w13 CS241, Carlos W Mercado

In [1]:
```
import pandas as pd
import os
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
pd.options.display.max_columns = None
```

In [3]:
```
players_original = pd.read_csv('basketball_players.csv', low_memory=False)
```

In [4]:
```
players = pd.read_csv('basketball_players.csv', low_memory=False)
players
```

Out[4]:

	playerID	year	stint	tmID	lgID	GP	GS	minutes	points	oRebounds	dRebounds	rebounds	assists	steals	blocks	turn
0	abramjo01	1946	1	PIT	NBA	47	0	0	527	0	0	0	35	0	0	
1	aubucch01	1946	1	DTF	NBA	30	0	0	65	0	0	0	20	0	0	
2	bakerno01	1946	1	CHS	NBA	4	0	0	0	0	0	0	0	0	0	
3	baltihe01	1946	1	STB	NBA	58	0	0	138	0	0	0	16	0	0	
4	barrjo01	1946	1	STB	NBA	58	0	0	295	0	0	0	54	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
23746	kaisero01	1962	0	PHT	ABL1	27	0	978	467	0	0	140	75	0	0	
23747	spragbr01	1962	0	PHT	ABL1	27	0	746	356	0	0	216	21	0	0	
23748	tayloro02	1962	0	PHT	ABL1	28	0	1007	355	0	0	107	134	0	0	
23749	wellsra01	1962	0	PHT	ABL1	2	0	36	4	0	0	6	3	0	0	
23750	wrighle01	1962	0	PHT	ABL1	28	0	813	195	0	0	257	32	0	0	

23751 rows × 42 columns

(1) Some players score a lot of points because they attempt a lot of shots. Among players that have scored a lot of points, are there some that are much more efficient (points per attempt) than others?

In [5]:
```
# fg: field goals
# ft: free throws
players_2 = players[['playerID', 'fgAttempted', 'fgMade', 'ftAttempted', 'ftMade', 'threeAttempted', 'threeMa
de']]
# Creating a new Data Frame from the original one
players_score = pd.DataFrame(players_2)
```

In [6]:
```
# Getting 3 new fields
sum_attempted = players_score['fgAttempted'] + players_score['ftAttempted'] + players_score['threeAttempted']
sum_made = players_score['fgMade'] + players_score['ftMade'] + players_score['threeMade']
# Adding 2 new columns
players_score['gAttempted'] = sum_attempted
players_score['gMade'] = sum_made
efficiency_scores = players_score['gMade'] / players_score['gAttempted'] * 100
```

In [7]:

```
# Adding 1 new column
players_score['gEfficiency'] = efficiency_scores.round(2)
players_score
```

Out[7]:

	playerID	fgAttempted	fgMade	ftAttempted	ftMade	threeAttempted	threeMade	gAttempted	gMade	gEfficiency
0	abramjo01	834	202	178	123	0	0	1012	325	32.11
1	aubucch01	91	23	35	19	0	0	126	42	33.33
2	bakerno01	1	0	0	0	0	0	1	0	0.00
3	baltihe01	263	53	69	32	0	0	332	85	25.60
4	barrjo01	438	124	79	47	0	0	517	171	33.08
...	...	...	...	...	...	...	...	...	...	...
23746	kaisero01	385	159	145	124	56	25	586	308	52.56
23747	spragbr01	318	147	96	61	7	1	421	209	49.64
23748	tayloro02	361	134	95	82	26	5	482	221	45.85
23749	wellsra01	4	1	3	2	0	0	7	3	42.86
23750	wrighle01	194	75	103	44	3	1	300	120	40.00

23751 rows × 10 columns

In [ ]:

```
# Who are the more efficient players?
```

(2) It seems like some players may excel in one statistical category,but produce very little in other areas. Are there any players that are exceptional across many categories?

In [8]:

```
# Let's just get the colums we are interested in
players_reduced = players_original[['playerID', 'points', 'rebounds', 'assists', 'steals', 'blocks', 'turnovers']]
players_reduced
```

Out[8]:

	playerID	points	rebounds	assists	steals	blocks	turnovers
0	abramjo01	527	0	35	0	0	0
1	aubucch01	65	0	20	0	0	0
2	bakerno01	0	0	0	0	0	0
3	baltihe01	138	0	16	0	0	0
4	barrjo01	295	0	54	0	0	0
...	...	...	...	...	...	...	...
23746	kaisero01	467	140	75	0	0	0
23747	spragbr01	356	216	21	0	0	0
23748	tayloro02	355	107	134	0	0	0
23749	wellsra01	4	6	3	0	0	0
23750	wrighle01	195	257	32	0	0	0

23751 rows × 7 columns

In [9]:

```
# There are repeated playerIDs, let's group them by name
players_grouped= players_reduced.groupby('playerID').sum()
players_grouped
```

Out[9]:

	points	rebounds	assists	steals	blocks	turnovers
playerID						
abdelal01	1465	846	85	71	70	247
abdulka01	38387	17440	5660	1160	3189	2527
abdulma01	9087	2146	3555	26	6	0
abdulma02	8553	1087	2079	487	46	963
abdulta01	1830	776	266	184	83	309
...	...	...	...	...	...	...
ziegeba01	45	0	0	0	0	0
zimmede01	4	4	7	0	0	4
zoetji01	2	8	1	1	3	4
zopfbi01	118	46	73	0	0	0
zunicma01	604	0	50	0	0	0

4903 rows × 6 columns

In [10]:

```
# Let's get the >90% values fot the following categories
stats_excel = players_grouped[['points', 'rebounds', 'assists', 'steals', 'blocks', 'turnovers']].quantile(0.
9)
stats_excel
# This will provide a threshold to compare how good all players are
```

Out[10]:

```
points       7714.4
rebounds     3208.0
assists      1660.4
steals        489.4
blocks        230.0
turnovers     963.0
Name: 0.9, dtype: float64
```

In [11]:

```
# Now, let's find the most exceptional players, those with some stats above the 90% of all the other players'
stats
players_excel = players_grouped[
    (players_grouped['points'] > stats_excel['points']) &
    (players_grouped['rebounds'] > stats_excel['rebounds']) &
    (players_grouped['assists'] > stats_excel['assists']) &
    (players_grouped['steals'] > stats_excel['steals']) &
    (players_grouped['blocks'] > stats_excel['blocks']) &
    (players_grouped['turnovers'] > stats_excel['turnovers'])
]
players_excel
```

Out[11]:

	points	rebounds	assists	steals	blocks	turnovers
playerID						
abdulka01	38387	17440	5660	1160	3189	2527
abdursh01	15028	6239	2109	820	638	2134
adamsal01	13910	6937	4012	1289	809	2194
aguirma01	18458	4578	2870	688	296	2306
anderni01	11529	4064	2087	1114	364	1358
...	...	...	...	...	...	...
webbech01	17182	8124	3526	1197	1200	2313
wilkeja01	14644	5117	2050	1049	262	1211
wilkido01	26668	7169	2677	1378	642	2669
willihe01	11944	6509	1856	605	1605	1929
worthja01	16320	4708	2791	1041	624	1859

124 rows × 6 columns

In [12]:

```
# Let's sort them by the points column (this is the criteria I'm going to use to get the best of the bests)
players_excel.sort_values(by=['points'], ascending=False)
```

Out[12]:

	points	rebounds	assists	steals	blocks	turnovers
playerID						
abdulka01	38387	17440	5660	1160	3189	2527
malonka01	36928	14968	5248	2085	1145	4524
jordami01	32292	6672	5633	2514	893	2924
ervinju01	30026	10525	5176	2272	1941	3940
malonmo01	29580	17834	1936	1199	1889	4264
...	...	...	...	...	...	...
mccraro01	9014	5087	2750	585	493	1419
foxri01	8966	3517	2649	967	355	1611
olberma01	8940	5033	2332	623	272	1650
kirilan01	8411	3837	1919	960	1382	1344
horryro01	7715	5269	2343	1158	1035	1378

124 rows × 6 columns

In [13]:

```python
# Let's play the top 10 of all times
players_excel.head(10)
```

Out[13]:

playerID	points	rebounds	assists	steals	blocks	turnovers
abdulka01	38387	17440	5660	1160	3189	2527
abdursh01	15028	6239	2109	820	638	2134
adamsal01	13910	6937	4012	1289	809	2194
aguirma01	18458	4578	2870	688	296	2306
anderni01	11529	4064	2087	1114	364	1358
anthoca01	15926	4092	2010	722	298	1933
artesro01	11755	3907	2463	1545	459	1651
ballagr01	9953	4858	1733	877	256	1059
barklch01	23757	12546	4215	1648	888	3376
barryri01	25279	6863	4952	1104	269	1364

(3) Much has been said about the rise of the three-point shot in recent years. It seems that players are shooting and making more three-point shots than ever. Recognizing that this dataset doesn't contain the very most recent data, do you see a trend of more three-point shots either across the league or among certain groups of players? Is there a point at which popularity increased dramatically?

In [14]:

```python
# Let's take a look at the original dataframe
players_original
```

Out[14]:

	playerID	year	stint	tmID	lgID	GP	GS	minutes	points	oRebounds	dRebounds	rebounds	assists	steals	blocks	turn
0	abramjo01	1946	1	PIT	NBA	47	0	0	527	0	0	0	35	0	0	
1	aubucch01	1946	1	DTF	NBA	30	0	0	65	0	0	0	20	0	0	
2	bakerno01	1946	1	CHS	NBA	4	0	0	0	0	0	0	0	0	0	
3	baltihe01	1946	1	STB	NBA	58	0	0	138	0	0	0	16	0	0	
4	barrjo01	1946	1	STB	NBA	58	0	0	295	0	0	0	54	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
23746	kaisero01	1962	0	PHT	ABL1	27	0	978	467	0	0	140	75	0	0	
23747	spragbr01	1962	0	PHT	ABL1	27	0	746	356	0	0	216	21	0	0	
23748	tayloro02	1962	0	PHT	ABL1	28	0	1007	355	0	0	107	134	0	0	
23749	wellsra01	1962	0	PHT	ABL1	2	0	36	4	0	0	6	3	0	0	
23750	wrighle01	1962	0	PHT	ABL1	28	0	813	195	0	0	257	32	0	0	

23751 rows × 42 columns

In [15]:

```python
# Get only the columns we need
players_reduced = pd.DataFrame(players_original[['year', 'threeMade', 'threeAttempted']])
# Sum all three points information available
threeShots = players_reduced['threeMade'] + players_reduced['threeAttempted']
players_reduced['threeShots'] = threeShots
players_reduced
```

Out[15]:

	year	threeMade	threeAttempted	threeShots
0	1946	0	0	0
1	1946	0	0	0
2	1946	0	0	0
3	1946	0	0	0
4	1946	0	0	0
...	...	...	...	...
23746	1962	25	56	81
23747	1962	1	7	8
23748	1962	5	26	31
23749	1962	0	0	0
23750	1962	1	3	4

23751 rows × 4 columns

In [16]:

```
# Let's sum all the threeShots made by year, it doesn't mader which player made it nor when he made it.
threeShots_byYear = players_reduced.groupby('year').sum()
threeShots_byYear
# 75 years of NBA games
```

Out[16]:

	threeMade	threeAttempted	threeShots
year			
1937	0	0	0
1938	0	0	0
1939	0	0	0
1940	0	0	0
1941	0	0	0
...	...	...	...
2007	16173	44502	60675
2008	16440	44420	60860
2009	15822	44622	60444
2010	15988	44555	60543
2011	12726	36502	49228

75 rows × 3 columns

In [17]:

```
# But before 1961 three point shots were not used, so let's get rid of the years prior to 1961
threeShots_byYear = threeShots_byYear[threeShots_byYear['threeShots'] > 0]
threeShots_byYear
```

Out[17]:

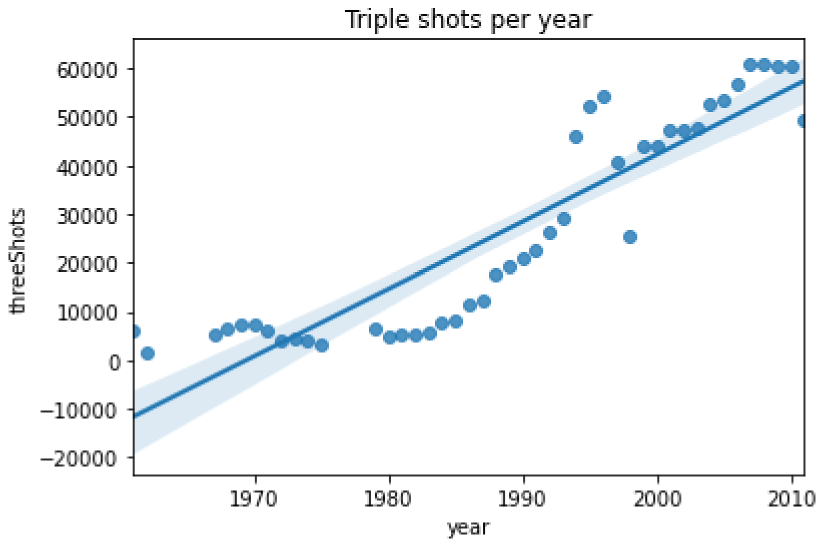
	threeMade	threeAttempted	threeShots
year			
1961	1416	4627	6043
1962	403	1386	1789
1967	1223	4285	5508
1968	1515	5060	6575
1969	1702	5842	7544
1970	1695	5675	7370
1971	1443	4857	6300
1972	914	3160	4074
1973	995	3512	4507
1974	911	3108	4019
1975	706	2395	3101
1979	1403	5003	6406
1980	936	3815	4751
1981	1129	4308	5437
1982	1011	4248	5259
1983	1120	4484	5604
1984	1671	5917	7588
1985	1774	6293	8067
1986	2687	8913	11600
1987	2979	9421	12400
1988	4332	13431	17763
1989	4829	14608	19437
1990	5055	15812	20867
1991	5587	16898	22485
1992	6668	19824	26492
1993	7301	21907	29208
1994	12153	33889	46042
1995	14000	38161	52161
1996	14383	39943	54326
1997	10450	30231	40681
1998	6463	19080	25543
1999	11513	32614	44127
2000	11524	32597	44121
2001	12402	35074	47476
2002	12200	34912	47112
2003	12321	35492	47813
2004	13777	38748	52525
2005	14086	39313	53399
2006	14926	41671	56597
2007	16173	44502	60675
2008	16440	44420	60860
2009	15822	44622	60444
2010	15988	44555	60543
2011	12726	36502	49228

In [18]:

```
# Now let's plot three shots made by year
sns.regplot(data=threeShots_byYear, x=threeShots_byYear.index, y='threeShots').set_title('Triple shots per ye
ar')
# Aroung 1995 there a huge increment on triple shots
```

Out[18]:

```
Text(0.5, 1.0, 'Triple shots per year')
```



In [ ]:

```
```