Intro and descriptive statistics

The ASTA team

Contents

1 Software

1.1 Integrated development environments (IDEs)

- We will use Google Colab for running Python code and Jupyter Notebook files online (i.e. no local installation).
- If you prefer you can install a local program for running this, such as VS code.
- In any case, it is a good idea to make a folder on your computer where you want to keep files to use.

1.2 Basics

```
import numpy as np
```

- Ordinary calculations:

```
4.6 * (2 + 3)**4
```

```
## 2875.0
```

- Make a (scalar) object and print it:

```
a = 4
a
```

```
## 4
```

- Make a (vector) object and print it:

```
b = np.array([2, 5, 7])
b
```

```
## array([2, 5, 7])
```

- Make a sequence of numbers and print it:

```
s = np.arange(1, 5)
s
```

```
## array([1, 2, 3, 4])
```

- Note: A more flexible command for sequences:

```
s = np.arange(1, 5, 2)  # same idea
s
```

```
## array([1, 3])
```

- Elementwise calculations:

```
a * b
```

```
## array([ 8, 20, 28])
```

```
a + b
```

```
## array([ 6,  9, 11])
```

```
b ** 2
```

```
## array([ 4, 25, 49])
```

- Sum and product of elements:

```
np.sum(b)
```

```
## np.int64(14)
```

```
np.prod(b)
```

```
## np.int64(70)
```

2 Data

2.1 Data example

Data: Magazine Ads Readability

- Thirty magazines were ranked by educational level of their readers.
- Three magazines were **randomly** selected from each of the following groups:
  - Group 1: highest educational level
  - Group 2: medium educational level
  - Group 3: lowest educational level.
- Six advertisements were **randomly** selected from each of the following nine selected magazines:
  - Group 1: [1] Scientific American, [2] Fortune, [3] The New Yorker
  - Group 2: [4] Sports Illustrated, [5] Newsweek, [6] People
  - Group 3: [7] National Enquirer, [8] Grit, [9] True Confessions
- So, the data contains information about a total of 54 advertisements.

2.2 Data example (continued) - variables and format

- For each advertisement (54 cases), the data below were observed.
- **Variable names**:
  - WDS = number of words in advertisement
  - SEN = number of sentences in advertisement
  - 3SYL = number of 3+ syllable words in advertisement
  - MAG = magazine (1 through 9 as above)
  - GROUP = educational level (1 through 3 as above)
- Take a look at the data:

```
import pandas as pd

magAds = pd.read_csv("https://asta.math.aau.dk/datasets?file=magazineAds.txt", sep='\t')
magAds.head()
```

```
##    WDS  SEN  X3SYL  MAG  GROUP
## 0  205    9     34    1      1
## 1  203   20     21    1      1
## 2  229   18     37    1      1
## 3  208   16     31    1      1
## 4  146    9     10    1      1
```

- Variable names are in the top row. They are not allowed to start with a digit, so an `X` has been prefixed in `X3SYL`.

2.3 Data types

2.3.1 Quantitative variables

- The measurements have numerical values.
- Quantative data often comes about in one of the following ways:
  - **Continuous variables**: measurements of e.g. waiting times in a queue, revenue, share prices, etc.
  - **Discrete variables**: counts of e.g. words in a text, hits on a webpage, number of arrivals to a queue in one hour, etc.

- Measurements like this have a well-defined scale and in **Python** they are stored as the type **float**.
- It is important to be able to distinguish between discrete count variables and continuous variables, since this often determines how we describe the uncertainty of a measurement.

2.3.2 Categorical/qualitative variables

- The measurement is one of a set of given categories, e.g. sex (male/female), social status, satisfaction score (low/medium/high), etc.
- The measurement is usually stored (which is also recommended) as a **categorical** type in **Python**. The possible categories are called **categories**. Sometimes these are also refered to as factors with levels. Example: the categories of the categorical type "sex" is male/female.
- Categorical types have two so-called scales:
  - **Nominal scale**: There is no natural ordering of the categories, e.g. sex and hair color.
  - **Ordinal scale**: There is a natural ordering of the categories, e.g. social status and satisfaction score.

3 Population and sample

3.1 Aim of statistics

- Statistics is all about "saying something" about a population.
- Typically, this is done by taking a random sample from the population.
- The sample is then analysed and a statement about the population can be made.
- The process of making conclusions about a population from analysing a sample is called **statistical inference**.

3.2 Selecting randomly

- For the magazine data:
  - First we select **randomly** 3 magazines from each group.
  - Then we select **randomly** 6 ads from each magazine.
  - An important detail is that the selection is done completely at **random**, i.e.
    - each magazine within a group have an equal chance of being chosen and
    - each ad within a magazine have an equal chance of being chosen.
- In the following it is a fundamental requirement that the data collection respects this principle of randomness and in this case we use the term **sample**.
- More generally:
  - We have a **population** of objects.
  - We choose completely at random n of these objects, and from the jth object we get the measurement y_j, $j = 1, 2, \dots, n$.
  - The measurements $y_1, y_2, \dots, y_n$ are then called a **sample**.
- If we e.g. are measuring the water quality 4 times in a year then it is a bad idea to only collect data in fair weather. The chosen sampling time is not allowed to be influenced by something that might influence the measurement itself.

4 Variable grouping and frequency tables

4.1 Binning

- The function `cut` will divide the range of a numeric variable in a number of equally sized intervals, and record which interval each observation belongs to. E.g. for the variable `X3SYL` (the number of words with more than three syllables) in the magazine data:

```
# Before 'cutting':
magAds["X3SYL"].iloc[0:5]
```

```
## 0    34
## 1    21
## 2    37
## 3    31
## 4    10
## Name: X3SYL, dtype: int64
```

```
# After 'cutting' into 4 intervals:
syll = pd.cut(magAds["X3SYL"], bins=4)

# First 5 values
syll.iloc[0:5]
```

```
## 0      (32.25, 43.0]
## 1      (10.75, 21.5]
## 2      (32.25, 43.0]
## 3      (21.5, 32.25]
## 4    (-0.043, 10.75]
## Name: X3SYL, dtype: category
## Categories (4, interval[float64, right]): [(-0.043, 10.75] < (10.75, 21.5] < (21.5, 32.25] <
##                                             (32.25, 43.0]]
```

- The result is a `category` and the labels are the interval end points by default. Custom ones can be assigned through the `labels` argument:

```
labs = ["few", "some", "many", "lots"]
syll = pd.cut(magAds["X3SYL"], bins = 4, labels = labs)
syll.iloc[0:5]
```

```
## 0    lots
## 1    some
## 2    lots
## 3    many
## 4     few
## Name: X3SYL, dtype: category
## Categories (4, object): ['few' < 'some' < 'many' < 'lots']
```

```
magAds["syll"] = syll
magAds.head()
```

```
##    WDS  SEN  X3SYL  MAG  GROUP  syll
## 0  205    9     34    1      1  lots
## 1  203   20     21    1      1  some
## 2  229   18     37    1      1  lots
## 3  208   16     31    1      1  many
## 4  146    9     10    1      1   few
```

4.2 Tables

- To summarize the results we can use the function `value_counts()` from `pandas` package:

```
magAds["syll"].value_counts()
```

```
## syll
## few     26
## some    14
## many    10
## lots     4
## Name: count, dtype: int64
```

- In percent:

```
magAds["syll"].value_counts(normalize = True) * 100
```

```
## syll
## few     48.148148
## some    25.925926
## many    18.518519
## lots     7.407407
## Name: proportion, dtype: float64
```

4.3 2 factors: Cross tabulation

- To make a table of all combinations of two factors:

```
pd.crosstab(magAds["syll"], magAds["GROUP"])
```

```
## GROUP  1   2  3
## syll
## few    8  11  7
## some   4   2  8
## many   3   5  2
## lots   3   0  1
```

- Relative frequencies (in percent) columnwise:

```
magAds.groupby("GROUP")["syll"].value_counts(normalize=True).mul(100)
```

```
## GROUP  syll
## 1      few     44.444444
##        some    22.222222
##        many    16.666667
##        lots    16.666667
## 2      few     61.111111
##        many    27.777778
##        some    11.111111
##        lots     0.000000
## 3      some    44.444444
##        few     38.888889
##        many    11.111111
##        lots     5.555556
## Name: proportion, dtype: float64
```

- So, the above table shows e.g. how many percentage of the advertisements in group 1 that have 'few', 'some', 'many' or 'lots' words with more than 3 syllables.

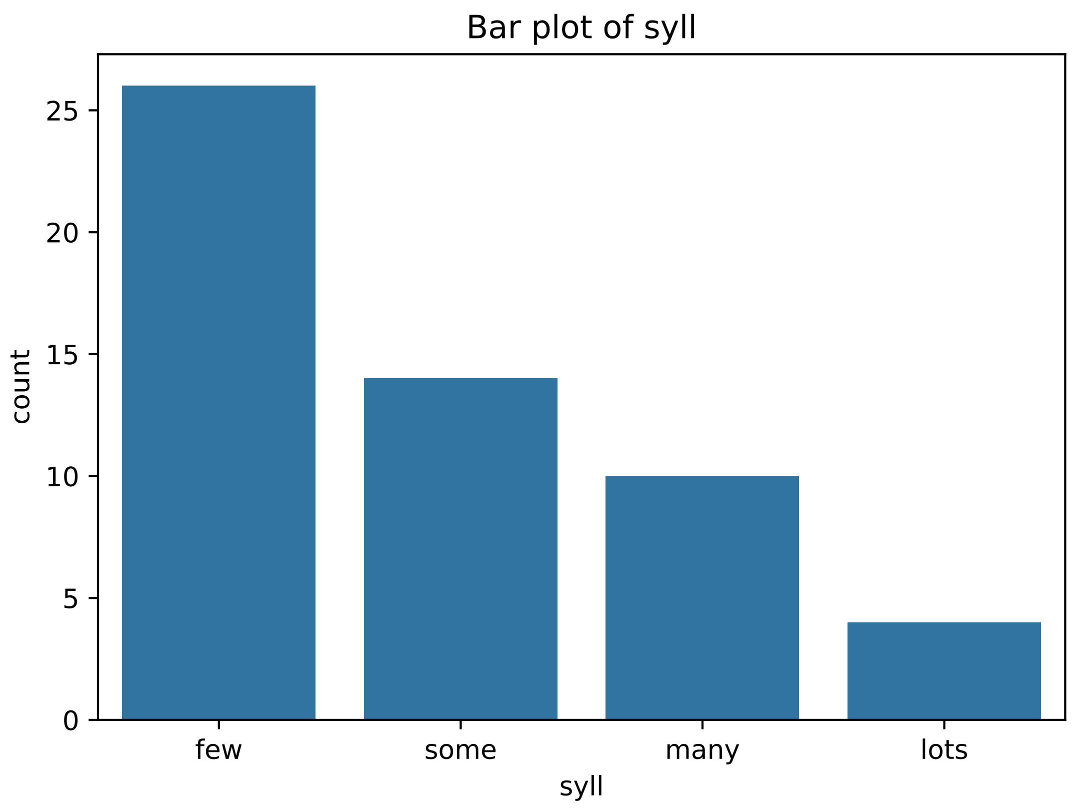
5 Graphics

5.1 Bar graph

- To create a bar graph plot of table data we use the function `countplot` from `seaborn`. For each level of the factor a box is drawn with the height proportional to the frequency (count) of the level.

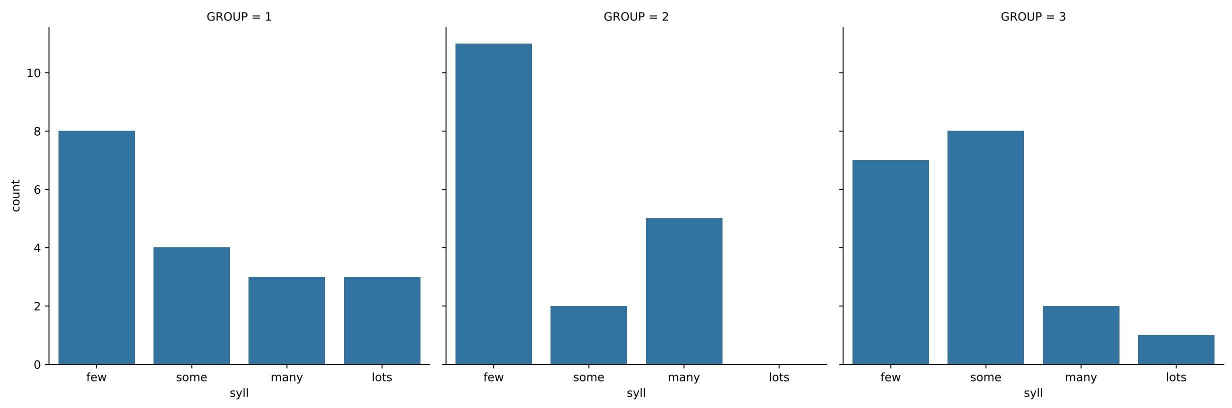
```python
import seaborn as sns
import matplotlib.pyplot as plt

g = sns.countplot(data = magAds, x = "syll")
g.set_title("Bar plot of syll")
```



- The bar graph can also be split by group:

```python
g = sns.catplot(data = magAds, x = "syll", kind = "count", col = "GROUP")
```



5.2 The Ericksen data

- Description of data: Ericksen 1980 U.S. Census Undercount.

- This data contains the following variables:
  - `minority`: Percentage black or Hispanic.
  - `crime`: Rate of serious crimes per 1000 individuals in the population.
  - `poverty`: Percentage poor.
  - `language`: Percentage having difficulty speaking or writing English.
  - `highschool`: Percentage aged 25 or older who had not finished highschool.
  - `housing`: Percentage of housing in small, multiunit buildings.
  - `city`: A factor with levels: `city` (major city) and `state` (state or state-remainder).
  - `conventional`: Percentage of households counted by conventional personal enumeration.
  - `undercount`: Preliminary estimate of percentage undercount.
- The Ericksen data has 66 rows/observations and 9 columns/variables.
- The observations are measured in 16 large cities, the remaining parts of the states in which these cities are located, and the other U.S. states.

```
Ericksen = pd.read_csv("https://asta.math.aau.dk/datasets?file=Ericksen.txt", sep='\t')
Ericksen.head()
```

```
##              name  minority  crime  poverty  language  highschool  housing  \
## 0         Alabama      26.1     49     18.9       0.2        43.5      7.6
## 1          Alaska       5.7     62     10.7       1.7        17.5     23.6
## 2         Arizona      18.9     81     13.2       3.2        27.6      8.1
## 3        Arkansas      16.9     38     19.0       0.2        44.5      7.0
## 4    California.R      24.3     73     10.4       5.0        26.0     11.8
##
##     city  conventional  undercount
## 0  state             0       -0.04
## 1  state           100        3.35
## 2  state            18        2.48
## 3  state             0       -0.74
## 4  state             4        3.60
```

```
pd.set_option('display.max_columns', None)  # Show all columns
Ericksen.head()
```

```
##              name  minority  crime  poverty  language  highschool  housing  \
## 0         Alabama      26.1     49     18.9       0.2        43.5      7.6
## 1          Alaska       5.7     62     10.7       1.7        17.5     23.6
## 2         Arizona      18.9     81     13.2       3.2        27.6      8.1
## 3        Arkansas      16.9     38     19.0       0.2        44.5      7.0
## 4    California.R      24.3     73     10.4       5.0        26.0     11.8
##
##     city  conventional  undercount
## 0  state             0       -0.04
## 1  state           100        3.35
## 2  state            18        2.48
## 3  state             0       -0.74
## 4  state             4        3.60
```

- Want to make a histogram for crime rate - how?

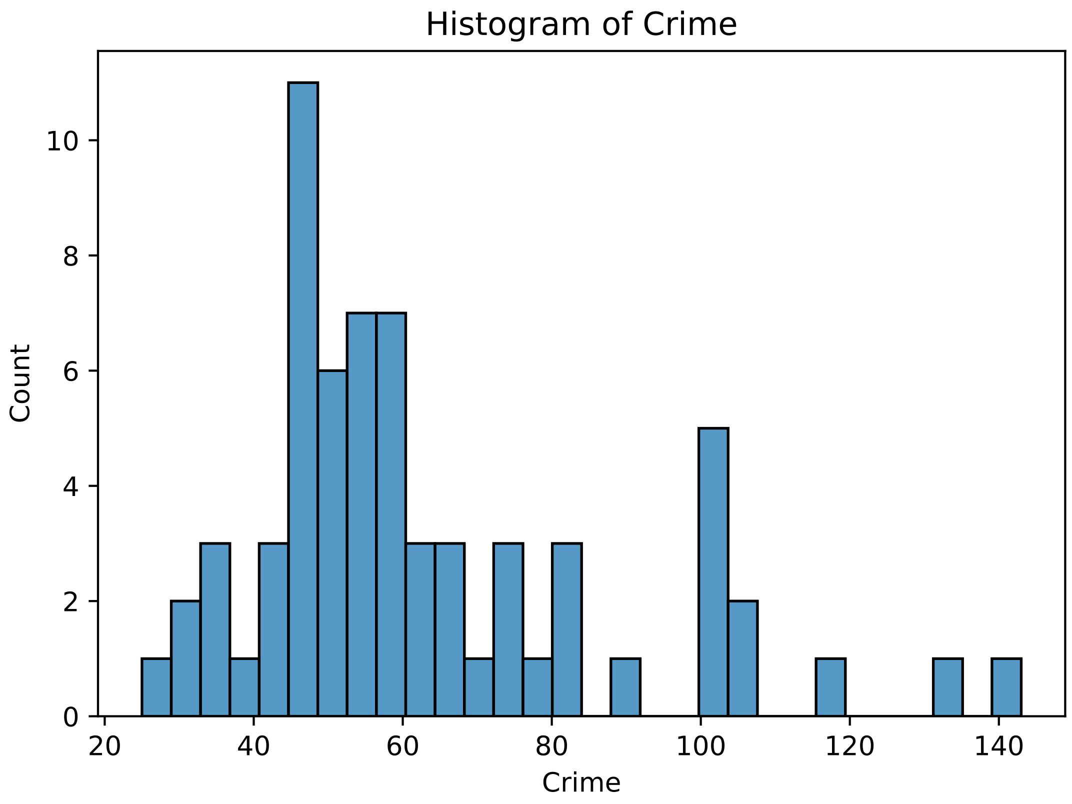
5.3 Histogram (quantitative variables)

- How to make a histogram for some variable `x`:
  - Divide the interval from the minimum value of `x` to the maximum value of `x` in an appropriate number of equal sized sub-intervals.
  - Draw a box over each sub-interval with the height being proportional to the number of observations

in the sub-interval.

- Histogram of crime rates for the Ericksen data

```
sns.histplot(data = Ericksen, x = "crime", bins = 30, edgecolor = "black")
plt.xlabel("Crime")
plt.ylabel("Count")
plt.title("Histogram of Crime")
plt.show()
```



6 Summary of quantitative variables

6.1 Measures of center of data: Mean and median

- We return to the magazine ads example (`WDS` = number of words in advertisement). A number of numerical summaries for `WDS` can be retrieved using the `favstats` function:

```
col = magAds["WDS"]
summary = {
    "min": col.min(),
    "Q1": col.quantile(0.25),
    "median": col.median(),
    "mean": col.mean(),
    "Q3": col.quantile(0.75),
    "max": col.max(),
    "sd": col.std(),
    "n": col.count(),
    "missing": col.isna().sum()
}
pd.DataFrame([summary])
```

```
##    min    Q1  median       mean     Q3  max         sd   n  missing
## 0   31  69.0    95.5  122.62963  201.5  230  65.877043  54        0
```

```
#or: magAds["WDS"].describe()
```

- The observed values of the variable `WDS` are $y_1 = 205$, $y_2 = 203, \ldots, y_n = 208$, where there are a total of $n = 54$ values. As previously defined this constitutes a **sample**.
- **mean** $= 122.63$ is the **average** of the sample, which is calculated by

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

  We may also call $\bar{y}$ the **(empirical) mean** or the **sample mean**.
- **median** $= 95.5$ is the 50-percentile, i.e. the value that splits the sample in 2 groups of equal size.
- An important property of the **mean** and the **median** is that they have the same unit as the observations (e.g. meter).

6.2 Measures of variability of data: range, standard deviation and variance

- The **range** is the difference of the largest and smallest observation.
- The **(empirical) variance** is the average of the squared deviations from the mean:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2.$$

- **sd** = **standard deviation** $= s = \sqrt{s^2}$.
- Note: If the observations are measured in meter, the **variance** has unit meter^2 which is hard to interpret. The **standard deviation** on the other hand has the same unit as the observations.
- The standard deviation describes how much data varies around the (empirical) mean.

6.3 Calculation of mean, median and standard deviation

The mean, median and standard deviation are just some of the summaries that can be read of the output (shown on previous page). They may also be calculated separately in the following way:

- Mean of `WDS`:

```
magAds["WDS"].mean()
```

```
## np.float64(122.62962962962963)
```

- Median of `WDS`:

```
magAds["WDS"].median()
```

```
## np.float64(95.5)
```

- Standard deviation for `WDS`:

```
magAds["WDS"].std()
```

```
## np.float64(65.87704278349153)
```

We may also calculate the summaries for each group (variable `GROUP`), e.g. for the mean:

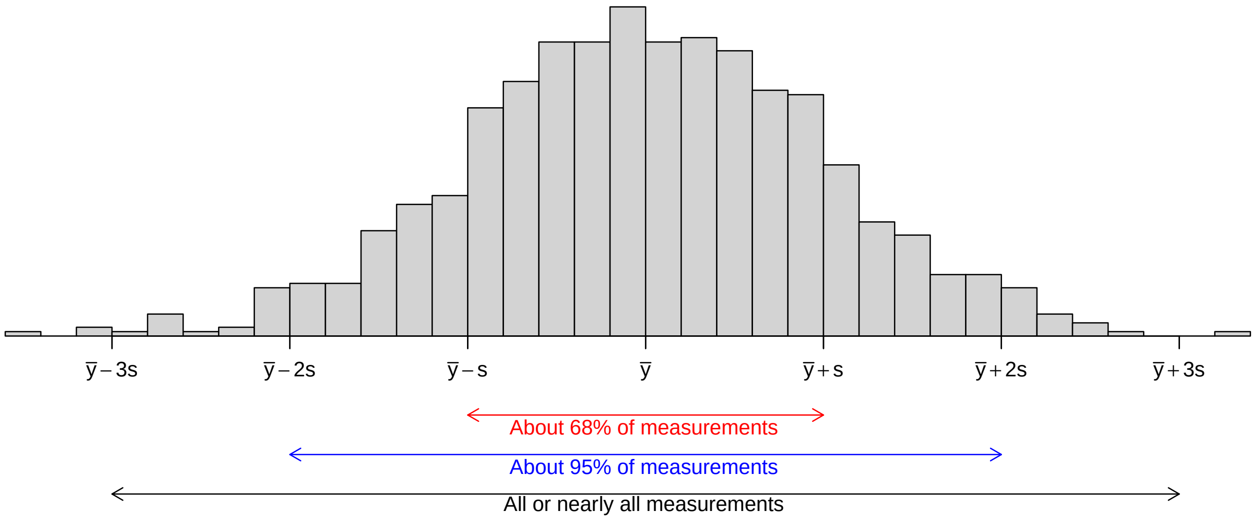
```
magAds.groupby("GROUP")["WDS"].mean().reset_index() # reset_index() resets the grouping again
```

```
##    GROUP         WDS
## 0      1  140.000000
## 1      2  121.388889
## 2      3  106.500000
```

6.4 A word about terminology

- **Standard deviation**: a measure of variability of a population or a sample.
- **Standard error**: a measure of variability of an estimate. For example, a measure of variability of the sample mean.

6.5 The empirical rule



If the histogram of the sample looks like a bell shaped curve, then

- about 68% of the observations lie between $\bar{y} - s$ and $\bar{y} + s$.
- about 95% of the observations lie between $\bar{y} - 2s$ and $\bar{y} + 2s$.
- All or almost all (99.7%) of the observations lie between $\bar{y} - 3s$ and $\bar{y} + 3s$.

6.6 Percentiles

- **The *p*th percentile** is a value such that about p% of the population (or sample) lies below or at this value and about $(100 - p)$% of the population (or sample) lies above it.

6.6.1 Percentile calculation for a sample:

- First, sort data in increasing order. For the `WDS` variable in the magazine data:

$$y_{(1)} = 31, y_{(2)} = 32, y_{(3)} = 34, \ldots, y_{(n)} = 230.$$

  Here the number of observations is $n = 54$.

- Find the 5th percentile (i. e. $p = 5$):
  - The observation number corresponding to the 5-percentile is $N = \frac{54 \cdot 5}{100} = 2.7$.
  - So the 5-percentile lies between the observations with observation number $k = 2$ and $k + 1 = 3$. That is, its value lies somewhere in the interval between $y_2 = 32$ and $y_3 = 34$

- One of several methods for estimating the 5-percentile from the value of N is defined as:

$$y_{(k)} + (N-k)(y_{(k+1)} - y_{(k)})$$

  which in this case states

$$y_2 + (2.7-2)(y_3 - y_2) = 32 + 0.7 \cdot (34-32) = 33.4$$

6.7 Median, quartiles and interquartile range

Recall

```
col = magAds["WDS"]
summary = {
    "min": col.min(),
    "Q1": col.quantile(0.25),
    "median": col.median(),
    "Q3": col.quantile(0.75),
    "max": col.max(),
    "n": col.count(),
    "missing": col.isna().sum()
}
pd.DataFrame([summary])
```

```
##    min    Q1  median     Q3  max   n  missing
## 0   31  69.0    95.5  201.5  230  54        0
```

```
#or: magAds["WDS"].describe()
```

- 50-percentile = 96 is the **median** and it is a measure of the center of data.
- 0-percentile = 31 is the **minimum** value.
- 25-percentile = 69 is called the **lower quartile** (Q1). Median of lower 50% of data.
- 75-percentile = 202 is called the **upper quartile** (Q3). Median of upper 50% of data.
- 100-percentile = 230 is the **maximum** value.
- **Interquartile Range (IQR)**: a measure of variability given by the difference of the upper and lower quartiles: 202 - 69 = 133.

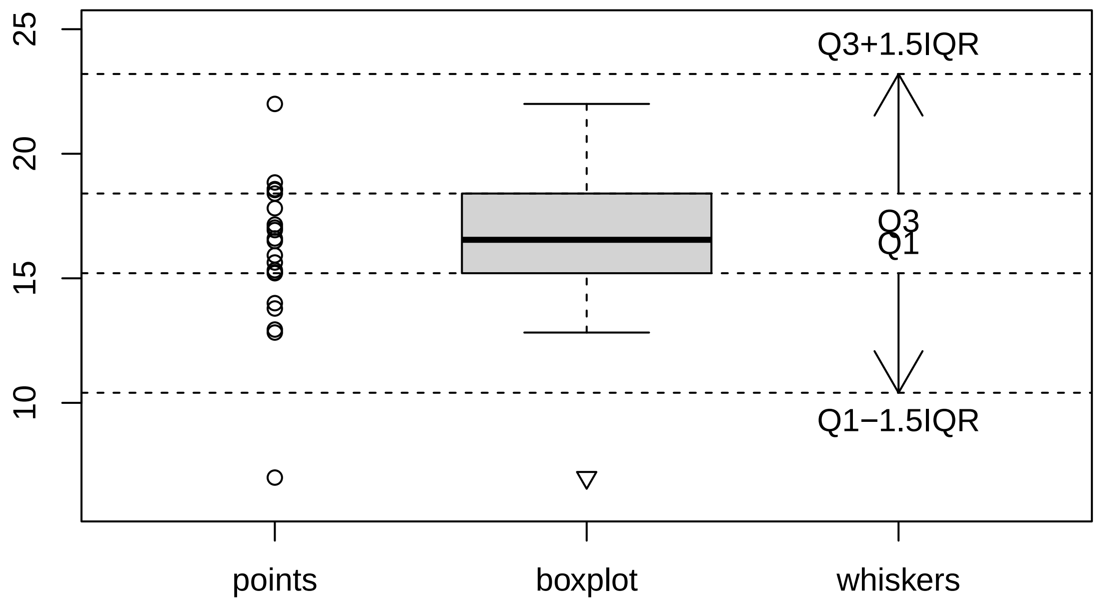
7 More graphics

7.1 Box-and-whiskers plots (or simply box plots)

How to draw a box-and-whiskers plot:

- Box:
  - Calculate the median, lower and upper quartiles.
  - Plot a line by the median and draw a box between the upper and lower quartiles.
- Whiskers:
  - Calculate interquartile range and call it IQR.
  - Calculate the following values:
    - L = lower quartile - 1.5*IQR
    - U = upper quartile + 1.5*IQR
  - Draw a line from lower quartile to the smallest measurement, which is larger than L.
  - Similarly, draw a line from upper quartile to the largest measurement which is smaller than U.
- Outliers: Measurements smaller than L or larger than U are drawn as circles.

Note: Whiskers are minimum and maximum of the observations that are not deemed to be outliers.



7.1.1 Boxplot for Ericksen data

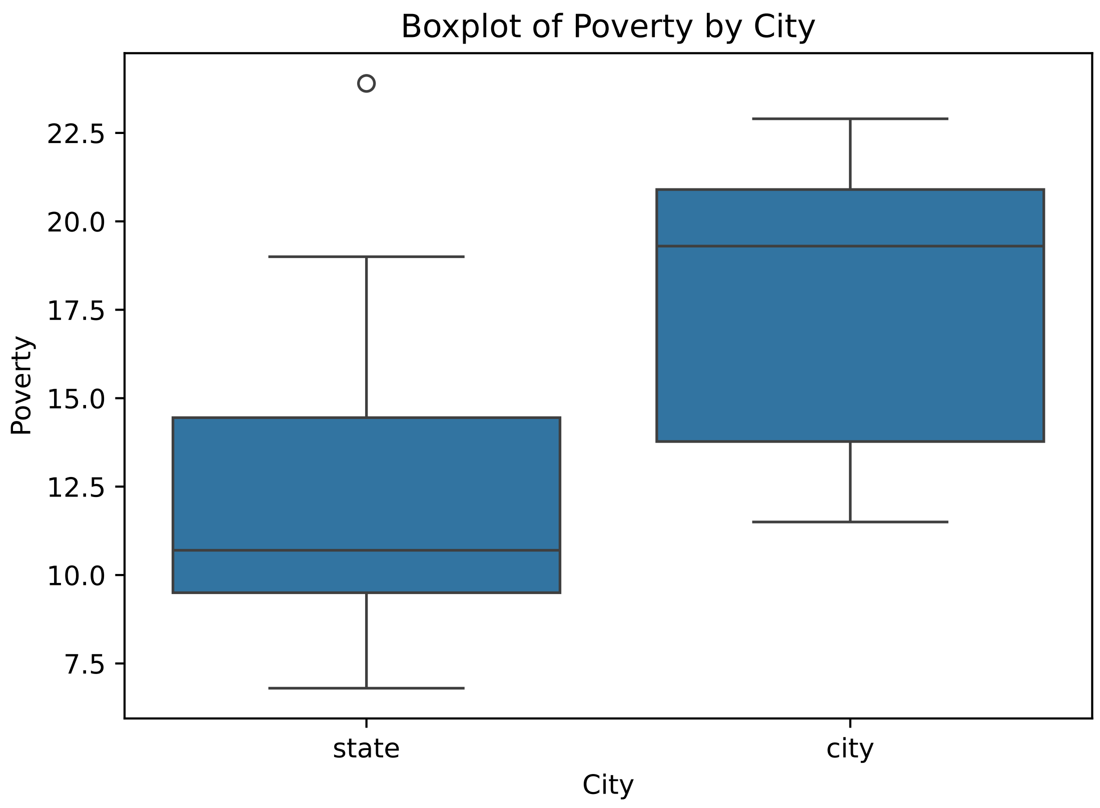
Boxplot of the poverty rates separately for cities and states (variable `city`):

```
summary_by_city = Ericksen.groupby("city")["poverty"].agg(
    ['min',
     lambda x: x.quantile(0.25),  # Q1
     'median',
     'mean',
     lambda x: x.quantile(0.75),  # Q3
     'max',
     'std',
     'count',
     lambda x: x.isna().sum()      # missing values
])

# Rename columns for clarity
summary_by_city.columns = ['min', 'Q1', 'median', 'mean', 'Q3', 'max', 'sd', 'n', 'missing']
summary_by_city
```

```
##          min     Q1  median      mean    Q3   max        sd   n  missing
## city
## city    11.5  13.775    19.3  17.69375  20.90  22.9  4.041859  16        0
## state    6.8   9.500    10.7  12.11600  14.45  23.9  3.733596  50        0
```

```
sns.boxplot(data=Ericksen, x="city", y="poverty")
plt.xlabel("City")
plt.ylabel("Poverty")
plt.title("Boxplot of Poverty by City")
plt.show()
```

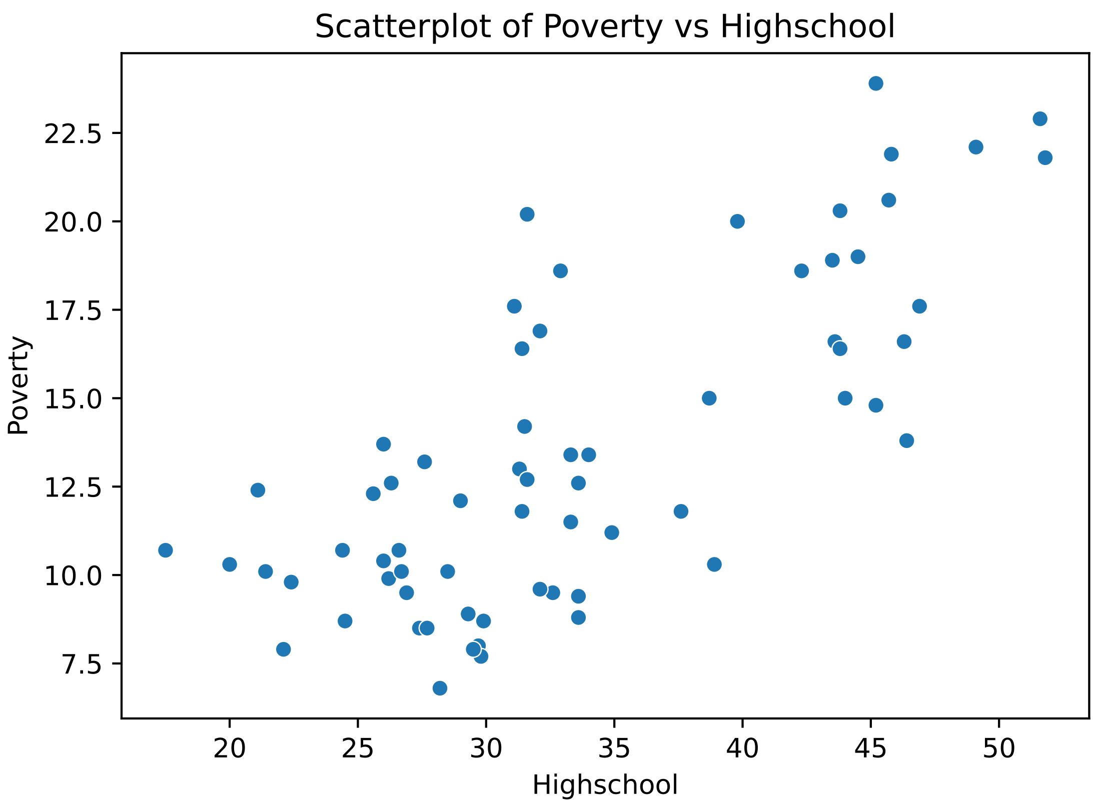


- There seems to be more poverty in the cities.
- A single state differs noticeably from the others with a high poverty rate.

7.2 2 quantitative variables: Scatter plot

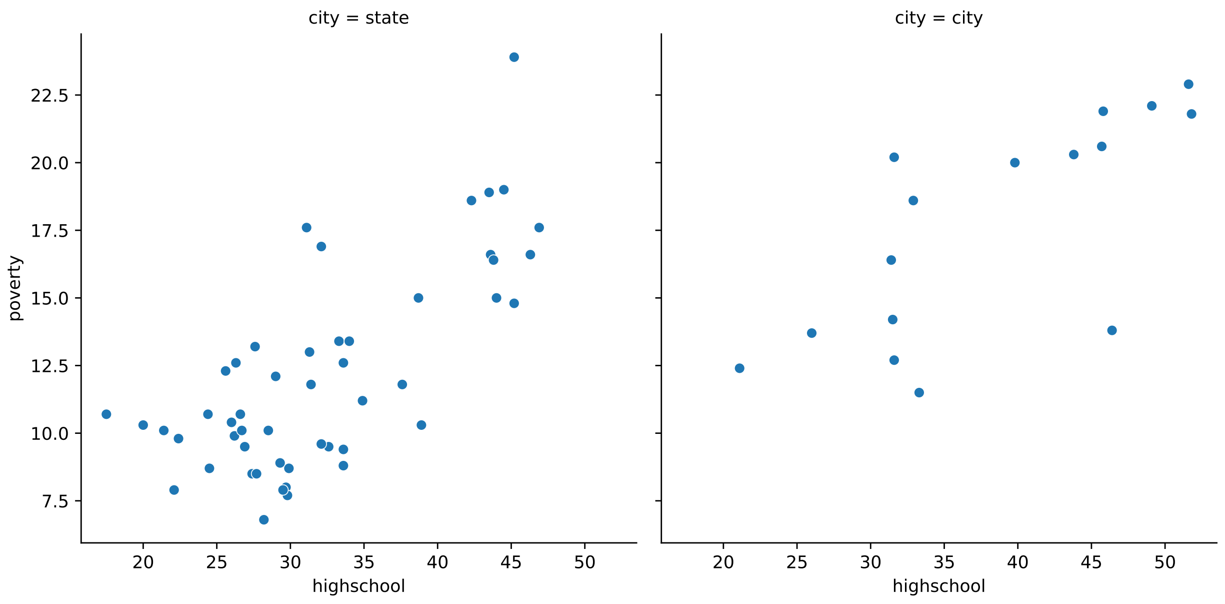
For two quantitative variables the usual graphic is a scatter plot:

```
sns.scatterplot(data=Ericksen, x="highschool", y="poverty")
plt.xlabel("Highschool")
plt.ylabel("Poverty")
plt.title("Scatterplot of Poverty vs Highschool")
plt.show()
```

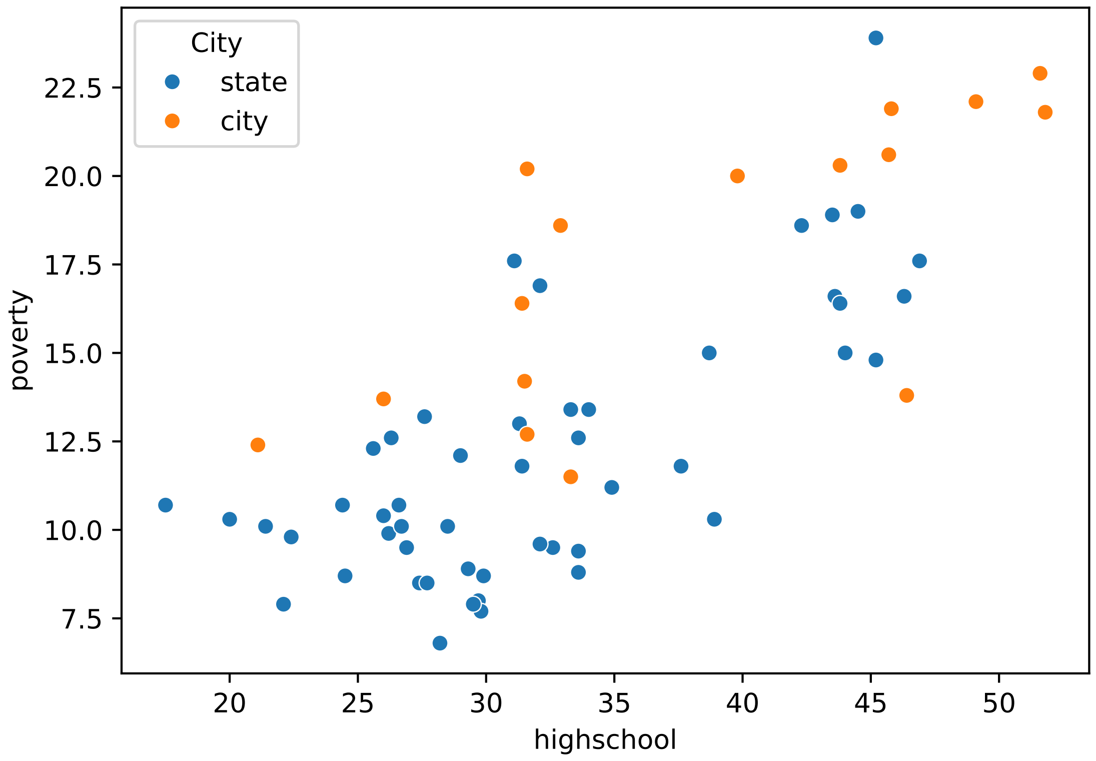


This can be either split or coloured according to the value of `city`:

```
g = sns.relplot(data=Ericksen, x="highschool", y="poverty", col="city", kind="scatter")
```

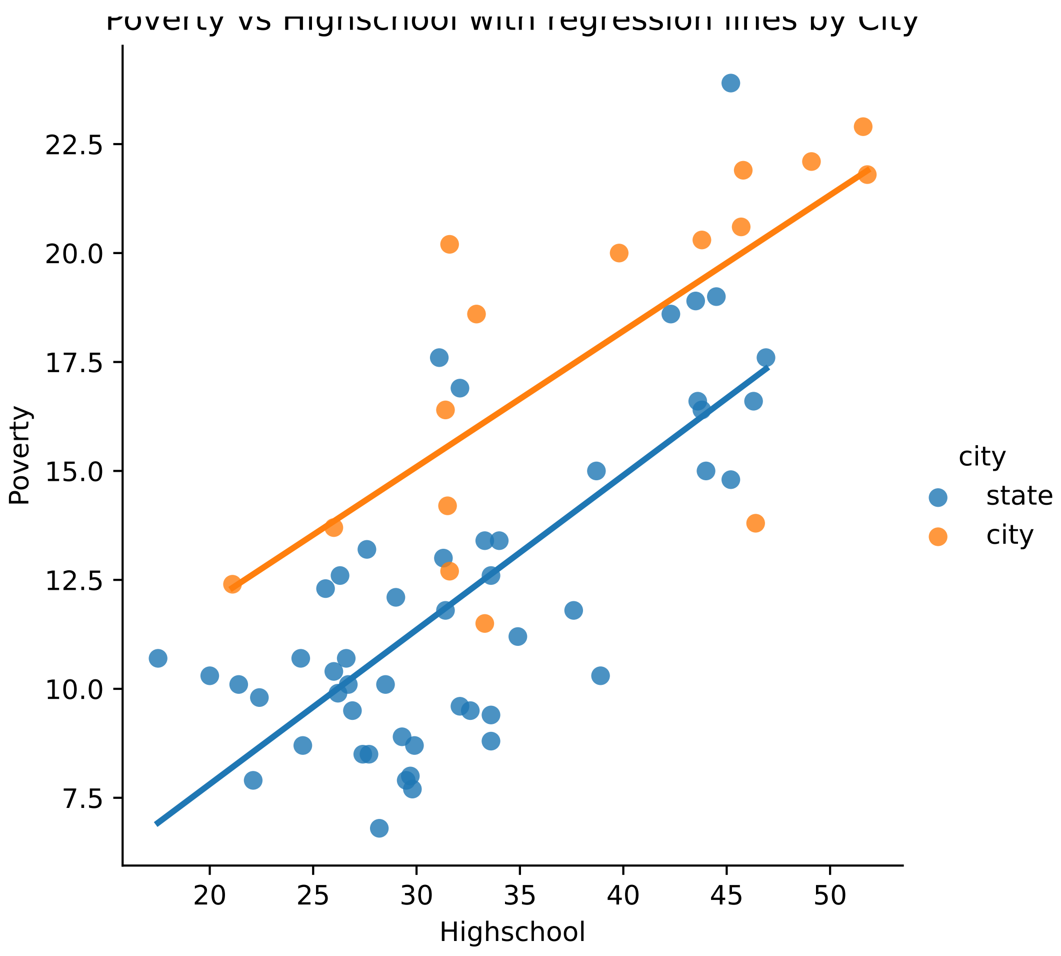


```
g = sns.scatterplot(data=Ericksen, x="highschool", y="poverty", hue="city")
plt.legend(title="City")
```



If we want a regression line along with the points we can do:

```
g = sns.lmplot(data=Ericksen, x="highschool", y="poverty", hue="city", ci=None)
plt.xlabel("Highschool")
plt.ylabel("Poverty")
plt.title("Poverty vs Highschool with regression lines by City")
```



8 Appendix

8.1 Recoding variables

- `astype` can be used to convert a vector of numerical values to be a categorical variable. E.g.:

```
magAds["GROUP"].head()
```

```
## 0    1
## 1    1
## 2    1
## 3    1
## 4    1
## Name: GROUP, dtype: int64
```

```
f = magAds["GROUP"].astype("category")
magAds["GROUP"] = f
magAds["GROUP"].head()
```

```
## 0    1
## 1    1
## 2    1
## 3    1
## 4    1
## Name: GROUP, dtype: category
## Categories (3, int64): [1, 2, 3]
```

- Custom names for the categories can also be used:

```
f = magAds["GROUP"].astype("category")
f.value_counts()
```

```
## GROUP
## 1    18
## 2    18
## 3    18
## Name: count, dtype: int64
```

```
f = f.cat.rename_categories({
    1: 'high',
    2: 'medium',
    3: 'low'
})
magAds['GROUP'] = f
magAds["GROUP"].head()
```

```
## 0    high
## 1    high
## 2    high
## 3    high
## 4    high
## Name: GROUP, dtype: category
## Categories (3, object): ['high', 'medium', 'low']
```

```
magAds['GROUP'].value_counts()
```

```
## GROUP
## high      18
## medium    18
## low       18
## Name: count, dtype: int64
```

- In this way the numbers are replaced by more informative labels describing the educational level.