Introduction to Data Science

Welcome to Zipfian Academy's (http://zipfianacademy.com) Introduction to Data Science course. Thank you for attending, we hope you enjoyed the lecture (we sure had fun presenting). This exercise will give you hands-on experience with the concepts covered, and will help solidify your understanding of the process of data science.

Getting Help

If you have any questions throughout this exercise or need help at all, please ask on Piazza (https://piazza.com/class#summer2013/101) and join the conversation. Chances are that if you have a question or get stuck, someone else is in the same situation. If you post to Piazza we (the instructors) can answer the question for everyone, and other students can provide insights as well.

As always, feel free to email us about anything at all (questions, issues, concerns, feedback) at class@zipfianacademy.com (mailto:class@zipfianacademy.com). We would love to hear how you liked the class, whether the content was technical enough (or too technical), or any other topics you wish were covered.

Next Steps

We hope you have fun with this exercise! If you want to learn more or dive deeper into any of these subjects, we are always happy to discuss (and can talk for days about these subjects). We will be continuing the data journey with the subsequent classes in the series (https://team-zipfian-statistics.eventbrite.com/). We encourage you to sign up for any class(es) covering topics which you would like to dive deeper into and continue your learning. And if you just can't get enough of this stuff (and want a completely immersive environment), you can apply (http://zipfiancollective.wufoo.com/forms/z7x3p3/) for our intensive data science bootcamp starting September 16th.

Learning Python

This assignment assumes a basic familiarity with Python and is intended to teach you how to leverage it for data science. If you do not feel comfortable enough with Python (and programming in general) I recommend these (freely available) resources:

- Think Python (http://www.greenteapress.com/thinkpython/thinkpython.pdf)
- MIT Open Courseware: A Gentle Introduction to Programming Using Python (http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-189-a-gentle-introduction-to-programming-using-python-january-iap-2008/index.htm)
- Learn Python the Hard Way (http://learnpythonthehardway.org/book/)
- Python Koans (https://github.com/gregmalcolm/python_koans/wiki)

Setup and Environment

This exercise is written in an IPython (http://ipython.org/) notebook (http://ipython.org/notebook.html) and uses many of wonderful libraries from the scientific Python community (http://strata.oreilly.com/2013/03/python-data-tools-just-keep-getting-better.html). While you do not need IPython locally to complete the exercise (there are PDF and .ipynb versions of these instructions), I recommend setting it up on your computer if you plan to continue learning and playing with data. IPython notebooks (http://ipython.org/ipython-doc/stable/interactive/htmlnotebook.html) not only provide an interface to

interactively run (and debug) code in a web browser, but also to document your file as you go along. Below are the steps to setup a scientific Python environment on your computer to complete this (and all future class') assignment. If you have tips or suggestions to make this process easier, please reach out either on Piazza or via email.

Version control and Environment Isolation

- Git (http://git-scm.com/): Distributed Version Control to keep track of changes and updates to files/data.
- virualenv (http://www.virtualenv.org/en/latest/): Python environment isolation to help manage dependencies with packages and versions.
- pythonbrew (https://github.com/utahta/pythonbrew): Manage and install multiple versions of Python. Can be handy if you want to experiment with Python 3.x.

Scientific Python packages

- Enthought Python Distribution (https://www.enthought.com/products/epd/free/): A freely available packaged environment for scientific Python.
- Scipy Superpack (http://fonnesbeck.github.io/ScipySuperpack/): Only for Mac OSX, but a one line shell script that installs all the fundamental scientific computing packages.
- pandas (http://pandas.pydata.org/): Data analysis and statistical library providing functionality in Python similar to R (http://www.r-project.org/).

if you are on OSX, you may need to install Xcode (http://developer.apple.com/library/ios/#documentation/DeveloperTools/Conceptual/WhatsNewXcode/Articles/xcode_4_3.html) (with command line utilities) or install gcc (https://medium.com/kr-projects/6e54e8c50dc8) directly

Tutorial (https://sites.google.com/site/pythonbootcamp/preparation/software) walking you through the installation of these tools, with tests (https://sites.google.com/site/pythonbootcamp/preparation/testing-that-it-all-works) to make sure it all works.

Exercise: Happy Healthy Hungry -- San Francisco

Now that your environment is all setup up... the fun begins! In this exercise we will walk through the data science process (http://zipfianacademy.com/data/data-science-process.png) covered in lecture. We will be analyzing the inspections of San Francisco restaurants using publicly available data (http://www.sfdph.org/dph/EH/Food/score/default.asp) from the Department of Public health. We will learn to explore this data to map the cleanliness of the city, and get a better perspective on the relative meaning of these scores by looking at summary statistics of the data. We will also use simple regression in addition to more advanced Machine Learning methods to attempt to predict the score of restaurants that have pending inspections. Once we understand how SF fares, we will compare its restaurants to those of NYC (https://nycopendata.socrata.com/Health/Restaurant-Inspection-Results/4vkw-7nck).

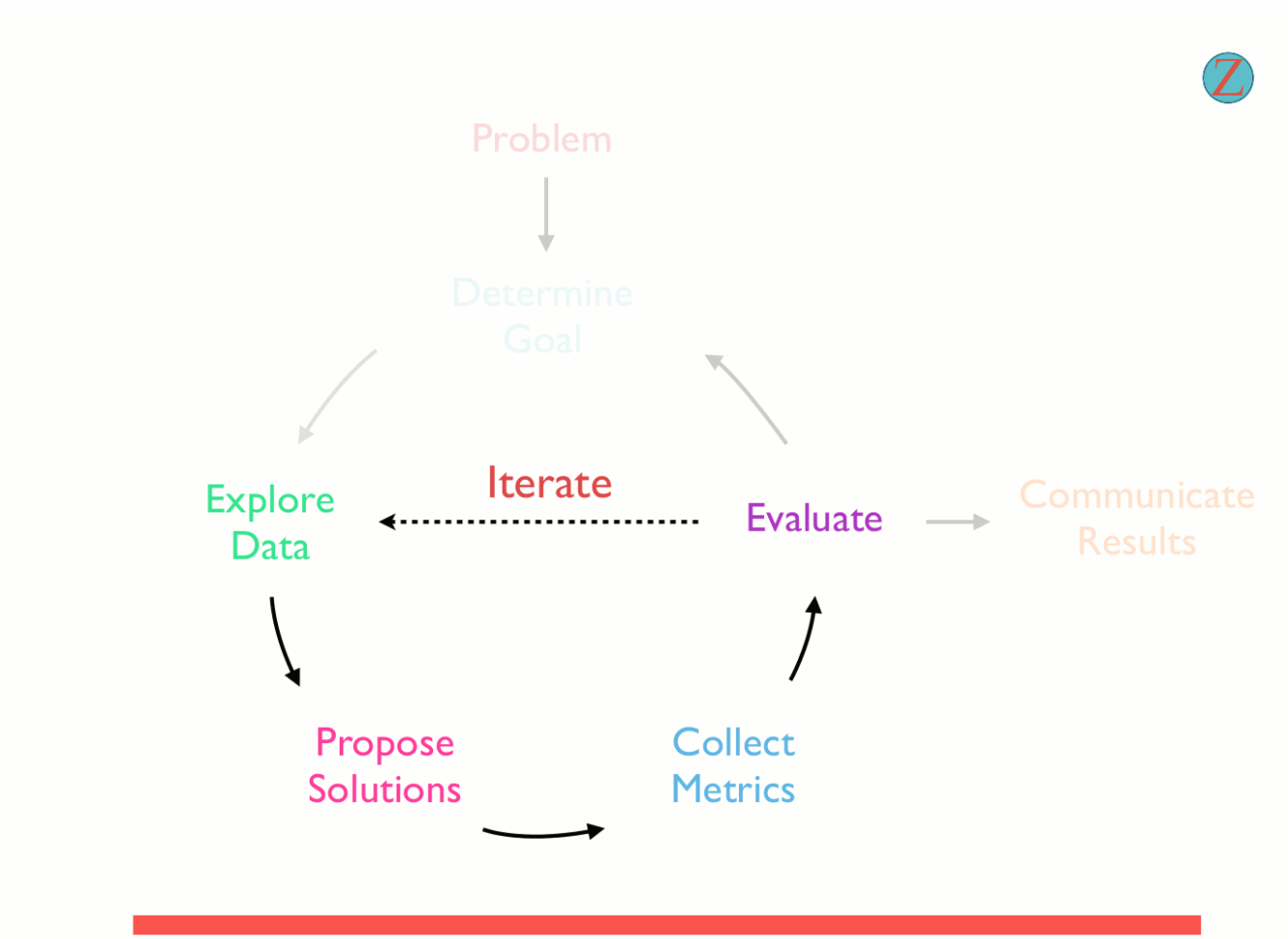
This case study will be explored throughout the class series (http://team-zipfian-statistics.eventbrite.com/). Each subsequent class will go into much more depth on each specific topic. This assignment wil provide an introduction to the problem, and focus on the data science process (http://zipfianacademy.com/data/data-science-workflow/animate.gif) itself more than individual subjects.

In [297]:
```
# Import pylab to provide scientific Python libraries (NumPy, SciPy, Matplotlib)
import pylab

# import the Image display module
from IPython.display import Image

# inline allows us to embed matplotlib figures directly into the IPython notebook
%pylab inline
```

```
Welcome to pylab, a matplotlib-based Python environment [backend:
module://IPython.kernel.zmq.pylab.backend_inline].
For more information, type 'help(pylab)'.
```

In [298]:
```
Image(url='http://zipfianacademy.com/data/data-science-workflow/animate.gif', width=700)
```

Out[298]:



Problem

The first step of the **Process** is to define the problem we want to address. To do so let us review what we have set out to accomplish and begin exploring questions we want answered.

Important points to keep in mind when defining our problem:

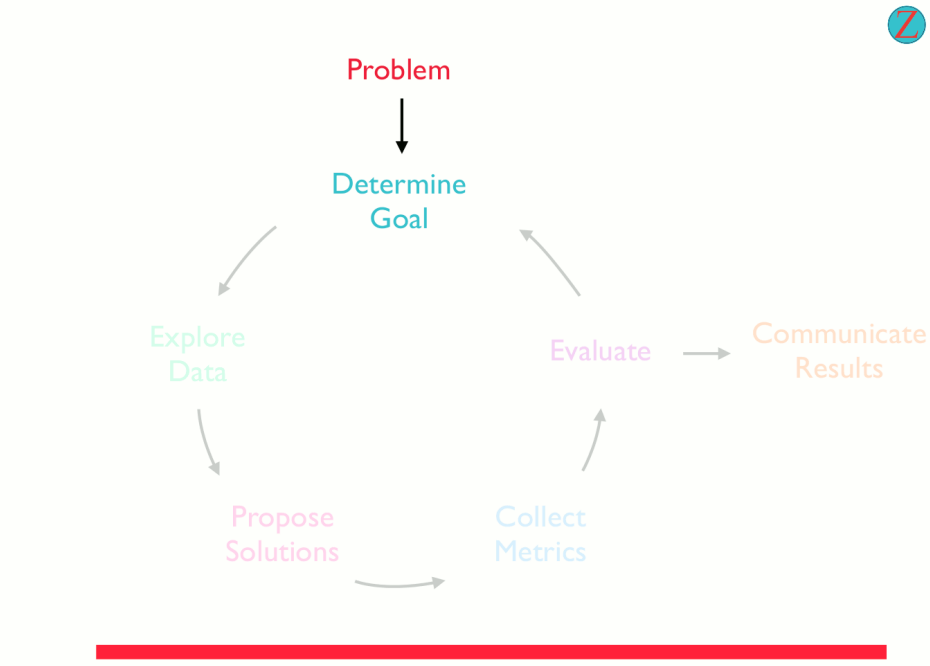
- The question can be ***qualitative***, but the approach must be ***quantifiable***
- What am I ***looking*** for?
- What do I want to ***learn***?
- Alright to be ***exploratory*** (and the best analysis often are)

Exercise 1: Formulate an appropriate problem statement given our case study

*** ANSWER HERE ***

In [299]:
```
Image(url='http://zipfianacademy.com/data/data-science-workflow/goal.png', width=500)
```

Out[299]:



Determine Goal

Now that we have a problem we hope to solve, let us begin to quantify our analysis. Since our *Problem Statement* is often qualitative and broad, we can ask further questions to better define what we hope to achieve.

Some things to note about our goals and approach:

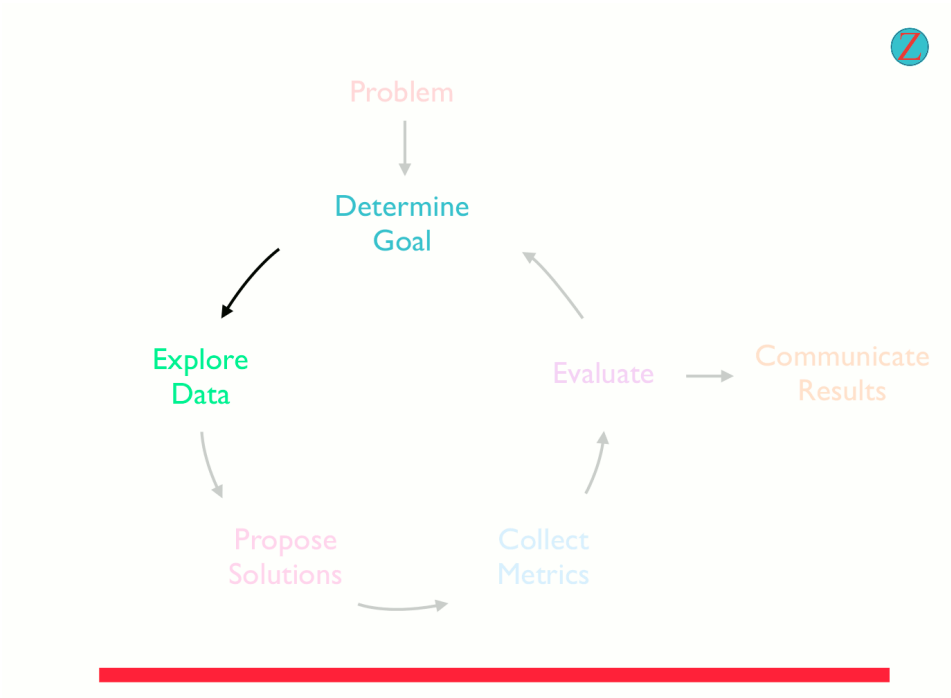
- Determine what defines **success**, and to what degree.
- Brainstorm **metrics** to visualize and/or calculate.
- Ask **questions** that have (or can have) a definitive answer.
- Be careful what you wish for, be aware of possible **correlations**, and take caution with how you measure (http://en.wikipedia.org/wiki/Observer-expectancy_effect) it.

Exercise 2: Derive three additional questions based on the probelm defined above (exercise 1)

*** ANSWER HERE ***

In [300]:
```
Image(url='http://zipfianacademy.com/data/data-science-workflow/explore.png', width=500)
```

Out[300]:



Explore Data

To recap where we are in our analysis:

- We have determined what we want to learn -- ***Exercise 1***
- We explored quantifiable metrics to collect -- ***Exercise 2***

The ***Explore*** stage of the analysis is where we will most likely spend most of our time (http://strataconf.com/stratany2012/public/schedule/detail/27495). Now comes the fun part (in my opinion)! At this stage we will use a variety of tools (the documentation of each linked to inline) to figure out where and how to obtain data, what it looks like once we have it, and how to use it to answer our questions to achieve our goals.

Acquire

Luckily, San Francisco has much of its public government data freely accessible (https://data.sfgov.org/) online. There are also great initiatives (http://www.yelp.com/healthscores) by SF companies (http://officialblog.yelp.com/2013/01/introducing-lives.html) collaborating with non-profits (http://codeforamerica.org/) and government (http://sfgov.org/) to develop open data standards (http://foodinspectiondata.us/). Such standardization allows for much more transparency, leading ultimately to a more engaged citizenry.

The relevant data (http://www.sfdph.org/dph/EH/Food/score/default.asp) has been downloaded for your convenience and can be found on Piazza in the Resources section (https://piazza.com/zipfian_academy/summer2013/101/resources).

Examine

If you are working with this IPython notebook, download the data files into the same directory which you ran the `ipython notebook` *command*

Now that we have found the relevant data we can begin to peer inside to understand what we are working with. I recommend starting with an iterative approach, using the quickest/easiest tools first and slowly build to more complicated analyes. UNIX provides us with many powerful tools and can carry us quite far by itself. In our case the dataset came with documentation (https://s3.amazonaws.com/piazza-resources/hgtp0qhpaps1d5/hhfjlqv3gii2l8/File_Specifications.pdf?AWSAccessKeyId=AKIAJKOQYKAYOBKKVTKQ&Expires=1370258028&Signature=ZsHGKBMNwbdv9Ptio3b6GYrhB08%3D) of its contents, but it still is essential to look at the raw data and compare it to the docs.

Exercise 3.1: Display the first 5 lines of each of the data files (CSV files in SFBusinesses directory).

Extra Credit:

- Display the last 5 lines of each file
- Format the output nicely into columns

Relevant resources:

- IPython tutorial (http://ipython.org/ipython-doc/rel-0.13.1/interactive/tutorial.html)
- IPthon: System Shell Access (http://ipython.org/ipython-doc/rel-0.13.1/interactive/reference.html#system-shell-access)
- head (UNIX) (http://en.wikipedia.org/wiki/Head_%28Unix%29)
- UNIX pipes (http://en.wikipedia.org/wiki/Pipeline_%28Unix%29)
- column (UNIX) (http://linux.about.com/library/cmd/blcmdl1_column.htm)
- tail (UNIX) (http://en.wikipedia.org/wiki/Tail_%28Unix%29)

There are two different data directories, each of which has similar files. Let's try to figure out the difference between the two, since the documentation on the data does not mention anything.

Exercise 3.2: Compare the files in SFBusinesses with SFFoodProgram_Complete_Data. For the corresponding files:

- Display the first 5 lines of each file.
- Compare the column names, column contents, and number of columns.
- Compare the number of lines (records), word counts, and file size.

Extra Credit:

- Use AWK (http://en.wikipedia.org/wiki/AWK) and Python to automate this for all the files in the directory (e.g. pass in two directories and spit out the relevant metrics)
- Use IPython cell magics to run an entire shell as a bash script/process

Relevant resources:

- wc (UNIX) (http://en.wikipedia.org/wiki/Wc_%28Unix%29)
- grep (UNIX) (https://en.wikipedia.org/wiki/Grep)
- IPython cell magics (http://nbviewer.ipython.org/url/github.com/ipython/ipython/raw/master/examples/notebooks/Cell%20Magics.ipynb)
- AWK (http://en.wikipedia.org/wiki/AWK)

Prepare

This is typically what people refer to as data 'munging' (or 'wrangling') and often is the most tedious process when working with messy data. Due to increasing awareness of the importance of data quality, the city of SF has been making great strides in more open and accessible (http://www.datasf.org/) data. If you (the city of SF) know the format (http://www.yelp.com/healthscores) you will need going into the data collection process (inspecting restaurants) you can hopefully avoid a lot of pain later in the analysis process.

The preparation process of our analysis is not as long and cumbersome as it typically might be due to the high quality of the raw data. Because of this, I will spare you much of the tedium of this step so we can focus on the more interesting aspects of the analysis. If you want to see (and experience) the pain (all you masochists out there), we will get much deeper into data acquisition and scrubbing techniques in our data wrangling class (http://team-zipfian-data-wrangling.eventbrite.com/) of this series.

Transform

Now that we know the structure of our data, we can start to begin examining it statistically to get a macrosopic look at its distribution. This part of our tutorial will use much of the powerful built in functionality of NumPy (http://www.numpy.org/), SciPy (http://www.scipy.org/), matplotlib (http://matplotlib.org/), and pandas (http://pandas.pydata.org/). If you want to get more experience with these, there are great resources (http://fperez.org/py4science/starter_kit.html) and tutorials (http://www.rexx.com/~dkuhlman/scipy_course_01.html) covering these libraries in much more depth (http://scipy-lectures.github.io/) than I will here. I highly recommend taking a look at these if this analysis interests you even in the least bit.

In [2]:
```
'''
To perform some interesting statistical analyses, we first need to "join" our CSV files ir
with their inspection scores. This data currently resides in SFBusinesses/businesses.csv a
'''

# import pandas library which provides an R like environment for python.
# if you do not have it installed: sudo easy_install pandas.
from pandas import Series, DataFrame

import pandas as pd

# store relevant file paths in variables since we may use them frequently
root_dir = 'SFBusinesses/'
businesses = root_dir + 'businesses.csv'
inspections = root_dir + 'inspections.csv'


# load each file into a Pandas DataFrame, pandas automatically converts the first line int

df_business = pd.read_csv(businesses)
df_inspection = pd.read_csv(inspections)

# inspect the first 10 rows of the DataFrame
df_inspection.head(10)
```

Out[2]:

	business_id	Score	date	type
0	10	98	20121114	routine
1	10	98	20120403	routine

2	10	100	20110928	routine
3	10	96	20110428	routine
4	10	100	20101210	routine
5	12	100	20121120	routine
6	12	98	20120420	routine
7	12	100	20111018	routine
8	12	100	20110401	routine
9	17	100	20120823	routine

Exercise 3.3: Join the businesses.csv and inspections.csv on the business_id column.

- Use pandas DataFrame functions to merge df_businesses and df_inspection.
- Inspect the resulting table.

Extra Credit:

- Print out the column names of df_businesses, df_inspection, and the resulting join table.
- Display the ['name', 'date', 'Score', 'type'] columns of the result join table for the first 10 records (rows)

Relevant resources:

- pandas homepage (http://pandas.pydata.org/)
- pandas documentation (http://pandas.pydata.org/pandas-docs/stable/index.html)
- pandas tutorial (http://www.randalolson.com/2012/08/06/statistical-analysis-made-easy-in-python/)
- pandas: Merge, Join, and Concatenate (http://pandas.pydata.org/pandas-docs/stable/merging.html)
- pandas: Selecting Data (http://pandas.pydata.org/pandas-docs/stable/indexing.html)
- Python for Data Analysis: O'Reilly book (http://my.safaribooksonline.com/book/programming/python/9781449323592)

Now that we have our joined data, we can start exploring it

To analyze our businesses we need to perform some basic aggregations and transformations. Currently each business has multiple rows in the join table (all with the same business_id). Let us group these records by business and only keep the most recent inspection.

Exercise 3.4: Transform the table to only contain each businesses most recent inspection

- Group the resulting join table by the 'business_id' column.
- Filter the table such that only the most recent score is listed.

Extra Credit:

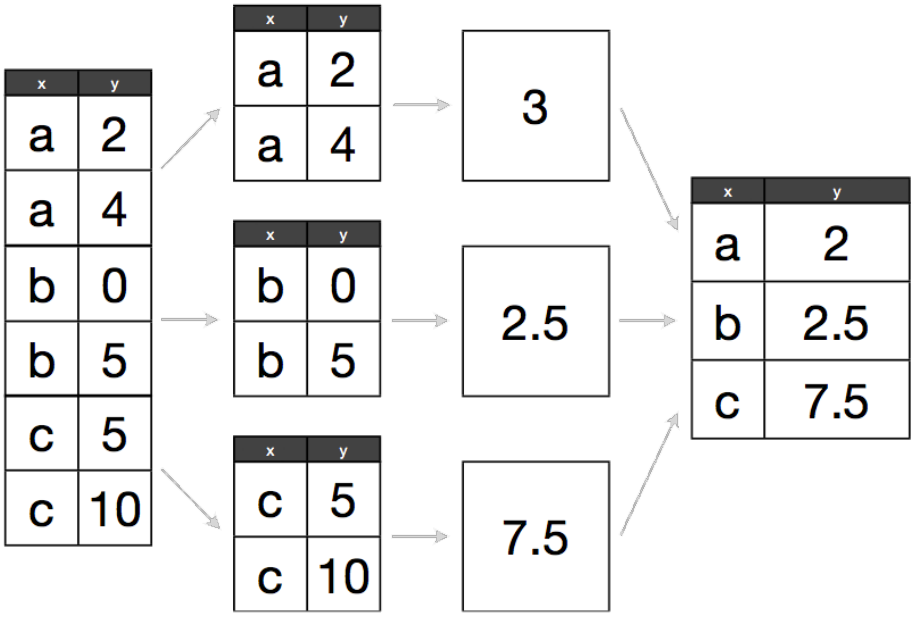
- Use pandas function application (http://pandas.pydata.org/pandas-docs/stable/basics.html#function-application) with a lambda (http://docs.python.org/2/tutorial/controlflow.html#lambda-forms) function.
- Instead of the most recent inspection score, return the average of all the inspections.

Relevant resources:

- pandas Group By: split-apply-combine (http://pandas.pydata.org/pandas-docs/stable/groupby.html)
- pandas `apply` (http://pandas.pydata.org/pandas-docs/stable/groupby.html#flexible-apply)
- pandas `sort_index` (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.sort_index.html#pandas.DataFrame.sort_index)
- pandas `head` (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.head.html#pandas.DataFrame.head)

In [312]:
```
Image(url='http://inundata.org/R_talks/meetup/images/splitapply.png', width=500)
```

Out[312]:



Split-Apply-Combine

A visual representation of how group-by, aggregate, and apply semantics work

We can bin the restaurants by scores to understand the distribution of inspections better. Here we create a histogram to understand the distribution of scores better

Exercise 3.5: Create a histogram of the most recent inspection scores of all the restaurants

- Create a matplotlib (http://matplotlib.org/) figure of a histogram with 100 bins.

Extra Credit:

- Scale the x-axis (inspection scores) to range from 40 to 100, incrementing by 2 (only even values).
- Add an appropriate title and axis labels to the graph.

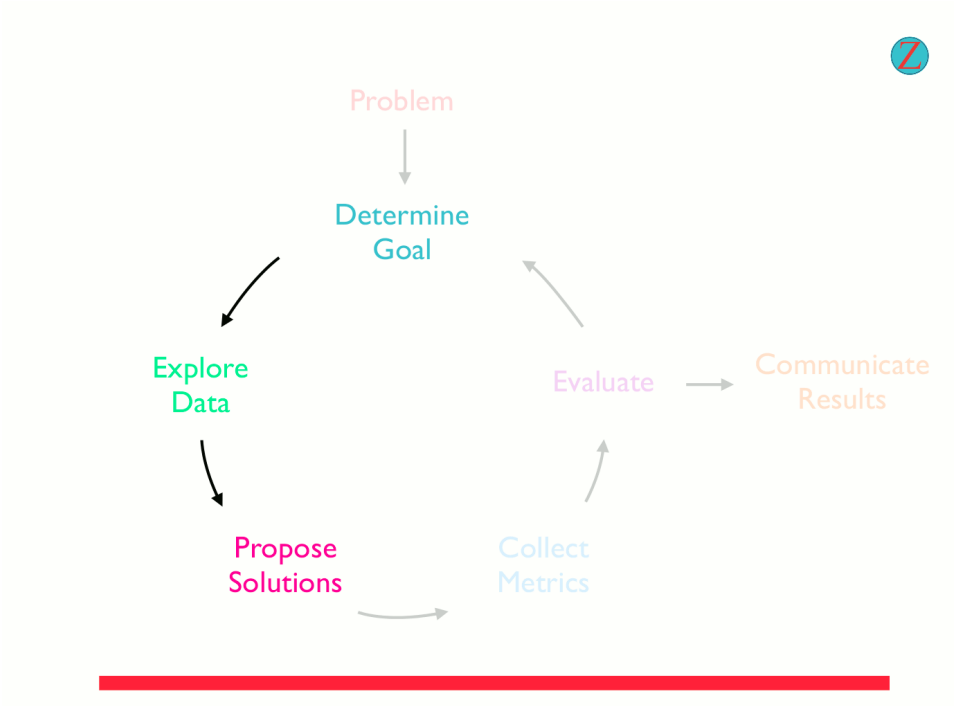
Relevant resources:

- pandas: Plotting with matplotlib (http://pandas.pydata.org/pandas-docs/stable/visualization.html)
- Scipy Lectures: matplotlib (http://scipy-lectures.github.io/intro/matplotlib/matplotlib.html)
- matplotlib documentation (http://matplotlib.org/1.2.1/index.html)
- pandas histograms: `hist()` (http://pandas.pydata.org/pandas-docs/stable/visualization.html#histograms)
- pandas: setting major and minor ticks and labels (http://stackoverflow.com/questions/12945971/pandas-

timeseries-plot-setting-x-axis-major-and-minor-ticks-and-labels)

In [319]:
```
Image(url='http://zipfianacademy.com/data/data-science-workflow/solutions.png', width=500)
```

Out[319]:



Propose Solutions

Since we have explored our data and have a better idea of its nature, we can begin to devise a plan to answer our questions. This is usually the most iterative part of the entire process: as we learn more about our data we modify our approach, and as modify our solutions we must re-examine our data.

Goals:

> How does an individual restaurants' score compare to the whole/aggregate of SF?
>
> Are SF's inspections better or worse than other cities?
>
> If a restaurant has not yet been inspected, can we approximate/predict what score it will receive?

Exercise 4.0: For each question of your goals, propose a solution.

This is simply a one or two line statement to drive you forward in your analyses. Refer to the lecure notes for an example of how we formulated solutions in the context of Wolfram's analysis.

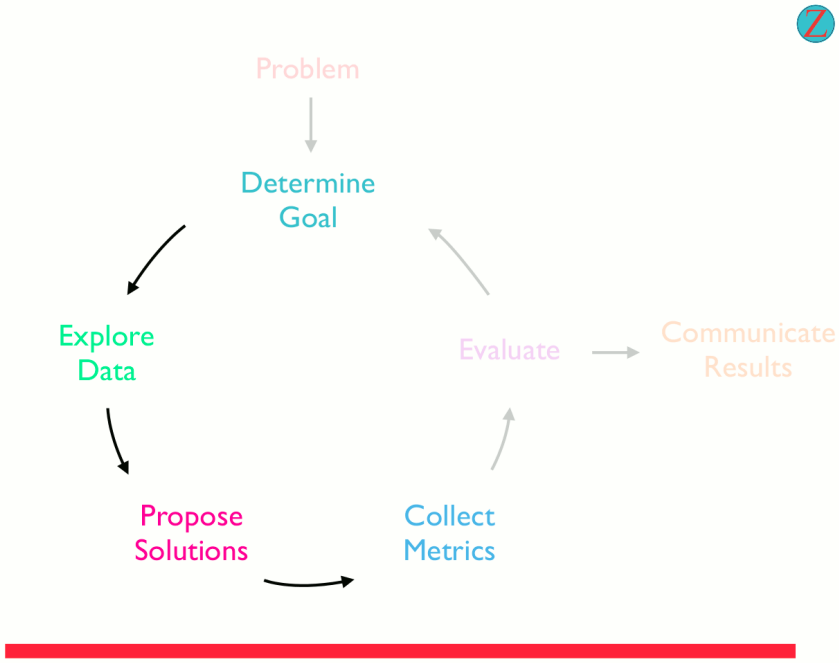
> *** ANSWER HERE ***

Some things to note about formulating solutions:

- Ask, how can I address the problem?
- In what ways can I use the data to achieve my goals?
- The simplest solution is often best (and the one you should try first)
- Quantize the metrics needed for your analysis

In [324]:
```
Image(url='http://zipfianacademy.com/data/data-science-workflow/metrics.png', width=500)
```

Out[324]:



Collect Metrics

This is the step where derivative values are often calculated, including **summary statistics**, **transformations** on the data, and **correlations**. There also is a bit of traditional **data mining** involved as most machine learning occurs in the solutions and metrics stages (in our formulation). We could even go so far as to say that the results of predictive models are simply additional metrics: the **probability** of defaulting on a loan, the **cluster** a new product belongs in, or the **score** of a restaurant that hasn't been inspected yet.

The purpose of this part of the process is to calculate the information you need to begin evaluating and testing you solutions and hypotheses.

Exercise 5.0: Compute descriptive statistics of the most recent restaurant inspection scores.

Calculate:

- total count of inspections
- mean
- standard deviation
- minimum and maximum
- quantiles

Extra Credit:

- Do not use pandas `describe()` convenience method.
- Add an appropriate title and axis labels to the graph.

Relevant resources:

- pandas: Essential Basic Functionality (http://pandas.pydata.org/pandas-docs/stable/basics.html)
- pandas: Descriptive Statistics (http://pandas.pydata.org/pandas-docs/stable/basics.html#descriptive-statistics)
- pandas `describe()` (http://pandas.pydata.org/pandas-docs/stable/basics.html#summarizing-data-describe)

In [315]:
```
# recall that in the Score Legend, each numeric score corresponds to a more qualitative de
!head -n 5 SFBusinesses/ScoreLegend.csv | column -t -s ','
```

```
"Minimum_Score"  "Maximum_Score"  "Description"
0                70               "Poor"
71               85               "Needs Improvement"
86               90               "Adequate"
91               100              "Good"
```

Exercise 5.1: Quantize the raw numeric inspection scores into the more qualitative scores ('Poor', 'Needs Imporvement', 'Adequate', 'Good') for the inspections.

- Discretize the current numeric scores into these 'categorical' descriptions.
- Print out the ['name', 'date', 'Score', 'type'] columns of this new table for the first 15 records (rows).

Extra Credit:

- Create a new DataFrame of the original records, but with the 'Score' numeric values replaced by these 'categorical' scores.

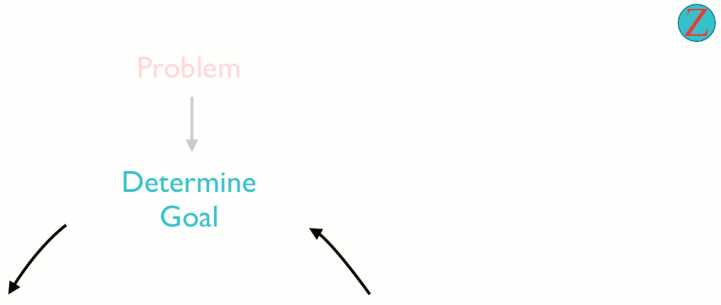
Relevant resources:

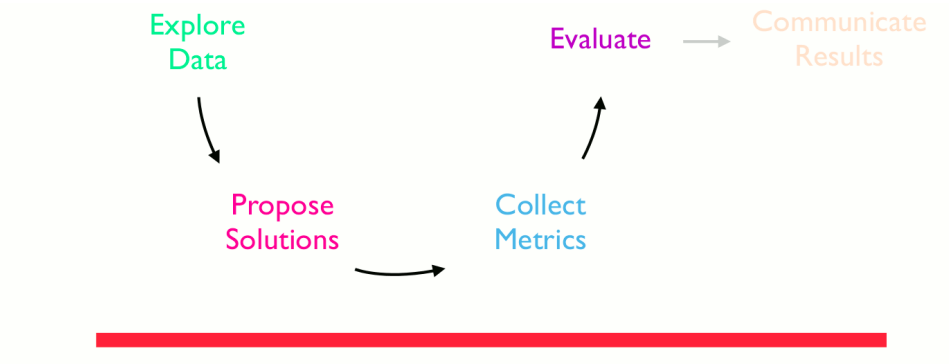
- pandas: Discretization and Quantiling (http://pandas.pydata.org/pandas-docs/stable/basics.html#discretization-and-quantiling)
- pandas `cut()` (http://pandas.pydata.org/pandas-docs/stable/reshaping.html?highlight=cut#tiling)
- pandas `copy()` (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.copy.html#pandas.DataFrame.copy)

By quantizing the scores of the restaurant inspections, we can get a better qualitative insight into the ratings. Let us compare this new distribution of quantized scores to the raw numeric values.

In [326]:
```
Image(url='http://zipfianacademy.com/data/data-science-workflow/evaluate.png', width=500)
```

Out[326]:





Evaluate

With the metrics we need properly calculated, it is time to draw some conclusions from our analyses. We need to evaluate whether the result we have arrived at:

- Answers our original question to an acceptable level of confidence.
- Has allowed us to achieve our goals?

Exercise 6.0: Create a histogram of these discretized scores ('Poor', 'Needs Imporvement', 'Adequate', 'Good')

- Plot the descriptive scores on a histogram with 4 bins -- ['Poor', 'Needs Imporvement', 'Adequate', 'Good']
- Print out the ['name', 'date', 'Score', 'type'] columns of this new table for the first 15 records (rows).

Extra Credit:

- Create a figure with 2 subplots () to display the raw (numeric) scores histogram next to the quantized (descriptions) histogram.
- Decorate both of these subplot each with its own unique title, axis labels, and x-axis ticks/scale.
- Arrange data such that 'Poor' is plotted as the left-most bar and 'Good' the right-most bar.
- Display the counts of each bin above its associated bar in the histogram.

Relevant resources:

- pandas `value_counts()` (http://pandas.pydata.org/pandas-docs/stable/basics.html#value-counts-histogramming)
- pandas bar charts (http://pandas.pydata.org/pandas-docs/stable/visualization.html#bar-plots)
- matplotlib tutorial (http://matplotlib.org/users/pyplot_tutorial.html)
- matplotlib `subplot()` (http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.subplot)

Exercise 7: Iterate!

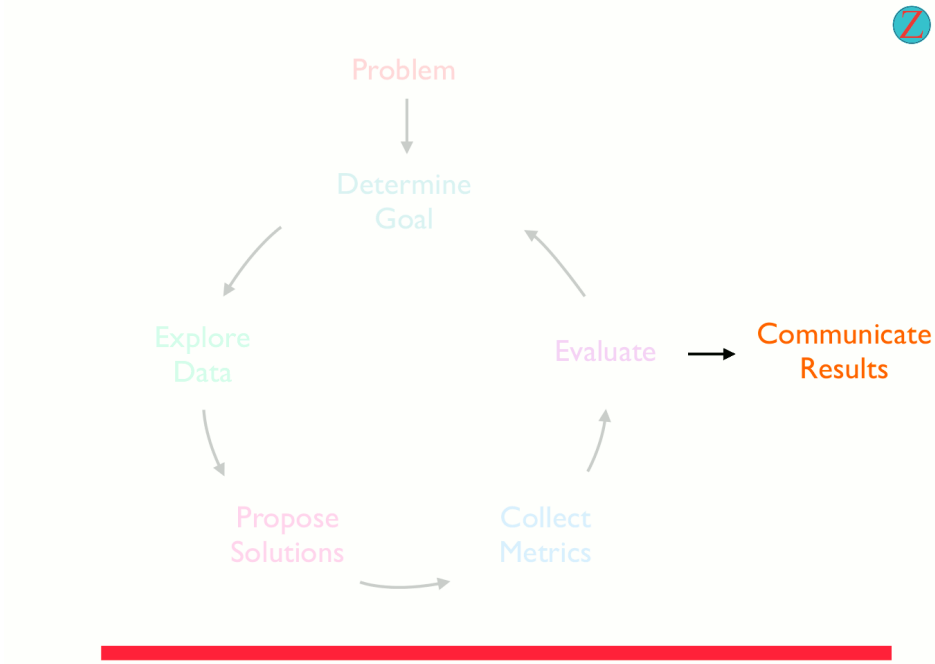
If we are not satisfied with our evaluation, we need to iterate on our approach:

- Do I need more/better data?
- Do I need to try a different proposed solution?
- Do I need to calculate different metrics?

I will leave the process of iteration (out of the scope of this exercise) as an additional task for the student. With such analyses as these you can endlessly refine your solution and find new questions to ask... and I whole heartedly encourage all of you to do so!

In [364]:
```
Image(url='http://zipfianacademy.com/data/data-science-workflow/communicate.png', width=5(
```

Out[364]:



Communicate Results

Once you are satisfied with your analysis and feel that you have accomplished your goals you set out to achieve, it is time to share your work with the world!.

Not only should your final visualization communicate your results, but it should also show your process. With such wonderful tools as IPython and with docstrings (http://www.pythonforbeginners.com/basics/python-docstrings/) built into the language, there is no excuse to not document your work and process. In keeping in line with the scientific method (https://en.wikipedia.org/wiki/Scientific_method), you want your analyses to be readily reproducible.

Exercise 8: Create a unique visualization the you feel effective communicates your analyses.

Some suggestions:

- Geographically plot inspection scores on a map of the city.
- Create plots of how much each individual inspection compares to the mean/average of SF restuarants.
- Overlay common distributions on the histogram (or scatterplot) of restaurant inspection scores for comparison.
- Create a heatmap of scores identifying any possible hotspots for 'dirty' restuarants.

Extra Credit:

- Make your map interactive using Javascript.
- Use an API (Yelp (http://www.yelp.com/developers/documentation), Google Places (https://developers.google.com/places/documentation/), Foursqaure (https://developer.foursquare.com/)) to augment the health inspection data with reviews, ratings, descriptions, menus, etc.

- Share your results by blogging (https://www.tumblr.com/) about your process and put your code on Github (https://github.com/) (or a similar service)

Relevant resources:

- matplotlib: Basemap (http://matplotlib.org/basemap/users/examples.html)
- folium: Python data. Meet Leaflet maps (https://github.com/wrobstory/folium)
- Polymaps Javascript library (http://polymaps.org/)
- D3.js maps tutorial (http://www.schneidy.com/Tutorials/MapsTutorial.html)
- vincent: Vega with Python (https://github.com/wrobstory/vincent)