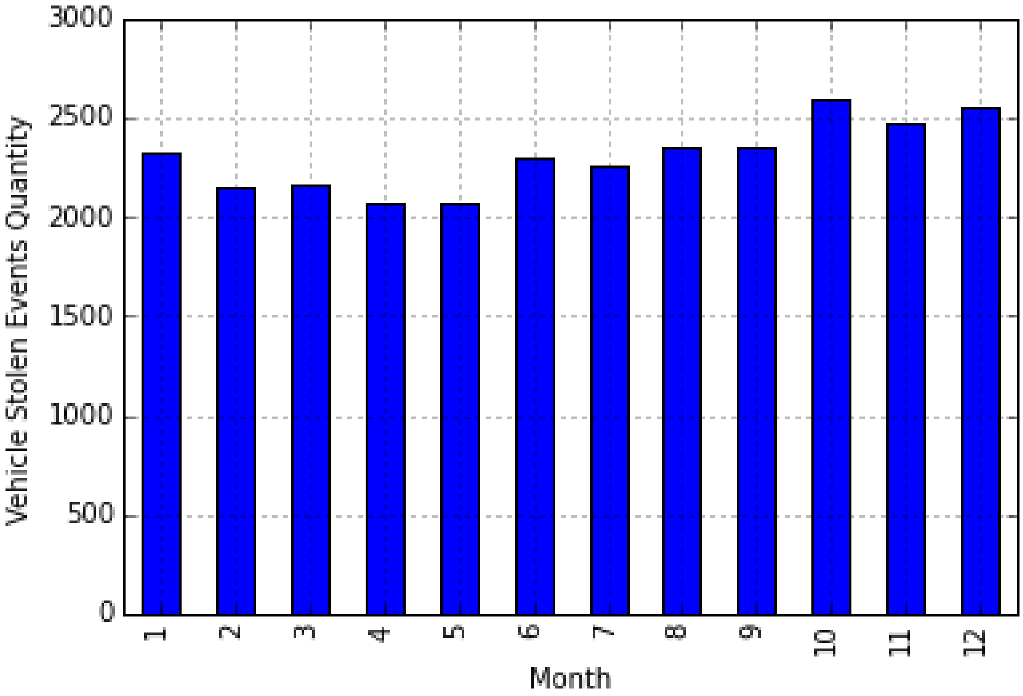
1. Executive Summary

[Write-up for Non-Technical Customer]

Weather, parking patterns and drivers' habits (which may all vary with seasons) can conceivably affect how criminals steal cars. Understanding seasonal effects on stolen vehicle activity can help peace officers and the public. Police departments can potentially improve availability and workload planning for staff focused on stolen vehicle activity. Officials can potentially warn vehicle owners about specific seasonal trends. Indeed, Figure 1 demonstrates, for the City of Los Angeles, some months (October and December) have higher amounts of stolen vehicle incidents than others (April and May). The visualization of seasonal daily Stolen Vehicle Events in Figure 2, reveals seasons have meaningful impact on stolen vehicle activity (see the Technical Analysis section for deeper discussion). Other factors may be at play (see the Validity Discussion section). The Next Steps section provides paths for further investigation.



\

Figure 1. Number of Reported Stolen Vehicle Events for each month (combined from 2013 and 2014 Crime and Collision Los Angeles Police Department data).

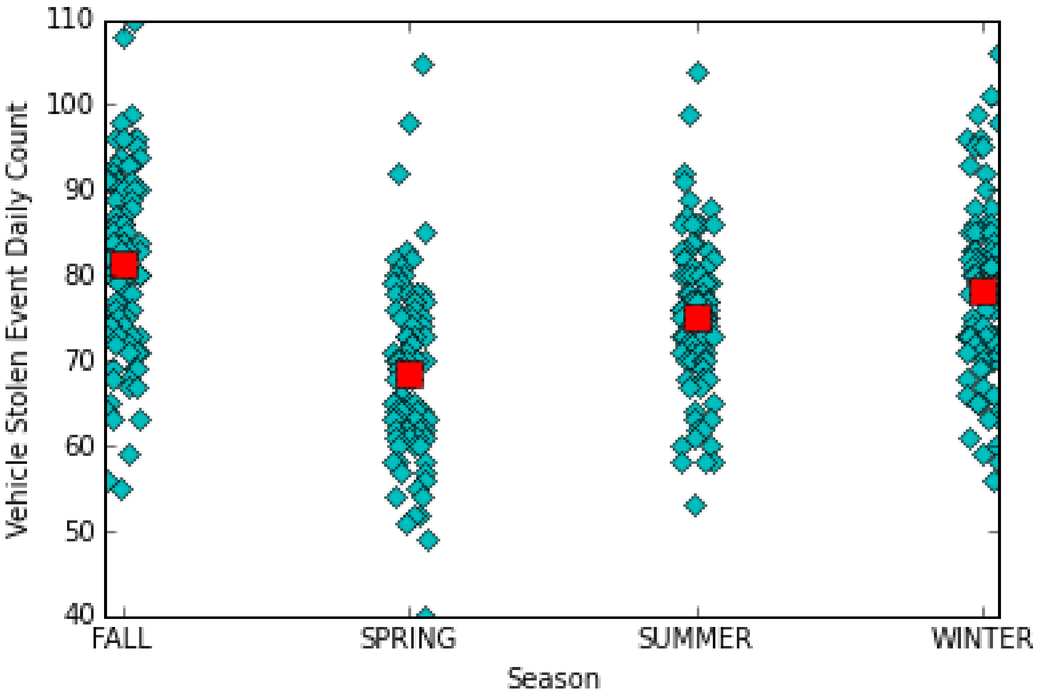


Figure 2. Number of Stolen Vehicle Events per Day compared across seasons (combined from 2013 and 2014 Crime and Collision Los Angeles Police Department data). Each blue diamond represents a calendar day. Each red square represents the average for the season.

2. Technical Analysis

[Summary for Data Science Peers]

Data from the Los Angeles Police Department spanning 2013 and 2014 was obtained from the LA City Open Data Portal (https://data.lacity.org/). By extracting data with the "Crm Cd" field, equal to 510, the records of "VEHICLE - STOLEN" reports were isolated.

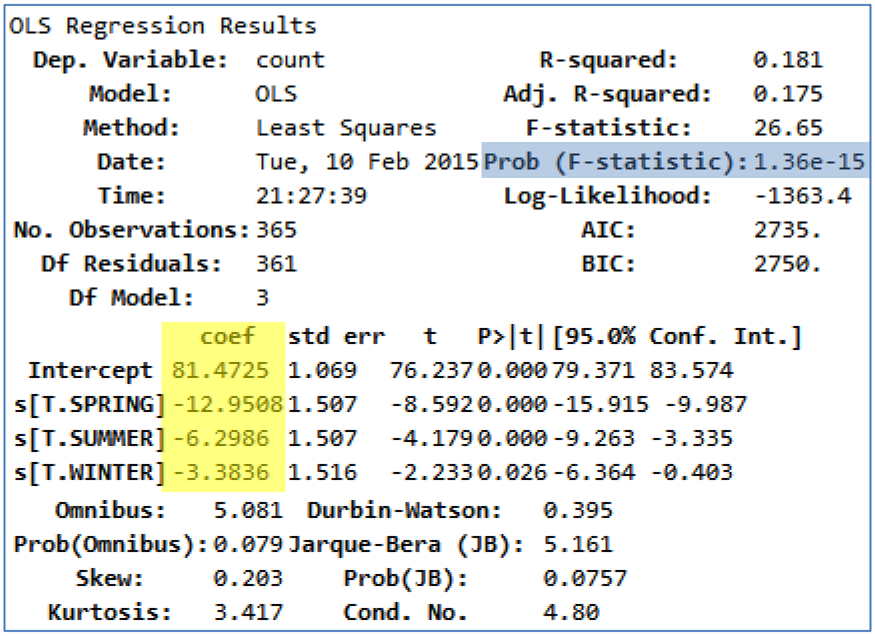
A simple value count for records that fall on the various 365 days of a year, followed by transforming the month into a season with a simple mapping function yields the groundwork to perform a multivariate linear regression using categorical variables.

The model followed the form:

$$vs = c_0 + c_1 S_{SPRING} + c_2 S_{SUMMER} + c_3 S_{WINTER} + e$$

Where
vs = Number of Stolen Vehicle Events per Day
c_0 = Average Number of Stolen Vehicle per Day in the Fall
$c_0 + c_1$ = Average Number of Stolen Vehicle per Day in the Spring
$c_0 + c_2$ = Average Number of Stolen Vehicle per Day in the Summer
$c_0 + c_3$ = Average Number of Stolen Vehicle per Day in the Winter
S_{SPRING} = Dummy Categorical variable with 1 representing Spring and 0 other seasons
S_{SUMMER} = Dummy Categorical variable with 1 representing Summer and 0 other seasons
S_{WINTER} = Dummy Categorical variable with 1 representing Winter and 0 other seasons
e = error term
When S_{SPRING}, S_{SUMMER}, and S_{WINTER} are all 0, this would indicate Fall.

The linear model yielded the following results:



```
OLS Regression Results
  Dep. Variable:  count               R-squared:      0.181
       Model:     OLS             Adj. R-squared:     0.175
      Method:     Least Squares      F-statistic:     26.65
        Date:     Tue, 10 Feb 2015 Prob (F-statistic): 1.36e-15
        Time:     21:27:39         Log-Likelihood:    -1363.4
No. Observations: 365                    AIC:         2735.
   Df Residuals:  361                    BIC:         2750.
     Df Model:    3
                coef    std err    t    P>|t| [95.0% Conf. Int.]
  Intercept   81.4725   1.069   76.237  0.000  79.371  83.574
s[T.SPRING]  -12.9508   1.507   -8.592  0.000 -15.915  -9.987
s[T.SUMMER]   -6.2986   1.507   -4.179  0.000  -9.263  -3.335
s[T.WINTER]   -3.3836   1.516   -2.233  0.026  -6.364  -0.403
   Omnibus:     5.081  Durbin-Watson:    0.395
Prob(Omnibus): 0.079  Jarque-Bera (JB): 5.161
     Skew:      0.203     Prob(JB):      0.0757
  Kurtosis:     3.417     Cond. No.      4.80
```

The values shaded in yellow correspond to c_0, c_1, c_2, and c_3. The Prob (F-statistic) value, shaded in blue, of 1.36e-15, indicates that there is greater than a 99.9% probability that the differences did not arise by chance.

3. Validity Discussion

[Commentary on Accuracy of Work]

Though the multivariate linear regression results show a statistically significant difference, the model has ample room to improve for greater accuracy. The Adj. R-squared value of 0.175 indicates that the model only describes less than 18% of the total variability. The choice of seasons delineated by three month blocks is arguably arbitrary and may undermine accuracy. The choice of season changes in this model followed a simplistic end of month assumption. Applying official season transitions may yield greater accuracy. The model essentially uses season as a proxy for other factors. A more accurate model would investigate the underlying factors that seasons approximate. More accurate prediction of stolen vehicle activity may be achieved instead by looking at location and time of day using statistical classification techniques. More accurate prediction may be achieved by inspecting specific weather events during low or high stolen vehicle activity.

4. Next Steps

- Investigate impact of nighttime length.
- Rerun project code to define seasons according to official seasonal transition definitions.
- Investigate weather events during and prior to high incidents of stolen vehicle activity.
- Run classification with respect to location and time of day.

5. Project Code (see next pages)

```
# -*- coding: utf-8 -*-
"""
# AlexKwan_ConnectHQ_project.py

@author: Alex Kwan
"""

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.formula.api import ols
from statsmodels.stats.anova import anova_lm


fpath = r"C:\Users\FS110729\Google Drive\DS\ConnnectHQ"
fname_2013 = r"\LAPD_Crime_and_Collision_Raw_Data_for_2013.csv"
fname_2014 = r"\LAPD_Crime_and_Collision_Raw_Data_-_2014.csv"

ds_2013 = pd.DataFrame.from_csv(path = fpath + fname_2013, index_col = 1)
ds_2014 = pd.DataFrame.from_csv(path = fpath + fname_2014, index_col = 1)


def season(x) :
    return {
        1 : 'WINTER',
        2 : 'WINTER',
        3 : 'SPRING',
        4 : 'SPRING',
        5 : 'SPRING',
        6 : 'SUMMER',
        7 : 'SUMMER',
        8 : 'SUMMER',
        9 : 'FALL',
        10 : 'FALL',
        11 : 'FALL',
        12 : 'WINTER',
    }.get(x,'WINTER')

# hokey function to change year in datetime to 2013
def ch_yr_to_2013(iyr) :
    return(iyr.replace(year=2013))

def jtr(s, factor):
    sep = float(s.max()) - float(s.min())
    b = float(factor) * sep / 50.
    return map(lambda x: x + np.random.uniform(-b, b), s)


# assemble 'VEHICLE - STOELN' records for 2013 and 2014
ds_2013_vs = ds_2013.loc[ds_2013['Crm Cd'] == 510]
ds_2014_vs = ds_2014.loc[ds_2014['Crm Cd'] == 510]

ds_2014_vsb = pd.to_datetime(ds_2014_vs['DATE OCC'])
ds_2014_vsb = ds_2014_vsb.apply(ch_yr_to_2013)
```

```
ds_2013_vsb = pd.to_datetime(ds_2013_vs['DATE OCC'])

ds_vsb = ds_2013_vsb.append(ds_2014_vsb)

vs_docc_freq = ds_vsb.value_counts()
vs_docc_freq = pd.DataFrame(vs_docc_freq)
vs_docc_freq.index.name = 'DATE OCC'
vs_docc_freq.rename(columns ={0 : 'count'},inplace=True)
vs_docc_freq['m']=vs_docc_freq.index.month
vs_docc_freq['d']=vs_docc_freq.index.day
vs_docc_freq['s']=vs_docc_freq['m'].apply(season)

# create barchart of vs per month
ds_vsb_mo = pd.to_datetime(ds_vsb)
ds_vsb_mo = pd.DataFrame(ds_vsb_mo)
ds_vsb_mo.set_index('DATE OCC', inplace=True)
ds_vsb_mo['m']=ds_vsb_mo.index.month
ds_vsb_mo_freq = pd.Series(ds_vsb_mo.m)
ds_vsb_mo_freq = ds_vsb_mo_freq.value_counts()
ds_vsb_mo_freq = pd.DataFrame(ds_vsb_mo_freq)
ds_vsb_mo_freq = ds_vsb_mo_freq.sort_index()

plt.figure()
ds_vsb_mo_freq.plot(kind='bar',legend=False)
plt.xlabel('Month')
plt.ylabel('Vehicle Stolen Events Quantity')


# model vs events per day with respect to season
seasons = pd.Categorical.from_array(vs_docc_freq['s'])

plt.plot(jtr(seasons.labels, 1.), vs_docc_freq['count'],'cD')
plt.xticks(np.unique(seasons.labels), seasons.levels)
plt.xlabel('Season')
plt.ylabel('Vehicle Stolen Events Daily Count')

meanDailyCounts = vs_docc_freq.groupby('s')['count'].mean()
plt.plot(range(4), meanDailyCounts, 'rs', markersize=10);

# ordinary least squares regression
linm1 = ols('count ~ s', vs_docc_freq).fit()

linm1.summary()

anova_lm(linm1)
```