

# Project 1

*Max Wagner*

*June 10, 2016*

---

## Data

The data was generated in excel. It is similar to the data from class on the movie rating, but with 26 imaginary people, and 10 movies. It uses a similar 1-5 scale, with 0 representing a movie that they have not seen. In this case, the idea of the recommendation system would be to recommend the best movies to someone. The main method will be collaborative filtering.

The head of the csv file can be seen below:

```
movies <- read.csv('movies.csv', stringsAsFactors = FALSE)
users <- movies[,1]
movies <- movies[,2:11]
movie.names <- colnames(movies)
head(movies)
```

```
##   m1 m2 m3 m4 m5 m6 m7 m8 m9 m10
## 1  0  1  1  4  0  1  1  0  3   3
## 2  3  4  0  3  3  3  2  5  3   5
## 3  4  4  4  0  0  5  1  1  2   0
## 4  2  4  2  1  0  2  5  2  1   0
## 5  3  2  2  4  1  0  1  4  3   5
## 6  4  4  2  1  5  3  5  4  1   5
```

A mean centered preview for movies can be seen below. This section also gets the mean centered information for users if we need to compare users later.

```
users.avg <- rowMeans(movies, na.rm = TRUE)
users.avg.total <- mean(users.avg)
movies.avg <- colMeans(movies, na.rm = TRUE)
movies.avg.total <- mean(movies.avg)

users.mean.centered <- round(movies[] - users.avg,2)
movies.mean.centered <- round(movies[] - movies.avg,2)
head(movies.mean.centered)
```

```
##      m1      m2      m3      m4      m5      m6      m7      m8      m9      m10
## 1 -3.12 -2.15 -1.23  1.38 -2.35 -2.12 -2.15 -2.23  0.38  0.65
## 2  0.19  1.04 -1.85  0.42 -0.23  0.19 -0.96  3.15  0.42  1.77
## 3  1.77  1.38  1.65 -3.12 -3.15  2.77 -1.62 -1.35 -1.12 -3.15
## 4  0.15  1.42 -1.23 -1.81 -2.96  0.15  2.42 -1.23 -1.81 -2.96
## 5  0.65 -1.12 -1.15  1.77 -1.62 -2.35 -2.12  0.85  0.77  2.38
## 6  0.77  1.19 -0.96 -0.85  2.42 -0.23  2.19  1.04 -0.85  2.42
```

## Movie CF

Calculating the similarity matrix by hand using cosines.

```
# Calculating cosines fn
gcos <- function(a,b) {
  return(sum(a*b) / (sqrt(sum(a^2)) * sqrt(sum(b^2))))
}

#10x10 matrix for the 10 movies
movie.simi <- matrix(NA, nrow = length(movie.names), ncol = length(movie.names), dimnames = list(colnames(movie.names), rownames(movie.names)))

for(i in 1:ncol(movie.simi)) {
  for(j in 1:ncol(movie.simi)) {
    movie.simi[i,j] <- gcos(movie.mean.centered[i],movie.mean.centered[j])
  }
}

movie.simi <- data.frame(movie.simi)
colnames(movie.simi) <- movie.names
rownames(movie.simi) <- movie.names
```

The similarity matrix looks good, it has 1's on the diagonal with numbers ranging from -1 to 1. From here some recommendations can be found. Let's find the most recommended movies to watch for each movie in decreasing order.

```
# another placeholder, make sure to drop 1st as it will be the same movie with a 1.00
movie.recs <- data.frame(matrix(NA, nrow = 10, ncol = 10))

#loop to order the similarites in each row
for(i in 1:ncol(movie.simi)) {
  movie.recs[i,] <- colnames(movie.simi[i,order(movie.simi[i,],decreasing = TRUE)])
}

#get rid of first col
movie.recs <- movie.recs[,-1]
colnames(movie.recs) <- paste("rec",1:9)
rownames(movie.recs) <- paste("movie", 1:10)
movie.recs
```

##	rec 1	rec 2	rec 3	rec 4	rec 5	rec 6	rec 7	rec 8	rec 9
## movie 1	m7	m2	m5	m9	m6	m8	m4	m3	m10
## movie 2	m7	m6	m1	m8	m5	m3	m4	m9	m10
## movie 3	m6	m7	m5	m9	m8	m2	m4	m10	m1
## movie 4	m9	m5	m10	m3	m8	m2	m6	m1	m7
## movie 5	m10	m7	m8	m1	m3	m4	m2	m9	m6
## movie 6	m3	m7	m2	m1	m9	m8	m5	m4	m10
## movie 7	m1	m5	m2	m6	m3	m8	m10	m9	m4
## movie 8	m10	m5	m7	m2	m3	m6	m9	m4	m1
## movie 9	m4	m1	m10	m3	m6	m5	m8	m2	m7
## movie 10	m5	m8	m9	m4	m3	m7	m6	m2	m1

Above we can see all 10 movies, and then 9 recommendations in the order of similarity.

## R Packages

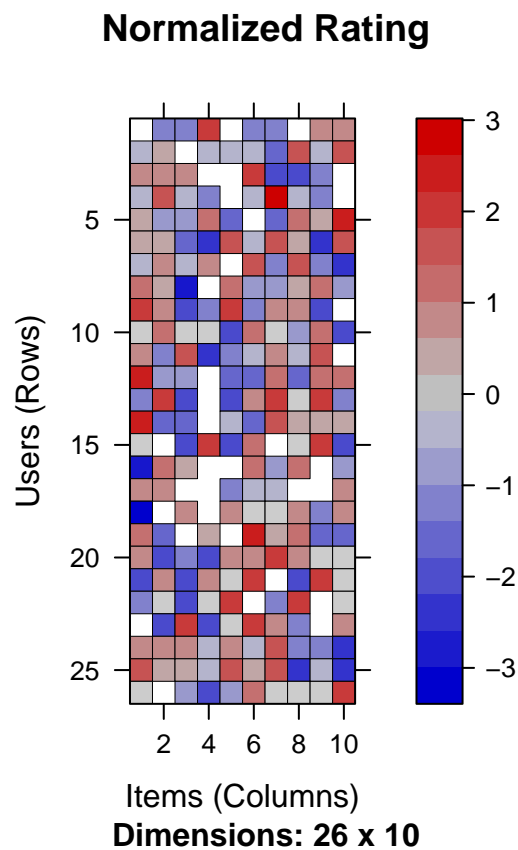
I'll use 'recommenderlab' to get a few different types of read outs.

```
library(recommenderlab)
movies <- read.csv('movies.csv', stringsAsFactors = FALSE)
movies[movies==0] <- NA
movies <- as.matrix(movies[,2:11])

#make it a rating matrix
movies.rm <- as(movies, "realRatingMatrix")

#normalize it, should do this automatically in Recommender as well
movies.rmn <- normalize(movies.rm)

#graph it
plot(image(movies.rmn, main = "Normalized Rating"))
```



```
#create a recommender for what the users should see next
movies.recom <- Recommender(movies.rm[1:20], method = "UBCF")
movies.predict <- predict(movies.recom, movies.rm[21:26], n=9)
as(movies.predict, "list")
```

```
## [[1]]
## [1] "m7"
```

```
##
## [[2]]
## [1] "m6" "m9"
##
## [[3]]
## [1] "m9" "m1"
##
## [[4]]
## character(0)
##
## [[5]]
## character(0)
##
## [[6]]
## [1] "m2"
```

The above list turned out slightly differently than I had predicted at first, but it made sense when I realized it was recommending users to see the movies they had no seen yet. In the case of users that had seen all the movies, it had no recommendations for them.

We can then try to predict ratings for users.

```
movies.predict2 <- predict(movies.recom, movies.rm[21:26], type = "ratingMatrix")
as(movies.predict2, "list")
```

```
## [[1]]
##      m1      m2      m3      m4      m5      m6      m7
## -2.000000  1.000000 -2.000000  1.000000  0.000000  2.000000 -4.936668
##      m8      m9      m10
## -2.000000  2.000000  0.000000
##
## [[2]]
##      m1      m2      m3      m4      m5      m6      m7
## -1.000000  0.000000 -2.000000  0.000000  2.000000  3.201132 -1.000000
##      m8      m9      m10
##  2.000000 -1.185667  0.000000
##
## [[3]]
##      m1      m2      m3      m4      m5      m6      m7
##  2.539408 -2.125000  1.875000 -2.125000 -0.125000  1.875000  0.875000
##      m8      m9      m10
## -1.125000  4.321640  0.875000
##
## [[4]]
##      m1  m2  m3  m4  m5  m6  m7  m8  m9  m10
##  0.8  0.8  0.8 -0.2  0.8 -0.2  1.8 -1.2 -1.2 -2.2
##
## [[5]]
##      m1  m2  m3  m4  m5  m6  m7  m8  m9  m10
##  1.5  0.5  0.5 -0.5  1.5  0.5  1.5 -2.5 -0.5 -2.5
##
## [[6]]
##      m1      m2      m3      m4      m5      m6
##  0.1111111  2.7359316 -0.8888889 -1.8888889 -0.8888889  1.1111111
```

```
##           m7           m8           m9           m10
## 0.1111111 0.1111111 0.1111111 2.1111111
```

The outcome of testing the “by hand” method versus using a package leads me to believe that using a pre-built package will be best for anything that you are not able to easily check by hand. The example I used was fairly small and I can eye results to see if they made sense. When the data set becomes significantly larger, it will be nearly impossible to check results to see if your by hand method works correctly. The inclusion of NA values also makes the pre-build package more appealing.