# Lecture 4
# Summary statistics

[Data visualization · 1-DAV-105]

Lecture by Broňa Brejová

More details in the [notebook version]

# Introduction

**Summary statistics (popisné charakteristiky / štatistiky):**
quantities that summarize basic properties of a single variable (a table column), such as the mean.

We can also characterize dependencies between pairs of variables.

Together with simple plots, they give us the first glimpse at the data
when working with a new data set.

# Data set for today

- The same data set as in group tasks 04.
- The data set describes 2049 movies.
- Originally downloaded from https://www.kaggle.com/rounakbanik/the-movies-dataset and preprocessed, keeping only movies with at least 500 viewer votes.

```
url = 'https://bbrejova.github.io/viz/data/Movies_small.csv'
movies = pd.read_csv(url)
display(movies.head())
```

| title | year | budget | revenue | original_language | runtime | release_date | vote_average | vote_count | overview |
|---|---|---|---|---|---|---|---|---|---|
| Toy Story | 1995 | 30000000.0 | 373554033.0 | en | 81.0 | 1995-10-30 | 7.7 | 5415.0 | Led by Woody, Andy's toys live happily in his ... |
| Jumanji | 1995 | 65000000.0 | 262797249.0 | en | 104.0 | 1995-12-15 | 6.9 | 2413.0 | When siblings Judy and Peter discover an encha... |
| Heat | 1995 | 60000000.0 | 187436818.0 | en | 170.0 | 1995-12-15 | 7.7 | 1886.0 | Obsessive master thief, Neil McCauley leads a ... |
| GoldenEye | 1995 | 58000000.0 | 352194034.0 | en | 130.0 | 1995-11-16 | 6.6 | 1194.0 | James Bond must unmask the mysterious head of ... |
| Casino | 1995 | 52000000.0 | 116112375.0 | en | 178.0 | 1995-11-22 | 7.8 | 1343.0 | The life of the gambling paradise – Las Vegas ... |

# Measures of central tendency (miery stredu / polohy)

These represent a **typical value** in a sample $x = x_1, \ldots, x_n$ (one numerical column).

**Mean (priemer)** $$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

This is the arithmetic mean, there are also geometric and harmonic means.

**Median (medián)** is the middle value when the values ordered by size.

For even $n$ usually defined as the average of the two middle values.

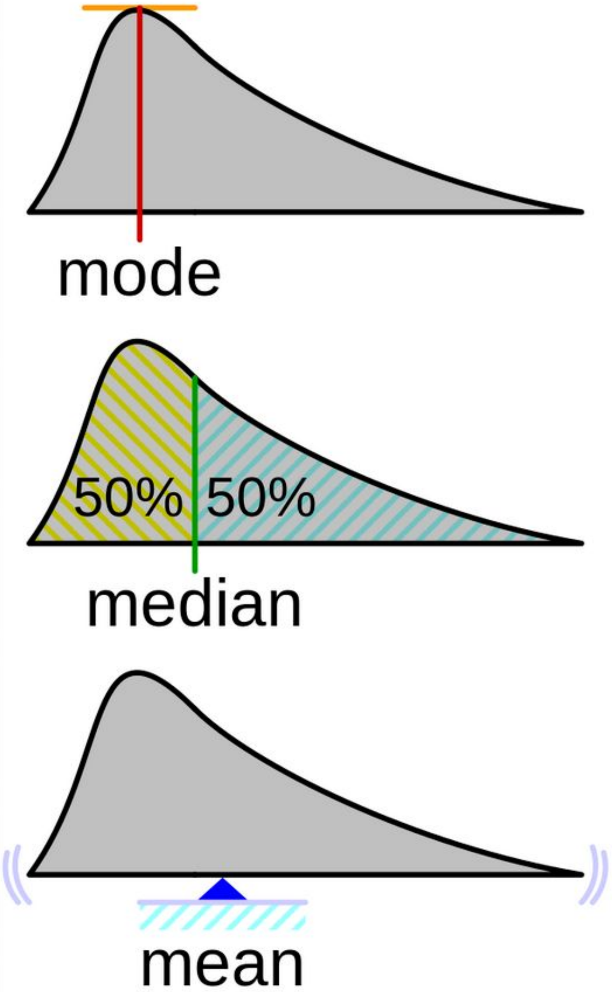
Example:

Median of 10, 12, 15, 16, 16 is 15.

Median of 10, 12, 15, 16, 16, 20 is 15.5.

# Measures of central tendency (cont.)

- **Mean (priemer)**
- **Median (medián)**
- **Mode (modus)** is the most frequent value (for a discrete variable).
  Mode of 10,12,15,16,16 is 16.
  For continuous variables, we may look for a mode in a histogram.
  This is sensitive to bin size.




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

# Properties of the measures

If we apply **linear transformation** $a \cdot x_i + b$ with the same $a$ and $b$ to all $x_i$, mean, median and mode will be transformed **in the same way**.

This corresponds e.g. to the change in the units of measurement (grams vs kilograms, degrees C vs degrees F)

**Mean can be heavily influenced by outliers**
800, 1000, 1100, 1200, 1800, 2000, 30000: mean 5414.3, median 1200
800, 1000, 1100, 1200, 1800, 2000, 10000: mean 2557.1, median 1200

Therefore we often **prefer median** (e.g. median salary).

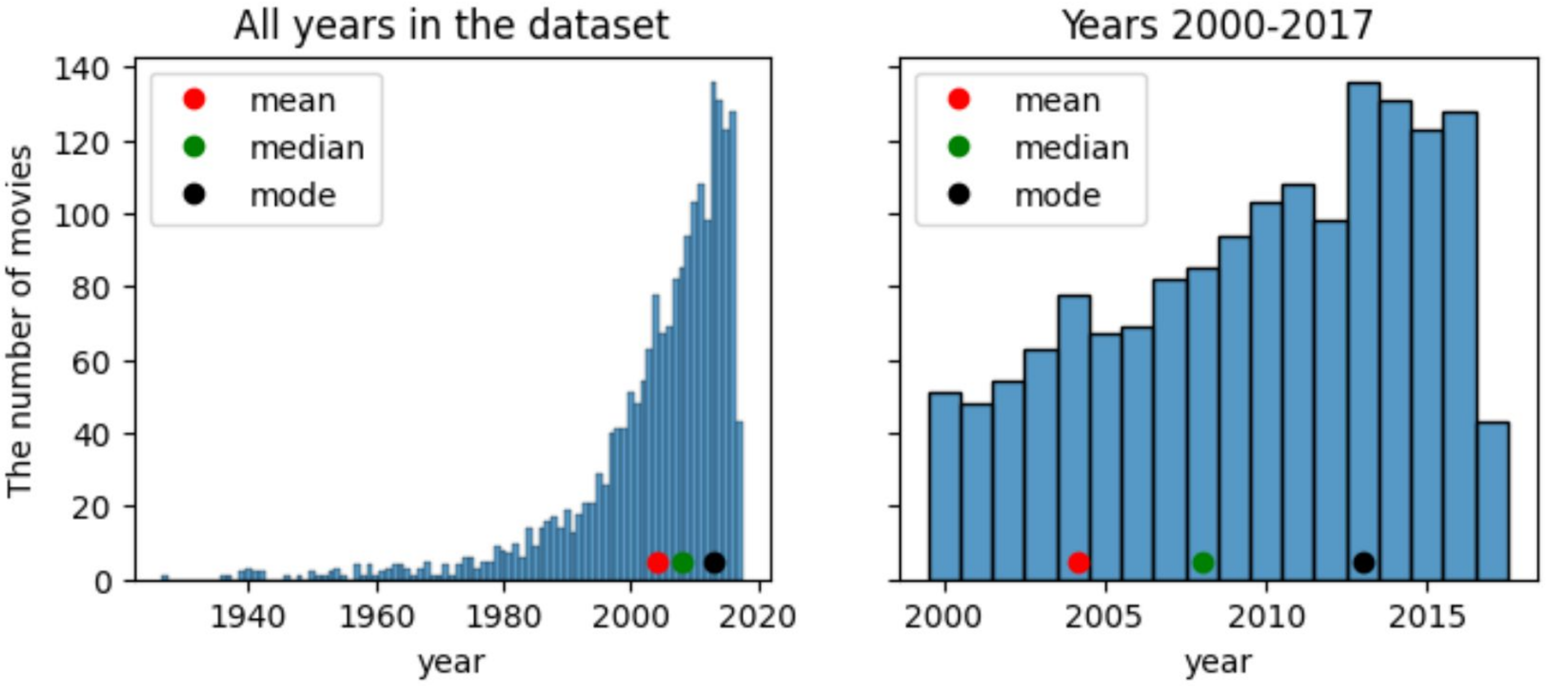
# Computation in Pandas

```python
display(Markdown("**Properties of the column `year` in our table:**"))
print(f"Mean: {movies['year'].mean():.2f}")
print(f"Median: {movies['year'].median()}")
print(f"Mode:\n{movies['year'].mode()}")
```

**Properties of the column year in our table:**

```
Mean: 2004.14
Median: 2008.0
Mode:
0    2013
Name: year, dtype: int64
```

# Shown in a histogram (whole and detail)




What causes the difference between the mean and the median?

# Summarizing many columns or rows

- Functions `mean` and `median` can be applied to all numerical columns
- With `axis=1` we get means or medians in rows

```
display(movies.mean(numeric_only=True))
```

```
year                    2,004.1
budget             55,108,939.7
revenue           198,565,134.3
runtime                   112.7
vote_average                6.6
vote_count              1,704.6
dtype: float64
```

# Quantiles, percentiles and quartiles (kvantily, percentily, kvartily)

**Median** is the middle value in a sorted order;
about 50% of values are smaller and 50% larger.

For a percentage $p$, the $\boldsymbol{p}$**-th percentile** is at position roughly $(p/100) \cdot n$ in the sorted order of values.

Similarly **quantile** (in Pandas), but we give fraction between 0 and 1 rather than percentage.

**Quartiles** are three values $Q_1$, $Q_2$ and $Q_3$ that split input data into quarters.
Therefore, $Q_2$ is the median.

# Quantiles in Pandas

```
display(Markdown("**Median:**"),
        movies['year'].median())
display(Markdown("**Quantile for 0.5:**"),
        movies['year'].quantile(0.5))
display(Markdown("**All quartiles:**"),
        movies['year'].quantile([0.25, 0.5, 0.75]))
display(Markdown("**With step 0.1:**"),
        movies['year'].quantile(np.arange(0.1, 1, 0.1)))
```

**Median:**

```
2008.0
```

**Quantile for 0.5:**

```
2008.0
```

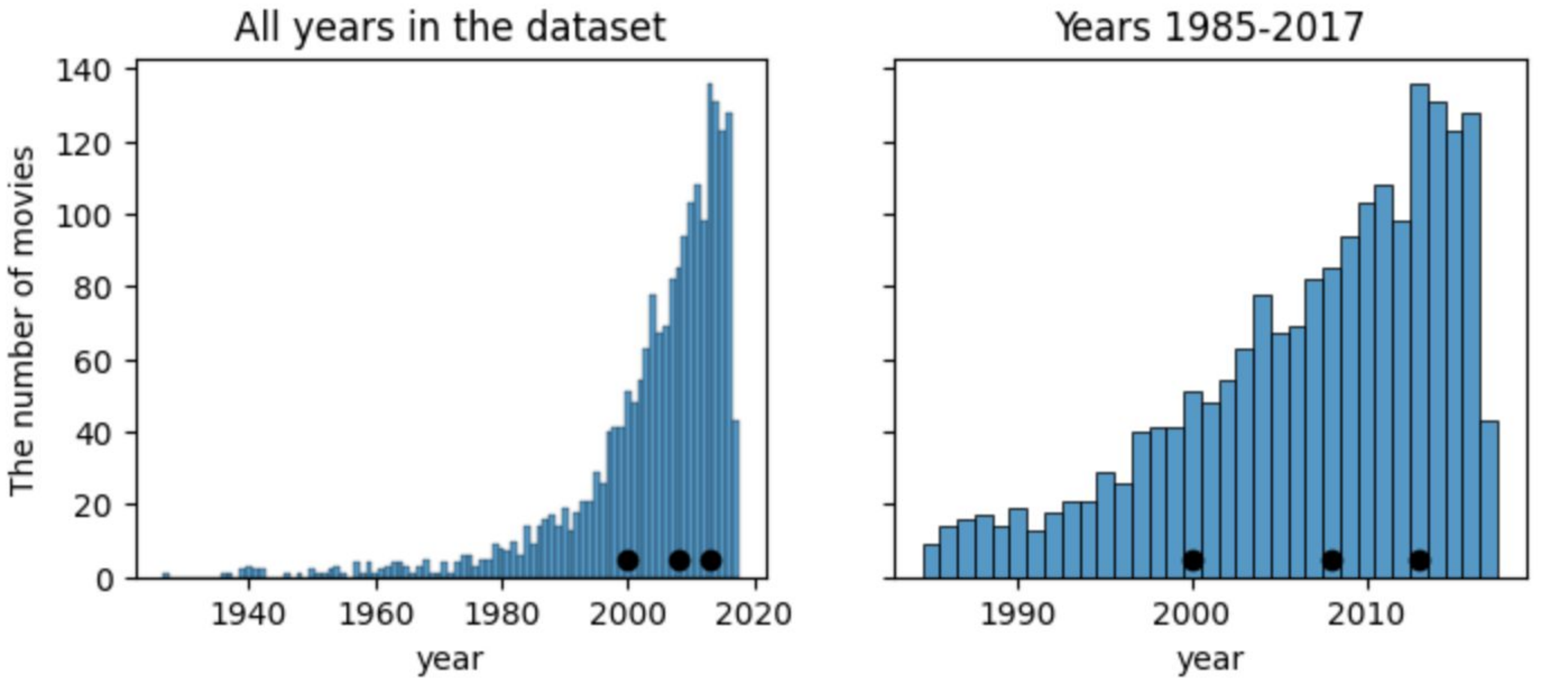
**All quartiles:**

```
0.25    2,000.00
0.50    2,008.00
0.75    2,013.00
Name: year, dtype: float64
```

**With step 0.1:**

```
0.10    1,988.80
0.20    1,998.00
0.30    2,002.00
0.40    2,005.00
0.50    2,008.00
0.60    2,010.00
0.70    2,012.00
0.80    2,014.00
0.90    2,015.00
Name: year, dtype: float64
```

# Quartiles in a histogram (whole and detail)



# Quantile interpolation

Optional parameter `interpolation` accepts values `'linear'` (default), `'lower'`, `'higher'`, `'midpoint'`, `'nearest'`.

Minimum is at quantile 0, maximum at quantile 1, the rest evenly spaced between.

The quantile between two elements is influenced only by its two neighbors.

Example: list [0,10,20,100]
p=0: 0,  p=1/3: 10,  p=2/3: 20,  p=1: 100
p=1/4: by default linear interpolation at 3/4 between 0 and 10, i.e. 7.5.

Linear interpolation is continuous as p changes from 0 to 1.

# Quantile interpolation

**Quantiles for [0, 10, 20, 100]**

```
0.01     0.30
0.25     7.50
0.50    15.00
0.75    40.00
dtype: float64
```

**Quantiles for [0, 10, 20, 100] with `interpolation='lower'`**

```
0.01      0
0.25      0
0.50     10
0.75     20
dtype: int64
```

# Measures of variability (miery variability)

Values in the sample may be close to central values or spread widely.

Examples of measures:

**Range** of values from **minimum** to **maximum** (sensitive to outliers).

**Interquartile range IQR (kvartilové rozpätie)**: range between $Q_1$ and $Q_3$ (contains the middle half of the data).

**Variance and standard deviation** (next)

# Variance (rozptyl)

For each $x_i$ square its difference from the mean

Squaring gives non-negative values (and squares are easier to work with mathematically than absolute values).

Variance is the mean of these squares, but we divide by $n-1$ rather than $n$:

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n - 1}$$

Division by $n$ would underestimate the true variance of the underlying population (more in the statistics course).

# Standard deviation (smerodajná odchýlka)

Square root of the variance

$$s = \sqrt{s^2}$$

It is in the same units as the original values
(variance is in units squared).

Recall:

$$s^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n - 1}$$

# Properties

Larger variance and standard deviation mean that data are spread farther from the mean.

If we apply **linear transformation** $a \cdot x_i + b$ with the $a$ and $b$ to all $x_i$:
Neither variance nor standard deviation change with $b$.
Variance is multiplied by $a^2$, standard deviation by $|a|$.

These measures are strongly **influenced by outliers**:
800, 1000, 1100, 1200, 1800, 2000, 30000: st. dev. 10850.0, IQR 850
800, 1000, 1100, 1200, 1800, 2000, 10000: st. dev. 3310.5, IQR 850.

```python
display(Markdown("**Minimum**"),
        movies['year'].min())
display(Markdown("**Maximum**"),
        movies['year'].max())
display(Markdown("**Mean**"),
        movies['year'].mean())
display(Markdown("**Variance**"),
        movies['year'].var())
display(Markdown("**Standard deviation**"),
        movies['year'].std())
q1 = movies['year'].quantile(0.25)
q3 = movies['year'].quantile(0.75)
display(Markdown("**Q1, Q3 and IQR:**"),
        q1, q3, q3-q1)
```

**Minimum**

1927

**Maximum**

2017

**Mean**

2004.1449487554905

**Variance**

161.2714600681735

**Standard deviation**

12.69927006044731З

**Q1, Q3 and IQR:**

2000.0
2013.0
13.0

# Outliers (odľahlé hodnoty)

Outliers are the values which are far from the typical range of values.

In data analysis, it is important to **check outliers.**

If they represent **errors**, we may try to correct or remove them.

They can also represent **interesting anomalies.**

**Different definitions** of outliers may be appropriate in different situations.

# One possible definition of outliers

The criterion by statistician John Tukey:

Outliers are the values outside of the range $[Q_1 - k \cdot IQR, Q_3 + k \cdot IQR]$, e.g. for $k = 1.5$.

In our example 800, 1000, 1100, 1200, 1800, 2000, 30000:
$Q_1 = 1050$, $Q_3 = 1900$, $IQR = 850$.
$Q_1 - 1.5 \cdot IQR = -225$, $Q_3 + 1.5 \cdot IQR = 3175$.

Outliers are values smaller than −225 or larger than 3175; here only 30000.

The range of outliers is not influenced if we change the outliers.

# Computation in Pandas

```
# get quartiles and iqr
q1 = movies['year'].quantile(0.25)
q3 = movies['year'].quantile(0.75)
iqr = q3 - q1
# compute thresholds for outliers
lower = q1 - 1.5 * iqr
upper = q3 + 1.5 * iqr
# count outliers
count = movies.query('year < @lower or year > @upper')['year'].count()
```

**Outliers outside of range:** [1980.5, 2032.5]

**Outlier count:** 112

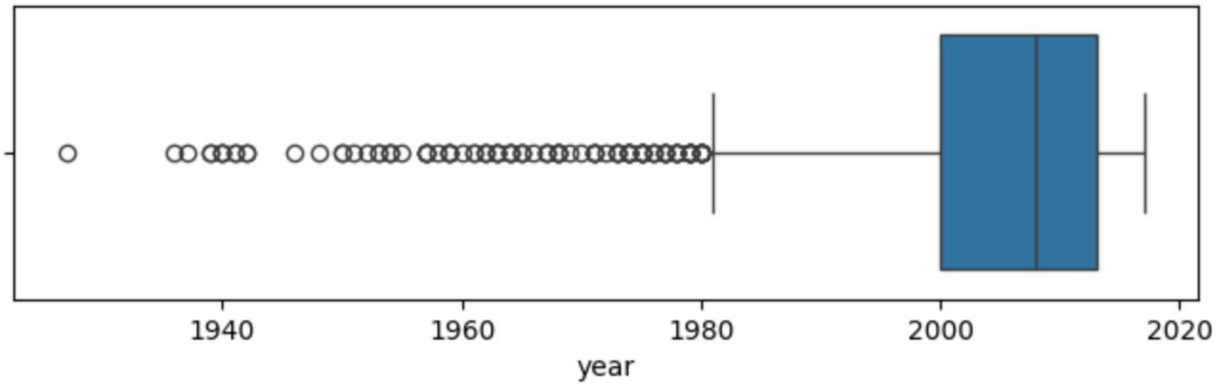
**Total count:** 2049

# Boxplot (krabicový graf)

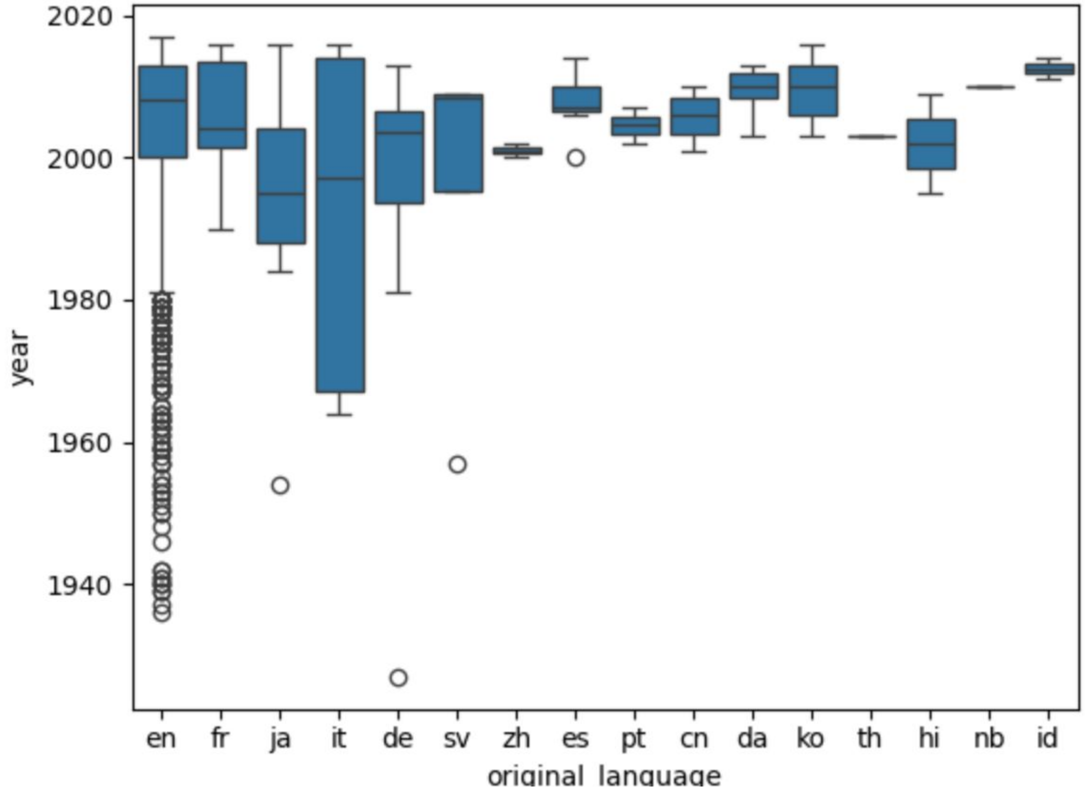
Developed by Mary Eleanor Hunt Spear and John Tukey.

It shows the **five-number summary:** min, $Q1$, median ($Q2$), $Q3$, max

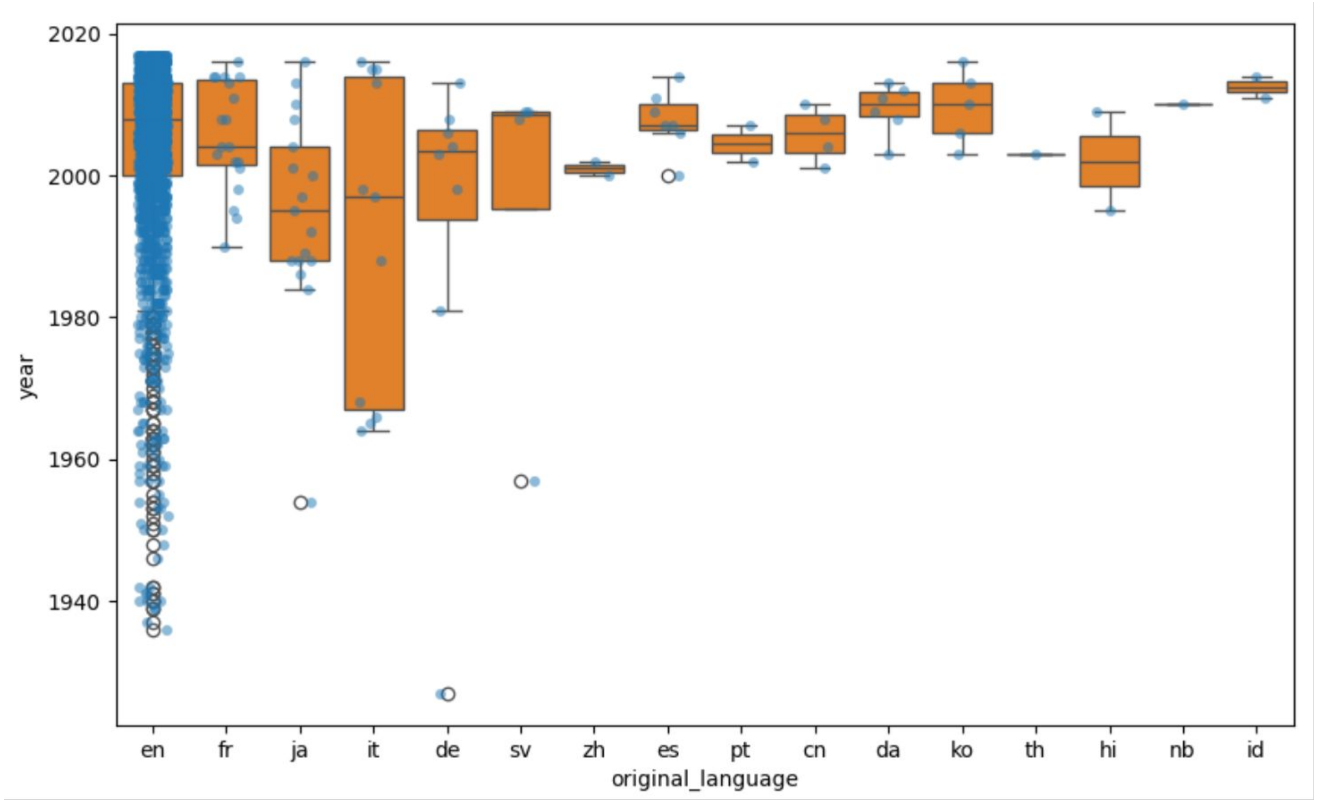
$Q1$ and $Q3$: a box, median: line through box, min and max: whiskers
Outliers often excluded from the whiskers and shown as points.



# Boxplots are used for quick comparison



# For small datasets we may add strip plot

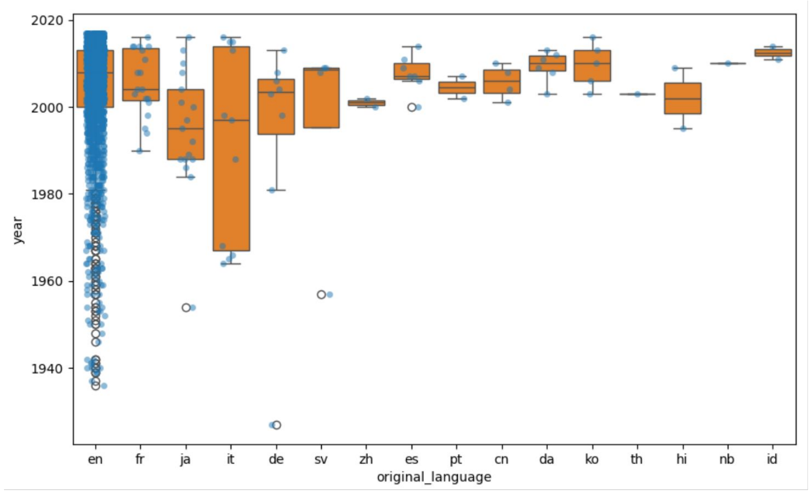



Works well for smaller languages, mess for en.

What do we see for sv, pt, th, hi, nb, id?

# Code for the plot

```
axes = sns.boxplot(data=movies, x='original_language',
                   y='year', color='C1')
sns.stripplot(data=movies, x='original_language',
              y='year', color='C0',
              alpha=0.5, size=5, jitter=0.2)
```



# Quick overview of a data set: `describe` in Pandas

```
movies.describe()
```

| | year | budget | revenue | runtime | vote_average | vote_count |
|---|---|---|---|---|---|---|
| **count** | 2,049.00 | 1,959.00 | 1,965.00 | 2,049.00 | 2,049.00 | 2,049.00 |
| **mean** | 2,004.14 | 55,108,939.70 | 198,565,134.28 | 112.66 | 6.63 | 1,704.64 |
| **std** | 12.70 | 53,139,663.86 | 233,028,732.94 | 24.76 | 0.77 | 1,607.89 |
| **min** | 1,927.00 | 1.00 | 15.00 | 7.00 | 4.00 | 501.00 |
| **25%** | 2,000.00 | 16,000,000.00 | 52,882,018.00 | 97.00 | 6.10 | 709.00 |
| **50%** | 2,008.00 | 38,000,000.00 | 122,200,000.00 | 109.00 | 6.60 | 1,092.00 |
| **75%** | 2,013.00 | 75,000,000.00 | 250,200,000.00 | 124.00 | 7.20 | 2,000.00 |
| **max** | 2,017.00 | 380,000,000.00 | 2,787,965,087.00 | 705.00 | 9.10 | 14,075.00 |

```python
movies.describe(include='all').transpose()
```

| | count | unique | top | freq | mean | std | min | 25% | 50% |
|---|---|---|---|---|---|---|---|---|---|
| **title** | 2049 | 2018 | Beauty and the Beast | 3 | NaN | NaN | NaN | NaN | NaN |
| **year** | 2,049.00 | NaN | NaN | NaN | 2,004.14 | 12.70 | 1,927.00 | 2,000.00 | 2,008.00 |
| **budget** | 1,959.00 | NaN | NaN | NaN | 55,108,939.70 | 53,139,663.86 | 1.00 | 16,000,000.00 | 38,000,000.00 |
| **revenue** | 1,965.00 | NaN | NaN | NaN | 198,565,134.28 | 233,028,732.94 | 15.00 | 52,882,018.00 | 122,200,000.00 |
| **original_language** | 2049 | 16 | en | 1958 | NaN | NaN | NaN | NaN | NaN |
| **runtime** | 2,049.00 | NaN | NaN | NaN | 112.66 | 24.76 | 7.00 | 97.00 | 109.00 |
| **release_date** | 2049 | 1740 | 2014-12-25 | 6 | NaN | NaN | NaN | NaN | NaN |
| **vote_average** | 2,049.00 | NaN | NaN | NaN | 6.63 | 0.77 | 4.00 | 6.10 | 6.60 |
| **vote_count** | 2,049.00 | NaN | NaN | NaN | 1,704.64 | 1,607.89 | 501.00 | 709.00 | 1,092.00 |
| **overview** | 2049 | 2049 | Led by Woody, Andy's toys live happily in his ... | 1 | NaN | NaN | NaN | NaN | NaN |

# Correlation (korelácia)

We are often interested in **relationships** among variables (data columns).

Next: two correlation coefficients that measure strength of such relationships.

Beware: **correlation does not imply causation**

# Correlation does not imply causation

If electricity consumption grows in a very cold weather, there might be **cause-and-effect** relationship: the cold weather is causing people to use more electricity for heating.

If healthier people tend to be happier, which is the cause and which is effect?

Both studied variables can be also influenced by some third, **unknown factor.** For example, within a year, deaths by drowning increase with increased ice cream consumption. Both increases are spurred by warm weather.

The observed correlation can be just a **coincidence**,
see the Redskins rule and a specialized webpage Spurious Correlations.
You can easily find such "coincidences" by comparing many pairs of variables.

# Pearson correlation coefficient

It measures linear relationship between two variables.

Consider pairs of values $(x_1, y_1), \ldots, (x_n, y_n)$, where $(x_i, y_i)$ are two different features of the same object.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

$$r = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{s_x}\right)\left(\frac{y_i - \bar{y}}{s_y}\right).$$

where $s_x$ is the standard deviation of variable $x$.

# Pearson correlation coefficient

$$r = \frac{1}{n-1} \sum_{i=1}^{n} \left( \frac{x_i - \bar{x}}{s_x} \right) \left( \frac{y_i - \bar{y}}{s_y} \right).$$

Expression $(x_i - \bar{x})/s_x$ is called the **standard score** or **z-score:**
how many standard deviations above or below the mean is $x_i$?

The product of z-scores for $x_i$ and $y_i$ is positive
iff they lie on the same side of the respective means of $x$ and $y$.

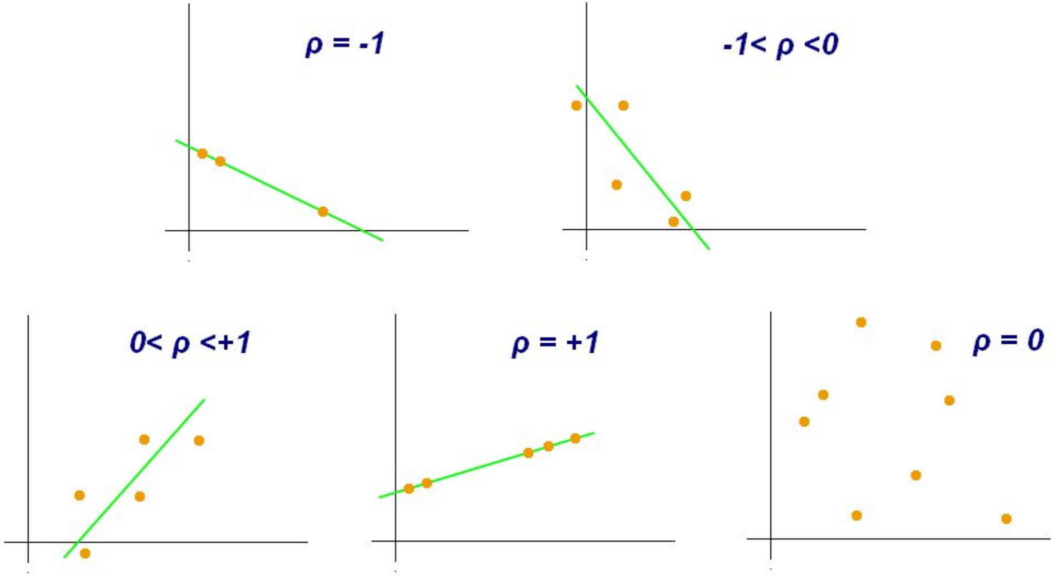
# Properties of Pearson correlation coefficient

The value of $r$ is always from interval $[-1,1]$.

1 if $y$ grows linearly with $x$,

-1 if $y$ decreases linearly

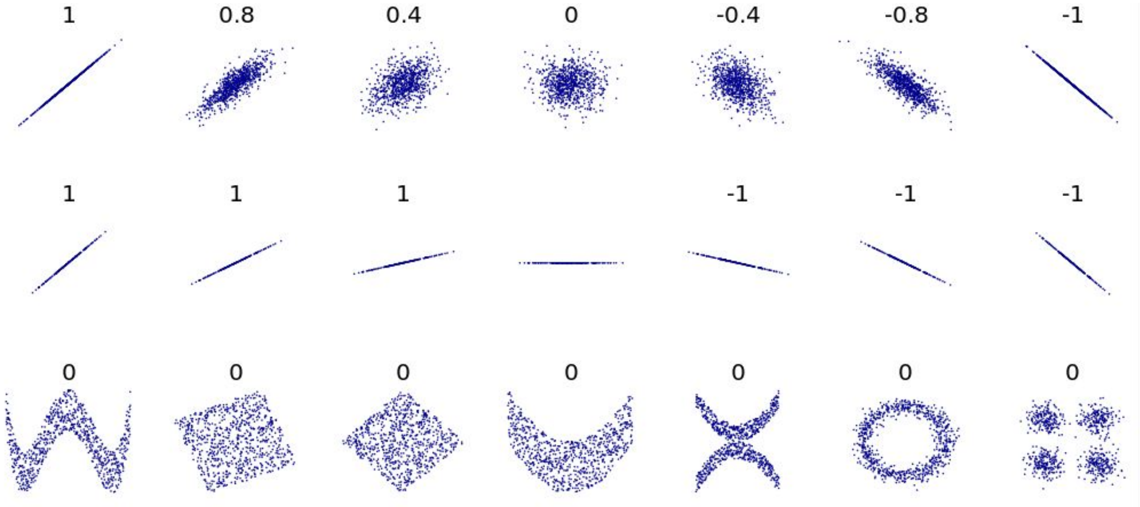
0 means no correlation.




https://commons.wikimedia.org/wiki/File:Correlation_coefficient.png

# Some cautions

Pearson correlation measures only linear relationships (bottom row)
Pearson correlation does not depend on the slope of the best-fit line (middle row)




https://commons.wikimedia.org/wiki/File:Correlation_examples2.svg

# Properties of Pearson correlation

What if we linearly scale each variable, i.e. $ax_i + b$, $cy_i + d$?
What if a>0, a=0, a<0?

What if we switch x and y?

$$r = \frac{1}{n-1} \sum_{i=1}^{n} \left( \frac{x_i - \bar{x}}{s_x} \right) \left( \frac{y_i - \bar{y}}{s_y} \right).$$

# Properties of Pearson correlation

Pearson correlation does not change if we linearly scale each variable, i.e. $ax_i+b$, $cy_i+d$ (for $a,c>0$).

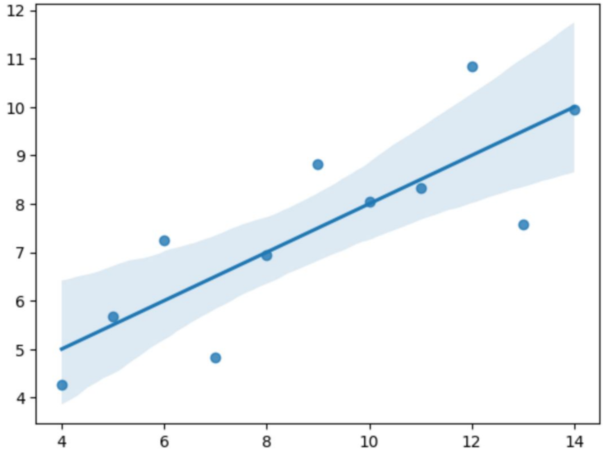
Pearson correlation is symmetric wrt. $x$ and $y$.

Due to reliance on mean and std.dev. it is sensitive to outliers.

$$r = \frac{1}{n-1} \sum_{i=1}^{n} \left( \frac{x_i - \bar{x}}{s_x} \right) \left( \frac{y_i - \bar{y}}{s_y} \right).$$

# Linear regression

The process of finding the line best representing the relationship of $x$ and $y$.
In higher dimensions we can predict one variable as a linear combination of many.
You will study linear regression in later courses.
We may draw regression lines in some plots.




```
x = [10, 8, 13, 9, 11, 14, 6, 4, 12, 7, 5]
y1 = [8.04, 6.95, 7.58, 8.81, 8.33, 9.96,
      7.24, 4.26, 10.84, 4.82, 5.68]
sns.regplot(x=x, y=y1)
```

# Spearman's rank correlation coefficient

It can detect **non-linear relationships**.

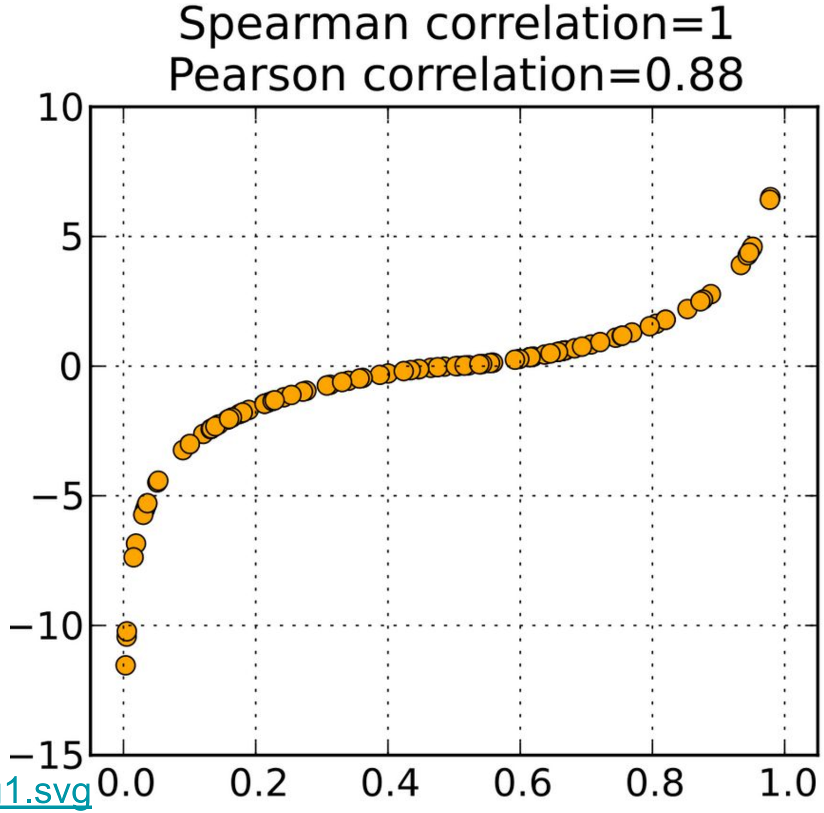
We first convert each variable into **ranks**:
Rank of $x_i$ is its index in the sorted order of $x_1, \ldots, x_n$.
Equal values get the same (average) rank.
For example, the ranks of 10, 0, 10, 20, 10, 20 are 3, 1, 3, 5.5, 3, 5.5.

Then we compute **Pearson correlation coefficient of the two rank sequences**.

# Properties of Spearman's rank correlation coeficient

Values of 1, -1 if $y$ monotonically increases or decreases with $x$.

It is less sensitive to outliers (actual values of $x$ and $y$ are not important).




https://commons.wikimedia.org/wiki/File:Spearman_fig1.svg

# Computation in Pandas

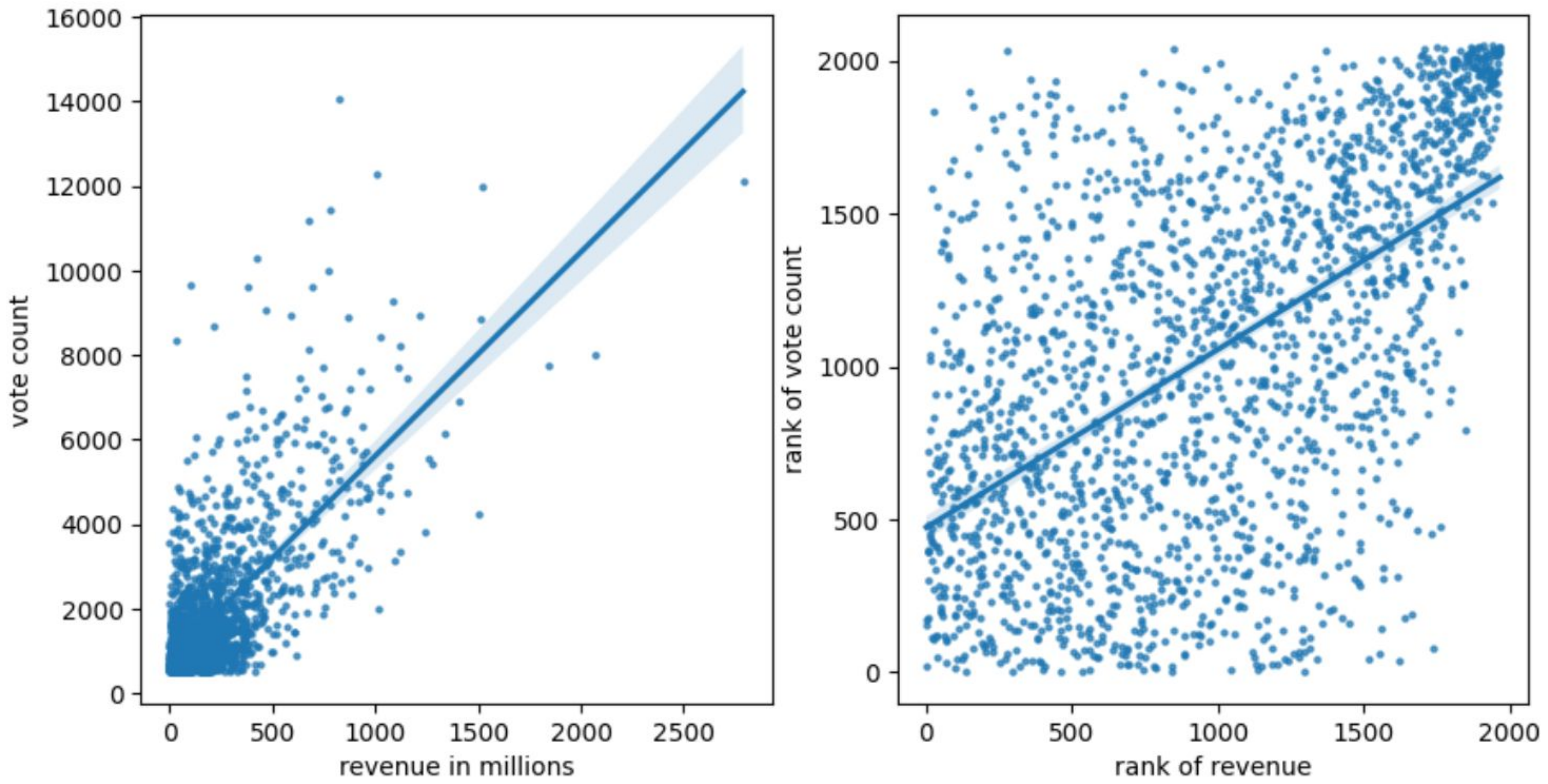
```
movies.corr(numeric_only=True)
```

| | year | budget | revenue | runtime | vote_average | vote_count |
|---|---|---|---|---|---|---|
| **year** | 1.00 | 0.28 | 0.12 | -0.07 | -0.34 | 0.12 |
| **budget** | 0.28 | 1.00 | 0.69 | 0.22 | -0.18 | 0.47 |
| **revenue** | 0.12 | 0.69 | 1.00 | 0.25 | 0.06 | 0.69 |
| **runtime** | -0.07 | 0.22 | 0.25 | 1.00 | 0.31 | 0.25 |
| **vote_average** | -0.34 | -0.18 | 0.06 | 0.31 | 1.00 | 0.33 |
| **vote_count** | 0.12 | 0.47 | 0.69 | 0.25 | 0.33 | 1.00 |

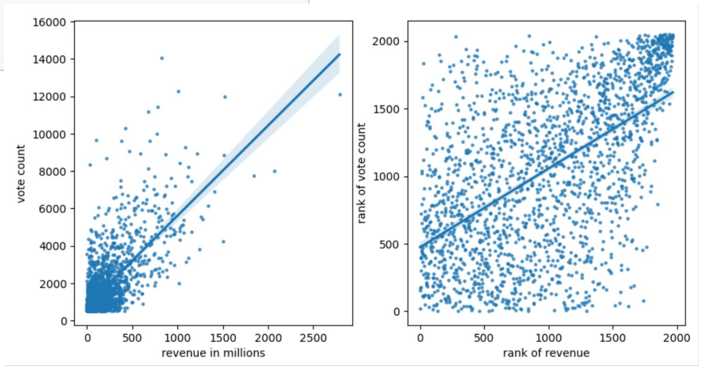
# Computation in Pandas (cont.)

```
movies.corr(method='spearman', numeric_only=True)
```

| | year | budget | revenue | runtime | vote_average | vote_count |
|---|---|---|---|---|---|---|
| **year** | 1.00 | 0.21 | 0.02 | -0.03 | -0.27 | 0.14 |
| **budget** | 0.21 | 1.00 | 0.68 | 0.24 | -0.28 | 0.37 |
| **revenue** | 0.02 | 0.68 | 1.00 | 0.21 | -0.08 | 0.56 |
| **runtime** | -0.03 | 0.24 | 0.21 | 1.00 | 0.32 | 0.27 |
| **vote_average** | -0.27 | -0.28 | -0.08 | 0.32 | 1.00 | 0.29 |
| **vote_count** | 0.14 | 0.37 | 0.56 | 0.27 | 0.29 | 1.00 |


16000
14000
12000
10000
8000
6000
4000
2000
0
vote count
0
500
1000
1500
2000
2500
revenue in millions
2000
1500
1000
500
0
rank of vote count
0
500
1000
1500
2000
rank of revenue

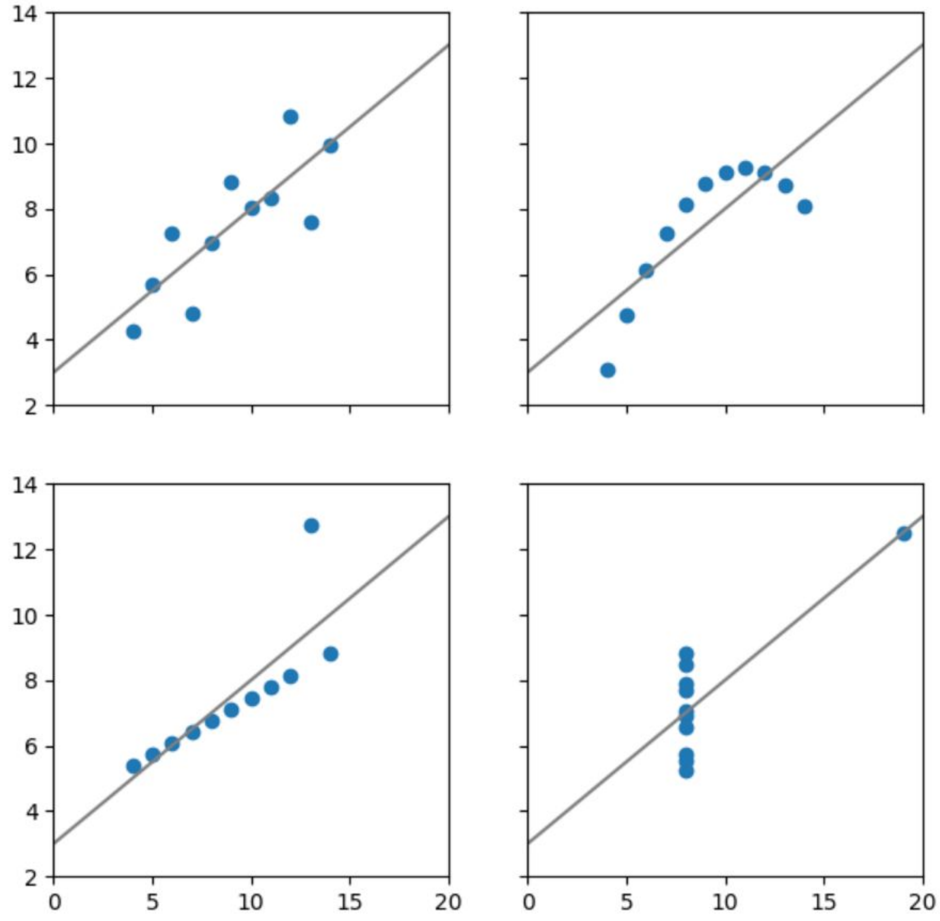
```python
figure, axes = plt.subplots(1, 2, figsize=(10,5))
# plot of values
sns.regplot(x=movies['revenue'] / 1e6, y=movies['vote_count'],
            ax=axes[0], scatter_kws={'alpha':0.7, 's':5})
axes[0].set_xlabel('revenue in millions')
axes[0].set_ylabel('vote count')
# compute ranks
revenue_rank = movies['revenue'].rank()
vote_count_rank = movies['vote_count'].rank()
# plot of ranks
sns.regplot(x=revenue_rank, y=vote_count_rank,
            ax=axes[1], scatter_kws={'alpha':0.7, 's':5})
axes[1].set_xlabel('rank of revenue')
axes[1].set_ylabel('rank of vote count')
```



# Anscombe's quartet

Four **artificial data sets** designed by Francis Anscombe.

The same or very similar values of: means and variances of $x$ and $y$, Pearson correlation coefficient (0.816) and linear regression line.

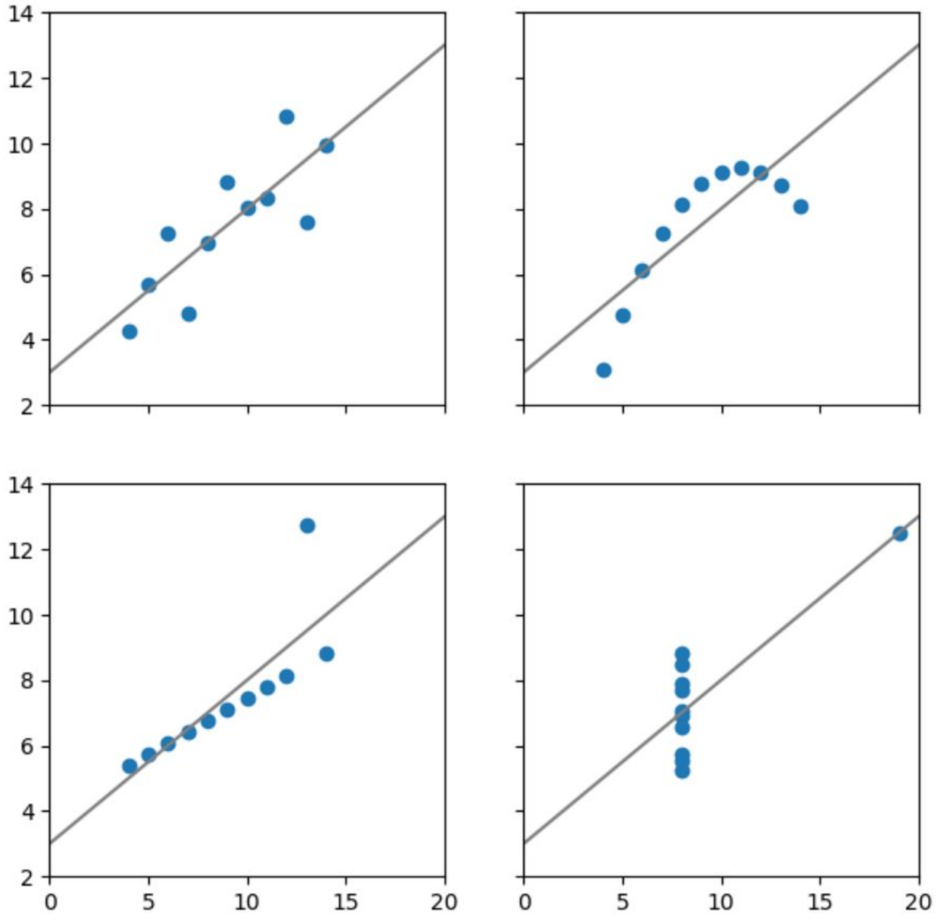


# Anscombe's quartet

The same summary statistics,
but visually very different

**Importance of visualization:**
Plots often give us a much better idea of the properties of a data set than simple numerical summaries.

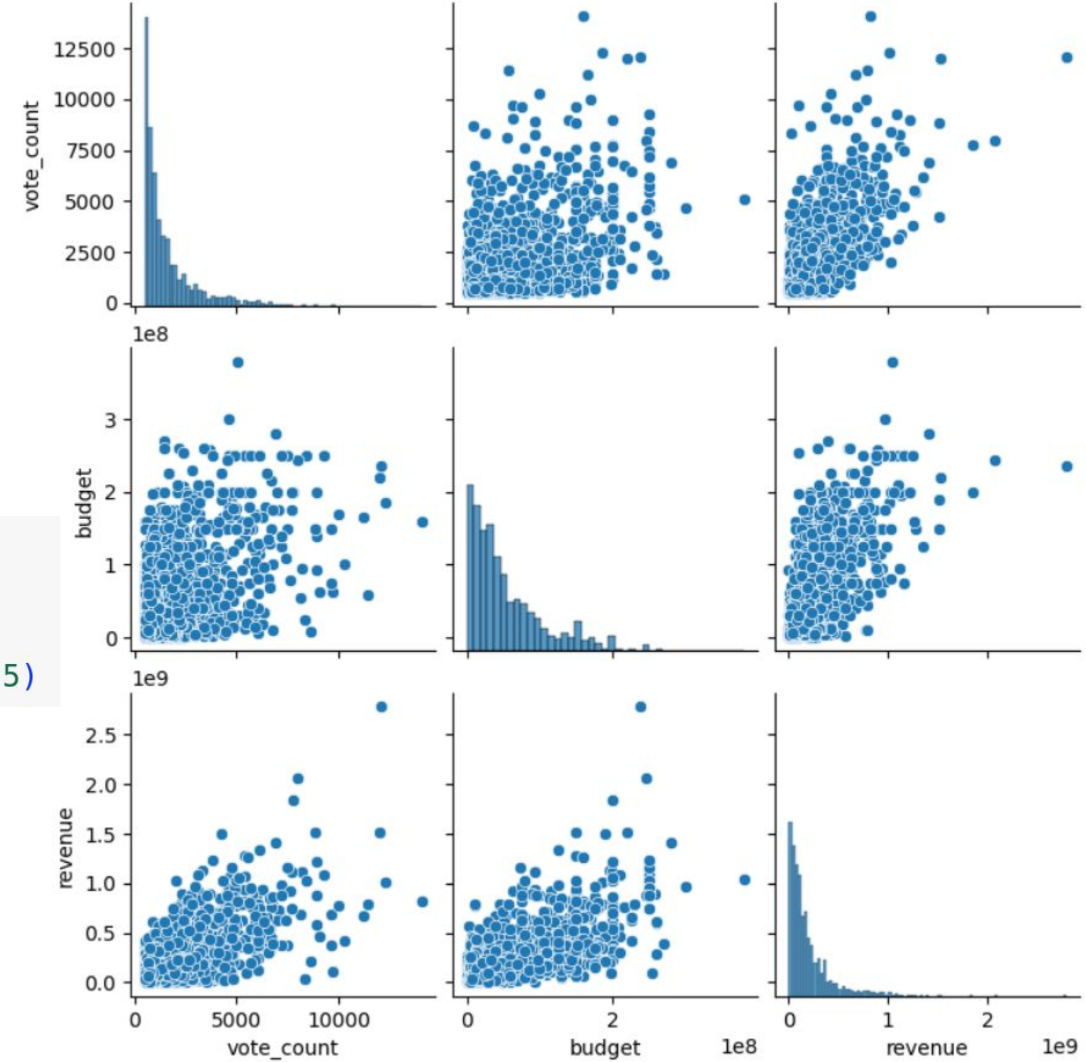
The bottom row illustrates the influence of outliers on correlation and regression.



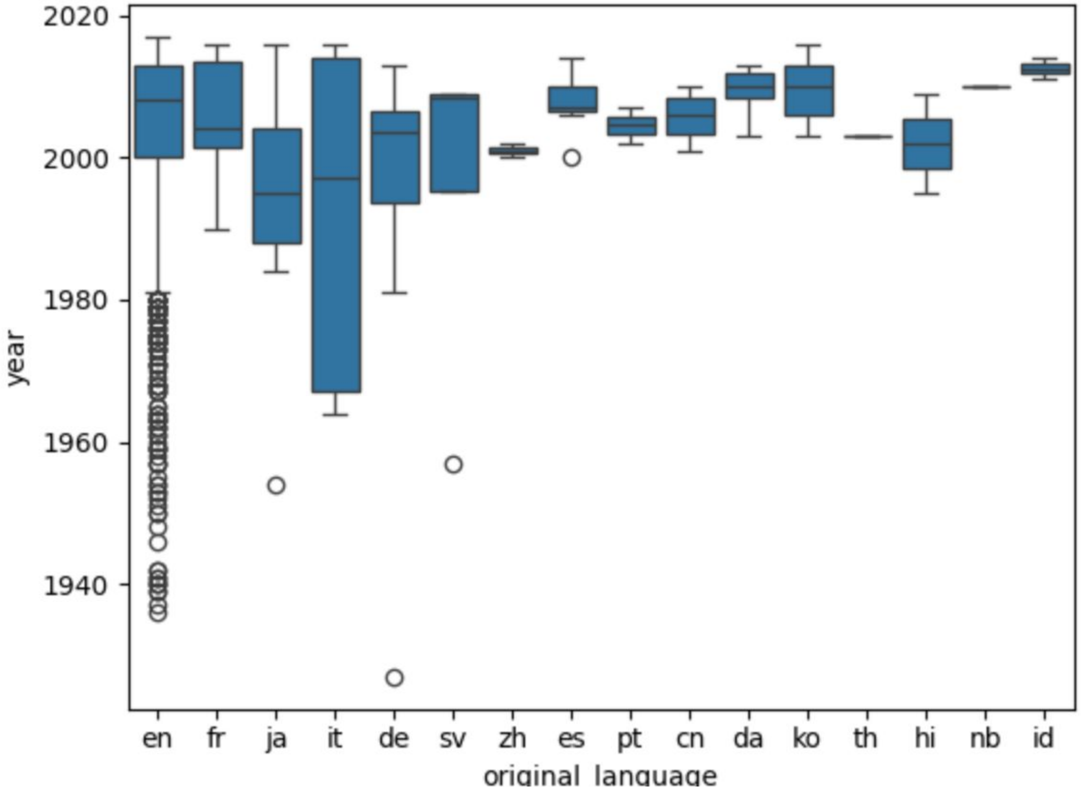
# Visual overview of a data set: `pairplot` in Seaborn

```
subset = movies.loc[:, ['vote_count',
                        'budget',
                        'revenue']]
grid = sns.pairplot(subset, height=2.5)
```

All histograms + scatterplots



# How to compute summaries for groups?



# How to compute summaries for groups?

Method `groupby` splits the table into groups based on values of some column.

We can apply a summary statistics function or `describe` on each group.

```python
movies.groupby('original_language').median(numeric_only=True).head()
```

| | year | budget | revenue | runtime | vote_average | vote_count |
|---|---|---|---|---|---|---|
| **original_language** | | | | | | |
| **cn** | 2,006.00 | 12,902,809.00 | 39,388,380.00 | 108.50 | 7.20 | 762.50 |
| **da** | 2,010.00 | 10,000,000.00 | 16,740,418.00 | 119.00 | 6.80 | 867.50 |
| **de** | 2,003.50 | 6,250,000.00 | 70,000,000.00 | 129.00 | 7.60 | 669.00 |
| **en** | 2,008.00 | 40,000,000.00 | 126,397,819.00 | 109.00 | 6.60 | 1,126.00 |
| **es** | 2,007.00 | 2,000,000.00 | 30,448,000.00 | 118.00 | 7.60 | 797.00 |

```
subset = movies.loc[:, ['original_language', 'year', 'budget']]
subset.groupby('original_language').describe().head(3).transpose()
```

| | original_language | cn | da | de |
|---|---|---|---|---|
| year | count | 4.00 | 6.00 | 8.00 |
| | mean | 2,005.75 | 2,009.33 | 1,992.50 |
| | std | 4.03 | 3.61 | 28.13 |
| | min | 2,001.00 | 2,003.00 | 1,927.00 |
| | 25% | 2,003.25 | 2,008.25 | 1,993.75 |
| | 50% | 2,006.00 | 2,010.00 | 2,003.50 |
| | 75% | 2,008.50 | 2,011.75 | 2,006.50 |
| | max | 2,010.00 | 2,013.00 | 2,013.00 |
| budget | count | 3.00 | 5.00 | 8.00 |
| | mean | 14,872,795.67 | 13,440,000.00 | 18,223,718.75 |

# Summary

**Summary statistics**:

- mean, median, mode
- percentiles, quantiles, quartiles
- min, max, interquartile range, variance, standard deviation
- Pearson and Spearman correlation

More details in a statistics course

**Visualization:**

- boxplot
- scatter plots with regression lines
- pairplot

**Pandas:**

- functions for computing statistics, `describe`
- `groupby`

Next week: more Pandas