Visualizing Big Ranking Data

STAT3319 Statistics Project Written Report

Name: Yiming Li
University Number: 2011810187

Supervisor: Dr. Philip L.H. Yu

8 May, 2014

Contents

1 Introduction **3**

2 Definitions and Notations **3**

2.1 Complete, Without-Ties Rankings . . . 3

2.2 Complete, With-Ties Rankings . . . 3

2.3 Incomplete Rankings . . . 3

3 Visualization Techniques **4**

3.1 Permutation Polytopes . . . 4

3.2 Multidimensional Scaling . . . 4

4 Calculating the Distance Between Rankings **6**

4.1 The Distance Formula . . . 6

4.1.1 Distance Between Permutations . . . 6

4.1.2 Distance Between Incomplete Rankings . . . 6

4.2 Normalizing the Kendall Distance . . . 7

4.2.1 Motivation . . . 7

4.2.2 Minimum and Maximum Distances Between Incomplete Rankings . . . 7

4.2.3 Normalized Distance . . . 8

4.2.4 Justification . . . 8

5 Experiments **10**

5.1 Data and Preparation . . . 10

5.1.1 Raw Data . . . 10

5.1.2 Data Preparation . . . 10

5.2 Data Visualization . . . 12

5.2.1 Plotting by Rater Attributes . . . 12

5.2.2 Plotting by "Fan Groups" . . . 12

5.3 Possible Interpretation . . . 16

5.3.1 Finding the Representative Rankings for Local Maximum Density Regions 16

5.3.2 Latent Meaning of the Axes . . . 17

5.4 The Goodness of Fit of 2-D Multidimensional Scaling . . . 17

6 Discussion **19**

Appendix A The Five Representative Rankings in Full **20**

Appendix B R codes **25**

References **49**

List of Tables

1 Goodness of Fit of MDS 19

List of Figures

1 Colored Permutation Polytope for Four Objects [4] 5
2 MDS Plot Using d^*_{orig} 8
3 MDS Plot Using $(d^*_{orig} - m^*)$ 9
4 MDS Plot Using the Normalized Distance 9
5 The 55 Selected Movies 11
6 MDS Plot with Hexagonal Binning 13
7 MDS Plot with Kernel Smoothing 13
8 MDS Plot by Rater Gender 14
9 MDS Plot by Rater Age 14
10 MDS Plots with Contour Curves for Different "Fan Groups" 15
11 The Five Regions with Local Maximum Density 16
12 Representative Rankings for Maximum Density Regions 18
13 Peak A 20
14 Peak B 21
15 Peak C 22
16 Peak D 23
17 Peak E 24

1 Introduction

A ranking represents the order of preference one has with respect to a set of objects. In reality, ranking data naturally arise from m raters ordering t objects according to some preference criterion. In the next section, we would provide rigorous definitions of different types of rankings.

Effective visualization of ranking data could reveal important properties of the population of raters, objects as well as the relationship between raters and objects [1]. In this project, we develop an intuitive, easy to use, and computationally efficient framework for the visualization of ranking data with a large number of raters. The visualization techniques will be applied to big ranking data such as movie preferences.

2 Definitions and Notations

2.1 Complete, Without-Ties Rankings

The simplest case of ranking is when all the objects are ranked without ties. A complete, without-ties ranking of the objects $S = \{1, \ldots, t\}$ is a permutation of S which we denote as $\mu = (\mu(1), \mu(2), \ldots, \mu(t))$, where $\mu(i)$ is the rank of object i. In fact, μ is a bijection $S \to S$ mapping objects to ranks.

In this report, we will represent a permutation with the notation used by Kidwell et al. [1] – a ranking is denoted by a sorted list of the objects, most preferred to least, separated by vertical bars, i.e. $\mu^{-1}(1)|\ldots|\mu^{-1}(t)$. For example, one complete ranking with the preference *(Most preferred)* $1 > 2 > 3 > 4 > 5 > 6$ *(Least preferred)* would be denoted as 1|2|3|4|5|6.

2.2 Complete, With-Ties Rankings

Complete, with-ties rankings allow some of the objects in S to be of tied rank. Such rankings are represented using the above notation with the tied objects separated by commas. For example, $1|2,3|4$ implies object 1 is the most preferred, object 4 is the least preferred, whereas objects 2 and 3 are tied for the middle ranks.

We assume that a tie is due to lack of information, and more information could break the tie. Under this assumption, $1|2,3|4$ corresponds to the set of complete, without-ties rankings $\{1|2|3|4; 1|3|2|4\}$.

2.3 Incomplete Rankings

In real life, complete rankings are almost non-existent, since many raters only rank a subset of the objects. The complete ranking μ of t objects is said to be compatible with an incomplete

ranking μ^* of a subset of k of these objects, $2 \leq k \leq t$, if the relative ranking of every pair of objects ranked in μ^* coincides with their relative ranking in μ [2]. For example, when $t = 4$, the compatible set of the incomplete ranking $1|2|4$ is $\{3|1|2|4; 1|3|2|4; 1|2|3|4; 1|2|4|3\}$.

3 Visualization Techniques

When the size of the ranking data is very large, it is extremely difficult to understand the data by merely looking at the raw data or the descriptive statistics generated from the data. A picture is worth a thousand words. An efficient visualization framework, which we aim to establish in our project, would certainly provide insight about the structure and underlying pattern of big ranking data. In this section, we would introduce two major visualization methods for ranking data – permutation polytopes and MDS.

3.1 Permutation Polytopes

The permutation polytope [3] has vertices corresponding to permutations and edges corresponding to adjacent transposition of items (Figure 1). Therefore, the Kendall distance between two vertices connected by an edge on the polytope is one. More generally, the distance between two permutations is the length of the shortest path between the two corresponding vertices on the polytope.

When the number of items is greater than four, the polytope could not be embedded in $\mathbb{R}^3$. Hence, the chance of using it for visualization purposes is rather low [3].

3.2 Multidimensional Scaling

As one of the most frequently used visualization methods, multidimensional scaling (MDS) takes an input matrix giving dissimilarities between pairs of items and outputs a coordinate matrix whose configuration minimizes a loss function called strain [5].

In our setting of ranking data, MDS would take a distance matrix whose $(i, j)^{th}$ element is the distance between the i^{th} and j^{th} rankings as input, and each point in the returned coordinate matrix would correspond to a certain ranking [6]. In the next section, we would introduce a scheme to measure the distance between different rankings, which are possibly incomplete and with-ties.

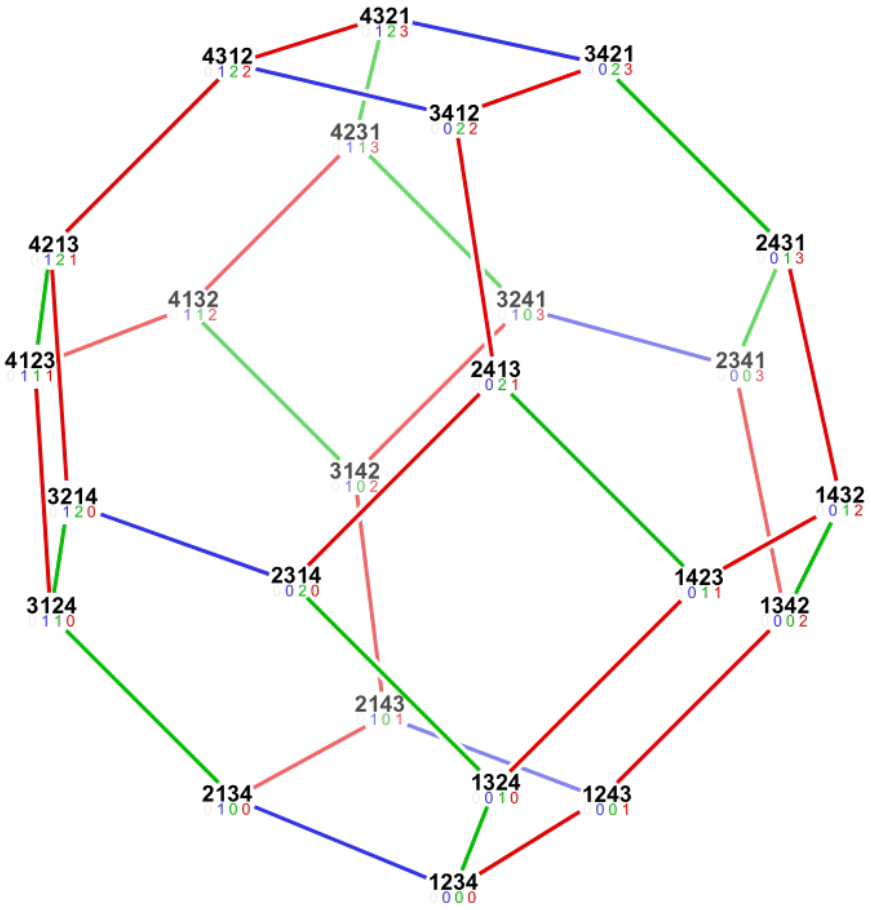

Figure 1: Colored permutation polytope for four objects [4]. The edges indicate the bitwise $\leq$ relation whereas the colors of the edges indicate which digit changes.

4 Calculating the Distance Between Rankings

4.1 The Distance Formula

4.1.1 Distance Between Permutations

Many different distance functions for measuring the discrepancy between two individual permutations have been proposed in the literature, including the Spearman distance [7], the Kendall distance [8], and the Hamming distance [9].

Given two permutations $\mu = (\mu(1), \mu(2), \ldots, \mu(t))$ and $\nu = (\nu(1), \nu(2), \ldots, \nu(t))$ ranking t objects, the Kendall distance is a metric that counts the number of pairs of items for which μ and ν have opposing orderings [8]. It is also called the bubble-sort distance since it is the minimum number of transpositions of adjacent items needed to bring μ to ν [10].

The formula for calculating the Kendall distance [11] is

$$d(\mu, \nu) = \sum_{i<j} \{1 - sgn[\mu(j) - \mu(i)] \cdot sgn[\nu(j) - \nu(i)]\}, \text{ where } sgn[x] = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x < 0 \end{cases} \quad (1)$$

It is straightforward that the mean of the Kendall distance is

$$c = \frac{t(t-1)}{2} \quad (2)$$

Since the Kendall distance is the most popular measure in current research [1], we would employ it in this project.

4.1.2 Distance Between Incomplete Rankings

The distance $d^*(\mu^*, \nu^*)$ between two incomplete rankings μ^* and ν^* is defined to be the average of the distances $d(\mu_i, \nu_j)$ taken over all pairs of complete rankings (μ_i, ν_j) compatible with μ^* and ν^* respectively [12].

The efficient calculation of $d^*(\mu^*, \nu^*)$ is based on the combinatorial properties of the Kendall distance [13]. For an incomplete ranking μ of k objects and a given pair of objects (i, j), $1 \leq i < j \leq t$, let

$$a(i,j) = \begin{cases} sgn[\mu^*(j) - \mu^*(i)] & \text{if both } i \text{ and } j \text{ are ranked,} \\ 1 - \frac{2\mu^*(i)}{k+1} & \text{if only } i \text{ is ranked,} \\ \frac{2\mu^*(j)}{k+1} - 1 & \text{if only } j \text{ is ranked,} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

The Kendall distance between the incomplete rankings μ^* and ν^* is then

$$d^*_{orig}(\mu^*, \nu^*) = c - A, \tag{4}$$

where

$$A = A(\mu^*, \nu^*) = \sum_{i<j} a_\mu(i,j) a_\nu(i,j),$$

c is defined in Equation 2, and $a_\mu(i,j)$ and $a_\nu(i,j)$ are the scores calculated according to Equation 3 corresponding to the two incomplete rankings μ^* and ν^*, respectively [13].

4.2 Normalizing the Kendall Distance

4.2.1 Motivation

A good distance measure requires identity, i.e. $d(X,Y) = 0$ if and only if $X = Y$. However, due to our definition of the distance between two incomplete rankings, when μ^* is an incomplete ranking, $d(\mu^*, \mu^*) \neq 0$. Therefore, this measure needs amendment such that $\forall$ incomplete ranking μ^*, $d(\mu^*, \mu^*) = 0$.

4.2.2 Minimum and Maximum Distances Between Incomplete Rankings

Alvo and Cabilio [14] proposed a way to calculate the minimum and maximum values of the distance between two incomplete rankings when observations are missing at random. For fixed $k_1 \leq k_2$, suppose the pattern of missing observations is randomly selected from the set of all possible patterns. Then, for the Kendall case, the minimum and maximum values of the distance are of the form

$$m^* = c - \gamma(i), M^* = c + \gamma(i) \tag{5}$$

where c is defined in Equation 2 and $\gamma(i)$ is given as

$$\gamma(1) = \frac{(k_1 - 1)[t(3k_2 - k_1) + k_2(k_1 + 3)]}{6(k_2 + 1)}, \; k_1 \text{ odd}$$

$$\gamma(2) = \frac{k_1[3k_1k_2(t+1) - (k_1^2 + 2)(t - k_2) - 3(k_2 + 1)]}{6(k_1 + 1)(k_2 + 1)}, \; k_1 \text{ even}$$

4.2.3 Normalized Distance

To define a new distance d^* such that $\forall$ incomplete ranking μ^*, $d^*(\mu^*, \mu^*) = 0$, we minus the minimum value of the Kendall distance from the original Kendall distance. To further normalize it, we set

$$d^* = \frac{d^*_{orig} - m^*}{M^* - m^*}, \text{ where } \begin{cases} d^*_{orig} \text{ is the original Kendall distance} \\ m^* \text{ is the minimum value of the Kendall distance} \\ M^* \text{ is the maximum value of the Kendall distance} \end{cases} \tag{6}$$

4.2.4 Justification

To justify the new measure we proposed in Equation 6, we present below a toy example.

Based on twelve rankings (1|2|3, 1|2|3, 1|3|2, 2|1|3, 2|3|1, 3|1|2, 3|2|1, 1|2|3|4, 1|2|3|4|5, 1|2|3|4|5|6, 1|2|3|6|5|4, 6|5|4|3|2|1) of six objects, we calculate three distance matrices using respectively d^*_{orig}, $(d^*_{orig} - m^*)$ and the normalized distance d^* as the metric. Next, we use MDS to visualize the three distance matrices, as shown in Figures 2, 3 and 4.

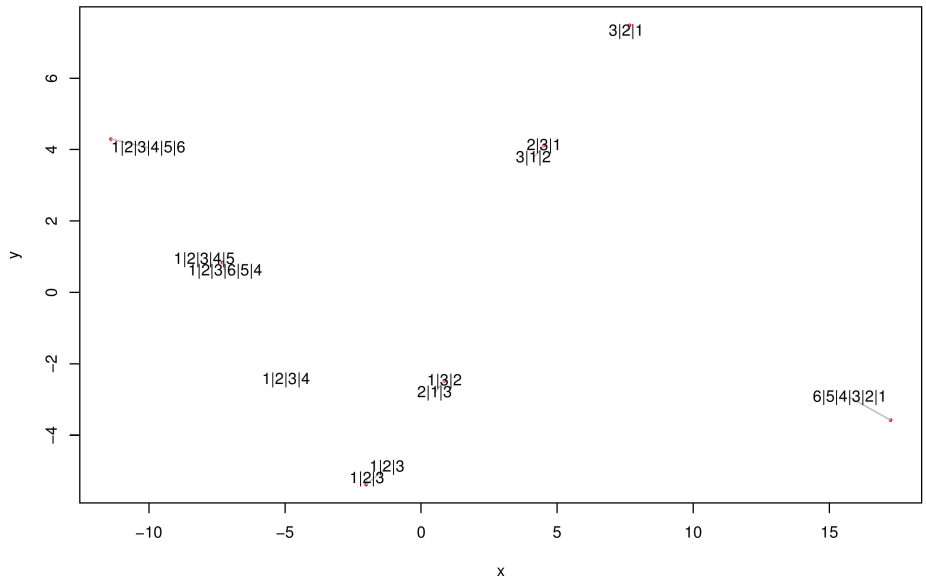

Figure 2: MDS plot using d^*_{orig}. The rankings 1|2|3 and 1|2|3, although close, do not overlap with each other. Besides, it is not sensible that the distance between 1|2|3 and 1|2|3|4|5|6 is almost the same as that between 1|2|3 and 6|5|4|3|2|1.

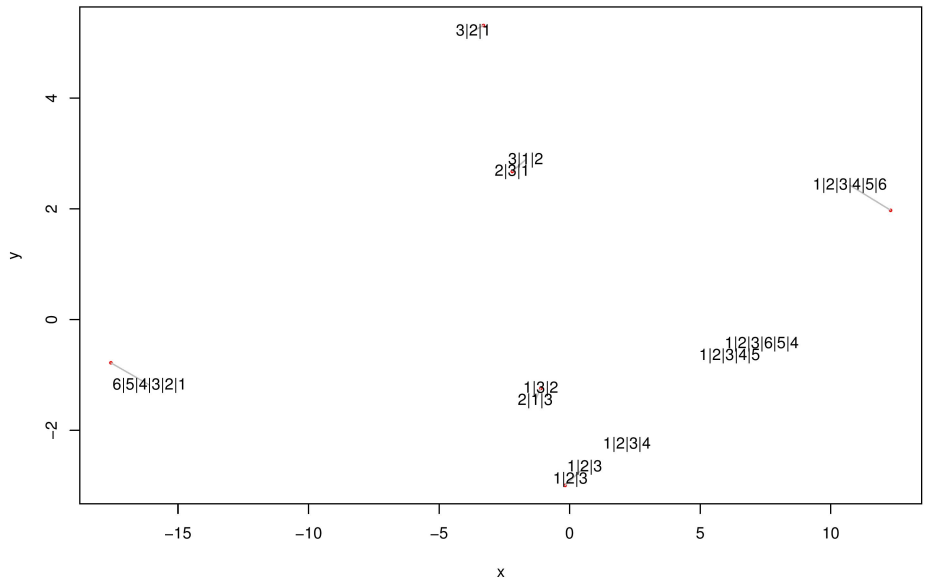

Figure 3: MDS plot using $(d^*_{orig} - m^*)$. The rankings 1|2|3 and 1|2|3 now overlap with each other. However, the relative locations of 1|2|3, 1|2|3|4|5|6 and 6|5|4|3|2|1 are still counter-intuitive.

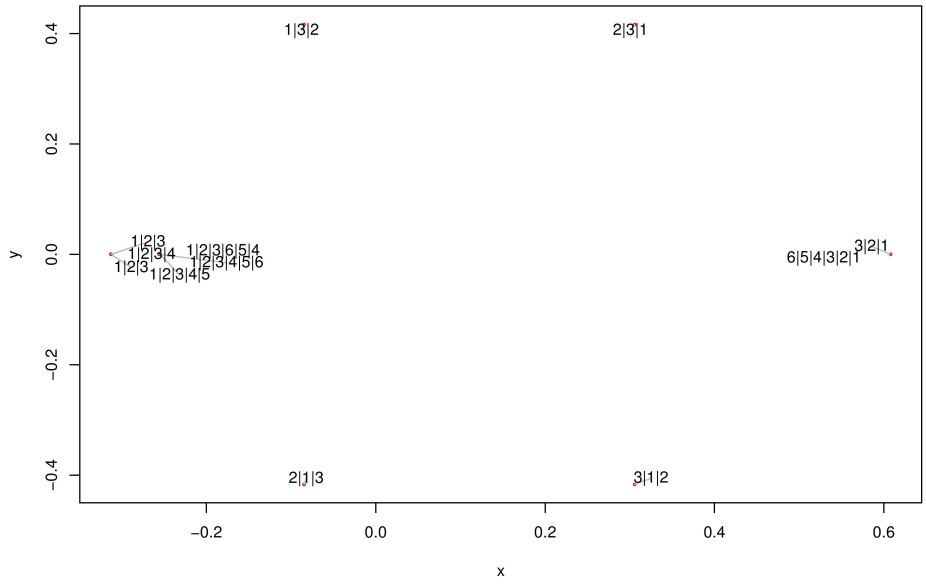

Figure 4: MDS plot using the normalized distance. The rankings with the same sub-strings (1|2|3, 1|2|3|4, 1|2|3|4|5, 1|2|3|4|5|6 and 1|2|3|6|5|4) now lie close to each other.

As demonstrated by our toy example, the normalized distance is a better measure than the other two in terms of visualizing ranking data. Therefore, we have utilized this measure instead of the original Kendall distance in our project.

5 Experiments

In this section, we employ the framework described in the preceding sections to visualize the data set MovieLens [15].

5.1 Data and Preparation

5.1.1 Raw Data

The data set we are using contains $1,000,209$ anonymous ratings of $3,952$ movies made by $6,040$ raters who joined the web site MovieLens in 2000 [15]. The ratings were made on a five-star scale with whole-star ratings only. All the raters in this data set have at least 20 ratings and have voluntarily provided their demographic information, including gender, age, occupation and zip code. Each movie is categorized by genre, and one movie could have multiple genres.

5.1.2 Data Preparation

We try to examine the raters' movie preferences by looking into six genres – action, children's, drama, horror, romance and thriller. We select the ten highest rated movies from each of the six genres and then take the union of them, resulting in 55 movies (Figure 5). Each of the 55 movies is assigned to one and only one genre. If a movie has been selected via two different genres, it would be allocated to the group in which it is higher ranked.

After we have finished the movie selection, we drop the raters who rated less than two movies among the 55, resulting in a total number of $5,625$ raters. Next, we also delete the obsolete ratings whose rater is not among the $5,625$ or whose rated movie is not among the 55.

For each rater, we could calculate his or her ranking of the 55 movies based on his or her ratings. Note that this ranking is very likely to be incomplete and with-ties. We then calculate the pair-wise normalized distance d^*, resulting in a $5,625 \times 5,625$ distance matrix.

Each rater has been allocated to one "fan group" for a specific genre. To do this, we calculate the average rating the rater gave to each of the six genres. The rater is then assigned to the genre he or she gave the highest average rating to. When several genres share the same highest average rating, the rater would be distributed to the genre whose movies he or she rated the most.

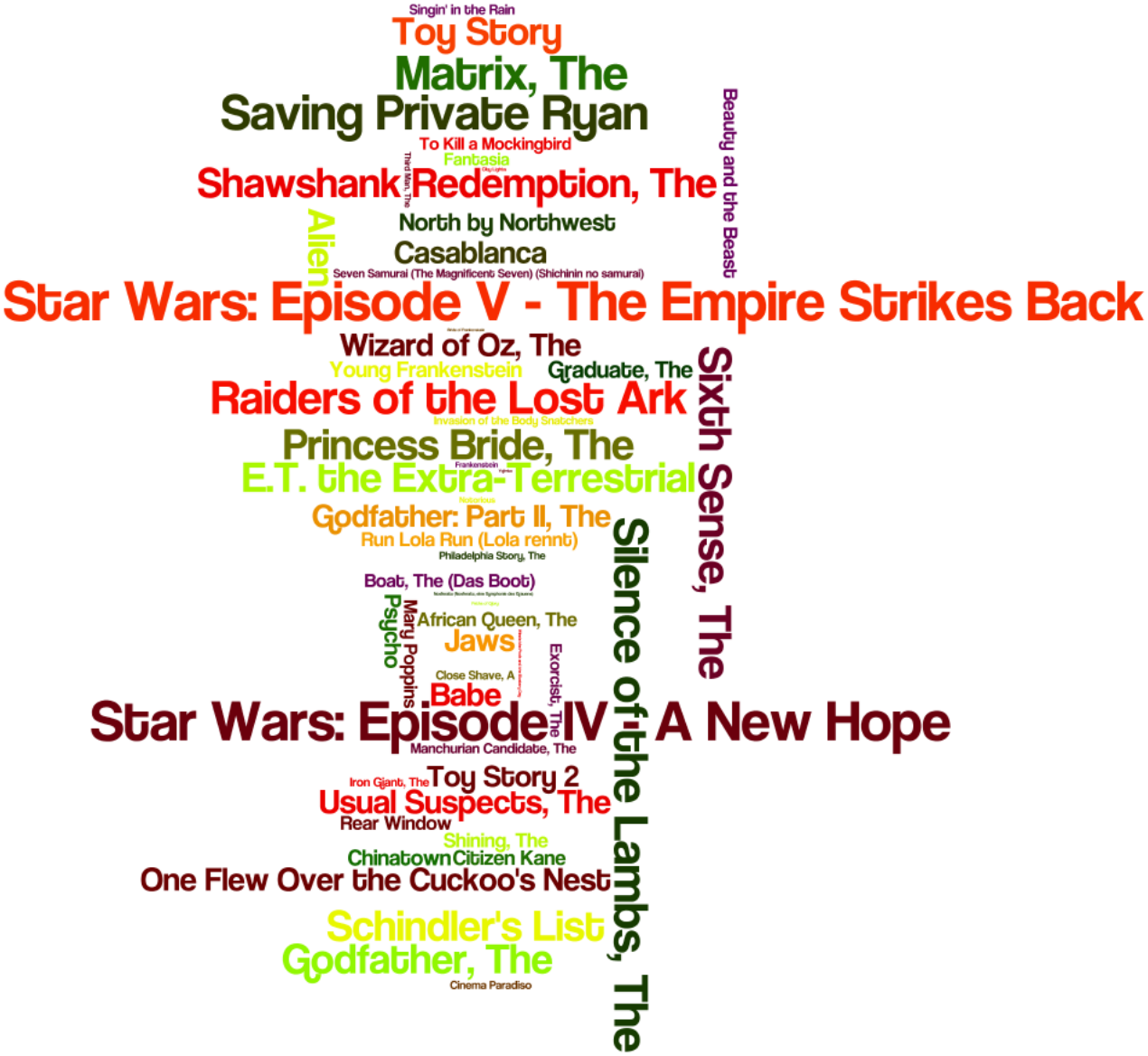

Figure 5: The 55 selected movies. This is a word cloud generated by Wordle [16]. After assigning each movie to a unique genre, there are eight action movies, ten children's movies, eight drama movies, ten horror movies, ten romance movies and nine thrillers. In this word cloud, the sizes of the movie titles are proportional to the total numbers of times they were rated by the 5,625 raters. The *Star Wars* series are the most popular in the sense that they were rated by the most people in this data set.

5.2 Data Visualization

We apply MDS to the distance matrix we have calculated and attain a set of $5,625$ points representing the different raters (rankings). Since the number of rankings $(5,625)$ is very large in our data set, using a scatterplot could be ineffective due to the overlaps. Hexagonal binning and kernel smoothing are used to better identify different clusters of movie users.

As shown in the Figures 6 and 7, the movie viewing population initially appears to be a single large cluster.

5.2.1 Plotting by Rater Attributes

We now would like to examine the relationship between the raters' movie preferences and demographic information, namely, gender and age.

We could notice from the Figures 8 and 9 that there seems to be no significant relationship between the raters' movie preferences and demographic information since the clusters are very hard to identify. We have found, though, male and young raters tend to have a higher density on the right of the graph. In order to interpret this discovery, we need to understand the latent meaning of the x and y axes.

5.2.2 Plotting by "Fan Groups"

As could be seen in Figure 10, though all the different fan groups cluster the most in the middle-left part, they have different peaks where the fans cluster the most. This fact gives us a rationale to find a representative ranking for each region where the density reaches a local maximum and interpret the meaning of the x and y axes accordingly. The details of this approach would be covered in Section 5.3.

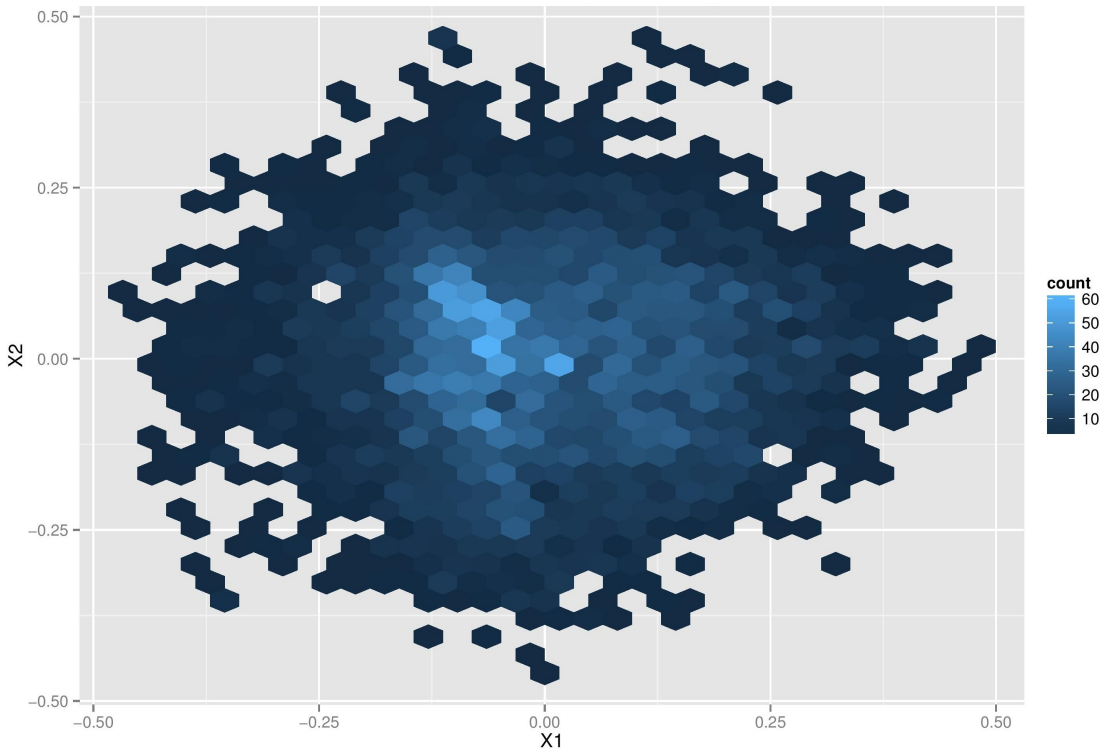

Figure 6: MDS plot with hexagonal binning.

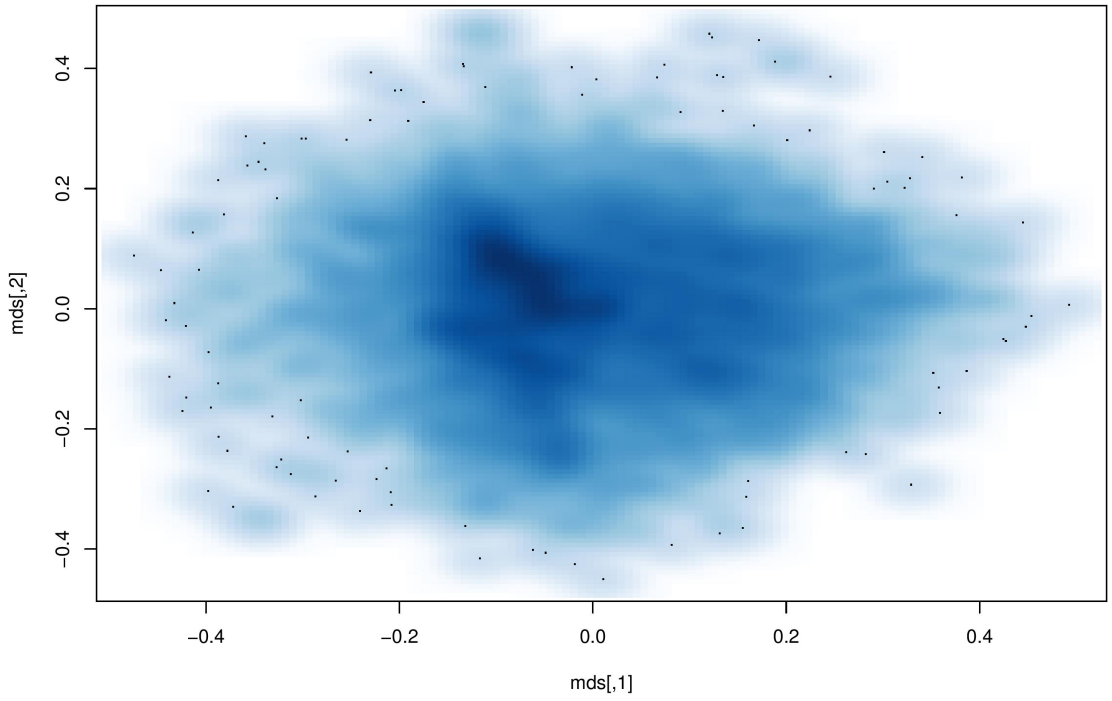

Figure 7: MDS plot with kernel smoothing.

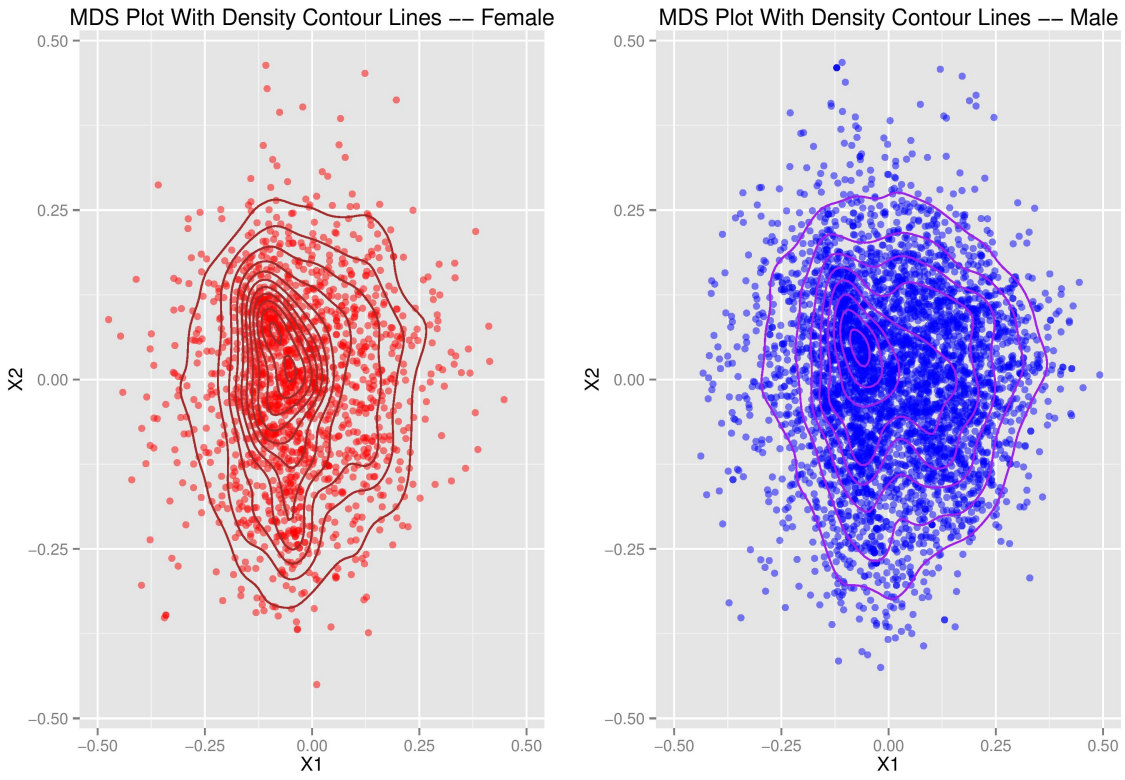

Figure 8: MDS plot by rater gender. It seems that compared to the female raters, the male raters are more widely spread and have a higher density on the right of the graph. However, this may be due to the difference between the sample sizes of males and females. $4,054$ males are in the data set, whereas the number of females is only $1,571$.

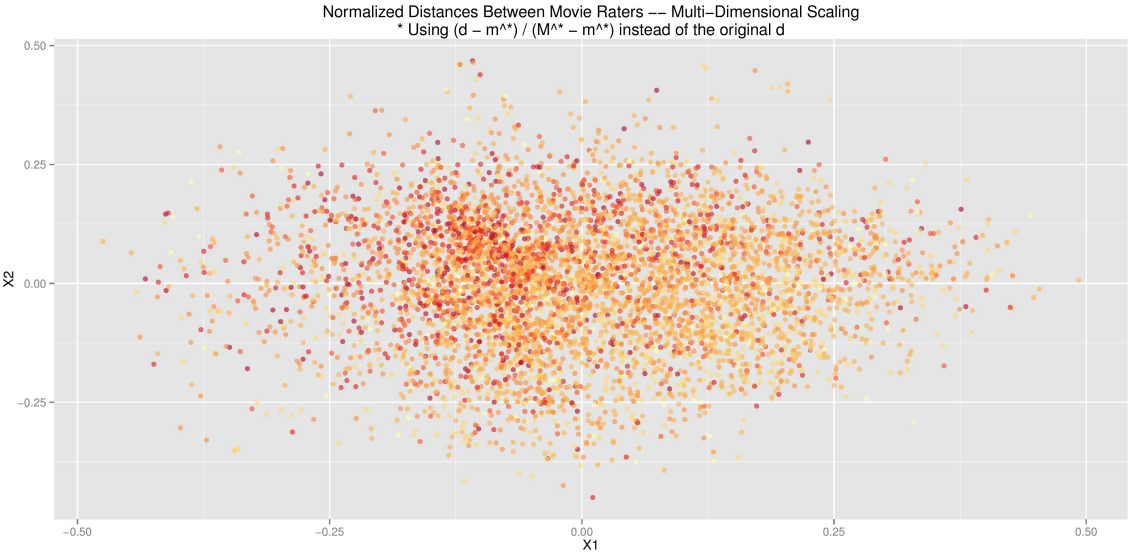

Figure 9: MDS plot by rater age. A deeper color means elder in age. On average, more of the younger raters lie on the right of the graph.

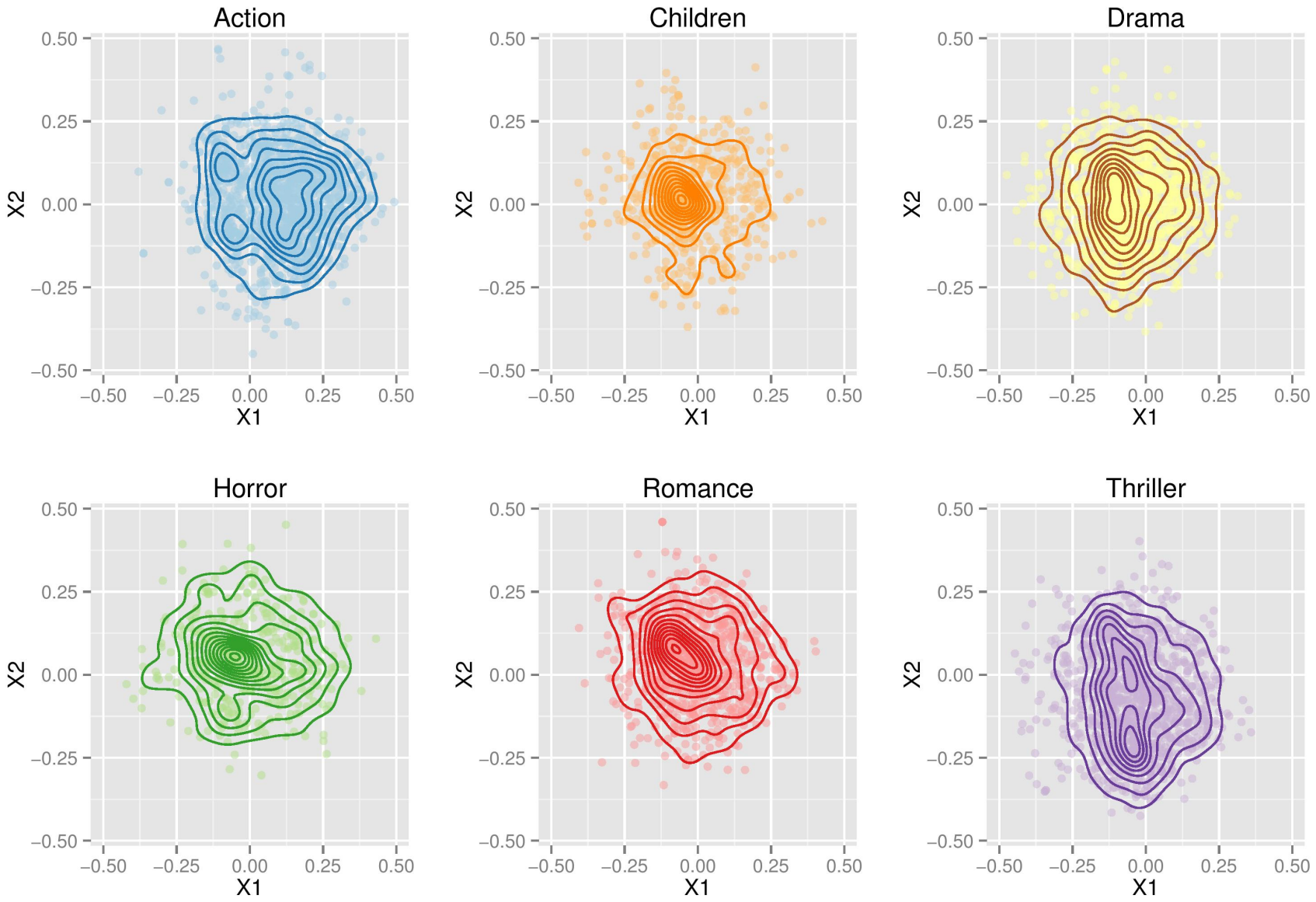

Figure 10: MDS plots with contour curves for different "fan groups".

5.3 Possible Interpretation

5.3.1 Finding the Representative Rankings for Local Maximum Density Regions

Five rectangular regions have been identified where the density reaches a local maximum (Figure 11). The rankings in our data set falling into each region are attained accordingly.

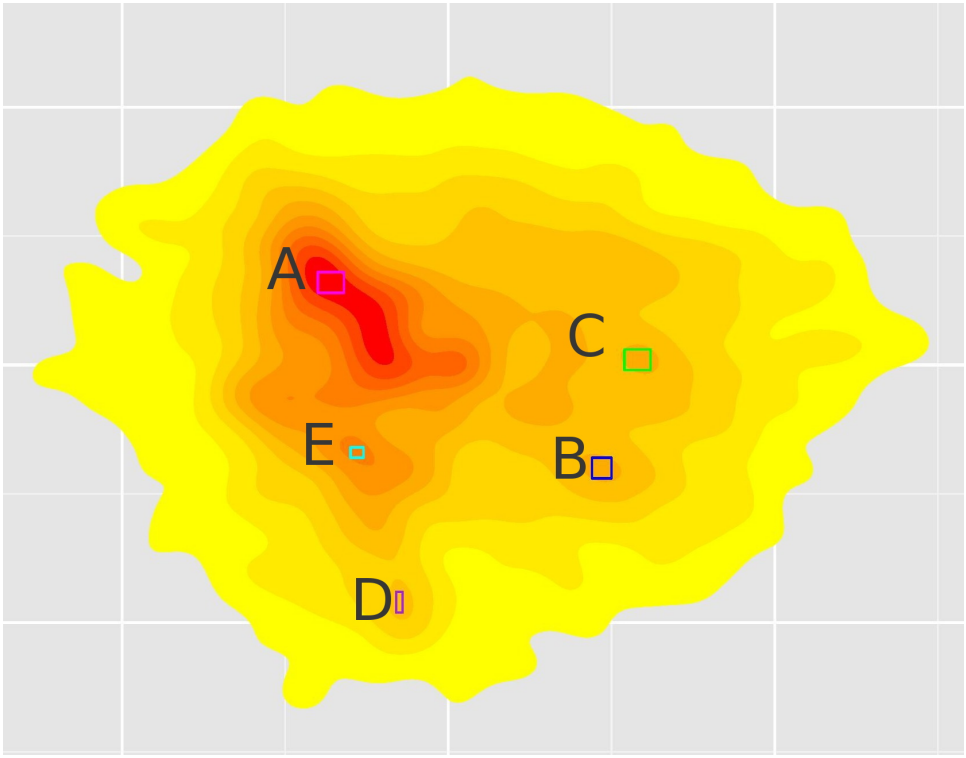

Figure 11: The five regions with local maximum density.

We formulate the problem of finding the representative rankings for the maximum density regions as a optimization problem:

> For a group G of rankings of t objects, we would like to find a complete ranking r such that the sum of the distances from r to each ranking in G is minimized.

The algorithm we have used is as follows.

1. Initialize r to be $1|2|\dots|t$, `CHANGED` to be `False`, `COUNT` to be t.
2. Decrease `COUNT` by one.
3. For i from 1 up to `COUNT`, if swapping the i^{th} and $(i+1)^{th}$ ranked objects in r would result in a lower sum of distances from r to each ranking in G, swap them and set `CHANGED` to be `True`.
4. If `COUNT` ≤ 1 or `CHANGED` $==$ `False`, return r as the representative ranking and `STOP`. Otherwise go back to 2.

Five representative rankings have been identified using the above algorithm (Figure 12).

5.3.2 Latent Meaning of the Axes

In Figure 12, we could see that in the highest density region (pink rectangle, A), the genres are rather mixed, though generally speaking, the highly rated movies by the raters in this region mainly deal with romance (e.g. *Casablanca* and *The Graduate*) and tend to have a bubbly tone (e.g. *Singin' in the Rain* and *Babe*) or high suspense (e.g. *Psycho*, *North by Northwest* and *The Godfather*). Intuitively, the movies with these features indeed appeal to the mass.

For the region in the bottom right corner (blue rectangle, B), action movies (e.g. *Star Wars* and *The Matrix*) have aced the representative rankings. The region in the upper right corner (green rectangle, C) is quite similar, with the top four rankings being all action movies.

The regions in the bottom left corner have more drama movies at the top than the regions A, B and C. Between these two, the upper one (cyan rectangle, E) has more obscure movies ranked among the top (e.g. *Seven Samurai* and *Cinema Paradiso*), whereas the lower one (magenta rectangle, D) rated movies dealing with serious issues at the top (e.g. *Schindler's List* and *To Kill a Mockingbird* – both of them are not present in the top ten list of the representative ranking of any of the other five regions).

We could see that in this plot, a dichotomy exists. The right half of the cluster appears to appreciate pure action films with sci-fi elements, while the left half lean towards miscellaneous non-sci-fi movie genres. In the left cluster, the upper part mainly consists of people interested in romance and children's movies, or movies with fancier and more exciting elements, while the people in the middle and lower part enjoy more serious drama movies.

Referring to the Figures 8 and 9, we could see that male and young raters (cluster more on the right) tend to be die-hard action fans. In terms of the fancy-serious dichotomy, there is no significant divergence among different gender and age groups.

5.4 The Goodness of Fit of 2-D Multidimensional Scaling

In general, when we perform MDS, an increment in the maximum dimension of the space in which the data are to be represented would greatly increase the goodness of fit. Our goal, however, is to explain the distance matrix in terms of fewer underlying dimensions. Therefore we could not use as many dimensions as possible, which would yield a perfect fit.

In our project, we have performed two-dimensional MDS to visualize the distance matrix. How well is our fit, then? We calculate our goodness of fit as

$$g = \frac{\sum_{j=1}^{k} \lambda(j)}{\sum_{j=1}^{n} max\{\lambda(j), 0\}}$$

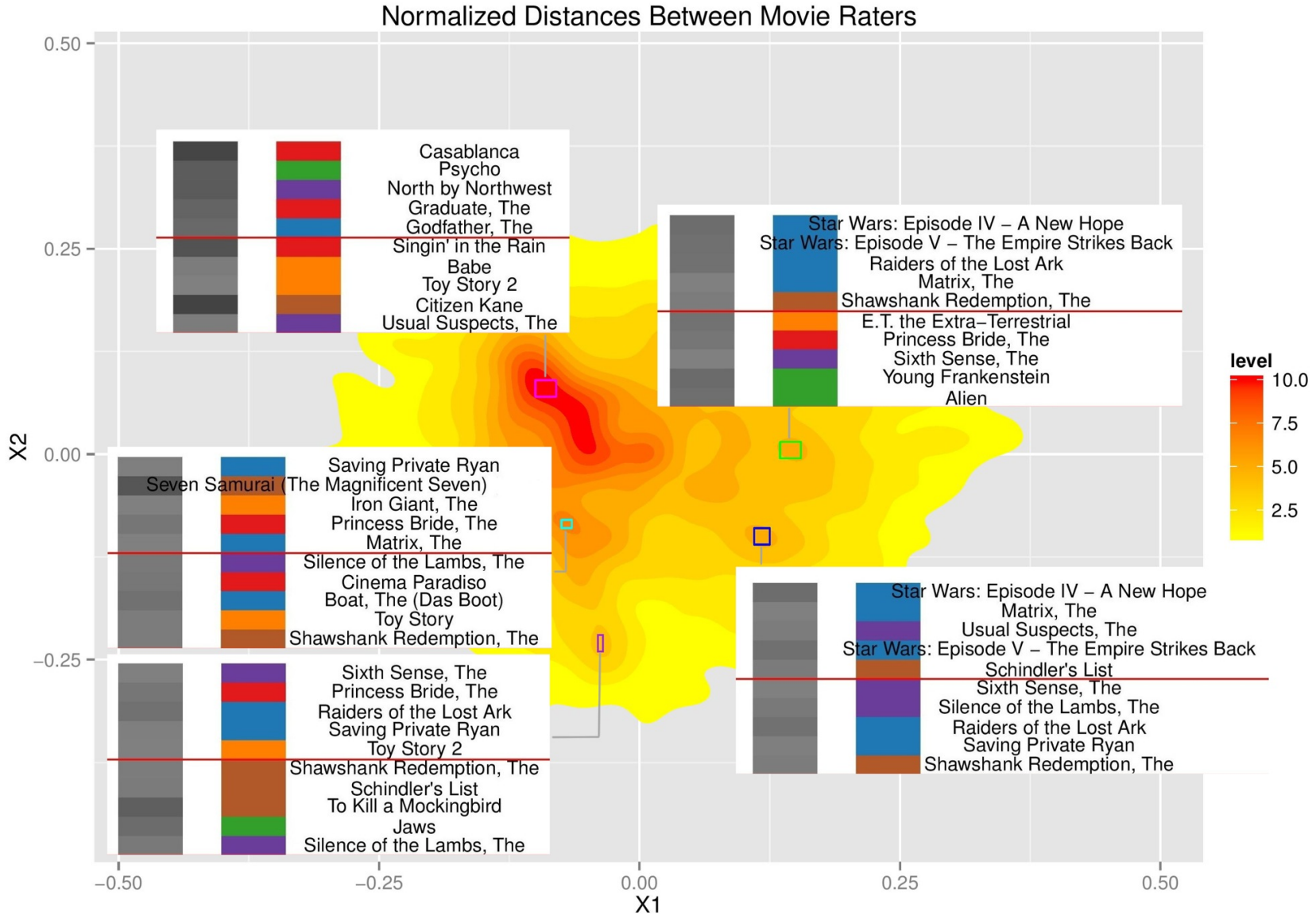

Figure 12: Representative rankings for maximum density regions. Only the top ten movies in the representative rankings are shown in this graph; the full representative rankings could be found in Appendix A. The color bar denotes the genres of the movies – red is for romance, green is for horror, purple is for thriller, blue is for action, brown is for drama and orange is for children's. The gray-scale bar stands for the year of the movie. A darker shade of gray stands for an older movie.

where $\lambda(j)$ are the eigenvalues computed during the scaling process sorted in decreasing order [17].

From Table 1, we could see that the goodness of fit is very low even when we use a 4-D space (approximately 16.74%). This may be acceptable, though, since the data we are using is based on real people, and the structure of the data may be quite complex.

Table 1: Goodness of fit of MDS (k is the number of dimensions)

k = 1	**k = 2**	**k =3**	**k = 4**
0.05088746	0.09386020	0.13226656	0.16738561

6 Discussion

Our visualization framework is three-fold. First, we introduce the normalized distance d^* for calculating the distance between different rankings. Second, we apply MDS to transform the data matrix to a set of data points (one point for each ranking), so that we could visualize the set of rankings in a two-dimensional space. Third, we apply hexagonal binning and kernel smoothing as well as calculate the representative rankings for different local maximum density regions to visualize the behavior of the raters, as demonstrated by visualizing the MovieLens data set.

The efficiency of our visualization system is guaranteed by using the shortcut formula proposed by Alvo and Cabilio [13]. However, when the ranking data set is very large and stems from social science data (as is our case), visualization in a two-dimensional space may not be sufficient for accounting for the dissimilarities between different rankings. Further work would be needed to develop a more accurate framework for visualizing big ranking data.

A The Five Representative Rankings in Full

The color bar denotes the genres of the movies – red is for romance, green is for horror, purple is for thriller, blue is for action, brown is for drama and orange is for children's. The gray-scale bar stands for the year of the movie. A darker shade of gray stands for an older movie. Please refer to Figure 11 for the relative locations of the peaks A, B, C, D and E.

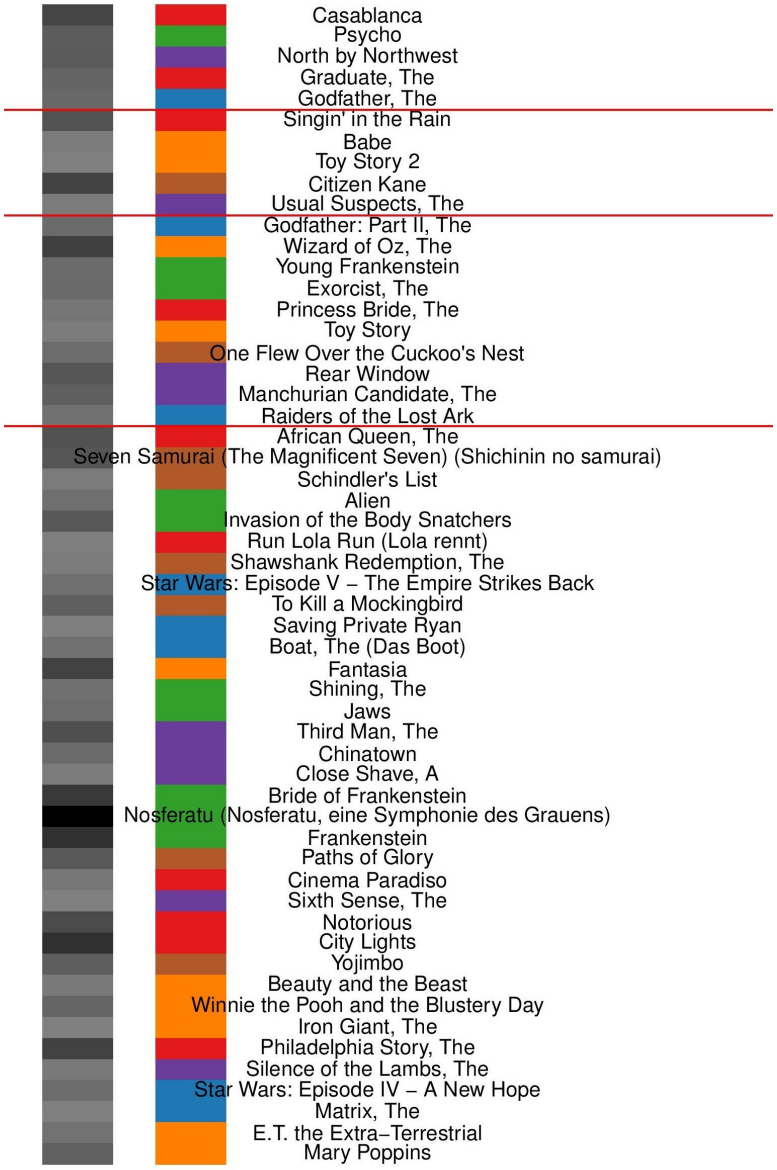

Figure 13: Peak A.

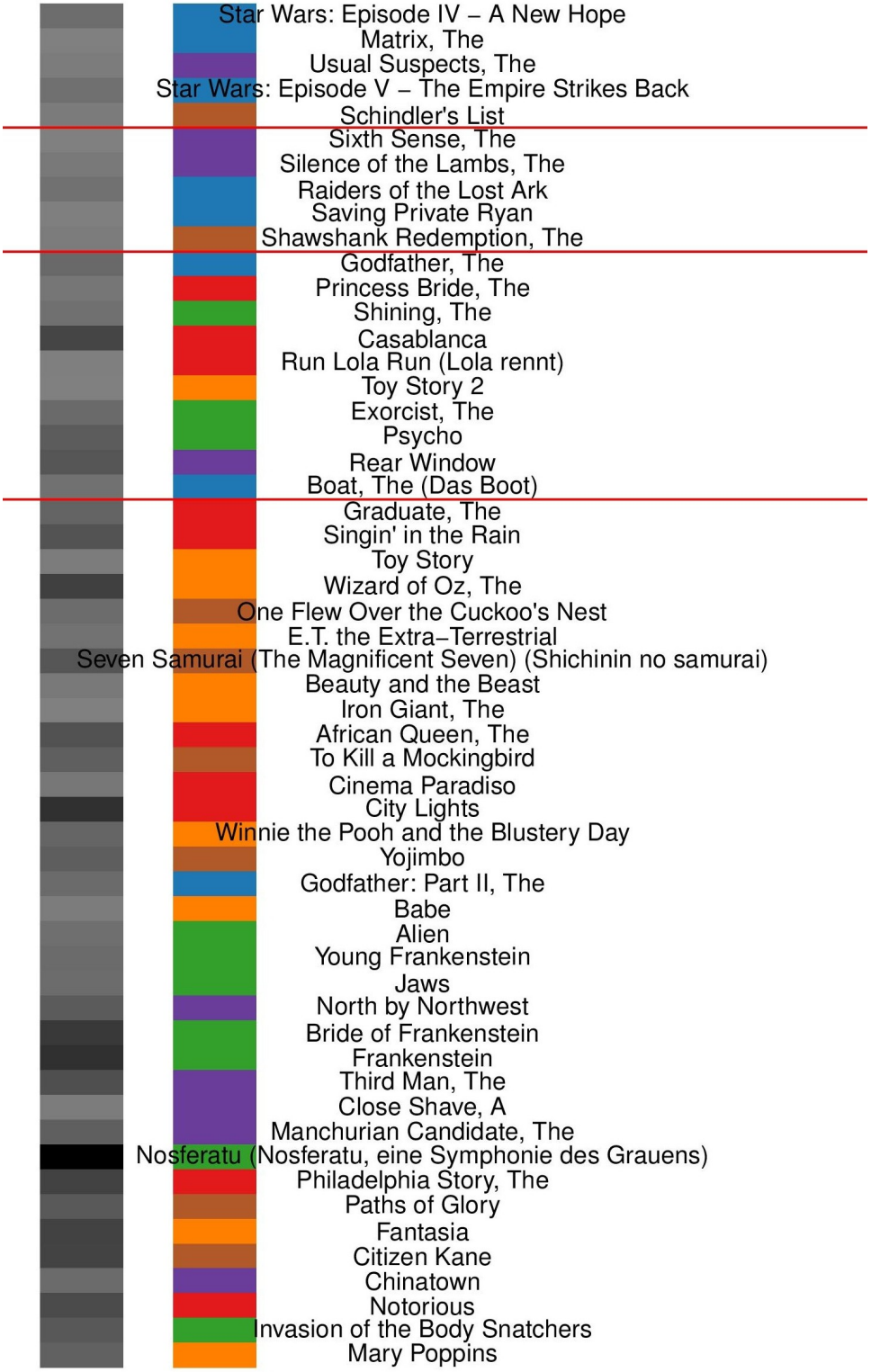

Figure 14: Peak B.

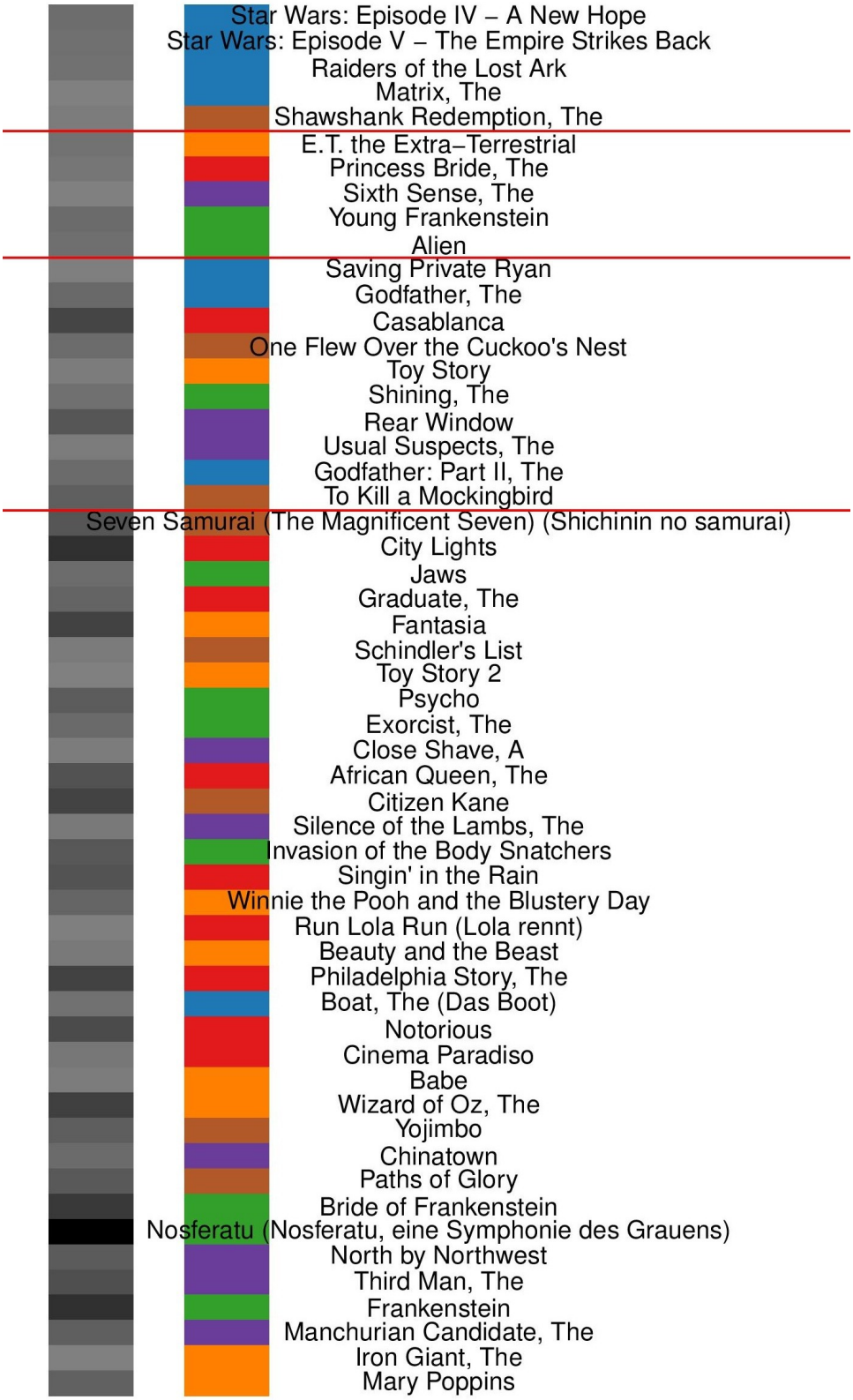

Figure 15: Peak C.

Figure 16: Peak D.

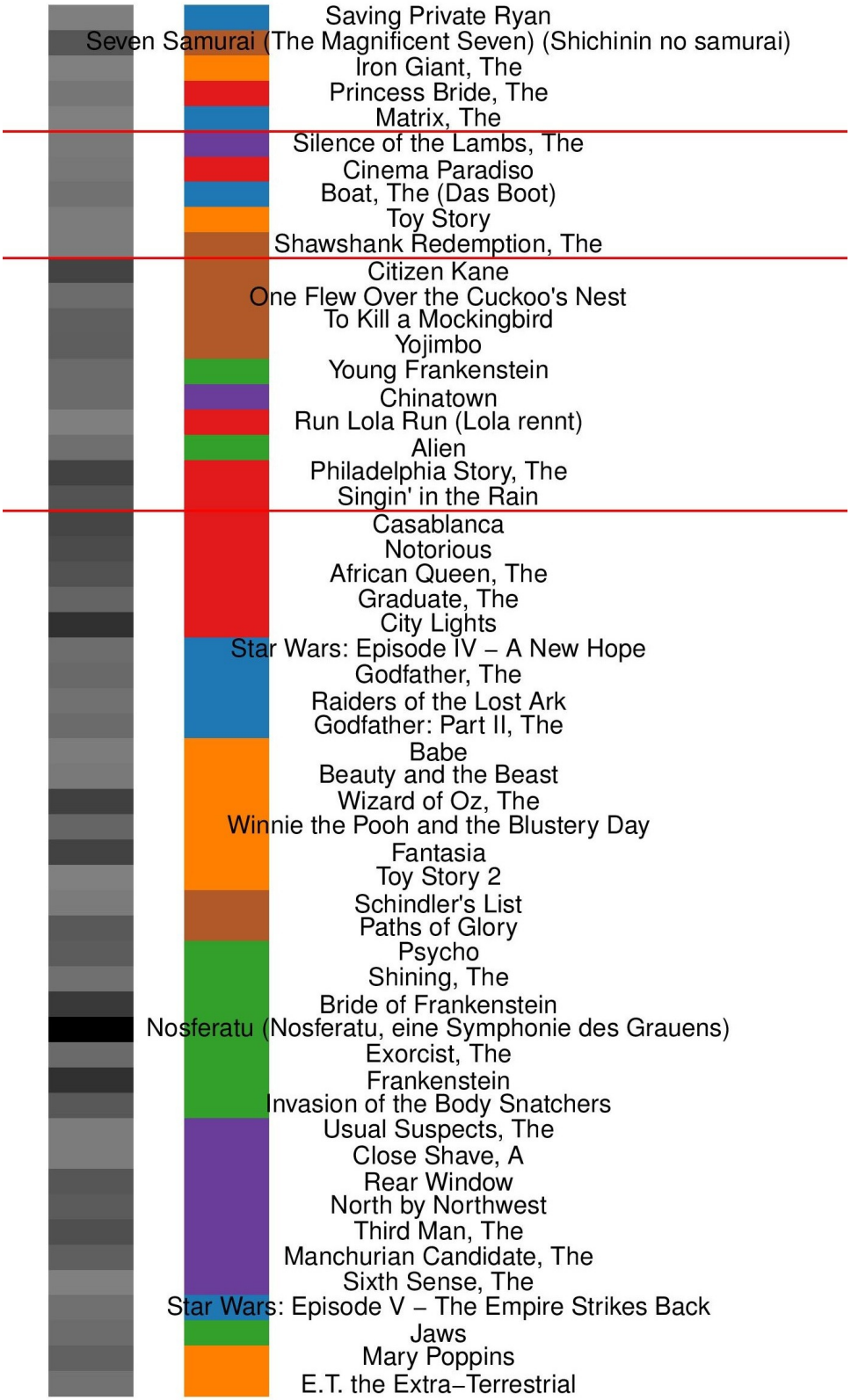

Figure 17: Peak E.

B R codes

Here are all the relevant R codes used in this project. Please refer to the comments in individual files for the corresponding descriptions.

```
#!/usr/bin/env Rscript
# Author: Yiming Li
#
# Toy example to justify our normalized distance.

library(wordcloud)

### Preliminary data / functions loading

source("ksMinMaxFuncs.R")

func_a2 <- function(r, i, j) {
  # (12) in "Rank correlation methods for missing data"
  # For a given pair of movies (i,j), 1 <= i < j <= t
  # and an incomplete ranking r, calculate the score
  k <- sum(!is.na(r))
  if (is.na(r[i]) & is.na(r[j])) {
    return(0)
  } else if (is.na(r[i]) & !is.na(r[j])) {
    return(2 * r[j] / (k + 1) - 1)
  } else if (is.na(r[j])) {
    return(1 - 2 * r[i] / (k + 1))
  } else {
    return(sign(r[j] - r[i]))
  }
}

k_dist <- function(r1, r2, c_k, t) {
  ### For two incomplete rankings, calculate the
  ### Kendall distance between them
  kdis <- c_k
  for (i in 1:(t-1)) {
    for (j in (i+1):t) {
      kdis <- kdis - func_a2(r1,i,j)*func_a2(r2,i,j)
    }
  }
  return(kdis)
}

t <- 6 # Number of rated items
c_k <- t * (t - 1) / 2 # Mean of Kendall distance

moviemat <- matrix(data = c(1,2,3,NA,NA,NA,1,3,2,NA,NA,NA,2,1,3,NA,NA,NA,2,3,1,NA,NA,
    NA,3,1,2,NA,NA,NA,3,2,1,NA,NA,NA,1,2,3,4,NA,NA,1,2,3,NA,NA,NA,1,2,3,4,5,NA
    ,1,2,3,4,5,6,6,5,4,3,2,1,1,2,3,6,5,4), byrow = TRUE, ncol = t)
raters <- nrow(moviemat)

noRatedmovies <- c(3,3,3,3,3,3,4,3,5,6,6,6)

D_k <- matrix(c_k, nrow = raters, ncol = raters)

for (p in 1:raters) {
  for (q in 1:p) {
    D_k[p,q] = k_dist(moviemat[p,],moviemat[q,],c_k,t)
    D_k[q,p] = D_k[p,q]
  }
}
# D_k is the Kendall distance matrix

```

```
D_k2 <- D_k
for (p in 1:raters) {
  for (q in 1:p) {
    D_k2[p,q] = D_k2[p,q] - min_k(p, q, noRatedmovies, c_k, c_s, t)
    D_k2[q,p] = D_k2[p,q]
  }
}
# D_k2 is another version of the Kendall distance matrix
# Here, we use (d - m^*) instead of the original d, to make d_{\pi\pi} = 0

D_k3 <- D_k2
for (p in 1:raters) {
  for (q in 1:p) {
    D_k3[p,q] = D_k3[p,q] / (max_k(p, q, noRatedmovies, c_k, c_s, t) - min_k(p, q,
        noRatedmovies, c_k, c_s, t))
    D_k3[q,p] = D_k3[p,q]
  }
}
# D_k3 is yet another version of the Kendall distance matrix
# Here, we use (d - m^*) / (M^* - m^*) instead of the original d

mds <- cmdscale(D_k)
mds2 <- cmdscale(D_k2)
mds3 <- cmdscale(D_k3)

pdf("toy.pdf", width = 10, height = 7)

mx <- apply(mds,2,max)
mn <- apply(mds,2,min)
textplot(mds[,1],mds[,2], c("1|2|3","1|3|2","2|1|3","2|3|1","3|1|2","3|2|1","1|2|3|4",
    "1|2|3","1|2|3|4|5","1|2|3|4|5|6","6|5|4|3|2|1","1|2|3|6|5|4"), xlim=c(mn[1],mx
    [1]),ylim=c(mn[2],mx[2]))

mx <- apply(mds2,2,max)
mn <- apply(mds2,2,min)
textplot(mds2[,1],mds2[,2], c("1|2|3","1|3|2","2|1|3","2|3|1","3|1|2","3|2|1","1|2|3|4
    ","1|2|3","1|2|3|4|5","1|2|3|4|5|6","6|5|4|3|2|1","1|2|3|6|5|4"), xlim=c(mn[1],mx
    [1]),ylim=c(mn[2],mx[2]))

mx <- apply(mds3,2,max)
mn <- apply(mds3,2,min)
textplot(mds3[,1],mds3[,2], c("1|2|3","1|3|2","2|1|3","2|3|1","3|1|2","3|2|1","1|2|3|4
    ","1|2|3","1|2|3|4|5","1|2|3|4|5|6","6|5|4|3|2|1","1|2|3|6|5|4"), xlim=c(mn[1],mx
    [1]),ylim=c(mn[2],mx[2]))

dev.off()
```

./codes/toyDemo.R

```
#!/usr/bin/env Rscript
#
# The dataset is available on the Web at
# http://files.grouplens.org/datasets/movielens/ml-1m.zip
#
# The whole dataset includes 1000209 ratings (1, 2, 3, 4, 5) of
# 3952 movies from 6040 users who joined MovieLens in 2000.

# Author: Yiming Li
#
# Data preparation and cleaning.

movies <- read.table("movies.txt", header = FALSE, sep = "\t", col.names = c("mid", "
    title", "year", "genres"), quote="\"", colClasses = c("integer", "character", "
    integer", "character"), stringsAsFactors = FALSE)
#> dim(movies)
#[1] 3883    4

```

```
users <- read.table("users.dat", header = FALSE, sep = "\t", col.names = c("uid", "gender", "age", "job", "zip"), colClasses = c("integer", "character", "character", "character", "character"))
#> dim(users)
#[1] 6040    5

ratings <- read.table("ratings.txt", header = FALSE, sep = "\t", col.names = c("uid", "mid", "rating", "time"), colClasses = c("integer", "integer", "character", "integer"))
#> dim(ratings)
#[1] 1000209       4

noRatedUsers <- sapply(1:nrow(movies), function(i) sum(ratings$mid == movies$mid[i]))
movies$times <- noRatedUsers

movies <- movies[which(movies$times > 200),]
rownames(movies) <- 1:nrow(movies)

### Check how many movies are of the genres --
# Action
action <- grep("Action", movies$genres)
#> length(action)
#[1] 493

# Children's
children <- grep("Children", movies$genres)
#> length(children)
#[1] 250

# Comedy
comedy <- grep("Comedy", movies$genres)
#> length(comedy)
#[1] 1162

# Drama
drama <- grep("Drama", movies$genres)
#> length(drama)
#[1] 1484

# Horror
horror <- grep("Horror", movies$genres)
#> length(horror)
#[1] 339

# Romance
romance <- grep("Romance", movies$genres)
#> length(romance)
#[1] 459

# Sci-fi
scifi <- grep("Sci-Fi", movies$genres)
#> length(scifi)
#[1] 274

# Thriller
thriller <- grep("Thriller", movies$genres)
#> length(thriller)
#[1] 484

# Check the overlapping of different pairs of genres.
#> length(intersect(action,scifi))
#[1] 107
#> length(intersect(action,thriller))
#[1] 130
#> length(intersect(action,romance))
#[1] 35
#> length(intersect(drama,romance))
#[1] 201
```

```
#> length(intersect(drama,action))
#[1] 95
#> length(intersect(drama,horror))
#[1] 11
#> length(intersect(action,horror))
#[1] 25
#> length(intersect(romance,horror))
#[1] 3
#> length(intersect(drama,thriller))
#[1] 104
#> length(intersect(drama,children))
#[1] 27
#> length(intersect(comedy,children))
#[1] 93

### Select the 10 highest rated movies from each of the following genres --
###   Action, children's, drama, horror, romance, thriller

avgStar <- sapply(1:nrow(movies), function(i) mean(as.integer(ratings[ratings$mid ==
    movies$mid[i],]$rating)))
# avgStar[i] is the average stars of the i^th movie in movies
movies$star <- avgStar

romTop10 <- intersect(which(avgStar >= sort(avgStar[romance], decreasing=TRUE)[10]),
    romance)
# [1]  287  288  298  311  327  401  414  456 1001 1232
actTop10 <- intersect(which(avgStar >= sort(avgStar[action], decreasing=TRUE)[10]),
    action)
# [1]  94 282 413 414 415 435 445 742 748 966
chiTop10 <- intersect(which(avgStar >= sort(avgStar[children], decreasing=TRUE)[10]),
    children)
# [1]    1   18  228  303  344  347  382  488 1033 1164
draTop10 <- intersect(which(avgStar >= sort(avgStar[drama], decreasing=TRUE)[10]),
    drama)
# [1]  116  200  282  298  307  403  411  423  742 1124
horrTop10 <- intersect(which(avgStar >= sort(avgStar[horror], decreasing=TRUE)[10]),
    horror)
# [1] 429 433 466 485 517 523 544 727 986 989
thriTop10 <- intersect(which(avgStar >= sort(avgStar[thriller], decreasing=TRUE)[10]),
     thriller)
# [1]   27  226  258  292  294  427  461  475  966 1034

genre <- rep(NA, nrow(movies))
genre[thriTop10] <- "thriller"
genre[actTop10] <- "action"
genre[chiTop10] <- "children"
genre[draTop10] <- "drama"
genre[282] <- "action"
genre[romTop10] <- "romance"
genre[horrTop10] <- "horror"

#> intersect(romTop10, draTop10)
#[1] 298
#> which(sort(avgStar[draTop10], decreasing=TRUE) == movies$star[298])
#[1] 7
#> which(sort(avgStar[romTop10], decreasing=TRUE) == movies$star[298])
#[1] 1
### Mark 298 as romance

#> intersect(actTop10, draTop10)
#[1] 282 742
#> which(sort(avgStar[draTop10], decreasing=TRUE) == movies$star[282])
#[1] 3
#> which(sort(avgStar[draTop10], decreasing=TRUE) == movies$star[742])
#[1] 1
#> which(sort(avgStar[actTop10], decreasing=TRUE) == movies$star[282])
#[1] 2
#> which(sort(avgStar[actTop10], decreasing=TRUE) == movies$star[742])
```

```
#[1] 1
### Mark 282 as action and 742 as drama

#> intersect(actTop10, thriTop10)
#[1] 966
#> which(sort(avgStar[actTop10], decreasing=TRUE) == movies$star[966])
#[1] 7
#> which(sort(avgStar[thriTop10], decreasing=TRUE) == movies$star[966])
#[1] 10
### Mark 966 as action

#> intersect(romTop10, actTop10)
#[1] 414
#> which(sort(avgStar[actTop10], decreasing=TRUE) == movies$star[414])
#[1] 8
#> which(sort(avgStar[romTop10], decreasing=TRUE) == movies$star[414])
#[1] 3
### Mark 414 as romance

movies$genre <- genre

movies <- movies[union(union(union(union(union(romTop10, actTop10), chiTop10),
    draTop10), horrTop10), thriTop10),]
rownames(movies) <- 1:nrow(movies)
# 55 movies altogether, including
#> sum(movies$genre == "romance")
#[1] 10
#> sum(movies$genre == "action")
#[1] 8
#> sum(movies$genre == "horror")
#[1] 10
#> sum(movies$genre == "drama")
#[1] 8
#> sum(movies$genre == "thriller")
#[1] 9
#> sum(movies$genre == "children")
#[1] 10

ratings <- ratings[ratings$mid %in% movies$mid,]
#> dim(ratings)
#[1] 73286      4

noRatedmovies <- sapply(1:nrow(users), function(i) sum(ratings$uid == users$uid[i]))

users <- users[which(noRatedmovies > 1),]
rownames(users) <- 1:nrow(users)

ratings <- ratings[ratings$uid %in% users$uid,]

# After dropping the users who rated 0 or 1 movie...
#
#> dim(ratings)
#[1] 72979      4
#> length(unique(ratings$mid))
#[1] 55
#> length(unique(ratings$uid))
#[1] 5625
#> dim(users)
#[1] 5625    5
#> dim(movies)
#[1] 55  7

save(list = ls(all = TRUE), file = "movie.RData")
```

./codes/movieDataCleaning.R

```
#!/usr/bin/env Rscript
# Author: Yiming Li
```

```
# Four functions for calculating the minimum / maximum
# values of the Kendall / Spearman distance.
#
# !!! Needs the global variables c_k, c_s, and t

min_k <- function(i, j, noRated, c_k, c_s, t) {
  # Calculate the minimum value of the Kendall distance
  # between the joke ranking of the i^th and j^th ranker
  k1 <- noRated[i]
  k2 <- noRated[j]
  if (k1 > k2) {
    tmp <- k2
    k2 <- k1
    k1 <- tmp
  }
  if (k1 %% 2 == 0) {
    return(c_k - k1*(3*k1*k2*(t+1)-(k1^2+2)*(t-k2)-3*(k2+1))/(6*(k1+1)*(k2+1)))
  } else {
    return(c_k - (k1-1)*(t*(3*k2-k1)+k2*(k1+3))/(6*(k2+1)))
  }
}

max_k <- function(i, j, noRated, c_k, c_s, t) {
  # Calculate the maximum value of the Kendall distance
  # between the joke ranking of the i^th and j^th ranker
  k1 <- noRated[i]
  k2 <- noRated[j]
  if (k1 > k2) {
    tmp <- k2
    k2 <- k1
    k1 <- tmp
  }
  if (k1 %% 2 == 0) {
    return(c_k + k1*(3*k1*k2*(t+1)-(k1^2+2)*(t-k2)-3*(k2+1))/(6*(k1+1)*(k2+1)))
  } else {
    return(c_k + (k1-1)*(t*(3*k2-k1)+k2*(k1+3))/(6*(k2+1)))
  }
}

min_s <- function(i, j, noRated, c_k, c_s, t) {
  # Calculate the minimum value of the Spearman distance
  # between the joke ranking of the i^th and j^th ranker
  k1 <- noRated[i]
  k2 <- noRated[j]
  if (k1 > k2) {
    tmp <- k2
    k2 <- k1
    k1 <- tmp
  }
  if (k1 %% 2 == 0) {
    return(c_s - (t+1)^2*k1*(k1*(3*k2-k1)-2)/(24*(k2+1)*(k2+1)))
  } else {
    return(c_s - (t+1)^2*(k1-1)*(3*k2-k1)/(24*(k2+1)))
  }
}

max_s <- function(i, j, noRated, c_k, c_s, t) {
  # Calculate the maximum value of the Spearman distance
  # between the joke ranking of the i^th and j^th ranker
  k1 <- noRated[i]
  k2 <- noRated[j]
  if (k1 > k2) {
    tmp <- k2
    k2 <- k1
    k1 <- tmp
  }
  if (k1 %% 2 == 0) {
    return(c_s + (t+1)^2*k1*(k1*(3*k2-k1)-2)/(24*(k2+1)*(k2+1)))
```

```
  } else {
    return(c_s + (t+1)^2*(k1-1)*(3*k2-k1)/(24*(k2+1)))
  }
}
```

./codes/ksMinMaxFuncs.R

```
#!/usr/bin/env Rscript
#
# The dataset is available on the Web at
# http://files.grouplens.org/datasets/movielens/ml-1m.zip
#
# The whole dataset includes 1000209 ratings (1, 2, 3, 4, 5) of
# 3955 movies from 6040 users who joined MovieLens in 2000.

# Author: Yiming Li
#
# Calculate the distance between raters.
# This script takes a LOT of time to run and eats up RAM!

load("movie.RData")
rm(list = ls()[!ls() %in% c("users", "ratings", "movies")])
# We are only using the data frames ratings, users and movies
# in the analysis below (5625 users and 55 movies)

movies <- movies[,!(names(movies) %in% c("genres"))]
moviemat <- matrix(data = NA, nrow = nrow(users), ncol = nrow(movies))
rownames(moviemat) <- users$uid
colnames(moviemat) <- movies$mid

noRatedmovies <- sapply(1:nrow(users), function(i) sum(ratings$uid == users$uid[i]))
# noRatedmovies is a numeric vector of length 5625
# noRatedmovies[i] is the number of movies rated by the user with uid
# users$uid[i]

t <- ncol(moviemat) # 55 movies altogether
raters <- nrow(moviemat) # 5625 users altogether

percent <- function(x, digits = 2, format = "f") {
  paste(formatC(100 * x, format = format, digits = digits), "%", sep = "")
}

for (i in 1:raters) {
  for (j in 1:t) {
    tmp <- ratings$rating[which(ratings$uid == users$uid[i] & ratings$mid == movies$
        mid[j])]
    if (length(tmp) != 0) moviemat[i,j] <- as.numeric(tmp)
  }
  cat(c("\n", format(Sys.time(), "%H:%M:%S>"), "i =", i, "###", percent(i/5625), "
      finished!\n"))
}
# > dim(moviemat)
# [1] 5625   55

rankValues <- function(x) {
  # Rank 1 is the most preferred (given the highest score)
  # NA would be ranked NA
  # e.g. rankValues(c(1,4200,0,100,NA,-9.99,100)) returns
  # [1]  4  1  5  2 NA  6  2
  return(rank(-x, na.last = "keep", ties.method = "min"))
}
moviemat <- t(matrix(apply(moviemat, 1, rankValues), nrow = t))
# Now moviemat is a 5625*55 matrix of integers, with the i-th row containing
# the ranking of movies by the i-th user
# As above, we use NA to represent missing data

c_s <- t * (t^2 - 1) / 12 # Mean of Spearman distance
c_k <- t * (t - 1) / 2 # Mean of Kendall distance
```

```
60
61 func_a <- function(moviemat, i, j, p) {
62   # (12) in "Rank correlation methods for missing data"
63   # For a given pair of movies (i,j), 1 <= i < j <= t
64   # and the p^th person, calculate the score
65   if (is.na(moviemat[p,i]) & is.na(moviemat[p,j])) {
66     return(0)
67   } else if (is.na(moviemat[p,i]) & !is.na(moviemat[p,j])) {
68     return(2 * moviemat[p,j] / (noRatedmovies[p] + 1) - 1)
69   } else if (is.na(moviemat[p,j])) {
70     return(1 - 2 * moviemat[p,i] / (noRatedmovies[p] + 1))
71   } else {
72     return(sign(moviemat[p,j] - moviemat[p,i]))
73   }
74
75 }
76
77 # Initialized to be all means
78 D_k <- matrix(c_k, nrow = raters, ncol = raters)
79
80 for (p in 1:raters) {
81   for (q in 1:p) {
82     for (i in 1:(t-1)) {
83       for (j in (i+1):t) {
84         D_k[p,q] = D_k[p,q] - func_a(moviemat, i,j,p)*func_a(moviemat, i,j,q)
85         if (p %% 5 == 0 && q == p && i == 54 && j == 55) {
86           cat(c("\n[1st]", format(Sys.time(), "%H:%M:%S>"), "p =", p, "q =", q, "i =",
                 i, "j =", j, "###", percent(p/5625), "finished!\n"))
87         }
88         # Last loop: p = 5625, q = 5625, i = 54, j = 55
89       }
90     }
91     D_k[q,p] = D_k[p,q]
92   }
93 }
94 # D_k is the Kendall distance matrix, of dimension 5625 * 5625
95
96 source("ksMinMaxFuncs.R")
97 # Now we have the functions min_k, max_k, min_s, max_s
98 # for calculating the minimum / maximum values of the Kendall / Spearman distance.
99
100 D_k2 <- D_k
101 for (p in 1:raters) {
102   for (q in 1:p) {
103     D_k2[p,q] = D_k2[p,q] - min_k(p, q, noRatedmovies, c_k, c_s, t)
104     D_k2[q,p] = D_k2[p,q]
105     if (p %% 5 == 0 && q == p) {
106       cat(c("\n[2nd]", format(Sys.time(), "%H:%M:%S>"), "p =", p, "q =", q, "###",
              percent(p/5625), "finished!\n"))
107     }
108   }
109 }
110 # D_k2 is another version of the Kendall distance matrix, of dimension 5625 * 5625
111 # Here, we use (d - m^*) instead of the original d, to make d_{\pi\pi} = 0
112
113 D_k3 <- D_k2
114 for (p in 1:raters) {
115   for (q in 1:p) {
116     D_k3[p,q] = D_k3[p,q] / (max_k(p, q, noRatedmovies, c_k, c_s, t) - min_k(p, q,
          noRatedmovies, c_k, c_s, t))
117     D_k3[q,p] = D_k3[p,q]
118     if (p %% 5 == 0 && q == p) {
119       cat(c("\n[3rd]", format(Sys.time(), "%H:%M:%S>"), "p =", p, "q =", q, "###",
              percent(p/5625), "finished!\n"))
120     }
121   }
122 }
123 # D_k3 is yet another version of the Kendall distance matrix, of dimension 5625 * 5625
```

```
# Here, we use (d - m^*) / (M^* - m^*) instead of the original d

mds <- cmdscale(D_k)
mds2 <- cmdscale(D_k2)
mds3 <- cmdscale(D_k3)

save(list = ls(all = TRUE), file = "movie2.RData")
```

./codes/movieDistanceCalc.R

```
#!/usr/bin/env Rscript
# Author: Yiming Li
# 1. Make the hexagonal binning MDS plot.
# 2. Make MDS plots with contour curves colored by genders.
# 3. Make the kernel smoothing MDS plot.
#
# Requires loading the packages specified in movieVisualization.R!

plotByGender <- function(users, mds, xl, yl) {
  mds.female <- data.frame(mds[which(users$gender == "F"),])
  mds.male <- data.frame(mds[which(users$gender == "M"),])

  mds.hex <- ggplot(data.frame(mds),aes(x=X1,y=X2)) + stat_binhex() + ggtitle("MDS
      Plot With Hexagonal Binning")

  mds.cont <- ggplot(data.frame(mds),aes(x=X1,y=X2)) + geom_point(alpha = 0.5) + geom_
      density2d() + ggtitle("MDS Plot With Density Contour Lines")

  pfemale <- ggplot(mds.female,aes(x=X1,y=X2)) + geom_point(alpha = 0.5, col = "red")
      + geom_density2d(col = "brown") + xlim(-xl, xl) + ylim(-yl, yl) + ggtitle("MDS
      Plot With Density Contour Lines -- Female")

  pmale <- ggplot(mds.male,aes(x=X1,y=X2)) + geom_point(alpha = 0.5, col = "blue") +
      geom_density2d(col = "purple") + xlim(-xl, xl) + ylim(-yl, yl) + ggtitle("MDS
      Plot With Density Contour Lines -- Male")

  mds.both <- ggplot(mds.female,aes(x=X1,y=X2)) + geom_point(alpha = 0.3, col = "red")
       + geom_density2d(col = "red", alpha = 0.8, lwd = 0.8) + geom_point(data = mds.
      male, alpha = 0.3, col = "blue") + geom_density2d(data = mds.male, col = "blue",
       alpha = 0.8, lwd = 0.8) + ggtitle("MDS Plot With Density Contour Lines\nRed --
      Female; Blue -- Male")

  grid.arrange(mds.hex)
  grid.arrange(mds.cont)
  grid.arrange(mds.both)
  grid.arrange(pfemale, pmale, ncol = 2)

  smoothScatter(mds)
}

pdf("bygender2.pdf", width = 10, height = 7)

plotByGender(users, mds3, max(abs(range(mds3[,1]))), max(abs(range(mds3[,2]))))

dev.off()
```

./codes/hex+smooth+compareGender.R

```
#!/usr/bin/env Rscript
# Author: Yiming Li
#
# Color the MDS plot by age groups
# Requires loading the packages specified in movieVisualization.R!

pdf("byage.pdf", width = 14, height = 7)
pal <- brewer.pal(7, "YlOrRd")
# Blues BuGn BuPu GnBu Greens Greys Oranges
```

```
# OrRd PuBu PuBuGn PuRd Purples RdPu Reds
# YlGn YlGnBu YlOrBr YlOrRd

# grays <- gray.colors(7, start = 0, end = 0.5, gamma = 2.2)
agecol <- ifelse(users$age == "1", pal[1], ifelse(users$age == "18", pal[2], ifelse(
    users$age == "25", pal[3], ifelse(users$age == "35", pal[4], ifelse(users$age == "
    45", pal[5], ifelse(users$age == "50", pal[6], pal[7])))))

age3 <- ggplot(data.frame(mds3),aes(x=X1,y=X2)) + geom_point(col = agecol, alpha =
    0.6) + ggtitle("Normalized Distances Between Movie Raters -- Multi-Dimensional
    Scaling\n* Using (d - m^*) / (M^* - m^*) instead of the original d")

grid.arrange(age3)

### Non-ggplot version
plot(mds3, main = "Normalized Distances Between Movie Raters -- Multi-Dimensional
    Scaling\n* Using (d - m^*) / (M^* - m^*) instead of the original d", xlab = "
    Coordinate 1", ylab = "Coordinate 2", col = agecol, pch = "+")

dev.off()
```

./codes/compareAge.R

```
#!/usr/bin/env Rscript
# Author: Yiming Li
# Demonstrates the relationship between jobs and movie preferences.
# Results not shown in the report.
#
# Requires loading the packages specified in movieVisualization.R!

pdf("byjob2.pdf", width = 14, height = 7)

totJobs <- c(433, 219, 145, 645, 93, 177, 544, 13, 71, 161, 113, 320, 104, 250, 124,
    197, 406, 56, 63, 247, 0)

xl <- c(-max(abs(range(mds3[,1]))), max(abs(range(mds3[,1]))))
yl <- c(-max(abs(range(mds3[,2]))), max(abs(range(mds3[,2]))))

plot(mds3[which(users$job == "5"),], main = "Student", xlab = "Coordinate 1", ylab = "
    Coordinate 2", pch = "+", col = cols[1], xlim = xl, ylim = yl)

plot(mds3[which(users$job == "5"),], main = "Student + Artist", xlab = "Coordinate 1",
     ylab = "Coordinate 2", pch = "+", col = cols[1], xlim = xl, ylim = yl)
points(mds3[which(users$job == "2"),], pch = "+", col = cols[2], xlim = xl, ylim = yl)

plot(mds3[which(users$job == "5"),], main = "Student + Artist + Engineer", xlab = "
    Coordinate 1", ylab = "Coordinate 2", pch = "+", col = cols[1], xlim = xl, ylim =
    yl)
points(mds3[which(users$job == "2"),], pch = "+", col = cols[2], xlim = xl, ylim = yl)
points(mds3[which(users$job == "17"),], pch = "+", col = cols[3], xlim = xl, ylim = yl
    )

plot(mds3[which(users$job == "5"),], main = "Student + Artist + Engineer + Farmer",
    xlab = "Coordinate 1", ylab = "Coordinate 2", pch = "+", col = cols[1], xlim = xl,
     ylim = yl)
points(mds3[which(users$job == "2"),], pch = "+", col = cols[2], xlim = xl, ylim = yl)
points(mds3[which(users$job == "17"),], pch = "+", col = cols[3], xlim = xl, ylim = yl
    )
points(mds3[which(users$job == "8"),], pch = "+", col = cols[4], xlim = xl, ylim = yl)

plot(mds3[which(users$job == "5"),], main = "Student + Artist + Engineer + Farmer +
    Lawyer", xlab = "Coordinate 1", ylab = "Coordinate 2", pch = "+", col = cols[1],
    xlim = xl, ylim = yl)
points(mds3[which(users$job == "2"),], pch = "+", col = cols[2], xlim = xl, ylim = yl)
points(mds3[which(users$job == "17"),], pch = "+", col = cols[3], xlim = xl, ylim = yl
    )
points(mds3[which(users$job == "8"),], pch = "+", col = cols[4], xlim = xl, ylim = yl)
```

```
33 points(mds3[which(users$job == "11"),], pch = "+", col = cols[5], xlim = xl, ylim = yl
       )
34
35 plot(mds3[which(users$job == "5"),], main = "Student + Artist + Engineer + Farmer +
       Lawyer + Clerical", xlab = "Coordinate 1", ylab = "Coordinate 2", pch = "+", col =
        cols[1], xlim = xl, ylim = yl)
36 points(mds3[which(users$job == "2"),], pch = "+", col = cols[2], xlim = xl, ylim = yl)
37 points(mds3[which(users$job == "17"),], pch = "+", col = cols[3], xlim = xl, ylim = yl
       )
38 points(mds3[which(users$job == "8"),], pch = "+", col = cols[4], xlim = xl, ylim = yl)
39 points(mds3[which(users$job == "11"),], pch = "+", col = cols[5], xlim = xl, ylim = yl
       )
40 points(mds3[which(users$job == "3"),], pch = "+", col = cols[6], xlim = xl, ylim = yl)
41
42 plot(mds3[which(users$job == "5"),], main = "Student + Artist + Engineer + Farmer +
       Lawyer + Clerical + Doctor", xlab = "Coordinate 1", ylab = "Coordinate 2", pch = "
       +", col = cols[1], xlim = xl, ylim = yl)
43 points(mds3[which(users$job == "2"),], pch = "+", col = cols[2], xlim = xl, ylim = yl)
44 points(mds3[which(users$job == "17"),], pch = "+", col = cols[3], xlim = xl, ylim = yl
       )
45 points(mds3[which(users$job == "8"),], pch = "+", col = cols[4], xlim = xl, ylim = yl)
46 points(mds3[which(users$job == "11"),], pch = "+", col = cols[5], xlim = xl, ylim = yl
       )
47 points(mds3[which(users$job == "3"),], pch = "+", col = cols[6], xlim = xl, ylim = yl)
48 points(mds3[which(users$job == "6"),], pch = "+", col = cols[7], xlim = xl, ylim = yl)
49
50 plot(mds3[which(users$job == "5"),], main = "Student + Artist + Engineer + Farmer +
       Lawyer + Clerical + Doctor + Manager", xlab = "Coordinate 1", ylab = "Coordinate 2
       ", pch = "+", col = cols[1], xlim = xl, ylim = yl)
51 points(mds3[which(users$job == "2"),], pch = "+", col = cols[2], xlim = xl, ylim = yl)
52 points(mds3[which(users$job == "17"),], pch = "+", col = cols[3], xlim = xl, ylim = yl
       )
53 points(mds3[which(users$job == "8"),], pch = "+", col = cols[4], xlim = xl, ylim = yl)
54 points(mds3[which(users$job == "11"),], pch = "+", col = cols[5], xlim = xl, ylim = yl
       )
55 points(mds3[which(users$job == "3"),], pch = "+", col = cols[6], xlim = xl, ylim = yl)
56 points(mds3[which(users$job == "6"),], pch = "+", col = cols[7], xlim = xl, ylim = yl)
57 points(mds3[which(users$job == "7"),], pch = "+", col = cols[8], xlim = xl, ylim = yl)
58
59 plot(mds3[which(users$job == "5"),], main = "Student + Artist + Engineer + Farmer +
       Lawyer + Clerical + Doctor + Manager + Sales", xlab = "Coordinate 1", ylab = "
       Coordinate 2", pch = "+", col = cols[1], xlim = xl, ylim = yl)
60 points(mds3[which(users$job == "2"),], pch = "+", col = cols[2], xlim = xl, ylim = yl)
61 points(mds3[which(users$job == "17"),], pch = "+", col = cols[3], xlim = xl, ylim = yl
       )
62 points(mds3[which(users$job == "8"),], pch = "+", col = cols[4], xlim = xl, ylim = yl)
63 points(mds3[which(users$job == "11"),], pch = "+", col = cols[5], xlim = xl, ylim = yl
       )
64 points(mds3[which(users$job == "3"),], pch = "+", col = cols[6], xlim = xl, ylim = yl)
65 points(mds3[which(users$job == "6"),], pch = "+", col = cols[7], xlim = xl, ylim = yl)
66 points(mds3[which(users$job == "7"),], pch = "+", col = cols[8], xlim = xl, ylim = yl)
67 points(mds3[which(users$job == "14"),], pch = "+", col = cols[9], xlim = xl, ylim = yl
       )
68
69 plot(mds3[which(users$job == "5"),], main = "Student + Artist + Engineer + Farmer +
       Lawyer + Clerical + Doctor + Manager + Sales + Retired", xlab = "Coordinate 1",
       ylab = "Coordinate 2", pch = "+", col = cols[1], xlim = xl, ylim = yl)
70 points(mds3[which(users$job == "2"),], pch = "+", col = cols[2], xlim = xl, ylim = yl)
71 points(mds3[which(users$job == "17"),], pch = "+", col = cols[3], xlim = xl, ylim = yl
       )
72 points(mds3[which(users$job == "8"),], pch = "+", col = cols[4], xlim = xl, ylim = yl)
73 points(mds3[which(users$job == "11"),], pch = "+", col = cols[5], xlim = xl, ylim = yl
       )
74 points(mds3[which(users$job == "3"),], pch = "+", col = cols[6], xlim = xl, ylim = yl)
75 points(mds3[which(users$job == "6"),], pch = "+", col = cols[7], xlim = xl, ylim = yl)
76 points(mds3[which(users$job == "7"),], pch = "+", col = cols[8], xlim = xl, ylim = yl)
77 points(mds3[which(users$job == "14"),], pch = "+", col = cols[9], xlim = xl, ylim = yl
       )
```

```
78 points(mds3[which(users$job == "13"),], pch = "+", col = cols[10], xlim = xl, ylim =
       yl)
79
80 dev.off()
```

./codes/compareJob.R

```
 1 #!/usr/bin/env Rscript
 2 #
 3 # The dataset is available on the Web at
 4 # http://files.grouplens.org/datasets/movielens/ml-1m.zip
 5 #
 6 # The whole dataset includes 1000209 ratings (1, 2, 3, 4, 5) of
 7 # 3952 movies from 6040 users who joined MovieLens in 2000.
 8
 9 # Author: Yiming Li
10 #
11 # Assign each user to one "fan group".
12
13 ### We have selected the 10 highest rated movies from
14 ### each of the following genres --
15 ###   Action, children's, drama, horror, romance, thriller
16
17 # Action
18 action <- movies$mid[grep("action", movies$genre)]
19 #> length(action)
20 #[1] 16
21
22 # Children's
23 children <- movies$mid[grep("children", movies$genre)]
24 #> length(children)
25 #[1] 10
26
27 # Drama
28 drama <- movies$mid[grep("drama", movies$genre)]
29 #> length(drama)
30 #[1] 18
31
32 # Horror
33 horror <- movies$mid[grep("horror", movies$genre)]
34 #> length(horror)
35 #[1] 8
36
37 # Romance
38 romance <- movies$mid[grep("romance", movies$genre)]
39 #> length(romance)
40 #[1] 12
41
42 # Thriller
43 thriller <- movies$mid[grep("thriller", movies$genre)]
44 #> length(thriller)
45 #[1] 14
46
47 actionAvg <- sapply(1:nrow(users), function(i) {
48   return(mean(as.integer(ratings[ratings$mid %in% action & ratings$uid == users$uid[i
       ], "rating"])))
49 })
50 # actionAvg[i] is the average stars the i^th user in users gave to all action movies (
       s)he rated
51
52 childrenAvg <- sapply(1:nrow(users), function(i) {
53   return(mean(as.integer(ratings[ratings$mid %in% children & ratings$uid == users$uid[
       i], "rating"])))
54 })
55
56 dramaAvg <- sapply(1:nrow(users), function(i) {
57   return(mean(as.integer(ratings[ratings$mid %in% drama & ratings$uid == users$uid[i],
        "rating"])))
```

```
})

horrorAvg <- sapply(1:nrow(users), function(i) {
  return(mean(as.integer(ratings[ratings$mid %in% horror & ratings$uid == users$uid[i
      ], "rating"])))
})

romanceAvg <- sapply(1:nrow(users), function(i) {
  return(mean(as.integer(ratings[ratings$mid %in% romance & ratings$uid == users$uid[i
      ], "rating"])))
})

thrillerAvg <- sapply(1:nrow(users), function(i) {
  return(mean(as.integer(ratings[ratings$mid %in% thriller & ratings$uid == users$uid[
      i], "rating"])))
})

actionNo <- sapply(1:nrow(users), function(i) {
  return(nrow(ratings[ratings$mid %in% action & ratings$uid == users$uid[i],]))
})

childrenNo <- sapply(1:nrow(users), function(i) {
  return(nrow(ratings[ratings$mid %in% children & ratings$uid == users$uid[i],]))
})

dramaNo <- sapply(1:nrow(users), function(i) {
  return(nrow(ratings[ratings$mid %in% drama & ratings$uid == users$uid[i],]))
})

horrorNo <- sapply(1:nrow(users), function(i) {
  return(nrow(ratings[ratings$mid %in% horror & ratings$uid == users$uid[i],]))
})

romanceNo <- sapply(1:nrow(users), function(i) {
  return(nrow(ratings[ratings$mid %in% romance & ratings$uid == users$uid[i],]))
})

thrillerNo <- sapply(1:nrow(users), function(i) {
  return(nrow(ratings[ratings$mid %in% thriller & ratings$uid == users$uid[i],]))
})

users$action <- actionAvg
users$children <- childrenAvg
users$drama <- dramaAvg
users$horror <- horrorAvg
users$romance <- romanceAvg
users$thriller <- thrillerAvg

users$actionNo <- actionNo
users$childrenNo <- childrenNo
users$dramaNo <- dramaNo
users$horrorNo <- horrorNo
users$romanceNo <- romanceNo
users$thrillerNo <- thrillerNo
```

./codes/markUsers.R

```
#!/usr/bin/env Rscript
# Author: Yiming Li
# Functions for making MDS plots for different "fan groups".
#
# Requires loading the packages specified in movieVisualization.R!

### Six pairs of colors for the genres
gCols <- brewer.pal(12, "Paired")

  # Return value <- User is of the group:
  # 1 <- action
```

```
  # 2 <- children
  # 3 <- drama
  # 4 <- horror
  # 5 <- romance
  # 6 <- thriller

### Each function returns the data frame for a certain genre
mds.rom <- function(users, mds) {
  data.frame(mds[which(users$ggroup == 5),])
}
mds.act <- function(users, mds) {
  data.frame(mds[which(users$ggroup == 1),])
}
mds.hor <- function(users, mds) {
  data.frame(mds[which(users$ggroup == 4),])
}
mds.thr <- function(users, mds) {
  data.frame(mds[which(users$ggroup == 6),])
}
mds.chi <- function(users, mds) {
  data.frame(mds[which(users$ggroup == 2),])
}
mds.dra <- function(users, mds) {
  data.frame(mds[which(users$ggroup == 3),])
}

### Density heatmap; uses genre as argument

plotGenre <- function(users, genre) {
  #actioncol <- ifelse((users$action > mean(users$action, na.rm = TRUE) & !is.na(users
      $action)), "blue", "black")
  if (genre == "action") {
    ggplot(mds.act(users,mds3),aes(x=X1,y=X2)) + stat_density2d(aes(fill=..level..),
        geom="polygon") + scale_fill_gradient(low=gCols[1], high=gCols[2]) + ggtitle("
        Action")
  } else if (genre == "children") {
    ggplot(mds.chi(users,mds3),aes(x=X1,y=X2)) + stat_density2d(aes(fill=..level..),
        geom="polygon") + scale_fill_gradient(low=gCols[7], high=gCols[8]) + ggtitle("
        Children")
  } else if (genre == "drama") {
    ggplot(mds.dra(users,mds3),aes(x=X1,y=X2)) + stat_density2d(aes(fill=..level..),
        geom="polygon") + scale_fill_gradient(low=gCols[11], high=gCols[12]) + ggtitle
        ("Drama")
  } else if (genre == "horror") {
    ggplot(mds.hor(users,mds3),aes(x=X1,y=X2)) + stat_density2d(aes(fill=..level..),
        geom="polygon") + scale_fill_gradient(low=gCols[3], high=gCols[4]) + ggtitle("
        Horror")
  } else if (genre == "romance") {
    ggplot(mds.rom(users,mds3),aes(x=X1,y=X2)) + stat_density2d(aes(fill=..level..),
        geom="polygon") + scale_fill_gradient(low=gCols[5], high=gCols[6]) + ggtitle("
        Romance")
  } else if (genre == "thriller") {
    ggplot(mds.thr(users,mds3),aes(x=X1,y=X2)) + stat_density2d(aes(fill=..level..),
        geom="polygon") + scale_fill_gradient(low=gCols[9], high=gCols[10]) + ggtitle(
        "Thriller")
  }
}

### Six functions for plotting the MDS with density contour lines

pRom <- function(users, mds, xl, yl) {
  ggplot(mds.rom(users, mds),aes(x=X1,y=X2)) + geom_point(alpha = 0.5, col = gCols[5])
       + geom_density2d(col = gCols[6]) + xlim(-xl, xl) + ylim(-yl, yl) + ggtitle("
      Romance")
}

pAct <- function(users, mds, xl, yl) {
```

```
  ggplot(mds.act(users, mds),aes(x=X1,y=X2)) + geom_point(alpha = 0.5, col = gCols[1])
      + geom_density2d(col = gCols[2]) + xlim(-xl, xl) + ylim(-yl, yl) + ggtitle("
      Action")
}

pHor <- function(users, mds, xl, yl) {
  ggplot(mds.hor(users, mds),aes(x=X1,y=X2)) + geom_point(alpha = 0.5, col = gCols[3])
      + geom_density2d(col = gCols[4]) + xlim(-xl, xl) + ylim(-yl, yl) + ggtitle("
      Horror")
}

pChi <- function(users, mds, xl, yl) {
  ggplot(mds.chi(users, mds),aes(x=X1,y=X2)) + geom_point(alpha = 0.5, col = gCols[7])
      + geom_density2d(col = gCols[8]) + xlim(-xl, xl) + ylim(-yl, yl) + ggtitle("
      Children")
}

pDra <- function(users, mds, xl, yl) {
  ggplot(mds.dra(users, mds),aes(x=X1,y=X2)) + geom_point(alpha = 0.5, col = gCols
      [11]) + geom_density2d(col = gCols[12]) + xlim(-xl, xl) + ylim(-yl, yl) +
      ggtitle("Drama")
}

pThr <- function(users, mds, xl, yl) {
  ggplot(mds.thr(users, mds),aes(x=X1,y=X2)) + geom_point(alpha = 0.5, col = gCols[9])
      + geom_density2d(col = gCols[10]) + xlim(-xl, xl) + ylim(-yl, yl) + ggtitle("
      Thriller")
}

### For comparing romance and action

compareRomAct <- function(users, mds) {
  xl <- ceiling(max(abs(range(mds[,1]))))
  yl <- ceiling(max(abs(range(mds[,2]))))

  mds.both <- ggplot(mds.rom(users, mds),aes(x=X1,y=X2)) + geom_point(alpha = 0.5, col
      = gCols[5]) + geom_density2d(col = gCols[6], alpha = 0.8, lwd = 0.8) + geom_
      point(data = mds.act(users, mds), alpha = 0.5, col = gCols[1]) + geom_density2d(
      data = mds.act(users, mds), col = gCols[2], alpha = 0.8, lwd = 0.8) + ggtitle("
      MDS Plot With Density Contour Lines\nRed -- Romance; Blue -- Action")

  grid.arrange(mds.both)
  grid.arrange(pRom(users,mds,xl,yl), pAct(users,mds,xl,yl), ncol = 2)
}

romact <- function() {
  pdf("new-romAct.pdf", width = 10, height = 7)
  compareRomAct(users, mds3)
  dev.off()
}

### For comparing all six genres

allsix <- function(mds) {
  xl <- max(abs(range(mds[,1])))
  yl <- max(abs(range(mds[,2])))
  mds.allsix <- ggplot(mds.rom(users, mds),aes(x=X1,y=X2)) + geom_point(alpha = 0.5,
      col = gCols[5]) + geom_density2d(col = gCols[6], alpha = 0.8, lwd = 0.8) + geom_
      point(data = mds.act(users, mds), alpha = 0.5, col = gCols[1]) + geom_density2d(
      data = mds.act(users, mds), col = gCols[2], alpha = 0.8, lwd = 0.8) + geom_point
      (data = mds.hor(users, mds), alpha = 0.5, col = gCols[3]) + geom_density2d(data
      = mds.hor(users, mds), col = gCols[4], alpha = 0.8, lwd = 0.8) + geom_point(data
       = mds.thr(users, mds), alpha = 0.5, col = gCols[9]) + geom_density2d(data = mds
      .thr(users, mds), col = gCols[10], alpha = 0.8, lwd = 0.8) + geom_point(data =
      mds.dra(users, mds), alpha = 0.5, col = gCols[11]) + geom_density2d(data = mds.
      dra(users, mds), col = gCols[12], alpha = 0.8, lwd = 0.8) + geom_point(data =
      mds.chi(users, mds), alpha = 0.5, col = gCols[7]) + geom_density2d(data = mds.
      chi(users, mds), col = gCols[8], alpha = 0.8, lwd = 0.8) + ggtitle("MDS Plot
```

```
      With Density Contour Lines")

  grid.arrange(mds.allsix)
  grid.arrange(pAct(users,mds,xl,yl), pChi(users,mds,xl,yl), pDra(users,mds,xl,yl),
      pHor(users,mds,xl,yl), pRom(users,mds,xl,yl), pThr(users,mds,xl,yl), ncol = 3)
}

plotAllSix <- function() {
  pdf("new-allsix.pdf", width = 10, height = 7)
  allsix(mds3)
  dev.off()
}

plotAllSixHeat <- function() {
  pdf("new-allsixheat.pdf", width = 10, height = 7)
  grid.arrange(plotGenre(users, "action"))
  grid.arrange(plotGenre(users, "children"))
  grid.arrange(plotGenre(users, "drama"))
  grid.arrange(plotGenre(users, "horror"))
  grid.arrange(plotGenre(users, "romance"))
  grid.arrange(plotGenre(users, "thriller"))
  grid.arrange(plotGenre(users, "action"),plotGenre(users, "children"),plotGenre(users
      , "drama"),plotGenre(users, "horror"),plotGenre(users, "romance"),plotGenre(
      users, "thriller"),ncol = 3)
  dev.off()
}
```

./codes/compareGenres.R

```
#!/usr/bin/env Rscript
#
# The dataset is available on the Web at
# http://files.grouplens.org/datasets/movielens/ml-1m.zip
#
# The whole dataset includes 1000209 ratings (1, 2, 3, 4, 5) of
# 3952 movies from 6040 users who joined MovieLens in 2000.

# Author: Yiming Li
#
# The driver script for all the visualization (except for the representative rank).

load("movie2.RData")
rm(list = ls()[!ls() %in% c("users", "ratings", "movies", "mds3")])
#> dim(ratings)
#[1] 72979      4
#> dim(users)
#[1] 5625    5
#> dim(movies)
#[1] 55  6

source("markUsers.R")
# Now "users" has the additional columns
#   action, children, drama, horror, romance and thriller
# which is the average score each user gave to that genre,
# as well as the additional columns
#   actionNo, childrenNo, dramaNo, horrorNo, romanceNo and thrillerNo
# which is the number of movies of that genre each user rated.
save(list = ls(all = TRUE), file = "movie3.RData")

# load("movie3.RData")
rm(list = ls()[!ls() %in% c("users", "ratings", "movies", "mds", "mds2", "mds3")])

#> dim(movies)
#[1] 55  6
#> dim(ratings)
#[1] 72979     4
#> dim(users)
#[1] 5625   17
```

```
40
41 # install.packages("ggplot2", dependencies = TRUE)
42 # install.packages("hexbin")
43 # install.packages("gridExtra")
44 library("ggplot2")
45 library("gridExtra")
46 # library("grDevices")
47 library("RColorBrewer")
48
49 source("compareGender.R")
50 source("compareAge.R")
51 source("compareJob.R")
52
53 ################## Start of assigning a group to each user
54
55 #users[i,6] action
56 #users[i,7] children
57 #users[i,8] drama
58 #users[i,9] horror
59 #users[i,10] romance
60 #users[i,11] thriller
61
62 genreGroup <- sapply(1:nrow(users), function(i) {
63   # Return value <- User is of the group:
64   # 1 <- action
65   # 2 <- children
66   # 3 <- drama
67   # 4 <- horror
68   # 5 <- romance
69   # 6 <- thriller
70   indices <- which(users[i,6:11] == max(users[i,6:11], na.rm = TRUE))
71   if (length(indices) == 1) {
72     return(indices)
73   } else {
74     indices <- indices[which(users[i, indices + 11] == max(users[i, indices + 11], na.
         rm = TRUE))]
75     return(ifelse((length(indices) == 1), indices, sample(indices, 1)))
76   }
77 })
78
79 #> sum(genreGroup == 1)
80 #[1] 1425
81 #> sum(genreGroup == 2)
82 #[1] 601
83 #> sum(genreGroup == 3)
84 #[1] 1194
85 #> sum(genreGroup == 4)
86 #[1] 424
87 #> sum(genreGroup == 5)
88 #[1] 805
89 #> sum(genreGroup == 6)
90 #[1] 1176
91
92 #allNA <- sapply(1:nrow(users), function(i) {
93 # sum(is.na(users[i,6:11])) == 6
94 #})
95 ##> sum(allNA)
96 ##[1] 0
97
98 users$ggroup <- genreGroup
99
100 save(list = ls(all = TRUE), file = "movie4.RData")
101
102 # load("movie4.RData")
103 rm(list = ls()[!ls() %in% c("users", "ratings", "movies", "mds", "mds2", "mds3")])
104
105 users <- users[,(names(users) %in% c("uid", "gender", "age", "job", "zip", "ggroup"))]
106
```

```
source("compareGenres.R")

romact()
plotAllSix()
plotAllSixHeat()

pdf("heat.pdf", width = 10, height = 7)

grid.arrange(ggplot(data.frame(mds3),aes(x=X1,y=X2)) + stat_density2d(aes(fill=..level
    ..), geom="polygon", h = rep(0.07, 2), n = c(800,800)) + scale_fill_gradient(low="
    yellow", high="red") + ggtitle("Normalized Distances Between Movie Raters") + geom
    _rect(xmin = -0.1, xmax = -0.08, ymin = 0.07, ymax = 0.09, fill = NA, color = "
    magenta", size = 0.4) + geom_rect(xmin = 0.11, xmax = 0.125, ymin = -0.11, ymax =
    -0.09, fill = NA, color = "blue", size = 0.4) + geom_rect(xmin = 0.135, xmax =
    0.155, ymin = -0.005, ymax = 0.015, fill = NA, color = "green", size = 0.4) + geom
    _rect(xmin = -0.04, xmax = -0.035, ymin = -0.24, ymax = -0.22, fill = NA, color =
    "purple", size = 0.4) + geom_rect(xmin = -0.075, xmax = -0.065, ymin = -0.09, ymax
     = -0.08, fill = NA, color = "cyan", size = 0.4))

dev.off()
```

./codes/movieVisualization.R

```
#!/usr/bin/env Rscript
# Author: Yiming Li
#
# 1. Identify the local maximum density regions.
# 2. Find the representative complete ranking.
# 3. Visualize the results.

library("KernSmooth")
library("ggplot2")
library("gridExtra")
library("RColorBrewer")

### Preliminary data / functions loading
load("movie2.RData")
load("movie4.RData")
rm(list = ls()[!ls() %in% c("users", "ratings", "movies", "mds", "mds2", "mds3", "
    moviemat")])

#> dim(mds3)
#[1] 5625    2
#> dim(moviemat)
#[1] 5625   55
#> dim(movies)
#[1] 55  6
#> dim(ratings)
#[1] 72979     4
#> dim(users)
#[1] 5625   18

inrange <- function(pt, rangex, rangey) {
  return (pt[1] >= rangex[1] & pt[1] <= rangex[2] & pt[2] >= rangey[1] & pt[2] <=
      rangey[2])
}

func_a2 <- function(r, i, j) {
  # (12) in "Rank correlation methods for missing data"
  # For a given pair of movies (i,j), 1 <= i < j <= t
  # and an incomplete ranking r, calculate the score
  k <- sum(!is.na(r))
  if (is.na(r[i]) & is.na(r[j])) {
    return(0)
  } else if (is.na(r[i]) & !is.na(r[j])) {
    return(2 * r[j] / (k + 1) - 1)
  } else if (is.na(r[j])) {
    return(1 - 2 * r[i] / (k + 1))
```

```
  } else {
    return(sign(r[j] - r[i]))
  }
}

k_dist <- function(r1, r2, c_k, t) {
  ### For two incomplete rankings, calculate the
  ### Kendall distance between them
  kdis <- c_k
  for (i in 1:(t-1)) {
    for (j in (i+1):t) {
      kdis <- kdis - func_a2(r1,i,j)*func_a2(r2,i,j)
    }
  }
  return(kdis)
}

est <- bkde2D(mds3, bandwidth = rep(min(dpik(mds3[,1]), dpik(mds3[,2])), 2), gridsize
    = c(800,800))

pdf("tmp.pdf")

contour(est$x1, est$x2, est$fhat,title = "MDS Plot With Density Contour Lines --
    Normalized Distance")
grid(100,100, lwd = 0.5)

# left, bottom, right, top
rect(-0.1, 0.07, -0.08, 0.09, border = "magenta")
rect(0.11, -0.11, 0.125, -0.09, border = "blue")
rect(0.135, -0.005, 0.155, 0.015, border = "green")
rect(-0.04, -0.24, -0.035, -0.22, border = "purple")
rect(-0.075, -0.09, -0.065, -0.08, border = "cyan")

# abline(h = c(0.1,-0.1), v = c(0.1,-0.1), col = "pink")

dev.off()

### 30 raters in the highest density region (pink)
group0 <- apply(mds3, 1, function(x) inrange(x, c(-0.1,-0.08), c(0.07,0.09)))
### group0 is a boolean vector of length 5625

### 13 raters in local maximum 1 (blue)
group1 <- apply(mds3, 1, function(x) inrange(x, c(0.11,0.125), c(-0.11,-0.09)))

### 18 raters in local maximum 2 (green)
group2 <- apply(mds3, 1, function(x) inrange(x, c(0.135,0.155), c(-0.005,0.015)))

### 5 raters in local maximum 3 (purple)
group3 <- apply(mds3, 1, function(x) inrange(x, c(-0.04,-0.035), c(-0.24,-0.22)))

### 7 raters in local maximum 4 (cyan)
group4 <- apply(mds3, 1, function(x) inrange(x, c(-0.075,-0.065), c(-0.09,-0.08)))

### Check the relative locations of the density local maximums
# ggplot(data.frame(mds3),aes(x=X1,y=X2)) + geom_density2d(alpha = 0.8, lwd = 0.8) +
    geom_point(data = data.frame(mds3[group0,]), alpha = 0.9, col = "pink", pch = "+")
     + geom_point(data = data.frame(mds3[group1,]), alpha = 0.9, col = "blue", pch =
    "+") + geom_point(data = data.frame(mds3[group2,]), alpha = 0.9, col = "green",
    pch = "+") + geom_point(data = data.frame(mds3[group3,]), alpha = 0.9, col = "
    purple", pch = "+") + geom_point(data = data.frame(mds3[group4,]), alpha = 0.9,
    col = "cyan", pch = "+")


### Now we have 5 groups (group0 -- group4).
### We would like to find a complete ranking r for each group s.t.
### the sum of the distances from r to each ranking in the group
### is minimized.
### To do this, we use a bubble-sort-like approach.
```

```
104
105 t <- 55 # Number of movies
106 c_k <- t * (t - 1) / 2 # Mean of Kendall distance
107
108 totalDist <- function(r, group, moviemat, c_k, t) {
109   ### For a ranking r and a given group, calculate the
110   ### sum of the distances from r to each ranking in the group
111   n <- sum(group) # Number of raters in the group
112   groupr <- moviemat[group,]
113   # A n * 55 matrix containing the rankings given by the raters
114   # in the group; the i^th row is the incomplete ranking of the
115   # movies by the i^th user
116   total <- k_dist(r, groupr[1,], c_k, t)
117   for (i in 2:n) {
118     total <- total + k_dist(r, groupr[i,], c_k, t)
119   }
120   return(total)
121 }
122
123 swapOrder <- function(r, i) {
124   ### Return the ranking resulting by swapping
125   ### the i^th and (i+1)^th ranked objects
126   ### in the original ranking.
127   ### e.g. r = c(1,3,5,2,4), i.e. A > D > B > E > C
128   ###      When i = 3, the returned result would be
129   ###  r' = c(1,4,5,2,3), i.e. A > D > E > B > C
130
131   ### Constraint: i <= length(r) - 1
132   if (i >= length(r)) {
133     warning("Invalid input")
134   } else {
135     r2 <- r
136     r2[which(r == i)] <- (i+1)
137     r2[which(r == (i+1))] <- i
138   }
139   return(r2)
140 }
141
142 findRepRank <- function(group, moviemat, c_k, t) {
143   ### Initialize the ranking to be movie1 > ... > movie55
144   r <- 1:t
145
146   count <- t
147   repeat {
148     changed <- FALSE
149     count <- count - 1
150     for(i in 1:count) {
151       rTemp <- swapOrder(r,i)
152       if (totalDist(r, group, moviemat, c_k, t) > totalDist(rTemp, group, moviemat, c_
              k, t)) {
153         r <- rTemp
154         changed <- TRUE
155       }
156     }
157     print(count)
158     print(totalDist(r, group, moviemat, c_k, t))
159     print(r)
160     if (!changed) break;
161     if (count <= 1) break;
162   }
163
164   return(r)
165 }
166
167 r0 <- findRepRank(group0, moviemat, c_k, t) # pink
168 # Stopped at count = 6
169 # [1] 50  6  1 44 21 42 15  4 26 45 52  5 28 20 11 31 22 30 53 16  7 47 12 48 55
170 #[26] 54 32 49  8 27 23  9 41 17 29 46 24  2 33 13 38 39 34 14 40 25 10 51 37 18
```

```
#[51]  3 35 36 19 43

#totalDist(r0, group0, moviemat, c_k, t)
#[1] 41724.67
#totalDist(1:55, group0, moviemat, c_k, t)
#[1] 44431.22

r1 <- findRepRank(group1, moviemat, c_k, t) # BLUE
# Stopped at count = 5
# [1] 48 22 14 53 30 32 12 21 15 33  1 11  4  8 36 20 27  9  2 23 37 28 24 34 55
#[26] 26 50 29 16 10  5 51 49 25 31 35 38 18 13 39 42 47 40 17 43 54  3  7 45 19
#[51] 41 44 52 46  6

#totalDist(r1, group1, moviemat, c_k, t)
#[1] 15766.23
#totalDist(1:55, group1, moviemat, c_k, t)
#[1] 19120.37

r2 <- findRepRank(group2, moviemat, c_k, t) # GREEN
# Stopped at count = 7
# [1] 39 35 13 41 31 42  7 24 37 22  1 12  2  3 19 40 21 11  4 15 43 38 44 36 55
#[26]  6 25 54 27  5 26 32 47 14 20 45 10 28 16  9 48 49 23 29 52 34 18 33 30 17
#[51] 50 51 46 53  8

#totalDist(r2, group2, moviemat, c_k, t)
#[1] 22988.19
#totalDist(1:55, group2, moviemat, c_k, t)
#[1] 25955.29

r3 <- findRepRank(group3, moviemat, c_k, t) # PURPLE
# Stopped at count = 1
# [1] 25 26 27 28 11 29  2 12 30 31 13 32 14  3 33 34 35  4 36 15 16 37 17 38 18
#[26] 55 19 39  5  6  7 54 40 20  8 41 42 21 43 22 44 23  9 45 46 47 24 10 48 49
#[51] 50 51 52 53  1

#totalDist(r3, group3, moviemat, c_k, t)
#[1] 6741.481
#totalDist(1:55, group3, moviemat, c_k, t)
#[1] 7549.111

r4 <- findRepRank(group4, moviemat, c_k, t) # cyan
# Stopped at count = 12
# [1] 19 20 21 22 23  7  4 24 17 25 26 27 52 28 29  8  2  1  5  9 30 31 32 33 54
#[26] 55 34  3 35 10 36 11 37 12 13 14 18 38 39 15 40 41 53 42 43 44 45  6 46 47
#[51] 48 49 16 50 51

#totalDist(r4, group4, moviemat, c_k, t)
#[1] 9447.143
#totalDist(1:55, group4, moviemat, c_k, t)
#[1] 10307.38

save(list = ls(all = TRUE), file = "lms.RData")

gCols <- brewer.pal(12, "Paipink")

genreToCol <- function(g) {
  if (g == "action") {
    return(gCols[2])
  } else if (g == "children") {
    return(gCols[8])
  } else if (g == "drama") {
    return(gCols[12])
  } else if (g == "horror") {
    return(gCols[4])
  } else if (g == "romance") {
    return(gCols[6])
  } else if (g == "thriller") {
    return(gCols[10])
```

```
  }
}

### Plot the color bars and grayscale bars
### for each representative ranking
pdf("peaks.pdf", width = 10, height = 7)

# Peak A
r0cols <- sapply(1:55, function(x) {
  genreToCol(movies$genre[which(r0 == x)])
})

r0titles <- sapply(1:55, function(x) {
  movies$title[which(r0 == x)]
})

grays <- gray.colors(78, start = 0, end = 0.5, gamma = 2.2)
r0years <- sapply(1:55, function(x) {
  grays[movies$year[which(r0 == x)]-1921]
})

plot(1:55,xlim = c(0,58),ylim = c(-0.1,5), type = "n", xaxt='n', , yaxt = "n", ann=
    FALSE)
rect(1:55,rep(0.8,55),2:56,rep(1.3,55),col = r0cols, border=NA)
rect(1:55,rep(0,55),2:56,rep(0.5,55),col = r0years, border=NA)
text(seq(1.5,55.5),rep(2.3,55),labels = r0titles, srt = 90,cex = 0.8)
abline(v = c(6,11,21),col = "pink")

# Peak B
r1cols <- sapply(1:55, function(x) {
  genreToCol(movies$genre[which(r1 == x)])
})

r1titles <- sapply(1:55, function(x) {
  movies$title[which(r1 == x)]
})

grays <- gray.colors(78, start = 0, end = 0.5, gamma = 2.2)
r1years <- sapply(1:55, function(x) {
  grays[movies$year[which(r1 == x)]-1921]
})

plot(1:55,xlim = c(0,58),ylim = c(-0.1,5), type = "n", xaxt='n', , yaxt = "n", ann=
    FALSE)
rect(1:55,rep(0.8,55),2:56,rep(1.3,55),col = r1cols, border=NA)
rect(1:55,rep(0,55),2:56,rep(0.5,55),col = r1years, border=NA)
text(seq(1.5,55.5),rep(2.3,55),labels = r1titles, srt = 90,cex = 0.8)
abline(v = c(6,11,21),col = "pink")

# Peak C
r2cols <- sapply(1:55, function(x) {
  genreToCol(movies$genre[which(r2 == x)])
})

r2titles <- sapply(1:55, function(x) {
  movies$title[which(r2 == x)]
})

grays <- gray.colors(78, start = 0, end = 0.5, gamma = 2.2)
r2years <- sapply(1:55, function(x) {
  grays[movies$year[which(r2 == x)]-1921]
})

plot(1:55,xlim = c(0,58),ylim = c(-0.1,5), type = "n", xaxt='n', , yaxt = "n", ann=
    FALSE)
rect(1:55,rep(0.8,55),2:56,rep(1.3,55),col = r2cols, border=NA)
rect(1:55,rep(0,55),2:56,rep(0.5,55),col = r2years, border=NA)
text(seq(1.5,55.5),rep(2.3,55),labels = r2titles, srt = 90,cex = 0.8)
```

```
abline(v = c(6,11,21),col = "pink")

# Peak D
r3cols <- sapply(1:55, function(x) {
  genreToCol(movies$genre[which(r3 == x)])
})

r3titles <- sapply(1:55, function(x) {
  movies$title[which(r3 == x)]
})

grays <- gray.colors(78, start = 0, end = 0.5, gamma = 2.2)
r3years <- sapply(1:55, function(x) {
  grays[movies$year[which(r3 == x)]-1921]
})

plot(1:55,xlim = c(0,58),ylim = c(-0.1,5), type = "n", xaxt='n', , yaxt = "n", ann=
    FALSE)
rect(1:55,rep(0.8,55),2:56,rep(1.3,55),col = r3cols, border=NA)
rect(1:55,rep(0,55),2:56,rep(0.5,55),col = r3years, border=NA)
text(seq(1.5,55.5),rep(2.3,55),labels = r3titles, srt = 90,cex = 0.8)
abline(v = c(6,11,21),col = "pink")

# Peak E
r4cols <- sapply(1:55, function(x) {
  genreToCol(movies$genre[which(r4 == x)])
})

r4titles <- sapply(1:55, function(x) {
  movies$title[which(r4 == x)]
})

grays <- gray.colors(78, start = 0, end = 0.5, gamma = 2.2)
r4years <- sapply(1:55, function(x) {
  grays[movies$year[which(r4 == x)]-1921]
})

plot(1:55,xlim = c(0,58),ylim = c(-0.1,5), type = "n", xaxt='n', , yaxt = "n", ann=
    FALSE)
rect(1:55,rep(0.8,55),2:56,rep(1.3,55),col = r4cols, border=NA)
rect(1:55,rep(0,55),2:56,rep(0.5,55),col = r4years, border=NA)
text(seq(1.5,55.5),rep(2.3,55),labels = r4titles, srt = 90,cex = 0.8)
abline(v = c(6,11,21),col = "pink")

dev.off()
```

./codes/localMaximums.R

```
#!/usr/bin/env Rscript
# Author: Yiming Li
#
# Determine the goodness of fit of the MDS when k = 1, 2, 3 and 4.

library("vegan")

### Preliminary data / functions loading
load("movie2.RData")

mds3_k1 <- cmdscale(D_k3, k = 1, eig = TRUE)
mds3_k2 <- cmdscale(D_k3, k = 2, eig = TRUE)
mds3_k3 <- cmdscale(D_k3, k = 3, eig = TRUE)
mds3_k4 <- cmdscale(D_k3, k = 4, eig = TRUE)

save(list = ls(all = TRUE), file = "whichdim.RData")
save(list = c("mds3_k1", "mds3_k2", "mds3_k3", "mds3_k4"), file = "whichdim-less.RData
    ")

load("whichdim-less.RData")
```

```
20
21 #  points: a matrix with up to    k    columns whose rows give the
22 #          coordinates of the points chosen to represent the
23 #          dissimilarities.
24
25 #     eig: the n eigenvalues computed during the scaling process if
26 #             e i g    is true.  *NB*: versions of R before 2.12.1 returned
27 #          only    k    but were documented to return n - 1.
28
29 #       x: the doubly centered distance matrix if   x . r e t   is true.
30
31 #      ac: the additive constant c*,    0    if   a d d = F A L S E  .
32
33 #     GOF: a numeric vector of length 2, equal to say (g.1,g.2), where
34 #          g.i = (sum{j=1..k} lambda[j]) / (sum{j=1..n} T.i(lambda[j])),
35 #          where lambda[j] are the eigenvalues (sorted in decreasing
36 #          order), T.1(v) = abs(v), and T.2(v) = max(v, 0).
37
38 mds3_k1$GOF
39 mds3_k2$GOF
40 mds3_k3$GOF
41 mds3_k4$GOF
42
43 #> mds3_k1$GOF
44 #[1] 0.02877173 0.05088746
45 #> mds3_k2$GOF
46 #[1] 0.05306849 0.09386020
47 #> mds3_k3$GOF
48 #[1] 0.07478342 0.13226656
49 #> mds3_k4$GOF
50 #[1] 0.09463971 0.16738561
51
52 plot(cumsum(mds3_k2$eig) / sum(mds3_k2$eig), type="h", lwd=5, las=1, xlab="Number of
       dimensions", ylab=expression(R^2))
53 plot(mds3_k2$eig, type="h", lwd=5, las=1, xlab="Number of dimensions", ylab="
       Eigenvalues")
```

./codes/whichDim.R

References

[1] Kidwell P, Lebanon G, and Cleveland WS. Visualizing incomplete and partially ranked data. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1356–1363, 2008.

[2] Prentice M. On the problem of m incomplete rankings. *Biometrika*, 66:167–170, 1979.

[3] Thompson GL. Generalized permutation polytopes and exploratory graphical methods for ranked data. *The Annals of Statistics*, 21(3):1401–1430, 1993.

[4] Piesk T. Permutohedron of the symmetric group S4. Available from: http://en.wikipedia.org/wiki/File:Symmetric_group_4;_permutohedron_3D;_permutations_and_inversion_vectors.svg.

[5] Kruskal JB and Wish M. *Multidimensional scaling*, volume 11. Sage, 1978.

[6] Carroll J. *Multidimensional Scaling: Theory and Applications in the Behavioral Sciences*. volume 1, chapter Individual differences and multidimensional scaling. Seminar Press, New York, 1972.

[7] Spearman C. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):pp. 72–101, 1904.

[8] Kendall MG. A new measure of rank correlation. *Biometrika*, 30:81–93, 1938.

[9] Hamming RW. Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2):147–160, 1950.

[10] Kendall M. *Rank correlation methods*. Griffin, Oxford, 1948.

[11] Kendall M and Gibbons J. *Rank correlation methods*. Edward Arnold, London, 1990.

[12] Alvo M and Cabilio P. On the balanced incomplete block design for rankings. *Annals of Statistics*, 19:1597–1613, 1991.

[13] Alvo M and Cabilio P. Correlation methods for incomplete rankings. Technical Report 200, Laboratory for Research in Statistics and Probability, Carleton University and University of Ottawa, 1992.

[14] Alvo M and Cabilio P. Rank correlation methods for missing data. *Canadian Journal of Statistics*, 23:345–358, 1995.

[15] Resnick P, Iacovou N, Suchak M, Bergstrom P, and Riedl J. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, pages 175–186, New York, NY, USA, 1994. ACM.

[16] Feinberg J. Wordle – Beautiful Word Clouds. Available from: http://www.wordle.net/.

[17] Cox TF and Cox MAA. *Multidimensional scaling*. Chapman and Hall, 2 edition, 2001.