CS598 - Coding Assignment 4

Xiaoming Ji

Step 1: Set the seed to be the last 4-dig of University ID.

```
set.seed(6682)
```

Step 2: Create

- **train data** that contains about 60% rows of the ratings.dat from the MovieLens 1M dataset (of the same format);
- **test data** that contains about 20% of the user-movie pairs from the ratings.dat from the MovieLens 1M dataset.

```
ratings = read.csv('ratings.dat', sep = ':',
    colClasses = c('integer', 'NULL'), header = FALSE)
colnames(ratings) = c('UserID', 'MovieID', 'Rating', 'Timestamp')

ratings$Timestamp = NULL;
train.id = sample(nrow(ratings), floor(nrow(ratings)) * 0.6)
train = ratings[train.id, ]

test = ratings[-train.id, ]
test.id = sample(nrow(test), floor(nrow(test)) * 0.5)
test = test[test.id, ]

label = test[c('UserID', 'Rating')]
test$Rating = NA
```

Step 3: Build two models to predict the movie rating for each user-movie pair in the test data.

Using `evaluate`, I run through cross-validation test among all available models and found `UBCF`, `POPULAR` and `SVDF` perform better (measured by RMSE) than other models. Thus, for this assignment, I will use two models as,

- UBCF: User-based collaborative filtering.
  - parameter: normalize = 'Z-score', method = 'cosine', nn = 5
- SVDF: Funk SVD with gradient descend. Funk SVD decomposes a matrix (with missing values) into two components U and V. The singular values are folded into these matrices. The approximation for the original matrix can be obtained by R = UV'. This model predicts new data rows by estimating the u vectors using gradient descend and then calculating the reconstructed complete matrix r for these users via r = uV'.
  - parameter: normalize = 'Z-score'

```
models = list(
  UBCF = list(normalize = 'Z-score', method = 'cosine', nn = 5),
  SVDF = list(normalize = 'Z-score')
)
```

```r
start.time = Sys.time()
R = acast(train, UserID ~ MovieID, value.var = 'Rating')
R = as(R, 'realRatingMatrix')

rmses = rep(0, length(models))
names(rmses) = names(models)
for (m in 1:length(models)) {
  rec = Recommender(R, method = names(models)[m],
      parameter = models[[m]])

  recom = predict(rec, R, type = 'ratings')
  rec_list = as(recom, 'list')

  # For all lines in test file, one by one
  for (u in 1:nrow(test)){
      userid = as.character(test$UserID[u])
      movieid = as.character(test$MovieID[u])

      rating = rec_list[[userid]][movieid]
      test$Rating[u] = ifelse(is.na(rating), 2.5, rating)
  }

  rmses[m] = RMSE(test$Rating, label$Rating)
}

end.time = Sys.time()
run.time = as.numeric(difftime(end.time, start.time, units = 'secs'))
```

Computation time: 3813 seconds

Step 4: Report the RMSE (Root-mean-square error) of these two models on the test data.

```
##      UBCF      SVDF
## 1.0352958 0.8768351
```