Lecture 7
More Statistics

[Data Visualization · 1-DAV-105]

Lecture by Broňa Brejová

More details in the [notebook version]

Data for today

A new dataset:

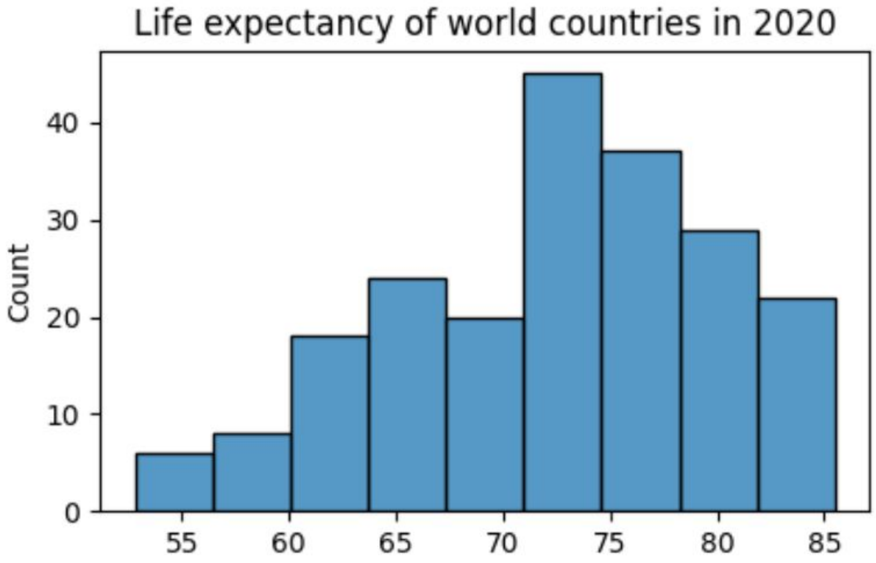
- an informal survey of preferences and opinions of young people done in 2013 among students of FSEV UK and their friends
- 1010 respondents, 150 question

Also our usual table of countries, namely columns

- life expectancy in 2020
- GDP per person in 2020
- region of the world

Histograms

An example of a histogram



What exactly is a histogram?
What kind of variables we use it for?

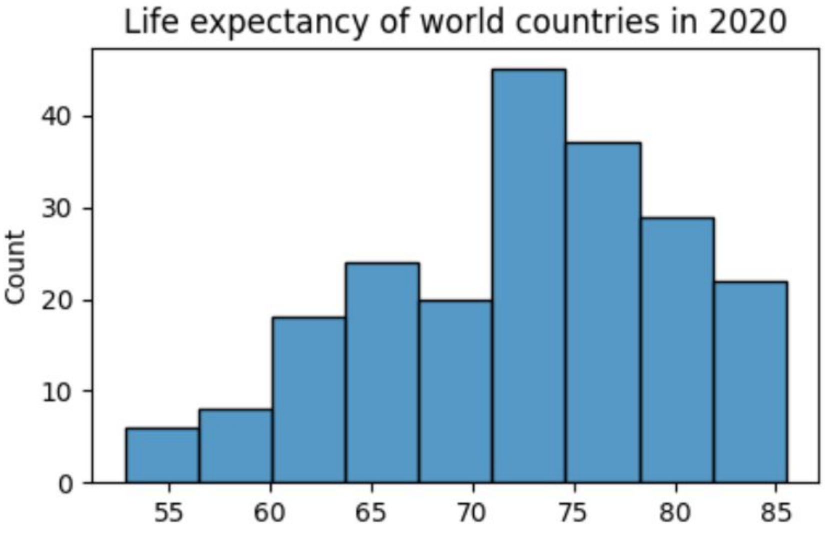
What can we find out from this histogram?

```
axes = sns.histplot(data=countries, x='Expectancy2020')
axes.set_title('Life expectancy of world countries in 2020')
axes.set_xlabel(None)
axes.figure.set_size_inches(5, 3)
```

Histograms

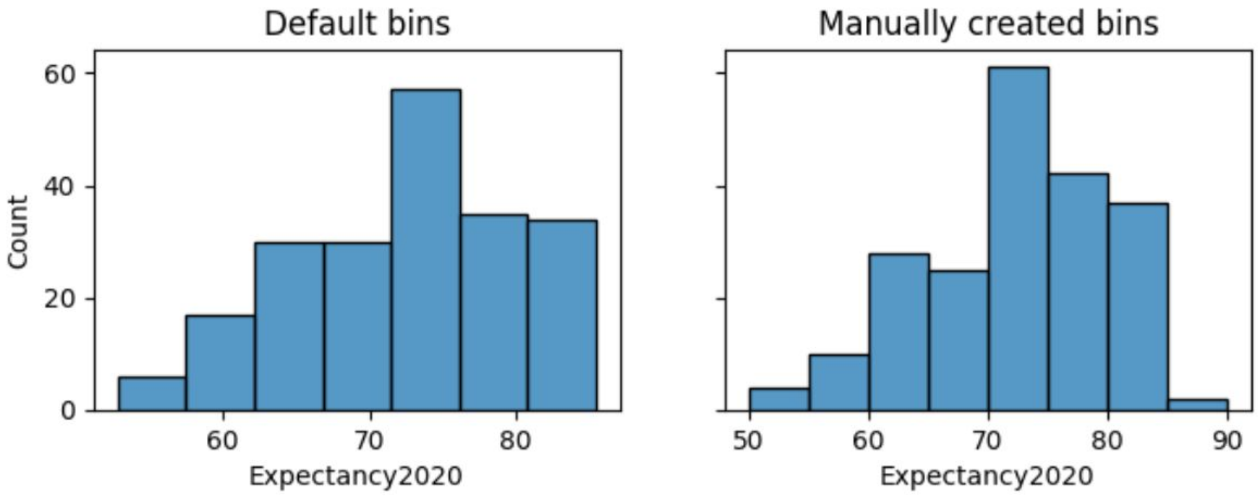
Histograms allow us to observe many aspects of the distribution of values of a variable:

- range of values, outliers
- central tendency
- unimodality / multimodality
- variance
- symmetry / skewness (šikmosť)



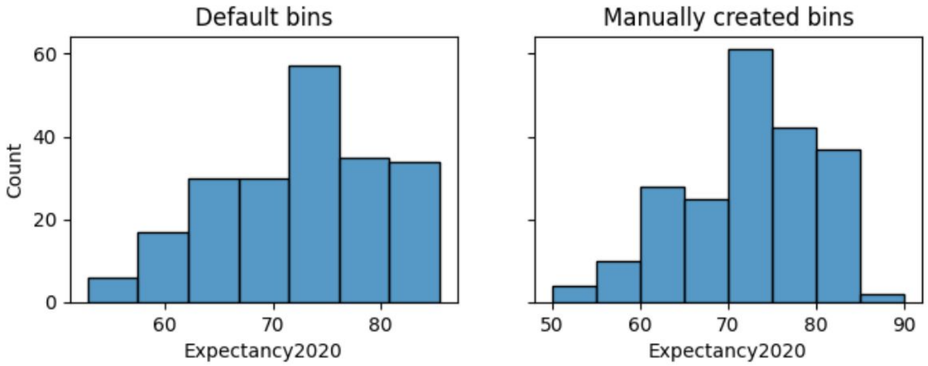
Custom bins

- Bins in Seaborn library: range of values split into equally sized intervals
- Often it is better to use round values at bin boundaries,
  e.g. intervals of 5 years 50-55, 55-60, 60-65,...



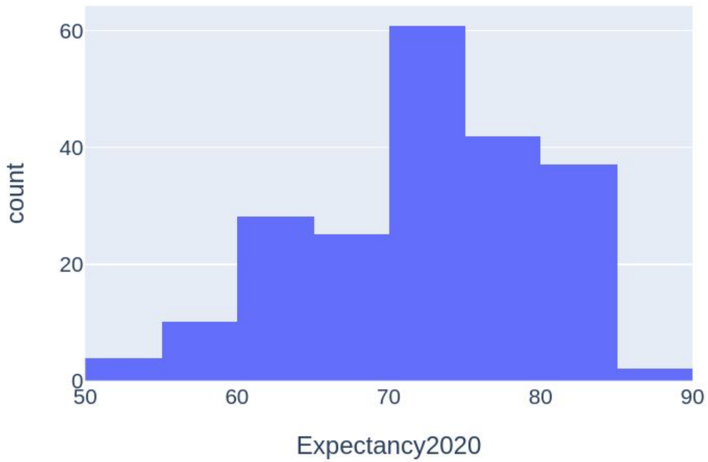
Custom bins in Seaborn

```
# the first plot has histogram with default bins of width 5
sns.histplot(data=countries, x='Expectancy2020', binwidth=5, ax=axes[0])
axes[0].set_title('Default bins')

# the second plot has manually set bin boundaries 50,55,60,...,90
sns.histplot(data=countries, x='Expectancy2020',
             bins=range(50, 95, 5), ax=axes[1])
axes[1].set_title('Manually created bins')
```

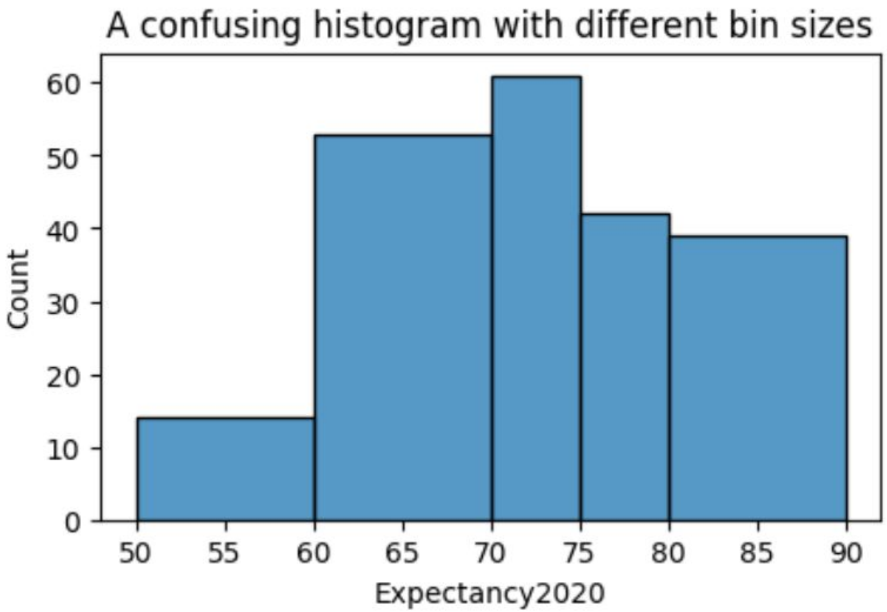


Plotly library creates more meaningful bins

```
# In Plotly, we specify the maximum number of bins.
# The library may choose a lower number to get "nice" bin boundaries
fig = px.histogram(countries, x="Expectancy2020",
                   nbins=8, width=500, height=350)
fig.show()
```



Use equally-sized bins



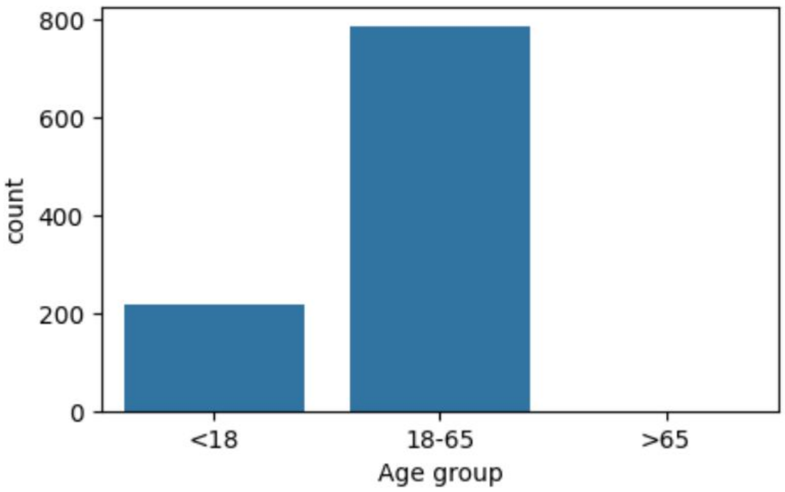

```
axes = sns.histplot(data=countries, x='Expectancy2020',
                    bins=[50, 60, 70, 75, 80, 90])
```

Use equally-sized bins

You may sometimes want special unequal bins
Example: age <18 years, 18-65 years, >65 years

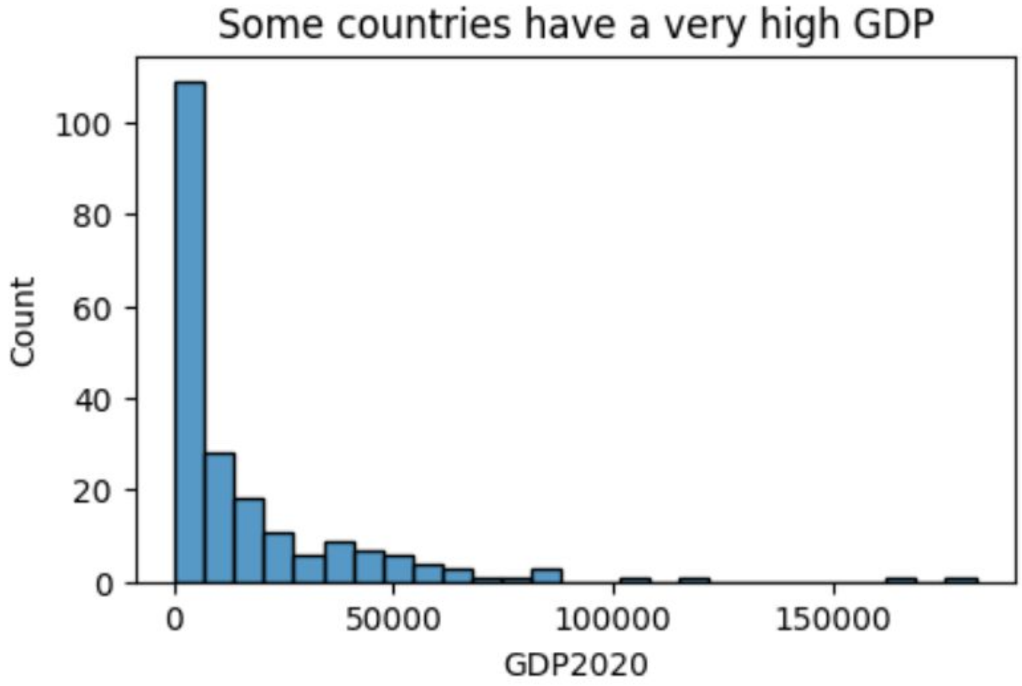
- Make a categorical variable
- Plot it as a bar graph
  (bars with equal width,
  spaces between bars)
- Clearly mark each bar

```
# split participants into 3 age groups
bin_ends = [0,18,65,150]
bin_labels = ['<18', '18-65', '>65']
age_groups = (pd.cut(fsev['Age'],
                     bins=bin_ends,
                     labels=bin_labels)
  .rename('Age group'))
# count participants in each group
age_counts = age_groups.value_counts()
# bar plot of group sizes
axes = sns.barplot(age_counts)
```



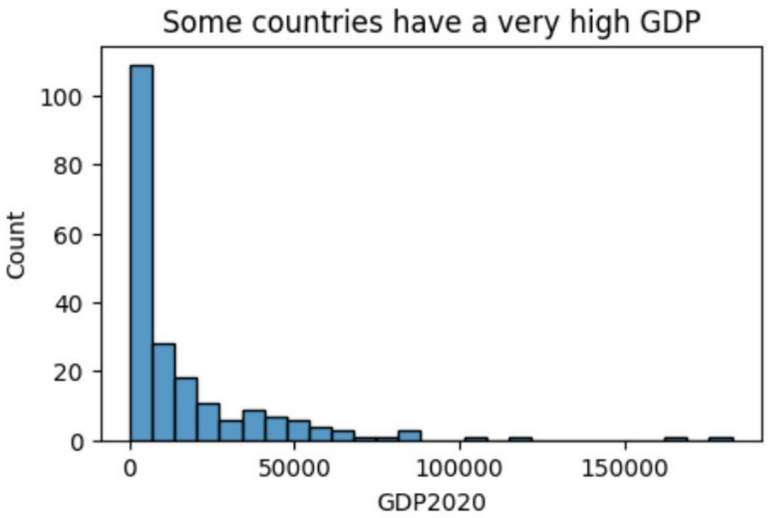
Outliers in histograms

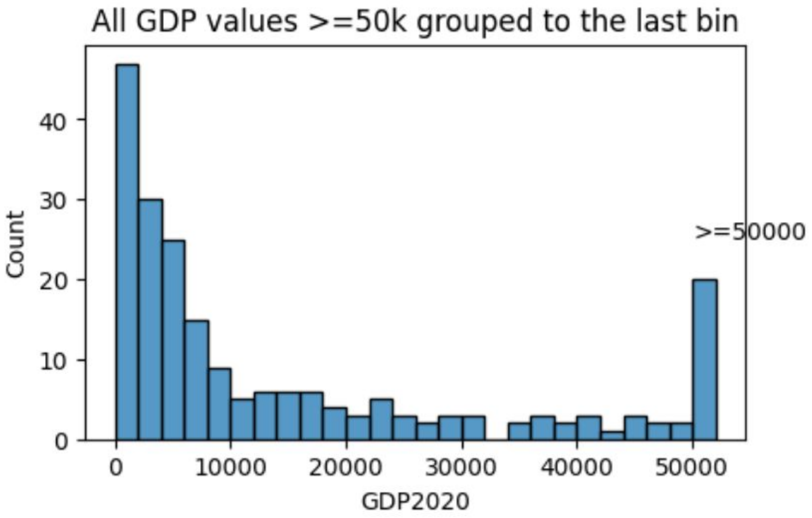
- Histograms are great for spotting outliers
- But outliers reduce the space given to more regular values
- Perhaps remove them in subsequent analysis



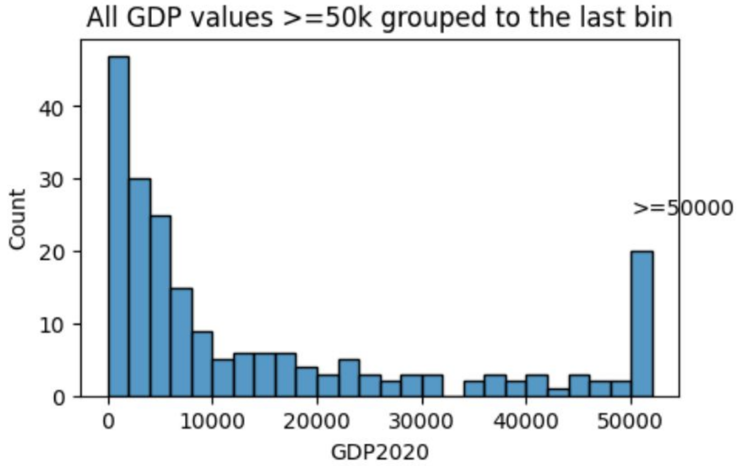
Removing outliers

- Remove them from the dataset if we believe them to be errors
- Or remove them from the plot only (`set_xlim` or custom bins, warn reader)
- Or clip values: place them to a clearly marked last bin





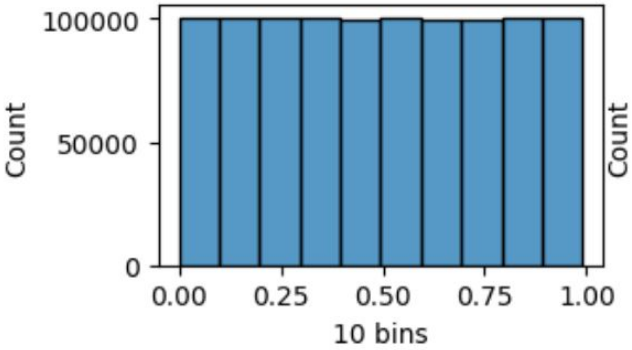
```
# replace values larger than 51k with 51k
gdp_clipped = countries['GDP2020'].clip(0, 51000)
# make histogram with manual bins, with last bin 50k-52k
axes = sns.histplot(x=gdp_clipped, bins=np.arange(0, 53000, 2000))
axes.figure.set_size_inches(5, 3)
# mention clipping in plot title
axes.set_title('All GDP values >=50k grouped to the last bin')
# also add a text label to the bin with clipped values
axes.text(x=50000, y=25, s='>=50000')
```



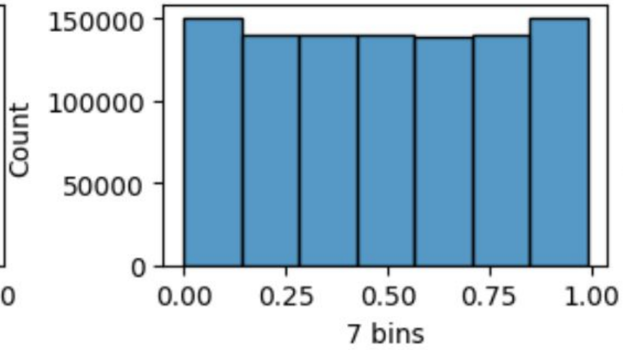
Problems with precision

When data contains a **small number of possible values** (integers or real numbers given with a small number of decimal points), we can get **artifacts** related to **different counts** of possible values falling to different bins.

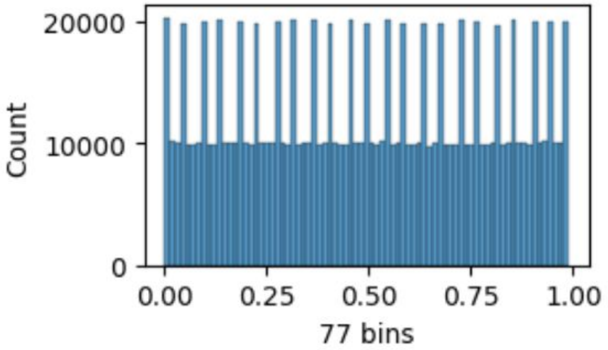
We plotted histograms of million points sampled from {0,0.01,0.02,…,0.99}.



Each bin 10 values



Each bin 14-15 values

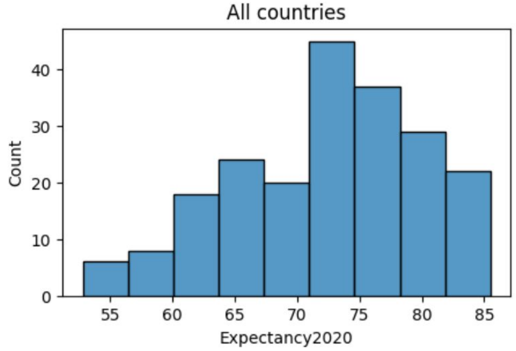


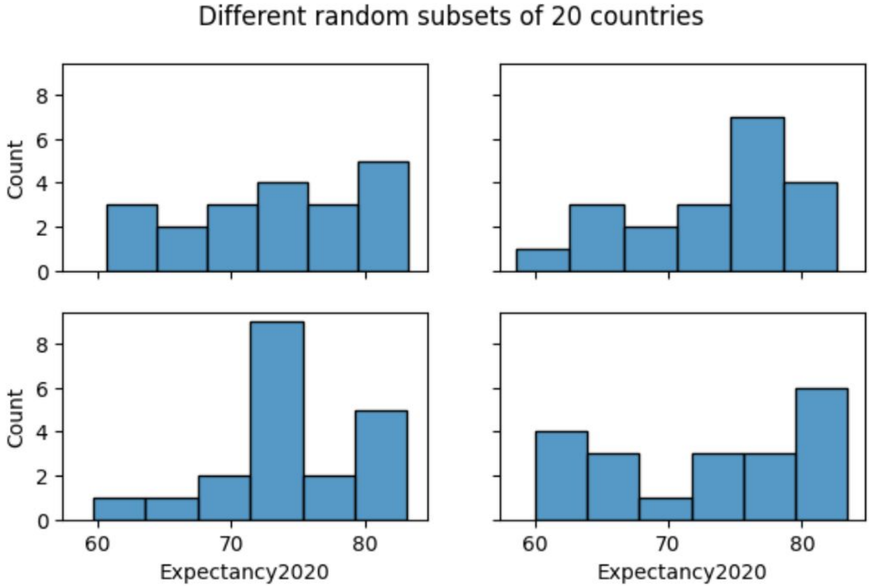
Each bin 1-2 values

Small samples

Any estimates (including histograms) from small samples are **subject to random noise**.

Example: expectancy for all countries / for random subsets of 20 countries each





Summary: Histogram bin size

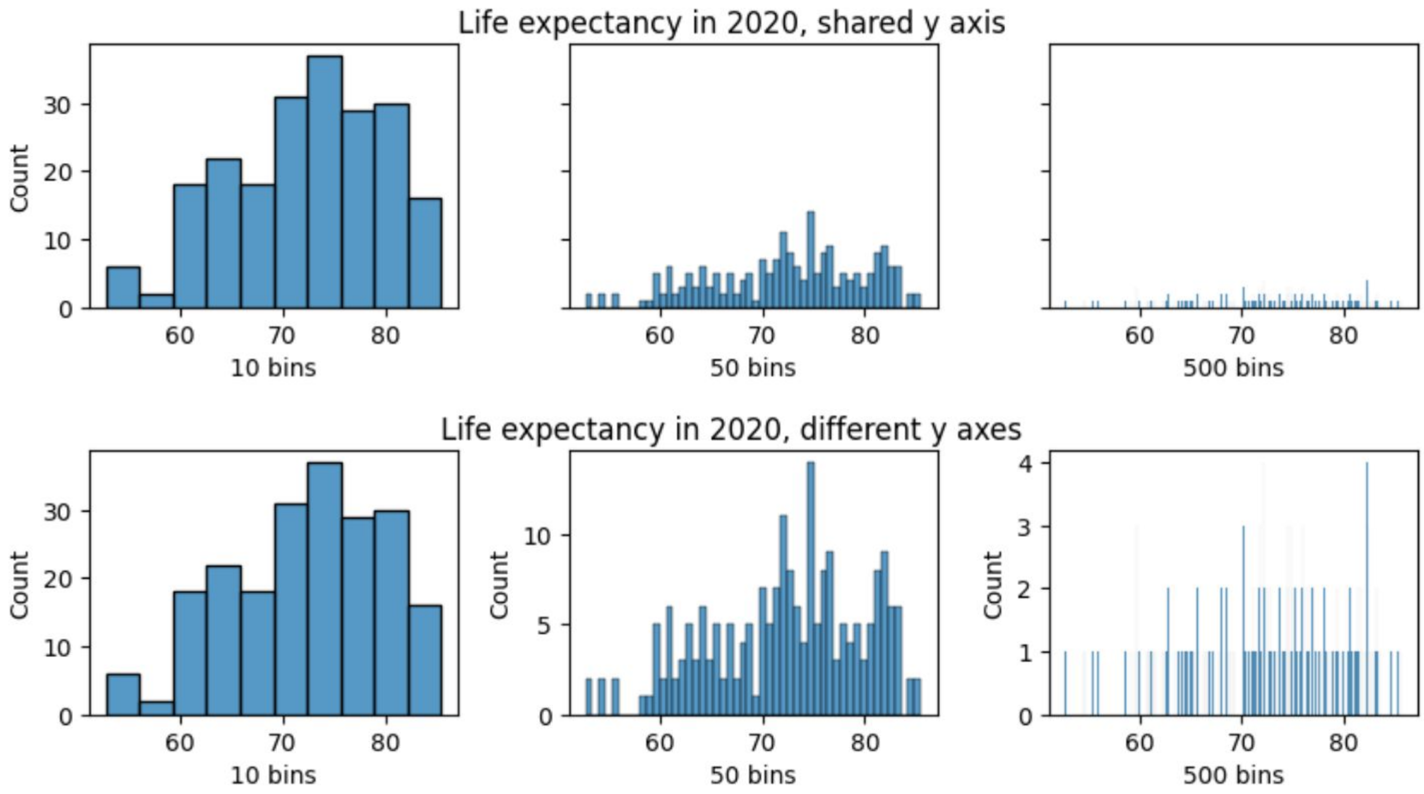
Smaller bins mean more details are visible, but some of those details may be artefacts:

- random fluctuations due to small number of points in the bin, or
- effects related to insufficient resolution of the data.

Thus choose bin size based on:

- the amount of data,
- the precision of input values,
- the meaningful resolution of the results.

Do we learn more from 50 or 500 bins than 10?

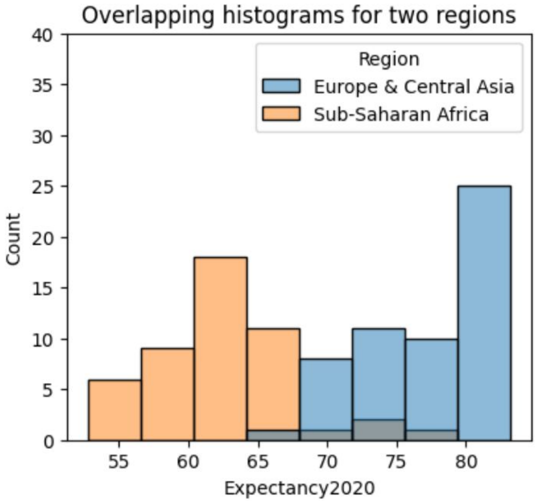


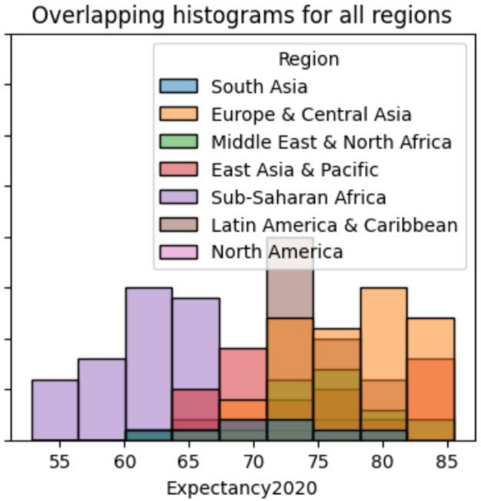
Comparing distributions with histograms

Comparing distributions with histograms

We can compare distributions of a numerical variable split into groups by a categorical variable.

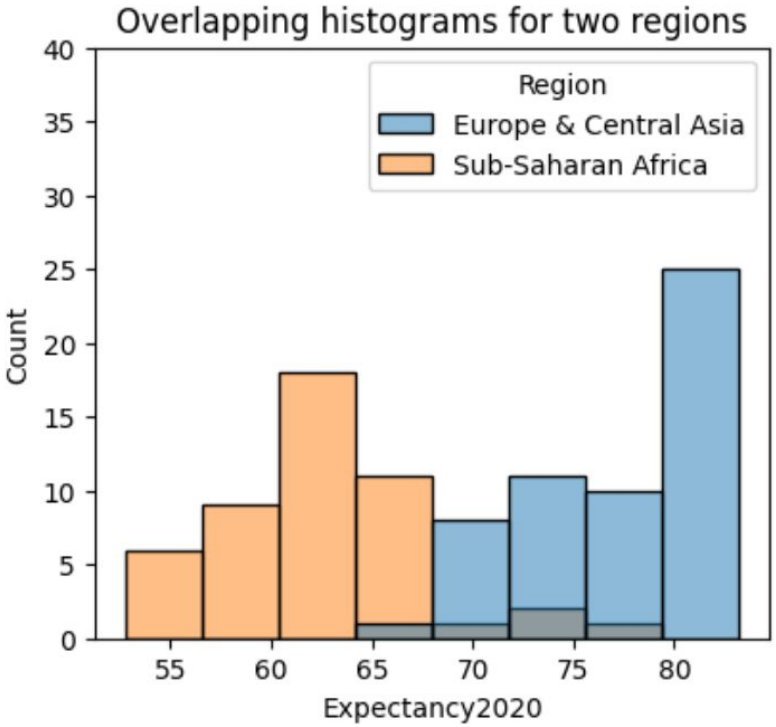
Example: life expectancy in different regions of the world.

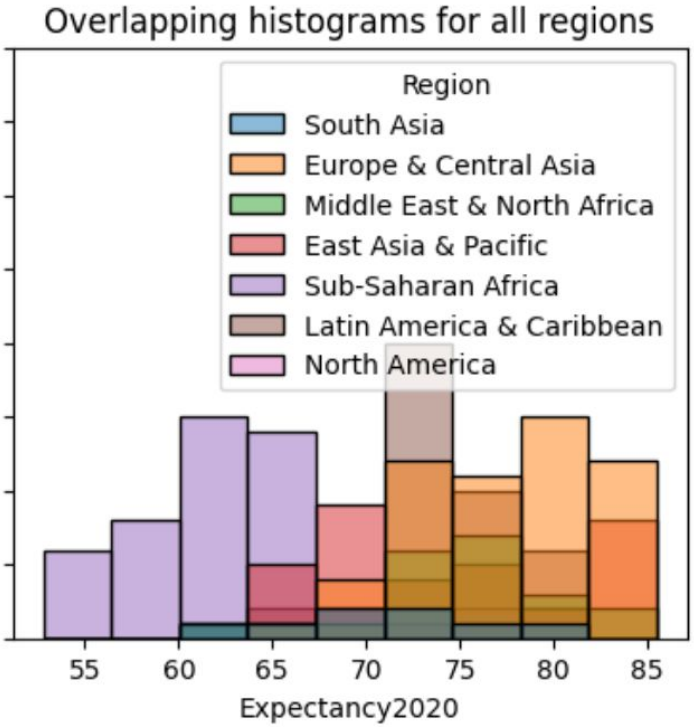




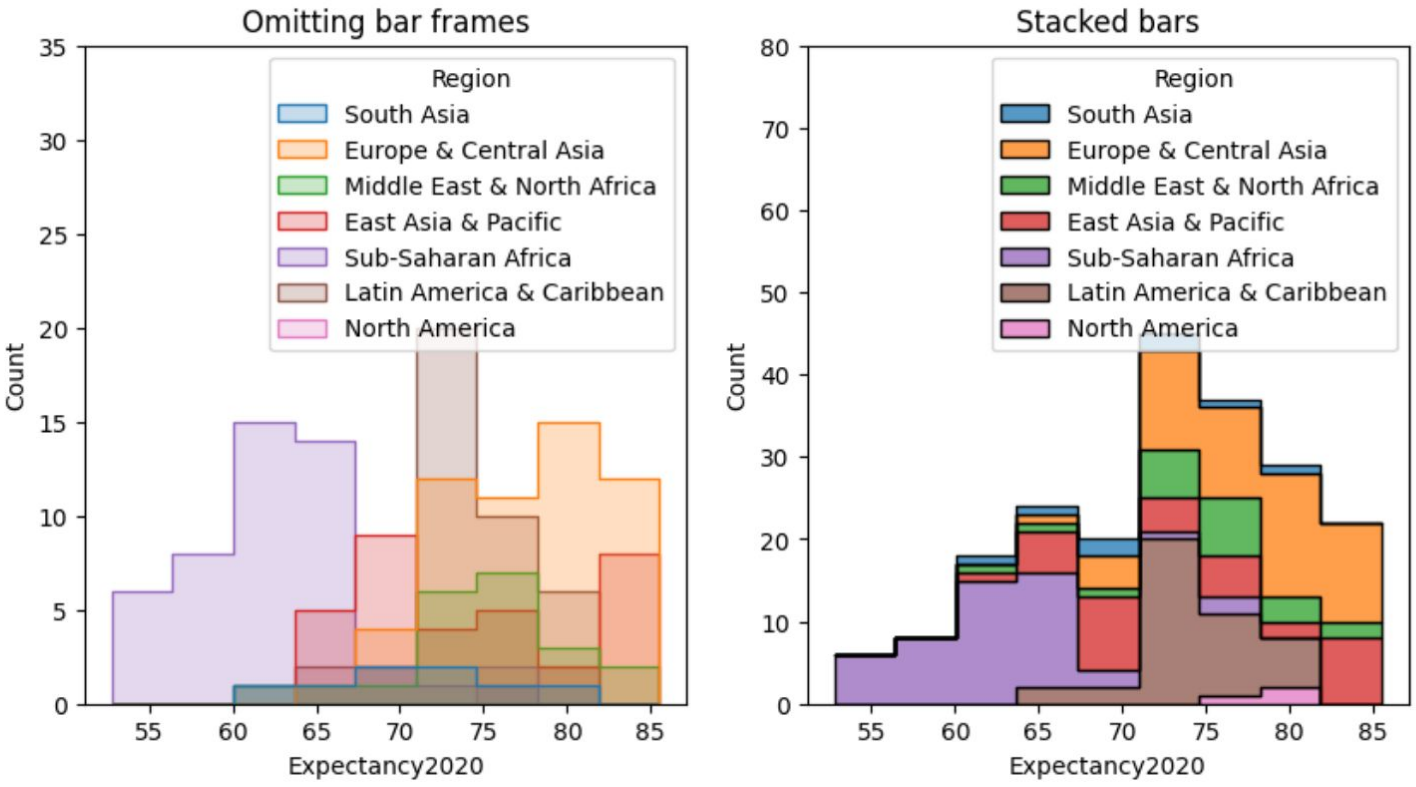

```
sns.histplot(data=countries,
             x='Expectancy2020',
             hue='Region',
             ax=axes[1])
```

Are these plots easy to read?





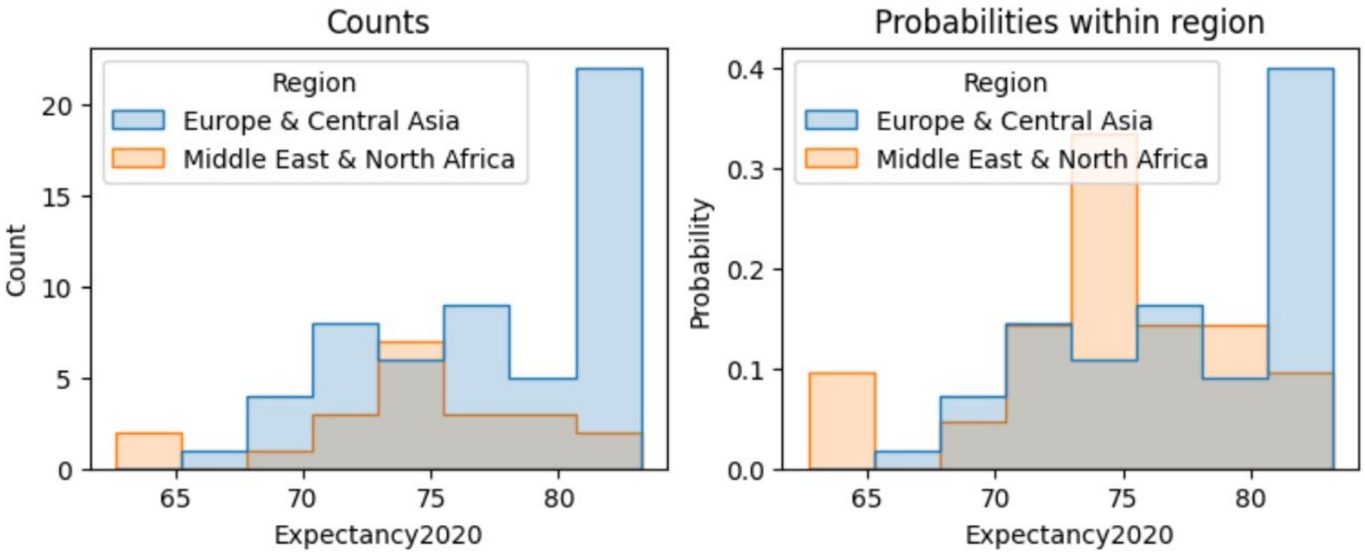
Possible improvements of the second plot



Normalization of groups

To better compare distribution of the expectancy within region,
use counts normalized to probabilities.

```
sns.histplot(data=countries_subset2,
             x='Expectancy2020',
             hue='Region',
             element='step',
             stat="probability",
             common_norm=False,
             ax=axes[1])
```

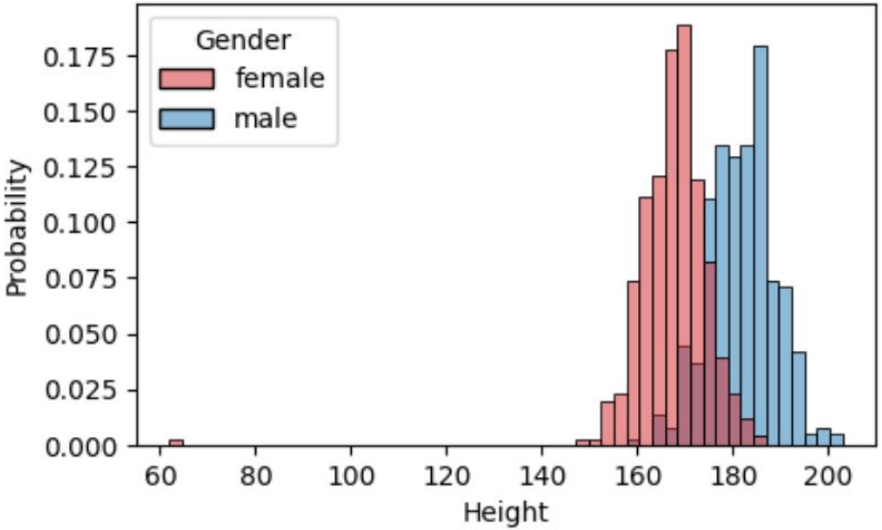


Middle East & North Africa	21
Europe & Central Asia	58

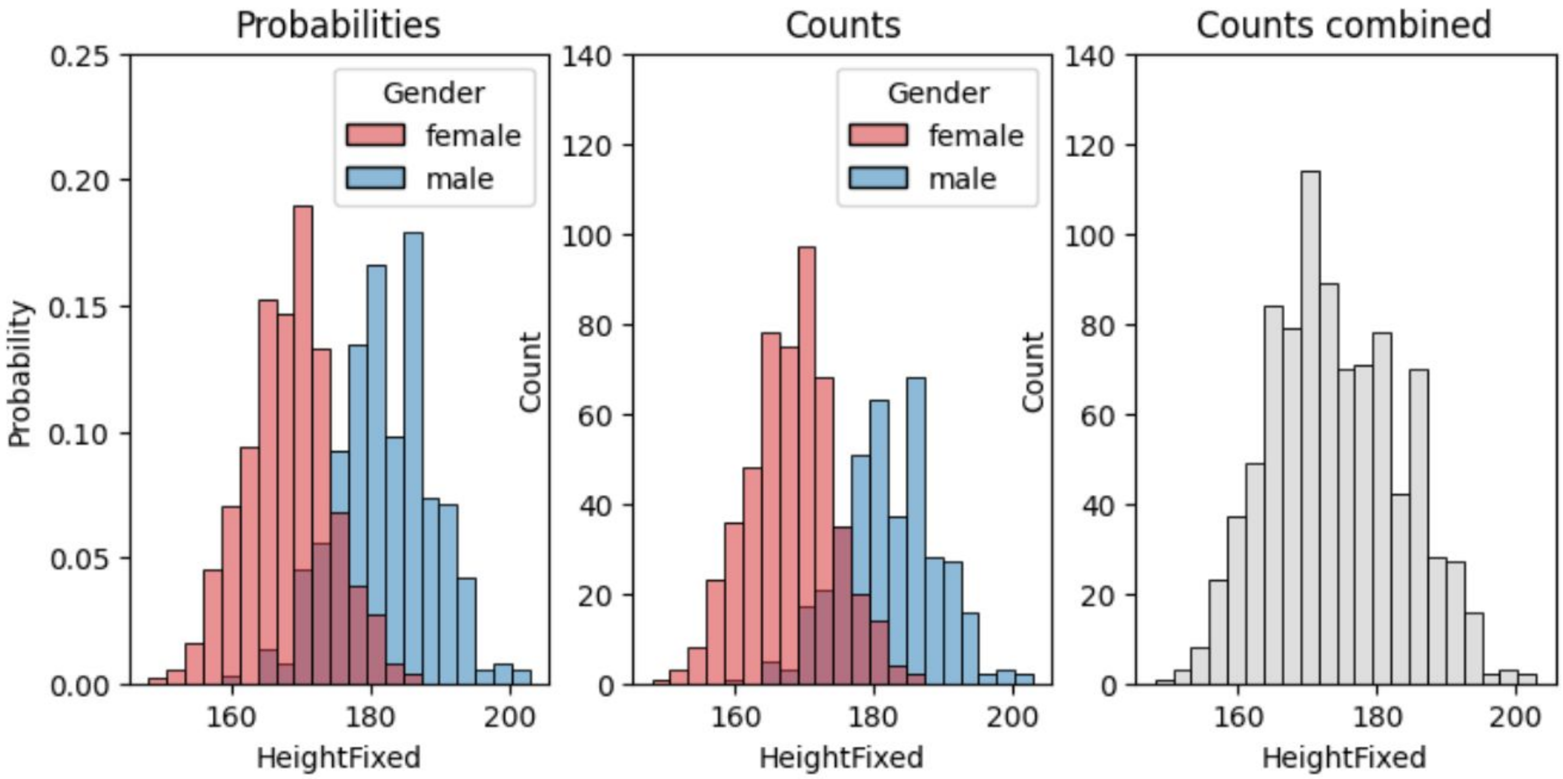
Final example: heights of men and women

FSEV survey, self-reported values, adults only

Outlier clearly visible, probably an error



After error removal



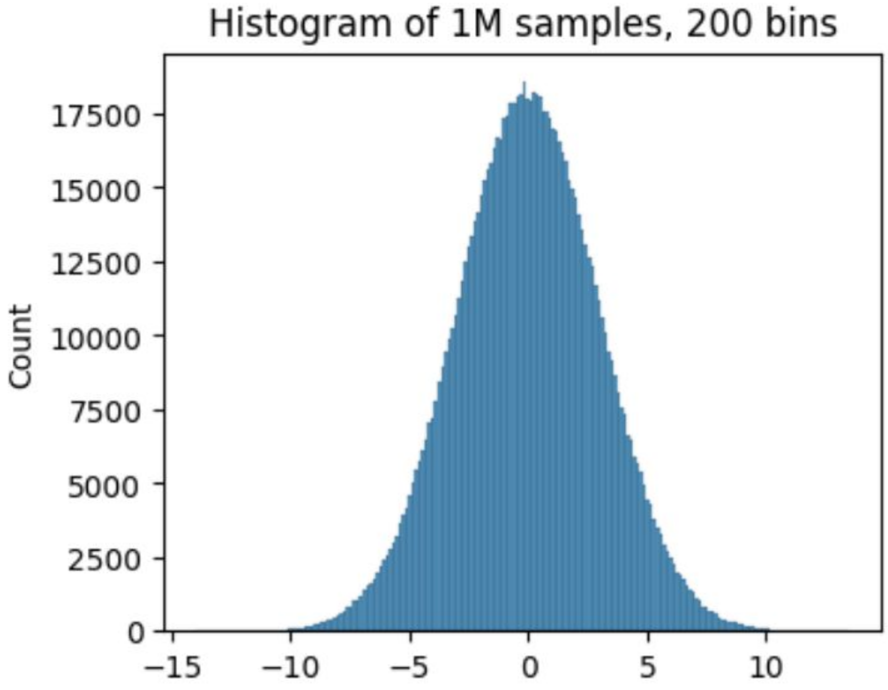
Probability distributions

Probability distributions

Imagine histogram of a great number of real values with tiny bins, keeping the area under the histogram equal to one.

In limit we obtain **probability density function (PDF) (hustota rozdelenia pravdepodobnosti).**

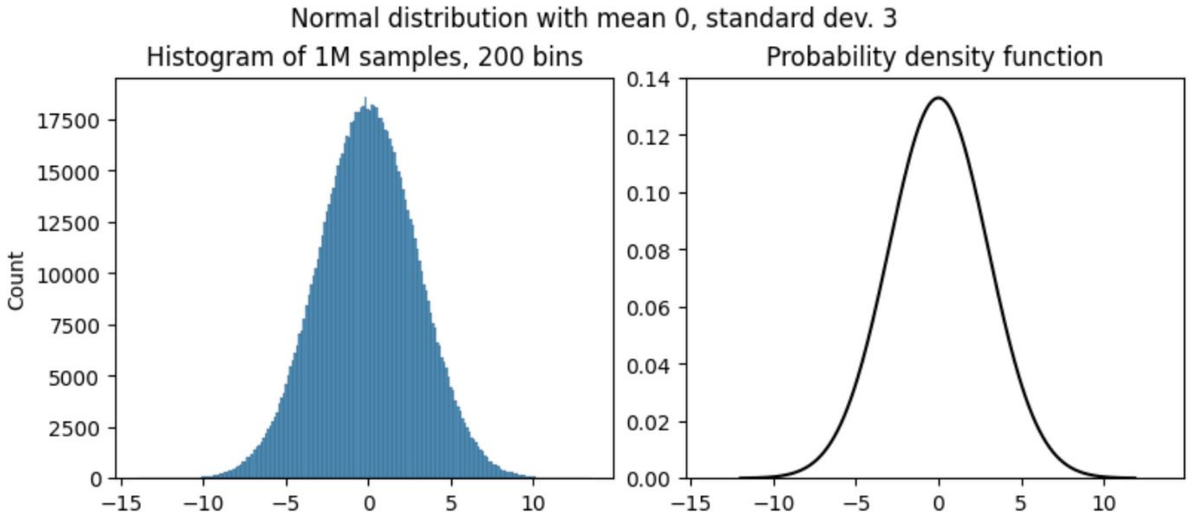
We often assume that our data are from a known probability distribution (rozdelenie pravdepodobnosti).



Normal (Gaussian) distribution

- It has two parameters: mean μ and standard deviation σ
- Density:

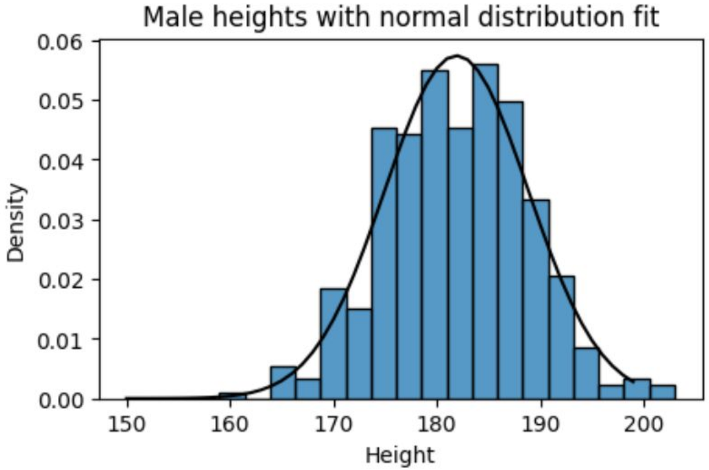
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$



```python
figure, axes = plt.subplots(1, 2, sharex=True,
                            figsize=(8, 3.5), layout="constrained")

# sample million points from the normal distrib. with mean 0 and std. dev. 3
sample_normal = np.random.normal(0, 3, 1000000)
# create histogram of the sampled points
sns.histplot(x=sample_normal, bins=200, ax=axes[0])
axes[0].set_title('Histogram of 1M samples, 200 bins')

# create an object representing normal distrib. with mean 0 and std. dev. 3
normal = scipy.stats.norm(0, 3)
# create equally-spaced points
x = np.arange(-12, 12, 0.1)
# compute values of pdf in these points
y = normal.pdf(x)
# plot the function
axes[1].plot(x, y, 'k-')
axes[1].set_title('Probability density function')
axes[1].set_ylim(0, 0.14)
```

Example with real data

- Normal distribution often arises in situations where a variable is a result of many small influences.
- One example is the height of a person within one gender and population.
- We **fit** the normal distribution to the histogram of the adult male heights.



Mean male height: 181.92

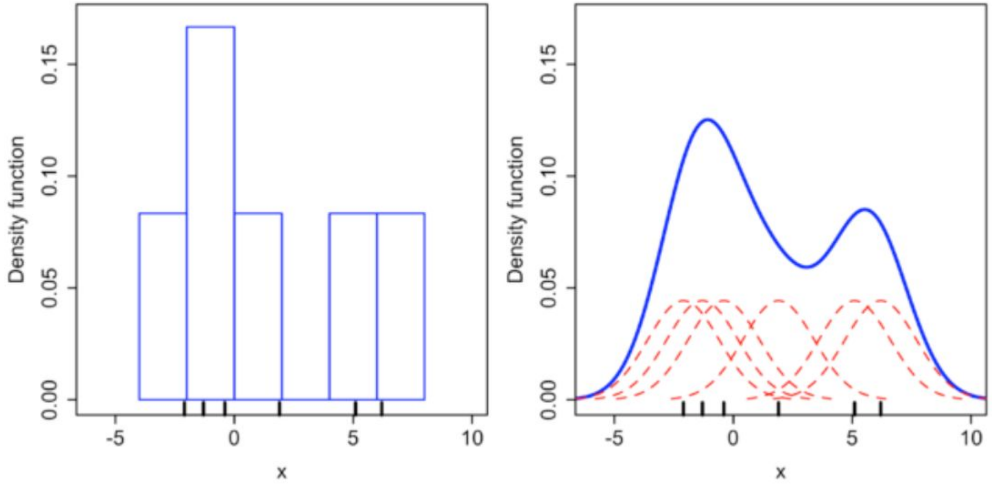
Std. dev. male height: 6.96

```python
# select male height, drop missing values
male_heights = adults.query("Gender=='male'")['Height'].dropna()
# compute the characteristics (means, stdev)
display(Markdown(f"**Mean male height:** {male_heights.mean():.2f}"),
        Markdown(f"**Std. dev. male height:** {male_heights.std():.2f}"))

# compute the best fit
parameters = scipy.stats.norm.fit(male_heights)
display(Markdown("**Best fit:**"), parameters)

# get function values for regularly distributed x values
x = np.arange(150, 200, 1)
pdf_fitted = scipy.stats.norm.pdf(x, loc=parameters[0], scale=parameters[1])

# plot histogram, normalized as density (area=1)
figure, axes = plt.subplots(figsize=(5,3))
sns.histplot(x=male_heights, stat='density', ax=axes)
# add a line for fitted density
axes.plot(x, pdf_fitted, 'k-')
axes.set_title('Male heights with normal distribution fit')
```

Kernel density estimation and violin plots

Kernel Density Estimation (KDE)

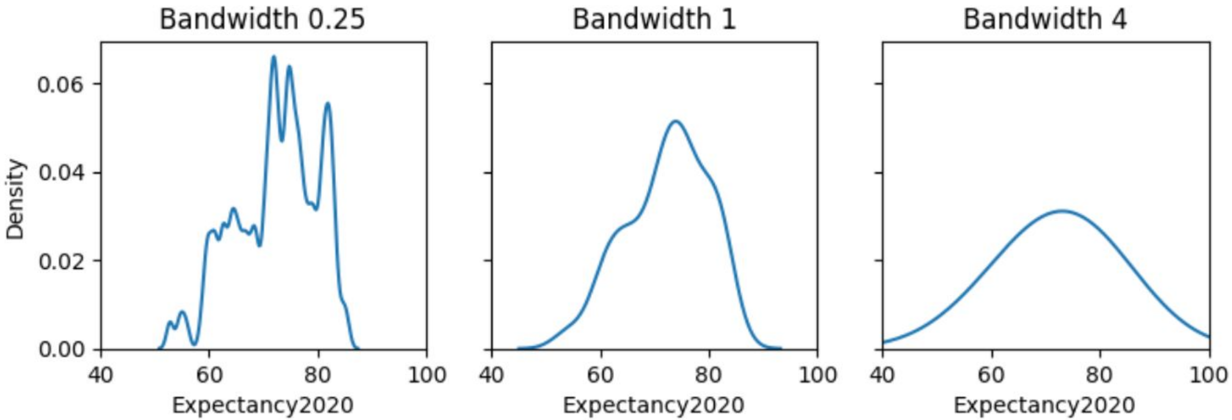
- A **smoothed** version of a **histogram**
- We choose a **kernel function**, e.g. the normal distribution
- For each point in the dataset, we create a "kernel" centered at that point
- We add up the heights of all kernels



https://commons.wikimedia.org/wiki/File:Comparison_of_1D_histogram_and_KDE.png Drleft at English Wikipedia, CC BY-SA 3.0

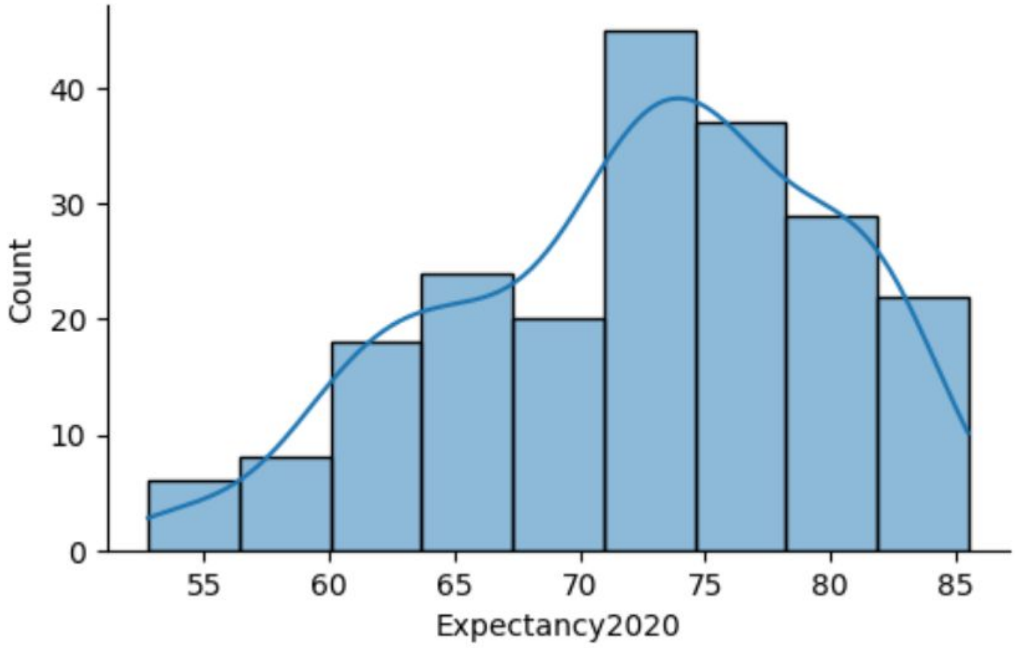
KDE in Seaborn

- KDE computed directly in Seaborn's displot/kdeplot functions
- The amount of smoothing is controlled by the **bandwidth** `bw_adjust` (standard deviation for the normal distribution)
- A small bandwidth: a bumpy plot not representing real trends
- A large bandwidth: can obscure real trends



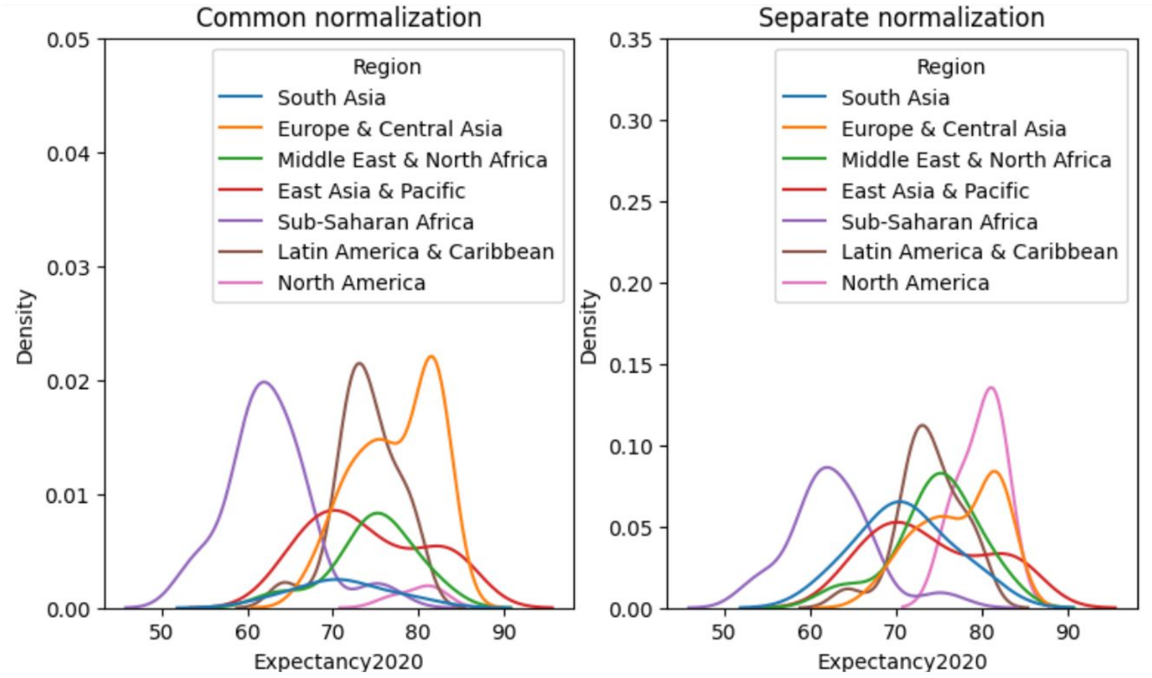
Combined histogram and KDE

```
axes = sns.displot(countries,
                   x="Expectancy2020",
                   kde=True)
axes.figure.set_size_inches(5, 3)
```



KDEs for comparing distributions

Smooth curves are easier to follow than histograms



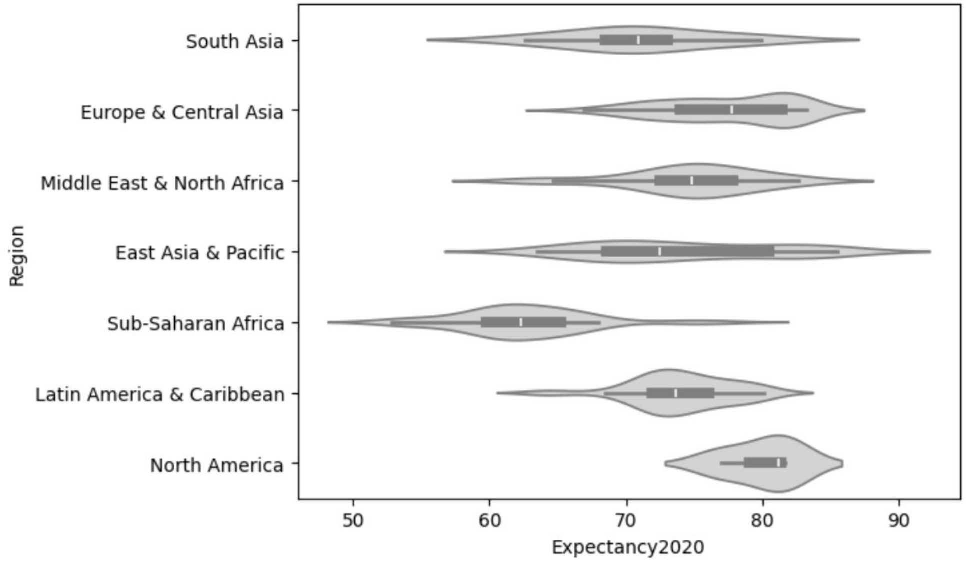
Violin plots

Compare distributions for different values of a categorical variable

Each violin: two symmetric KDE plots

Often combined with boxplot / strip plot

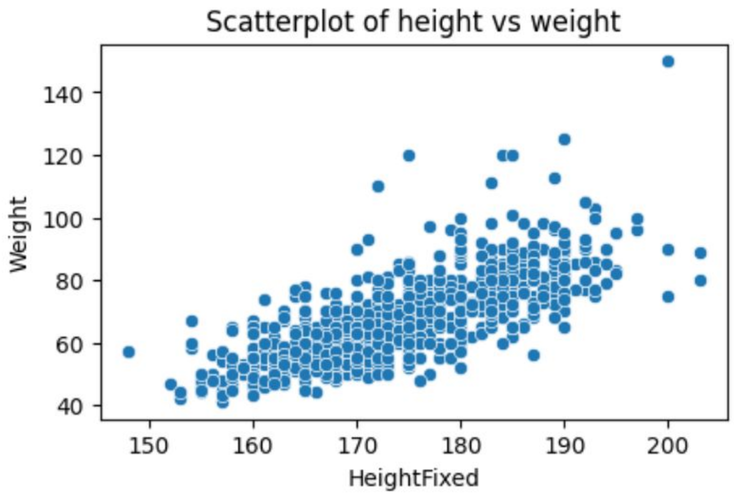
```
sns.violinplot(data=countries,
               y="Region",
               x="Expectancy2020",
               color="lightgrey")
```

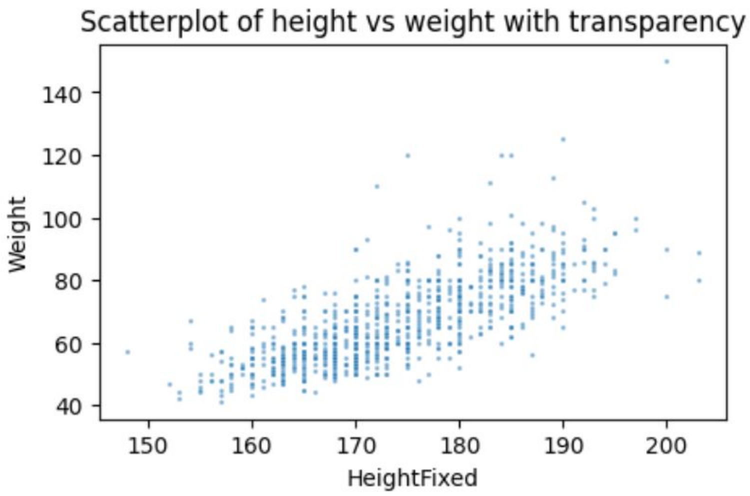


Two-dimensional histograms / KDE

Two-dimensional data can be drawn as a scatterplot.

Problems with **overplotting** if we have a lot of similar points.

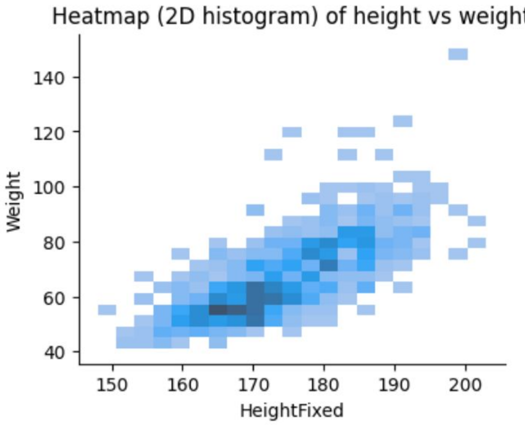


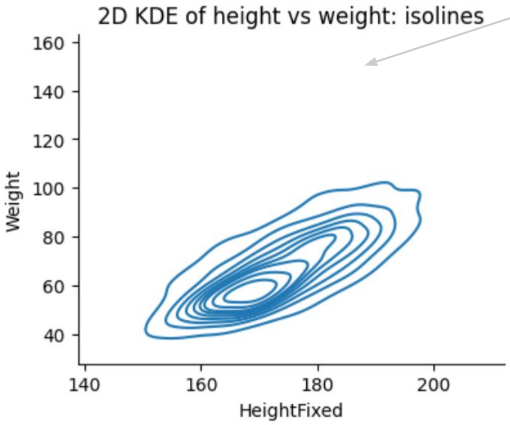


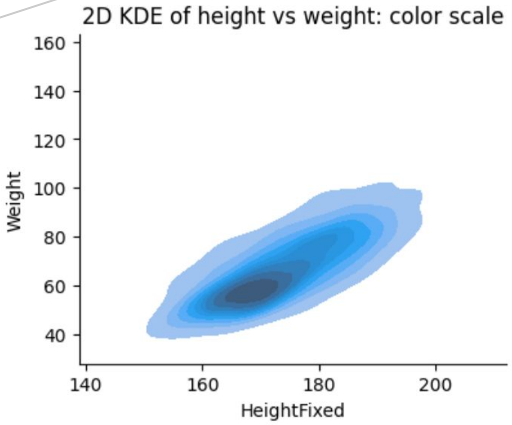
Two-dimensional histograms / KDE

Instead of scatterplots: 2D histograms shown as a heatmap or smoothed by KDE

```
sns.displot(data=adults,
            x='HeightFixed',
            y='Weight',
            kind="kde")
```







Cumulative distribution function

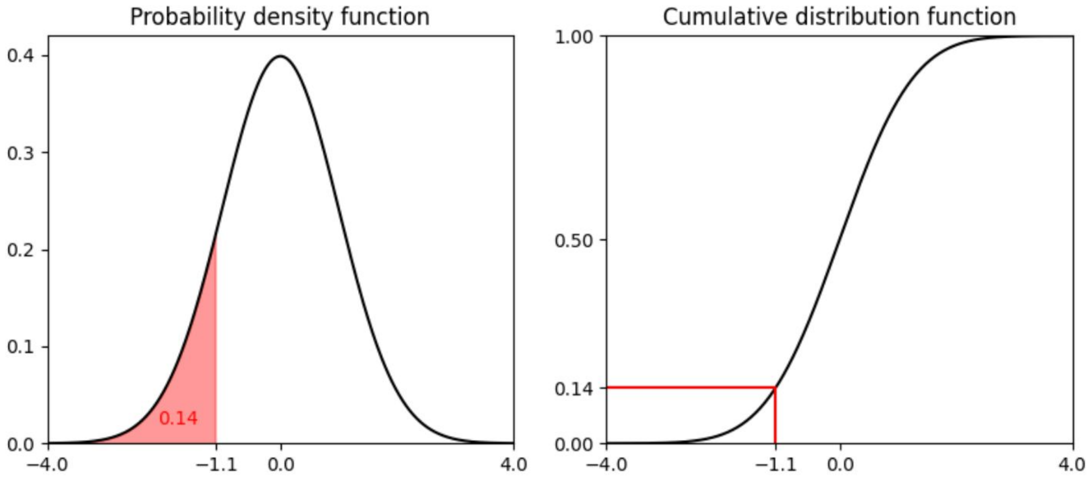
Cumulative distribution function (CDF)

Consider probability density function *f(x)*

Its CDF (distribučná funkcia) is the area under the curve from left up to point *x*

$$F(x) = \int_{-\infty}^{x} f(t)\,dt.$$

F(x) is the probability that the random point from the distribution is ≤ *x*

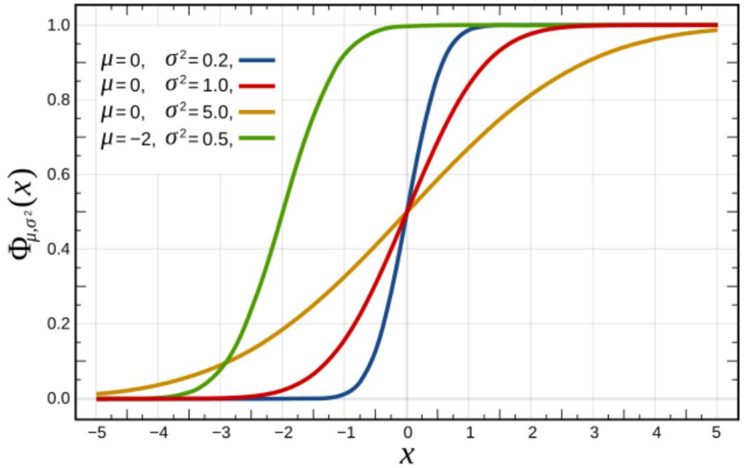


Cumulative distribution function (CDF)

CDF *F(x)* is the probability that the random point from the distribution is ≤ *x*

CDF is non-decreasing

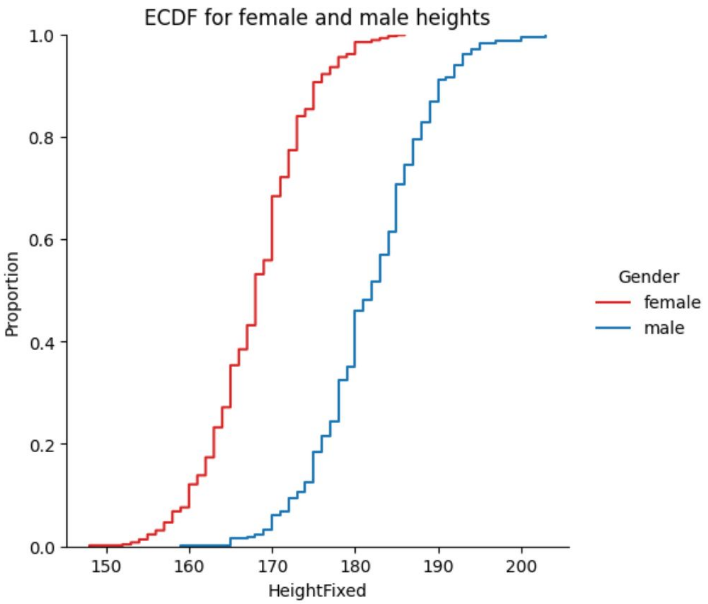
$\lim_{x \to -\infty} F(x)=0$ and $\lim_{x \to \infty} F(x)=1$



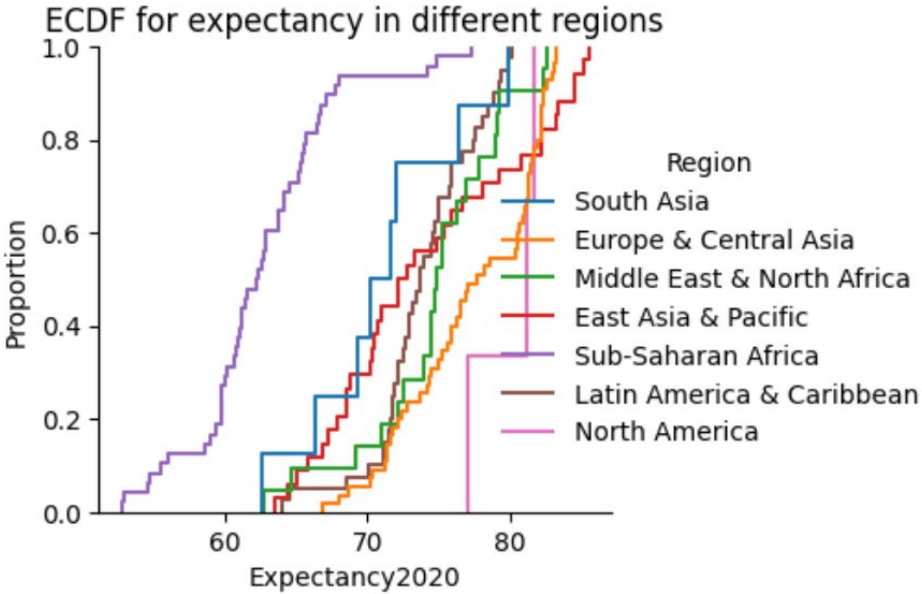
https://commons.wikimedia.org/wiki/File:Normal_Distribution_CDF.svg

Empirical cumulative distribution function (ECDF)

- A similar concept for a finite sample
- For each *x*, *F(x)* is the fraction of the sample which is ≤ *x*
- A stepwise function, can be visualized
- Unlike histograms and KDE, no parameters need to be set
- Allows comparison of quantiles (how?)
- But harder to interpret than histogram in terms of shape



Empirical cumulative distribution function (ECDF)




```
grid = sns.displot(countries, x="Expectancy2020", hue="Region", kind="ecdf")
grid.axes[0,0].set_title('ECDF for expectancy in different regions')
grid.figure.set_size_inches(5, 3)
```

Multi-dimensional data:
clustering and dimensionality reduction

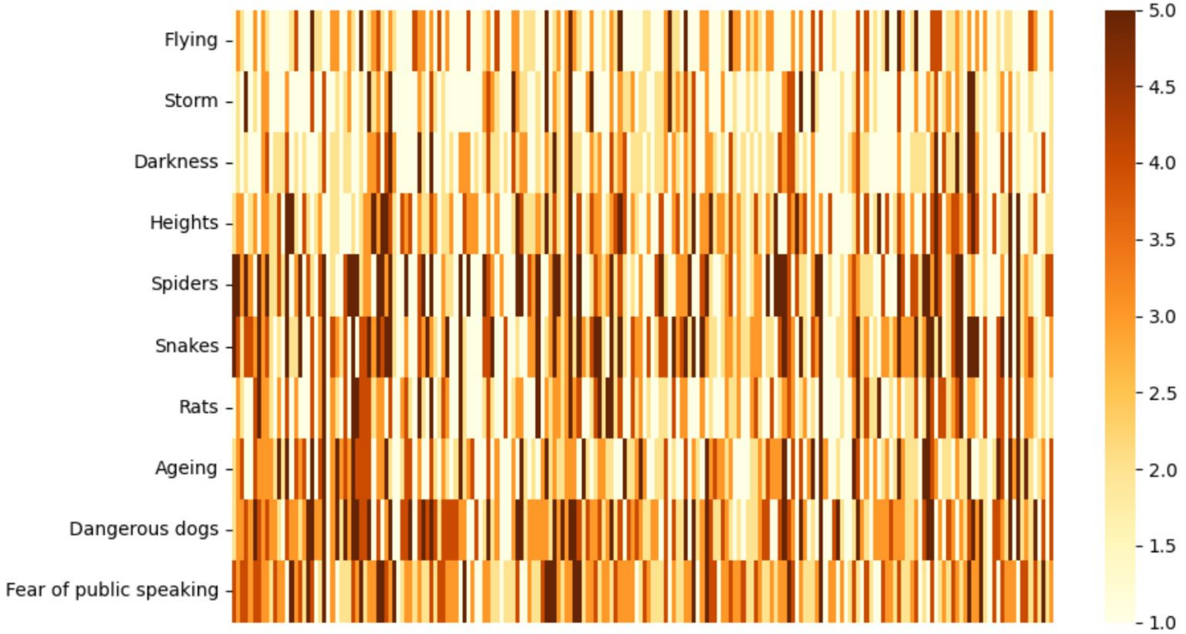
Dataset

The FSEV survey contains questions about phobias and fears, each with answers 1-5 (5 means highest fear)

Flying	Storm	Darkness	Heights	Spiders	Snakes	Rats	Ageing	Dangerous dogs	Fear of public speaking
1.0	1.0	1.0	2.0	5.0	5	1.0	2.0	2.0	4.0
3.0	2.0	2.0	3.0	5.0	4	3.0	3.0	3.0	3.0
2.0	1.0	1.0	3.0	3.0	2	2.0	4.0	3.0	4.0
1.0	5.0	2.0	1.0	5.0	4	1.0	1.0	4.0	4.0
1.0	1.0	1.0	2.0	3.0	4	1.0	1.0	3.0	3.0

Heatmap

200 randomly selected participants without missing values

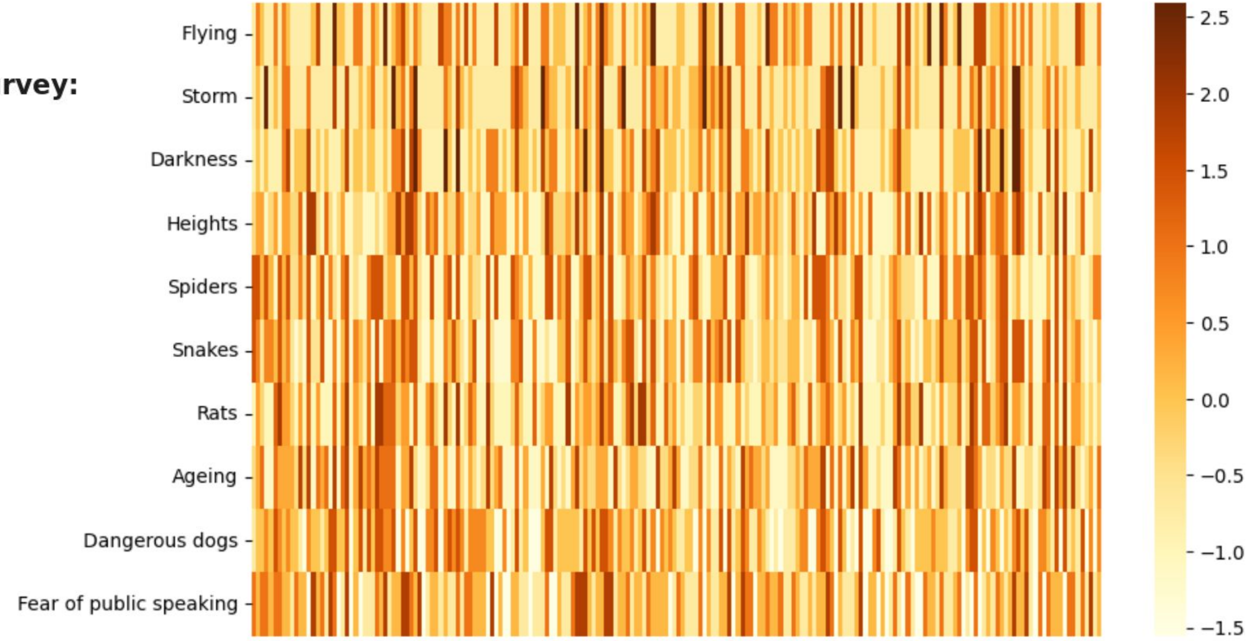


Means, standardization

Subtract the mean, divide by the standard deviation for each phobia

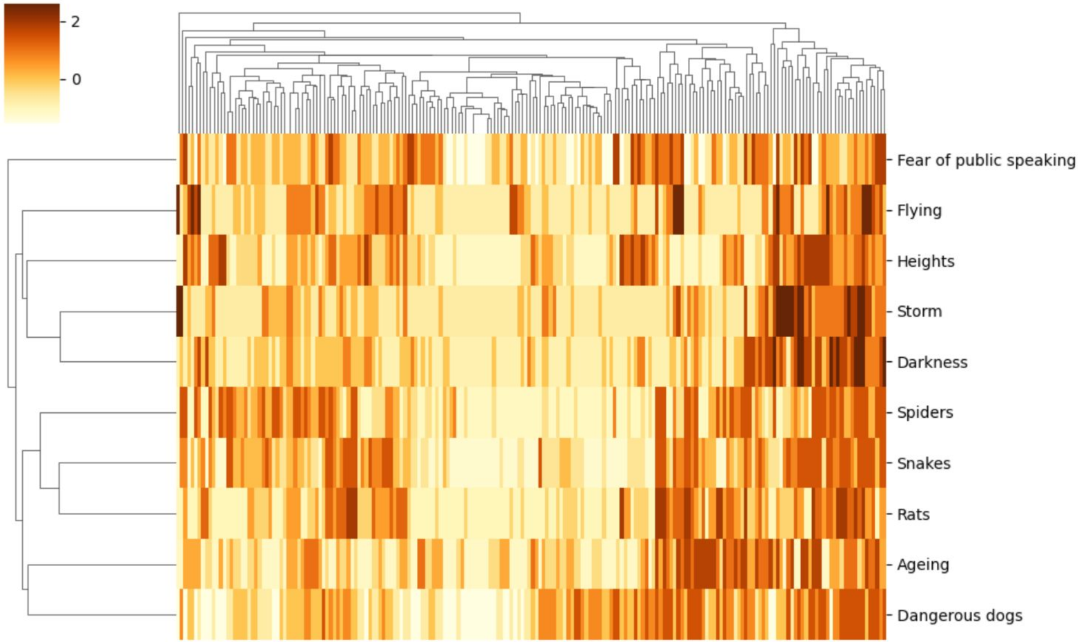
Phobias sorted by mean score in the survey:

Storm	1.885
Flying	1.980
Darkness	2.025
Rats	2.360
Heights	2.480
Ageing	2.555
Spiders	2.715
Fear of public speaking	2.840
Snakes	2.855
Dangerous dogs	3.015



Clustering (zhlukovanie)

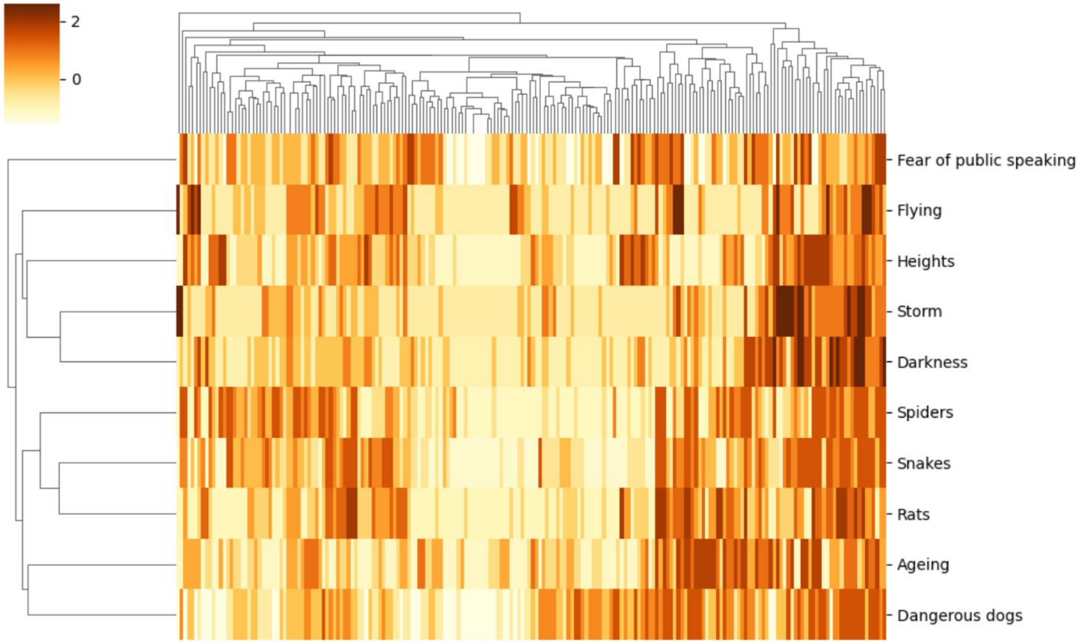
- Find similar groups of data
- Here **hierarchical clustering** (hierarchy of smaller and bigger groups)
- Applied to both people and phobias




```
sns.clustermap(fsev_sample_standardized.transpose(),
               xticklabels=False, figsize=(10,6), cmap="YlOrBr")
```

Clustering (zhlukovanie)

- Rows and columns of matrix were reordered according to clustering
- Some areas of dark and light colors now appear



Dimensionality reduction

Project high-dimensional data into lower dimensions, while trying to preserve some structure from the original data

- Principal component analysis (PCA) uses a linear projection: each new dimension is a linear combination (weighted sum) of the original dimensions. Weights are chosen to maximize variance.

Some methods do not use linear projections, but try to preserve distances between points, for example:

- Multidimensional scaling (MDS),
- T-distributed Stochastic Neighbor Embedding (t-SNE).

Principal component analysis (PCA)

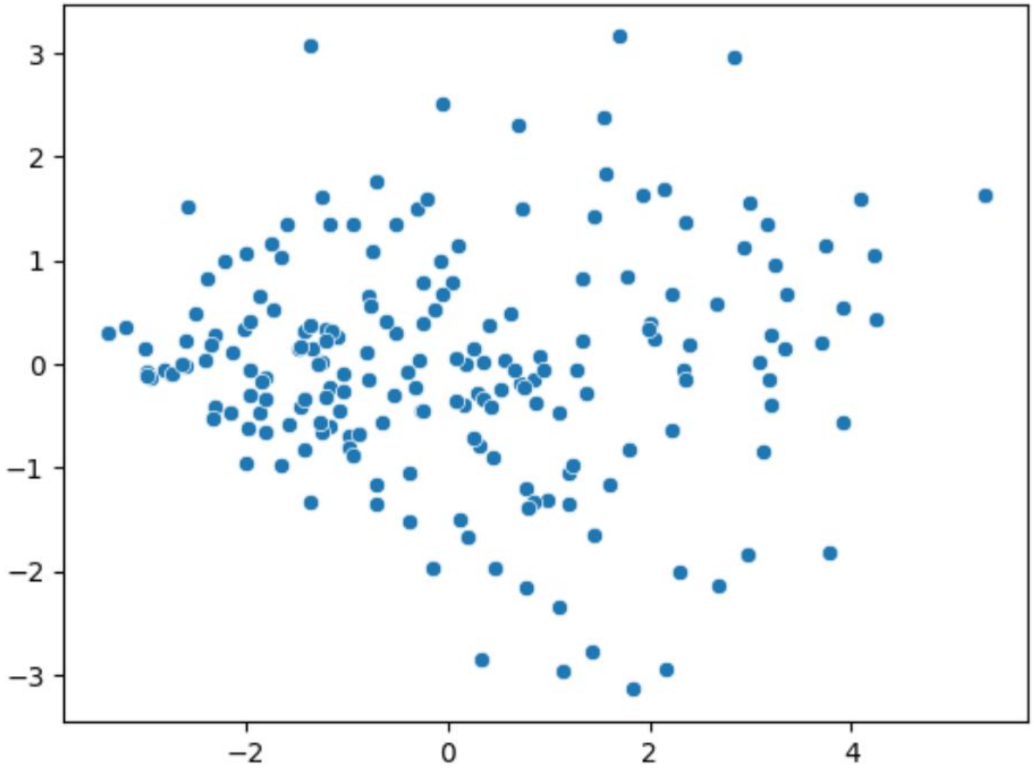
We use scikit-learn library for machine learning in Python

```
from sklearn.decomposition import PCA
# compute PCA of our standardized data with 2 dimensions
fsev_pca = PCA(n_components=2).fit_transform(fsev_sample_standardized)
display(Markdown("**PCA transformed values** (first five lines):"))
display(fsev_pca[0:5, :])
```

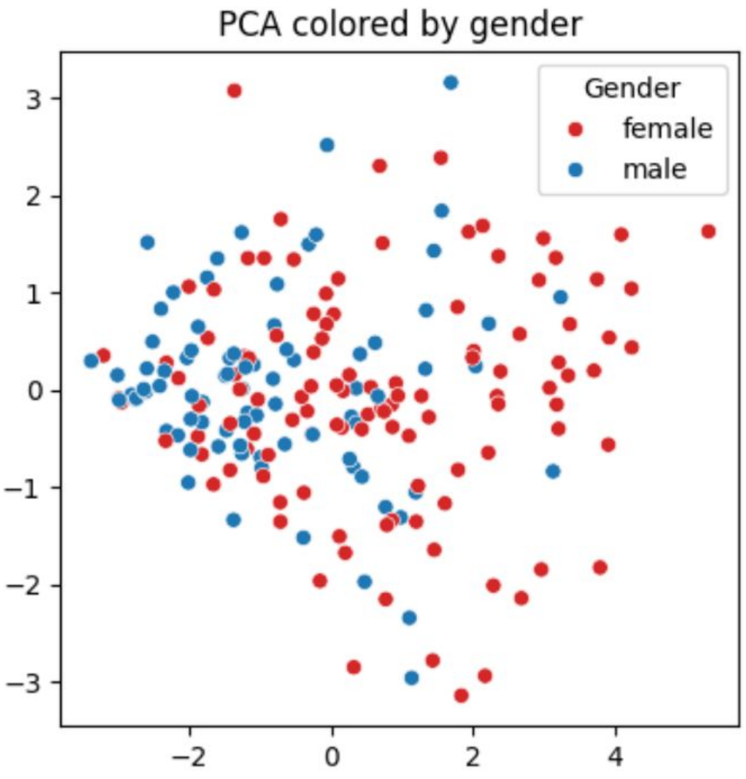
PCA transformed values (first five lines):

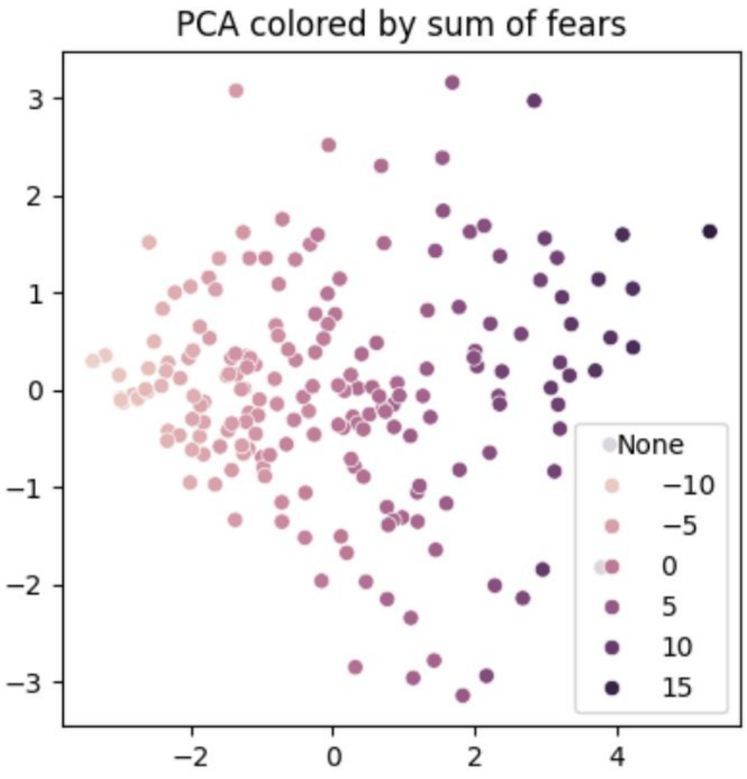
```
array([[-0.37918129, -1.05787296],
       [ 1.37937901, -0.2818021 ],
       [-0.2685224 , -0.45707943],
       [ 0.9147904 ,  0.06866801],
       [-1.18098851, -0.60787696]])
```

Scatterplot of PCA dimensions

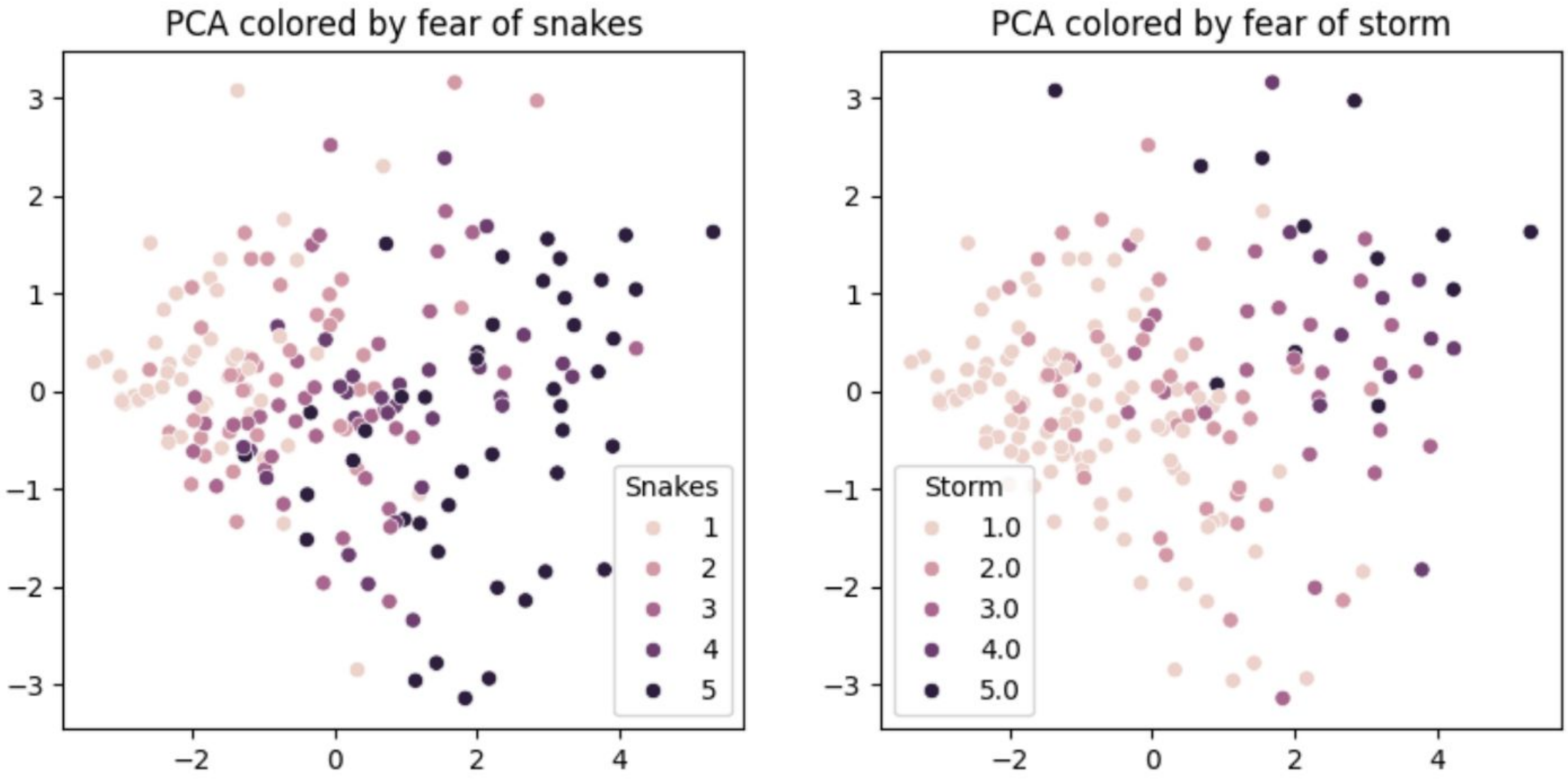


Displaying various variables as color





Displaying various variables as color



Conclusion and other courses

We briefly covered several statistical concepts often used in visualization:

- histogram
- kernel density estimation
- empirical cumulative distribution function
- clustering
- dimensionality reduction

You will learn more in the next years of your study:

- Fundamentals of Probability and Statistics, 2W (DAV) or 3W (BIN)
- Principles of Data Science 3W (DAV)
- Linear Algebra this semester