Projet-Anime

Marwan

15/09/2021

Projet : Programmation sous R.

Partie 1

Jeu de données : anime.zip

Dans un premier temps, on charge les librairies nécessaires pour la suite du code.

```
library("dplyr")
library("plyr")
library("readr")
library("tidyverse")
library("ggplot2")
library("goeveg")
library("purrr")
library("rlang")
library("ggforce")
```

1. Proposer un code permettant de regrouper les données contenues dans anime.zip dans un seul et même data.frame.

```
# dir.create("temp")
# temp <- tempfile()
# con <- unzip("anime.zip", exdir = "temp")
dat <- list.files(path = "C:/Users/Marwan/Desktop/M2 TIDE/prérentré/intro R/Projet R/temp",
                        pattern = "*.csv", full.names = TRUE) %>%
  ldply(read_csv)
```

On crée un répertoire “temp” où l’on va unzippé nos données pour pouvoir ensuite stocker l’ensemble des données dans une data frame nommée “dat”

2. Observer pour chaque variable, le nombre et la proportion de valeurs manquantes.

```
dat %>%  glimpse()
```

```
## Rows: 12,294
## Columns: 10
## $ anime_id    <dbl> 32281, 15335, 28851, 199, 12355, 164, 7311, 28957, 431, 31~
## $ name        <chr> "Kimi no Na wa.", "Gintama Movie: Kanketsu-hen - Yorozuya ~
## $ genre       <chr> "Drama, Romance, School, Supernatural", "Action, Comedy, H~
## $ type        <chr> "Movie", "Movie", "Movie", "Movie", "Movie", "Movie", "Mov~
## $ episodes    <chr> "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1"~
## $ rating      <dbl> 9.37, 9.10, 9.05, 8.93, 8.84, 8.81, 8.81, 8.75, 8.74, 8.73~
## $ members     <dbl> 200630, 72534, 102733, 466254, 226193, 339556, 240297, 322~
## $ rating_10   <dbl> 9.37, 9.10, 9.05, 8.93, 8.84, 8.81, 8.81, 8.75, 8.74, 8.73~
```

```
## $ rating_100  <dbl> 127.7577, 121.6781, 120.5519, 121.4881, 130.1210, 125.6115~
## $ rating_1000 <dbl> 937, 910, 905, 893, 884, 881, 881, 875, 874, 873, 868, 866~
```

```
dat %>%
  select_if(function(x) any(is.na(x))) %>%
  summarise_each((funs(sum(is.na(.)))))
```

```
##   genre type rating rating_10 rating_100 rating_1000
## 1    62   25    230       230        230         230
```

```
dat %>%
  select_if(function(x) any(is.na(x))) %>%
  summarise_each((funs(100*mean(is.na(.)))))
```

```
##      genre      type   rating rating_10 rating_100 rating_1000
## 1 0.504311 0.2033512 1.870831  1.870831   1.870831    1.870831
```

On observe que les variables manquantes se concentrent sur les variables [genre,type,rating,ratin_10,rating_100,rating_1000] On doit gérer les NAs pour pouvoir travailler avec la base de données. On remarque par la même occasion que la variable episode est de type character.

```
dat <- dat %>%
         drop_na()
```

```
dat %>%
    select(episodes) %>%
    distinct()
```

```
##     episodes
## 1          1
## 2         10
## 3        100
## 4       1006
## 5        101
## 6        102
## 7        103
## 8        104
## 9        105
## 10       108
## 11       109
## 12        11
## 13       110
## 14       112
## 15       113
## 16       114
## 17       115
## 18       117
## 19       119
## 20        12
## 21       120
## 22       124
## 23       125
## 24       127
## 25      1274
## 26       128
## 27        13
## 28       130
```

```
## 29      1306
## 30       132
## 31       136
## 32       137
## 33        14
## 34       140
## 35       141
## 36       142
## 37      1428
## 38       143
## 39       145
## 40       147
## 41      1471
## 42       148
## 43        15
## 44       150
## 45       151
## 46       153
## 47       154
## 48       155
## 49       156
## 50      1565
## 51        16
## 52       161
## 53       162
## 54       163
## 55       164
## 56       167
## 57        17
## 58       170
## 59       172
## 60       175
## 61       178
## 62      1787
## 63        18
## 64       180
## 65      1818
## 66       182
## 67        19
## 68       191
## 69       192
## 70       193
## 71       195
## 72       199
## 73         2
## 74        20
## 75       200
## 76       201
## 77       203
## 78        21
## 79        22
## 80       220
## 81       224
## 82       225
```

```
## 83         23
## 84        237
## 85         24
## 86        240
## 87        243
## 88         25
## 89         26
## 90        260
## 91        263
## 92         27
## 93        276
## 94         28
## 95        283
## 96         29
## 97        291
## 98        296
## 99          3
## 100        30
## 101       300
## 102       305
## 103        31
## 104       312
## 105        32
## 106        33
## 107       330
## 108       331
## 109        34
## 110        35
## 111       358
## 112        36
## 113       365
## 114       366
## 115        37
## 116       373
## 117        38
## 118        39
## 119         4
## 120        40
## 121        41
## 122        42
## 123        43
## 124        44
## 125        45
## 126        46
## 127        47
## 128       475
## 129        48
## 130        49
## 131         5
## 132        50
## 133        51
## 134       510
## 135        52
## 136       526
```

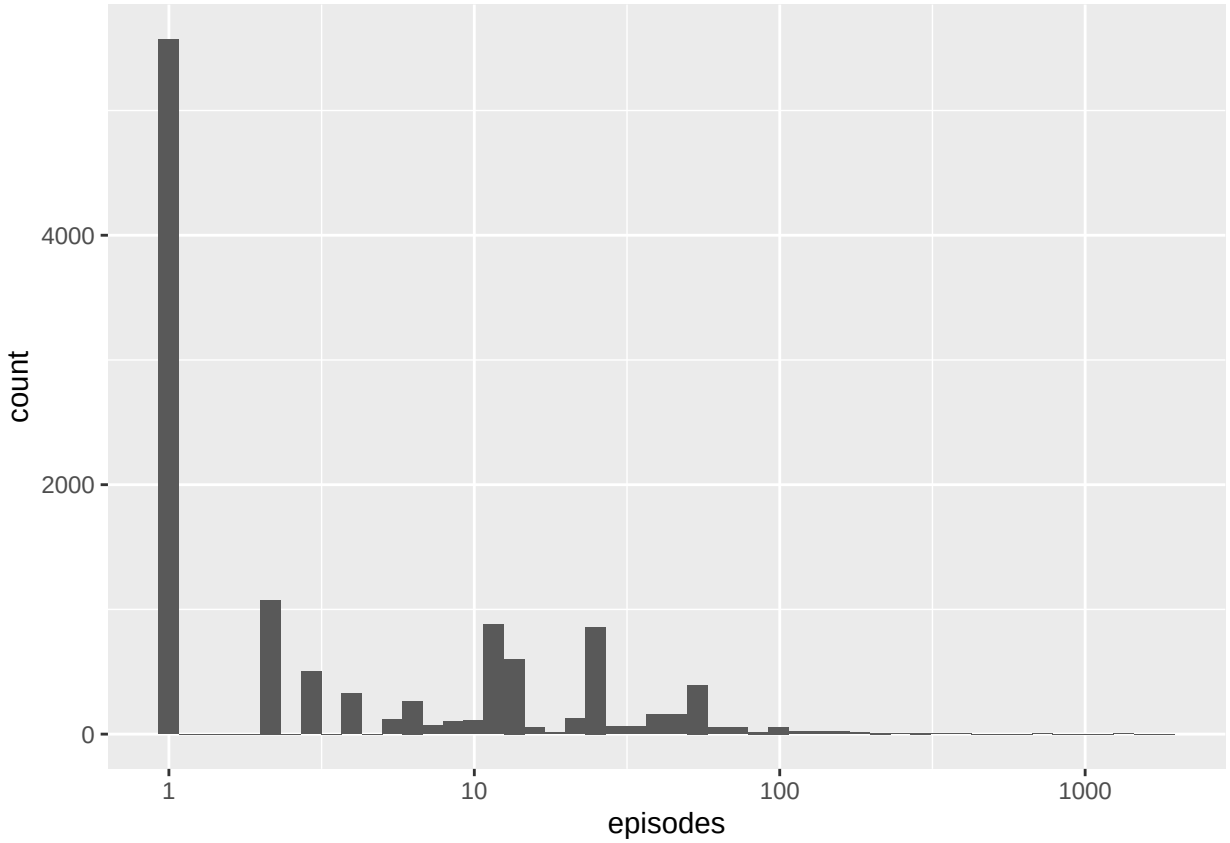
```
## 137       53
## 138       54
## 139       55
## 140       56
## 141       58
## 142       59
## 143        6
## 144       60
## 145       61
## 146       62
## 147       63
## 148       64
## 149       65
## 150       66
## 151       67
## 152       68
## 153       69
## 154      694
## 155        7
## 156       70
## 157       71
## 158       72
## 159      726
## 160       73
## 161       74
## 162       75
## 163       76
## 164       77
## 165      773
## 166       78
## 167       79
## 168        8
## 169       80
## 170       83
## 171       84
## 172       85
## 173       86
## 174       87
## 175       88
## 176        9
## 177       90
## 178       91
## 179       92
## 180       93
## 181       94
## 182       95
## 183       96
## 184       97
## 185       98
## 186       99
## 187  Unknown
```

```r
dat <- dat %>%
              filter(!(episodes == "Unknown"))
```

On décide de supprimer les lignes où des valeurs manquantes sont présentes. On remarque que la variable prend la valeur “Unknown”. On décide de supprimer les lignes où les épisodes sont “Unknown”.

3. Représenter graphiquement la distribution du nombre d’épisodes.

```
dat <- dat %>%
          mutate(episodes = as.numeric(episodes))
dat %>%
        group_by(anime_id) %>%
        ggplot(., aes(x = episodes)) +
        geom_histogram(bins = 50) +
        scale_x_log10()
```



4. Combien d’animes n’ont pas le genre « Shounen » ?

```
dat %>%
  filter( !grepl("Shounen",genre)) %>%
  tally
```

```
##       n
## 1 10100
```

Il y a 10 100 anime qui n’ont pas le genre “Shounen”.

5. Donner la proportion de « Shounen » au sein de chaque type d’anime.

```
dat %>%
  filter(grepl("Shounen",genre)) %>%
  group_by(type) %>%
```

```
  dplyr::summarize(n = n()) %>% # On précise la librairie où on cherche la fonction
  mutate(freq = n / sum(n))     # summarize pcq conflit avec une autre librairie
```

```
## # A tibble: 5 x 3
##   type        n   freq
##   <chr>   <int>  <dbl>
## 1 Movie     372 0.215
## 2 ONA        20 0.0116
## 3 OVA       367 0.212
## 4 Special   262 0.151
## 5 TV        709 0.410
```

On a pour les types suivants [Movie,ONA,OVA,Special,TV] les proportions associées [0.21, 0.01, 0.21, 0.15, 0.40] respictivement.

6. Créer une fonction permettant de donner la proportion d'un genre quelconque au sein de chaque type d'anime.

```
prop_table_by_genre <- function(genre_){

              prop_table <- dat %>%
                            filter(grepl({{ genre_ }},genre)) %>%
                            group_by(type) %>%
                            dplyr::summarize(n = n()) %>%
                            mutate(freq = n / sum(n))


              return(prop_table)
}
```

7. Proposer une représentation graphique permettant d'observer au sein de chaque type d'anime, quels genres sont les plus représentés.

```
dat %>%
            separate_rows(genre, sep = ",") %>%
            select(genre) %>%
            mutate(genre = str_trim(genre, side = "both")) %>%
            distinct()
```

```
## # A tibble: 43 x 1
##    genre
##    <chr>
##  1 Drama
##  2 Romance
##  3 School
##  4 Supernatural
##  5 Action
##  6 Comedy
##  7 Historical
##  8 Parody
##  9 Samurai
## 10 Sci-Fi
## # ... with 33 more rows
```

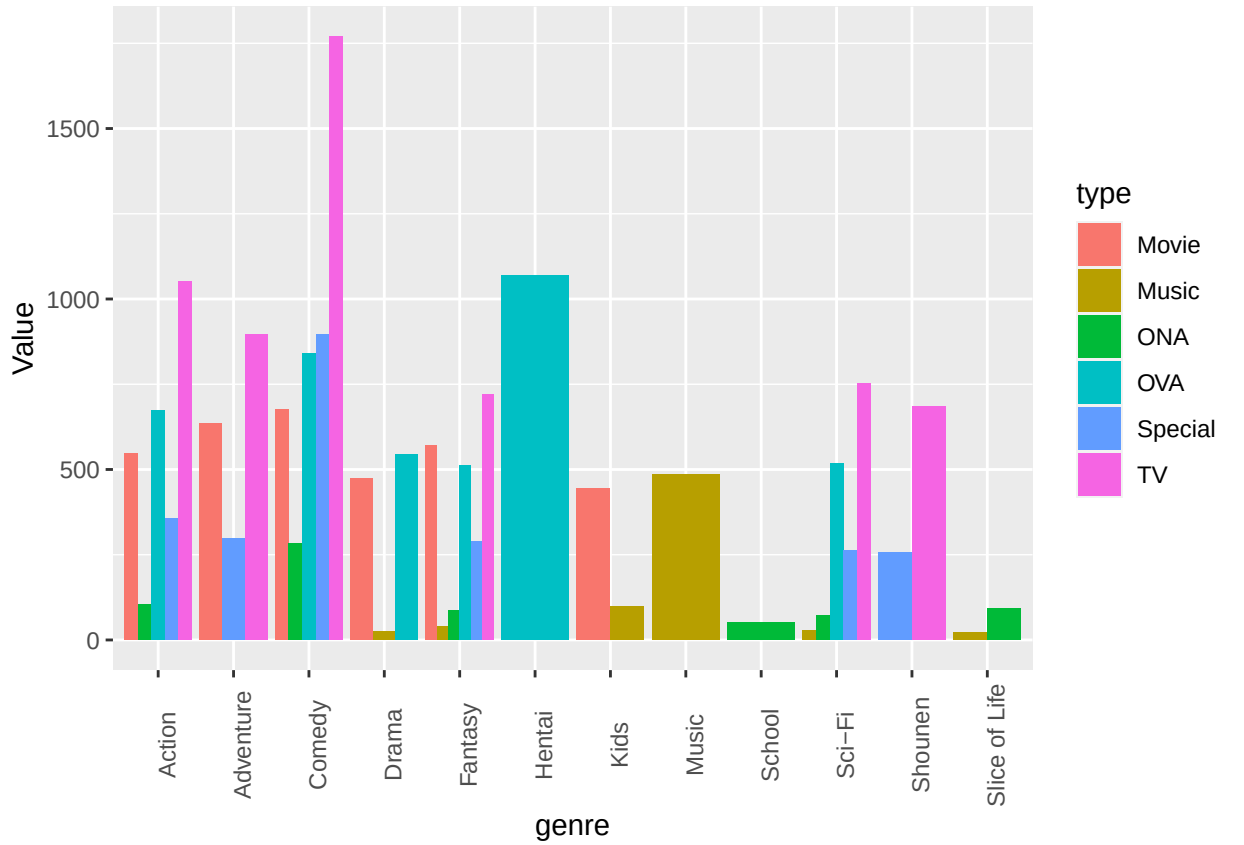
```
dat %>% select(type) %>% distinct()
```

```
##      type
```

```
## 1   Movie
## 2 Special
## 3     OVA
## 4   Music
## 5     ONA
## 6      TV
```

Dans le code ci-dessous on identfie les différents genre et type présent dans la base de données. On observe qu'il existe plus de 43 genres et 6 types différents.

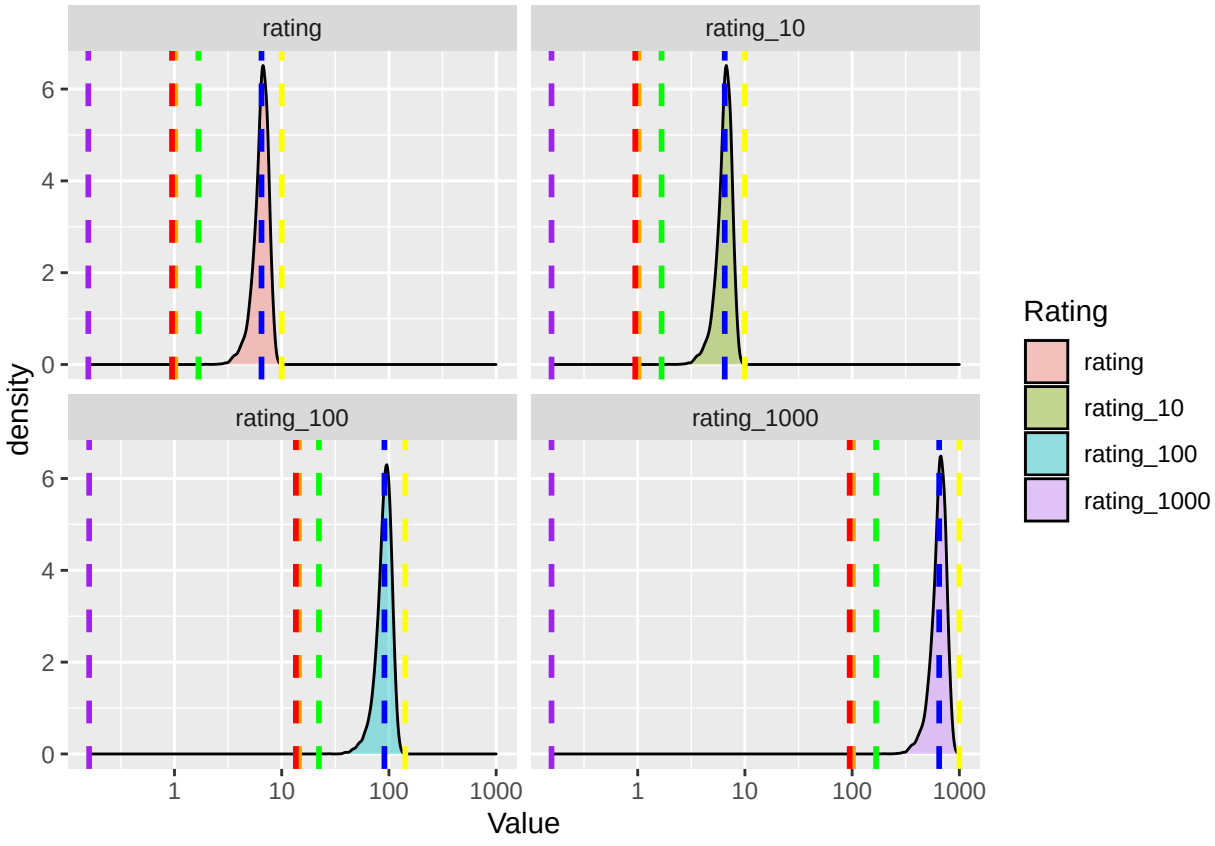
```
dat  %>%
        separate_rows(genre, sep = ",") %>%
        mutate(genre = str_trim(genre, side = "both")) %>%
        select(genre,type) %>% group_by(genre) %>%
        mutate(., value = 1) %>%
        pivot_wider(names_from = genre, values_from = value, values_fill = 0, values_fn = length) %>%
        gather(key = genre, value = Value, Drama:Yuri) %>%
        group_by(type) %>%
        slice_max(Value, n=6) %>%
        ggplot(., aes(genre, Value, fill = type)) + geom_col(position = "dodge") +
        theme(axis.text.x = element_text(angle = 90))
```



8. Pour toutes les variables de types « rating ». Calculer la moyenne, la médiane, l'écart type, la mad (median absolute deviation), le coefficient de variation, la valeur maximum et la valeur minimum. Représentez graphiquement ces résultats.

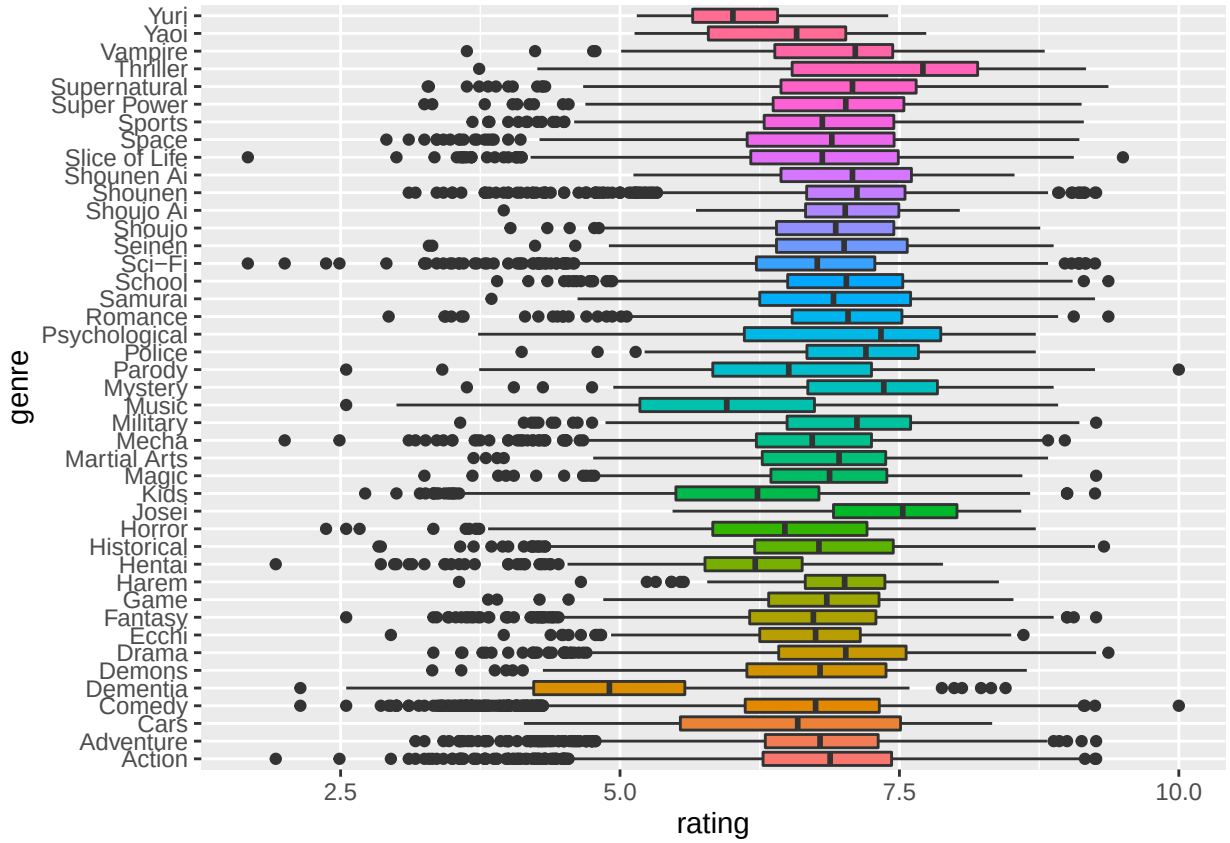
```
mu <- dat %>% select(rating, rating_10,rating_100,rating_1000) %>%
              gather(key = Rating, value = Value, rating:rating_1000) %>%
```

```
              group_by(Rating) %>%
              dplyr::summarize(grp.mean = mean(Value),
                               grp.mad = mad(Value),
                               grp.sd  = sd(Value),
                               grp.max = max(Value),
                               grp.min = min(Value),
                               grp.cv = goeveg::cv(Value)) #import de la librairies goeveg pour la fut c



dat %>%  select(rating, rating_10,rating_100,rating_1000) %>%
         gather(key = Rating, value = Value, rating:rating_1000) %>%
         ggplot(., aes(x=Value, group=Rating, fill=Rating)) +
         geom_density(adjust=1.5, alpha=.4) +
         facet_wrap(~Rating) +
         scale_x_log10() +
         geom_vline(aes(xintercept = grp.mean), data = mu, color="blue", linetype="dashed", size=1) +
  geom_vline(aes(xintercept = grp.sd), data = mu, color="orange", linetype="dashed", size=1) +
  geom_vline(aes(xintercept = grp.mad), data = mu, color="red", linetype="dashed", size=1) +
  geom_vline(aes(xintercept = grp.max), data = mu, color="yellow", linetype="dashed", size=1) +
  geom_vline(aes(xintercept = grp.min), data = mu, color="green", linetype="dashed", size=1) +
  geom_vline(aes(xintercept = grp.cv), data = mu, color="purple", linetype="dashed", size=1)
```



9. Proposer une représentation graphique permettant d'observer s'il existe une différence de notation en fonction du genre d'anime.

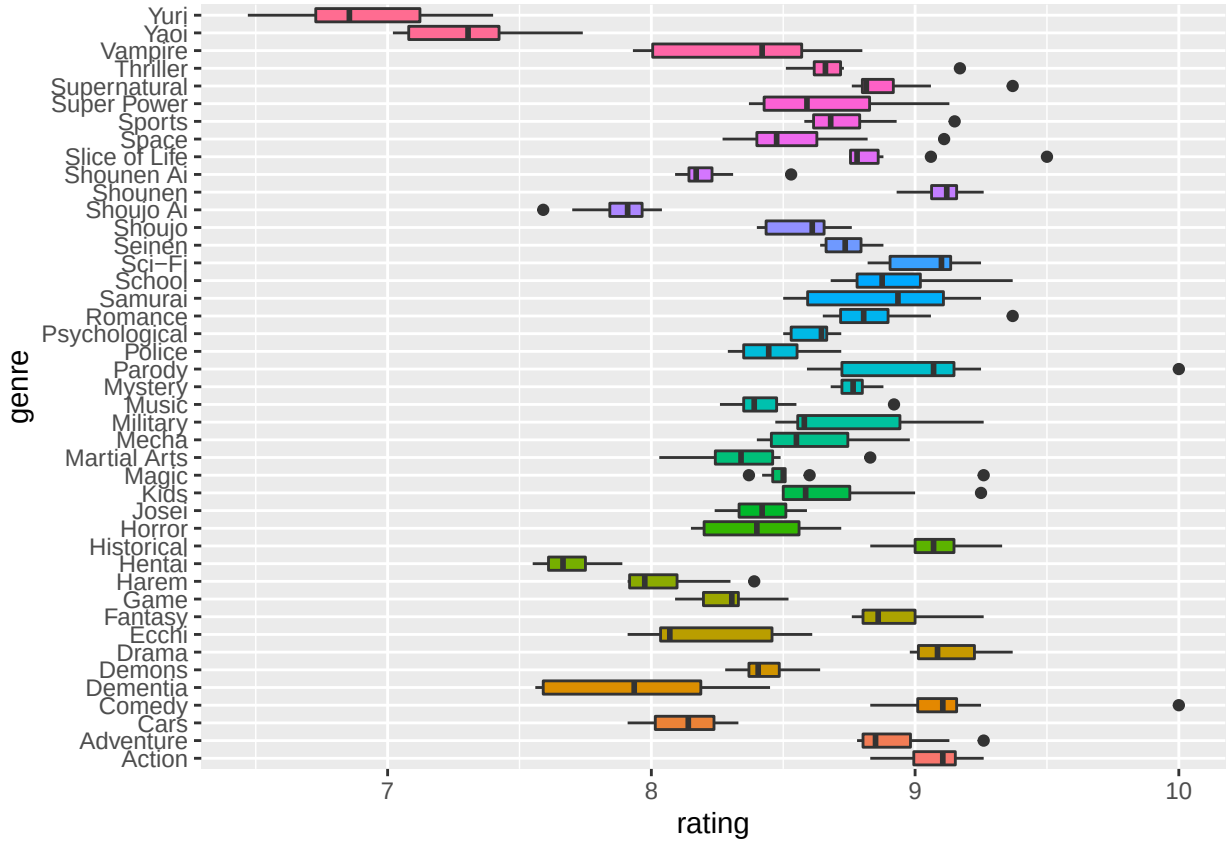
```r
dat %>%
        separate_rows(genre, sep = ",") %>%
        mutate(genre = str_trim(genre, side = "both")) %>%
        ggplot(., mapping = aes(x = genre, y = rating, fill = genre)) +
        geom_boxplot(alpha=25) +
        coord_flip() +
        theme(legend.position = "none")
```



On propose un BOXPLOT pour pouvoir observer la différence de notation entre les différents genres.

10. Pour chaque genre d'anime, représenter graphiquement le top 10 des animes les mieux notés.

```r
dat %>%
        separate_rows(genre, sep = ",") %>%
        mutate(genre = str_trim(genre, side = "both")) %>%
        group_by(genre) %>%
        slice_max(rating, n=10) %>%
        ggplot(., mapping = aes(x = genre, y = rating, fill = genre)) +
        geom_boxplot(alpha=25) +
        coord_flip() +
        theme(legend.position = "none")
```



11. Etablir une critique sur les données et les statistiques que vous avez produites. Proposer éventuellement d'autres analyses pour compléter.

La première difficulté de cette base de données était le fait qu'elle soit répartit sur plusieurs fichier qu'il a fallut réunir pour pouvoir l'étudier. En ce qui concerne l'exploration de la base de donnée elle ne présentait que très peu valeurs manquantes. Le faible nombre de variable ne permet pas d'avoir davantage d'information sur cette base de donnée. Il aurait peut-être avoir une autre source de données pour compléter notre base de donnée.

PARTIE 2

Consigne :

```
Proposer une application shiny avec ces données. Vous mettrez en avant l'utilité de votre appl
L'application devra être déployée sur shinyapps.io.
```

Mon application a pour but de proposer à l'utilisateur une selection de film les mieux notés selon ses choix.

L'application fonctionne par plateau. dans un premier il peut choisir selon le type d'anime qu'il préfère quels genre d'anime il peut trouver en majorité. Par la suite il peut regarder si dans le genre qu'il a choisi les notations des films sont plutôt bon ou mauvais. Au final il choisi selon le type et le genre choisi aux étapes précédents sur liste d'anime dans une tranche de note de son souhait.

Voici le code qui à servit a dévelloper notre application shiny : https://marwanouzaid.shinyapps.io/projet_r/