BoxScores Updating

PLAYER BOXSCORES

Need to Know:

- We download data by the MONTH.
- Nba.com "Months" start in october (i.e., the beginning of the season). So 10 = 1.

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

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

In [ ]:

```
def grab_player_data2(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)
                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 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 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 [ ]:
```
today = datetime.date.today()
print(today)
month = today.month
print(month)
```

```
2022-12-24
12
```

In [ ]:
```
# read current boxscores, get the last date
boxscores = pd.read_csv('data/player/aggregates/Trad&Adv_box_scores_GameView.csv')
boxscores.head()
```

Out[ ]:

	Unnamed: 0	trad_unnamed: 0	trad_player	trad_team	trad_match up	trad_game date	trad_w/l	trad_min
0	7	1	Damian Lillard	POR	POR vs. LAL	10/31/2012	W	35.0
1	8	2	Kobe Bryant	LAL	LAL @ POR	10/31/2012	L	38.0
2	9	3	Nicolas Batum	POR	POR vs. LAL	10/31/2012	W	40.0
3	10	4	Al Jefferson	UTA	UTA vs. DAL	10/31/2012	W	29.0
4	11	5	Paul Millsap	UTA	UTA vs. DAL	10/31/2012	W	33.0

5 rows × 53 columns

In [ ]:
```
boxscores['date'] = pd.to_datetime(boxscores['trad_game date'])
boxscores = boxscores.sort_values(by = 'date', ascending = False)
boxscores = boxscores.reset_index(drop = True)
last_date = boxscores['date'][0]
# change to date from datetime
last_date = last_date.date()
last_date
```

Out[ ]:
```
datetime.date(2022, 12, 23)
```

In [ ]:
```
# get months since oct 2022
today = datetime.date.today()
months_since_oct_2022 = (today.year - 2022) * 12 + today.month - 9
this_season_months = np.arange(1, months_since_oct_2022+1, 1)

s_17 = '2017-18'
s_18 = '2018-19'
s_19 = '2019-20'
s_20 = '2020-21'
s_21 = '2021-22'
s_22 = '2022-23'

s_16 = '2016-17'
s_15 = '2015-16'
s_14 = '2014-15'
s_13 = '2013-14'
s_12 = '2012-13'
s_11 = '2011-12'

yearz = ['2022', '2021', '2020', '2019', '2018', '2017', '2016', '2015', '2014', '2
s_years = [s_22, s_21, s_20, s_19, s_18, s_17, s_16, s_15, s_14, s_13, s_12, s_11]

#months played by year -- the months that HAD GAMES during said season
# October is counted as Month #1 --- Most should go 1-7.
months_2022 = this_season_months
months_2021 = ['1', '2','3', '4', '5', '6', '7']
months_2020 = ['3', '4', '5', '6', '7','8']
months_2019 = ['1', '2','3', '4', '5', '6']
months_2018 = ['1', '2','3', '4', '5', '6', '7']
months_2017 = ['1', '2','3', '4', '5', '6', '7']
months_2016 = ['1', '2','3', '4', '5', '6', '7']
months_2015 = ['1', '2','3', '4', '5', '6', '7']
months_2014 = ['1', '2','3', '4', '5', '6', '7']
months_2013 = ['1', '2','3', '4', '5', '6', '7']
months_2012 = ['1', '2','3', '4', '5', '6', '7']
months_2011 = ['1', '2','3', '4', '5', '6', '7']

years = ['2022-23','2021-22', '2020-21', '2019-20', '2018-19', '2017-18', '2016-17'
types = ['boxscores-advanced', 'boxscores-traditional']
season_types = ['Regular%20Season', 'Playoffs']
```

In [ ]:
```
print(this_season_months)
current_month = max(this_season_months)
print(f'Current Month is {current_month}')
```

```
[1 2 3]
Current Month is 3
```

In [ ]:
```
trad_url_update = ('https://www.nba.com/stats/players/boxscores-traditional/?Season
adv_url_update = ('https://www.nba.com/stats/players/boxscores-advanced/?Season=202
```

In [ ]:

```
def get_all_urls():
    # List all months to scrape
    urls1 = []

    for m in months_2022:
        for t in types:
            url = 'https://www.nba.com/stats/players/' + str(t) +'/?Season=2022-23&
            urls1.append(url)

    for m in months_2021:
        for t in types:
                url = 'https://www.nba.com/stats/players/' + str(t) +'/?Season=2021
                urls1.append(url)

    for m in months_2020:
        for t in types:
                url = 'https://www.nba.com/stats/players/' + str(t) +'/?Season=2020
                urls1.append(url)

    for m in months_2019:
        for t in types:
                url = 'https://www.nba.com/stats/players/' + str(t) +'/?Season=2019
                urls1.append(url)

    for m in months_2018:
        for t in types:
                url = 'https://www.nba.com/stats/players/' + str(t) +'/?Season=2018
                urls1.append(url)

    for m in months_2017:
        for t in types:
                url = 'https://www.nba.com/stats/players/' + str(t) +'/?Season=2017
                urls1.append(url)

    for m in months_2016:
        for t in types:
                url = 'https://www.nba.com/stats/players/' + str(t) +'/?Season=2016
                urls1.append(url)

    for m in months_2015:
        for t in types:
                url = 'https://www.nba.com/stats/players/' + str(t) +'/?Season=2015
                urls1.append(url)

    for m in months_2014:
        for t in types:
                url = 'https://www.nba.com/stats/players/' + str(t) +'/?Season=2014
                urls1.append(url)

    for m in months_2013:
        for t in types:
                url = 'https://www.nba.com/stats/players/' + str(t) +'/?Season=2013
                urls1.append(url)

    for m in months_2012:
        for t in types:
```

```
                url = 'https://www.nba.com/stats/players/' + str(t) +'/?Season=2012
                urls1.append(url)

    return urls1
```

In [ ]:
```
print(today)
print(last_date)
```

```
2022-12-24
2022-12-23
```

In [ ]:
```
# if the last_date is before yesterday, then scrape the new data
yesterday = today - datetime.timedelta(days=1)

if last_date < yesterday:
    driver = webdriver.Chrome()
    # update this month's data
    trad_url_update = ('https://www.nba.com/stats/players/boxscores-traditional/?Se
    adv_url_update = ('https://www.nba.com/stats/players/boxscores-advanced/?Season
    urls2 = [trad_url_update, adv_url_update]
    # scrape trad and adv updates
    grab_player_data2(urls2, 'data/player/box_scores/')

else:
    print('Data is Updated')
```

```
Data is Updated
```

Append

In [ ]:
```
all_boxes_trad = append_the_data('data/player/box_scores/', 'trad_', 'traditional')
all_boxes_adv = append_the_data('data/player/box_scores/', 'adv_', 'advanced')

all_boxes_trad.shape
```

Out[ ]:
```
(261950, 28)
```

In [ ]:
```
all_boxes = pd.merge(all_boxes_trad,all_boxes_adv,
                     left_on=['trad_player', 'trad_team', 'trad_season', 'trad_s
                     right_on=['adv_player', 'adv_team', 'adv_season', 'adv_seas
                     how='left')
all_boxes
```

Out[ ]:

	trad_unnamed: 0	trad_player	trad_team	trad_matchup	trad_game date	trad_w/l	trad_min	trad_
0	0	NaN	NaN	NaN	NaN	NaN	NaN	N
1	0	NaN	NaN	NaN	NaN	NaN	NaN	N
2	0	NaN	NaN	NaN	NaN	NaN	NaN	N
3	0	NaN	NaN	NaN	NaN	NaN	NaN	N
4	0	NaN	NaN	NaN	NaN	NaN	NaN	N
...	...	...	...	...	...	...	...	
262347	3536	Killian Hayes	DET	DET vs. DAL	12/01/2022	W	33.0	2
262348	3537	Christian Wood	DAL	DAL @ DET	12/01/2022	L	35.0	2
262349	3538	Marvin Bagley III	DET	DET vs. DAL	12/01/2022	W	34.0	1
262350	3539	Tim Hardaway Jr.	DAL	DAL @ DET	12/01/2022	L	40.0	2
262351	3540	Bojan Bogdanovic	DET	DET vs. DAL	12/01/2022	W	41.0	3

262352 rows × 52 columns

In [ ]:
```
all_boxes = all_boxes.dropna(subset=['trad_player'])
```

In [ ]:
```
all_boxes.to_csv('data/player/aggregates/Trad&Adv_box_scores_GameView.csv')
```

In [ ]:
```
```