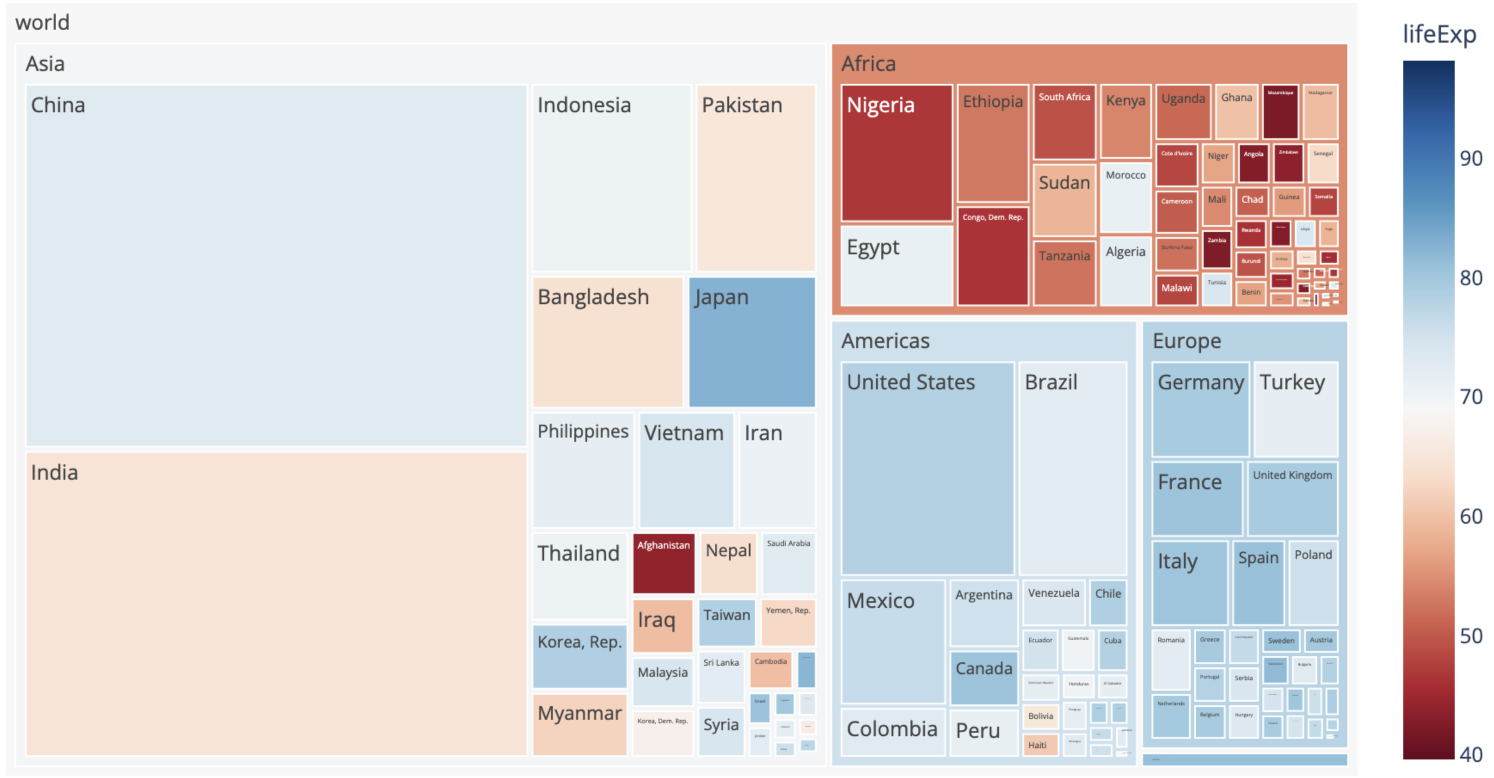
- 2D / 3D
- Plotting with coordinates
- Hover interactions





Treemap

- Creative visualisation of nested data



howest
hogeschool

Interactivity

Interactions

- Hover
- Subplots, grids …

- Slider controls
- Dropdown menu’s

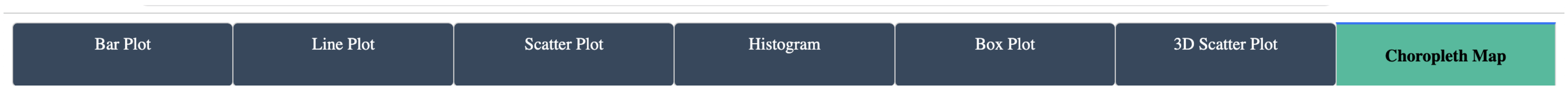
These can also be done by using Gradio instead of Plotly / Dash

howest
hogeschool

Dash

Dash

- Used to create dashboards
- Interactive interfaces with **input** and **output** options
- Hostable in apps and websites
- HTML-ready
- Easy styling using CSS
- High performance for extremely large datasets



```
app = Dash(__name__)
app.layout = html.Div([
        dcc.Tabs(id='tabs', value='1', children=[
                dcc.Tab(label='Bar Plot', value='1’),
                dcc.Tab(label='Line Plot', value='2’),
                dcc.Tab(label='Scatter Plot', value='3’),
                dcc.Tab(label='Histogram', value='4’),
                dcc.Tab(label='Box Plot', value='5’),
                dcc.Tab(label='3D Scatter Plot', value='6’),
                dcc.Tab(label='Choropleth Map', value='7')
        ]),
        dcc.Graph(id='plot-output', figure=demo_fig_01())
])

@app.callback(
        Output('plot-output', 'figure’),
        Input('tabs', 'value')
)
def update_plot(tab_value):
        return show_demo(int(tab_value))

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

Dash vs Gradio

- Dash is well integrated with Plotly
- Dash is often an easy way when you’re working in Jupyter Notebooks
- Quick editing of Tabular data without leaving Jupyter environment

- Gradio is integrated with Plotly and other visualisation libraries
  - But not quite as well as Dash is...
- Gradio offers the option to integrate the other Gradio components as well …