

# Data Analysis

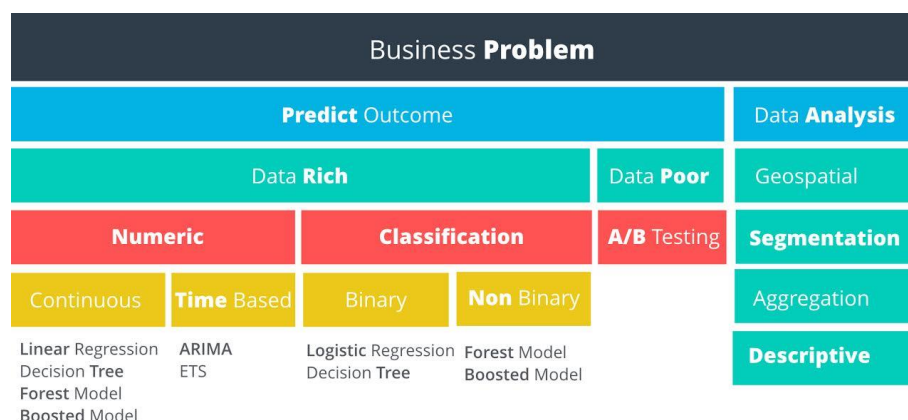
## P1 – Statistics

### Data Analysis Process Phases:

1. Question
2. Wrangling
  - a. Data acquisition
  - b. Data cleaning
3. Explore
  - a. Build intuition
  - b. Find patterns
4. Drawing conclusions or making predictions
  - a. Usually requires statistics or
  - b. Machine learning
5. Communication
  - a. Blog post, paper, email, PowerPoint, conversation
  - b. Data visualization is useful

### CRISP Framework (CROSS Industry Standard Process for Data Mining)

1. Business Issue Understanding
2. Data Understanding
3. Data Preparation
4. Analysis / Modelling
5. Validation
6. Presentation / Visualization



### Non-Predictive Analysis

- Geospatial (location based data e.g. geographic information)
- Segmentation (process of grouping data together e.g. demographic information)
- Aggregation (calculating a value across a group or dimension e.g. “slice and dice” information)
- Descriptive (simple summaries of a data sample e.g. mean, median, mode, standard deviation, interquartile range)

## **Predictive Analysis** (predict a future outcome)

- **Data Poor**

- If there is not sufficient usable data to solve the problem, then we need to set up an experiment to help us get the data we need, which is usually referred to as an **A/B Test**.

- **Data Rich**

- Numeric Analysis
  - Regression Models
    - Continuous (Linear Regression, Decision Tree, Forest Model, Boosted Model)
    - Time-Based (Arima, ETS)
    - Count (not common in business)
- Non-Numeric (Classification) Analysis
  - Classification Models
    - Binary (Logistic Regression, Decision Tree e.g. yes or no)
    - Non-Binary (Forest Model, Boosted Model e.g. small, medium, large)

- **Linear Regression**

- $y = ax + b$
- $a \rightarrow \text{SLOPE}(y, x)$
- $b \rightarrow \text{INTERCEPT}(y, x)$
- $R \rightarrow \text{CORREL}(y, x) \rightarrow$  The closer  $r$  is to  $+1$  or  $-1$ , the higher the correlation between  $x$  and  $y$
- R-squared  $\rightarrow \text{RSQ}(y, x) \rightarrow$  close to 1 would mean that nearly all variance in the target variable is explained by the model  $\rightarrow > 0.7$  strong model,  $> 0.5$  pretty good,  $< 0.3$  no useful.

- **Multiple Linear Regression**

- $y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \dots$
- Analysis ToolPak Add-In Excel  $\rightarrow$  Data Analysis  $\rightarrow$  Regression
- The adjusted R-squared value should be used

- **Linear Regression with Categorical Variables**

- A much better way of using categorical variables in regression is to use “dummy variables”. A dummy variable can only take on two values, generally 0 or 1. You would add one dummy variable for one less than the number of unique values in the categorical variable. So, if the variable is binary, you'd add one dummy. If there are four categories, you'd add three dummy variables.
- The **P value** is the probability that observed results occurred by chance, and that there is no actual relationship between the predictor and target variable. In other words, the p-value is the probability that the coefficient is zero. The lower the p-value the higher the probability that a relationship exists between the predictor and target variable. If the p-value is high, we should not rely on the coefficient estimate. When a predictor variable has a p-value below 0.05, the relationship between it and the target variable is considered statistically significant.

## Valid research (survey)

- Good sample size
- Representative sample
- Sound methodology

## Constructs – Operational Definition

- Population parameters (such as  $\mu$ ) are values that describe the entire population. Sample statistics (such as  $\bar{x}$ ) are values that describe our sample; we use statistics to estimate the population parameters.
- Independent or predictor variable
- Dependent variable or outcome
- Extraneous factors or lurking variables
- Correlation does not imply causation!
- Double Blind experiment
- Show relationships → Observational studies or surveys
- Show causation → Controlled experiment

## Visualizing Data

- Frequency  $f$ , Relative Frequency (0, 1) or Proportion  $p$  (0-100 %)
- Total Sum  $\Sigma$
- Measures of centre
  - Mode: value where  $f$  is highest, most common value, occurs on the X-axis. Can describe any type of data (numerical or categorical). It isn't the same in different samples of the dataset.
    - no mode, uniform distributions
    - one mode distributions
    - multi modal mode distributions
  - Median (value in the middle) more robust than Mean when we have highly skewed distributions
  - Mean or Average take all values into account unlike the Mode
- Bin Size or Interval Size, Smaller bin size → less information, bigger bin size → larger frequency
  - Infinitely small bin size → the distribution shape is lost
    - We can model a distribution with a theoretical continuous distribution described by an equation. In normal theoretical distributions Mean = Median = Mode
- Histogram: x-axis is numerical/quantitative vs. Bar Graph: x-axis is categorical/qualitative
- $n$  sample size vs.  $N$  population size
- Sometimes we cut off the upper and lower 25 %
  - Inter-Quartile Range = IQR =  $Q_3 - Q_1$
  - Outlier considered when it is  $< Q_1 - 1.5(IQR)$  or  $> Q_3 + 1.5(IQR)$
- Visualize outliers, IQR, median, min and max with Match Boxplots
- Variability = average distance between each value and the mean (Mean of Deviation from Mean  $x_i - \bar{x}$ )
  - Because Average Deviation → 0, We consider Average Absolute Deviation or Squared Deviation
- Sum of Squares SS

- Variance = Average Squared Deviation (Sum of Squared Deviations divided by n)
- Standard Deviation  $\sigma$  = Square Root of Variance = most common measure of spread (Variability)
  - 68 % of data between  $1\sigma$  and 95 % between  $2\sigma$
- In general, samples underestimate the amount of variability in a population, because samples tend to be values in the middle of the population
  - Bessel's Correction to correct this, dividing Variance with n-1 instead of n
  - Sample Standard Deviation s
- Standardizing the distribution → Find the z-score, the number of Standard Deviations each value x is from the Mean  $\mu$ :  $z = \frac{x-\mu}{\sigma}$ 
  - The new Mean of the standardizing distribution is 0 and the Standard Deviation  $\sigma$  is 1
- We can convert any normal distribution to standard normal distribution and then to a new normal distribution again with a new Mean and Standard Deviation.

## Normal Distribution

- Probability Density Function (PDF) theoretically models the Normal Distribution curve.
- Absolute Frequency → Relative Frequency = Probability
- X-axis is horizontal asymptote
- Area = Probability of randomly selecting less than x = proportion in sample/population with scores less than x.
- z-table → gives probability p from z-scores for a standard normal distribution

## Sampling Distributions

- Sampling Distribution = Distribution of Sample Means
- Mean of population = Mean of all possible sample means
- It is a Normal Distribution
- Ratio:  $\sigma$  of Population / SE of Sample Means = square root of our sample size →  $\frac{\sigma}{SE} = \sqrt{n}$
- So, we can find the Standard Deviation of Sampling Distribution  $SE = \frac{\sigma}{\sqrt{n}}$
- This is called Central Limit Theorem
  - From any population of any shape we can take samples – assuming their size is large enough – and plot the Sampling Distribution → we will get a Normal Distribution with a Standard Deviation equals to the Standard Deviation  $\sigma$  of the population divided by the square root of the Sample size n and this is called Standard Error SE.
- Larger Sample size → skinnier Distribution
  - As the Sample size n increases, the Standard Error decreases
  - Surer that the Mean of Sampling Distribution  $\approx$  Mean of Population
  - Prove some kind of relationship
- The Mean of Sampling Distribution,  $M \approx$  Mean of Population,  $\mu$

## Estimation

- Point Estimate = Mean of Sample
- Margin of Error (half the width of CI),  $2 SE = \frac{2 \cdot \sigma}{\sqrt{n}}$
- 95% of Sample Means will be within 2 SE or exactly 1.96 SE (from z-table) in a Sampling Distribution
- Critical Values of  $z = \pm 1.96$
- Confidence Intervals CI with 95%
  - Interval Estimate:  $\bar{x} \pm (z - score) \cdot \left(\frac{\sigma}{\sqrt{n}}\right)$
  - Bigger Sample  $\rightarrow$  Smaller CI (more precise estimates)
  - A "treatment effect" occurs when the intervention affects the population mean. When the sample mean is far on the tails of the sampling distribution, and therefore unlikely to have occurred by chance, there is evidence for a treatment effect.

## Hypothesis Testing

### Z-Test (know population parameters)

- Critical Region and z-critical value
- One-Tailed
  - $\alpha$ -Levels: 0.05 (5%),  $z^* = 1.65$  / 0.01 (1%),  $z^* = 2.32$  / 0.001 (0.1%),  $z^* = 3.08$ 
    - If the probability of getting a sample mean is less than  $\alpha$ , it is "unlikely" to occur
    - If a sample mean has a z-score greater than  $z^*$ , it is "unlikely" to occur
- Significance
  - $z = \frac{\bar{x} - \mu}{(\sigma/\sqrt{n})}$ , if  $z > z^*$  then  $\bar{x}$  is significant at  $p < \alpha$  (didn't occur by chance, something is going on)
- Two-Tailed (more conservative  $\rightarrow$  cannot miss the opposite direction if we choose one-tailed)
  - $\alpha$ -Levels: 0.05 (5%),  $z^* = \pm 1.96$  / 0.01 (1%),  $z^* = \pm 2.57$  / 0.001 (0.1%),  $z^* = \pm 3.32$
- Hypotheses
  - "I" is for Intervention
  - Null,  $H_0: \mu = \mu_I$ , we can only obtain evidence to reject the null hypothesis (states no change)
  - Alternative,  $H_A: \mu < \mu_I$  or  $\mu > \mu_I$  or  $\mu \neq \mu_I$
- Decision Errors

Udacity.com

		Decision	
		Reject $H_0$	Retain $H_0$
State of the world	$H_0$ true	WRONG Type I error	CORRECT
	$H_0$ false	CORRECT	WRONG Type II error

## T-Test (not know population parameters)

- T-distribution: more prone to error, more spread out and thicker in the tails
- Defined by their degrees of freedom: e.g. the last option is always forced
- Effective Sample Size,  $n-1$  (independent pieces of information)
- As degrees of freedom increases, the t-distribution better approximates the normal distribution
- T-Table, t critical values, column headers are  $\alpha$  levels
- The further the value of  $\bar{x}$  from  $\mu_0$  in either direction the stronger the evidence that  $\mu \neq \mu_0$
- One Sample t-test
  - Standard Error of the mean:  $SE = \frac{s}{\sqrt{n}}$
  - $t_{statistic} = \frac{\bar{x} - \mu_0}{(s/\sqrt{n})}$
  - $H_0: \mu = \mu_0$ , reject null if t-statistic is inside critical region (p-value <  $\alpha$  level)
  - $H_A: \mu < \mu_0$  or  $\mu > \mu_0$  or  $\mu \neq \mu_0$
- P-value: probability of getting t-value
- Cohen's d:  $d = \frac{\bar{x} - \mu_0}{s}$ 
  - Standardized mean difference that measures the distance between means in standardized units
  - The larger d is, the further  $\bar{x}$  is from  $\mu_0$
- Confidence Interval CI, e.g. 100% - 5% (from  $\alpha$  level = 0.05) = 95% CI
  - Interval Estimate:  $\bar{x} \pm (t_{critical}) \cdot \left(\frac{s}{\sqrt{n}}\right)$
- Unlike a z-critical value, a t-critical value will change when you increase the sample size.

## Dependent sample's design, t-test for paired samples

- Within-subject designs
  - Two conditions
  - Pre-test, post-test
    - Same variable – same sample before and after treatment
    - $H_0: \mu_{pre} = \mu_{post}$
  - Growth over time (longitudinal study)
    - $H_0: \mu_{time1} = \mu_{time2}$
  - $D_i = x_i - y_i$ , take the difference of the paired samples
  - $(D_i - \bar{D})^2 \rightarrow SS = \sum (D_i - \bar{D})^2 \rightarrow s_D = \sqrt{\frac{\sum (D_i - \bar{D})^2}{n-1}}$  or  $s_D = \sqrt{s_1^2 + s_2^2}$
  - $t_{statistic} = \frac{\bar{D} - 0}{(s_D/\sqrt{n})}$  or  $t_{statistic} = \frac{\mu_1 - \mu_2}{(s_D/\sqrt{n})}$
  - $CI = \bar{D} \pm t_{critical} \cdot \left(\frac{s_D}{\sqrt{n}}\right)$ , t-critical is taken from the t-table for two-tailed test

## Effect Size Measure Types

- Mean difference,  $\bar{x} - \mu$ 
  - It is good for variables with easy to understand meanings
- Standardized difference
  - Cohen's d – SD units
- Correlation measures
  - Coefficient of Determination:  $r^2$  – proportion (%) of variation in one variable that is related to (“explained by”) another variable → not related at  $0.00 < r^2 < 1.00$  perfectly related
    - $r^2 = \frac{t_{statistic}^2}{t_{statistic}^2 + df}$

## Statistical Significance

- Rejected the null
- Results are not likely due to chance (sampling error)

## Meaningfulness of results

- What was measured?
  - Variables with practical, social, or theoretical importance
- Effect size
- Can we rule out random chance?
- Can we rule out alternative explanations (lurking variables)?

## Results Sections

1. Descriptive Statistics (M, SD) in
  - Text – Graphs – Tables
2. Inferential Statistics
  - Hypothesis Test
    - kind of test, e.g. one-sample t-test
    - test statistic, e.g. value of t
    - df
    - p-value
    - direction of test, e.g. one or two tail test
    - $\alpha$  level
    - APA style →  $t(df) = x.xx, p = .xx$ , direction
  - Confidence Interval
    - Confidence level, e.g. 95%
    - Lower limit

- Upper limit
- CI on what?
- APA style → Confidence interval on the mean difference; 95% CI = (4, 6)

### 3. Effect size measures

- Cohen's d
- $r^2$
- APA style →  $d = x.xx$ ,  $r^2 = .xx$  (we never write the leading 0 for proportions < 1)

### Example (one-sample t-test)

$\mu = \$151$ ; dependent variable = \$ spent per week on food; treatment = cost-saving program;  $n = 25$

null = the program did not change the cost of food; alternative = the program reduced the cost of food

this is a one-tailed test in negative direction

$$H_0: \mu_{\text{program}} \geq 151 \quad H_A: \mu_{\text{program}} < 151 \quad df = n - 1 = 24$$

$$t_{\text{critical}} = -1.711 \text{ for } \alpha = .05$$

$$SE = \frac{s}{\sqrt{n}} = \frac{50}{\sqrt{25}} = 10.00$$

$$\text{Mean difference, } \bar{x} - \mu = 126 - 151 = -25$$

$$t_{\text{statistic}} = \frac{\bar{x} - \mu}{(s/\sqrt{n})} = \frac{-25}{10.00} = -2.50 \quad \rightarrow \quad t_{\text{statistic}} \text{ falls in the critical region} \quad \rightarrow \quad p < .05$$

So, we can reject the null hypothesis and say that these results are statistically significant.

Are the results meaningful?

It depends because saving \$25 per week depends on how much money the people of our study make.

$$\text{Cohen's } d = \frac{\bar{x} - \mu}{s} = \frac{-25}{50} = -0.50 \quad \rightarrow \quad \text{these two means are half a standard deviation apart}$$

$$r^2 = \frac{t_{\text{statistic}}^2}{t_{\text{statistic}}^2 + df} = \frac{(-2.50)^2}{(-2.50)^2 + 24} = .21$$

21% of the differences in food prices for the sample of 25 people are due to the cost saving program.

$$\text{Margin of Error} = t_{\text{critical}} \cdot SE = 2.064 \cdot 10.00 = 20.64$$

$$CI = \bar{x} \pm t_{\text{critical}} \cdot \left( \frac{s}{\sqrt{n}} \right) = 126 \pm 20.64 \quad \rightarrow \quad 95\%CI = (105.36, 146.64)$$

If we implemented this cost saving program for all members, they would likely spend on average per week between \$105 and \$147.



## Independent samples

Dependent Samples	Independent Samples
Two conditions	Experimental
Longitudinal	Observational
Pre-test, post-test	-
<b>Advantages</b>	<b>Disadvantages</b>
Controls for individual differences	The opposite
Use fewer subjects	
Cost-effective	
Less time-consuming	
Less expensive	
<b>Disadvantages</b>	<b>Advantages</b>
Carry-over effects	The opposite
Order may influence results	

- $SD = \sqrt{S_1^2 + S_2^2}$
- Standard Error:  $SE = \sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}$ , assumes samples are approximately the same size
- Pooled Variance:  $S_p^2 = \frac{SS_1 + SS_2}{df_1 + df_2} \rightarrow \text{corrected } SE = \sqrt{\frac{S_p^2}{n_1} + \frac{S_p^2}{n_2}}$
- $t_{\text{statistic}} = \frac{(\bar{X}_1 - \bar{X}_2) - (\mu_1 - \mu_2)}{SE}$
- $H_0: \mu_1 - \mu_2 = 0$
- $H_A: \mu_1 - \mu_2 \neq 0$
- $df = (n_1 - 1) + (n_2 - 1) = n_1 + n_2 - 2$
- $d = \frac{\bar{X}_1 - \bar{X}_2}{S_p^2}$
- $CI = (\bar{X}_1 - \bar{X}_2) \pm t_{\text{critical}} \cdot SE \quad \text{or} \quad CI = (\bar{X}_2 - \bar{X}_1) \pm t_{\text{critical}} \cdot SE$

## Excel

- Get external data
- String Functions: FIND, LEFT, RIGHT, MID, etc.
- Hide, Unhide columns
- Conditional Functions: IF, SUMIF, COUNTIF, IFNA, etc.
- VLOOKUP Function
  - 4th Parameter => True (classifying items) or False
  - Cannot look up in both directions
  - Slow for large data sets
- INDEX-MATCH Function
  - Can look up in both directions
  - Faster for large data sets
- Time and Date Functions: TIME, DATE, MONTH, WEEKDAY, WEEKNUM etc.
- Pivot Tables
  - They are more suitable for data analysis than functions
  - They do only simple analysis
  - They don't support scenario analysis when we compare two different filters => Array formulas can do this
  - Insert -> Pivot Table
  - Check => Add this data to the data model (useful to combine multiple tables)
  - Distinct Count
  - Power Pivot
  - Power Map
- Comparisons => Inside quotes “ ” followed by an ampersand &
  - COUNTIF(\$A\$1:\$A\$10,“>”&B2)
- Named Ranges => Cell reference (Right click on a cell or Formulas => Define a name)
  - There is no need to worry about dollar signs
  - Formulas are much easier to understand
  - \$A\$1:\$A\$10 => MaxLevel => COUNTIF(MaxLevel,“>”&B2)
- Name Manager
  - The problem with Named Ranges comes up, when adding data to them.
    - Expand the named range
    - Better strategy is to make your named range consist of an entire column. Downsides are
      - Accidentally input unwanted values into the named range
      - Cannot use the column for anything else besides values in the named range
- Most common value function: MODE
- Data Tables => alternative practice to address the two previous problems
  - You don't have to update the Named Range
  - Select the data in worksheet => Insert Table => Check My table has headers
  - Table Tools -> Design => Change the name of the Table

- We can Insert Slicer => Like an interactive filter
- ISERROR Function
- Intersection
- Paste special => Transpose => Named Ranges do NOT change!
- CONCATENATE Function
- FORMULAS -> Evaluate Formula
- Array Formulas and Structure => Make room in a spreadsheet
  - We can do calculations between ranges too, not just numbers
  - Control + Shift + Enter => Curly braces => Array calculations => Result = List
  - HOME => Filter
    - On rows only
    - Column filter => TRANSPOSE function => Array => Control + Shift + Enter
      - Make room for the transposed result. Not just a cell!
      - The transposed array will be updated # Copy => Paste Special (Transposed)
      - The function only maps the input data to the transposed array, it will not change the data types.
    - Some formulas can behave like “array”
    - Some formulas are “array” (type need to be array!) like Transpose
  - Double minus sign => -- (TRUE) = 1 and -- (FALSE) = 0
- Keep spreadsheet safe - Stress-test our spreadsheet!
  - Sanity checks, e.g. what property should probably hold for a margin? - Margin >= 0
  - Boundaries checks, e.g. 0 <= Percentage <= 100
  - Totals checks, e.g. SUM(Percentages) = 100%
  - Cross check
  - Validating user input
  - Tests are documentation
  - How to test
    - Home -> Conditional formatting
    - Funnel-style structure using formulas to check for errors
    - Gather all the tests in the spreadsheets together and give a final test in one value in one worksheet.
- Import CSV files
  - Refresh!
    - Uncheck Prompt for file name on refresh
    - Check Refresh data when opening the file
  - Merge csv files in terminal => `cat *.csv >merged.csv`
    - Data tab -> Remove Duplicates
- Pivot Tables and Charts
  - Insert Slicers
  - We can add column and formulas through Power Pivot -> Manage

- RELATED Function
- Conditional Formatting
  - Colour Formatting
  - Spark Lines => Visualise the data
    - See the trend in a single data series
    - Graphs are much more useful to compare different categories/datasets
  - Win/Loss spark lines shows whether each value is positive or negative instead of how high or low the values are.
- Pivot Charts - Add Chart Element
  - Add Trend Line
    - Linear Forecast => Add prediction line - Right click for more, e.g. equation
- Data -> Text to Columns => split the text into two columns
- Pivot Tables
  - Right click on cell -> Group
    - Grouping feature only works if “Add this data to the data model” is unchecked
- What-If Analysis
  - Goal Seek - Find the right input for the value you want
    - Data -> What-If Analysis -> Goal Seek
  - Data Table - See the results of multiple inputs at the same time
    - Data -> What-If Analysis -> Data Table
- Solver - More powerful than What-If analysis
  - Tools -> Excel Add-ins -> Enable Solver Add-In
    - Data -> Solver
      - We can change multiple variable cells
      - We can add constraints
- SUMPRODUCT Function
- Tables & Graphs for effective communication of data
  - Define my goal
    - Tables used when
      - look up individual values
      - compare individual values
      - precise values are required
      - there is more than one unit measured
      - both summary and detail values are included
    - Graphs used when
      - you want to show a pattern/trend
      - you want to reveal relationships between sets of values
  - Table Effective Design
    - Unidirectional with 1 category, Unidirectional with 2 categories & Bidirectional
    - Derived values close to the related columns

- Columns we want to compare should be located close
  - Make the grid less heavy or remove the grid
  - Keep only the important lines to separate for example the headers from the body
  - Use more white space and increase the row height but not very much
  - Use more white space in columns when we want column orientation => if I want my table to be read column by column
  - Fill the cell to show attention or light fill a row and a column to guide the eyes
  - Align text left and numbers right
  - Increase levels of precision if it is necessary
  - Include thousand separator
- Graph Effective Design
  - Maintain visual correspondence to quantity
  - Try to stay close to the truth
  - Do not try to manipulate your readers
  - Stay away from 3D!
  - Basic Graphs
    - Scatter plot to highlight a relation between two variables
    - Line chart to show a trend or a time series
    - Bar or Column chart to compare different categories with each other
      - Bar or Column depends on the labels fit
      - Time series are best represented with Column charts
  - Avoid Pie charts
    - It is difficult for humans to estimate an area
  - White space
  - Do not use patterns, at least use greyscale if you cannot use colours with good contrast
  - Put legend to the bottom, not the default right of excel
  - Use aspect ratio wisely => 1:1.5 is quite good
  - 5 variables are the max
    - For more variables create several smaller graphs oriented horizontally or vertically (depends on what I want to show). Separate the graphs based on what you want to focus.
- Dashboard Design
  - What is the goal
  - What is the message
  - The design depends on the kind of screen or paper
  - Create a grid
    - Dashboard Title
    - Area Title
    - Chart Title
    - Chart Footer, and so on...
- CHAR(13) to add a line break on the Mac
- Chart Design tab -> Switch Row/Column

- Thermometer custom chart types => Compare similar variables
- Copy - Paste as Linked Picture
  - with linked picture, you can change original data content and have live updates
- Pivot Table -> Filter -> Top 10 functionality
- Open VBA Editor => Fn + ⌘ + F11

## P2 – Intro to Data Analysis

### Lists, functions and methods

- sub setting a list with brackets []
- sub setting lists of lists with brackets [[]]
- slicing a list with x[begin:end]
- change element of a list to another element, e.g. x[2] = 'c'
- add an element at the end of a list, e.g. x = x + ['c']
- remove an element from a list, e.g. del(x[2])
- If you want to prevent changes in a list copy to also take effect in initial list, you'll have to do a more explicit copy of the list. You can do this with list() or by using [:].
- pip is a very commonly used tool to install and maintain Python packages.
- If numpy is imported as np, you need e.g. np.array() => import numpy as np
- The from numpy import array version will make it less clear in the code that you are using Numpy's array() function.

### Numpy arrays

- numpy.array()
- In Numpy, calculations are performed element-wise. The first element of x and the first element of y are added. Similar for the second and third element of x and y.
- Numpy arrays can only hold elements with the same basic type. The string is the most 'general' and free form to store data, so all other data types are converted to strings. This is called type coercion.
- To subset both regular Numpy arrays, you can use square brackets []. You can also use Boolean Numpy arrays, e.g. high = y > 5 ; y[high]
- Sub setting using the square bracket notation on lists or arrays works the same.
- 2D Numpy arrays also offer more advanced ways of sub setting compared to regular Python lists of lists. To select the second row, use the index 1 before the comma. To select the third column, use the index 2 after the comma.
- Regular list of lists
  - x = [["a", "b"], ["c", "d"]]
  - [x[0][0], x[1][0]]
  - 'a' 'c'
- numpy
  - import numpy as np
  - np\_x = np.array(x)
  - np\_x[:,0]
  - 'a' 'c'
- Numpy offers many functions to calculate basic statistics, such as np.mean(), np.median() and np.std(), np.corrcoef(), e.g. np.mean(x[:,1])

## Matplotlib

- `import matplotlib.pyplot as plt`  
`a = [1, 2, 3, 4]`  
`b = [3, 9, 2, 6]`  
`plt.plot(a, b)`  
`plt.scatter(a, b)`  
`plt.show()`
- `plt.xscale("log"); plt.hist(); plt.clf(); plt.xlabel(); plt.ylabel(); plt.fill_between(); plt.title(); plt.xticks(); plt.xticks(); plt.grid(); plt.text()`
- e.g. `plt.scatter(x = gdp_cap, y = life_exp, s = np.array(pop) * 2, c = col, alpha = 0.8)`  
where `c = col` is a dictionary { 'Asia':'red', 'Europe':'green', 'Africa':'blue', 'Americas':'yellow', 'Oceania':'black' }

## Pandas DataFrame

- 2D Numpy arrays can only contain values of the same basic type, a downside compared to Pandas if you're working on typical Data Science problems.
- Rows correspond to observations; columns correspond to variables, or properties, of these observations.
- `import pandas as pd ; pd.read_csv('example', index_col = 0)`
- To select an entire row by its row label when accessing data in a Pandas DataFrame use `loc`. Square brackets are used to get specific columns from a Pandas DataFrame. `iloc` is used if you want to select a row based on its position in the DataFrame, and not based on its row label.
- The single bracket version gives a Pandas Series, the double bracket version gives a Pandas DataFrame.

## Questions to answer usually in data analysis

- How something varies over time?
- What is the highest and lowest levels?
  - Which subject have them?
  - Where is the subject of interest on the spectrum?
- How the variables relate to each other?
- Are there any consistent trends across variables?



## Investigate a Dataset

### Anaconda

- Create a new environment, e.g. `conda create -n new_project python=3`
- Activate the new environment, e.g. `source activate new_project`
- Install packages in my new environment, e.g. `conda install numpy pandas matplotlib jupyter notebook`
- List my packages, i.e. `conda list`

### Python

#### CSVs in Python

1. Each row is a list, e.g. 

```
csv = [['A1', 'A2', 'A3'],  
      ['B2', 'B2', 'B3']]
```
2. Each row is a dictionary (works well if we have headers),  
e.g. 

```
csv = [{'name1': 'A1', 'name2': 'A2', 'name3': 'A3'},  
      {'name1': 'B2', 'name2': 'B2', 'name3': 'B3'}]
```

#### Import csv in Python

```
def read_csv(filename):  
    with open(filename, 'rb') as f:  
        reader = unicodedsv.DictReader(f)  
        return list(reader)
```

#### Fix data types

csv library does not try to detect what type of data each column has

#### Investigate the data for inconsistencies

e.g. with `'len()'` function find the total number of rows in the csv or the number of unique students with `'set()'`

#### Import csv with Pandas

```
Example: import pandas as pd  
daily_engagement = pd.read_csv('daily_engagement_full.csv')  
len(daily_engagement['acct'].unique())
```

Table 1. NumPy Arrays and Python Lists

Similarities	Differences
Access elements by position	Each element should have same type
Access a range of elements	Convenient functions
Use loops	Can be multi-dimensional
-	Vector operations
-	NumPy index Arrays
-	NumPy modify Arrays Not copy
-	NumPy implemented in C (fast!)

## Examples

# Looping

if True:

for country in countries:

print 'Examining country {}'.format(country)

for i in range(len(countries)):

country = countries[i]

country\_employment = employment[i]

print 'Country {} has employment {}'.format(country,

country\_employment)

Table 2. Vector operations

Math	Logical	Comparison
Add: +	And: &	Greater: >
Subtract: -	Or:	Greater or equal: >=
Multiply: *	Not: ~	Less: <
Divide: /		Less or equal: <=
Exponentiate: **		Equal: ==
		Not equal: !=

## Standardizing data

How does one data point compare to the rest?

- Convert each data point to number of standard deviations away from the mean
  - $(\text{values} - \text{values.mean()}) / \text{values.std()}$  → we keep negative signs!

## In-Place vs. Not In-Place

`+=` operates in-place while `+` does not

Operations that are not in-place are much easier to think about!

Table 3. Comparison of NumPy and Pandas

Pandas	NumPy (Numerical Python)
Series (more advanced)	Array (simpler)
Accessing elements	Accessing elements
Looping	Looping
Conventional functions	Conventional functions
Vector operations	Vector operations
Implemented in C (fast!)	Implemented in C (fast!)
Series index	-
Adding by Index	Adding by position
In 2D, columns can be of different type	-
2D (DataFrames) have index	-
DataFrames functions to each column	Functions (axis = 0 or axis = 1)
Functions (axis = 0 or axis = 1) or (axis = 'index' or axis = 'columns')	-

## Examples

if True:

```
s = pd.Series([1, 2, 3, 4, 5])
```

```
def add_one(x):
```

```
    return x + 1
```

```
print s.apply(add_one)      ≠      instead of applymap() for dataframes
```

```
import pandas as pd
```

```
import seaborn as sns
```

```
# The following code reads all the Gapminder data into Pandas DataFrames.
```

```
path = '/datasets/ud170/gapminder/'
```

```
employment = pd.read_csv(path + 'employment_above_15.csv', index_col='Country')
```

```
# The following code creates a Pandas Series for each variable for the United States.
```

```
employment_us = employment.loc['United States']
```

```
# Uncomment the following line of code to see the available country names
```

```
print employment.index.values
```

```
# Use the Series defined above to create a plot of each variable over time for the country of your choice.
```

```
employment_us.plot()
```

## Two-Dimensional Data

- Python: List of lists
- Numpy: 2D array
- Pandas: DataFrame
- 2D arrays, as opposed to array of arrays:
  - More memory efficient
  - Accessing elements is a bit different → `a[1, 3]` rather than `a[1][3]`
  - `mean()`, `std()`, etc. operate on entire array
- By default, numpy calculates a population standard deviation, with "ddof = 0". On the other hand, pandas calculate a sample standard deviation, with "ddof = 1".
- `groupby()`
- `merge()`

## Correlation Pearson's r

1. Standardize each value
2. Multiply each pair of values
3. Take the average

$$r = \text{average of } (x \text{ in standard units}) \times (y \text{ in standard units})$$

- It is important to take the uncorrected standard deviation: `var.std(ddof=0)`

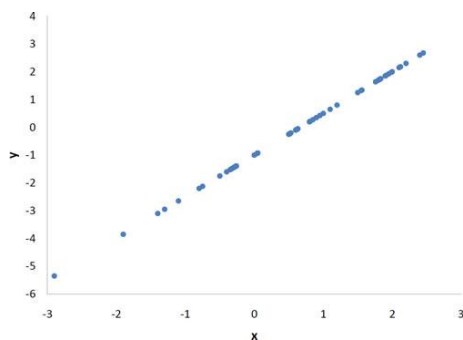


Figure 1. A perfect positive linear relationship,  $r = 1$ .

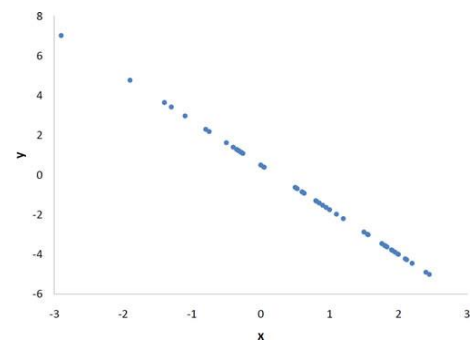


Figure 2. A perfect negative linear relationship,  $r = -1$ .

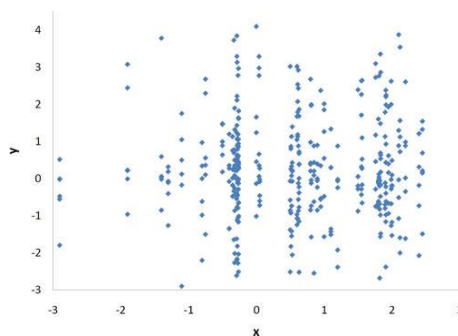


Figure 3. A scatter plot for which  $r = 0$ . Notice that there is no relationship between X and Y.

## P3 – Data Wrangling

- **Gather Data**

- Python string method `strip()` will come in handy to get rid of the extra whitespace (that includes newline character at the end of line).
- Use csv module: `import csv`
- Use xlr module to work with excel files: `import xlr`
- Data Modelling in JSON
  - Items may have different fields
  - May have nested objects
  - May have nested arrays

- **Asses Data**

- Test Assumptions about
  - Values
  - Data Types
  - Shape
- Identify Errors or Outliers

- **Blueprint for Cleaning**

- Audit your data, i.e. statistical analysis to check for outliers.
- Create a data cleaning plan
  - Identify causes that dirty data appears
  - Define operations that will correct our data
  - Run a few tests to check the data cleaning plan
- Execute the plan
- Manually correct

- **SQL**

### SQL CHEAT SHEET <http://www.sqltutorial.org>

#### QUERYING DATA FROM A TABLE

**SELECT** c1, c2 **FROM** t;  
Query data in columns c1, c2 from a table

**SELECT \* FROM** t;  
Query all rows and columns from a table

**SELECT** c1, c2 **FROM** t  
**WHERE** condition;  
Query data and filter rows with a condition

**SELECT DISTINCT** c1 **FROM** t  
**WHERE** condition;  
Query distinct rows from a table

**SELECT** c1, c2 **FROM** t  
**ORDER BY** c1 **ASC** [**DESC**];  
Sort the result set in ascending or descending order

**SELECT** c1, c2 **FROM** t  
**ORDER BY** c1  
**LIMIT** n **OFFSET** offset;  
Skip offset of rows and return the next n rows

**SELECT** c1, aggregate(c2)  
**FROM** t  
**GROUP BY** c1;  
Group rows using an aggregate function

**SELECT** c1, aggregate(c2)  
**FROM** t  
**GROUP BY** c1  
**HAVING** condition;  
Filter groups using HAVING clause

#### QUERYING FROM MULTIPLE TABLES

**SELECT** c1, c2  
**FROM** t1  
**INNER JOIN** t2 **ON** condition;  
Inner join t1 and t2

**SELECT** c1, c2  
**FROM** t1  
**LEFT JOIN** t2 **ON** condition;  
Left join t1 and t2

**SELECT** c1, c2  
**FROM** t1  
**RIGHT JOIN** t2 **ON** condition;  
Right join t1 and t2

**SELECT** c1, c2  
**FROM** t1  
**FULL OUTER JOIN** t2 **ON** condition;  
Perform full outer join

**SELECT** c1, c2  
**FROM** t1  
**CROSS JOIN** t2;  
Produce a Cartesian product of rows in tables

**SELECT** c1, c2  
**FROM** t1, t2;  
Another way to perform cross join

**SELECT** c1, c2  
**FROM** t1 A  
**INNER JOIN** t2 B **ON** condition;  
Join t1 to itself using INNER JOIN clause

#### USING SQL OPERATORS

**SELECT** c1, c2 **FROM** t1  
**UNION** [**ALL**]  
**SELECT** c1, c2 **FROM** t2;  
Combine rows from two queries

**SELECT** c1, c2 **FROM** t1  
**INTERSECT**  
**SELECT** c1, c2 **FROM** t2;  
Return the intersection of two queries

**SELECT** c1, c2 **FROM** t1  
**MINUS**  
**SELECT** c1, c2 **FROM** t2;  
Subtract a result set from another result set

**SELECT** c1, c2 **FROM** t1  
**WHERE** c1 [**NOT**] **LIKE** pattern;  
Query rows using pattern matching % \_

**SELECT** c1, c2 **FROM** t  
**WHERE** c1 [**NOT**] **IN** value\_list;  
Query rows in a list

**SELECT** c1, c2 **FROM** t  
**WHERE** c1 **BETWEEN** low **AND** high;  
Query rows between two values

**SELECT** c1, c2 **FROM** t  
**WHERE** c1 **IS** [**NOT**] **NULL**;  
Check if values in a table is NULL or not



## MANAGING TABLES

```
CREATE TABLE t (
  id INT PRIMARY KEY,
  name VARCHAR NOT NULL,
  price INT DEFAULT 0
);
```

Create a new table with three columns

```
DROP TABLE t;
```

Delete the table from the database

```
ALTER TABLE t ADD column;
```

Add a new column to the table

```
ALTER TABLE t DROP COLUMN c;
```

Drop column c from the table

```
ALTER TABLE t ADD constraint;
```

Add a constraint

```
ALTER TABLE t DROP constraint;
```

Drop a constraint

```
ALTER TABLE t1 RENAME TO t2;
```

Rename a table from t1 to t2

```
ALTER TABLE t1 RENAME c1 TO c2;
```

Rename column c1 to c2

```
TRUNCATE TABLE t;
```

Remove all data in a table

## USING SQL CONSTRAINTS

```
CREATE TABLE t(
  c1 INT, c2 INT, c3 VARCHAR,
  PRIMARY KEY (c1,c2)
);
```

Set c1 and c2 as a primary key

```
CREATE TABLE t1(
  c1 INT PRIMARY KEY,
  c2 INT,
  FOREIGN KEY (c2) REFERENCES t2(c2)
);
```

Set c2 column as a foreign key

```
CREATE TABLE t(
  c1 INT, c1 INT,
  UNIQUE(c2,c3)
);
```

Make the values in c1 and c2 unique

```
CREATE TABLE t(
  c1 INT, c2 INT,
  CHECK(c1 > 0 AND c1 >= c2)
);
```

Ensure c1 > 0 and values in c1 >= c2

```
CREATE TABLE t(
  c1 INT PRIMARY KEY,
  c2 VARCHAR NOT NULL
);
```

Set values in c2 column not NULL

## MODIFYING DATA

```
INSERT INTO t(column_list)
VALUES(value_list);
```

Insert one row into a table

```
INSERT INTO t(column_list)
VALUES (value_list), ...;
```

Insert multiple rows into a table

```
INSERT INTO t1(column_list)
SELECT column_list
FROM t2;
```

Insert rows from t2 into t1

```
UPDATE t
SET c1 = new_value;
```

Update new value in the column c1 for all rows

```
UPDATE t
SET c1 = new_value,
    c2 = new_value
WHERE condition;
```

Update values in the column c1, c2 that match the condition

```
DELETE FROM t;
```

Delete all data in a table

```
DELETE FROM t
WHERE condition;
```

Delete subset of rows in a table



## MANAGING VIEWS

```
CREATE VIEW v(c1,c2)
AS
SELECT c1, c2
FROM t;
```

Create a new view that consists of c1 and c2

```
CREATE VIEW v(c1,c2)
AS
SELECT c1, c2
FROM t;
WITH [CASCADED | LOCAL] CHECK OPTION;
```

Create a new view with check option

```
CREATE RECURSIVE VIEW v
AS
select-statement -- anchor part
UNION [ALL]
select-statement; -- recursive part
```

Create a recursive view

```
CREATE TEMPORARY VIEW v
AS
SELECT c1, c2
FROM t;
```

Create a temporary view

```
DROP VIEW view_name;
```

Delete a view

## MANAGING INDEXES

```
CREATE INDEX idx_name
ON t(c1,c2);
```

Create an index on c1 and c2 of the table t

```
CREATE UNIQUE INDEX idx_name
ON t(c3,c4);
```

Create a unique index on c3, c4 of the table t

```
DROP INDEX idx_name;
```

Drop an index

## SQL AGGREGATE FUNCTIONS

**AVG** returns the average of a list

**COUNT** returns the number of elements of a list

**SUM** returns the total of a list

**MAX** returns the maximum value in a list

**MIN** returns the minimum value in a list

## MANAGING TRIGGERS

```
CREATE OR MODIFY TRIGGER trigger_name
WHEN EVENT
ON table_name TRIGGER_TYPE
EXECUTE stored_procedure;
```

Create or modify a trigger

**WHEN**

- **BEFORE** – invoke before the event occurs
- **AFTER** – invoke after the event occurs

**EVENT**

- **INSERT** – invoke for INSERT
- **UPDATE** – invoke for UPDATE
- **DELETE** – invoke for DELETE

**TRIGGER\_TYPE**

- **FOR EACH ROW**
- **FOR EACH STATEMENT**

```
CREATE TRIGGER before_insert_person
BEFORE INSERT
ON person FOR EACH ROW
EXECUTE stored_procedure;
```

Create a trigger invoked before a new row is inserted into the person table

```
DROP TRIGGER trigger_name;
```

Delete a specific trigger

- Always put 'single quotes' around text strings and date/time values.
- **Text and string types**
  - **text** – a string of any length, like Python **str** or **unicode** types.
  - **char(n)** – a string of exactly *n* characters.
  - **varchar(n)** – a string of up to *n* characters.
- **Numeric types**
  - **integer** – an integer value, like Python **int**.
  - **real** – a floating-point value, like Python **float** (up to 6 decimal places).
  - **double precision** – a higher-precision floating-point value (up to 15 decimal places).

- **decimal** — an exact decimal value.
- **Date and time types**
  - **date** — a calendar date; including year, month, and day.
  - **time** — a time of day.
  - **timestamp** — a date and time together.

- **Examples**

```
SELECT Composer, Name FROM Track WHERE Composer='Van Halen';
SELECT Composer, SUM(Milliseconds) FROM Track WHERE Composer='Johann Sebastian Bach';
SELECT FirstName, LastName, Title, Birthdate FROM Employee;
INSERT INTO animals(name, birthdate, species) values('Alex', '2017-04-24', 'opossum');
```

```
SELECT BillingCountry, COUNT(*)
FROM Invoice
GROUP BY BillingCountry
ORDER BY COUNT(*)
DESC
LIMIT 3;
```

```
SELECT Email, FirstName, LastName, SUM(Total)
FROM Customer, Invoice
WHERE Customer.CustomerID = Invoice.CustomerID
GROUP BY Email
ORDER BY SUM(Total)
DESC
LIMIT 1;
```

```
SELECT Email, FirstName, LastName, Genre.Name
FROM Customer, Invoice, InvoiceLine, Track, Genre
WHERE Customer.CustomerId = Invoice.CustomerId
AND Invoice.InvoiceId = InvoiceLine.InvoiceId
AND InvoiceLine.TrackId = Track.TrackId
AND Track.GenreId = Genre.GenreId
AND Genre.Name = 'Rock'
GROUP BY Email
ORDER BY Email;
```

```
SELECT BillingCity, sum(Total)
FROM Invoice
GROUP BY BillingCity
ORDER BY sum(Total)
DESC
LIMIT 1;
```

```
SELECT BillingCity, COUNT(*), Genre.Name
FROM Invoice, InvoiceLine, Track, Genre
WHERE Invoice.InvoiceId = InvoiceLine.InvoiceId
AND InvoiceLine.TrackId = Track.TrackId
AND Track.GenreId = Genre.GenreId
AND BillingCity = 'Prague'
GROUP BY Genre.Name
ORDER BY COUNT(*)
DESC
LIMIT 3;
```

```
SELECT Artist.Name, COUNT(Genre.Name)
FROM Genre, Track, Album, Artist
WHERE Genre.GenreID = Track.GenreID
AND Track.AlbumID = Album.AlbumId
AND Album.ArtistId = Artist.ArtistId
AND Genre.Name = 'Rock'
```

```
GROUP BY Artist.Name
ORDER BY COUNT(Genre.Name)
DESC
LIMIT 10;
```

```
SELECT BillingCity, COUNT(Track.TrackId) AS NumInvoices
FROM Invoice, InvoiceLine, Track, Genre
WHERE Invoice.InvoiceId = InvoiceLine.InvoiceId
AND InvoiceLine.TrackId = Track.TrackId
AND Track.GenreId = Genre.GenreId
GROUP BY BillingCity, Genre.Name
HAVING Invoice.BillingCountry='France' AND Genre.Name = 'Alternative & Punk'
ORDER BY NumInvoices
DESC
```

## Normalised Design

In a normalised database, the relationships among the tables match the relationships that are really there among the data.

- Every row has the same number of columns.
- There is a unique key, and everything in a row says something about the key.
- Facts that don't relate to the key belong in different tables.
- Tables should not imply relationships that don't exist.

- **CREATE and DROP databases**

```
CREATE database name[options];
```

```
DROP database name[options];
```

```
DROP table name[options];
```

- Primary key
- Declaring Relationships: references

- **Self JOINS**

```
SELECT a.id, b.id, a.building, a.room
FROM residences as a, residences as b
WHERE a.building = b.building
AND a.room = b.room
AND a.id < b.id
ORDER BY a.building, a.room;
```

```
SELECT products.name, products.sku, count(sales.sku) AS num
FROM products LEFT join sales
ON products.sku = sales.sku
GROUP BY products.sku;
```

SQL supports a number of variations on the theme of joins. The kind of join that you have seen earlier in this course is called an *inner* join, and it is the most common kind of join — so common that SQL doesn't actually make us say "inner join" to do one.

But the second most common is the **left join**, and its mirror-image partner, the **right join**. The words "left" and "right" refer to the tables to the left and right of the join operator. (Above, the left table is **products** and the right table is **sales**.)



A regular (inner) join returns only those rows where the two tables have entries matching the join condition. A **left join** returns all those rows, plus the rows where the *left* table has an entry but the right table doesn't. And a **right join** does the same but for the *right* table. (Just as "join" is short for "inner join", so too is "left join" actually short for "left outer join". But SQL lets us just say "left join", which is a lot less typing. So, we'll do that.)

## SQL in Python

```
import sqlite3

# Fetch some student records from the database.
db = sqlite3.connect("students")
c = db.cursor()
query = "select name, id from students order by name;"
c.execute(query)
rows = c.fetchall()

# First, what data structure did we get?
print "Row data:"
print rows

# And let's loop over it too:
print
print "Student names:"
for row in rows:
    print " ", row[0]

db.close()
```

---

```
import sqlite3

db = sqlite3.connect("testdb")
c = db.cursor()
c.execute("insert into balloons values ('blue', 'water')")
db.commit()
db.close()
```

### • Subqueries

```
# Two Queries
def lightweights(cursor):
    """Returns a list of the players in the db whose weight is less than the average."""
    cursor.execute("select avg(weight) as av from players;")
    av = cursor.fetchall()[0][0] # first column of first (and only) row
    cursor.execute("select name, weight from players where weight < " + str(av))
    return cursor.fetchall()

# Or One Query
def lightweights(cursor):
    """Returns a list of the players in the db whose weight is less than the average."""
    cursor.execute("select name, weight from players, (select avg(weight) as av from players) as subq where weight < av;")
    return cursor.fetchall()

# Or
def lightweights(cursor):
    """Returns a list of the players in the db whose weight is less than the average."""
    cursor.execute("select name, weight from players where weight < (select avg(weight) as av from players);")
    return cursor.fetchall()
```

```
SELECT sum(total)
FROM (SELECT count(*) AS total
      FROM Invoice
      GROUP BY BillingCountry
      ORDER BY total DESC
      LIMIT 5);
```

```
SELECT BillingCity, BillingState, BillingCountry, Total
FROM Invoice,
(SELECT avg(Total) as average FROM Invoice) as subquery
WHERE Total > average;
```

```
SELECT FirstName, LastName, BillingCity, BillingCountry, Total
FROM Invoice JOIN Customer ON Invoice.CustomerId = Customer.Customer.Id JOIN
(SELECT avg(Total) As average
 FROM Invoice) AS subquery
WHERE Total > average;
```

- **VIEWS**

A VIEW is a SELECT query stored in the database in a way that lets you use it like a table.

```
CREATE VIEW course_size AS
SELECT course_id, count(*) AS num
FROM enrolment
GROUP BY course_id;
```

- **Example**

After all the success promoting your music tour last section, a new friend has asked to partner up and build your own music website! You'll need to rebuild your own database and import the data to your new system. Let's first take a closer look at how to build and populate your local database. The box below shows the Album table schema including Primary and Foreign Keys. Have a look at this table and the CREATE TABLE statement below to see how they relate. First, disconnect from your Chinook database.

> .exit

Create a new database named whatever you'd like your store to be called.

\$ sqlite3 UdaciousMusic.db

Now we can populate this database with our first table. Here's a graphic showing some information about the Album table. We can use this to build a table in our new database.

```
#####
#           Table: Album           #
#####
```

Columns	Data Type	Primary Key	Foreign Key
AlbumId	INTEGER	YES	NO
Title	TEXT	NO	NO
ArtistId	INTEGER	NO	YES
UnitPrice	REAL	NO	NO
Quantity	INTEGER	NO	NO

```
#####
```

We can use this information to decide how our schema should look. Do you see how the schema below reflects the table above?

```
CREATE TABLE Album
(
    AlbumId INTEGER PRIMARY KEY,
    Title TEXT,
    ArtistId INTEGER,
    FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId)
);
```

Try pasting the schema into your local database. Let's check to see if anything happened.

```
sqlite> .tables
```

Album <--- Do you see the Album table? I hope so!  
Now, do we have any data in our new table?

```
sqlite> SELECT * FROM Album;
```

Do you see data? I hope not, we haven't added any yet!  
Open the Album.sql tab. You can copy and paste these lines directly into your sqlite terminal. (Use Ctrl+A or Command+A to select all lines when the code editor is selected to select all the lines at once.)  
Now try to run your query again. You've got data... NICE!

- **Export Data to CSV from Database**

```
sqlite > .mode csv
sqlite > .output newFile.csv
sqlite > SELECT * FROM myTable;
sqlite > .exit
```

- **Import CSV into a Table**

```
$ sqlite3 new.db <--- If you'd like your csv's in a new database remember to make it first.
sqlite > CREATE TABLE myTable() <--- Build your schema!
sqlite > .mode csv
sqlite > .import newFile.csv myTable
```

- **Example**

Connect to the Chinook database.

```
import sqlite3

# Fetch records from chinook.db
db = sqlite3.connect("chinook.db")
c = db.cursor()
QUERY = "SELECT * FROM Invoice;"
c.execute(QUERY)
rows = c.fetchall()
# See query in Python
print "Row data:"
print rows

# See query by row
print "your output:"
for row in rows:
    print " ", row[0:]

# See query as a pandas dataframe
import pandas as pd
```

```
df = pd.DataFrame(rows)
print df
```

```
db.close()
```

- **Example**

```
SELECT count(DISTINCT Customer.CustomerId)
FROM Customer JOIN Invoice JOIN InvoiceLine JOIN Track JOIN Genre
ON Customer.CustomerId = Invoice.CustomerId
AND Invoice.InvoiceId = InvoiceLine.InvoiceId
AND InvoiceLine.TrackID = Track.TrackId
AND Track.GenreId = Genre.GenreId
WHERE Genre.name = 'Jazz';
```

## Intro to Machine Learning

- **Naive Bayes**

- Features (x) and Labels (y) → Scatter Plot
- Classification by Eye
- From Scatterplots to Predictions
- From Scatterplots to Decision Surfaces
  - A Good Linear Decision Surface
- Transition to Using Naive Bayes (NB)
  - NB Decision Boundary in Python
    - Getting Started With sklearn (Gaussian Naïve Bayes)

```
from sklearn.naive_bayes import GaussianNB
# create classifier
clf = GaussianNB()
# fit the classifier on the training features and labels
clf.fit(features_train, labels_train)
# make prediction
pred = clf.predict(features_test)
# calculate the accuracy on the test data
accuracy = clf.score(features_test, labels_test) # or
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(pred, labels_test)
```

-