

Données spatiales

Contents

Introduction aux statistiques spatiales	1
Indice de Moran	1
Indice de Moran Local:	7
Modèles théoriques du variogramme	11
Variogramme empirique	14
Variogramme et données temporelles	14
Modèle de régression avec corrélation spatiale	14
Modèles géostatistiques dans R	15
Modèles d'autorégression spatiale	20
Données aréales dans R	22
Corrélation spatiale dans des modèles complexes	27
Modèles géostatistiques avec <i>nlme</i>	27
Modèles géostatistiques avec <i>brms</i>	27
Modèles autorégressifs spatiaux avec <i>brms</i>	28
Références	28

Introduction aux statistiques spatiales

Comme toute analyse, les analyses spatiales commencent par l'exploration des données. La présence d'un patron spatial dans les données dépend très souvent de notre échantillonnage, mais elle peut aussi être le résultat de circonstances que nous n'avons pas pu contrôler. Que ce soit l'un ou l'autre, il existe des façons de vérifier si des patrons spatiaux sont présents dans nos modèles et si nos mesures sont sujettes à une autocorrélation spatiale. Nous allons donc utiliser des indices pour vérifier la présence de ces patrons:

Indice de Moran

L'indice I de Moran, ou Moran Global permet de tester si une corrélation significative est présente entre régions voisines.

L'indice de Moran est un coefficient d'autocorrélation spatiale des z , pondéré par les poids w_{ij} . Il prend donc des valeurs entre -1 et 1.

$$I = \frac{N}{\sum_i \sum_j w_{ij}} \frac{\sum_i \sum_j w_{ij}(z_i - \bar{z})(z_j - \bar{z})}{\sum_i (z_i - \bar{z})^2}$$

Dans cette équation, nous reconnaissions l'expression d'une corrélation, soit le produit des écarts à la moyenne de deux variables z_i et z_j , divisé par le produit de leurs écarts-types (qui est le même, donc on obtient la variance). La contribution de chaque paire (i, j) est multipliée par son poids w_{ij} et le terme à gauche (le nombre

de régions N divisé par la somme des poids) assure que le résultat soit borné entre -1 et 1. L'auto-corrélation spatiale positive indique que les éléments ou les valeurs sont regroupés dans l'espace, et les emplacements voisins ont tendance à avoir des valeurs ou des caractéristiques similaires. L'auto-corrélation spatiale négative, en revanche, indique que les éléments ou les valeurs sont dispersés dans l'espace, et les emplacements voisins ont tendance à avoir des valeurs ou des caractéristiques différentes. Puisque la distribution de I est connue en l'absence d'auto-corrélation spatiale, cette statistique permet de tester l'hypothèse nulle selon laquelle il n'y a pas de corrélation spatiale entre régions voisines.

Afficher les données sur une carte est fort utile pour planifier l'analyses. Il y a plusieurs façon de réaliser des cartes sur R. Parmi les packages qui vont vous permettre d'afficher des cartes, vous trouverez rnaturalearth. Pour s'entraîner, nous allons utiliser un jeu de données où le débourrement (budbre), un phase phénologique qui détermine le début de la saison de croissance chez les arbres. Chaque point indique les coordonnées d'un peuplement où la date du débourrement a été observé. La date est indiqué en jour julien (DOY-Day of the year).

Pour commencer, nous importons les données accompagnées de leurs coordonnées. Lors de cette étape, nous allons produire un shapefile avec la fonction st_as_sf du package sf. A cette étape, il est crucial de spécifier le système de référence géodésique (CRS - Coordinate Reference System) utilisé pour positionner les points dans l'espace. Le CRS ne se limite pas seulement à fournir des informations sur l'unité de mesure (angles, distances) et la projection utilisée, mais il englobe également des détails essentiels tels que le sphéroïde de référence (la forme approximative de la terre : conique, cylindrique, ou planaire) et un datum (fournissant des informations sur l'orientation du sphéroïde par rapport à la terre). Ces détails sont cruciaux pour garantir la précision des calculs effectués sur les données spatiales. Par exemple, lorsque nous calculons les distances entre les points, l'unité de mesure de la distance dépend du CRS utilisé. De plus, comprendre les paramètres du CRS permet de projeter correctement les données dans d'autres systèmes de coordonnées si nécessaire, assurant ainsi la cohérence des analyses géospatiales. Vous pouvez explorer et vérifier les informations relatives au CRS sur des sites spécialisés tels que <https://epsg.io/>. Le CRS le plus couramment utilisé est généralement le WGS84, qui est également celui employé dans les systèmes de positionnement par satellite GPS (<https://epsg.io/4326>).

```
require(rnaturalearth)
```

```
## Le chargement a nécessité le package : rnaturalearth
## Warning: le package 'rnaturalearth' a été compilé avec la version R 4.3.2
## The legacy packages maptools, rgdal, and rgeos, underpinning the sp package,
## which was just loaded, will retire in October 2023.
## Please refer to R-spatial evolution reports for details, especially
## https://r-spatial.org/r/2023/05/15/evolution4.html.
## It may be desirable to make the sf package available;
## package maintainers should consider adding sf to Suggests:.
## The sp package is now running under evolution status 2
##     (status 2 uses the sf package in place of rgdal)
## Support for Spatial objects (`sp`) will be deprecated in {rnaturalearth} and will be removed in a fu
require(sp)
```

```
## Le chargement a nécessité le package : sp
require(sf)
```

```
## Le chargement a nécessité le package : sf
## Linking to GEOS 3.11.2, GDAL 3.6.2, PROJ 9.2.0; sf_use_s2() is TRUE
P1phasespatialtraining1 <- P1phasespatialtraining1 <-
read.csv("C:/Users/buttoval/Documents/ECL8202/donnees/P1phasespatialtraining1.csv",
sep = ",")
```

```
P1phasespatialtraining1_shapefile <- st_as_sf(P1phasespatialtraining1,
  coords = c("x_long", "y_lat"), crs = "WGS84")
```

Une fois le shapefile produit, nous pouvons utiliser la fonction ne_countries pour télécharger une carte du pays où nous avons extrait les données. Les données sont associées au catalogue de cartes publiques : Natural Earth Data (<https://www.naturalearthdata.com/>). Les cartes peuvent également être téléchargées depuis le site et sont proposées à différentes échelles : large (1:10 m), medium (1:50 m) et small (1:110 m), selon le niveau de détail souhaité.

On peut télécharger les cartes directement depuis R avec la fonction ne_download. Voici une première carte du monde, à laquelle on peut superposer des couches représentant les lacs et les océans. Une liste complète des couches disponibles est présentée dans les tableaux de la vignette de rnaturalearth (<https://cran.r-project.org/web/packages/rnaturalearth/vignettes/rnaturalearth.html>)

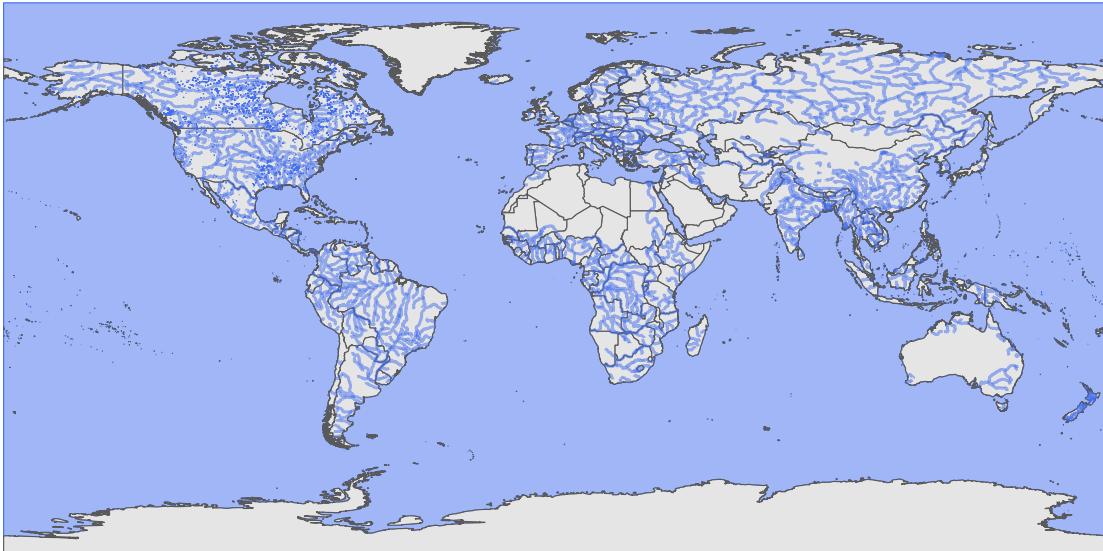
```
sfdf_world <- ne_download(scale = 10, type = "countries", returnclass = "sf")

sfdf_lakes <- ne_download(scale = 10, type = "lakes_north_america",
  category = "physical", returnclass = "sf")

sfdf_oceans <- ne_download(scale = 10, type = "ocean", category = "physical",
  returnclass = "sf")

sfdf_riverslake <- ne_download(scale = 10, type = "rivers_lake_centerlines",
  category = "physical", returnclass = "sf")

ggplot(data = sfdf_world) + geom_sf(data = sfdf_oceans, col = "royalblue2",
  fill = "royalblue2", alpha = 0.5) + geom_sf() + geom_sf(data = sfdf_lakes,
  col = "royalblue2", alpha = 0.5) + geom_sf(data = sfdf_riverslake,
  col = "royalblue2", alpha = 0.5)
```

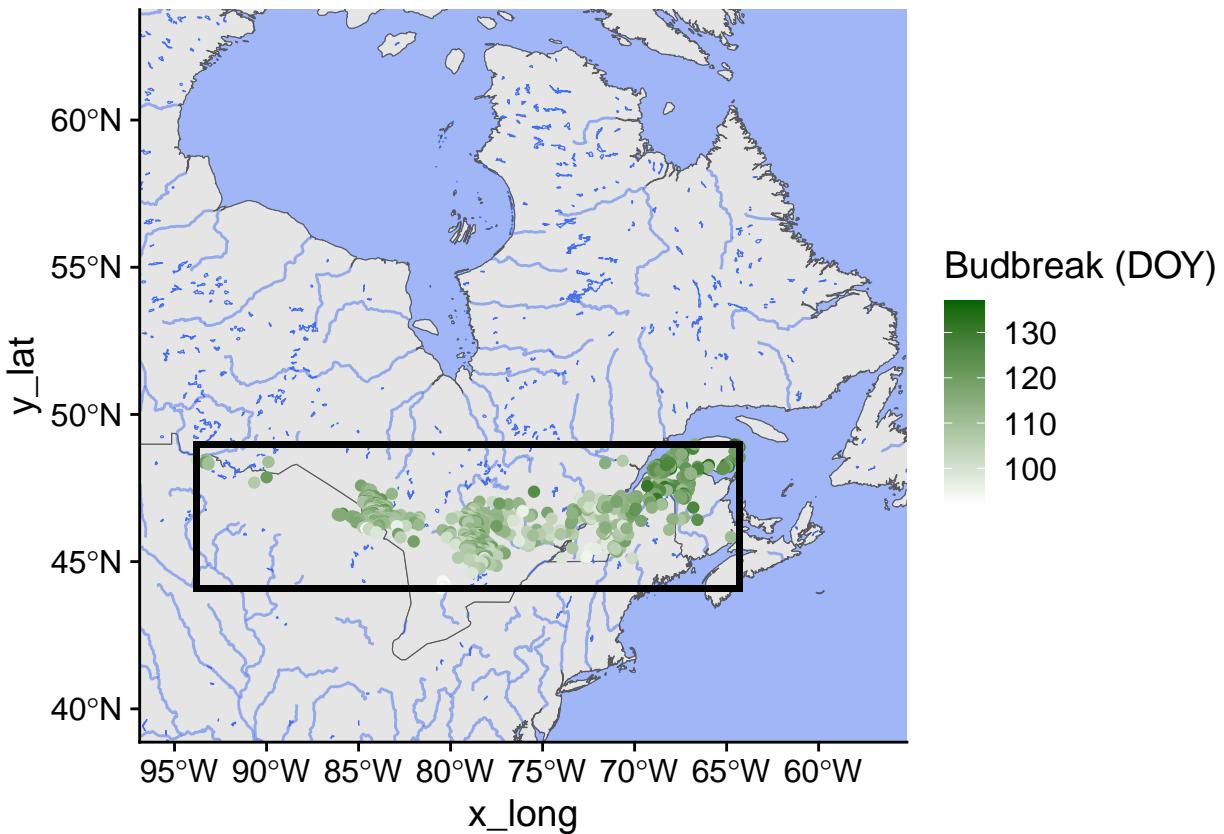


Afin de mieux visualiser nos points, il est maintenant utile de restreindre la carte à notre zone d'intérêt, qui est le nord de l'Amérique. Nous pouvons utiliser `geom_rect` pour afficher uniquement cette zone. Si nécessaire, nous pouvons également créer un carré englobant nos coordonnées en utilisant `geom_rect`. Les coordonnées du carré peuvent être obtenues à l'aide de la fonction `st_box`. Le CRS par défaut des objets obtenus par `rnatualearth` est WGS84, nous n'avons donc pas besoin de le transformer mais on aurait pu utiliser rajouter le code “`shapefile %>% st_transform(notre CRS)`” pour une reprojection.

```
coordbox <- as.data.frame(st_bbox(P1phasespatialtraining1_shapefile))

ggplot(data = sfdf_world) + geom_sf(data = sfdf_oceans, col = "royalblue2",
  fill = "royalblue2", alpha = 0.5) + geom_sf() + geom_sf(data = sfdf_lakes,
  col = "royalblue2", alpha = 0.5) + geom_sf(data = sfdf_riverslake,
  col = "royalblue2", alpha = 0.5) + geom_point(P1phasespatialtraining1,
  mapping = aes(x_long, y_lat, color = DOY)) + scale_color_gradient(name = "Budbreak (DOY)",
  low = "white", high = "darkgreen") + coord_sf(xlim = c(-95,
  -57.1), ylim = c(40, 62.62)) + geom_rect(xmin = coordbox$x[1],
  ymin = coordbox$x[2], xmax = coordbox$x[3], ymax = coordbox$x[4],
  fill = NA, colour = "black", size = 1)

## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



On observe déjà une tendance à un débourrement (budbreak) plus précoce vers le sud-ouest et plus tardif vers le nord-est. Mais comment valider ces patrons?

On peut commencer par calculer l'indice I de Moran. On va faire ça en différentes étapes:

On commence par la création de la matrice de coordonnées : Nous commençons par créer une matrice contenant les coordonnées latitudinales et longitudinales de chaque point d'observation. Cela nous permettra de représenter spatialement nos données.

```
require(spdep)

## Le chargement a nécessité le package : spdep
## Warning: le package 'spdep' a été compilé avec la version R 4.3.2
## Le chargement a nécessité le package : spData
## To access larger datasets in this package, install the spDataLarge
## package with: `install.packages('spDataLarge',
## repos='https://nowosad.github.io/drat/', type='source')`
latlon = cbind(P1phasespatialtraining1$x_long, P1phasespatialtraining1$y_lat)
```

Conversion en objet spatial : Ensuite, nous convertissons cette matrice en un objet spatial afin de pouvoir effectuer des analyses spatiales.

```
latlon<-coordinates(latlon)
```

Création des identifiants : Nous générions des identifiants uniques pour chaque point, ce qui sera utile pour la création de la liste de voisins.

```
IDs <- row.names(as.data.frame(x = P1phasespatialtraining1$x_long,
y = P1phasespatialtraining1$y_lat))
```

Création de la liste de voisins : Nous utilisons l'algorithme des k plus proches voisins pour créer une liste de voisins pour chaque point. Ici, nous utilisons k=1 pour considérer uniquement les liens directs entre les points les plus proches. Cette ligne crée une liste de voisins basée sur le plus proche voisin (k=1) en utilisant les coordonnées latitudinales et longitudinales. Cela signifie que chaque point est lié à son voisin le plus proche. étant donné que k=1 alors chaque observation a un voisin. On souhaite évaluer l'autocorrélation spatiale en considérant uniquement les liens directs entre les points les plus proches. Cela peut être approprié dans certains cas, notamment lorsque l'on suppose que l'influence spatiale décroît rapidement avec la distance et que les valeurs sont fortement influencées par leur voisinage immédiat.

```
Neigh_nb <- knn2nb(knearneigh(latlon, k = 1, longlat = TRUE),
row.names = IDs)
```

Standardisation des poids : Nous standardisons les poids de la liste de voisins de manière à ce que la somme des poids pour chaque unité spatiale soit égale à 1. Cela permet une comparaison appropriée des poids entre différentes unités spatiales. Ici, style = "W" indique que les poids pour chaque unité spatiale sont standardisés de manière à ce que leur somme soit égale à 1 (ceci est appelé standardisation par ligne).

```
Neigh_nb_moran<-nb2listw(Neigh_nb,style="W")
```

Test de Moran : Enfin, nous appliquons le test de Moran sur nos données. Cela nous permet d'évaluer l'autocorrélation spatiale de notre variable d'intérêt (dans ce cas, le jour de l'année du débourrement des plantes) en utilisant la liste de voisins standardisée que nous avons créée précédemment.

```
Moran_I <- moran.test(P1phasespatialtraining1$DOY, listw = Neigh_nb_moran)
```

```
Moran_I
```

```
##
## Moran I test under randomisation
##
## data: P1phasespatialtraining1$DOY
## weights: Neigh_nb_moran
##
## Moran I statistic standard deviate = 12.302, p-value < 2.2e-16
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic      Expectation      Variance
##          0.557497347     -0.001297017     0.002063322
```

Dans le tableau, nous trouvons les valeurs de l'écart-type standard de la statistique de Moran I (Moran I statistic standard deviate). Cela indique la déviation standardisée de la statistique de Moran I par rapport à sa moyenne sous l'hypothèse nulle d'absence d'autocorrélation spatiale. Lorsque la déviation standardisée de la statistique de Moran I s'écarte de zéro vers des valeurs positives élevées, cela suggère une forte autocorrélation spatiale positive, ce qui signifie que les valeurs similaires ont tendance à être regroupées dans l'espace. Dans notre cas, nous avons obtenu une valeur de 12.302, ce qui est plutôt élevée.

La statistique de Moran I est de 0.6, ce qui indique une autocorrélation spatiale positive. Cela signifie qu'il existe une tendance pour les valeurs similaires à se regrouper dans l'espace

La valeur p associée à la statistique de Moran I est inférieure à 0,05, ce qui suggère un rejet de l'hypothèse nulle. Dans ce cas, l'hypothèse nulle serait l'absence d'autocorrélation spatiale.

Dans l'ensemble ces résultats suggèrent une forte autocorrélation spatiale (statistique Moran I élevée) avec une très faible probabilité que cela soit dû au hasard (p-value très faible), et l'hypothèse alternative est que l'autocorrélation spatiale est plus grande que prévu par le hasard.

“Expectation” fait référence à la valeur attendue de la statistique de Moran I sous l’hypothèse nulle d’absence d’autocorrélation spatiale. Plus précisément, c’est la moyenne théorique de la statistique de Moran I calculée sur de nombreuses répétitions aléatoires de l’échantillon dans lequel la distribution spatiale des valeurs est aléatoire. Si la statistique de Moran I observée diffère considérablement de cette valeur attendue, cela suggère une autocorrélation spatiale significative dans vos données.

Indice de Moran Local:

Si l'autocorrelation spatiale est significatif, il est possible de calculer le Moran Local qui évalue l'autocorrelation spatiale pour chaque unité spatiale individuelle (points, zones etc..). Cet indicateur identifie les zones spécifiques où l'autocorrelation spatiale est significativement différente de celle attendue par hasard.

Pour calculer l'indice de Moran sur R il est possible

$$I_i = \frac{(z_i - \bar{z})}{S^2} \sum_j w_{ij}(z_j - \bar{z})$$

où z_i est la valeur de la variable pour l'unité spatiale i , \bar{z} est la moyenne des valeurs de la variable pour toutes les unités spatiales. w_{ij} est le poids entre les unités spatiales i et j , z_j est la valeur de la variable pour l'unité spatiale j ; S^2 est la variance de la variable dans l'ensemble des unités spatiales.

Pour calculer l'indice de Moran Local on commence par le calcul des distances entre les voisins les plus proches :

```
dstsP1 <- unlist(nbdists(Neigh_nb, latlon))
```

On Calcul ensuite la distance maximale entre les voisins les plus proches et on crée différentes structures de voisinage basées sur la distance :

```
max_1nnP1 <- max(dstsP1)
```

```
Neigh_kd2P1 <- dnearneigh(latlon, d1 = 0, d2 = 2 * max_1nnP1,
  row.names = IDs)
```

neighbors within 2X maximum distance

Création de poids pour la structure de voisinage et Calcul de l'indice de Moran local. Nous pouvons choisir le style B (binaire) pour le création de poids de la structure de voisinage. Nous choisissons d'utiliser le Style de pondération binaire (B) car chaque paire d'unités spatiales est considérée comme soit connectée (1) soit non connectée (0). Ceci facilitera l'identification des clusters spatiaux, ce qui est en fin de compte l'objectif principal de l'analyse de Moran local.

```
weightsP1 <- nb2listw(Neigh_kd2P1, style = "B")
```

*# row standardized binary weights, using minimum distance
for one neighbor*

```
weightsP1
```

```
## Characteristics of weights list object:  
## Neighbour list object:  
## Number of regions: 772  
## Number of nonzero links: 182720  
## Percentage nonzero weights: 30.65854  
## Average number of links: 236.6839  
## 2 disjoint connected subgraphs
```

```

## 
## Weights style: B
## Weights constants summary:
##   n      nn      S0      S1      S2
## B 772 595984 182720 365440 187478256
localP1 <- localmoran(x = P1phasespatialtraining1$DOY, listw = nb2listw(Neigh_kd2P1,
  style = "B"))

head(localP1)

##           Ii        E.Ii     Var.Ii      Z.Ii Pr(z != E(Ii))
## 1 -62.134967 -0.067335093 30.035740 -11.325203 9.845025e-30
## 2  49.310064 -0.111485688 46.483906  7.248782 4.205357e-13
## 3 148.952876 -0.760734852 465.290603  6.940634 3.903441e-12
## 4  2.430195 -0.004170537  2.115739  1.673612 9.420693e-02
## 5 -47.977739 -0.737517583 359.903025 -2.490114 1.277022e-02
## 6  95.099477 -0.230898178 139.861426  8.060878 7.574863e-16

```

Dans l'objet créé par la fonction localmoran, nous pouvons trouver:

- L'indice de Moran local pour l'unité spatiale i : une mesure l'autocorrélation spatiale locale pour cette unité spatiale en prenant en compte les valeurs de ses voisins locaux.
- L'espérance de l'indice de Moran local ($E(I_i)$): la valeur moyenne de l'indice de Moran local dans un ensemble d'échantillons aléatoires, sous l'hypothèse nulle d'absence d'autocorrélation spatiale.
- La variance de l'indice de Moran local ($Var(I_i)$): la dispersion des valeurs de l'indice de Moran local autour de son espérance. Une variance élevée suggère une grande variabilité dans les valeurs de l'indice de Moran local.
- Le score Z de l'indice de Moran local ($Z(I_i)$): le nombre d'écart-types par rapport à l'espérance dans l'indice de Moran local observé
- $Pr(z \neq E(I_i))$: La probabilité que le score Z de l'indice de Moran local diffère de l'espérance. Si cette probabilité est inférieure à un seuil alpha spécifié (généralement 0.05), on rejette l'hypothèse nulle d'absence d'autocorrélation spatiale locale en faveur de l'hypothèse alternative d'autocorrélation spatiale locale significative.

Avec les valeurs de Z , il est possible de séparer les observations en groupes basés sur l'indice de Moran local et d'identifier les zones où l'agrégation spatiale est significative. Ces catégories sont basées sur les seuils de significativité des scores Z pour l'indice de Moran local. Ils sont utilisés pour identifier les clusters spatiaux significatifs à différents niveaux de confiance statistique (par exemple, 90%, 95% et 99%). La catégorie "4-random" est utilisée pour les unités spatiales où l'autocorrélation spatiale n'est pas significative.

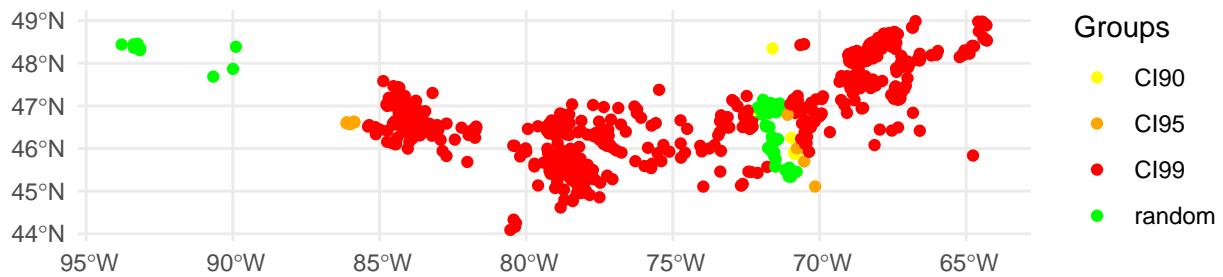
```

moran.mapP1 <- cbind(P1phasespatialtraining1_shapefile,
  localP1)

moran.mapP1$groups_class <- with(moran.mapP1, ifelse(Z.Ii >=
  -24 & Z.Ii < -2.58, "CI99", ifelse(Z.Ii >= -2.58 &
  Z.Ii < -1.96, "CI95", ifelse(Z.Ii >= -1.96 & Z.Ii <
  -1.65, "CI90", ifelse(Z.Ii >= -1.65 & Z.Ii < 1.65,
  "random", ifelse(Z.Ii >= 1.65 & Z.Ii < 1.96, "CI90",
  ifelse(Z.Ii >= 1.96 & Z.Ii < 2.58, "CI95",
  "CI99"))))))
ggplot() + geom_sf(data = moran.mapP1, aes(color = groups_class)) +
  scale_color_manual(values = c(CI99 = "#ff0000", CI95 = "#ffa500",
  CI90 = "#ffff00", random = "#00ff00"), name = "Groups") +
  labs(title = "Moran Local Groups") + theme_minimal()

```

Moran Local Groups



La plus part des valeurs tombent dans le CI99 (intervalle de confiance à 99%). Les valeurs de la variable étudiée, le jour de débourrement (DOY), sont fortement regroupées dans l'espace, formant des clusters spatiaux significatifs.

Si nous sommes intéressés par l'investigation des clusters spatiaux significatifs, il serait possible de réaliser une analyse des points chauds et des points froids (hotspot-coldspot) pour déterminer si les valeurs très élevées et celles qui sont plus petites ont tendance à s'agglomérer dans des parties différentes de la carte. Pour ce faire, nous avons besoin de calculer un autre indice, le Getis-Ord G G_{i*} qui indique la concentration spatiale et qui évalue si les valeurs d'une variable sont réparties de manière aléatoire dans l'espace ou si elles présentent une tendance à la concentration spatiale, c'est-à-dire des clusters de valeurs élevées ou faibles. G_{i*} calcule des statistiques z pour chaque unité spatiale et peut être utilisé pour identifier les hotspots (valeurs élevées entourées de valeurs élevées) et les coldspots (valeurs faibles entourées de valeurs faibles). Encore une fois, la fonction localG_perm du package spdep nous aide à calculer G_{i*} à partir de la matrice de pondération spatiale que nous avons utilisée avant:

```
local_gpermP1 <- localG_perm(P1phasespatialtraining1_shapefile$DOY,
                                weightsP1)

P1phasessta_shape <- cbind(P1phasespatialtraining1_shapefile,
                            gstat = as.matrix(local_gpermP1))
```

Également, il est possible de séparer les groupes sur la base des valeurs de Z pour identifier les points chauds (Z positifs) et les points froids (Z négatifs). Les hotspots sont associés à des valeurs de Z positives, ce qui signifie que les valeurs dans ces zones sont plus élevées que prévu par rapport à la répartition spatiale globale des données. Les coldspots sont associés à des valeurs de Z négatives, ce qui signifie que les valeurs dans ces zones sont plus faibles que prévu par rapport à la répartition spatiale globale des données.

```

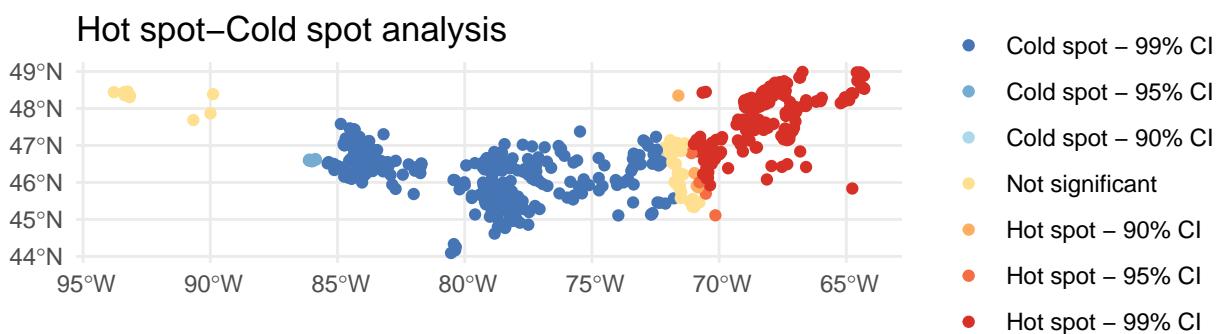
P1phasessta_shape$groups_class <- with(P1phasessta_shape,
  ifelse(gstat >= min(P1phasessta_shape$gstat) & gstat <
    -2.58, "1-CI99", ifelse(gstat >= -2.58 & gstat <
    -1.96, "2-CI95", ifelse(gstat >= -1.96 & gstat <
    -1.65, "3-CI90", ifelse(gstat >= -1.65 & gstat <
    1.65, "4-random", ifelse(gstat >= 1.65 & gstat <
    1.96, "5-CI90", ifelse(gstat >= 1.96 & gstat <
    2.58, "6-CI95", "7-CI99"))))))
}

P1phasessta_shape$groups_classlabelled <- as.factor(P1phasessta_shape$groups_class)

levels(P1phasessta_shape$groups_classlabelled) <- list(`Cold spot - 99% CI` = "1-CI99",
  `Cold spot - 95% CI` = "2-CI95", `Cold spot - 90% CI` = "3-CI90",
  `Not significant` = "4-random", `Hot spot - 90% CI` = "5-CI90",
  `Hot spot - 95% CI` = "6-CI95", `Hot spot - 99% CI` = "7-CI99")

ggplot() + geom_sf(data = P1phasessta_shape, aes(color = groups_classlabelled)) +
  scale_color_manual("", values = c("#4575b4", "#74add1",
  "#abd9e9", "#fee090", "#fdae61", "#f46d43",
  "#d73027")) + labs(title = "Hot spot-Cold spot analysis") +
  theme_minimal()

```



#Variogramme

En géostatistique, le variogramme est un outil fondamental pour étudier la structure de l'autocorrélation spatiale d'une variable géographique. Le variogramme mesure la variance des différences entre les valeurs de cette variable à différents emplacements spatiaux, en fonction de la distance qui les sépare. Il fournit ainsi des informations sur la manière dont la similarité entre les valeurs de la variable diminue à mesure que la distance spatiale entre les emplacements augment.

Le variogramme γ_z de la variable z est égal à la moitié de l'écart carré moyen entre les valeurs de z pour deux points (x_i, y_i) et (x_j, y_j) séparés par une distance h .

$$\gamma_z(h) = \frac{1}{2} E \left[(z(x_i, y_i) - z(x_j, y_j))^2 \right]_{d_{ij}=h}$$

Dans cette équation, la fonction E avec l'indice $d_{ij} = h$ désigne l'espérance statistique (autrement dit, la moyenne) de l'écart au carré entre les valeurs de z pour les points séparés par une distance h .

Si on préfère exprimer l'autocorrélation $\rho_z(h)$ entre mesures de z séparées par une distance h , celle-ci est reliée au variogramme par l'équation:

$$\gamma_z = \sigma_z^2 (1 - \rho_z)$$

,

où σ_z^2 est la variance globale de z .

Modèles théoriques du variogramme

Plusieurs modèles paramétriques ont été proposés pour représenter la corrélation spatiale en fonction de la distance entre points d'échantillonnage. Considérons d'abord une corrélation qui diminue de façon exponentielle:

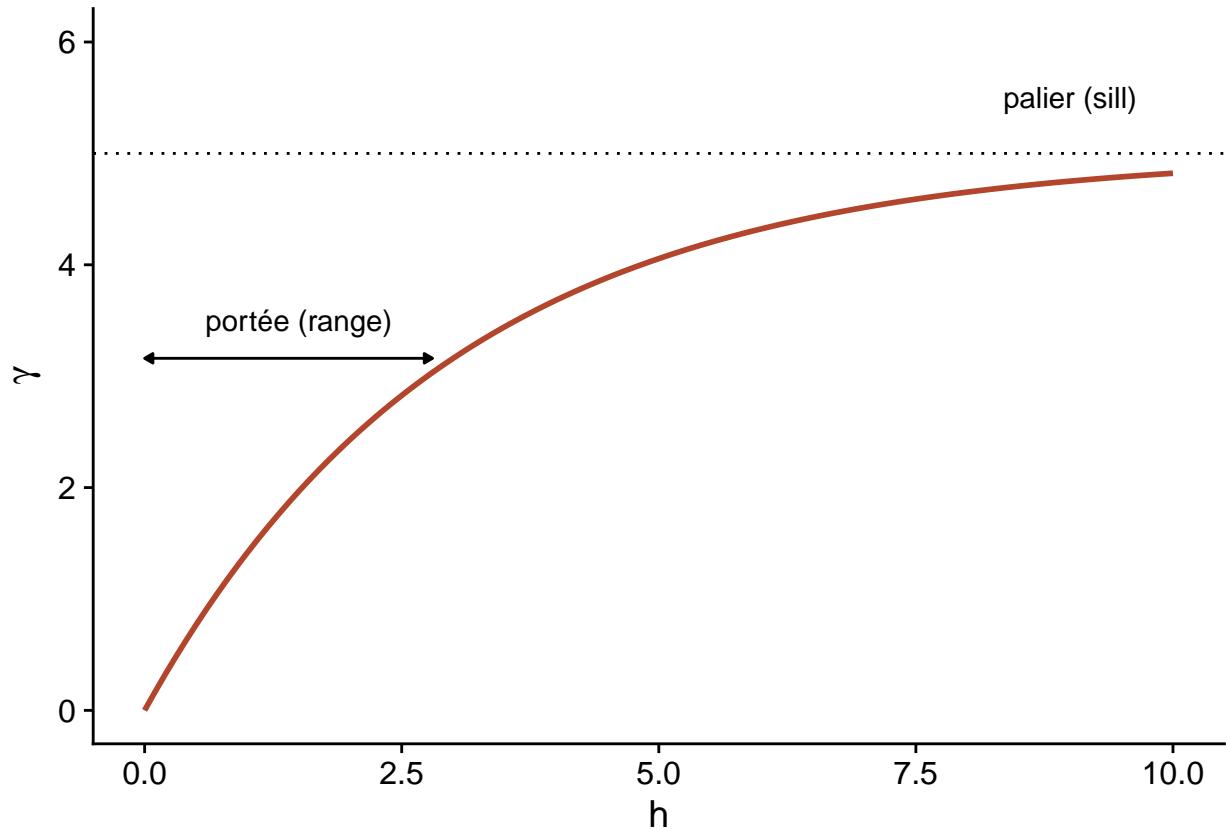
$$\rho_z(h) = e^{-h/r}$$

Ici, $\rho_z = 1$ pour $h = 0$ et la corrélation est multipliée par $1/e \approx 0.37$ pour chaque augmentation de r de la distance. Dans ce contexte, r se nomme la portée (*range*) de la corrélation.

À partir de l'équation ci-dessus, nous pouvons calculer le variogramme correspondant.

$$\gamma_z(h) = \sigma_z^2 (1 - e^{-h/r})$$

Voici une représentation graphique de ce variogramme.

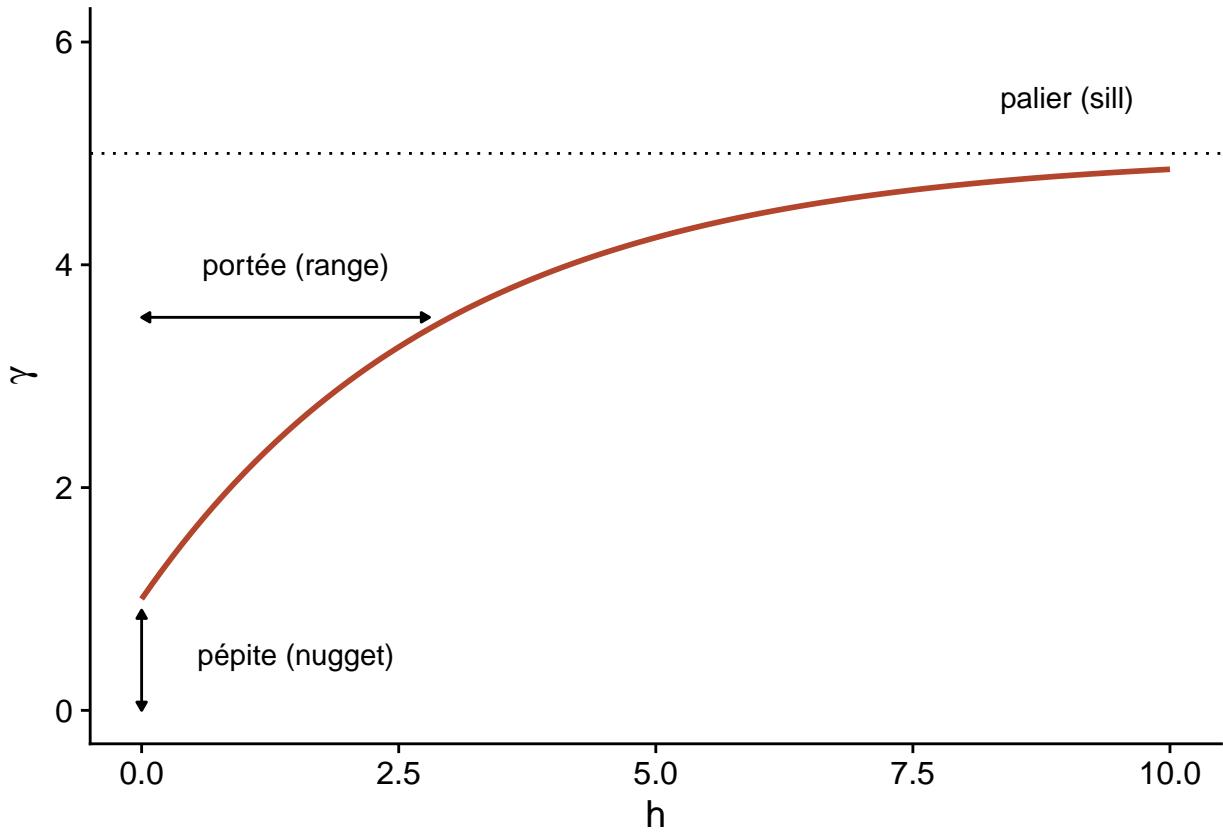


En raison de la fonction exponentielle, la valeur de γ à des grandes distances s'approche de la variance globale σ_z^2 sans exactement l'atteindre. Cette asymptote est appelée palier (*sill*) dans le contexte géostatistique et représentée par le symbole s .

Finalement, il n'est parfois pas réaliste de supposer une corrélation parfaite lorsque la distance est nulle, en raison d'une variation possible de z à très petite échelle. On peut ajouter au modèle un effet de pépite (*nugget*), noté n , pour que $\gamma = n$ si $h = 0$. Le terme pépite provient de l'origine minière de ces techniques, où une pépite d'un minerai pourrait être la source d'une variation abrupte de la concentration à petite échelle.

En ajoutant l'effet de pépite, le reste du variogramme est “compressé” pour conserver le même palier, ce qui résulte en l'équation suivante.

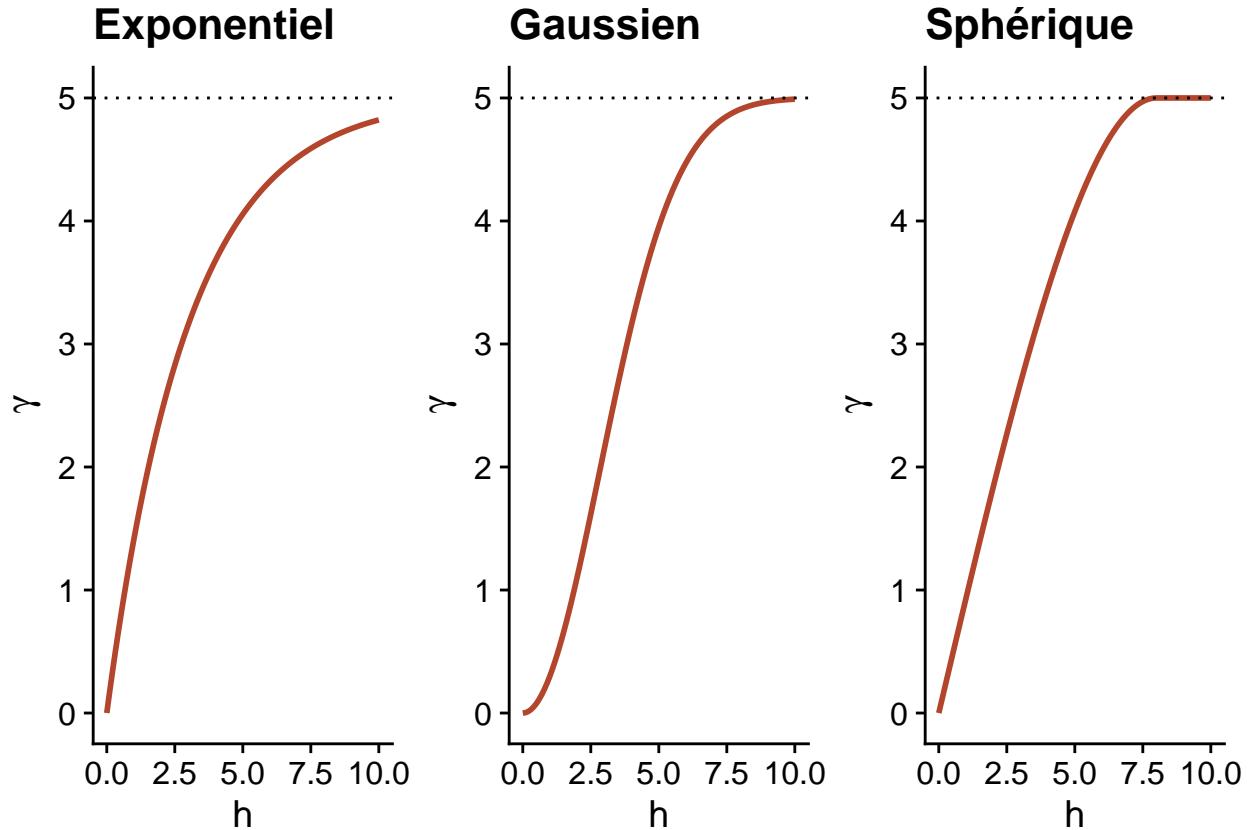
$$\gamma_z(h) = n + (s - n)(1 - e^{-h/r})$$



En plus du modèle exponentiel, deux autres modèles théoriques courants pour le variogramme sont le modèle gaussien (où la corrélation suit une courbe demi-normale), ainsi que le modèle sphérique (où le variogramme augmente de façon linéaire au départ pour ensuite courber et atteindre le palier à une distance égale à sa portée r). Le modèle sphérique permet donc à la corrélation d'être exactement 0 à grande distance, plutôt que de s'approcher graduellement de zéro dans le cas des autres modèles.

Modèle	$\rho(h)$	$\gamma(h)$
Exponentiel	$\exp\left(-\frac{h}{r}\right)$	$s\left(1 - \exp\left(-\frac{h}{r}\right)\right)$
Gaussien	$\exp\left(-\frac{h^2}{r^2}\right)$	$s\left(1 - \exp\left(-\frac{h^2}{r^2}\right)\right)$
Sphérique ($h < r$) *	$1 - \frac{3}{2} \frac{h}{r} + \frac{1}{2} \frac{h^3}{r^3}$	$s\left(\frac{3}{2} \frac{h}{r} - \frac{1}{2} \frac{h^3}{r^3}\right)$

* Pour le modèle sphérique, $\rho = 0$ et $\gamma = s$ si $h \geq r$.



Variogramme empirique

Pour estimer $\gamma_z(h)$ à partir de données empiriques, nous devons définir des classes de distance, donc grouper différentes distances dans une marge $\pm\delta$ autour d'une distance h , puis calculer l'écart-carré moyen pour les paires de points dans cette classe de distance.

$$\hat{\gamma}_z(h) = \frac{1}{2N_{\text{paire}}^z} \sum \left[(z(x_i, y_i) - z(x_j, y_j))^2 \right]_{d_{ij}=h \pm \delta}$$

Nous verrons dans la partie suivante comment estimer un variogramme dans R.

Variogramme et données temporelles

Un variogramme peut aussi être estimé en fonction des écarts dans le temps, qui est dans ce cas considéré comme un espace à 1 dimension. Ceci permet de modéliser la dépendance temporelle pour une série de mesures prises à intervalles irréguliers, lorsque les modèles autorégressifs vus au dernier cours ne s'appliquent pas.

Modèle de régression avec corrélation spatiale

L'équation suivante représente une régression linéaire multiple incluant une corrélation spatiale résiduelle:

$$v = \beta_0 + \sum_i \beta_i u_i + z + \epsilon$$

Ici, v désigne la variable réponse et u les prédicteurs, pour ne pas confondre avec les coordonnées spatiales x et y .

En plus du résidu ϵ qui est indépendant entre les observations, le modèle inclut un terme z qui représente la portion spatialement corrélée de la variance résiduelle.

Voici une suggestions d'étapes à suivre pour appliquer ce type de modèle:

1. Ajuster le modèle de régression sans corrélation spatiale.
2. Vérifier la présence de corrélation spatiale à partir du variogramme empirique des résidus.
3. Ajuster un ou plusieurs modèles de régression avec corrélation spatiale. On peut comparer avec l'AIC au besoin pour choisir la forme de la corrélation.

Nous verrons dans la dernière partie du cours comment inclure des termes de corrélation spatiale dans des modèles complexes, comme les modèles mixtes ou bayésiens.

Modèles géostatistiques dans R

Les modèles géostatistiques visent à représenter la distribution spatiale des variables continues qui sont mesurés à certains points échantillonnage. Ils supposent que les mesures de ces variables à différents points sont corrélées en fonction de la distance entre ces points. Parmi les applications des modèles géostatistiques, notons le lissage des données spatiales (ex.: produire une carte d'une variable sur ensemble d'une région en fonction des mesures ponctuelles) et la prédiction de ces variables pour des points non-échantillonés.

Le package *gstat* contient des fonctions liées à la géostatistique. Pour cet exemple, nous utiliserons le jeu de données *oxford* de ce package, qui contient des mesures de propriétés physiques et chimiques pour 126 échantillons du sol d'un site, ainsi que leurs coordonnées XCOORD et YCOORD.

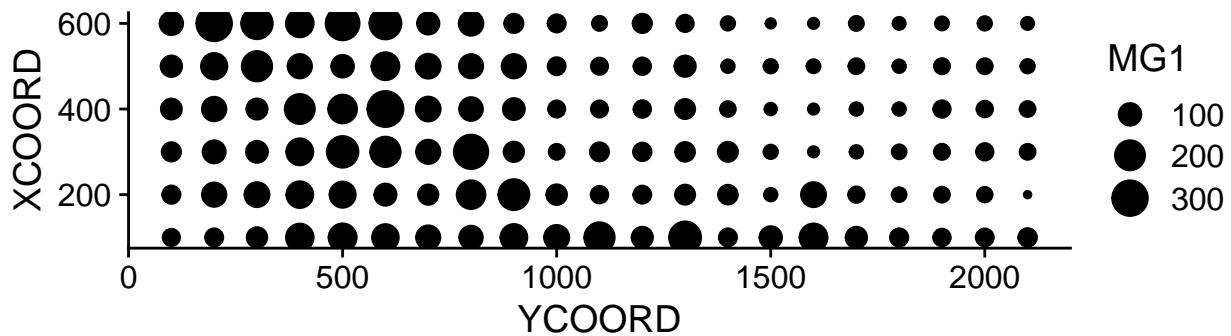
```
library(gstat)

data(oxford)
str(oxford)

## 'data.frame':   126 obs. of  22 variables:
## $ PROFILE  : num  1 2 3 4 5 6 7 8 9 10 ...
## $ XCOORD   : num  100 100 100 100 100 100 100 100 100 ...
## $ YCOORD   : num  2100 2000 1900 1800 1700 1600 1500 1400 1300 1200 ...
## $ ELEV     : num  598 597 610 615 610 595 580 590 598 588 ...
## $ PROFCCLASS: Factor w/ 3 levels "Cr","Ct","Ia": 2 2 2 3 3 2 3 2 3 3 ...
## $ MAPCLASS : Factor w/ 3 levels "Cr","Ct","Ia": 2 3 3 3 3 2 2 3 3 3 ...
## $ VAL1      : num  3 3 4 4 3 3 4 4 4 3 ...
## $ CHR1      : num  3 3 3 3 3 2 2 3 3 3 ...
## $ LIME1     : num  4 4 4 4 4 0 2 1 0 4 ...
## $ VAL2      : num  4 4 5 8 8 4 8 4 8 8 ...
## $ CHR2      : num  4 4 4 2 2 4 2 4 2 2 ...
## $ LIME2     : num  4 4 4 5 5 4 5 4 5 5 ...
## $ DEPTHCM  : num  61 91 46 20 20 91 30 61 38 25 ...
## $ DEP2LIME : num  20 20 20 20 20 20 20 20 40 20 ...
## $ PCLAY1   : num  15 25 20 20 18 25 25 35 35 12 ...
## $ PCLAY2   : num  10 10 20 10 10 20 10 20 10 10 ...
## $ MG1       : num  63 58 55 60 88 168 99 59 233 87 ...
## $ OM1       : num  5.7 5.6 5.8 6.2 8.4 6.4 7.1 3.8 5 9.2 ...
## $ CEC1      : num  20 22 17 23 27 27 21 14 27 20 ...
## $ PH1       : num  7.7 7.7 7.5 7.6 7.6 7 7.5 7.6 6.6 7.5 ...
## $ PHOS1    : num  13 9.2 10.5 8.8 13 9.3 10 9 15 12.6 ...
## $ POT1      : num  196 157 115 172 238 164 312 184 123 282 ...
```

Supposons que nous souhaitons modéliser la concentration de magnésium (MG1), représentée en fonction de la position spatiale dans le graphique suivant.

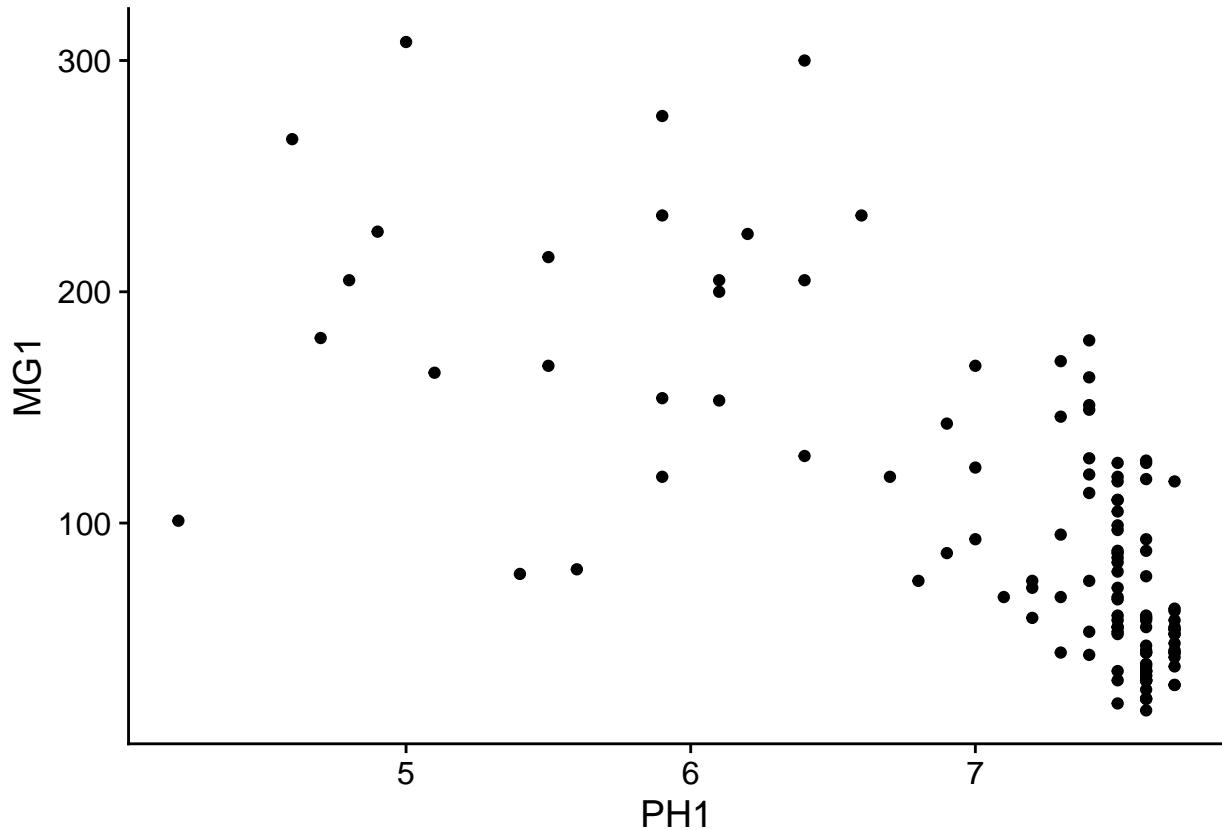
```
library(ggplot2)
ggplot(oxford, aes(x = YCOORD, y = XCOORD, size = MG1)) +
  geom_point() +
  coord_fixed()
```



Notez que les axes x et y ont été inversés par souci d'espace. La fonction `coord_fixed()` de `ggplot2` assure que l'échelle soit la même sur les deux axes, ce qui est utile pour représenter des données spatiales.

Nous voyons tout de suite que ces mesures ont été prises sur une grille de 100 m de côté. Il semble que la concentration de magnésium soit spatialement corrélée, bien qu'il puisse s'agir d'une corrélation induite par une autre variable. Nous savons notamment que la disponibilité de l'ion magnésium est reliée (négativement) au pH du sol (PH1).

```
ggplot(oxford, aes(x = PH1, y = MG1)) +
  geom_point()
```



La fonction `variogram` de `gstat` sert à estimer un variogramme à partir de données empiriques. Voici le résultat obtenu pour la variable MG1.

```
require(gstat)

var_mg <- variogram(MG1 ~ 1, locations = ~ XCOORD + YCOORD, data = oxford)
var_mg

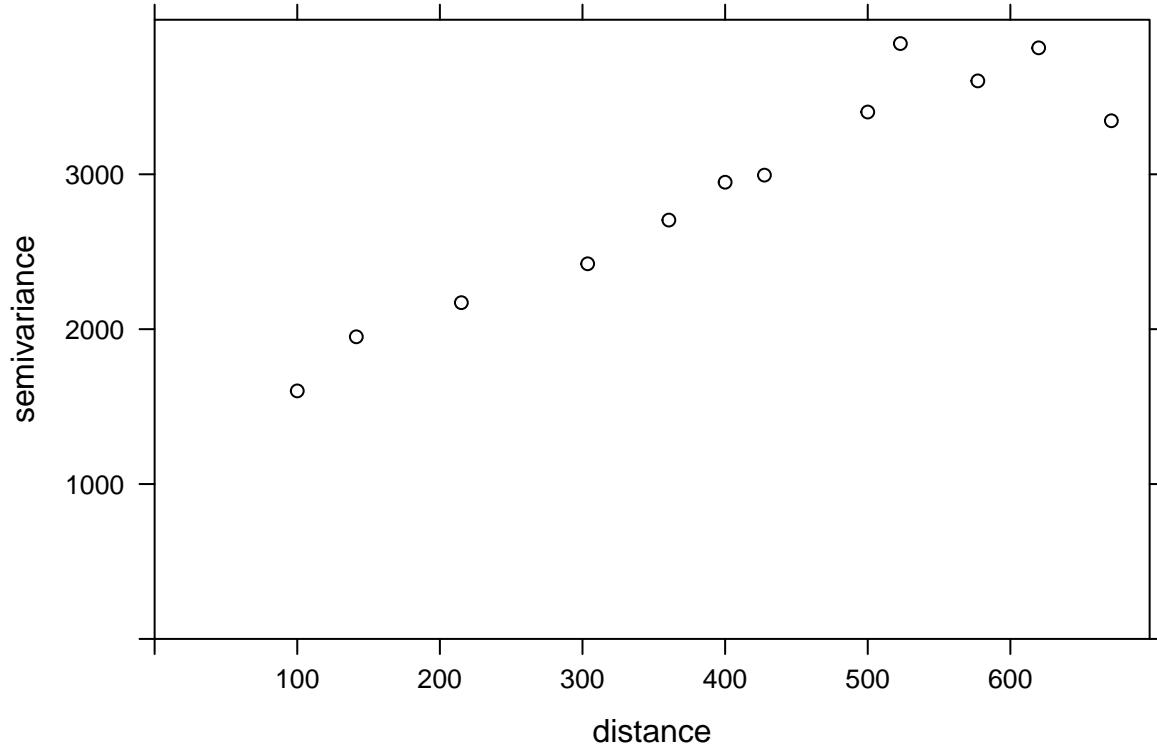
##      np      dist     gamma dir.hor dir.ver   id
## 1 225 100.0000 1601.404      0      0 var1
## 2 200 141.4214 1950.805      0      0 var1
## 3 548 215.0773 2171.231      0      0 var1
## 4 623 303.6283 2422.245      0      0 var1
## 5 258 360.5551 2704.366      0      0 var1
## 6 144 400.0000 2948.774      0      0 var1
## 7 570 427.5569 2994.621      0      0 var1
## 8 291 500.0000 3402.058      0      0 var1
## 9 366 522.8801 3844.165      0      0 var1
## 10 200 577.1759 3603.060      0      0 var1
## 11 458 619.8400 3816.595      0      0 var1
## 12 90 670.8204 3345.739      0      0 var1
```

La formule `MG1 ~ 1` indique qu'aucun prédicteur linéaire n'est inclus dans ce modèle, tandis que l'argument `locations` indique quelles variables du tableau correspondent aux coordonnées spatiales.

Dans le tableau obtenu, `gamma` est la valeur du variogramme pour la classe de distance centrée sur `dist`, tandis que `np` est le nombre de paires de points dans cette classe. Ici, puisque les points sont situés sur une grille, nous obtenons des classes de distance régulières (ex.: 100 m pour les points voisins sur la grille, 141 m

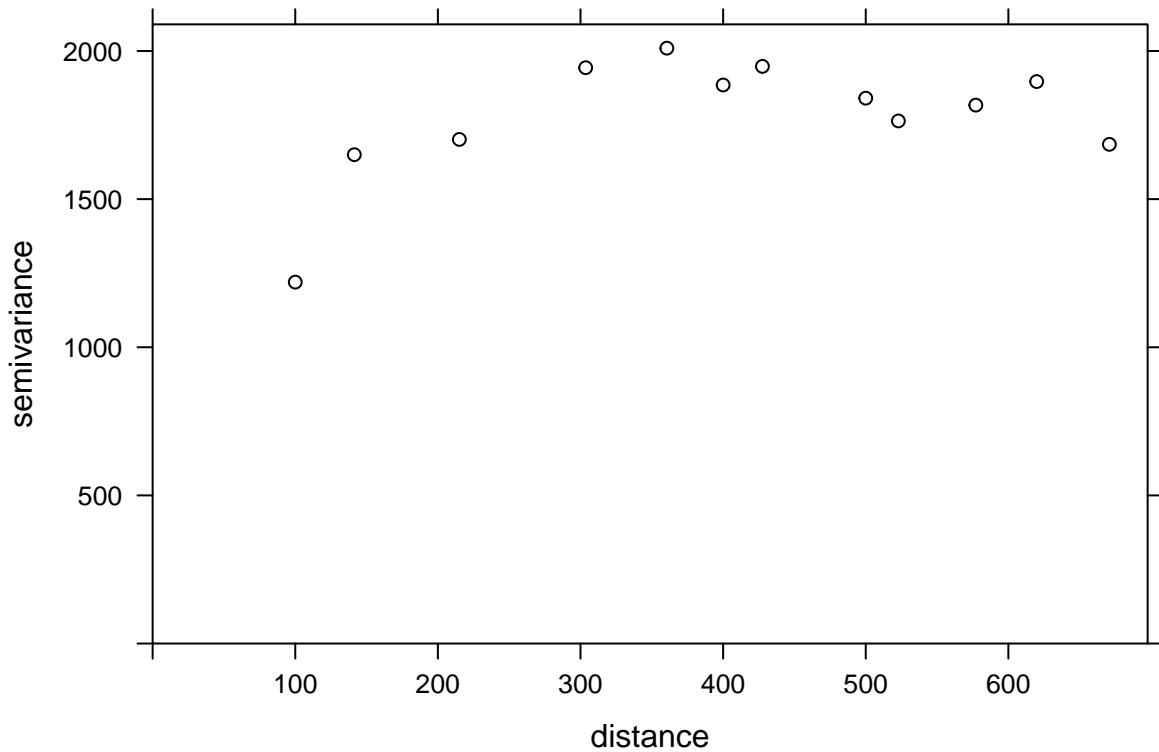
pour les voisins en diagonale, etc.) Nous pouvons illustrer le variogramme avec `plot`.

```
plot(var_mg, col = "black")
```



Si nous voulons estimer la corrélation spatiale résiduelle de MG1 après avoir inclus l'effet de PH1, nous pouvons ajouter ce prédicteur à la formule.

```
var_mg <- variogram(MG1 ~ PH1, locations = ~ XCOORD + YCOORD, data = oxford)
plot(var_mg, col = "black")
```



En incluant l'effet du pH, la portée de la corrélation spatiale semble diminuer, alors que le plateau est atteint autour de 300 m. Il semble même que le variogramme diminue au-delà de 400 m. En général, nous supposons que la variance entre deux points ne diminue pas avec la distance, à moins d'avoir un patron spatial périodique.

La fonction `fit.variogram` accepte comme arguments un variogramme estimé à partir des données, ainsi qu'un modèle théorique décrit dans une fonction `vgm`, puis estime les paramètres de ce modèle en fonction des données. L'ajustement se fait par la méthode des moindres carrés.

Par exemple, `vgm("Exp")` indique d'ajuster un modèle exponentiel.

```
vfit <- fit.variogram(var_mg, vgm("Exp"))
vfit
```

```
##   model    psill     range
## 1   Nug    0.000  0.00000
## 2   Exp 1951.496 95.11235
```

Il n'y a aucun effet de pépite, car `psill = 0` pour la partie `Nug` (*nugget*) du modèle. La partie exponentielle a un palier à 1951 (correspondant à σ_z^2) et une portée de 95 m.

Pour comparer différents modèles, on peut donner un vecteur de noms de modèles à `vgm`. Dans l'exemple suivant, nous incluons les modèles exponentiel, gaussien ("Gau") et sphérique ("Sph").

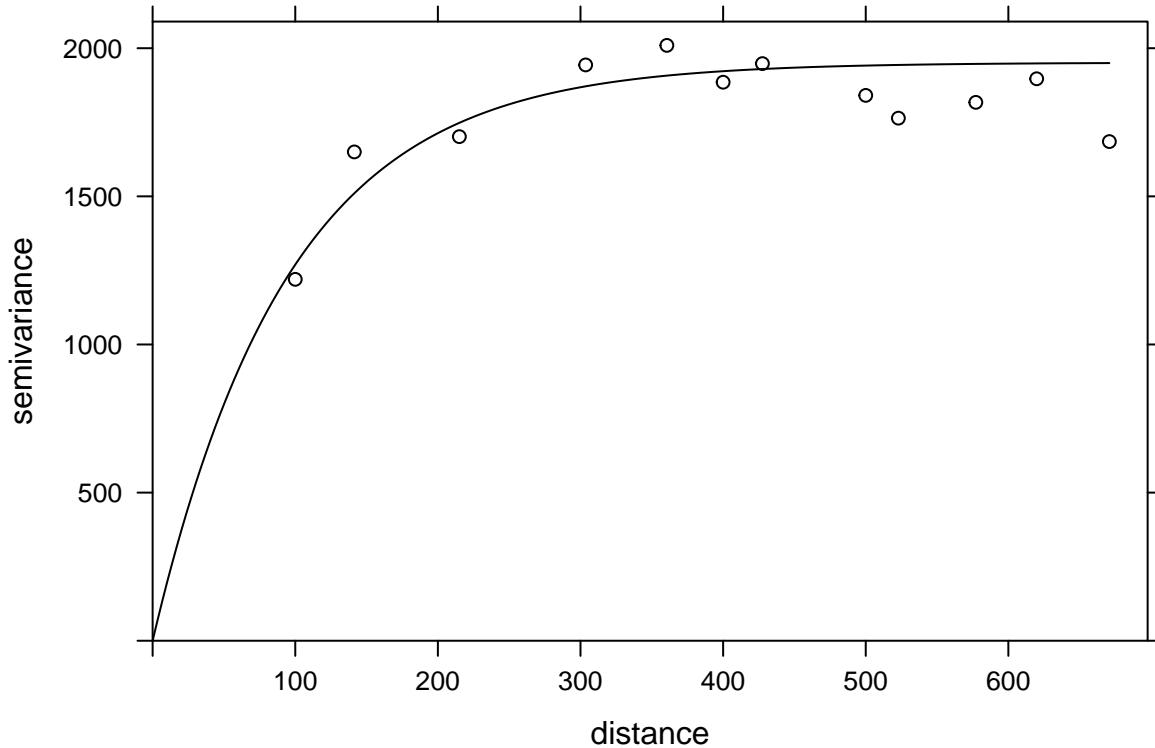
```
vfit <- fit.variogram(var_mg, vgm(c("Exp", "Gau", "Sph")))
vfit
```

```
##   model    psill     range
## 1   Nug    0.000  0.00000
## 2   Exp 1951.496 95.11235
```

Le modèle exponentiel demeure le mieux ajusté.

Finalement, nous pouvons superposer le modèle théorique et le variogramme empirique sur un même graphique.

```
plot(var_mg, vfit, col = "black")
```



Nota bene: le variogramme peut fournir des informations pour aider à choisir la distance à laquelle considérer les points voisins dans le calcul de l'indice de Moran, en fonction de la structure de corrélation spatiale des données. Par exemple, dans des contextes où la corrélation spatiale décroît lentement avec la distance ou où des motifs spatiaux complexes sont présents, vous pouvez vous attendre à avoir besoin d'un k plus grand pour capturer adéquatement la structure de dépendance spatiale des données. Cependant le choix du k dépend aussi de la taille de l'échantillon (k plus grand pour tenir compte d'un plus grand nombre de voisins et obtenir une représentation plus robuste de la corrélation spatiale); objectif spécifique de l'analyse e.g. tendances globales, des structures en patchs ou des gradients spatiaux, vous pouvez avoir besoin d'un k plus grand pour capturer ces motifs. Si vous êtes intéressé par une analyse plus locale (k petit) ou globale (k grand) de la dépendance spatiale, cela influencera le nombre de voisins à considérer.

Modèles d'autorégression spatiale

Rappelons-nous la formule pour une régression linéaire avec dépendance spatiale:

$$v = \beta_0 + \sum_i \beta_i u_i + z + \epsilon$$

,

où z est la portion de la variance résiduelle qui est spatialement corrélée.

Il existe deux principaux types de modèles autorégressifs pour représenter la dépendance spatiale de z : l'autorégression conditionnelle (CAR) et l'autorégression simultanée (SAR).

Autorégression simultanée (SAR)

Les modèles SAR considèrent que les valeurs des données à des emplacements voisins ont une relation linéaire entre elles. En d'autres termes, la valeur d'une observation est influencée de manière linéaire par les valeurs de ses voisins. Ces modèles sont souvent mis en œuvre en utilisant des matrices qui montrent comment les observations spatiales sont liées les unes aux autres. Dans le modèle d'autorégression simultanée, la valeur de z_i est donnée directement par la somme de contributions des valeurs voisines z_j , multipliées par ρw_{ij} , avec un résidu indépendant ν_i d'écart-type σ_z .

$$z_i = \sum_j \rho w_{ij} z_j + \nu_i$$

À première vue, cela ressemble à un modèle autorégressif temporel. Il existe cependant une différence conceptuelle importante. Pour les modèles temporels, l'influence causale est dirigée dans une seule direction: $v(t - 2)$ affecte $v(t - 1)$ qui affecte ensuite $v(t)$. Pour un modèle spatial, chaque z_j qui affecte z_i dépend à son tour de z_i . Ainsi, pour déterminer la distribution conjointe des z , il faut résoudre simultanément (d'où le nom du modèle) un système d'équations.

Pour cette raison, même si ce modèle ressemble à la formule du modèle conditionnel (CAR), les solutions des deux modèles diffèrent et dans le cas du SAR, le coefficient ρ n'est pas directement égal à la corrélation partielle due à chaque région voisine.

Pour plus de détails sur les aspects mathématiques de ces modèles, vous pouvez consulter l'article de Ver Hoef et al. (2018) suggéré dans les références.

Pour l'instant, nous considérerons les SAR et les CAR comme deux types de modèles possibles pour représenter une corrélation spatiale sur un réseau. Nous pouvons toujours ajuster plusieurs modèles et les comparer avec l'AIC pour choisir la meilleure forme de la corrélation ou la meilleure matrice de poids.

Les modèles CAR et SAR partagent un avantage sur les modèles géostatistiques au niveau de l'efficacité. Dans un modèle géostatistique, les corrélations spatiales sont définies entre chaque paire de points, même si elles deviennent négligeables lorsque la distance augmente. Pour un modèle CAR ou SAR, seules les régions voisines contribuent et la plupart des poids sont égaux à 0, ce qui rend ces modèles plus rapides à ajuster qu'un modèle géostatistique lorsque les données sont massives.

Notez finalement qu'il existe aussi un équivalent spatial des modèles de moyenne mobile (MA) vus dans un contexte temporel. Cependant, puisque leur application est plus rare, nous n'en discutons pas dans ce cours.

Autorégression conditionnelle (CAR)

Les modèles CAR considèrent une dépendance spatiale conditionnelle, où la valeur d'une observation est influencée par une moyenne pondérée des valeurs de ses voisins, sans qu'une relation linéaire stricte entre elles ne soit spécifiée. Ils intègrent un paramètre de précision qui régule l'intensité de la dépendance spatiale, en décrivant la manière dont les observations sont corrélées les unes avec les autres.

Dans le modèle d'autorégression conditionnelle, la valeur de z_i pour la région i suit une distribution normale: sa moyenne dépend de la valeur z_j des régions voisines, multipliée par le poids w_{ij} et un coefficient de corrélation ρ ; son écart-type σ_{z_i} peut varier d'une région à l'autre.

$$z_i \sim N \left(\sum_j \rho w_{ij} z_j, \sigma_{z_i} \right)$$

Dans ce modèle, si w_{ij} est une matrice binaire (0 pour les non-voisins, 1 pour les voisins), alors ρ est le coefficient de corrélation partielle entre régions voisines. Cela est semblable au modèle AR(1) dans le contexte temporel, où le coefficient d'autorégression indiquait la corrélation partielle.

Données aréales dans R

Les données aréales sont des mesures prises non pas à des points, mais pour des régions de l'espace représentées par des polygones (ex.: divisions du territoire, cellules d'une grille). Les modèles représentant ces types de données définissent un réseau de voisinage reliant les régions et incluent une corrélation spatiale entre régions voisines

Pour illustrer l'analyse de données aréales dans R, nous chargeons les packages *spData* (contenant des exemples de données spatiales), *spdep* (pour définir des réseaux spatiaux et calculer l'indice de Moran) et *spatialreg* (pour les modèles SAR et CAR).

```
library(spData)
library(spdep)
library(spatialreg)
```

Nous utiliserons comme exemple le jeu de données spatial **us_states** qui contient des polygones pour 49 états américains (tous les états excluant l'Alaska et Hawaii, plus le District de Columbia).

```
data(us_states)
head(us_states)
```

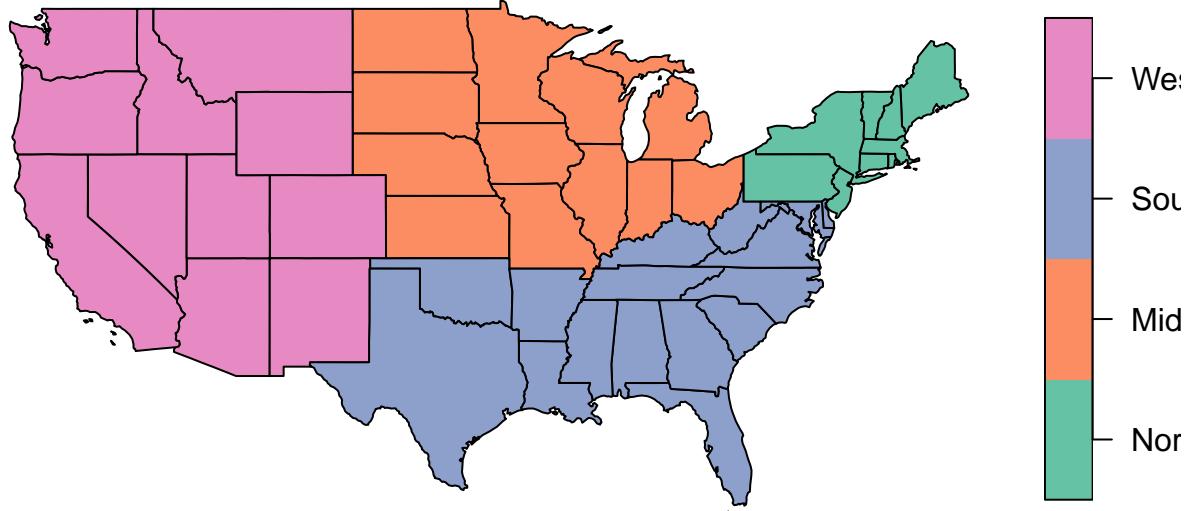
```
## Simple feature collection with 6 features and 6 fields
## Geometry type: MULTIPOLYGON
## Dimension:      XY
## Bounding box:  xmin: -114.8136 ymin: 24.55868 xmax: -71.78699 ymax: 42.04964
## Geodetic CRS:  NAD83
##   GEOID      NAME    REGION          AREA total_pop_10 total_pop_15
## 1  01    Alabama    South 133709.27 [km^2]     4712651     4830620
## 2  04    Arizona     West 295281.25 [km^2]     6246816     6641928
## 3  08    Colorado     West 269573.06 [km^2]     4887061     5278906
## 4  09 Connecticut  Northeast 12976.59 [km^2]     3545837     3593222
## 5  12    Florida     South 151052.01 [km^2]     18511620    19645772
## 6  13    Georgia     South 152725.21 [km^2]     9468815    10006693
##
##           geometry
## 1 MULTIPOLYGON (((-88.20006 3...
## 2 MULTIPOLYGON (((-114.7196 3...
## 3 MULTIPOLYGON (((-109.0501 4...
## 4 MULTIPOLYGON (((-73.48731 4...
## 5 MULTIPOLYGON (((-81.81169 2...
## 6 MULTIPOLYGON (((-85.60516 3...
```

Il s'agit d'un tableau de données spatial dont la dernière colonne définit le polygone correspondant à l'état et les autres colonnes définissent des variables qui y sont associées. Nous ne discuterons pas en détail de cette structure de données, mais notez que le packge *sf* permet d'importer des fichiers SIG vectoriels (*shapefiles*) dans ce format de données pour R.

Pour illustrer une des variables sur une carte, nous appelons la fonction **plot** avec le nom de la colonne entre crochets et guillemets.

```
plot(us_states["REGION"])
```

REGION



Nous voulons ici modéliser le revenu médian dans chaque état en 2015. Cette variable `median_income_15` se trouve dans un autre jeu de données, `us_states_df`.

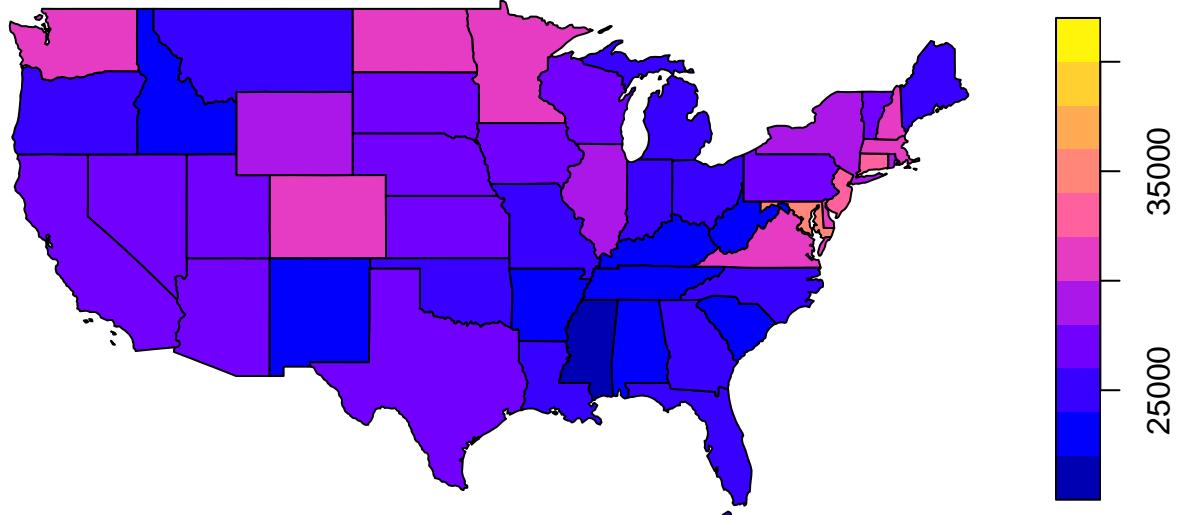
```
data(us_states_df)
head(us_states_df)

## # A tibble: 6 x 5
##   state      median_income_10 median_income_15 poverty_level_10 poverty_level_15
##   <chr>          <dbl>           <dbl>          <dbl>           <dbl>
## 1 Alabama        21746          22890          786544         887260
## 2 Alaska         29509          31455          64245           72957
## 3 Arizona        26412          26156          933113         1180690
## 4 Arkansas       20881          22205          502684          553644
## 5 California     27207          27035          4919945        6135142
## 6 Colorado       29365          30752          584184          653969
```

Nous utilisons la fonction `inner_join` de `dplyr` pour joindre les deux jeux de données, en spécifiant avec `by` que la colonne `NAME` de `us_states` correspond à la colonne `state` de `us_states_df`.

```
library(dplyr)
us_states <- inner_join(us_states, us_states_df,
                        by = c("NAME" = "state"))
plot(us_states[["median_income_15"]])
```

median_income_15



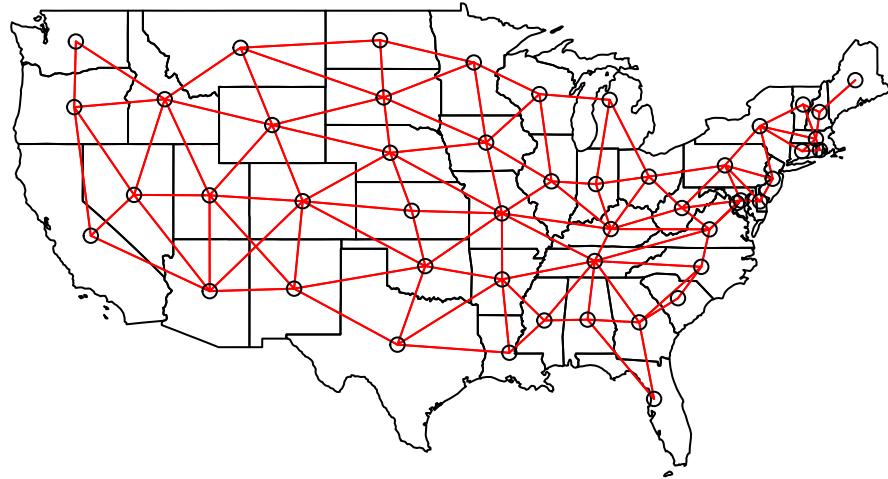
La fonction `poly2nb` du package `spdep` définit un réseau de voisinage à partir de polygones. Le résultat `vois` est une liste de 49 éléments où chaque élément contient les indices des polygones voisins d'un polygone donné.

```
vois <- poly2nb(us_states)
vois[[1]]
```

```
## [1] 5 6 36 44
```

Nous pouvons illustrer ce réseau en faisant l'extraction des coordonnées du centre de chaque état, en créant une carte muette avec `plot(us_states["geometry"])`, puis en ajoutant le réseau comme couche additionnelle avec `plot(vois, add = TRUE, coords = coords)`.

```
coords <- st_centroid(us_states) %>%
  st_coordinates()
plot(us_states["geometry"])
plot(vois, add = TRUE, col = "red", coords = coords)
```



Il nous reste à ajouter des poids à chaque lien du réseau avec la fonction `nb2listw`. Le style de poids “B” correspond aux poids binaires, soit 1 pour la présence de lien et 0 pour l’absence de lien entre deux états.

Une fois ces poids définis, nous pouvons vérifier s’il y a une autocorrélation significative du revenu médian entre états voisins, avec le test de Moran.

```
poids <- nb2listw(vois, style = "B")

moran.test(us_states$median_income_15, poids)

##
##  Moran I test under randomisation
##
## data: us_states$median_income_15
## weights: poids
##
## Moran I statistic standard deviate = 4.127, p-value = 1.838e-05
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic      Expectation      Variance
##          0.342670652     -0.020833333    0.007758162
```

La valeur de $I = 0.34$ est très significative à en juger par la valeur p du test.

Finalement, nous ajustons des modèles SAR et CAR à ces données avec la fonction `spautolm` (*spatial autoregressive linear model*) de `spatialreg`. Voici le code pour un modèle SAR incluant l’effet fixe de la région (ouest, mid-ouest, sud ou nord-est).

```

modsp <- spautolm(median_income_15 ~ REGION, data = us_states,
                   listw = poids)
summary(modsp)

##
## Call: spautolm(formula = median_income_15 ~ REGION, data = us_states,
##                 listw = poids)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -5632.95 -2243.24  -856.84  1781.90 11770.13
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 28703.411   1607.825 17.8523 <2e-16
## REGIONMidwest    42.832   2157.389  0.0199  0.9842
## REGIONSouth   -1131.312   1904.512 -0.5940  0.5525
## REGIONWest     -815.428   2393.545 -0.3407  0.7333
##
## Lambda: 0.12915 LR test value: 7.7579 p-value: 0.0053479
## Numerical Hessian standard error of lambda: 0.033239
##
## Log likelihood: -466.35
## ML residual variance (sigma squared): 9804100, (sigma: 3131.2)
## Number of observations: 49
## Number of parameters estimated: 6
## AIC: 944.7

```

La valeur donnée par `Lambda` dans le sommaire correspond au coefficient ρ dans notre description du modèle. Le test du rapport de vraisemblance (`LR test`) confirme que cette corrélation spatiale résiduelle (après avoir tenu compte de l'effet de la région) est significative.

Pour évaluer un modèle CAR plutôt que SAR, nous devons spécifier `family = "CAR"`.

```

modsp2 <- spautolm(median_income_15 ~ REGION, data = us_states,
                     listw = poids, family = "CAR")
summary(modsp2)

##
## Call: spautolm(formula = median_income_15 ~ REGION, data = us_states,
##                 listw = poids, family = "CAR")
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -5709.20 -1999.90  -682.38  2072.01 11328.25
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 29022.2     1484.4 19.5519 <2e-16
## REGIONMidwest   -408.0     2049.4 -0.1991  0.8422
## REGIONSouth    -1436.2     1820.6 -0.7888  0.4302
## REGIONWest     -1378.0     2239.7 -0.6153  0.5384
##
## Lambda: 0.16539 LR test value: 5.8956 p-value: 0.015179
## Numerical Hessian standard error of lambda: 0.031414

```

```

## 
## Log likelihood: -467.2811
## ML residual variance (sigma squared): 10168000, (sigma: 3188.7)
## Number of observations: 49
## Number of parameters estimated: 6
## AIC: 946.56

```

Pour un modèle CAR avec des poids binaires, la valeur de `Lambda` (que nous avions appelé ρ) donne directement le coefficient de corrélation partiel entre états voisins. Notez que l'AIC ici est légèrement supérieur au modèle SAR, donc ce dernier donnait un meilleur ajustement.

Corrélation spatiale dans des modèles complexes

Comme pour le cours sur les séries temporelles, nous terminons ce cours en présentant quelques pistes pour intégrer les corrélations spatiales dans des modèles plus complexes.

Modèles géostatistiques avec `nlme`

Au dernier cours, nous avons vu que la fonction `lme` du package `nlme` permettait d'inclure des corrélations temporelles avec des termes de type `corARMA`. Le même package contient des fonctions de corrélation spatiale, incluant une corrélation exponentielle (`corExp`), gaussienne (`corGaus`) et sphérique (`corSpher`).

Voici un exemple de modèle linéaire mixte ajusté avec `lme`, où `v` est la réponse, `u` est un effet fixe et `groupe` est un effet aléatoire. L'argument `correlation` indique une corrélation exponentielle en fonction de la distance définie par les coordonnées `x` et `y`, avec un effet de pépite `nugget = TRUE`.

```

library(nlme)
mod <- lme(v ~ u, data, random = list(groupe = ~1),
            correlation = corExp(form = ~ x + y, nugget = TRUE))

```

Notez que les limites du package `nlme` mentionnées au dernier cours s'appliquent encore ici: il n'est pas possible d'inclure plusieurs effets aléatoires croisés (non-nichés) et le package n'est pas très efficace pour l'estimation de modèles généralisés.

Pour intégrer une corrélation spatiale à un modèle linéaire, sans effet aléatoire, nous pouvons remplacer `lme` par `gls`, pour *generalized least squares*. Cette fonction est semblable à `lm`, mais permet des corrélations entre les résidus du modèle.

```

library(nlme)
mod <- gls(v ~ u, data,
            correlation = corExp(form = ~ x + y, nugget = TRUE))

```

Finalement, comme nous avons vu au dernier cours, la fonction `gamm` du package `mgcv` combine les fonctionnalités de `lme` avec la possibilité d'inclure des effets additifs (splines de lissage) pour les prédicteurs.

```

library(mgcv)
mod <- gamm(v ~ s(u), data, random = list(groupe = ~1),
            correlation = corExp(form = ~ x + y, nugget = TRUE))

```

Modèles géostatistiques avec `brms`

Pour inclure une corrélation spatiale de type géostatistique dans un modèle bayésien estimé avec `brms`, nous devons spécifier un terme `gp`, qui décrit un processus gaussien.

```

library(brms)
mod <- brm(v ~ u + gp(x, y, cov = "exp_quad"), data)

```

Le terme `gp` indique les variables contenant les coordonnées spatiales (`x, y`) ainsi que la forme de la covariance. Actuellement, seule la corrélation gaussienne (`exp_quad`, pour *exponential quadratic*) est disponible.*

* Des processus gaussiens avec d'autres fonctions de corrélation sont possibles, s'ils sont codés manuellement avec Stan. Le terme "gaussien" dans "processus gaussien" réfère à la distribution normale des erreurs, pas à la forme de la corrélation spatiale.

Modèles autorégressifs spatiaux avec `brms`

D'autre part, la fonction `brm` permet de spécifier une structure autorégressive spatiale avec les termes `sar` et `car`, ce qui est utile pour combiner un modèle autorégressif spatial avec des effets aléatoires non-spatiaux. Les termes `sar` et `car` sont seulement permis dans les modèles où la réponse suit une distribution normale ou t , donc on ne peut pas les combiner à des modèles généralisés.

```
library(brms)
mod_sar <- brm(v ~ u + sar(W, type = "error"), data, data2 = list(W = W))
mod_car <- brm(v ~ u + car(W), data, data2 = list(W = W))
```

- `W` est la matrice de poids. Puisque cette matrice ne fait pas partie des données `data`, elle est donnée séparément dans l'argument `data2`.
- L'argument `type = "error"` dans `sar` représente le type de modèle SAR vu dans ce cours, où la portion non-expliquée de la réponse est autocorrélée. Il existe d'autres types de SAR, notamment ceux où la valeur de la réponse elle-même est autocorrélée.

Références

Ce cours a été modifié à partir du cours de Philippe Marchand qui a été donnée le 12 janvier 2021. Le cours originale est encore disponible dans le site <https://bios2.github.io/posts/2021-01-12-4-day-training-in-spatial-statistics-with-philippe-marchand/#statistiques-spatiales-en-%C3%A9cologie>

Ce cours ne constitue qu'une brève introduction aux principales techniques d'analyse spatiale utiles en sciences de l'environnement. Pour aller plus loin dans ce domaine, le manuel de Fortin et Dale donne un portrait très complet de ces méthodes et d'autres.

Fortin, M.-J. et Dale, M.R.T. (2005) *Spatial Analysis: A Guide for Ecologists*. Cambridge University Press: Cambridge, UK.

Ver Hoef, J.M., Peterson, E.E., Hooten, M.B., Hanks, E.M. et Fortin, M.-J. (2018) Spatial autoregressive models for statistical inference from ecological data. *Ecological Monographs* 88: 36-59.

Wiegand, T. et Moloney, K.A. (2013) *Handbook of Spatial Point-Pattern Analysis in Ecology*, CRC Press.