HarvardX: PH125.9x Data Science - Capstone Project
Kickstarter - Crowdfunding Success Prediction

Léo Dange - https://github.com/ldange

January 1st, 2021

Introduction

Related to the ***Harvardx Data Science Program - Capstone PH125.9x***, this second project of building a machine learning algorithm concerns the Kickstarter platform and its crowdfunding project. I have been using Kickstarter as a “pledger/founder” for many years and I recently helped a friend to launch a project that indeed has not been successfully funded.

Kickstarter as a platform is for me an incredible opportunity to help brilliant people to develop a product and launch their company. I like to concept and all its aspects, but It is sometimes complicated to analyze and understand what determines the success or failure of a project. For some of them, it can be obvious but some are more tricky. I often have issues understanding why some projects did get the finance when I thought (in my own opinion) they would not have success, and why sometimes projects failed when I thought they should have succeeded.

This project will help us to determine if we can predict the success or failure of a project regarding a few variables.

This Report will present you the ***Project Goal, the Data used, The Methods and Analysis of the results*** and at the end some ***Conclusion***.

The project will be delivered with the following files :

- The Report in .pdf
- The Report in .Rmd
- The Code and Script in .R

for readability purposes, the plot codes of the .pdf report are hidden, they are available in the .Rmd version of the report.

The code of all calculations, database construction, data frame modifications, or algorithm construction are displayed on all versions.

Project's Goal

As explained in the Introduction, the Project's Goal is to build a machine learning algorithm able to predict the success of a project. We will build several models and make them compete to determine which one is the most accurate.

To evaluate our work we are going to use the Root Mean Square Error or RMSE :

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i} (\hat{y}_{u,i} - y_{u,i})^2}$$

The RMSE is frequently used to measure the difference between values predicted by a model and the observed values, in our case between the ***edx_kickstater set (Train subset) and the Validation_kickstarter subset***.

The best model will be defined as the one with the lowest RMSE, this time we do not have a targeted goal so we will pick the best one of our several models.

Data

Our two sets are based on the Kickstarter project Database available on Kaggle (link in the code below) originally contains **378'661 projects, 15 variables/observation**. Before using We will :

1) start by removing renaming and reordering columns.

2) Filtering the status and only keep the "successful" "failed" & "canceled" projects.

After those modifications our dataframe will contain **370'454 observations & 11 variables**.

Then we split this data set into two partitions, for this operation I chose to keep the 90/10 as we did in our MovieLens Analysis. the train_kickstarter set (train set) will hold 90% of the data, and the validation_kickstarter 10% left.

For simplicity, the Dataset has been download from Kaggle then upload to my GitHub repository, dedicated to this project.

```
##########################################################
# Create kickstarter train set, validation set (final hold-out test set,
similar to the edx set for the Movielens Project)
##########################################################

# Note: this process could take a couple of minutes

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")
```

```r
if(!require(data.table)) install.packages("data.table", repos =
"http://cran.us.r-project.org")
if(!require(stringr)) install.packages("stringr", repos = "http://cran.us.r-
project.org")
if(!require(gridExtra)) install.packages("gridExtra", repos =
"http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(data.table)
library(stringr)

# Kickstarter's Projects Database:
# https://www.kaggle.com/kemical/kickstarter-projects?select=ks-projects-
201612.csv
# https://www.kaggle.com/kemical/kickstarter-projects/download

# Data can be viewed/downloaded from my github repository
# https://github.com/ldange/HarvardX-Data-Science-Capstone/blob/master/ks-
projects-201801.csv

# Creating the dataframe from csv file
kickstarter <- read.csv("https://raw.githubusercontent.com/ldange/HarvardX-
Data-Science-Capstone/master/ks-projects-201801.csv")

# Removing unecessary column
drops <- c("deadline","usd.pledged","usd_pledged_real","usd_goal_real")
kickstarter <- kickstarter[ , !(names(kickstarter) %in% drops)]

# Renaming and Reordering column
kickstarter <- kickstarter %>%
  rename(
    sub_category = category,
    status = state,
    projectId = ID
  )
kickstarter <- kickstarter[ , c(1,2,4,3,11,5,7,10,8,6,9)]

# Creating subset datafram without the status "live, suspended & undefined"
kickstarter <- subset(kickstarter, kickstarter$status %in%
c("canceled","failed","successful") )

# Validation set will be 10% of Kickstarter Project data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use
`set.seed(1)`
test_index_kickstarter <- createDataPartition(y = kickstarter$status, times =
1, p = 0.1, list = FALSE)
train_kickstarter <- kickstarter[-test_index_kickstarter,]
validation_kickstarter <- kickstarter[test_index_kickstarter,]
```

As stated before, the two sets are now composed of 11 variables instead of the original 15 and we can see that the ***train_kickstarter set is, as chosen, containing 90% of the data***.

```r
str(train_kickstarter)
```

```
## 'data.frame':    333408 obs. of  11 variables:
##  $ projectId    : int  1000002330 1000004038 1000007540 1000011046
1000014025 1000023410 1000030581 1000034518 100004195 100004721 ...
##  $ name         : chr  "The Songs of Adelaide & Abullah" "Where is Hank?"
"ToshiCapital Rekordz Needs Help to Complete Album" "Community Film Project:
The Art of Neighborhood Filmmaking" ...
##  $ main_category: chr  "Publishing" "Film & Video" "Music" "Film & Video"
...
##  $ sub_category : chr  "Poetry" "Narrative Film" "Music" "Film & Video"
...
##  $ country      : chr  "GB" "US" "US" "US" ...
##  $ currency     : chr  "GBP" "USD" "USD" "USD" ...
##  $ launched     : chr  "2015-08-11 12:12:28" "2013-01-12 00:20:50" "2012-
03-17 03:24:11" "2015-07-04 08:35:03" ...
##  $ backers      : int  0 3 1 14 224 16 40 58 43 0 ...
##  $ pledged      : num  0 220 1 1283 52375 ...
##  $ goal         : num  1000 45000 5000 19500 50000 1000 25000 125000 65000
2500 ...
##  $ status       : chr  "failed" "failed" "failed" "canceled" ...
```

The 11 remaining features/variables/columns in both datasets are as follow :

- **projectId**, ***integer*** containing the identification number of the project.
- **name**, ***character*** containing the name.
- **main-category**, ***character*** containing the main-category (parent of the category).
- **sub-category**, ***character*** containing the “Sub-category”.
- **country**, ***character*** containing the country of origin of the project/company behind the project
- **currency**, ***character*** containing the currency related to the country
- **launched**, ***character*** containing the launched date of the project
- **backers**, ***integer*** containing the number of backers of the project
- **pledged**, ***numeric*** containing the amount of money pledged to the project
- **goal**, ***numeric*** containing the target amount of money aimed for the project

Exploratory Data Analysis

```r
head(train_kickstarter, n=3)
```

```
##    projectId                                              name
main_category
## 1 1000002330                   The Songs of Adelaide & Abullah
Publishing
## 3 1000004038                                    Where is Hank?  Film &
Video
```

```
## 4 1000007540 ToshiCapital Rekordz Needs Help to Complete Album
Music
##     sub_category country currency            launched backers pledged
goal
## 1         Poetry      GB      GBP 2015-08-11 12:12:28       0       0
1000
## 3 Narrative Film      US      USD 2013-01-12 00:20:50       3     220
45000
## 4          Music      US      USD 2012-03-17 03:24:11       1       1
5000
##   status
## 1 failed
## 3 failed
## 4 failed
```

Our EDA will be conducted in three steps.

1) Funding Analysis - Aiming to analyze if any patterns can be found regarding the funding (and success of a project).

2) Category Analysis - What category are more likely to have a project funded, are they all the same, is there any “go-to” sub_category that the project creator should relate to?

3) Country Analysis - Which country is most likely to use Kickstarter, are they all funded the same, …

Then we will summarize our analysis.

To be able to conduct an Exploratory Data Analysis those data will need at least one transformation, in addition to cleaning/reorganizing we proceed before. We will use an ***eda*** data frame to do our exploration, without any risk of alteration for the train set. This set will be ***train_eda_kickstarter***.

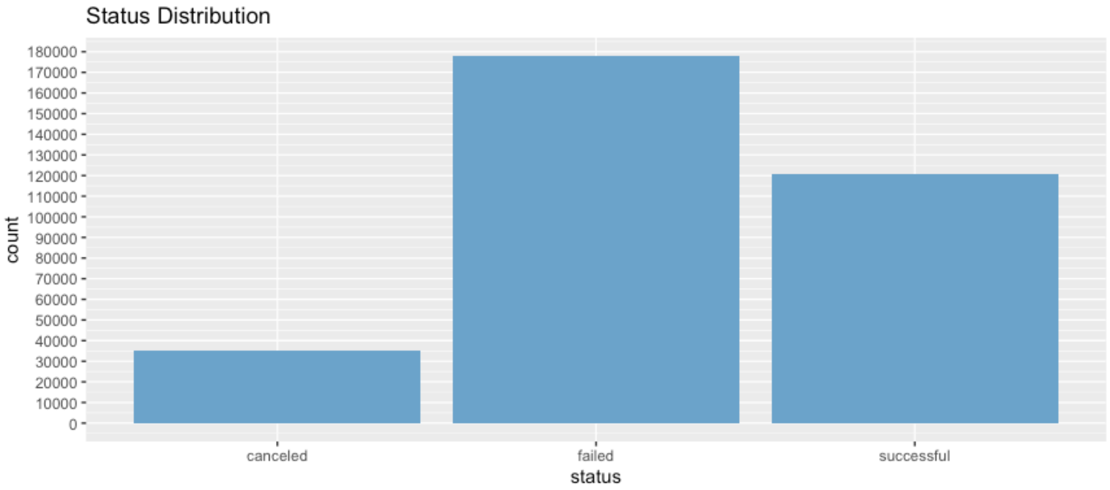
- Add a **funding_percentage** variable that tells us the percentage of funding of the project

```r
# Create a completion percentage for each project and limiting the number to
only two digits (for the whole project).

train_eda_kickstarter <- train_kickstarter %>% mutate(funding_percentage =
as.numeric(pledged / goal)*100)

options(digits=2)
```

Funding Analysis

We will start our analysis with a quick Funding Analysis. This first plot will help us to understand which part of the project is successful.

First major information, we can see that most of the projects failed on Kickstarter.



Failed project represent 53.37% of all projects when **successful projects are only 36.15%**. Finally, **Canceled projects represent the remaining 10.48%**.

```
# Looking at the Failing & Canceled ratio of projects
kickstarter_failed <- train_eda_kickstarter %>% filter(status == "failed")
%>% count() / train_eda_kickstarter %>% count
kickstarter_failed

##      n
## 1 0.53

kickstarter_successful <- train_eda_kickstarter %>% filter(status ==
"successful") %>% count() / train_eda_kickstarter %>% count
kickstarter_successful

##      n
## 1 0.36
```

Still, one-third of the project getting marked as successful seems a lot, comparing to the average success of a company/product "in real life". when we look at the data frame, lots of successful projects had a goal of 1 / 10 or 100 dollars, which is quickly reached but doesn't guarantee any product release.

2'386 projects have a goal inferior to 200 dollars and less than 10 backers in the project. Some even have 0 backers and seems successful because of the creator pledge itself.

```
#Counting Sucessfull projects with less than 200 of goals and 10 backers
train_eda_kickstarter %>% filter(status == "successful" & goal <= 200 &
backers <= 10) %>% count()
```

```
##      n
## 1 2386
```

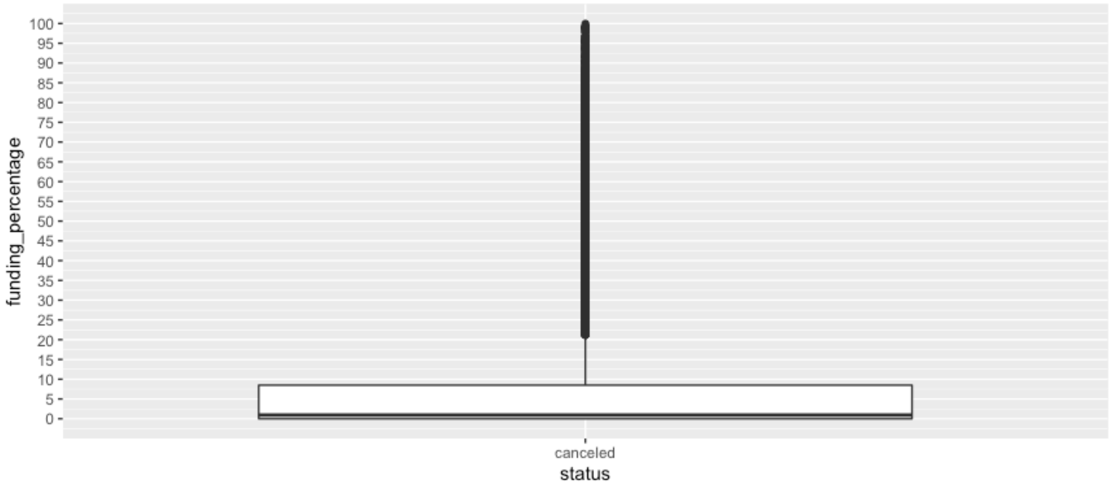
Canceled projects are often related to insufficient backers/funds, crater rather canceled the project before the end of the financing schedule instead of waiting knowing it won’t be successful anyway. In our case, we do not have the reason why the project was canceled but we can analyze what percentage of founding received the canceled project.

Only 649 of the 34’901 canceled projects reach at least 100% of the funding goal, which means **34’252 projects were not funded before they were canceled**.

On the 649 funded projects but canceled, **103 projects had goals below 200$**, which reduce more the number of projects that reach a real funding goal.

```
# Project funded but canceled
train_eda_kickstarter %>% filter(status == "canceled" & funding_percentage >=
100) %>% count()

##     n
## 1 646

# Project funded with a funding goal bellow 200
train_eda_kickstarter %>% filter(status == "canceled"& goal >= 200 &
funding_percentage >= 100) %>% count()

##     n
## 1 501
```

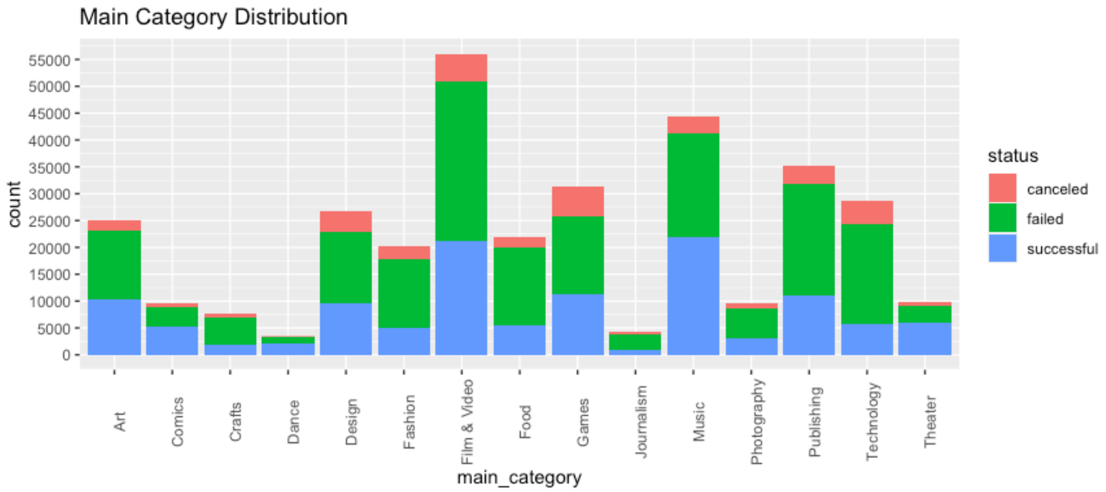
If we look at the projects that were canceled without been founded, we see that ***most of the projects got between 0 (first quartile) and 8% (third quartile) of their funding goal, with a median bellow 2%.*** We exclude canceled project with over 100% of funding on this boxplot



Category Analysis

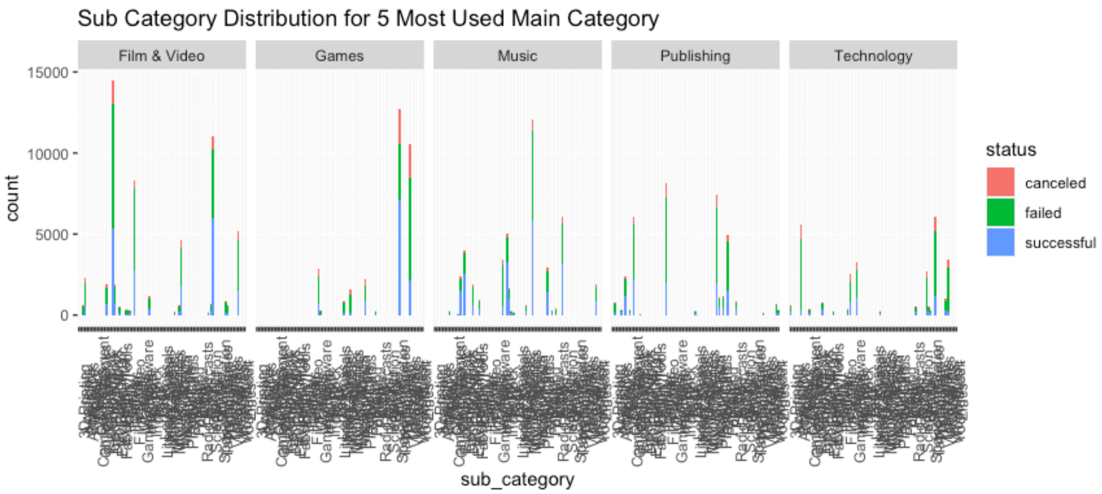
We will now analyze the category.

By looking at the main category distribution bellow, we can see that the most used main category is in order **Film & Video, Music, Publishing, Games & Technology**.



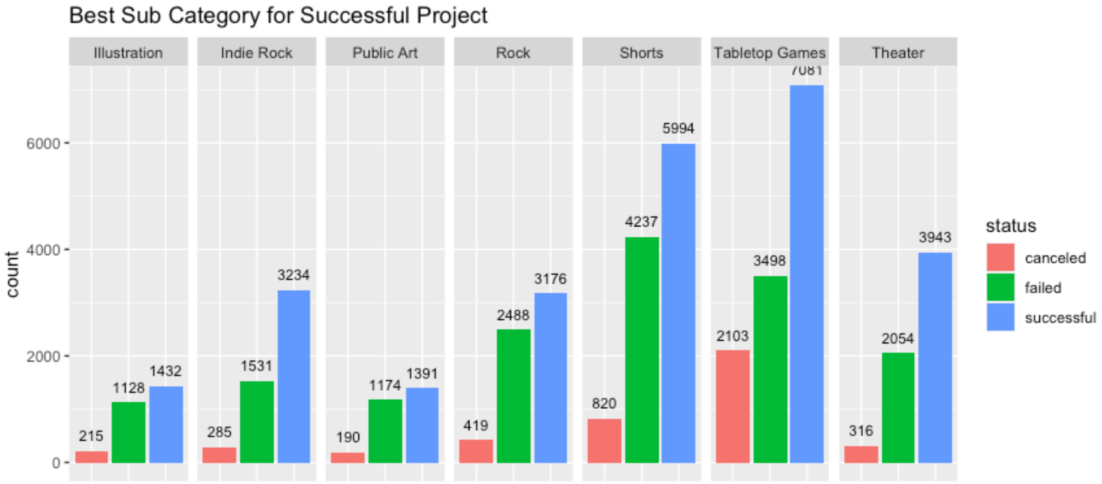
For all category excepted **Comics, Dance & Theater**, there is more canceled & Failed projects than a successful one. All those main categories ***Comics, Dance & Theater*** are in the & least used the main category of the entire platform.

```
comics_ratio <- train_eda_kickstarter %>% filter(status == "successful" &
main_category == "Comics") %>% count() / train_eda_kickstarter %>%
filter(main_category == "Comics") %>% count()
comics_ratio

##      n
## 1 0.55

dance_ratio <- train_eda_kickstarter %>% filter(status == "successful" &
main_category == "Dance") %>% count() / train_eda_kickstarter %>%
filter(main_category == "Dance") %>% count()
dance_ratio

##      n
## 1 0.62

theater_ratio <- train_eda_kickstarter %>% filter(status == "successful" &
main_category == "Theater") %>% count() / train_eda_kickstarter %>%
filter(main_category == "Theater") %>% count()
theater_ratio

##     n
## 1 0.6
```



For each of the 5 most used main category, is always two or three sub_category that are way more used than de rest of the entire category.

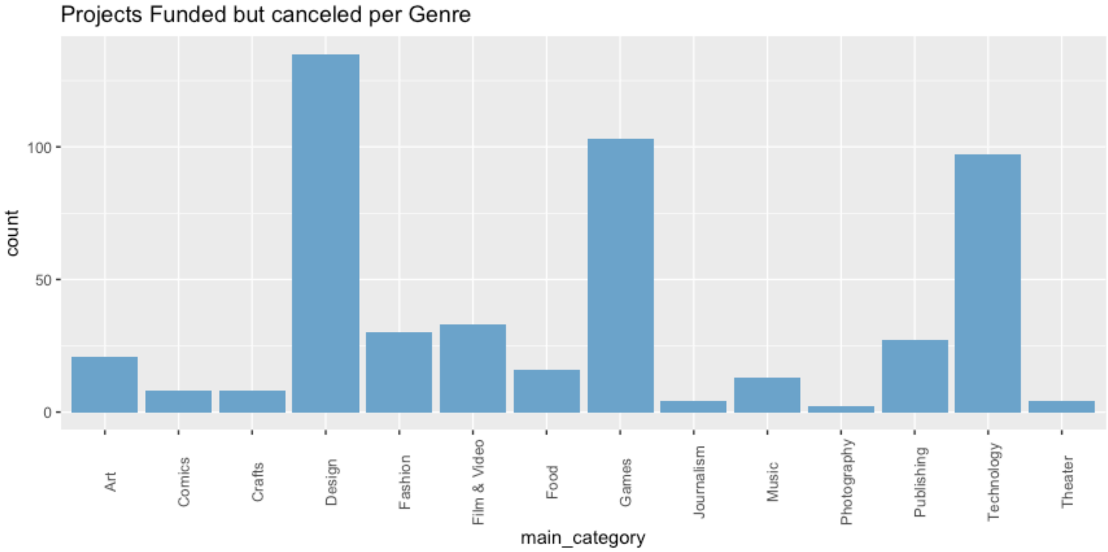
For our next plot we've looked on all the sub_category and we saw there is 7 sub_category where there are more successful projects than failed and canceled one. We've plotted the status for each of them. This information could help Kickstarters to understand where to post their projects to maximize the chance of funding.

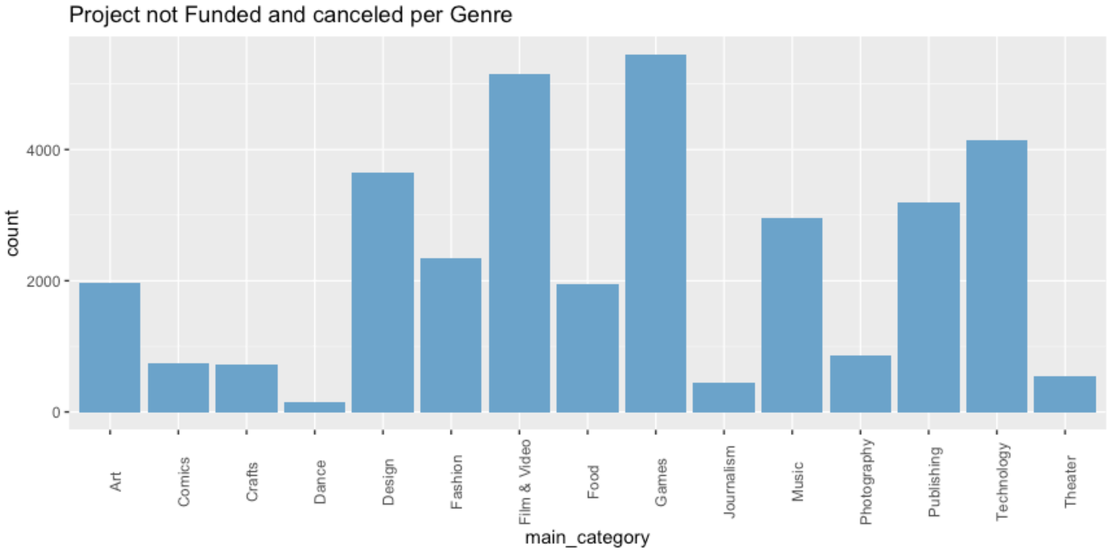


We will plot the best and the worst "canceled project" regarding their genre.

```
#Creating dataframes for the highest & lowest funding_percentage but canceled
project

best_canceled <- train_eda_kickstarter %>%
  filter(status == "canceled" & goal >= 200 & funding_percentage >= 100)

best_canceled <- data.frame(main_category =
best_canceled$main_category,funding_percentage =
best_canceled$funding_percentage)
```

```
worst_canceled <- train_eda_kickstarter %>%
  filter(status == "canceled" & funding_percentage <= 100)

worst_canceled <- data.frame(main_category =
worst_canceled$main_category,funding_percentage =
worst_canceled$funding_percentage)
```





We can see on those two plots that, Design, Games & Technology are vastly present in the canceled projects. Except for the Design category, almost all project categories mostly failed and were canceled, rather than being funded but canceled, which is not surprising.

We will construct a kick data frame regarding the amount of project for each main_category and the ratio of a successful project to see if there is any correlation.

```
# create dataframes for the total projects and sucessful project for each
main_category
```

```
total_maincategory <- train_eda_kickstarter %>%
  group_by(main_category) %>%
  count()

total_maincategory <- data.frame(main_category =
total_maincategory$main_category,
                                 total = as.numeric(total_maincategory$n))

successful_maincategory <- train_eda_kickstarter %>%
  group_by(main_category) %>%
  filter(status == "successful") %>%
  count()

successful_maincategory <- data.frame(main_category =
successful_maincategory$main_category,
                                      successful =
as.numeric(successful_maincategory$n))

projects_maincategory <-
left_join(total_maincategory,successful_maincategory)

## Joining, by = "main_category"

projects_maincategory <- projects_maincategory[order(-
projects_maincategory$total),]

success_ratio <- data.frame(main_category =
projects_maincategory$main_category,
                              success_ratio =
as.numeric(projects_maincategory$successful / projects_maincategory$total))

projects_maincategory <- left_join(projects_maincategory,success_ratio)

## Joining, by = "main_category"
```

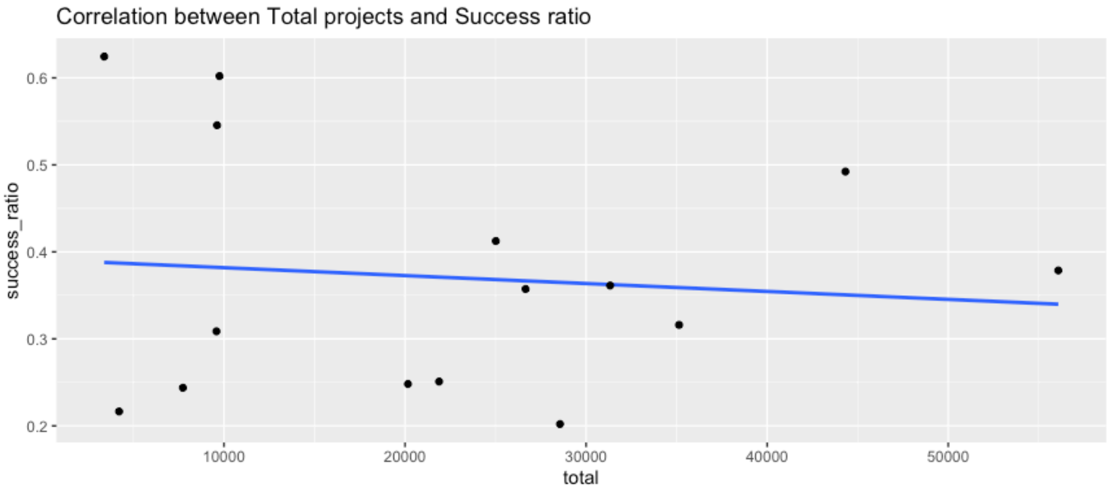
A quick look at the values.

```
# Looking at the data
head(projects_maincategory, n=5)

##   main_category total successful success_ratio
## 1  Film & Video 56080      21232          0.38
## 2         Music 44321      21819          0.49
## 3    Publishing 35134      11104          0.32
## 4         Games 31329      11319          0.36
## 5    Technology 28559       5770          0.20

# Correlation test between total project and success ratio
projects_maincategory %>% summarize(r = cor(total,success_ratio))
```

```
##      r
## 1 -0.1
```

We could have thought that the bigger the category is, to bigger the chance of funding is (due to traffic on this category), but we can see that there is strictly no correlation between the number of projects and the ratio of the successfully funded projects.

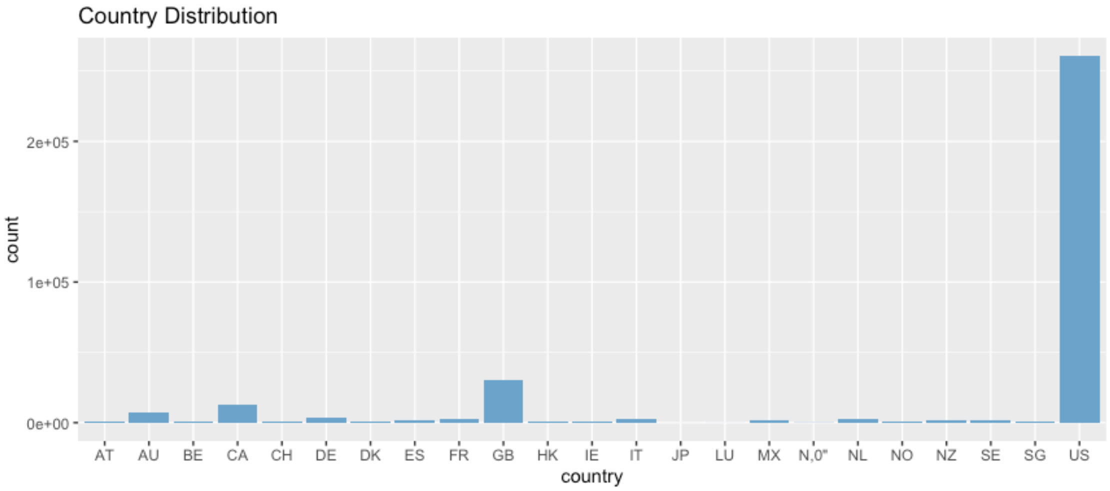
```
## `geom_smooth()` using formula 'y ~ x'
```



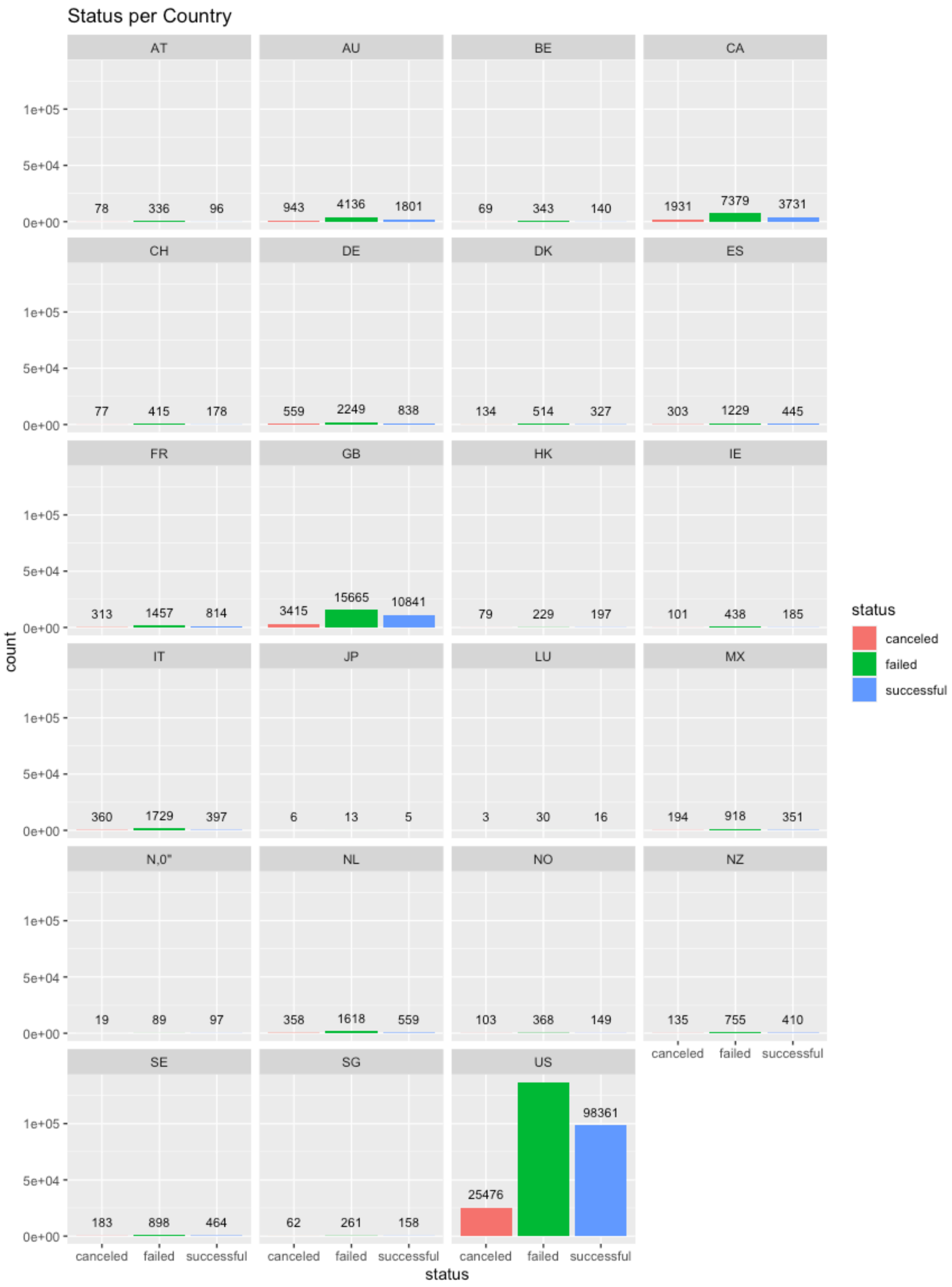
Country Analysis

Lastly, we will conduct an analysis to analyze the trend in different countries.

We start our analysis with a country distribution that shows the US as the biggest provider of projects, followed by Great Britain, Canada, and Australia. All English speaker country.



If we look at the status of each country, we see that no country has more successful projects than failed/canceled projects.

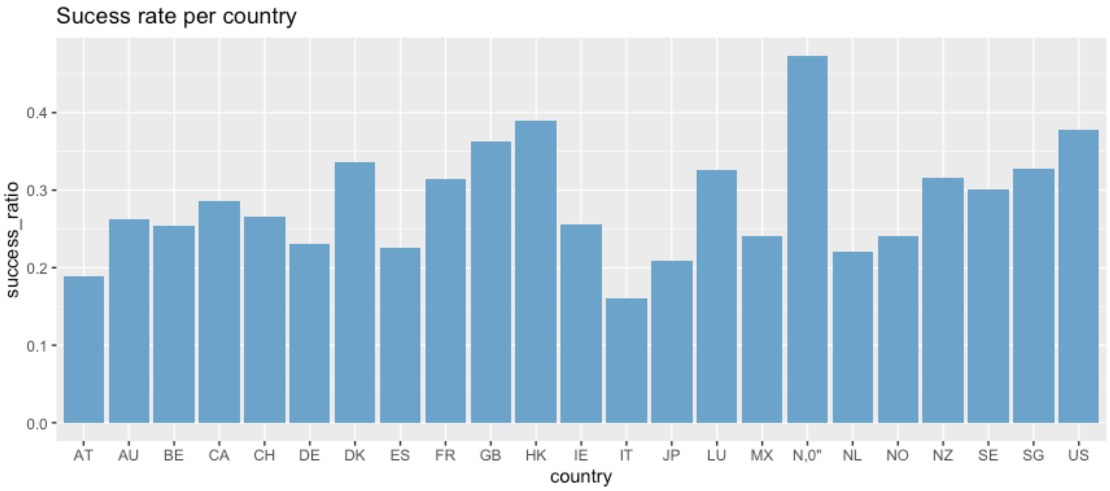


This is confirmed by the success rate for each country.

```
# Creating dataframe regarding sucess ratio per country
country_total_success <- train_eda_kickstarter %>%
     group_by(country) %>%
     count()
```

```
country_total_success <- data.frame(country = country_total_success$country,
                                     total =
as.numeric(country_total_success$n))

country_success <- train_eda_kickstarter %>%
     group_by(country) %>%
     filter(status == "successful") %>%
     count()
country_success <- data.frame(country = country_success$country,
                               successful = as.numeric(country_success$n))

country_ratio <- data.frame(country = country_total_success$country,
                             success_ratio =
as.numeric(country_success$successful / country_total_success$total))

country_total_success <- left_join(country_total_success,country_success)

## Joining, by = "country"

country_total_success <- left_join(country_total_success,country_ratio)

## Joining, by = "country"
```

Here is the plot of the success rate. We quickly see that the number of projects (previous graph) has zero impact on the success ratio for each country.

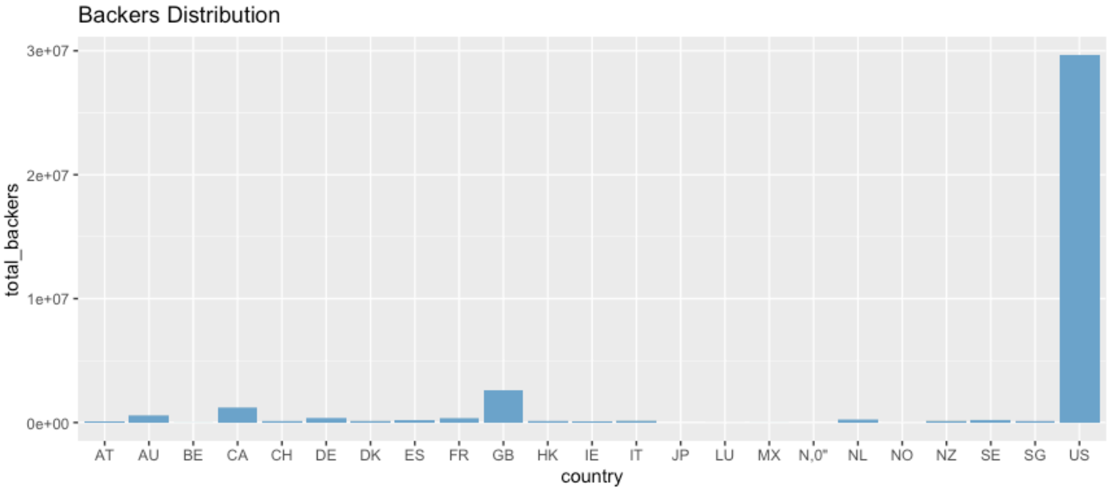


Now we are interested in the number of backers regarding the origins/country of the projects. We see that this distribution is almost the same as the one regarding the project per country.

```
# Count Backers per Country
country_backers <- train_eda_kickstarter %>%
  group_by(country) %>%
  count(backers)
```

```
country_backers <- data.frame(country = country_backers$country,
                              backers = as.numeric(country_backers$backers),
                              total = as.numeric(country_backers$n))

country_backers<- country_backers %>%
     group_by(country) %>%
     dplyr::summarise(total_backers = backers * total) %>%
     mutate(total_backers = total_backers)
```

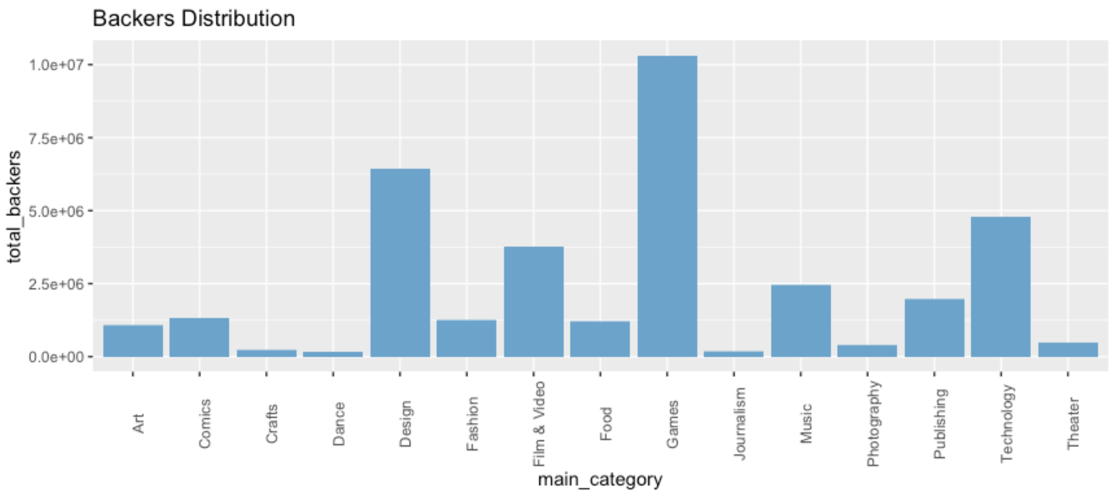
```
## `summarise()` regrouping output by 'country' (override with `.groups`
argument)
```



It would have been interesting to know the origins of the backers to try to establish a trend of project backing. Having a look at the backer's distribution per main_category highlights that, even if ***Games, Design & Technology*** are not the 3 most used main_category regarding projects, they are regarding backers interest.

```
# Count Backers per Country
category_backers <- train_eda_kickstarter %>%
     group_by(main_category) %>%
     count(backers)

category_backers <- data.frame(main_category =
category_backers$main_category,
                              backers = as.numeric(category_backers$backers),
                              total = as.numeric(category_backers$n))

category_backers<- category_backers %>%
     group_by(main_category) %>%
     dplyr::summarise(total_backers = backers * total) %>%
     mutate(total_backers = total_backers)
```

```
## `summarise()` regrouping output by 'main_category' (override with
`.groups` argument)
```



EDA Analysis Summary

Our EDA was exhaustive but gave us some interesting information, here is what we discovered :

- More than ***half (53.37%) of the project failed*** on Kickstarter
- Nearly ***two-third (63.85%) of the project are not successful** (Failed + Canceled)
- 9'441 projects (2.8% of all projects) were launched with a ***goal bellow 200$***
- Some projects have been mysteriously canceled despite having been funded with tens of thousands.
- On average, ***a project is canceled if he did not get more than 2%*** of the requested funding.
- The most used Main_Category are in order Film & Video, Music, Publishing, Games & Technology
- ***Comics (55%), Dance (62%) & Theater (60%)*** are ***the only main_category with more successful projects*** than canceled and failed one. Those categories are also the least used.
- ***Illustration, Indie Rock, Public Art, Rock, Shorts, Tabletop Games & Theater*** are the only sub_category with more successful projects than canceled and failed ones.
- There is ***no correlation between the size of a main_category (number of the project) and the sucess_ratio*** (project funded/total projects), but ***the only "positive ratio" are found in main_category with small amount of projects*** (less than 10'000)
- Us is by far the most represented country on Kickstarter, in general, ***English speaking countries are more present than other countries***
- ***No country has a positive ratio of funded projects***, the success of a project does not seem relative to its origins.
- As for the origins of a project, Backers are most likely to be ***from the US or other English speaking countries.***
- Projects relative to the main_category Games got the highest number of backers.

Methods & Analysis

We will now try to find a model with the appropriate approach to correctly predict rating. As stated previously, The best model will be defined as the one with the lowest RMSE. We will evaluate the different approach to try to reach our goal :

- A basic model based on **the average funded project** of the data where ***mu = average pledged*** .
- An improvement of the basic model with the addition of ***main_category bias term***
- An improvement of the second model with the addition of ***sub_category bias term*** in addition to the first one
- An improvement of the third model by changing the second bias term to ***country bias term*** instead of sub_category
- A failed attempt of Regularized model of our last improvement, as an independent error term may induce error our the RMSE.

Average Funded project model

In this first model, we start by building the easiest possible predicting system, while predicting all the unknown funding based on ***mu***, as defined before.

$$Y_{u,i} = \mu,$$

```
#Compute mu & predict unknown funding based on mu
mu <- mean(train_kickstarter$pledged)
average_funding_project <- RMSE(train_kickstarter$pledged, mu)
average_funding_project

## [1] 97359
```

This approach gives us the first RMSE of **97359**.

Average Funded project model with main_category bias model

In our second model, we will start to implement an independent error term that takes into consideration different pledges for a main_category as b_i is the main_category bias term. As we saw earlier in our analysis projects are not funded the same (in number and average rate), this bias term considers this.

```
#compute the movie bias b_i
b_i <- train_kickstarter %>%
  group_by(main_category) %>%
  summarize(b_i = mean(pledged - mu))

## `summarise()` ungrouping output (override with `.groups` argument)
```

$$Y_{u,i} = \mu + b_i$$

```
#Predicting the funding with mu and the main_category bias term.
predicted_funding_1 <- validation_kickstarter %>%
  left_join(b_i, by='main_category') %>%
  mutate(pred = mu + b_i) %>%
  pull(pred)
average_funding_project_w_category_biais <- RMSE(predicted_funding_1,
validation_kickstarter$pledged)
average_funding_project_w_category_biais

## [1] 86014
```

We improved the accuracy of our RMSE with this first bias ***from 97359 to 86014.***

Average project funding with main_category & sub_category bias model

In this third iteration of the average project funding model, we are adding a sub_category bias term in addition to the main_category bias term introduce before. This second bias aims to reduce the effect of extreme funding of some projects, related to some sub_category as seen before (few categories have more successful project than failed/canceled ones which is an anomaly compared to the rest of the base)

```
# Compute the bias term, b_u
b_u <- train_kickstarter %>%
  left_join(b_i, by='main_category') %>%
  group_by(sub_category) %>%
  summarize(b_u = mean(pledged - mu - b_i))
```

$$Y_{u,i} = \mu + b_i + b_u$$

```
#Predicting the rating with mu, the main_category and sub_category bias term.
predicted_funding_2 <- validation_kickstarter %>%
  left_join(b_i, by='main_category') %>%
  left_join(b_u, by='sub_category') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

avg_movie_rating_w_subcategory_bias <- RMSE(predicted_funding_2,
validation_kickstarter$pledged)
avg_movie_rating_w_subcategory_bias

## [1] 85768
```

We improved the accuracy of our RMSE with this first bias ***from 86014 to 85768.***

Average project funding with main_category & country bias model

In this fourth iteration of the average project funding model, we are changing our second bias term by country instead of sub_category. As the sub_category is directly related to the first category, It could influence the results and not give us more accuracy.

```
# Compute the bias term, b_u
b_u <- train_kickstarter %>%
  left_join(b_i, by='main_category') %>%
  group_by(country) %>%
  summarize(b_u = mean(pledged - mu - b_i))

#Predicting the funding with mu, the main_category and country bias term.
predicted_funding_3 <- validation_kickstarter %>%
  left_join(b_i, by='main_category') %>%
  left_join(b_u, by='country') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

avg_movie_rating_w_country_bias <- RMSE(predicted_funding_3,
validation_kickstarter$pledged)
avg_movie_rating_w_country_bias

## [1] 85603
```

We once again improved our RMSE from ***85768. to 85603**.

Regularized main_category & country effect model

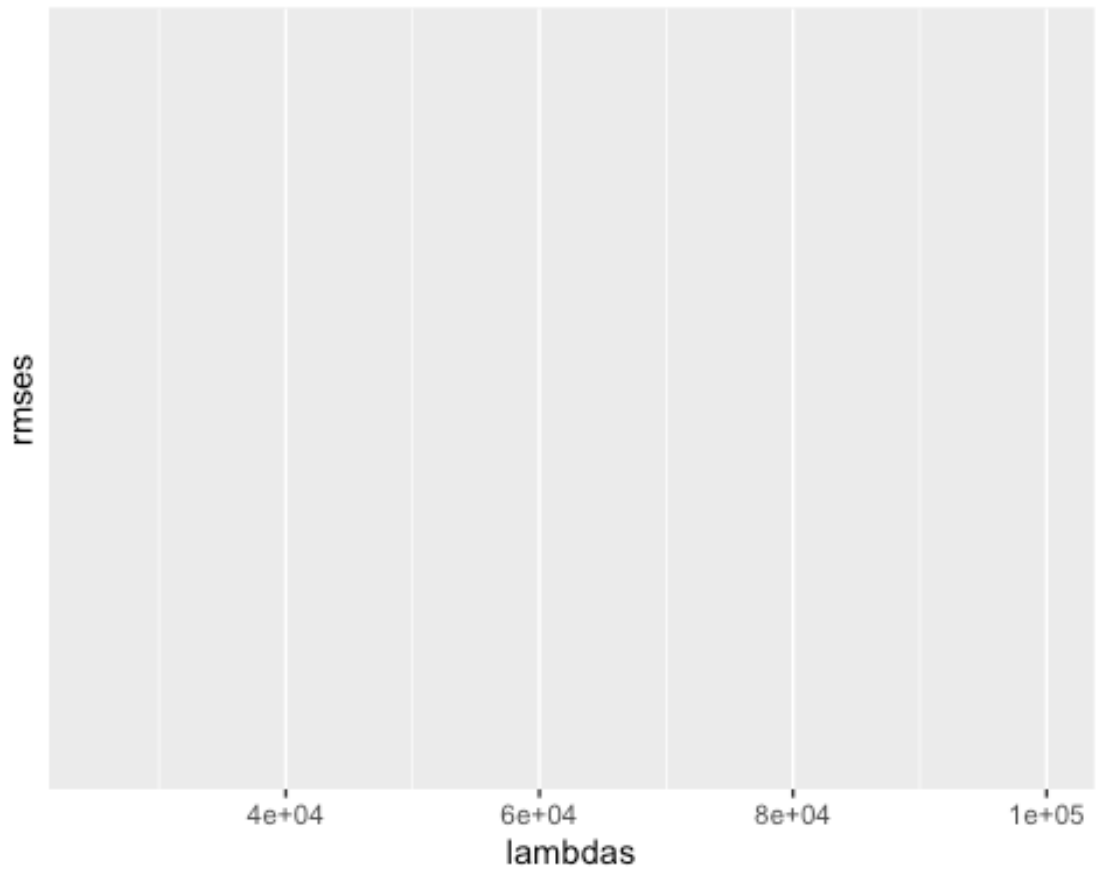
I have tried for the MovieLens project to regularized my previous model but I, unfortunately, failed to do so. Still, here is the code I've tried.

```
lambdas <- seq(from=25000, to=100000, by=2500)

rmses <- sapply(lambdas, function(l){

  mu <- mean(train_kickstarter$pledged)

   b_i <- train_kickstarter %>%
    group_by(main_category) %>%
    summarize(b_i = sum(pledged - mu)/(n()+l))

   b_u <- train_kickstarter %>%
    left_join(b_i, by='main_category') %>%
    group_by(country) %>%
    summarize(b_u = sum(pledged - b_i - mu)/(n()+l))

  predicted_funding_3 <- validation_kickstarter %>%
    left_join(b_i, by = "main_category") %>%
    left_join(b_u, by = "country") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)

  return(RMSE(predicted_funding_3, validation_kickstarter$rating))
})
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
qplot(lambdas, rmses)
```

```
## Warning: Removed 31 rows containing missing values (geom_point).
```



```
lambdas[which.min(rmses)]
```

```
## numeric(0)
```

Results

By improving step by step our RMSE with the introduction of our first bias term, up to the fourth iteration, we improved the accuracy of our RMSE from ***97359*** to ***85603***. My failed attempt of regularization does not help to improve the results.

We could have explored other approaches such as basing our entire model on a regression line approach through a GLM and KNN approach but due to a lack of time I've chosen another path.

Conclusion

Our RMSE is not as precise as it could have been. Having a user profile and data frame or having the data frame constructed around user funding instead of the project could have helped to improve our model.