
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 $S = \{1, \dots, t\}$ is a permutation of S which we denote as $\mu = (\mu(1), \mu(2), \dots, \mu(t))$, where $\mu(i)$ is the rank of object i . In fact, μ is a bijection $S \rightarrow S$ 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. $\mu^{-1}(1)|\dots|\mu^{-1}(t)$. 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.

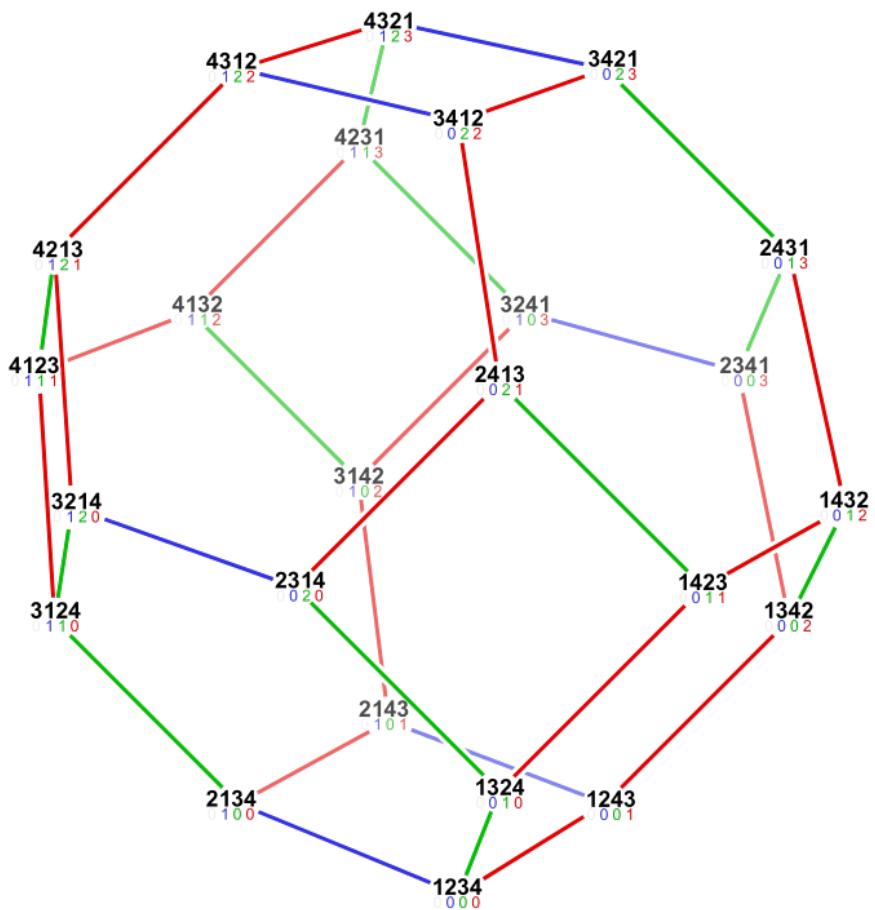


Figure 1: Colored permutation polytope for four objects [4]. The edges indicate the bitwise \leq relation whereas the colors of the edges indicate which digit changes.

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 $\mu = (\mu(1), \mu(2), \dots, \mu(t))$ and $\nu = (\nu(1), \nu(2), \dots, \nu(t))$ 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

$$d_{orig}^*(\mu^*, \nu^*) = c - A, \quad (4)$$

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

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

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

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

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

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

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

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.

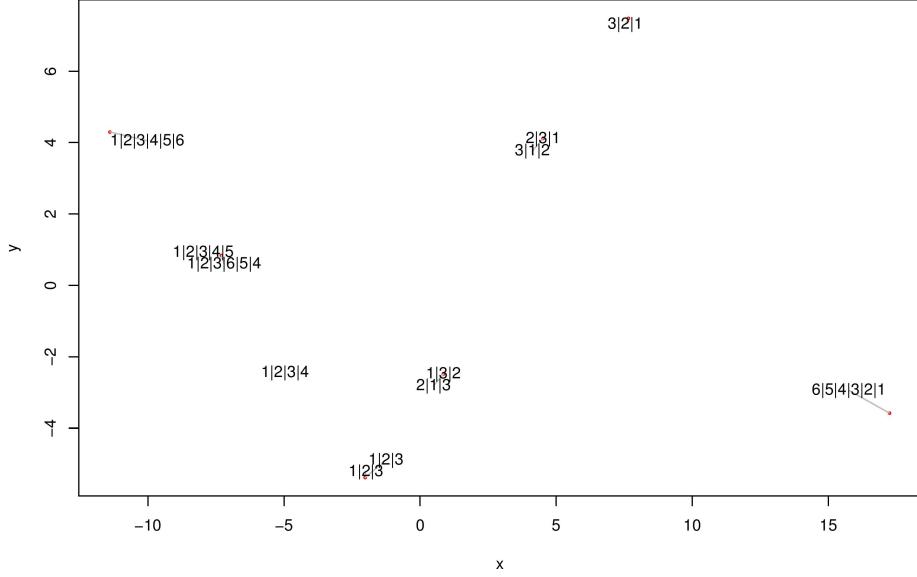


Figure 2: MDS plot using d_{orig}^* . The rankings 1|2|3 and 1|2|3, although close, do not overlap with each other. Besides, it is not sensible that the distance between 1|2|3 and 1|2|3|4|5|6 is almost the same as that between 1|2|3 and 6|5|4|3|2|1.

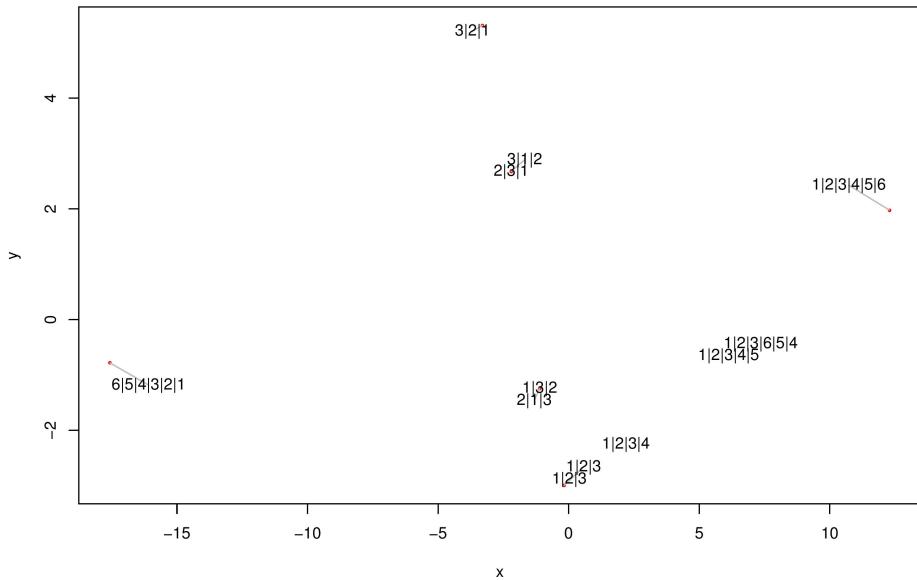


Figure 3: MDS plot using $(d^*_{orig} - m^*)$. The rankings $1|2|3$ and $1|2|3$ now overlap with each other. However, the relative locations of $1|2|3$, $1|2|3|4|5|6$ and $6|5|4|3|2|1$ are still counter-intuitive.

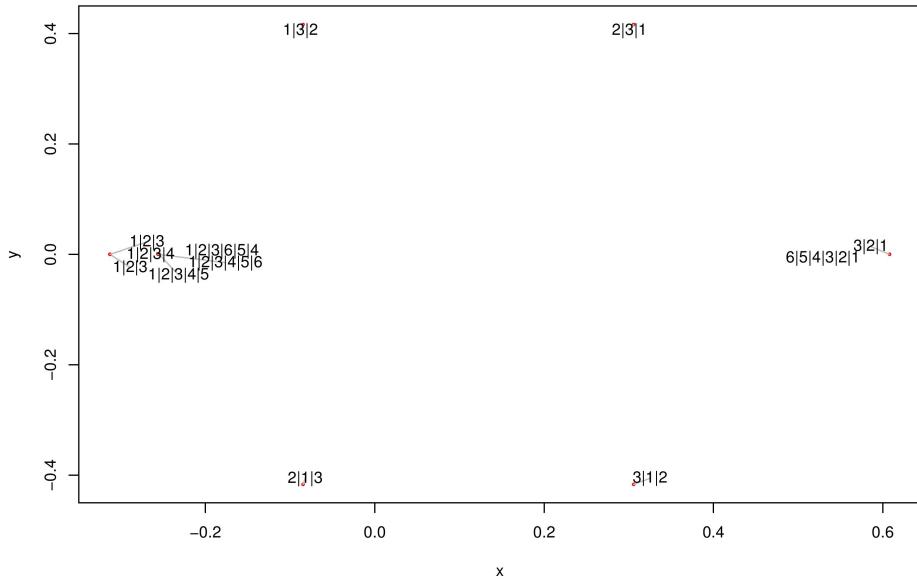


Figure 4: MDS plot using the normalized distance. The rankings with the same sub-strings ($1|2|3$, $1|2|3|4$, $1|2|3|4|5$, $1|2|3|4|5|6$ and $1|2|3|6|5|4$) now lie close to each other.

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.

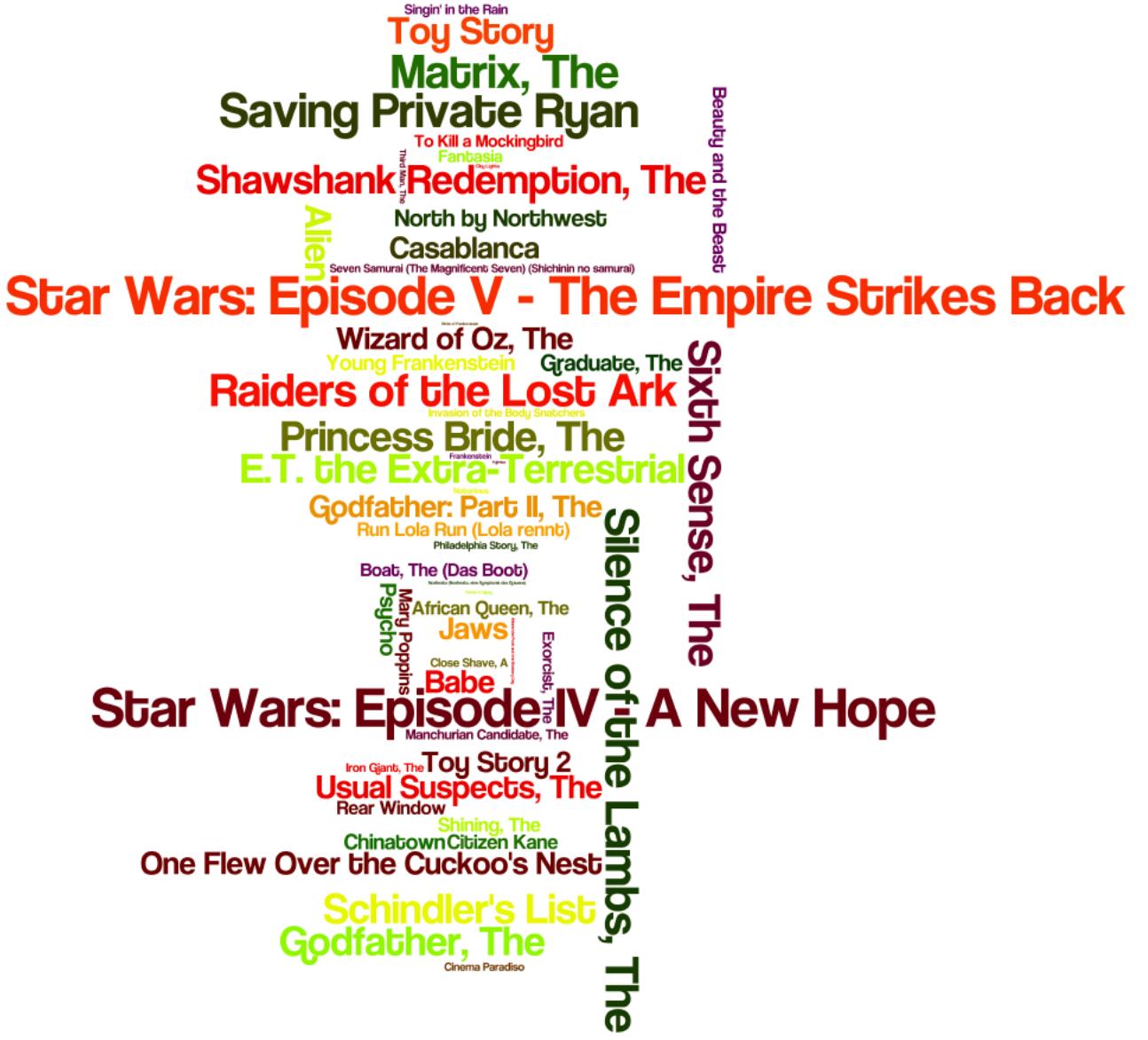


Figure 5: The 55 selected movies. This is a word cloud generated by Wordle [16]. After assigning each movie to a unique genre, there are eight action movies, ten children’s movies, eight drama movies, ten horror movies, ten romance movies and nine thrillers. In this word cloud, the sizes of the movie 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.

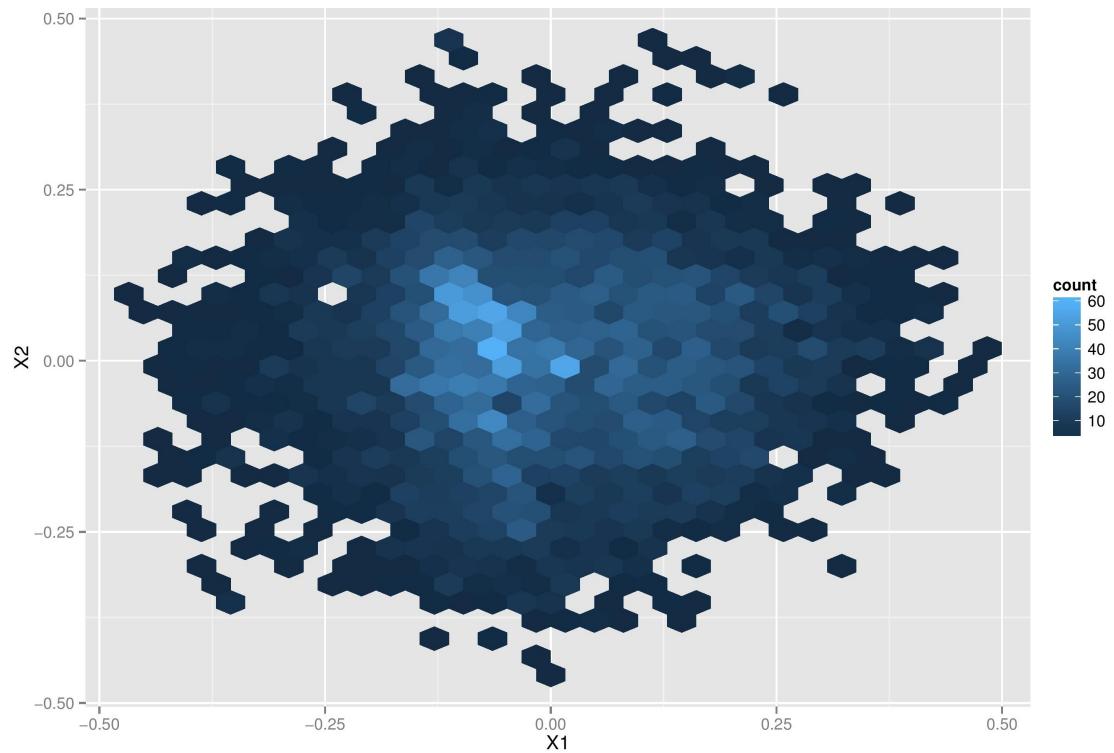


Figure 6: MDS plot with hexagonal binning.

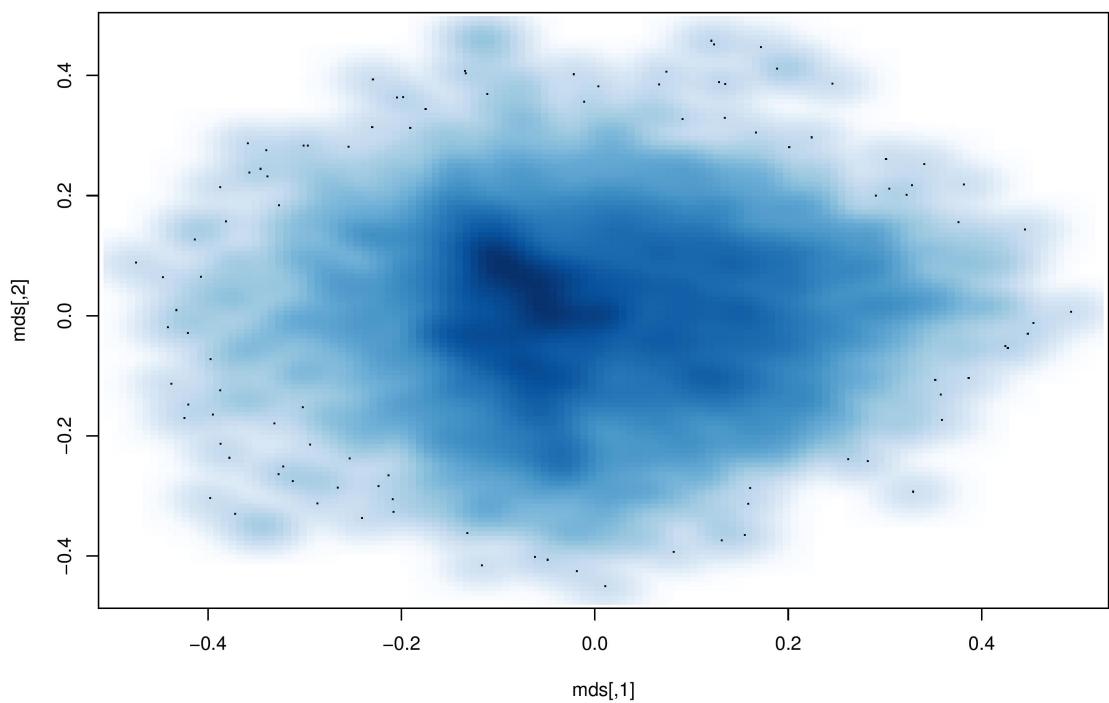


Figure 7: MDS plot with kernel smoothing.

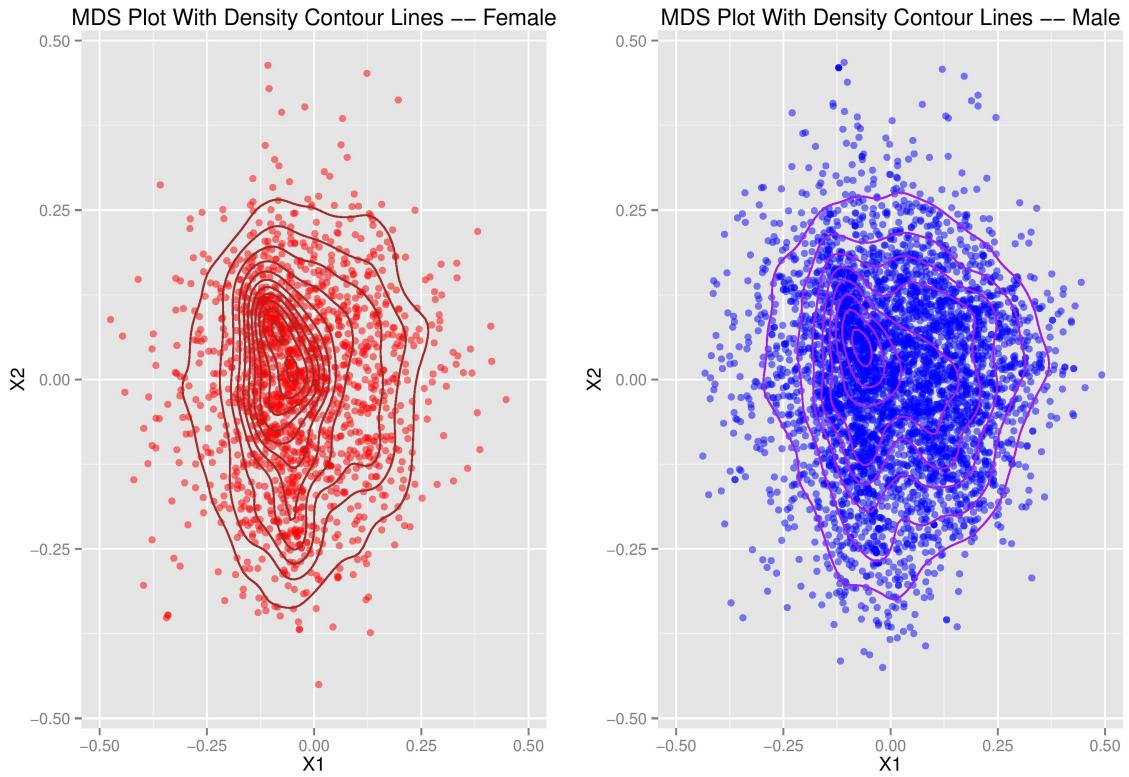


Figure 8: MDS plot by rater gender. It seems that compared to the female raters, the male raters are more widely spread and have a higher density on the right of the graph. However, this may be due to the difference between the sample sizes of males and females. 4,054 males are in the data set, whereas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

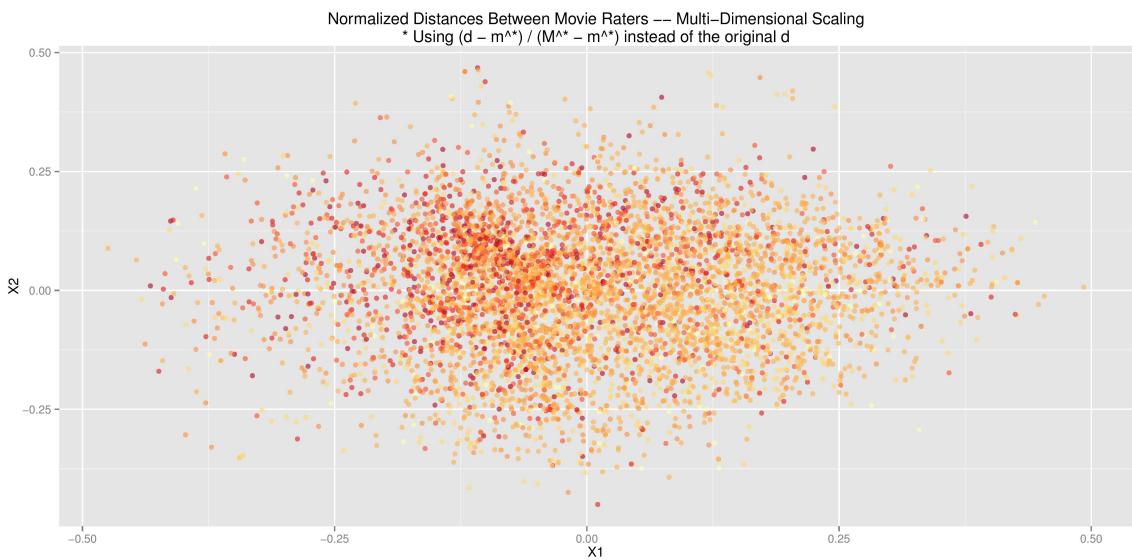


Figure 9: MDS plot by rater age. A deeper color means elder in age. On average, more of the younger raters lie on the right of the graph.

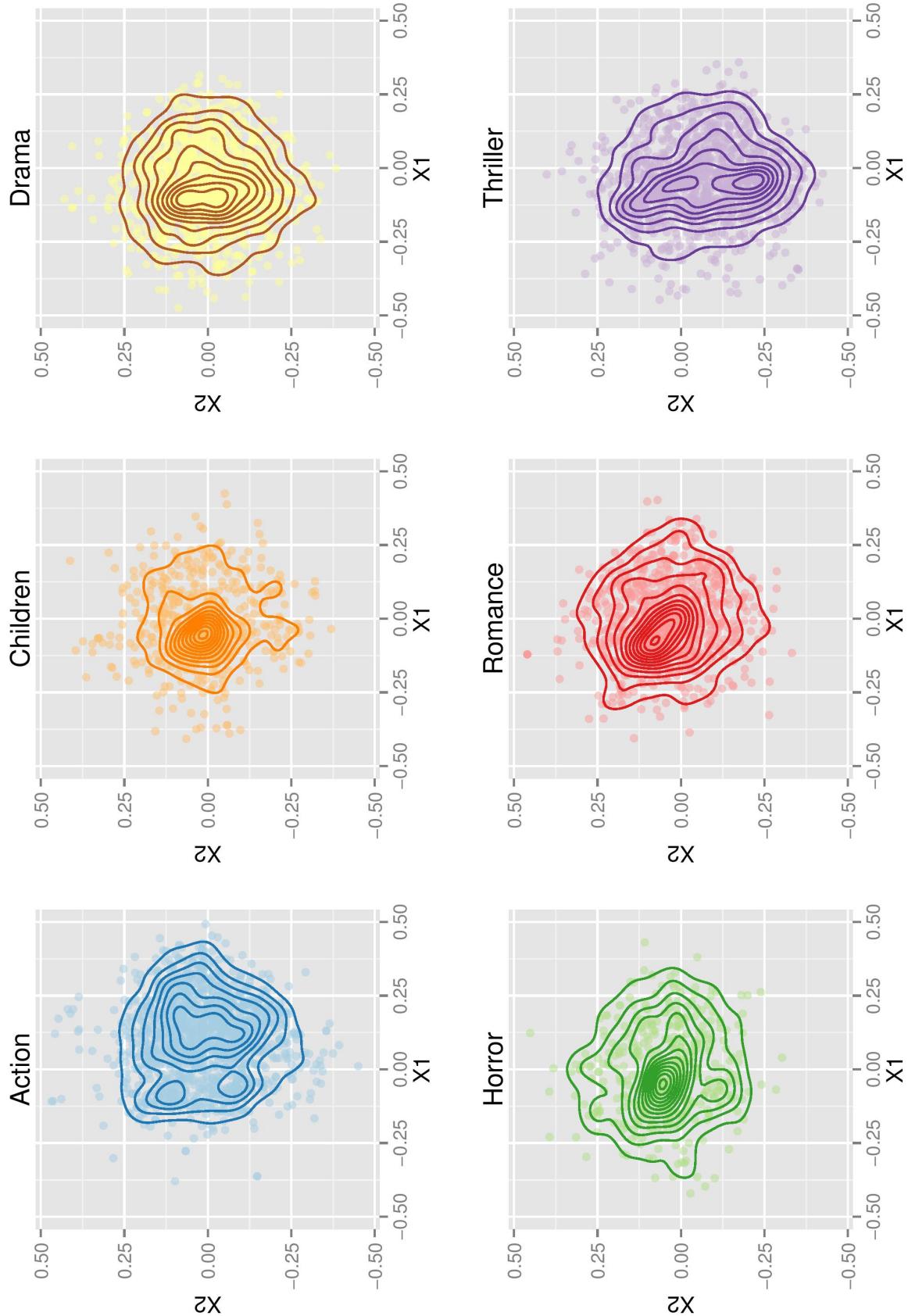


Figure 10: MDS plots with contour curves for different “fan groups”.

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.

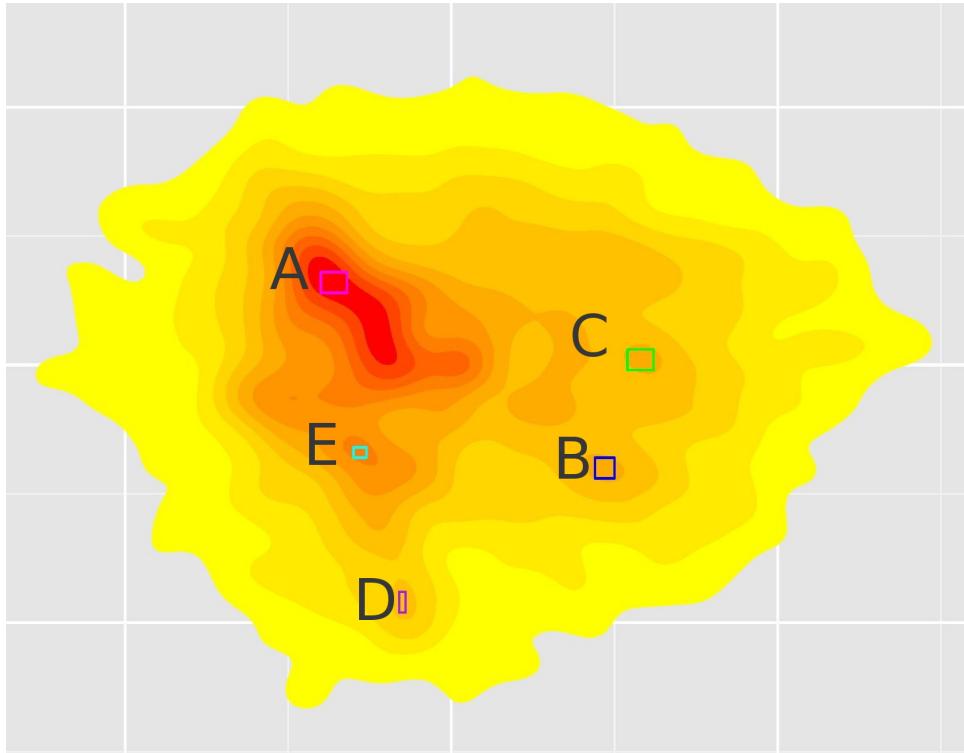


Figure 11: The five regions with local maximum density.

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 $\text{COUNT} \leq 1$ 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

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

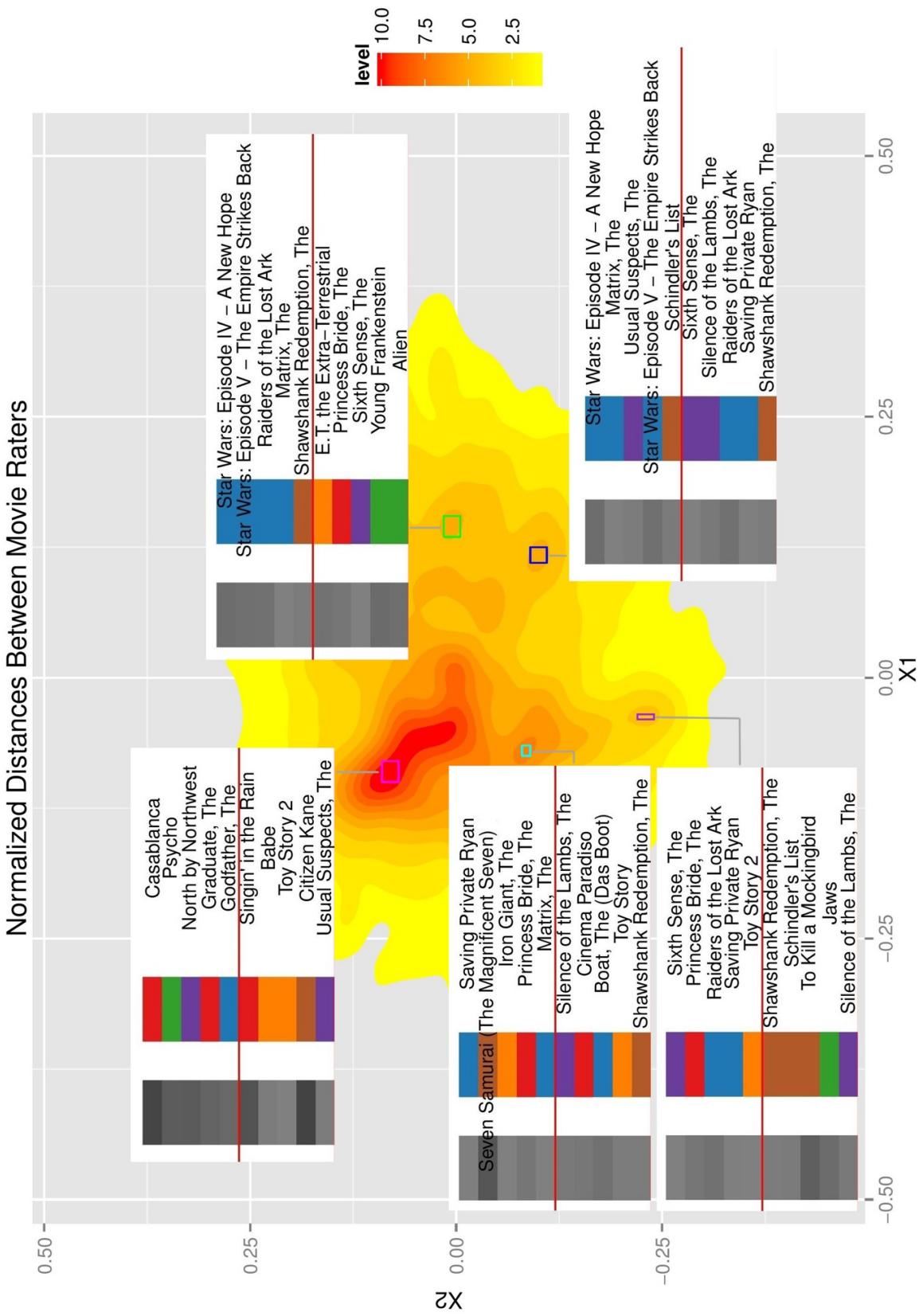


Figure 12: Representative rankings for maximum density regions. Only the top ten movies in the representative rankings are shown in this graph; the full representative rankings could be found in Appendix A. The color bar denotes the genres of the movies – red is for romance, green is for horror, purple 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.

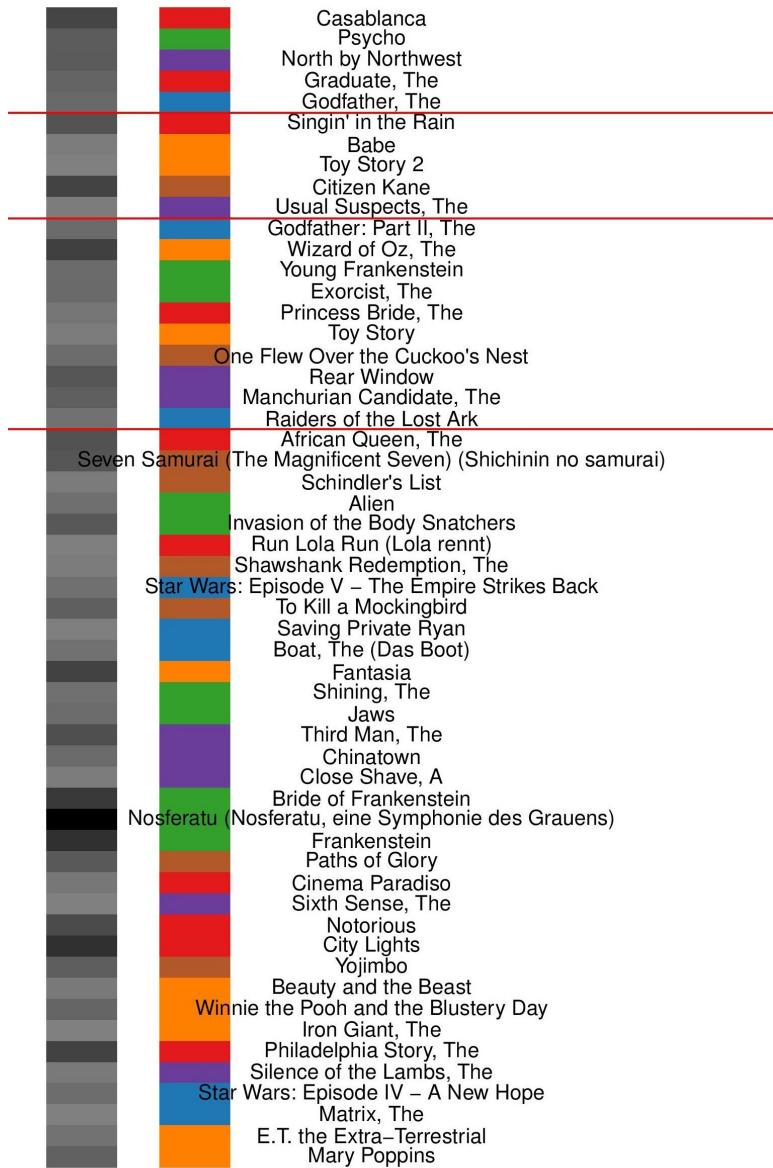


Figure 13: Peak A.

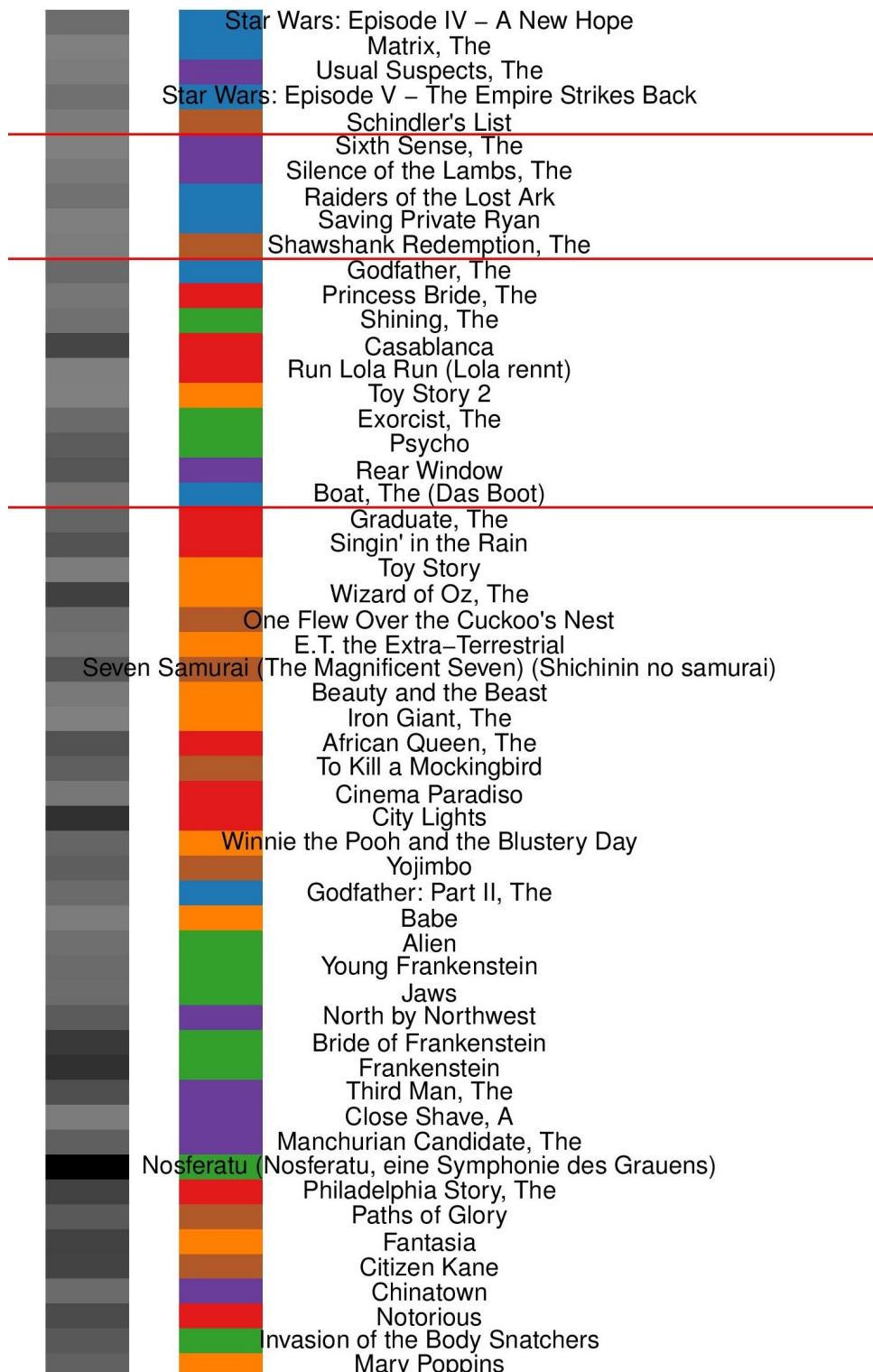


Figure 14: Peak B.

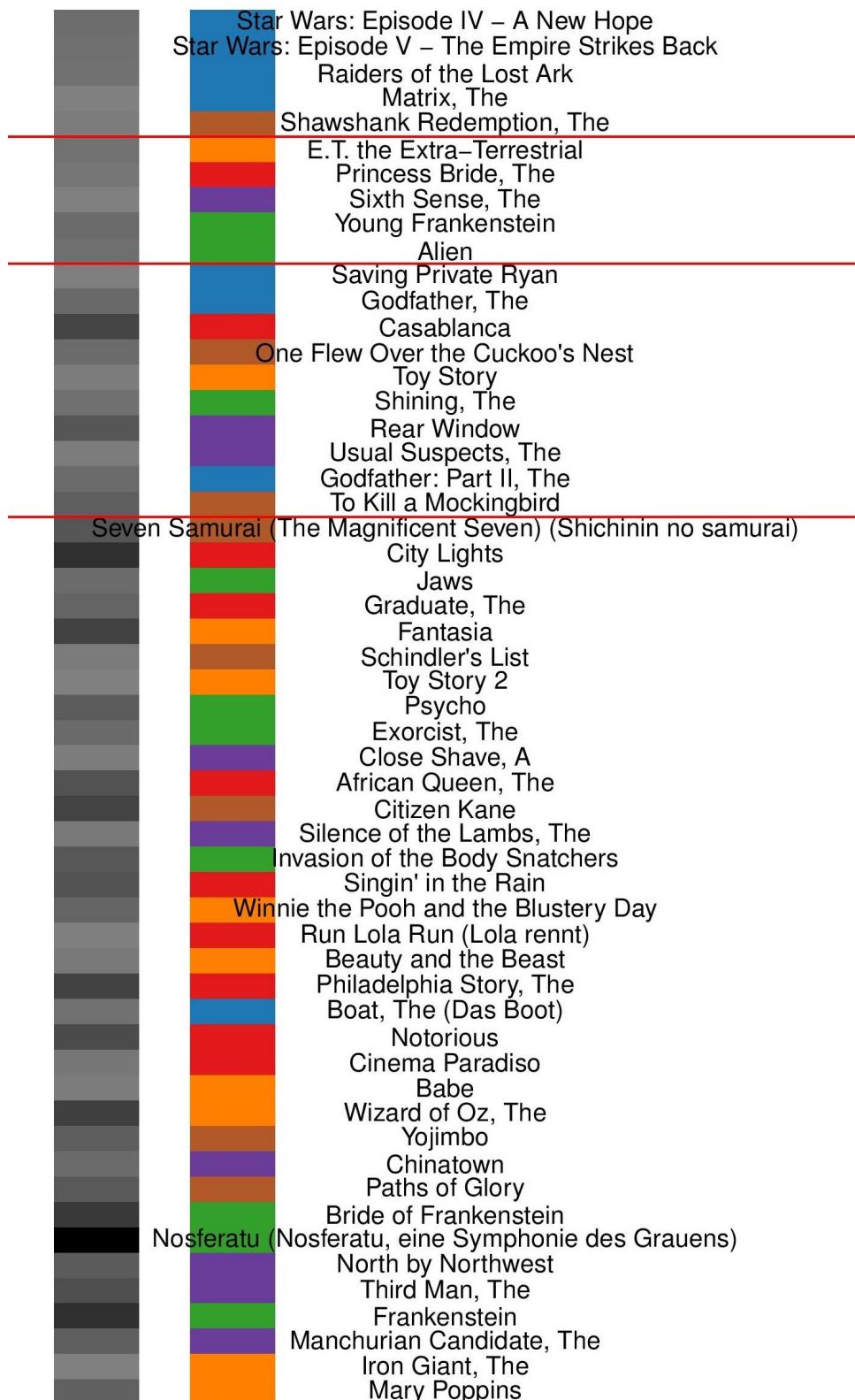


Figure 15: Peak C.

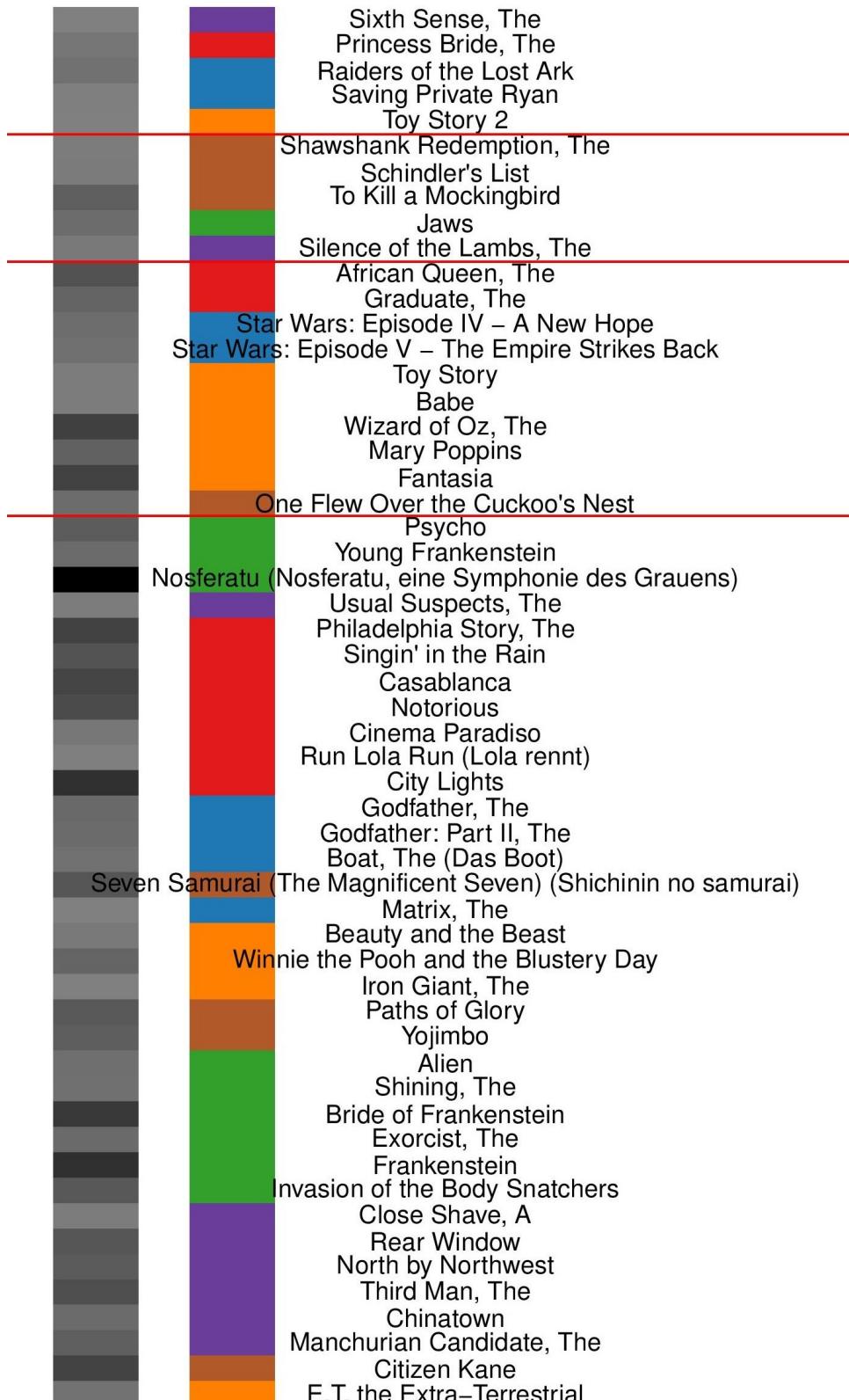


Figure 16: Peak D.

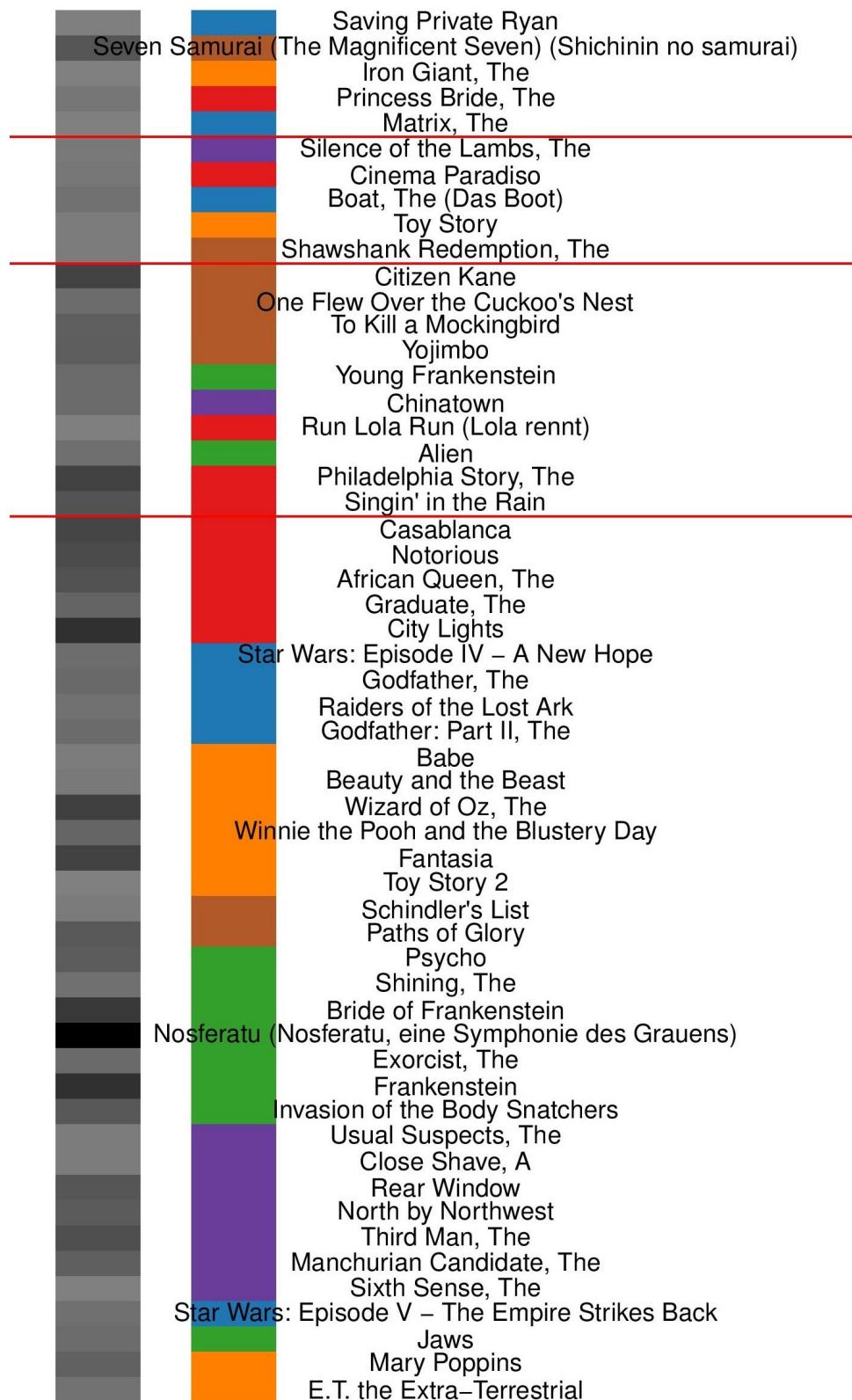


Figure 17: Peak E.

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.

```
1 #!/usr/bin/env Rscript
2 # Author: Yiming Li
3 #
4 # Toy example to justify our normalized distance.
5
6 library(wordcloud)
7
8 ### Preliminary data / functions loading
9
10 source("ksMinMaxFuncs.R")
11
12 func_a2 <- function(r, i, j) {
13   # (12) in "Rank correlation methods for missing data"
14   # For a given pair of movies (i,j), 1 <= i < j <= t
15   # and an incomplete ranking r, calculate the score
16   k <- sum(!is.na(r))
17   if (is.na(r[i]) & is.na(r[j])) {
18     return(0)
19   } else if (is.na(r[i]) & !is.na(r[j])) {
20     return(2 * r[j] / (k + 1) - 1)
21   } else if (is.na(r[j])) {
22     return(1 - 2 * r[i] / (k + 1))
23   } else {
24     return(sign(r[j] - r[i]))
25   }
26 }
27
28 k_dist <- function(r1, r2, c_k, t) {
29   ### For two incomplete rankings, calculate the
30   ### Kendall distance between them
31   kdis <- c_k
32   for (i in 1:(t-1)) {
33     for (j in (i+1):t) {
34       kdis <- kdis - func_a2(r1,i,j)*func_a2(r2,i,j)
35     }
36   }
37   return(kdis)
38 }
39
40 t <- 6 # Number of rated items
41 c_k <- t * (t - 1) / 2 # Mean of Kendall distance
42
43 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,
44   NA,3,1,2,NA,NA,NA,3,2,1,NA,NA,NA,1,2,3,4,NA,NA,NA,1,2,3,NA,NA,NA,1,2,3,4,5,NA
45   ,1,2,3,4,5,6,6,5,4,3,2,1,1,2,3,6,5,4), byrow = TRUE, ncol = t)
46 raters <- nrow(moviemat)
47
48 noRatedmovies <- c(3,3,3,3,3,4,3,5,6,6,6)
49
50 D_k <- matrix(c_k, nrow = raters, ncol = raters)
51
52 for (p in 1:raters) {
53   for (q in 1:p) {
54     D_k[p,q] = k_dist(moviemat[p,],moviemat[q,],c_k,t)
55     D_k[q,p] = D_k[p,q]
56   }
57 }
58 # D_k is the Kendall distance matrix
```

```

58 D_k2 <- D_k
59 for (p in 1:raters) {
60   for (q in 1:p) {
61     D_k2[p,q] = D_k2[p,q] - min_k(p, q, noRatedmovies, c_k, c_s, t)
62     D_k2[q,p] = D_k2[p,q]
63   }
64 }
65 # D_k2 is another version of the Kendall distance matrix
66 # Here, we use (d - m^*) instead of the original d, to make d_{\pi\pi} = 0
67
68 D_k3 <- D_k2
69 for (p in 1:raters) {
70   for (q in 1:p) {
71     D_k3[p,q] = D_k3[p,q] / (max_k(p, q, noRatedmovies, c_k, c_s, t) - min_k(p, q,
72       noRatedmovies, c_k, c_s, t))
73     D_k3[q,p] = D_k3[p,q]
74   }
75 # D_k3 is yet another version of the Kendall distance matrix
76 # Here, we use (d - m^*) / (M^* - m^*) instead of the original d
77
78 mds <- cmdscale(D_k)
79 mds2 <- cmdscale(D_k2)
80 mds3 <- cmdscale(D_k3)
81
82 pdf("toy.pdf", width = 10, height = 7)
83
84 mx <- apply(mds, 2, max)
85 mn <- apply(mds, 2, min)
86 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",
87   "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
88   [1]), ylim=c(mn[2], mx[2]))
89
90 mx <- apply(mds2, 2, max)
91 mn <- apply(mds2, 2, min)
92 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",
93   "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
94   [1]), ylim=c(mn[2], mx[2]))
95
96 dev.off()

```

./codes/toyDemo.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# Data preparation and cleaning.
12
13 movies <- read.table("movies.txt", header = FALSE, sep = "\t", col.names = c("mid", "title",
14   "year", "genres"), quote="", colClasses = c("integer", "character", "integer",
15   "character"), stringsAsFactors = FALSE)
16 #> dim(movies)
17 #[1] 3883      4
18

```

```

17 users <- read.table("users.dat", header = FALSE, sep = "\t", col.names = c("uid",
18   "gender", "age", "job", "zip"), colClasses = c("integer", "character", "character",
19   "character"))
20 #> dim(users)
21 #[1] 6040      5
22
23 ratings <- read.table("ratings.txt", header = FALSE, sep = "\t", col.names = c("uid",
24   "mid", "rating", "time"), colClasses = c("integer", "integer", "character", "integer"))
25 #> dim(ratings)
26 #[1] 1000209      4
27
28 noRatedUsers <- sapply(1:nrow(movies), function(i) sum(ratings$mid == movies$mid[i]))
29 movies$times <- noRatedUsers
30
31 ##### Check how many movies are of the genres --
32 # Action
33 action <- grep("Action", movies$genres)
34 #> length(action)
35 #[1] 493
36
37 # Children's
38 children <- grep("Children", movies$genres)
39 #> length(children)
40 #[1] 250
41
42 # Comedy
43 comedy <- grep("Comedy", movies$genres)
44 #> length(comedy)
45 #[1] 1162
46
47 # Drama
48 drama <- grep("Drama", movies$genres)
49 #> length(drama)
50 #[1] 1484
51
52 # Horror
53 horror <- grep("Horror", movies$genres)
54 #> length(horror)
55 #[1] 339
56
57 # Romance
58 romance <- grep("Romance", movies$genres)
59 #> length(romance)
60 #[1] 459
61
62 # Sci-fi
63 scifi <- grep("Sci-Fi", movies$genres)
64 #> length(scifi)
65 #[1] 274
66
67 # Thriller
68 thriller <- grep("Thriller", movies$genres)
69 #> length(thriller)
70 #[1] 484
71
72 # Check the overlapping of different pairs of genres.
73 #> length(intersect(action,scifi))
74 #[1] 107
75 #> length(intersect(action,thriller))
76 #[1] 130
77 #> length(intersect(action,romance))
78 #[1] 35
79 #> length(intersect(drama,romance))
80 #[1] 201

```

```

81 #> length(intersect(drama,action))
82 #[1] 95
83 #> length(intersect(drama,horror))
84 #[1] 11
85 #> length(intersect(action,horror))
86 #[1] 25
87 #> length(intersect(romance,horror))
88 #[1] 3
89 #> length(intersect(drama,thriller))
90 #[1] 104
91 #> length(intersect(drama,children))
92 #[1] 27
93 #> length(intersect(comedy,children))
94 #[1] 93
95
96 ### Select the 10 highest rated movies from each of the following genres --
97 ### Action, children's, drama, horror, romance, thriller
98
99 avgStar <- sapply(1:nrow(movies), function(i) mean(as.integer(ratings[ratings$mid ==
  movies$mid[i],]$rating)))
100 # avgStar[i] is the average stars of the i^th movie in movies
101 movies$star <- avgStar
102
103 romTop10 <- intersect(which(avgStar >= sort(avgStar[romance], decreasing=TRUE)[10]),
  romance)
104 # [1] 287 288 298 311 327 401 414 456 1001 1232
105 actTop10 <- intersect(which(avgStar >= sort(avgStar[action], decreasing=TRUE)[10]),
  action)
106 # [1] 94 282 413 414 415 435 445 742 748 966
107 chiTop10 <- intersect(which(avgStar >= sort(avgStar[children], decreasing=TRUE)[10]),
  children)
108 # [1] 1 18 228 303 344 347 382 488 1033 1164
109 draTop10 <- intersect(which(avgStar >= sort(avgStar[drama], decreasing=TRUE)[10]),
  drama)
110 # [1] 116 200 282 298 307 403 411 423 742 1124
111 horrTop10 <- intersect(which(avgStar >= sort(avgStar[horror], decreasing=TRUE)[10]),
  horror)
112 # [1] 429 433 466 485 517 523 544 727 986 989
113 thriTop10 <- intersect(which(avgStar >= sort(avgStar[thriller], decreasing=TRUE)[10]),
  thriller)
114 # [1] 27 226 258 292 294 427 461 475 966 1034
115
116 genre <- rep(NA, nrow(movies))
117 genre[thriTop10] <- "thriller"
118 genre[actTop10] <- "action"
119 genre[chiTop10] <- "children"
120 genre[draTop10] <- "drama"
121 genre[282] <- "action"
122 genre[romTop10] <- "romance"
123 genre[horrTop10] <- "horror"
124
125 #> intersect(romTop10, draTop10)
126 #[1] 298
127 #> which(sort(avgStar[draTop10], decreasing=TRUE) == movies$star[298])
128 #[1] 7
129 #> which(sort(avgStar[romTop10], decreasing=TRUE) == movies$star[298])
130 #[1] 1
131 ### Mark 298 as romance
132
133 #> intersect(actTop10, draTop10)
134 #[1] 282 742
135 #> which(sort(avgStar[draTop10], decreasing=TRUE) == movies$star[282])
136 #[1] 3
137 #> which(sort(avgStar[draTop10], decreasing=TRUE) == movies$star[742])
138 #[1] 1
139 #> which(sort(avgStar[actTop10], decreasing=TRUE) == movies$star[282])
140 #[1] 2
141 #> which(sort(avgStar[actTop10], decreasing=TRUE) == movies$star[742])

```

```

142 #[1] 1
143 ### Mark 282 as action and 742 as drama
144
145 #> intersect(actTop10, thriTop10)
146 #[1] 966
147 #> which(sort(avgStar[actTop10], decreasing=TRUE) == movies$star[966])
148 #[1] 7
149 #> which(sort(avgStar[thriTop10], decreasing=TRUE) == movies$star[966])
150 #[1] 10
151 ### Mark 966 as action
152
153 #> intersect(romTop10, actTop10)
154 #[1] 414
155 #> which(sort(avgStar[actTop10], decreasing=TRUE) == movies$star[414])
156 #[1] 8
157 #> which(sort(avgStar[romTop10], decreasing=TRUE) == movies$star[414])
158 #[1] 3
159 ### Mark 414 as romance
160
161 movies$genre <- genre
162
163 movies <- movies[union(union(union(union(romTop10, actTop10), chiTop10),
164 draTop10), horrTop10), thriTop10),]
164 rownames(movies) <- 1:nrow(movies)
165 # 55 movies altogether, including
166 #> sum(movies$genre == "romance")
167 #[1] 10
168 #> sum(movies$genre == "action")
169 #[1] 8
170 #> sum(movies$genre == "horror")
171 #[1] 10
172 #> sum(movies$genre == "drama")
173 #[1] 8
174 #> sum(movies$genre == "thriller")
175 #[1] 9
176 #> sum(movies$genre == "children")
177 #[1] 10
178
179 ratings <- ratings[ratings$mid %in% movies$mid,]
180 #> dim(ratings)
181 #[1] 73286      4
182
183 noRatedmovies <- sapply(1:nrow(users), function(i) sum(ratings$uid == users$uid[i]))
184
185 users <- users[which(noRatedmovies > 1),]
186 rownames(users) <- 1:nrow(users)
187
188 ratings <- ratings[ratings$uid %in% users$uid,]
189
190 # After dropping the users who rated 0 or 1 movie...
191 #
192 #> dim(ratings)
193 #[1] 72979      4
194 #> length(unique(ratings$mid))
195 #[1] 55
196 #> length(unique(ratings$uid))
197 #[1] 5625
198 #> dim(users)
199 #[1] 5625      5
200 #> dim(movies)
201 #[1] 55    7
202
203 save(list = ls(all = TRUE), file = "movie.RData")

```

./codes/movieDataCleaning.R

```

1#!/usr/bin/env Rscript
2# Author: Yiming Li

```

```

3 # Four functions for calculating the minimum / maximum
4 # values of the Kendall / Spearman distance.
5 #
6 # !!! Needs the global variables c_k, c_s, and t
7
8 min_k <- function(i, j, noRated, c_k, c_s, t) {
9   # Calculate the minimum value of the Kendall distance
10  # between the joke ranking of the i^th and j^th ranker
11  k1 <- noRated[i]
12  k2 <- noRated[j]
13  if (k1 > k2) {
14    tmp <- k2
15    k2 <- k1
16    k1 <- tmp
17  }
18  if (k1 %% 2 == 0) {
19    return(c_k - k1*(3*k1*k2*(t+1)-(k1^2+2)*(t-k2)-3*(k2+1))/(6*(k1+1)*(k2+1)))
20  } else {
21    return(c_k - (k1-1)*(t*(3*k2-k1)+k2*(k1+3))/(6*(k2+1)))
22  }
23}
24
25 max_k <- function(i, j, noRated, c_k, c_s, t) {
26   # Calculate the maximum value of the Kendall distance
27   # between the joke ranking of the i^th and j^th ranker
28   k1 <- noRated[i]
29   k2 <- noRated[j]
30   if (k1 > k2) {
31     tmp <- k2
32     k2 <- k1
33     k1 <- tmp
34   }
35   if (k1 %% 2 == 0) {
36     return(c_k + k1*(3*k1*k2*(t+1)-(k1^2+2)*(t-k2)-3*(k2+1))/(6*(k1+1)*(k2+1)))
37   } else {
38     return(c_k + (k1-1)*(t*(3*k2-k1)+k2*(k1+3))/(6*(k2+1)))
39   }
40}
41
42 min_s <- function(i, j, noRated, c_k, c_s, t) {
43   # Calculate the minimum value of the Spearman distance
44   # between the joke ranking of the i^th and j^th ranker
45   k1 <- noRated[i]
46   k2 <- noRated[j]
47   if (k1 > k2) {
48     tmp <- k2
49     k2 <- k1
50     k1 <- tmp
51   }
52   if (k1 %% 2 == 0) {
53     return(c_s - (t+1)^2*k1*(k1*(3*k2-k1)-2)/(24*(k2+1)*(k2+1)))
54   } else {
55     return(c_s - (t+1)^2*(k1-1)*(3*k2-k1)/(24*(k2+1)))
56   }
57}
58
59 max_s <- function(i, j, noRated, c_k, c_s, t) {
60   # Calculate the maximum value of the Spearman distance
61   # between the joke ranking of the i^th and j^th ranker
62   k1 <- noRated[i]
63   k2 <- noRated[j]
64   if (k1 > k2) {
65     tmp <- k2
66     k2 <- k1
67     k1 <- tmp
68   }
69   if (k1 %% 2 == 0) {
70     return(c_s + (t+1)^2*k1*(k1*(3*k2-k1)-2)/(24*(k2+1)*(k2+1)))

```

```

71     } else {
72         return(c_s + (t+1)^2*(k1-1)*(3*k2-k1)/(24*(k2+1)))
73     }
74 }
```

./codes/ksMinMaxFuncs.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 # 3955 movies from 6040 users who joined MovieLens in 2000.
8 #
9 # Author: Yiming Li
10 #
11 # Calculate the distance between raters.
12 # This script takes a LOT of time to run and eats up RAM!
13 #
14 load("movie.RData")
15 rm(list = ls()[!ls() %in% c("users", "ratings", "movies")])
16 # We are only using the data frames ratings, users and movies
17 # in the analysis below (5625 users and 55 movies)
18 #
19 movies <- movies[,-(names(movies) %in% c("genres"))]
20 moviemat <- matrix(data = NA, nrow = nrow(users), ncol = nrow(movies))
21 rownames(moviemat) <- users$uid
22 colnames(moviemat) <- movies$mid
23 #
24 noRatedmovies <- sapply(1:nrow(users), function(i) sum(ratings$uid == users$uid[i]))
25 # noRatedmovies is a numeric vector of length 5625
26 # noRatedmovies[i] is the number of movies rated by the user with uid
27 # users$uid[i]
28 #
29 t <- ncol(moviemat) # 55 movies altogether
30 raters <- nrow(moviemat) # 5625 users altogether
31 #
32 percent <- function(x, digits = 2, format = "f") {
33   paste(formatC(100 * x, format = format, digits = digits), "%", sep = "")
34 }
35 #
36 for (i in 1:raters) {
37   for (j in 1:t) {
38     tmp <- ratings$rating[which(ratings$uid == users$uid[i] & ratings$mid == movies$mid[j])]
39     if (length(tmp) != 0) moviemat[i,j] <- as.numeric(tmp)
40   }
41   cat(c("\n", format(Sys.time(), "%H:%M:%S>"), " i =", i, "###", percent(i/5625), " finished!\n"))
42 }
43 # > dim(moviemat)
44 # [1] 5625 55
45 #
46 rankValues <- function(x) {
47   # Rank 1 is the most preferred (given the highest score)
48   # NA would be ranked NA
49   # e.g. rankValues(c(1,4200,0,100,NA,-9.99,100)) returns
50   # [1] 4 1 5 2 NA 6 2
51   return(rank(-x, na.last = "keep", ties.method = "min"))
52 }
53 moviemat <- t(matrix(apply(moviemat, 1, rankValues), nrow = t))
54 # Now moviemat is a 5625*55 matrix of integers, with the i-th row containing
55 # the ranking of movies by the i-th user
56 # As above, we use NA to represent missing data
57 #
58 c_s <- t * (t^2 - 1) / 12 # Mean of Spearman distance
59 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,
87                 "j =", j, "####", percent(p/5625), "finished!\n"))
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, "####",
107             percent(p/5625), "finished!\n"))
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,
117                               noRatedmovies, c_k, c_s, t))
118     D_k3[q,p] = D_k3[p,q]
119     if (p %% 5 == 0 && q == p) {
120       cat(c("\n[3rd]", format(Sys.time(), "%H:%M:%S>"), "p =", p, "q =", q, "####",
121             percent(p/5625), "finished!\n"))
122     }
123 # D_k3 is yet another version of the Kendall distance matrix, of dimension 5625 * 5625

```

```

124 # Here, we use (d - m^*) / (M^* - m^*) instead of the original d
125
126 mds <- cmdscale(D_k)
127 mds2 <- cmdscale(D_k2)
128 mds3 <- cmdscale(D_k3)
129
130 save(list = ls(all = TRUE), file = "movie2.RData")

```

./codes/movieDistanceCalc.R

```

1#!/usr/bin/env Rscript
2# Author: Yiming Li
3# 1. Make the hexagonal binning MDS plot.
4# 2. Make MDS plots with contour curves colored by genders.
5# 3. Make the kernel smoothing MDS plot.
6#
7# Requires loading the packages specified in movieVisualization.R!
8
9 plotByGender <- function(users, mds, xl, yl) {
10   mds.female <- data.frame(mds[which(users$gender == "F"),])
11   mds.male <- data.frame(mds[which(users$gender == "M"),])
12
13   mds.hex <- ggplot(data.frame(mds), aes(x=X1, y=X2)) + stat_binhex() + ggtitle("MDS
      Plot With Hexagonal Binning")
14
15   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")
16
17   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")
18
19   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")
20
21   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")
22
23   grid.arrange(mds.hex)
24   grid.arrange(mds.cont)
25   grid.arrange(mds.both)
26   grid.arrange(pfemale, pmale, ncol = 2)
27
28   smoothScatter(mds)
29 }
30
31 pdf("bygender2.pdf", width = 10, height = 7)
32
33 plotByGender(users, mds3, max(abs(range(mds3[,1]))), max(abs(range(mds3[,2]))))
34
35 dev.off()

```

./codes/hex+smooth+compareGender.R

```

1#!/usr/bin/env Rscript
2# Author: Yiming Li
3#
4# Color the MDS plot by age groups
5# Requires loading the packages specified in movieVisualization.R!
6
7 pdf("byage.pdf", width = 14, height = 7)
8 pal <- brewer.pal(7, "YlOrRd")
9 # Blues BuGn BuPu GnBu Greens Greys Oranges

```

```

10 # OrRd PuBu PuBuGn PuRd Purples RdPu Reds
11 # YlGn YlGnBu YlOrBr YlOrRd
12
13 # grays <- gray.colors(7, start = 0, end = 0.5, gamma = 2.2)
14 agecol <- ifelse(users$age == "1", pal[1], ifelse(users$age == "18", pal[2], ifelse(
15   users$age == "25", pal[3], ifelse(users$age == "35", pal[4], ifelse(users$age == "
16   45", pal[5], ifelse(users$age == "50", pal[6], pal[7])))))
17
18 age3 <- ggplot(data.frame(mds3),aes(x=X1,y=X2)) + geom_point(col = agecol, alpha =
19   0.6) + ggtitle("Normalized Distances Between Movie Raters -- Multi-Dimensional
20   Scaling\n* Using  $(d - m^*) / (M^* - m^*)$  instead of the original  $d$ ")
21
22 grid.arrange(age3)
23
24 ### Non-ggplot version
25 plot(mds3, main = "Normalized Distances Between Movie Raters -- Multi-Dimensional
26   Scaling\n* Using  $(d - m^*) / (M^* - m^*)$  instead of the original  $d$ ", xlab =
27   "Coordinate 1", ylab = "Coordinate 2", col = agecol, pch = "+")
28
29 dev.off()

```

./codes/compareAge.R

```

1#!/usr/bin/env Rscript
2# Author: Yiming Li
3# Demonstrates the relationship between jobs and movie preferences.
4# Results not shown in the report.
5#
6# Requires loading the packages specified in movieVisualization.R!
7
8pdf("byjob2.pdf", width = 14, height = 7)
9
10totJobs <- c(433, 219, 145, 645, 93, 177, 544, 13, 71, 161, 113, 320, 104, 250, 124,
11 197, 406, 56, 63, 247, 0)
12
13xl <- c(-max(abs(range(mds3[,1])), max(abs(range(mds3[,1]))))
14yl <- c(-max(abs(range(mds3[,2])), max(abs(range(mds3[,2])))))
15
16plot(mds3[which(users$job == "5"),], main = "Student", xlab = "Coordinate 1", ylab = "
17  Coordinate 2", pch = "+", col = cols[1], xlim = xl, ylim = yl)
18
19plot(mds3[which(users$job == "5"),], main = "Student + Artist", xlab = "Coordinate 1",
20  ylab = "Coordinate 2", pch = "+", col = cols[1], xlim = xl, ylim = yl)
21points(mds3[which(users$job == "2"),], pch = "+", col = cols[2], xlim = xl, ylim = yl)
22
23plot(mds3[which(users$job == "5"),], main = "Student + Artist + Engineer", xlab = "
24  Coordinate 1", ylab = "Coordinate 2", pch = "+", col = cols[1], xlim = xl, ylim =
25  yl)
26points(mds3[which(users$job == "2"),], pch = "+", col = cols[2], xlim = xl, ylim = yl)
27points(mds3[which(users$job == "17"),], pch = "+", col = cols[3], xlim = xl, ylim = yl
28  )
29
30plot(mds3[which(users$job == "5"),], main = "Student + Artist + Engineer + Farmer",
31  xlab = "Coordinate 1", ylab = "Coordinate 2", pch = "+", col = cols[1], xlim = xl,
32  ylim = yl)
33points(mds3[which(users$job == "2"),], pch = "+", col = cols[2], xlim = xl, ylim = yl)
34points(mds3[which(users$job == "17"),], pch = "+", col = cols[3], xlim = xl, ylim = yl
35  )
36points(mds3[which(users$job == "8"),], pch = "+", col = cols[4], xlim = xl, ylim = yl)
37
38plot(mds3[which(users$job == "5"),], main = "Student + Artist + Engineer + Farmer +
39  Lawyer", xlab = "Coordinate 1", ylab = "Coordinate 2", pch = "+", col = cols[1],
40  xlim = xl, ylim = yl)
41points(mds3[which(users$job == "2"),], pch = "+", col = cols[2], xlim = xl, ylim = yl)
42points(mds3[which(users$job == "17"),], pch = "+", col = cols[3], xlim = xl, ylim = yl
43  )
44points(mds3[which(users$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 +
36 Lawyer + Clerical", xlab = "Coordinate 1", ylab = "Coordinate 2", pch = "+", col =
37 cols[1], xlim = xl, ylim = yl)
38 points(mds3[which(users$job == "2"),], pch = "+", col = cols[2], xlim = xl, ylim = yl)
39 points(mds3[which(users$job == "17"),], pch = "+", col = cols[3], xlim = xl, ylim = yl
40 )
41 points(mds3[which(users$job == "8"),], pch = "+", col = cols[4], xlim = xl, ylim = yl)
42 points(mds3[which(users$job == "11"),], pch = "+", col = cols[5], xlim = xl, ylim = yl
43 )
44 points(mds3[which(users$job == "3"),], pch = "+", col = cols[6], xlim = xl, ylim = yl)
45 plot(mds3[which(users$job == "5"),], main = "Student + Artist + Engineer + Farmer +
46 Lawyer + Clerical + Doctor", xlab = "Coordinate 1", ylab = "Coordinate 2", pch = "
47 +", col = cols[1], xlim = xl, ylim = yl)
48 points(mds3[which(users$job == "2"),], pch = "+", col = cols[2], xlim = xl, ylim = yl)
49 points(mds3[which(users$job == "17"),], pch = "+", col = cols[3], xlim = xl, ylim = yl
50 )
51 points(mds3[which(users$job == "8"),], pch = "+", col = cols[4], xlim = xl, ylim = yl)
52 points(mds3[which(users$job == "11"),], pch = "+", col = cols[5], xlim = xl, ylim = yl
53 )
54 points(mds3[which(users$job == "3"),], pch = "+", col = cols[6], xlim = xl, ylim = yl)
55 points(mds3[which(users$job == "6"),], pch = "+", col = cols[7], xlim = xl, ylim = yl)
56 points(mds3[which(users$job == "7"),], pch = "+", col = cols[8], xlim = xl, ylim = yl)
57 plot(mds3[which(users$job == "5"),], main = "Student + Artist + Engineer + Farmer +
58 Lawyer + Clerical + Doctor + Manager", xlab = "Coordinate 1", ylab = "Coordinate 2
59 ", 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 )
63 points(mds3[which(users$job == "8"),], pch = "+", col = cols[4], xlim = xl, ylim = yl)
64 points(mds3[which(users$job == "11"),], pch = "+", col = cols[5], xlim = xl, ylim = yl
65 )
66 points(mds3[which(users$job == "3"),], pch = "+", col = cols[6], xlim = xl, ylim = yl)
67 points(mds3[which(users$job == "6"),], pch = "+", col = cols[7], xlim = xl, ylim = yl)
68 points(mds3[which(users$job == "7"),], pch = "+", col = cols[8], xlim = xl, ylim = yl)
69 points(mds3[which(users$job == "14"),], pch = "+", col = cols[9], xlim = xl, ylim = yl
70 )
71 plot(mds3[which(users$job == "5"),], main = "Student + Artist + Engineer + Farmer +
72 Lawyer + Clerical + Doctor + Manager + Sales", xlab = "Coordinate 1",
73 ylab = "Coordinate 2", pch = "+", col = cols[1], xlim = xl, ylim = yl)
74 points(mds3[which(users$job == "2"),], pch = "+", col = cols[2], xlim = xl, ylim = yl)
75 points(mds3[which(users$job == "17"),], pch = "+", col = cols[3], xlim = xl, ylim = yl
76 )
77 points(mds3[which(users$job == "8"),], pch = "+", col = cols[4], xlim = xl, ylim = yl)
78 points(mds3[which(users$job == "11"),], pch = "+", col = cols[5], xlim = xl, ylim = yl
79 )
80 points(mds3[which(users$job == "3"),], pch = "+", col = cols[6], xlim = xl, ylim = yl)
81 points(mds3[which(users$job == "6"),], pch = "+", col = cols[7], xlim = xl, ylim = yl)
82 points(mds3[which(users$job == "7"),], pch = "+", col = cols[8], xlim = xl, ylim = yl)
83 points(mds3[which(users$job == "14"),], pch = "+", col = cols[9], xlim = xl, ylim = yl
84 )

```

```

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$uid[i],
49   , "rating"])))
50 } )
51 # actionAvg[i] is the average stars the i^th user in users gave to all action movies (
52 # he rated
53
54 childrenAvg <- sapply(1:nrow(users), function(i) {
55   return(mean(as.integer(ratings[ratings$mid %in% children & ratings$uid == users$uid[i],
56   , "rating"])))
57 } )
58
59 dramaAvg <- sapply(1:nrow(users), function(i) {
60   return(mean(as.integer(ratings[ratings$mid %in% drama & ratings$uid == users$uid[i],
61   , "rating"])))
62 }

```

```

58  })
59
60 horrorAvg <- sapply(1:nrow(users), function(i) {
61   return(mean(as.integer(ratings[ratings$mid %in% horror & ratings$uid == users$uid[i], "rating"])))
62 })
63
64 romanceAvg <- sapply(1:nrow(users), function(i) {
65   return(mean(as.integer(ratings[ratings$mid %in% romance & ratings$uid == users$uid[i], "rating"])))
66 })
67
68 thrillerAvg <- sapply(1:nrow(users), function(i) {
69   return(mean(as.integer(ratings[ratings$mid %in% thriller & ratings$uid == users$uid[i], "rating"])))
70 })
71
72 actionNo <- sapply(1:nrow(users), function(i) {
73   return(nrow(ratings[ratings$mid %in% action & ratings$uid == users$uid[i],]))
74 })
75
76 childrenNo <- sapply(1:nrow(users), function(i) {
77   return(nrow(ratings[ratings$mid %in% children & ratings$uid == users$uid[i],]))
78 })
79
80 dramaNo <- sapply(1:nrow(users), function(i) {
81   return(nrow(ratings[ratings$mid %in% drama & ratings$uid == users$uid[i],]))
82 })
83
84 horrorNo <- sapply(1:nrow(users), function(i) {
85   return(nrow(ratings[ratings$mid %in% horror & ratings$uid == users$uid[i],]))
86 })
87
88 romanceNo <- sapply(1:nrow(users), function(i) {
89   return(nrow(ratings[ratings$mid %in% romance & ratings$uid == users$uid[i],]))
90 })
91
92 thrillerNo <- sapply(1:nrow(users), function(i) {
93   return(nrow(ratings[ratings$mid %in% thriller & ratings$uid == users$uid[i],]))
94 })
95
96 users$action <- actionAvg
97 users$children <- childrenAvg
98 users$drama <- dramaAvg
99 users$horror <- horrorAvg
100 users$romance <- romanceAvg
101 users$thriller <- thrillerAvg
102
103 users$actionNo <- actionNo
104 users$childrenNo <- childrenNo
105 users$dramaNo <- dramaNo
106 users$horrorNo <- horrorNo
107 users$romanceNo <- romanceNo
108 users$thrillerNo <- thrillerNo

```

./codes/markUsers.R

```

1 #!/usr/bin/env Rscript
2 # Author: Yiming Li
3 # Functions for making MDS plots for different "fan groups".
4 #
5 # Requires loading the packages specified in movieVisualization.R!
6
7 ### Six pairs of colors for the genres
8 gCols <- brewer.pal(12, "Paired")
9
10 # Return value <- User is of the group:
11 # 1 <- action

```

```

12  # 2 <- children
13  # 3 <- drama
14  # 4 <- horror
15  # 5 <- romance
16  # 6 <- thriller
17
18 ### Each function returns the data frame for a certain genre
19 mds.rom <- function(users, mds) {
20   data.frame(mds[which(users$ggroup == 5),])
21 }
22 mds.act <- function(users, mds) {
23   data.frame(mds[which(users$ggroup == 1),])
24 }
25 mds.hor <- function(users, mds) {
26   data.frame(mds[which(users$ggroup == 4),])
27 }
28 mds.thr <- function(users, mds) {
29   data.frame(mds[which(users$ggroup == 6),])
30 }
31 mds.chi <- function(users, mds) {
32   data.frame(mds[which(users$ggroup == 2),])
33 }
34 mds.dra <- function(users, mds) {
35   data.frame(mds[which(users$ggroup == 3),])
36 }
37
38 ### Density heatmap; uses genre as argument
39
40 plotGenre <- function(users, genre) {
41   #actioncol <- ifelse((users$action > mean(users$action, na.rm = TRUE) & !is.na(users
42   # $action)), "blue", "black")
43   if (genre == "action") {
44     ggplot(mds.act(users, mds3), aes(x=X1, y=X2)) + stat_density2d(aes(fill=..level..),
45       geom="polygon") + scale_fill_gradient(low=gCols[1], high=gCols[2]) + ggtitle("Action")
46   } else if (genre == "children") {
47     ggplot(mds.chi(users, mds3), aes(x=X1, y=X2)) + stat_density2d(aes(fill=..level..),
48       geom="polygon") + scale_fill_gradient(low=gCols[7], high=gCols[8]) + ggtitle("Children")
49   } else if (genre == "drama") {
50     ggplot(mds.dra(users, mds3), aes(x=X1, y=X2)) + stat_density2d(aes(fill=..level..),
51       geom="polygon") + scale_fill_gradient(low=gCols[11], high=gCols[12]) + ggtitle("Drama")
52   } else if (genre == "horror") {
53     ggplot(mds.hor(users, mds3), aes(x=X1, y=X2)) + stat_density2d(aes(fill=..level..),
54       geom="polygon") + scale_fill_gradient(low=gCols[3], high=gCols[4]) + ggtitle("Horror")
55   } else if (genre == "romance") {
56     ggplot(mds.rom(users, mds3), aes(x=X1, y=X2)) + stat_density2d(aes(fill=..level..),
57       geom="polygon") + scale_fill_gradient(low=gCols[5], high=gCols[6]) + ggtitle("Romance")
58   } else if (genre == "thriller") {
59     ggplot(mds.thr(users, mds3), aes(x=X1, y=X2)) + stat_density2d(aes(fill=..level..),
60       geom="polygon") + scale_fill_gradient(low=gCols[9], high=gCols[10]) + ggtitle("Thriller")
61   }
62 }
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
737
738
739
739
740
741
742
743
744
745
745
746
747
747
748
749
749
750
751
752
753
754
755
755
756
757
757
758
759
759
760
761
762
763
764
764
765
766
766
767
767
768
768
769
769
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
```

```

64  ggplot(mds.act(users, mds),aes(x=X1,y=X2)) + geom_point(alpha = 0.5, col = gCols[1])
65  + geom_density2d(col = gCols[2]) + xlim(-xl, xl) + ylim(-yl, yl) + ggtitle("Action")
66 }
67 pHor <- function(users, mds, xl, yl) {
68  ggplot(mds.hor(users, mds),aes(x=X1,y=X2)) + geom_point(alpha = 0.5, col = gCols[3])
69  + geom_density2d(col = gCols[4]) + xlim(-xl, xl) + ylim(-yl, yl) + ggtitle("Horror")
70 }
71 pChi <- function(users, mds, xl, yl) {
72  ggplot(mds.chi(users, mds),aes(x=X1,y=X2)) + geom_point(alpha = 0.5, col = gCols[7])
73  + geom_density2d(col = gCols[8]) + xlim(-xl, xl) + ylim(-yl, yl) + ggtitle("Children")
74 }
75 pDra <- function(users, mds, xl, yl) {
76  ggplot(mds.dra(users, mds),aes(x=X1,y=X2)) + geom_point(alpha = 0.5, col = gCols[11])
77  + geom_density2d(col = gCols[12]) + xlim(-xl, xl) + ylim(-yl, yl) +
78  ggtitle("Drama")
79 }
80 pThr <- function(users, mds, xl, yl) {
81  ggplot(mds.thr(users, mds),aes(x=X1,y=X2)) + geom_point(alpha = 0.5, col = gCols[9])
82  + geom_density2d(col = gCols[10]) + xlim(-xl, xl) + ylim(-yl, yl) + ggtitle("Thriller")
83 }
84 #### For comparing romance and action
85 compareRomAct <- function(users, mds) {
86  xl <- ceiling(max(abs(range(mds[,1]))))
87  yl <- ceiling(max(abs(range(mds[,2]))))
88
89  mds.both <- ggplot(mds.rom(users, mds),aes(x=X1,y=X2)) + geom_point(alpha = 0.5, col
90  = gCols[5]) + geom_density2d(col = gCols[6], alpha = 0.8, lwd = 0.8) + geom_
91  point(data = mds.act(users, mds), alpha = 0.5, col = gCols[1]) + geom_density2d(
92  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")
93
94  grid.arrange(mds.both)
95  grid.arrange(pRom(users,mds,xl,yl), pAct(users,mds,xl,yl), ncol = 2)
96 }
97 romact <- function() {
98  pdf("new-romAct.pdf", width = 10, height = 7)
99  compareRomAct(users, mds3)
100 dev.off()
101 }
102 #### For comparing all six genres
103 allsix <- function(mds) {
104  xl <- max(abs(range(mds[,1])))
105  yl <- max(abs(range(mds[,2])))
106  mds.allsix <- ggplot(mds.rom(users, mds),aes(x=X1,y=X2)) + geom_point(alpha = 0.5,
107  col = gCols[5]) + geom_density2d(col = gCols[6], alpha = 0.8, lwd = 0.8) + geom_
108  point(data = mds.act(users, mds), alpha = 0.5, col = gCols[1]) + geom_density2d(
109  data = mds.act(users, mds), col = gCols[2], alpha = 0.8, lwd = 0.8) + geom_point(
110  data = mds.hor(users, mds), alpha = 0.5, col = gCols[3]) + geom_density2d(data =
111  mds.hor(users, mds), col = gCols[4], alpha = 0.8, lwd = 0.8) + geom_point(data =
112  mds.thr(users, mds), alpha = 0.5, col = gCols[9]) + geom_density2d(data = mds.
113  thr(users, mds), col = gCols[10], alpha = 0.8, lwd = 0.8) + geom_point(data =
114  mds.dra(users, mds), alpha = 0.5, col = gCols[11]) + geom_density2d(data = mds.
115  dra(users, mds), col = gCols[12], alpha = 0.8, lwd = 0.8) + geom_point(data =
116  mds.chi(users, mds), alpha = 0.5, col = gCols[7]) + geom_density2d(data = mds.
117  chi(users, mds), col = gCols[8], alpha = 0.8, lwd = 0.8) + ggtitle("MDS Plot"

```

```

107     With Density Contour Lines")
108
109     grid.arrange(mds.allsix)
110     grid.arrange(pAct(users,mds,xl,yl), pChi(users,mds,xl,yl), pDra(users,mds,xl,yl),
111       pHor(users,mds,xl,yl), pRom(users,mds,xl,yl), pThr(users,mds,xl,yl), ncol = 3)
112   }
113
114 plotAllSix <- function() {
115   pdf("new-allsix.pdf", width = 10, height = 7)
116   allsix(mds3)
117   dev.off()
118 }
119
120 plotAllSixHeat <- function() {
121   pdf("new-allsixheat.pdf", width = 10, height = 7)
122   grid.arrange(plotGenre(users, "action"))
123   grid.arrange(plotGenre(users, "children"))
124   grid.arrange(plotGenre(users, "drama"))
125   grid.arrange(plotGenre(users, "horror"))
126   grid.arrange(plotGenre(users, "romance"))
127   grid.arrange(plotGenre(users, "thriller"))
128   grid.arrange(plotGenre(users, "action"),plotGenre(users, "children"),plotGenre(users,
129     "drama"),plotGenre(users, "horror"),plotGenre(users, "romance"),plotGenre(
130       users, "thriller"),ncol = 3)
131   dev.off()
132 }

```

./codes/compareGenres.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 # The driver script for all the visualization (except for the representative rank).
12 #
13 load("movie2.RData")
14 rm(list = ls()[!ls() %in% c("users", "ratings", "movies", "mds3")])
15 #> dim(ratings)
16 #[1] 72979      4
17 #> dim(users)
18 #[1] 5625      5
19 #> dim(movies)
20 #[1] 55      6
21 #
22 source("markUsers.R")
23 # Now "users" has the additional columns
24 #   action, children, drama, horror, romance and thriller
25 # which is the average score each user gave to that genre,
26 # as well as the additional columns
27 #   actionNo, childrenNo, dramaNo, horrorNo, romanceNo and thrillerNo
28 # which is the number of movies of that genre each user rated.
29 save(list = ls(all = TRUE), file = "movie3.RData")
30 #
31 # load("movie3.RData")
32 rm(list = ls()[!ls() %in% c("users", "ratings", "movies", "mds", "mds2", "mds3")])
33 #
34 #> dim(movies)
35 #[1] 55      6
36 #> dim(ratings)
37 #[1] 72979      4
38 #> dim(users)
39 #[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.
75       rm = TRUE))]
76     return(ifelse((length(indices) == 1), indices, sample(indices, 1)))
77   }
78 })
79
80 #> sum(genreGroup == 1)
81 #[1] 1425
82 #> sum(genreGroup == 2)
83 #[1] 601
84 #> sum(genreGroup == 3)
85 #[1] 1194
86 #> sum(genreGroup == 4)
87 #[1] 424
88 #> sum(genreGroup == 5)
89 #[1] 805
90 #> sum(genreGroup == 6)
91 #[1] 1176
92
93 allNA <- sapply(1:nrow(users), function(i) {
94   # sum(is.na(users[i,6:11])) == 6
95 })
96 ##> sum(allNA)
97 ##[1] 0
98
99 users$ggroup <- genreGroup
100
101 save(list = ls(all = TRUE), file = "movie4.RData")
102
103 # load("movie4.RData")
104 rm(list = ls()[!ls() %in% c("users", "ratings", "movies", "mds", "mds2", "mds3")])
105 users <- users[,names(users) %in% c("uid", "gender", "age", "job", "zip", "ggroup")]
106

```

```

107 source("compareGenres.R")
108 romact()
109 plotAllSix()
110 plotAllSixHeat()
111
112 pdf("heat.pdf", width = 10, height = 7)
113
114 grid.arrange(ggplot(data.frame(mds3),aes(x=X1,y=X2)) + stat_density2d(aes(fill=..level
115     ..), 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))
116 dev.off()
117
```

./codes/movieVisualization.R

```

1#!/usr/bin/env Rscript
2# Author: Yiming Li
3#
4# 1. Identify the local maximum density regions.
5# 2. Find the representative complete ranking.
6# 3. Visualize the results.
7
8library("KernSmooth")
9library("ggplot2")
10library("gridExtra")
11library("RColorBrewer")
12
13### Preliminary data / functions loading
14load("movie2.RData")
15load("movie4.RData")
16rm(list = ls()[!ls() %in% c("users", "ratings", "movies", "mds", "mds2", "mds3", "
moviemat")])
17
18#> dim(mds3)
19#[1] 5625      2
20#> dim(moviemat)
21#[1] 5625      55
22#> dim(movies)
23#[1] 55      6
24#> dim(ratings)
25#[1] 72979      4
26#> dim(users)
27#[1] 5625      18
28
29inrange <- function(pt, rangex, rangey) {
30  return (pt[1] >= rangex[1] & pt[1] <= rangex[2] & pt[2] >= rangey[1] & pt[2] <=
rangey[2])
31}
32
33func_a2 <- function(r, i, j) {
34  # (12) in "Rank correlation methods for missing data"
35  # For a given pair of movies (i,j), 1 <= i < j <= t
36  # and an incomplete ranking r, calculate the score
37  k <- sum(!is.na(r))
38  if (is.na(r[i]) & is.na(r[j])) {
39    return(0)
40  } else if (is.na(r[i]) & !is.na(r[j])) {
41    return(2 * r[j] / (k + 1) - 1)
42  } else if (is.na(r[j])) {
43    return(1 - 2 * r[i] / (k + 1))
```

```

44 } else {
45   return(sign(r[j] - r[i]))
46 }
47 }
48
49 k_dist <- function(r1, r2, c_k, t) {
50   ## For two incomplete rankings, calculate the
51   ### Kendall distance between them
52   kdis <- c_k
53   for (i in 1:(t-1)) {
54     for (j in (i+1):t) {
55       kdis <- kdis - func_a2(r1,i,j)*func_a2(r2,i,j)
56     }
57   }
58   return(kdis)
59 }
60
61 est <- bkde2D(mds3, bandwidth = rep(min(dpik(mds3[,1]), dpik(mds3[,2])), 2), gridsize
62   = c(800,800))
63
64 pdf("tmp.pdf")
65
66 contour(est$x1, est$x2, est$fhat, title = "MDS Plot With Density Contour Lines --
67   Normalized Distance")
68 grid(100,100, lwd = 0.5)
69
70 # left, bottom, right, top
71 rect(-0.1, 0.07, -0.08, 0.09, border = "magenta")
72 rect(0.11, -0.11, 0.125, -0.09, border = "blue")
73 rect(0.135, -0.005, 0.155, 0.015, border = "green")
74 rect(-0.04, -0.24, -0.035, -0.22, border = "purple")
75 rect(-0.075, -0.09, -0.065, -0.08, border = "cyan")
76
77 # abline(h = c(0.1,-0.1), v = c(0.1,-0.1), col = "pink")
78
79 dev.off()
80
81 #### 30 raters in the highest density region (pink)
82 group0 <- apply(mds3, 1, function(x) inrange(x, c(-0.1,-0.08), c(0.07,0.09)))
83 #### group0 is a boolean vector of length 5625
84
85 #### 13 raters in local maximum 1 (blue)
86 group1 <- apply(mds3, 1, function(x) inrange(x, c(0.11,0.125), c(-0.11,-0.09)))
87
88 #### 18 raters in local maximum 2 (green)
89 group2 <- apply(mds3, 1, function(x) inrange(x, c(0.135,0.155), c(-0.005,0.015)))
90
91 #### 5 raters in local maximum 3 (purple)
92 group3 <- apply(mds3, 1, function(x) inrange(x, c(-0.04,-0.035), c(-0.24,-0.22)))
93
94 #### 7 raters in local maximum 4 (cyan)
95 group4 <- apply(mds3, 1, function(x) inrange(x, c(-0.075,-0.065), c(-0.09,-0.08)))
96
97 #### Check the relative locations of the density local maximums
98 # ggplot(data.frame(mds3),aes(x=X1,y=X2)) + geom_density2d(alpha = 0.8, lwd = 0.8) +
99   geom_point(data = data.frame(mds3[group0,]), alpha = 0.9, col = "pink", pch = "+")
100  + geom_point(data = data.frame(mds3[group1,]), alpha = 0.9, col = "blue", pch =
101    "+") + geom_point(data = data.frame(mds3[group2,]), alpha = 0.9, col = "green",
102    pch = "+") + geom_point(data = data.frame(mds3[group3,]), alpha = 0.9, col =
103    "purple", pch = "+") + geom_point(data = data.frame(mds3[group4,]), alpha = 0.9,
104    col = "cyan", pch = "+")
105
106 #### Now we have 5 groups (group0 -- group4).
107 #### We would like to find a complete ranking r for each group s.t.
108 #### the sum of the distances from r to each ranking in the group
109 #### is minimized.
110 #### To do this, we use a bubble-sort-like approach.

```

```

104 t <- 55 # Number of movies
105 c_k <- t * (t - 1) / 2 # Mean of Kendall distance
106
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_
153         _k, t)) {
154         r <- rTemp
155         changed <- TRUE
156       }
157     }
158     print(count)
159     print(totalDist(r, group, moviemat, c_k, t))
160     print(r)
161     if (!changed) break;
162     if (count <= 1) break;
163   }
164
165   return(r)
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

```

```

171 #[51] 3 35 36 19 43
172
173 #totalDist(r0, moviemat, c_k, t)
174 #[1] 41724.67
175 #totalDist(1:55, group0, moviemat, c_k, t)
176 #[1] 44431.22
177
178 r1 <- findRepRank(group1, moviemat, c_k, t) # BLUE
179 # Stopped at count = 5
180 # [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
181 #[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
182 #[51] 41 44 52 46 6
183
184 #totalDist(r1, group1, moviemat, c_k, t)
185 #[1] 15766.23
186 #totalDist(1:55, group1, moviemat, c_k, t)
187 #[1] 19120.37
188
189 r2 <- findRepRank(group2, moviemat, c_k, t) # GREEN
190 # Stopped at count = 7
191 # [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
192 #[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
193 #[51] 50 51 46 53 8
194
195 #totalDist(r2, group2, moviemat, c_k, t)
196 #[1] 22988.19
197 #totalDist(1:55, group2, moviemat, c_k, t)
198 #[1] 25955.29
199
200 r3 <- findRepRank(group3, moviemat, c_k, t) # PURPLE
201 # Stopped at count = 1
202 # [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
203 #[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
204 #[51] 50 51 52 53 1
205
206 #totalDist(r3, group3, moviemat, c_k, t)
207 #[1] 6741.481
208 #totalDist(1:55, group3, moviemat, c_k, t)
209 #[1] 7549.111
210
211 r4 <- findRepRank(group4, moviemat, c_k, t) # cyan
212 # Stopped at count = 12
213 # [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
214 #[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
215 #[51] 48 49 16 50 51
216
217 #totalDist(r4, group4, moviemat, c_k, t)
218 #[1] 9447.143
219 #totalDist(1:55, group4, moviemat, c_k, t)
220 #[1] 10307.38
221
222 save(list = ls(all = TRUE), file = "lms.RData")
223
224 gCols <- brewer.pal(12, "Paipink")
225
226 genreToCol <- function(g) {
227   if (g == "action") {
228     return(gCols[2])
229   } else if (g == "children") {
230     return(gCols[8])
231   } else if (g == "drama") {
232     return(gCols[12])
233   } else if (g == "horror") {
234     return(gCols[4])
235   } else if (g == "romance") {
236     return(gCols[6])
237   } else if (g == "thriller") {
238     return(gCols[10])

```

```

239 }
240 }
241
242 ### Plot the color bars and grayscale bars
243 ### for each representative ranking
244 pdf("peaks.pdf", width = 10, height = 7)
245
246 # Peak A
247 r0cols <- sapply(1:55, function(x) {
248   genreToCol(movies$genre[which(r0 == x)])
249 })
250
251 r0titles <- sapply(1:55, function(x) {
252   movies$title[which(r0 == x)]
253 })
254
255 grays <- gray.colors(78, start = 0, end = 0.5, gamma = 2.2)
256 r0years <- sapply(1:55, function(x) {
257   grays[movies$year[which(r0 == x)]-1921]
258 })
259
260 plot(1:55,xlim = c(0,58),ylim = c(-0.1,5), type = "n", xaxt='n', , yaxt = "n", ann=
261 FALSE)
262 rect(1:55,rep(0.8,55) ,2:56,rep(1.3,55),col = r0cols, border=NA)
263 rect(1:55,rep(0,55) ,2:56,rep(0.5,55),col = r0years, border=NA)
264 text(seq(1.5,55.5),rep(2.3,55),labels = r0titles, srt = 90,cex = 0.8)
265 abline(v = c(6,11,21),col = "pink")
266
267 # Peak B
268 r1cols <- sapply(1:55, function(x) {
269   genreToCol(movies$genre[which(r1 == x)])
270 })
271
272 r1titles <- sapply(1:55, function(x) {
273   movies$title[which(r1 == x)]
274 })
275
276 grays <- gray.colors(78, start = 0, end = 0.5, gamma = 2.2)
277 r1years <- sapply(1:55, function(x) {
278   grays[movies$year[which(r1 == x)]-1921]
279 })
280
281 plot(1:55,xlim = c(0,58),ylim = c(-0.1,5), type = "n", xaxt='n', , yaxt = "n", ann=
282 FALSE)
283 rect(1:55,rep(0.8,55) ,2:56,rep(1.3,55),col = r1cols, border=NA)
284 rect(1:55,rep(0,55) ,2:56,rep(0.5,55),col = r1years, border=NA)
285 text(seq(1.5,55.5),rep(2.3,55),labels = r1titles, srt = 90,cex = 0.8)
286 abline(v = c(6,11,21),col = "pink")
287
288 # Peak C
289 r2cols <- sapply(1:55, function(x) {
290   genreToCol(movies$genre[which(r2 == x)])
291 })
292
293 r2titles <- sapply(1:55, function(x) {
294   movies$title[which(r2 == x)]
295 })
296
297 grays <- gray.colors(78, start = 0, end = 0.5, gamma = 2.2)
298 r2years <- sapply(1:55, function(x) {
299   grays[movies$year[which(r2 == x)]-1921]
300 })
301
302 plot(1:55,xlim = c(0,58),ylim = c(-0.1,5), type = "n", xaxt='n', , yaxt = "n", ann=
303 FALSE)
304 rect(1:55,rep(0.8,55) ,2:56,rep(1.3,55),col = r2cols, border=NA)
305 rect(1:55,rep(0,55) ,2:56,rep(0.5,55),col = r2years, border=NA)
306 text(seq(1.5,55.5),rep(2.3,55),labels = r2titles, srt = 90,cex = 0.8)

```

```

304 abline(v = c(6,11,21),col = "pink")
305
306 # Peak D
307 r3cols <- sapply(1:55, function(x) {
308   genreToCol(movies$genre[which(r3 == x)])
309 })
310
311 r3titles <- sapply(1:55, function(x) {
312   movies$title[which(r3 == x)]
313 })
314
315 grays <- gray.colors(78, start = 0, end = 0.5, gamma = 2.2)
316 r3years <- sapply(1:55, function(x) {
317   grays[movies$year[which(r3 == x)]-1921]
318 })
319
320 plot(1:55,xlim = c(0,58),ylim = c(-0.1,5), type = "n", xaxt='n', , yaxt = "n", ann=
321   FALSE)
322 rect(1:55,rep(0.8,55),2:56,rep(1.3,55),col = r3cols, border=NA)
323 rect(1:55,rep(0,55),2:56,rep(0.5,55),col = r3years, border=NA)
324 text(seq(1.5,55.5),rep(2.3,55),labels = r3titles, srt = 90,cex = 0.8)
325 abline(v = c(6,11,21),col = "pink")
326
327 # Peak E
328 r4cols <- sapply(1:55, function(x) {
329   genreToCol(movies$genre[which(r4 == x)])
330 })
331
332 r4titles <- sapply(1:55, function(x) {
333   movies$title[which(r4 == x)]
334 })
335
336 grays <- gray.colors(78, start = 0, end = 0.5, gamma = 2.2)
337 r4years <- sapply(1:55, function(x) {
338   grays[movies$year[which(r4 == x)]-1921]
339 })
340
341 plot(1:55,xlim = c(0,58),ylim = c(-0.1,5), type = "n", xaxt='n', , yaxt = "n", ann=
342   FALSE)
343 rect(1:55,rep(0.8,55),2:56,rep(1.3,55),col = r4cols, border=NA)
344 rect(1:55,rep(0,55),2:56,rep(0.5,55),col = r4years, border=NA)
345 text(seq(1.5,55.5),rep(2.3,55),labels = r4titles, srt = 90,cex = 0.8)
346 abline(v = c(6,11,21),col = "pink")
347
348 dev.off()

```

./codes/localMaximums.R

```

1#!/usr/bin/env Rscript
2# Author: Yiming Li
3#
4# Determine the goodness of fit of the MDS when k = 1, 2, 3 and 4.
5
6library("vegan")
7
8### Preliminary data / functions loading
9load("movie2.RData")
10
11mds3_k1 <- cmdscale(D_k3, k = 1, eig = TRUE)
12mds3_k2 <- cmdscale(D_k3, k = 2, eig = TRUE)
13mds3_k3 <- cmdscale(D_k3, k = 3, eig = TRUE)
14mds3_k4 <- cmdscale(D_k3, k = 4, eig = TRUE)
15
16save(list = ls(all = TRUE), file = "whichdim.RData")
17save(list = c("mds3_k1", "mds3_k2", "mds3_k3", "mds3_k4"), file = "whichdim-less.RData
18  ")
19load("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 #         eig     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 .ret    is true.
30
31 #   ac: the additive constant c*,      0      if    add = FALSE .
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.