MSBX 5420 Team Project

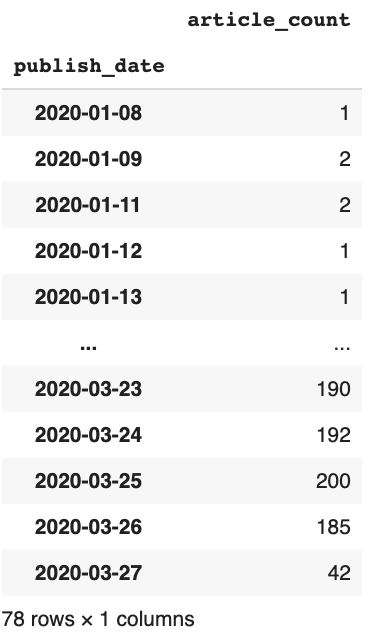
April 25, 2020

Team Torrey’s Peak

**Design Document**

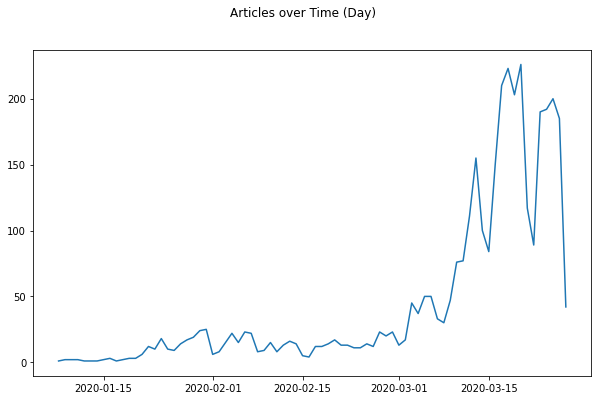
*Descriptive Stats*

Team Torrey’s Peak used the COVID-19 dataset comprising more than 3500 CBC news articles for our MSBX 5420 team project. This dataset retrieved from kaggle contains the authors, the title, the publish date, the description about the story, the main story and the url. For the start of this project, we wanted a closer look into the descriptive statistics within the dataset. With a goal in mind to focus on the sum of articles over time, we first created a line of code to put the publish dates into the correct date form. Next, we created an index naming this news\_2020 to adjust the publish date to January 1, 2020 and created a new column to count how many articles were in each day. Figure A shows the results of the new article count.



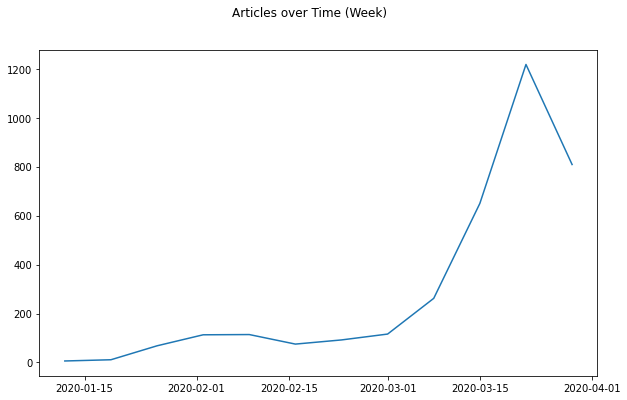
**Figure A**

With this new column, we imported matplotlib and pivoted the index. Using the plt.subplot function, we set the figsize to 10 and 6. Also giving the suptitle “Articles over Time (Day).”



**Figure B**

The next step into our descriptive statistics looked into the sum of the number of articles per week. Using the news\_2020 index, we grabbed the publish\_date and created a weekly\_count index using resample(‘W’).sum(). Then, we plotted the articles over time once again using the subplots and figsize to 10 and 6. Also giving the suptitle the name “Articles over Time (Week).” Finally plotting the weekly count and article count shown below in Figure C.



**Figure C**

*Word Cloud*

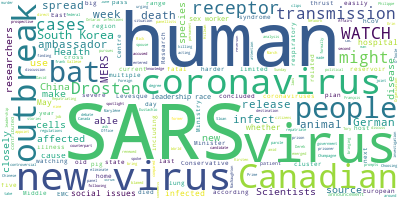
Created word clouds for all text in the articles and also created word clouds for December, January, February and March to see if the articles had changed over time. Used Python to read the csv data and used the publish\_date as the index. Found 15.8M words in the text of the articles and created a total word cloud.



**Figure D**

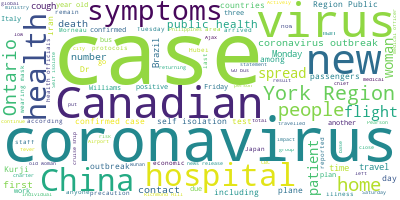
After running a word cloud for all articles, we split up the articles by month.

Here is December.



**Figure E**

And March.



**Figure F**

*Spark Repetitive Words*

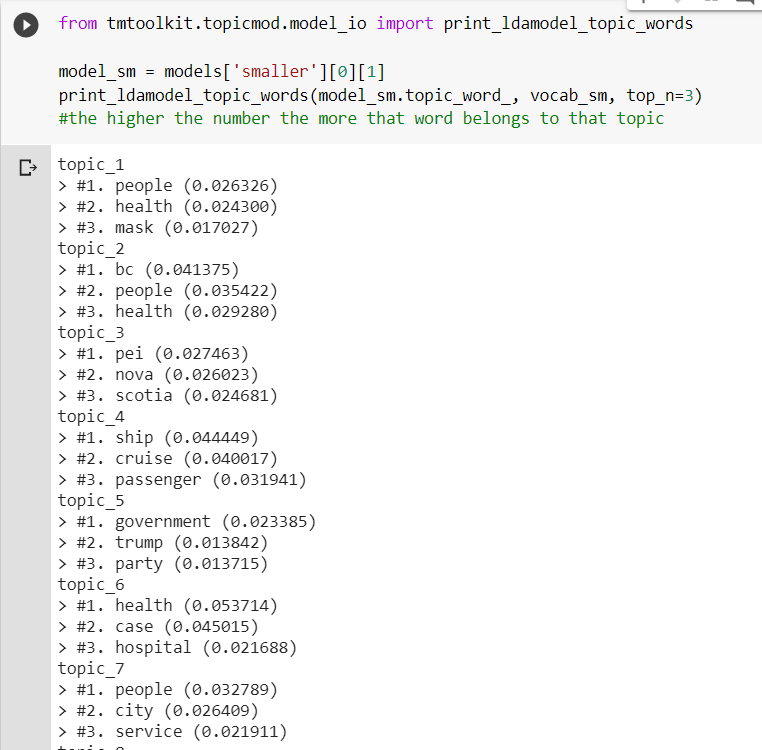
Since almost everything in our dataset are words (strings), and we learned how to use spark to do word count through this semester, so we decided to count occurrences of all words that were used in all these news. We did it in two ways, one with pyspark rdd and the other with pyspark dataframe. With pyspark rdd, we applied the trick that Prof. Zhang showed to us which is split words by delimiter “(space)” for all news with flatmap function; then we assign ones for all words and count them for the same words. With pyspark Dataframe, we used the same logic, but just with different functions since the object type has changed. Also for fun, we counted number of words for each news by counting the length of each string after we convert them into lists.

*Topic Modeling*

For the bulk of this project, we decided that it would be interesting to create a topic model that analyzed the text of all of these articles and would create 12 topics based off of words within these articles. By first creating a jupyter notebook, we install tmtoolkit. From there, a corpus of all text from the articles needed to be created, and this was done by using csv.reader and with a for loop that appended text from each article. The length of this corpus was 3567, which matches the amount of articles that were examined. We then imported nltk in order to remove stopwords from the corpus since these words do not give any insight into the topics of the articles. TMPreproc was also downloaded to tokenize, remove special characters, add stopwords that we deemed should be avoided like ‘http’ or ‘nt.’ In addition, we also converted the corpus to lowercase.

*Small - Topic Model*

In order to look more specifically into these topic models, we divided them into a smaller (more specific) and bigger (less specific) half. By doing this, we were able to see which topic model would perform better. With the smaller topic model, we removed uncommon tokens with a threshold of 0.01 and removed common tokens with a threshold of 0.9. This was to ensure that any word that was rarely used or overused would not be included in the topic model. A numPy array was created with the words and from there an LDA model was run with a number of topics of 12, beta of 0.5, and number of iterations of 1000. We played with these parameters, and these appeared to give us the best ‘smaller’ topic model. Below in Figure I , we provide a preview of the topics and includes some topics from Canada given that our articles were from a Canadian news company. Figure I demonstrates several interesting topics that could give insight into the Coronavirus outbreak such as *cruise* and *passengers* as well as *health* and *masks*.

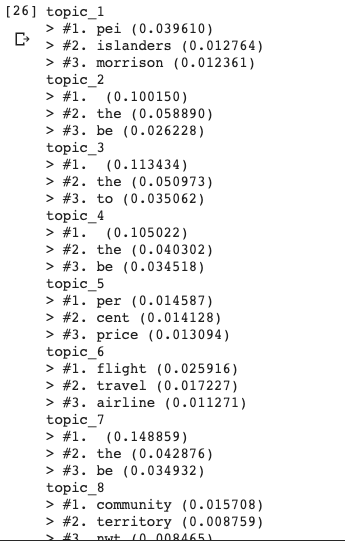


**Figure I**

*Big - Topic Model*

For the Big Topic Model, we relaxed some of the filters of the Smaller Topic Model to see if we could capture more variance or topics in the corpus. The bigger topic model still includes position tagging, tokenizing, lemmatizing, converting tokens to lowercase, and the removal of special characters and stopwords. However, the bigger model does not include some of the more robust preprocessing such as filtering positions and removing uncommon tokens. Figure J shows the output of the bigger topic model. As we can see, while some topics such as Topic 6 (“flight”, “airline”, and “travel”) do illustrate the major topics or themes of these documents, many of the topics (such as Topics 4, 5, 7) are too broad to provide a clear understanding of the information in these articles. Thus, the smaller topic models performs much better.

**Figure J**

****