Table of Contents

0.1 If you're viewing this as a pdf:

Unfortunately, some of my 3D interactive graphs and my bar chart race do not show up properly in PDF format. For those who just want a quick overview of the project, however, I've still provided this PDF file for your convenience.

1 Overview

This notebook explores various data across the careers of Lebron James, Michael Jordan, and Kobe Bryant. The data comes from Kaggle and involves 6 csv files that display different statistics about each player's career.

Here's a link to the data (https://www.kaggle.com/xvivancos/michael-jordan-kobe-bryant-and-lebron-james-stats/activity)

2 Imports & Loading Data

In [1]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import bar_chart_race as bcr
import numpy as np
import plotly.express as px
import cpi
import plotly.graph_objects as go
import warnings
from pandas.core.common import SettingWithCopyWarning
```

```
C:\Users\student\Anaconda3\lib\site-packages\statsmodels\tools\_testing.py:19: FutureWarning: pandas.util.t
esting is deprecated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
C:\Users\student\Anaconda3\lib\site-packages\cpi\__init__.py:46: StaleDataWarning: CPI data is out of date.
To accurately inflate to today's dollars, you must run `cpi.update()`.
  warnings.warn(StaleDataWarning())
```

In [2]:

```
advanced_stats = pd.read_csv('data/advanced_stats.csv')
allgames_stats = pd.read_csv('data/allgames_stats.csv')
allstar = pd.read_csv('data/allstar_games_stats.csv')
game_highs = pd.read_csv('data/game_highs_stats.csv')
per_game = pd.read_csv('data/per_game_stats.csv')
salaries = pd.read_csv('data/salaries.csv')
total_stats = pd.read_csv('data/totals_stats.csv')
```

3 General Analysis & Data Exploration

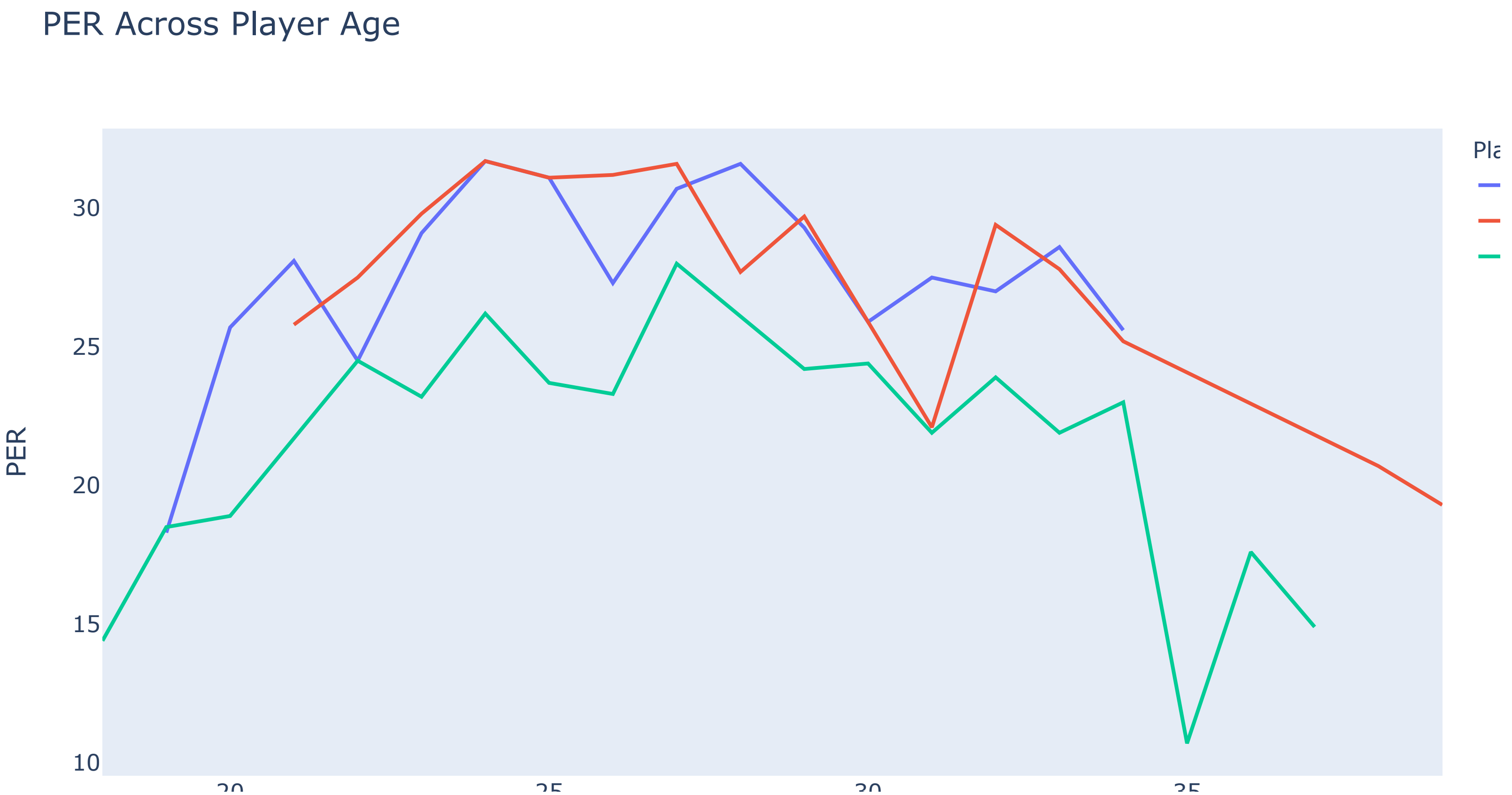
3.1 PER Across Player Age

For basic exploratory purposes, I've made a lineplot of advanced_stats and correlation heatmap of game highs.

For the lineplot, I analyzed player PER (player efficiency rating) across all ages of each player. PER is a simple, but widely used basketball statistic in which many positive statistics a player can acquire (points, rebounds, blocks, etc) are multiplied by a 'weight' value and then summed up. This sum is then subtracted by a few negative statistics (fouls, turnovers, etc.) that are also multiplied by weights. This calculation returns PER - the average PER of current nba players is around 15, for reference.

In [3]:

```
advanced_stats = advanced_stats[advanced_stats['RSorPO'] == 'Regular Season']

fig = px.line(advanced_stats, x="Age", y="PER", color="Player",
                 title="PER Across Player Age")
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
fig.show()
```



3.2 Correlation Heatmap

3.2 Correlation Heatmap

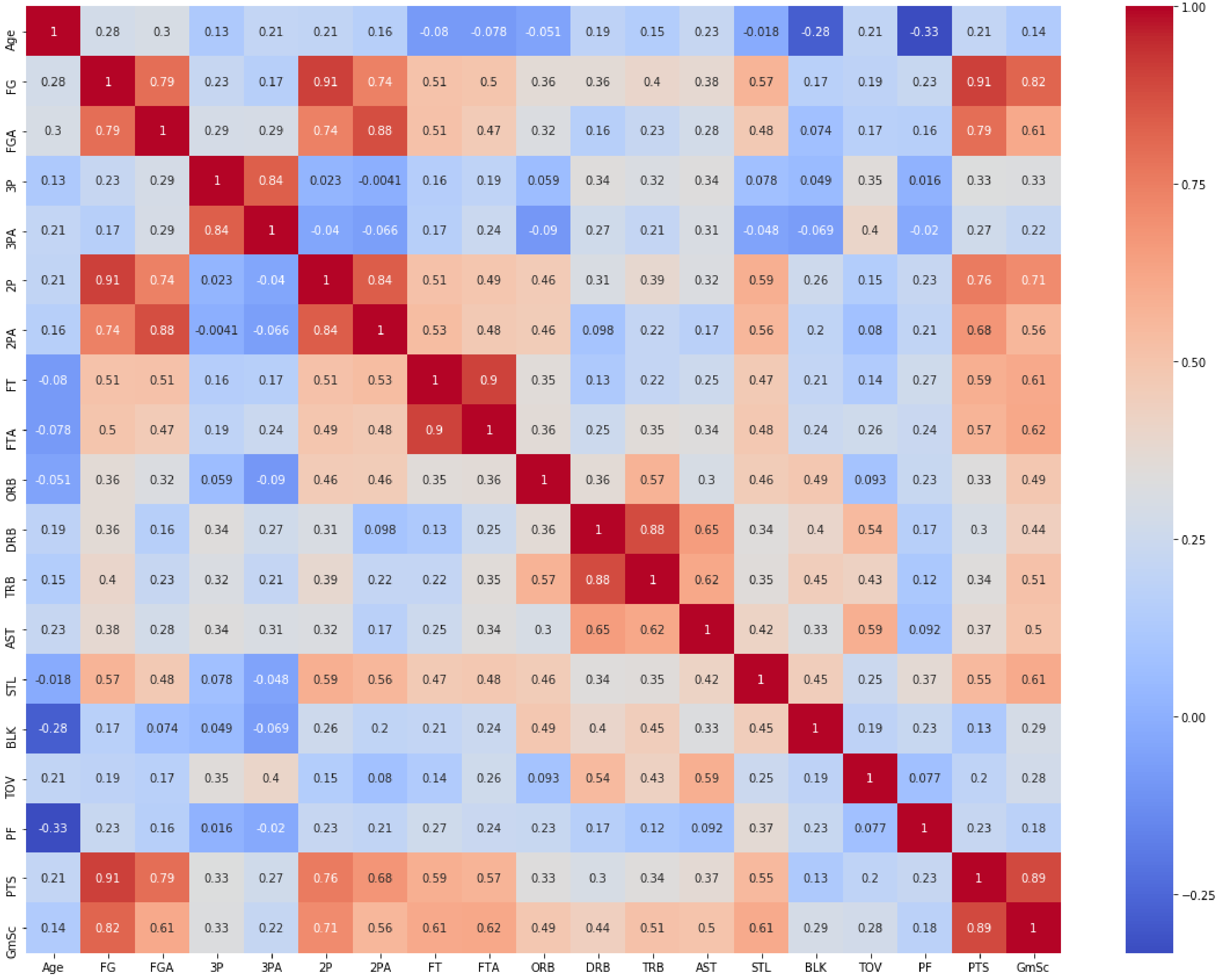
For the heatmap, I just wanted to see if there were any strongly correlated or anticorrelated features. I looked through a few of the dataframes, but game highs was the most readable, so I've placed it here.

In [4]:

```
plt.subplots(figsize=(20,15))
sns.heatmap(game_highs.corr(), cmap= 'coolwarm', annot=True)
```

Out[4]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x23f8a972048>
```



	Age	FG	FGA	3P	3PA	2P	2PA	FT	FTA	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS	GmSc
Age	1	0.28	0.3	0.13	0.21	0.21	0.16	-0.08	-0.078	-0.051	0.19	0.15	0.23	-0.018	-0.28	0.21	-0.33	0.21	0.14
FG	0.28	1	0.79	0.23	0.17	0.91	0.74	0.51	0.5	0.36	0.36	0.4	0.38	0.57	0.17	0.19	0.23	0.91	0.82
FGA	0.3	0.79	1	0.29	0.29	0.74	0.88	0.51	0.47	0.32	0.16	0.23	0.28	0.48	0.074	0.17	0.16	0.79	0.61
3P	0.13	0.23	0.29	1	0.84	0.023	-0.0041	0.16	0.19	0.059	0.34	0.32	0.34	0.078	0.049	0.35	0.016	0.33	0.33
3PA	0.21	0.17	0.29	0.84	1	-0.04	-0.066	0.17	0.24	-0.09	0.27	0.21	0.31	-0.048	-0.069	0.4	-0.02	0.27	0.22
2P	0.21	0.91	0.74	0.023	-0.04	1	0.84	0.51	0.49	0.46	0.31	0.39	0.32	0.59	0.26	0.15	0.23	0.76	0.71
2PA	0.16	0.74	0.88	-0.0041	-0.066	0.84	1	0.53	0.48	0.46	0.098	0.22	0.17	0.56	0.2	0.08	0.21	0.68	0.56
FT	-0.08	0.51	0.51	0.16	0.17	0.51	0.53	1	0.9	0.35	0.13	0.22	0.25	0.47	0.21	0.14	0.27	0.59	0.61
FTA	-0.078	0.5	0.47	0.19	0.24	0.49	0.48	0.9	1	0.36	0.25	0.35	0.34	0.48	0.24	0.26	0.24	0.57	0.62
ORB	-0.051	0.36	0.32	0.059	-0.09	0.46	0.46	0.35	0.36	1	0.36	0.57	0.3	0.46	0.49	0.093	0.23	0.33	0.49
DRB	0.19	0.36	0.16	0.34	0.27	0.31	0.098	0.13	0.25	0.36	1	0.88	0.65	0.34	0.4	0.54	0.17	0.3	0.44
TRB	0.15	0.4	0.23	0.32	0.21	0.39	0.22	0.22	0.35	0.57	0.88	1	0.62	0.35	0.45	0.43	0.12	0.34	0.51
AST	0.23	0.38	0.28	0.34	0.31	0.32	0.17	0.25	0.34	0.3	0.65	0.62	1	0.42	0.33	0.59	0.092	0.37	0.5
STL	-0.018	0.57	0.48	0.078	-0.048	0.59	0.56	0.47	0.48	0.46	0.34	0.35	0.42	1	0.45	0.25	0.37	0.55	0.61
BLK	-0.28	0.17	0.074	0.049	-0.069	0.26	0.2	0.21	0.24	0.49	0.4	0.45	0.33	0.45	1	0.19	0.23	0.13	0.29
TOV	0.21	0.19	0.17	0.35	0.4	0.15	0.08	0.14	0.26	0.093	0.54	0.43	0.59	0.25	0.19	1	0.077	0.2	0.28
PF	-0.33	0.23	0.16	0.016	-0.02	0.23	0.21	0.27	0.24	0.23	0.17	0.12	0.092	0.37	0.23	0.077	1	0.23	0.18
PTS	0.21	0.91	0.79	0.33	0.27	0.76	0.68	0.59	0.57	0.33	0.3	0.34	0.37	0.55	0.13	0.2	0.23	1	0.89
GmSc	0.14	0.82	0.61	0.33	0.22	0.71	0.56	0.61	0.62	0.49	0.44	0.51	0.5	0.61	0.29	0.28	0.18	0.89	1

3.3 Player Age by Points and Field Goal Percentage

Here, I wanted to explore how each player's points and field goal percentage changed as they got older. We can see that, in general, Lebron James has had the best field goal percentage which seemed to get better as he got older. Kobe Bryant's field goal percentage was typically the lowest. Both retired players (Kobe Bryant and Michael Jordan) saw significant field goal percentage decrease in their last two years in the NBA.

In [5]:

```
fig = px.scatter_3d(total_stats, x='Age', y='PTS', z='FG%',
              color='Player', title='Player Points & FG% by Age')


fig.show() # title
```

Player Points & FG% by Age

4 Scatter Plot

4.1 Data Cleaning/Feature Engineering

In the following cells, I merge the salaries and advanced stats dataframe so I can use PER as a metric later in our visualization. I then format salary so it's a more readable field.

In [6]:
```
salaries_full = pd.merge(salaries, advanced_stats,  how='outer', left_on=['Season','Player'], right_on = ['S
```

In [7]:
```
salaries_full.Salary = salaries_full.Salary.str.replace('$', '')
salaries_full.Salary = salaries_full.Salary.apply(lambda x: float(x))
salaries_full['Salary (in Millions)'] = (salaries_full['Salary'].astype(float)/1000000) # for readability
```

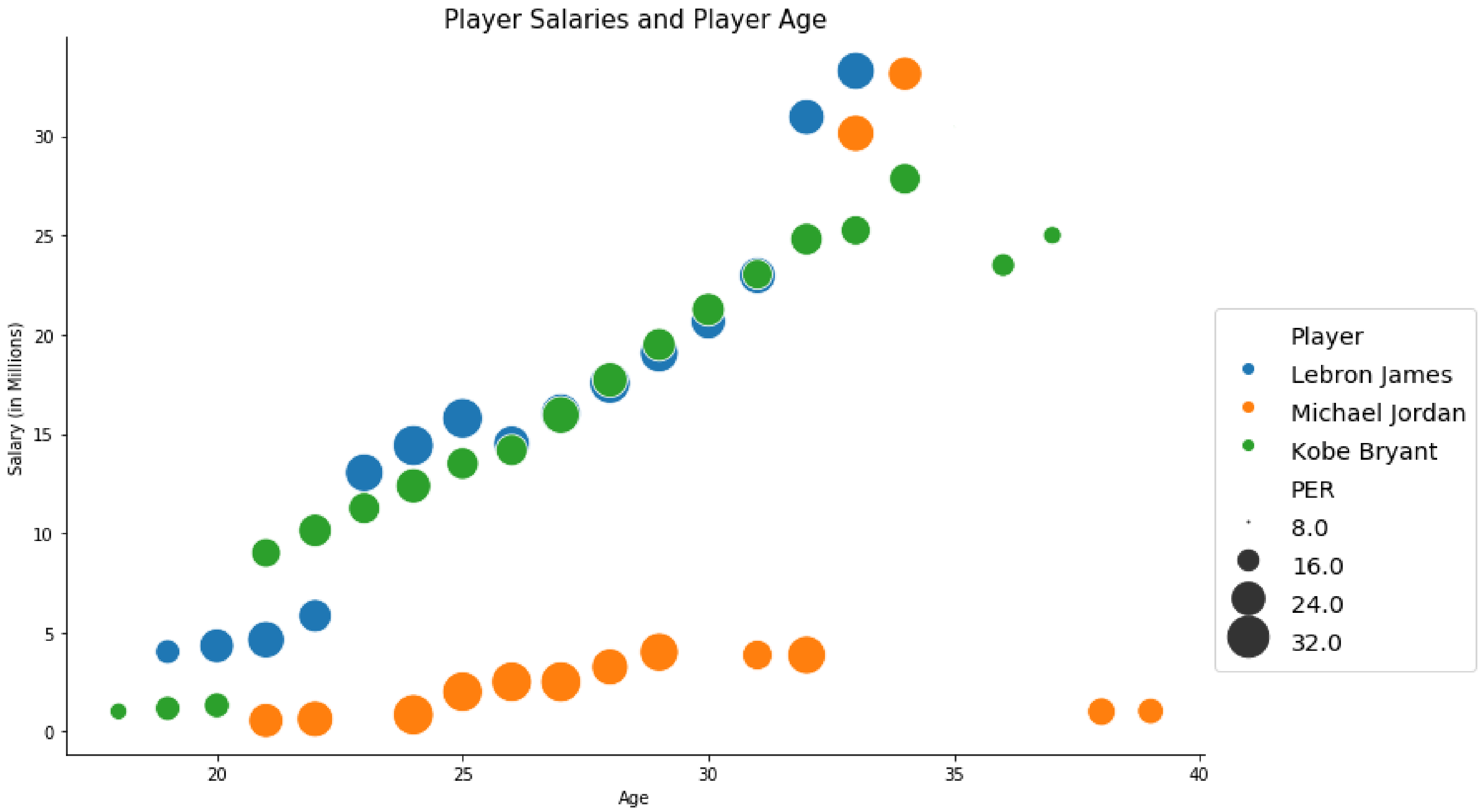
In [8]:
```
salaries_full = salaries_full.dropna(subset=['Age'])
salaries_full.Age = salaries_full.Age.apply(lambda x: int(x))
salaries_full = salaries_full[salaries_full['RSorPO'] == 'Regular Season']
```

4.2 Player Salaries by Age

Here, I plot player salary on the y-axis and player age on the x-axis. PER is used here again, represented by the size of each dot. The goal here is to see how much each player earned at each age (and if they were worth it, based on PER).

```
In [9]:  g = sns.relplot(data=salaries_full[salaries_full['RSorPO'] == 'Regular Season'], x="Age", y="Salary (in Mill
                                 size='PER', sizes=(1,600))

         g.fig.set_size_inches(15,7)
         g._legend.remove()
         plt.legend(fontsize='x-large', ncol=1, handleheight=2, labelspacing=.05, loc = "lower right", bbox_to_anchor
         plt.title("Player Salaries and Player Age", fontsize=15)
```

```
Out[9]: Text(0.5, 1, 'Player Salaries and Player Age')
```



4.3 Salaries, Adjusted for Inflation

To see just how absurd Michael Jordan's contract was in the 1996 and 1997 seasons. I'm using the CPI library to reflect what his contract would be worth in today's dollars.

In [10]:

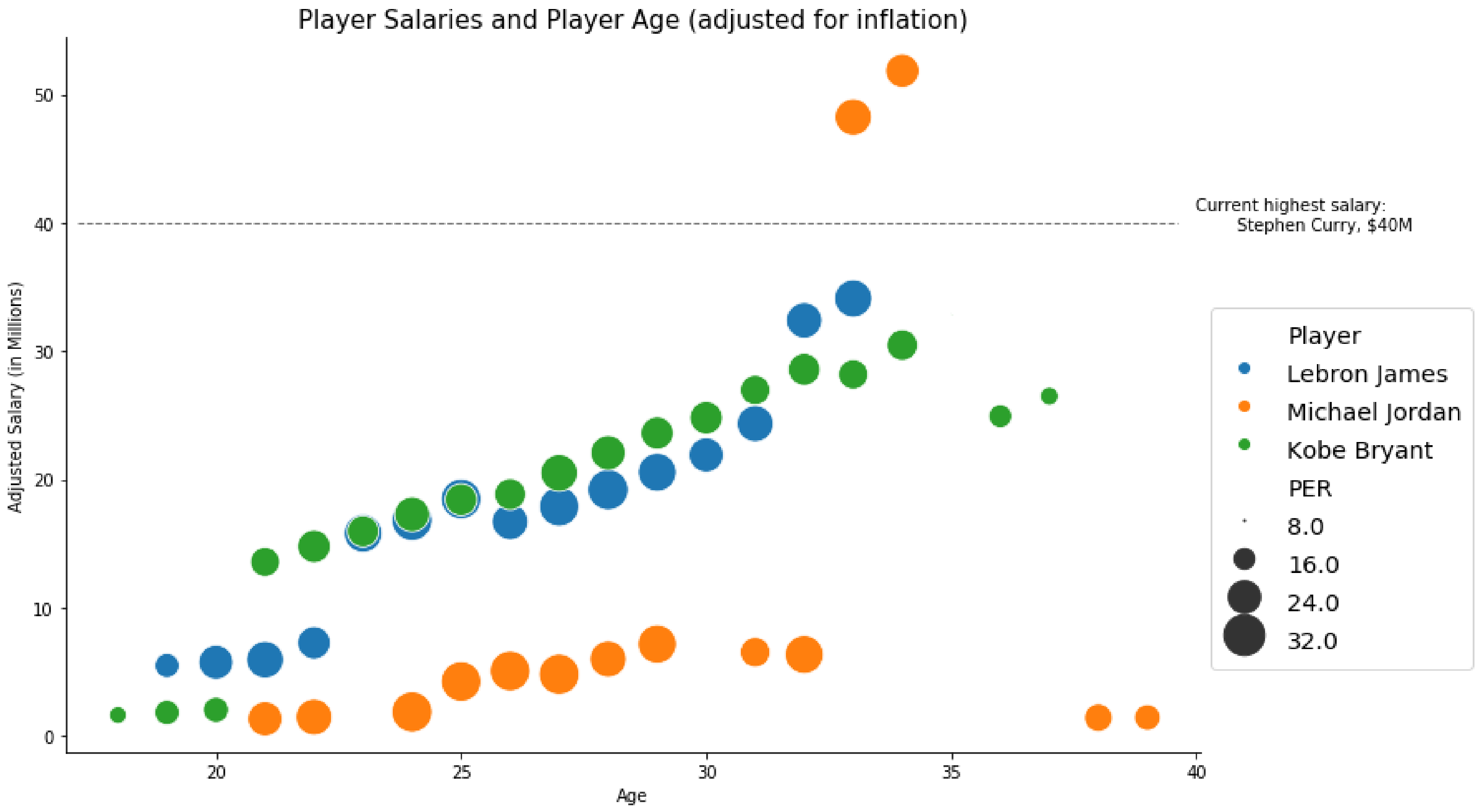
```
salaries_full['inflation_year'] = salaries_full.Season.apply(lambda x: int(x[0:4]))
salaries_full['inflation_year'] = salaries_full['inflation_year'].apply(lambda x: int(x))
salaries_full['adjusted_salary'] = np.round(salaries_full.apply(lambda x: cpi.inflate(x.Salary, x.inflation_
salaries_full['Adjusted Salary (in Millions)'] = (salaries_full['adjusted_salary'].astype(float)/1000000)
```

In [11]:

```
g = sns.relplot(data=salaries_full[salaries_full['RSorPO'] == 'Regular Season'], x="Age", y="Adjusted Salary
                       size='PER', sizes=(1,600))

g.fig.set_size_inches(15,7)
g._legend.remove()
plt.title("Player Salaries and Player Age (adjusted for inflation)", fontsize=15)
plt.annotate("""Current highest salary:
        Stephen Curry, $40M""",
                 xy=(40, 39.5))
plt.axhline(y=40, ls='--', c='grey', lw=1, xmin=0.01, xmax=.98)
plt.legend(fontsize='x-large', ncol=1, handleheight=2, labelspacing=.05, loc = "lower right", bbox_to_anchor
```

Out[11]:

```
<matplotlib.legend.Legend at 0x23f8c5056d8>
```



5 Line Plot w/ Dropdown Menu

5.1 Data Cleaning/Feature Engineering

Data for number of finals, championships, etc. was missing from the dataset, so I add it in manually here. I wanted to do more with some of the first round exits, second round exits, etc. but ended up focusing mainly on championships and just adding functionality to that plot.

In [12]:

```
def count_finals(season,player):
    """count finals appearances for each player"""
    if season == '2010-11' and player == 'Lebron James': return 1
    elif season == '2011-12' and player == 'Lebron James': return 1
    elif season == '2012-13' and player == 'Lebron James': return 1
    elif season == '2006-07' and player == 'Lebron James': return 1
    elif season == '2013-14' and player == 'Lebron James': return 1
    elif season == '2014-15' and player == 'Lebron James': return 1
    elif season == '2015-16' and player == 'Lebron James': return 1
    elif season == '2016-17' and player == 'Lebron James': return 1
    elif season == '2017-18' and player == 'Lebron James': return 1
    elif season == '2019-20' and player == 'Lebron James': return 1

    elif season == '1990-91' and player == 'Michael Jordan': return 1
    elif season == '1991-92' and player == 'Michael Jordan': return 1
    elif season == '1992-93' and player == 'Michael Jordan': return 1
    elif season == '1995-96' and player == 'Michael Jordan': return 1
    elif season == '1996-97' and player == 'Michael Jordan': return 1
    elif season == '1997-98' and player == 'Michael Jordan': return 1

    elif season == '1999-00' and player == 'Kobe Bryant': return 1
    elif season == '2000-01' and player == 'Kobe Bryant': return 1
    elif season == '2001-02' and player == 'Kobe Bryant': return 1
    elif season == '2003-04' and player == 'Kobe Bryant': return 1
    elif season == '2007-08' and player == 'Kobe Bryant': return 1
    elif season == '2008-09' and player == 'Kobe Bryant': return 1
    elif season == '2009-10' and player == 'Kobe Bryant': return 1

    else: return 0


def count_chips(season, player):
    """count finals championships for each player"""
    if season == '2011-12' and player == 'Lebron James': return 1
    elif season == '2012-13' and player == 'Lebron James': return 1
    elif season == '2015-16' and player == 'Lebron James': return 1
    elif season == '2019-20' and player == 'Lebron James': return 1

    elif season == '1990-91' and player == 'Michael Jordan': return 1
    elif season == '1991-92' and player == 'Michael Jordan': return 1
    elif season == '1992-93' and player == 'Michael Jordan': return 1
    elif season == '1995-96' and player == 'Michael Jordan': return 1
    elif season == '1996-97' and player == 'Michael Jordan': return 1
    elif season == '1997-98' and player == 'Michael Jordan': return 1

    elif season == '1999-00' and player == 'Kobe Bryant': return 1
    elif season == '2000-01' and player == 'Kobe Bryant': return 1
    elif season == '2001-02' and player == 'Kobe Bryant': return 1
    elif season == '2008-09' and player == 'Kobe Bryant': return 1
    elif season == '2009-10' and player == 'Kobe Bryant': return 1

    else: return 0


def first_round_exits(season, player):
    """count first round exits for each player"""
    if season == '1984-85' and player == 'Michael Jordan': return 1
    elif season == '1985-86' and player == 'Michael Jordan': return 1
    elif season == '1986-87' and player == 'Michael Jordan': return 1

    elif season == '2005-06' and player == 'Kobe Bryant': return 1
    elif season == '2006-07' and player == 'Kobe Bryant': return 1
    elif season == '2012-13' and player == 'Kobe Bryant': return 1

    else: return 0


def second_round_exits(season, player):
    """count second round exits for each player"""
    if season == '2005-06' and player == 'Lebron James': return 1
    elif season == '2007-08' and player == 'Lebron James': return 1
    elif season == '2009-10' and player == 'Lebron James': return 1

    elif season == '1987-88' and player == 'Michael Jordan': return 1

    elif season == '1996-97' and player == 'Kobe Bryant': return 1
    elif season == '1998-99' and player == 'Kobe Bryant': return 1
    elif season == '2002-03' and player == 'Kobe Bryant': return 1
    elif season == '2010-11' and player == 'Kobe Bryant': return 1
    elif season == '2011-12' and player == 'Kobe Bryant': return 1

    else: return 0


def third_round_exits(season, player):
    """count third round exits for each player"""
    if season == '2008-09' and player == 'Lebron James': return 1

    elif season == '1988-89' and player == 'Michael Jordan': return 1
    elif season == '1989-90' and player == 'Michael Jordan': return 1

    elif season == '1997-98' and player == 'Kobe Bryant': return 1

    else: return 0


def missed_playoffs(finals, first, second, third):
    """counts number of playoff misses"""
    if finals == 0 & first == 0 & second == 0 & third == 0:
        return 1
    else:
        return 0

def playoff_eliminations(chips, finals, first, second, third):
    """counts number of times a player was eliminated from playoffs"""
    if finals == 1 & chips == 0:
        return 1
    elif first == 1:
        return 1
    elif second == 1:
        return 1
    elif third == 1:
        return 1
    else:
        return 0
```

In [13]:

```
# applying functions to new dataframe columns
total_stats['count_chips'] = total_stats.apply(lambda x: count_chips(x.Season, x.Player), axis=1)
total_stats['finals_appearances'] = total_stats.apply(lambda x: count_finals(x.Season, x.Player), axis=1)
total_stats['first_round_exits'] = total_stats.apply(lambda x: first_round_exits(x.Season, x.Player), axis=1
total_stats['second_round_exits'] = total_stats.apply(lambda x: second_round_exits(x.Season, x.Player), axis
total_stats['third_round_exits'] = total_stats.apply(lambda x: third_round_exits(x.Season, x.Player), axis=1
total_stats['missed_playoffs'] = total_stats.apply(lambda x: missed_playoffs(x.finals_appearances, x.first_r
total_stats['playoff_elims_total'] = total_stats.apply(lambda x: playoff_eliminations(x.count_chips, x.final
```

In [14]:

```
# take cumulative sum of each feature
warnings.simplefilter(action="ignore", category=SettingWithCopyWarning)
total_stats = total_stats[total_stats['RSorPO'] == 'Regular Season']

lebron_stats = total_stats[total_stats['Player'] == 'Lebron James']
michael_stats = total_stats[total_stats['Player'] == 'Michael Jordan']
kobe_stats = total_stats[total_stats['Player'] == 'Kobe Bryant']


lebron_stats['Number of Championships'] = lebron_stats['count_chips'].cumsum()
lebron_stats['finals_sum'] = lebron_stats['finals_appearances'].cumsum()
lebron_stats['first_sum'] = lebron_stats['first_round_exits'].cumsum()
lebron_stats['second_sum'] = lebron_stats['second_round_exits'].cumsum()
lebron_stats['third_sum'] = lebron_stats['third_round_exits'].cumsum()
lebron_stats['missed_playoffs_sum'] = lebron_stats['missed_playoffs'].cumsum()
lebron_stats['playoff_elims_sum'] = lebron_stats['playoff_elims_total'].cumsum()
lebron_stats['Minutes_Cumulative'] = lebron_stats['MP'].cumsum()
lebron_stats['Points_Cumulative'] = lebron_stats['PTS'].cumsum()
lebron_stats['Age'] = lebron_stats['Age'].apply(lambda x: int(x))


michael_stats['Number of Championships'] = michael_stats['count_chips'].cumsum()
michael_stats['finals_sum'] = michael_stats['finals_appearances'].cumsum()
michael_stats['first_sum'] = michael_stats['first_round_exits'].cumsum()
michael_stats['second_sum'] = michael_stats['second_round_exits'].cumsum()
michael_stats['third_sum'] = michael_stats['third_round_exits'].cumsum()
michael_stats['missed_playoffs_sum'] = michael_stats['missed_playoffs'].cumsum()
michael_stats['playoff_elims_sum'] = michael_stats['playoff_elims_total'].cumsum()
michael_stats['Minutes_Cumulative'] = michael_stats['MP'].cumsum()
michael_stats['Points_Cumulative'] = michael_stats['PTS'].cumsum()
michael_stats['Age'] = michael_stats['Age'].apply(lambda x: int(x))

kobe_stats['Number of Championships'] = kobe_stats['count_chips'].cumsum()
kobe_stats['finals_sum'] = kobe_stats['finals_appearances'].cumsum()
kobe_stats['first_sum'] = kobe_stats['first_round_exits'].cumsum()
kobe_stats['second_sum'] = kobe_stats['second_round_exits'].cumsum()
kobe_stats['third_sum'] = kobe_stats['third_round_exits'].cumsum()
kobe_stats['missed_playoffs_sum'] = kobe_stats['missed_playoffs'].cumsum()
kobe_stats['playoff_elims_sum'] = kobe_stats['playoff_elims_total'].cumsum()
kobe_stats['Minutes_Cumulative'] = kobe_stats['MP'].cumsum()
kobe_stats['Points_Cumulative'] = kobe_stats['PTS'].cumsum()
kobe_stats['Age'] = kobe_stats['Age'].apply(lambda x: int(x))
```

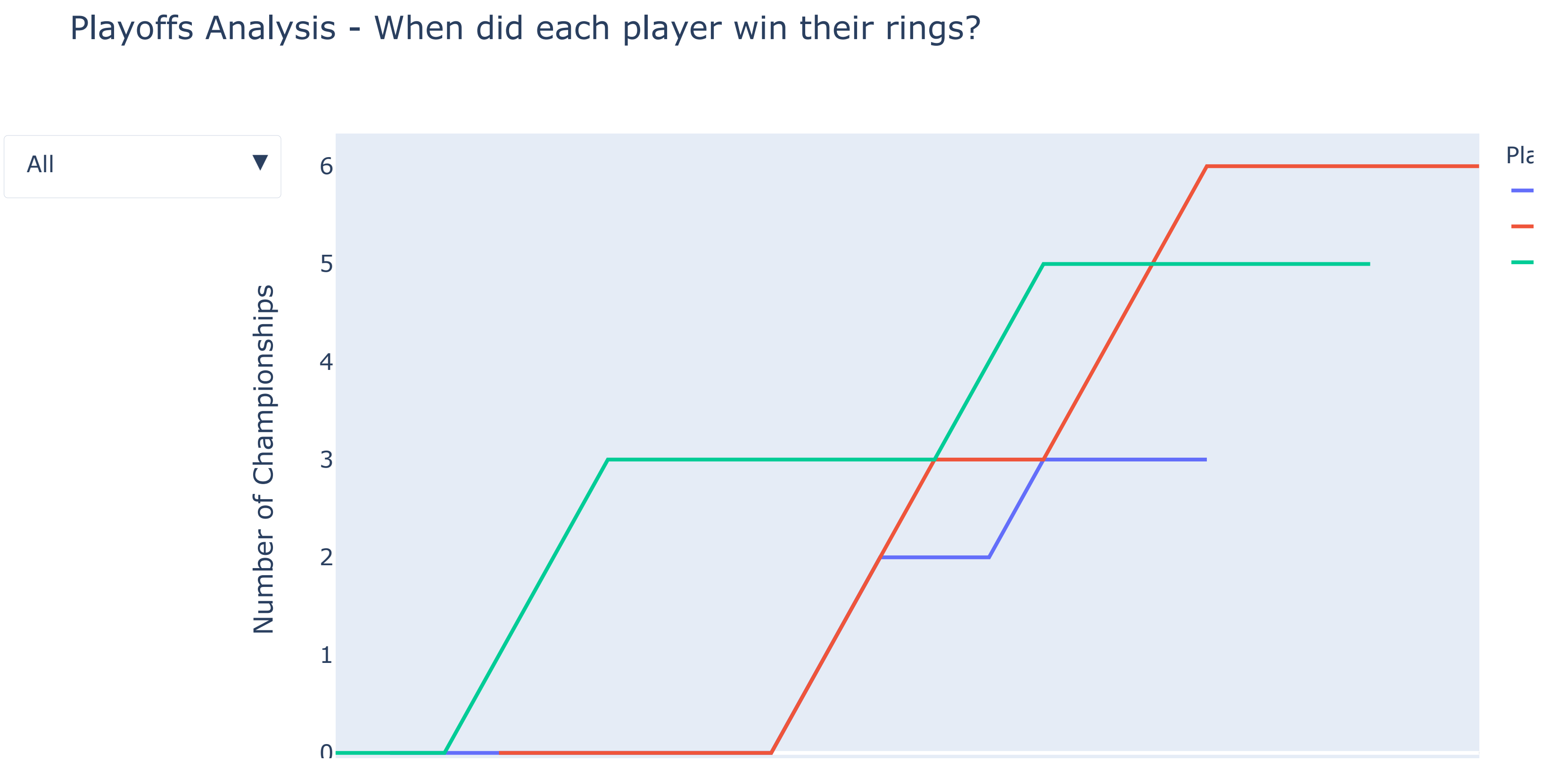
```
total_stats = pd.concat([lebron_stats, michael_stats, kobe_stats], axis=0)
```

5.2 Player Championships Across Age

Here, I implemented a drop down menu that allows you to look at each player's individual championships across time, or compare just two players

In [15]:

```
df = total_stats
fig = px.line(total_stats, x="Age", y="Number of Championships", color="Player",
                 title="Playoffs Analysis - When did each player win their rings?"
                )

fig.update_layout(
    updatemenus=[go.layout.Updatemenu(
        active=0,
        buttons=list(
            [dict(label = 'All',
                  method = 'update',
                  args = [{'visible': [True, True, True]},
                          {'title': 'All',
                           'showlegend':True}]),

             dict(label = 'Lebron James',
                  method = 'update',
                  args = [{'visible': [True, False, False]},
                          {'title': 'Lebron James',
                           'showlegend':True}]),

             dict(label = 'Michael Jordan',
                  method = 'update',
                  args = [{'visible': [False, True, False]},
                          {'title': 'Michael Jordan',
                           'showlegend':True}]),

             dict(label = 'Kobe Bryant',
                  method = 'update',
                  args = [{'visible': [False, False, True]},
                          {'title': 'Kobe Bryant',
                           'showlegend':True}]),

             dict(label = 'James vs. Bryant',
                  method = 'update',
                  args = [{'visible': [True, False, True]},
                         {'title': 'James vs. Bryant',
                           'showlegend':True}]),
             dict(label = 'James vs. Jordan',
                  method = 'update',
                  args = [{'visible': [True, True, False]},
                         {'title': 'James vs. Jordan',
                           'showlegend':True}]),

             dict(label = 'Jordan vs. Bryant',
                  method = 'update',
                  args = [{'visible': [False, True, True]},
                         {'title': 'Jordan vs. Bryant',
                           'showlegend':True}

                         ])
            ])
        )
    ])

fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```



Playoffs Analysis - When did each player win their rings?

6 Bar Chart Race

6.1 Data Cleaning/Feature Engineering

Here, we'll use the bar_chart_race library. We need to change the data to a wide format in order to make the package work

In [16]:

```
total_stats = total_stats[total_stats['RSorPO'] == 'Regular Season'] # only take data from regular season
stats = total_stats[['Player', 'PTS', 'Age']] # new dataframe with only player, points, and age
```

Our "PTS" column only shows each player's points in a given season - we want this to be cumulative. First, we'll split up each player into an individual dataframe. Then, we'll take the cumulative sum and concatenate the dataframes back into one big one.

In [17]:

```
lebron_stats = stats[stats['Player'] == 'Lebron James']
michael_stats = stats[stats['Player'] == 'Michael Jordan']
kobe_stats = stats[stats['Player'] == 'Kobe Bryant']
```

In [18]:

```
# take cumsum of each player's points
lebron_stats = lebron_stats.assign(Points=lebron_stats.PTS.cumsum())
michael_stats = michael_stats.assign(Points=michael_stats.PTS.cumsum())
kobe_stats = kobe_stats.assign(Points=kobe_stats.PTS.cumsum())

stats = pd.concat([lebron_stats, michael_stats, kobe_stats], axis=0)
stats = stats.drop(['PTS'], axis=1) # drop non-cumulative points column
```

Not all players played at all ages. For example, Kobe Bryant started his career at 18, Lebron James started at 19, and Michael Jordan at 22. Michael Jordan retired for a couple of years in the middle of his career. Lebron James is now 34 and has not played his full career yet.

The following adds in some values to make the visualization look better:

- If a player wasn't in the NBA yet, gave them a row with 0 points

- If a player didn't play for a couple of years (Michael Jordan), I used their previous season's point total for that season
- If a player didn't play as long as the others, I added their final season's cumulative points

In [19]:

```
row1 = {'Player':'Lebron James', 'Points':0, 'Age': 18}
row2 = {'Player':'Lebron James', 'Points':32543, 'Age': 35}
row3 = {'Player':'Lebron James', 'Points':32543, 'Age': 36}
row4 = {'Player':'Lebron James', 'Points':32543, 'Age': 37}
row5 = {'Player':'Lebron James', 'Points':32543, 'Age': 38}
row6 = {'Player':'Lebron James', 'Points':32543, 'Age': 39}
row7 = {'Player':'Michael Jordan', 'Points':0, 'Age': 18}
row8 = {'Player':'Michael Jordan', 'Points':0, 'Age': 19}
row9 = {'Player':'Michael Jordan', 'Points':0, 'Age': 20}
row10 = {'Player':'Michael Jordan', 'Points':21541, 'Age': 30}
row11 = {'Player':'Michael Jordan', 'Points':29277, 'Age': 35}
row12 = {'Player':'Michael Jordan', 'Points':29277, 'Age': 36}
row13 = {'Player':'Michael Jordan', 'Points':29277, 'Age': 37}
row14 = {'Player':'Kobe Bryant', 'Points':33643, 'Age': 38}
row15 = {'Player':'Kobe Bryant', 'Points':33643, 'Age': 39}

stats = stats.append(row1, ignore_index=True)
stats = stats.append(row2, ignore_index=True)
stats = stats.append(row3, ignore_index=True)
stats = stats.append(row4, ignore_index=True)
stats = stats.append(row5, ignore_index=True)
stats = stats.append(row6, ignore_index=True)
stats = stats.append(row7, ignore_index=True)
stats = stats.append(row8, ignore_index=True)
stats = stats.append(row9, ignore_index=True)
stats = stats.append(row10, ignore_index=True)
stats = stats.append(row11, ignore_index=True)
stats = stats.append(row12, ignore_index=True)
stats = stats.append(row13, ignore_index=True)
stats = stats.append(row14, ignore_index=True)
stats = stats.append(row15, ignore_index=True)

stats.Age = stats.Age.apply(lambda x: int(x))
stats.Points = stats.Points.apply(lambda x: int(x))
```

Now, let's take our full stats dataframe and group by age and player. We'll then unstack it go get it into the correct format for this package.

In [20]:

```
stats_gb = stats.groupby(['Age', 'Player']).sum()
stats_gb.head()
```

Out[20]:

		Points
Age	**Player**	
18	**Kobe Bryant**	539
	Lebron James	0
	Michael Jordan	0
19	**Kobe Bryant**	1759
	Lebron James	1654

In [21]:

```
stats_us = stats_gb.unstack()
stats_us.head()
```

Out[21]:

	Points		
Player	**Kobe Bryant**	**Lebron James**	**Michael Jordan**
Age			
18	539	0	0
19	1759	1654	0
20	2755	3829	0
21	4240	6307	2313
22	6178	8439	2721

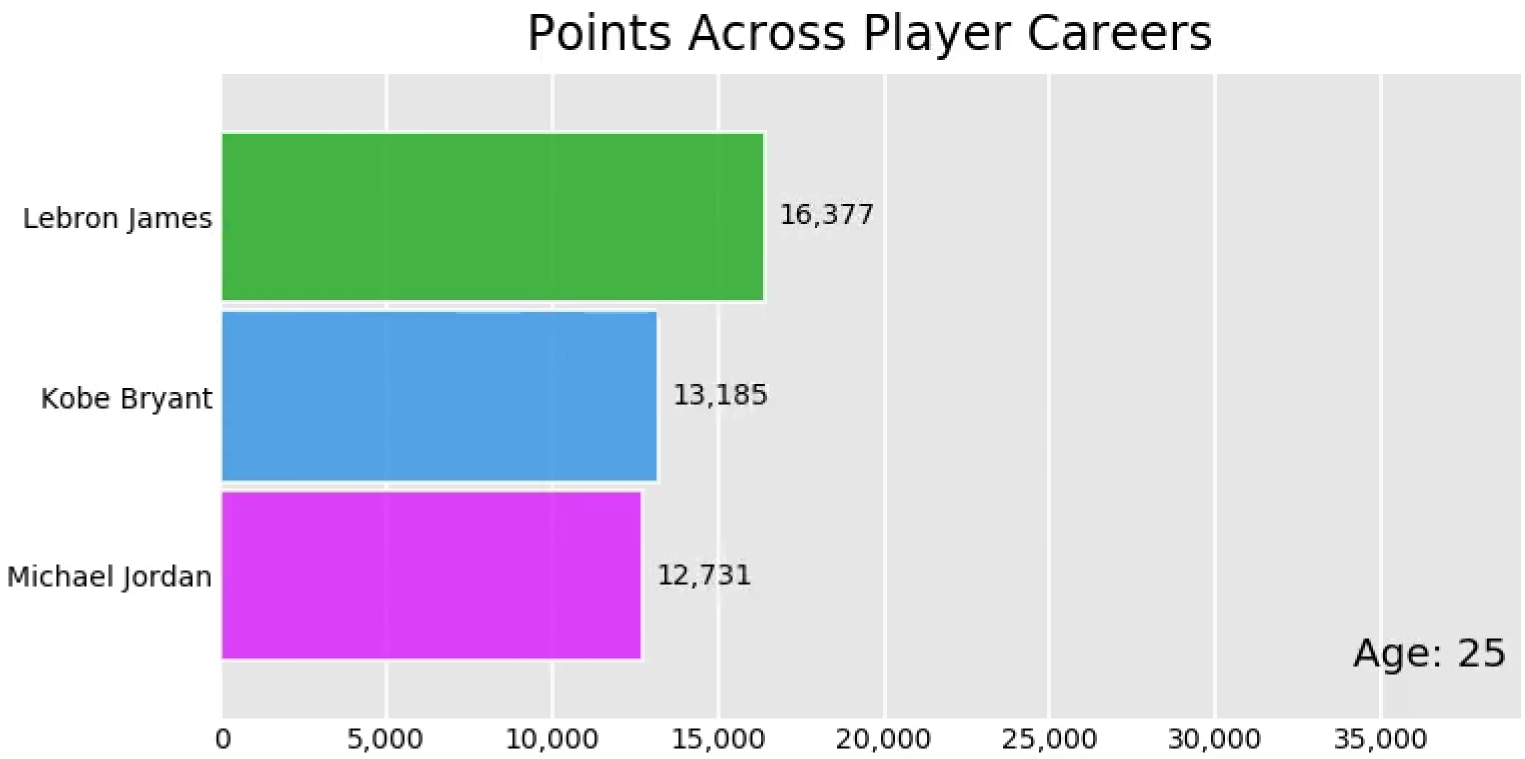
6.2 Points Across Player Careers

I had to install a few different dependencies to get this to work - if the chart doesn't load, you can view it here (https://drive.google.com/file/d/1_GYLalqpbXuYn9ANC8SpHWAPeLBe5ZQU/view?usp=sharing).

In [22]:

```
bcr.bar_chart_race(
    df=stats_us.Points,
    orientation='h',
    sort='desc',
    n_bars=3,
    fixed_max=True,
    steps_per_period=15,
    interpolate_period=False,
    label_bars=True,
    bar_size=.95,
    period_length=500,
    figsize=(5, 3),
    dpi=144,
    cmap='dark12',
    title='Points Across Player Careers',
    bar_label_size=7,
    tick_label_size=7,
    shared_fontdict={'family' : 'DejaVu Sans', 'color' : '0'},
    scale='linear',
    period_label={'x': .99, 'y': .1, 'ha': 'right', 'va': 'center'},
    period_fmt='Age: {x:,.0f}')
```

Out[22]:


Points Across Player Careers
Lebron James
Kobe Bryant
Michael Jordan
16,377
13,185
12,731
Age: 25
0
5,000
10,000
15,000
20,000
25,000
30,000
35,000