hotHandEffect

October 1, 2017

1 Investigation of "Hot-hand effect"

There is a widely held belief in basketball that some players have periods of time where they are much better shooters than normal. This can be called having a "hot hand". An example would be if a player hit 3 shots in a row, then many fans would expect that player to be more likely (than their usual percentage) to hit their next shot.

Whether this is a real effect or some kind of cognitive bias has been previously studied. For instance in 1984(83?, 85? FIX THIS), [RESEARCHERS] looked at actual shooting results of the Philadelphia 76ers, free throws of the Boston Celtics, and controlled-experiment shots of college students. Their data did not support the existence of a hot-hand effect. This question has been revisited, for instance in [PAPERS FROM 2005?? or so??] which also failed to find evidence to support the existence of the hot-hand effect. Interestingly, in 2015 [RESEARCHERS] noted that the sampling method used in previous studies was flawed. The flaw is subtle, but leads to some evidence for the hot-hand effect.

My intent is to investigate recent shooting results of NBA players with the goals of:

Looking for evidence to support (or reject) the existence of the hot-hand effect.

Understand the subtleties of the sampling flaw found by [RESEARCHERS].

1.1 Getting the data

The reason I chose to study NBA data was that I found a resource that makes shooting data easy to download for the NBA: www.nbasavant.com. I have downloaded all shooting data for 2016-2017 and placed it in the files nba_savant__.csv in the data/nba_savant folder. [Note: you can ostensibly download a .csv file of the shots data for the entire year, but the files seem to be limited to 50,000 lines which is not enough. That's why I split the data by month when downloading.]

There is some concern about the complete validity of the data. My biggest concern is the data for April 2017 seems to be incomplete. There are not near enough total shots for the month and spot checking some players shows many fewer shots than expected for that player. However, working with this data will at least provide a framework for studying similar datasets.

Before reading in the data, we will load pandas, a python library used for data analysis.

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

Now, we can read in the data from the downloaded .csv files and store the data as a DataFrame (a pandas data structure).

```
In [2]: import glob
        files_to_read = glob.glob('../data/nba_savant/*.csv')

        shots = pd.concat((pd.read_csv(f) for f in files_to_read))
```

We can look at the first few rows of data to get a sense of what data we have (and if the read did what we expected it to do):

```
In [3]: shots.head()
```

```
Out[3]:             name            team_name   game_date  season  espn_player_id  \
        0  Andre Drummond     Detroit Pistons  2017-01-01    2016          6585.0
        1    Nerlens Noel  Philadelphia 76ers  2017-01-30    2016       2991280.0
        2       Jon Leuer     Detroit Pistons  2017-01-01    2016          6452.0
        3   Dwight Howard       Atlanta Hawks  2017-01-29    2016          2384.0
        4  Andre Drummond     Detroit Pistons  2017-01-01    2016          6585.0

              team_id  espn_game_id  period  minutes_remaining  seconds_remaining  \
        0  1610612765     400899381       4                  5                 57
        1  1610612755             0       3                  7                 18
        2  1610612765     400899381       1                  3                 29
        3  1610612737     400900132       4                  0                 45
        4  1610612765     400899381       1                  7                 57

              ...        shot_type shot_distance          opponent  x  y  dribbles  \
        0     ...     2PT Field Goal             0        Miami Heat  0  1         0
        1     ...     2PT Field Goal             0  Sacramento Kings  0  1         0
        2     ...     2PT Field Goal             0        Miami Heat  0  1         0
        3     ...     2PT Field Goal             0   New York Knicks  0  1         0
        4     ...     2PT Field Goal             0        Miami Heat  0  1         0

           touch_time  defender_name  defender_distance  shot_clock
        0         0.0            NaN                0.0         0.0
        1         0.0            NaN                0.0         0.0
        2         0.0            NaN                0.0         0.0
        3         0.0            NaN                0.0         0.0
        4         0.0            NaN                0.0         0.0

        [5 rows x 22 columns]
```

Each row of the DataFrame consists of one shot (the observation) and 22 variables. Those variables are the columns. Note that the ... indicates we are not seeing all of the columns. Pandas has a setting that gives the maximum number of columns to print. It appears that value defaults to 20. We can increase this value and then look at the first few rows again (using new max of 60, but 22 would suffice).

```
In [4]: pd.set_option('display.max_columns', 60)
        shots.head()
```

```
Out[4]:             name            team_name   game_date  season  espn_player_id  \
        0  Andre Drummond     Detroit Pistons  2017-01-01    2016          6585.0
        1    Nerlens Noel  Philadelphia 76ers  2017-01-30    2016       2991280.0
        2       Jon Leuer     Detroit Pistons  2017-01-01    2016          6452.0
        3   Dwight Howard       Atlanta Hawks  2017-01-29    2016          2384.0
        4  Andre Drummond     Detroit Pistons  2017-01-01    2016          6585.0

              team_id  espn_game_id  period  minutes_remaining  seconds_remaining  \
        0  1610612765     400899381       4                  5                 57
        1  1610612755             0       3                  7                 18
        2  1610612765     400899381       1                  3                 29
        3  1610612737     400900132       4                  0                 45
        4  1610612765     400899381       1                  7                 57

           shot_made_flag          action_type       shot_type  shot_distance  \
        0               1  Alley Oop Dunk Shot  2PT Field Goal              0
        1               1  Alley Oop Dunk Shot  2PT Field Goal              0
```

```
2              0  Alley Oop Dunk Shot  2PT Field Goal              0
3              1  Alley Oop Dunk Shot  2PT Field Goal              0
4              1  Alley Oop Dunk Shot  2PT Field Goal              0

           opponent  x  y  dribbles  touch_time  defender_name  \
0        Miami Heat  0  1         0         0.0            NaN
1  Sacramento Kings  0  1         0         0.0            NaN
2        Miami Heat  0  1         0         0.0            NaN
3   New York Knicks  0  1         0         0.0            NaN
4        Miami Heat  0  1         0         0.0            NaN

   defender_distance  shot_clock
0                0.0         0.0
1                0.0         0.0
2                0.0         0.0
3                0.0         0.0
4                0.0         0.0
```

1.2 Strategy

For an initial investigation, I plan on making the following assumptions:

Each player's shots are to be investigated as one sequence throughout the entire year. A different, and perhaps more useful, choice would be to split up each players shots by game. We will look at the data split by game later.

To consider the existence of the hot-hand effect we will only look at whether the previous shot was a make or miss. The literature standard seems to be looking at shooting percentages after streaks of 1, 2, or 3 consecutive makes or misses. We will handle more complicated scenarios later.

The only variables we will consider for analyzing each shot are those that determine:

Which player took the shot (name, espn_player_id)

When the shot was taken (game_date, period, minutes_remaining, seconds_remaining) [Note: these are used to determine the order the shots occurred in.]

Whether the shot was a make or miss (shot_made_flag)

We will not, at least initially, be considering other variables such as those associated to shot difficulty, opponents, or effects of other players shooting on a given night.

1.3 Rearranging the data

Let's start by removing columns we are not interested in. Actually, we are only keeping the columns we are interested in.

In [5]:
```
shots = shots[:][['name', 'game_date', 'espn_player_id', 'period', 'minutes_remaining', 'seconds
shots.head()
```

Out[5]:
```
             name   game_date  espn_player_id  period  minutes_remaining  \
0  Andre Drummond  2017-01-01          6585.0       4                  5
1    Nerlens Noel  2017-01-30       2991280.0       3                  7
2       Jon Leuer  2017-01-01          6452.0       1                  3
3   Dwight Howard  2017-01-29          2384.0       4                  0
4  Andre Drummond  2017-01-01          6585.0       1                  7

   seconds_remaining  shot_made_flag
0                 57               1
1                 18               1
2                 29               0
```

```
3                  45                1
4                  57                1
```

Now we can sort the dataframe by date. That will help us to easily find the previous shot for each player.

In [6]:
```
shots = shots.sort_values(by=['game_date','period','minutes_remaining','seconds_remaining'],axis
shots.head()
```

Out[6]:
```
                    name   game_date  espn_player_id  period  \
6243         Rodney Hood  2016-10-25       2581177.0       1
675         Derrick Rose  2016-10-25          3456.0       1
3037  LaMarcus Aldridge  2016-10-25          2983.0       1
2133      AlFarouq Aminu  2016-10-25          4248.0       1
1792          Kevin Love  2016-10-25          3449.0       1

      minutes_remaining  seconds_remaining  shot_made_flag
6243                 11                 44               1
675                  11                 40               1
3037                 11                 36               0
2133                 11                 27               1
1792                 11                 26               0
```

Now for the more interesting processing step. This will involve:
Group the shots by player.
Add a 'previous_shot_made_flag' for every shot that indicates if the player made the previous shot.
Group and aggregate each player's data by 'previous_shot_made_flag' to calculate the mean shooting percentage for both when the player made/missed the previous shot.
Collect the individual player percentages on made/missed shot back into a dataframe.

In [7]:
```
shots_grouped_by_player = shots.groupby('espn_player_id')

# to collect the perecentages afer make/miss for each player
player_shot_dict = {}

for id,group in shots_grouped_by_player:
    player_stats = {}
    group['previous_shot_made_flag'] = group['shot_made_flag'].shift(1)
    player_stats[group.iloc[0]['espn_player_id']] = group[['shot_made_flag','previous_shot_made_

    # filter out players who did not attempt at least one shot after both a make and a miss
    if not player_stats[id].empty and player_stats[id].index.size == 2:
            player_shot_dict[id] = {'percent_after_miss':player_stats[id].loc[0]['mean'], 'perce
player_shot_df = pd.DataFrame.from_dict(data=player_shot_dict,orient='index')
player_shot_df.head()
```

```
-c:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing
```

Out[7]:
```
       percent_after_miss  percent_after_make
25.0             0.285714            0.000000
136.0            0.380435            0.412429
165.0            0.430435            0.386997
272.0            0.385593            0.386667
558.0            0.285714            0.500000
```

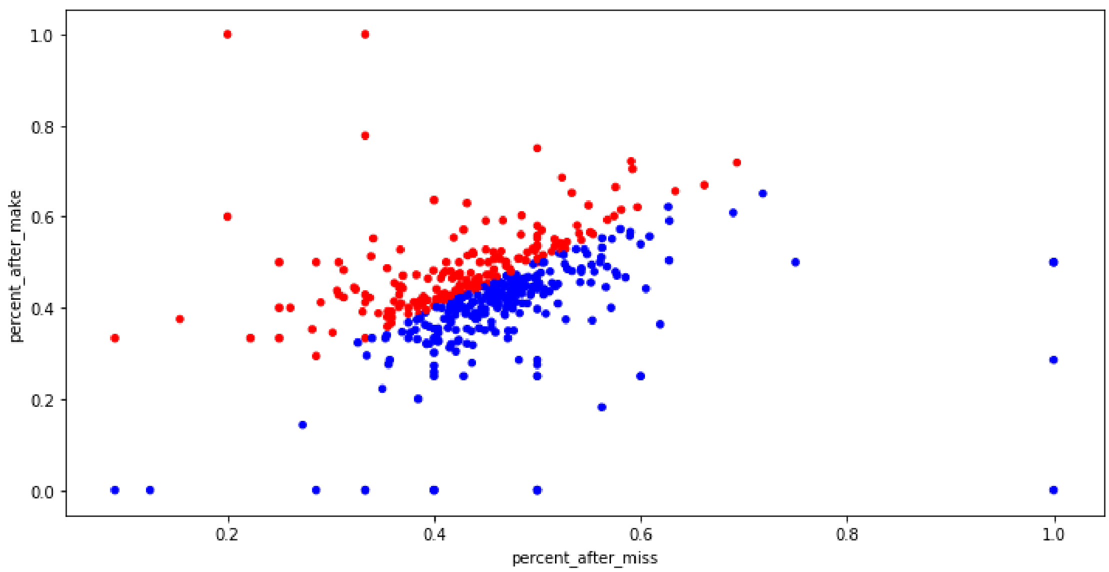
We now have the shooting percentage for every player both after a miss and after a make. If shooters do not get "hot", then it seems reasonable that for each player that percent_after_miss and percent_after_make should be roughly equal. If shooters do get "hot", then it seems reasonable that for each player that percent_after_miss should typically be less than percent_after_make.

1.4 Analyzing the data

To start investigating these questions we can first plot the data in a scatter plot. Points will be plotted with red points for a player that has a higher percentage after a make and blue points for a player that has a higher percentage after a miss.

```
In [8]: colors = np.where(player_shot_df['percent_after_miss']>player_shot_df['percent_after_make'], 'b
        player_shot_df.plot(x='percent_after_miss', y='percent_after_make', kind='scatter', c=colors, f
```

```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb4b805c8d0>
```



The previous scatter plot does not seem to allow us to defnitively conclude much of anything. Players seem roughly split between those that shoot better after a miss and those that shoot better after a make.

We can run statistical tests to evaluate the results objectively.

The first test we will try is a t-test. Our null hypothesis will be that shooting percentage after a make and shooting percentage after a miss are the same. The alternative hypothesis will be that shooting percentage after a make is greater than shooting percentage after a miss. Let's assume an alpha value of 0.05 which means that we need a p-value less than 0.05 to reject the null hypothesis and conclude the data support that players have a better shooting percentage after a make. Note that this results in a one-tailed test. Annoyingly, python's standard libraries seem to only compute p-values for a two-tailed test. That means we need to divide the python calculated p-value by 2 and then possibly subtracting from 1 (depending on whether the t-statistic is positive or negative) to get our true p-value.

```
In [9]: import scipy.stats

        (t, p) = scipy.stats.ttest_1samp(player_shot_df.percent_after_make-player_shot_df.percent_after_
        if t > 0:
            # t>0 implies percent_after_make is generally greater than percent_after_miss
            # so this is the tail that (at least somewhat) supports our alternative hypothesis
```

```
        p = p/2
    else:
        # t<0 implies percent_after_make is generally less than percent_after_miss
        # so this is the tail that definitely does not support our alternative hypothesis
        p = 1 - p/2
    p
```

Out[9]: 0.99951561688842272

So, our p-value is (much) larger than 0.05 which means the data do not support the hot hand hypothesis. In fact, the data would have supported the hypothesis that the average shooting percentage of players is higher after a miss than after a make.

Where to go from here? There are a number of things we could tidy up. Some of these include the following.

Each player's shots are viewed as one sequence for the entire year. We could split each player's shots on a per game basis.

The t-test assumes that the distribution of differences between percent after make and miss is normally distributed. We could look into the validity of that assumption.

There seem to be a fair number of outliers in the scatter plot. It seems likely that a lot of the outliers are players that took very few shots. One option to deal with the outliers would be to only consider players that took at least a certain number of shots. That seems tempting at first, but once we split each players shots into individual games we will need to consider relatively small numbers of shots anyway. Hopefully, we can use appropriate statistics to account for a player with a small number of shots. The essential problem we have is that we are using something roughly akin to an average-of-averages which is problematic with each inner average having a different sample size.

We can extend our analysis to look at more than the previous shot. Maybe the previous n shots for some n=2 or n=3.

We could account for the different shot types. Maybe do something like only looking at 3-point shots or jump shots.

We could apply machine learning (or other) techniques to look for patterns in the players that do have a better shooting percentage after a make or a miss.

We could investigate the grouping step of the data arranging. This step takes a fair amount of time. Perhaps there is a more efficient way to perform the same task.