Lecture 3a
Overview of Plot Types

Data visualization · 1-DAV-105

Lecture by Broňa Brejová

Plan for today

- Types of variables (columns)
- Gallery of different plot types, some discussion of their properties
- Some notes on how to draw them in Python (more in a notebook)

Types of variables (columns)

Categorical / qualitative

- **Nominal:** values have no fixed ordering (for example, gender, country, color)
- **Ordinal:** values are ordered (for example education level primary / secondary / university; star ranking 0-5)

Numerical / quantitative

- **Discrete:** typically counts
- **Continuous:** typically measurements

Types of variables (columns)

Numerical / quantitative

- **Discrete:** typically counts
- **Continuous:** typically measurements

Numerical variables also categorized as follows:

- **Ratio (pomerová):** if zero means "none", and it is meaningful to compute ratios / percentages (mass, length, duration, cost, ...)
- **Interval:** does not have "true zero", we can subtract but not make ratios (temperature in degrees C, date)

Data for today

- Various country indicators downloaded from the World Bank for years 2000, 2010, 2020
- Population, area, GDP per capita, life expectancy, fertility (number of children per woman)
- Also classification into regions and income groups
- Which are categorical / numerical?

We will also use Gapminder life expectancy 1990-2021 from I01

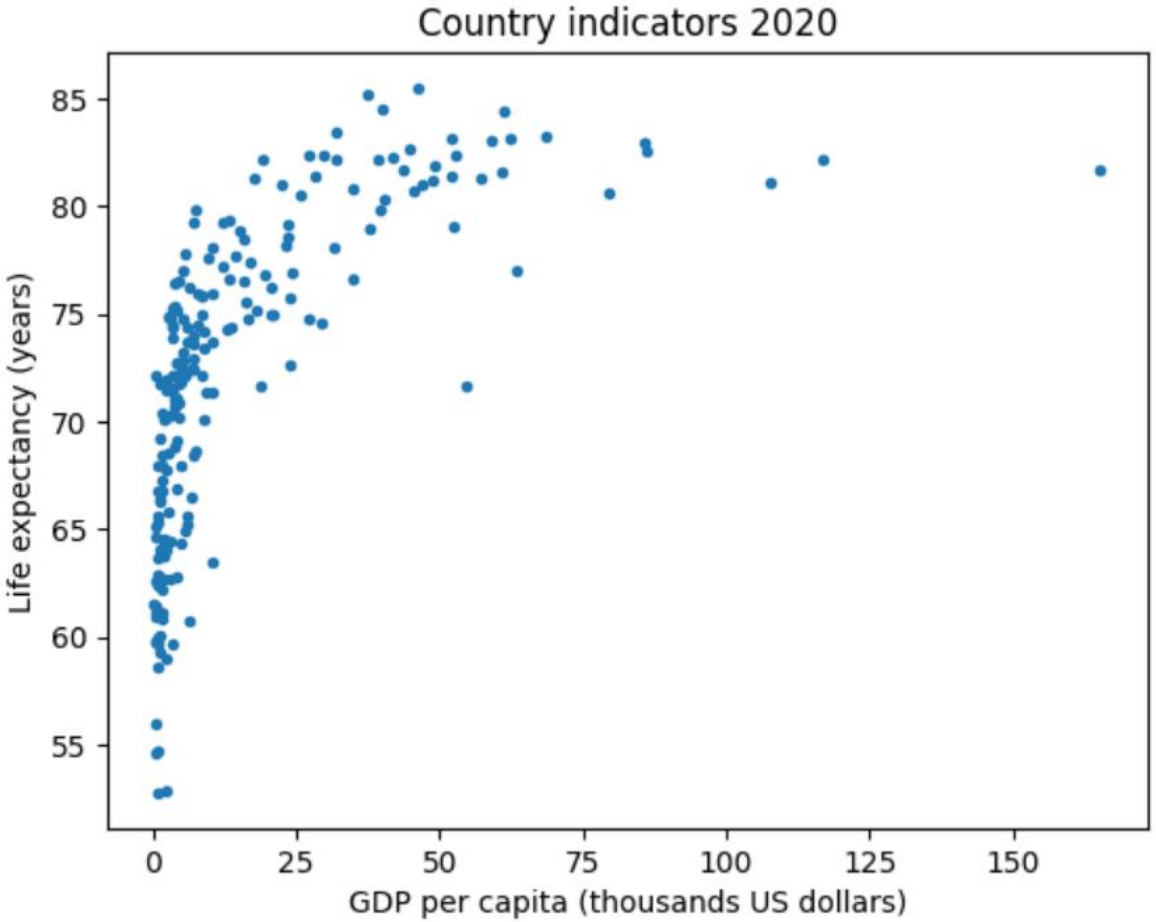
Scatter plot (bodový graf)

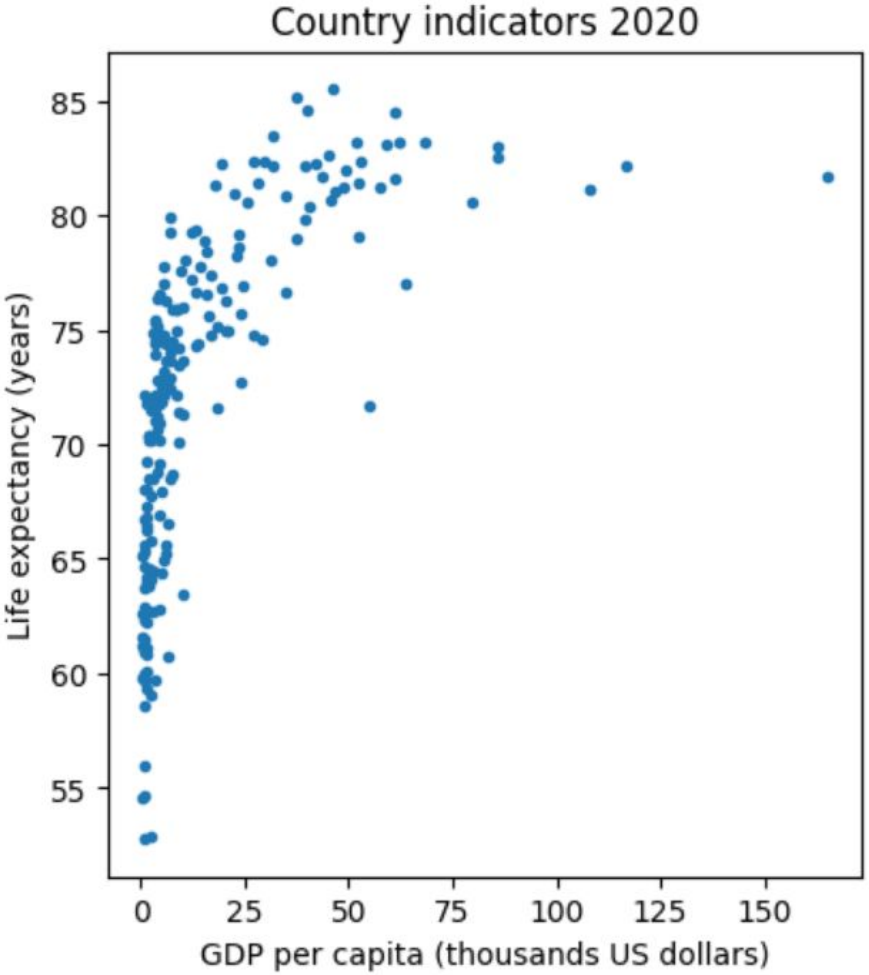
Good for two numerical variables (x and y).

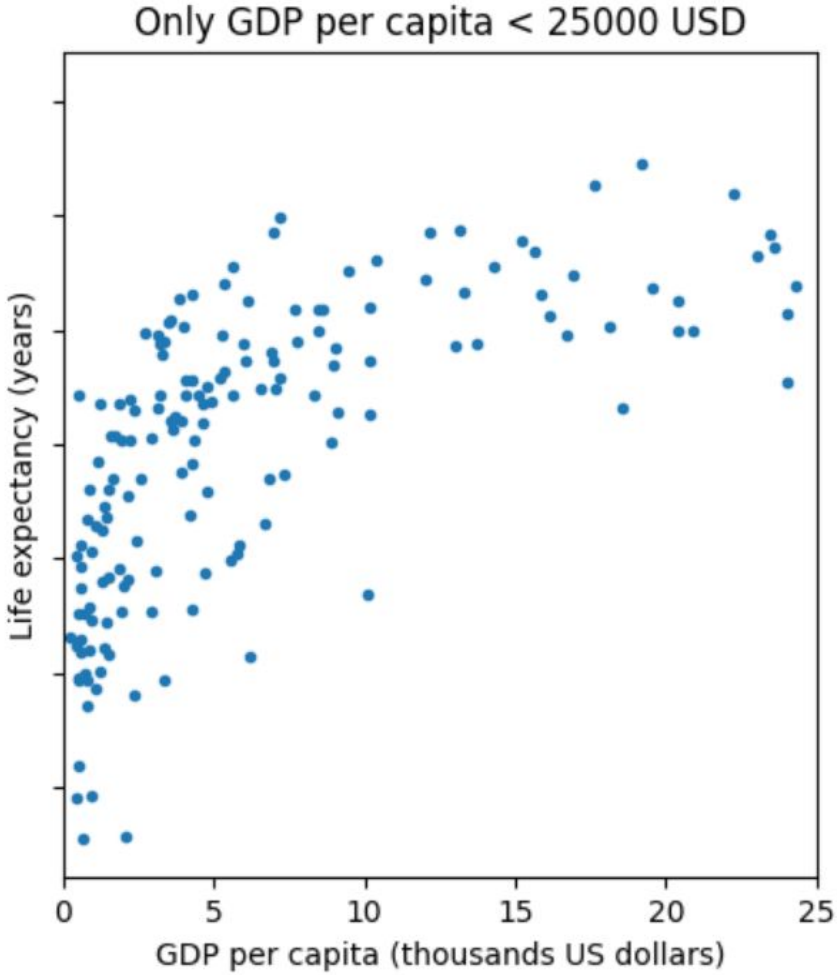
In this plot, many points near left boundary, most space empty.

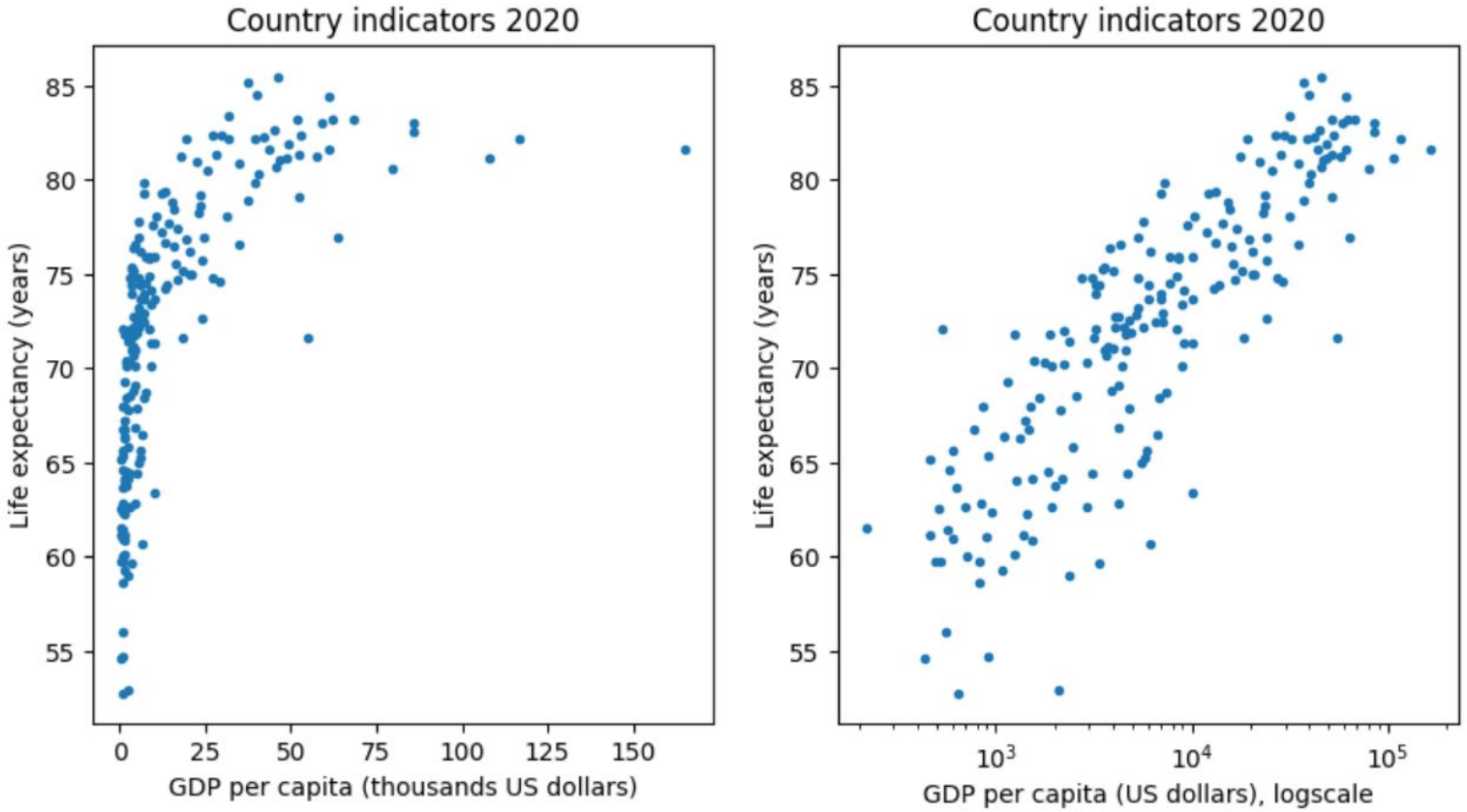
Solution 1: combine overall view and detail

Solution 2: log scale



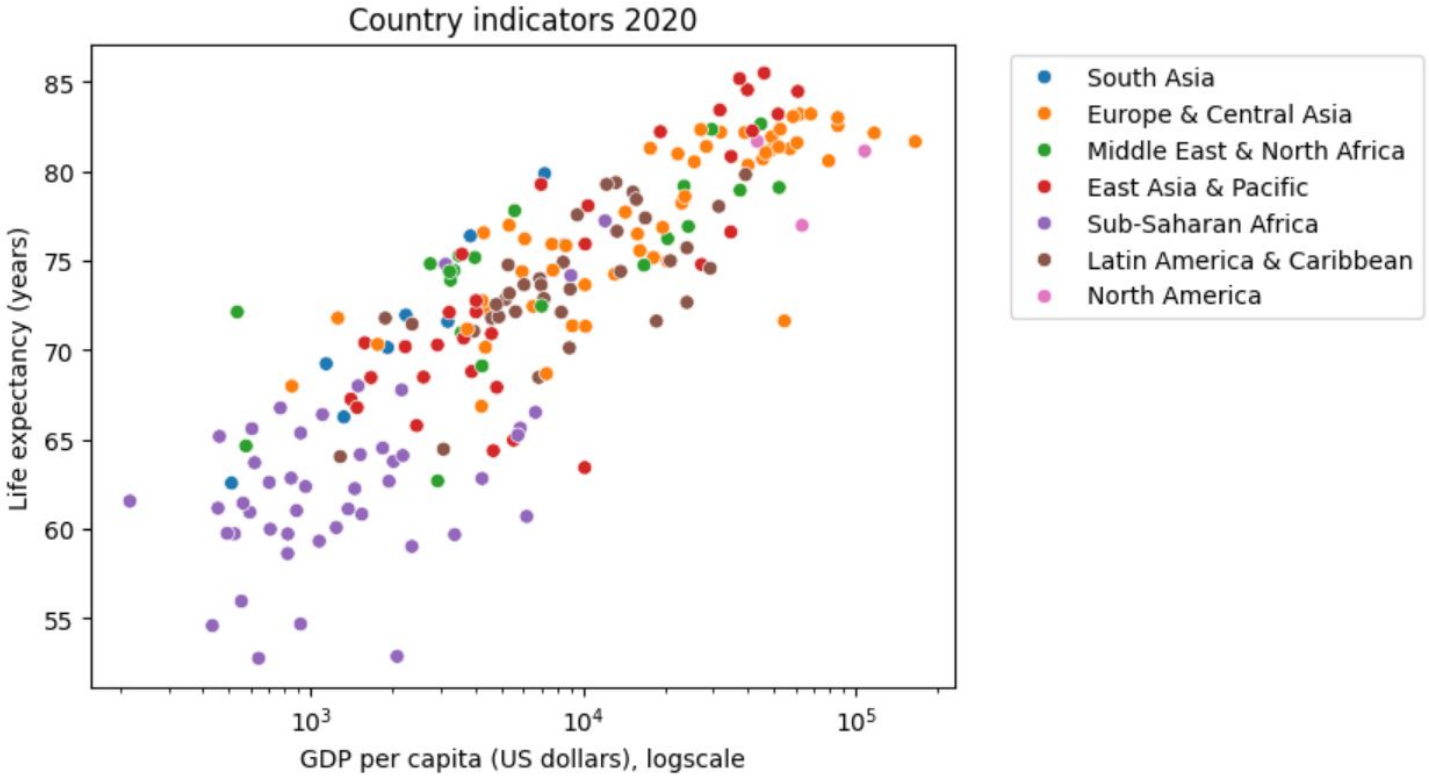

Country indicators 2020
Life expectancy (years)
85
80
75
70
65
60
55
0
25
50
75
100
125
150
GDP per capita (thousands US dollars)


Only GDP per capita < 25000 USD
Life expectancy (years)
0
5
10
15
20
25
GDP per capita (thousands US dollars)

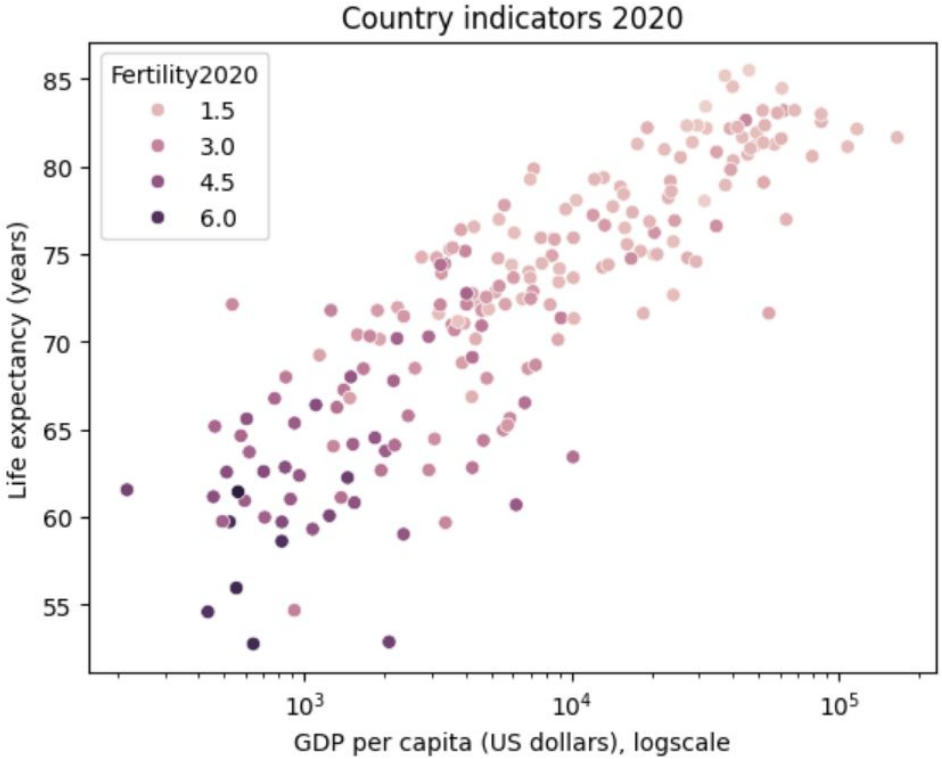


Log-scale x-axis: draw at log(x) instead of x, but axis ticks show values of x

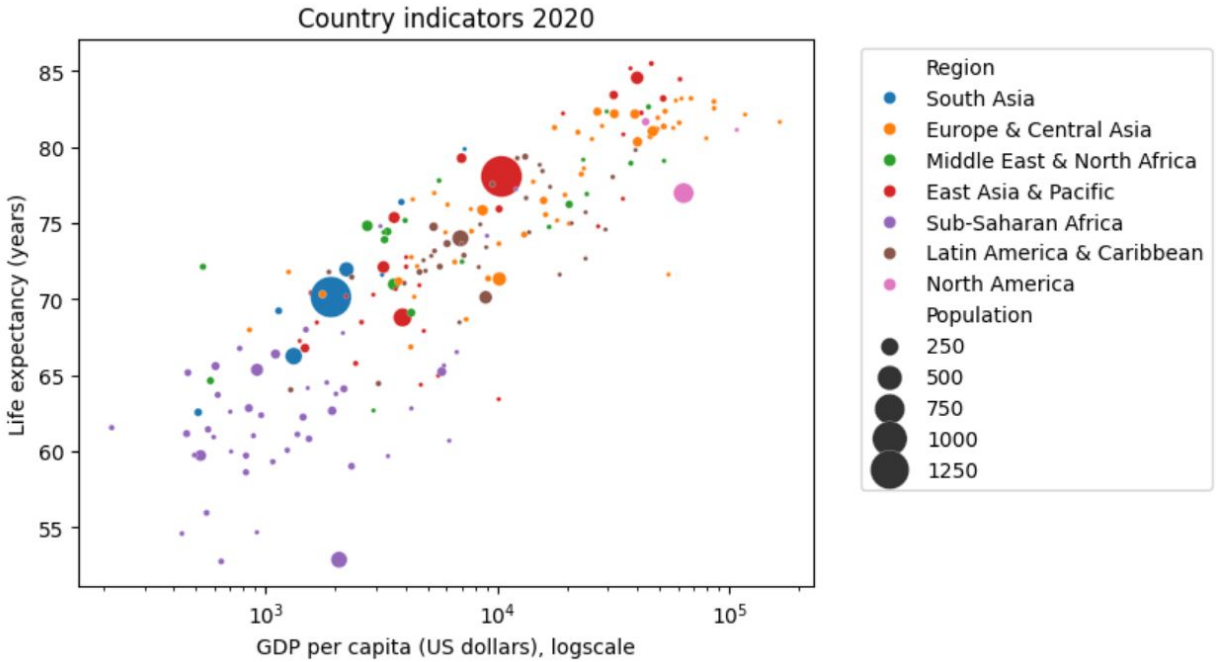
Adding a categorical variable with color



Adding a numerical variable with color scale

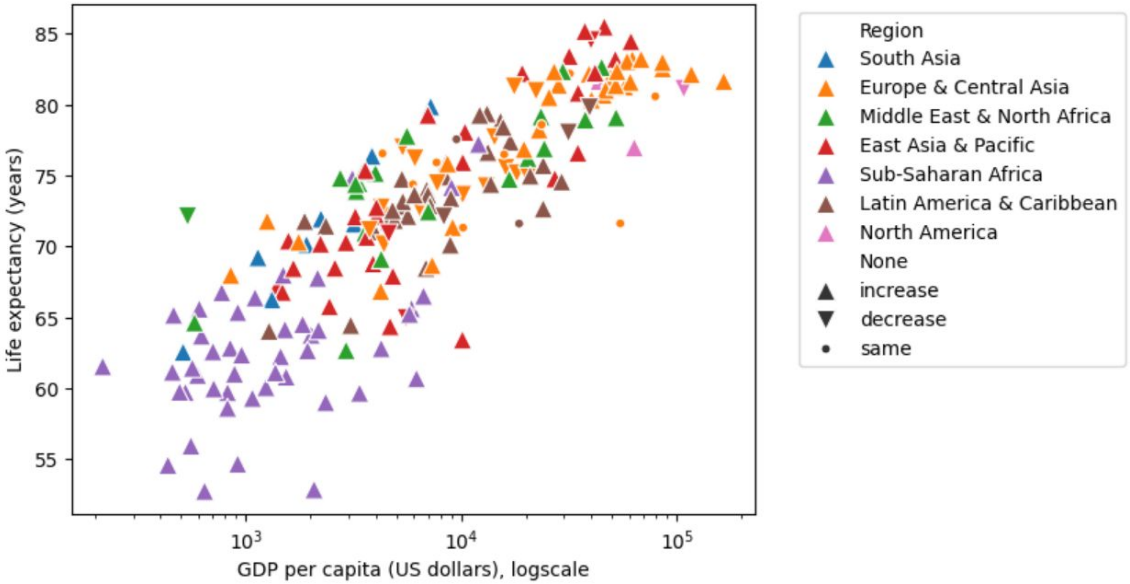


Adding numerical variable with marker size



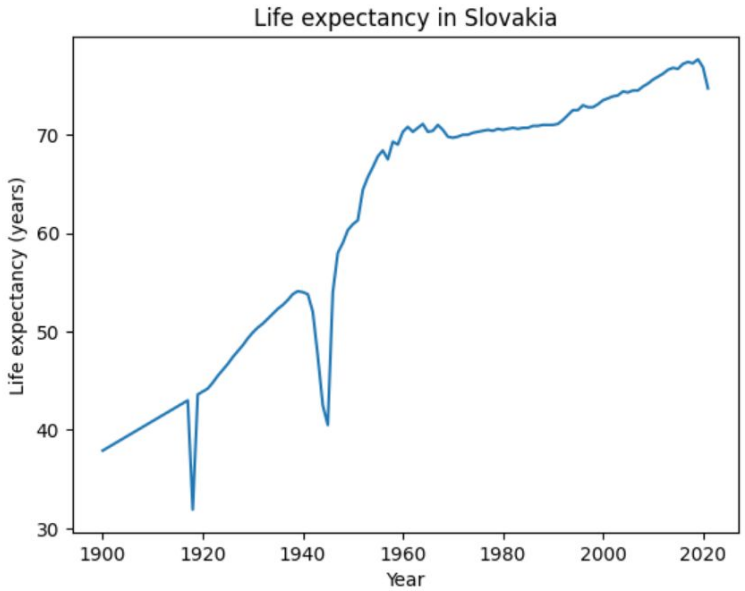
Variable value should be proportional to circle area, not diameter!

Adding categorical variable with marker shape



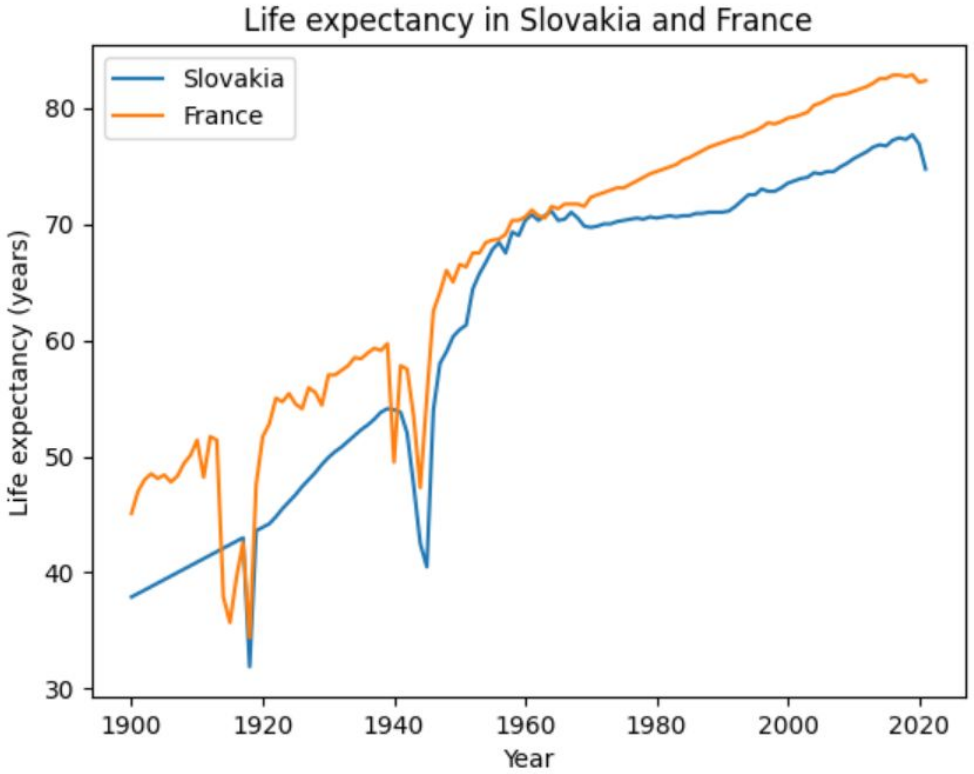
Hard to read, particularly for many data points
Showing population change between 2000 and 2020
If less than 1% change, marked as equal

Line graph (čiarový graf)

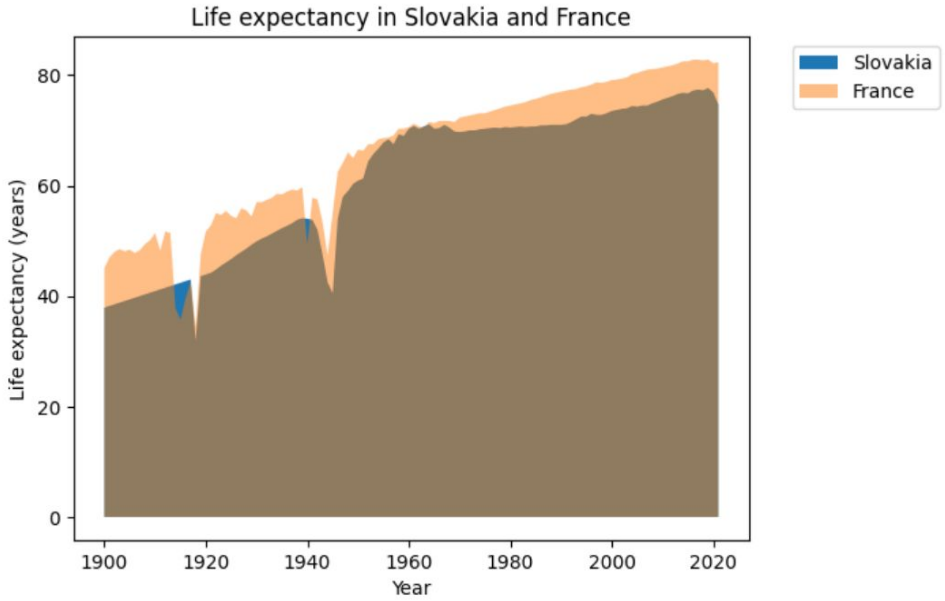


Emphasizing continuity between data points
Data points can be also shown as markers

Adding categorical variable with color

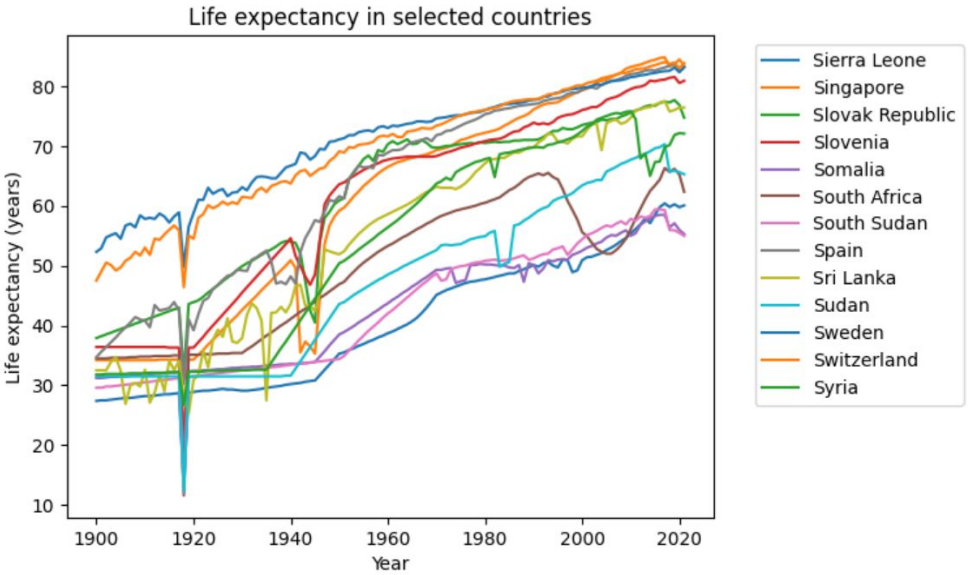


Area graph (plošný graf)



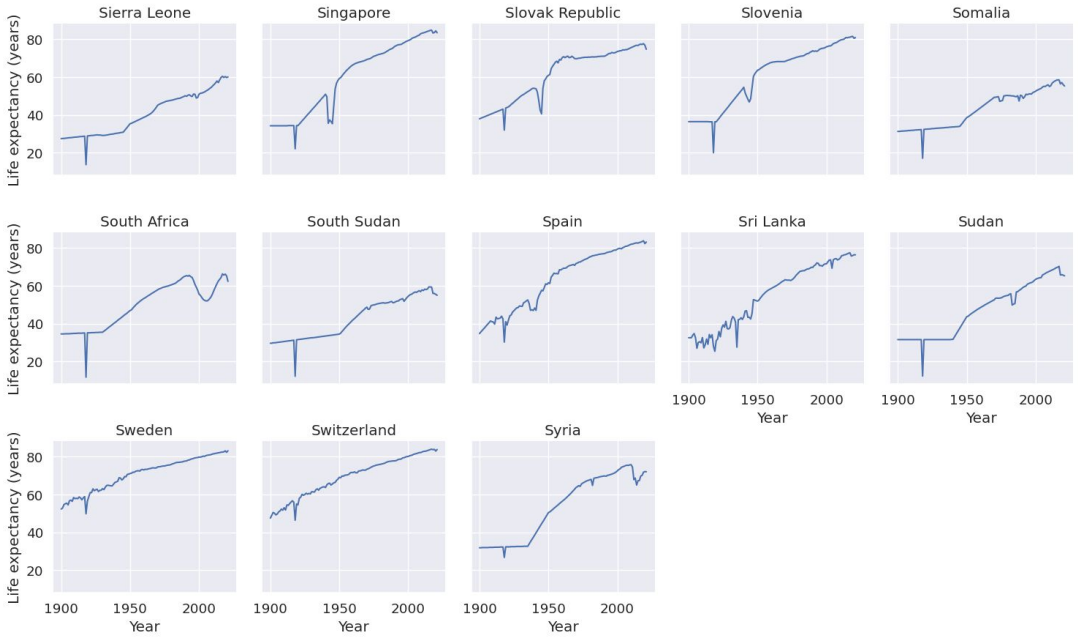
Y-axis must start at 0
Emphasizes differences more than line graph, but also more cluttered

Line graph with many lines



Hard to follow individual lines, but shows general trends and comparisons.
Countries with names Si..Sw, and having population at least 1 million.
Note that colors start to repeat.

Small multiples



A small plot for each value of a categorical variable

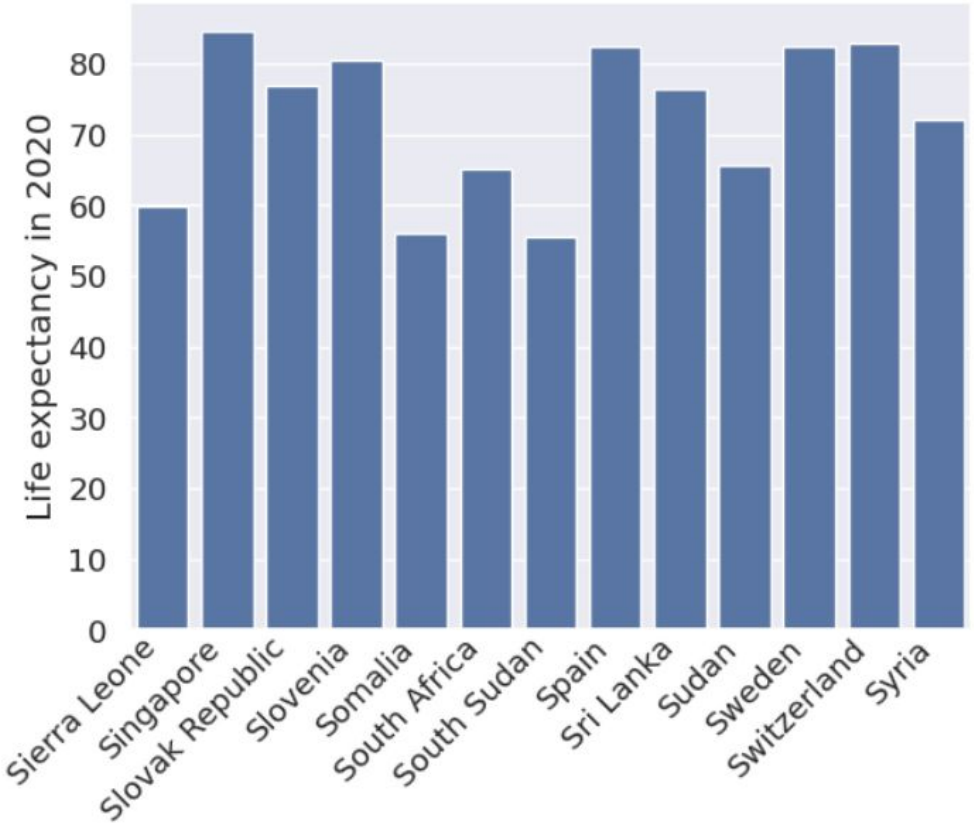
Must have the same axes!

Exact comparison difficult, but it is possible to notice different trends

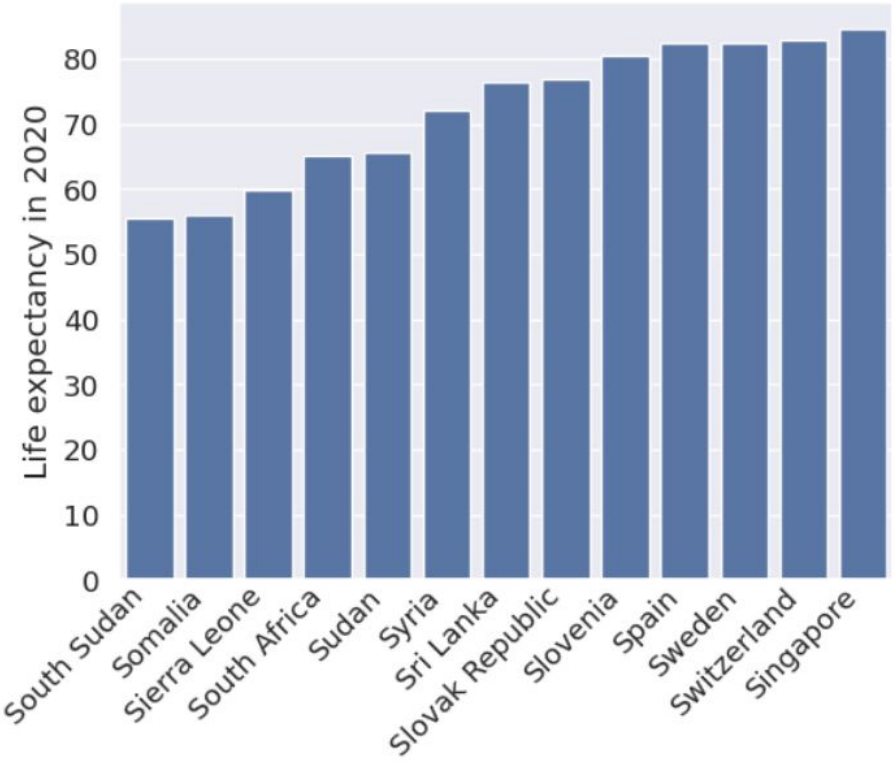
Bar graph (stĺpcový/pruhový graf)

X-axis is categorical

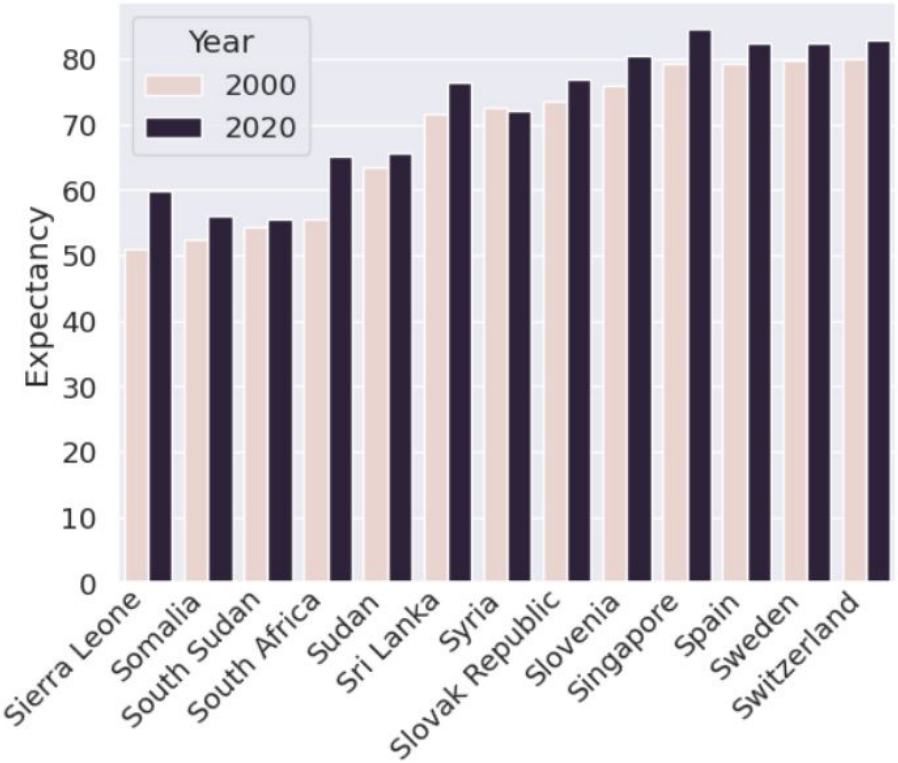
Y-axis must start at 0



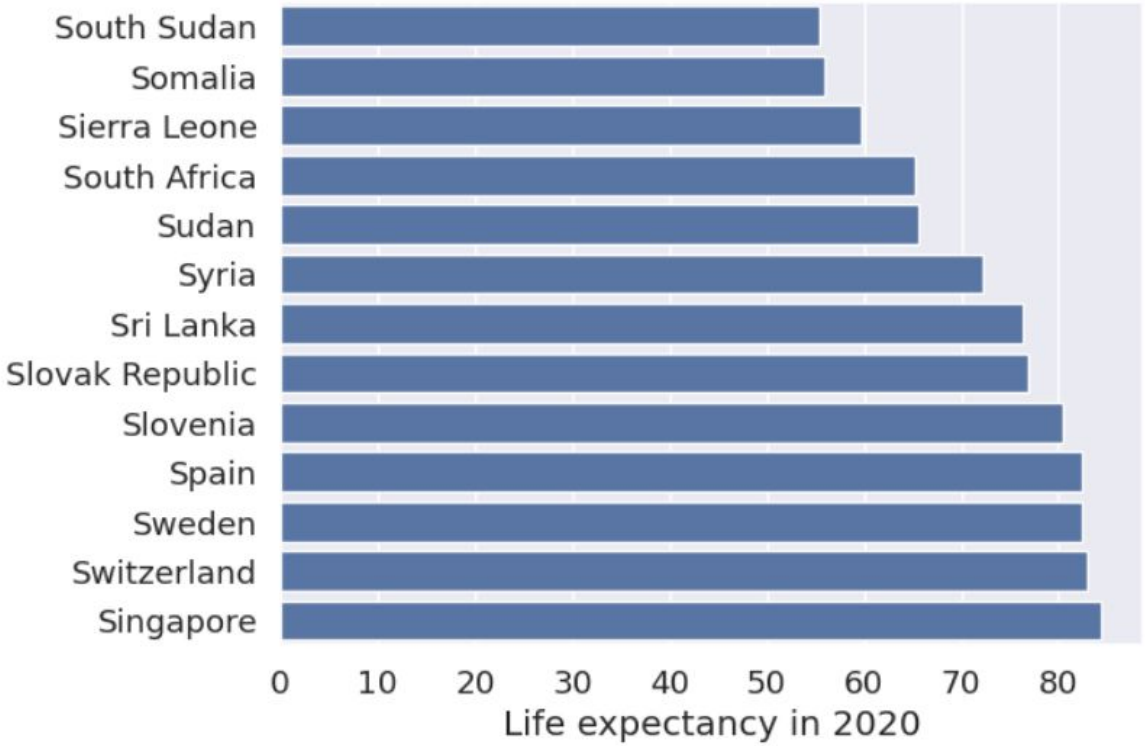
Bar graph with sorted columns



Bar graph with colored columns



Bar graphs can be horizontal



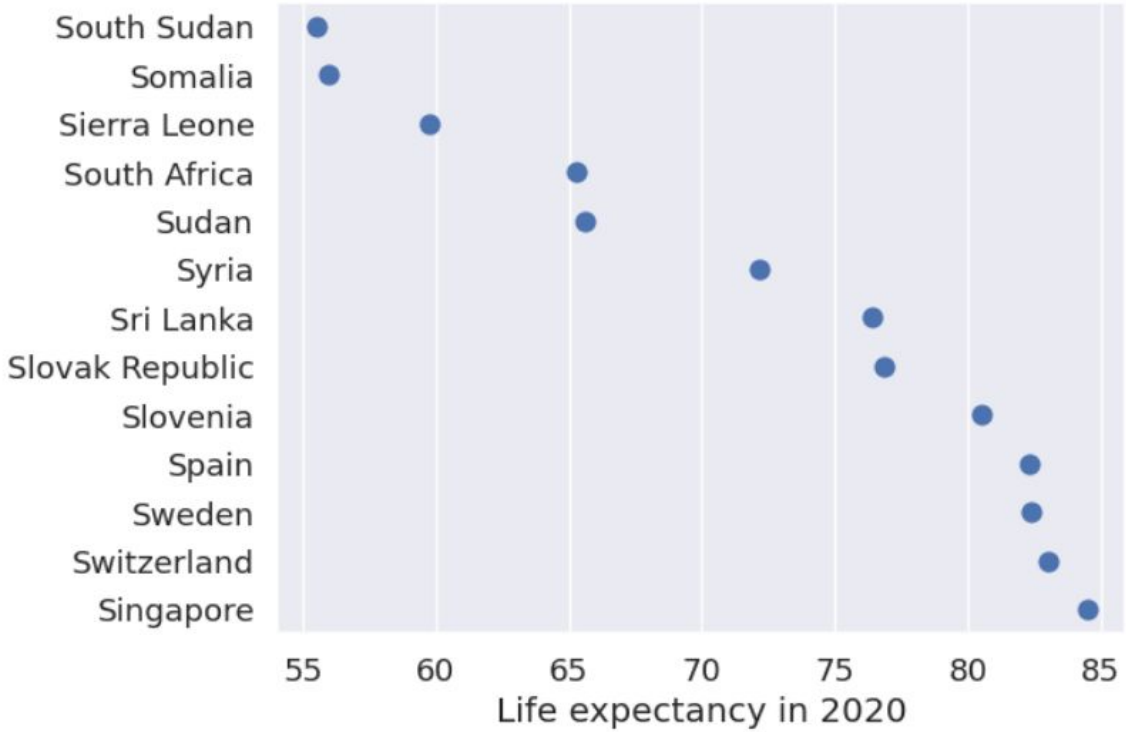
Dot plot

As bar graph but only dots shown at the top of the bar

Less clutter

X-axis does not need to start at 0 - better use of space

Can use multiple colors

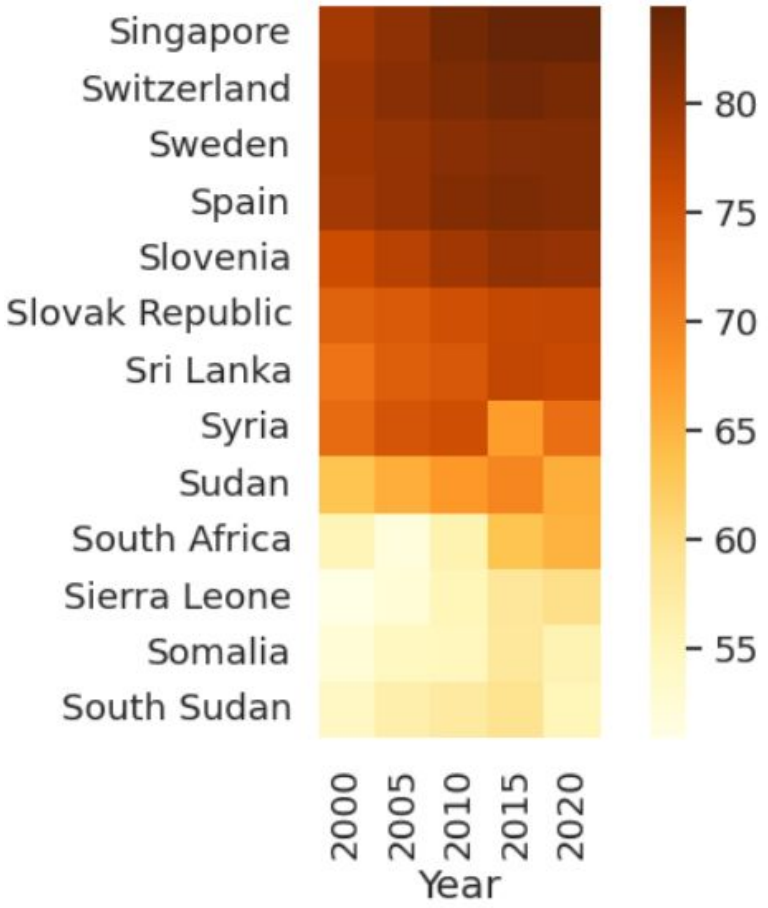


Heatmap

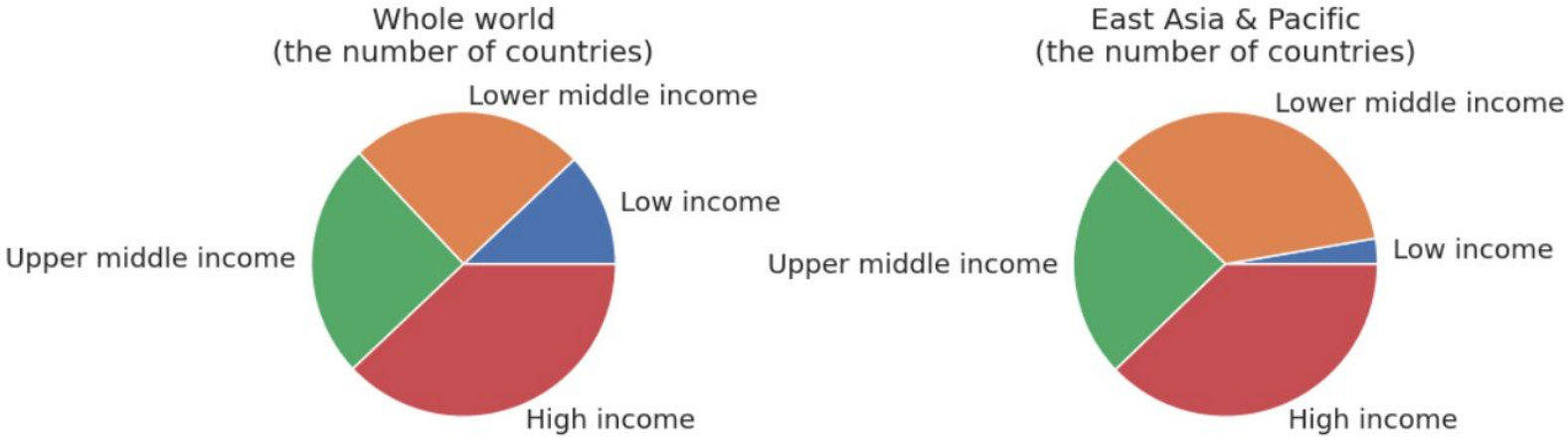
Both axes categorical

Numerical value shown in a color scale

Compact display, but color scales harder to read

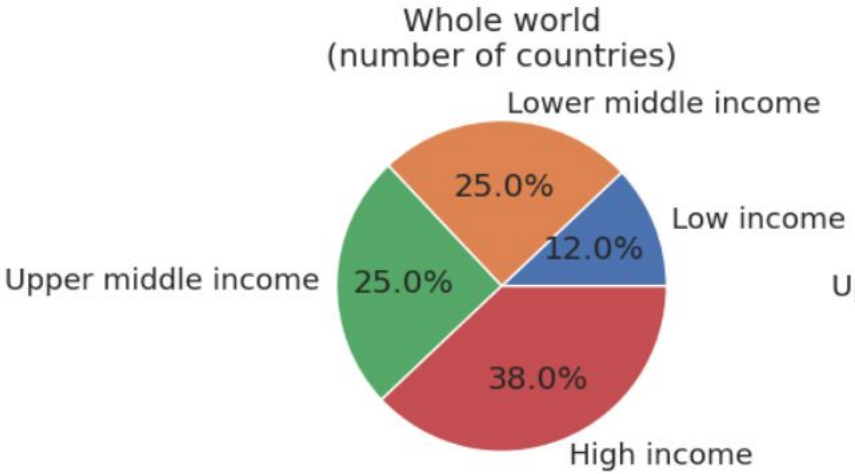


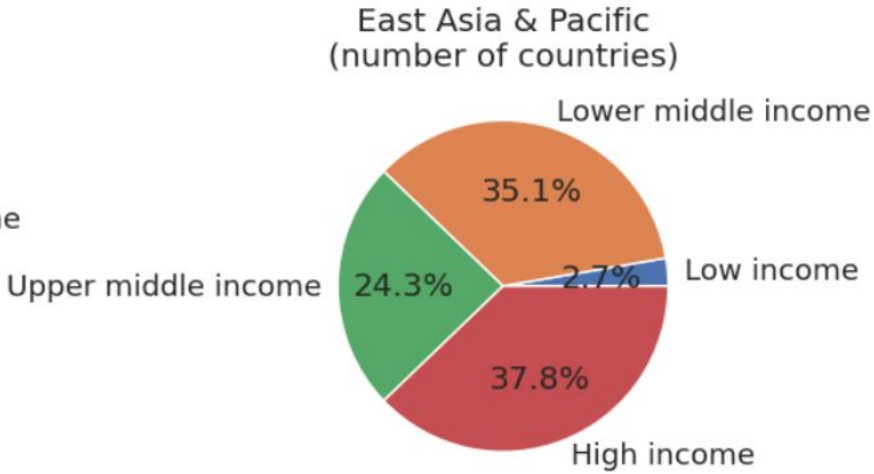
Pie chart (koláčový graf)



Obvious that percentages displayed
Very large values are easy to see (here high income)
Hard to compare similar values to each other
Space use not good

Pie chart with values labeled



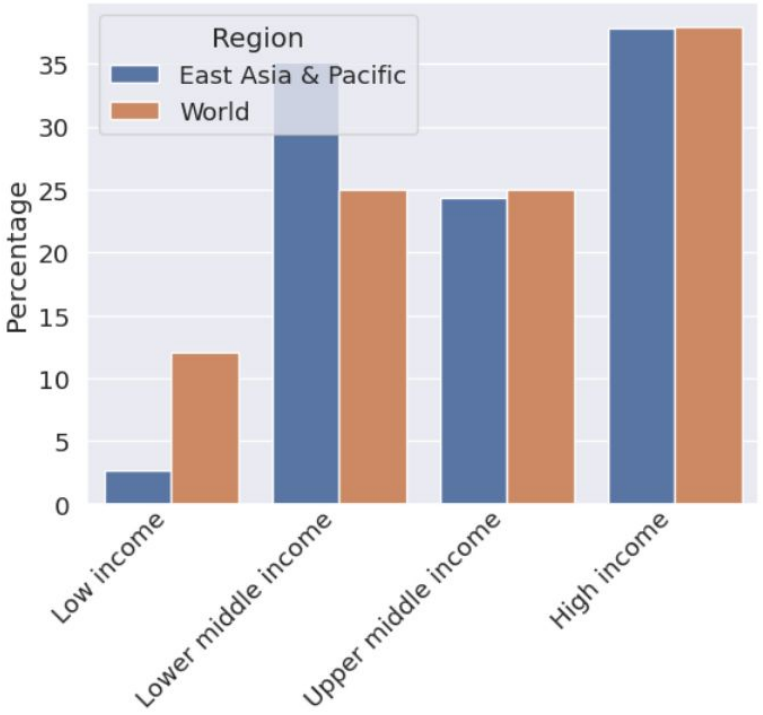


Easier to compare but still not ideal
Labeling values also useful in other types of graphs

Stacked (skladaný) bar graph instead of pie chart

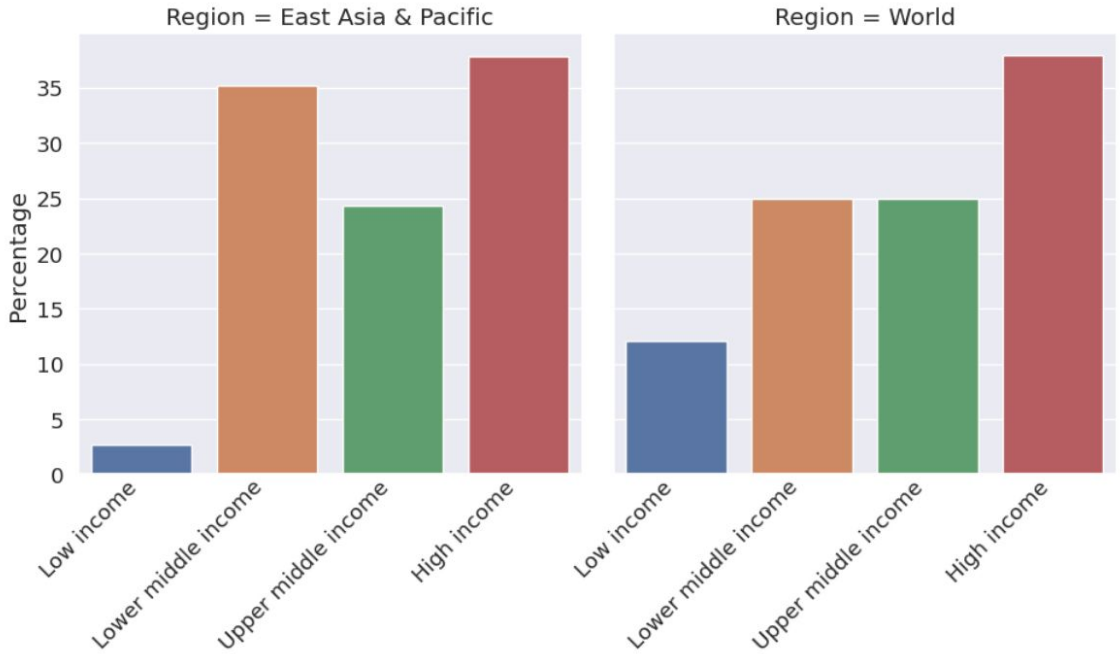
Rectangles easier to compare than wedges
Benefits from labeled values
Middle colors hard to compare across bars
Similar idea: stacked area plot (change in percentages over time)

Colored bar graph instead of pie chart



Easy to compare East Asia vs whole world.
Not obvious that we show parts of a whole.

Colored bar graph instead of pie chart



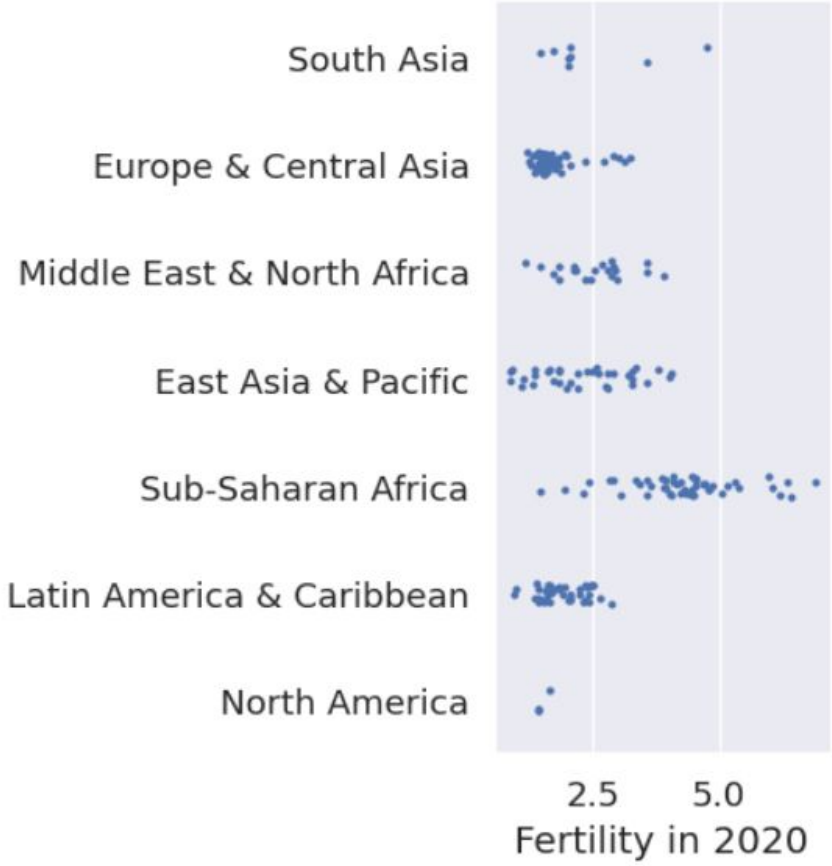
Easy to compare income groups within region

Strip plot

One axis categorical

Other axis shows individual data points

Jitter added in categorical axis to avoid point overlap

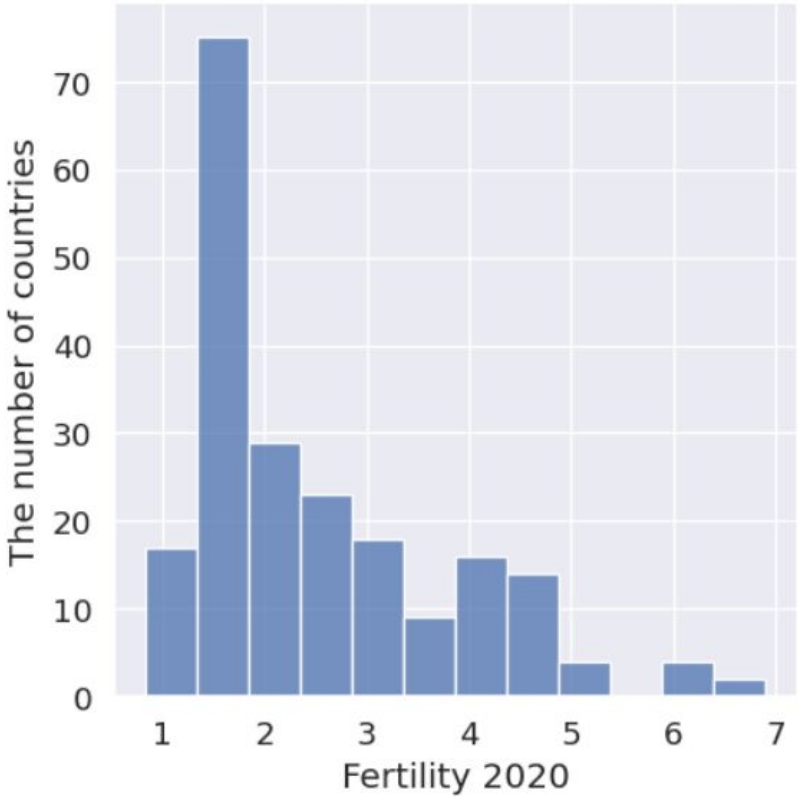


Histogram

For 1D numerical data

Split values into bins, show bin sizes as bar graph

We could use colors to display 2 or more histograms



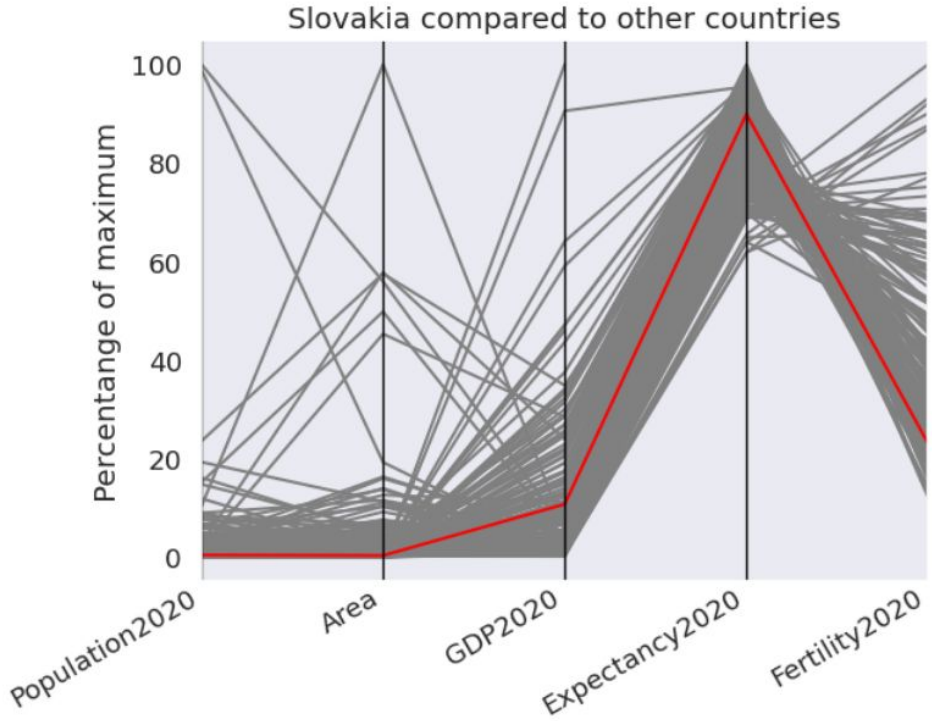
Parallel coordinates

Good for multidimensional numerical data

Each column one dimension

Here scaled as % of maximum value

Hard to see individual lines, but can show trends, compare groups shown in color or selected data point vs others

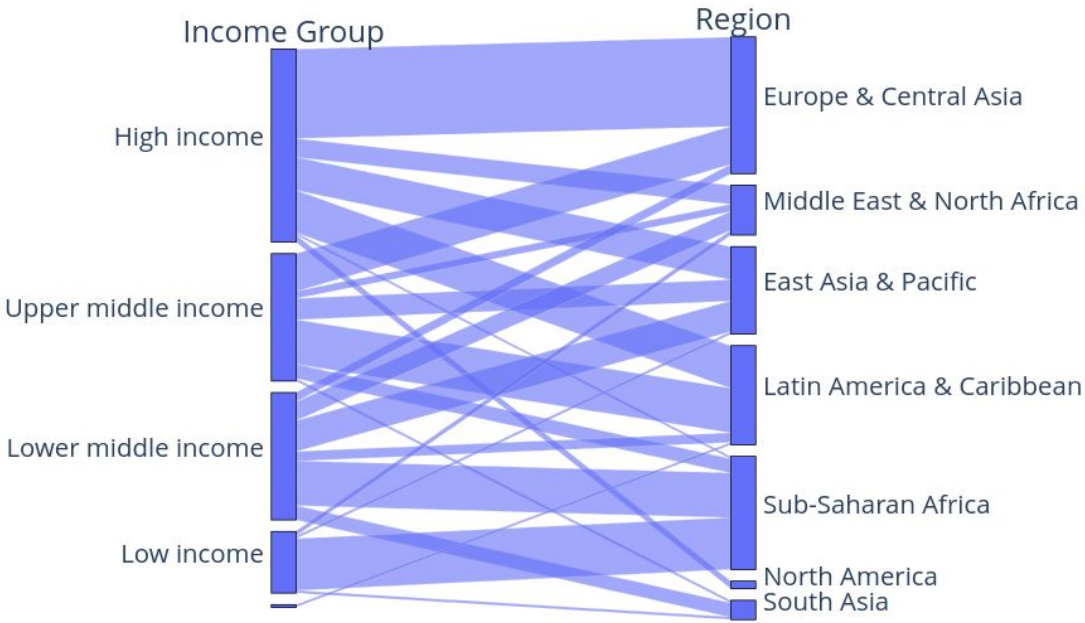


Parallel categories

Good for multidimensional categorical data

Each column one dimension

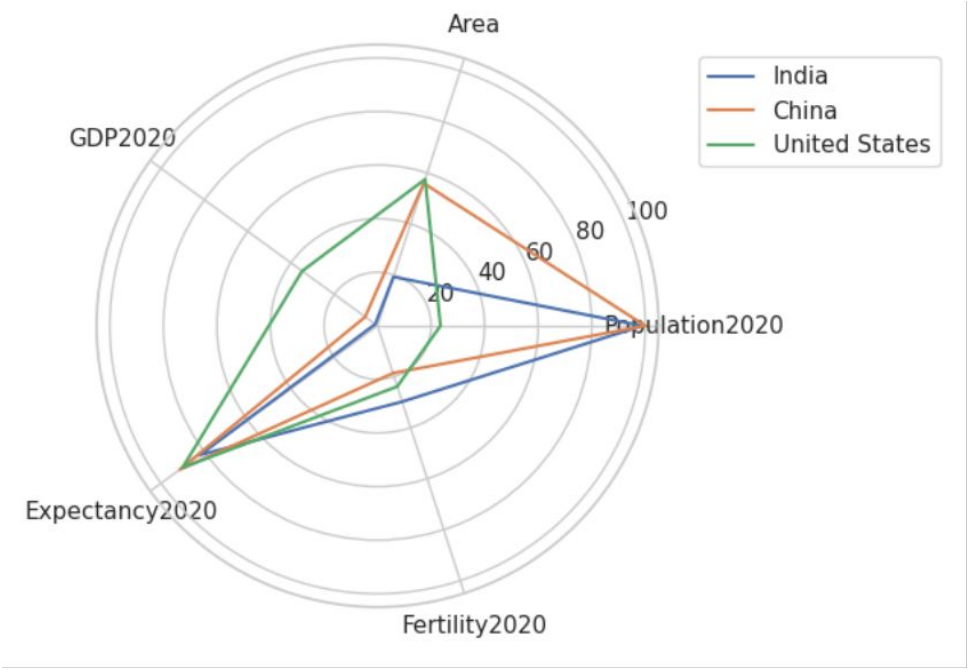
The widths of ribbons correspond to the number of countries



Radar chart (radarový graf)

Hard-to-read version of parallel coordinates

Perhaps some justification in cyclical domains, such as average temperature in months of a year



Now some Python

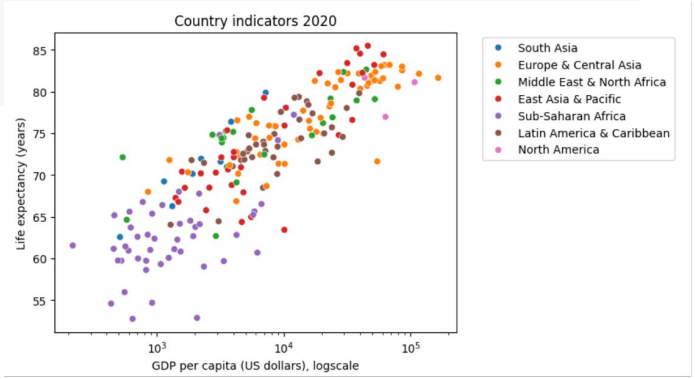
Overview of libraries

- Matplotlib
- Seaborn: an extension of Matplotlib, convenient for many types of plots
- Plotly: basic usage similar to Seaborn, plots interactive by default

Part of the main table `countries`

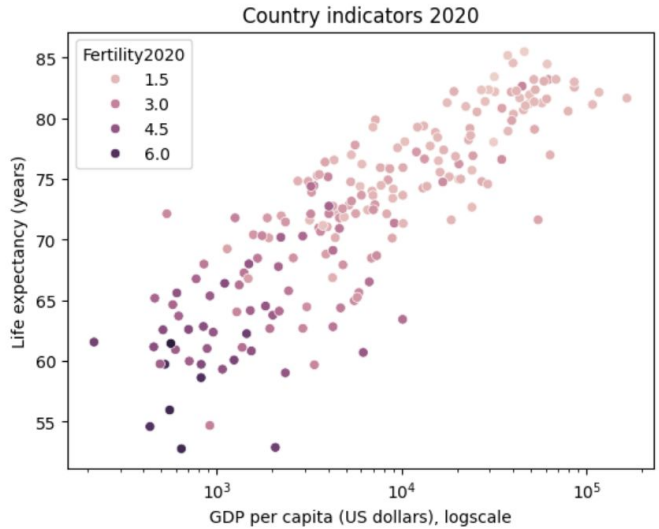
Country	ISO3	Region	Income Group	Population2000	Population2010	Population2020	Area	GDP2000	GDP2010
Afghanistan	AFG	South Asia	Low income	19542983.0	28189672.0	38972231.0	652860.0	NaN	562.499219
Albania	ALB	Europe & Central Asia	Upper middle income	3089026.0	2913021.0	2837849.0	28750.0	1126.683340	4094.349686
Algeria	DZA	Middle East & North Africa	Lower middle income	30774621.0	35856344.0	43451666.0	2381741.0	1780.376063	4495.921476
American Samoa	ASM	East Asia & Pacific	High income	58229.0	54849.0	46189.0	200.0	NaN	10446.863206
Andorra	AND	Europe & Central Asia	High income	66097.0	71519.0	77699.0	470.0	21620.465102	48237.890541

```
# create plot using Seaborn
axes = sns.scatterplot(data=countries, x='GDP2020', y='Expectancy2020',
                       hue='Region')
# set plot properties using methods from Matplotlib
axes.set_xlabel('GDP per capita (US dollars), logscale')
axes.set_ylabel('Life expectancy (years)')
axes.set_title('Country indicators 2020')
axes.semilogx()
# place legend outside the plot:
axes.legend(bbox_to_anchor=(1.05, 1), loc=2)
pass
```

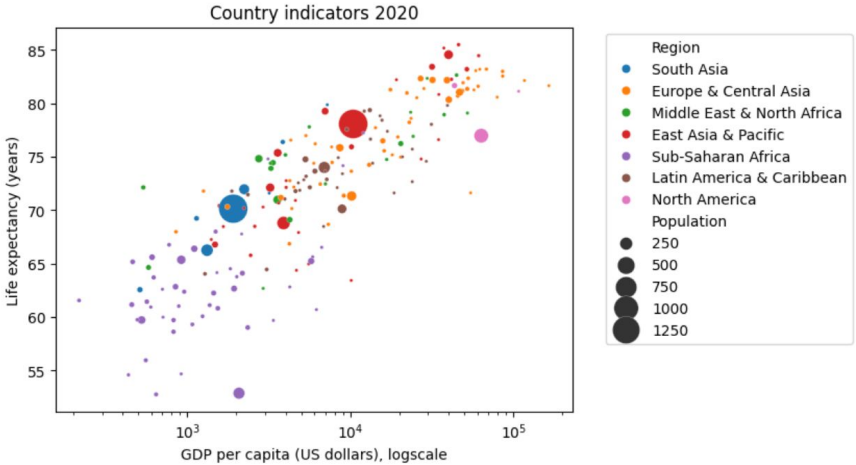


```
# The same plot in Plotly
# We want to use index (country name) in the figure for tooltip info
# therefore we create a temporary table with column Country instead of index
temp_table = countries.reset_index()
# how to rename automated axis labels
fig_labels = {'GDP2020':'GDP per capita (US dollars), logscale',
              'Expectancy2020':'Life expectancy (years)'}
# create Plotly plot, add countyr name to tooltip data
fig = px.scatter(data_frame=temp_table,
                 x="GDP2020", y="Expectancy2020", color="Region",
                 hover_data=['Country'], log_x=True,
                 labels = fig_labels)
fig.show()
```

```
# Seaborn automatically detects if the column used as hue is categorical or numerical
axes = sns.scatterplot(data=countries, x='GDP2020', y='Expectancy2020',
                       hue='Fertility2020')
axes.set_xlabel('GDP per capita (US dollars), logscale')
axes.set_ylabel('Life expectancy (years)')
axes.set_title('Country indicators 2020')
axes.semilogx()
pass
```

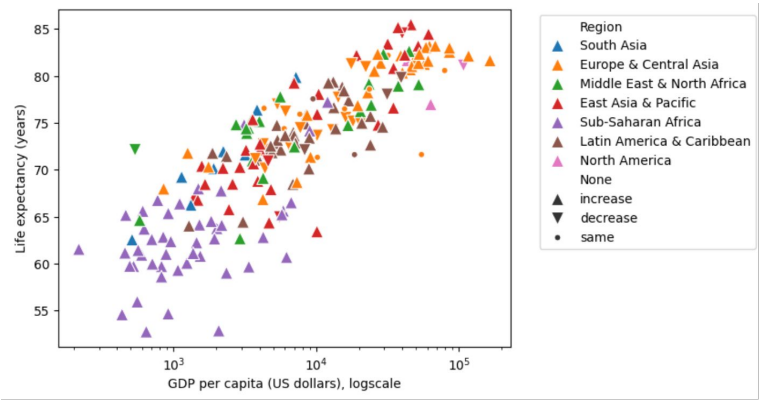


```
# add a column representing population in millions to table countries
countries['Population'] = countries['Population2020'] / 1e6
# create the plot
# parameter sizes sets the minimum and maximum point size to be used
axes = sns.scatterplot(data=countries,
                       x='GDP2020', y='Expectancy2020', hue='Region',
                       size='Population', sizes=(5, 400))
# set titles, log axes and legend location as before
```



```python
# compute relative differences in population between years 2010 and 2020
diff = (countries.Population2020 - countries.Population2010) / countries.Population2010
# new series with values 'increase', 'decrease' and 'same'
diff_class = diff.apply(lambda x : 'decrease' if x < -0.01
                        else 'increase' if x > 0.01 else 'same')

# create plot
# parameter s sets marker size
axes = sns.scatterplot(data=countries,
                       x='GDP2020', y='Expectancy2020', hue='Region',
                       style=diff_class, s=100,
                       markers={'increase':'^', 'decrease':'v', 'same':'.'})
```

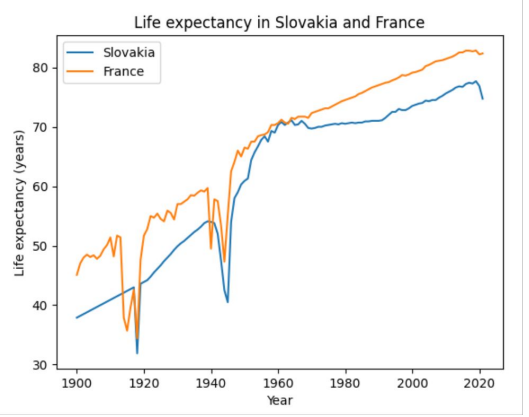


```
display(life_exp_years)
```

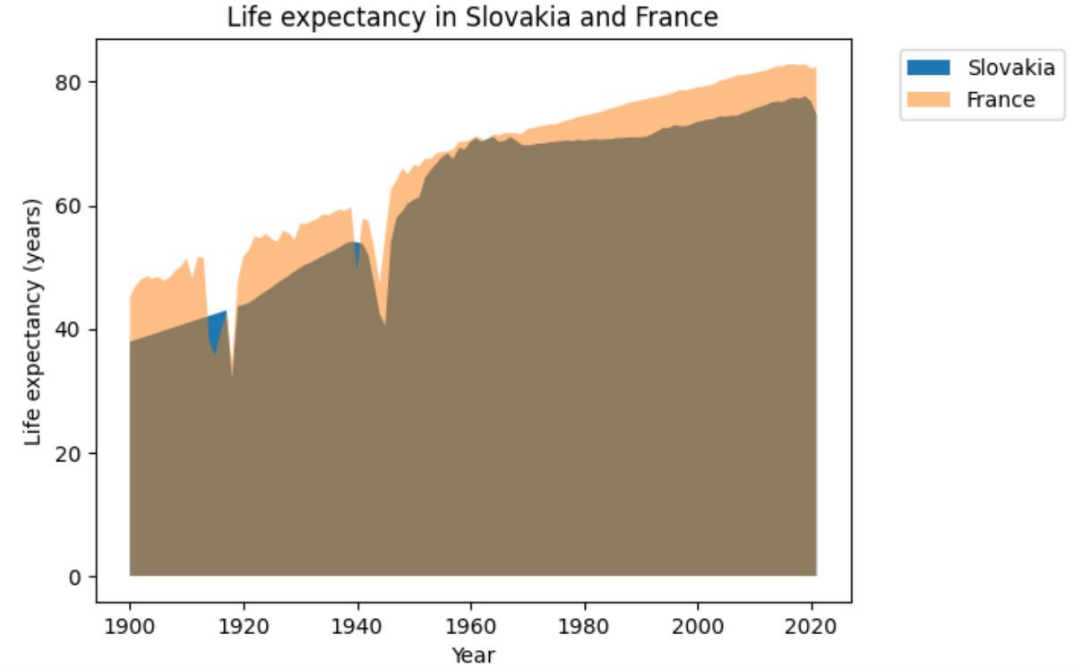
Country	1900	1901	1902	1903	1904	1905	1906	1907	1908	1909	...	2
Afghanistan	29.4	29.5	29.5	29.6	29.7	29.7	29.8	29.9	29.9	30.0	...	
Albania	35.4	35.4	35.4	35.4	35.4	35.4	35.4	35.4	35.4	35.4	...	
Algeria	30.2	30.3	30.4	31.4	25.4	28.1	29.6	29.5	29.5	31.0	...	
Angola	29.0	29.1	29.2	29.3	29.3	29.4	29.4	29.5	29.6	29.7	...	
Antigua and Barbuda	33.8	33.8	33.8	33.8	33.8	33.8	33.8	33.8	33.8	33.8	...	
...	...	...	...	...	...	...	...	...	...	...	...	...

```python
# list of numerical years from column names
years = [int(x) for x in life_exp_years.columns]

figure, axes = plt.subplots()
# plot two lines
axes.plot(years, life_exp_years.loc['Slovak Republic'], label='Slovakia')
axes.plot(years, life_exp_years.loc['France'], label='France')
# plot settings
axes.set_xlabel('Year')
axes.set_ylabel('Life expectancy (years)')
axes.set_title('Life expectancy in Slovakia and France')
axes.legend()
pass
```



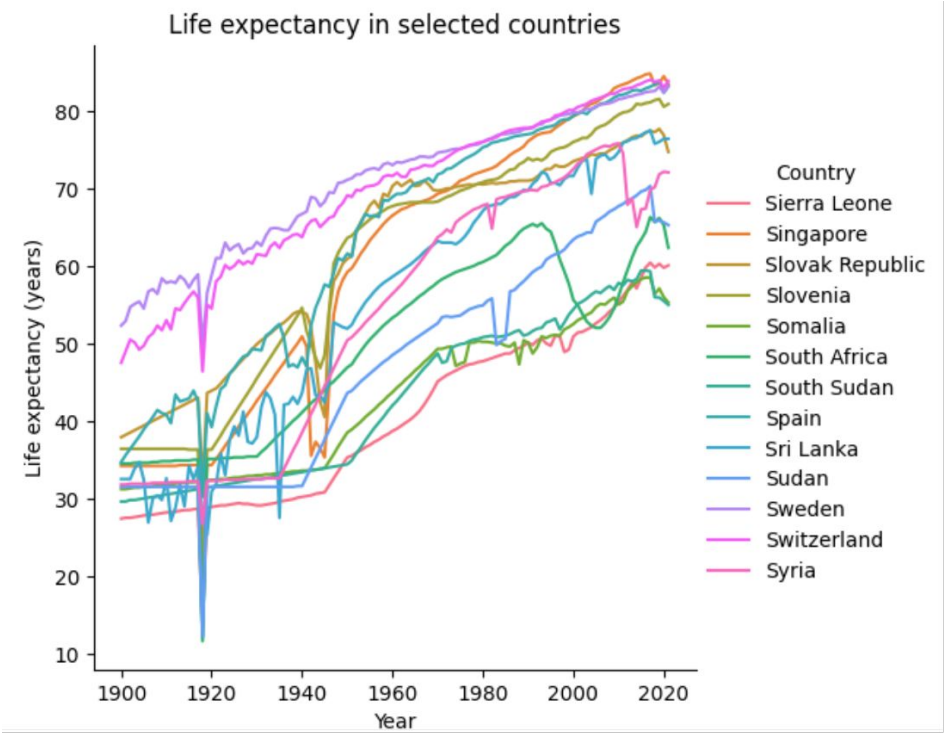
```
figure, axes = plt.subplots()
# two filled areas, the second is semi-transparent
axes.fill_between(years, 0, life_exp_years.loc['Slovak Republic'], label='Slovakia')
axes.fill_between(years, 0, life_exp_years.loc['France'], label='France', alpha=0.5)
# plot settings as before...
```



```
# Lines for many countries easy in Seaborn
# ... works better with long table
life_exp_sel_long = (
    life_exp_sel.reset_index()
    .melt(id_vars=['Country'])
    .rename(columns={'variable':'Year', 'value':'Expectancy'})
    .astype({'Year':'int32'})
  )
display(life_exp_sel_long)
```

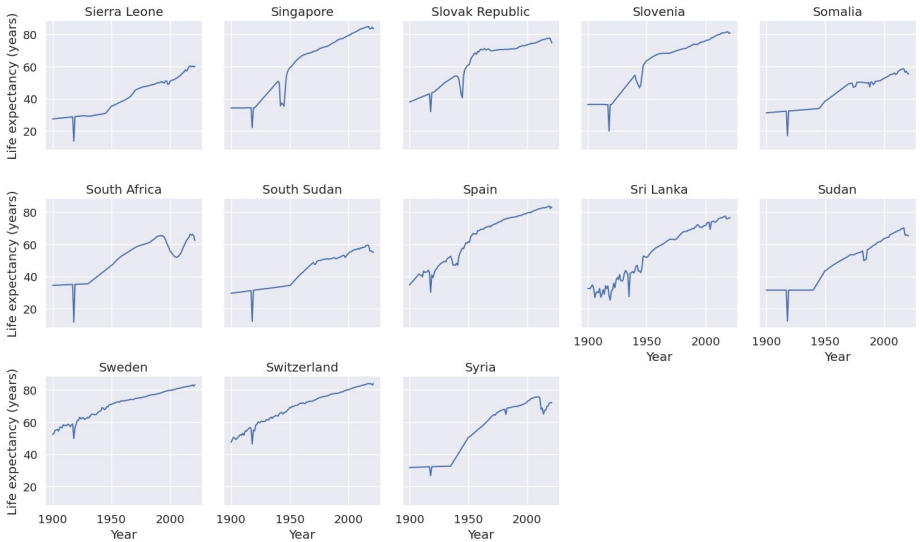
	Country	Year	Expectancy
0	Sierra Leone	1900	27.400000
1	Singapore	1900	34.200000
2	Slovak Republic	1900	37.900000
3	Slovenia	1900	36.400000
4	Somalia	1900	31.200000
...	...	...	...
1581	Sri Lanka	2021	76.399000
1582	Sudan	2021	65.267000
1583	Sweden	2021	83.156098
1584	Switzerland	2021	83.851220
1585	Syria	2021	72.063000

```
grid = sns.relplot(data=life_exp_sel_long, x='Year', y='Expectancy',
                   hue='Country', kind="line")
grid.set_axis_labels('Year', 'Life expectancy (years)')
grid.set(title='Life expectancy in selected countries')
```



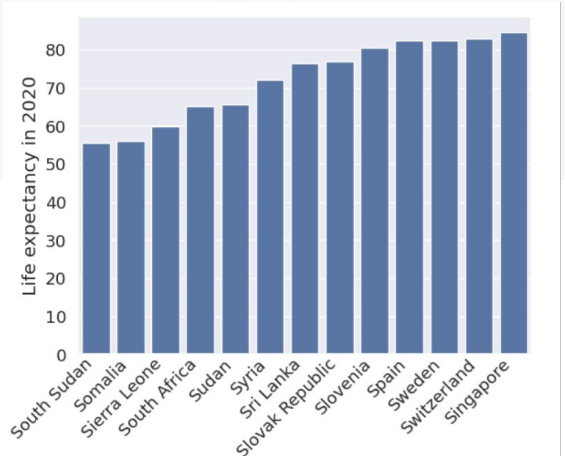
```
sns.set_theme(font_scale=1.2)
# create a grid of small multiple plots
# use col_wrap=5 columns
grid = sns.relplot(data=life_exp_sel_long,
                   x='Year', y='Expectancy', col='Country',
                   col_wrap=5, kind="line", height=3, aspect=1)

grid.set_axis_labels('Year', 'Life expectancy (years)')
grid.set_titles("{col_name}")  # title of each plot will be country name
```



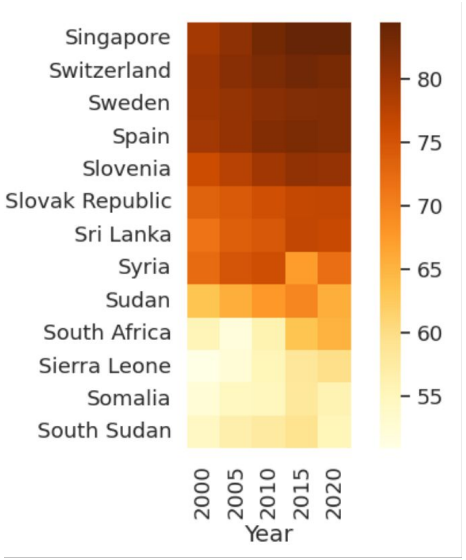
```
def rotate_bar_labels(axes, angle=45):
    """Auxiliary function for rotating bar plot labels by 45 degrees"""
    axes.tick_params(axis='x', labelrotation=angle, pad=-5)
    plt.setp(axes.get_xticklabels(), ha='right')
```

```
# sorting
life_exp_sel_2020_sorted = life_exp_sel_2020.sort_values('Expectancy')
# plotting
axes = sns.barplot(data=life_exp_sel_2020_sorted,
                   x='Country', y='Expectancy', color="C0")
axes.set_ylabel("Life expectancy in 2020")
axes.set_xlabel(None)
rotate_bar_labels(axes)
```



```
# set of years to be used
sel_years={2000, 2005, 2010, 2015, 2020}
# create desired wide table
life_exp_sel_wide = (life_exp_sel_long.query('Year in @sel_years')
                     .pivot(index='Country', columns='Year', values='Expectancy')
                     .sort_values(2020, ascending=False))
# show the table
display(life_exp_sel_wide)
```

```
axes = sns.heatmap(data=life_exp_sel_wide, square=True, cmap="YlOrBr")
```



Year	2000	2005	2010	2015	2020
Country					
Singapore	79.3	81.1	83.2	84.4	84.465854
Switzerland	80.1	81.5	82.5	83.5	83.000000
Sweden	79.8	80.6	81.5	82.2	82.356098
Spain	79.4	80.5	82.0	82.6	82.331707
Slovenia	76.0	77.7	79.5	80.8	80.531707
Slovak Republic	73.5	74.3	75.6	76.7	76.865854