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 full dataset with size of 200M could be downloaded from Kaggle website (https://www.kaggle.com/geo thomas/playstore-dataset/download). For this project, due to file size limitation on GITHUB, I trimmed the dataset to with size of 60M.

Zipfile of the dataset is downloaded from GITHUB and unzipped. 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.

```
# download dataset from GITHUB and load dataset into R
dl <- tempfile()
download.file("https://github.com/annalee0107/PLAYSTORE/raw/main/GOOGLESTORE.zip",
    dl)

loc <- unzip(dl, "GOOGLESTORE.csv")
```

```
d0 <- read.csv2(loc, header = TRUE, sep = ",", quote = "\"",
    encoding = "UTF-8")
```

The dataset summarized mean Rating per applications,the rating total of every application is calculated

```
d0a <- d0 %>%
    mutate(ratingtotal = as.numeric(Rating) * as.numeric(Rating.Count))
```

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

```
d0b <- d0a %>%
    select(App.Id, Category, Content.Rating, Rating, Rating.Count,
        ratingtotal, Developer.Id, Developer)
d1 <- d0b %>%
    mutate(Rating = as.numeric(Rating), Rating.Count = as.numeric(Rating.Count),
        ratingtotal = as.numeric(ratingtotal))

# Preview d1 dataset
head(d1)
```

```
##                                App.Id    Category Content.Rating
## 1          com.eniseistudio.economics   Education       Everyone
## 2 com.eniseistudio.course.cosmetology   Education       Everyone
## 3             education.kids.learning   Education       Everyone
## 4                   desert.runner.boy   Adventure       Everyone
## 5                     kids.word.games Educational       Everyone
## 6           word.link.connect.puzzle Educational       Everyone
##     Rating Rating.Count ratingtotal        Developer.Id
## 1 4.138614          223    922.9109 4656446977926344285
## 2 2.421053           19     46.0000 4656446977926344285
## 3 4.271845          846   3613.9809 7415446147207133288
## 4 4.588235          111    509.2941 7415446147207133288
## 5 5.000000            6     30.0000 7415446147207133288
## 6 4.588235           17     78.0000 7415446147207133288
##      Developer
## 1 eniseistudio
## 2 eniseistudio
## 3  Early Learn
## 4  Early Learn
## 5  Early Learn
## 6  Early Learn
```

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

```
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, ]

# 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
```

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

Store dataset:

Number of rows and columns in store dataset :

```
## [1] 126327      8
```

Preview Top 3 rows in store dataset :

```
##                                App.Id  Category Content.Rating
## 1          com.eniseistudio.economics Education       Everyone
## 2 com.eniseistudio.course.cosmetology Education       Everyone
## 3             education.kids.learning Education       Everyone
##     Rating Rating.Count ratingtotal        Developer.Id
## 1 4.138614          223    922.9109 4656446977926344285
## 2 2.421053           19     46.0000 4656446977926344285
## 3 4.271845          846   3613.9809 7415446147207133288
##      Developer
## 1 eniseistudio
## 2 eniseistudio
## 3  Early Learn
```

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

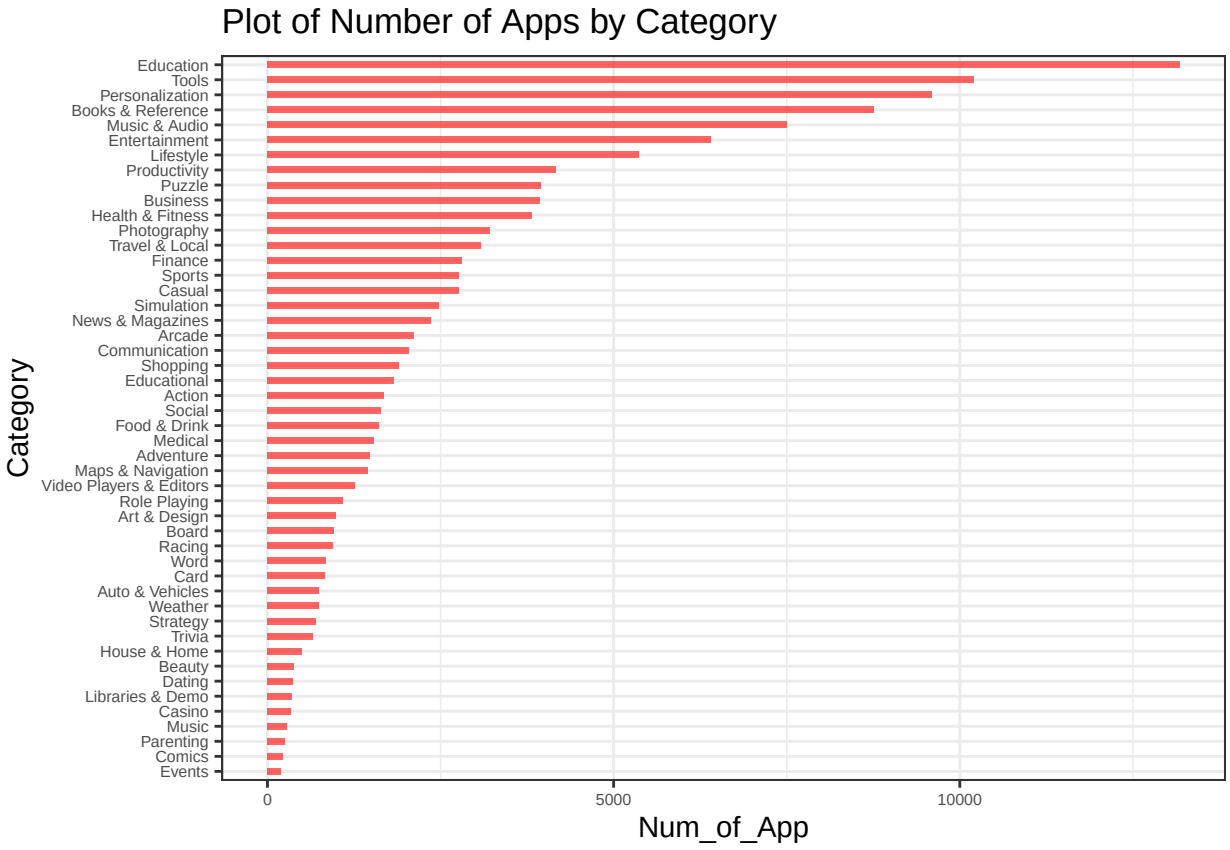
```
##   n
## 1 0
```

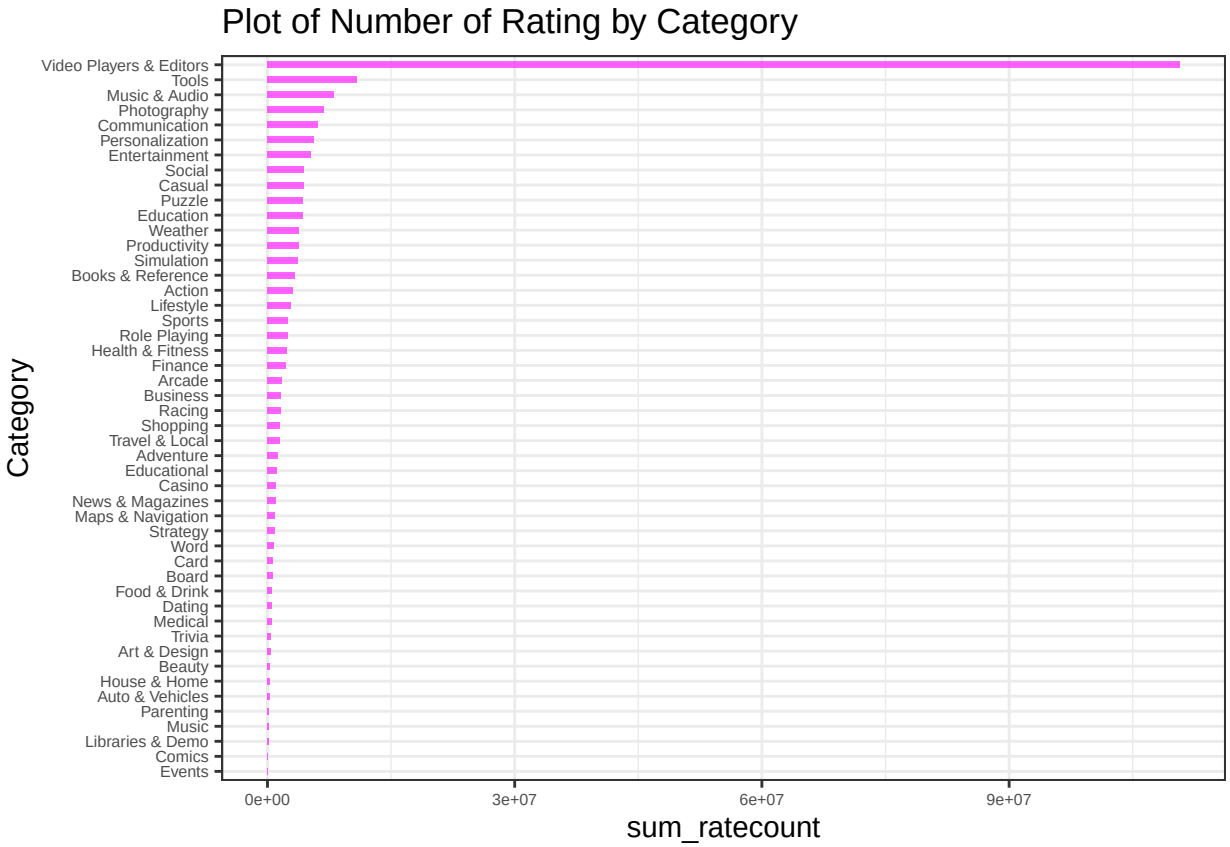
Category

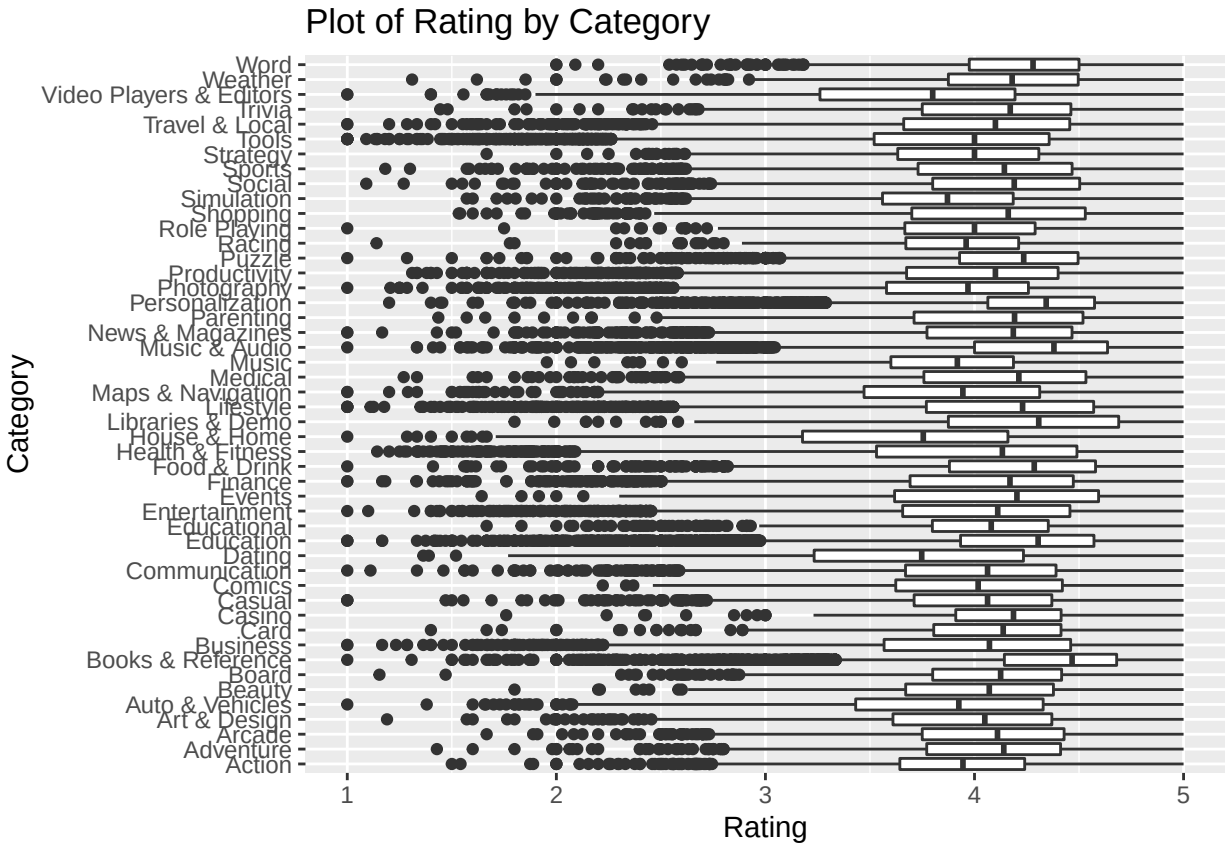
How many applications, ratings and mean rating by category in the store dataset:

```
## # A tibble: 48 x 5
##    Category        Num_of_App sum_rating sum_ratecount Avg_rating
##    <chr>                <int>      <dbl>         <dbl>      <dbl>
##  1 Education            13177  18748697.       4340818       4.32
##  2 Tools                10208  45225708.      10852537       4.17
##  3 Personalization       9591  24037598.       5671409       4.24
##  4 Books & Refere~       8755  14689841.       3372246       4.36
##  5 Music & Audio         7497  36235511.       8128101       4.46
##  6 Entertainment         6406  21481188.       5225238       4.11
##  7 Lifestyle             5371  11946443.       2839064       4.21
##  8 Productivity          4169  15855150.       3763034       4.21
##  9 Puzzle                3951  18643694.       4356391       4.28
## 10 Business              3938   7001753.       1668610       4.20
## # ... with 38 more rows
```

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

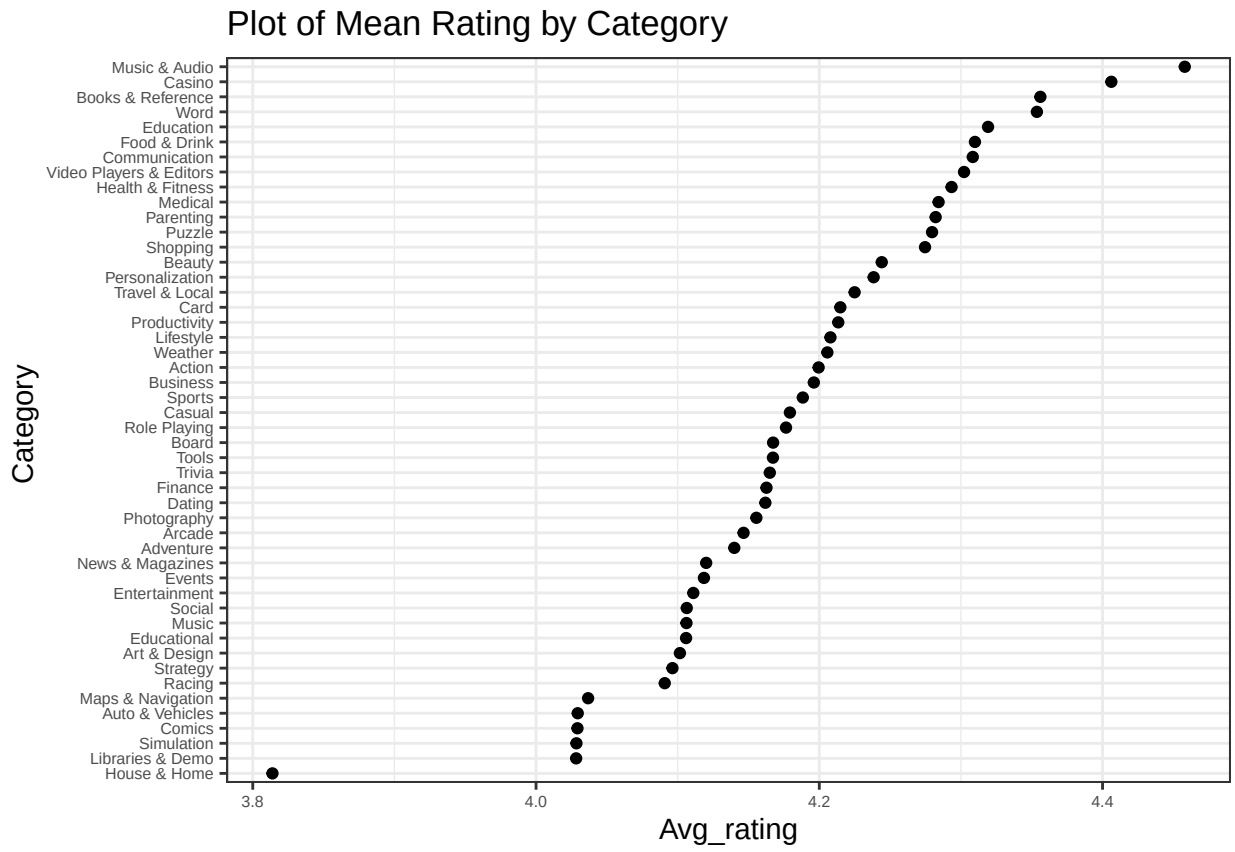

Plot of Number of Apps by Category
Category
Education
Tools
Personalization
Books & Reference
Music & Audio
Entertainment
Lifestyle
Productivity
Puzzle
Business
Health & Fitness
Photography
Travel & Local
Finance
Sports
Casual
Simulation
News & Magazines
Arcade
Communication
Shopping
Educational
Action
Social
Food & Drink
Medical
Adventure
Maps & Navigation
Video Players & Editors
Role Playing
Art & Design
Board
Racing
Word
Card
Auto & Vehicles
Weather
Strategy
Trivia
House & Home
Beauty
Dating
Libraries & Demo
Casino
Music
Parenting
Comics
Events
0
5000
10000
Num_of_App


Plot of Number of Rating by Category
Category
Video Players & Editors
Tools
Music & Audio
Photography
Communication
Personalization
Entertainment
Social
Casual
Puzzle
Education
Weather
Productivity
Simulation
Books & Reference
Action
Lifestyle
Sports
Role Playing
Health & Fitness
Finance
Arcade
Business
Racing
Shopping
Travel & Local
Adventure
Educational
Casino
News & Magazines
Maps & Navigation
Strategy
Word
Card
Board
Food & Drink
Dating
Medical
Trivia
Art & Design
Beauty
House & Home
Auto & Vehicles
Parenting
Music
Libraries & Demo
Comics
Events
0e+00
3e+07
6e+07
9e+07
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?
```

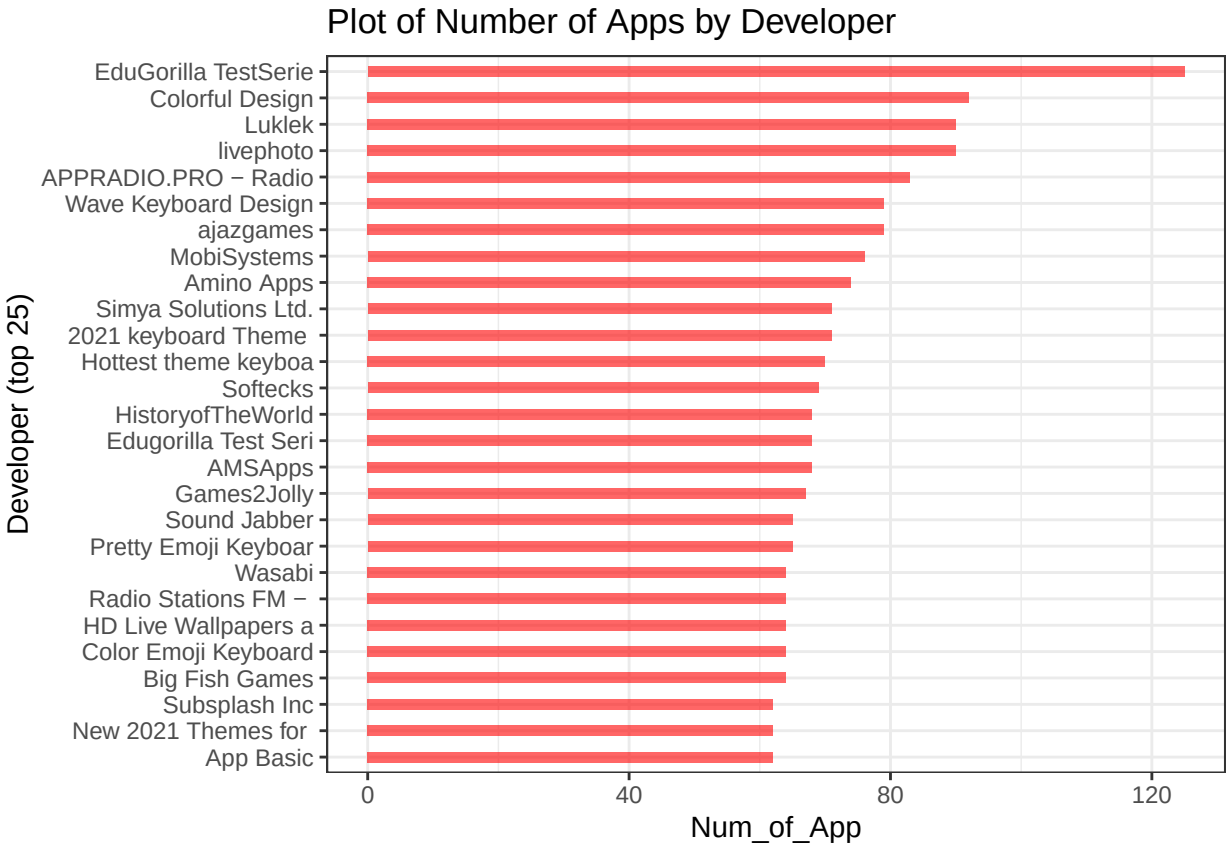


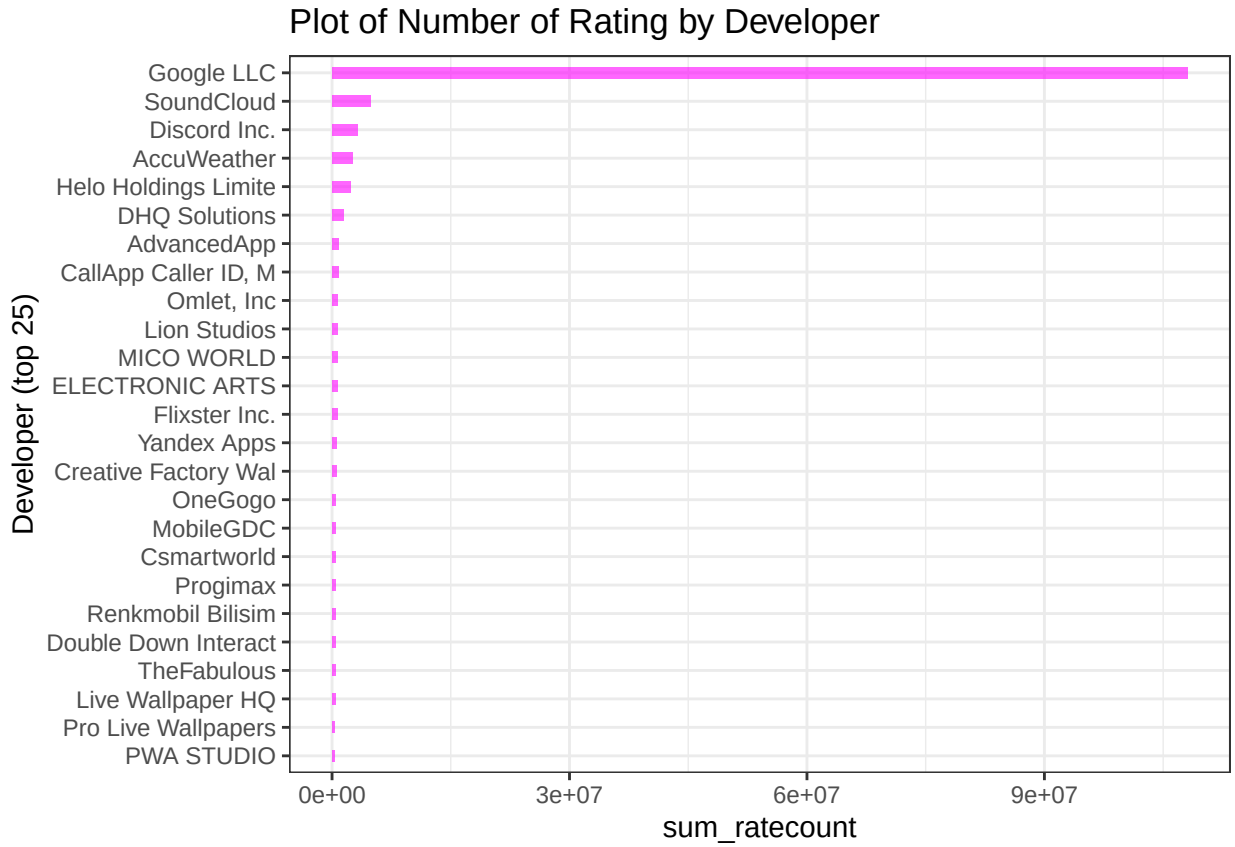
Developer

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

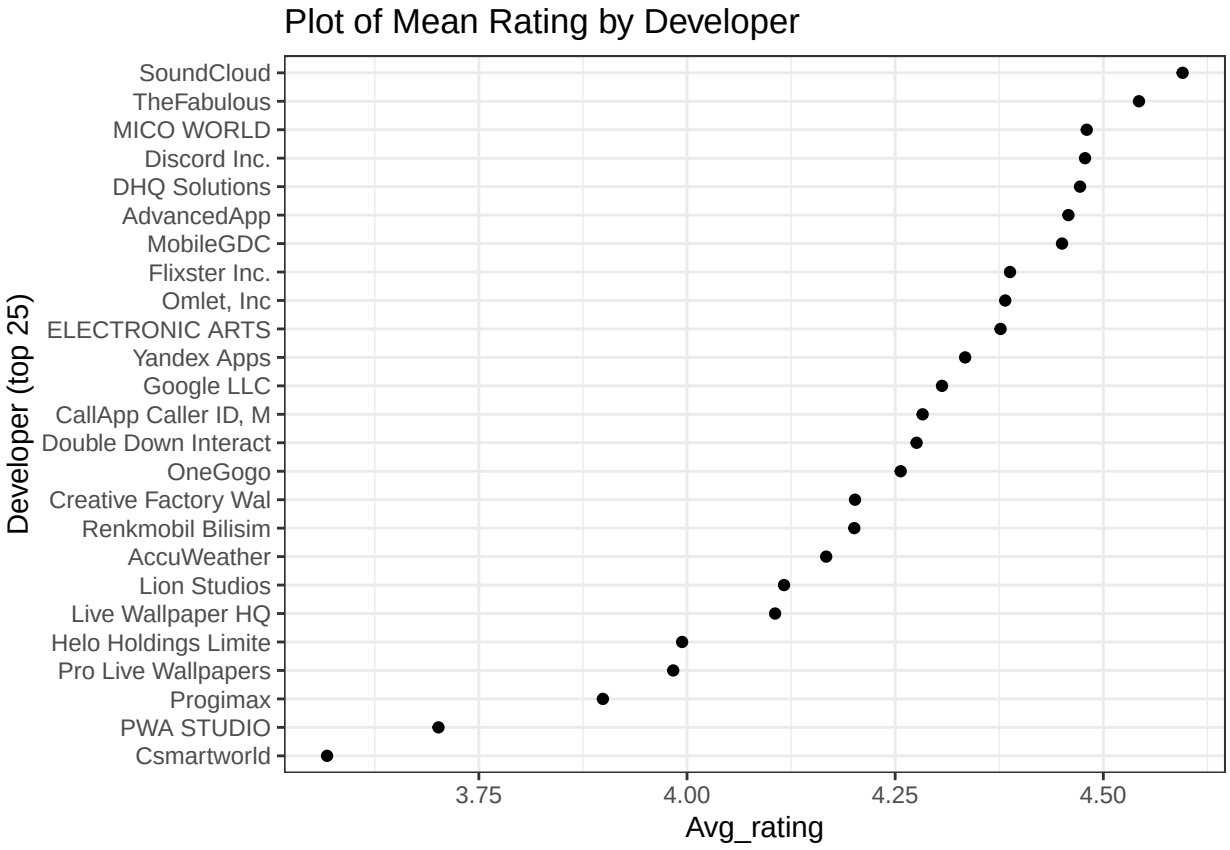
```
## # A tibble: 40,393 x 5
##    Developer       Num_of_App sum_rating sum_ratecount Avg_rating
##    <chr>                <int>      <dbl>         <dbl>      <dbl>
##  1 "EduGorilla Te~        125      8331.          1991       4.18
##  2 "Colorful Desi~         92    149163.         35545       4.20
##  3 "livephoto"             90     54130.         12571       4.31
##  4 "Luklek"                90    102304.         24080       4.25
##  5 "APPRADIO.PRO ~         83     30951.          6782       4.56
##  6 "ajazgames"             79     13215.          3370       3.92
##  7 "Wave Keyboard~         79    181993.         41246       4.41
##  8 "MobiSystems"           76    222889.         53832       4.14
##  9 "Amino Apps"            74    410322.         90041       4.56
## 10 "2021 keyboard~         71    117554.         25065       4.69
## # ... with 40,383 more rows
```

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


Plot of Number of Apps by Developer
Developer (top 25)
EduGorilla TestSerie
Colorful Design
Luklek
livephoto
APPRADIO.PRO − Radio
Wave Keyboard Design
ajazgames
MobiSystems
Amino Apps
Simya Solutions Ltd.
2021 keyboard Theme
Hottest theme keyboa
Softecks
HistoryofTheWorld
Edugorilla Test Seri
AMSApps
Games2Jolly
Sound Jabber
Pretty Emoji Keyboar
Wasabi
Radio Stations FM −
HD Live Wallpapers a
Color Emoji Keyboard
Big Fish Games
Subsplash Inc
New 2021 Themes for
App Basic
0
40
80
120
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?
```

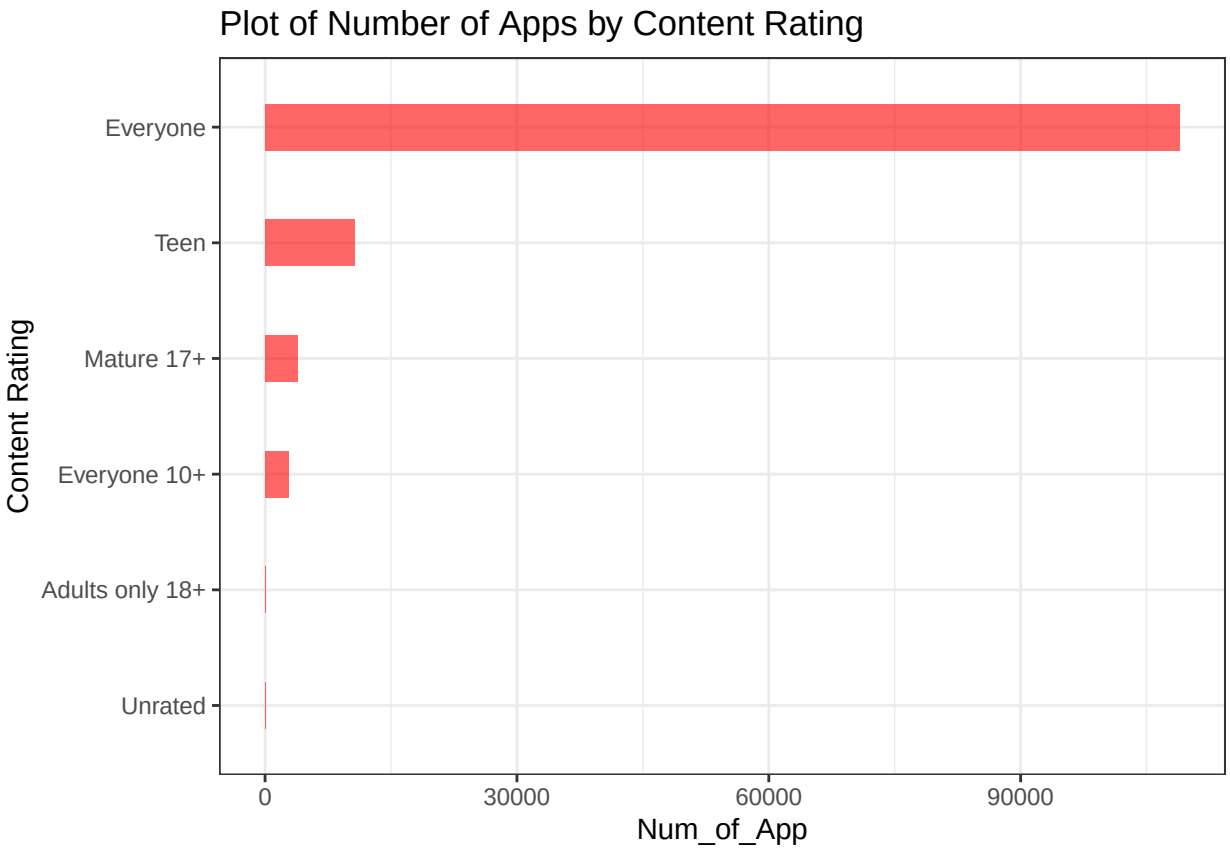


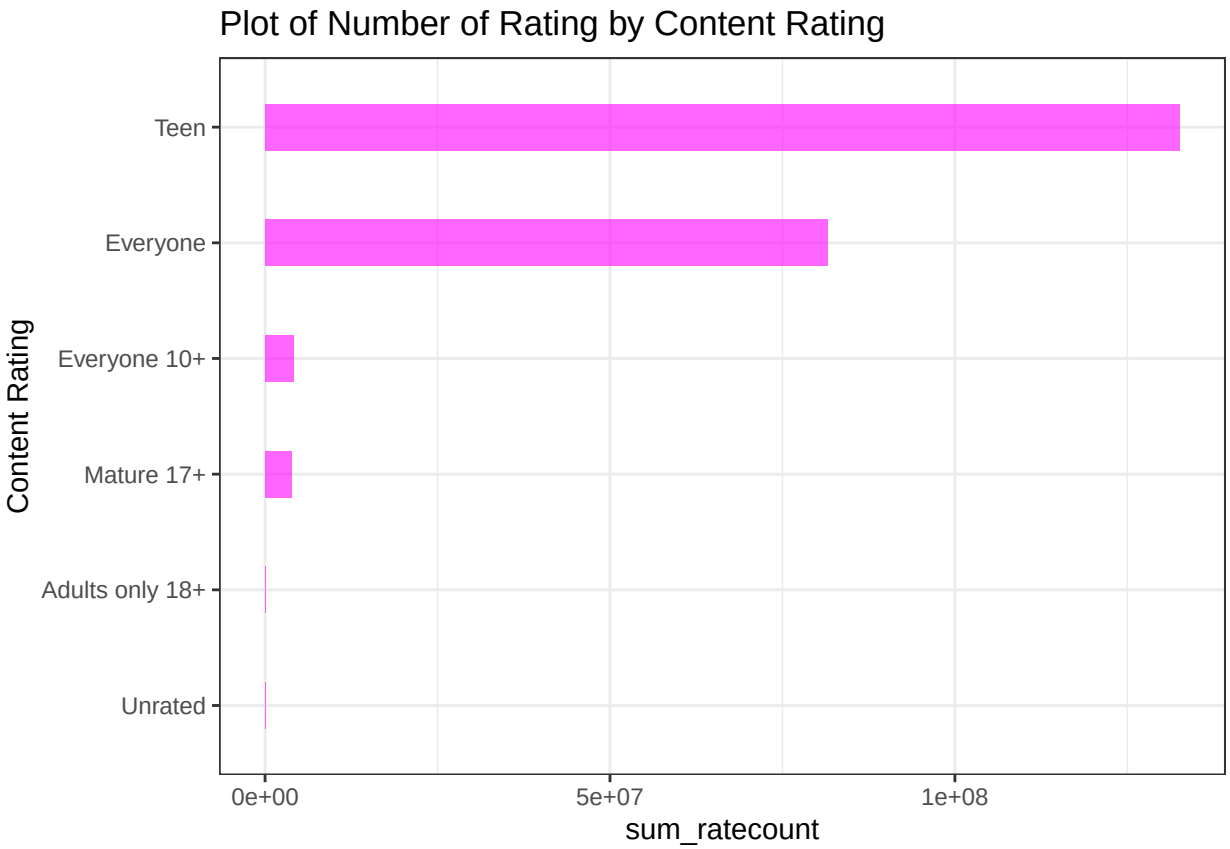
Content Rating

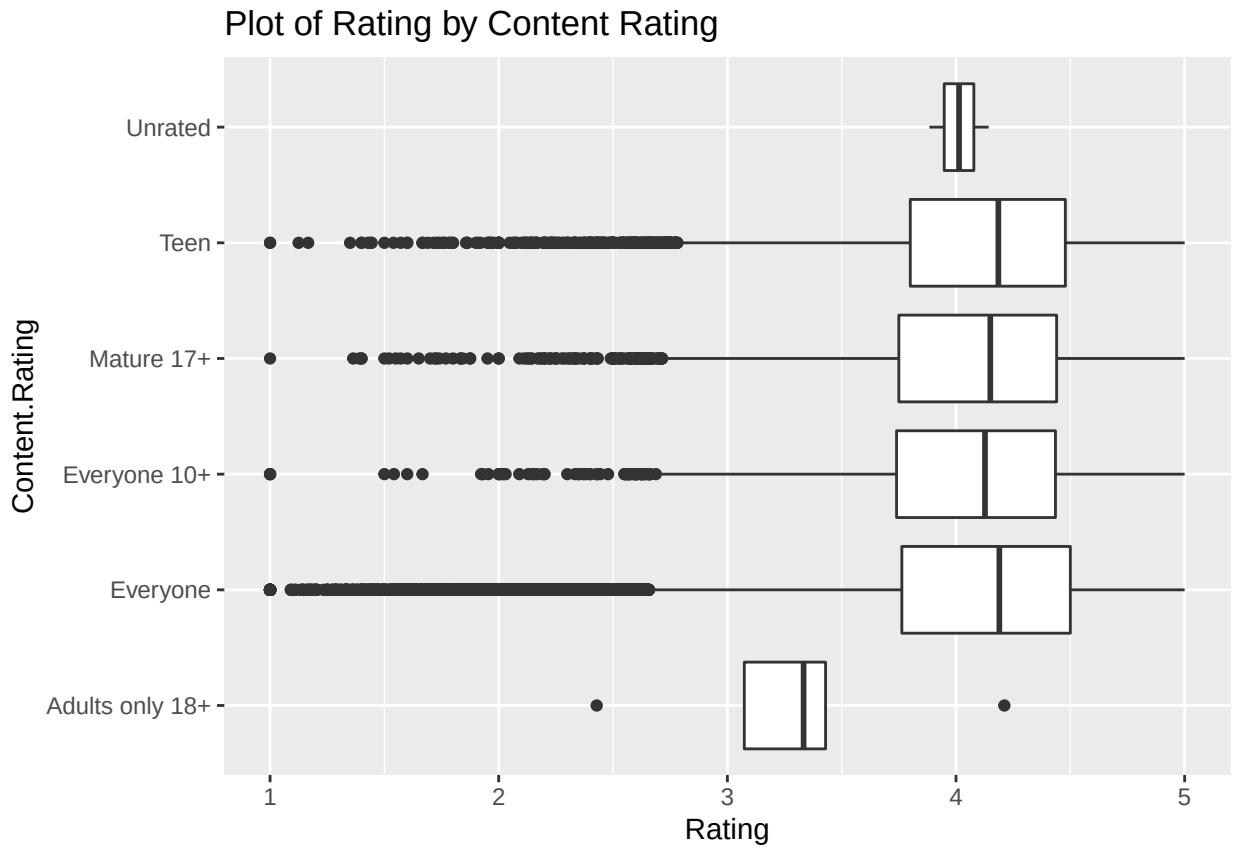
Number of ratings and mean rating by Content Rating in the store dataset:

```
## # A tibble: 6 x 5
##   Content.Rating  Num_of_App sum_rating sum_ratecount Avg_rating
##   <chr>                <int>      <dbl>         <dbl>      <dbl>
## 1 Everyone            108941 341115882.      81538741       4.18
## 2 Teen                 10672 571010826.     132581176       4.31
## 3 Mature 17+            3883  16058407.       3793637       4.23
## 4 Everyone 10+          2824  17342061.       4097522       4.23
## 5 Adults only 18+          5      5002.          1452       3.45
## 6 Unrated                  2       363.            93       3.90
```

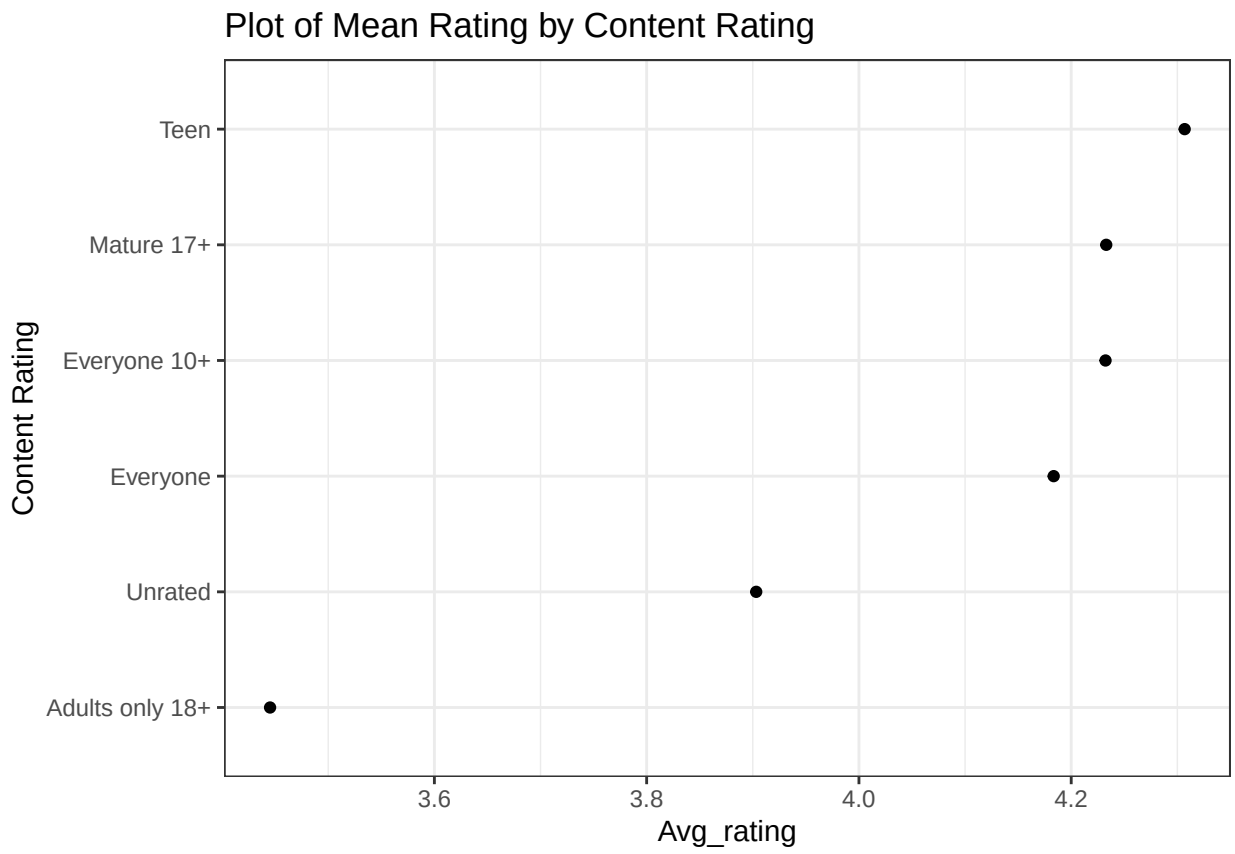
Plot of number of apps, number of ratings, ratings and mean rating by Content Rating:


Plot of Number of Apps by Content Rating
Content Rating
Everyone
Teen
Mature 17+
Everyone 10+
Adults only 18+
Unrated
0
30000
60000
90000
Num_of_App


Plot of Number of Rating by Content Rating
Content Rating
Teen
Everyone
Everyone 10+
Mature 17+
Adults only 18+
Unrated
0e+00
5e+07
1e+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?
```



RMSE

##Mean rating :

```
## [1] 4.258913
```

##Predict using :

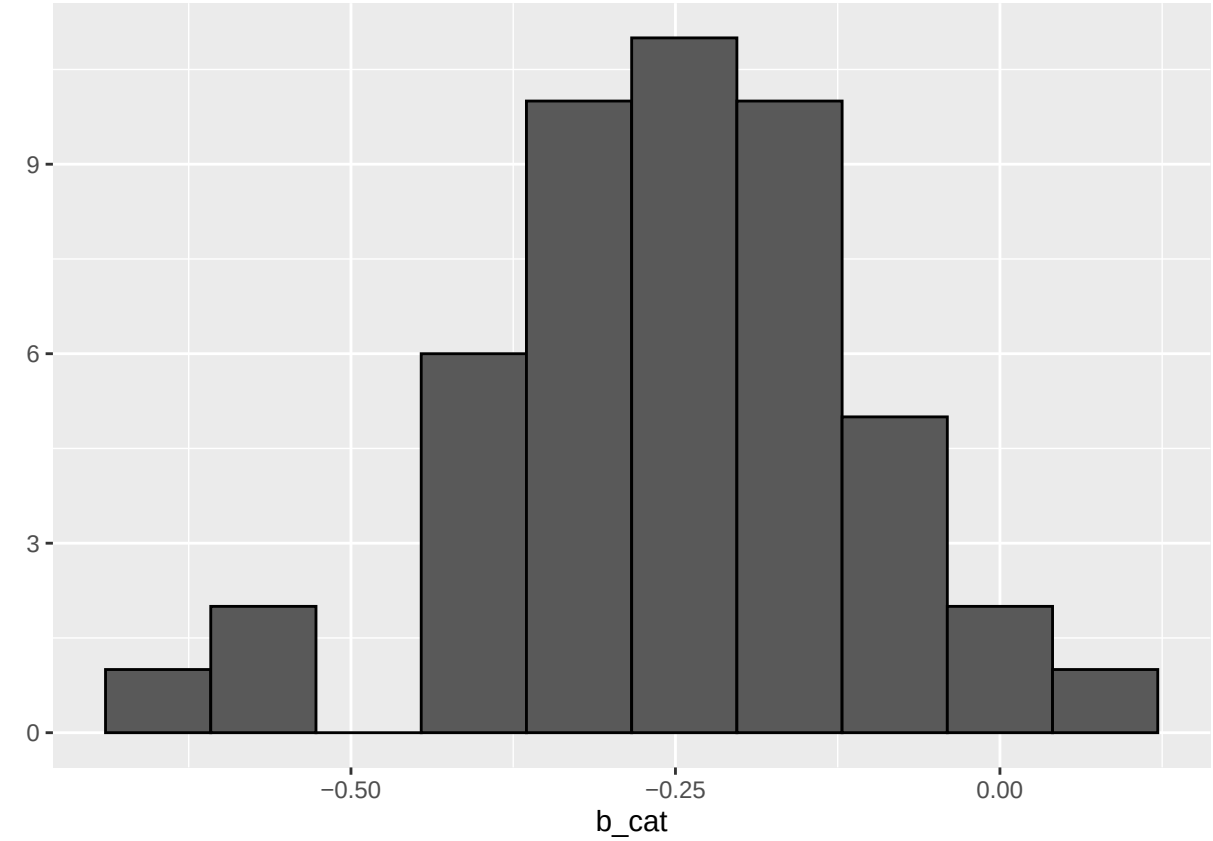
##1. Mean rating

```
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	0.6464709

##2. Mean rating and Category effect :

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

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

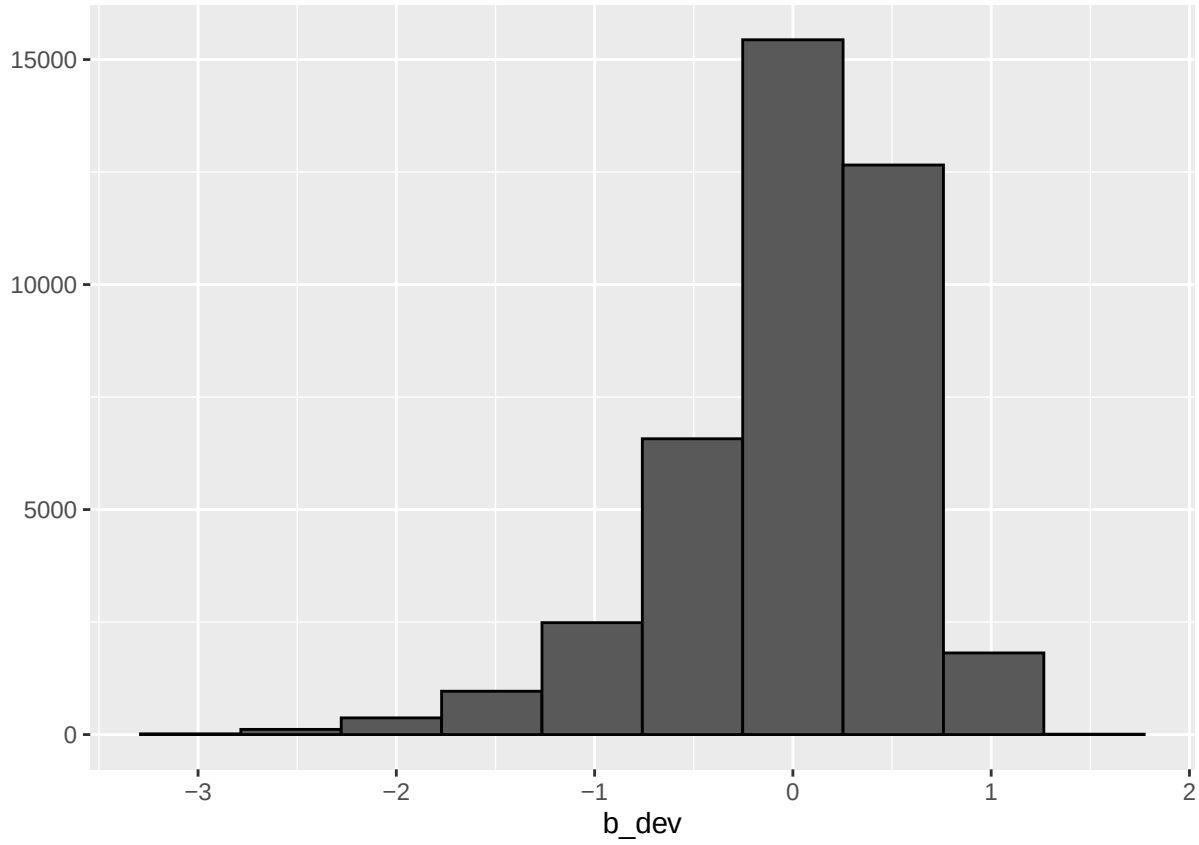



```
predicted_ratings <- store_mean + validation %>%
    left_join(Category_avgs, by = "Category") %>%
    .$b_cat

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	0.6464709
Category Effect Model	0.5974575

##3. Category + Developer.Id effect :

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



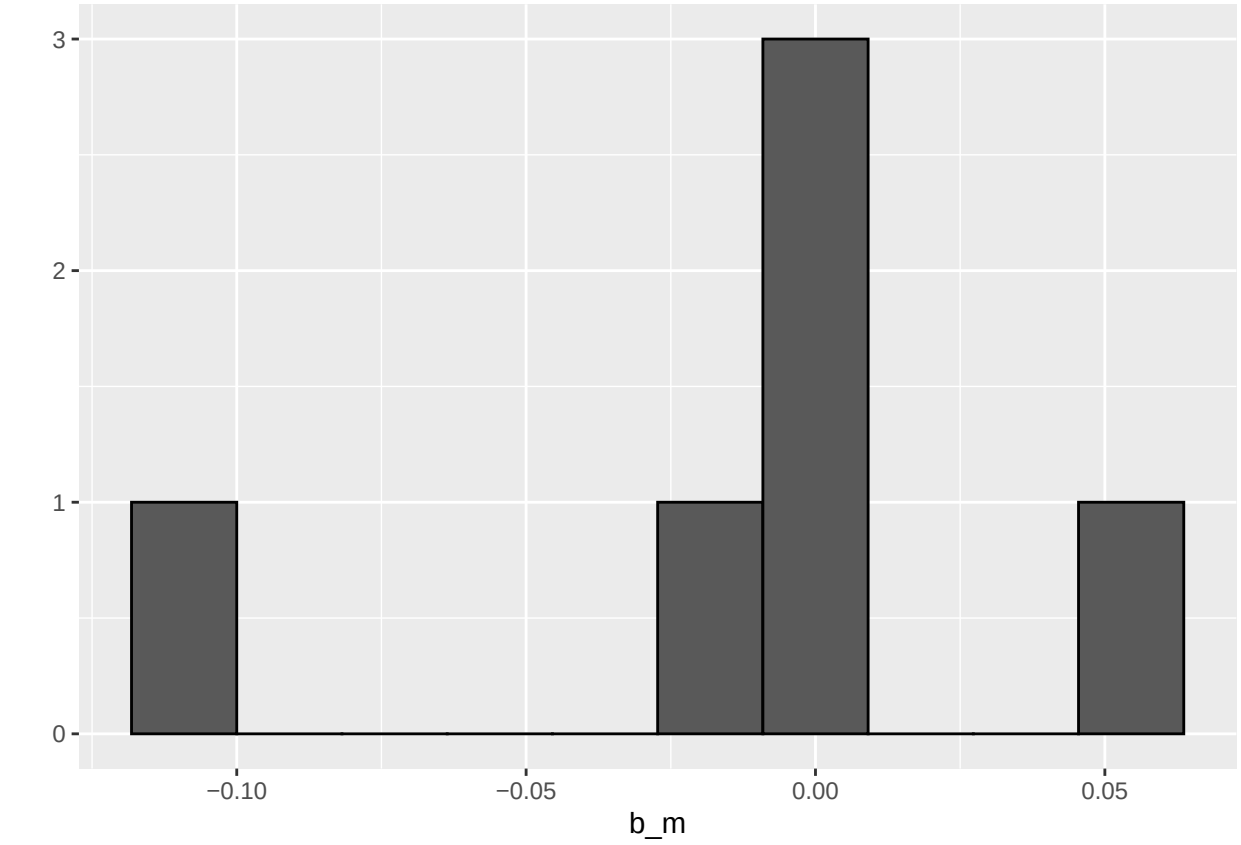

```
predicted_ratings <- validation %>%
    select(Category, Developer.Id) %>%
    left_join(Category_avgs, by = "Category") %>%
    left_join(Developer.Id_avgs, by = "Developer.Id") %>%
    mutate(pred = store_mean + b_cat + b_dev) %>%
    .$pred

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	0.6464709
Category Effect Model	0.5974575

method	RMSE
Category + Developer.Id Effects Model	0.5397790

##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_cat -
        b_dev))
Content.Rating_avgs %>%
    qplot(b_m, geom = "histogram", bins = 10, data = ., color = I("black"))
```



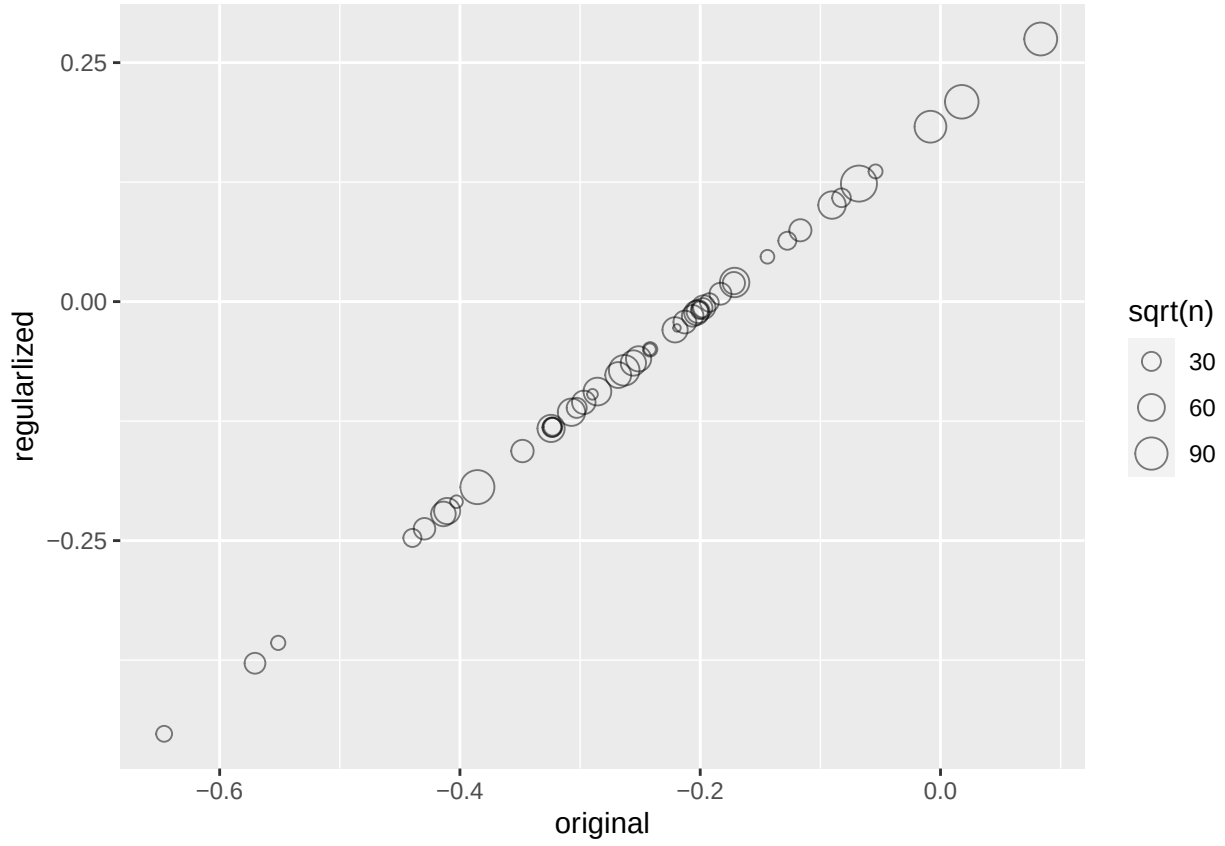

```
predicted_ratings <- validation %>%
    select(Category, Developer.Id, Content.Rating) %>%
    left_join(Category_avgs, by = "Category") %>%
    left_join(Developer.Id_avgs, by = "Developer.Id") %>%
    left_join(Content.Rating_avgs, by = "Content.Rating") %>%
    mutate(pred = store_mean + b_cat + b_dev + 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	0.6464709
Category Effect Model	0.5974575
Category + Developer.Id Effects Model	0.5397790
Category + Developer.Id + Content.Rating Effects Model	0.5397401

##5. Regularized Category Effect

```
lambda <- 3
mu <- mean(as.numeric(store$Rating))
Category_reg_avgs <- store %>%
    group_by(Category) %>%
    summarize(b_cat = sum(as.numeric(Rating) - mu)/(n() + lambda),
        n_i = n())
data_frame(original = Category_avgs$b_cat, regularlized = Category_reg_avgs$b_cat,
    n = Category_reg_avgs$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_cat)) %>%
    select(b_cat, n) %>%
    slice(1:10) %>%
    knitr::kable()
```

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

b_cat	n
0.2747126	8755
0.2090294	9591
0.1829474	7497
0.1361713	354
0.1234750	13177
0.1086045	846
0.1010218	3951
0.0746154	1615
0.0636468	739
0.0469316	338

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

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

b_cat	n
-0.4520900	46
-0.3782982	125
-0.3569692	29
-0.2471792	63
-0.2376086	120
-0.2220965	255
-0.2190477	296
-0.2092736	27
-0.1940245	890
-0.1562924	139

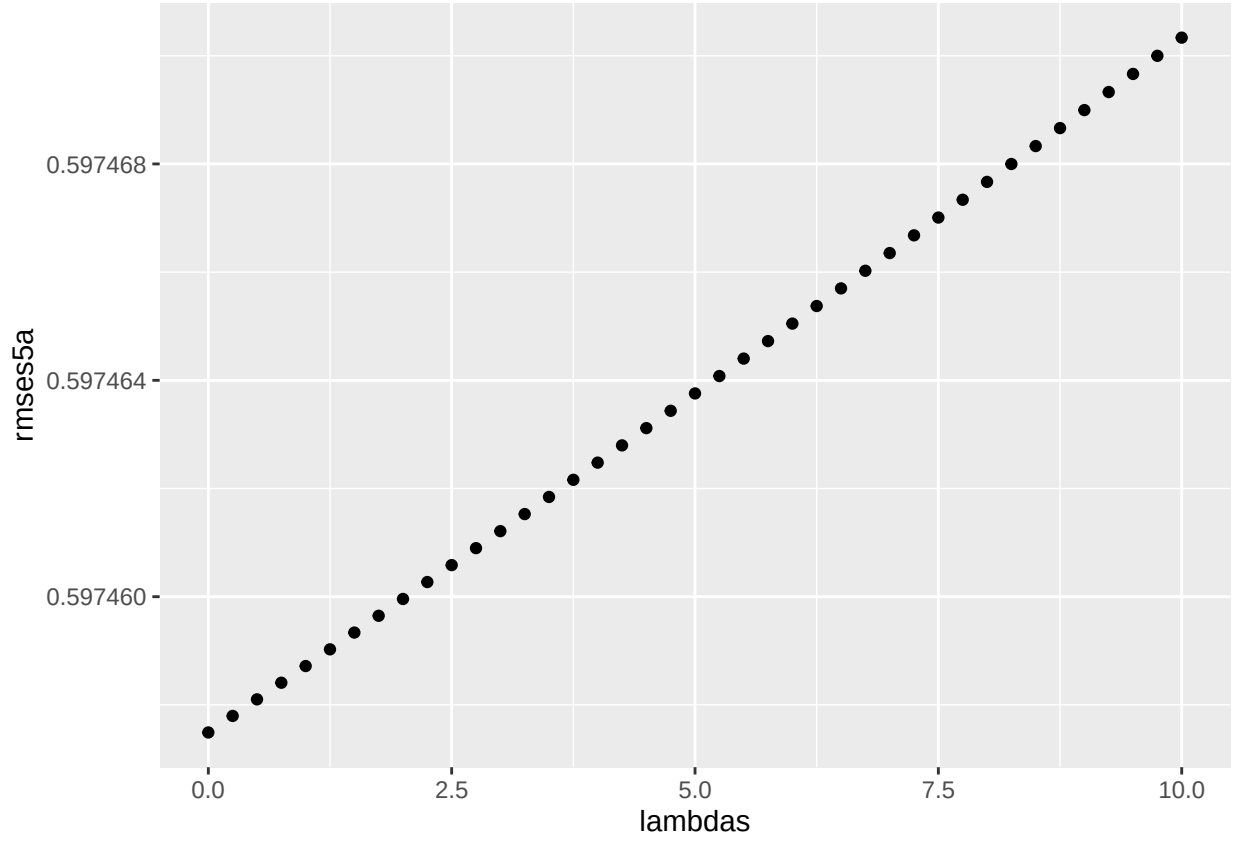
```
predicted_Ratings <- validation %>%
    left_join(Category_reg_avgs, by = "Category") %>%
    mutate(pred = mu + b_cat) %>%
    .$pred
model_5_rmse <- RMSE(predicted_Ratings, as.numeric(validation$Rating))
rmse_results5 <- bind_rows(rmse_results, data_frame(method = "Regularized Category Effect Model",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

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

method	RMSE
Using Mean Rating	0.6464709
Regularized Category Effect Model	0.5974612

```
rm(Category_reg_avgs)
```

##6. optimise lamdas for Category effect

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




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

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

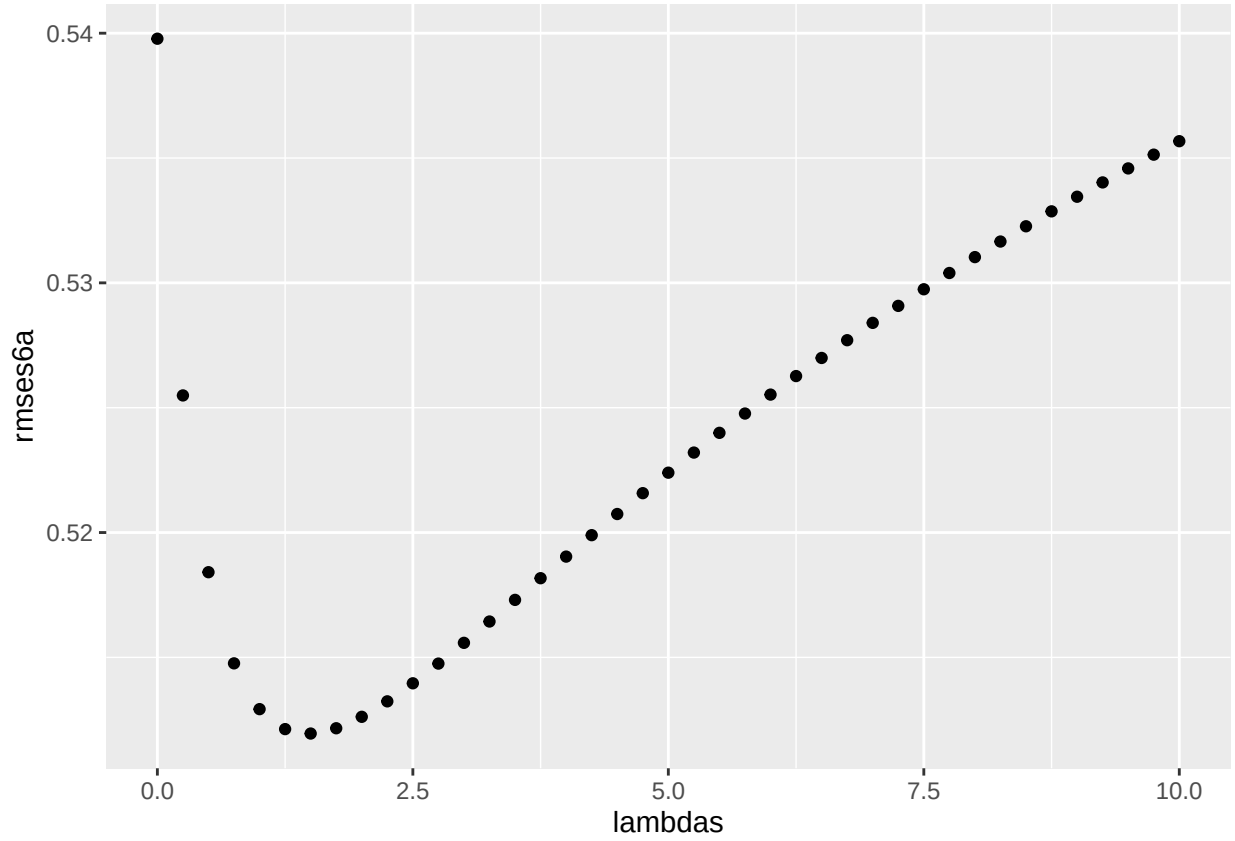
```
rmses5 <- validation %>%
    left_join(just_the_sum, by = "Category") %>%
    mutate(b_cat = s/(n_i + l1)) %>%
    mutate(pred = mu + b_cat) %>%
    .$pred
model_5_rmse <- RMSE(rmses5, as.numeric(validation$Rating))
rmse_results5 <- bind_rows(rmse_results, data_frame(method = "Regularized Category Effect Model",
    RMSE = model_5_rmse))

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

method	RMSE
Using Mean Rating	0.6464709
Regularized Category Effect Model	0.5974575

##7. optimise lambdas for Developer.Id effect

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

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

u1 <- store %>%
    select(Developer.Id, Rating) %>%
    cbind(Category_avgs2e$b_cat) %>%
    set_names("Developer.Id", "Rating", "b_cat") %>%
    group_by(Developer.Id) %>%
    summarize(s = sum(as.numeric(Rating) - mu - b_cat), 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_cat") %>%
        left_join(u1, by = "Developer.Id") %>%
        mutate(b_dev = s/(n_i + l)) %>%
        mutate(pred = mu_b_cat + b_dev) %>%
        .$pred
    return(RMSE(predicted_Ratings, as.numeric(validation$Rating)))
})
```

```
qplot(lambdas, rmses6a)
```




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

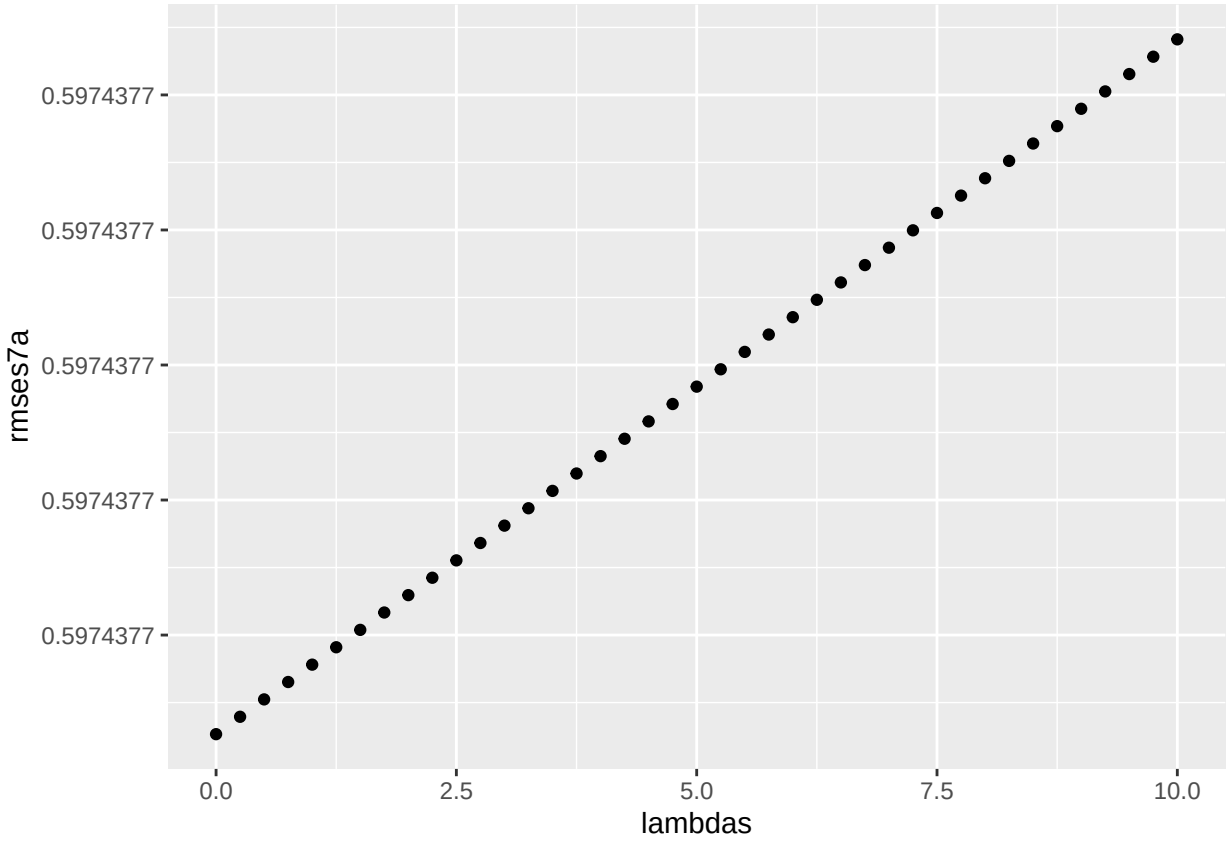
```
## [1] 1.5
```

```
rmses6 <- validation %>%
    select(Developer.Id) %>%
    cbind(rmses5) %>%
    set_names("Developer.Id", "mu_b_cat") %>%
    left_join(u1, by = "Developer.Id") %>%
    mutate(b_dev = s/(n_i + l2)) %>%
    mutate(pred = mu_b_cat + b_dev) %>%
    .$pred
model_6_rmse <- RMSE(rmses6, as.numeric(validation$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	0.6464709
Regularized Category Effect Model	0.5974575
Regularized Developer.Id Effect Model	0.5119498

##8. optimise lambdas for Content.Rating effect

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

Developer.Id_avgs2 <- store %>%
    select(Developer.Id) %>%
    left_join(u1, by = "Developer.Id") %>%
    mutate(b_dev = s/(n_i + l2)) %>%
    select(Developer.Id, b_dev)

rm(u1)

g1 <- store %>%
    select(Content.Rating, Rating) %>%
    cbind(Category_avgs2e$b_cat, Developer.Id_avgs2$b_dev) %>%
    set_names("Content.Rating", "Rating", "b_cat", "b_dev") %>%
    group_by(Content.Rating) %>%
    summarize(s = sum(as.numeric(Rating) - mu - b_cat - b_dev),
        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_cat_bu") %>%
        left_join(g1, by = "Content.Rating") %>%
        mutate(b_CR = s/(n_i + l)) %>%
        mutate(pred = mu_b_cat_bu + b_CR) %>%
        .$pred
    return(RMSE(predicted_Ratings, as.numeric(validation$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_cat_bu") %>%
    left_join(g1, by = "Content.Rating") %>%
    mutate(b_CR = s/(n_i + l3)) %>%
    mutate(pred = mu_b_cat_bu + b_CR) %>%
    .$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()
```

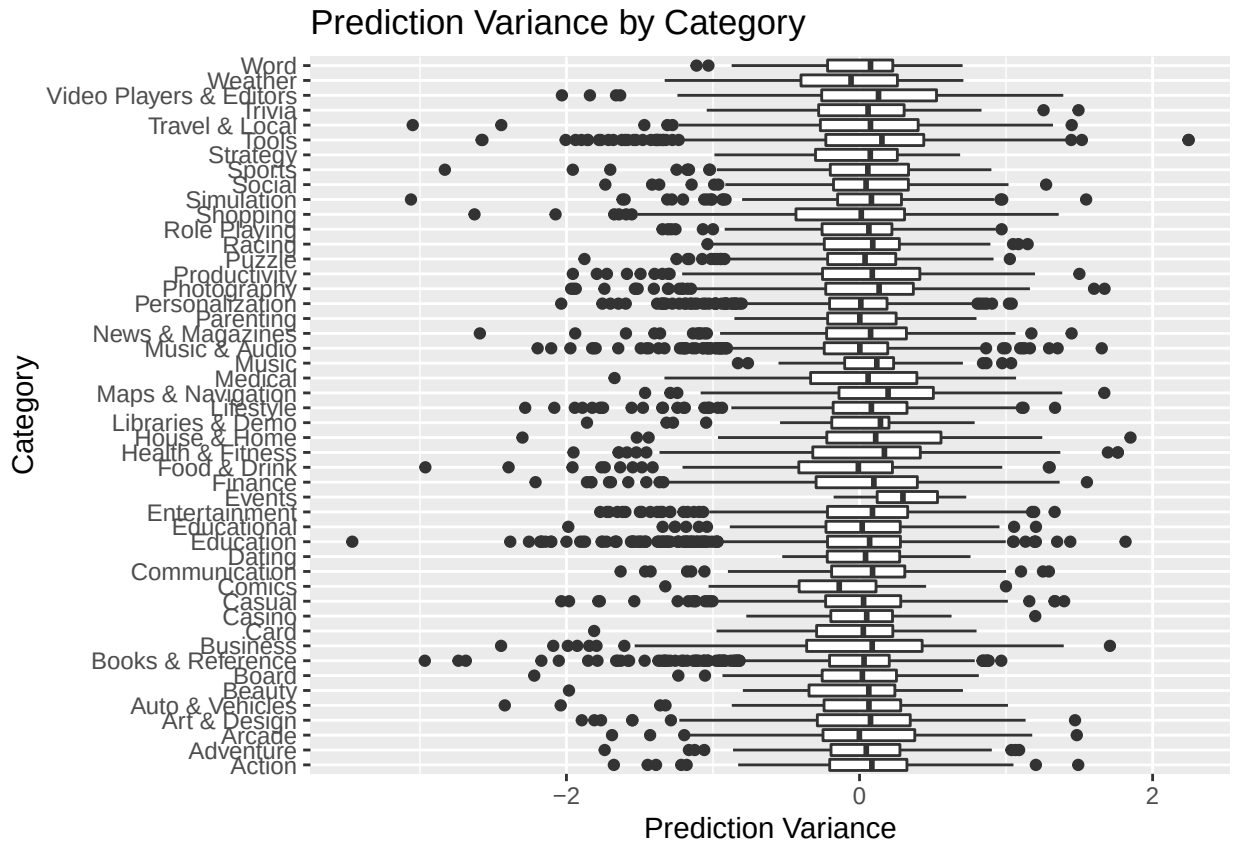
method	RMSE
Using Mean Rating	0.6464709
Regularized Category Effect Model	0.5974575
Regularized Developer.Id Effect Model	0.5119498
Regularized Content.Rating Effect Model	0.5119175

Conclusion

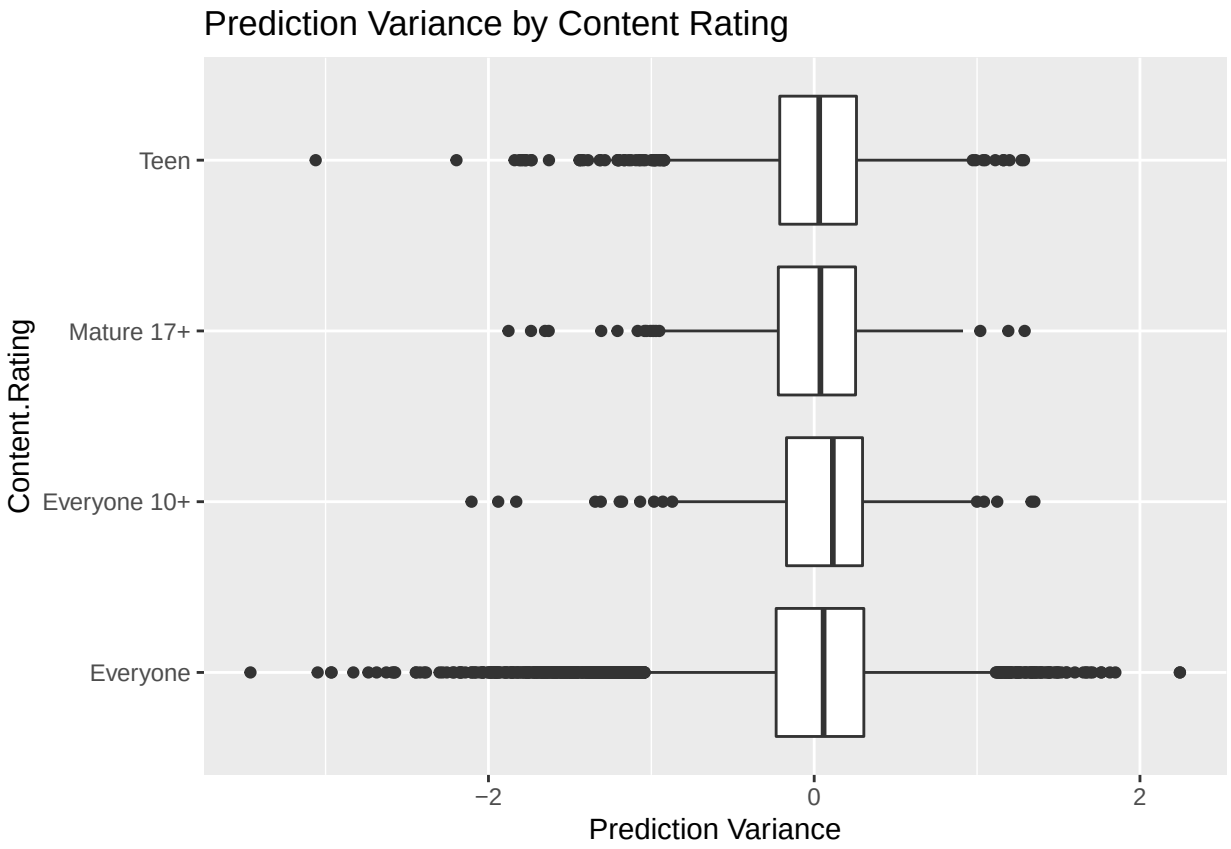
Even though there are many attributes to an applications in the GOOGLE PLAYSTORE dataset, as the dataset already summarised Mean Rating and Rating Count by application, most columns with unique values and TRUE/FALSE values are not useful in prediction (eg. App Name, App Id, Free, In App purchases, Editor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

The Category, DeveloperId and Content.Rating Model is able to reduce the RMSE to 0.5397, while the Regularized Category, DeveloperId and Content.Rating Model is able to reduce the RMSE to 0.5119175, 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