Factors influencing Taxi Driver Tips

An Investigative Study by Kaushik G, Evan P, Matthew K, Cherrise M

In this investigative data driven study we take a look at how several factors influence Taxi Driver Tips. We have understood and realised that our analysis is as good as the data we get, so we take a look at taxi rides in the city of Chicago during the entire year of 2017. After a thorough analysis of the data in the columns we plan to rigourously test and explore the following questions in the data:

1. Are passengers who travel for longer periods of time in a cab tend to tip more?
2. Are there any geographic regions within the City of Chicago which tend to tip more?
3. Are there any specific companies whose drivers recieve more tips?
4. Do Taxi driver tips increase during times of the year when people feel particularly generous (Holiday Season etc.)

Part 0 (Module Imports and Computing basic statistics on columns of the dataset)

In [1]:
```
# All the modules we ever need for analysis see inline comments for more in
```

In [3]:
```
import dask  #For large scale data analysis
import dask.dataframe as dd #Provides pandas like evironment for large scal
from dask.distributed import Client #A compute cluster to paralleize Tasks
import matplotlib.pyplot as plt #The Ubiquitous Matplotlib
import seaborn as sns #Seaborn: Just because it looks way cooler
import scipy.stats as stats #Scipy: For statsitical analysis
import swifter
import geopandas as gpd #Geopandas: For Geospatial Analysis
import itertools #For getting Combinations (Pairs) in an Array etc.
import numpy as np #Accelrated Linear Algebra Computations
from IPython.display import HTML, Image #Display .png/.html file on the not
```

In [4]:
```
client = Client()
client
```

```
/Users/kaushikramganapathy/opt/anaconda3/lib/python3.7/site-packages/dist
ributed/dashboard/core.py:72: UserWarning:
Port 8787 is already in use.
Perhaps you already have a cluster running?
Hosting the diagnostics dashboard on a random port instead.
  warnings.warn("\n" + msg)
```

Out[4]:

Client
Scheduler: tcp://127.0.0.1:64135
Dashboard: http://127.0.0.1:64136/status (http://127.0.0.1:64136/status)

Cluster
Workers: 4
Cores: 4
Memory: 8.59 GB

In [5]:
```
df = dd.read_csv('2017_*.csv', dtype={'Trip Seconds': 'float64'})
```

In [5]:
```
number_of_records = len(df)
```

In [6]:
```
print('There are {} records present'.format(number_of_records))
```

```
There are 24902446 records present
```

Hence there are around **25 million records** each of which is a taxi rides during the year of 2017 in Chicago

Displaying and Selecting Appropriate columns for the Data Analysis

In [7]:
```
############INSERT EXPLANATION/TABLE AS TO WHICH COLUMNS WE CHOSE AND WHY##
df.describe().compute()
```

Out[7]:

	Trip Seconds	Trip Miles	Pickup Census Tract	Dropoff Census Tract	Pickup Community Area	Dropoff Community Area	
count	2.490100e+07	2.490197e+07	1.688284e+07	1.679711e+07	2.260698e+07	2.215315e+07	2.490
mean	8.119212e+02	3.611162e+00	1.703139e+10	1.703137e+10	2.479693e+01	2.237687e+01	1.329
std	1.112590e+03	5.825753e+00	3.464109e+05	3.352513e+05	2.019170e+01	1.804805e+01	2.860
min	0.000000e+00	0.000000e+00	1.703101e+10	1.703101e+10	1.000000e+00	1.000000e+00	0.000
25%	4.070000e+02	8.000000e-01	1.703108e+10	1.703108e+10	8.000000e+00	8.000000e+00	6.500
50%	6.600000e+02	1.740000e+00	1.703132e+10	1.703132e+10	3.200000e+01	2.800000e+01	9.250
75%	1.200000e+03	4.500000e+00	1.703184e+10	1.703184e+10	3.200000e+01	3.200000e+01	1.625
max	8.638900e+04	2.151860e+03	1.703198e+10	1.703198e+10	7.700000e+01	7.700000e+01	9.999

Part 1. Do passengers who travel for longer periods of time in a cab tend to tip more?

Computing Pearson Correlation Coefficients between Time and Tips to check for a linear relationship

In [30]:
```
time_df = df[['Trip Seconds', 'Tips']]
```

In [31]:
```
correlation = time_df.corr()
```

In [32]:
```
correlation = correlation.compute()
```

In [33]:
```
corr_coef = correlation['Tips']['Trip Seconds']
```

In [34]:
```
corr_coef
```

Out[34]:
```
0.32091693711879066
```

The correlation coefficient of 0.3209 has a moderately positive correlation. However, this includes all the data that is present including outliers.

Let us now try to remove outliers on both Tips and Trip Times and compute the correlation coefficient

On contrast to the typical definition of an ourlier being any datapoint that is $\pm\ 1.5 \times \text{IQR}$, we offer a slightly more liberal definiton of an outlier to be anything that falls within the 97th percentile of the tip. We suggest this because from past experience we realise that the most generous tip that a taxi driver is going to get would be USD 5 to USD 10, and upon checking the distribution of quantiles as shown below, we can see that the 97th percentile in tip values is 10.3, we feel that is a suitable arbitrary threhold based off of personal experience. Having more stringent cutoffs for outliers on Tips may affect the nature of out statistical analysis.

In [38]:
```
outlier_tips[0].compute()
```

Out[38]:
```
10.3
```

In [20]:
```
outlier_time = dask.array.percentile(time_df['Trip Seconds'].dropna().value
outlier_tips = dask.array.percentile(time_df['Tips'].dropna().values, q = 9
```

In [41]:
```
filtered_time_df = time_df[(time_df['Trip Seconds'] <= outlier_time[0]) & (
```

In [42]:
```
correlation_filtered = filtered_time_df.corr()
```

In [44]:
```
correlation_filtered = correlation_filtered.compute()
corr_coef = correlation_filtered['Tips']['Trip Seconds']
```

In [45]:
```
corr_coef
```

Out[45]:
```
0.4570143507428285
```

We can see that the correlation coefficient has improved to around 0.46. This signifies a stronger positive relationship between Tips and Time Duration

Finally, we compute the correlation coefficient between Tips and time duration for those trips which actually gave a non-zero tip

In [46]:
```
filtered_time_df_nonzeros = filtered_time_df[filtered_time_df['Tips'] != 0]
correlation_filtered = filtered_time_df_nonzeros.corr()
correlation_filtered = correlation_filtered.compute()
corr_coef = correlation_filtered['Tips']['Trip Seconds']
```

In [47]:
```
corr_coef
```

Out[47]:
```
0.7591330770971997
```

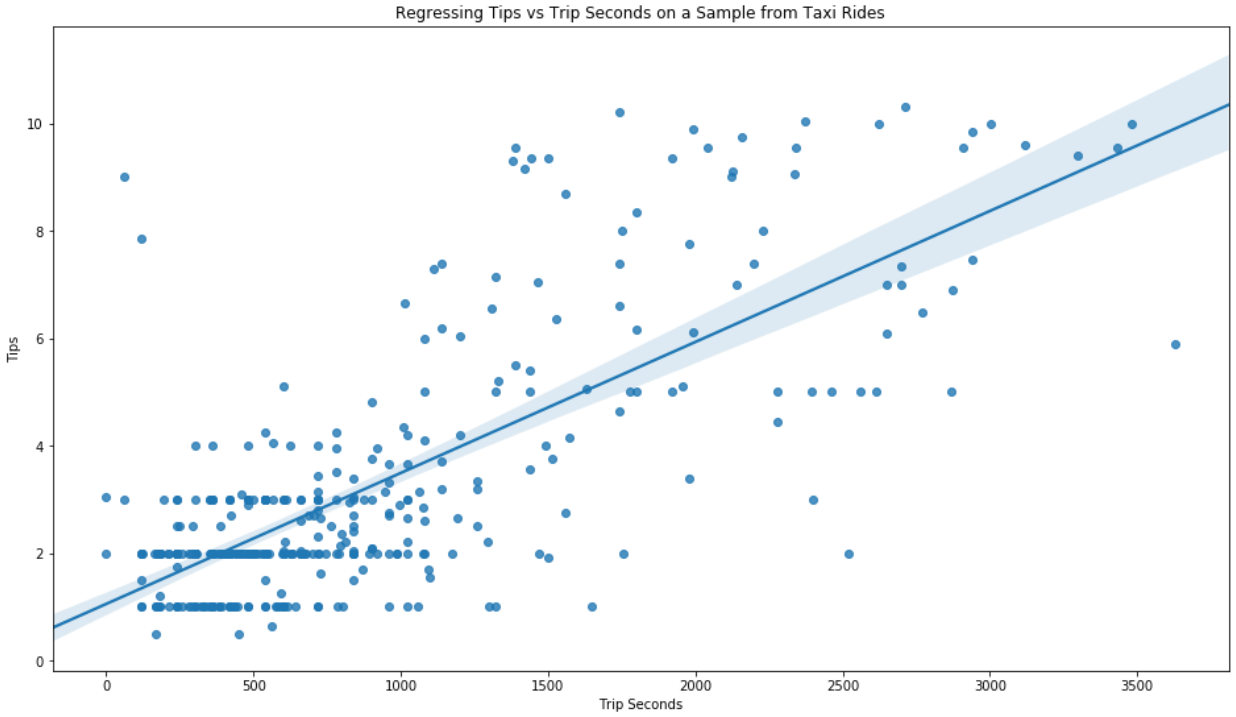
THIS IS A HUGE IMPROVEMENT!

As we can see the correlation coefficient shows that 75% of the variance in the Tips can be simply explained with the help of the Trip Seconds data. Also, higher degrees of correlation means for those passengers who do indeed tip (barring outliers) there exists a greater linear association between Tips recieved and the Amount of time a passenger spends inside a taxi.

To further substantiate this finding, we visualize and plot a linear regression line with a 95 percentile confidence interval. Since there are around 25 million records, we select a random sample of 500 rides to show the general trend of the line regression line

In [52]:
```
thousand_random = filtered_time_df_nonzeros.sample(frac = 4.015669786012185
thousand_random = thousand_random.compute()
```

In [53]:
```
sns.regplot(x = 'Trip Seconds', y = 'Tips', data = thousand_random)
plt.title('Regressing Tips vs Trip Seconds on a Sample from Taxi Rides')
plt.show()
```



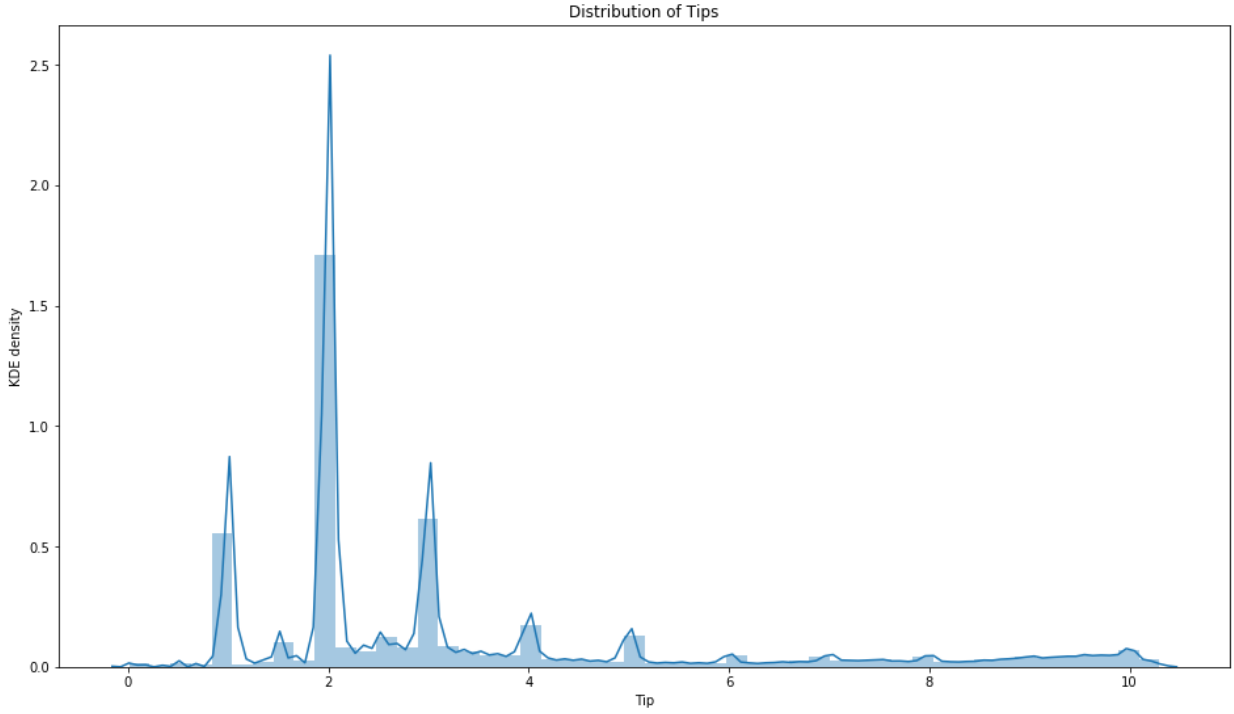
The following graph shows us that the width of the confidence interval is not too high, showing that the linear trend is infact fairly significcant

Seeing this, we now try to perform statistical validation of our strong linear relationship

Let us check if our Tips (y-values) are normally distributed

VISUAL TEST

In [48]:
```
y_values = filtered_time_df_nonzeros['Tips'].dropna().to_dask_array()
sns.distplot(y_values, kde = True)
plt.title('Distribution of Tips')
plt.xlabel('Tip')
plt.ylabel('KDE density')
plt.show()
```



From the historgram we can see that there is a huge peak centered at 2 USD. However, we see smaller peaks at 1 USD and 3 USD. However, we see that the distribution is heavily right skewed with smaller and smaller peaks corresponding to the various other integer value of tips. The distribution is hence *unimodal* with a heavy right skew. This suggests that the distribution is not normal, which is to be expected since intuitively speaking, tip values are supposed to have a right-skew accounting for those people who give the uncommonly high tip!

Note that the KDE density can be used as a proxy for frequency.

STATISTICAL TEST

We perform the following test wil the following null and alternative hypothesis using a significance level (α) of 0.05:

H_0 : The distribution of tip values is normal
H_A : The distribution of tip values is not normal

In [145]:
```
results = stats.normaltest(y_values)
```

In [146]:
```
results
```

Out[146]:
```
NormaltestResult(statistic=2441504.3506951965, pvalue=0.0)
```

The p-value clearly indicates that we can reject the null hypothesis. Hence the distribution of tips is indeed concluded to be statistically not-normal.

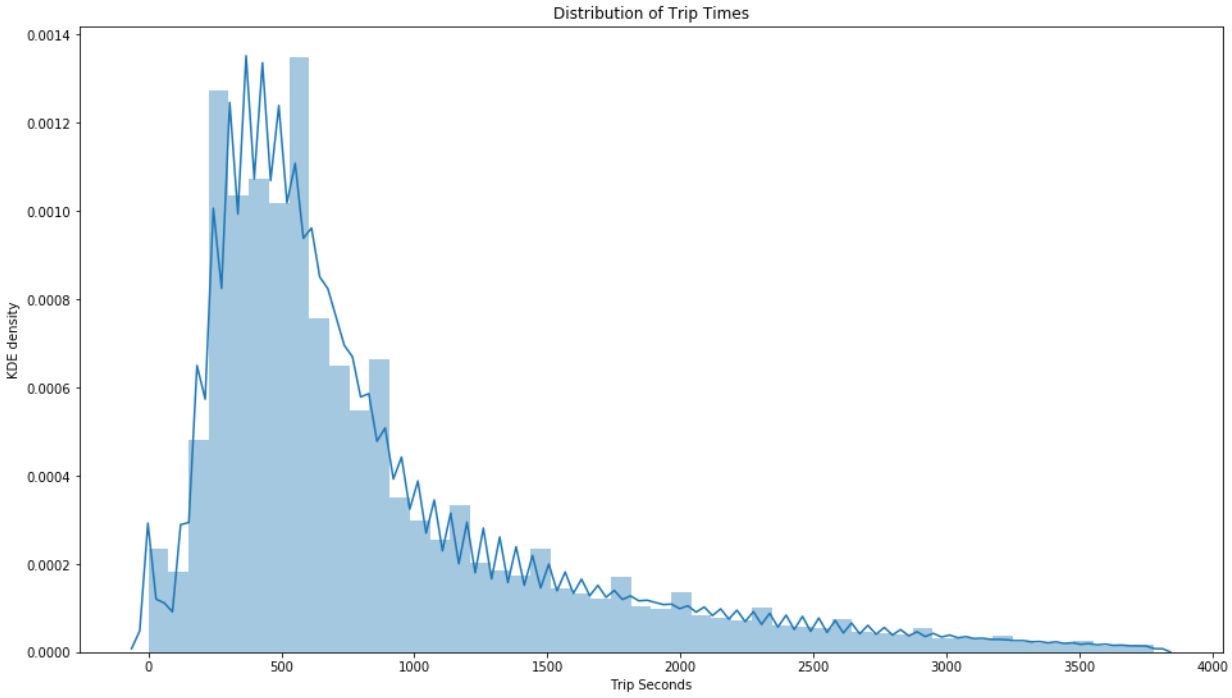
However, our goal is to estabilish that there is a statisically significant linear relationship between tips and time duration.

Hence, we use the simple fact that $\rho(x, y) = \rho(y, x)$ where ρ is the correlation coefficient between two variables x and y namely tips and time duration in cab.

Let us check if time duration (the previous x-variable) is normally distributed instead.

VISUAL TEST

In [50]:
```
y_values = filtered_time_df_nonzeros['Trip Seconds'].dropna().to_dask_array
sns.distplot(y_values, kde = True)
plt.title('Distribution of Trip Times')
plt.xlabel('Trip Seconds')
plt.ylabel('KDE density')
plt.show()
```



From the historgram we can see that the distribution has a geavt right skew again, however, it is also unimodal but has a hevy right skew. Visually it appears to be not normal.

Note that the KDE density can be used as a proxy for frequency.

STATISTICAL TEST

We perform the following test wil the following null and alternative hypothesis using a significance level (α) of 0.05:

H_0 : The distribution of time in cabs are normal
H_A : The distribution of time in cabs are not normal

In [151]:
```
results = stats.normaltest(y_values)
```

In [152]:
```
results
```

Out[152]:
```
NormaltestResult(statistic=3021937.71738119, pvalue=0.0)
```

The p-value clearly indicates that we can reject the null hypothesis. Hence the distribution of trip durations is indeed concluded to be statistically not-normal.

This shows that either variable is not normally distributed, thus hindering our ability to validate our strong linear correlation between Tips and Trip Duration.

Despite failing one of the requirements, let us try to see what the lienar regression report says:

We perform the following test wil the following null and alternative hypothesis using a significance level (α) of 0.05:

H_0 : The slope of regression line b_0[1] is 0
H_A : The slope of regression line b_0[1] is not 0

[1]. When normalized the standardarized the slope of regressiion line is the correlation coefficient between the dependent and independent variable (ρ).

In [14]:
```
filtered_time_df_nonzeros_dropna = filtered_time_df_nonzeros.dropna()
slope, intercept, r_value, p_value, std_err = stats.linregress(filtered_tim
```

In [15]:
```
print('correlation coefficient is: {}'.format(r_value), 'p-value is: {}'.fo
```

```
correlation coefficient is: 0.7591330770972028 p-value is: 0.0
```

The p-value clearly indicates that we can reject the null hypothesis. Hence the fact that there exists a statsitically significant strong linear relationship between Tips and Trip Durations is True.

Since we fail the assumptions for calculation of linear regression, and since we have so much of data, let us attempt to use the non-parametric Spearman Rank Correlation to estimate if there is a monotonically increasing relationship between cab travel times and tips

Assumptions for Spearman Rank Correlation

1. Variables are either ordinal, ratio or interval ✓

STATISTICAL TEST

We perform the following test wil the following null and alternative hypothesis using a significance level (α) of 0.05:

H_0 : There is no monotonic relationship between Trip Times and Tips
H_A : There is a monotonic relationship between Trip Times and Tips

In [16]:
```
results = stats.spearmanr(filtered_time_df_nonzeros_dropna['Trip Seconds'],
```

In [17]:
```
results
```

Out[17]:
```
SpearmanrResult(correlation=0.6203530484450192, pvalue=0.0)
```

The moderately high positive value of the statsitic suggests that the relationship is strongly monotonically increasing. This means that greater times spend in cabs results in greater tips being recieved by the drivers for those passengers who do tip. The low p value of suggests that the findings statistically favor the rejection of the null hypothesis suggesting that the monotically increasing relationship is statistically significant

Results

- On the surface there exists a strong linear correlation between Tips and Time Duration for those rides falling within the 97th percentile of taxi rides that provide a tip. This would indicate that greater the amount of time spent by a rider in a taxi, greater the tip they would give for these rides.

- However, the distribution of tips and/or time durations are not normal, thus breaking one of the primary conditions required for the regression slope test which can statitically validate the significance of this strong linear relationship.

- Thus, despite obtaining statistically significant results in the slope-significance test, our linear regression results are **invalid** because of the violation of the non-normality condition.

- However, with the help of non parametric statsitic based tests such as spearman rank we were able to find a monotonically increasing relationship between Tips and Travel Times in Cabs. This ultimately translates to the fact that there are statsitically significant results which suggest the relationship "greater the amount of time a persons spends in a cab, greater the tp they would provide, provided they are the average rider and do tip."

  Hence we have sucessfully shown that there is a strong monotically increasing relationship between Travel Times in Taxis and Tips.

Part 2. Do drivers in certain companies recieve more tips?

We select the company column which is contains the names of the company to which the driver belongs for each trip.

In [9]:
```
company_df = df[['Tips', 'Company']]
```

We see that the number of trips for which the company is null is in fact zero

In [10]:
```
numnulls = company_df['Company'].isna().sum().compute()
numnulls
```

Out[10]: 0

In [11]:
```
unique_companies = company_df['Company'].unique().compute()
```

We see that the there are 86 companies!

In [12]:
```
len(unique_companies)
```

Out[12]: 86

Let us select only the "major-players" in Chicago

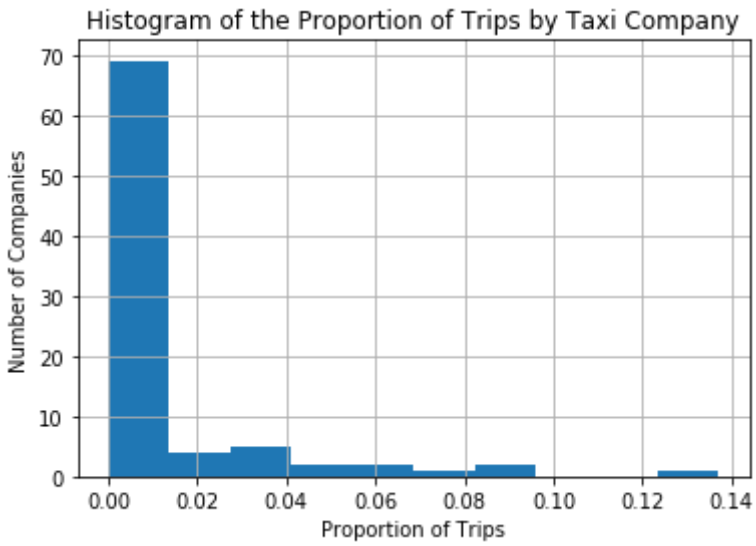
To do this we list the proportion of rides with non-null tips for each company

In [13]:
```
company_df_nonnulltips = company_df.dropna()
```

In [14]:
```
proportion_of_trips = (company_df_nonnulltips['Company'].value_counts())/le
```

In [15]:
```
proportion_of_trips = proportion_of_trips.compute()
```

In [50]:
```
proportion_of_trips.hist()
plt.title('Histogram of the Proportion of Trips by Taxi Company')
plt.ylabel('Number of Companies')
plt.xlabel('Proportion of Trips')
plt.show()
```



We see that most (~70 out of 86) of the companies have less than 2% of the rides in the city. Find some descriptive statistics which will help us identify the major players.

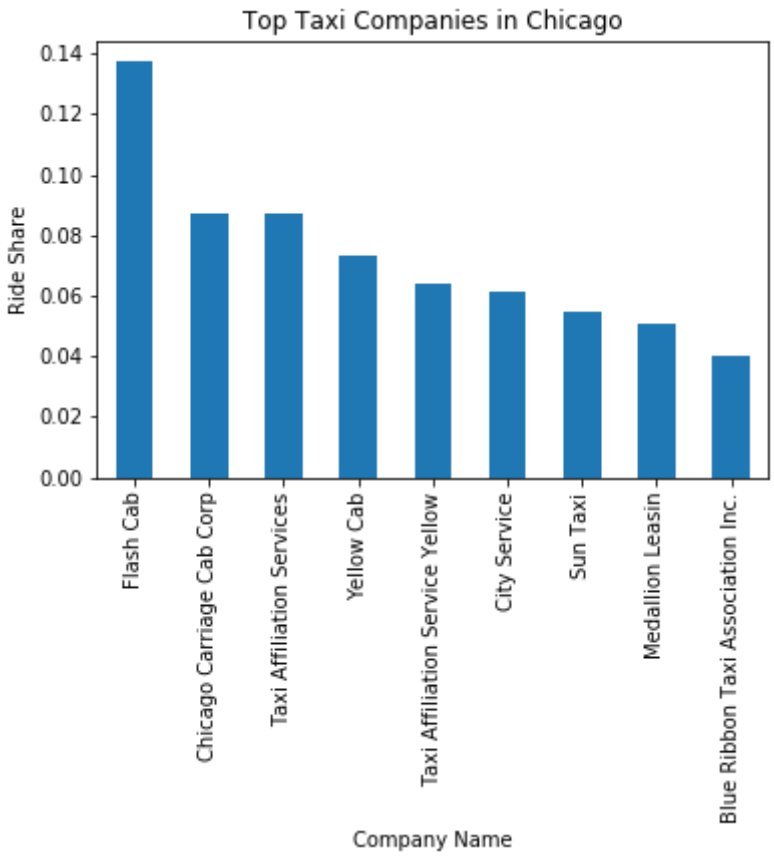
In [72]:
```
proportion_of_trips.describe(percentiles=[0.25, 0.5, 0.75, 0.9, 0.95, 0.98])
```

Out[72]:
```
count    8.600000e+01
mean     1.162791e-02
std      2.411193e-02
min      2.409458e-07
25%      9.597676e-05
50%      2.270513e-04
75%      1.066553e-02
90%      3.809464e-02
95%      6.317934e-02
98%      8.716755e-02
max      1.371148e-01
Name: Company, dtype: float64
```

From the descriptive statistics, let us define a major player as one which as greater than the 90th percentile of shares in taxi rides in the city of chicago.

In [16]:
```
ninety_percentile = proportion_of_trips[proportion_of_trips > 3.809464e-02]
```

In [83]:
```
ninety_percentile.plot(kind = 'bar')
plt.title('Top Taxi Companies in Chicago')
plt.xlabel('Company Name')
plt.ylabel('Ride Share')
plt.show()
```



Let us compute the Tips for each of the trips offered by these companies which are not-null and falls within the 97th percentile of Tips to eliminate the outliers.

In [21]:
```
top_companies_tips = company_df[(company_df['Tips'] <= outlier_tips[0]) & (
```

In [22]:
```
categories = dict(zip(list(ninety_percentile.index), [i+1 for i in range(le
```

In [23]:
```
top_companies_tips['Company Category'] = top_companies_tips['Company'].appl
```

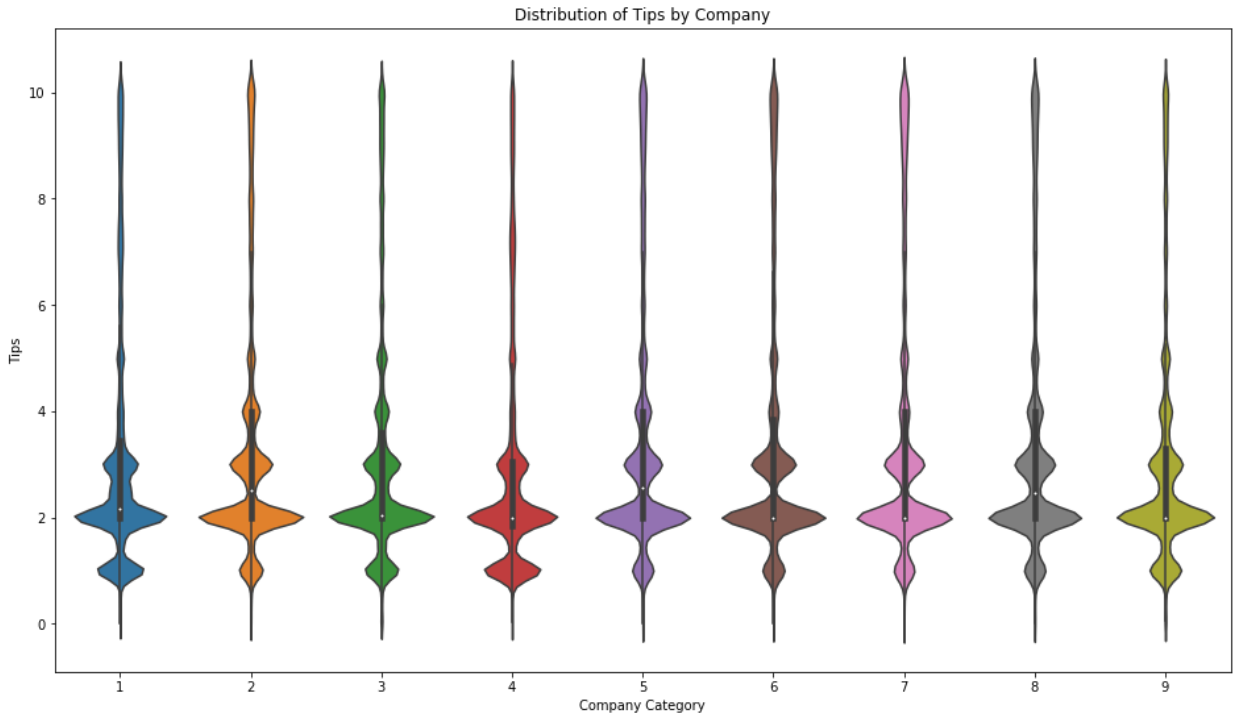
In [29]:
```
import pandas as pd
pd.DataFrame.from_dict(categories, orient = 'index', columns = ['Category N
```

Out[29]:

	Category Number
Flash Cab	1
Chicago Carriage Cab Corp	2
Taxi Affiliation Services	3
Yellow Cab	4
Taxi Affiliation Service Yellow	5
City Service	6
Sun Taxi	7
Medallion Leasin	8
Blue Ribbon Taxi Association Inc.	9

Let us plot a violin plot to see if there are any differences in the distribution of Tips across companies

In [25]:
```
plt.rcParams["figure.figsize"] = [16,9]
sns.violinplot(x="Company Category", y="Tips", data=top_companies_tips.comp
plt.title('Distribution of Tips by Company')
plt.savefig('CATPLOT.png', dpi = 400)
```



We see that the distribution of tips are extremely similar across companies. There are peaks corresponding to USD1, USD2, and USD3. However, we see that companies 2, 5, 8, 9 have slight peaks corresponding to USD 4 or more.

However, the boxplots within these distributions tell a completely different story. We can see quite clearly that the means of the tips across the various companies are not the same. We can see that company Chicago Carriage Cab Corp, Taxi Affiliation Service Yellow, Medallion Leasin have a mean tip close to USD 3, whereas companies Flash Cab, Taxi Affiliation Services, and Yellow Cab have a mean closer to USD 2. Lastly companies City Service, Sun Taxi, and Blue Ribbon Taxi Association Inc. have a mean tip that is lower than USD2.

Since neither of these distributions are normal, we use a non-parametric test (Kruskal Wallis Test) to test whether there are differences in tips recieved across companies. Since each of these companies have greater than 5 rides, we are able to satisfy the assumptions of this test.

We perform the following test wil the following null and alternative hypothesis using a significance level (α) of 0.05:

H_0 : The median of tips recieved by drivers across companies is the same
H_A : The median of tips recieved by drivers across companies not the same

In [152]:
```
for cat in categories.values():
    globals()['arr_{}'.format(cat)] = top_companies_tips[top_companies_tips
```

In [153]:
```
stats.kruskal(arr_1, arr_2, arr_3, arr_4, arr_5, arr_6, arr_7, arr_8, arr_9
```

Out[153]:
```
KruskalResult(statistic=47710.711086033174, pvalue=0.0)
```

This suggests that the median of tips across the companies is in-fact not the same. This observation is once again clearly shown in the boxplots in the fugure above. The differences in Tips recieved can be most likely due to differences in business models across the companies. However such causal relationships are merely speculative and further analysis is necessary to discover the causality from the association.

Results

- We can see quite clearly the existance of differences in the tips recieved by drivers across companies, which has been validated with the help of statistical and visual tests.
- However, the underlying force behind such differences is difficuilt to understand without further analysis of each company's business model.
- In effect, we have shown that there are dependencies between Tips recieved by Drivers and the Company that they work for, on typical rides where passengers do tip.

Part 3. Geographical differences in Taxi Driver Tips

In this example we are forced to use the single core Pandas/ Geopandas which is inherently slower than dask. So in order to faciliate feasible computation, we choose to work with a representative sample of the dataset. For sampling strategy we choose to go with Proportionate allocation based on a Simple Random Sample. To make our results more robust we simulate and run our analysis over 4 successive iterations. Finally, we choose to deal with ethical considerations by dropping the

coordinates themselves, and choose the work with Census tracts, which are far more granular. Further, with the dataset's description in mind, we choose to drop all locations where we have no information on the census tract.

In [18]:
```
df_geographic = df[['Tips', 'Dropoff Census Tract']]
df_geographic = df_geographic.dropna(subset = ['Dropoff Census Tract'])
```

In [19]:
```
print('There are {} records left'.format(len(df_geographic)))
```

```
There are 16797109 records left
```

So we see that we still have a large number of records (~17 Million) left! Let us now filter the Tips which are outliers or are 0 USD.

In [21]:
```
outlier_tips = dask.array.percentile(df['Tips'].dropna().values, q = 97)
```

In [22]:
```
filtered_geo_df = df_geographic[(df_geographic['Tips'] <= outlier_tips[0])
```

In [23]:
```
print('There are {} records left'.format(len(filtered_geo_df)))
```

```
There are 7310603 records left
```

Even after this filtering, we notice that we still have a large number of records (7 Million) left for analysis. In order to simplify our analysis we choose a random subset of 12000 taxi rides (an arbitrarily high threshold before our computers started taking too long). We realise that the choice to only use a random subset will affect the power of the conclusions we draw. However, due to the fact that Dask does not natively support Geopandas, so we are left with lttle choice computationally speaking.

Using a random sample of ~12,000 taxi rides, we do an inner join with Census Tract Information retrieved from [the United States Census Bureau (https://www.census.gov/acs/www/data/data-tables-and-tools/data-profiles/2017/)](https://www.census.gov/acs/www/data/data-tables-and-tools/data-profiles/2017/) for the State of Illinois for 2017. This provides us with information of the various polygons (geographical areas) which consitute each census tract.

In [144]:
```
df_geographic = filtered_geo_df[['Tips', 'Dropoff Census Tract']]
```

In [145]:
```
sample = df_geographic.sample(frac = 12000/7310603)
```

In [146]:
```
sample_df = sample.compute()
```

Reading the the Census Tracts for the State of Illinois from the U.S Census Bureau

In [ ]:
```
shapefile_df = gpd.read_file('./shapefiles/cb_2017_17_tract_500k.shp')
```

In [147]:
```
shapefile_df['GEOID'] = shapefile_df['GEOID'].astype(float)
```

In [148]:
```
geo_df_merged = sample_df.merge(shapefile_df, how = 'inner', left_on='Dropo
```

Merging Entries and converting it to a Geopandas Geo Data Frame

In [158]:
```
geo_df_merged_shp = geo_df_merged[['Tips', 'GEOID', 'geometry']]
```

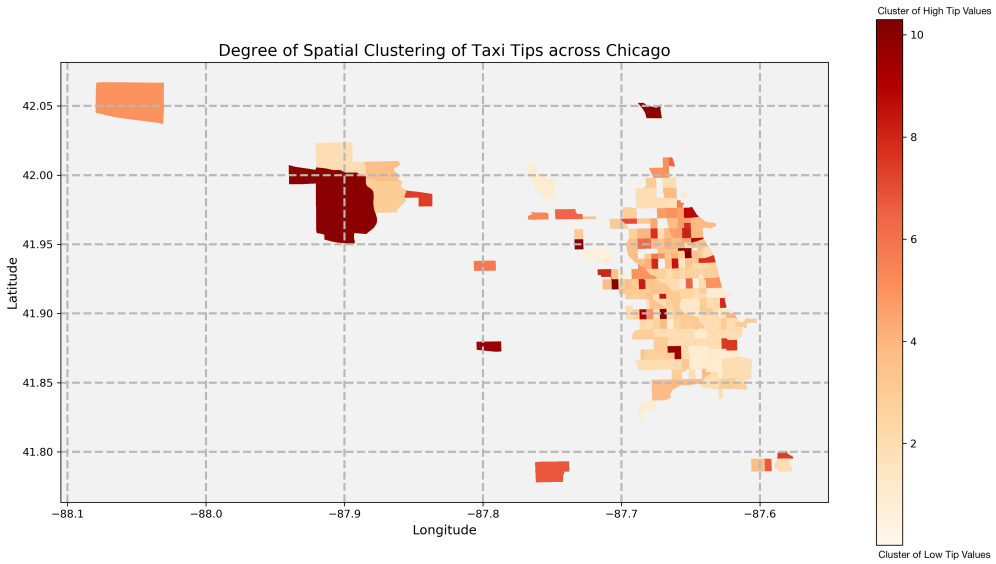
In [159]:
```
geo_df_merged_shp = gpd.GeoDataFrame(geo_df_merged_shp, crs=shapefile_df.cr
```

Plotting everything out using Matplotlib

In [240]:
```
plt.rcParams["figure.figsize"] = [16,9]
ax = plt.gca()
ax.grid(color='#b8b8b8', linestyle='--', linewidth=2, zorder=0)
ax.set_facecolor("#f2f2f2")
geo_df_merged_shp.plot(column = 'Tips', cmap='OrRd', legend=True, ax = ax)
plt.title('Degree of Spatial Clustering of Taxi Tips across Chicago', fonts
plt.xlabel('Longitude', fontsize=12)
plt.ylabel('Latitude',  fontsize=12)
plt.savefig('plotted.png', dpi = 300)
plt.close()
```

In [244]:
```
### We have manually added the anotation on the scale to make things a bit
Image('plotted.png')
```

Out[244]:



We can see that this map does not look anything like the city of Chicago! However, we do see certain areas having cluters of high tips and certain areas having low tips. This brings the need to perform/test for spatial autocorrelation.

Spatial Autocorrelation [1 (https://www.sciencedirect.com/topics/computer-science/spatial-autocorrelation)]

Spatial autocorrelation is the term used to describe the presence of systematic spatial variation in a variable. Positive spatial autocorrelation is the tendency for sites or areas that are close to one another to have similar values of the variable (i.e., both high or both low). Negative spatial autocorrelation is the tendency for adjacent values to be dissimilar (high values next to low values). The presence of spatial autocorrelation in map data has often been taken as indicative that there is something of interest in the map that calls for further investigation in order to understand the reasons behind the observed spatial variation.

In order to test for overall spatial autocorrelation in this graph we use a statistic known as the Global Moran's I coefficient. The interpretation of the value is much akin to a correlation coefficient. The Moran's I coefficient can range between -1 and 1. where 1 indicates completely positive spatial autocorrelation and -1 indicates complete negative spatial autocorrelation. A value of 0 indicates spatial randomness. We compute the Moran's-I using the following formula:

The spatial matrix W has been chosen to be computed thorugh an Inverse Distance Based metric where points further away from a chosen region in space are weighted more heavily when compared to points nearby.

We attempted to compute the Gobal Moran's I using the library PySAL. However, for some reason, the latest version of the library failed to install on our computer. To mitigate this, we chose a free trial of the Geospatial Software ArcGIS Pro. ArcGIS Pro is an industry-standard alternative to PySAL which offers comprehensive spatial statsitics through an in-house modified version of Python known as ArcPy, in addition to ArcMap which is the equivalent of PySAL. The code used to compute the spatial analysis was actually interfaced using geopandas! Geopandas allowed us to create a shapefile from the dataset, which we could then load into ArcPy to write Python Scripts to perform spatial analysis. We are attaching any/all of the python code we wrote within the code cells of this notebook.

We found out that the only way to transfer our data now would be to convert our Geo-DataFrame to a shapefile

In [172]:
```
geo_df_merged_shp.to_file('tipsfil.shp')
```

Now, we can use this to work with Python within the ARCGIS Pro console. First we see compute the global spatial autocorrelation that exists for the tips. Note that our shapefile is 'tipsfil'. The second argument refers to which column of the shapefile we want to test for autocorrelation (which is Tips), we pass in the GENERATE_REPORT to get a visualization of the spatial correlation statistic. INVERSE DISTANCE refers to the method to compute the weights matrix. Alternatives included Queen's Distance, Rook's Distance etc. EUCLIDEAN DISTANCE is a parameter for distance threshold which is automatically set by Arcmap

Code

```
 python
arcpy.stats.SpatialAutocorrelation("tipsfil", "Tips", "GENERATE_REPORT",
```

```
"INVERSE_DISTANCE", "EUCLIDEAN_DISTANCE", "NONE", None, None)
```

From the report we can clearly see the presence of a autocorrelation between dropoff location and taxi driver tips. Additionally we can see that the Moran's Global I statistic is weakly positive which suggest that The spatial distribution of high Tips and/or low Tips in the dataset is more spatially clustered than would be expected if underlying spatial processes were random albeit weakly. This suggests that There is a dependence between Dropoff Location and the Tips Recieved for drivers that do recieve a tip for the ride.

Having Quantified Location dependence of Tips, let us figure out are any particular locations which are hotspots for High Tips!

We compute this using the Getis Ord G_i^* Statistic. According to the documentation of ArcGIS Pro, the HotSpots function in ArcMap allows one to detect statististically significant clusters of high values of tips (hot-spots) and low values of tips (cold-spots). The formula of the G_i^* statistic is provided below.

Based on the value we have color hotspots are "Red" and coldspots as "Blue". Further, the intensity of the color is varied with the level of statistical significance of the respective value.

Code

In [51]:
```
import arcpy
arcpy.stats.HotSpots("tipsfil", "Tips", r"C:\Users\gani1\OneDrive\Documents
```

```
True
```

Output

Airports a center for high tips?

We have chosen the U.S Labelled based map in ARCGIS Pro. Using this we use the to_featurelayer() function in ArcPy to overlay the results on the basemap. Using this, and the code above, we can see that the entire City of Chicago is a cold-spot for Tips. We attribute this to the reason that taxi rides within the city limits do not cost as much, and hence do not warrant a huge tip. However, as we can clearly see, Taxi Rides to the outskirts of the City typically end up being Hotspots (Which Agrees with our Conclusion from Part-1 that greater times spend in the cab, results in great tips) Finally, we see that both the Airports serving Chicago (MDW, ORD) are hotspots for Tips! This is probably because of the airport being quite a distance away from the city's center. However, an association does not mean causation and further investigation is necessary to understand why Airports in Chicago attract higher Tips.

Results

- Through the Global Moran's I, we have discovered statistically significant spatial dependencies in the Driver Tips in Taxi Cab Rides.

- Additionally, through our Hotspot analysis we disovered that areas on the outskirts of Chicago have significant clusters of high Tips. Additionally, Airports are also areas with significant clusters of high tips.

- Hence, we have shown that there are Spatial Dependencies in Taxi Driver Tips in the city of Chicago.

Part 4. Are there any Seasonal (Temporal) Differences to Taxi Tips?

In the final section we investigate whether there are any significant differences in Taxi Tips Recieved during the Holiday Season (Thanksgiving/Christmas) compared to other portions of the year. Our Hypothesis is that if people are more generous during the holiday season, we would expect greater tips on a typical ride!

We first isolate the Trip Start Timestamp and the Tips Column. Per usual, we then retain all the tips which are within the 97th percentile of tip values. In addition, we also dropp all the tips which are nil (0 USD).

In [256]:
```
holiday_df = df[['Trip Start Timestamp', 'Tips']]
```

In [257]:
```
outlier_tips = dask.array.percentile(df['Tips'].dropna().values, q = 97)
```

In [262]:
```
filtered_holiday_df = holiday_df[(holiday_df['Tips'] <= outlier_tips[0]) &
filtered_holiday_df = filtered_holiday_df.dropna()
```

In [284]:
```
tg_df = filtered_holiday_df.copy()
xmas_df = filtered_holiday_df.copy()
non_hol_df = filtered_holiday_df.copy()
```

We write user defined functions to filter the dates with respect to the dates for Christmas and Thanksgiving for the year of 2017. Thankgiving was 11/23/2017, So we start filtering dates from 11/18/17 through 11/27/17 to account for people travelling to Chicago for Thanksgiving. Similarly, We filter dates from 12/20/17 to 12/31/17 for Christmas travel. Finally we also write a function to get the dates which do not fall into any of these brackets (i.e. non-christmas, non-thanksgiving time-frames). Note that there are plenty of holidays celebrated in the U.S, and it is our subjective choice to choose the two biggest holiday seasons in the U.S.

In [ ]:
```
def date_check_tg(x):
    x = x[:10]
    if x[:2] != "11":
        return np.nan
    elif 18 <= int(x[3:5]) <= 27:
        return x
    else:
        return np.nan


def non_tg_non_xmas(x):
    x = x[:10]
    if x[:2] != "11" or x[:2] != "12":
        return x
    elif 18 <= int(x[3:5]) <= 27:
        if x[:2] != '11':
            return x
    elif 20 <= int(x[3:5]) <= 31:
        if x[:2] != '12':
            return x
    else:
        return np.nan

def date_check_xmas(x):
    x = x[:10]
    if x[:2] != "12":
        return np.nan
    elif 20 <= int(x[3:5]) <= 31:
        return x
    else:
        return np.nan

tg_df["Trip Start Timestamp"] = tg_df["Trip Start Timestamp"].apply(date_ch
non_hol_df["Trip Start Timestamp"] = non_hol_df["Trip Start Timestamp"].app
xmas_df["Trip Start Timestamp"] = xmas_df["Trip Start Timestamp"].apply(dat
```

We drop nulls, since that is how we return non-conforming dates in our user defined function

In [286]:
```
tg_df = tg_df.dropna()
non_hol_df = non_hol_df.dropna()
xmas_df = xmas_df.dropna()
```

In order to facilitate the drawing of a seaborn violin plot, we need a categorical variable. So we manually create a feature known as season which represents which holiday season the date of the ride was a part of. The variable season takes on 3 values:

1. xmas: For travels within the Christmas timeframe.
2. thanksgiving: For travels within the Thanksgiving timeframe.
3. non-holiday: All other travel days other than Christmas or Thanksgiving.

We then append these datasets to create on large dataset known as all_data_with_season and all_data_with_thanksgiving. This contains the categorical variable season!

In [292]:
```
xmas_df['season'] = 'xmas'
non_hol_df['season'] = 'non-holiday'
tg_df['season'] = 'thankgiving'
```

In [289]:
```
all_data_with_season = dd.concat([xmas_df, non_hol_df])
```

In [293]:
```
all_data_with_thanksgiving = dd.concat([tg_df, non_hol_df])
```

Note that we could have easily concatanated xmas_df, non_hol_df and tg_df into a single dataframe but we choose to not. This is because in the following cell, we plan to perform the kruskal wallis test. The test allows us to is essentially a non-parametric version of ANOVA. In short the kruskal wallis test will test the following hypothesis:

H_0 : The distribution of tips during the Thanksgiving and Non-Holiday Seasons is the same
H_A : The distribution of tips recieved by drivers during Thanksgiving and Non-Holiday Seasons companies not the same

In [275]:
```
stats.kruskal(tg_df.Tips, non_hol_df.Tips)
```

Out[275]:
```
KruskalResult(statistic=0.1477533527744232, pvalue=0.7006916468591153)
```

Using our earlier defined cut-off of α = 0.05, We see that we can accept the null Hypothesis. Hence, ths statistical test suggests that there are no significant differences between Thanksgiving Tips and Non-Holiday Tips. This is also clearly reflected in the visual violin-plot generated below!

In [ ]:
```
sns.violinplot(x = 'season', y = 'Tips', data = all_data_with_thanksgiving.
plt.savefig('thanksgiving.png')
plt.close()
```

Similarly we do the same anaysis for Christmas and other periods of the year

H_0 : The median tips during the Christmas and Non-Holiday Seasons is the same
H_A : The median tips recieved by drivers during Christmas and Non-Holiday Seasons companies not the same

Using our earlier defined cut-off of α = 0.05, We see that we can accept the alternative Hypothesis. Hence, ths statistical test suggests that there are significant differences between Christmas Tips and Non-Holiday Tips. While the distribution of these Tips are pretty consistent as shown in the violin plots, we see that the inner side-by-side boxplots between the two variables shows that the medians are indeed different. However, we see quite clearly that the 75th percentile of tips in christmas is lower than other periods of the year. This would in turn affect the median value too!

In [280]:
```
stats.kruskal(xmas_df.Tips, non_hol_df.Tips)
```

Out[280]:
```
KruskalResult(statistic=1324.1087127486671, pvalue=6.517742122275904e-29
0)
```

In [ ]:
```
sns.violinplot(x = 'season', y = 'Tips', data = all_data_with_season.comput
plt.savefig('xmas.png')
plt.close()
```

Results

- Through our analysis we have shown that there are statistically significant differences in the median tip recieved by drivers who do recieve tips during Christmas, but not during Thanksgiving which seems interesting.

- Furthermore, we see from the visual test above that the median tip recieved and other percentiles such as the 75th percentile are actually lower for Christmas (implying a lower tip recieved) compared to other times of the year.

- The nature of such dependency should be a subject for further exploration. Potential ways to analyse and explore this relationship would be to test the differences across trips which ended in various sections of the city. Maybe Trips that ended at the Airport during Christmas have a greater tip when compared to typical airport tips during the rest of the year.

  Hence we have sucessfully shown that there is a relationship between which Taxi Company a driver works for and the tips recieved.

Conclusions and Discussion

In [ ]:
```
```

Ethics & Privacy:

According to the City of Chicago's data collection team, they prioritize personal privacy when developing the data set. They have done this through specially developed de-identification and aggregation techniques. This technique involves: aggregation by time, in which all trips were rounded to the nearest 15 minute interval. Using this, the data collection team exploits the sheer size of the city of Chicago, which is spread over 800 census tracts each of which ranging between 89,000 sq. ft and 84 million sq. ft. As a result of this, coupled with the time-based de-identification, it is impossible to know the precise time and place the trip occurred beyond a 15-minute window and an 89,000 square foot area. As the dataset does provide the approximate location of a trip, another layer of protection was added to avoid linking individuals' trip location data to their identities. These techniques and explanations shown below are directly obtained from the city of Chicago's Data Privacy Site referenced below:

- If the above method resulted in any aggregation having two or fewer unique trips in the same census tract and 15-minute time window, the geographical space published was widened to the Community Area level for both ends of that trip.
- Even if one acquires separate data about a trip location and trip time along with identifying information about a passenger/rider, the presence of at least three matching trips would inhibit isolating a specific trip's census tracts in this dataset.
- As a result of this protection, approximately a third of census tracts that would otherwise be shown in the initial dataset are blank. (Others are blank because of missing data or falling outside Chicago.) By removing the census tract from these particular records, we limit the location information that could be reidentified by providing only the Community Area in which the trip started and ended. On average, a Community Area covers 3 square miles of the City.

Source: City of Chicago: Data Privacy (http://dev.cityofchicago.org/open%20data/data%20portal/2019/04/12/tnp-taxi-privacy.htm)

In addition to the above facts, we performed a quick analysis on the columns of the dataset which was obtained. We saw, there was no information about the rider published; and that the only columns related to the rider in the data set are the fare, the locations, and the start/stop times of the trip. There is no data collection such as their name, address, phone number, email, or any other information which may invade their privacy. The columns which we chose to analyse namely Tips, Company, and Trip Start Time are indeed only related to taxi rides and adhere to the safe harbor methods discussed in class.

Our efforts to ensure Privacy and Ethics in Analysis

Despite the City of Chicago's efforts in anonymizing the data, securing personal privacy, and aggregating by geographical location and time, there is still an obvious concern about the exact locations (lat, long) of pick-up and drop-off provided. While it cannot be linked back to a specific individual, patterns could easily be found using this data set of individuals, which is an invasion of privacy. An example of this would be that a frequent drop-off location could be extrapolated to a specific individual - a clear invasion of privacy. This also violates the safe harbor "geographical subdivisions" method, indicating that it should not be published publicly. Also, there are trip and taxi IDs which are disclosed in the data-set. While these are anonymized, they still relate to a unique individual which could be an invasion of privacy as it is unique.

We handled this issue by ignoring the latitude and longitude coordinates (dropping them from the data set), and instead focused our geographical analysis on the census tracts disclosed by the data set. These census tracts are far more general than the coordinates provided,since they are by definition aggregations rather than specific locations and are simply abstract regions created by the City of Chicago.

The drop-off census tracts which did not guarantee anonymity (indicated by NaNs in the dataset), were dropped from the geospatial portion of our analysis. Thus, there is no zip-code or specific coordinates involved in the analysis, just a general region (akin to a neighbourhood) associated with the drop-off location.

Similarly, in a light to respect privacy/ethical laws, we simply chose to drop the taxi ID and trip ID columns in our analysis, following the insightful discussion in class where one could potentially "hack" the anonymization (hashing) process.

Other than this, the data set is open for public use so there are no issues with us utilizing it for our project. There should not be any bias within the data set either, as it is collected by an agency unrelated to the cab services. The files are simply for documentation purposes and for analysis like this.