Hotel Review Project

Alexander Cruz

1/9/2020

Overview

This final project is related to HarvardX Data Science: Capstone course. I decided to build a machine learning algorithm due to my love to travel. github repo: https://github.com/tclex7/havard_edx_final_project

Introduction

As I mentioned above I am really interested in the hotel industry, and I wanted to see if using the skills I learned in the previous machine learning course I could build an effective recommendation system. I used the Hotel Reviews dataset from Kaggle.com, and was supprised to see that the average rating of over 4, using 1 to 5 scale.

Dataset

As mentioned above, for this project I used the Hotel Reviews datasets from kaggle.com, that was provided by Datafiniti's Business Database. I used two datasets that ranged from dates January 2018 to September 2018 and December 2018 to May 2019. These two datasets were combined and uploaded to github(https://github.com/tclex7/havard_edx_final_project) as an .rds file, github has a policy that does not allow files over 25mb, so .csv was not possible. The dataset included 19,758 reviews, 2,753 unique hotels, 15,558 users, and all 50 states.

Methodology

After Cleaning the data, the first step was to set a baseline for recommendation system. The average rating will be used as that baseline. We will buildrecommendation models using the variables hotel, user, and state, which are also included in the dataset. We will look that the variables effects, and then add regularized linear regresssion to each variable.

Read and Clean data set

Verify that all R packages needed for this project are installed, and activate libraries

```r
if(!require(readr)) install.packages("readr", repos = "http://cran.us.r-project.org")
library(readr)
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
library(tidyverse)
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
library(caret)
```

```r
if(!require(knitr)) install.packages("knitr", repos = "http://cran.us.r-project.org")
library(knitr)
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
library(ggplot2)
if (!require(scales)) install.packages('scales')
library(scales)
```

Read and Clean Data

Read in hotel rds file, it has been saved in github ripository: https://github.com/tclex7/havard_edx_final_project/blob/master/hotel.rds

```r
hotel_reviews <- read_rds("hotel.rds")
```

Remove duplicate rows in the dataframe

```r
hotel_reviews <- distinct(hotel_reviews)
```

Summary of Data

```r
glimpse(hotel_reviews)
```

```
## Observations: 19,758
## Variables: 25
## $ id                  <chr> "AVwc252WIN2L1WUfpqLP", "AVwc252WIN2L1WUfpqLP", ...
## $ dateAdded           <dttm> 2016-10-30 21:42:42, 2016-10-30 21:42:42, 2016-...
## $ dateUpdated         <dttm> 2018-09-10 21:06:27, 2018-09-10 21:06:27, 2018-...
## $ address             <chr> "5921 Valencia Cir", "5921 Valencia Cir", "5921 ...
## $ categories          <chr> "Hotels,Hotels and motels,Hotel and motel reserv...
## $ primaryCategories   <chr> "Accommodation & Food Services", "Accommodation ...
## $ city                <chr> "Rancho Santa Fe", "Rancho Santa Fe", "Rancho Sa...
## $ country             <chr> "US", "US", "US", "US", "US", "US", "US", "US", ...
## $ keys                <chr> "us/ca/ranchosantafe/5921valenciacir/359754519",...
## $ latitude            <dbl> 32.99096, 32.99096, 32.99096, 39.15593, 39.15593...
## $ longitude           <dbl> -117.18614, -117.18614, -117.18614, -76.71634, -...
## $ name                <chr> "Rancho Valencia Resort Spa", "Rancho Valencia R...
## $ postalCode          <dbl> 92067, 92067, 92067, 21076, 21076, 21076, 21076,...
## $ province            <chr> "CA", "CA", "CA", "MD", "MD", "MD", "MD", "MD", ...
## $ reviews.date        <dttm> 2013-11-14, 2014-07-06, 2015-01-02, 2016-05-15,...
## $ reviews.dateSeen    <chr> "2016-08-03T00:00:00Z,2016-07-26T00:00:00Z,2016-...
## $ reviews.rating      <dbl> 5, 5, 5, 2, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, ...
## $ reviews.sourceURLs  <chr> "https://www.hotels.com/hotel/125419/reviews%20/...
## $ reviews.text        <chr> "Our experience at Rancho Valencia was absolutel...
## $ reviews.title       <chr> "Best romantic vacation ever!!!!", "Sweet sweet ...
## $ reviews.userCity    <chr> NA, NA, NA, "Richmond", "Laurel", "Laurel", NA, ...
## $ reviews.userProvince <chr> NA, NA, NA, "VA", "MD", "MD", NA, "NC", "MA", NA...
## $ reviews.username    <chr> "Paula", "D", "Ron", "jaeem2016", "MamaNiaOne", ...
## $ sourceURLs          <chr> "http://www.hotels.com/ho125419/%25252525253Floc...
## $ websites            <chr> "http://www.ranchovalencia.com", "http://www.ran...
```

```r
summary(hotel_reviews)
```

```
##       id              dateAdded                      dateUpdated
##  Length:19758       Min.   :2014-10-24 13:52:43   Min.   :2018-01-01 00:00:46
##  Class :character   1st Qu.:2016-05-21 19:54:11   1st Qu.:2018-03-23 17:08:30
##  Mode  :character   Median :2017-03-04 12:22:32   Median :2018-09-04 21:27:52
##                     Mean   :2017-02-18 07:58:00   Mean   :2018-10-06 11:17:20
##                     3rd Qu.:2017-12-21 00:01:02   3rd Qu.:2019-04-01 15:06:15
##                     Max.   :2018-12-28 06:33:31   Max.   :2019-05-20 23:55:47
##
##    address           categories        primaryCategories      city
##  Length:19758       Length:19758       Length:19758       Length:19758
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##    country              keys              latitude       longitude
##  Length:19758       Length:19758       Min.   :19.44   Min.   :-159.48
##  Class :character   Class :character   1st Qu.:32.76   1st Qu.:-117.17
##  Mode  :character   Mode  :character   Median :36.17   Median : -90.06
##                                        Mean   :36.00   Mean   : -97.12
##                                        3rd Qu.:39.89   3rd Qu.: -81.27
##                                        Max.   :70.13   Max.   : -68.20
##
##      name             postalCode      province
##  Length:19758       Min.   : 1033   Length:19758
##  Class :character   1st Qu.:30303   Class :character
##  Mode  :character   Median :60601   Mode  :character
##                     Mean   :57528
##                     3rd Qu.:92037
##                     Max.   :99801
##                     NA's   :2768
##   reviews.date                 reviews.dateSeen   reviews.rating
##  Min.   :2002-07-24 00:00:00   Length:19758       Min.   :1.000
##  1st Qu.:2015-03-02 00:00:00   Class :character   1st Qu.:3.950
##  Median :2016-02-05 00:00:00   Mode  :character   Median :4.000
##  Mean   :2015-08-23 22:28:43                      Mean   :4.058
##  3rd Qu.:2016-08-13 00:00:00                      3rd Qu.:5.000
##  Max.   :2019-01-30 00:00:00                      Max.   :5.000
##
##  reviews.sourceURLs reviews.text       reviews.title      reviews.userCity
##  Length:19758       Length:19758       Length:19758       Length:19758
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  reviews.userProvince reviews.username    sourceURLs          websites
##  Length:19758         Length:19758       Length:19758       Length:19758
##  Class :character     Class :character   Class :character   Class :character
```

```
##  Mode  :character     Mode  :character   Mode  :character   Mode  :character
##
##
##
##
```

```r
head(hotel_reviews)
```

```
## # A tibble: 6 x 25
##   id    dateAdded           dateUpdated         address categories
##   <chr> <dttm>              <dttm>              <chr>   <chr>
## 1 AVwc... 2016-10-30 21:42:42 2018-09-10 21:06:27 5921 V... Hotels,Ho...
## 2 AVwc... 2016-10-30 21:42:42 2018-09-10 21:06:27 5921 V... Hotels,Ho...
## 3 AVwc... 2016-10-30 21:42:42 2018-09-10 21:06:27 5921 V... Hotels,Ho...
## 4 AVwd... 2015-11-28 19:19:35 2018-09-10 21:06:16 7520 T... Hotels,Ho...
## 5 AVwd... 2015-11-28 19:19:35 2018-09-10 21:06:16 7520 T... Hotels,Ho...
## 6 AVwd... 2015-11-28 19:19:35 2018-09-10 21:06:16 7520 T... Hotels,Ho...
## # ... with 20 more variables: primaryCategories <chr>, city <chr>, country <chr>,
## #   keys <chr>, latitude <dbl>, longitude <dbl>, name <chr>, postalCode <dbl>,
## #   province <chr>, reviews.date <dttm>, reviews.dateSeen <chr>,
## #   reviews.rating <dbl>, reviews.sourceURLs <chr>, reviews.text <chr>,
## #   reviews.title <chr>, reviews.userCity <chr>, reviews.userProvince <chr>,
## #   reviews.username <chr>, sourceURLs <chr>, websites <chr>
```

Select only the variables that we will be using for machine learning algorithm

```r
hotel_reviews <- hotel_reviews %>% select(name, reviews.rating,reviews.username,province)
```

Rename variables

```r
colnames(hotel_reviews) <- c("hotel","rating","user","state")
```

check to see if any of the variables have N/As

```r
sapply(hotel_reviews,function(x)sum(is.na(x)))
```

```
##  hotel rating   user  state
##      0      0      1      0
```

Since there is only one N/A for user, we will call that user "Mr. Unknown"

```r
hotel_reviews[is.na(hotel_reviews)] <- "Mr. Unknown"
```

now we can verify no N/As exist in the dataset

```r
sapply(hotel_reviews,function(x)sum(is.na(x)))
```

```
##  hotel rating   user  state
##      0      0      0      0
```

Summary of distinct reviews, hotels, users, and states

```r
hotel_reviews %>% summarize(total_reviews = n(),
                            total_hotels = n_distinct(hotel),
                            total_users = n_distinct(user),
                            total_states = n_distinct(state))
```

```
## # A tibble: 1 x 4
##   total_reviews total_hotels total_users total_states
##           <int>        <int>       <int>        <int>
## 1         19758         2753       15558           50
```

check how many different ratings were given

```r
n_distinct(hotel_reviews$rating)
```

```
## [1] 30
```

We see 1 is the smallest rating and 5 was the highest

```r
summary(hotel_reviews$rating)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   3.950   4.000   4.058   5.000   5.000
```

Table breakdown of all possible ratings

```r
data.frame(table(hotel_reviews$rating))
```

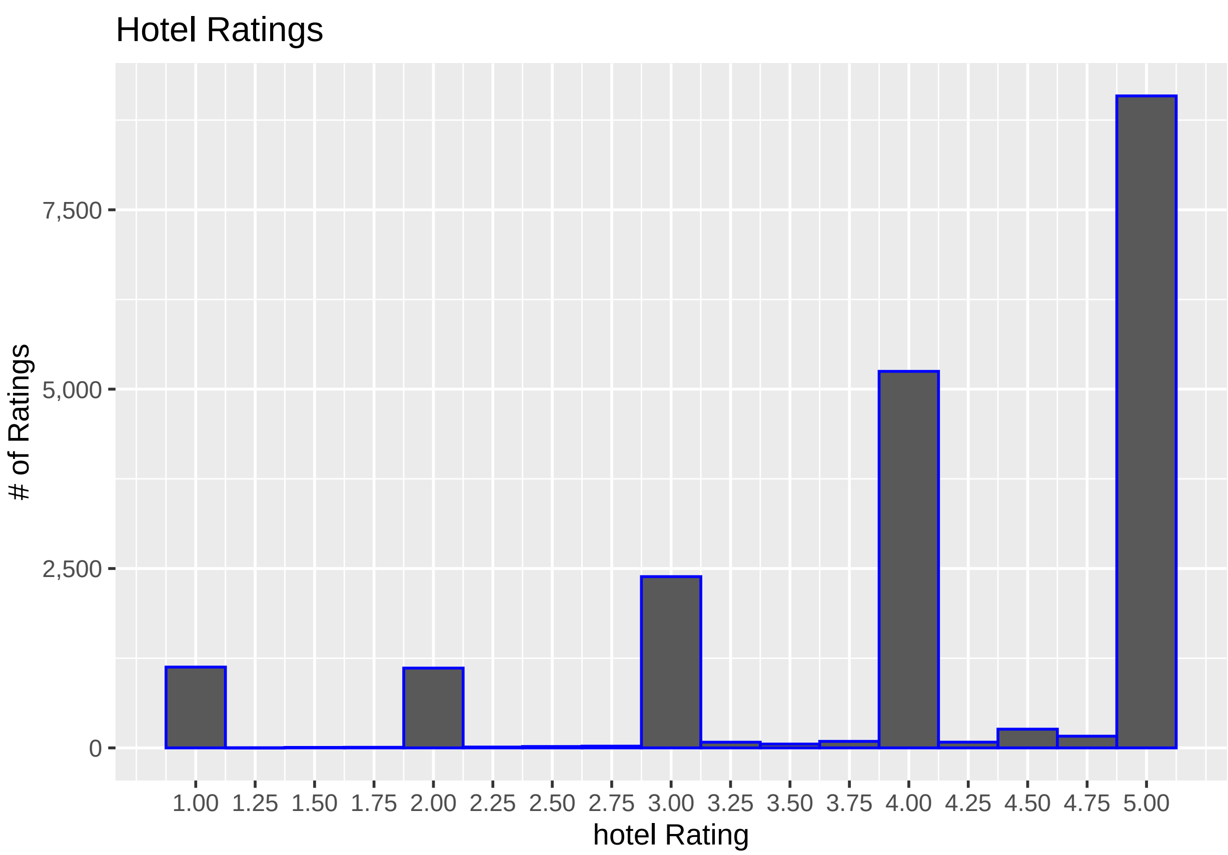
```
##    Var1 Freq
## 1     1 1126
## 2  1.25    2
## 3  1.45    6
## 4  1.65    8
## 5   1.9    4
## 6     2 1097
## 7   2.1   11
## 8   2.3   12
## 9   2.5   19
## 10  2.7   23
## 11 2.75    1
## 12  2.9   34
## 13    3 2353
## 14 3.15   37
## 15 3.25    2
## 16 3.35   40
## 17 3.45    1
## 18  3.5    2
## 19 3.55   50
## 20 3.75   91
## 21 3.95   52
## 22    4 5196
```

```
## 23 4.15   78
## 24 4.25    2
## 25  4.4  102
## 26  4.5    4
## 27  4.6  155
## 28 4.75    1
## 29  4.8  162
## 30    5 9087
```

Explore and visualize data set

Histrogram showing distribution of rating, we can see that majority of ratings were either 5s or 4s

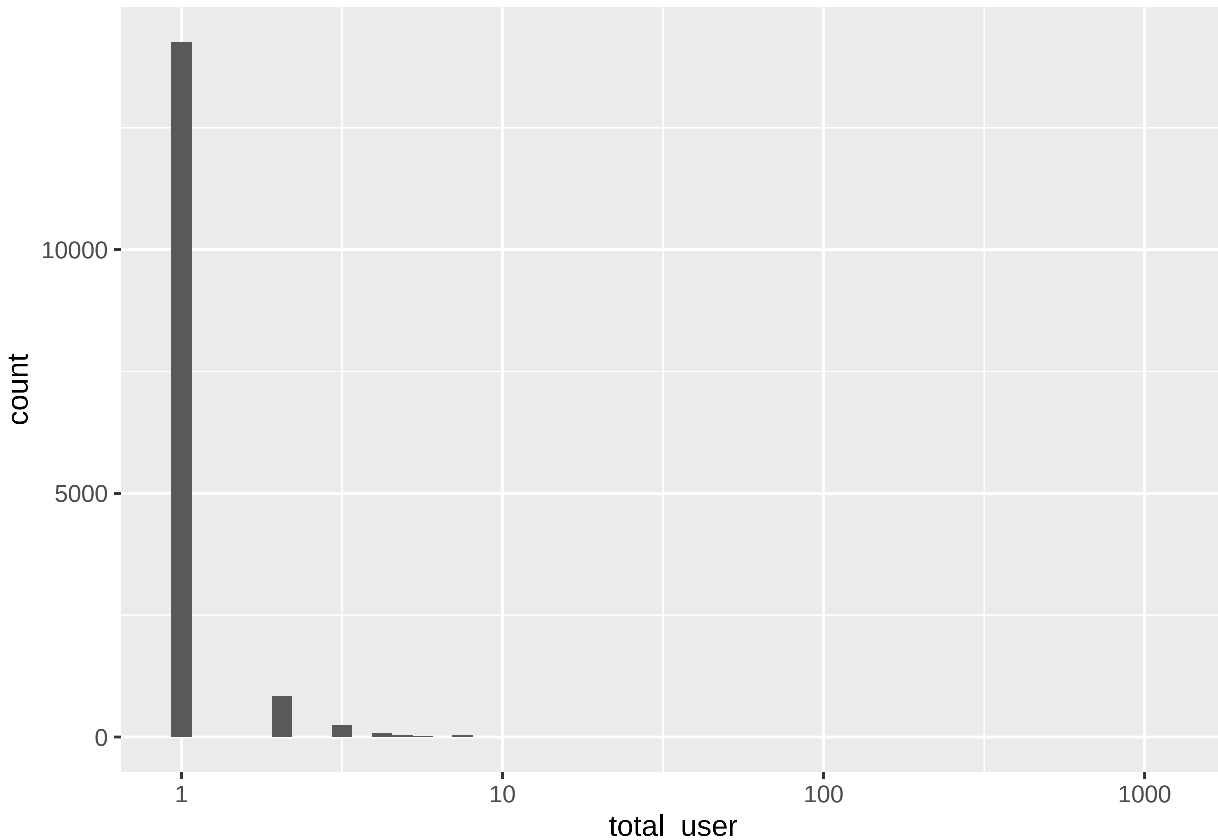
```
hotel_reviews %>%
  ggplot(aes(rating)) +
  geom_histogram(binwidth = .25, color = "blue") +
  scale_x_continuous(breaks=seq(1, 5, .25)) +
  scale_y_continuous(labels=comma) +
  labs(x="hotel Rating", y="# of Ratings") +
  ggtitle("Hotel Ratings")
```



Number of ratings per user, we can see that the majority of users only submitted one review

```
hotel_reviews %>%
  group_by(user) %>%
  summarize(total_user = as.numeric(n())) %>%
```

```
  ggplot(aes(total_user)) +
  geom_histogram(bins = 50) +
  scale_x_log10()
```



We will find out exactly what portion of the users only completed 1 review in the dataset

```
hotel <- hotel_reviews %>%
  group_by(user) %>%
  summarize(total_user = as.numeric(n())) %>%
  arrange(desc(total_user)) %>%
  mutate(one_or_more = ifelse(total_user==1,"just_one","more_than_one"))
table(hotel$one_or_more)
```

```
##
##      just_one more_than_one
##         14259          1299
```

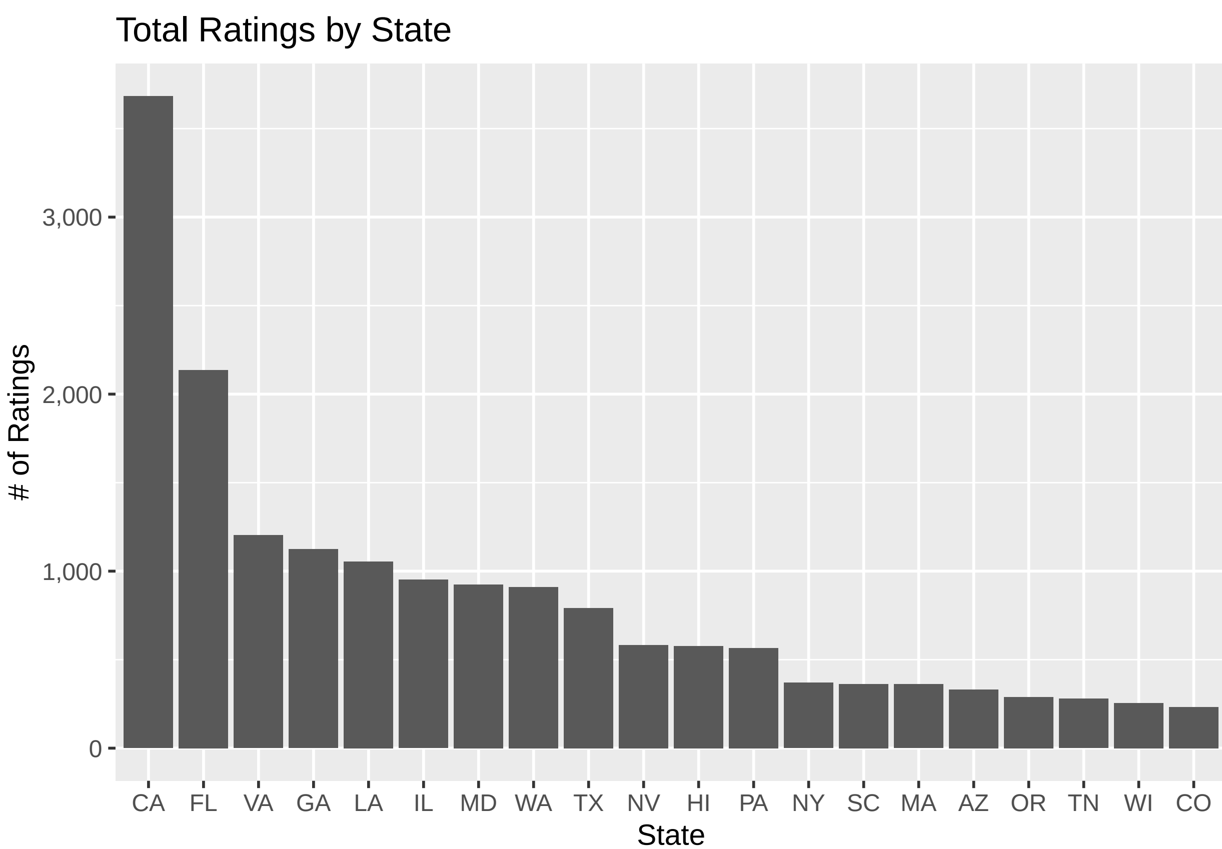
Looks like about 92% of users only completed 1 review

```
round(prop.table(table(hotel$one_or_more))*100,0)
```

```
##
##      just_one more_than_one
##            92             8
```

Visualize top 20 total ratings by state, we can see California and Florida get the most reviews

```
hotel_reviews %>%
  group_by(state) %>%
  summarize(total_state = as.numeric(n())) %>%
  arrange(desc(total_state)) %>%
  slice(1:20) %>%
  ggplot(aes(x = reorder(state, -total_state),total_state), colour ="blue") +
  geom_col() +
  scale_y_continuous(labels=comma) +
  labs(x="State", y="# of Ratings") +
  ggtitle("Total Ratings by State")
```



Breakout Data so we have a training and a test set.

set seed, if you have R version 3.5 or below use set.seed(1), below you can see what your current version is

```
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
version$version.string
```

```
## [1] "R version 3.6.2 (2019-12-12)"
```

Break out data to train and test sets

```
test_index <- createDataPartition(y = hotel_reviews$rating, times = 1, p = 0.4, list = FALSE)
hotel_train <- hotel_reviews[-test_index,]
temp <- hotel_reviews[test_index,]
```

Make sure user and hotel in hotel_test set are also in hotel_train set

```
hotel_test <- temp %>%
  semi_join(hotel_train, by = "hotel") %>%
  semi_join(hotel_train, by = "user")
```

Add rows removed from hotel_test set back into hotel_train set

```
removed <- anti_join(temp, hotel_test)
```

```
## Joining, by = c("hotel", "rating", "user", "state")
```

```
hotel_train <- rbind(hotel_train, removed)
```

Remove variables no longer needed

```
rm(test_index, temp, removed)
```

Rename variables to make it easier to run functions

```
train_set <- hotel_train
test_set <- hotel_test
rm(hotel_train, hotel_test)
```

Build recommendation system

Define RMSE function, this will measure how far our predictions are from true rating

```
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

For this model we will use a simple average of the training set for prediction

```
mu <- mean(train_set$rating)
mu
```

```
## [1] 4.065365
```

Compute RMSE on the test set

```r
average_rmse <- RMSE(test_set$rating, mu)
average_rmse
```

```
## [1] 1.161623
```

Show RMSE in a clean way using knitr

```r
rmse_results <- data_frame(method = "Average Hotel Rating Model", RMSE = average_rmse)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```r
rmse_results %>% knitr::kable()
```

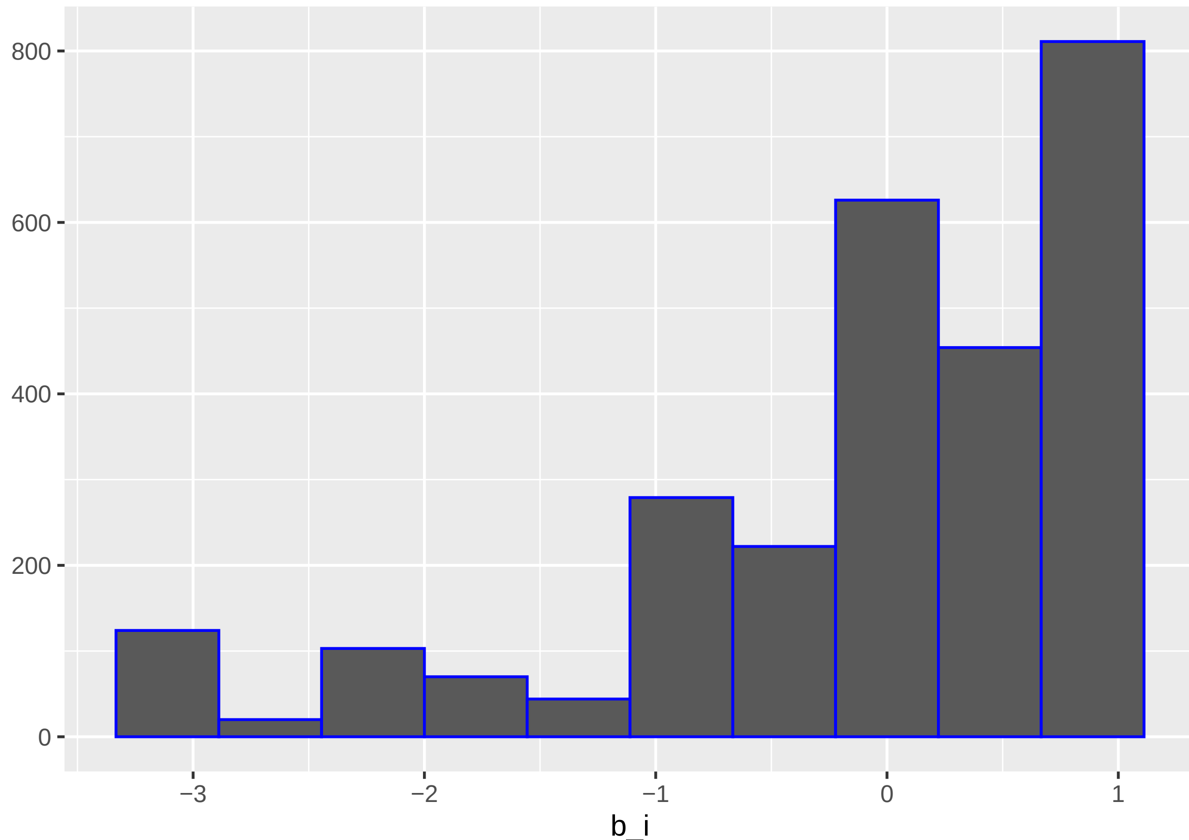
method	RMSE
Average Hotel Rating Model	1.161623

Now we will add the hotel effect to the model

```r
hotel_avgs <- train_set %>%
  group_by(hotel) %>%
  summarize(b_i = mean(rating - mu))
```

visualize how close ratings are to the mean

```r
hotel_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("blue"))
```



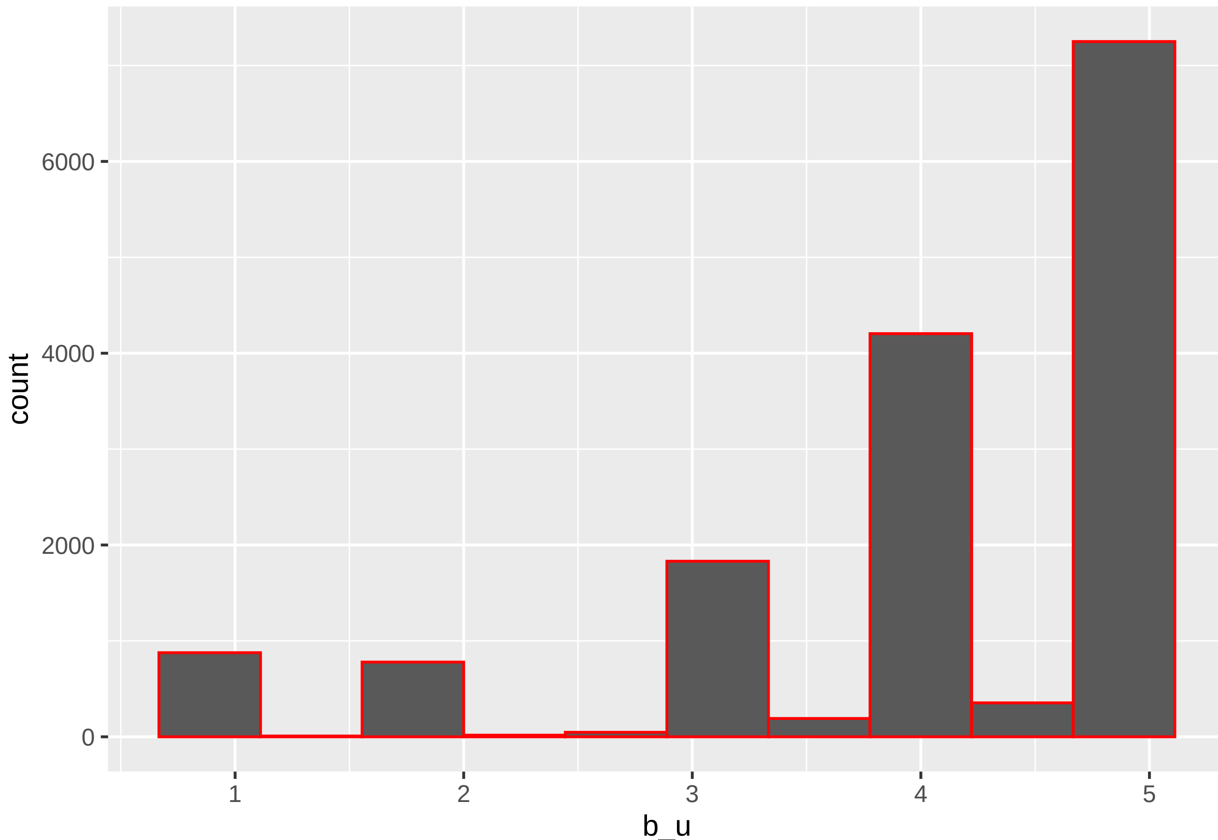
We will calculate predicted ratings

```
predicted_ratings <- mu + test_set %>%
  left_join(hotel_avgs, by='hotel') %>%
  .$b_i

model_1_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="hotel Effect Model",
                                     RMSE = model_1_rmse ))

rmse_results %>% knitr::kable()
```

method	RMSE
Average Hotel Rating Model	1.161623
hotel Effect Model	1.095019

Now we will add users to the previous model

```
train_set %>%
  group_by(user) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 10, color = "red")
```

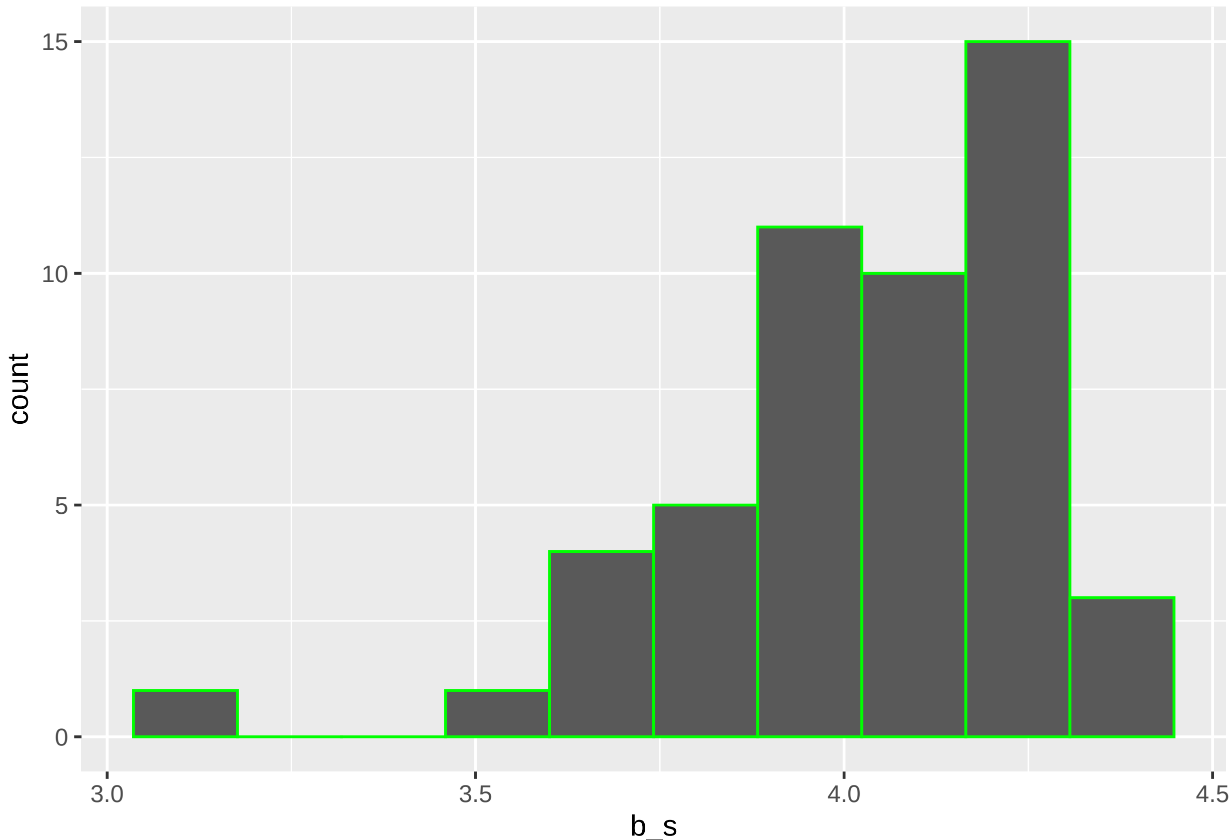



```
user_avgs <- train_set %>%
  left_join(hotel_avgs, by='hotel') %>%
  group_by(user) %>%
  summarize(b_u = mean(rating - mu - b_i))

predicted_ratings <- test_set %>%
  left_join(hotel_avgs, by='hotel') %>%
  left_join(user_avgs, by='user') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred

model_2_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Hotel + User Effects Model",
                                     RMSE = model_2_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Average Hotel Rating Model	1.161623
hotel Effect Model	1.095019
Hotel + User Effects Model	1.194913

We will add state to the previous model, and follow similar process as previous lines of code

```r
train_set %>%
  group_by(state) %>%
  summarize(b_s = mean(rating)) %>%
  ggplot(aes(b_s)) +
  geom_histogram(bins = 10, color = "green")
```



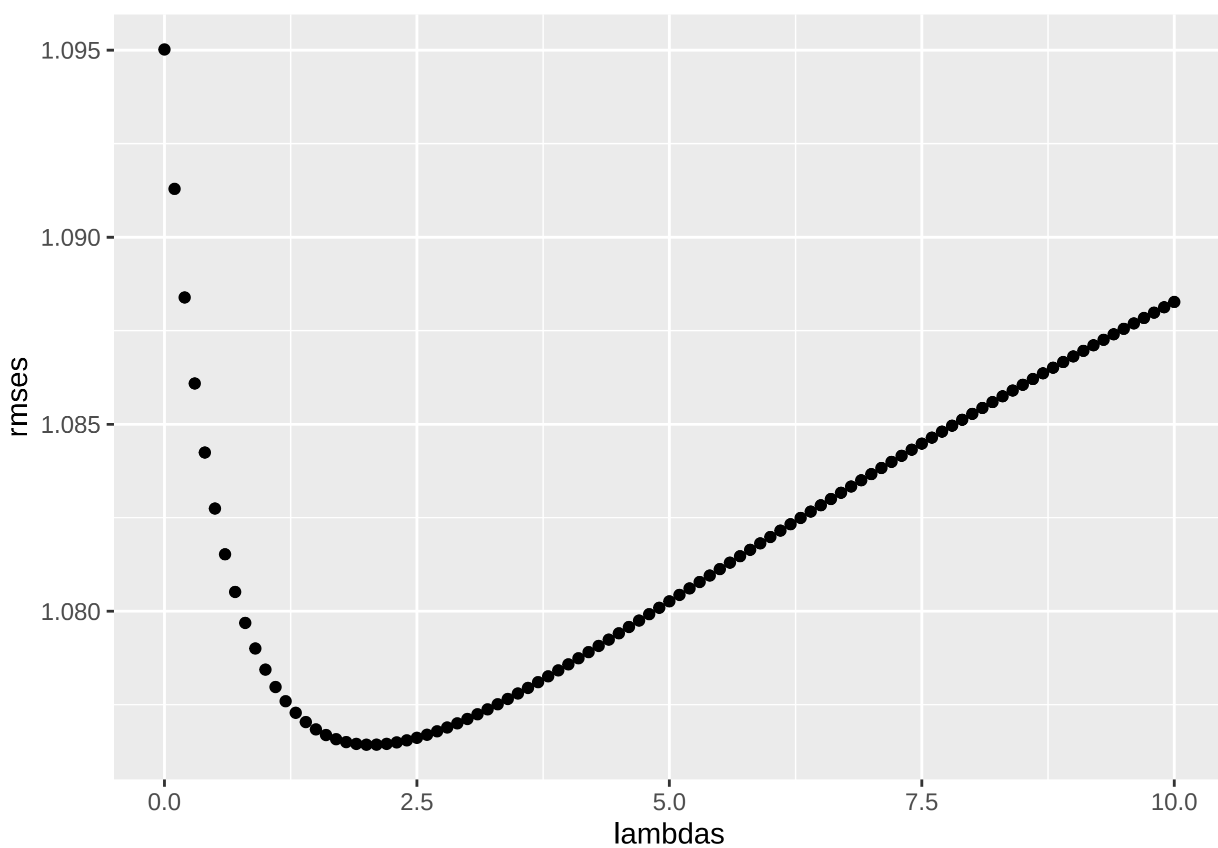

```r
state_avgs <- train_set %>%
  left_join(hotel_avgs, by='hotel') %>%
  left_join(user_avgs, by='user') %>%
  group_by(state) %>%
  summarize(b_s = mean(rating - mu - b_i - b_u))

predicted_ratings <- test_set %>%
  left_join(hotel_avgs, by='hotel') %>%
  left_join(user_avgs, by='user') %>%
  left_join(state_avgs, by='state') %>%
  mutate(pred = mu + b_i + b_u + b_s) %>%
  .$pred

model_3_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Hotel + User + State Effects Model",
                                     RMSE = model_3_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Average Hotel Rating Model	1.161623
hotel Effect Model	1.095019
Hotel + User Effects Model	1.194913
Hotel + User + State Effects Model	1.195006

Regularization

Now we will try using regularization on Hotel to improve RMSE score

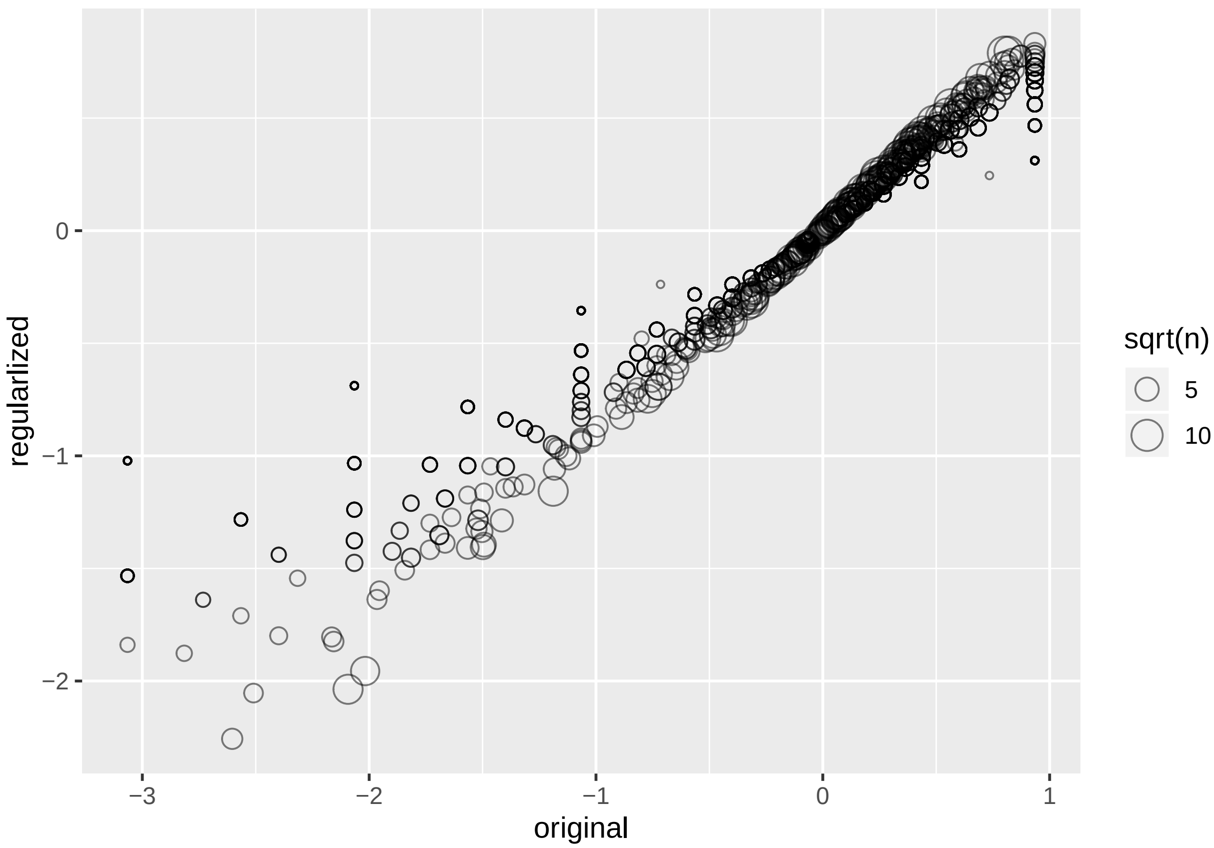
```
lambdas <- seq(0, 10, 0.1)
mu <- mean(train_set$rating)
just_the_sum <- train_set %>%
  group_by(hotel) %>%
  summarize(s = sum(rating - mu), n_i = n())
rmses <- sapply(lambdas, function(l){
  predicted_ratings <- test_set %>%
    left_join(just_the_sum, by='hotel') %>%
    mutate(b_i = s/(n_i+l)) %>%
    mutate(pred = mu + b_i) %>%
    .$pred
  return(RMSE(predicted_ratings, test_set$rating))
})
qplot(lambdas, rmses)
```



```
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 2
```

```
mu <- mean(train_set$rating)
hotel_reg_avgs <- train_set %>%
  group_by(hotel) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())

data_frame(original = hotel_avgs$b_i,
           regularlized = hotel_reg_avgs$b_i,
           n = hotel_reg_avgs$n_i) %>%
  ggplot(aes(original, regularlized, size=sqrt(n))) +
  geom_point(shape=1, alpha=0.5)
```

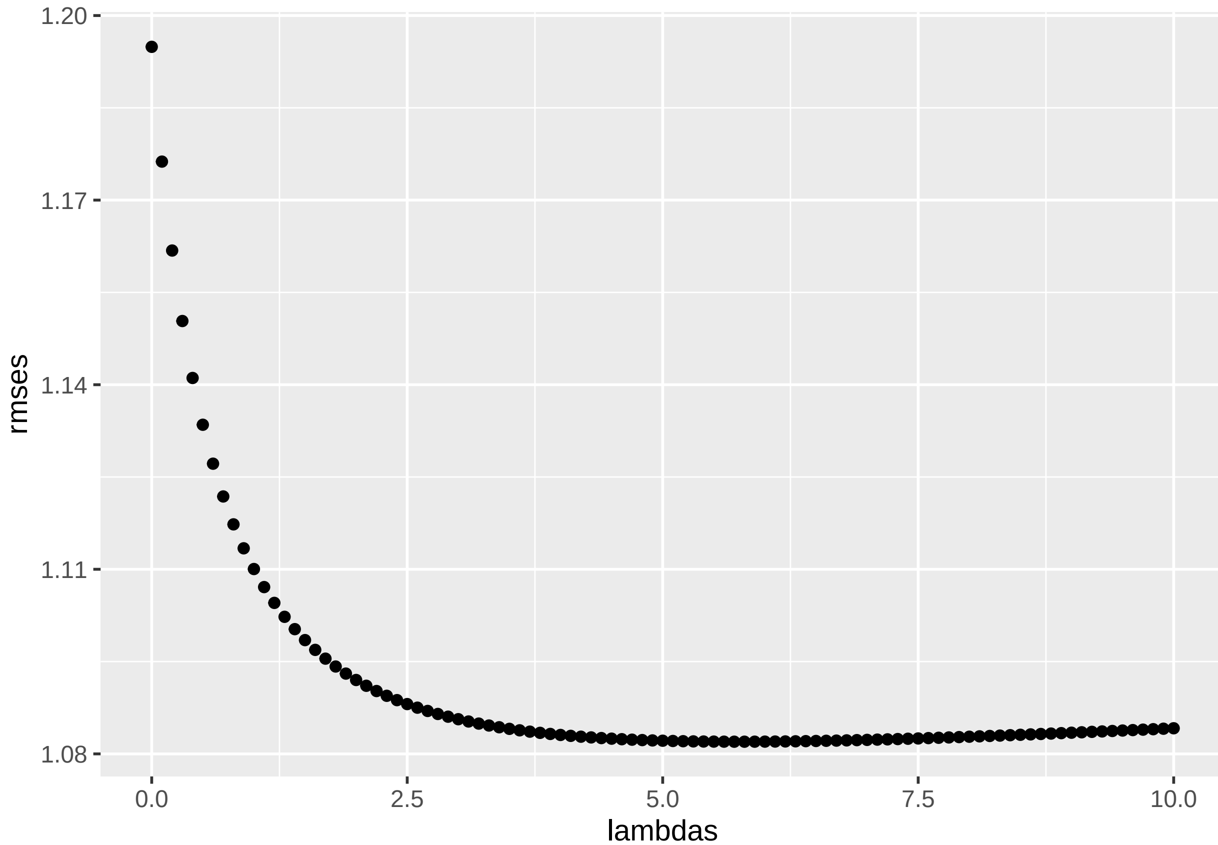



```
predicted_ratings <- test_set %>%
  left_join(hotel_reg_avgs, by='hotel') %>%
  mutate(pred = mu + b_i) %>%
  .$pred

model_3_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized hotel Model",
                                     RMSE = model_3_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Average Hotel Rating Model	1.161623
hotel Effect Model	1.095019
Hotel + User Effects Model	1.194913
Hotel + User + State Effects Model	1.195006
Regularized hotel Model	1.076430

Now we will try using regularization on user to improve RMSE score, like with the user effect, it ends up making RMSE score worse

```
lambdas <- seq(0, 10, 0.1)
rmses <- sapply(lambdas, function(l){
  mu <- mean(train_set$rating)
  b_i <- train_set %>%
    group_by(hotel) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  b_u <- train_set %>%
    left_join(b_i, by="hotel") %>%
    group_by(user) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  predicted_ratings <-
    test_set %>%
    left_join(b_i, by = "hotel") %>%
    left_join(b_u, by = "user") %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred
  return(RMSE(predicted_ratings, test_set$rating))
})

qplot(lambdas, rmses)
```




```
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 5.8
```

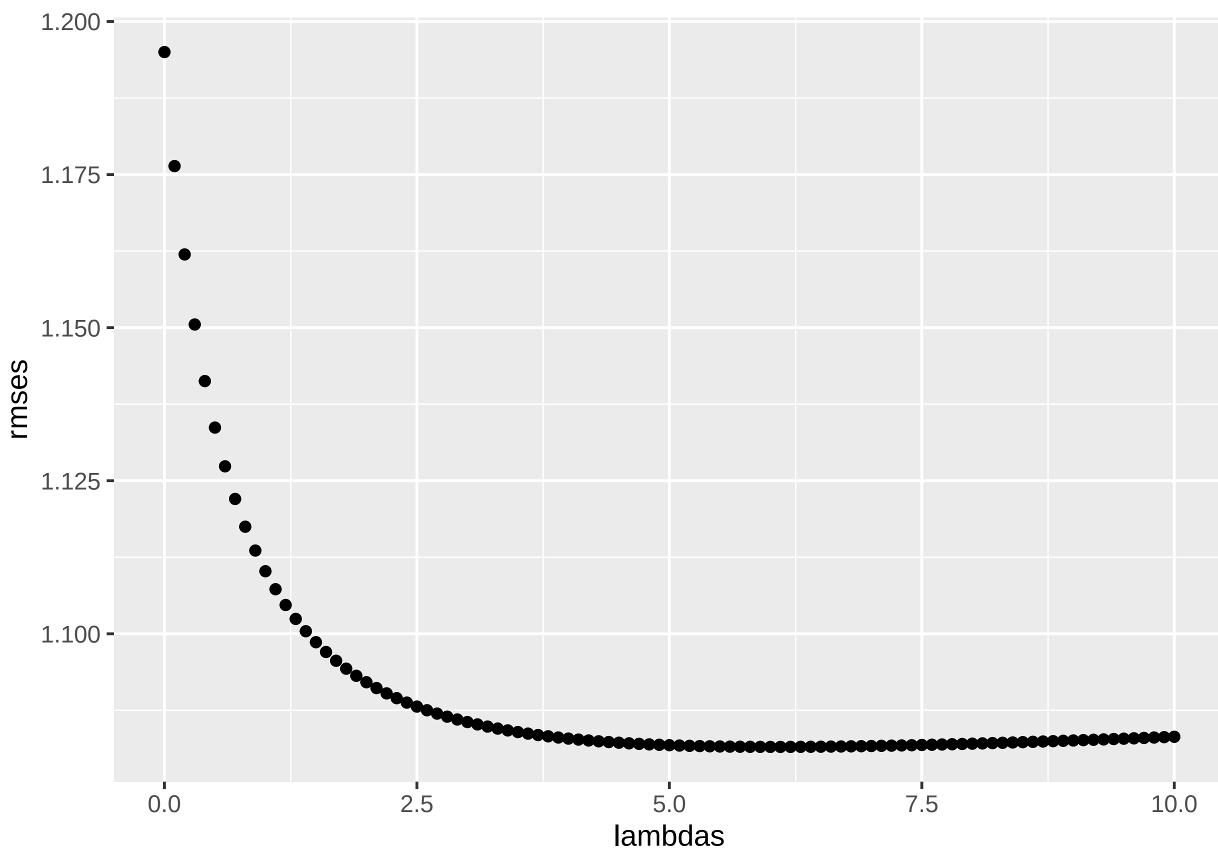
```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized hotel + User Effect Model",
                                     RMSE = min(rmses)))
rmse_results %>% knitr::kable()
```

method	RMSE
Average Hotel Rating Model	1.161623
hotel Effect Model	1.095019
Hotel + User Effects Model	1.194913
Hotel + User + State Effects Model	1.195006
Regularized hotel Model	1.076430
Regularized hotel + User Effect Model	1.081978

Now we will try using regularization on state to improve RMSE score

```
lambdas <- seq(0, 10, 0.1)
rmses <- sapply(lambdas, function(l){
  mu <- mean(train_set$rating)
```

```r
  b_i <- train_set %>%
    group_by(hotel) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  b_u <- train_set %>%
    left_join(b_i, by="hotel") %>%
    group_by(user) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  b_s <- train_set %>%
    left_join(b_i, by="hotel") %>%
    left_join(b_u, by="user") %>%
    group_by(state) %>%
    summarize(b_s = sum(rating - b_i - b_u - mu)/(n()+l))
  predicted_ratings <-
    test_set %>%
    left_join(b_i, by = "hotel") %>%
    left_join(b_u, by = "user") %>%
    left_join(b_s, by = "state") %>%
    mutate(pred = mu + b_i + b_u + b_s) %>%
    pull(pred)
  return(RMSE(predicted_ratings, test_set$rating))
})

qplot(lambdas, rmses)
```



```r
lambda <- lambdas[which.min(rmses)]
lambda
```

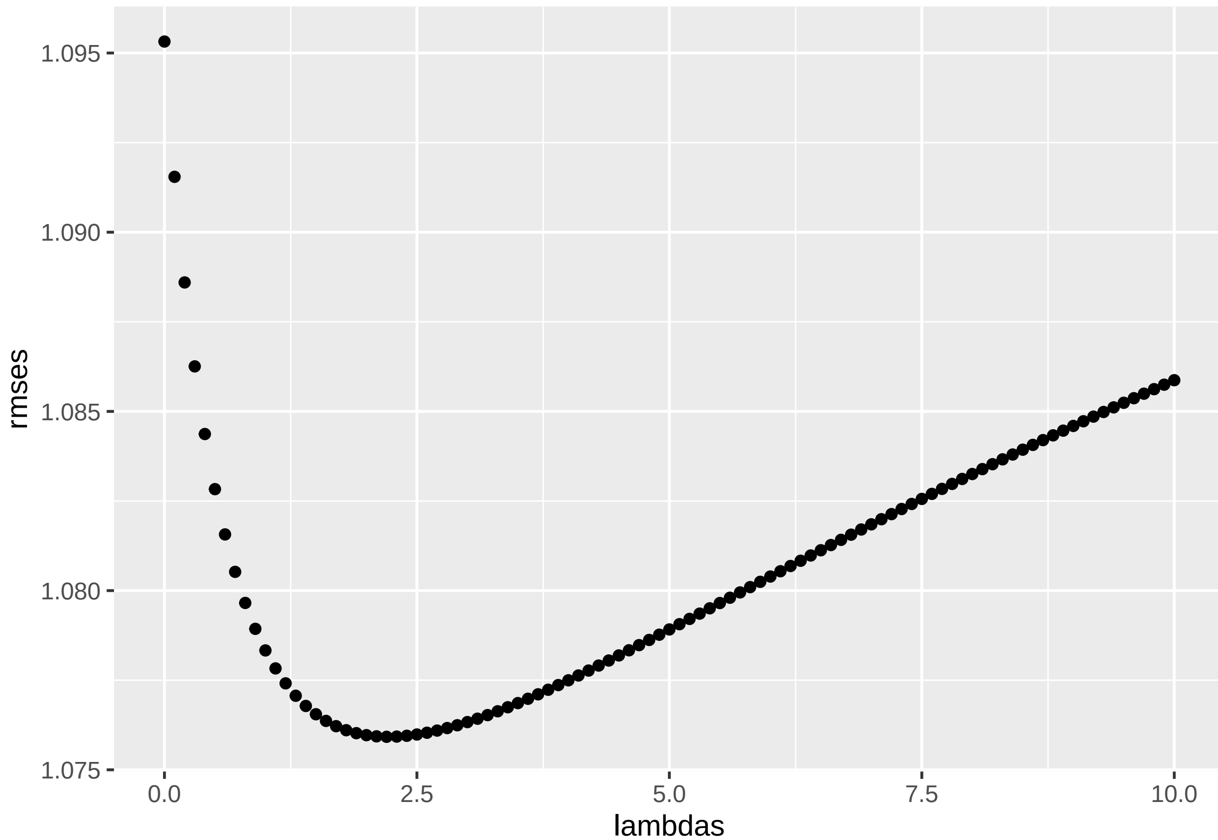
```
## [1] 6.1
```

```r
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized hotel + User Effect Model+ State",
                                     RMSE = min(rmses)))
rmse_results %>% knitr::kable()
```

method	RMSE
Average Hotel Rating Model	1.161623
hotel Effect Model	1.095019
Hotel + User Effects Model	1.194913
Hotel + User + State Effects Model	1.195006
Regularized hotel Model	1.076430
Regularized hotel + User Effect Model	1.081978
Regularized hotel + User Effect Model+ State	1.081522

Hotel and State regularization effect without user, since user was negatively affected our score

```r
lambdas <- seq(0, 10, 0.1)
rmses <- sapply(lambdas, function(l){
  mu <- mean(train_set$rating)
  b_i <- train_set %>%
    group_by(hotel) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  b_s <- train_set %>%
    left_join(b_i, by="hotel") %>%
    group_by(state) %>%
    summarize(b_s = sum(rating - b_i - mu)/(n()+l))
  predicted_ratings <-
    test_set %>%
    left_join(b_i, by = "hotel") %>%
    left_join(b_s, by = "state") %>%
    mutate(pred = mu + b_i + b_s) %>%
    .$pred
  return(RMSE(predicted_ratings, test_set$rating))
})

qplot(lambdas, rmses)
```




```
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 2.2
```

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized hotel + State Effect Model",
                                     RMSE = min(rmses)))
rmse_results %>% knitr::kable()
```

method	RMSE
Average Hotel Rating Model	1.161623
hotel Effect Model	1.095019
Hotel + User Effects Model	1.194913
Hotel + User + State Effects Model	1.195006
Regularized hotel Model	1.076430
Regularized hotel + User Effect Model	1.081978
Regularized hotel + User Effect Model+ State	1.081522
Regularized hotel + State Effect Model	1.075921

Conclusion

After trying 7 different models, the regularized regression model that took hotel and state into account outperformed all other models with an RMSE of 1.075. I found this dataset a bit frustrating that I could not bring the RMSE below 1.0. It was interesting to see that unlike in movie ratings, hotel ratings are generally a lot higher, where 5 and 4 were the most common, but 1 was the third most common. I found it interesting that taking user effect actaully made the RMSE worse. I believe with a larger dataset we would have been able to bring the RMSE down below 1.0.