Fiche TD avec le logiciel R : histoMeteo

Récupération de l'historique des données météo quotidiennes d'un site en France métropolitaine

Pr Jean R. Lobry

Depuis début 2024, le site de Météo-France distribue des fichiers de données climatiques quotidiennes en France métropolitaine avec une résolution spatiale de 8 × 8 km et une plage temporelle allant du 1er août 1958 à nos jours. On montre ici comment extraire les données d'un site particulier à partir de ses coordonnées GPS.

Table des matières

1 Source

Les données SAFRAN [5] sont disponibles sur le site de Météo-France[1] sous la forme de fichiers par décennie au format `cvs` compressé. Le fichier d'une décennie complète pèse de l'ordre de 4 Gio après décompression, ce qui le rend difficilement importable directement avec un logiciel de type tableur, et nécessite quelques précautions pour l'importer sous R tournant sur un ordinateur personnel[2].

2 Conversion en format binaire

La première étape va consister à convertir les fichiers au format `csv` au format binaire `XDR` [4]. Les avantages du format binaire sont d'une part qu'il est compatible multi-plateformes (on peut l'importer dans R directement sous Linux, MacOS et windows) et d'autre part qu'il est beaucoup plus proche de la représentation en mémoire vive des différents types de données (entiers, flottants, etc.) ce qui accélère considérablement les opérations d'entrées-sorties. Pour donner un exemple concret, l'importation du fichier `csv` d'une décennie complète sur mon ordinateur avec une instruction du type :

```
dta <- read.table("../SIM2/data/csv/QUOT_SIM2_1960-1969.csv", header = TRUE, sep = ";")
```

prend de l'ordre de 15 *minutes*, alors que l'importation du fichier binaire correspondant prend de l'ordre de 15 ... *secondes*. Le script que j'ai utilisé est donné ci-après, la seule modification apportée aux données consiste à convertir la colonne `DATE` dans la classe `Date` qui permet de manipuler facilement sous R des données calendaires[3].

```
mkrda <- function(prefix, pathcsv = "data/csv/", pathrda = "data/rda/" ){
  fname <- paste0(pathcsv, prefix, ".csv")
  dta <- read.table(fname, header = TRUE, sep = ";")
  # Seule modif apportée
  dta$DATE <- as.Date(as.character(dta$DATE), format = "%Y%m%d")
  save(dta, file = paste0(pathrda, prefix, ".Rda"))
}
# Exemple d'utilisation
mkrda("QUOT_SIM2_1960-1969")
```

Les fichiers binaires de chaque décennie ont été sauvegardées sur le serveur du pôle de bio-informatique lyonnais, on peut y accéder directement de la façon suivante :

```
chmin <- "http://pbil.univ-lyon1.fr/R/donnees/histoMeteo/"
load(url(paste0(chmin, "QUOT_SIM2_1960-1969.Rda")))
```

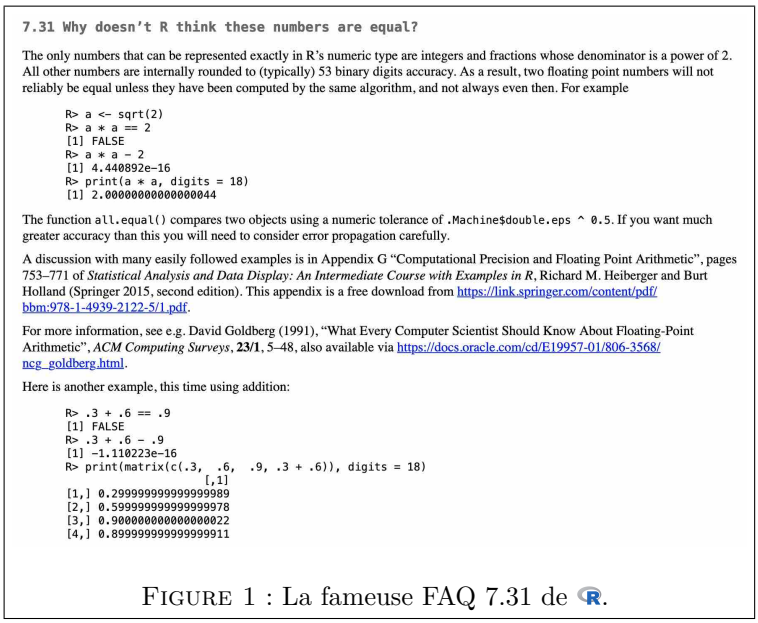
Le temps de chargement de la table `dta` va dépendre de la bande passante du réseau, de chez moi[4] une décennie complète met environ 2 minutes pour charger, aussi je recommande de faire une copie locale de ces fichiers si vous souhaitez y accéder plusieurs fois. Le chargement de la table se fait alors avec une instruction du type :

[1] `https://meteo.data.gouv.fr/datasets/6569b27598256cc583c917a7`

[2] J'ai utilisé ici un ordinateur portable MacBook Pro avec 16 Gio de RAM.

[3] Pour la manipulation de ce type de données on pourra se référer à la fiche « [m]anipulation de données calendaires appliquée au suivi hebdomadaire de la croissance de dix chênes pendant sept ans » à `http://pbil.univ-lyon1.fr/R/pdf/dendroCHS57.pdf`

[4] J'ai une connexion ADSL accédée en WiFi avec un débit descendant de l'ordre 200 Mb/s



7.31 Why doesn't R think these numbers are equal?

The only numbers that can be represented exactly in R's numeric type are integers and fractions whose denominator is a power of 2. All other numbers are internally rounded to (typically) 53 binary digits accuracy. As a result, two floating point numbers will not reliably be equal unless they have been computed by the same algorithm, and not always even then. For example

```
R> a <- sqrt(2)
R> a * a == 2
[1] FALSE
R> a * a - 2
[1] 4.440892e-16
R> print(a * a, digits = 18)
[1] 2.0000000000000044
```

The function `all.equal()` compares two objects using a numeric tolerance of `.Machine$double.eps ^ 0.5`. If you want much greater accuracy than this you will need to consider error propagation carefully.

A discussion with many easily followed examples is in Appendix G "Computational Precision and Floating Point Arithmetic", pages 753–771 of *Statistical Analysis and Data Display: An Intermediate Course with Examples in R*, Richard M. Heiberger and Burt Holland (Springer 2015, second edition). This appendix is a free download from https://link.springer.com/content/pdf/bbm:978-1-4939-2122-5/1.pdf.

For more information, see e.g. David Goldberg (1991), "What Every Computer Scientist Should Know About Floating-Point Arithmetic", *ACM Computing Surveys*, **23/1**, 5–48, also available via https://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html.

Here is another example, this time using addition:

```
R> .3 + .6 == .9
[1] FALSE
R> .3 + .6 - .9
[1] -1.110223e-16
R> print(matrix(c(.3,  .6,  .9, .3 + .6)), digits = 18)
                    [,1]
[1,] 0.299999999999999989
[2,] 0.599999999999999978
[3,] 0.900000000000000022
[4,] 0.899999999999999911
```

FIGURE 1 : La fameuse FAQ 7.31 de R.

```
load("../SIM2/data/rda/QUOT_SIM2_1960-1969.Rda")
colnames(dta)
```

```
 [1] "LAMBX"          "LAMBY"          "DATE"           "PRENEI_Q"       "PRELIQ_Q"
 [6] "T_Q"            "FF_Q"           "Q_Q"            "DLI_Q"          "SSI_Q"
[11] "HU_Q"           "EVAP_Q"         "ETP_Q"          "PE_Q"           "SWI_Q"
[16] "DRAINC_Q"       "RUNC_Q"         "RESR_NEIGE_Q"   "RESR_NEIGE6_Q"  "HTEURNEIGE_Q"
[21] "HTEURNEIGE6_Q"  "HTEURNEIGEX_Q"  "SNOW_FRAC_Q"    "ECOULEMENT_Q"   "WG_RACINE_Q"
[26] "WGI_RACINE_Q"   "TINF_H_Q"       "TSUP_H_Q"
```

```
nrow(dta)
```

```
[1] 36135476
```

La description et les unités des variables sont disponibles sur le site de Météo-France[5]. Toutes les variables sont de type `numeric` sauf les coordonnées (`LAMBX` et `LAMBY`) du centre des mailles SAFRAN en Lambert II étendu[6] qui sont de type `integer`, ce qui nous permettra de faire des tests d'égalité sans tomber sous les fourches caudines de la fameuse FAQ 7.31 (figure 1 page 3), et la variable `DATE` que nous avons convertie dans la classe `Date` lors de la transformation au format `XDR` [4]. La table `dta` compte plus de 36 millions de lignes, avec une résolution de 8 × 8 km il faut 9892 mailles pour couvrir la France métropolitaine[7], et donc pour une décennie de valeurs quotidiennes on retrouve le nombre de lignes de notre table :

```
njours <- length(seq.Date(as.Date("1960-01-01"), as.Date("1969-12-31"), by  = "day"))
njours*9892 == nrow(dta)
```

```
[1] TRUE
```

[5] Fichier `liste_parametres.odt` à `https://meteo.data.gouv.fr/datasets/6569b27598256cc583c917a7`

[6] `EPSG:27572`

[7] Les données disponibles débordent un peu de la France métropolitaine, voir la figure à la fin de la section 4.1 page 4.

Unité	Nombre d'octets	Capacité de stockage
octet	2^0	un caractère
Kio	2^{10}	une page
Mio	2^{20}	un livre
Gio	2^{30}	une bibliothèque personnelle
Tio	2^{40}	une bibliothèque universitaire

Table 1 : Les principales unités de mesure de la taille des objets

3 La taille compte

La fonction `file.size()` nous permet d'obtenir la taille en octets occupée par un fichier. Pour faciliter la lecture on travaillera en Gio (voir la table 1 page 4). Un fichier pour une décennie pèse environ 1 Gio, soit à peu près la même chose que les fichiers compressés distribués par Météo-France.

```
file.size("../SIM2/data/rda/QUOT_SIM2_1960-1969.Rda")/2^30
```
```
[1] 0.8870066
```

Mais ce qui nous importe c'est la taille de la table une fois importée en mémoire. C'est la fonction `object.size()` qui va nous renseigner :

```
print(object.size(dta), units = "GiB")
```
```
7.3 GiB
```

La table d'une décennie pèse donc de l'ordre de 7 Gio en mémoire. Sur mon ordinateur avec 16 Gio de mémoire vive, il n'est donc pas raisonnable de vouloir charger plus d'une décennie à la fois pour extraire les informations qui m'intéressent. L'astuce va consister à détruire avec `rm()` la table `dta` dès que l'on en n'a plus besoin et d'invoquer dans la foulée le ramasse-miettes avec `gc()` pour libérer la mémoire (section 4.2 page 6).

4 Sélection d'une maille

Supposons que je ne sache pas encore très bien quelle est la variable climatique qui m'intéresse, mais qu'en revanche je sois déjà fixé sur un site donné. Pour fixer les idées mettons que ce soit le site du capteur RNSA de Bourg-en-Bresse dont je connais les coordonnées GPS[8]. Je veux donc récupérer toutes les données climatiques de la maille SAFRAN qui contient ce site.

4.1 Localisation d'une maille

Je commence par récupérer les coordonnées des mailles documentées en sélectionnant un jour particulier, puis je passe en coordonnées GPS.

```
load("../SIM2/data/rda/QUOT_SIM2_1958-1959.Rda")
mailles <- subset(dta, DATE == as.Date("1959-08-01"))[ , c("LAMBX", "LAMBY")]
ptsLII <- with(mailles, terra::vect(100*cbind(LAMBX, LAMBY), crs = "EPSG:27572"))
ptsGPS <- terra::project(ptsLII, "EPSG:4326")
```

[8] +5.220700 de longitude et +46.21010 de latitude.

Je définis une fonction `getmaille()` qui à partir de coordonnées GPS va trouver la maille SAFRAN la plus proche. Comme on a une résolution de 8 × 8 km, si la distance la plus proche est supérieure à $\sqrt{4^2 + 4^2}$ c'est que nous sommes en dehors de la zone couverte par les mailles SAFRAN, on émet alors un message d'avis.

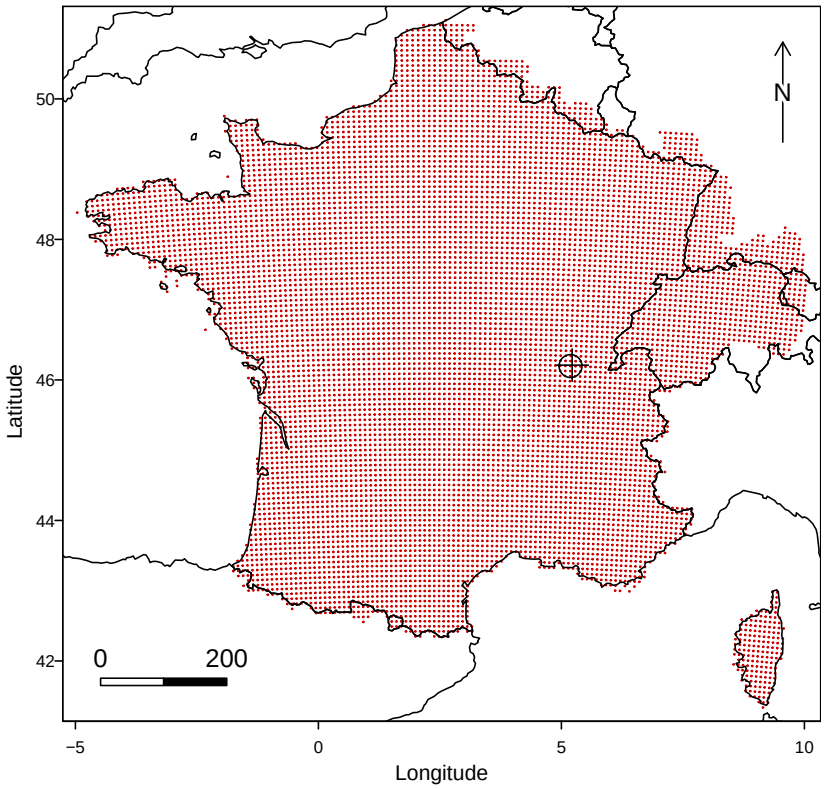
```
getmaille <- function(xlon, ylat){
  target <- terra::vect(cbind(xlon, ylat), crs = "EPSG:4326") # GPS
  mailles$dist <<- terra::distance(ptsGPS, target)[, 1]/1000 # en km
  ii <- which.min(mailles$dist)
  dist <- mailles[ii, "dist"]
  if(dist > sqrt(2*4^2)) warning(paste("Site trop loin : distance =", dist, "km"))
  return(list(LAMBX = mailles[ii, "LAMBX"],
              LAMBY = mailles[ii, "LAMBY"],
              dist = dist))
}
test <- getmaille(0, 0)    # C'est loin de chez nous l'équateur!
test <- getmaille(0, 90)   # Le pôle Nord aussi
test <- getmaille(0, -90)  # Quant au pôle Sud...
getmaille(+5.220700, +46.21010) # Le site RNSA de Bourg-en-Bresse
```

```
$LAMBX
[1] 8200
$LAMBY
[1] 21370

$dist
[1] 2.930094
```

Les coordonnées en Lambert II hectométrique nous donnent une clef d'identification de la maille SAFRAN. Il nous suffit donc de récupérer uniquement les informations climatiques de cette maille. On fait une petite représentation graphique avec les paquets `terra` [2] et `geodata` [3] pour vérifier qu'il n'y a pas d'erreur grossière dans la fonction `getmaille()`.

```
terra::plot(ptsGPS, pch = 19, cex = 0.1, las = 1, col = "red3",
            xlab = "Longitude", ylab = "Latitude")
terra::lines(geodata::world(resolution = 4, path = tempdir()))
terra::sbar(type = "bar")
terra::north()
xlon <- +5.220700 ; ylat <- +46.21010
terra::points(terra::vect(cbind(xlon, ylat), crs = "EPSG:4326"), pch = 3, cex = 2)
mamaille <- getmaille(xlon, ylat)
cible <- with(mamaille, {
  ptLII <- terra::vect(100*cbind(LAMBX, LAMBY), crs = "EPSG:27572")
  ptGPS <- terra::project(ptLII, "EPSG:4326")
  terra::points(ptGPS, pch = 1, cex = 2)
})
```



Tout va bien, la croix n'est pas parfaitement au centre du cercle, mais c'est normal puisque le centre de la maille SAFRAN la plus proche est à 2.9 km du site RNSA de Bourg-en-Bresse. On note au passage que les mailles SAFRAN disponibles débordent un petit peu du territoire de la France métropolitaine.

4.2 Extraction des données d'une maille

Dans le script suivant on charge les données d'une décennie à la fois, on extrait les données de la maille d'intérêt, puis on libère la mémoire. Son exécution prend environ 3 minutes sur mon ordinateur.

```
mamaille <- getmaille(+5.220700, +46.21010)
# préparation de la table vide
load("../SIM2/data/rda/QUOT_SIM2_1958-1959.Rda")
tabSite <- dta[1, ] # Pour récupérer le nom et le type des colonnes
tabSite <- tabSite[-1, ] # Pour vider la table
# Boucle sur les décennies
ficnames <- dir(path = "../SIM2/data/rda", pattern = "QUOT_SIM2", full.names = TRUE)
for(fic in ficnames){
  print(fic)
  load(fic)
  tabSite <- rbind(tabSite, subset(dta,
          LAMBX == mamaille$LAMBX &
          LAMBY == mamaille$LAMBY))
  rm(dta) ; gc() # Pour vider la mémoire vive
}
```

```
comment(tabSite) <- paste("Généré le : ", Sys.time())
save(tabSite, file = "../../web/donnees/histoMeteo/tabSite.Rda")

system.time(load("../../web/donnees/histoMeteo/tabSite.Rda"))
   user  system elapsed
  0.023   0.002   0.025
print(object.size(tabSite), units = "GiB", digits = 3)
0.006 GiB
nrow(tabSite)
[1] 23894
length(seq.Date(as.Date("1958-08-01"), as.Date("2023-12-31"), by  = "day"))
[1] 23894
```

Les données climatiques pour une maille ne pèsent plus que 0.006 Gio, d'où un temps de chargement quasi-instantané (de l'ordre de 20 millisecondes sur mon ordinateur). Le nombre de lignes correspond au nombre de jours entre 1er août 1958 au 31 décembre 2023.

5 Données GPS manquantes

Si vous ne connaissez pas les coordonnées GPS mais que vous seriez capables de vous repérer sur une carte routière, le plus simple est peut-être de faire une carte interactive avec le paquet `leaflet` [1]. En cliquant sur la maille SAFRAN vous visualiserez ses coordonnées en Lambert II hectométrique. Le code suivant a été utilisé pour produire la figure 2 page 8.

```
n <- nrow(mailles) ; id <- rep(1:n, each = 4) ; part <- rep(1, n)
# Calcul des coordonnées des sommets des carrés des mailles
shm <- 100
lon <- shm*rep(mailles[1:n, "LAMBX"], each = 4)
lon <- lon + shm*40*c(-1, +1, +1, -1)
lat <- shm*rep(mailles[1:n, "LAMBY"], each = 4)
lat <- lat + shm*40*c(+1, +1, -1, -1)
lonlat <- cbind(id, part, lon, lat)
# Création d'un SpatVector en Lambert II
safgridLII <- terra::vect(lonlat, type = "polygons", crs = "EPSG:27572")
safgridGPS <- terra::project(safgridLII, "EPSG:4326")
# Pour pouvoir visualiser les coordonnées
safgridGPS$LAMBX <- mailles$LAMBX ; safgridGPS$LAMBY <- mailles$LAMBY
mamap <- terra::plet(safgridGPS, col = "transparent")
mamap
```

Références

[1] J. Cheng, B. Schloerke, B. Karambelkar, and Y. Xie. *leaflet : Create Interactive Web Maps with the JavaScript 'Leaflet' Library*, 2024. R package version 2.2.2.

[2] R.J. Hijmans. *terra : Spatial Data Analysis*, 2024. R package version 1.7-71.

[3] R.J. Hijmans, M. Barbosa, A. Ghosh, and A. Mandel. *geodata : Download Geographic Data*, 2023. R package version 0.5-9.

[4] Sun Microsystems. XDR : external data representation standard. RFC 1014. Technical report, Network Working Group, 1987.

[5] J.-P. Vidal, E. Martin, L. Franchistéguy, M. Baillon, and J.-M. Soubeyroux. A 50-year high-resolution atmospheric reanalysis over France with the safran system. *International Journal of Climatology*, 30(11) :1627–1644, 2010.

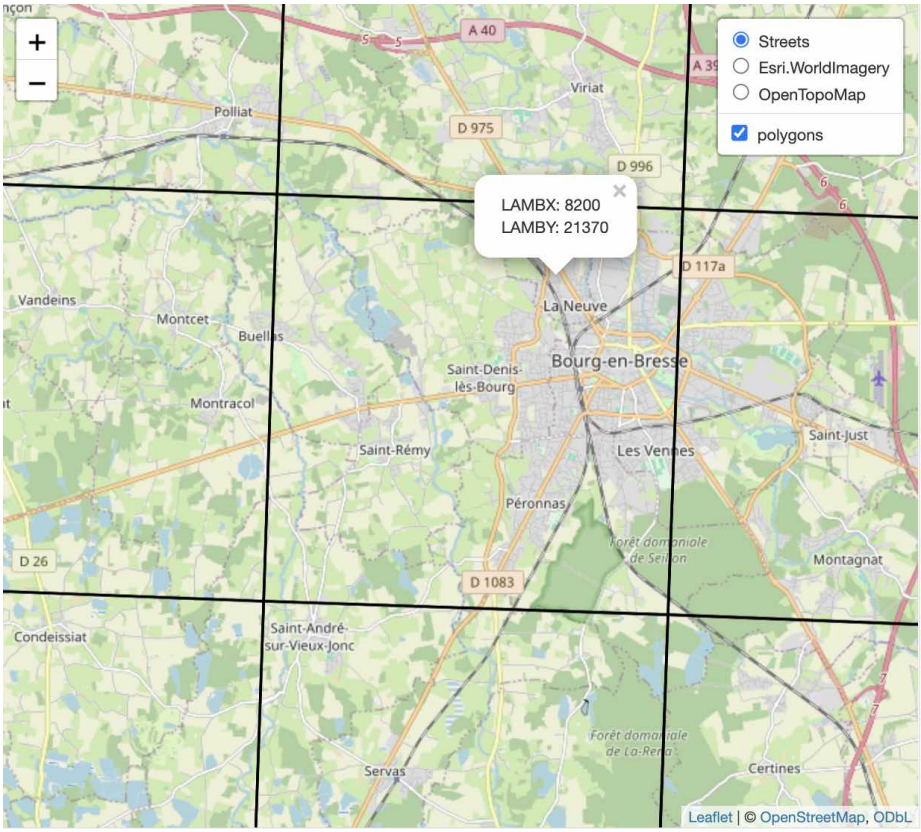


Figure 2 : Avec une carte interactive on peut facilement récupérer les coordonnées en Lambert II hectométrique d'une maille SAFRAN donnée. On a retrouvé ici « à la main » la maille SAFRAN correspondant au site RNSA de Bourg-en-Bresse. Si on compare avec les valeurs retournées par la fonction `getmaille()` dans la section 4.1 page 4 tout semble cohérent.