7 minutes of data science with Python, Pandas, Jupyter, and an Alien Outlier

- Jason Vandecreek

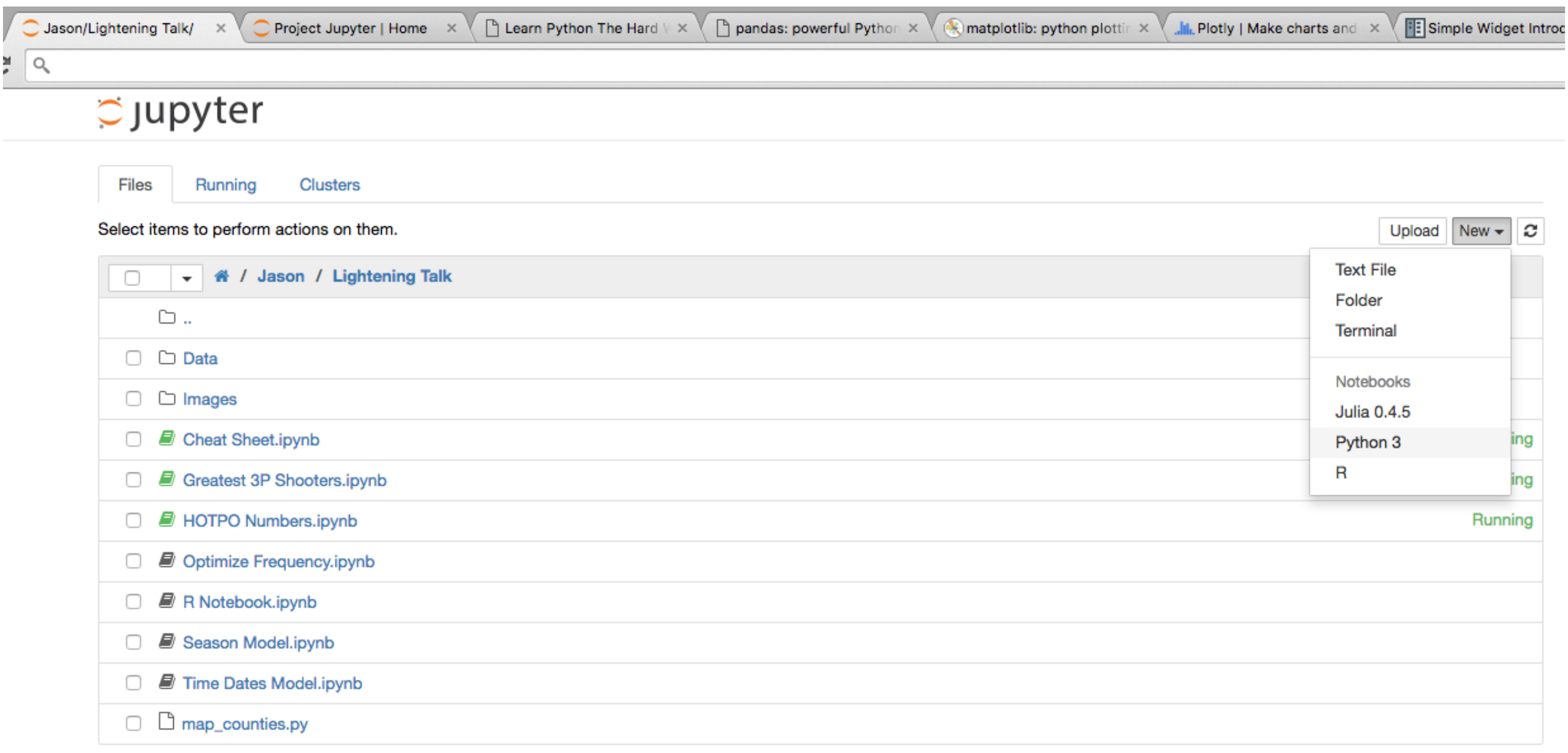
- Mark Earnest

What?

- Data Science (cross discipline)
- Python (friendly language)
- Pandas (A DataFrame library)
- Jupyter (Interactive notebook)
- and an Alien Outlier (?)

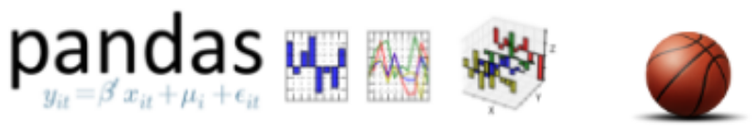
The Jupyter Notebook is a web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text.

http://jupyter.org



jupyter Greatest 3P Shooters Last Checkpoint: 9 hours ago (autosaved)

IPython Notebook: Analyze and Chart Greatest 3 Point Shooters



In [1]:
```
import pandas as pd
```

In [2]:
```
# Read in the player data from a file
players = pd.read_csv('Data/players.csv')
```

★ Note: read_csv creates a DataFrame data structure and stores that into the variable named "players".

In [3]:
```
# Show number of (Rows, Columns)
players.shape
```

Out[3]: (217, 14)

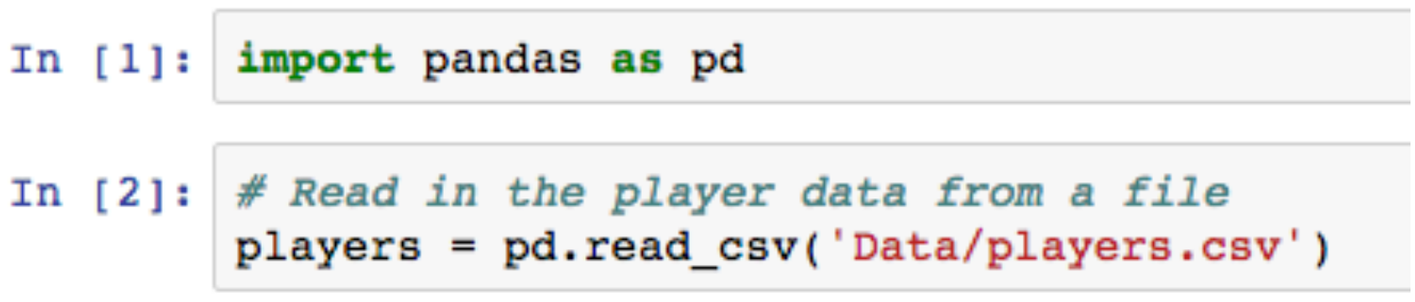
In [4]:
```
# Display first five rows
players.head()
```

Out[4]:

	Name	Age	Games	Minutes	Points	Rebounds	Assists	FG%	3P%	FT%	TS%	3P	3PAr	Efficiency Rating
0	Wilt Chamberlain*	24	79	3773.0	3033.0	2149.0	148.0	0.509	NaN	0.504	0.519	NaN	NaN	27.8
1	LeBron James	24	81	3054.0	2304.0	613.0	587.0	0.489	0.344	0.780	0.591	132	0.238	31.7
2	Rick Barry*	NaN	78	NaN	2775.0	714.0	282.0	0.451	NaN	0.884	0.531	NaN	NaN	24.2
3	Kevin Durant	23	66	2546.0	1850.0	527.0	231.0	0.496	0.387	0.860	0.610	133	0.265	26.2
4	Kareem Abdul-Jabbar*	23	NaN	3288.0	NaN	1311.0	NaN	0.577	NaN	0.690	0.606	NaN	NaN	29.0

Notebook Cells can have code or markdown/html

Code cells are numbered



Tab completion

```
In [ ]: players = pd.read_
```

```
pd.read_clipboard
pd.read_csv
pd.read_excel
pd.read_fwf
pd.read_gbq
pd.read_hdf
pd.read_html
pd.read_json
pd.read_msgpack
pd.read_pickle
```

```
In [ ]: pd.read_csv('Data/')
```

```
Data/kscounties.geojson
Data/players.csv
Data/players_30pt_name.xlsx
Data/time_data.csv
```

Data Frame

In [3]:
```
# Show number of (Rows, Columns)
players.shape
```

Out[3]: (217, 14)

In [4]:
```
# Display first five rows
players.head()
```

Out[4]:

	Name	Age	Games	Minutes	Points	Rebounds	Assists	FG%	3P%	FT%	TS%	3P	3PAr	Efficiency Rating
0	Wilt Chamberlain*	24	79	3773.0	3033.0	2149.0	148.0	0.509	NaN	0.504	0.519	NaN	NaN	27.8
1	LeBron James	24	81	3054.0	2304.0	613.0	587.0	0.489	0.344	0.780	0.591	132	0.238	31.7
2	Rick Barry*	NaN	78	NaN	2775.0	714.0	282.0	0.451	NaN	0.884	0.531	NaN	NaN	24.2
3	Kevin Durant	23	66	2546.0	1850.0	527.0	231.0	0.496	0.387	0.860	0.610	133	0.265	26.2
4	Kareem Abdul-Jabbar*	23	NaN	3288.0	NaN	1311.0	NaN	0.577	NaN	0.690	0.606	NaN	NaN	29.0

Slice the data frame

In [5]:
```
# Start : Stop : Step
players[5:100:10]
```

Out[5]:

	Name	Age	Games	Minutes	Points	Rebounds	Assists	FG%	3P%	FT%	TS%	3P	3PAr	Efficiency Rating
5	Tiny Archibald*	24	80	3681.0	2719.0	223.0	910.0	0.488	NaN	0.847	0.555	NaN	NaN	25.2
15	Wilt Chamberlain*	27	80	3689.0	2948.0	1787.0	403.0	0.524	NaN	0.531	0.537	NaN	NaN	31.6
25	LeBron James	29	77	2902.0	2089.0	533.0	488.0	0.567	0.379	0.750	0.649	116	0.226	29.3
35	Russell Westbrook	27	80	2749.0	1878.0	626.0	834.0	0.454	0.296	0.812	0.554	101	0.236	27.6
45	Allen Iverson*	30	72	3103.0	2377.0	232.0	532.0				NaN	NaN	0.122	25.9
55	Kobe Bryant		68	2783.0	1938.0	399.0	338.0	0.464	0.305	0.853	0.552	61	0.132	24.5
65	Dominique Wilkins*	31	81	NaN	NaN	NaN	NaN	0.470	0.341	0.829	0.555	85	0.152	23.5
75	Dirk Nowitzki	30	81	3050.0	2094.0	681.0	197.0	0.479	0.359	0.890	0.564	61	0.105	23.1
85	Kevin Martin	27	eighty	2603.0	1876.0	258.0	198.0	0.436	0.383	0.888	0.601	176	0.362	21.4
95	Carmelo Anthony	26	77	2751.0	1970.0	563.0	221.0	0.455	0.378	0.838	0.557	95	0.167	21.7

Data Cleaning: Remove unwanted characters

```
In [6]: # Remove * from the player names
        players['Name'] = players['Name'].str.replace('*', '')
        players['Name'].head() # first five Names
```

```
Out[6]: 0       Wilt Chamberlain
        1           LeBron James
        2             Rick Barry
        3           Kevin Durant
        4    Kareem Abdul-Jabbar
        Name: Name, dtype: object
```

Columns: show, drop, rename

In [7]:
```
players.columns
```

Out[7]:
```
Index(['Name', 'Age', 'Games', 'Minutes', 'Points', 'Rebounds', 'Assists',
       'FG%', '3P%', 'FT%', 'TS%', '3P', '3PAr', 'Efficiency Rating'],
      dtype='object')
```

In [8]:
```
# Drop columns which aren't going to be used
columns_to_drop = ['FT%', 'Rebounds', 'Assists']
players.drop(columns_to_drop, axis=1, inplace=True)
```

In [9]:
```
# Rename columns to make it clear what they are
columns_to_rename = {'3P': 'Three Pointers Made', '3PAr': 'Three Point Attempt Rate'}
players.rename(columns=columns_to_rename, inplace=True)
```

Check data types and convert

```
In [11]: players.dtypes
```

```
Out[11]: Name                        object
         Age                         object
         Games                       object
         Minutes                    float64
         Points                     float64
         FG%                         object
         3P%                         object
         TS%                        float64
         Three Pointers Made         object
         Three Point Attempt Rate   float64
         Efficiency Rating          float64
         dtype: object
```

```
In [12]: players['Three Pointers Made'] = pd.to_numeric(players['Three Pointers Made'], errors='force')
```

```
In [13]: players['Three Pointers Made'].dtypes
```

```
Out[13]: dtype('float64')
```

Filter values
(players with 100+ 3s in a season)

In [14]:
```
# Filter for players who have made 100 or more three pointers in a season
three_pointers = players['Three Pointers Made']
players = players[three_pointers >= 100]
players.shape # show the new shape (rows, columns)
```

Out[14]:
```
(156, 11)
```

In [15]:
```
# Convert Three Pointers from float to int
players['Three Pointers Made'] = players['Three Pointers Made'].astype(int)
```

In [16]:
```
# Sort and show first five ascending rows
players.sort_values('Three Pointers Made').head()
```

Out[16]:

	Name	Age	Games	Minutes	Points	FG%	3P%	TS%	Three Pointers Made
35	Russell Westbrook	27	80	2749.0	1878.0	0.454	0.296	0.554	101
88	Gary Payton	32	79	3244.0	1823.0	0.456	0.375	0.522	102
51	Tracy McGrady	22	76	2912.0	1948.0	0.451	0.364	0.532	103
18	LeBron James	28	76	2877.0	2036.0	0.565	0.406	0.640	103
74	Allen Iverson	29	75	3174.0	2302.0	0.424	0.308	0.532	104

Calculate avg points per game
(oops, we found an unexpected value)

```
In [17]: players['Points'] /= players['Games']

         TypeError: unsupported operand type(s) for /: 'float' and 'str'

In [18]: players['Games'].value_counts().sort_values()

Out[18]: 69          1
         72          1
         eighty      1
         70          2
         66          2
         71          2
         74          3
         67          4
         73          4
         76          6
         75          8
```

Handle string value in a numeric column

```
In [19]: players[players.Games == 'eighty']
```

Out[19]:

	Name	Age	Games	Minutes	Points	FG%	3P%	TS%	Three Pointers Made	Three Point Attempt Rate	Efficiency Rating
85	Kevin Martin	27	eighty	2603.0	1876.0	0.436	0.383	0.601	176	0.362	21.4

```
In [20]: players.loc[85, 'Games'] = 80

In [21]: players.loc[85]
```

```
Out[21]: Name                          Kevin Martin
         Age                                 27
         Games                               80
         Minutes                           2603
         Points                            1876
         FG%                              0.436
         3P%                              0.383
         TS%                              0.601
         Three Pointers Made                176
         Three Point Attempt Rate         0.362
         Efficiency Rating                 21.4
         Name: 85, dtype: object
```

```
In [22]: players.Games = pd.to_numeric(players.Games)

In [23]: players.Games.dtypes
```

```
Out[23]: dtype('int64')
```

Recalculate avg points per game

```
In [24]: players.Points /= players.Games
         players.Points.head()
```

```
Out[24]: 1      28.444444
         3      28.030303
         9      26.555556
         18     26.789474
         20     29.710526
         Name: Points, dtype: float64
```

```
In [25]: players.Points = players.Points.round(2)
         players.Points.tail()
```

```
Out[25]: 212     15.29
         213     16.50
         214     10.91
         215     14.89
         216     11.97
         Name: Points, dtype: float64
```

Deal with duplicates

In [26]:
```
# Find duplicated rows
dupes = players[players.duplicated()]
dupes
```

Out[26]:

	Name	Age	Games	Minutes	Points	
99	Kevin Martin	27	80	2603.0	23.45	
128	James Harden	23	78	2985.0	25.94	
197	James Harden	25	81	2981.0	27.37	

In [27]:
```
# Remove duplicates
players.drop(dupes.index, inplace=True)
```

In [28]:
```
# Check to be sure
players[players.duplicated()]
```

Out[28]:

	Name	Age	Games	Minutes	Points	FG%	3P%

Show descriptives (stats)

```
In [29]: three_pointers = players['Three Pointers Made']
         three_pointers.describe().round(2)
```

```
Out[29]: count    153.00
         mean     182.24
         std       44.85
         min      101.00
         25%      150.00
         50%      185.00
         75%      205.00
         max      402.00
         Name: Three Pointers Made, dtype: float64
```

Calculate mean and find outliers

In [30]:
```
mean = three_pointers.mean()
mean
```

Out[30]: 182.24183006535947

In [31]:
```
# outliers are 2 standard deviations or more above mean
outlier_boundary = x = mean + (2 * three_pointers.std())
outlier_boundary
```

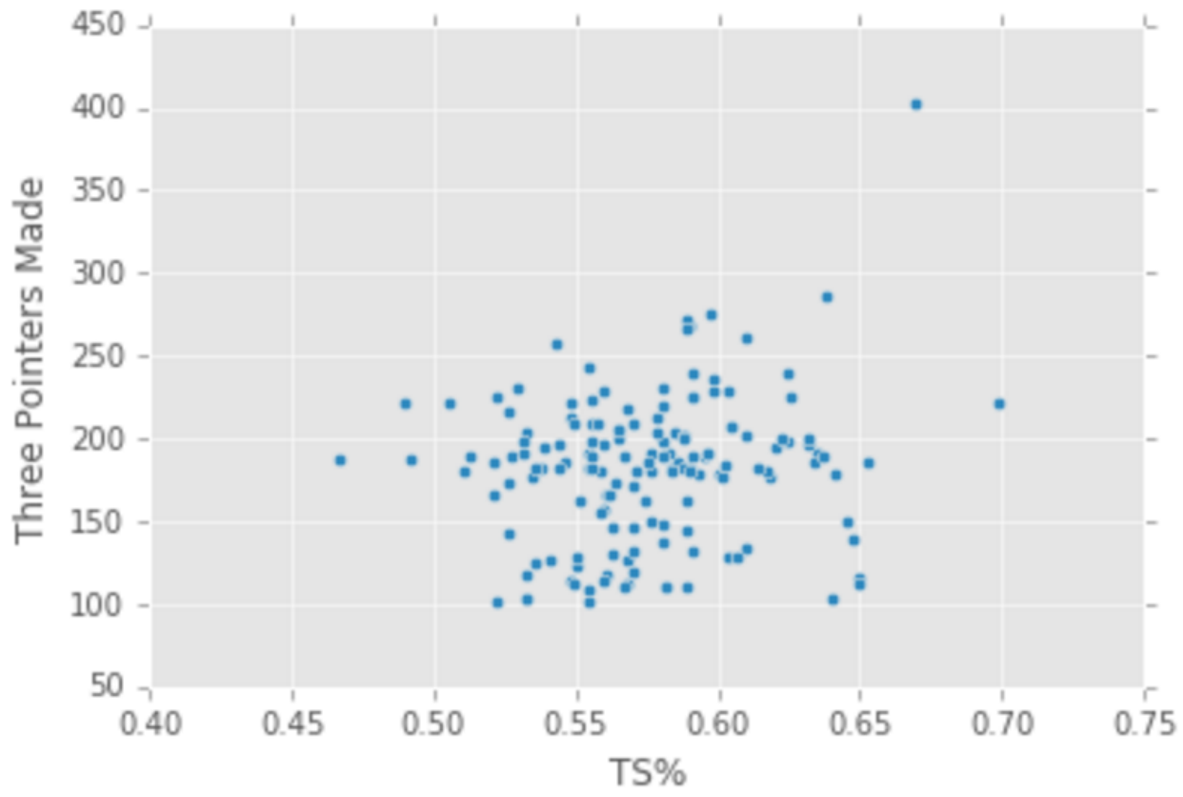
Out[31]: 271.94309630760756

Start plotting

In [32]:
```
# Enable plotting inside notebook
%matplotlib inline

# Libraries which will be used
import matplotlib
import matplotlib.pyplot as plt
from scipy.misc import imread

# R Style plot
matplotlib.style.use('ggplot')
```

In [33]:
```
players.plot.scatter(x='TS%', y='Three Pointers Made')
```

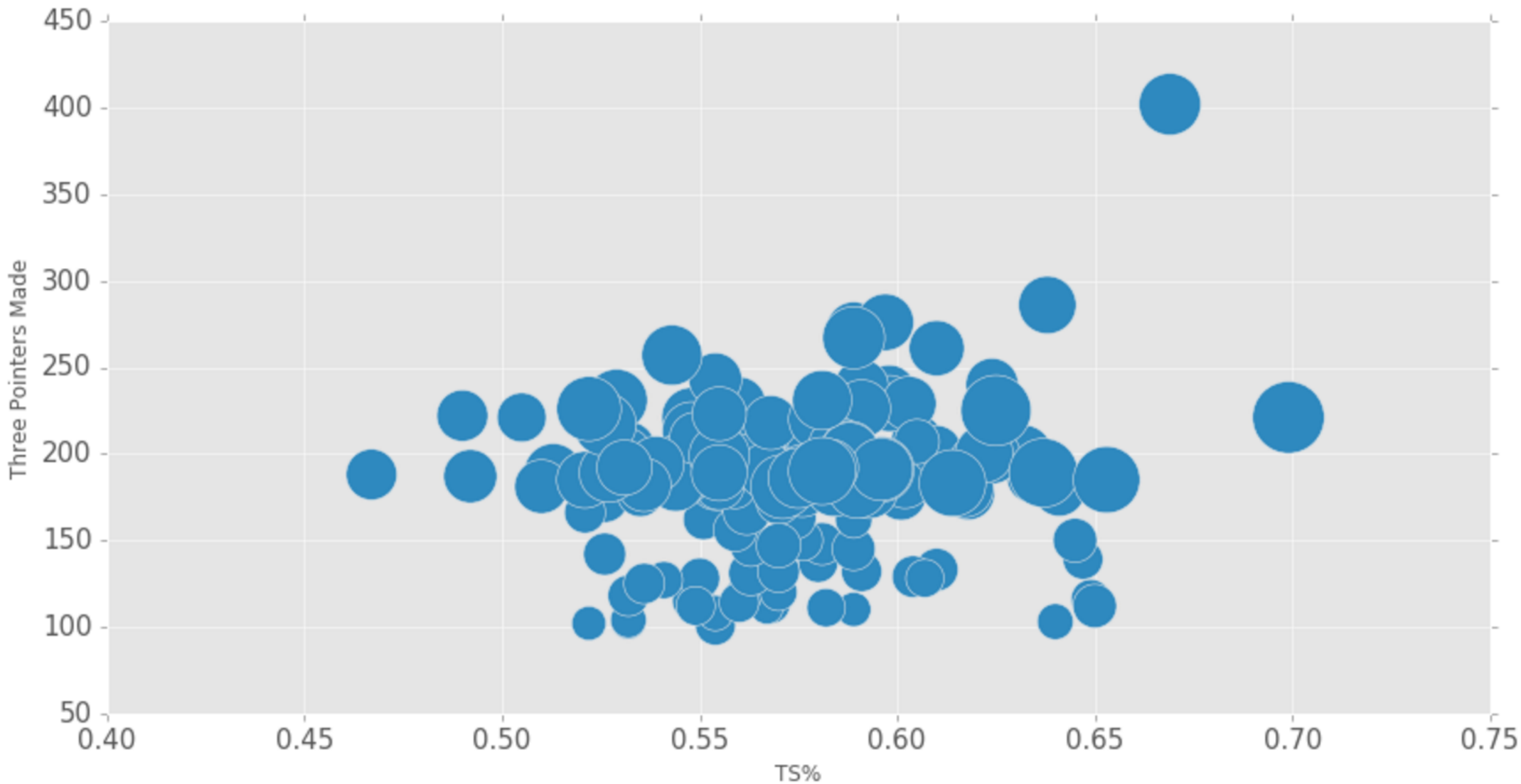
Out[33]: `<matplotlib.axes._subplots.AxesSubplot at 0x7f26b032e1d0>`



Re-plot with extra params

In [34]:
```
# replot with bigger chart and computer circle sizes
circle_size = players['Three Point Attempt Rate'] * 2200
players.plot.scatter(x='TS%', y='Three Pointers Made', s=circle_size,  figsize=(14, 7), fontsize=15)
```

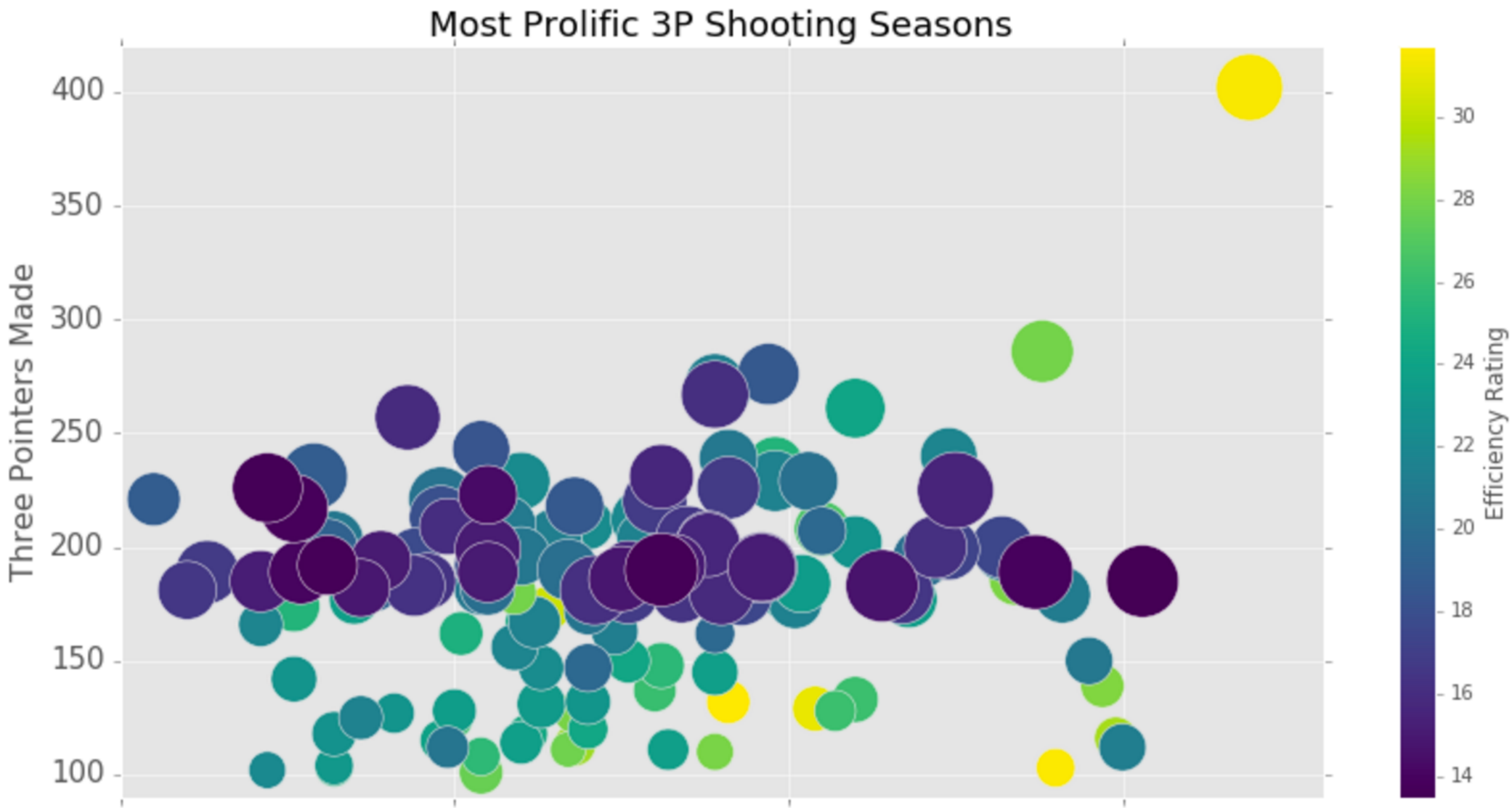
Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe41ab5a0b8>



Add colors & set labels

In [36]:
```
ax = players.plot.scatter(x='TS%', y='Three Pointers Made', s=circle_size, c='Efficiency Rating',
                          cmap='viridis', figsize=(14, 7), fontsize=15)

ax.set_ylim(90, 420)
ax.set_xlim(0.5, 0.68)
ax.set_title("Most Prolific 3P Shooting Seasons", fontsize=18)
ax.set_ylabel("Three Pointers Made", fontsize=16)
```

Out[36]: <matplotlib.text.Text at 0x7fe412942278>

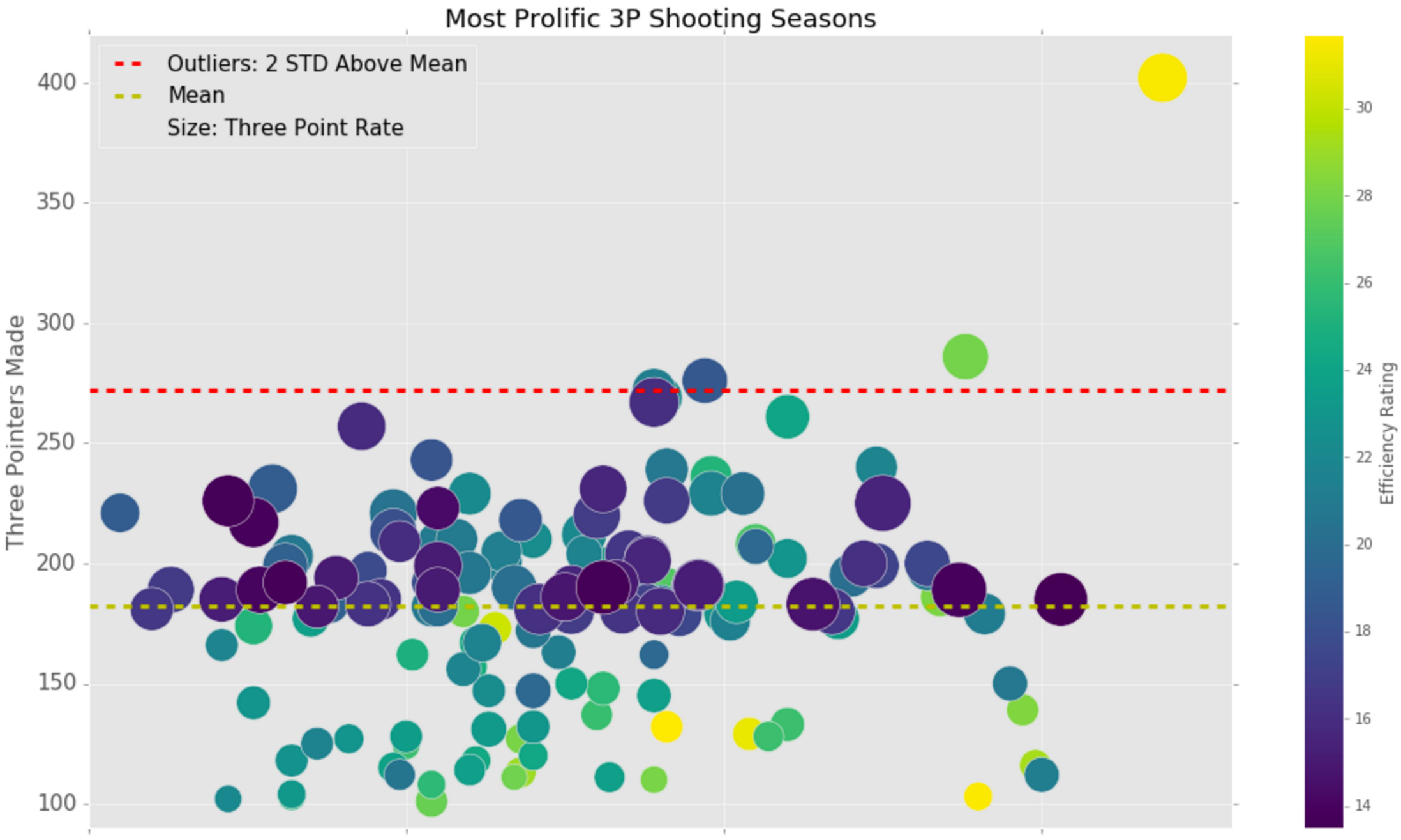


Define a custom plotting function

In [39]:
```
# define function to make full plot
def scatter_plot():
    circle_size = players['Three Point Attempt Rate'] * 2200
    ax = players.plot.scatter(x='TS%', y='Three Pointers Made', c='Efficiency Rating',
                              s=circle_size, cmap='viridis', figsize=(18,10), fontsize=15)

    ax.set_ylim(90, 420)
    ax.set_xlim(0.5, 0.68)
    ax.set_title("Most Prolific 3P Shooting Seasons", fontsize=18)
    ax.set_ylabel("Three Pointers Made", fontsize=16)

    ax.text(0.58, 70, "True Shooting %", fontsize=16)
    #ax.text(0.51, outlier_boundary + 2, "Outliers: 2 STD Above Mean", fontsize=14)

    ax.axhline(outlier_boundary, color='r', linestyle='--', lw=3)
    ax.axhline(mean, color='y', linestyle='--', lw=3)

    outlier_line, = plt.plot([1,2,3,4], ls='--', color='r', lw=3, label='Outliers: 2 STD Above Mean')
    mean_line, = plt.plot([1,2,3,4], ls='--', color='y', lw=3, label='Mean')
    circle_size, =  plt.plot([3,2,1], ls='None', lw=3, label='Size: Three Point Rate')

    ax.legend(handles=[outlier_line, mean_line, circle_size], loc=2, fontsize=15)

    return ax
```

In [40]:
```
scatter_plot()
```

Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe412b89978>



Find the outlier player rows, so we can annotate them

In [41]:
```
outliers = players[three_pointers > outlier_boundary]
outliers = outliers.sort_values('Three Pointers Made', ascending=False)
outliers
```

Out[41]:

	Name	Age	Games	Minutes	Points	FG%	3P%	TS%	Three Pointers Made	Three Point Attempt Rate	Efficiency Rating
116	Stephen Curry	27	79	2700.0	30.06	0.504	0.454	0.669	402	0.554	31.5
120	Stephen Curry	26	80	2613.0	23.75	0.487	0.443	0.638	286	0.482	28.0
164	Klay Thompson	25	80	2666.0	22.14	0.470	0.425	0.597	276	0.469	18.6
140	Stephen Curry	24	78	2983.0	22.90	0.451	0.453	0.589	272	0.432	21.3

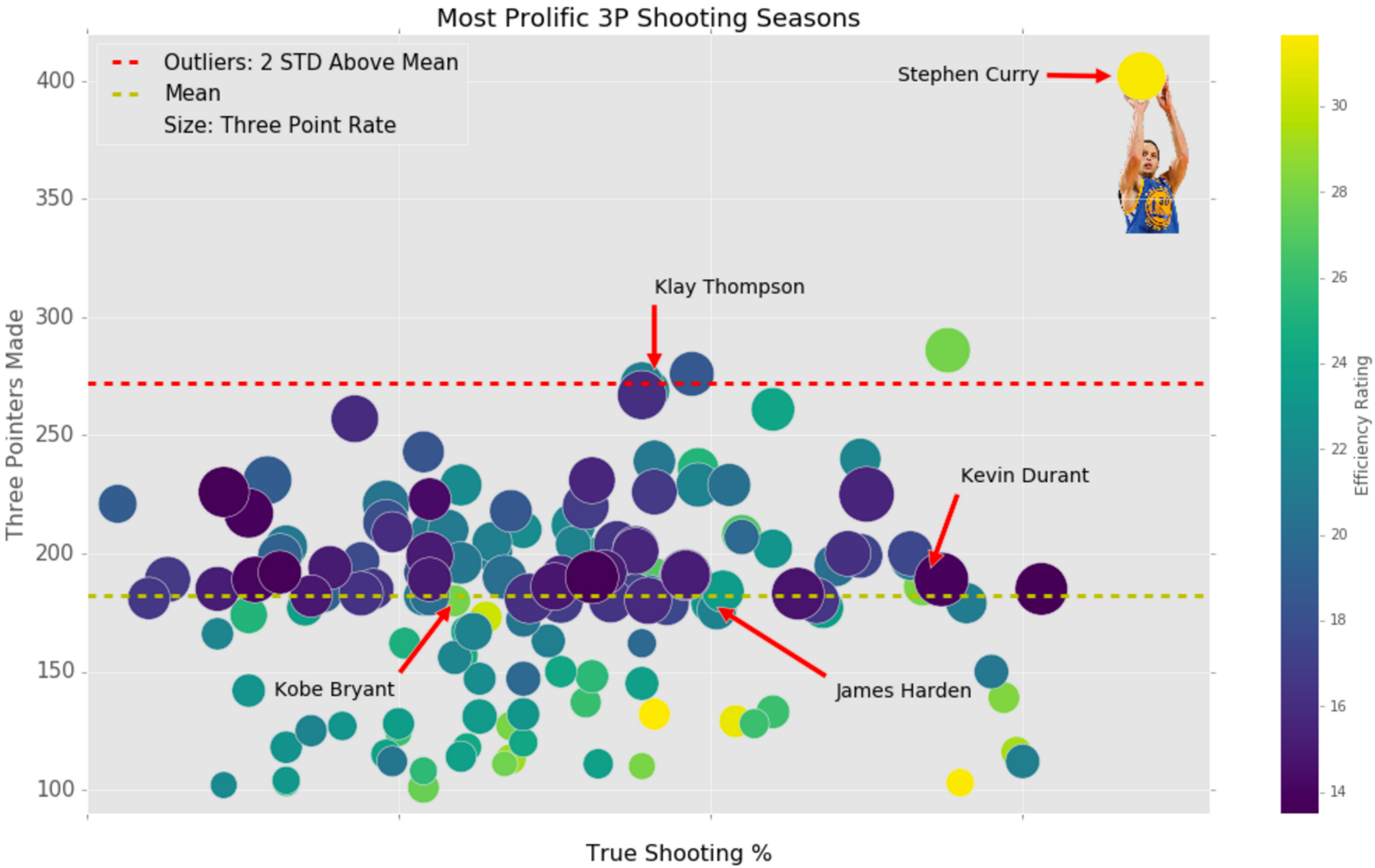
In [42]:
```
player_names = ['Kevin Durant', 'Kobe Bryant', 'Klay Thompson', 'James Harden']
players[players.Name.isin(player_names)].sort_values('Three Pointers Made', ascending=False)[:10]
```

Out[42]:

	Name	Age	Games	Minutes	Points	FG%	3P%	TS%	Three Pointers Made	Three Point Attempt Rate	Efficiency Rating
164	Klay Thompson	25	80	2666.0	22.14	0.470	0.425	0.597	276	0.469	18.6
147	Klay Thompson	24	77	2455.0	21.66	0.463	0.439	0.591	239	0.420	20.8
123	James Harden	26	82	3125.0	28.98	0.439	0.359	0.598	236	0.406	25.3
209	Klay Thompson	23	81	2868.0	18.37	0.444	0.417	0.555	223	0.425	14.3
122	James Harden	25	81	2981.0	27.37	0.440	0.375	0.605	208	0.378	26.7
117	Kevin Durant	25	81	3122.0	32.01	0.503	0.391	0.635	192	0.291	29.8
118	Kevin Durant	27	72	2578.0	28.18	0.505	0.388	0.634	186	0.348	28.2
119	Kobe Bryant	27	80	3277.0	35.40	0.450	0.347	0.559	180	0.238	28.0
77	James Harden	23	78	2985.0	25.94	0.438	0.368	0.600	179	0.364	23.0
70	James Harden	24	73	2777.0	25.36	0.456	0.366	0.618	177	0.401	23.5

Annotate our plot

In [43]:
```
ax = scatter_plot()
fc = 'red'; fs = 14  # font color and size

ax.annotate('Stephen Curry', xy=(.665, 402), xytext=(.63, 400),
            arrowprops=dict(facecolor=fc, shrink=0.05), fontsize=fs)

ax.annotate('Klay Thompson', xy=(.591, 276), xytext=(.591, 310),
            arrowprops=dict(facecolor=fc, shrink=0.05), fontsize=fs)

ax.annotate('Kevin Durant', xy=(.635, 192), xytext=(.64, 230),
            arrowprops=dict(facecolor=fc, shrink=0.05), fontsize=fs)

ax.annotate('Kobe Bryant', xy=(.559, 180), xytext=(.530, 140),
            arrowprops=dict(facecolor=fc, shrink=0.05), fontsize=fs)

ax.annotate('James Harden', xy=(.6, 179), xytext=(.62, 139),
            arrowprops=dict(facecolor=fc, shrink=0.05), fontsize=fs)

img = imread("Images/curry-shooting.png")

# Set the part of the graph to show the image on
x0, x1 = (.6615, .6795) # Based on player name index 0 to 9
y0, y1 = (335, 409) # Based on y range 0 to 45

# Add the image
ax.imshow(img, extent=[x0, x1, y0, y1], aspect='auto')
```


Most Prolific 3P Shooting Seasons
Outliers: 2 STD Above Mean
Mean
Size: Three Point Rate
Stephen Curry
Klay Thompson
Kevin Durant
Kobe Bryant
James Harden
Three Pointers Made
True Shooting %
Efficiency Rating
400
350
300
250
200
150
100
30
28
26
24
22
20
18
16
14

Import interactive Plot.ly charting library (local and off-line, open source)

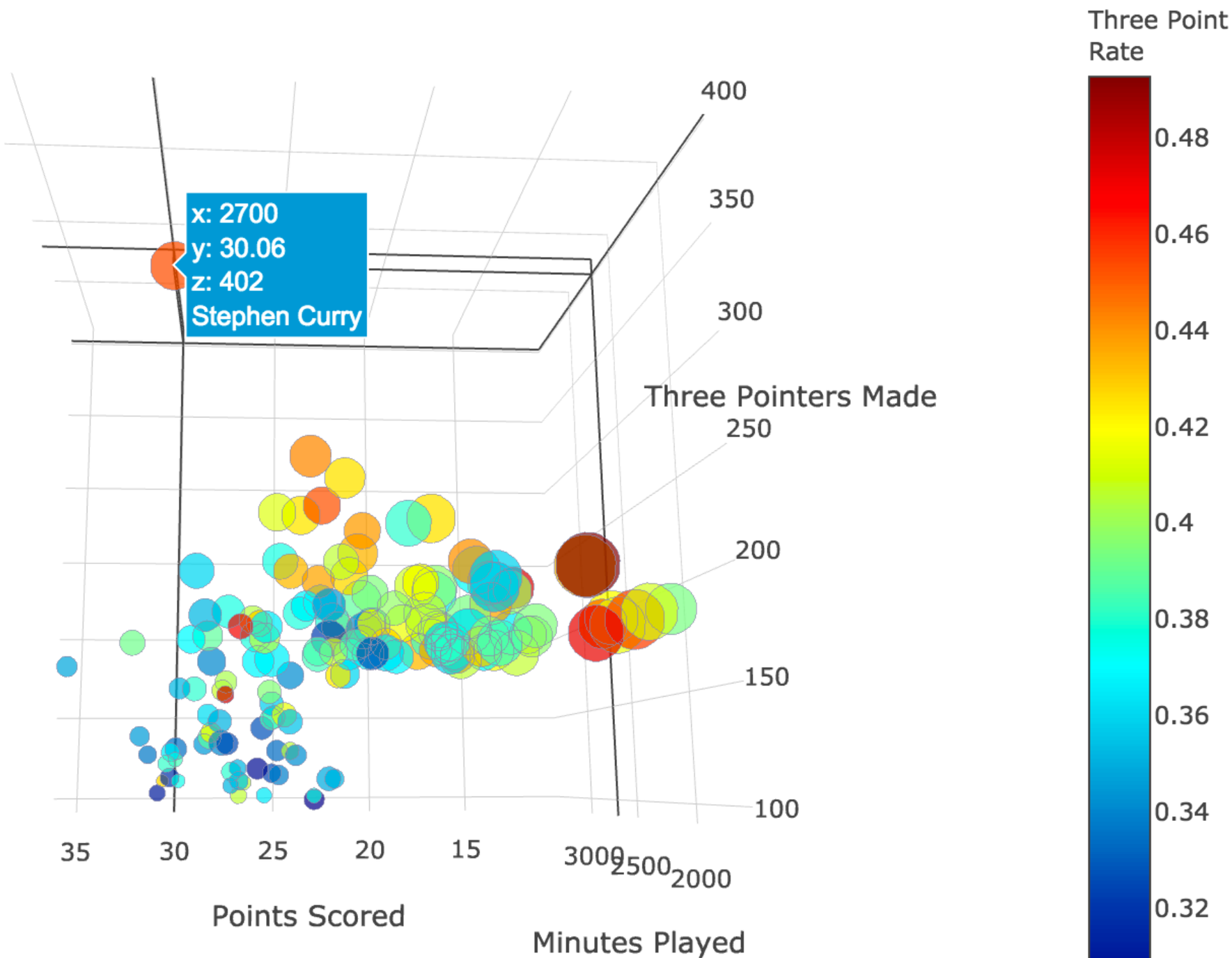
In [44]:
```
import cufflinks as cf
from plotly.offline import download_plotlyjs, init_notebook_mode, iplot
import plotly.graph_objs as go

cf.go_offline()
init_notebook_mode()
```

Create interactive scatter chart

In [50]:

```
# Line numbers turned on with ESC l

# Data
trace = go.Scatter3d(
    x = players['Minutes'],
    y = players['Points'],
    z = players['Three Pointers Made'],
    text = players['Name'], # + ' ' + players['Season'] + ' ' + players['Team'],
    mode = 'markers',
    marker = dict(
        sizemode = 'diameter',
        size = players['Three Point Attempt Rate'] * 50,
        color = players['3P%'],
        colorscale = 'Jet',
        colorbar = dict(title = 'Three Point<br>Rate'),
        line = dict(color='rgb(140, 140, 170)')
    )
)

data = [trace]

# Style Layout
layout = go.Layout(width=900, height=700, title = 'NBA Greatest 3PT Shooters on History',
      scene = dict(xaxis=dict(title='Minutes Played'),
            yaxis=dict(title='Points Scored'),
            zaxis=dict(title='Three Pointers Made'),
           )
     )


# Plot
figure = {'data': data, 'layout': layout}
iplot(figure, show_link=False)
```

NBA Greatest 3PT Shooters on History



Interactive widget library

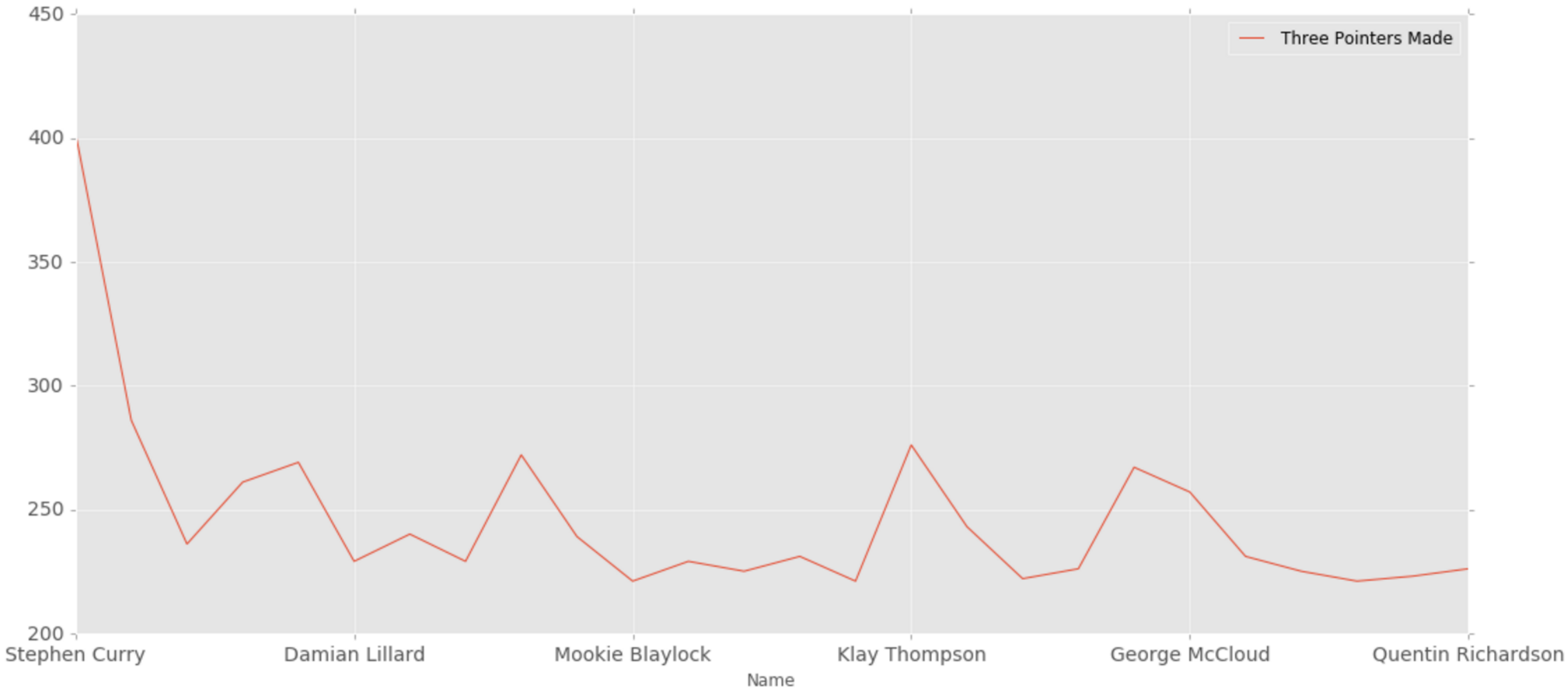
In [51]:
```
from ipywidgets import widgets
from IPython.display import display
from IPython.display import clear_output
```

```
# <snip>

slider = widgets.IntSlider(value = max_value, min = min_value, max = max_value, step = 10,
                           description='Three Pointers: ')

btn = widgets.Button(description="Update")

toggle = widgets.ToggleButtons(description='Filter By:', options=["Less Than", "Greater Than"])

# </snip>
```

```
# <snip>

display(container)
players.plot(x='Name', y='Three Pointers Made', figsize=figure_size, fontsize=15)
```

Three Pointers: 221 Update Filter By: Less Than Greater Than

```
Greater Than 221

<matplotlib.axes._subplots.AxesSubplot at 0x7fe408b76f60>
```



Go play

https://github.com/Mearnest/jupyter_python_pandas