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 building a server that will fetch data smartly from NBA.com and use the induced temporality in the data to plot some aggregate statistics.

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

We wrote our code in java and python. The java parts call's the python code implicitly, so no need to do anything. The java parts has two main. The first, in class Bootstrap.java is the server side of our app. It launches the services that fills the database. The second main is to be found in Analyzer.java, it's job is to query the database and plot stem graphs.
Analyzer.java comes precompiled as a jar file you can simply double click on it or type in `java -jar Analyzer.jar`. The actual database should be put exactly as:

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

Those two files comes in the runnable folder of the 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 can get nice stem plots. On the x-axis lies the time, and on the y axis lies the aggregate 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 problems and that it clearly shows the user was the program has to offer.

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

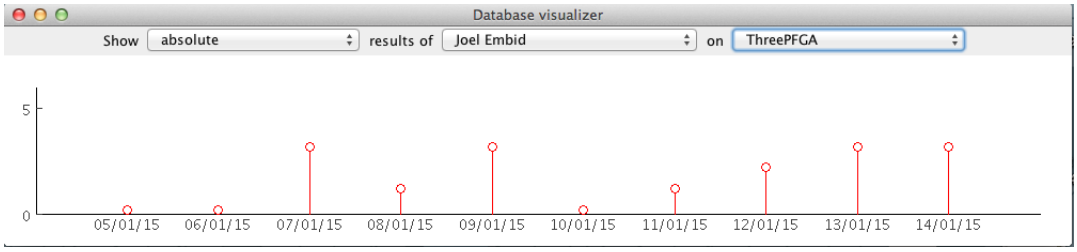

Figure 1: Our GUI with some data

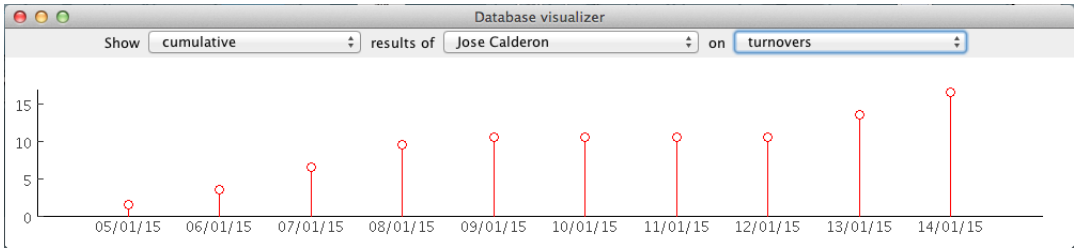

Figure 2: Our GUI with some data and another query.

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.

6 Description of the database

TODO: Niki

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

(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

(From various methods in `SQLStorer.java`.) All the parameters starting with ":" are binded 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 the database

(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
```