NBA Play-by-Play Database
601.615 Database
Final Project Phase II

Shiqing Sun: ssun27@jhu.edu
Bohao Tang: bhtang@jhu.edu

December 19, 2019

1 General Introduction

1.1 Application Domain

In this project, we aim to build up a database that saves all the play-by-play data in NBA. As play-by-play data is the most fundamental data in NBA and is the source of all advanced statistics regarding awards and game tactics, we seek to implement the advanced data mining procedure on the database.

1.2 Guide to View Results

We here list all the steps needed to run the database:

1. Create Schema: Run `nba_database.sql` to create schema and tables
2. Load Data: Run `nba_update.py` in Python, remember to change username and password parameter of **"NBA_Update"** object to your database specific. And you can re-run the file on new dataset to update the data.
3. Create Views for statistics: Run `nba_statistics.sql` and `call.sql` to create views for statistics and stored procedures to query them. Then run `shot_chart_data.sql` to create views for necessary data to plot shot chart.
4. Setup for R and MySQL
   - Packages needed in R: shiny,ggplot2,hexbin,dplyr,httr,readr,jsonlite,RMySQL
   - Version for MySQL: Recent Version containing "Rank Over" function
5. Run the App: Just run R command `shiny::runApp()` in the project folder and the website will be created. Then you can select players by name and date range to calculate statistics and plot shot-chart. (There's a issue related to Rstudio preview and please open the preview in browser or you can not see players' photo)

6. Customize the output: Currently, the attributes in the output of statistics for each player cannot be customized in the user interface. The default attributes in the outputs are total scores, total assists, total rebounds, total steal, Hollinger Efficiency and all their ranks. If you want to customize the output, we suggest you change the stored procedure `selectall` defined in `call.sql`. The suggested date range here is around a month when outputs contain uper (Hollinger Efficiency) because of the high computational complexity and therefore latency from the complicated design.

2 Data Preparation

2.1 Raw Data

The raw data of this project the NBA play-by-play detailed log data in Season 2018-2019. To be specific, play-by-play data refer to the records of the happenings on court in each second from each game. In this project, we use python to upload data into local mysql server for further steps. The corresponding code is `nba_update.py` in the attached zip file.

As the happenings on court belong to various categories, including shoots, assists and etc, to clarify the relationships contained in the raw data, we build up an entity relation model with each table in BCNF.

2.2 Data Loading

As discussed in last subsection, we build up an entity relation model to store the data better. For each tuple loaded from the raw data, we decompose it into several tuples based on its category of events. The final decomposition and the corresponding attributes are in the attached file `nba_database.sql`.

From the definition of the tables in the model, it is easy observe that the only dependency in each table is related with its super key. Thus, the final ER model is in BCNF.

3 Specialization

3.1 Data Mining

As is mentioned above, we aim to derive advanced NBA statistics from the fundamental play-by-play data. In this project, we implement more than 500 lines of SQL query for advanced data mining. As a result, we now can compute not only regular statistics, like total points and total rebounds, but also advanced statistics that requires plenty of hierarchical efforts, like Hollinger Efficiency.

Taking Hollinger Efficiency as an example, this statistic combines 12 different measurements of efficiency, each involving multiple queries to compute. In return, the delicate design grants Hollinger Efficiency good prediction power for MVPs. Based our checking, the order of Hollinger Efficiency is highly correlated with both month MVPs and week MVPs.

3.2 Data Visualization and Interface

We also offer a visualization of player's shot chart by hexagonal plot, which will illustrate the density and shooting efficiency of the chosen player compare to whole league. For efficiency, we have choices of 'FG% vs. League Average', 'FG% only' and "Points Per Shot" in the interface. And you can see the output samples in section 5.

The UI is based on a github project BallR at `https://github.com/toddwschneider/ballr`, but we changed the design and output. And we need to mention here, although we use R shiny as the framework because it's easy to deploy and design the UI with advanced plot, all the data procedure is done in the MySQL side and we only connect to database and asking necessary data from it.

4 Pros and Cons

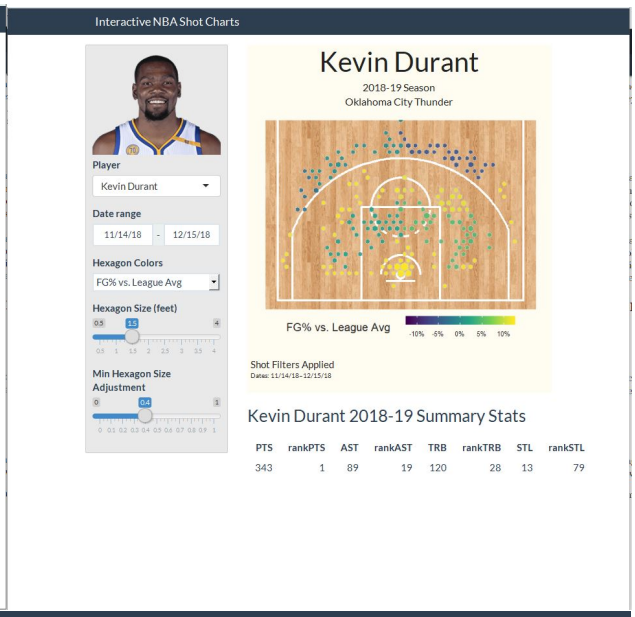
4.1 Pros

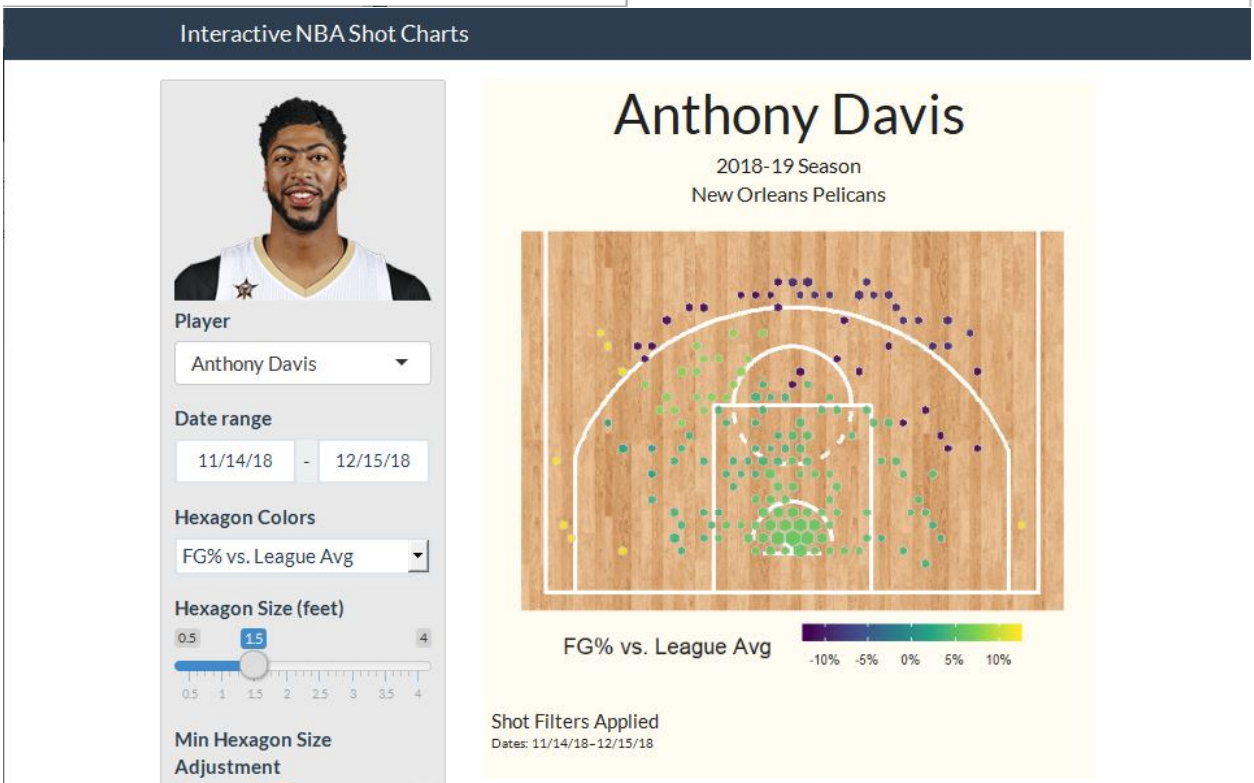
- We have data updating script `'nba_update.py'` and therefore can update data easily.
- We provide hierarchical statistics computed from the very fundamental datasets. The advanced statistics shows good prediction power for MVPs.
- We build the app by R shiny interface adjusting github project [1], which makes plotting and modeling much more easy. Also the UI would be more user friendly.

4.2 Cons

- Since we implement the data mining from fundamental datasets, the computation complexity for the SQL part is huge, which makes the computation very time-consuming.
- More advanced statistics can be invented with more time.
- Now we only have the data from season 2018-2019. More data is required

5 Output Presentation





References

[1] T. Schneider, "ballr," https://github.com/toddwschneider/ballr/, 2019.