Step 1: Identify all possible 10K forms

- Notes
  - The normal search page proved too difficult to use
  - The SEC Edgar database has an Archive site: sec.gov/Archives/edgar/full-index/
    - This contains a list of all filings the SEC has received in the past 2ish decades
    - These are quarterly, not yearly, files
    - Most of the files are functionally the same, but compressed differently

Step 2: Download all 10K forms since 2016

- Relevant Files I made:
  - virginiaSECscraping.R
  - SEC10KList.csv
  - remaining-SEC10KList.csv
- Notes
  - Edgar needs a very specific header in order to permit downloads. Neil first let me in on this, though I made some tweaks. This is set in the dowloadEdgarTxt function
  - If using a windows computer, downloads can be weird. The download.file call specifies “wb” mode to resolve this, so don’t get rid of it unless running on Linux
  - Downloading the compressed versions and unzipping them locally proved fastest after trial and error

Step 3: Interpret all downloaded 10K forms

- Relevant Files I made:
  - virginiaKeywordIdentification.R
  - combinedSummaryData.csv
- General approach:
  - Get rid of all HTML tags, and reduce duplicated whitespace
  - Split the single string the file started as into a bunch of ‘sentences’, using periods followed by whitespace as a separator
  - Find all sentences that match at least one of the targets
  - Count up the number of times each target occurs
- Issues
  - Although the forms look like HTML and usually render in a browser, they have formatting errors that prevent interpretation in R using standard HTML packages
    - Because of this, the best way I found to separate the files into chunks was to clear all the tags and split the file into pieces wherever there’s a period followed by whitespace. This approach had its own issues, such as:
      - Forcing sections like tables into hard-to-read continuous prose
      - Resulting in some ridiculously long sentences that exceed the 32k character limit for CSVs
  - The larger files are mostly full of garbage, which I was unable to reliably identify and purge

File and Function Overview

<table>
<tr>
<td rowspan="2">virginiaSECscraping.R</td>
<td>getFilings(startYear, endYear, filingType)

Searches over the sec.gov/Archives/edgar/full-index/ site, which stores lists of all SEC filings sorted by year and quarter

Returns a list of all filings for the specified time range and filing type. The list contains the location of the filings for future downloads, and can be saved as a CSV for the downloadAllEdgarTxts function to read</td>
</tr>
<tr>
<td>downloadAllEdgarTxts(csvOfTargets)

Uses the csv file generated above to download filings for all targets specified.
Includes a timing function to make sure it doesn’t request more than 10 files per second, which is against Edgar policy and could cause problems.
Periodically saves a list of remaining files to download

Returns nothing, but saves all files to the active directory</td>
</tr>
<tr>
<td>virginiaKeywordIdentification.R</td>
<td>identifyKeywordSentences(filename, targetExpressions)

Opens the supplied file, removes html tags, trims excess whitespace, splits the file into “sentences” using periods followed by whitespace as separators, then looks for each instance of the supplied regular expressions

Returns a dataframe where the first row is a summary of how many times each target regular expression appeared in the file, and each following row represents a sentence where at least one target showed up. Example:
<table>
<tr><th>(rowname)</th><th>Sentence Number</th><th>target1</th><th>target2</th><th>sentenceText</th><th>filename</th><th>rowType</th></tr>
<tr><td>[filename]-Total</td><td>NA</td><td>5</td><td>8</td><td></td><td>[filename]</td><td>Total</td></tr>
<tr><td>[filename]-S1</td><td>12</td><td>5</td><td>2</td><td>[sentence where targets 1 and 2 were found]</td><td>[filename]</td><td>Detail</td></tr>
<tr><td>[filename]-S2</td><td>55</td><td>0</td><td>6</td><td>[sentence where target 2 was found]</td><td>[filename]</td><td>Detail</td></tr>
</table>
</td>
</tr>
</table>