Visualisation de processus spatiaux à l'aide de la correction de Ripley

Arthur Charpentier[a], Ewen Gallic[b]
[a] Département de mathématiques, UQAM, charpentier.arthur@uqam.ca
[b] Faculté des sciences économiques, Rennes 1, ewen.gallic@etudiant.univ-rennes1.fr

Introduction

Dans la plupart des cas, les emplacements des accidents de la route ne sont pas purement aléatoires mais forment des *points chauds*. L'analyse de la structure spatiale d'observations ponctuelles permet de les identifier [1]. Une estimation de la densité peut être réalisée à l'aide d'un noyau gaussien. Cependant, cette méthode souffre d'un effet de frontière [2], corrigible par la méthode de la circonférence de Ripley.

Méthode

Soit $\boldsymbol{Z} = (X, Y)$, avec X une latitude et Y une longitude. L'estimation de la densité à un point $\boldsymbol{z} = (x, y)$ est donnée par :

$$\widehat{f}(\boldsymbol{z}) = \frac{1}{nh_Xh_Y}\sum_{i=1}^{n} K_Z\left(\frac{x-X_i}{h_X}, \frac{y-Y_i}{h_Y}\right), \tag{1}$$

avec $K_Z(\cdot)$ un noyau symétrique, par exemple gaussien, n le nombre d'observations, h_X et h_Y les largeurs des fenêtres spatiales.

Effet de bord

Considérons des points $\boldsymbol{Z}_i$ appartenant à une surface $\mathcal{S}$. Le fait que la somme des poids n'égale pas 1 est problématique.
Les estimateurs par la méthode du noyau peuvent être vus comme l'espérance de la densité pour un échantillon $\{\widetilde{\boldsymbol{Z}}_i = \boldsymbol{Z}_i + \boldsymbol{\varepsilon}_i\}$, où les ε_i sont des bruits i.i.d. d'espérance nulle. On a donc :

$$\widehat{f}(\boldsymbol{z}) = \frac{1}{n}\sum_{i=1}^{n} \mu_{\boldsymbol{Z}_i}(\boldsymbol{z}), \tag{2}$$

avec $\mu_{\boldsymbol{Z}_i}$ la densité d'un vecteur gaussien centré en $\boldsymbol{Z}_i$ de matrice de variance-covariance $\boldsymbol{H}$. Si la densité de $\boldsymbol{Z}$ a un support compact, alors $\mu_{\boldsymbol{Z}_i}$ va distribuer des poids dans des zones où il est impossible de trouver des observations. Il semble alors naturel de considérer une distribution tronquée, limitée au support $\mathcal{S}$:

$$\mu_{\boldsymbol{Z}_i|\mathcal{S}}(\boldsymbol{z}) = \frac{\mu_{\boldsymbol{Z}_i}(\boldsymbol{z})}{\mu_{\boldsymbol{Z}_i}(\mathcal{S})}. \tag{3}$$

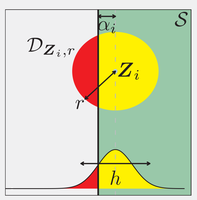
L'estimation de f devient :

$$\widehat{f}(\boldsymbol{z}) = \frac{1}{n}\sum_{i=1}^{n} \mu_{\boldsymbol{Z}_i|\mathcal{S}}(\boldsymbol{z}) = \frac{1}{n}\sum_{i=1}^{n} \omega_i \cdot \mu_{\boldsymbol{Z}_i}(\boldsymbol{z}) \quad \text{avec } \omega_i = \mu_{\boldsymbol{Z}_i}(\mathcal{S})^{-1}. \tag{4}$$

En considérant un bruit gaussien, $\mu_{\boldsymbol{Z}_i}$ peut être approchée par :

$$\mu^0_{\boldsymbol{Z}_i,r}(\mathcal{S}) = \frac{\mathcal{A}(\mathcal{D}_{\boldsymbol{Z}_i,r} \cap \mathcal{S})}{\mathcal{A}(\mathcal{D}_{\boldsymbol{Z}_i,r})}, \tag{5}$$

avec $\mathcal{A}$ la fonction d'aire et $\mathcal{D}_{\boldsymbol{Z}_i,r}$ le cercle de centre $\boldsymbol{Z}_i$ et de rayon r.



L'idée est alors d'utiliser un estimateur pondéré :

$$\widehat{f}(\boldsymbol{z}) = \sum_{i=1}^{n} \omega(\boldsymbol{Z}_i) \cdot \det(\boldsymbol{H})^{-1} K\left(\boldsymbol{H}^{-1}(\boldsymbol{z}-\boldsymbol{Z}_i)\right), \tag{6}$$

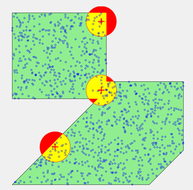
où les poids $\omega(\boldsymbol{Z}_i)$ rendent compte de la part de l'aire autour de $\boldsymbol{Z}_i$ qui appartient à $\mathcal{S}$. Les poids de l'expression 6 dépendent du rayon de $\mathcal{D}_{\boldsymbol{Z}_i,r}$. Sous l'hypothèse où K est le produit de deux noyaux gaussiens, la matrice de variance-covariance $\boldsymbol{H}$ est diagonale. Sous l'hypothèse supplémentaire que les éléments h de la diagonale sont identiques, on peut penser que le rayon optimal r^* dépend linéairement de h, i.e. $r^* = \beta^* h$. Dans le cas où $\mathcal{S}$ est un demi plan, on peut montrer que $r^* = 1.759h$.

Simulations

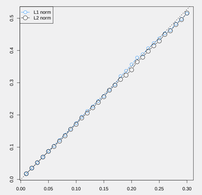
Dans un cas plus général, on tire uniformément 1,000 points dans une surface. Les poids *théoriques* $\omega_i(h)$ sont calculés par simulations de Monte Carlo. Pour différentes valeurs de r, les poids $\omega_i^0(h)$, basés sur $\mu^0_{\boldsymbol{Z}_i,r}(\mathcal{S})$, sont aussi calculés. Pour une norme $\|\cdot\|$ donnée, le rayon optimal r^*, en fonction de la fenêtre h, est donné par

$$r^\star(h) = \text{argmin}\left\{\sum_{i=1}^{n} \|\omega_i^0(h) - \omega_i(h)\|\right\}. \tag{7}$$

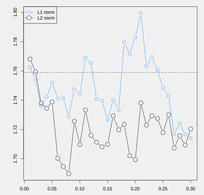
Une relation linéaire peut être identifiée pour la norme L_1 comme pour la norme L_2 :



Échantillon (taille 1000).



$r^*(h)$ en fonction de h.



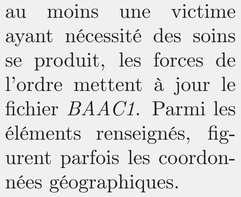
$r^*(h)$ sur h en fonction de h.

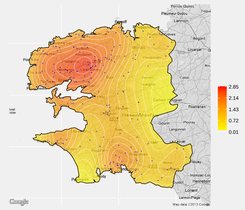
Application

Lorsqu'un accident corporel impliquant au moins un véhicule et ayant fait au moins une victime ayant nécessité des soins se produit, les forces de l'ordre mettent à jour le fichier *BAAC1*. Parmi les éléments renseignés, figurent parfois les coordonnées géographiques.
En 2008, le nombre d'accidents géolocalisés dénombrés dans le département du Finistère s'élève à 186, et à 180 dans le Morbihan.

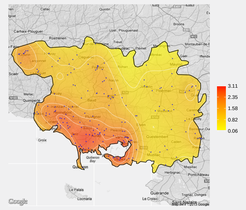
Il est possible de lier la densité des accidents avec celle des routes. OpenStreetMap fournit, pour plusieurs types de routes, les coordonnées des extrémités de tronçons. Une interpolation entre ces extrémités permet d'obtenir des points, et donc d'appliquer la méthode d'estimation de la densité.

Le ratio de la densité d'accidents sur celle des routes permet alors d'identifier les *points-chauds*.

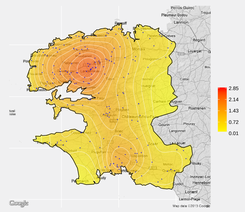




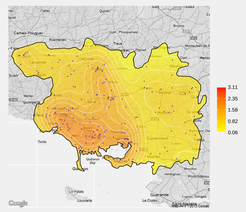
Densité accidents Finistère (avec correction)



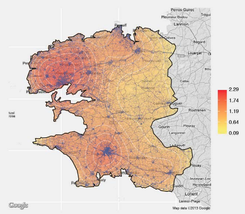
Densité accidents Morbihan (avec correction)



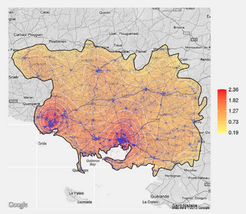
Densité accidents Finistère (sans correction)



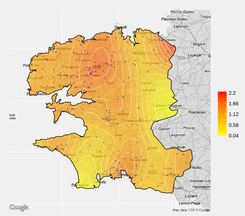
Densité accidents Morbihan (sans correction)



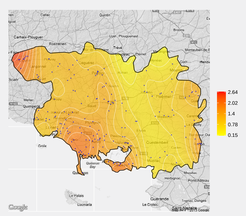
Densité routes Finistère



Densité routes Morbihan



Hot-spots Finistère



Hot-spots Morbihan

Code R

La mise en pratique dans R est facilitée par l'existence des fonctions `area.poly` et `intersect` du *package* `rgeos` qui permettent de calculer l'aire d'intersection de polygones. Si `sCircle` permet d'obtenir le polygone circulaire centré en `x`, c'est-à-dire :

```
sCircle <- function(n=100,centre=c(0,0),radius){
  theta <- seq(0,2*pi,length=n)
  m <- cbind(cos(theta),sin(theta))*radius
  m[,1] <- m[,1]+centre[1] ; m[,2] <- m[,2]+centre[2]
  colnames(m) <- c("x","y") ; return(m)
}
```

alors la fonction de poids associée à l'observation `x` est :

```
sWeights <-function(x,h,pol){
  leCercle <- sCircle(centre=x,radius=1.759*h)
  POLcercle <- as(leCercle[-nrow(leCercle),], "gpc.poly")
  return(area.poly(rgeos::intersect(pol,POLcercle))/area.poly(POLcercle))
}
```

L'estimation de la densité par la méthode du noyau commence par le calcul de la fenêtre optimale, puis par le calcul des poids, pour chaque observation :

```
H <- Hpi(X)
H <- matrix(c(sqrt(H[1,1]*H[2,2]),0,0,sqrt(H[1,1]*H[2,2])),2,2)
wHelp <- function(i){
  sWeights(x=as.numeric(X[i,]), h=sqrt(H[1,1]),
    pol=as(pol, "gpc.poly"))
}
OMEGA <- Vectorize(wHelp)(1:nrow(X))
```

Enfin, vient le calcul de l'estimateur du noyau pondéré, sur une grille :

```
fhat=kde(X,H,w=1/OMEGA,xmin=c(min(pol[,1]),min(pol[,2])),xmax=c(max(pol[,1]),max(pol[,2])))
fhat$estimate <- fhat$estimate*sum(1/OMEGA)/n
vx <- unlist(fhat$eval.points[1]) ; vy <- unlist(fhat$eval.points[2])
VX<-cbind(rep(vx,each=length(vy))); VY<-cbind(rep(vy,length(vx))); VXY<-cbind(VX,VY)
Ind <- matrix(point.in.polygon(VX,VY, pol[,1],pol[,2]),length(vy),length(vx))
f0 <- fhat ; f0$estimate[t(Ind)==0] <- NA
resul <- list(X=fhat$eval.points[[1]], Y=fhat$eval.points[[2]], Z=fhat$estimate, ZNA=f0$
    estimate,  H=fhat$H,W=fhat$w)
```

Soit `listroad` une liste dont les $n-1$ premiers éléments correspondent aux tronçons de route, le n^e élément étant le type de route. Soit `types.weights` une data frame dont chaque ligne associe les types de routes avec les poids nécessaires pour l'interpolation. Alors, `splitroad` permet d'obtenir les points représentant la route :

```
splitroad <- function(listroad, h=0.0025){
  pts <- NULL
  weights <- types.weights[match(unique(listroad$type),
          types.weights$type), "weight"]
  for (i in 1:(length(listroad)-1)){
    section <- listroad[[i]]
    d <- diag(as.matrix(dist(section))[,2:nrow(section)])
    for (j in 1:(nrow(section)-1)){
      pts <- rbind(pts,
        cbind(seq(section[j,1], section[j+1, 1], length=weights * d[j]/h),
        seq(section[j,2], section[j+1,2], length=weights * d[j]/h)))
  }}
  return(pts)}
```

Références

[1] Ripley, B. 1981. Spatial Statistics, Wiley, New York.
[2] Yamada, I. and Rogerson, P.A. 2003. An empirical comparison of edge effect correction methods applied to K-function analysis. *Geographical Analysis* 35: 97–109.
[3] R Core Team (2013). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.
[4] Tarn Duong (2013). ks: Kernel smoothing. R package version 1.8.12. http://CRAN.R-project.org/package=ks.