# Using python to explore and predict an NFL Quarterback's career

Carson Ward and Jorge Sempere

# Introduction

**What we want to do?**

Use python and its built in packages to predict the trajectory of an NFL quarterback based upon stats entered by the user

**What are we using in Python to accomplish this?**

In order to get this done we are using the following concepts and python libraries/packages:

- Pandas
- Numpy
- BeautifulSoup
- Matplotlib
- Random
- Object-Oriented Programming
- Simple Statistical Analysis

# Obtaining Current Player Stats

Some of the most important stats for a quarterback are completion %, yards, touchdowns, interceptions, and yards/attempt. Pro-football-reference.com offered the data necessary to complete our calculations. We created a DataFrame from the data obtained with pandas in order to run calculations. Averages and standard deviations were key components in finding a players career length.


# What this looks like

```
def compilestats(year):

    #All passing pages follow same format, minus year
    html = urlopen('https://www.pro-football-reference.com/years/' + str(year) +'/passing.htm')
    statspage = BeautifulSoup(html)
    column_headers = statspage.findAll('tr')[0]
    column_headers = [i.getText() for i in column_headers.findAll('th')]

    # Collect table rows
    rows = statspage.findAll('tr')[1:]

    # Get stats from each row
    stats = []
    for i in range(len(rows)):
        stats.append([col.getText() for col in rows[i].findAll('td')])

    # Create DataFrame from our scraped data
    data = pd.DataFrame(stats, columns=column_headers[1:])

    #Rename categories for readability of the layman
    newcols = data.columns.values
    newcols[9] = 'Completion %'
    newcols[11] = 'Touchdowns'
    newcols[13] = 'Interceptions'
    newcols[10] = 'Yards'
    newcols[17] = 'Yards/Attempt'
    data.columns = newcols
    data.columns

    # Select stat categories to be used in program
    categories = ['GS','Completion %', 'Yards', 'Touchdowns', 'Interceptions', 'Yards/Attempt']

    # Create data subset for league average chart
    data_avg = data[['Player', 'Tm'] + categories]

    # Convert data to numerical values
    pd.options.mode.chained_assignment = None  # default='warn'
    for i in categories:
        data_avg[i] = pd.to_numeric(data[i])


    # Filter by passing yards to eliminate players that should not qualify when calculating league averages
    # Ensures all data will be useful and will not drag down league averages
    data_avg_filtered = data_avg[data_avg['GS'] > 7]
    data_avg_filtered = data_avg_filtered[data_avg_filtered['Yards'] > 400]
    data = data_avg_filtered
    return data
```

```
#NEXT TWO FUNCTIONS ARE HELPER FUNCTIONS FOR COMPILING LEAGUE STATISTICS
#quick function to scrape data from page and convert to averages for given year
def compilestatsAvg(data):
    average = (data['Completion %'].mean(),data['Touchdowns'].mean(),data['Interceptions'].mean(),data['Yards'].mean(),data['Yards/Attempt'].mean())
    return average

#quick function to scrape data from page and convert to standard deviations for given year
def compilestatsSD(data):
    average = (data['Completion %'].std(),data['Touchdowns'].std(),data['Interceptions'].std(),data['Yards'].std(),data['Yards/Attempt'].std())
    return average

#returns tuple of league average statistics for a 5 year sample
#input is starting year for the sample
def leagueAvg(startyear):
    total = (0,0,0,0,0)
    for i in range(0,5):
        average = compilestatsAvg(compilestats(int(startyear)+i))
        total = tuple([sum(x) for x in zip(total,average)])
    avg = tuple(a/5 for a in total)
    avg = tuple(round(a,2) for a in avg)
    return avg

#returns tuple of standard deviations of league average statistics for a 5 year sample
#input is starting year for the sample
def leagueSD(startyear):
    total = (0,0,0,0,0)
    for i in range(0,5):
        average = compilestatsSD(compilestats(int(startyear)+i))
        total = tuple([sum(x) for x in zip(total,average)])
    avg = tuple(a/5 for a in total)
    avg = tuple(round(a,3) for a in avg)
    return avg
```

For the sake of computing the league average quarterback statistics, we used the past 5 years worth of data

# How our program is structured

Essentially, our program follows this structure:

1. The user inputs the statistics they want to test, which are saved in a quarterback class
2. Those statistics are compared to the league average quarterback, and the statistics for that season are given a score
3. The statistics from that season are then used to predict the next year's statistics using a bit of randomization
4. This process is repeated until the quarterback is deemed replaceable

```
class Quarterback:
    def __init__(self, age):
        self.age = age
        self.longevity = 0
        self.ageMultiplier = 1
        self.seasonStats = (0,)

    def obtainStats(self):
        for i in range(1,3):
            print("SEASON " + str(i) + " (Age: " + str(self.age + i - 1) + ")")
            compPer = float(input("Completion %: "))
            TD = float(input("TD: "))
            interceptions = float(input("Interceptions: "))
            yards = float(input("Yards: "))
            yardsPerAttempt = float(input("Yards/Attempt: "))
            print("\n")
            if i == 1:
                self.seasonOneStats = (compPer, TD, interceptions, yards, yardsPerAttempt)
            elif i == 2:
                self.seasonTwoStats = (compPer, TD, interceptions, yards, yardsPerAttempt)
        self.avgStats = tuple([sum(x) for x in zip(self.seasonOneStats, self.seasonTwoStats)])
        self.avgStats = tuple(a/2 for a in self.avgStats)
        self.avgStats = tuple(round(a,3) for a in self.avgStats)


qb = Quarterback(23)
qb.obtainStats()
```

# Predicting statistics using randomization

In order to recreate the variation that occurs in between seasons, we made use of the python random module. It was used to vary the Quarterback's statistics by ranges from -5% to +5% and -20% to +20%  depending on the certain statistic. This ensures that no two simulations are the same, yielding fresh results each time the program is run.

```
import random

#Predicts the next years stats for the player given the previous years stats by randomly generating multipliers for each stat
#These multipliers are around 1 to ensure the stats are believable based upon the previous year
def predictStats(stats):
    multipliers = (random.uniform(0.95,1.05),random.uniform(0.8,1.2),random.uniform(0.8,1.2),random.uniform(0.9,1.1),random.uniform(0.9,1.1))
    newstats = tuple(ele1 * ele2 for ele1, ele2 in zip(stats,multipliers))
    newstats = tuple(round(a,2) for a in newstats)
    templist = list(newstats)
    templist[1] = round(templist[1],0)
    templist[2] = round(templist[2],0)
    templist[3] = round(templist[3],0)
    newstats = tuple(templist)
    return newstats
```

# Taking a look at the career length of an NFL Quarterback

The average career length of an NFL Quarterback is 4.44 years. While this is the average for all quarterbacks, it is not representative of a starting QB in the NFL. In most cases, these are guys that teams invest in in the higher rounds of the NFL Draft, and thus, stay around much longer than just over 4 years. The average career length of a first round pick in the NFL is 9.3 years.

Tim Tebow: 3 years

Sam Bradford: 8 years

Peyton Manning: 17 years

# Longevity

## How we determined how long a player's career will last

After each new "season", a player's statistics were graded based upon how well they performed compared to the league-average starting quarterback. Each individual statistic was compared with the average to find within how many standard deviations it was to the mean. A statistic above the mean was good, and would help to prolong a career, while performing below the average quarterback would make our Quarterback more replaceable, shortening his career

```
def longevity(stats,leagueaverage,lsd):
    sd = tuple(map(lambda i, j: i - j, stats,leagueaverage ))
    # makes the standard deviation for interceptions negative, as a positive standard deviation for a negative stat
    # was making the career of a player who threw a lot of interceptions longer than it should've been
    lsd = list(lsd)
    lsd[2] = -lsd[2]
    lsd = tuple(lsd)
    sd = tuple(ele1 / ele2 for ele1, ele2 in zip(sd,lsd ))
    long = sum(sd)/5
    if long <= 0.4 and long > -0.4:
        long = 0.08
    elif long <= -0.4 and long > -1.2:
        long = 0.13
    elif long > 0.4 and long < 1.2:
        long = 0.06
    elif long <= -1.2:
        long = 0.17
    else:
        long = 0.04
    return long
```

# Putting it all together to simulate a career

```
while qb.longevity < 1:
    qb.seasonStats = predictStats(qb.seasonStats)
    qb.longevity += qb.ageMultiplier*longevity(qb.seasonStats,la,lsd)
    qb.ageMultiplier+=0.1
    statsList = list(qb.seasonStats)
    print("SEASON " + str(qb.age-startAge+1) + " (Age: " + str(qb.age) + ")")
    print('Completion %: ' + str(statsList[0]))
    print('Touchdowns: ' + str(statsList[1]))
    print('Interceptions: ' + str(statsList[2]))
    print('Yards: ' + str(statsList[3]))
    print('Yards / Attempt: ' + str(statsList[4]))

    comppercents.append(statsList[0])
    touchdowns.append(statsList[1])
    ints.append(statsList[2])
    yards.append(statsList[3])
    yardsattempts.append(statsList[4])

    if qb.longevity < 1:
        print("Your Quarterback's career is " + str(round(qb.longevity*100,1)) + "% over")
    else:
        print("Your Quarterback's career is over")
    print("")
    qb.age+=1
    careerStats = tuple([sum(x) for x in zip(careerStats,qb.seasonStats)])
```

# Simulated trial

## User input

```
SEASON 1 (Age: 23)
Completion %:  66.6
TD:  31
Interceptions:  10
Yards:  4336
Yards/Attempt:  7.3


SEASON 2 (Age: 24)
Completion %:  66.6
TD:  31
Interceptions:  10
Yards:  4336
Yards/Attempt:  7.3
```

These numbers were based on Justin Herbert's 2020 rookie season

## Output

```
SEASON 3 (Age: 25)
Completion %: 63.27
Touchdowns: 27.0
Interceptions: 11.0
Yards: 4164.0
Yards / Attempt: 7.19
Your Quarterback's career is 21.6% over

SEASON 4 (Age: 26)
Completion %: 65.22
Touchdowns: 25.0
Interceptions: 12.0
Yards: 4536.0
Yards / Attempt: 7.35
Your Quarterback's career is 32.0% over

SEASON 5 (Age: 27)
Completion %: 64.84
Touchdowns: 21.0
Interceptions: 13.0
Yards: 4234.0
Yards / Attempt: 7.64
Your Quarterback's career is 43.2% over

SEASON 6 (Age: 28)
Completion %: 66.47
Touchdowns: 22.0
Interceptions: 12.0
Yards: 4168.0
Yards / Attempt: 7.97
Your Quarterback's career is 55.2% over

SEASON 7 (Age: 29)
Completion %: 68.09
Touchdowns: 24.0
Interceptions: 14.0
Yards: 3921.0
Yards / Attempt: 7.38
Your Quarterback's career is 68.0% over

SEASON 8 (Age: 30)
Completion %: 66.07
Touchdowns: 19.0
Interceptions: 14.0
Yards: 4147.0
Yards / Attempt: 6.7
Your Quarterback's career is 81.6% over

SEASON 9 (Age: 31)
Completion %: 68.51
Touchdowns: 15.0
Interceptions: 13.0
Yards: 4261.0
Yards / Attempt: 6.98
Your Quarterback's career is 96.0% over

SEASON 10 (Age: 32)
Completion %: 68.87
Touchdowns: 12.0
Interceptions: 13.0
Yards: 4332.0
Yards / Attempt: 7.02
Your Quarterback's career is over
```

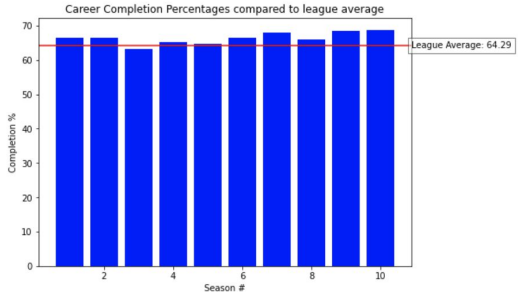
# Using matplotlib to visualize the numbers

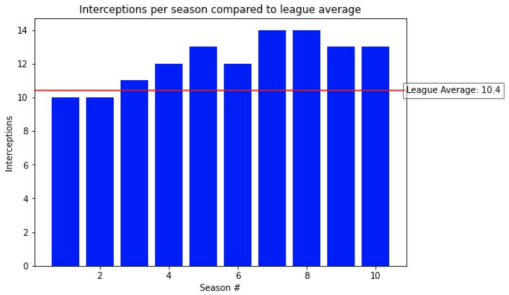
For each individual statistic, we created a bar graph with a horizontal line representing the league average for the statistic. This allows us to gauge the performance of the QB vs. the average during his entire career.

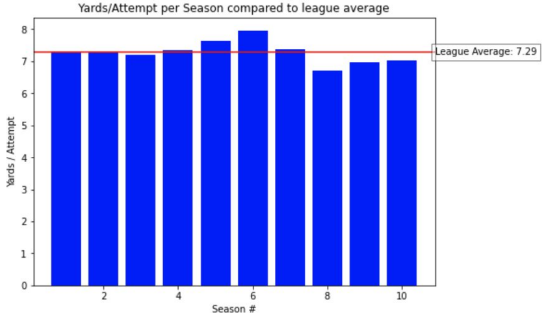
```
#Graphs all stats per season compared to league average stats
la = list(la)

cps = plt.figure().add_axes([0,0,1,1])
cps.bar(range(1,len(ints)+1),height = comppercents, color = 'b')
cps.axhline(la[0], color="r")
cps.text(1.00, la[0], "League Average: " + str(la[0]), va='center', ha="left", bbox=dict(facecolor="w",alpha=0.5),
        transform=cps.get_yaxis_transform())
cps.set_title('Career Completion Percentages compared to league average')
cps.set_ylabel('Completion %')
cps.set_xlabel('Season #')
```

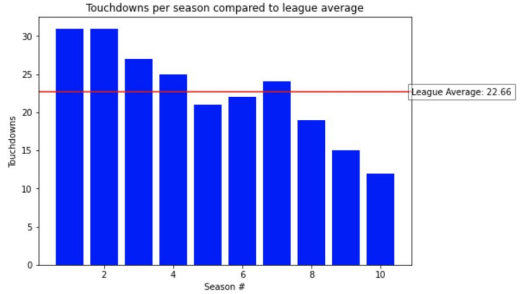
# Using matplotlib to visualize the numbers (cont'd)

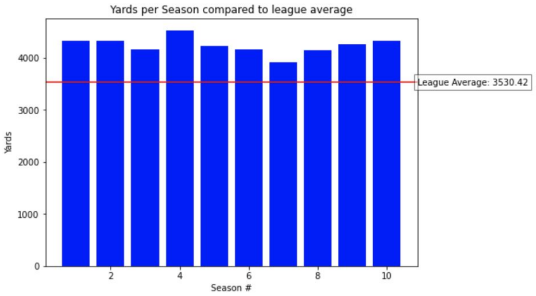
For each individual statistic, we created a bar graph with a horizontal line representing the league average for the statistic. This allows us to gauge the performance of the QB vs. the average during his entire career.



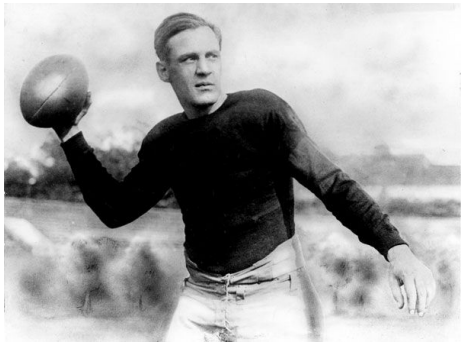



# Life after the League

Most players can be easily forgotten after they leave the league. One of the biggest privileges for any player in the NFL is being inducted into the Hall of Fame. 5 years after retirement, any pro-player/coach can be nominated. We wanted to find a way to calculate how likely a player is to be inducted into the hall of fame based on their career stats. Over the years, the selection process has become more and more competitive. We ran two calculations, one taking into account all current quarterbacks in the hall of fame and one adjusted to better reflect the modern NFL.



Jimmy Conzelman: member since 1964

# Predicting whether our QB is a Hall-of-Famer

```python
#compares qb career stats to the average hof qb
# per Pro Football Reference HOF QB monitor
if careerStats[0] > 59.1:
    hofchecker +=1
if careerStats[1] > 260:
    hofchecker +=1
if careerStats[2] < 195:
    hofchecker +=1
if careerStats[3] > 38166:
    hofchecker +=1
if careerStats[4] > 7.42:
    hofchecker +=1
```

```python
# compares qb career stats to a
# more accurate set of statistics
# that would reflect a qb inducted into the hof today
if careerStats[0] > 62:
    hofchecker2 +=1
if careerStats[1] > 360:
    hofchecker2 +=1
if careerStats[2] < 200:
    hofchecker2 +=1
if careerStats[3] > 45000:
    hofchecker2 +=1
if careerStats[4] > 7.42:
    hofchecker2 +=1
```

```python
print('When compared to the modern day hall-of-fame quarterback,')
if hofchecker2 < 3:
    print('Your Quarterback had a decent career, but will not be remembered as one of the all-time greats')
if hofchecker2 == 3:
    print('There is a chance your quarterback had a hall-of-fame career, but it is by no means a gaurantee')
if hofchecker2 == 4:
    print('Your Quarterback had a fantastic career and will be remembered by many in the hall-of-fame')
if hofchecker2 == 5:
    print('Your Quarterback had one of the best careers of all-time and is sure to be a first-ballot hall-of-famer')
```

```
When compared to all hall-of-fame quarterbacks,
There is a chance your quarterback had a hall-of-fame career, but it is by no means a gaurantee

When compared to the modern day hall-of-fame quarterback,
Your Quarterback had a decent career, but will not be remembered as one of the all-time greats

CAREER STATS
Completion %: 66.5
Touchdowns: 227.0
Interceptions: 122.0
Yards: 42435.0
Yards / Attempt: 7.3
```

# Comparing our QB to QBs from the 2020 NFL Season

Using the DataFrame we created of NFL Quarterback stats from the 2020 season, we compared the average career stats of our quarterback to his real-life counterparts.

```
data2020 = compilestats(2020)
match = 1
delta = 1
delta2 = 1
#for loop that takes ratios of each stat from our qb and compares it to every qb from the 2020 season
#whichever set of ratios average out closest to 1 will be our qb of choice for the comparison
for index,row in data2020.iterrows():
    tempMatch = (row['Completion %'],row['Touchdowns'],row['Interceptions'],row['Yards'],row['Yards/Attempt'])
    tempMatch1 = tuple(ele1 / ele2 for ele1, ele2 in zip(avgStatsCareer,tempMatch))
    #flips interception ratio as it would positively benefit players comparisons for throwing interceptions
    tempMatch1 = list(tempMatch1)
    if tempMatch1[2] != 0:
        tempMatch1[2] = 1/tempMatch1[2]
    tempMatch1 = tuple(tempMatch1)
    matchpercent = sum(list(tempMatch1))
    matchpercent = matchpercent/5
    delta = abs(1-matchpercent)
    if  delta < delta2:
        delta2 = delta
        match = matchpercent
        playermatch = (row['Player'],row['Completion %'],row['Touchdowns'],row['Interceptions'],row['Yards'],row['Yards/Attempt'])
```

```
Your quarterback is similar to Kyler Murray* based on his stats from the 2020 NFL season
His stats from that season were:
Completion %: 67.2
Touchdowns: 26.0
Interceptions: 12.0
Yards: 3971.0
Yards / Attempt: 7.1

Your Quarterback's average stats over his career were:
Completion %: 66.5
Touchdowns: 22.7
Interceptions: 12.2
Yards: 4244.0
Yards / Attempt: 7.3
```

# Simulated trial #2

These numbers were based on Sam Darnold's past two seasons (2019 & 2020)


## User Input

```
SEASON 1 (Age: 23)
Completion %:  60
TD:  19
Interceptions:  13
Yards:  3400
Yards/Attempt:  7.0


SEASON 2 (Age: 24)
Completion %:  59.6
TD:  9
Interceptions:  11
Yards:  2200
Yards/Attempt:  6.1
```

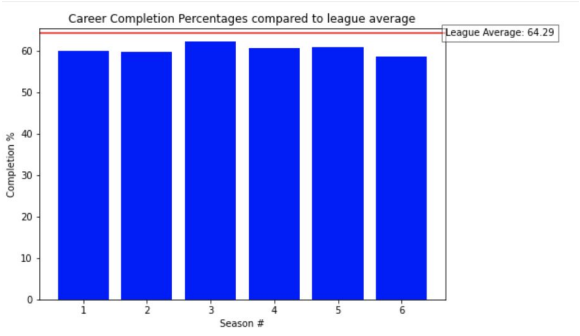
## Output

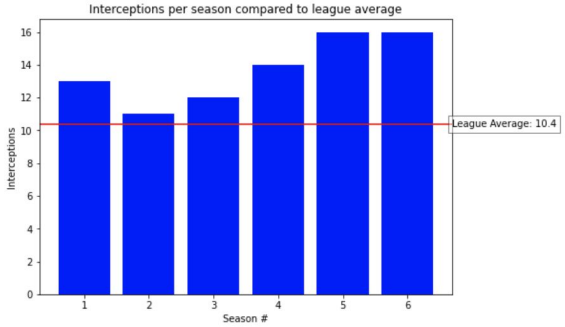
```
SEASON 3 (Age: 25)
Completion %: 62.31
Touchdowns: 8.0
Interceptions: 12.0
Yards: 2066.0
Yards / Attempt: 6.67
Your Quarterback's career is 45.6% over

SEASON 4 (Age: 26)
Completion %: 60.65
Touchdowns: 7.0
Interceptions: 14.0
Yards: 2007.0
Yards / Attempt: 7.24
Your Quarterback's career is 62.5% over

SEASON 5 (Age: 27)
Completion %: 60.86
Touchdowns: 8.0
Interceptions: 16.0
Yards: 1870.0
Yards / Attempt: 6.86
Your Quarterback's career is 86.3% over

SEASON 6 (Age: 28)
Completion %: 58.54
Touchdowns: 9.0
Interceptions: 16.0
Yards: 2042.0
Yards / Attempt: 7.02
Your Quarterback's career is over
```

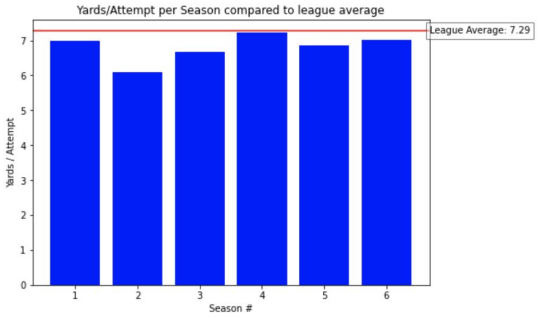
```
Your quarterback is similar to Sam Darnold based on his stats
His stats from that season were:
Completion %: 59.6
Touchdowns: 9.0
Interceptions: 11.0
Yards: 2208.0
Yards / Attempt: 6.1

Your Quarterback's average stats over his career were:
Completion %: 60.3
Touchdowns: 10.0
Interceptions: 13.7
Yards: 2264.0
Yards / Attempt: 6.8
```

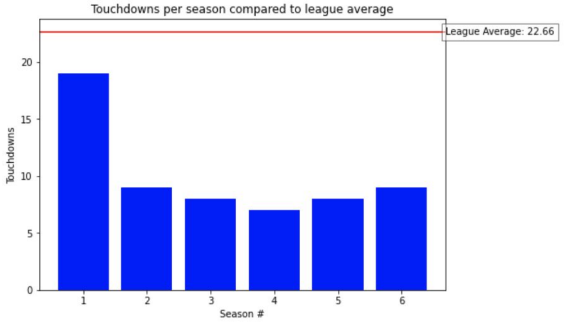
# Simulated Trial #2

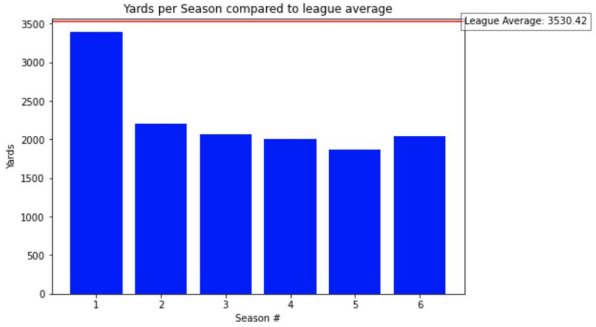

These numbers were based on Sam Darnold's past two seasons (2019 & 2020)



# Limitations

Obviously, the predictions created by this program will not match those of real-life, but various factors are out of our control.

- Using random to generate future stats will never account for whether or not a player can improve or regress, or the things the player can not control
- Comparing our quarterback to real life ones is only partially successful as the the function that compares statistics of the quarterbacks weights each value equally

# Questions?