GOOGLE PLAYSTORE

annalee0107

2022/2/18

Introduction to this project:

This is a project for the for the capstone project of HardvardX's data science course. The project used R codes and Google PlayStore dataset to analyse the Ratings on Applications released on Google Store for download and instalation in Android mobile phones.

R Packages needed for the project:

The packages required for the project included :

tidyverse,

caret,

data.table,

lubridate,

ggplot2,

formatR

The packages are instaled and load into the library:

```
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(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
if (!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if (!require(formatR)) install.packages("formatR", repos = "http://cran.us.r-project.org")


library(tidyverse)
library(caret)
library(data.table)
library(lubridate)
library(ggplot2)
library(formatR)
```

GOOGLE PLAYSTORE dataset:

The Google Store dataset summarised several information about the Applications released on Google Store for download and instalation in Android mobile phones. The Google Store Dataset included the following 29 columns:

1. App Name
2. App Id

3. Category
4. Rating
5. Rating Count
6. Installs
7. Minimum Installs
8. Free
9. Price
10. Currency
11. Size
12. Minimum Android
13. Developer Id
14. Developer Website
15. Developer Email
16. Released
17. Last update
18. Privacy Policy
19. Content Rating
20. Ad Supported
21. In app purchases
22. Editor Choice
23. Summary
24. Reviews
25. Android version Text
26. Developer
27. Developer Address
28. Developer Internal ID
29. Version

The dataset was downloaded from Kaggle website (https://www.kaggle.com/geothomas/playstore-dataset/download) in a zip file. After extracting the zip file, a csv data file (Playstore_final.csv) is obtained.

In the CSV file, the columns are separated with , charcter , and the text quoted with " charcter. The csv data file loaded into R studio using read.csv2 function.

```
loc <- "D:/R/GOOGLE STORE/Playstore_final.csv/Playstore_final.csv"
if (!exists("d1")) d0 <- read.csv2(loc, header = TRUE, sep = ",",
    quote = "\"", encoding = "UTF-8")
```

The columns "Rating" , "Reviews", "Rating.Count", "Price" should be numerics. NA occurs upon conversion of character to numerics due to erroreous data lines, the NA lines in dataset are removed from dataset:

```r
d0a <- d0[!is.na(as.numeric(d0$Rating)), ]
d0b <- d0a[!is.na(as.numeric(d0a$Reviews)), ]
d0c <- d0b[!is.na(as.numeric(d0b$Rating.Count)), ]
d0d <- d0c[!is.na(as.numeric(d0c$Price)), ]

# the dataset summarized mean Rating per applications the
# rating total of every application is calculated
d0e <- d0d %>%
    mutate(ratingtotal = as.numeric(Rating) * as.numeric(Rating.Count))
d0f <- d0e[!is.na(as.numeric(d0e$ratingtotal)), ]
```

Relevant columns (App.Id, Category,Rating, Rating.Count, ratingtotal, Developer.Id, Developer) are selected :

```r
d0g <- d0f %>%
    select(App.Id, Category, Content.Rating, Rating, Rating.Count,
        ratingtotal, Developer.Id, Developer)
d1 <- d0g %>%
    mutate(Rating = as.numeric(Rating), Rating.Count = as.numeric(Rating.Count),
        ratingtotal = as.numeric(ratingtotal))

# Preview d1 dataset
head(d1)
```

```
##                                                 App.Id
## 1                  com.eniseistudio.logistics_management
## 2                   com.eniseistudio.news.estados_unidos
## 3                      com.eniseistudio.dental_assistant
## 4              com.eniseistudio.course.medical_assistant
## 5 com.eniseistudio.majors.course.business_administration
## 6                             com.eniseistudio.economics
##           Category Content.Rating   Rating Rating.Count
## 1        Education       Everyone 4.090909           66
## 2 News & Magazines       Everyone 4.000000            8
## 3        Education       Everyone 3.866667           15
## 4        Education       Everyone 4.000000           18
## 5        Education       Everyone 4.023256           86
## 6        Education       Everyone 4.138614          223
##   ratingtotal         Developer.Id    Developer
## 1    270.0000 4656446977926344285 eniseistudio
## 2     32.0000 4656446977926344285 eniseistudio
## 3     58.0000 4656446977926344285 eniseistudio
## 4     72.0000 4656446977926344285 eniseistudio
## 5    346.0000 4656446977926344285 eniseistudio
## 6    922.9109 4656446977926344285 eniseistudio
```

Training data set and Test data set are created. Validation set will be 10% of GOOGLE STORE data :

```r
set.seed(1, sample.kind = "Rounding")  # if using R 3.5 or earlier, use `set.seed(1)`
test_index <- createDataPartition(y = d1$Rating, times = 1, p = 0.1,
    list = FALSE)

store <- d1[-test_index, ]
temp <- d1[test_index, ]
```

```r
# Make sure Category and Developer.Id in validation set are
# also in store set
validation <- temp %>%
    semi_join(store, by = "Developer.Id") %>%
    semi_join(store, by = "Category")

# Add rows removed from validation set back into store set

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

```
## Joining, by = c("App.Id", "Category", "Content.Rating", "Rating", "Rating.Count", "ratingtotal", "De
```

```r
store <- rbind(store, removed)
```

Store dataset:

Number of rows and columns in store dataset :

```
## [1] 301678      8
```

Preview Top 3 rows in store dataset :

```
##                                  App.Id         Category
## 1 com.eniseistudio.logistics_management        Education
## 2  com.eniseistudio.news.estados_unidos News & Magazines
## 3     com.eniseistudio.dental_assistant        Education
##   Content.Rating   Rating Rating.Count ratingtotal
## 1       Everyone 4.090909           66         270
## 2       Everyone 4.000000            8          32
## 3       Everyone 3.866667           15          58
##          Developer.Id    Developer
## 1 4656446977926344285 eniseistudio
## 2 4656446977926344285 eniseistudio
## 3 4656446977926344285 eniseistudio
```

How many zeros were given as ratings in the store dataset:

```
##   n
## 1 0
```

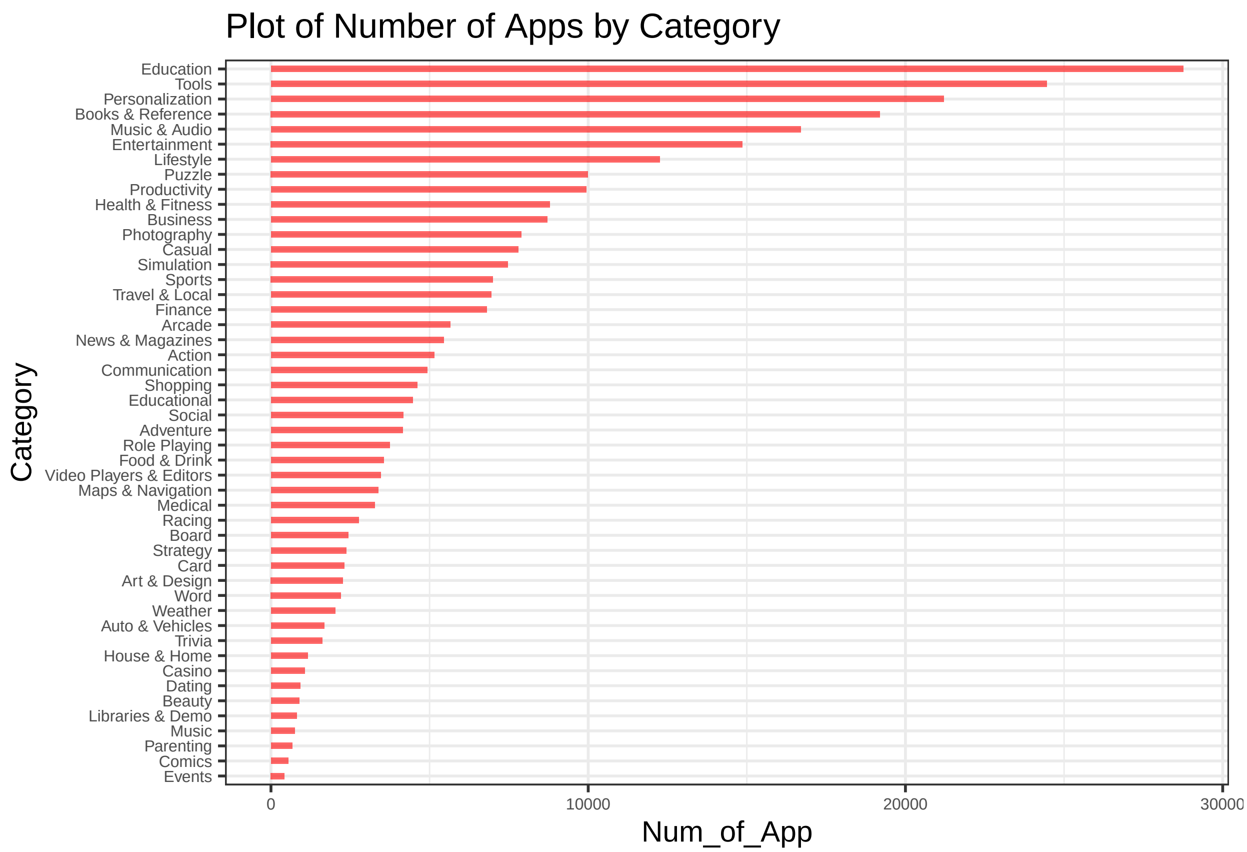
Category

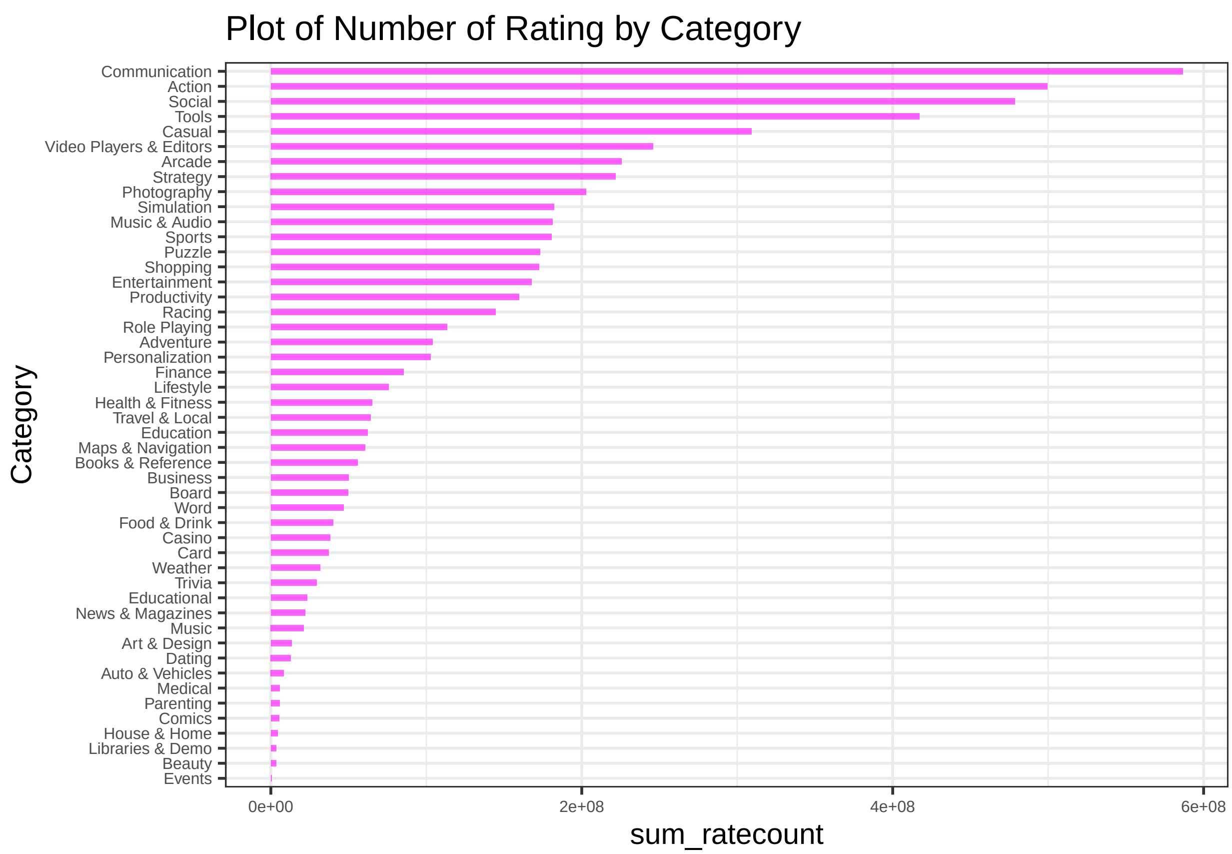
How many applications, ratings and mean rating by category in the store dataset:

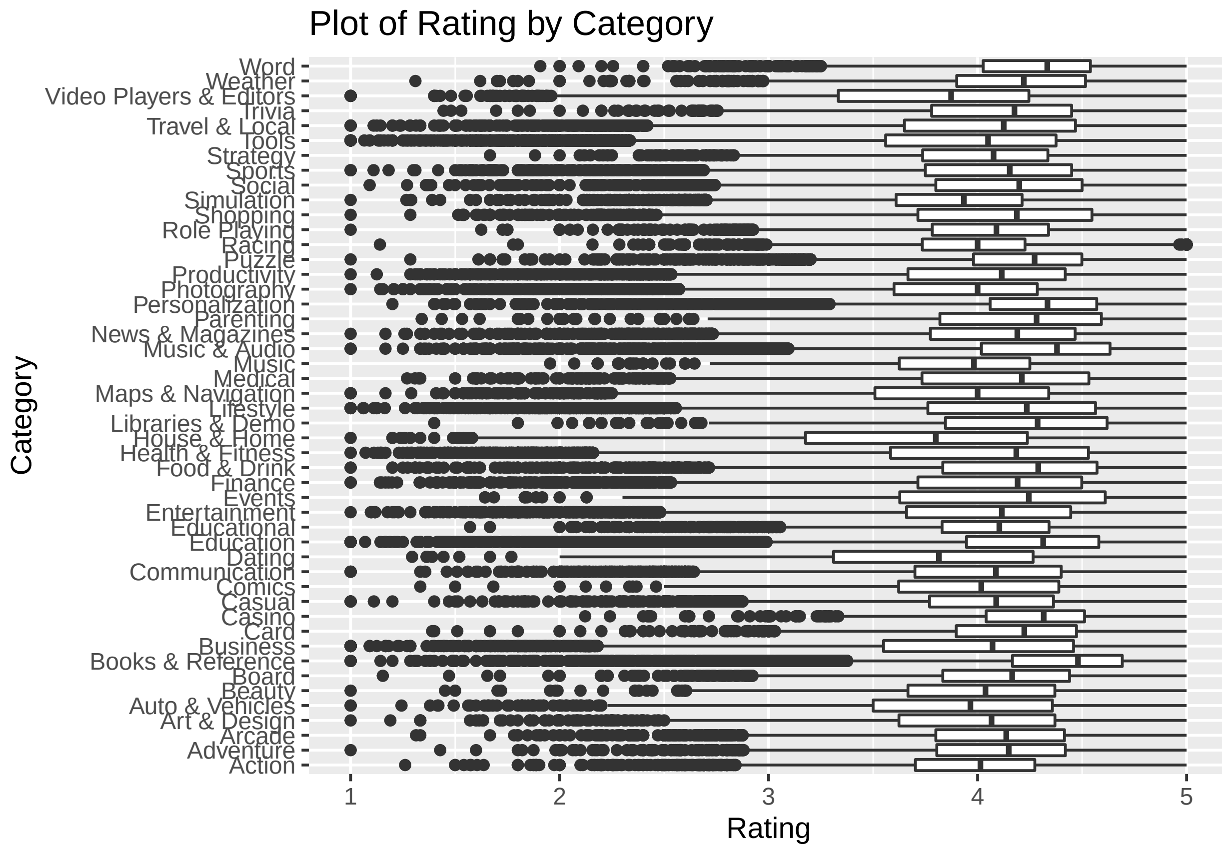
```
## # A tibble: 48 x 5
##    Category        Num_of_App sum_rating sum_ratecount Avg_rating
##    <chr>                <int>      <dbl>         <dbl>      <dbl>
##  1 Education            28757     2.74e8      62341349       4.39
##  2 Tools                24448     1.81e9     417218148       4.34
##  3 Personalization      21205     4.48e8     102656636       4.36
##  4 Books & Refere~      19197     2.50e8      55775014       4.49
##  5 Music & Audio        16701     7.98e8     181195191       4.41
##  6 Entertainment        14863     7.11e8     167602447       4.24
##  7 Lifestyle            12257     3.29e8      75695895       4.35
##  8 Puzzle                9993     7.59e8     173027482       4.38
##  9 Productivity          9936     6.95e8     159545968       4.35
## 10 Health & Fitne~       8787     2.96e8      65219871       4.54
```

```
## # ... with 38 more rows
```

Plot of Number of Apps, Number of Ratings, Rating and Mean Rating by Category:

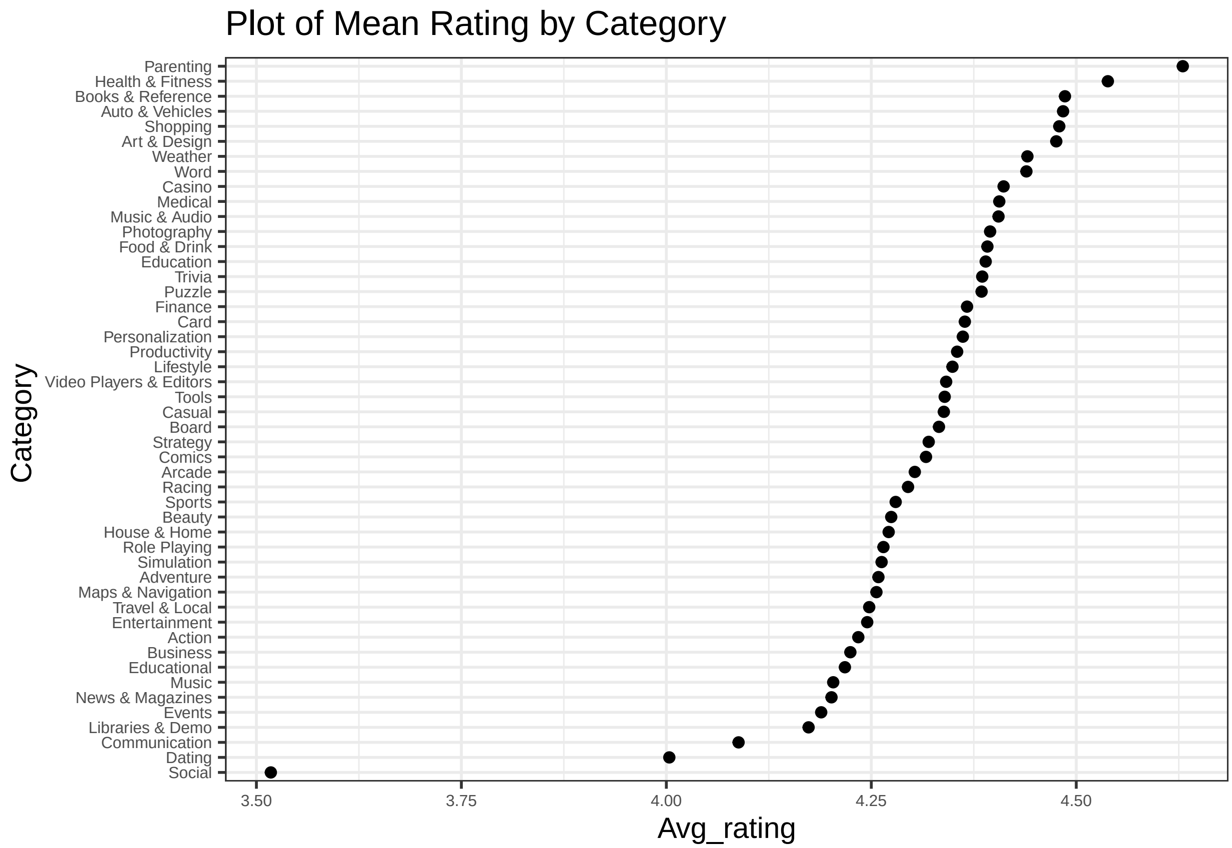



Plot of Number of Rating by Category
Communication
Action
Social
Tools
Casual
Video Players & Editors
Arcade
Strategy
Photography
Simulation
Music & Audio
Sports
Puzzle
Shopping
Entertainment
Productivity
Racing
Role Playing
Adventure
Personalization
Finance
Lifestyle
Health & Fitness
Travel & Local
Education
Maps & Navigation
Books & Reference
Business
Board
Word
Food & Drink
Casino
Card
Weather
Trivia
Educational
News & Magazines
Music
Art & Design
Dating
Auto & Vehicles
Medical
Parenting
Comics
House & Home
Libraries & Demo
Beauty
Events
Category
0e+00
2e+08
4e+08
6e+08
sum_ratecount




```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

```
## geom_path: Each group consists of only one observation. Do you
## need to adjust the group aesthetic?
```

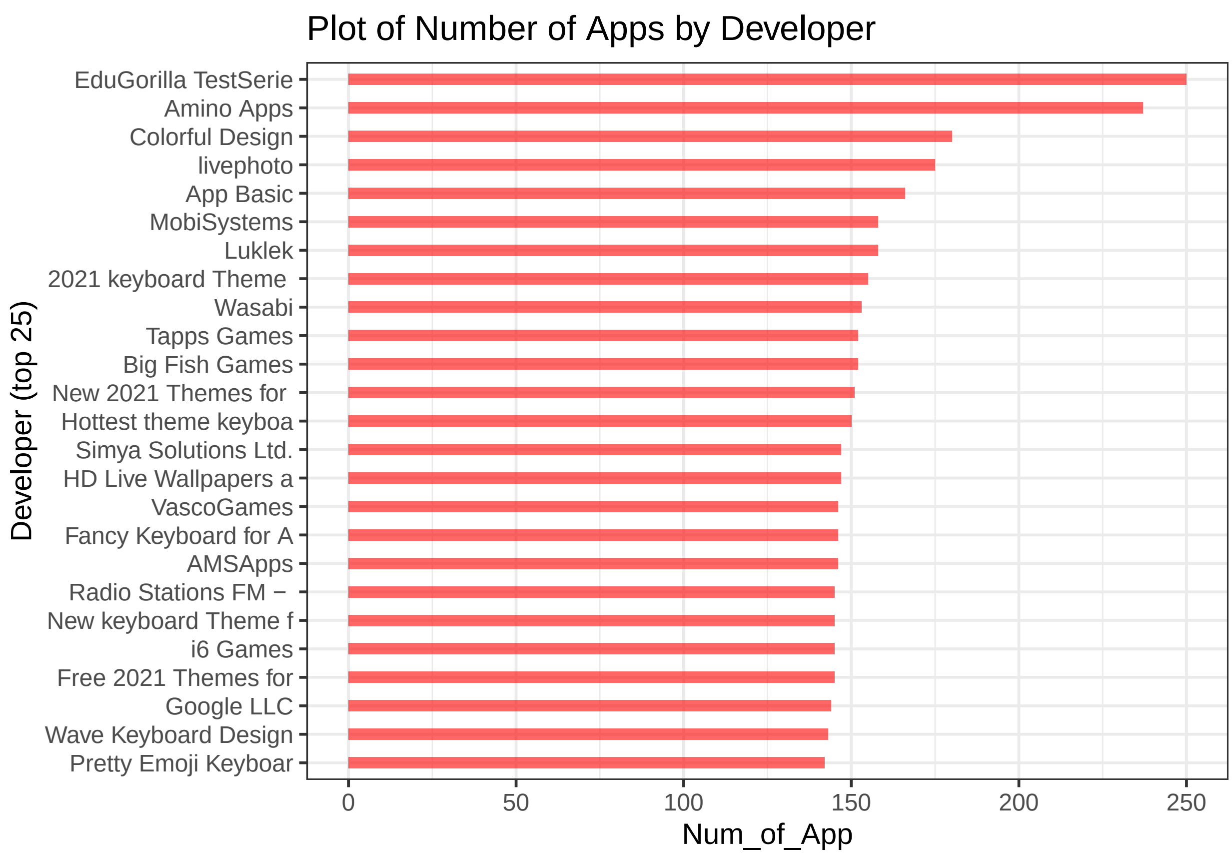


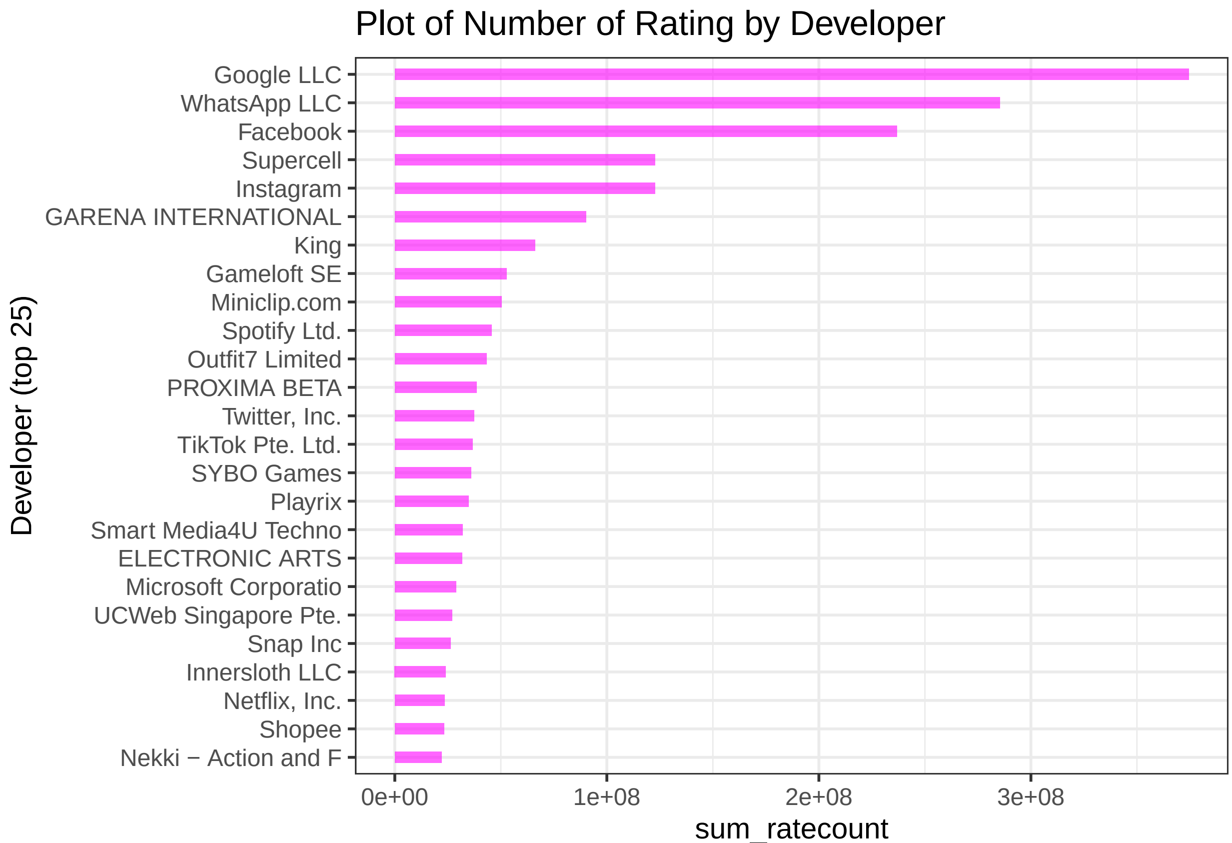
Developer

How many ratings and mean rating by Developer in the store dataset:

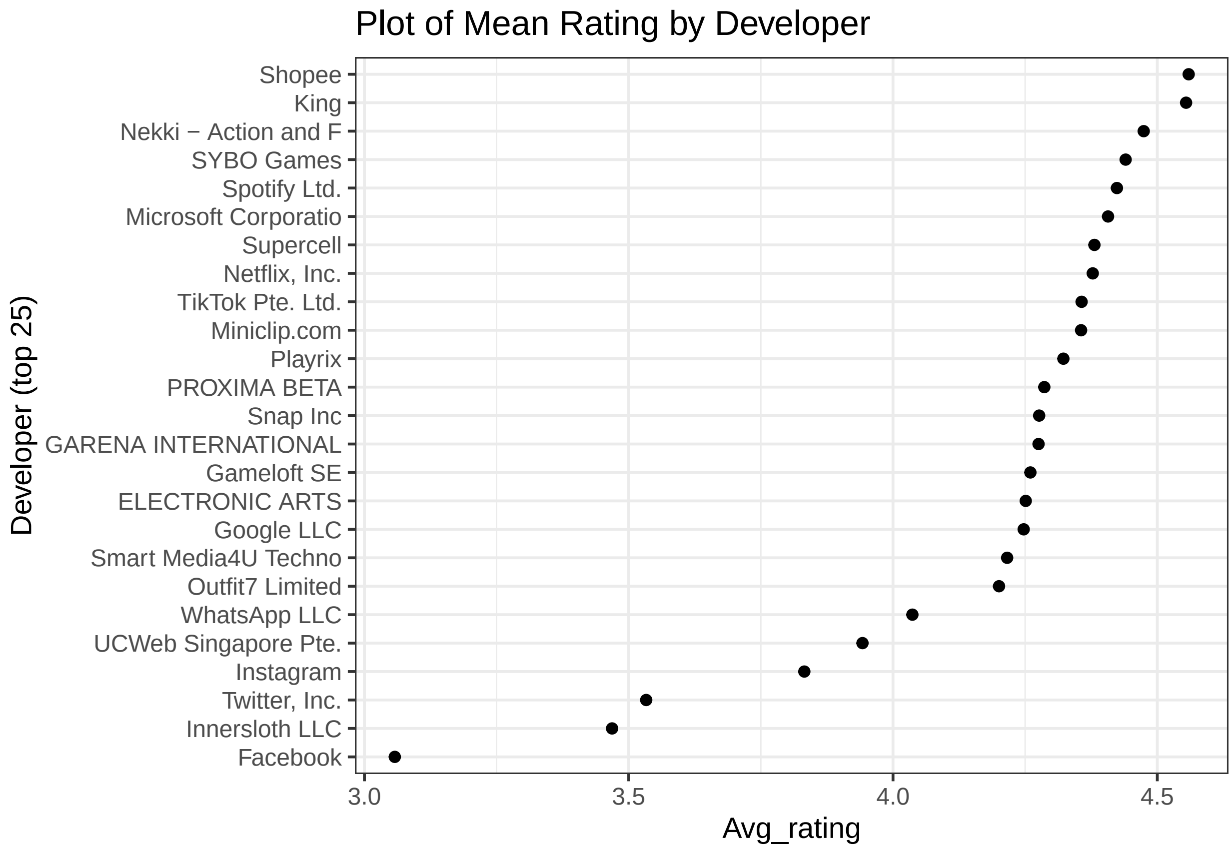
```
## # A tibble: 64,210 x 5
##    Developer       Num_of_App sum_rating sum_ratecount Avg_rating
##    <chr>                <int>      <dbl>         <dbl>      <dbl>
##  1 "EduGorilla Te~        250     16181.          3845       4.21
##  2 "Amino Apps"           237  19050043.       4253775       4.48
##  3 "Colorful Desi~        180    902366.        218990       4.12
##  4 "livephoto"            175    125062.         28931       4.32
##  5 "App Basic"            166   1532289.        392976       3.90
##  6 "Luklek"               158    169417.         40054       4.23
##  7 "MobiSystems"          158  13327434.       3176990       4.19
##  8 "2021 keyboard~        155    601412.        127940       4.70
##  9 "Wasabi"               153   4084947.        968323       4.22
## 10 "Big Fish Game~        152  12777588.       2901092       4.40
## # ... with 64,200 more rows
```

Plot of Number of Apps, Number of Ratings, Rating and Mean Rating by Developer:


Plot of Number of Apps by Developer
EduGorilla TestSerie
Amino Apps
Colorful Design
livephoto
App Basic
MobiSystems
Luklek
2021 keyboard Theme
Wasabi
Tapps Games
Big Fish Games
New 2021 Themes for
Hottest theme keyboa
Simya Solutions Ltd.
HD Live Wallpapers a
VascoGames
Fancy Keyboard for A
AMSApps
Radio Stations FM –
New keyboard Theme f
i6 Games
Free 2021 Themes for
Google LLC
Wave Keyboard Design
Pretty Emoji Keyboar
Developer (top 25)
0
50
100
150
200
250
Num_of_App




```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'

## geom_path: Each group consists of only one observation. Do you
## need to adjust the group aesthetic?
```

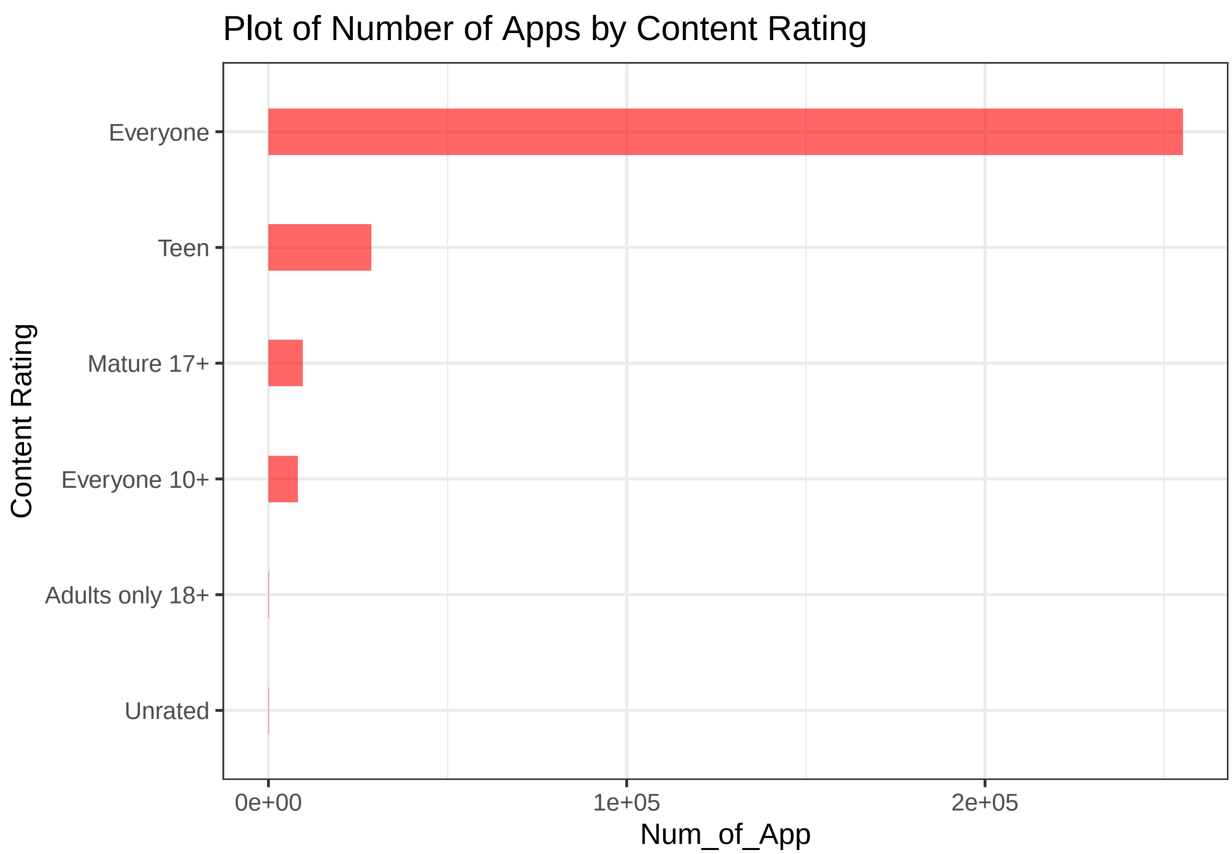


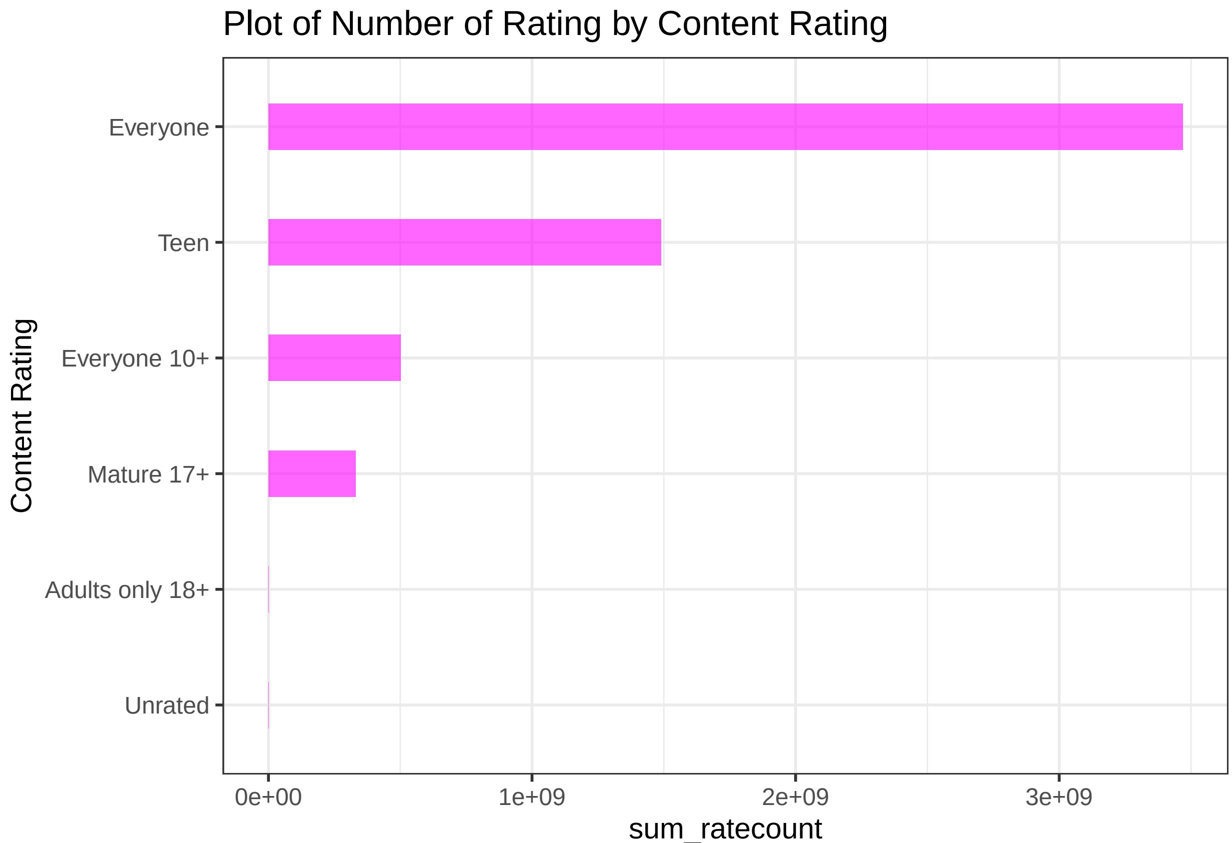
Content Rating

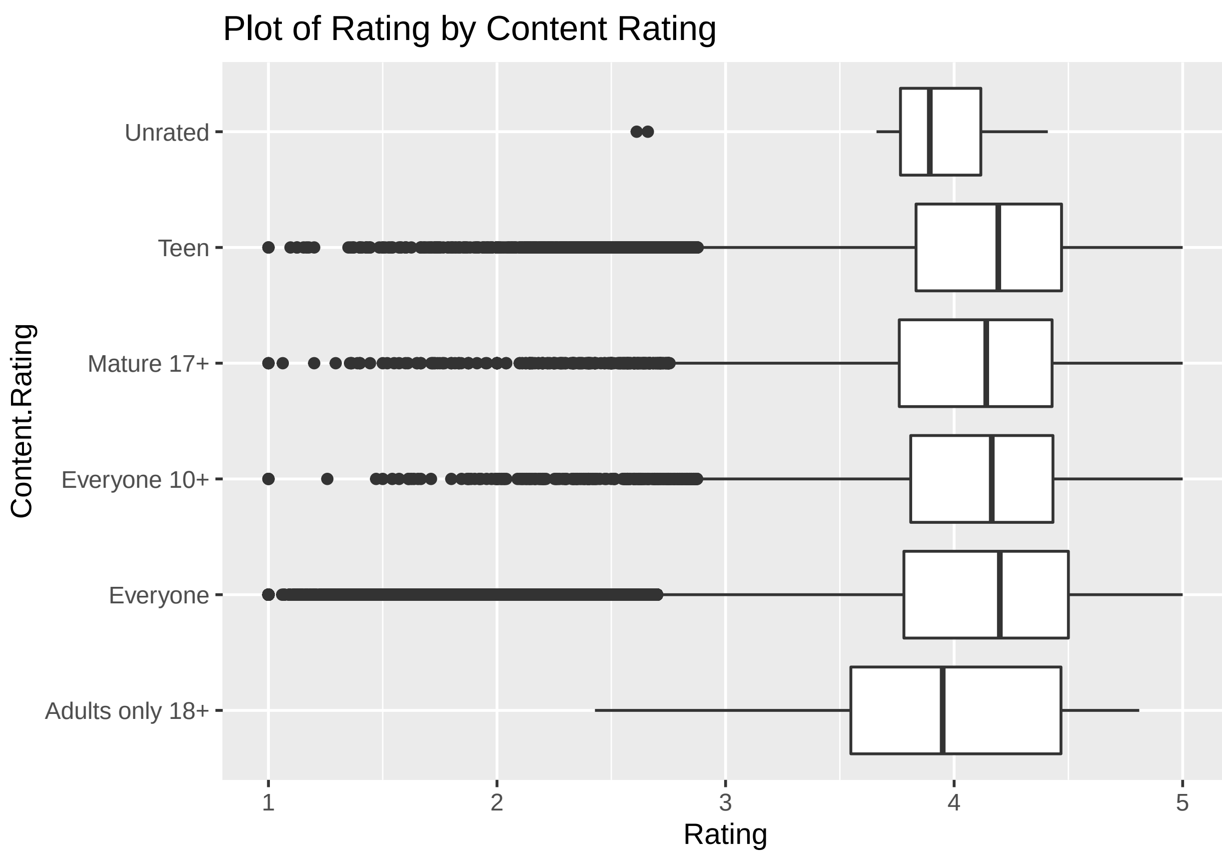
Number of ratings and mean rating by Content Rating in the store dataset:

```
## # A tibble: 6 x 5
##   Content.Rating  Num_of_App  sum_rating sum_ratecount Avg_rating
##   <chr>                <int>       <dbl>         <dbl>      <dbl>
## 1 Everyone            255163     1.49e10    3468498810       4.30
## 2 Teen                 28718     6.08e 9    1488764233       4.08
## 3 Mature 17+            9593     1.38e 9     330993626       4.16
## 4 Everyone 10+          8174     2.15e 9     500905433       4.30
## 5 Adults only 18+         19     3.60e 6        822009       4.38
## 6 Unrated                 11     2.99e 3           777       3.85
```

Plot of number of apps, number of ratings, ratings and mean rating by Content Rating:

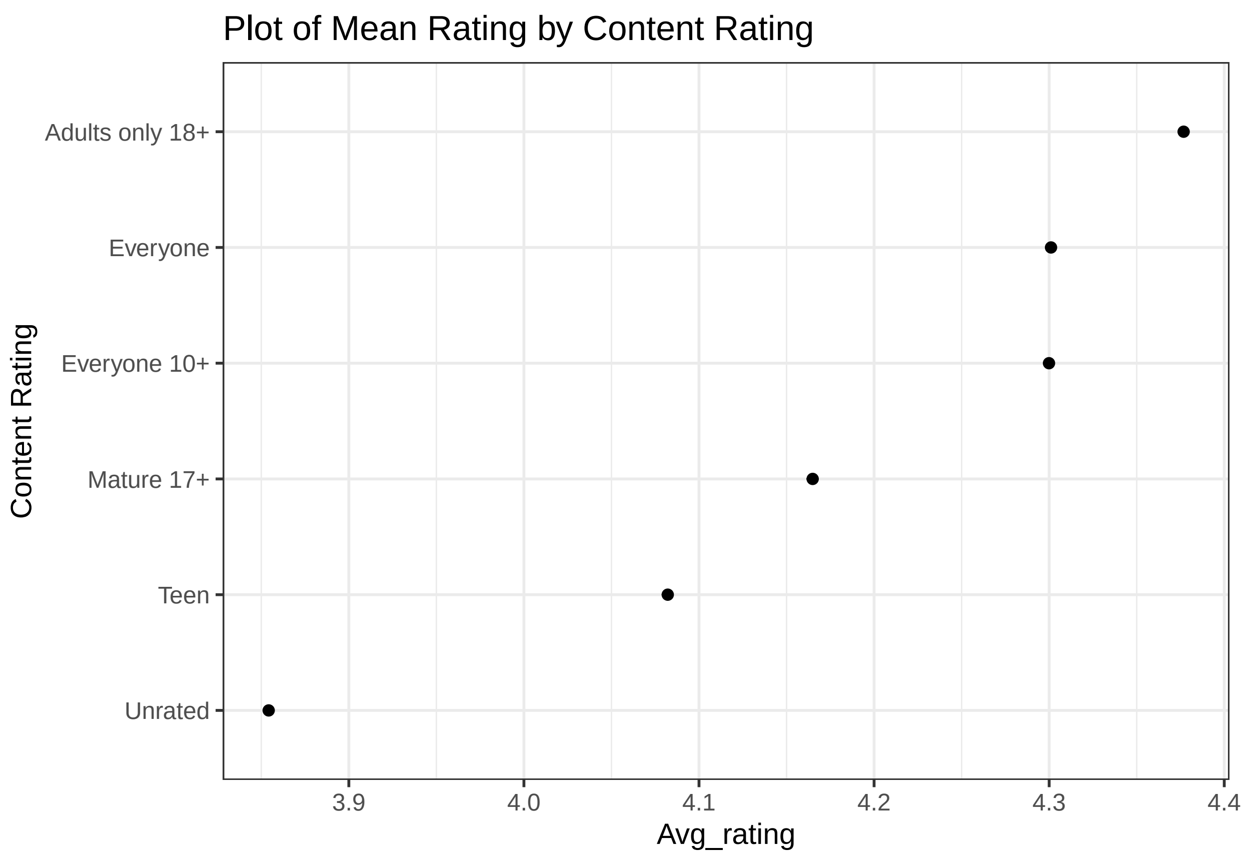

Plot of Number of Apps by Content Rating
Everyone
Teen
Mature 17+
Everyone 10+
Adults only 18+
Unrated
Content Rating
0e+00
1e+05
2e+05
Num_of_App


Plot of Number of Rating by Content Rating
Everyone
Teen
Everyone 10+
Mature 17+
Adults only 18+
Unrated
Content Rating
0e+00
1e+09
2e+09
3e+09
sum_ratecount




```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

```
## geom_path: Each group consists of only one observation. Do you
## need to adjust the group aesthetic?
```



RMSE

##Mean rating :

```
## [1] 4.236896
```

##Predict using :

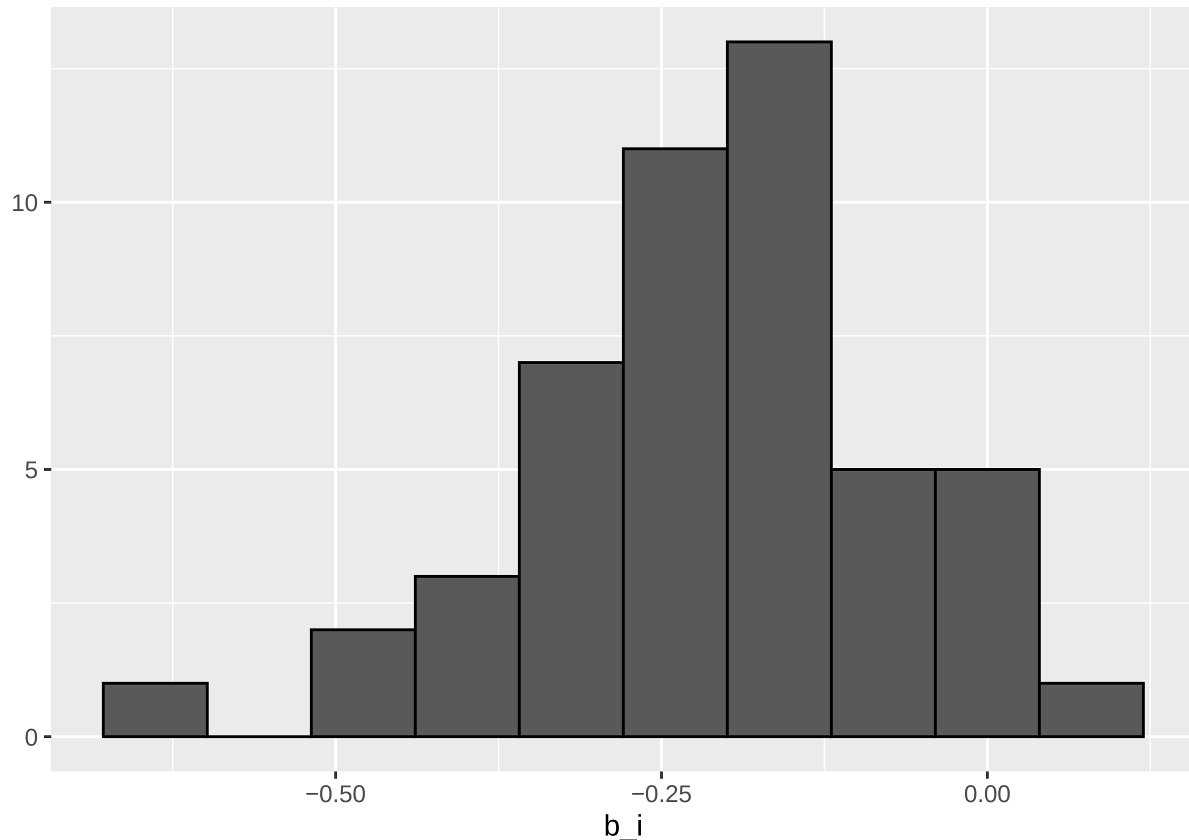
##1. Mean rating

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

naive_rmse <- RMSE(as.numeric(validation$Rating), store_mean)

predictions <- rep(round(store_mean, 1), nrow(validation))

rmse_results <- data_frame(method = "Using Mean Rating", RMSE = naive_rmse)

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

method	RMSE
Using Mean Rating	0.6366628

##2. Mean rating and Category effect :

```
Category_avgs <- store %>%
    group_by(Category) %>%
    summarize(b_i = mean(as.numeric(Rating) - store_mean))

Category_avgs %>%
    qplot(b_i, geom = "histogram", bins = 10, data = ., color = I("black"))
```

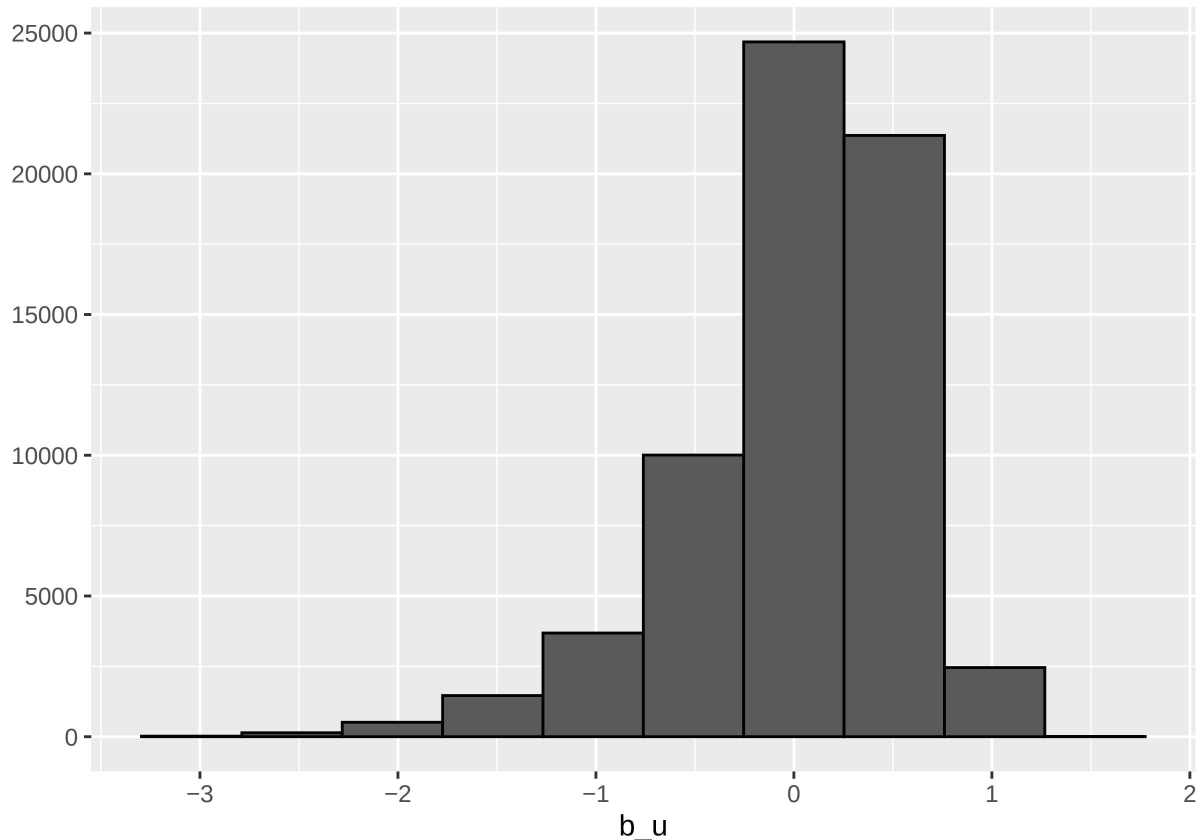



```
predicted_ratings <- store_mean + validation %>%
    left_join(Category_avgs, by = "Category") %>%
    .$b_i

model_1_rmse <- RMSE(predicted_ratings, as.numeric(validation$Rating))
rmse_results2 <- bind_rows(rmse_results, data_frame(method = "Category Effect Model",
    RMSE = model_1_rmse))

rmse_results2 %>%
    knitr::kable()
```

method	RMSE
Using Mean Rating	0.6366628
Category Effect Model	0.5959630

##3. Category + Developer.Id effect :

```r
Developer.Id_avgs <- store %>%
    select(Category, Developer.Id, Rating) %>%
    left_join(Category_avgs, by = "Category") %>%
    group_by(Developer.Id) %>%
    summarize(b_u = mean(as.numeric(Rating) - store_mean - b_i))
Developer.Id_avgs %>%
    qplot(b_u, geom = "histogram", bins = 10, data = ., color = I("black"))
```

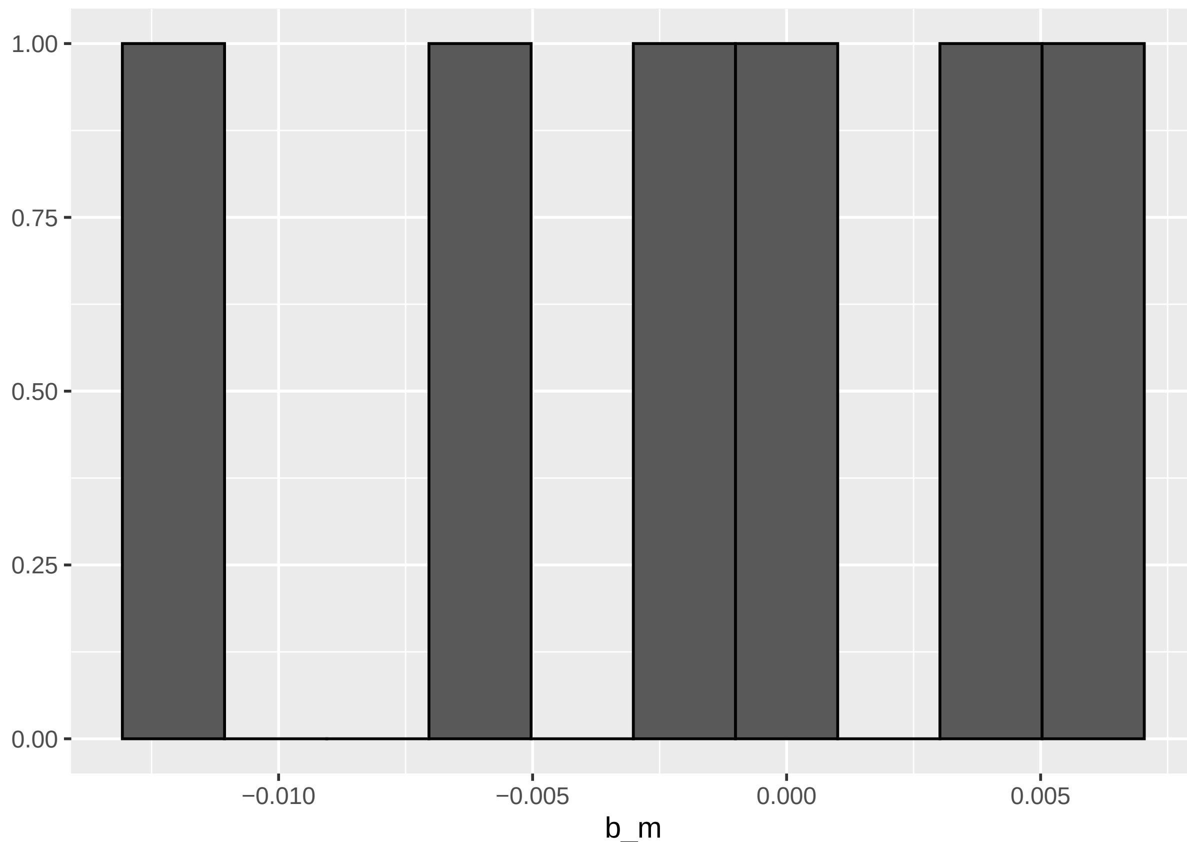



```r
predicted_ratings <- validation %>%
    select(Category, Developer.Id) %>%
    left_join(Category_avgs, by = "Category") %>%
    left_join(Developer.Id_avgs, by = "Developer.Id") %>%
    mutate(pred = store_mean + b_i + b_u) %>%
    .$pred

model_2_rmse <- RMSE(predicted_ratings, as.numeric(validation$Rating))
rmse_results3 <- bind_rows(rmse_results2, data_frame(method = "Category + Developer.Id Effects Model",
    RMSE = model_2_rmse))
rmse_results3 %>%
    knitr::kable()
```

method	RMSE
Using Mean Rating	0.6366628
Category Effect Model	0.5959630
Category + Developer.Id Effects Model	0.5152374

##4. Content.Rating effect

```
Content.Rating_avgs <- store %>%
    select(Category, Developer.Id, Content.Rating, Rating) %>%
    left_join(Category_avgs, by = "Category") %>%
    left_join(Developer.Id_avgs, by = "Developer.Id") %>%
    group_by(Content.Rating) %>%
    summarize(b_m = mean(as.numeric(Rating) - store_mean - b_i -
        b_u))
Content.Rating_avgs %>%
    qplot(b_m, geom = "histogram", bins = 10, data = ., color = I("black"))
```

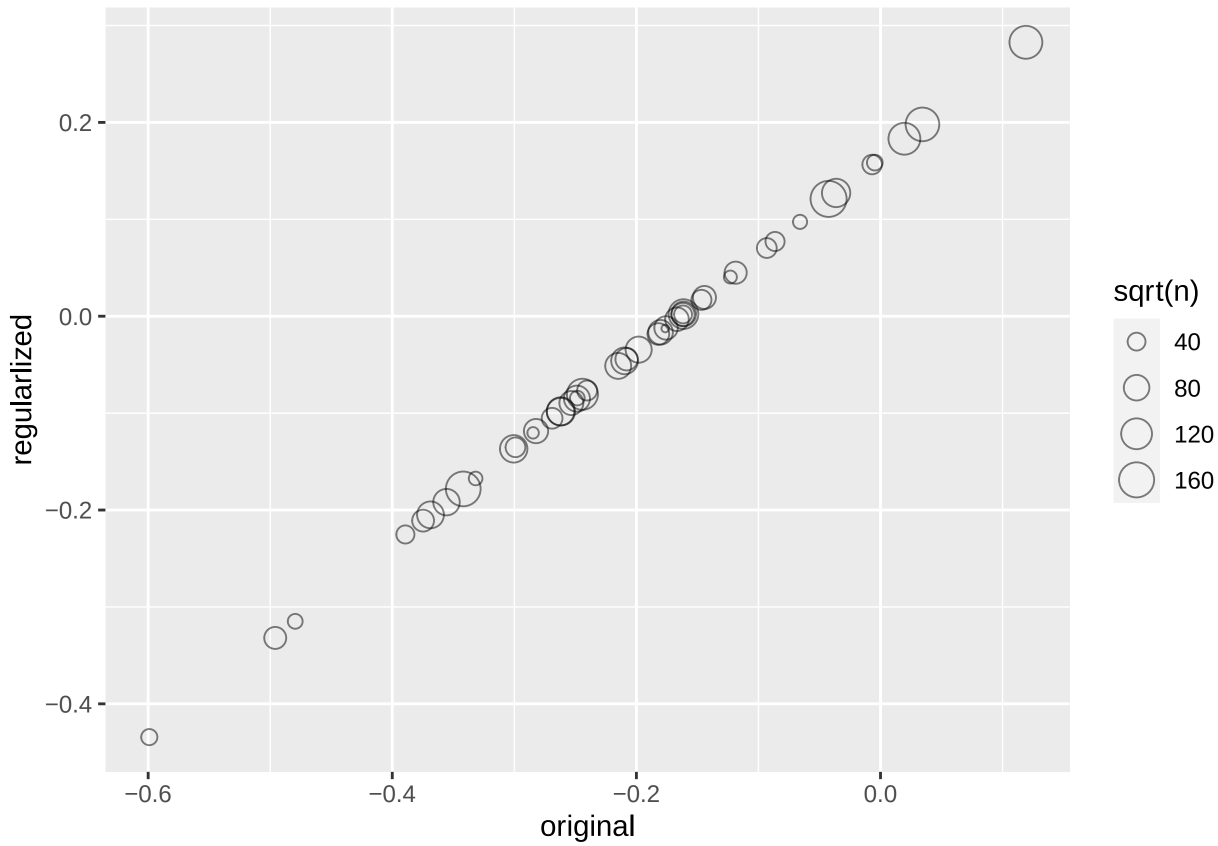



```
predicted_ratings <- validation %>%
    select(Category, Developer.Id, Content.Rating) %>%
    left_join(Category_avgs, by = "Category") %>%
    left_join(Developer.Id_avgs, by = "Developer.Id") %>%
    left_join(Content.Rating_avgs, by = "Content.Rating") %>%
    mutate(pred = store_mean + b_i + b_u + b_m) %>%
    .$pred
model_3_rmse <- RMSE(predicted_ratings, as.numeric(validation$Rating))
rmse_results4 <- bind_rows(rmse_results3, data_frame(method = "Category + Developer.Id + Content.Rating
    RMSE = model_3_rmse))
rmse_results4 %>%
    knitr::kable()
```

method	RMSE
Using Mean Rating	0.6366628
Category Effect Model	0.5959630
Category + Developer.Id Effects Model	0.5152374
Category + Developer.Id + Content.Rating Effects Model	0.5152033

##5. Regularized Category Effect

```
lambda <- 3
mu <- mean(as.numeric(store$Rating))
Category_reg_avgs <- store %>%
    group_by(Category) %>%
    summarize(b_i = sum(as.numeric(Rating) - mu)/(n() + lambda),
        n_i = n())
data_frame(original = Category_avgs$b_i, regularlized = Category_reg_avgs$b_i,
    n = Category_reg_avgs$n_i) %>%
    ggplot(aes(original, regularlized, size = sqrt(n))) + geom_point(shape = 1,
    alpha = 0.5)
```




```
store %>%
    dplyr::count(Category) %>%
    left_join(Category_reg_avgs) %>%
    arrange(desc(b_i)) %>%
    select(b_i, n) %>%
    slice(1:10) %>%
    knitr::kable()
```

```
## Joining, by = "Category"
```

b_i	n
0.2826767	19197
0.1979439	21205
0.1831534	16701
0.1585038	1070
0.1565298	2207
0.1272060	9993
0.1210695	28757
0.0973413	813
0.0771053	2022
0.0703744	2315

```r
validation %>%
    dplyr::count(Category) %>%
    left_join(Category_reg_avgs) %>%
    arrange(b_i) %>%
    select(b_i, n) %>%
    slice(1:10) %>%
    knitr::kable()
```

```
## Joining, by = "Category"
```

b_i	n
-0.4343312	91
-0.3319806	360
-0.3148709	88
-0.2252381	186
-0.2109490	301
-0.2049770	854
-0.1918714	780
-0.1782137	2412
-0.1674354	74
-0.1368513	798

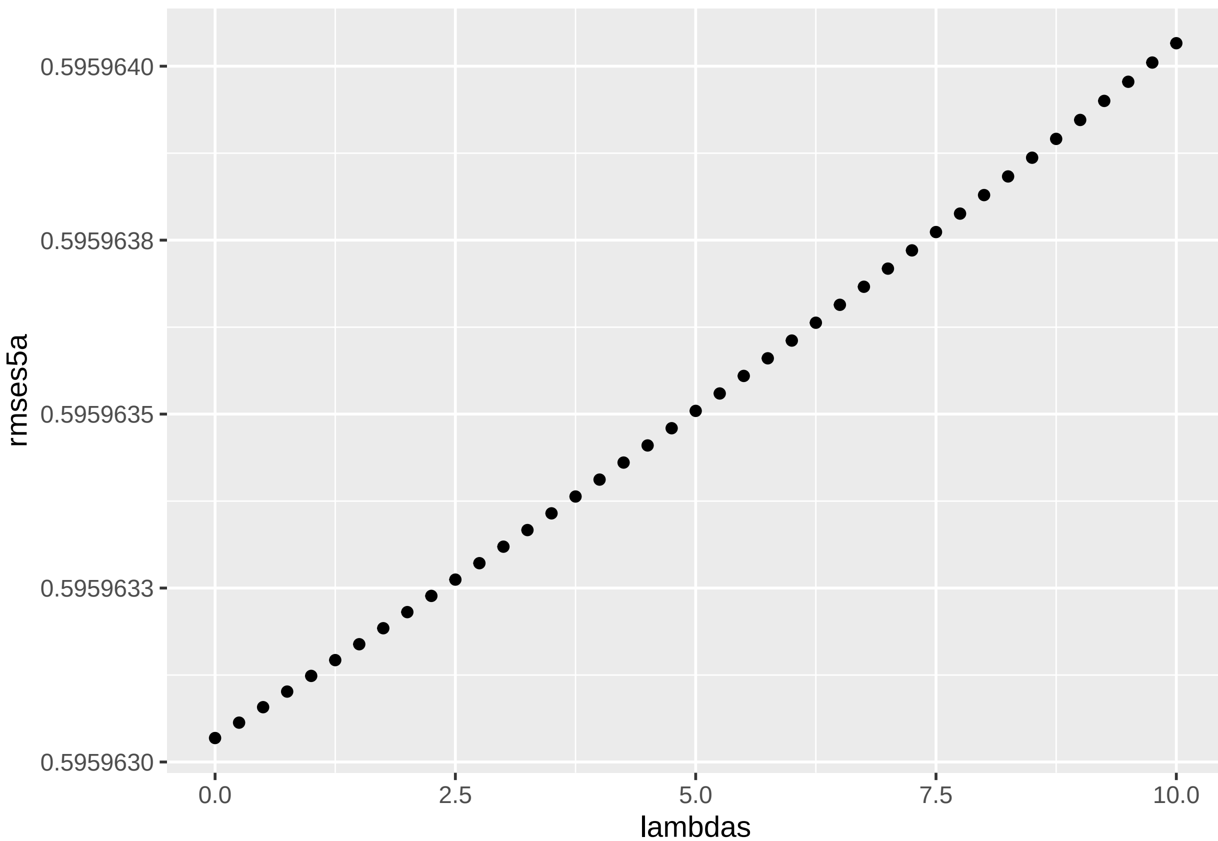
```r
predicted_Ratings <- validation %>%
    left_join(Category_reg_avgs, by = "Category") %>%
    mutate(pred = mu + b_i) %>%
    .$pred
model_5_rmse <- RMSE(predicted_Ratings, as.numeric(validation$Rating))
rmse_results5 <- bind_rows(rmse_results, data_frame(method = "Regularized Category Effect Model",
    RMSE = model_5_rmse))

rmse_results5 %>%
    knitr::kable()
```

method	RMSE
Using Mean Rating	0.6366628
Regularized Category Effect Model	0.5959633

```r
rm(Category_reg_avgs)
```

##6. optimise lamdas for Category effect

```r
lambdas <- seq(0, 10, 0.25)
mu <- mean(as.numeric(store$Rating))
just_the_sum <- store %>%
    group_by(Category) %>%
    summarize(s = sum(as.numeric(Rating) - mu), n_i = n())
rmses5a <- sapply(lambdas, function(l) {
    predicted_Ratings <- validation %>%
        left_join(just_the_sum, by = "Category") %>%
        mutate(b_i = s/(n_i + l)) %>%
        mutate(pred = mu + b_i) %>%
        .$pred
    return(RMSE(predicted_Ratings, as.numeric(validation$Rating)))
})
qplot(lambdas, rmses5a)
```




```r
l1 <- lambdas[which.min(rmses5a)]
l1
```

```
## [1] 0
```

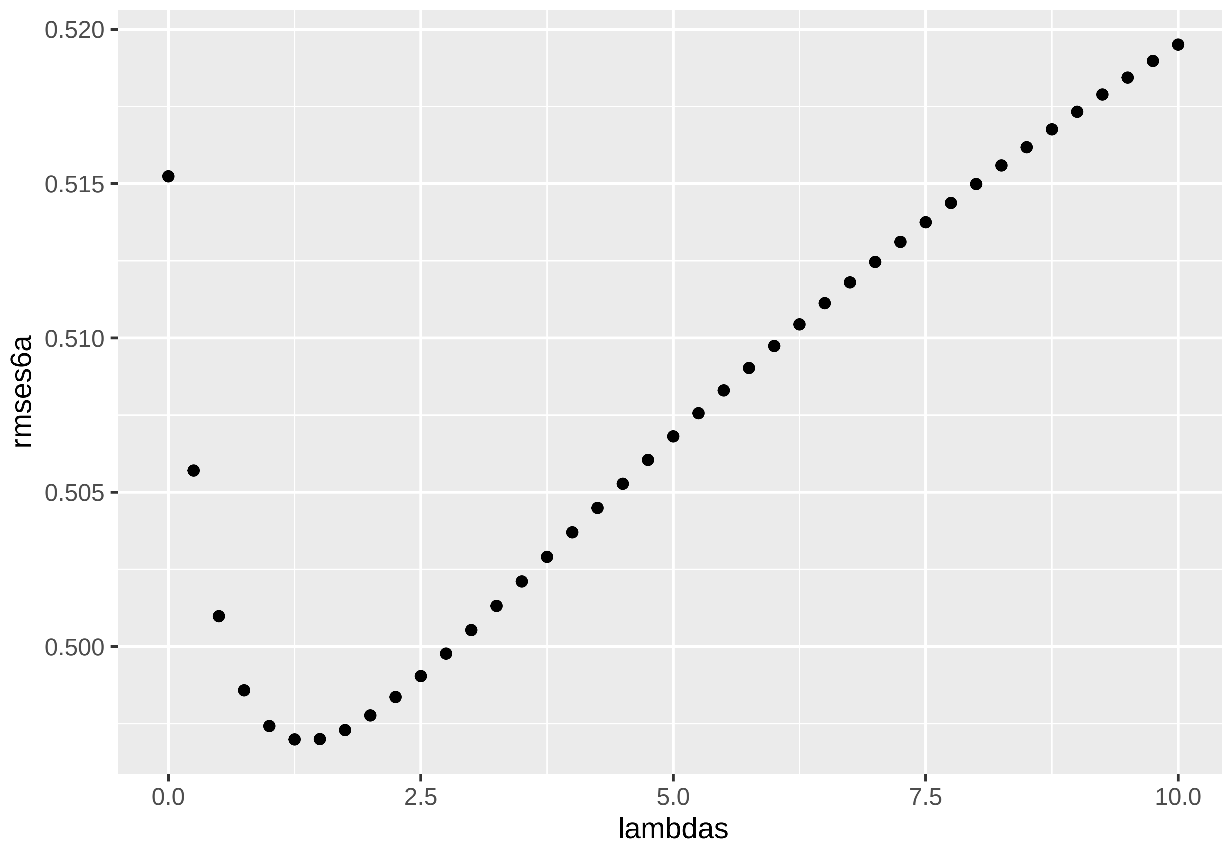
```r
rmses5 <- validation %>%
    left_join(just_the_sum, by = "Category") %>%
    mutate(b_i = s/(n_i + l1)) %>%
    mutate(pred = mu + b_i) %>%
```

```r
    .$pred
model_5_rmse <- RMSE(rmses5, as.numeric(validation$Rating))
rmse_results5 <- bind_rows(rmse_results, data_frame(method = "Regularized Category Effect Model",
    RMSE = model_5_rmse))

rmse_results5 %>%
    knitr::kable()
```

method	RMSE
Using Mean Rating	0.6366628
Regularized Category Effect Model	0.5959630

##7. optimise lambdas for Developer.Id effect

```r
lambdas <- seq(0, 10, 0.25)

Category_avgs2e <- store %>%
    select(Category) %>%
    left_join(just_the_sum, by = "Category") %>%
    mutate(b_i = s/(n_i + l1)) %>%
    select(Category, b_i)
Category_avgs2v <- validation %>%
    select(Category) %>%
    left_join(just_the_sum, by = "Category") %>%
    mutate(b_i = s/(n_i + l1)) %>%
    select(Category, b_i)

u1 <- store %>%
    select(Developer.Id, Rating) %>%
    cbind(Category_avgs2e$b_i) %>%
    set_names("Developer.Id", "Rating", "b_i") %>%
    group_by(Developer.Id) %>%
    summarize(s = sum(as.numeric(Rating) - mu - b_i), n_i = n()) %>%
    select(Developer.Id, s, n_i)


rmses6a <- sapply(lambdas, function(l) {
    predicted_Ratings <- validation %>%
        select(Developer.Id) %>%
        cbind(rmses5) %>%
        set_names("Developer.Id", "mu_b_i") %>%
        left_join(u1, by = "Developer.Id") %>%
        mutate(b_u = s/(n_i + l)) %>%
        mutate(pred = mu_b_i + b_u) %>%
        .$pred
    return(RMSE(predicted_Ratings, as.numeric(validation$Rating)))
})
qplot(lambdas, rmses6a)
```




```
l2 <- lambdas[which.min(rmses6a)]
l2
```

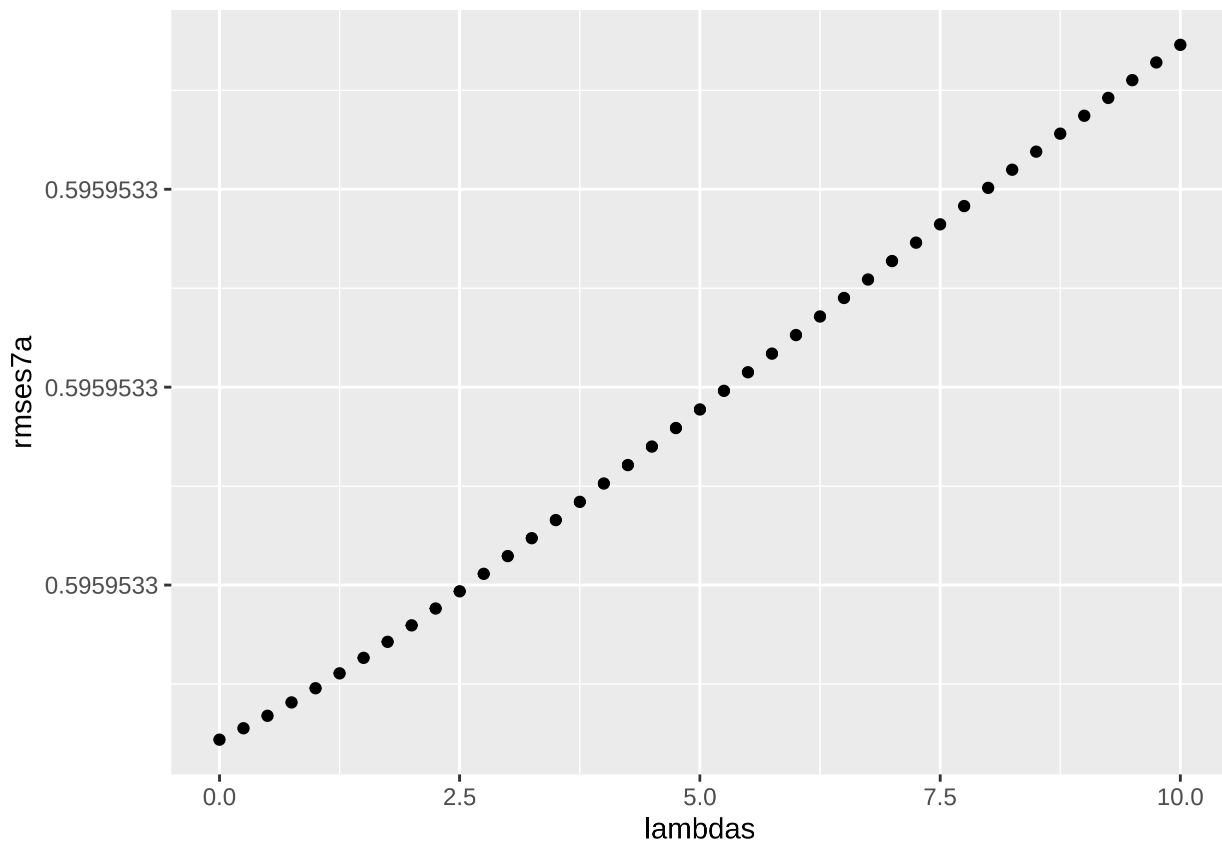
```
## [1] 1.25
```

```
rmses6 <- validation %>%
    select(Developer.Id) %>%
    cbind(rmses5) %>%
    set_names("Developer.Id", "mu_b_i") %>%
    left_join(u1, by = "Developer.Id") %>%
    mutate(b_u = s/(n_i + l2)) %>%
    mutate(pred = mu_b_i + b_u) %>%
    .$pred
model_6_rmse <- RMSE(rmses6, as.numeric(validation$Rating))
rmse_results6 <- bind_rows(rmse_results5, data_frame(method = "Regularized Developer.Id Effect Model",
    RMSE = model_6_rmse))

rmse_results6 %>%
    knitr::kable()
```

method	RMSE
Using Mean Rating	0.6366628
Regularized Category Effect Model	0.5959630
Regularized Developer.Id Effect Model	0.4969881

##8. optimise lambdas for Content.Rating effect

```r
lambdas <- seq(0, 10, 0.25)

Developer.Id_avgs2 <- store %>%
    select(Developer.Id) %>%
    left_join(u1, by = "Developer.Id") %>%
    mutate(b_u = s/(n_i + 12)) %>%
    select(Developer.Id, b_u)

rm(u1)

g1 <- store %>%
    select(Content.Rating, Rating) %>%
    cbind(Category_avgs2e$b_i, Developer.Id_avgs2$b_u) %>%
    set_names("Content.Rating", "Rating", "b_i", "b_u") %>%
    group_by(Content.Rating) %>%
    summarize(s = sum(as.numeric(Rating) - mu - b_i - b_u), n_i = n()) %>%
    select(Content.Rating, s, n_i)


rmses7a <- sapply(lambdas, function(l) {
    predicted_Ratings <- validation %>%
        select(Content.Rating) %>%
        cbind(rmses5) %>%
        set_names("Content.Rating", "mu_b_i_bu") %>%
        left_join(g1, by = "Content.Rating") %>%
        mutate(b_g = s/(n_i + l)) %>%
        mutate(pred = mu_b_i_bu + b_g) %>%
        .$pred
    return(RMSE(predicted_Ratings, as.numeric(validation$Rating)))
})
qplot(lambdas, rmses7a)
```




```
l3 <- lambdas[which.min(rmses7a)]
l3
```

```
## [1] 0
```

```
rmses7 <- validation %>%
    select(Content.Rating) %>%
    cbind(rmses6) %>%
    set_names("Content.Rating", "mu_b_i_bu") %>%
    left_join(g1, by = "Content.Rating") %>%
    mutate(b_g = s/(n_i + l3)) %>%
    mutate(pred = mu_b_i_bu + b_g) %>%
    .$pred

model_7_rmse <- RMSE(rmses7, as.numeric(validation$Rating))
rmse_results7 <- bind_rows(rmse_results6, data_frame(method = "Regularized Content.Rating Effect Model"
    RMSE = model_7_rmse))

rmse_results7 %>%
    knitr::kable()
```

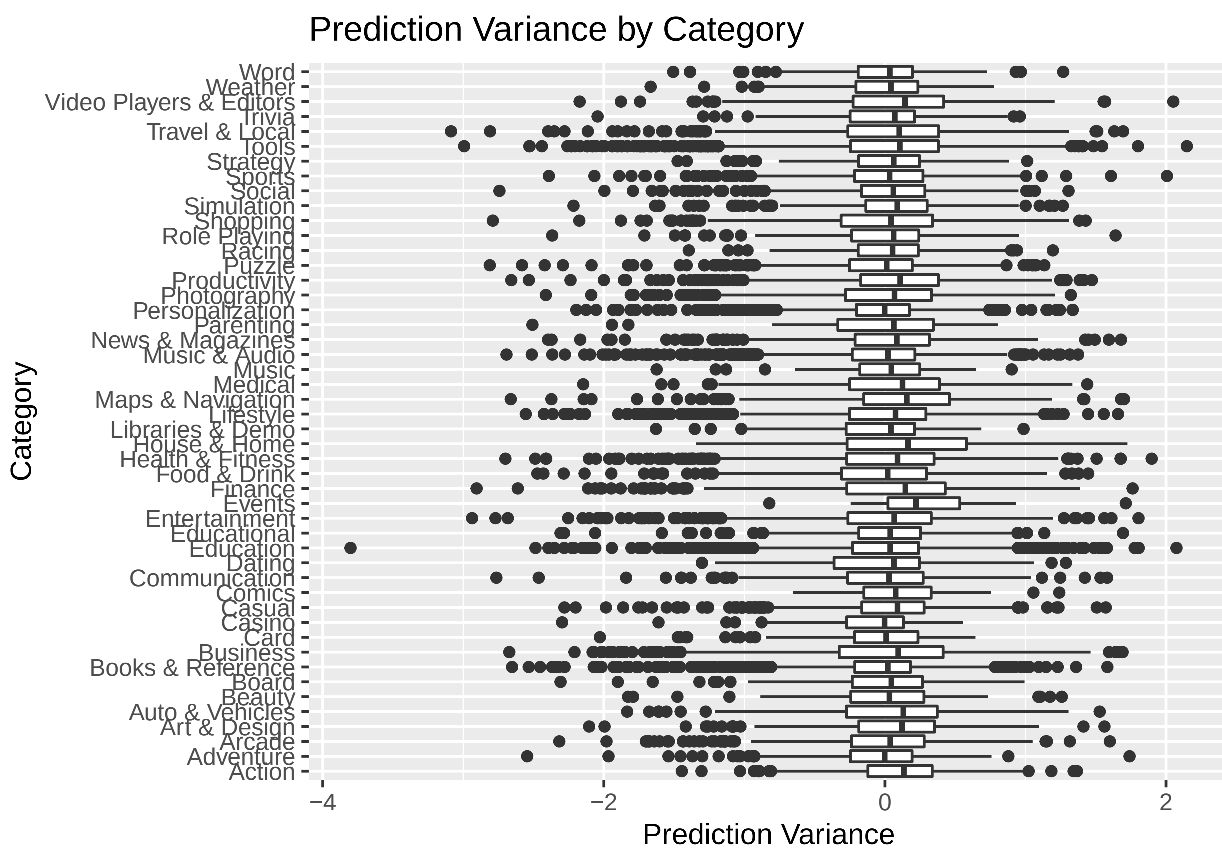
method	RMSE
Using Mean Rating	0.6366628
Regularized Category Effect Model	0.5959630
Regularized Developer.Id Effect Model	0.4969881
Regularized Content.Rating Effect Model	0.4969641

Conclusion

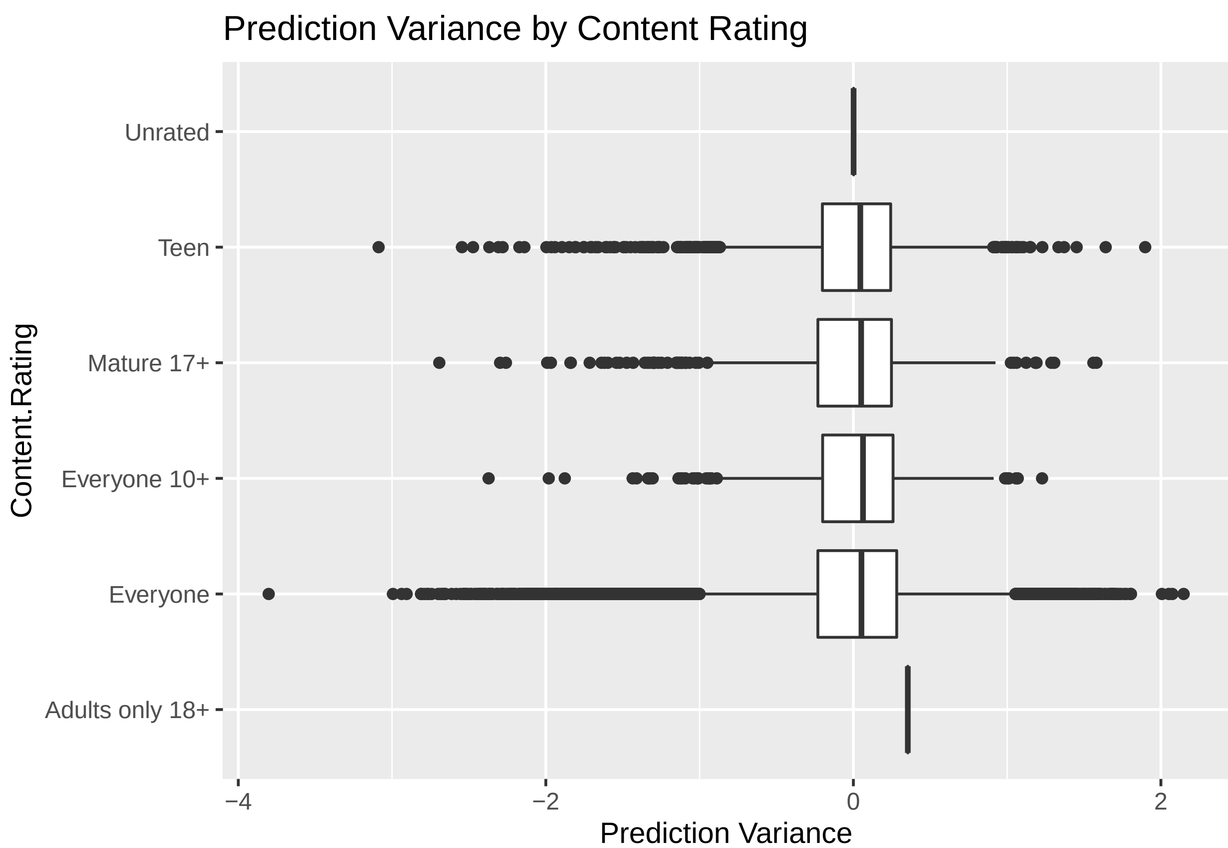
Even though there are many attributes to an applications in the GOOGLE PLAYSTORE dataset, as the dataset already summarised Mean Rating and Rating Count by application, most columns with unique values and TRUE/FALSE values are not useful in prediction (eg. App Name, App Id, Free, In App purchases, Editor Choice).

The Category, DeveloperId and Content.Rating Model is able to reduce the RMSE to 0.515, while the Regularized Category, DeveloperId and Content.Rating Model is able to reduce the RMSE to less than 0.5, which is moderately low for the Rating maximum as 5.

```
v2 <- validation %>%
    mutate(diff = validation$Rating - rmses7)
ggplot(v2, aes(factor(Category), as.numeric(diff))) + geom_boxplot() +
    xlab("Category") + ylab("Prediction Variance") + coord_flip() +
    ggtitle("Prediction Variance by Category")
```




```
ggplot(v2, aes(factor(Content.Rating), as.numeric(diff))) + geom_boxplot() +
    xlab("Content.Rating") + ylab("Prediction Variance") + coord_flip() +
    ggtitle("Prediction Variance by Content Rating")
```



End of report