Simulation d'un jeu de points agrégé et calcul de K()

Processus ponctuels

Les bases

Les processus ponctuels sont l'équivalent des variables aléatoires : ils produisent des jeux de points.

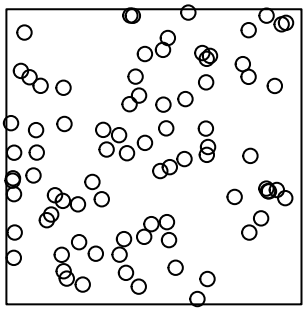
Lis la présentation des principaux dans ma thèse, annexe 1. Les détails techniques sur les différents processus ne sont pas importants, mais il faut bien saisir la distinction entre propriétés de premier et second ordre.

Le processus de base, complètement aléatoire (CSR : *complete spatial randomness*), est le Poisson homogène :

— au premier ordre, les points sont distribués uniformément (l'intensité du processus, toujours notée λ , est la même probabilité partout),
— au second ordre, les points sont indépendants les uns des autres.

On le simule avec `rpoispp()` dans **spatstat**.

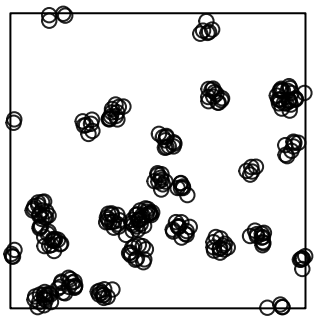
```
library("spatstat")
# 100 points (en espérance) dans une fenêtre carrée de côté 1
X_Poisson <- rpoispp(lambda = 100)
plot(X_Poisson)
```



Processus agrégatifs

Le processus de Matérn place les points uniformément (selon un Poisson homogène) dans des clusters circulaires, eux-mêmes distribués selon un Poisson homogène.

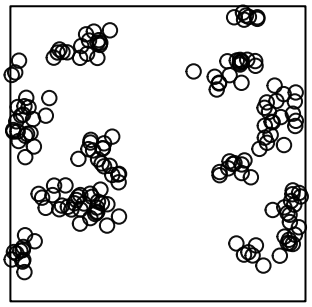
```
# Matérn
X_Matern <- rMatClust(
  # Intensité des centres de cluster: points par unité de surface, ici 2/ha
  kappa = 2 / 10000,
  # Rayon des clusters
  scale = 10,
  # Nombre de points par cluster
  mu = 10,
  # Fenêtre de 9ha
  win = square(300, unitname = c("meter", "meters"))
)
plot(X_Matern)
```



Le processus de Thomas place les points dans les clusters indépendamment mais pas uniformément : l'intensité du processus est une gaussienne bidimensionnelle d'écart-type `scale`.

```
X_Thomas <- rThomas(
  # Intensité des centres de cluster: points par unité de surface, ici 2/ha
  kappa = 2 / 10000,
  # Ecart-type de la gaussienne qui place les points autour des centre des clusters
  scale = 10,
  # Nombre de points par cluster
  mu = 10,
  # Fenêtre de 9ha
  win = square(300, unitname = c("meter", "meters"))
)
plot(X_Thomas)
```

X_Thomas



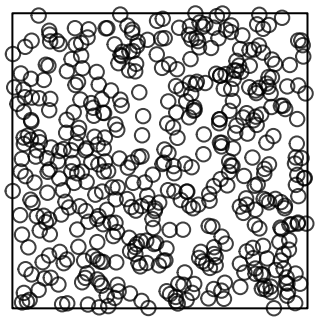
Création d'un ppp

L'objet de base est un ppp (*planar point process*) à créer à partir des coordonnées. Ici, tirage à la main d'un processus de Poisson, 50 points par ha, fenêtre carrée de 9ha.

```
# Nombre de points (aléatoire, tiré dans une loi de Poisson)
n = rpois(1, lambda = 50 * 9)
# Tableau des coordonnées
(xy <- tibble(x = runif(n, min = 0, max = 300), y = runif(n, min = 0, max = 300)))
```

```
# A tibble: 467 x 2
       x     y
   <dbl> <dbl>
 1 169.  285.
 2  19.6 272.
 3 247.  221.
 4 235.   97.0
 5  75.5 163.
 6 255.  260.
 7  35.2 153.
 8 187.  275.
 9  48.2  57.4
10 120.   15.1
# i 457 more rows
```

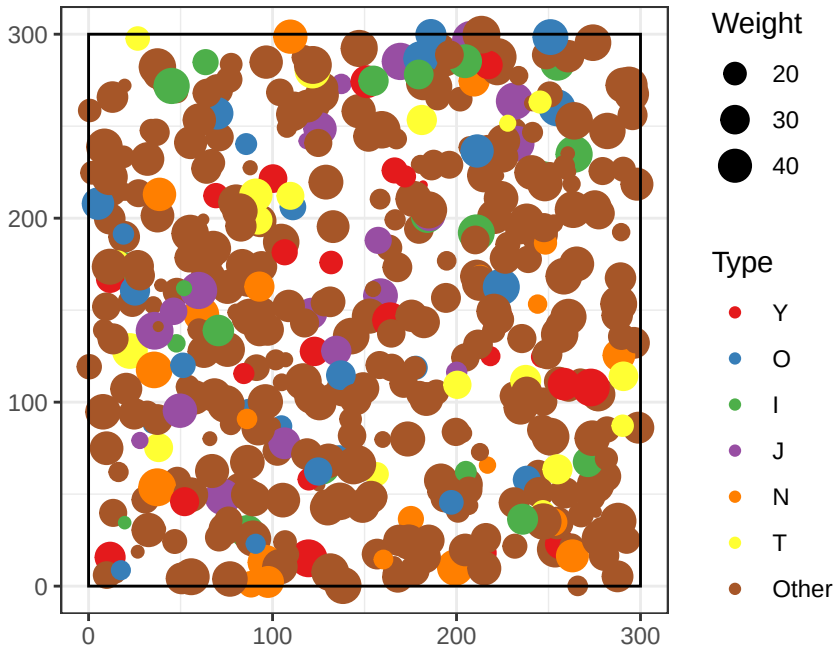
```
# Création d'un ppp
X_Poisson_9ha <- as.ppp(xy, W = square(300, unitname = c("meter", "meters")))
plot(X_Poisson_9ha)
```



dbmss permet de traiter plus facilement des jeux de points marqués, avec un type (par exemple l'espèce) et un poids (par exemple la surface terrière. L'objet de base est un `wmppp` (weighted, marked, planar point pattern) qui est aussi un `ppp`.

Pour le créer, il faut un tableau dont les colonnes s'appellent x, y, PointType et PointWeight.

```
xy_type_poids <- tibble(
  xy,
  PointType = sample(LETTERS, size = n, replace = TRUE),
  PointWeight = runif(n, min = 10, max = 50)
)
library("dbmss")
xy_type_poids %>%
  as.wmppp(window = square(300, unitname = c("meter", "meters"))) %>%
  autoplot()
```



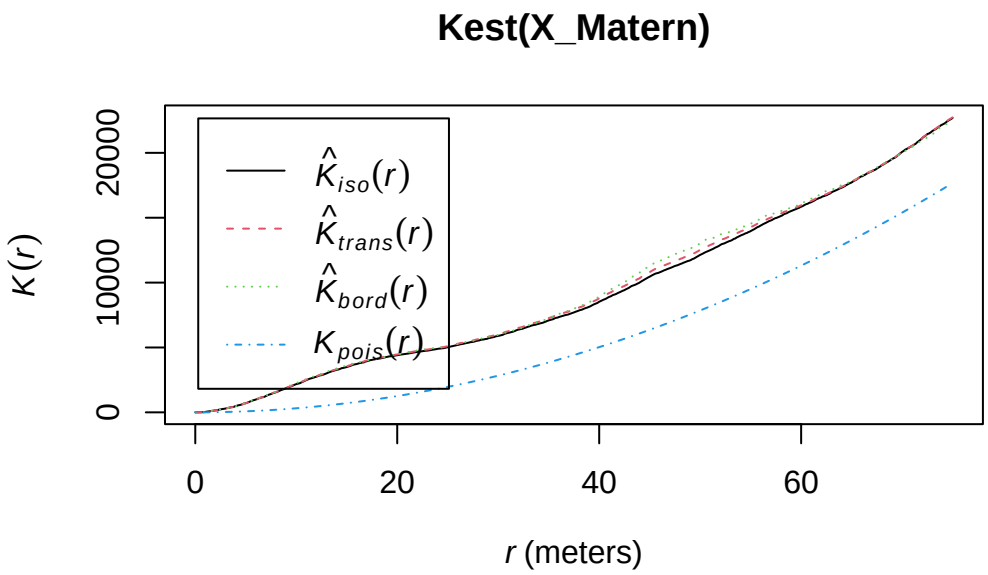
K

La fonction K de Ripley est présentée page 15. La correction des effets de bord est automatique dans **spatstat** donc inutile de s'attarder.

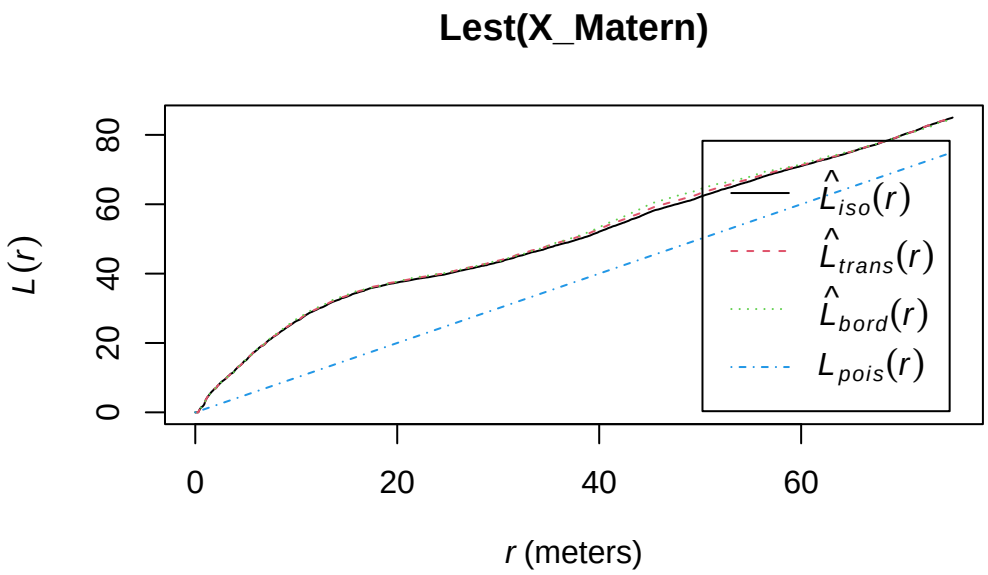
K est croissante : sous l'hypothèse nulle CSR, $K(r) = \pi r^2$. On utilise plutôt $L(r) - r = \sqrt{\frac{K(r)}{\pi}} - r$, qui vaut 0 sous l'hypothèse nulle.

On peut estimer K avec le package **spatstat** ou avec le package **dbmss** plus facile d'accès, et **dbmss** fournit de meilleures figures.

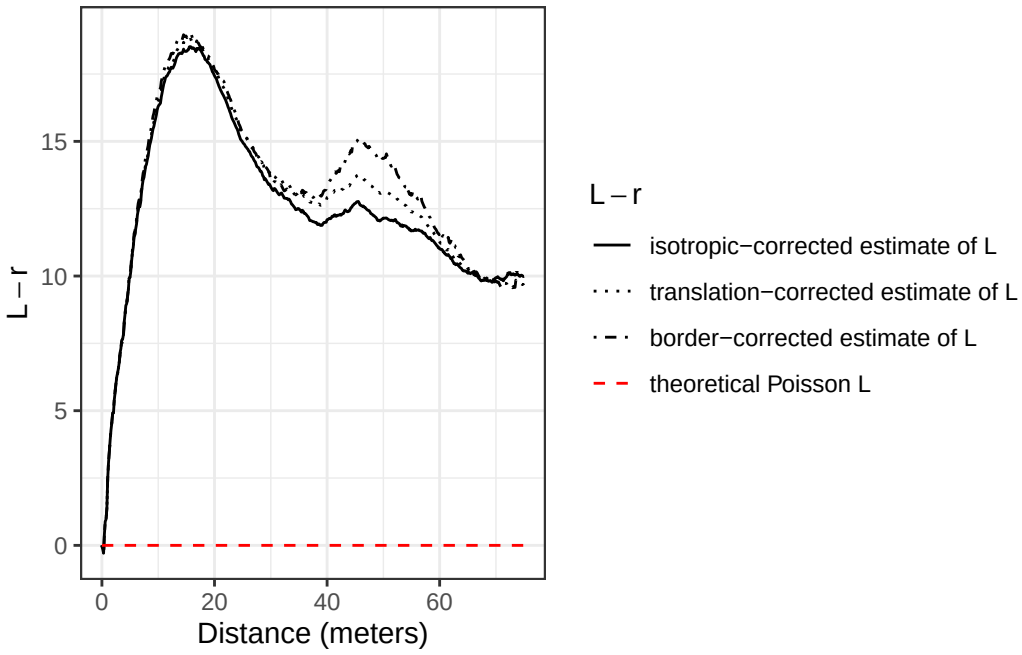
```
# Calcul de K
plot(Kest(X_Matern))
```




```
# Plutôt L
plot(Lest(X_Matern))
```



```
# Avec de beaux graphiques, et surtout L - r
library("dbmss")
autoplot(Lest(X_Matern), fmla = . - r ~ r,)
```

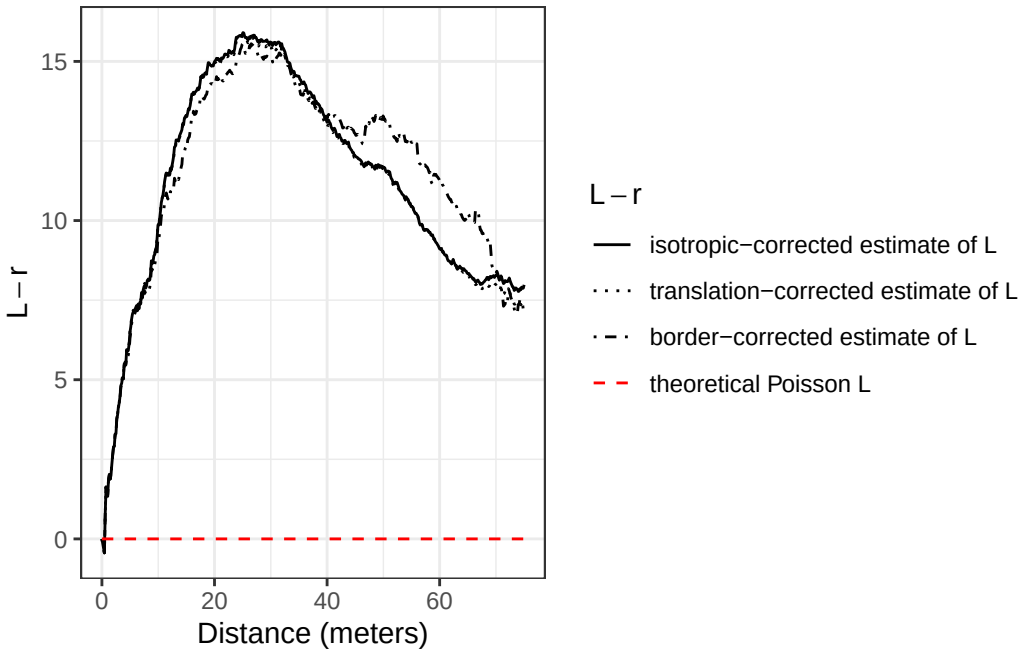


spatstat calcule trois corrections d'effets de bord par défaut.

Le pic de concentration correspond à peu près au diamètre des agrégats selon la culture populaire, plutôt une fois et demie le rayon ici.

Le pic est plus loin pour le processus de Thomas où les arbres peuvent être loin du centre de leur cluster :

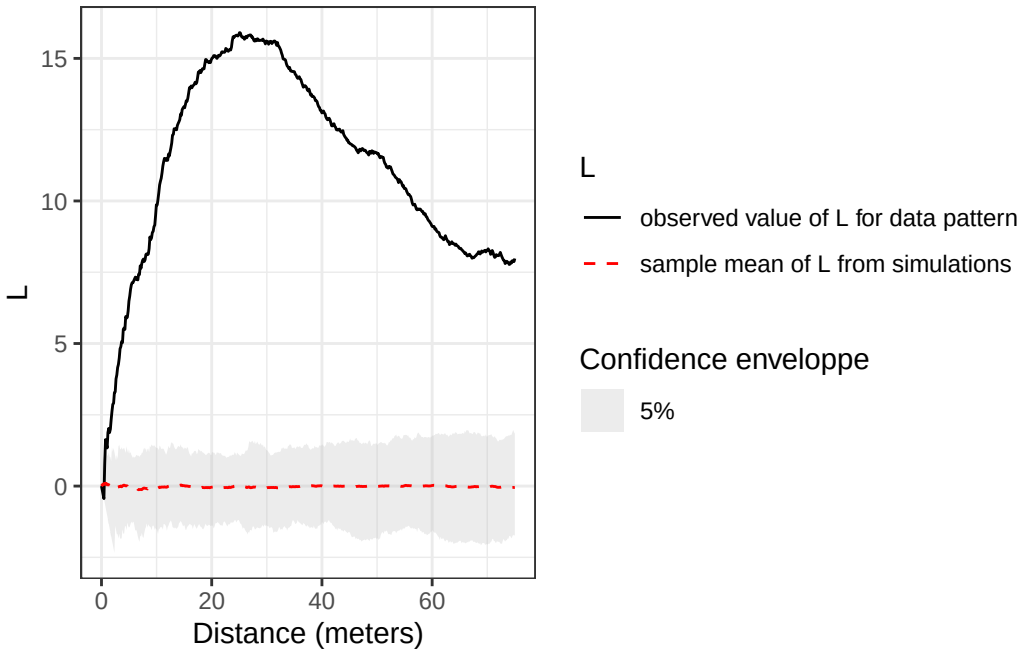
```
autoplot(Lest(X_Thomas), fmla = . - r ~ r)
```



Tests

On peut tester un processus ponctuel contre l'hypothèse nulle CSR par simulations. C'est plus simple avec **dbmss** (attention, la définition de $L(r)$ dans **dbmss** est $L(r) - r$ dans **spatstat**).

```
X_Thomas %>%
  # wmppp obligatoire
  as.wmppp() %>%
  # Calcul de l'intervalle de confiance global avec 100 simulations par défaut
  LEnvelope(Global = TRUE) %>%
  autoplot()
```



Il existe un test exact tant que la fenêtre est rectangulaire.

```
# Produit une p_value. Il faut une quinzaine de valeurs de distances.
Ktest(X_Matern, r = seq(0, 75, by = 5))
```

```
[1] 0
```

```
# à comparer au processus de Poisson
Ktest(X_Poisson, r = seq(0, 0.5, by = .02))
```

```
[1] 0.928985
```

Calcul du nombre local de voisins

```
sapply(
  # Tous les points (vecteur de 1 à n)
  seq_len(X_Thomas$n),
  FUN = \(i) {
    # Pour le point i, cercle de 10m de rayon
    neighborhood <- disc(
      radius = 15,
      centre = c(X_Thomas$x[i], X_Thomas$y[i])
    )
    # Sélection du jeu de points
    X_i <- X_Thomas[neighborhood]
    # Nombre de voisins
```

```
    X_i$n
  }
)
```

```
  [1]  7  2  8  7  2  8  7  8  6  8  8  7  3  9  9
 [16]  8  8  9  3  8  3  5  3  6  6  6  1  6  8  6
 [31]  1  4  2  4  4  3  7  7 10 10  3  9  8  5  8
 [46]  4  8  9  8  8 10  9  2  8  4  5  5  3  8  6
 [61]  3  2  5  4  4  5  1  7  4  6  6  2  8  7  7
 [76]  7  7  8  7  8  7  6  1  1  5 11  8 11  9  9
 [91]  8  9  7  8  4  8  3  3  2  4 10  5  6  6  6
[106]  2  3  7  4  8  7  7  7  5  5  2  7  7  1  3
[121]  6  6  3  5  6  4  4  5  5  6  5  5  6  3  2
[136]  2  4  5  5  5  1  4  9  9  6  6  6  6  6  6
[151]  2  6  6  4  7  5  5  7  5  7  7  5  7  5  6
[166]  9  3  7  1  4  5  7  4  5  5  6
```