1 Lecture 4: Summary statistics

Data Visualization · 1-DAV-105

Lecture by Broňa Brejová

1.1 Introduction

- Summary statistics (popisné charakteristiky / štatistiky) are 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, such as histograms, they give us the first glimpse at the data when working with a new data set.
- We start by loading the movie data set, which we use to illustrate these terms.

1.2 Importing the movie data set

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

[1]:
```
import numpy as np
import pandas as pd
from IPython.display import Markdown
import matplotlib.pyplot as plt
import seaborn as sns
pd.options.display.float_format = '{:,.2f}'.format
```

[2]:
```
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  \
0  Toy Story  1995 30,000,000.00 373,554,033.00                en    81.00
1    Jumanji  1995 65,000,000.00 262,797,249.00                en   104.00
2       Heat  1995 60,000,000.00 187,436,818.00                en   170.00
3  GoldenEye  1995 58,000,000.00 352,194,034.00                en   130.00
4     Casino  1995 52,000,000.00 116,112,375.00                en   178.00

  release_date  vote_average  vote_count  \
0   1995-10-30          7.70    5,415.00
1   1995-12-15          6.90    2,413.00
2   1995-12-15          7.70    1,886.00
3   1995-11-16          6.60    1,194.00
4   1995-11-22          7.80    1,343.00

                                            overview
0  Led by Woody, Andy's toys live happily in his …
```

```
1    When siblings Judy and Peter discover an encha…
2    Obsessive master thief, Neil McCauley leads a …
3    James Bond must unmask the mysterious head of …
4    The life of the gambling paradise - Las Vegas …
```

1.3 Measures of central tendency (miery stredu / polohy)

These represent a typical value in a sample x with values $x_1, \dots, x_n$ (one numerical column of a table).

- **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 from smallest to largest.
  - For even n usually defined as the average of the two middle values.
  - Median of 10, 12, 15, 16, 16 is 15.
  - Median of 10, 12, 15, 16, 16, 20 is 15.5.
- **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 Cmglee, CC BY-SA 3.0

1.3.1 Properties of the measures

- If we apply linear transformation $a \cdot x_i + b$ with the same constants a and b to all values x_i, mean, median and mode will be also 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.
  - Mean of 800, 1000, 1100, 1200, 1800, 2000 and 30000 is 5414.3, median 1200.
  - Mean of 800, 1000, 1100, 1200, 1800, 2000 and 10000 is 2557.1, median 1200.
- Therefore we often prefer median (e.g. median salary).

1.3.2 Computation in Pandas

Below we apply functions `mean`, `median`, `mode` to a Series (column `year` of our table).

Note that `mode` returns a Series of results (for case of ties). Here just a single value 2013.

Note the use of Python f-strings to print the results.

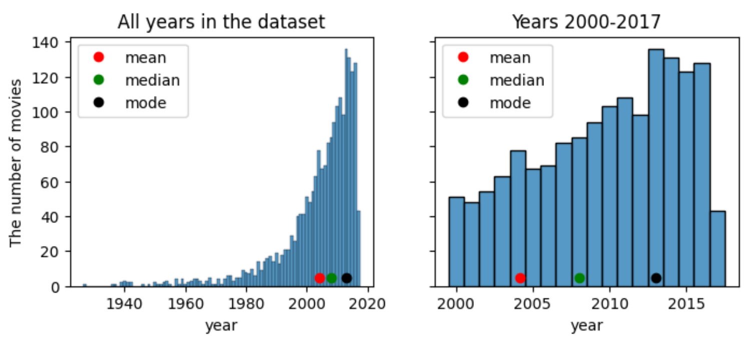
[3]:
```python
display(Markdown("**Properties of the column `year` in our table:**"))
print(f"Mean: {movies['year'].mean():.2f}")
print(f"Median: {movies['year'].median()}")
print(f"Mode:\n{movies['year'].mode()}")
```

Properties of the column `year` in our table:

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

Let us see these values in a histogram of the column values (overall view and detail).

[4]:
```python
# set up figure with two plots
figure, axes = plt.subplots(1, 2, figsize=(8,3), sharey=True)

# plot histograms, use discrete=True to have each year in one bin
sns.histplot(data=movies, x='year', discrete=True, ax=axes[0])
sns.histplot(x=movies.query('year>=2000')['year'], discrete=True, ax=axes[1])

# titles and axis labels
axes[0].set_ylabel("The number of movies")
axes[0].set_title('All years in the dataset')
axes[1].set_title('Years 2000-2017')

# compute three summary statics, set up their color and label
stats = [{'label':'mean', 'value':movies['year'].mean(), 'color':'red'},
         {'label':'median', 'value':movies['year'].median(), 'color':'green'},
         {'label':'mode', 'value':movies['year'].mode(), 'color':'black'}]
# add dots for all statistics to both plots (at y=5)
for a in axes:
  for s in stats:
    a.plot(s['value'], 5, 'o', color=s['color'], label=s['label'])
  a.legend()

pass
```



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

[5]:
```
display(Markdown("**`movies.mean(numeric_only=True)`:**"), movies.
  ↪mean(numeric_only=True))
display(Markdown("**`movies.median(numeric_only=True)`:**"), movies.
  ↪median(numeric_only=True))
```

`movies.mean(numeric_only=True)`:

```
year                  2,004.14
budget           55,108,939.70
revenue         198,565,134.28
runtime                 112.66
vote_average              6.63
vote_count            1,704.64
dtype: float64
```

`movies.median(numeric_only=True)`:

```
year                  2,008.00
budget           38,000,000.00
revenue         122,200,000.00
runtime                 109.00
vote_average              6.60
vote_count            1,092.00
dtype: float64
```

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

- Median is the middle value in a sorted order.
- Therefore about 50% of values are smaller and 50% larger.
- For a different percentage p, the 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.
- Specifically **quartiles** are three values Q_1, Q_2 and Q_3 that split input data into quarters.
  - Therefore, Q_2 is the median.
- Many definitions exist regarding situations when the desired fraction falls between two values (we can take lower, higher, mean, weighted mean etc).

1.4.1 Computation in Pandas

- Function `quantile` gets a single value between 0 and 1 or a list of values and returns corresponding quantiles.
- To get quantiles for 0.1, 0.2, ..., 0.9, we generate a regular sequence of values using `np.arange`.

[6]:
```
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:

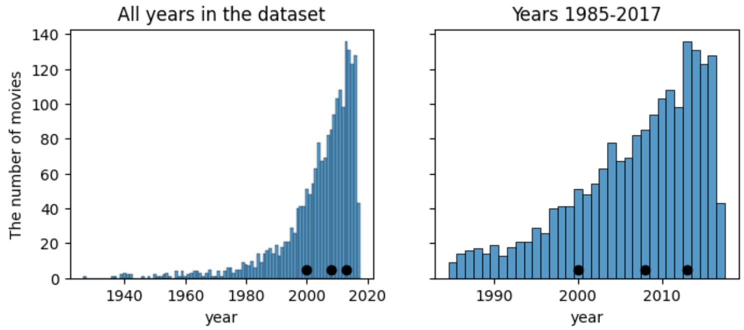
```
0.25    2,000.00
0.50    2,008.00
0.75    2,013.00
Name: year, dtype: float64
```

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
```

The code below plots the quartiles highlighted in a histogram.

[7]:
```
# setup histograms
figure, axes = plt.subplots(1, 2, figsize=(8,3), sharey=True)
sns.histplot(data=movies, x='year', discrete=True, ax=axes[0])
sns.histplot(x=movies.query('year>=1985')['year'], discrete=True, ax=axes[1])
axes[0].set_ylabel("The number of movies")
axes[0].set_title('All years in the dataset')
axes[1].set_title('Years 1985-2017')

# compute and display quartiles
quartiles = movies['year'].quantile([0.25, 0.5, 0.75])
for a in axes:
    a.plot(quartiles, [5] * len(quartiles), 'o', color='black')
pass
```



- The code below illustrates how the `quantile` function works when returning quantiles which do not correspond to a single input value.
- Optional parameter `interpolation` accepts values `'linear'` `(default)`, `'lower'`, `'higher'`, `'midpoint'`, `'nearest'`.
- Imagine the lowest element at quantile 0, the highest element at quantile 1 and the rest evenly spaced between. The quantile at position between two elements is influenced only by its two neighbors.

For example, consider list of values [0,10,20,100]. * Values taken from the list: $p = 0$: 0, $p = 1/3$: 10, $p = 2/3$: 20, $p = 1$: 100 * Example of a different value: $p = 1/4$: by default linear interpolation at 3/4 between 0 and 10, i.e. 7.5.

Note that linear interpolation is continuous as p changes from 0 to 1.

[8]:
```
a = pd.Series([0, 100])
b = pd.Series([0, 10, 20, 30, 100])
c = pd.Series([0, 10, 20, 100])
quantiles = [0.01, 0.25, 0.5, 0.75]
display(Markdown(f"**Quantiles for {list(a)}**"), a.quantile(quantiles))
display(Markdown(f"**Quantiles for {list(b)}**"), b.quantile(quantiles))
display(Markdown(f"**Quantiles for {list(c)}**"), c.quantile(quantiles))
display(Markdown(f"**Quantiles for {list(c)} with `interpolation='lower'`**"),
        c.quantile(quantiles, interpolation='lower'))
```

Quantiles for [0, 100]

```
0.01     1.00
0.25    25.00
0.50    50.00
0.75    75.00
dtype: float64
```

Quantiles for [0, 10, 20, 30, 100]

```
0.01     0.40
0.25    10.00
0.50    20.00
```

```
0.75   30.00
dtype: float64
```

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
```

1.5 Measures of variability (miery variability)

- Values in the sample may be close to their mean or median, or they can spread widely.
- It is important to consider how representative is the mean or median of the whole set.

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 (described next).

1.5.1 Variance and standard deviation (rozptyl a smerodajná odchýlka)

Variance

- For each value in the sample compute its difference from the mean and square it: $(x_i - \bar{x})^2$.
- After squaring, we get 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:

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

- We divide by $n-1$ rather than n, because we would otherwise underestimate the true variance of the underlying population (more in the statistics course).
- For large n, the difference between dividing by n and $n-1$ is negligible.

Standard deviation

- Square root of the variance

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

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

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 same constants a and b to all values 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:
  - For 800, 1000, 1100, 1200, 1800, 2000, 30000 st. dev. is 10850.0, IQR 850.
  - For 800, 1000, 1100, 1200, 1800, 2000, 10000 st. dev. is 3310.5, IQR 850.

1.5.2 Computation in Pandas

We can use functions `min`, `max`, `std`, `var`, which work similarly to `mean`.

[9]:
```
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 interquartile range:**"), q1, q3, q3-q1)
```

Minimum

```
1927
```

Maximum

```
2017
```

Mean

```
2004.1449487554905
```

Variance

```
161.2714600681735
```

Standard deviation

```
12.699270060447313
```

Q1, Q3 and interquartile range:

```
2000.0
```

```
2013.0
```

```
13.0
```

1.6 Outliers (odľahlé hodnoty)

- Outliers are the values which are far from the typical range of values.
- In data analysis, it is important to **check the 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.
- The criterion by statistician John Tukey is often used:
  - 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 outliers values (as long as they stay outside of range Q1-Q3).

1.6.1 Computation in Pandas

- The code below finds outliers in the `year` column.
- We compute the lower and upper thresholds manually from quartiles.
- Then we use `query` to select rows and count how many there are.
- Function `count` counts the values in a Series or columns of a DataFrame, ignoring missing values.

[10]:
```
# 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()
# print results
display(Markdown(f"**Outliers outside of range:** [{lower}, {upper}]"))
display(Markdown(f"**Outlier count:** {count}"))
display(Markdown(f"**Total count:** {movies['year'].count()}"))
```

Outliers outside of range: [1980.5, 2032.5]

Outlier count: 112

Total count: 2049

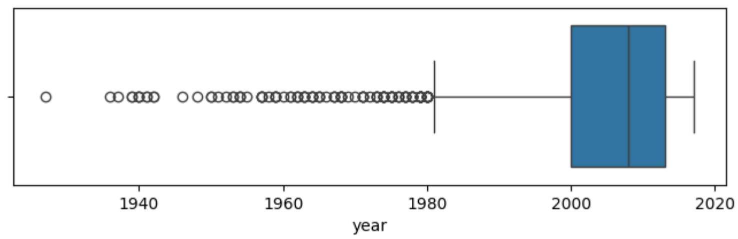
1.7 Boxplot (krabicový graf)

- Boxplots were developed by Mary Eleanor Hunt Spear and John Tukey.
- For a single numerical variable it shows the **five-number summary** consisting of the minimum, Q_1, median (Q_2), Q_3 and the maximum.
- Median is shown as a thick line, Q_1 and Q_3 as a box and minimum and maximum as “whiskers”.
- Outliers are often excluded from the whiskers and shown as individual points.
- Summaries of different samples are often compared in a single boxplot.
- Boxplots allow clear comparison of basic characteristics.

1.7.1 Boxplots in Seaborn

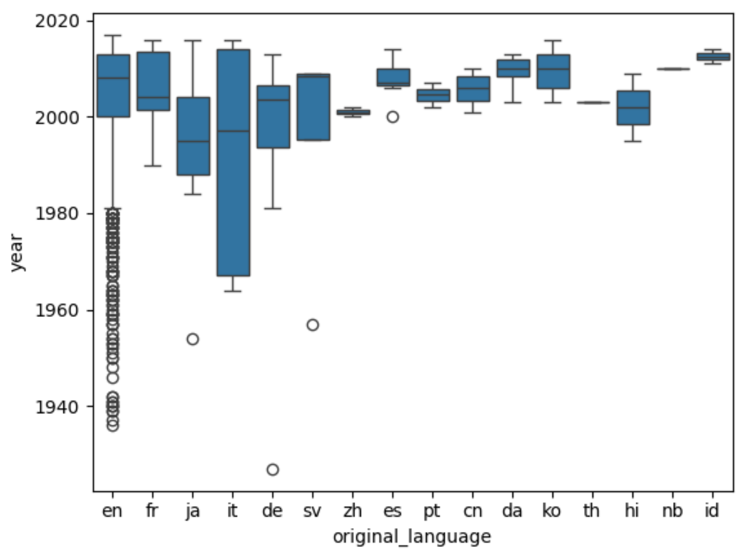
- We use `boxplot` finction from Seaborn.
- Below is a simple horizontal boxplot of the `year` column.
- Recall that quartiles are 2000, 2008 and 2013, minimum 1927, maximum 2017, outliers outside of [1980.5, 2032.5].

[11]:
```
axes = sns.boxplot(data=movies, x='year')
axes.figure.set_size_inches(8,2)
```



- Below is a vertical boxplot of the `year` column split into groups according to language.
- This is achieved by specifying both `x` and `y` options.

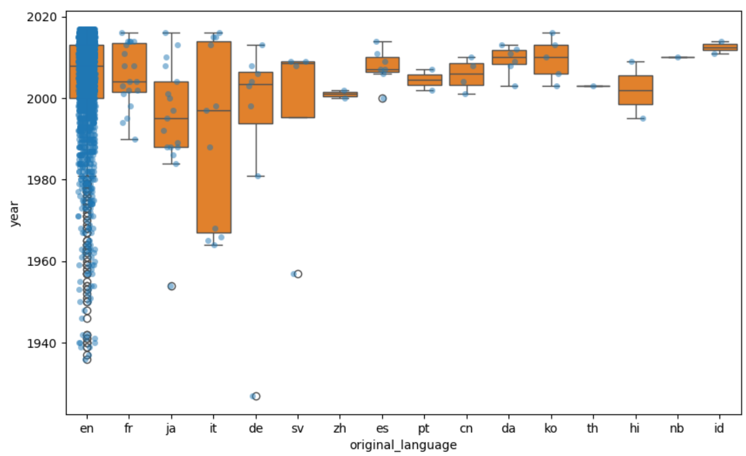
[12]:
```
sns.boxplot(data=movies, x='original_language', y='year')
pass
```



- Below we draw a strip plot on top of the boxplot.
- This allows us to see both individual data points and the summary.
- Here it does not work very well for `en`, better suited for smaller datasets.

- We see that some languages have extremely low number of points, boxplots not ideal in that case.

[13]:
```python
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)
axes.figure.set_size_inches(10,6)
pass
```



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

Function `describe` gives a quick overview of a data set with many statistics described today.

[14]:
```python
movies.describe()
```

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

       vote_count
count    2,049.00
mean     1,704.64
std      1,607.89
min        501.00
25%        709.00
50%      1,092.00
```

```
75%       2,000.00
max      14,075.00
```

- By default `describe` only considers numerical columns.
- Other columns can be included by `include='all'`.
- Different statistics reported for categorical columns (`unique`, `top`, `freq`).

[15]:
```
movies.describe(include='all').transpose()
```

[15]:
```
                     count  unique  \
title                 2049    2018
year              2,049.00     NaN
budget            1,959.00     NaN
revenue           1,965.00     NaN
original_language     2049      16
runtime           2,049.00     NaN
release_date          2049    1740
vote_average      2,049.00     NaN
vote_count        2,049.00     NaN
overview              2049    2049

                                                           top  freq  \
title                                     Beauty and the Beast     3
year                                                       NaN   NaN
budget                                                     NaN   NaN
revenue                                                    NaN   NaN
original_language                                           en  1958
runtime                                                    NaN   NaN
release_date                                        2014-12-25     6
vote_average                                               NaN   NaN
vote_count                                                 NaN   NaN
overview           Led by Woody, Andy's toys live happily in his …     1

                           mean            std      min           25%  \
title                       NaN            NaN      NaN           NaN
year                   2,004.14          12.70 1,927.00      2,000.00
budget            55,108,939.70  53,139,663.86     1.00 16,000,000.00
revenue          198,565,134.28 233,028,732.94    15.00 52,882,018.00
original_language           NaN            NaN      NaN           NaN
runtime                  112.66          24.76     7.00         97.00
release_date                NaN            NaN      NaN           NaN
vote_average               6.63           0.77     4.00          6.10
vote_count             1,704.64       1,607.89   501.00        709.00
overview                    NaN            NaN      NaN           NaN

                            50%            75%              max
title                       NaN            NaN              NaN
```

year	2,008.00	2,013.00	2,017.00
budget	38,000,000.00	75,000,000.00	380,000,000.00
revenue	122,200,000.00	250,200,000.00	2,787,965,087.00
original_language	NaN	NaN	NaN
runtime	109.00	124.00	705.00
release_date	NaN	NaN	NaN
vote_average	6.60	7.20	9.10
vote_count	1,092.00	2,000.00	14,075.00
overview	NaN	NaN	NaN

1.9 Correlation (korelácia)

- We are often interested in relationships among different variables (data columns).
- We will see two correlation coefficients that measure the strength of such relationships.
- Beware: **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 (a practice called data dredging).

1.9.1 Pearson correlation coefficient

- It measures linear relationship between two variables.
- Consider pairs of values $(x_1, y_1), \dots, (x_n, y_n)$, where (x_i, y_i) are two different features of the same object.

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

- Or equivalently:

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

- where s_x is the standard deviation of variable x.
- Expression $(x_i - \bar{x})/s_x$ is called the **standard score** or **z-score**, and it tells us how many standard deviations above or below the mean value x_i is.
- The product of $(x_i - \bar{x})/s_x$ and $(y_i - \bar{y})/s_y$ is positive if x_i and y_i lie on the same side of the respective means of x and y and negative if they lie on the opposite sides.

1.9.2 Properties of Pearson correlation coefficient

Values of Pearson correlation coefficient

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

- It is 1 if y grows linearly with x, -1 if y decreases linearly with increasing x.
- Zero means no correlation.
- Values between 0 and 1 mean intermediate value of positive correlation, values between -1 and 0 negative correlation.

https://commons.wikimedia.org/wiki/File:Correlation_coefficient.png Kiatdd, CC BY-SA 3.0

Some cautions

- Pearson correlation measures only linear relationships (x and y in the bottom row have non-linear relationships but their correlation is 0).
- Pearson correlation does not depend on the slope of the best-fit line (see the middle row below).

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

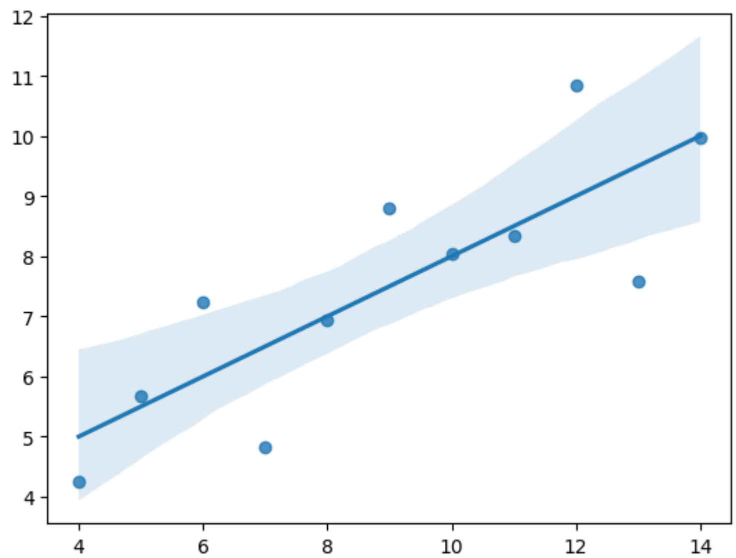
Other properties

- 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 $r_{x,y} = r_{y,x}$.

1.9.3 Linear regression

- The process of finding the line best representing the relationship of x and y is called linear regression.
- It can be used in higher dimensions to predict one variable as a linear combination of many others.
- You will study linear regression in later courses, but we may draw regression lines in some plots.

[16]:
```
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)
pass
```



1.9.4 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 $x_1, \dots, x_n$.
  - 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.
- Values of 1, -1 if y monotonically increases or decreases with x.
- It is less sensitive to distant outliers (actual values of x and y are not important).

https://commons.wikimedia.org/wiki/File:Spearman_fig1.svg Skbkekas, CC BY-SA 3.0

1.9.5 Computation in Pandas

Function `corr` computes correlation between all pairs of numerical columns. There is also a version to compare two Series.

In our table, the highest Pearson correlation is 0.69 for pairs (`budget`, `revenue`), (`vote_count`, `revenue`)

[17]:
```
movies.corr(numeric_only=True)
```

[17]:

	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

With Spearman rank correlation, the correlation between `revenue` and `budget` remains similar, but correlation between `vote_count` and `budget` decreases from 0.69 to 0.56.

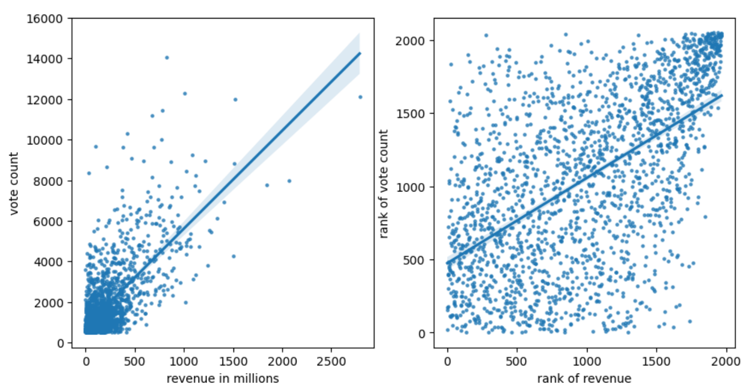
[18]:
```
movies.corr(method='spearman', numeric_only=True)
```

[18]:

	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

- Here we illustrate the regression line for `revenue` versus `vote_count`.
- We use Seaborn `regplot` to draw scatterplot together with the regression line.
- Points are made smaller and transparent by `scatter_kws={'alpha':0.7, 's':5}`.
- The plot on the right shows ranks instead of actual values.
- Ranks are computed using rank function for Series.
- Pearson correlation coefficient is probably driven by outliers.

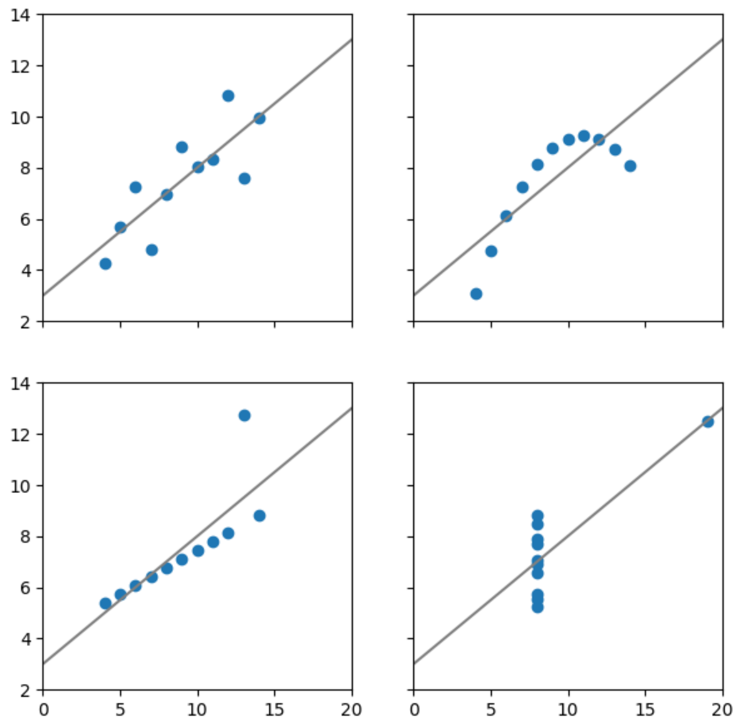
[19]:
```
# figure with two plots
figure, axes = plt.subplots(1, 2, figsize=(10,5))
# plot of values
sns.regplot(x=movies['revenue'] / 1e6, y=movies['vote_count'],
            ax=axes[0], scatter_kws={'alpha':0.7, 's':5})
axes[0].set_xlabel('revenue in millions')
axes[0].set_ylabel('vote count')
# compute 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')
pass
```



1.10 Anscombe's quartet and importance of visualization

- Anscombe's quartet are four artificial data sets designed by Francis Anscombe.
- All have the same or very similar values of means and variances of both x and y, Pearson correlation coefficient (0.816) and linear regression line.
- But visually we see each has a very different character.
- The bottom row illustrates the influence of outliers on correlation and regression.
- Overall this shows that plots give us a much better idea of the properties of a data set than simple numerical summaries.

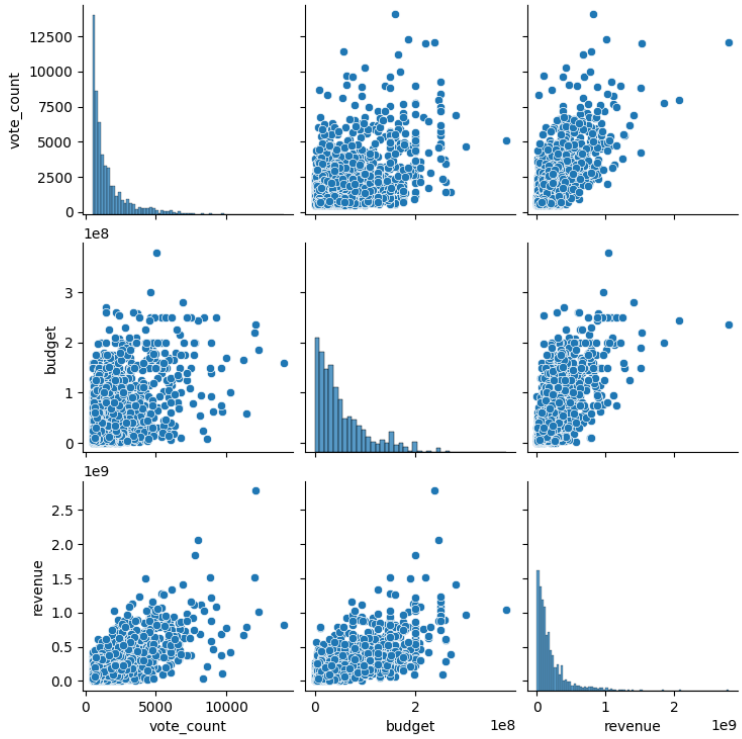
[20]:
```
# adapted from https://matplotlib.org/stable/gallery/specialty_plots/anscombe.
↪html
x = [10, 8, 13, 9, 11, 14, 6, 4, 12, 7, 5]
y1 = [8.04, 6.95, 7.58, 8.81, 8.33, 9.96, 7.24, 4.26, 10.84, 4.82, 5.68]
y2 = [9.14, 8.14, 8.74, 8.77, 9.26, 8.10, 6.13, 3.10, 9.13, 7.26, 4.74]
y3 = [7.46, 6.77, 12.74, 7.11, 7.81, 8.84, 6.08, 5.39, 8.15, 6.42, 5.73]
x4 = [8, 8, 8, 8, 8, 8, 8, 19, 8, 8, 8]
y4 = [6.58, 5.76, 7.71, 8.84, 8.47, 7.04, 5.25, 12.50, 5.56, 7.91, 6.89]
datasets = [(x, y1), (x, y2), (x, y3), (x4, y4)]

figure, axes = plt.subplots(2, 2, sharex=True, sharey=True, figsize=(7, 7))
axes[0, 0].set(xlim=(0, 20), ylim=(2, 14))

for ax, (x, y) in zip(axes.flat, datasets):
    ax.plot(x, y, 'o')
    # linear regression
    slope, intercept = np.polyfit(x, y, deg=1)
    ax.axline(xy1=(0, intercept), slope=slope, color='gray')
```



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

- Seaborn `pairplot` generates a matrix of plots for all numerical columns.
- The diagonal contains histograms of individual columns.
- Off-diagonal entries are scatterplots of two columns.
- Here only 3 columns shown for simpler examination.

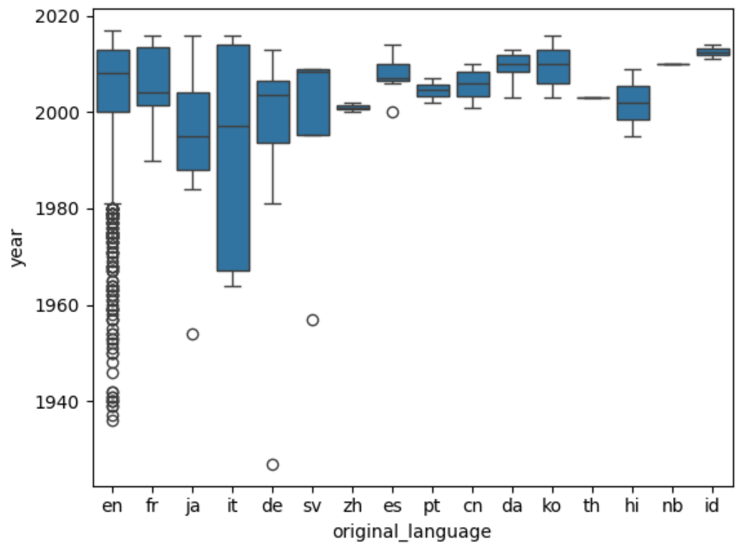
[21]:
```
subset = movies.loc[:, ['vote_count', 'budget', 'revenue']]
grid = sns.pairplot(subset, height=2.5)
pass
```



1.11 Computing summaries of subsets of data: `groupby` from Pandas

- We have seen that Seaborn can create plots where data are split into groups according to a categorical variable.
- One example are boxplots, which we have seen today.
- How can we compute summary statistics for each such group in Pandas?

[22]:
```
sns.boxplot(data=movies, x='original_language', y='year')
pass
```



- Pandas DataFrame supports function `groupby` which splits the table into groups based on values of some column.
- We can apply a summary statistics function on each group.
- Below we compute medians of all numerical columns for each language and show the first 5 languages.

[23]:
```
movies.groupby('original_language').median(numeric_only=True).head()
```

[23]:
```
                      year        budget        revenue  runtime  \
original_language
cn                2,006.00 12,902,809.00  39,388,380.00   108.50
da                2,010.00 10,000,000.00  16,740,418.00   119.00
de                2,003.50  6,250,000.00  70,000,000.00   129.00
en                2,008.00 40,000,000.00 126,397,819.00   109.00
es                2,007.00  2,000,000.00  30,448,000.00   118.00

                   vote_average  vote_count
original_language
cn                         7.20      762.50
da                         6.80      867.50
de                         7.60      669.00
en                         6.60    1,126.00
es                         7.60      797.00
```

- We can also apply `describe` on the `groupby` groups.
- Here only two numerical columns of the original table are shown.

[24]:
```
subset = movies.loc[:, ['original_language', 'year', 'budget']]
subset.groupby('original_language').describe().head()
```

```
[24]:                        year                                                    \
                            count     mean   std      min      25%      50%      75%
original_language
cn                           4.00 2,005.75  4.03 2,001.00 2,003.25 2,006.00 2,008.50
da                           6.00 2,009.33  3.61 2,003.00 2,008.25 2,010.00 2,011.75
de                           8.00 1,992.50 28.13 1,927.00 1,993.75 2,003.50 2,006.50
en                       1,958.00 2,004.30 12.54 1,936.00 2,000.00 2,008.00 2,013.00
es                           7.00 2,007.71  4.39 2,000.00 2,006.50 2,007.00 2,010.00

                                    budget                                          \
                              max    count          mean           std           min
original_language
cn                       2,010.00     3.00 14,872,795.67  4,479,793.25 11,715,578.00
da                       2,013.00     5.00 13,440,000.00 12,369,640.25  3,800,000.00
de                       2,013.00     8.00 18,223,718.75 30,623,544.47  1,530,000.00
en                       2,017.00 1,891.00 56,637,200.97 53,394,829.52          1.00
es                       2,014.00     5.00  7,500,000.00  8,046,738.47  1,500,000.00


                                   25%           50%           75%            max
original_language
cn                       12,309,193.50 12,902,809.00 16,451,404.50  20,000,000.00
da                        7,400,000.00 10,000,000.00 11,000,000.00  35,000,000.00
de                        4,100,000.00  6,250,000.00 15,084,937.50  92,620,000.00
en                       18,000,000.00 40,000,000.00 80,000,000.00 380,000,000.00
es                        2,000,000.00  2,000,000.00 13,000,000.00  19,000,000.00
```

1.12 Summary

We have seen several summary statistics:

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

Visualization:

- boxplot
- scatter plots with regression lines
- pairplot

Pandas:

- functions for computing statistics, `describe`
- `groupby`
- next week: more Pandas

More details in a statistics course.