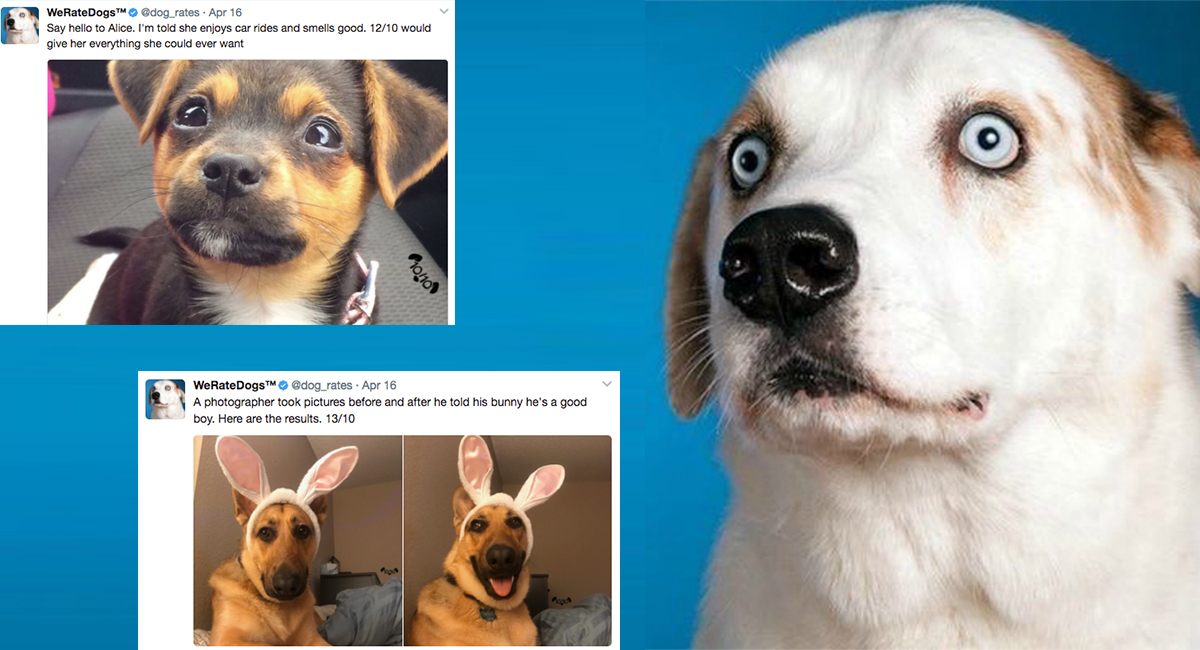
**Introduction**

Real-world data rarely comes clean. Using Python and its libraries, you will gather data from a variety of sources and in a variety of formats, assess its quality and tidiness, then clean it. This is called data wrangling. You will document your wrangling efforts in a Jupyter Notebook, plus showcase them through analyses and visualizations using Python (and its libraries) and/or SQL.

The dataset that you will be wrangling (and analyzing and visualizing) is the tweet archive of Twitter user [**@dog\_rates**](https://twitter.com/dog_rates), also known as **[WeRateDogs](https://en.wikipedia.org/wiki/WeRateDogs" \t "_blank)**. WeRateDogs is a Twitter account that rates people's dogs with a humorous comment about the dog. These ratings almost always have a denominator of 10. The numerators, though? Almost always greater than 10. 11/10, 12/10, 13/10, etc. Why? Because "[**they're good dogs Brent**](http://knowyourmeme.com/memes/theyre-good-dogs-brent)." WeRateDogs has over 4 million followers and has received international media coverage.

WeRateDogs [**downloaded their Twitter archive**](https://support.twitter.com/articles/20170160) and sent it to Udacity via email exclusively for you to use in this project. This archive contains basic tweet data (tweet ID, timestamp, text, etc.) for all 5000+ of their tweets as they stood on August 1, 2017. More on this soon.



*Image via*[***Boston Magazine***](http://www.bostonmagazine.com/arts-entertainment/blog/2017/04/18/dog-rates-mit/)

**Project Steps Overview**

Your tasks in this project are as follows:

Step 1: Gathering data

Step 2: Assessing data

Step 3: Cleaning data

Step 4: Storing data

Step 5: Analyzing, and visualizing data

Step 6: Reporting

* your data wrangling efforts
* your data analyses and visualizations

**What Software Do I Need?**

The entirety of this project can be completed inside the Udacity classroom on the **Project Workspace: Complete and Submit Project** page using the Jupyter Notebook provided there. (Note: This Project Workspace may not be available in all versions of this project, in which case you should follow the directions below.)

If you want to work outside of the Udacity classroom, the following software requirements apply:

* You need to be able to work in a Jupyter Notebook on your computer. Please revisit our Jupyter Notebook and Anaconda tutorials earlier in the Nanodegree program for installation instructions.
* The following packages (libraries) need to be installed. You can install these packages via conda or pip. Please revisit our Anaconda tutorial earlier in the Nanodegree program for package installation instructions.
  + pandas
  + NumPy
  + requests
  + tweepy
  + json
* You need to be able to create written documents that contain images and you need to be able to export these documents as PDF files. This task can be done in a Jupyter Notebook, but you might prefer to use a word processor like [**Google Docs**](https://www.google.com/docs/about/), which is free, or Microsoft Word.
* A text editor, like [**Sublime**](https://www.sublimetext.com/), which is free, will be useful but is not required.

**Project Motivation**

**Context**

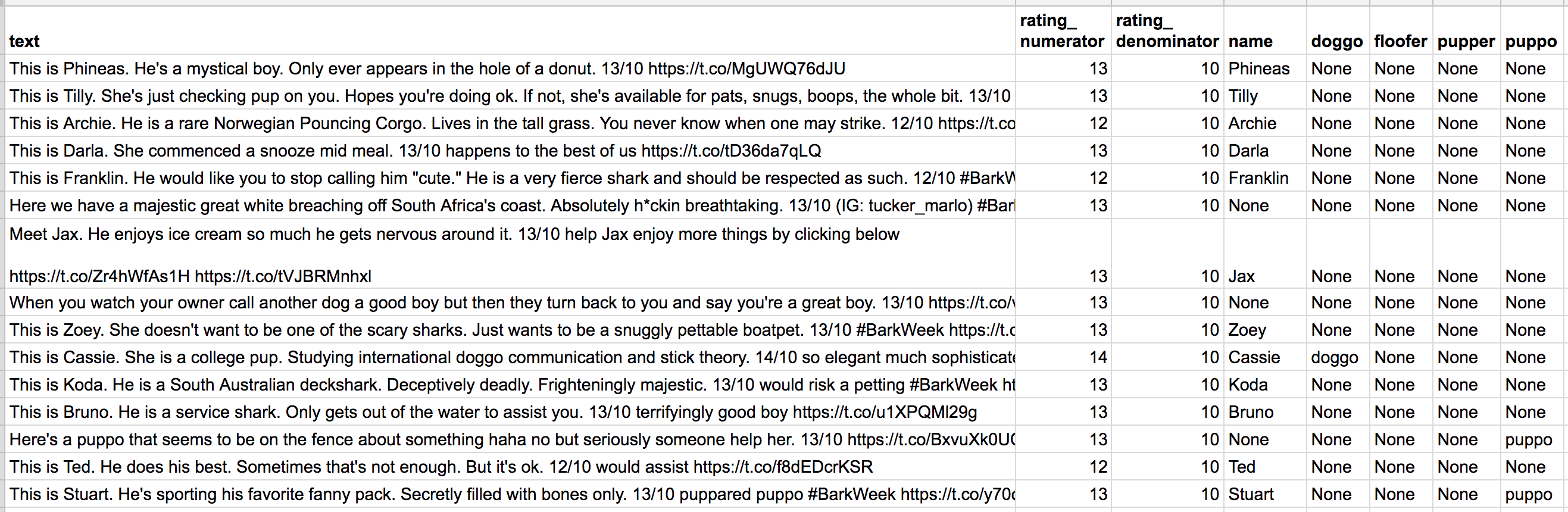
Your goal: wrangle WeRateDogs Twitter data to create interesting and trustworthy analyses and visualizations. The Twitter archive is great, but it only contains very basic tweet information. Additional gathering, then assessing and cleaning is required for "*Wow!*"-worthy analyses and visualizations.

**The Data**

In this project, you will work on the following three datasets.

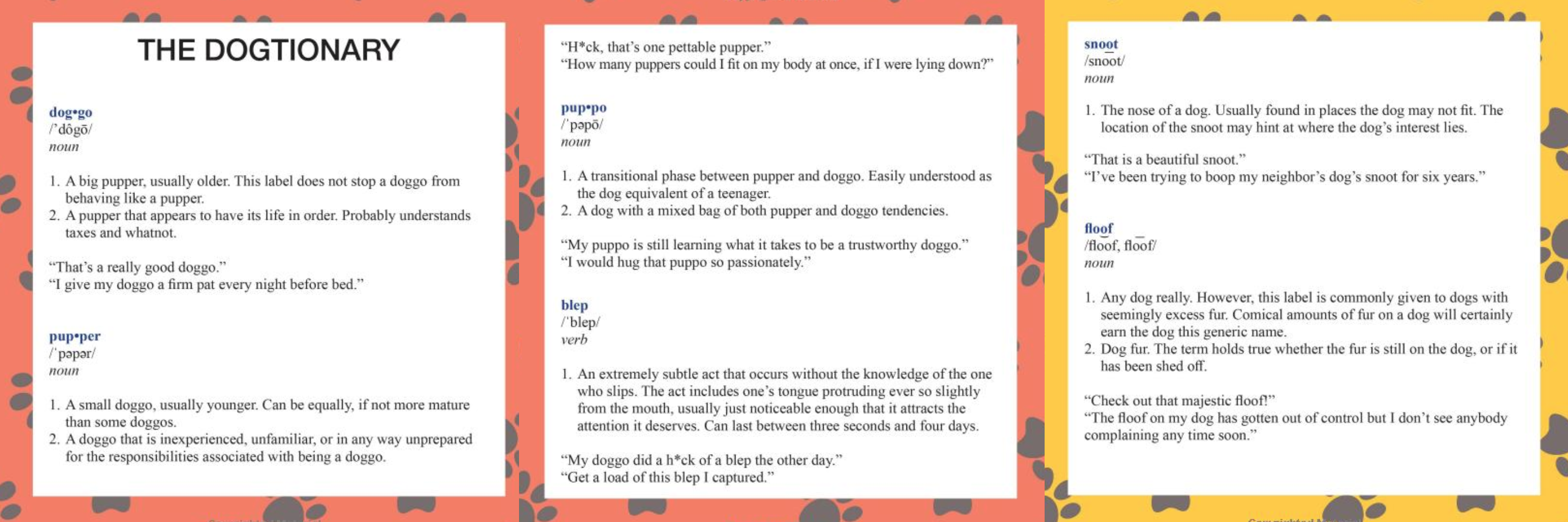
**Enhanced Twitter Archive**

The WeRateDogs Twitter archive contains basic tweet data for all 5000+ of their tweets, but not everything. One column the archive does contain though: each tweet's text, which I used to extract rating, dog name, and dog "stage" (i.e. doggo, floofer, pupper, and puppo) to make this Twitter archive "enhanced." Of the 5000+ tweets, I have filtered for tweets with ratings only (there are 2356).



*The extracted data from each tweet's text*

I extracted this data programmatically, but I didn't do a very good job. The ratings probably aren't all correct. Same goes for the dog names and probably dog stages (see below for more information on these) too. You'll need to assess and clean these columns if you want to use them for analysis and visualization.



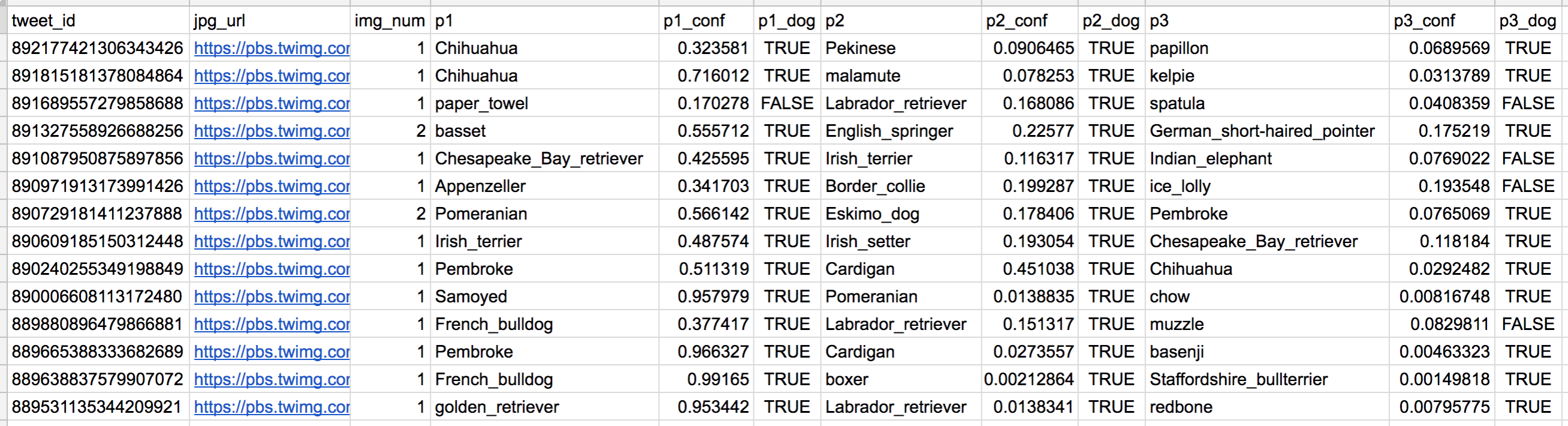
*The Dogtionary explains the various stages of dog: doggo, pupper, puppo, and floof(er) (via the*[***#WeRateDogs book***](https://www.amazon.com/WeRateDogs-Most-Hilarious-Adorable-Youve/dp/1510717145)*on Amazon)*

**Additional Data via the Twitter API**

Back to the basic-ness of Twitter archives: retweet count and favorite count are two of the notable column omissions. Fortunately, this additional data can be gathered by anyone from Twitter's API. Well, "anyone" who has access to data for the 3000 most recent tweets, at least. But you, because you have the WeRateDogs Twitter archive and specifically the tweet IDs within it, can gather this data for all 5000+. And guess what? You're going to query Twitter's API to gather this valuable data.

**Image Predictions File**

One more cool thing: I ran every image in the WeRateDogs Twitter archive through a [**neural network**](https://www.youtube.com/watch?v=2-Ol7ZB0MmU) that can classify breeds of dogs\*. The results: a table full of image predictions (the top three only) alongside each tweet ID, image URL, and the image number that corresponded to the most confident prediction (numbered 1 to 4 since tweets can have up to four images).



*Tweet image prediction data*

So for the last row in that table:

* tweet\_id is the last part of the tweet URL after "*status/*" → [**https://twitter.com/dog\_rates/status/889531135344209921**](https://twitter.com/dog_rates/status/889531135344209921)
* p1 is the algorithm's #1 prediction for the image in the tweet → **golden retriever**
* p1\_conf is how confident the algorithm is in its #1 prediction → **95%**
* p1\_dog is whether or not the #1 prediction is a breed of dog → **TRUE**
* p2 is the algorithm's second most likely prediction → **Labrador retriever**
* p2\_conf is how confident the algorithm is in its #2 prediction → **1%**
* p2\_dog is whether or not the #2 prediction is a breed of dog → **TRUE**
* etc.

And the #1 prediction for the image in that tweet was spot on:



*A golden retriever named Stuart*

So that's all fun and good. But all of this additional data will need to be gathered, assessed, and cleaned. This is where you come in.

\**Fun fact: creating this neural network is one of the projects in Udacity's*[***Data Scientist Nanodegree***](https://www.udacity.com/course/data-scientist-nanodegree--nd025)*,*[***Machine Learning Engineer Nanodegree***](https://www.udacity.com/course/machine-learning-engineer-nanodegree--nd009)*and*[***Artificial Intelligence Nanodegree programs***](https://www.udacity.com/course/artificial-intelligence-nanodegree--nd889)*.*

## Step 1: Gathering Data

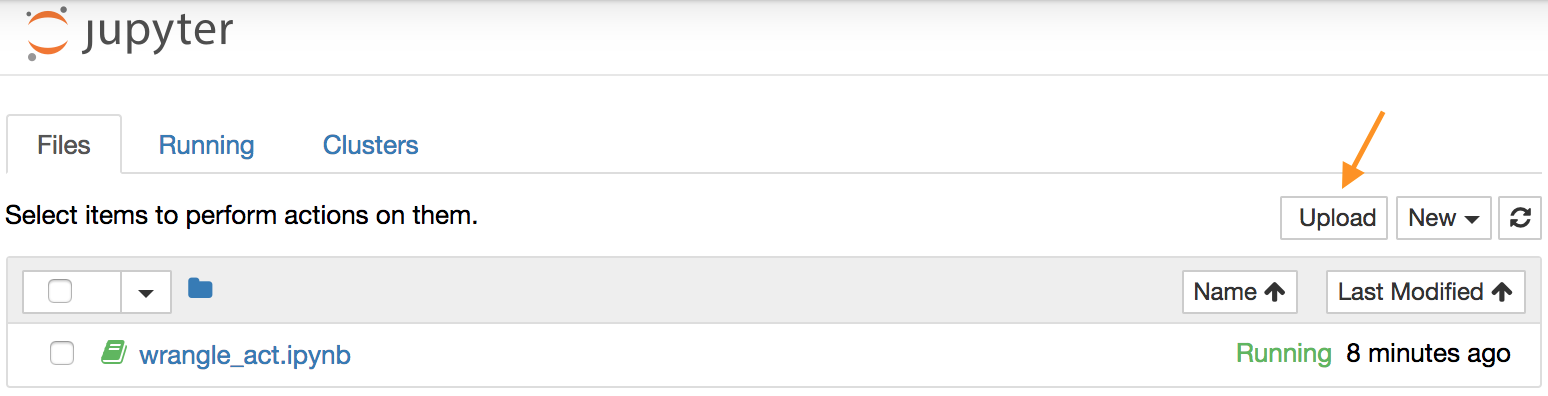
In this step, you will gather **all**three pieces of data as described below in the "Data Gathering" section in the wrangle\_act.ipynb notebook.

**Note:** the methods required to gather each data are different.

### The WeRateDogs Twitter archive

I am giving this file to you, so imagine it as a file on hand. Download this file manually by clicking the following link: [**twitter\_archive\_enhanced.csv**](https://d17h27t6h515a5.cloudfront.net/topher/2017/August/59a4e958_twitter-archive-enhanced/twitter-archive-enhanced.csv). Once it is downloaded, upload it and read the data into a pandas DataFrame.

If you decide to complete your project in the Project Workspace, note that you can upload files to the Jupyter Notebook Workspace by clicking the "Upload" button in the top right-hand corner of the dashboard.



Jupyter Notebook dashboard with arrow pointing to the "Upload" button

### The tweet image predictions

This file (image\_predictions.tsv) is present in each tweet according to a neural network. It is hosted on Udacity's servers and should be downloaded programmatically using the [**Requests**](https://pypi.org/project/requests/) library and the following URL: [**https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad\_image-predictions/image-predictions.tsv**](https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv)

### Additional data from the Twitter API

Gather **each tweet's retweet count** and **favorite ("like") count** at the minimum and any additional data you find interesting. Using the tweet IDs in the WeRateDogs Twitter archive, query the Twitter API for each tweet's JSON data using Python's **[Tweepy](http://www.tweepy.org/" \t "_blank)** library and store each tweet's entire set of JSON data in a file called tweet\_json.txt file.

Each tweet's JSON data should be written to its own line. Then read this .txt file line by line into a pandas DataFrame with (at minimum) **tweet ID, retweet count, and favorite count**. Note: do not include your Twitter API keys, secrets, and tokens in your project submission.

**Note:** if you are not familiar with querying data from an API, you can read the next page on Twitter API, which gives a tutorial on how to Query Twitter Data.

## How to Query Twitter Data

In this project, you'll be using **[Tweepy](http://www.tweepy.org/" \t "_blank)** to query Twitter's API for additional data beyond the data included in the WeRateDogs Twitter archive. This additional data will include retweet count and favorite count.

Some APIs are completely open, like MediaWiki (accessed via the **[wptools](https://github.com/siznax/wptools/wiki" \t "_blank)** library) in Lesson 2. Others require authentication. The Twitter API is one that requires users to be authorized to use it. This means that before you can run your API querying code, you need to set up your own Twitter application. Here are the steps to do that on the Twitter site:

* First, if you do not already have one, you need to [**sign up for a Twitter account**](https://help.twitter.com/en/create-twitter-account).
* Next, to set up a developer account, follow the directions on [**Twitter’s Developer Portal, in the “How to Apply” section**](https://developer.twitter.com/en/docs/basics/developer-portal/overview).
* You will be guided through the steps, and asked to describe in your own words what you are building. Here is some suggested language you can use: "As a Udacity student, I need to access the Twitter API in order to complete a Data Wrangling student project. In this project, I'll be using Tweepy to query Twitter's API for data included in the WeRateDogs Twitter archive. This data will include retweet count and favorite count. Before I can run my API querying code, I need to set up my own Twitter application. Once I have this set up, I will develop some code to create an API object that I'll use to gather Twitter data. After querying each tweet ID, I will write its JSON data to a tweet\_json.txt file with each tweet's JSON data on its own line. I will then read this file, line by line, to create a pandas DataFrame that I will assess and clean. I may post this completed project on my GitHub account, where it will get viewers. Otherwise there will be no other readers or users of my Twitter data or project analysis beyond the Udacity instructors and reviewers."
* Once you submit your application, you should soon receive an email from Twitter letting you know they have approved your new Twitter developer account. Follow the link in the email from Twitter to a page of directions to get started creating your app.
* If you are asked for an app name, it can be anything appropriate, and if you’re asked for a Website URL, it can be anything in a standard URL format. You can do the same with other requested URLs, or perhaps leave them blank.
* If you’re asked to explain how your app will be used, you could say something like "I'm creating this for a student Data Wrangling project with Udacity, where we need to query and analyze Twitter data from WeRateDogs."
* You should then be given a Success message, and a new developer page displayed to you where you can manage your app.
* You can then go to the Keys and Tokens tab on this page to find or generate the Consumer API keys, and the Access Token and Access Token Secret that you will need.

**Note**: If you have any trouble creating this Twitter account or accessing the data, please see the section at the bottom of this page "Accessing Project Data Without a Twitter Account."

Once you have your Twitter account and Twitter app set up, the following code, which is provided in the [**Getting started**](https://docs.tweepy.org/en/latest/getting_started.html) portion of the Tweepy documentation, will create an API object that you can use to gather Twitter data.

**import** tweepy

consumer\_key = 'YOUR CONSUMER KEY'

consumer\_secret = 'YOUR CONSUMER SECRET'

access\_token = 'YOUR ACCESS TOKEN'

access\_secret = 'YOUR ACCESS SECRET'

auth = tweepy.OAuthHandler(consumer\_key, consumer\_secret)

auth.set\_access\_token(access\_token, access\_secret)

api = tweepy.API(auth)

Tweet data is stored in JSON format by Twitter. Getting tweet JSON data via tweet ID using Tweepy is described well in this **[StackOverflow answer](https://stackoverflow.com/questions/28384588/twitter-api-get-tweets-with-specific-id" \t "_blank)**. Note that setting the tweet\_mode parameter to 'extended' in the get\_status call, i.e., api.get\_status(tweet\_id, tweet\_mode='extended'), can be useful.

Also, note that the tweets corresponding to a few tweet IDs in the archive may have been deleted. [**Try-except blocks**](https://wiki.python.org/moin/HandlingExceptions) may come in handy here.

## Do Not Include Your API Keys, Secrets, and Tokens in Your Submission

Do **not** include your API keys, secrets, and tokens in your project submission. This is standard practice for APIs and public code.

## Twitter's Rate Limit

Twitter's API has a rate limit. Rate limiting is used to control the rate of traffic sent or received by a server. As per [**Twitter's rate limiting info page**](https://developer.twitter.com/en/docs/basics/rate-limiting):

Rate limits are divided into 15 minute intervals

To query all of the tweet IDs in the WeRateDogs Twitter archive, 20-30 minutes of running time can be expected. Printing out each tweet ID after it was queried and [**using a code timer**](https://stackoverflow.com/questions/7370801/measure-time-elapsed-in-python) were both helpful for sanity reasons. Setting the wait\_on\_rate\_limit and wait\_on\_rate\_limit\_notify parameters to True in the **[tweepy.api class](http://docs.tweepy.org/en/v3.2.0/api.html" \l "API" \t "_blank)** is useful as well.

## Writing and Reading Twitter JSON

After querying each tweet ID, you will write its JSON data to the required tweet\_json.txt file with each tweet's JSON data on its own line. You will then read this file, line by line, to create a pandas DataFrame that you will soon assess and clean. This [**Reading and Writing JSON to a File in Python**](http://stackabuse.com/reading-and-writing-json-to-a-file-in-python/) article from Stack Abuse, will be useful.

## Accessing Project Data Without a Twitter Account

If you can't set up a Twitter developer account using the steps above, or you prefer not to create a Twitter account for some reason, you may instead follow the directions below to access the data necessary for the project. **Note**: We recommend that you follow the steps above to access the data using a Twitter developer account, because with the shortcut detailed below, you will miss practicing the valuable skill of gathering this data on your own. However, Twitter's updated process may not work for everyone, and we realize there are legitimate reasons that some students may prefer this approach, so we provide it to you here. **You choose the approach to access the data that works best for you. This shortcut approach will certainly work for you to pass the project equally well.**

#### Directions for accessing the Twitter data without actually creating a Twitter account:

At the bottom of this page you can find two files you can download:

* twitter\_api.py: This is the Twitter API code to gather some of the required data for the project. Read the code and comments, understand how the code works, then copy and paste it into your notebook.
* tweet\_json.txt: This is the resulting data from twitter\_api.py. You can proceed with the following part of "Gathering Data for this Project" on the Project Details page: "Then read this tweet\_json.txt file line by line into a pandas DataFrame with (at minimum) tweet ID, retweet count, and favorite count."

**Step 2: Assessing Data**

After gathering all three pieces of data, assess them visually and programmatically for quality and tidiness issues. Detect and document at least **eight (8) quality issues** and **two (2) tidiness issues** in the "Accessing Data" section in the wrangle\_act.ipynb Jupyter Notebook.

You need to use two types of assessment:

* **Visual assessment:** each piece of gathered data is displayed in the Jupyter Notebook for visual assessment purposes. Once displayed, data can additionally be assessed in an external application (e.g. Excel, text editor).
* **Programmatic assessment:**pandas' functions and/or methods are used to assess the data.

To meet specifications, the following issues must be assessed.

* You only want original ratings (no retweets) that have images. Though there are 5000+ tweets in the dataset, not all are dog ratings and some are retweets.
* Assessing and cleaning the entire dataset completely would require a lot of time, and is not necessary to practice and demonstrate your skills in data wrangling. Therefore, the requirements of this project are only to assess and clean at least 8 quality issues and at least 2 tidiness issues in this dataset.
* The fact that the rating numerators are greater than the denominators does not need to be cleaned. This [**unique rating system**](http://knowyourmeme.com/memes/theyre-good-dogs-brent) is a big part of the popularity of WeRateDogs.
* You do *not* need to gather the tweets beyond August 1st, 2017. You can, but note that you won't be able to gather the image predictions for these tweets since you don't have access to the algorithm used.

If you need some help with the datasets, you can read the page: *Project Motivation.*

**Step 2: Cleaning Data**

Clean **all** of the issues you documented while assessing. Perform this cleaning in the "Cleaning Data" section in the wrangle\_act.ipynb.

Make sure you complete the following items in this step.

* Before you perform the cleaning, you will make a copy of the original data.
* During cleaning, use the define-code-test framework and clearly document it.
* Cleaning includes merging individual pieces of data according to the rules of [**tidy data**](https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html). The result should be a high-quality and tidy **master pandas DataFrame** (or DataFrames, if appropriate).

## Step 4: Storing Data

In the "Storing Data" section in the wrangle\_act.ipynb notebook, store the cleaned master DataFrame in a CSV file with the main one named twitter\_archive\_master.csv. If additional files exist because multiple tables are required for tidiness, name these files appropriately. Additionally, you may store the cleaned data in a SQLite database (which is to be submitted as well if you do).

**Step 5: Analyzing and Visualizing Data**

In the Analyzing and Visualizing Data section in your wrangle\_act.ipynb Jupyter Notebook, analyze and visualize your wrangled data.

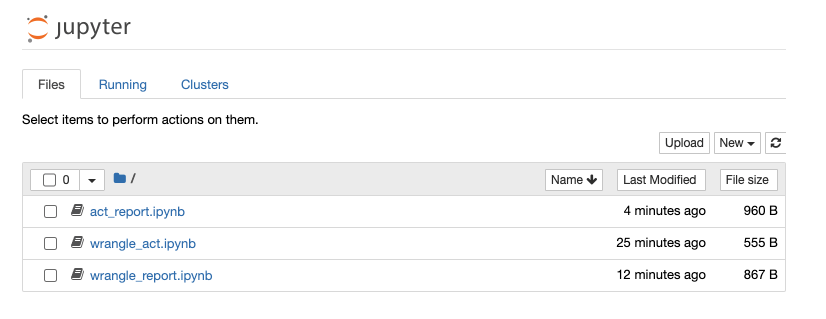
* You must produce at least **three (3) insights and one (1) visualization**.
* You must clearly document the piece of assessed and cleaned (if necessary) data used to make each analysis and visualization.

## Step 6: Reporting

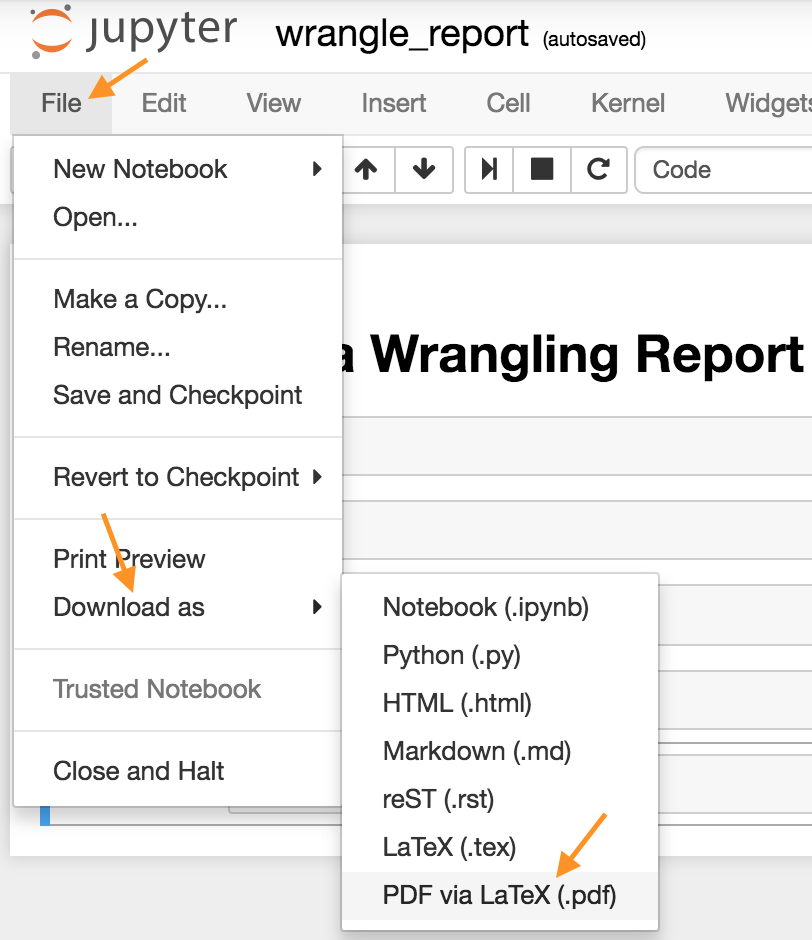
Create a **300-600 word written report** called wrangle\_report.pdf or wrangle\_report.html that briefly describes your wrangling efforts. This is to be framed as an internal document.

Create a **250-word-minimum written report** called act\_report.pdf or act\_report.html that communicates **all** the insights and displays the visualization(s) produced from your wrangled data. This is to be framed as an external document, like a blog post or magazine article, for example.

Both of these documents can be created in separate Jupyter Notebooks using the [**Markdown functionality**](http://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Working%20With%20Markdown%20Cells.html) of Jupyter Notebooks, then downloading those notebooks as PDF files or HTML files (see image below). If you want to write a report this way, there are two Jupyter Notebook files, named "wrangle\_report" and "act\_report" for you in the workspace. You might prefer to use a word processor like Google Docs or Microsoft Word, however.



Files in the workspace



**Download Jupyter Notebook in PDF form**