SLU04 - Basic Stats with Pandas

Nov 26, 2023

Motivation

- How to **make sense** of thousands or millions of data points in your data set?
- How to inspect the **distribution** of the data?
- How to deal with **outliers**?

Overview

Objective: learn how to use pandas to obtain simple statistics of datasets.

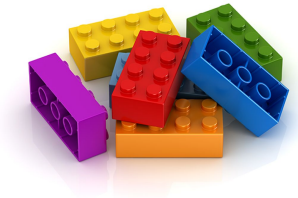
We will cover:

- Minimum, maximum, argmin, argmax
- Mode
- Mean & median
- Standard deviation
- Skewness & Kurtosis
- Quantiles
- Outliers & how to deal with them

2. Topic Explanation

Descriptive Statistics → Inspecting the distribution of the data → Outlier detection

Dataset: legos



```
In [2]: lego = pd.read_csv('data/sets.csv')
```

```
In [3]: lego.head()  # let's have a look at the first lines of the dataframe
```

Out[3]:

	set_num	name	year	theme_id	num_parts
0	00-1	Weetabix Castle	1970	414	471
1	0011-2	Town Mini-Figures	1978	84	12
2	0011-3	Castle 2 for 1 Bonus Offer	1987	199	2
3	0012-1	Space Mini-Figures	1979	143	12
4	0013-1	Space Mini-Figures	1979	143	12

set_num	Unique set ID.
name	The name of the set.
year	Year the set was published.
theme_id	Unique ID for the theme used for the set (from `themes.csv`).
num_parts	The number of parts included in the set.

Maximum & minimum

- ***What are the maximum and minimum values for the number of parts?***
- ***Which sets do these values correspond to?***

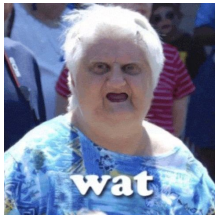
```
In [6]: lego.num_parts.max()
Out[6]: 5922

In [7]: lego.num_parts.idxmax()
Out[7]: 170
```

`idxmax()` gives the ***first index*** of the maximum value

```
In [10]: lego.num_parts.idxmin()
Out[10]: 1683

In [11]: lego.num_parts.min()
Out[11]: -1
```



Maximum & minimum

Basic Statistics can help us finding data problems.

- ***Are there** more than one **set with -1 parts?***

```
In [12]: lego.loc[lego.num_parts == lego.num_parts.min()]
```

Out[12]:

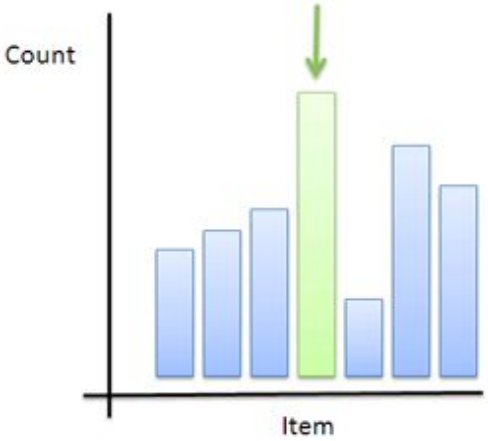
	set_num	name	year	theme_id	num_parts
1683	240-1	Wooden Storage Box Large, Empty	1967	383	-1
6545	66392-1	Duplo Cars Super Pack 3 in 1 (5816, 5817, 5818)	2012	506	-1
11645	Vancouver-1	LEGO Store Grand Opening Exclusive Set, Oakrid...	2012	408	-1

Mode

- ***What is the year that had more sets published?***

```
In [14]: lego.year.mode()

Out[14]: 0    2014
         dtype: int64
```



The mode is the most frequently appearing value in a population or sample.

Mean

- ***What is the average number of parts in the sets of legos?***

```
In [16]: lego.num_parts.mean()
Out[16]: 162.3043701799486
```

The mean is the sum all the all of the observations and dividing by the number of observations.

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

Median

First, arrange the observations in an ascending order.

If the number of observations (n) is odd: the median is the value at position

$$\left(\frac{n+1}{2}\right)$$

If the number of observations (n) is even:

1. Find the value at position $\left(\frac{n}{2}\right)$
2. Find the value at position $\left(\frac{n+1}{2}\right)$
3. Find the average of the two values to get the median.

Median

In [17]:
```
pd.Series([1,
           2,
           3, # <--- ok, this one is easy
           4,
           5]
          ).median()
```

Out[17]: 3.0

In [18]:
```
pd.Series([1,
           2, # <--- both 2 and 3 are in "the middle", as there are only 4
           3, # <--- so the median will split the difference!
           4,
          ]
          ).median()
```

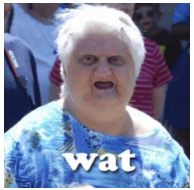
Out[18]: 2.5

Mean vs Median

- ***What is the average number of parts in the sets of legos?***

```
In [16]: lego.num_parts.mean()
Out[16]: 162.3043701799486
```

```
In [19]: lego.num_parts.median()
Out[19]: 45.0
```



Mean vs Median

In [22]:
```
s = pd.read_csv('data/student_income.csv', header=None)[0]
```

In [23]:
```
print('The mean income of students is   %0.1f' % s.mean())
print('The median income of students is %0.1f' % s.median())
```

```
The mean income of students is   600.0
The median income of students is 625.0
```

In [24]:
```
rich_mc_money = pd.Series([10000])  # <--- Wooooow!
s = s.append(rich_mc_money)
```

In [25]:
```
print('The mean income of students is   %0.1f' % s.mean())
print('The median income of students is %0.1f' % s.median())
```

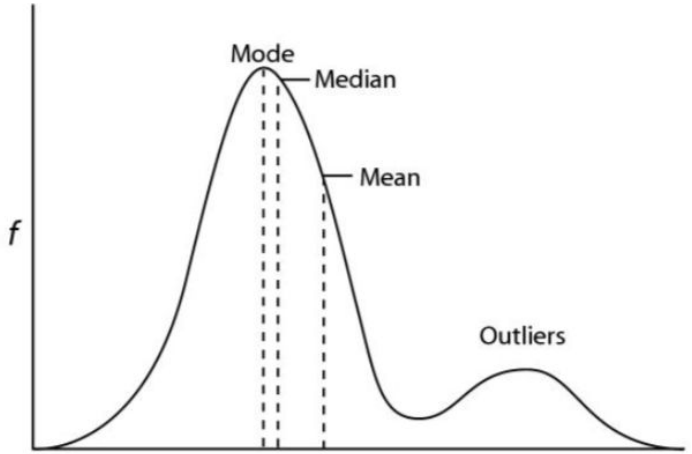
```
The mean income of students is   903.2
The median income of students is 630.0
```


BREAKING NEWS
AVERAGE STUDENT BUDGET UP 300$
10:58
"THIS IS THE RESULT OF OUR GOOD GOVERNMENT", SAYS USELESS POLITICIAN

Mean vs Median

- The median may be a better indicator of the most typical value if a set of scores has an outlier.

- With skewed distributions the median is not as strongly influenced by the skewed values when compared with the mean.

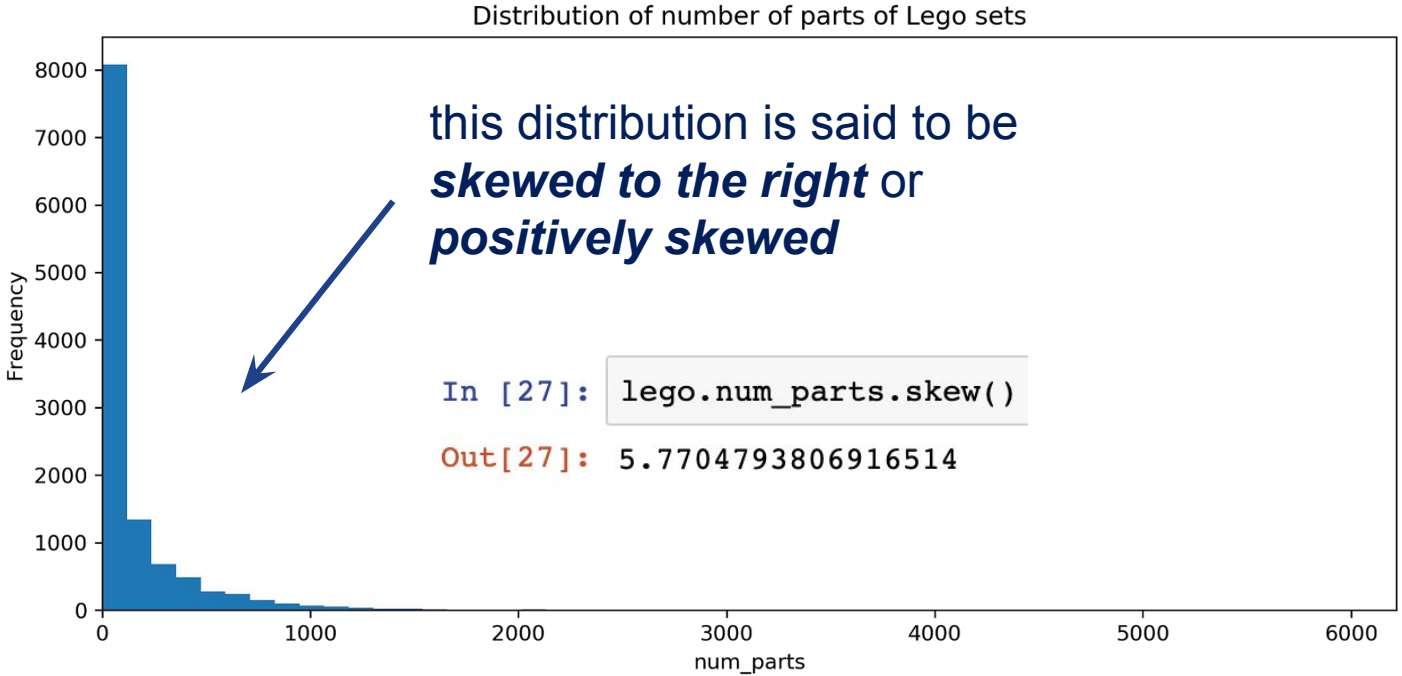


Descriptive Statistics

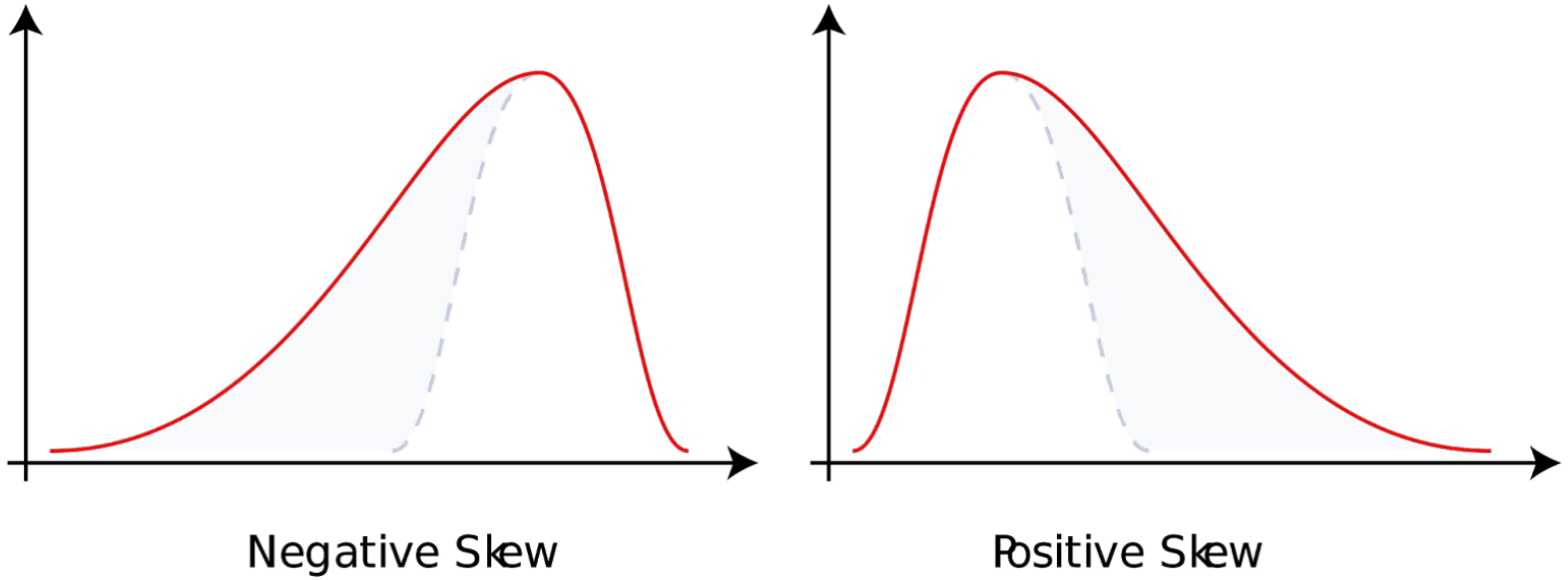
Inspecting the distribution of the data

Outlier detection

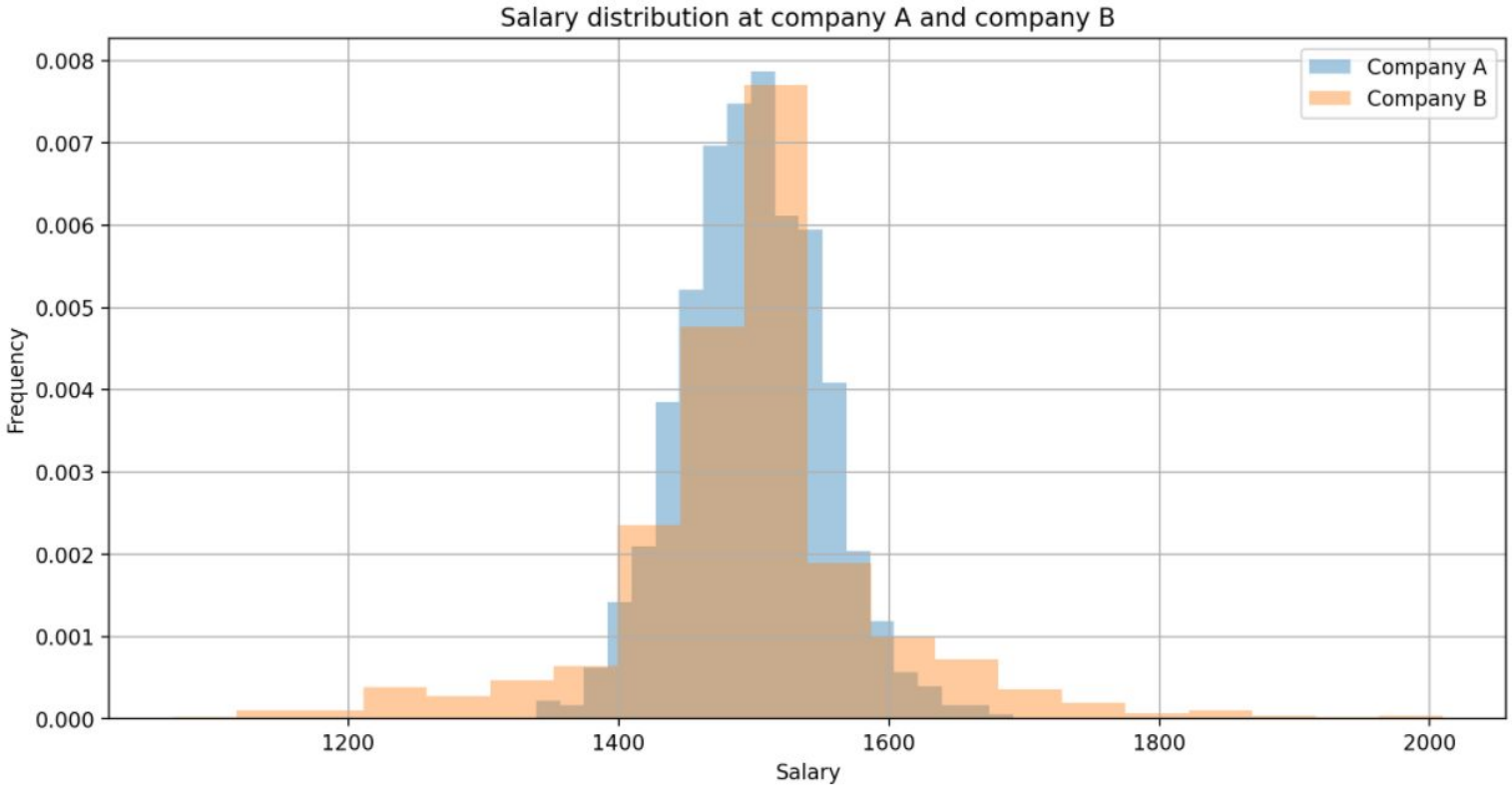
Skewness


Distribution of number of parts of Lego sets
Frequency
num_parts
0
1000
2000
3000
4000
5000
6000
7000
8000
this distribution is said to be
skewed to the right or
positively skewed
In [27]: lego.num_parts.skew()
Out[27]: 5.7704793806916514

Skewness



Standard deviation



Standard deviation

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \bar{X}\right)^2}$$

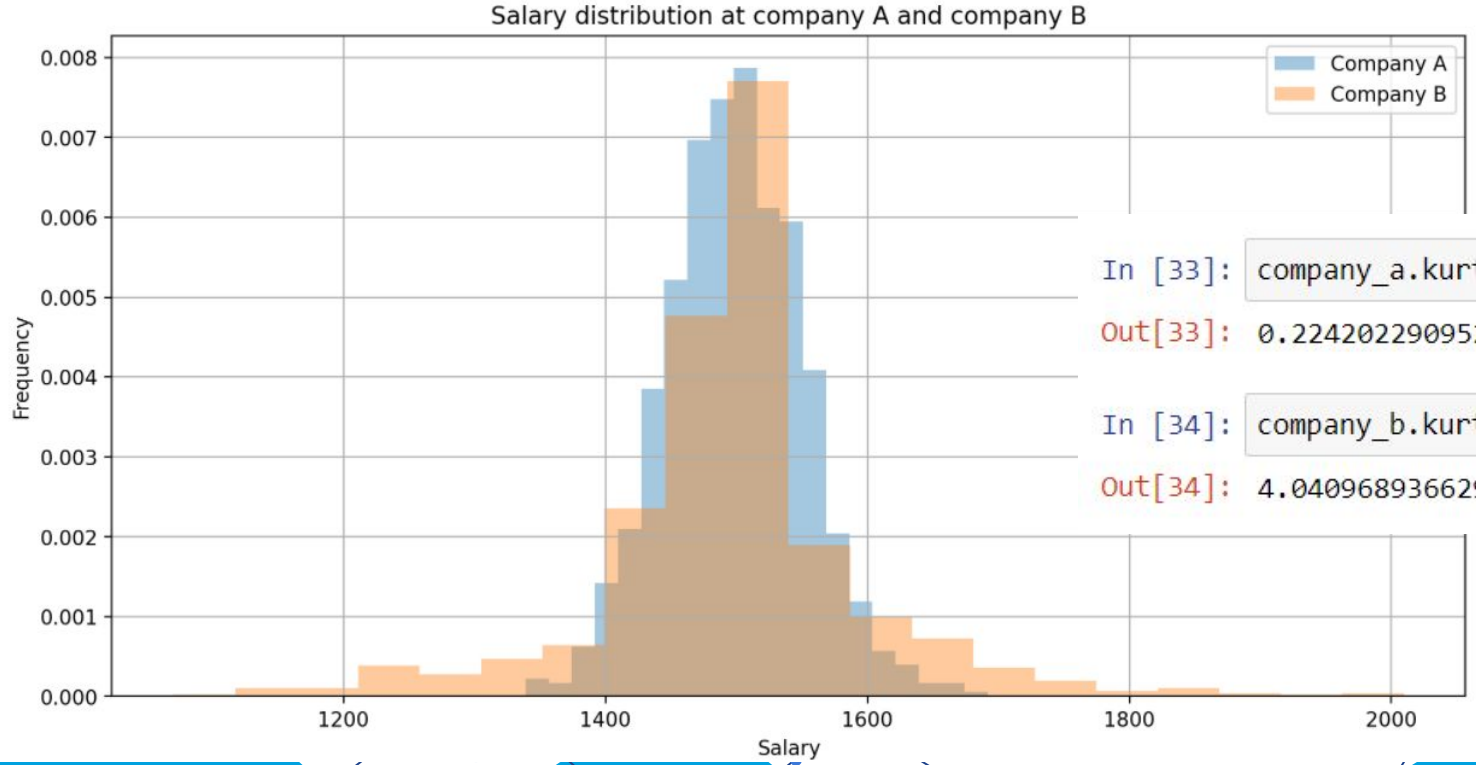
mean


```
print('Company A has a mean of %0.1f and a standard deviation of %0.1f' % (
    company_a.mean(), company_a.std()))

print('Company B has a mean of %0.1f and a standard deviation of %0.1f' % (
    company_b.mean(), company_b.std()))
```

```
Company A has a mean of 1498.7 and a standard deviation of 52.3
Company B has a mean of 1496.8 and a standard deviation of 101.3
```

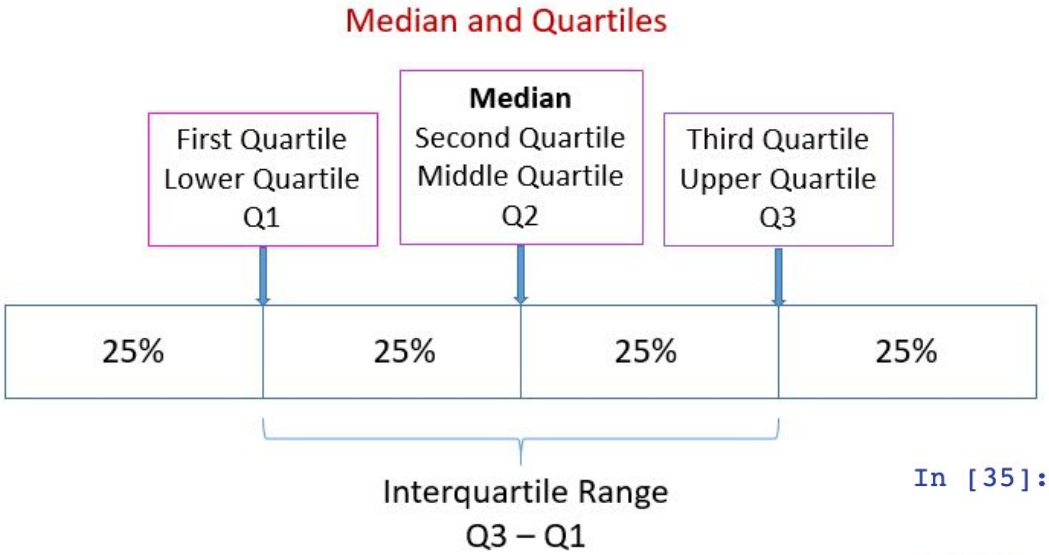
Kurtosis




```
In [33]: company_a.kurt()
Out[33]: 0.22420229095281652

In [34]: company_b.kurt()
Out[34]: 4.0409689366297314
```

Quantiles




```
In [35]: quartiles = [.25, .5, .75]
         lego.num_parts.quantile(q=quartiles)

Out[35]: 0.25     10.0
         0.50     45.0
         0.75    172.0
         Name: num_parts, dtype: float64
```

Descriptive Statistics

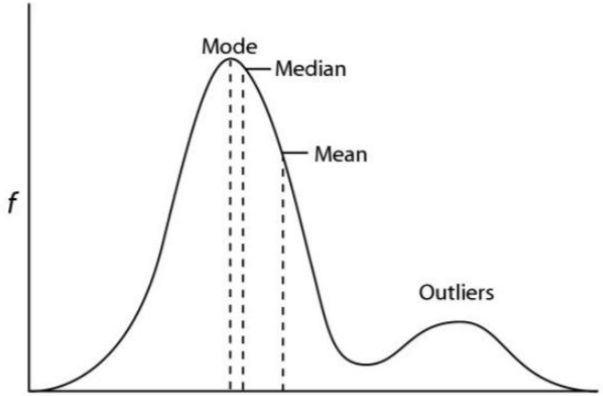
Inspecting the distribution of the data

Outlier detection

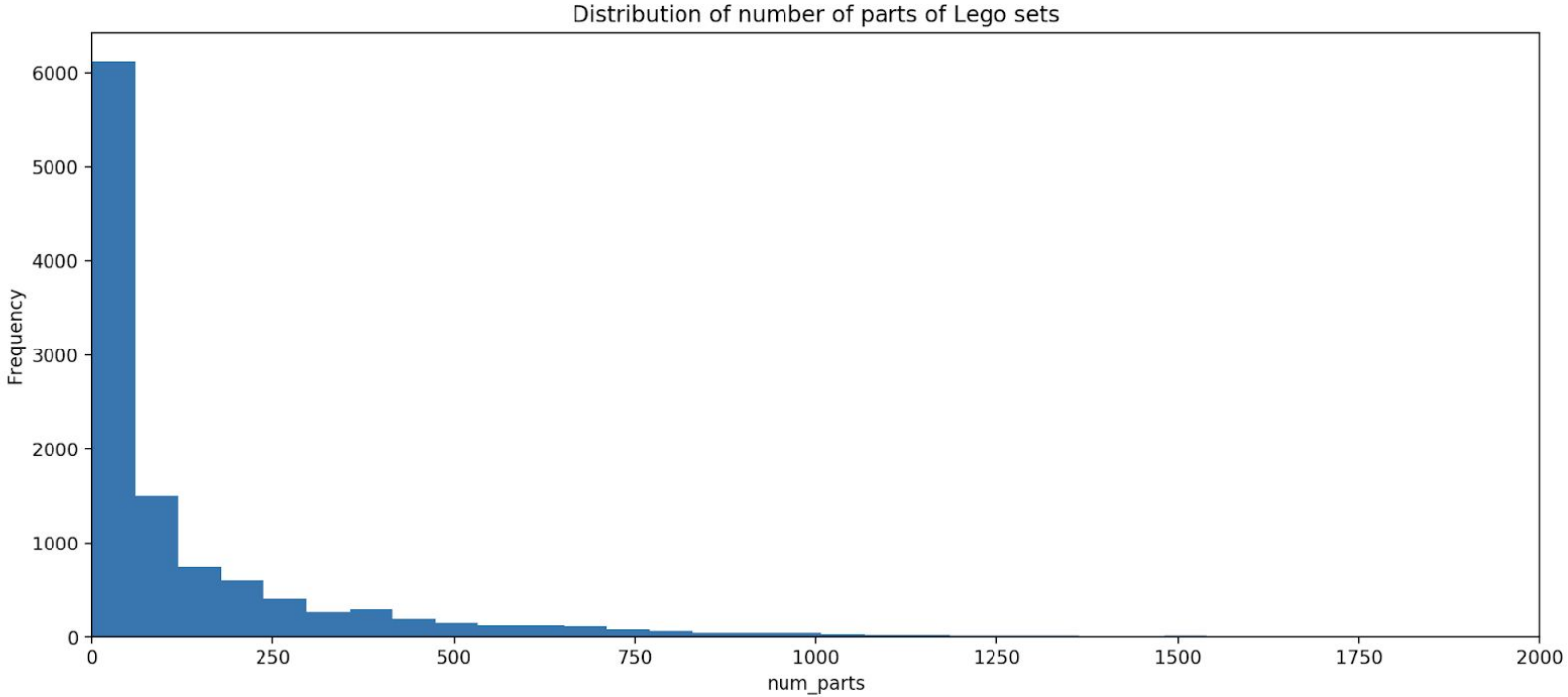
Dealing with outliers

A few strategies to deal with outliers are the following:

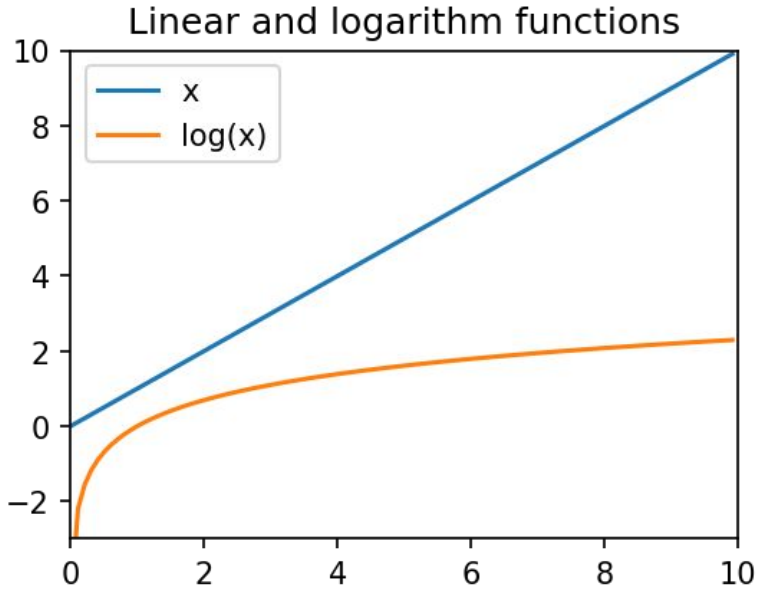
- do nothing;
- drop instances with outliers;
- drop column with outliers;
- impute values;
- binning;
- transforming data: e.g. log transformation.



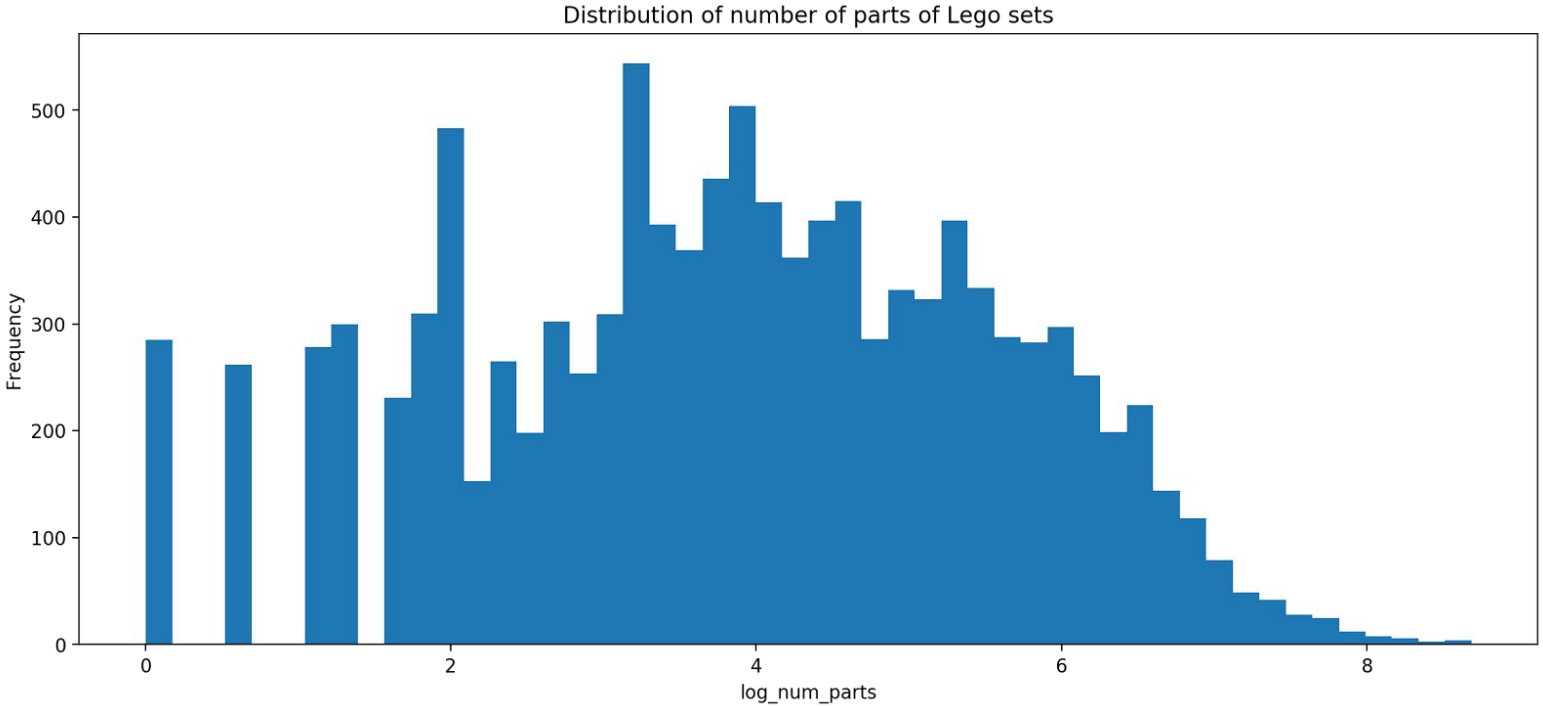
log transformation



log transformation



log transformation



3. Recap

Recap

Pandas methods covered:

```
.max()
.idxmax()
.min()
.idxmin()
.mode()
.mean()
.median()
.var()
.std()
.skew()
.kurt()
.quantile()
```

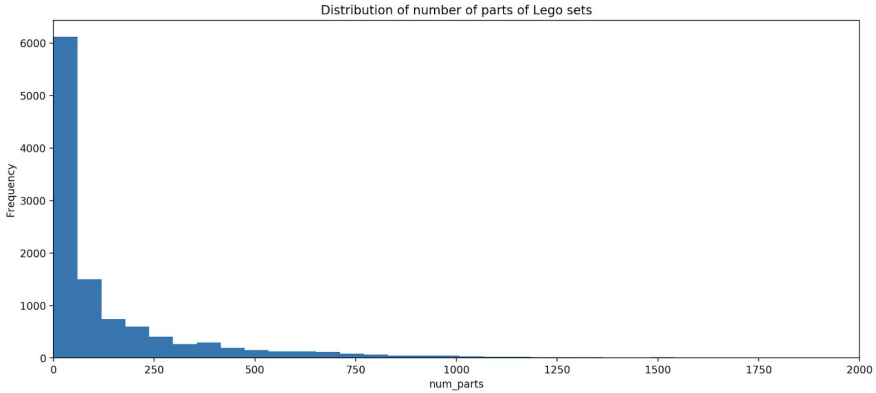
Useful method to summarize the data:

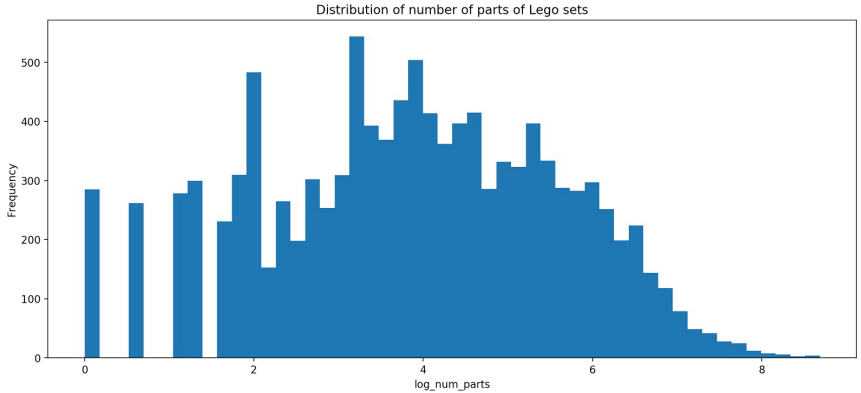
```
In [37]: lego.num_parts.describe()

Out[37]: count    11670.000000
         mean       162.304370
         std        330.224172
         min          0.000000
         25%         10.000000
         50%         45.000000
         75%        172.000000
         max       5922.000000
         Name: num_parts, dtype: float64
```

Recap

How to deal with outliers:





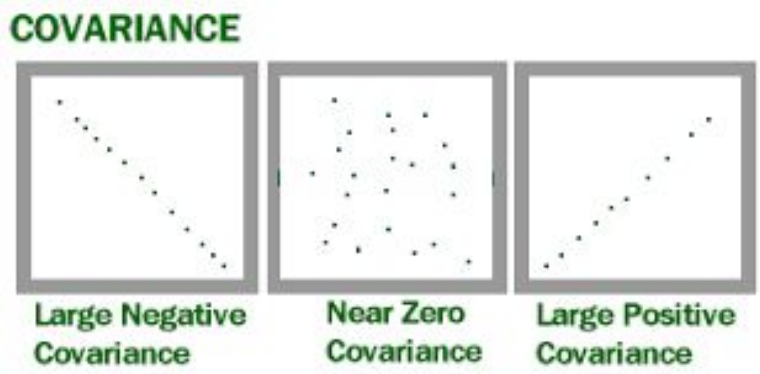
Log transformation

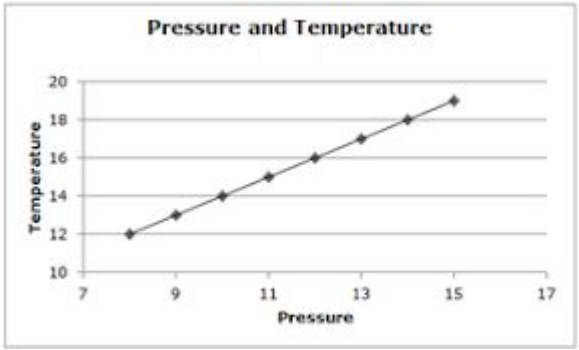
SLU05 - Covariance and Correlation

Nov 26, 2023

Covariance

- **Does, looking to the pairs, tell us something more than looking at individual observations?**
  - Positive slope / trend
  - Negative slope / trend
  - No relationship



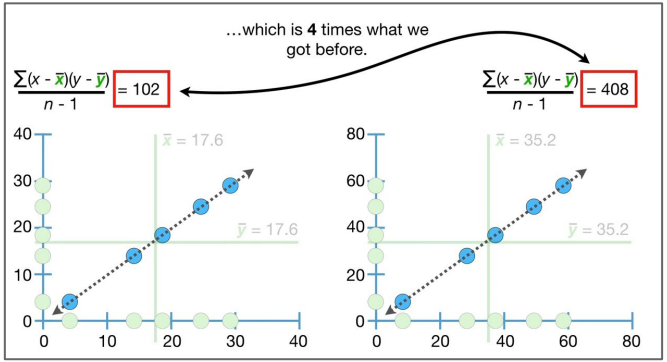


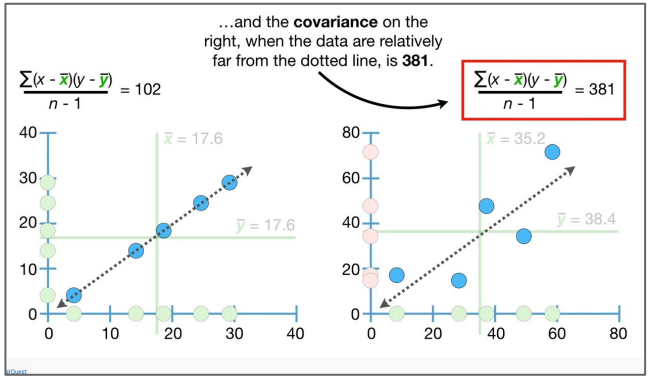
$$cov(X, Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

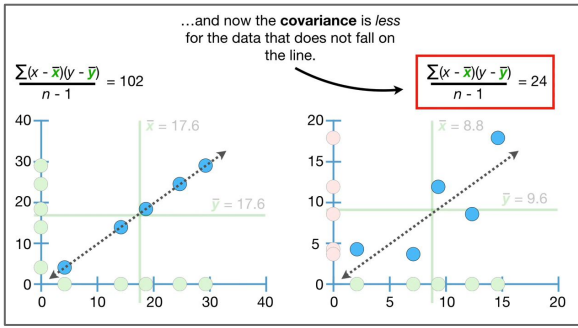
Any problems with using covariance?

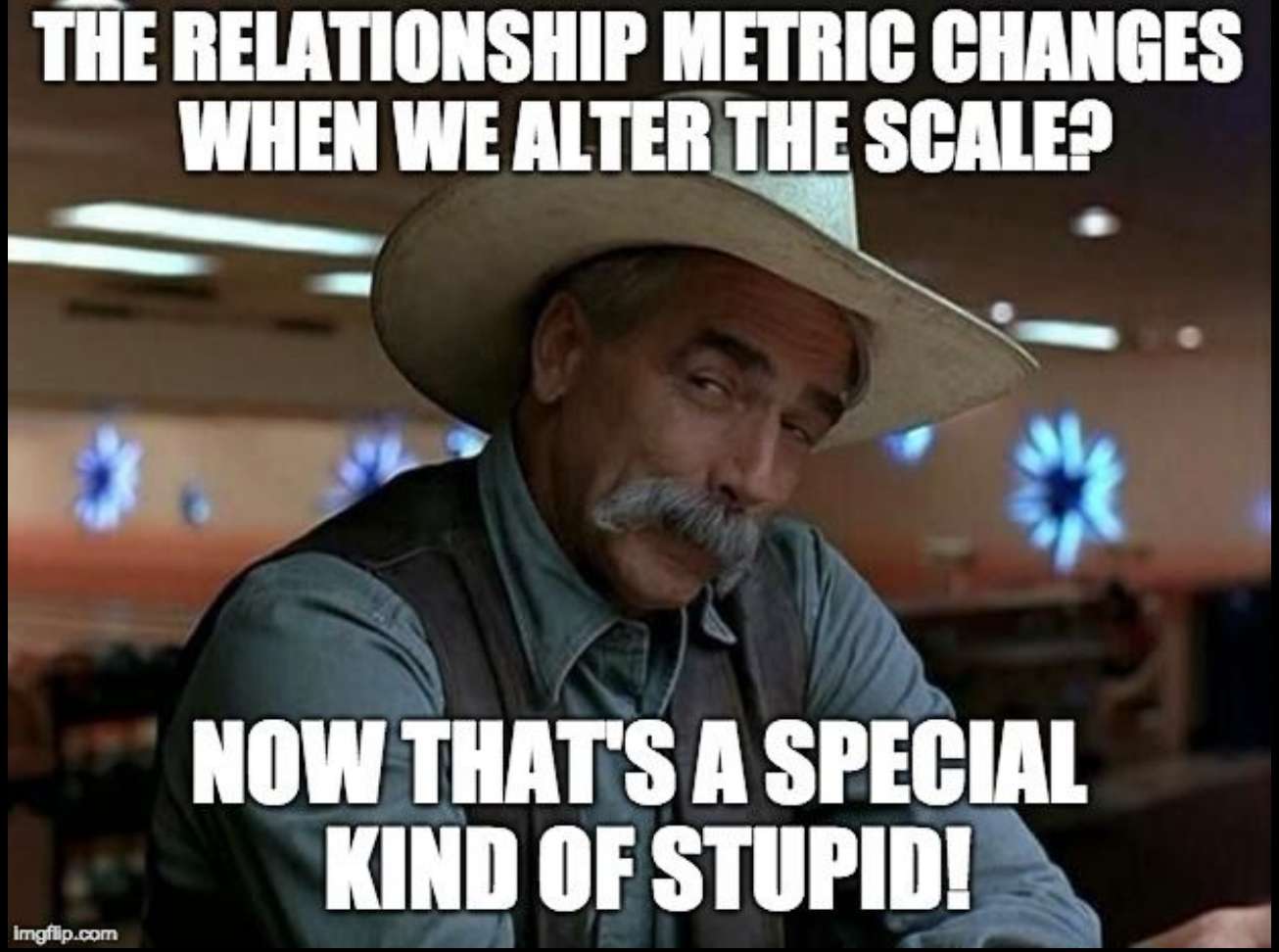
Covariance

- **Important to know**
  - The slope steepness doesn't say anything about the Covariance
  - The point distances to the line also dont
  - So, what happens when you change the scales or points move further from the line? Covariance seems sensitive for this.







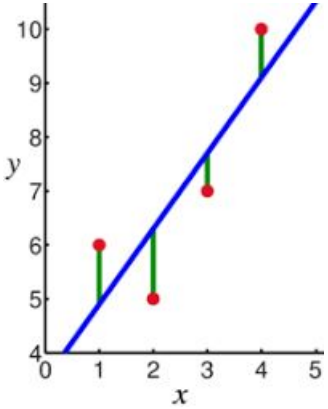

THE RELATIONSHIP METRIC CHANGES WHEN WE ALTER THE SCALE?
NOW THAT'S A SPECIAL KIND OF STUPID!
imgflip.com

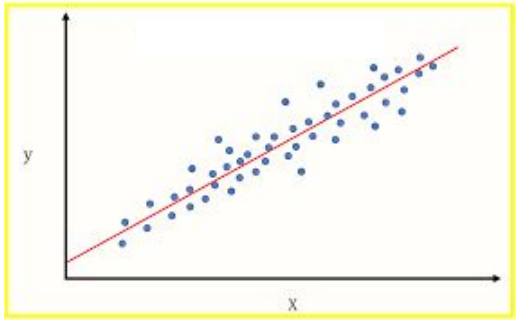
So what works better?

Yay! Correlation!

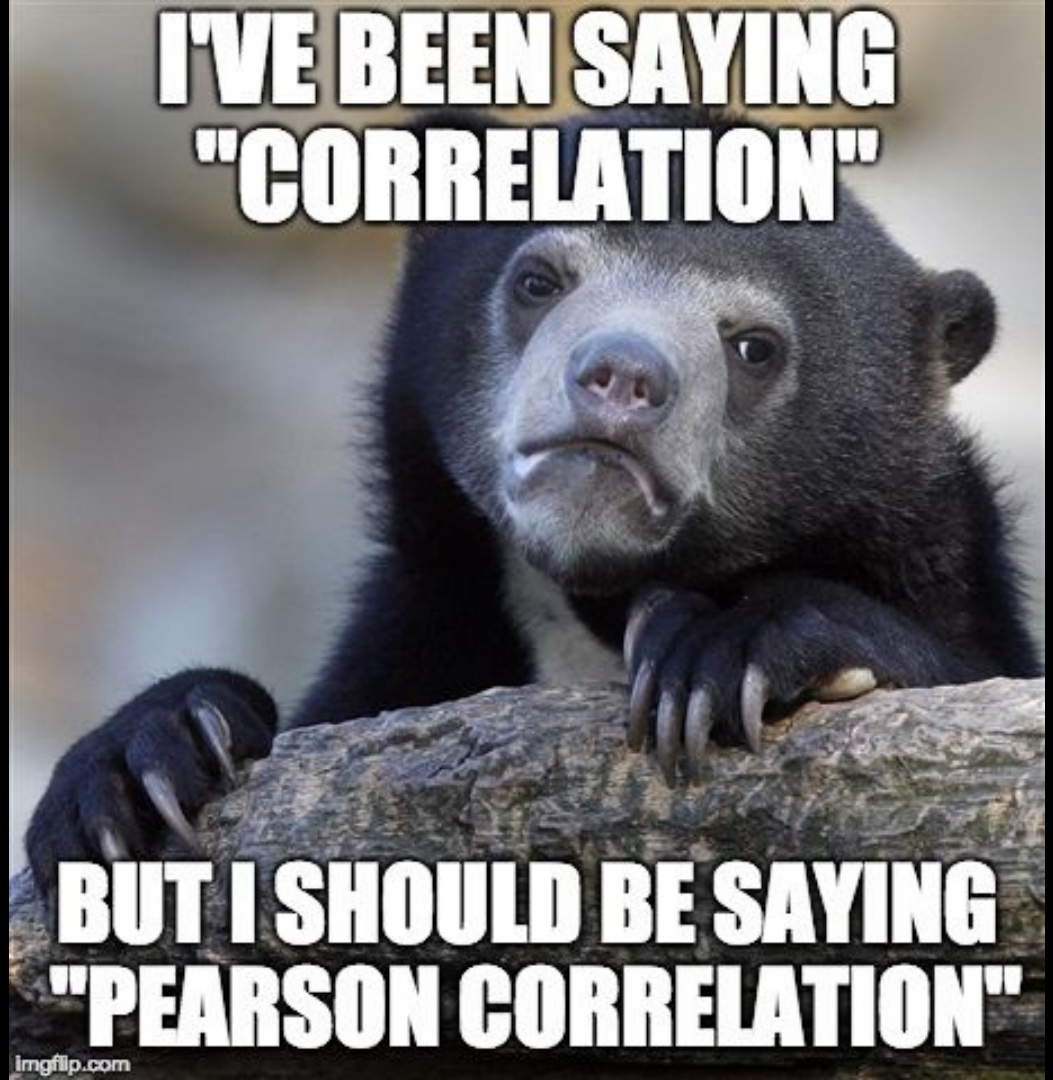
Correlation

- **Important to know**
  - Describes relationships that are not sensitive to the scale of the data
  - We need confidence -> p-value
  - Values between -1 and 1, but what do they mean?
    - 0.0<0.1 no correlation
    - 0.1<0.3 little correlation
    - 0.3<0.5 medium correlation
    - 0.5<0.7 high correlation
    - 0.7<1 very high correlation





Pearson correlation
and
Spearman correlation


I'VE BEEN SAYING
"CORRELATION"
BUT I SHOULD BE SAYING
"PEARSON CORRELATION"
imgflip.com

Pearson Correlation

For normally distributed data

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

Where,

r = Pearson Correlation Coefficient

x_i = x variable samples

y_i = y variable sample

$\bar{x}$ = mean of values in x variable

$\bar{y}$ = mean of values in y variable

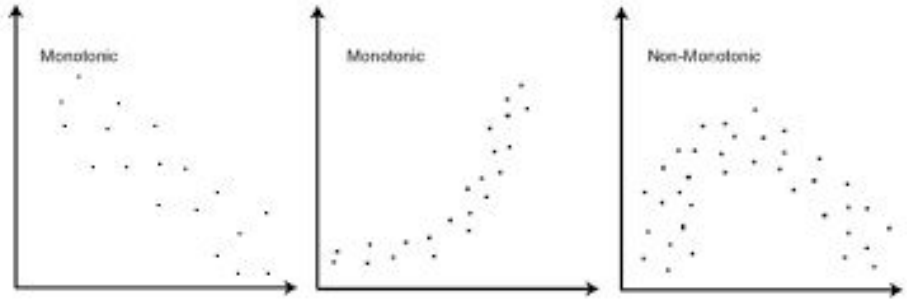
x	y	$(x-\bar{x})$	$(y-\bar{y})$	$(x-\bar{x})(y-\bar{y})$	$(x-\bar{x})^2$	$(y-\bar{y})^2$
2	58	-2	-2	4	4	4
4	32	0	-28	0	0	784
5	63	1	3	3	1	9
7	87	3	27	81	9	729
3	67	-1	7	-7	1	49
1	45	-3	-15	45	9	225
6	68	2	8	16	4	64
$\bar{x} = 4$	$\bar{y} = 60$			$\Sigma = 142$	$\Sigma = 28$	$\Sigma = 1864$

Spearman Correlation

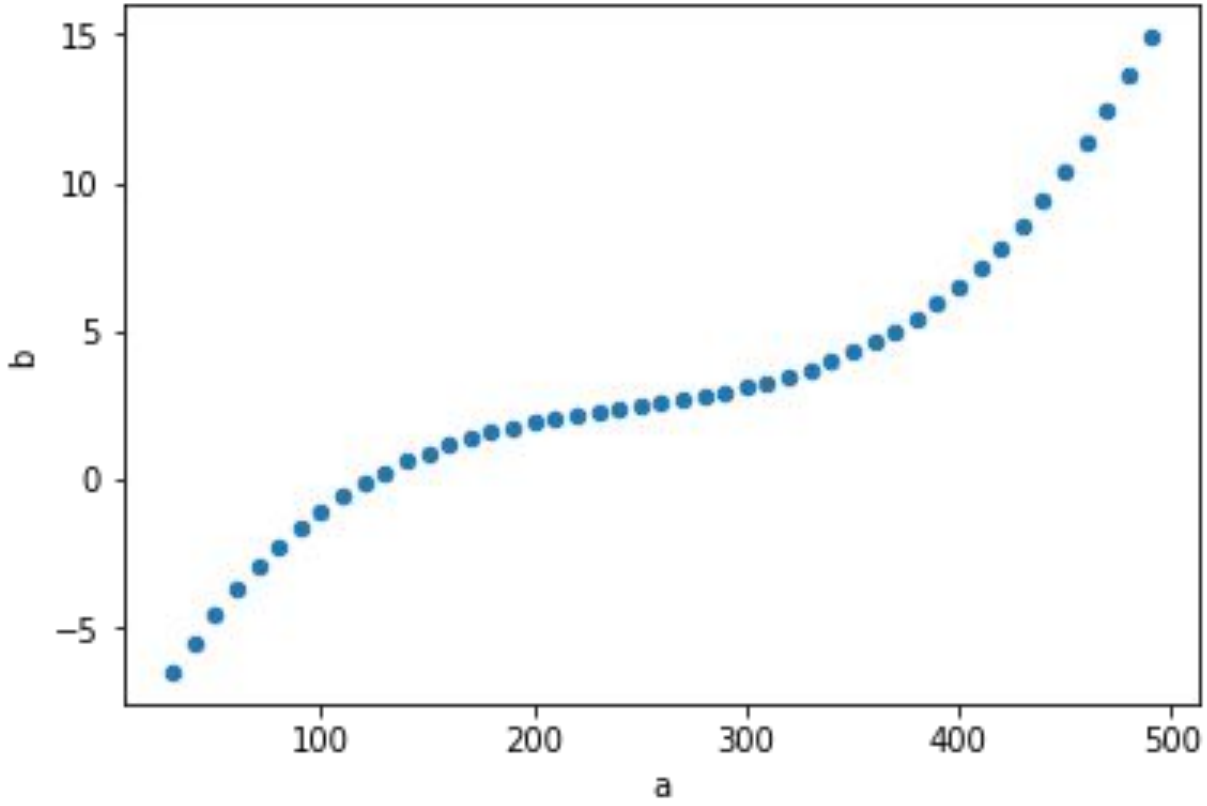
- **Important to know**
  - Monotonic association/relationship
  - Rank
  - Less strict; non linear related is possible

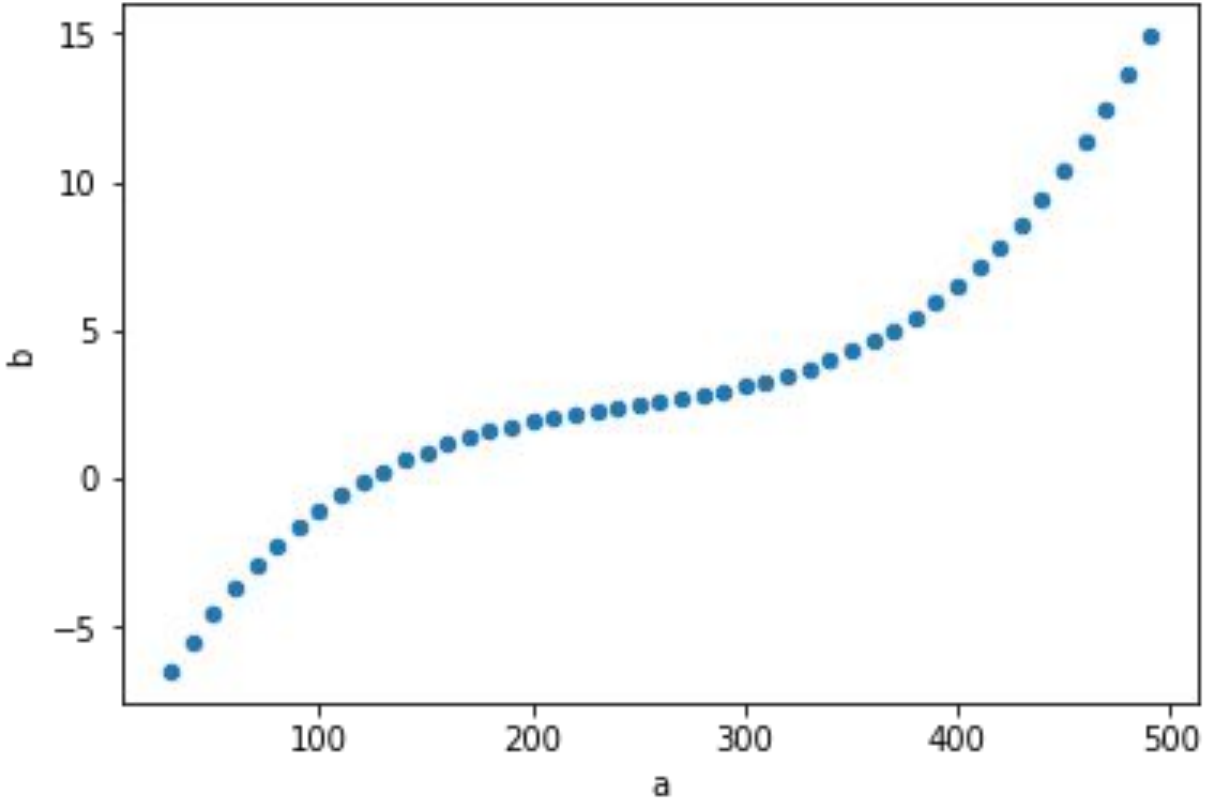
X	Y
2	21
5	17
8	14
11	10
15	5
16	3

X	Y
1	6
2	5
3	4
4	3
5	2
6	1



What is the Spearman Correlation useful for?


15
10
5
0
−5
b
100
200
300
400
500
a




```
Pearson correlation: 0.95
Spearman correlation: 1.00
```

Spearman Correlation

$$r_s = \frac{\sum x_i y_i - \frac{(\sum x_i)(\sum y_i)}{n}}{\sqrt{\sum x_i^2 - \frac{(\sum x_i)^2}{n}}\sqrt{\sum y_i^2 - \frac{(\sum y_i)^2}{n}}}$$

X	Y	XY
1	6	6
2	5	10
3	4	12
4	3	12
5	2	10
6	1	6
21	21	56

$$r_s = \frac{56 - \frac{(21)(21)}{6}}{\sqrt{91 - \frac{21^2}{6}}\sqrt{91 - \frac{21^2}{6}}}$$

$$r_s = \frac{56 - 73.5}{(4.1833)(4.1833)}$$

$$r_s = \frac{-17.5}{17.5} = -1.00$$

x and y have a strong negative relationship.

Take-home message about correlation

1. Correlation is adimensional (unit free)

2. The value of correlation takes place between -1 and +1, while covariance lies between $-\infty$ and $+\infty$.

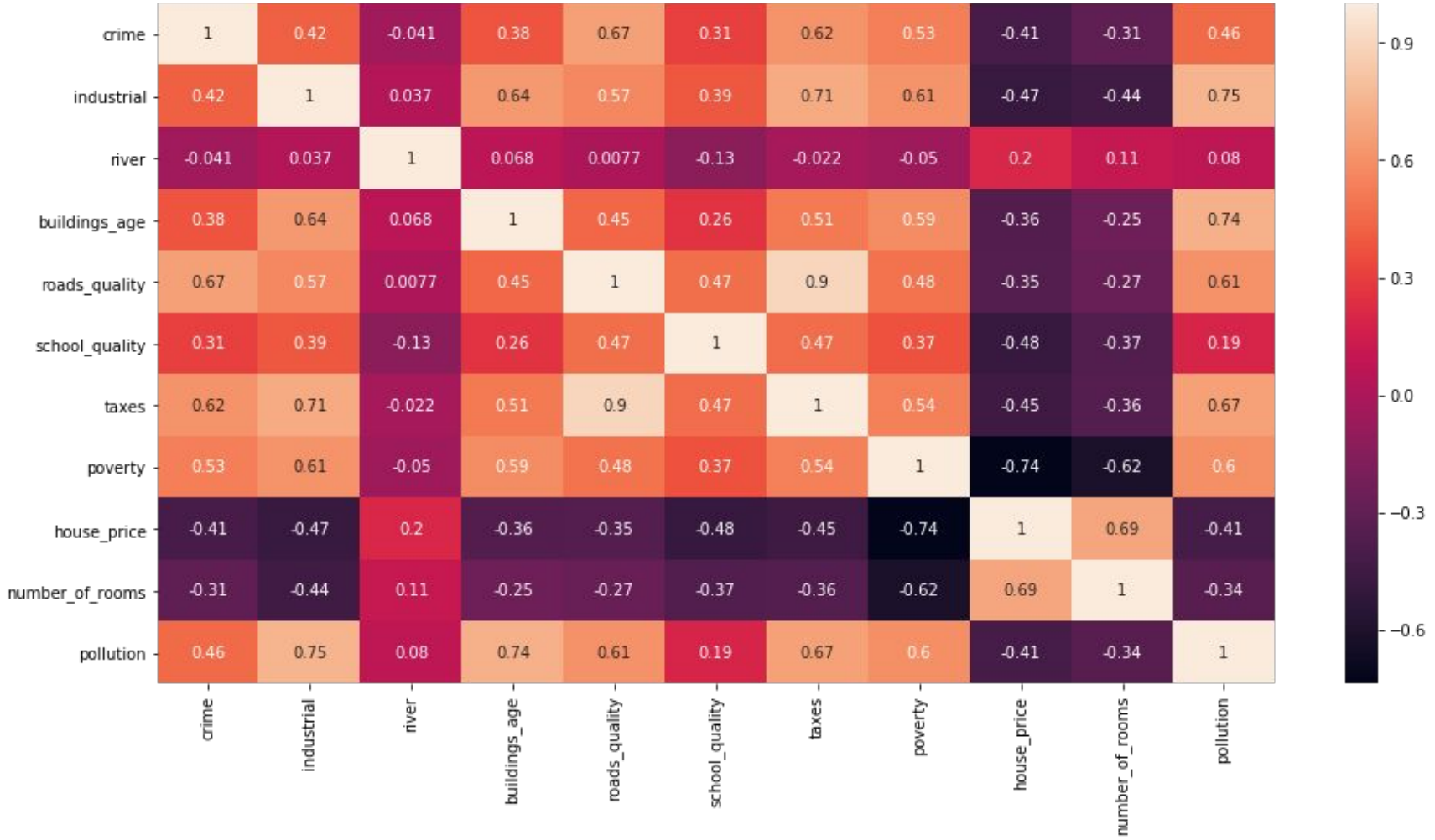
3. Correlation is not affected by the change in scale, while covariance is.

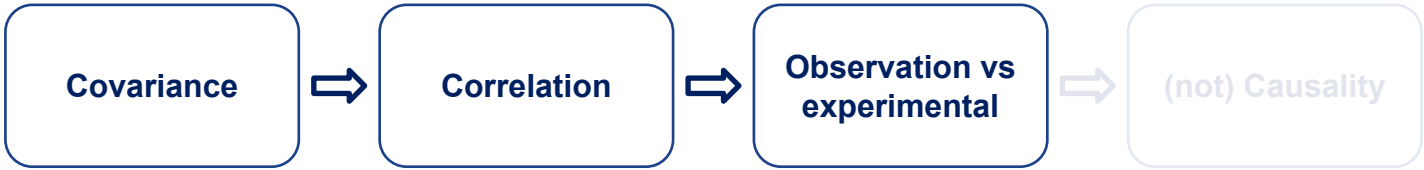
And now…
The correlation matrix

	crime	industrial	river	buildings_age	roads_quality	school_quality	taxes	poverty	house_price	number_of_rooms	pollution
0	0.00632	2.31	0	65.2	1	15.3	296	4.98	24.0	6.575	0.538
1	0.02731	7.07	0	78.9	2	17.8	242	9.14	21.6	6.421	0.469
2	0.03237	2.18	0	45.8	3	18.7	222	2.94	33.4	6.998	0.458
3	0.06905	2.18	0	54.2	3	18.7	222	5.33	36.2	7.147	0.458
4	0.08829	7.87	0	66.6	5	15.2	311	12.43	22.9	6.012	0.524

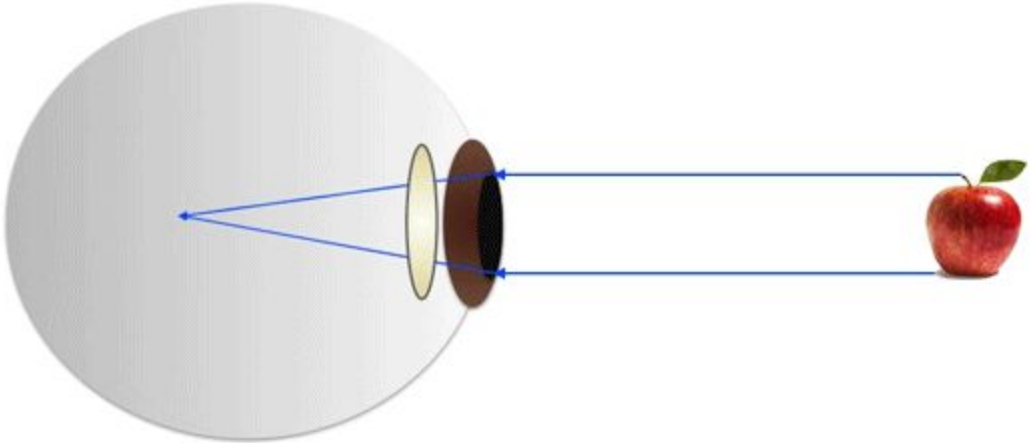
```
data.corr()
```

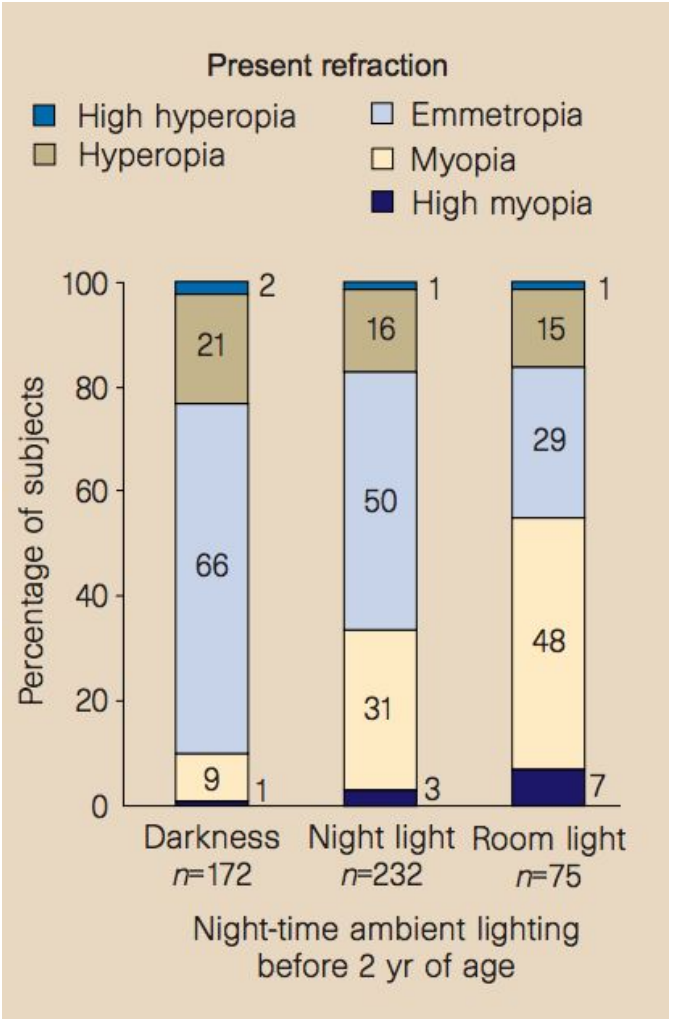
	crime	industrial	river	buildings_age	roads_quality	school_quality	taxes	poverty	house_price	number_of_rooms	pollution
crime	1.000000	0.422228	-0.041195	0.379034	0.666636	0.313409	0.617081	0.532077	-0.407454	-0.310180	0.463001
industrial	0.422228	1.000000	0.037496	0.638378	0.569779	0.391087	0.708313	0.614155	-0.473932	-0.440365	0.750087
river	-0.041195	0.037496	1.000000	0.068286	0.007714	-0.125067	-0.021826	-0.050055	0.204390	0.112251	0.080275
buildings_age	0.379034	0.638378	0.068286	1.000000	0.447380	0.259293	0.511893	0.588834	-0.358888	-0.248573	0.736000
roads_quality	0.666636	0.569779	0.007714	0.447380	1.000000	0.470849	0.903562	0.484568	-0.352251	-0.272783	0.612180
school_quality	0.313409	0.391087	-0.125067	0.259293	0.470849	1.000000	0.467437	0.374802	-0.481376	-0.366927	0.192513
taxes	0.617081	0.708313	-0.021826	0.511893	0.903562	0.467437	1.000000	0.544485	-0.448078	-0.356987	0.670722
poverty	0.532077	0.614155	-0.050055	0.588834	0.484568	0.374802	0.544485	1.000000	-0.738600	-0.615747	0.598874
house_price	-0.407454	-0.473932	0.204390	-0.358888	-0.352251	-0.481376	-0.448078	-0.738600	1.000000	0.689598	-0.413054
number_of_rooms	-0.310180	-0.440365	0.112251	-0.248573	-0.272783	-0.366927	-0.356987	-0.615747	0.689598	1.000000	-0.338515
pollution	0.463001	0.750087	0.080275	0.736000	0.612180	0.192513	0.670722	0.598874	-0.413054	-0.338515	1.000000


crime 1 0.42 -0.041 0.38 0.67 0.31 0.62 0.53 -0.41 -0.31 0.46
industrial 0.42 1 0.037 0.64 0.57 0.39 0.71 0.61 -0.47 -0.44 0.75
river -0.041 0.037 1 0.068 0.0077 -0.13 -0.022 -0.05 0.2 0.11 0.08
buildings_age 0.38 0.64 0.068 1 0.45 0.26 0.51 0.59 -0.36 -0.25 0.74
roads_quality 0.67 0.57 0.0077 0.45 1 0.47 0.9 0.48 -0.35 -0.27 0.61
school_quality 0.31 0.39 -0.13 0.26 0.47 1 0.47 0.37 -0.48 -0.37 0.19
taxes 0.62 0.71 -0.022 0.51 0.9 0.47 1 0.54 -0.45 -0.36 0.67
poverty 0.53 0.61 -0.05 0.59 0.48 0.37 0.54 1 -0.74 -0.62 0.6
house_price -0.41 -0.47 0.2 -0.36 -0.35 -0.48 -0.45 -0.74 1 0.69 -0.41
number_of_rooms -0.31 -0.44 0.11 -0.25 -0.27 -0.37 -0.36 -0.62 0.69 1 -0.34
pollution 0.46 0.75 0.08 0.74 0.61 0.19 0.67 0.6 -0.41 -0.34 1
crime
industrial
river
buildings_age
roads_quality
school_quality
taxes
poverty
house_price
number_of_rooms
pollution
0.9
0.6
0.3
0.0
−0.3
−0.6


Covariance
Correlation
Observation vs experimental
(not) Causality

Great!
Let’s use this in science!





scientific correspondence

Myopia and ambient lighting at night

Myopia, or short-sightedness, occurs when the image of distant objects, focused by the cornea and lens, falls in front of the retina. It commonly arises from excessive postnatal eye growth, particularly in the vitreous cavity. Its prevalence is increasing and now reaches 70–90% in some Asian populations[1,2]. As well as requiring optical correction, myopia is a leading risk factor for acquired blindness in adults because it predisposes individuals to retinal detachment, retinal degeneration and glaucoma. It typically develops in the early school years but can manifest into early adulthood[2]. Its aetiology is poorly understood but may involve genetic and environmental factors[1,2], such as viewing close objects, although how this stimulates eye growth is not known[3]. We have looked recognizable pattern of light exposure. Because early neonatal visual experience markedly affects refractive development in animals[4–6], we evaluated light exposure both at the child's present age and before the age of two years, a period during which the eye grows rapidly[7] but before the usual onset of myopia[2].

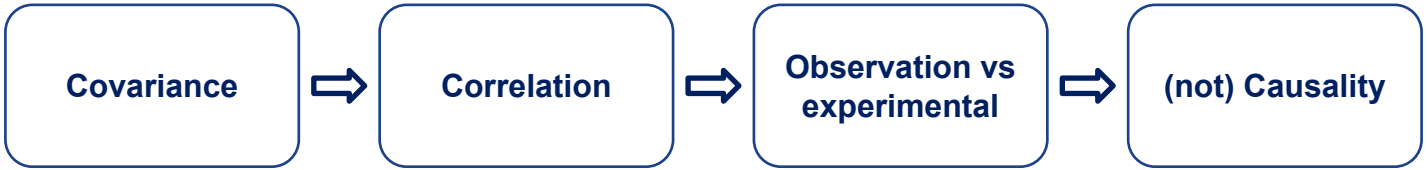
Between January and June 1998, parents of children aged 2–16 years (median age 8.0 years; $n = 479$ children, 55% males; 70% Caucasian, 30% African-American, less than 1% Asian-American) that were seen as outpatients in a university paediatric ophthalmology clinic completed a questionnaire on the child's light exposure both at present and before the age of two years. Children with amblyopia, cataract, glaucoma or a history of prematurity were ed light exposure, including no relation with night-time lighting at the child's present age.

An influence of ambient lighting during sleep on refractive development is plausible, because eyelids of human adults and infants transmit some visible light, mostly at longer wavelengths[8]. The scotopic retinal sensitivity of infants is relatively good compared with that of adults, particularly by the age of 18 weeks[9]. Further, sutured eyelids of infant monkeys transmit a degraded image and perturb refractive development[4].

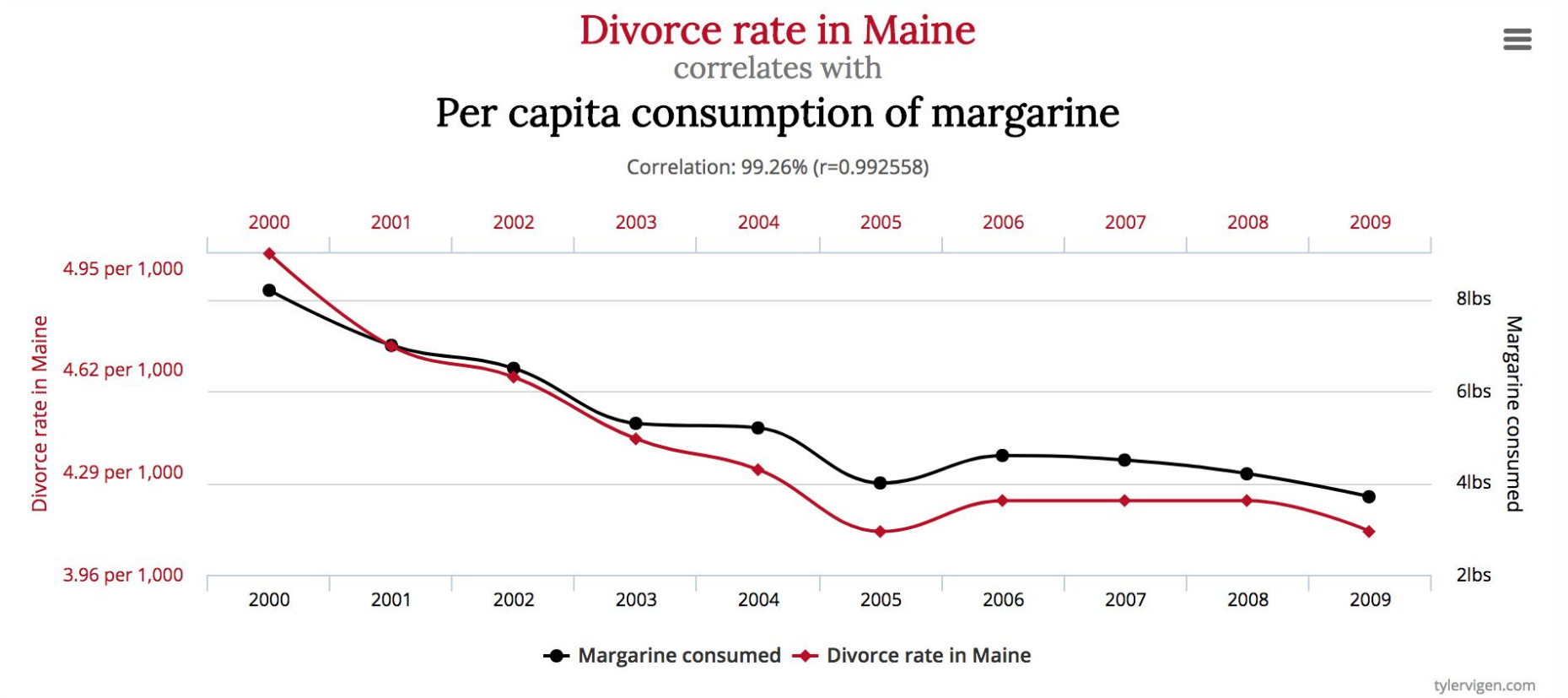
This study does not indicate whether early visual experience influences ocular anatomy by age two or only later, and does not permit conclusions to be made about the timing of the onset or progression of myopia. It raises the possibility of a 'critical

nature

Confounding variable

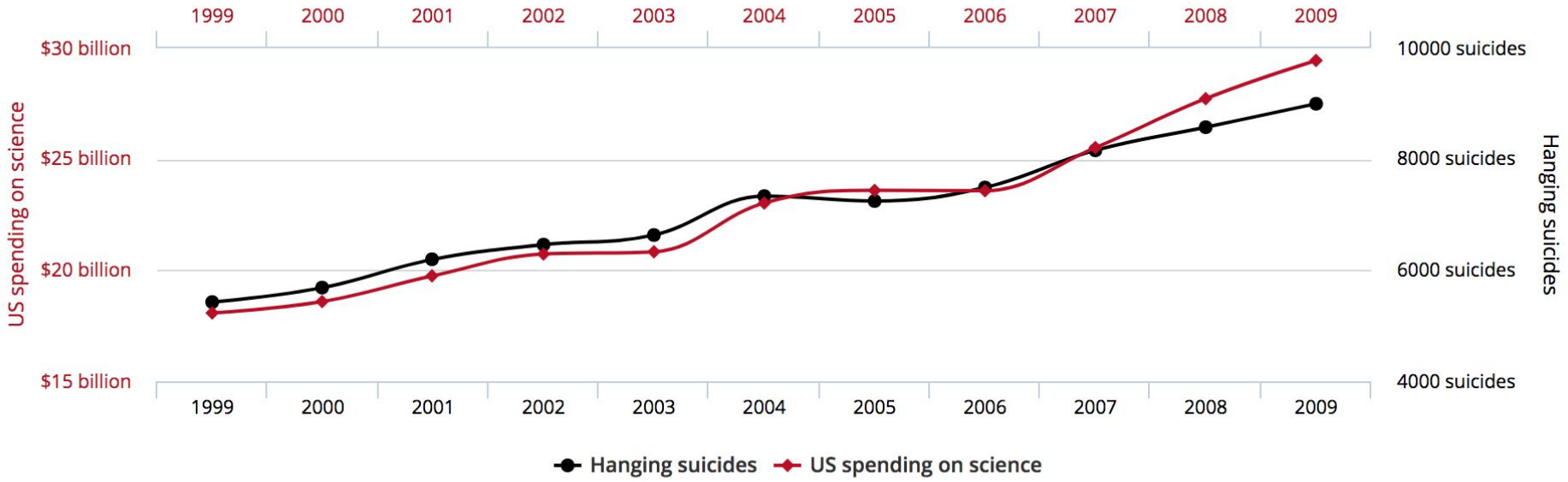

Covariance
Correlation
Observation vs experimental
(not) Causality

Correlation is **not** causation.


Divorce rate in Maine
correlates with
Per capita consumption of margarine
Correlation: 99.26% (r=0.992558)
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
4.95 per 1,000
4.62 per 1,000
4.29 per 1,000
3.96 per 1,000
Divorce rate in Maine
8lbs
6lbs
4lbs
2lbs
Margarine consumed
Margarine consumed
Divorce rate in Maine
tylervigen.com
Data sources: National Vital Statistics Reports and U.S. Department of Agriculture

US spending on science, space, and technology
correlates with
Suicides by hanging, strangulation and suffocation

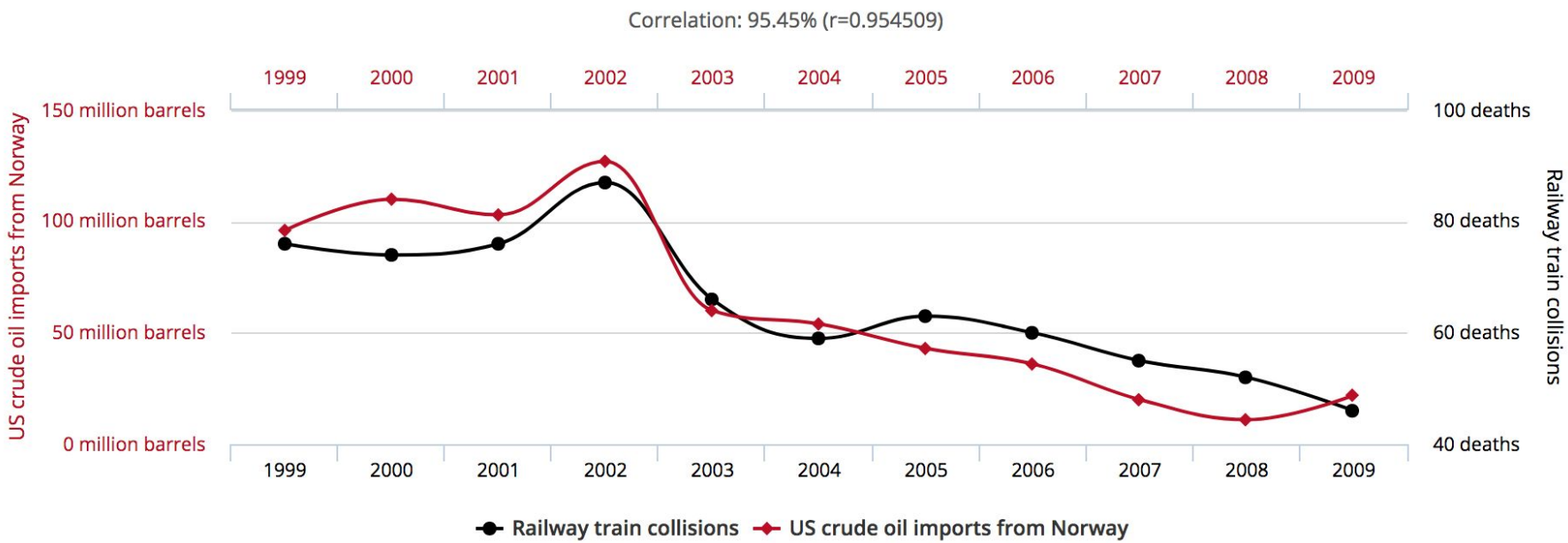
Correlation: 99.79% (r=0.99789126)



tylervigen.com

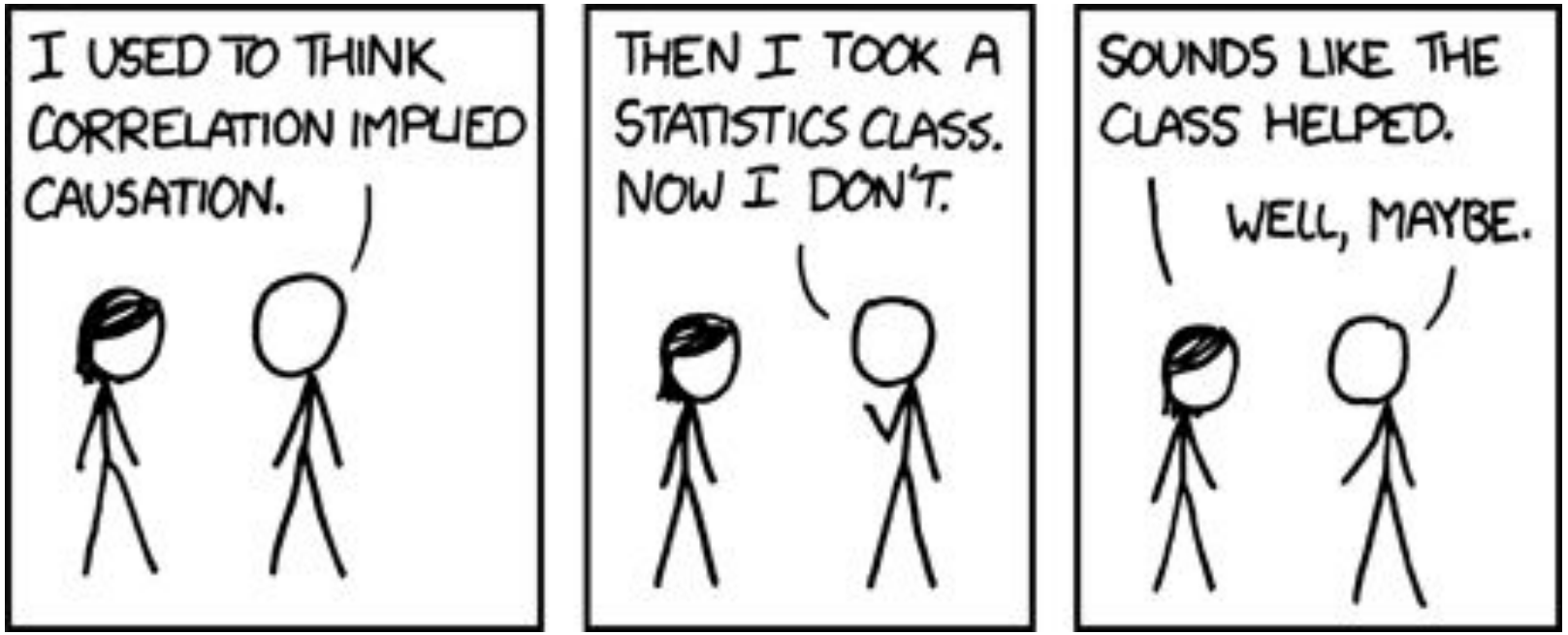
Data sources: U.S. Office of Management and Budget and Centers for Disease Control & Prevention

US crude oil imports from Norway
correlates with
Drivers killed in collision with railway train



tylervigen.com

Data sources: Dept. of Energy and Centers for Disease Control & Prevention


I USED TO THINK CORRELATION IMPLIED CAUSATION.
THEN I TOOK A STATISTICS CLASS. NOW I DON'T.
SOUNDS LIKE THE CLASS HELPED.
WELL, MAYBE.

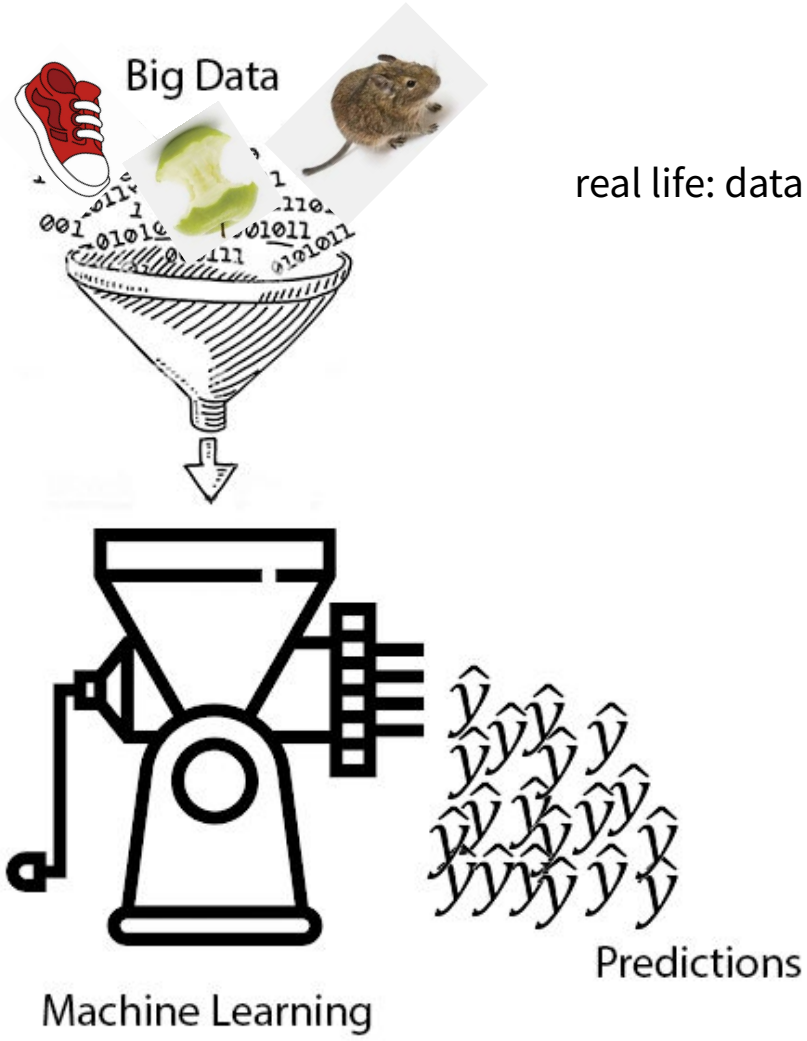
SLU06
Dealing with Data Problems

Nov 26, 2023

Big Data

Predictions

Machine Learning



real life: data is messy

real life: data is messy

bad predictions

Machine Learning

Predictions

No models create meaningful results with messy data.


IN
OUT

What to do with messy data:

Data tidying is the process of **organizing data into properly structured tables**.

Data cleaning or cleansing is the process of **detecting and correcting (or removing) corrupt or inaccurate records** from a record set, table, or database.

- Most of the times, the **datasets** that a data scientist has to work with **are not clean!**
- So, **cleaning your dataset is always the first step** of building a machine learning model.
- There's no one size fits all solution for cleaning a dataset.

Overview

In this **SLU** we will cover the following:

- Tidy data
- Data Entry Problems
  - Data entry problems in categorical variables
  - Data entry problems in numerical variables
- Duplicated entries
- Missing Values

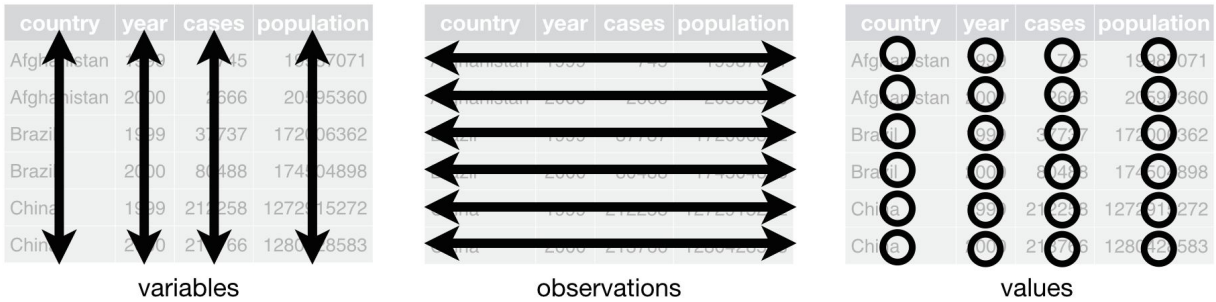
2. Topic Explanation

| Tidy Data |

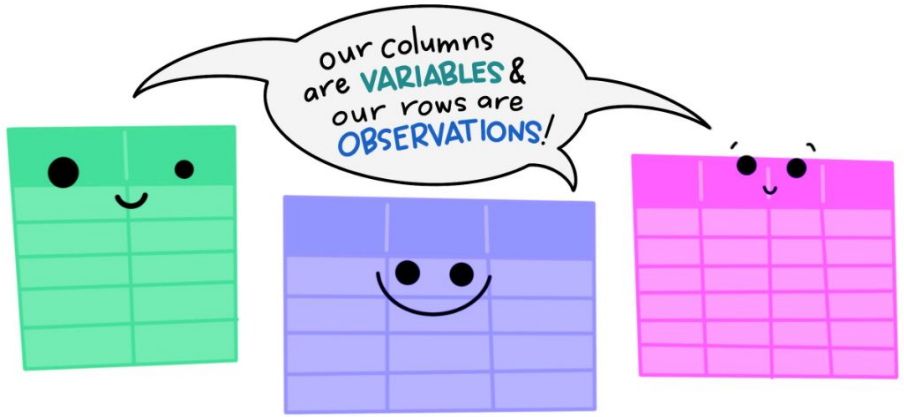
Tidy data

Tidy datasets are easy to manipulate, model and visualize, and have **a specific structure**:

- each **variable** is a *column*
- each **observation** is a *row*
- each *cell* is a **single measurement**
- each **type of observational unit** is a *table*

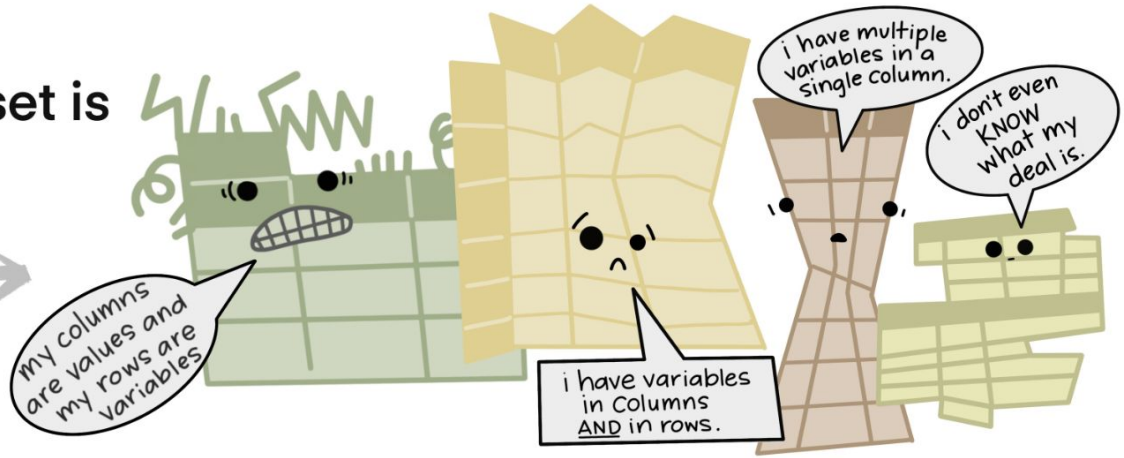


The standard structure of tidy data means that "tidy datasets are all alike..."



"...but every messy dataset is messy in its own way."

–HADLEY WICKHAM



Example

Column / variable names are values!

	religion	<$10k	$10-20k	$20-30k	$30-40k	$40-50k	$50-75k
	Agnostic	27	34	60	81	76	137
	Atheist	12	27	37	52	35	70
2	Buddhist	27	21	30	34	33	58
3	Catholic	418	617	732	670	638	1116
4	Don’t know/refused	15	14	15	11	10	35
5	Evangelical Prot	575	869	1064	982	881	1486
6	Hindu	1	9	7	9	11	34
7	Historically Black Prot	228	244	236	238	197	223
8	Jehovah’s Witness	20	27	24	24	21	30
9	Jewish	19	19	25	25	30	95

Example

Column / variable names are values!

	religion	<$10k	$10-20k	$20-30k	$30-40k	$40-50k	$50-75k
	Agnostic	27	34	60	81	76	137
1	Atheist	12	27	37	52	35	70
2	Buddhist	27	21	30	34	33	58
3	Catholic	418	617	732	670	638	1116
4	Don’t know/refused	15	14	15	11	10	35
5	Evangelical Prot	575	869	1064	982	881	1486
6	Hindu	1	9	7	9	11	34
7	Historically Black Prot	228	244	236	238	197	223
8	Jehovah’s Witness	20	27	24	24	21	30
9	Jewish	19	19	25	25	30	95

income

pd.melt()

```
# Getting the income values from the column names
income_values = [x for x in df_messy_1.columns.tolist() if x!='religion']
print(f"Income values stored in the columns:\n{income_values}")

# Using the melt function to 'melt' the income values into a column
df_tidy_1 = pd.melt(df_messy_1, id_vars=['religion'], value_vars=income_values, var_name='income', value_name='freq')
df_tidy_1.head(15) #showing the top 15 observations
```

```
Income values stored in the columns:
['<$10k', '$10-20k', '$20-30k', '$30-40k', '$40-50k', '$50-75k']
```

	religion	income	freq
0	Agnostic	<$10k	27
1	Atheist	<$10k	12
2	Buddhist	<$10k	27
3	Catholic	<$10k	418
4	Don't know/refused	<$10k	15

the values in the the df_messy dataframe are stored in a new column named “freq”

| Data Entry Problems |


To err is not only human but expected.
Inês Mendes, on an all nighter while creating these materials

Name: PETER Pan

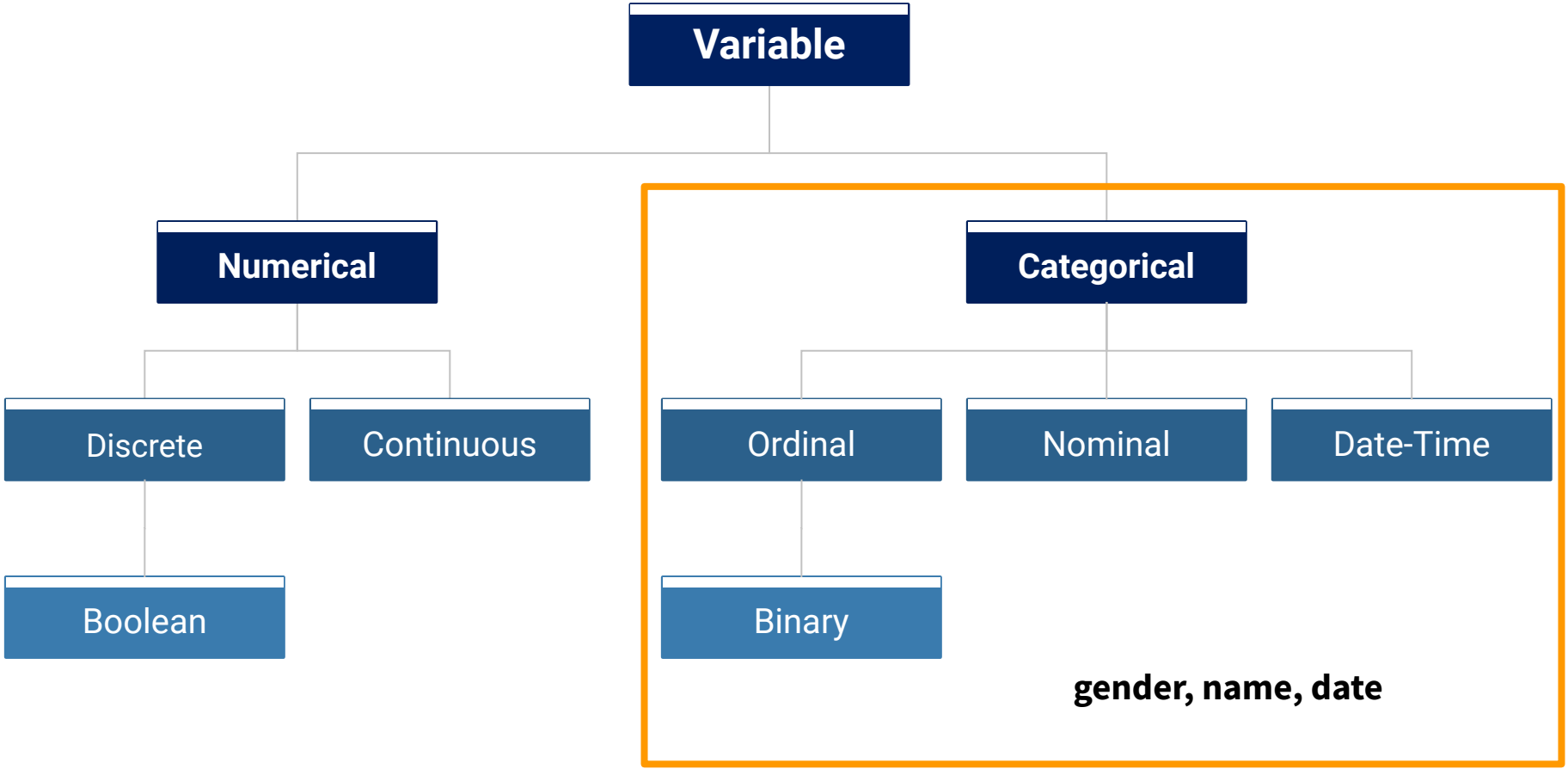
Address: 2753-357

Age: 300

Profession:

Hobbies: Flying

Name: PETER Pan	***case mismatch***
Address: 2753	***wrong datatype***
Age: 300	***nonsensical value***
Profession:	***missing value***
Hobbies: Flying	***extra white spaces***


Variable
Numerical
Categorical
Discrete
Continuous
Ordinal
Nominal
Date-Time
Boolean
Binary
gender, name, date

Data entry problems in categorical variables

nunique()

- counts the number of unique values in a Series (attention to dropna attribute)

```
data.gender.nunique()
```

```
7
```

```
data.gender.nunique(dropna=False)
```

```
8
```

value_counts()

- Shows us the unique values and its counts in a Series (attention to dropna attribute)

```
data['gender'].value_counts()
```

```
female    109
male       66
MALE        9
m           3
F           2
f           1
  female    1
Name: gender, dtype: int64
```

```
data['gender'].value_counts(dropna=False)
```

```
female    109
male       66
NaN         9
MALE        9
m           3
F           2
f           1
  female    1
Name: gender, dtype: int64
```

Problems with free text fields

- We can apply python's string methods to a Series with string values by calling **.str after the Series name**

```
In [3]: df = pd.DataFrame(data)
        df
```

Out[3]:

	A	B
0	0.455924	WORD
1	0.547292	WoRd
2	0.206068	Word
3	0.687819	Two WORDS

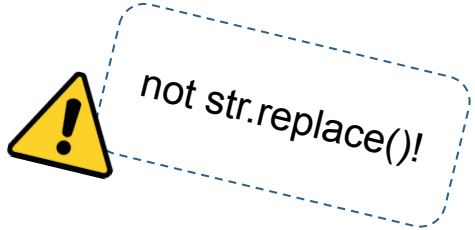
```
In [4]: df.B.str.lower()
```

```
Out[4]: 0          word
        1          word
        2          word
        3     two words
        Name: B, dtype: object
```

Quick overview of string methods

- **str.lower**: makes strings all lower case
- **str.upper**: makes strings all uppercase
- **str.strip**: removes specified characters (whitespace by default) from the strings
- **str.cat**: concatenate something to the strings in Series, like other strings, another Series…
- **str.split**: splits the strings in a Series at a given character into a list, or multiple new Series
- **str.replace**: replaces occurrences of a pattern in the strings of a Series with another pattern. **Accepts regexes.** Very powerful!
- Examples of all of these in the Learning Notebook!

replace



- replace() can be used to replace occurrences of a certain value with another value. Very handy!
- Values to replace can be passed as (among others):
  - `to_replace` as a **list** and `value` as a **list / single value** (useful in the numerical values cases)
  - `to_replace` as **dictionary** (useful in the strings case!)

```
In [14]: df.replace(
             to_replace=[0.142501, 0.605315],
             value=[1, 2]
         )
```

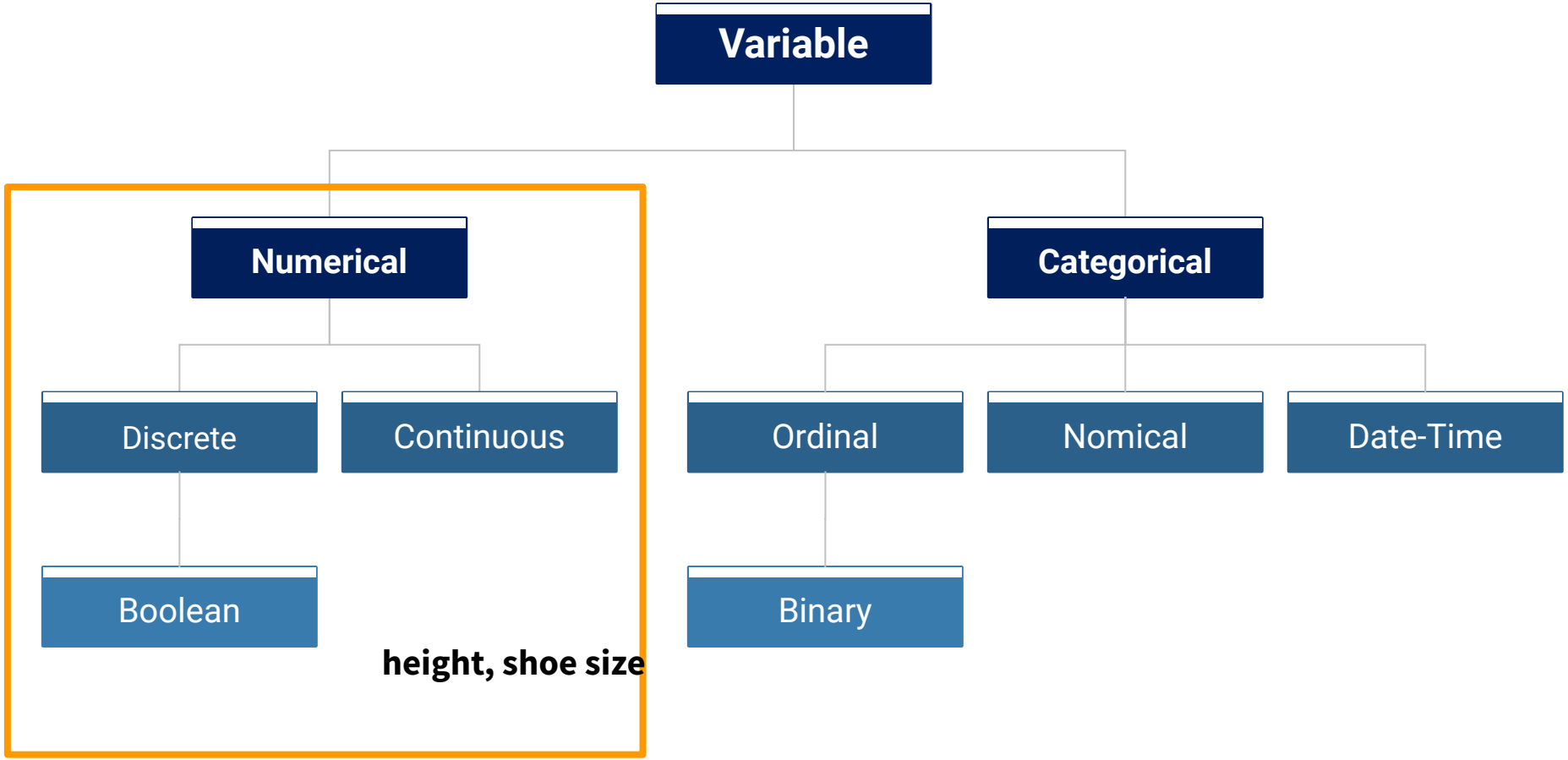
Out[14]:

	A	B
0	1.000000	WORD
1	0.675757	WoRd
2	2.000000	Word
3	0.714094	Two WORDS

```
In [10]: df.replace(to_replace={'WORD': 'ola'})
```

Out[10]:

	A	B
0	0.142501	ola
1	0.675757	WoRd
2	0.605315	Word
3	0.714094	Two WORDS


Variable
Numerical
Categorical
Discrete
Continuous
Ordinal
Nomical
Date-Time
Boolean
Binary
height, shoe size

Data entry problems in numerical variables

- Highly dependent on the **nature** of a variable:
  - What is and isn't out of range?
  - Can we consider outliers?
  - Can there be gaps?

SLU02 - Subsetting Data in Pandas

```
data['age'][(data['age'] < 0) | (data['age'] > 123)]
```

```
TSHRBGFOJU    300000000.0
SQYVEHAWSW          224.0
Name: age, dtype: float64
```

| Duplicated Entries |

Duplicated data

Dealing with duplicated entries requires two steps:

1. Define **what a duplicate is** in the context of our problem;
2. **Search** for duplicated entries **and dro**p them from the dataset.

Detecting duplicates

- We can detect duplicate rows in our dataset using [duplicated()]
-

```
data.head(10)
```

	age	height	gender
CFLOXRHMDR	88.0	163.0	female
FXLJSNLSOG	29.0	158.0	female
FWDIVJKGOI	42.0	159.0	female
YWEBKQWHRE	25.0	179.0	male
YPUQAPSOYJ	32.0	169.0	male
YPUQAPSOYJ	32.0	169.0	male
YPUQAPSOYJ	32.0	169.0	male
YPUQAPSOYJ	32.0	169.0	male
SSZQEGTLNK	NaN	162.0	male
PRFEFXNGWN	36.0	166.0	female

```
duplicated_mask = data.duplicated(keep='first')

print('Number of duplicates:', duplicated_mask.sum())
```

```
Number of duplicates: 6
```

```
data[duplicated_mask]
```

	age	height	gender
YPUQAPSOYJ	32.0	169.0	male
YPUQAPSOYJ	32.0	169.0	male
YPUQAPSOYJ	32.0	169.0	male
XUAJJPLVOI	18.0	168.0	female
TRMMGYEEPC	21.0	156.0	female
ZNLRYQHPXJ	25.0	155.0	male

Eliminating duplicates

- We can eliminate duplicates in a DataFrame using [drop_duplicates()]

```
data.head(10)
```

	age	height	gender
CFLOXRHMDR	88.0	163.0	female
FXLJSNLSOG	29.0	158.0	female
FWDIVJKGOI	42.0	159.0	female
YWEBKQWHRE	25.0	179.0	male
YPUQAPSOYJ	32.0	169.0	male
SSZQEGTLNK	NaN	162.0	male
PRFEFXNGWN	36.0	166.0	female
IIVXDNOAIV	1.0	165.0	female
VVQYVNRAGQ	18.0	134.0	female
YVEDWPTEEB	31.0	149.0	female

```
print(f"Shape before dropping duplicates: {data.shape}")
data = data.drop_duplicates()
print(f"Shape after dropping duplicates: {data.shape}")
```

```
Shape before dropping duplicates: (200, 3)
Shape after dropping duplicates: (194, 3)
```

| Missing Values |

Detecting missing values

- We can detect missing values with **isnull()**
- Works like a **boolean mask**: True or False depending on whether the value is null or not
- We can count the missing values per variable by calling a **sum()** afterwards

```
data.isnull()
```

	age	height	gender
AGFHBQDTEG	False	False	False
HYTVHSPPVG	False	False	False
DSBFYTZEQN	False	False	False
VYAQBLJKXJ	True	False	False
BLAKTCGBMO	False	False	False

```
data.isnull().sum()
```

```
age       9
height    4
gender    9
dtype: int64
```

SLU02 - Subsetting Data in Pandas

Handling missing values - dropping them

- First possible strategy is to drop the null values, with dropna()
- We can do it either with the '**all**' strategy or with the '**any**' strategy

```
df
```

	A	B
0	NaN	None
1	0.950714	WoRd
2	NaN	Word
3	0.598658	Two WORDS

```
df.dropna(how='all')
```

	A	B
1	0.950714	WoRd
2	NaN	Word
3	0.598658	Two WORDS

Removes a row if **all** values are missing

```
df.dropna(how='any')
```

	A	B
1	0.950714	WoRd
3	0.598658	Two WORDS

Removes a row if **one or more** values are missing

Handling missing values - filling them in

- In order to input missing values, we can use **fillna()**
- **If the variable is numerical,** it's usual to replace it with the **mean or median of the distribution**
- If the **variable is categorical,** it's usual to replace it with **a new category or with the most common value**

```
df
```

	A	B
0	NaN	None
1	0.950714	WoRd
2	NaN	Word
3	0.598658	Two WORDS

```
df.fillna({'A': df.A.mean(), 'B': 'unknown'})
```

	A	B
0	0.774686	unknown
1	0.950714	WoRd
2	0.774686	Word
3	0.598658	Two WORDS

3. Recap

- **First step** when working on a data problem is always to **tidy and clean the dataset**
- To handle dirty **free-text input** data we can use pandas **string methods or replace**
- We can detect and drop **duplicates**
- When handling **missing values** we can:
    - **Drop** observations with missing values
    - **Input missing values**, by using knowledge about the distribution of the variables, or by using a different label