# a6-Ritvika-Xia Review - High Fidelity

*balaraj-bhanu*

*October 30, 2017*



# Details

**Commit Info**

There is one commit after the Friday 2:00 pm deadline. As the report at the time of deadline was incomplete, we are choosing the last commit after deadline.

Commit-id : `493fbe8b75b307c78410902245ff3d19142f936b`

```
commit 493fbe8b75b307c78410902245ff3d191he42f936b
Author: ritvikareddy18 <ritvikareddy18@ccs.neu.edu>
Date:   Fri Oct 27 15:11:13 2017 -0400

    Report

commit b3aa32e9ee2d1a82bad7f1c84f68ce38089ec52a
Author: ritvikareddy18 <ritvikareddy18@ccs.neu.edu>
Date:   Fri Oct 27 13:55:03 2017 -0400

    Time counter
```

**Review**

a6-Ritvika-Xia

**Reviewer**

Balaraj, Bhanu

# Execution

- Code runs as mentioned in the `readme` and `Makefile`

# Report review

- Report is generated on the subset of the million songs data, it would be interesting to see the report on the entire dataset.
- Report is well formatted and provides detailed information with graphs comparing execution time.

- Report does not mention Distinct artists, distinct albums, distinct songs.
- No information is given on Top 5 Familiar Artists, Top 5 Hottest Genre, Top 5 Hottest Artists, Top 5 Popular Keys in the report.
- The most common words include non-alphabetic characters. You can use stripping or character checking to ignore non letter characters.

```
##       Word Count
## 1 VERSION)   716
## 2 ME         341
## 3 (ALBUM     321
## 4 LOVE       309
## 5 MY         292
```

- System specifications are not provided.

# Code review

- The authors could have directly written the code in `a6-ritvika-xia` than `a6-ritvika-xia/assignment6`. Dir `assignment6` seems redundant, `pdpmr-f17/a6-ritvika` should have been base dir than `pdpmr-f17/a6-ritvika/assignment6`.
- There is a file with name `untitled.scala` in the src which is not used anywhere. Could have been removed.
- The code is well written and easy to understand.
- The following code comments can be removed.

```
//    val fw = new FileWriter("output/common words")
//    for(i <- top5Common){
//      fw.append(i._1)
//      fw.append(",")
//      fw.append(i._2.toString)
//      fw.append("\n")
//    }
//    fw.close()
```

- Since the method functionality is same for all three below functions. `Any` could have been used to make it more generic.

```
def makeCSVLong(thingsToPrint:List[(String,Long)],fileName:String):Unit ={
    val  name = "output/"+fileName
    val fw = new FileWriter(name)
    for(i <- thingsToPrint){
      fw.append(i._1)
      fw.append(",")
      fw.append(i._2.toString)
      fw.append("\n")
    }
    fw.close()
  }

  def makeCSVInt(thingsToPrint:List[(String,Int)],fileName:String):Unit ={
    val  name = "output/"+fileName
    val fw = new FileWriter(name)
    for(i <- thingsToPrint){
      fw.append(i._1)
      fw.append(",")
      fw.append(i._2.toString)
      fw.append("\n")
    }
    fw.close()
  }

  def makeCSVDouble(thingsToPrint:List[(String,Double)],fileName:String):Unit ={
    val  name = "output/"+fileName
    val fw = new FileWriter(name)
    for(i <- thingsToPrint){
      fw.append(i._1)
      fw.append(",")
      fw.append(i._2.toString)
      fw.append("\n")
    }
    fw.close()
  }
```

Can be replaced with

```
def makeCSVAny(thingsToPrint:List[(String,Any)],fileName:String):Unit ={
    val  name = "output/"+fileName
    val fw = new FileWriter(name)
    for(i <- thingsToPrint){
      fw.append(i._1)
      fw.append(",")
      fw.append(i._2.toString)
      fw.append("\n")
    }
    fw.close()
  }
```

- For calculating most common words, the code does not check if the word is a valid word or not, resulting in words with additional characters.

```
val commonWords = songInfo.flatMap(record => record(TITLE).toUpperCase().split(" ")).
      filter(!_.isEmpty).filter(!ignored.contains(_)).countByValue()

Additional check for valid characters are needed, can be done as follows.

val commonWords = songInfo.flatMap(record => record(TITLE).toUpperCase().split(" ")).
      filter(!_.isEmpty).filter(!ignored.contains(_) || _ => _.isLetter).countByValue()
```

# Final Thoughts

- Code readable and easy to understand.
- As the approach is different we can’t compare the results, but code seems to work.