“Big Data” analysis methods

Presentation

Frank Dávid

Tools used

Data

http://mtg.upf.edu/static/datasets/last.fm/lastfm-dataset-360K.tar.gz

17,5M rows

Music listening data of 360K users

Data

1. profile

user id	gender	age	country	date of registration

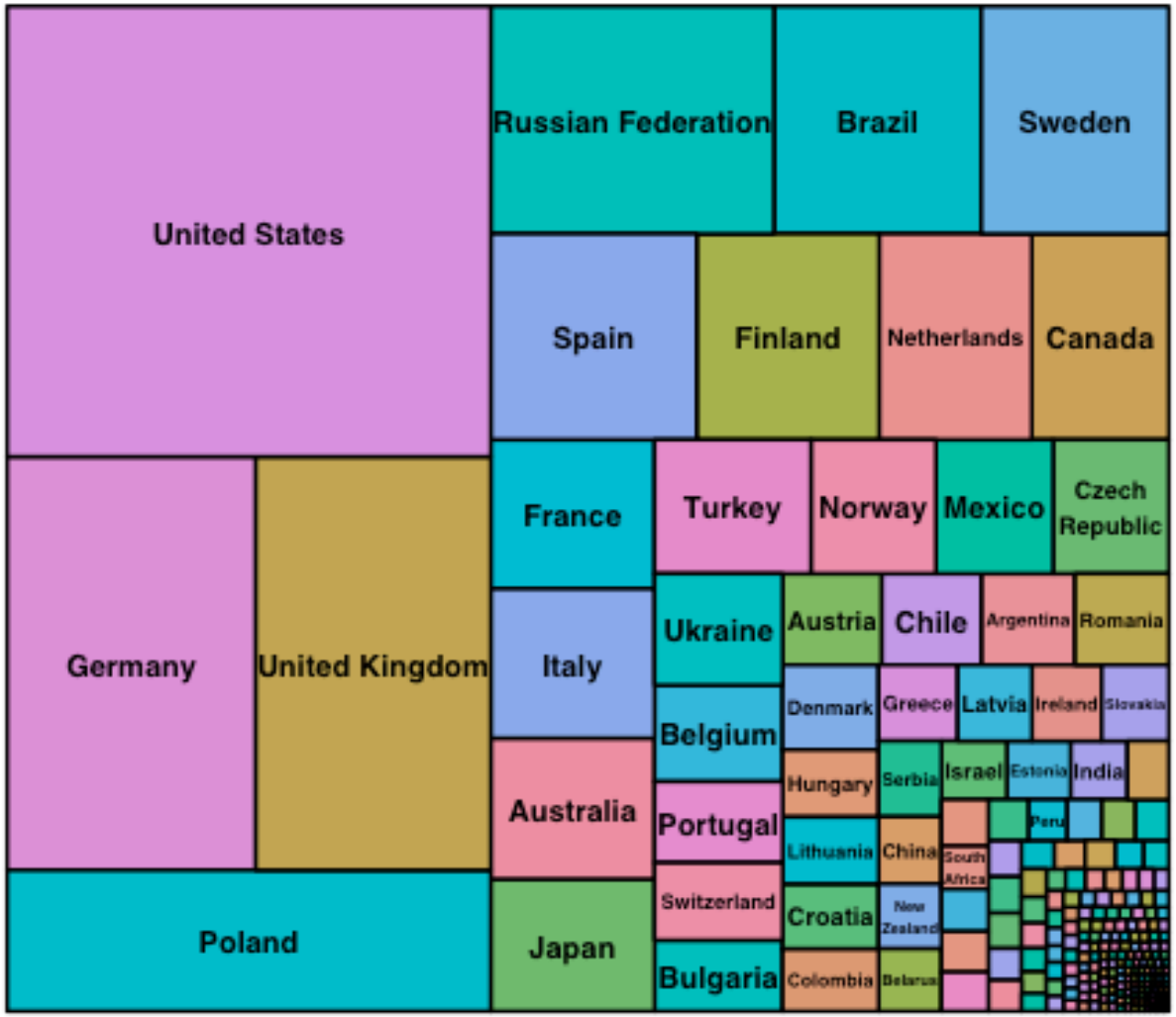
2. listening history

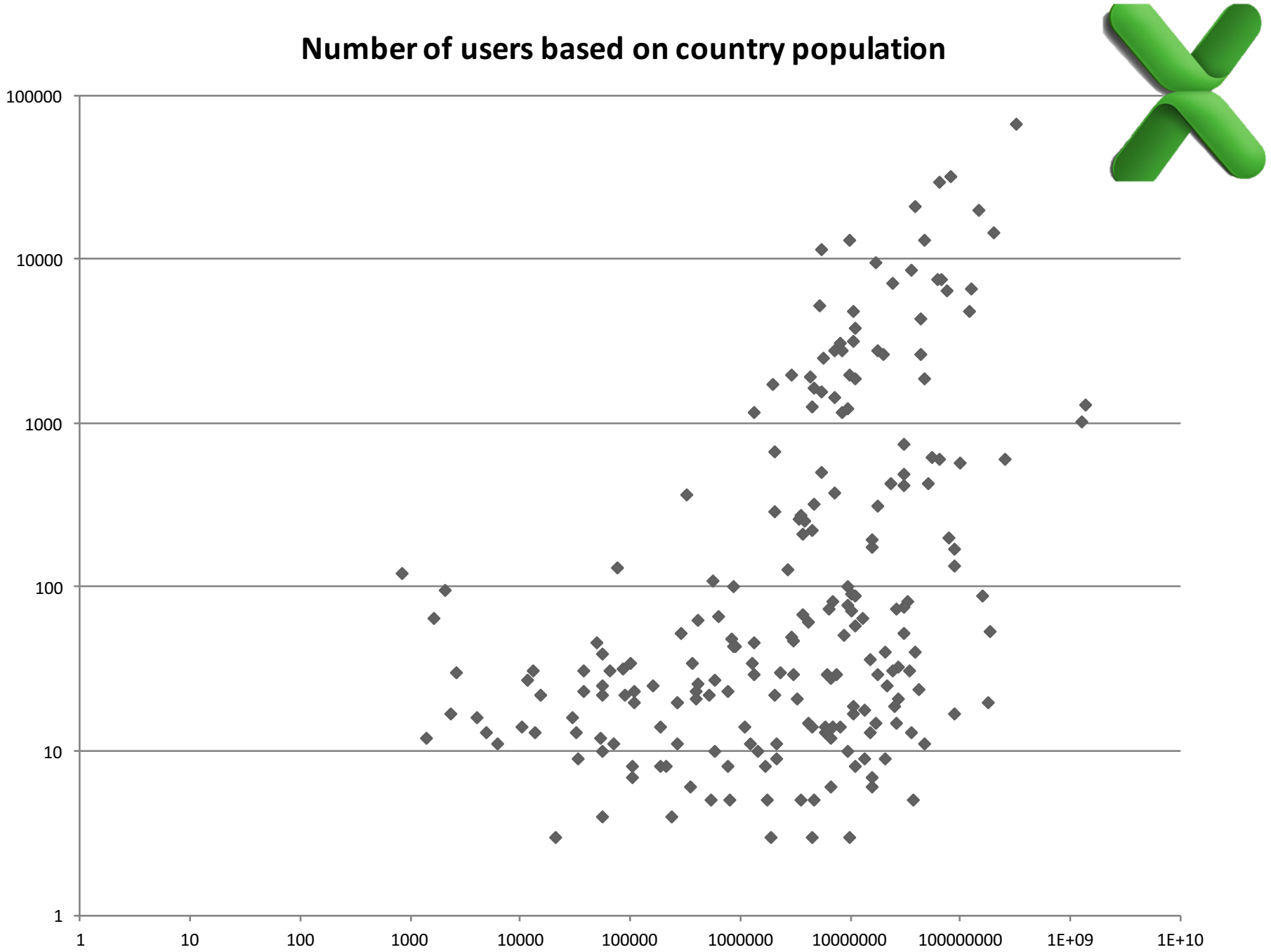
user id	artist id	artist name	number of listening

Distribution of users

```scala
def main(args: Array[String]) {
  val sc = new SparkContext(master = "local", appName = "countries")

  val profiles = sc.textFile("/user/hadoop/mit/profiles.tsv")

  val result = profiles.map {_.split('\t')(3)}.countByValue()

  result.toSeq.sortBy(-_._2).foreach { case (country, count) =>
    println(s"$country\t$count")
  }
```

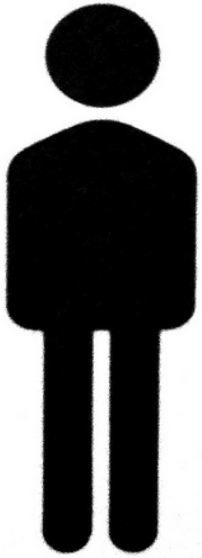
Distribution of users




Number of users based on country population
100000
10000
1000
100
10
1
1
10
100
1000
10000
100000
1000000
10000000
100000000
1E+09
1E+10

Females - males

?



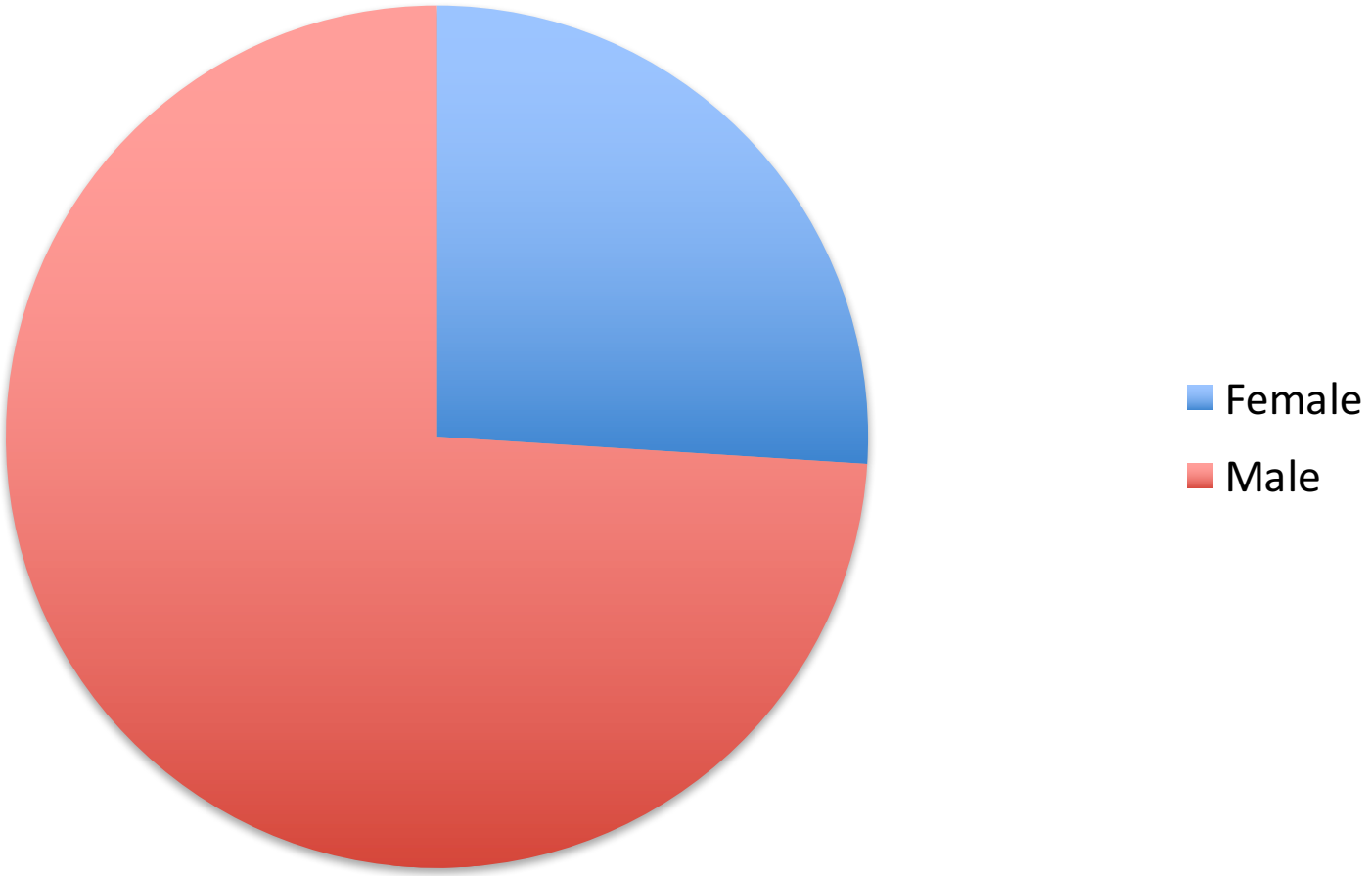
?

Females - males

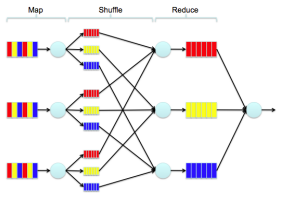
84,930

241,642

Females - males



Females - males 2

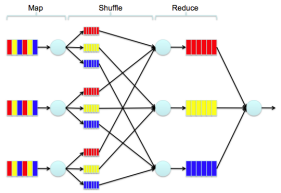


- Which countries have the largest female representation?

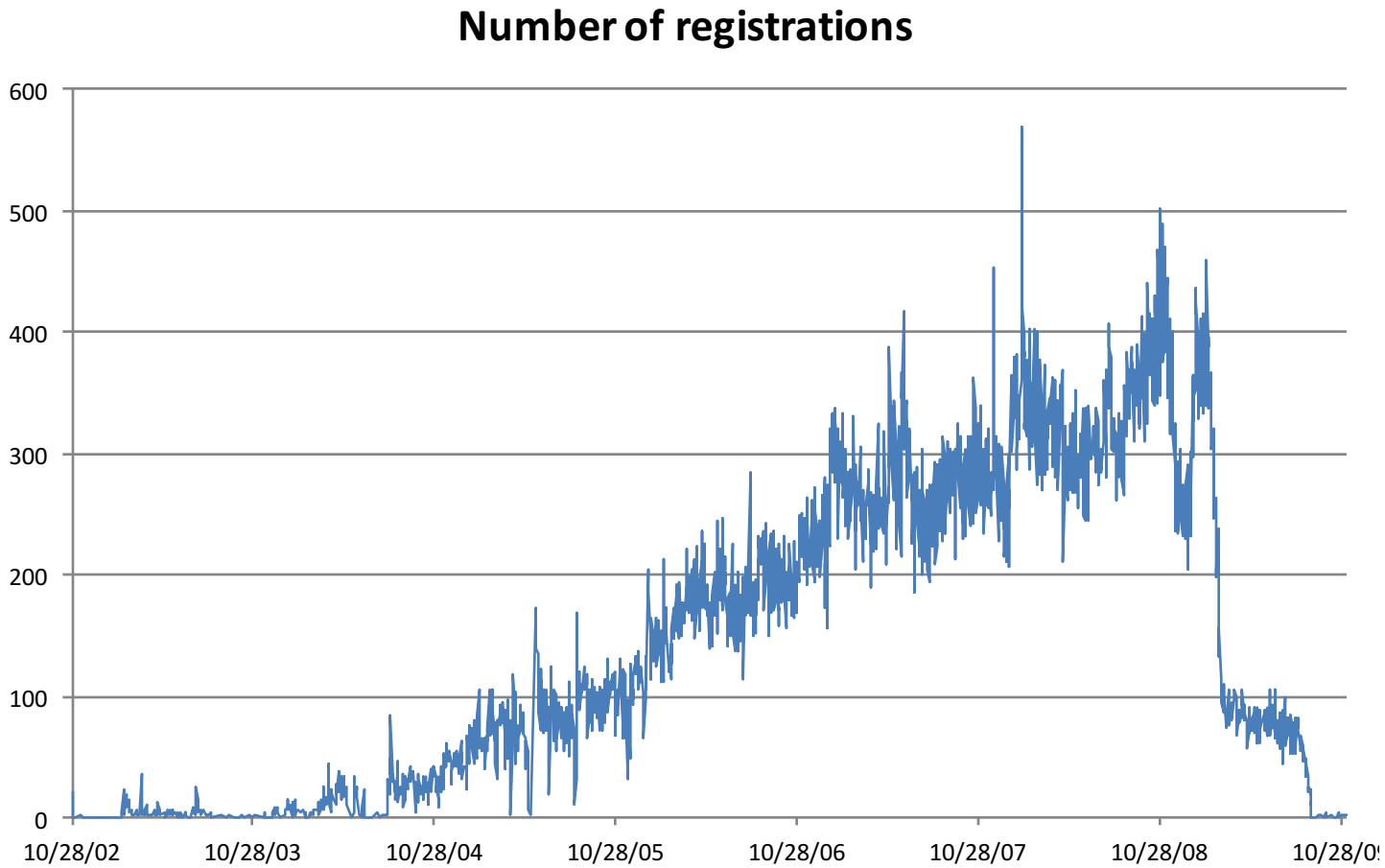
Liberia	100.0%
Palau	100.0%
Cameroon	80.0%
Saint Kitts and Nevis	80.0%
Suriname	75.0%
Dominica	62.5%
Turkmenistan	55.6%
Guam	54.5%
Bahamas	52.0%
Cocos (Keeling) Islands	51.9%
Benin	50.0%
Gabon	50.0%
Haiti	50.0%
Maldives	50.0%
Myanmar	50.0%
Northern Mariana Islands	50.0%

```scala
class ManAndWomen2
object ManAndWomen2 {

  class Mapper extends MapReduceBase with org.apache.hadoop.mapred.Mapper[LongWritable, Text, Text, Text] {
    def map(key: LongWritable, value: Text, output: OutputCollector[Text, Text], reporter: Reporter) {
      val parts = value.toString.split('\t')
      // filter for f and m values only
      if (parts(1) == "m" || parts(1) == "f") {
        output.collect(new Text(parts(3)), new Text(parts(1)))
      }
    }
  }

  class Reducer extends MapReduceBase with org.apache.hadoop.mapred.Reducer[Text, Text, Text, DoubleWritable] {
    def reduce(k2: Text, iterator: util.Iterator[Text], outputCollector: OutputCollector[Text, DoubleWritable], reporter:
    Reporter) = {
      var female, male = 0
      iterator.foreach {
        _.toString match {
          case "m" => male += 1
          case "f" => female += 1
          case _ => // should not happen in theory
        }
      }
      outputCollector.collect(k2, new DoubleWritable(female / (female + male).toDouble))
    }
  }

  def main(args: Array[String]) {
    val conf: JobConf = new JobConf(classOf[SignupDate])
    conf.setJobName("men_and_women_2")
    conf.setOutputKeyClass(classOf[Text])
    conf.setOutputValueClass(classOf[Text])
    conf.setMapperClass(classOf[Mapper])
    conf.setReducerClass(classOf[Reducer])
    conf.setInputFormat(classOf[TextInputFormat])
    conf.setOutputFormat(classOf[TextOutputFormat[_, _]])
    FileInputFormat.setInputPaths(conf, new Path("/user/hadoop/mit/profiles.tsv"))
    FileOutputFormat.setOutputPath(conf, new Path("/user/hadoop/mit/men_and_women_2"))
    JobClient.runJob(conf)
  }
}
```

How the site popularity fluctuated



2003-03-18
2003-07-06
2003-07-07
2003-07-08
2004-01-05
2004-08-01
2005-03-30
2005-05-09
2005-05-19
2005-05-20
2005-08-05
2005-08-09
2005-08-10
2005-11-19
2006-05-21
2006-05-31
2006-07-24
2007-05-29
2007-11-27
2008-01-23
2008-04-17



The most popular artists

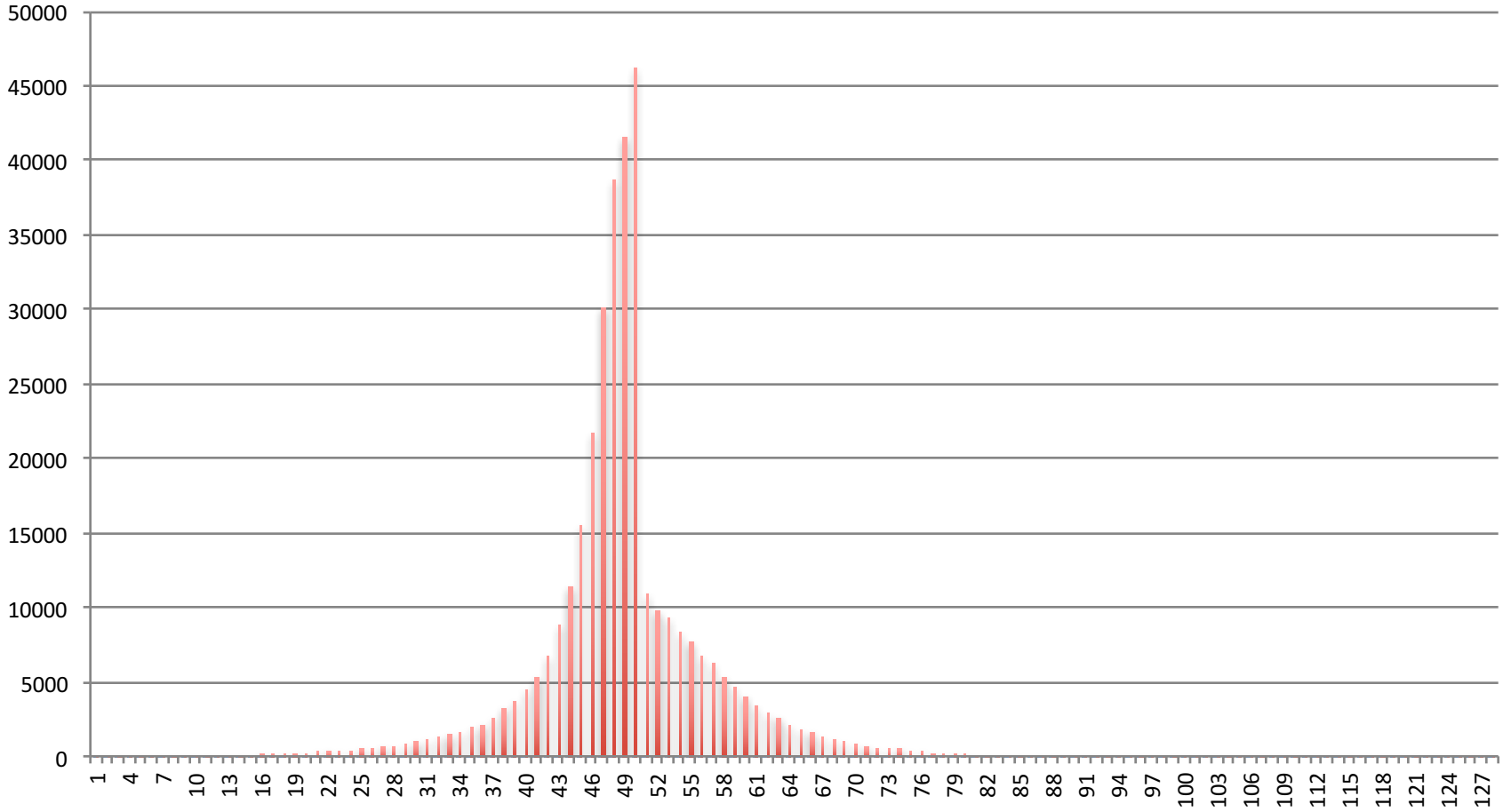
- the beatles: 30499140
- radiohead: 27452136
- coldplay: 16701858
- pink floyd: 15965959
- metallica: 15498759
- muse: 15463089
- nine inch nails: 14090643
- red hot chili peppers: 13562637
- linkin park: 12848569
- system of a down: 11927204

… for users under 30

- the beatles: 20801873
- radiohead: 19374267
- coldplay: 12337184
- muse: 12296218
- metallica: 12005206
- pink floyd: 11183693
- linkin park: 10688851
- red hot chili peppers: 10580026
- nine inch nails: 9929267
- system of a down: 9830666

```scala
object MostPopularUnder30 {

  def main(args: Array[String]) {
    val sc = new SparkContext(master = "local", appName = "most_popular")

    //read data files and group by user id
    val plays = sc.textFile("/user/hadoop/mit/plays.tsv").keyBy(_.split('\t')(0))
    val profiles = sc.textFile("/user/hadoop/mit/profiles.tsv").keyBy(_.split('\t')(0))

    // join by user id and calculate the sum of play count
    val topArtists = plays.join(profiles).flatMap { case (user, (play, profile)) =>
      // split tab separated rows
      val profileValues = profile.split('\t')
      val playValues = play.split('\t')
      try {
        // if age < 30, emit artist and play count
        if (profileValues(2).toInt < 30) {
          val playCount = playValues(3).toLong
          Some((playValues(2), playCount))
        } else {
          None
        }
      } catch {
        // handle wrong values
        case _: NumberFormatException => None
      }
    }
    // for each artist calculate the sum of play count
    .reduceByKey(_ + _)
    // pick top 10 by the sum of play count
    .top(10)(Ordering.by(_._2))

    //display the result
    topArtists.foreach { case (artist, plays) =>
      println(s"$artist: $plays")
    }
```

How many distinct artists people listen to



Who are similar users

- for 00000c289a1829a808ac09c00daf10bc3c4e223b
  - 55465b640e972d1131a14caede9c33f9148b5510
  - 88efae305a7d3a10446943ca97568e0bd6a773e1
  - 9b1897890ca93c819511599c229f404b001e3601

```scala
object Similar {

  def main(args: Array[String]) {
    val sc = new SparkContext(master = "local", appName = "similar")

    val targetId = "00000c289a1829a808ac09c00daf10bc3c4e223b"
    val plays = sc.textFile("/user/hadoop/mit/plays.tsv")

    // create tuples of (artist, user)
    val artistUsers = plays.map { row =>
      val values = row.split('\t')
      (values(2), values(0))
    }

    // collect the target's favorite artists
    val targetFavorite = artistUsers.filter(_._2 == targetId).map(_._1).collect()

    // for each artist-user pair, the user should not equal the target and the artist must be
   in the targetFavorite
    val results = artistUsers.filter(artistUser => artistUser._2 != targetId && targetFavorite
   .contains(artistUser._1))
                  // group by user
                  .keyBy(_._2).groupByKey()
                  // select top 3 by the number of shared artists
                  .top(3)(Ordering.by(_._2.size))

    // display the result
    results.foreach(result => println(result._1))

  }
```