Singular Value Decomposition

DATA612 - Recommender Systems

William Outcault

23 June 2020

```
library(mice)
library(recommenderlab)
library(Matrix)
```

In this project I will be creating a 10x10 toy matrix which includes NA values. I will be imputing those values and performing SVD. Lastly I will be calculating the perfomance of IBCF, UBCF and SVD to compare the models.

We begin by creating a toy matrix which includes 10 users, 10 movies and ratings ranging from 0-5 with missing values.

Creating Toy Matrix

```
set.seed(10)

dimnames <- list(c("user1", "user2", "user3", "user4", "user5", "user6",
                   "user7", "user8", "user9", "user10"),
                 c("Movie1", "Movie2", "Movie3", "Movie4", "Movie5", "Movie6",
                   "Movie7", "Movie8", "Movie9", "Movie10"))

data <- sample(c(0:5,NA), size = 1000, replace=T)

mat <- matrix(data, nrow=10, ncol=10)

summary(as.vector(as.matrix(mat)))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   1.000   2.000   2.284   4.000   5.000      19
```

From our summary function we see we have 19 missing values and a uniform distribution.

Next we will scale and center our matrix, and use the predictive mean matching method from the `mice` package to replace NA's.

```
mat_scaled <- scale(mat)
imp <- mice(mat_scaled, meth=c('pmm'))
```

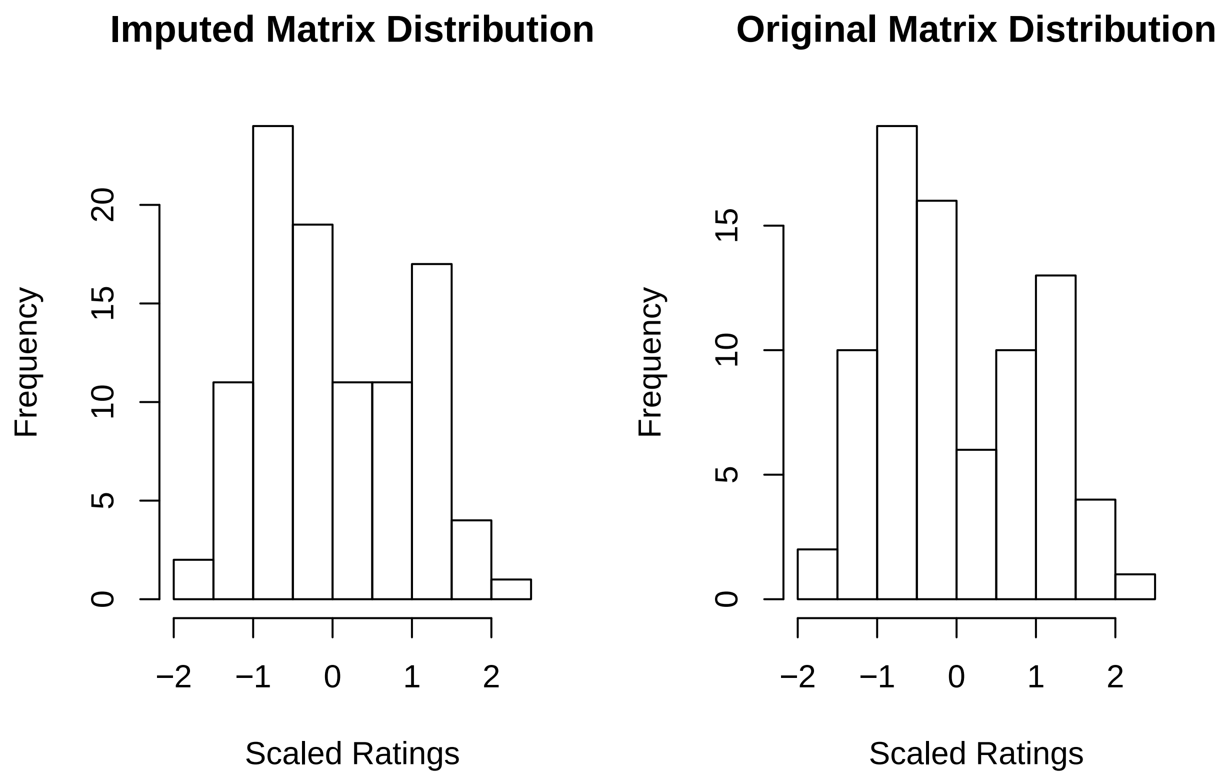
```
## Warning: Number of logged events: 283
```

```r
imp_mat <- complete(imp)
```

```r
summary(as.vector(as.matrix(imp_mat)))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1.54919 -0.77638 -0.09785  0.00839  0.77460  2.04900
```

```r
par(mfrow=c(1,2))
hist(as.vector(as.matrix(imp_mat)), main = "Imputed Matrix Distribution", xlab = "Scaled Ratings")
hist(as.vector(as.matrix(mat_scaled)), main = "Original Matrix Distribution", xlab = "Scaled Ratings")
```

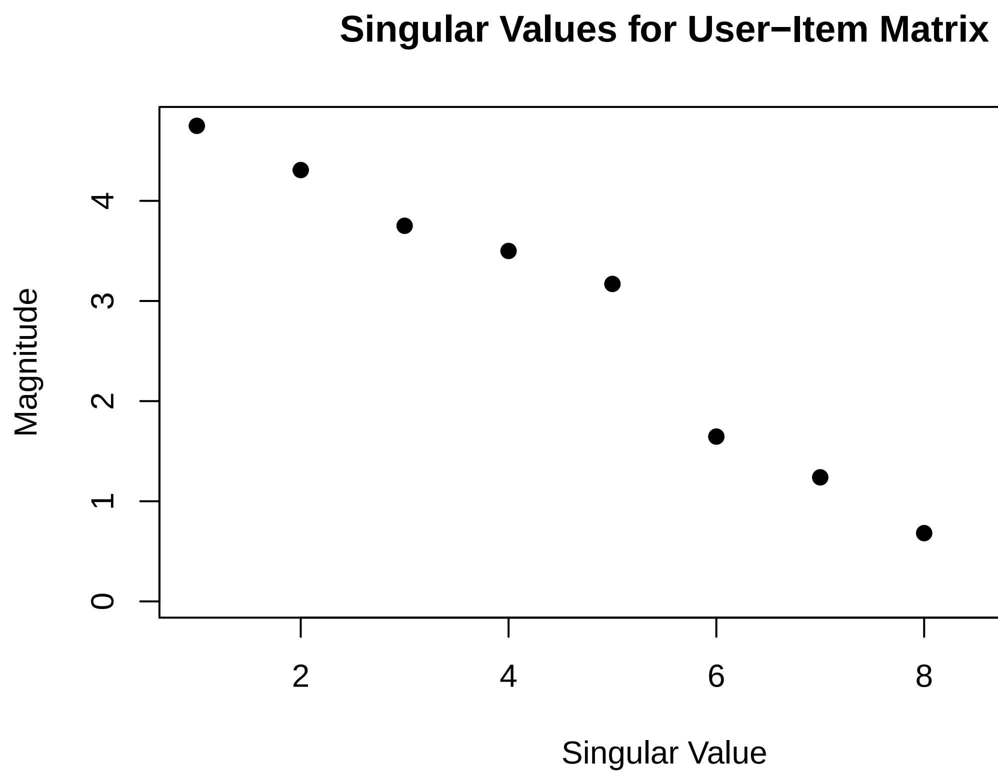


Our mean is centered around zero now and imputations did not change our distributions drastically because we used the predictive mean matching method.

Now that we have imputed missing values we can run out `svd` function.

```r
s <- svd(imp_mat)

plot(s$d, pch=20, cex = 1.5, xlab='Singular Value', ylab='Magnitude',
     main = "Singular Values for User-Item Matrix")
```



We plot the singular values in decreasing order to visualize each value's magnitude. Out goal is to find the first `k` singular values whose square sum compensates for `90%` of the total of the square sum of all singular values. The `k` value will define our factors to be used for predictions.

```
all_sing_sq <- sum(s$d^2)
percent <- 0
k <- 0
while (percent < 0.90){
    k <- k + 1
    percent <- sum(s$d[1:k]^2)/all_sing_sq
}
percent <- round(percent, 2)
```

With k = 5 we will retain 0.94% of our variability within the imputed values matrix. We can now create a 10x10 matrix of predictions using SVD, with only 5 factors.

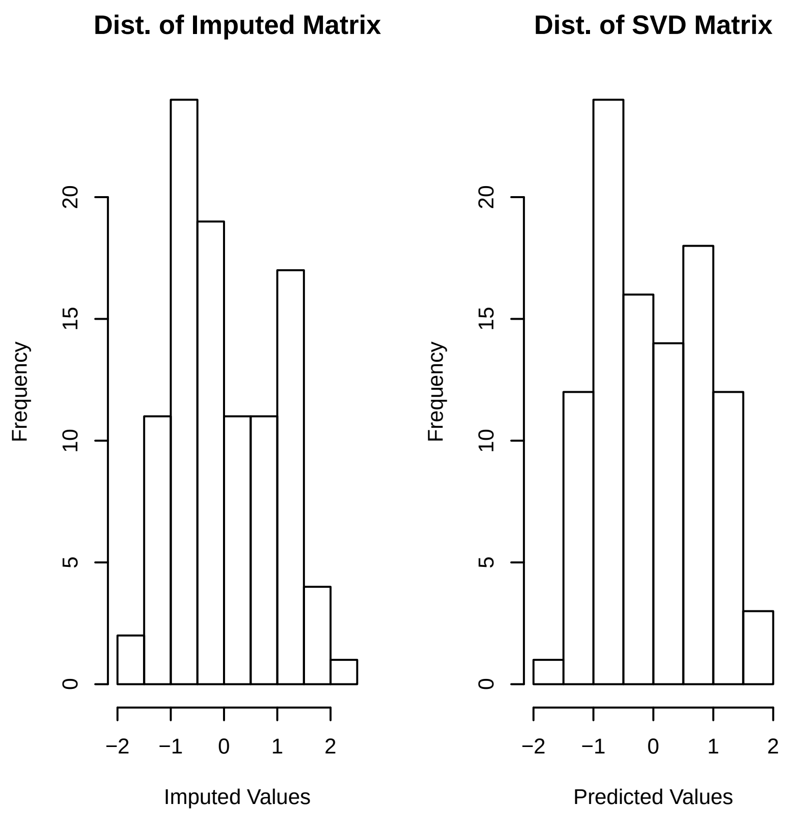
```
s_k <- Diagonal(x = s$d[1:k])
U_k <- s$u[, 1:k]
V_k <- t(s$v)[1:k, ]

predicted <- as.matrix(U_k %*% s_k %*% V_k)
```

```
summary(as.vector(as.matrix(predicted)))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1.70044 -0.77000 -0.07166 -0.01688  0.74562  1.83393
```

```r
par(mfrow=c(1,3))
hist(as.vector(as.matrix(imp_mat)), main = "Dist. of Imputed Matrix", xlab = "Imputed Values")
hist(as.vector(predicted), main = "Dist. of SVD Matrix", xlab = "Predicted Values")
```



Our predicted matrix using SVD shares a similar distribution to that of the imputed matrix. Now we will quanitatively analyse the differences between SVD, CBRS and UBRS.

```r
predicted <- as(predicted,'realRatingMatrix')
imp_mat_RRM <- as(as.matrix(imp_mat),'realRatingMatrix')

e1 <- evaluationScheme(imp_mat_RRM, method="split", train=0.8, given=-1, goodRating=0)

UBCF_Z_E <- Recommender(getData(e1, "train"), "UBCF",
      param=list(normalize = "Z-score",method="Euclidean"))

IBCF_N_C <- Recommender(getData(e1, "train"), "IBCF",
      param=list(normalize = NULL, method="Cosine"))


p1 <- predict(UBCF_Z_E, getData(e1, "known"), type="ratings")

p2 <- predict(IBCF_N_C, getData(e1, "known"), type="ratings")

errors <- rbind(
  UBCF = calcPredictionAccuracy(p1, getData(e1, "unknown")),
  IBCF = calcPredictionAccuracy(p2, getData(e1, "unknown")),
  SVD = calcPredictionAccuracy(x = predicted, data = imp_mat_RRM)
)

knitr::kable(errors)
```

	RMSE	MSE	MAE
UBCF	1.4225038	2.0235172	1.2422533
IBCF	1.4653098	2.1471329	1.3888358
SVD	0.2185486	0.0477635	0.1778368

Our SVD performed drastically better than UBCF and IBCF. However this was only a 10x10 matrix and I did not use a cross-validation technique therefore these performances may change given different distributions, matrix sizes, or cross-validation sets. Regardless the purpose of this project was to handle missing values without significantly changing the matrix distribution, so that we can continue with SVD.

References

https://rpubs.com/jt_rpubs/287285 - Matrix Factorization via Singular Value Decomposition https://datascienceplus.com/imputing-missing-data-with-r-mice-package/ - Imputing Missing Data with R, Mice