Hotel Recommendation System

Florian Javelle

18/05/2021

Contents

1. Introduction

Recommendation systems can be broadly defined as algorithms emitting specific recommendations to users based on their former ratings or purchases of items [1]. As theoretically presenting the most relevant items to users, the accuracy of these recommendation systems is of major importance for a wide spectrum of companies (e.g., Netflix, Amazon, Facebook). In other words, the more accurate the recommendation system will be, the more likely the user will like the presented items, and therefore, the more likely the company will generate large incomes.

In the first assignment of the Capstone course [2], students were asked to put in evidence the utility and the functioning of recommendation systems via developing an algorithm predicting movie ratings using the 10 Million version MovieLens data set. To do so, students were asked to create an algorithm with a root mean squared error (RMSE) inferior to 0.8649 (for ratings going from 1 to 5). The RMSE is a scale of precision that has the same unit as the predicted variable [1]. The smaller it is, the more accurate is the recommendation system.

Using the same concept, but a different data set, different predictors and additional machine learning techniques (i.e., singular value decomposition and principal component analysis [PCA]), **this report aims to create a recommendation system for hotel selection** based on data scraped from Booking.com [3]. **The secondary aim of this report is to compare the accuracy of a regression-based versus PCA recommendation system.** The predicted variable will be the review score given by the reviewer after his/her stay at the hotel. I will first explore the data set, reshape it if necessary, and then determine potential predictors to include in my recommendation system. Then, I will construct a recommendation system for hotel selection. I will continue by comparing the accuracy of a regular (regression-based) recommendation system and one based on PCA. I will finish by discussing the differences between the two systems.

1.1 Brief data description (from kaggle)

This data set has been downloaded from kaggle interface (https://www.kaggle.com/jiashenliu/515k-hotel-reviews-data-in-europe) [3] and is available via the following Dropbox link (https://www.dropbox.com/s/pehhztgk97ftvsu/Hotel.zip?dl=0) [4] for this report.
The emitter of the data set, Jiashen Liu, uploaded it in 2017 after scraping the data from Booking.com (a hotel selection interface) [3]. This large data set contains 515 738 reviews, rating 1492 European luxury hotels. It includes 17 variables, among which three would require text-mining to be potentially useful in a recommendation system (i.e., “Positive_Review”, “Negative_Review”, and “Tags”). As the purpose of this report is not to demonstrate text-mining abilities, I will not use these variables.

1.2 Workspace preparation

```r
# To clear workspace
rm(list = ls())
# To clear console
cat("\014")
```

```r
# To install and charge the libraries
if(!require(colorspace)) install.packages("colorspace", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
library(colorspace)
library(ggplot2)
library(tidyverse)
library(caret)
library(data.table)
library(dplyr)
```

1.3 Charge the dataset

The data set was too big to be uploaded on GitHub, even when compressed. Thus, I did create a public link on my Dropbox to access it [4].

```r
# To download the data from Dropbox
temp <- tempfile()
download.file("https://www.dropbox.com/s/lkxeeb7kp0ryma8/Hotel_Reviews.csv?dl=1",temp)
Data<-read.csv(temp, header=TRUE, sep=";")
unlink(temp)
```

As previously mentioned, I will not perform text mining in this report, therefore to facilitate the visualization of the data, I am selecting only 14 out of the 17 variables available (variables excluded: “Positive_Review”, “Negative_Review” and “Tags”).

```r
# To select the useful variables
Data<-Data%>%
  select(Hotel_Address, Additional_Number_of_Scoring, Review_Date, Average_Score,
         Hotel_Name, Reviewer_Nationality, Review_Total_Negative_Word_Counts,
         Total_Number_of_Reviews, Review_Total_Positive_Word_Counts,
         Total_Number_of_Reviews_Reviewer_Has_Given, Reviewer_Score, days_since_review,
         lat, lng)
```

Ideally, the accuracy of a machine-learning algorithm should be tested on newly gathered data from the real world. Nevertheless, I am developing an algorithm to create a recommendation system where I already know the outcome (i.e., Reviewer_Score). Therefore, I need to split the data set to mimic the evaluation process. As I am using a very large data set with more than 515 000 reviews, I can go up to a split of 90%/10% between training and validation sets. This will maximize the precision of the training step and will let an acceptable sample size to the validation set that will properly represent real-world results’ variance. Furthermore, to be able to tune my algorithm before testing it, I will also split my training set (i.e., Hotel) into training (i.e., train_set) and test sets (i.e., test_set). Considering the size of the data set, I will also use a 90%/10% split.

```r
# To split the data set into Hotel and validation sets
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = Data$Reviewer_Score, times = 1, p = 0.1, list = FALSE)
Hotel <- Data[-test_index,]
temp <- Data[test_index,]
```

```r
# To be sure that all Hotels, reviewer nationalities, and total number of reviews per reviewer
# are in the "validation" and "Hotel" sets. Total_Number_of_Reviews_Reviewer_Has_Given and
# "Reviewer_Nationality" were added after the first analysis.
# We will see later that it is important to have all the levels of these 2 variables in both sets.
validation <- temp %>%
  semi_join(Hotel, by = "Hotel_Name")%>%
  semi_join(Hotel, by = "Total_Number_of_Reviews_Reviewer_Has_Given")%>%
  semi_join(Hotel, by = "Reviewer_Nationality")


# To split the Hotel set into training and testing sets
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = Hotel$Reviewer_Score, times = 1, p = 0.1, list = FALSE)
train_set <- Data[-test_index,]
test_set <- Data[test_index,]

test_set <- test_set %>%
  semi_join(train_set, by = "Hotel_Name")%>%
  semi_join(train_set, by = "Total_Number_of_Reviews_Reviewer_Has_Given")%>%
  semi_join(train_set, by = "Reviewer_Nationality")
```

1.4 Data set description (from R)

The following functions will help me to get an idea of the size, and characteristics of the training data set and its variables.

```r
# To see how train_set is constructed
glimpse(train_set)
```

```
## Rows: 469,320
## Columns: 14
## $ Hotel_Address                              <chr> "Gudrunstra e 184 10 Favori~
## $ Additional_Number_of_Scoring               <int> 191, 9, 838, 954, 471, 134,~
## $ Review_Date                                <chr> "6/27/2017", "5/31/2016", "~
## $ Average_Score                              <dbl> 8.2, 7.9, 8.4, 8.6, 7.9, 8.~
## $ Hotel_Name                                 <chr> "Rainers Hotel Vienna", "Me~
## $ Reviewer_Nationality                       <chr> " Switzerland ", " United K~
## $ Review_Total_Negative_Word_Counts          <int> 23, 5, 32, 22, 113, 0, 9, 2~
## $ Total_Number_of_Reviews                    <int> 1852, 168, 3274, 4426, 2037~
## $ Review_Total_Positive_Word_Counts          <int> 66, 5, 3, 7, 9, 12, 6, 29, ~
## $ Total_Number_of_Reviews_Reviewer_Has_Given <int> 355, 355, 315, 315, 315, 29~
## $ Reviewer_Score                             <dbl> 9.2, 5.8, 7.5, 8.3, 6.3, 8.~
## $ days_since_review                          <chr> "37 days", "429 day", "127 ~
## $ lat                                        <dbl> 48.17919, 48.84957, 51.4956~
## $ lng                                        <dbl> 16.3625367, 2.3798518, -0.1~
```

```r
# To see the dimensions of train_set
dim(train_set)
```

```
## [1] 469320     14
```

```r
# To see the mean, median, max and min values of Reviewer_Score
summary(train_set$Reviewer_Score)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   2.500   7.500   8.800   8.395   9.600  10.000
```

```r
# To see the mean, median, max and min values of Average_Score
summary(train_set$Average_Score)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   5.200   8.100   8.400   8.398   8.800   9.800
```

```r
# To see how many different hotels there are in the data set
n_distinct(train_set$Hotel_Name)
```

```
## [1] 1492
```

```r
# To see how many different reviewer nationalities there are in the data set
n_distinct(train_set$Reviewer_Nationality)
```

```
## [1] 227
```

From these very basic functions, I have extracted important information that will direct the next steps of my work. First, the data set has been pre-analyzed. Indeed, two variables (i.e., "Total_Number_of_Reviews", "Total_Number_of_reviews_Reviewer_has_given") have already been extracted. Second, there is no user ID. This will decrease the accuracy of my recommendation system since the variability between individual user's scores will not be possible to evaluate. Nevertheless, I will still be able to use the variable "Total_Number_of_reviews_Reviewer_has_given created" as a predictor to get part of the missing information. Third, the information provided by the variables "Review_Date" and "days_since_review" overlap. This is the same thing for "Hotel_Address" and "lat"/"lng". In each case, I will just consider one of the presented variables. Fourth, the reviewers have many different nationalities (i.e., 227), and thus I will range them into continents to facilitate data visualization. However, even though, classifying it into continents might facilitate the visualization, it might lose information for the recommendation system. Therefore, I will make sure to also keep the original variable. Finally, the date does not have a usable format and thus I will have to change it.

Let's note that the reviewer scores are ranging from 2.5 to 10 out of 10, with a mean of 8.395. The average score is more constrained, going from 5.2 to 9.8 with a mean of 8.398.

2. Methods

In this section, I will adapt the parts of the data set that are currently not usable. Then, I will explore the data set and consider potential useful predictors. I will finish by briefly describe the machine-learning techniques that I will implement in the recommendation system.

2.1 Data set transformation

Before going further, I need to adapt few parameters from the data set. First, I have to change the date format.

```r
# To change the date format
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
library(lubridate)

train_set$Review_Date_correct<-mdy(train_set$Review_Date)
train_set$Review_Date_correct<-as_date(train_set$Review_Date_correct)
```

Second, I want to create a new variable entitle "Reviewer_continent" without overwriting the variable "Reviewer_Nationality".

```r
# To create a new variable entitled "Reviewer_continent" based on "Reviewer_Nationality"
if(!require(countrycode)) install.packages("countrycode", repos = "http://cran.us.r-project.org")
library(countrycode)

train_set$Reviewer_continent <- countrycode(sourcevar = train_set[, "Reviewer_Nationality"],
                              origin = "country.name",destination = "continent")

# Some country abbreviations are not recognized. Thus, I am classifying them manually.
train_set$Reviewer_continent <- as.character(train_set$Reviewer_continent)
train_set$Reviewer_continent[train_set$Reviewer_Nationality==
                              " Central Africa Republic "]<-"Africa"
train_set$Reviewer_continent[train_set$Reviewer_Nationality==" Antarctica "]<-"Antarctica"
train_set$Reviewer_continent[train_set$Reviewer_Nationality==" Cocos K I "]<-"Oceania"
train_set$Reviewer_continent[train_set$Reviewer_Nationality==" Cura ao "]<-"Americas"
train_set$Reviewer_continent[train_set$Reviewer_Nationality==" Crimea "]<-"Europe"
train_set$Reviewer_continent[train_set$Reviewer_Nationality==" Kosovo "]<-"Europe"
train_set$Reviewer_continent[train_set$Reviewer_Nationality==" Saint Barts "]<-"Americas"
train_set$Reviewer_continent[train_set$Reviewer_Nationality==" Saint Martin "]<-"Americas"
train_set$Reviewer_continent[train_set$Reviewer_Nationality==
                              " United States Minor Outlying Islands "]<-"Americas"
```

Finally, I want to extract the city where each hotel is located based on the variable "Hotel_Address". If there are too many different cities, I will also extract the country.

```r
# To extract the city of each hotel based on "Hotel_Address"
if(!require(plyr)) install.packages("plyr", repos = "http://cran.us.r-project.org")
if(!require(maps)) install.packages("maps", repos = "http://cran.us.r-project.org")
library(plyr)
library(maps)
# To split the variable Hotel_Address in different sections based on the space between words
train_set$Hotel_Address<-as.character(train_set$Hotel_Address)
addresses<-strsplit(train_set$Hotel_Address, " ")
# Match on cities in "world cities"
train_set$Hotel_city<- llply(addresses, function(x)x[max(which(x %in% world.cities$name))])
train_set$Hotel_city<-as.character(train_set$Hotel_city)
n_distinct(train_set$Hotel_city)
```

```
## [1] 6
```

```r
length(train_set$Hotel_city)
```

```
## [1] 469320
```

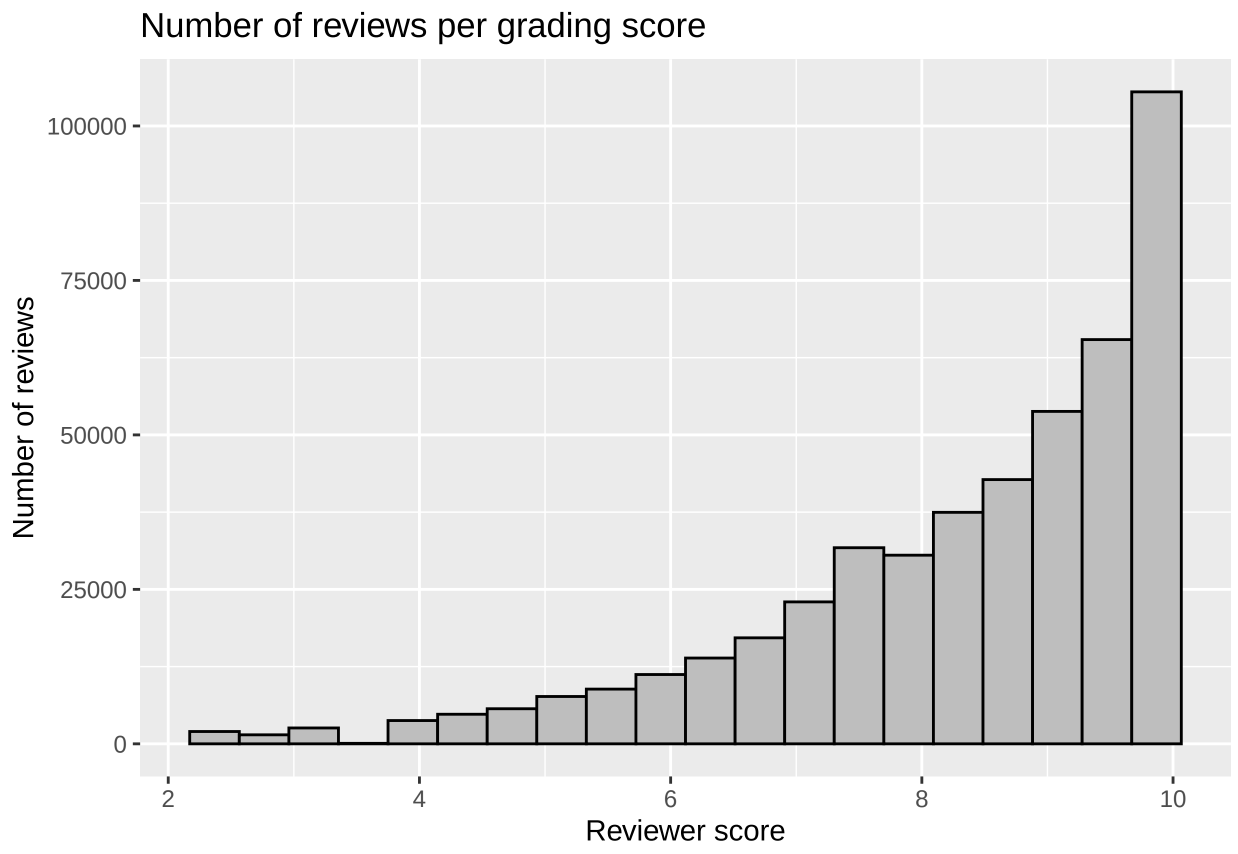
```r
# The packages plyr and dplyr do not work very well when charged together.
# They overlap a lot. To avoid issues, I discharge plyr after use.
detach("package:plyr", unload=TRUE)
```

The 1492 hotels are located in only 6 distinct cities from 6 different countries, thus I do not need to create a new categorical variable reporting the country from each hotel.

2.2 Data exploration

Now, I can begin to explore the data. First, I am interested in knowing the distribution of the target variable for the recommendation system; "Reviewer_Score".

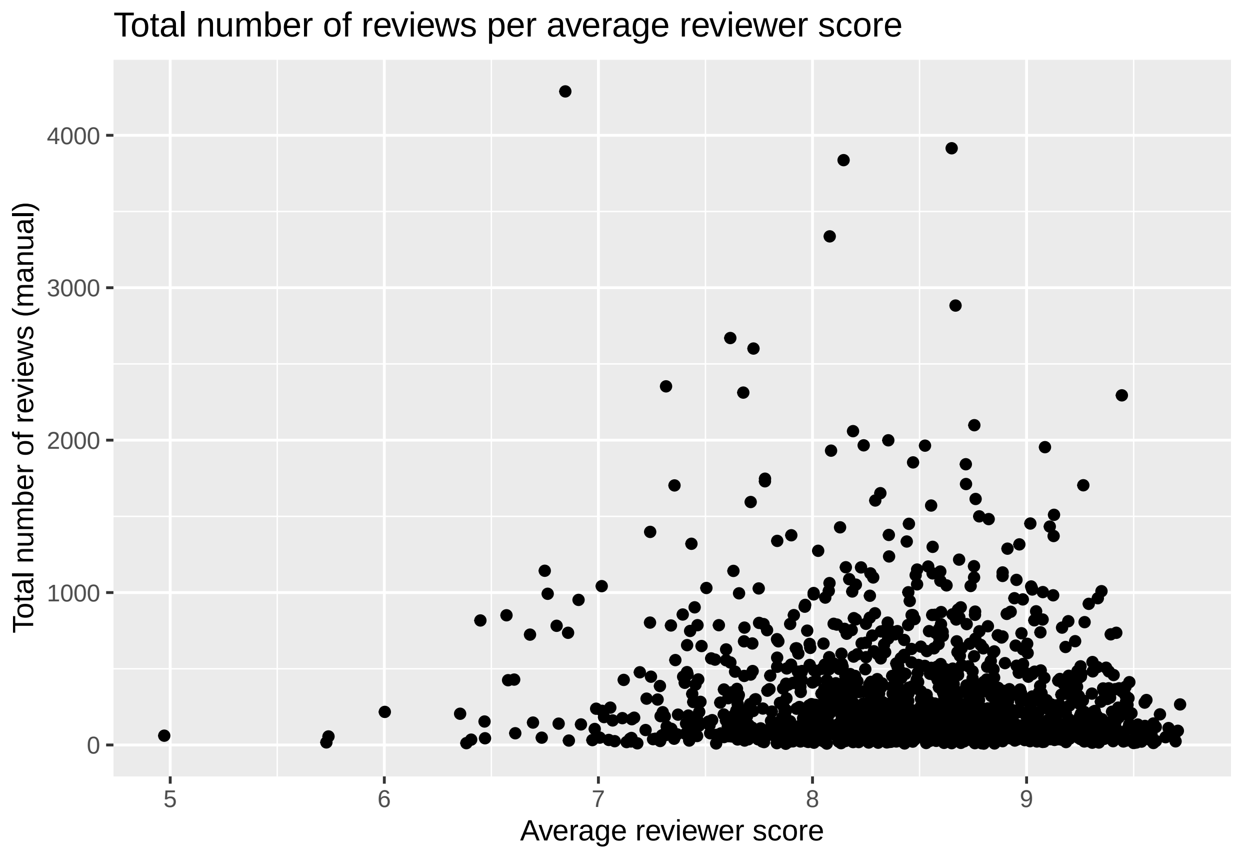
```r
# To see the "Reviewer_Score" distribution
train_set %>%
  group_by(Reviewer_Score)%>%
  ggplot(aes(Reviewer_Score)) +
  geom_histogram(bins = 20, fill="grey", colour="black")+
  labs(x="Reviewer score", y="Number of reviews")+
  ggtitle("Number of reviews per grading score")
```



The graphic shows that when the reviewer score increases, the number of reviews does too. This result is relatively logical. The better the hotel is, the more people come.

Next, I am going to average the score per hotel and evaluate its distribution.

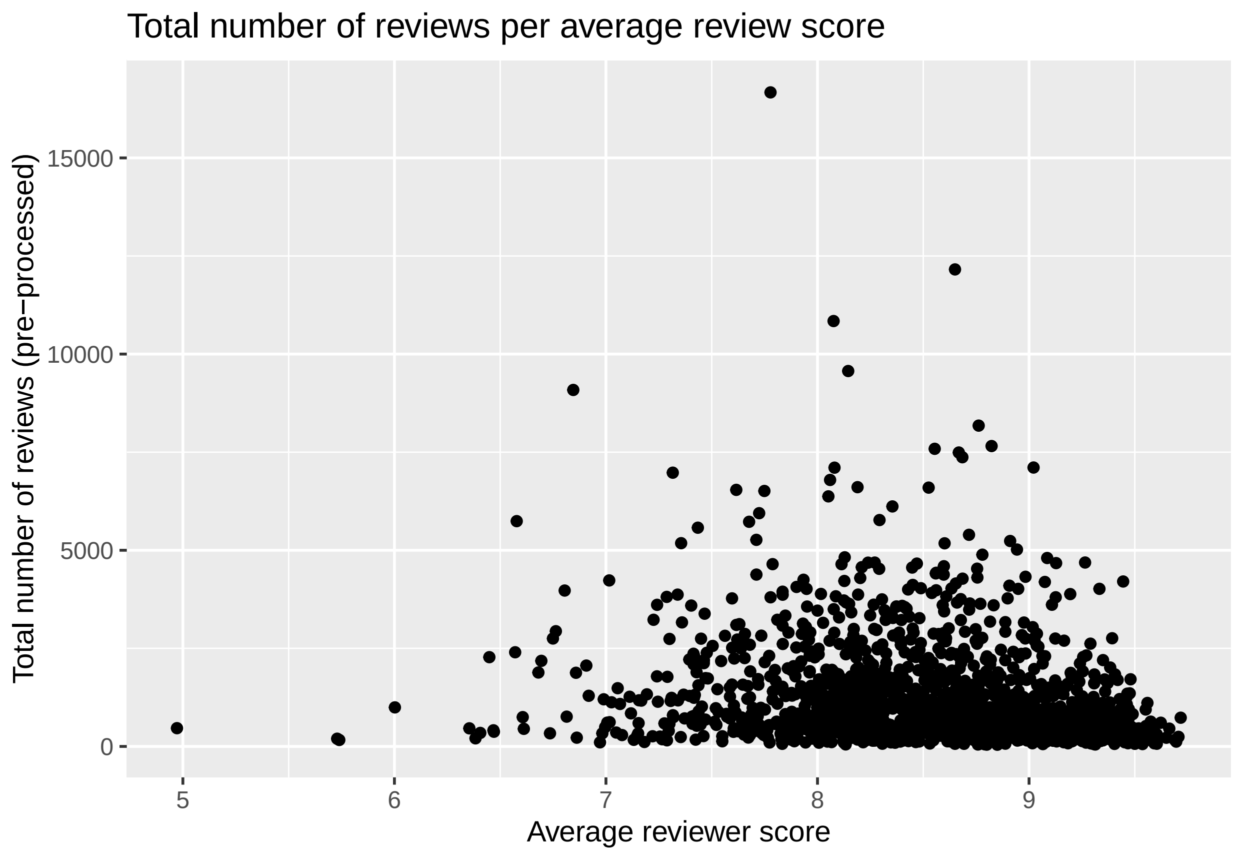
```r
# To see the average "Reviewer_Score" distribution
train_set %>%
  group_by(Hotel_Name)%>%
  summarize (avg_rating=mean(Reviewer_Score), n=n())%>%
  ggplot(aes(avg_rating, n))+ geom_point() + labs(x="Average reviewer score",
                                                   y="Total number of reviews (manual)")+
  ggtitle("Total number of reviews per average reviewer score")
```



The graphic shows that the best and the worst reviewer average scores have only a few ratings. This information suggests that it might be useful to implement regularization techniques (i.e., to penalize large estimates that are created because of small sample sizes [1]) in the recommendation system.

Considering that there is a variable reporting the total number of reviews (i.e., Total_Number_of_Reviews), I am going to verify if I obtain the same results.

```r
# To see the average "Reviewer_Score" distribution
# (using the pre-processed variable "Total_Number_of_Reviews")
train_set %>% group_by(Hotel_Name)%>%
  summarize (avg_rating=mean(Reviewer_Score), avg_n=mean(Total_Number_of_Reviews))%>%
  ggplot(aes(avg_rating, avg_n))+ geom_point()+ labs(x="Average reviewer score",
                                          y="Total number of reviews (pre-processed)")+
  ggtitle("Total number of reviews per average review score")
```



Even though the graphics look very similar, they are not the same. Furthermore, the range of the variable "Total_number_of_Reviews" is three times bigger than when I've computed it manually. This variable might take into account a longer period than the 2 years of reviews gathered in the data set. The other explanation could be that the data set I use, even though huge, does not include all reviews emitted between 2015 and 2017. By curiosity, I am going to check if it changes the top 10 of hotels the most rated.

```
# To see the most rated hotels (manual)
train_set %>%
  group_by(Hotel_Name)%>%
  summarize (n=n())%>%
  top_n(10) %>%
  arrange(desc(n))
```

```
## # A tibble: 10 x 2
##    Hotel_Name                                        n
##    <chr>                                         <int>
##  1 Britannia International Hotel Canary Wharf     4288
##  2 Park Plaza Westminster Bridge London           3915
##  3 Strand Palace Hotel                            3837
##  4 Copthorne Tara Hotel London Kensington         3337
##  5 DoubleTree by Hilton Hotel London Tower of London  2883
##  6 Grand Royale London Hyde Park                  2670
##  7 Holiday Inn London Kensington                  2601
##  8 Hilton London Metropole                        2353
##  9 Millennium Gloucester Hotel London             2312
## 10 Intercontinental London The O2                 2294
```

```r
# To see the most rated hotels (pre-processed)
train_set %>%
  group_by(Hotel_Name)%>%
  summarize (Total_Number_of_Reviews=mean(Total_Number_of_Reviews))%>%
  arrange(desc(Total_Number_of_Reviews))
```

```
## # A tibble: 1,492 x 2
##    Hotel_Name                                        Total_Number_of_Reviews
##    <chr>                                                               <dbl>
##  1 Hotel Da Vinci                                                      16670
##  2 Park Plaza Westminster Bridge London                                12158
##  3 Hotel degli Arcimboldi                                              10842
##  4 Strand Palace Hotel                                                  9568
##  5 Britannia International Hotel Canary Wharf                           9086
##  6 Best Western Premier Hotel Couture                                   8177
##  7 The Student Hotel Amsterdam City                                     7656
##  8 Golden Tulip Amsterdam West                                          7586
##  9 DoubleTree by Hilton Hotel London Tower of London                    7491
## 10 Glam Milano                                                          7371
## # ... with 1,482 more rows
```

These lists confirm that the two variables analysed (i.e., Total number of reviews manual or pre-existing) are different. The pre-existing variable "Total_Number_of_reviews" reports between 3 and 4 times more reviews per hotel than when I compute it manually. The top 10 also changes.

This might be the same thing for the other variable that I was thinking pre-processed (i.e., Average_score). Thus, I am also going to print the top 10 of average reviewer scores for both cases (i.e., manually computed versus pre-existing variable)

```r
# To see the best rated hotels (manual)
train_set %>%
  group_by(Hotel_Name)%>%
  summarize (avg_Reviewer_score=mean(Reviewer_Score))%>%
  arrange(desc(avg_Reviewer_score))
```

```
## # A tibble: 1,492 x 2
##    Hotel_Name                                   avg_Reviewer_score
##    <chr>                                                     <dbl>
##  1 Hotel Casa Camper                                          9.72
##  2 41                                                         9.71
##  3 Ritz Paris                                                 9.70
##  4 H tel de La Tamise Esprit de France                        9.69
##  5 H10 Casa Mimosa 4 Sup                                      9.66
##  6 Le Narcisse Blanc Spa                                      9.65
##  7 Hotel The Serras                                           9.62
##  8 45 Park Lane Dorchester Collection                         9.60
##  9 Taj 51 Buckingham Gate Suites and Residences               9.60
## 10 Hotel Eiffel Blomet                                        9.59
## # ... with 1,482 more rows
```

```r
# To see the best rated hotels (manual)
train_set %>%
  group_by(Hotel_Name)%>%
  summarize (avg_Reviewer_score=mean(Average_Score))%>%
  arrange(desc(avg_Reviewer_score))
```

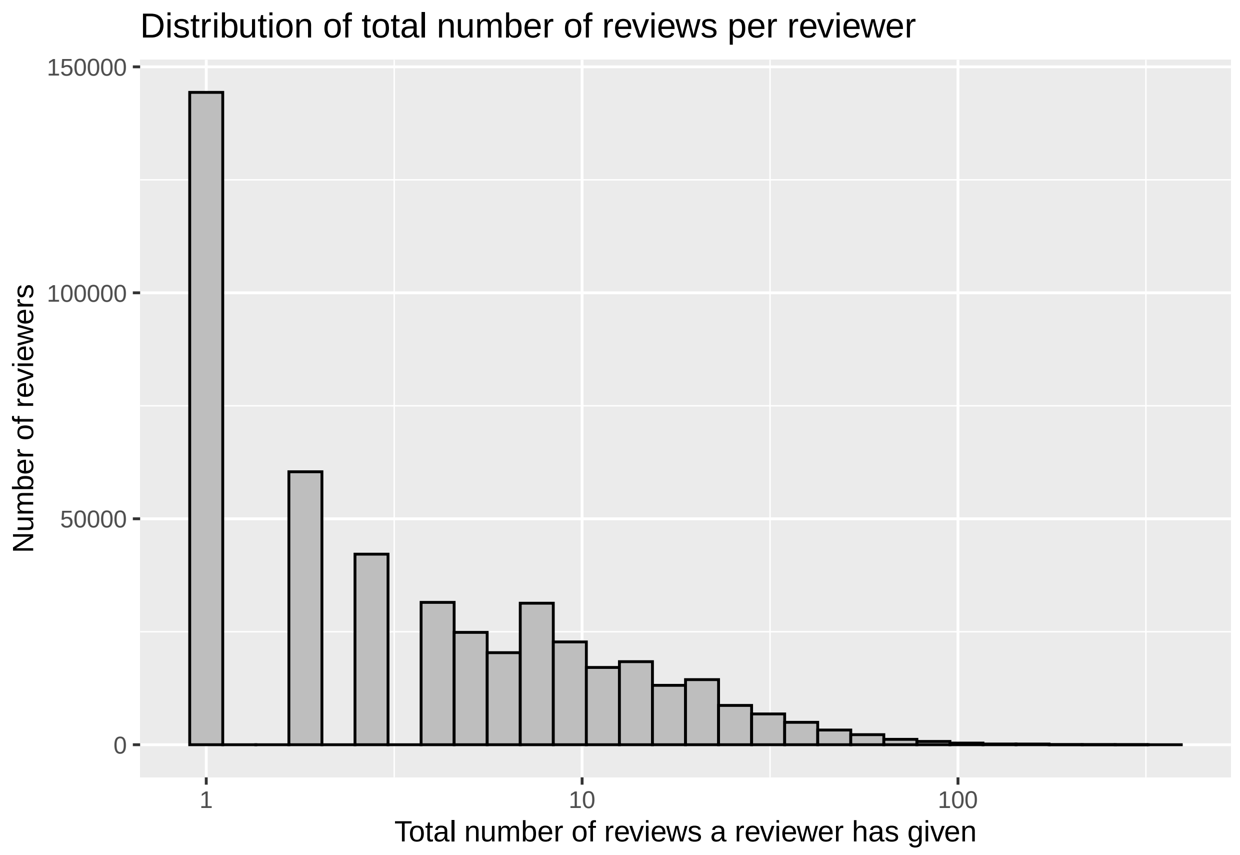
```
## # A tibble: 1,492 x 2
##    Hotel_Name                            avg_Reviewer_score
##    <chr>                                              <dbl>
##  1 Ritz Paris                                           9.8
##  2 41                                                   9.6
##  3 H tel de La Tamise Esprit de France                  9.6
##  4 H10 Casa Mimosa 4 Sup                                9.6
##  5 Haymarket Hotel                                      9.6
##  6 Hotel Casa Camper                                    9.6
##  7 Hotel The Serras                                     9.6
##  8 Charlotte Street Hotel                               9.5
##  9 Ham Yard Hotel                                       9.5
## 10 Hotel Sacher Wien                                    9.5
## # ... with 1,482 more rows
```

Once again, the tops 10 are relatively different.

On kaggle, the variable "Average_Score" is defined as the "Average Score of the hotel, calculated based on the latest comment in the last year". This description is not very clear, but I assume that it means that it is a yearly average score.

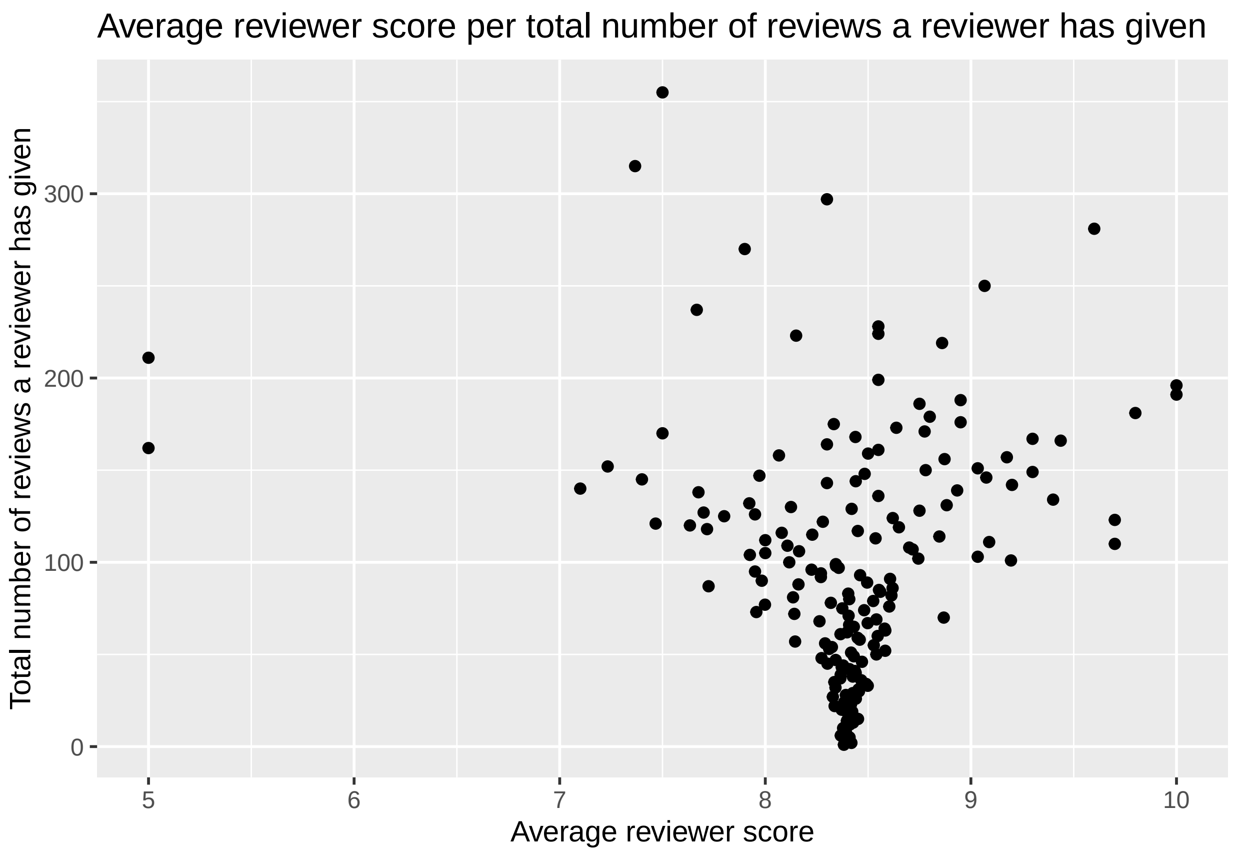
I am now going to assess the distribution of the number of reviews givens per reviewer.

```
# To see the distribution of "Total Number of Reviews a reviewers has given"
train_set %>%
  filter(!is.na(Total_Number_of_Reviews_Reviewer_Has_Given))%>%
  group_by(Total_Number_of_Reviews_Reviewer_Has_Given)%>%
  ggplot(aes(Total_Number_of_Reviews_Reviewer_Has_Given)) +
  geom_histogram(bins = 30, fill="grey", colour="black")+
  labs(x="Total number of reviews a reviewer has given", y="Number of reviewers")+
  ggtitle("Distribution of total number of reviews per reviewer") +
  scale_x_continuous(trans = "log10")
```



From this graph, it appears clear that there is a very large portion of users that gave only one review. I am going to assess if it does influence the average reviewer score.

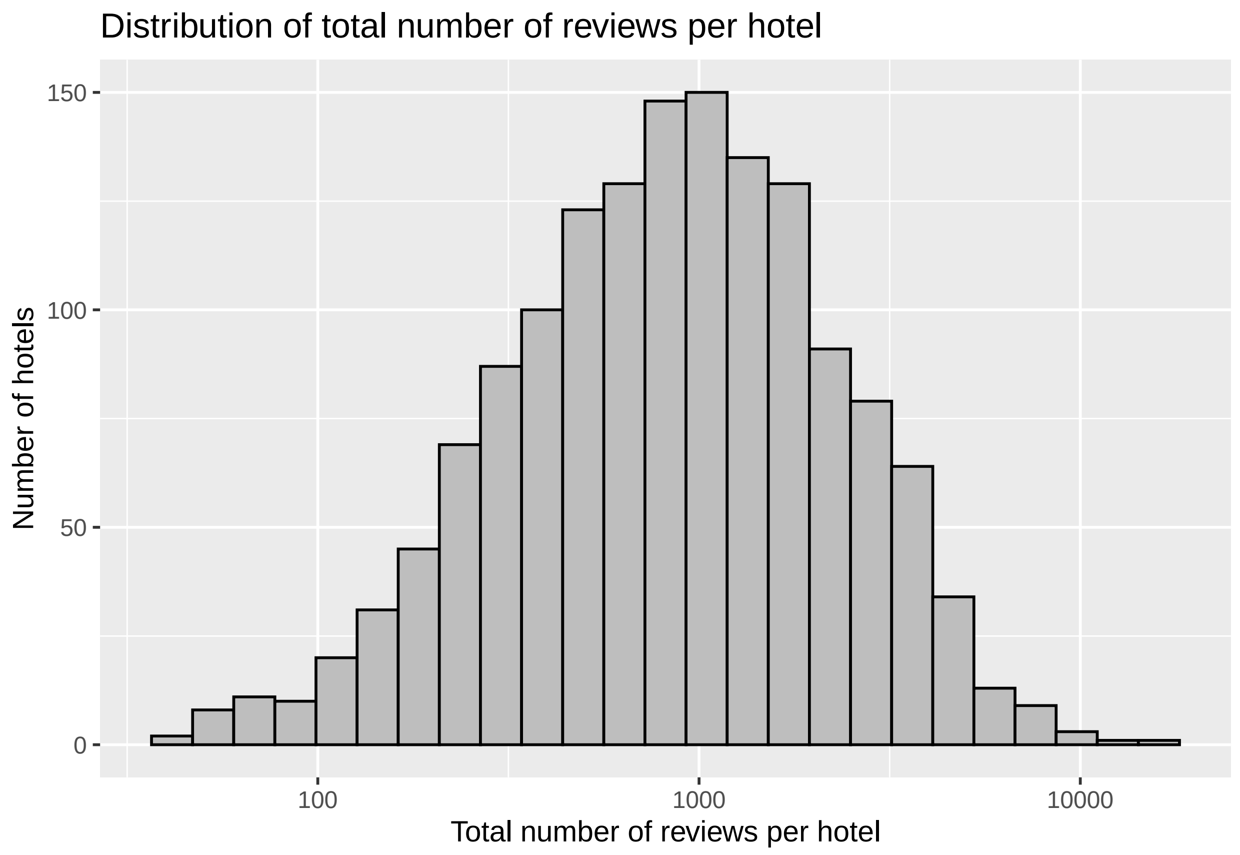
```
# To see the average reviewer score in function of the total number of reviews a reviewer
# has given
train_set %>%
  group_by(Total_Number_of_Reviews_Reviewer_Has_Given)%>%
  summarize (avg_rating=mean(Reviewer_Score),
             avg_n=mean(Total_Number_of_Reviews_Reviewer_Has_Given))%>%
  ggplot(aes(avg_rating, avg_n))+ geom_point()+ labs(x="Average reviewer score",
                                        y="Total number of reviews a reviewer has given")+
  ggtitle("Average reviewer score per total number of reviews a reviewer has given")
```



From this graph, it appears that when the total number of reviews a reviewer has given is low, the average score is consistent around 8.4. Nevertheless, when the total number of reviews a reviewer has given goes over 50, the average score becomes very heterogeneous (from 5 to 10) with some reviewers being very grumpy and others being very positive. Thus, I am going to use the variable "Total_Number_of_Reviews_Reviewer_Has_Given" as a potential predictor.

I am now going to evaluate the distribution of the total number of reviews per hotel.

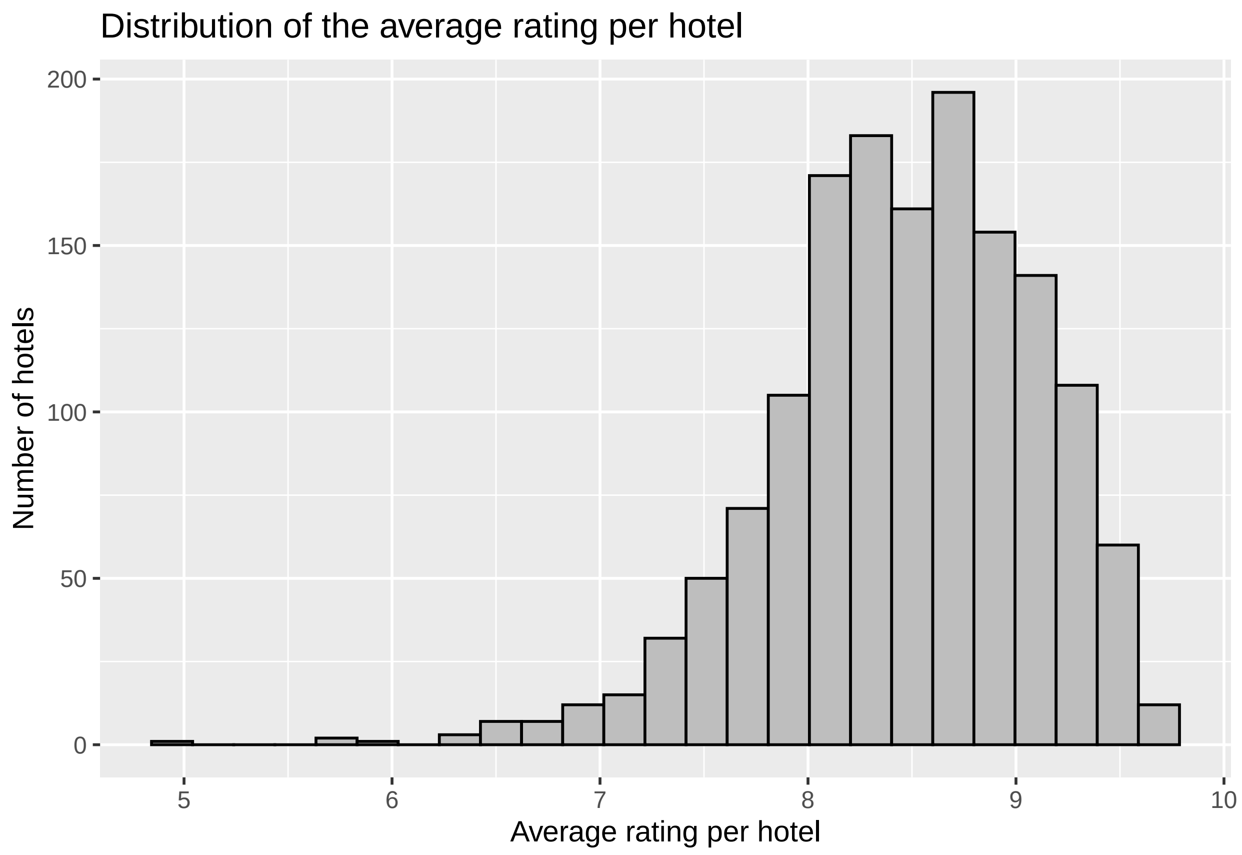
```r
# To see the distribution of Total Number of Reviews per hotel
train_set %>%
  group_by(Hotel_Name)%>%
  summarize(Total_Number_of_Reviews_avg=mean(Total_Number_of_Reviews))%>%
  ggplot(aes(Total_Number_of_Reviews_avg)) +
  geom_histogram(bins=25, fill="grey", colour="black")+
  labs(x="Total number of reviews per hotel", y="Number of hotels")+
  ggtitle("Distribution of total number of reviews per hotel")+
  scale_x_continuous(trans = "log10")
```



This very simple histogram displays that there are only a few hotels with a very small or very large number of reviews. This information comforts me in the idea that it might be important to penalize large estimates that are created because of the small sample size for hotels with a very small or very large number of reviews. Thus, I will implement a regularization technique to the recommendation system.

Next, I am going to evaluate if the number of reviews per hotel influences the average score.

```
# To see the distribution of the average reviewer score per hotel
train_set %>%
  group_by(Hotel_Name)%>%
  summarize(avg_rating=mean(Reviewer_Score))%>%
  ggplot(aes(avg_rating)) +
  geom_histogram(bins=25, fill="grey", colour="black")+
  labs(x="Average rating per hotel", y="Number of hotels")+
  ggtitle("Distribution of the average rating per hotel")
```



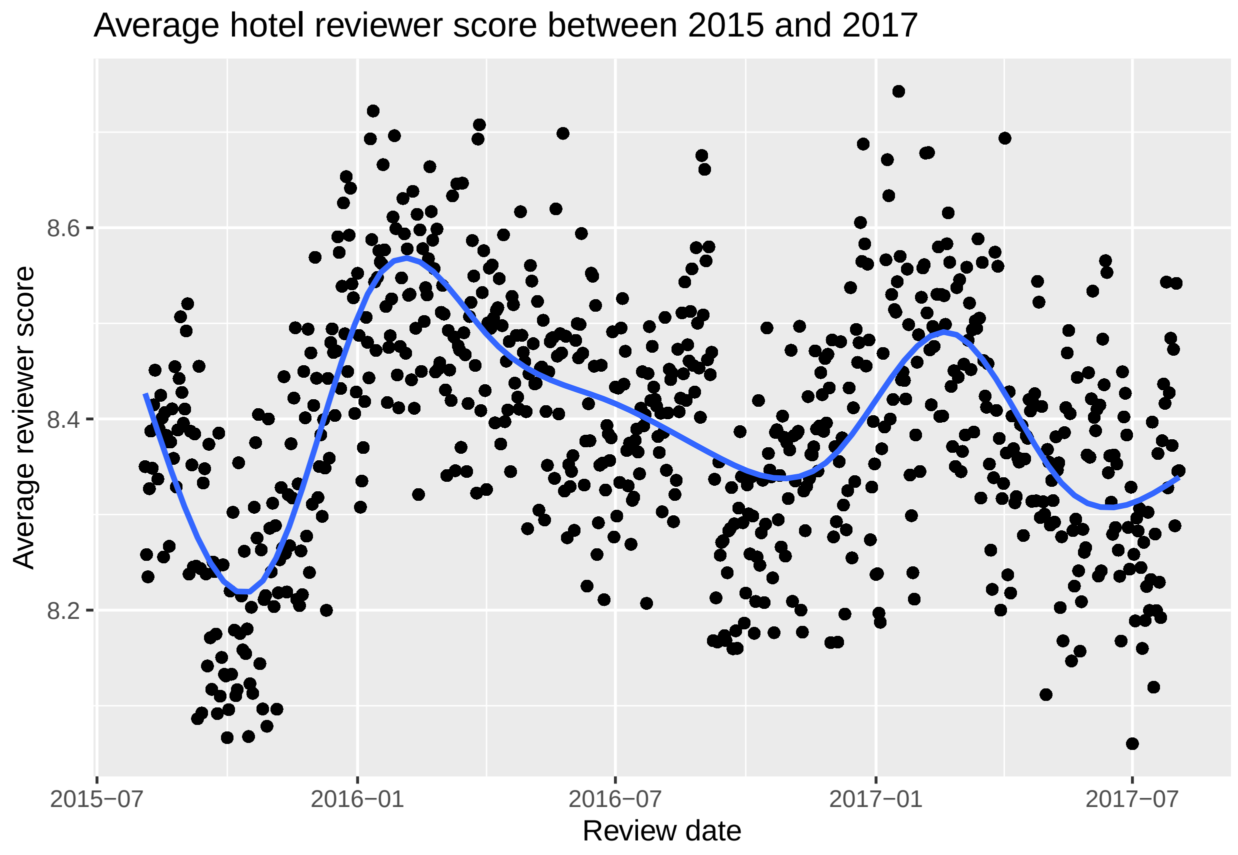
```r
train_set %>%
  group_by(Hotel_Name, Total_Number_of_Reviews)%>%
  summarize (avg_rating=mean(Reviewer_Score), n=n()) %>%
  arrange(desc(avg_rating))
```

```
## # A tibble: 1,494 x 4
## # Groups:   Hotel_Name [1,492]
##    Hotel_Name                          Total_Number_of_Revi~ avg_rating     n
##    <chr>                                               <int>      <dbl> <int>
##  1 Hotel Casa Camper                                     732       9.72   266
##  2 41                                                    244       9.71    93
##  3 Ritz Paris                                            122       9.70    24
##  4 H tel de La Tamise Esprit de France                   166       9.69    56
##  5 H10 Casa Mimosa 4 Sup                                 454       9.66   111
##  6 Le Narcisse Blanc Spa                                 222       9.65    51
##  7 Hotel The Serras                                      604       9.62   201
##  8 45 Park Lane Dorchester Collection                     68       9.60    26
##  9 Taj 51 Buckingham Gate Suites and Res~                310       9.60   118
## 10 Hotel Eiffel Blomet                                    79       9.59    13
## # ... with 1,484 more rows
```

The best and the worst hotels do not have many reviews. This is another argument to implement regularization to the recommendation system.

Now, I give interest to the reviewing period. I am going to see if the average reviewer score changes over the assessed reviewing period.

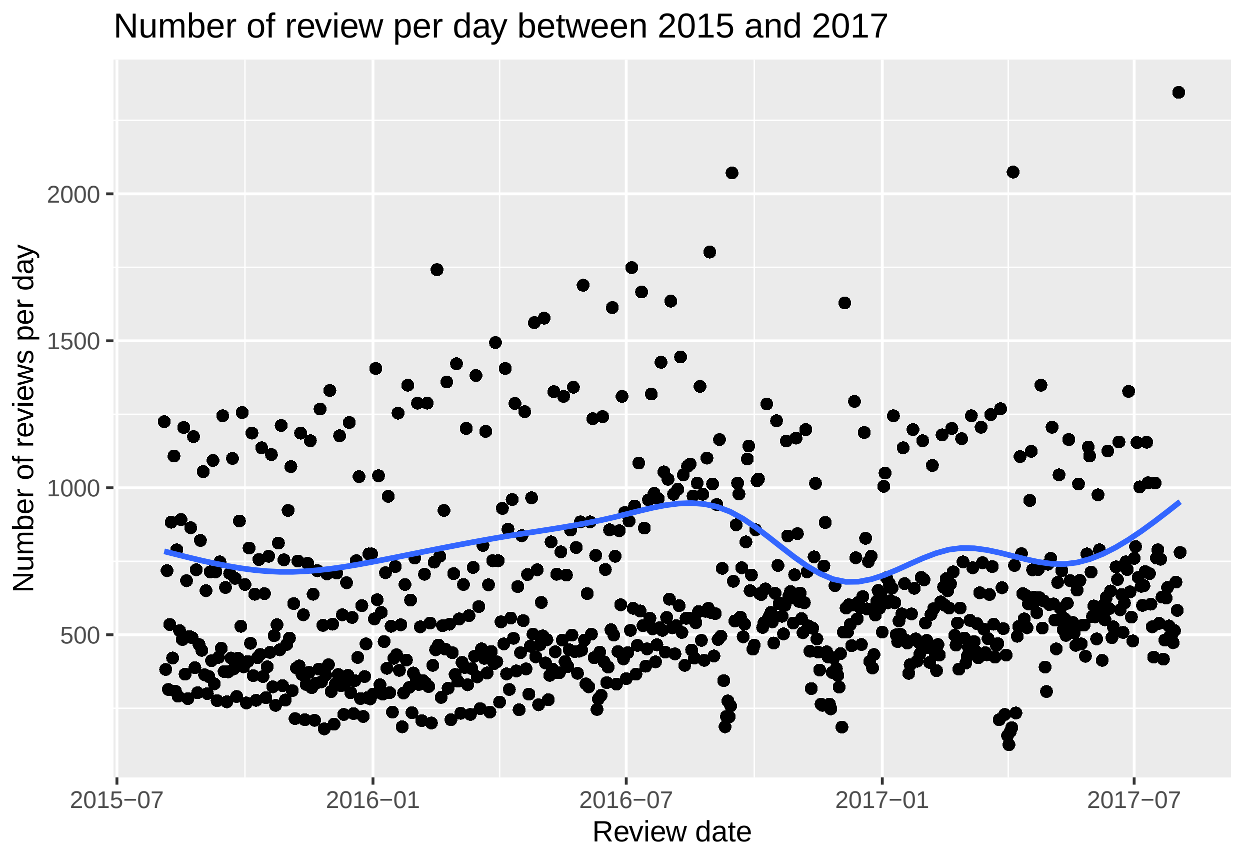
```
# To see the average reviewer score in function of time
train_set%>%
  group_by(Review_Date_correct)%>%
  summarize (avg_rating=mean(Reviewer_Score),Review_Date_correct)%>%
  ggplot(aes(Review_Date_correct, avg_rating))+geom_point()+
  stat_smooth()+ labs(x="Review date", y="Average reviewer score") +
  ggtitle("Average hotel reviewer score between 2015 and 2017")
```



From this graph, it appears that the average reviewer score is fluctuating across the year. It seems that the average rating during the winter period is better than during the summer period.

I am going to do the same thing but using the number of reviews to see if some periods provided more reviews than others did.

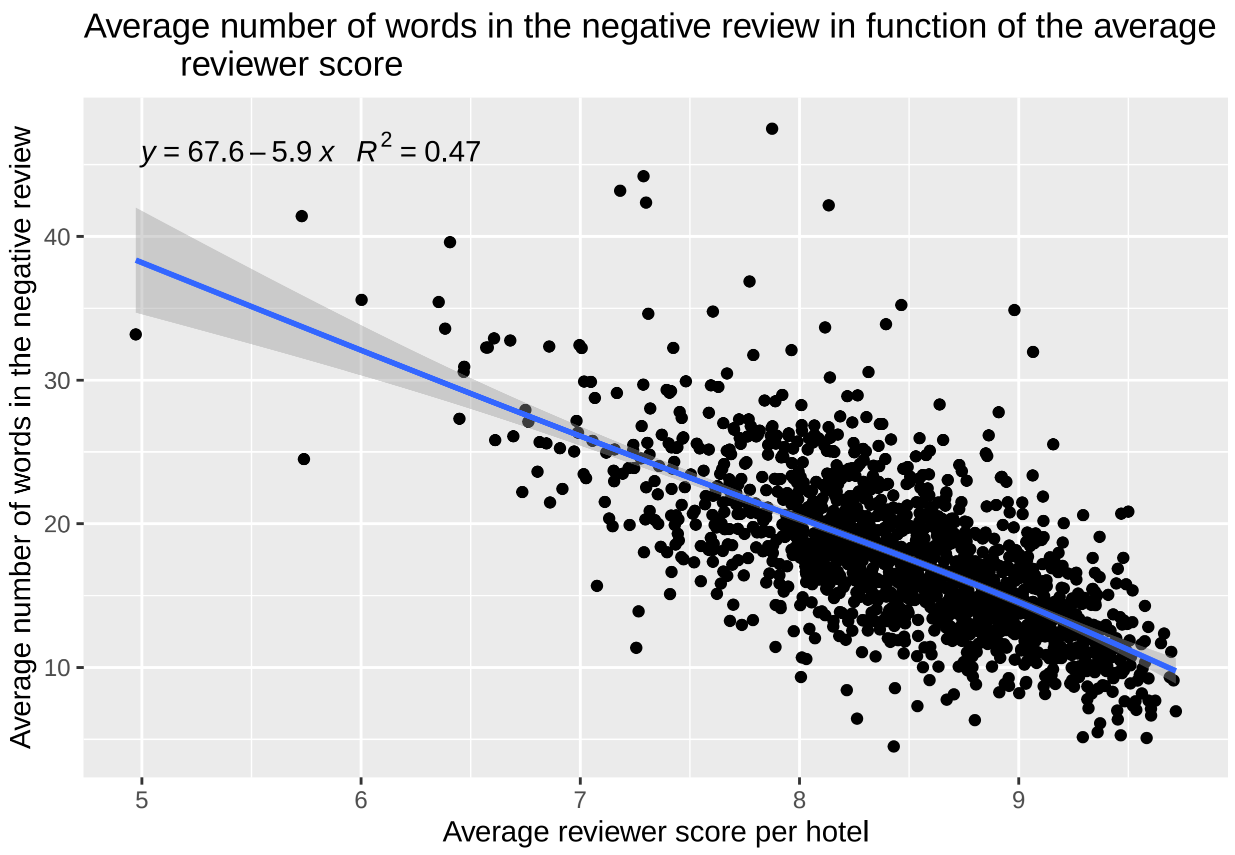
```
# To see the number of reviews in function of time
train_set%>%
  group_by(Review_Date_correct)%>%
  summarize (n=n(),Review_Date_correct)%>%
  ggplot(aes(Review_Date_correct, n))+geom_point()+
  stat_smooth()+ labs(x="Review date", y="Number of reviews per day") +
  ggtitle("Number of review per day between 2015 and 2017")
```



The average number of reviews over the two years analyzed is relatively stable with a small increase at the end of the summer periods. Therefore, the date of review may influence the accuracy of the recommendation system but its effect size is likely to be small.

Now, I am going to evaluate if there is a relationship between the average reviewer score and the number of words in the negative reviews.

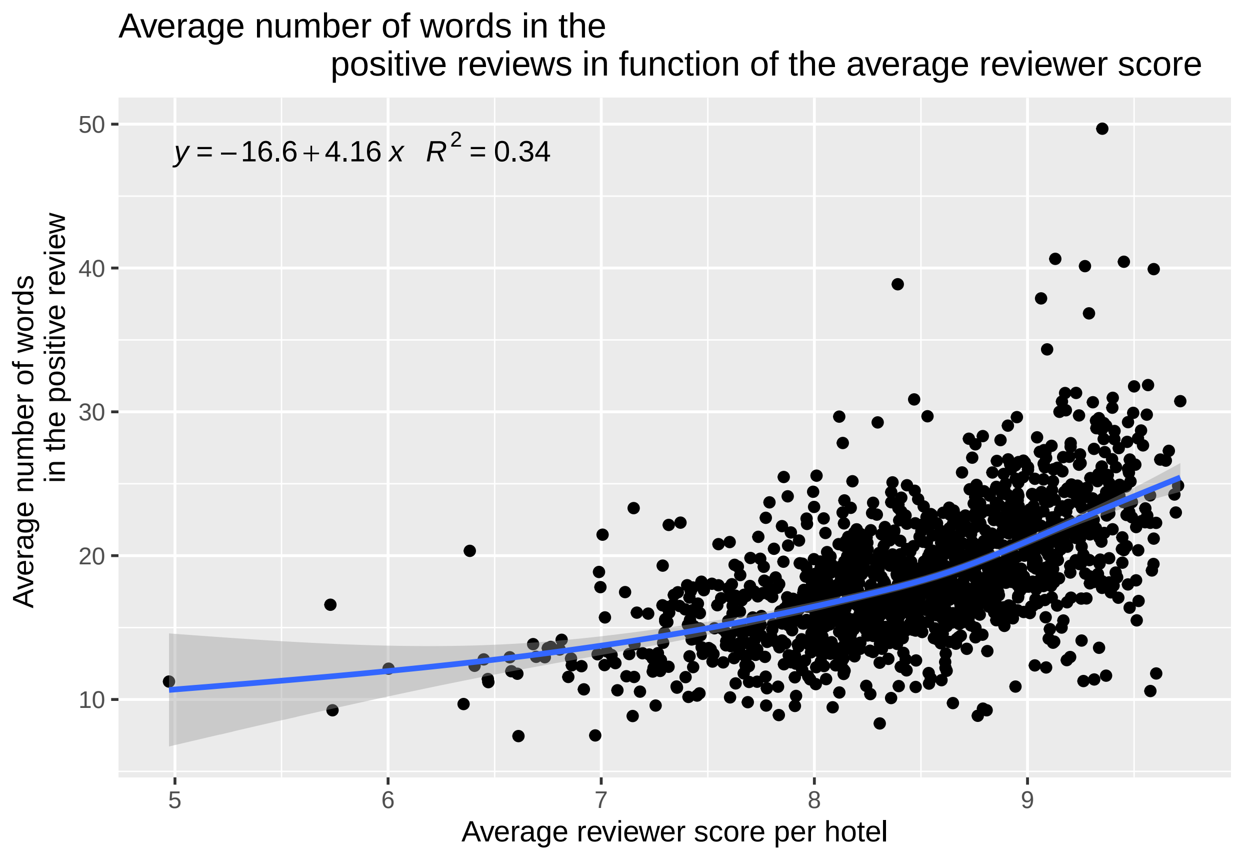
```r
# To add the equation of the regression curve to the graph
if(!require(ggpmisc)) install.packages("ggpmisc", repos = "http://cran.us.r-project.org")
library(ggpmisc)
my_formula <- y ~ x

# To see the average reviewer score in function of the number of words in the negative review
train_set %>%
  group_by(Hotel_Name)%>%
  summarize (avg_rating=mean(Reviewer_Score),
         avg_Review_Total_Negative_Word_Counts=mean(Review_Total_Negative_Word_Counts))%>%
  ggplot(aes(avg_rating, avg_Review_Total_Negative_Word_Counts)) +
  geom_point()+stat_smooth()+labs(x="Average reviewer score per hotel",
                                  y="Average number of words in the negative review")+
  ggtitle("Average number of words in the negative review in function of the average
          reviewer score") +
  stat_poly_eq(formula = my_formula, aes(label = paste(..eq.label.., ..rr.label..,
                                       sep = "~~~")), rr.digits = 2, parse = TRUE)
```



Logically, the graph displays a negative relationship with a large effect size (R^2=.47) between the average reviewer score and the number of words in the negative reviews. In other words, the more words there are in the negative review the worst will be the average score.

I am going to do the same thing but using the total number of positive words.

```
# To see the average reviewer score in function of the number of words in the positive reviews
train_set %>%
  group_by(Hotel_Name)%>%
  summarize (avg_rating=mean(Reviewer_Score),
        avg_Review_Total_Positive_Word_Counts=mean(Review_Total_Positive_Word_Counts))%>%
  ggplot(aes(avg_rating, avg_Review_Total_Positive_Word_Counts)) + geom_point()+
  stat_smooth()+ labs(x="Average reviewer score per hotel", y="Average number of words
                      in the positive review")+ ggtitle("Average number of words in the
                      positive reviews in function of the average reviewer score") +
  stat_poly_eq(formula = my_formula, aes(label = paste(..eq.label.., ..rr.label..,
                                           sep = "~~~")), rr.digits = 2, parse = TRUE)
```



The graph displays a large positive relationship between the two variables, also with a large effect size. Nevertheless, the effect size is not as strong (R^2=.34) as for the negative reviews.

I am now going to classify the reviewers per nationality and see if some nationalities have reported more reviews than others have. I am also going to compare the average reviewer score per nationality.

```
# To see a list presenting average reviewer score and number of reviews per reviewer nationality
# (ranged based on average reviewer score)
train_set %>%
  group_by(Reviewer_Nationality)%>%
  summarize (avg_rating=mean(Reviewer_Score), n=n())%>%
  arrange(desc(avg_rating))
```

```
## # A tibble: 227 x 3
##    Reviewer_Nationality              avg_rating     n
##    <chr>                                  <dbl> <int>
##  1 " Comoros "                            10        1
##  2 " Crimea "                             10        5
##  3 " Equatorial Guinea "                  10        1
##  4 " Niger "                              10        1
##  5 " Svalbard Jan Mayen "                 10        1
##  6 " Cape Verde "                          9.6      1
##  7 " Tajikistan "                          9.36    16
##  8 " Central Africa Republic "             9.3      3
##  9 " Saint Martin "                        9.28     4
## 10 " Bonaire St Eustatius and Saba "       9.2      2
## # ... with 217 more rows
```

```
# To see a list presenting average reviewer score and number of reviews per reviewer nationality
# (ranged based on number of reviews)
train_set %>%
  group_by(Reviewer_Nationality)%>%
  summarize (avg_rating=mean(Reviewer_Score), n=n())%>%
  arrange(desc(n))
```

```
## # A tibble: 227 x 3
##    Reviewer_Nationality         avg_rating      n
##    <chr>                             <dbl>  <int>
##  1 " United Kingdom "                 8.49 223451
##  2 " United States of America "       8.79  32434
##  3 " Australia "                      8.59  19668
##  4 " Ireland "                        8.46  13540
##  5 " United Arab Emirates "           7.88   9318
##  6 " Saudi Arabia "                   7.88   8117
##  7 " Netherlands "                    8.13   7978
##  8 " Switzerland "                    8.17   7851
##  9 " Germany "                        8.14   7209
## 10 " Canada "                         8.55   7196
## # ... with 217 more rows
```

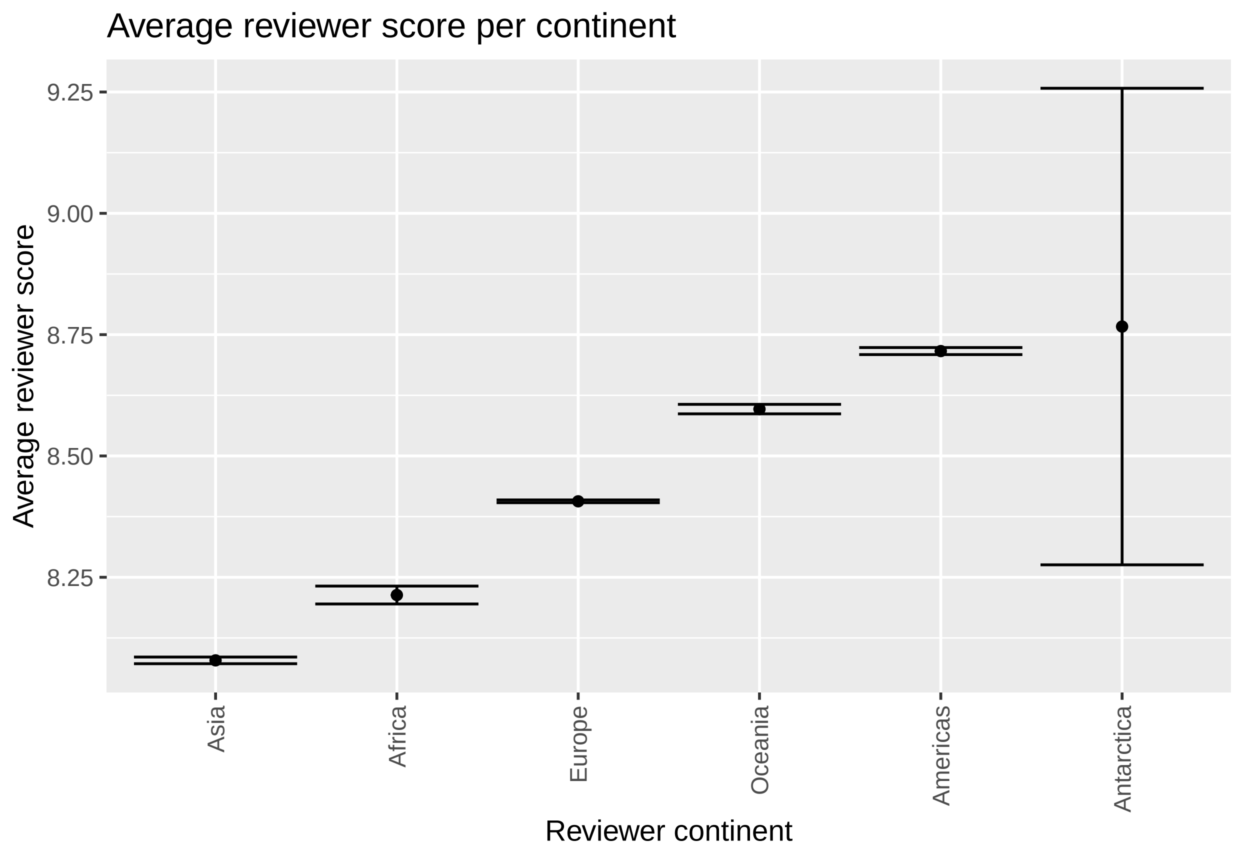
There are 227 different nationalities. The countries with the best reviewer scores also have just a few reviews. When considering the total number of reviews, United Kingdom is largely first, followed by the United States and Australia.

Just by curiosity, I am going to see if these differences can be summarized by grouping reviewers per continent of origin.

```
# To see a list presenting average reviewer score and number of reviews per reviewer continent
# (ranged based on average reviewer score)
train_set %>%
  group_by(Reviewer_continent)%>%
  summarize (avg_rating=mean(Reviewer_Score), n=n())%>%
  arrange(desc(avg_rating))
```

```
## # A tibble: 7 x 3
##   Reviewer_continent avg_rating      n
##   <chr>                   <dbl>  <int>
## 1 Antarctica               8.77      3
## 2 Americas                 8.72  44435
## 3 Oceania                  8.60  22679
## 4 Europe                   8.41 326827
## 5 Africa                   8.21   8136
## 6 Asia                     8.08  66769
## 7 <NA>                     8.00    471
```

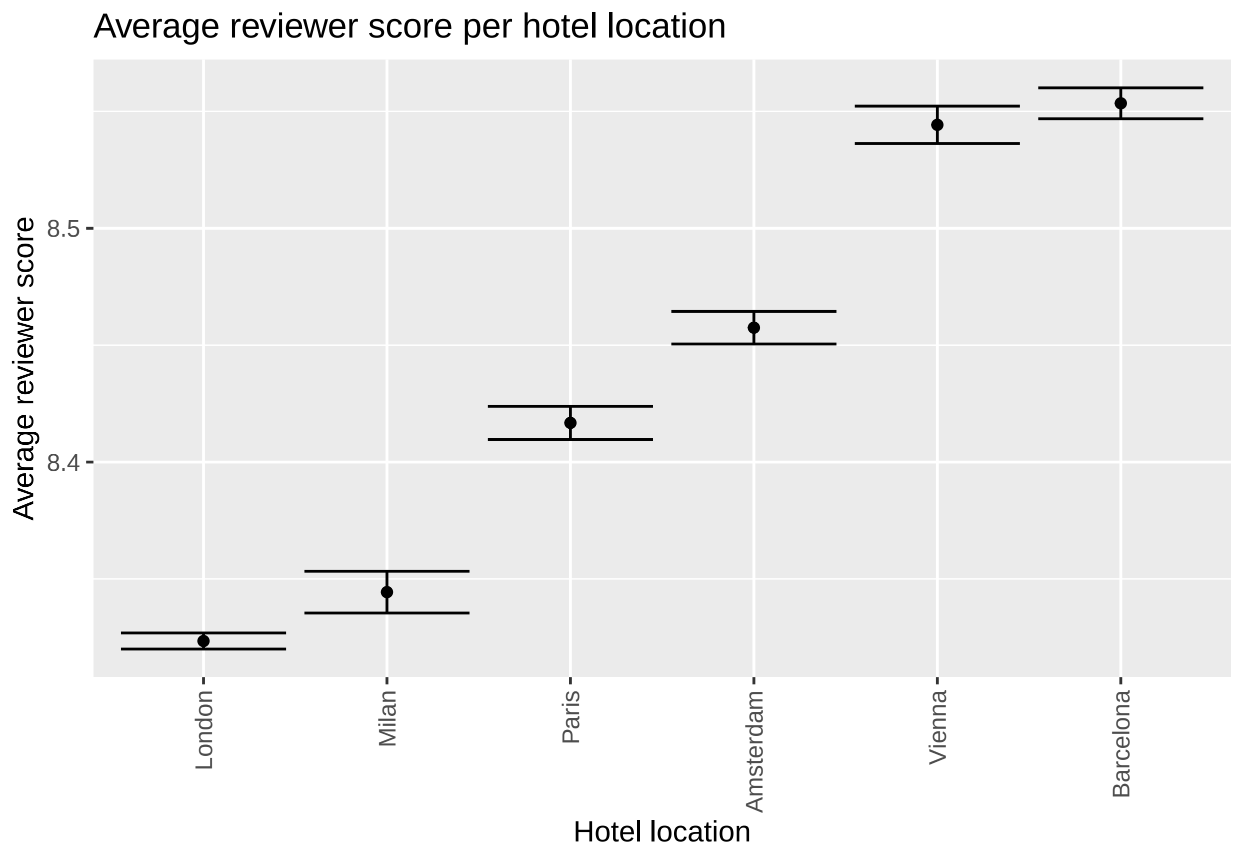
```
# To plot the average reviewer score per continent
plot1<-train_set%>%
  filter(Reviewer_continent!="") %>%
  group_by(Reviewer_continent)%>%
  summarize(count=n(), means=mean(Reviewer_Score), std=sd(Reviewer_Score)/sqrt(count)) %>%
  mutate(Reviewer_continent=reorder(Reviewer_continent, means))%>%
  ggplot(aes(x=Reviewer_continent, y=means, ymin=means-std, ymax=means+std ))+
  geom_point()+geom_errorbar() +
  labs(x="Reviewer continent", y="Average reviewer score")+
  ggtitle("Average reviewer score per continent")

plot1+theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



From the list, I can see that there are 471 missing values. Additionally, I see that Antarctica only has three reviews. This explains the very large standard error in the subsequent plot. The plot also shows that there is a clear difference in average reviewer scores between continents. Even though I think that the variable "reviewer_nationality" will provide more information to the recommendation system, it is interesting to see that differences between countries still exist when grouping the reviewer per continent. '

Finally, I know from earlier analyses that the hotels are only coming from six distinct cities. Thus, I am going to evaluate if the city itself influences the average reviewer score.

```
# To plot the average reviewer score per hotel location city
plot<-train_set%>%
  group_by(Hotel_city)%>%
  summarize(count=n(), means=mean(Reviewer_Score), std=sd(Reviewer_Score)/sqrt(count)) %>%
  mutate(Hotel_city=reorder(Hotel_city, means))%>%
  ggplot(aes(x=Hotel_city, y=means, ymin=means-std, ymax=means+std ))+
  geom_point()+geom_errorbar()+ labs(x="Hotel location", y="Average reviewer score")+
  ggtitle("Average reviewer score per hotel location")

plot+theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



There is a clear difference between cities. The hotels from Vienna and Barcelona have the best average reviewer score, while the hotels from London and Milan have the worst.

2.3 Brief description of the machine-learning techniques used in this report

2.3.1 Root Mean Squared Error (RMSE)

The RMSE is the accuracy metric that I am using in this report. It can be defined by the following function [1]:

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

This function is an evaluation of how spread are the residuals of a specific model, and thus, is a measure of fit of the model [1]. As the RMSE is on the same scale as the predicted variable [1], there are no standardized guidelines to evaluate its precision. However, all definition agree on the fact that the smaller the RMSE is, the better is the fit of the model. Therefore, I will create different models, select the one with the smallest RMSE during the training phase and test it on the validation set.

2.3.2 Regularization

Description The motivation to use regularization is based on the large errors made on specific predictions trained with small sample sizes (e.g., in the MovieLens the best and the worst-rated movies only had a few ratings, and thus the rating predictions for these specific movies were of mediocre quality) [1]. These errors increase the RMSE (counterproductive). Regularization permits penalizing these large estimates and thus constrain the total variability of the estimates [1].

Penalized least square I will penalize the large estimates with a specific lambda term selected based on each computed model. Lambda is a penalty used to constrain the variability of the estimates [1]. The bigger it is, the more shrunken the large estimates will be [1]. To choose the proper lambda for each model, I will use cross-validation. In other words, I will test several lambdas (e.g., lambdas<-seq(0,20,0.25)) and select the one giving the smallest RMSE.

2.3.3 Predictors

The predictors put in evidence in the data exploration step are the following:

- Number of reviews per hotel
- Number of reviews a reviewer has given
- Reviewer nationality
- City in which the hotel is located
- Review date

The variables "Review_Total_Negative_Word_Counts", and "Review_Total_Positive_Word_Counts" cannot be included in the recommendation system because they are part of the reviewing process. Indeed, they are realized by the customers after their stay in a hotel, during the rating process where the score is emitted.

2.3.4 Principal Component Analysis (PCA)

PCA is a technique more and more used in multiple disciplines (e.g., electroencephalography, structural equation modelling, recommendation systems) to simplify and isolate the sources of variability of a data set. In electroencephalography analysis, for example, principal component analysis can be used to isolate and remove artefacts (e.g., eye blinks, cardio-ballistic artefacts) from the data set [5]. PCA is based on singular value decomposition that allows simplifying the dimensions of variability between variables, and thus with just a few components, it is possible to explain a large part of its variability.

Theoretically, a PCA based on Hotel name and user ID would provide the best outcome. But as the user ID is not available, this is not possible. I could replace the user ID with another predictor to construct the scoring matrix. Nonetheless, all the detected predictors have multiple reviewer scores for each level forcing the creation of a matrix with three dimensions. Therefore, I am going to chose a simpler option based on multiple predictors. Such PCA is, however, not suitable for categorical predictors (i.e., not accurate and very long computation time) [6]. Thus, I am only going to use the numeric predictors detected in the data exploration step (i.e., review date, the total number of reviews the reviewer has given, and the total number of review per hotel). To perform such analysis, I am going to apply the concept of PCA to regression. Additionally, I will use k_fold cross folding validation to have more robust results. This analysis is based on the book of Prof. Raphael Irizarry (last update 2021) [1] and on the tutorial of Zeng and colleagues (2017) [7].

Finally, to have an objective comparison between the accuracy of PCA and regression-based recommendation systems, I am going to do a second regression-based model only using the three numeric predictors. I will, then, compare the RMSE of both techniques.

3. Results

3.1 Training the model

To begin, I am going to create a naive model that I will use as a reference. In this model, I assume that all hotels are rated the same and that the random error can explain all the variance in the results. Thus, the only predictor in this model is the average reviewer score from the training set.

```r
# Model 1 (m1) - reference rmse (naive)
mean_RS<-mean(train_set$Reviewer_Score)
ref_rmse<-RMSE(test_set$Reviewer_Score, mean_RS)
rmse_results <-data_frame(method="Just the average reviewer score", RMSE=ref_rmse)
```

The RMSE reference value is 1.630. As displayed in the data exploration section, this naive model does not properly consider the total reviewer score variance. Indeed, I have displayed potential predictors that could be used to give a better fit to my model.

As displayed in the data exploration section, some hotels were rated higher than others. Therefore, I am going to add a bias term (i.e., effect in statistical textbooks [1]) for hotels (b_h) to the recommendation system.

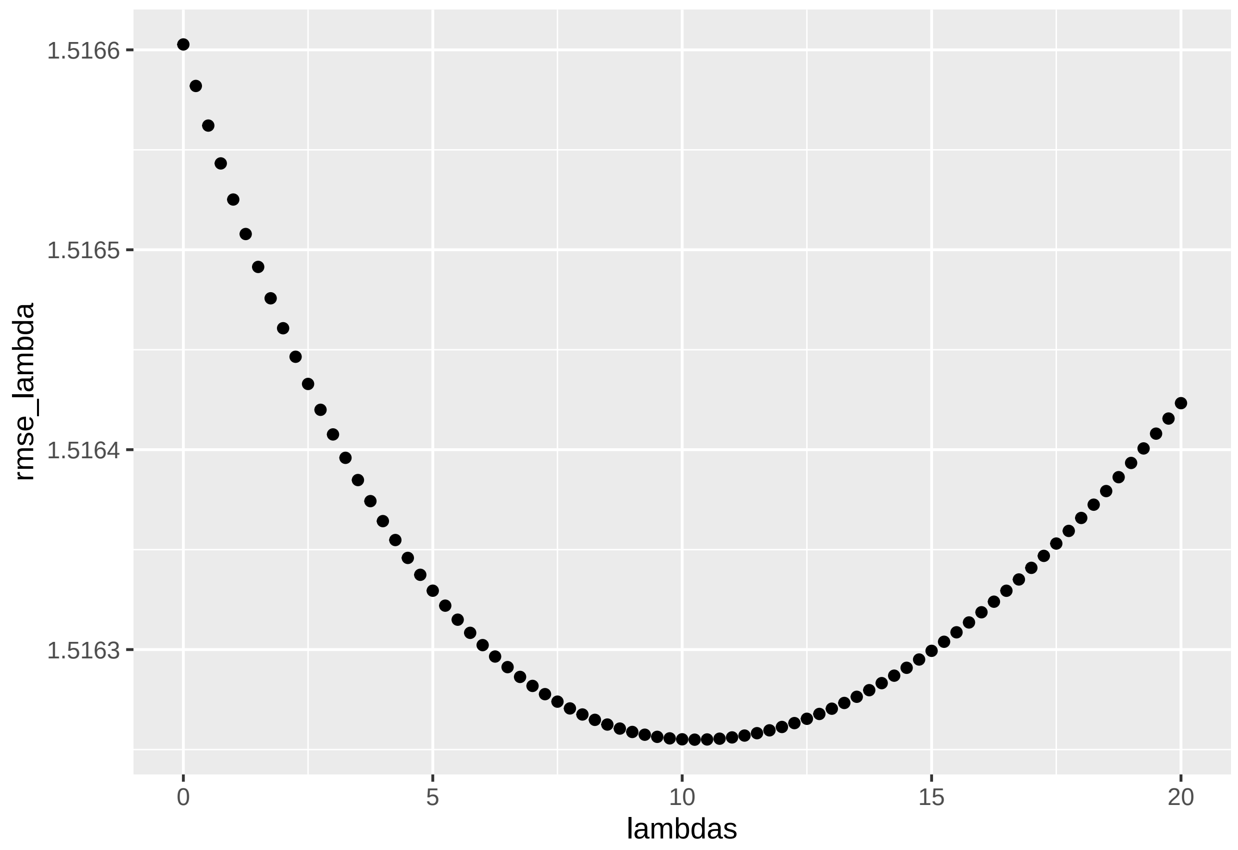
```r
# Model 2 (m2) - average rating + hotel bias (b_h)
b_h<-train_set%>%
  group_by(Hotel_Name)%>%
  summarize(b_h=mean(Reviewer_Score - mean_RS))

predicted_scores<-mean_RS+test_set%>%
  left_join(b_h, by="Hotel_Name")%>%
  .$b_h

rmse_2<-RMSE(test_set$Reviewer_Score, predicted_scores)
rmse_results<-data_frame(method=c("Just the average reviewer score", "m1 + Hotel bias"),
                         RMSE=c(ref_rmse, rmse_2))
format.data.frame(rmse_results, digits=6)
```

```
##                            method    RMSE
## 1 Just the average reviewer score 1.63011
## 2                 m1 + Hotel bias 1.51660
```

The RMSE has largely decreased (RMSE m2= 1.517).

As demonstrated in the data exploration step, the hotels with the highest and the lowed average reviewer scores also had the lowest number of reviews. Therefore, I am going to use the regularization technique over the recommendation system (see section 2.3.2). I am, first, going to create a function to determine the lambda term (see section 2.3.3) giving the lowest RMSE and then apply it to the current model.

```r
# Model 3 (m3) - regularization ==> hotel bias (b_h)
lambdas <- seq(0, 20, 0.25)#Will be used in all the subsequent models

rmse_lambda <- sapply(lambdas, function(l){
  mean_RS<-mean(train_set$Reviewer_Score)

  b_h <- train_set %>%
  group_by(Hotel_Name) %>%
  summarize(b_h = sum(Reviewer_Score - mean_RS)/(n()+l))
  predicted_scores <- test_set %>%
    left_join(b_h, by = "Hotel_Name") %>%
    mutate(pred = mean_RS + b_h) %>%
    pull(pred)
  return(RMSE(test_set$Reviewer_Score, predicted_scores))
})

# To plot rmse against lambda
qplot(lambdas, rmse_lambda)
```



```
# To determine what lambda corresponds to the smallest RMSE
lambda<-lambdas[which.min(rmse_lambda)]

# RMSE with the selected lambda
b_h <- train_set %>%
  group_by(Hotel_Name) %>%
  summarize(b_h = sum(Reviewer_Score - mean_RS)/(n()+lambda))

predicted_scores <- test_set %>%
  left_join(b_h, by="Hotel_Name") %>%
  mutate(pred = mean_RS + b_h) %>%
  .$pred

rmse_3 <- RMSE(test_set$Reviewer_Score, predicted_scores)
rmse_results <- data_frame(method = c("Just the average rating", "m1 + Hotel bias",
                                      "m2 + Regularization"),
                           RMSE = c(ref_rmse, rmse_2, rmse_3))

format.data.frame(rmse_results, digit=6)
```

```
##                     method    RMSE
## 1 Just the average rating 1.63011
## 2          m1 + Hotel bias 1.51660
## 3     m2 + Regularization 1.51625
```

Although the RMSE is slightly smaller than in model 2 (m2), the hotel bias did not benefit from the regularization as much as expected. As there is no user ID in the data set, I am going to continue by adding the variable “Total_Number_of_Reviews_Reviewer_Has_Given”.

Note: To avoid having pages of codes in the report and as the algorithm is very repetitive, I am going to stop printing the script in the report for upcoming models. Nevertheless, I will print the final model in the validation step. Furthermore, as required, the full script is reported in the R and Rmd files.

```
##                             method    RMSE
## 1 Just the average reviewer score 1.63011
## 2                 m1 + Hotel bias 1.51660
## 3               m2+ Regularization 1.51625
## 4               m3 + Reviewer bias 1.51608
```

The RMSE went once again very slightly down (RMSE m4 = 1.516). This was, however, expected as the variable “Total_Number_of_Reviews_Reviewer_Has_Given” provides only a small portion of the variance user ID would have provided.

I am going to add an error term for the date of the review. As shown in the data exploration step, this predictor is likely to have only a small effect.

```
##                             method    RMSE
## 1 Just the average reviewer score 1.63011
## 2                 m1 + Hotel bias 1.51660
## 3            m2 +   Regularization 1.51625
## 4               m3 + Reviewer bias 1.51608
## 5                   m4 + Date bias 1.51366
```

The RMSE continued going down (RMSE m5=1.514). Unexpectedly, even though small, the effect size for the date of the review was bigger than for the two previous models.

I am going to continue by adding the city in which the hotel is located as a predictor.

```
##                             method    RMSE
## 1 Just the average reviewer score 1.63011
## 2                 m1 + Hotel bias 1.51660
## 3               m2+ Regularization 1.51625
## 4               m3 + Reviewer bias 1.51608
## 5                   m4 + Date bias 1.51366
## 6                   m5 + City bias 1.51358
```

The RMSE very slightly decreased (RMSE m6= 1.514).

In the data exploration step, I have assessed the difference in the averaged reviewer score per reviewer nationality and reviewer continent. As generalizing reviewer nationality into a more constraint variable (i.e., reviewer continent) is also losing information on the variance of specific groups of reviewers, I expect that the reviewer nationality will be a better predictor than reviewer continent. Nevertheless, for demonstrative purposes, I am going to test both. I begin with the reviewer continent.

```
##                              method    RMSE
## 1 Just the average reviewer score 1.63011
## 2                  m1 + Hotel bias 1.51660
## 3              m2 + Regularization 1.51625
## 4               m3 + Reviewer bias 1.51608
## 5                   m4 + Date bias 1.51366
## 6                   m5 + City bias 1.51358
## 7       m6 + rev. Continent error 1.50754
```

Now, I am going to replace the predictor "Reviewer_continent" by "Reviewer_Nationality".

```
##                              method    RMSE
## 1 Just the average reviewer score 1.63011
## 2                  m1 + Hotel bias 1.51660
## 3              m2 + Regularization 1.51625
## 4               m3 + Reviewer bias 1.51608
## 5                   m4 + Date bias 1.51366
## 6                   m5 + City bias 1.51358
## 7        m6 + rev. Continent bias 1.50754
## 8      m6 + rev. Nationality bias 1.50093
```

As expected, using reviewer nationality is better than using reviewer continent.
Model 7 has a RMSE of 1.511. It is the best model produced in this training step and therefore will be tested using the validation set. The lambda term from this model has been saved under the name **"lambda_val1"**.

3.2 Testing the model

First, I have to adapt the predictors from the validation set (as for the two other sets).

```r
# To adapt the date format
validation$Review_Date_correct<-mdy(validation$Review_Date)
validation$Review_Date_correct<-as_date(validation$Review_Date_correct)

# To get city from each hotel
validation$Hotel_Address<-as.character(validation$Hotel_Address)
addresses_validation<-strsplit(validation$Hotel_Address, " ")

# Match on country in world cities
if(!require(plyr)) install.packages("plyr", repos = "http://cran.us.r-project.org")
library(plyr)
validation$Hotel_city<- llply(addresses_validation,
                              function(x)x[max(which(x %in% world.cities$name))])
validation$Hotel_city<-as.character(validation$Hotel_city)
detach("package:plyr", unload=TRUE)
```

Now, I am going to test my model. The lambda term has been saved from the previous section (i.e., "lambda_val1), and therefore do not need to be re-computed.

```
# Model FINAL - regularization ==> hotel + date + n reviews per reviewer + hotel city
# + reviewer nationality

b_h <- train_set %>%
  group_by(Hotel_Name) %>%
  summarize(b_h = sum(Reviewer_Score - mean_RS)/(n()+lambda_val1))

b_r <-train_set %>%
  left_join(b_h, by="Hotel_Name") %>%
  group_by(Total_Number_of_Reviews_Reviewer_Has_Given) %>%
  summarize(b_r = sum(Reviewer_Score - b_h - mean_RS)/(n()+lambda_val1))

b_d <-train_set %>%
  left_join(b_h, by="Hotel_Name") %>%
  left_join(b_r, by = "Total_Number_of_Reviews_Reviewer_Has_Given") %>%
  group_by(Review_Date_correct) %>%
  summarize(b_d = sum(Reviewer_Score - b_h - b_r - mean_RS)/(n()+lambda_val1))

b_c <-train_set %>%
  left_join(b_h, by="Hotel_Name") %>%
  left_join(b_r, by = "Total_Number_of_Reviews_Reviewer_Has_Given") %>%
  left_join(b_d, by = "Review_Date_correct") %>%
  group_by(Hotel_city) %>%
  summarize(b_c = sum(Reviewer_Score - b_h - b_r -b_d - mean_RS)/(n()+lambda_val1))

b_r_n <-train_set %>%
  left_join(b_h, by="Hotel_Name") %>%
  left_join(b_r, by = "Total_Number_of_Reviews_Reviewer_Has_Given") %>%
  left_join(b_d, by = "Review_Date_correct") %>%
  left_join(b_c, by = "Hotel_city") %>%
  group_by(Reviewer_Nationality) %>%
  summarize(b_r_n = sum(Reviewer_Score - b_h - b_r -b_d - b_c - mean_RS)/(n()+lambda_val1))

predicted_scores <- validation %>%
  left_join(b_h, by="Hotel_Name") %>%
  left_join(b_r, by = "Total_Number_of_Reviews_Reviewer_Has_Given") %>%
  left_join(b_d, by = "Review_Date_correct") %>%
  left_join(b_c, by = "Hotel_city") %>%
  left_join(b_r_n, by = "Reviewer_Nationality") %>%
  mutate(pred = mean_RS + b_h + b_r + b_d + b_c + b_r_n) %>%
  .$pred

VALIDATION_RMSE <- RMSE(predicted_scores, validation$Reviewer_Score)
rmse_results <- data_frame(method = c("Just the average reviewer score", "m1 + Hotel bias",
                                      "m2 + Regularization",
                                      "m3 + Reviewer bias", "m4 + Date bias",
                                      "m5 + City bias",
                                      "m6 + rev. Continent bias",
                                      "m6 + rev. Nationality bias",
                                      "VALIDATION"),
                           RMSE = c(ref_rmse, rmse_2, rmse_3, rmse_4, rmse_5, rmse_6,
                                    rmse_7, rmse_8, VALIDATION_RMSE))


format.data.frame(rmse_results, digit=6)
```

```
##                             method    RMSE
## 1 Just the average reviewer score 1.63011
## 2                 m1 + Hotel bias 1.51660
## 3             m2 + Regularization 1.51625
## 4              m3 + Reviewer bias 1.51608
## 5                  m4 + Date bias 1.51366
## 6                  m5 + City bias 1.51358
## 7        m6 + rev. Continent bias 1.50754
## 8      m6 + rev. Nationality bias 1.50093
## 9                      VALIDATION 1.50974
```

The validation RMSE is 1.510. It is significantly better than the naive model. However, there is still room for improvements. In the next section to apply another machine learning technique to try to enhance the accuracy of this recommendation system.

3.3 Comparison of PCA and regression-based recommendation model (demonstrative purpose)

3.3.1 Regression-based recommendation model

In this section, I am going to test the effect of PCA in a recommendation system and compare its results to an equivalent regression-based system. As described in section 2.3.4, I am going to include only the numerical predictors (i.e., Total number of reviews a reviewer has given, review date and the total number of review per hotel) in the PCA. The number of reviews per hotel was covered in the previous recommendation system by the hotel bias and regularization however as do not want to include categorical variables in the PCA, I am going to use the variable "Total_Number_of_Reviews" already existing in the data set. Additionally, to ease the PCA computation I also going to use another format for the reviewing date. As one variable equivalent variable directly reporting the number of days since the review already exists in the data set, I am just going to switch to this variable. However, this variable also has the words "day" or "days" after each number in all cells. Thus, I will have to do a small pre-processing step to isolate the number.

```
# To split the column "days_since_review" based on the space between the number and the word
# "day"
train_set <- train_set%>%
  extract(days_since_review, c("n_days", "rest"), regex = "^(\\S+)\\s+(.*)", convert = TRUE)
test_set <- test_set%>%
  extract(days_since_review, c("n_days", "rest"), regex = "^(\\S+)\\s+(.*)", convert = TRUE)
validation <- validation %>%
  extract(days_since_review, c("n_days", "rest"), regex = "^(\\S+)\\s+(.*)", convert = TRUE)
```

Now that everything is ready, I am going to first create the regression-based recommendation system using only the numeric predictors.

Note: As this part of the script is very similar to the previous section, I have chosen to do not print it in the pdf document. It is, however, available in the Rmd and R files.

```
##                             method    RMSE
## 1 Just the average reviewer score 1.63011
## 2    m1 - 2nd sys + Reviewer bias 1.62993
## 3        m2 - 2nd sys + Date bias 1.62703
## 4      m3 - 2nd sys + Review bias 1.52544
## 5                      VALIDATION 1.53550
```

The validation RMSE is 1.536 and the number of reviews is the predictor with the best effect size. I am now going to build the equivalent PCA recommendation system.

As there is no "tuning" of the algorithm, I only need two sets; the training set and the validation set. So first, to avoid losing power in not using the test set, I am going to merge it with the training set (i.e., train_set). Furthermore, the split 90%/10% is perfectly adapted to maximize the accuracy of the training and leaving enough data to mimic real-world results' variance.

```r
# To merge the training and the test sets
train_set<-rbind(train_set, test_set)
```

Second, I create the correlation matrix to assess if my predictors are highly correlated.

```r
# Required libraries for the PCA
if(!require(chillR)) install.packages("chillR", repos = "http://cran.us.r-project.org")
if(!require(factoextra)) install.packages("factoextra", repos = "http://cran.us.r-project.org")
if(!require(pls)) install.packages("pls", repos = "http://cran.us.r-project.org")
library(chillR)
library(factoextra)
library(pls)

# To create a data set with the variables of interest
train_PCA<-train_set%>%
  select (Total_Number_of_Reviews_Reviewer_Has_Given, n_days, Total_Number_of_Reviews,
          Reviewer_Score)

# To remove the variable that is going to be predicted
predictor_PCA<- train_PCA[,-4]

# To change the names that are a bit long for the correlation matrix
colnames(predictor_PCA)<- c("n rev. per Reviewer", "days since review", "n Reviews" )
# To do the correlation matrix
cor(predictor_PCA)
```

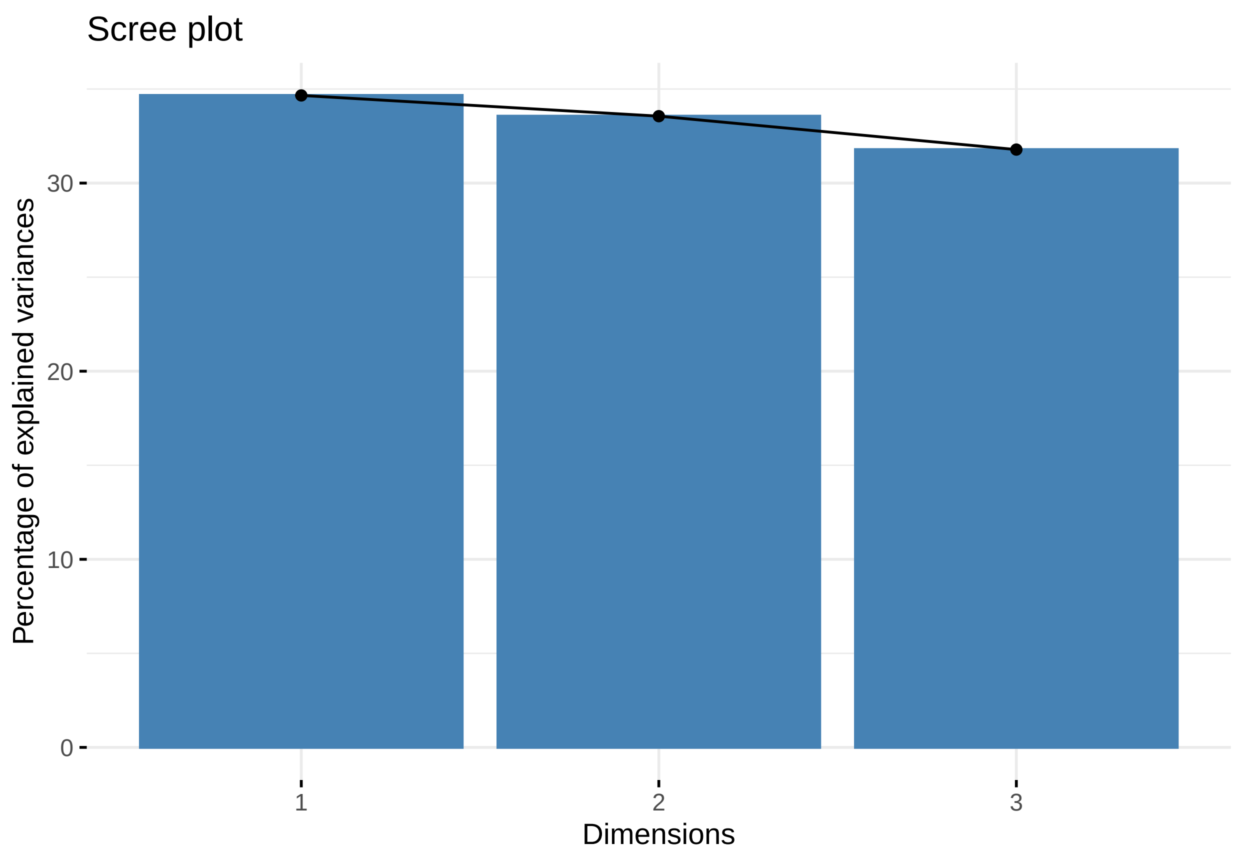
```
##                     n rev. per Reviewer days since review    n Reviews
## n rev. per Reviewer          1.00000000       0.035021198 -0.024911134
## days since review            0.03502120       1.000000000  0.007157442
## n Reviews                   -0.02491113       0.007157442  1.000000000
```

That's excellent. The predictors are independent and do not correlate at all.
Third, I going to analyse the principal variance components.

```r
# PCA
rec_pca<-prcomp(predictor_PCA, center=TRUE, scale. =TRUE)

# To summarize the results
summary(rec_pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3
## Standard deviation     1.0197 1.0034 0.9764
## Proportion of Variance 0.3466 0.3356 0.3178
## Cumulative Proportion  0.3466 0.6822 1.0000
```

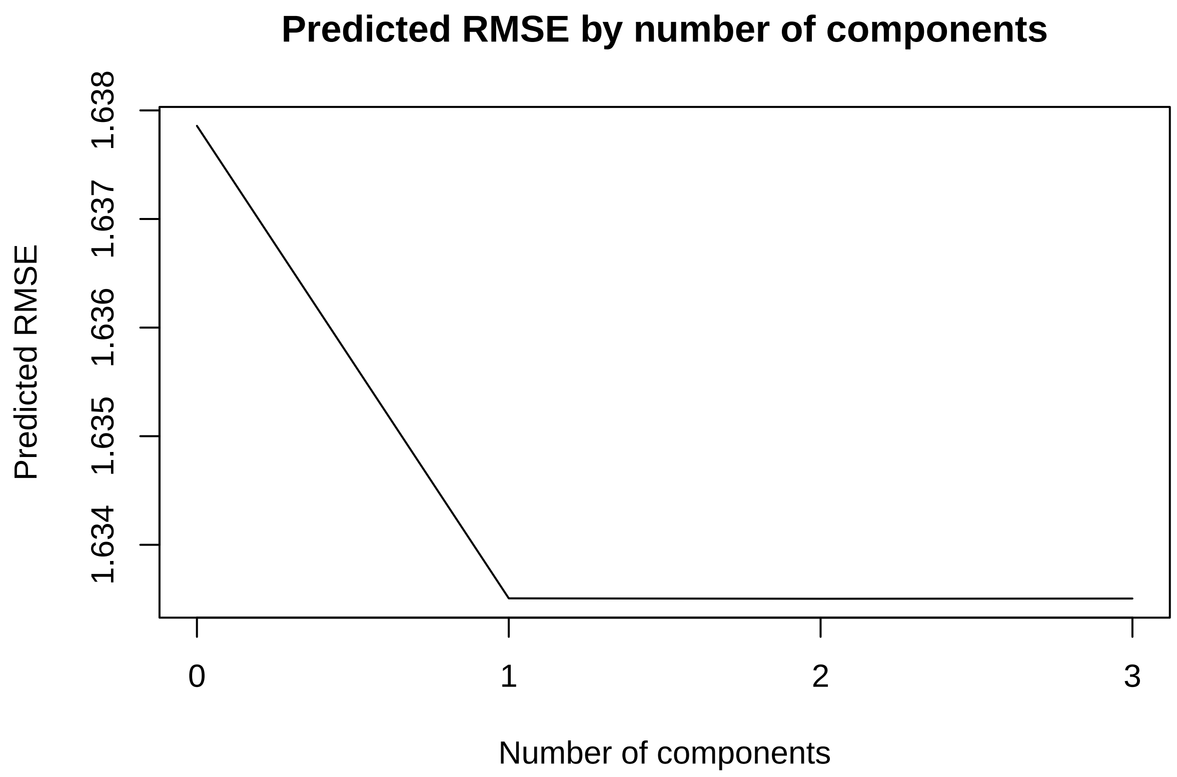
```
# To do a scree plot with the different components
fviz_eig(rec_pca)
```



The scree plot displays that the three components each explain approximately 30% of the total results' variance.

Fourth, I am not going to apply the concept of PCA on a regression. To do so, I am going to use 10-fold cross-validation to be sure to have robust results.

```
# Regression PCA
pcr_fit <- pcr(Reviewer_Score ~ Total_Number_of_Reviews_Reviewer_Has_Given + n_days +
                 Total_Number_of_Reviews, data = train_PCA, segments=10, validation="CV")

rmse_CV <- RMSEP(pcr_fit, estimate="CV")

# Scree plot
plot(rmse_CV, xaxt="n", main="Predicted RMSE by number of components",
     xlab="Number of components", ylab="Predicted RMSE")

# To customize the axis
axis(1, at=c(0,1,2,3), labels = c(0,1,2,3))
```



```r
# To see how ,any components are required to have the smallest RMSE
n_comp <- which.min(rmse_CV$val)
```

The results show that using three components would provide the best RMSE. Nevertheless, the graph shows that the first component is the most important. It also appear that the RMSE will not be good (close to the reference/naive RMSE).
Finally, I am going to test this approach on the validation set.

```r
fit <- predict(pcr_fit, validation, ncomp=(as.numeric(n_comp)-1))
rmse_PCA <- RMSE(validation$Reviewer_Score, fit)
rmse_results<-data_frame(method=c("Just the average reviewer score", "m1 - 2nd sys + Reviewer bias",
                                  "m2 - 2nd sys + Date bias", "m3 - 2nd sys + Review bias",
                                  "VALIDATION", "PCA"),
                         RMSE=c(ref_rmse, rmse_2, rmse_3, rmse_4, rmse_5, rmse_PCA ))
format.data.frame(rmse_results, digits=6)
```

```
##                             method    RMSE
## 1 Just the average reviewer score 1.63011
## 2    m1 - 2nd sys + Reviewer bias 1.62993
## 3        m2 - 2nd sys + Date bias 1.62703
## 4      m3 - 2nd sys + Review bias 1.52544
## 5                      VALIDATION 1.53550
## 6                             PCA 1.63543
```

The table displays that with this data set and the selected predictors, the regression-based recommendation system provides the best RMSE, and therefore can be considered as the most

accurate. The RMSE for the PCA (1.635) is even higher than the one from the naive model that only used the average reviewer score. PCA was not a suitable approach for this data set and predictors.

4. Conclusion

This manuscript reports the creation of a recommendation system using different machine learning techniques (i.e., regressions-based, regularization, and PCA). The reference RMSE value in the naive model was 1.630. When including all potential predictors in the approach combining regression-based and regularization techniques, **the best RMSE was 1.510**. The most important predictor was the hotel bias. In a second step, to answer the requirement of the assignment I did compare a PCA recommendation system (including only numeric predictors) and its regression-based equivalent. The RMSE for the PCA was 1.635 while it was 1.536 for the regression-based approach. **Therefore, in this data set, the regression-based approach was the most suitable to obtain the smallest RMSE.** When combined with regularization its results were slightly better.

Nevertheless, several limitations are to be mentioned. Indeed, the data set used in this report had missing information (i.e., no user ID) that have affected the results, especially in the PCA and in the regularization. The absence of user ID pushed me to use a predictors-based approach of PCA. Indeed without user ID, a two dimensions score matrix was impossible to construct. Replacing user ID by another predictor (e.g., Total number of reviews reviewer has given) would have led to a three dimensions matrix (multiple reviewer scores for each level of all predictors and each Hotel). **Therefore, it cannot be concluded that a regression-based approach is better than PCA overall**. It is very likely that with the proper variables, the PCA would have obtained better results than the ones reported in this report.

Future works might consider re-doing this comparison in a complete data set. These results might even be developed using an artificial neural network like restricted Boltzmann machines.

5. References

[1] https://rafalab.github.io/dsbook/
[2] https://learning.edx.org/course/course-v1:HarvardX+PH125.9x+1T2021/home
[3] https://www.kaggle.com/jiashenliu/515k-hotel-reviews-data-in-europe
[4] https://www.dropbox.com/s/pehhztgk97ftvsu/Hotel.zip?dl=0
[5] Subasi A, Gursoy MI. EEG signal classification using PCA, ICA, LDA and support vector machines. Expert Syst Appl. 2010;37:8659–8666.
[6] https://towardsdatascience.com/pca-is-not-feature-selection-3344fb764ae6
[7] https://jbhender.github.io/Stats506/F17/Projects/G18.html