We will use the 2016-2017 basketball shot log data to demonstrate how to test the hot hand.

Import useful libraries and the shot log data

Please note that the 3 lecture notebooks for this week must be run in order, as the following notebooks rely on the output of the previous

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

Shotlog=pd.read_csv("../../Data/Week 6/Shotlog_16_17.csv")
Shotlog.head()
```

Out[1]:

	team_previous_shot	player_position	home_game	location_x	opponent_previous_shot	home_team	shot_type	points	away_team	location_y	time	
0	NaN	SF	Yes	97.0	SCORED	ATL	Pullup Jump Shot	2	WAS	405.0	1:09	10
1	MISSED	C	Yes	52.0	SCORED	ATL	Tip Dunk Shot	2	WAS	250.0	1:11	10
2	SCORED	SG	Yes	239.0	MISSED	ATL	Jump Shot	2	WAS	223.0	1:41	10
3	SCORED	PG	Yes	102.0	SCORED	ATL	Pullup Jump Shot	2	WAS	385.0	2:16	10
4	SCORED	PF	Yes	128.0	MISSED	ATL	Turnaround Jump Shot	2	WAS	265.0	2:40	10

In [2]:
```
Shotlog.shape
```

Out[2]: (210072, 16)

Data Preparation

Missing Value

In [3]:
```
Shotlog.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210072 entries, 0 to 210071
Data columns (total 16 columns):
team_previous_shot        207612 non-null object
player_position           210072 non-null object
home_game                 210072 non-null object
location_x                209675 non-null float64
opponent_previous_shot    208462 non-null object
home_team                 210072 non-null object
shot_type                 210072 non-null object
points                    210072 non-null int64
away_team                 210072 non-null object
location_y                209675 non-null float64
time                      210072 non-null object
date                      210072 non-null object
shoot_player              210072 non-null object
time_from_last_shot       200072 non-null float64
quarter                   210072 non-null int64
current_shot_outcome      210072 non-null object
dtypes: float64(3), int64(2), object(11)
memory usage: 25.6+ MB
```

Let's create some useful variables.

- Create dummy variables to indicate hit or miss of current shot and previous shot.

In [4]:
```
Shotlog['current_shot_hit'] = np.where(Shotlog['current_shot_outcome']=="SCORED", 1, 0)
Shotlog.head()
```

Out[4]:

	team_previous_shot	player_position	home_game	location_x	opponent_previous_shot	home_team	shot_type	points	away_team	location_y	time	
0	NaN	SF	Yes	97.0	SCORED	ATL	Pullup Jump Shot	2	WAS	405.0	1:09	10
1	MISSED	C	Yes	52.0	SCORED	ATL	Tip Dunk Shot	2	WAS	250.0	1:11	10
2	SCORED	SG	Yes	239.0	MISSED	ATL	Jump Shot	2	WAS	223.0	1:41	10
3	SCORED	PG	Yes	102.0	SCORED	ATL	Pullup Jump Shot	2	WAS	385.0	2:16	10
4	SCORED	PF	Yes	128.0	MISSED	ATL	Turnaround Jump Shot	2	WAS	265.0	2:40	10

- Make sure the variable "date" is stored as a date type variable.

In [5]:
```
import datetime as dt
Shotlog['date']=pd.to_datetime(Shotlog['date'])
```

- Convert the variable "time" to be datetime type variable
  1. We will first add the hour (00) to the time variable since the time variable will be stored in the format 'HH:MM:SS';
  2. We will use "to_timedelta" to work with variable with only time information.

In [6]:
```
Shotlog['time'] = pd.to_timedelta('00:'+ Shotlog['time'])
Shotlog['time'].describe()
```

Out[6]:
```
count                    210072
mean     0 days 00:06:08.994773
std      0 days 00:03:28.346263
min             0 days 00:00:00
```

```
25%             0 days 00:03:08
50%             0 days 00:06:06
75%             0 days 00:09:10
max             0 days 00:12:00
Name: time, dtype: object
```

- Create lagged variable to indicate the result of the previous shot by the same player in the same game.
  1. We will first sort the shot outcome by the quarter and time in the game;
  2. We will group the data by player and game (date) and use the "shift" command to create a lag variable.

In [7]:
```
Shotlog['lag_shot_hit']=Shotlog.sort_values(by=['quarter','time'], ascending=[True, True]).groupby(['shoot_player','date'])['
Shotlog.head()
```

Out[7]:

	team_previous_shot	player_position	home_game	location_x	opponent_previous_shot	home_team	shot_type	points	away_team	location_y	time
0	NaN	SF	Yes	97.0	SCORED	ATL	Pullup Jump Shot	2	WAS	405.0	00:01:09
1	MISSED	C	Yes	52.0	SCORED	ATL	Tip Dunk Shot	2	WAS	250.0	00:01:11
2	SCORED	SG	Yes	239.0	MISSED	ATL	Jump Shot	2	WAS	223.0	00:01:41
3	SCORED	PG	Yes	102.0	SCORED	ATL	Pullup Jump Shot	2	WAS	385.0	00:02:16
4	SCORED	PF	Yes	128.0	MISSED	ATL	Turnaround Jump Shot	2	WAS	265.0	00:02:40

We can sort the shot log data by player, game(date), quarter, and time of the shot.

In [8]:
```
Shotlog.sort_values(by=['shoot_player', 'date', 'quarter', 'time'], ascending=[True, True, True, True])
```

Out[8]:

	team_previous_shot	player_position	home_game	location_x	opponent_previous_shot	home_team	shot_type	points	away_team	location_y
42660	MISSED	C	No	210.0	SCORED	GSW	Jump Shot	2	DAL	269.0
42661	SCORED	C	No	308.0	SCORED	GSW	Jump Shot	3	DAL	202.0
42664	MISSED	C	No	167.0	SCORED	GSW	Jump Shot	2	DAL	318.0
42667	SCORED	C	No	131.0	MISSED	GSW	Jump Shot	2	DAL	337.0
42668	MISSED	C	No	72.0	MISSED	GSW	Tip Layup Shot	2	DAL	248.0
43139	SCORED	C	Yes	882.0	SCORED	DAL	Running Reverse Layup	2	LAC	264.0
43228	MISSED	C	No	84.0	SCORED	CLE	Turnaround	2	DAL	112.0

Notice that for the first shots of the game by the given players, the lagged outcome variable will have missing value.

Let's create a dataframe for average success rate of players over the season.

Since the "current_shot_hit" variable is a dummy variable (=1 if hit, =0 if miss), the average of this variable would indicate the success rate of the player over the season.

In [9]:
```
Player_Stats=Shotlog.groupby(['shoot_player'])['current_shot_hit'].mean()
Player_Stats=Player_Stats.reset_index()
Player_Stats.head()
```

Out[9]:

	shoot_player	current_shot_hit
0	A.J. Hammons	0.404762
1	Aaron Brooks	0.403333
2	Aaron Gordon	0.454861
3	Aaron Harrison	0.000000
4	Adreian Payne	0.425926

- Let's rename the "current_shot_hit" variable in the newly created date frame as "average_hit".

In [10]:
```
Player_Stats.rename(columns={'current_shot_hit':'average_hit'}, inplace=True)
```

We will use the player statistics to analyze the hot hand. So we will merge average player statistics dataframe back to the shot log dataframe.

In [11]:
```
Shotlog=pd.merge(Shotlog, Player_Stats, on=['shoot_player'])
Shotlog.head()
```

Out[11]:

	team_previous_shot	player_position	home_game	location_x	opponent_previous_shot	home_team	shot_type	points	away_team	location_y	time
0	NaN	SF	Yes	97.0	SCORED	ATL	Pullup Jump Shot	2	WAS	405.0	00:01:09
1	MISSED	SF	Yes	279.0	SCORED	ATL	Jump Shot	3	WAS	130.0	00:03:11
2	MISSED	SF	Yes	58.0	SCORED	ATL	Cutting Layup Shot	2	WAS	275.0	00:09:53
3	SCORED	SF	Yes	868.0	SCORED	ATL	Jump Shot	3	WAS	475.0	00:01:02
4	SCORED	SF	Yes	691.0	MISSED	ATL	Pullup Jump Shot	3	WAS	100.0	00:04:50

- Create a variable to indicate the total number of shots recorded in the dataset for each player.

In [12]:
```
Player_Shots=Shotlog.groupby(['shoot_player']).size().reset_index(name='shot_count')
```

In [13]:
```
Player_Shots.sort_values(by=['shot_count'], ascending=[False]).head()
```

Out[13]:

	shoot_player	shot_count

402	Russell Westbrook	1940
25	Andrew Wiggins	1568
106	DeMar DeRozan	1545
193	James Harden	1532
28	Anthony Davis	1525

We should also note that players have different number of shots in each individual game. We will need to treat the data differently for a player who had only two shots in a game compared to those who had attempted 30 in a game.

- Create a variable to indicate the number of shots in each game for by each player.

In [14]:
```
Player_Game=Shotlog.groupby(['shoot_player','date']).size().reset_index(name='shot_per_game')
Player_Game.head()
```

Out[14]:

	shoot_player	date	shot_per_game
0	A.J. Hammons	2016-11-09	5
1	A.J. Hammons	2016-11-23	1
2	A.J. Hammons	2016-11-25	1
3	A.J. Hammons	2016-12-03	2
4	A.J. Hammons	2016-12-07	2

We will merge the shot count data frames back to the shot log dataframe.

In [15]:
```
Shotlog=pd.merge(Shotlog, Player_Shots, on=['shoot_player'])
Shotlog=pd.merge(Shotlog, Player_Game, on=['shoot_player','date'])
display(Shotlog)
```

	team_previous_shot	player_position	home_game	location_x	opponent_previous_shot	home_team	shot_type	points	away_team	location_y
0	NaN	SF	Yes	97.0	SCORED	ATL	Pullup Jump Shot	2	WAS	405.0
1	MISSED	SF	Yes	279.0	SCORED	ATL	Jump Shot	3	WAS	130.0
2	MISSED	SF	Yes	58.0	SCORED	ATL	Cutting Layup Shot	2	WAS	275.0
3	SCORED	SF	Yes	868.0	SCORED	ATL	Jump Shot	3	WAS	475.0
4	SCORED	SF	Yes	691.0	MISSED	ATL	Pullup Jump Shot	3	WAS	100.0
5	MISSED	SF	Yes	691.0	MISSED	ATL	Pullup Jump Shot	2	WAS	181.0
6	MISSED	SF	Yes	679.0	MISSED	ATL	Step Back Jump Shot	3	WAS	109.0

We will sort the data again after merging.

In [16]:
```
Shotlog.sort_values(by=['shoot_player', 'date', 'quarter', 'time'], ascending=[True, True, True, True])
```

Out[16]:

	team_previous_shot	player_position	home_game	location_x	opponent_previous_shot	home_team	shot_type	points	away_team	location_y	...
50484	MISSED	C	No	210.0	SCORED	GSW	Jump Shot	2	DAL	269.0	...
50485	SCORED	C	No	308.0	SCORED	GSW	Jump Shot	3	DAL	202.0	...
50486	MISSED	C	No	167.0	SCORED	GSW	Jump Shot	2	DAL	318.0	...
50487	SCORED	C	No	131.0	MISSED	GSW	Jump Shot	2	DAL	337.0	...
50488	MISSED	C	No	72.0	MISSED	GSW	Tip Layup Shot	2	DAL	248.0	...
50489	SCORED	C	Yes	882.0	SCORED	DAL	Running Reverse Layup	2	LAC	264.0	...
50490	MISSED	C	No	84.0	SCORED	CLE	Turnaround	2	DAL	112.0	...

We will treat the "points" and "quarter" variables as objects.

In [17]:
```
Shotlog['points'] = Shotlog['points'].astype(object)
Shotlog['quarter'] = Shotlog['quarter'].astype(object)
```

Missing values

- Drop observations with missing value in lagged variable.

In [18]:
```
Shotlog=Shotlog[pd.notnull(Shotlog["lag_shot_hit"])]
```

Let's take a quick look at the number of variables and the number of observations in our clean dataframe.

In [19]:
```
Shotlog.shape
```

Out[19]: (185052, 21)

Save our updated data

In [20]:
```
Shotlog.to_csv("../../Data/Week 6/Shotlog1.csv", index=False)
Player_Stats.to_csv("../../Data/Week 6/Player_Stats1.csv", index=False)
Player_Shots.to_csv("../../Data/Week 6/Player_Shots1.csv", index=False)
Player_Game.to_csv("../../Data/Week 6/Player_Game1.csv", index=False)
```

In [ ]: