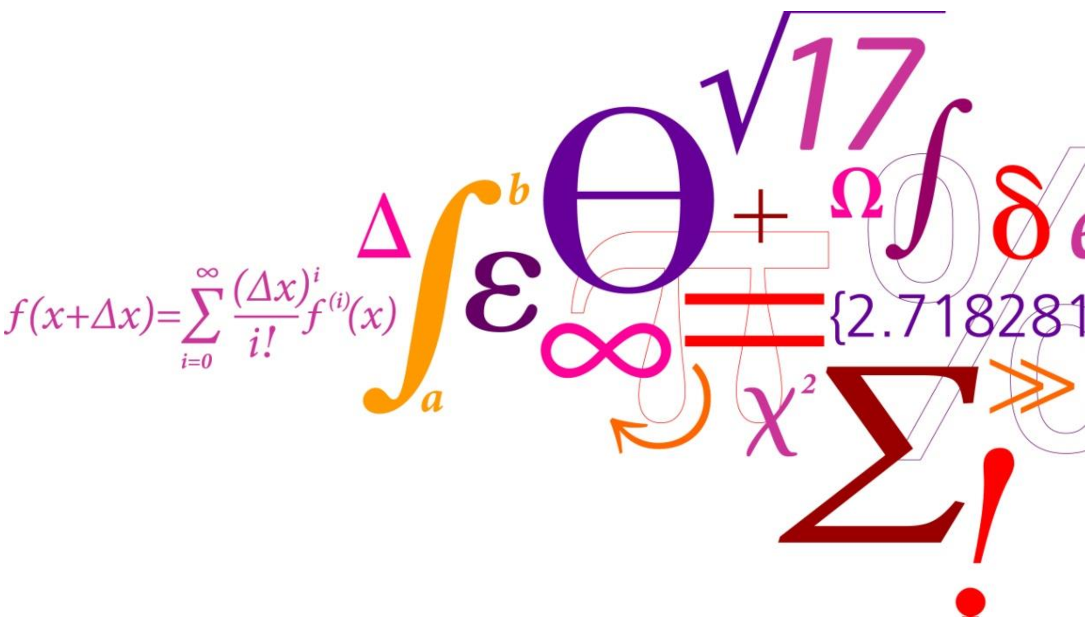
Data preparation



Data preparation

- All sorts of little tasks
  - Parse datasets
  - Convert value types (e.g. numeric to nominal)
  - Eliminate errors, (useless) outliers
  - Obtain intermediate values (e.g. $x_{n+1}=f(x_1,x_2)$)

- Descriptive statistics

- This is where we spend MOST of the time! Some people say 90%...

Jupyter notebook

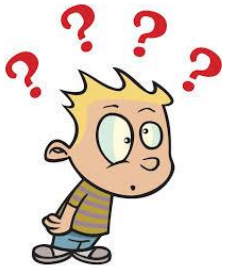
- Data in the slides comes from the Hubway bike sharing dataset,
- In the exercise notebook (“3. Data wrangling with Pandas.ipynb”), however, we instead reuse the dataset from last week

Why prepare data?

- Real-life data does not come in perfect condition, need to work on it in order to get something out of it
- **Incomplete data**
  - some people don't report their income, gender, education, etc.
  - lack of measurement at some times of the day

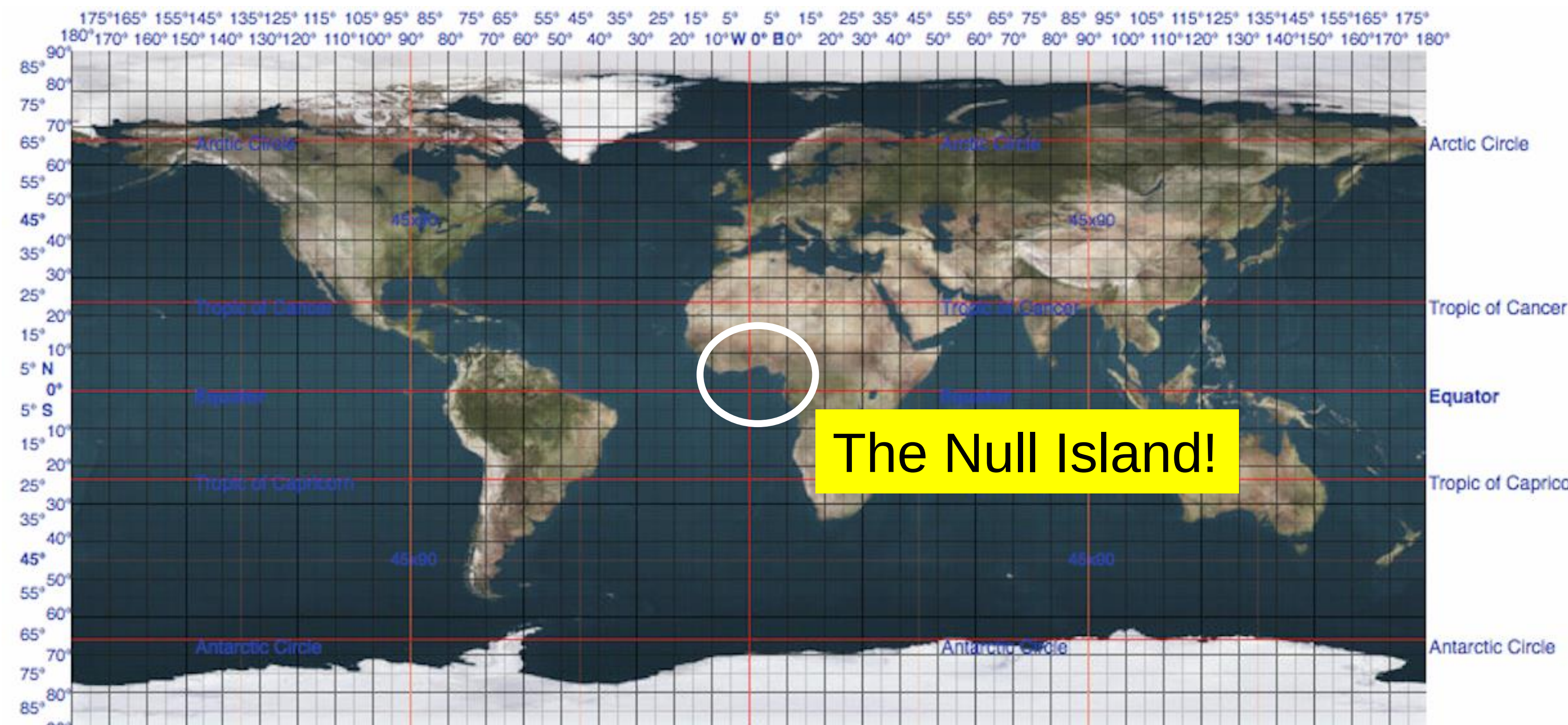
Why prepare data?

- **Noisy data**: errors in data collection, measurement, coding etc. May lead to outliers.
  - errors in recording of boarding/alighting passengers
  - Illogical values as a result of recording errors:
    - e.g. Age = -10 ??



Why prepare data?

– Errors in GPS locations based on the precision of the GPS device



Why prepare data?

- **Inconsistent data**: discrepancies in coding
  - Age is recorded as 65 BUT birth year is 2005

  - Daytime shuttle ends the operation around 6pm BUT there is still data after 6pm

An example

- Hubway trips

- Dataset available
http://hubwaydatachallenge.org/trip-history-data/
 All trips from 2011 to 2013!

- Some detail
  - starting/ending station
  - Time of day
  - Gender, Age, zipcode
  - Type of user
  - …

seq_id	23
hubway_id	37
status	Closed
duration	1582
start_date	7/28/2011 12:01:00
strt_statn	38
…	…
Gender	Female

Assumptions on x and y

- *y* is the target variable

    Continuous, nominal, ordinal, interval?...

    y=number of trips

        Continuous

    y=destination choice

        nominal/categorical

    y=departure time choice

        ordinal

    y=number of trips

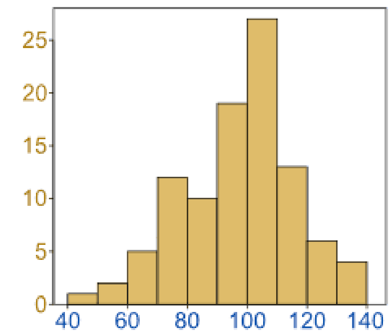
        interval

seq_id	23
hubway_id	37
status	Closed
duration	1582
start_date	7/28/2011 12:01:00
strt_statn	38
...	...
Gender	Female

Assumptions on x and y

- Domain of y?
  - Can it be negative? Can it be 0?
  - Are there abnormal cases in the dataset (what to do with them?)?

- Consider transformations for y (look at its distribution)
  - $y' = \log(y)$
  - $y' = 1/y$
  - $y' = e^y$



Assumptions on x and y

- Generally same questions for x
  - Case of nominal. Example for day of week:
    - Usually in data, Monday=1, Tuesday=2, …, Sunday=7

    - Why not other numbers?
      - Monday=0, Tuesday=1, … Sunday=6

    - These numbers are just symbols

Assumptions on x and y

- Nominal x
  - Common treatment is through *dummy variables*

    Monday→ isMonday {0,1};

    Tuesday→ isTuesday {0, 1}

    …

- For example, type of user:
  - Registered VS Casual

      Registered=1; Casual=0

Assumptions on x and y

- The classical problems with dates
  - Separator is / or -?
  - Order (YYYY-MM-DD or something else?)
  - Date and time together?

Tasks in Data Processing

- **Data parsing**: identify individual data elements in a source file

A1 fx AJA 0 0 00:00 05:00 1 1 05:00 14:00 2 1 14:00 16:00 1 1 16:00 18:00 1 2 18:00 20:00 1 1 20:00 00:00

	A
1	AJA 0 0 00:00 05:00 1 1 05:00 14:00 2 1 14:00 16:00 1 1 16:00 18:00 1 2 18:00 20:00 1 1 20:00 00:00
2	AMS 1 1 00:00 07:00 4 2 07:00 09:00 1 1 09:00 12:00 2 3 12:00 15:00 1 1 15:00 17:00 3 3 17:00 19:00 1 1 19:00 00:00
3	AVN 0 0 00:00 05:00 1 1 05:00 21:00 0 0 21:00 00:00
4	BES 1 1 00:00 05:00 3 1 05:00 07:00 1 1 07:00 09:00 3 3 09:00 14:00 2 2 14:00 00:00
5	BIA 1 1 00:00 00:00
6	BIQ 1 1 00:00 05:00 2 1 05:00 07:00 1 1 07:00 20:00 1 2 20:00 22:00 1 1 22:00 00:00
7	BOD 1 1 00:00 05:00 2 1 05:00 07:00 2 2 07:00 20:00 1 2 20:00 22:00 1 1 22:00 00:00
8	CDG 4 4 00:00 06:00 8 10 06:00 08:00 10 8 08:00 11:00 12 14 11:00 13:00 8 8 13:00 17:00 12 8 17:00 18:00 8 6 18:00 20:00 12 8 20:00 22:00 4 4 22:00 00:00
9	CFE 0 0 00:00 05:00 2 1 05:00 07:00 1 1 07:00 15:00 2 2 15:00 18:00 1 1 18:00 00:00
10	CFR 0 0 00:00 05:00 1 1 05:00 21:00 0 0 21:00 00:00
11	CLY 0 0 00:00 05:00 1 1 05:00 21:00 0 0 21:00 00:00
12	ETZ 0 0 00:00 05:00 1 1 05:00 00:00
13	FSC 0 0 00:00 05:00 1 1 05:00 00:00
14	LDE 0 0 00:00 05:00 1 1 05:00 21:00 0 0 21:00 00:00
15	LEH 0 0 00:00 05:00 1 1 05:00 00:00
16	LGW 1 1 00:00 07:00 2 2 07:00 09:00 1 1 09:00 00:00
17	LIG 0 0 00:00 05:00 2 1 05:00 07:00 1 1 07:00 00:00
18	LIL 1 1 00:00 00:00
19	LRT 0 0 00:00 05:00 1 1 05:00 13:00 2 2 13:00 15:00 1 1 15:00 21:00 0 0
20	LYS 2 2 00:00 06:00 10 8 06:00 08:00 4 3 08:00 11:00 3 2 11:00 13:00 4
21	MLH 1 1 00:00 09:00 2 2 09:00 15:00 1 1 15:00 00:00
22	MPL 1 1 00:00 05:00 2 1 05:00 07:00 1 1 07:00 09:00 2 2 09:00 11:00 4
23	MRS 1 1 00:00 05:00 3 1 05:00 07:00 1 3 07:00 09:00 4 4 09:00 15:00 4
24	NCE 1 1 00:00 05:00 4 1 05:00 07:00 2 4 07:00 09:00 4 4 09:00 11:00 5 5
25	NTE 1 1 00:00 05:00 2 1 05:00 07:00 3 1 07:00 09:00 5 4 09:00 11:00 3 3
26	ORY 4 4 00:00 06:00 15 15 06:00 08:00 17 12 08:00 11:00 15 15 11:00 1
27	PGF 0 0 00:00 05:00 1 1 05:00 00:00
28	PUF 1 1 00:00 05:00 2 1 05:00 07:00 1 1 07:00 09:00 2 2 09:00 11:00 1 1
29	RDZ 0 0 00:00 05:00 1 1 05:00 21:00 0 0 21:00 00:00
30	RNS 1 1 00:00 05:00 2 1 05:00 07:00 1 1 07:00 09:00 1 2 09:00 11:00 1 1
31	SXB 1 1 00:00 05:00 2 1 05:00 07:00 1 2 07:00 09:00 3 3 09:00 11:00 2 2
32	TLN 0 0 00:00 05:00 1 1 05:00 00:00
33	TLS 1 1 00:00 05:00 3 1 05:00 07:00 6 4 07:00 09:00 4 4 09:00 12:00 6 2
34	UIP 0 0 00:00 05:00 1 1 05:00 21:00 0 0 21:00 00:00
35	URO 0 0 00:00 05:00 1 1 05:00 00:00
36	#

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	AJA	0	0	0:00	5:00	1	1	5:00	14:00	2	1	14:00	16:00	1	1	16:00	18:00	1
2	AMS	1	1	0:00	7:00	4	2	7:00	9:00	1	1	9:00	12:00	2	3	12:00	15:00	1
3	AVN	0	0	0:00	5:00	1	1	5:00	21:00	0	0	21:00	0:00					
4	BES	1	1	0:00	5:00	3	1	5:00	7:00	1	1	7:00	9:00	3	3	9:00	14:00	2
5	BIA	1	1	0:00	0:00													
6	BIQ	1	1	0:00	5:00	2	1	5:00	7:00	1	1	7:00	20:00	1	2	20:00	22:00	1
7	BOD	1	1	0:00	5:00	2	1	5:00	7:00	2	2	7:00	20:00	1	2	20:00	22:00	1
8	CDG	4	4	0:00	6:00	8	10	6:00	8:00	10	8	8:00	11:00	12	14	11:00	13:00	8
9	CFE	0	0	0:00	5:00	2	1	5:00	7:00	1	1	7:00	15:00	2	2	15:00	18:00	1
10	CFR	0	0	0:00	5:00	1	1	5:00	21:00	0	0	21:00	0:00					
11	CLY	0	0	0:00	5:00	1	1	5:00	21:00	0	0	21:00	0:00					
12	ETZ	0	0	0:00	5:00	1	1	5:00	0:00									
13	FSC	0	0	0:00	5:00	1	1	5:00	0:00									
14	LDE	0	0	0:00	5:00	1	1	5:00	21:00	0	0	21:00	0:00					
15	LEH	0	0	0:00	5:00	1	1	5:00	0:00									
16	LGW	1	1	0:00	7:00	2	2	7:00	9:00	1	1	9:00	0:00					
17	LIG	0	0	0:00	5:00	2	1	5:00	7:00	1	1	7:00	0:00					
18	LIL	1	1	0:00	0:00													
19	LRT	0	0	0:00	5:00	1	1	5:00	13:00	2	2	13:00	15:00	1	1	15:00	21:00	0
20	LYS	2	2	0:00	6:00	10	8	6:00	8:00	4	3	8:00	11:00	3	2	11:00	13:00	4
21	MLH	1	1	0:00	9:00	2	2	9:00	15:00	1	1	15:00	0:00					
22	MPL	1	1	0:00	5:00	2	1	5:00	7:00	1	1	7:00	9:00	2	2	9:00	11:00	4
23	MRS	1	1	0:00	5:00	3	1	5:00	7:00	1	3	7:00	9:00	4	4	9:00	15:00	4
24	NCE	1	1	0:00	5:00	4	1	5:00	7:00	2	4	7:00	9:00	4	4	9:00	11:00	5
25	NTE	1	1	0:00	5:00	2	1	5:00	7:00	3	1	7:00	9:00	5	4	9:00	11:00	3
26	ORY	4	4	0:00	6:00	15	15	6:00	8:00	17	12	8:00	11:00	15	15	11:00	13:00	12
27	PGF	0	0	0:00	5:00	1	1	5:00	0:00									
28	PUF	1	1	0:00	5:00	2	1	5:00	7:00	1	1	7:00	9:00	2	2	9:00	11:00	1
29	RDZ	0	0	0:00	5:00	1	1	5:00	21:00	0	0	21:00	0:00					
30	RNS	1	1	0:00	5:00	2	1	5:00	7:00	1	1	7:00	9:00	1	2	9:00	11:00	1
31	SXB	1	1	0:00	5:00	2	1	5:00	7:00	1	2	7:00	9:00	3	3	9:00	11:00	2
32	TLN	0	0	0:00	5:00	1	1	5:00	0:00									
33	TLS	1	1	0:00	5:00	3	1	5:00	7:00	6	4	7:00	9:00	4	4	9:00	12:00	6
34	UIP	0	0	0:00	5:00	1	1	5:00	21:00	0	0	21:00	0:00					
35	URO	0	0	0:00	5:00	1	1	5:00	0:00									

Parsing

- Generally, parsing is about reading and converting the dataset file into computable data structures
  - Dataframes in Pandas (Python), R, Julia
  - Matrices/vectors in Numpy (Python), Matlab…
  - Dictionaries
  - etc.

Parsing

- A LOT of boring work involved:
  - Attention to separators and new lines
  - Eliminate clutter (e.g. empty space, trailing quotes…)
  - Convert types (e.g. string to numeric)
  - Detect problems in file (missing data, impossible values)
- Well, in Pandas, a lot of it is done with one line!

- Warning: it can take **a LOT** of time because it detects date format automatically

```
import pandas as pd
f=pd.read_csv("hubway_trips.csv", parse_dates=['start_date', 'end_date'])
```

Cleaning and converting data

- What is inside the dataframe?

```
f.head()
```

	seq_id	hubway_id	status	duration	start_date	strt_statn	end_date	end_statn	bike_nr	subsc_type	zip_code	birth_date	gender
0	1	8	Closed	9	2011-07-28 10:12:00	23	2011-07-28 10:12:00	23	B00468	Registered	'97217	1976	Male
1	2	9	Closed	220	2011-07-28 10:21:00	23	2011-07-28 10:25:00	23	B00554	Registered	'02215	1966	Male
2	3	10	Closed	56	2011-07-28 10:33:00	23	2011-07-28 10:34:00	23	B00456	Registered	'02108	1943	Male
3	4	11	Closed	64	2011-07-28 10:35:00	23	2011-07-28 10:36:00	23	B00554	Registered	'02116	1981	Female
4	5	12	Closed	12	2011-07-28 10:37:00	23	2011-07-28 10:37:00	23	B00554	Registered	'97214	1983	Female

- Eliminate impossible trips (<1 min; >5hrs)

```
f = f[f.duration > 60]
f = f[f.duration < 5*60*60]
```

- Convert types (numeric to nominal/categorical)*

```
f['start_station']=f['strt_statn'].astype("category")
f['end_station']=f['end_statn'].astype("category")
f['h_id']=f['hubway_id'].astype("category")
```

* Only available after Pandas 0.15

Cleaning and converting data

- Getting the weekdays

```
f['weekday']=[d.weekday() for d in f['start_date']]
```

- Only use registered users (throw away casual users)

```
f=f[f.subsc_type=='Registered']
```

- Well, now we don’t need that column anymore! ;-)

```
f=f.drop('subsc_type', 1)
```

- We can also clean the zipcode strings, if we want

```
f['zip_code']=[str(d).strip("'") for d in f['zip_code']]
```

More on data preparation with Pandas on the notebook…

Creating "new" data

- Examples:
  - Distance between origin and destination station
  - Average speed
  - Type of day (weekend, weekday, holiday)
  - Time of day (e.g. morning, afternoon, peak hour,…)

- We use some of the above in the .ipynb notebooks

Still on data preparation…

- **Data cleaning**: fill in missing values, identify and remove outliers etc.
  - Missing values could be inferred by other available sources/attributes
    - If GPS did not provide data we can check WiFi traces
    - If person does not provide education we can infer it from the profession
    - The mean (of the segment) can be used for the missing values
  - If the shuttle seems 2 hours late, there is probably an error, and it can be identified as an outlier

Still on data preparation…

- **Data integration:** integration of several data sources
  - We have separate files for hubway trips and station location that need to be combined for a more comprehensive analysis (e.g. calculate distances)
  - We could use other data sources (e.g. weather, special events)
  - See examples in the Homework notebook 3. HW Data preparation - Data Fusion.ipynb

Still on data preparation…

- **Data transformation:** Transforming variables for the sake of an efficient/appropriate data analysis
  - **Normalization:** Normalize variables in order to have similar scales. e.g. max-min normalization
  - **Discretization:** Defining categories for continuous variables. It is also a data reduction. e.g. age into age groups (<16, 16-25, 25-40, 40-65, > 65)
  - **Aggregation:** e.g. we have time stamps for hubway rentals, we may want to have counts for 1-hour time intervals.

Descriptive statistics

- Understand the data better

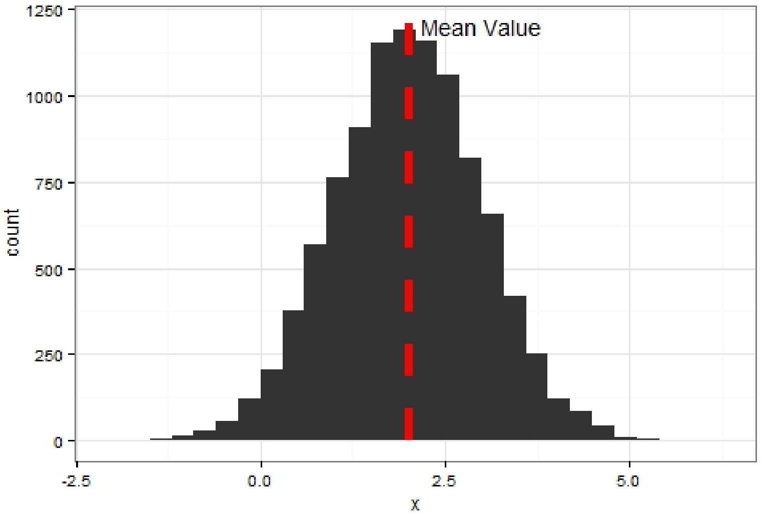
- Key tools for descriptive analysis
  - Central Tendency
    - Mean, Median & Mode
  - Dispersion of data
    - Min & Max, Range, Quartiles, Outliers, Variance, Standard deviation, Histogram and boxplot analysis
  - Covariance & Correlation

Mean

- Convenient way to summarise a variable
- The mean of a sample is the average value calculated as :

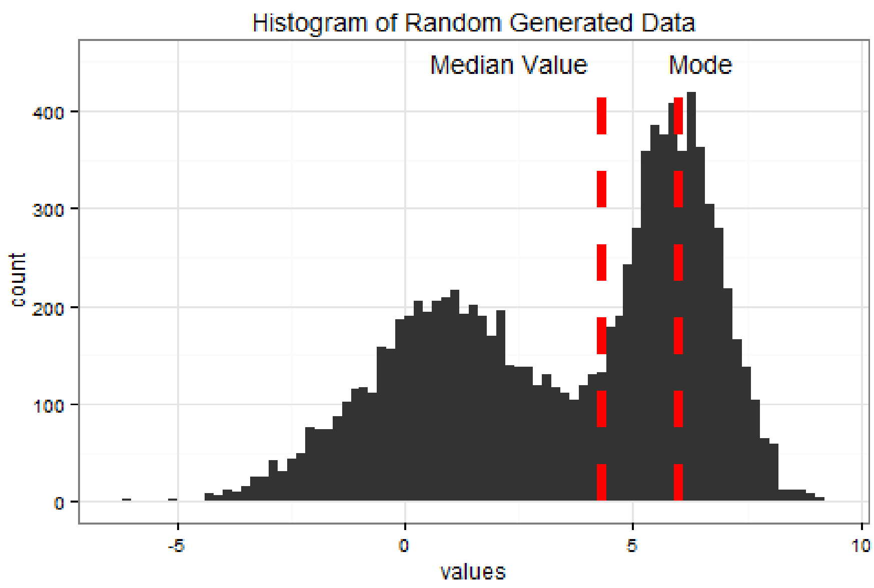
$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

- Mean also can be trimmed:
  extreme values removed



Median and Mode

- Median the value at which 50% of values lie on either side. Can be different to the mean value.
- Mode is the peak value of distribution (i.e. that occurs most frequently). Unimodal, bimodal, trimodal, multimodal



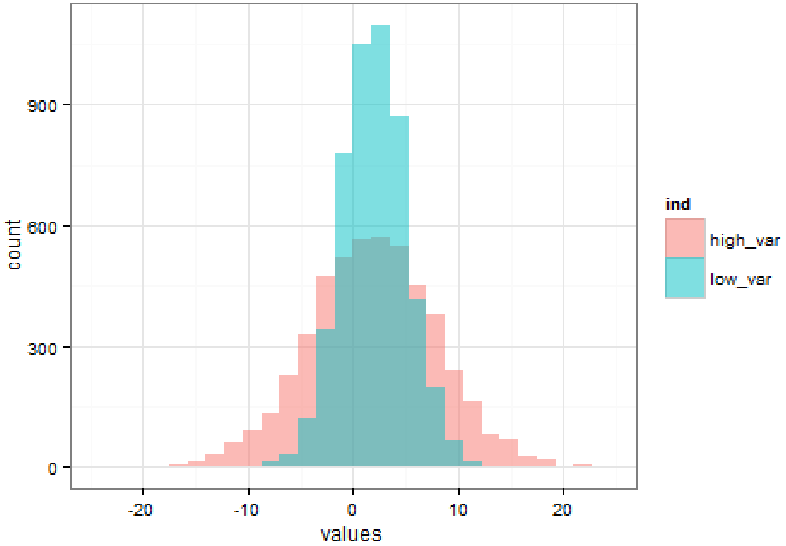
Variance & Standard deviation

- A measure of the spread or variation in a data set:

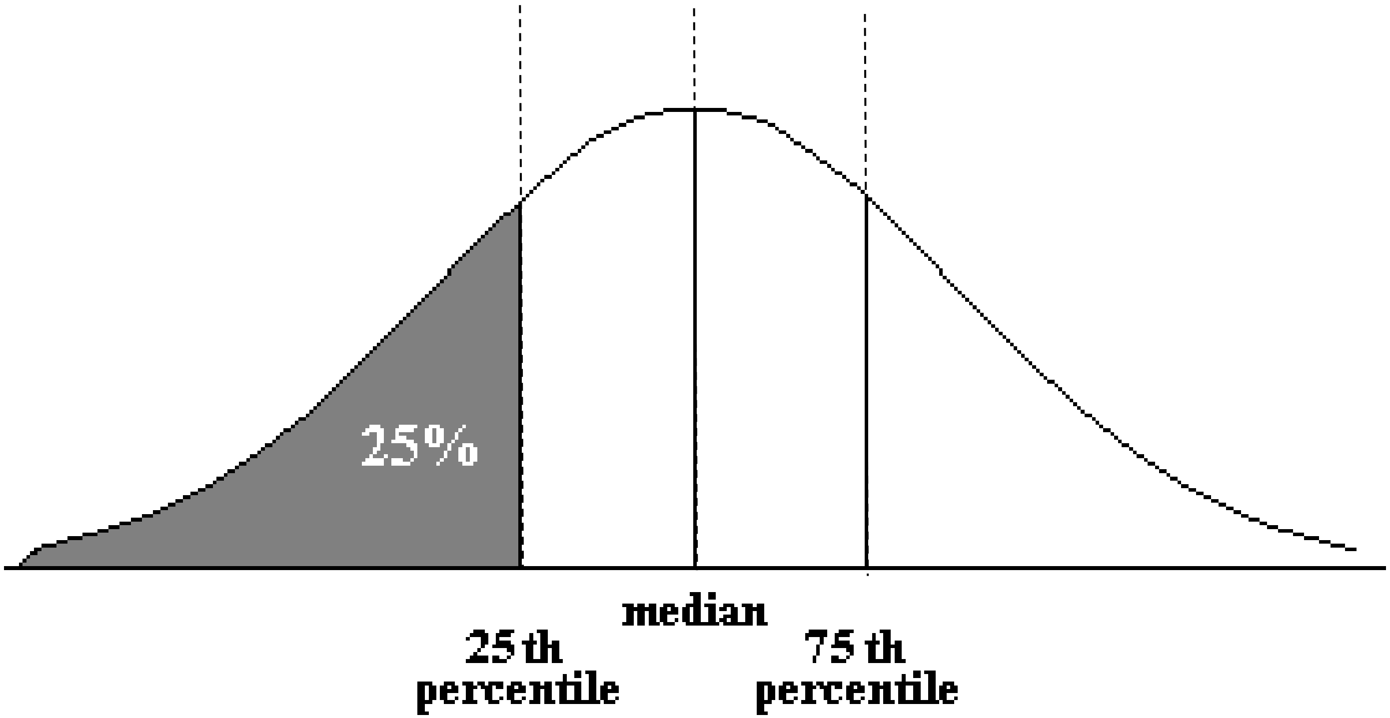
$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

- Variance: s^2 Stdev: s

- Small variance: values are mostly close to the mean.
- High variance: values are very spread out



Dispersion of the data



Lower quartile

Upper quartile

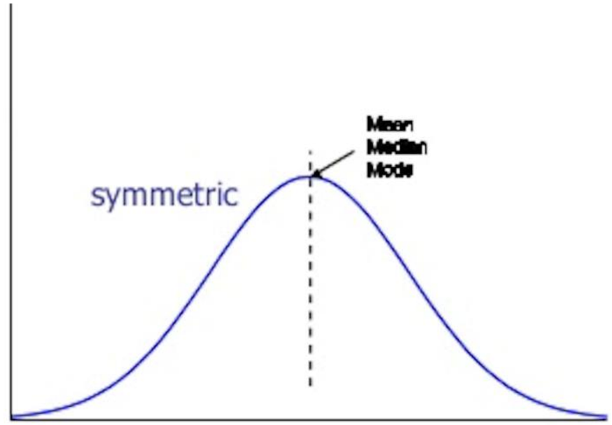
Descriptive stats in Pandas

- Simple!

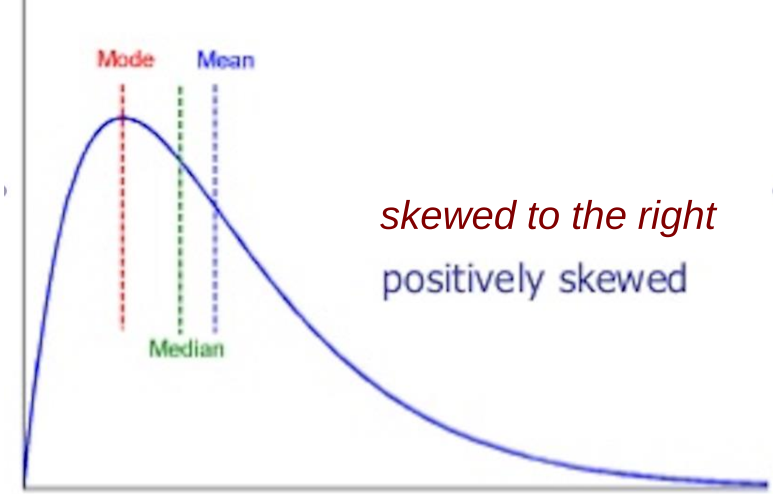
```
f.describe()
```

	seq_id	hubway_id	duration	strt_statn	end_statn	birth_date	weekday
count	1093932.000000	1093932.000000	1093932.000000	1093921.000000	1093912.000000	347072.000000	1093932.000000
mean	822381.635340	922956.155488	693.482774	54.770578	54.636752	1976.278899	2.607186
std	454871.055455	504791.350059	1224.180605	34.416483	34.233878	11.003634	1.843148
min	2.000000	9.000000	61.000000	3.000000	3.000000	1932.000000	0.000000
25%	432207.500000	489286.500000	360.000000	26.000000	26.000000	1969.000000	1.000000
50%	826228.500000	936173.500000	540.000000	48.000000	48.000000	1979.000000	2.000000
75%	1227173.500000	1373691.500000	840.000000	75.000000	75.000000	1985.000000	4.000000
max	1579025.000000	1748022.000000	86280.000000	145.000000	145.000000	1995.000000	6.000000

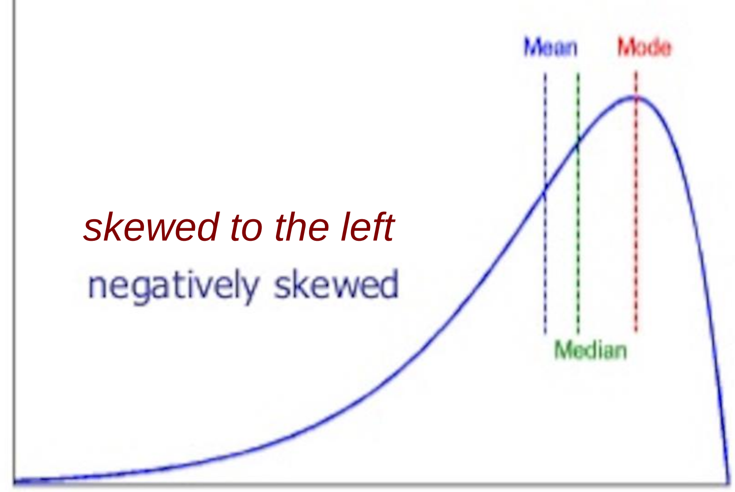
Symmetric vs. Skewed Data



mean > median > mode



mean < median < mode

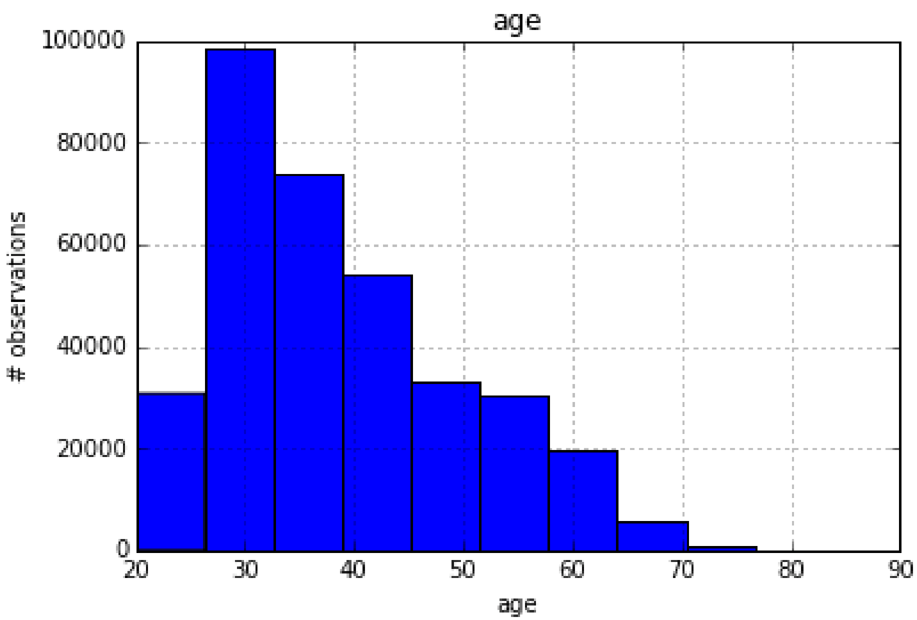


Histograms

- Frequency / number of occurrences

```
f.hist(column="age", bins=10)
xlabel("age")
ylabel("# observations")
```

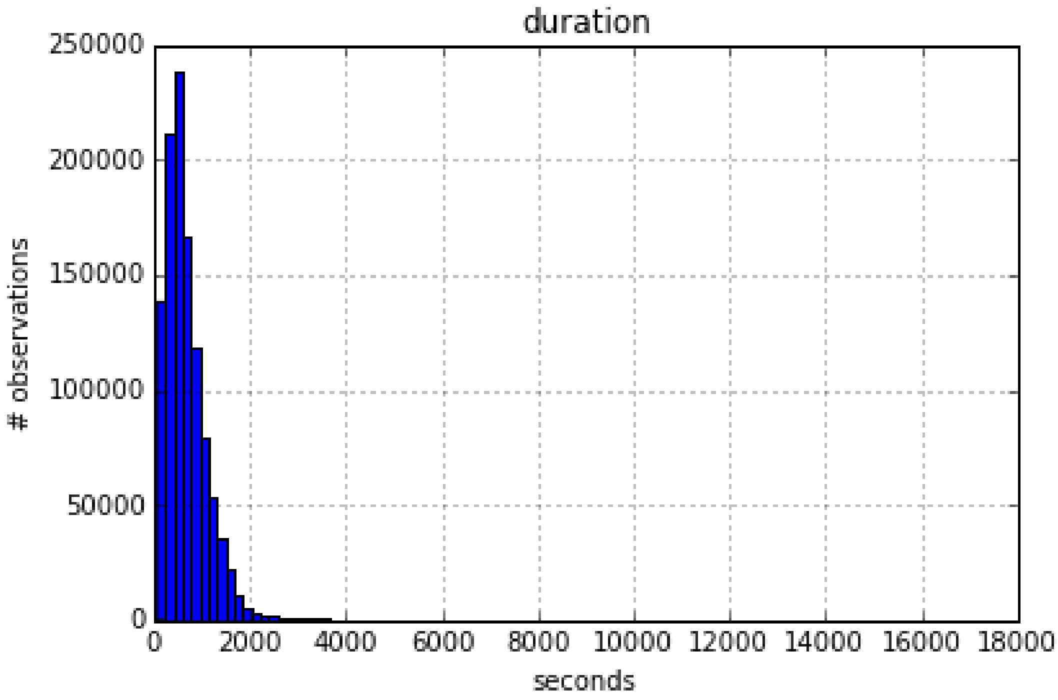
```
<matplotlib.text.Text at 0x111e28610>
```



Histograms

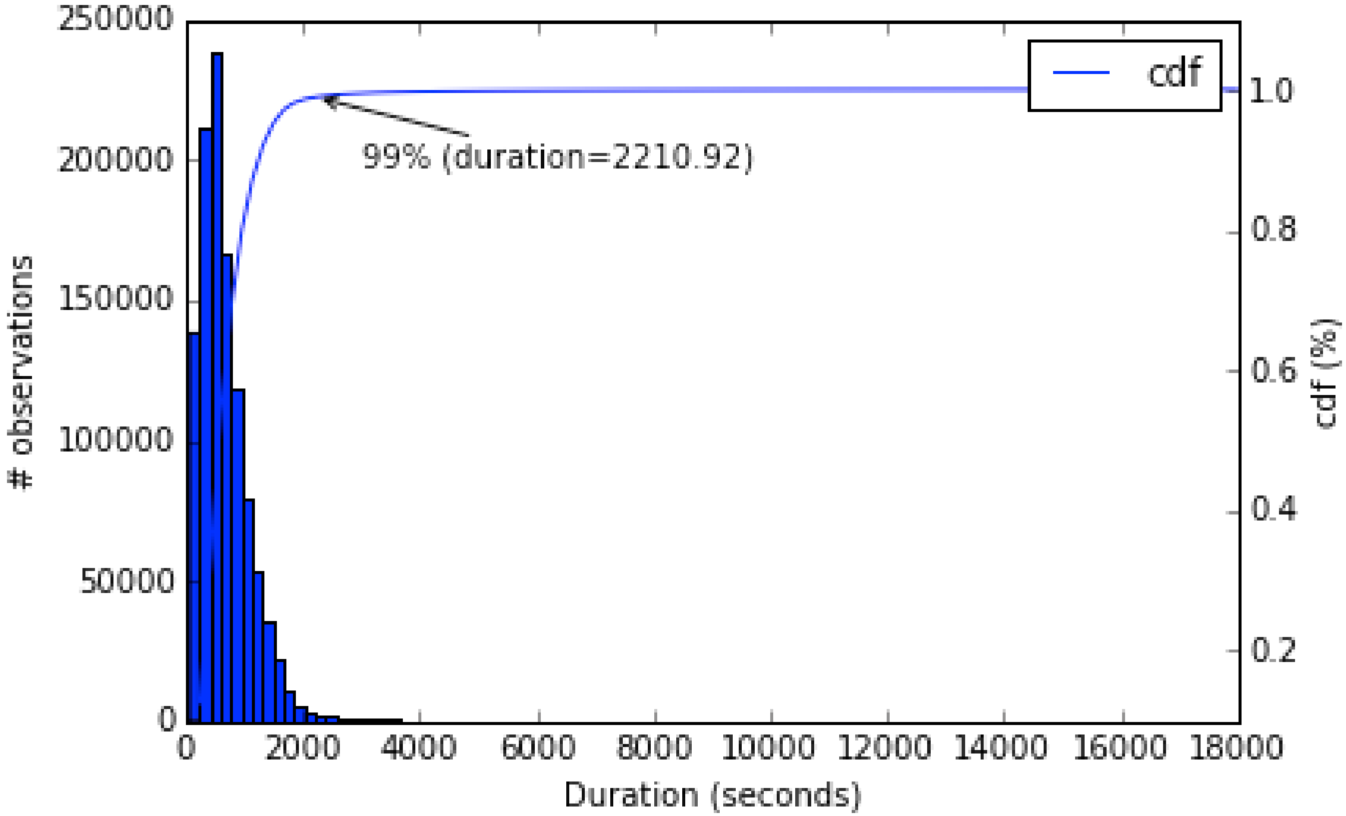
```
f.hist(column='duration', bins=100)
xlabel("seconds")
ylabel("# observations")
```

```
<matplotlib.text.Text at 0x10e6a5ad0>
```



Histograms

- Let’s add a cdf



Histograms

- The code is a little longer (just fyi)

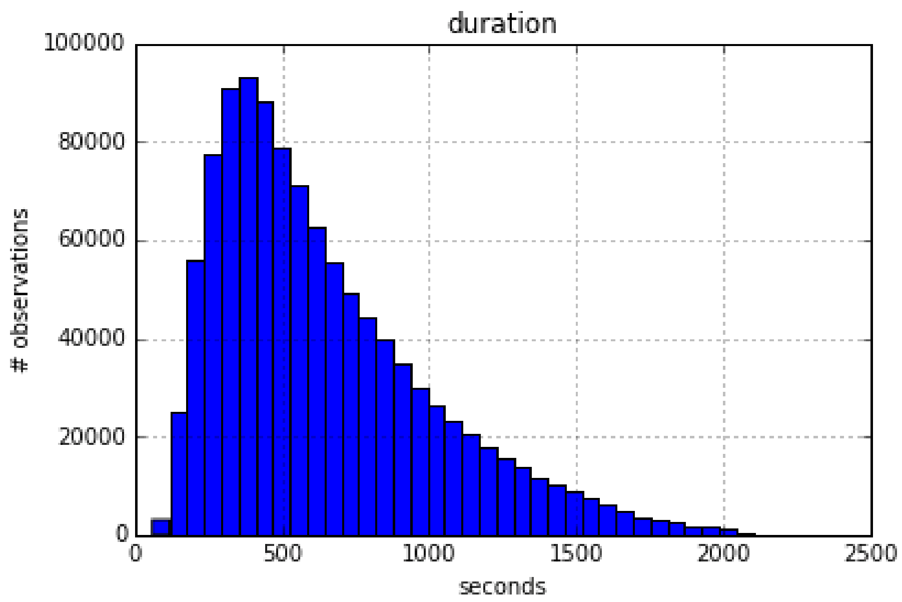
```
import numpy as np

hbins, xx=np.histogram(f['duration'], bins=100)
cm=np.cumsum(hbins/(float(sum(hbins))))
ax = subplot()
ax.hist(list(f['duration']),bins=100, label = 'hist')
tw=twinx()
tw.plot(xx[1:], cm, label = 'cdf')
tw.annotate("99% (duration=2210.92)", xy=(2210.92,0.98936955),
            xytext=(3000,0.89), ha='left',
            arrowprops=dict(arrowstyle='->', shrinkA=0))
tw.legend(loc=1)

ax.set_xlabel("Duration (seconds)")
ax.set_ylabel("# observations")
tw.set_ylabel("cdf (%)")
```

Histograms

We can use histogram to determine data transformation. For example:

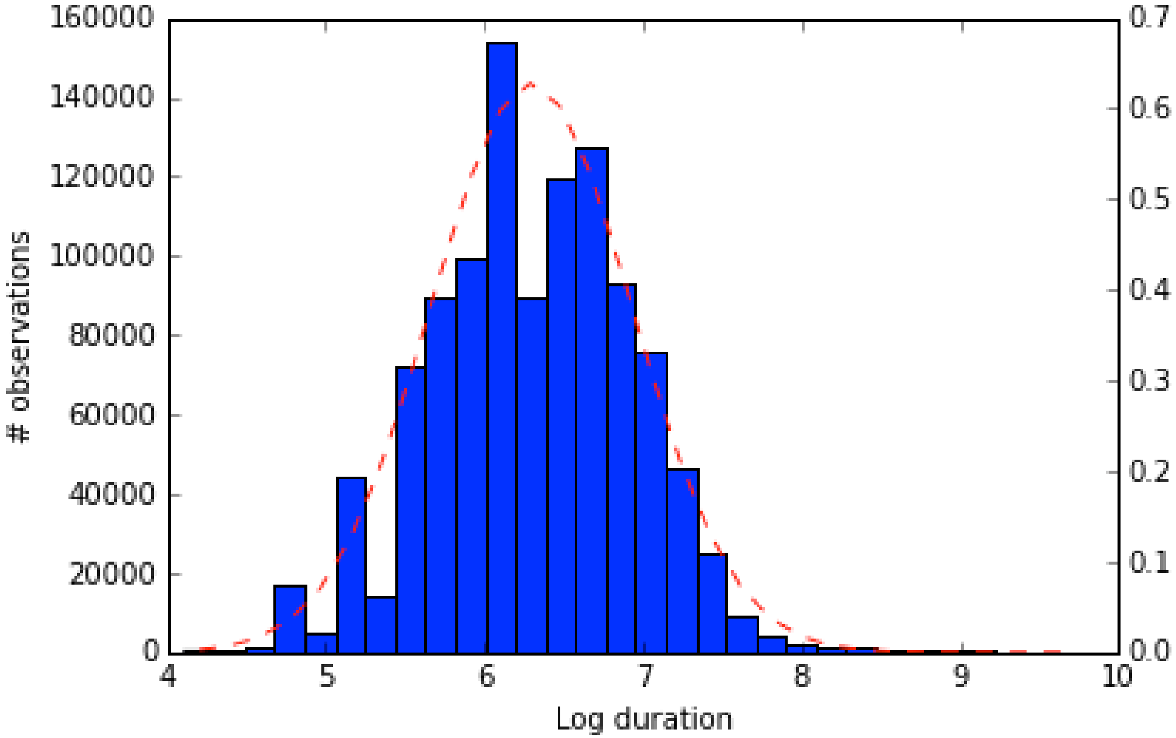
```
#let's get durations below 35 minutes (2100 seconds)
f[f.duration<2100].hist(column="duration", bins=35)
xlabel("seconds")
ylabel("# observations")
```

```
<matplotlib.text.Text at 0x110ada8d0>
```



Histograms

What if we apply log()?



Interesting. Have we found something?

Histograms

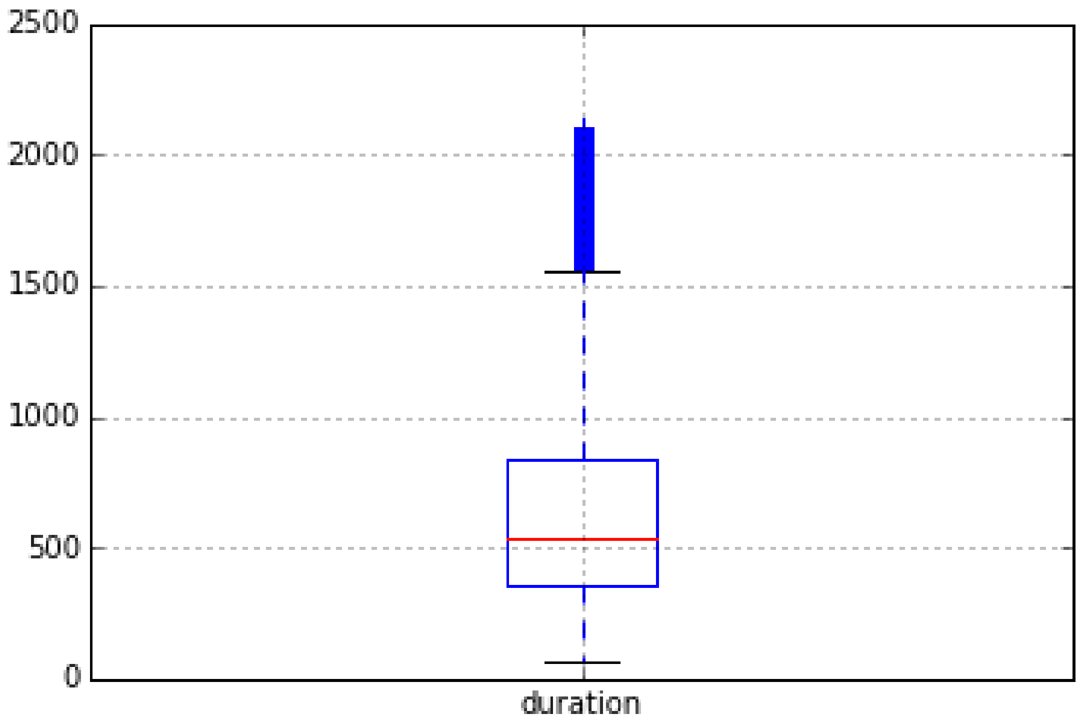
- Again, the code is not that small (fyi)…

```
f['logduration']=log(f['duration'])
bins=30
ax = subplot()
ax.hist(list(f['logduration']),bins, label = 'hist')
tw=twinx()
#ADAPTED FROM matplolib gallery http://matplotlib.org/1.2.1/examples/ap
#
# hist uses np.histogram under the hood to create 'n' and 'bins'.
# np.histogram returns the bin edges, so there will be 50 probability
# density values in n, 51 bin edges in bins and 50 patches.  To get
# everything lined up, we'll compute the bin centers
import matplotlib.mlab as mlab

mu=np.average(f['logduration'])
sigma=np.std(f['logduration'])

bins, xx=np.histogram(f['logduration'], bins)
bincenters = 0.5*(xx[1:]+xx[:-1])
# add a 'best fit' line for the normal PDF
y = mlab.normpdf( bincenters, mu, sigma)
l = tw.plot(bincenters, y, 'r--', linewidth=1)

ax.set_xlabel("Log duration")
ax.set_ylabel("# observations")
```

Boxplots

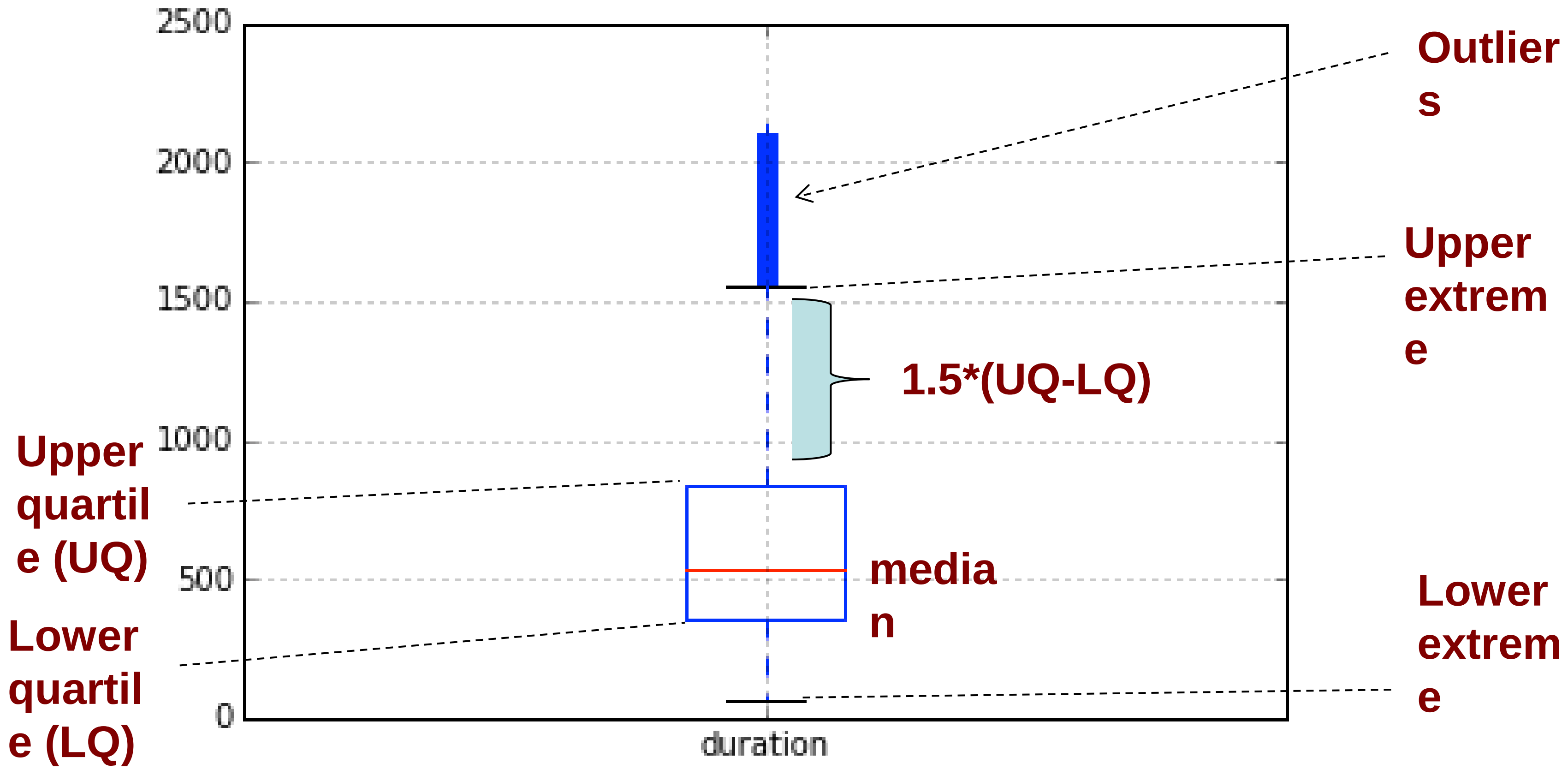
- Condensed visualization of distribution

```
_=f[f.duration<2100].boxplot(column='duration')
```



Boxplots

- Durations

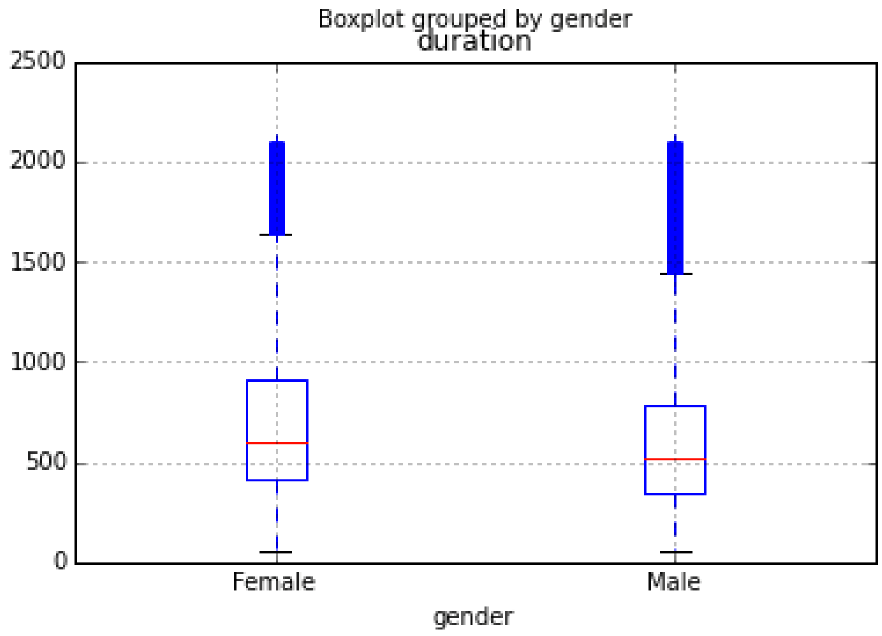


Boxplots

- A simple (research) question: does gender impact duration?

```
f[f.duration<2100].boxplot(column='duration', by="gender")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x117d9f4d0>
```



Boxplots

- We could do hypothesis testing (homework! ;-))…
- …but it's likely a significant difference

```
comparison=pd.DataFrame({'Male':f[f.duration<2100][f.gender=='Male']['duration'],
                         'Female':f[f.duration<2100][f.gender=='Female']['duration']})
comparison.describe()
```

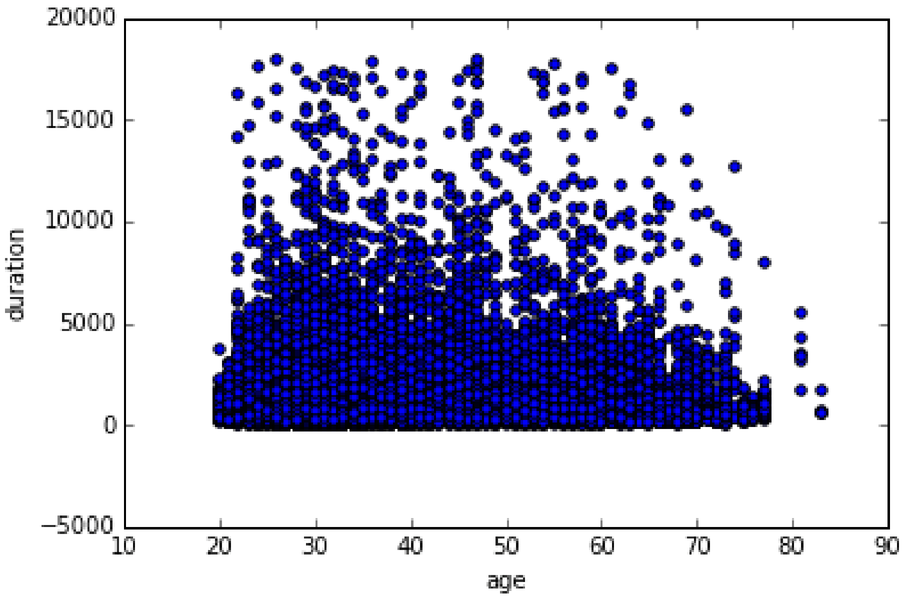
	Female	Male
count	263623.000000	816160.000000
mean	701.700440	609.619980
std	387.482596	366.768158
min	61.000000	61.000000
25%	420.000000	342.000000
50%	600.000000	524.000000
75%	907.000000	783.000000
max	2099.000000	2099.000000

Scatter plots

- Gives an understanding of the data to see the correlation among variables, clusters of points, outliers etc.

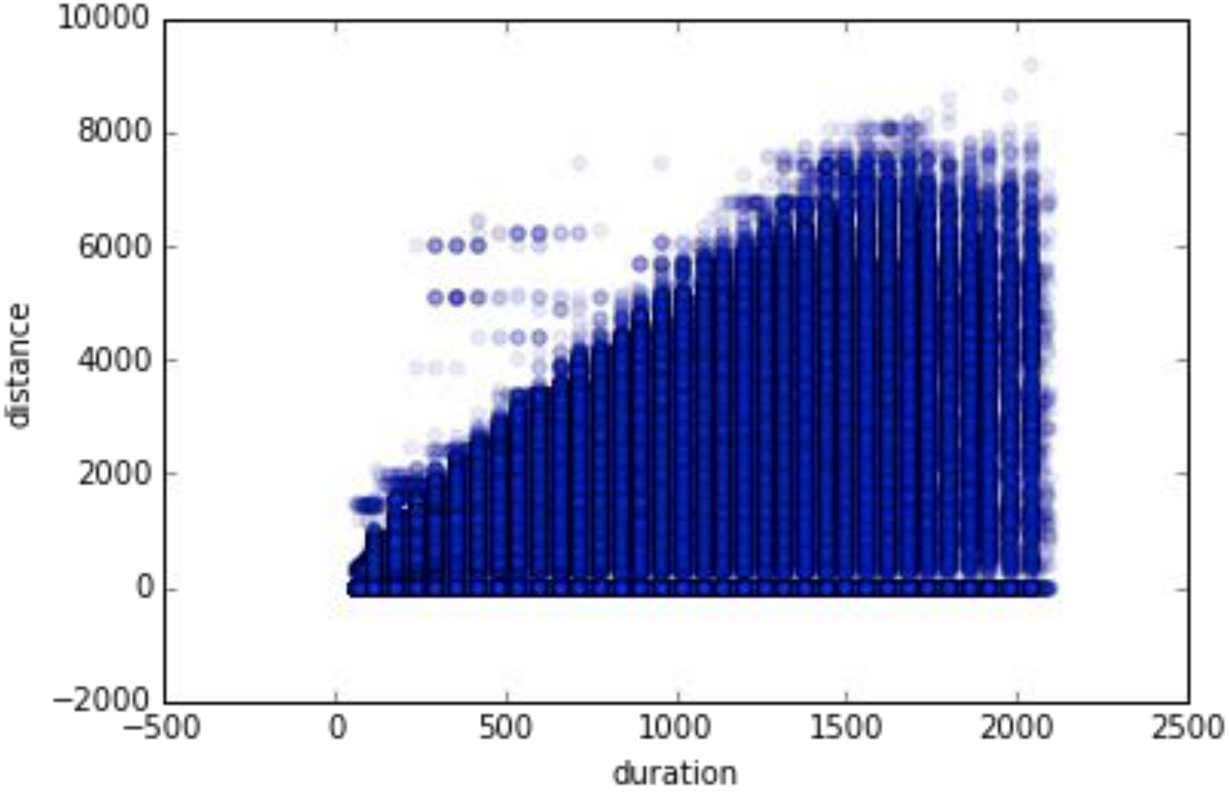
```
scatter(f['age'], f['duration'])
xlabel("age")
ylabel("duration")
```

```
<matplotlib.text.Text at 0x112227b90>
```



Scatter plots

- Relationship between duration and distance?



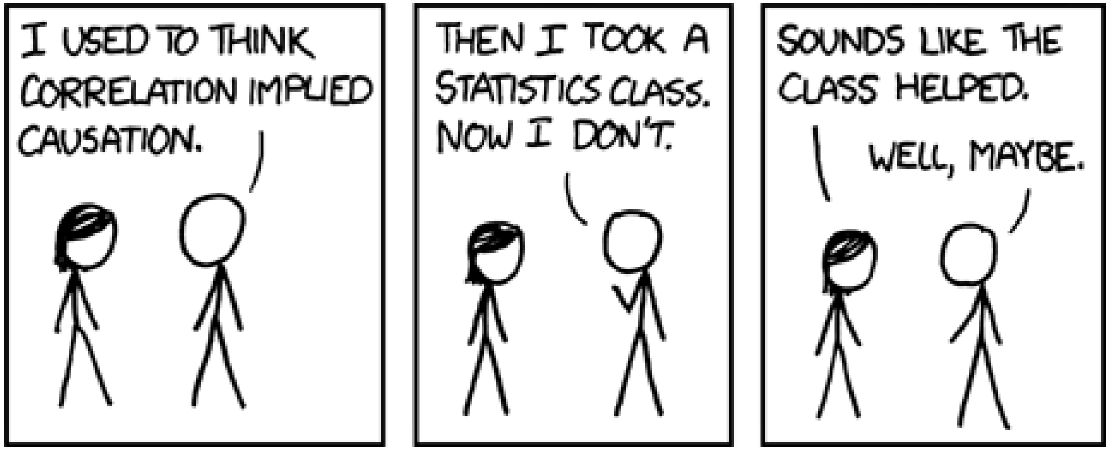
Scatter plots

- Again, some code under the hood (determine distances)

```
import pandas as pd
import geopy.distance

s=pd.read_csv("hubway_stations.csv")

j=pd.DataFrame(s[s.status=='Existing'],columns=['id', 'lat', 'lng'])

dists=[[ido, idd, geopy.distance.vincenty((ox, oy), (dx, dy)).meters]
        for (_, ido, ox, oy) in j.itertuples()
        for (_, idd, dx, dy) in j.itertuples()]
dist=pd.DataFrame(dists, columns=['o', 'd', 'dist'])

newf=pd.merge(f[f.duration<2100], dists, left_index=True,
              left_on=["strt_statn", "end_statn"],
              right_on=["o", "d"]).loc[:, ["seq_id","strt_statn",
                                          "end_statn","duration", "dist"]]
scatter(newf['duration'], newf['dist'], alpha=0.05)
xlabel("duration")
ylabel("distance")
```

Covariance and Correlation

- Covariance - a measure of how much variables in a data set change *together.*
- Correlation - normalised version of the covariance.
- Gives a measure of the dependence between two variables.



Correlation and covariance

- Covariance

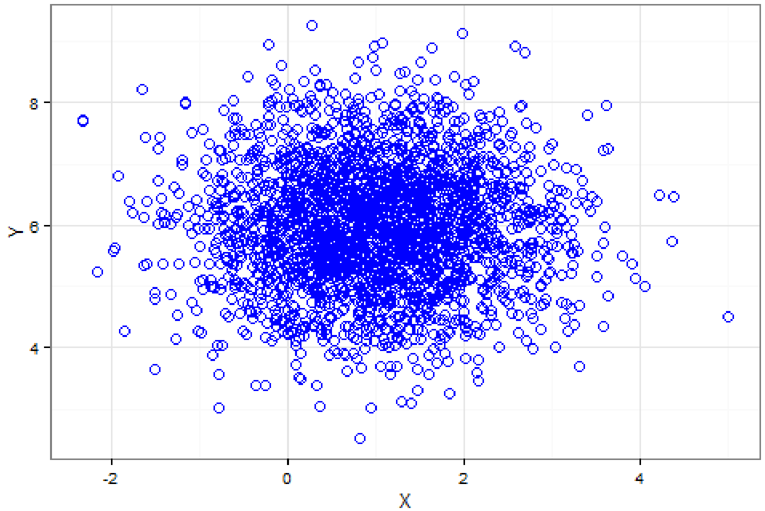
$$cov(x,y) = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})$$

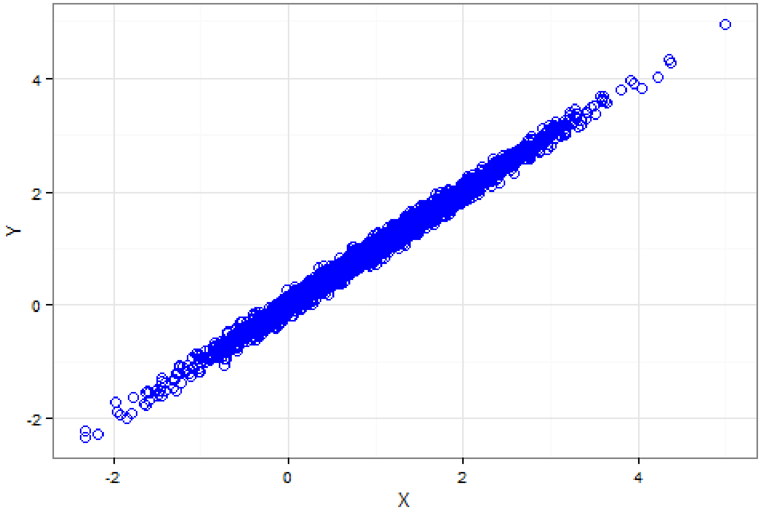
- Correlation

$$\mathrm{corr}(x,y) = \frac{\mathrm{cov}(x,y)}{s_x s_y}$$

Correlation

- What does the *correlation coefficient tell?*
  - Strong negative correlation: close to -1
  - Strong positive correlation: close to 1
  - No correlation at all: close to 0
- If variables are perfectly correlated, knowing one variable allows you to predict the other variable.





Correlation

- Python Pandas has a very simple way to see correlation across many variables at the same time!

```
newf.corr()
```

	seq_id	strt_statn	end_statn	duration	dist	weekday	age
seq_id	1.000000	0.339710	0.340009	-0.026937	0.067356	-0.022592	-0.063192
strt_statn	0.339710	1.000000	0.321706	0.027431	0.074310	0.003746	-0.033738
end_statn	0.340009	0.321706	1.000000	0.029036	0.073916	0.001634	-0.030228
duration	-0.026937	0.027431	0.029036	1.000000	0.598594	0.129483	0.030143
dist	0.067356	0.074310	0.073916	0.598594	1.000000	-0.006546	-0.056556
weekday	-0.022592	0.003746	0.001634	0.129483	-0.006546	1.000000	-0.064327
age	-0.063192	-0.033738	-0.030228	0.030143	-0.056556	-0.064327	1.000000