Traitement des données de grande taille avec M

Eric Marcon

Florence Puech

12 juillet 2024

Résumé

L'accès grandissant à d'importants jeux de données individuelles et spatialisées couplé au développement de la puissance de calcul ont encouragé la recherche d'outils statistiques adaptés pour analyser au mieux de telles données. Dans un article récent publié dans cette revue, Tidu et al. (ress) soulignent les qualités de la fonction M (Marcon and Puech, 2010), une mesure de concentration spatiale en espace continu. Ils émettent aussi une réserve sur les temps de calculs nécessaires. Notre travail méthodologique cherche à préciser le traitement de jeux de données spatialisées de grande taille avec M sous le logiciel R. Deux voies sont explorées pour déterminer les performances computationnelles de M. Tout d'abord, une évaluation précise des besoins en temps et en mémoire de calcul sur des données géo-localisées est réalisée avec le package dbmss sous R grâce à des tests de performance. Puis, comme suggéré par Tidu et al. (ress), nous considérons également la possibilité d'approximer les positions géographiques des entités analysées. L'importance de la détérioration de l'estimation de M que cette approche crée peut ainsi être estimée, tout comme les gains en temps de calculs permis par l'approximation spatiale des localisations. L'ensemble du code R est donné pour la reproductibilité des résultats.

1 Motivation

L'accès grandissant à d'importants jeux de données individuelles et spatialisées et une plus grande puissance de calcul ont encouragé le développement d'outils d'analyse statistiques permettant de traiter au mieux de telles données (Baddeley et al., 2016). Des études empiriques à des niveaux géographiques très fins ont ainsi pu été proposées au cours des dernières années pour des jeux de données de grande taille. Une attention particulière a porté sur la détection des structures spatiales (attraction, répulsion, indépendance) de données individuelles spatialisées à partir d'analyses ne reposant plus des données zonées mais sur des données géolocalisées. Ce type d'approche a l'avantage de préserver les

positions exactes des entités analysées et par conséquent de ne pas gommer les spécificités individuelles. Diverses études ont montré toute l'importance de retenir une telle méthodologie dans des domaines très variés comme en géographie (Sweeney and Feser, 1998 ; Deurloo and De Vos, 2008), en économie (Arbia, 1989 ; Marcon and Puech, 2003), en écologie (Cressie, 1993 ; Lentz et al., 2011), en biologie (Dray et al., 2021) etc. Dans un article récent, Tidu et al. (ress) soulignent l'intérêt d'une mesure statistique particulière, la fonction M proposée par Marcon and Puech (2010). Cette mesure, que nous dénommerons M dans la suite de l'article, permet de mettre en évidence et les structures spatiales au sein d'une distribution spatialisée (attraction, répulsion, indépendance) à partir d'une étude fondée sur les distances séparant les entités analysées. Toutefois, si cette mesure permet de préserver toute la richesse des données individuelles géolocalisées, elle nécessite un temps de calcul plus long que d'autres mesures fondées sur les distances puisqu'elle est à la fois une mesure cumulative et relative (voir Marcon and Puech (2017) pour une revue de littérature sur les avantages et les limites d'une dizaine mesures existantes fondées sur les distances). Tidu et al. (ress) proposent de limiter les temps de calculs de M en introduisant une erreur de positionnement volontaire des entités analysées. Ainsi, dans leur étude, les établissements industriels en Sardaigne (Italie) ne sont plus localisés à leur adresse postale exacte mais au centroïde de leur municipalité. Ce repositionnement permet de réduire les temps de calcul car le nombre de distances possibles entre les établissements est, de fait, limité aux distances séparant les centroïdes des communes. Cette approche est analogue à celle de Scholl and Brenner (2015) qui ont proposé, pour la fonction K_d (Duranton and Overman, 2005) qui permet de caractériser les structures spatiales par une autre méthode, d'approximer les distances entre paires d'entités en les regroupant par classes. La méthode de Scholl and Brenner (2015), implémentée dans le package *dbmss* (Marcon et al., 2015) pour R (R Core Team, 2024) apporte un gain de performance computationnelle considérable avec une faible perte de précision.

Dans notre article, nous proposons de tester l'efficacité de la méthode de Tidu et al. (ress). Dans un premier temps, nous montrons les avantages de l'utilisation du package *dbmss* pour estimer la fonction M sur des jeux de données dont l'ordre de grandeur est de 100 000 points ou moins et nous montrons que les temps de calcul deviennent excessifs au-delà, sur un ordinateur personnel. Nous étudions ensuite l'effet de l'approximation géographique des localisations des entités analysées. Ce travail méthodologique reposant sur un nombre d'emplacements des entités volontairement limité permet de quantifier l'importance de la détérioration de l'information que cette approche crée. Ces tests de performance permettent d'apporter une réponse précise sur les avantages et les limites computationnelles de la fonction M en fonction de la taille des jeux de données

Le plan de l'article est le suivant. Une première section génère les données nécessaires. Des jeux de points de grande taille (de l'ordre de plusieurs dizaines de milliers de points) complètement aléatoires ou (géographiquement) concentrés sont tirés. La deuxième section détaille l'utilisation du package *dbmss* pour calculer la fonction M et son intervalle de confiance à partir d'un tableau donnant la position et les caractéristiques des points ou bien une matrice de distances

entre eux. La troisième section mesure la performance de *dbmss* en fonction de la taille du jeu de points, en termes de temps de calcul et de mémoire nécessaire. Enfin, la dernière section teste l'approximation qui consiste à les rassembler au centre des cases d'une grille, selon l'approche de Tidu et al. (ress) qui les positionnent au centre des unités administratives dans lesquelles ils se trouvent.

2 Simulation des données

Les jeux de données que nous allons considérer dans cet article sont obtenus par simulation. Le code R est donné en annexe, ce qui permet une parfaite reproductibilité les exemples traités ou d'en développer d'autres.

2.1 Tirage des points

Un jeu de points est tiré par un processus binomial dans une fenêtre carrée de côté 1. On associe à chaque point une marque qualitative : "Cas" ou "Contrôle". La majorité des points sont des "Contrôles" et une partie constitue des "Cas", dont la structure spatiale est étudiée. Le poids des points est tiré dans une loi gamma dont les paramètres de forme et d'échelle sont libres.

Dans cet exemple, le tirage des points est complètement aléatoire (*complete spatial randomness* : CSR), c'est-à-dire qu'il n'y a pas ici de simulation d'attraction ou de dispersion de points qui pourraient générer des concentrations spatiales de points (agrégats) ou, au contraire, des régularités spatiales (dispersions).

Des jeux de points agrégés peuvent être tirés dans un processus de Matérn (1960).

Les Cas apparaissent dans la figure 1 : les agrégats sont nettement visibles. Les contrôles (non représentés) sont distribués complètement aléatoirement.

2.2 Maillage de l'espace

Considérons la simulation des Cas obtenus par le processus de Matérn et découpons la fenêtre en une grille. Un maillage est de l'espace précédemment considéré est obtenu, qui permet de simuler l'approximation habituelle de la position des points d'une unité administrative sur la position de son centre.

La position recentrée est présentée sur la carte de la figure 2. Chaque cellule ne contient plus qu'un seul point de chaque type dont le poids est la somme des poids des points individuels.

Les valeurs de la fonction M peuvent maintenant être calculées à partir du jeu de point original ou de son approximation après recentrage.

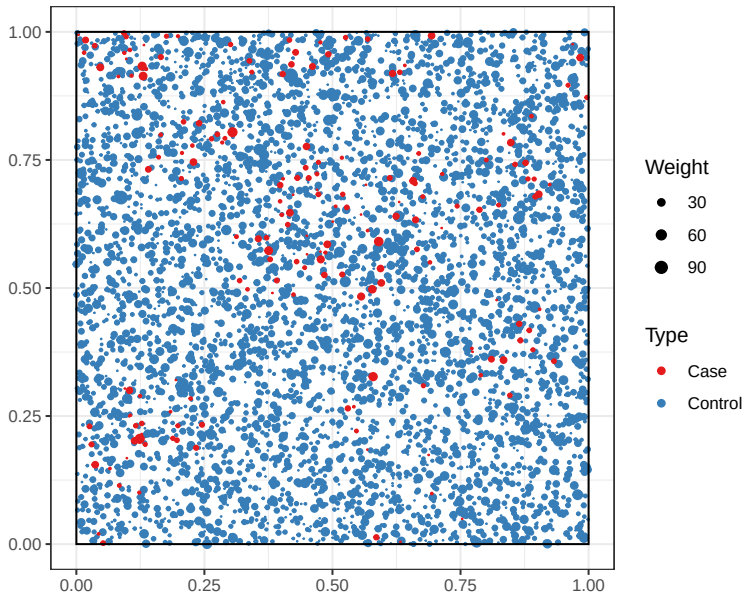


FIG. 1 : Tirage d'un jeu de points dont les cas (en rouge) sont agrégés alors que les contrôles (en bleu) sont distribués complètement aléatoirement. La taille des points est proportionnelle à leur poids.

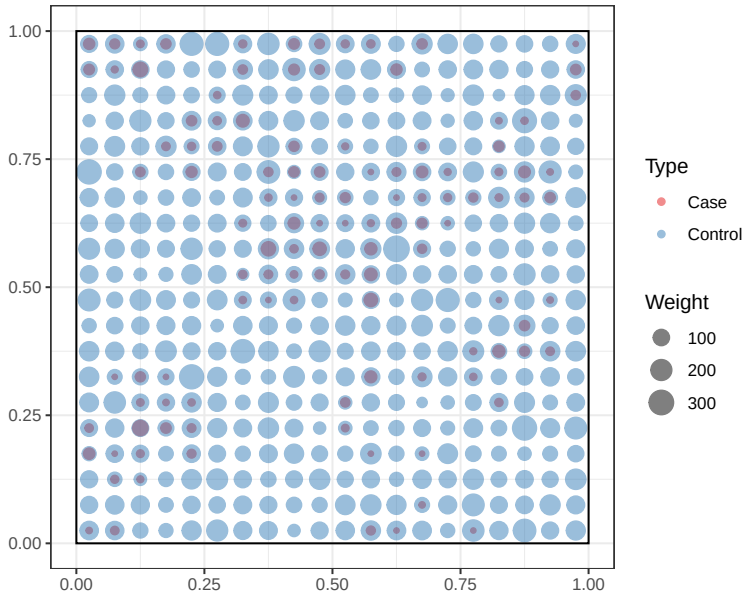


FIG. 2 : Repositionnement des points dans une grille arbitraire. L'absence de Cas dans une cellule est aisément détectable (point unicolore bleu), tout comme la forte présence de Cas dans une cellule (point bicolore mais à dominante rouge).

3 Calcul de M avec le package *dbmss*

3.1 Données nécessaires

Dans le package *dbmss*, la fonction s'applique à un jeu de points ou à une matrice de distances. Le jeu de points de la figure 1 est utilisé. La matrice de distances entre toutes les paires de ses points est calculée pour constituer les données sur lesquelles les tests de performance seront réalisés.

3.2 Jeu de points

La fonction `Mhat()` du package *dbmss* permet d'estimer la fonction M. La valeur de référence théorique de M est 1 car cette fonction rapporte la proportion de Cas jusqu'à une distance r à celle observée sur toute la fenêtre. L'agrégation des Cas sera mise en évidence par des valeurs de M supérieures à 1 (la présence relative de Cas est plus importante localement que sur l'ensemble de la fenêtre) et la dispersion des Cas par des valeurs inférieures à 1. On observe (figure 3) que M détecte une agglomération des Cas, ce qui est en accord avec la simulation de ce type de points (les contrôles ayant une localisation complètement aléatoire sur la fenêtre). L'avantage d'une fonction fondée sur les distances est nettement visible : elle permet de détecter exactement à quelle(s) distance(s) les phénomènes d'attraction se produisent et sont les plus importants (pour les fonctions dont les valeurs peuvent être comparées à différents rayons, comme M).

La fonction `Menvelope()` permet de calculer en plus de l'estimation de la fonction M son intervalle de confiance global (Duranton and Overman, 2005) sous l'hypothèse nulle de localisation aléatoire des points. Son résultat est représenté sur la figure 3.

3.3 Matrice de distances

Les matrices permettent de traiter des distances non euclidiennes (temps de transport, distance routière…) qui ne peuvent pas être représentées par un jeu de points.

Les fonctions `Mhat()` et `MEnvelope()` sont les mêmes, et fournissent les mêmes résultats quelle que soit la forme des données utilisées ici (jeu de point ou matrice de distances).

4 Performance du calcul

L'utilisation de la fonction M pour caractériser la structure spatiale de grands jeux de points peut être limitée par le temps de calcul ou la mémoire nécessaire.

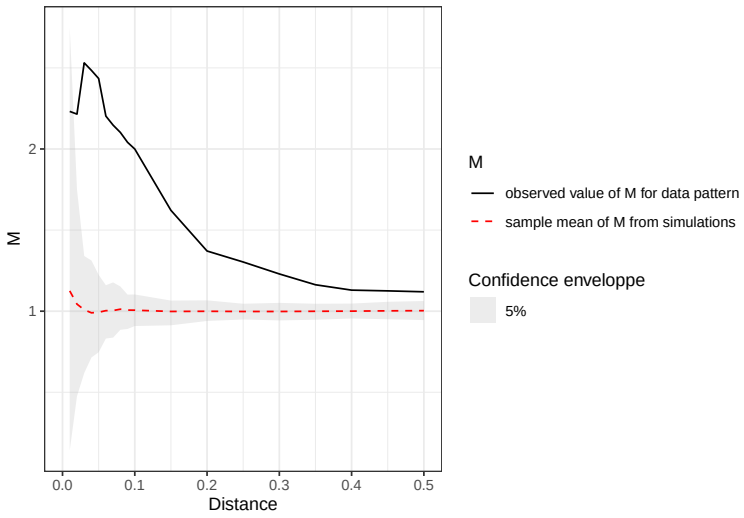


FIG. 3 : Valeur de M en fonction de la distance au point de référence. L'intervalle de confiance, simulé à 95%, apparaît en gris et est centré sur la valeur 1.

4.1 Temps de calcul

Le calcul des distances entre toutes les paires de points est nécessaire pour estimer M. On s'attend donc à ce que le temps de calcul augmente comme le carré du nombre de points.

Le temps de calcul nécessaire au calcul exact est évalué pour une gamme de nombres de points (figure 4).

Le temps de calcul est lié à la taille du jeu de points par une loi puissance.

Il augmente moins vite que le carré du nombre de points. Il peut être estimé très précisément ($R^2 = 0.97$) par la relation $t = t_0(n/n_o)^p$ où t est le temps estimé pour n points connaissant le temps t_0 (ex. : 2.64 secondes) pour n_0 points (ex. : 100000) et p la relation de puissance (1.52).

L'utilisation d'une matrice de distances peut sembler efficace pour en économiser le temps de calcul mais le calcul des distances est en réalité extrêmement rapide et le traitement complet à partir d'une matrice est finalement plus long (tableau 1).

4.2 Mémoire

La mémoire utilisée est évaluée pour les mêmes tailles de données (figure 5).

La mémoire nécessaire augmente linéairement avec le nombre de points et n'est jamais critique pour des tailles de jeux de points traitables dans des temps raisonnables. Cela permet de relativiser la conclusion de Tidu et al. (ress) sur la puissance et le temps de calculs nécessaires en utilisant M sur des jeux de données de taille importante.

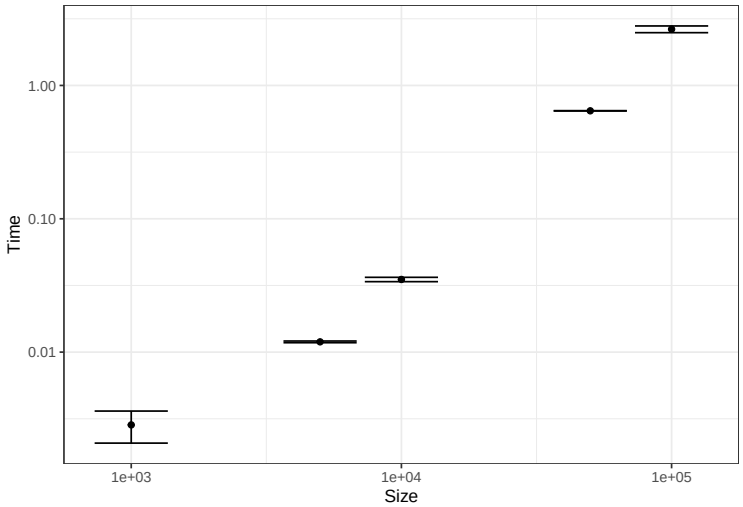


FIG. 4 : Temps de calcul (secondes) de l'estimation de la fonction M en fonction de la taille du jeu de points. Les barres représentent l'intervalle de ± 1 écart-type.

TAB. 1 : Temps d'exécution médian, en millisecondes, de l'estimation de la fonction M à partir d'un jeu de 5000 points ou de la matrice de distances correspondante.

Méthode	Temps
Jeu de points	10
Matrice de distances	13

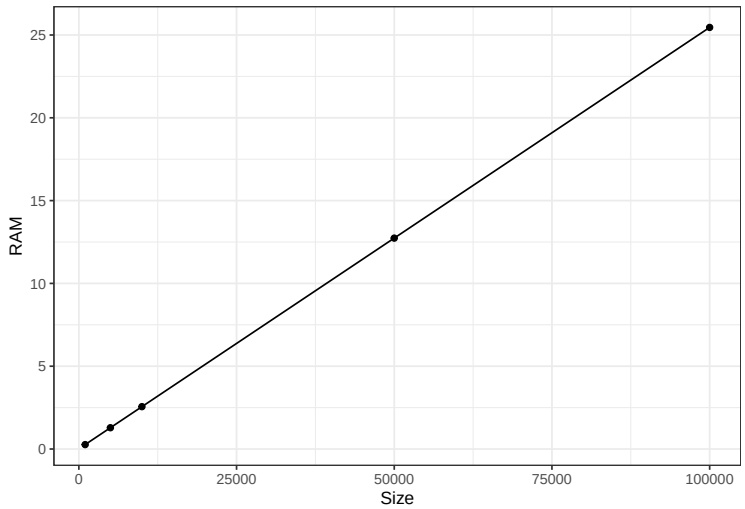


FIG. 5 : Mémoire nécessaire (Mo) pour l'estimation de la fonction M en fonction de la taille du jeu de points.

La mémoire utilisée par les objets `Dtable` pour le calcul de M à partir d'une matrice de distance est bien supérieure : c'est celle d'une matrice numérique, de l'ordre de 8 octets fois le nombre de points au carré, soit 800 Mo pour 10000 points seulement. Comme le temps de calcul n'est pas réduit par cette approche, son utilisation est à réserver aux distances non-euclidiennes.

5 Effets de l'approximation de la position des points

Il est évident que l'approximation de la position des points entraîne une perte d'information : dans chaque cellule de la grille, la distance entre tous les points est fixée à 0, et la distance entre deux points de cellules différentes est approchée par celle entre les centroïdes des deux cellules. On s'attend donc à une sévère erreur dans l'estimation de M à petite échelle (de l'ordre de grandeur de la taille des cellules) et une erreur diminuant avec la distance, quand la taille relative des cellules diminue.

L'effet de l'approximation de la localisation est testé d'abord sur un jeu de points agrégés, similaire aux données réelles de Tidu et al. (ress). Dans un deuxième temps, le cas d'un jeu de points non structuré est traité.

5.1 Cas d'une distribution agrégée (Matérn)

100 jeux de points agrégés sont simulés. Pour évaluer l'effet de l'approximation de la position, le calcul exact et le calcul sur les points de la grille sont effectués sur chaque jeu de points.

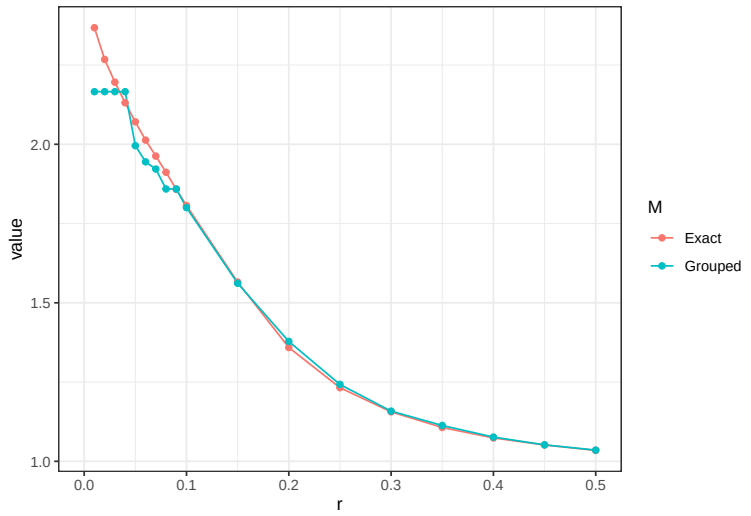


FIG. 6 : Estimation moyenne de la fonction M à partir de la position exacte des points comparée aux valeurs obtenues en regroupant les points.

Les valeurs moyennes des estimations de M sont présentées dans la figure 6.

La taille des cellules de la grille est égale à 0.05. Tous les voisins à distance inférieure à ce seuil sont placés à distance nulle : l'estimation de la fonction est constante jusqu'à ce seuil.

La corrélation entre les valeurs de M estimées par chaque méthode est calculée à chaque distance (figure 7).

La corrélation est très proche de 1, et les valeurs estimées très similaires, dès que la distance prise en compte dépasse la maille de la grille : l'approximation n'est pas un problème si les interactions entre les points sont étudiées au-delà de cette distance. L'information sur les interactions à courte distance, c'est-à-dire à l'intérieur de chaque cellule de la grille, est perdue, ou, plus précisément, approximée par sa valeur à l'échelle de la grille.

5.2 Cas d'une distribution complètement aléatoire (CSR)

Les mêmes simulations sont effectuées avec un jeu de points complètement aléatoire. Le calcul exact et le calcul sur les points de la grille sont effectués sur chaque jeu de points.

Les valeurs moyennes sont présentées dans la figure 8.

La valeur moyenne de M est égale à 1 à toutes les distances par construction : les cas et les contrôles sont distribués complètement aléatoirement. Les approximations sont relativement faibles en valeur (quelques pourcents) mais comme la valeur réelle de M varie peu autour de 1, les corrélations sont bien plus faibles (figure 9) en absence de structure spatiale.

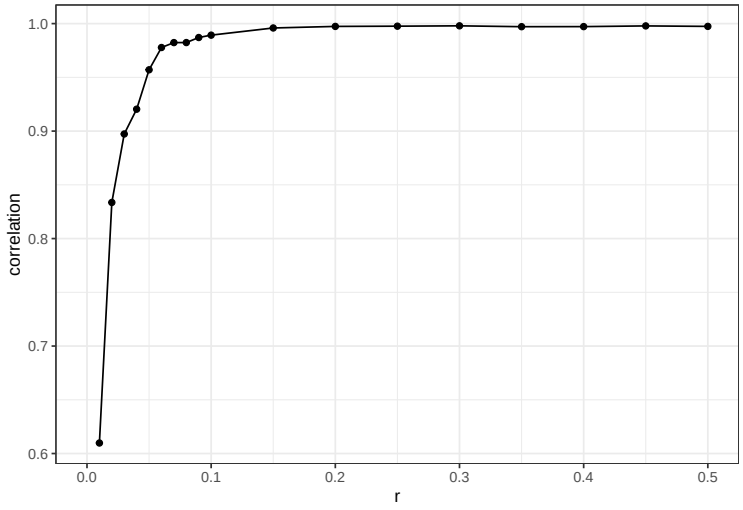


FIG. 7 : Corrélation entre les valeurs de M estimées à partir de la position exacte des points et en regroupant les points.

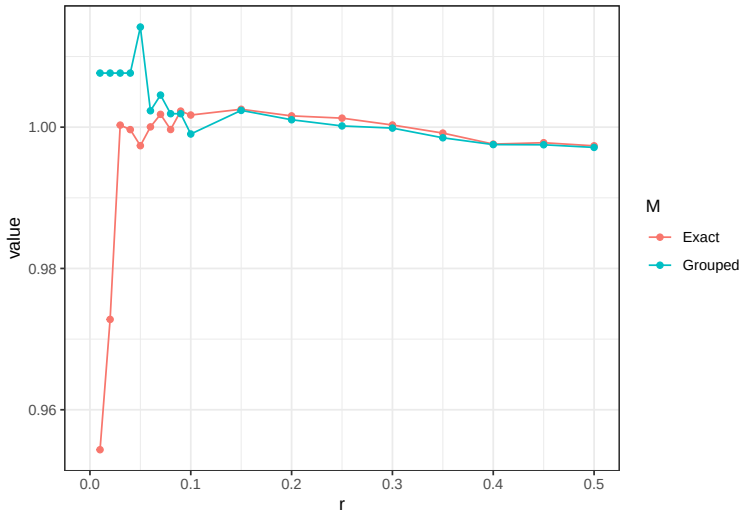


FIG. 8 : Estimation moyenne de la fonction M à partir de la position exacte des points comparée aux valeurs obtenues en regroupant les points.

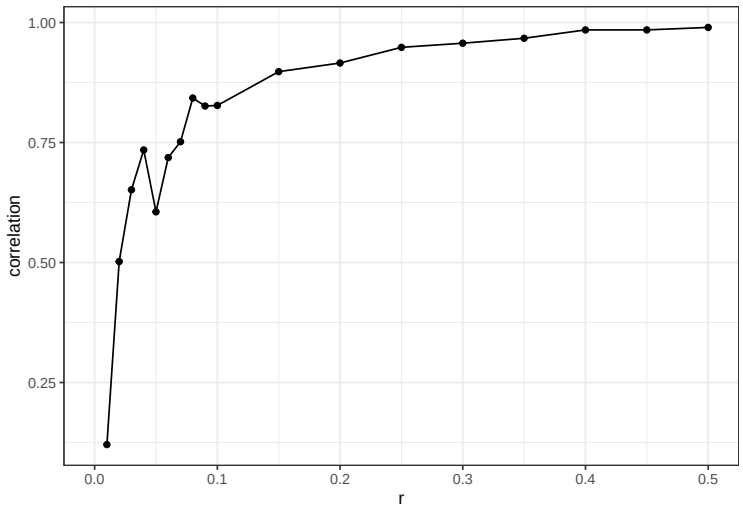


FIG. 9 : Corrélation entre les valeurs de M estimées à partir de la position exacte des points et en regroupant les points.

6 Conclusion

Pour résumé et d'après les cas proposés dans cet article, il semble que l'approximation sur la localisation peut être considérée pour gagner en temps de calculs, étant donné les fortes corrélation observées entre les valeurs de M sur données exactes et approximées mais en conservant un maillage très fin. Notre résultat va donc dans le sens de l'article de Tidu et al. (ress) qui mentionne de fortes corrélations sur des données de localisations d'entreprises italiennes. La structure spatiale de leurs données devant être probablement un cas intermédiaire entre les deux cas traités dans notre article (distributions théoriques agrégée et aléatoire), les résultats apportés par nos deux contributions sont donc complémentaires. En revanche, si l'objet de l'étude est de s'intéresser aux structures spatiales à très petites distances, l'approximation des positions géographiques n'est alors pas souhaitable car ce sont pour ces distances que les écarts entre les résultats de M sont les plus grands.

Le temps de calcul de M est de l'ordre de 6 secondes pour un jeu de 100 000 points sur un ordinateur portable (processeur Intel i7-1360P 2.20 GHz), et nécessite 25 Mo de RAM. Le calcul d'un intervalle de confiance à partir de 1000 simulations prend donc moins de deux heures.

Pour un jeu de cinq millions de points, le temps de calcul attendu est $6 \times 50^{1.8} = 6860$ secondes, près de deux heures. 1000 simulations nécessiteraient alors environ trois mois. Le calcul des distances est parallélisé : un serveur de calcul augmenterait drastiquement la performance mais au prix d'une complexité de mise en oeuvre qui limite son usage.

En se limitant à la puissance de calcul d'un ordinateur personnel, le calcul exact se justifie donc pleinement pour des données de l'ordre de 10^5 points :

quelques heures suffisent à calculer des intervalles de confiance.

Au-delà, l'approximation de la localisation permet de ramener la taille du jeu de points à celle du nombre de localisations retenues. Le prix à payer est l'absence d'information à l'échelle des unités géographiques élémentaires (les cellules de la grille ici). Selon les questions traitées, cette limite peut être acceptable ou non : la description globale de la structure spatiale est assez peu dégradée, mais l'étude des externalités, particulièrement intéressante à petite distance, est très limitée.

7 Annexe

Le code utilisé dans cet article est détaillé ici.

7.1 Simulation des données

7.1.1 Tirage des points

Un jeu de points est tiré aléatoirement avec les paramètres suivants :

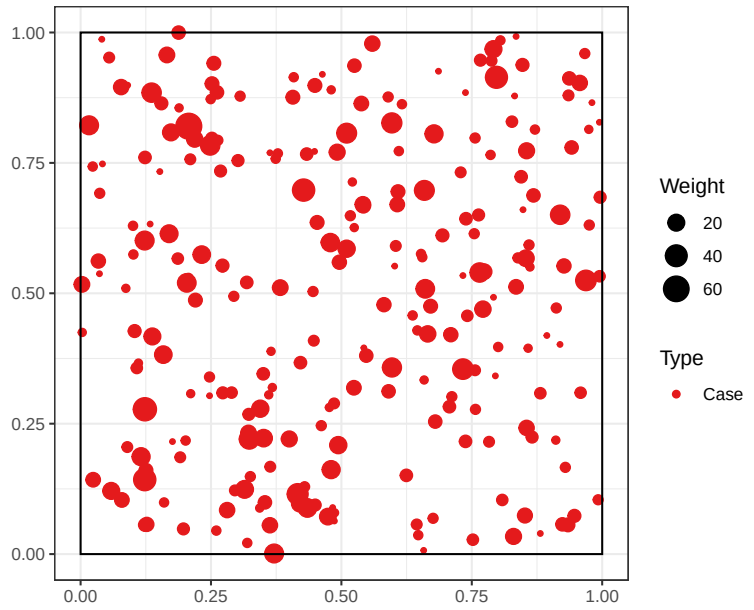
- le nombre de points,
- la proportion de contrôles,
- la forme et l'échelle de la loi gamma.

```
library("tidyverse")
library("spatstat")
library("dbmss")

par_points_nb <- 5000
par_case_ratio <- 1/20
par_size_gamma_shape <- 0.95
par_size_gamma_scale  <- 10
```

La fonction `X_csr()` permet de tirer un semis de points selon des paramètres déterminés. L'argument `points_nb` qui fixe le nombre de points peut être modifié ; les autres paramètres ont leur valeur fixée plus haut.

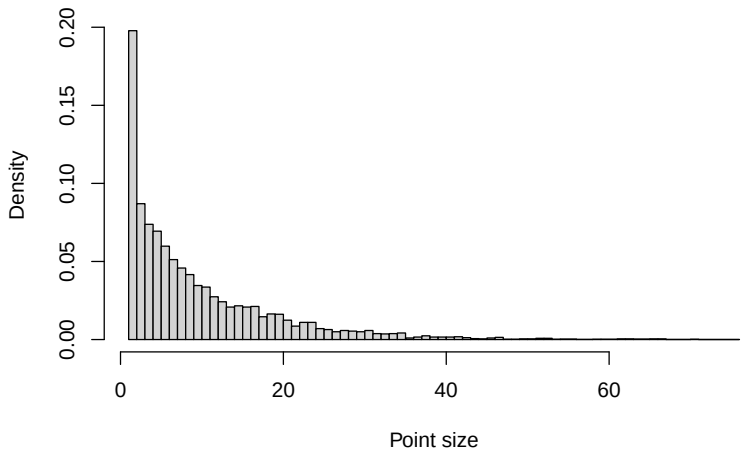
```
X_csr <- function(
    points_nb,
    case_ratio = par_case_ratio,
    size_gamma_shape = par_size_gamma_shape,
    size_gamma_scale = par_size_gamma_scale) {
  points_nb %>%
    runifpoint() %>%
    as.wmppp() ->
    X
  cases_nb <- round(points_nb *  case_ratio)
  controls_nb <- points_nb - cases_nb
  c(rep("Control", controls_nb), rep("Case", cases_nb)) %>%
    as.factor() ->
    X$marks$PointType
  rgamma(
    X$n,
    shape = size_gamma_shape,
    scale = size_gamma_scale
  ) %>%
    ceiling() ->
```

```
    X$marks$PointWeight
  X
}

# Example
X <- X_csr(par_points_nb)
# Map the cases
autoplot(X[X$marks$PointType == "Case"])
```



La distribution des tailles est donnée par l'histogramme suivant :

```
# Point size distribution
hist(
  X$marks$PointWeight,
  breaks = unique(X$marks$PointWeight),
  main = "",
  xlab = "Point size"
)
```



La fonction `X_matern()` permet de tirer un semis de points dont les Cas sont concentrés par un processus de Matérn (1960). Les paramètres sont :

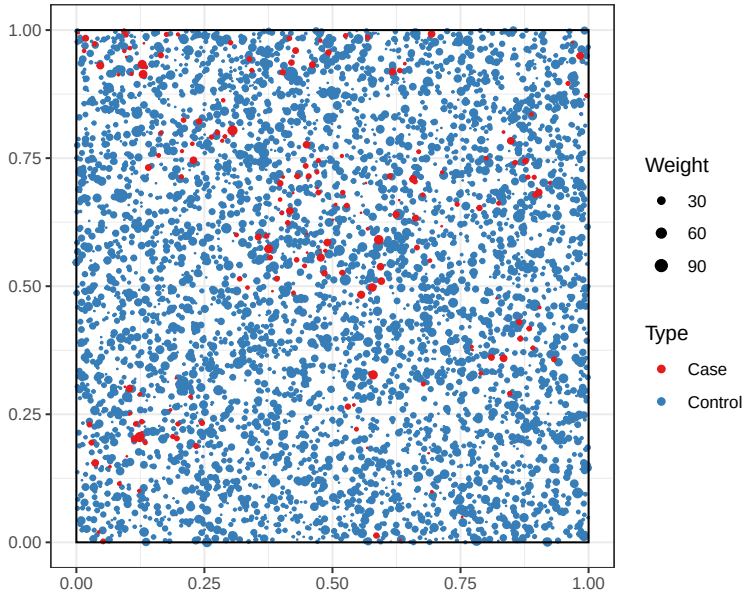
- κ : le nombre d'agrégats attendu,
- `scale` : leur rayon.

```
# Expected number of clusters
par_kappa <- 20
# Cluster radius
par_scale <-  0.1
```

Le code de la fonction est le suivant :

```
X_matern <- function(
    points_nb,
    case_ratio = par_case_ratio,
    kappa = par_kappa,
    scale = par_scale,
    size_gamma_shape = par_size_gamma_shape,
    size_gamma_scale = par_size_gamma_scale) {
  cases_nb <- round(points_nb *  case_ratio)
  controls_nb <- points_nb - cases_nb
  # CSR controls
  controls_nb %>%
    runifpoint() %>%
    superimpose(
      # Matern cases
      rMatClust(
        kappa = kappa,
        scale = scale,
        mu = cases_nb / kappa
      )
    ) %>%
    as.wmppp() ->
    X
  # Update the number of cases
  cases_nb <- X$n - controls_nb
  c(rep("Control", controls_nb), rep("Case", cases_nb)) %>%
    as.factor() ->
    X$marks$PointType
  rgamma(
    X$n,
    shape = size_gamma_shape,
    scale = size_gamma_scale
```

```
  ) %>%
    ceiling() ->
    X$marks$PointWeight
  X
}

# Example
X <- X_matern(par_points_nb)
# Map the cases
autoplot(X) +
  scale_size(range = c(0, 3))
```



7.1.2 Maillage de l'espace

Le nombre de lignes et de colonnes est paramétré :

```
# Number of rows and columns
par_partitions <- 20
```

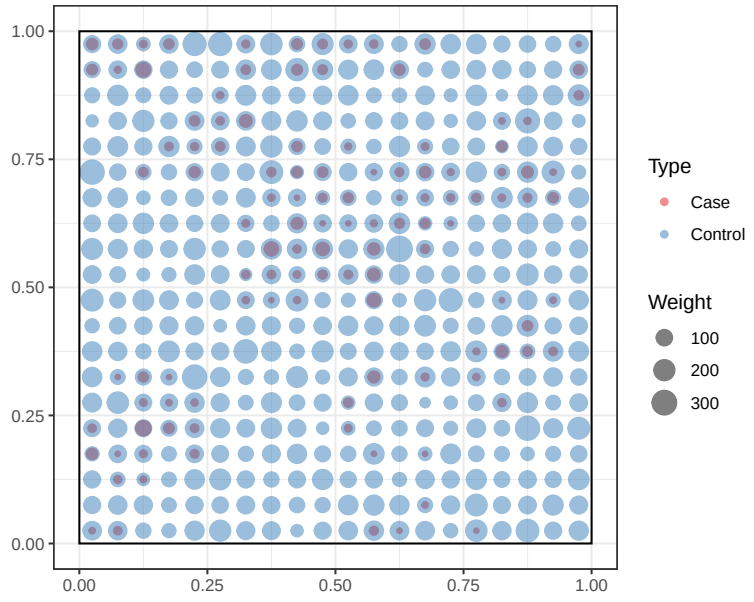
La fonction `group_points()` rassemble au centre de chaque cellule de la grille tous les points qu'elle contient. Cela permet de simuler l'approximation habituelle de la position des points d'une unité administrative sur la position de son centre. La position des points est légèrement bruitée pour permettre le calcul de M. La fonction `group_points_to_plot()` fusionne les points pour produire une carte.

```
# Group points into cells
group_points <- function(X, partitions = par_partitions) {
  X %>%
    with(tibble(
      x,
      y,
      PointType = marks$PointType,
```

```
      PointWeight = marks$PointWeight)
    ) %>%
    mutate(
      x_cell = ceiling(x * partitions) / partitions - 1 / 2 / partitions,
      y_cell = ceiling(y * partitions) / partitions - 1 / 2 / partitions,
      .keep = "unused"
    ) %>%
    rename(x = x_cell, y = y_cell) |>
    as.wmppp(window = X$window, unitname = X$window$units) |>
    rjitter()
}
# Group points and merge them
group_points_to_plot <- function(X, partitions = par_partitions) {
  X %>%
    with(tibble(
      x,
      y,
      PointType = marks$PointType,
      PointWeight = marks$PointWeight)
    ) %>%
    mutate(
      x_cell = ceiling(x * partitions) / partitions - 1 / 2 / partitions,
      y_cell = ceiling(y * partitions) / partitions - 1 / 2 / partitions
    ) %>%
    group_by(PointType, x_cell, y_cell) %>%
    summarise(n = n(), PointWeight = sum(PointWeight)) %>%
    rename(x = x_cell, y = y_cell) |>
    as.wmppp(window = X$window, unitname = X$window$units)
}
```

La figure est obtenue par le code suivant :

```
X |> group_points_to_plot() |> autoplot(alpha = 0.5)
```



7.2 Calcul de M

Les distances auxquelles la fonction M est calculées sont choisies dans `r`.

```r
r <- c((0:9) / 100, (2:10) / 20)
```

7.2.1 Données nécessaires

Dans le package *dbmss*, la fonction s'applique à un jeu de points, objet de classe `wmppp`, ou à une matrice de distances, objet de classe `Dtable`.

Nous partons d'un tableau (data.frame) contenant les colonnes `x`, `y`, `PointType` et `PointWeight`.

Les coordonnées spatiales des points sont données par les colonnes `x` et `y`.

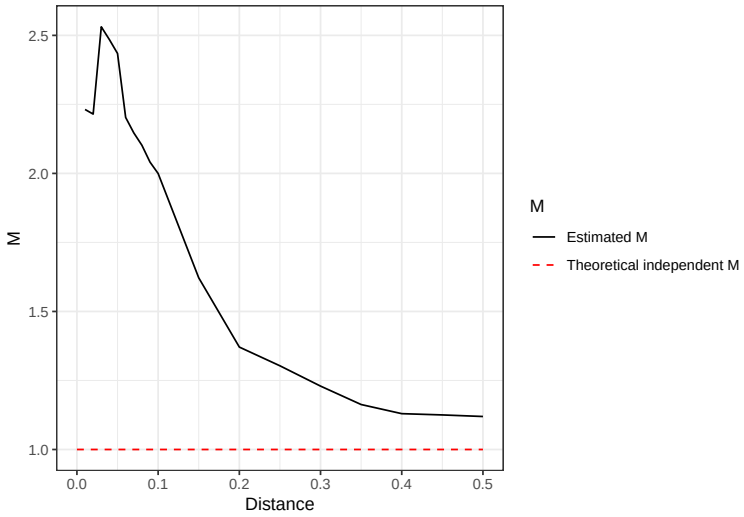
```r
# Extract a dataframe from the point set
points_df <- with(X, data.frame(x, y, marks))
head(points_df)
```

```
##           x          y PointWeight PointType
## 1 0.4550716 0.31775637           4   Control
## 2 0.6463730 0.04396279           2   Control
## 3 0.9156488 0.23361975           2   Control
## 4 0.9724551 0.87464816           1   Control
## 5 0.8907927 0.26205266           4   Control
## 6 0.9561687 0.96813173           7   Control
```

7.2.2 Jeu de points

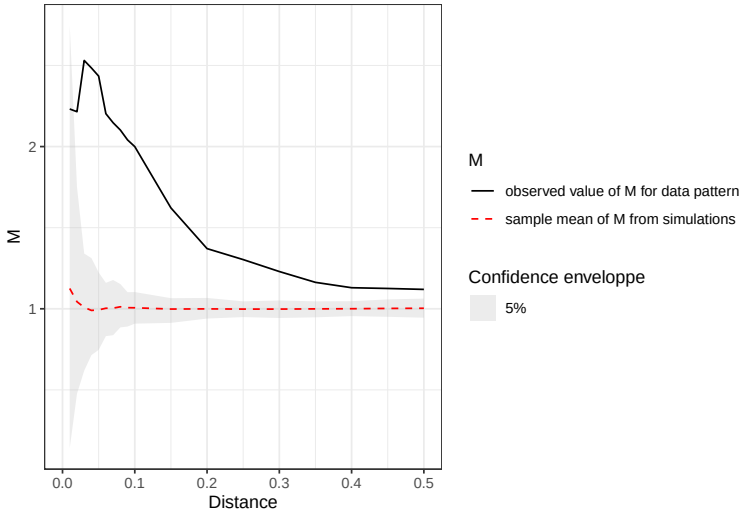
La fonction `Mhat()` permet d'estimer la valeur de la fonction M.

```r
X %>%
  Mhat(r = r, ReferenceType = "Case") %>%
  autoplot()
```



La fonction `Menvelope()` permet de calculer l'intervalle de confiance de la valeur de la fonction sous l'hypothèse nulle de localisation aléatoire des points. L'intervalle de confiance global (Duranton and Overman, 2005) est calculé en précisant l'argument `Global = TRUE`.

```
X %>%
  MEnvelope(r = r, ReferenceType = "Case", Global = TRUE) %>%
  autoplot()
```



7.2.3 Matrice de distances

La fonction `as.Dtable()` permet de créer un objet `Dtable`.

```
d_matrix <- as.Dtable(points_df)
```

Il peut aussi être créé à partir d'une matrice de distances obtenue autrement, contenant par exemple des distances non euclidiennes (temps de transport, distance routière…).

```
# A Dtable containing two points
Dmatrix <- matrix(c(0, 1, 1, 0), nrow = 2)
PointType <- c("Type1", "Type2")
PointWeight <- c(2, 3)
Dtable(Dmatrix, PointType, PointWeight)
```

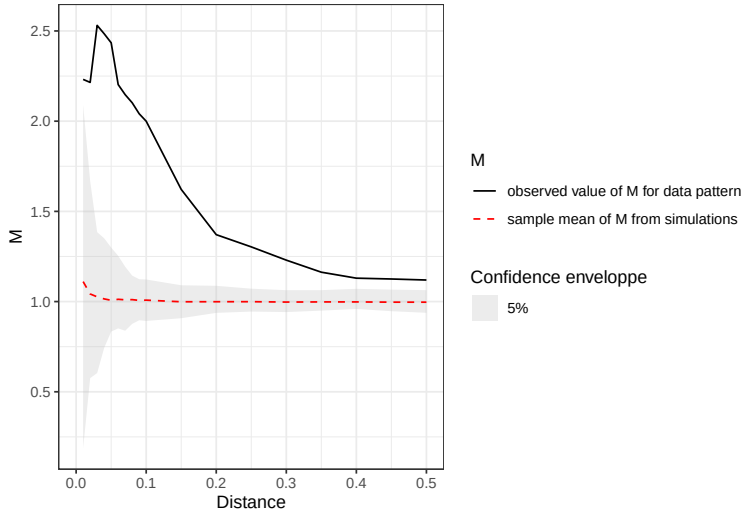
```
## $Dmatrix
##      [,1] [,2]
## [1,]    0    1
## [2,]    1    0
##
## $n
## [1] 2
##
## $marks
## $marks$PointType
## [1] Type1 Type2
## Levels: Type1 Type2
##
## $marks$PointWeight
## [1] 2 3
##
##
## attr(,"class")
## [1] "Dtable"
```

Les fonctions `Mhat()` et `MEnvelope()` sont les mêmes que pour les jeux de points.

```
identical(
  Mhat(X, r = r, ReferenceType = "Case", NeighborType = "Control"),
  Mhat(d_matrix, r = r, ReferenceType = "Case", NeighborType = "Control")
)
```

```
## [1] TRUE
```

```
d_matrix %>%
  MEnvelope(r = r, ReferenceType = "Case", Global = TRUE) %>%
  autoplot()
```



7.3 Performance de M

La fonction `X_to_M()` calcule la fonction M et renvoie le vecteur de ses valeurs pour chaque distance. Elle est utile pour mesurer les temps d'exécution.

```
# Compute M
X_to_M <- function(X) {
  X %>%
    Mhat(r = r, ReferenceType = "Case") %>%
    pull("M")
}
```

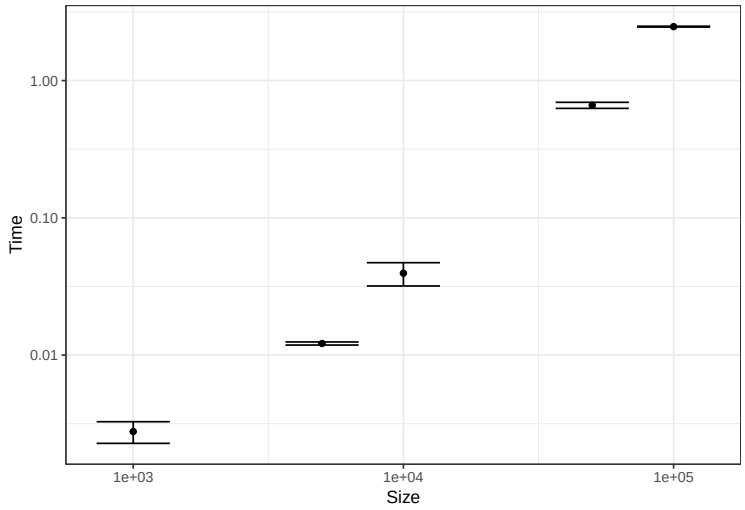
7.3.1 Temps de calcul

Le temps nécessaire au calcul exact est évalué pour une gamme de nombres de points précisée dans `X_sizes`.

```
X_sizes <- c(1000, 5000, 10000, 50000, 100000)
```

La fonction `test_time()` permet de mesurer le temps d'exécution d'une évaluation de la fonction M.

```r
library("microbenchmark")
test_time <- function(points_nb) {
  X <- X_csr(points_nb)
  microbenchmark(X_to_M(X), times = 4L) %>%
    pull("time")
}

X_sizes %>%
  sapply(FUN = test_time) %>%
  as_tibble() %>%
  pivot_longer(cols = everything()) %>%
  rename(Size = name) %>%
  group_by(Size) %>%
  summarise(Time = mean(value) / 1E9, sd = sd(value) / 1E9) %>%
  mutate(
    Size = as.double(
      plyr::mapvalues(
        .$Size,
        from = paste0("V", seq_along(X_sizes)),
        to = X_sizes
      )
    )
  ) -> M_time
M_time %>%
  ggplot(aes(x = Size, y = Time)) +
    geom_point() +
    geom_errorbar(aes(ymin = Time - sd, ymax = Time + sd)) +
    scale_x_log10() +
    scale_y_log10()
```



Le temps de calcul est lié à la taille du jeu de points par une loi puissance.

```r
# Model
M_time %>%
  mutate(logTime = log(Time), logSize = log(Size)) ->
  M_time_log
M_time_lm <- lm(logTime ~ logSize, data = M_time_log)
summary(M_time_lm)
```

```
##
## Call:
```

```
## lm(formula = logTime ~ logSize, data = M_time_log)
##
## Residuals:
##        1        2        3        4        5
##  0.47838 -0.47762 -0.34658  0.03725  0.30857
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -16.8107     1.2296  -13.67 0.000847
## logSize       1.5120     0.1289   11.73 0.001333
##
## (Intercept) ***
## logSize     **
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4739 on 3 degrees of freedom
## Multiple R-squared:  0.9786, Adjusted R-squared:  0.9715
## F-statistic: 137.5 on 1 and 3 DF,  p-value: 0.001333
```

Le package *microbenchmark* proposé par Mersmann (2023) est retenu pour comparer le temps de calcul de la fonction entre un jeu de points et une matrice de distances.

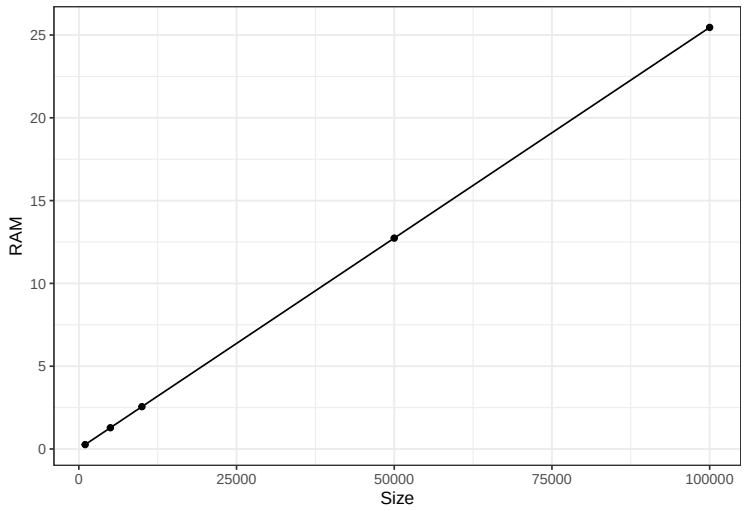
Le calcul des distances est extrêmement rapide dans la fonction `Mhat()` : la matrice le fait économiser, mais le traitement complet à partir d'une matrice est finalement plus long.

```
library("microbenchmark")
mb <- microbenchmark(
  Mhat(X, r = r, ReferenceType = "Case", NeighborType = "Control"),
  Mhat(d_matrix, r = r, ReferenceType = "Case", NeighborType = "Control"),
  times = 4L
)
```

7.3.2 Mémoire

La mémoire utilisée est évaluée avec le package *profmem*.

```
# RAM
library("profmem")
test_ram <- function(points_nb) {
  X <- X_csr(points_nb)
  profmem(X_to_M(X)) %>%
    pull("bytes") %>%
    sum(na.rm = TRUE)
}
sapply(X_sizes, FUN = test_ram) %>%
  tibble(Size = X_sizes, RAM = . / 2^20) ->
  M_ram
M_ram %>%
  ggplot(aes(x = Size, y = RAM)) +
    geom_point() +
    geom_line()
```



La mémoire nécessaire (en Mo) augmente linéairement avec le nombre de points.

```
# Model
lm(RAM ~ Size, data = M_ram) |> summary()
```

```
##
## Call:
## lm(formula = RAM ~ Size, data = M_ram)
##
## Residuals:
##          1          2          3          4
## -0.0005618 -0.0002689  0.0006505  0.0004524
##          5
## -0.0002722
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.228e-02  3.591e-04    34.2 5.49e-05
## Size        2.545e-04  7.146e-09 35610.2 4.88e-14
##
## (Intercept) ***
## Size        ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0006027 on 3 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 1.268e+09 on 1 and 3 DF,  p-value: 4.884e-14
```

7.4 Effets de l'approximation de la position des points

Le nombre de répétition des tests est fixé par `simulations_n`.

```
simulations_n <- 100
```

7.4.1 Cas d'une distribution agrégée (Matérn)

`X_matern_list` contient `simulations_n` tirages du jeu de points. `X_matern_grouped_list` contient les mêmes simulations, dont les points ont été regroupés dans les cases de la grille.

```
# Simulate X
X_matern_list <- replicate(
  simulations_n,
  expr = X_matern(par_points_nb),
  simplify = FALSE
)
# Group points and compute M
X_matern_grouped_list <- lapply(
  X_matern_list,
  FUN = group_points,
  partitions = par_partitions
)
```

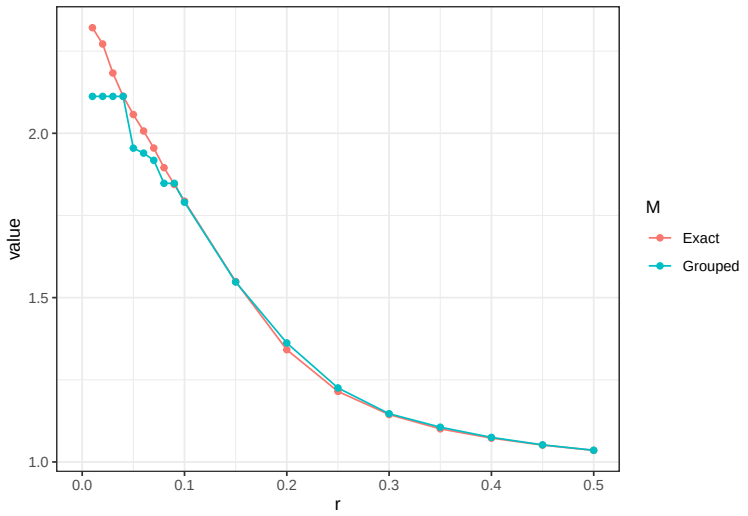
Pour évaluer l'effet de l'approximation de la position, le calcul exact et le calcul sur les points de la grille sont effectués sur chaque jeu de points.

```
library("pbapply")
# Compute M
M_matern_original <- pbsapply(X_matern_list, FUN = X_to_M)
M_matern_grouped <- pbsapply(X_matern_grouped_list, FUN = X_to_M)
```

Le calcul approximé est très rapide parce qu'il réduit le nombre de points à celui du nombre de cellules, à condition d'en tirer partie dans le code utilisé pour le calcul. Ce n'est pas le cas ici : le package *dbmss* ne prévoit pas cette approximation. La fonction `M_hat()` est donc appliquée au jeu de points regroupé mais calculée de la même façon qu'avec le jeu de points original.

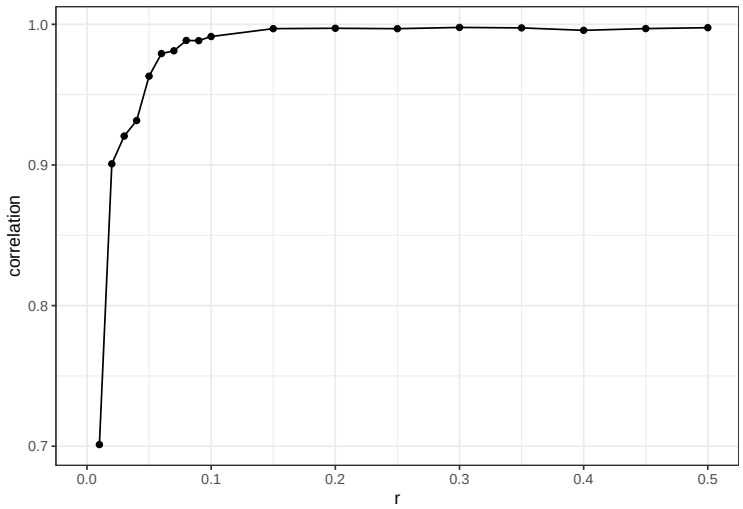
Les valeurs moyennes des estimations de M sont présentées ci-dessous.

```
tibble(
  r,
  Exact = rowMeans(M_matern_original),
  Grouped =  rowMeans(M_matern_grouped)
) %>%
  pivot_longer(
    cols = !r,
    names_to = "M",
    values_to = "value"
  ) %>%
  ggplot(aes(x = r, y = value, color = M)) +
  geom_line() +
  geom_point()
```



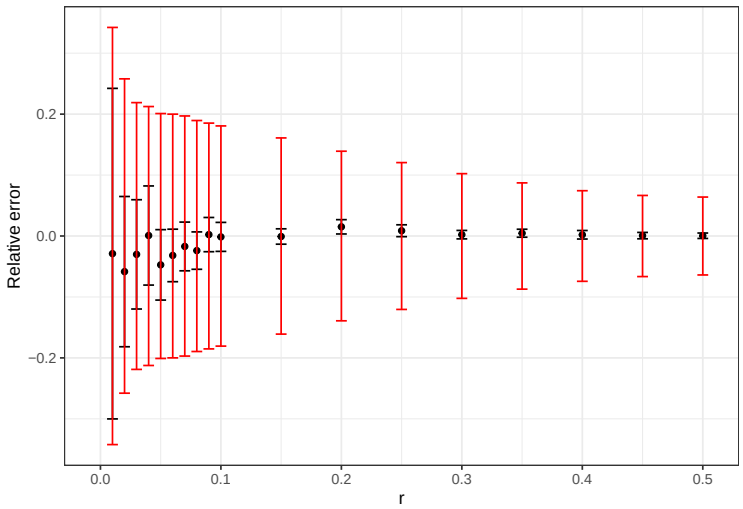
La corrélation entre les valeurs de M estimées par chaque méthode est calculée à chaque distance.

```
# Correlation
M_cor <- function(r_value, M_original, M_grouped) {
  r_index <- which(r == r_value)
  # Return
  c(
    # Distance
    r_value,
    # Correlation
    cor(M_original[r_index, ], M_grouped[r_index, ])
  )
}
sapply(
  r,
  FUN = M_cor,
  M_original = M_matern_original,
  M_grouped = M_matern_grouped
) %>%
  t() %>%
  as_tibble() %>%
  rename(r = V1, correlation = V2) %>%
  ggplot(aes(x = r, y = correlation)) +
    geom_point() +
    geom_line()
```



La corrélation est très élevée dès que la distance prise en compte dépasse la maille de la grille. Les valeurs sont ensuite comparées.

```
# Compare values
M_bias <- function(r_value, M_original, M_grouped) {
  r_index <- which(r == r_value)
  # Return
  c(
    # Distance
    r_value,
    # Relative error
    mean((M_grouped[r_index, ] - M_original[r_index, ]) / M_original[r_index, ]),
    # Standardised error sd
    sd(M_grouped[r_index, ] - M_original[r_index, ]) / mean(M_grouped[r_index, ]),
    # Coefficient of variation
    sd(M_original[r_index, ] / mean(M_original[r_index, ]))
  )
}
sapply(
  r,
  FUN = M_bias,
  M_original = M_matern_original,
  M_grouped = M_matern_grouped
) %>%
  t() %>%
  as_tibble() %>%
  rename(r = V1, `Relative error` = V2, `Error CV` = V3, `M CV` = V4) %>%
  ggplot() +
    geom_point(aes(x = r, y = `Relative error`)) +
    geom_errorbar(
      aes(
        x = r,
        ymin = `Relative error` - `Error CV`,
        ymax = `Relative error` + `Error CV`
      )
    ) +
    geom_errorbar(aes(x = r, ymin = -`M CV`, ymax = `M CV`), col = "red")
```



La figure ci-dessus montre, en rouge, la variabilité de la valeur de M (son coefficient de variation) au cours des simulations. Par définition, la valeur moyenne est sans erreur. A courte distance, les valeurs de M varient beaucoup pour un même processus ponctuel, en fonction de la stochasticité de ses tirages. Comme M est une fonction cumulative, elle se stabilise quand la distance augmente.

L'erreur relative (à la valeur exacte de M) moyenne, due à l'approximtion de la position des points, est présentée en noir, avec son écart-type normalisé par la valeur exacte de M. Elle est faible, inférieure à 10%, même à courte distance.

7.4.2 Cas d'une distribution complètement aléatoire (CSR)

`X_csr_list` contient `simulations_n` tirages du jeu de points.

```
# Simulate X
X_csr_list <- replicate(
  simulations_n,
  expr = X_csr(par_points_nb),
  simplify = FALSE
)
# Group points and compute M
X_csr_grouped_list <- lapply(
  X_csr_list,
  FUN = group_points,
  partitions = par_partitions
)
```

Le calcul exact et le calcul sur les points de la grille sont effectués sur chaque jeu de points.

```
# Compute M
system.time(M_csr_original <- pbsapply(X_csr_list, FUN = X_to_M))
```

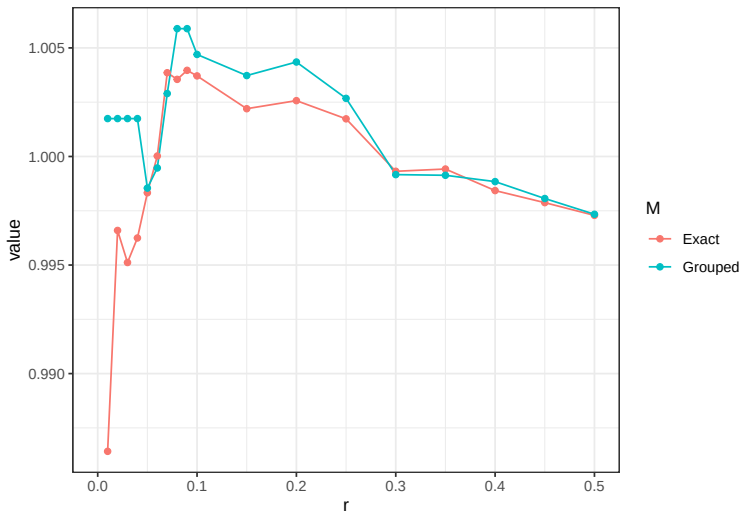
```
##    user  system elapsed
##   2.286   0.041   1.152
```

```
system.time(M_csr_grouped <- sapply(X_csr_grouped_list, FUN = X_to_M))
```

```
##    user  system elapsed
##   2.254   0.041   1.088
```

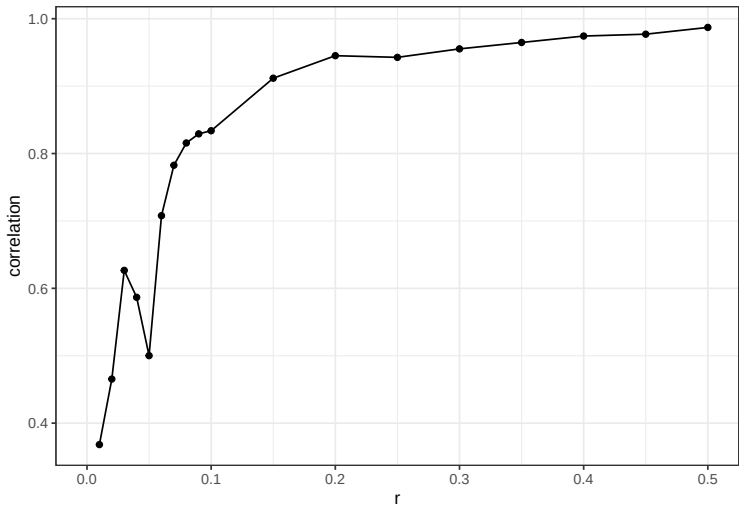
Les valeurs moyennes sont présentées ci-dessous.

```
tibble(
  r,
  Exact = rowMeans(M_csr_original),
  Grouped =  rowMeans(M_csr_grouped)
) %>%
  pivot_longer(
    cols = !r,
    names_to = "M",
    values_to = "value"
  ) %>%
  ggplot(aes(x = r, y = value, color = M)) +
  geom_line() +
  geom_point()
```



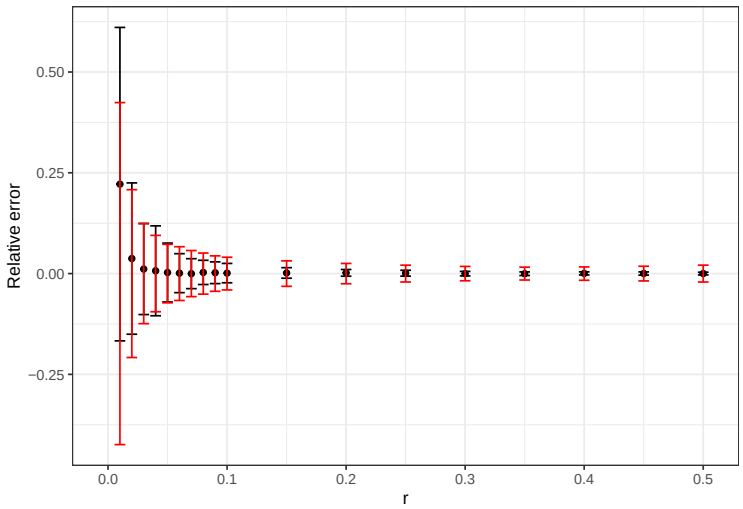
La corrélation entre les valeurs de M calculées par chaque méthode est calculée à chaque distance.

```
# Correlation
sapply(
  r,
  FUN = M_cor,
  M_original = M_csr_original,
  M_grouped = M_csr_grouped
) %>%
  t() %>%
  as_tibble() %>%
  rename(r = V1, correlation = V2) %>%
  ggplot(aes(x = r, y = correlation)) +
    geom_point() +
    geom_line()
```



En absence de structure spatiale, les corrélations sont bien plus faibles. Les valeurs sont comparées.

```
# Compare values
sapply(
  r, FUN = M_bias,
  M_original = M_csr_original,
  M_grouped = M_csr_grouped
) %>%
  t() %>%
  as_tibble() %>%
  rename(r = V1, `Relative error` = V2, `Error CV` = V3, `M CV` = V4) %>%
  ggplot() +
    geom_point(aes(x = r, y = `Relative error`)) +
    geom_errorbar(
      aes(
        x = r,
        ymin = `Relative error` - `Error CV`,
        ymax = `Relative error` + `Error CV`
      )
    ) +
    geom_errorbar(aes(x = r, ymin = -`M CV`, ymax = `M CV`), col = "red")
```



La figure ci-dessus est construite de la même façon que pour les jeux de points agrégés. En absence de structure spatiale, la valeur de M varie beaucoup moins.

En présence d'une structure spatiale, l'erreur d'estimation est grande à courte distance. Elle devient négligeable au-delà de la maille de la grille.

Références

Arbia, G. (1989). *Spatial Data Configuration in Statistical Analysis of Regional Economic and Related Problems*. Dordrecht : Kluwer.

Baddeley, A., E. Rubak, and R. Turner (2016). *Spatial Point Patterns : Methodology and Applications with R*. Chapman & Hall/CRC Interdisciplinary Statistics Series. Boca Raton London New York : CRC Press.

Cressie, N. A. (1993). *Statistics for Spatial Data*. New York : John Wiley & Sons.

Deurloo, M. C. and S. De Vos (2008). Measuring segregation at the micro level : An application of the M measure to multi-ethnic residential neighbourhoods in Amsterdam. *Tijdschrift voor economische en sociale geografie 99*(3), 329–347.

Dray, N., L. Mancini, U. Binshtok, F. Cheysson, W. Supatto, P. Mahou, S. Bedu, S. Ortica, E. Than-Trong, M. Krecsmarik, S. Herbert, J.-B. Masson, J.-Y. Tinevez, G. Lang, E. Beaurepaire, D. Sprinzak, and L. Bally-Cuif (2021). Dynamic spatiotemporal coordination of neural stem cell fate decisions occurs through local feedback in the adult vertebrate brain. *Cell Stem Cell 28*(8), 1457–1472.e12.

Duranton, G. and H. G. Overman (2005). Testing for localisation using micro-geographic data. *Review of Economic Studies 72*(4), 1077–1106.

Lentz, J. A., J. K. Blackburn, and A. J. Curtis (2011). Evaluating Patterns of a White-Band Disease (WBD) Outbreak in Acropora palmata Using Spatial Analysis : A Comparison of Transect and Colony Clustering. *PLoS ONE 6*(7), e21830.

Marcon, E. and F. Puech (2003). Evaluating the geographic concentration of industries using distance-based methods. *Journal of Economic Geography 3*(4), 409–428.

Marcon, E. and F. Puech (2010). Measures of the geographic concentration of industries : Improving distance-based methods. *Journal of Economic Geography 10*(5), 745–762.

Marcon, E. and F. Puech (2017). A typology of distance-based measures of spatial concentration. *Regional Science and Urban Economics 62*, 56–67.

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.

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

Mersmann, O. (2023). *Microbenchmark : Accurate Timing Functions*.

R Core Team (2024). *R : A Language and Environment for Statistical Computing*. Vienna, Austria : R Foundation for Statistical Computing.

Scholl, T. and T. Brenner (2015). Optimizing distance-based methods for large data sets. *Journal of Geographical Systems 17*(4), 333–351.

Sweeney, S. H. and E. J. Feser (1998). Plant size and clustering of manufacturing activity. *Geographical Analysis 30*(1), 45–64.

Tidu, A., F. Guy, and S. Usai (in press). Measuring Spatial Dispersion : An Experimental Test on the M -Index. *Geographical Analysis*.