Quelques notions de statistique spatiale et bayésienne

Atelier INLA/inlabru, Avignon avril 2023

Les grands domaines de la statistique spatiale

1. **Données continues irrégulièrement espacées**. Ce sont les données de type géostatistique. Elles peuvent être uni- ou multi-variées. Les données sont en général situées sur une partie du plan ou de l'espace à trois dimensions.

2. **Données sur lattice**. Les modèles les plus utilisés sont les champs Markoviens.

3. **Processus ponctuels**. Les points indiquent seulement la présence d'un événement. Les processus ponctuels peuvent être généralisés aux processus ponctuels marqués, lorsqu'une valeur (une marque) est associée aux points.

Les grands domaines de la statistique spatiale

1. **Données continues irrégulièrement espacées**. Ce sont les données de type géostatistique. Elles peuvent être uni- ou multi-variées. Les données sont en général situées sur une partie du plan ou de l'espace à trois dimensions.

2. **Données sur lattice**. Les modèles les plus utilisés sont les champs Markoviens.

3. **Processus ponctuels**. Les points indiquent seulement la présence d'un événement. Les processus ponctuels peuvent être généralisés aux processus ponctuels marqués, lorsqu'une valeur (une marque) est associée aux points.

Les grands domaines de la statistique spatiale

1. **Données continues irrégulièrement espacées**. Ce sont les données de type géostatistique. Elles peuvent être uni- ou multi-variées. Les données sont en général situées sur une partie du plan ou de l'espace à trois dimensions.

2. **Données sur lattice**. Les modèles les plus utilisés sont les champs Markoviens.

3. **Processus ponctuels**. Les points indiquent seulement la présence d'un événement. Les processus ponctuels peuvent être généralisés aux processus ponctuels marqués, lorsqu'une valeur (une marque) est associée aux points.

Les grands domaines de la statistique spatiale

1. **Données continues irrégulièrement espacées**. Ce sont les données de type géostatistique. Elles peuvent être uni- ou multi-variées. Les données sont en général situées sur une partie du plan ou de l'espace à trois dimensions.

2. **Données sur lattice**. Les modèles les plus utilisés sont les champs Markoviens.

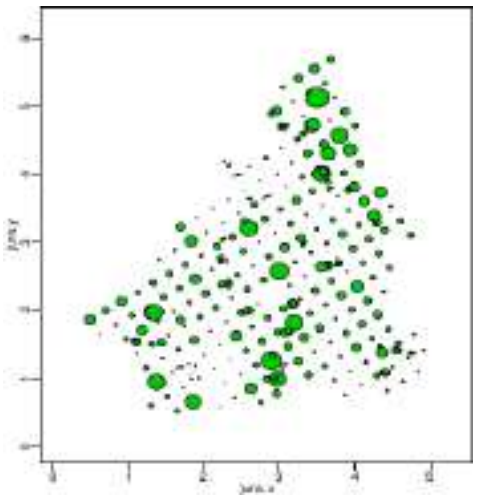
3. **Processus ponctuels**. Les points indiquent seulement la présence d'un événement. Les processus ponctuels peuvent être généralisés aux processus ponctuels marqués, lorsqu'une valeur (une marque) est associée aux points.

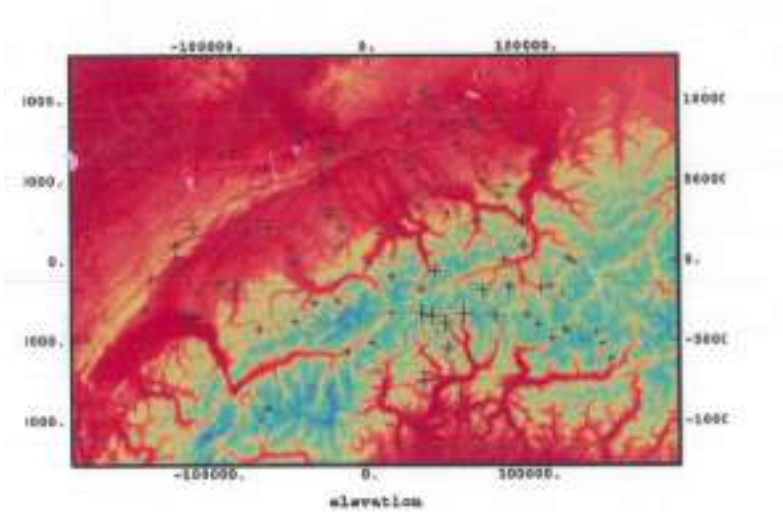
Données géostatistiques

Exemples : Température, précipitation, qualité d'un sol, pollution atmosphérique...

Questions typiques :

- Caractériser la variabilité spatiale,
- Interpoler (cartographier) la variable entre les points mesurés,
- Simuler des variations spatiales du même type,
- Evaluer l'erreur d'interpolation et la qualité de l'échantillonnage.



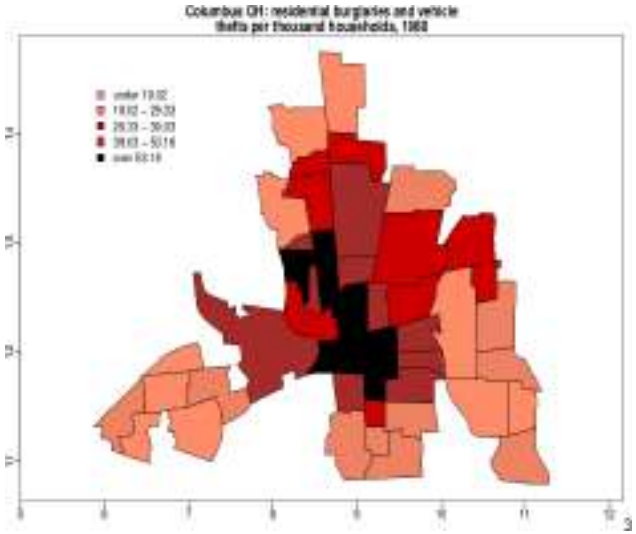


Données sur réseaux

Exemples : Données de population, données épidémiologiques sur un ensemble d'entités administratives...

Questions typiques :

- Caractériser la variabilité spatiale (indépendance entre voisins),
- Expliquer la distribution des caractéristiques en fonction des distributions dans un voisinage,
- Simuler des distributions spatiales du même type.

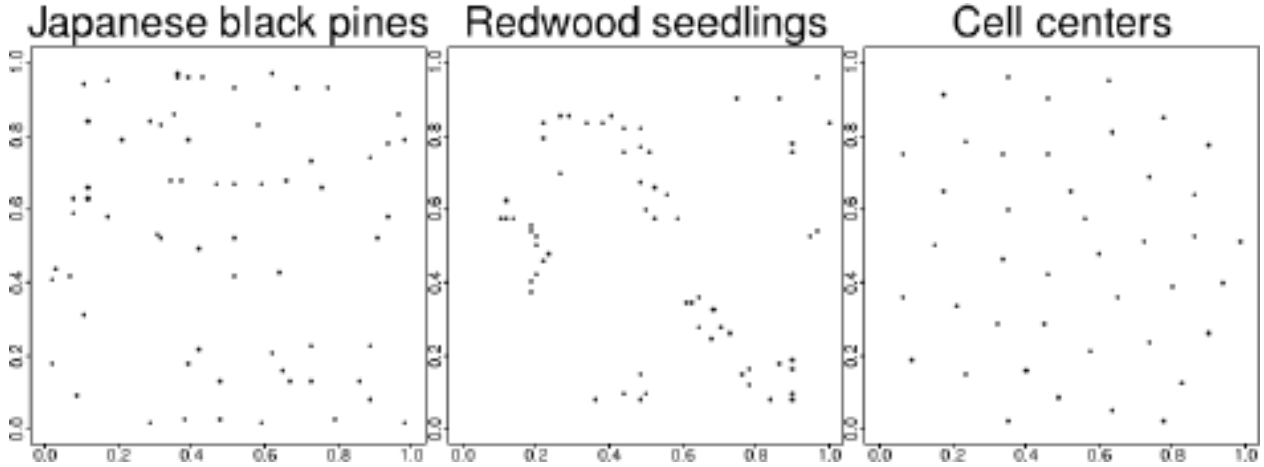


Processus ponctuels ou processus d'objets

Exemples : Répartition d'espèces végétales, d'animaux, cas d'une maladie...

Questions typiques :

- Caractériser la distribution spatiale des objets : indépendance, régularité, agrégation ?
- Expliquer la distribution des caractéristiques des objets en fonction de leurs positionnements relatifs,
- Simuler des distributions spatiales du même type.



Spécificités/difficultés des statistiques spatiales

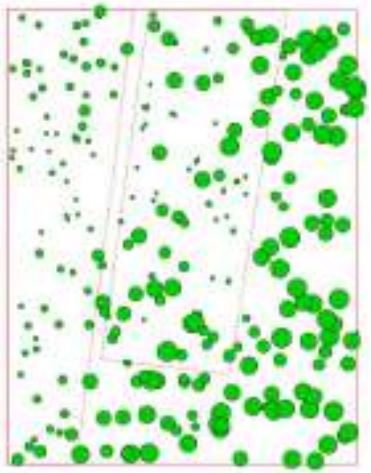
- Données non indépendantes.
- Pas de relation d'ordre dans les espaces $d \geq 2$.
- Vraisemblance adaptée ? Calculable ?
- Parfois : effets de bords, dépendances fortes,...

La géostatistique classique

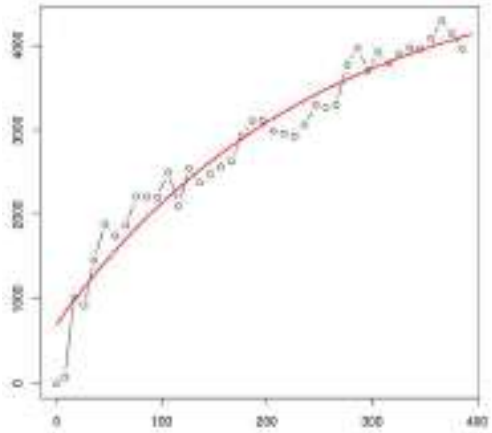
1. Réalisation unique et un suffisamment grand nombre de données résultant d'un échantillonnage du domaine d'étude,
2. Analyse structurale ou variographique (choix d'un modèle de variabilité spatiale et estimation des paramètres de cette variabilité sur un jeu de données),
3. Interpolation de la variable en tout point par Krigeage.

La géostatistique classique (en image)

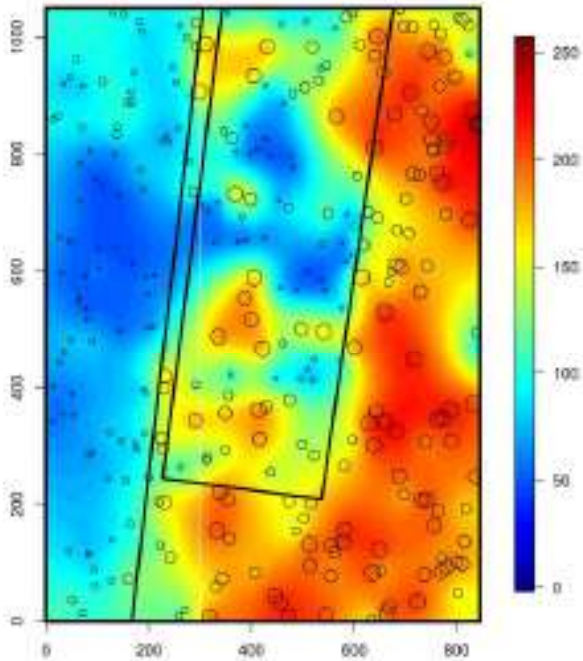
1. Echantillonnage



2. Modélisation



3. Interpolation



Les champs aléatoires

Modèles

Résumé

Pour atteindre les objectifs :

- **Décrire** la variabilité spatiale,
- **Interpoler** (cartographier) le mieux possible,
- **Evaluer l'erreur** d'interpolation.

Il faudra un modèle :

- Un **champ aléatoire** $Z(x)$ où $x \in D$ est un point du domaine d'étude et chaque $Z(x)$ une variable aléatoire.

Et quelques hypothèses du fait de la réalisation unique :

- **Stationnarité** d'ordre 2,
- **Ergodicité**.

Modèles

Hypothèses de la géostatistique

1. **Stationnarité :** la loi est invariante par translation. Dans la plupart des cas il est suffisant de supposer la stationnarité d'ordre 2.

2. **Ergodicité :** on peut inférer les paramètres (moments) de la loi spatiale à partir d'une réalisation unique si le domaine d'étude est suffisamment grand.

Qu'es aquò ?

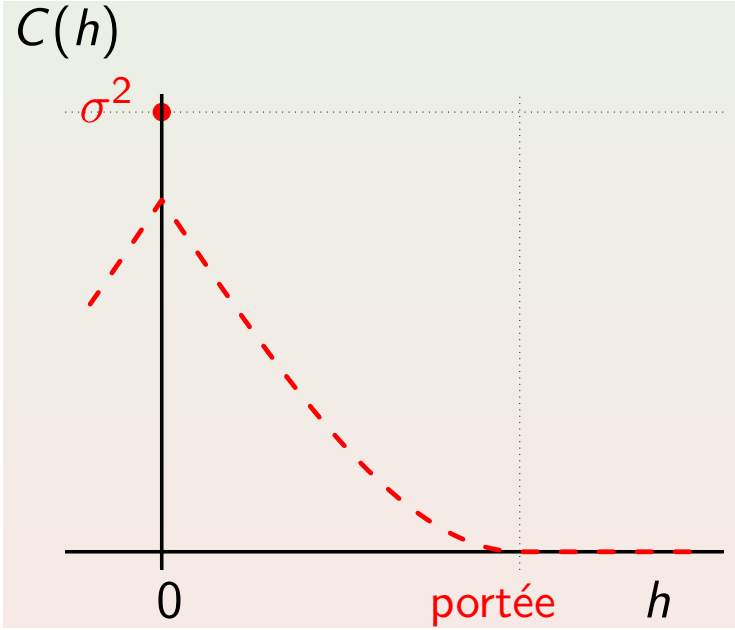
Les deux premiers moments de $Z(x)$ existent et sont invariants par translation :

$$\begin{cases} \mathrm{E}[Z(x)] = m \quad \forall x, \\ Cov\Big(Z(x), Z(x+h)\Big) = C(h) \quad \forall (x,h). \end{cases}$$

Le modèle se résume donc à une moyenne et à une fonction de covariance $C(h)$.

$C(h)$, la fonction de covariance

- $C(-h) = C(h)$
  (fonction paire)
- $C(0) = var(Z(s)) = \sigma^2$
- $C(h)$ peut être négative mais
  $|C(h)| \leq C(0)$



- Il faut que $\forall x_1, \ldots, x_n, \quad \forall \lambda_1, \ldots, \lambda_n \quad :$

$$\sum_{ij} \lambda_i \lambda_j C(x_i - x_j) \geq 0 \Rightarrow C(h) \text{ est définie positive.}$$

Modèles

Simulations de ces modèles théoriques

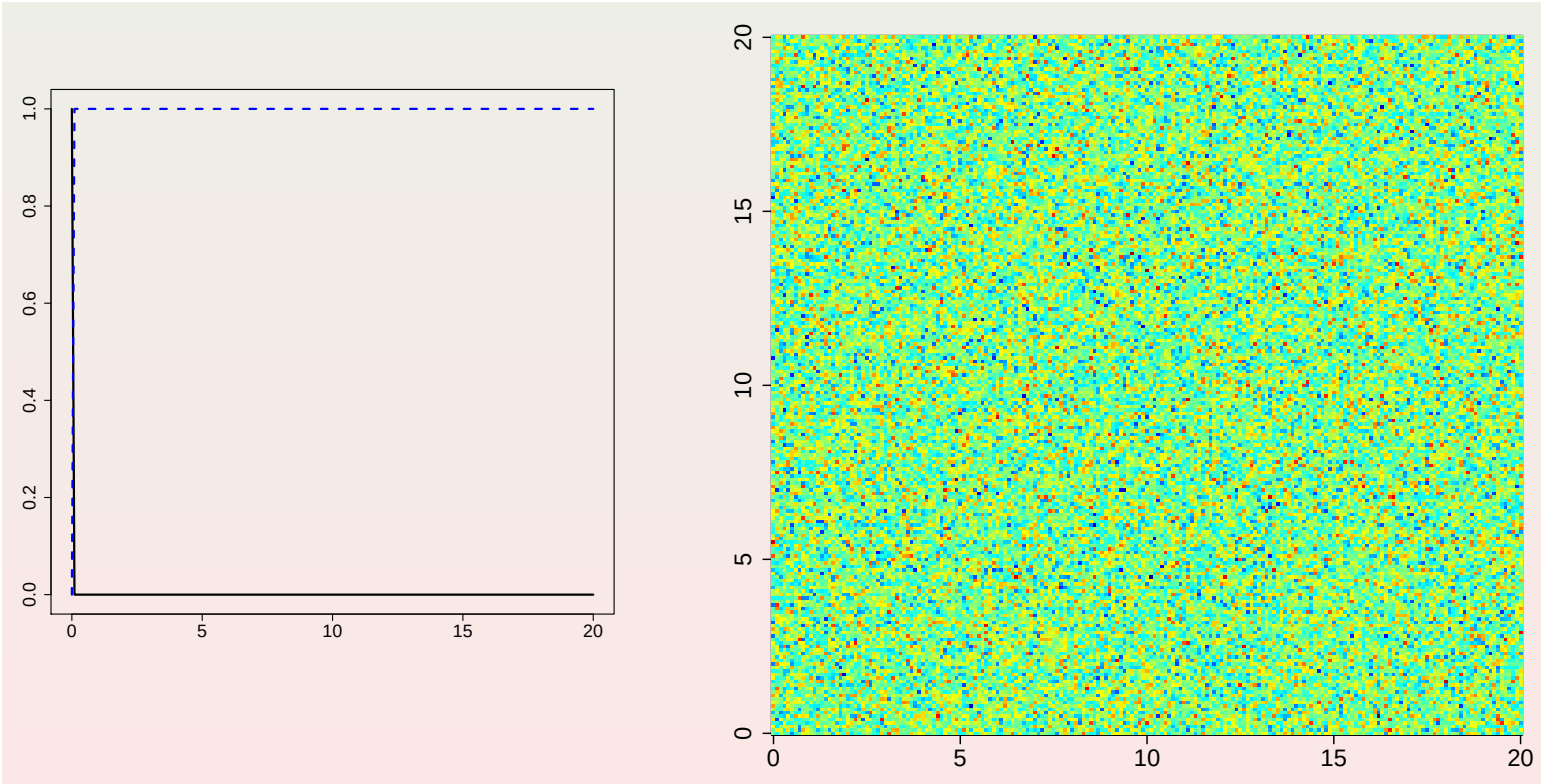
A quoi ressemblent les champs générés par ces modèles ?

$\Rightarrow$ le cas particulier de champs aléatoires $Z(x)$ gaussiens.

Modèles

Simulations de ces modèles théoriques

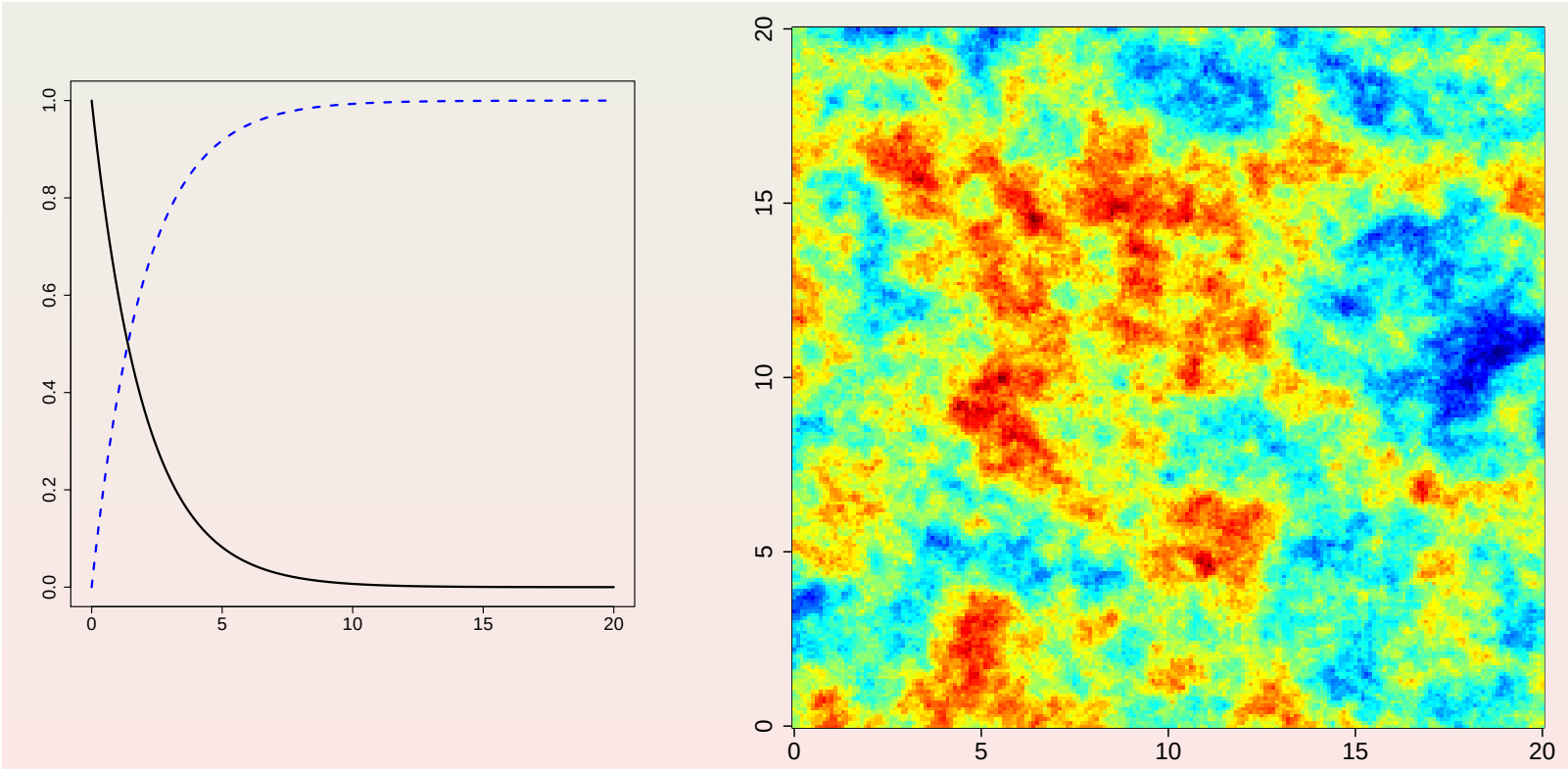
Effet aléatoire pur (bruit blanc spatial)



Modèles

Simulations de ces modèles théoriques

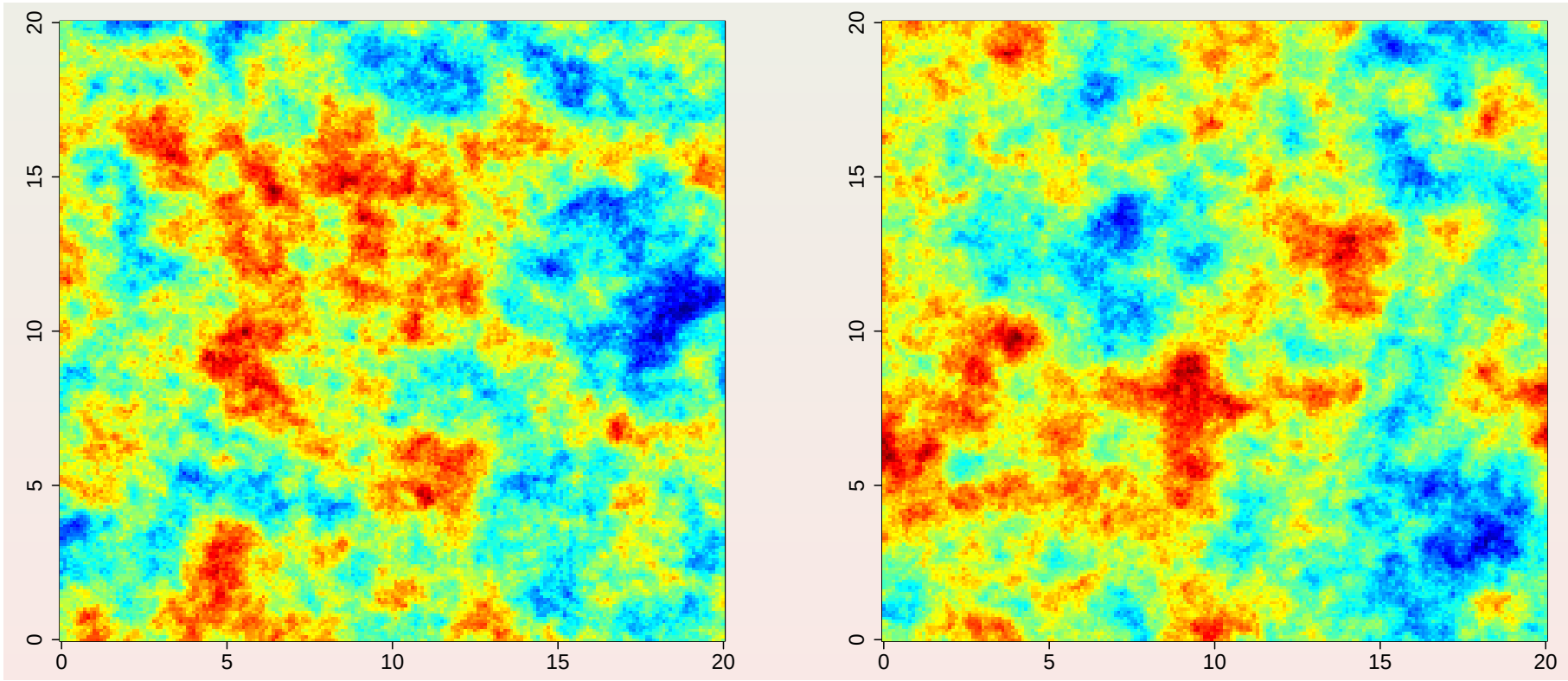
Modèle de covariance exponentiel



Modèles

Simulations de ces modèles théoriques

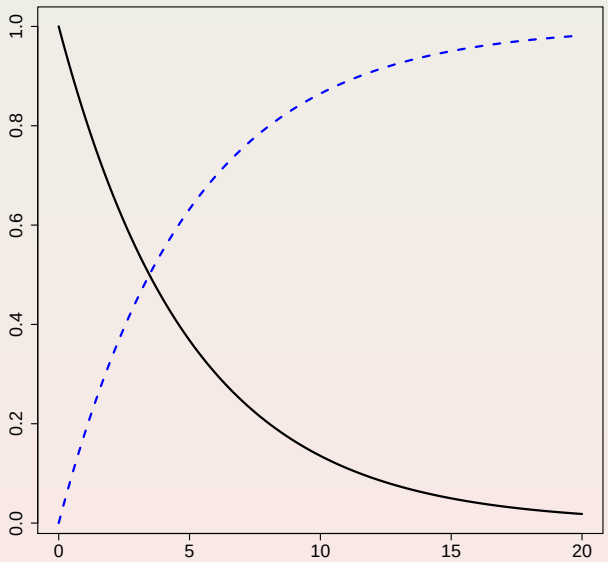
Deux réalisations avec les mêmes paramètres

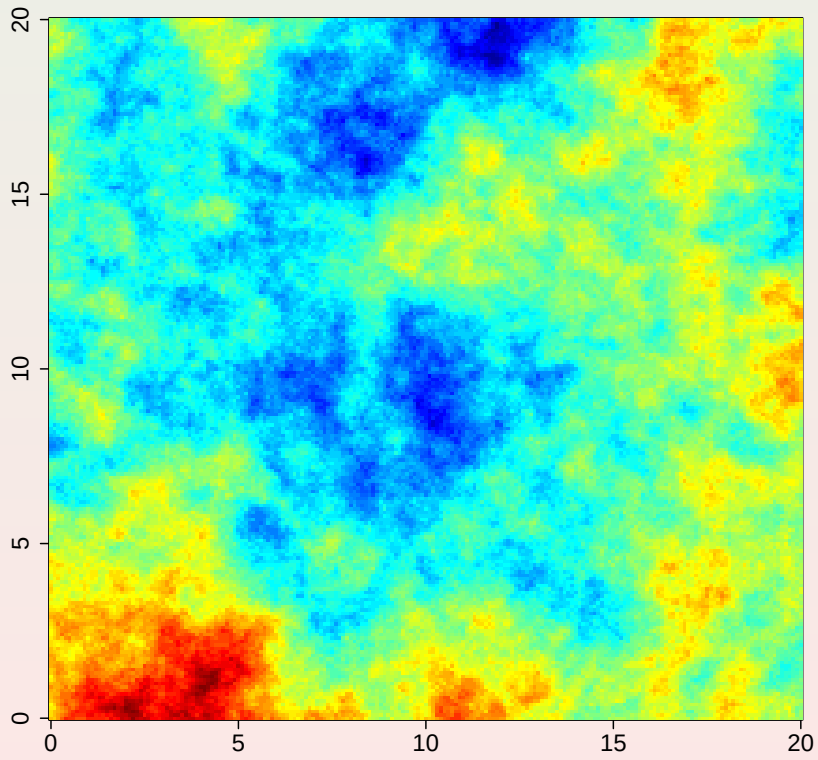


Modèles

Simulations de ces modèles théoriques

Modèle de covariance exponentiel avec une plus grande portée

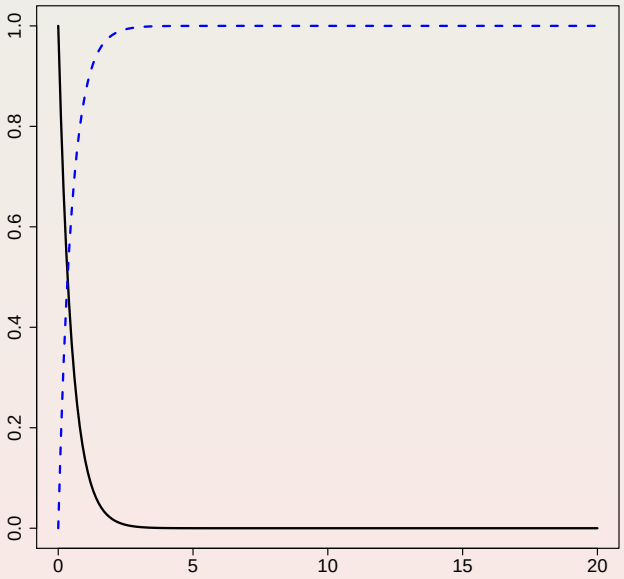


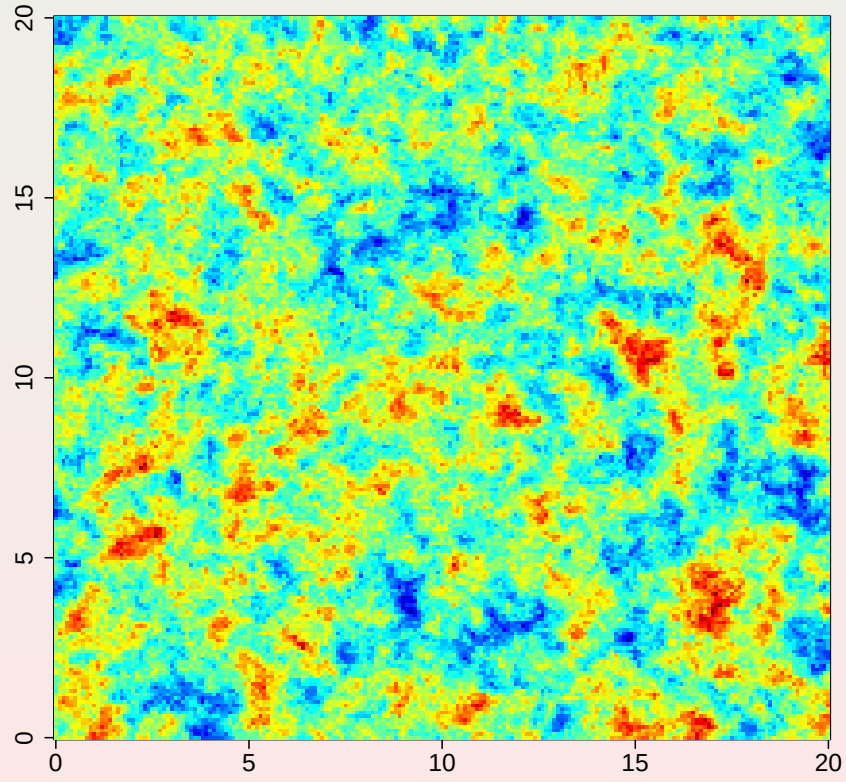


Modèles

Simulations de ces modèles théoriques

Modèle de covariance exponentiel avec une plus petite portée

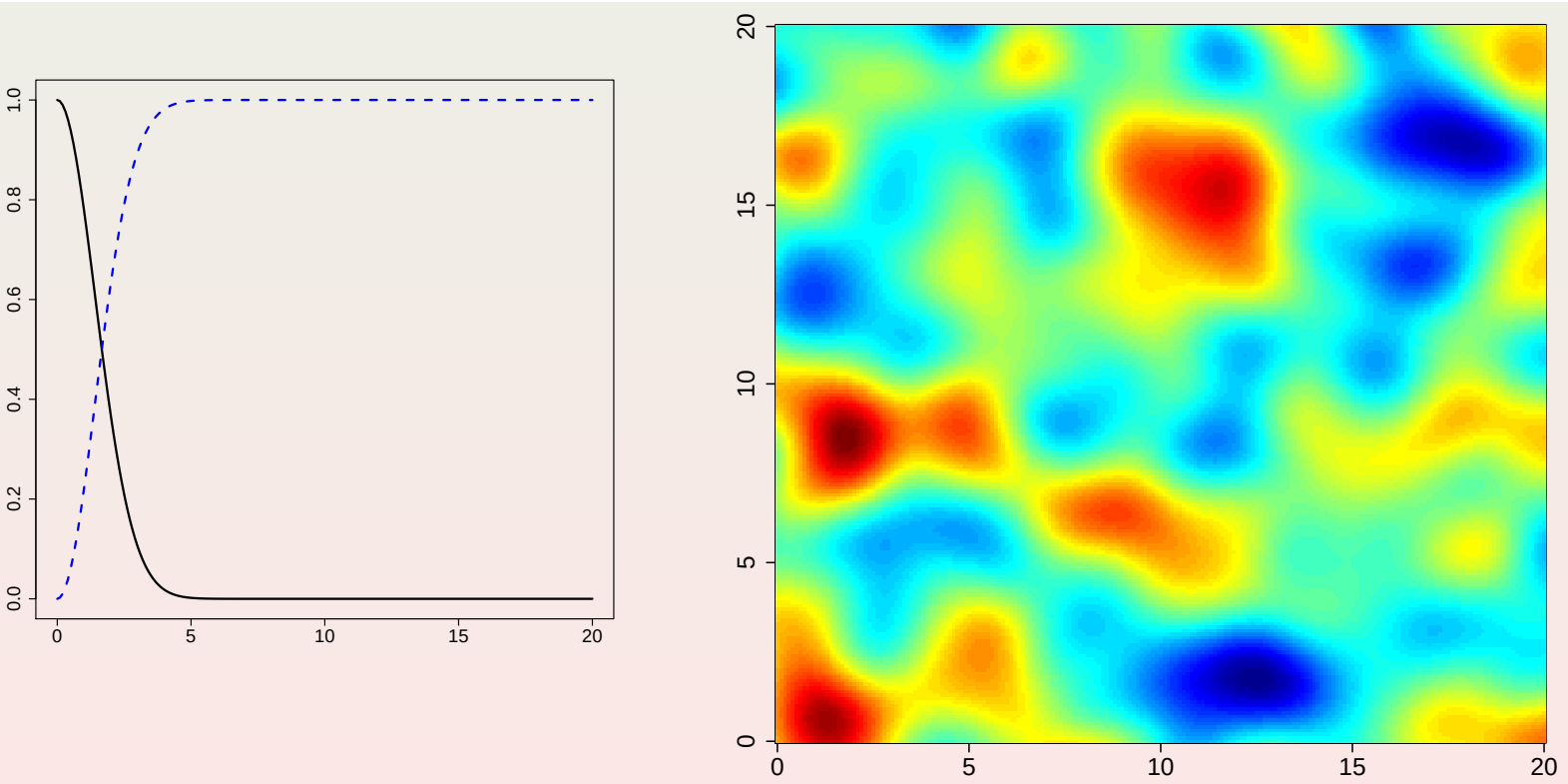




Modèles

Simulations de ces modèles théoriques

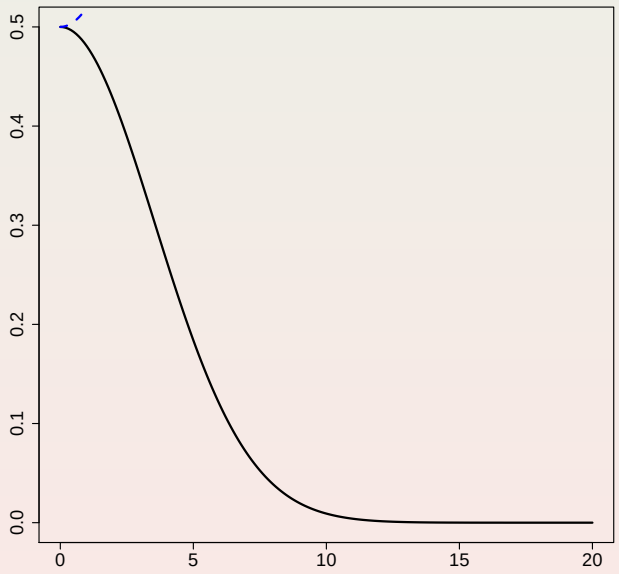
Modèle de covariance gaussien (plus lisse)

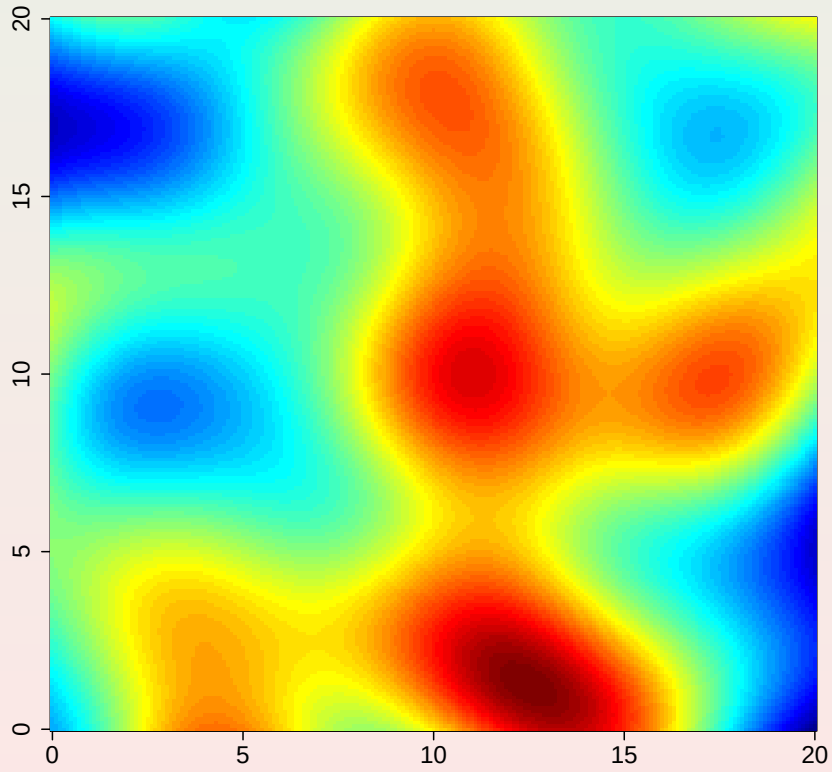


Modèles

Simulations de ces modèles théoriques

Modèle de covariance gaussien avec une plus grande portée

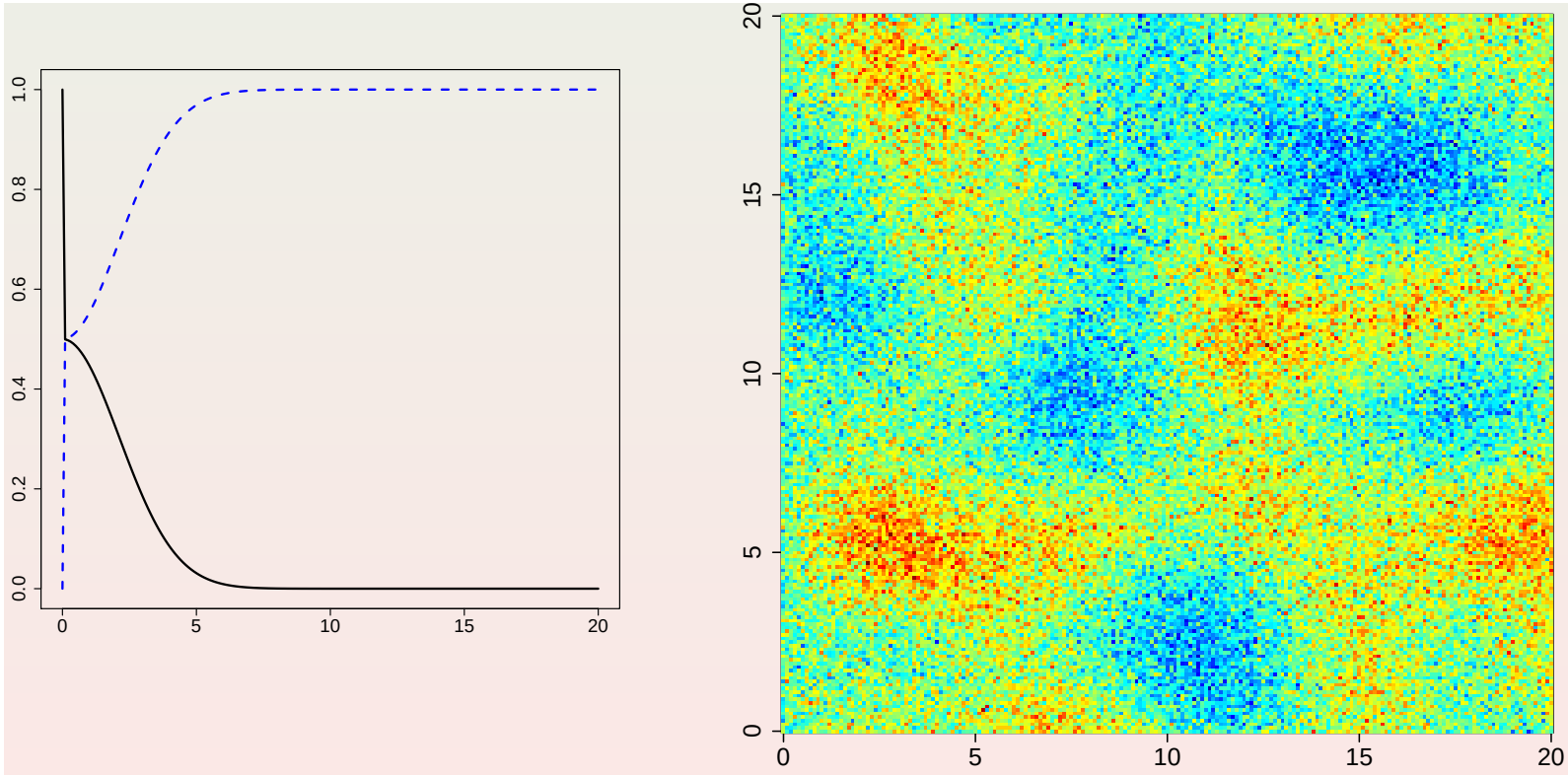




Modèles

Simulations de ces modèles théoriques

Modèle de covariance gaussien avec effet de pépite



Approche bayésienne

Modélisation paramétrique

Observations $y_1, \ldots, y_n$, avec

$$\mathrm{y} = (y_1, \ldots, y_n) \sim [\mathrm{y}|\theta],\ \theta \text{ paramètre inconnu.}$$

Objectif : estimer le paramètre θ à partir de l'échantillon $y_1, \ldots, y_n$.

Exemple : régression linéaire classique, $\mathrm{y} \sim N(\mathrm{X}\beta, \sigma^2 I)$, c'est à dire

$$y_i = \beta_0 + \beta_1 * x_{1i} + \ldots + \varepsilon_i, \text{ avec } \varepsilon_i \sim N(0, \sigma^2).$$

Une approche classique : le maximum de vraisemblance

Vraisemblance : mesure de l'adéquation entre la distribution observée sur un échantillon aléatoire et une loi de probabilité supposée décrire une réalité sur la population dont l'échantillon est issu.

$$l(\theta) \propto [\mathrm{y}|\theta].$$

On cherche la valeur de θ qui maximise la vraisemblance, c'est à dire on cherche la valeur de θ qui rend l'observation de y la plus probable.

Approche bayésienne : formule de Bayes

Le paramètre inconnu θ devient une variable aléatoire au même titre que les observations.

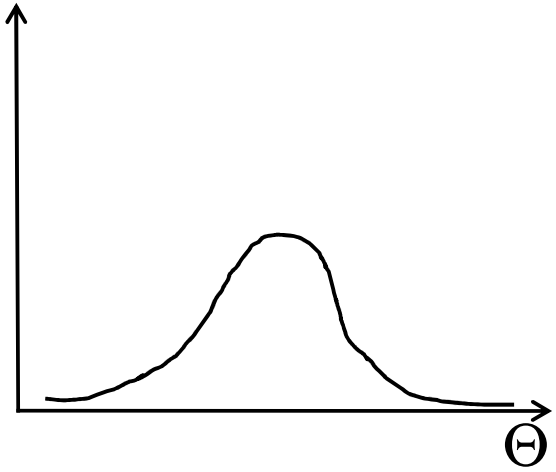
$\Rightarrow$ on cherche donc à caractériser la distribution *a posteriori* des paramètres, $[\theta|\mathrm{y}]$, à l'aide de la formule de Bayes.

$$[\theta|\mathrm{y}] = \frac{[\theta][\mathrm{y}|\theta]}{[\mathrm{y}]}$$

Approche bayésienne : principe général

Connaissance a priori

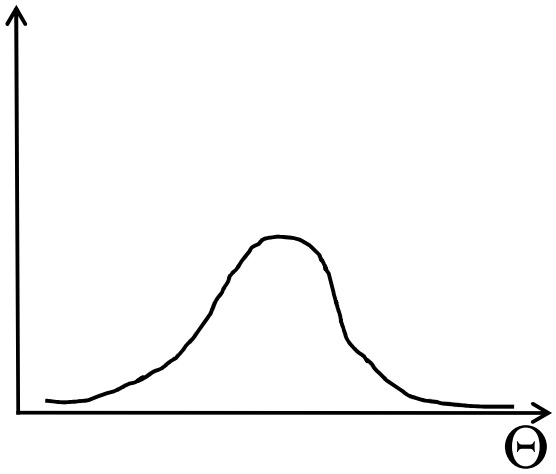
$[\Theta]$



Approche bayésienne : principe général

Connaissance a priori

$$[\Theta]$$



Données
+
Vraisemblance

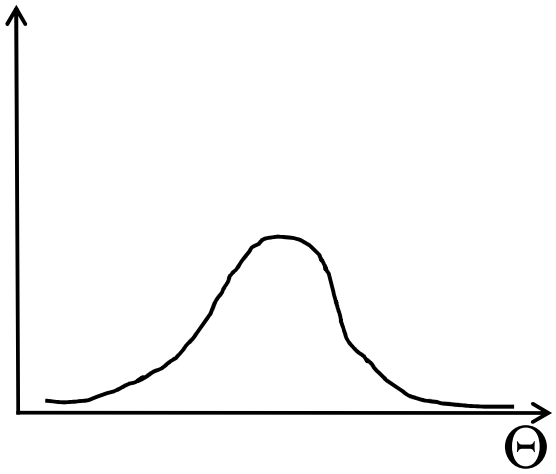
$$[\mathbf{y}|\Theta]$$

$$\begin{cases} \mu_i = \beta_0 + \beta_1 x_{1i} + \dots \\ y_i \sim N(\mu_i, \sigma^2) \end{cases}$$

Approche bayésienne : principe général

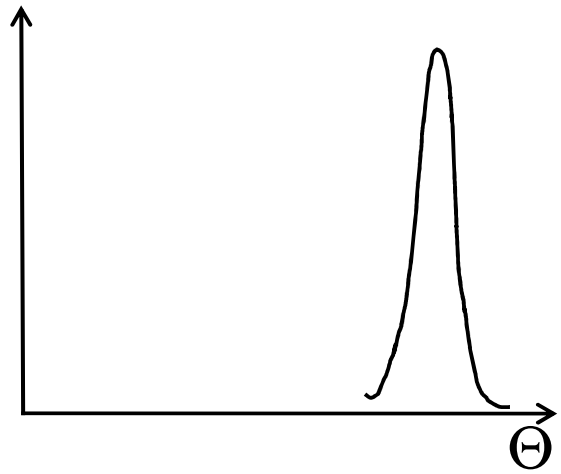
Connaissance a priori

$[\Theta]$



Connaissance après mise à jour

$[\Theta|\mathbf{y}]$



Données
+
Vraisemblance

$[\mathbf{y}|\Theta]$

$$\begin{cases} \mu_i = \beta_0 + \beta_1 x_{1i} + \dots \\ y_i \sim N(\mu_i, \sigma^2) \end{cases}$$

Approche bayésienne : estimation

Problème $\rightarrow$ multiples intégrales à calculer :

- $[\theta|\mathrm{y}] = [\theta][\mathrm{y}|\theta]/[\mathrm{y}]$, avec $[\mathrm{y}] = \int_{\Theta}([\mathrm{y}|\alpha][\alpha])d\alpha$,
- Calcul des distributions marginales si θ est multivarié,
- Modèles à variables latentes...

Approche bayésienne : estimation

Problème $\rightarrow$ multiples intégrales à calculer :

- $[\theta|\mathrm{y}] = [\theta][\mathrm{y}|\theta]/[\mathrm{y}]$, avec $[\mathrm{y}] = \int_{\Theta}([\mathrm{y}|\alpha][\alpha])d\alpha$,
- Calcul des distributions marginales si θ est multivarié,
- Modèles à variables latentes...

Approche bayésienne : estimation

Problème $\rightarrow$ multiples intégrales à calculer :

- $[\theta|\mathrm{y}] = [\theta][\mathrm{y}|\theta]/[\mathrm{y}]$, avec $[\mathrm{y}] = \int_{\Theta}([\mathrm{y}|\alpha][\alpha])d\alpha$,
- Calcul des distributions marginales si θ est multivarié,
- Modèles à variables latentes...

Approche bayésienne : estimation

Problème $\rightarrow$ multiples intégrales à calculer :

- $[\theta|\mathrm{y}] = [\theta][\mathrm{y}|\theta]/[\mathrm{y}]$, avec $[\mathrm{y}] = \int_{\Theta}([\mathrm{y}|\alpha][\alpha])d\alpha$,
- Calcul des distributions marginales si θ est multivarié,
- Modèles à variables latentes...

Les outils et logiciels

- INLA (Integrated Nested Laplace Approximations) : R-INLA et inlabru
- MCMC (Markov Chain Monte Carlo) : WinBUGS, OpenBUGS, JAGS, Nimble
- HMC (Hamiltonian Monte Carlo) : R-Stan
- PMC (PopulationMonte Carlo) : BIIPS, Nimble

Modèle hiérarchique

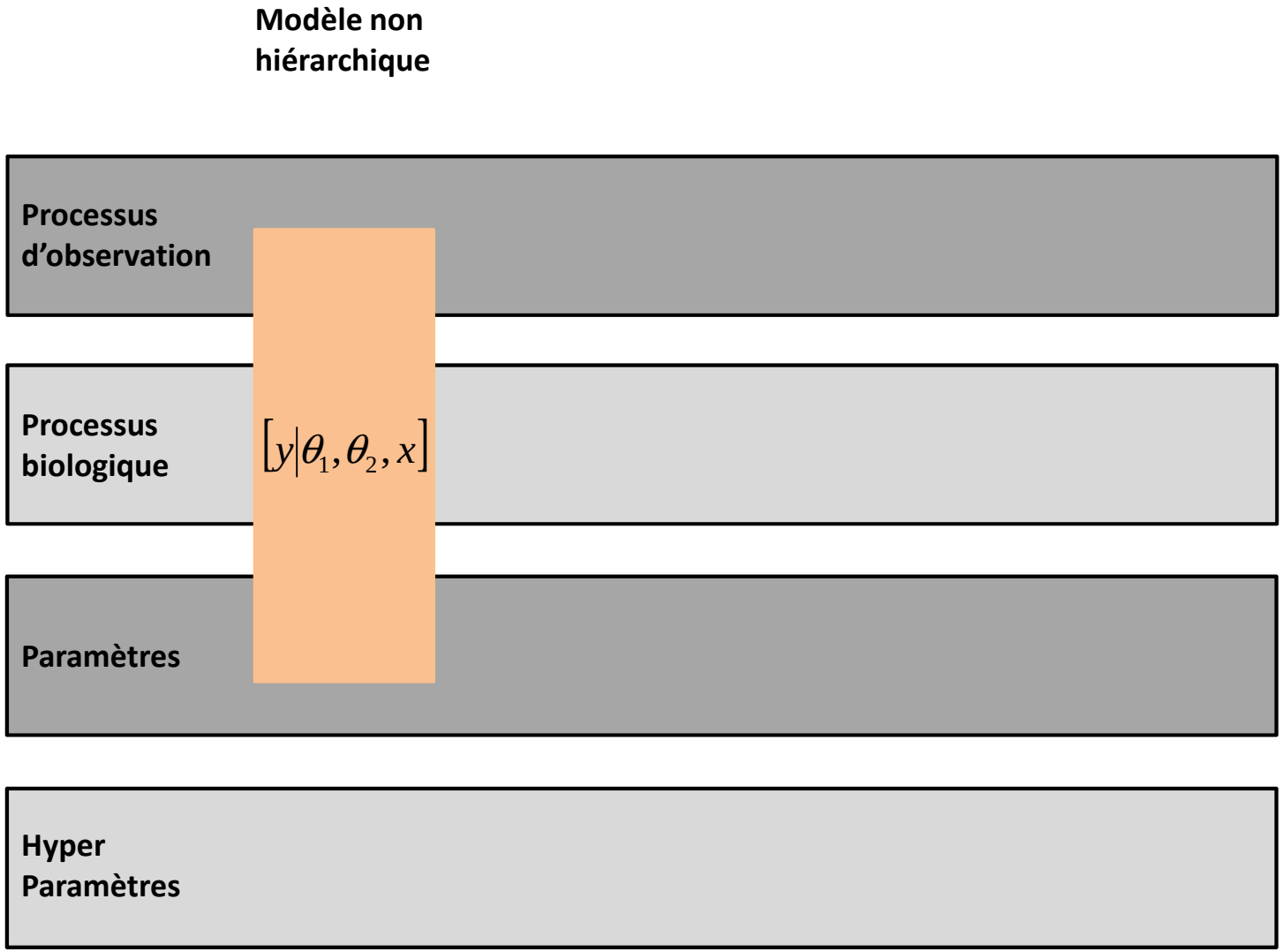
Processus d'observation

Processus biologique

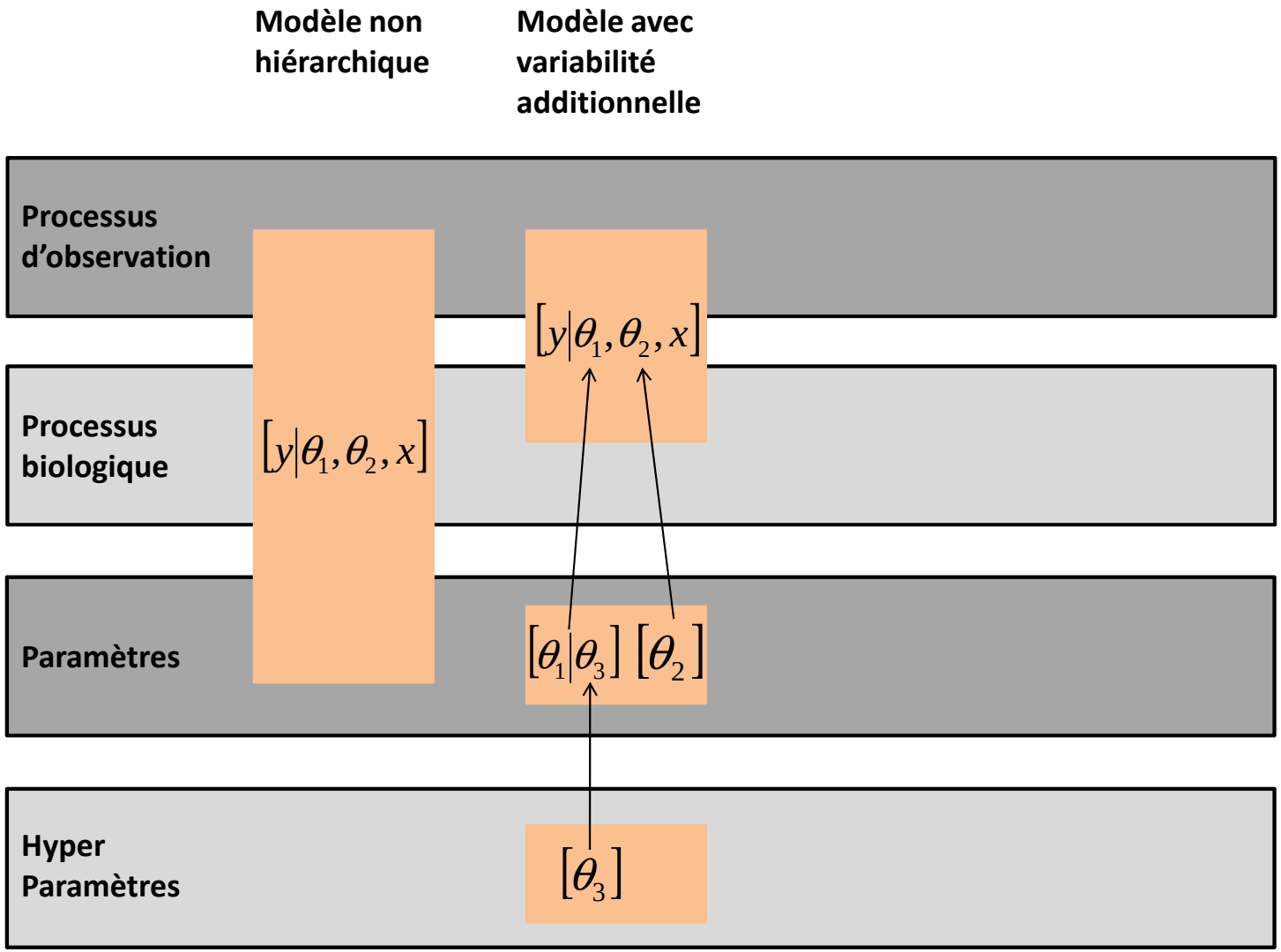
Paramètres

Hyper Paramètres

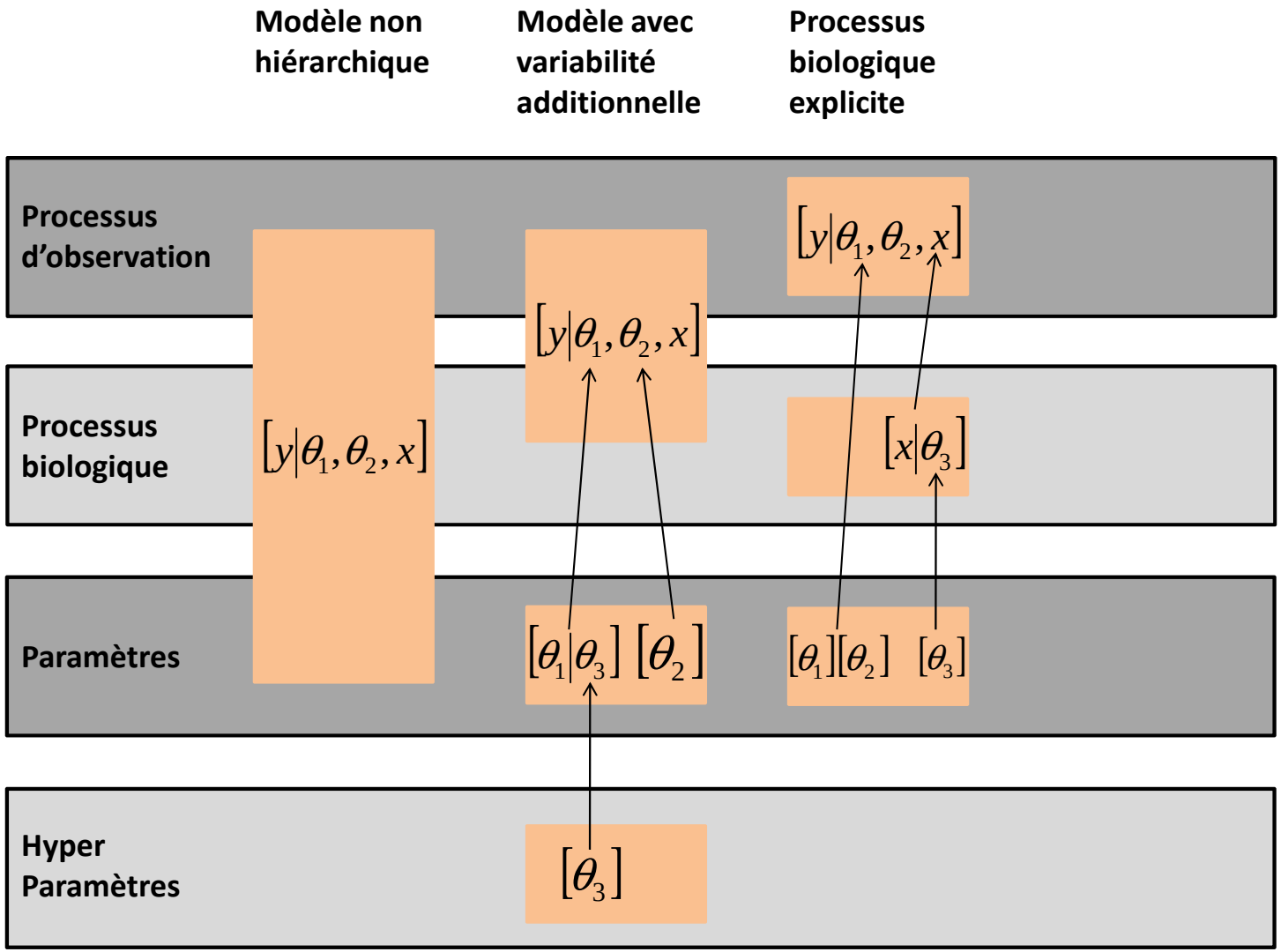
Exemple non hiérarchique



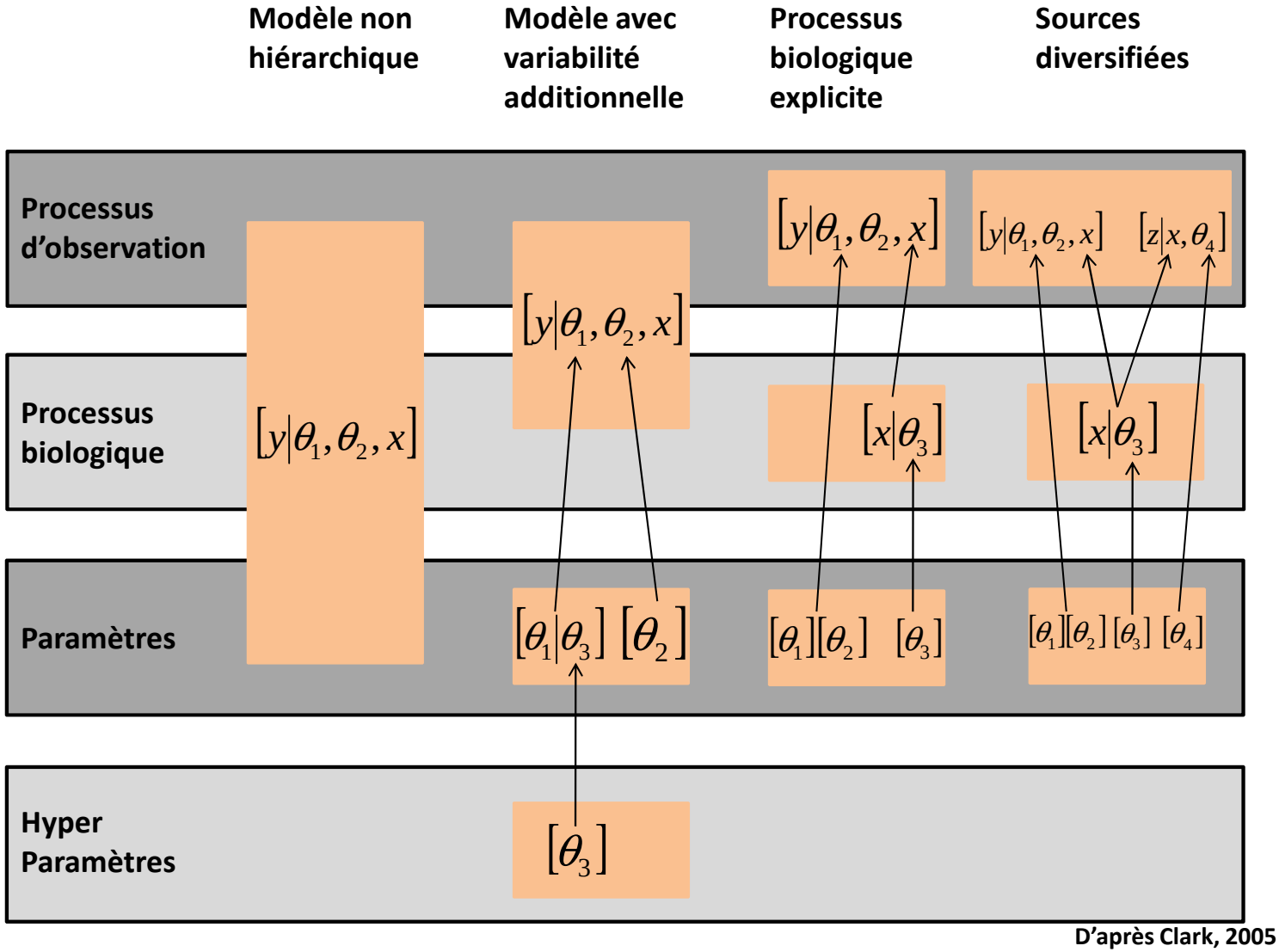
Un premier cas hiérarchique : la variabilité individuelle



Modèle avec processus identifié



Intégration de différentes sources de données



D'après Clark, 2005

Les modèles linéaires généralisés mixtes spatialisés

Régression linéaire classique et mixte

$$
\begin{cases}
\textbf{A priori} \\
\sigma, \beta_1, \beta_2, \\
\textbf{Espérance} \\
\mathsf{E}[y_i] = \mu_i = \beta_0 + \beta_1 x_i^{(1)} + \beta_2 x_i^{(2)} + \ldots, \\
\textbf{Vraisemblance} \\
y_i \sim \mathcal{N}(\mu_i, \sigma^2).
\end{cases}
$$

Les modèles linéaires généralisés mixtes spatialisés

Régression linéaire classique et mixte

$$
\begin{cases}
\textbf{A priori} \\
\sigma, \beta_1, \beta_2, \\
\textbf{Espérance} \\
\mathrm{E}[y_i | a_i] = \mu_i = \beta_0 + \beta_1 x_i^{(1)} + \beta_2 x_i^{(2)} + \ldots + a_i, \\
\textbf{Vraisemblance} \\
y_i \sim \mathcal{N}(\mu_i, \sigma^2).
\end{cases}
$$

Les modèles linéaires généralisés mixtes spatialisés

Régression linéaire classique et mixte

$$
\begin{cases}
\textbf{A priori} \\
a_i \sim \mathcal{N}(0, \tau^2), \\
\sigma, \beta_1, \beta_2, \\
\textbf{Espérance} \\
\mathrm{E}[y_i | a_i] = \mu_i = \beta_0 + \beta_1 x_i^{(1)} + \beta_2 x_i^{(2)} + \ldots + a_i, \\
\textbf{Vraisemblance} \\
y_i \sim \mathcal{N}(\mu_i, \sigma^2).
\end{cases}
$$

Les modèles linéaires généralisés mixtes spatialisés

Régression linéaire classique et mixte

$$
\begin{cases}
\textbf{Hyper - a priori} \\
\tau, \\
\textbf{A priori} \\
a_i \sim \mathcal{N}(0, \tau^2), \\
\sigma, \beta_1, \beta_2, \\
\textbf{Espérance} \\
\mathsf{E}[y_i | a_i] = \mu_i = \beta_0 + \beta_1 x_i^{(1)} + \beta_2 x_i^{(2)} + \ldots + a_i, \\
\textbf{Vraisemblance} \\
y_i \sim \mathcal{N}(\mu_i, \sigma^2).
\end{cases}
$$

Les modèles linéaires généralisés mixtes spatialisés

1er changement : le processus d'observation

Principale différence : le processus d'observation n'est plus forcément normal,

$$y_i \sim \mathcal{L}(\theta), \text{ avec } \mathcal{L} \text{ une Poisson, binomiale, ...}$$

Les modèles linéaires généralisés mixtes spatialisés

1er changement : le processus d'observation

Principale différence : le processus d'observation n'est plus forcément normal,

$$y_i \sim \mathcal{L}(\theta), \text{ avec } \mathcal{L} \text{ une Poisson, binomiale, ...}$$

On modélise ensuite l'**espérance** de y,

$$\mathrm{E}[y_i] = g(\beta_0 + \beta_1 x_i^{(1)} + \beta_2 x_i^{(2)} + \ldots), \text{ avec } g \text{ une fonction de lien.}$$

La fonction de lien permet de **respecter des contraintes sur l'espérance** de y, par exemple positivité dans le cas d'une loi de Poisson.

Les modèles linéaires généralisés mixtes spatialisés

2nd changement : structuration spatiale de l'effet aléatoire

Idée : les a_i de l'effet aléatoire ne sont plus indépendants et leur corrélation dépend de leur distance dans l'espace,

$$\text{cov}(a_i, a_j) = C(\text{dist}(a_i, a_j)).$$

Les modèles linéaires généralisés mixtes spatialisés

2nd changement : structuration spatiale de l'effet aléatoire

Idée : les a_i de l'effet aléatoire ne sont plus indépendants et leur corrélation dépend de leur distance dans l'espace,

$$\text{cov}(a_i, a_j) = C(\text{dist}(a_i, a_j)).$$

De ce fait on a :

$\text{a} \sim \mathcal{MVN}(0, \Sigma)$, avec $\Sigma_{i,j} = C(\text{dist}(a_i, a_j))$.

Les modèles linéaires généralisés mixtes spatialisés

2nd changement : structuration spatiale de l'effet aléatoire

Idée : les a_i de l'effet aléatoire ne sont plus indépendants et leur corrélation dépend de leur distance dans l'espace,

$$\text{cov}(a_i, a_j) = C(\text{dist}(a_i, a_j)).$$

De ce fait on a :

$\text{a} \sim \mathcal{MVN}(0, \Sigma)$, avec $\Sigma_{i,j} = C(\text{dist}(a_i, a_j))$.

La fonction C est appelée **fonction de covariance** et doit vérifier certaines contraintes.

Les modèles linéaires généralisés mixtes spatialisés

En résumé

Modèle linéaire mixte :

$$
\begin{cases}
\textbf{Hyper - a priori} \\
\tau, \\
\textbf{A priori} \\
a_i \sim \mathcal{N}(0, \tau^2), \\
\sigma, \beta_1, \beta_2, \\
\textbf{Espérance} \\
\mathrm{E}[y_i | a_i] = \mu_i = \beta_0 + \beta_1 x_i^{(1)} + \beta_2 x_i^{(2)} + \ldots + a_i, \\
\textbf{Vraisemblance} \\
y_i \sim \mathcal{N}(\mu_i, \sigma^2).
\end{cases}
$$

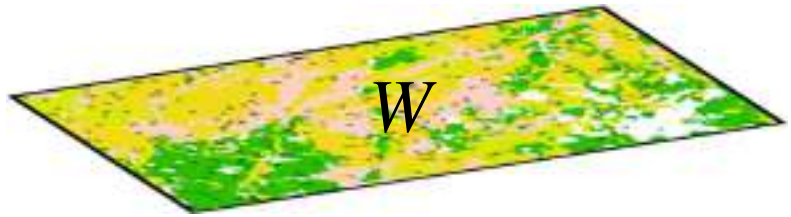
Les modèles linéaires généralisés mixtes spatialisés

En résumé

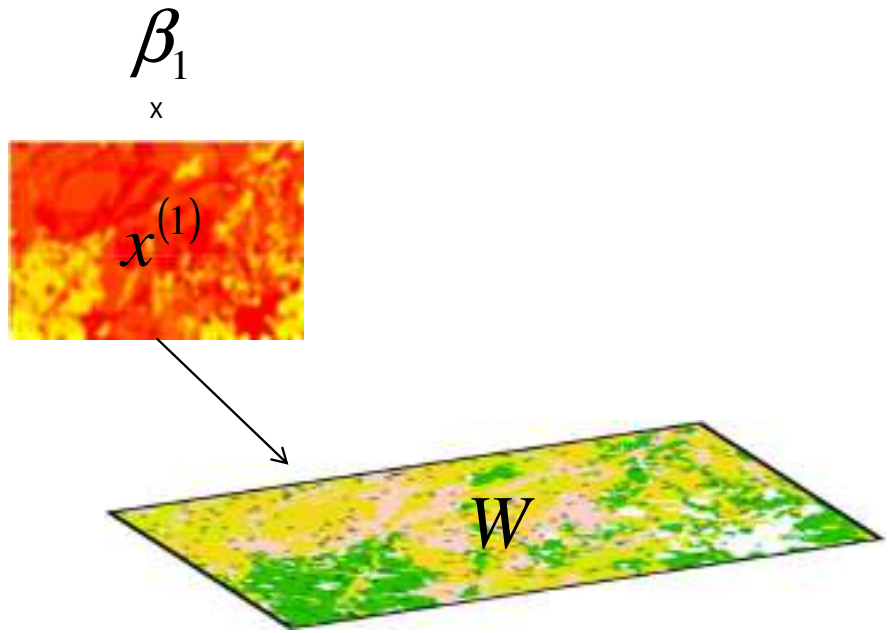
Modèle linéaire généralisé mixte spatial :

$$\begin{cases} \textbf{Hyper - a priori} \\ \tau, \\ \textbf{A priori} \\ \mathrm{a} \sim \mathcal{MVN}(0, \Sigma), \text{avec } \Sigma_{i,j} = C(\tau, \text{dist}(a_i, a_j)) \\ \sigma, \beta_1, \beta_2, \\ \textbf{Espérance} \\ \mathrm{E}[y_i | a_i] = g(\beta_0 + \beta_1 x_i^{(1)} + \beta_2 x_i^{(2)} + \ldots + a_i), \\ \textbf{Vraisemblance} \\ y_i \sim \mathcal{L}(\theta). \end{cases}$$

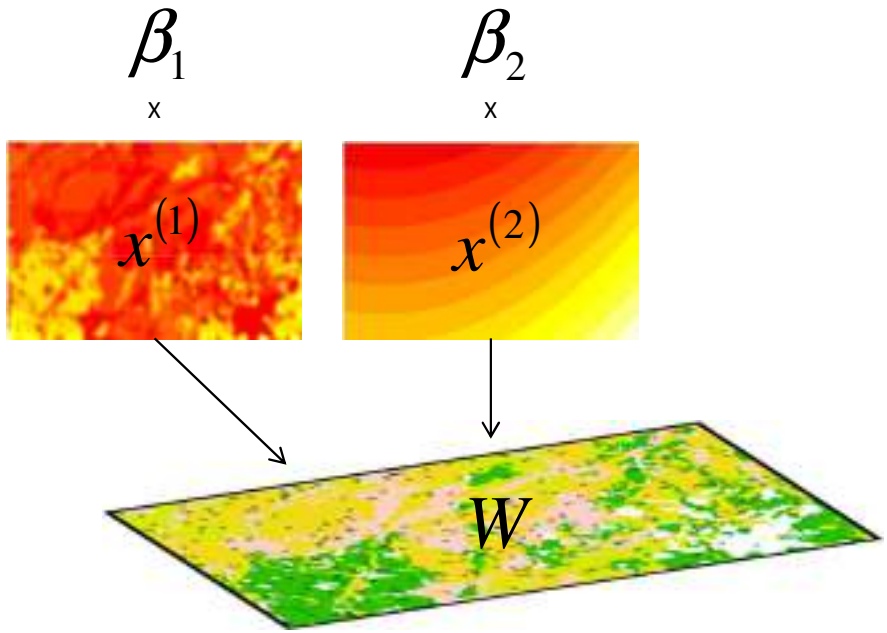
Les modèles linéaires généralisés mixtes spatialisés



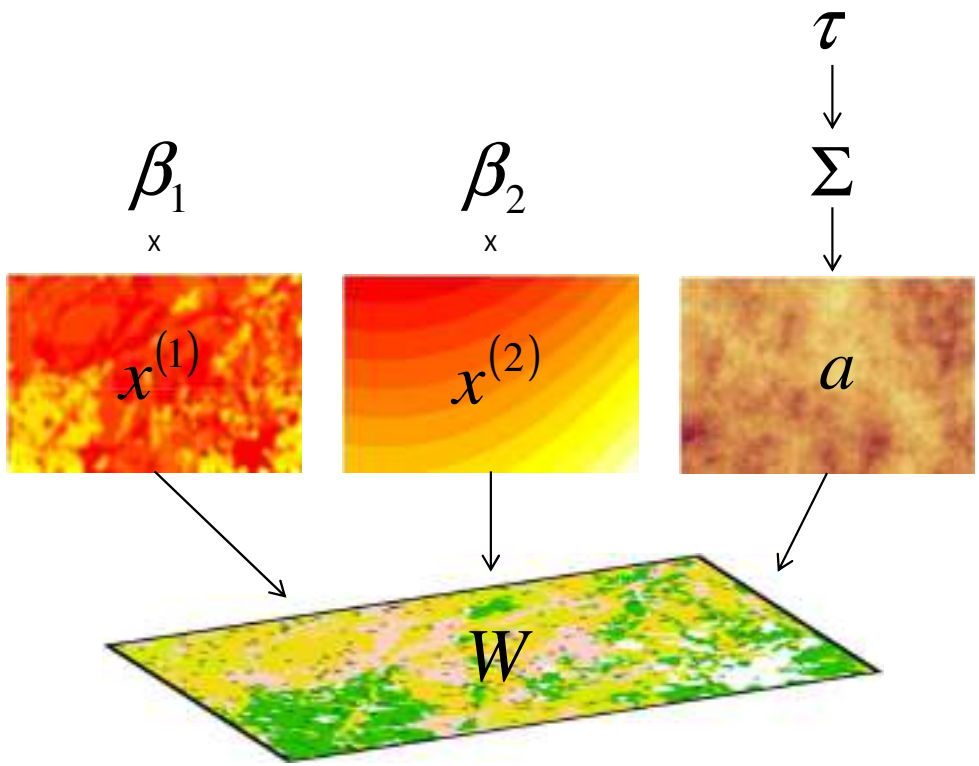
Les modèles linéaires généralisés mixtes spatialisés



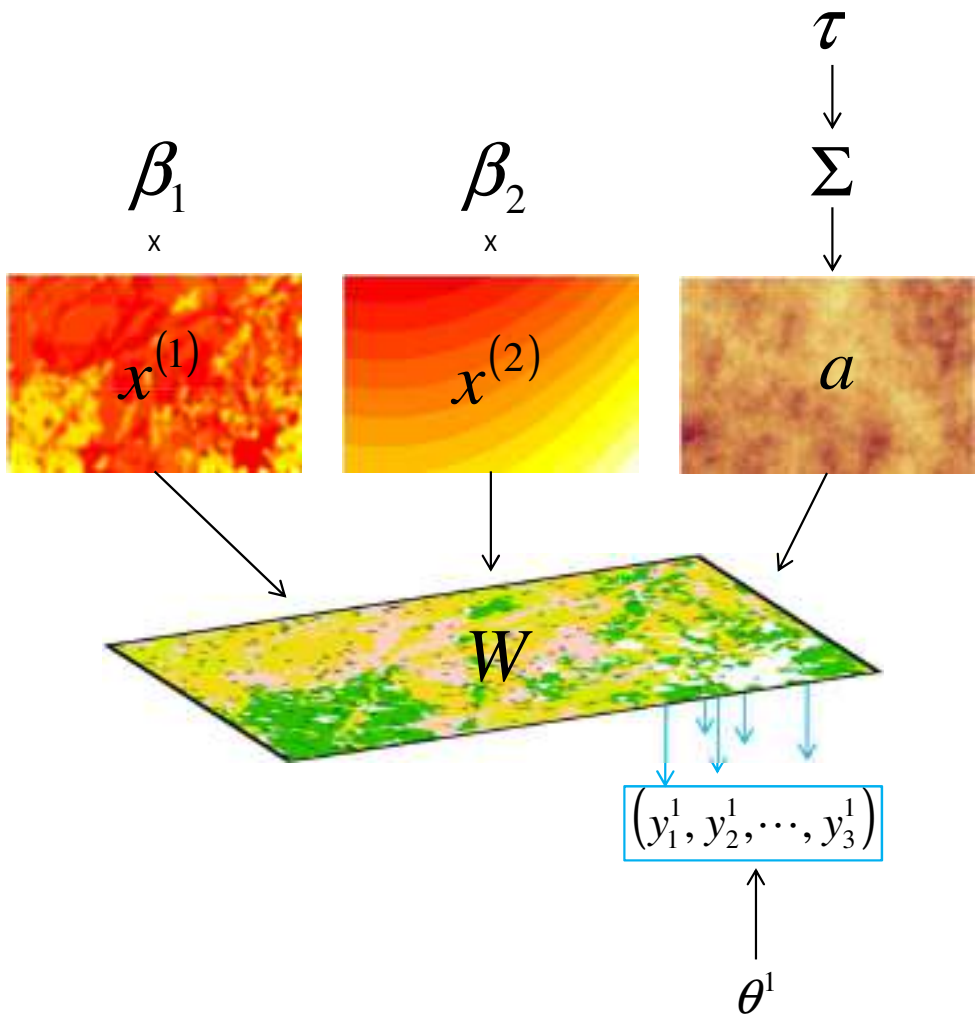
Les modèles linéaires généralisés mixtes spatialisés



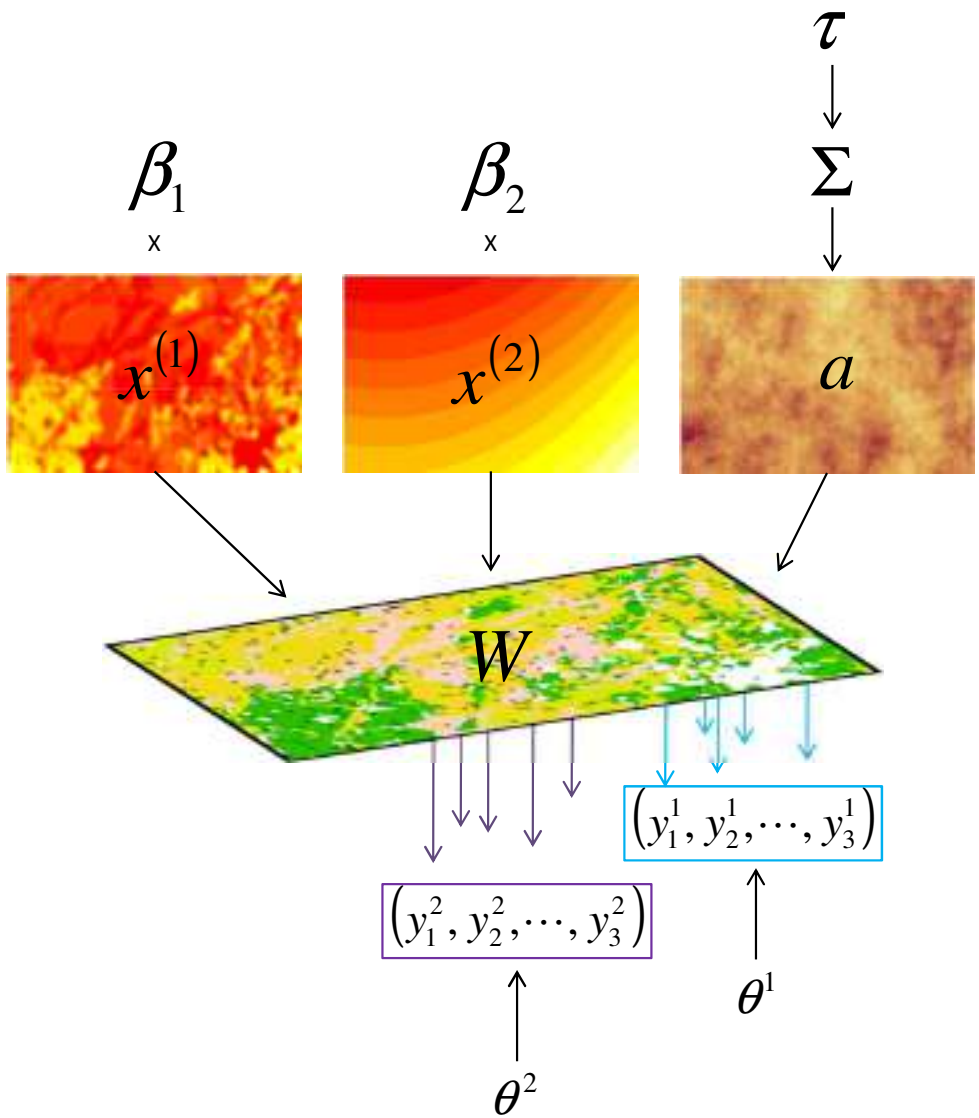
Les modèles linéaires généralisés mixtes spatialisés



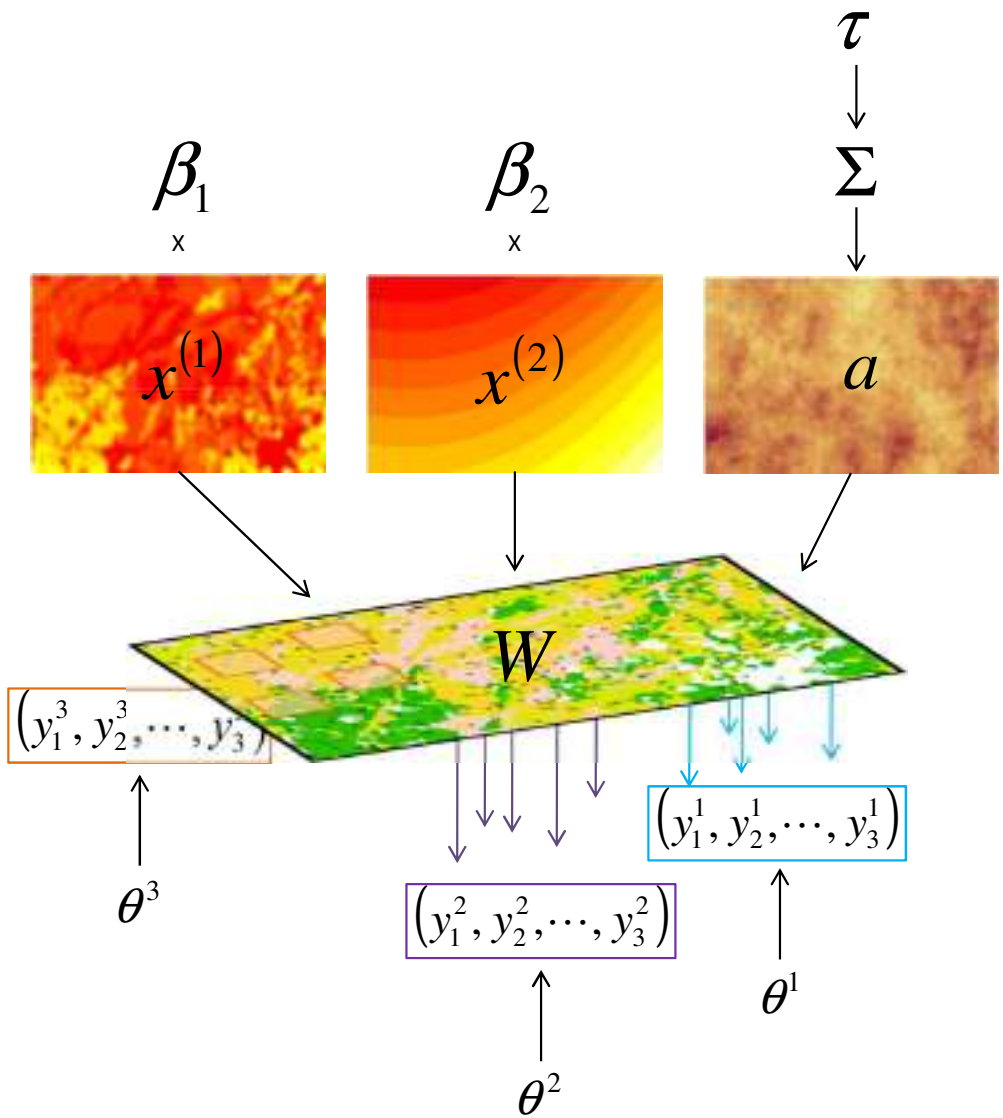
Les modèles linéaires généralisés mixtes spatialisés



Les modèles linéaires généralisés mixtes spatialisés



Les modèles linéaires généralisés mixtes spatialisés



INLA et inlabru : quels modèles, quelles fonctionnalités ?

Thomas Opitz

Atelier

Modèles bayésiens spatialisés pour l'écologie avec `INLA` et `inlabru`

Avignon, 13-14 avril 2023

Contexte de modélisation

- Une ou plusieurs **"variables" discrètes à expliquer et prédire** en fonction des **covariables environnementales**

  $\Rightarrow$ *Modèles de régression (GLMs, GAMs...)*

- **Variabilité spatiale résiduelle** même après prise en compte des covariables, en raison de covariables inconnues/indisponibles ou de dynamiques de population (p. ex. colonisation)

  $\Rightarrow$ *Effets aléatoires spatiaux, "modèles mixtes"*

- Couplage de **plusieurs jeux d'observations** :
  - Une seule espèce, mais différents types d'observations ("presences seules opportunistes" + "présences-absences protocolées")
  - Deux espèces
  - Espèce focale ("Target") + Espèces de fond ("Target Background")

  $\Rightarrow$ *Régression multi-réponse, effets aléatoires partagés*

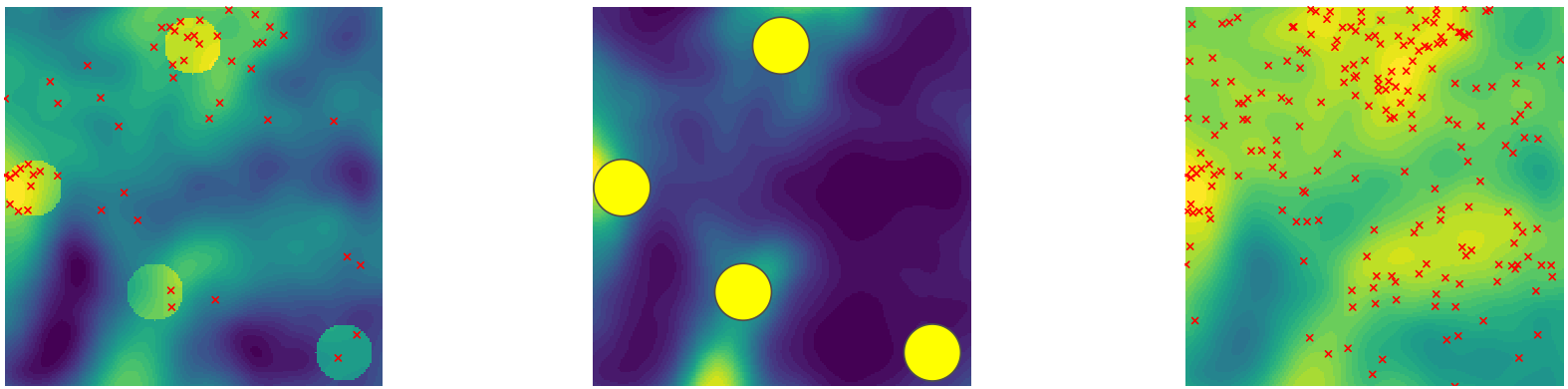
- **Objectifs :**
  - Caractérisation des **niches** et des **interactions** entre espèces
  - **Cartographie** (présence, abondance...)
  - **Identification et correction des biais** d'observation

Effort d'observation variable

- **Données protocolées** typiquement sous forme de **"presences-absences"** (ou de comptages) dans une fenêtre spatiotemporelle connue (p. ex. plaçette, point d'écoute)
- **Données opportunistes** typiquement sous forme de **"présences-seules"**
- **Effort d'observation hétérogène** :
  - Inconnu pour les données opportunistes
  - Connu (et localement exhaustif) pour les données protocolées

Illustration schématique :
Cercles jaunes = plaçettes pour données protocolées
$\Rightarrow$ On souhaite connaître les champs de l'effort et de l'abondance réelle

Observations = **Effort d'observation** $\times$ **Abondance réelle**



Représentation des "présences seules"

Différentes granularités (du plus précis vers le moins précis) :

- **Coordonnées géographiques** des occurrences dans le domaine d'étude D
  = Semis de points
  = Réalisation d'un processus ponctuel :

$$\{S_1, S_2, \ldots, S_n\} \sim \text{Processus ponctuel}(\lambda(s))$$

  - Fonction d'intensité $\lambda(s)$
  - Modèle classique : **Processus de Poisson** (indépendance des occurrences)

- **Comptages** dans des **unités spatiales** A_k avec $D = \bigcup_k A_k$

$$N_k = \#\{S_i \in A_k\} \in \{0, 1, 2 \ldots\}$$

  - A_k selon une grille régulière ou des unités biophysiques/administratives
  - N_k suit une **loi de Poisson** si $\{S_i\} \sim$ Processus de Poisson

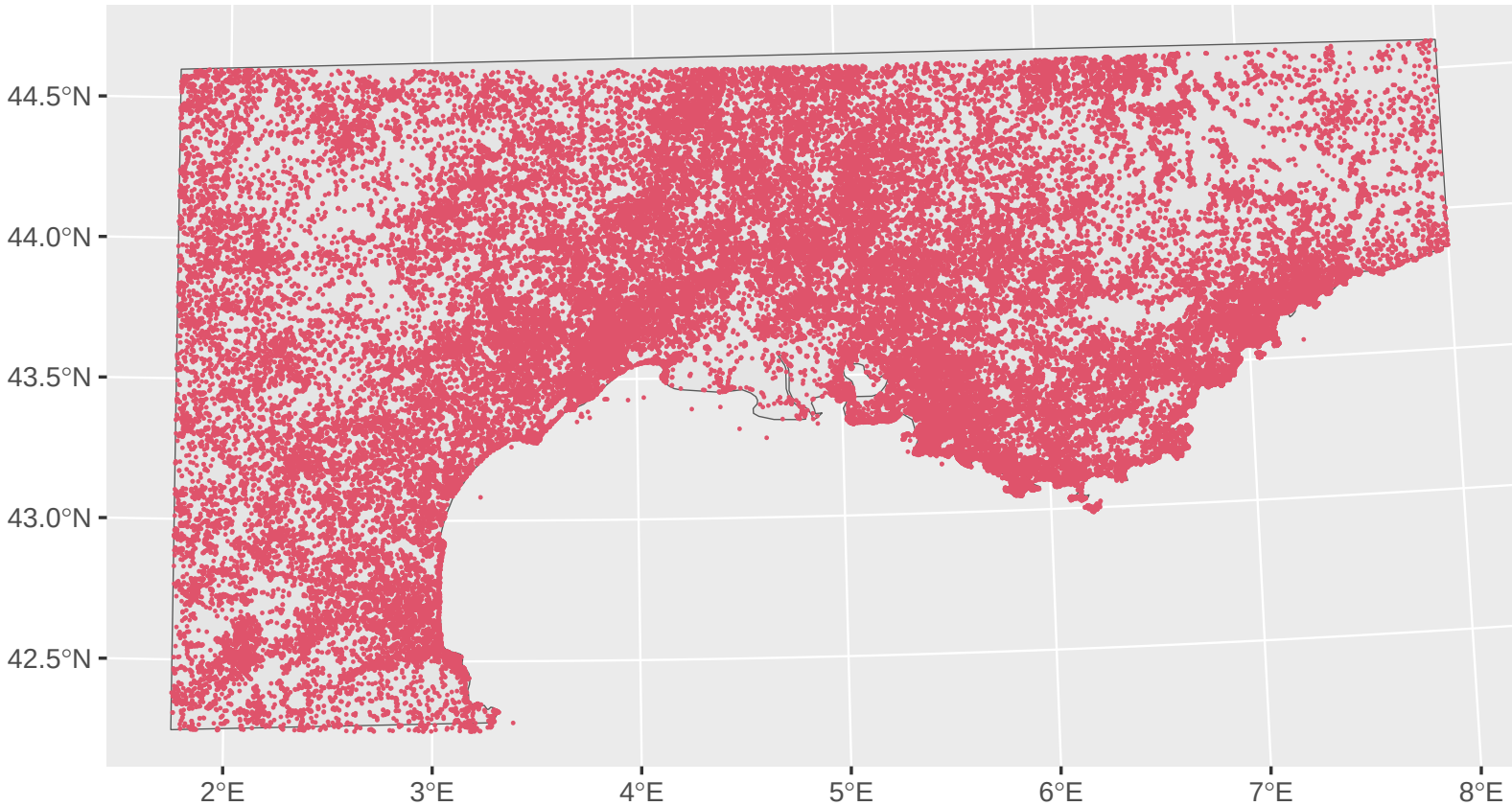
- **Présences-absences** dans les unités spatiales A_k:

$$Z_k = \begin{cases} 1 & \text{si } N_k > 0, \\ 0 & \text{sinon,} \end{cases} \qquad \text{avec } Z_k \in \{0, 1\}$$

  - Z_k suit une **loi de Bernoulli**

Exemple : Occurrences de différentes espèces (PI@ntNet)

Coordonnées géographiques exactes S_i

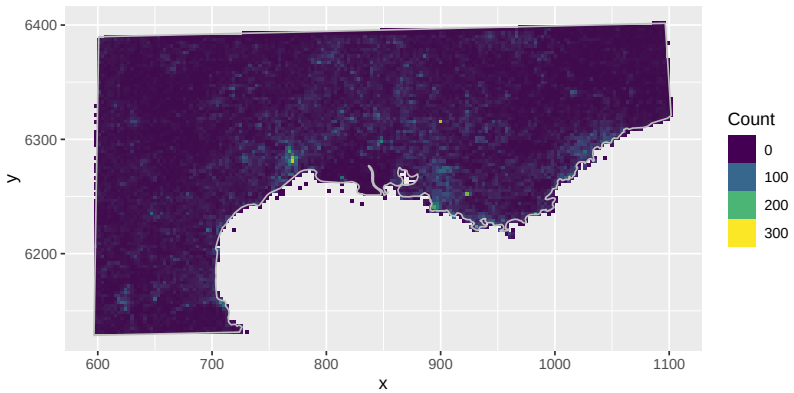


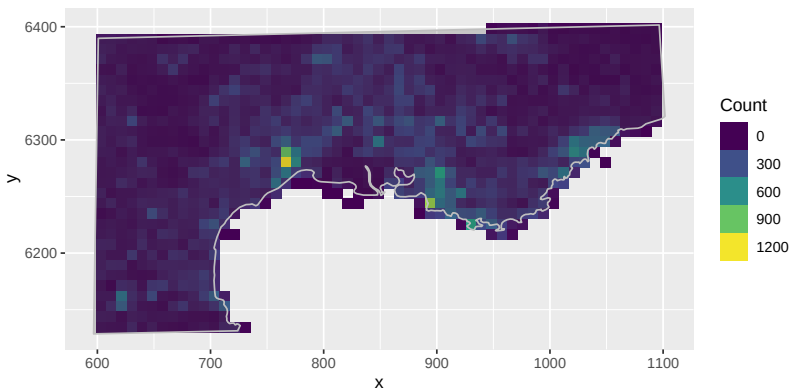
Exemple (suite) : Effet de la résolution spatiale

Agrégation des données non protocolées Pl@ntNet selon deux tailles de pixels

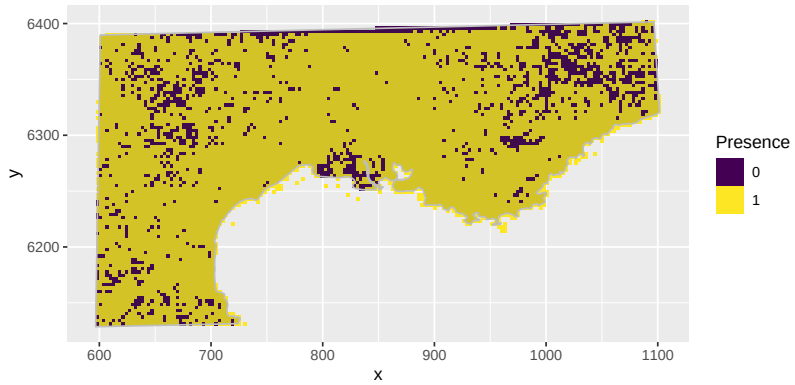
⚠ Attention à la perte d'information, surtout avec les présences-absences

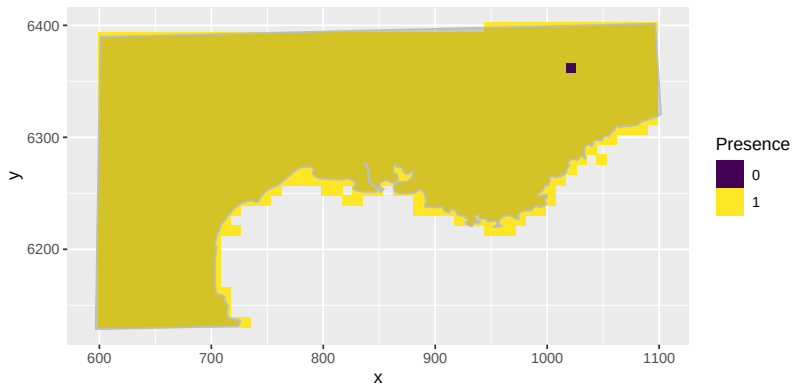
Comptages





Présences-Absences





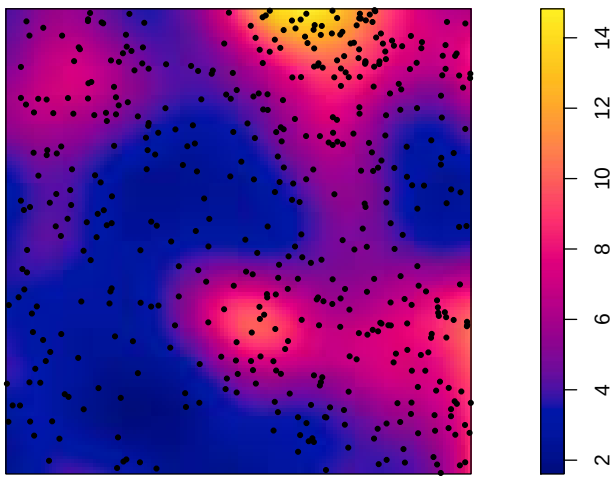
Les processus ponctuels

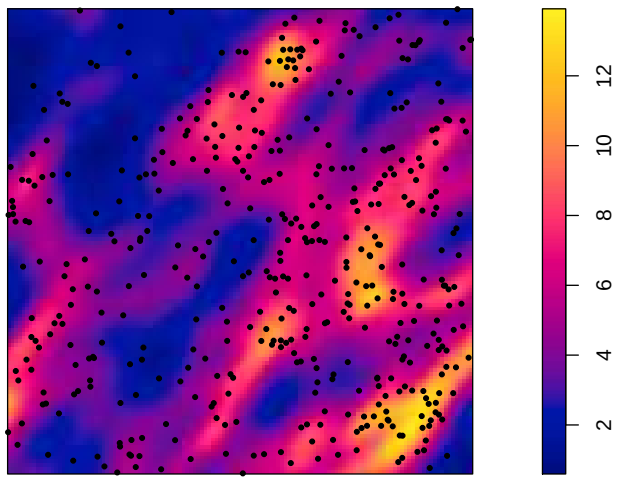
- Observations = coordonnées des occurrences (dans un domaine D)
- Intensité $\lambda(s)$ = Nombre d'occurrences moyen par unité spatiale en s
- **Processus ponctuel de Poisson** (occurrences indépendantes) :
  - Nombre d'occurrences dans $B \subset D$:

$$N(B) \sim \text{Poisson}(\Lambda(B)), \quad \text{avec } \Lambda(B) = \int_B \lambda(s)\mathrm{d}s$$

  - Vraisemblance : $\ell(s_1, \ldots, s_n) = \exp\left(-\int_D \lambda(s)\,\mathrm{d}s\right) \times \prod_{i=1}^n \lambda(s_i)$

    ⚠ En pratique : schéma de discrétisation pour calculer $\Lambda(D) = \int_D \lambda(s)\,\mathrm{d}s$
- **Processus de Cox** si $\lambda(s)$ est un champ aléatoire
  - Variabilité stochastique pour représenter des covariables inconnues
  - **Processus de Cox log-gaussien** : $\log(\lambda(s))$ est un champ gaussien





- Typiquement, un paramètre clé à estimer,
  relié à un "prédicteur linéaire" μ_i via une fonction lien
- μ_i intègre les covariables environnementales et les "effets aléatoires"
  (espace, observateur, année...)

  - **Processus de Poisson :**

    $$\{S_1, S_2, \ldots, S_n\} \sim \mathrm{PPP}(\lambda(s)), \quad \text{avec} \quad \mu(s) = \log(\lambda(s))$$

    (**lien log**)
  - **Comptages** (unité spatiale A_k)

    $$N_k \sim \mathrm{Poisson}(\lambda_k) \quad \text{avec} \quad \mu_k = \log(\lambda_k)$$

    où $\lambda_k = \int_{A_k} \lambda(s)\,\mathrm{d}s$
    (**lien log**)
  - **Présences-absences** (unité spatiale A_k)

    $$Z_k \sim \mathrm{Bernoulli}(p_k) \quad \text{avec} \quad \mu_k = \log(-\log(1 - p_k)),$$

    car

    $$\Pr(Z_k = 1) = \Pr(N_k > 0) = 1 - \exp(-\lambda_k) = 1 - \exp(-\exp(\mu_k))$$

    (**lien log-log complémentaire** "*cloglog*")

Exemple de modèle bivarié (protolé / opportuniste)

Deux variables :

- **Présences-seules non protocolées** :

$$\{S_1, \ldots, S_n\} \sim \mathrm{PPP}(\lambda_{PO}(s))$$

- **Présences-absences protocolées** (plaçettes localisées en s_ℓ) :

$$Z(s_\ell) \sim \mathrm{Bernoulli}(p_{PA}(s_\ell))$$

Modèle avec effets aléatoires spatiaux :

$$\log\left(\lambda_{PO}(s)\right) = \underbrace{\alpha_{PO}}_{\text{Intercept}} + \beta_1 \times \underbrace{x(s)}_{\text{Elevation}} + \underbrace{W_1(s)}_{\text{specific Gaussian field}} + \underbrace{W_0(s)}_{\text{Gaussian field for sampling effort}}$$

$$\mathrm{cloglog}\left(p_{PA}(s)\right) = \underbrace{\alpha_{PA}}_{\text{Intercept}} + \beta_1 \times \underbrace{x(s)}_{\text{Elevation}} + \underbrace{W_1(s)}_{\text{specific Gaussian field}}$$

avec des champs spatiaux W_0, W_1 supposés gaussiens (a priori)

$\Rightarrow$ Implémentation sous forme d'un **modèle bayésien hiérarchique**

Les modèles bayésiens hiérarchiques (rappel)

Les **modèles bayésiens hiérarchiques** emboîtent **trois couches**:

Observations
↑
Processus latent
↑
Hyperparametres

- **Observations**
  avec leur modèle de **vraisemblance** (PPP / Poisson / Bernoulli)

  dépendent du

- **Processus latent** (champ gaussien) modélisant les tendances et dépendances,

  et tous les deux dépendent des

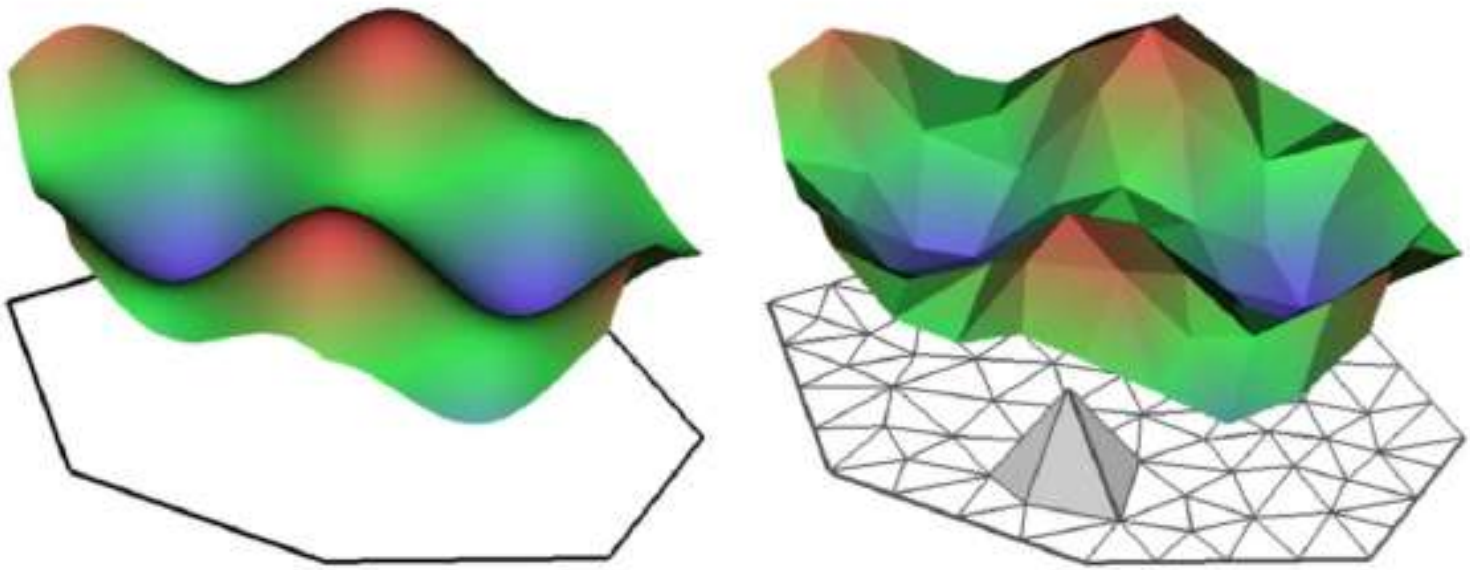
- **Hyperparamètres** : variances, portées spatiales...

Estimation des paramètres via l'**inférence bayésienne approximative** (MCMC, INLA...)

Représentation des champs spatiaux

- **Discrétisation :** construction d'un vecteur gaussien $(W(s_1), \ldots, W(s_m))$ pour certains points de support $(s_1, \ldots, s_m)$
- **Interpolation** entre ces points via des fonctions de base
  ⚠ Une représentation précise nécessite beaucoup de points de support
- **Calculs numériques allégés** via des représentations de champs gaussiens markoviens (matrices de précision creuses grâce à l'approche SPDE)

Implémentation : via les éléments finis (= fonctions "pyramides")



SPDE (Stochastic Partial Differential Equation)

Résultat théorique (Whittle, 1954) :

Les champs gaussiens $W(s)$ avec fonction de **covariance de Matérn** sont solution de

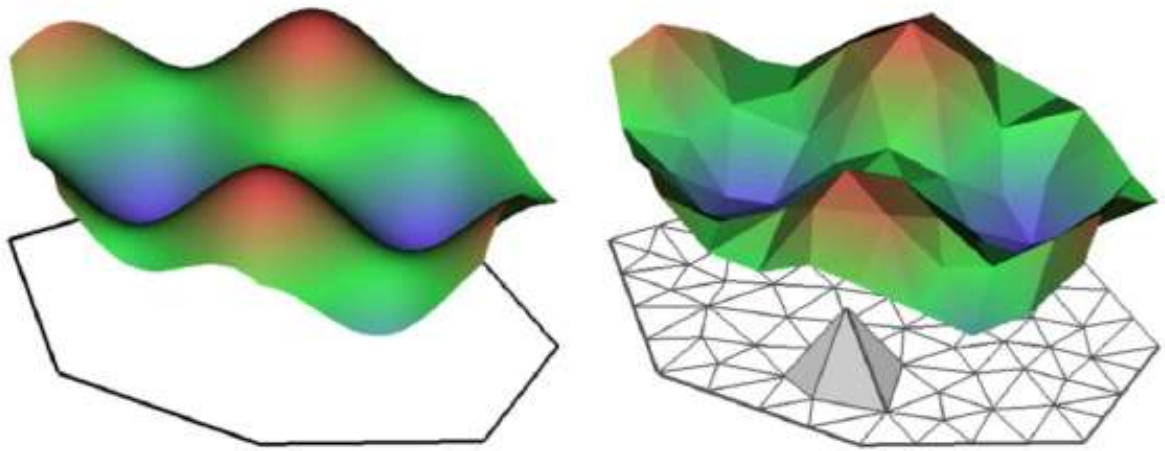
$$(\kappa - \Delta)^{\alpha/2} W(s) = \tau \varepsilon(s), \quad \alpha = \nu + d/2, \quad s \in \mathbb{R}^d$$

avec un bruit blanc gaussien $\varepsilon(s)$

(Paramètres de la Matérn : régularité $\nu \geq 0$, portée $\propto \kappa^{-1}$, variance $1/\tau^2$)

Utilité pratique (Lindgren et al. 2011) :

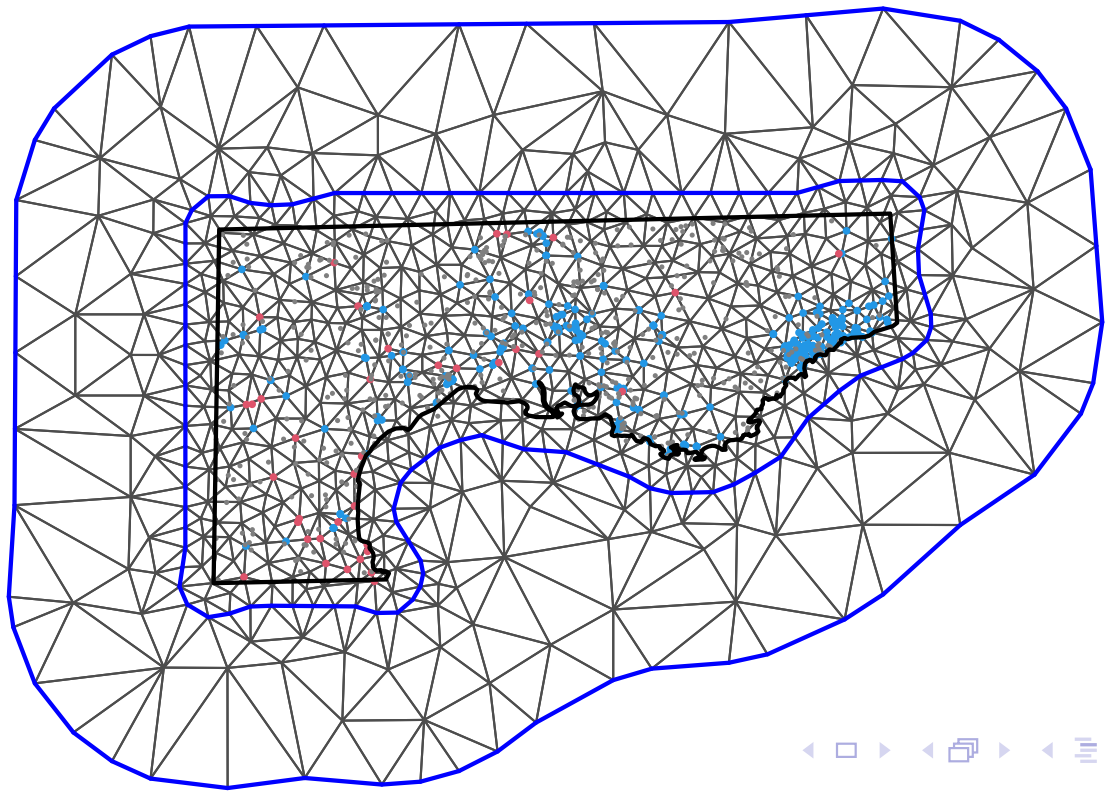
- Solution approximative $(W(s_1), \ldots, W(s_m)) \sim \mathcal{MVN}(\mathbf{0}, \boldsymbol{Q}^{-1})$
- **Matrice de précision creuse $\boldsymbol{Q}$**
- Calculs matriciels rapides quand m est grand



Triangulation ("mesh") pour l'approche SPDE

Construire le "mesh", une étape pratique importante (`inla.mesh....`) :

- Maillage relativement dense dans le domaine d'étude D (zone intérieure)
- Possibilité d'un maillage plus dense dans les zones avec beaucoup d'occurrences
- Zone d'extension pour un bord extérieur éloigné de D afin d'éviter des effets de bords



L'implémentation R-INLA "classique"

INLA = Integrated Nested Laplace Approximation (Rue et al. 2009)

- Estimation des modèles bayésiens hiérarchiques basés sur les **processus latent gaussiens**
- Un cadre adapté aux **processus de Cox log-gaussiens** !
  (Illian et al. 2012, AOAS ; Simpson et al. 2016, Biometrika ; Laxton et al. 2022, MEE)
- Utilisation des **approximations de Laplace** (déterministes) pour calculer les lois a posteriori (effets fixes β, champs W, hyperparamètres...)
- **Penalized-Complexity (PC) priors** pour les hyperparamètres
  - Champs spatiaux : le "baseline" est un champs constant zéro
  - Loi exponentielle sur l'écart-type et la portée spatiale
  - En pratique, spécification de (u, α) dans $\Pr(\text{hyperparametre} <> u) = \alpha$

inlabru : focus sur l'écologie

https://cran.r-project.org/web/packages/inlabru/index.html

inlabru: an R package for Bayesian spatial modelling from ecological survey data

Fabian E. Bachl[1] | Finn Lindgren[1] | David L. Borchers[2] | Janine B. Illian[2]

[1]School of Mathematics, University of Edinburgh, Edinburgh, UK

[2]Centre for Research into Ecological and Environmental Modelling, School of Mathematics and Statistics, University of St Andrews, The Observatory, St Andrews, Fife, UK

Correspondence
David L. Borchers
Email: dlb@st-andrews.ac.uk

Handling Editor: Robert Freckleton

Abstract

1. Spatial processes are central to many ecological processes, but fitting models that incorporate spatial correlation to data from ecological surveys is computationally challenging. This is particularly true of point pattern data (in which the primary data are the locations at which target species are found), but also true of gridded data, and of georeferenced samples from continuous spatial fields.
2. We describe here the `R` package `inlabru` that builds on the widely used `R-INLA` package to provide easier access to Bayesian inference from spatial point process, spatial count, gridded, and georeferenced data, using integrated nested Laplace approximation (INLA, Rue et al., 2009).
3. The package provides methods for fitting spatial density surfaces and estimating abundance, as well as for plotting and prediction. It accommodates data that are points, counts, georeferenced samples, or distance sampling data.
4. This paper describes the main features of the package, illustrated by fitting models to the gorilla nest data contained in the package `spatstat` (Baddeley, & Turner, 2005), a line transect survey dataset contained in the package `dsm` (Miller, Rexstad, Burt, Bravington, & Hedley, 2018), and to a georeferenced sample from a simulated continuous spatial field.

KEYWORDS
Bayesian inference, georeferenced data, point process, spatial count, spatial modelling

`inlabru` : plus d'ergonomie et de fonctions

- **Syntaxe plus intuitive** pour coder les modèles et leurs composantes
- Fonctions pour les **processus de Cox log-gaussiens**
  - Vraisemblance pour les coordonnées géographiques exactes des occurrences
  - Calcul automatique de $\Lambda(D) = \int_D \lambda(s)\,\mathrm{d}s$
- Nouvelles fonctionnalités, par exemple pour appliquer des **transformations nonlinéaires** à un champ gaussien dans le prédicteur linéaire :

$$\mu(s) = \beta_0 + g(W(s)) + \ldots$$

- **Estimations et prédictions plus variées** et plus faciles à obtenir, notamment via la simulation a posteriori

Estimations/simulations a posteriori et cartographie

- **Les sorties "standard" :**
  - Moyenne / médiane / écarts-types / quantiles a posteriori
  - Disponible pour les hyperparamètres, les effets fixes, le prédicteur linéaire, les "fitted values"
- **Autres estimations a posteriori "sur mesure" :** via un calcul Monte–Carlo en simulant selon le modèle estimé

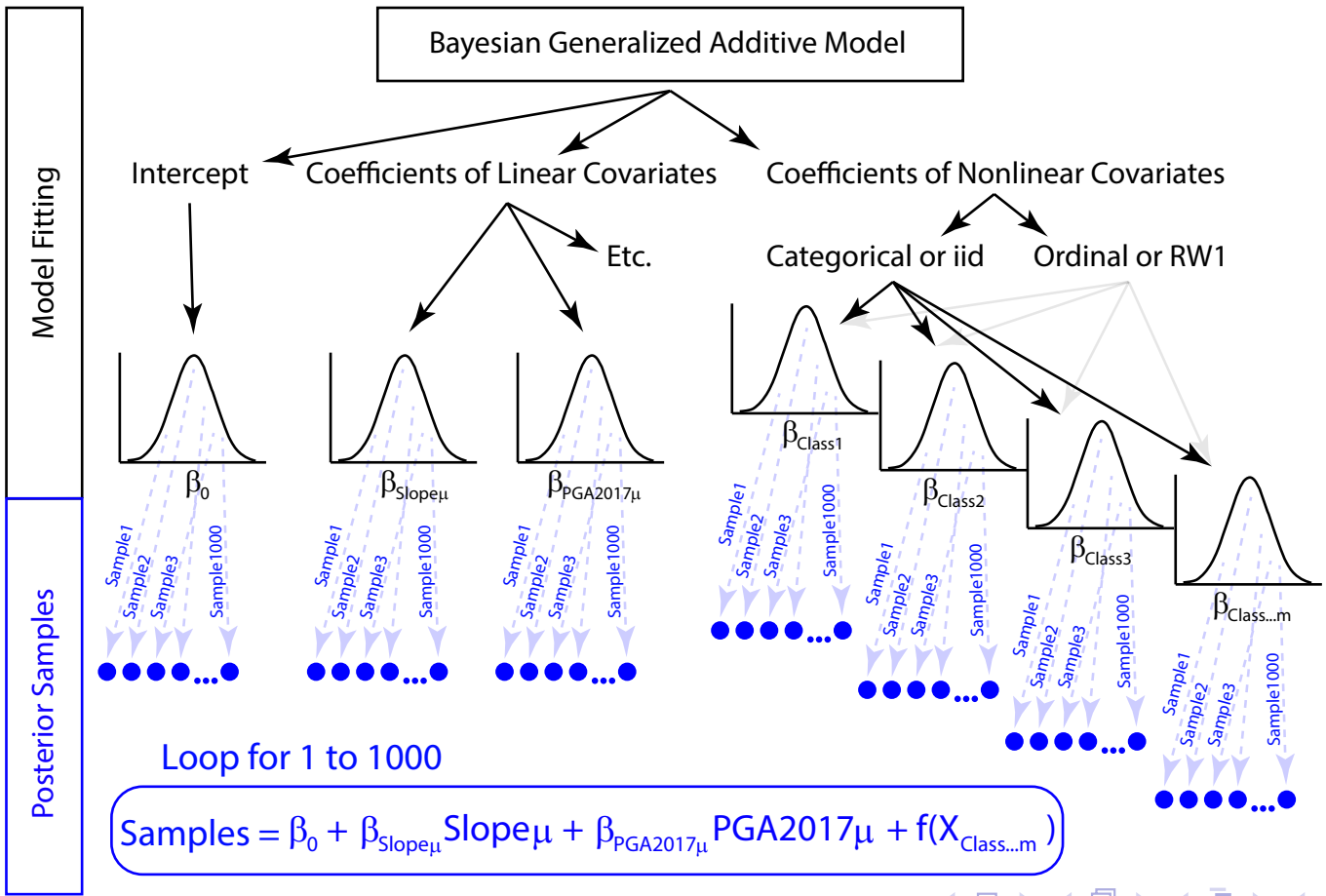
Simulation a posteriori

Estimation Monte–Carlo de quantités a posteriori en simulant, de façon répétée, le processus latent selon le modèle estimé.

$\Rightarrow$ Propagation des incertitudes a posteriori

(**Avec inlabru :** `generate.bru(fit,...)` et `predict.bru(fit, ...)`)

Illustration :



Pour finir

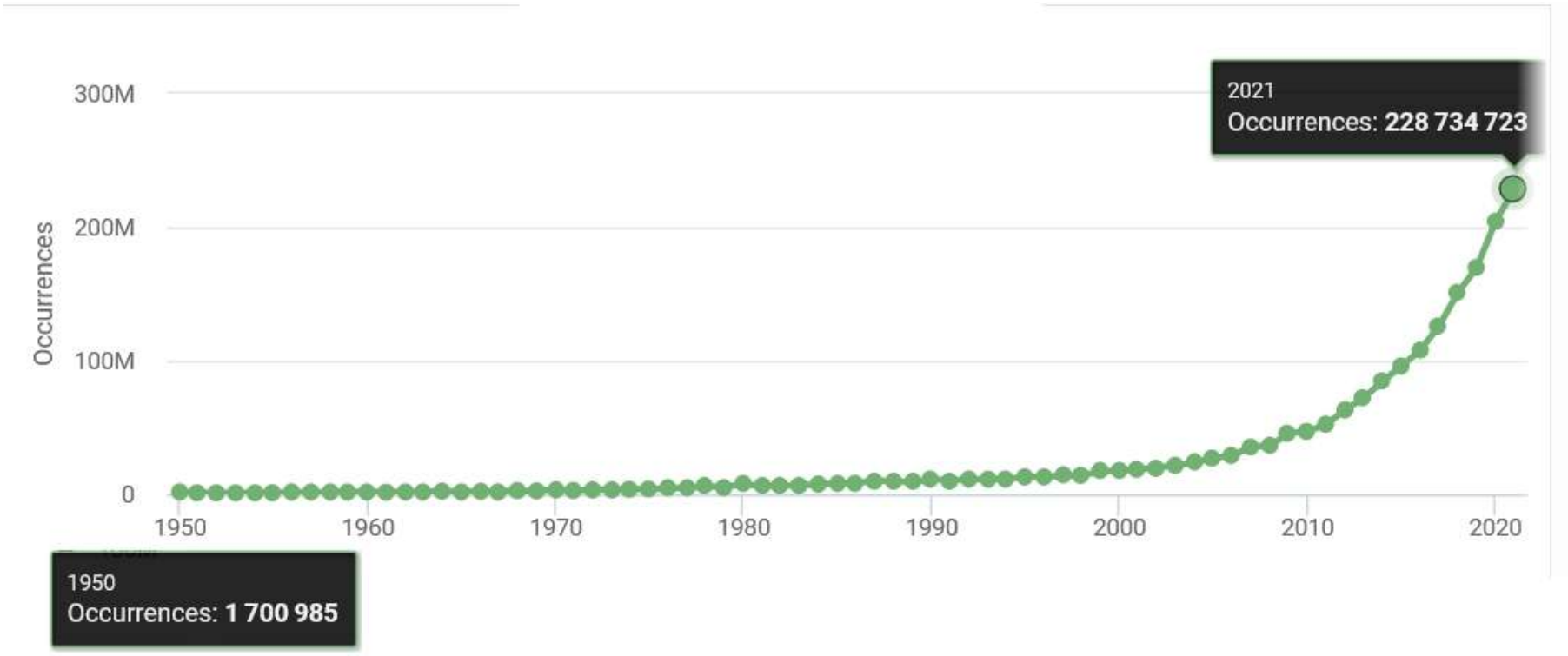
- `inlabru` simplifie la vie du modélisateur mais ne fournit pas des approches "clique-bouton"
- Bonne intégration des objets spatiaux de type `sp`
- Un package très complet pour modéliser les "présences-seules" avec les processus de Cox log-gaussiens
- Idéalement, on commence à construire des modèles simples et facilement interprétables avant de les complexifier progressivement

C'est en forgeant qu'on devient forgeron.

Données de présence-seule opportunistes et biais d’échantillonnage dans Pl@ntNet

Christophe Botella

Données de présence-seule massives, une opportunité



Pl@ntNet, identification automatique des plantes par l'image

Pl@ntNet

Pl@ntNet, identification automatique des plantes par l'image

Species ranked per confidence value:

A

1) *Acer monspessulanum* L. – 99.6
...

Species ranked per confidence value:

B

1) *Acer campestre* L. – 7.3
2) *Acer monspessulanum* L. – 5.9
...

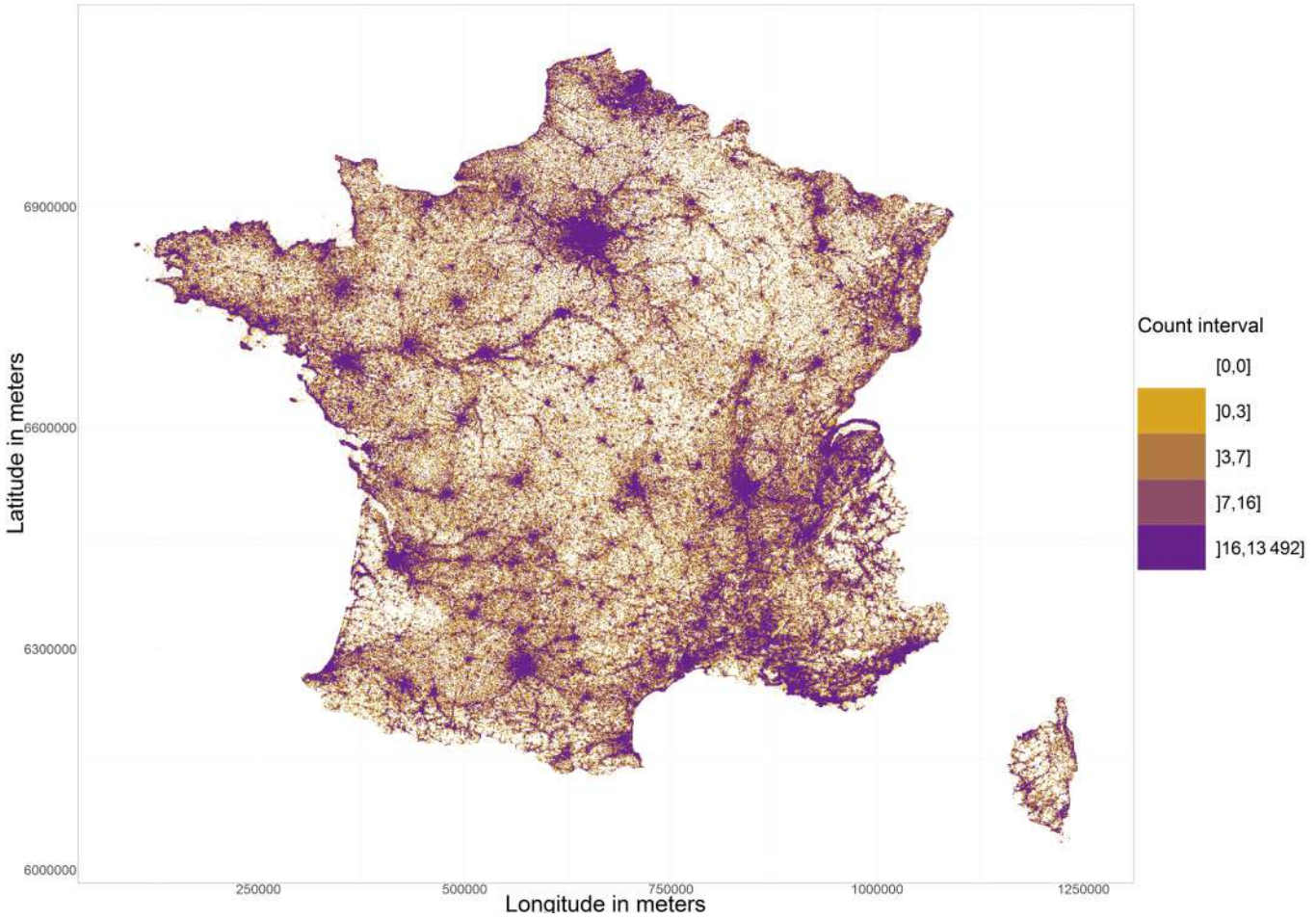
C

1) *Acer monspessulanum* L. – 21.7
2) *Acer pseudoplatanus* L. – 20.8
3) *Acer saccharinum* L. – 8.4
...

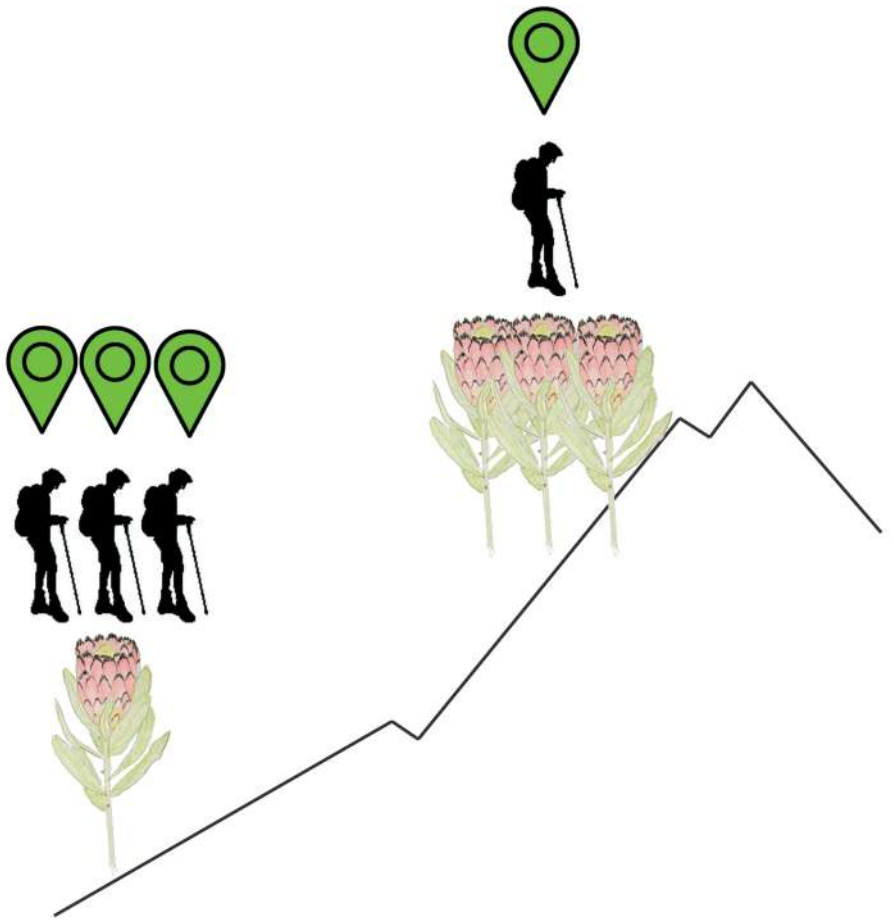
D

1) *Acer monspessulanum* L. – 44.3
2) *Acer campestre* L. – 31.7
3) *Hedera helix* L. – 1.6
...

Distribution spatiale des données Pl@ntNet



Biais d’échantillonnage spatial des modèles de niche



Biais d'échantillonnage spatial des modèles de niche

- Formalisation biais avec processus ponctuels de Poisson
  - Modèle SDM naïf : $Z \sim PP(f_\theta \circ x)$
  - Réalité : $Z \sim PP(s \times f \circ x)$
- Estimateur SDM biaisé : $\mathbb{E}(\hat{f}_\theta) \approx s_x \times f$

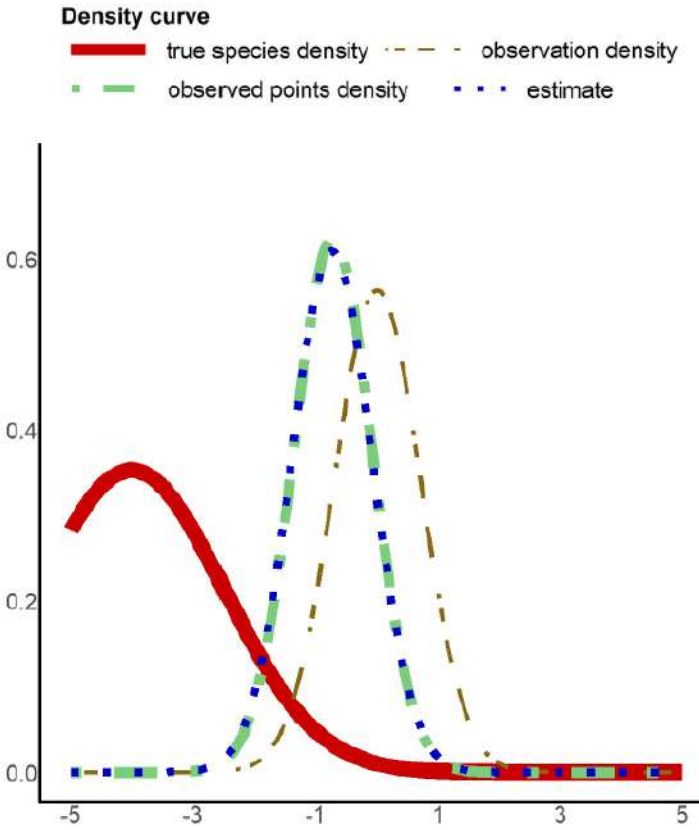


Figure – Distribution d'une espèce vs gradient environnemental, effort d'échantillonnage et estimateur SDM naïf.

Correction Target-Group Background

- Approximer effort avec les points d'un group cible d'espèces (TG)
- $\mathbb{E}(\hat{\theta}_{TGB}) =$

$$\underset{\theta}{\text{argmin}}\ D_{KL}(f s_x || f_\theta s_x tg)$$ [a]

i.e. $\mathbb{E}(f_{\hat{\theta}_{TGB}}) \approx f/tg$

- Travaux connexes :
  - Estimation jointe de l'effort d'échantillonnage et distributions des espèces. [b].

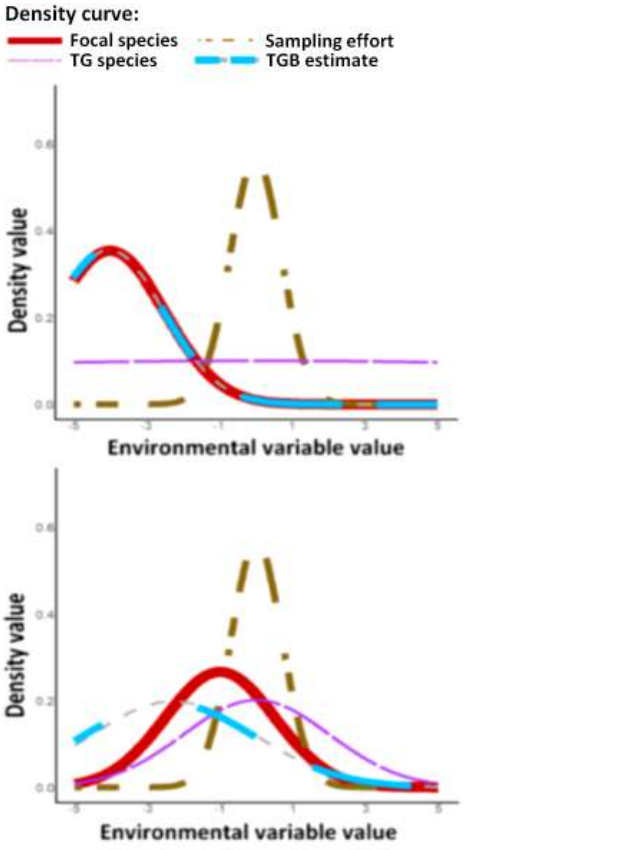


Figure – Distribution d'une espèce vs gradient environnemental, effort d'échantillonnage et estimateur TGB selon deux TG.

a. Botella et al., 2020.
b. Botella et al., 2021.

Atelier 1

Introduction à inlabru et R-INLA

Christophe Botella, Florian Lasgorceux, Juliette Legrand, Thomas Opitz, Julien Papaïx,

4/14/23

Table of contents

1 Libraries

We first install and load the `R-INLA` and `inlabru` packages. See https://www.r-inla.org/home and https://inlabru-org.github.io/inlabru/index.html for more details.

```
# install.packages("INLA",repos=c(getOption("repos"),
#                  INLA="https://inla.r-inla-download.org/R/stable"), dep=TRUE)
# install.packages("inlabru")
library(INLA)
library(inlabru)
```

We then load other libraries useful for the following of this tutorial.

```
library(ggplot2)
library(raster)
library(rgeos)
library(sf)
library(tidyverse)
library(viridis) #for viridis color palette (from dark blue to bright yellow)
```

2 Data used in this tutorial

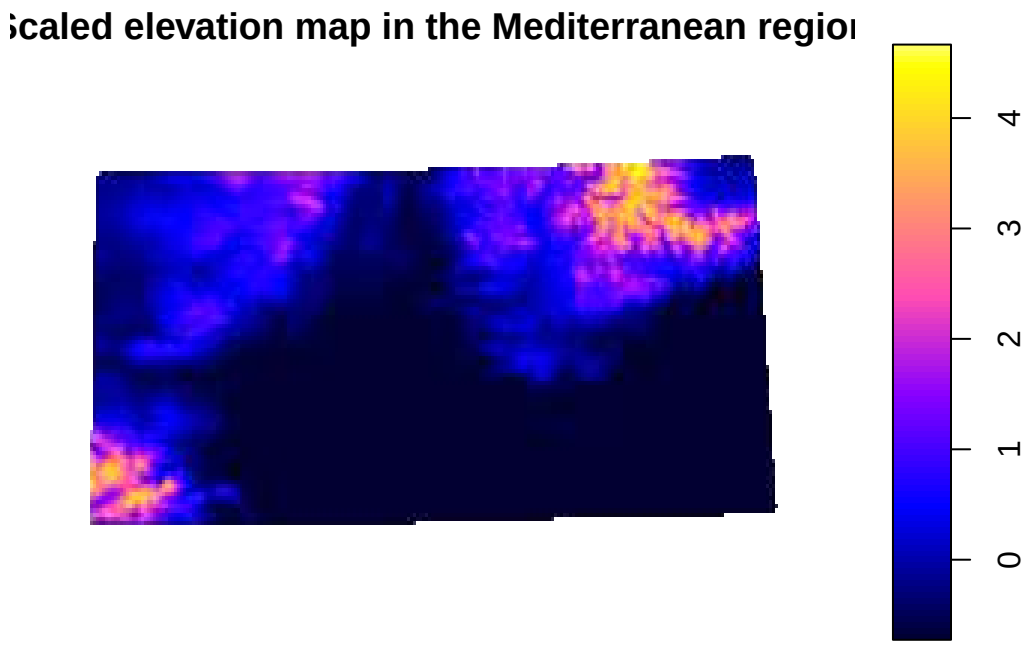
This tutorial focuses on modeling opportunistic data using the `R-INLA` and `inlabru` packages. We will be using presence-only observations from Pl@ntNet in the Mediterranean region. Since the use of such spatial data is always technical and time-consuming, the data has been pre-processed and can be downloaded from the `data.RData` file provided. The file contains the following components:

- `ctab`: a `data.frame` used in the `f.tg` function. See the tutorial 2.
- `domaineSP`: the spatial domain of study in `SpatialPolygonDataFrame` format, which is the Mediterranean region.
- `elevSG`: the map of elevation of the Mediterranean region in `SpatialGridDataFrame` format.
- `PA.sp`: the presence-absence data in `SpatialPointsDataFrame` format. See the tutorial 2.
- `PO`: the presence-only data from Pl@ntNet in the Mediterranean region in the usual `data.frame` format.
- `rTGB`: a `raster` with the number of observations of the computed target group of species. See the tutorial 2.
- `TGdf`: a `data.frame` with the species computed by the Target-Group Background approach. See the tutorial 2.
- `f.elev`: a function that allows computing the elevation at any points `(x,y)` using `elevSG`.
- `f.tg`: a function that allows computing the number of observations of the target-group of species of the cell including `(x,y)` using `rTGB`. See the turorial 2.

Note that the code to obtain the `data.RData` file is not necessary for this tutorial, but it is provided in the file `Annex.R`. We use Lambert-93 as the spatial projection, and the distance is measured in kilometers (the "x" and "y" columns). Also note that the elevation values are scaled.

```
load("data.RData")

# Elevation map
```

```
terra::plot(elevSG, main = "Scaled elevation map in the Mediterranean region")
```




```
summary(elevSG$ASTER_Elevation)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
-0.725  -0.724  -0.448   0.000   0.418   4.666    8637
```

3 First model using `inlabru`

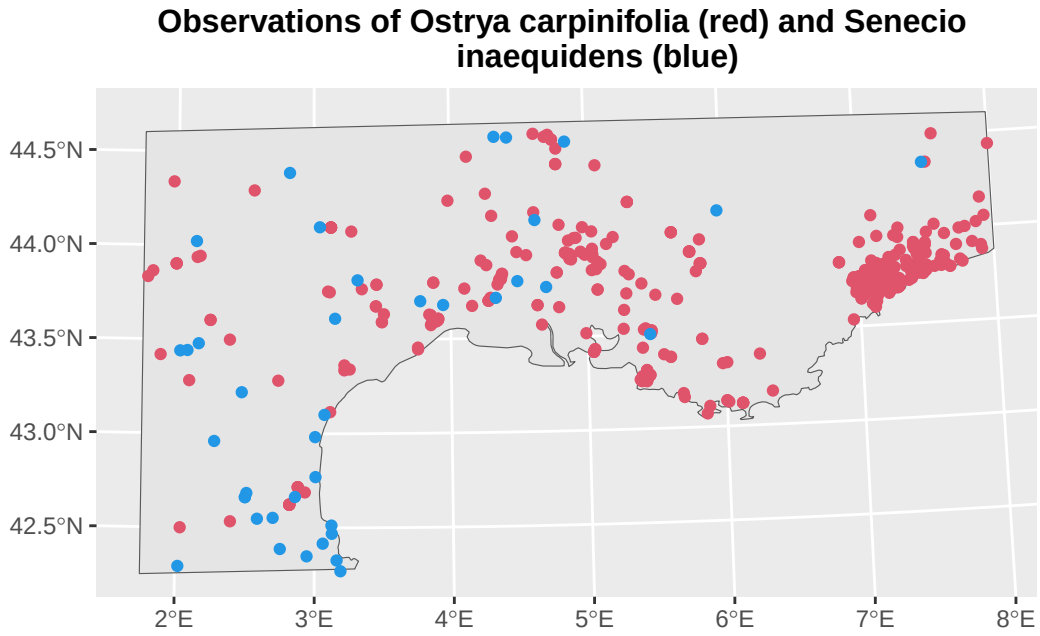
An important function of `inlabru` is the `lgcp` function for fitting Log-Gaussian Cox Processes (LGCPs) to presence-only data. Below, we provide detailed steps for building an LGCP using the Pl@ntNet data.

3.1 Choose a species of study

We begin by selecting two species of interest that have contrasting spatial distributions across the Mediterranean region: *Ostrya carpinifolia* with 413 observations and *Senecio inaequidens* with 39 observations. When calling the `lgcp` function, the geographic coordinates must be named "x" and "y," and the data must be in the `SpatialPointsDataFrame` (spdf) format of the `sp` package.

```
# Filter the PO data by the name of the studied species
PO_Ostrya_carpinifolia = PO %>% filter(species == "Ostrya carpinifolia")

# Convert the df to a spdf
PO_Ostrya_carpinifolia <-
  SpatialPointsDataFrame(coords = PO_Ostrya_carpinifolia[,c("x", "y")],
```

```
                        data = PO_Ostrya_carpinifolia,
                        proj4string = CRS(domaineSP@proj4string@projargs))


# Filter the PO data by the name of the studied species
PO_Senecio_inaequidens = PO %>% filter(species == "Senecio inaequidens")
# Convert the df to a spdf
PO_Senecio_inaequidens <-
  SpatialPointsDataFrame(coords = PO_Senecio_inaequidens[,c("x", "y")],
                        data = PO_Senecio_inaequidens,
                        proj4string = CRS(domaineSP@proj4string@projargs))

# Plot of the domain with the observation points
ggplot() + geom_sf(data=st_as_sf(domaineSP)) +
  geom_sf(data = st_as_sf(PO_Ostrya_carpinifolia), color = 2) +
  geom_sf(data = st_as_sf(PO_Senecio_inaequidens), color = 4) +
  labs(title = "Observations of Ostrya carpinifolia (red) and Senecio
       inaequidens (blue)") +
  theme(plot.title = element_text(size = 12, face = "bold", hjust = 0.5))
```



3.2 Build a mesh with INLA to implement spatial Gaussian fields

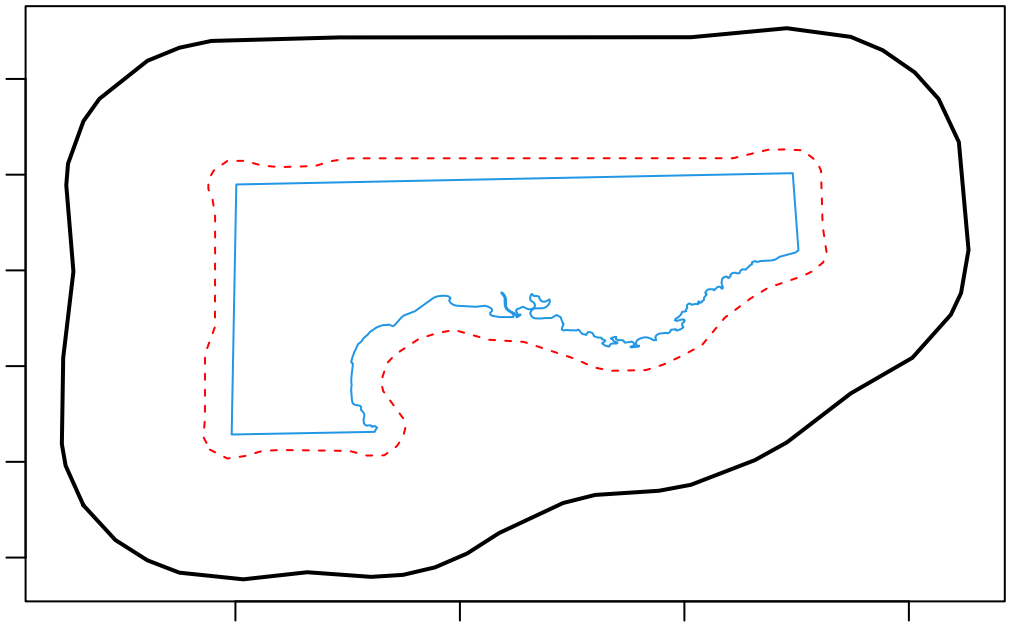
`INLA` and `inlabru` use a space triangulation method to estimate spatial Gaussian effects with a Matérn covariance function. The spatial Gaussian random field is computed at the mesh nodes by resolving a Stochastic Partial Differential Equation (SPDE), while it is computed elsewhere by interpolation. The mesh definition is based on a trade-off between the finer spatial scale of the spatial effect (higher resolution) and the number of nodes (lower resolution). Therefore, the mesh nodes do not have to coincide with the

observation locations of species occurrences, but it usually makes sense to use these locations as a starting point to generate the mesh. The mesh generation is done automatically but according to some constraints that we can impose on the form of the mesh, such as the minimum and maximum distance between two adjacent mesh nodes. Below, we present how to build a mesh from the data.

First, we create a matrix `coordDOM` with regular coordinates of the spatial domain. Next, we define the boundaries of the domain used for computing the spatial latent effect with the SPDE approach. The boundary is automatically calculated as a nonconvex envelope of a set of points that we use to characterize the study domain. Here, we construct this set of points by including the vertices of the bounding polyon and a coarse grid covering the study domain. Note that this coarse grid has no other purpose than help us generate an accurate boundary for the SPDE model of the study domain. Generally, we compute an internal boundary and an external boundary with different resolutions. The role of the extension zone between the interior and the exterior boundary is to avoid that boundary effects at the exterior boundary have a strong influence on the behavior of the spatial field in the interior zone, in which we would like to have an approximately stationary behavior of the Gaussian prior field. To keep the computational cost to a minimum, we try to avoid having too many mesh nodes, and therefore we can choose a coarser resolution of the mesh in the extension zone, for instance by allowing the maximum distance between two mesh nodes (`max.edge`) to be larger. We can specify two values for `max.edge`, the first one for the interior zone and the second one for the extension zone.

The `inla.mesh.2d` function creates a triangle mesh based on initial point locations, specified or automatic boundaries, and mesh quality parameters.

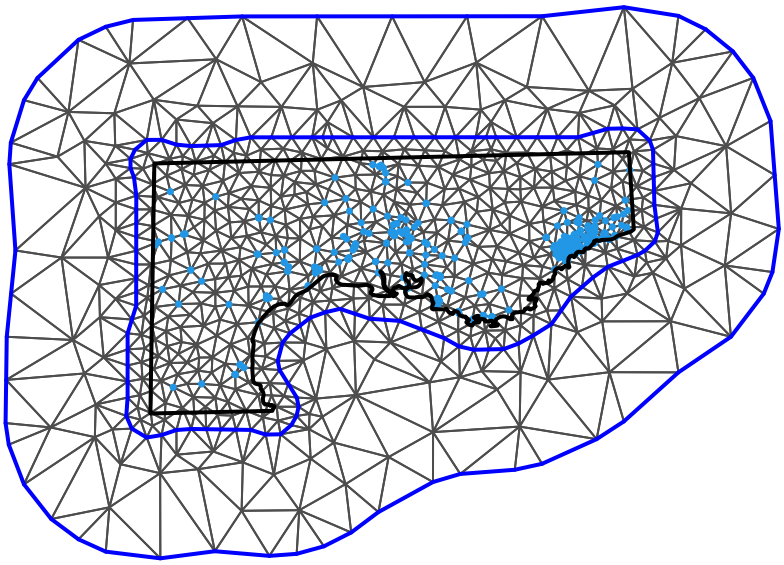
```
# Regularly sample 1000 points in the domain polygon
tmp = spsample(domaineSP@polygons[[1]], n = 1000, type = "regular")

# Coordinates to build the mesh
coordDOM <- rbind(domaineSP@polygons[[1]]@Polygons[[1]]@coords, tmp@coords)
rm(tmp)

# Boundaries
bndint <- inla.nonconvex.hull(coordDOM, convex=-.05)
bndext <- inla.nonconvex.hull(coordDOM, convex=-.3)

# Plot the domain and the boundaries
par(mar=rep(1,4),mfrow=c(1,1))
plot(rbind(bndext$loc,bndext$loc[1,]),type="l",lwd=2)
lines(rbind(bndint$loc, bndint$loc[1,]), pch = 19, cex = .05, col = "red",
      lwd = 1, lty=2)
plot(domaineSP,add=T,border=4)
```



```
# Use of inla.mesh.2d
mesh = inla.mesh.2d(loc=rbind(coordinates(PO_Ostrya_carpinifolia)),
                    boundary = list(int = bndint, out = bndext),
                    max.edge = c(25, 100), cutoff = c(5),
                    crs = "+proj=lcc +lat_0=46.5 +lon_0=3 +lat_1=49 +lat_2=44
                    +x_0=700000 +y_0=6600000 +ellps=GRS80 +units=km +no_defs")

# Number of nodes of the mesh
mesh$n
```

```
[1] 736
```

```
# Plot the mesh
par(mar=rep(1,4),mfrow=c(1,1))
plot(mesh, main = "", asp = 1)
plot(PO_Ostrya_carpinifolia,add=T,col=4,pch=16,cex=0.5)
#plot(PO_Senecio_inaequidens,add=T,col=2,pch=16,cex=0.5)
plot(domaineSP,add=T,border=1,lwd=2)
```



3.3 How to fit the model

Our first generic LGCP model concerns the spatial distribution of the species *Ostrya carpinifolia* using presence-only data. We model the intensity of the studied species i at the point s as follows:

$$\ln\left(\Lambda_i(s)\right) = \underbrace{\alpha_i}_{\text{Intercept}} + \underbrace{x(s)}_{\text{Elevation}}\ \beta_i + \underbrace{W_i(s)}_{\text{Gaussian field}}$$

The linear predictor $\ln(\Lambda_i)$ captures a linear effect of elevation through β_i, while the Gaussian random field W_i represents an unmeasured covariate associated with the species' distribution.

We choose the Matérn covariance function for the Gaussian random field because it can be easily fitted in `INLA` using an SPDE. The Matérn covariance in `INLA` depends on three parameters: a fractional order parameter *alpha* in the SPDE controlling the smoothness of the solution, a standard deviation parameter *sigma*, and a spatial correlation parameter known as the *range*. We specify these parameters in our model by selecting a penalized complexity prior using the `inla.spde2.pcmatern` function. For Gaussian fields defined over a discrete mesh at a latent layer of the model, the choice of the *alpha* parameter is usually less important, and here we set it to the fixed value 2, for which the numerical algorithms implemented by `INLA` are known to be particularly simple and efficient. For more details, please refer to the introduction to spatial models with `INLA` in chapter 7 at https://becarioprecario.bitbucket.io/inla-gitbook/ch-spatial.html.

```
matern <- inla.spde2.pcmatern(mesh,
                              alpha = 2, # fractional operator which is related
                              #to the smoothness of the Gaussian field
                              prior.sigma = c(1, 0.5), # P(sigma > 1) = 0.5
                              prior.range = c(100, 0.9)) # P(range < 100) = 0.9
```

We then specify in the `cmp` object the formulation of the model with all the variables and covariates involved, such as the coordinates of our observations, in relation to all the latent effects, including an

intercept, elevation, and the Gaussian random field.

```
cmp <- coordinates ~ Intercept(1) +
  elev(main = f.elev(x,y), model = "linear") +
  W1(coordinates, model = matern)
```

Finally, we fit the Log-Gaussian Cox Process to the data using the `lgcp` function of the `inlabru` package. This function requires the model components defined earlier (`cmp`), the dataset (`PO_Ostrya_carpinifolia`), the spatial domain where the data were collected (`domainSP`), the mesh (`mesh`) where the model will be evaluated, and several options can be set manually to control the behavior of the INLA algorithm. In particular, we set `int.strategy = "eb"`, which leads to relatively fast but still reliable approximations of the posterior model that we estimate. Again, we refer to general references on `INLA` for more details.

```
T1 <- Sys.time()
fit <- lgcp(cmp, PO_Ostrya_carpinifolia,
            samplers = domaineSP,
            domain = list(coordinates = mesh),
            options = list(control.inla = list(int.strategy = "eb")))
T2 <- Sys.time()

fit1 = fit  # keep the output
summary(fit)
```

```
inlabru version: 2.7.0
INLA version: 22.12.16
Components:
Intercept: main = linear(1)
elev: main = linear(f.elev(x, y))
W1: main = spde(coordinates)
Likelihoods:
  Family: 'cp'
    Data class: 'SpatialPointsDataFrame'
    Predictor: coordinates ~ .
Time used:
    Pre = 0.509, Running = 0.986, Post = 0.0349, Total = 1.53
Fixed effects:
            mean    sd 0.025quant 0.5quant 0.975quant   mode kld
Intercept -7.138 0.349     -7.822   -7.138     -6.453 -7.138   0
elev      -1.043 0.226     -1.487   -1.043     -0.599 -1.043   0

Random effects:
  Name    Model
    W1 SPDE2 model

Model hyperparameters:
              mean    sd 0.025quant 0.5quant 0.975quant  mode
Range for W1 58.94 10.14      42.05    57.88      82.21 55.43
Stdev for W1  1.67  0.15       1.40     1.66       2.00  1.64

Deviance Information Criterion (DIC) ...............: -5886.00
```

```
Deviance Information Criterion (DIC, saturated) ....: NA
Effective number of parameters .....................: -6839.66

Watanabe-Akaike information criterion (WAIC) ...: 1572.95
Effective number of parameters .................: 381.05

Marginal log-Likelihood:  -4095.44
 is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

```r
Tdiff= difftime(T2, T1) ; Tdiff
```

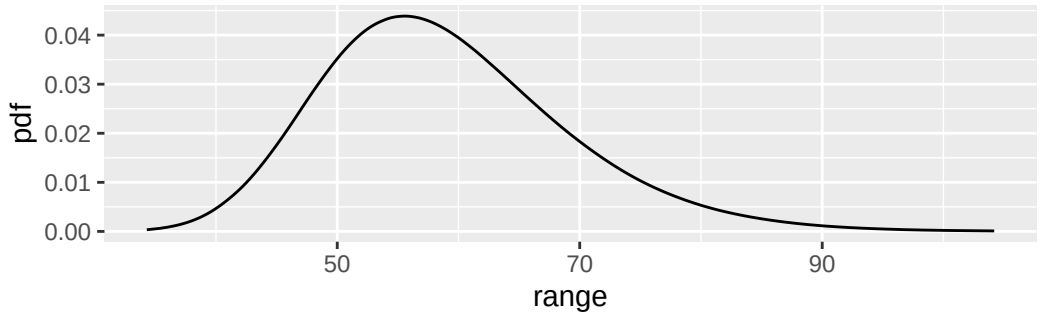
```
Time difference of 3.430972 secs
```

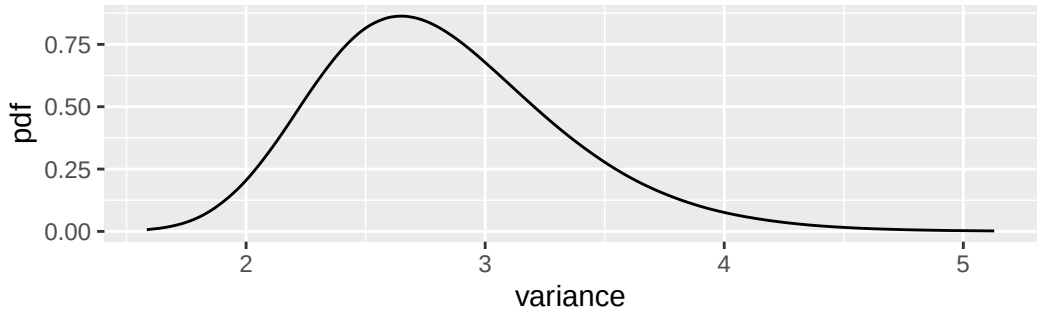
The summary gives the posterior estimates of fixed effects (intercept and elevation) and hyperparameters (standard deviation and range of the Gaussian random field).

3.4 Some plots

We first compute the posterior distribution of the estimated parameters of the spatial field.

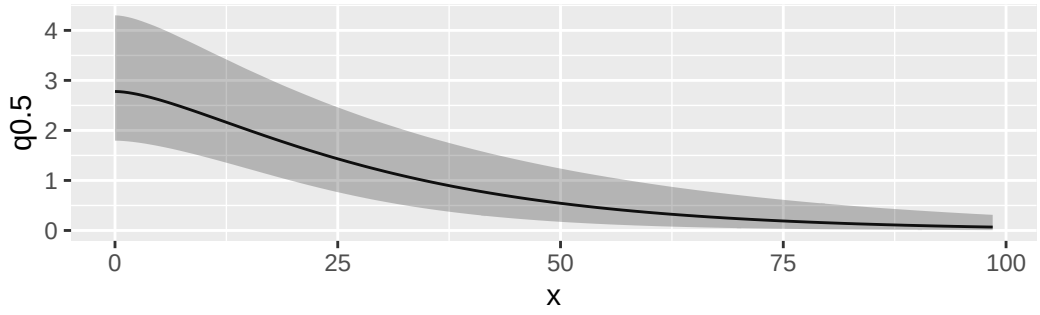
```r
# Compute the posterior of each hyperparameter
spde.range <- spde.posterior(fit, "W1", what = "range")
spde.logvar <- spde.posterior(fit, "W1", what = "variance")

# Plots
range.plot <- plot(spde.range)
var.plot <- plot(spde.logvar)
multiplot(range.plot, var.plot)
```

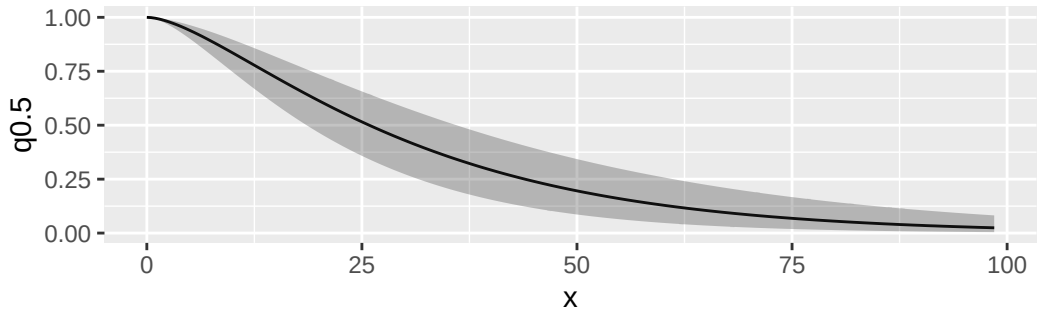




We plot the estimated Matérn covariance and correlation functions.

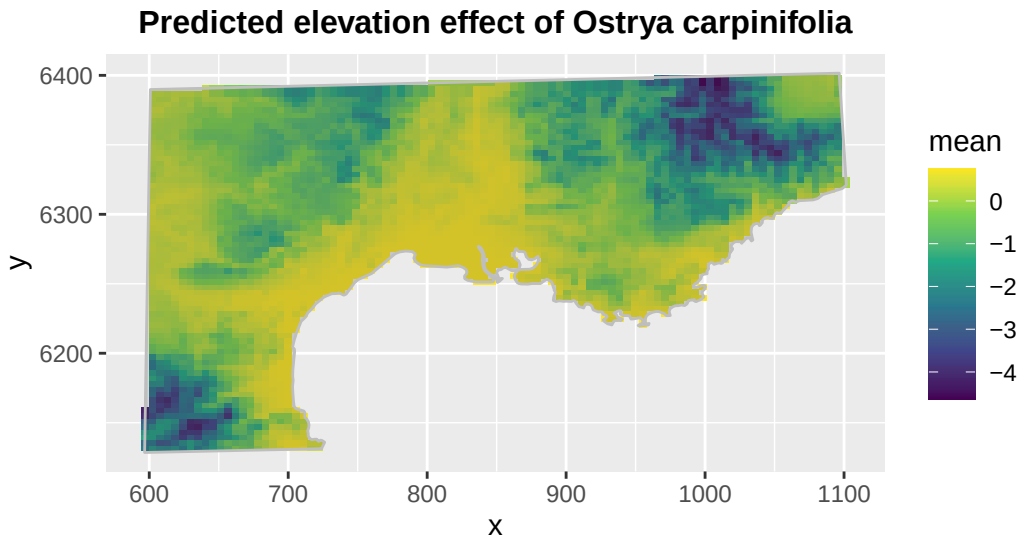
```
corplot <- plot(spde.posterior(fit, "W1", what = "matern.correlation"))
covplot <- plot(spde.posterior(fit, "W1", what = "matern.covariance"))
multiplot(covplot, corplot)
```



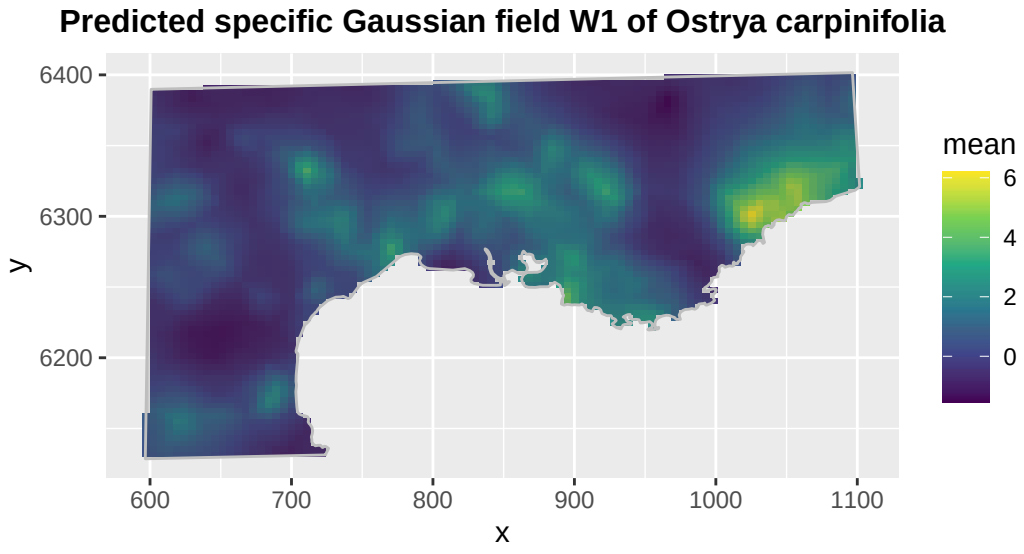


We now compute the contribution of each latent effect and plot them.

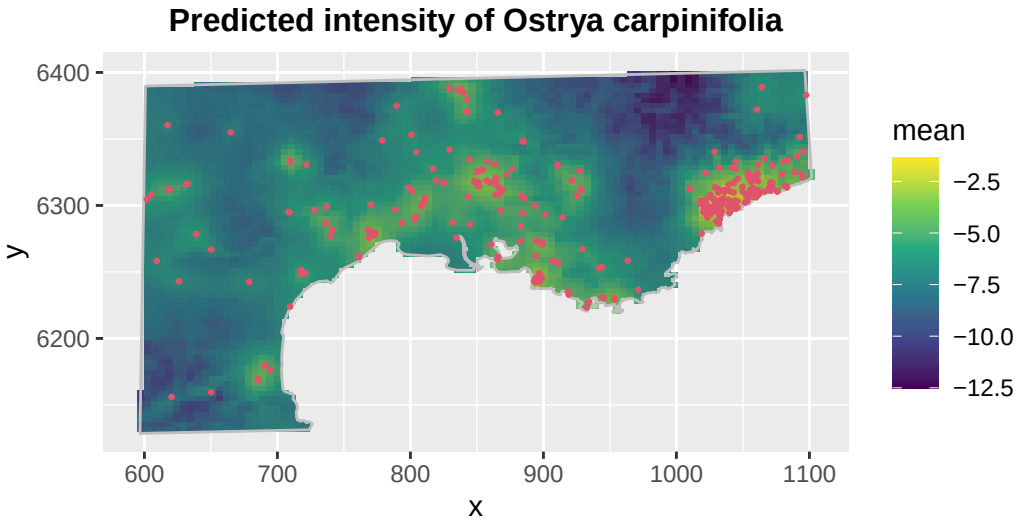
```
pxl <- pixels(mesh, mask=domaineSP)
pr.int.elev <-  predict(fit, pxl, ~ elev)
pr.int.W1 <- predict(fit, pxl, ~ W1)
pr.int.tot <-  predict(fit, pxl, ~ Intercept+elev+W1)


p1=ggplot() +
  gg(pr.int.elev[,"mean"]) +
  gg(domaineSP, color = "grey") +
  scale_fill_continuous(type = "viridis") +
  coord_equal() +
  labs(title = "Predicted elevation effect of Ostrya carpinifolia") +
  theme(plot.title = element_text(size = 12, face = "bold", hjust = 0.5))

p2=ggplot() +
  gg(pr.int.W1[,"mean"]) +
  gg(domaineSP, color = "grey") +
  scale_fill_continuous(type = "viridis") +
  coord_equal() +
  labs(title = "Predicted specific Gaussian field W1 of Ostrya carpinifolia") +
  theme(plot.title = element_text(size = 12, face = "bold", hjust = 0.5))

p3=ggplot() +
  gg(pr.int.tot[,"mean"]) +
  gg(domaineSP, color = "grey") +
  gg(PO_Ostrya_carpinifolia, size = 0.5, alpha = 1, col=2) +
  scale_fill_continuous(type = "viridis") +
  coord_equal() +
  labs(title = "Predicted intensity of Ostrya carpinifolia") +
  theme(plot.title = element_text(size = 12, face = "bold", hjust = 0.5))

# Plot of the elevation estimated effect
p1
```




```
# Plot of the estimated random spatial effect
p2
```



```
# Plot of the linear predictor
p3
```



We can also estimate the abundance. We provide two ways to estimate properties of its posterior distribution. The difference between the two approaches is whether we predict the mean parameter of the abundance distribution defined as the aggregated point-process intensity (where we do not simulate the occurrences of the point process; this is the first approach), or whether we predict directly the abundance distribution by first simulating the point-process intensity and then the species occurrences generated according to this intensity function. Note: we have experienced problems when running the code of the second approach on certain computers (depending on the operating system).

```
#Estimating abundance (variance only due to unknown parameters):
Lambda <- predict(
  fit, ipoints(domaineSP, mesh),
  ~ sum(weight * exp(Intercept+elev+W1))
)
Lambda
```

```
     mean       sd   q0.025     q0.5   q0.975   median mean.mc_std_err
1 479.0001 25.41874 439.1196 478.4485 534.3563 478.4485        2.541874
  sd.mc_std_err
1      1.652013
```

```
#Estimating abundance (variance includes variance in point locations):
Nplant <- predict(
  fit, ipoints(domaineSP, mesh),
```

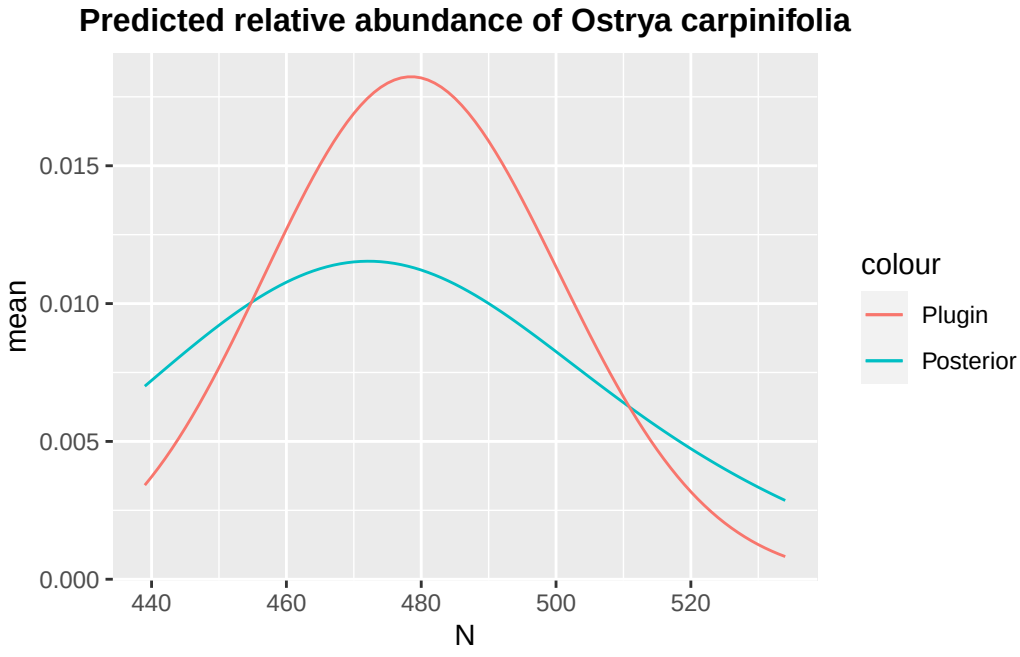
```
    ~ data.frame(
      N = round(Lambda$`q0.025`):round(Lambda$`q0.975`),
      dpois(round(Lambda$`q0.025`):round(Lambda$`q0.975`),
            lambda = sum(weight * exp(Intercept+W1+elev))
      )
    )
  )

  inla.qmarginal(c(0.025, 0.5, 0.975),
                 marginal = list(x = Nplant$N, y = Nplant$mean))
```

```
[1] 441.6874 478.3288 527.8278
```

```
  inla.emarginal(identity, marginal = list(x = Nplant$N, y = Nplant$mean))
```

```
[1] 480.1913
```

```
  Nplant$plugin_estimate <- dpois(Nplant$N, lambda = Lambda$mean)

  ggplot(data = Nplant) +
    geom_line(aes(x = N, y = mean, colour = "Posterior")) +
    geom_line(aes(x = N, y = plugin_estimate, colour = "Plugin")) +
    labs(title = "Predicted relative abundance of Ostrya carpinifolia") +
    theme(plot.title = element_text(size = 12, face = "bold", hjust = 0.5))
```



3.5 Exercice: DIY inlabru ! Fit and compare estimates for the *Senecio inaequidens* species

1) Create a SpatialPointsDataFrame with the observations of the species *Senecio inaequidens*
2) Define a mesh with higher resolution (about 1500 points).
3) Create a Matérn model with a less informative prior. For instance you could take a pcprior which satisfies $P(\sigma > 1) = 0.5$ and $P(\rho < 50) = 0.5$ where ρ is the range parameter.
4) Fit a LGCP model.
5) Check the estimates.
6) Plot the linear predictor.

3.6 How we did it

In the following, we provide an example solution for the above problem.

```
#1) already done
# Filter the PO data by the name of the studied species
PO_Senecio_inaequidens = PO %>% filter(species == "Senecio inaequidens")
# Convert the df to a spdf
PO_Senecio_inaequidens <-
  SpatialPointsDataFrame(coords = PO_Senecio_inaequidens[,c("x", "y")],
                         data = PO_Senecio_inaequidens,
                         proj4string = CRS(domaineSP@proj4string@projargs))

#2) use the cutoff or max.edge parameters
# Use of inla.mesh.2d
mesh = inla.mesh.2d(loc=rbind(coordinates(PO_Senecio_inaequidens)),
                    boundary = list(int = bndint, out = bndext),
                    max.edge = c(15, 100), cutoff = c(5),
                    crs = "+proj=lcc +lat_0=46.5 +lon_0=3 +lat_1=49 +lat_2=44
                    +x_0=700000 +y_0=6600000 +ellps=GRS80 +units=km +no_defs")
mesh$n
# Plot the mesh
par(mar=rep(1,4),mfrow=c(1,1))
plot(mesh, main = "", asp = 1)
plot(PO_Senecio_inaequidens,add=T,col=4,pch=16,cex=0.5)
plot(domaineSP,add=T,border=1,lwd=2)


#3) Less informative prior
matern <- inla.spde2.pcmatern(mesh,
                              alpha = 2, # fractional operator which is related
                              #to the smoothness of the Gaussian field
                              prior.sigma = c(1, 0.5), # P(sigma > 1) = 0.5
                              prior.range = c(50, 0.5)) # P(range < 50) = 0.5

#4) LGCP
cmp <- coordinates ~ Intercept(1) +
```

```
  elev(main = f.elev(x,y), model = "linear") +
  W1(coordinates, model = matern)
fit <- lgcp(cmp, PO_Senecio_inaequidens,
            samplers = domaineSP,
            domain = list(coordinates = mesh),
            options = list(control.inla = list(int.strategy = "eb")))

#5)
summary(fit)

#6)
pxl <- pixels(mesh, mask=domaineSP)
pr.int.tot <-  predict(fit, pxl, ~ Intercept+elev+W1)
p3=ggplot() +
  gg(pr.int.tot[,"mean"]) +
  gg(domaineSP, color = "grey") +
  gg(PO_Senecio_inaequidens, size = 0.5, alpha = 1, col=5) +
  scale_fill_continuous(type = "viridis") +
  coord_equal() +
  labs(title = "Predicted intensity of Senecio inaequidens") +
  theme(plot.title = element_text(size = 12, face = "bold", hjust = 0.5))
p3
```

4 Bivariate model with one shared spatial field

4.1 How to fit the model

Our second model is a bivariate LGCP. In this case, the intensity Λ is a vector (Λ_1, Λ_2), where Λ_1 represents the intensity of the species *Ostrya carpinifolia* and Λ_2 represents the intensity of the species *Senecio inaequidens*. For this example, we assume that the two species can have different responses to the elevation but they share a common spatial effect W_0.

$$\ln(\Lambda_1(s)) = \underbrace{\alpha_1}_{\text{Intercept}} + \underbrace{x(s)}_{\text{Elevation}}\ \beta_1 + \underbrace{W_0(s)}_{\text{shared Gaussian field}}$$

$$\ln(\Lambda_2(s)) = \underbrace{\alpha_2}_{\text{Intercept}} + \underbrace{x(s)}_{\text{Elevation}}\ \beta_2 + \underbrace{W_0(s)}_{\text{shared Gaussian field}}$$

In a bivariate model, the `lgcp` function is not directly applicable since it only works for univariate models. To model the bivariate intensity, we need to define two likelihoods, one for each species. We do this using the `like` function, which allows us to specify the distribution of the observations given the latent variables.

Once we have defined the likelihoods, we use the `bru` function to fit the bivariate LGCP model. The `bru` function is a wrapper of the `inla` function from the `R-INLA` package and allows us to fit more complex models that are not available directly in `inlabru`. As before, we define a `cmp`object where we generate a formula that lists all the predictor components of the model. After that, we can define the two likelihoods where each one can incorporate some of the components defined in `cmp`.

```r
matern <- inla.spde2.pcmatern(mesh,
                              prior.sigma = c(1, 0.001), # P(sigma > 1) = 0.001
                              prior.range = c(1, 0.9))   # P(range < 100) = 0.9


cmp <- ~ -1+interceptOc(1)+interceptSi(1)+
  elevationOc(main = f.elev(x,y), model = "linear")+
  elevationSi(main = f.elev(x,y), model = "linear")+
  W0(main=coordinates, model=matern)



lik1 <- like("cp",
             formula = coordinates ~ -1+interceptOc+elevationOc+W0,
             data = PO_Ostrya_carpinifolia,
             samplers = domaineSP,
             domain = list(coordinates = mesh)
)

lik2 <- like("cp",
             formula = coordinates ~ -1+interceptSi+elevationSi+W0,
             data = PO_Senecio_inaequidens,
             samplers = domaineSP,
             domain = list(coordinates = mesh)
)


bru_options_set(control.compute = list(cpo=T, dic=T, mlik=T, waic=T,
                                       return.marginals.predictor = F,
                                       config=TRUE),
                control.inla = list(int.strategy = "eb", strategy = "gaussian"),
                verbose = T,
                bru_max_iter = 1)

T1 <- Sys.time()
fit <- bru(cmp,lik1,lik2)
T2 <- Sys.time()

fit2 = fit # keep the output
summary(fit)
```

```
inlabru version: 2.7.0
INLA version: 22.12.16
Components:
interceptOc: main = linear(1)
interceptSi: main = linear(1)
elevationOc: main = linear(f.elev(x, y))
elevationSi: main = linear(f.elev(x, y))
W0: main = spde(coordinates)
Likelihoods:
```

```
  Family: 'cp'
    Data class: 'SpatialPointsDataFrame'
    Predictor: coordinates ~ -1 + interceptOc + elevationOc + W0
  Family: 'cp'
    Data class: 'SpatialPointsDataFrame'
    Predictor: coordinates ~ -1 + interceptSi + elevationSi + W0
Time used:
    Pre = 0.387, Running = 2.39, Post = 0.0229, Total = 2.8
Fixed effects:
              mean    sd 0.025quant 0.5quant 0.975quant   mode kld
interceptOc -5.512 0.255     -6.012   -5.512     -5.011 -5.512   0
interceptSi -7.888 0.298     -8.472   -7.888     -7.303 -7.888   0
elevationOc -1.025 0.191     -1.400   -1.025     -0.650 -1.025   0
elevationSi -0.971 0.316     -1.591   -0.971     -0.352 -0.971   0

Random effects:
  Name    Model
    W0 SPDE2 model

Model hyperparameters:
              mean    sd 0.025quant 0.5quant 0.975quant  mode
Range for W0 52.46 7.412      39.87    51.76      69.00 50.13
Stdev for W0  1.32 0.108       1.12     1.31       1.54  1.30

Deviance Information Criterion (DIC) ...............: -6698.70
Deviance Information Criterion (DIC, saturated) ....: NA
Effective number of parameters .....................: -7726.39

Watanabe-Akaike information criterion (WAIC) ...: 1597.01
Effective number of parameters .................: 306.64

Marginal log-Likelihood:  -4593.94
CPO, PIT is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

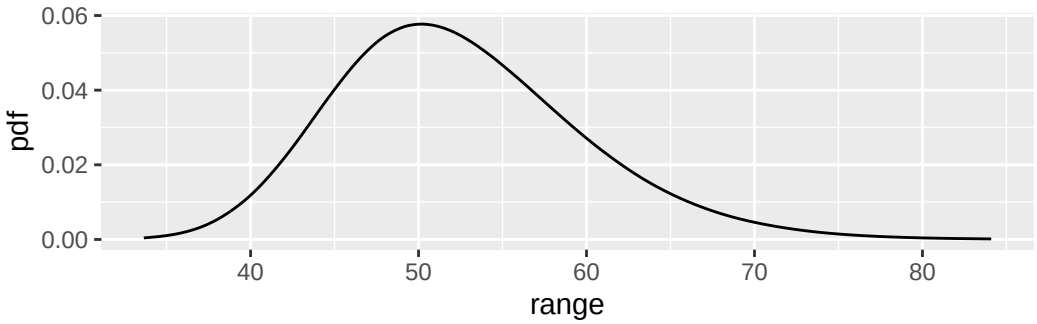
```
  Tdiff= difftime(T2, T1) ; Tdiff
```

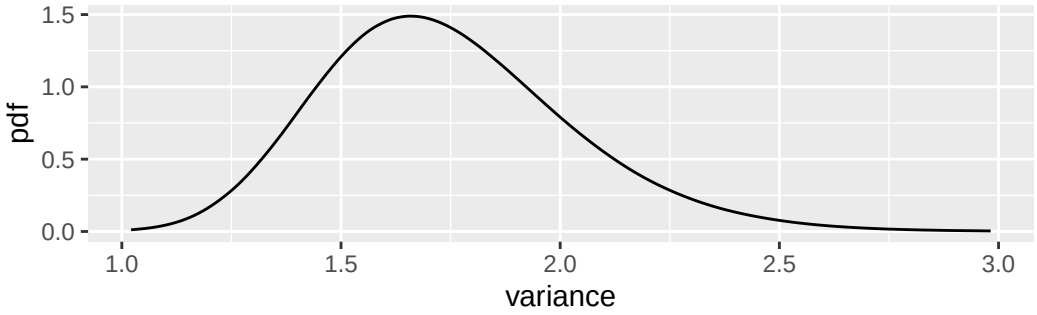
```
Time difference of 5.313381 secs
```

4.2 Some plots

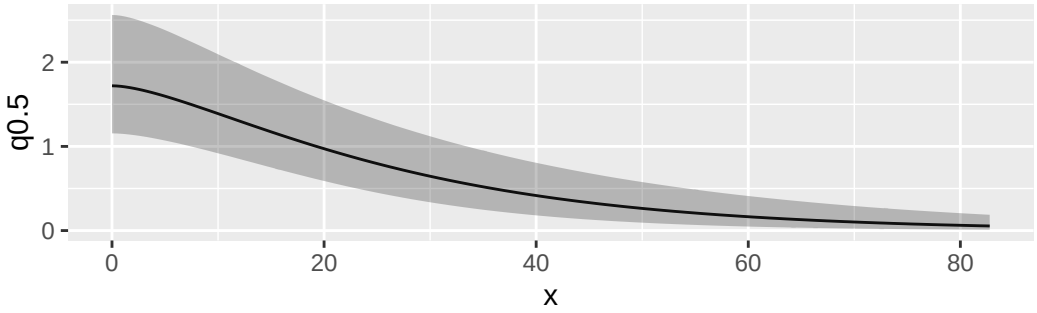
Shared field W_0:

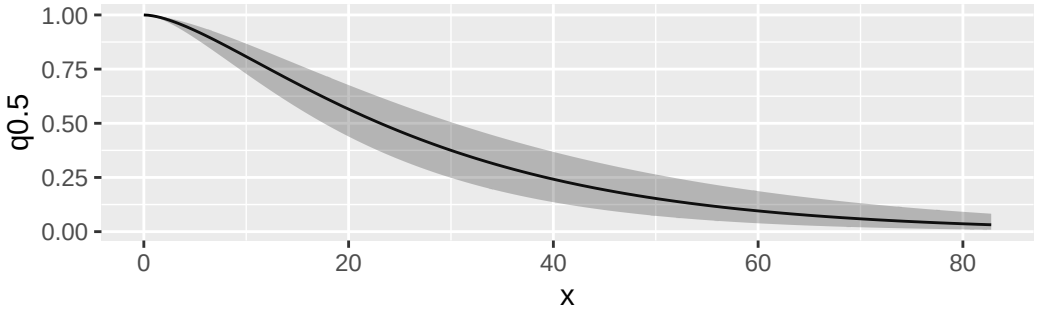
```
  spde.range.W0 <- spde.posterior(fit, "W0", what = "range")
  spde.logvar.W0 <- spde.posterior(fit, "W0", what = "variance")
  range.plot.W0 <- plot(spde.range.W0)
  var.plot.W0 <- plot(spde.logvar.W0)
  multiplot(range.plot.W0, var.plot.W0)
```





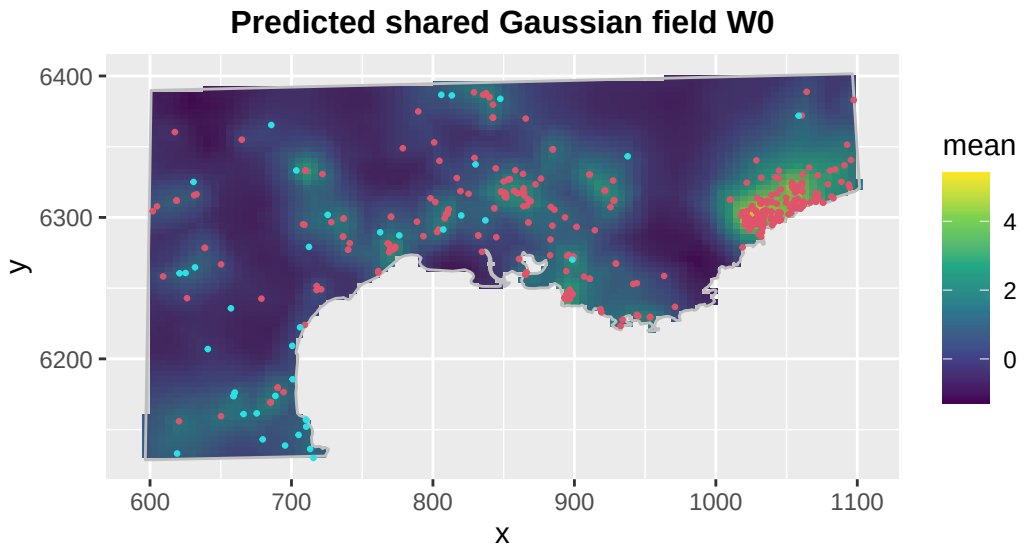

```
corplot.W0 <- plot(spde.posterior(fit, "W0", what = "matern.correlation"))
covplot.W0 <- plot(spde.posterior(fit, "W0", what = "matern.covariance"))
multiplot(covplot.W0, corplot.W0)
```



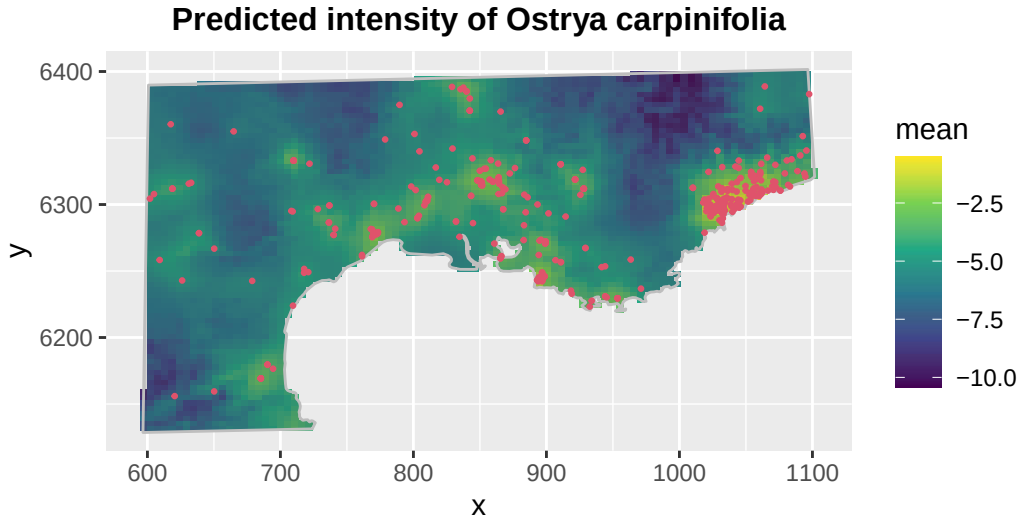


Plot of the predictors

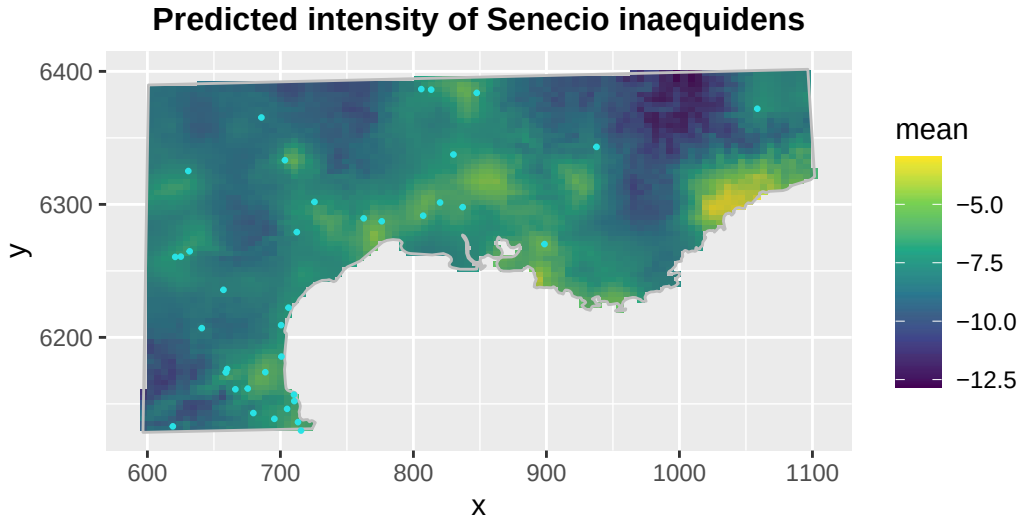
```
pxl <- pixels(mesh, mask=domaineSP)
pr.int.W0 <-  predict(fit, pxl, ~ W0)
pr.int.Oc <-  predict(fit, pxl, ~ interceptOc+elevationOc+W0)
pr.int.Si <-  predict(fit, pxl, ~ interceptSi+elevationSi+W0)



plot.W0 = ggplot() +
  gg(pr.int.W0[,"mean"]) +
  gg(domaineSP, color = "grey") +
  gg(PO_Ostrya_carpinifolia, size = 0.5, alpha = 1, col=2) +
  gg(PO_Senecio_inaequidens, size = 0.5, alpha = 1, col = 5) +
  scale_fill_continuous(type = "viridis") +
  coord_equal() +
  labs(title = "Predicted shared Gaussian field W0") +
  theme(plot.title = element_text(size = 12, face = "bold", hjust = 0.5))

plot.Oc = ggplot() +
  gg(pr.int.Oc[,"mean"]) +
  gg(domaineSP, color = "grey") +
  gg(PO_Ostrya_carpinifolia, size = 0.5, alpha = 1, col=2) +
  scale_fill_continuous(type = "viridis") +
  coord_equal() +
  labs(title = "Predicted intensity of Ostrya carpinifolia") +
  theme(plot.title = element_text(size = 12, face = "bold", hjust = 0.5))

plot.Si = ggplot() +
  gg(pr.int.Si[,"mean"]) +
  gg(domaineSP, color = "grey") +
  gg(PO_Senecio_inaequidens, size = 0.5, alpha = 1, col=5) +
  scale_fill_continuous(type = "viridis") +
  coord_equal() +
  labs(title = "Predicted intensity of Senecio inaequidens") +
  theme(plot.title = element_text(size = 12, face = "bold", hjust = 0.5))

plot.W0
```




```
plot.Oc
```




```
plot.Si
```



We finally estimate the abundances of the species *Ostrya carpinifolia* and *Senecio inaequidens* using the same two approaches as before.

```
# Abundance of Ostrya carpinifolia
#Estimating abundance (variance only due to unknown parameters):
Lambda.Oc <- predict(
  fit, ipoints(domaineSP, mesh),
  ~ sum(weight * exp(interceptOc+elevationOc+W0))
)
Lambda.Oc
```

```
     mean       sd  q0.025     q0.5  q0.975   median mean.mc_std_err
1 1292.199 70.27474 1135.12 1288.664 1425.99 1288.664        7.027474
  sd.mc_std_err
1      4.816423
```

```
#Estimating abundance (variance includes variance in point locations):
Nplant.Oc <- predict(
  fit, ipoints(domaineSP, mesh),
  ~ data.frame(
    N = round(Lambda.Oc$`q0.025`):round(Lambda.Oc$`q0.975`),
    dpois(round(Lambda.Oc$`q0.025`):round(Lambda.Oc$`q0.975`),
          lambda = sum(weight * exp(interceptOc+elevationOc+W0))
    )
  )
)
```

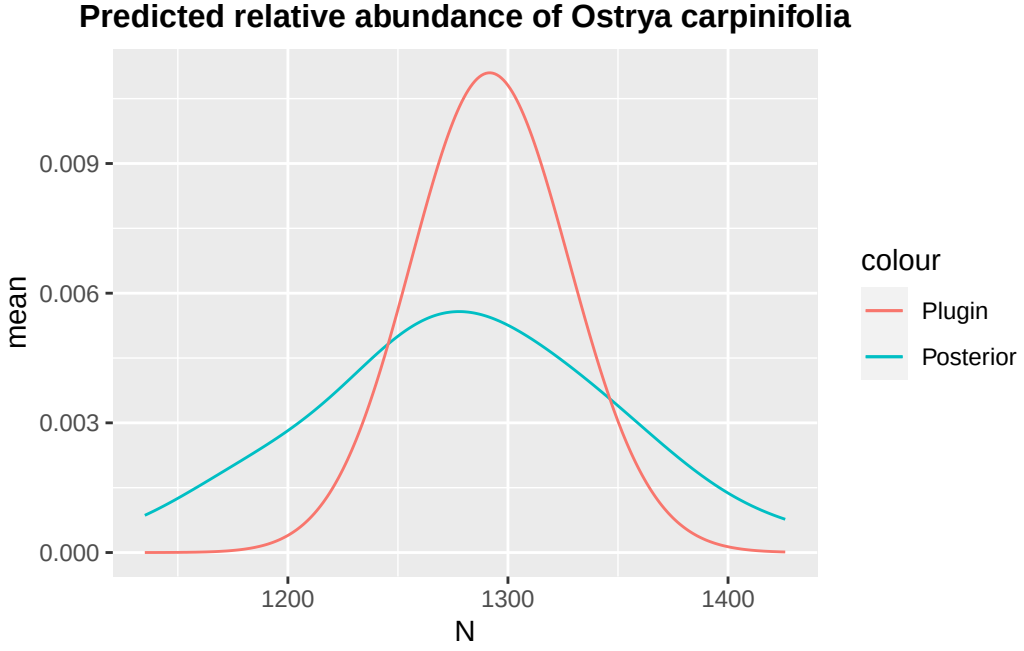
```r
inla.qmarginal(c(0.025, 0.5, 0.975),
               marginal = list(x = Nplant.Oc$N, y = Nplant.Oc$mean))
```

```
[1] 1155.749 1279.904 1402.585
```

```r
inla.emarginal(identity, marginal = list(x = Nplant.Oc$N, y = Nplant.Oc$mean))
```

```
[1] 1279.741
```

```r
Nplant.Oc$plugin_estimate <- dpois(Nplant.Oc$N, lambda = Lambda.Oc$mean)

ggplot(data = Nplant.Oc) +
  geom_line(aes(x = N, y = mean, colour = "Posterior")) +
  geom_line(aes(x = N, y = plugin_estimate, colour = "Plugin")) +
  labs(title = "Predicted relative abundance of Ostrya carpinifolia") +
  theme(plot.title = element_text(size = 12, face = "bold", hjust = 0.5))
```




```r
# Abundance of Senecio inaequidens
#Estimating abundance (variance only due to unknown parameters):
Lambda.Si <- predict(
  fit, ipoints(domaineSP, mesh),
  ~ sum(weight * exp(interceptSi+elevationSi+W0))
)
Lambda.Si
```

```
    mean       sd    q0.025     q0.5    q0.975    median mean.mc_std_err
```

```
1 121.0274 19.07204 84.33834 120.9683 157.1947 120.9683        1.907204
  sd.mc_std_err
1      1.526996
```

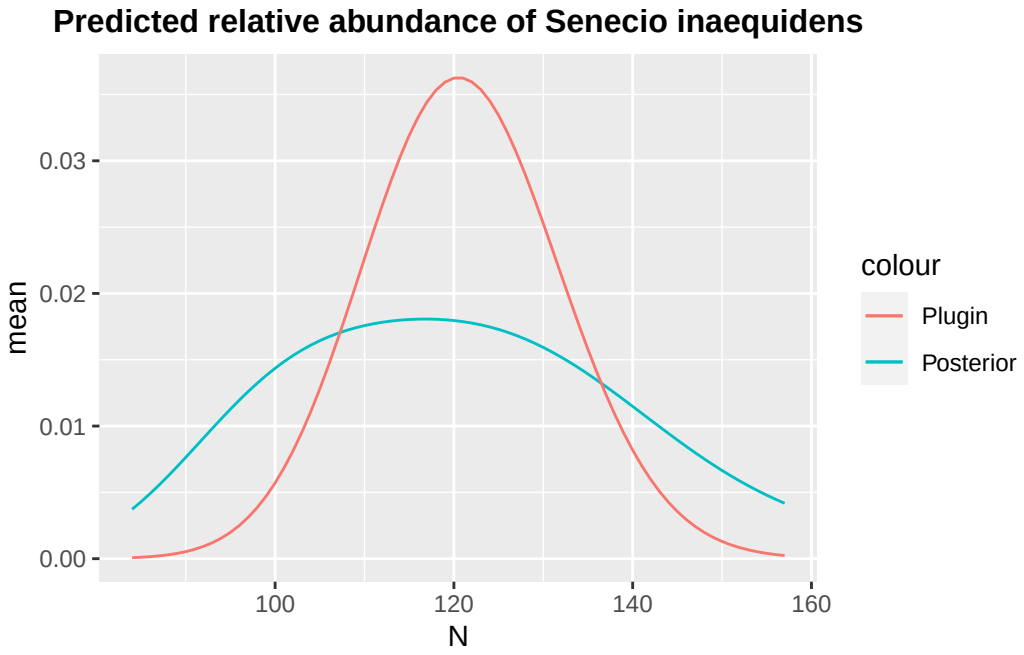
```
#Estimating abundance (variance includes variance in point locations):
Nplant.Si <- predict(
  fit, ipoints(domaineSP, mesh),
  ~ data.frame(
    N = round(Lambda.Si$`q0.025`):round(Lambda.Si$`q0.975`),
    dpois(round(Lambda.Si$`q0.025`):round(Lambda.Si$`q0.975`),
          lambda = sum(weight * exp(interceptSi+elevationSi+W0))
    )
  )
)

inla.qmarginal(c(0.025, 0.5, 0.975),
               marginal = list(x = Nplant.Si$N, y = Nplant.Si$mean))
```

```
[1]  88.50958 118.59049 152.29581
```

```
inla.emarginal(identity, marginal = list(x = Nplant.Si$N, y = Nplant.Si$mean))
```

```
[1] 119.1106
```

```
Nplant.Si$plugin_estimate <- dpois(Nplant.Si$N, lambda = Lambda.Si$mean)

ggplot(data = Nplant.Si) +
  geom_line(aes(x = N, y = mean, colour = "Posterior")) +
  geom_line(aes(x = N, y = plugin_estimate, colour = "Plugin")) +
  labs(title = "Predicted relative abundance of Senecio inaequidens") +
  theme(plot.title = element_text(size = 12, face = "bold", hjust = 0.5))
```



5 A bivariate model with specific and shared spatial fields

5.1 Exercices: DIY, part II

The aim of this section is to develop a new bivariate model for the species *Ostrya carpinifolia* and *Senecio inaequidens*. In addition to a shared spatial field, this model accounts for specific spatial fields for each species to capture different patterns of point distribution between them. The model can thus be written as follows:

$$\ln(\Lambda_1(s)) = \underbrace{\alpha_1}_{\text{Intercept}} + \underbrace{x(s)}_{\text{Elevation}}\ \beta_1 + \underbrace{W_1(s)}_{\text{specific Gaussian field}} + \underbrace{W_0(s)}_{\text{shared Gaussian field}}$$

$$\ln(\Lambda_2(s)) = \underbrace{\alpha_2}_{\text{Intercept}} + \underbrace{x(s)}_{\text{Elevation}}\ \beta_2 + \underbrace{W_2(s)}_{\text{specific Gaussian field}} + \underbrace{W_0(s)}_{\text{shared Gaussian field}}$$

5.2 How we did it

In the following, we provide an example solution for the above problem.

```
matern <- inla.spde2.pcmatern(mesh,
                              prior.sigma = c(1, 0.001), # P(sigma > 1) = 0.001
                              prior.range = c(1, 0.9))   # P(range < 100) = 0.9

cmp <- ~ -1+interOc(1)+interSi(1)+
  elevOc(main = f.elev(x,y), model = "linear")+
  elevSi(main = f.elev(x,y), model = "linear")+
  W0(main=coordinates, model=matern)+
```

```
    W1(main=coordinates, model=matern)+
    W2(main=coordinates, model=matern)


  lik1 <- like("cp",
               formula = coordinates ~ -1+interOc+elevOc+W1+W0,#
               data = PO_Ostrya_carpinifolia,
               samplers = domaineSP,
               domain = list(coordinates = mesh)
  )

  lik2 <- like("cp",
               formula = coordinates ~ -1+interSi+elevSi+W2+W0,#
               data = PO_Senecio_inaequidens,
               samplers = domaineSP,
               domain = list(coordinates = mesh)
  )


  bru_options_set(bru_verbose = TRUE,
                  control.compute = list(cpo=T, dic=T, mlik=T, waic=T,
                                         return.marginals.predictor = T,
                                         config=TRUE),
                  control.inla = list(int.strategy = "eb", strategy = "gaussian"))
  #,bru_max_iter = 1)

  debut=Sys.time()
  fit <- bru(cmp,lik1,lik2)
```

```
iinla: Iteration 1 [max:1]
```

```
  fin=Sys.time()
  fin-debut
```

```
Time difference of 1.593497 mins
```

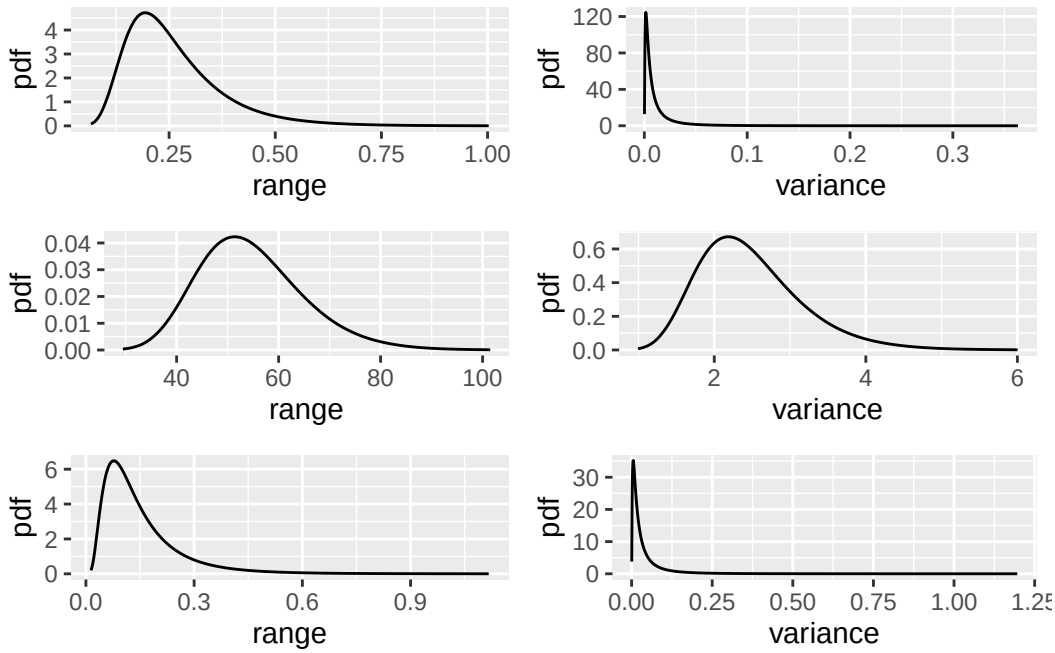
```
  summary(fit)
```

```
inlabru version: 2.7.0
INLA version: 22.12.16
Components:
interOc: main = linear(1)
interSi: main = linear(1)
elevOc: main = linear(f.elev(x, y))
elevSi: main = linear(f.elev(x, y))
W0: main = spde(coordinates)
W1: main = spde(coordinates)
W2: main = spde(coordinates)
Likelihoods:
  Family: 'cp'
```

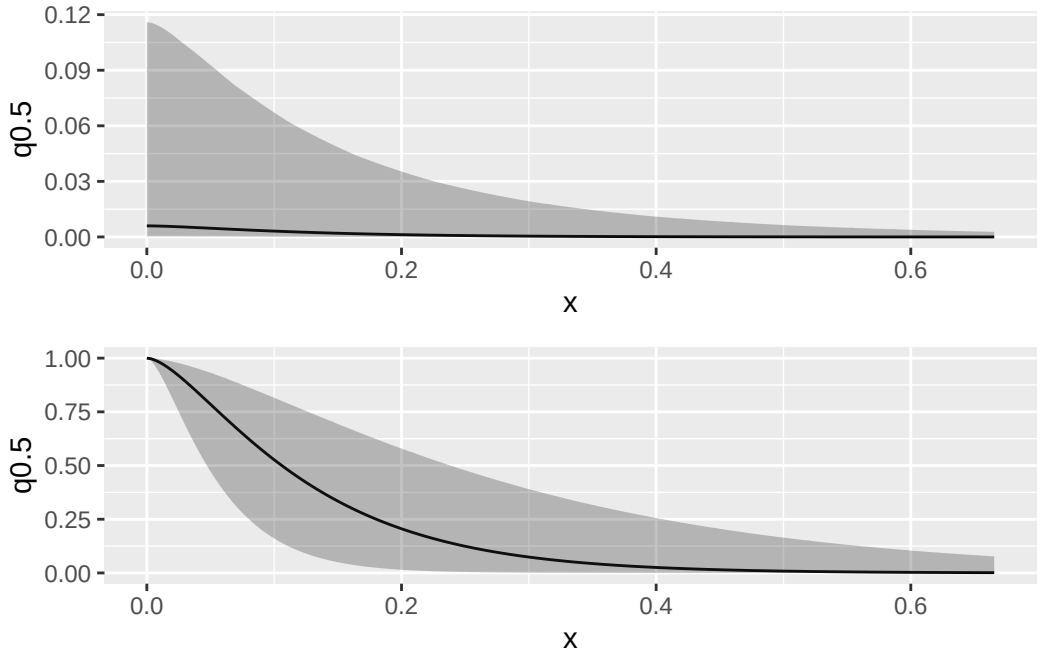
```
    Data class: 'SpatialPointsDataFrame'
    Predictor: coordinates ~ -1 + interOc + elevOc + W1 + W0
  Family: 'cp'
    Data class: 'SpatialPointsDataFrame'
    Predictor: coordinates ~ -1 + interSi + elevSi + W2 + W0
Time used:
    Pre = 0.771, Running = 87.3, Post = 0.413, Total = 88.5
Fixed effects:
          mean    sd 0.025quant 0.5quant 0.975quant   mode kld
interOc -6.034 0.311     -6.643   -6.034     -5.424 -6.034   0
interSi -6.585 0.163     -6.904   -6.585     -6.267 -6.585   0
elevOc  -1.056 0.219     -1.485   -1.056     -0.627 -1.056   0
elevSi  -0.850 0.264     -1.368   -0.850     -0.333 -0.850   0

Random effects:
  Name    Model
    W0 SPDE2 model
   W1 SPDE2 model
   W2 SPDE2 model

Model hyperparameters:
               mean     sd 0.025quant 0.5quant 0.975quant   mode
Range for W0  0.260  0.116      0.109    0.235      0.555  0.194
Stdev for W0  0.093  0.064      0.024    0.077      0.264  0.053
Range for W1 54.723 10.443     37.277   53.636     78.625 51.333
Stdev for W1  1.559  0.215      1.184    1.542      2.036  1.504
Range for W2  0.151  0.114      0.033    0.120      0.456  0.077
Stdev for W2  0.173  0.117      0.044    0.143      0.487  0.099

Deviance Information Criterion (DIC) ...............: -6587.26
Deviance Information Criterion (DIC, saturated) ....: NA
Effective number of parameters .....................: -7630.22

Watanabe-Akaike information criterion (WAIC) ...: 1927.50
Effective number of parameters .................: 478.35

Marginal log-Likelihood:  -4556.92
CPO, PIT is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

```r
spde.range <- spde.posterior(fit, "W0", what = "range")
spde.logvar <- spde.posterior(fit, "W0", what = "variance")
range.plot0 <- plot(spde.range)
var.plot0 <- plot(spde.logvar)
spde.range <- spde.posterior(fit, "W1", what = "range")
spde.logvar <- spde.posterior(fit, "W1", what = "variance")
range.plot1 <- plot(spde.range)
var.plot1 <- plot(spde.logvar)
spde.range <- spde.posterior(fit, "W2", what = "range")
```

```
spde.logvar <- spde.posterior(fit, "W2", what = "variance")
range.plot2 <- plot(spde.range)
var.plot2 <- plot(spde.logvar)
multiplot(range.plot0, range.plot1,range.plot2,
          var.plot0,var.plot1,var.plot2,cols=2)#, int.plot)
```




```
corplot <- plot(spde.posterior(fit, "W0", what = "matern.correlation"))
covplot <- plot(spde.posterior(fit, "W0", what = "matern.covariance"))
multiplot(covplot, corplot)
```




```
pxl <- pixels(mesh, mask=domaineSP)
pr.intW0 <-  predict(fit, pxl, ~ W0)
pr.intW1 <-  predict(fit, pxl, ~ W1)
pr.intW2 <-  predict(fit, pxl, ~ W2)
pr.int.Oc <- predict(fit, pxl, ~ interOc+elevOc+W1+W0)
pr.int.Si <- predict(fit, pxl, ~ interSi+elevSi+W2+W0)


plotW0 = ggplot() +
  gg(pr.intW0[,"mean"]) +
  gg(domaineSP, color = "grey") +
  gg(PO_Ostrya_carpinifolia, size = 0.5, alpha = 1, col=2) +
  gg(PO_Senecio_inaequidens, size = 0.5, alpha = 1, col=5) +
  scale_fill_continuous(type = "viridis") +
  coord_equal() +
  labs(title = "Predicted shared Gaussian field W0") +
  theme(plot.title = element_text(size = 12, face = "bold", hjust = 0.5))
```

Regions defined for each Polygons

```
plotW1 = ggplot() +
  gg(pr.intW1[,"mean"]) +
  gg(domaineSP, color = "grey") +
  gg(PO_Ostrya_carpinifolia, size = 0.5, alpha = 1, col=2) +
  scale_fill_continuous(type = "viridis") +
  coord_equal() +
  labs(title = "Predicted specific Gaussian field W1 of Ostrya carpinifolia") +
```

```
    theme(plot.title = element_text(size = 12, face = "bold", hjust = 0.5))
```

Regions defined for each Polygons

```
  plotW2 = ggplot() +
    gg(pr.intW2[,"mean"]) +
    gg(domaineSP, color = "grey") +
    gg(PO_Senecio_inaequidens, size = 0.5, alpha = 1, col=5) +
    scale_fill_continuous(type = "viridis") +
    coord_equal() +
    labs(title = "Predicted specific Gaussian field W2 of Senecio inaequidens") +
    theme(plot.title = element_text(size = 12, face = "bold", hjust = 0.5))
```

Regions defined for each Polygons

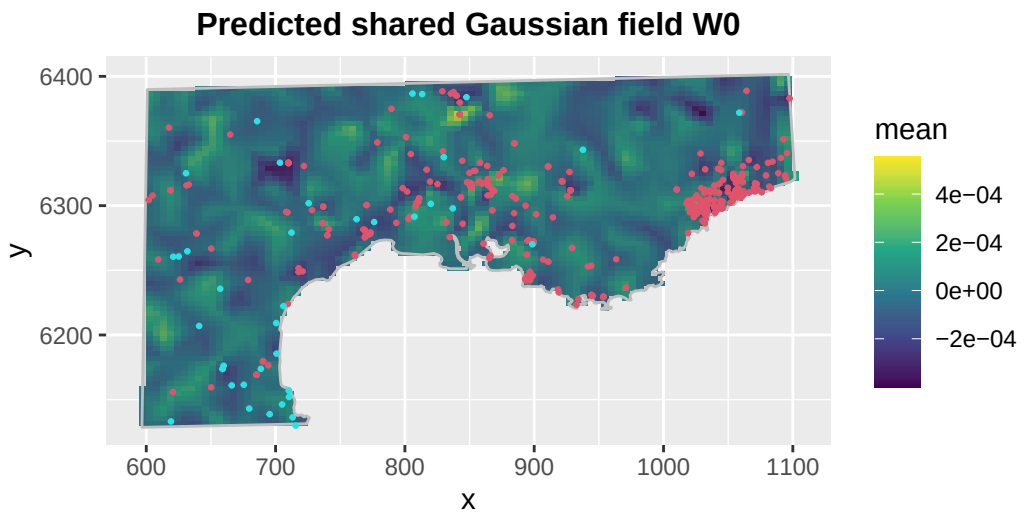
```
  plot.Oc = ggplot() +
    gg(pr.int.Oc[,"mean"]) +
    gg(domaineSP, color = "grey") +
    gg(PO_Ostrya_carpinifolia, size = 0.5, alpha = 1, col=2) +
    scale_fill_continuous(type = "viridis") +
    coord_equal() +
    labs(title = "Predicted intensity of Ostrya carpinifolia") +
    theme(plot.title = element_text(size = 12, face = "bold", hjust = 0.5))
```

Regions defined for each Polygons

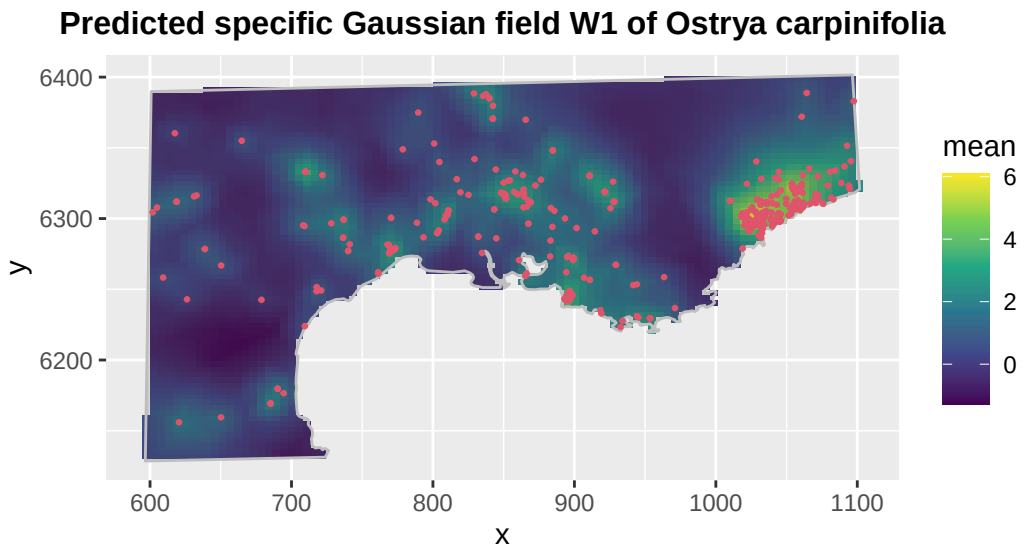
```
  plot.Si = ggplot() +
    gg(pr.int.Si[,"mean"]) +
    gg(domaineSP, color = "grey") +
    gg(PO_Senecio_inaequidens, size = 0.5, alpha = 1, col=5) +
    scale_fill_continuous(type = "viridis") +
    coord_equal() +
    labs(title = "Predicted intensity of Senecio inaequidens") +
    theme(plot.title = element_text(size = 12, face = "bold", hjust = 0.5))
```

Regions defined for each Polygons

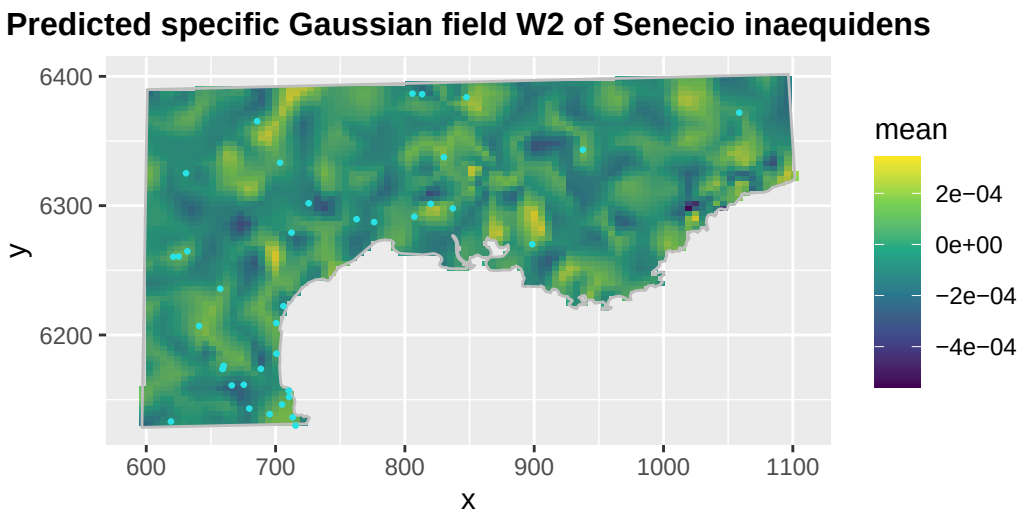
```
  plotW0
```



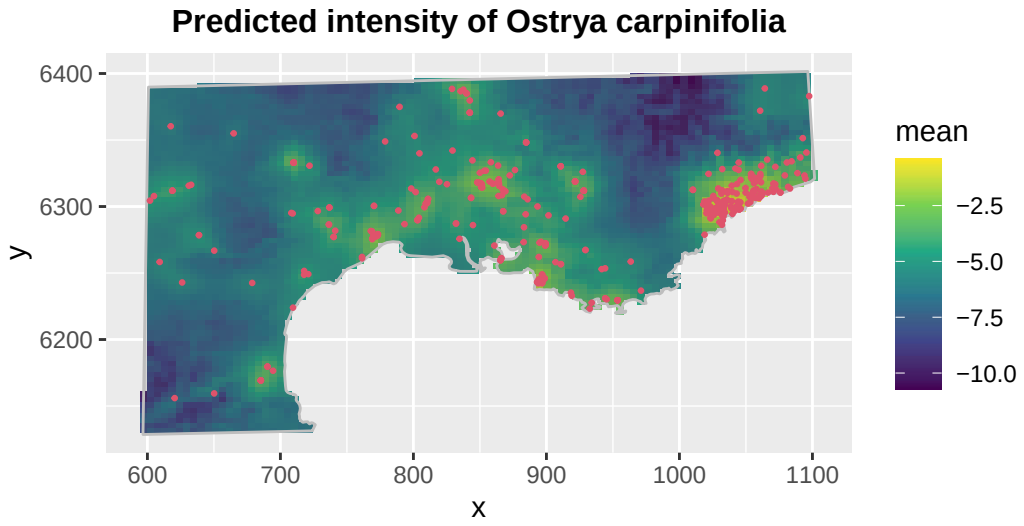

```
plotW1
```



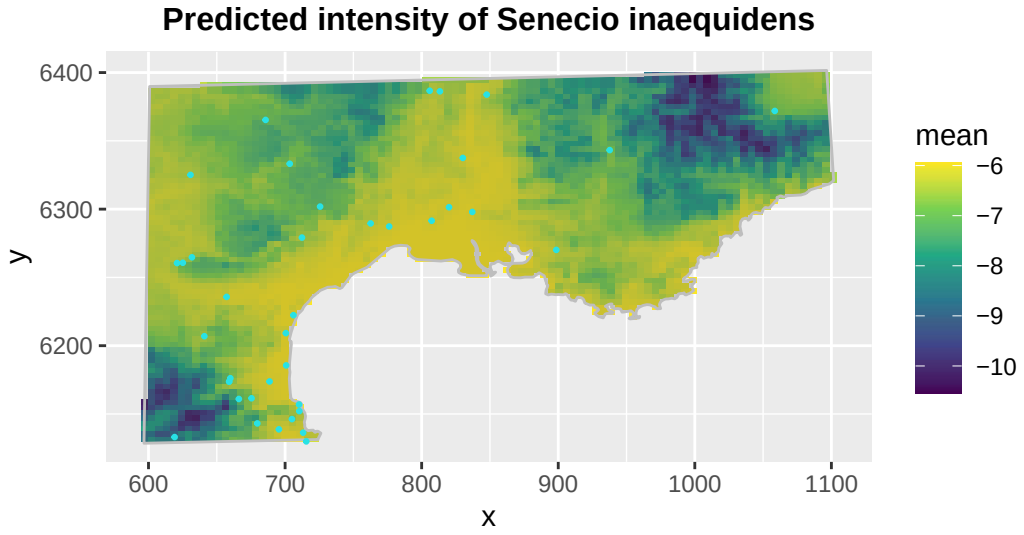

```
plotW2
```




```
plot.Oc
```




```
plot.Si
```


Predicted intensity of Senecio inaequidens
6400
6300
6200
y
600
700
800
900
1000
1100
x
mean
−6
−7
−8
−9
−10

Atelier 2: Models accounting for sampling bias

Christophe Botella, Florian Lasgorceux, Juliette Legrand, Thomas Opitz, Julien Papaïx

4/14/23

Table of contents

In this second tutorial, our general goal is to model the spatial distribution of plant species based on the Pl@ntNet presence-only data by constructing models that take the sampling bias into account and allow correcting it in the predictions and maps provided by the model. We will explore different solutions for this problem. Some of them use only the opportunistic Pl@ntNet data, others combine opportunistic data with presence-absence data collected with a more strict sampling protocol.

1 Getting ready

To begin, we load the R packages that we will use, and you have to set the path to the folder where R code files and data are located.

Remember: to view the help page for a command in R, just type "?command" with command replaced by the name of the function.

When running the code "live" during this tutorial, we may want to choose tuning parameters for algorithms and maps that allow faster execution of commands. Therefore, when running the code "live", you may want to set "fast_run" to TRUE in the first line below. This will result in some coarser approximations, but the interpretation of results remains qualitatively very similar to more computer-intensive case where `fast_run=TRUE`.

```
fast_run = TRUE
if(fast_run){
  n.samples <- 25 # number of Monte-Carlo draws when doing posterior sampling
  bru_max_iter <- 1 # number of iterations of estimation in inlabru
  nx <- 100 # number of pixels in maps (x-direction)
  ny <- 75 # number of pixels in maps (y-direction)
  n_ss_TG <- 2000 # subsample size for the Target-Group Background
}else{
  n.samples <- 100
  bru_max_iter <- 10 # (this is also the default in inlabru)
  nx <- 200
  ny <- 150
  n_ss_TG <- 10000
}
```

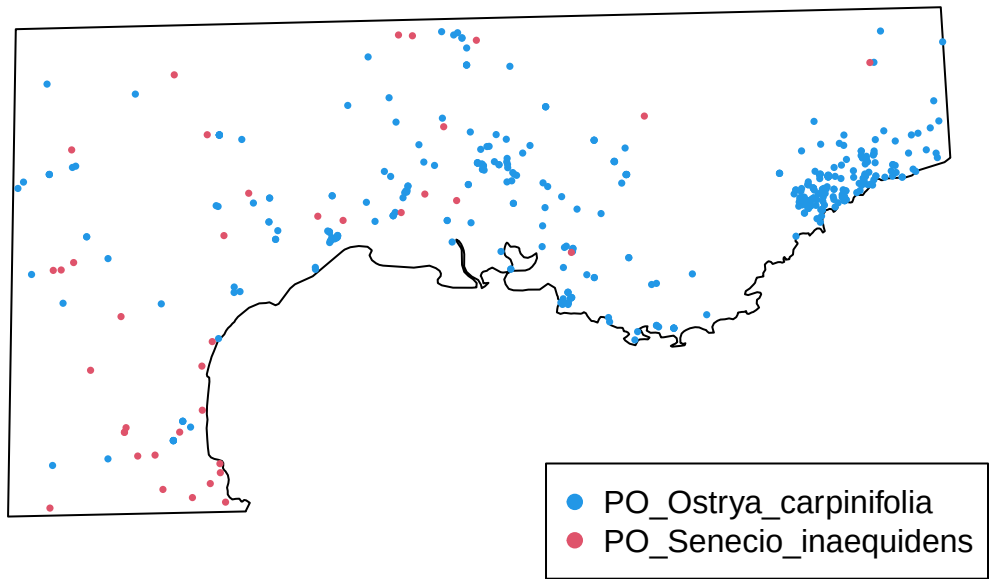
1.1 Occurrence data for the focal species

We first extract the corresponding occurrence data and show their distribution on a map. If not already done, we extract the Presence-Only occurrence data for the two species that we will study here: *Ostrya carpinifolia* and *Senecio inaequidens*. We prepare a SpatialPointsDataFrame for each species.

```
# Get the presences of Ostrya carpinifolia
PO_Ostrya_carpinifolia = PO %>%
  filter(species == "Ostrya carpinifolia")%>%
  SpatialPointsDataFrame(coords = select(.,x,y),data = .,
      proj4string = CRS(domaineSP@proj4string@projargs))
# Remove points outside domain
inside=sapply(1:dim(PO_Ostrya_carpinifolia)[1],
              function(i)gContains(domaineSP,PO_Ostrya_carpinifolia[i,,drop=F]))
PO_Ostrya_carpinifolia=PO_Ostrya_carpinifolia[inside,]
```

```
# Get the presences of Senecio inaequidens
PO_Senecio_inaequidens = PO %>%
  filter(species == "Senecio inaequidens")%>%
  SpatialPointsDataFrame(coords = select(.,x,y),data = .,
      proj4string = CRS(domaineSP@proj4string@projargs))
# Remove points outside domain
inside=sapply(1:dim(PO_Senecio_inaequidens)[1],
              function(i)gContains(domaineSP,PO_Senecio_inaequidens[i,,drop=F]))
PO_Senecio_inaequidens=PO_Senecio_inaequidens[inside,]
```

We map the presence-only occurrences of the two species that we are interested in.



2 Model 1 (univariate): Target-Group Background as covariate

In this section, we will fit a Log-Gaussian process to the presences of the species *Ostrya carpinifolia* by integrating the spatial density of Target-Group species occurrences (hereafter TG density) as a fixed offset that we pre-compute in the data preparation steps based on the species richness in the pixels of a certain spatial grid. The Target-Group of species was built automatically using the script **Annex.R**. The TG density is used as a proxy of the sampling effort to control for sampling bias when fitting an LGCP to the occurrences of the focal species.

```
print(head(TGdf))
```

```
                   spAdded  entropy cumulOcc  cellRate
1 Aphyllanthes monspeliensis 5.584869     3739 0.2754717
2   Cephalanthera longifolia 5.846513     4613 0.3679245
3        Saponaria ocymoides 5.996189     6012 0.4157233
```

```
4       Lathraea clandestina 6.099810     6439 0.4522013
5     Dactylorhiza sambucina 6.166557     6849 0.4786164
6     Urospermum dalechampii 6.200775     9175 0.5018868
```

```
cat('Number of TG species:',dim(TGdf)[1],'\n')
```

```
Number of TG species: 193
```

```
cat('Number of TG occurrences (among TG species):',sum(PO$species%in%TGdf$spAdded),'\n')
```

```
Number of TG occurrences (among TG species): 62962
```

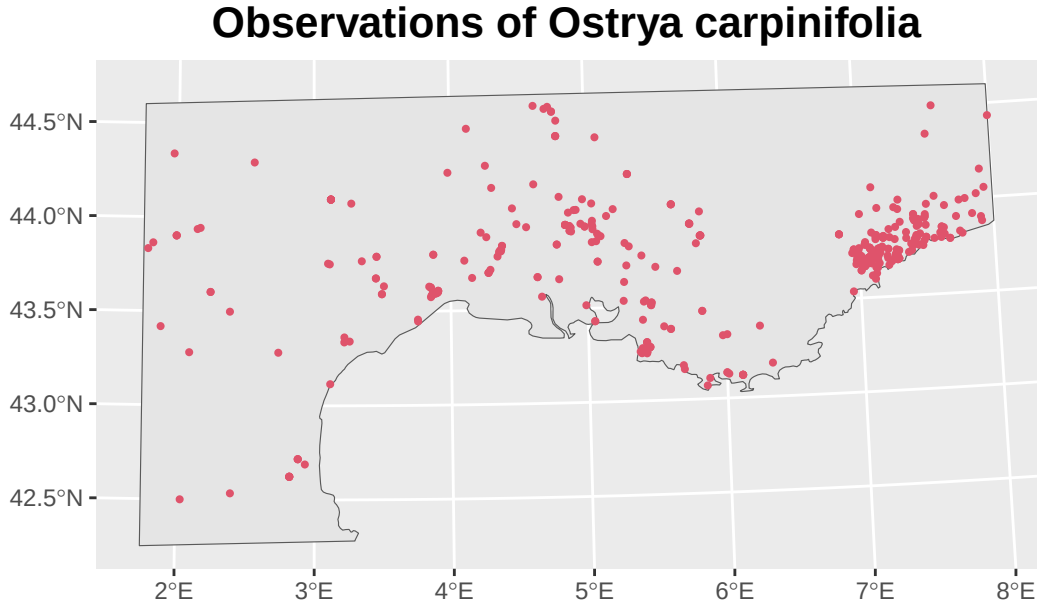
The model that we want to estimate has the following structure of the point-process intensity function $\Lambda(s)$:

$$\log\left(\Lambda(s)\right) = \underbrace{x_1(s)}_{\log(\text{Target-Group density})} + \underbrace{\alpha_1}_{\text{Intercept}} + \beta_2 \times \underbrace{x_2(s)}_{\text{Elevation}} + \underbrace{W(s)}_{\text{Gaussian field}}$$

Here, the covariate $x_1(s)$ provides the logarithm of the Target-Group density for each spatial location s, annd we include it as a fixed offset for which the log-transform is applied to be coherent with the log-link function applied to $\Lambda(s)$. The role of the spatial field $W(s)$ is to capture spatial variability that is not explained by the two covariates $x_1(s)$ and $x_2(s)$.

To start, we show the distribution of its occurrences on a map (without any other species).

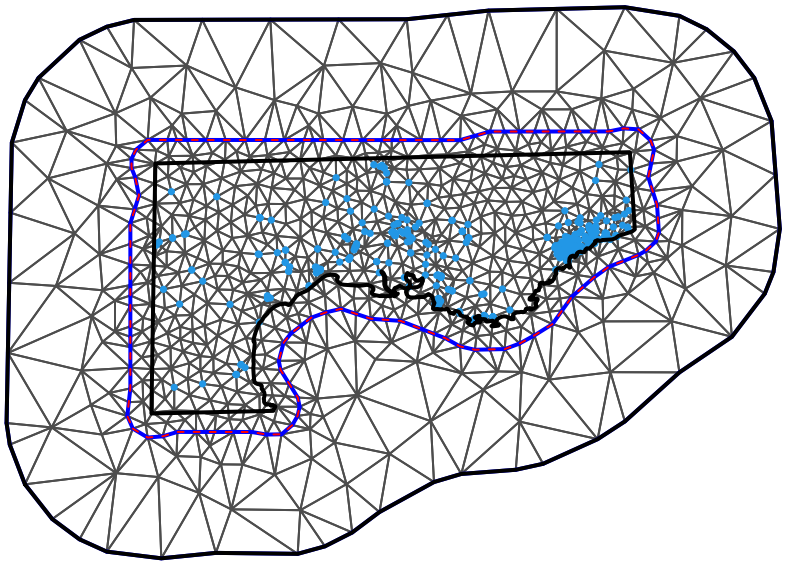
```
ggplot() + geom_sf(data=st_as_sf(domaineSP)) +
  geom_sf(data = st_as_sf(PO_Ostrya_carpinifolia), color = 2, size = .75) +
  labs(title = "Observations of Ostrya carpinifolia") +
  theme(plot.title = element_text(size = 16, face = "bold", hjust = 0.5))
```



2.1 Building the spatial model

We now build the triangulation mesh that we need for using the SPDE approach to include spatial fields in the log-intensity function of the Log-Gaussian Cox Process (LGCP). Here, we first construct an interior domain covering the study area with relatively high resolution of the triangulation, and then an exterior domain which we need to avoid having strong boundary effects of the SPDE in the study area (i.e., we push the boundary used for the SPDE far away from the study area). To construct an interior boundary that follows relatively closely the contour of our study area, we generate a regular grid (here containing around 1000 cells) and then construct a nonconvex hull for this grid, which we use as the interior boundary. To reduce the computational cost as far as possible, we can use a coarser resolution of the triangulation in the outer boundary. In the following code, the two values in `max.edge = c(25,100)` define the maximum edge length of the triangulation segments in the interior and exterior domains, respectively. Finally, we plot the study domain and the interior and exterior boundaries.

```
tmp = spsample(domaineSP@polygons[[1]], n = 1000, type = "regular")
coordDOM <- rbind(domaineSP@polygons[[1]]@Polygons[[1]]@coords, tmp@coords)
# Boundaries
bndint <- inla.nonconvex.hull(coordDOM, convex=-.05)
bndext <- inla.nonconvex.hull(coordDOM, convex=-.3)
# Use of inla.mesh.2d
mesh = inla.mesh.2d(loc=rbind(coordinates(PO_Ostrya_carpinifolia)),
                    boundary = list(int = bndint, out = bndext),
                    max.edge = c(25,100), cutoff = 3,
                    crs = domaineSP@proj4string@projargs)
# Plot the domain, boundaries, INLA mesh and points
par(mar=rep(1,4),mfrow=c(1,1))
plot(mesh, main = "", asp = 1)
lines(rbind(bndext$loc,bndext$loc[1,]),type="l",lwd=2)
lines(rbind(bndint$loc, bndint$loc[1,]), pch = 19, cex = .05, col = "red",
      lwd = 1,lty=2)
plot(PO_Ostrya_carpinifolia,add=T,col=4,pch=16,cex=0.5)
plot(domaineSP,add=T,border=1,lwd=2)
```



The next step is to define the prior distributions for the two hyperparameters (standard deviation and correlation range) of the Matern covariance function used for the latent Gaussian processes in the following. In the current model, we have one such process (`logSpDens`), written $W(s)$ in the model formula. Here we choose some moderately `extreme` threshold values for the standard deviation (`sigma`) and the correlation range (`range`), and we fix the prior probability of exceeding these values to 0.1. Another parameter α is fixed, where $\alpha - 1$ is the smoothness parameter of the Matérn covariance. We here use the default choice $\alpha = 2$. Note that the choice of this parameter is generally much less important than the other two hyperparameters that we estimate.

```
matern <- inla.spde2.pcmatern(mesh,
                              alpha = 2, # fractional operator which is
                              #related to the smoothness of the Gaussian field
                              prior.sigma = c(1, 0.1), # P(sigma > 1) = 0.1
                              prior.range = c(50, 0.1)) # P(range < 50 km) = 0.1
```

2.2 Constructing and fitting the model

We continue by defining the model structure and then fit the model.

```
cmp <- coordinates ~ -1 + Intercept(1) +
  logTGdens(main=log(f.tg(x,y)),model="offset") +
  elev(main = f.elev(x,y), model = "linear") +
  W(coordinates, model = matern)

#gc(reset=TRUE)
## FIT
```

```r
bru_options_reset() # reset inlabru options to defaults before modifying them
bru_options_set(control.compute = list(cpo=T, dic=T, mlik=T, waic=T,
                                       return.marginals.predictor = F,
                                       config=TRUE),
                control.inla = list(int.strategy = "eb", strategy = "gaussian"),
                verbose = FALSE,
                bru_max_iter = bru_max_iter)
T1 <- Sys.time()
fit <- lgcp(cmp, PO_Ostrya_carpinifolia,
            samplers = domaineSP,
            domain = list(coordinates = mesh),
            options = list(control.inla = list(int.strategy = "eb"),
                           verbose = FALSE)
)
T2 <- Sys.time()
fit1 <- fit # we keep a copy of the fit for later comparisons with other models
summary(fit)
```

```
inlabru version: 2.7.0
INLA version: 22.12.16
Components:
Intercept: main = linear(1)
logTGdens: main = const(log(f.tg(x, y)))
elev: main = linear(f.elev(x, y))
W: main = spde(coordinates)
Likelihoods:
  Family: 'cp'
    Data class: 'SpatialPointsDataFrame'
    Predictor: coordinates ~ .
Time used:
    Pre = 0.437, Running = 2.55, Post = 0.0375, Total = 3.02
Fixed effects:
             mean    sd 0.025quant 0.5quant 0.975quant    mode kld
Intercept -10.991 0.316    -11.611  -10.991    -10.371 -10.991   0
elev       -1.183 0.224     -1.623   -1.183     -0.744  -1.183   0

Random effects:
  Name    Model
    W SPDE2 model

Model hyperparameters:
             mean    sd 0.025quant 0.5quant 0.975quant  mode
Range for W 53.41 9.190      38.63    52.29      74.82 49.53
Stdev for W  1.61 0.179       1.30     1.59       2.01  1.54

Deviance Information Criterion (DIC) ...............: -5703.41
Deviance Information Criterion (DIC, saturated) ....: NA
Effective number of parameters .....................: -6647.29

Watanabe-Akaike information criterion (WAIC) ...: 1464.57
```

```
Effective number of parameters .................: 319.01

Marginal log-Likelihood:  -3993.30
CPO, PIT is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

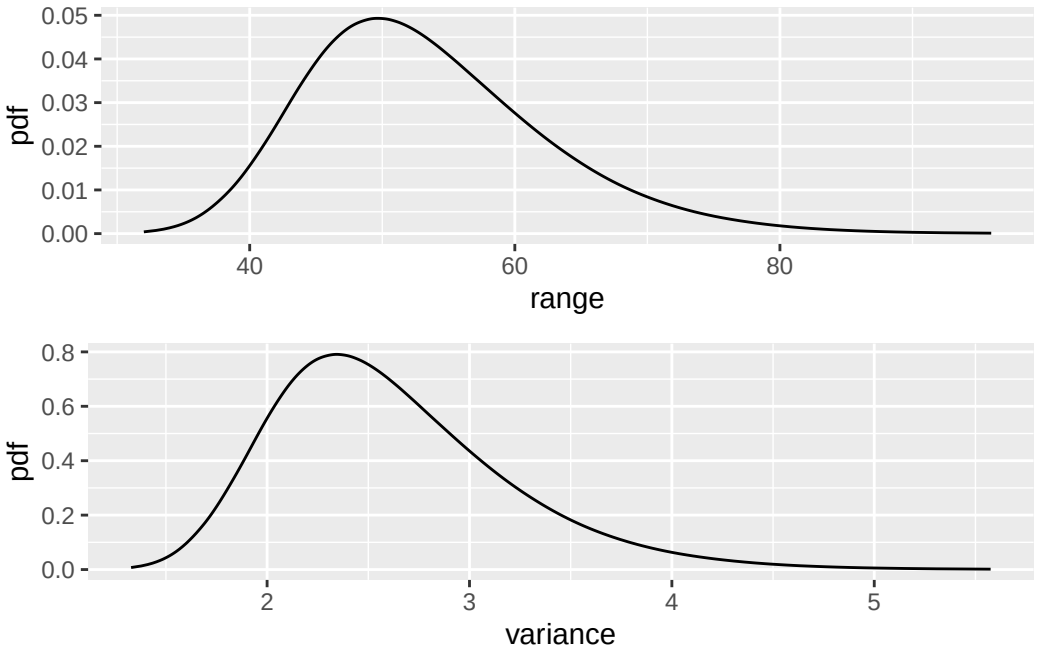
```
Tdiff <- difftime(T2, T1) ; paste('Model fitting time:',Tdiff)
```

```
[1] "Model fitting time: 5.11519336700439"
```

2.3 Posterior predictions, maps and visual displays

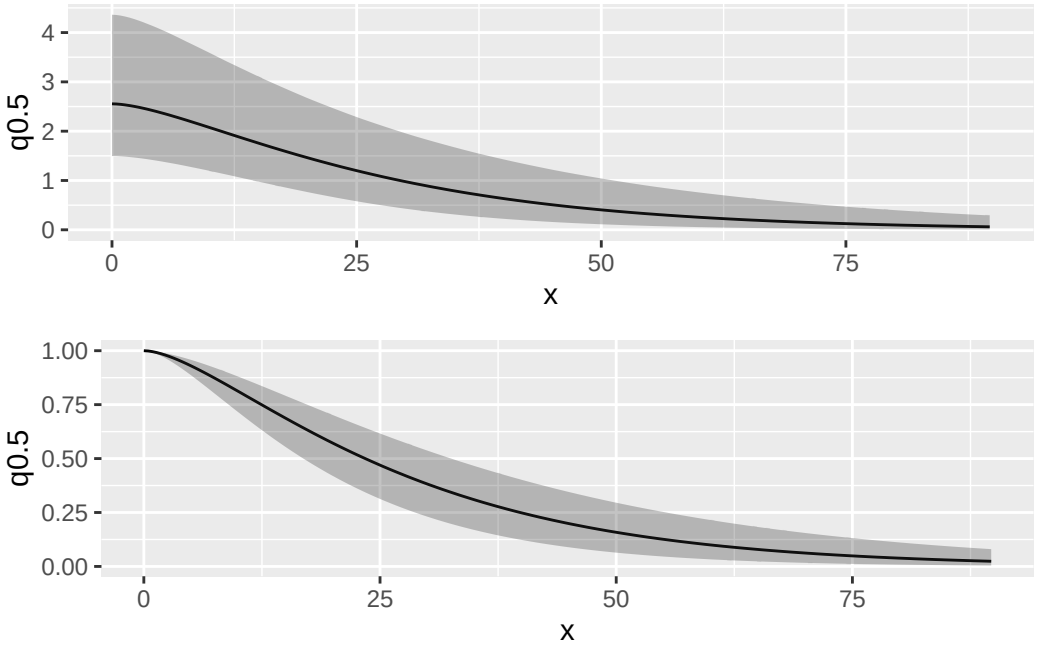
We can take a closer look at the posterior distribution of the estimated hyperparameters of the Matérn covariance function, namely its variance and range.

```
# Compute the posterior of each hyperparameter
spde.range <- spde.posterior(fit, "W", what = "range")
spde.logvar <- spde.posterior(fit, "W", what = "variance")
# Plots
range.plot <- plot(spde.range)
var.plot <- plot(spde.logvar)
multiplot(range.plot, var.plot)
```



Moreover, we can visualize the posterior median of the Matern covariance (and correlation) function by plotting its values as a function of spatial distance (in km).

```
covplot <- plot(spde.posterior(fit, "W", what = "matern.covariance"))
corplot <- plot(spde.posterior(fit, "W", what = "matern.correlation"))
multiplot(covplot, corplot)
```



Based on the posterior model, we can now calculate and plot the density of *Ostrya carpinifolia* corrected for the TG density. For the corrected species density, we here set the covariate $x_1(s)$ (which is supposed to represent sampling effort) to zero, that is, we assume that it is spatially constant. Note that due to the log-link function, adding $x_1(s) = 0$ in the linear predictor is equivalent to multiplying $\Lambda(s)$ by $\exp(0) = 1$, which does not change the value.

In the resulting map, we can interpret the spatial variability of the species density, which is a relative intensity as compared to the Target Group. However, since we have no precise information about how exhaustive the opportunistic sampling effort is, it is difficult to provide a good interpretation in terms of total abundance of the species.

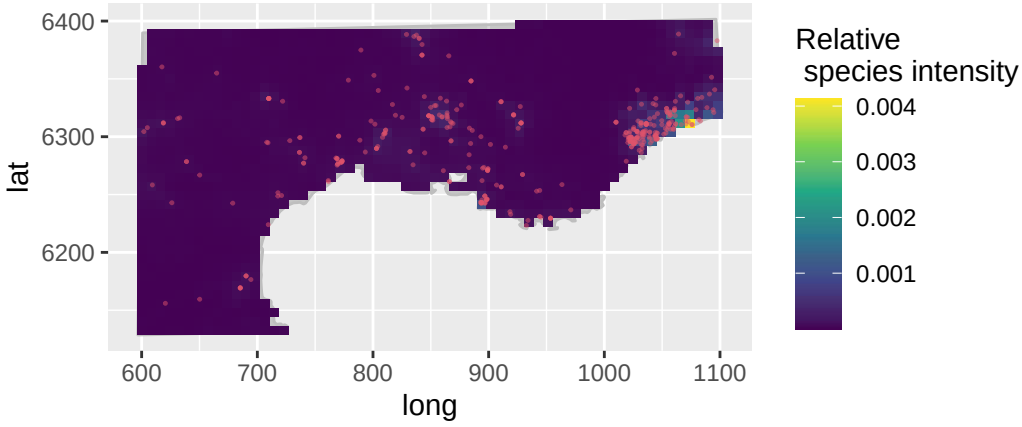
$$\log\left(\Lambda_{\text{species}}(s)\right) = \underbrace{\alpha_1}_{\text{Intercept}} + \beta_2 \times \underbrace{x_2(s)}_{\text{Elevation}} + \underbrace{W(s)}_{\text{Gaussian field}}$$

```
pxl <- pixels(mesh, mask=domaineSP, nx = nx, ny = ny)
pred <-  predict(fit, pxl,
                 ~ data.frame(species=exp(Intercept+elev+W)),
                 n.samples = n.samples)

p2 <- ggplot() + coord_equal()+
  gg(domaineSP, color = "grey") +
  gg(pred[,"mean"]) +
```

```
    scale_fill_continuous(type = "viridis") +
    gg(PO_Ostrya_carpinifolia, pch=19, color = 2, alpha = .5, size = 0.2)+
    labs(fill='Relative \n species intensity')
```

```
Regions defined for each Polygons
```

```
  print(p2)
```



3 Model 2 (bivariate): Shared Gaussian field for the log-TG density

3.1 Model structure

Our second model is a bivariate LGCP where the first variable corresponds to the presence-only observations of the focal species and the second variable corresponds to the presence-only observations of the Target-Group (TG) of species. Here, the intensity Λ is a vector (Λ, Λ_{TG}) with Λ the intensity of the focal species *ostrya carpinifolia* and Λ_{TG} the intensity of the TG. The idea behind this bivariate model is to use a shared spatial Gaussian random field W_0 between the two spatial intensity fields to estimate interactions between the two intensities. Λ_{TG} is only composed of an intercept and the shared spatial Gaussian field W_0, whose role is to to model an unmeasured covariate associated with the distribution of all the species of the TG. Since the TG is constructed using 193 species that can share very different ecological niches, we assume that this shared field W_0 is a proxy of the sampling pressure across the Mediterranean region. The two regression equations are as follows:

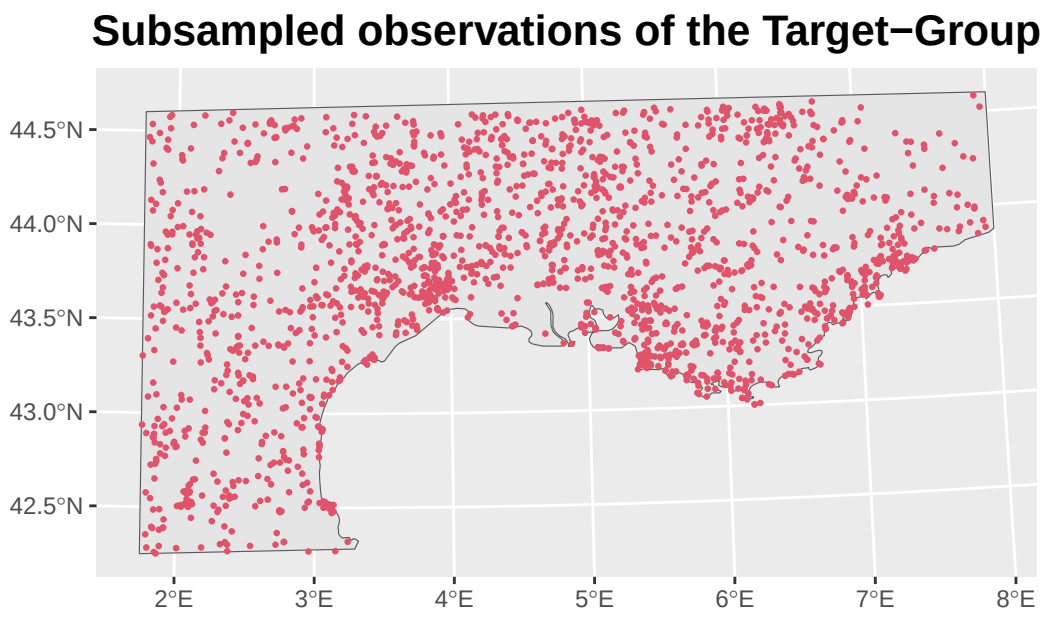
$$\ln(\Lambda(s)) = \underbrace{\alpha_1}_{\text{Intercept}} + \underbrace{x(s)}_{\text{Elevation}}\beta_1 + \underbrace{W(s)}_{\text{specific Gaussian field}} + \underbrace{W_0(s)}_{\text{shared Gaussian field}}$$

$$\ln\left(\Lambda_{TG}(s)\right) = \underbrace{\alpha_{TG}}_{\text{Intercept}} + \underbrace{W_0(s)}_{\text{shared Gaussian field}}$$

Preparing the Target-Group data

Here, the observations of the Target-Group represent more than 60,000 observations. This could lead to very time-consuming estimation runs, and so we randomly subsample them (using a fixed random seed for reproducibility). We also provide a map of the domain with the subsampled TG occurrences.

```
PO_TG = PO %>% filter(species %in% TGdf$spAdded)
set.seed(seed = 27)
idx_subsampling = sample(1:nrow(PO_TG), size = n_ss_TG)
PO_TG_ss = PO_TG[idx_subsampling,]
PO_TG_ss = SpatialPointsDataFrame(PO_TG_ss[,c("x", "y")], PO_TG_ss,
                                  proj4string = CRS(domaineSP@proj4string@projargs))
ggplot() + geom_sf(data=st_as_sf(domaineSP)) +
  geom_sf(data = st_as_sf(PO_TG_ss), color = 2, size = .5) +
  labs(title = "Subsampled observations of the Target-Group") +
  theme(plot.title = element_text(size = 16, face = "bold", hjust = 0.5))
```



3.2 Constructing and fitting the model

The implementation of bivariate model is slightly different from the previous univariate case since the `lgcp` function of `inlabru` works only for univariate models. We need to define two likelihoods, one for each intensity with the `like` function and then call the `bru` function which is a wrapper of the `inla` function of the `R-INLA` package. There are now two spatial fields, one that is included only in the model of the focal species to capture its specific niche, while the other is included in the models for the focal species and the Target Group to represent the sampling effort.

```r
cmp <- ~ -1+interceptOc(1)+interceptTG(1)+
  elevationOc(main = f.elev(x,y), model = "linear")+
  W0(main=coordinates, model=matern)+
  W(main=coordinates, model=matern)
# first regression equation for Target Group
lik1 <- like("cp",
             formula = coordinates ~ -1+interceptOc+elevationOc+W+W0,#
             data = PO_Ostrya_carpinifolia,
             samplers = domaineSP,
             domain = list(coordinates = mesh)
)
# second regression equation for focal species
lik2 <- like("cp",
             formula = coordinates ~ -1+interceptTG+W0,#
             data = PO_TG_ss,
             samplers = domaineSP,
             domain = list(coordinates = mesh)
)

bru_options_set(control.compute = list(cpo=T, dic=T, mlik=T, waic=T,
                                       return.marginals.predictor = F,
                                       config=TRUE),
                control.inla = list(int.strategy = "eb", strategy = "gaussian"),
                verbose = FALSE,
                bru_max_iter = bru_max_iter)

T1 <- Sys.time()
fit <- bru(cmp,lik1,lik2)
T2 <- Sys.time()
fit2 <- fit # again, we keep a copy for later comparisons with other models
summary(fit)
```

```
inlabru version: 2.7.0
INLA version: 22.12.16
Components:
interceptOc: main = linear(1)
interceptTG: main = linear(1)
elevationOc: main = linear(f.elev(x, y))
W0: main = spde(coordinates)
W: main = spde(coordinates)
Likelihoods:
  Family: 'cp'
    Data class: 'SpatialPointsDataFrame'
    Predictor: coordinates ~ -1 + interceptOc + elevationOc + W + W0
  Family: 'cp'
    Data class: 'SpatialPointsDataFrame'
    Predictor: coordinates ~ -1 + interceptTG + W0
Time used:
    Pre = 0.656, Running = 21.8, Post = 0.0641, Total = 22.5
Fixed effects:
```

```
              mean    sd 0.025quant 0.5quant 0.975quant   mode kld
interceptOc -6.098 0.443     -6.966   -6.098     -5.230 -6.098   0
interceptTG -2.740 0.358     -3.442   -2.740     -2.038 -2.740   0
elevationOc -1.009 0.215     -1.430   -1.009     -0.587 -1.009   0

Random effects:
  Name    Model
    W0 SPDE2 model
   W SPDE2 model

Model hyperparameters:
                mean     sd 0.025quant 0.5quant 0.975quant    mode
Range for W0 169.201 43.859    103.466  162.173     273.92 148.415
Stdev for W0   0.943  0.176      0.651    0.924       1.34   0.884
Range for W   42.660  7.003     30.498   42.106      58.03  41.034
Stdev for W    1.530  0.139      1.278    1.523       1.83   1.506

Deviance Information Criterion (DIC) ...............: -41778.09
Deviance Information Criterion (DIC, saturated) ....: NA
Effective number of parameters .....................: -46846.53

Watanabe-Akaike information criterion (WAIC) ...: 6821.86
Effective number of parameters .................: 1145.63

Marginal log-Likelihood:  -26354.19
CPO, PIT is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

```r
Tdiff <- difftime(T2, T1) ; paste('Model fitting time:', Tdiff)
```

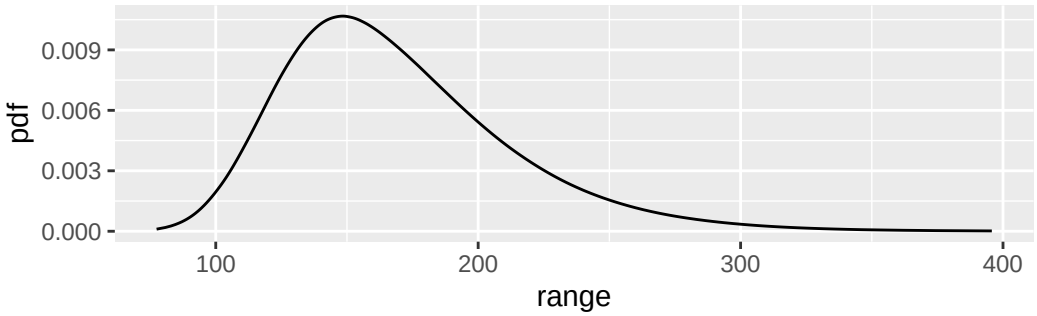
```
[1] "Model fitting time: 28.0336339473724"
```

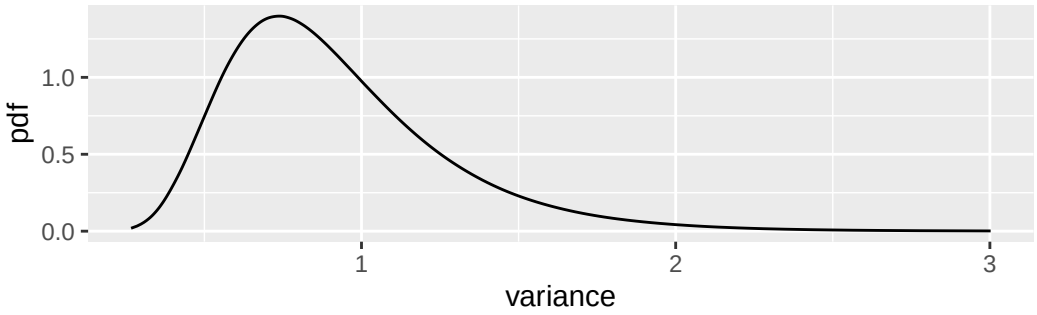
The estimated regression constant of the focal species is much lower than that of the Target Group, which makes sense since we have a lot more occurrences in the Target Group data. Elevation has a significant negative effect on the focal species. The range of the spatial field of the focal species is smaller (and therefore more *specific*) than that of the Target Group, which makes sense. It also has slightly stronger standard deviation than that of the target group, meaning the there is relatively strong spatial variation for the focal species that is not explained by elevation and the Target-Group intensity.

3.3 Posterior predictions, maps and visual displays

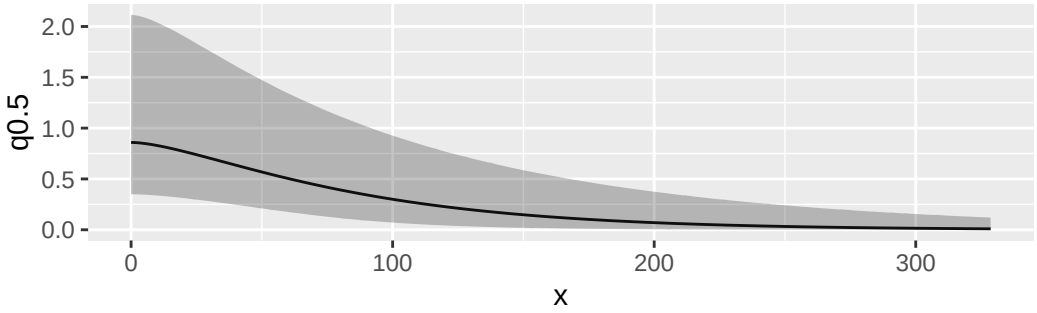
We provide a number of maps and visual displays based on the estimated posterior model. The posterior covariance structure of the shared field W_0 is as follows.

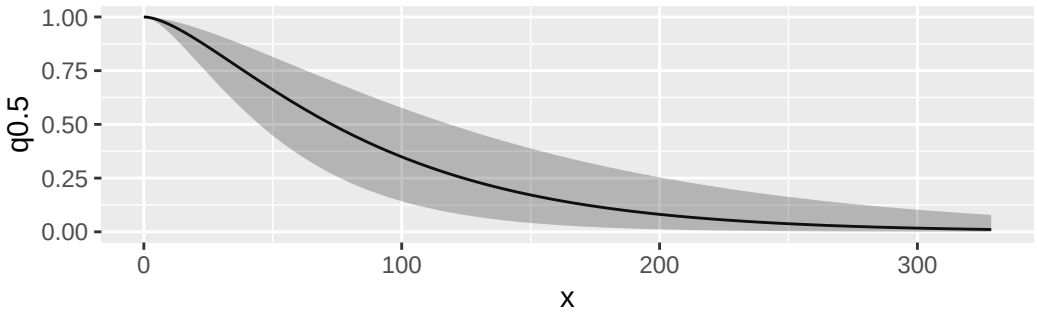
```r
spde.range.W0 <- spde.posterior(fit, "W0", what = "range")
spde.logvar.W0 <- spde.posterior(fit, "W0", what = "variance")
range.plot.W0 <- plot(spde.range.W0)
var.plot.W0 <- plot(spde.logvar.W0)
multiplot(range.plot.W0, var.plot.W0)
```





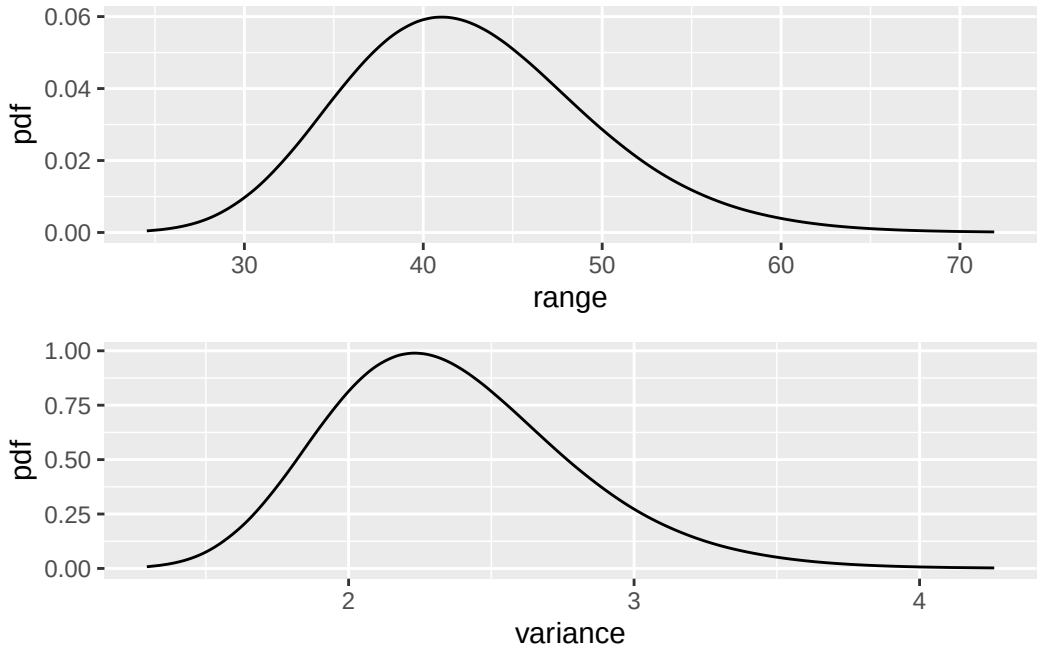

```
corplot.W0 <- plot(spde.posterior(fit, "W0", what = "matern.correlation"))
covplot.W0 <- plot(spde.posterior(fit, "W0", what = "matern.covariance"))
multiplot(covplot.W0, corplot.W0)
```



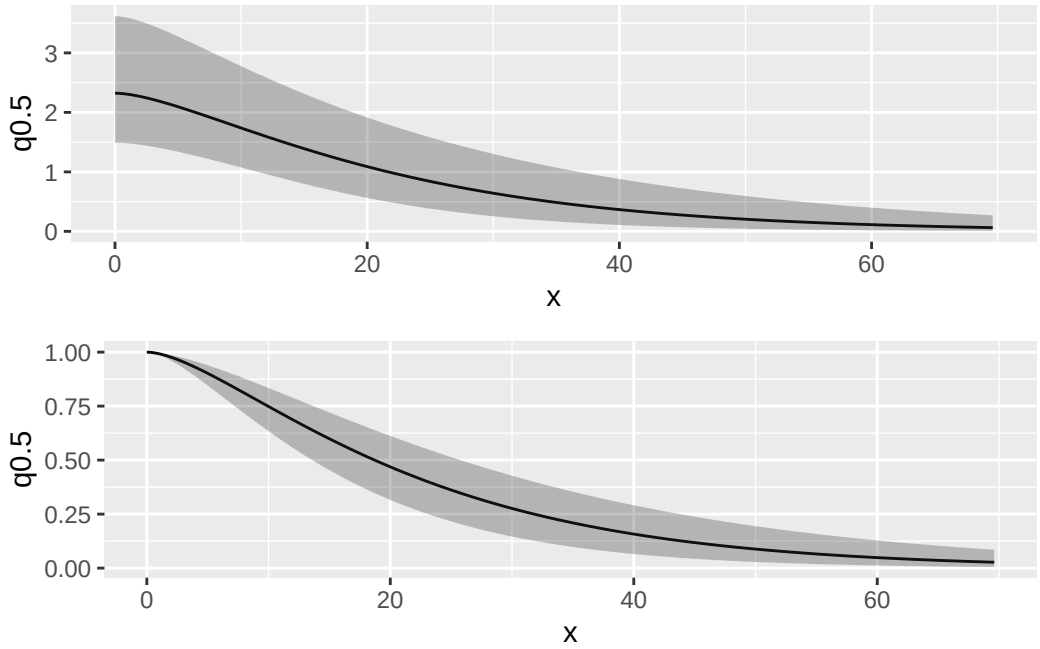


We can produce the same plots for the field W associated with the niche of the focal species.

```
spde.range.W <- spde.posterior(fit, "W", what = "range")
spde.logvar.W <- spde.posterior(fit, "W", what = "variance")
range.plot.W <- plot(spde.range.W)
var.plot.W <- plot(spde.logvar.W)
multiplot(range.plot.W, var.plot.W)
```




```
corplot.W <- plot(spde.posterior(fit, "W", what = "matern.correlation"))
covplot.W <- plot(spde.posterior(fit, "W", what = "matern.covariance"))
multiplot(covplot.W, corplot.W)
```

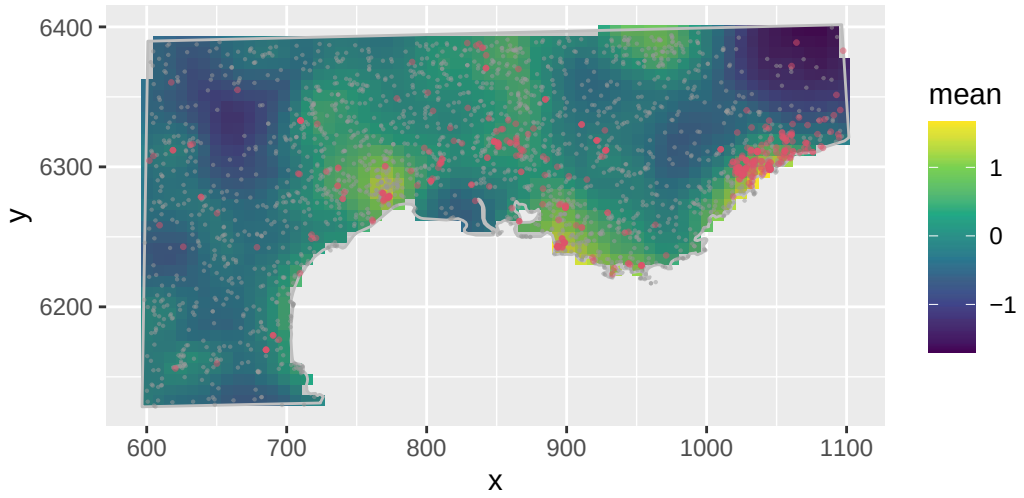


Next, we provide maps of certain components of the linear predictors for the Target Group and the focal species. In particular, we plot the field W_0 whose role is to capture the spatial variation of the sampling effort in the Target Group, and the field W whose role is to capture the spatial variation in the distribution of the species not explained by the elevation covariate. We also plot the full linear predictor $\log \Lambda(s)$, for which the spatial patterns should relatively closely follow those of the the observed presence-only occurrences of the focal species.

```
pxl <- pixels(mesh, mask=domaineSP, nx = nx, ny = ny)
pred_W0 <-  predict(fit, pxl,
                    ~ W0,
                    n.samples = n.samples)
pred_W <- predict(fit, pxl,
                  ~ W,
                  n.samples = n.samples)
pred_Occ <-  predict(fit, pxl,
                    ~ interceptOc+elevationOc+W+W0,
                    n.samples = n.samples)

plot_W0 <- ggplot() + coord_equal() +
  gg(pred_W0[,"mean"]) +
  scale_fill_continuous(type = "viridis") +
  gg(domaineSP, color = "grey") +
  gg(PO_TG_ss, size = 0.1, alpha = .5, col = "grey60") +
  gg(PO_Ostrya_carpinifolia, size = 0.5, alpha = 0.5, col=2)
```

```
Regions defined for each Polygons
```

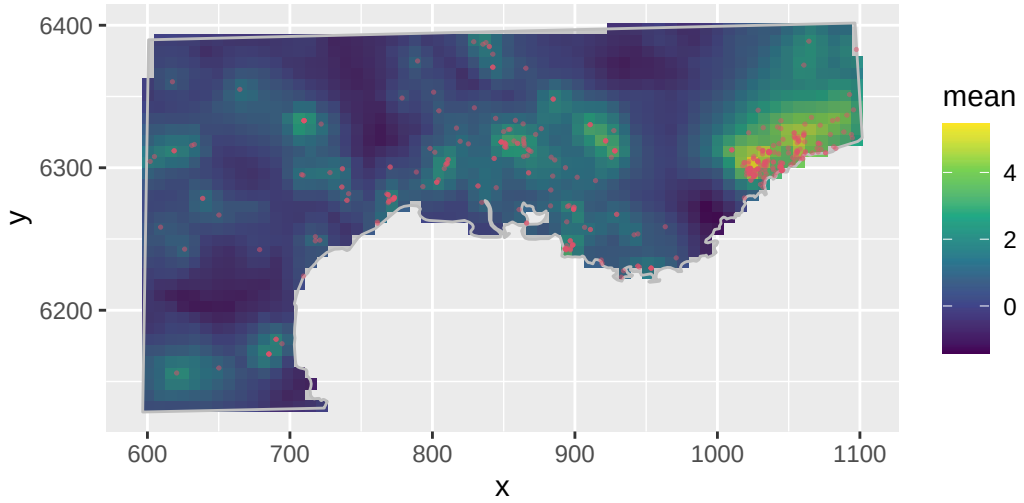
```
print(plot_W0)
```




```
plot_W <- ggplot() + coord_equal() +
  gg(pred_W[,"mean"]) +
  scale_fill_continuous(type = "viridis") +
  gg(domaineSP, color = "grey") +
  gg(PO_Ostrya_carpinifolia, size = 0.25, alpha = .5, col=2)
```

```
Regions defined for each Polygons
```

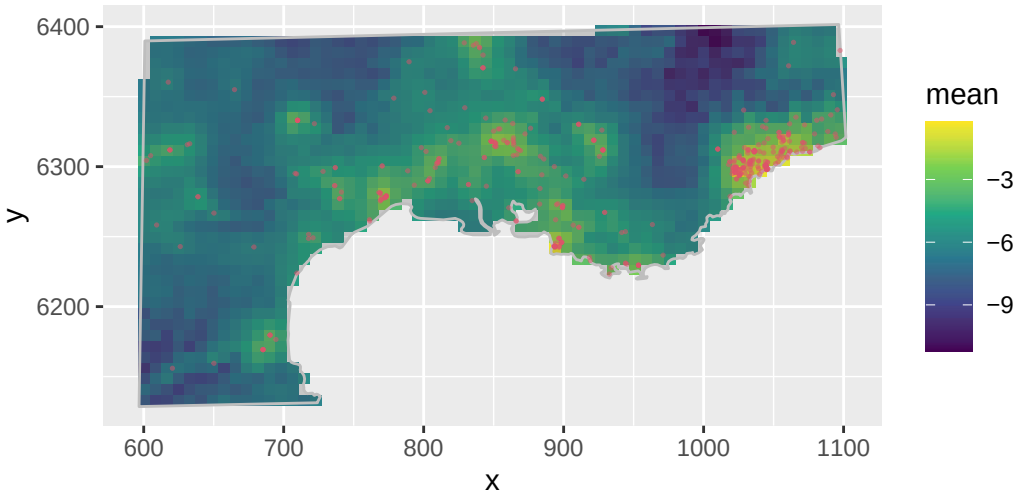
```
print(plot_W)
```




```
plot_Occ <- ggplot() + coord_equal() +
  gg(pred_Occ[,"mean"]) +
  scale_fill_continuous(type = "viridis") +
  gg(domaineSP, color = "grey") +
  gg(PO_Ostrya_carpinifolia, size = 0.25, alpha = 0.5, col=2)
```

Regions defined for each Polygons

```
print(plot_Occ)
```



We finally estimate the expected abundance along with uncertainty bounds. With the structure of our model, the abundance has Poisson distribution with intensity parameter Lambda (that is, with expected value Lambda). We use posterior simulation to provide a summary of the properties of Lambda.

```
Lambda <- predict(
  fit, ipoints(domaineSP, mesh),
    ~ sum(weight * exp(interceptOc+elevationOc+W+W0),
    n.samples = n.samples)
)
Lambda
```

```
     mean       sd   q0.025    q0.5  q0.975  median mean.mc_std_err
1 1352.166 67.32454 1238.842 1352.68 1479.53 1352.68        6.732454
  sd.mc_std_err
1      3.936525
```

Moreover, we can construct a relatively large uncertainty interval for the abundance (and not for its expected value) as follows. For a lower bound, we consider a low quantile of the Poisson distribution with the lower uncertainty bound of the expected value as its intensity; for an upper bound, we consider a high quantile of the Poisson distribution with the upper uncertainty bound of the expected value as its intensity. Note: this will tend to overestimate the uncertainty since we are taking low/high quantiles of both the intensity parameter and the Poisson distribution!

```
qpois(0.025, Lambda$q0.025)
```

```
[1] 1170
```

```
qpois(0.975, Lambda$q0.975)
```

```
[1] 1555
```

A more accurate way of obtaining such uncertainty intervals for the abundance would be to simulate a Poisson random variable for each pixel-based simulation of the intensity function. Note: this code may not work on some computers, and it is commented for this reason!

#Estimating abundance (variance includes variance in point locations): # (The following code is deactivated since it does not run on all machines:) #{r} #| warning: false # Nplant <- predict(#   fit, ipoints(domaineSP, mesh), #   ~ data.frame(#   N = round(Lambda$smin):round(Lambda$smax), #   dpois(round(Lambda$smin):round(Lambda$smax), #   lambda = sum(weight * exp(interceptOc+elevationOc+W1+W0)) #   ) #   ), #   n.samples = n.samples #) #inla.qmarginal(c(0.025, 0.5, 0.975), marginal = list(x = Nplant$N, y = Nplant$mean)) #inla.emarginal(identity, marginal = list(x = Nplant$N, y = Nplant$mean)) #Nplant$plugin_estimate <- dpois(Nplant$N, lambda = Lambda$mean) #ggplot(data = Nplant) + # geom_line(aes(x = N, y = mean, colour = "Posterior")) + # geom_line(aes(x = N, y = plugin_estimate, colour = "Plugin")) #

4 Model 3 (bivariate): PO and PA

Next, we load presence-absence data from a csv file and transform them to a `SpatialPointsDataFrame`. We generate separate objects for the two species under study. The csv file contains only the presences (one per line). To generate both presences and absences, we add a new column where we put 1 for a presence and 0 for an absence. For repeated observations in the same plot but at different times, we only keep one observation of presence, which makes sense for purely spatial modeling. For space-time modeling (which is beyond the scope of this tutorial), we could have presences and absences at the same plot but at different times. For all the sampling plots (x-y coordinnates) where the species was never present, we add a line with an absence to the data frame. The code below (which may not be the most efficient possible solution) implements these preprocessing step.

First, we read the presence-absence data into a data frame and then generate a `SpatialPointsDataFrame`.

```
head(PA.sp)
```

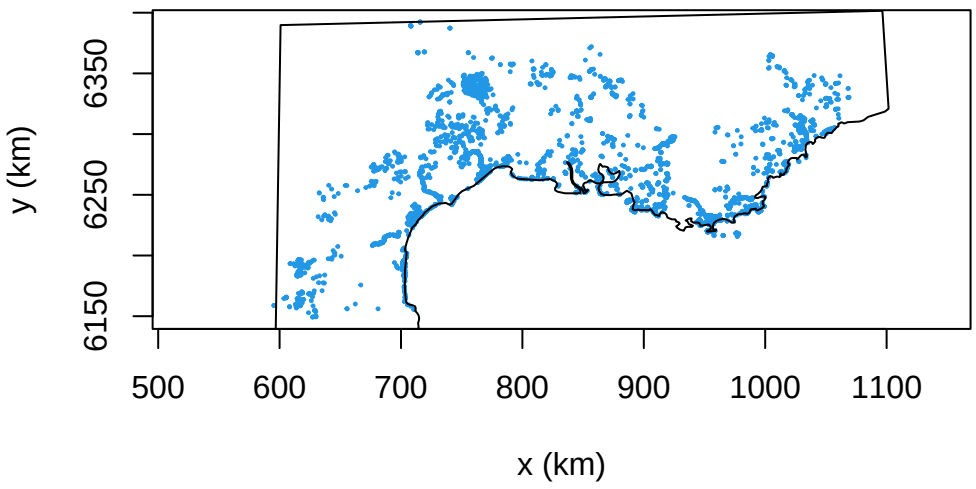
```
# A tibble: 6 x 9
# Groups:   lon [6]
    lon   lat ostrya senecio opuntia amorpha baccharis cortaderia robinia
  <dbl> <dbl> <lgl>  <lgl>   <lgl>   <lgl>   <lgl>     <lgl>      <lgl>
1  1.73  42.5 FALSE  FALSE   FALSE   FALSE   FALSE     FALSE      FALSE
2  1.73  42.5 FALSE  FALSE   FALSE   FALSE   FALSE     FALSE      FALSE
3  1.73  42.5 FALSE  FALSE   FALSE   FALSE   FALSE     FALSE      FALSE
4  1.73  42.5 FALSE  FALSE   FALSE   FALSE   FALSE     FALSE      FALSE
5  1.83  42.6 FALSE  FALSE   FALSE   FALSE   FALSE     FALSE      FALSE
6  1.84  42.6 FALSE  FALSE   FALSE   FALSE   FALSE     FALSE      FALSE
```

```
dim(PA.sp)
```

```
[1] 4833    9
```

We can map spatial coordinates of the presence-absence plots.

```
plot(PA.sp@coords, asp = 1, xlab = "x (km)", ylab = "y (km)", pch = 19,
     cex = .2, col = 4)
plot(domaineSP, add = TRUE)
```



How many observations do we have for our two focal species in the PA and PO data?

```
cat('presences of Ostrya carpinifolia in PA: ',
    sum(PA.sp@data$ostrya),'\n');
```

```
presences of Ostrya carpinifolia in PA:  90
```

```
cat('presences of Ostrya carpinifolia in PO: ',
    sum(PO$species=="Ostrya carpinifolia"),'\n');
```

```
presences of Ostrya carpinifolia in PO:  413
```

```
cat('presences of Senecio inaequidens in PA: ',
    sum(PA.sp@data$senecio),'\n');
```

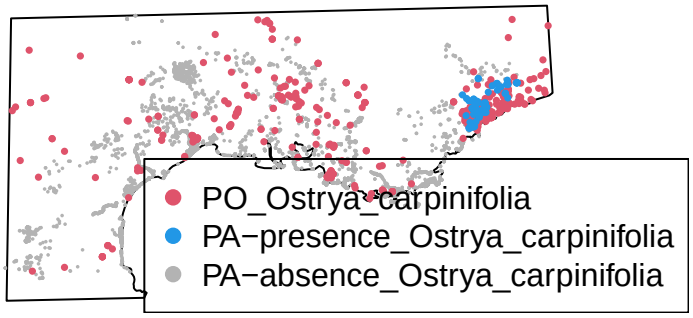
```
presences of Senecio inaequidens in PA:  89
```

```
cat('presences of Senecio inaequidens in PO: ',
    sum(PO$species=="Senecio inaequidens"))
```

```
presences of Senecio inaequidens in PO:  39
```

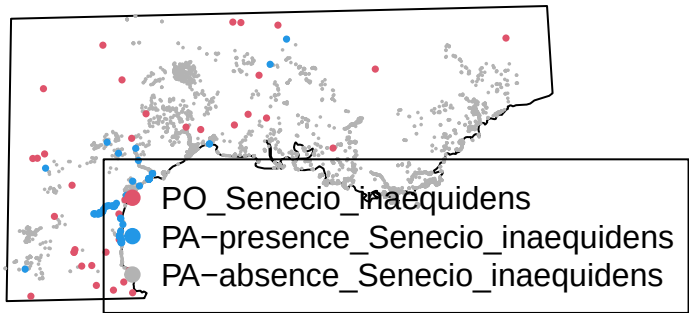
For each of the two species, we generate a map with three colors for Presence-Only, Presences from Presence-Absence and Absences from Presence-Absence.

```
plot(domaineSP)
plot(PA.sp[PA.sp@data$ostrya== 0,],add=T,col="gray70",pch=16,cex=0.25)
plot(PO_Ostrya_carpinifolia,add=T,col=2,pch=16,cex=0.5)
plot(PA.sp[PA.sp@data$ostrya== 1,],add=T,col=4,pch=16,cex=0.5)
legend("bottomright", c("PO_Ostrya_carpinifolia", "PA-presence_Ostrya_carpinifolia",
                        "PA-absence_Ostrya_carpinifolia"),
       pch = 19, col = c(2,4, "gray70"), cex = 1)
```



The presences in the presence-absence dataset of this species are all located in the Eastern cluster.

```
plot(domaineSP)
plot(PA.sp[PA.sp@data$senecio== 0,],add=T,col="gray70",pch=16,cex=0.25)
plot(PO_Senecio_inaequidens,add=T,col=2,pch=16,cex=0.5)
plot(PA.sp[PA.sp@data$senecio == 1,],add=T,col=4,pch=16,cex=0.5)
legend("bottomright", c("PO_Senecio_inaequidens", "PA-presence_Senecio_inaequidens",
                        "PA-absence_Senecio_inaequidens"),
       pch = 19, col = c(2,4, "gray70"), cex = 1)
```



The presences in the presence-absence dataset of this species seem to be mostly located along the coast and along a river.

4.1 Model structure

We now implement a bivariate model where the first variable corresponds to presence-only data modeled with an LGCP, and the second variable corresponds to the presence-absence data modeled with a Bernoulli distribution. To represent the presences-absences in a way the is coherent with a point process model, we use the complementary log-log link function for them. We start with the species Ostrya carpinifolia. The two regression equations are as follows:

$$\ln\left(\Lambda_{PO}(s)\right) = \underbrace{\alpha_{PO}}_{\text{Intercept}} + \beta_1 \times \underbrace{x(s)}_{\text{Elevation}} + \underbrace{W(s)}_{\text{specific Gaussian field}} + \underbrace{W_0(s)}_{\text{Gaussian field for sampling effort}}$$

$$\text{cloglog}\left(p_{PA}(s)\right) = \underbrace{\alpha_{PA}}_{\text{Intercept}} + \beta_1 \times \underbrace{x(s)}_{\text{Elevation}} + \underbrace{W(s)}_{\text{specific Gaussian field}}$$

Note: if we write μ_{PA} for the linear predictor of the presence-absence data (right-hand side of the PA equation), then we have $p_{PA} = 1 - \exp(-\exp(\mu))$, where $\Lambda = \exp(\mu)$ can be interpreted as the Poisson intensity for the presence-absence sampling plot. Indeed, if $N \sim \text{Poisson}(\exp(\mu))$, then $\Pr(N > 0) = p_{PA}$.

Let's first specify the two SPDE prior models for the spatial random effects. As before, we specify that (a priori) the standard deviation has a 10% chance of being larger than 1, and the correlation range has a 10% chance of being smaller than $50km$.

4.2 Constructing and fitting the model

```
maternW_species <- inla.spde2.pcmatern(mesh,
                                prior.sigma = c(1, 0.1),
                                prior.range = c(50, 0.1))
maternW_effort <- inla.spde2.pcmatern(mesh,
                                prior.sigma = c(1, 0.1),
                                prior.range = c(50, 0.1))
```

Next, we give a model formula that lists all the components that appear in one of the linear predictors.

```
cmp <- ~ -1+inter_PO(1)+inter_PA(1)+
  elev(main = f.elev(x,y), model = "linear")+
  W_effort(main=coordinates, model=maternW_effort)+
  W_species(main=coordinates, model=maternW_species)
```

We specify the structures of the two regression equations, the first one for the presence-only data (in lik1), second one for the presence-absence data (in lik2). The triangulation mesh we use for the spatial effects is the same as before.

To accurately control the inlabru options we need for estimation here, we first revert to the default options using `bru_options_reset` before setting the new options, where we provide details for the response distribution and the link functions in the `control.family` list. In particular, we use the complementary log-log link (`cloglog`) for presence-absence data, which means that we interpret observations as corresponding to a zero or positive count in their sampling plot. Finally, we run the estimation with `bru(...)` and show the summary of the fit.

```
lik1 <- like("cp",
             formula = coordinates ~ -1 + inter_PO + elev + W_effort + W_species,
             data = PO_Ostrya_carpinifolia,
             samplers = domaineSP,
             domain = list(coordinates = mesh)
)
lik2 <- like("binomial",
             formula = ostrya ~ -1 + inter_PA + elev + W_species,
             data = PA.sp,
             domain = list(coordinates = mesh)
)
bru_options_reset()
bru_options_set(verbose = FALSE,
                control.compute = list(cpo=T, dic=T, mlik=T, waic=T,
                                       return.marginals.predictor = T,
                                       config=TRUE),
                control.inla = list(int.strategy = "eb", strategy = "gaussian"),
                control.family = list(list(control.link = list(model = "log")),
                                      list(control.link = list(model = "cloglog"))),
                bru_max_iter = bru_max_iter
)
T1 <- Sys.time()
fit <- bru(cmp,lik1,lik2)
```

```r
T2 <- Sys.time()
fit3 <- fit # again, we keep a copy of the model for later comparisons
summary(fit)
```

```
inlabru version: 2.7.0
INLA version: 22.12.16
Components:
inter_PO: main = linear(1)
inter_PA: main = linear(1)
elev: main = linear(f.elev(x, y))
W_effort: main = spde(coordinates)
W_species: main = spde(coordinates)
Likelihoods:
  Family: 'cp'
    Data class: 'SpatialPointsDataFrame'
    Predictor: coordinates ~ -1 + inter_PO + elev + W_effort + W_species
  Family: 'binomial'
    Data class: 'SpatialPointsDataFrame'
    Predictor: ostrya ~ -1 + inter_PA + elev + W_species
Time used:
    Pre = 0.655, Running = 13.7, Post = 0.237, Total = 14.6
Fixed effects:
           mean    sd 0.025quant 0.5quant 0.975quant   mode kld
inter_PO -6.955 0.516     -7.966   -6.955     -5.944 -6.955   0
inter_PA -5.008 0.631     -6.245   -5.008     -3.772 -5.008   0
elev     -1.598 0.246     -2.081   -1.598     -1.116 -1.598   0

Random effects:
  Name    Model
    W_effort SPDE2 model
   W_species SPDE2 model

Model hyperparameters:
                      mean     sd 0.025quant 0.5quant 0.975quant  mode
Range for W_effort  27.95  4.464      20.89    27.40      38.50 25.84
Stdev for W_effort   1.20  0.079       1.07     1.20       1.39  1.18
Range for W_species 94.25 20.653      62.62    92.17     148.56 84.51
Stdev for W_species  1.81  0.137       1.58     1.81       2.14  1.77

Deviance Information Criterion (DIC) ...............: -5156.19
Deviance Information Criterion (DIC, saturated) ....: NA
Effective number of parameters .....................: -6522.71

Watanabe-Akaike information criterion (WAIC) ...: 2111.12
Effective number of parameters .................: 462.51

Marginal log-Likelihood:  -4221.03
CPO, PIT is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

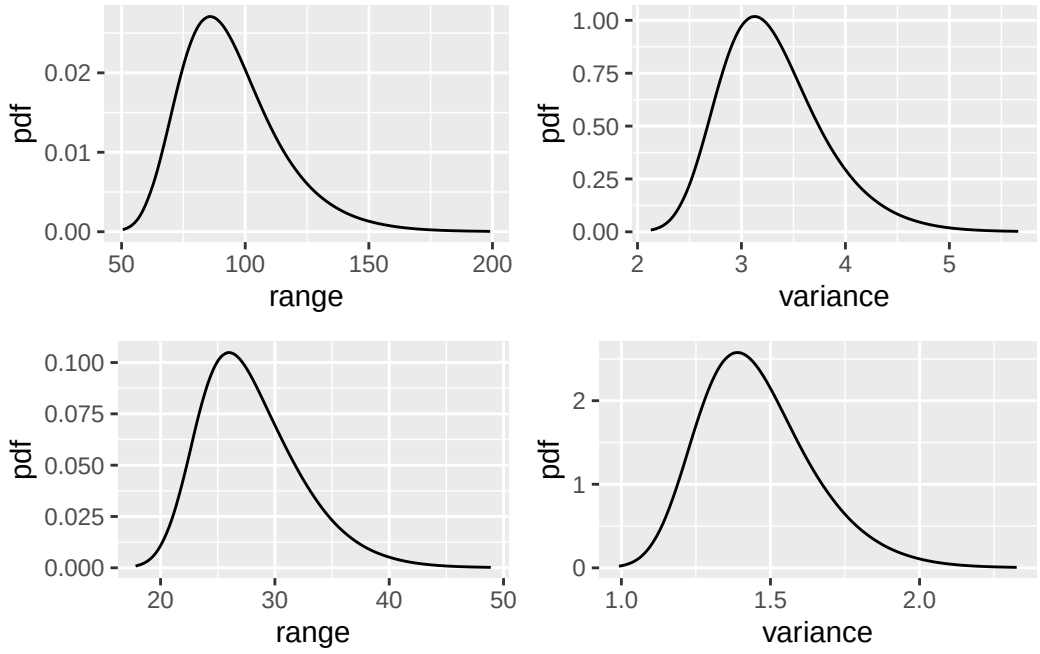
```
Tdiff <- difftime(T2, T1) ; paste('Model fitting time:',Tdiff)
```

```
[1] "Model fitting time: 21.3601276874542"
```

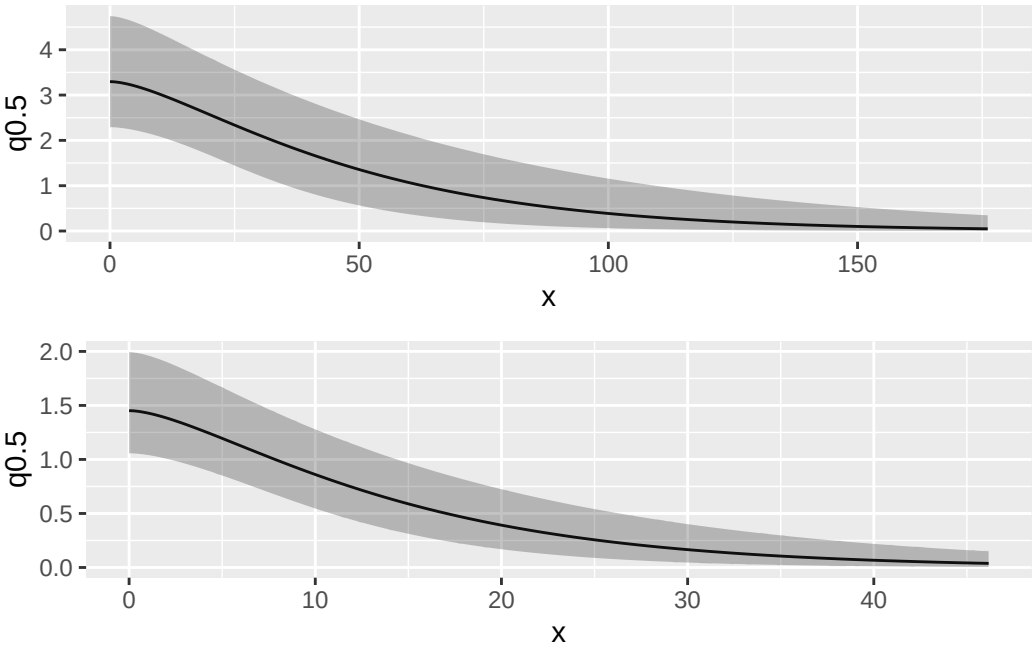
4.3 Posterior predictions, maps and visual displays

As before, we can provide plots to illustrate the posterior distributions of the Matérn covariance function and its hyperparameters.

```
spde.range <- spde.posterior(fit, "W_species", what = "range")
spde.logvar <- spde.posterior(fit, "W_species", what = "variance")
range.plot1 <- plot(spde.range)
var.plot1 <- plot(spde.logvar)
spde.range <- spde.posterior(fit, "W_effort", what = "range")
spde.logvar <- spde.posterior(fit, "W_effort", what = "variance")
range.plot2 <- plot(spde.range)
var.plot2 <- plot(spde.logvar)
multiplot(range.plot1, range.plot2,var.plot1,var.plot2,cols=2)
```




```
covplot1 <- plot(spde.posterior(fit, "W_species", what = "matern.covariance"))
covplot2 <- plot(spde.posterior(fit, "W_effort", what = "matern.covariance"))
multiplot(covplot1, covplot2)
```

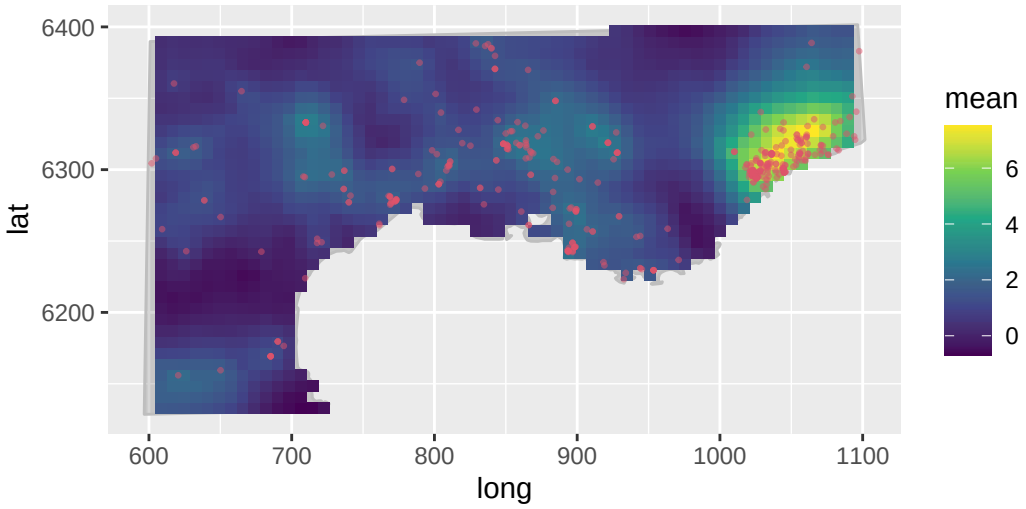


For visualisation and interpretation of results, we generate maps of posterior means for various quantities. Here, we plot W_{species}, W_{effort}, and the species abundance (on log-scale). For species abundance, we here use the predictor of the presence-absence data.

```
pxl <- pixels(mesh, mask=domaineSP, nx = nx, ny = ny)
pr_W_species <-  predict(fit, pxl,
                         ~ W_species,
                         n.samples = n.samples)
pr_W_effort <-  predict(fit, pxl,
                        ~ W_effort,
                        n.samples = n.samples)
pr_full <-  predict(fit, pxl,
                    ~ inter_PA + elev + W_species,
                    n.samples = n.samples)
# plot for W_species
plotW_species <- ggplot() + coord_equal() +
  xlim(domaineSP@bbox[1,]) + ylim(domaineSP@bbox[2,]) +
  gg(domaineSP, color = "grey") +
  gg(pr_W_species[,"mean"]) +
  scale_fill_continuous(type = "viridis") +
  gg(PO_Ostrya_carpinifolia, size = 0.5, alpha = 0.5, col=2) +
  gg(PA.sp[PA.sp@data$ostrya== 1,], size = 0.5, alpha = 0.5, col=5) +
  gg(PA.sp[PA.sp@data$ostrya== 0,], size = 0.5, alpha = 0.5, col="gray60")
```

```
Regions defined for each Polygons
```

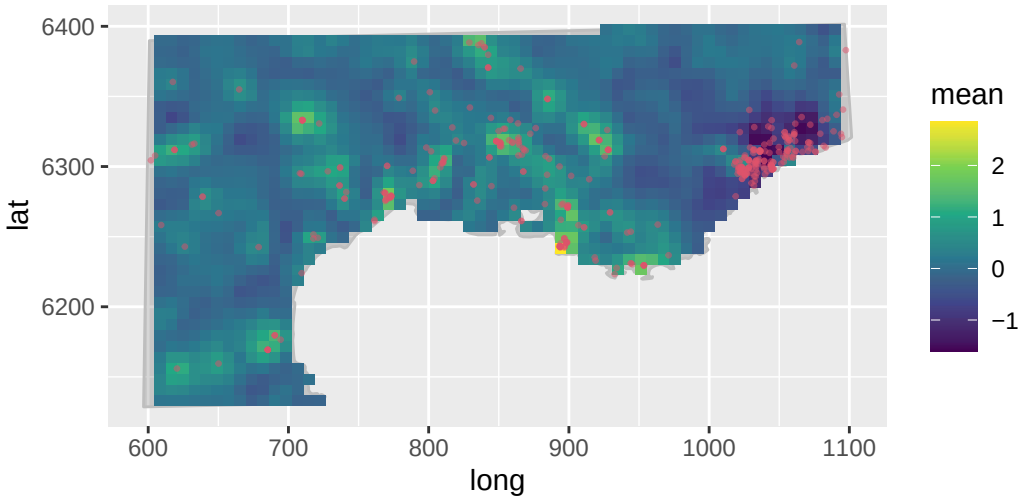
```
print(plotW_species)
```




```
# plot for W_effort
plotW_effort <- ggplot() +  coord_equal() +
  xlim(domaineSP@bbox[1,]) + ylim(domaineSP@bbox[2,]) +
  gg(domaineSP, color = "grey") +
  gg(pr_W_effort[,"mean"]) +
  scale_fill_continuous(type = "viridis") +
  gg(PO_Ostrya_carpinifolia, size = 0.5, alpha = 0.5, col=2) +
  gg(PA.sp[PA.sp@data$ostrya== 1,], size = 0.5, alpha = 0.5, col=5) +
  gg(PA.sp[PA.sp@data$ostrya== 0,], size = 0.5, alpha = 0.5, col="gray60")
```

```
Regions defined for each Polygons
```

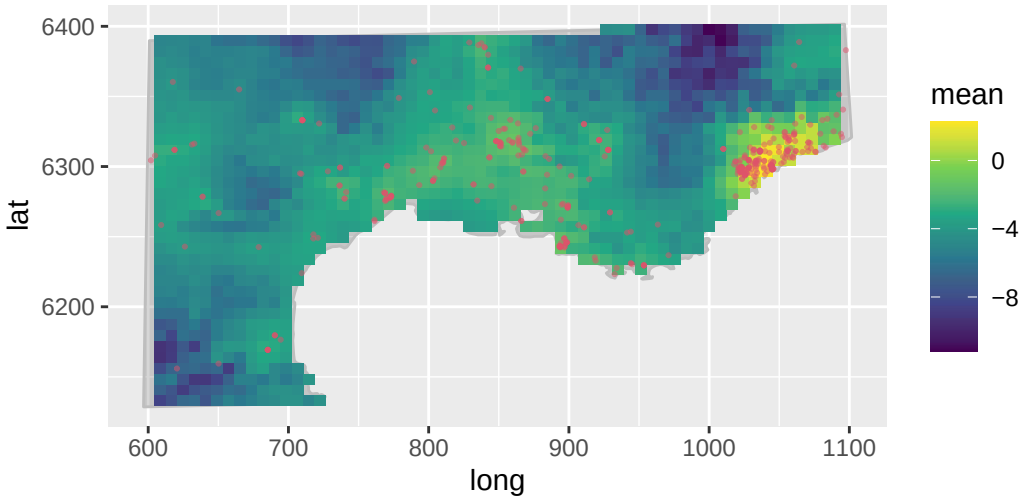
```
print(plotW_effort)
```




```
# plot for log-abundance
plot_full <- ggplot() +  coord_equal() +
   xlim(domaineSP@bbox[1,]) + ylim(domaineSP@bbox[2,]) +
  gg(domaineSP, color = "grey") +
  gg(pr_full[,"mean"]) +
  scale_fill_continuous(type = "viridis") +
  gg(PO_Ostrya_carpinifolia, size = 0.4, alpha = 0.5, col=2) +
  gg(PA.sp[PA.sp@data$ostrya== 1,], size = 0.4, alpha = 0.5, col=5) +
  gg(PA.sp[PA.sp@data$ostrya== 0,], size = 0.4, alpha = 0.5, col="gray60")
```

Regions defined for each Polygons

```
print(plot_full)
```



We can also estimate the total abundance, that is, the spatial aggregate of the point process intensity corrected for sampling effort. For this, we use as basis the predictor of the presence-absence data (where the spatial distance unit of the model is $1km$). Moreover, we also calculate a second prediction `Lambda_accurate` where we also take into account the size of the presence-absence sampling plots ($100m^2 = (0.01km)^2$) where the species has been sampled exhaustively. The value of `Lambda_accurate` can be interpreted as an estimate of the total number of individuals of the species in the study domain.

```
Lambda <- predict(fit, ipoints(domaineSP, mesh),
                  ~ sum(weight * exp(inter_PA + elev + W_species)),
                  n.samples = n.samples
)
Lambda
```

```
     mean       sd   q0.025     q0.5   q0.975   median mean.mc_std_err
1 13178.69 4024.492 8653.557 12649.29 22486.63 12649.29        804.8983
  sd.mc_std_err
1      794.6008
```

```
fact <- 1 / 0.01^2 # correct for the size of sampling plots
Lambda_accurate <- predict(fit, ipoints(domaineSP, mesh),
                  ~ fact * sum(weight * exp(inter_PA + elev + W_species)),
                  n.samples = n.samples
)
Lambda_accurate
```

```
      mean       sd   q0.025      q0.5    q0.975    median mean.mc_std_err
1 119855422 35104439 74583540 113012491 193073634 113012491         7020888
  sd.mc_std_err
```

```
1       5180031
```

5 Comparison of estimated efforts and species distributions

We here provide a visual comparison of the maps of the sampling effort and the species distribution for the species *Ostrya carpinifolia*. The maps are provided on the scale of the linear predictor; in particular, species distribution maps correspond to $\log(\Lambda_{\text{species}}(s))$. We first simulate the fields for the effort and the species according to the three fitted models. The effort in the three models is represented by their components $\beta_1 x_1(s)$, $W_0(s)$ and $W_0(s)$, respectively. For the distribution of the species we have the same representation in the three models:

$$\log(\Lambda_{\text{species}}(s)) = \underbrace{\alpha}_{\text{Intercept}} + \beta \times \underbrace{x(s)}_{\text{Elevation}} + \underbrace{W(s)}_{\text{specific Gaussian field}},$$

where β is the coefficient estimated for the Elevation covariate $x(s)$, and $W(s)$ represents the specific Gaussian field. Note that the model-specific notations in the previous sections can be slightly different.

```
pxl <- pixels(mesh, mask = domaineSP, nx = nx, ny = ny)
pred1 <-  predict(fit1, pxl,
                  ~ data.frame(effort = Intercept+logTGdens,
                               species = Intercept+elev+W),
                  n.samples = n.samples)
pred2 <-  predict(fit2, pxl, ~ data.frame(effort = W0,
                                          species = interceptOc+elevationOc+W),
                  n.samples = n.samples)
pred3 <-  predict(fit3, pxl, ~ data.frame(effort = W_effort,
                                          species = inter_PA + elev + W_species),
                  n.samples = n.samples)
```

Let us first compare the different spatial maps estimated for the sampling effort. Due to the different model specifications, the `intercepts` (that is, the spatial average) are usually not the same. However, if the models provide similar representations of the sampling effort, the spatial patterns should be similar. One solution to make the maps more easily comparable is to check whether the range of values are similar. For each map, we here first subtract the spatial average to make the maps better comparable.

```
pred1$effort$mean=pred1$effort$mean-mean(pred1$effort$mean)
pred2$effort$mean=pred2$effort$mean-mean(pred2$effort$mean)
pred3$effort$mean=pred3$effort$mean-mean(pred3$effort$mean)
M = max(c(pred3$effort$mean,pred2$effort$mean,pred1$effort$mean))
m = min(c(pred3$effort$mean,pred2$effort$mean,pred1$effort$mean))
p1_effort <- ggplot() + coord_equal() +
  gg(pred1$effort[,'mean']) +
  gg(domaineSP, color = "grey") +
  scale_fill_continuous(type = "viridis",limits=c(m,M))
```

```
Regions defined for each Polygons
```

```
p2_effort <- ggplot() + coord_equal() +
  gg(pred2$effort[,'mean']) +
  gg(domaineSP, color = "grey") +
```

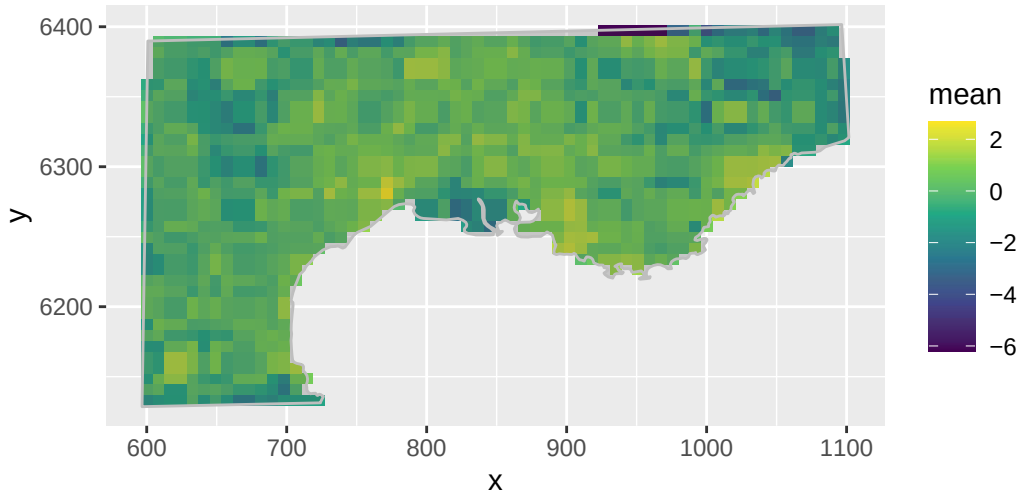
```
    scale_fill_continuous(type = "viridis",limits=c(m,M))
```

Regions defined for each Polygons

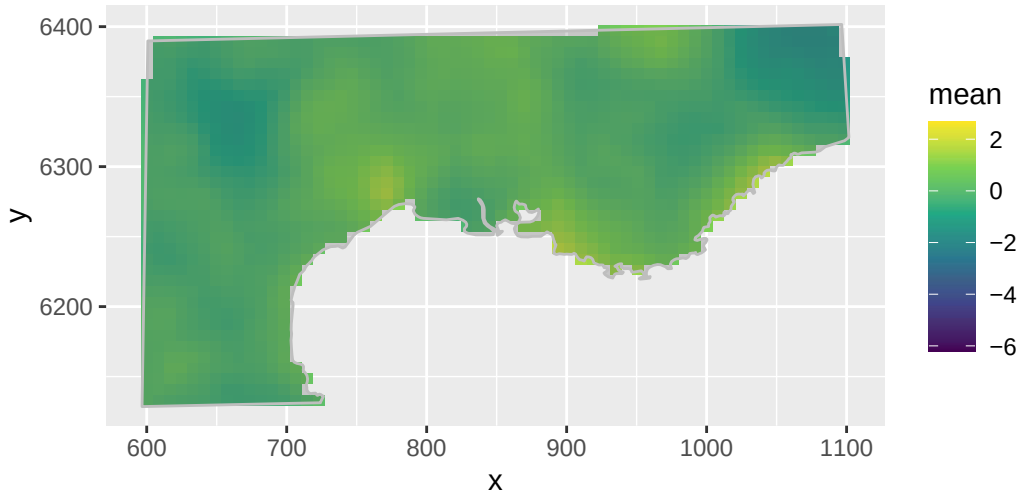
```
  p3_effort <- ggplot() + coord_equal() +
    gg(pred3$effort[,'mean']) +
    gg(domaineSP, color = "grey") +
    scale_fill_continuous(type = "viridis",limits=c(m,M))
```

Regions defined for each Polygons

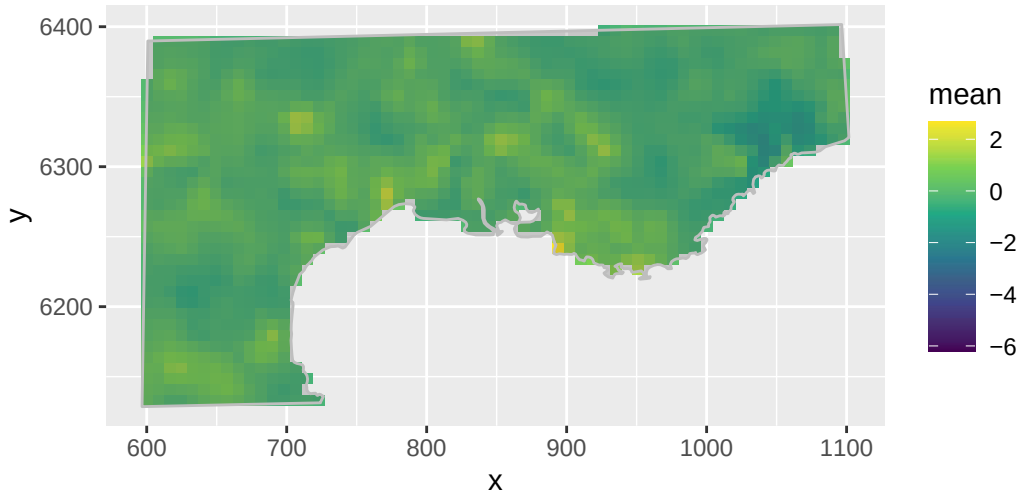
```
  print(p1_effort)
```




```
  print(p2_effort)
```




```
print(p3_effort)
```



Next, we also compare the different spatial maps estimated for the target species. Again, due to the different model specifications, the `intercepts` (that is, the spatial averages of the maps) are usually not the same,

and we first subtract the spatial average and then plot the three maps using exactly the same color scale.

```
pred1$species$mean = pred1$species$mean-mean(pred1$species$mean)
pred2$species$mean = pred2$species$mean-mean(pred2$species$mean)
pred3$species$mean = pred3$species$mean-mean(pred3$species$mean)
M = max(c(pred3$species$mean,pred2$species$mean,pred1$mean))
m = min(c(pred3$species$mean,pred2$species$mean,pred1$mean))
p1_species <- ggplot() + coord_equal() +
  gg(pred1$species[,'mean']) +
  gg(domaineSP, color = "grey") +
  scale_fill_continuous(type = "viridis",limits=c(m,M))
```

```
Regions defined for each Polygons
```

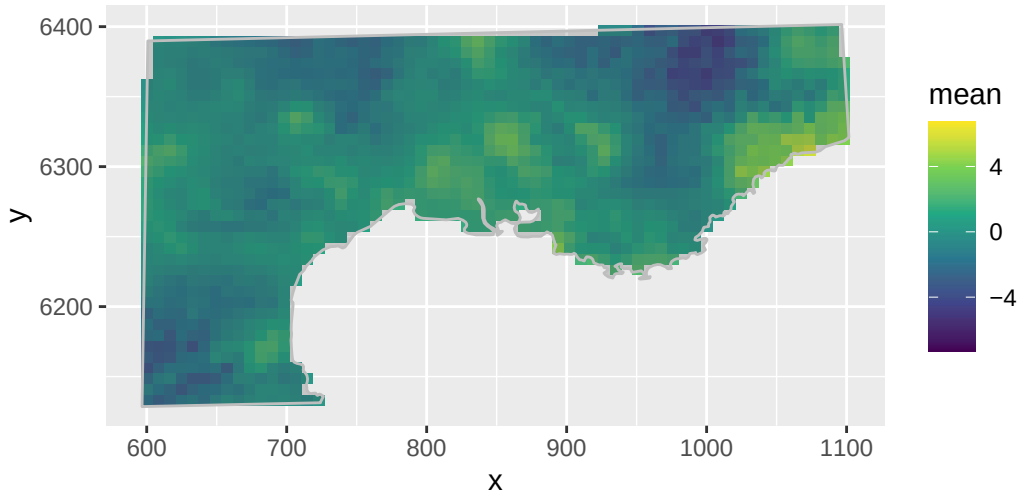
```
p2_species <- ggplot() + coord_equal() +
  gg(pred2$species[,'mean']) +
  gg(domaineSP, color = "grey") +
  scale_fill_continuous(type = "viridis",limits=c(m,M))
```

```
Regions defined for each Polygons
```

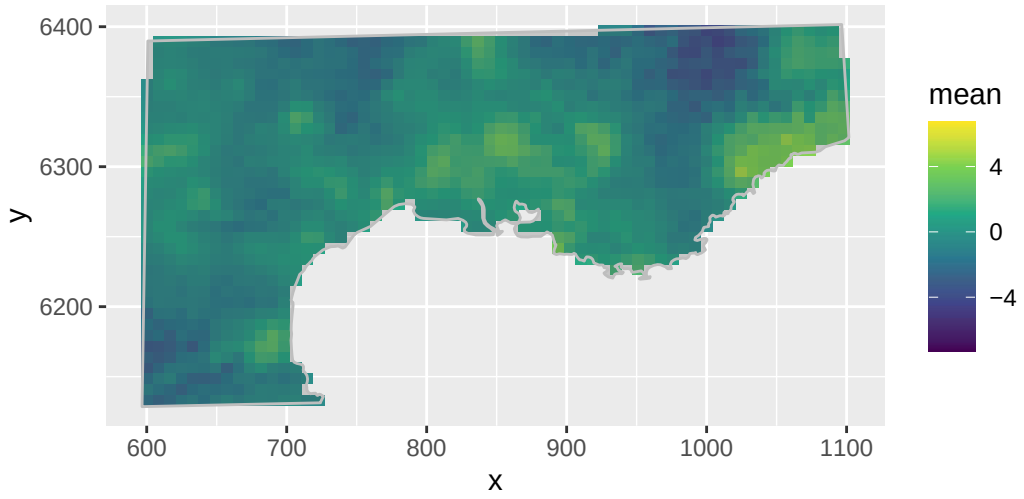
```
p3_species <- ggplot() + coord_equal() +
  gg(pred3$species[,'mean']) +
  gg(domaineSP, color = "grey") +
  scale_fill_continuous(type = "viridis",limits=c(m,M))
```

```
Regions defined for each Polygons
```

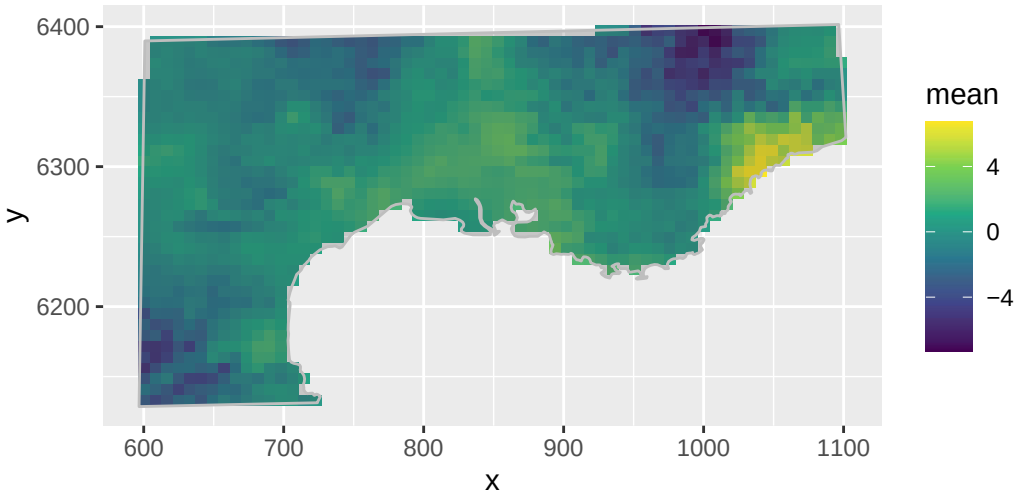
```
print(p1_species)
```




```
print(p2_species)
```




```
print(p3_species)
```



6 Exercices

Now, it's your turn to play and to implement your own model by taking inspiration from the above examples.

1. Implement a trivariate model for the Target Group and two focal species.
2. Implement the PO-PA model for the other species available in `PA.sp` and compare the new model with the one for *Ostrya carpinifolia*, in particular the estimation of the sampling effort.

Below, we provide example solutions for the exercice 1.

6.1 Example solution (Trivariate model)

```
cmp <- ~ -1+intercept_species1(1)+intercept_species2(1)+intercept_TG(1)+
  elevation_species1(main = f.elev(x,y), model = "linear")+
  elevation_species2(main = f.elev(x,y), model = "linear")+
  W0(main=coordinates, model=matern)+
  W_species1(main=coordinates, model=matern) +
  W_species2(main=coordinates, model=matern)
# first regression equation for first focal species
lik1 <- like("cp",
             formula = coordinates ~ -1+intercept_species1+ elevation_species1+W_species1+ W0,
             data = PO_Ostrya_carpinifolia,
             samplers = domaineSP,
             domain = list(coordinates = mesh)
)
# second regression equation for second focal species
lik2 <- like("cp",
```

```
              formula = coordinates ~ +intercept_species2+ elevation_species2+W_species2+ W0,
              data = PO_Senecio_inaequidens,
              samplers = domaineSP,
              domain = list(coordinates = mesh)
  )
  # third regression equation for Target Group
  lik3 <- like("cp",
              formula = coordinates ~ -1+intercept_TG+W0,#
              data = PO_TG_ss,
              samplers = domaineSP,
              domain = list(coordinates = mesh)
  )

  bru_options_reset()
  bru_options_set(control.compute = list(cpo=T, dic=T, mlik=T, waic=T,
                                         return.marginals.predictor = F,
                                         config=TRUE),
                  control.inla = list(int.strategy = "eb", strategy = "gaussian"),
                  verbose = FALSE,
                  bru_max_iter = bru_max_iter)
  T1 <- Sys.time()
  fit <- bru(cmp, lik1, lik2, lik3)
  T2 <- Sys.time()
  summary(fit)
```

```
inlabru version: 2.7.0
INLA version: 22.12.16
Components:
intercept_species1: main = linear(1)
intercept_species2: main = linear(1)
intercept_TG: main = linear(1)
elevation_species1: main = linear(f.elev(x, y))
elevation_species2: main = linear(f.elev(x, y))
W0: main = spde(coordinates)
W_species1: main = spde(coordinates)
W_species2: main = spde(coordinates)
Likelihoods:
  Family: 'cp'
    Data class: 'SpatialPointsDataFrame'
    Predictor:
        coordinates ~ -1 + intercept_species1 + elevation_species1 +
            W_species1 + W0
  Family: 'cp'
    Data class: 'SpatialPointsDataFrame'
    Predictor:
        coordinates ~ +intercept_species2 + elevation_species2 + W_species2 +
            W0
  Family: 'cp'
    Data class: 'SpatialPointsDataFrame'
    Predictor: coordinates ~ -1 + intercept_TG + W0
```

```
Time used:
    Pre = 0.786, Running = 51.5, Post = 0.0654, Total = 52.4
Fixed effects:
                      mean    sd 0.025quant 0.5quant 0.975quant   mode kld
intercept_species1 -6.099 0.439     -6.960   -6.099     -5.238 -6.099   0
intercept_species2 -7.879 0.940     -9.721   -7.879     -6.036 -7.879   0
intercept_TG       -2.741 0.357     -3.440   -2.741     -2.042 -2.741   0
elevation_species1 -1.008 0.215     -1.429   -1.008     -0.587 -1.008   0
elevation_species2 -0.637 0.276     -1.178   -0.637     -0.096 -0.637   0

Random effects:
  Name    Model
    W0 SPDE2 model
   W_species1 SPDE2 model
   W_species2 SPDE2 model

Model hyperparameters:
                        mean      sd 0.025quant 0.5quant 0.975quant    mode
Range for W0         165.606  42.231    100.619  159.480     265.70 147.206
Stdev for W0           0.956   0.194      0.643    0.932       1.40   0.881
Range for W_species1  43.299   7.426     31.187   42.431      60.31  40.449
Stdev for W_species1   1.518   0.137      1.265    1.512       1.80   1.502
Range for W_species2 376.779 116.508    197.797  360.214     652.23 329.280
Stdev for W_species2   1.127   0.358      0.585    1.074       1.98   0.974

Deviance Information Criterion (DIC) ...............: -42401.39
Deviance Information Criterion (DIC, saturated) ....: NA
Effective number of parameters .....................: -47562.51

Watanabe-Akaike information criterion (WAIC) ...: 7121.96
Effective number of parameters .................: 1216.39

Marginal log-Likelihood:  -26784.28
CPO, PIT is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

```r
Tdiff <- difftime(T2, T1) ; paste('Model fitting time:', Tdiff)
```

```
[1] "Model fitting time: 1.08502564827601"
```

```r
pxl <- pixels(mesh, mask=domaineSP, nx = nx, ny = ny)
pred1 <-  predict(fit, pxl,
                  ~ data.frame(effort=W0),
                  n.samples = n.samples)
pred2 <-  predict(fit, pxl, ~ data.frame(species = intercept_species1+elevation_species1+W_species1),
                  n.samples = n.samples)
pred3 <-  predict(fit, pxl, ~ data.frame(species = intercept_species2 + elevation_species2 + W_species2)
                  n.samples = n.samples)
```

```
p_effort <- ggplot() + coord_equal() +
  gg(pred1[,'mean']) +
  gg(domaineSP, color = "grey") +
  scale_fill_continuous(type = "viridis") #+
```

```
Regions defined for each Polygons
```

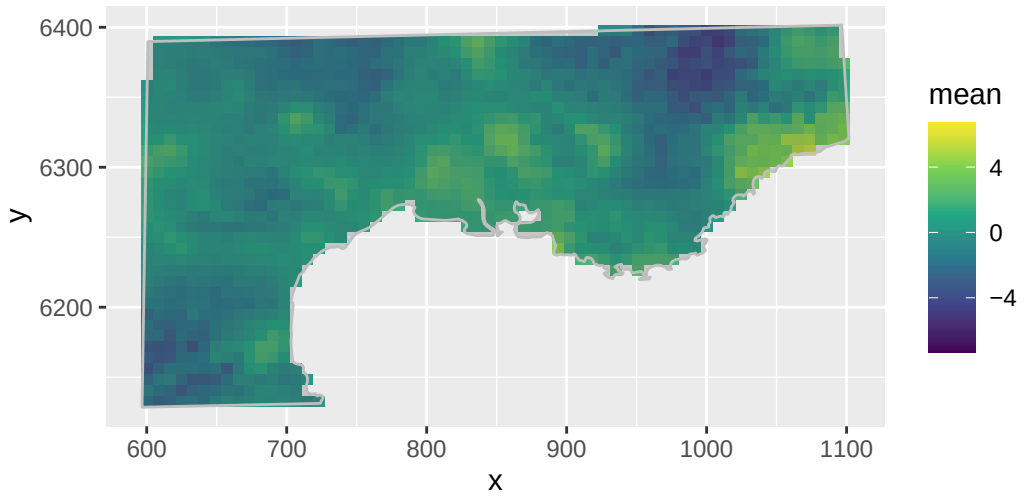
```
  # labs(fill='Relative \n sampling effort')
p_species1 <- ggplot() + coord_equal() +
  gg(pred2[,'mean']) +
  gg(domaineSP, color = "grey") +
  scale_fill_continuous(type = "viridis") #+
```

```
Regions defined for each Polygons
```

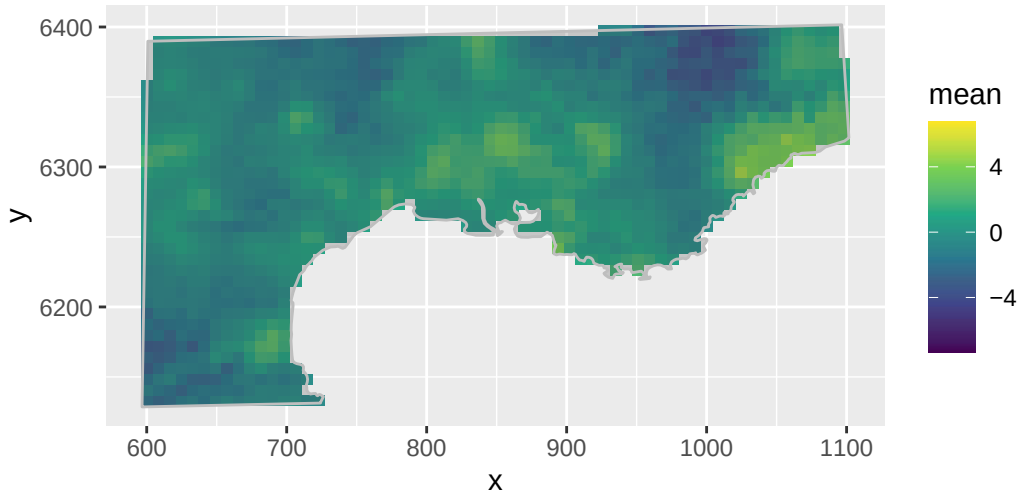
```
  # labs(fill='Relative \n sampling effort')
p_species2 <- ggplot() + coord_equal() +
  gg(pred3[,'mean']) +
  gg(domaineSP, color = "grey") +
  scale_fill_continuous(type = "viridis") #+
```

```
Regions defined for each Polygons
```

```
  # labs(fill='Relative \n sampling effort')
print(p1_species)
```



```
print(p2_species)
```




```
print(p3_species)
```