Tutoriel Lubridate de Gaspard PALAY

Tarik HAKAM

22/12/2020

1. Critères d'évaluation

1. Comportement du Rmd à l'exécution
2. Qualité de la rédaction du dossier
3. Accessibilité, didactisme et pertinence du dossier
4. Qualité et lisibilité du Rmarkdown
5. Qualité des applications permettant d'illustrer le package

2. Lien vers le document commenté

En cliquant **ici**, vous trouverez le lien menant au GitHub de Gaspard PALAY hébergeant le fruit de sa réalisation.

3. Auteurs du document commenté

Le document évalué dans le cadre de ce rendu a été produit par Gaspard PALAY, étudiant en Master of Science Data Management à Paris School of Business.

4. Synthèse du document

L'objectif de ce document est d'expliciter et de présenter comment gérer des données d'horodatage avec le package **lubridate**.

Le package **lubridate** a été spécialement conçu pour gérer les données de temps et les rendre facilement manipulables.

A noter que **lubridate** est disponible avec la collection de package **TidyVerse** en open source.

5. Extrait commenté des parties de code

Ici, nous allons commenté la partie de code concernant la manipulation des dates/heures.

- La fonction `ceiling_date()` permet d'arrondir l'heure de la date à l'heure supérieure :

```
t <- ymd_hms("2020.11.09_17.56.32")
ceiling_date(t,"hour")
```

⇒ La sortie affichera : `"2020-11-09 18:00:00 UTC"`

- La fonction `floor_date()` permet d'arrondir l'heure de la date à l'heure inférieure :

```
t <- ymd_hms("2020.11.09_17.56.32")
floor_date(t,"hour")
```

⇒ La sortie affichera : `"2020-11-09 17:00:00 UTC"`

- La fonction `round_date()` permet d'arrondir l'heure de la date à l'heure la plus proche :

```
t <- ymd_hms("2020.11.09_17.56.32")
round_date(t,"hour")
```

⇒ La sortie affichera : `"2020-11-09 18:00:00 UTC"`

Ainsi de la même manière, on peut arrondir respectivement au plus proche avec la fonction `round_date()`:

- les minutes
- les heures
- le jour
- le mois
- l'année

```
t <- ymd_hms("2019.04.11 14h37min52s")
round_date(t,"minute")
round_date(t,"hour")
round_date(t,"day")
round_date(t,"month")
round_date(t,"year")
```

Dans un second temps, nous allons commenté la partie de code concernant le calcul des périodes de temps écoulées ou de durée.

Ici l'auteur définit tout d'abord respectivement les dates t1 et t2, telles que :

- `t1 <- dmy("12/09/2020")`
- `t2 <- dmy("30/01/2016")`

la fonction "`diff`" nous renseigne sur la "différence de temps" entre t1 et t2. Il s'agit d'un objet de classe difftime

```
t1 <- dmy("12/09/2020")
t2 <- dmy("30/01/2016")
diff <- t1-t2
as.duration(diff)
as.period(diff)
```

- `as.duration(diff)` affiche la différence de temps en secondes, ici : `"145756800s (~4.62 years)"`

- `as.period(diff)` affiche la différence de temps en jours, en heures, en minutes et en secondes, ici : `"1687d 0H 0M 0S"`

Ici, l'auteur souhaite afficher la récurrence d'un évènement, comme dans cet exemple, tous les 38 ans. Il utilise donc la fonction suivante en ne considérant uniquement que les années :

```
t0 <- dmy_h("12/09/2020 22h")
t0+years(38)
```

⇒ La sortie de la commande `t0+years(38)` affichera la date avec 38 années de plus, ici : `"2058-09-12 22:00:00 UTC"`

Cependant, il a noté que le résultat est inexact.

La fonction prédit un événement le 12/09 sans prendre en compte les années bissextiles.

Pour obtenir le bon résultat, il faut réaliser la commande suivante en tenant compte des jours écoulés sur les 38 années :

```
t0+dyears(38)
```

Pour savoir si une année est bissextile, la fonction `leap_year()` permet d'afficher pour une année choisie un booléen (TRUE / FALSE) :

```
leap_year(2016)
leap_year(2017)
```

Les sorties affichées sont les suivantes :

- `leap_year(2016)` ⇒ TRUE
- `leap_year(2017)` ⇒ FALSE

Pour finir, l'auteur aborde le sujet de l'heure POSIX ou heure Unix.

Les instants sont des moments précis du temps. **Date**, **POSIXct** et **POSIXlt** sont les trois classes d'objets que la base R reconnaît comme des instants.

Chaque objet POSIXct est enregistré comme le nombre de secondes qui s'est ecoulées depuis le 1970-01-01 00:00:00.

- `is.Date()` vérifie si un objet hérite de la classe **Date**.

```
is.Date(as.Date("2009-08-03"))
```

⇒ Ici, la sortie affichera `TRUE`.

- `is.POSIXt()` vérifie si un objet hérite des classes **POSIXlt** ou **POSIXct.**
- `is.instant()` vérifie si un objet hérite de l'une des trois classes.

```
is.POSIXt(as.Date("2009-08-03"))
is.POSIXt(as.POSIXct("2009-08-03"))
```

- `now()` renvoie l'heure système actuelle sous la forme d'un objet POSIXct.

```
now(tzone = "")
```

- `today()` renvoie la date système actuelle.

```
today(tzone = "")
```

6. Evaluation du travail suivant les 5 critères précités

1. Comportement du Rmd à l'exécution

Sur ma machine, le fichier Rmd ne s'exécute pas sans erreur si nous souhaitons exécuter les chunks pour obtenir les différentes sorties.

Il ne s'exécute pas non plus, même en n'exécutant pas les chunks mais en les incluant. Cela est dû à un oubli de la part de l'auteur en ligne 145. Un chunk s'exécute sans avoir exécuter le reste de code. Il aurait été pertinent d'intégrer ce morceau de code : "`{r,echo=TRUE,eval=FALSE}`.

2. Qualité de la rédaction du dossier

La rédaction de ce document est correcte.

A mon sens, afin de finaliser sa pensée, l'auteur aurait dû ajouter des traces d'exécution de son code dans son fichier PDF de façon à illustrer ses exemples. Autrement, le contenu se rattrape avec des explications claires.

3. Accessibilité, didactisme et pertinence du dossier

La lecture de ce dossier est relativement aisée et accessible.

Les descriptions sont bien explicitées et bien illustrées.

L'auteur arrive à faire adhérer le lecteur à sa production. Le document a vocation à transmettre une connaissance et le but, pour ma part, est atteint.

Le sujet est pertinent et a vocation à servir de mémo pour de futures réalisations.

4. Qualité et lisibilité du RMarkdown

Le RMarkdown est bien écrit hormis quelques inattentions. Il reste toutefois lisible et aéré.

J'aurais éventuellement qu'une seule remarque :

- J'aurais, pour ma part, exécuter le code dans le document RMarkdown pour rendre plus facile encore la lecture du document PDF final afin de visualiser les différentes sorties. Autrement, il s'agit d'un code bien réalisé.

5. Qualité des applications permettant d'illustrer le package

Une erreur en ligne 80 ⇒ `round_date()` au lieu de `floor_date()`. Cela induit une redondance dans le code en ligne 85.

Les applications sont toutefois de bonne qualité et maitrisées.

7. Conclusion

Selon moi, il s'agit globalement d'un bon travail.

L'auteur fournit un dossier recherché, documenté et globalement facile à lire.

On y apprend des choses nouvelles. Personnellement, j'aurais aimé y retrouver des traces d'exécution et par conséquent les résultats des sorties.

Le document se rattrape par des explications lexicales claires.

Vous retrouvez ce document sur mon **GitHub**.