Voisinages

Eric Marcon

14 février 2022

Résumé

Manipulation des voisinages dans un jeu de points avec spatstat.

1 Objectif

Obtenir le contenu du voisinage d'arbres cartographiés et calculer des statistiques sur eux, incluant la correction des effets de bord. La package spatstat (Baddeley and Turner, 2005), complété par dbmss (Marcon et al., 2015) fournit les outils nécessaires.

2 Données

2.1 Lecture de la base de Paracou

Une copie des données est stockée localement.

```
load("data/Paracoudb.rda")
```

2.2 Nettoyage

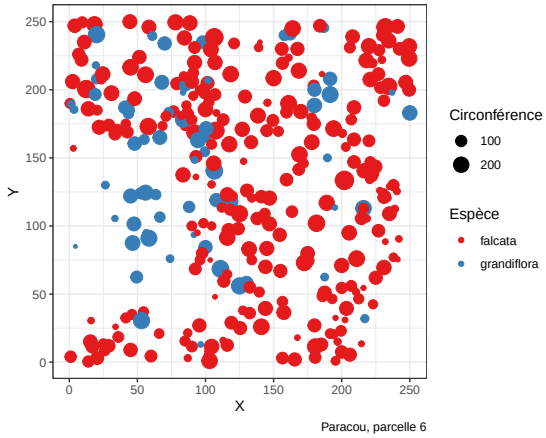
Suppression des colonnes inutiles et ajout d'une colonne avec le nom complet de l'espèce

```
library("tidyverse")
Paracoudb %>%
    as_tibble %>%
    filter(CodeAlive == TRUE) %>%
    select(Plot, SubPlot:Yfield, -Projet, -Protocole,
        Family:Species, CircCorr) %>%
    unite(col = spName, Genus, Species, remove = FALSE) ->
    Paracou
```

2.3 Vérification des données : Carte

Carte des wapas de la P6.

```
Paracou %>%
    filter(Plot == 6 & Genus == "Eperua") %>%
    ggplot() + geom_point(aes(x = Xfield, y = Yfield,
    size = CircCorr, color = Species)) + coord_fixed() +
    scale_color_brewer(palette = "Set1") + labs(x = "X",
    y = "Y", caption = "Paracou, parcelle 6", size = "Circonférence",
    color = "Espèce")
```



3 Utilisation de spatstat

3.1 Création d'un semis de points (planar point pattern)

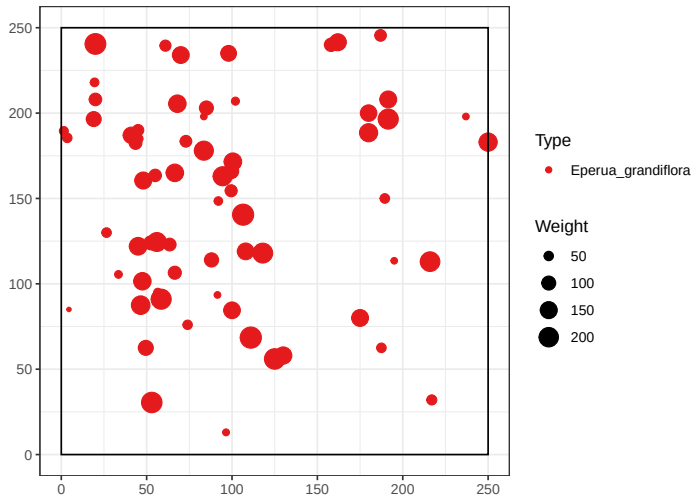
```
library("dbmss")
Paracou %>%
    filter(Plot == 6) %>%
    rename(X = Xfield, Y = Yfield, PointType = spName,
        PointWeight = CircCorr) %>%
    as.data.frame %>%
    wmppp(window = owin(xrange = c(0, 250), yrange = c(0,
        250), unitname = c("meter", "meters"))) ->
    Plot6
```

`Plot6` est un objet *wmppp*.

3.2 Choix des points

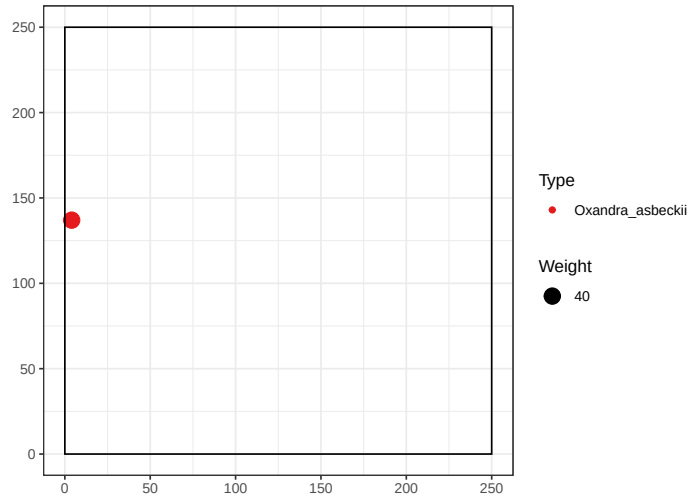
Sélection des Wapa grandiflora.

```
Plot6[Plot6$marks$PointType == "Eperua_grandiflora"] %>%
    autoplot()
```



Sélection du dixième point.

```
Plot6[10] %>%
    autoplot
```



La sélection retourne un *wmppp*.

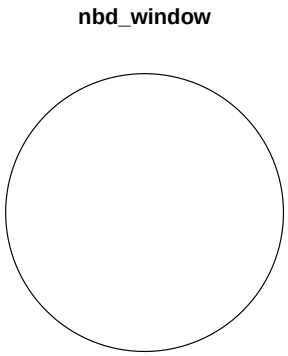
3.3 Voisinage d'un point

Sélection du point

```
Plot6[10] -> point_10
```

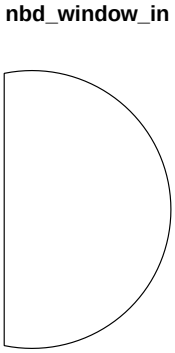
Le voisinage d'un point est le contenu d'une fenêtre circulaire de rayon choisi.

```
r <- 20
nbd_window <- disc(radius = r, centre = c(point_10$x,
    point_10$y))
plot(nbd_window)
```



Le voisinage contenu dans la fenêtre est l'intersection entre les deux fenêtres.

```
nbd_window_in <- intersect.owin(Plot6$window, nbd_window)
plot(nbd_window_in)
```



Sa surface est calculable, comme le facteur de correction pour une correction d'effet de bord par extrapolation.

```
area(nbd_window_in)
```

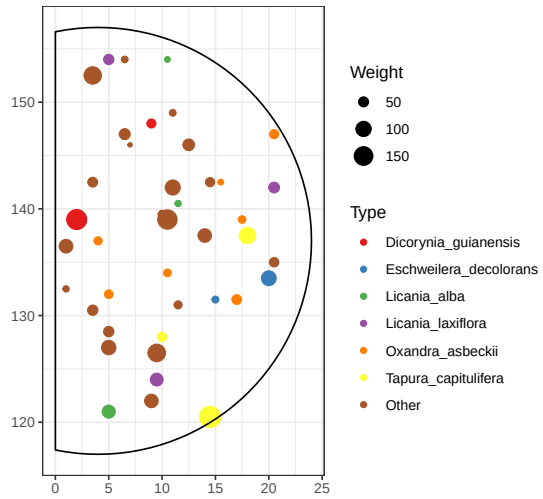
```
## [1] 786.9613
```

```
pi * r^2/area(nbd_window_in)
```

```
## [1] 1.596822
```

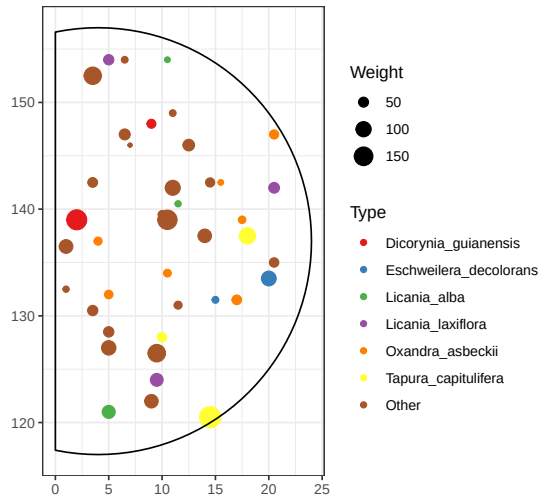
Le voisinage est obtenu par sélection directe.

```
Plot6[nbd_window_in] %>%
    autoplot
```



Une fonction pour extraire le voisinage d'un point :

```
nbd_point <- function(i, Plot, r) {
    nbd_window <- disc(radius = r, centre = c(Plot$x[i],
        Plot$y[i]))
    nbd_window_in <- intersect.owin(Plot$window, nbd_window)
    return(Plot[nbd_window_in])
}
autoplot(nbd_point(10, Plot6, r))
```



4 Calcul de la surface terrière du voisinage

4.1 Exemple pour un point

```
basal_area <- function(neighborhood) {
    return(sum(neighborhood$marks$PointWeight^2 * pi/4))
}
basal_area(Plot6[nbd_window_in])
```

```
## [1] 200425.8
```

Avec correction de l'effet de bord

```
basal_area <- function(neighborhood, r) {
    G <- sum(neighborhood$marks$PointWeight^2 * pi/4)
    Correction <- pi * r^2/area(neighborhood$window)
    return(G * Correction)
}
basal_area(Plot6[nbd_window_in], r)
```

```
## [1] 320044.2
```

4.2 Calcul pour tous les Wapas grandiflora

```
r <- 20
wmppp_wapa <- Plot6[Plot6$marks$PointType == "Eperua_grandiflora"]
(sapply(1:wmppp_wapa$n, function(i) basal_area(nbd_point(i,
    wmppp_wapa, r), r)) -> competition)
```

```
##  [1]  52798.0579  16957.2723  19195.9986
##  [4]  25138.3194  42758.7340  25138.3194
##  [7]  25138.3194  25138.3194  24386.2740
## [10]  42758.7340  46747.4585  28683.2630
## [13]  21047.5103  54030.0421  83335.7990
```

```
## [16]  30004.0477  13764.9763  95173.1155
## [19]  66208.5670 111089.3172  90596.1370
## [22]  50821.5316  93651.9739  22777.4524
## [25]   2083.9091  25403.5551   6761.9323
## [28]   9177.9213  61636.5361  29768.2075
## [31]  31421.0879  92419.9626  75209.6905
## [34]   2083.9091   6082.3897  69040.0704
## [37]  92419.9626   1924.9985   1090.6186
## [40]  10391.0632  83060.9343  90233.9107
## [43]  89008.0010  39423.9688  68680.0173
## [46]   1964.2842   8254.9042  63156.6465
## [49]  51055.8709  51055.8711 101618.1166
## [52] 104072.4899  67961.6789  62014.6121
## [55]  62014.6121  62014.6121  17142.1114
## [58]  89008.0010  20840.4657   1153.8299
## [61]   1214.2752  58057.9545  16978.4866
## [64]  32858.7421   2420.1945   2043.6412
## [67]    962.4992
```

```
# Pour mémoire: facteur de correction par
# extrapolation
sapply(1:wmppp_wapa$n, function(i) pi * r^2/area(nbd_point(i,
    wmppp_wapa, r)))
```

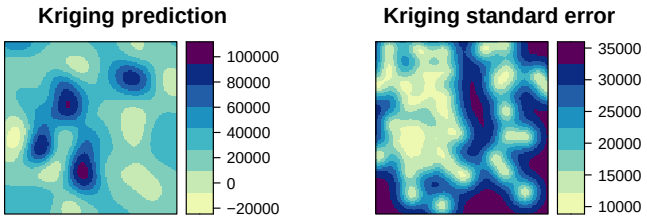
```
##  [1] 1.265279 1.000402 1.007071 1.000402 1.000402
##  [6] 1.000402 1.000402 1.000402 1.222653 1.000402
## [11] 1.000402 1.000402 1.055257 1.000402 1.000402
## [16] 1.000402 1.078115 1.000402 1.000402 1.000402
## [21] 1.000402 1.000402 1.000402 1.637697 1.000402
## [26] 1.826513 1.002744 1.000402 1.000402 1.243424
## [31] 1.312465 1.000402 1.000402 1.000402 1.558139
## [36] 1.000402 1.000402 2.000803 1.133566 1.000402
## [41] 1.000402 1.000402 1.000402 1.000402 1.000402
## [46] 1.000402 1.000402 1.000402 1.000402 1.000402
## [51] 1.000402 1.000402 1.000402 1.000402 1.000402
## [56] 1.000402 1.000402 1.000402 1.000402 1.133566
## [61] 1.558139 1.000402 1.000402 1.000402 1.000402
## [66] 1.000402 1.000402
```

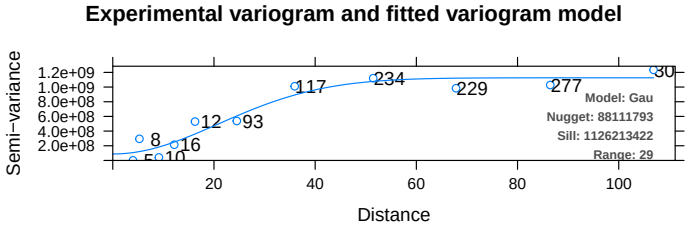
4.3 Carte de la concurrence

```
# Préparation d'une grille de 128 points de côté
xy <- gridcentres(Plot6$window, 128, 128)
# Formatage de la grille
library("sp")
Grille <- SpatialPoints(cbind(xy$x, xy$y))
gridded(Grille) <- TRUE
# Création d'un SpatialPointsDataFrame avec les
# données
sdf_competition <- SpatialPointsDataFrame(coords = data.frame(x = wmppp_wapa$x,
    y = wmppp_wapa$y), data = data.frame(competition))
# Krigeage du SpatialPointsDataFrame
library("automap")
AutoKrige <- autoKrige(formula = competition ~ 1, input_data = sdf_competition,
    new_data = Grille)
```

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

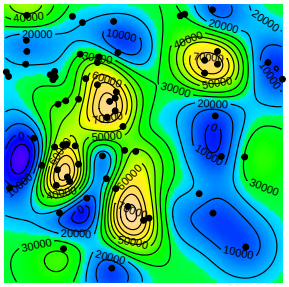
```
# Résultat du krigeage
plot(AutoKrige)
```





Retraitement pour placer les arbres.

```
image(AutoKrige$krige_output, col = topo.colors(128,
    alpha = 1), asp = 1)
contour(AutoKrige$krige_output, add = TRUE)
points(x = wmppp_wapa$x, y = wmppp_wapa$y, pch = 20)
```



Références

Baddeley, A. J. and R. Turner (2005). Spatstat : an R package for analyzing spatial point patterns. *Journal of Statistical Software 12*(6), 1–42.

Marcon, E., S. Traissac, F. Puech, and G. Lang (2015). Tools to Characterize Point Patterns : dbmss for R. *Journal of Statistical Software 67*(3), 1–15.