Fiche TD avec le logiciel R : CDN

Comment ajouter des courbes de niveau à une carte déjà dessinée ?

Pr Jean R. Lobry

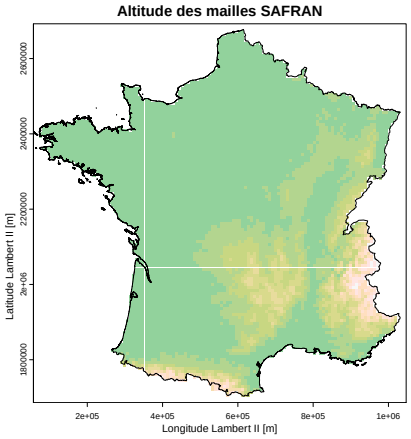
Cette fiche donne des indications pour ajouter des courbes de niveau sur une carte. On distingue deux cas selon que les données sont disposées sur une grille régulière ou non.

Table des matières

1 Données illustratives

On utilise ici les données sur l'altitude[1] [1] des mailles SAFRAN [2]. Avec une résolution de 8 × 8 km on a l'information sur l'altitude moyenne en mètre des mailles. La représentation de cette information nous donne la localisation des principaux massifs montagneux.

```
chmin <- "https://pbil.univ-lyon1.fr/R/donnees/CDN/"
load(url(paste0(chmin, "mailles_safran_drias-20200206.RData")))
# Sans la Corse pour ne pas avoir de discontinuité spatiale :
x <- subset(x, longitude < 8.5)
ptsGPS <- with(x, terra::vect(cbind(longitude, latitude), crs = "EPSG:4326"))
ptsGPS$z <- x$altitude
ptsLII <- terra::project(ptsGPS, "EPSG:27572")
FR <- geodata::gadm("France", 0, path = ".")
FRLII <- terra::project(FR, "EPSG:27572")
myexample <- function(mySV, myFR, nclass = 10, breaks = NULL, palette = "Terrain", ...){
  if(is.null(breaks)){
    breaks <- seq(min(mySV$z), max(mySV$z), length = nclass + 1)
  } else {
    nclass <- length(breaks) - 1
  }
  mypal <- hcl.colors(nclass, palette, alpha = 0.5)
  cls <- cut(mySV$z, breaks = breaks, include.lowest = TRUE)
  terra::plot(mySV, pch = 15, col = mypal[cls], cex = 0.5, ...)
  terra::lines(myFR)
}
myexample(ptsLII, FRLII,
          main = "Altitude des mailles SAFRAN", xlab = "Longitude Lambert II [m]",
          ylab = "Latitude Lambert II [m]")
```



[1] https://doi.org/10.57745/1PDFNL

2 Cas de données sur une maille régulière

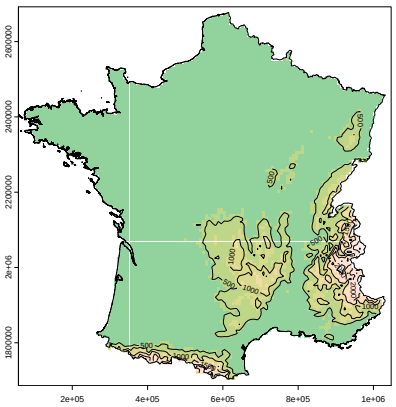
On commence par récupérer dans la table `df` les coordonnées `x` et `y` en Lambert II des mailles SAFRAN que l'on arrondi à l'hectomètre pour pallier les erreurs d'arrondi lors de la projection. Dans la colonne `z` on récupère l'altitude moyenne en mètre de la maille SAFRAN.

```
df <- as.data.frame(terra::geom(ptsLII))[ , c("x", "y")]
df$x <- 100*round(df$x/100) # arrondi à l'hectomètre
df$y <- 100*round(df$y/100)
df$z <- ptsLII$z
head(df)
```

```
      x       y z
1 604000 2673000 2
2 612000 2673000 2
3 572000 2665000 4
4 580000 2665000 3
5 588000 2665000 2
6 596000 2665000 1
```

On ne peux pas utiliser directement la table `df` comme argument de la fonction de base `graphics::contour()` pour tracer les courbes de niveau. En effet, elle s'attend à ce que les données d'altitude soient présentées sous la forme d'une matrice `z` (avec éventuellement des `NA`) dont on peut préciser les coordonnées, dans l'ordre croissant, avec les vecteurs `x` et `y`. On a la possibilité de rassembler ces trois éléments dans une liste. On définit donc la fonction `mkxyzlist()` pour créer cette liste à partir de la table `df`.

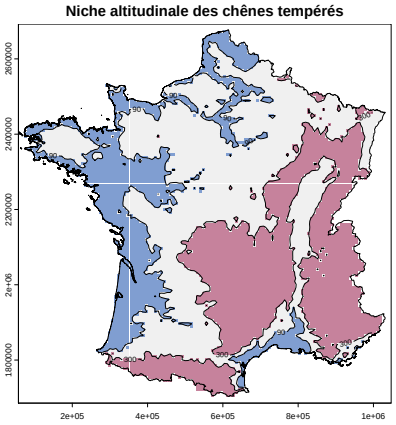
```
mkxyzlist <- function(df, xname = "x", yname = "y", zname = "z"){
  x <- unique(sort(df[ , xname])) ; nx <- length(x)
  y <- unique(sort(df[ , yname])) ; ny <- length(y)
  z <- matrix(NA, nrow = nx, ncol = ny)
  for(i in seq_len(nrow(df))){
    ii <- which(x == df[i, xname])
    jj <- which(y == df[i, yname])
    z[ii, jj] <- df[i, zname]
  }
  return(list(x = x, y = y, z = z))
}
xyzlist <- mkxyzlist(df)
myexample(ptsLII, FRLII, nclass = length(pretty(x$altitude, 10)))
graphics::contour(xyzlist, add = TRUE)
```



Et voilà, le tour est joué ! On peut maintenant ne pas être satisfait par le rendu, les courbes représentées par défaut sont celles retournées par

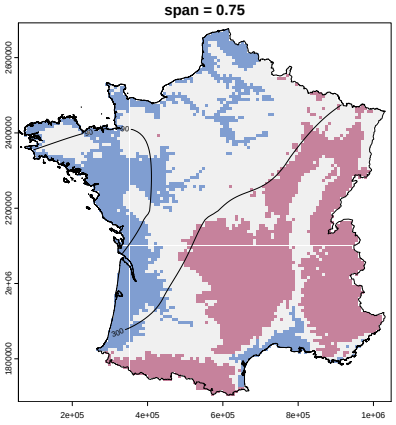
`pretty(df$z, 10)` et comme la majorité (80 %) des mailles sont en dessous de 500 m, on voit surtout les courbes des massifs montagneux. Mais supposons que nous soyons intéressés par la niche altitudinale des chênes tempérés entre 90 et 300 m, cette carte n'est vraiment pas informative. Heureusement, il est très facile de spécifier les courbes que l'on veut voir figurer avec l'argument `levels` de `contour()`.

```
# palette de type "diverging"
myexample(ptsLII, FRLII, breaks = c(0, 90, 300, 4000), main = "Niche altitudinale des chênes tempérés", palett
graphics::contour(xyzlist, levels = c(90, 300), add = TRUE)
```



Le deuxième aspect sur lequel on aimerait pouvoir jouer c'est de lisser un peu les courbes de niveau. L'idée est d'utiliser la fonction `loess()` pour faire une régression polynomiale locale en fonction des coordonnées spatiales. L'altitude de chaque point va être moyennée avec celle de ses voisins, les reliefs vont être érodés et ainsi les courbes de niveau plus régulières.

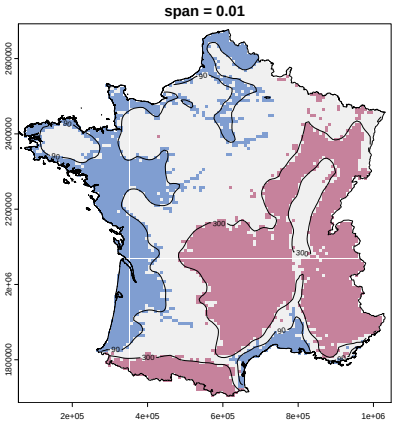
```
df$fitted <- loess(z~x+y, df)$fitted
myexample(ptsLII, FRLII, breaks = c(0, 90, 300, 4000), main = "span = 0.75", palette = "Blue-Red")
graphics::contour(mkxyzlist(df, z = "fitted"), levels = c(90, 300), add = TRUE)
```



Le résultat n'est pas très satisfaisant parce que l'on a utilisé la valeur par défaut `span = 0.75` de la fonction `loess()`. Les $\frac{3}{4}$ des points sont considérés

comme faisant parti du voisinage, l'érosion du relief est bien trop sévère. Il faut donc jouer avec la valeur du paramètre `span` pour obtenir une représentation plus proche de ce qui est souhaité.

```
df$fitted <- loess(z~x+y, df, span = 0.01)$fitted
myexample(ptsLII, FRLII, breaks = c(0, 90, 300, 4000), main = "span = 0.01",
          palette = "Blue-Red")
graphics::contour(mkxyzlist(df, z = "fitted"), levels = c(90, 300), add = TRUE)
```

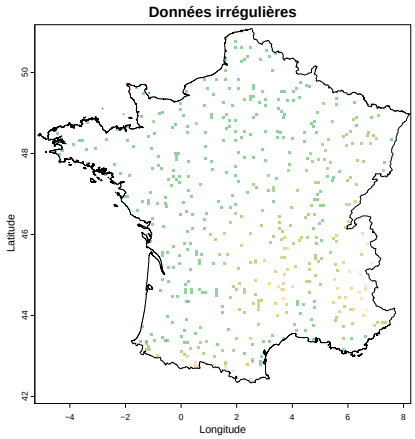


3 Cas de données n'étant pas sur une maille régulière

Ce cas de figure va se rencontrer si vos données sont issues de stations réparties plus ou moins aléatoirement sur le territoire[2]. Pour simuler ce cas de figure nous allons considérer un sous ensemble tiré au hasard des mailles SAFRAN en coordonnées GPS pour casser la belle régularité des coordonnées en Lambert II.

```
set.seed(1) # pour la reproductibilité
subGPS <- ptsGPS[sample(1:nrow(ptsGPS), size = 500, replace = FALSE), ]
myexample(subGPS, FR, xlab = "Longitude", ylab = "Latitude",
          main = "Données irrégulières", xlim = c(-5, 8), ylim = c(42, 51))
```

[2]Pour une présentation très didactique et graphique de la démarche suivie ici on pourra se reporter à la fiche « [c]ourbes de niveau » à `https://pbil.univ-lyon1.fr/R/pdf/tdr26.pdf`

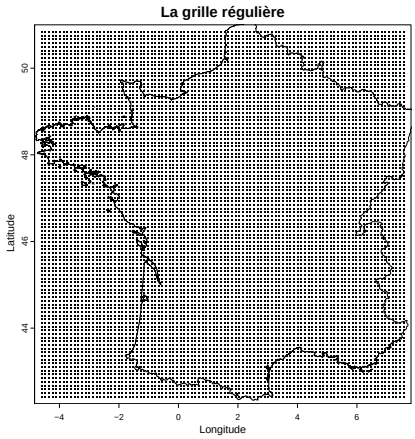


On construit la table `dfGPS` avec les coordonnées GPS et l'altitude du sous-ensemble des mailles SAFRAN. On utilise ces données pour faire une régression polynomiale locale de l'altitude en fonction des coordonnées géographiques.

```
dfGPS <- as.data.frame(terra::geom(subGPS))[ , c("x", "y")]
dfGPS$z <- subGPS$z
mylo <- loess(z~x+y, dfGPS, span = 0.1)
```

On utilise maintenant ce modèle pour interpoler les valeurs d'altitude sur une grille régulière. C'est la fonction `expand.grid()` qui permet de générer facilement toutes les valeurs de cette grille. On utilise ici une grille de 100 × 100 éléments pour avoir à peu près la même résolution que pour les mailles SAFRAN.

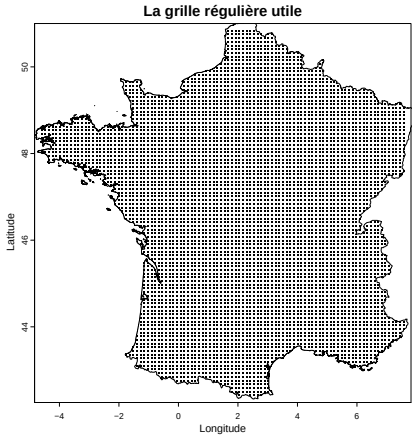
```
mygrid <- with(dfGPS, expand.grid(seq(min(x), max(x), le = 100),
                                  seq(min(y), max(y), le = 100)))
colnames(mygrid) <- c("x", "y")
ptsGrid <- with(mygrid, terra::vect(cbind(x, y), crs = "EPSG:4326"))
terra::plot(ptsGrid, pch = ".", main = "La grille régulière", xlab = "Longitude",
            ylab = "Latitude")
terra::lines(FR)
```



Le problème de cette grille est qu'elle comporte de nombreux points en dehors de France, et que l'on veut éviter de faire des extrapolations. La fonction

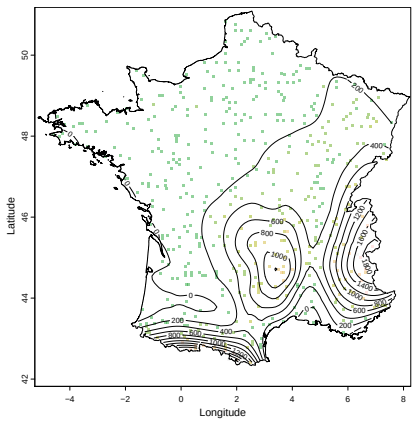
`terra::intersect()` nous permet d'éliminer facilement tout ce qui est hors zone[3].

```
ptsGrid <- terra::intersect(ptsGrid, FR)
terra::plot(ptsGrid, pch = ".", main = "La grille régulière utile", xlab = "Longitude",
            ylab = "Latitude")
terra::lines(FR)
```



Il suffit maintenant d'utiliser notre modèle pour interpoler les valeurs de l'altitude sur cette grille régulière. À partir de ce point, nous nous sommes ramenés au cas traité dans la section 2 page 3.

```
dfGrid <- as.data.frame(terra::geom(ptsGrid))[ , c("x", "y")]
dfGrid$z <- predict(mylo, newdata = dfGrid)
myexample(subGPS, FR, xlab = "Longitude", ylab = "Latitude",
          xlim = c(-5, 8), ylim = c(42, 51))
graphics::contour(mkxyzlist(dfGrid), add = TRUE)
```



[3]Si vous n'avez pas de fond de carte de référence, un palliatif est d'éliminer tout ce qui n'est pas dans l'enveloppe convexe des points observés avec `chull()`.

4 Annexe

Le code R utilisé pour faire la petite animation sur le site[4] montrant l'effet du paramètre `span` de `loess()` sur le rendu des courbes de niveau est le suivant :

```
spans <- seq(from = 0.5, to = 0.01, by = -0.01) ; ns <- length(spans)
for(i in seq_len(ns)){
  ichar <- sprintf("%06d", i)
  png(paste0("png/image", ichar, ".png"), width = 6, height = 6, units = "in", res = 36)

  df$fitted <- loess(z~x+y, df, span = spans[i])$fitted
  myexample(ptsLII, FRLII, breaks = c(0, 90, 300, 4000), main = paste("span =", round(spans[i], 2)), palette =
  graphics::contour(mkxyzlist(df, z = "fitted"), levels = c(90, 300), add = TRUE)

  dev.off()
}
library(gifski)
png_files <- list.files(path = "png/", pattern = ".png", full.names = TRUE)
gifski::gifski(png_files, gif_file = "CDN.gif",
               delay = 0.5)
```

Références

[1] P. Bertuzzi and P. Clastre. Information sur les mailles SAFRAN, 2022.

[2] 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.

[4] http://pbil.univ-lyon1.fr/R/enseignement_div.php?contents=html/quercus