In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import matplotlib.ticker as mtick
import sqlite3
import seaborn as sns
from matplotlib.offsetbox import OffsetImage, AnnotationBbox
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from bs4 import BeautifulSoup
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time
import requests
import shutil
import datetime
from scipy.stats import norm
import os
import winsound
```

In [ ]:
```
home_folder = 'C:\\Users\\Travis\\OneDrive\\Data Science\\Personal_Projects\\Sports
os.chdir(home_folder)
```

In [ ]:
```
def replace_name_values2(filename):
        # replace values with dashes for compatibility
    filename = filename.replace('%','_')
    filename = filename.replace('=','_')
    filename = filename.replace('?','_')
    filename = filename.replace('&','_')
    filename = filename.replace('20Season_','')
    filename = filename.replace('_20Season','')
    filename = filename.replace('SeasonType_','')
    filename = filename.replace('sort_gdate_dir_-1_','')
    filename = filename.replace('SeasonYear_','')
    return filename
```

In [ ]:
```
def trans_urls(url):
    new_url = str(url)[34:].replace('/', '_')
    filename = replace_name_values2(new_url)
    filename = filename.replace('SeasonYear_', '')
    return filename
```

In [ ]:

```
def append_the_data(folder, data_prefix, filename_selector):
    # Appending data together via folder and/or file name

    path = folder
    p = os.listdir(path)
    pf = pd.DataFrame(p)


    # filter for files that contain the filename_selector
    pf_reg = pf.loc[pf[0].astype(str).str.contains(filename_selector)]

    appended_data = []
    for file in pf_reg[0]:
        data = pd.read_csv(folder + '/' + file)
        # if "Season" a column, drop it
        if 'Season' in data.columns:
            data = data.drop(columns = ['Season'])

        data['season'] = file[(file.find('20')):(file.find('20'))+4]
        data['season_type'] = np.where('Regular' in file, 'Regular', 'Playoffs')
        # add prefix to columns
        data = data.add_prefix(data_prefix)
        data.columns = data.columns.str.lower()
        appended_data.append(data)

    appended_data = pd.concat(appended_data)
    return appended_data
```

In [ ]:

```
def grab_playtype(url_list, file_folder):
        # Scrape Season-Level player data from the url_list

        i = 0
        for u in url_list:

                driver.get(u)
                time.sleep(2)

                # if the page does not load, go to the next in the list
                try:
                        xpath = '//*[@id="__next"]/div[2]/div[2]/div[3]/section[2]/
                        elem = WebDriverWait(driver, 30).until(EC.presence_of_eleme
                except:
                        print(f'{u} did not load. Moving to next url.')
                        continue

                # click "all pages"
                xpath_all = '//*[@id="__next"]/div[2]/div[2]/div[3]/section[2]/div/
                elem = WebDriverWait(driver, 30).until(EC.presence_of_element_locat

                driver.find_element(by=By.XPATH, value=xpath_all).click()
                src = driver.page_source
                parser = BeautifulSoup(src, "lxml")
                table = parser.find("table", attrs = {"class":"Crom_table__p1iZz"})
                headers = table.findAll('th')
                headerlist = [h.text.strip() for h in headers[0:]]
                row_names = table.findAll('a')                                  # find r
                row_list = [b.text.strip() for b in row_names[0:]]
                rows = table.findAll('tr')[0:]
                player_stats = [[td.getText().strip() for td in rows[i].findAll('td
                tot_cols = len(player_stats[1])                                 #set the
                headerlist = headerlist[:tot_cols]
                stats = pd.DataFrame(player_stats, columns = headerlist)

                # assign filename
                filename = file_folder + str(u[34:]).replace('/', '_') + '.csv'
                filename = replace_name_values2(filename)
                filename = filename.replace('SeasonYear_', '')
                pd.DataFrame.to_csv(stats, filename)
                i += 1
                lu = len(url_list)
                # close driver
                print(f'{filename} Completed Successfully! {i} / {lu} Complete!')

        winsound.Beep(523, 500)
```

In [ ]:

```
def grab_player_playtype_defense(url_list, file_folder):
        driver = webdriver.Chrome()
        i = 0
        for u in url_list:
                driver.get(u)
                time.sleep(1)

                #  go to defense page
                try:
                        xpath_defense = '//*[@id="__next"]/div[2]/div[2]/div[3]/sec
                        elem = WebDriverWait(driver, 30).until(EC.presence_of_eleme

                except:
                        print(f'{u} did not load. Moving to next url.')
                        continue

                driver.find_element(by=By.XPATH, value=xpath_defense).click()

                # select all
                # if the page does not load, go to the next in the list
                try:
                        xpath = '//*[@id="__next"]/div[2]/div[2]/div[3]/section[2]/
                        elem = WebDriverWait(driver, 30).until(EC.presence_of_eleme
                except:
                        print(f'{u} did not load. Moving to next url.')
                        continue

                driver.find_element(by=By.XPATH, value=xpath).click()
                src = driver.page_source
                parser = BeautifulSoup(src, "lxml")
                table = parser.find("table", attrs = {"class":"Crom_table__p1iZz"})
                headers = table.findAll('th')
                headerlist = [h.text.strip() for h in headers[0:]]
                row_names = table.findAll('a')                                  # find r
                row_list = [b.text.strip() for b in row_names[0:]]
                rows = table.findAll('tr')[0:]
                player_stats = [[td.getText().strip() for td in rows[i].findAll('td
                tot_cols = len(player_stats[1])                                 #set the
                headerlist = headerlist[:tot_cols]
                stats = pd.DataFrame(player_stats, columns = headerlist)
                filename = file_folder + str(u[34:]).replace('/', '_') + '.csv'
                filename = replace_name_values2(filename)
                filename = filename.replace('SeasonYear_','')
                pd.DataFrame.to_csv(stats, filename)
                i += 1
                lu = len(url_list)
                print(f'{filename} Completed Successfully! {i} / {lu} Complete!')
        winsound.Beep(523, 500)
```

In [ ]:
```
if os.path.isdir('data/player/playtype/defense') is False:
    os.mkdir('data/player/playtype/defense')

def_playtypes = ['isolation', 'ball-handler', 'roll-man',
            'playtype-post-up','spot-up', 'hand-off','off-screen']
season_types = ['Playoffs', 'Regular%20Season']

yearz = ['2021-22', '2020-21', '2019-20', '2018-19', '2017-18', '2016-17', '2015-16
def_playtype_urls = []
for year in yearz:
    for play in def_playtypes:
        for s_types in season_types:
            url = 'https://www.nba.com/stats/players/'+ play +'/?SeasonYear=' + yea
            def_playtype_urls.append(str(url))
```

In [ ]:
```
def_playtypes = pd.DataFrame(def_playtype_urls, columns=['url'])
def_playtypes.head(2)
```

Out[ ]:

	url
0	https://www.nba.com/stats/players/isolation/?S...
1	https://www.nba.com/stats/players/isolation/?S...

In [ ]:
```
def_playtypes['filename'] = def_playtypes.apply(lambda row: trans_urls(row['url']),
def_playtypes.head(2)
```

Out[ ]:

	url	filename
0	https://www.nba.com/stats/players/isolation/?S...	isolation__2021-22_Playoffs
1	https://www.nba.com/stats/players/isolation/?S...	isolation__2021-22_Regular

In [ ]:
```
# find already downloaded files
already_downloaded_play = os.listdir('data/player/playtype/defense/playoffs')
already_downloaded_reg = os.listdir('data/player/playtype/defense/regular_season')
already_downloaded = already_downloaded_play + already_downloaded_reg
already_downloaded = [x.replace('.csv', '') for x in already_downloaded]

# remove already downloaded files from the list
def_playtypes_not_dl = def_playtypes[~def_playtypes['filename'].isin(already_downlo

need_to_dl = def_playtypes_not_dl['url'].to_list()

len(need_to_dl)
```

Out[ ]:
```
0
```

In [ ]:
```
if len(need_to_dl) > 0:
    grab_player_playtype_defense(need_to_dl, 'data/player/playtype/defense/')
else:
    print('All files already downloaded')
```

```
All files already downloaded
```

Update This Year

In [ ]:
```
# get list of this years urls
def_playtypes = ['isolation', 'ball-handler', 'roll-man',
            'playtype-post-up','spot-up', 'hand-off','off-screen']

this_year_urls = []
for play in def_playtypes:
    url = 'https://www.nba.com/stats/players/'+ play +'/?SeasonYear=2022-23&SeasonT
    this_year_urls.append(str(url))
```

In [ ]:
```
driver = webdriver.Chrome()
grab_playtype(this_year_urls, 'data/player/playtype/')
```

```
data/player/playtype/isolation__2022-23_Regular.csv Completed Successfully! 1 / 7 C
omplete!
data/player/playtype/ball-handler__2022-23_Regular.csv Completed Successfully! 2 /
7 Complete!
data/player/playtype/roll-man__2022-23_Regular.csv Completed Successfully! 3 / 7 Co
mplete!
data/player/playtype/playtype-post-up__2022-23_Regular.csv Completed Successfully!
4 / 7 Complete!
data/player/playtype/spot-up__2022-23_Regular.csv Completed Successfully! 5 / 7 Com
plete!
data/player/playtype/hand-off__2022-23_Regular.csv Completed Successfully! 6 / 7 Co
mplete!
data/player/playtype/off-screen__2022-23_Regular.csv Completed Successfully! 7 / 7
Complete!
```

In [ ]:
```
# move files to the correct folder
for file in os.listdir('data/player/playtype/defense/'):
    if file.endswith('.csv'):
        if 'Playoffs' in file:
            shutil.move('data/player/playtype/defense/'+ file, 'data/player/playtyp

        elif 'Regular' in file:
            shutil.move('data/player/playtype/defense/'+ file, 'data/player/playtyp
```

Aggreagate

In [ ]:
```
playtype_defense_reg = os.listdir('data/player/playtype/defense/regular_season/')
playtype_defense_playoffs = os.listdir('data/player/playtype/defense/playoffs/')
```

In [ ]:

```
# agg reg_season ball handler
# agg each sub-category

d_ball_handler_reg = append_the_data('data/player/playtype/defense/regular_season/'
d_ball_handler_playoffs = append_the_data('data/player/playtype/defense/playoffs/',
d_ball_handler = pd.concat([d_ball_handler_reg, d_ball_handler_playoffs], axis = 0)

d_ball_handler_reg.to_csv('data/player/aggregates/Playtype_Defense_Ball_Handler_Reg
d_ball_handler_playoffs.to_csv('data/player/aggregates/Playtype_Defense_Ball_Handle
d_ball_handler.to_csv('data/player/aggregates/Playtype_Defense_Ball_Handler_ALL.csv

d_hand_off_reg = append_the_data('data/player/playtype/defense/regular_season/', 'p
d_hand_off_playoffs = append_the_data('data/player/playtype/defense/playoffs/', 'pl
d_hand_off = pd.concat([d_hand_off_reg, d_hand_off_playoffs], axis = 0)

d_hand_off_reg.to_csv('data/player/aggregates/Playtype_Defense_Hand_Off_Regular_Sea
d_hand_off_playoffs.to_csv('data/player/aggregates/Playtype_Defense_Hand_Off_Playof
d_hand_off.to_csv('data/player/aggregates/Playtype_Defense_Hand_Off_ALL.csv')

d_iso_reg = append_the_data('data/player/playtype/defense/regular_season/', 'playty
d_iso_playoffs = append_the_data('data/player/playtype/defense/playoffs/', 'playtyp
d_iso = pd.concat([d_iso_reg, d_iso_playoffs], axis = 0)

d_iso_reg.to_csv('data/player/aggregates/Playtype_Defense_Isolation_Regular_Season.
d_iso_playoffs.to_csv('data/player/aggregates/Playtype_Defense_Isolation_Playoffs.c
d_iso.to_csv('data/player/aggregates/Playtype_Defense_Isolation_ALL.csv')

d_off_screen_reg = append_the_data('data/player/playtype/defense/regular_season/',
d_off_screen_playoffs = append_the_data('data/player/playtype/defense/playoffs/', '
d_off_screen = pd.concat([d_off_screen_reg, d_off_screen_playoffs], axis = 0)

d_off_screen_reg.to_csv('data/player/aggregates/Playtype_Defense_Off_Screen_Regular
d_off_screen_playoffs.to_csv('data/player/aggregates/Playtype_Defense_Off_Screen_Pl
d_off_screen.to_csv('data/player/aggregates/Playtype_Defense_Off_Screen_ALL.csv')

d_postup_reg = append_the_data('data/player/playtype/defense/regular_season/', 'pla
d_postup_playoffs = append_the_data('data/player/playtype/defense/playoffs/', 'play
d_postup = pd.concat([d_postup_reg, d_postup_playoffs], axis = 0)

d_postup_reg.to_csv('data/player/aggregates/Playtype_Defense_Post_Up_Regular_Season
d_postup_playoffs.to_csv('data/player/aggregates/Playtype_Defense_Post_Up_Playoffs.
d_postup.to_csv('data/player/aggregates/Playtype_Defense_Post_Up_ALL.csv')

d_rollman_reg = append_the_data('data/player/playtype/defense/regular_season/', 'pl
d_rollman_playoffs = append_the_data('data/player/playtype/defense/playoffs/', 'pla
d_rollman = pd.concat([d_rollman_reg, d_rollman_playoffs], axis = 0)

d_rollman_reg.to_csv('data/player/aggregates/Playtype_Defense_Roll_Man_Regular_Seas
d_rollman_playoffs.to_csv('data/player/aggregates/Playtype_Defense_Roll_Man_Playoff
d_rollman.to_csv('data/player/aggregates/Playtype_Defense_Roll_Man_ALL.csv')

d_spotup_reg = append_the_data('data/player/playtype/defense/regular_season/', 'pla
d_spotup_playoffs = append_the_data('data/player/playtype/defense/playoffs/', 'play
d_spotup = pd.concat([d_spotup_reg, d_spotup_playoffs], axis = 0)

d_spotup_reg.to_csv('data/player/aggregates/Playtype_Defense_Spot_Up_Regular_Season
```

```
d_spotup_playoffs.to_csv('data/player/aggregates/Playtype_Defense_Spot_Up_Playoffs.
d_spotup.to_csv('data/player/aggregates/Playtype_Defense_Spot_Up_ALL.csv')
```

In [ ]:
```
d_ball_handler
```

Out[ ]:

	playtype_ball_handler__unnamed: 0	playtype_ball_handler__player	playtype_ball_handler__team	pla
0	0	NaN	NaN	
1	1	Reggie Jackson	DET	
2	2	Damian Lillard	POR	
3	3	Chris Paul	LAC	
4	4	Kemba Walker	CHA	
...	...	...	...	
46	46	Patty Mills	BKN	
47	47	Damion Lee	GSW	
48	48	Nikola Vucevic	CHI	
49	49	Andre Drummond	BKN	
50	50	Dewayne Dedmon	MIA	

2577 rows × 20 columns

In [ ]:
```
print(f' ball handler size: {d_ball_handler.shape}, hand off size: {d_hand_off.shap
```

```
 ball handler size: (2577, 20), hand off size: (2280, 20), iso size: (2306, 20), of
f screen size: (2295, 20), post up size: (1814, 20), rollman: (1931, 20), spotup:
(3141, 20)
```

In [ ]:
```
# Merge the data

defensive_playtypes1 = pd.merge(d_spotup, d_ball_handler,
                                left_on = ['playtype_spot_up__team', 'playtype_spot
                                right_on = ['playtype_ball_handler__team', 'playtyp
                                how = 'left')

defensive_playtypes2 = pd.merge(defensive_playtypes1, d_hand_off,
                                left_on = ['playtype_spot_up__team', 'playtype_spot
                                right_on = ['playtype_hand_off__team', 'playtype_ha
                                how = 'left')

defensive_playtypes3 = pd.merge(defensive_playtypes2, d_iso,
                                left_on = ['playtype_spot_up__team', 'playtype_spot
                                right_on = ['playtype_isolation__team', 'playtype_i
                                how = 'left')

defensive_playtypes4 = pd.merge(defensive_playtypes3, d_off_screen,
                                left_on = ['playtype_spot_up__team', 'playtype_spot
                                right_on = ['playtype_off_screen__team', 'playtype_
                                how = 'left')

defensive_playtypes5 = pd.merge(defensive_playtypes4, d_postup,
                                left_on = ['playtype_spot_up__team', 'playtype_spot
                                right_on = ['playtype_post_up__team', 'playtype_pos
                                how = 'left')

defensive_playtypes6 = pd.merge(defensive_playtypes5, d_rollman,
                                left_on = ['playtype_spot_up__team', 'playtype_spot
                                right_on = ['playtype_roll_man__team', 'playtype_ro
                                how = 'left')
```

In [ ]:
```
# save to csv
defensive_playtypes6.to_csv('data/player/aggregates/Playtype_Defense_ALL.csv')
```