## Basic data scientist skills:

1. knows which questions to ask
2. can interpret the data well
3. understand the structure of the data
4. work in teams

## Three sources for data:

1. files
2. database
3. API (Application Programming Interface)

Common data formats:

1. csv (head row and entry rows)
2. XML (like HTML)
3. JSON (in the style of "{ }" like python dictionary)

## Dealing with missing data

1. partial deletion (listwise deletion: delete and exclude from any analysis; pairwise deletion)
2. imputation (fill in with mean, estimate using linear regression)
3. have negative effects

Significance Test

1. using our data, can we disapprove an assumption with a pre-defined level of confidence?
2. **t-test: accept or reject a null hypothesis**
3. **null hypothesis: a statement we are trying to disapprove by running our test**

## t-test

1. specified in terms of a test statistics
2. two-sample t-test: determine means of two samples are equal or not

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1^2}{N_1} + \frac{\sigma_2^2}{N_2}}}$$

The number of degrees of freedom: $\nu \approx \frac{\left(\frac{\sigma_1^2}{N_1} + \frac{\sigma_2^2}{N_2}\right)^2}{\frac{\sigma_1^4}{N_1^2 \nu_1} + \frac{\sigma_2^4}{N_2^2 \nu_2}}$

where $\nu_i = N_i - 1$.

Calculate $t$ + calculate $\nu$ -> estimate $p$ value

$p$ **value:** A P-value is the probability of observing data as or more extreme in favor of the alternative than was actually obtained, where the probability is calculated assuming that the null hypothesis is true.
(if p < p_critical, reject; otherwise, cannot reject)

The way to interpret P-values is as follows. If the P-value is small, then either $H_0$ is true and we have observed a rare event or $H_0$ is false (or possibly the null model is incorrect).

one-sided p-value is half of the two-sided p-value.

1. p/2 < p_critical, and t > 0 -> reject
2. p/2 > p_critical, and t < 0 -> reject

**Non-normal data**

1. **Shapiro-Wilk** test: determine a distribution is normal or not (*W, p = scipy.stats.shapiro (data)*)
2. **Mann-Whitney U** test: test null hypothesis that two populations are the same (*U, p = scipy.stats.mannwhitneyu (x, y)*)

**Statistics**: draw conclusions and make analysis from existing data
**Machine learning**: making predictions

K means + PCA

**Coefficient of determination:** $R^2 = 1 - \frac{\sum_n (y_i - f_i)^2}{\sum_n (y_i - avg(y))^2}$

*Effective Communication:*

1. clarity
2. precision
3. efficiency

*Four important ingredients:*

1. visual cues/visual encoding
2. coordinate systems
3. scale/Data types
4. context

**ggplot**: grammar of graphics, adding layers to the figure
*ggplot (pandas_data_frame, aes (xvar, yvar)): xvar and yvar are columns*

*print ggplot + geom_point (color = 'coral') -> scatter plot*
*+ geom_line (color = 'coral') -> connect with lines*
*+ ggtitle ('title') + xlab ('x_label') + ylab ('y_label')*

***time-series data***: data collected via repeated measurements over time

loess: weighted regression

**Example:** geom_smooth(data = data[1:n, ], method="loess", span=spanv, fullrange=FALSE, se=TRUE, na.rm=TRUE)

**MapReduce**: split a large job into several smaller chunks that each fits into one machine and occurs simultaneously (these machines do not communicate) -> a parallel programming model

1. Mapper
2. Reducer

Procedure:

1. a collection of documents or records
2. send these documents in a distributed way to many mappers
3. each mapper performs the same mapping their respective documents and produces a series of intermediate key value pairs
4. shuffle these intermediate results and send all key value pairs of the same key to the same reducer for processing
5. each reducer can produce one final key value pair for each key

Only use MapReduce when your data is truly big!!

Example:

1. mapper: take into a document, return various intermediate key value pairs (word (key), # of word)
   ***def mapper ():***
2. the key value pairs will be shuffled to ensure every key value pair with the same key ends up in the same reducer
   ***def reducer ():***

**Hadoop**: MapReduce programming model with distributed file systems

1. **Hive**: run MapReduce jobs through SQL-like querying language (called Hive Query language)
2. **Pig**: more explicit about the execution of data processing, split data pipeline