Vizualizing Company Reivews Data from Glassdoor

Darshan Tina ¶

In [2]:

```
import pandas as pd
data = pd.read_csv('data/employee_reviews.csv')
#Fixes formatting issue
advice = data['advice-to-mgmt']
advice = advice[advice!='none'].astype('U')
data['advice-to-mgmt'] = advice

advice = data['advice-to-mgmt']
advice = advice[advice!='none'].astype('U')
data['advice-to-mgmt'] = advice
```

In [3]:

```
data.shape
```

Out[3]:

```
(67529, 17)
```

In [4]:

```
data.head()
```

Out[4]:

	Unnamed: 0	company	location	dates	job-title	summary	pros	cons	advice-to-mgmt	overa ratin
0	1	google	none	Dec 11, 2018	Current Employee - Anonymous Employee	Best Company to work for	People are smart and friendly	Bureaucracy is slowing things down	nan	
1	2	google	Mountain View, CA	Jun 21, 2013	Former Employee - Program Manager	Moving at the speed of light, burn out is inev...	1) Food, food, food. 15+ cafes on main campus ...	1) Work/life balance. What balance? All those ...	1) Don't dismiss emotional intelligence and ad...	
						Great	* If			

In [5]:

```
review_data = data.drop('Unnamed: 0',axis=1)
```

1) Ratings for Each Company

Question: Who are the best companies to work for?

A. Overall Ratings

In [6]:

```
company_unique = set(review_data['company'])
company_by_rating = review_data.groupby('company')['overall-ratings'].mean()
company_by_rating.sort_values(inplace=True, ascending=False)
company_by_rating
```

Out[6]:

```
company
facebook     4.511950
google       4.339430
apple        3.958224
microsoft    3.816564
amazon       3.587363
netflix      3.411111
Name: overall-ratings, dtype: float64
```

In [7]:

```
import matplotlib.pyplot as plt
import numpy as np
plt.figure(figsize=(8,6))
plt.bar(company_by_rating.index,company_by_rating.values,color='brown')
plt.xlabel('Company')
plt.ylabel('Ratings')
plt.title('Company Ratings')
plt.show()
```

```
<Figure size 800x600 with 1 Axes>
```

B. Various Metric Ratings

In [8]:

```
Career_opp = review_data
data.loc[:,['company','overall-ratings', 'work-balance-stars',
       'culture-values-stars', 'carrer-opportunities-stars',
       'comp-benefit-stars', 'senior-mangemnet-stars','job-title']]

Career_opp['carrer-opportunities-stars'] = pd.to_numeric(Career_opp['carrer-opportunities-s
Career_opp['overall-ratings'] = pd.to_numeric(Career_opp['overall-ratings'],errors='coerce'
Career_opp['work-balance-stars'] = pd.to_numeric(Career_opp['work-balance-stars'],errors='c
Career_opp['culture-values-stars'] = pd.to_numeric(Career_opp['culture-values-stars'],error
Career_opp['comp-benefit-stars'] = pd.to_numeric(Career_opp['comp-benefit-stars'],errors='c
Career_opp['senior-mangemnet-stars'] = pd.to_numeric(Career_opp['senior-mangemnet-stars'],e

Career_opp.to_csv('career_data.csv')

Career_opp
```

Out[8]:

	company	location	dates	job-title	summary	pros	cons	advice-to mgm
0	google	none	Dec 11, 2018	Current Employee - Anonymous Employee	Best Company to work for	People are smart and friendly	Bureaucracy is slowing things down	nar
1	google	Mountain View, CA	Jun 21, 2013	Former Employee - Program Manager	Moving at the speed of light, burn out is inev...	1) Food, food, food. 15+ cafes on main campus ...	1) Work/life balance. What balance? All those ...	1) Don' dismiss emotiona intelligence and ad..
2	google	New York, NY	May 10, 2014	Current Employee - Software Engineer III	Great balance between big-company security and...	* If you're a software engineer, you're among ...	* It *is* becoming larger, and with it comes g...	Keep the focus on the user Everything else wi..

2) Ratings by Location

Question: Where are the best places to work?

In [9]:

```
def state(x):
    a,b,*c = x.strip().split(',')
    return 'USA:'+str(b).strip()
```

In [10]:

```
def country(x):
    *a,b = x.strip().split('(')
    return b[:-1].upper()
```

In [11]:

```
def country_or_state(location):
    if '(' in location:
        return country(location)
    elif ',' in location:
        return state(location)
    else:
        return location
```

In [12]:

```
#Remove reviews without a location
clean_location_df = review_data[review_data['location'] != 'none']

#Create new column (cleaned_location) with cleaned country/state values
clean_location_df=clean_location_df.assign(cleaned_location=clean_location_df['location'].a
clean_location_df
```

Out[12]:

	company	location	dates	job-title	summary	pros	cons	advice-to-mg
1	google	Mountain View, CA	Jun 21, 2013	Former Employee - Program Manager	Moving at the speed of light, burn out is inev...	1) Food, food, food. 15+ cafes on main campus ...	1) Work/life balance. What balance? All those ...	1) Don't dism emotio intelligence a a
2	google	New York, NY	May 10, 2014	Current Employee - Software Engineer III	Great balance between big-company security and...	* If you're a software engineer, you're among ...	* It *is* becoming larger, and with it comes g...	Keep the foc on the us Everything e w
3	google	Mountain View, CA	Feb 8, 2015	Current Employee - Anonymous Employee	The best place I've worked and also the most d...	You can't find a more well-regarded company th...	I live in SF so the commute can take between 1...	Keep on N micromanagin that is a hu be

In [13]:

```
def state_and_country_df(clean_loc_df, loc_as_index=False):
    #Create DataFrame for states
    if loc_as_index:
        values = clean_loc_df.index
    else:
        values = clean_loc_df['cleaned_location']
    state_mask = values.str[:3]=='USA'
    state_df = clean_loc_df[state_mask]

    #The rest of the dataframe is countries
    country_df = clean_loc_df.loc[~state_mask].copy()

    if loc_as_index:
        #Put USA average rating and count in countries dataframe
        usa_avg = state_df['overall-ratings'].mean()
        usa_cnt = len(state_df)
        country_df.loc['USA'] = [usa_cnt, usa_avg]

    return (state_df, country_df)
```

A. State Ratings

In [14]:

```
state_df, country_df = state_and_country_df(clean_location_df)
#Group up locations in state dataframe
state_location_groups = state_df.groupby('cleaned_location')

#Gather only locations with more than 10 reviews
review_counts = state_location_groups.count()['dates'].sort_values(ascending=False)
mask = (review_counts >= 10).sort_index()
highly_reviewed_states = state_location_groups.mean()[mask]
highly_reviewed_states.sort_values('overall-ratings', inplace=True, ascending=False)
highly_reviewed_states
```

Out[14]:

cleaned_location	overall-ratings	work-balance-stars	culture-values-stars	carrer-opportunities-stars	comp-benefit-stars	senior-mangemnet-stars	helpful-count
USA:MS	4.727273	4.181818	4.500000	3.909091	4.136364	3.909091	0.272727
USA:AR	4.388889	3.777778	4.388889	4.000000	4.333333	3.833333	0.500000
USA:HI	4.214286	3.944444	4.083333	3.629630	4.185185	3.611111	0.428571
USA:NH	4.189189	3.375000	4.310345	3.680556	4.222222	3.930556	0.459459
USA:IA	4.181818	3.000000	3.555556	3.800000	4.750000	3.950000	1.636364
USA:AL	4.172414	3.785714	4.407407	4.053571	4.321429	3.892857	0.344828
USA:NY	4.058647	3.651562	4.123684	3.662109	4.188134	3.519639	1.882707
USA:OK	4.000000	3.625000	4.282051	3.412500	3.687500	3.600000	0.650000
USA:DC	3.993197	3.424138	4.067669	3.837931	4.151724	3.447552	0.829932

In [15]:

```
#Get data for state tableau graph
states_ratings = highly_reviewed_states[['overall-ratings']]
states_ratings.index.set_names(['state'], inplace=True)

states_ratings.to_csv('StateTest_data.csv')
```

It is interesting that Mississippi and Arkansas are the top two states. What companies contribute to the high ratings?

In [16]:

```
for state in highly_reviewed_states.index[:6]:
    print(state)
    print(clean_location_df[clean_location_df['cleaned_location']==state]['company'].value_
```

```
USA:MS
apple       8
facebook    1
amazon      1
google      1
Name: company, dtype: int64
USA:AR
apple        9
microsoft    3
amazon       3
google       2
facebook     1
Name: company, dtype: int64
USA:HI
apple        26
microsoft     1
google        1
Name: company, dtype: int64
USA:NH
```

Apple seems to dominate the US top locations.

B. Country Ratings

In [17]:

```
#Create dataframe for count and average overall-rating for a country
country_groups = country_df.groupby('cleaned_location')
country_cnt = country_groups.count()['dates']
country_avg = country_groups.mean()['overall-ratings']

country_cnt_avg = pd.concat([country_cnt, country_avg], axis=1)
country_cnt_avg.rename(index=str, inplace=True, columns={"dates": "count"})

#Put USA average rating and count in countries dataframe
usa_avg = state_df['overall-ratings'].mean()
usa_cnt = len(state_df)
country_cnt_avg.loc['USA'] = [usa_cnt, usa_avg]

#Only keep countries with more than 20 ratings
country_cnt_avg = country_cnt_avg[country_cnt_avg['count']>20]
country_cnt_avg
```

Out[17]:

cleaned_location	count	overall-ratings
ARGENTINA	27.0	4.148148
AUSTRALIA	290.0	3.851724
BRAZIL	160.0	4.281250
CANADA	710.0	3.791549
CHINA	319.0	4.115987
COSTA RICA	149.0	4.328859
CZECH REPUBLIC	96.0	3.156250
DENMARK	28.0	4.000000
EGYPT	50.0	4.060000
FINLAND	42.0	3.523810

In [18]:

```
country_cnt_avg.sort_values('overall-ratings', inplace=True, ascending=False)
country_cnt_avg['rank'] = ['#'+str(i) for i in range(1, len(country_cnt_avg)+1)]
top_5_countries = country_cnt_avg[:5]
top_5_countries.to_csv('top_5_countries.csv')
```

In [19]:

```
top_5_countries
```

Out[19]:

cleaned_location	count	overall-ratings	rank
COSTA RICA	149.0	4.328859	#1
TAIWAN	29.0	4.310345	#2
ISRAEL	103.0	4.281553	#3
BRAZIL	160.0	4.281250	#4
SWEDEN	26.0	4.230769	#5

Again, it is interesting that these are the top countries. What companies contribute to each?

In [20]:

```
for country in country_cnt_avg.index[:5]:
    print(country)
    print(clean_location_df[clean_location_df['cleaned_location']==country]['company'].valu
```

```
COSTA RICA
amazon       143
microsoft      5
google         1
Name: company, dtype: int64
TAIWAN
google       13
microsoft    12
apple         4
Name: company, dtype: int64
ISRAEL
microsoft    58
amazon       15
google       14
facebook     10
apple         6
Name: company, dtype: int64
BRAZIL
microsoft    85
```

Amazon does very well in Costa Rica, but Microsoft does better in Israel and Brazil.

C. Company Ratings by Location

In [21]:

```
def count_and_avg_for_company(clean_df, company):
    '''Calculates the number of ratings and average rating for each location of a company.

    Groups US states and countries together
    Only count the locations with more than 10 ratings.
    Returns a dataframe with count and average of ratings per location

    Keyword arguments:
    clean_df -- df with a "cleaned_location" column
    company -- name of company as a string
    '''

    cmp = clean_df[clean_df['company']==company]
    location_groups = cmp.groupby('cleaned_location')

    #Find number of ratings and the average rating for all locations
    all_loc_cnts_cmp = location_groups.count()['dates']
    all_loc_avg_cmp = location_groups.mean()['overall-ratings']
    all_loc_cnts_cmp.name = 'count'

    #Return a dataframe with the count and mean for all locations
    loc_df_cmp = pd.concat([all_loc_cnts_cmp, all_loc_avg_cmp], axis=1)
    loc_df_cmp.sort_values('overall-ratings', ascending = False)
    return loc_df_cmp
```

In [22]:

```
def compute_z_scores(series):
    return (series-series.mean())/series.std()
```

In [23]:

```
def get_colors(start, end, num_divisions):
    '''Get color map between two colors

    Keyword arguments:
    start/end -- starting/ending color in format #123bef
    num_divisions -- number of colors to return in between start/end
    '''

    r_start = int(start[1:3], 16)
    g_start = int(start[3:5], 16)
    b_start = int(start[5:7], 16)

    r_width = int(end[1:3], 16) - r_start
    g_width = int(end[3:5], 16) - g_start
    b_width = int(end[5:7], 16) - b_start

    r_increment = r_width/(num_divisions-1)
    g_increment = g_width/(num_divisions-1)
    b_increment = b_width/(num_divisions-1)

    def hex_color(i):
        r = str(hex(r_start+int(r_increment*i)))[2:]
        g = str(hex(g_start+int(g_increment*i)))[2:]
        b = str(hex(b_start+int(b_increment*i)))[2:]
        return '#'+r+g+b

    return [hex_color(i) for i in range(num_divisions)]
```

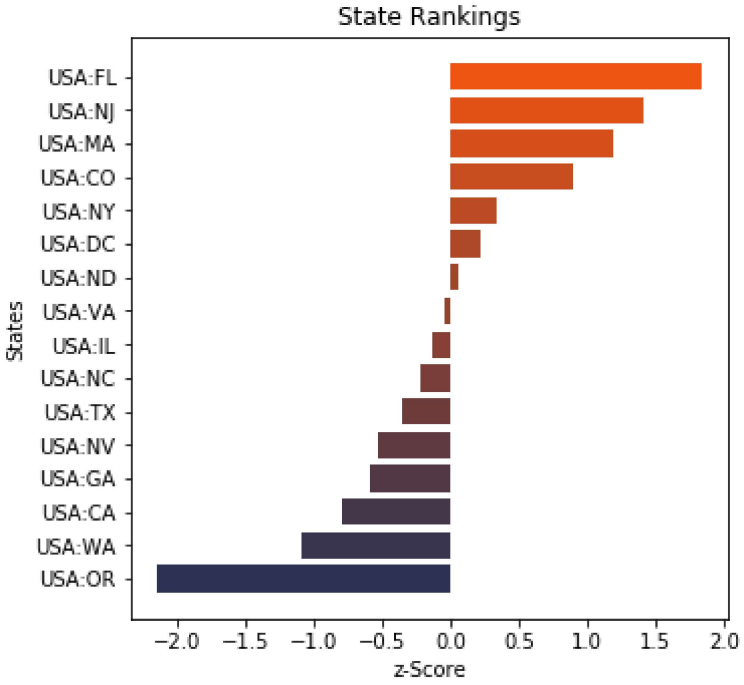
In [24]:

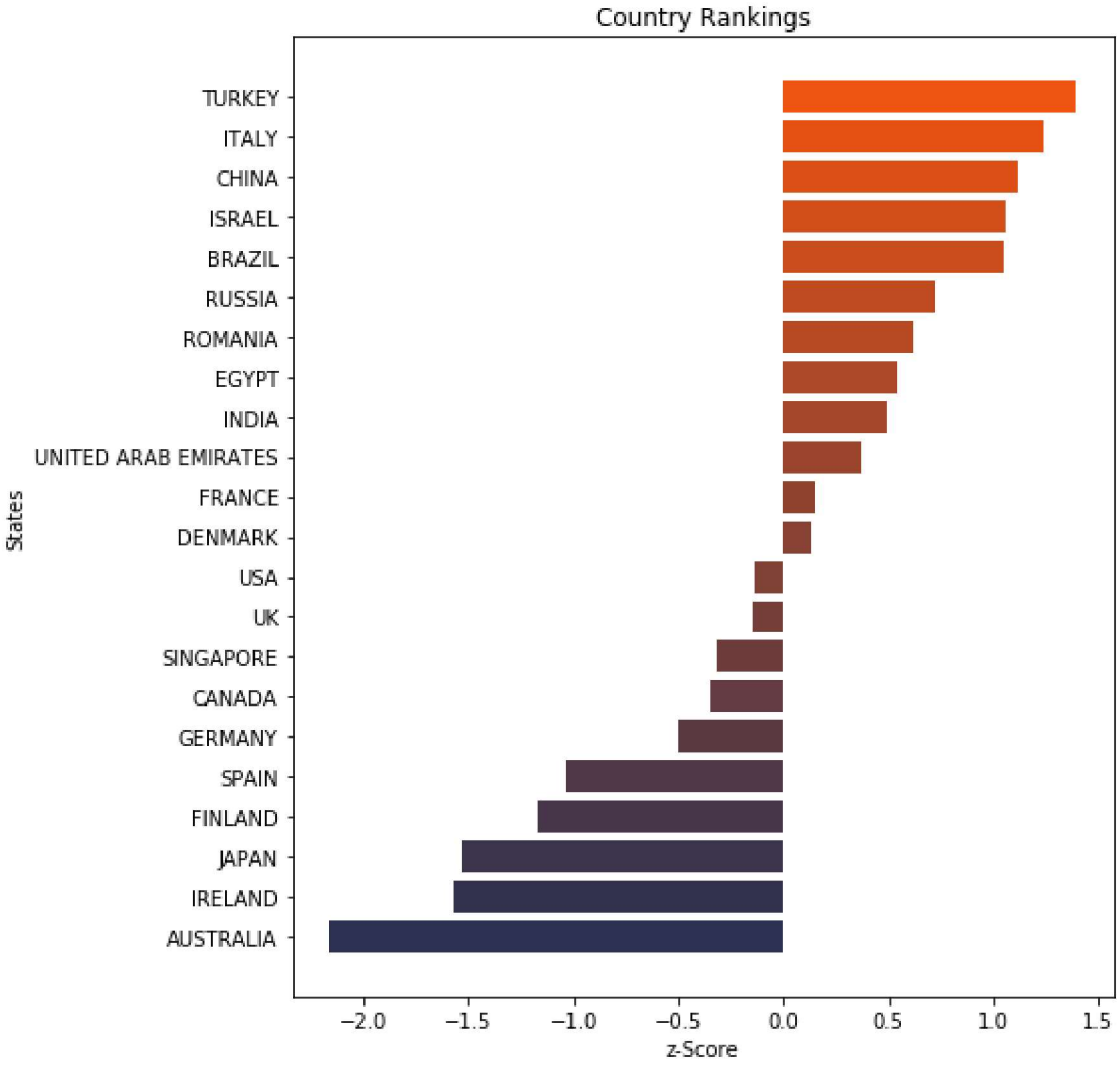
```
from operator import itemgetter
def state_and_country_plots(state_df, country_df):
    '''Displays two plots by average rating: a US plot and an all countries plot

    Keyword arguments:
    state_df -- DataFrame of state count, means used in the plot
    country_df -- DataFrame of country count, means used in the plot
    '''

    states = state_df['overall-ratings'].sort_values(ascending=True)
    countries = country_df['overall-ratings'].sort_values(ascending=True)

    state_z_scores = compute_z_scores(states)
    country_z_scores = compute_z_scores(countries)
    color1 = '#2b3252'
    color2 = '#ef5410'
#     color1 = '#141A46'
#     color2 = '#EC8A5E'

    if len(state_z_scores) > 1:
        plt.figure(figsize=(1/3*len(state_z_scores), 1/3*len(state_z_scores)))
        plt.title('State Rankings')
        plt.ylabel('States')
        plt.xlabel('z-Score')
        plt.barh(state_z_scores.index, state_z_scores, color=get_colors(color1, color2, ler
        plt.show()

    if len(country_z_scores) > 1:
        plt.figure(figsize=(1/3*len(country_z_scores), 1/2.5*len(country_z_scores)))
        plt.title('Country Rankings')
        plt.ylabel('States')
        plt.xlabel('z-Score')
        plt.barh(country_z_scores.index, country_z_scores, color=get_colors(color1, color2,

    plt.show()
```

In [25]:

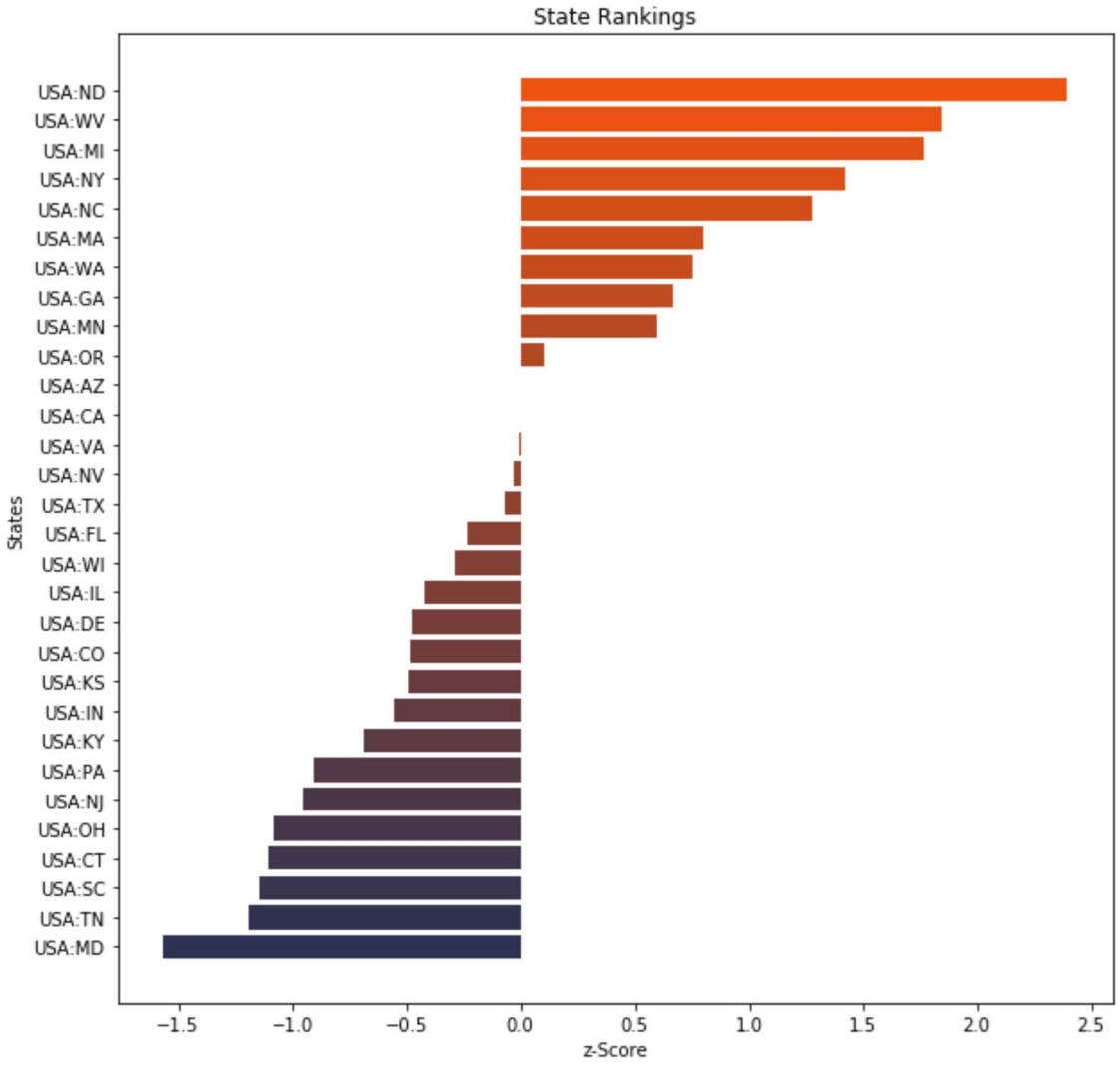
```
for company in set(review_data['company']):
    print(company)

    cmp_df = count_and_avg_for_company(clean_location_df, company)

    #Only include locations with more than 20 ratings
    loc_ten_ratings = cmp_df[cmp_df['count'] > 20]
    state_df, country_df = state_and_country_df(loc_ten_ratings, True)
    state_and_country_plots(state_df, country_df)
```

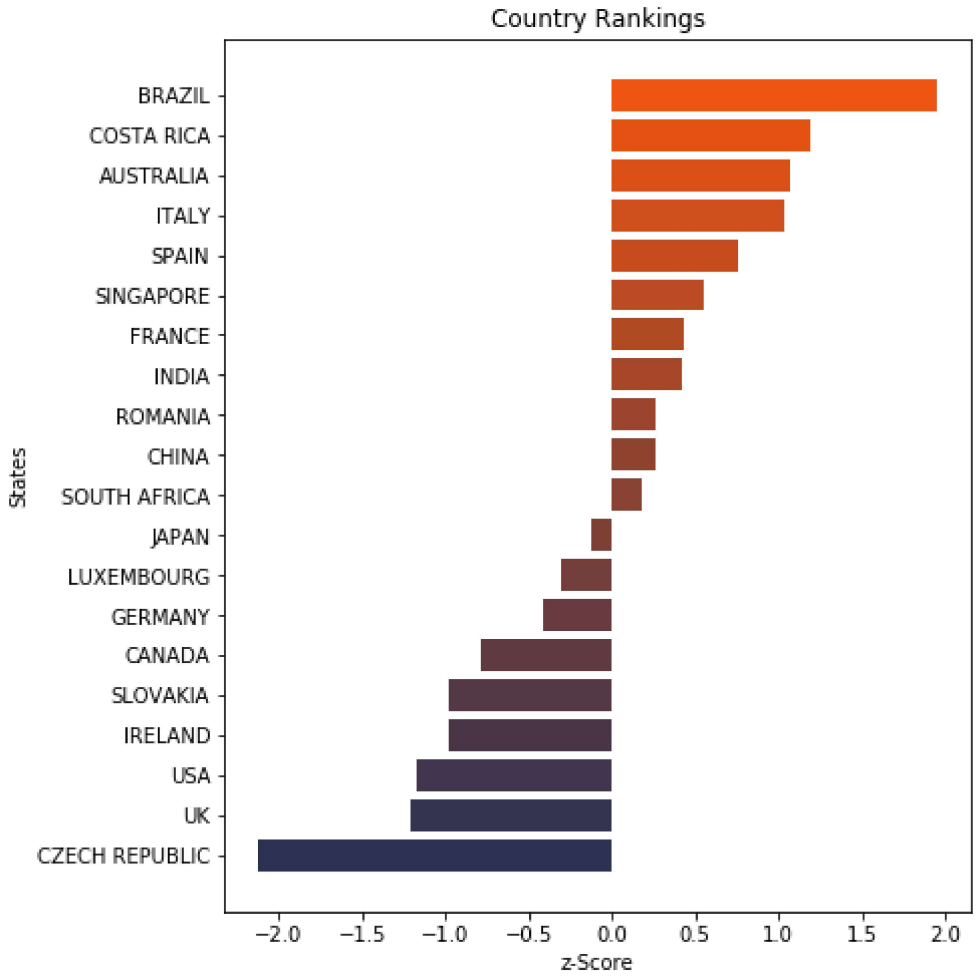
```
microsoft
```



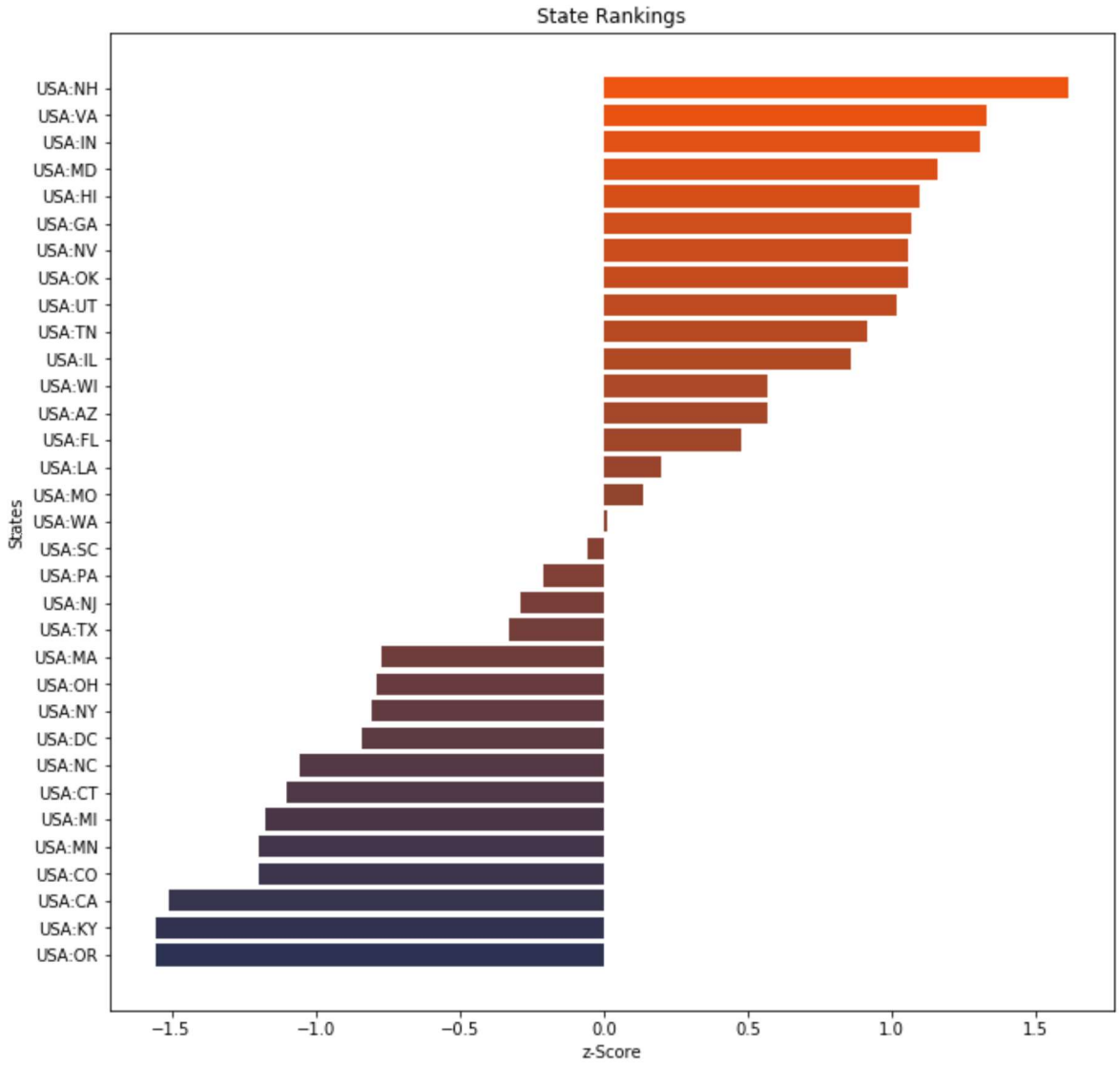

Country Rankings
TURKEY
ITALY
CHINA
ISRAEL
BRAZIL
RUSSIA
ROMANIA
EGYPT
INDIA
UNITED ARAB EMIRATES
FRANCE
DENMARK
USA
UK
SINGAPORE
CANADA
GERMANY
SPAIN
FINLAND
JAPAN
IRELAND
AUSTRALIA
States
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
z-Score

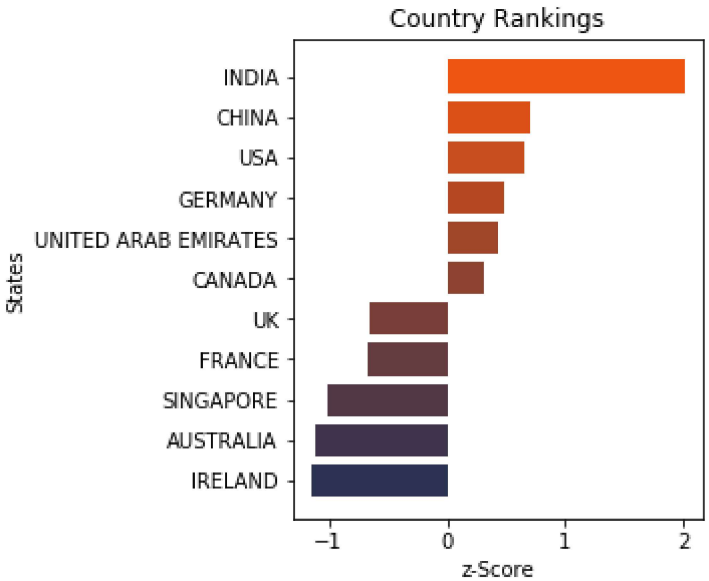
amazon


State Rankings
USA:ND
USA:WV
USA:MI
USA:NY
USA:NC
USA:MA
USA:WA
USA:GA
USA:MN
USA:OR
USA:AZ
USA:CA
USA:VA
USA:NV
USA:TX
USA:FL
USA:WI
USA:IL
USA:DE
USA:CO
USA:KS
USA:IN
USA:KY
USA:PA
USA:NJ
USA:OH
USA:CT
USA:SC
USA:TN
USA:MD
States
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
z-Score

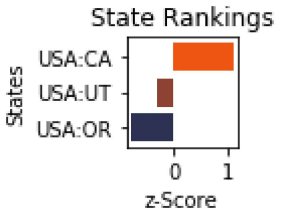

Country Rankings
BRAZIL
COSTA RICA
AUSTRALIA
ITALY
SPAIN
SINGAPORE
FRANCE
INDIA
ROMANIA
CHINA
SOUTH AFRICA
JAPAN
LUXEMBOURG
GERMANY
CANADA
SLOVAKIA
IRELAND
USA
UK
CZECH REPUBLIC
States
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
z-Score

```
apple
```

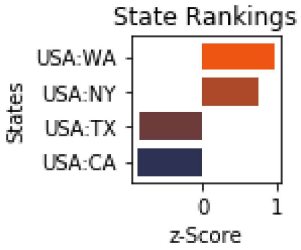

State Rankings
USA:NH
USA:VA
USA:IN
USA:MD
USA:HI
USA:GA
USA:NV
USA:OK
USA:UT
USA:TN
USA:IL
USA:WI
USA:AZ
USA:FL
USA:LA
USA:MO
USA:WA
USA:SC
USA:PA
USA:NJ
USA:TX
USA:MA
USA:OH
USA:NY
USA:DC
USA:NC
USA:CT
USA:MI
USA:MN
USA:CO
USA:CA
USA:KY
USA:OR
States
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
z-Score

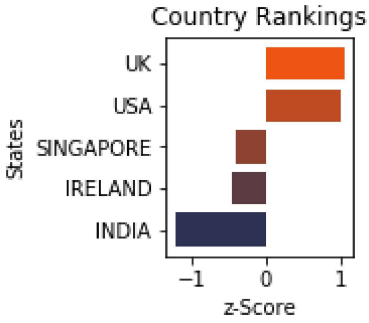

Country Rankings
INDIA
CHINA
USA
GERMANY
UNITED ARAB EMIRATES
CANADA
UK
FRANCE
SINGAPORE
AUSTRALIA
IRELAND
States
−1
0
1
2
z-Score

netflix

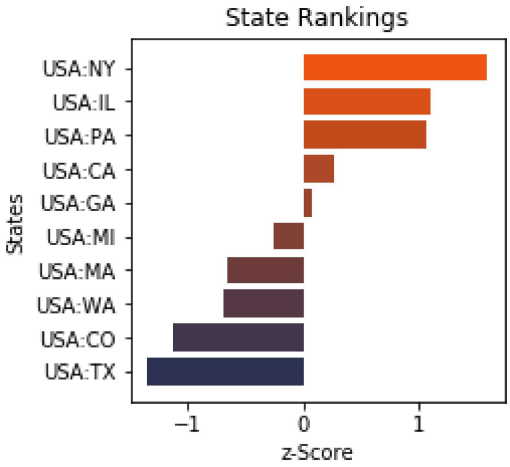


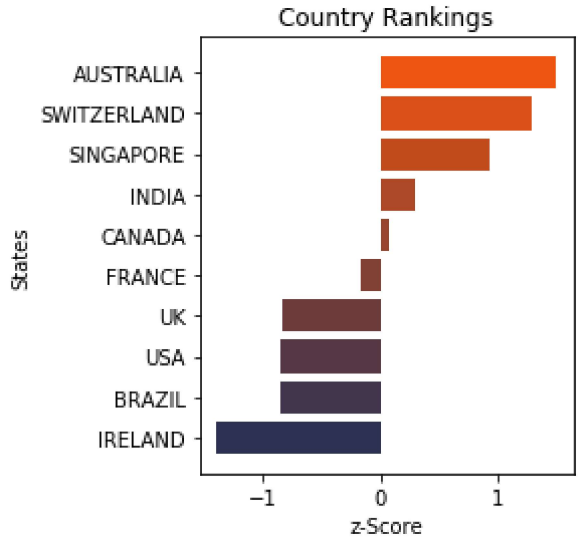
facebook





google





Case Study 1: Amazon in North Dakota

Why is USA:ND ranked so highly for Amazon?

In [26]:

```
df = clean_location_df[clean_location_df['cleaned_location']=='USA:ND']
df2 = df[df['company']=='amazon'].copy()
print(len(df2))
```

```
27
```

There seems to be plenty of reviews so upon further research, Amazon sent many of their employees to work from home in 2018. This may have resulted in some higher reviews for 2018.

In [27]:

```
from datetime import datetime
df2['datetimes'] = df2.apply(lambda s: datetime.strptime(s['dates'].strip(), '%b %d, %Y'),
# display(df2.sort_values('datetimes', ascending=False)[['overall-ratings','location','date
all_other_ratings = df2[df2['datetimes']<='2018-01-01']['overall-ratings'].mean()
rating_2018 = df2[df2['datetimes']>'2018-01-01']['overall-ratings'].mean()
print('Prior to 2018: '+str(all_other_ratings))
print('During 2018: '+str(rating_2018))
```

```
Prior to 2018: 3.7142857142857144
During 2018: 4.666666666666667
```

In fact, the reviews were much better during 2018, and may be part of the reason the ratings are so high for this state.

In [28]:

```
df2['overall-ratings'].mean()
```

Out[28]:

```
3.925925925925926
```

Also, as we saw before, Amazon had pretty low ratings in the US compared to other countries. The overall rating of Amazon in ND was only 3.93 so the bar was fairly low for state ratings.

Case Study 2: Facebook Highest Ratings

Why does Facebook have the highest ratings? Facebook has a higher rating than Google, maybe there are an inadeaquate number of reviews.

In [29]:

```
face = clean_location_df[clean_location_df['company']=='facebook']
f = len(face)
g = len(clean_location_df[clean_location_df['company']=='google'])
print('Facebook: '+str(f))
print('Google: '+str(g))
```

```
Facebook: 1590
Google: 4158
```

It seems like there are plenty reviews of both companies, but maybe facebook's reviews are limited in some way.

In [30]:

```
loc_counts = face['cleaned_location'].value_counts()
loc_counts
```

Out[30]:

```
USA:CA                984
IRELAND               113
USA:WA                 91
UK                     91
USA:NY                 84
USA:TX                 49
INDIA                  32
SINGAPORE              22
USA:IL                 12
BRAZIL                 12
ISRAEL                 10
USA:MA                  9
USA:DC                  6
CANADA                  6
ARGENTINA               6
BANGLADESH              4
PHILIPPINES             4
AUSTRALIA               3
```

In [31]:

```
loc_counts[0]/sum(loc_counts)
```

Out[31]:

```
0.6188679245283019
```

In [32]:

```
ca = face[face['cleaned_location']=='USA:CA'].mean()['overall-ratings']
ireland = face[face['cleaned_location']=='IRELAND'].mean()['overall-ratings']
india = face[face['cleaned_location']=='INDIA'].mean()['overall-ratings']
print('USA:CA: '+str(ca))
print('IRELAND: '+str(ireland))
print('INDIA: '+str(india))
```

```
USA:CA: 4.5060975609756095
IRELAND: 4.353982300884955
INDIA: 4.21875
```

It seems like their offices in California are doing great, but Ireland and India have slightly lower ratings. Facebook is the overall best company, but it seems as though they are limited to their USA:CA office in some way because 61% of their reviews come from this location. It would be interesting to see this number change as facebook grows globally, and as they continue to struggle with the "content moderator" position.

Case Study 3: Microsoft in Washington

Microsoft dominates in Isreal and Brazil (two of the highest rated countries to work in), but we see here that it struggles back in the states. It is interesting to see that the second worst reviews are coming from its own home state (USA:WA - Washington).

In [33]:

```
mask = (clean_location_df['company']=='microsoft') & (clean_location_df['cleaned_location']
washington_reviews = clean_location_df[mask]
redmond_reviews = washington_reviews[washington_reviews['location']=='Redmond, WA']
other_reviews = washington_reviews[washington_reviews['location']!='Redmond, WA']
len(redmond_reviews)/len(washington_reviews)
```

Out[33]:

```
0.8066234701223902
```

In [34]:

```
redmond_reviews['overall-ratings'].mean()
```

Out[34]:

```
3.649589432345591
```

In [35]:

```
other_reviews['overall-ratings'].mean()
```

Out[35]:

```
3.768428890543559
```

In fact, 80% of the reviews in Washington are from its headquarters in Redmond, and those reviews are a little worse on average than the rest of Washington. Regardless, Microsoft struggles in its home state.

3) Pros, Cons, Advice Analysis

Goal: Analyze these three sections using a bag of words model with the help of CountVectorizer from sklearn.

In [36]:

```
from sklearn.feature_extraction.text import CountVectorizer
from operator import itemgetter
def print_freq_table(entries, prnt=True, num_top_words=40,ngram=(1,2)):
    '''Prints out the frequency table for all words in the given entries.

    Uses CountVectorizer from sklearn.
    Eliminates words included in more than 90% of entries.
    Eliminates words included in less than 10 entries.
    Removes stop_words from the CountVectorizer dictionary.

    Keyword arguments:
    entries -- the entries to fit CountVectorizer on
    num_top_words -- number of top words to display in frequency table
    '''

    vectorizer = CountVectorizer(min_df=10, max_df=.9, stop_words="english", ngram_range=ng
    bag_of_words = vectorizer.fit_transform(entries)
    sum_of_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_of_words[0, idx]) for word, idx in vectorizer.vocabulary_.item
    sorted_list = sorted(words_freq, reverse = True, key=itemgetter(1))

    if prnt:
        print('Rank'.ljust(10), 'Lemma'.ljust(20), 'Raw Count')
        for i in range(num_top_words):
            if(i >= len(sorted_list)): break
            print(str(i+1).ljust(10), sorted_list[i][0].ljust(20), sorted_list[i][1])

    return sorted_list
```

In [37]:

```
def word_chart_for_column(col_name, prnt=True, num_top_words=40, ngram=(2,2)):
    '''Return Bag of words model for each company for given column

    Keyword arguments:
    col_name -- "pros" or "cons"
    prnt -- True or false for printing out freq tables
    num_top_words -- top words to display
    ngram -- ngram range for count vectorizer
    '''
    companies = set(data['company'])
    words_company = {}
    for company in companies:
        if prnt:
            print(f'\nCompany {col_name}: '+company.title())
        entries = data[data['company']==company][col_name]
        test_data = print_freq_table(entries, prnt, num_top_words, ngram)
        words_company[company.title()]=test_data
    return words_company
```

In [38]:

```
def top_words(column, num_words, companies):
    '''Returns most frequent words in pros, cons, or advice-to-mgmt col for given companies

    Keyword arguments:
    column -- pros, cons, or advice-to-mgmt
    num_words -- number of words to include in list
    companies -- a list of companies to include in ouput list
    '''

    pros = word_chart_for_column(column, False, ngram=(1,2)).items()
    company_pros = pd.DataFrame([(company, word) for company, words in pros
                                                        for word, count in words])
    word_col = column+' word'
    count_col = column+' count'
    company_pros.columns = ['company', 'word']

    #Get words that only occur for that company
    unique_words = company_pros[company_pros.groupby('word').word.transform(len) == 1].copy
    unique_words.rename(columns={'word': word_col}, inplace=True)

    #Join lists together for each company
    dfs = []
    for company in companies:
        df = unique_words[unique_words['company']==company][:num_words]
        company_num_words = len(df)
        df['rank'] = list(range(1, company_num_words+1))
        dfs.append(df)

    return pd.concat(dfs)
```

A. Top Three Unique Words in Pros, Cons, Advice

In [39]:

```
cols = ['pros', 'cons', 'advice-to-mgmt']
companies = ['Google', 'Amazon', 'Facebook']
num_words = 3
all_top_words = [top_words(col, num_words, companies) for col in cols]
```

In [40]:

```
print('Pros')
display(all_top_words[0])
print('Cons')
display(all_top_words[1])
print('Advice')
display(all_top_words[2])
```

Pros

	company	pros word	rank
12808	Google	working google	1
12840	Google	work google	2
12864	Google	google great	3
3838	Amazon	principles	1
3864	Amazon	leadership principles	2
3938	Amazon	working amazon	3
12036	Facebook	fb	1
12158	Facebook	billion people	2
12228	Facebook	working facebook	3

Cons

In [41]:

```
from functools import reduce

df_final = reduce(lambda left,right: pd.merge(left,right,how='outer',on=['company', 'rank']
```

In [42]:

```
df_final[['rank', 'company', 'pros word', 'cons word', 'advice-to-mgmt word']].to_csv('word
```

B. Wordcloud Visualizations

In [43]:

```
!pip install wordcloud
```

```
Requirement already satisfied: wordcloud in /Library/Frameworks/Python.frame
work/Versions/3.7/lib/python3.7/site-packages (1.5.0)
Requirement already satisfied: pillow in /Library/Frameworks/Python.framewor
k/Versions/3.7/lib/python3.7/site-packages (from wordcloud) (5.4.1)
Requirement already satisfied: numpy>=1.6.1 in /Library/Frameworks/Python.fr
amework/Versions/3.7/lib/python3.7/site-packages (from wordcloud) (1.16.1)
You are using pip version 19.0.2, however version 19.0.3 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

In [44]:

```
from wordcloud import WordCloud
```

In [45]:

```
def word_plot(company, col_data):
    '''Word plot for company with word_chart_for_col

    Keyword arguments:
    company -- as given
    col_data -- return value of word_chart_for_col
    '''
    cmp_data = [k.replace(' ','') for k,v in col_data[company]]
    cmp_corp = ' '.join(cmp_data)

    wc = WordCloud(width = 800, height = 800,
                   background_color ='white',
                   min_font_size = 10).generate(cmp_corp)

    plt.figure(figsize = (8, 8), facecolor = None)
    plt.imshow(wc)
    plt.axis("off")
    plt.tight_layout(pad = 0)
    plt.show()
```

In [46]:

```
pros = word_chart_for_column('pros', False, ngram=(2,2))
cons = word_chart_for_column('cons', False, ngram=(2,2))
advice = word_chart_for_column('advice-to-mgmt', False, ngram=(2,2))
for company in set(data['company']):
    cmp = company.title()
    print(cmp)
    print('Pros')
    word_plot(cmp, pros)

    print('Cons')
    word_plot(cmp, cons)

    print('Advice to management')
    word_plot(cmp, advice)
```

Microsoft
Pros

Cons

Advice to management


individualcontributor
productgroup
stackranking
changereview
goodmanager
peoplemanagement
makesure
goodjob
donknow
goodwork
managementlayer
greatjob
seniormanagement
reviewprocess
shortterm
hirepeople
smartpeople
ridstack
donjust
performancereview
peoplemanager
goodpeople
greatwork
peoplework
doingdoing
placework
payattention
lowerlevel
makedecision
stackrank
middlemanagement
uppermanagement
managementneed
reviewmodel
listenemployees
stockprice
windowslive
focusemployee
worklife
longterm
levelmanagers
spendtime
middlemanagers
focuscustomer
opensource
stoptrying
lifebalance
decisionmaking

Amazon
Pros

Cons

Advice to management


minutebreak
15minute
lifebalance
donjust
leadershipprinciple
workhard
payattention
peoplework
placework
bettertraining
donknow
carepeople
middlemanagement
peoplemanager
hardwork
hardworking
makesure
workemployee
employeeswork
highturnover
customercentric
employeeshuman
longway
likepeople
treatemployee
goodmanager
customerservice
seniormanagement
hardworker
fulfillmentcenter
reallyneed
peoplelike
companyemployee
goodemployee
doingthing
stoptreating
likehuman
areamanager
employeescustomer
advicemanagement
doinggreat
employeeslike
goodpeople
justnumber
worklife
uppermanagement
workenvironment
listenemployee
bettercommunication
managementteam
performancereview
10hour
teammembers
managementneed
treatpeople
longterm
spendtime
teamleader
peopleskills
doingdoing
goodjob
focusemployee
greatjob
hirepeople
newhire
greatwork
stophiring
employeesbetter
careemployee
goodwork
newemployees
turnoverrate
companyneed
hourshift
betterwork
humanbeings
makeemployee
peopledon

Apple
Pros


retailjob
greatproduct
employeediscount
reallycare
applestore
retailstore
greatteam
flexibleschedule
worklot
greatemployee
goodbenefit
companylot
greatbenefit
timeemployee
bestreason
trainingprogram
environmentlot
productsdiscount
peoplelot
meetlot
appleproduct
greatcustomer
gooddiscount
newproduct
collegestudent
benefitslot
benefitsemployee
greatdiscount
productdiscount
goodthing
careerexperience
appleemployee
excellentbenefits
flexiblehours
trulycare
healthinsurance
customerservice
workingenvironment
environmentgreat
smartpeople
decentdiscount
goodproduct
greatpay
stockpurchase
healthbenefits
benefitstime
benefitsgreat
greatcompany
workenvironment
peoplegood
placework

Cons

Advice to management

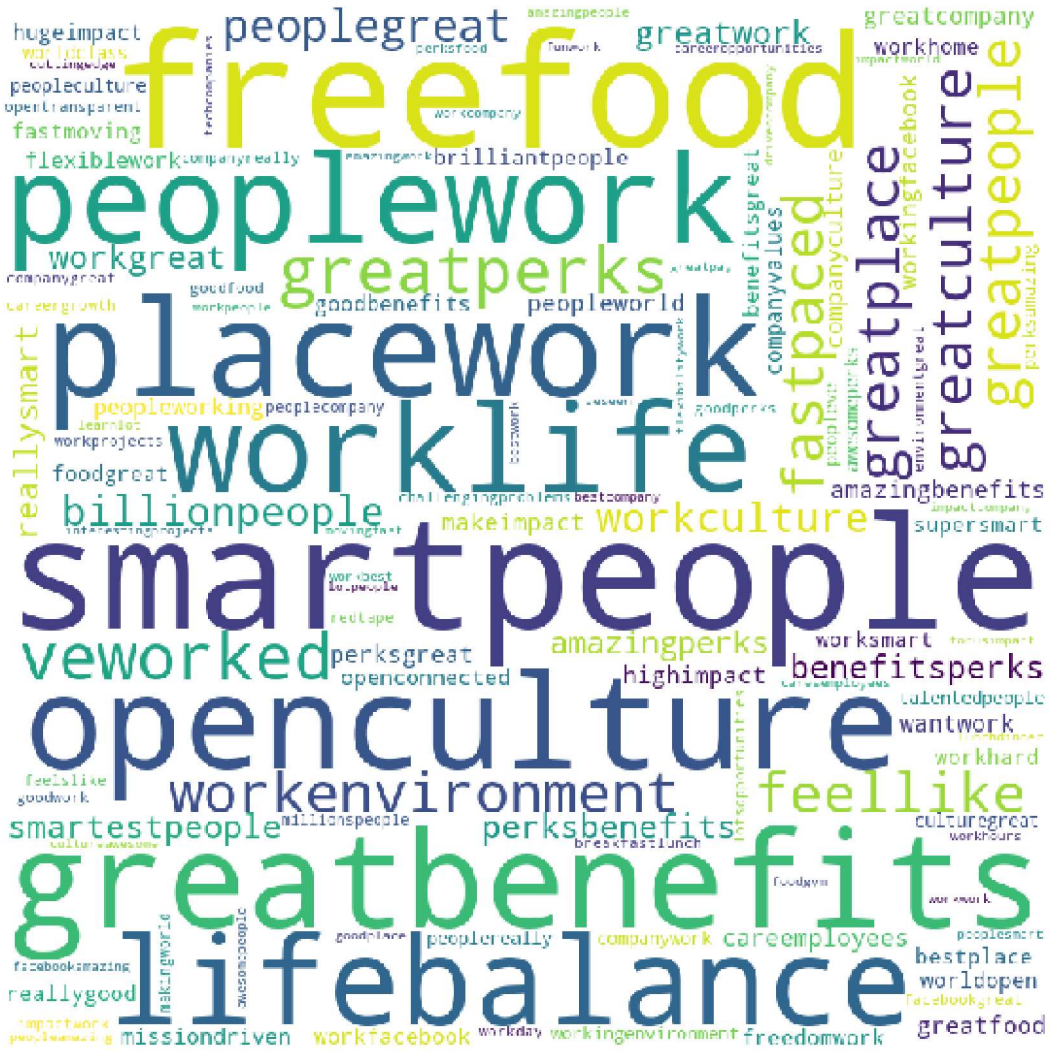
Netflix
Pros

Cons


freedomresponsibility
highturnover
veworked
jobsecurity
placework
highperformance
performanceculture
lifebalance
longterm
feellike
worklife
culturefear
customerservice
culturedeck
uppermanagement
longhours
peoplelet
vacationpolicy

Advice to management

Facebook
Pros


freefood
peoplework
placework
worklife
smartpeople
openculture
greatbenefits
lifebalance
peoplegreat
greatwork
greatcompany
workhome
greatplace
greatpeople
greatculture
fastpaced
workgreat
greatperks
goodbenefits
peopleworld
billionpeople
workculture
makeimpact
amazingbenefits
supersmart
veworked
perksgreat
openconnected
amazingperks
worksmart
benefitsperks
highimpact
talentedpeople
wantwork
workhard
workenvironment
feellike
smartestpeople
perksbenefits
culturegreat
careemployees
bestplace
worldopen
greatfood
missiondriven
worktacebook
freedomwork
workingenvironment
companywork
peoplereally
brilliantpeople
flexiblework
fastmoving
peopleculture
hugeimpact
reallysmart
foodgreat
peopleworking
companyculture
companyvalues
benefitsgreat
workingfacebook
reallygood
facebookamazing

Cons

Advice to management

Google
Pros

Cons

Advice to management

These wordplots give us an overall sentiment about a company.

Netflix Word Clouds

The goal of the word clouds are to get an overall sentiment of a company, what areas of the company are excelling or not doing well. Netflix has an interesting "cons" word cloud becuase we can match directly the word cloud sentiment with the metrics we recieved in the study. For example, the word cloud tells us that Netflix has "high turnover", "culture fear", and "job security" most repeated. We can see in our plots from Tableau that Netflix does in fact have very low ratings for career opportunities.

Google Word Clouds

The wordplot of Google's pros from above show us the overall sentiment of the pros of the company. Google seems to have a lot of perks outside of the "work" part of work. They have "food massage", "great place", "food lot", etc. so they must have a good environment to work in.

On the other hand, Google's "cons" wordplot shows they may struggle with management.

Amazon Word Clouds

Amazon's "pros" wordplot describes things like "work schedule", "work day", and "work week", which might suggest it gives employees good hours.

However, their "cons" section tells a different story. We can see that there are many references to time and breaks so there might be two subsets of employees at the company. The kind who enjoy the time/work (those who put work in the pros section) and the kind who do not enjoy the time/work.

Lastly, we wanted to check out the advice to management for Amazon. We can see that in fact Amazon could do a lot of work in treating employees more like real people. An interesting observation given the fact that Amazon employees so many (556,000), and they struggle most at treating them "like human".

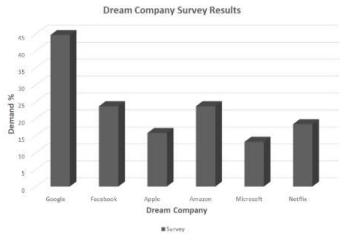
Tableau work

1. Comparing the survey results with the ratings given by the employees

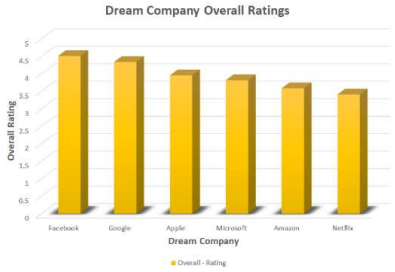
In [47]:

```
from IPython.display import HTML, display
display(HTML("<table><tr><td><img src='survey.PNG'></td><td><img src='Overall-Rating.PNG'><
```

Heinz Survey - Dream Company



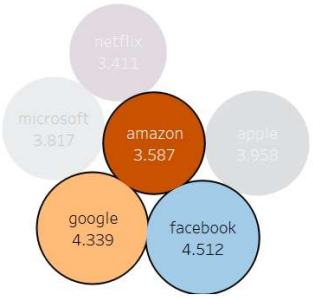
Overall Ratings



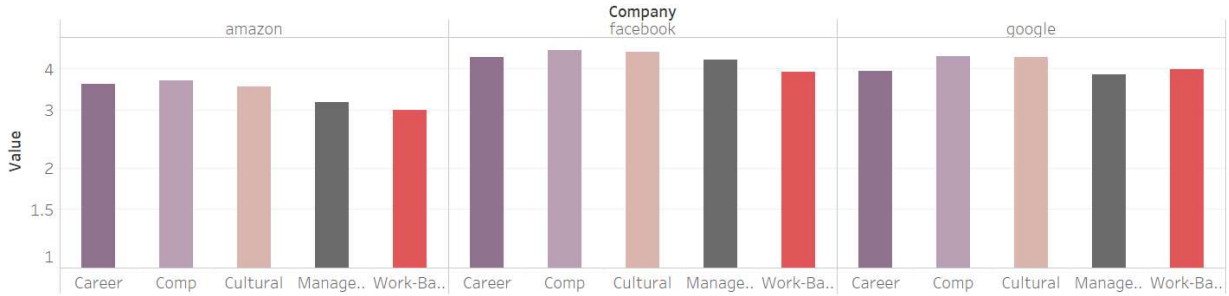
2. Drivers affecting the overall performance of the company

1. Career Opportunities 2. Compensational Benefits 3. Cultural values 4. Senior Management 5. Work Life Balance

Company Overall-Ratings



Ratings in various departments



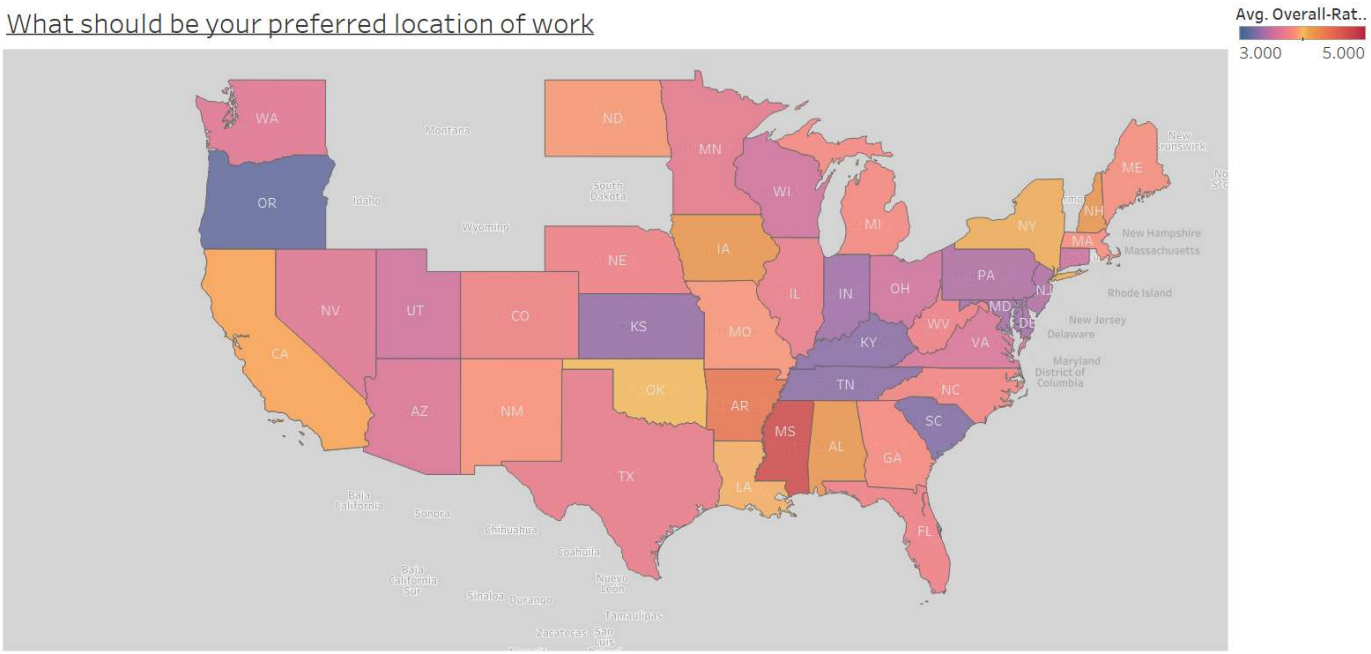
3. Comparing the metrics across current and former employees :

In [48]:

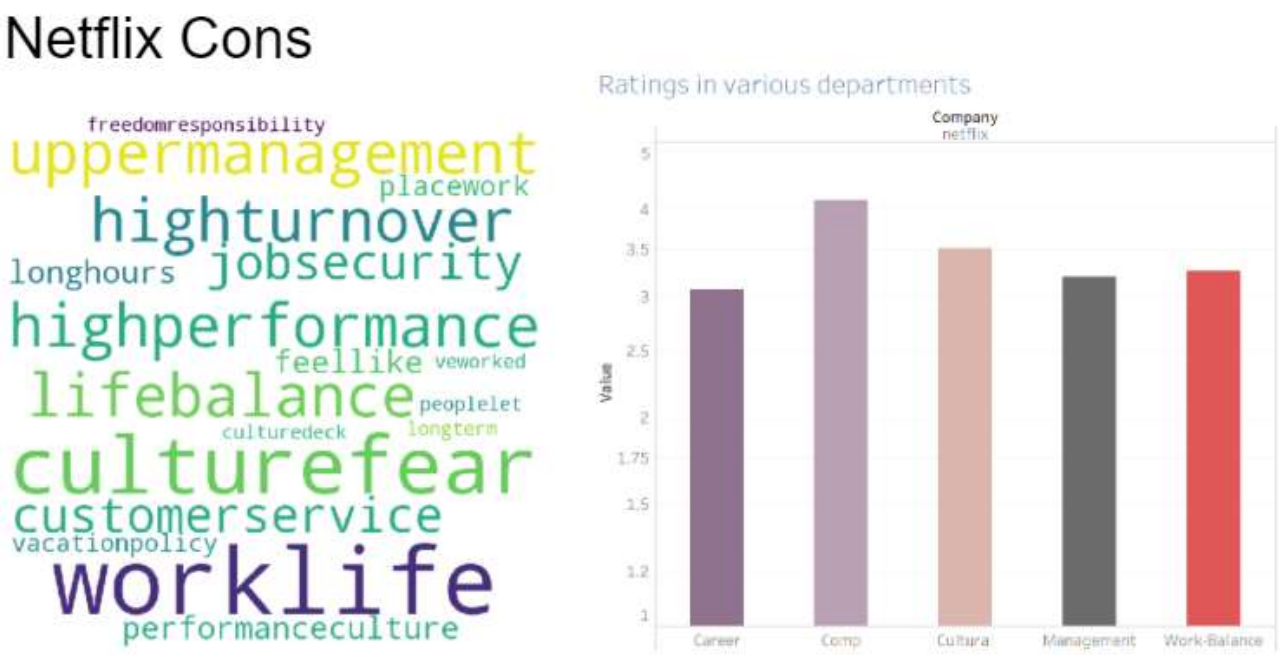
```
display(HTML("<table><tr><td><img src='Senior_FormerVsCurrent.PNG'></td><td><img src='Work_
```

4. Best Countries to work for

5. Preferred locations to work in United States



6. Reasons that affected company ratings in each metric

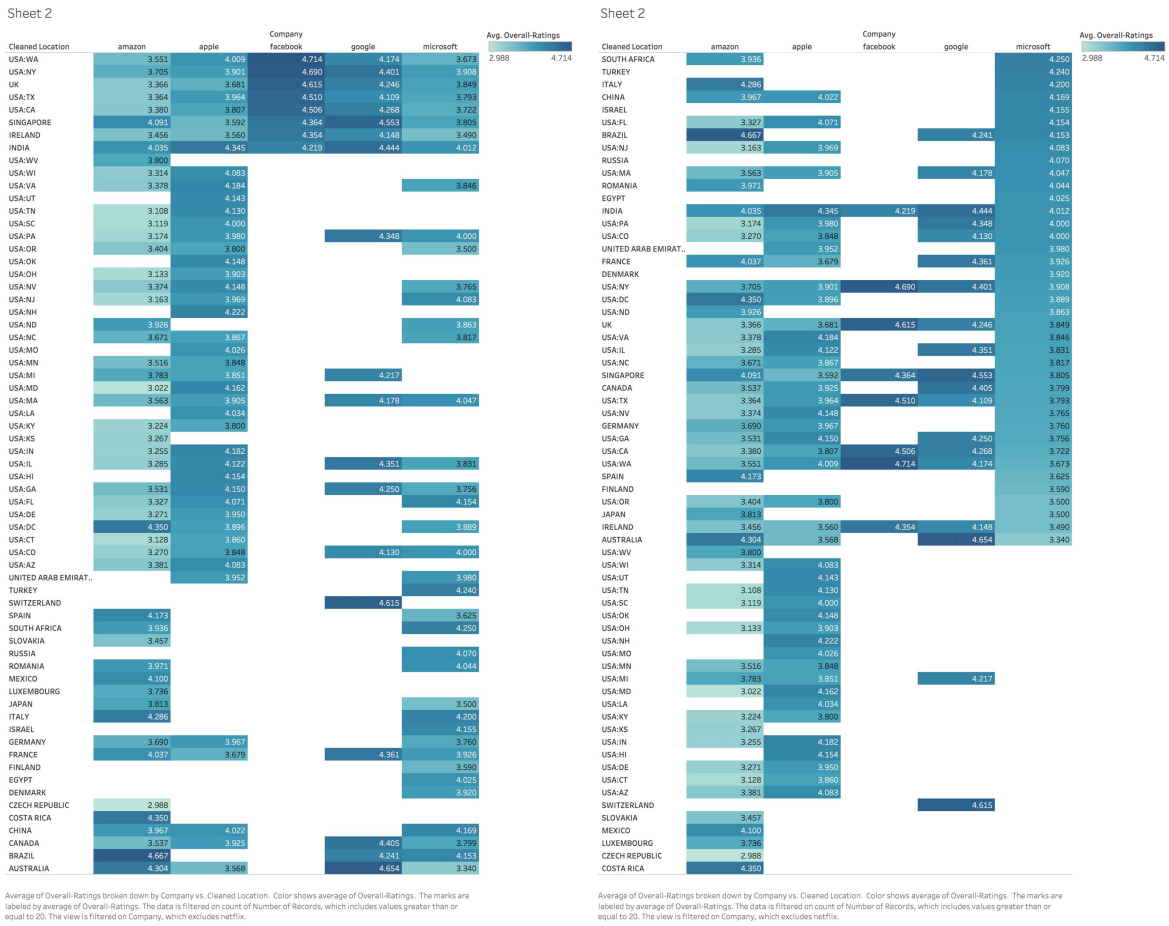


Here we can see that, words like job security indicates not much career opportunitites in the firm which has resulted in lower rating for the company in the career opportunities domain. Also, words like uppermanagement and culture fear support our analysis for the cultural and the management domain

7. Comparing each location for each company

In [49]:

```
display(HTML("<table><tr><td><img src='Locations_Tableau.png'></td><td><img src='Locations_
```



On the left we sorted the rows based on Facebook's top reviewed locations. We can see that Facebook dominates many locations in which many other companies are present.

On the right we sorted the rows based on Microsoft's top locations. We can see that Microsoft seems to do best in places where other big tech companies have not found a site. This may suggest that Microsoft has reached out to more locations to get first choice on the best talent.