Capstone Project Proposal: Predicting NBA Scores

Springboard Data Science Career Track - March 2, 2020 Cohort

By:	Justin Huang
Mentor:	Blake Arensdorf
Date:	July 7, 2020

Problem Statement

Predicting the score of an NBA Game (and thus predicting the game winners)

The NBA Season was suspended on March 11 2020 due to the global COVID-19 pandemic. After more than 2 months of uncertainty, the NBA announced that the season will resume starting July 30 2020 albeit in a condensed format with reduced number of teams thus scraping the originally planned schedule.

Since the original remaining games of the NBA 2019-2020 season are being cancelled, I would like to simulate the scores of the cancelled regular season games to determine the winners. Next follows the playoff games and therefore an overall NBA champion for the season. As a lifelong Toronto Raptors fan, I am also curious how likely it is that they will emerge as repeat champions!

For simplicity, the simulations will be carried out with the originally scheduled remaining games followed by traditional playoffs format to ultimately declare an NBA Champion. It will be interesting to see how the simulation results compare to the actual games given that now the season will resume!

Description of Dataset

Where

The source of raw data for my NBA Scores Capstone project is a Kaggle repository:
https://www.kaggle.com/nathanlauga/nba-games#games_details.csv
https://www.kaggle.com/nathanlauga/nba-games?select=teams.csv

This user compiled high level NBA box score data for games played between 2003-2004 up to 2019-2020 (up to March 1 2020, right before the COVID suspension). This dataset had basic box score data: PTS, REB, AST (and their respective percentages). 2 primary *.csv files were downloaded from Kaggle and served as the basis for the Capstone Project.

1) Games - game results for previous 14 seasons (games.csv) and
2) Teams - team information (teams.csv)

A local copy of these 2 files were downloaded and imported using python (Jupyter Notebooks) where a preliminary analysis was carried out with respect to cleaning, filling (if empty) and wrangling the data. Below outlines the steps carried out.

What Was Done

Team Conferences and Division

Conference and division data was missing from Kaggle data. This information would be needed when sorting and grouping teams for playoff seeding.

I manually created a *.csv containing the information and loaded it as a “teams_conf_div” dataframe. This included the human readable city name and city abbreviation. Next I combined the kaggle teams.csv with my teams_conf_div.csv and removed the columns irrelevant to the analysis (Team Owner, Year Team Established, Name of Arena etc.)

Finally a new clean file was ready and outputted for downstream use (teams_df.csv).

```
teams_df.head()
```

TEAM_ID	ABBREVIATION	NICKNAME	CITY	CONFERENCE	DIVISION
1610612737	ATL	Hawks	Atlanta	East	Southeast
1610612738	BOS	Celtics	Boston	East	Atlantic
1610612740	NOP	Pelicans	New Orleans	West	Southwest
1610612741	CHI	Bulls	Chicago	East	Central
1610612742	DAL	Mavericks	Dallas	West	Southwest

Game Metadata

Initially, simple date parsing was carried out to enrich the games metadata so to easily slice by month and year.

```
# Extracting the D, M, Y from the Game Date
loading_games_df['dt_MONTH'] = pd.DatetimeIndex(loading_games_df['GAME_DATE_EST']).month
loading_games_df['dt_YEAR'] = pd.DatetimeIndex(loading_games_df['GAME_DATE_EST']).year
loading_games_df['dt_YEAR_MONTH'] = pd.to_datetime(loading_games_df['GAME_DATE_EST']).dt.to_period('M')
loading_games_df
```

_home	...	PTS_away	FG_PCT_away	FT_PCT_away	FG3_PCT_away	AST_away	REB_away	HOME_TEAM_WINS	dt_MONTH	dt_YEAR	dt_YEAR_MONTH
0.900	...	93.0	0.402	0.762	0.226	20.0	61.0	0	3	2020	2020-03
0.400	...	111.0	0.468	0.632	0.275	28.0	56.0	0	3	2020	2020-03
0.805	...	130.0	0.505	0.650	0.488	27.0	37.0	1	3	2020	2020-03
0.700	...	118.0	0.461	0.897	0.263	24.0	36.0	1	3	2020	2020-03
0.885	...	100.0	0.413	0.667	0.429	23.0	42.0	1	3	2020	2020-03
...	...	...	...	...	...	...	...	...	...	...	...
0.821	...	87.0	0.366	0.643	0.375	17.0	43.0	1	10	2014	2014-10
0.719	...	85.0	0.411	0.636	0.267	17.0	47.0	0	10	2014	2014-10
0.682	...	95.0	0.387	0.659	0.500	19.0	43.0	1	10	2014	2014-10
0.771	...	94.0	0.469	0.725	0.385	18.0	45.0	1	10	2014	2014-10
0.679	...	98.0	0.462	0.706	0.438	19.0	42.0	0	10	2014	2014-10

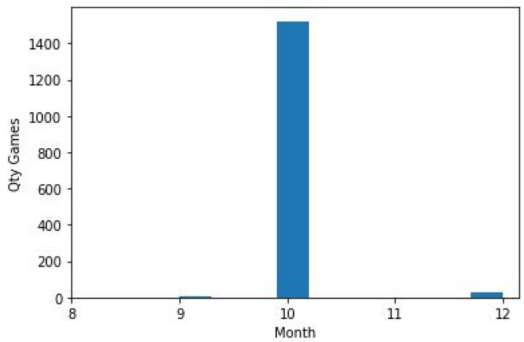
It was discovered that all types of games (pre-season, regular season and postseason) were included in the Kaggle dataset. However this was not clearly indicated in any of the metadata columns.

Through exploration, It was discovered that the type of game can be deduced using the first digit in the GAME_ID field.

- 1 = pre-season
- 2 = regular season
- 4 = post season

The values were parsed into a new dataframe column. A concise check was carried out to ensure accuracy and verify the results. Below shows examining the month of game, against the type for pre-season games:

```python
# Spot checking Pre Season

preseason_games_df = games_type_df[games_type_df['GAME_TYPE_CODE'] == 1]

plt.hist(preseason_games_df.dt_MONTH)
plt.xticks(np.arange(8,13,1))
plt.xlabel('Month')
plt.ylabel('Qty Games')

plt.show()

#preseason_games_df.dt_MONTH.plot(kind='hist')

```



Being an NBA fan, I know that the typical NBA season starts in late October or early November. Therefore the preseason games would take place beforehand. The examination of source data is consistent with this, except for a few outliers in September and December. The September games are acceptable as they take place at the end of September which aligns with the “normal” timeline. The December outliers are more interesting and after further investigation were found to be as a result of the “strike” season (where the season didn’t start until January).

Changing The Grain - “Individual Team Games”

From the original source data, each single row represented a single team’s result for a given game. In other words, each game had 2 records. However I recognized in order to carry out my analysis, I would require each data row to represent the results of *both* teams for a given game -- resulting in a different level of granularity aka “grain”.

I relabelled the columns by appending “_home” and “_opp” denoting the home and away teams accordingly. Additional columns were added to aid with downstream analysis such as indicating who was the home team and who was the winning team. Finally each line pair was joined together (on GAME_ID) to give the resulting dataframe, *indv_team_games_df*

```
indv_team_games_df
# 43268 rows
```

	AST	AST_opp	FG3_PCT	FG3_PCT_opp	FG_PCT	FG_PCT_opp	FT_PCT	FT_PCT_opp	GAMES	GAME_DATE_EST	...	REB	REB_opp
0	22.0	20.0	0.229	0.226	0.354	0.402	0.900	0.762	1	2020-03-01	...	47.0	61.0
1	19.0	28.0	0.310	0.275	0.364	0.468	0.400	0.632	1	2020-03-01	...	57.0	56.0
2	25.0	27.0	0.542	0.488	0.592	0.505	0.805	0.650	1	2020-03-01	...	37.0	37.0
3	38.0	24.0	0.500	0.263	0.566	0.461	0.700	0.897	1	2020-03-01	...	41.0	36.0
4	18.0	23.0	0.257	0.429	0.407	0.413	0.885	0.667	1	2020-03-01	...	51.0	42.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
43263	21.0	13.0	0.222	0.167	0.440	0.308	0.719	0.743	1	2014-10-29	...	44.0	50.0
43264	19.0	23.0	0.125	0.379	0.407	0.448	0.808	0.773	1	2014-10-29	...	43.0	42.0
43265	17.0	20.0	0.364	0.235	0.381	0.406	0.762	0.484	1	2014-10-28	...	56.0	62.0
43266	17.0	23.0	0.381	0.500	0.487	0.529	0.842	0.813	1	2014-10-28	...	33.0	38.0
43267	22.0	16.0	0.414	0.300	0.425	0.354	0.680	0.795	1	2014-10-28	...	47.0	36.0

43268 rows × 32 columns

For clarity, the new dataframe will have 2 entries per game. The difference for each row is the home/away teams are swapped. For example, on March 1 2020 between LAC and PHI (GameID = 21900897):

```
# spot checking the new dataframe

# https://watch.nba.com/game/20200301/PHILAC
indv_team_games_df.query('GAME_ID == 21900897')

# this infact is the correct data!
```

	GAME_DATE_EST	GAME_ID	TEAM_ID	SEASON	PTS	FG_PCT	FT_PCT	FG3_PCT	AST	REB
2	2020-03-01	21900897	1610612746	2019	136.0	0.592	0.805	0.542	25.0	37.0
21636	2020-03-01	21900897	1610612746	2019	130.0	0.505	0.650	0.488	27.0	37.0

dt_YEAR	TEAM_ABBR	TEAM_CITY	TEAM_NICKNAME	GAME_TYPE_CODE	GAME_TYPE	HomeAway	IsHomeTeam	IsWinner	GAMES
2020	LAC	Los Angeles	Clippers	2	Regular Season	Home	1	1	1
2020	PHI	Philadelphia	76ers	2	Regular Season	Away	0	0	1

Aggregation

To prepare for exploratory data analysis (EDA), the regular season game scores were aggregated by team, by season. The aggregate function .sum() was applied to the stats (PTS, AST, REB). Then with the cumulative stats, a per game ratio was calculated by dividing the total by the total number of games played. This this represented by the dataframe *RegSeasonTeamSeason*

```
RegSeasonTeamSeason['PTS_per_Game'] = RegSeasonTeamSeason['PTS'] / RegSeasonTeamSeason['GAMES']
RegSeasonTeamSeason['AST_per_Game'] = RegSeasonTeamSeason['AST'] / RegSeasonTeamSeason['GAMES']
RegSeasonTeamSeason['REB_per_Game'] = RegSeasonTeamSeason['REB'] / RegSeasonTeamSeason['GAMES']
RegSeasonTeamSeason['WIN_pct'] = RegSeasonTeamSeason['WINS'] / RegSeasonTeamSeason['GAMES']
```

```
RegSeasonTeamSeason
```

		PTS	AST	REB	WINS	GAMES	PTS_per_Game	AST_per_Game	REB_per_Game	WIN_pct
TEAM_ABBR	**SEASON**									
ATL	**2003**	7611.0	1648.0	3503.0	28	82	92.817073	20.097561	42.719512	0.341463
	2004	7605.0	1614.0	3435.0	13	82	92.743902	19.682927	41.890244	0.158537
	2005	7972.0	1625.0	3301.0	26	82	97.219512	19.817073	40.256098	0.317073
	2006	7680.0	1573.0	3288.0	30	82	93.658537	19.182927	40.097561	0.365854
	2007	8054.0	1804.0	3462.0	37	82	98.219512	22.000000	42.219512	0.451220
...	...	...	...	...	...	...	...	...	...	...
WAS	**2015**	8534.0	2005.0	3431.0	41	82	104.073171	24.451220	41.841463	0.500000
	2016	8953.0	1956.0	3514.0	49	82	109.182927	23.853659	42.853659	0.597561
	2017	8742.0	2065.0	3536.0	43	82	106.609756	25.182927	43.121951	0.524390
	2018	9350.0	2154.0	3473.0	32	82	114.024390	26.268293	42.353659	0.390244
	2019	6840.0	1500.0	2485.0	22	59	115.932203	25.423729	42.118644	0.372881

509 rows × 9 columns

Initial Findings

Through the exploration phase, it was discovered that the Kaggle source data has the shot data pre-aggregated (unfortunately). This means that only the overall shooting percentages are available without showing the number of attempts and misses.

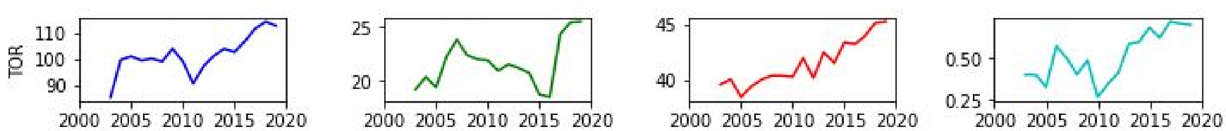
Exploratory data analysis (EDA) was carried out. Below are some interesting findings which will serve as a starting point for further statistical analysis.

Champions are an Exclusive Club

Over 19 seasons, only 9 teams have won a championship (out of 30 total teams):
SAS x4, DET x1, MIA x3, BOS x1, LAL x2, DAL x1, GSW x3, CLE x1 and TOR x1.

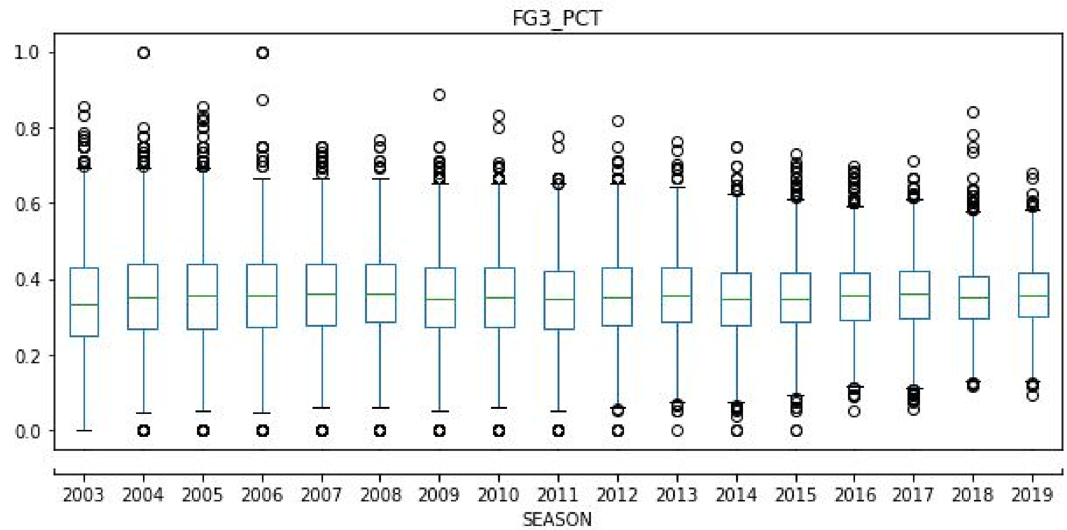
Scoring is Trending Up

From 2010 and onward, it appears that scoring (total PTS and AST) have an upwards trend. Below are for the Toronto Raptors (with remaining team plots available in the Appendix).



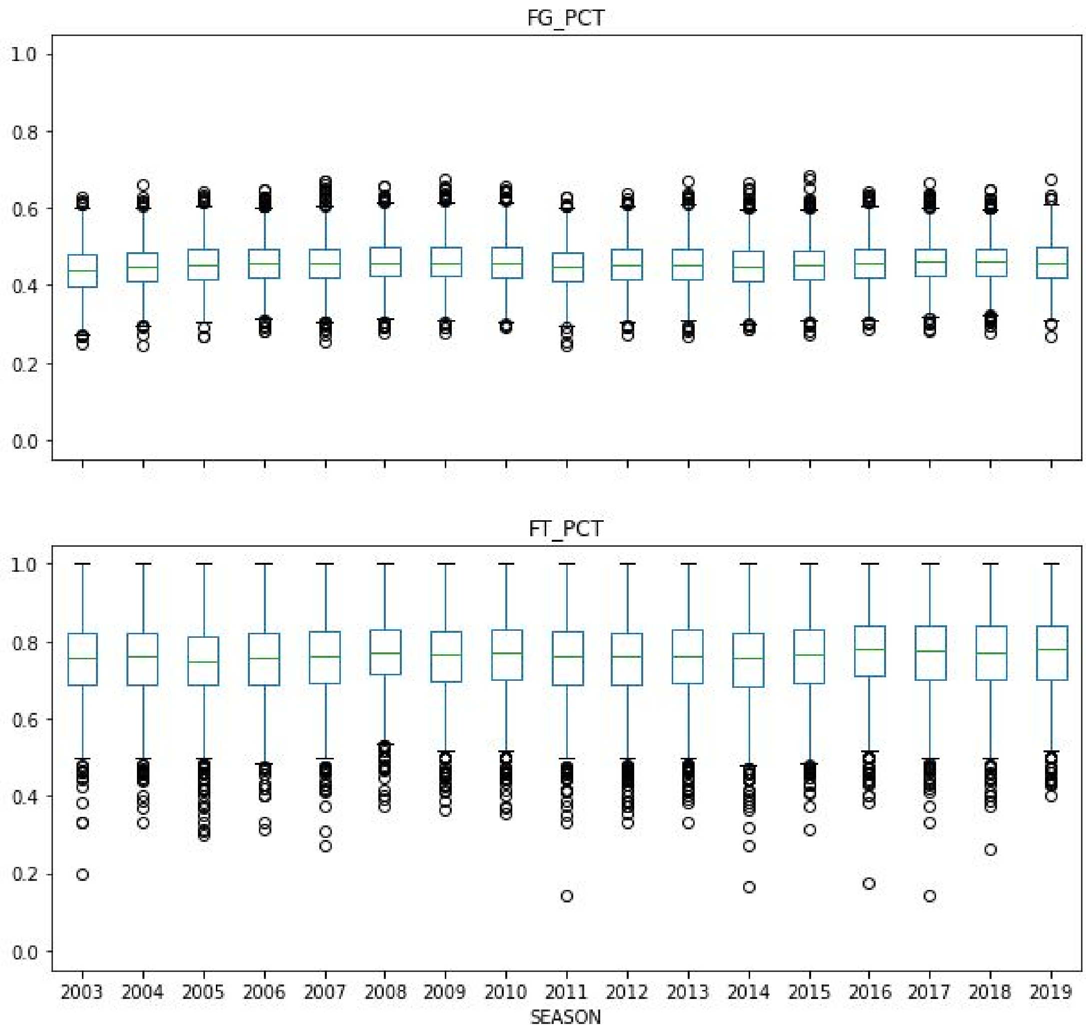
3PT Accuracy is Improving

The league as a whole is better (percentage wise) at 3 pt shooting. The box and whisker plots show less spread over time.



FG and FT Accuracy is Steady

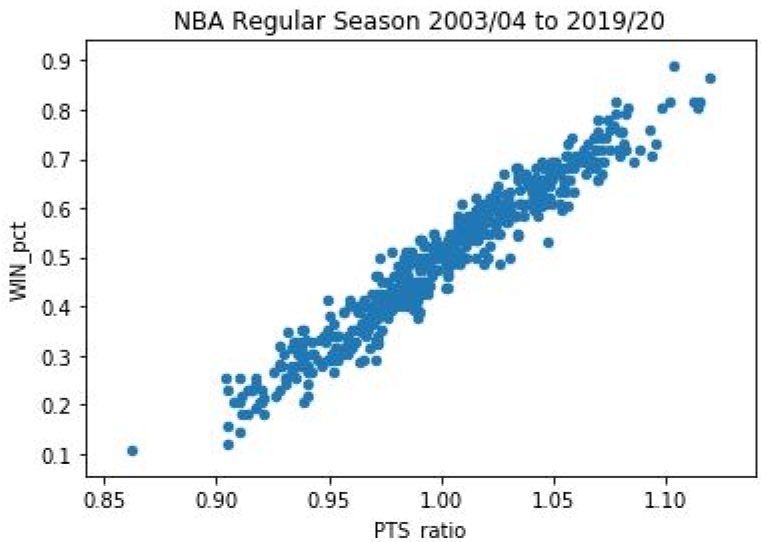
However with 2FG FT pct, no noticeable trends from the bar plots:



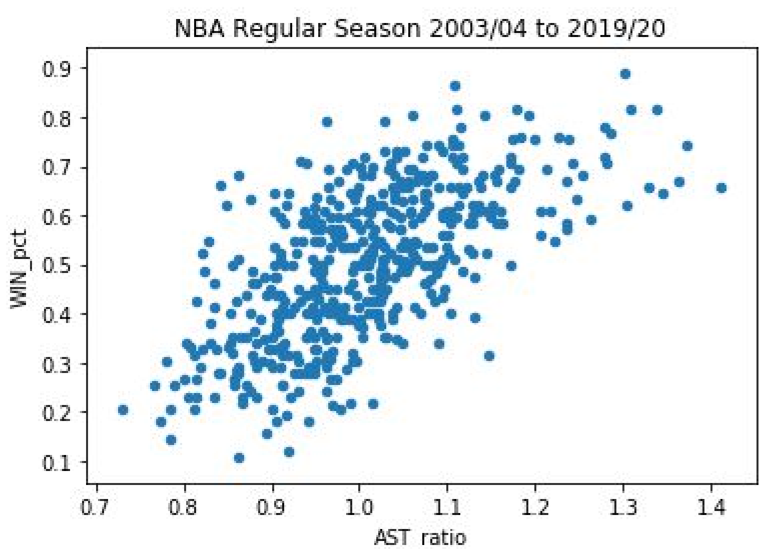
Correlations

The ratio of each stat (defined as points for vs. against) was plotted against the win percentage in an attempt to identify if any correlations might exist.

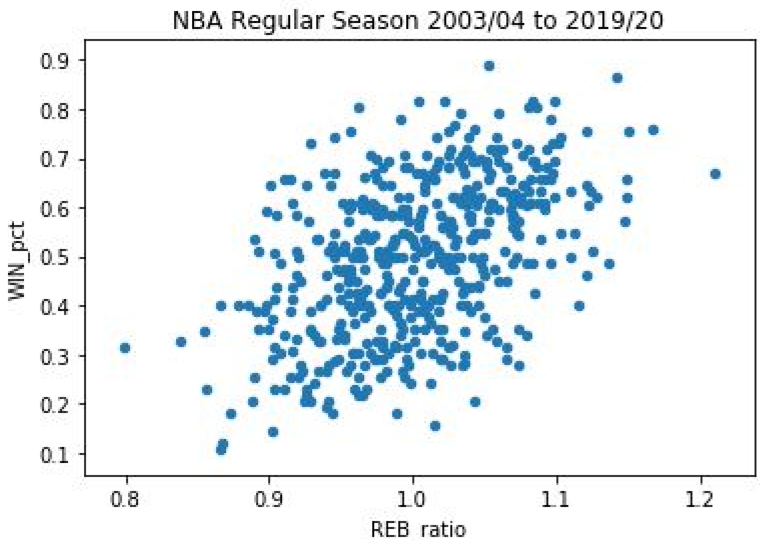
- The higher the PTS ratio, the higher the win pct (This should go without saying).



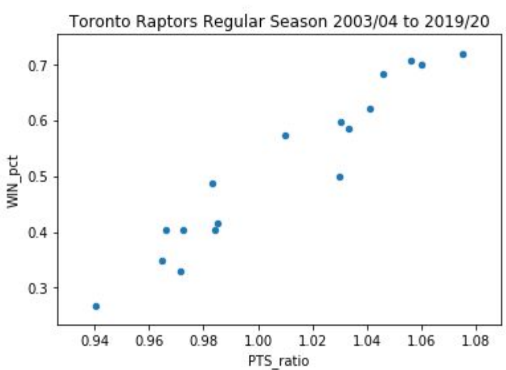
- The higher the AST ratio, the higher the win pct. This is obvious too but relative to the PTS ratio, there is more spread in the data.

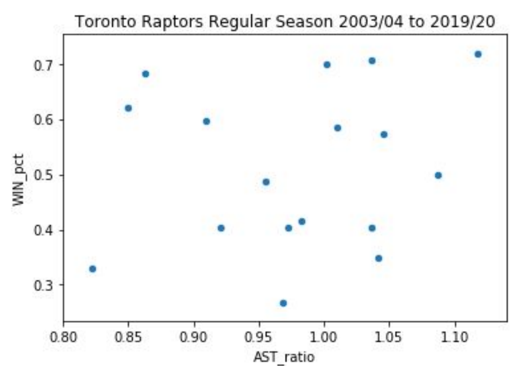


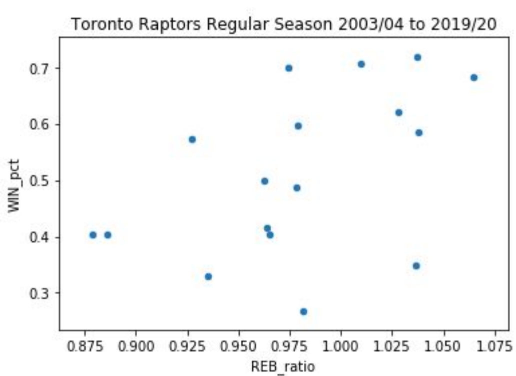
- The higher the REB ratio, the higher the win pct. However this had the most spread amongst the 3 stats.



- Finally out of curiosity, I filtered out only the Toronto Raptors data and found while PTS had a strong correlation to win percentage, AST and REB did not.

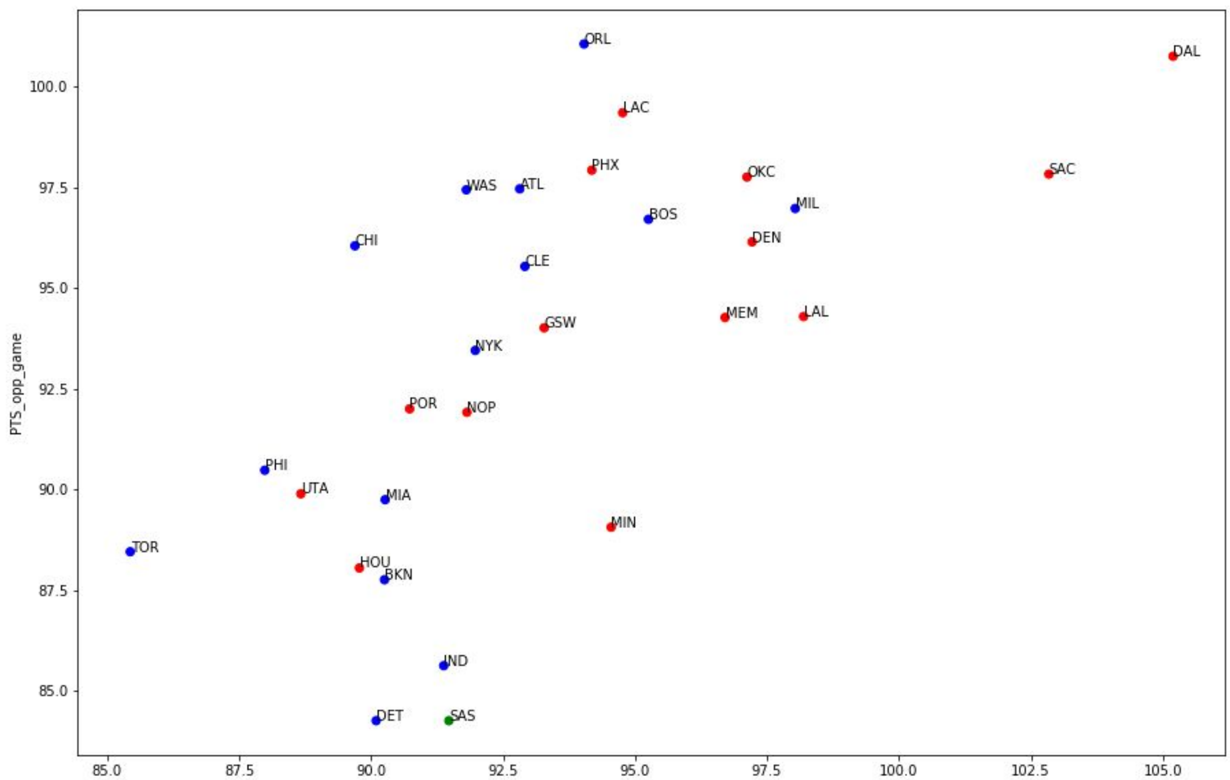






High Scoring Doesn't Necessarily Mean Championships

For the 2003-04 season, I plotted the points for and points against for each team. That year, the SAS were the eventual champions however were one of the lower scoring teams in the league. Instead they were able to offset this by having the lowest opponent points allowed.



Statistical Analysis

I decided to carry out further statistical analysis on the scoring trend increasing over time. From initial analysis, it appears that most teams have had a steady increase in PTS per game since 2010. Has it really changed?

Hypothesis

H0 - Null Hypothesis

There is *no* difference in the means between score before vs after the 2010 season. In other words, the avg pts scored in the NBA has not changed over the years.

HA - Alternative Hypothesis

There *is* a difference in the means between score before vs after the 2010 season.

Test Statistic

I split the original dataframe into 2: 2010 and before, 2011 and after. With each subset, I calculated the mean pts and difference in mean:

```
In [27]:
empirical_diff_mean = diff_of_means(games_after.PTS, games_before.PTS)
empirical_diff_mean

Out[27]: 5.003299481180363
```

```
In [47]:
np.mean(games_after.PTS)

Out[47]: 102.93476518672864
```

95% CI and p-value

Next we need to shift the observed data so that both data sets have the same mean. We do this because our null hypothesis assumes that there is no difference between them, hence they should have the same mean. This way each dataset has the same mean, but still keep their respective std deviations.

Finally a new array of values containing the shifted mean is sampled with replacement (bootstrapping) with N=10,000 to ultimately obtain a p-value and 95% CI.

In [28]:
```
mean_pts = np.mean(games_before.PTS)
mean_pts
```
Out[28]:
```
97.93146570554828
```

In [30]:
```
# Generate shifted arrays
games_before_shifted = games_before.PTS - np.mean(games_before.PTS) + mean_pts
games_before_shifted
```
Out[30]:
```
5620     83.0
5621     89.0
5622     92.0
5623     96.0
5624     84.0
         ...
36405    83.0
36406    92.0
36407    74.0
36408    82.0
36409    93.0
Name: PTS, Length: 18312, dtype: float64
```

In [32]:
```
games_after_shifted = games_after.PTS - np.mean(games_after.PTS) + mean_pts
games_after_shifted
```
Out[32]:
```
0         79.996701
1         85.996701
2        130.996701
3        127.996701
4        100.996701
            ...
43263     89.996701
43264     83.996701
43265     78.996701
43266     94.996701
43267    102.996701
Name: PTS, Length: 24956, dtype: float64
```

In [35]:
```python
bs_replicates_games_before_mean = draw_bs_reps(data=games_before_shifted, func=np.mean, size=N_rep)
```

In [36]:
```python
bs_replicates_games_after_mean = draw_bs_reps(data=games_after_shifted, func=np.mean, size=N_rep)
```

In [48]:
```python
bs_replicates_mean = bs_replicates_games_after_mean - bs_replicates_games_before_mean
bs_replicates_mean
```

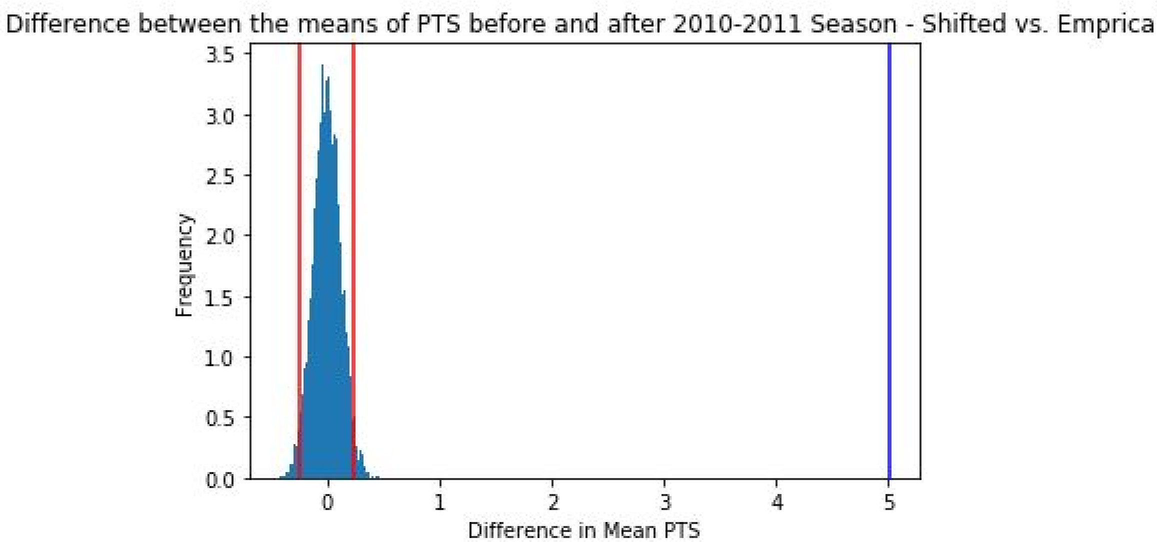
Out[48]:
```
array([-0.14860086,  0.06712067, -0.08813161, ...,  0.05327338,
       -0.15989478,  0.03760627])
```

In [49]:
```python
# Compute and print p-value: p

p = np.sum(np.abs(bs_replicates_mean) > empirical_diff_mean) / len(bs_replicates_mean)

print('p-value =', p)
```

```
p-value = 0.0
```

In [50]:
```python
print(np.percentile(a=bs_replicates_mean, q=2.5), np.percentile(a=bs_replicates_mean, q=97.5))
```

```
-0.2427966762349353 0.24138836602309152
```



Conclusion

The p-value tells you that there is about a 0.0% chance that you would get the difference of means observed in the experiment if mean scores were exactly the same. Therefore we reject the null hypothesis and accept the alternative hypothesis.

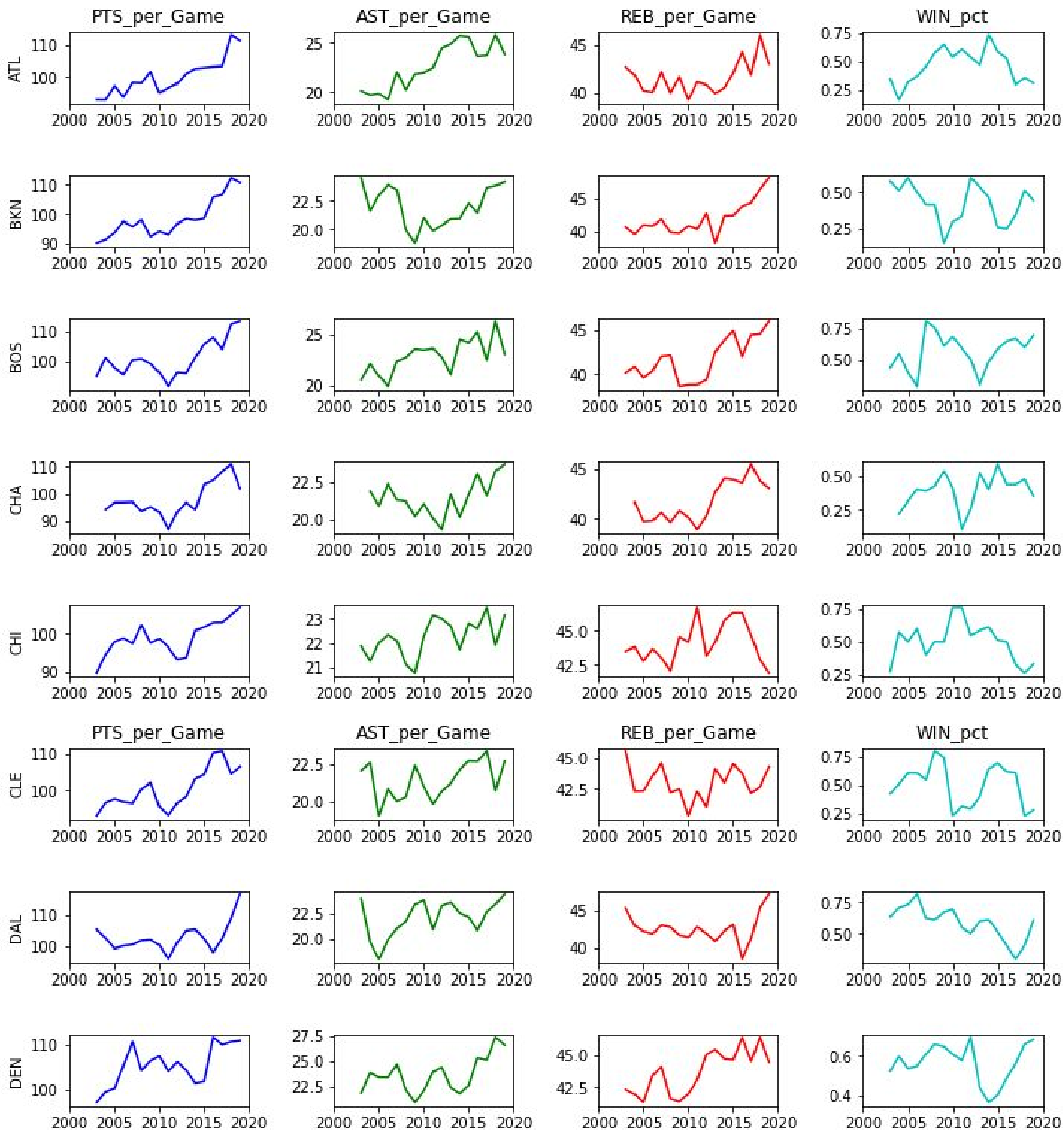
In other words, scoring has gone up since 2010 (statistically significant) and the trend we are seeing is not a random event!

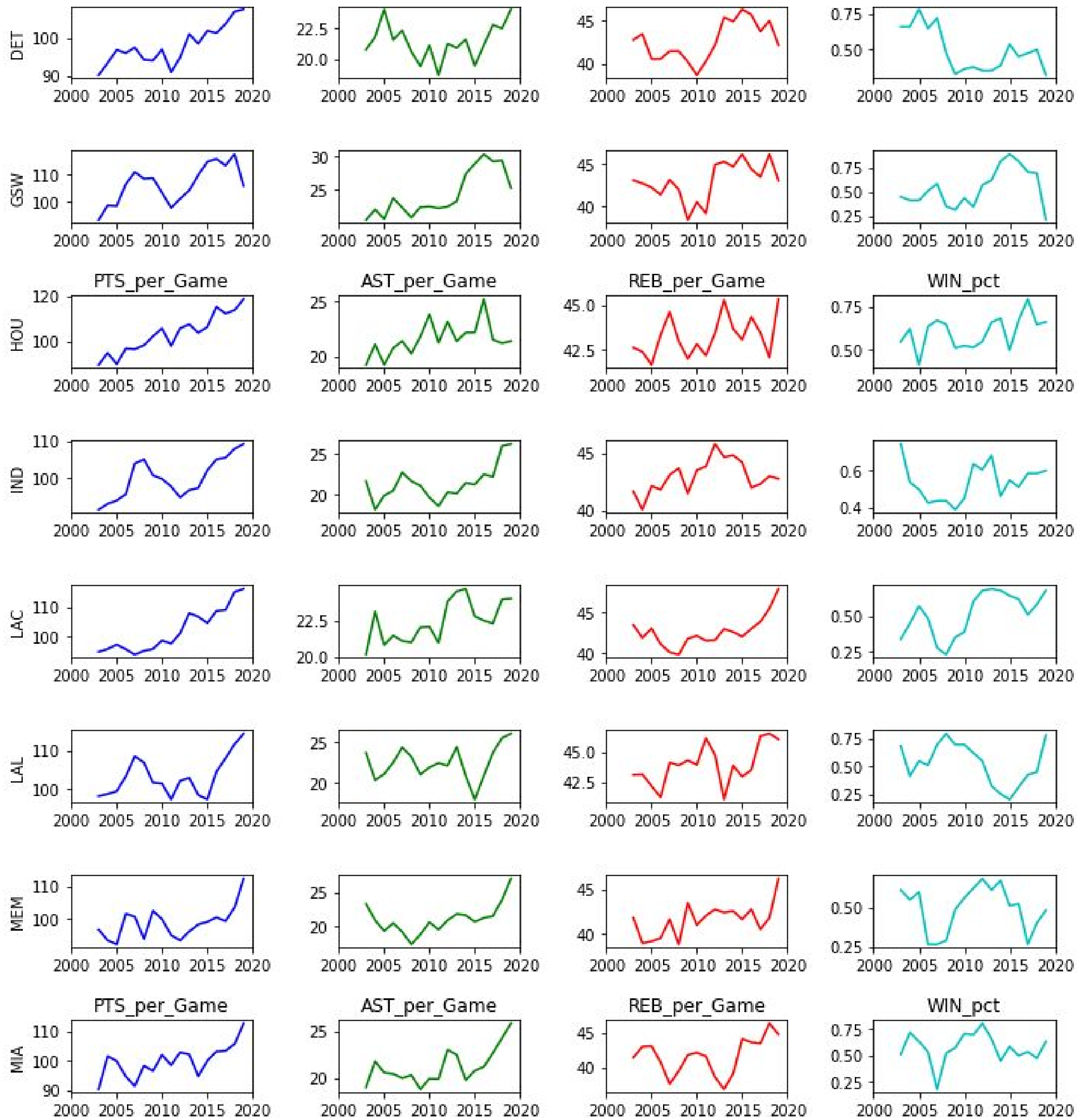
As a fan, this is consistent with what I know about the NBA: pace of play is up with teams being faster and players being less confined to traditional position slots while teams are attempting (and making) more 3PT shots.

Appendix

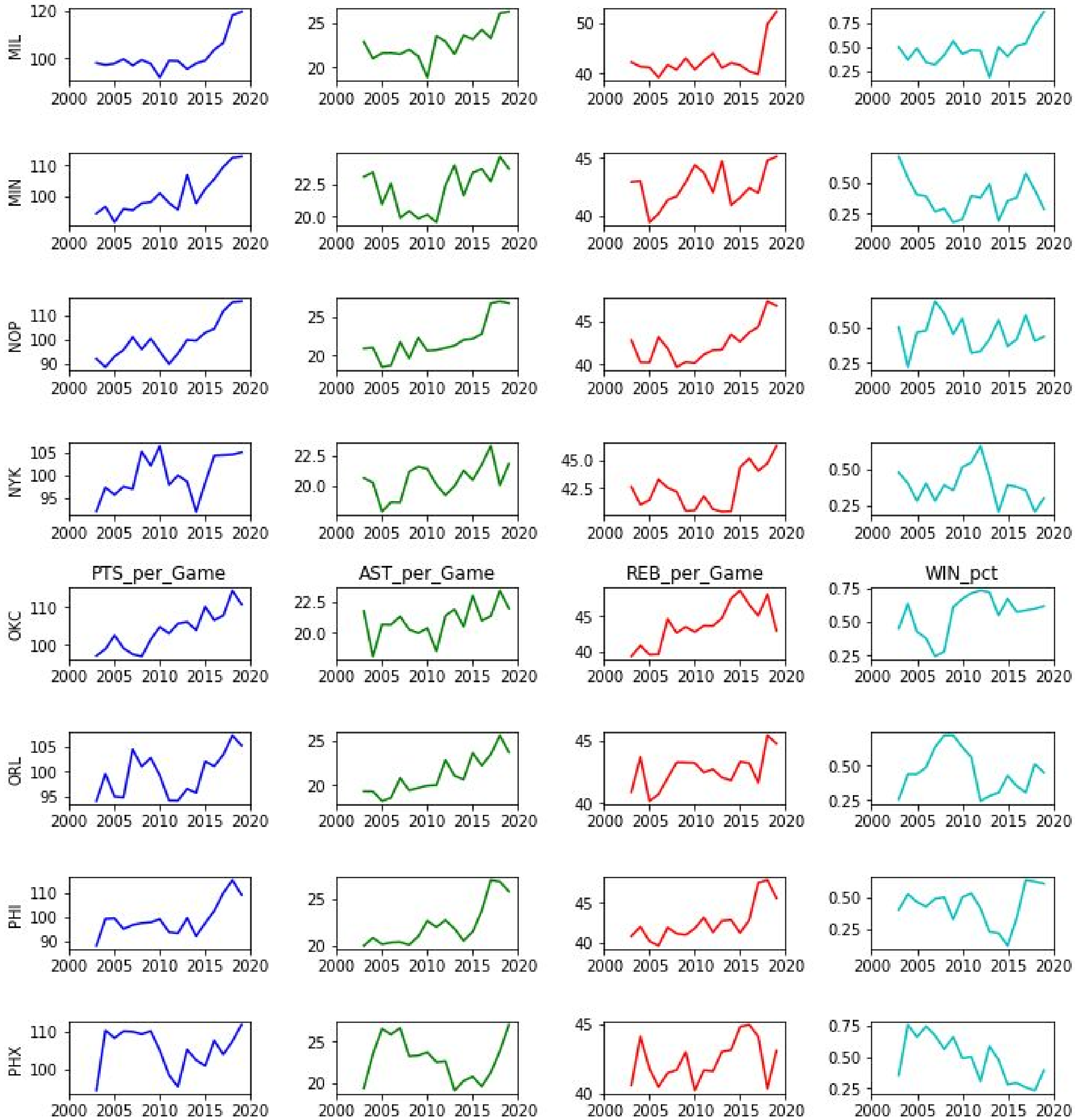
1 - Line Plots

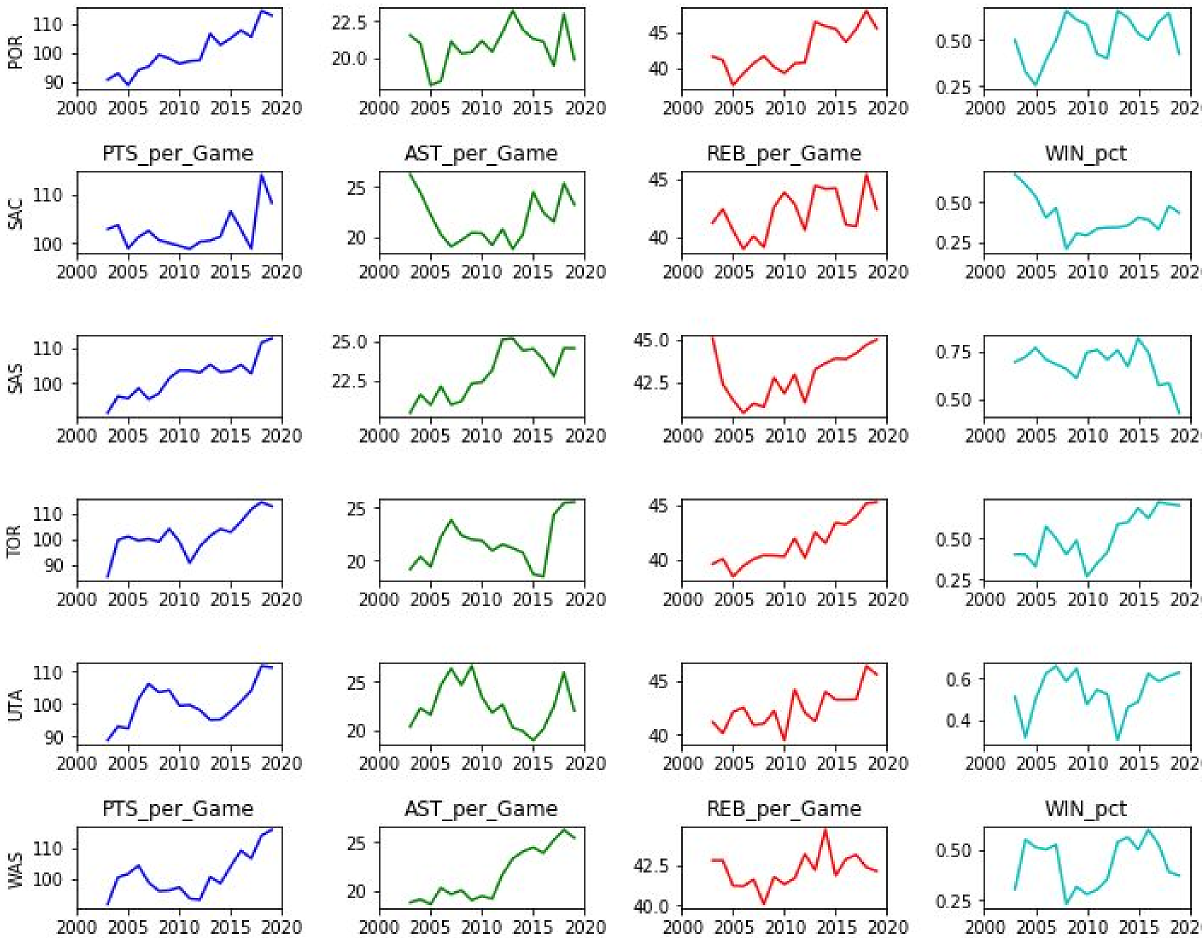
Plots for PTS, AST, REB and Win Percentage from 2003-2004 to 2019-2020 per team.





PTS_per_Game
AST_per_Game
REB_per_Game
WIN_pct
DET
GSW
HOU
IND
LAC
LAL
MEM
MIA


MIL
MIN
NOP
NYK
OKC
ORL
PHI
PHX
PTS_per_Game
AST_per_Game
REB_per_Game
WIN_pct
2000 2005 2010 2015 2020


PTS_per_Game
AST_per_Game
REB_per_Game
WIN_pct
POR
SAC
SAS
TOR
UTA
WAS
PTS_per_Game
AST_per_Game
REB_per_Game
WIN_pct

2 - Bar Plots

Field Goal Percentage and Free Throw Percentage (all teams) from 2003-04 to 2019-20