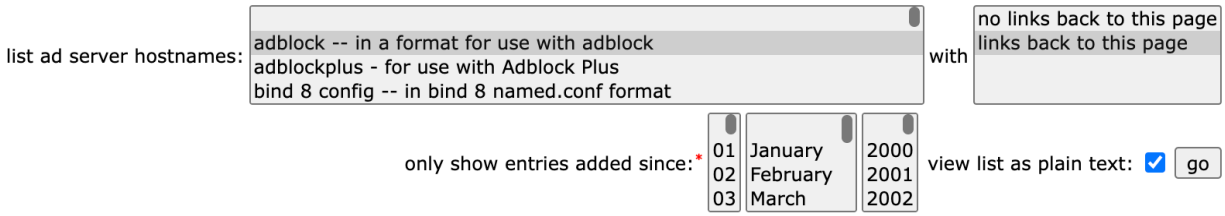
Data Collection Problem Set

Environment:

Python 3.9.18
Required modules are listed in requirements.txt.
Adblocklist.txt is downloaded from https://pgl.yoyo.org/adservers/ in the following path.



Data Loading:

Read domainsToInspect.csv as DataFrame, and Adblocklist.txt as a list.

WebDriver Initialization:

Challenge 1: Some websites will detect bot browsing with request or selenium. Then the ads will not be loaded as it should be.
Solution 1: When setting options for driver, disable the automation indicator web driver flags. Also, set header in options like a human visitor to a website.

Iframe Detectation:

Most of ads tends to be dynamic. I detect iframes on the website and compare the src of the iframe with the ad block list using regular expression match. Create an counter for precense of different ad servers.

Challenge 2: Potential instability of internet will influence the accessibility of a website.
Solution 2: Set a parameter PATIENCE which is the maximum number of times the web driver would try to get information from the specific websites. Also, use try except to avoid some unpredicted glitches due to non-syntax error.

Challenge 3: Servers of some websites may have bad connection. The webdriver may finish loading the page before the ad is fully loaded.
Solution 3: Use time.sleep() or WebDriverWait until expected condition is fulfilled.

Challenge 4: Some websites have ad anti-detect mechanism. They put ads in a nested iframe where we need to look for additional iframe inside the outer iframe to get the information of the ad.
Solution 4: Design a recursive nested extractor which will look inside each iframe using Depth-First-Search. Then record src of nested iframe in counter.

Challenge 5: For those websites without ad anti-detect mechanism, applying solution 4 may increase the number of ads detected by accident, since some related ads tends to be presented in an ad server and we don't want to recount those presence.
Solution 5: Here is an assumption that if the developer used the mechanism to avoid ad block, then all the ads they intended to put on the website would be set in the same layer. Here are couple of reason. If the developer found a way to avoid ad detection, he would

make all the ads “safe” instead of putting part of them on the most outer iframe and rest in the nested layer. We don’t know on which layer this developer put ads, so we will consider the count of the layer where first ads could be found.

Analysis:

Save ad server information as a tsv file for using tab split when storing. Read it as DataFrame and merge it with domainsToInspect on domain. Use plt.hist to draw the histogram.

Project DATA Problem Set

RQ1.

Data Cleaning:

For “Text” and “Title” in Ad data, perform text preprocessing which includes converting text to lowercase, removing punctuation, removing stop words, removing contractions for clarity.

Topic Modeling:

Using document containing term frequency using tools like TF-IDF.
Use NLP techniques like Latent Dirichlet Allocation which works in topic discovery to automatically classify election fraud relevant topics from the constructed frequency document.

Sentiment Analysis:

Use a pre-trained sentiment analysis model or train a model on labeled data to classify the sentiment of each ad as positive, negative, or neutral.
Specially, for the ads that are classified under topics related to election fraud, perform a detailed sentiment analysis to understand the overall sentiment.

Pseudo code:

```
# Text cleaning
ads['cleaned_text'] = ads['Text'].apply(preprocess_text_function)

# Topic Modeling
freq_vectorizer = TfidfVectorizer()
text_matrix = freq_vectorizer.fit_transform(ads['cleaned_text'])
lda_topics = LDA().fit_transform(text_matrix)

# Sentiment Analysis
sentiments = sentiment_analysis_model.predict(ads['cleaned_text'])
ads['sentiment'] = sentiments
```

RQ2.

Data Processing:

Read data as DataFrame. Merge the Ad data and survey data on "User_ID" to associate demographic information with ad exposure data. Handle missing values, outliers to ensure data consistency.

Descriptive Analysis:

1.Calculate frequency and percentage of ad exposures of demographic variables (gender, age, race), income, party ID, geographic variables (state, zipcode).
2.Draw bar chart and pie chart for some variables and draw heatmap for geographic variables to show the distribution of targeted population.
3.Create some cross-tabulations showing relationship between ads-exposure and variables like ages, races, incomes and party ID.

4.Variables like past voting behavior, engagement with political content or internet use frequency would be some valuable covariates in this research.

Pseudo code:

```
# Data Processing
merged_data = pd.merge(ad_data, survey_data, on='User_ID')

# Descriptive Analysis
# creating bar chart, pie chart and heatmap showing the frequency.
# creating cross-tabulations to show relationships
descriptive_stats = merged_data.groupby(['gender', 'age', 'race', 'income', 'party_id',
'state']).agg({'Ad_ID': 'count'})

# visualization
visualize(merged_data)
```

RQ3.

Statistical Analysis:

1. Consider ad exposure to election fraud ads as predictor. This can be a binary indicator (exposed or not) or number of times exposed.
2. Consider Turnout, Wide_Fraud, Ballot_Tampered as responses.
3. Identify potential covariates that could confound the relationship between ad exposure and belief in election fraud, for example, those potential covariates mentioned in RQ2 and some basic information recorded in Survey 1.
4. Applied inferences like t-test, F-test(ANOVA), Chi-Square and logistic regression analysis to figure out the potential relationship.

Pseudo Code:

```
# correlation
correlation_matrix = data.corr()

# chi-square analysis
chi2, p, dof, expected = chi2_contingency(age_ads_cross_tab)

# t-test
t_stat, p_value = ttest_ind(low_income_ads, high_income_ads)

# ANOVA
model = ols(data).fit()
anova_results = sm.stats.anova_lm(model, typ=ANOVA)

# Logistic Analysis
logit_model = sm.Logit(Y, X)
result = logit_model.fit()
```

Mock Results of Analysis:

T-test:

Instance: Ad exposure or not on Wide_Fraud
Results: t_stat = 3, p_value = 0.001 (lower than significant level alpha=0.05)
Interpretation: There is a statistically significant difference in the belief in wide fraud between those exposed to election fraud ads and those not exposed, with the exposed group showing higher belief levels.

ANOVA:

Instance: Ad Exposure Frequency on Turnout among Different Age Groups
Results: F-stat = 4, p-value = 0.001
Interpretation: There is a statistically significant difference in turnout among different age groups based on the frequency of ad exposure. The effect of ad exposure on turnout varies by age group.

Chi-Squre:

Instance: Ad Exposure vs. Party ID on Turnout
Results: chi2-stat = 5, p-value = 0.001
Interpretation: There is a statistically significant association between Party ID and turnout among those exposed to election fraud ads.

Logistic Regression Analysis:

Instance: Effect of Ad Exposure on Believing Ballot Tampering (With covariates mentioned in RQ2)
Results: Odd Ratio(Ad Exposure): 1.5, p-value(Ad exposure)=0.001
Interpretation: After controlling for past voting behavior, political engagement, and internet use frequency, there is a 50% increase in the odds of believing in ballot tampering for individuals exposed to election fraud ads.

Referennce:

https://www.analyticsvidhya.com/blog/2022/01/text-cleaning-methods-in-nlp/
https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation
https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/#:~:text=Term%20Frequency%20%2D%20Inverse%20Document%20Frequency%20(TF%2DIDF)%20is,%2C%20relative%20to%20a%20corpus).