Data 3402: Project 1
Gathering Raw Data

Unlike Data 3401, where projects were disjointed and unrelated, in this course, we will work a series of projects that build on one another, so that by the end of the semester, you will have built an engine that will extract raw data from multiple sources (freely available on the internet), process (parse, clean, transform, merge, etc) that raw data, and store it in a database for future analysis. In this first project, you will use bash scripting to download, organize, and store raw data.

1 The Data and Your Task

Baseball, more than any other sport, has become a business that is driven by data. There are vast quantities of data freely available through web, so much, in fact, that entire games can be constructed down to the path, velocity, and rotation of each pitch. While we will not be dealing with the data to this minute level, we will be gathering data from three major sources.

1. MLB.com
2. Retrosheets.org
3. The Lahman Database

Your task will be to write a multithreaded python script that gathers data from each of these sources (specified in the following sections) and stores the data according to the scheme that will be laid out.
Your script should take two arguments:

1. a number of threads, and
2. a directory to store the data.

2 MLB.com

We will be particularly interested in the MLB gameday data that is found at **https://gd2.mlb.com/components/game/mlb/**. You can navigate the links, but what you need to know is that game data for a specific date MM/DD/YYYY can be found at

https://gd2.mlb.com/components/game/mlb/year_YYYY/month_MM/day_DD

Notice that I do not include a '/' at the end of that URL. Doing so will result in an error.

On this page, you will see a number of links with names of the form **gid_YYYY_MM_DD_***. Each of these is a directory that stores data for the specified game. For example, **gid_2018_07_01_anamlb_balmlb_1/** contains data for the game played between the Anaheim Angles and the Baltimore Orioles on 07/01/2018. The '1' at the end indicates that this was the first game played between the two on that date. A '2' would indicate that the game was the second game in a double-header. The names of these directories are the "Game IDs" (GID).

Inside this directory, there are three files that you should capture.

1. **players.xml**: contains information
2. **inning_all.xml**: contains the result of every event (pitch, pickoff attempt, steal attempt, etc.) that occurred during the game.
3. **inning_hit.xml**: contains information about every hit (including location where the ball landed or first touched a fielder).

Your script should do the following:

1. Create a directory called MLB
2. Inside **MLB**, create a directory for each game played between 01/01/2010 and the current date. The name of each game directory should be its GID.
3. In each game directory, you store the three files **players.xml**, **inning_all.xml**, and **inning_hit.xml**.

These tasks should be accomplished using a pool of threads. The pool should

1. take a list of dates (or date objects)
2. extract the GIDs for that date and add them to a queue
3. read the games from the queue and download the game data files.

I know this one was a bit complicated to explain, but don't worry, the other two sources should be a bit straightforward.

3 Retrosheet.org

According to the Society for American Baseball Research (SABR), this website contains data that was compiled by ”a small army of volunteers, combing historical sources to try to re-create the play-by-play of every game in baseball history and digitizing it for download and analysis.” While there are a lot of compiled statistics on the site, we’re primarily interested in the play-by-play event data (as we were with the MLB.com data).

If you navigate to the page **https://www.retrosheet.org/game.htm**, you will find the following.

1. A link to a listing of players, coaches, managers, and umpires (past and present). This data will be needed for decoding player IDs in the data.
2. Links for regular season event files. For each year, there is a link to a zipped file containing three types of data.
   (a) Event data for each MLB team for that year (.eva and .evn extensions)
   (b) Roster data for each MLB team for that year (.ros extension)
   (c) A file named TeamXXXX (where XXXX is the year) that contains a list of MLB teams in year XXXX. This data will be needed for decoding team IDs in the data (among other things).
3. Links for post-season event file. Again, for each year there is a link to a zipped file containing the same three types of data. You will need to keep the event files, but the team and roster data will have already been acquired from the regular season files.

Your script should do the following.

1. Create a directory called **Retrosheets**.
2. Inside **Retrosheets**,
   (a) download the list of players, coaches, managers, and umpires to a file called **player.csv** and
   (b) create three directories called **Events**, **Teams**, and **Rosters**.
3. For each year between 1900 and 2019, download and unzip the regular season event data and distribute the event, roster, and team files to the appropriate directories.
4. For each year between 1900 and 2019, download and unzip the post-season event data and copy the event files to the appropriate directory. You do not need to keep the Team and Roster data since that will already have been acquired from the regular season download.

4 The Lahman Database

The Lahman Database contains a large number of statistics for every MLB player aggregated by year. You won't be able to determine if Jose Altuve got a hit in the 6^{th} inning of the Astros' 2018 season opener, but you will be able to find out his batting average, RBI count, etc. for the 2018 season.

Your script should do the following:

1. Create a directory called **Lahman**.
2. Download the 2017 Lahman database in CSV format from http://www.seanlahman.com/baseball-archive/statistics.
3. Unzip the archive into **Lahman**

5 Miscellanea

And that's the project. Your script should be able to be run as a cron job once daily to pick up any newly released data. It should not just re-download all of the data every time it runs. It should check each date, determine if any data is missing, and only download what is needed.

Upload your completed scripts to blackboard by 11:59pm Thursday, 10/10/19.