Final_Draft

December 14, 2017

1 Predicting NBA Efficiency

1.0.1 Spencer Giddens and Gabriella Smith


```
In [1]: import pandas as pd
        import numpy as np
        from matplotlib import pyplot as plt
        import json
        %run scraping.py
        %matplotlib inline
        plt.rcParams['figure.figsize'] = [15,8]
```

2 Introduction

2.0.1 Purpose

Our goal is to use statistics to predict an NBA player's quality over the course of his career using only data from his rookie season. In other words, the question we want to answer is: "What will an athlete's career performance look like based on their rookie season?" This predictive ability would be very valuable to NBA teams and other sports analysts to effectively invest in players for the future of their franchises.

We hope to find out that it is possible to predict how effective a player will be over his career based on his rookie season. If this is the case, NBA scouts and coaches can use this information to be more effective in their decisions.

2.0.2 Current Research

ESPN has a forecast panel which makes many sports-related predictions. Their predictions consist of team performance. For example, what team will win a certain game or the overall championship. They also predict which players will win MVP, ROY or achieve All-Star status. FiveThirtyEight, a blog site for data and statistics-loving individuals, also has a system that predicts a player's overall performance. The techniques that these systems use are not specified. However, we did not find any specific research that use a player's rookie performance to predict their career performance. We did find one article that discussed the skill vs luck tradeoff in the sports world. They discussed that the likelihood of a team winning a basketball game is more dependent on skill than luck (Arbesman). This is due to the size of the team and quick pace of the game. This skill surety allowed us to be more confident that our data analysis techniques will be able to provide us with an answer to our question.

2.0.3 Overview of Data

This dataset is a collection of statistics from 205 NBA players for each season of their careers. For each player and for each season, the dataset contains information on statistics such as minutes per game, points per game, assists per game, field goal percentage, etc. This data was collected from the statistics page of the official NBA website. In addition to this collected data, we added our own "feature columns" representing the overall efficiency rating of each player for each of their seasons. We calculated these columns in four different ways, which will be explained later.

The data was collected from stats.nba.com. We consider this source to be highly reliable and free of virtually all bias. The site simply lists statistics that were recorded in each season, so there is no opportunity for bias to be introduced. In addition, the data is already very consistant and none of it is missing, similar to most sports data. This data is obviously relevant to the project objective because the best way to quantitatively measure a player's quality for a season or for a career is to use his statistics.

3 Data Collection

3.0.1 Data Gathering

Our data was acquired from stats.nba.com using Selenium and web scraping tecniques. To best answer the question from our proposal (how good will an NBA player be over the course of his career based only on his rookie season?), we decided the we needed data that fit two criteria. First, we decided we would start by selecting players from the 1990-91 season. Since the 3 point line was introduced in 1980 we didn't want to have any missing or significanlty low 3 point statistics becuase the 3 point shot is an integral part of the game now. Finally, we decided to choose a random sample of NBA players in order to avoid biasing the dataset as much as possible. We wrote a function that, given a list of seasons and the number of players to select from each season, would randomly select that many players from each season, return a list of players scraped, and save the scraped data into a csv file. We read the csv files into a pandas DataFrame with the index set to be the season year. We saved each pandas DataFrame into a dictionary where the key for each player is a string with the player's first and last names and position. The functions we used are included in the auxilary file scraping.py.

```
In [ ]: # Create a list of the seasons to scrape data from and call the scrape function
        seasons = ['199{}-9{}'.format(i,i+1) for i in range(9)] + ['1999-00']
        seasons += ['200{}-0{}'.format(i,i+1) for i in range(9)] + ['2009-10']
        seasons += ['201{}-1{}'.format(i,i+1) for i in range(7)]
        players = scrape_data(seasons, 20)
```

The following is how we loaded the data into a dictionary of Pandas DataFrames and an example of what the data looks like:

```
In [2]: def make_dict():
            """Construct and return the dictionary of statistics for each player."""
            d = dict()
            for player in players:
                d[player] = pd.read_csv("csvs/" + player+'.csv', header=0, index_col=0)
            return d
```

```
        players = json.load(open('player.txt')) # The names of the players scraped was
                                                # stored in a .txt file for access later
        player_data = make_dict()
```

```
In [3]: print(player_data['Kobe Bryant, F-G'].head())
```

```
        TEAM  AGE  GP  GS   MIN   PTS  FGM   FGA   FG%  3PM ...   FTA   FT%  \
SEASON                                                     ...
2015-16  LAL   37  66  66  28.2  17.6  6.0  16.9  35.8  2.0 ...   4.3  82.6
2014-15  LAL   36  35  35  34.5  22.3  7.6  20.4  37.3  1.5 ...   6.9  81.3
2013-14  LAL   35   6   6  29.5  13.8  5.2  12.2  42.5  0.5 ...   3.5  85.7
2012-13  LAL   34  78  78  38.6  27.3  9.5  20.4  46.3  1.7 ...   8.0  83.9
2011-12  LAL   33  58  58  38.5  27.9  9.9  23.0  43.0  1.5 ...   7.8  84.5

         OREB  DREB  REB  AST  STL  BLK  TOV   PF
SEASON
2015-16   0.6   3.1  3.7  2.8  0.9  0.2  2.0  1.7
2014-15   0.7   4.9  5.7  5.6  1.3  0.2  3.7  1.9
2013-14   0.3   4.0  4.3  6.3  1.2  0.2  5.7  1.5
2012-13   0.8   4.7  5.6  6.0  1.4  0.3  3.7  2.2
2011-12   1.1   4.3  5.4  4.6  1.2  0.3  3.5  1.8

[5 rows x 23 columns]
```

3.0.2 Data Cleaning

As mentioned in the introduction, the data we collected was already very clean. The data is uniform, whitespace and formats are consistent, and there is no missing data for any player. The only issue that we had with the data was the following. On occasion, a player would play for more than one team in a season. In these cases, the dataset had a column for stats for each team the player played for in that season, as well as a column for the player's total stats for that season. We decided to only keep the total column to ensure consistency with the rest of the data. To do this, we used the following code:

```
In [4]: for key in player_data.keys():    # Keep only total row if player has
            p = player_data[key]          # more than one team in a season
            player_data[key] = p[~p.index.duplicated(keep='last')]
```

4 Feature Engineering

We decided to add four new columns with the goal of measuring a player's quality, or efficiency in a season. They are each explained and justified below. The following four formulae were used to calculate our efficiency columns:

```
(points + rebounds + assists + steals + blocks + field goals made + free throws made
- field goals attempted - free throws attempted - turnovers)/game.
```

```
(points + rebounds + assists + steals + blocks + field goals made + free throws made
- field goals attempted - free throws attempted - turnovers)/minute.

(points + rebounds + assists + steals + blocks)/game.

(points + rebounds + assists + steals + blocks)/minute.
```

We used the first metric becuase it is similiar to the way that the NBA also calculates the efficiency. Plotting the efficiency of each player, we found that the results we got generally matched the results of a players expected efficiency. In other words, players that are generally considered to be good, are plotted to be good, and vice versa. We chose the second metric because it uses the same statistics that the NBA efficiency rating does, but on a per minute rather than on a per game played basis. We chose the third metric because it doesn't take into account how many shots were taken or made, just the more game-impactful stats like points and assists. The last metric is the same as the previously described metric but on a per minute basis.

In summary, metrics 1 and 2 give players penalties for missed shots and turnovers, while metrics 3 and 4 don't. Also, metrics 1 and 3 are per game, while metrics 2 and 4 are per minute.

Our plan is to use the data we collected as well as these four columns we engineered to determine the best combination of data and machine learning models to predict career performance. Due to the accuracy of the data and the variety of efficiency metrics, we feel confident that these statistics will be suffient to make predictions. The following code was used to add these columns:

```
In [5]: for key in player_data.keys():
            p = player_data[key]
            # EFF1 (pts + reb + ast + stl + blk + fgm + ftm - fga - fta - tov)/game
            p["EFF1"] = (p["PTS"] + p["REB"] + p["AST"] +
                         p["STL"] + p["BLK"] + p["FGM"] +
                         p["FTM"] - p["FGA"] - p["FTA"] -
                         p["TOV"]) # Each of the stats are already per game

            # EFF2 (pts + reb + ast + stl + blk + fgm + ftm - fga - fta - tov)/minute
            p["EFF2"] = (p["PTS"] + p["REB"] + p["AST"] +
                         p["STL"] + p["BLK"] + p["FGM"] +
                         p["FTM"] - p["FGA"] - p["FTA"] -
                         p["TOV"])/p["MIN"] # (stat/game)/(min/game) = stat/min

            # EFF3 (pts + reb + ast + stl + blk)/game
            p["EFF3"] = (p["PTS"] + p["REB"] + p["AST"] +
                         p["STL"] + p["BLK"]) # Stats already per game

            # EFF4 (pts + reb + ast + stl + blk)/minute
            p["EFF4"] = (p["PTS"] + p["REB"] + p["AST"] +
                         p["STL"] + p["BLK"])/p["MIN"] # (stat/game)/(min/game) = stat/min
```

We also created a function that generates dataframes of rookie and career efficiencies for each of the four metrics, given a list of player names to use. This is mostly useful for plotting later.

```
In [6]: def create_df_rookie_career(players):
            """
```

```
    Create a DataFrame that contains rookie and career average
    efficiences for each of the four metrics. Only add player
    to dictionary if his name is found in players.
    Parameters:
        players (list): List of strings representing the players
            to add to the dictionary.
    Returns:
        A Pandas DataFrame containing the above described information.
    """
    # Initialize dictionary
    eff_dict = dict()
    for i in range(1,5):
        eff_dict["Rookie{}".format(i)] = []
        eff_dict["Career{}".format(i)] = []
    # Construct dictionary for DataFrame
    for p in players:
        for i in range(1,5):
            rookie = player_data[p]["EFF{}".format(i)].iloc[-1]
            career = player_data[p]["EFF{}".format(i)].mean()
            eff_dict["Rookie{}".format(i)].append(rookie)
            eff_dict["Career{}".format(i)].append(career)
    return pd.DataFrame(eff_dict)
```

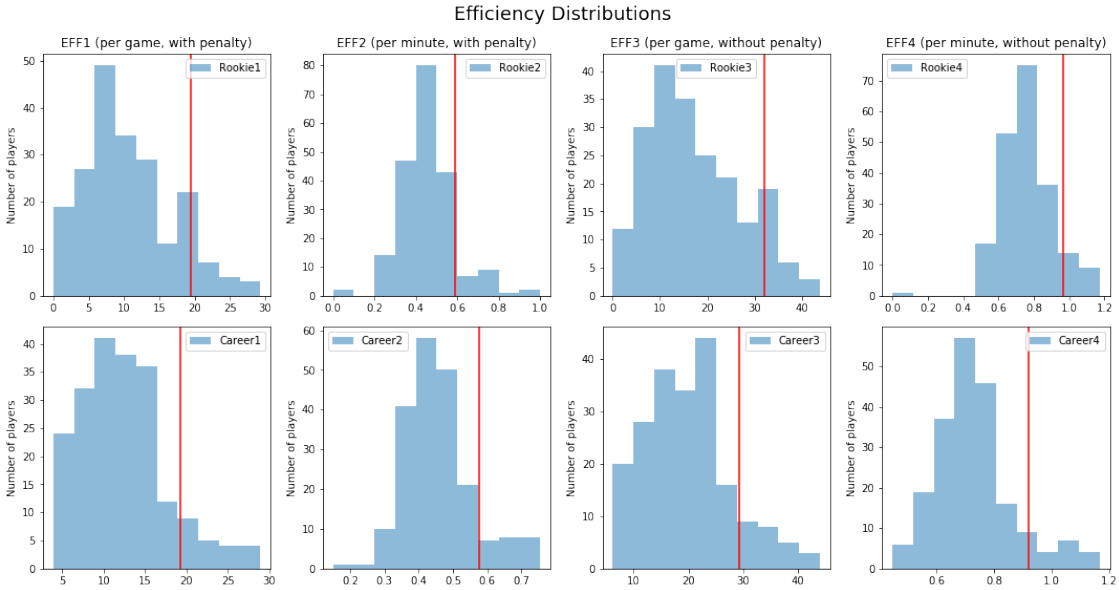
5 Data Visualization

5.0.1 Distributions of the 4 Efficiency Metrics

```
In [7]: # Plot histograms of the four metrics
        eff_df = create_df_rookie_career(players) # Use all players
        rookie_ninety, career_ninety =[],[]
        fig, axes = plt.subplots(nrows=2, ncols=4)
        for idx, (i, j) in enumerate([(0,0), (0,1), (0,2), (0,3)]):
            # For first row
            eff_df.plot(ax=axes[i,j], kind='hist', y=["Rookie{}".format(idx+1)], alpha = .5)
            axes[i,j].axvline(eff_df['Rookie{}'.format(idx+1)].quantile(.9) ,color = 'red')
            axes[i,j].set_ylabel("Number of players")
            # For second row
            eff_df.plot(ax=axes[i+1,j], kind='hist', y=["Career{}".format(idx+1)], alpha = .5)
            axes[i+1,j].axvline(eff_df['Career{}'.format(idx+1)].quantile(.9) ,color = 'red')
            axes[i+1,j].set_ylabel("Number of players")
            # Set titles
            if j == 0 or j == 1:
                penalty = ''
            else:
                penalty = 'out'
            if j == 0 or j == 2:
                frequency = 'game'
            else:
```

```
        frequency = 'minute'
    axes[i,j].set_title("EFF{} (per {}, with{} penalty)".format(idx+1, frequency, penal
    fig.suptitle('Efficiency Distributions',fontsize =18)
    plt.tight_layout()
    fig.subplots_adjust(top=.9)

    # Keep track of 90th percentile
    rookie_ninety.append(round(eff_df['Rookie{}'.format(idx+1)].quantile(.9) ,3))
    career_ninety.append(round(eff_df['Career{}'.format(idx+1)].quantile(.9) ,3))
```



We have plotted the rookie and career distributions for each efficiency to evaluate the spread, mean and outliers. We also plotted the 90th percentile line, to acknowledge what a high efficiency score looks like. For any efficiency metric a player will be considered good if they have an efficiency score above the the corresponding 90th percentile scores listed below. These observations will help us interpret which efficiency models to use when we are making our predictions.

```
In [8]: print("Good Efficiency scores:")
        print('Rookie: \nEFF1:', rookie_ninety[0], ' EFF2: ', rookie_ninety[1],
              ' EFF3: ', rookie_ninety[2],' EFF4: ', rookie_ninety[3])
        print('Career: \nEFF1:', career_ninety[0], ' EFF2: ', career_ninety[1],
              ' EFF3: ', career_ninety[2],' EFF4: ', career_ninety[3])
```

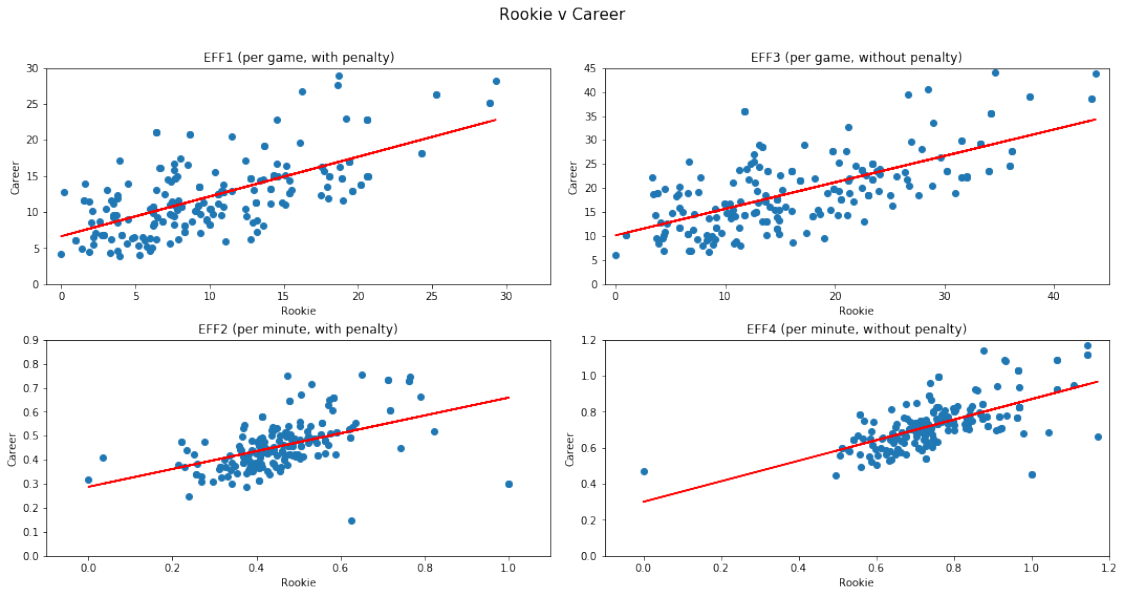
```
Good Efficiency scores:
Rookie:
EFF1: 19.4  EFF2:  0.589  EFF3:  32.0  EFF4:  0.965
Career:
EFF1: 19.225  EFF2:  0.578  EFF3:  29.3  EFF4:  0.921
```

5.0.2 Scatter Plot of Rookie vs Career Efficiencies

```
In [9]: def plot_rookie_v_career(player_list, title, subplots, xlims, ylims, nrows=2, ncols=2)
            """
            Plot a scatter plot of the rookie efficiency and the career
            efficiency for each metric and for each player in player_list.
            Parameters:
                player_list (list): List of players to use.
                title (str): Suptitle for plot.
                subplots (list): Subplots to use.
                xlims (list): List of xlims for each metric.
                ylims (list): List of ylims for each metric.
                nrows (int): number of rows to use.
                ncols (int): number of columns to use.
            """
            # Get dictionary of only desired players
            eff_dict = create_df_rookie_career(player_list)
            fig, ax = plt.subplots(nrows=nrows, ncols=ncols)
            for idx, subplot in enumerate(subplots):
                x, y = eff_dict['Rookie{}'.format(idx+1)],eff_dict['Career{}'.format(idx+1)]
                ax[subplot].scatter(x,y)
                # Plot regression line
                ax[subplot].plot(x, np.poly1d(np.polyfit(x, y, 1))(x), color = 'red')
                ax[subplot].set_xlabel("Rookie")
                ax[subplot].set_xlim(xlims[idx])
                ax[subplot].set_ylabel("Career")
                ax[subplot].set_ylim(ylims[idx])

                # Set titles
                j = idx+1
                if j == 1 or j == 2:
                    penalty = ''
                else:
                    penalty = 'out'
                if j == 1 or j == 3:
                    frequency = 'game'
                else:
                    frequency = 'minute'
                ax[subplot].set_title("EFF{} (per {}, with{} penalty)".format(idx+1, frequency
            fig.suptitle(title, fontsize=15)
            plt.tight_layout()
            fig.subplots_adjust(top=.88)

        plt.rcParams['figure.figsize'] = [15,8]
        xlims = [(-1,33),(-.1,1.1),(-1,45),(-.1,1.2)]
        ylims = [(0,30),(0,.9),(0,45),(0,1.2)]
        plot_rookie_v_career(players, 'Rookie v Career',[(0,0),(1,0),(0,1),(1,1)], xlims, ylims
```



These scatter plots show the relationship between a player's rookie efficiency and career efficiency for our four metrics. There is a lot of variance in efficiency measures 1 and 3 where the metric is measured per game. For efficiency 2 the graph is clustered however the regression line does not fit the plot as well as the other metrics. For efficiency 4, it is clear that the regression line fits the data better than the other three and the data has lower variance so we will be more likely to use efficiency 4.

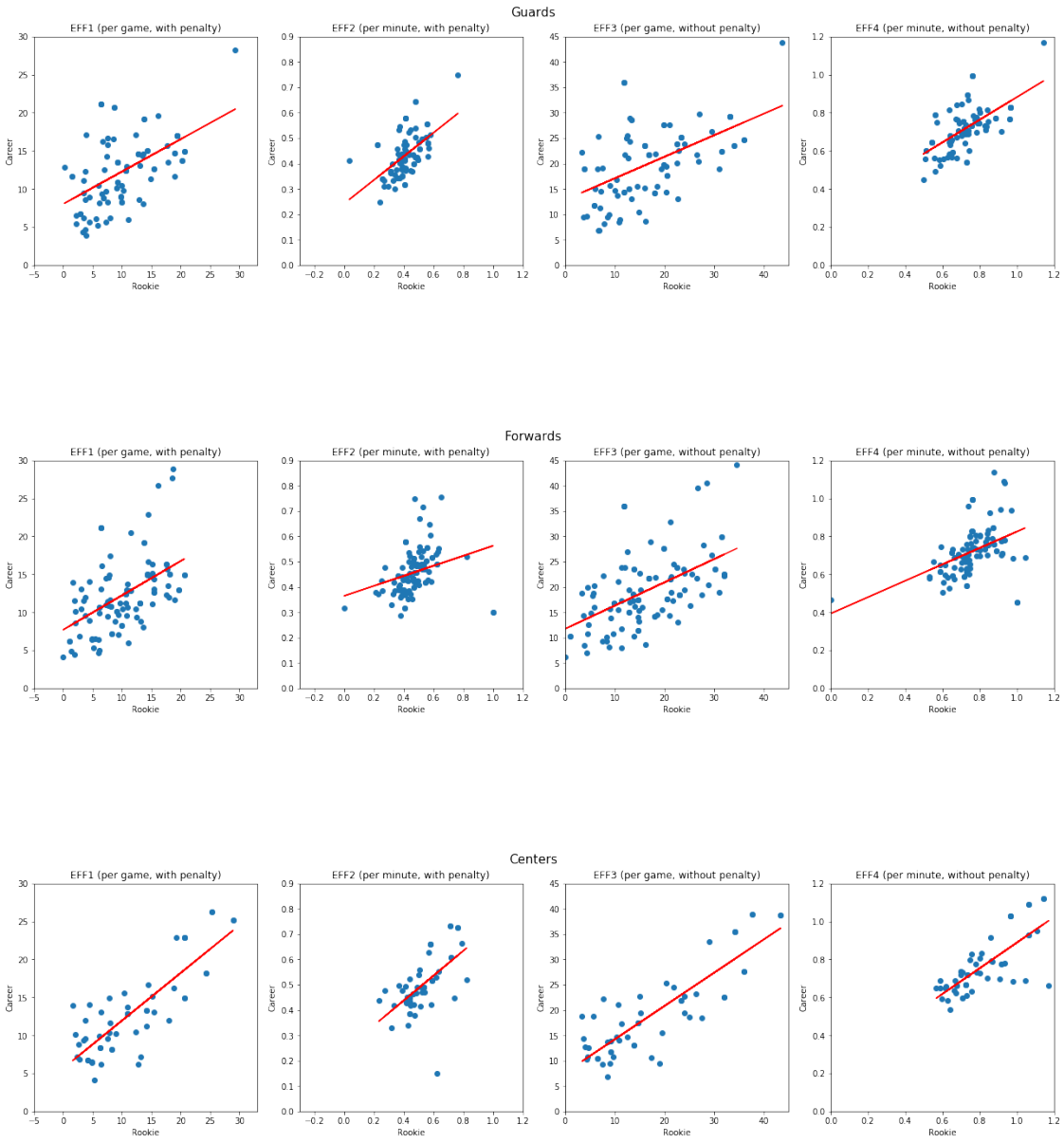
5.0.3 Efficiencies per Position

Below we will be sort the data to gather a list of players by position. The postions are Guard, Forward, Center. If a player is listed for two positions for example F-G then he would be included in both the guard and forward list.

```
In [10]: positions = {'GUARDS':[],'FORWARDS':[],'CENTERS':[]}
         for p in players:
             pos = p.split(',')[1][1:]
             if pos.find('G') >= 0:
                 positions['GUARDS'].append(p)
             if pos.find('F') >= 0:
                 positions['FORWARDS'].append(p)
             if pos.find('C') >= 0:
                 positions['CENTERS'].append(p)

         plt.rcParams['figure.figsize'] = [18,5]
         subplots = list(range(4))
         nrows, ncols = 1,4
         xlims = [(-5,33),(-.3,1.2),(0,45),(0,1.2)]
         ylims = [(0,30),(0,.9),(0,45),(0,1.2)]
         plot_rookie_v_career(positions['GUARDS'],'Guards', subplots,xlims,ylims,nrows,ncols)
```

```
plot_rookie_v_career(positions['FORWARDS'], 'Forwards',subplots,xlims,ylims,nrows,ncol
plot_rookie_v_career(positions['CENTERS'], 'Centers',subplots,xlims,ylims,nrows,ncols)
```



We can see that the data is more clustered and displayed similarly for guards and forwards. Centers seem to have the most accurate regression lines. From first glance, the most accurate efficiency measure for guards and forwards is 4 but for centers is 2. This leads us to believe that we will seperate the players into position before we begin to analyze and predict the data.

5.0.4 Plot the 4 efficiency metrics of sample players

In this section, we will use various plots to highlight trends that we notice in our data. Specifically, we will point out differences in the four metrics we decided to use to measure the quality of a

player's season. As it would be useless to put all player's data on one graph, we will only use a selected subset of players that we chose to try to get a picture of a wide range of career qualities.

In [11]:
```
# Collect all the player data into combined DataFrame to ease plotting
eff_dicts = [{}, {}, {}, {}]
effs = ["EFF1", "EFF2", "EFF3", "EFF4"]
for p in players:
    for i, eff in enumerate(effs):
        eff_dicts[i][p] = player_data[p][eff][::-1] # Data is stored newest to oldest
efficiencies = pd.DataFrame(eff_dicts)
```

In [13]:
```
# get the last 5 players to show in plot
test_players = players[-5:]
y = [efficiencies[p] for p in test_players]

#Plot the data
fig, ax = plt.subplots(nrows=2, ncols=2)
plt.rcParams['figure.figsize'] = [15,9]
for idx, (i, j) in enumerate([(0,0), (1,0), (0,1), (1,1)]):
    for k, ys in enumerate(y):
        x = np.linspace(1,len(ys[idx]),len(ys[idx]))
        ax[i,j].plot(x, ys[idx],label = test_players[k])
        ax[i,j].set_xticks(np.arange(0,21))
    ax[i,j].set_xlabel("Seasons in NBA")
    ax[i,j].set_ylabel("Efficiency ({})".format(idx+1))
    ax[i,j].set_title("EFF{}".format(idx+1))
    ax[i,j].legend()
    fig.suptitle('Efficiency Metrics for 5 players', fontsize = 14)
    plt.tight_layout()
    fig.subplots_adjust(top=.9)
```

In general we can see that all four efficiency metrics calculate a players performance accurately. The all-star players, Kobe Bryant and Lebron James, have a higher efficiency the other non-all star players.

5.0.5 Exploring Correlation between Team and Player Efficiency

We considered looking at the effect that a player's team had on his efficiency for a given season. However, due to our small sample size and the way we randomly selected the data, we did not collect enough infomation to adequately explore that question. For example, to get an idea of a team's effect on a player's efficiency for a given season, we would need to obtain the data for every player for that season. This may introduce bias into our dataset if we only did this for one season and is beyond the scope of our project. Hence, though we did consider this as an interesting option, we decided it was out of the frame of our question to explore.

6 Conclusion

In summary, we scraped data from the NBA website and cleaned it appropriately. We also added our own columns corresponding to the efficiency rating of players for each season. We feel confident that the data that we have collected and produced will be sufficient for predicting NBA players' future career success based on their rookie stats.

After examining and visualizing the data for our various metrics, we learned a couple of things. First, we noticed the efficiency rate for the average career player and the superstar player was often very close for their rookie year. Second, we learned that we will likely have to compare metrics to determine which one ends up being a better predictor of future success. However, plotting the rookie vs career efficiencies we saw that the fourth efficiency measure had a relatively low variance and a linear trend. This measure will most likely be our best and the one we will focus on

when we use our machine learning techniques to make predictions. Lastly, we noticed the metrics behaved differently for each position. This led us to conclude that we may have to split the data by position before making any further analysis and predictions.

Overall, the accuracy and linear trends of these efficiency metrics will help us to answer our question by allowing us to evaluate a player's rookie performance and predict how good that player will be for his career.

6.0.1 Works Cited

Arbesman, Samuel. "Luck and Skill Untangled: The Science of Success." Wired, Conde Nast, 3 June 2017, www.wired.com/2012/11/luck-and-skill-untangled-qa-with-michael-mauboussin/.