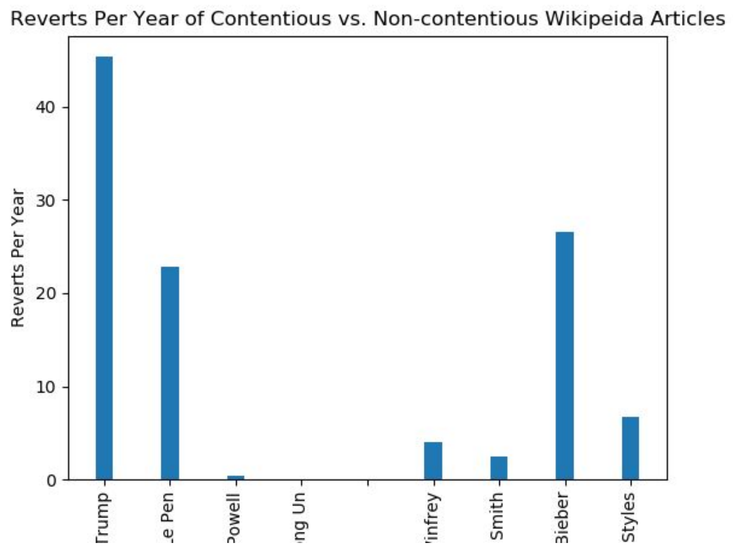
Emma Mack
Miniproject 3: Writeup and reflection

Project Overview
In middle school, I was taught more than a handful of times that *Wikipedia is not a trustworthy source.* Anyone can edit it, meaning that any misinformation can stay on a page as if it were fact until some good samaritan with a Wikipedia account comes along to fix it. So, I parsed the html of wikipedia edit history pages, pulled out reverted edits and relevant dates and did some math. With this project, I aimed to estimate the amount of misinformation that tends to be on a wikipedia page, and whether more politically contentious subjects tend to have more misinformation than non-contentious ones.

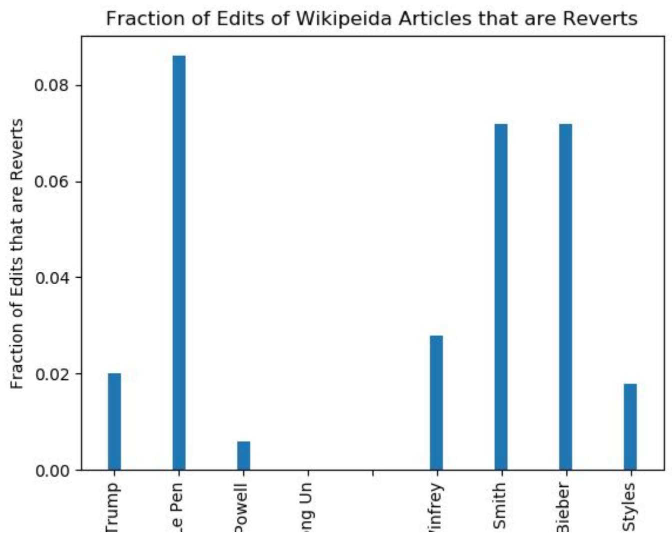
Implementation
The program ran as such:

- Create an arbitrary number of "wikipedia edit page" objects from a list of names. The objects have methods such as "html", which finds the link of the edits page of the given name and loads it using BeautifulSoup, and "text" which pulls all text from the html.
- Find the percentage of total edits that are reverts in each page. Each revert in the edits page is labeled with a tag "revert" so it is simply a matter of counting all the instances of the word in the text.
- Find the reverts per year for a different perspective on the data. There were two key pieces of information that I needed to perform this analysis. I needed the number of reverted edits on a page, and some sort of time indicator to calculate the amount of time that those edits were contained in. The program finds all the "revert" tags, as well as finding the last date on the page. Along with today's date, it is able to find the time window the page spans, and estimate the total reverts per year of the page.
- Compile the data into a MatPlotLib graph.

Results
According to my analysis, the number of reverted edits on a certain wikipedia page depends on much more than the amount of contention that surrounds the subject. Looking at just "reverts per year", reverted edits could line up with the total number of edits to the page in a year.



More reverted edits total would mean more misinformation. However, the fraction of reverts tells a clearer story of the character of the edits on a whole.



There are certainly some pages that have a much, much higher fraction of reverts than others. However, if my hypothesis were right, the entries on the left side of the graph should all be higher than the entries on the right, whereas they seem reasonably split. My guess is that Wikipedia subjects (Matt Smith, Justin Bieber) with "fangirls", i.e. younger users who feel the need to add to Wikipedia pages without the knowledge of how to source their additions, will have just as high a fraction of reverted edits as politically contentious ones.

Alignment

The process the program follows, and the tools that I used, line up well with the way I initially planned to answer the question on misinformation. However, the simple metric of fraction/number of reverted edits is a gross simplification of misinformation. It is a proxy, yes, but edits can be reverted for any number of reason, including trivial things like spelling and punctuation. And an edit containing misinformation could go un-reverted.

A more robust program would follow the "details" link on each reverted edit and determine the reason for each one. It would then throw out the trivial edits and compile data on the ones cited as poorly sourced or incorrect.

Reflection

The two major drawbacks of this project are the simplification detailed above and the fact that it was much slower than it needed to be. Parsing html is not fast, and in an ideal world I would have been able to use some Wikipedia library to do this work. The one that was given in the project description was not powerful enough, but there must be better ones out there.

In my final program, you will see no unit tests or data pickling. I pickled for a while, but then decided switching between programs was wasting more time than it was saving and removed it. I also got very frustrated with doctests. They involve much more typing than simply testing each function manually, don’t make much sense for a program with unpredictable output, and make debugging much harder since the suppress print statements that come before an error message. I ended up not using them, but you can trust that I did test each function as I went by calling it from __main__.

All this being said, the project was appropriately scoped, and I feel like I learned a lot about saving data and scraping html.