Analysis of Crime Neighboring Sporting Events

Springboard data science capstone project

francisco salas

2018

Table of Contents

[Introduction 2](#_Toc520325399)

[Data Acquisition 2](#_Toc520325400)

[Crime data 2](#_Toc520325401)

[Sports data 3](#_Toc520325402)

[Houston Texans 3](#_Toc520325403)

[Stadium 4](#_Toc520325404)

[Houston Astros 4](#_Toc520325405)

[Stadium 5](#_Toc520325406)

[Houston Rockets 5](#_Toc520325407)

[Stadium 5](#_Toc520325408)

[Houston Dynamo 6](#_Toc520325409)

[Stadium 6](#_Toc520325410)

[University of Houston Football 6](#_Toc520325411)

[Stadium 7](#_Toc520325412)

[Rice University Football 7](#_Toc520325413)

[Stadium 8](#_Toc520325414)

[Data Cleaning 8](#_Toc520325415)

[Tools and Libraries 8](#_Toc520325416)

[Crime data 8](#_Toc520325417)

[Sports data 12](#_Toc520325418)

[Data Exploration 14](#_Toc520325419)

[Stadium data 18](#_Toc520325420)

[Modeling 19](#_Toc520325421)

[Data Pre-processing 19](#_Toc520325422)

[Feature Extraction 20](#_Toc520325423)

[EDF 23](#_Toc520325424)

[Football: Texans 23](#_Toc520325425)

[Baseball: Astros 25](#_Toc520325426)

[Soccer: Dynamo 26](#_Toc520325427)

[Basketball: Rockets 28](#_Toc520325428)

[College Football: University of Houston 29](#_Toc520325429)

[College Football: Rice University 30](#_Toc520325430)

[Using Models 32](#_Toc520325431)

[Results 33](#_Toc520325432)

[College Football: University of Houston 33](#_Toc520325433)

[College Football: Rice University 34](#_Toc520325434)

[Football: Texans 35](#_Toc520325435)

[Basketball: Rockets 37](#_Toc520325436)

[Baseball: Astros 39](#_Toc520325437)

[Soccer: Dynamo 40](#_Toc520325438)

# Introduction

Houston Texas with its 2.4 million residents is the fourth most populous city in the United States, just behind New York, Los Angeles, and Chicago. As with any large city, Houston has a rich sporting culture with five professional major league teams and two Division I college athletic programs. With so many sporting events through the year, what is the likelihood of crime around a sports stadium given event?

Crime happens, given the density of a population there exist [some connection] (https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1745-9125.1979.tb01285.x)of increase of crime.

However, how often does it happen around specific areas like a sports arena? It would be helpful to sports fans if they know the chance that crime around them given the arena.

The local police department could increase/decrease staff given the right information; also, city planners could use the information to determine the best way to use a city's land and resources. The goal of this project is to develop such a predictive model for only crime around stadium arenas in the city of Houston, Texas from the years 2010 to 2017.

# Data Acquisition

## Crime data

Datasets were acquired from several different sources. The first dataset contains [HPD Beat Crime Statistics](http://www.houstontx.gov/police/cs/crime-stats-archives.htm) crime data from the Houston police department and is part of the Uniform Crime Report program or UCR. It complies official data collected by law enforcement agencies across the United States. UCR criminal offenses are divided into two major groups: part I and part II.

Part I offenses are considered to be serious and are broken into two categories: violent and property crimes; they include murder, rape, robbery, aggravated assault, burglary, theft, and auto theft.

Part II offenses are all crime classifications other than those defined as Part I. some of those include: forgery, fraud, vandalism, prostitution, disorderly conduct.

The information contains in the reports are a monthly breakdown of Part I crimes for which HPD wrote police reports. The data shows the number of reports for the following crimes: murder, rape, robbery, aggravated assault, burglary, theft, and auto theft.

The ICR data is provided monthly in Microsoft Access format along with Microsoft Excel spreadsheet format.

Total of 96 files was downloaded, (12 months x 8 years), here is the crime dataset breakdown

|  |  |
| --- | --- |
| Variable | Description |
| *date* | Date of offense, include month/date/year |
| *Hour* | Approximate time when an event occurs, value form 0-24 |
| *Offense Type* | Type I offense |
| *Beat* | The geographic area of the city broken down for patrol and statistical purpose |
| *Premise* | Identify the type of location where crime occurs (apartment complex, parking lot, etc.) |
| *Block Range* | The value range of street |
| *Street Name* | Name of the street where the offense occurred |
| *Type* | Street type, rd, Blvd |
| *Suffix* | N, S, E, W |
| *Offenses* | Times offense happen within the time frame |

## Sports data

Houston has four major sports teams and two Division I schools. Data from each sport was acquired from separate locations

### Houston Texans

The Houston Texas is a professional American football team based in you guessed it, Houston, Texas. They compete in the National Football League (NFL). Every home game is played at the NRG Stadium (formerly Reliant Stadium). To obtain dates and scores [sportradar.us](https://sportradar.us/) free trial was used.

|  |  |
| --- | --- |
| Variable | Description |
| *schedule* | Game date and time |
| *home.alias* | Home team |
| *scoring.home\_points* | Home team score |
| *away.alias* | Away team |
| *scoring.away\_points* | Score from away team |
| *WIN* | Team that won |

### Stadium

|  |  |
| --- | --- |
| Name: NRG Stadium  Address: NRG Pkwy, Houston, TX 77054  Coordinates: 29.684722, -95.410833  Police Beat: 15E40 |  |

### Houston Astros

Houston Astros are an American professional baseball and current champions in Major League Baseball (MLB). Every home game is played at Minute Maid Park, (formerly Enron field). Dataset was acquired from [baseball-reference.com](https://www.baseball-reference.com/teams/HOU/2018-schedule-scores.shtml)

|  |  |
| --- | --- |
| Variable | Description |
| *Gm#* | Game number |
| *Year* | Season year |
| *date* | Date of game |
| *‘blank’* | Boxscore, link to more data from this game |
| *Tm* | Current team |
| *“blank”* | Has two values, “none” or “@ |
| *Opp* | Opponent team |
| *W/L* | Win or lost |
| *R* | Runs scored |
| *RA* | Runs allowed |
| *INN* | More than nine innings? |
| *W-L* | Win/loss record |
| *Rank* | Current rank |
| *GB* | Games back of division/league leader |
| *Time* | Time of game |
| *D/N* | Day or night game |
| *Attendance* | Sum of people attendance of the game |

### Stadium

|  |  |
| --- | --- |
| Name: Minute Maid Park  Address: 501 Crawford St, Houston, TX 77054  Coordinates: 29.756944, -95.355556  Police Beats: 1A10, 10H30,10H10 |  |

### Houston Rockets

Houston Rockets are an American basketball team and compete in the Nationa Basketball Association (NBA). Since 2001, every home game is played at the Toyota Center. Game data was acquired from [basketball-reference.com](https://www.basketball-reference.com/teams/HOU/2017_games.html)

|  |  |
| --- | --- |
| Variable | Description |
| *G* | Games |
| *date* | Date of game |
| *time* | Time value when the game happens |
| *‘blank’* | Boxscore, link to more data from this game |
| *“blank”* | Has two values, “none” or “@ |
| *opponent* | Opponent team |
| *“blank”* | Contains two values ‘W” & “L.” |
| *‘blank* | OT? |
| *Tm* | Points scored |
| *Opp* | Points scored by the opponent team |
| *W* | Wins |
| *L* | Losses |
| *Streak* | Games won or lost in a row. |
|  |  |

### Stadium

|  |  |
| --- | --- |
| Name : Toyota Center  Address: 1510 Polk St, Houston, TX 77054  Coordinates: 29.750833, -95.362222  Police Beats: 1A10, 10H30,10H40, 10H50 |  |

### Houston Dynamo

Houston Dynamo is an American professional soccer club that competes in the Major League Soccer (MLS). Every home game is played at BBVA Compass Stadium. Game data was acquired from github repo [FootballData](https://github.com/jokecamp/FootballData).

|  |  |
| --- | --- |
| Variable | Description |
| *full\_date* | Date of games |
| *home\_team* | Local team |
| *home\_score* | Local team score |
| *away\_team* | Away team |
| *away\_score* | Away team score |
| *winner* | Winner of match |

### Stadium

|  |  |
| --- | --- |
| Name: BBVA Compass Stadium  Address: 2200 Texas Ave Houston, TX  Coordinates: 29.7522, -95.3524  Police Beats: 1A10, 10H30, 10H10 |  |

### University of Houston Football

University of Houston football program is an NCAA Division I college football. Every home game is played at TDECU Stadium, which was built on the site formerly occupied by Robertson Stadium, where they played before. Game data was acquired from [sports-reference.com](https://www.sports-reference.com/cfb/schools/houston/2017-schedule.html)

|  |  |
| --- | --- |
| Variable | Description |
| *G* | Games |
| *date* | Date of game |
| *time* | Time value when the game happens |
| *day* | weekday |
| *school* | Home team |
| *“blank”* | Has two values, “none” or “@ |
| *opponent* | Opponent school |
| *conf* | Conference |
| *“blank”* | Contains two values ‘W” & “L.” |
| *Pts* | Points scored by the “School” team |
| *Opp* | Points scored by the opponent team |
| *W* | Wins |
| *L* | Losses |
| *TV* | Channel this game will be on |
|  |  |

### Stadium

|  |  |
| --- | --- |
| Name: BBVA Compass Stadium  Address: 2200 Texas Ave Houston, TX  Coordinates: 29.7522, -95.3524  Police Beats: 1A10, 10H30, 10H10 |  |

### Rice University Football

Rice Owls football program is an NCAA Division 1 college football. Every home game is played at Rice Stadium. Game data was acquired from [sports-reference.com](https://www.sports-reference.com/cfb/schools/rice/2017-schedule.html).

|  |  |
| --- | --- |
| Variable | Description |
| *G* | Games |
| *date* | Date of game |
| *time* | Time value when the game happens |
| *day* | weekday |
| *school* | Home team |
| *“blank”* | Has two values, “none” or “@ |
| *opponent* | Opponent school |
| *conf* | Conference |
| *“blank”* | Contains two values ‘W” & “L.” |
| *Pts* | Points scored by the “School” team |
| *Opp* | Points scored by the opponent team |
| *W* | Wins |
| *L* | Losses |
| *TV* | Channel this game will be on |
|  |  |

### Stadium

|  |  |
| --- | --- |
| Name: Rice Stadium  Address: 610 South Main St Houston, TX  Coordinates: 29.721944, -95.349167  Police Beats: 10H70, 10H80 |  |

# Data Cleaning

## Tools and Libraries

* *Pandas* used to analyze the data
* *glob*: a python module that implements globbing of directory contents
* *os*: a python module that allows python to ‘talk’ to the operating system.
* *NumPy*: a powerful scientific computing library in python.

## Crime data

Excel files were combine into one data frame

# combine all files into one df

all\_files = glob.glob(os.path.join(path, "\*.xls"))

df\_from\_each\_file = (pd.read\_excel(f) for f in all\_files)

df = pd.concat(df\_from\_each\_file, ignore\_index=True)

Several columns were named differently between months

# combine similar columns

df['BlockRange'] = pd.concat([df['Block Range'].dropna(),

df['BlockRange'].dropna()]).reindex\_like(df)

Method used to check for null values

df.apply(lambda x: sum(x.isnull()))

Some values had extra characters or empty space, pandas’ methods were used to clean up some columns.

# replace extra ' with empty space

crimes['Beat'] = crimes.Beat.str.replace("'", " ")

# strip empty spaces

crimes.Beat = crimes.Beat.str.strip()

Setting date column as index

# set date as datetime, index & sort

crimes.Date = pd.to\_datetime(crimes.Date)

crimes = crimes.set\_index('Date').sort\_index(ascending=True)

Extracting day data from index

# get day, weekday,month ,year

crimes['day'] = crimes.index.strftime('%d')

crimes['weekday'] = crimes.index.strftime('%A')

crimes['month'] = crimes.index.strftime('%b')

crimes['year'] = crimes.index.strftime('%Y')

After cleaning the crime dataset from 2010 to 2017, they were combined into one data frame

df.info()

<class 'pandas.core.frame.DataFrame'>

DatetimeIndex: 1075199 entries, 1914-09-08 to 2033-04-21

Data columns (total 11 columns):

Beat 1075199 non-null object

BlockRange 1075199 non-null object

StreetName 1075192 non-null object

OffenseType 1075199 non-null object

Premise 1075199 non-null object

NumOffenses 1075199 non-null float64

Hour 1075199 non-null float64

day 1075199 non-null object

weekday 1075199 non-null object

month 1075199 non-null object

year 1075199 non-null object

dtypes: float64(2), object(9)

memory usage: 98.4+ MB

As we can see from the datetimeIndex range, some values entered wrong. I created a fitter to select only events from 2010 to 2017

df = df['1/1/2010':'12/31/2017']

df.info()  
<class 'pandas.core.frame.DataFrame'>

DatetimeIndex: 1072618 entries, 2010-01-01 to 2017-12-31

Data columns (total 11 columns):

Beat 1072618 non-null object

BlockRange 1072618 non-null object

StreetName 1072611 non-null object

OffenseType 1072618 non-null object

Premise 1072618 non-null object

NumOffenses 1072618 non-null float64

Hour 1072618 non-null float64

day 1072618 non-null object

weekday 1072618 non-null object

month 1072618 non-null object

year 1072618 non-null object

dtypes: float64(2), object(9)

memory usage: 98.2+ MB

We now have a semi-clean dataset

df.head()

Beat BlockRange StreetName OffenseType \

Date

2010-01-01 5F30 13200-13299 northwest Theft

2010-01-01 20G10 9900-9999 richmond Theft

2010-01-01 14D20 8500-8599 rubin Aggravated Assault

2010-01-01 14D40 4200-4299 friar point Burglary

2010-01-01 10H70 4800-4899 austin Burglary

Premise NumOffenses Hour day weekday month \

Date

2010-01-01 department/discount store 1.0 22.0 1 Friday Jan

2010-01-01 apartment parking lot 1.0 16.0 1 Friday Jan

2010-01-01 road/street/sidewalk 1.0 7.0 1 Friday Jan

2010-01-01 residence/house 1.0 20.0 1 Friday Jan

2010-01-01 residence/house 1.0 21.0 1 Friday Jan

year

Date

2010-01-01 2010

2010-01-01 2010

2010-01-01 2010

2010-01-01 2010

2010-01-01 2010

Based on the location of the stadiums, we will select specific police beats that are within 1 mile radius of each stadium

# create a list of Beat names that we want

beats = ['10H10','10H30', '10H40', '10H50', '10H60','10H70', '10H80', '15E40', '1A10']

# filter column based on our list

selected\_beats = df.Beat.isin(beats)

# create a new dataframe for each selected beat and save

beat\_10H30 = df\_sb[df\_sb.Beat == '10H30']

beat\_10H10 = df\_sb[df\_sb.Beat == '10H10']

beat\_1A10 = df\_sb[df\_sb.Beat == '1A10']

beat\_10H40 = df\_sb[df\_sb.Beat == '10H40']

beat\_15E40 = df\_sb[df\_sb.Beat == '15H40']

beat\_10H50 = df\_sb[df\_sb.Beat == '10H50']

beat\_10H60 = df\_sb[df\_sb.Beat == '10H60']

beat\_10H70 = df\_sb[df\_sb.Beat == '10H70']

beat\_10H80 = df\_sb[df\_sb.Beat == '10H80']

Changed some values in the Hour column showed 24 instead of 0

## change 24 to 0 value

df\_sb.Hour.replace(24,0,inplace=True)

## Sports data

For the sports data

For the Dynamo dataset a function was created to combine several files

def cleanup(df,year):

'''function that cleans up dataframe'''

df['year'] = year # create col with var year

df['full\_date'] = df['date'] + ' ' + df['year'] # append date and year cols

df['full\_date'] = pd.to\_datetime(df['full\_date']) # convert full\_date to datetime

df['home\_score']= df['result'].str.split('-').apply(lambda x: x[0]) # split score vals

df['away\_score']= df['result'].str.split('-').apply(lambda x: x[1]) # split score vals

df = df[['full\_date','home\_team','home\_score','away\_team','away\_score']] # org df

# winner cols given value scores

df['winner'] = np.where(df['home\_score'] > df['away\_score'], df['home\_team'], df['away\_team'])

df = df.set\_index('full\_date').sort\_index(ascending=True) # set full\_date as index

return df

Astros dataset had parenthesis within the date column, they were removed using regular expression

mlb['full\_date'] = mlb['full\_date'].str.replace(r"\(.\*\)"," ")

Change the name of columns

hou\_rockets\_plo.rename(columns={

'Unnamed: 2':'Time',

'team': 'Team',

'Tm': 'Team\_score',

'Unnamed: 5': 'Location',

'Opp': 'Opponent\_score',

'Unnamed: 7': 'Result'

},inplace=True)

Create a function to get the median value of block range

def block\_split(df):

'''

split blockrange col values

then give median value as a string

'''

first = df.BlockRange.str.split(pat='-',expand=True)[0].astype('int')

second = df.BlockRange.str.split(pat='-',expand=True)[1].astype('int')

med = np.ceil((second + first)/2).astype('int')

med = med.astype('str')

street = df.StreetName

return med

Create a new column with a correct address

df['address'] = df[['block', 'StreetName']].apply(lambda x: ' '.join(x), axis=1)

create a new column that calls google maps API that returns various data given a full address.

def gm\_geocode(address,API\_KEY):

loc = '{}, Houston, TX'.format(address)

gmaps = googlemaps.Client(key=API\_KEY)

r = gmaps.geocode(loc)

#lat\_lng = tuple(r[0]['geometry']['location'].values())

#full\_add = r[0]['formatted\_address']

#return lat\_lng, full\_add

return r

df['tup\_add'] = df['address'].apply(gm\_geocode,args=(API\_KEY,))

# Data Exploration

Only selected Beats

df.Beat.value\_counts(dropna=False)

1A10 30650

10H70 22846

10H40 20920

10H50 16034

10H60 15394

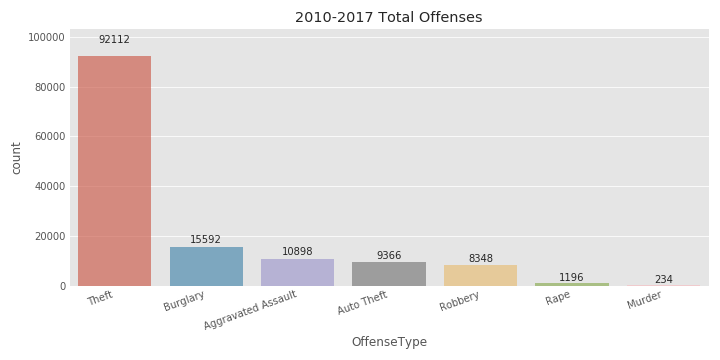
10H80 14962

10H30 8796

10H10 8144

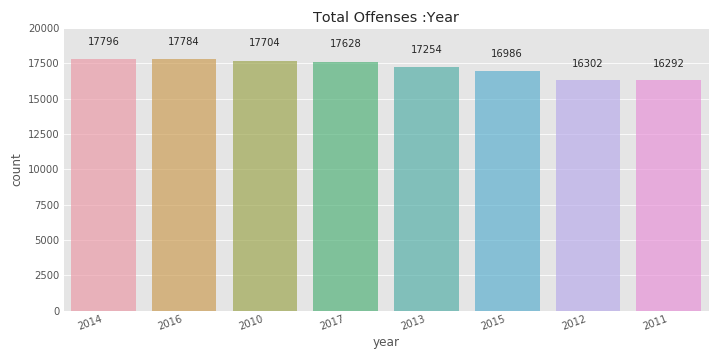
Name: Beat, dtype: int64

#### Sum of Total Offenses



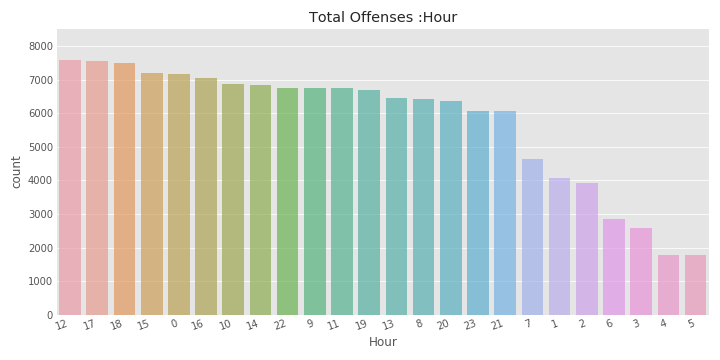
Out of 137746 cimes committed in the 8 Police beats from 2010 to 2017 Theft was the most common by a long shot.It actually surpasses all other offense types combined.

#### Sum of Offenses by Year



Out of the eight years, 2014 had the most offenses by just a few. 2011 and 2012 had the fewest in the 16000 range

#### Sum of Offenses by Hour



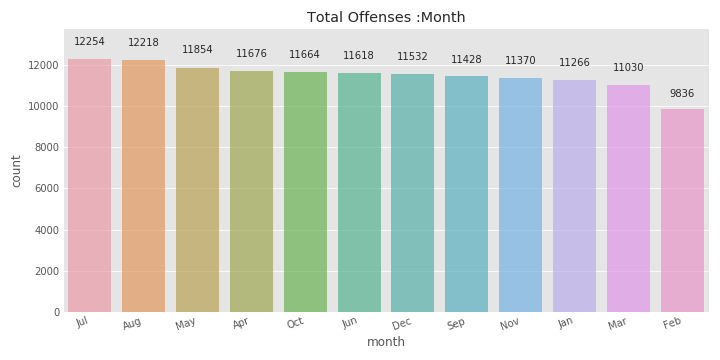
Midday is the most popular time when a crime is committed followed by 7 pm. It looks like 4-5 am are the lowest, but it could be that crimes are not reported during those time because most people are asleep.

#### Sum of Offenses by Weekday



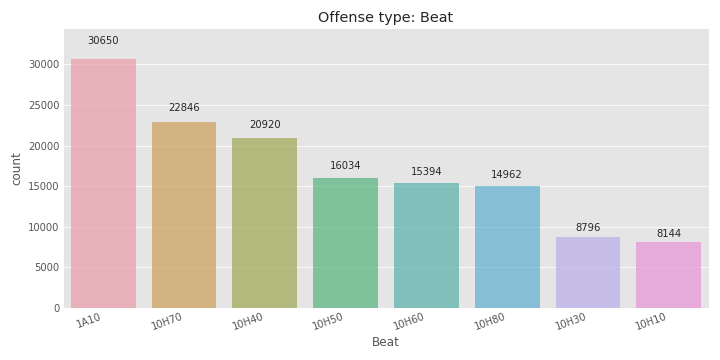
It seems that Friday is the most popular with 21710 offenses followed by Saturday with 20836. Sunday has the lowest crimes reported with 18446 offenses, almost 2000 less than Saturday

#### Sum of Offenses by Month



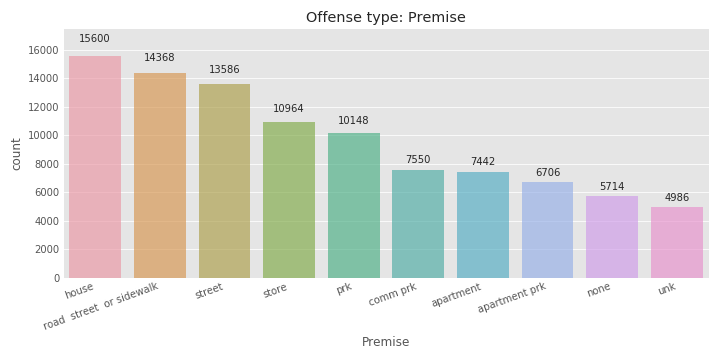
I don’t know if it's because of the hot weather but July is the highest month with 12,254 crimes committed. February is only two days short days (not counting leap years) and its 1,194 crimes shorter that March. Could 1000 crimes happen in 48 hrs?

#### Sum of Offenses by Beat



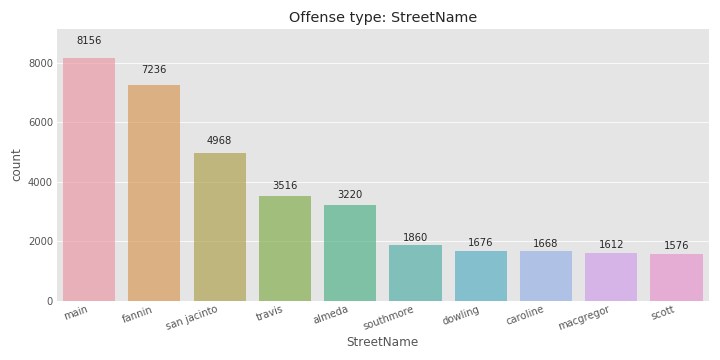
Police beat 1A10 has the highest crime with 30,650. Its probably because it’s the center of downtown

#### Sum of Offenses by Premise



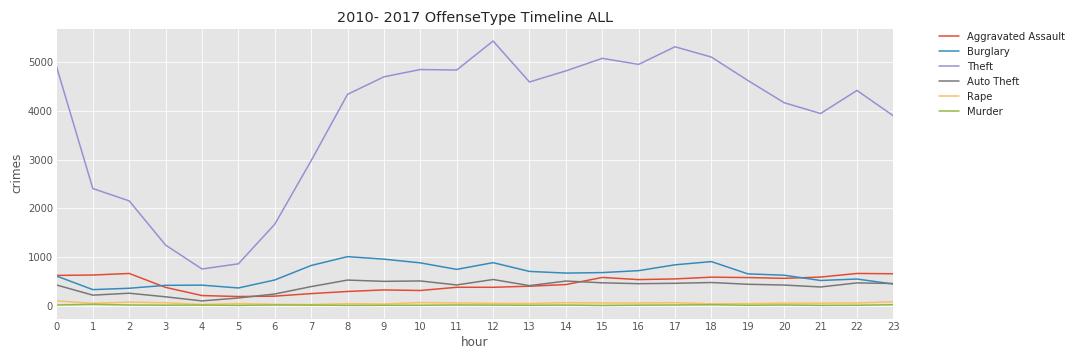
House or home is the highest with 15,600. Streets and sidewalks are also popular, a few unknown of missing data fills the bottom two columns.

#### Sum of Offenses by Street name

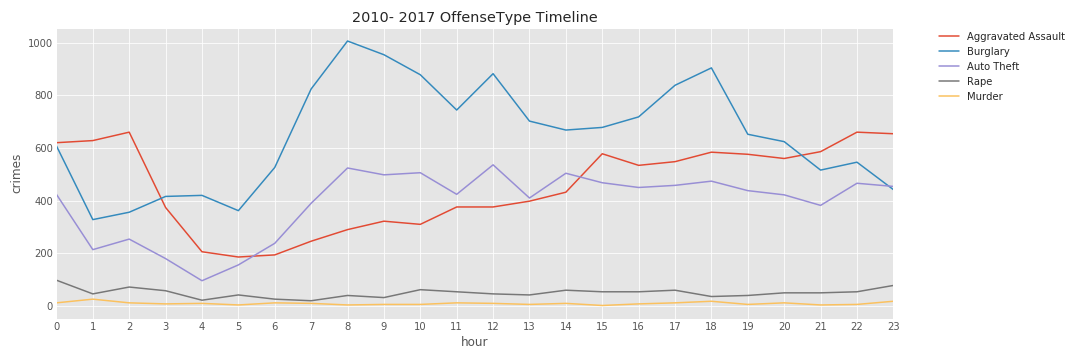


The top 4 streets are all in the area of downtown. with Main street as the most popular.

#### Timeline

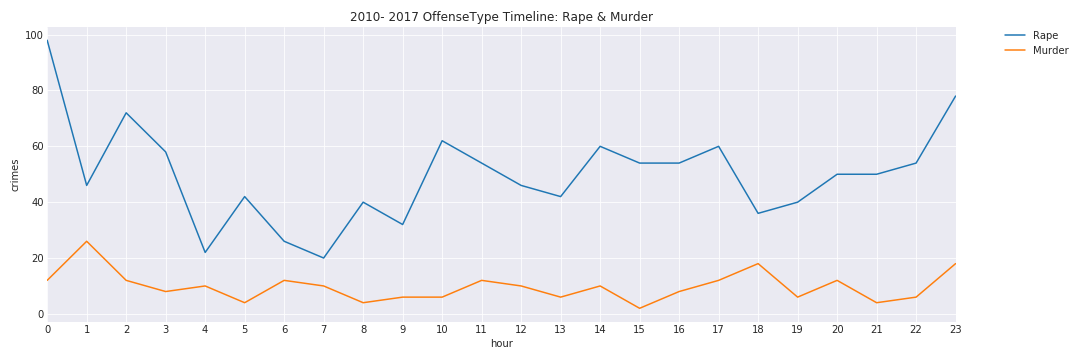


Theft seems to overflow the timeline since it has the most values. Let's remove it from the graph and plot the rest.



We can see that all crime drops between 2 am and 6 am. Burglary and auto theft are at their highest at 8 am. Aggravated Assault peaks at 3 pm.

Rape and Murder are too low to differentiate. Let's plot them separately.



3 pm is the lowest value for murder, and one is the highest. Midnight is the heist with rape with 4 am, and 7 am the lowest.

### Stadium data

To visualize the crime within each location, I plotted the stadium with a heat map of crime with a 1-mile radius

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Just by looking at these heat maps, NRG stadiums seem to have the least crimes followed by Rice Stadium BBVA Stadium, Minute Maid Park and Toyota Center are less than half a mile apart.

# Modeling

We have now cleaned the crime data that span from 2010 to 2017 and selected only eight police beats that surround each stadium; we will now join the score and schedule data for each team with the crime data.

### Data Pre-processing

Created an empty data frame with days

days = pd.date\_range(start='01/01/2010', end='12/30/2017')

days = pd.DataFrame(days)

days.columns = ['days']

days = days.set\_index('days').sort\_index(ascending=True)

Merge days data frame with crime data frame

calendar\_crimes = pd.merge(days,date\_crimes, left\_index=True, right\_index=True, how='left')

calendar\_crimes.head()

merge calendar\_crimes dataframe with scores dataframe

merge\_data = pd.merge(calendar\_crimes,df, left\_index=True, right\_index=True, how='left')

# change column names

merge\_data.columns = ['offenses','away\_team','win']

merge\_data.index.name = 'date'

Create a function that returns values bases on game score or game scheduled

def game\_feature(df):

if df.win == 1:

val = 'Won Game'

elif df.win == 0:

val = 'Lost Game'

else:

val = 'No Game'

return val

merge\_data['game'] = merge\_data.apply(game\_feature,axis=1)

OffenseType Premise hour weekday month year dist\_stadium \

date

2010-01-01 Auto Theft bar\_nc prk 0 Friday Jan 2010 0.137184

2010-01-01 Theft bar\_nc 0 Friday Jan 2010 0.549562

2010-01-01 Burglary office bld 0 Friday Jan 2010 0.480008

2010-01-01 Theft unk 0 Friday Jan 2010 0.734357

2010-01-01 Theft convention 0 Friday Jan 2010 0.403381

game

date

2010-01-01 No Game

2010-01-01 No Game

2010-01-01 No Game

2010-01-01 No Game

### Feature Extraction

Create a function that extract part of the day feature from hour column

def day\_feature(df):

mo = [6,7,8,9,10,11] # morning, sunrise to 11

af = [12,13,14,15,16] # afternoon to fiveish

ev = [17,18,19,20] # evening to sunset

ni = [21,22,23,0,1,2,3,4,5] # night, sunset to sunrise

if df.hour in mo:

val = 'Morning'

elif df.hour in af:

val = 'Afternoon'

elif df.hour in ev:

val = 'Evening'

else:

val = 'Night'

return val

df['part\_day'] = df.apply(day\_feature,axis=1)

Create a function that extracts weather season from DateTime index

def season\_feature(df):

'''

spring (March, April, May),

summer (June, July, August),

autumn (September, October, November)

winter (December, January, February).

'''

sp = ['Mar','Apr','May'] # spring

su = ['Jun','Jul','Aug'] # summer

au = ['Sep','Oct','Nov'] # autumn/fall

wi = ['Dec','Jan','Feb'] # winter

if df.month in sp:

val = 'Spring'

elif df.month in su:

val = 'Summer'

elif df.month in au:

val = 'Autumn'

else:

val = 'Winter'

return val

df['season'] = df.apply(season\_feature,axis=1)

OffenseType Premise hour weekday month year dist\_stadium \

date

2010-01-01 Auto Theft bar\_nc prk 0 Friday Jan 2010 0.137184

2010-01-01 Theft bar\_nc 0 Friday Jan 2010 0.549562

2010-01-01 Burglary office bld 0 Friday Jan 2010 0.480008

2010-01-01 Theft unk 0 Friday Jan 2010 0.734357

2010-01-01 Theft convention 0 Friday Jan 2010 0.403381

game part\_day season

date

2010-01-01 No Game Night Winter

2010-01-01 No Game Night Winter

2010-01-01 No Game Night Winter

2010-01-01 No Game Night Winter

2010-01-01 No Game Night Winter

Groupby date index and get mode values

# get mode value of of part\_day column

df['part\_day\_mode'] = df.groupby(df.index)['part\_day'].agg(lambda x: scipy.stats.mode(x, axis=None)[0][0])

# get mode value of hour column

df['hour\_mode'] = df.groupby(df.index)['hour'].agg(lambda x: scipy.stats.mode(x, axis=None)[0][0])

#get mode value of Premise column

df['premise\_mode'] = df.groupby(df.index)['Premise'].agg(lambda x: scipy.stats.mode(x, axis=None)[0][0])

# get mode value from offenseType column

df['offenseType\_mode'] = df.groupby(df.index)['OffenseType'].agg(lambda x: scipy.stats.mode(x, axis=None)[0][0])

OffenseType Premise hour weekday month year dist\_stadium \

date

2010-01-01 Auto Theft bar\_nc prk 0 Friday Jan 2010 0.137184

2010-01-01 Theft bar\_nc 0 Friday Jan 2010 0.549562

2010-01-01 Burglary office bld 0 Friday Jan 2010 0.480008

2010-01-01 Theft unk 0 Friday Jan 2010 0.734357

2010-01-01 Theft convention 0 Friday Jan 2010 0.403381

game part\_day season part\_day\_mode hour\_mode premise\_mode \

date

2010-01-01 No Game Night Winter Night 0 bar\_nc

2010-01-01 No Game Night Winter Night 0 bar\_nc

2010-01-01 No Game Night Winter Night 0 bar\_nc

2010-01-01 No Game Night Winter Night 0 bar\_nc

2010-01-01 No Game Night Winter Night 0 bar\_nc

offenseType\_mode

date

2010-01-01 Theft

2010-01-01 Theft

2010-01-01 Theft

2010-01-01 Theft

Select specific columns

df = df[['OffenseType', 'weekday', 'month', 'year',

'dist\_stadium', 'game', 'season', 'part\_day\_mode',

'hour\_mode', 'premise\_mode', 'offenseType\_mode']]

Finalize dataset by grouped by date index and getting median value of dist\_stadium

cdf = df.groupby(df.index).agg(

{'OffenseType':'count',

'weekday':'first',

'month':'first',

'year': 'first',

'dist\_stadium':'median',

'season':'first',

'part\_day\_mode':'first',

'hour\_mode':'first',

'premise\_mode':'first',

'offenseType\_mode':'first',

'game':'first'})

cdf.head()

OffenseType weekday month year dist\_stadium season \

date

2010-01-01 16 Friday Jan 2010 0.499216 Winter

2010-01-02 12 Saturday Jan 2010 0.575038 Winter

2010-01-03 10 Sunday Jan 2010 0.493969 Winter

2010-01-04 5 Monday Jan 2010 0.648818 Winter

2010-01-05 7 Tuesday Jan 2010 0.706555 Winter

part\_day\_mode hour\_mode premise\_mode offenseType\_mode game

date

2010-01-01 Night 0 bar\_nc Theft No Game

2010-01-02 Night 14 street Theft No Game

2010-01-03 Night 0 street Theft No Game

2010-01-04 Afternoon 14 store Theft No Game

2010-01-05 Morning 6 comm bld Theft No Game

Rename columns again for simple understanding

cdf.rename(columns={

'OffenseType':'crime\_total',

'dist\_stadium':'dist\_stadium\_meadian',

'offenseType\_mode':'offense\_mode'}, inplace=True)

crime\_total weekday month year dist\_stadium\_meadian season \

date

2010-01-01 16 Friday Jan 2010 0.499216 Winter

2010-01-02 12 Saturday Jan 2010 0.575038 Winter

2010-01-03 10 Sunday Jan 2010 0.493969 Winter

2010-01-04 5 Monday Jan 2010 0.648818 Winter

2010-01-05 7 Tuesday Jan 2010 0.706555 Winter

part\_day\_mode hour\_mode premise\_mode offense\_mode game

date

2010-01-01 Night 0 bar\_nc Theft No Game

2010-01-02 Night 14 street Theft No Game

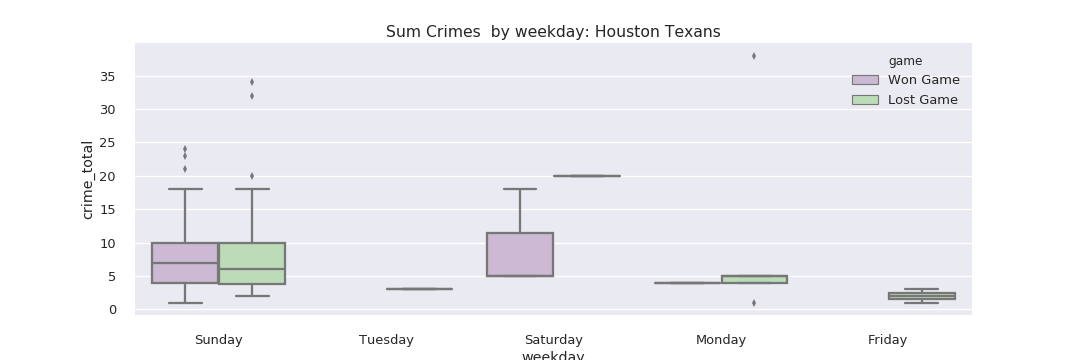
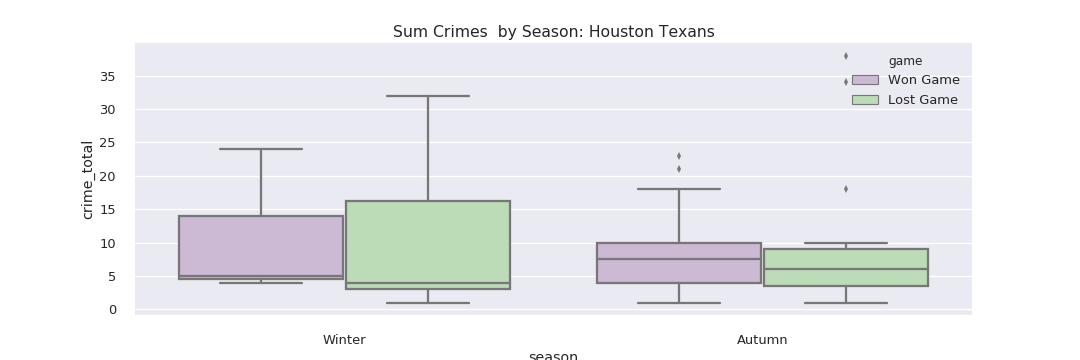
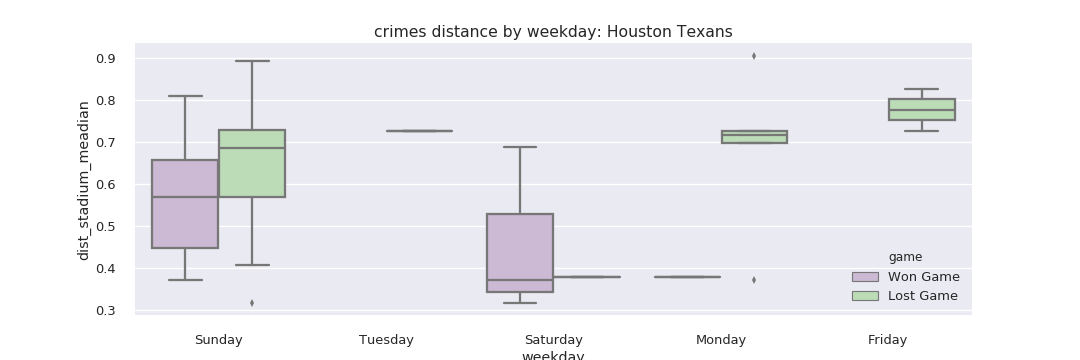
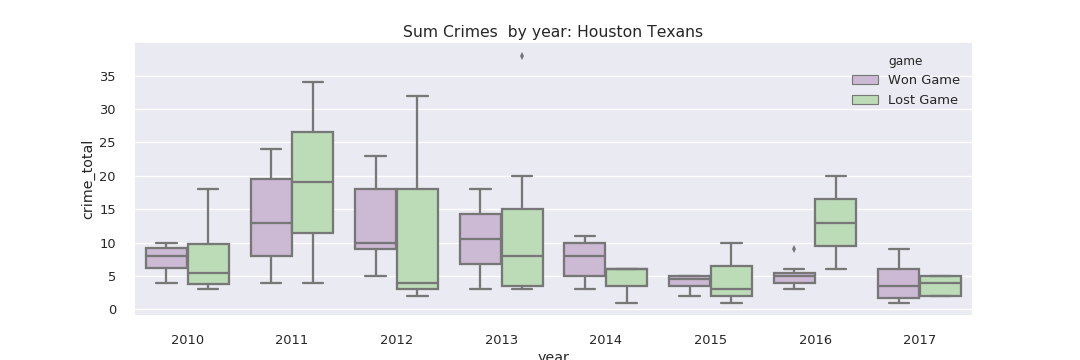
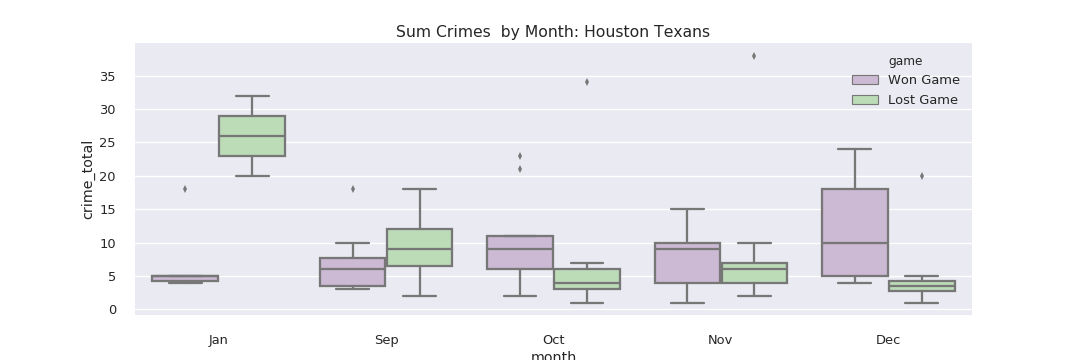
2010-01-03 Night 0 street Theft No Game

2010-01-04 Afternoon 14 store Theft No Game

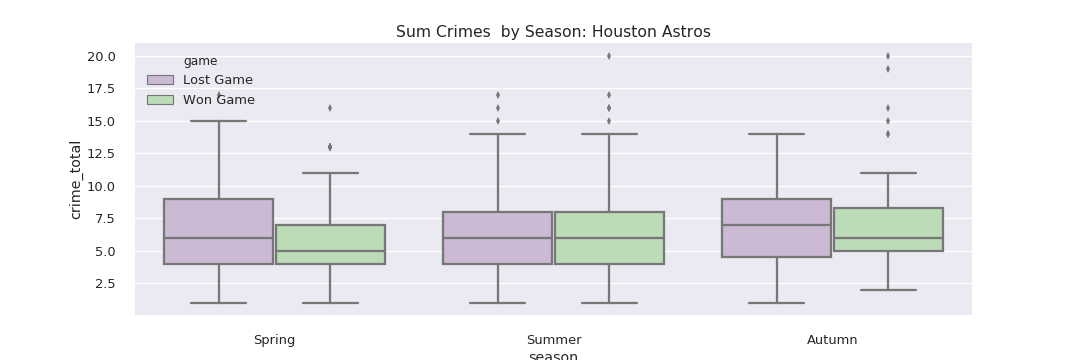
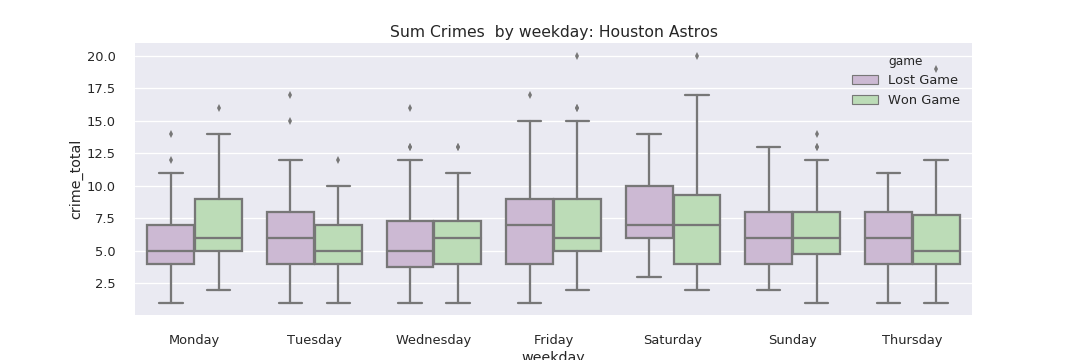
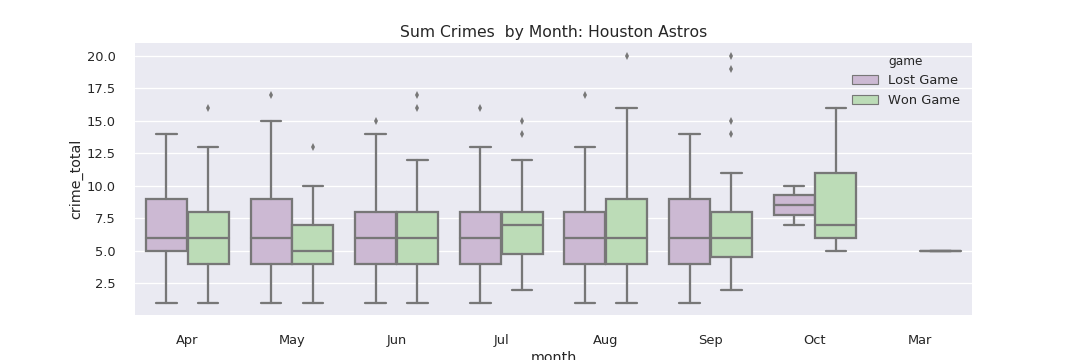
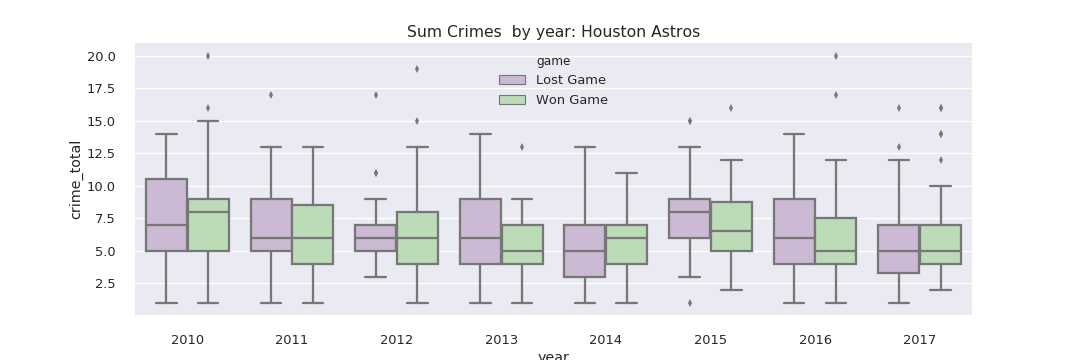
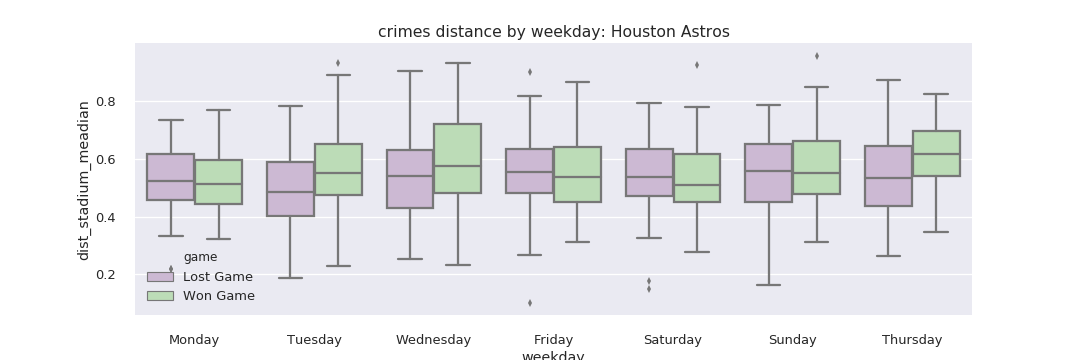
2010-01-05 Morning 6 comm bld Theft No Game

### EDF

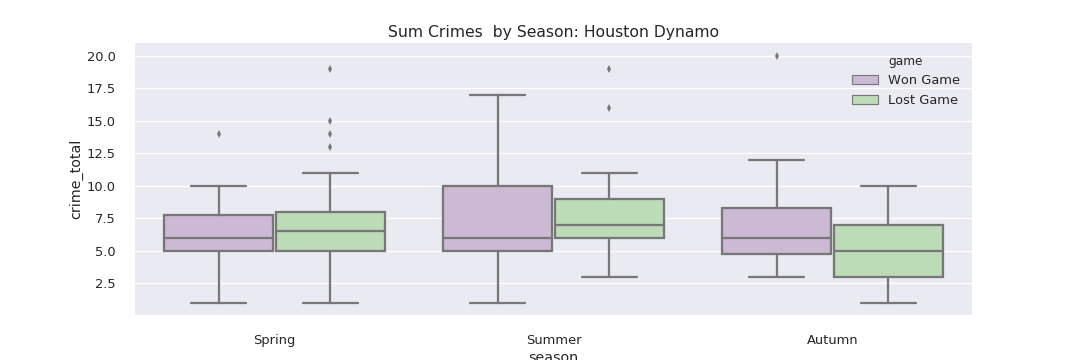
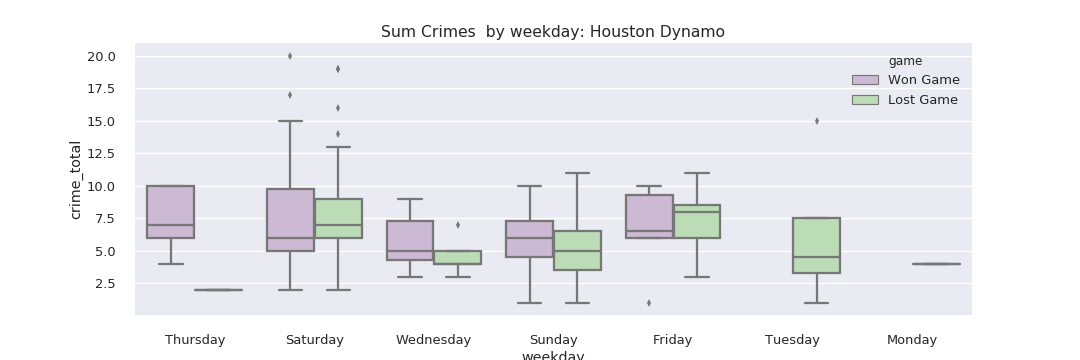
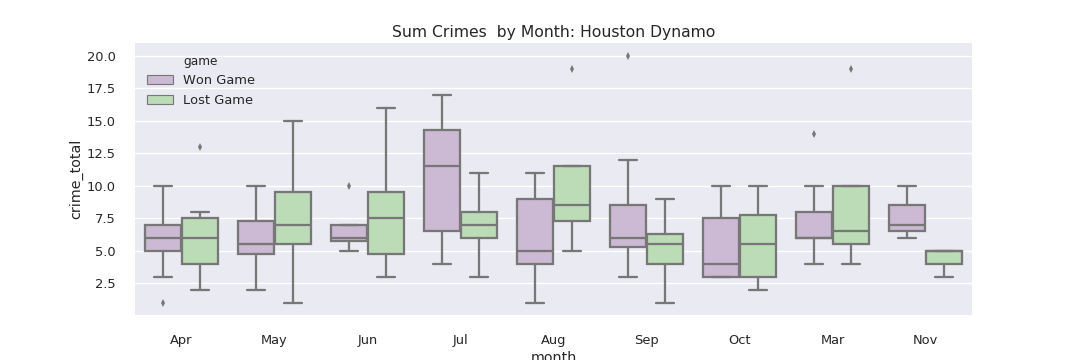
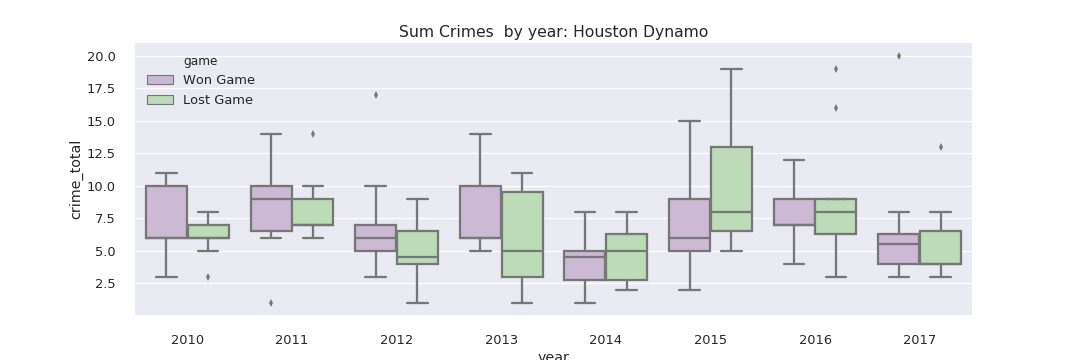
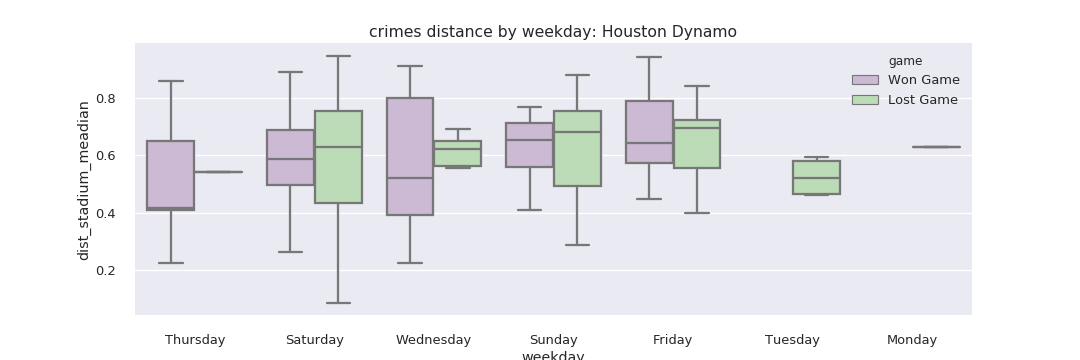
### Football: Texans



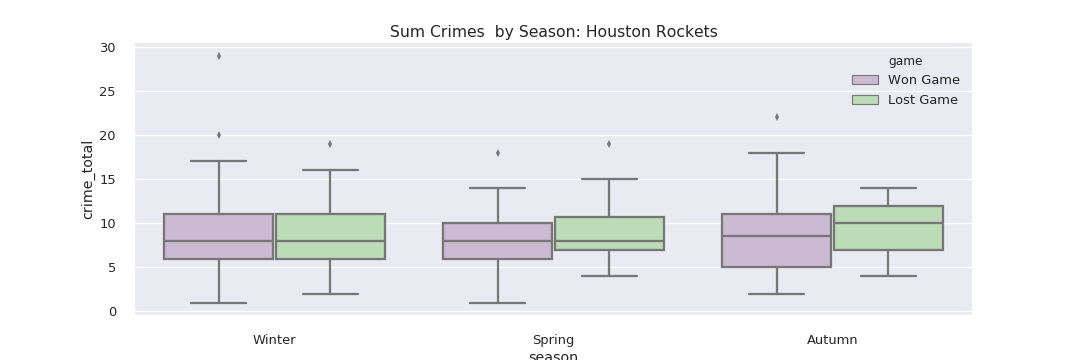
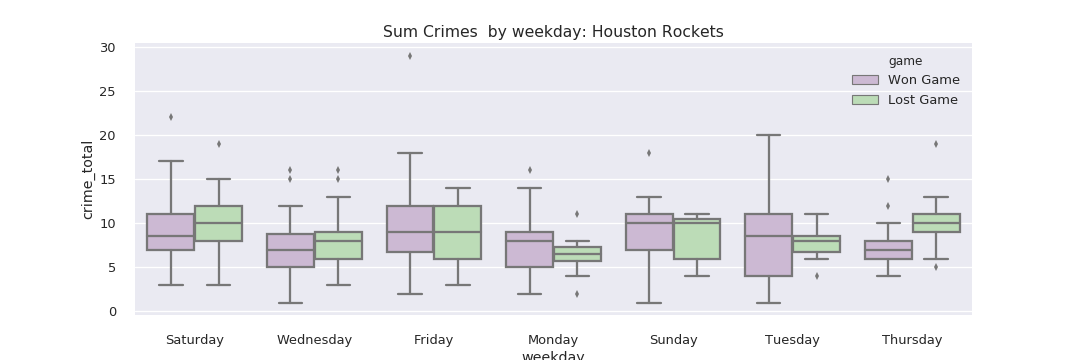
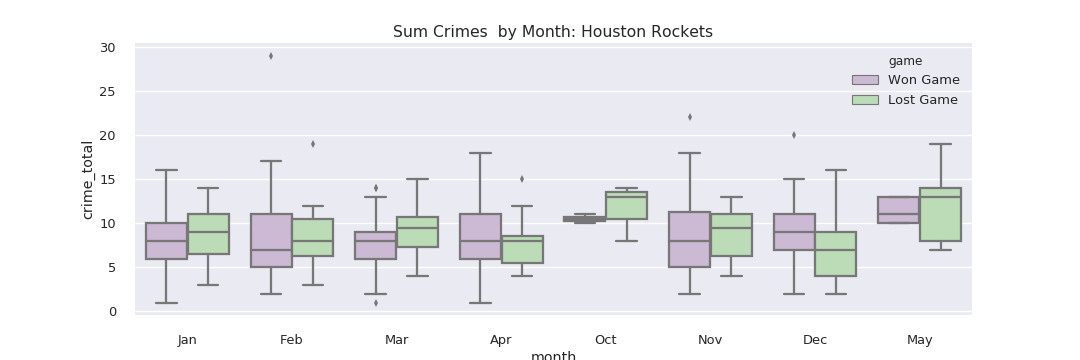
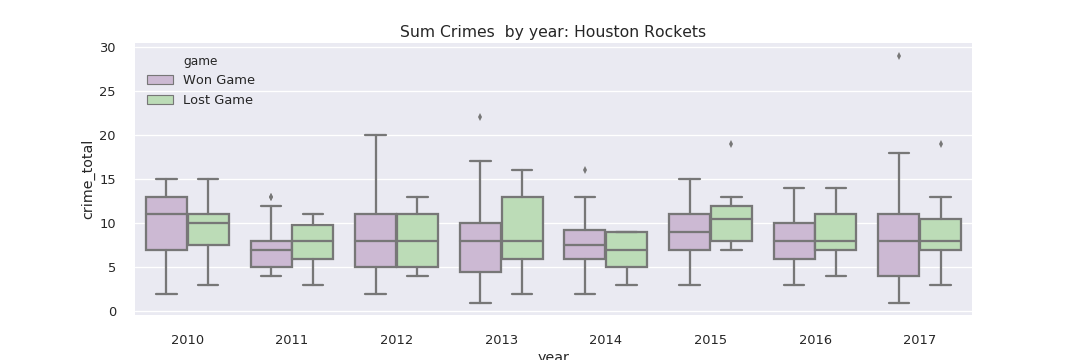
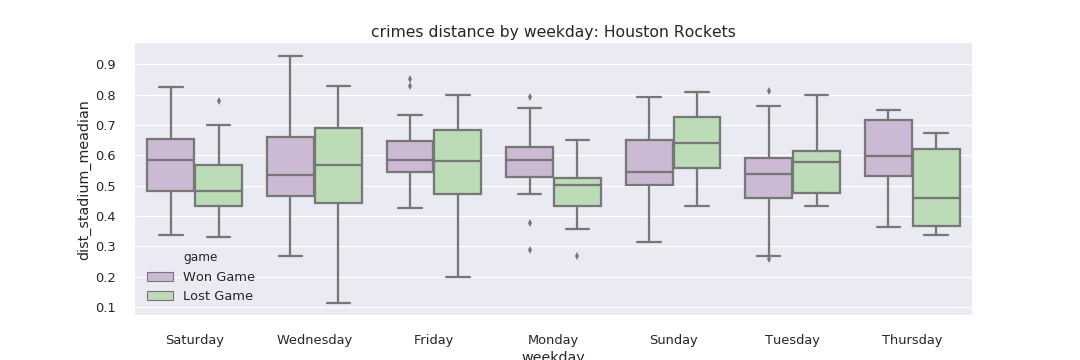
### Baseball: Astros



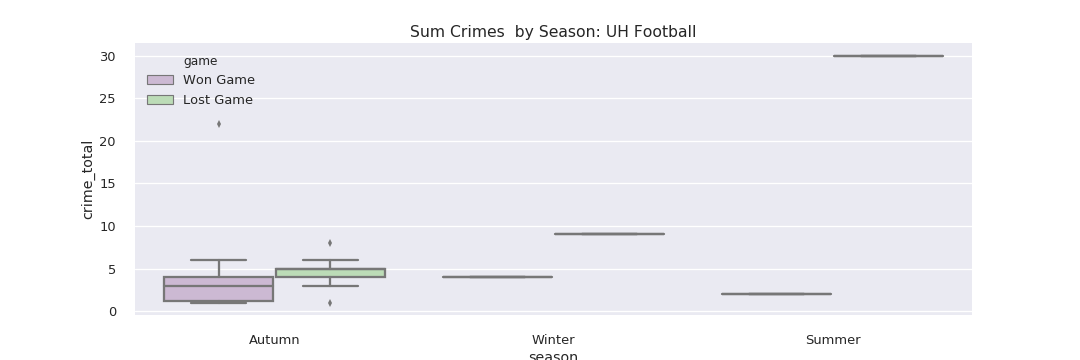
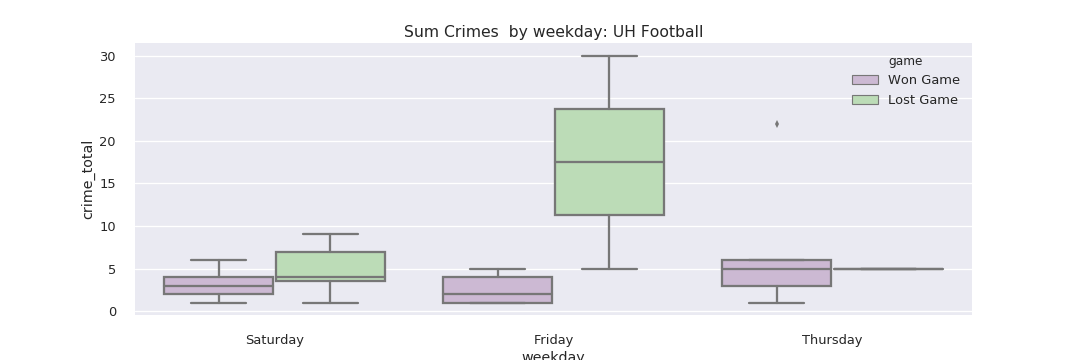
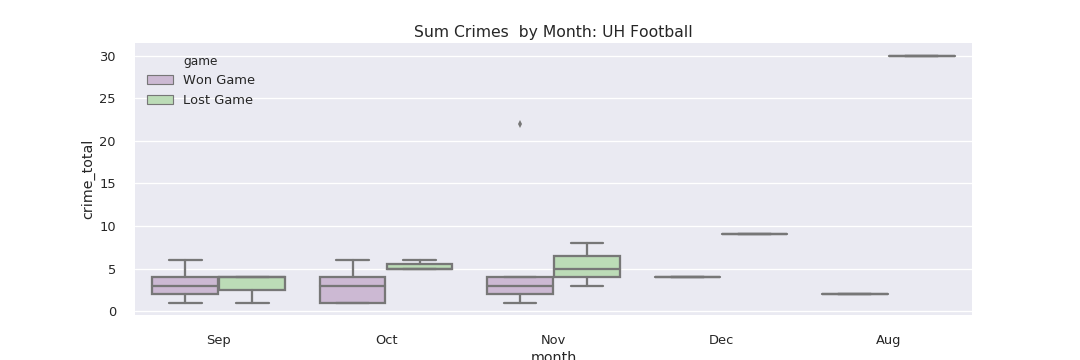
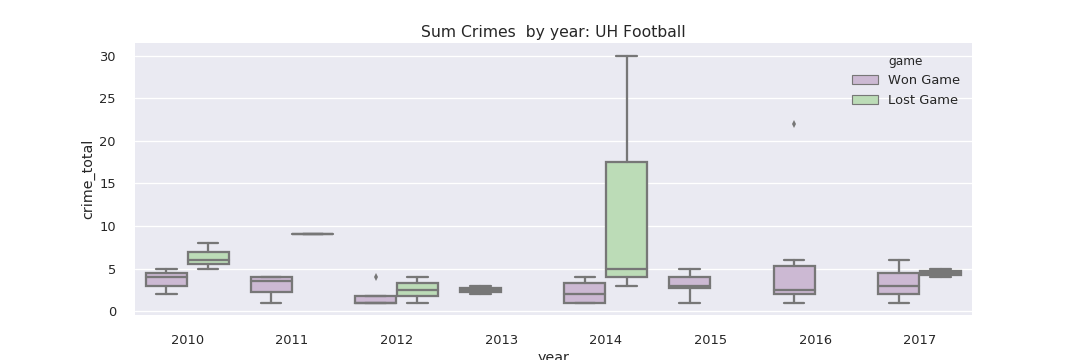
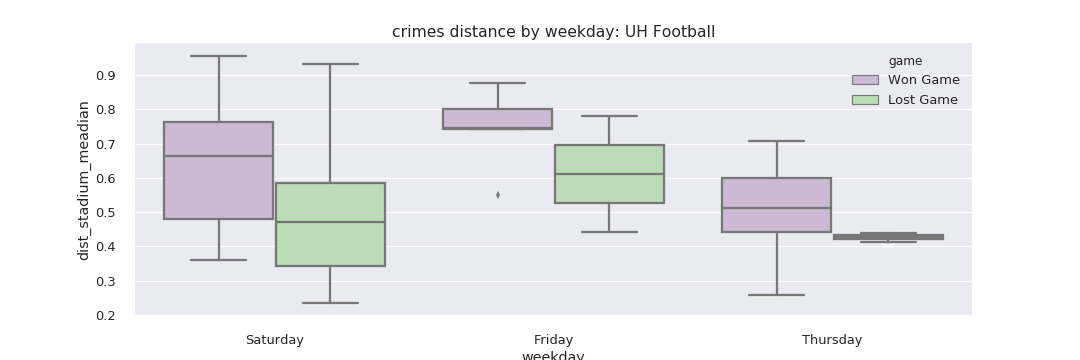
### Soccer: Dynamo



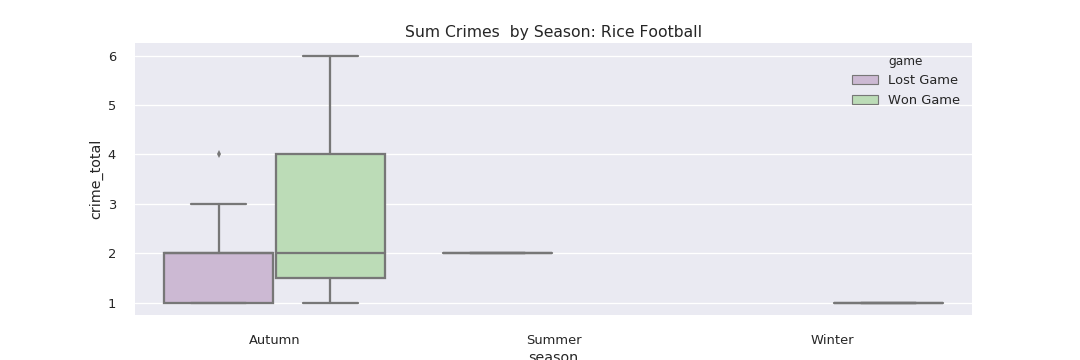
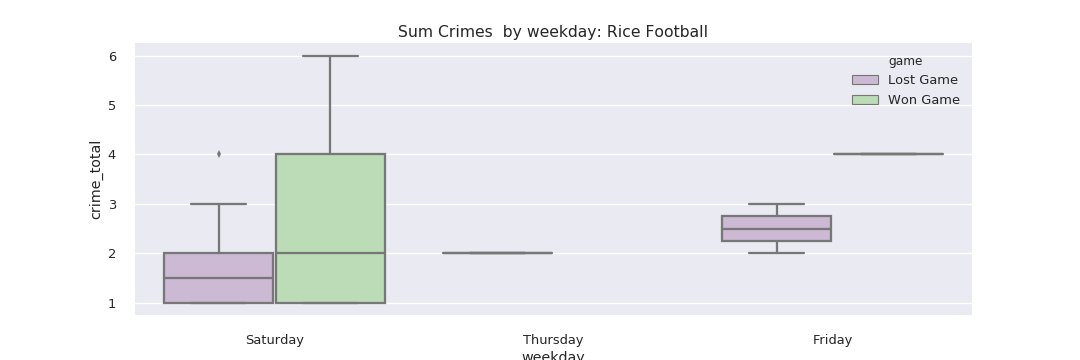
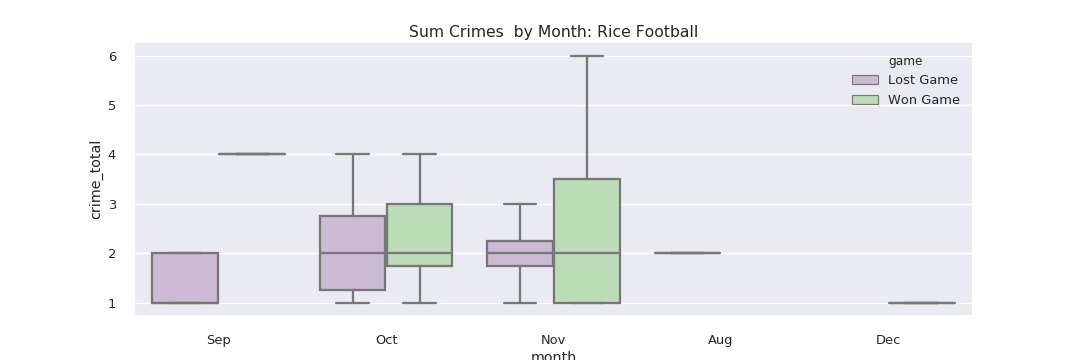
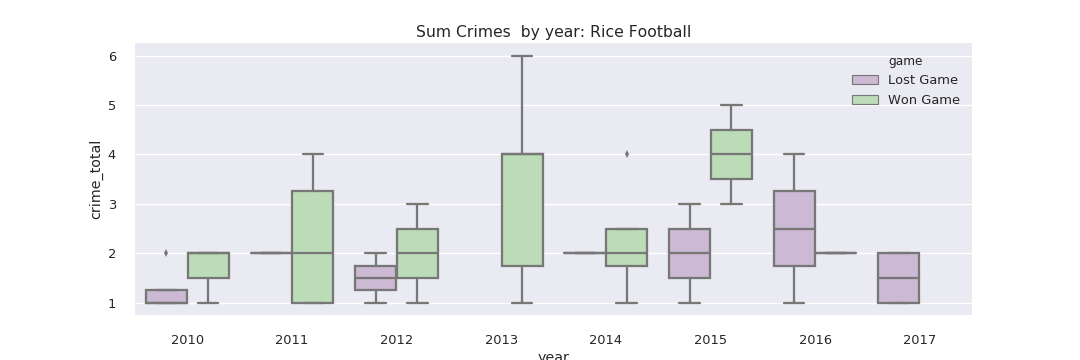
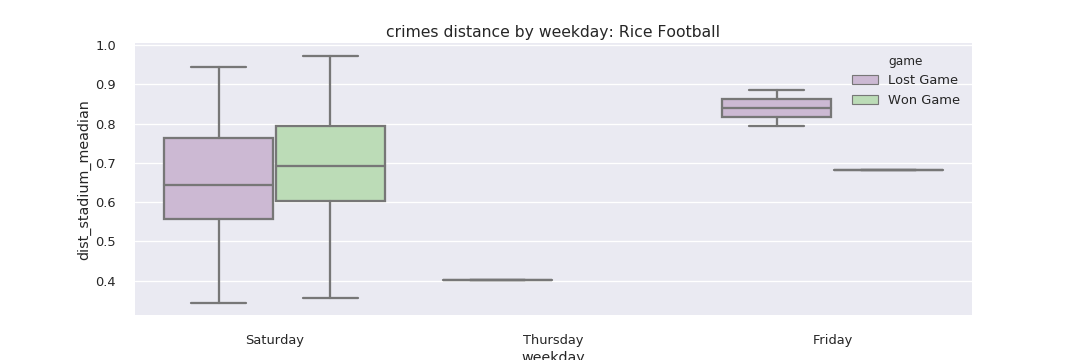
### Basketball: Rockets



### College Football: University of Houston



### College Football: Rice University



# Using Models

A function was created to expedite process

def modelfit(alg, X,y):

'''target = y, predictors = X, alg = algorithm used

'''

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.3, random\_state=42)

#Fit the algorithm on the data

alg.fit(X\_train,y\_train)

#Predict training set:

train\_predictions = alg.predict(X\_train)

#Perform cross-validation:

cv\_score = cross\_val\_score(alg, X, y, cv=10, scoring='neg\_mean\_squared\_error')

cv\_score = np.sqrt(np.abs(cv\_score))

#Print model report:

print ("\nModel Report")

print( "RMSE : %.4g" % np.sqrt(metrics.mean\_squared\_error(y\_train, train\_predictions)))

print ("CV Score : Mean %.4g | Std %.4g | Min %.4g | Max %.4g" % \

(np.mean(cv\_score),np.std(cv\_score),np.min(cv\_score),np.max(cv\_score)))

* **Linear Regression:** Ordinary least squares Linear Regression.
* **Ridge:** Linear least squares with l2 regularization.
* **Lasso:** Linear Model trained with L1 prior as regularized.

alg1 = LinearRegression(normalize=True)

alg2 = Ridge(alpha=0.1,normalize=True)

alg3 = Lasso(alpha=0.1,normalize=True)

# Results

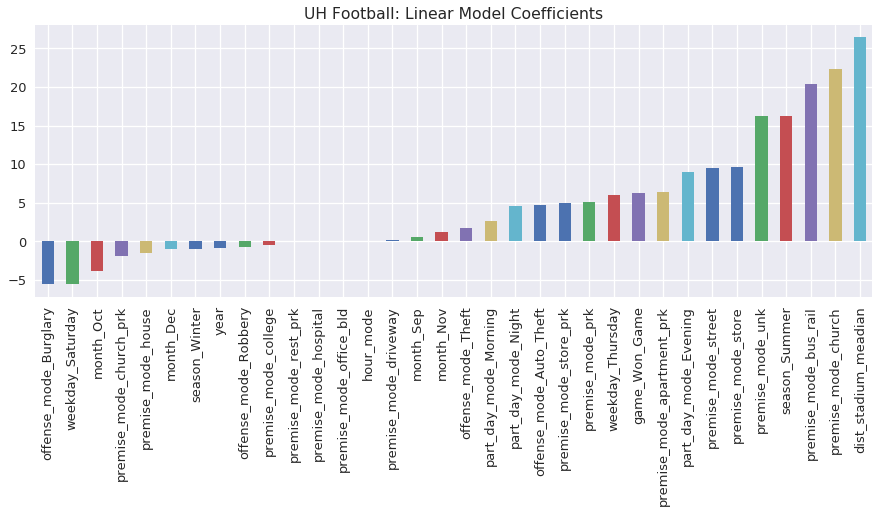
### College Football: University of Houston

#### Linear

Model Report

RMSE : 1.077

CV Score : Mean 5.234e+13 | Std 1.317e+14 | Min 3.493 | Max 4.45e+14

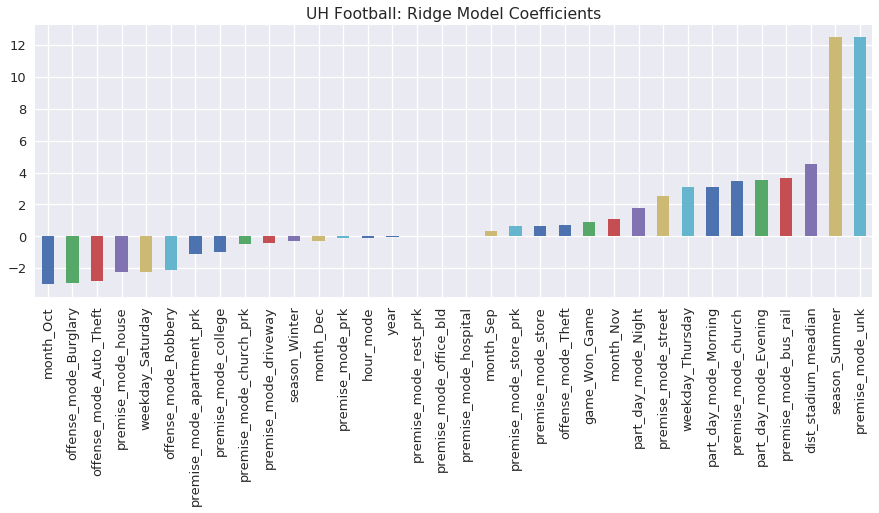


#### Ridge

Model Report

RMSE : 1.704

CV Score : Mean 4.767 | Std 3.785 | Min 1.255 | Max 13.5

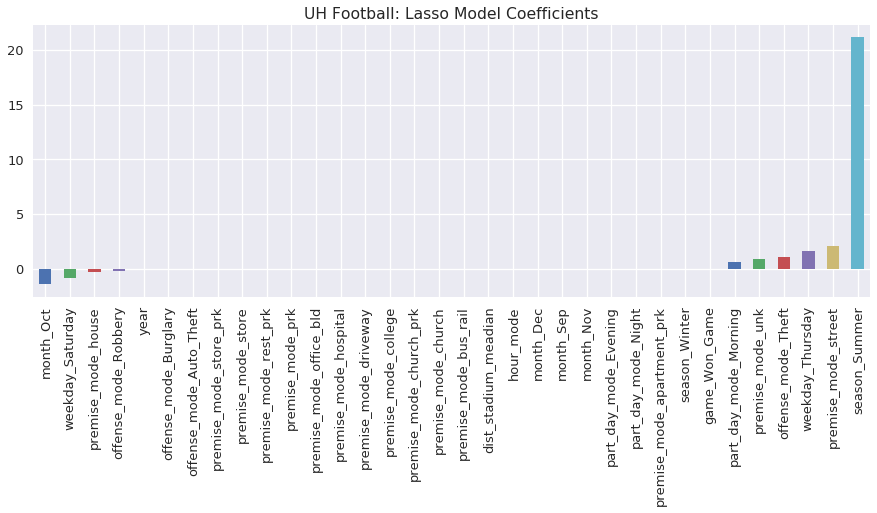


#### Lasso

Model Report

RMSE : 2.782

CV Score : Mean 4.701 | Std 4.044 | Min 1.393 | Max 12.09



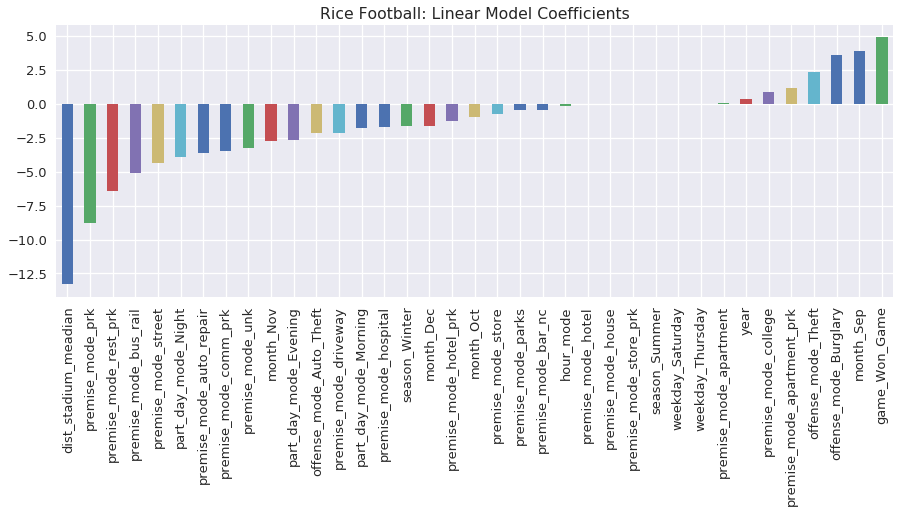
### College Football: Rice University

#### Linear

Model Report

RMSE : 0.0716

CV Score : Mean 2.004e+14 | Std 4.137e+14 | Min 1.32 | Max 1.32e+15

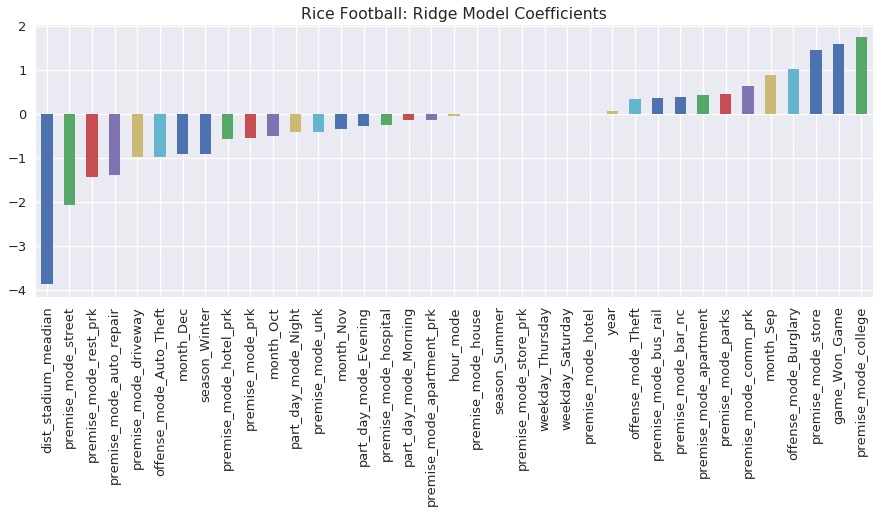


#### Ridge

Model Report

RMSE : 0.6496

CV Score : Mean 1.595 | Std 0.7066 | Min 0.2395 | Max 2.417

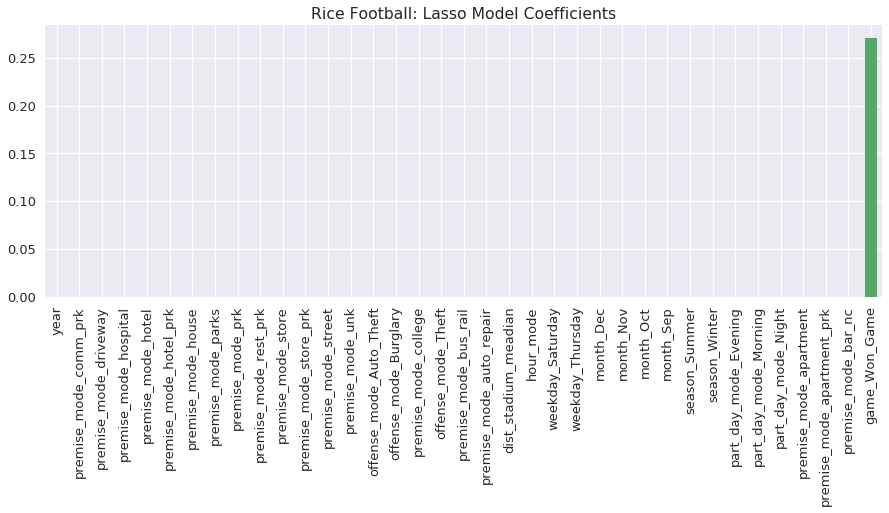


#### Lasso

Model Report

RMSE : 1.357

CV Score : Mean 1.283 | Std 0.4084 | Min 0.7061 | Max 2.274



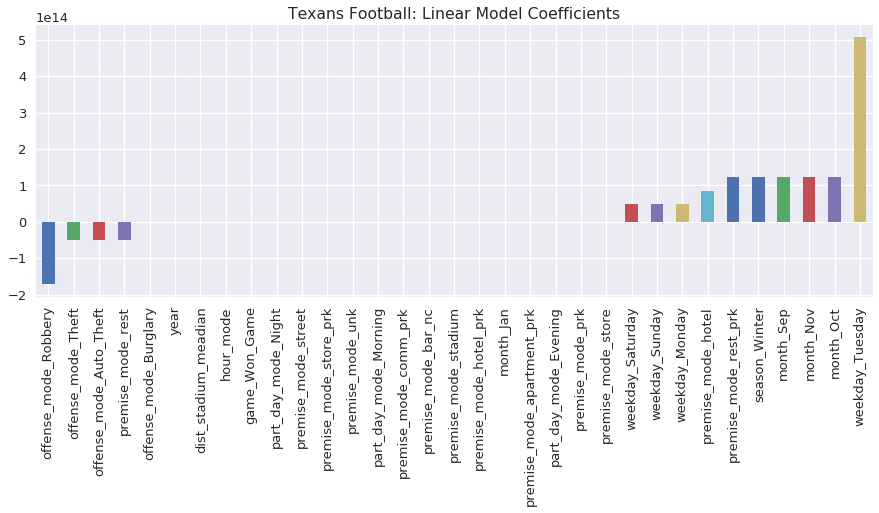
### Football: Texans

#### Linear

Model Report

RMSE : 4.733

CV Score : Mean 3.006e+13 | Std 9.019e+13 | Min 5.866 | Max 3.006e+14

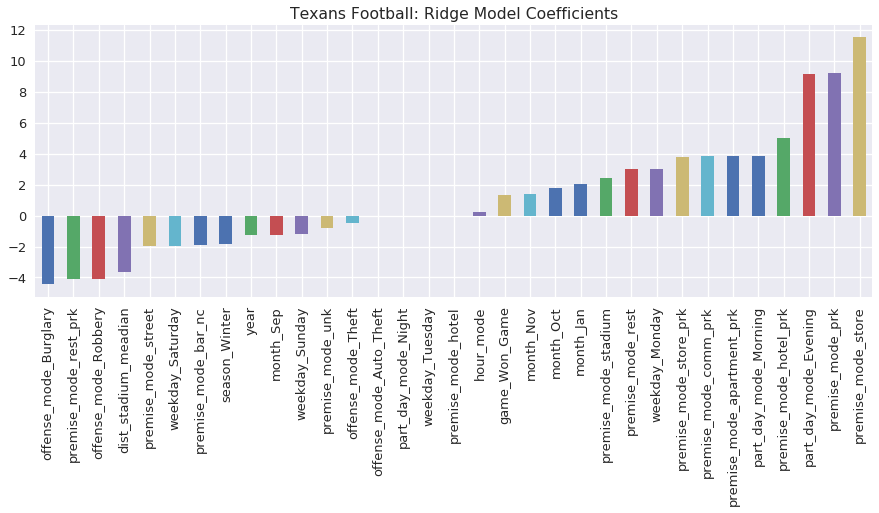


#### Ridge

Model Report

RMSE : 5.195

CV Score : Mean 8.908 | Std 3.495 | Min 4.1 | Max 15.28



#### Lasso

Model Report

RMSE : 6.08

CV Score : Mean 7.667 | Std 3.242 | Min 3.67 | Max 12.71



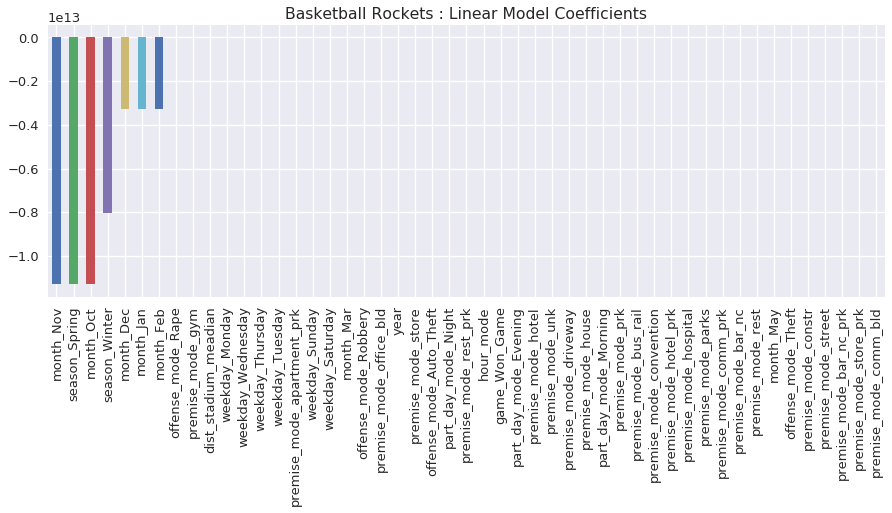
### Basketball: Rockets

#### Linear

Model Report

RMSE : 3.04

CV Score : Mean 2.718e+06 | Std 8.084e+06 | Min 2.718 | Max 2.697e+07

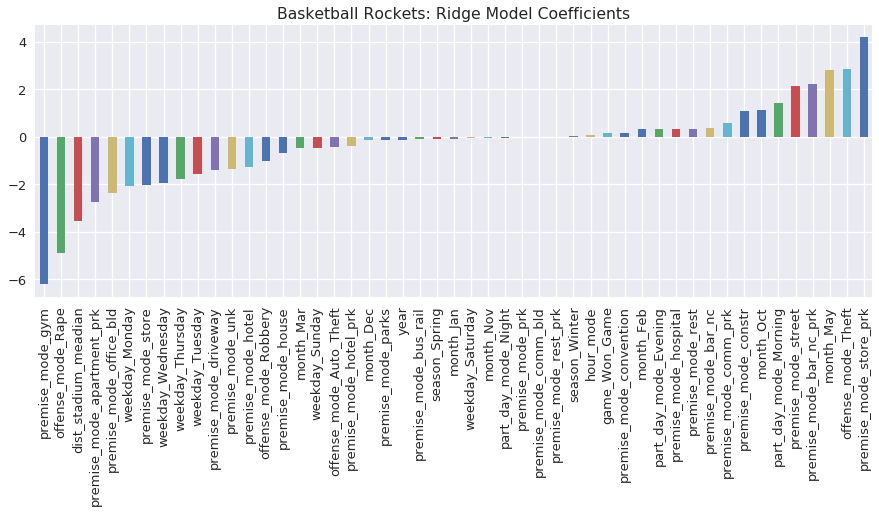


#### Ridge

Model Report

RMSE : 3.079

CV Score : Mean 3.47 | Std 0.7354 | Min 2.501 | Max 5.116

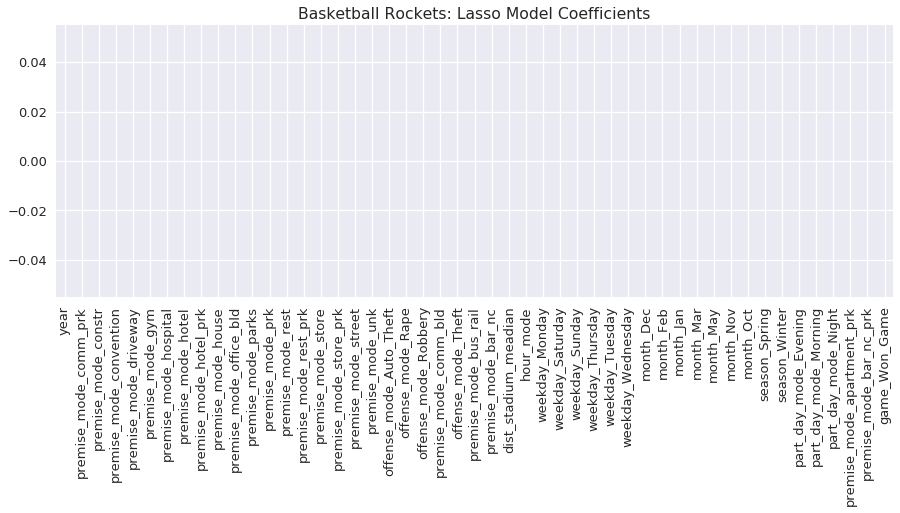


#### Lasso

Model Report

RMSE : 3.907

CV Score : Mean 3.689 | Std 0.8254 | Min 2.687 | Max 5.62



### Baseball: Astros

#### Linear

Model Report

RMSE : 2.811

CV Score : Mean 6.38e+13 | Std 1.081e+14 | Min 2.994 | Max 3.154e+14

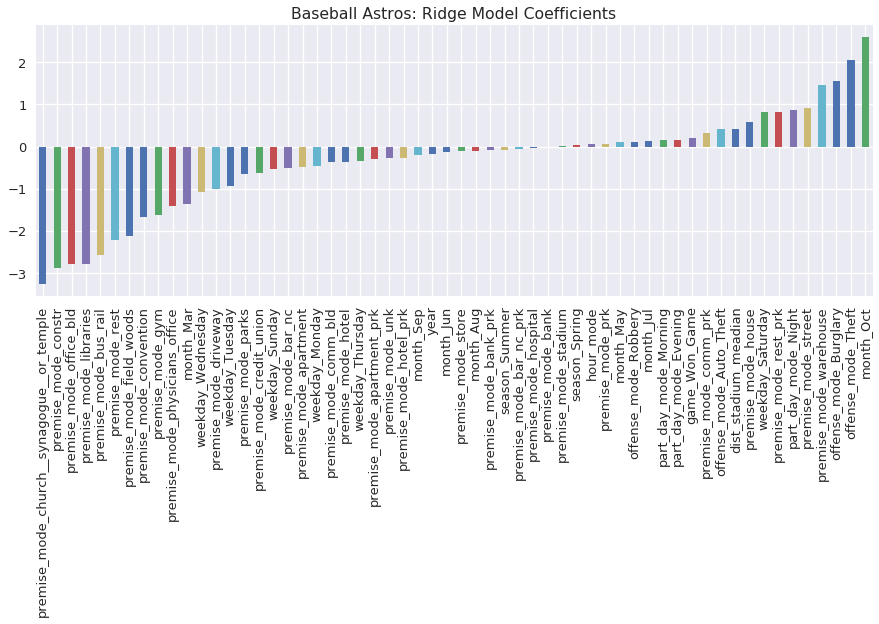


#### Ridge

Model Report

RMSE : 2.818

CV Score : Mean 3.085 | Std 0.3041 | Min 2.538 | Max 3.513

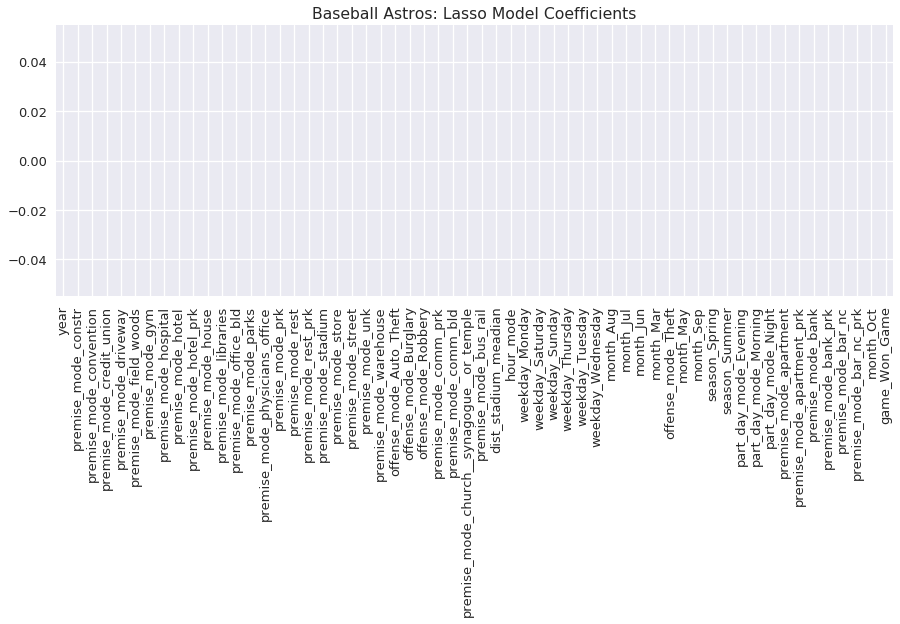


#### Lasso

Model Report

RMSE : 3.231

CV Score : Mean 3.249 | Std 0.3112 | Min 2.772 | Max 3.79



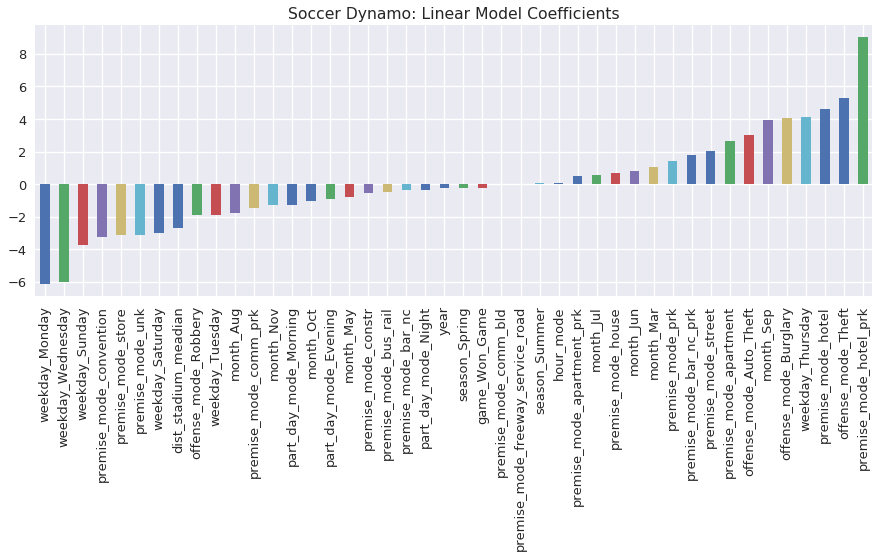
### Soccer: Dynamo

#### Linear

Model Report

RMSE : 2.478

CV Score : Mean 2.521e+13 | Std 4.274e+13 | Min 2.7 | Max 1.355e+14

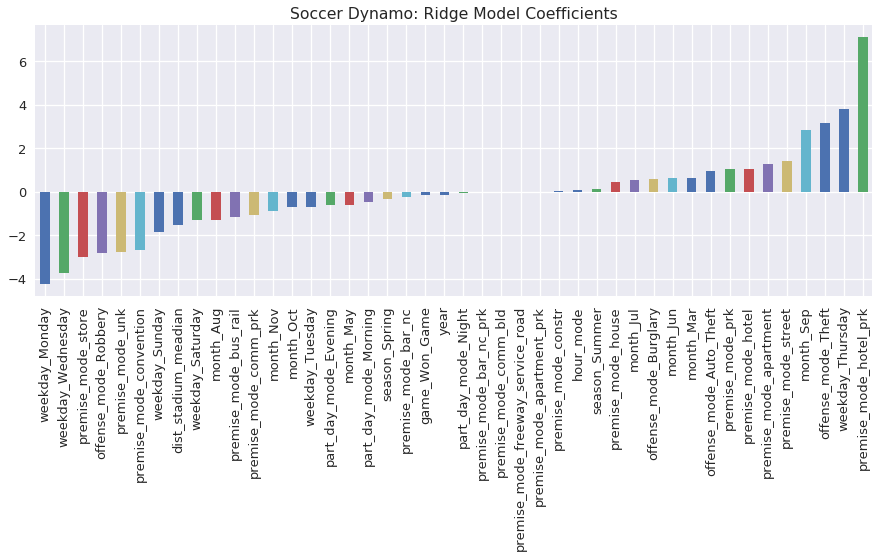


#### Ridge

Model Report

RMSE : 2.554

CV Score : Mean 3.653 | Std 0.9231 | Min 2.467 | Max 5.433



#### Lasso

Model Report

RMSE : 3.515

CV Score : Mean 3.602 | Std 0.8621 | Min 2.05 | Max 5.239

