Sports Analytics - Final Project

Major League Baseball Pitcher Statistics Package

Author: Sikyun (George) Lee

UCLA MSBA

Date: 2020-12-17

Description of MLB-Pitcher-Stats Package

1. Summary of topic and goal

The goal of the MLB-Pitcher-Stats package is to create a group of functions that can quickly measure a pitcher's performance in terms of:

- Innings thrown over games/season (whether recent or all games) to measure a pitcher's usage
- Number of pitches thrown over games/season to measure possible fatigue issues
- Number of batters faced by the pitcher over games/season to evaluate efficiency
- Compare the pitcher to another pitcher in above metrics for an apple-to-apple comparison
- Rank top pitchers in above metrics per team

The purpose of this package and its functions is to help on-field management (i.e.: managers and pitching coaches) quicly track their pitchers' usage status, how they've performed recently (or over the season), and decide whether or not these pitchers should pitch in the next scheduled game.

The data used for this package development is a pitch-by-pitch data along with at-bats, games, and players data posted up from Kaggle, but originally webscraped from http://gd2.mlb.com/components/game/mlb/ (http://gd2.mlb.com/components/game/mlb/).

2. Results of the MLB-Pitcher-Stats Package

The result of the package is a series of functions that manipulate the raw pitch-by-pitch data combined with at-bats, games, and player data to visualize player performance in regards to above metrics.

Some of the key metrics that was used in the package includes:

- Total Innings Thrown by Pitcher
- Average Innings Thrown by Pitcher per Game
- Total Pitches Thrown by Pitcher
- Total Batters Faced by Pitcher
- Average Batters Faced by Pitcher per Inning

These metrics are displayed as numerical figures and interactive visualization plots using the plotly package.

3. In order to improve this package in the future, some of the following have been considered.

1) Create function to automatically webscrape data directly from baseball statistics website such as retrosheet.com and elsewhere.

Currently, the package uses a set of pitch-by-pitch .csv format data that was uploaded in Kaggle Post https://www.kaggle.com/pschale/mlb-pitch-data-20152018?select=2019_pitches.csv (https://www.kaggle.com/pschale/mlb-pitch-data-20152018?select=2019_pitches.csv).

2) Webscrape from website that will remain constant over time

However, there is a slight difference in the column formats of this data compared to the Retro Sheet's pitch-by-pitch data. What's more, the original data used for this package was webscraped from http://gd2.mlb.com/components/game/mlb/ (http://gd2.mlb.com/components/game/mlb/) but currently cannot be accessed. Therefore in order to further improve this package, a webscraping function to webscrape and transform the raw data into a useable dataframe is needed.

4. Next Steps in Package Development

The project was initially to perform Exploratory Data Analysis of pitchers' performance in the 2019 season. However, this soon developed into making a series of functions to evaluate pitchers' performance so that end-users (in this case, managers and pitching coaches) can quickly see their pitching staff's status.

Building off from this package in its current stage, additional metrics such as:

- How many runners does a pitcher allow on-base per game
- How many runners does a pitcher leave on-base (and end the inning) per game
- How many strikes and balls does a pitcher record per game (strike/ball ratio)
- How does the speed of a pitcher's ball change inning to inning and how does the number of on-base events change accordingly

Could be considered to measure further pitcher efficiency. Other considerations in the current package were:

- How can I make it so that entering the pitcher's name instead of pitcher ID is more convenient and accurate?
- Change from entering team's city abbreviation to the actual team name
- Presenting some of the tables in a prettier format that current

5. Understanding Saber-metrics and other baseball expert metrics would have helped.

Baseball analytics is the most advanced analytics among all of sports. There are multiple metrics that have arisen from Saber metrics and traditional metrics such as ERA and Win-Loss have become obsolete due to the introduction of Wins Above Replacement (WAR).

Having a deeper understanding of these metrics and how they are calculated could help in developing new metrics that are a bit more intuitive to the coaching staff and help them make faster on-the-field decisions regarding their pitching staff.

Note: Because this was my first time creating a Python package, I had some slight problems in properly creating the package for Py download/install. Please use the *mlb_pitcher_stats.py* file's functions to run this notebook

Walkthrough of the Package by each line of code chunk

The package consists of multiple functions and I will go over each functions, explaining what the results could mean for the target team coaching staff.

In [10]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

from mlb_pitcher_stats import get_data
from mlb_pitcher_stats import metric_pitches_per_game_per_pitcher
from mlb_pitcher_stats import plot_pitch_trend_per_pitcher
from mlb_pitcher_stats import plot_inning_trend_per_pitcher
from mlb_pitcher_stats import plot_batters_faced_per_pitcher
from mlb_pitcher_stats import plot_compare_innings_batters
from mlb_pitcher_stats import plot_compare_two_pitchers
from mlb_pitcher_stats import plot_top_pitchers_per_team
```

Part 1: Read, manipulate, and return data that will be used for the entire package

This package's functions will use a common dataframe that is returned from the *get_data* function. Therefore, after loading the package, this function should be used first.

In [2]:
```
def get_data():
    #import required packages
    import pandas as pd
    import numpy as np

    #print('This function automatically reads the MLB Pitch by Pitch Data in 2
019 Season when called.')
    #read base datasets -- 2019 MLB Pitch by Pitch Data with At Bats, Games, a
nd Players Data
    atbats = pd.read_csv('2019_atbats.csv')
    games = pd.read_csv('2019_games.csv')
    pitches = pd.read_csv('2019_pitches.csv')
    names = pd.read_csv('player_names.csv')

    #Merge the dataset into an one big pitch by pitch dataset
    joined_atbats = atbats.copy(deep = True)
    joined_atbats = pd.merge(pitches, joined_atbats, how = 'left', on = 'ab_i
d')
    joined_atbats = pd.merge(joined_atbats, games, how = 'left', on ='g_id')
    joined_atbats = pd.merge(joined_atbats, names, how = 'left', left_on = 'pi
tcher_id', right_on = 'id')
    joined_atbats['full_name'] = joined_atbats.first_name + " " + joined_atbat
s.last_name

    #print('-------------------------------------------------')
    #print('When you see THIS text, the data is loaded and other functions in
 this package can be used.')
    #print('To get pitcher statistics, use their pitcher_id in the "player_nam
es.csv" or try ID#: 502239, 547943, 607192 for example.')
    #print("To get a team's pitcher rankings, use the team's abbreviated city
 name such as 'oak' for Oakland A's or 'bos' for Boston Red Sox.")
    return joined_atbats
```

In [5]:
```
get_data()
```

Out[5]:

	px	pz	start_speed	end_speed	spin_rate	spin_dir	break_angle	break_length
0	0.00	2.15	88.8	80.7	placeholder	placeholder	22.8	4.8
1	0.34	2.31	89.9	81.8	placeholder	placeholder	22.8	3.6
2	-0.05	2.03	85.7	79.6	placeholder	placeholder	9.6	6.0
3	0.49	0.92	85.4	78.5	placeholder	placeholder	24.0	7.2
4	-0.13	1.11	84.6	77.6	placeholder	placeholder	26.4	8.4
...	...	...	...	...	...	...	...	...
728785	0.30	1.99	95.8	87.3	placeholder	placeholder	40.8	3.6
728786	1.00	-0.38	87.2	80.3	placeholder	placeholder	7.2	7.2
728787	0.36	2.02	95.0	86.6	placeholder	placeholder	33.6	3.6
728788	-0.26	2.60	84.0	75.3	placeholder	placeholder	2.4	8.4
728789	0.22	1.06	85.8	77.6	placeholder	placeholder	4.8	8.4

728790 rows × 69 columns

Part 2: Get metrics and visualizations for one (1) pitcher

After running the *get_data* function, we are now able to look at various functions in the package. The first few functions are looking at the trends of the pitcher's innings thrown and batters faced.

This idea comes from eCommerce companies' Growth Accounting, where they look at how their customer base is growing over time and how many customers are lapsing over time. Using this concept, I've applied it to draw out how pitcher's innings thrown and batters faced are changing over time to measure how much fatigue they could potentially accrued and how efficient each pitcher is in handling batters.

Pitches per game per pitcher metric

This metric is used to return a table of the pitcher's (by entering its pitcher_id) total pitches thrown (in the nonzero column) over the entire 2019 season.

There are also additional metrics such as

- Avg. pitches thrown per game
- Total pitches thrown by the pitcher

Below is an example of Hyun-Jin Ryu (with the LA Dodgers at the time, but now with TOR Blue Jays in 2020) with pitcher_id: 547943

For other players such as David Price, please use pitcher_id: 456034 or for Bartolo Colon, please use pitcher_id: 112526

```
In [ ]: def metric_pitches_per_game_per_pitcher():
            import pandas as pd
            import numpy as np

            print('This metric returns Pitches thrown by a Pitcher per Game, per Seaso
        n and Avg. Pitches thrown by a Pitcher per Game over a Season')
            print('\n')
            print('Enter Pitcher ID Number and Press Enter: ')
            pitcher_id = int(input())
            joined_atbats = get_data()

            #Search for that specific pitcher by ID
            joined_atbats = joined_atbats[joined_atbats['pitcher_id'] == pitcher_id]
            pitcher_name = joined_atbats[joined_atbats['pitcher_id'] == pitcher_id]['f
        ull_name'].unique()

            #Aggregate to get total pitches thrown by pitcher per game
            total_pitches_pitcher_game = joined_atbats.groupby(by = ['date','pitcher_i
        d'], as_index=False)['batter_id'].agg([np.count_nonzero]).reset_index()
            total_pitches_pitcher_game = total_pitches_pitcher_game.sort_values(['dat
        e','count_nonzero','pitcher_id'], ascending=[True, False,True])
            total_pitches_pitcher_game = total_pitches_pitcher_game.sort_values(['dat
        e','count_nonzero'], ascending=[True, False])

            #Aggregate to get total pitches thrown by pitcher over season
            total_pitches_pitcher_season = total_pitches_pitcher_game.groupby(['pitche
        r_id'], as_index=False)['count_nonzero'].agg([np.sum]).reset_index()
            temp = total_pitches_pitcher_game.groupby(['pitcher_id'], as_index=False)[
        'count_nonzero'].agg([np.sum]).reset_index()
            total_pitches_pitcher_season = temp.sort_values(['sum'], ascending=False)

            #Aggregate to get average pitches thrown by pitcher over season
            total_games_pitcher = total_pitches_pitcher_game['date'].nunique()
            total_pitches_pitcher = total_pitches_pitcher_game['count_nonzero'].agg([n
        p.sum])
            avg_pitches_pitcher = np.round(total_pitches_pitcher / total_games_pitcher
        , decimals = 0, out=None)

            #Return a DataFrame of Average Pitches
            print('---------------------------------------------------------')
            print('For Pitcher ID: ', pitcher_id)
            print('Pitcher Name is ', pitcher_name)
            print('Total Pitches Thrown per Game :')
            print(pd.DataFrame(total_pitches_pitcher_game))
            print('\n')
            print('Total Pitches Thrown per Season :', total_pitches_pitcher_season['s
        um'])
            print('\n')
            print('Average Pitches Thrown per Game :', avg_pitches_pitcher)
            print('---------------------------------------------------------')
```

In [8]:
```
metric_pitches_per_game_per_pitcher()
```

```
This metric returns Pitches thrown by a Pitcher per Game, per Season and Avg.
Pitches thrown by a Pitcher per Game over a Season


Enter Pitcher ID Number and Press Enter:
547943
--------------------------------------------------------
For Pitcher ID:  547943
Pitcher Name is  ['Hyun-Jin Ryu']
Total Pitches Thrown per Game :
          date  pitcher_id  count_nonzero
0   2019-03-28      547943             82
1   2019-04-02      547943             87
2   2019-04-08      547943             33
3   2019-04-20      547943             92
4   2019-04-26      547943            105
5   2019-05-01      547943            107
6   2019-05-07      547943             93
7   2019-05-12      547943            116
8   2019-05-19      547943             88
9   2019-05-25      547943             93
10  2019-05-30      547943            106
11  2019-06-04      547943            104
12  2019-06-10      547943             99
13  2019-06-16      547943             94
14  2019-06-22      547943            107
15  2019-06-28      547943             81
16  2019-07-04      547943             89
17  2019-07-14      547943             94
18  2019-07-19      547943            102
19  2019-07-26      547943            103
20  2019-07-31      547943             80
21  2019-08-11      547943             91
22  2019-08-17      547943            101
23  2019-08-23      547943             94
24  2019-08-29      547943             97
25  2019-09-04      547943             93
26  2019-09-14      547943             90
27  2019-09-22      547943             95
28  2019-09-28      547943             97


Total Pitches Thrown per Season : 0    2713
Name: sum, dtype: int64


Average Pitches Thrown per Game : sum    94.0
Name: count_nonzero, dtype: float64
--------------------------------------------------------
```

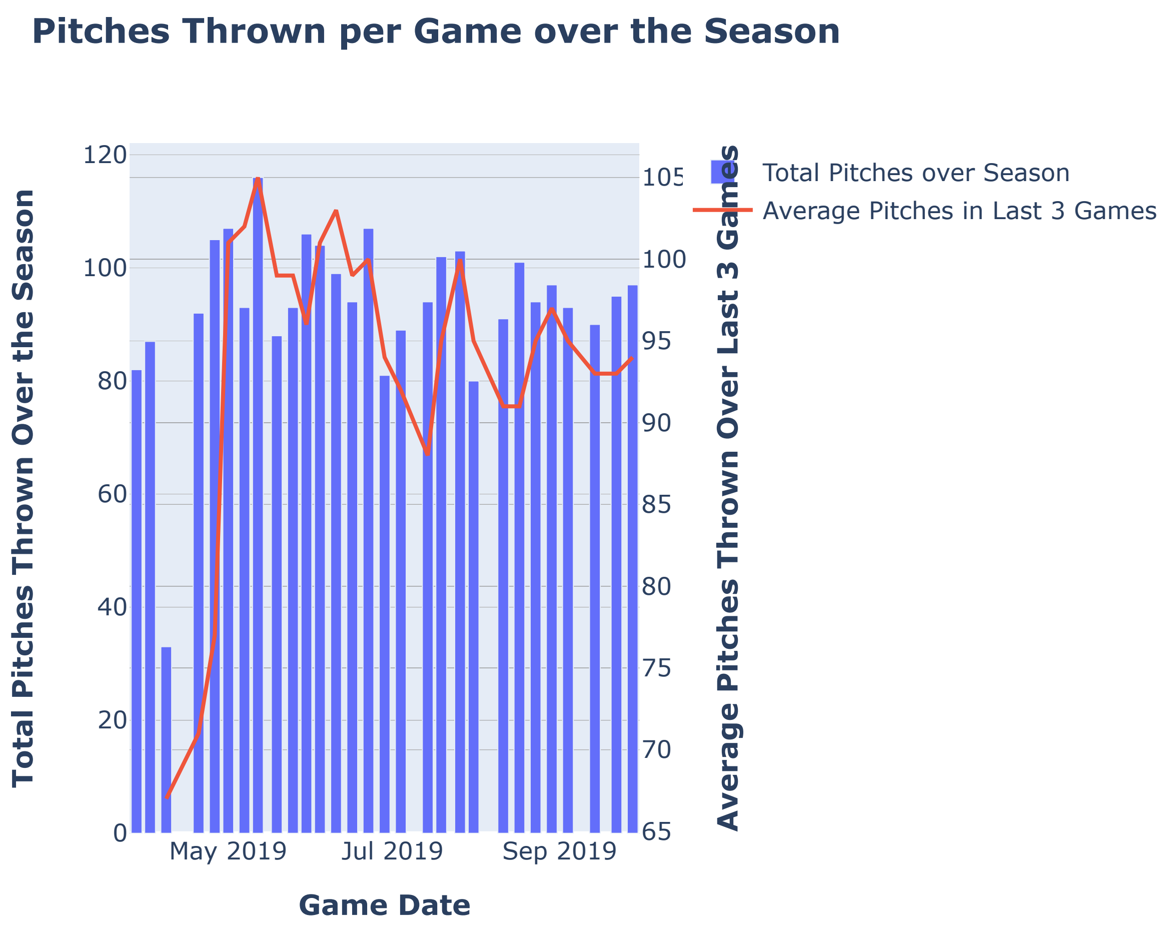
Plot pitch trend per pitcher

Starting from this function, we start to use the Growth Accounting concept mentioned previously to plot how the number pitches thrown by a pitcher have changed over the season and also shows a rolling average of the last 3 games for additional trend analysis.

In [ ]:
```
#plots of the pitches thrown per game for a pitcher

def plot_pitch_trend_per_pitcher():
    import plotly.express as px
    import plotly.graph_objects as go
    from plotly.subplots import make_subplots
    import pandas as pd
    import numpy as np

    print('This metric returns Pitches thrown by a Pitcher per Game, per Seaso
n and Avg. Pitches thrown by a Pitcher per Game over a Season')
    print('\n')
    print('Enter Pitcher ID Number and Press Enter: ')
    pitcher_id = int(input())
    joined_atbats = get_data()

    #Search for that specific pitcher by ID
    joined_atbats = joined_atbats[joined_atbats['pitcher_id'] == pitcher_id]
    pitcher_name = joined_atbats[joined_atbats['pitcher_id'] == pitcher_id]['f
ull_name'].unique()

    #Aggregate to get total pitches thrown by pitcher per game
    total_pitches_pitcher_game = joined_atbats.groupby(by = ['date','pitcher_i
d'], as_index=False)['batter_id'].agg([np.count_nonzero]).reset_index()
    total_pitches_pitcher_game = total_pitches_pitcher_game.sort_values(['dat
e','count_nonzero','pitcher_id'], ascending=[True, False,True])
    total_pitches_pitcher_game = total_pitches_pitcher_game.sort_values(['dat
e','count_nonzero'], ascending=[True, False])

    #Get the Rolling Average of Pitches Thrown by pitcher per game
    total_pitches_pitcher_game['rolling_mean'] = np.round(total_pitches_pitche
r_game['count_nonzero'].rolling(window=3).mean(), decimals = 0)


    #Output: Text and Visualization
    print('---------------------------------------------------------')
    print('Visualization of Total Pitches thrown by ID:', pitcher_id, ' ', pit
cher_name, ' over the Season')
    print('---------------------------------------------------------')

#     fig = px.line(total_pitches_pitcher_game, x="date", y="rolling_mean", ti
tle='Average Pitches Thrown over Last 3 Games')
#     fig = fig.add_bar(total_pitches_pitcher_game, x='date', y='count_nonzer
o', title='Total Pitches Thrown per Game over the Season',
#             hover_data = ['date', 'count_nonzero'], color = 'count_nonzero',
#         labels = {'count_nonzero' : 'Total Pitches Thrown', 'date' : 'Game D
ate'})

    fig = make_subplots(specs = [[{'secondary_y' : True}]])

    fig.add_trace(
        go.Bar(x=total_pitches_pitcher_game['date'], y=total_pitches_pitcher_g
ame['count_nonzero'], name = 'Total Pitches over Season'),
        secondary_y = False,
    )
    fig.add_trace(
        go.Line(x=total_pitches_pitcher_game["date"], y=total_pitches_pitcher_
game["rolling_mean"], name = 'Average Pitches in Last 3 Games'),
        secondary_y = True,
    )

    fig.update_layout(title_text = "<b>Pitches Thrown per Game over the Season
</b>")
    fig.update_xaxes(title_text = '<b>Game Date</b>')
    fig.update_yaxes(title_text = '<b>Total Pitches Thrown Over the Season</b
>', secondary_y = False)
    fig.update_yaxes(title_text = '<b>Average Pitches Thrown Over Last 3 Games
</b>', secondary_y = True)

    fig.show()
```

```
In [11]: plot_pitch_trend_per_pitcher()
```

```
This metric returns Pitches thrown by a Pitcher per Game, per Season and Avg.
Pitches thrown by a Pitcher per Game over a Season


Enter Pitcher ID Number and Press Enter:
547943
---------------------------------------------------------
Visualization of Total Pitches thrown by ID: 547943   ['Hyun-Jin Ryu']  over
the Season
---------------------------------------------------------

C:\Users\sikyu\AppData\Local\Continuum\anaconda3\lib\site-packages\plotly\gra
ph_objs\_deprecations.py:385: DeprecationWarning:

plotly.graph_objs.Line is deprecated.
Please replace it with one of the following more specific types
  - plotly.graph_objs.scatter.Line
  - plotly.graph_objs.layout.shape.Line
  - etc.
```



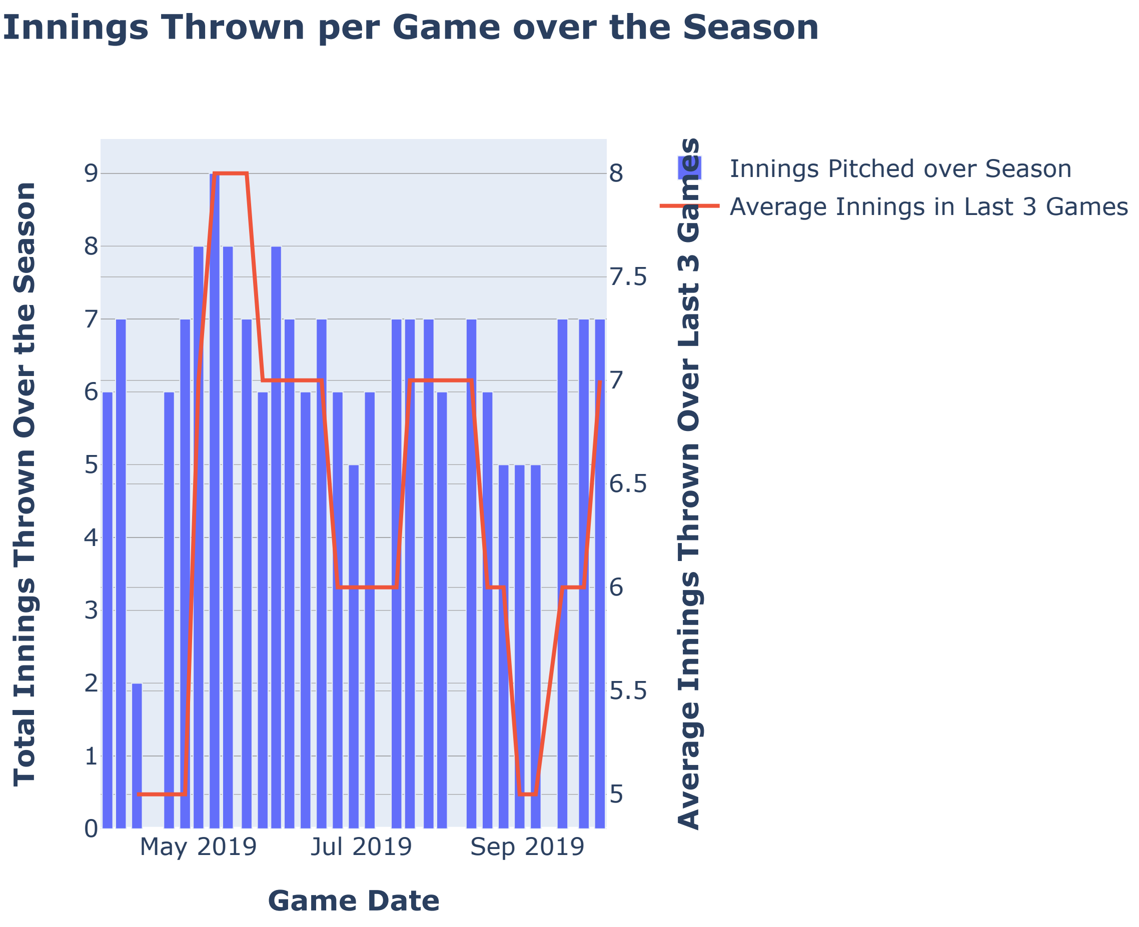
Plot inning trend per pitcher ¶

Next, we plot how the innings thrown by a pitcher have changed over the season and also shows a rolling average of the last 3 games for additional trend analysis.

In [ ]:
```
def plot_inning_trend_per_pitcher():
    import plotly.express as px
    import plotly.graph_objects as go
    from plotly.subplots import make_subplots
    import pandas as pd
    import numpy as np

    print('This metric returns innings thrown by a Pitcher per Game, per Seaso
n and Avg. innings thrown by a Pitcher per Game over a Season')
    print('\n')
    print('Enter Pitcher ID Number and Press Enter: ')
    pitcher_id = int(input())
    joined_atbats = get_data()

    #Search for that specific pitcher by ID
    joined_atbats = joined_atbats[joined_atbats['pitcher_id'] == pitcher_id]
    pitcher_name = joined_atbats[joined_atbats['pitcher_id'] == pitcher_id]['f
ull_name'].unique()

    #Aggregate to get innings information for pitcher_id
    #Get innings as a column
    all_games_innings = joined_atbats.groupby(by=['g_id'], as_index=False)['in
ning'].nunique().reset_index()
    #Get unique game dates that the pitcher pitched as a column
    all_games_dates = joined_atbats.groupby(by=['g_id'], as_index=False)['dat
e'].agg([np.unique]).reset_index()
    #Concatenate into one Pandas dataframe, drop the index column
    combined_innings = pd.concat([all_games_dates, all_games_innings], axis=1)
.drop(columns=['index'])
    combined_innings = combined_innings.rename(columns = {'unique': 'date'})

    #Get some performance metrics
    #Total innings pitched
    total_innings = all_games_innings.sum()
    #Average innings pitched
    avg_innings = all_games_innings.mean()
    #Get Rolling Average of last 3 games pitched innings
    combined_innings['rolling_mean'] = np.round(combined_innings['inning'].rol
ling(window=3).mean(), decimals = 0)


    #Output: Text and Visualization
    print('---------------------------------------------------------')
    print('Visualization of Innings Pitched by ID:', pitcher_id, ' ', pitcher_
name, ' over the Season')
    print('Pitcher ', pitcher_name, 'Pitched a Total of ', total_innings, ' an
d an Average of ', avg_innings, ' over the Season')
    print('---------------------------------------------------------')

    fig = make_subplots(specs = [[{'secondary_y' : True}]])

    fig.add_trace(
        go.Bar(x=combined_innings['date'], y=combined_innings['inning'], name
= 'Innings Pitched over Season'),
        secondary_y = False,
    )
    fig.add_trace(
        go.Line(x=combined_innings["date"], y=combined_innings["rolling_mean"
], name = 'Average Innings in Last 3 Games'),
        secondary_y = True,
    )

    fig.update_layout(title_text = "<b>Innings Thrown per Game over the Season
</b>")
    fig.update_xaxes(title_text = '<b>Game Date</b>')
    fig.update_yaxes(title_text = '<b>Total Innings Thrown Over the Season</b
>', secondary_y = False)
    fig.update_yaxes(title_text = '<b>Average Innings Thrown Over Last 3 Games
</b>', secondary_y = True)

    fig.show()
```

In [12]:
```
plot_inning_trend_per_pitcher()
```

```
This metric returns innings thrown by a Pitcher per Game, per Season and Avg.
innings thrown by a Pitcher per Game over a Season


Enter Pitcher ID Number and Press Enter:
547943
--------------------------------------------------------
Visualization of Innings Pitched by ID: 547943   ['Hyun-Jin Ryu']  over the S
eason
Pitcher  ['Hyun-Jin Ryu'] Pitched a Total of  index     406
inning    187
dtype: int64  and an Average of  index     14.000000
inning     6.448276
dtype: float64  over the Season
--------------------------------------------------------

C:\Users\sikyu\AppData\Local\Continuum\anaconda3\lib\site-packages\plotly\gra
ph_objs\_deprecations.py:385: DeprecationWarning:

plotly.graph_objs.Line is deprecated.
Please replace it with one of the following more specific types
  - plotly.graph_objs.scatter.Line
  - plotly.graph_objs.layout.shape.Line
  - etc.
```



Analysis of Hyun-Jin Ryu (ID: 547943)

In case of Ryu, both of this innings thrown and pitches thrown peaked just before summer and started to see a decrease in the summer and stabilized as the season progressed into the fall.

This is reasonable since many starting pitchers start to lose stamina over the hot summer season and the coaching staff start to curtail pitcher's innings and/or pitches to keep them performing strong into the fall season.

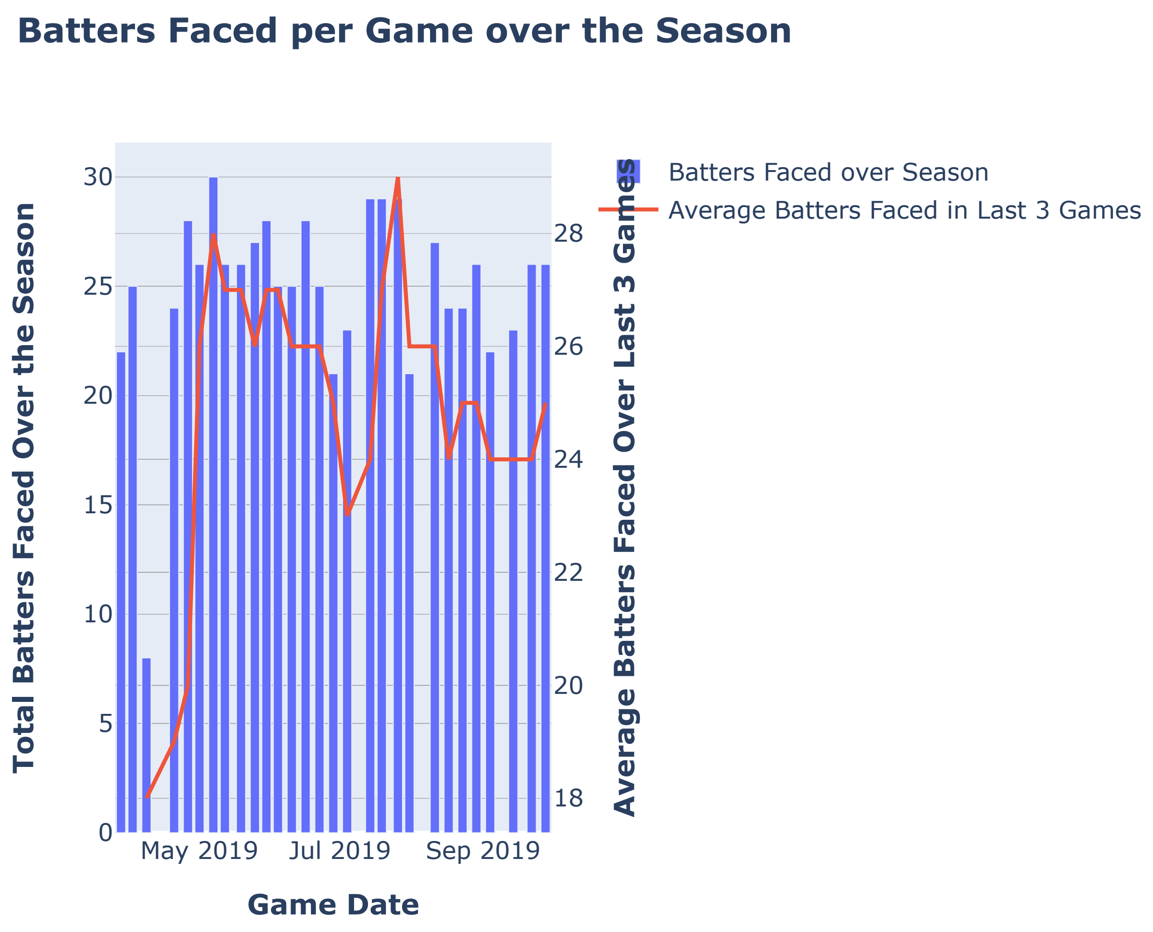
Plot batters faced per pitcher

Next, we plot how many batters faced by a pitcher have changed over the season and also shows a rolling average of the last 3 games for additional trend analysis.

In [ ]:
```
def plot_batters_faced_per_pitcher():
    import plotly.express as px
    import plotly.graph_objects as go
    from plotly.subplots import make_subplots
    import pandas as pd
    import numpy as np

    print('This metrics return Batters faced by a Pitcher per Game, per Season
and Avg. batters faced by a Pitcher per Game over a Season')
    print('\n')
    print('Enter Pitcher ID Number and Press Enter: ')
    pitcher_id = int(input())
    joined_atbats = get_data()

    #Search for that specific pitcher by ID
    pitcher_data = joined_atbats[joined_atbats['pitcher_id'] == pitcher_id] #t
est
    pitcher_name = joined_atbats[joined_atbats['pitcher_id'] == pitcher_id]['f
ull_name'].unique()

    #Get aggregated table of batters faced per game
    innings_num = pitcher_data.groupby(by=['g_id'], as_index=False)['inning'].
nunique().reset_index() #test1
    games_num = pitcher_data.groupby(by=['g_id'], as_index=False)['date'].agg
([np.unique]).reset_index() #test2
    c1 = pd.concat([games_num, innings_num], axis=1).drop(columns=['index'])
    batters_num = pitcher_data.groupby(by=['g_id','inning'], as_index=True)['b
atter_id'].nunique().reset_index() #test3
    c2 = batters_num.groupby(by=['g_id'], as_index=True)['batter_id'].sum().re
set_index() #25, 21
    combined_data = pd.merge(c1, c2,how='left', on = 'g_id')
    combined_data = combined_data.rename(columns = {'unique': 'date', 'batter_
id': 'batters_faced'})

    #Get some performance metrics
    #Total innings pitched
    total_innings = combined_data['inning'].sum()
    #Average innings pitched
    avg_innings = combined_data['inning'].mean()
    #Get Rolling Average of last 3 games pitched innings
    combined_data['rolling_mean_innings'] = np.round(combined_data['inning'].r
olling(window=3).mean(), decimals = 0)

    #Total batters faced
    total_batters = combined_data['batters_faced'].sum()
    #Average batters faced per game
    avg_batters = combined_data['batters_faced'].mean()
    #Get Rolling Average of last 3 games' batters faced number
    combined_data['rolling_mean_batters_faced'] = np.round(combined_data['batt
ers_faced'].rolling(window=3).mean(), decimals = 0)

    #Average batters faced per inning
    avg_batters_inning = np.round(total_batters / total_innings, decimals = 0)

    #Output: Text and Visualization
    print('---------------------------------------------------------')
    print('Visualization of Batters faced by ID:', pitcher_id, ' ', pitcher_na
me, ' over the Season')
    print('Pitcher ', pitcher_name, 'Pitched a Total of ', total_innings, ' in
nings and an Average of ', avg_innings, ' innings over the Season')
    print('Pitcher ', pitcher_name, 'Faced a Total of ', total_batters, ' batt
ers and an Average of ', avg_batters, ' batters over the Season')
    print('Pitcher ', pitcher_name, 'Faces an Average of ', avg_batters_inning
, ' batters per inning on a given game')
    print('--------------------------------------------------------')

    fig = make_subplots(specs = [[{'secondary_y' : True}]])

    fig.add_trace(
        go.Bar(x=combined_data['date'], y=combined_data['batters_faced'], name
= 'Batters Faced over Season'),
        secondary_y = False,
    )

    fig.add_trace(
        go.Line(x=combined_data["date"], y=combined_data["rolling_mean_batters
_faced"], name = 'Average Batters Faced in Last 3 Games'),
        secondary_y = True,
    )

    fig.update_layout(title_text = "<b>Batters Faced per Game over the Season
</b>")
    fig.update_xaxes(title_text = '<b>Game Date</b>')
    fig.update_yaxes(title_text = '<b>Total Batters Faced Over the Season</b>'
, secondary_y = False)
    fig.update_yaxes(title_text = '<b>Average Batters Faced Over Last 3 Games
</b>', secondary_y = True)

    fig.show()
```

```
In [13]: plot_batters_faced_per_pitcher()
```

```
This metrics return Batters faced by a Pitcher per Game, per Season and Avg.
batters faced by a Pitcher per Game over a Season


Enter Pitcher ID Number and Press Enter:
547943
--------------------------------------------------------
Visualization of Batters faced by ID: 547943   ['Hyun-Jin Ryu']  over the Sea
son
Pitcher  ['Hyun-Jin Ryu'] Pitched a Total of  187  innings and an Average of
6.448275862068965  innings over the Season
Pitcher  ['Hyun-Jin Ryu'] Faced a Total of  723  batters and an Average of  2
4.93103448275862  batters over the Season
Pitcher  ['Hyun-Jin Ryu'] Faces an Average of  4.0  batters per inning on a g
iven game
--------------------------------------------------------
```



Analysis of Hyun-Jin Ryu (547943)

Compared to a stabilization of pitches and innings starting from the summer, Ryu has seen an occassional spikes in the number of batters faced. This is probably due Ryu facing more batters in certain games than in most games.

Plot compare innings batters

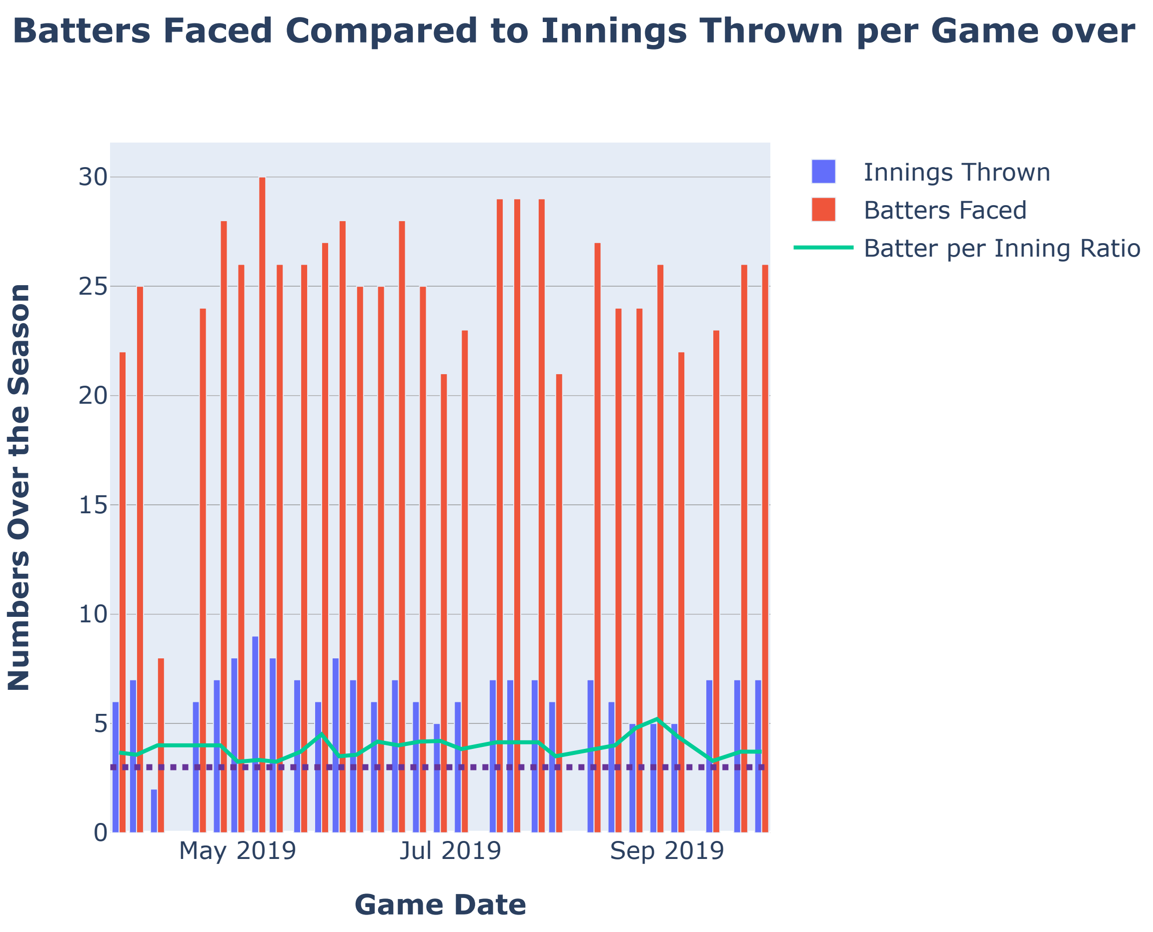
This looks at 3 components; innings thrown, batters faced, and batters faced per inning.

These metrics and visualization (including the 3.0 batters faced per inning threshold in green) helps the coaching staff see if their pitcher is performing consistently or starting to lose stability.

In [ ]:
```
#Comparing Innings Faced VS Batters Faced
def plot_compare_innings_batters():
    import plotly.express as px
    import plotly.graph_objects as go
    from plotly.subplots import make_subplots
    import pandas as pd
    import numpy as np

    print('This metrics return Batters faced by a Pitcher per Game, per Season
and Innings thrown by a Pitcher per Game over a Season')
    print('\n')
    print('Enter Pitcher ID Number and Press Enter: ')
    pitcher_id = int(input())
    joined_atbats = get_data()

    #Search for that specific pitcher by ID
    pitcher_data = joined_atbats[joined_atbats['pitcher_id'] == pitcher_id] #t
est
    pitcher_name = joined_atbats[joined_atbats['pitcher_id'] == pitcher_id]['f
ull_name'].unique()

    #Get aggregated table of batters faced per game
    innings_num = pitcher_data.groupby(by=['g_id'], as_index=False)['inning'].
nunique().reset_index() #test1
    games_num = pitcher_data.groupby(by=['g_id'], as_index=False)['date'].agg
([np.unique]).reset_index() #test2
    c1 = pd.concat([games_num, innings_num], axis=1).drop(columns=['index'])
    batters_num = pitcher_data.groupby(by=['g_id','inning'], as_index=True)['b
atter_id'].nunique().reset_index() #test3
    c2 = batters_num.groupby(by=['g_id'], as_index=True)['batter_id'].sum().re
set_index() #25, 21
    combined_data = pd.merge(c1, c2,how='left', on = 'g_id')
    combined_data = combined_data.rename(columns = {'unique': 'date', 'batter_
id': 'batters_faced'})
    combined_data['ratio'] = np.round(combined_data['batters_faced'] / combine
d_data['inning'], decimals = 2)

    #Get some performance metrics
    #Total innings pitched
    total_innings = combined_data['inning'].sum()
    #Average innings pitched
    avg_innings = combined_data['inning'].mean()
    #Get Rolling Average of last 3 games pitched innings
    combined_data['rolling_mean_innings'] = np.round(combined_data['inning'].r
olling(window=3).mean(), decimals = 0)

    #Total batters faced
    total_batters = combined_data['batters_faced'].sum()
    #Average batters faced per game
    avg_batters = combined_data['batters_faced'].mean()
    #Get Rolling Average of last 3 games' batters faced number
    combined_data['rolling_mean_batters_faced'] = np.round(combined_data['batt
ers_faced'].rolling(window=3).mean(), decimals = 0)

    #Average batters faced per inning
    avg_batters_inning = np.round(total_batters / total_innings, decimals = 0)
    #Output: Text and Visualization
    #Text
    print('---------------------------------------------------------')
    print('Visualization of Batters faced by ID:', pitcher_id, ' ', pitcher_na
me, ' over the Season')
    print('Pitcher ', pitcher_name, 'Pitched a Total of ', total_innings, ' in
nings and an Average of ', np.round(avg_innings, decimals=1), ' innings over t
he Season')
    print('Pitcher ', pitcher_name, 'Faced a Total of ', total_batters, ' batt
ers and an Average of ', np.round(avg_batters, decimals=0), ' batters over the
Season')
    print('Pitcher ', pitcher_name, 'Faces an Average of ', np.round(avg_batte
rs_inning, decimals=1), ' batters per inning on a given game')
    print('--------------------------------------------------------')

    #Visualization
    plot = go.Figure(data=[go.Bar(
        name = 'Innings Thrown',
        x = combined_data['date'],
        y = combined_data["inning"]
       ),
                           go.Bar(
        name = 'Batters Faced',
        x = combined_data['date'],
        y = combined_data['batters_faced']
       ),
                           go.Line(
        name = 'Batter per Inning Ratio',
        x = combined_data['date'],
        y = combined_data['ratio']
       ),
    ])
    plot.add_shape(type = "line", line_color = "RebeccaPurple", line_width = 3
, opacity = 1, line_dash = "dot",
                  x0=0, x1=1, xref= "paper", y0=3, y1=3, yref="y")
    plot.update_layout(title_text = "<b>Batters Faced Compared to Innings Thro
wn per Game over the Season</b>")
    plot.update_xaxes(title_text = '<b>Game Date</b>')
    plot.update_yaxes(title_text = '<b>Numbers Over the Season</b>')

    plot.show()
```

```
In [14]: plot_compare_innings_batters()
```

```
This metrics return Batters faced by a Pitcher per Game, per Season and Innin
gs thrown by a Pitcher per Game over a Season


Enter Pitcher ID Number and Press Enter:
547943
--------------------------------------------------------
Visualization of Batters faced by ID: 547943   ['Hyun-Jin Ryu']  over the Sea
son
Pitcher  ['Hyun-Jin Ryu'] Pitched a Total of  187  innings and an Average of
6.4  innings over the Season
Pitcher  ['Hyun-Jin Ryu'] Faced a Total of  723  batters and an Average of  2
5.0  batters over the Season
Pitcher  ['Hyun-Jin Ryu'] Faces an Average of  4.0  batters per inning on a g
iven game
--------------------------------------------------------
```



Analysis of Hyun-Jin Ryu (547943)

Overall, Ryu's innings and batters faced totals seem to be consistent. One interesting point from this chart is that Ryu never seems to exactly hit the 3.0 batters faced per inning, which means that Ryu often faces more than 3 batters per inning due to either:

- The batters get on base with a hit
- The batters get on base with a non-hit event (e.g.: walks, not-outs, etc.)

Ryu didn't have a high walk rate, which means that he probably gave up a hit or two per inning.

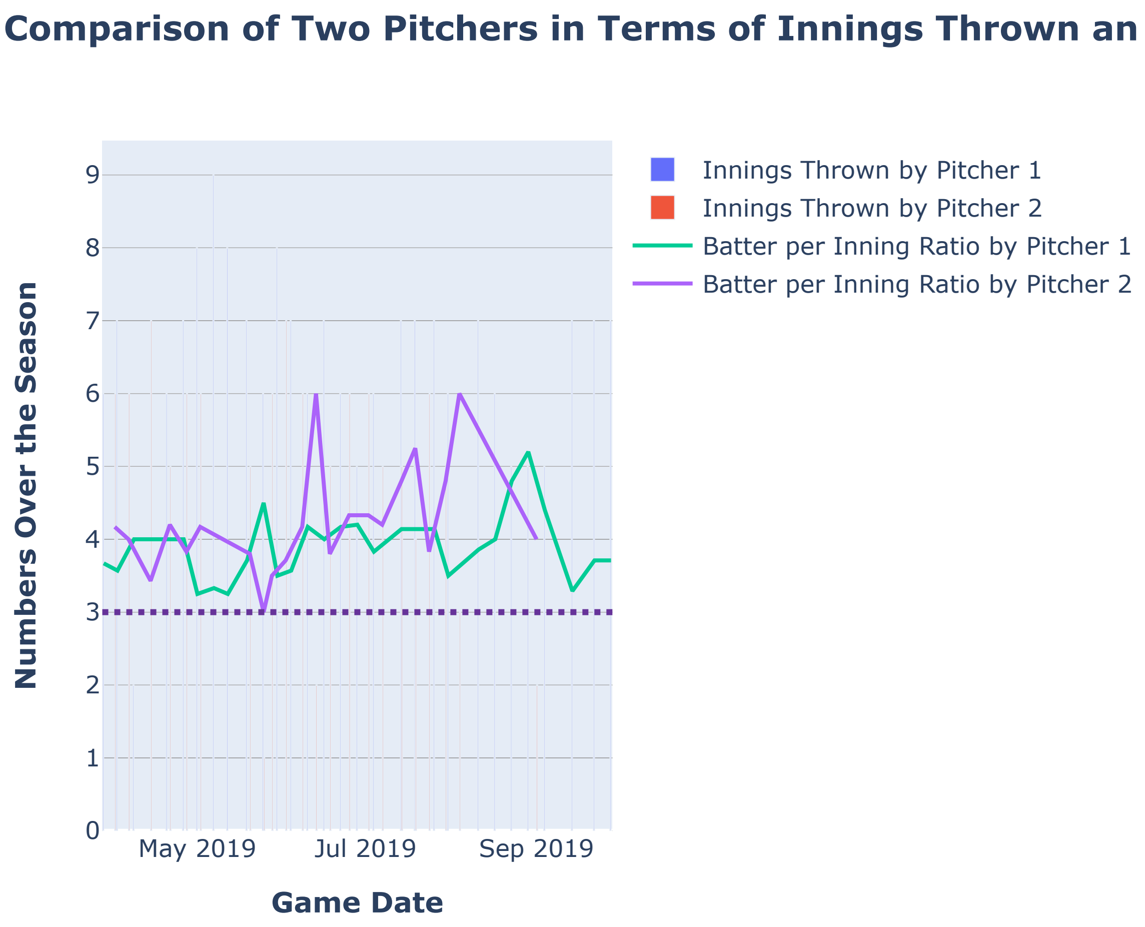
Plot compare two pitchers

This function compares two pitcher's performance in one visualization chart with the above metrics.

In [ ]:
```
#Compare two pitchers in terms of batters faced and innings thrown
def plot_compare_two_pitchers():
    import plotly.express as px
    import plotly.graph_objects as go
    from plotly.subplots import make_subplots
    import pandas as pd
    import numpy as np

    print('This metrics return a comparison of two pitchers in terms of batter
s faced and innings thrown over a season')
    print('\n')
    print('Enter First Pitcher ID Number and Press Enter: ')
    pitcher_id1 = int(input())
    print('Enter Second Pitcher ID Number and Press Enter: ')
    pitcher_id2 = int(input())

    joined_atbats = get_data()

    #Search for that specific pitcher by ID
    pitcher_data1 = joined_atbats[joined_atbats['pitcher_id'] == pitcher_id1]
#test
    pitcher_data2 = joined_atbats[joined_atbats['pitcher_id'] == pitcher_id2]
#test
    pitcher_name1 = joined_atbats[joined_atbats['pitcher_id'] == pitcher_id1][
'full_name'].unique()
    pitcher_name2 = joined_atbats[joined_atbats['pitcher_id'] == pitcher_id2][
'full_name'].unique()

    #Aggregate data by games and innings and dates
    innings_num1 = pitcher_data1.groupby(by=['g_id'], as_index=False)['inning'
].nunique().reset_index() #test1
    games_num1 = pitcher_data1.groupby(by=['g_id'], as_index=False)['date'].ag
g([np.unique]).reset_index() #test2
    c1 = pd.concat([games_num1, innings_num1], axis=1).drop(columns=['index'])
    batters_num1 = pitcher_data1.groupby(by=['g_id','inning'], as_index=True)[
'batter_id'].nunique().reset_index() #test3
    c2 = batters_num1.groupby(by=['g_id'], as_index=True)['batter_id'].sum().r
eset_index() #25, 21
    combined_data1 = pd.merge(c1, c2,how='left', on = 'g_id')
    combined_data1 = combined_data1.rename(columns = {'unique': 'date', 'batte
r_id': 'batters_faced'})
    combined_data1['ratio'] = np.round(combined_data1['batters_faced'] / combi
ned_data1['inning'], decimals = 2)

    #Get aggregated table of batters faced per game for Pitcher1
    innings_num1 = pitcher_data1.groupby(by=['g_id'], as_index=False)['inning'
].nunique().reset_index() #test1
    games_num1 = pitcher_data1.groupby(by=['g_id'], as_index=False)['date'].ag
g([np.unique]).reset_index() #test2
    c1 = pd.concat([games_num1, innings_num1], axis=1).drop(columns=['index'])
    batters_num1 = pitcher_data1.groupby(by=['g_id','inning'], as_index=True)[
'batter_id'].nunique().reset_index() #test3
    c2 = batters_num1.groupby(by=['g_id'], as_index=True)['batter_id'].sum().r
eset_index() #25, 21
    combined_data1 = pd.merge(c1, c2,how='left', on = 'g_id')
    combined_data1 = combined_data1.rename(columns = {'unique': 'date', 'batte
r_id': 'batters_faced'})
    combined_data1['ratio'] = np.round(combined_data1['batters_faced'] / combi
ned_data1['inning'], decimals = 2)

    #Get aggregated table of batters faced per game for Pitcher2
    innings_num2 = pitcher_data2.groupby(by=['g_id'], as_index=False)['inning'
].nunique().reset_index() #test1
    games_num2 = pitcher_data2.groupby(by=['g_id'], as_index=False)['date'].ag
g([np.unique]).reset_index() #test2
    c3 = pd.concat([games_num2, innings_num2], axis=1).drop(columns=['index'])
    batters_num2 = pitcher_data2.groupby(by=['g_id','inning'], as_index=True)[
'batter_id'].nunique().reset_index() #test3
    c4 = batters_num2.groupby(by=['g_id'], as_index=True)['batter_id'].sum().r
eset_index() #25, 21
    combined_data2 = pd.merge(c3, c4,how='left', on = 'g_id')
    combined_data2 = combined_data2.rename(columns = {'unique': 'date', 'batte
r_id': 'batters_faced'})
    combined_data2['ratio'] = np.round(combined_data2['batters_faced'] / combi
ned_data2['inning'], decimals = 2)

    #Get some performance metrics
    #Pitcher1
    #Total innings pitched
    total_innings1 = combined_data1['inning'].sum()
    #Average innings pitched
    avg_innings1 = combined_data1['inning'].mean()
    #Get Rolling Average of last 3 games pitched innings
    combined_data1['rolling_mean_innings'] = np.round(combined_data1['inning']
.rolling(window=3).mean(), decimals = 0)

    #Total batters faced
    total_batters1 = combined_data1['batters_faced'].sum()
    #Average batters faced per game
    avg_batters1 = combined_data1['batters_faced'].mean()
    #Get Rolling Average of last 3 games' batters faced number
    combined_data1['rolling_mean_batters_faced'] = np.round(combined_data1['ba
tters_faced'].rolling(window=3).mean(), decimals = 0)

    #Average batters faced per inning
    avg_batters_inning1 = np.round(total_batters1 / total_innings1, decimals =
0)

    #Get some performance metrics
    #Pitcher2
    #Total innings pitched
    total_innings2 = combined_data2['inning'].sum()
    #Average innings pitched
    avg_innings2 = combined_data2['inning'].mean()
    #Get Rolling Average of last 3 games pitched innings
    combined_data2['rolling_mean_innings'] = np.round(combined_data2['inning']
.rolling(window=3).mean(), decimals = 0)

    #Total batters faced
    total_batters2 = combined_data2['batters_faced'].sum()
    #Average batters faced per game
    avg_batters2 = combined_data2['batters_faced'].mean()
    #Get Rolling Average of last 3 games' batters faced number
    combined_data2['rolling_mean_batters_faced'] = np.round(combined_data2['ba
tters_faced'].rolling(window=3).mean(), decimals = 0)

    #Average batters faced per inning
    avg_batters_inning2 = np.round(total_batters2 / total_innings2, decimals =
0)

    #Output: Text and Visualization
    #Text
    print('--------------------------------------------------------')
    print('Comparing Two Pitchers in terms of Innings Thrown and Batters Faced
per Game over a Season')
    compare = pd.DataFrame(columns = ['Pitcher ID', 'Pitcher Name', 'Total Inn
ings', 'Avg. Innings',
                                     'Total Batters Faced', 'Avg. Batters Face
d'],
                          data = [[pitcher_id1, pitcher_name1, total_innings1,
avg_innings1,
                                  total_batters1, avg_batters1],
                                 [pitcher_id2, pitcher_name2, total_innings2,
avg_innings2,
                                 total_batters2, avg_batters2]])
    print('--------------------------------------------------------')

    #Visualization
    plot = go.Figure(data=[go.Bar(
        name = 'Innings Thrown by Pitcher 1',
        x = combined_data1['date'],
        y = combined_data1["inning"],
        offsetgroup = 0,
       ),
                           go.Bar(
        name = 'Innings Thrown by Pitcher 2',
        x = combined_data2['date'],
        y = combined_data2['inning'],
        offsetgroup = 1,
       ),
                           go.Line(
        name = 'Batter per Inning Ratio by Pitcher 1',
        x = combined_data1['date'],
        y = combined_data1['ratio']
       ),
                            go.Line(
        name = 'Batter per Inning Ratio by Pitcher 2',
        x = combined_data2['date'],
        y = combined_data2['ratio']
       ),
    ])
    plot.add_shape(type = "line", line_color = "RebeccaPurple", line_width = 3
, opacity = 1, line_dash = "dot",
                  x0=0, x1=1, xref= "paper", y0=3, y1=3, yref="y")
    plot.update_layout(title_text = "<b>Comparison of Two Pitchers in Terms of
Innings Thrown and Batters Faced</b>")
    plot.update_xaxes(title_text = '<b>Game Date</b>')
    plot.update_yaxes(title_text = '<b>Numbers Over the Season</b>')

    return compare, plot.show()
```

```
In [15]: plot_compare_two_pitchers() # 502239,  547943 # 607192, 502239
```

```
This metrics return a comparison of two pitchers in terms of batters faced an
d innings thrown over a season


Enter First Pitcher ID Number and Press Enter:
547943
Enter Second Pitcher ID Number and Press Enter:
456034
--------------------------------------------------------
Comparing Two Pitchers in terms of Innings Thrown and Batters Faced per Game
over a Season
--------------------------------------------------------
```



```
Out[15]: (   Pitcher ID    Pitcher Name  Total Innings  Avg. Innings  \
          0      547943  [Hyun-Jin Ryu]            187      6.448276
          1      456034   [David Price]            110      5.000000

             Total Batters Faced  Avg. Batters Faced
          0                  723           24.931034
          1                  458           20.818182  , None)
```

Analysis between HJ Ryu VS David Price

Ryu as Pitcher 1 and Price as Pitcher 2.

Based on the innings thrown, Ryu seems to have pitched a slightly bit longer in games than Price in 2019.

Surprisingly, Price tends to have a higher batter per inning ratio than Ryu in 2019. This could be reasoned with the fact that Price plays in a more competitive AL East division and has to face stronger batters more often than Ryu.

Plot top pitchers per team

This function allows managers to quickly look at which pitcher has accrued the highest number of innings and lost substantial stamina from facing batters.

The outcome should show 3 charts of top 5 pitchers with:

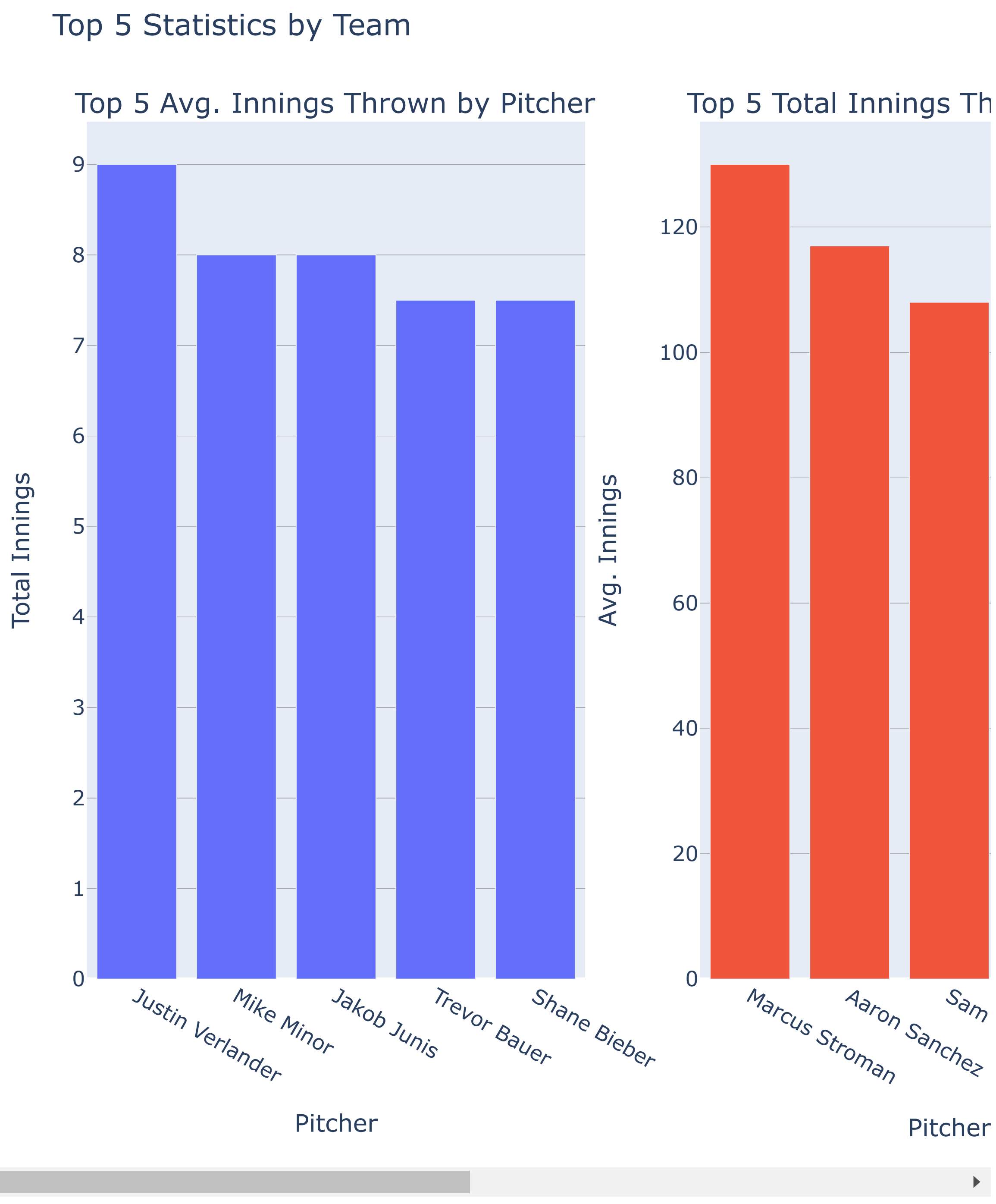
- Avg. Innings thrown
- Total Innings thrown
- Batters faced per Inning

Note: To get Toronto Blue Jays, I've entered "tor." To get other teams such the Red Sox, enter "bos" and "oak" for Oakland A's. If the abbreviation is entered incorrectly, this will run into an error and exception handling hasn't been addressed yet.

In [ ]:
```
def plot_top_pitchers_per_team():
    import plotly.express as px
    import plotly.graph_objects as go
    from plotly.subplots import make_subplots
    import pandas as pd
    import numpy as np

    print('This metrics return a comparison of top 5 pitchers in terms of batt
ers faced and innings thrown over a season per team')
    print('\n')
    print('Enter a team by its abbreviation (e.g.: "oak" for Oakland Athletics
and "bos" for Boston Red Sox)')
    team = input()

    joined_atbats = get_data()


    teams_home = joined_atbats[joined_atbats['home_team'] == team] #e.g.: "oa
k"
    teams_away = joined_atbats[joined_atbats['away_team'] == team] #e.g.: "oa
k"
    teams_comb = pd.concat([teams_home, teams_away], axis = 0)

    #Get total innings thrown by pitcher
    total_innings = teams_comb.groupby(by = ['full_name', 'g_id'], as_index =
True)['inning'].nunique().reset_index().sort_values('inning', ascending=False)
    total_innings = total_innings.groupby(by = ['full_name'], as_index = True)
['inning'].sum().reset_index().sort_values('inning', ascending=False)
    #total_innings.head()

    #Get total games played by pitcher
    total_games = teams_comb.groupby(by = ['full_name'], as_index = True)['g_i
d'].nunique().reset_index().sort_values('g_id', ascending=False)
    #total_games.head()

    #Get total batters faced by pitcher
    total_batters_faced = teams_comb.groupby(by = ['full_name'], as_index = Tr
ue)['batter_id'].nunique().reset_index().sort_values('batter_id', ascending=Fa
lse)
    #total_batters_faced.head()

    #Combine into one datafrme
    combined_df = pd.merge(total_innings, total_games, how='left', on='full_na
me')
    combined_df = pd.merge(combined_df, total_batters_faced, how='left', on='f
ull_name')
    combined_df = combined_df.rename(columns = {'g_id': 'games_played', 'innin
g': 'innings_thrown', 'batter_id': 'batters_faced'})
    combined_df['avg_innings'] = np.round(combined_df['innings_thrown'] / comb
ined_df['games_played'], decimals = 1)
    combined_df['avg_batters_faced'] = np.round(combined_df['batters_faced'] /
combined_df['innings_thrown'], decimals = 1)

    #Lets plot the top 5 pitchers with the most innings, games played, and bat
ters_faced
    #Get dataframes
    combined_df1 = combined_df.sort_values('avg_innings', ascending=False).hea
d(5)
    combined_df2 = combined_df.sort_values('innings_thrown', ascending=False).
head(5)
    combined_df3 = combined_df.sort_values('avg_batters_faced', ascending=Fals
e).head(5)

    #Visualization
    fig = make_subplots(
        rows=1, cols=3, subplot_titles=("Top 5 Avg. Innings Thrown by Pitcher"
,
                                        "Top 5 Total Innings Thrown by Pitche
r",
                                        "Top 5 Avg. Batters Faced per Inning b
y Pitcher")
    )

    fig.add_trace(
        go.Bar(x=combined_df1['full_name'],
               y=combined_df1['avg_innings']),
        row=1, col=1
    )

    fig.add_trace(
        go.Bar(x=combined_df2['full_name'],
               y=combined_df2['innings_thrown']),
        row=1, col=2
    )

    fig.add_trace(
        go.Bar(x=combined_df3['full_name'],
               y=combined_df3['avg_batters_faced']),
        row=1, col=3
    )

    # Update xaxis properties
    fig.update_xaxes(title_text="Pitcher", row=1, col=1)
    fig.update_xaxes(title_text="Pitcher", row=1, col=2)
    fig.update_xaxes(title_text="Pitcher", row=1, col=3)

    # Update yaxis properties
    fig.update_yaxes(title_text="Total Innings", row=1, col=1)
    fig.update_yaxes(title_text="Avg. Innings", row=1, col=2)
    fig.update_yaxes(title_text="Avg. # of Batters", row=1, col=3)

    # Update title and height
    fig.update_layout(title_text="Top 5 Statistics by Team", width = 1200, hei
ght=700)

    fig.show()
```

```
In [19]: plot_top_pitchers_per_team()
```

```
This metrics return a comparison of top 5 pitchers in terms of batters faced
and innings thrown over a season per team


Enter a team by its abbreviation (e.g.: "oak" for Oakland Athletics and "bos"
for Boston Red Sox)
tor
```



Analysis of Toronta Blue Jays Pitchers

Based on Avg. Innings thrown by Pitcher, Justin Verlander (HOU) and Mike Minor (TEX) comes in first and second. There seems to be an error in the function where I haven't taken account of teams yet. This could be fixed in a future iteration and bump up the package version from 0.0.1 to 0.0.2.

On Total Innings and Batters Faced, Marcus Stroman makes sense where he accrued over 184 innings in 2019 and Richard Urena seems to be correct where he has thrown atrociously (he has a OPS of nearly 0.600).

Based on this data, the coaching staff could have considered giving Stroman an extra day of rest (or a week) and consider bringing in another pitcher than Urena in case of tight (important) games.

```
In [ ]:
```

```
In [ ]:
```