Meta Musical Memes

Fall 2015 W205 Project Report

Megan Jasek, James King, Sean Underwood

Table of Contents

Overview

For several decades (at least), music fans have been referring to many kinds of pop music -- particularly the extremely popular songs -- as “formulaic.” This is usually meant to be a derogatory term, but it causes those of a certain disposition (Berkeley Data Science students) to wonder if we can actually articulate the “formula” or at least give a corpus of data to songwriters and music producers which helps assess the commercial potential for some work.

Scope

The scope of this project was to create a model of a common big data architecture and demonstrate its usefulness by creating a prototype. The general architecture consists of ingesting data, processing it, storing it for analysis and analyzing it. The prototype illustrates the following:

- Ingest data. Ingest musical data from 3 sources listed below (Python).
- Load data. Load data to a temporary location for processing (Amazon S3).
- Transform data. Clean and transform data into a useable format (Hive).
- Store data for processing. Load merged data into a relational database (MySQL).
- Process data: Analyze data to accomplish business objective (RStudio Server, Shiny Server).

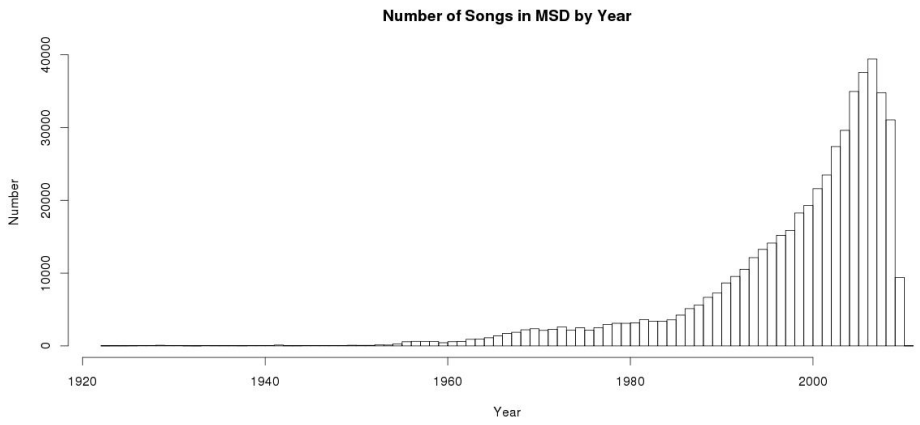
The Million Song Dataset (MSD) is a body of data created by LabROSA at Columbia University, and provided to the public via a public Amazon Public Dataset. It primarily consists of the data that The Echo Nest publishes for 1,000,000 songs. This data was processed, cleaned, and joined with Billboard chart data and authorship data from the American Society of Composers, Authors and Publishers (ASCAP), then analyzed in various ways which would be interesting to the general population of music fans as well as music industry professionals. The following report describes details of this data preparation and analysis.

Data Sources

This project used three different sources of input data, each providing extra information of interest to the musically-minded. These were the Million Song Dataset (MSD), historical Billboard charts, and the writers credit and ownership database by the American Society of Composers, Authors and Publishers.

MSD

The million song dataset (MSD) is a collection of metadata about popular songs in the years 1922 to 2011. It is provided as an Amazon Public Dataset ,is approximately 500GB in size, and is provided in the HDF5 data format.



Information included in this data set include things like author and title, but more interestingly also include data fields related to the content of the song (example here). This content metadata was measured automatically by software created by the Echo Nest and LabROSA at Columbia University. Of particular interest for this project were the music-theoretical components of the song such as key and tempo which were used in the final regression analysis.

Billboard Charts

Billboard has been publishing lists relating to the popularity of songs since the late 19th century and their classic “Billboard Top 40” has been the rating of choice for generations.

For this project, a data set of all the chart-listed songs (and artists) in the years 1890-2011 were collected from the Whitburn project. This data contains information such as song, artist, year and peak position on the charts and was merged with the MSD in the processing layer, as described below.

1	Year	Yearly F	Sou	Prefix	CH	40	10	PK	High	Ver	Artist	Inverted	Featured	U	Album	B-Side	Track	Time	Source	xplic	Time	Artist ID
400	2010	409	h	2010_050	3	1	1	1	8		Taylor Swift	Swift, Taylor					Speak Now	4:01				
401	2010	304	h	2010_051	29	24	7	5	9		Flo Rida	Flo Rida	Featuring	I	s		Club Can't Handle Me	3:52				
402	2010	189	h	2010_052	18	10	2	2	9		Black Eyed Peas, The	Black Eyed Peas, T			The E.N.D. (The		Rock That Body	4:28				
403	2010	122	h	2010_053	28	19	7	1	9		Jason DeRulo	DeRulo, Jason			Jason Derulo		Ridin' Solo	3:35				
404	2010	20	h	2010_054	29	24	2	1	9		Trey Songz	Songz, Trey			Ready (Deluxe Vi		Say Aah	3:27				
405	2010	201	h	2010_055	18	13	1	1	9		3OH!3	3OH!3	Featuring	I	s		My First Kiss	3:12				
406	2010	44	h	2010_056	30	21	1	1	10		Adam Lambert	Lambert, Adam			For Your Entertai		Whataya Want From Me	3:47				
407	2010	17	h	2010_057	20	18	1	1	10		Jay Sean	Sean, J	Featuring	I	All Or Nothing (B		Do You Remember	3:31				
408	2010	136	h	2010_058	20	16	1	1	10		Jay-Z	Jay-Z	Featuring	I	The Blueprint 3		Young Forever	4:13				
409	2010	193	h	2010_059	20	13	1	1	10		B.o.B.	B.o.B	Featuring	I	The Adventures		Magic	3:16				
410	2010	73	h	2010_060	26	20	0	3	11		Timbaland	Timbala	Featuring	.	Shock Value II		Carry Out	3:52				
411	2010	113	h	2010_061	20	14	0	2	11		Ludacris	Ludacri	Featuring	I	Battle Of The Se		My Chick Bad	3:36				

ASCAP

The American Society of Composers, Authors and Publishers (ASCAP) is one of the major licensing organizations for people in the music industry. They keep track of who owns songs and compensates them when those songs are used in various ways.
In order to accomplish their goals, they keep a reasonably large database with information on every song that’s ever been registered. If you ask nicely, they’ll send it; it’s about 250MB in size compressed, and opens up to 1.56GB.

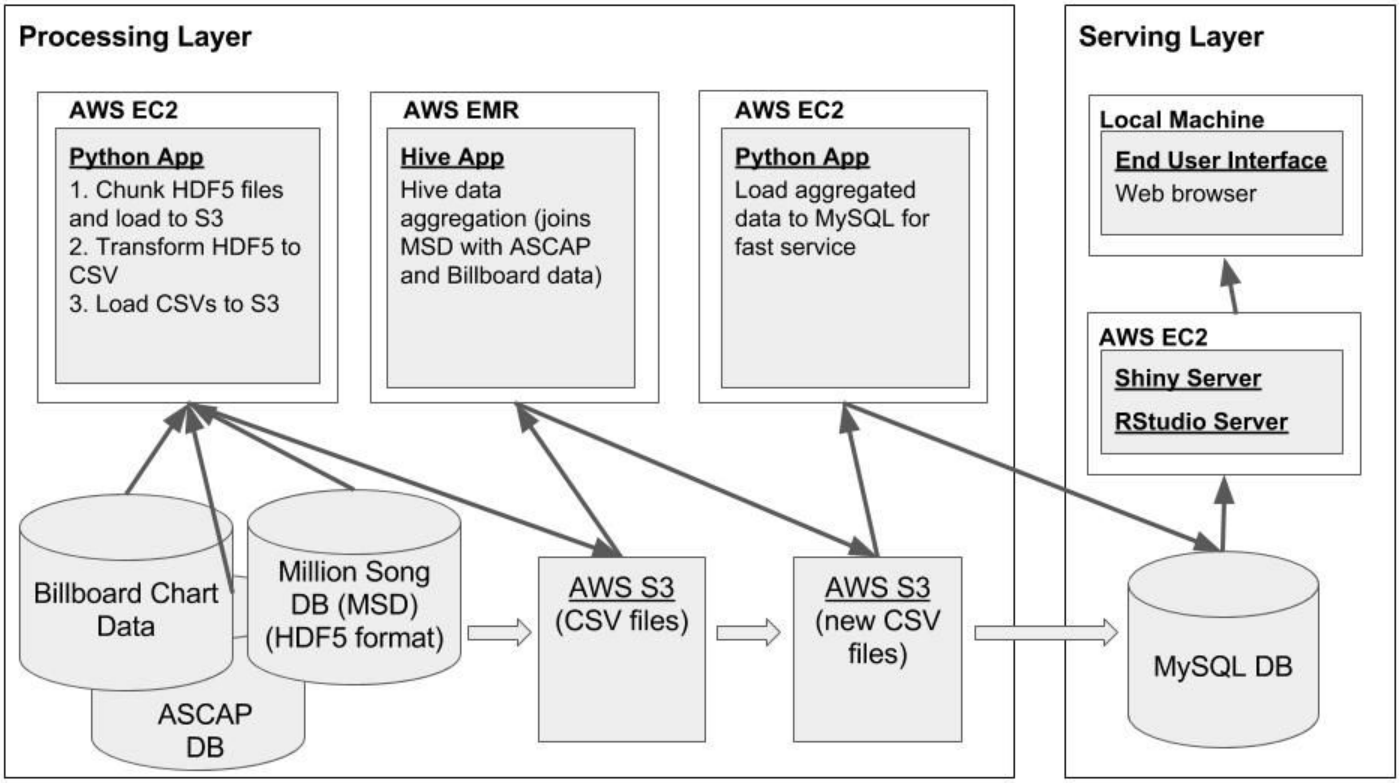
The intent of using this data was to allow the merging of writers with the MSD, because it’s likely that writers have a strong effect on a song’s success. Unfortunately, the data set they provided doesn’t provide enough information to deal with the fact that there are often MANY songs with the same name. Thus, we ended up not using this data set. It remains available, however, in case another use for it becomes clear.

Title	RoleType	Name	Shares	Note
"PATHETIQUE" SONATA OP. 13 (ADAGIO CANTABILE)	ASCAP	Total Controlled by ASCAP	100	
"PATHETIQUE" SONATA OP. 13 (ADAGIO CANTABILE)	W	ROMM RONALD		
"PATHETIQUE" SONATA OP. 13 (ADAGIO CANTABILE)	W	VAN BEETHOVEN LUDWIG		
"PATHETIQUE" SONATA OP. 13 (ADAGIO CANTABILE)	P	CANADIAN BRASS PUBLICATIONS		
"PATHETIQUE" SONATA OP. 13 (ADAGIO CANTABILE)	P	UNIVERSAL MUSIC- MGB SONGS		

Architecture

Meta Musical Memes (MMM) uses a Netflix-like architecture in order to keep costs down. Specifically, AWS S3 is used to store the data instead of something like HDFS. S3 storage is cheaper and easier to access from a variety of sources. It may be slightly slower, but it is worth the small slowdown in performance for a lower cost. An AWS EC2 instance is used as the server to drive the data acquisition, conversion and analysis (see architecture diagram). Data flows through the system as follows.

1. MSD data is sourced on AWS in an EBS snapshot. The data is stored in HDF5 format. This data needs to be converted to a format that is more easily manipulated. A python program running from the AWS EC2 instance converts the data from HDF5 format to a CSV-like format. The data is then batched up into groups of songs and stored in AWS S3 buckets. Data from other data sources is also then loaded into S3 buckets.
2. Once the data is in AWS S3, a Hive job filters out only the data and joins the different data sets to get it ready for end-user analysis and then load it into a relational database (MySQL) for further analysis.
3. The MySQL database was chosen because the team members were familiar with using it. Upon loading into the relational database, the data is transformed into a relational schema that facilitates analysis.
4. At this point, the data is accessible from both an RStudio server and a Shiny Server which allows users to perform analysis and gives them some interactive examples (respectively).



Processing Layer

In order to get this data into a form usable for analysis, a very large amount of processing was necessary. Each type of data required its own processing stream followed by a mechanism for merging the data in preparation for analysis.

Million Song Dataset ETL

Performing ETL on the Million Song Dataset (MSD) posed a couple of interesting problems. First, the data set was relatively large at 500GB (too big to move around easily). The data is provided as an EBS snapshot, so all we had to do is attach an EC2 instance to it to get access. Second, the data is in HDF5 format (not to be confused with the Hadoop file system, HDFS).

The HDF5 format is hierarchical, like XML, but is binary instead of human-readable. This makes it compact and performant, but difficult to explore. Reliably extracting data is also problematic because there is not a guarantee that all songs will have all the same fields. Thus, the extraction first read the data while keeping track of the fields and ensured that the output CSV file had all *possible* fields before writing the output.

More importantly, however, the HDF5 format cannot be split into pieces arbitrarily and is thus not compatible with HDFS. This made it impossible to process with Hadoop applications that rely on the MapReduce framework, and less convenient to process with Spark. This necessitated using old-fashioned approaches like "letting the code run for a long time" (LCRLT).

We used the h5py library, which is provided by the HDF Group, to process the files. To improve speed, we were able to scale up by running it on an xLarge AWS instance with 8 VCPUs and run the code in parallel using python's `multiprocessing` library. Even with the up-scaling, it still took about 12 hours to extract all the data into a CSV format. This file was stored into the project's S3 bucket so it wouldn't have to be repeated.

Billboard & ASCAP ETL

The Billboard data is a 30MB Microsoft Excel spreadsheet. The file contains about 35 columns. This project only needed to use 4 of these columns: song, artist, peak position and peak year. 'Peak position' is the highest position that the song reached on the Billboard chart and 'peak year' is the year in which the song reached that point. These 4 attributes were extracted to a CSV file using python. The file is about 2MB. After conversion, the Billboard data only needed one tweak before it could be joined with the rest of the data sets: It stored band names such as "The Beatles" as "Beatles, The". We used a regular expression in Hive to convert from the latter format to the former one.

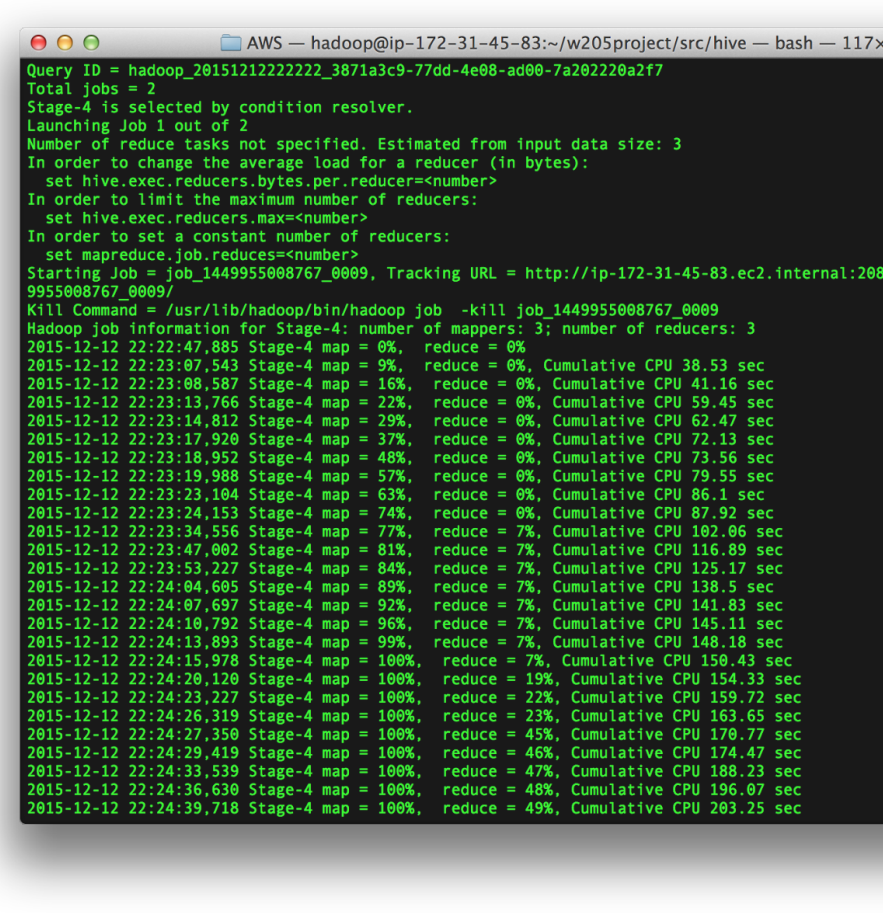
The ASCAP catalog was provided as a 1.56GB CSV file that followed a long format. Each row in the CSV describes a relationship between an individual and a song. The principal metadata we were concerned with for each relationship was the song title, the kind of relationship (e.g., writer, performer), and the person. People's names were stored in an unusual "last middle first" format with no comma.

For processing, we staged this data using an external Hive table, and then transformed it into a secondary table that only stored song and artist name. In the course of this transformation we excluded all rows that

described a relationship other than ‘writer’. We also used a regular expression to move surnames to the end of the name string in order to bring it into line with the way names are represented in the other data sets.

Data Merge

All data were normalized to CSV format and uploaded to Amazon S3. We then ran a Hive job in Elastic MapReduce (EMR) to clean and merge the data sets. Output was again placed in S3, in CSV format. Total time to process on a cluster of three m3.xlarge instances was approximately 5 minutes.



The Hive job performed the transformations to the data described above. The Million Song Dataset coded not available values as 0, so it also coalesced those values to 0 for applicable fields, and it replaced the numeric encodings for key and mode with textual representations.

We were not able to identify a satisfactory way to merge the ASCAP data with the rest of the dataset. It provided information about song titles, writers and publishers, but not the artists who recorded or the year in which they were released. There was no corresponding data in the other two data sets that we could use to establish a link between the data sets. We attempted to join on song title alone, but the resulting data set contained nearly 70,000,000 rows. This extreme level of data duplication threatened the validity of any downstream processing.

We ultimately opted to produce two data sets: One containing only MSD and Billboard data, and another containing the full set. Both are available for SQL access, but we only used the MSD/Billboard data to generate visualizations because of its smaller size and greater reliability.

Serving Layer

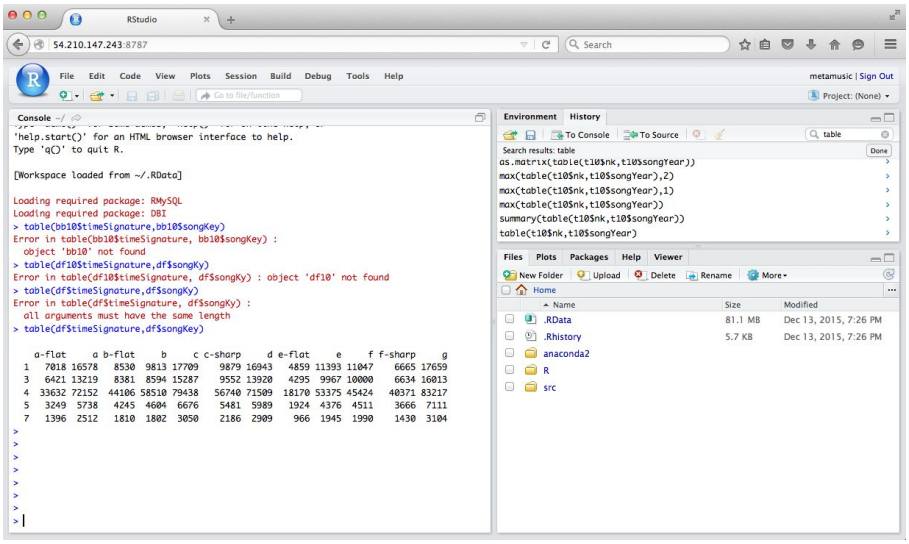
We loaded the output of the Hive analysis into a MySQL database for serving purposes. The database contains a single flat table with a few indexes to allow for better performance in serving some of the visualizations that are more selective about the data they show. We did not attempt to create indexes to support visualizations such as the regression analyzer. They take data from a large percentage of rows, so the potential performance benefit of using an index would have been modest at best.

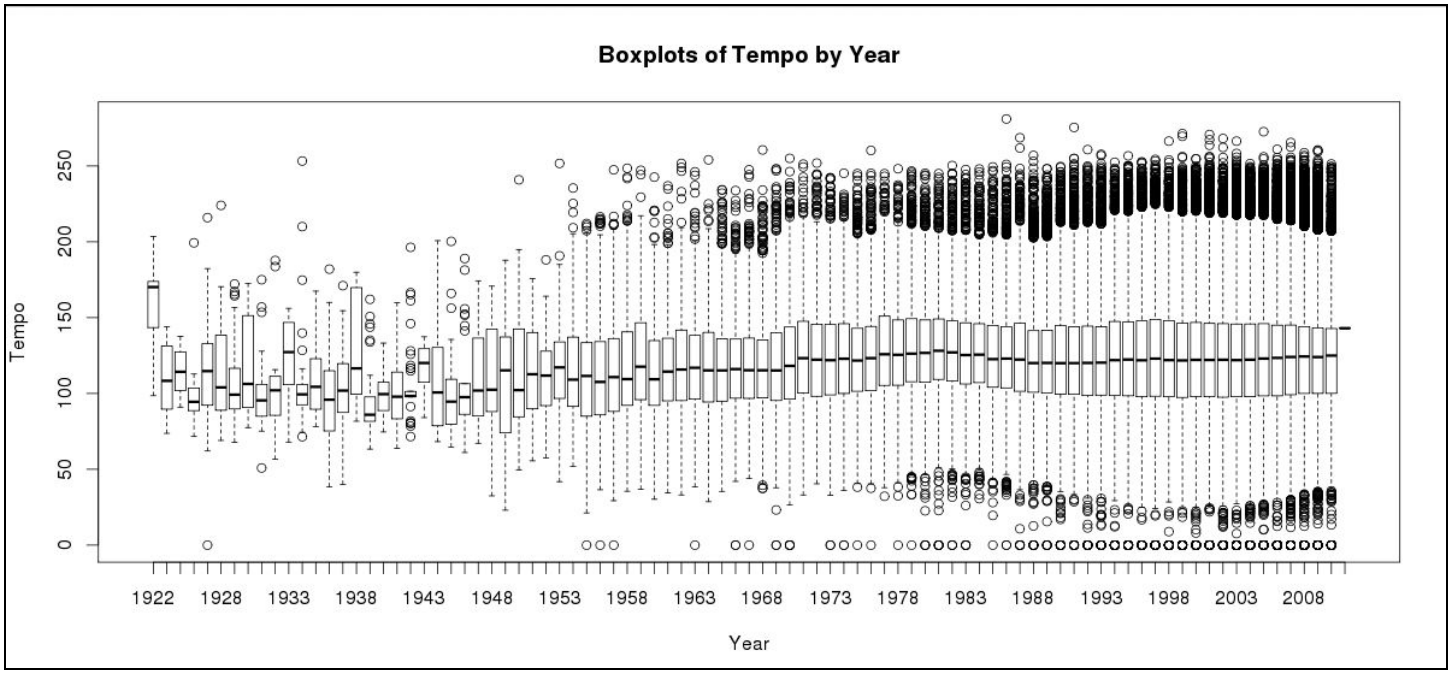
Interactive Output

There are many alternatives for serving the music data after it has been loaded and processed. The selection of which solution to use depends on the requirements of the user. The users of this service will require something interactive, so that they can explore the data. Two services were ultimately provided with two different use cases in mind: RStudio Server and Shiny Server.

RStudio Server

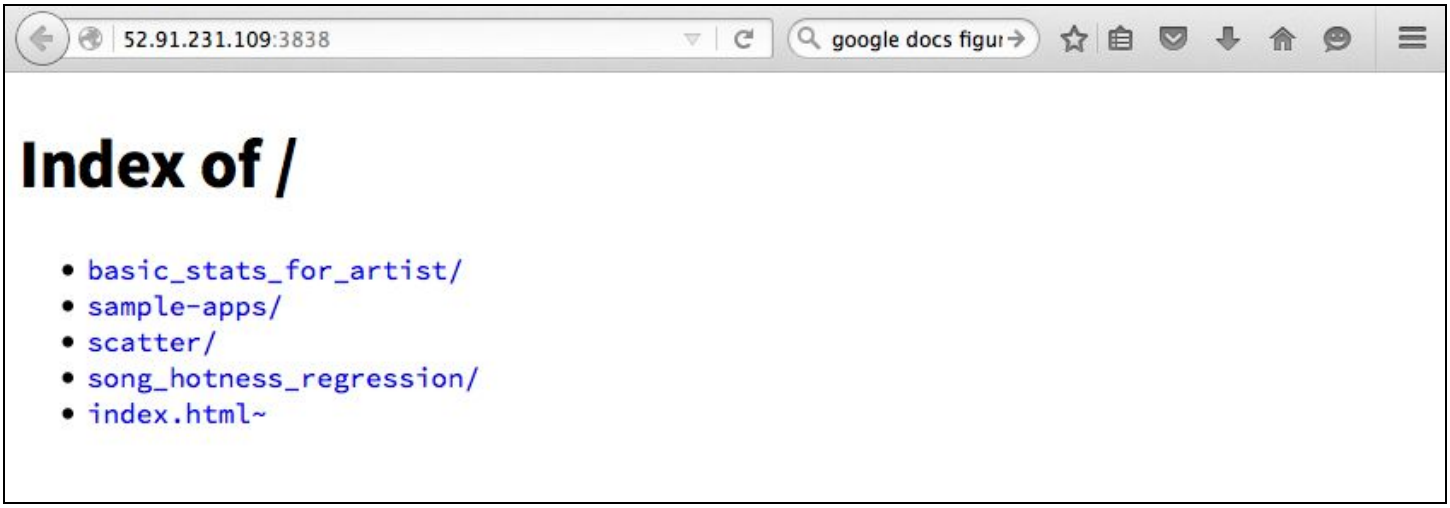
RStudio server provides an identical environment to RStudio (desktop) within a browser window. This allows data-savvy users to directly interact with the results data in a standard environment giving users the full power of R. Below are a selection of visualizations created with R.





Shiny Server

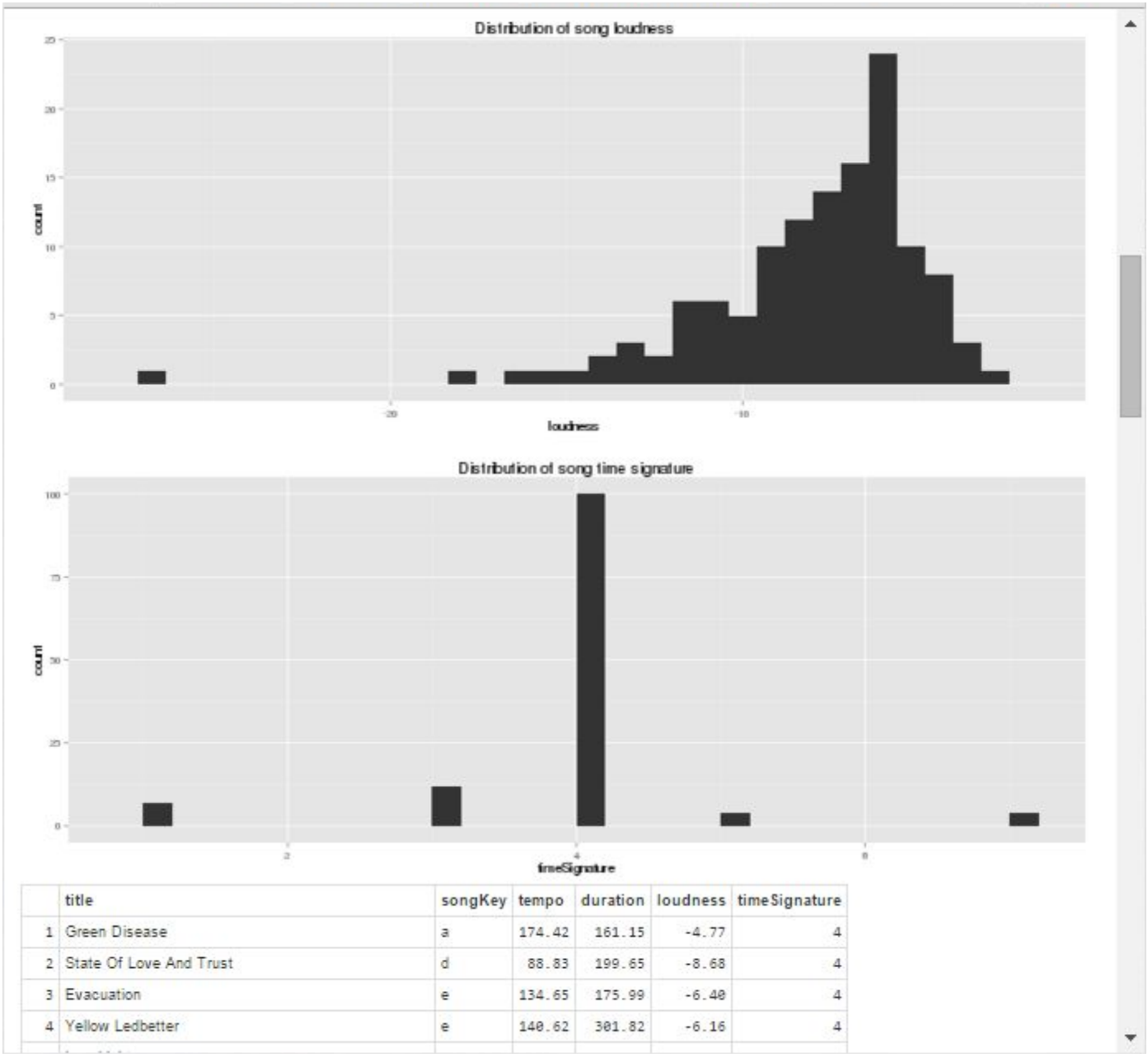
Shiny is a toolkit for writing web applications in R. It offers a simple reactive development model built on existing R idioms that makes it easy to develop interactive data analysis applications. We found this to be valuable because it gave us access to the full R ecosystem for our data analysis and visualization toolkit, and because all project members are already familiar with R. We decided to use the free Open Source edition, which is well-suited to our needs. In this environment, Shiny server is deployed on an EC2 instance and serves the applications on port 3838. Below is a screenshot of the landing page for this project. Each bullet point shows an application that can be run (sample-apps and index.html~ are for testing purposes only). Below is a description of these applications.



Analysis

This project contains three Shiny applications written for analysis: Basic Statistics for Artist (basic_stats_for_artist/ in the diagram above), Song Hotness Regression (song_hotness_regression/ in the diagram above) and Scatter (scatter/ in the diagram above).

Basic Statistics for Artist

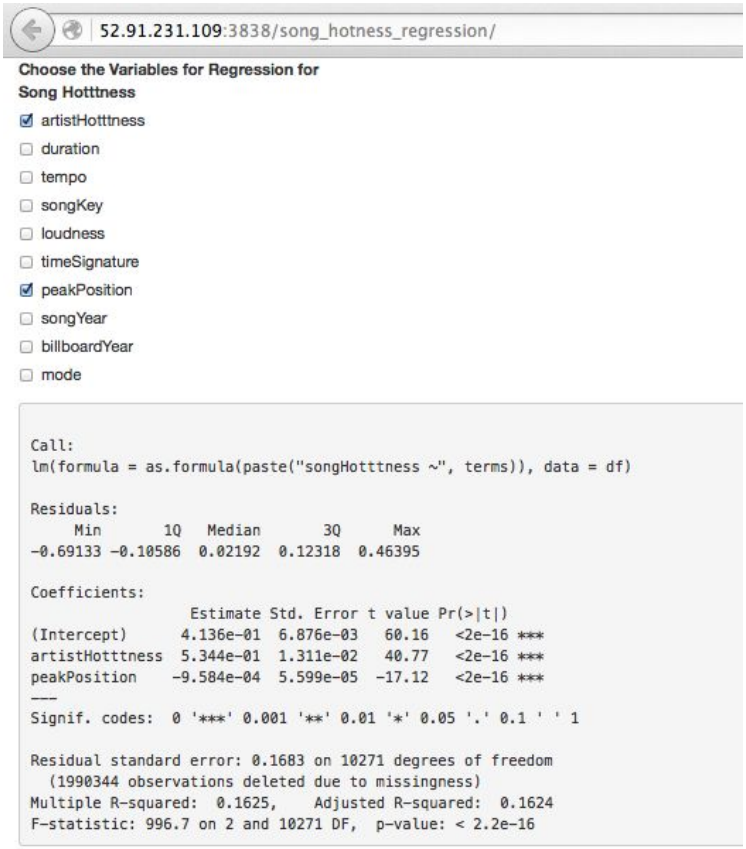


	title	songKey	tempo	duration	loudness	timeSignature
1	Green Disease	a	174.42	161.15	-4.77	4
2	State Of Love And Trust	d	88.83	199.65	-8.68	4
3	Evacuation	e	134.65	175.99	-6.40	4
4	Yellow Ledbetter	e	140.62	301.82	-6.16	4

This application allows the user to enter an artist's name, and it will yield information about the distribution of musical characteristics based on the songs available in the dataset. The application displays 5 histograms: distribution of song keys, distribution of song tempo, distribution of song duration, distribution of song loudness and distribution of time signature. It also displays a table of attributes for each song that the artist has written. Above is a partial screenshot of the output.

Song Hotness Regression

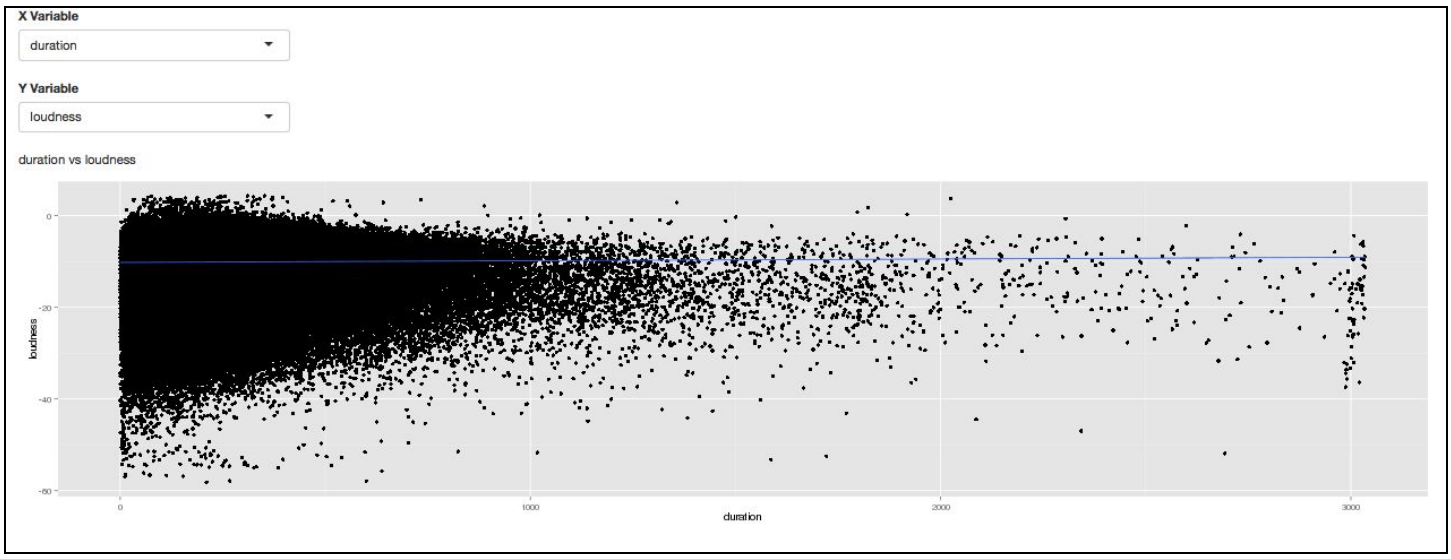
In this application the user is shown checkboxes for the numerical and ordinal song attributes in the dataset: Artist Hotness, Duration, Tempo, Song Key, Loudness, Time Signature, Peak Position, Song Year, Billboard Year and Mode. The user can select one, some or all of the attributes and the application will run a linear regression in R to predict the variable Song Hotness. A text summary of the linear model that was created is then shown to the user.



By running different combinations of variables, the user can see which predictor variables best predict Song Hottness. In our analysis, the attribute that best predict Song Hotness is Artist Hotness. With such a big sample size, many of the attributes show low p-values indicating that they significantly predict Song Hotness, so it is difficult to understand which variables make a difference. However, when you remove Artist Hotness from the analysis the R^2 values are impacted the most.

Scatterplot

This application allows users to experiment with scatterplots based on the dataset. The user is given two dropdown menus from which they can choose any of the non-categorical variables in the data set: Song Year, Peak Position, Billboard Year, Artist Hotness, Song Hotness, Tempo, Duration, Loudness, Time Signature. The application will then generate a scatterplot of those two variables and fit a regression line to it.



Architecture - Alternatives

Processing Layer

Options for processing HDF5 files were limited. It is possible that Spark could have been used to accelerate the processing of these files, but in the absence of a distributed file system that interacts well with HDF5 files it would not help significantly with scale-out. That leaves its potential benefit in doubt.

Using Hive to process the transformed data worked well and demonstrated the utility of a Netflix-style architecture. Processing was quite performant since the EMR cluster could store intermediate data on the instances ephemeral storage, which was backed by solid state drives. Placing work product in S3 buckets had the added convenience of making the data easy to access, including being able to download or view it with a Web browser.

Since we were working with static data, we had no need for the real-time processing capabilities introduced by the Lambda and Kappa architectures.

Serving Layer

There are many alternatives for the serving layer for both storage and processing. For storage, any relational database system could have been used that works with R (or the chosen statistical analysis package) like Postgres. R did show some scalability issues given the volume of data, though. User interface performance could have been improved without significantly harming the accuracy of the data visualizations if the data had been stored in a sampling database such as BlinkDB. Once the data is stored in a relational database, there are many choices for analysis.

For processing, we chose to use RStudio Server and Shiny Server. We evaluated several other options as well. Tableau was considered to make charts and graphs regarding the top songs of all time or the top songs of a decade. This could be added later, but we decided against using it due to licensing restrictions. We also considered writing an interactive layer in Python with visualizations in D3, but decided against it based on our estimates of the level of effort involved and lack of expertise among the team members.

Scaling Out

Currently, this project uses static data, so the only considerations for scaling out are in the serving layer. The project could get the commercial version of the Shiny Server which offers enhanced security and performance features, but it would probably be better to save an image with everything installed, and let AWS load balance automatically based on traffic.

To enable more real-time data for analysis, the application could connect directly to the Echo Nest archive (the source of most of the MSD information) and allow the application to update with the latest information. In particular, new songs would be available as they are analyzed and time-dependent variables like

“artistHotttness” and “songHotttness” could be updated in near real-time. The major constraint against doing this is the proprietary nature of obtaining up-to-date Billboard data which could be costly to license.

Additional work would also have to be done with regard to file format. While the MSD was provided in HDF5 file format, most likely their API would provide something more standard like JSON. This would be easier to manage.

Additionally, the code will be available, so adventurous users can set up their own instances and play with it.

Conclusion

This project was a successful demonstration of modeling a big data architecture and building a prototype to implement it. During this process the following data-related concepts were uncovered and can be used as working knowledge in future projects:

- When incoming data is not record-oriented (like HDF5) using common tools can be difficult. A thoughtful design is required to deal with this type of data and should be tested on a small scale before being implemented.
- When working with many small files like the ones generated from HDF5, design time is required to engineer a solution that manages the data and scales out.
- More data is not always better. The full dataset threatened to overwhelm our selected analysis tool: R. It also caused many statistical tests to show significance. Particularly for the interactive data exploration portion of the project, having some way to limit the volume of data being analyzed without significantly altering its statistical properties would improve the utility of the solution.
- Hive does not always output easily-readable debugging information, but it is effective in speeding up large data transformations.
- R is a powerful statistical analysis tool even where there is a large volume of data.
- Shiny is an effective way and easy-to-learn tool to showcase data analyses.

Acknowledgements

In using the million song dataset, we acknowledge the following paper [pdf] [bib]:

```
Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In Proceedings of
the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), 2011.
```

References

https://www.hdfgroup.org/pubs/papers/Big_HDF_FAQs.pdf
http://top40charts.net/
https://www.ee.columbia.edu/~dpwe/pubs/Ellis07-timbrechroma.pdf
http://labrosa.ee.columbia.edu/millionsong/faq
http://the.echonest.com/

Appendix A: S3 Links

ASCAP Dataset:
https://s3.amazonaws.com/w205-mmm/ascap/ASCAP_CATALOG.csv

Billboard Dataset (Original):
https://s3.amazonaws.com/w205-mmm/billboard_xls/charts.rar

Billboard Dataset (CSV):
https://s3.amazonaws.com/w205-mmm/billboard/billboard.txt

Source Million Song Dataset files, batched into 1,000 objects containing 1,000 songs each:
https://s3.amazonaws.com/w205-msd/

Processed Million Song Dataset extract (CSV):
https://s3.amazonaws.com/w205-mmm/million_song_dataset/MSD_Flat.csv

MSD/Billboard Merged Data:
https://s3.amazonaws.com/w205-mmm/merged/f6c4ee9e-5fb3-4088-96fd-286a830bf530-000000
https://s3.amazonaws.com/w205-mmm/merged/f6c4ee9e-5fb3-4088-96fd-286a830bf530-000001

MSD/Billboard/ASCAP Merged Data (39 files, CSV format):
https://s3.amazonaws.com/w205-mmm/writers/

Appendix B: GitHub Link

https://github.com/SeanU/w205project