# Effective Programming Practices for Economists

# Data management with pandas

## Inspecting and summarizing data

Janoś Gabler and Hans-Martin von Gaudecker

# Motivation

- So far we have looked at tiny DataFrames
- Real datasets don’t fit on a screen
- Need quick ways to:
  - Look at subsets
  - Calculate summary statistics
  - Plot distributions

# Example

| | country | continent | year | life_exp | pop | gdp_per_cap | iso_alpha | iso_num |
|---|---|---|---|---|---|---|---|---|
| **0** | Afghanistan | Asia | 1952 | 28.801 | 8425333 | 779.445314 | AFG | 4 |
| **1** | Afghanistan | Asia | 1957 | 30.332 | 9240934 | 820.85303 | AFG | 4 |
| **2** | Afghanistan | Asia | 1962 | 31.997 | 10267083 | 853.10071 | AFG | 4 |
| **3** | Afghanistan | Asia | 1967 | 34.02 | 11537966 | 836.197138 | AFG | 4 |
| **4** | Afghanistan | Asia | 1972 | 36.088 | 13079460 | 739.981106 | AFG | 4 |
| **5** | Afghanistan | Asia | 1977 | 38.438 | 14880372 | 786.11336 | AFG | 4 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **1699** | Zimbabwe | Africa | 1987 | 62.351 | 9216418 | 706.157306 | ZWE | 716 |
| **1700** | Zimbabwe | Africa | 1992 | 60.377 | 10704340 | 693.420786 | ZWE | 716 |
| **1701** | Zimbabwe | Africa | 1997 | 46.809 | 11404948 | 792.44996 | ZWE | 716 |
| **1702** | Zimbabwe | Africa | 2002 | 39.989 | 11926563 | 672.038623 | ZWE | 716 |
| **1703** | Zimbabwe | Africa | 2007 | 43.487 | 12311143 | 469.709298 | ZWE | 716 |

# Summarize an entire DataFrame

assume that `df` is the full gapminder data

```
>>> relevant = ["life_exp", "pop", "gdp_per_cap"]
>>> df[relevant].describe()
```

| | life_exp | pop | gdp_per_cap |
|---|---|---|---|
| **count** | 1704.00 | 1704.00 | 1704.00 |
| **mean** | 59.47 | 29601212.32 | 7215.33 |
| **std** | 12.92 | 106157896.74 | 9857.45 |
| **min** | 23.60 | 60011.00 | 241.17 |
| **25%** | 48.20 | 2793664.00 | 1202.06 |
| **50%** | 60.71 | 7023595.50 | 3531.85 |
| **75%** | 70.85 | 19585221.75 | 9325.46 |
| **max** | 82.60 | 1318683096.00 | 113523.13 |

- `.describe` can summarize entire DataFrames
- Result is again a DataFrame
- Often good idea to select a subset of columns

# Calculate specific statistics

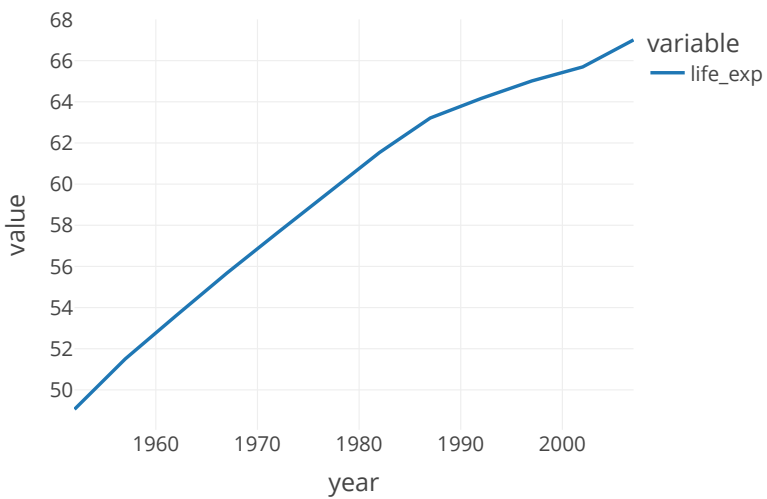
assume that `df` is the full gapminder data

```
>>> df["life_exp"].mean()
59.474439366197174

>>> df.groupby("year").mean()

year
1952    49.057620
1957    51.507401
1962    53.609249
...
```

- Standard summary statistics are implemented and named as expected:
  - `std`
  - `min` and `max`
  - `median` and `quantile`
- Vectorized and really fast implementations

# Quick plotting: Series

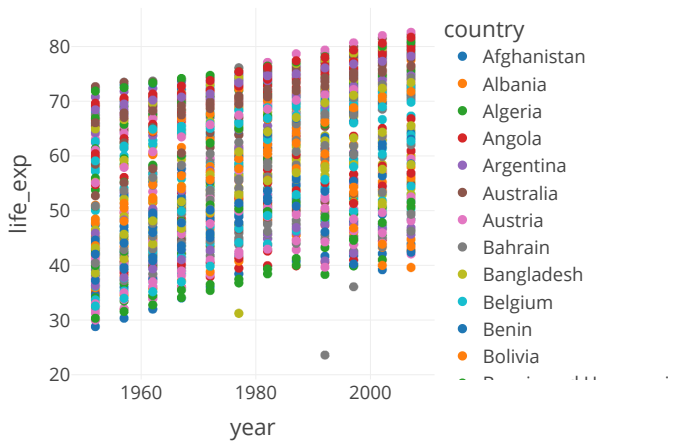
```
>>> pd.options.plotting.backend = "plotly"
>>> df.groupby("year")["life_exp"].mean().plot()
```




- Any Series has a `.plot` method
- Any Series has a `.hist` method
- Summary statistics based on groupby return Series which can again be plotted

# Quick plotting: DataFrames

```
>>> pd.options.plotting.backend = "plotly"
>>> df.plot.scatter(x="year", y="life_exp",
                    color="country")
```




- Any DataFrame has a `.plot` method
- Defaults to line plot, can access `.scatter` and many more
- Notebook gives you interactive plots

# Statistics for categorical data

```
>>> df["country"].unique()[:2]

<ArrowStringArrayNumpySemantics>
['Afghanistan', 'Albania']
Length: 2, dtype: string

>>> df["country"].value_counts().sort_index()[:2]

country
Afghanistan    12
Albania        12
Name: count, dtype: int64
```