Potential problems with the data:

1. Our data is based on a fairly well-reputed sources, namely, the Fortune 1000 company ranking, tl Harvard Business Review. As many other websites and companies base their evaluations on the decision to use information found in these websites as reliable data. Nevertheless, while these a rankings and information are known to be legitimate, it is fair to recognize that these sources ar such as the different companies who may be donors to these sources. However, we feel that du both the Glassdoor and HBR CEO rankings, this helps account for this bias somewhat. Another v fact that existing on a ranking system can be an indicator in of itself. With company on 1000 co reputable source is a variable which can compare different companies and their results.
2. Our data is imperfect, and some missing values do exist due to the fact that we scraped from w websites difficult to scrape from. This can be resolves by rescraping the data at a slower pace, b of data, it is also a possibility to simply remove those values. Otherwise, thanks to the reputabili we need to do is cleaning to allow for a clean merging of our data.

Suitability of your data for answering the questions in the proposal. If it is not suit acquire new data as necessary.

Our intial proposal was primarily focused on collecting profile data for individuals and how the overall can help us predict decision biases.

However, the data required for our original proposal is expensive to collect in the timeframe given, thu focus instead on collecting profile data on high profile individuals namely C-suite executives and Boar vital company statistics, CEO approval ratings/rankings, board of executives composition and diversit describing the type of community the company exists in.

Data on CEO rankings and approval ratings are scraped from both Glassdoor and the Harvard Busines Magazine website and Wall Street Journal data regarding assets, revenue, profit and its relative chang approach data is much more accessible yet still can help us effectively answer a piece of our orginal

Proposal Revisions

Where as before we were eager to understand the impacts that a particular C-suite board's affiliations making of a company and how that would bias their ultimate decision, with this data we have elected and his or her board members would have on the general performance of a company, and to understa that performance according to different kind metrics such as profitability or assets under possession

Although much simpler than our original proposal question, solving this simplified profile c-suite/com create hypothesis and models that explain and predict bias in organizational deicision making. We pla preliminary project to help us begin desigining weighting algorithims to predict said biases for the nex

Interesting resources we've focused on:

- Info to scrape
    - https://hbr.org/2019/11/the-ceo-100-2019-edition
    - https://www.ceotodaymagazine.com/top-50-ceos/
    - https://fortune.com/fortune500/2019/search/
    - https://www.usnews.com/news/best-countries/articles/2018-01-23/people-around-the-w world-leaders
    - https://www.glassdoor.com/Award/Top-CEOs-2018-LST_KQ0,13.htm
- References:
    - https://business.linkedin.com/talent-solutions/blog/trends-and-research/2018/what-1200
    - https://www.glassdoor.com/Award/Top-CEOs-LST_KQ0,8.htm

Scraping

Scraping the Fortune 1000 companies

```
import re
import time
import requests
from bs4 import BeautifulSoup
import matplotlib.pyplot as plt
import numpy as np
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.common.exceptions import NoSuchElementException
import pickle
from time import sleep
import pandas as pd

browser = webdriver.Chrome('chromedriver.exe')
base_url="https://fortune.com/fortune500/2019/search/"

page_data = []   #data found on each page
CEO_data = []    #data found on the page describing the CEO and further details
try:
    browser.get(base_url)
    next_page  = browser.find_elements_by_xpath('//button[@type="button"]')
    sleep(5)
    for p in range(10):
        print(p)
        browser.get(base_url)
        next_page  = browser.find_elements_by_xpath('//button[@type="button"]')
        sleep(5)
        for i in range(p):
            #find the next button, and move to the page we want to be on
            next_page  = browser.find_elements_by_xpath('//button[@type="button"]')
            next_page[-3].click()
        #find the hyperlinks
        hl = browser.find_elements_by_class_name('searchResults__cellWrapper--39MAj')
        hyperlinks1 = [w.get_attribute('href') for w in hl]
        hyperlinks2 = []
        for h in hyperlinks1:
            if h not in hyperlinks2:
                hyperlinks2.append(h)

        page_data.append([w.get_attribute('text') for w in hl])
        #getting the CEO data
        for h in hyperlinks2:
            browser.get(h)
            sleep(5)
            stuff = browser.find_elements_by_class_name('dataTable__row--34F3j')
            CEO_data.append([s.get_attribute('innerHTML') for s in stuff])

except NoSuchElementException:
    print("don't swear")
    raise


len(page_data) #check we got all the pages


comp_data = [[a[i:i+11] for i in range(0,len(a),11)] for a in page_data] #separate int
#clean up all the variables
index = np.concatenate(np.array(comp_data)[:,:,0])
companies = np.concatenate(np.array(comp_data)[:,:,1])
revenues = np.concatenate(np.array(comp_data)[:,:,2])
rev_perc_change = np.concatenate(np.array(comp_data)[:,:,3])
profit = np.concatenate(np.array(comp_data)[:,:,4])
prof_perc_change = np.concatenate(np.array(comp_data)[:,:,5])
assets = np.concatenate(np.array(comp_data)[:,:,6])
market_value = np.concatenate(np.array(comp_data)[:,:,7])
rank_change1000 = np.concatenate(np.array(comp_data)[:,:,8])
employees = np.concatenate(np.array(comp_data)[:,:,9])
rank_change500 = np.concatenate(np.array(comp_data)[:,:,10])
#create a pandas dataframe
companies = pd.DataFrame({'index':index,'companies':companies,'revenues':revenues,'rev
                         'prof_perc_change':prof_perc_change,'assets':assets,'market_v



CEO_data2 = [[re.findall(r">[\%\$\(\)\-\,\;\&\s\.a-zA-Z0-9]+<",t) for t in c] for c ir
#get the CEO names
ceo_names = []
for i in CEO_data2:
    if i != []:
        ceo_names.append(i[0][-1])
    else:
        ceo_names.append(np.nan)

#add this information to the dataframe
companies['ceos_page_data'] = CEO_data2
companies['ceo_name'] = ceo_names
companies.to_csv('fortune.com_companies.csv')
```

Scraping the WSJ S&P 500 Board Member Data

```
url = 'http://graphics.wsj.com/boards-of-directors-at-SP-500-companies/'

companies = browser.find_elements_by_xpath('//*[@companyName]')
info = []
for ii in companies:
    outerhtml = ii.get_attribute('outerHTML') #// to extract outerHTML
    tag_value=outerhtml.split("\" ") #// to extract board member info
    info.append(tag_value)

#organizing the board-member data
companynames = []
marketcap = []
industry = []
tot_dir = []
med_age = []
medpay = []
perc_woman = []
board_ind = []
tenure = []
for i in info:
    element_added = [False for i in range(9)]
    for e in i:
        if 'companyName' in e:
            companynames.append(e[13:])
            element_added[0] = True
        elif 'mktcap' in e:
            marketcap.append(e[8:])
            element_added[1] = True
        elif 'industry' in e:
            industry.append(e[10:])
            element_added[2] = True
        elif 'directorstotal' in e:
            tot_dir.append(e[16:])
            element_added[3] = True
        elif 'age' in e:
            med_age.append(e[5:])
            element_added[4] = True
        elif 'unrelated' in e:
            board_ind.append(e[11:])
            element_added[5] = True
        elif 'tenure' in e:
            tenure.append(e[8:])
            element_added[6] = True
        elif 'medianpay' in e:
            medpay.append(e[11:])
            element_added[7] = True
        elif 'female' in e:
            perc_woman.append(e[8:])
            element_added[8] = True
    for n,e in enumerate(element_added):
        if e == False:
            if n == 0:
                companynames.append(np.nan)
            elif n == 1:
                marketcap.append(np.nan)
            elif n == 2:
                industry.append(np.nan)
            elif n == 3:
                tot_dir.append(np.nan)
            elif n == 4:
                med_age.append(np.nan)
            elif n == 5:
                board_ind.append(np.nan)
            elif n == 6:
                tenure.append(np.nan)
            elif n == 7:
                medpay.append(np.nan)
            elif n == 8:
                perc_woman.append(np.nan)

#turn into dataframe and save as csv for cleaning
wsj = pd.DataFrame({'Company':companynames,'Market Cap':marketcap,'Industry':industry,
wsj.to_csv('wsjS&P.csv')
```

Scraping the Glassdoor CEO rankings

```
browser = webdriver.Chrome('chromedriver.exe')
url = 'https://www.glassdoor.com/Award/Top-CEOs-LST_KQ0,8.htm'
browser.get(url)
ids = browser.find_elements_by_xpath("//*[@class ='h2 m-0']")

#get the names from Glassdoor
glassdoor_name = []
for ii in ids:
    outerhtml = ii.get_attribute('outerHTML') #// to extract outerHTML
    glassdoor_name.append(outerhtml)
```

```
#get the corresponding companies
glassdoor_comp = []
companies = browser.find_elements_by_xpath("//*[@class ='mt-xsm mr-xl mb-0']")
for c in companies:
    glassdoor_comp.append(c.get_attribute('outerHTML'))

#clean out the CEO names and their approval rating
gd_names, gd_approve = [],[]
for j in range(len(glassdoor_name)):
    n,a = re.findall(r">[\%\$\(\)\-\,\;\&\s\.a-zA-Z0-9]+<",glassdoor_name[j])
    gd_names.append(n)
    gd_approve.append(a)

gd_comps = [re.findall(r">[\''\\\%\$\(\)\-\,\;\&\s\.a-zA-Z0-9]+<",glassdoor_comp[j])
```

Scraping the Harvard Business Review CEO Rankings

```
hbr = 'https://hbr.org/2019/11/the-ceo-100-2019-edition'
browser = webdriver.Chrome('chromedriver.exe')
browser.get(hbr)
hbr_name = browser.find_elements_by_xpath("//*[@class ='organisationname']")
hbr_info = browser.find_elements_by_xpath("//*[@class ='organisationinfo']")

#getting the hbr names and company
hbr_stuff = []
for h in hbr_info:
    outerhtml = h.get_attribute('outerHTML') #// to extract outerHTML
    hbr_stuff.append(outerhtml)

hbr_names, hbr_comp = [],[]
for j in range(len(hbr_ceo)):

    search = re.findall(r">[\''\\\%\$\(\)\-\,\;\&\s\.\wé’èüçîáa-zA-Z0-9]+<",hbr_ceo[j
    if len(search) == 2:
        n,a = re.findall(r">[\''\\\%\$\(\)\-\,\;\&\s\.\wé’üèçîáa-zA-Z0-9]+<",hbr_ceo[
    else:
        print(j)
        continue
    hbr_names.append(n)
    hbr_comp.append(a)

# get the variables found associated with the ranking
hbr_industry, hbr_country,hbr_year_started,hbr_insider,hbr_MBA,hbr_finrank, hbr_susta
for j in range(len(hbr_stuff)):
    print(j)
    search = re.findall(r">[\|\''\\\%\$\(\)\-\,\;\&\s\.\wé’èüçîáa-zA-Z0-9]+<",hbr_stu
    hbr_industry.append(search[0])
    hbr_country.append(search[1])
    hbr_country.append(search[1])
    hbr_year_started.append(search[2])
    hbr_insider.append(search[3])
    hbr_MBA.append(search[4])
    hbr_finrank.append(search[5])
    hbr_sustainalytics.append(search[6])
    hbr_csrhub.append(search[7])
```

```
#put all CEO data together
ceo_rank = pd.DataFrame({'GD_CEO':gd_names, 'GD_Approval':gd_approve,'GD_company':gd_

ceo_rank['HBR_CEO'] = hbr_names
ceo_rank['HBR_Company'] = hbr_comp
ceo_rank['HBR_Industry'] = hbr_industry
ceo_rank['HBR_Country'] = hbr_country
ceo_rank['HBR_YearStarted'] = hbr_year_started
ceo_rank['HBR_Insider'] = hbr_insider
ceo_rank['HBR_MBA'] = hbr_MBA
ceo_rank['HBR_finrank'] = hbr_finrank
ceo_rank['HBR_sustainalytics'] = hbr_sustainalytics
ceo_rank['HBR_csrhub'] = hbr_csrhub

ceo_rank['Rank'] = ceo_rank.index + 1

#send to csv for further cleaning
glassdoor = [col for col in ceo_rank if col.startswith('GD')]
HBR = [col for col in ceo_rank if col.startswith('HBR')]
ceo_rank[['Rank'] + glassdoor ].to_csv('ceo_rank_glassdoor.csv')
ceo_rank[['Rank'] + HBR ].to_csv('ceo_rank_hbr.csv')
```

Cleaning The Data

Here are the following steps that we used to clean and merge the different data sources:

1. Clean the Fortune 1000 Data and extract the infor nested in the intial scrapped data
2. Clean the WSJ data and prepare the company names to match company name format in the fortune 1000 an
3. Clean the Glassdoor and Harvard Business Review and prepare company names for additional outer join into
4. Outer-join all four data sources together

Fortune 1000 Data Cleaning and extraction

1. Fortune 1000 Preparation and Extraction for merge

```
import pandas as pd
import numpy as np
import scipy as sc

#Reading in the scraped data from Fortune 1000 list
df = pd.read_csv("fortune.com_companies.csv")

#Splitting the format seperated by "], ["
df["ceos_page_data"].values[0].split("], [")

#We will extract each element of the larger blob and make a dictionary pointin each el
#We will make the diamond from the junk and find stuff.
def make_dict(junk):
    diamond = dict()
    dumpster  = junk.split("], [")

    #Enumerating the index and values to iterate through and extract the string wanted
    for i,trash in enumerate(dumpster):
        rubbish = trash.split("<', '>")

        #If the split string is only length one it means the element points to no valu
        if len(rubbish) == 1:
            if i == 0:
                key = rubbish[0][4:-2]
            elif i == len(dumpster) - 1:
                key = rubbish[0][2:-4]
            else:
                key = rubbish[0][2:-2]
            diamond[key] = None

        #If more than one element in each thing then we point to the value, and the pe
        else:
            #Different index positioning has different formatting just for term and la
            if i == 0:
                key = rubbish[0][4:]
                value = rubbish[-1][:-2]
                diamond[key] = value
            elif i == len(dumpster) - 1:
                key = rubbish[0][2:]
                value = rubbish[-1][:-4]
                diamond[key] = value
            else:
                if len(rubbish) == 3:
                    key = rubbish[0][2:]
                    value = rubbish[1]
                    diamond[key] = value
                    key2 = key + "Growth"
                    value2 = rubbish[-1][:-2]
                    diamond[key2] = value2
                else:
                    key = rubbish[0][2:]
                    value = rubbish[-1][:-2]
                    diamond[key] = value
    #returning the dictionary to be initialized as a new dataframe and then merged wi
    return diamond

#returning the correctly formated dataframe
test_df = pd.DataFrame(df["ceos_page_data"].apply(make_dict).values.tolist())
cleaned_df = df.join(test_df).drop(columns = ["ceos_page_data","ceo_name","","Unnamed
cleaned_df.to_csv("Cleaned_F500data.csv")
```

2. WSJ Preparation and merge with Fortune 1000 data

```
import pandas as pd
import numpy as np
import scipy as sc
import re

#Reading in the data that will be prepped for merging
df2 = pd.read_csv("wsjS&P.csv")
df1 = pd.read_csv("Cleaned_F500data.csv")
df1 = df1.rename(columns = {"companies":"Company"})
test2 = df2.copy()
test1 = df1.copy()

#Cleaning and parsing operation to be performed on individual strings in each entry
clean_amp = lambda text: re.sub(r"&amp;","&",text)
clean_space = lambda text: text[:-1] if text[-1] == " " else text
clean_period = lambda text: re.sub(r"\."," ",text)
clean_white = lambda text: re.sub(r" ","",text)
clean_company = lambda text: re.sub(r"company$","",text)
clean_companies = lambda text: re.sub(r"companies$","",text)
clean_group = lambda text: re.sub(r"group$","",text)
clean_corp = lambda text: re.sub(r"corporation","corp",text)
lowercase = lambda text: str.lower(text)
clean_adobe = lambda text: "adobe" if text == "adobesystems" else text

#Cleaning operation loaded into a list to be iterated and applied to the DataFrame co
clean_funcs = [clean_amp,clean_space,clean_period,clean_white,clean_company,clean_com

#Applying the functions to the data column
for func in clean_funcs:
    test2["Company"] = test2["Company"].apply(func)
    test1["Company"] = test1["Company"].apply(func)

#Merging on prepped data
wsj_F500_merged = test1.merge(test2,on = "Company",how = "outer")
wsj_F500_merged.to_csv("wsj_F500_merged.csv")
```

3. Glassdoor and HBR Cleaning code

↳ 1 cell hidden

4. Final outer join of all data sources

```
import pandas as pd
import numpy as np
import scipy as sc
import re

#first data set is already merged
df1 = pd.read_csv("wsj_F500_merged.csv")

#Final 2 remaining data sets to be merged onto the first
df2 = pd.read_csv('Cleaned_ceo_rank_glassdoor_data.csv')
df3 = pd.read_csv('Cleaned_ceo_rank_hbr.csv')

#Renaming to match columns names for the joining company names
df2 = df2.rename(columns = {"GD_company": "Company","CEO Rank": "GD CEO Rank"})
df2 = df2.iloc[:,1:]
df3 = df3.rename(columns = {"HBR_Company": "Company","Rank": "HBR CEO Rank"})
df3 = df3.iloc[:,1:]
test3 = df3.copy()
test2 = df2.copy()
test1 = df1.copy()

#Cleaning Functionalities to merge on company
clean_amp = lambda text: re.sub(r"&amp;","&",text)
clean_space = lambda text: text[:-1] if text[-1] == " " else text
clean_period = lambda text: re.sub(r"\."," ",text)
clean_white = lambda text: re.sub(r" ","",text)
clean_company = lambda text: re.sub(r"company$","",text)
clean_companies = lambda text: re.sub(r"companies$","",text)
clean_group = lambda text: re.sub(r"group$","",text)
clean_corp = lambda text: re.sub(r"corporation","corp",text)
lowercase = lambda text: str.lower(text)
clean_adobe = lambda text: "adobe" if text == "adobesystems" else text
clean_accented_e = lambda text: re.sub(r"é","e",text)
clean_accented_o = lambda text: re.sub(r"ó","o",text)

#Operations to be performed in iterable
clean_funcs = [clean_amp,
               clean_space,
               clean_period,
               clean_white,
               clean_company,
               clean_companies,
               clean_group,
               lowercase,
               clean_adobe,
               clean_accented_e,
               clean_accented_o]

#Applying each cleaning functional on to the data frames to be merged
for func in clean_funcs:
    test2["Company"] = test2["Company"].apply(func)
    test3["Company"] = test3["Company"].apply(func)

#Final merging of all data sets
final_merge = test1.merge(test2, on = "Company", how = "outer")
final_merge = final_merge.merge(test3, on = "Company", how = "outer")
final_merge.drop(columns = ["Unnamed: 0", "Unnamed: 0_x"], inplace = True)
final_merge.to_csv("Nov_15_CheckpointFinalMerge.csv")
```