

# Créer une carte avec le package RASTER

Thomas MASSÉ

10/11/2020

## INTRODUCTION

Lorsque l'on fait des analyses avec comme base de comparaisons des entités géographiques (pays, régions etc...) la représentation des résultats sous forme de carte est plus pertinente.

Pour se besoin nous allons donc voir ici comment créer une carte avec R avec le package **raster**.

Plusieurs packages existent pour créer des cartes (ggmap, ggplot2, tmap, etc...)

2 types de cartes différentes

- **Vectorielle** (composé d'éléments géographique)
- **Raster** (images)

## LE PACKAGE RASTER

**INSTALLATION DES PACKAGES** La première étape consiste à installer le package, il faut également définir un répertoire de travail, dans lequel les jeux de données seront téléchargés.

```
install.packages("raster", repos = "http://cran.us.r-project.org")
install.packages("rgdal", repos = "http://cran.us.r-project.org")
install.packages("RCurl", repos = "http://cran.us.r-project.org")
#Définition du répertoire de travail
#setwd("C:/R/Maps")
```

```
library(raster)
library(rgdal)
library(RCurl)
```

**LA FONCTION GETDATA** Ensuite nous allons charger une carte, pour cela il est nécessaire de comprendre la fonction **getData**. Comme expliqué dans la documentation R du package (Hijmans (n.d.))

La fonction **getData** permet d'obtenir des données géographiques de n'importe où dans le monde. Elle possède 3 arguments principaux :

- *name* : c'est le nom du jeu de données
- *download* : si on télécharge le jeu de données ou non
- *path* : le chemin dans lequel stocker les données
- ainsi que des arguments additionnels selon le jeu de données choisi.

L'argument “name” peut prendre les valeurs suivantes :

- *alt*
- *GADM*

- *worldclim*
- *SRTM*

Si le nom “**alt**” ou “**GADM**” est utilisé il faut également donner un argument “country”.

Nous allons utiliser d'abord **GADM** pour générer les contour du pays, à noter que l'argument country utilise le code ISO (3 lettres) du pays comme expliquer sur le site gis-blog Martin (n.d.) Nous pouvons consulter la liste des codes avec la fonction **ccodes()**

```
ccodes()
```

```
if (url.exists("https://biogeo.ucdavis.edu/data/gadm3.6/Rsp/gadm36_GBR_0_sp.rds")){
  maCarte <- getData(name="GADM", country="GBR", level=0)
  plot(maCarte, main="Carte de la Grande Bretagne")
}else{
  print("L'adresse est momentanément indisponible, la carte ne sera pas affichée. Veuillez réessayer ultérieurement")
}
```

## CREATION DE LA CARTE DES CONTOURS

### Carte de la Grande Bretagne



L'argument **level** permet d'affiner l'affichage des démarcations géographiques.

Dans notre cas ici augmenter de 1 le niveau nous permettra de voir les frontières entre Angleterre, Pays de Galles, Ecosse et Irlande.

Tandis qu'un level à 2 nous permettra de voir les provinces (ex: Aberdeenshire)

```
maCarte <- getData(name="GADM", country="GBR", level=1)
plot(maCarte, main="Carte de la Grande Bretagne")
```

## Carte de la Grande Bretagne



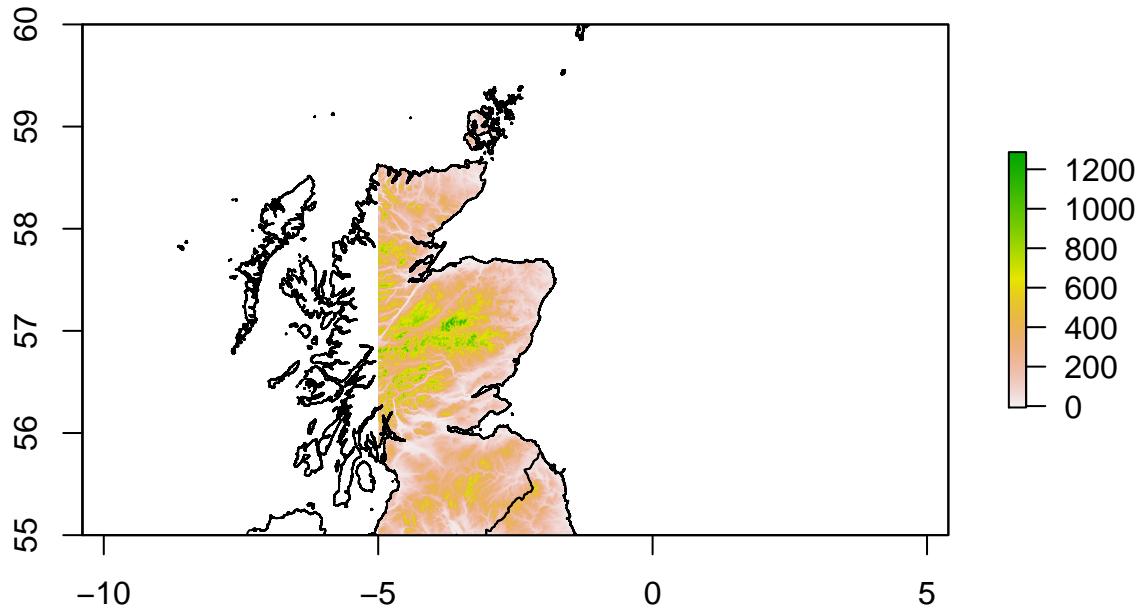
**CREATION DE LA CARTE TOPOGRAPHIQUE** Nous allons désormais nous intéresser à **SRTM** pour ajouter des données topographiques à notre carte.

Le jeu de données **SRTM** utilise les longitudes et latitude pour identifier les dalles.

Cette information est facilement trouvable sur internet (wikipedia par exemple) On va cibler une zone plus petite, l'Ecosse.

```
maCarteSRTM <- getData("SRTM", lon=-2, lat=57)
plot(maCarteSRTM, main="Ma Carte de l'Ecosse")
plot(maCarte, add=TRUE)
```

## Ma Carte de l'Ecosse



Si la dalle affiché n'est pas assez grande on peut en ajouter.

```
maCarteSRTM2 <- getData("SRTM", lon=-8, lat=57)
maCarteSRTM3 <- getData("SRTM", lon=-2, lat=53)
maCarteSRTM4 <- getData("SRTM", lon=-8, lat=53)
```

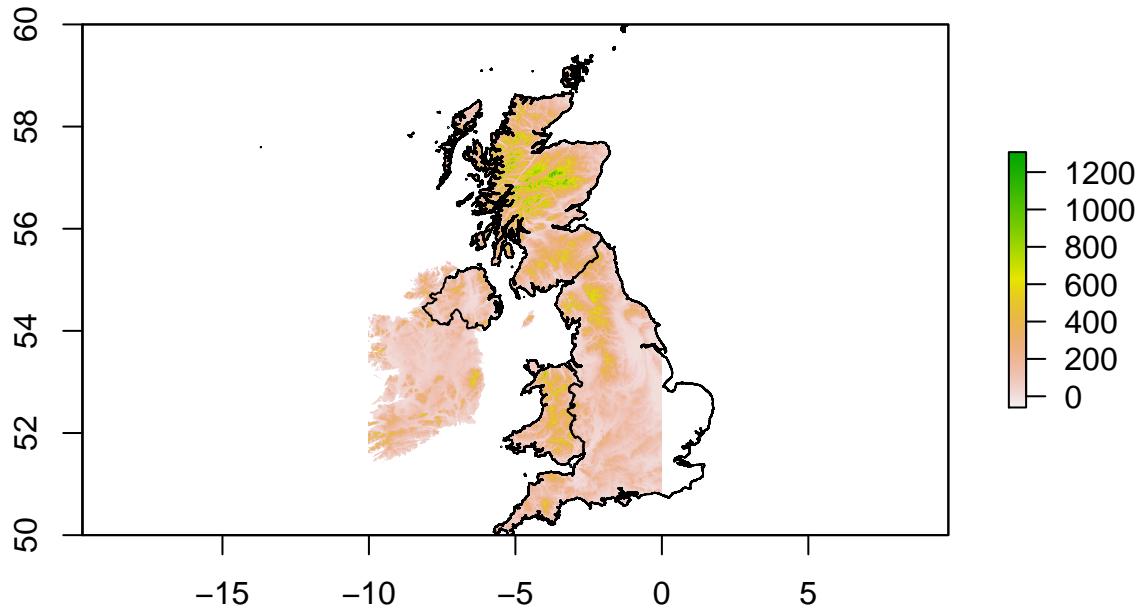
Ensuite on assemble les dalles ensemble pour faire une carte. On va utiliser ici la fonction **mosaic**.

Dans la fonction mosaic les arguments sont les différentes cartes et un argument **fun**.

Cet argument est une **function** à appliquer pour les parties superposées, il peut prendre la valeur *min*, *max*, ou *mean*.

```
maCarteMerge <- mosaic(maCarteSRTM, maCarteSRTM2, maCarteSRTM3, maCarteSRTM4, fun=mean)
plot(maCarteMerge, main="Ma Carte Merge")
plot(maCarte, add=TRUE)
```

## Ma Carte Merge



A noter qu'il existe d'autres méthodes d'aggrégation que l'on peut consulter ici <https://cran.r-project.org/web/packages/raster/raster.pdf> à la page 8.

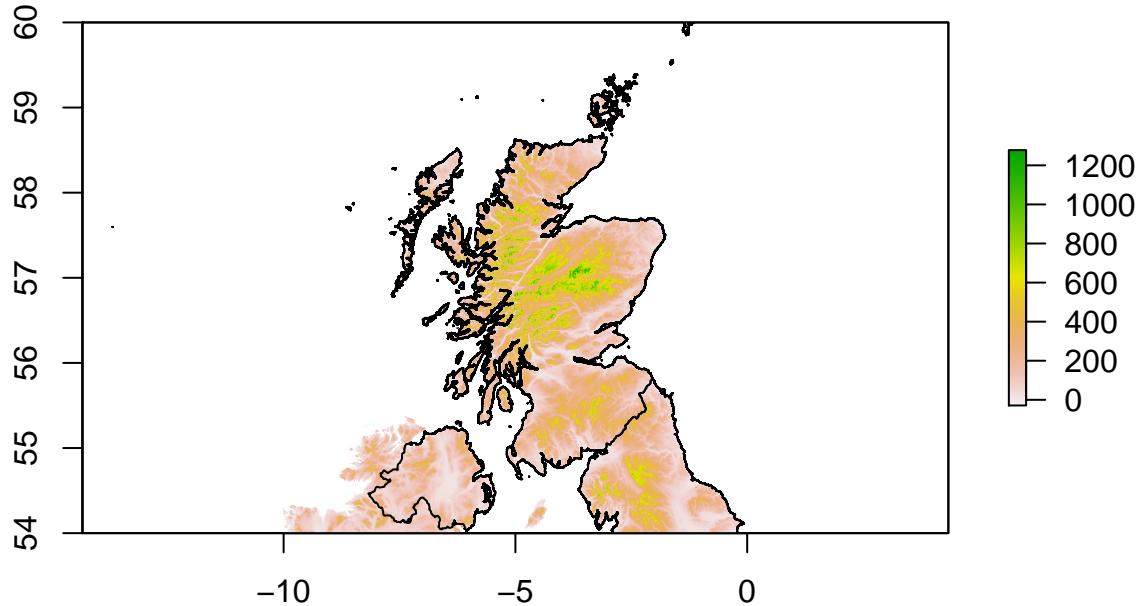
Il y a par exemple la fonction **merge**, **trim**, ou **aggregate**.

Mais également la fonction **crop** que l'on va utiliser ici car on constate que la zone est trop large, on va donc utiliser la fonction **crop** pour redéfinir la zone.

Dans l'argument *extent* nous allons spécifier dans cet ordre, latitude minimale, latitude maximale, longitude minimale et longitude maximale.

```
maCarteCrop <- crop(maCarteMerge, extent(-10, 0, 54, 60))
plot(maCarteCrop, main="Ma Carte Cropped")
plot(maCarte, add=TRUE)
```

## Ma Carte Cropped



**AJOUTS DE POI (POINTS D'INTERETS)** Pour finir nous souhaitons ajouter des points d'intérêt, par exemple des villes.

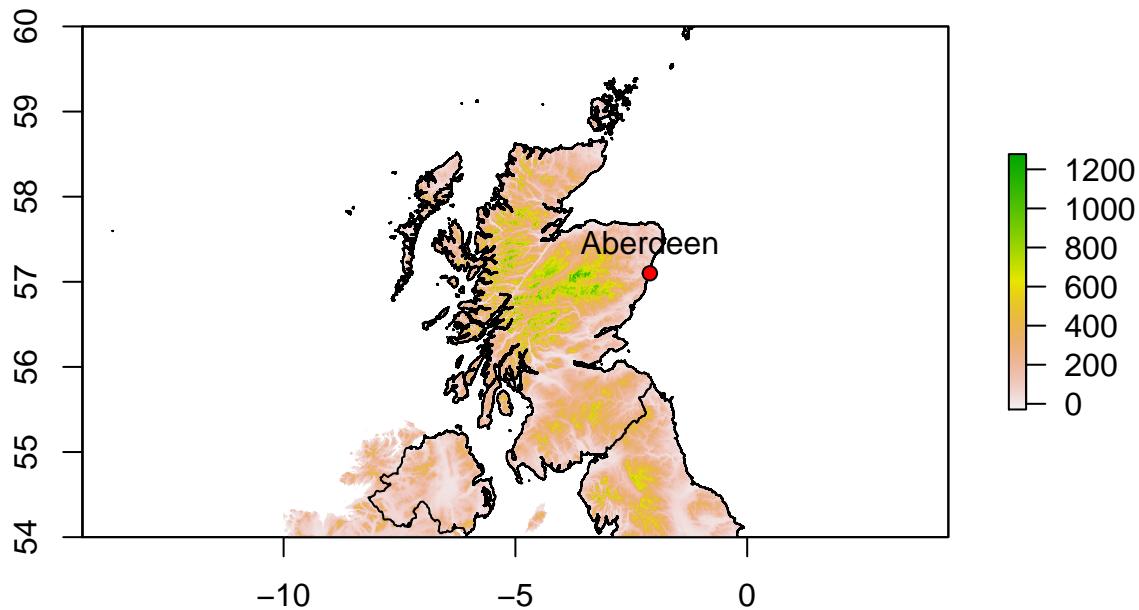
Le fonctionnement est globalement le même que pour les données topographiques (**SRTM**), ce qui signifie que nous aurons besoin des coordonnées des villes.

Pour ajouter les points on utilise la fonction `point` de R, les 2 premiers arguments dans l'exemple ci-dessous sont les coordonnées, `pch` c'est la forme du point ("Les Différents Styles de Points Dans R," n.d.), `col` la couleur du contour et `bg` la couleur de remplissage.

Pour ajouter les textes, on utilise la fonction `text` de R, ici on utilise les mêmes coordonnées pour la longitude et latitude car on souhaite que le texte soit au dessus du point créé, `cex` c'est un facteur entre 0 et 1 de la taille finale des caractères, `pos` c'est ce qui nous permet de décaler le texte au dessus. Pour en savoir plus sur la fonction on peut consulter ("Comment Ajouter Du Texte Sur Un Graphique R," n.d.).

```
plot(maCarteCrop, main="Ma Carte Cropped")
plot(maCarte, add=TRUE)
points(-2.1, 57.1, pch=21, col="black", bg="red")
text(-2.1, 57.1, labels="Aberdeen", cex=1, pos=3)
```

## Ma Carte Cropped



A noter que l'on peut également ajouter plusieurs points en même temps à l'aide de **vecteurs**.

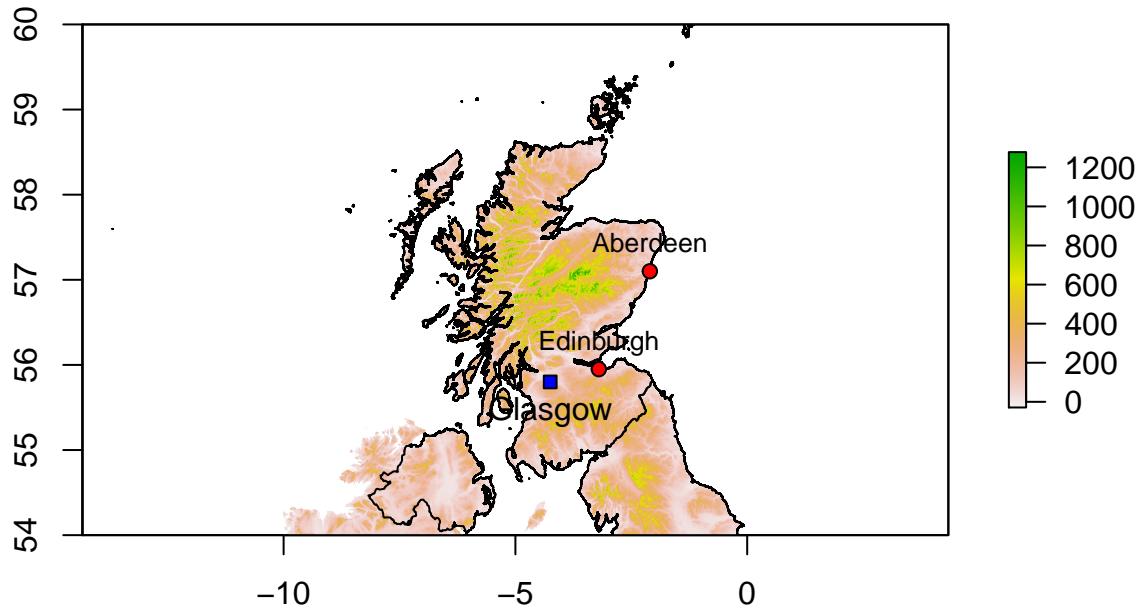
```
plot(maCarteCrop, main="Ma Carte Cropped")
plot(maCarte, add=TRUE)

p.lon <- c(-2.1, -3.2)
p.lat <- c(57.1, 55.95)
p.lbl <- c("Aberdeen", "Edinburgh")

points(p.lon, p.lat, pch=21, col="black", bg="red")
text(p.lon, p.lat, labels=p.lbl, cex=.8, pos=3)

points(-4.25, 55.8, pch=22, col="black", bg="blue")
text(-4.25, 55.8, labels="Glasgow", cex=1, pos=1)
```

## Ma Carte Cropped



## REFERENCES

- “Comment Ajouter Du Texte Sur Un Graphique R.” n.d. <http://www.sthda.com/french/wiki/ajouter-du-texte-a-un-graphique-avec-le-logiciel-r>.
- Hijmans, Robert. n.d. “RDocumentation Du Package Raster.” <https://www.rdocumentation.org/packages/raster/versions/3.3-13/topics/getData>.
- “Les Différents Styles de Points Dans R.” n.d. <http://www.sthda.com/french/wiki/les-differents-types-de-points-dans-r-comment-utiliser-pch>.
- Martin. n.d. “R Raster: Data Acquisition – Srtm, Worldclim, Global Adm. Boundaries.” <https://www.gis-blog.com/r-raster-data-acquisition/>.