kadaoui_alexandre_Lubridate

Alexandre

23/12/2020

R Markdown

Dans le cadre de notre partiel, nous devons réaliser un total de 12 travaux retracant notre parcours et notre travail durant les 30 heures de cours.
Le travail à faire est le suivant :
- Une entête comportant un titre, un lien Github avec le ou les noms des auteurs.
- Une synthese de ce travail
- Un extrait commenté avec des parties de codes clé avec explication et commentaire.
- Une évalutation du travail avec nos 5 criteres.
- Une conclusion du travail

Definition des 5 critères de notations :

1) Effort de présentation :
2) Le knit est réalisable et bien présenté.
3) Explications simples et efficaces.
4) Le Code reproductible à d'autres DataFrame avec facilité.
5) Description des fonctions utilsés et du raisonnement.

Tutoriel Lubridate

Travail réalisé par " Gaspard PALAY " le 11/11/2020.

GitHub: https://github.com/GaspardPalay/PSBX/tree/main/TutorielLubridate (https://github.com/GaspardPalay/PSBX/tree/main/TutorielLubridate)

Synthese :

Le package Lubridate a pour objectif d'harmoniser les dates et temps (heures, minutes, secondes) au même format afin que toutes les observations soient comparables.
Ce package permet également de convertir ces dates et données de temps généralement considérées comme des strings dans un format que R sera capable de reconnaitre comme des mesures de temps (notamment les formats "POSIXct" et "Date")
Une fois reconnues comme des données de temps par R, il sera ensuite possible de manipuler et analyser ces données via R

Extrait commenté du code :

```
library(lubridate) #inclus dans le package tidyverse
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

Via **lubridate::dmy** on convertit le 30 mai 2020 en date au format Date Month Year (DMY)

```
jourJ <- lubridate::dmy("30 may 2020")
class(jourJ)
```

```
## [1] "Date"
```

```
jourJ
```

```
## [1] "2020-05-30"
```

Conversion d'un vecteur cnotenant date et heure sous forme de string en en format "date-time" interprétable par R

```
ymd("2019/04_11")
```

```
## [1] "2019-04-11"
```

```
ymd_hm("2019.04.11 14h37")
```

```
## [1] "2019-04-11 14:37:00 UTC"
```

```
ymd_hms("20190407143752")
```

```
## [1] "2019-04-07 14:37:52 UTC"
```

```
hms("14h37min52s")
```

```
## [1] "14H 37M 52S"
```

Récupération d'éléments précis tels que la date ou l'heure dans un format "date-time"

```
t <- ymd_hms("2020.11.09_17.56.32")
date(t)
```

```
## [1] "2020-11-09"
```

```
hour(t)
```

```
## [1] 17
```

```
minute(t)
```

```
## [1] 56
```

```
second(t)
```

```
## [1] 32
```

Déterminer l'écart entre deux dates et/ou heures entre deux temps donnés via la fonction **diff**

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

```
## [1] "145756800s (~4.62 years)"
```

```
as.period(diff)
```

```
## [1] "1687d 0H 0M 0S"
```

Manipulation de dates, ajout de mois, jours, semaines, heures etc…

```
t1+months(9) # On ajoute 9 mois à la date t1
```

```
## [1] "2021-06-12"
```

```
t1+ddays(287) # On ajoute 287 jours à la date t1
```

```
## [1] "2021-06-26"
```

```
ddays(287)/dweeks(1) # On calcule le nombre de semaines que contituent 287 jours
```

```
## [1] 41
```

```
t2-dweeks(7) # On soustrait 7 semaine à la date t2
```

```
## [1] "2015-12-12"
```

La fonction **interval** permet de ernvoyer l'interval de temps entre deux instants précis (ici on prend également en compte les fuseaux horaires, ou timezones, grâce au terme **tz**)

Ici on détermine l'interval de temps entre un départ de Dakar en Afrique et une arrivée à Toronto tout en prenant en compte les fuseaux horaires des deux lieux.

```
date_depart <- dmy_hm("27/12/2016 5:45", tz="Africa/Dakar")
date_arrive <- mdy_hm("dec 28, 2017  19:30", tz="America/Toronto")
duree <- interval(date_depart, date_arrive)
duree
```

```
## [1] 2016-12-27 05:45:00 GMT--2017-12-29 00:30:00 GMT
```

Evaluation du travail :

1) Effort de présentation : Le tuto est très bien réalisé, bien présenté et facilement lisible
2) Le knit est réalisable et bien présenté : Le knit est facile à réaliser et ne rpésente aucun conflit
3) Explications simples et efficaces : Les explications du codes et des divers concepts (tels que l'heure POSIX par exemple) sont très clairs et simples à comprendre. Les diverses fonctions de codes ainisi que les termes qui les composent sont tous bien explicités et facilement compréhensibles
4) Le Code reproductible à d'autres DataFrame avec facilité : Les exemples utilisés ici dans le code sont très généralistes, le code est donc facilement reproductible et réadaptable pour une utilisation dans d'autres contextes.
5) Description des fonctions utilsés et du raisonnement : Les fonctoins clefs utilisées sont **lubridate::dmy** et ses variations pour la conversion de dates string en date interprétables par R ainsi que les fonctions **interval** ou encore **diff**.

Conclusion :

Ce tutoriel très bien réalisé et simple permet de facilement s'approprier les différentes fonctionnalités et usages du package Lubridate afin de manipuler les données datées via R.