Exploratory Data Analysis - Data Storytelling

Sinead O'Connor

In [1]:

```
import pyspark
from pyspark.sql import SparkSession

import pyspark.sql.functions as f
from pyspark.sql.functions import udf
from pyspark.sql.functions import col, split
from pyspark.sql.functions import array_contains

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns

import datetime

import geopandas as gpd
import descartes
import plotly as py
import plotly.graph_objs as go
import plotly.figure_factory as ff

import json
import nltk
from nltk.tokenize import word_tokenize

import collections
from collections import Counter

import re
from wordcloud import WordCloud, STOPWORDS
from PIL import Image


%matplotlib inline
```

In [2]:

```
spark = SparkSession.builder.getOrCreate()
sc = spark.sparkContext
```

In [3]:

```
restBizDF = spark.read.json('../SavedFiles/restBiz.json')
restReviewDF = spark.read.json('../SavedFiles/restReview.json')
```

In [4]:

```
restBizDF.show(5)
```

```
+--------------------+--------------------+---------
-----------+--------------------+-------------------
-+--------+--------------------+-------+-----------+--
----------+--------------------+-----------+--------
----+-----+-----+
|             address|          attributes|
business_id|          categories|      categoriesLis
t|    city|               hours|is_open|   latitude|
longitude|                name|postal_code|review_co
unt|stars|state|
+--------------------+--------------------+---------
-----------+--------------------+-------------------
-+--------+--------------------+-------+-----------+--
----------+--------------------+-----------+--------
----+-----+-----+
| 1859 Dickerson Blvd|[,,,,,,,, True,,,...|yApj23pqS
4s3dHX3G...|Chinese, Restaurants|[Chinese, Restaur..
.| Monroe|[11:0-23:0, 11:0-...|      1|35.0070923| -
80.5639574|      Jade Kitchen 2|      28110|
5|  2.5|   NC|
|  365 Bloor Street E|[, u'beer_and_win...|Hnyd6ie_c
xV_H1Vqv...|Sandwiches, Resta...|[Sandwiches, Rest..
.|Toronto|[7:30-19:30, 7:30...|      1|43.6719155| -
79.3779708|  Aroma Espresso Bar|    M4W 3L4|
13|  4.0|   ON|
|7240 Woodbine Ave...|[, u'none', {'rom...|qyLjLT8Wq
UQWRyUKe...|Sandwiches, Delis...|[Sandwiches, Deli..
.|Markham|                null|      0|43.8197869| -
79.3509435|            Shopsy's|    L3R 1A4|
18|  3.0|   ON|
|163 Quarry Park B...|[,, {'romantic': ...|6qNd-Hu4X
e9RVCFk-...|Japanese, Sushi B...|[Japanese, Sushi ..
.|Calgary|                null|      0|50.9630987|-1
14.0113272|Sushi Kitchan Jap...|    T2C 4J2|
```

```
3|  3.0|   AB|
|401 9 Avenue SW, ...|[, u'full_bar', {...|mr-w96_vg
HpfLmQtf...|Chinese, Vietname...|[Chinese, Vietnam..
.|Calgary|[11:0-21:0, 11:0-...|      1|51.0444388|-1
14.0698895|Oriental Phoenix ...|    T2P 3C5|
17|  3.0|   AB|
+--------------------+--------------------+---------
-----------+--------------------+-------------------
-+-------+--------------------+-------+----------+--
----------+--------------------+-----------+--------
----+-----+-----+
only showing top 5 rows
```

In [5]:

```
restReviewDF.show(5)
```

```
+--------------------+-------------------+----------
----------+-----+--------------------+------+
|         business_id|               date|
review_id|stars|                text|useful|
+--------------------+-------------------+----------
----------+-----+--------------------+------+
|uSI9HRI2lszekJzar...|2011-02-27 16:16:28|TWEj-8FYux
mXOW0jg...|  3.0|A local joint whi...|     2|
|kbaXNZLUyVuWbeQxH...|2017-02-23 02:25:41|GlpRwX-Wx6
gYs_noo...|  5.0|The big foodie su...|     0|
|k5pA0N9K2zy5OQZSq...|2018-03-09 06:05:35|rqwiAf2LRU
s-60gd8...|  4.0|I love eating her...|     0|
|4dCOilGYflzGzizOP...|2016-11-29 05:02:55|yx10EwGKTu
EnLjTJu...|  4.0|This place is in ...|     0|
|o2nt0ZJP0WmdDL-4H...|2018-02-17 01:54:22|gnQHrAkM94
4U7e5z3...|  1.0|We went in and sa...|     1|
+--------------------+-------------------+----------
----------+-----+--------------------+------+
only showing top 5 rows
```

Question 1: What is the distribution of star ratings for restaurant reviews?

In [6]:

```
restReviewDF[['stars']].describe().show()
```

```
+-------+------------------+
|summary|             stars|
+-------+------------------+
|  count|           3643450|
|   mean|3.7080508858362267|
| stddev|  1.37636938633259|
|    min|               1.0|
|    max|               5.0|
+-------+------------------+
```

In [7]:

```
restReviewDF.groupBy().agg(f.expr('percentile_approx(stars, 0.5)
').alias('med_val')).show()
```

```
+-------+
|med_val|
+-------+
|    4.0|
+-------+
```

In [8]:

```
star_counts = restReviewDF.groupBy('stars').count().toPandas()
```

In [9]:

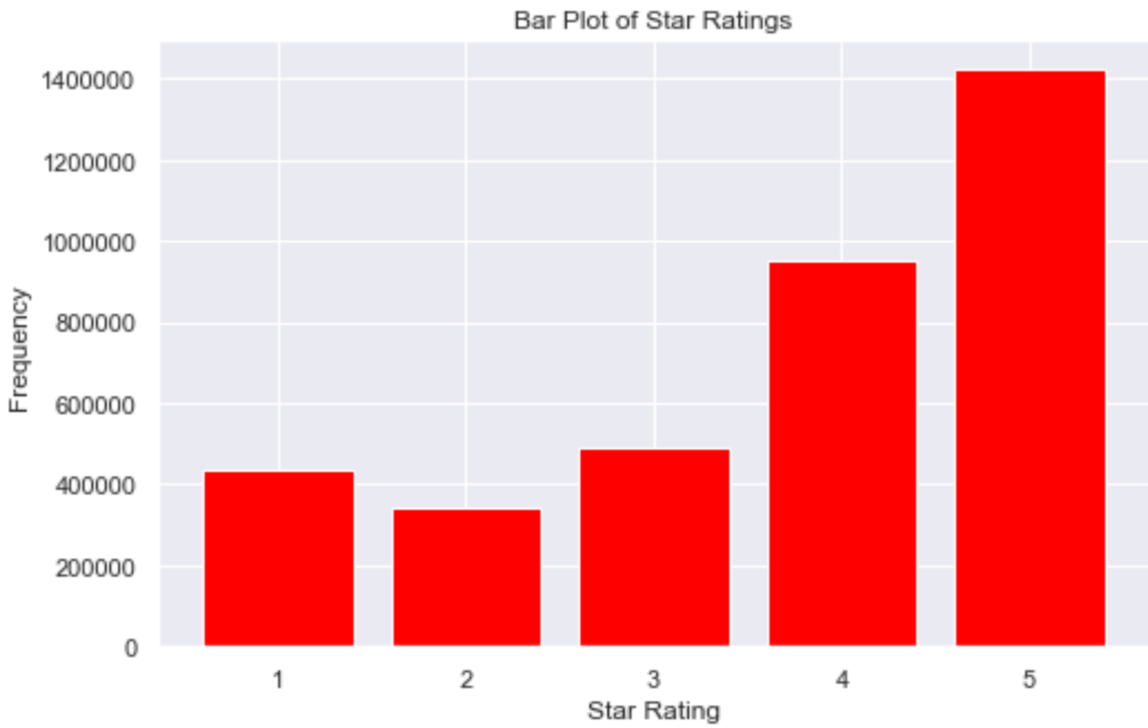
```
star_counts = star_counts.sort_values('count')
star_counts
```

Out[9]:

	stars	count
3	2.0	343775
0	1.0	436295
2	3.0	489091
1	4.0	952465
4	5.0	1421824

In [10]:

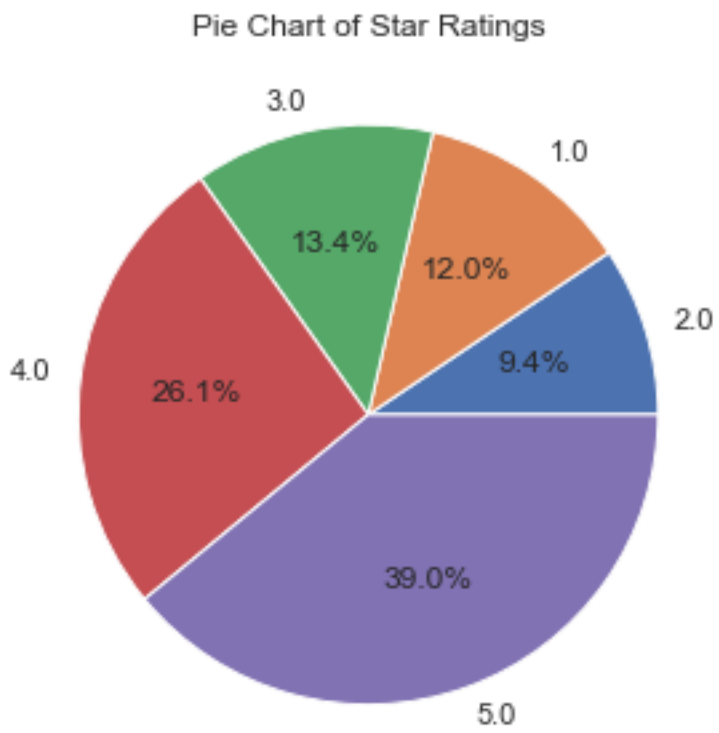
```
sns.set()
plt.figure(figsize=(8,5))
ax = plt.bar(star_counts['stars'],star_counts['count'], color =
'red')
plt.xlabel('Star Rating')
plt.ylabel('Frequency')
plt.title('Bar Plot of Star Ratings')
plt.savefig('../SavedFiles/ratingsBar')
```



In [11]:

```
sns.set()
plt.figure(figsize=(8,5))
ax = plt.pie(star_counts['count'], labels = star_counts['stars']
, autopct='%1.1f%%')

plt.title('Pie Chart of Star Ratings')
plt.savefig('../SavedFiles/ratingsPie')
```



From the above bar chart and pie chart, we see that most of the reviews were 4- and 5-star reviews. There may be more highly-rated reviews because the restaurant's good. People also may want to reserve 1- or 2-star ratings for restaurants that are particurly bad. Another reason the ratings may tend to be more favorable is because people are less likely to go to bad restaurants, so there would in turn be fewer reviews.

Question 2: Does average star rating differ by state?

In [12]:

```
rel_bizDF = restBizDF[['business_id','state']]
revs_with_state = restReviewDF.join(rel_bizDF, on = 'business_id
', how='left_outer')
```

In [13]:

```
starsByState = revs_with_state.groupBy('state').agg(f.avg(revs_w
ith_state.stars), f.count(revs_with_state.stars)).toPandas()
```

In [14]:

```
starsByState.sort_values('avg(stars)')
```

Out[14]:

	state	avg(stars)	count(stars)
4	BC	1.333333	3
19	AR	2.000000	7
18	FL	2.200000	10
9	WA	3.333333	3
14	ON	3.541205	518338
1	SC	3.579360	13905
8	IL	3.588263	25560
15	AB	3.624103	56010
7	NC	3.650980	238187
13	NY	3.654321	81
11	OH	3.666499	202569
16	CON	3.666667	3
12	PA	3.685806	178864
6	WI	3.689474	80573
0	AZ	3.737472	1056142
5	NV	3.776024	1154070
3	QC	3.825868	119105
10	XGM	4.000000	3
2	VA	4.111111	9
17	XWY	4.500000	8

In [15]:

```
starsByState.shape[0]
```

Out[15]:

```
20
```

In [16]:

```
starsByState = starsByState[starsByState['count(stars)']>10000]
```

In [17]:

```
fp = '../DownloadedFiles/us_can_shapefiles/ne_50m_admin_1_states
_provinces_shp.shp'
map_df = gpd.read_file(fp)
map_df.head()
```

Out[17]:

	featurecla	scalerank	adm1_code	diss_me	adm1_cod_1	iso_3166_2
0	Admin-1 scale rank	2.0	AUS-2651	2651	AUS-2651	AU-
1	Admin-1 scale rank	2.0	AUS-2650	2650	AUS-2650	AU-
2	Admin-1 scale rank	2.0	AUS-2655	2655	AUS-2655	AU-
3	Admin-1 scale rank	2.0	AUS-2657	2657	AUS-2657	AU-
4	Admin-1 scale rank	2.0	AUS-2660	2660	AUS-2660	AU-

5 rows × 41 columns

In [18]:

```
map_df.sr_adm0_a3.unique()
```

Out[18]:

```
array(['AUS', 'BRA', 'CAN', 'USA'], dtype=object)
```

In [19]:

```
map_df = map_df[(map_df.sr_adm0_a3 == 'USA')|(map_df.sr_adm0_a3
== 'CAN')]
map_df.head()
```

Out[19]:

	featurecla	scalerank	adm1_code	diss_me	adm1_cod_1	iso_3166_2
36	Admin-1 scale rank	2.0	CAN-632	632	CAN-632	CA-AB
37	Admin-1 scale rank	2.0	CAN-633	633	CAN-633	CA-BC
38	Admin-1 scale rank	2.0	CAN-630	630	CAN-630	CA-MB
39	Admin-1 scale rank	2.0	CAN-684	684	CAN-684	CA-NB
40	Admin-1 scale rank	2.0	CAN-686	686	CAN-686	CA-NL

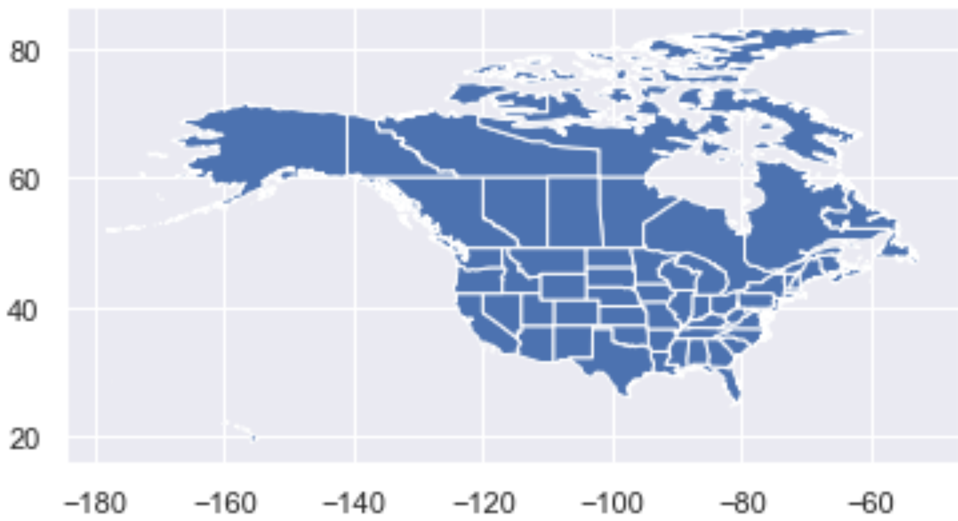
5 rows × 41 columns

In [20]:

```
from descartes.patch import PolygonPatch
map_df.plot()
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1e741c
f8>
```



In [21]:

```
merged_plot = map_df.set_index('postal').join(starsByState.set_i
ndex('state'))
```

In [22]:

```
merged_plot = merged_plot.fillna(1)
merged_plot.head()
```

Out[22]:

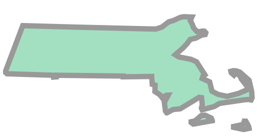
postal	featurecla	scalerank	adm1_code	diss_me	adm1_cod_1	iso_31(
AB	Admin-1 scale rank	2.0	CAN-632	632	CAN-632	CA
BC	Admin-1 scale rank	2.0	CAN-633	633	CAN-633	CA
MB	Admin-1 scale rank	2.0	CAN-630	630	CAN-630	CA
NB	Admin-1 scale rank	2.0	CAN-684	684	CAN-684	CA
NL	Admin-1 scale rank	2.0	CAN-686	686	CAN-686	CA

5 rows × 42 columns

In [23]:

```
merged_plot.geometry['MA']
```

Out[23]:



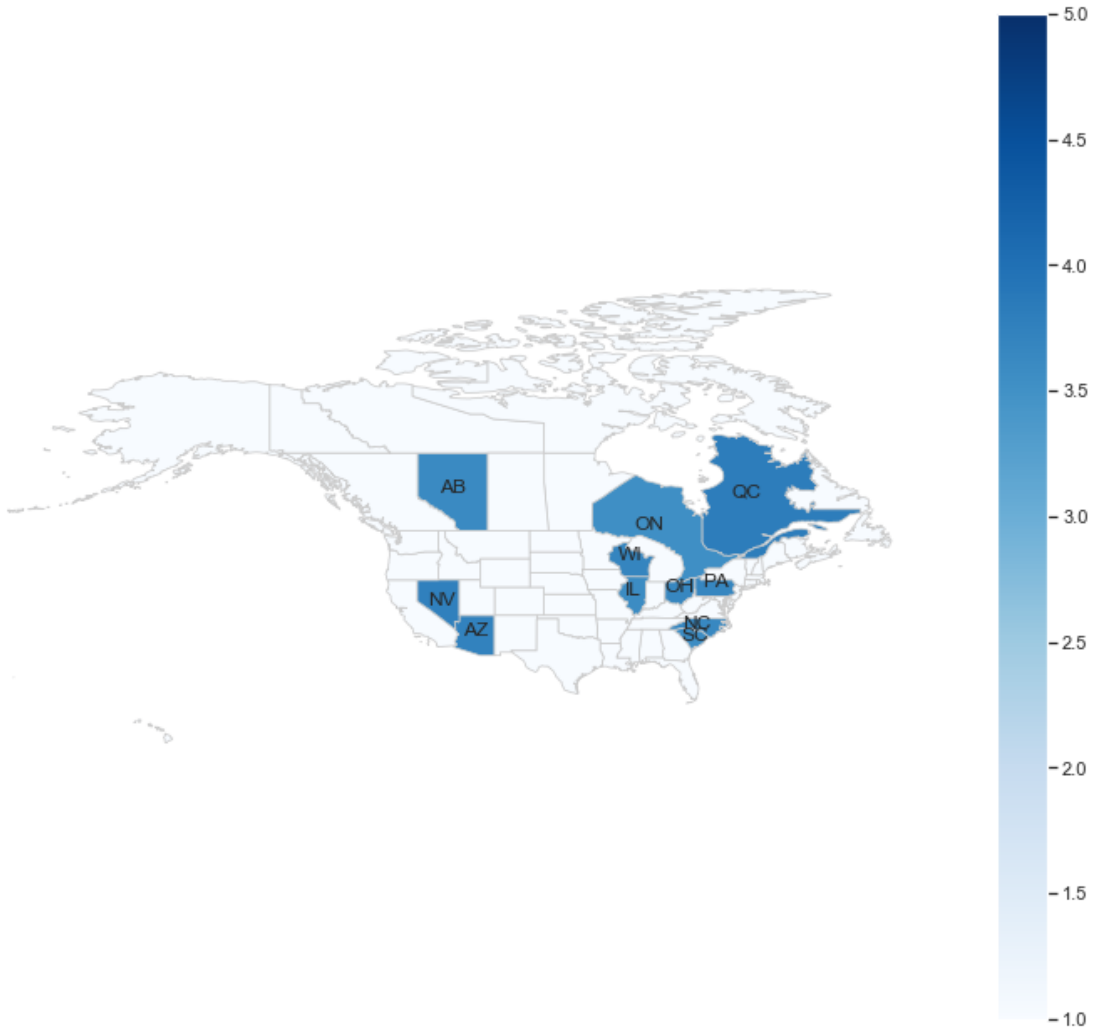
In [24]:

```
variable = 'avg(stars)'

fig, ax = plt.subplots(1, figsize=(13,11))
merged_plot.plot(column = variable, vmin = 1, vmax = 5, cmap = '
Blues', legend= True,
                 linewidth = 0.8, ax=ax, edgecolor = '0.8')

merged_plot['coords'] = merged_plot['geometry'].apply(lambda x:
x.representative_point().coords[:])
merged_plot['coords'] = [coords[0] for coords in merged_plot['co
ords']]
for idx, row in merged_plot.iterrows():
    if row['avg(stars)']>1:
        plt.annotate(s=row['iso_3166_2'][-2:], xy=row['coords'],
horizontalalignment='center')

ax.axis('off')
plt.savefig('../SavedFiles/revsByStateMap1to5')
```



Of the states/provinces with more than 10,000 restaurant reviews in the dataset, we see that average rating does not appear to differ greatly from state to state. The lowest possible rating is 1 and the highest possible is 5, and we see on the above plot that it is difficult to distinguish between the shades of blue, however Nevada and Quebec seem to have higher ratings, while areas like Ontario and Illinois appear to have lower ratings, which is confirmed in the above dataframe (cell). The lowest average rating is 3.54 (Ontario) and the highest is 3.82 (Quebec). The map below which uses a color bar ranging from 3 to 4 instead, allows us to better visualize the slight differences in rating between the different states and provinces.

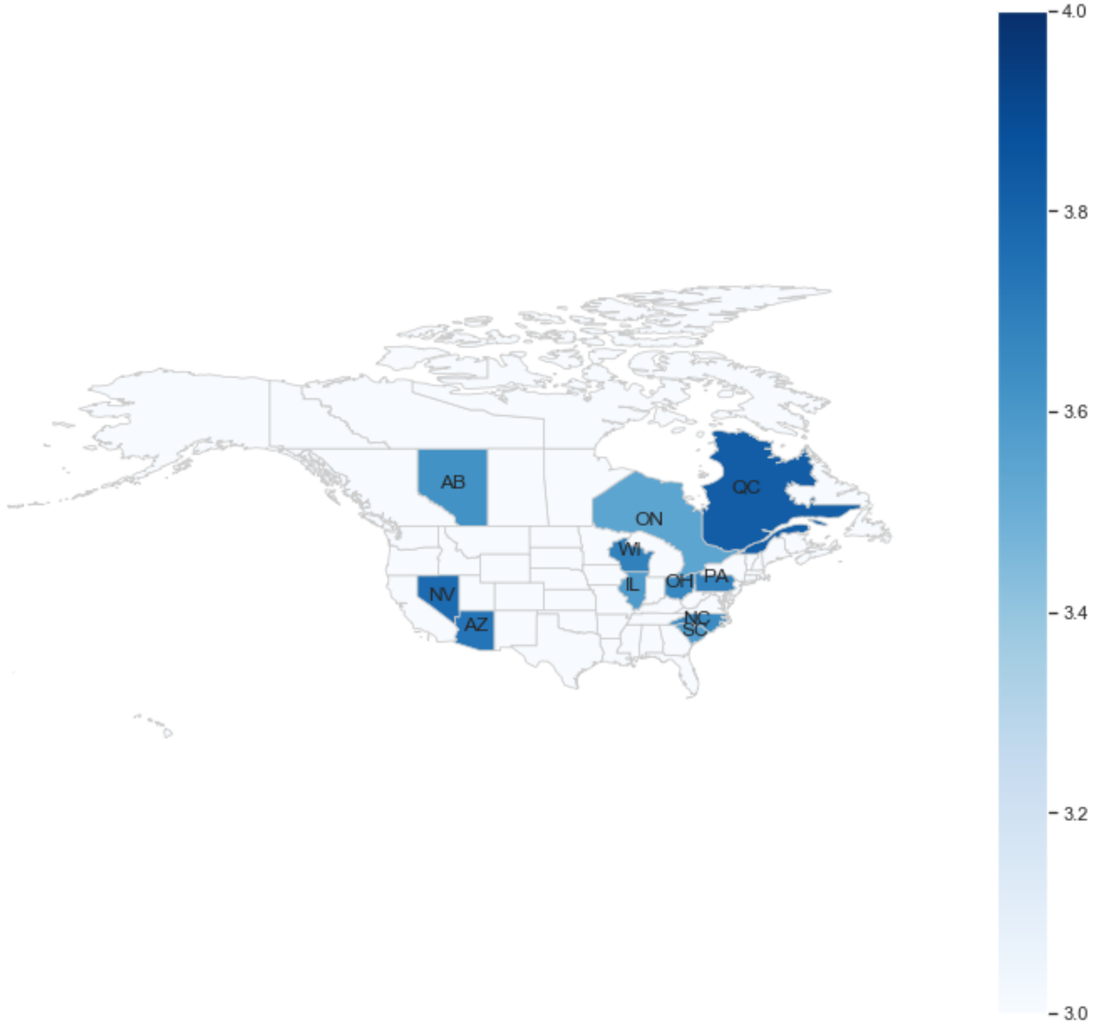
In [25]:

```
variable = 'avg(stars)'

fig, ax = plt.subplots(1, figsize=(13,11))
merged_plot.plot(column = variable, vmin = 3, vmax = 4, cmap = 'Blues', legend= True,
                 linewidth = 0.8, ax=ax, edgecolor = '0.8')

merged_plot['coords'] = merged_plot['geometry'].apply(lambda x: x.representative_point().coords[:])
merged_plot['coords'] = [coords[0] for coords in merged_plot['coords']]
for idx, row in merged_plot.iterrows():
    if row['avg(stars)']>1:
        plt.annotate(s=row['iso_3166_2'][-2:], xy=row['coords'], horizontalalignment='center')

ax.axis('off')
plt.savefig('../SavedFiles/revsByStateMap3to4')
```



Question 3: Are fast-food establishments rated differently?

In [26]:

```
fast_food = restBizDF.filter(array_contains(restBizDF.categories
List, 'Fast Food'))
fast_food_id = fast_food.select('business_id').rdd.flatMap(lambd
a x: x).collect()
```

In [27]:

```
fast_food[['business_id','name','city','state','stars']].show(10
)
```

```
+--------------------+--------------------+---------
+-----+-----+
|         business_id|                name|     city
|state|stars|
+--------------------+--------------------+---------
+-----+-----+
|zXb5pP4zMHvmusweY...|          McDonald's|  Madison
|   WI|  3.0|
|2O1GOiDhvXywzysqA...| Budweiser Brewhouse|Las Vegas
|   NV|  2.5|
|vCFIhkndnlmJg5-Uz...|Ste-Catherine & B...| Montréal
|   QC|  3.5|
|vp77iQDlV1OkbjNhm...|           Ichi Bowl|  Phoenix
|   AZ|  2.0|
|NJ0RzuWd5xDqfJejY...|               Delux|  Phoenix
|   AZ|  4.0|
|PlIeEj0LRr3p1rX5S...|      Firehouse Subs|     Mesa
|   AZ|  3.5|
|sAZGdlYTp41UWEjHd...|Cheba Hut Toasted...|Las Vegas
|   NV|  4.5|
|YF8HRRyDdEglMg9gz...|                 KFC|  Toronto
|   ON|  3.5|
|7kXrUSjG67NitjRfR...|          McDonald's|Las Vegas
|   NV|  2.0|
|kUEBUoKxJLhgs2Bp_...|              Subway|  Calgary
|   AB|  1.0|
+--------------------+--------------------+---------
+-----+-----+
only showing top 10 rows
```

In [28]:

```
fast_food_rev = restReviewDF.where(restReviewDF.business_id.isin
(fast_food_id))
```

In [29]:

```
not_fast_rev = restReviewDF.where(~restReviewDF.business_id.isin
(fast_food_id))
```

In [30]:

```
fast_food_rev.select('stars').describe().show()
```

```
+-------+------------------+
|summary|             stars|
+-------+------------------+
|  count|            220507|
|   mean| 3.175350442389584|
| stddev|1.5765903477405725|
|    min|               1.0|
|    max|               5.0|
+-------+------------------+
```

In [31]:

```
not_fast_rev.select('stars').describe().show()
```

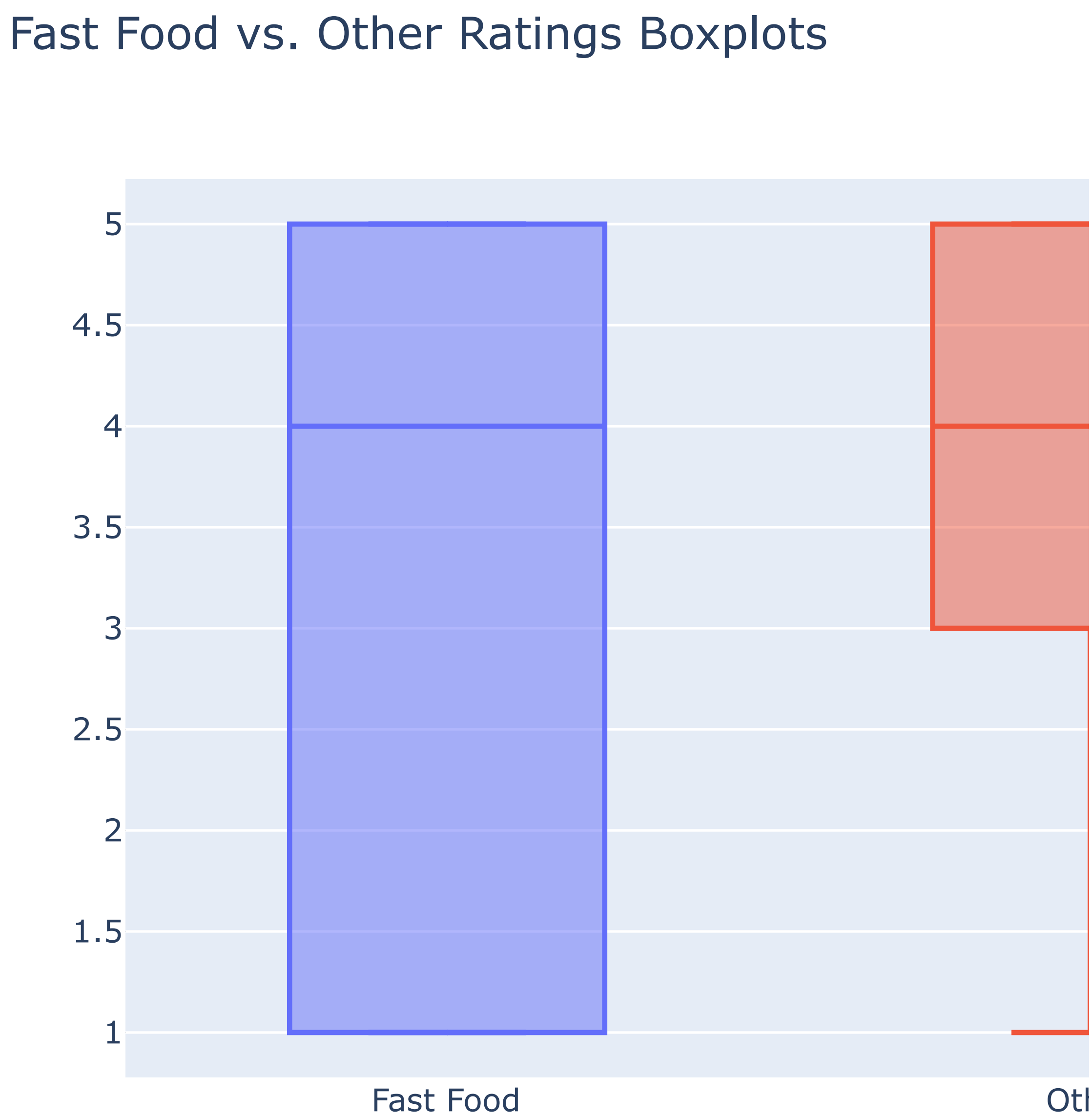
```
+-------+-----------------+
|summary|            stars|
+-------+-----------------+
|  count|          3422943|
|   mean|3.742367605887682|
| stddev|1.355303318931365|
|    min|              1.0|
|    max|              5.0|
+-------+-----------------+
```

In the Yelp dataset, there are 220,507 reviews of fast food restaurants and 3,422,943 reviews of non-fast food restaurants. The average rating of fast food establishments (3.175 stars) is about 0.56 stars lower than the average rating of other restaurants (3.742 stars).

In [32]:

```
data = [
    go.Box(y = fast_food_rev.select('stars').toPandas()['stars']
, name = 'Fast Food'),
    go.Box(y = not_fast_rev.select('stars').toPandas()['stars'],
name = 'Other')
]

layout = go.Layout(title = 'Fast Food vs. Other Ratings Boxplots
')
go.Figure(data=data, layout=layout).show()
```



From the above box plots, we see that while both fast food and non-fast food restaurants have a median rating of 4 stars, the spread is much different. The 1st quartile of fast food rating is 1, meaning at least 25% of fast-food reviews in the Yelp dataser rated 1-stars. On the other hand, for non-fast-food restaurants, at least 75% of Yelp reviewers gave ratings of 3-stars or higher. Now, let's visualize the distribution of star-ratings for fast food establishments.

In [33]:

```
star_counts = fast_food_rev.groupBy('stars').count().toPandas()
```

In [34]:

```
star_counts.sort_values('count')
```

Out[34]:

	stars	count
3	2.0	24896
2	3.0	28701
1	4.0	45886
0	1.0	56093
4	5.0	64931

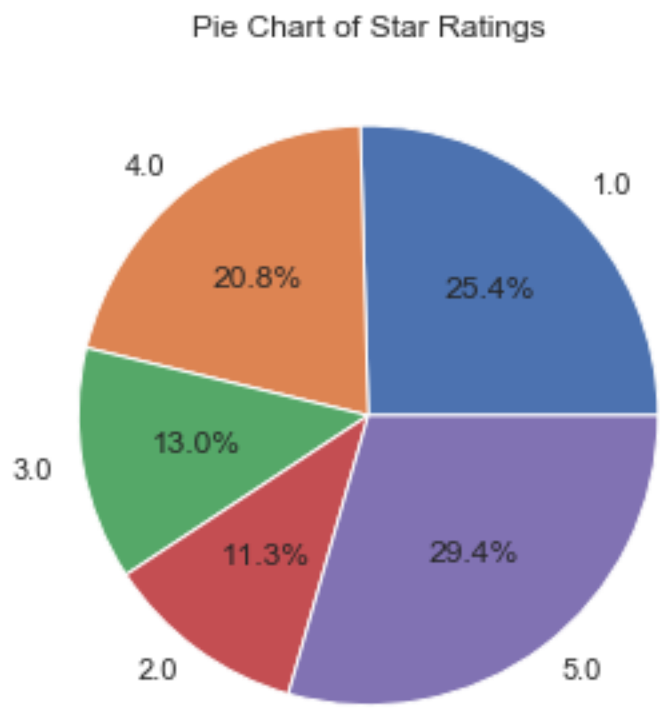
In [35]:

```
sns.set()
plt.figure(figsize=(8,5))
ax = plt.bar(star_counts['stars'],star_counts['count'], color =
'red')
plt.xlabel('Star Rating')
plt.ylabel('Frequency')
plt.title('Bar Plot of Star Ratings of Fast Food Restaurants')
plt.savefig('../SavedFiles/fastRatingsBar')
```

In [36]:

```
plt.figure(figsize=(8,5))
ax = plt.pie(star_counts['count'], labels = star_counts['stars']
, autopct='%1.1f%%')

plt.title('Pie Chart of Star Ratings of Fast Food Restaurants')
plt.savefig('../SavedFiles/fastRatingsPie')
```



In the above plots, we see that the most common ratings of fast-food restaurant are 5 and 1

Question 4: Has average star-ratings of fast food restaurants changed over time? Non-fast-food restaurants?

In [ ]:

```

```

In [37]:

```
#get average reviews and # of reviews per month for fast food re
staurants
fastReviewByM = fast_food_rev.groupby(fast_food_rev['date'][0:7]
).agg(f.avg(fast_food_rev.stars), f.count(fast_food_rev.stars)).
toPandas()
```

In [38]:

```
notfastReviewByM = not_fast_rev.groupby(not_fast_rev['date'][0:7
]).agg(f.avg(not_fast_rev.stars), f.count(not_fast_rev.stars)).t
oPandas()
```

In [39]:

```
fastReviewByM.head()
```

Out[39]:

	substring(date, 0, 7)	avg(stars)	count(stars)
0	2013-05	3.344106	1052
1	2009-07	3.657025	242
2	2018-10	2.940550	4037
3	2013-09	3.377023	1236
4	2010-08	3.609375	512

In [40]:

```
fastReviewByM = fastReviewByM.rename(columns = {'substring(date,
0, 7)':'date'})
notfastReviewByM = notfastReviewByM.rename(columns = {'substring
(date, 0, 7)':'date'})
```

In [41]:

```
fastReviewByM['date'] = pd.to_datetime(fastReviewByM['date'])
notfastReviewByM['date'] = pd.to_datetime(notfastReviewByM['date
'])
```

In [42]:

```
fastReviewByM['date'].describe()
```

Out[42]:

```
count                      164
unique                     164
top        2016-05-01 00:00:00
freq                         1
first      2004-10-01 00:00:00
last       2018-11-01 00:00:00
Name: date, dtype: object
```

In [43]:

```
notfastReviewByM['date'].describe()
```

Out[43]:

```
count                      168
unique                     168
top        2016-05-01 00:00:00
freq                         1
first      2004-10-01 00:00:00
last       2018-11-01 00:00:00
Name: date, dtype: object
```

In [44]:

```
fastReviewByM.describe()
```

Out[44]:

	avg(stars)	count(stars)
count	164.000000	164.000000
mean	3.387523	1344.554878
std	0.342146	1422.429630
min	1.000000	1.000000
25%	3.181272	117.250000
50%	3.394214	805.500000
75%	3.542407	2405.750000
max	4.500000	4778.000000

In [45]:

```
notfastReviewByM.describe()
```

Out[45]:

	avg(stars)	count(stars)
count	168.000000	168.000000
mean	3.746329	20374.660714
std	0.146198	19268.152662
min	3.500000	2.000000
25%	3.657918	2357.500000
50%	3.704977	15350.000000
75%	3.789534	39456.500000
max	4.541667	61602.000000

In [46]:

```
fastReviewByM = fastReviewByM.set_index('date').sort_index()
notfastReviewByM = notfastReviewByM.set_index('date').sort_index
()

plt.figure(figsize = (12,4))

plt.plot(fastReviewByM['avg(stars)'], color = 'red', label = 'Fa
st Food')
plt.plot(notfastReviewByM['avg(stars)'], color = 'black', label
= 'Other')

plt.ylabel('Avg. Star Rating')
plt.title('Avg. Star Rating of Restaurants per Month')
plt.legend()
```

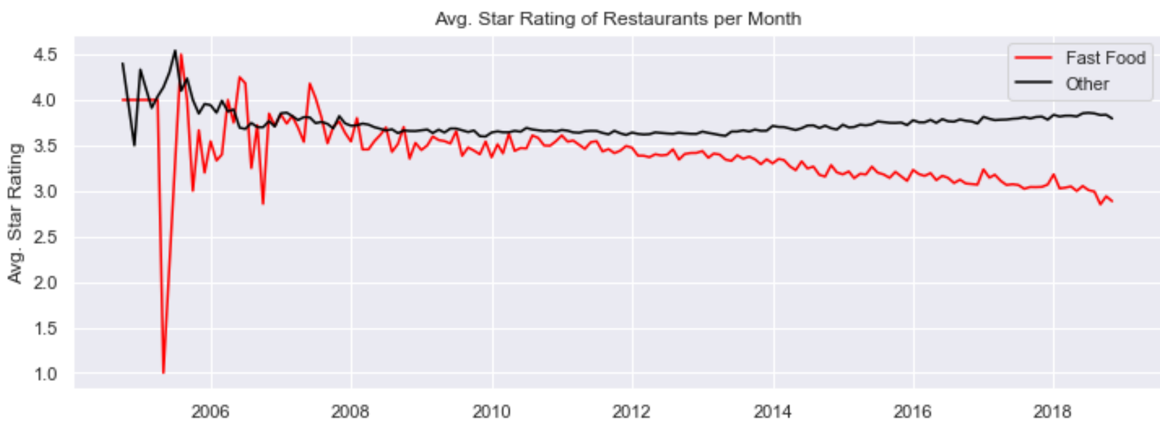
```
/Users/sineadoconnor/miniconda3/lib/python3.7/site-p
ackages/pandas/plotting/_converter.py:129: FutureWar
ning:

Using an implicitly registered datetime converter fo
r a matplotlib plotting method. The converter was re
gistered by pandas on import. Future versions of pan
das will require you to explicitly register matplotl
ib converters.

To register the converters:
        >>> from pandas.plotting import register_mat
plotlib_converters
        >>> register_matplotlib_converters()
```

Out[46]:

```
<matplotlib.legend.Legend at 0x1a2c724fd0>
```



We see that ratings were very volatile from month-to-month before 2008. This is due to the fact that in this Yelp dataset there were not many reviews per month before that, so I will zoom in...

In [47]:

```python
fastrevsPerMonth = fastReviewByM['count(stars)'].sort_index()
fastrevsPerMonth.head(50)
```

Out[47]:

```
date
2004-10-01       1
2005-02-01       1
```

2005-04-01	2
2005-05-01	1
2005-08-01	2
2005-09-01	1
2005-10-01	2
2005-11-01	6
2005-12-01	5
2006-01-01	11
2006-02-01	3
2006-03-01	5
2006-04-01	3
2006-05-01	4
2006-06-01	8
2006-07-01	11
2006-08-01	8
2006-09-01	18
2006-10-01	7
2006-11-01	20
2006-12-01	7
2007-01-01	19
2007-02-01	46
2007-03-01	57
2007-04-01	36
2007-05-01	26
2007-06-01	39
2007-07-01	47
2007-08-01	46
2007-09-01	44
2007-10-01	44
2007-11-01	34
2007-12-01	36
2008-01-01	59
2008-02-01	60
2008-03-01	68
2008-04-01	79
2008-05-01	79
2008-06-01	101
2008-07-01	119
2008-08-01	112
2008-09-01	132
2008-10-01	131
2008-11-01	127
2008-12-01	144
2009-01-01	138
2009-02-01	112
2009-03-01	157

```
2009-03-01    157
2009-04-01    154
2009-05-01    201
Name: count(stars), dtype: int64
```

In [48]:

```
fastrevsPerMonth[fastrevsPerMonth < 100]
```

Out[48]:

```
date
2004-10-01     1
2005-02-01     1
2005-04-01     2
2005-05-01     1
2005-08-01     2
2005-09-01     1
2005-10-01     2
2005-11-01     6
2005-12-01     5
2006-01-01    11
2006-02-01     3
2006-03-01     5
2006-04-01     3
2006-05-01     4
2006-06-01     8
2006-07-01    11
2006-08-01     8
2006-09-01    18
2006-10-01     7
2006-11-01    20
2006-12-01     7
2007-01-01    19
2007-02-01    46
2007-03-01    57
2007-04-01    36
2007-05-01    26
2007-06-01    39
2007-07-01    47
2007-08-01    46
2007-09-01    44
2007-10-01    44
2007-11-01    34
2007-12-01    36
2008-01-01    59
2008-02-01    60
2008-03-01    68
2008-04-01    79
2008-05-01    79
Name: count(stars), dtype: int64
```

In [49]:

```
notfastrevsPerMonth = notfastReviewByM['count(stars)'].sort_inde
x()
notfastrevsPerMonth.head(50)
```

Out[49]:

```
date
2004-10-01       5
2004-12-01       2
2005-01-01       3
2005-03-01      34
2005-04-01      26
2005-05-01      36
2005-06-01      10
2005-07-01      24
2005-08-01      10
2005-09-01      17
2005-10-01      81
2005-11-01      73
2005-12-01     113
2006-01-01     314
2006-02-01     156
2006-03-01     116
2006-04-01     189
2006-05-01     114
2006-06-01     134
2006-07-01     144
2006-08-01     272
2006-09-01     291
2006-10-01     279
2006-11-01     292
2006-12-01     297
2007-01-01     638
2007-02-01     691
2007-03-01     877
2007-04-01     608
2007-05-01     703
2007-06-01     961
2007-07-01    1079
2007-08-01    1203
2007-09-01     999
2007-10-01    1014
```

```
2007-11-01     931
2007-12-01    1059
2008-01-01    1622
2008-02-01    1575
2008-03-01    1780
2008-04-01    1918
2008-05-01    2140
2008-06-01    2430
2008-07-01    2619
2008-08-01    3257
2008-09-01    3401
2008-10-01    3042
2008-11-01    3009
2008-12-01    2999
2009-01-01    4493
Name: count(stars), dtype: int64
```

In [50]:

```
notfastrevsPerMonth[notfastrevsPerMonth < 3000]
```

Out[50]:

```
date
2004-10-01       5
2004-12-01       2
2005-01-01       3
2005-03-01      34
2005-04-01      26
2005-05-01      36
2005-06-01      10
2005-07-01      24
2005-08-01      10
2005-09-01      17
2005-10-01      81
2005-11-01      73
2005-12-01     113
2006-01-01     314
2006-02-01     156
2006-03-01     116
2006-04-01     189
2006-05-01     114
2006-06-01     134
2006-07-01     144
2006-08-01     272
2006-09-01     291
```

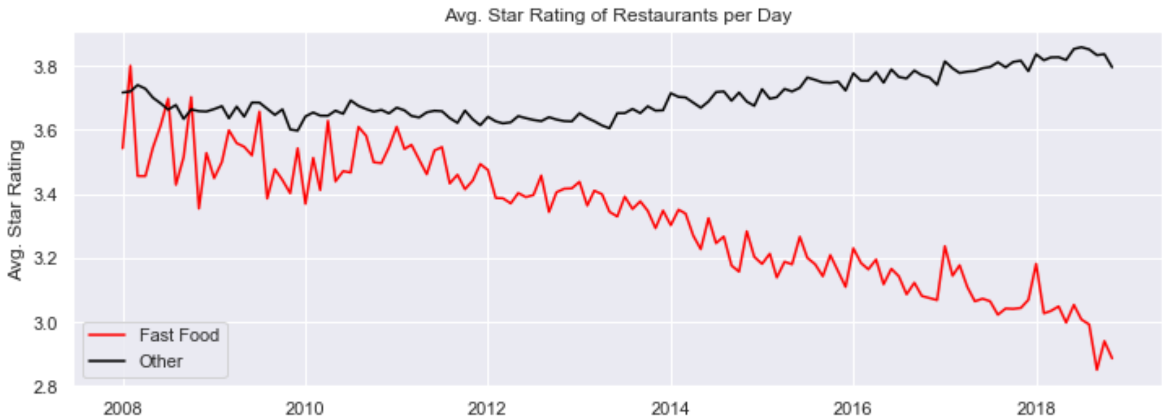
```
2006-10-01     279
2006-11-01     292
2006-12-01     297
2007-01-01     638
2007-02-01     691
2007-03-01     877
2007-04-01     608
2007-05-01     703
2007-06-01     961
2007-07-01    1079
2007-08-01    1203
2007-09-01     999
2007-10-01    1014
2007-11-01     931
2007-12-01    1059
2008-01-01    1622
2008-02-01    1575
2008-03-01    1780
2008-04-01    1918
2008-05-01    2140
2008-06-01    2430
2008-07-01    2619
2008-12-01    2999
Name: count(stars), dtype: int64
```

In [51]:

```
#zoom
plt.figure(figsize = (12,4))

plt.plot(fastReviewByM['avg(stars)']['2008-01':], color = 'red', label = 'Fast Food')
plt.plot(notfastReviewByM['avg(stars)']['2008-01':], color = 'black', label = 'Other')

plt.ylabel('Avg. Star Rating')
plt.title('Avg. Star Rating of Restaurants per Day')
plt.legend()
plt.savefig('../SavedFiles/FastvOtherTime')
```



We see that reviews of non fast food restaurants have consistently been higher than those of fast food restaurants. In addition, fast food ratings seem to have decreased overall from about 3.6 in 2011 to about 2.9 by the end of 2018. This may be because Americans have realized that fast food is not healthy. For other restaurants, we see a slight overall increase in average reviews per month from about 3.6 in 2013 to about 3.8 at the end of 2018. We are seeing more volatility in the fast food monthly ratings because the sample size is much smaller (about 200K vs about 3.4 million).

Question 5: What is the distribution of the number of words per review?

In [52]:

```
from pyspark.sql.types import IntegerType
wordCount_udf = udf((lambda text: len(re.findall('(d\+|\w+)', te
xt))), IntegerType())
```

In [53]:

```
restReviewDF = restReviewDF.withColumn('word_count', wordCount_u
df(restReviewDF.text))
restReviewDF['text','stars','word_count'].show(5)
```

```
+--------------------+-----+----------+
|                text|stars|word_count|
+--------------------+-----+----------+
|A local joint whi...|  3.0|       160|
|The big foodie su...|  5.0|       132|
|I love eating her...|  4.0|       126|
|This place is in ...|  4.0|        43|
|We went in and sa...|  1.0|        48|
+--------------------+-----+----------+
only showing top 5 rows
```

In [54]:

```
restReviewDF.select('word_count').describe().show()
```

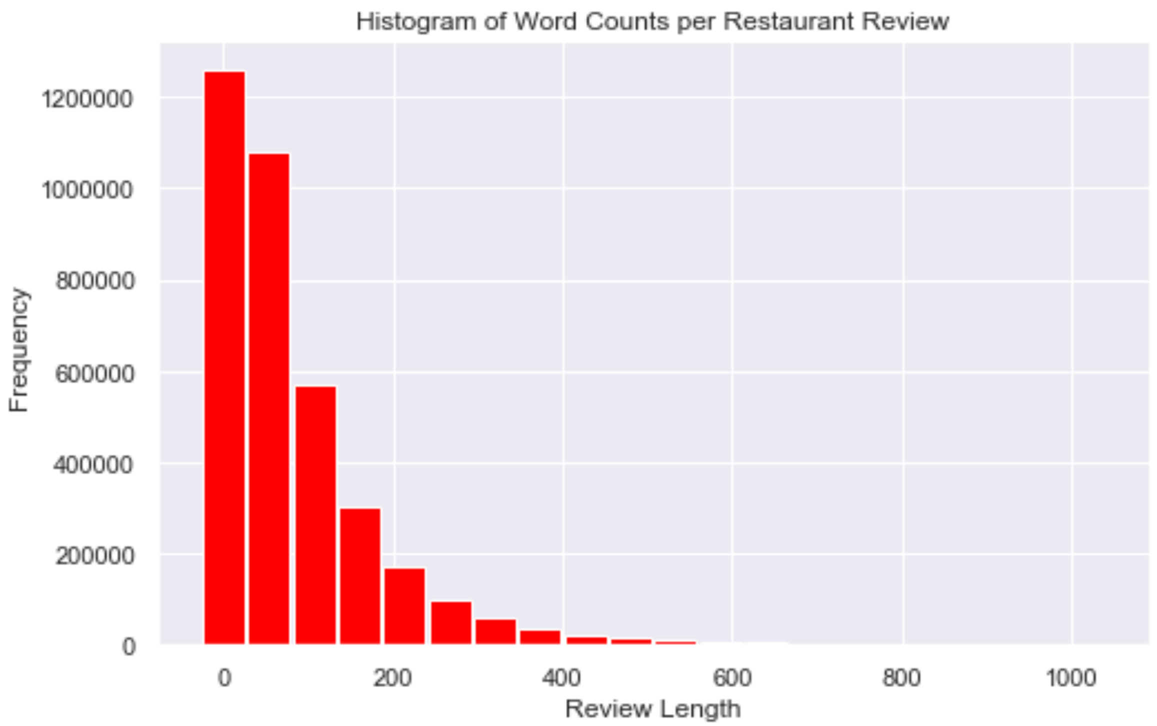
```
+-------+------------------+
|summary|        word_count|
+-------+------------------+
|  count|           3643450|
|   mean|108.85686066777367|
| stddev|  102.168260683591|
|    min|                 0|
|    max|              1067|
+-------+------------------+
```

In [55]:

```
bins, freqs = restReviewDF.select('word_count').rdd.flatMap(lamb
da x: x).histogram(20)
```

In [56]:

```
freqs_for_hist = pd.DataFrame(list(zip(bins, freqs)), columns=['
bin', 'freq'])
plt.figure(figsize=(8,5))
plt.bar(x = freqs_for_hist.bin, height = freqs_for_hist.freq, co
lor = 'red', width=50)
plt.xlabel('Review Length')
plt.ylabel('Frequency')
plt.title('Histogram of Word Counts per Restaurant Review')
plt.savefig('../SavedFiles/wordCountHist')
```



In this datset, the smallest review length is 0 words, and the highest review length is 1,067 words. We see a strong right-skew in the above histogram with most reviews bein under about 100 words. There appears to be an exponential distribution. The frequency decreases as the length of the review (in words) increases. This makes sense because most people just write a few sentences on Yelp reviews, rather than paragraphs and paragraphs.

Question 6: What are the most common words in reviews for each rating level?

In [ ]:

```

```

In [57]:

```
oneStar = fast_food_rev[fast_food_rev.stars == 1]
twoStar = fast_food_rev[fast_food_rev.stars == 2]
threeStar = fast_food_rev[fast_food_rev.stars == 3]
fourStar = fast_food_rev[fast_food_rev.stars == 4]
fiveStar = fast_food_rev[fast_food_rev.stars == 5]
```

In [58]:

```
star = np.array(Image.open('../DownloadedFiles/Plain_Yellow_Star
.png'))
#star
```

In [59]:

```
def create_cloud(df, mask = star, filename = None):
    stopwords = set(STOPWORDS)
    stopwords.update(['got','burger','burgers','location','drive
', 'thru','order','ordered','place', 'really',
                        'one', 'fast', 'food', 'thing', 'locatio
n', 'know', 'restaurant', 'now', 'meal',
                        'eat', 'drink','fries', 'sandwich', 'piz
za', 'chicken', 'made', 'make', 'think',
                        'come', 'came', 'good', 'go'])

    revs = df.select('text').rdd.flatMap(lambda x: x).collect()
    text = ' '.join(revs)
    text = text.lower()

    wordcloud = WordCloud(background_color ='white', stopwords =
stopwords,
                contour_color = 'red', contour_width = 3,
                mask = mask).generate(text)

    # plot the WordCloud image
    plt.figure(figsize = (8, 8), facecolor = None)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.tight_layout(pad = 0)

    if filename != None:
        plt.savefig('../SavedFiles/' + filename, format="png")
```

In [60]:

```
create_cloud(oneStar, filename ='1StarCloud.png')
```

In [61]:

```
create_cloud(twoStar, filename ='2StarCloud.png')
```

In [62]:

```
create_cloud(threeStar, filename ='3StarCloud.png')
```

In [63]:

```
create_cloud(fourStar, filename ='4StarCloud.png')
```

In [64]:

```
create_cloud(fiveStar, filename ='5StarCloud.png')
```

In the above word clouds, I removed words that appeared in all of the stars and that were not very informative such as restaurant and burger.

In the word-cloud for 1-star fast food reviews, we see words such as "customer service", "employee", "staff", and "manager." This indicates that people who are giving 1-star ratings are often unhappy with service. We also see the word "manager" in only the 1-star review, which makes sense because customers only typically need to speak with a manager if they are very dissatisfied. We also see the word "service" in the 2-star cloud. We also see the word "cold" in the one-star and two-star couds. Fast food customers typically expect hot food. It is also interesting that "McDonald" appears rather prominenlty in the 1-star cloud, and there are not any other restaurant names in any of the word clouds (except for a small McDonald's in the three-star cloud). McDonald's is a very popular fast-food chain, so it could be reviewers comparing other restaurants to McDonald's or it could be that McDonald's has more reviews in the dataset than other restaurants. Nonetheless, McDonald's probably does not want there brand name associated with 1-star.

The 3-star cloud, has the word "great", but keep in mind it could be "not great." Note that I removed the word "good" because all 5 clouds had it prominently featured, but before I removed it one of the words was "pretty good." There are also the words "ok" and "bad." The word "though" appearing in the word cloud also indicates that the customer may have liked some things and disliked others.

The 4-star cloud has many positive words prominently featured including "love", "delicious", "nice", and "best."

The 5-star could clearly has the most positive words. It has all the same words I mentioned in the 4-star cloud as well as adjectives like "amazing", "awesome" and "favorite", and "perfect".

It is also interesting to note that the 5-star word cloud has the phrases "customer service" and "great service." It shows that good or bad service may really make or break a review. It may be one of the things that brings a 3-star review down to a 1- or 2-star review, or what brings a 4-star review to a 5-star review.

Add columns to & save the fast food dataframe for use in inferential statistics and machine learning...

In [65]:

```
fast_food_rev.show(5)
```

```
+--------------------+-------------------+----------
----------+-----+--------------------+------+
|         business_id|               date|
review_id|stars|                text|useful|
+--------------------+-------------------+----------
----------+-----+--------------------+------+
|k5pA0N9K2zy5OQZSq...|2018-03-09 06:05:35|rqwiAf2LRU
s-60gd8...|  4.0|I love eating her...|     0|
|86ljmJqOTTeV_htLQ...|2016-02-04 15:21:16|mfWCvfsOch
HnWty89...|  5.0|Great restaurant,...|     0|
|dF11zSNDXbq6DBchU...|2012-08-29 04:50:42|w2YeqDPD52
Ul0vpr0...|  4.0|I love this locat...|     0|
|dDZqtA1me453GrMtp...|2017-04-09 21:49:20|C6RrFiWfxz
r9tIRsn...|  1.0|I like Jimmy John...|     0|
|qQsrcouREdFUk4adi...|2016-04-06 00:41:35|KysgGNwV-L
Ot_zk7i...|  5.0|First time trying...|     5|
+--------------------+-------------------+----------
----------+-----+--------------------+------+
only showing top 5 rows
```

In [66]:

```
fast_food = fast_food['business_id','name', 'city', 'state']
fastFoodDF = fast_food_rev.join(fast_food, on = 'business_id', h
ow = 'left_outer')
```

In [67]:

```
fastFoodDF = fastFoodDF.withColumn('word_count', wordCount_udf(f
astFoodDF.text))
```

In [68]:

```
fastFoodDF.show()
```

```
+--------------------+-------------------+----------
----------+-----+--------------------+------+-------
------------+-------------+-----+----------+
|         business_id|               date|
review_id|stars|                text|useful|
name|          city|state|word_count|
+--------------------+-------------------+----------
----------+-----+--------------------+------+-------
------------+-------------+-----+----------+
|k5pA0N9K2zy5OQZSq...|2018-03-09 06:05:35|rqwiAf2LRU
s-60gd8...|  4.0|I love eating her...|     0|
Burgerim|     Las Vegas|   NV|       126|
|86ljmJqOTTeV_htLQ...|2016-02-04 15:21:16|mfWCvfsOch
HnWty89...|  5.0|Great restaurant,...|     0|    Cor
vette Express|      Montréal|   QC|        18|
|dF1lzSNDXbq6DBchU...|2012-08-29 04:50:42|w2YeqDPD52
Ul0vpr0...|  4.0|I love this locat...|     0|Chipotl
e Mexican ...|    Scottsdale|   AZ|        22|
|dDZqtA1me453GrMtp...|2017-04-09 21:49:20|C6RrFiWfxz
r9tIRsn...|  1.0|I like Jimmy John...|     0|
Jimmy John's|         Tempe|   AZ|        76|
|qQsrcouREdFUk4adi...|2016-04-06 00:41:35|KysgGNwV-L
Ot_zk7i...|  5.0|First time trying...|     5|Mamajou
n Armenian...|       Toronto|   ON|       303|
|W2CzAePJakvARgoQu...|2016-04-19 22:37:40|LJm1Gb3w15
xPbQBKY...|  4.0|Been here multipl...|     1|     In
-N-Out Burger|         Tempe|   AZ|        73|
|WXSsJIO_uGGSxS9qC...|2015-07-09 19:43:54|h7djjUHWrQ
_oT3NFf...|  2.0|First let me say,...|     2|  The P
rotein Source|     Las Vegas|   NV|       274|
|i2YFMkIMqstA9GxUk...|2017-10-17 18:47:24|VCZ_0w3Fj3
xH1qn6D...|  5.0|Got the Californi...|     0|
Senor Taco|Fountain Hills|   AZ|        20|
|vofsKB4Y8MKyytL4d...|2017-07-13 11:59:08|L6TJZ6yjmc
bCnv-9p...|  1.0|Absolutely Nasty!...|     3|
Wingstop|     Charlotte|   NC|       182|
|A6bnXx1see4yZSaVV...|2017-03-11 16:33:37|vDL5v88grw
2GIEvJv...|  3.0|The food is a lit...|     1|
Rolltation|       Toronto|   ON|       106|
|F9tePBgROEAcd8xZq...|2018-09-27 21:06:40|ARndTu84Bm
c_RTkN0...|  5.0|The first time I ...|     0|
Jollibee|       Toronto|   ON|        78|
|MDtMV0ld7q0BsQPKN...|2014-10-18 20:12:15|RojWUGaHhI
zgUtz5Y...|  5.0|Consistent good e...|     0|
```

```
Five Guys|          Markham|    ON|          36|
|hE_U-gUwu-3-7CVKX...|2014-08-18 04:37:09|Gc74BXin4K
4kkyhhc...|  4.0|The thing about f...|      2|
Wendy's|           Tempe|    AZ|         210|
|40B7HIv74ZqU9tBxb...|2018-04-09 20:56:17|V1fiB2PgWI
kRRD9L4...|  5.0|Good grilled chee...|      0|        S
onic Drive-In|           Mesa|    AZ|          14|
|MJW4XuO_seWA19-fb...|2018-01-27 03:41:13|KnM9NwFYBm
ooLnP6X...|  5.0|Made to order piz...|      0|Blaze F
ast-Fire'd...|     Scottsdale|    AZ|          46|
|yBHNlSLfAd7uXFqly...|2017-12-12 20:47:52|99B_QZHq4N
sjVk4Km...|  4.0|Anything you want...|      0|
MOD Pizza|         Goodyear|    AZ|         185|
|9NrRvbS29aAav-BuO...|2018-02-07 01:29:44|a3WTjTeAJn
Lk0Wts8...|  1.0|The first time we...|      0|Popeyes
Louisiana...|        Chandler|    AZ|         108|
|Y-vxPbvPKcaLiYa9Z...|2010-06-15 22:52:11|csAiNMxU1C
Vx4OnMH...|  2.0|This little mom a...|      4|
McDonald's|      Pittsburgh|    PA|         238|
|ckbDWYPT8TP7etP0W...|2017-08-04 18:47:44|h92XAavdaE
Ap1SJ3C...|  5.0|Great, Fast Pizza...|      0|Blaze F
ast Fire'd...|     Pittsburgh|    PA|          23|
|aZshH0IfszzZyXkoV...|2017-03-20 00:25:50|sigagxKmFM
C6lgOZD...|  5.0|We've been in the...|      0|
Jimmy John's|         Phoenix|    AZ|          33|
+--------------------+-------------------+----------
----------+-----+--------------------+------+-------
-------------+--------------+-----+----------+
only showing top 20 rows
```

In [70]:

```
fastFoodDF.write.json('../SavedFiles/fastFood.json')
```

Conclusions

There were not many surprising findings in the this portions of the explratory data analysis.

Of all the restaurant reviews, about 39