Using Regression Analysis to Test the "Hot Hand"

In this section, we will use regression analysis to test for the "hot hand."

Import useful libraries and the shot log data

In [ ]:
```
import pandas as pd
import numpy as np
import datetime as dt
import statsmodels.formula.api as sm
import matplotlib.pyplot as plt
import seaborn as sns

Shotlog=pd.read_csv("../../Data/Week 6/Shotlog2.csv")
Player_Stats=pd.read_csv("../../Data/Week 6/Player_Stats2.csv")
Player_Shots=pd.read_csv("../../Data/Week 6/Player_Shots2.csv")
Shotlog.head()
```

Prediction Error

Let's create a variable that equals to the difference between the outcome of the shot and the average success rate. Since we typically use the average success rate to predict the outcome of the shot, this difference will capture the prediction error.

In [ ]:
```
Shotlog['error']=Shotlog['current_shot_hit']-Shotlog['average_hit']
Shotlog['lagerror']=Shotlog['lag_shot_hit']-Shotlog['average_hit']
```

We can graph the outcome of the shots to see if there is any pattern over time in the variable.

We will look at LeBron James' performance during the regular season as an example.

In [ ]:
```
Shotlog['time'] = pd.to_timedelta(Shotlog['time'])
Shotlog['time'].describe()
```

We will first graph the outcome of LeBron James' shots in a single game on April 9th, 2017.

(To make this graph, we use a small trick. Instead of asking Python to produce a scatter plot with the "plot.scatter" command, we ask Python to graph a line plot, but specify the width of the line to be 0. So essentially we produce a scatter plot. The reason we do it this way is because in Python, scatter plot requires the x axis to be numeric. It does not allow scatter plot where the x axis is a date or time variable.)

In [ ]:
```
Shotlog[(Shotlog.shoot_player == 'LeBron James')&(Shotlog.date=='2017-04-09')].plot(x='time', y='current_shot_hit', marker='
```

Let's create a graph of the outcomes of individual shots for Lebron James throughout the regular season. We will create a subgraph for each game he played.

We will first subset a dataset that includes only LeBron James' data.

In [ ]:
```
LeBron_James=Shotlog[(Shotlog.shoot_player == 'LeBron James')]
LeBron_James.head()
```

Now we can graph prediction error for LeBron James for all the games separately in the season.

In [ ]:
```
g = sns.FacetGrid(LeBron_James, col="date", col_wrap=4)
g = g.map(plt.plot, "time", "current_shot_hit", marker='o', linewidth=0)
g.set_axis_labels("Game", "Shots");
```

We will do a similar exercise for the statistics of Cheick Diallo.

In [ ]:
```
Cheick_Diallo=Shotlog[(Shotlog.shoot_player == 'Cheick Diallo')]
g = sns.FacetGrid(Cheick_Diallo, col="date", col_wrap=4)
g = g.map(plt.plot, "time", "current_shot_hit", marker='o', linewidth=0)
```

Self Test - 1

Graph the prediction error for James Jones

- Separate the shots by game
- Interpret your result

In [ ]:
```
#Your Code Here
```

Regression analysis on prediction error

We will first run a simple regression of the prediction error of current period on the prediction error of previous period.

In [ ]:
```
reg1 = sm.ols(formula = 'error ~ lagerror', data= Shotlog).fit()
print(reg1.summary())
```

The estimated coefficient of the lagged error is statistically significant. However, the R-Squared for this regression is also zero. This means that our specified linear model is not a good fit for our data at all!

There are a lot of factors that may influence the success of shot, for example, the player's own skill as a shooter, the type of the shot, the atmosphere of the stadium (whether it is home or away game), and whether it is at the beginning or towards the end of the game. Let's add these control variables in our regression.

In [ ]:
```
reg2 = sm.ols(formula = 'error ~ lagerror+player_position+home_game+opponent_previous_shot+C(points)+time_from_last_shot+C(qu
print(reg2.summary())
```

We can see that the R-squared is now increased to 0.015 which is still very small. The estimate on lagerror is statistically significant, but the magnitude of the estimate is -0.0136 which is still very small. And it is negative, meaning that the success of the previous shot would hurt the chance of the subsequent shot.

This is contrary to what the hot hand predicts.

Weighted least squares regression

As we have seen, some players had a lot of shot per game while some just had a few. Different players may have different variations in their success rate in the shots. We can run a weighted least squared regression to address this problem.

Weighted least squares estimation weights the observations proportional to the reciprocal of the error variance of the observation. Thus weighted least squares can overcome the issue of non-constant variance.

We can use the "sm.wls" command to run the weighted least square regression weighted by the number of shot per game (weight=1/shot_per_game).

In [ ]:
```
reg3 = sm.wls(formula = 'error ~ lagerror+player_position+home_game+opponent_previous_shot+points+time_from_last_shot+quarter
print(reg3.summary())
```

From our summary statistics, some players exhibit a stream of the success while some don't. In our previous regressions, we are grouping all the players together. Let's see if we can find any effect if we look at individual players.

Regression analysis on individual players

Run a regression of current error on lagged error for LeBron James.

In [ ]:
```
reg_LeBron = sm.ols(formula = 'error ~ lagerror+home_game+opponent_previous_shot+C(points)+time_from_last_shot+C(quarter)', d
print(reg_LeBron.summary())
```

Similarly, we can run a weighted least squares estimation on LeBron James' prediction error, weighted by the number of shot he made in each game.

In [ ]:
```
reg_LeBron_wls = sm.wls(formula = 'error ~ lagerror+home_game+opponent_previous_shot+points+time_from_last_shot+quarter',  we
print(reg_LeBron_wls.summary())
```

We can also take a look back at LeBron James' autocorrelation coefficient.

In [ ]:
```
Shotlog[(Shotlog.shoot_player == 'LeBron James')][['current_shot_hit','lag_shot_hit']].corr()
```

The autocorrelation coefficient between the outcomes of the current shot and the previous shot for LeBron James is very small.

We can do a similar exercise for James Jones. We will start with an ordinary least square regression.

In [ ]:
```
reg_Jones = sm.ols(formula = 'error ~ lagerror+home_game+opponent_previous_shot+C(points)+time_from_last_shot+C(quarter)', da
print(reg_Jones.summary())
```

We will also run a weighted least squares estimation on Jones' statistics. Weight=1/shot_per_game.

In [ ]:
```
reg_Jones_wls = sm.wls(formula = 'error ~ lagerror+home_game+opponent_previous_shot+points+time_from_last_shot+quarter',  wei
print(reg_Jones_wls.summary())
```

Self Test - 2

Use regression analysis to test "hot hand" for Cheick Diallo

1. Run an ordinary least square regression of current error on lagged error for Cheick Diallo.
2. Run a weighted least sqaure regression of current error on lagged error for Cheick Diallo, weight=1/shot_per_game.
3. Interpret your regression results.

In [ ]:
```
#Your Code Here
```

In [ ]:
```
#Your Code Here
```

More generally, we can define functions to run regressions for each individual player.

- Define a function to run ordinary least square regression by player.

In [ ]:
```
def reg_player(player):
    Shotlog_player=Shotlog[Shotlog.shoot_player==player]
    reg_player=sm.ols(formula = 'error ~ lagerror+home_game+opponent_previous_shot+points+time_from_last_shot+quarter', data=
    print(reg_player.summary())
    return;
```

We can then use this function for individual player, for example, Russell Westbrook.

In [ ]:
```
reg_player('Russell Westbrook')
```

- Define a function to run weighted least square regression by player.

In [ ]:
```
def reg_wls_player(player):
    Shotlog_player=Shotlog[Shotlog.shoot_player==player]
    reg_wls_player=sm.wls(formula = 'error ~ lagerror+home_game+opponent_previous_shot+points+time_from_last_shot+quarter',we
    print(reg_wls_player.summary())
    return;
```

Let's use this function to run a weighted least squares estimation for Russell Westbrook.

In [ ]:
```
reg_wls_player('Russell Westbrook')
```

We can extract estimated coefficient on the lagged error for each player.

- Create a list of unique player names

In [ ]:
```
player_list = np.array(Shotlog['shoot_player'])
player_list = np.unique(player_list)
```

In [ ]:
```
player_list[0]
```

- Run weighted least squares regression for each player by specifying "shoot_play==player_list[index]"

In [ ]:
```
Shotlog_player=Shotlog[Shotlog.shoot_player==player_list[0]]
reg_player=sm.wls(formula = 'error ~ lagerror+home_game+opponent_previous_shot+points+time_from_last_shot+quarter', weights=1
print(reg_player.summary())
```

- Extract the estimated coefficients, along with the p-value and t-statistics of the estimates and store them in a dataframe

In [ ]:
```
RegParams = pd.DataFrame(reg_player.params).reset_index()
RegTvals = pd.DataFrame(reg_player.tvalues).reset_index()
RegPvals = pd.DataFrame(reg_player.pvalues).reset_index()

RegOutput = pd.merge(RegParams, RegTvals, on=['index'])
RegOutput = pd.merge(RegOutput, RegPvals, on=['index'])
RegOutput
```

- Write a loop to extract regression outputs for each player

In [ ]:
```
i = 0
Player_Results = {}
while i <= len(player_list) - 1:
    Shotlog_player=Shotlog[Shotlog.shoot_player==player_list[i]]
    reg_player=sm.wls(formula = 'error ~ lagerror+home_game+opponent_previous_shot+points+time_from_last_shot+quarter', weigh
    RegParams = pd.DataFrame(reg_player.params).reset_index()
    RegTvals = pd.DataFrame(reg_player.tvalues).reset_index()
    RegPvals = pd.DataFrame(reg_player.pvalues).reset_index()

    RegOutput = pd.merge(RegParams, RegTvals, on=['index'])
    RegOutput = pd.merge(RegOutput, RegPvals, on=['index'])
    RegOutput

    LagErr = RegOutput[RegOutput['index'] == 'lagerror']
    LagErr = LagErr.drop(columns=['index'])
    LagErr = LagErr.rename(columns={"0_x":"Coef", "0_y":"T_Statistics", 0:"P_Value"})
    LagErr['shoot_player'] = player_list[i]
    Headers = ['shoot_player', 'Coef', 'T_Statistics', 'P_Value']
    Player_Results[i] = LagErr[Headers]
    i = i+1
```

- Write another loop to build a dataframe to store the regression output for all the players

In [ ]:
```
RegPlayer = Player_Results[0]
j = 1
while j <= len(player_list) - 1:
    RegPlayer = RegPlayer.append(Player_Results[j])
    j = j+1
RegPlayer = RegPlayer.reset_index()
RegPlayer = RegPlayer.drop(columns=['index'])
RegPlayer
```

- Merge the total number of shots captured in "Player_Shots" to the regression result dataframe. This total number of shots represents the sample size of each regression

In [ ]:
```
RegPlayer=pd.merge(RegPlayer, Player_Shots, on=['shoot_player'])
RegPlayer.head()
```

- Display players with statistically significant estimates on the lagged error variable

In [ ]:
```
display(RegPlayer.loc[RegPlayer['P_Value']<=0.05])
```

There are a total of 38 players with statistically significant estimates on the lagged error variable, that is, the success of their previous shots impact the success rate of their current shot. Interestingly, more than half of these estimates are negative, which means that a success in the previous shot actually hurts the chance of scoring in the current shot. This is the opposite of a "hot hand."

Overall from our regression analyses, 8 players, Boris Diaw, Brandon Rush, Frank Kaminsky, Joe Young, Jose Calderon, Kyle Wiltjer, Omri Casspi, Robert Covington, and Tony Parker have positive and statistically significant estimate on the lagged error variable. Thus, these players may have "hot hand." Note that the estimate for Kyle Wiltjer is 1 and there are only a total of 14 observations for him. We need to interpret his result with caution.

In [ ]:
```
#Save updated data to csv file
Shotlog.to_csv("../../Data/Week 6/Shotlog3.csv")
Player_Stats.to_csv("../../Data/Week 6/Player_Stats3.csv", index=False)
Player_Shots.to_csv("../../Data/Week 6/Player_Shots3.csv", index=False)
```

In [ ]:
```

```