Science des Données - Lab 2

Nolwenn Le Meur & Pascal Crépey

15 mars, 2022

Contents

1 Charger les données

Commencez par charger les données que vous avez sauvegardées lors du précédent atelier. Si vous avez bien les fichiers "baseLong.rds" et "baseAllHosp.rds", **utilisez la fonction `readRDS()` pour les lire**.

solution

```
baseLong <- readRDS("baseLong.rds")
baseAllHosp <- readRDS("baseAllHosp.rds")
```

2 Créer une série temporelle à partir des données

Pour pouvoir être décomposée, la série temporelle doit être d'abord transformée en un objet spécial avec la fonction `ts()` à partir des données. Cette fonction permet d'inclure l'échelle de temps de la mesure dans les données.

Il faut cependant définir la période de temps qui vous intéresse. Cela se fait en définissant la fréquence d'échantillonage. Par exemple, si vous avez des données quotidiennes et que vous voulez définir la période de temps en semaine, alors vous indiquerez `frequency = 7`.

Par exemple (si `x` contient les valeurs pour chaque pas de temps). serie1 <- ts(x, frequency=7)

Le paramètre `frequency` précise le nombre d'observations dans une période (i.e: 7 pour de l'hebdomadaire, 30.5 pour du mensuel, etc).

Utilisez la fonction `ts()` pour créer une variable `hospit_ts` qui contiendra une série temporelle des hospitalisations avec une fenêtre hebdomadaire à partir de `baseLong`.

solution

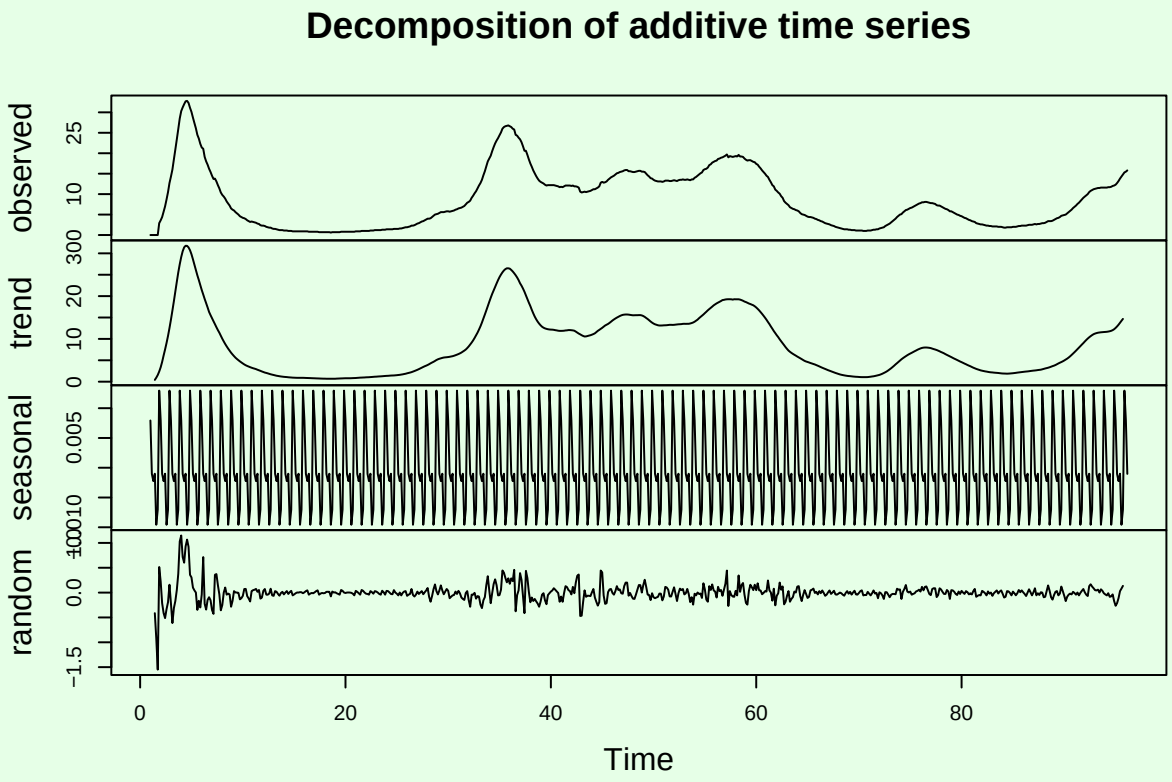
```r
#en R de base
baseH <- baseLong[baseLong$type == "hospitalisation",]

#en dplyr
baseH <- baseLong  |> filter(type == "hospitalisation")

#en data.table
#baseH = baseLong[type=="hospitalisation",]

myts <- ts(baseH$valeur,
           frequency = 7)
```

Cet objet `ts` est ensuite facilement exploitable par les outils de décomposition de séries temporelles.

Utilisez la fonction `decompose()` pour extraire la tendance, la saisonalité hebdomadaire et le résidu. Vous pouvez directement utiliser la fonction `plot()` sur cette décomposition.

solution

```r
mydec <- decompose(myts, type = "additive")

plot(mydec)
```

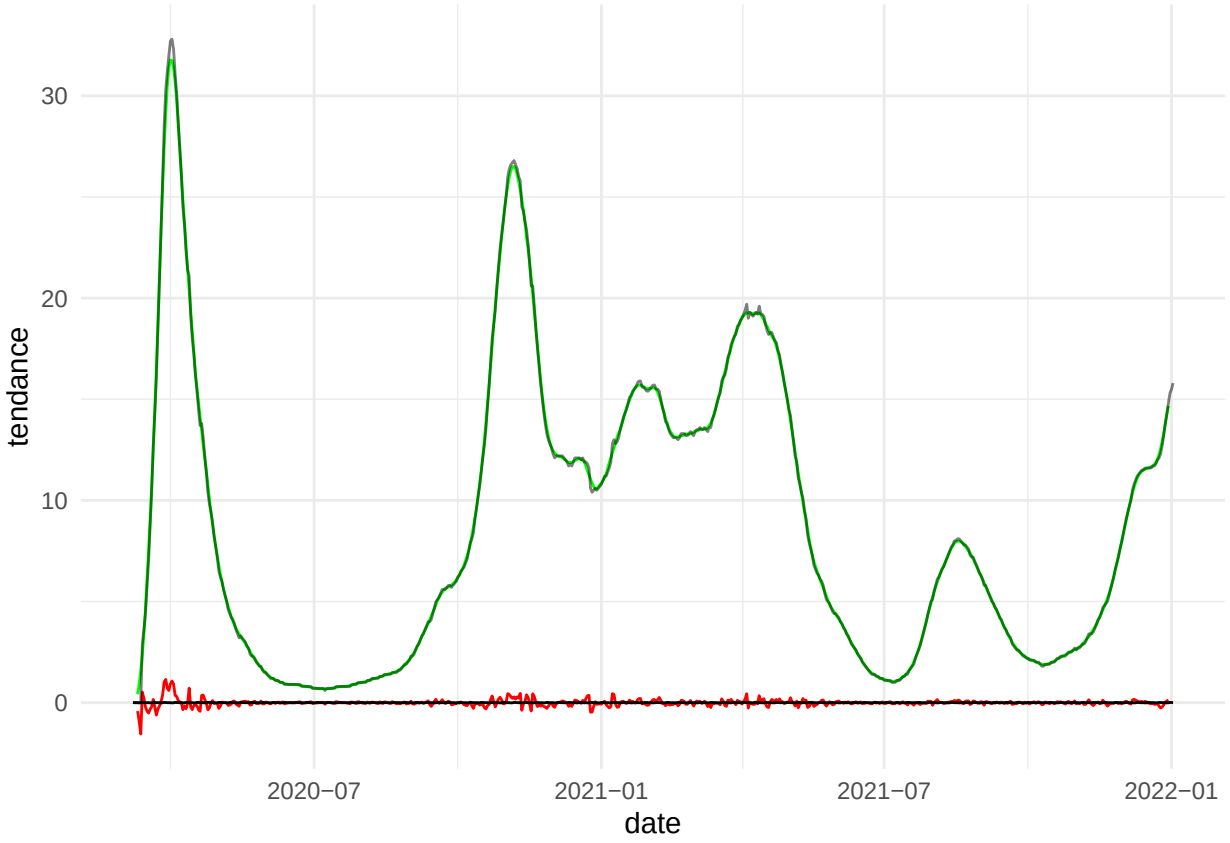


3 Visualiser une série temporelle avec ggplot

Utilisez `ggplot`pour visualisez ces composantes sur un même graphique

```r
baseH$tendance <- as.numeric(mydec$trend)
baseH$residu <- as.numeric(mydec$random)
baseH$saison <- as.numeric(mydec$seasonal)

ggplot(data = baseH, aes(x = date)) +
  geom_line(aes(y = tendance), color = "green") +
  geom_line(aes(y = residu), color = "red") +
  geom_line(aes(y = saison)) +
  geom_line(aes(y = valeur), alpha = 0.5)
```

```
## Warning: Removed 6 row(s) containing missing values (geom_path).
## Removed 6 row(s) containing missing values (geom_path).
```



Que concluez-vous ?

solution

```r
# les variations hebdomadaires sont négligeables par rapport à la tendance.
# Les variations "aléatoires" ne semblent pas l'être totalement et semblent
# être plus importante lors des phases d'incidence forte
#     -> processus multiplicatifs plutôt qu'additif.
```

3.1 Changer l'agrégation temporelle

Pour changer l'agrégation temporelle de jour à semaine ou mois, il suffit de créer une nouvelle variable correspondant à cette unité de temps puis de sommer les valeurs quotidiennes pour chacune des valeurs de cette variable.

Faites une agrégation par semaine et par mois.

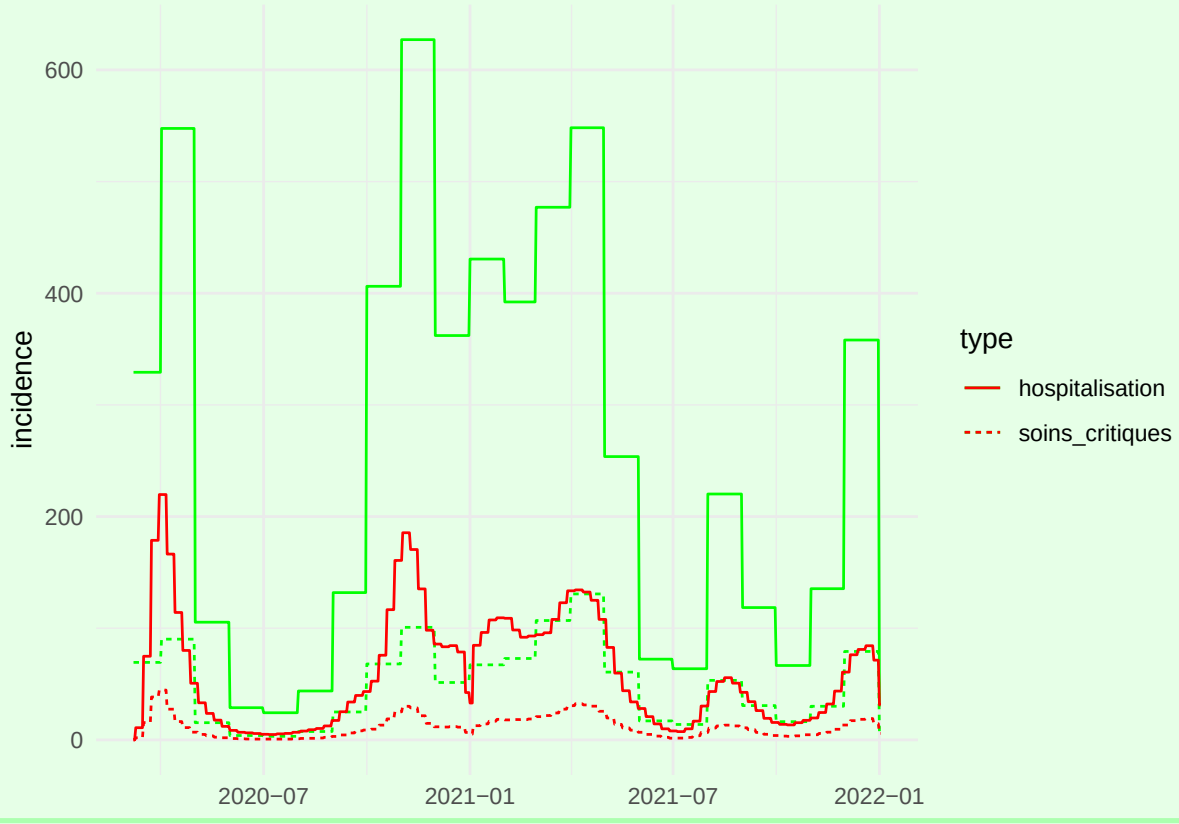
solution

```
baseLong$mois <- format(baseLong$date, "%Y-%B")
baseLong$semaine <- format(baseLong$date, "%Y-%V")

baseLongAgg <- baseLong %>% group_by(mois, type) %>% mutate(valeur_mois = sum(valeur))

baseLongAgg <- baseLongAgg %>% group_by(semaine, type) %>% mutate(valeur_semaine = sum(valeur))
```

Visualisez graphiquement ces nouvelles valeurs.

solution

```
ggplot(baseLongAgg, aes(x = date, linetype = type)) +
  geom_line(aes(y = valeur_mois), color = "green") +
  geom_line(aes(y = valeur_semaine), color = "red") +
  scale_x_date(name = NULL) +
  scale_y_continuous(name = "incidence")
```



3.2 Faire une moyenne glissante

Pour réaliser une moyenne glissante, vous pouvez utiliser des fonctions disponibles dans la librairie `zoo`, telle que `rollmean()`. L'argument k de cette fonction fixe la taille de la fenêtre.

Utilisez `rollmean` pour calculer des moyennes glissantes avec des fenêtre de 7 jours ou de 30 jours. Visualisez les résultats et comparez aux résultats précédents.

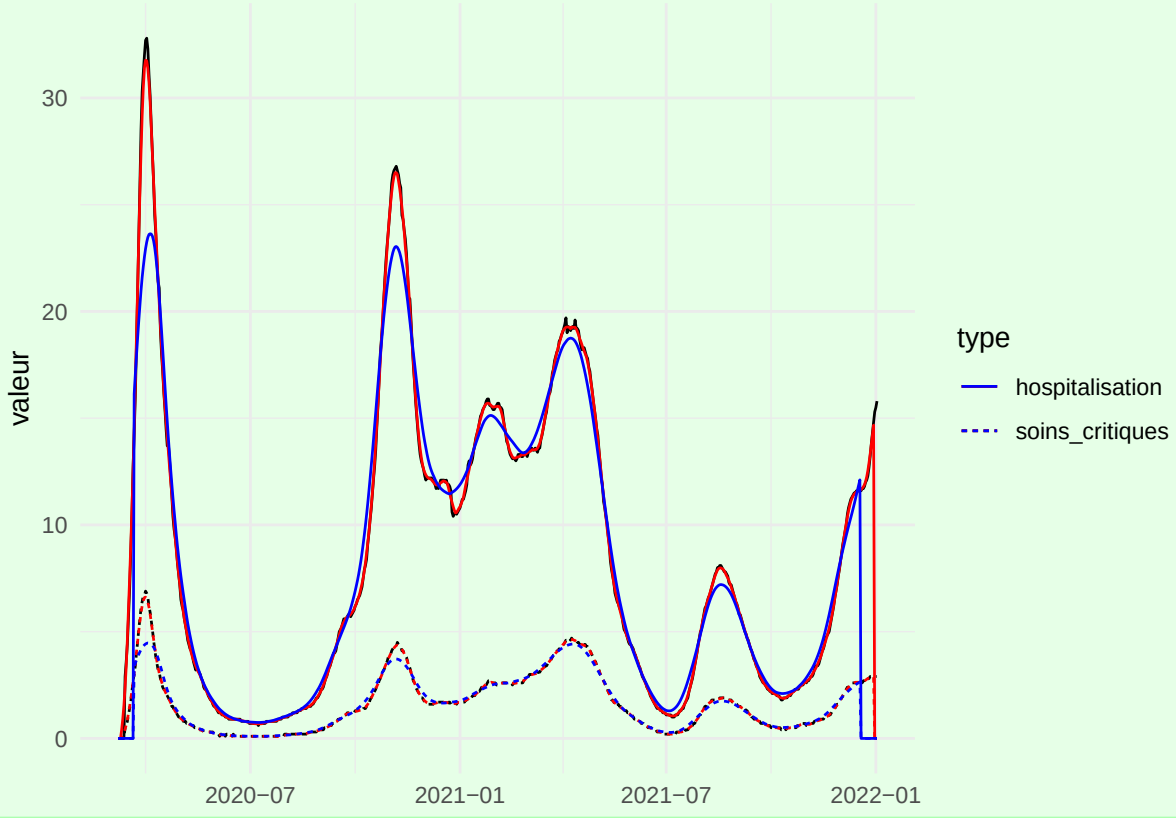
solution

```
library(zoo)
```

```
##
## Attachement du package : 'zoo'
```

```
## Les objets suivants sont masqués depuis 'package:base':
##
##     as.Date, as.Date.numeric
```

```
baseLongRmean <- baseLong %>% group_by(type) %>%
  mutate(valeurM7 = rollmean(x = valeur, k = 7, fill = 0))

baseLongRmean <- baseLongRmean %>% group_by(type) %>%
  mutate(valeurM30 = rollmean(x = valeur, k = 30, fill = 0))

ggplot(baseLongRmean, aes(x = date, linetype = type)) +
  geom_line(aes(y = valeur), color = "black")+
  geom_line(aes(y = valeurM7), color = "red")+
  geom_line(aes(y = valeurM30), color = "blue")+
  scale_x_date(name = NULL)
```



4 Réaliser des opérations et les présenter sous forme de tableau

4.1 Réaliser une opération sur un sous-ensemble de données

La base baseAllHosp contient les données d'incidence journalière d'hospitalisation, de réanimation, de décès et de retour à domicile.

Faites la somme par mois du nombre de nouvelles hospitalisations sur Paris. Pensez à créer une variable *mois*, et utilisez la fonction `aggregate()` pour calculer cette somme.

solution

```
URL <- "https://www.data.gouv.fr/fr/datasets/r/6fadff46-9efd-4c53-942a-54aca783c30c"

# data provenant de data.gouv.fr
baseAllHosp <- read.csv(URL, sep = ";")

# création de la variable date
baseAllHosp$date <- as.Date(baseAllHosp$jour)

# transformation des données
baseAllHosp$mois <- format(baseAllHosp$date, "%Y-%m")

sumByMonthParis <- aggregate(baseAllHosp$incid_hosp[baseAllHosp$dep == "75"],
          by = list(mois = baseAllHosp$mois[baseAllHosp$dep == "75"]),
          FUN = sum)
```

Faites la même chose pour l'Ile-de-France (departements: 75, 77, 78, 91, 92, 93, 94, 95) en utilisant l'opérateur %in%

solution

```
sumByMonthIDF = aggregate(baseAllHosp$incid_hosp[baseAllHosp$dep %in% c("75", "77", "78", "91", "92"
          by = list(mois = baseAllHosp$mois[baseAllHosp$dep %in% c("75", "77", "78", "91", "92", "93
          FUN = sum)
```

Présentez les deux séries dans le même tableau. Vous pouvez rendre vos tableaux plus présentable en installant le package flextable et en utilisant la fonction du même nom dans ce package.

Rendre le tableau présentable avec flextable.

```
dplyr::inner_join(sumByMonthParis, sumByMonthIDF, by = "mois") %>%
  flextable() %>%
  delete_part(part = "header") %>%
  add_header( mois = "Mois", x.x="Paris", x.y="IdF") %>%
  theme_booktabs()
```

Mois	Paris	IdF
2020-03	3,075	12,035
2020-04	5,031	24,092
2020-05	802	4,289
2020-06	163	1,058

Mois	Paris	IdF
2020-07	175	850
2020-08	258	1,278
2020-09	843	4,107
2020-10	2,176	10,196
2020-11	2,099	11,286
2020-12	1,169	6,319
2021-01	1,305	6,892
2021-02	1,665	8,295
2021-03	3,073	14,152
2021-04	3,468	15,289
2021-05	1,448	6,502
2021-06	352	1,760
2021-07	335	1,436
2021-08	638	3,456
2021-09	486	2,052
2021-10	322	1,300
2021-11	490	2,496
2021-12	1,703	7,498
2022-01	4,207	17,180
2022-02	1,763	7,324
2022-03	417	1,591

5 Cartographier des données

Les outils de cartographie sont de plus en plus utilisés en Science des données.

Dans le logiciel *R* il existe de nombreux modules pour la représentation cartographique comme *ggmap*, *maps* ou *mapsf*. Pour une vue d'ensemble des lodumes disponibles sur CRAN vous pouve consulter la *Task View* associée (https://CRAN.R-project.org/view=Spatial).

Dans cet atelier nous vous proposons un exemple d'utilisation de *mapsf*.

Dans un premier temps, nous allons aggréger les incidences d'hospitalisation avec l'instruction suivante.

```
sumByMonthDept <- aggregate(baseAllHosp$incid_hosp,
          by = list(mois = baseAllHosp$mois, code_insee = baseAllHosp$dep),
          FUN = sum)
names(sumByMonthDept)[3] <- "incidence"
```

Quel est le niveau d'aggrégation ?

solution

```
# Le niveau d'agrégation est le mois et le département.
```

Ensuite, nous allons avoir besoin de lire des fichiers *Shape* qui définissent à l'aide de polygones les frontrières géographiques. Pour se faire vous devrez télécharger et charger le package *sf*. Puis, pour la représentation cartographique nous allons utliser le package *mapsf*.

```
# Charger les libraries utiles
library(sf)
```

```
## Linking to GEOS 3.10.2, GDAL 3.4.1, PROJ 8.2.1; sf_use_s2() is TRUE
```

```
library(mapsf)
```

Nous allons lire les polygones géographiques des départements français. Pour ce faire vous devez télécharger les données sources soit sur le site de IGN ou le dossier departements-2017 (version simplifiée) sur https://github.com/nolwenn/DataScience.

- Sauvegarder le dossier dans votre dossier de projet R.
- Utiliser les instructions ci-dessous pour lire le fichier shape
- Que font les différentes instructions?

```
# Lire le fichier shape pour la France
FrMap <- st_read(dsn="departements-2017/.", quiet = TRUE)
FrMap$code_insee[FrMap$code_insee=="69D"] <- "69"

# Jointure avec les taux d'incidence
FrMap_incidence <- full_join( FrMap, sumByMonthDept, by="code_insee" )
```

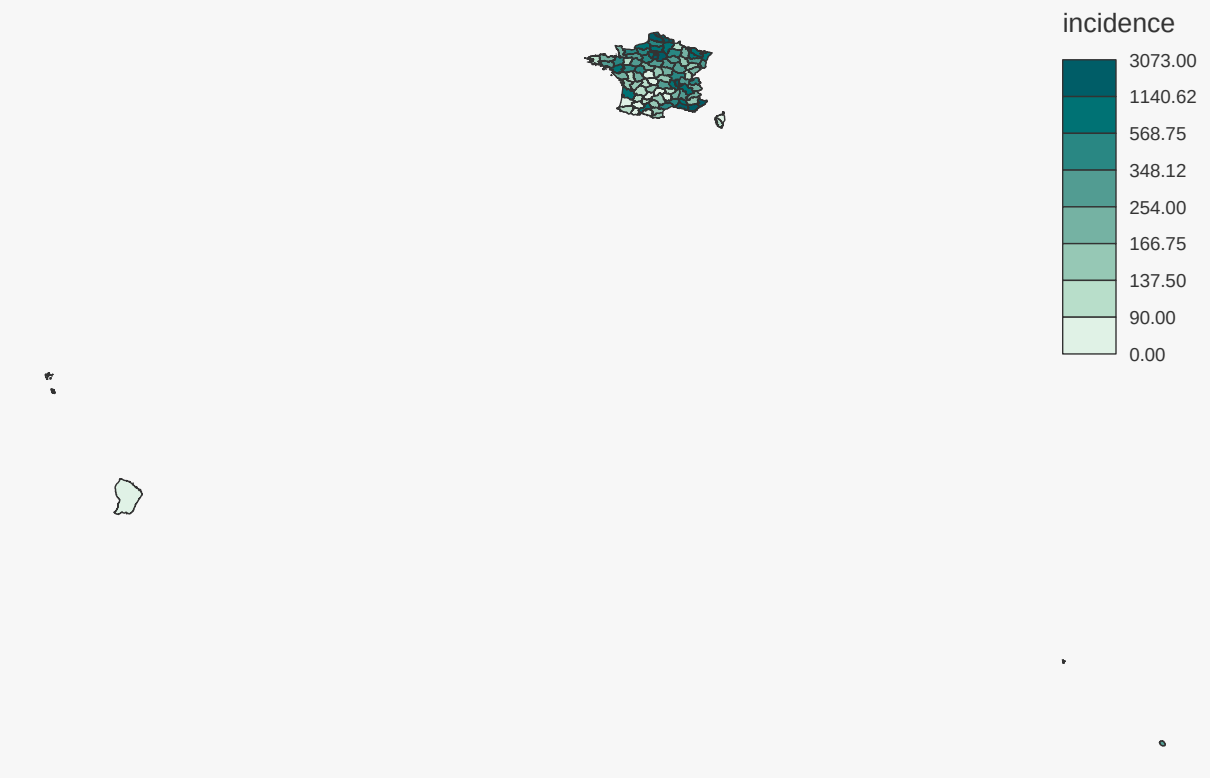
Nous allons nous focaliser sur le mois de mars 2021.

- Utiliser l'instruction ci-dessous pour filtrer les données.

```
incidence_mars2021 <- FrMap_incidence %>%
  dplyr::filter(mois == "2021-03")
```

- Utiliser la fonction *mf_map()* de la *library mapsf* pour faire une crate chorapleth de l'incidence de mars 2021. [Pensez à regarder l'aide de la fonction].

```
mf_map(x = incidence_mars2021, var = "incidence", type = "choro")
```



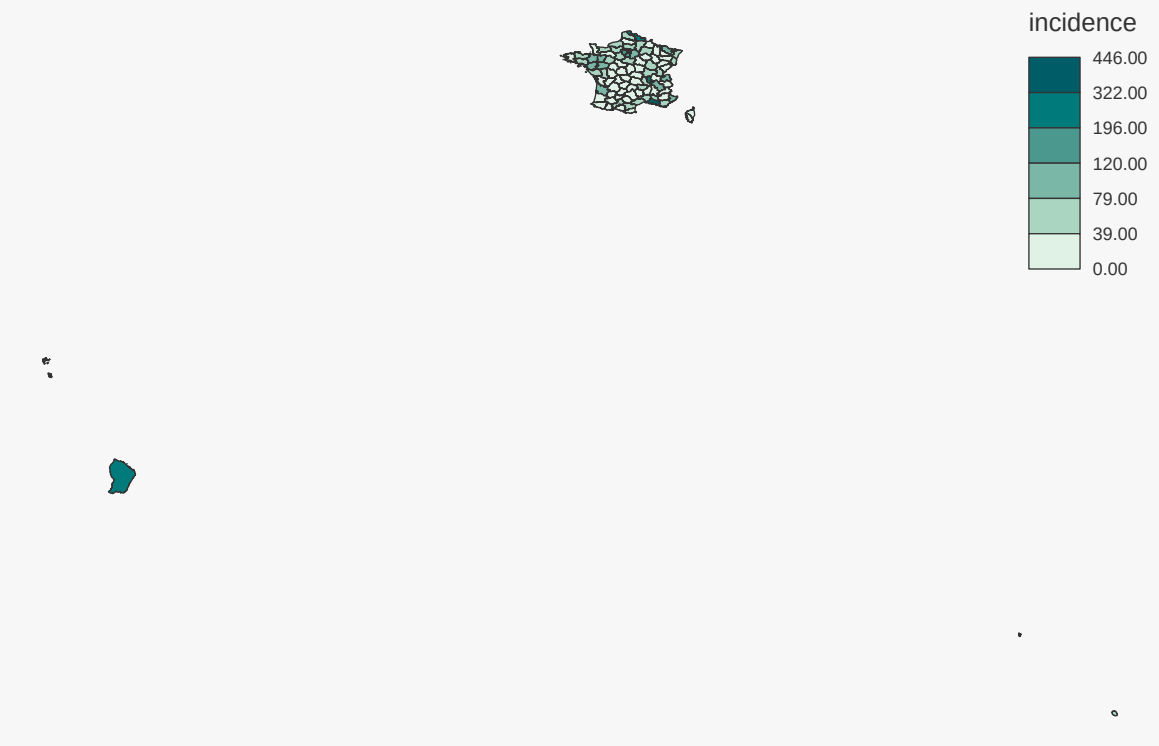
- Refaire la même chose pour le mois d'octobre 2021

solution

```
incidence_oct2021 <- FrMap_incidence %>%
  dplyr::filter(mois == "2021-10")
```

solution

```
mf_map(x = incidence_oct2021,
       var = "incidence",
       type = "choro",
       nbreaks=6,
       breaks = "jenks")
```



Vous pouvez remarquer que la carte n'est pas lisible puisque n'est pas centrer sur la France. Pour un focus sur la métropole, nous allons mettre à part les départements ultra-marins.

- Suivez les instructions ci-dessous pour se faire.

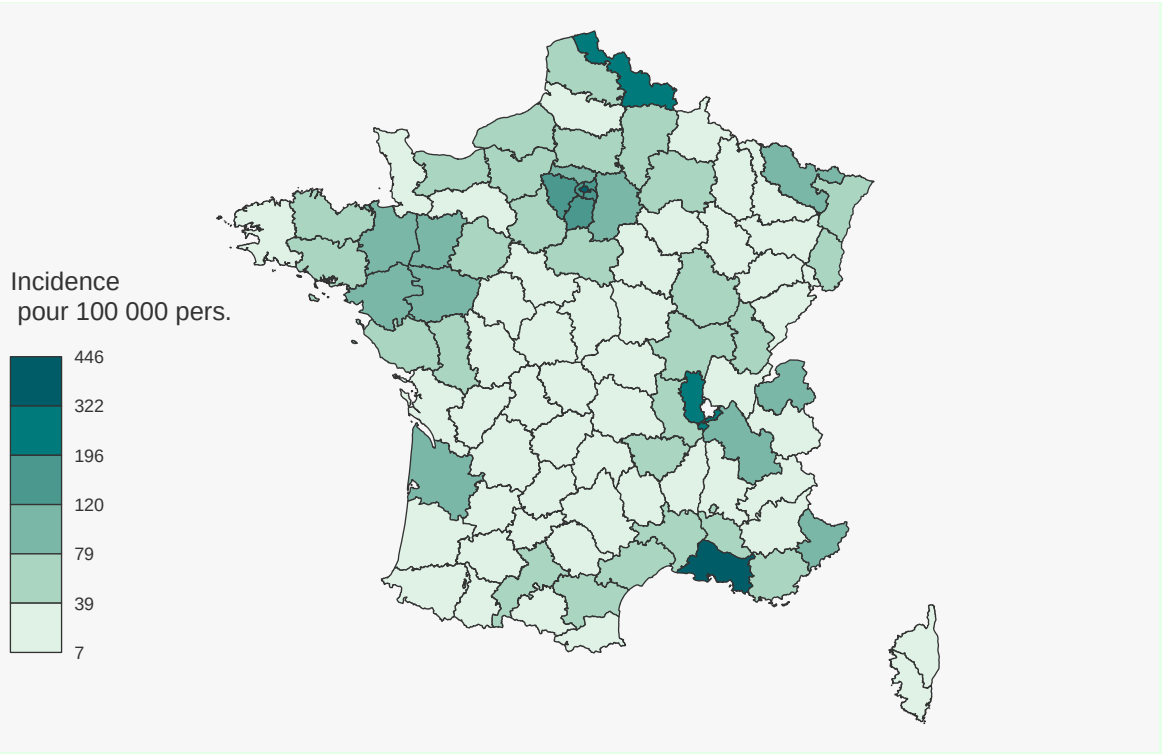
Présentez la carte en retirant les départements ultra-marins.

```
ultra_marin <- grep("^97", incidence_oct2021$code_insee)
incidence_oct2021_metro <- incidence_oct2021[- ultra_marin, ]
```

- Refaire la représentation cartographique
- Sauvegarder la carte

solution

```
mf_map(x = incidence_oct2021_metro,
       var = "incidence",
       type = "choro",
       nbreaks=6,
       breaks = "jenks",
       leg_val_rnd= 0,
       col = "brown4",
       leg_pos = "bottomleft2",
       leg_title = "Incidence \n pour 100 000 pers.")
```


Incidence
pour 100 000 pers.
446
322
196
120
79
39
7