Janitor

Zakaria RIDADARAJAT

02/02/2021

Entête :

Nous allons explorer dans ce document le package Janitor faite par Olfa LAMTI

Voici le lien vers son GitHub est : **https://github.com/OlfaLmt/PSBX/tree/main/Janitor**

Synthèse :

Le package JANITOR intervient dans l'étape du Preprocessing des datasets. En effet ce package offre plusieurs fonctionnalités tel que le formatages colonnes d'un dataframe, calcul des tableaux de fréquences et tableaux croisés et isolement des doublons.

Pour mieux comprendre le fonctionnement de ce package, nous allons suivre sur l'exemple mentionné sur le rapport de Olfa LAMTI.

Vu que la non-disponibilité que la table de donnée avec laquelle Olfa a travaillé, je me suis limité à la description des fonctions liées au package Janitor.

Extrait commenté :

```
# NETTOYONS LES NOMS AVEC **clean_names()**

x = janitor::clean_names(mymsa)

data.frame(mymsa = colnames(mymsa), x = colnames(x))
```

Les :: nous aide ici à appeler la fonction clean_names du package Janitor. Cette fonction fournit en une sortie un tableau avec les intitules de colonnes remis au même format.

```
#OBTENIR LA FREQUENCE D'UN VECTEUR EN UTLISANT **tabyL()**

#Sur ce meme data frame remis au bon format nous allons chercher a étudier la frequende d'un vecteur.

#Faisons un test sur meat_colour.

tabyl(x, meat_colour)
```

Ici la fonction tabyl() nous permet d'étudier les fréquences d'un vecteur d'une data frame.

```
#ENLEVER LES COLONNES VIDES

#Ici nous avons besoin de la fonction **remove_empty()**.

x = remove_empty(x, which = c("rows","cols"))
```

La fonction remove_empty nous permet d'enlever les valeurs manquantes d'une data frame.

```
#FORMATTAGE DES DONNEES AVEC **adorn_pct_formatting()**

x %>%

tabyl(meat_colour, plant) %>%

adorn_totals(where = c("row","col")) %>%

adorn_percentages(denominator = "col") %>%

adorn_pct_formatting(digits = 0)
```

L'instruction ci-dessus nous donne affiche les valeurs numériques en pourcentage.

```
#GET DUPES

#Afin d'etudier les doublons nous regardons pour chaque donnée si nous

#disposons d'un identifiant unique et nous allons chercher s'il existe une #version doublonnée de cet i

x %>% get_dupes(rfid)
```

La fonction get_dups() permet de récupérer les doublons d'une data Frame.

Evaluation :

1) Exécution du fichier rmd:

La non-disponibilité du jeu de données.

2) Qualité de rédaction :

Pour moi il s'agit d'un bon travail. L'auteur fournit un rapport bien organisé et totalement facile à lire.

3) Aspect didactique

Le document est parfaitement clair.je trouve que les descriptions des fonctions sont compréhensibles a tous niveau et que les descriptions sont bien détaillées et illustrées dans le code.

4) Lisibilité du rmarkdown:

Le code est bien organisé et expliqué et le rmarkdown est bien fait.

5) Bibliographie :

Non, il y'a pas bibliographie.

Conclusion :

À mon sens, il s'agit globalement d'un bon travail. L'auteur fournit un dossier documenté et facile à lire. On y apprend des choses qui sont importantes durant la phase du preprocessing.