Manipulation avancees

Contents

Introduction

On s'intéresse ici à des opérations plus complexes. On va commencer par réaliser des résumés de données pour un des sous groupes de caractéristiques. L'idée étant d'obtenir des tableaux du type :

Une fois ces "nouvelles" données obtenues , on les ajouteras aux données de départ.

Agréger des données

On s'intéresse au nombre de scenes. L'objectif est ici de connaitre , pour chaque acteur le nombre de scene où il apparait afin de comparer des temps de présences. La durée moyenne d'une sequence par episode ainsi que le nombre moyen de mort au cours d'une séance par épisode, sont aussi calculées

Commande de base

Il existe dans R une commande `aggregate`. Elle subdivise le jeu de données en sous groupe, calcul les statistiques sur les sous groupes puis elle remet les sous groupes ensemble.

Ainsi pour calculer la durée moyenne par episodes

```
Moy_episode<-aggregate(list(duree_moyenne=episodes$total_duration),by=list(episode=episodes$episodeId),
                FUN=mean)
```

```
aggregate(list(duree_moyenne=scenes$duration, nb_morts_moy=scenes$nbdeath),by=list(episodes=scenes$epis
                FUN=mean)
```

```
## Selecting by nb_morts_moy
```

```
##   episodes duree_moyenne nb_morts_moy
## 1       33      60.50943    0.2452830
## 2       51      70.33333    0.3846154
## 3       52      68.45455    0.3409091
## 4       53      79.16216    0.2972973
```

Summarise

Résumé une seule variable

```
resume<-summarise (scenes, duree_moy=mean(duration))
```

On résume plus de deux variables à l'aide de summarise

```
resume<-summarise (scenes, duree_moy_seq=mean(duration),nb_morts_moy=mean(nbdeath))
```

description des sequences par episodes

duree_moy_seq

nb_morts_moy

seq_long

60.91484

0.0979167

661

Le résultat est une seule ligne , on a les moyennes et le maximum pour tout le tableau et non par épisode . On va donc utiliser la commande `group_by`

Group by

On calcule pour chaque personnage le nombre d'apparition.

```
apparitions2<-apparitions%>%
  group_by(name)%>%
  summarise(nb_scene = n())
```

```
## Selecting by nb_scene
```

description des sequences par episodes ; premieres lignes du tableau

name

nb_scene

Arya Stark

360

Bran Stark

248

Cersei Lannister

330

Daenerys Targaryen

509

Davos Seaworth

256

Jaime Lannister

330

Jon Snow

632

Jorah Mormont

257

Sansa Stark

349

Tyrion Lannister

544

et alors . . . On regarde aussi le temps moyen par episodes pour chaque saison ?

Fusionner des données : les jointures

On va ici se servir de la jointure pour lier les données et leur traduction

```
trad<-read.csv(here(chemin_donnees,"tranlate.csv"),sep=";")
```

Traduction de google

```
##                    En             Fr
## 1            Astapor        Astapor
## 2            Braavos        Braavos
## 3              Dorne          Dorne
## 4            Meereen        Meereen
## 5 North of the Wall Au nord du mur
## 6             Pentos         Pentos
```

Jointure avec la fonction 'merge'

```
scenes_fr<-merge(scenes,trad,by.x="location",by.y="En")
```

On a donc une colonne en francais

```
head(scenes_fr)
```

```
##   location sceneStart sceneEnd subLocation episodeId duration nbc sceneId
## 1  Astapor    0:46:40  0:49:06        <NA>        24      146   7     888
## 2  Astapor    0:34:40  0:35:15        <NA>        23       35   4     830
## 3  Astapor    0:49:06  0:49:23        <NA>        24       17   5     889
## 4  Astapor    0:50:13  0:51:24        <NA>        24       71   4     892
## 5  Astapor    0:49:23  0:49:30        <NA>        24        7   2     890
## 6  Astapor    0:45:31  0:46:40        <NA>        24       69   6     887
##   nbdeath      Fr
## 1       0 Astapor
## 2       0 Astapor
## 3       1 Astapor
## 4       0 Astapor
## 5       0 Astapor
## 6       0 Astapor
```

Les jointures avec tidyverse

On pourra se referer à ce billet relativement complet (surtout pour les dessins !)

L'avantage tient en

1. une fonction par type de jointure
2. On peut faire des jointures en utilisant plusieurs variables pour avoir la clé primaire
3. c'est plus rapide

```
join<-inner_join(scenes,trad,by=c("location"="En"))
```

Mettre en forme des données répétée

pivot_wider() est utilisé pour changer le sens des données répétées pivoter une ligne/tps par une ligne et tous les temps en colonne

```
loc_got <- scenes %>%
  group_by(episodeId) %>%
  mutate(rang=row_number()) %>%
  select(episodeId,location,rang) %>%
  pivot_wider(names_from = rang, values_from = location)
```

Next level : Combiner des commandes

Ici , on combine pour obtenir un tableau utilisable pour faire des analyses de reseaux. Le tableau créé permet de connaître le temps passé ensemble par une paire d'acteurs.

```
lien<-apparitions %>% left_join(apparitions,by=c("sceneId"="sceneId")) %>%
  filter(name.x!=name.y) %>%
  left_join(scenes %>% select(sceneId,duration)) %>%
  group_by(name.x,name.y) %>%
  summarise(commonTime=sum(duration)) %>%
  arrange(desc(commonTime))
```

```
## Joining, by = "sceneId"
## `summarise()` has grouped output by 'name.x'. You can override using the
## `.groups` argument.
```

```
lien
```

```
## # A tibble: 6 x 3
## # Groups:   name.x [6]
##   name.x             name.y             commonTime
##   <chr>              <chr>                   <int>
## 1 Daenerys Targaryen Jorah Mormont           12923
## 2 Jorah Mormont      Daenerys Targaryen      12923
## 3 Lord Varys         Tyrion Lannister        10764
## 4 Tyrion Lannister   Lord Varys              10764
## 5 Davos Seaworth     Jon Snow                10380
## 6 Jon Snow           Davos Seaworth          10380
```