We want to obtain implicit user ratings for a TV program based on the users consumption behavior. From the BBC data set the percentage of a TV show watched by a user is readily available from column 3 of the CSV file (aggregation:totalUserProgramFraction). However, I think we can obtain a better estimate of the users interest in a TV program than just using this percentage by considering the 'locality' of the watched time span within the total time span of the TV program:

- *Program is watched from the beginning but not to the end:* We can argue that a user who watched a TV show from the very beginning, had an a priori expectation of the shows high quality (from its own perspective) and hence decided to watch the entire show beforehand. If the user tuned out prematurely we can assume that the quality of the program did not meet the users expectations. For this reason we do not map the rating linearly to the percentage of the watched program but with a positive curvature (see red graph in figure 1).

- *Program is watched to the end but not from the beginning:* In this case we can assume that the user was not aware of the program or did not know it at all. Hence, the interest of the user in the program could actually be high, even though the user watched only a short time span of the entire program. For this reason we map the watched program share to the user rating with a negative curvature (see blue graph in figure 1).

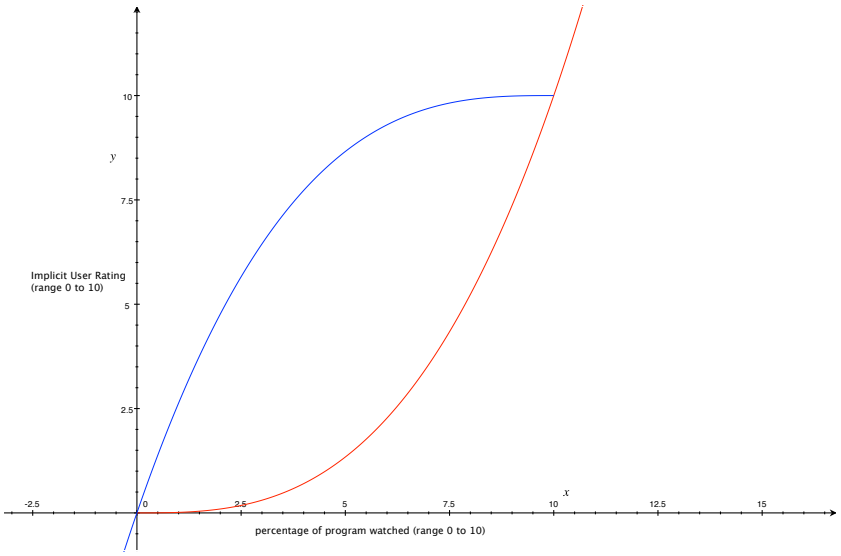

Figure 1: In red: watched share user rating mapping for premature termination. In blue: watched share user rating mapping for delayed tune in.

The afore-mentioned considerations can be translated into a simple algorithm (see listing 1), where p_1 and p_2 adjust the curvature as desired by the user. The share of TV program watched is in the range of 0 to 1 and we want to obtain a user interest rating

in the range of o to 10.

Listing 1: Main method for implicit user rating generation.

```
1
2  double getImplicitRating(UsageDataSetEntry entry) {
3
4          double percentageWatched <- entry.get("aggregation:
               totalUserProgramFraction") * 10; // is in range
               0-1 in data set
5
6          boolean fromBeginning <-
               watchedFromBeginningOfProgram(entry);
7          boolean tillEnd <- watchedTillEndOfProgram(entry);
8
9          if (fromBeginning && tillEnd)
10                 return percentageWatched;
11
12         if (fromBeginning)
13                 return  $\frac{percentageWatched^{p_1}}{10^{p_1-1}}$;
14
15         if (tillEnd)
16                 return  $10 - \frac{(10 - percentageWatched)^{p_2}}{10^{p_2-1}}$;
17
18         return percentageWatched;
19 }
```

Listing 2: Method watchedFromBeginningOfProgram.

```
1
2  boolean watchedFromBeginningOfProgram(UsageDataSetEntry
       entry) {
3
4          long epgStartTime <- epgData.get(entry.get("epg:id")
               ).get("epg:start"); // in seconds since start of
               epoch
5
6          long userStartTime <- entry.get("program_session:
               start_time"); // in seconds since start of epoch
7
8          double tolerance <- entry.get("epg:duration"); // in
                seconds
9          tolerance <- (tolerance *
               START_TIME_TOLERANCE_PERCENTAGE) / 100.0;
```

```
10
11          if (epgStartTime + tolerance >= userStartTime)
12                  return true;
13
14          return false;
15  }
```

Listing 3: Method watchedTillEndOfProgram.

```
1
2   boolean watchedTillEndOfProgram(UsageDataSetEntry entry) {
3
4           long epgEndTime <- epgData.get(entry.get("epg:id")).
               get("epg:end"); // in seconds since start of
               epoch
5
6           long userEndTime <- entry.get("program_session:
               start_time"); // in seconds since start of epoch
7           userEndTime <- userEndTime + entry.get("
               program_session:duration"); // in seconds
8
9           double tolerance <- entry.get("epg:duration"); // in
                seconds
10          tolerance <- (tolerance *
               END_TIME_TOLERANCE_PERCENTAGE) / 100.0;
11
12          if (epgEndTime - tolerance <= userEndTime)
13                  return true;
14
15          return false;
16  }
```