Project 1 Baseline Predictors and RMSE

Cheryl Bowersox

June 12, 2018

This project creates a very simple recommendation system using averages to determine if a joke will be rated highly by a given user. It was created based on the joke rating data from Jester online joke recommendation system. For the purposes of this project a subset, consisting of 100 users and 100 jokes, was selected.

The given information is organized with each row corresponding to a user, the number of items rated for each user in the first column, and each additional column corresponding to a joke. Ratings were provides between -10 and 10, with a value of '99' entered if the joke was not rate by that user.

This data was broken into a training set of 80% of the jokes, and a test set of 20%, and cleaned to remove the null values.

Source data: http://www.ieor.berkeley.edu/%7Egoldberg/jester-data/ (http://www.ieor.berkeley.edu/%7Egoldberg/jester-data/)

Github link: https://github.com/cherylb/IS643 (https://github.com/cherylb/IS643)

```
# data set

# take mean of entire set
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(reshape)
```

```
##
## Attaching package: 'reshape'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, smiths
```

```
## The following object is masked from 'package:dplyr':
##
##     rename
```

```
library(RCurl)
```

```
## Loading required package: bitops
```

```
##
## Attaching package: 'RCurl'
```

```
## The following object is masked from 'package:tidyr':
##
##     complete
```

```
#read data
#options(RCurlOptions = list(cainfo = system.file("CurlSSL", "cacert.pem", package =
"RCurl")))
#fileurl= "https://raw.githubusercontent.com/cherylb/IS643/master/jester-data-1.csv"
#data <- getURL(fileurl)
#dfdm <- read.csv(text = data, , stringsAsFactors=FALSE)

#shucks that a big file
# -or - local
#
joke <- read.csv("~/GitHub/IS643/jester-data-1.csv", header = FALSE)

#sort by # of reviews completed, select to 100 users
joke <- head(joke[order(-joke[,1]),],100)

#replacing column 1 with a user id
joke[,1] <- seq.int(nrow(joke))
names(joke) <- c("user", seq(1:100))


#take a look at the data
head(joke[,1:5])
```

```
##    user     1     2     3     4
## 2     1  4.08 -0.29  6.36  4.37
## 6     2 -6.17 -3.54  0.44 -8.50
## 8     3  6.84  3.16  9.17 -6.21
## 9     4 -3.79 -3.54 -9.42 -6.89
## 12    5  1.31  1.80  2.57 -2.38
## 14    6  9.22  9.27  9.22  8.30
```

```
#melt the joke data

mlt.joke <- melt(joke, id=c("user"))
names(mlt.joke) <- c("user", "joke", "value")
# keep complete cases
mlt.joke <- mlt.joke%>%filter(value != 99)

#replace '99' values with null values
#joke[joke == 99] <- NA



#split into training and test data

n <- .8*nrow(mlt.joke)
set.seed(42)
s <- sample(seq_len(nrow(mlt.joke)),n) #index #
#n <- sample(seq(1:nrow(mlt.joke)), .2*nrow(mlt.joke))

trn.joke <- mlt.joke[s,]
tst.joke <- mlt.joke[-s,]
```

```
#find mean of the training data

trn.mean <- mean(trn.joke$value)
```

The mean value across the training data was 0.9696925. This is very close to a zero value, which makes sense if the moves are rated between -10 and 10. The positive value indicates jokes are rated positively more often than negatively, but the small magnitude of the mean shows they are fairly well balanced in aggregate between positive and negative ratings.

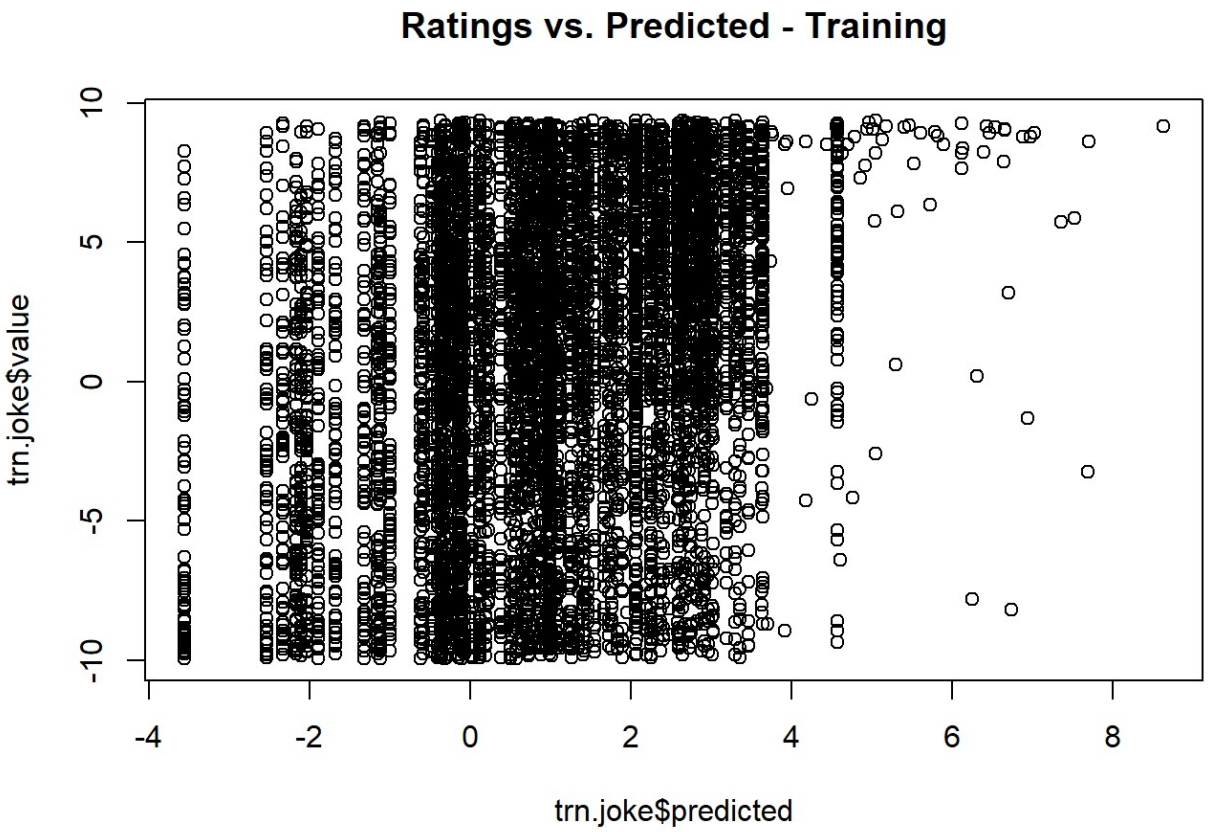
```
#Calculate error

pred.joke <- rep(trn.mean, nrow(trn.joke))
error <- pred.joke - trn.joke$value
trn.rmse <- sqrt(mean(error^2))

tst.err <- rep(trn.mean, nrow(tst.joke))-tst.joke$value
tst.rmse <- sqrt(mean(tst.err^2))
```

The RMSE comparing the mean to the actual values on the training data is 5.5134074, while the RMSE comparing this training mean with the actual values of the test data is 5.3760752. These are very similar, which could indicate the test data is well represented by the training data, however the it is a fairly large error give the actual values range between -10 and 10.

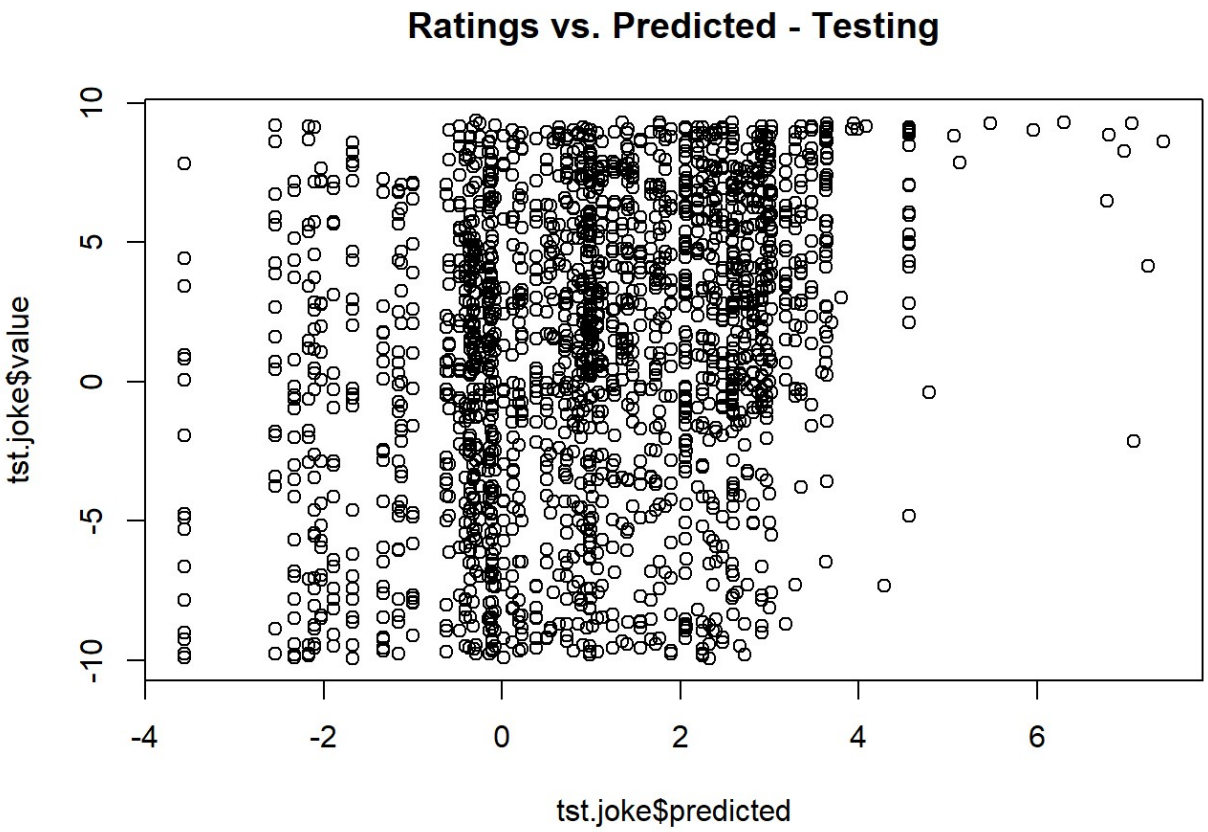
A better estimate might be given by calculating the individual user bias or joke bias. This was done using the mean values for each user an item, adjusting for the overall mean, and then using these bias values to create an estimate for each item for each user.

Because some users or items in the test data may not be in the training data, and therefor may not have a related bias these values were estimated using the overall mean previously calculated.

```
# calculate bias for each user/item interaction
# y = trn.mean + b1 + b2

#user bias - simple approach takes the difference between mean rating for user - trn.m
ea
#mean for each user
trn.user.bias <- data.frame(trn.joke %>%group_by(user)%>%
    summarise(usermean = mean(value))%>%mutate(userbias = usermean - trn.mean))


#mean for joke
trn.joke.bias <- data.frame(trn.joke %>%group_by(joke)%>%
    summarise(jokemean = mean(value))%>%mutate(jokebias = jokemean - trn.mean))

#calc predictions for training data
calcpred <- function(d){

  ubias<- trn.user.bias[trn.user.bias$user == d[[1]], 3]
  if (length(ubias)== 0) ubias = 0

  jbias <- try(trn.joke.bias[trn.joke.bias$joke == d[[2]], 3])
  if (length(jbias)== 0) jbias = 0

  #if they exist in training will return value, otherwise mean
  pred <- trn.mean + ubias + jbias
  return(pred)
}

z <- apply(trn.joke, 1, calcpred)

trn.joke$predicted  <-  z
#plot
plot(trn.joke$predicted, trn.joke$value, main = "Ratings vs. Predicted - Training")
```




```
z <- apply(tst.joke, 1, calcpred)

tst.joke$predicted  <-  z
#plot
plot(tst.joke$predicted, tst.joke$value, main = "Ratings vs. Predicted - Testing")
```



A plot of the predicted values vs. the actual ratings shows the training data is somewhat related to the prediction, but when tested on the actual data it is less useful.

```
error2 <- trn.joke$predicted - trn.joke$value
trn.rmse2 <- sqrt(mean(error2^2))

error3 <- tst.joke$predicted - tst.joke$value
tst.rmse2 <- sqrt(mean(error3^2))
```

The RMSE for the training data using this new model is 5.2590161

The RMSE for the test data using this new model is 5.1339

These are very similar to the error found just using the mean value for all items, although the accuracy of both is slightly better, it is still very high, and not much more accurate than the simpler model.

The overall accuracy may be slightly better using the simple bias calculation, but neither method provides a good estimate. The graph of the prediction on test data shows very little relationship between our predictions and outcome, this system may be helpful for general trending information but is not terribly helpful for making a useful recommendation.