Automatic and periodical data extractions for some NBA aggregate statistics

Nikhil Panu (415), Gilbert Maystre (415)

December 20, 2015

1 Overview

In this project, we focused on designing a service that fetches data periodically from NBA.com and use the induced temporality in the data to plot some aggregate statistics. We designed a simple GUI to explore a bit the fetched data.

2 Data URL

TODO: Nik (put where is the source of data). For a more precise text, please refer to the "Fetching the data" section.

3 Running our code

TODO: Nik verify this section

The code was written in Java and Python. The Python part is invoked directly in the Java code, so there is no way to run it on it's own. There are two main classes. First, there is `Bootstrap.java` which represent the server which fetches data periodically. Second, there is `Analyzer.jar` which provides a GUI to explore the data.

The server does not come precompiled, but the Analyzer does. To run it, just double click on `Analyzer.java` or type in `java -jar Analyzer.jar`. The actual database should be put exactly as that:

```
<folder where the jar is>/data/db.sql
```

Those two files comes in the runnable folder of the zip archive.

4 Specialization

4.1 Fetching the data

TODO: Nik

4.2 Designing a GUI to render temporality

Since we fetch the data in a periodical fashion, we have statistic with one more dimension: the time. This is what led us to the idea of showing the statistic by displaying stem plots.

On the x-axis lies the time, and on the y axis lies the statistic. This can show new hidden relations in data such as: "Does this particular player do better on week-end games?".

Sometimes, simplicity is good. We decided to make only one query available but to make it parametrizable via combo boxes (see Figure 1, 2). The advantage of using combo boxes

is that it prevents SQL injections and that it clearly shows the user was the program has to offer.

We coded the GUI from scratch with only `Java.Swing` API.

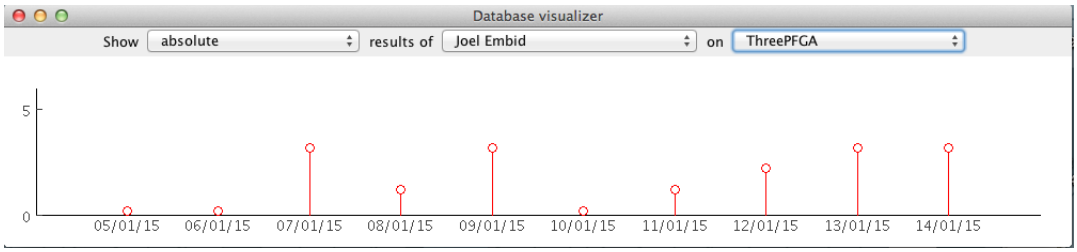

Figure 1: Our GUI with some data

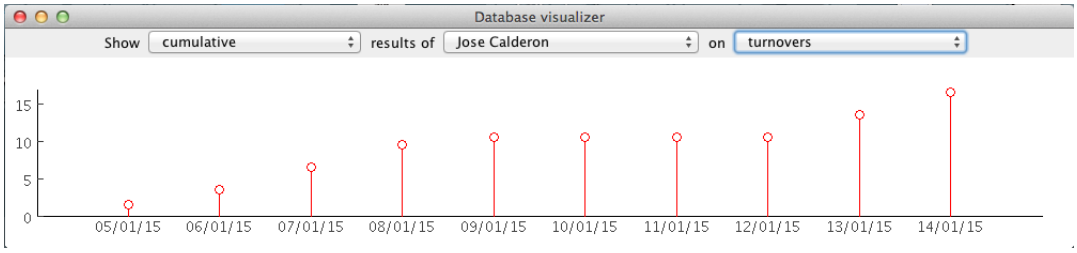

Figure 2: Our GUI with some data and another query.

5 Limitation and further improvements

TODO: fill the data fetching part Niki.

On the GUI part, we could quite easily add more feature and queries. This would not require much time, but the point of this executable was to show that it was possible and how to use the time that we have in the data.

6 Description of the database

TODO: Niki (you can inspire yourself from section Constructing the database)

7 All SQL code

We did not use extensively SQL since the goal of our project was mainly to fetch data smartly, but those are the queries our program uses.

7.1 Constructing the database

Those queries come from `SQLStorer.prepareCuteDatabase()`.

```
CREATE TABLE IF NOT EXISTS Player(FName TEXT, LName TEXT, Age Integer,
   Height DECIMAL(5, 2), Weight DECIMAL(5, 2), Years_Pro Integer,
   Position TEXT, PRIMARY KEY (FName, LName))

CREATE TABLE IF NOT EXISTS Statistics(FName TEXT, LName TEXT,
   TimeStep Integer, Games_Played Integer, Points Integer, Rebounds Integer,
   Assists Integer, Steals Integer, Blocks Integer, Turnovers Integer, FGM Integer,
   ThreePFGM Integer, FGA Integer, ThreePFGA Integer, FTM Integer, FTA Integer,
   PRIMARY KEY(FName, LName, TimeStep))

CREATE TABLE IF NOT EXISTS Team(TeamName TEXT, Location TEXT, Conference TEXT,
   Division TEXT, Wins Integer, Losses Integer, PRIMARY KEY(TeamName))

CREATE TABLE IF NOT EXISTS PlaysFor(FName TEXT, LName TEXT, TeamName TEXT,
   PRIMARY KEY(FName, LName, TeamName))

CREATE TABLE IF NOT EXISTS Date(TimeStep Integer, Date TEXT, PRIMARY KEY(TimeStep))
```

7.2 Inserting fetched data

(Those queries come from various methods in `SQLStorer.java`. All the parameters starting with ":" are linked after some kind of processing to avoid SQL injection.

```
INSERT INTO Player Values(:FName, :LName, :Age, :Height, :Weight, :Years_Pro,
   :Position)

INSERT INTO Team Values(:TeamName, :Location, :Conference, :Division, :Wins,
   :Losses)

INSERT INTO PlaysFor Values(:FName, :LName, :TeamName)

INSERT INTO Statistics Values(:FName , :LName, :TimeStep, :Games_Played,
   :Points, :Rebounds, :Assists, :Steals, :Blocks, :Turnovers, :FGM, :ThreePFGM,
   :FGA, :ThreePFGA, :FTM, :FTA)

INSERT INTO Date Values(:TimeStep , :Date)
```

```
SELECT max(timestep) as u FROM Statistics

SELECT * FROM Player
```

7.3 Fetching data

Those queries come from various methods in `WrapperUtility.java`.

```
SELECT P.FName, P.LName FROM Player as P

SELECT D.Date, S.<attribute>
FROM Statistics as S, Date as D
WHERE S.FName = :FName AND S.LName = :LName AND D.TimeStep = S.TimeStep

SELECT max(S."+attribute+") as m FROM Statistics as S
```