Krigeage avec R

Eric Marcon

11 décembre 2023

Résumé

Techniques pour interpoler les valeurs d'une variable continue.

Table des matières

Pour cartographier facilement une variable continue, 5 méthodes sont disponibles, dans les packages **akima**, **spatial**, **spatstat**, **gstat** et **automap**.

Des méthodes plus élaborées ne sont pas traitées ici :

- l'estimation d'un modèle de prédiction de la valeur à partir de variables explicatives (krigeage ordinaire ou krigeage universel). Voir l'aide de la fonction `gstat` pour leur utilisation.
- l'estimation bayesienne de ces modèles ou le co-krigeage (estimation de plusieurs variables non indépendantes). Les packages **RGeostats** [1] ou **INLA** [2] permettent une modélisation complexe, mais au prix d'un effort bien supérieur.

[1] http://rgeostats.free.fr/
[2] http://www.r-inla.org/spde-book

1 Interpolation et cartographie locales

1.1 Création des données

Les données représentent le niveau de concentration spatiale locale autour de points (Marcon and Puech, 2023).

Un jeu de deux types de points est généré dans une fenêtre carrée. Les cas sont générés par un processus de Matérn (Matérn, 1960) avec κ agrégats (attendus) de μ points (attendus) dans un cercle de rayon *scale*. Les contrôles suivent un processus de Poisson dont la densité λ décroit exponentiellement vers le haut de la carte.

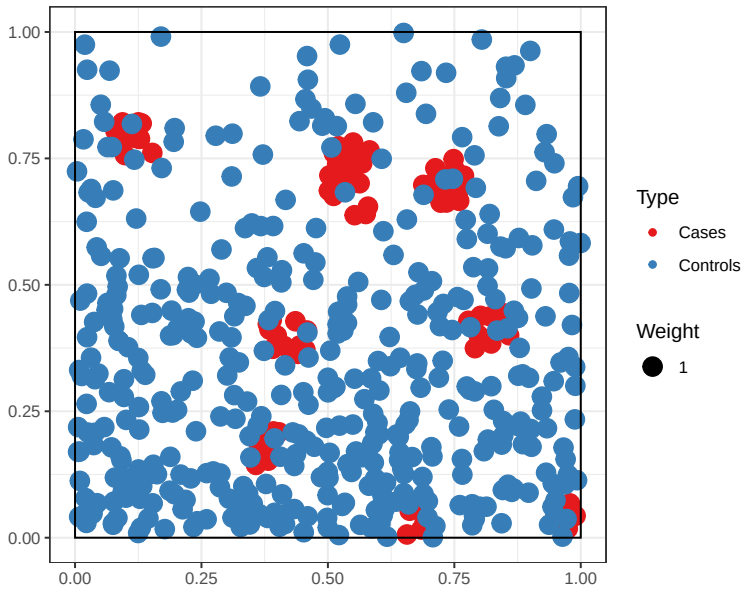
```
library("dplyr")
library("dbmss")
# Simulation of cases (clusters)
rMatClust(kappa = 10, scale = 0.05, mu = 10) %>%
    as.wmppp -> CASES
CASES$marks$PointType <- "Cases"
# Number of points
CASES$n
```

```
## [1] 97
```

```
# Simulation of controls (random distribution)
rpoispp(function(x, y) {
    1000 * exp(-2 * y)
}) %>%
    as.wmppp -> CONTROLS
CONTROLS$marks$PointType <- "Controls"
# Number of points
CONTROLS$n
```

```
## [1] 435
```

```
# Mixed patterns (cases and controls)
ALL <- superimpose(CASES, CONTROLS)
autoplot(ALL)
```



La valeur de la concentration relative autour de chaque cas à la distance $r = 0,1$ est définie comme le rapport entre la proportion de cas observés et la proportion moyenne sur tout le domaine d'étude. Si elle est supérieure à 1, on observe plus de voisins qu'attendu.

La valeur est calculée en chaque point et stockée dans un objet de type `fv` dont les valeurs sont récupérées pour produire un dataframe.

```
# Choose the distance to plot
Distance <- 0.1
# Calculate the M values to plot
ALL %>%
    Mhat(r = c(0, Distance), ReferenceType = "Cases", NeighborType = "Cases",
        Individual = TRUE) -> M_TheoEx
# Make a dataframe with x, y and M
the_df <- data.frame(x = CASES$x, y = CASES$y, z = as.numeric(as.data.frame(M_TheoEx)[2,
    -(1:3)]))
```

A ce stade, nous disposons d'un tableau contenant les coordonnées et une valeur décrivant chaque point.

```
head(the_df)
```

```
##           x         y        z
## 1 0.12834741 0.7888727 3.519886
## 2 0.15282281 0.7608785 3.687500
## 3 0.10257350 0.8022610 3.226562
## 4 0.09695011 0.7898824 3.366848
## 5 0.08039593 0.7800437 3.366848
## 6 0.12748183 0.7886126 3.519886
```

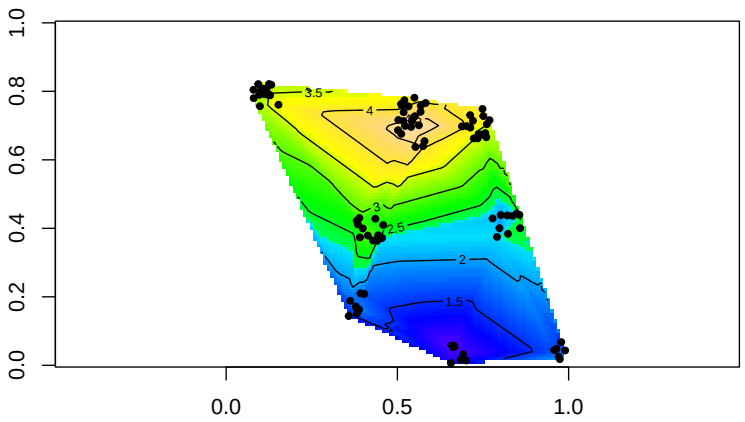
L'objectif est maintenant de cartographier cette valeur.

1.2 Cartographie

1.2.1 akima

La méthode d'Akima est une interpolation entre les valeurs des points, faite dans chaque triangle constitué par les triplets de points les plus proches. La valeur des points est conservée.

```
library("akima")
# interp() function
akima_interp <- interp(
  # Coordinates
  x = the_df$x,
  y = the_df$y,
  # Value
  z = the_df$z,
  # Coordinates to interpolate
  xo = seq(from = 0, to = 1, by = .01),
  yo = seq(from = 0, to = 1, by = .01)
)
# Map
image(akima_interp, col = topo.colors(128, alpha = 1), asp = 1)
# Contour lines
contour(akima_interp, add = TRUE)
# Add the points
with(the_df, points(x, y, pch = 20))
```



L'interpolation se limite au polygone convexe contenant les points.

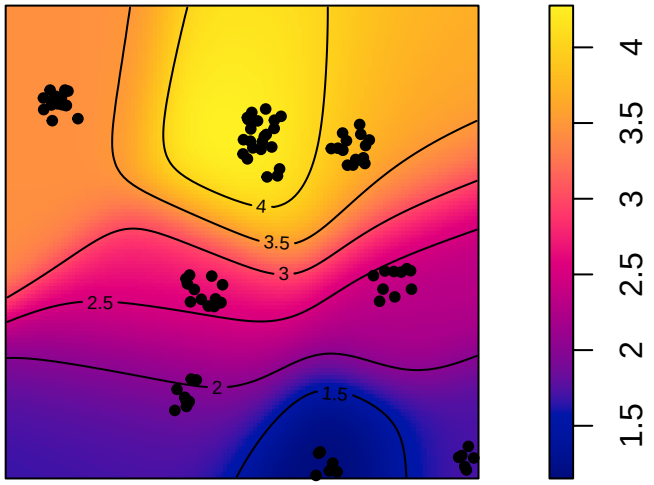
1.2.2 spatstat

La fonction `Smooth.ppp()` du package **spatstat** permet d'obtenir le même résultat mais ne se limite pas au polygone contenant les points.

Elle s'applique à un objet de type `ppp` (*planar point pattern*) à créer.

```
library("spatstat")
the_df %>%
    # Create a ppp in a square window
as.ppp(W = square(1)) %>%
    # Smooth it. The bandwidth is computed by bw.scott()
Smooth(sigma = bw.scott(.)) -> spatstat_interp
```

```
# Plot the result
plot(spatstat_interp, main = "")
# Contour lines
contour(spatstat_interp, add = TRUE)
# Add the points
with(the_df, points(x, y, pch = 20))
```



Le choix de la largeur de bande est capital. La règle de Scott, implémentée dans `bw.scott()` est généralement efficace.

1.2.3 spatial

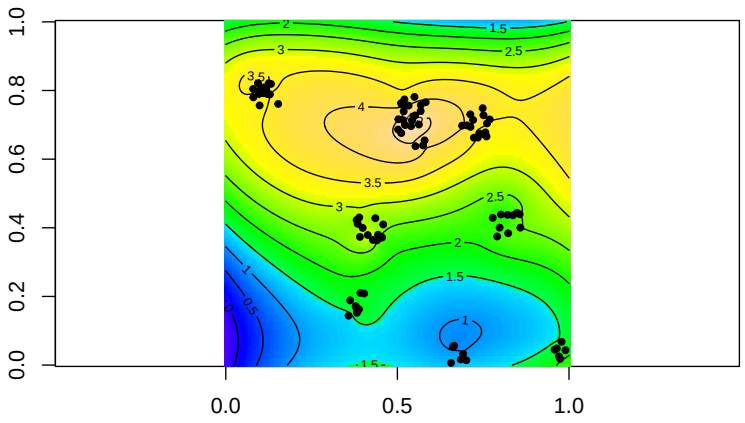
Le package **spatial** permet de kriger au lieu de simplement interpoler, mais renvoie des erreurs si la méthode de calcul de la covariance n'est pas exponentielle. L'ordre du polynôme du modèle et la distance de dépendance doivent être choisis explicitement.

```
library("spatial")
```

```
##
## Attaching package: 'spatial'

## The following object is masked from 'package:spatstat.model':
##
##     Strauss
```

```
# Function surf.gls() creates a trend surface object
surf.gls(np = 3, covmod = expcov, x = the_df$x, y = the_df$y, z = the_df$z,
    d = 0.5) %>%
    # prmat() performs kriging over a grid
prmat(xl = 0, xu = 1, yl = 0, yu = 1, n = 128) -> spatial_krig
# Map it
image(spatial_krig, col = topo.colors(128, alpha = 1), asp = 1)
# Contour lines
contour(spatial_krig, add = TRUE)
# Add the points
with(the_df, points(x, y, pch = 20))
```



1.2.4 gstat

Le package **gstat** étend les possibilités de krigeage en permettant de spécifier un modèle de tendance pour la variable cartographiée (inutile ici, on utilise `formula=Richness~1`). Le variogramme doit être calculé et un modèle ajusté (dans l'exemple, un modèle gaussien et non exponentiel).

```
library("sp")
# Create a SpatialPointsDataFrame with the data
sp_spdf <- SpatialPointsDataFrame(
  # Coordinates are x and y
  coords = the_df[, 1:2],
  # Data is z
  data = the_df[3]
)
library("gstat")
```

```
##
## Attaching package: 'gstat'

## The following object is masked from 'package:spatial':
##
##     variogram

## The following object is masked from 'package:spatstat.explore':
##
##     idw
```

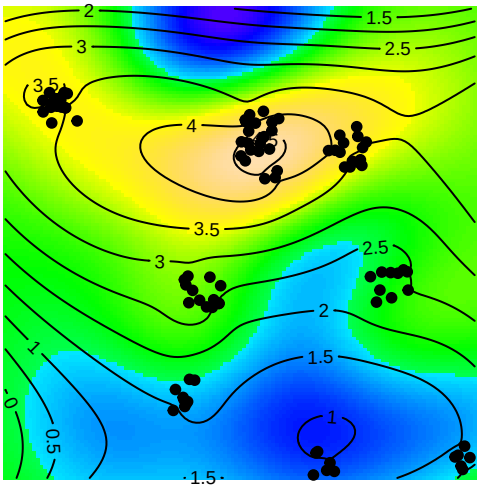
```
# Empirical variogram
gstat::variogram(z ~ 1, data = sp_spdf) %>%
  # Fit a Gaussian model
  fit.variogram(vgm("Gau")) ->
  gstat_variogram

# Build an oject that describes all features of the model.
# The formula defines the trend
geoX <- gstat(
  formula = z ~ 1,
  locations = sp_spdf,
  model = gstat_variogram
)
# Préparation d'une grille de 128 points de côté
xy <- expand.grid((0:128) / 128, (0:128) / 128)
```

```
names(xy) <- c("x", "y")
gridded(xy) <- ~ x + y
# Calcul de la valeur de Richness sur les points de la grille (krigeage)
geoXprd <- predict(geoX, newdata = xy)
```

```
## [using ordinary kriging]
```

```
# Carte
image(geoXprd, col = topo.colors(128, alpha = 1), asp=1)
# Contour lines
contour(spatial_krig, add = TRUE)
# Add the points
with(the_df, points(x, y, pch = 20))
```



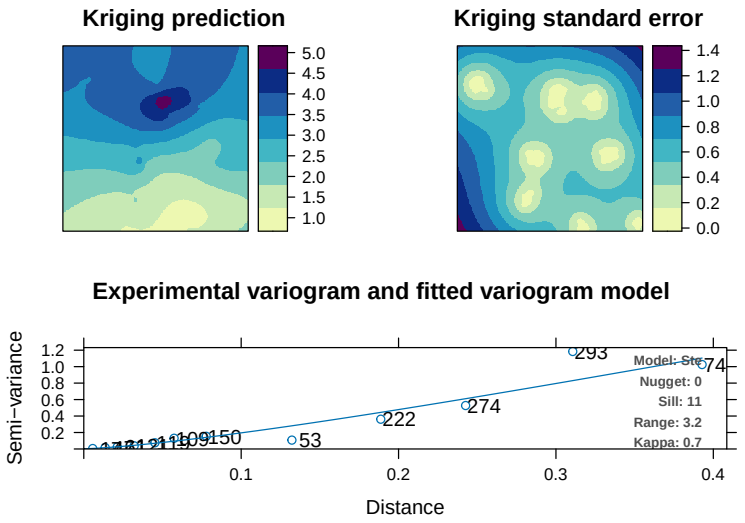
1.2.5 automap

Le package **automap** s'appuie sur **gstat** mais automatise toutes les étapes de sélection du modèle de covariance (celui qui s'ajuste le mieux aux données est choisi). Le modèle sélectionné est affiché dans le variogramme. La grille précédente peut être utilisée, mais une grille calculée à partir de la fenêtre de l'objet 'ppp (librairie **spatstat**) est plutôt utilisée ici.

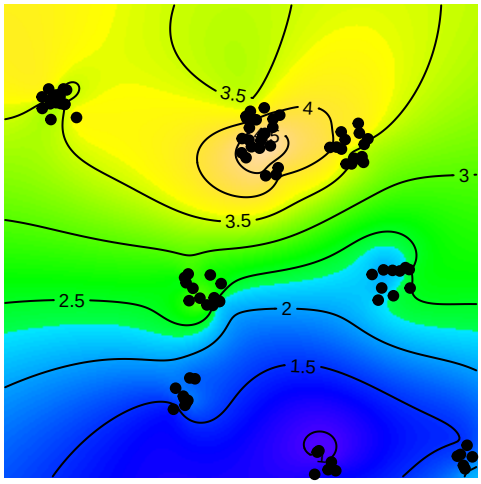
```
library("spatstat")
# Build a ppp object
the_df %>%
    # Create a ppp in a square window
as.ppp(W = square(1)) -> the_ppp
# Preparing a 256-square-point grid
xy <- gridcentres(the_ppp, 256, 256)
# Filtering of grid nodes inside the window (not necessary here)
ok <- inside.owin(xy$x, xy$y, the_ppp)
# Formatting the grid
sp_grid <- SpatialPoints(cbind(xy$x[ok], xy$y[ok]))
gridded(sp_grid) <- TRUE
# Kriging the SpatialPointsDataFrame
library("automap")
automap_krig <- autoKrige(formula = z ~ 1, input_data = sp_spdf, new_data = sp_grid)
```

```
## [using ordinary kriging]
```

```
# Result
plot(automap_krig)
```




```
# The usual map
image(automap_krig$krige_output, col = topo.colors(128, alpha = 1), asp = 1)
contour(automap_krig$krige_output, add = TRUE)
with(the_df, points(x, y, pch = 20))
```



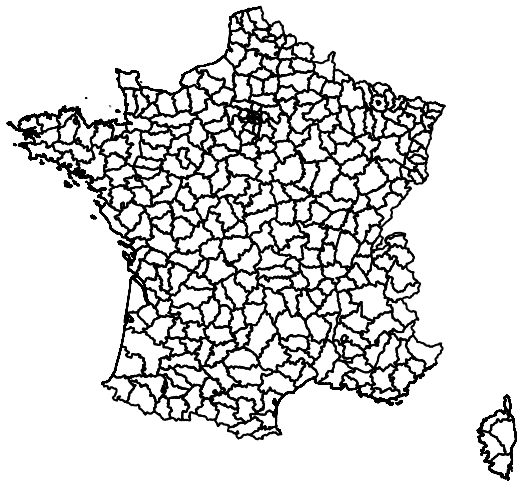
2 Utilisation de fonds de carte

L'objectif est ici d'interpoler une variable continue du même type sur les centroïdes de polygones d'une carte vectorielle (un shapefile) plutôt que sur une grille.

2.1 Obtention des cartes

Le package *raster* permet de télécharger des fonds de carte administratifs, des modèles numériques de terrain, des cartes de climat : voir l'aide de la fonction `getData`.

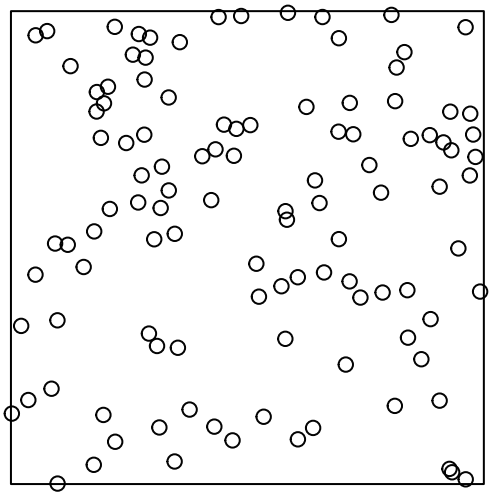
```
library("raster")
# Retrieving the shapefile for the boundaries of France's regions
France <- raster::getData("GADM", country = "FRA", level = 3)
# Projection of France in Lambert 93
France <- spTransform(France, CRS("+init=epsg:2154"))
plot(France)
```



2.2 Fabrication des données

Les données sont 100 points placés aléatoirement dans un rectangle contenant la France. Leur marque est une valeur numérique continue, augmentant linéairement de l'ouest vers l'est et avec la distance à la latitude moyenne, et contenant un bruit gaussien.

```
library("spatstat")
# Draw in a Poisson process, 1000 expected points, in a 1x1 window
plot(X <- rpoispp(100), main = "")
```

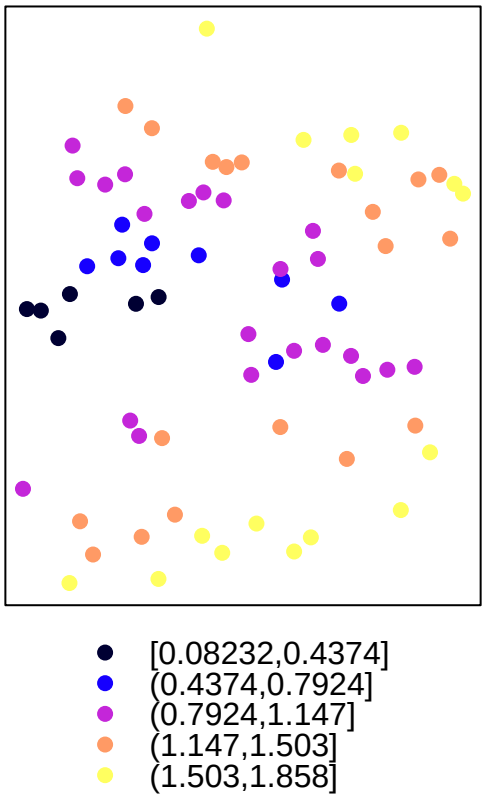


```
# Values of the mark
X$marks <- X$x + 3 * abs(X$y - 0.5) + rnorm(X$n, sd = 0.1)
# Rough projection on the map
X$x <- 8e+05 * X$x + 3e+05
X$y <- 9e+05 * X$y + 6200000
```

2.3 Interpolation

Les valeurs de x, y et z doivent être intégrées dans un objet *Spatial*.

```
# Make a spatial point dataframe
SpatialX <- SpatialPointsDataFrame(coords = data.frame(x = X$x, y = X$y),
    data = data.frame(m = X$marks), proj4string = CRS("+init=epsg:2154"))
# Trim it
SpatialX <- SpatialX[France, ]
# Map the points
spplot(SpatialX, "m")
```

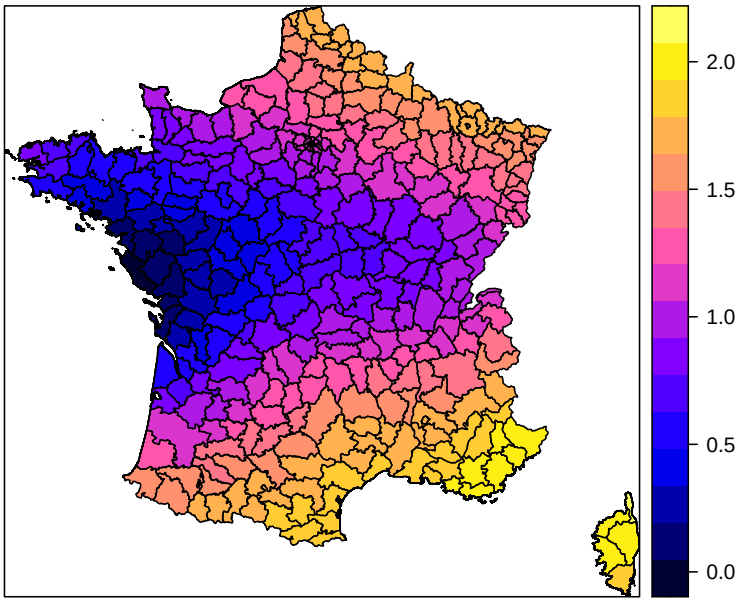


L'interpolation est faite avec *gstat*.

```
library("gstat")
# Empirical variogram
gstat::variogram(m ~ 1, data = SpatialX) %>%
    # Fit a Gaussian model
fit.variogram(vgm("Gau")) -> gstat_variogram
# Build an oject that describes all features of the model.
geoX <- gstat(formula = m ~ 1, locations = SpatialX, model = gstat_variogram)
# Calculating the value of m on the centroids of polygons
geoXprd <- predict(geoX, newdata = France)
```

```
## [using ordinary kriging]
```

```
# Final map
spplot(geoXprd, "var1.pred")
```



Références

Marcon, É. and F. Puech (2023, December). Mapping distributions in non-homogeneous space with distance-based methods. *Journal of Spatial Econometrics 4*(1), 13.

Matérn, B. (1960). Spatial variation. *Meddelanden från Statens Skogsforskningsinstitut 49*(5), 1–144.