R Notebook

$4. Recommendation System $ We attempt to create a recommendation system that will recommend books given a user based on collaborative filtering method.

Data Exploration and pre processing

I used the goodreads data set which only had meta data about books, to see whether those could help me in making the recommending system. However, I quickly found out that these variables will not help me and don't significantly influence ratings of a book and won't help in making the recommendation engine. At some point, trying to make a model with that dataset did not work.

I also looked at how netflix does its recommending, and discovered it uses a technique called collaborative filtering (Netflix Recommendation Engine, n.d.). In this model, a probability model is used that scores the relationship between the user and movie type to predict preferences. For this reason I downloaded more goodreads data which has user ratings as well. This(Foxtrot, 2018) dataset has over 53.4k user ratings, books, genre and other book related information.

```
library(data.table)
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────────────────
──────────── tidyverse 1.3.0 ──
```

```
## ✓ ggplot2 3.3.0     ✓ purrr   0.3.3
## ✓ tibble  3.0.1     ✓ dplyr   0.8.5
## ✓ tidyr   1.0.0     ✓ stringr 1.4.0
## ✓ readr   1.3.1     ✓ forcats 0.4.0
```

```
## Warning: package 'tibble' was built under R version 3.6.2
```

```
## ── Conflicts ────────────────────────────────────────────────────────────
─────── tidyverse_conflicts() ──
## x dplyr::between()   masks data.table::between()
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks data.table::first()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
## x purrr::transpose() masks data.table::transpose()
```

```
books10k<-fread("goodbooks-10k/books.csv")
ratings10k<-fread("goodbooks-10k/ratings.csv")
booktags10k<-fread("goodbooks-10k/book_tags.csv")
tags10k<-fread("goodbooks-10k/tags.csv")
head(books10k)
```

id <int>	book_id <int>	best_book_id <int>	work_id <int>	books_count <int>	isbn <chr>	isbn13 <dbl>	authors <chr>
1	2767052	2767052	2792775	272	439023483	9.780439e+12	Suzanne Collins
2	3	3	4640799	491	439554934	9.780440e+12	J.K. Rowling, Mar
3	41865	41865	3212258	226	316015849	9.780316e+12	Stephenie Meyer
4	2657	2657	3275794	487	61120081	9.780061e+12	Harper Lee
5	4671	4671	245494	1356	743273567	9.780743e+12	F. Scott Fitzgerald
6	11870085	11870085	16827462	226	525478817	9.780525e+12	John Green

6 rows | 1-8 of 23 columns

```
head(ratings10k)
```

book_id <int>	user_id <int>	rating <int>
1	314	5
1	439	3
1	588	5
1	1169	4
1	1185	4
1	2077	4

6 rows

```
head(tags10k)
```

tag_id <int>	tag_name <chr>
0	-

1	--1-
2	--10-
3	--12-
4	--122-
5	--166-

6 rows

Pre processing to clean data.

Data contains duplicates that need to be removed, ie same user has rated the same book multiple times. I also removed records of users that have only rated less than 3 books, since their ratings are insignificant. Their information will not influence the recommendation system.

Since this data set is large, I chose to use only 40% of the user data to create the system.

book_id <int>	user_id <int>	rating <int>	N <int>
1	314	5	1
1	439	3	1
1	588	5	1
1	1169	4	1
1	1185	4	1
1	2077	4	1
1	2487	4	1
1	2900	5	1
1	3662	4	1
1	3922	5	1

1-10 of 10,000 rows Previous **1** 2 3 4 5 6 … 1000 Next

```
## [1] "Number of ratings (before): 960595 "
```

```
## [1] "Number of ratings (after): 389621 "
```

Exploration

Distribution of ratings, mean user ratings, etc were explored. Those that rate frequently, do they rate differently from less frequent users ? We can explore this by seeing the next plot. Results: The plots showed books are rated 4-5. I saw how users rate, ie number of ratings per user. Then, the ones that rate more, how differently do they rate? I found that frequent users give more lower ratings, could either be they critique more as they read, etc.

```
library(grid)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

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

```
library(corrplot)
```

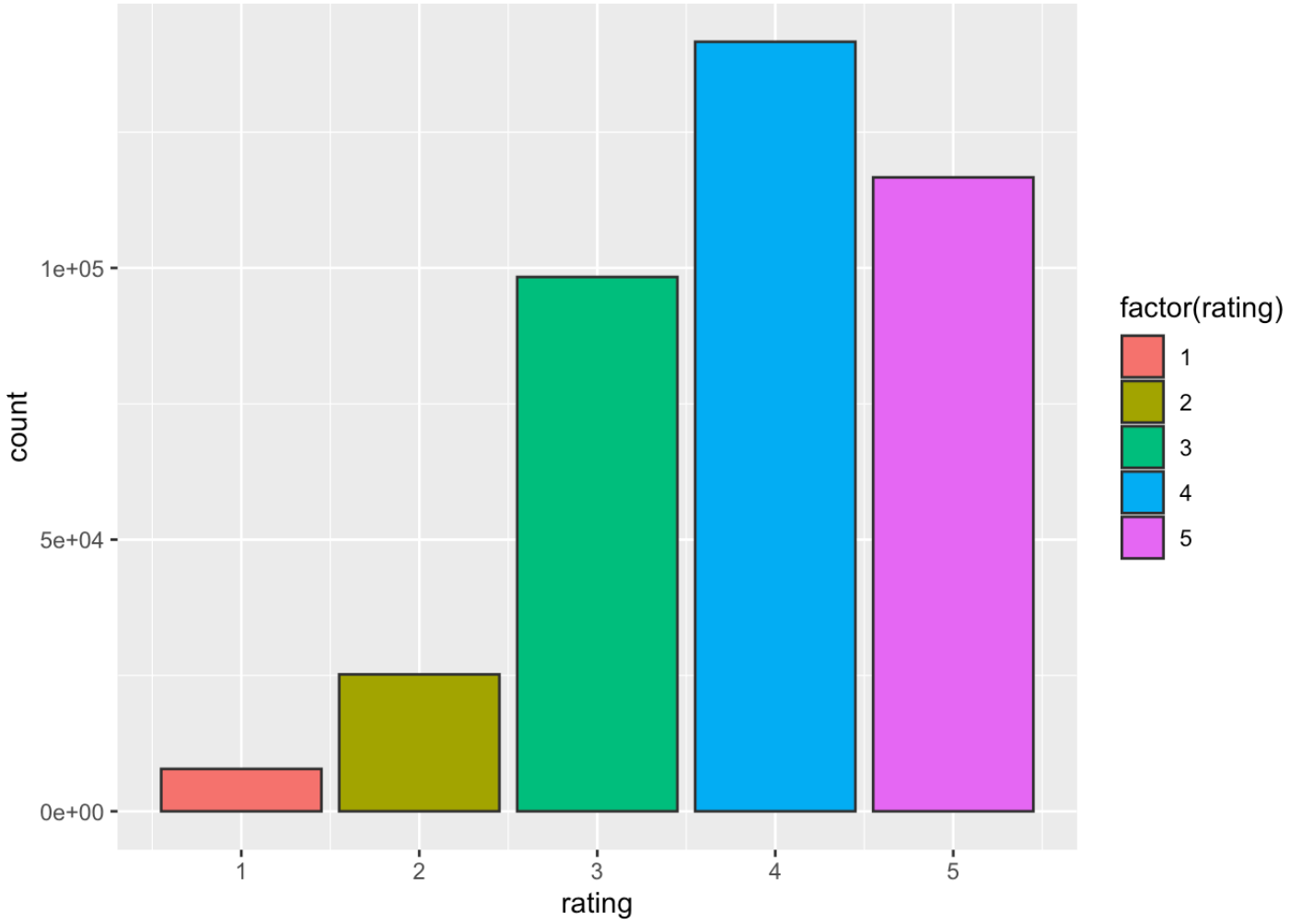
```
## corrplot 0.84 loaded
```

```
library(methods)
library(Matrix)
```

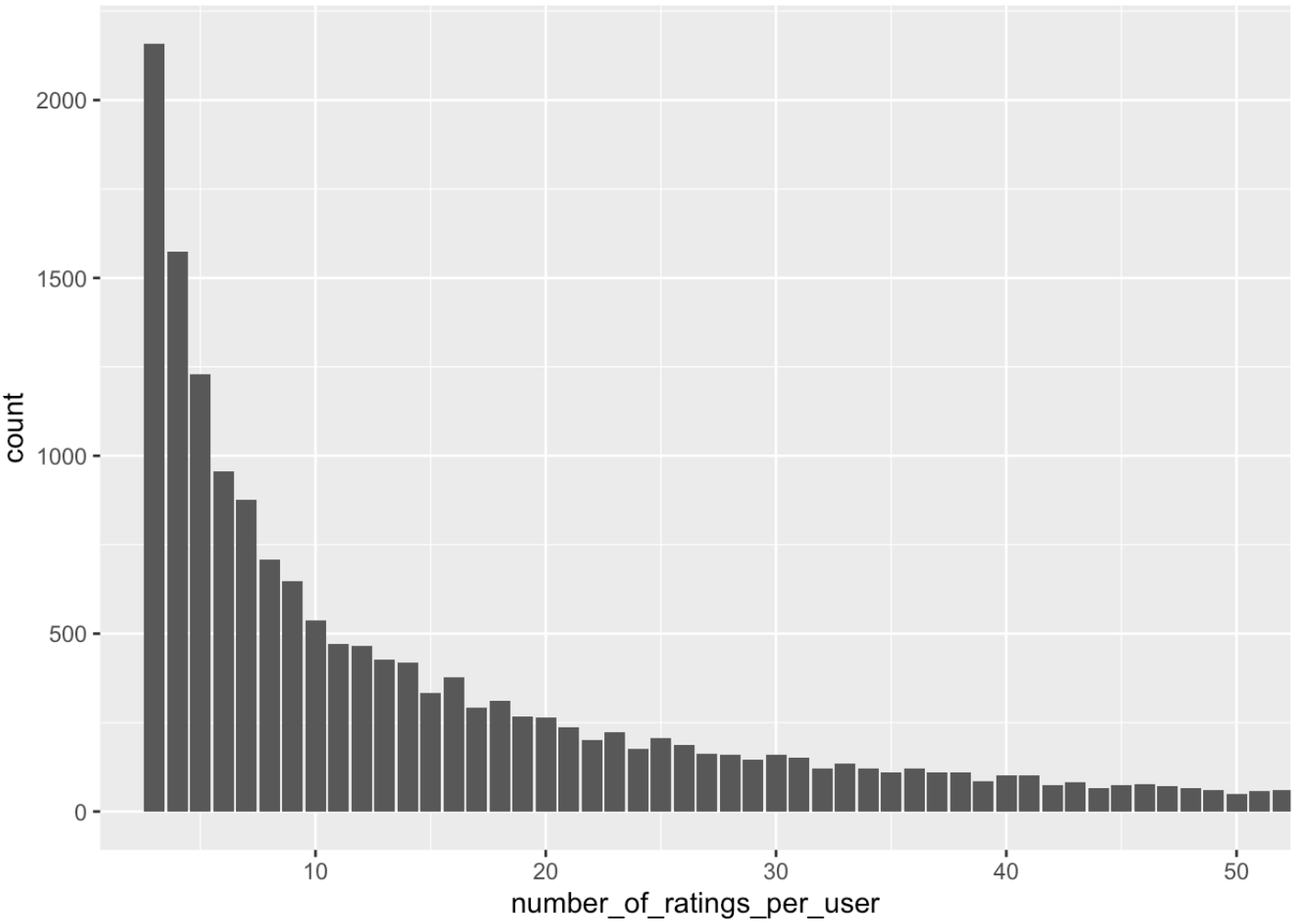
```
##
## Attaching package: 'Matrix'
```

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

```
#how are ratings distributed
ratings10k %>%
  ggplot(aes(x = rating, fill = factor(rating))) +
  geom_bar(color = "grey20")
```




```
# No of ratings per user
ratings10k %>%
  group_by(user_id) %>%
  summarize(number_of_ratings_per_user = n()) %>%
  ggplot(aes(number_of_ratings_per_user)) +
  geom_bar() + coord_cartesian(c(3, 50))
```

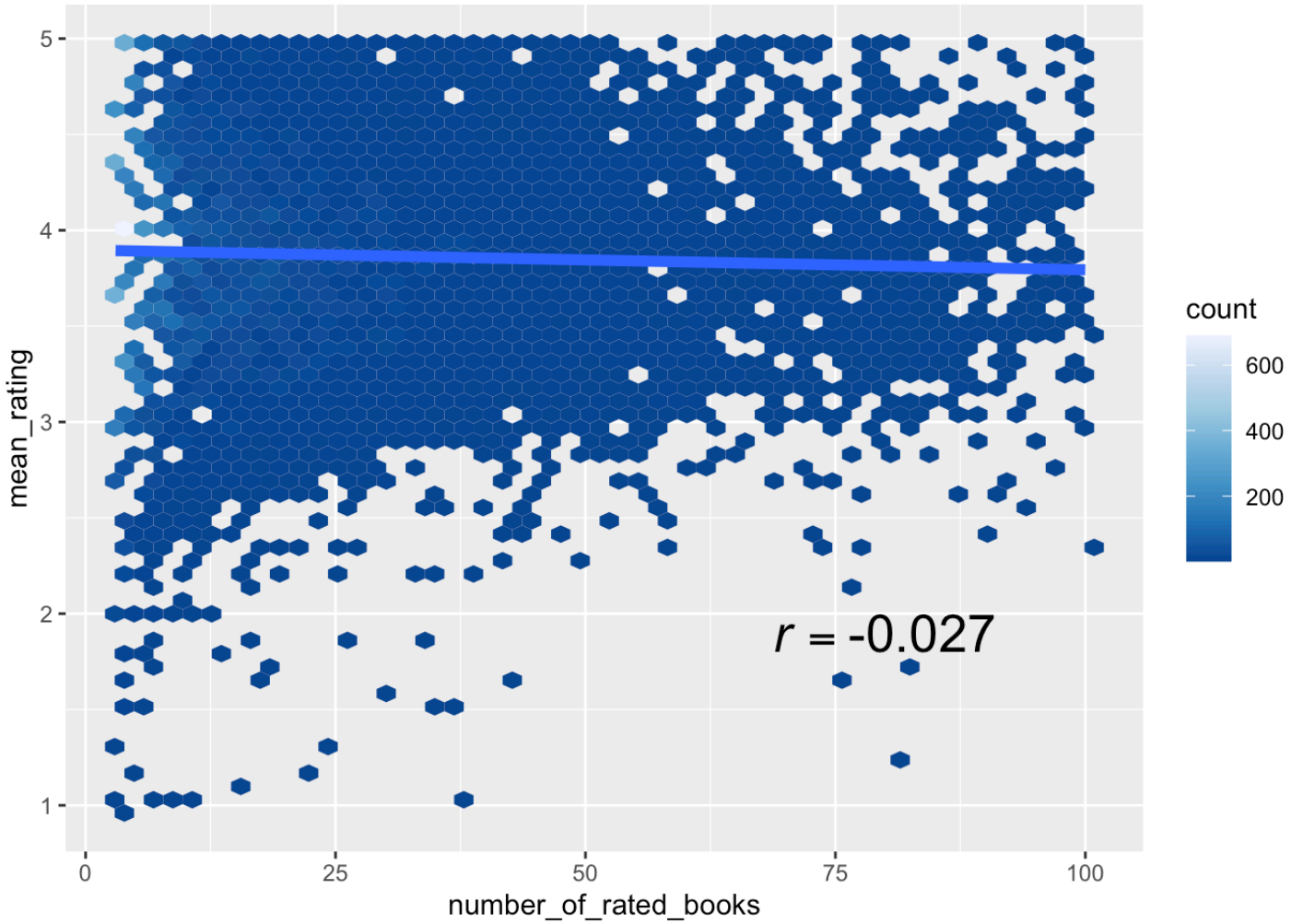



```
# view how users rate and their frequency
get_cor <- function(df){
    m <- cor(df$x,df$y, use="pairwise.complete.obs");
    eq <- substitute(italic(r) == cor, list(cor = format(m, digits = 2)))
    as.character(as.expression(eq));
}

#How do users rate according to their frequency?
tmp <- ratings10k %>%
  group_by(user_id) %>%
  summarize(mean_rating = mean(rating), number_of_rated_books = n())

tmp %>% filter(number_of_rated_books <= 100) %>%
  ggplot(aes(number_of_rated_books, mean_rating)) + stat_bin_hex(bins = 50) + scale_f
ill_distiller() + stat_smooth(method = "lm", size = 2, se = FALSE) +
  annotate("text", x = 80, y = 1.9, label = get_cor(data.frame(x = tmp$number_of_rate
d_books, y = tmp$mean_rating)), size = 7, parse = TRUE)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



How are genres distributed?

Since there are user assigned tags and genres, I am choosing those tags that match good-reads built in genres. Since, looking at the tags, some tags do not make sense like some tags are numeric that do not make sense and will not help in recommending.

Therefore I made a list of most common genre tags and then removed some generic genres like fiction, non fiction, ebooks, etc.

I then looked at how genres are distributed.

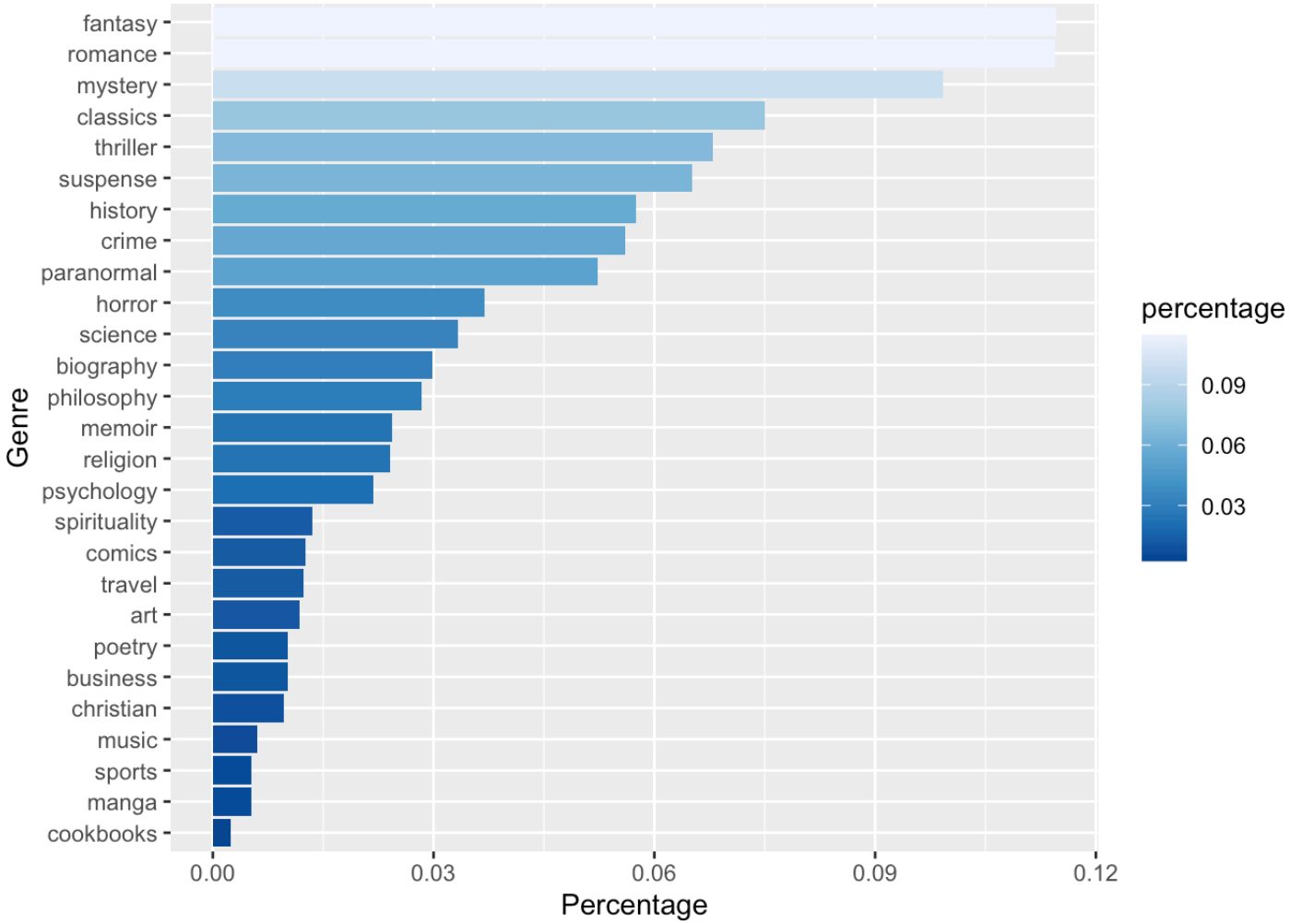
```
#selecting only goodreads genre tags
genres <- str_to_lower(c("Art", "Biography", "Business", "Chick Lit", "Children's", "
Christian", "Classics", "Comics", "Contemporary", "Cookbooks", "Crime", "Ebooks", "Fa
ntasy", "Fiction", "Gay and Lesbian", "Graphic Novels", "Historical Fiction", "Histor
y", "Horror", "Humor and Comedy", "Manga", "Memoir", "Music", "Mystery", "Nonfiction"
, "Paranormal", "Philosophy", "Poetry", "Psychology", "Religion", "Romance", "Science
", "Science Fiction", "Self Help", "Suspense", "Spirituality", "Sports", "Thriller",
"Travel", "Young Adult"))

exclude_genres <- c("fiction", "nonfiction", "ebooks", "contemporary")
genres <- setdiff(genres, exclude_genres)

available_genres <- genres[str_to_lower(genres) %in% tags10k$tag_name]
available_tags <- tags10k$tag_id[match(available_genres, tags10k$tag_name)]

#Distribution of genres
genredist <- booktags10k %>%
  filter(tag_id %in% available_tags) %>%
  group_by(tag_id) %>%
  summarize(n = n()) %>%
  ungroup() %>%
  mutate(sumN = sum(n), percentage = n / sumN) %>%
  arrange(-percentage) %>%
  left_join(tags10k, by = "tag_id")

genredist %>%
  ggplot(aes(reorder(tag_name, percentage), percentage, fill = percentage)) + geom_ba
r(stat = "identity") + coord_flip() + scale_fill_distiller() + labs(y = 'Percentage',
x = 'Genre')
```



This shows that most books fall under the fantasy, romance and mystery genre. When a user first joins Netflix and has not rated any shows or movies so far, Netflix just recommends the most popular or trending shows/ movies. A similar thing could be done looking at this genre information.

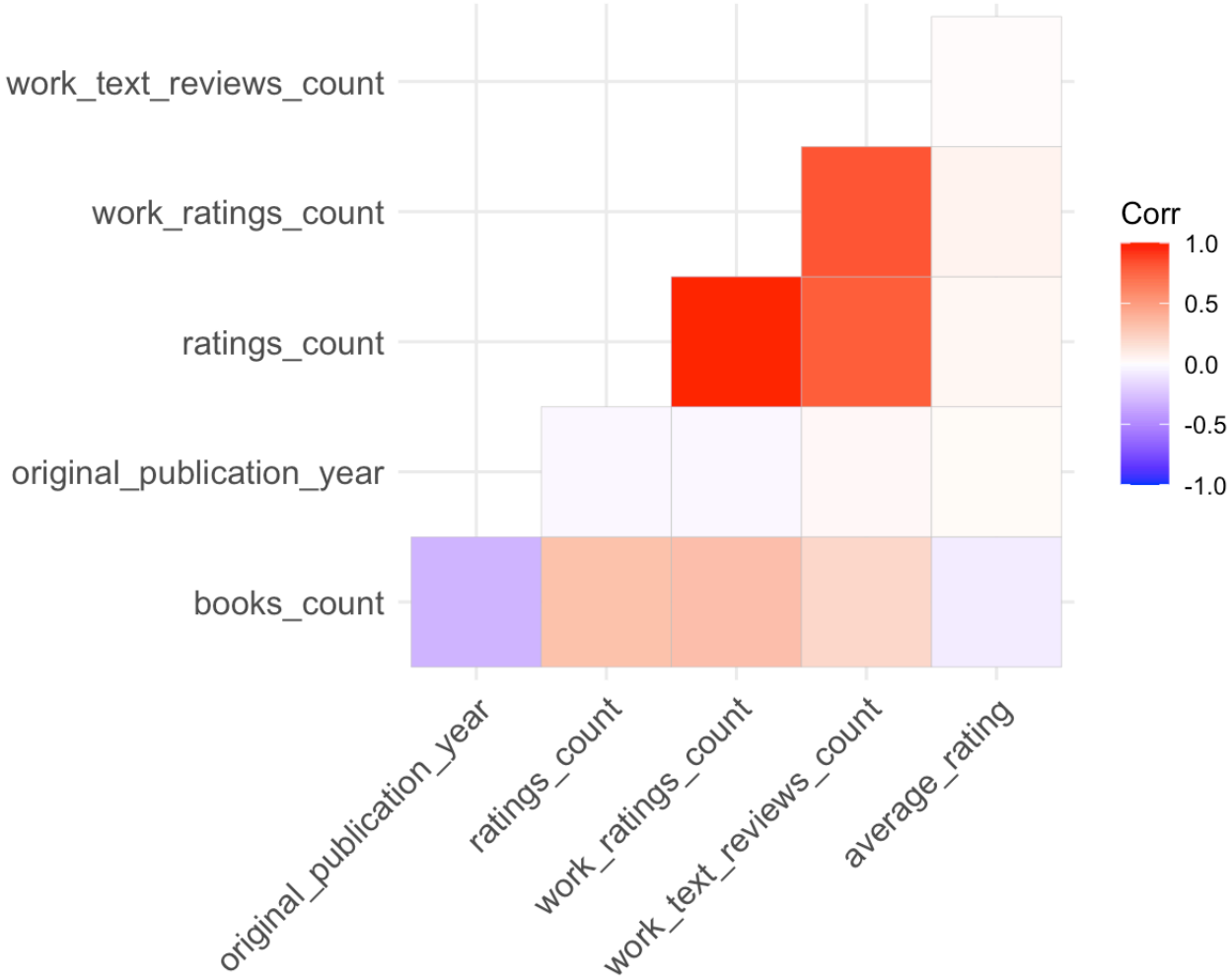
What influences a book's rating?

Correlation plot reveals no strong relations between rating and rating count, text review count etc. Hence other factors might affect it.

```
library(ggcorrplot)

cormatrix <- books10k %>%
  select(one_of(c("books_count","original_publication_year","ratings_count", "work_ratings_count", "work_text_reviews_count", "average_rating"))) %>%
  as.matrix()

ggcorrplot(cor(cormatrix, use = 'pairwise.complete.obs'), type = "lower")
```



Collaborative filtering

From the EDA it looks like effects of meta data type variables like ratings count is insignificant, and other quality aspects matter more.

We use the aforementioned user-based collaborative filtering method to make recommendations. It is a popular technique used by Netflix for recommending movies. The way it works is very intuitivw; when thinking of what book to read next, you are most likely going to ask that one friend who has similar taste as you to recommend some to you. Hence, similarly, we will first find similar users to a chosen current user in terms of their ratings on the same set of books. Then, we will find avg ratings for books rated by those similar users that the current user has not yet read. In theory, the highest avg rated books can be good recommendations to that chosen user.

Pre-processing before creating recommendation engine

We need to create a sort of utility matrix on books rated by users so that we know which users have rated which books.

```
dimension_names <- list(user_id = sort(unique(ratings10k$user_id)), book_id = sort(un
ique(ratings10k$book_id)))

ratingmat <- spread(select(ratings10k, book_id, user_id, rating), book_id, rating) %>
% select(-user_id)

ratingmat <- as.matrix(ratingmat)
dimnames(ratingmat) <- dimension_names
ratingmat[1:5, 1:5]
```

```
##         book_id
## user_id  1  2  3  4  5
##       8  NA NA NA NA NA
##       11 NA NA NA NA NA
##       16 NA NA NA NA NA
##       18 NA NA NA NA NA
##       19 NA NA NA NA NA
```

```
rt<-as.data.frame(ratingmat)
```

Find similar users

Lets select a random user id 173. We find users that rated atleast one book rated by this user, lets call the user Bob. For this user, there are over 700 similar users. We find similarity with Bob's ratings. We can use Pearson's correlation coefficient. We sort these according to highest similarity. we now have most similar users.

```
library(igraph)
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
```

```
## The following objects are masked from 'package:purrr':
##
##     compose, simplify
```

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

```
## The following object is masked from 'package:tibble':
##
##     as_data_frame
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
chosen_user<-"173"

rated_items <- which(!is.na((as.data.frame(ratingmat[chosen_user, ]))))
selected_users <- names(which(apply(!is.na(ratingmat[ ,rated_items]), 1, sum) >= 2))
head(selected_users,40)
```

```
##  [1] "35"   "75"   "173"  "230"  "314"  "343"  "368"  "430"  "446"  "725"
## [11] "794"  "907"  "946"  "951"  "1026" "1027" "1038" "1041" "1088" "1127"
## [21] "1186" "1255" "1260" "1294" "1303" "1369" "1407" "1431" "1440" "1456"
## [31] "1484" "1518" "1645" "1713" "1737" "1766" "1813" "1824" "1844" "1853"
```

```
#Normalizing the ratings so as to equalise variance
rmat<-scale(ratingmat[selected_users, ])

# Finding similarity and sorting acc to highest similarity
similarities <- cor(t(rmat[rownames(rmat)!=chosen_user, ]), rmat[chosen_user, ], use
= 'pairwise.complete.obs')
sim <- as.vector(similarities)
names(sim) <- rownames(similarities)
res <- sort(sim, decreasing = TRUE)
head(res, 40)
```

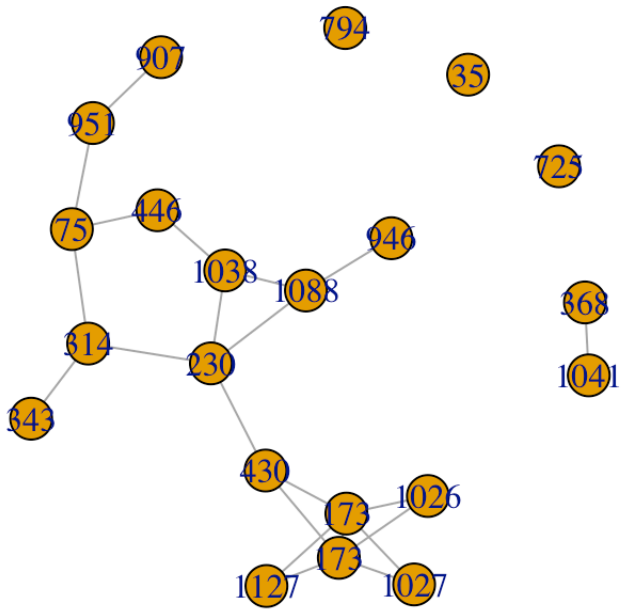
```
##   430  1026  1027  1127  1255  1303  1920  2288  2522  2641  3829  4005
##     1     1     1     1     1     1     1     1     1     1     1     1
##  4212  5443  5551  6728  8167  8951  9474  9909  9971 10008 10097 10402
##     1     1     1     1     1     1     1     1     1     1     1     1
## 10500 10691 10762 11154 11214 11935 12364 12838 12989 13073 14460 15891
##     1     1     1     1     1     1     1     1     1     1     1     1
## 16837 16963 17957 18224
##     1     1     1     1
```

I created a network graph to visualize this similarity with 20 users.

```
#Visualizing similarity
random_users <- selected_users[1:20]
sim_mat <- cor(t(rmat), use = 'pairwise.complete.obs')
```

```
## Warning in cor(t(rmat), use = "pairwise.complete.obs"): the standard
## deviation is zero
```

```
g<-graph_from_adjacency_matrix(sim_mat[c(chosen_user, random_users), c(chosen_user, r
andom_users)], mode="undirected", diag=FALSE)
plot(g)
```



Getting predictions for other books

We take most similar books and avg their ratings for books Bob has not yet rated. We only include those books that have been rated by many other users

```
similar_users <- names(res[1:4])

similar_users_ratings <- data.frame(item = rep(colnames(rmat), length(similar_users))
, rating = c(t(as.data.frame(rmat[similar_users,])))) %>% filter(!is.na(rating))

chosen_user_ratings <- data.frame(item = colnames(rmat), rating = rmat[chosen_user,])
%>% filter(!is.na(rating))

predictions <- similar_users_ratings %>%
  filter(!(item %in% chosen_user_ratings$item)) %>%
  group_by(item) %>% summarize(mean_rating = mean(rating))


#Recommend best 5 predictions
bookrecs<-predictions %>%
  arrange(-mean_rating) %>%
  top_n(5, wt = mean_rating) %>%
  mutate(book_id = as.numeric(as.character(item))) %>%
  left_join(select(books10k, authors, title, id), by = c("book_id" = "id")) %>%
  select(-item)

bookrecs
```

mean_rating <dbl>	book_id <dbl>	authors <chr>
2.057138	2013	Ian McEwan
1.581139	3508	Marcel Proust, Simon Vance, Lydia Davis
1.500886	5010	Clive Cussler
1.360419	2544	Christopher Marlowe
1.290994	3755	Herman Melville

5 rows | 1-3 of 4 columns

Using recommender lab

Recommender Lab is a powerful R package solely used for recommendation systems. It has many collaborative filtering based algorithms and uses clustering as well. It also has inbuilt cross validation function to help us test the model. We will be using recommender lab to generate 5 books and compare the results with the results found using our own collaborative filtering algorithm above.

```
library(recommenderlab)
ratingmat0 <- ratingmat
ratingmat0[is.na(ratingmat0)] <- 0
sparse_ratings <- as(ratingmat0, "sparseMatrix")
rm(ratingmat0)
gc()
```

```
##             used   (Mb) gc trigger   (Mb) limit (Mb)  max used   (Mb)
## Ncells   2947783  157.5    5721704  305.6         NA   3881407  207.3
## Vcells 197185353 1504.5  539438437 4115.6      16384 557149239 4250.8
```

```
real_ratings <- new("realRatingMatrix", data = sparse_ratings)
real_ratings
```

```
## 18006 x 10000 rating matrix of class 'realRatingMatrix' with 389621 ratings.
```

```
model <- Recommender(real_ratings, method = "UBCF", param = list(method = "pearson",
nn = 4))

#Making predictions
prediction <- predict(model, real_ratings[chosen_user, ], type = "ratings")

reclabresult<-as(prediction, 'data.frame') %>%
  arrange(-rating) %>% .[1:5,] %>%
  mutate(book_id = as.numeric(as.character(item))) %>%
  left_join(select(books10k, authors, title, id), by = c("book_id" = "id")) %>%
  select(-item)
reclabresult
```

u...	rating	book_id	authors	title
<fctr>	<dbl>	<dbl>	<chr>	<chr>
173	3.739413	1198	Hanya Yanagihara	A Little Life
173	3.739413	1322	Peter S. Beagle	The Last Unicorn (The Last Unic
173	3.739413	1517	Justin Cronin	The Twelve (The Passage, #2)
173	3.739413	2064	Kate DiCamillo, Bagram Ibatoulline	The Miraculous Journey of Edw
173	3.739413	2486	Kazuo Ishiguro	The Buried Giant

5 rows

As you can see, using clustering algorithm, the results seem more accurate, because the ratings of these books are higher than the ratings of books found via collaborative filtering.

Comparing performance when number of neighbours are increased

```
scheme <- evaluationScheme(real_ratings[1:500,], method = "cross-validation", k = 10,
given = -1, goodRating = 5)

algorithms <- list(
                   "UBCF_05" = list(name = "UBCF", param = list(nn = 5)),
                   "UBCF_10" = list(name = "UBCF", param = list(nn = 10)),
                   "UBCF_30" = list(name = "UBCF", param = list(nn = 30)),
                   "UBCF_50" = list(name = "UBCF", param = list(nn = 50))
                   )

results <- evaluate(scheme, algorithms, type = "ratings")
```

```
## UBCF run fold/sample [model time/prediction time]
##   1  [0.003sec/1.26sec]
##   2  [0.004sec/0.716sec]
##   3  [0.004sec/0.706sec]
##   4  [0.002sec/0.716sec]
##   5  [0.004sec/0.717sec]
##   6  [0.004sec/0.735sec]
##   7  [0.004sec/0.733sec]
##   8  [0.004sec/0.75sec]
##   9  [0.004sec/0.767sec]
##   10  [0.003sec/0.729sec]
## UBCF run fold/sample [model time/prediction time]
##   1  [0.003sec/0.723sec]
##   2  [0.003sec/0.72sec]
##   3  [0.003sec/0.707sec]
##   4  [0.004sec/0.749sec]
##   5  [0.004sec/0.747sec]
##   6  [0.004sec/0.755sec]
##   7  [0.004sec/0.727sec]
##   8  [0.007sec/1.084sec]
##   9  [0.003sec/0.75sec]
##   10  [0.003sec/0.721sec]
## UBCF run fold/sample [model time/prediction time]
##   1  [0.003sec/0.72sec]
##   2  [0.003sec/0.729sec]
##   3  [0.004sec/0.737sec]
##   4  [0.003sec/0.744sec]
##   5  [0.003sec/0.73sec]
##   6  [0.004sec/1.153sec]
##   7  [0.003sec/0.73sec]
##   8  [0.002sec/0.719sec]
##   9  [0.002sec/0.729sec]
##   10  [0.003sec/0.72sec]
## UBCF run fold/sample [model time/prediction time]
##   1  [0.003sec/0.741sec]
##   2  [0.004sec/0.734sec]
##   3  [0.003sec/0.731sec]
##   4  [0.004sec/1.433sec]
##   5  [0.003sec/0.718sec]
##   6  [0.004sec/0.734sec]
##   7  [0.002sec/0.706sec]
##   8  [0.003sec/0.73sec]
##   9  [0.002sec/0.719sec]
##   10  [0.002sec/0.724sec]
```

```
tmp <- lapply(results, function(x) slot(x, "results"))
res <- tmp %>%
  lapply(function(x) unlist(lapply(x, function(x) unlist(x@cm[ ,"RMSE"])))) %>%
  as.data.frame() %>%
  gather(key = "Algorithm", value = "RMSE")

rmse<-aggregate(res,by=list(res$Algorithm),FUN = mean)
```

```
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA
```

```
rmse
```

Group.1	Algorithm	RMSE
<chr>	<dbl>	<dbl>
UBCF_05	*NA*	0.9600184
UBCF_10	*NA*	0.9620263
UBCF_30	*NA*	0.9647466
UBCF_50	*NA*	0.9672828

4 rows

Comparing the models :

We see that recommenderLab's algorithm works slightly better, in that the books it outputs have a better mean rating than the collaborative filter we created. We can also compare the RMSE of the algorithm when the number of nearest neighbours is increased. As you can see, when the number increases, the RMSE gets better. That means if you take into consideration more similar users, your recommendation will be better.

Conclusion :

We have learnt several methods of data analysis implementation in this class. With our simple project, we learnt how to do data exploration, which is a key process before building any sort of model. This project uses descriptive statistics to first analyse the data and distribution of its variables. We then built a recommendation system using two methods- a collaborative filter designed over finding similar users and recommending books they have read, but which the current user has not, and then using recommenderLab's built in UBCF (user based collaborative filtering) algorithm and seeing how both compare. When the top-k nearset neighbours are chosen we get the best result as seen.

References :

goodbooks-10k, Foxtrot, 2018. Ten thousand books, one million ratings. Also books marked to read, and tags. Retrieved from kaggle.com:https://www.kaggle.com/zygmunt/goodbooks-10k (https://www.kaggle.com/zygmunt/goodbooks-10k)

Netflix Recommendation Engine. Retrieved from codeacademy.com:https://www.codecademy.com/articles/how-netflix-recommendation-works-data-science (https://www.codecademy.com/articles/how-netflix-recommendation-works-data-science)

Soumik. (2020, March 9). Goodreads-books. Retrieved from https://www.kaggle.com/jealousleopard/goodreadsbooks/kernels (https://www.kaggle.com/jealousleopard/goodreadsbooks/kernels)

What Makes a Book Worth Reading? (2018, May 1). Retrieved from http://booksmakeadifference.com/bookgood/ (http://booksmakeadifference.com/bookgood/)