Quelques notions de statistique spatiale et bayésienne

Atelier INLA, Avignon mars 2025

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.

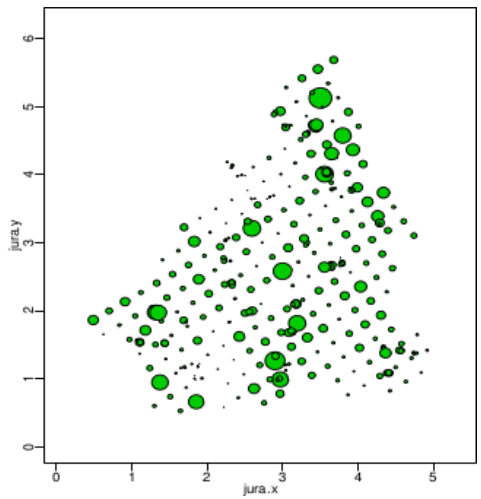
3. **Processus ponctuels**. Les points indiquent seulement la présence d'un événement. Les processus ponctuels peuvent être généralisés aux processus ponctuels marqués, lorsqu'une valeur (une marque) est associée aux points.

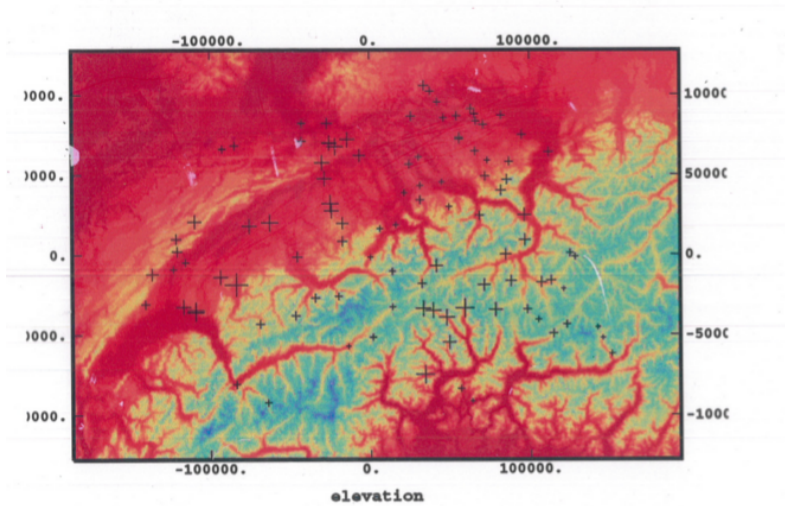
Données géostatistiques

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.



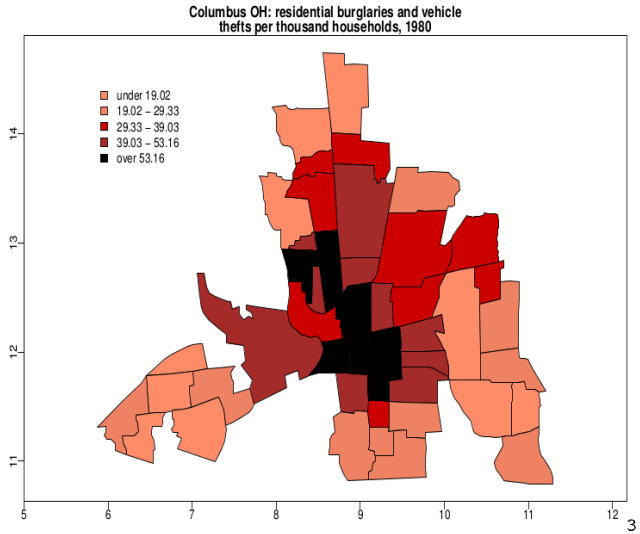


Données sur réseaux

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.

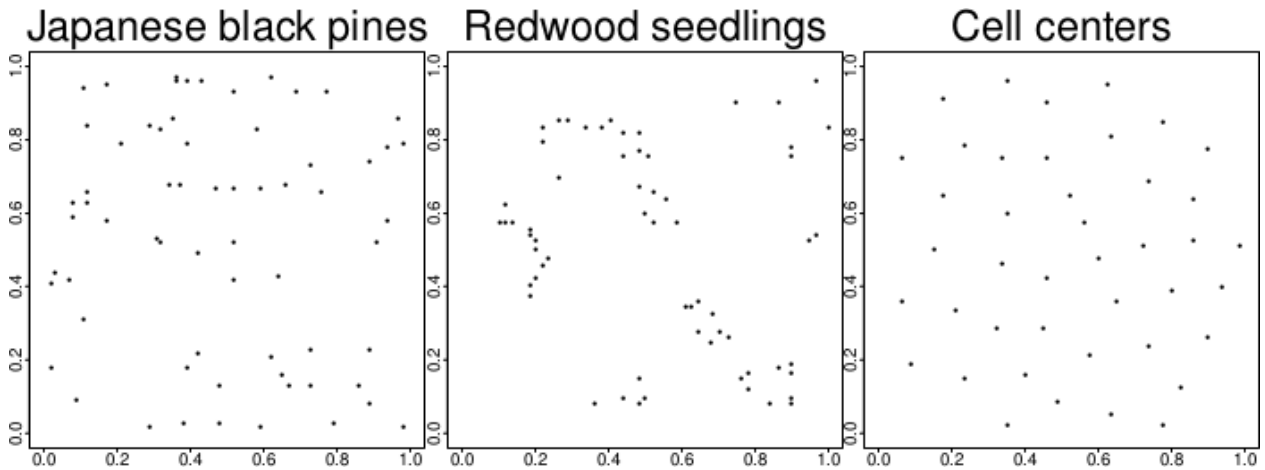


Processus ponctuels ou processus d'objets

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

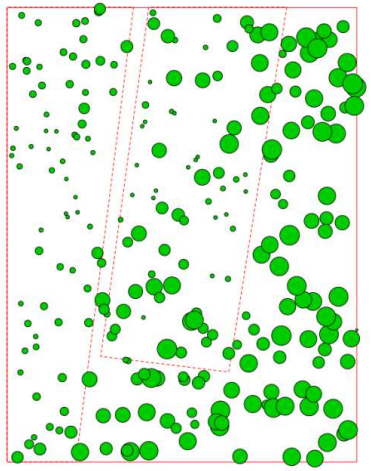
- Données non indépendantes.
- Pas de relation d'ordre dans les espaces $d \geq 2$.
- Vraisemblance adaptée ? Calculable ?
- Parfois : effets de bords, dépendances fortes,...

La géostatistique classique

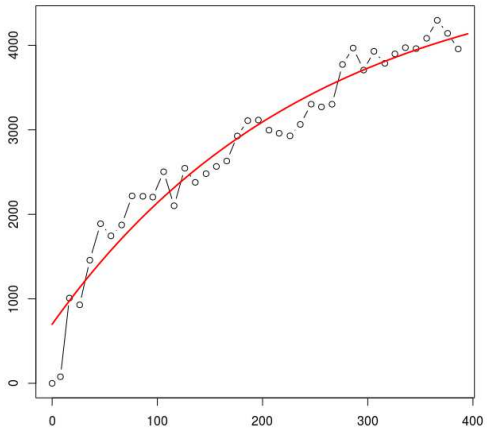
1. Réalisation unique et un suffisamment grand nombre de données résultant d'un échantillonnage du domaine d'étude,
2. Analyse structurale ou variographique (choix d'un modèle de variabilité spatiale et estimation des paramètres de cette variabilité sur un jeu de données),
3. Interpolation de la variable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

La géostatistique classique (en image)

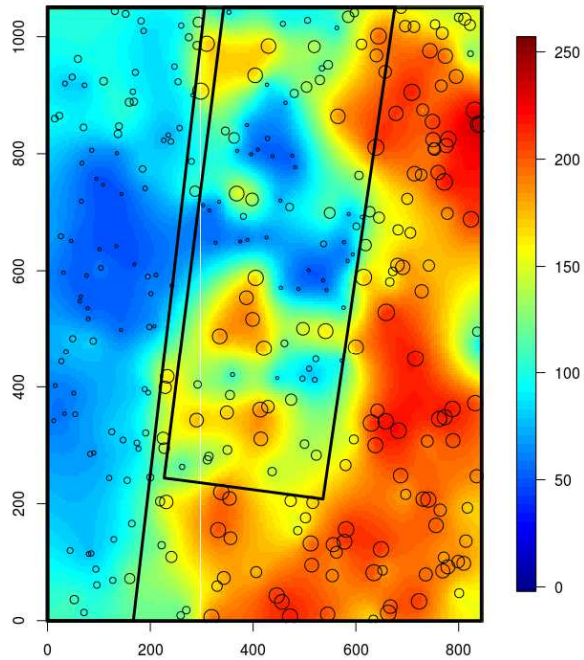
1. Echantillonnage



2. Modélisation



3. Interpolation



Les champs aléatoires

Champ aléatoire

Soit un **champ aléatoire** $Z(x)$ où $x \in D$ est un point du domaine d'étude.

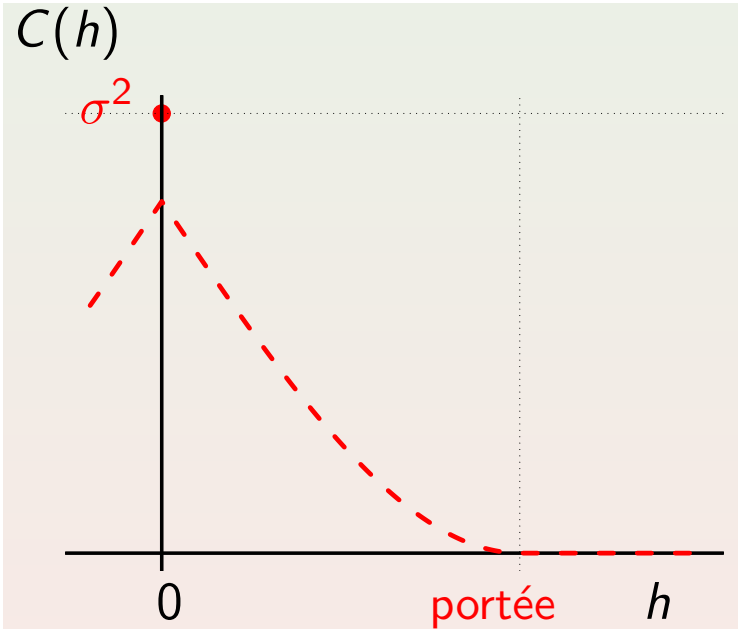
On suppose que $Z(x)$ est stationnaire d'ordre 2, les deux premiers moments de $Z(x)$ existent et sont invariants par translation :

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

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

Champ aléatoire

$C(h)$, 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 $\forall x_1, \ldots, x_n, \quad \forall \lambda_1, \ldots, \lambda_n$:

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

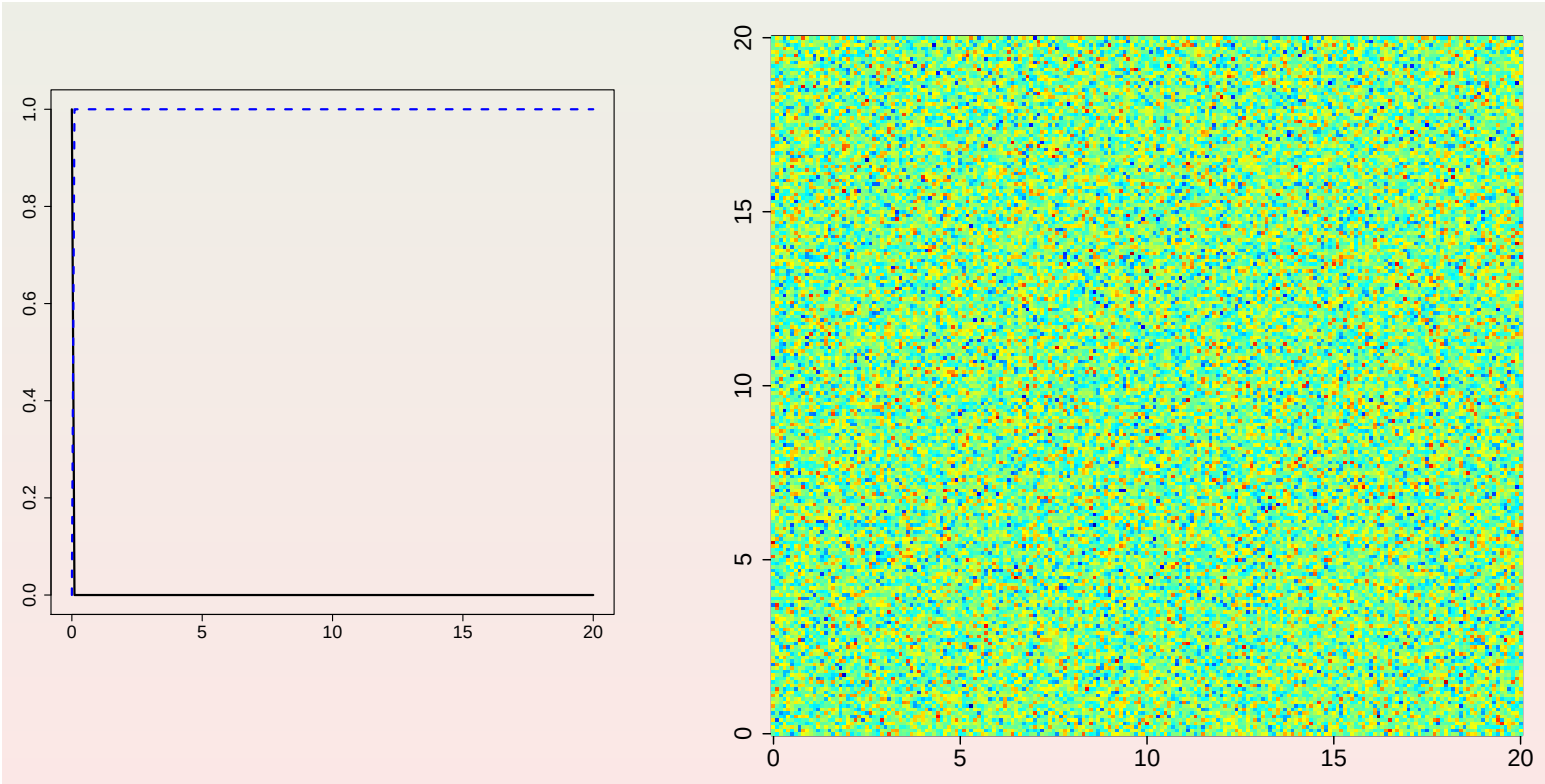
Champ aléatoire

A quoi ressemblent les champs générés par ces modèles ?

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

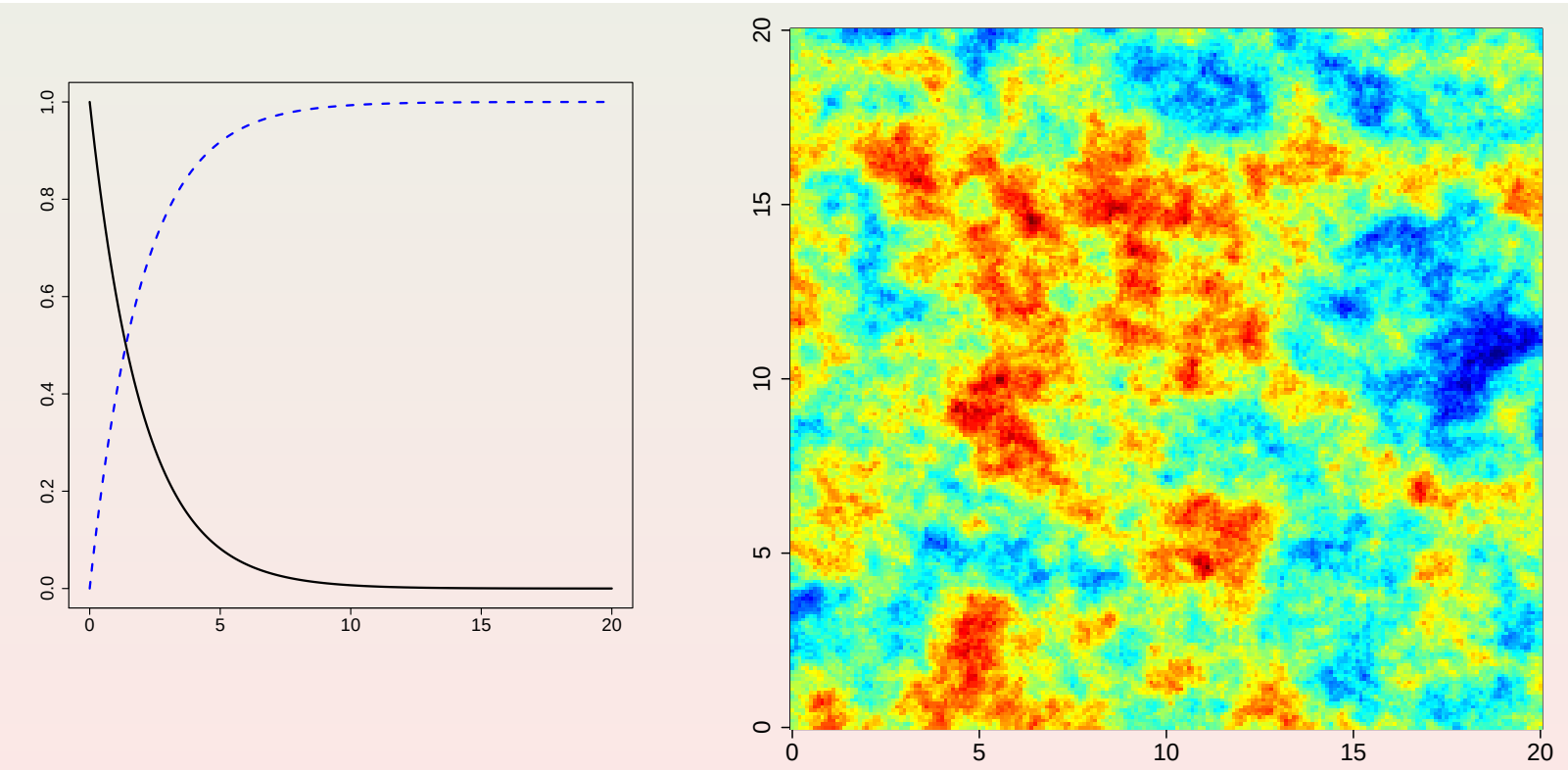
Champ aléatoire

Effet aléatoire pur (bruit blanc spatial)



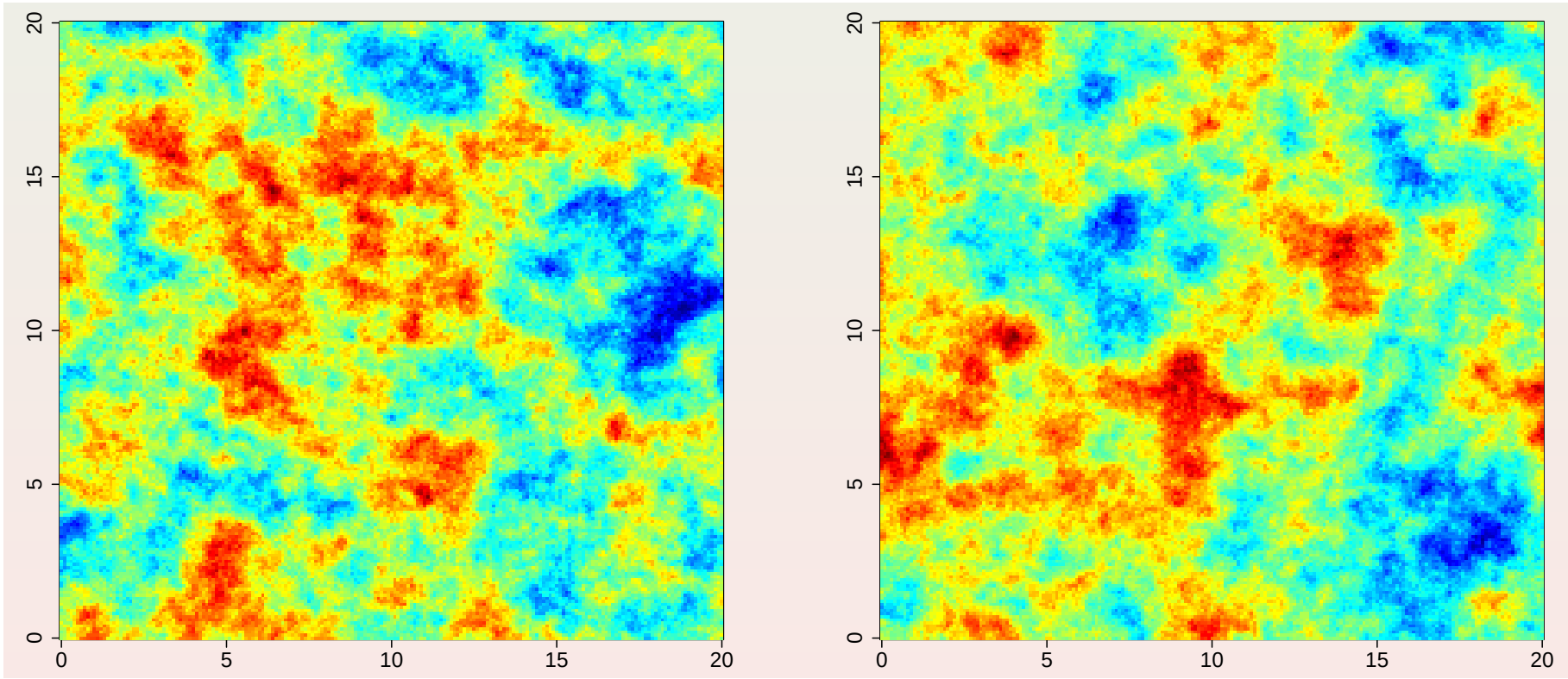
Champ aléatoire

Modèle de covariance exponentiel



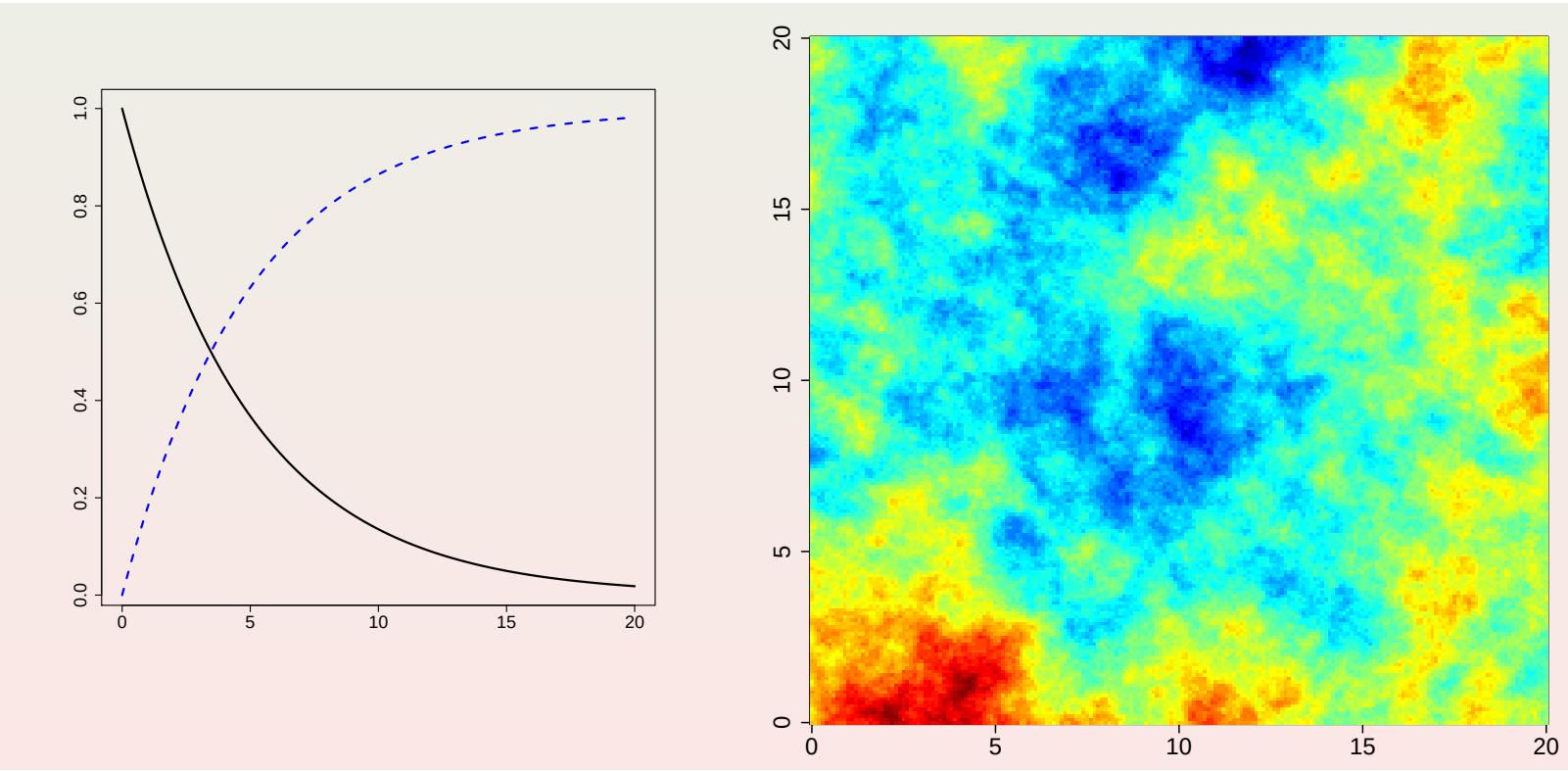
Champ aléatoire

Deux réalisations avec les mêmes paramètres



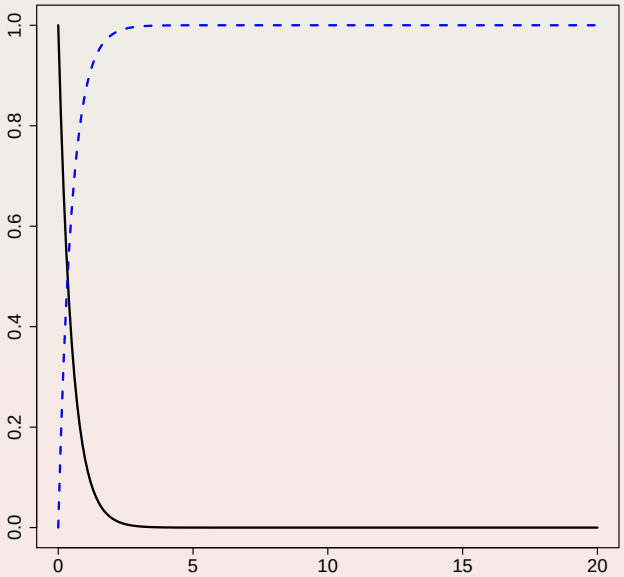
Champ aléatoire

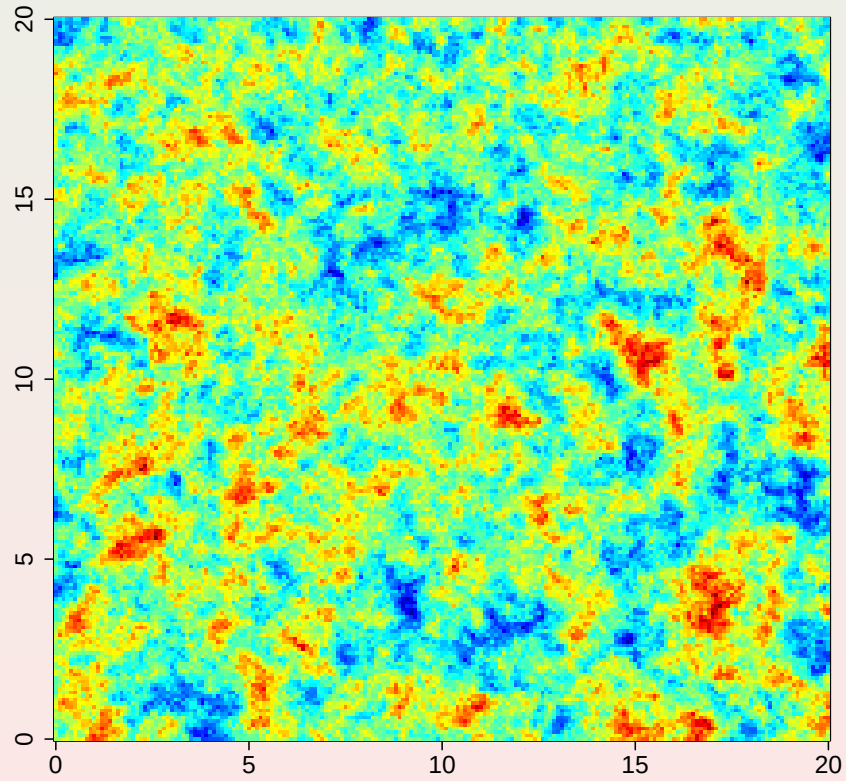
Modèle de covariance exponentiel avec une plus grande portée



Champ aléatoire

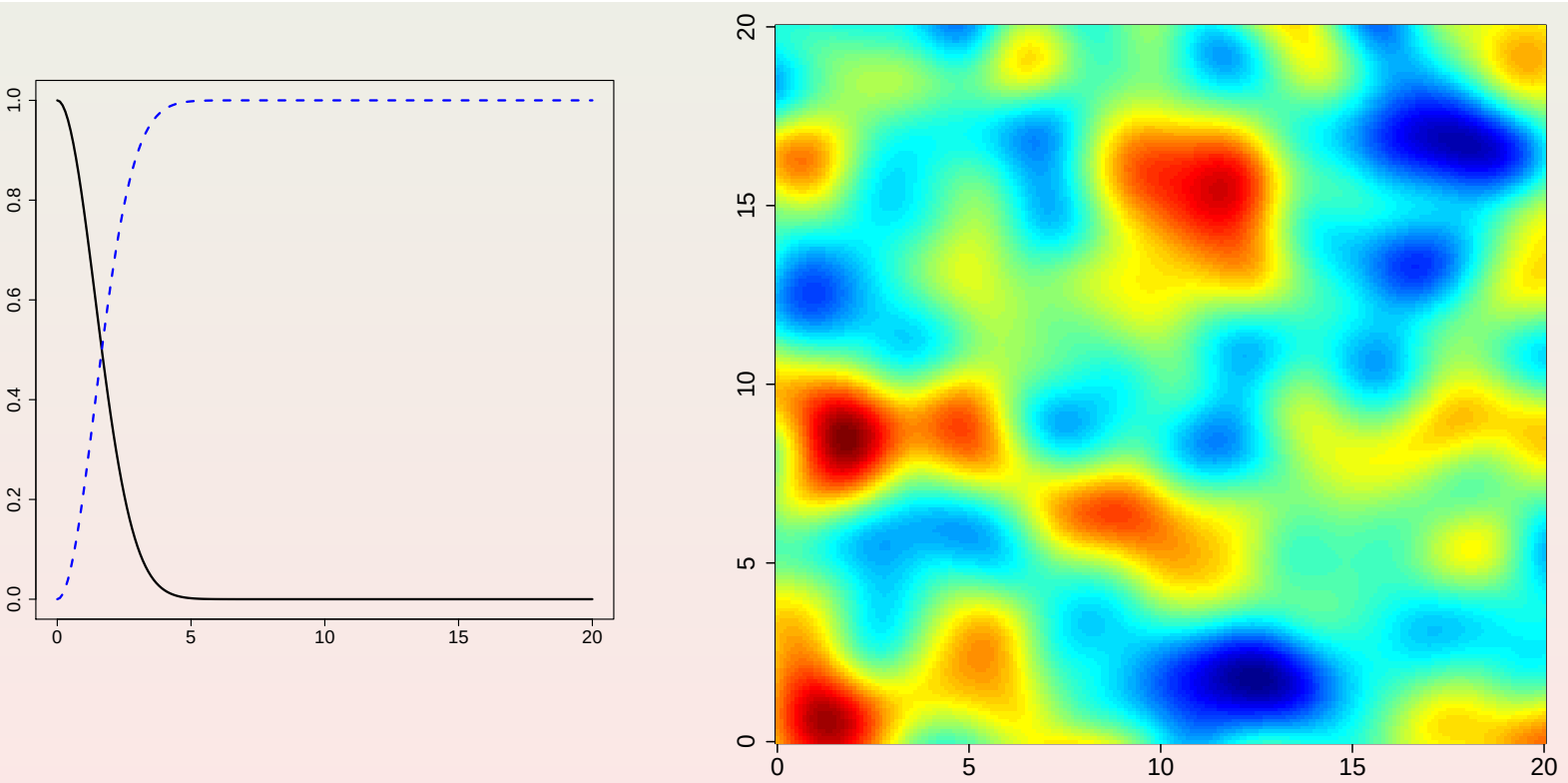
Modèle de covariance exponentiel avec une plus petite portée





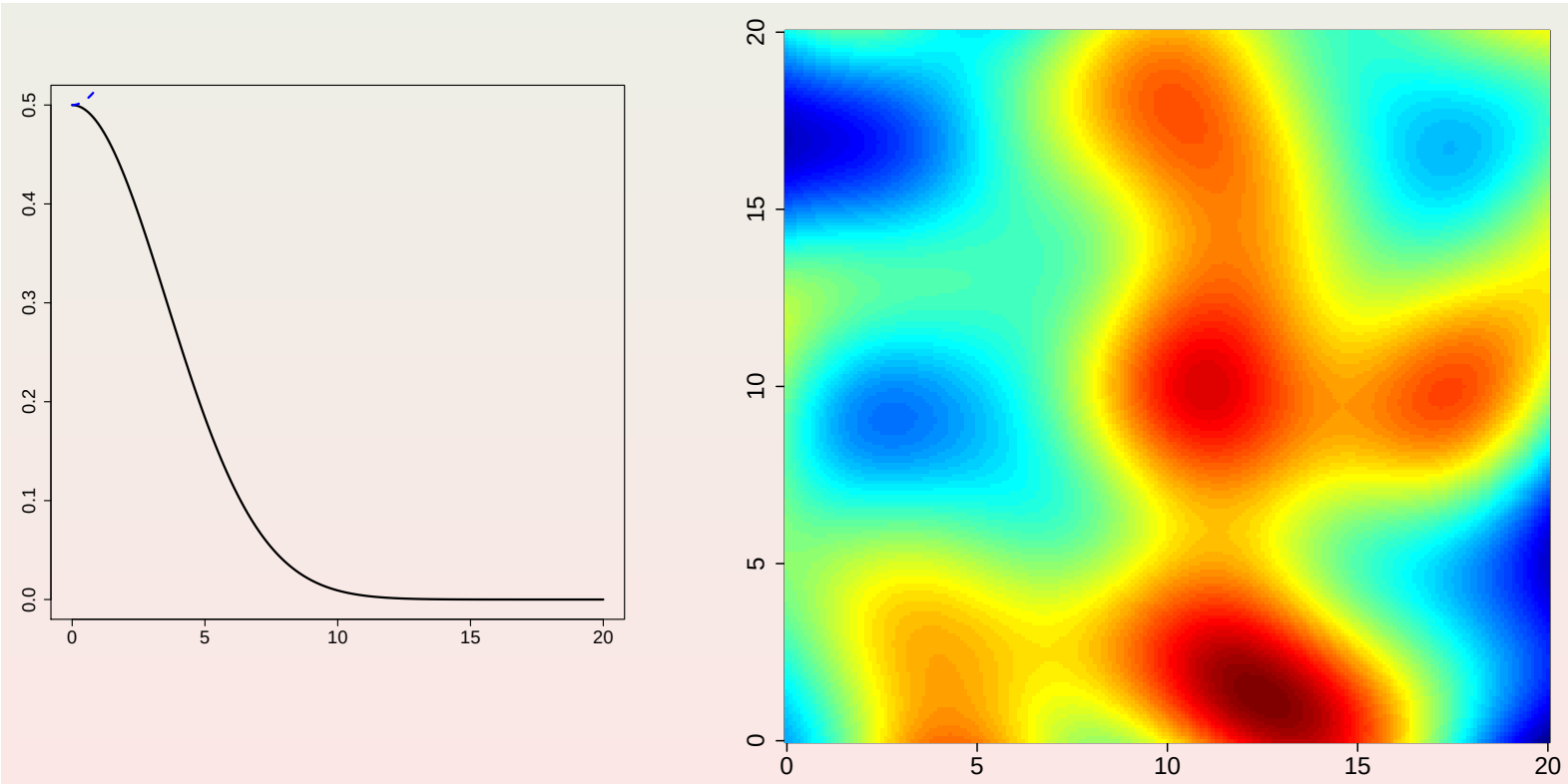
Champ aléatoire

Modèle de covariance gaussien (plus lisse)



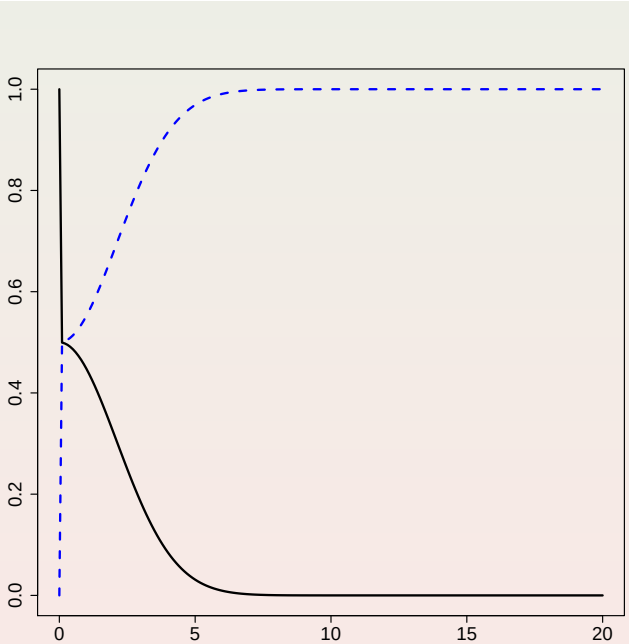
Champ aléatoire

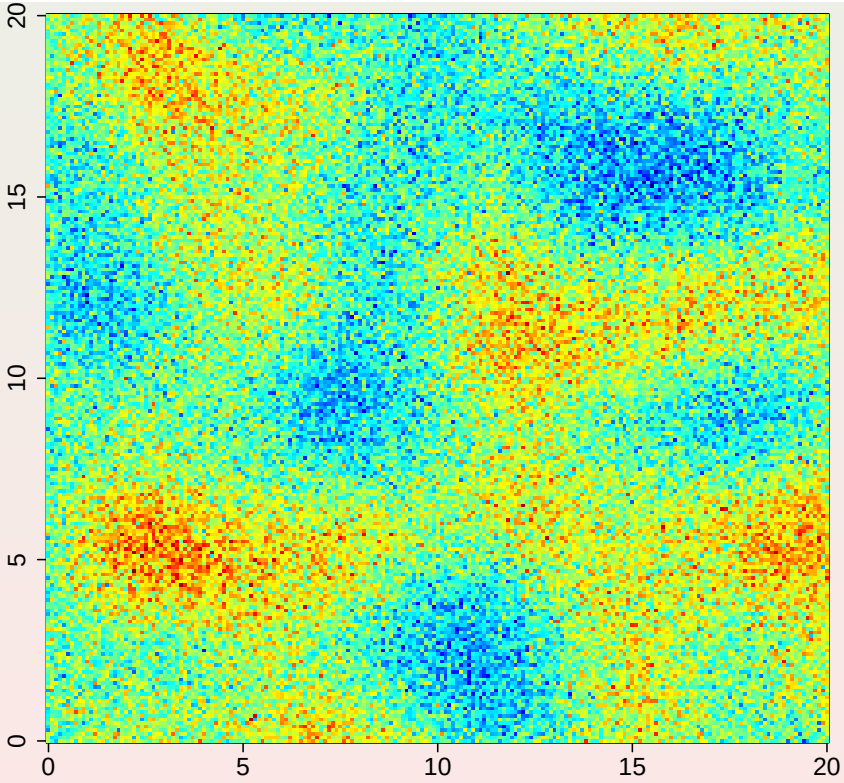
Modèle de covariance gaussien avec une plus grande portée



Champ aléatoire

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.

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

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.

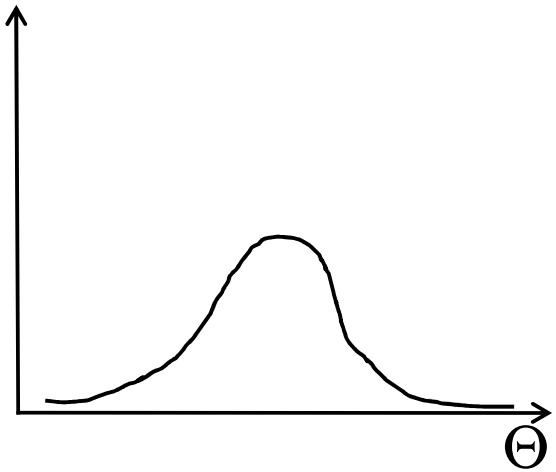
$\Rightarrow$ on cherche donc à caractériser la distribution *a posteriori* des paramètres, $[\theta|\mathrm{y}]$, à l'aide de la formule de Bayes.

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

Approche bayésienne : principe général

Connaissance a priori

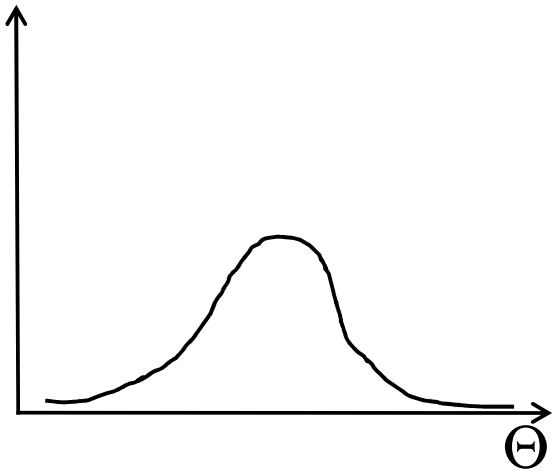
$[\Theta]$



Approche bayésienne : principe général

Connaissance a priori

$$[\Theta]$$



Données
+
Vraisemblance

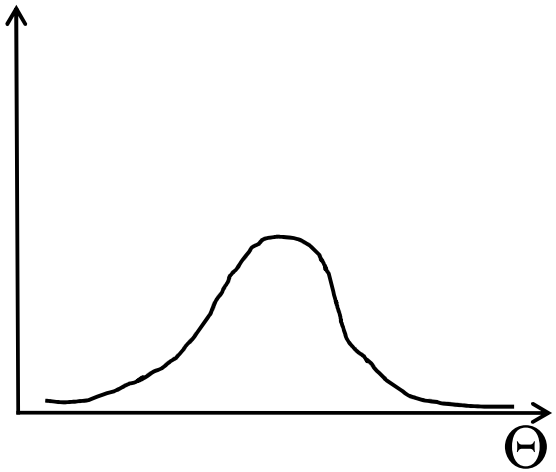
$$[\mathbf{y}|\Theta]$$

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

Approche bayésienne : principe général

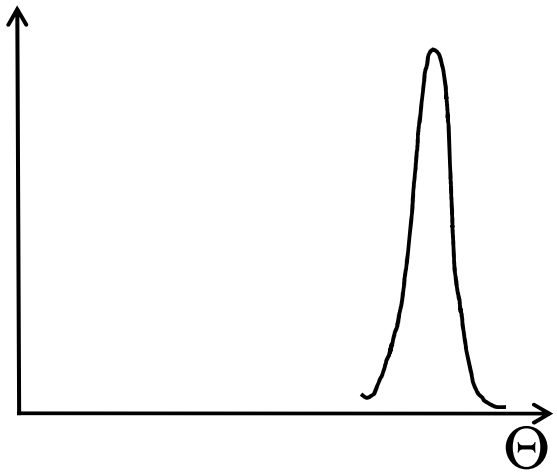
Connaissance a priori

$[\Theta]$



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

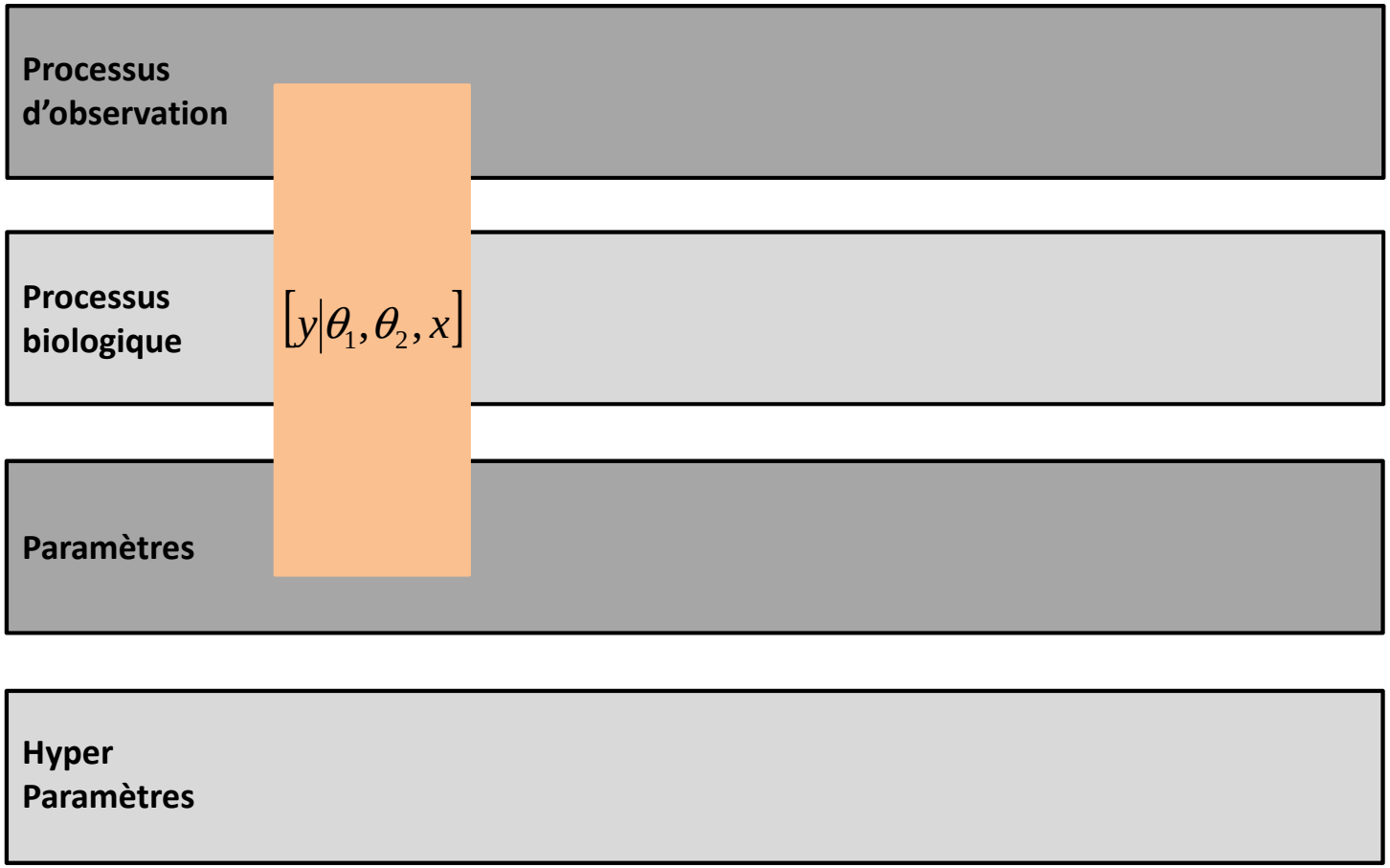
Processus biologique

Paramètres

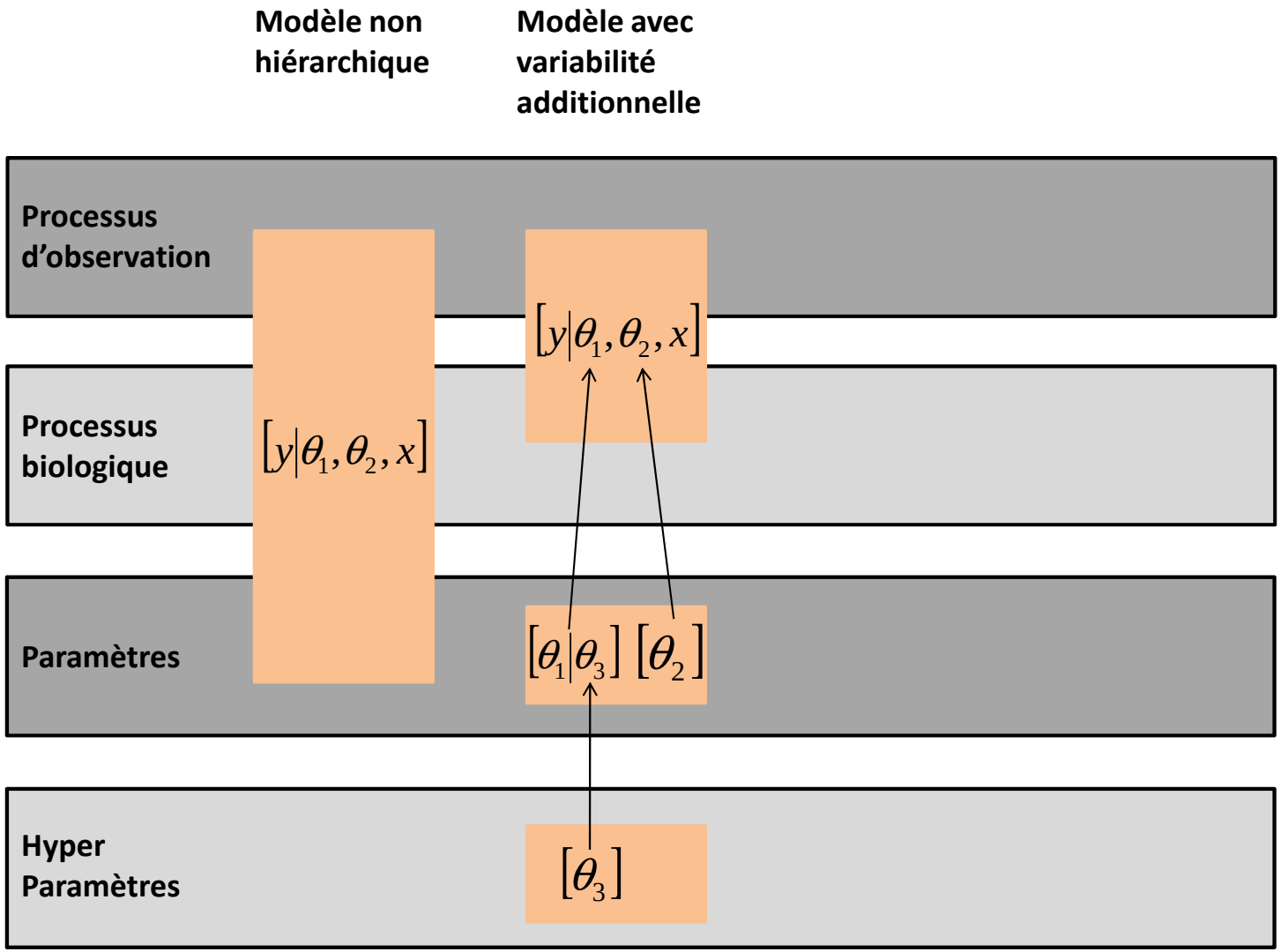
Hyper Paramètres

Exemple non hiérarchique

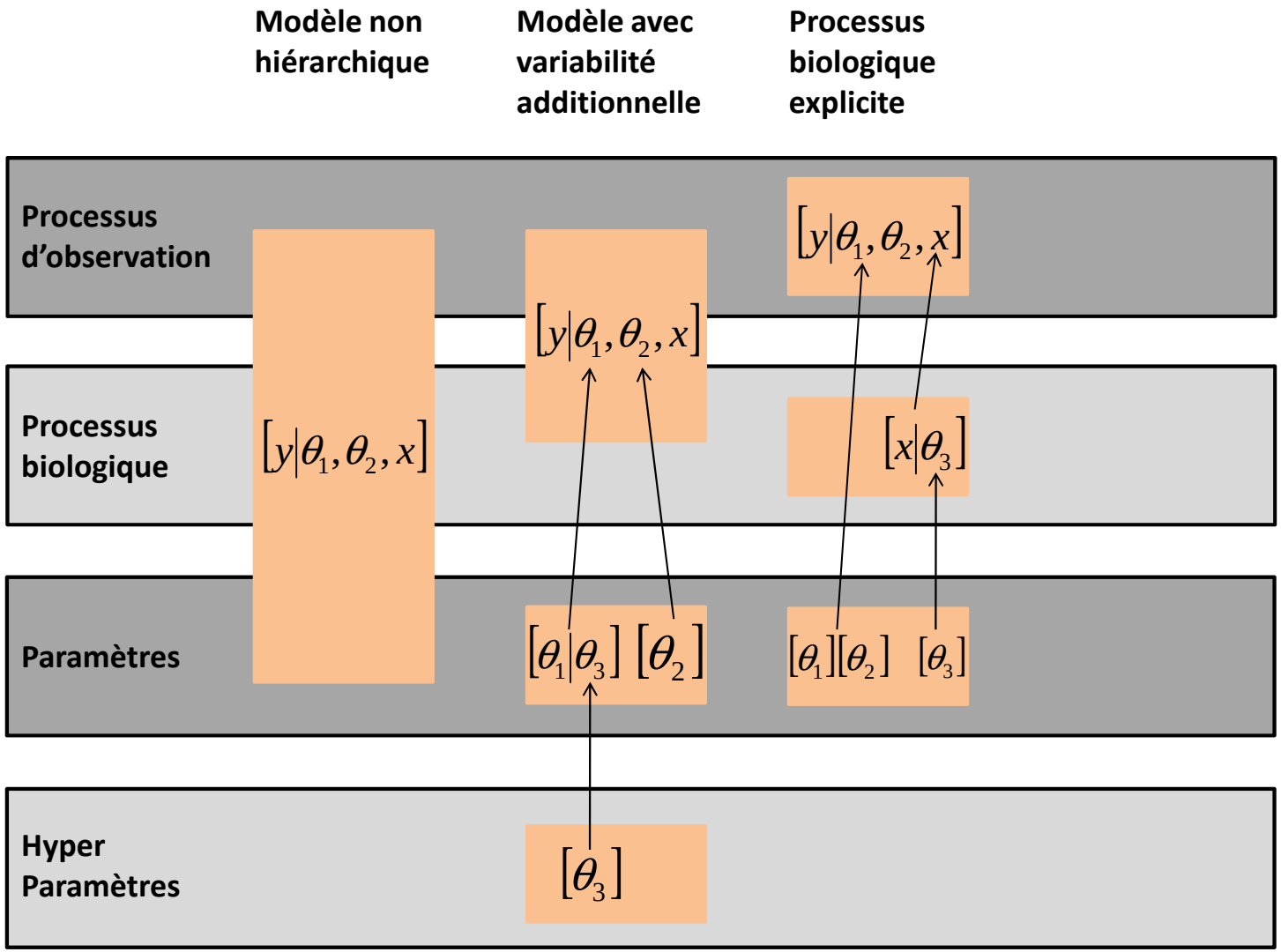
Modèle non hiérarchique



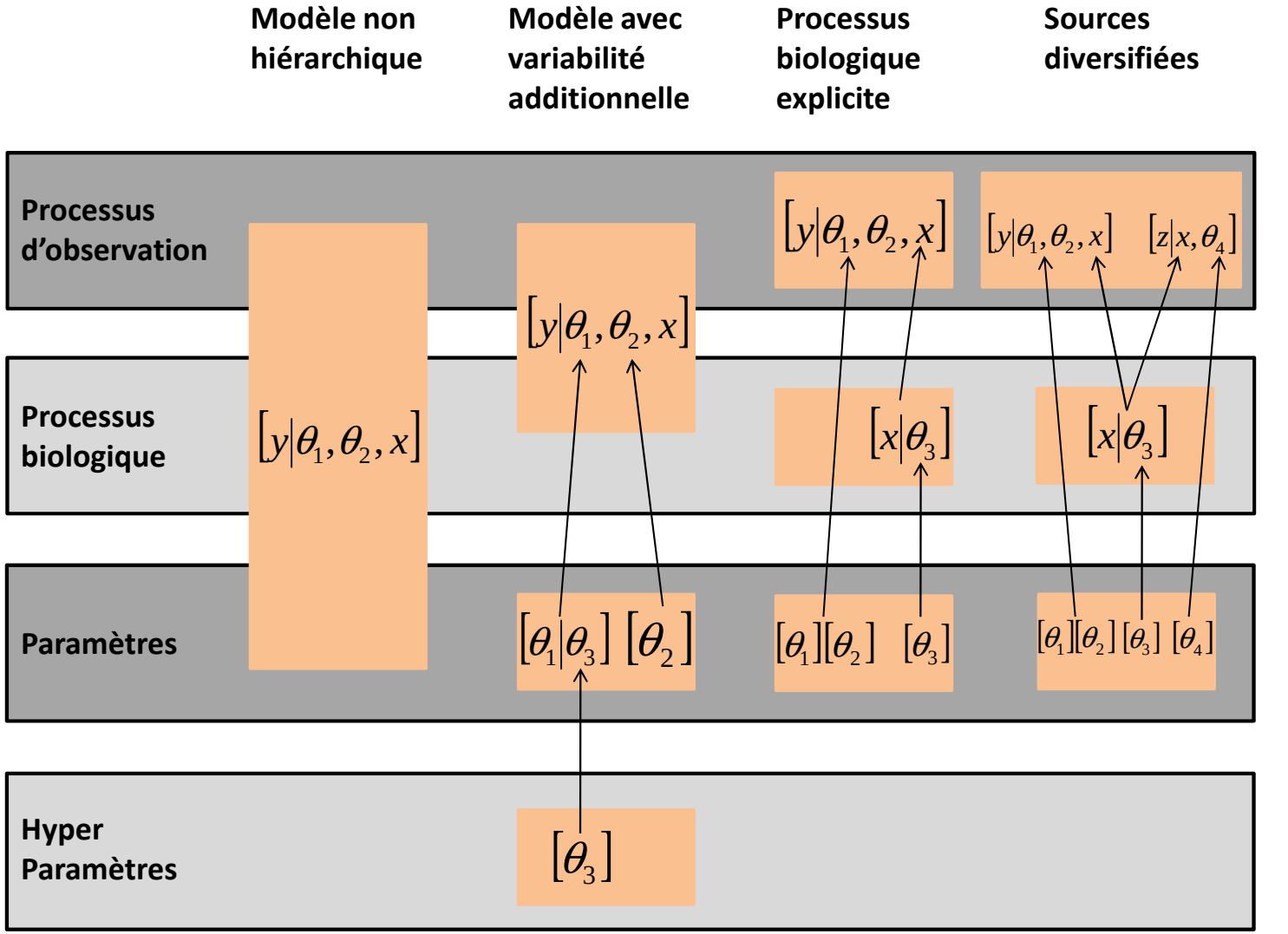
Un premier cas hiérarchique : la variabilité individuelle



Modèle avec processus identifié



Intégration de différentes sources de données



D'après Clark, 2005

Les modèles linéaires généralisés mixtes spatialisés

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

Les modèles linéaires généralisés mixtes spatialisés

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

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

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

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

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

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

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

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

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

De ce fait on a :

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

Les modèles linéaires généralisés mixtes spatialisés

En résumé

Modèle linéaire généralisé mixte spatial :

$$
\begin{cases}
\textbf{Hyper - a priori} \\
\theta_3, \\
\textbf{A priori} \\
\mathrm{a} \sim \mathcal{MVN}(0, \Sigma), \text{avec } \Sigma_{i,j} = C(\theta_2, \text{dist}(a_i, a_j)) \\
\theta_1, \theta_2, \\
\textbf{Espérance} \\
\mathrm{E}[y_i | a_i] = g(\beta_0 + \beta_1 x_i^{(1)} + \beta_2 x_i^{(2)} + \ldots + a_i), \\
\textbf{Vraisemblance} \\
y_i \sim \mathcal{L}(\theta_1).
\end{cases}
$$

Modèles statistiques de distribution d'espèces

– Des positions d'individus vers les différents types de modèle –

Thomas Opitz

Atelier

Modélisation spatio-temporelle en écologie avec INLA

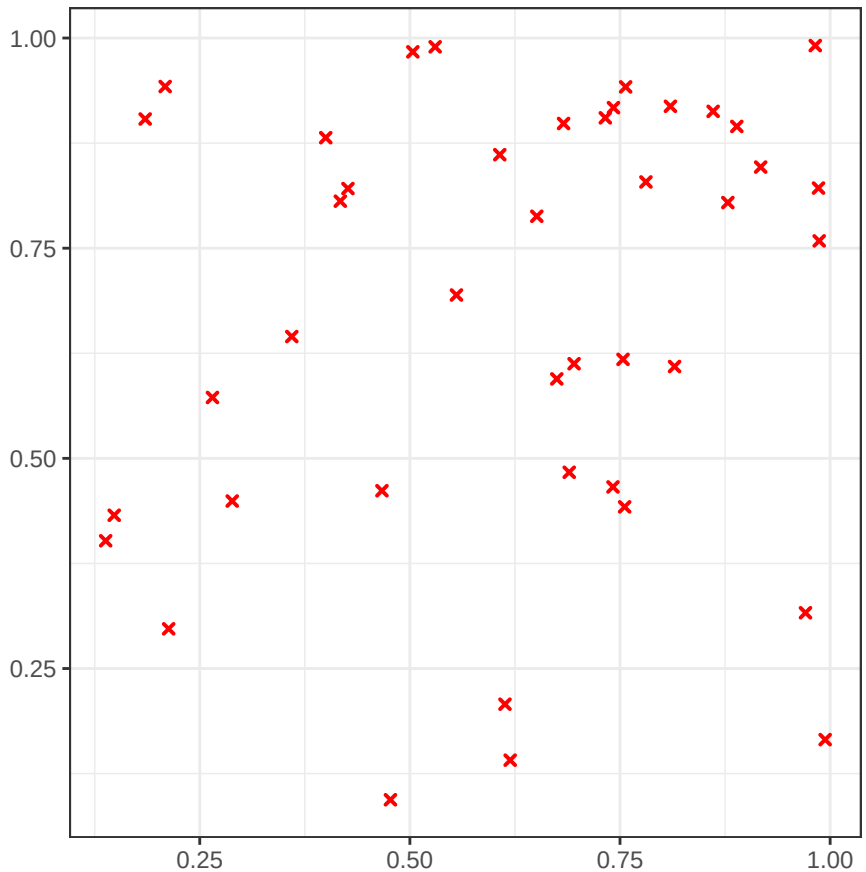
Avignon, 11-12 mars 2025

Processus ponctuel spatial

- Un ensemble de N **points aléatoires** : $\{s_1, \ldots, s_N\}$ (N est aléatoire aussi)
- Pour représenter les **positions d'objets, d'événements ou d'individus**
- Par exemple : données de présence seule (*Presence-Only*) en écologie
- Information la plus complète possible (non agrégée spatialement)

Autres termes : *Semis de point*, *pattern de points*

Exemple : semis de points simulé dans la fenêtre d'observation $D = [0, 1]^2$

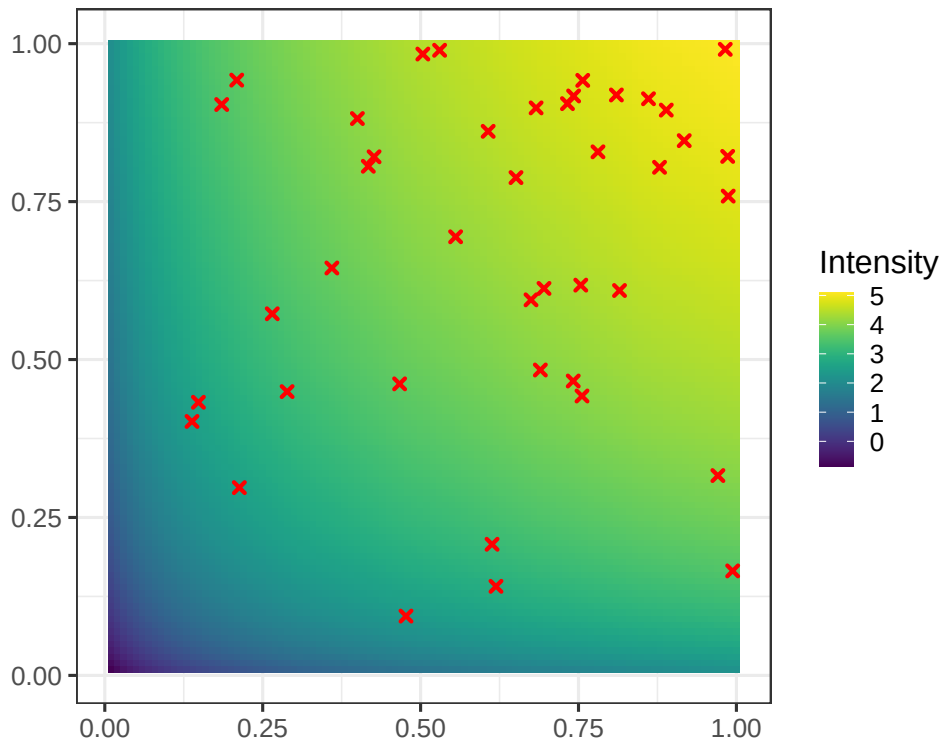


Fonction d'intensité et PP de Poisson

- La **fonction d'intensité** $\lambda(s)$, une caractéristique clé (*intensité* = *taux d'occurrence*)
- **Nombre de points attendu** dans un domaine A = Intensité agrégée sur A:

$$\lambda(A) := \mathbb{E}[N(A)] = \int_A \lambda(s)\mathrm{d}s$$

- Modèle fondamental : **processus de Poisson** (homogène si $\lambda(s) = \lambda_0$)
  $\Rightarrow$ Les événements se réalisent de façon indépendante



Modèle de régression pour l'intensité

Modèle additif généralisée (GAM) :
fonction de lien log pour relier l'intensité λ à un **prédicteur linéaire** $\eta(s)$:

$$\log \lambda(s) = \beta_0 + \sum_{k=1}^{K} \beta_k z_k(s) = \boldsymbol{\beta}^T \boldsymbol{z}(s) =: \eta(s)$$

- **Prédicteurs** $z_k(s)$, déterministes et connus :
  - covariables "physiques" (effets linéaires/fixes)
  - fonctions de base pour représenter un effet nonlinéaire/aléatoire (p. ex. fonctions splines, éléments finis pour un effet spatial)
- **Coefficients** β_k à estimer
- β_k stochastique $\Rightarrow$ **Effet stochastique**
- K potentiellement très grand (p. ex. champ spatial)
  $\Rightarrow$ **Pénalité de smoothness** (fréquentiste),
  ou **loi a priori gaussienne corrélée** (bayésien)
  $\Rightarrow$ Les deux approches rajoutent $-\boldsymbol{\beta}^T Q \boldsymbol{\beta}$ à la log-vraisemblance !

Densité de probabilité du **processus de Poisson** observé sur un domaine D:

$$\exp\left(-\int_D \lambda(s)\mathrm{d}s\right)\prod_{i=1}^{N}\lambda(s_i)$$

⇒ **Log-vraisemblance:** $\ell(\beta) = -\int_D \exp(\eta(s))\mathrm{d}s + \sum_{i=1}^{N}\eta(s_i)$

Intégration numérique : $\int_D \exp(\eta(s))\mathrm{d}s \approx \sum_{j=1}^{m}\omega_j \exp(\eta(\tilde{s}_j))$

- Schéma de discrétisation avec m points $\tilde{s}_j$
- Poids $\omega_j > 0$, et typiquement $\sum_{j=1}^{m}\omega_j = |D|$

⇒ **Calcul d'une log-vraisemblance approximative :**
Définissons $s_{N+j} = \tilde{s}_j$, $j = 1, \ldots, m$

- $y_j = 1$ et $\omega_j = 0$ pour $j = 1, \ldots, N$ (points observés)
- $y_j = 0$ pour $j = m+1, \ldots N+m$ (points de fond)

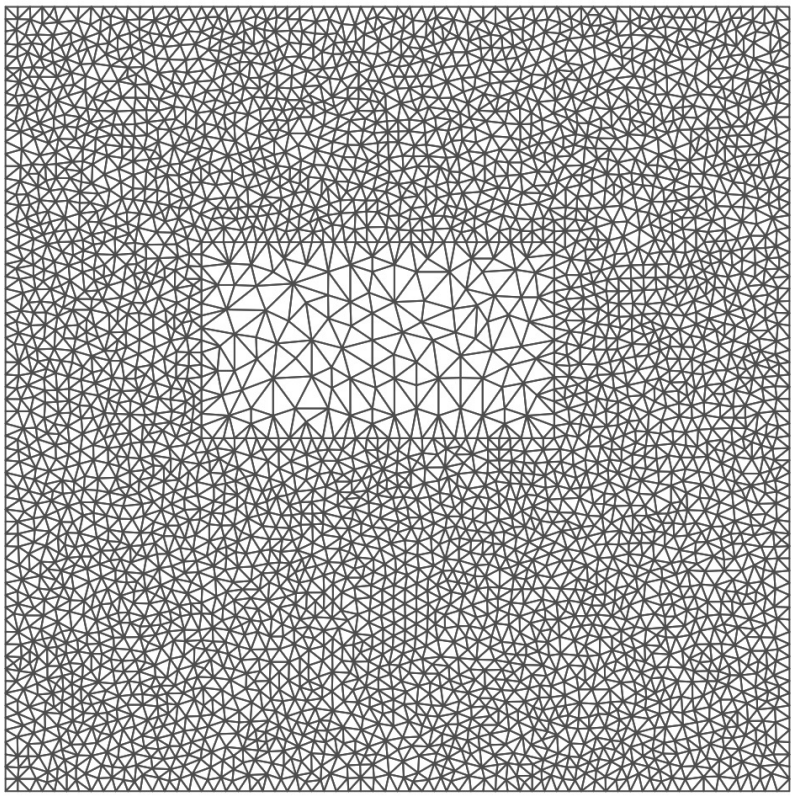
$$\ell(\beta) \approx \sum_{j=1}^{N+m}\left(y_j\eta(s_j) - \omega_j \exp(\eta(s_j))\right)$$

⇒ Formellement, une log-vraisemblance de Poisson avec observations y_j, moyennes $\omega_j \exp(\eta(s_j))$ et expositions ω_j
(INLA : `inla(y ~ ..., E = omega, family = "poisson")`)

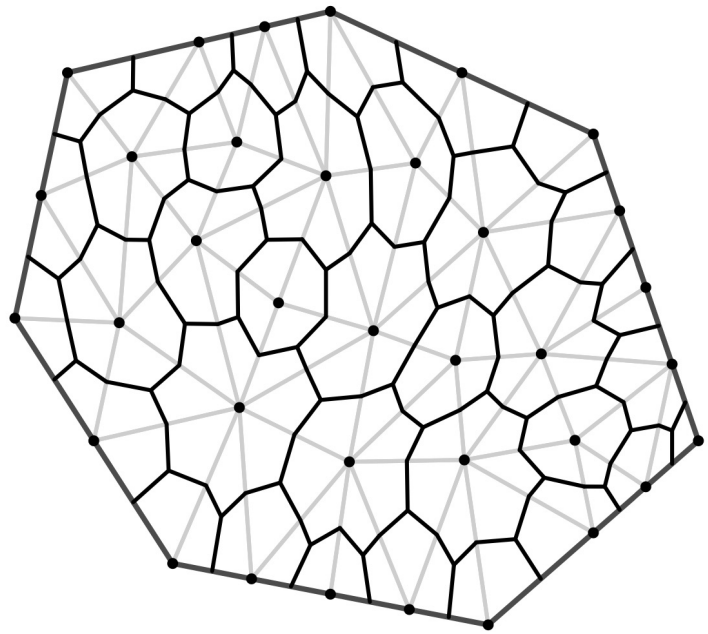
Exemple de discrétisation

- Discrétisation par **éléments finis** (triangulation) des approches "SPDE"
- Basée sur une triangulation (**mesh**) de l'espace
- Poids ω_j = Aires des cellules de la mesh duale

Exemple de mesh



Exemple de mesh duale



Simpson, D., Illian, J. B., Lindgren, F., Sørbye, S. H., Rue, H. (2016, Biometrika). Going off grid: Computationally efficient inference for log-Gaussian Cox processes.

Variante (Berman-Turner)

Idée : poids $\omega_j > 0$ pour les points observés $\{s_i\}$ dans le calcul de l'intégral

Log-vraisemblance approximative :

$$\ell(\beta) \approx \sum_{j=1}^{m+N} \omega_j \left(\frac{y_j}{\omega_j} \eta(s_k) - \exp(\eta(s_j)) \right)$$

- **Loi de Poisson** avec pseudo-observations y_i/ω_j
- Astucieux car fonctionnant avec les implémentations standard des modèles de régression sous `R` (`glm`, `gam`...)
- Implémentation par défaut dans `spatstat`
- ⚠ Pas pratique avec des gros jeux de données ($N \gg 0$)

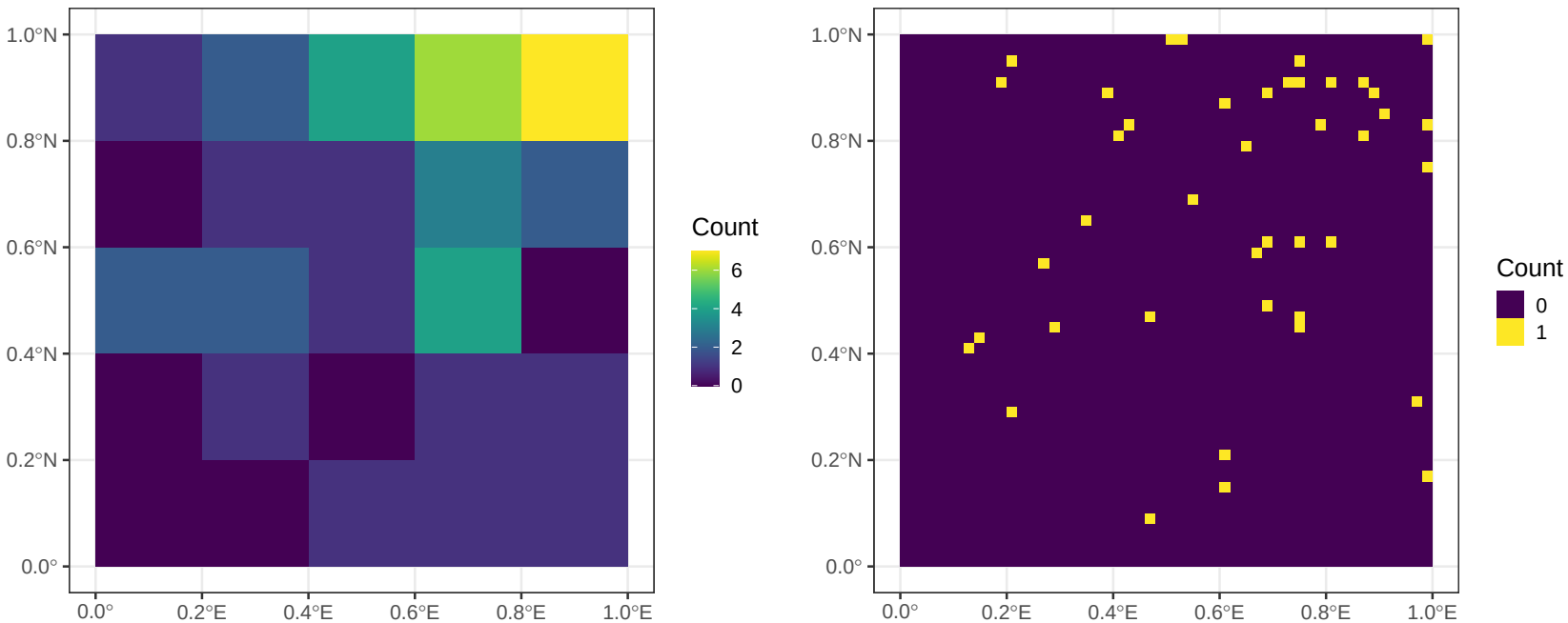
Berman, Turner (1993, JRSS-C). Approximating Point Process Likelihoods with GLIM.
Baddeley, Turner (1998, AAP). Practical maximum pseudolikelihood for spatial point patterns.

Approche "PLN" (Poisson log-linéaire)

Régression de Poisson log-linéaire :

- Diviser D en petites cellules C_j, $j = 1, \ldots, m$ (p. ex. pixels)
- Pour chaque cellule, compter les points : $N_j = \sum_{i=1}^{N} \mathbb{I}(s_i \in C_j)$
- Alors $N_j \sim \text{Poisson}\left(\int_{C_j} \lambda(s)\mathrm{d}s\right)$
- En pratique, on suppose $N_j \sim \text{Poisson}(|C_j| \times \lambda(\tilde{s}_j))$ avec un point représentatif $\tilde{s}_j$ de C_j (typiquement le barycentre)
- **Log-vraisemblance de Poisson :** $\ell(\beta) = \sum_{j=1}^{m} \left(-|C_j|\lambda(\tilde{s}_j) + N_j \log \lambda(\tilde{s}_j)\right)$

Données de comptage pour deux résolutions spatiales



Approche "PA" (Présence-Absence)

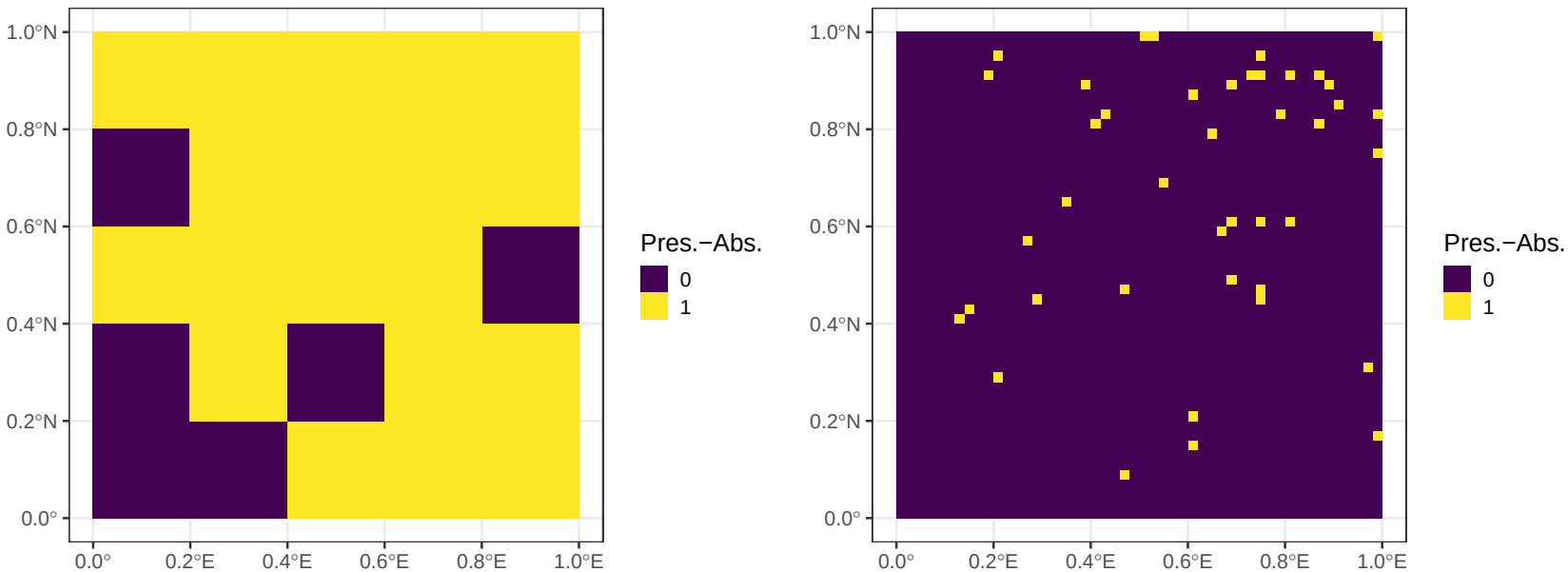
Régression binaire avec lien clogog pour les Présences-Absences :

- Discrétisation spatiale comme avant
- **Lien "log-log complémentaire"** venant de la loi de Poisson :

$$p_j = \Pr(N_j > 0) = 1 - \exp(-\exp(\eta(s))), \quad \eta(s) = \log(-\log(1 - p_j)))$$

- Utile si pas de comptages précis disponibles

Données de présence-absence pour deux résolutions spatiales



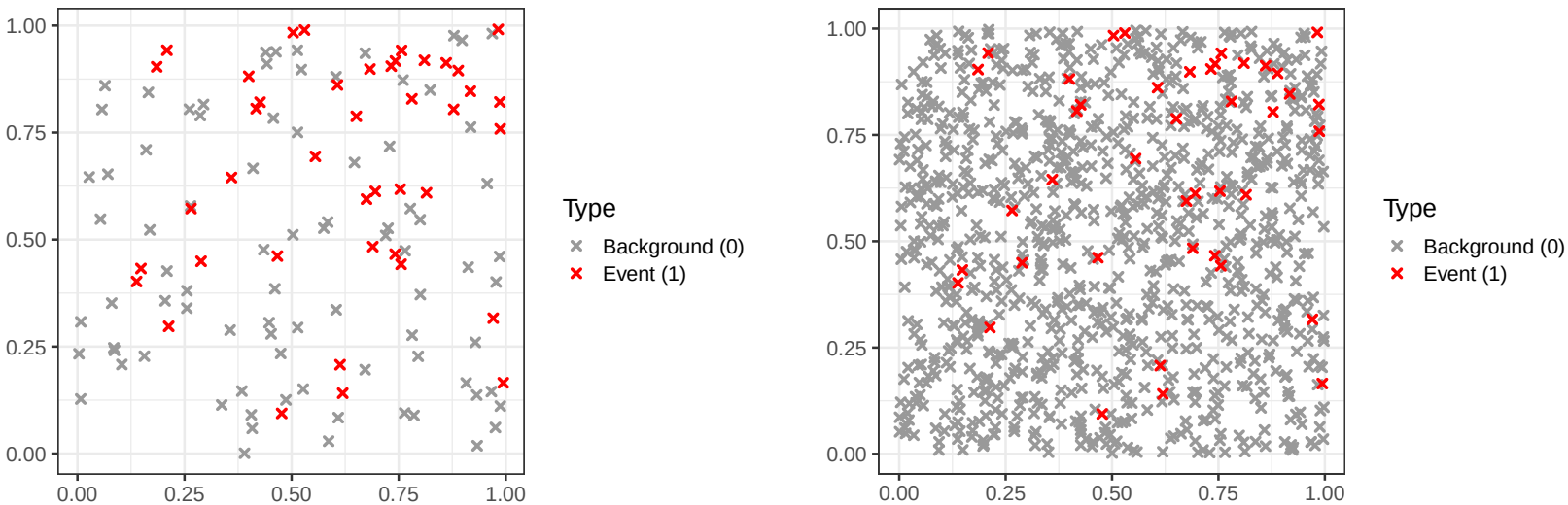
Approche "Classification"

Régression logistique pour la classification des points :

- Définir m points de fond selon une intensité $\rho(s)$ (p. ex. $\rho(s) = m/|D|$)
- Pour $s_j \in \{\text{points observes}, \text{points de fond}\}$,
  définir $y_j = 1$ si observé et $y_j = 0$ si fond
  $\Rightarrow p_j = P(y_j = 1) = \lambda(s_j)/(\lambda(s_j) + \rho(s_j))$
  $\Rightarrow \log \frac{p_j}{1-p_j} = -\log \rho(s_j) + \eta(s_j)$
  $\Rightarrow$ Régression logistique avec offset $-\log \rho(s)$

$\Rightarrow$ **Log-vraisemblance :** $\ell(\beta) = \sum_{y_j=1} \log p_j + \sum_{y_j=0} \log(1 - p_j)$

Classification avec peu/beaucoup de points de fond



Variante : régression logistique *infinitely weighted*

- Constat : bias d'estimation si prédicteur $\eta(s)$ pas bien spécifié et $m \not\gg N$
- Augmenter m ($m \gg N$)? ⚠ Coût de calcul augmente avec m !
  $\Rightarrow$ Mieux : m modérément large, et poids "infini" pour les points de fond
  $\Rightarrow \omega \gg 1$ si $s_j \in \{\text{points de fond}\}$, p. ex. $\omega = 10,000$

$\Rightarrow$ **Log-vraisemblance "infinitely weighted" :**

$$\ell(\beta) = \sum_{y_j=1} \log p_j + \omega \sum_{y_j=0} \log(1 - p_j)$$

(argument `weights` dans les fonctions `R` comme `glm`, `gam`, `inla`...)

Fithian, Hastie (2012, AOAS). Finite-sample equivalence in statistical models for presence-only data.

Comparaison des approches

- Approches “PP” et “Classification” nécessitent des **points de fond** (*dummy points*, *background points*, *control points*)

- Approches “PLN” et “PA” nécessitent une **discrétisation de l’espace**

- Quid de la gestion des covariables ?
  - Approches “PLN”, “PA” :
    agrégation des covariables par cellule est nécessaire
    (*Rightarrow* souvent moyennes ou proportions par cellule)
  - Approches “PP”, “Classification” :
    agrégation des covariables pas nécessaire
    - On a besoin des valeurs des covariables aux points de fond
    - ⚠ Attention si les points d’événements se concentrent sur des covariables “rares”

Comparaison asymptotique – Euréka !

- Laissons m représenter le nombre de cellules ou de points de fond
- Placement/tirage uniforme des cellules ou points de fond dans D
  $\Rightarrow$ En moyenne, une cellule ou un point couvre un surface $|D|/m$

Comportement asymptotique pour toutes les approches discutées :

$$\ell(\beta) \sim -\int_D \lambda(s)\mathrm{d}s + \sum_{i=1}^{N} \eta(s_i) \quad \text{lorsque } m \to \infty$$

= Log-vraisemblance exacte du PPP !

En pratique :

- Idéalement, m très grand...
- ⚠ Nombre de réponses du modèle de régression $\sim m$
  $\Rightarrow$ Côut de calcul numérique augmente avec m
  $\Rightarrow$ Il faut trouver un**compromis** !
- Besoin de stratégies pour "bien placer" les cellules ou points de fond :
  - Triangulation et mesh duale (implémentations SPDE)
  - Stratification des points de fond selon les cellules d'une grille
    Baddeley et al (2014, Biometrika). Logistic regression for spatial Gibbs point processes
  - Suites de points à discrépance faible (Sobol, Halton...) – Quasi-Monte-Carlo

Processus de Cox log-gaussien

Fonction d'intensité log-gaussienne :
$\Rightarrow$ Variables gaussiennes (corrélées ou non) pour les coefficients β_k :

$$\log \lambda(s) = \beta_0 + \sum_{k=1}^{K} \beta_k z_k(s) = \boldsymbol{\beta}^T \boldsymbol{z}(s) =: \eta(s), \quad \boldsymbol{\beta} \sim \mathcal{N}(\mathbf{0}, \Sigma)$$

Par exemple, champ gaussien $W(s)$ selon une fonction de covariance :
$\log \lambda(s) = \beta_0 + W(s)$

Utilités et interprétations :

- **Effet aléatoire :** stochasticité represente variabilité environnementale non captée par les covariables
- **Interprétation bayésienne :** loi a priori gaussienne
- 'Régularisation" et smoothness des effets si beaucoup de coefficients β_k

Implémentation : GAM(M)s (fréquentiste) ; INLA (bayésien)

Toutes les approches sont possibles — on estime toujours $\log \lambda(s)$ **!**

Fonction d'intensité modulée

Processus ponctuel observé selon l'intensité suivante :

$$\lambda_{OBS}(s) = \lambda(s) \times e(s), \quad e(s) \geq 0$$

- Terme $e(s)$ représentant une modulation de l'intensité $\lambda(s)$
  - **Exposition** (p. ex. : maladies ; surface forestière pour les incendies de forêt)
  - **Effort d'échantillonnage** $\Rightarrow$ Différents cas de figure :
    - Connu exactement (le rêve !)
    - Fortement déterminé par une covariable (e.g. distance sampling)
    - Essentiellement inconnu (Données opportunistes)
  - Cas particulier du **thinning**
    - $e(s) \leq 1$
    - On retient un point d'un PP d'intensité $\lambda(s)$ avec probabilité $e(s)$

Groupe cible pour présences-seules opportunistes

Approche par *Target-group background* :

- Espèce focale avec observations selon $\lambda_{j_0}^{OBS}(s) = \lambda_{j_0}(s)e(s)$
- Groupe cible d'espèces (y compris j_0) avec observations $\lambda_{TG}^{OBS}(s)$

Hypothèse de modélisation : $\lambda_{TG}^{OBS}(s) = \lambda_{TG}(s)e(s)$
($e(s)$ est commun à toutes les espèces)

$$\lambda_{j_0}^{OBS}(s) = \lambda_{TG}^{OBS}(s) \times \tilde{\lambda}_{j_0}(s)$$

$\Rightarrow$ On peut estimer l'**intensité relative** $\tilde{\lambda}_{j_0}(s) = \lambda_{j_0}(s)/\lambda_{TG}(s)$

Approches d'estimation :

- PLN : approcher $\lambda_{TG}^{OBS}(s)$ par comptages empiriques $\Rightarrow$ Offset $\log \lambda_{TG}^{OBS}(s)$
- Estimer $\hat{\lambda}_{TG}^{OBS}(s)$ avec un modèle
  - Variante 1 : en deux étapes
    1. Modèle univarié pour $\hat{\lambda}_{TG}^{OBS}(s)$
    2. Modèle pour $\lambda_{j_0}^{OBS}(s)$ avec offset $\log \hat{\lambda}_{TG}^{OBS}(s)$
  - Modèle bivarié pour $\lambda_{j_0}^{OBS}(s)$ et $\lambda_{TG}^{OBS}(s)$ avec champ partagé

Quelques messages-clés à retenir

- **Processus ponctuel** = représentation la plus résolue et informative possible

- Quel intérêt pour discrétiser le domaine et agréger les données ?
  - **Erreurs de saisie :**
    Si les positions des points sont imprécises
  - **Surdispersion**
    S'il y a forte multiplicité de points $\Rightarrow$ Présence-Absence
    (p. ex. observation d'espèces gregaires)
  - **Côut numérique**
    S'il a un très grand nombre de points

Toutes les représentations numériques approximent le processus ponctuel !

Rapide état des lieux des packages R reliant SDM et modèle SPDE Matern

Pascal Monestiez
CiSStats, BioSP, FaCiDoSi
11 mars 2025, Avignon

Remarques préliminaires

- Cet état des lieux n’a pas la prétention d’être exhaustif.
- Ne pas hésiter à partager vos connaissances au sujet d’autres packages non mentionnés ici.
- Cette présentation ne va probablement plus être valable dans quelques semaines ou mois : packages remplacés par d’autres, nouveaux packages produits chaque semaine … et nouvelles approches méthodologiques.

Limites de cette présentation

- On s'est limité aux applications de type SDM – Species Distribution Models – en écologie à partir de données spatiales (et spatio-temporelles) de diverses sortes (protocolées, opportunistes, présence seule, présence absence, comptages, Distance Sampling, multi-sources, multi-espèces, etc…).
- On s'est centré sur les modélisations de type Modèles mixtes généralisés intégrant des effets fixes (GLM, GAM) et une partie Géostatistique (effet aléatoire, GLMM) modélisée par l'approche SPDE et une covariance de Matern.
- L'inférence sera le plus souvent menée dans un cadre bayésien avec comme base SPDE-INLA

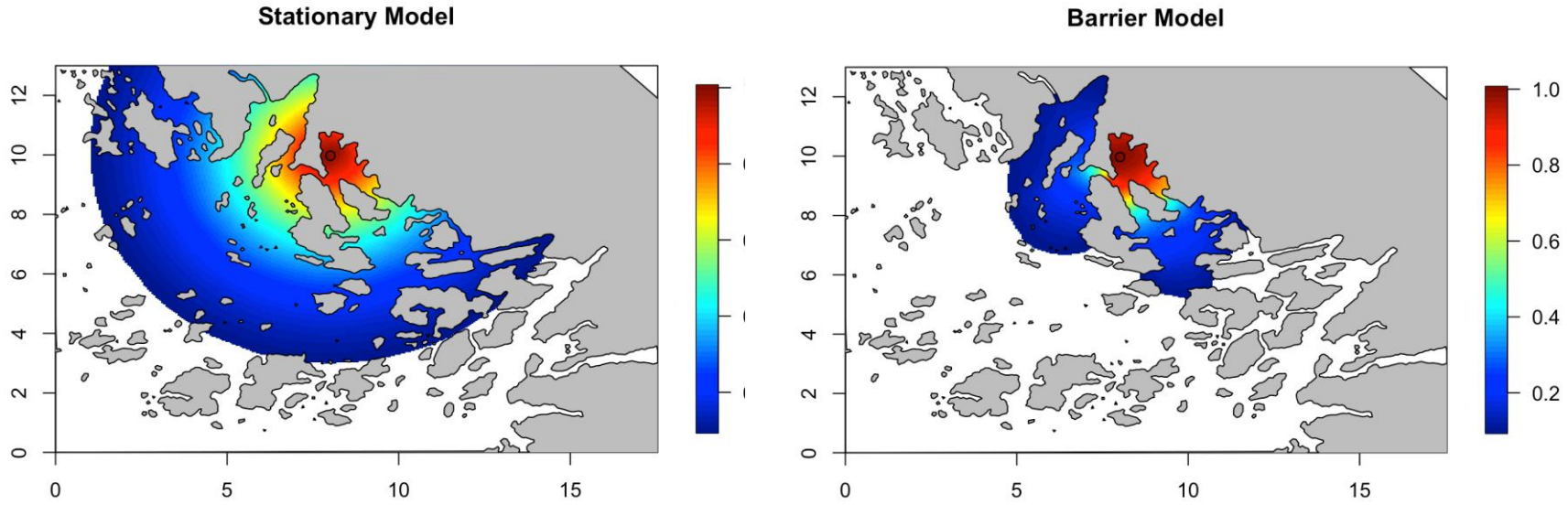
SPDE INLA (package INLA)

- Le package INLA pourrait être suffisant dans la très grande majorité des cas.
- Résolution de l'espace très efficace via les meshs (paramétrables)
- Définitions des modèles et tuning des paramètres très complets
- Très grande efficacité computationnelle
- Abord un peu difficile, bien que rendu plus accessible via la définition des « inla stacks » dans les cas multi espèces ou multi dates
- refs : INLA books, site officiel https://www.r-inla.org/

  https://www.r-inla.org/learnmore/books

SPDE INLA (package INLA)

- Certaines applications nécessitent le passage par les construction de « meshs » avec INLA , tels que les Barrier Models (covariances spatiales non stationnaires)



- Beaucoup de tutorials disponibles + livre très complet sur SPDE INLA
- Liens : https://haakonbakkagit.github.io/alltopics.html
- https://becarioprecario.bitbucket.io/spde-gitbook/

Directement liés à SPDE INLA (package inlabru)

- Inlabru package plus à destination des écologues statisticiens avec une orientation sur les SDM construits sur l'approche PP (Processus ponctuels)
- Deux fonctions principales : bru() et lgcp()
- Syntaxe plus simple pour bru() mais possibilité de faire appel à des fonctions de INLA directement (compatibilité)

Bachl FE, Lindgren F, Borchers DL, Illian JB. `inlabru`: an R package for Bayesian spatial modelling from ecological survey data. *Methods Ecol Evol*. 2019; 10: 760–766. https://doi.org/10.1111/2041-210X.13168

Directement liés à SPDE INLA (package inlabru)

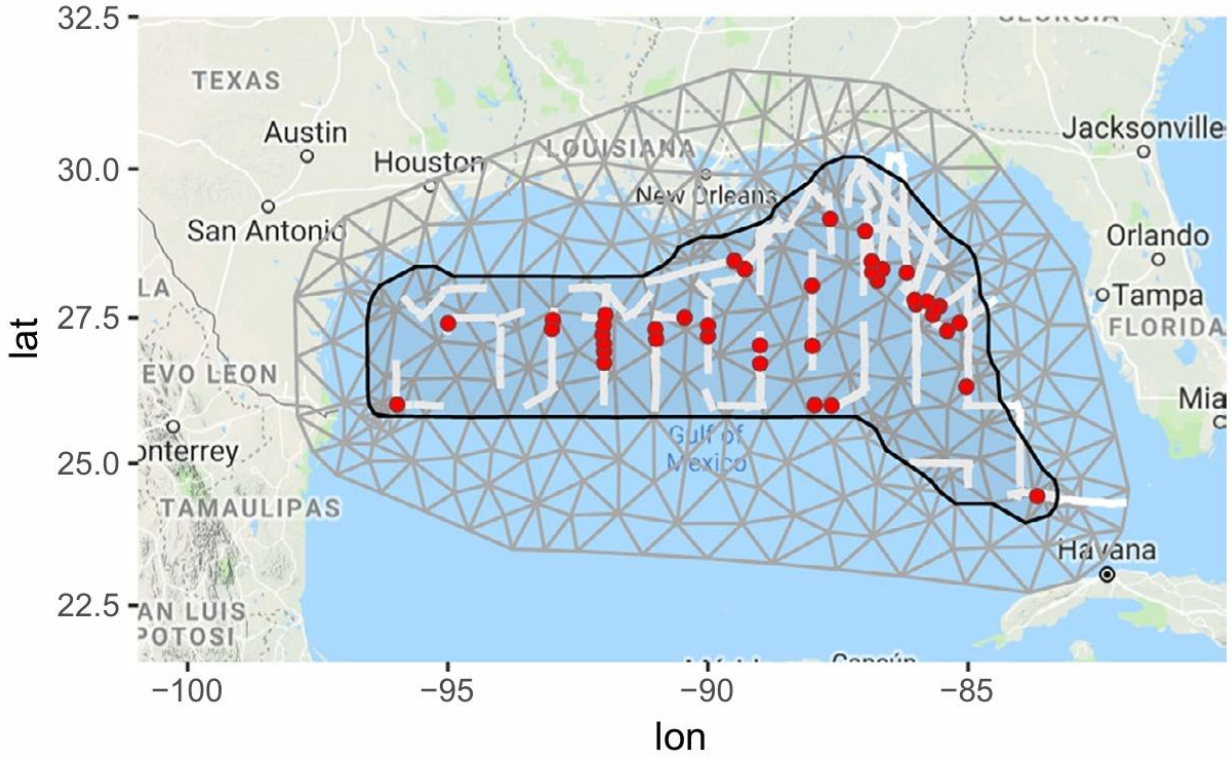
- Fonction spécifique lgcp() pour les Log gaussian Cox Process s'appuyant sur des données d'observations ponctuelles (espèce,x,y)
- Abondances avec predict() et fm_int()
- Fonction gg() spécifique pour les cartes et graphiques sur les objets bru et interopérabilité avec ggplot2.
- Nombreux tutorials. Dans les mises à jour, inlabru a pu être momentanément en retard et pas toujours compatible (cas par exemple du passage de inla.mesh au package fmesher)

https://sites.google.com/inlabru.org/inlabru/home

https://inlabru-org.github.io/inlabru/articles/2d_lgcp.html

Directement liés à SPDE INLA (package inlabru)

- Inlabru a été développé sur des applications particulières du type Distance Sampling (inférence conjointe du lgcp et de la fonction de détection, spécifique inlabru car inclusion d'un seuil)



Point process models for spatio-temporal distance sampling data. Y. Yuan, F. E. Bachl, F. Lindgren, D. L. Brochers, J. B. Illian, S. T. Buckland, H. Rue, T. Gerrodette. arXiv:1604.06013

Directement liés à SPDE INLA (package fmesher)

- fmesher : package permettant la construction et le paramétrage des « meshs » (triangulations en éléments finis)
- Toutes les possibilités des fonctions inla.mesh.2d, avec la compatibilité sp et sf
- Interopérabilité avec des objets géographiques sf et terra, et graphiquement avec les « geom » de ggplot2

Directement liés à SPDE INLA (package excursion)

- excursion : package exploitant les sorties de SPDE INLA
- Fonction principale : calcule des ensembles d'excursions (zones supérieures ou inférieures à un seuil), les intervalles de crédibilité de ces régions

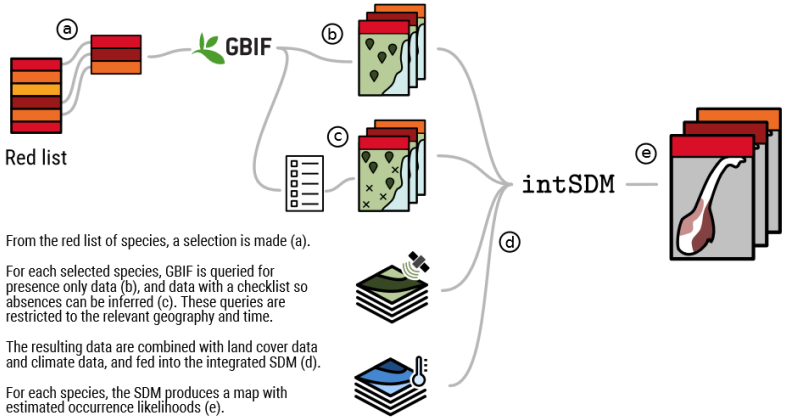
Bolin, D., & Lindgren, F. (2018). Calculating Probabilistic Excursion Sets and Related Quantities Using excursions. Journal of Statistical Software, 86(5), 1–20. https://doi.org/10.18637/jss.v086.i05

Directement liés à SPDE INLA (package INLAspacetime) espace et temps continus

- INLAspacetime : package permettant de définir des modèles spatio-temporels plus généraux pour ensuite les implémenter dans INLA
- Extension spatio-temporelle des Gaussian Matern Fields, paramétrage des dépendance spatiales (variance, smoothness, correlation range) et dans le temps (smoothness, correlation range) ainsi que le type de non séparabilité.
- Extension à des supports courbés (manifolds, process sur la sphère par exemple)
- https://eliaskrainski.github.io/INLAspacetime/
- *A diffusion-based spatio-temporal extension of Gaussian Matérn fields* (2024). *Finn Lindgren, Haakon Bakka, David Bolin, Elias Krainski and Håvard Rue*. SORT 48 (1) January-June 2024, 3-66. https://www.idescat.cat/sort/sort481/48.1.1.Lindgren-etal.pdf

Directement liés à inlabru : packages pointedSDMs et intSDM

- Objectif : rendre plus facile l'utilisation des modèles SDM à base de processus ponctuels (lgcp) par les écologues.
- Essentiellement basé sur inlabru et INLA
- IntSDM intégration des sources multiple et reproductibilité (produit des Workflows)



Mostert, P. S., & O'Hara, R. B. (2023). PointedSDMs: An R package to help facilitate the construction of integrated species distribution models. Methods in Ecology and Evolution, 14, 1200–1207. https://doi.org/10.1111/2041-210X.14091s

Philip Mostert, Ragnhild Bjørkås, Angeline J.H.M. Bruls, Wouter Koch, Ellen C. Martin, Sam W. Perrin intSDM : an R package for building a reproducible workflow for the field of integrated species distribution models
bioRxiv 2022.09.15.507996; doi: https://doi.org/10.1101/2022.09.15.507996

Package sdmTMB : modèles SDM avec un terme spatial du type SPDE

- Modélisation SPDE sans forcément INLA pour l'inférence (usage direct de fmesher ou de make_mesh() qui en dépend.
- Approche facilitée si usage habituel de glmmTMB, lme4 ou mgcv (syntaxe similaire pour les formula des modèles)
- Inclusion possible de seuils (ruptures, lignes brisées) ou de lissages pénalisés s(), bivariés s(x,y) (syntaxe GAM de mgcv), coefficients variant spatialement
- Approche ML ou STAN (package tmbstan)

https://pbs-assess.github.io/sdmTMB/#citation

Anderson, S.C., E.J. Ward, P.A. English, L.A.K. Barnett., J.T. Thorson. 2024. sdmTMB: an R package for fast, flexible, and user-friendly generalized linear mixed effects models with spatial and spatiotemporal random fields. bioRxiv 2022.03.24.485545

Package mgcv : modèles SDM avec un terme spatial du type SPDE

- Pour permettre une approche spatiale qui soit à la fois non paramétrique (splines) et géostatistique (SPDE Matern)
- Modélisation SPDE via fmesher

*Miller, D.L., Glennie, R. & Seaton, A.E. Understanding the Stochastic Partial Differential Equation Approach to Smoothing. JABES **25**, 1–16 (2020). https://doi.org/10.1007/s13253-019-00377-z*

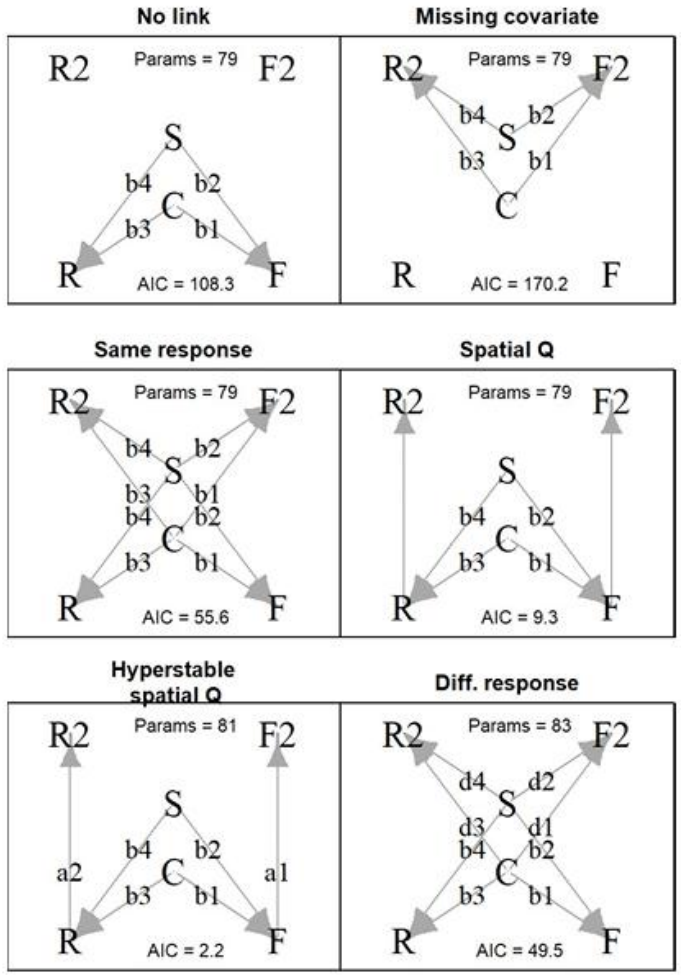
Package tinyVAST : modèles SDM multivariables avec un terme spatial du type SPDE

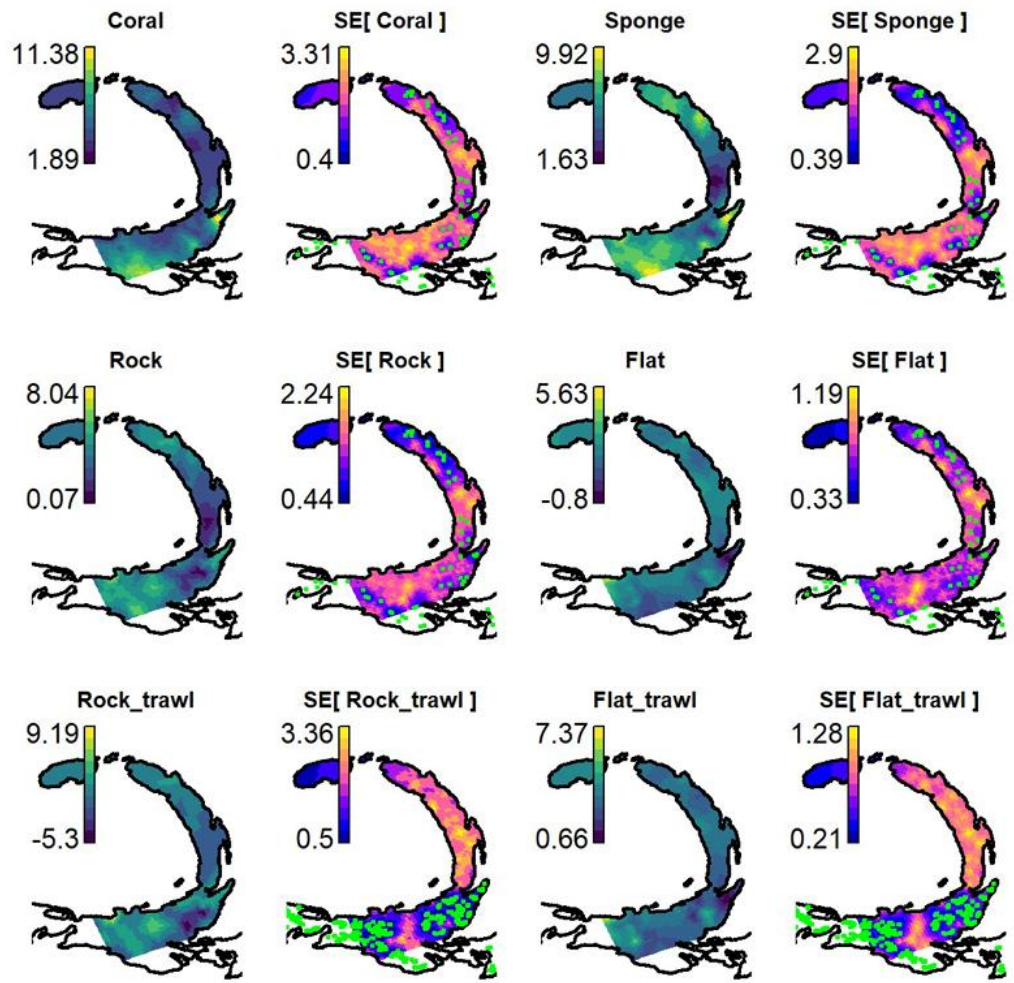
- Plus spécialement destiné aux approches multivariables
- tinyVAST tend actuellement à remplacer VAST
- Modélisation SPDE sans forcément INLA pour l'inférence

https://github.com/vast-lib/tinyVAST

Thorson, J. T., S. C. Anderson, P. Goddard and C. N. Rooper. "tinyVAST: R package with an expressive interface to specify lagged and simultaneous effects in multivariate spatio-temporal models." (2024) https://doi.org/10.48550/arXiv.2401.10193

Package tinyVAST : modèles SDMmultiespèces avec un terme spatial du type SPDE





Comparaison (Tableau issu du papier sdmTMB)

	sdmTMB	VAST	R-INLA/inlabru	mgcv	spBayes	spaMM
Time-varying coefficients	✓	–[1]	✓	✓	✓	–
Spatially varying coefficients (SVC)	✓	✓	✓	✓	✓	–
GAMs[2]	✓	–	✓	✓	–	–
Threshold covariates	✓	–	✓[3]	–	–	–
Offsets	✓	✓	✓	✓	✓	✓
Spatiotemporal fields	✓	✓	✓	✓	✓[4]	–
Spatial + spatiotemporal fields	✓	✓	✓	✓	–	–
Anisotropy	✓	✓	–	✓	–	–
Correlation barriers	✓	✓	✓	✓	–	–
Separate range parameters for fields	✓	–	✓	✓	–	–
Share range parameters across fields	✓	✓	✓	–	–	–
SPDE-based	✓	✓	✓	✓[5]	–[6]	✓
NB1 distribution	✓	–	✓	✓	–	✓
NB2 distribution	✓	✓[7]	✓	✓	–	✓
Zero-truncated distributions	✓	–	✓	–	–	✓
Zero-inflated distributions	✓	✓	✓	–	–	✓
Tweedie distribution	✓	✓	✓	✓[8]	–	–
Student-t distribution	✓	–	✓	✓	–	–
Censored Poisson distribution	✓	–	✓	–	–	–
log Gaussian Cox processes	–[9]	–[9]	✓	–[9]	–[9]	–[9]
Multivariate responses	–	✓	✓	✓	✓	✓
Built-in delta/hurdle models	✓	✓	✓	–[10]	–	✓
Poisson-link delta models	✓	✓	✓	–	–	–
Likelihood weights	✓	–	✓	✓	✓	✓
Maximum/marginal likelihood	✓	✓	–	✓	–	–
Bayesian/optionally Bayesian	✓	✓	✓	✓	✓	–
Priors/penalties	✓	–	✓	–	✓	–
Matern PC priors	✓	–	✓	–	–	–
Spatial (or spatial dynamic) factor analysis	–	✓	–	–	–	–
Empirical Orthogonal Function (EOF) analysis	–	✓	–	–	–	–
Built-in area-weighted index standardization	✓	✓	–	–	–	–
Built-in cross-validation	✓	–	–	–	–	–

Discussion

Deux mondes (ou deux communautés)

- (1) basé sur INLA et SPDE Matern : avec des accès pour spécialiste ou plus pour profane (limite hypothèse multigaussienne mais approche bayésienne très efficiente en calcul)
- (2) basé sur des approches GLMM complexes (au niveau des distributions) avec un terme aléatoire du type SPDE Matern et des inférences soit fréquentielles (ML) soit bayésiennes (via STAN, package tmbstan)
- A vous de choisir celui qui vous va le mieux

Discussion

Développements en cours ou probables

- Facilitation de l'accès aux modèles spatiaux multivariables et multi-dates
- Aspects spatio-temporels en temps continu avec covariances non séparables package *INLAspacetime*
- Intégration des approches modélisatrices ou des techniques d'inférences (packages pour TOUT faire)
- ou plutôt spécialisations vis-à-vis des applications (un package par application)

Merci pour votre attention

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*

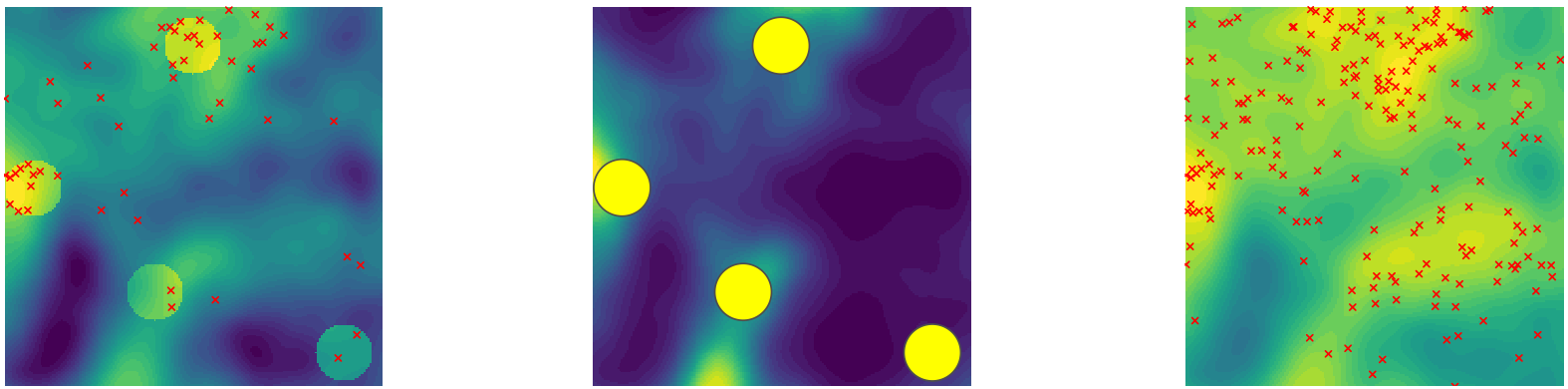
- **Objectifs :**
  - Caractérisation des **niches** et des **interactions** entre espèces
  - **Cartographie** (présence, abondance...)
  - **Identification et correction des biais** d'observation

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

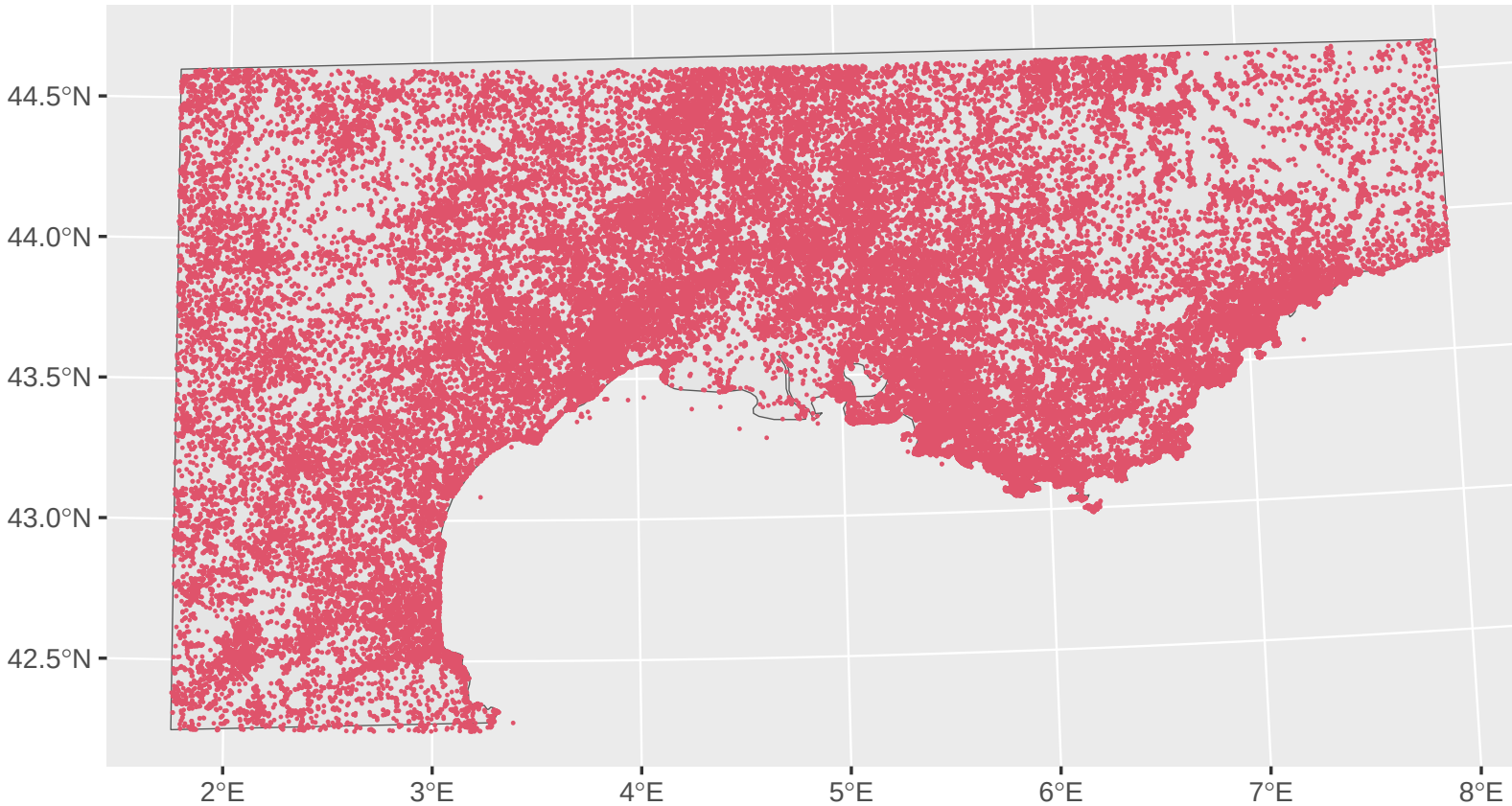
- **Présences-absences** dans les unités spatiales A_k:

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

  - Z_k suit une **loi de Bernoulli**

Exemple : Occurrences de différentes espèces (PI@ntNet)

Coordonnées géographiques exactes S_i

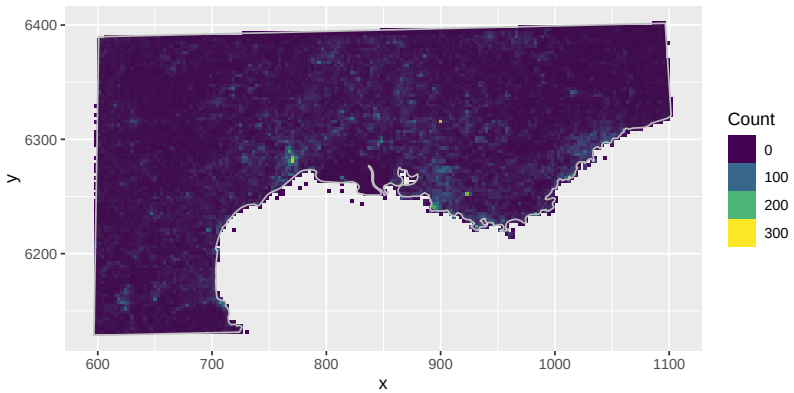


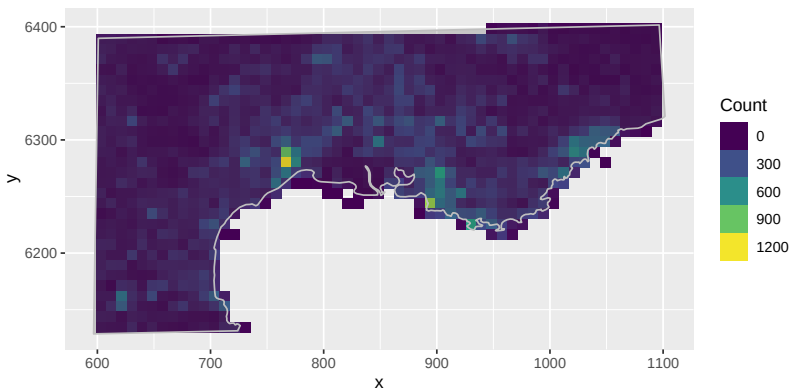
Exemple (suite) : Effet de la résolution spatiale

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

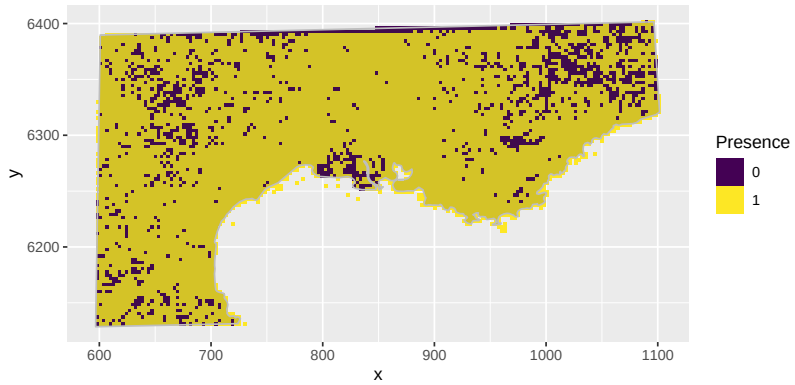
⚠ Attention à la perte d'information, surtout avec les présences-absences

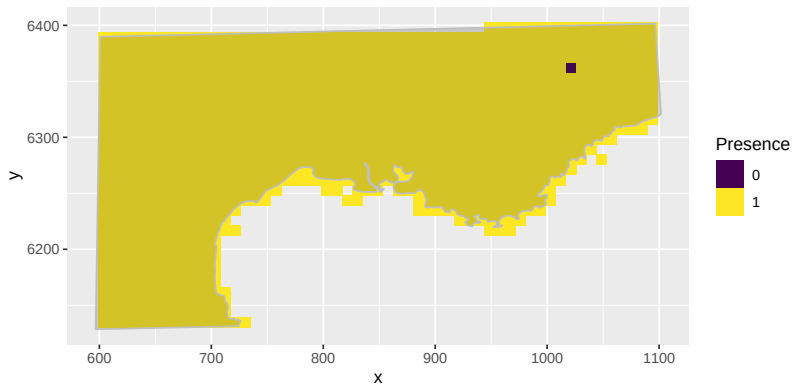
Comptages





Présences-Absences





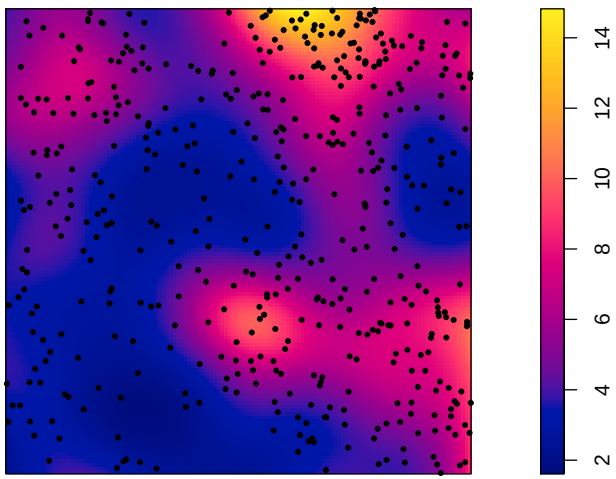
Les processus ponctuels

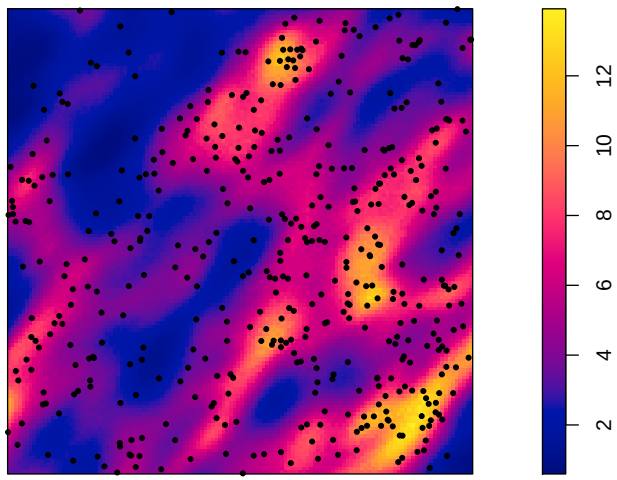
- Observations = coordonnées des occurrences (dans un domaine D)
- Intensité $\lambda(s)$ = Nombre d'occurrences moyen par unité spatiale en s
- **Processus ponctuel de Poisson** (occurrences indépendantes) :
  - Nombre d'occurrences dans $B \subset D$:

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

  - 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

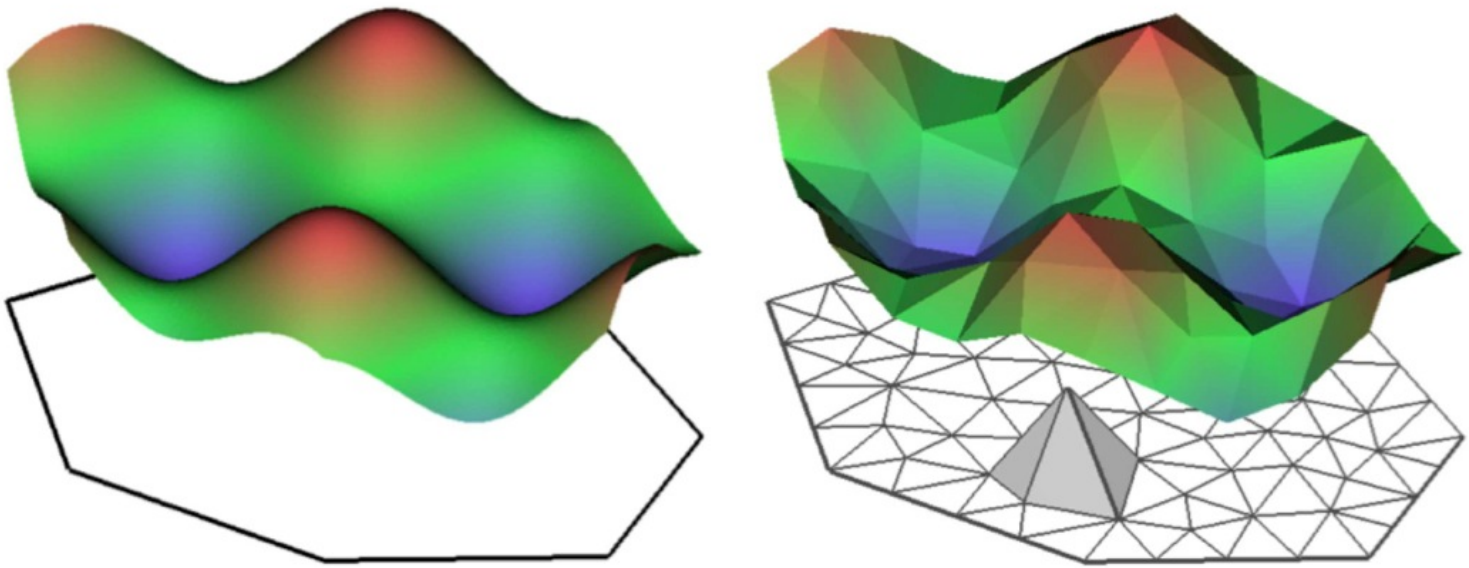
- **Hyperparamètres** : variances, portées spatiales...

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

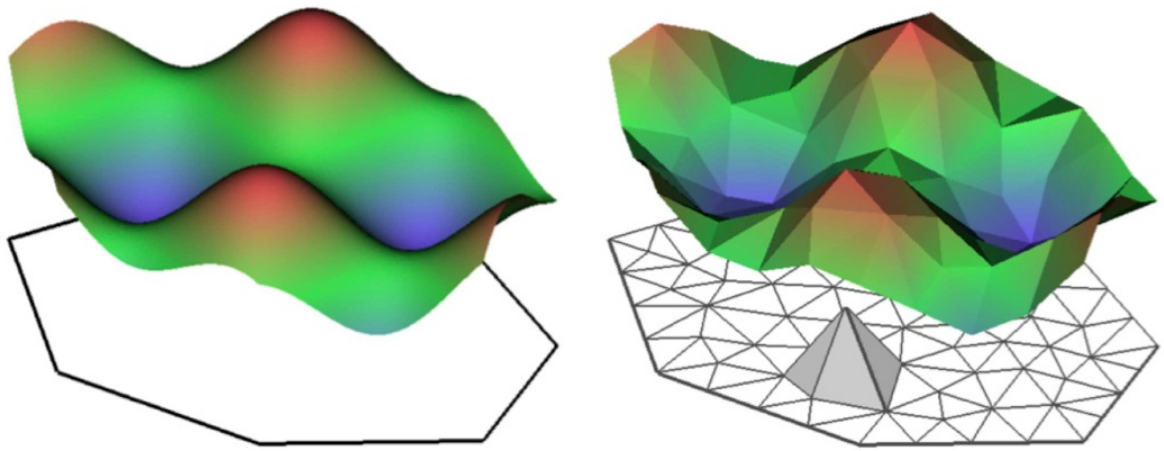
$$(\kappa - \Delta)^{\alpha/2} W(s) = \tau \varepsilon(s), \quad \alpha = \nu + d/2, \quad s \in \mathbb{R}^d$$

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) :

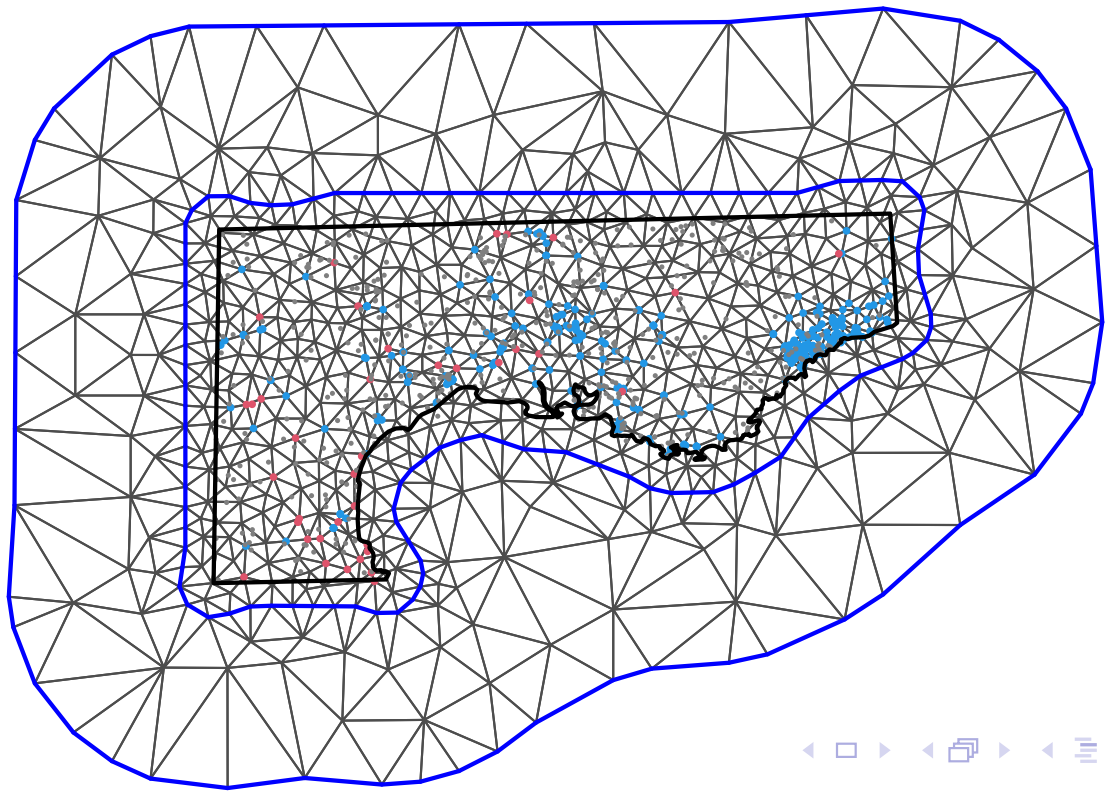
- Solution approximative $(W(s_1), \ldots, W(s_m)) \sim \mathcal{MVN}(\mathbf{0}, \boldsymbol{Q}^{-1})$
- **Matrice de précision creuse $\boldsymbol{Q}$**
- Calculs matriciels rapides quand m est grand



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 `RINLA` 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** que `R-INLA` classique 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 :

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

- **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

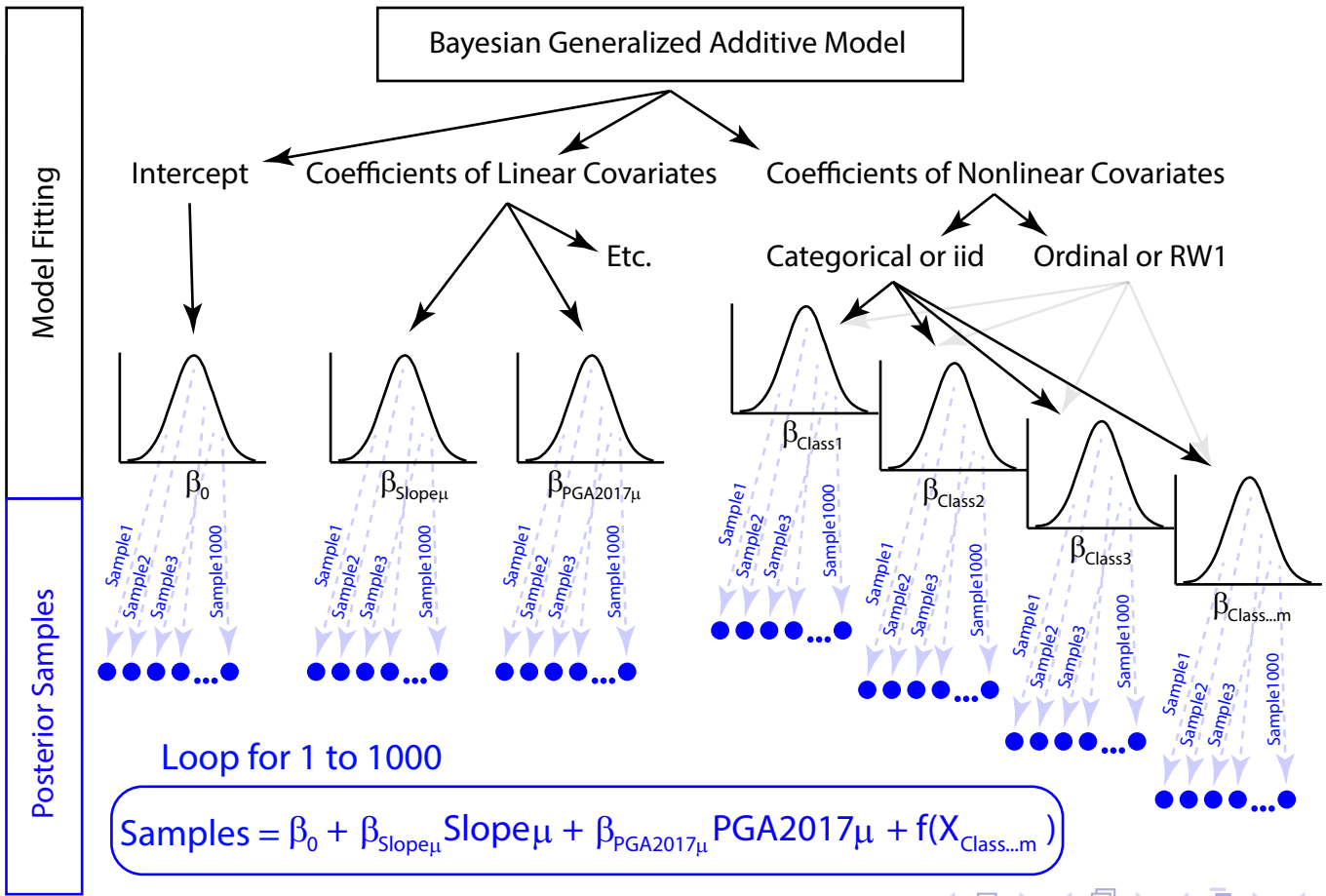
Simulation a posteriori

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.

Modèles statistiques de distribution d'espèces

– Des positions d'individus vers les différents types de modèle –

Thomas Opitz

Atelier

Modélisation spatio-temporelle en écologie avec INLA

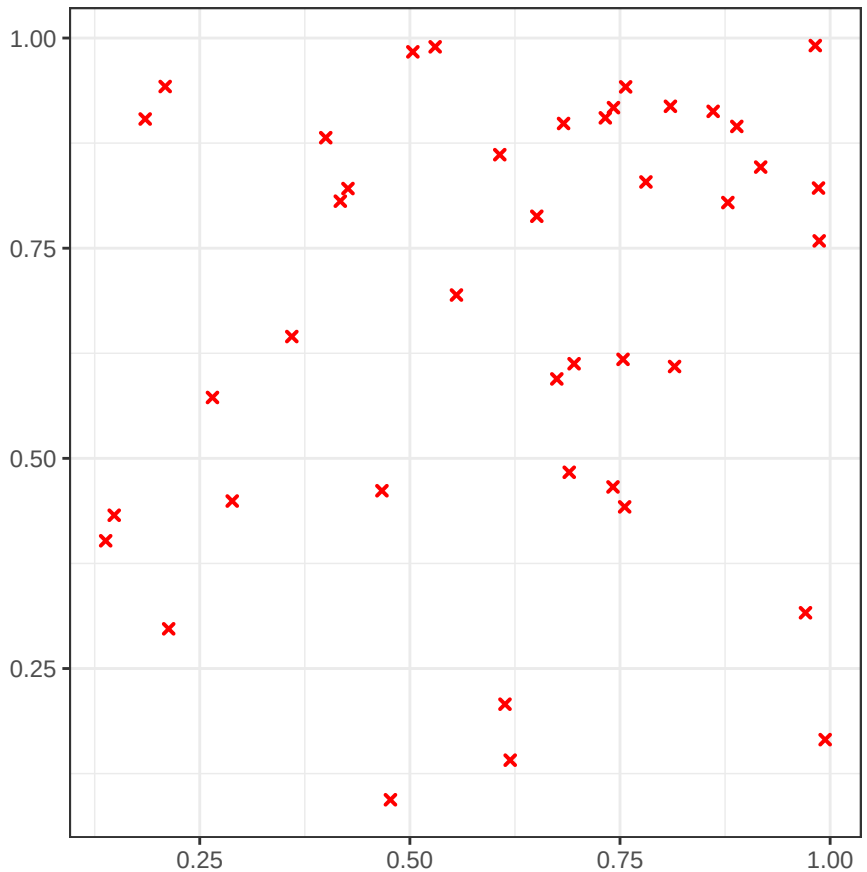
Avignon, 11-12 mars 2025

Processus ponctuel spatial

- Un ensemble de N **points aléatoires** : $\{s_1, \ldots, s_N\}$ (N est aléatoire aussi)
- Pour représenter les **positions d'objets, d'événements ou d'individus**
- Par exemple : données de présence seule (*Presence-Only*) en écologie
- Information la plus complète possible (non agrégée spatialement)

Autres termes : *Semis de point*, *pattern de points*

Exemple : semis de points simulé dans la fenêtre d'observation $D = [0, 1]^2$

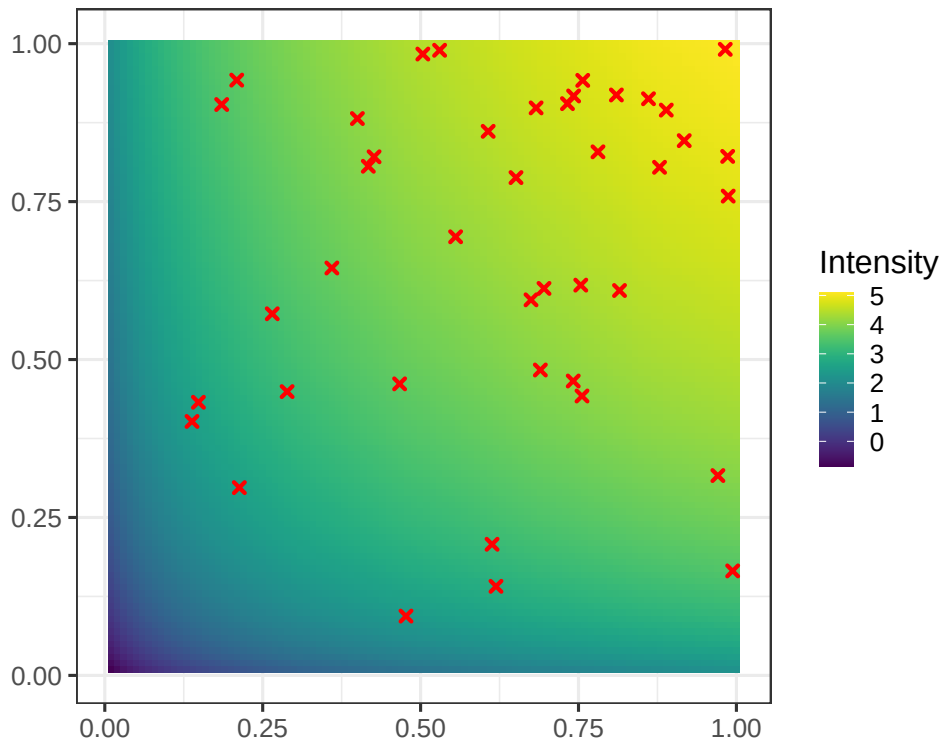


Fonction d'intensité et PP de Poisson

- La **fonction d'intensité** $\lambda(s)$, une caractéristique clé (*intensité* = *taux d'occurrence*)
- **Nombre de points attendu** dans un domaine A = Intensité agrégée sur A:

$$\lambda(A) := \mathbb{E}[N(A)] = \int_A \lambda(s)\mathrm{d}s$$

- Modèle fondamental : **processus de Poisson** (homogène si $\lambda(s) = \lambda_0$)
  $\Rightarrow$ Les événements se réalisent de façon indépendante



Modèle de régression pour l'intensité

Modèle additif généralisée (GAM) :
fonction de lien log pour relier l'intensité λ à un **prédicteur linéaire** $\eta(s)$:

$$\log \lambda(s) = \beta_0 + \sum_{k=1}^{K} \beta_k z_k(s) = \boldsymbol{\beta}^T \boldsymbol{z}(s) =: \eta(s)$$

- **Prédicteurs** $z_k(s)$, déterministes et connus :
  - covariables "physiques" (effets linéaires/fixes)
  - fonctions de base pour représenter un effet nonlinéaire/aléatoire (p. ex. fonctions splines, éléments finis pour un effet spatial)
- **Coefficients** β_k à estimer
- β_k stochastique $\Rightarrow$ **Effet stochastique**
- K potentiellement très grand (p. ex. champ spatial)
  $\Rightarrow$ **Pénalité de smoothness** (fréquentiste),
  ou **loi a priori gaussienne corrélée** (bayésien)
  $\Rightarrow$ Les deux approches rajoutent $-\boldsymbol{\beta}^T Q \boldsymbol{\beta}$ à la log-vraisemblance !

Densité de probabilité du **processus de Poisson** observé sur un domaine D:

$$\exp\left(-\int_D \lambda(s)\mathrm{d}s\right)\prod_{i=1}^{N}\lambda(s_i)$$

⇒ **Log-vraisemblance:** $\ell(\beta) = -\int_D \exp(\eta(s))\mathrm{d}s + \sum_{i=1}^{N}\eta(s_i)$

Intégration numérique : $\int_D \exp(\eta(s))\mathrm{d}s \approx \sum_{j=1}^{m}\omega_j \exp(\eta(\tilde{s}_j))$

- Schéma de discrétisation avec m points $\tilde{s}_j$
- Poids $\omega_j > 0$, et typiquement $\sum_{j=1}^{m}\omega_j = |D|$

⇒ **Calcul d'une log-vraisemblance approximative :**
Définissons $s_{N+j} = \tilde{s}_j$, $j = 1, \ldots, m$

- $y_j = 1$ et $\omega_j = 0$ pour $j = 1, \ldots, N$ (points observés)
- $y_j = 0$ pour $j = m+1, \ldots N+m$ (points de fond)

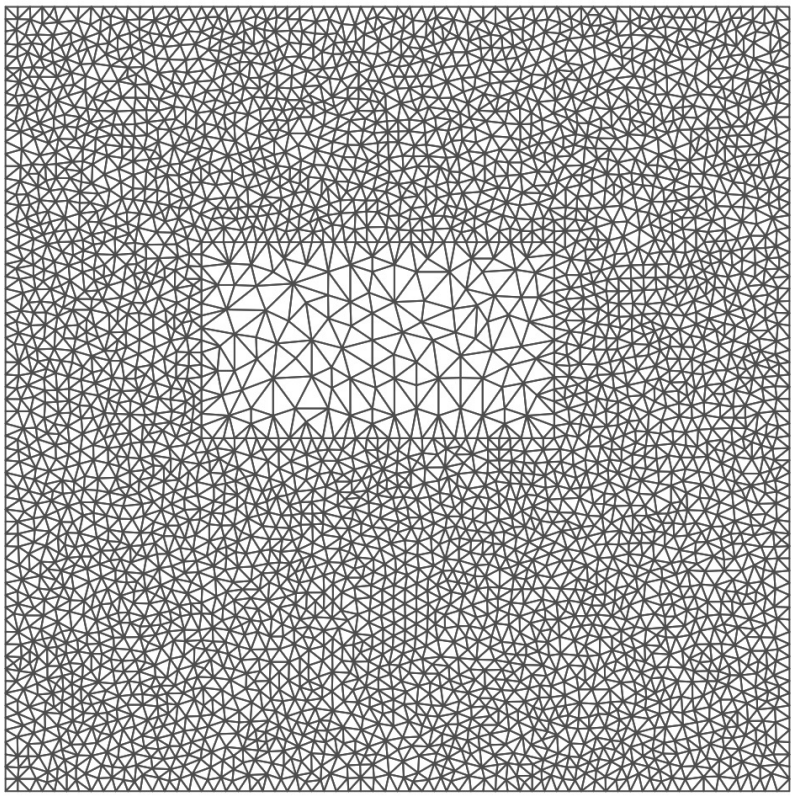
$$\ell(\beta) \approx \sum_{j=1}^{N+m}\left(y_j\eta(s_j) - \omega_j \exp(\eta(s_j))\right)$$

⇒ Formellement, une log-vraisemblance de Poisson avec observations y_j, moyennes $\omega_j \exp(\eta(s_j))$ et expositions ω_j
(INLA : `inla(y ~ ..., E = omega, family = "poisson")`)

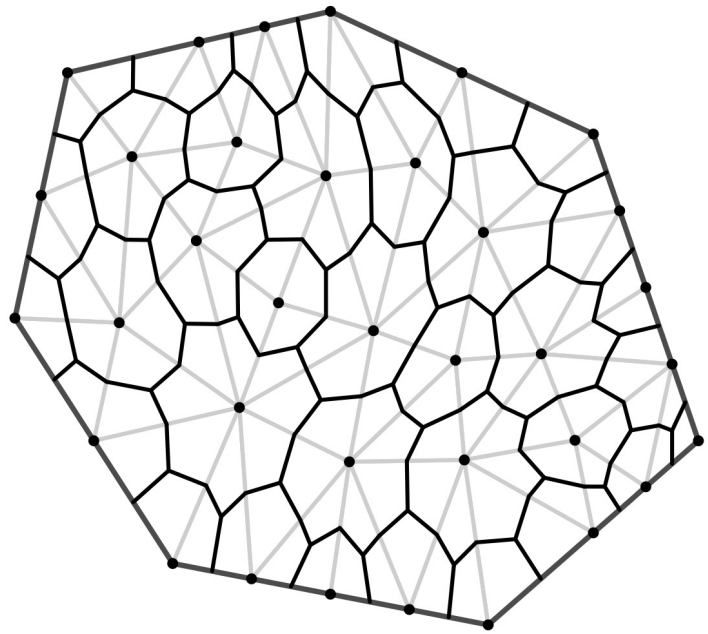
Exemple de discrétisation

- Discrétisation par **éléments finis** (triangulation) des approches "SPDE"
- Basée sur une triangulation (**mesh**) de l'espace
- Poids ω_j = Aires des cellules de la mesh duale

Exemple de mesh



Exemple de mesh duale



Simpson, D., Illian, J. B., Lindgren, F., Sørbye, S. H., Rue, H. (2016, Biometrika). Going off grid: Computationally efficient inference for log-Gaussian Cox processes.

Variante (Berman-Turner)

Idée : poids $\omega_j > 0$ pour les points observés $\{s_i\}$ dans le calcul de l'intégral

Log-vraisemblance approximative :

$$\ell(\beta) \approx \sum_{j=1}^{m+N} \omega_j \left(\frac{y_j}{\omega_j} \eta(s_k) - \exp(\eta(s_j)) \right)$$

- **Loi de Poisson** avec pseudo-observations y_i/ω_j
- Astucieux car fonctionnant avec les implémentations standard des modèles de régression sous `R` (`glm`, `gam`...)
- Implémentation par défaut dans `spatstat`
- ⚠ Pas pratique avec des gros jeux de données ($N \gg 0$)

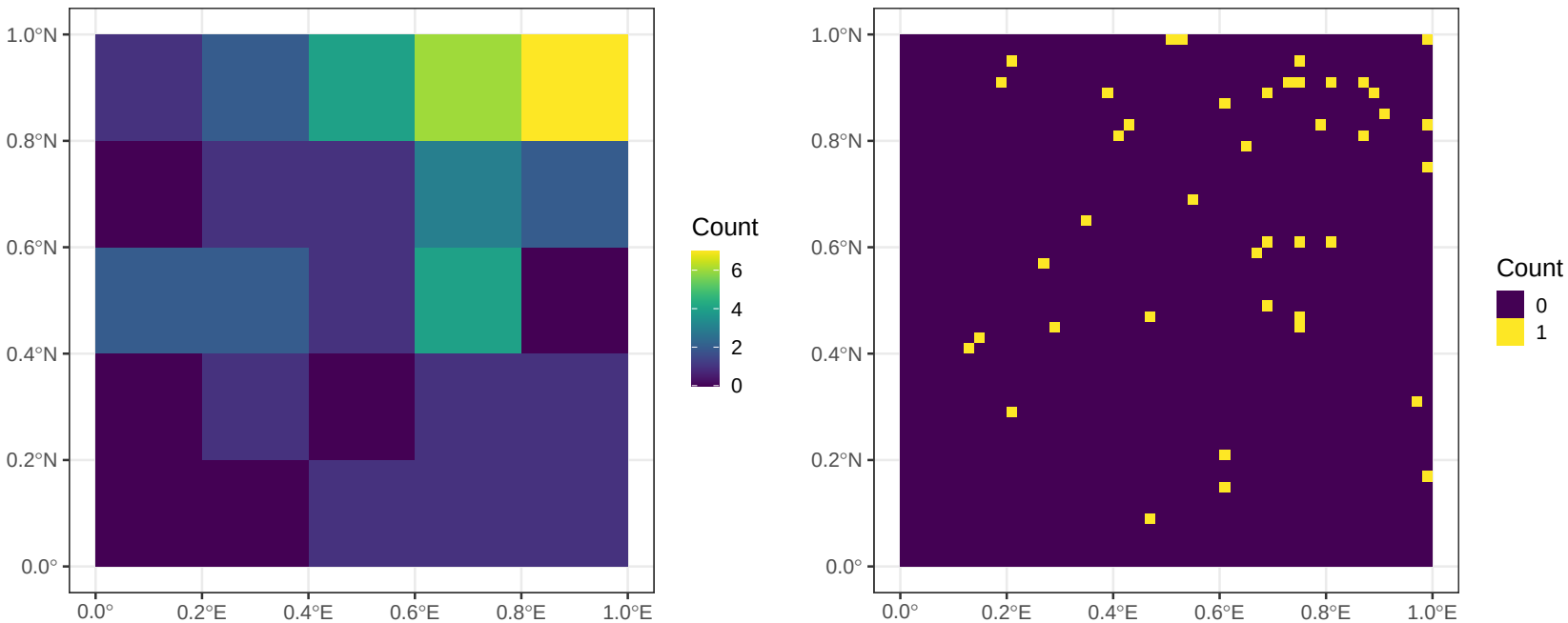
Berman, Turner (1993, JRSS-C). Approximating Point Process Likelihoods with GLIM.
Baddeley, Turner (1998, AAP). Practical maximum pseudolikelihood for spatial point patterns.

Approche "PLN" (Poisson log-linéaire)

Régression de Poisson log-linéaire :

- Diviser D en petites cellules C_j, $j = 1, \ldots, m$ (p. ex. pixels)
- Pour chaque cellule, compter les points : $N_j = \sum_{i=1}^{N} \mathbb{I}(s_i \in C_j)$
- Alors $N_j \sim \text{Poisson}\left(\int_{C_j} \lambda(s)\mathrm{d}s\right)$
- En pratique, on suppose $N_j \sim \text{Poisson}(|C_j| \times \lambda(\tilde{s}_j))$ avec un point représentatif $\tilde{s}_j$ de C_j (typiquement le barycentre)
- **Log-vraisemblance de Poisson :** $\ell(\beta) = \sum_{j=1}^{m} \left(-|C_j|\lambda(\tilde{s}_j) + N_j \log \lambda(\tilde{s}_j)\right)$

Données de comptage pour deux résolutions spatiales



Approche "PA" (Présence-Absence)

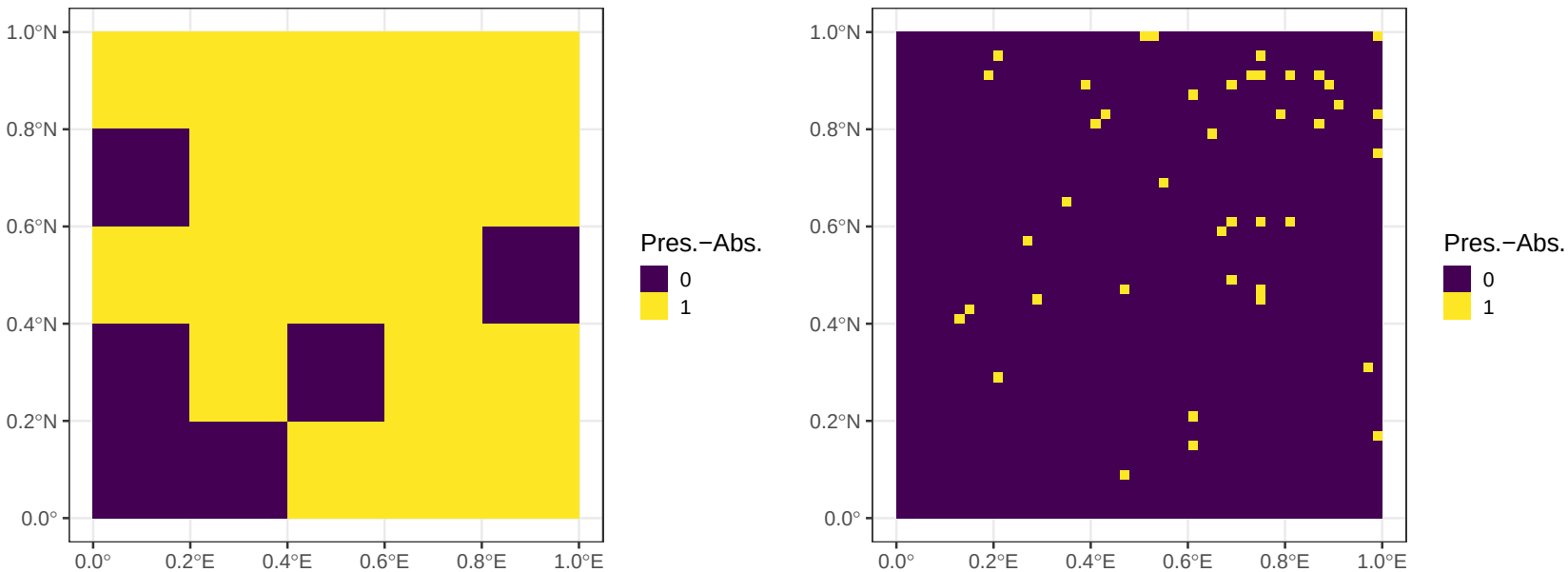
Régression binaire avec lien clogog pour les Présences-Absences :

- Discrétisation spatiale comme avant
- **Lien "log-log complémentaire"** venant de la loi de Poisson :

$$p_j = \Pr(N_j > 0) = 1 - \exp(-\exp(\eta(s))), \quad \eta(s) = \log(-\log(1 - p_j)))$$

- Utile si pas de comptages précis disponibles

Données de présence-absence pour deux résolutions spatiales



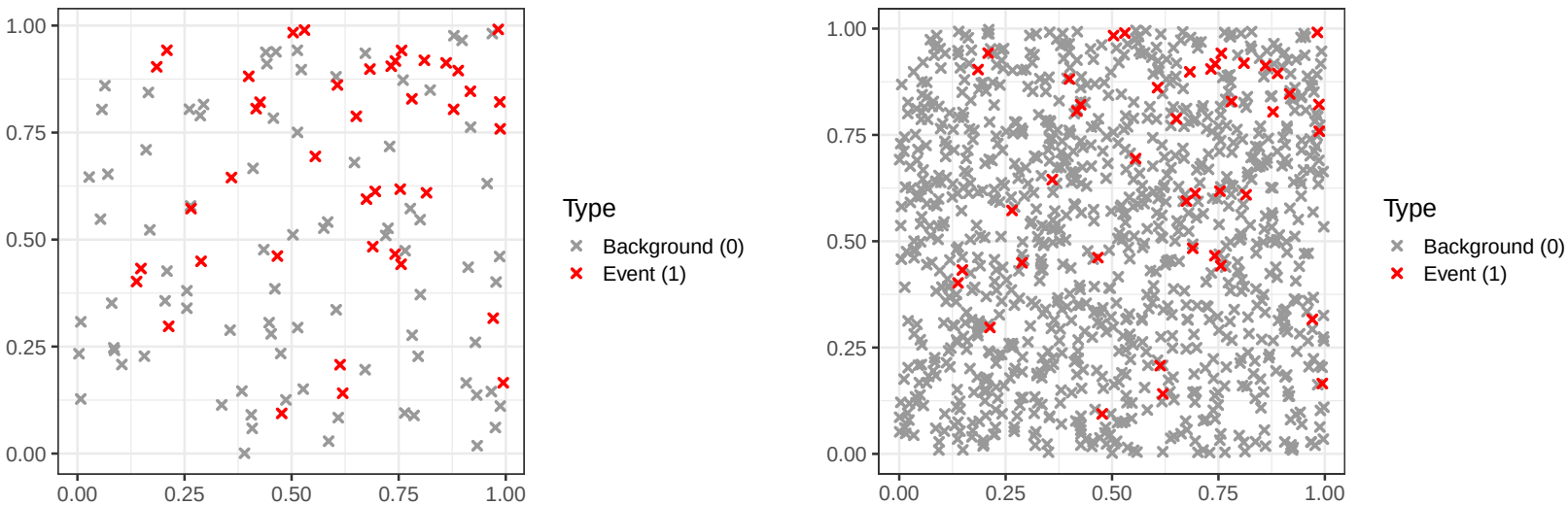
Approche "Classification"

Régression logistique pour la classification des points :

- Définir m points de fond selon une intensité $\rho(s)$ (p. ex. $\rho(s) = m/|D|$)
- Pour $s_j \in \{\text{points observes}, \text{points de fond}\}$,
  définir $y_j = 1$ si observé et $y_j = 0$ si fond
  $\Rightarrow p_j = P(y_j = 1) = \lambda(s_j)/(\lambda(s_j) + \rho(s_j))$
  $\Rightarrow \log \frac{p_j}{1-p_j} = -\log \rho(s_j) + \eta(s_j)$
  $\Rightarrow$ Régression logistique avec offset $-\log \rho(s)$

$\Rightarrow$ **Log-vraisemblance :** $\ell(\beta) = \sum_{y_j=1} \log p_j + \sum_{y_j=0} \log(1 - p_j)$

Classification avec peu/beaucoup de points de fond



Variante : régression logistique *infinitely weighted*

- Constat : bias d'estimation si prédicteur $\eta(s)$ pas bien spécifié et $m \not\gg N$
- Augmenter m ($m \gg N$)? ⚠ Coût de calcul augmente avec m !
  $\Rightarrow$ Mieux : m modérément large, et poids "infini" pour les points de fond
  $\Rightarrow \omega \gg 1$ si $s_j \in \{\text{points de fond}\}$, p. ex. $\omega = 10,000$

$\Rightarrow$ **Log-vraisemblance "infinitely weighted" :**

$$\ell(\beta) = \sum_{y_j=1} \log p_j + \omega \sum_{y_j=0} \log(1 - p_j)$$

(argument `weights` dans les fonctions `R` comme `glm`, `gam`, `inla`...)

Fithian, Hastie (2012, AOAS). Finite-sample equivalence in statistical models for presence-only data.

Comparaison des approches

- Approches “PP” et “Classification” nécessitent des **points de fond** (*dummy points*, *background points*, *control points*)

- Approches “PLN” et “PA” nécessitent une **discrétisation de l’espace**

- Quid de la gestion des covariables ?
  - Approches “PLN”, “PA” :
    agrégation des covariables par cellule est nécessaire
    (*Rightarrow* souvent moyennes ou proportions par cellule)
  - Approches “PP”, “Classification” :
    agrégation des covariables pas nécessaire
    - On a besoin des valeurs des covariables aux points de fond
    - ⚠ Attention si les points d’événements se concentrent sur des covariables “rares”

Comparaison asymptotique – Euréka !

- Laissons m représenter le nombre de cellules ou de points de fond
- Placement/tirage uniforme des cellules ou points de fond dans D
  $\Rightarrow$ En moyenne, une cellule ou un point couvre un surface $|D|/m$

Comportement asymptotique pour toutes les approches discutées :

$$\ell(\beta) \sim -\int_D \lambda(s)\mathrm{d}s + \sum_{i=1}^{N} \eta(s_i) \quad \text{lorsque } m \to \infty$$

= Log-vraisemblance exacte du PPP !

En pratique :

- Idéalement, m très grand...
- ⚠ Nombre de réponses du modèle de régression $\sim m$
  $\Rightarrow$ Côut de calcul numérique augmente avec m
  $\Rightarrow$ Il faut trouver un**compromis** !
- Besoin de stratégies pour "bien placer" les cellules ou points de fond :
  - Triangulation et mesh duale (implémentations SPDE)
  - Stratification des points de fond selon les cellules d'une grille
    Baddeley et al (2014, Biometrika). Logistic regression for spatial Gibbs point processes
  - Suites de points à discrépance faible (Sobol, Halton...) – Quasi-Monte-Carlo

Processus de Cox log-gaussien

Fonction d'intensité log-gaussienne :
$\Rightarrow$ Variables gaussiennes (corrélées ou non) pour les coefficients β_k :

$$\log \lambda(s) = \beta_0 + \sum_{k=1}^{K} \beta_k z_k(s) = \boldsymbol{\beta}^T \boldsymbol{z}(s) =: \eta(s), \quad \boldsymbol{\beta} \sim \mathcal{N}(\mathbf{0}, \Sigma)$$

Par exemple, champ gaussien $W(s)$ selon une fonction de covariance :
$\log \lambda(s) = \beta_0 + W(s)$

Utilités et interprétations :

- **Effet aléatoire :** stochasticité represente variabilité environnementale non captée par les covariables
- **Interprétation bayésienne :** loi a priori gaussienne
- 'Régularisation" et smoothness des effets si beaucoup de coefficients β_k

Implémentation : GAM(M)s (fréquentiste) ; INLA (bayésien)

Toutes les approches sont possibles — on estime toujours $\log \lambda(s)$ **!**

Fonction d'intensité modulée

Processus ponctuel observé selon l'intensité suivante :

$$\lambda_{OBS}(s) = \lambda(s) \times e(s), \quad e(s) \geq 0$$

- Terme $e(s)$ représentant une modulation de l'intensité $\lambda(s)$
  - **Exposition** (p. ex. : maladies ; surface forestière pour les incendies de forêt)
  - **Effort d'échantillonnage** $\Rightarrow$ Différents cas de figure :
    - Connu exactement (le rêve !)
    - Fortement déterminé par une covariable (e.g. distance sampling)
    - Essentiellement inconnu (Données opportunistes)
  - Cas particulier du **thinning**
    - $e(s) \leq 1$
    - On retient un point d'un PP d'intensité $\lambda(s)$ avec probabilité $e(s)$

Groupe cible pour présences-seules opportunistes

Approche par *Target-group background* :

- Espèce focale avec observations selon $\lambda_{j_0}^{OBS}(s) = \lambda_{j_0}(s)e(s)$
- Groupe cible d'espèces (y compris j_0) avec observations $\lambda_{TG}^{OBS}(s)$

Hypothèse de modélisation : $\lambda_{TG}^{OBS}(s) = \lambda_{TG}(s)e(s)$
($e(s)$ est commun à toutes les espèces)

$$\lambda_{j_0}^{OBS}(s) = \lambda_{TG}^{OBS}(s) \times \tilde{\lambda}_{j_0}(s)$$

$\Rightarrow$ On peut estimer l'**intensité relative** $\tilde{\lambda}_{j_0}(s) = \lambda_{j_0}(s)/\lambda_{TG}(s)$

Approches d'estimation :

- PLN : approcher $\lambda_{TG}^{OBS}(s)$ par comptages empiriques $\Rightarrow$ Offset $\log \lambda_{TG}^{OBS}(s)$
- Estimer $\hat{\lambda}_{TG}^{OBS}(s)$ avec un modèle
  - Variante 1 : en deux étapes
    1. Modèle univarié pour $\hat{\lambda}_{TG}^{OBS}(s)$
    2. Modèle pour $\lambda_{j_0}^{OBS}(s)$ avec offset $\log \hat{\lambda}_{TG}^{OBS}(s)$
  - Modèle bivarié pour $\lambda_{j_0}^{OBS}(s)$ et $\lambda_{TG}^{OBS}(s)$ avec champ partagé

Quelques messages-clés à retenir

- **Processus ponctuel** = représentation la plus résolue et informative possible

- Quel intérêt pour discrétiser le domaine et agréger les données ?
  - **Erreurs de saisie :**
    Si les positions des points sont imprécises
  - **Surdispersion**
    S'il y a forte multiplicité de points ⇒ Présence-Absence
    (p. ex. observation d'espèces gregaires)
  - **Côut numérique**
    S'il a un très grand nombre de points

Toutes les représentations numériques approximent le processus ponctuel !

Champs gaussians spatiotemporels $W(s,t)$

Espace et temps sont **separables** si la fonction de covariance satisfait

$$C_{ST}(s,t) = C_S(s) \times C_T(t)$$

De façon équivalente, pour $t_1 < t_2$ and sites $s_1 \neq s_2$, nous avons

$$W(s_1,t_2) \text{ indépendant de } W(s_2,t_1) \text{ conditionnellement à } W(s_1,t_1),$$

Matrice de précision de $W(s,t)$ séparable:
si Σ_{ST} est la matrice de variance-covariance de $W(s_i,t_k)$ pour $(i,k) \in \{1,2,\ldots,I\} \times \{1,\ldots,K\}$, alors la matrice de précision Q_{ST} se factorise (produit de Kronecker)

$$Q_{ST} = Q_S \otimes Q_T$$

$\Rightarrow$ **Calculs numériques facilités** (*big-n problem* des stats spatiotemporelles), surtout si les matrices Q_S et Q_T sont **creuses** (champs markoviens)

Exemples séparables pour $W(s,t)$

Modèle additivement séparable :

$$W(s,t) = W(s) + W(t)$$

Modèle avec autorégression temporelle d'ordre 1 :
(coefficient d'autocorrélation $\rho \in [-1,1]$)

$$\varepsilon_t(s) \overset{ind.}{\sim} \text{Matérn-SPDE(range)}, \quad t = 1,2,... \tag{1}$$
$$W(s,1) = \varepsilon_1(s) \tag{2}$$
$$W(s,t+1) = \rho W(s,t) + \sqrt{1-\rho^2}\varepsilon_{t+1}(s), \quad t = 1,2,... \tag{3}$$

Modèle avec marche aléatoire temporelle :

$$\varepsilon_t(s) \overset{ind.}{\sim} \text{Matérn-SPDE(range)}, \quad t = 1,2,... \tag{4}$$
$$W(s,1) = \varepsilon_1(s) \tag{5}$$
$$W(s,t+1) = W(s,t) + \varepsilon_{t+1}(s), \quad t = 1,2,... \tag{6}$$

Matrice de précision d'un modèle autorégressif temporel :

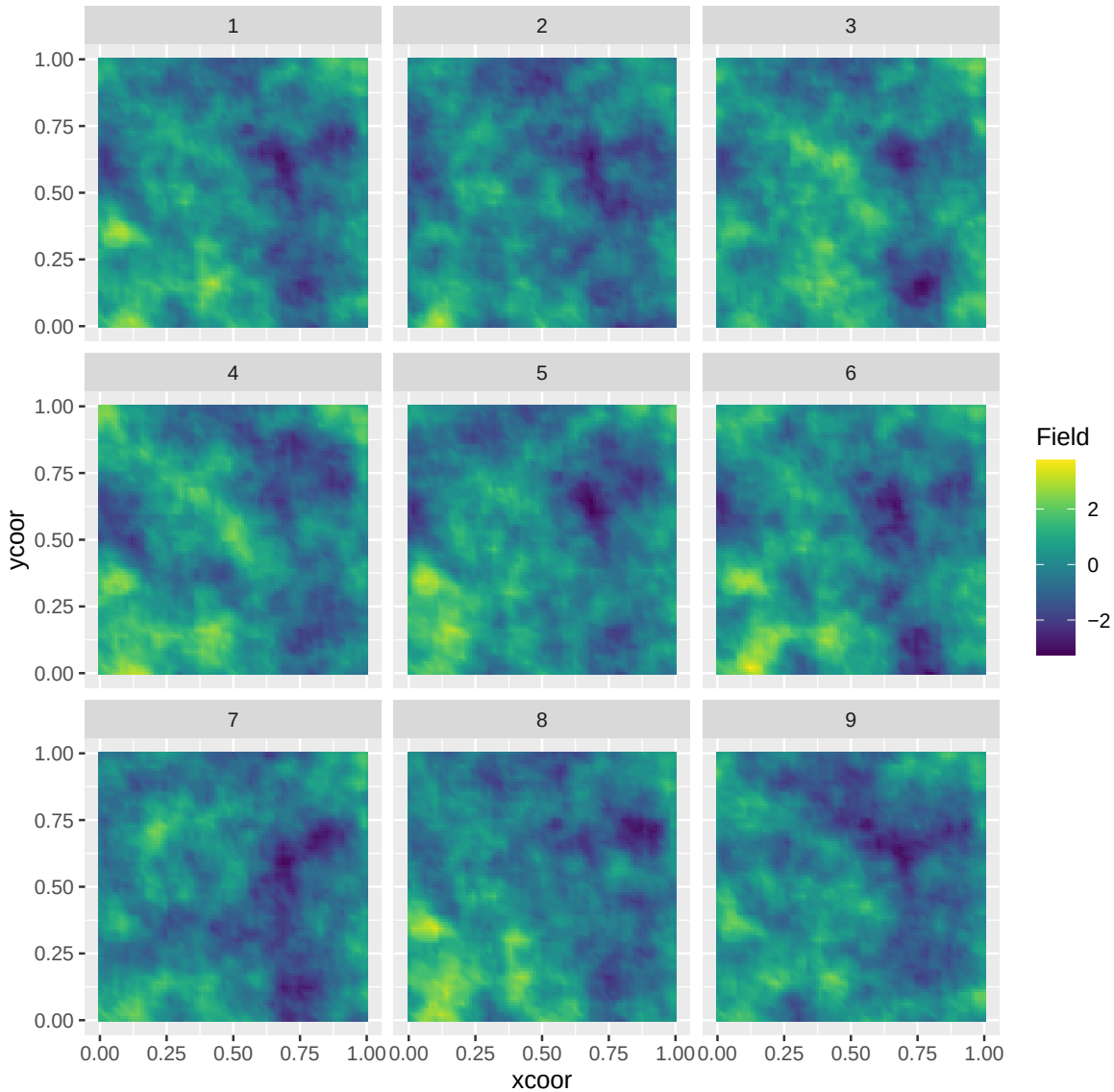
$$Q_T^{\mathrm{AR1}} = \frac{1}{1-\rho^2}\begin{pmatrix} 1 & -\rho & & & & \\ -\rho & 1+\rho^2 & -\rho & & & \\ & -\rho & 1+\rho^2 & -\rho & & \\ & & \ddots & \ddots & \ddots & \\ & & & -\rho & 1+\rho^2 & -\rho \\ & & & & -\rho & 1 \end{pmatrix}.$$

Matrice de précision spatiotemporelle :

$$Q_{ST} = Q_T^{\mathrm{AR1}} \otimes Q_S = \frac{1}{1-\rho^2}\begin{pmatrix} Q_S & -\rho Q_S & \cdots & \cdots & 0 \\ -\rho Q_S & (1+\rho^2)Q_S & -\rho Q_S & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & -\rho Q_S & (1+\rho^2)Q_S & -\rho Q_S \\ 0 & \cdots & \cdots & -\rho Q_S & Q_S \end{pmatrix}.$$

Exemple simulé

Modèle avec autorégression temporelle d'ordre 1 sur $[0,1]^2$ et $t = 1, \ldots, 9$: range$= 0.25$, sd$= 1$, $\rho = 0.8$



Implémentation comme *group models* dans INLA

- Un modèle de groupe relie (corrèle) les instances de vecteurs gaussiens avec la même structure
- Si nous avons un champ spatial par période, nous pouvons les relier avec un modèle de groupe (temporel)
- Exemples de modèles de groupe :
  - *ar1* (autorégressif d'ordre 1)
  - *rw1* (marche aléatoire d'ordre 1)
  - *exchangeable* (symétrique)
  - *iid* (pas de corrélation)

Quelques extensions nonséparables dans le package `spacetimeINLA`
(voir aussi `https://github.com/finnlindgren/spacetime-paper-code`)

Référence générale pour les modèles spatiotemporels :
`https://spacetimewithr.org/`

Spatial models with INLA

Christophe Botella, Thomas Optiz, Pascal Monestiez, Julien Papaïx

2025-03-11

Load dependencies and dataset

Our dataset is taken from:

Lasgorceux, F., Papaix, J. J., Bunz, Y., Combrisson, D., & Opitz, T. (2024). Space-time species distribution modeling for opportunistic presence-only data: a case study of passerines in a protected area. https://hal.science/hal-04616332/

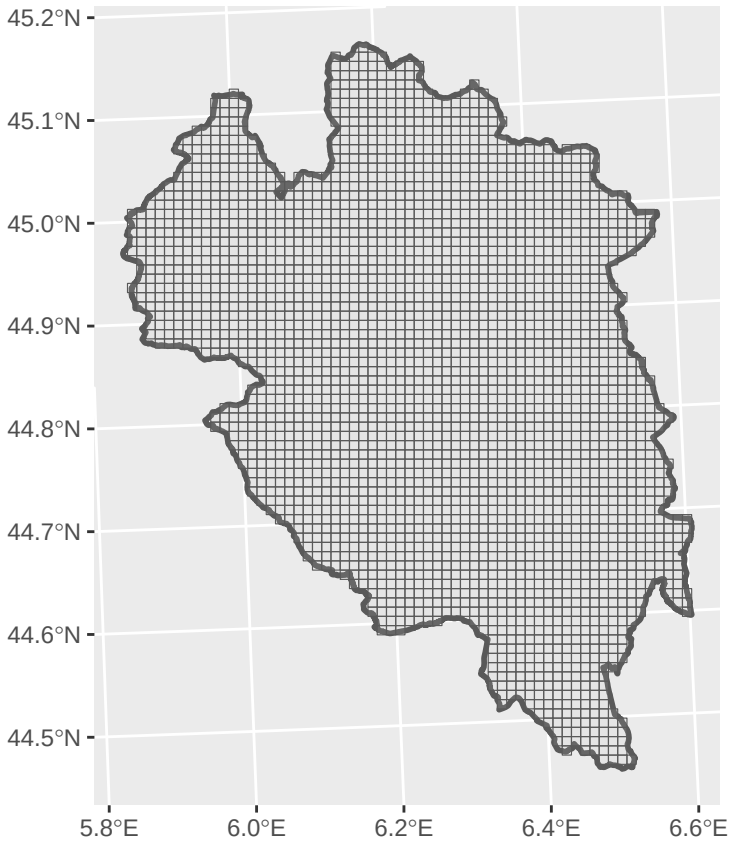
It notably contains:

- "ENP_domain": sf polygon delimitating the Park study area.
- "ENP_grid": sf data.frame of 500m cells inside the study area with various environmental covariates
- "list_of_species": Data.frame containing information for each bird species found in the Park
- "Passeriformes": Data.frame of bird species occurrences

```
#dir="C:/Users/user/pCloud local/boulot/data/atelier INLA Cisstats 2025 /atelier 2025/"
dir="/mnt/c/Users/jpapaix/Documents/modelisation/atelierINLA2025/bon/"
setwd(dir)
#### Libraries ####
source("inlabookfunctions.R")
## Spatial data packages
library(sf) # For generic spatial data manipulation
## Data manipulation packages
library(tidyverse) # For generic data manipulation and visualization
## Compute TSS to evaluate spatial predictions
library(pROC)
## INLA
library(INLA) # For Bayesian inference using INLA
library(fmesher) # For building meshes
library(viridis)

## Dataset
load(paste0(dir,"Data_for_modeling_fixed.RData"))
```

We look at the domain polygon "ENP_domain" and "ENP_grid"

```
ggplot()+
  geom_sf(data=ENP_domain,lwd=1)+
  geom_sf(data=ENP_grid,alpha=.1)
```



And look at the most commonly observed bird species in the Park.

```
list_of_species%>%
  arrange(desc(Freq))%>%
  select(-id,-Name_en)%>%
  head()
```

```
##                     Name              Name_fr             Status Freq
## 1      Fringilla coelebs    Pinson des arbres           Resident 5447
## 2   Phoenicurus ochruros      Rougequeue noir Breeding migratory 4183
## 3         Periparus ater        Mésange noire           Resident 3595
## 4     Erithacus rubecula  Rougegorge familier           Resident 3354
## 5            Parus major Mésange charbonnière           Resident 3165
## 6 Phylloscopus collybita      Pouillot véloce Breeding migratory 2944
```

Prepare focal species and background occurrences

We select a focal species in the list.

```
species_name="Accenteur alpin"
## Number of occurrences for the corresponding species
print(paste0(length(Passeriformes$species[Passeriformes$species == species_name]), " observations of ",
```

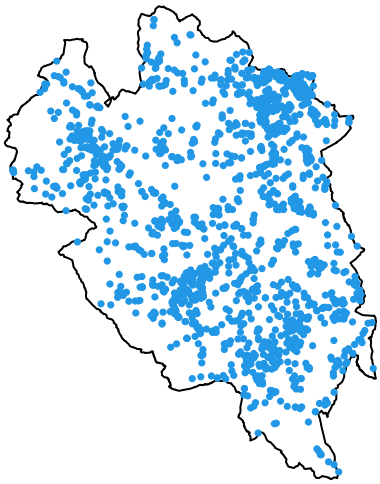
```
## [1] "1775 observations of Accenteur alpin in the Ecrins National Park between 1994 et 2021"
```

We generate a sf data.frame for the occurrences of the focal species and the Target-Group Background (TGB, used for spatial sampling bias correction when fitting the intensity). This data.frame also gathers 6 covariates for each point.

```
Passeriformes=st_as_sf(Passeriformes)
covarNames=c("Axis1","Axis2","Axis3","Axis4","Axis5","Axis6")

# Focal species occurrences
# add cell ID, and covariates for each
sp_dat_sf = Passeriformes%>%
  filter(species==species_name)%>%
  select(Month,Time_period,Year,geometry)%>%
  st_join(ENP_grid[c('id_cell',covarNames)],join=st_within)

# Target-Group background occurrences
TG_dat_sf = Passeriformes%>%
  select(Month,Time_period,Year,geometry)%>%
  st_join(ENP_grid[c('id_cell',covarNames)],join=st_within)
```

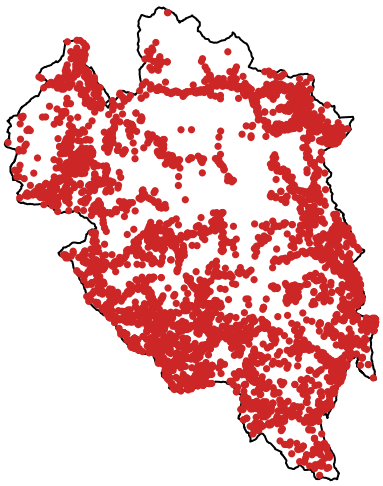
We can now map the observations of the focal species within the study domain of the Park.

```
  plot(ENP_domain)
  plot(sp_dat_sf,pch=16,col=4,add=T,cex=0.5)
```



We can also show a subset of the TGB points, including the observation of all included bird species. Their density is classically used as a proxy for sampling effort.

```
  plot(ENP_domain)
  plot(TG_dat_sf[sample(1:nrow(TG_dat_sf),5000),],
       pch=16,col='firebrick3',add=T,cex=0.5)
```



Fit a Poisson-GLMM model with a latent spatial field to the species occurrences

Noting C the number of grid cells within our study area, and T the number of time periods considered. We aim at fitting the following Poisson-GLMM model.

[

$$
\begin{aligned}
Y_{i,t} &\sim P(\lambda_{i,t}) \quad i \in [1, C], t \in [1, T] \\
\log(\lambda_{i,t}) &= \alpha + \log(nTG_{i,t}) + \beta^T Covar_i + TP_t + G(z_i) \\
\text{Where:} & \\
nTG_i &: \text{Count of TGB occurrences in cell i for period t} \\
Covar_i &: \text{Covariate vector for cell i} \\
TP_t &: \text{Gaussian random effect of time-period t} \\
G(z_i) &: \text{Spatial gaussian effect at center } z_i \text{ of cell i}
\end{aligned}
\tag{1}
$$

]

Specifically, thanks to INLA, we'll model and fit G as a spatial gaussian field with a Matérn correlation function. The other components are classical components of Genelarized Linear Mixed Models (GLMM), with the vector of parameters β being tied to the fixed effects, and TP_t being a random effect tied to the time period t. The latter models the impact of the time period on the overall intensity value, without modulating its relative value across space. In this model, $nTG_{i,t}$ is an offset which can be interpreted as a multiplicative factor (varying across space and time) of the occurrence count, aiming at correcting spatio-temporal sampling bias.

Count TG and focal species occurrences per cell

```
tmp <- sp_dat_sf %>%
  st_drop_geometry()%>%
  select(id_cell, Time_period)%>%
  group_by(id_cell, Time_period)%>%
  summarize(nFocal=n())

Poisson_sf <- TG_dat_sf %>%
  group_by(id_cell, Time_period) %>%
  summarize(
    across(-c(Month,Year), \(x) mean(x, na.rm = TRUE)),
    nTG=n()) %>%
  merge(tmp,by=c('id_cell','Time_period'),all.x=T)%>%
  as.data.frame()
Poisson_sf$nFocal[is.na(Poisson_sf$nFocal)]=0

head(Poisson_sf)
```

```
##   id_cell Time_period       Axis1     Axis2     Axis3      Axis4      Axis5
## 1  cell_1           4 -0.08895126 0.7822310 0.3855413 0.06767522  0.1853005
## 2  cell_1           5 -0.08895126 0.7822310 0.3855413 0.06767522  0.1853005
## 3  cell_1           6 -0.08895126 0.7822310 0.3855413 0.06767522  0.1853005
## 4 cell_10           0 -0.69735524 0.7431553 0.3827320 0.74265032 -0.7415284
## 5 cell_10           4 -0.69735524 0.7431553 0.3827320 0.74265032 -0.7415284
## 6 cell_10           5 -0.69735524 0.7431553 0.3827320 0.74265032 -0.7415284
##       Axis6 nTG nFocal                       geometry
## 1 0.3435648   2      0       POINT (976247.9 6379313)
## 2 0.3435648  14      0 MULTIPOINT ((975984 6379653...
## 3 0.3435648   5      0 MULTIPOINT ((976254.8 63793...
## 4 0.6235026  11      0       POINT (977996.1 6380472)
## 5 0.6235026   1      0       POINT (978178.2 6380853)
## 6 0.6235026  10      0 MULTIPOINT ((977830 6380380...
```

Fixed + (temporal) random effects

Prepare the fixed effects of the environmental covariates.

```
fixed_effects = Poisson_sf[,c('nTG',paste0('Axis',1:6))]%>%
  mutate(intercept = 1,
         log_TG_Count=log(nTG)# linear predictor->log-scale
         )%>%
  select(-nTG)%>%st_drop_geometry()%>%as.data.frame()
```

Prepare the random time-period effect.

```
Time_period = data.frame(Time_period = Poisson_sf$Time_period)
Tipe_period_precision_prior = list(theta = list(initial = log(4),
                                                prior = "pc.prec",
                                                param = c(1, 0.5)))
# P(sig>sig_0) = alpha
# sig_0 = 1, alpha = 0.5
```

Set up components for the latent spatial field

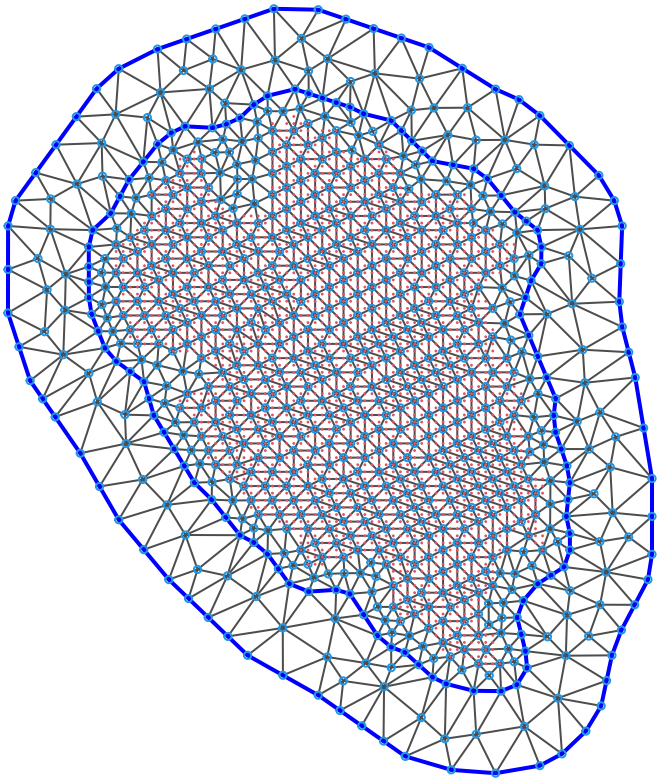
We first need to define the spatial mesh of latent gaussian variables. We get the coordinates of grid centroids and use them to build inner and outter boundaries for the mesh. The area between the inner and outter boundaries aims at minimizing boundary effects. Then, build the mesh of vertices at which the latent gaussian variables will be estimated.

```
grid_coords=as.matrix(ENP_grid[,c('x','y')]%>%st_drop_geometry())
## Compute the boundaries of the mesh
# - Inner boundary with slight concavity to follow data
bndint = inla.nonconvex.hull(grid_coords, convex = -0.05)
# - Outer boundary with stronger concavity for extension
bndext = inla.nonconvex.hull(grid_coords, convex = -0.2)
# Build triangular mesh:
# - max.edge: Maximum allowed triangle edge length (inner, outer zones)
# - cutoff: Minimum distance between mesh vertices
mesh = fm_mesh_2d_inla(grid_coords,
                       boundary = list(bndint, bndext),
                       max.edge = c(2500, 8000), cutoff = 2000)
print(mesh$n) # Number of mesh vertices
```

```
## [1] 851
```

We can now plot this mesh and see that it is rougher than the resolution of the grid, but also extends its spatial coverage.

```
par(mar = rep(0.5, 4))
plot(mesh, main = "", asp = 1)
points(grid_coords,col=2,cex=0.2,pch=16)
points(mesh$loc, col=4, cex=0.5)
```



Get coordinates of cell centroids for each sampled spatio-temporal cell. Compute the observation matrix A, which enables to linearly approximate latent gaussian field value in each cell from its value in surrounding mesh vertices.

```
# cell centroids coordinates per sampled spatio-temporal cell.
Poisson_sf=Poisson_sf %>%
merge(ENP_grid[,c('x','y','id_cell')]%>%st_drop_geometry(),
      by="id_cell",all.x=T)
coord_data = as.matrix(Poisson_sf[,c('x','y')])
# Compute the observation matrix A
A = inla.spde.make.A(mesh, loc = coord_data)
# Indices for the total observation matrix A implemented with inla.stack.
idx_sp = inla.spde.make.index("sp",n.spde = mesh$n)
```

Create the INLA stack

```
stk = inla.stack(
  data = list(y = as.numeric(Poisson_sf$nFocal)),  # Specify the response variable
  A = list(1, A, 1),  # Vector of multiplication factors for random and fixed effects
  effects = list(fixed_effects, idx_sp, Time_period),
  tag = "obs"  # Detail the random and fixed effects
)
```

Define the model and fit it

Define the 2D-SPDE priors for the spatial effect

```
nu = 1
d = 2 # dimension
alpha = nu + d / 2
spde_sp = inla.spde2.pcmatern(
  mesh = mesh, alpha = alpha,constr = T,
  prior.range = c(10, 0.5), prior.sigma = c(1, 0.5)
)
```

Define the model formula.

```
## Express y as a function of the linear predictor in the formulate variable
formula = y ~ Axis1 + Axis2 + Axis3 + Axis4 + Axis5 + Axis6 +
  f(sp, model = spde_sp) +
  f(Time_period, model = "iid")
```

Call inla to fit the model

```
fit = inla(
  formula, # Formula of the model
  offset = fixed_effects$log_TG_Count,
  data = inla.stack.data(stk), # data = Observation matrix
  family = "poisson",  # Distribution of the response variable
  control.compute = list(cpo=T,
                         dic=T,
                         mlik=T,
                         waic=T,
                         return.marginals.predictor = T,
                         config = TRUE),  # Statistics criteria
  control.predictor=list(compute = FALSE,
                         A = inla.stack.A(stk),
                         link = 1),# link = 1 to compute the fitted values with the links function
  control.inla = list(int.strategy = "eb", strategy = "adaptive"),# Optimization settings
  verbose = F,#T, # Text option
  #inla.mode = "experimental",
  num.threads = 10)
summary(fit)
```

```
## Time used:
##     Pre = 0.402, Running = 1.66, Post = 0.231, Total = 2.29
## Fixed effects:
##               mean    sd 0.025quant 0.5quant 0.975quant   mode kld
## (Intercept) -0.271 0.157     -0.578   -0.271      0.036 -0.271   0
## Axis1        0.794 0.030      0.736    0.794      0.852  0.794   0
## Axis2       -0.537 0.037     -0.611   -0.537     -0.464 -0.537   0
## Axis3       -0.073 0.033     -0.136   -0.073     -0.009 -0.073   0
## Axis4        0.034 0.042     -0.048    0.034      0.116  0.034   0
## Axis5        0.081 0.040      0.002    0.081      0.160  0.081   0
## Axis6        0.038 0.051     -0.062    0.038      0.139  0.038   0
##
## Random effects:
##   Name     Model
##     sp SPDE2 model
##    Time_period IID model
##
## Model hyperparameters:
```

```
##                              mean      sd 0.025quant 0.5quant 0.975quant   mode
## Range for sp              763.222 834.926     23.806  476.648   2995.104 45.745
## Stdev for sp               10.420  23.224      0.938    4.507     58.159  1.752
## Precision for Time_period   0.141   0.085      0.033    0.122      0.355  0.085
##
## Deviance Information Criterion (DIC) ...............: 9321.70
## Deviance Information Criterion (DIC, saturated) ....: 6508.12
## Effective number of parameters .....................: 291.19
##
## Watanabe-Akaike information criterion (WAIC) ...: 9465.34
## Effective number of parameters .................: 393.65
##
## Marginal log-Likelihood:  -4843.47
## 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)')
```

Visualize the intensity surface

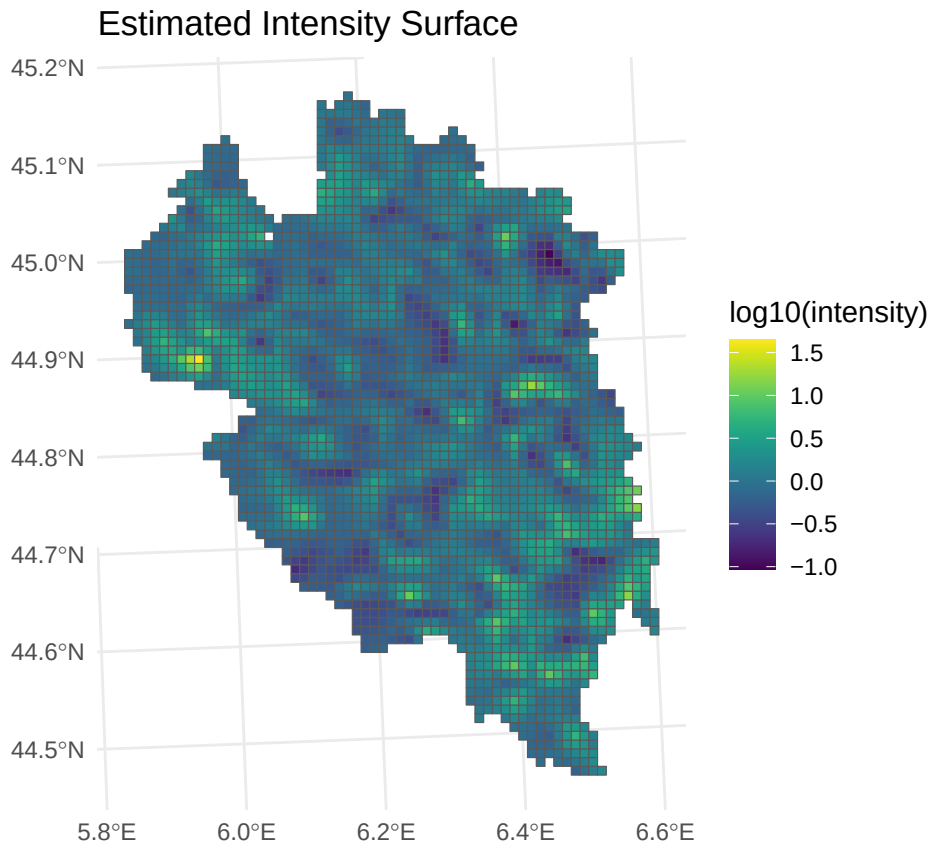
We'll compute the intensity over the grid without considering the time period effect (constant factor).

We use the 500m grid. We first predict the intensity component on this grid (covariates fixed effects + Gaussian spatial field).

```
# Create projection matrix from mesh to prediction points
A_pred <- inla.spde.make.A(mesh = mesh, loc = grid_coords)

# Get estimated spatial field values (posterior mean)
spatial_field <- fit$summary.random$sp$mean

# Calculate spatial component at prediction points
spatial_part <- as.vector(A_pred %*% spatial_field)
# Calculate Fixed Effects
X=ENP_grid[,covarNames]%>%
  st_drop_geometry()%>%
  as.matrix()
X=cbind(intercept=1,X)
# Get fixed effects coefficients
beta <- fit$summary.fixed$mean

# Calculate fixed effects component (intercept + covariates)
fixed_part <- X %*% beta
```

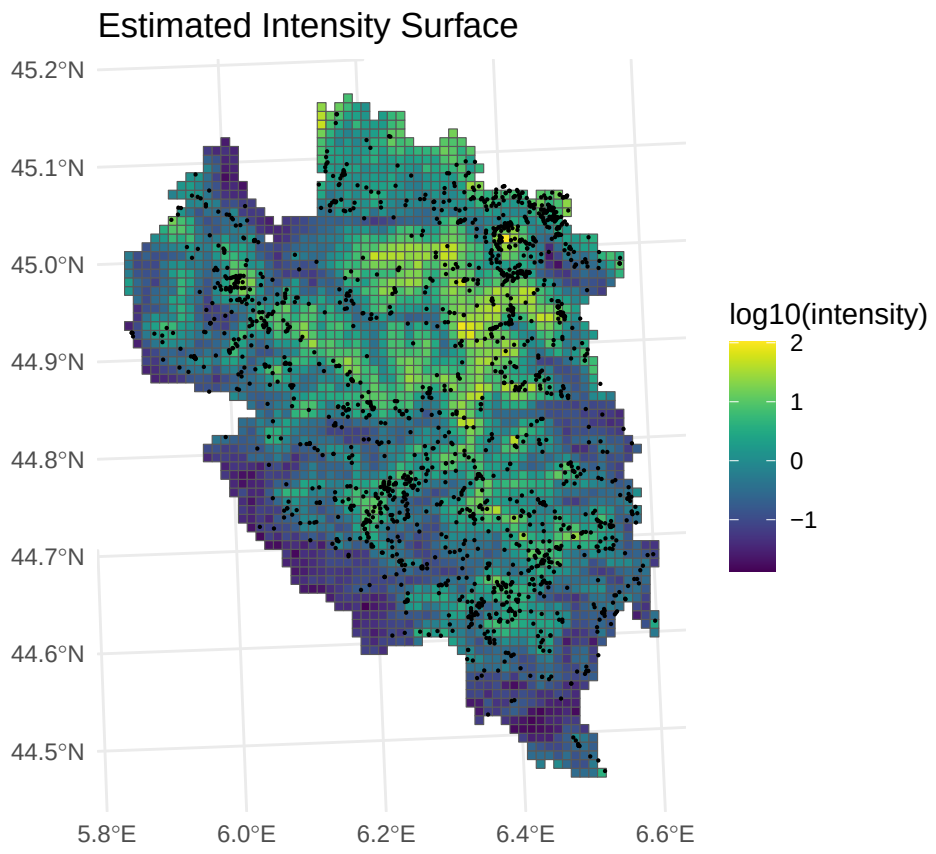
Let's first look at the purely spatial factor of the intensity (log-gaussian field).

```
ENP_grid$intensity <- exp(spatial_part)

# With geom_sf
ggplot() +
  geom_sf(data=ENP_grid,
          aes(fill=log10(intensity))) +
  scale_fill_viridis()+
  theme_minimal() +
  labs(title = "Estimated Intensity Surface")
```



Then, compute and plot the TGB-corrected intensity surface by adding log-components (Spatial and fixed effects) and exponentiating.

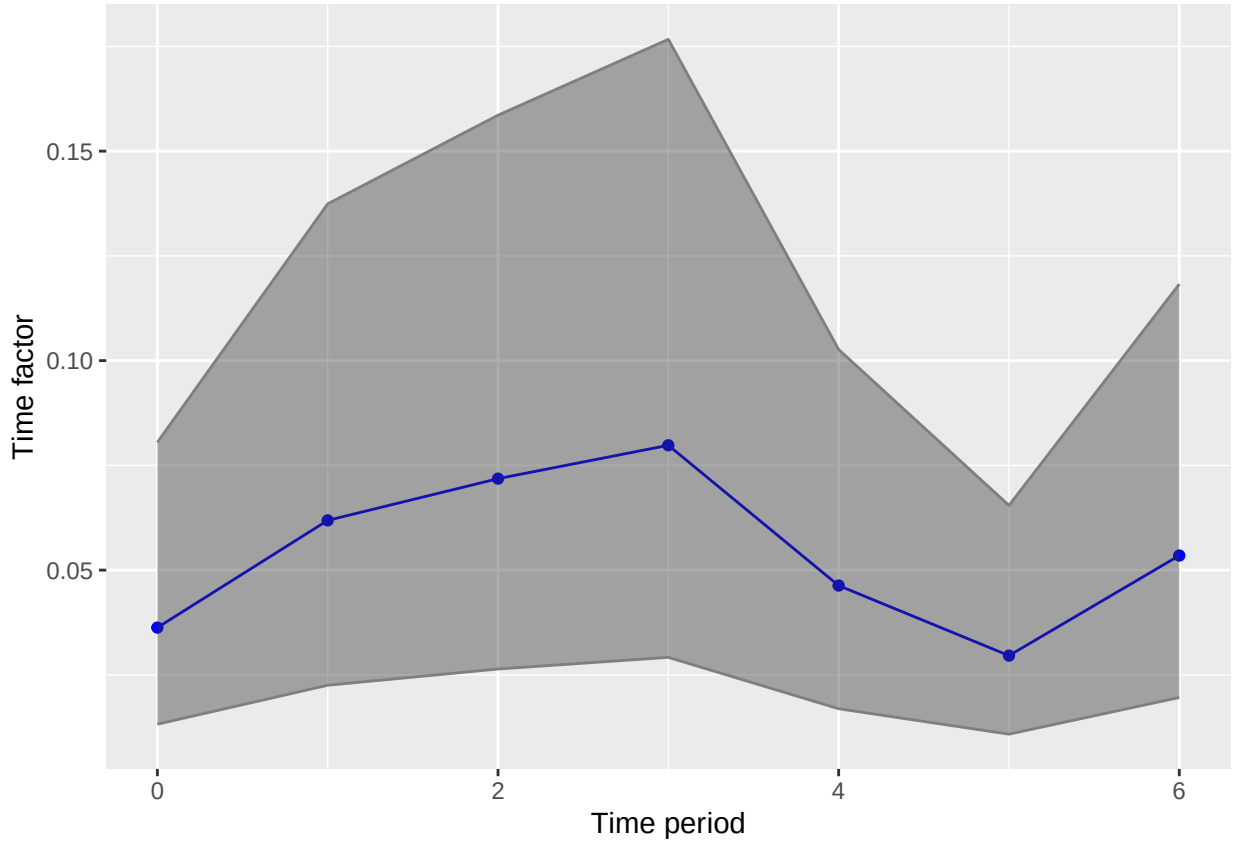
```
ENP_grid$intensity <- exp(fixed_part+spatial_part)

# With geom_sf
ggplot() +
  geom_sf(data=ENP_grid,
          aes(fill=log10(intensity))) +
  geom_sf(data=sp_dat_sf,size=.1)+
  scale_fill_viridis()+
  theme_minimal() +
  labs(title = "Estimated Intensity Surface")
```



Time-period effect

The effect of the time-period consists in a random temporal factor multiplying the total intensity. We can plot this factor as a function of the time period, each representing four consecutive years.

```
periods=fit$summary.random$Time_period%>%
  mutate(mean=exp(mean+sd^2/2),
         median=exp(mean),
         quant025=exp(`0.025quant`),
         quant975=exp(`0.975quant`))

ggplot(periods,aes(x=ID,y=mean))+
    geom_line(color="blue")+
  geom_point(color="blue")+
    geom_ribbon(data=periods,aes(x=ID,ymin=quant025,ymax=quant975),
                color="grey50",alpha=.4)+
  ylab('Time factor')+
  xlab('Time period')
```



Evaluate fit of intensity to data

We can now evaluate the fit of our fitted intensity (multiplied by the function across the grid cells to the occurrence count. One way to do so is to compute the R^2 between the observed count and the fitted intensity in the cell, or the AUC between the cells of detection and non-detection.

```
eval=ENP_grid%>%
  select(id_cell,intensity)%>%
  merge(Poisson_sf[,c('id_cell','nFocal','nTG')],by="id_cell",all.x=T)%>%
  filter(!is.na(nFocal))%>%
  mutate(intensity=intensity*nTG)
print(paste0('R2 = ',cor(eval$intensity,eval$nFocal)^2))
```

```
## [1] "R2 = 0.386359667674507"
```

```
pos=eval$intensity[eval$nFocal>0]
neg=eval$intensity[eval$nFocal==0]
AUC=as.numeric(wilcox.test(pos,neg)$statistic)/(length(pos)*length(neg))
print(paste0('AUC = ',AUC))
```

```
## [1] "AUC = 0.820210616738744"
```

```
test_roc <- roc(as.numeric(eval$nFocal>0), eval$intensity)
print(paste0('TSS = ',max(test_roc$sensitivities + test_roc$specificities - 1)))
```

```
## [1] "TSS = 0.478995339755471"
```

Spatial models with INLA

Julien Papaïx, Thomas Optiz, Pascal Monestiez, Christophe Botella

2025-03-11

Load dependencies and dataset

Our dataset notably contains:

- "ENP_domain": sf polygon delimitating the Park study area.
- "ENP_grid": sf data.frame of 500m cells inside the study area with various environmental covariates
- "list_of_species": Data.frame containing information for each bird species found in the Park
- "Passeriformes": Data.frame of bird species occurrences

```
#dir="C:/Users/user/pCloud local/boulot/data/atelier INLA Cisstats 2025 /atelier 2025/"
dir="/mnt/c/Users/jpapaix/Documents/modelisation/atelierINLA2025/bon/"
setwd(dir)
#### Libraries ####
source("inlabookfunctions.R")
## Spatial data packages
library(sf) # For generic spatial data manipulation
## Data manipulation packages
library(tidyverse) # For generic data manipulation and visualization
library(viridis)
library(pROC)
library(patchwork)
library(ggplot2)

## INLA
library(INLA) # For Bayesian inference using INLA
library(fmesher) # For building meshes
## Dataset
load(paste0(dir,"Data_for_modeling_fixed.RData"))
```

Prepare species and backgrounbd occurrences

We select a focal species in the list of bird species found in the Park.

```
print(list_of_species$Name_fr)
```

```
##  [1] "Sizerin flammé"             "Rousserolle verderolle"
##  [3] "Orite à longue queue"       "Alouette des champs"
##  [5] "Pipit spioncelle"           "Pipit des arbres"
##  [7] "Jaseur boréal"              "Chardonneret élégant"
##  [9] "Venturon montagnard"        "Grimpereau des jardins"
## [11] "Grimpereau des bois"        "Rougequeue à front blanc"
## [13] "Cincle plongeur"            "Grosbec casse-noyaux"
## [15] "Corneille noire"            "Corneille mantelée"
```

```
## [17] "Choucas des tours"           "Mésange bleue"
## [19] "Hirondelle de fenêtre"       "Bruant fou"
## [21] "Bruant zizi"                 "Bruant jaune"
## [23] "Bruant ortolan"              "Rougegorge familier"
## [25] "Gobemouche noir"             "Pinson des arbres"
## [27] "Pinson du nord"              "Geai des chênes"
## [29] "Hirondelle rustique"         "Pie-grièche écorcheur"
## [31] "Linotte mélodieuse"          "Mésange huppée"
## [33] "Bec-croisé des sapins"       "Alouette lulu"
## [35] "Rossignol philomèle"         "Monticole de roche"
## [37] "Niverolle alpine"            "Bergeronnette grise"
## [39] "Bergeronnette des ruisseaux" "Bergeronnette printanière"
## [41] "Cassenoix moucheté"          "Traquet motteux"
## [43] "Mésange charbonnière"        "Moineau domestique"
## [45] "Moineau friquet"             "Mésange noire"
## [47] "Moineau soulcie"             "Rougequeue noir"
## [49] "Verdier d'Europe"            "Pouillot de Bonelli"
## [51] "Pouillot véloce"             "Pie bavarde"
## [53] "Mésange boréale"             "Mésange nonnette"
## [55] "Accenteur alpin"             "Accenteur mouchet"
## [57] "Hirondelle de rochers"       "Bouvreuil pivoine"
## [59] "Roitelet à triple bandeau"   "Roitelet huppé"
## [61] "Tarier des prés"             "Tarier pâtre"
## [63] "Serin cini"                  "Sittelle torchepot"
## [65] "Tarin des aulnes"            "Étourneau sansonnet"
## [67] "Fauvette à tête noire"       "Fauvette des jardins"
## [69] "Fauvette grisette"           "Fauvette babillarde"
## [71] "Tichodrome échelette"        "Troglodyte mignon"
## [73] "Merle noir"                  "Grive musicienne"
## [75] "Grive litorne"               "Merle à plastron"
## [77] "Grive draine"
```

```
species_name="Accenteur alpin"
## Number of occurrences for the corresponding species
print(paste0(length(Passeriformes$species[Passeriformes$species == species_name]),
   " observations of ", species_name,
   " in the Ecrins National Park between 1994 et 2021"))
```

```
## [1] "1775 observations of Accenteur alpin in the Ecrins National Park between 1994 et 2021"
```

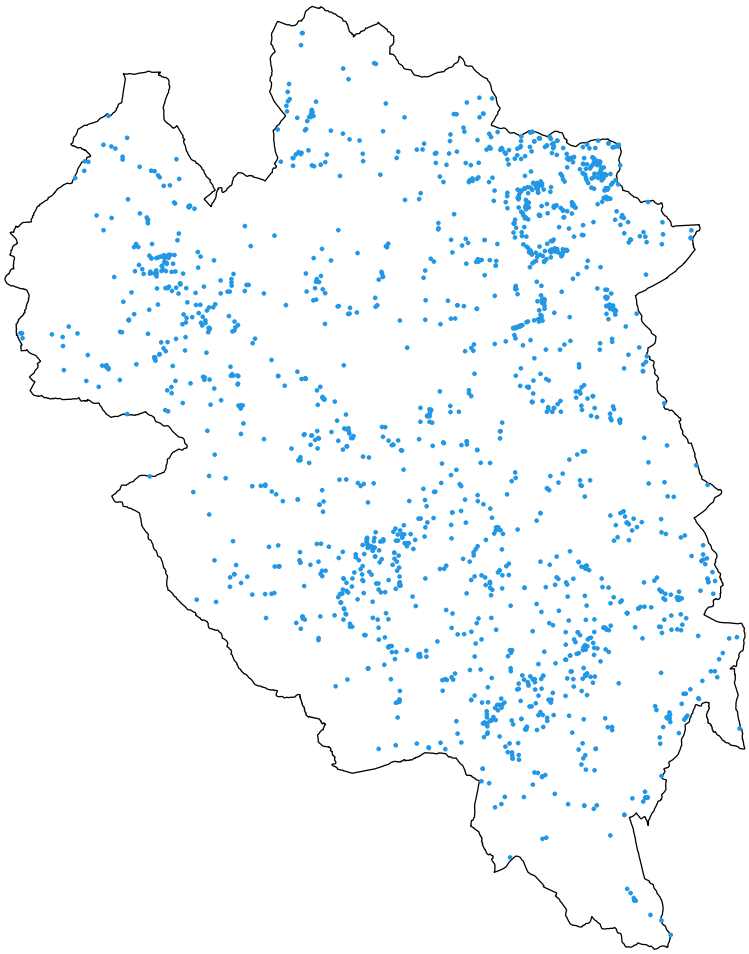
We generate a sf data.frame for the occurrences of the focal species and the Target-Group Background (TGB, used for spatial sampling bias correction when fitting the intensity). This data.frame also gathers 6 covariates for each point.

```
Passeriformes=st_as_sf(Passeriformes)
covarNames=c("Axis1","Axis2","Axis3","Axis4","Axis5","Axis6")

# Focal species occurrences
# add cell ID, and covariates for each
sp_dat_sf = Passeriformes%>%
  filter(species==species_name)%>%
  select(Month,Time_period,Year,geometry)%>%
  st_join(ENP_grid[c('id_cell',covarNames)],join=st_within)

# Target-Group background occurrences
TG_dat_sf = Passeriformes%>%
```

```
  select(Month,Time_period,Year,geometry)%>%
  st_join(ENP_grid[c('id_cell',covarNames)],join=st_within)
```

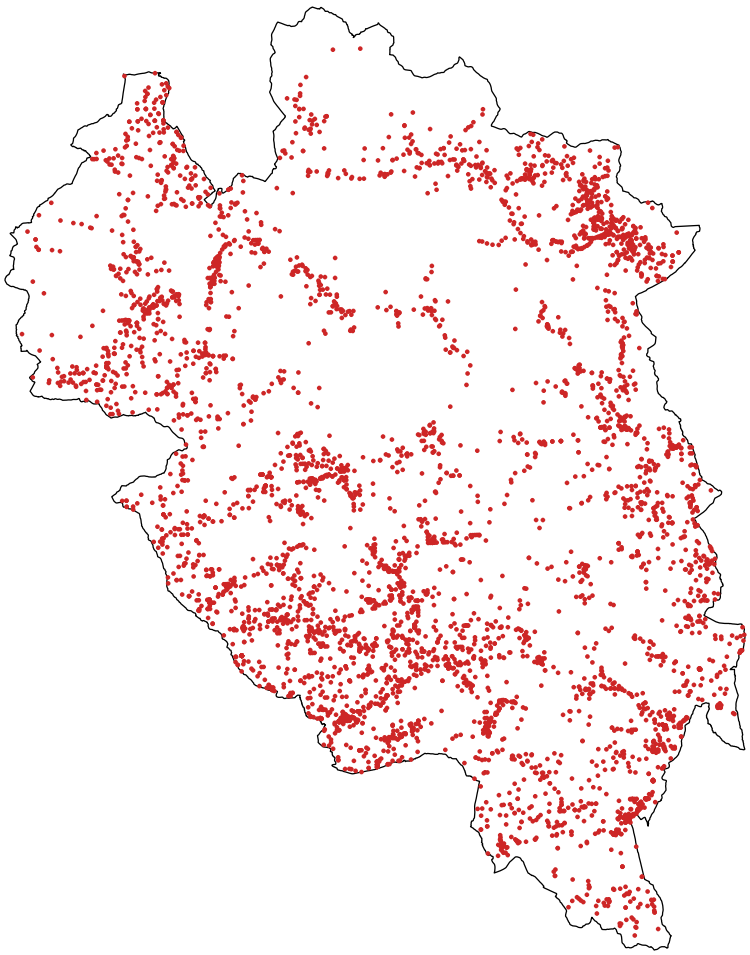
Here is what look like the study domain of the Park, and the points of the focal species.

```
  plot(ENP_domain)
  plot(sp_dat_sf,pch=16,col=4,add=T,cex=0.5)
```



We can also show a subset of the TGB points.

```
  plot(ENP_domain)
  plot(TG_dat_sf[sample(1:nrow(TG_dat_sf),5000),],
       pch=16,col='firebrick3',add=T,cex=0.5)
```



Fit a LGCP to the focal species occurrences (without the TG)

Model representation

1. Poisson Likelihood for Point Process

The model assumes a **log-Gaussian Cox process** where the presence of occurrences $y(s)$ at location s follows:

$$y(s) \sim \text{LGCP}(\lambda(s)E(s))$$

Where:

- $y(s)$ is the **occurrence** of an event at location s.
- $E(s)$ is an **exposure term** (often representing area or sampling effort).
- $\lambda(s)$ is the **intensity function** that describes the expected number of occurrences at location s.

2. Log-Linear Intensity Function

The **log-intensity function** is modeled as:

$$\log \lambda(s) = \beta_0 + \sum_{i=1}^{6} \beta_i \cdot \text{Axis}_i(s) + w(s)$$

where:

- **Intercept:** β_0 is the baseline intensity.
- **Environmental Covariates:** β_i are the regression coefficients for environmental variables $\text{Axis}_i(s)$ (e.g., habitat predictors).
- **Spatial Effect:** $w(s)$ is a spatially structured random effect.

Primary mesh of latent gaussian variables

We first build the spatial mesh for SPDE approximation. In that case we include the observation points' coordinates when building the primary mesh, ensuring that each observation point corresponds to a dual mesh cell (see below).

Why Should Observation Points Be Included in the Primary Mesh?

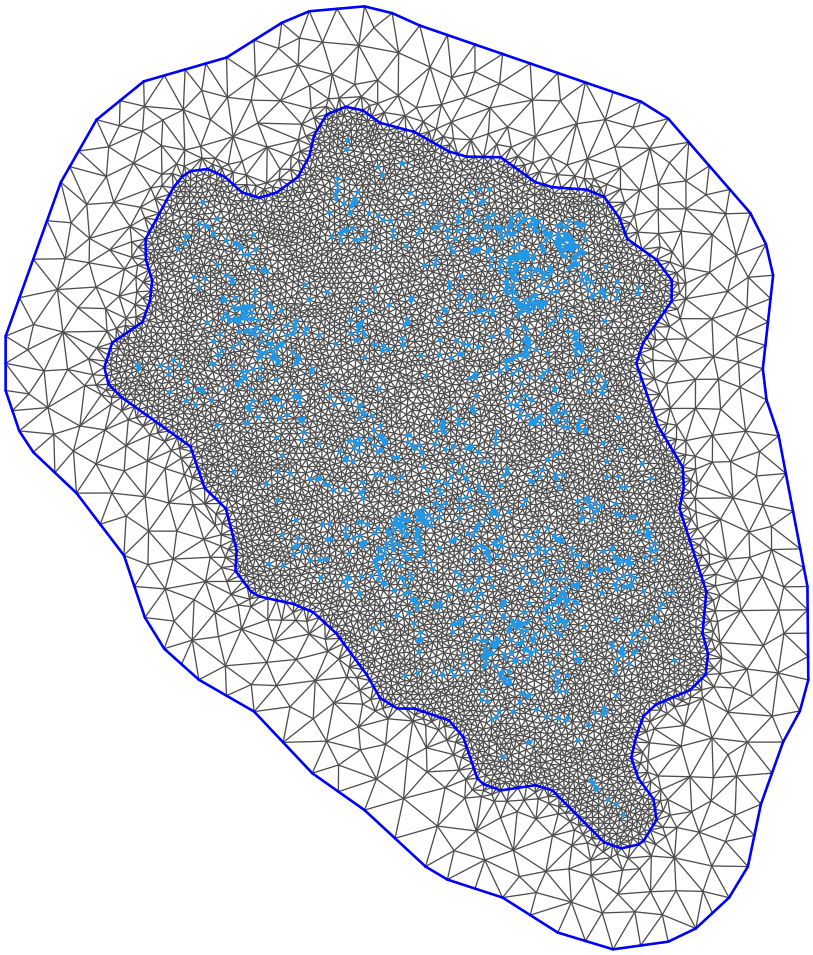
1. **Accurate Representation of the Intensity Function**
   - The **LGCP model** estimates the intensity function at mesh nodes, which is then integrated over the **dual mesh cells**.
   - If observation points are not included, they may fall into large, arbitrarily shaped cells, leading to a poor approximation of the intensity.
2. **Ensuring Each Observation Has a Corresponding Dual Cell**
   - The **dual mesh cells** are derived from the **primary mesh nodes**.
   - If an observation falls outside the **primary mesh nodes**, it might be poorly represented or assigned to an incorrect region.
   - By including observation locations in the **primary mesh**, you guarantee that each observation falls into a well-defined **dual cell**.
3. **Better Integration for Likelihood Approximation**
   - The LGCP likelihood relies on **integration over dual mesh cells**.
   - If observation points are not used in mesh construction, some may fall into **overly large or irregular cells**, leading to **poor numerical approximations** of the likelihood.

```
# - get coordinates observation points
grid_coords=st_coordinates(sp_dat_sf)
# Define mesh boundaries:
# - Inner boundary with slight concavity to follow data
bndint = inla.nonconvex.hull(grid_coords, convex = -0.05)
# - Outer boundary with stronger concavity for extension
bndext = inla.nonconvex.hull(grid_coords, convex = -0.2)

# Build triangular mesh:
# - max.edge: Maximum allowed triangle edge length (inner, outer zones)
# - cutoff: Minimum distance between mesh vertices
mesh = fm_mesh_2d_inla(grid_coords,
                       boundary = list(bndint, bndext),
```

```
                        max.edge = c(1000, 5000), cutoff = 500)
print(mesh$n) # Number of mesh vertices
```

```
## [1] 9062
```

```
plot(mesh) # the mesh !
points(st_coordinates(sp_dat_sf),col=4,pch=16,cex=0.5)
```



Dual mesh for integration areas

1. What is a Dual Mesh?

A **dual mesh** is a structure derived from a **primary mesh (triangulation)**. The primary mesh consists of nodes connected by edges forming triangular elements, while the **dual mesh is constructed by associating each triangular element with a corresponding polygonal cell**.

- The dual mesh is typically a **Voronoi tessellation (Dirichlet cells)** of the mesh nodes.
- Each **dual cell** surrounds a **mesh node** and contains all the spatial locations that are closer to that node than to any other node.

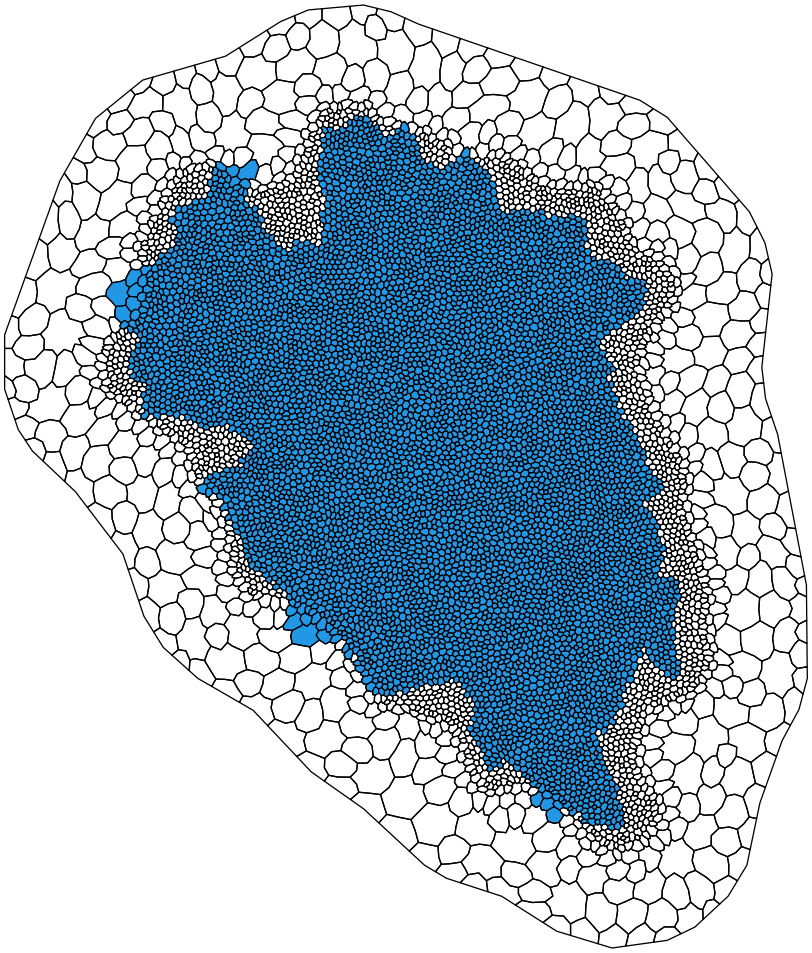
Thus, the dual mesh represents a partition of space, where each cell corresponds to an area of influence of a particular node in the original triangulated mesh.

2. Why Do We Need the Dual Mesh for LGCP in INLA?

When modeling a **Log-Gaussian Cox Process (LGCP)**, the intensity function of the point process is driven by an underlying **spatial Gaussian random field (GRF)**. The **SPDE-INLA** approach discretizes the domain using a **finite element method (FEM) mesh**, but likelihood computation requires integration over spatial regions. This is where the **dual mesh** becomes essential:

1. **Computing Integrals Over the Study Region**
   - The LGCP model requires integration of the intensity function over space.
   - The **primary mesh** is used to approximate the spatial process, but **the integration needs to be performed over non-overlapping regions**.
   - The **dual mesh** provides these non-overlapping integration regions.
2. **Assigning Areas to Mesh Nodes**
   - Each **node in the primary mesh** contributes to defining the spatial field.
   - The corresponding **dual mesh cell gives the area assigned to that node**.
   - This helps in approximating the integral of the **exponentiated Gaussian field (i.e., the intensity function of the LGCP)** over space.
3. **Avoiding Over- or Under-Representation of Areas**
   - Without the dual mesh, we might assign intensities incorrectly, especially in irregularly shaped or non-uniformly spaced meshes.
   - The dual mesh ensures that each node's contribution is weighted appropriately.

```r
# Create dual mesh (Voronoi polygons around mesh vertices)
dmesh <- book.mesh.dual(mesh)
dmesh = st_as_sf(dmesh)
st_crs(dmesh)=st_crs(ENP_domain)  # Set coordinate system

# Calculate integration weights:
# 1. Find which dual polygons intersect study domain
intersect_idx <- st_intersects(dmesh, ENP_domain, sparse = FALSE)
# 2. Compute intersection areas
intersections <- st_intersection(dmesh[intersect_idx, ], ENP_domain)
# 3. Create weight vector proportional to polygon area within domain
w <- numeric(nrow(dmesh))
w[intersect_idx] <- as.numeric(st_area(intersections))
#plot
plot(dmesh,col=(w>0)*4)
points(st_coordinates(sp_dat_sf),col=2,cex=0.5,pch=16)
```



Preparation of fixed effects

```
# Extract environmental covariates at mesh nodes:
# 1. Convert mesh vertices to spatial points
mesh_sf <- st_as_sf(data.frame(mesh$loc), coords = 1:2, crs = st_crs(ENP_domain))
# 2. Join with grid covariates using spatial join
mesh_sf <- st_join(mesh_sf, ENP_grid[, covarNames], join = st_within)
# 3. Identify rows where a covariate is NA
na_rows <- which(is.na(mesh_sf$Axis1))
# 4. Find the nearest neighbor and replace NA values
if (length(na_rows) > 0) {
  for (covar in covarNames) {
   # Find nearest neighbor index
```

```
    nearest_idx <- st_nearest_feature(mesh_sf[na_rows, ], ENP_grid)
    mesh_sf[na_rows, covar] <- ENP_grid[nearest_idx, covar, drop = TRUE]
  }
}

# Combine covariates from mesh nodes and observation points
# Intercept column
fixed_effects = data.frame(intercept = rep(1,mesh$n+nrow(sp_dat_sf)))
for(cov in covarNames){
   fixed_effects[,cov]=c(st_drop_geometry(mesh_sf)[,cov],
                         st_drop_geometry(sp_dat_sf)[,cov])
}
```

SPDE prior configuration

```
# Matérn covariance parameters:
nu = 1  # Smoothness parameter
d = 2    # Spatial dimension
alpha = nu + d/2  # SPDE parameterization

# Define PC priors for spatial field:
spde_sp = inla.spde2.pcmatern(
  mesh = mesh,
  alpha = alpha,
  prior.range = c(10000, 0.5),  # P(range < 10km) = 0.5
  prior.sigma = c(1, 0.5)       # P(sigma > 1) = 0.5
)
```

Projection matrix construction

1. Define Number of Mesh Vertices and Observations

We first extract the number of **mesh vertices** (`nv`) and the **number of observed presence points** (`n`).

```
nv = mesh$n  # Number of mesh vertices
n = nrow(sp_dat_sf)  # Number of observations
```

- `nv`: Represents the **number of latent random field locations** (mesh nodes).
- `n`: Represents the **number of species presence observations**.

2. Create the Response Vector (`y.pp`)

The response vector encodes **presence-absence** information for the LGCP model.

```
y.pp <- rep(0:1, c(nv, n))
```

- The first `nv` elements are **0** (representing integration points/background locations).
- The last `n` elements are **1** (representing observed presences).

3. Create the Exposure Vector (`e.pp`)

The exposure vector provides **weights** for background points and zeros for presence points.

```
e.pp <- c(w, rep(0, n))
```

- `w`: Contains **weights** for the integration points.

- `0` values are assigned to observed presences, as they do not need weights.

4. Create Projection Matrices for the Spatial Field

Identity Matrix for Mesh Nodes (`imat`) Each mesh node is treated as a **latent basis function**.

```
imat <- Diagonal(nv, rep(1, nv))  # Identity matrix for mesh nodes
```

Observation to Mesh Projection (`lmat`) Observations are projected onto the mesh using **barycentric interpolation**.

```
lmat <- inla.spde.make.A(mesh, st_coordinates(sp_dat_sf))
```

Combine Projection Matrices (`A.pp`) The final projection matrix links **background points** and **presence points** to the mesh.

```
A.pp <- rbind(imat, lmat)
```

Summary of the Data Structures

Component	Purpose	Why is it Needed?
`nv, n`	Number of mesh vertices and presence points	Defines problem dimensions
`y.pp`	Binary response vector (`0` = background, `1` = observed presence)	Required for LGCP likelihood
`e.pp`	Exposure vector (weights for integration points, `0` for presences)	Corrects for intensity scaling
`imat`	Identity matrix for mesh vertices	Defines latent spatial field basis
`lmat`	Projection matrix (maps observations to mesh)	Ensures proper spatial interpolation
`A.pp`	Combined projection matrix (`imat + lmat`)	Links spatial process to both background & presence points

Data Stacking

Data stacking in **INLA** is a technique used to efficiently structure data for modeling, particularly when using the **SPDE approach** for spatial modeling. The goal is to combine multiple components of the data (observations, latent effects, exposure weights) into a single structure that INLA can process effectively.

```
stk.pp <- inla.stack(
  data = list(y = y.pp, e = e.pp),  # Response + exposure
  A = list(1, A.pp),                # Projection matrices
  effects = list(
    fixed_effects,                  # Fixed effects covariates
    list(sp = 1:nv)                 # Spatial random effect
  ),
  tag = 'pp')
```

Explanation of Components:

1. `data = list(y = y.pp, e = e.pp)`

- `y.pp`: Response variable, combining background (`0`) and observed presences (`1`).
- `e.pp`: Exposure term, used to scale the likelihood appropriately.

2. `A = list(1, A.pp)`
   - Defines **projection matrices** that link the different data components.
   - `1`: Represents an identity matrix for fixed effects.
   - `A.pp`: Links presence locations to the mesh through spatial interpolation.
3. `effects = list(fixed_effects, list(sp = 1:nv))`
   - `fixed_effects`: Environmental covariates.
   - `list(sp = 1:nv)`: The **spatial random effect**, which is linked to the mesh nodes.
4. `tag = 'pp'`
   - Provides a tag for identifying this stack in INLA later.

Model Formula Definition

```
formula = y ~ -1 +                    # Remove intercept
  intercept +                         # Explicit intercept
  Axis1 + Axis2 + Axis3 +            # Environmental covariates
  Axis4 + Axis5 + Axis6 +
  f(sp, model = spde_sp)              # Spatial random effect
```

Explanation of Components:

1. `y ~ -1 +`
   - `-1` removes the default intercept (explicitly defining it below).
2. `intercept +`
   - Adds a manually defined intercept to the model.
3. `Axis1 + Axis2 + Axis3 + Axis4 + Axis5 + Axis6 +`
   - Represents **environmental covariates** influencing species presence.
4. `f(sp, model = spde_sp)`
   - Defines the **spatial random effect** modeled using the **SPDE approach**.

Model Fitting with INLA

```
fit <- inla(
  formula,
  E = inla.stack.data(stk.pp)$e,     # Exposure term
  data = inla.stack.data(stk.pp),     # Stacked data
  family = 'poisson',                # LGCP as Poisson process
  control.compute = list(            # Compute model metrics
    cpo=T, dic=T, mlik=T, waic=T,
    return.marginals.predictor = T,
    config = TRUE),
  control.predictor = list(
    A = inla.stack.A(stk.pp)),       # Projection matrix
  control.inla=list(
    strategy="adaptive",             # Optimization strategy
    int.strategy="eb"),
  num.threads = 30)
```

Explanation of Components:

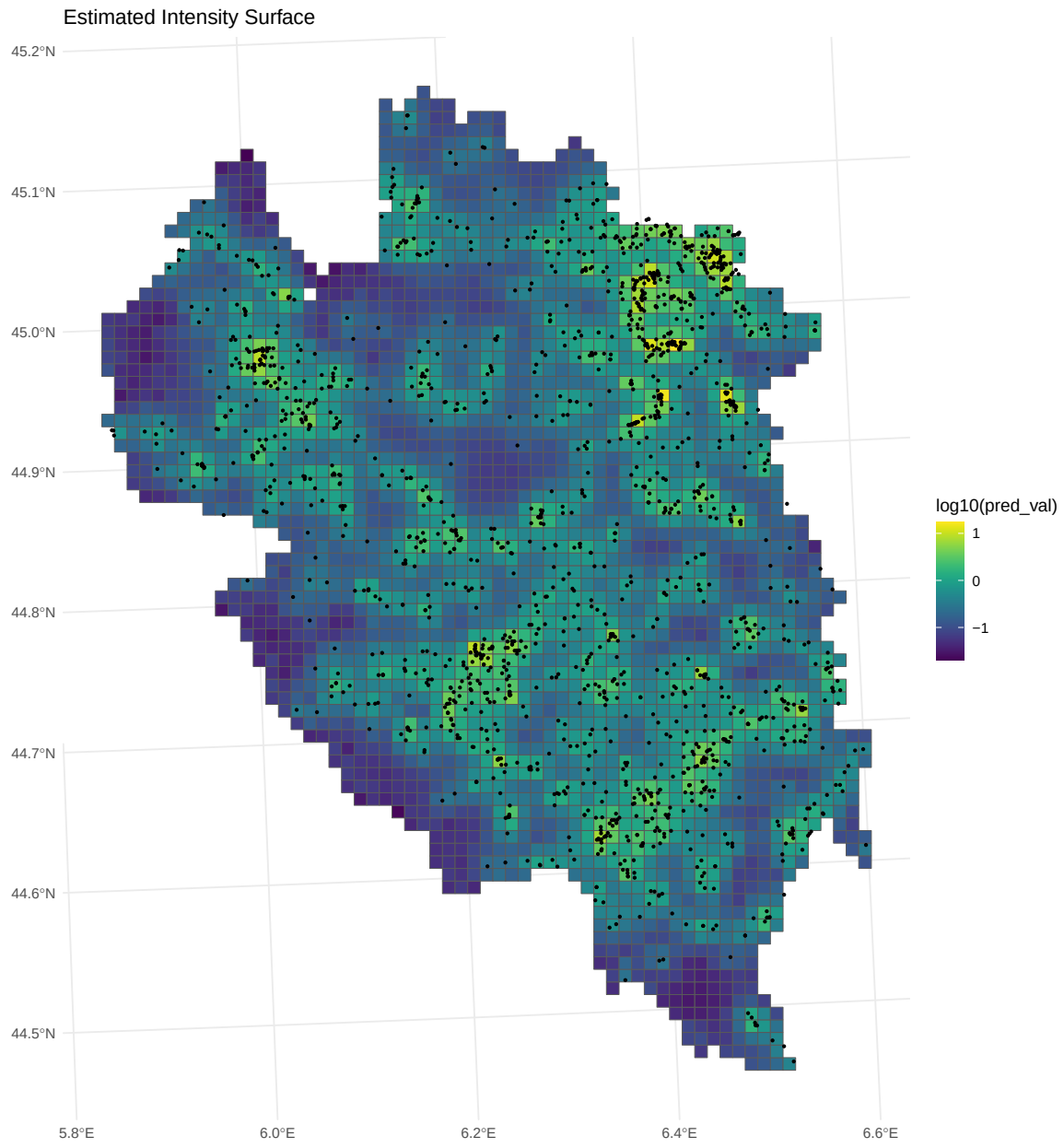
1. `formula`
   - Uses the previously defined **spatial LGCP model**.
2. `E = inla.stack.data(stk.pp)$e`

- Provides the **exposure term**, ensuring proper intensity scaling.

3. `data = inla.stack.data(stk.pp)`
   - Supplies the **stacked data** to INLA.
4. `family = 'poisson'`
   - LGCP is modeled as a **Poisson process**.
5. `control.compute`
   - Computes key model metrics:
     - **CPO** (Conditional Predictive Ordinate)
     - **DIC** (Deviance Information Criterion)
     - **MLIK** (Marginal Likelihood)
     - **WAIC** (Watanabe-Akaike Information Criterion)
     - **Return marginals for predictions**.
6. `control.predictor = list(A = inla.stack.A(stk.pp))`
   - Ensures correct **projection of predictions** using the stacked data.
7. `control.inla`
   - Uses an **adaptive strategy** for optimization.
   - **Empirical Bayes (EB)** integration strategy.
8. `num.threads = 10`
   - Utilizes **parallel computing** for efficiency.

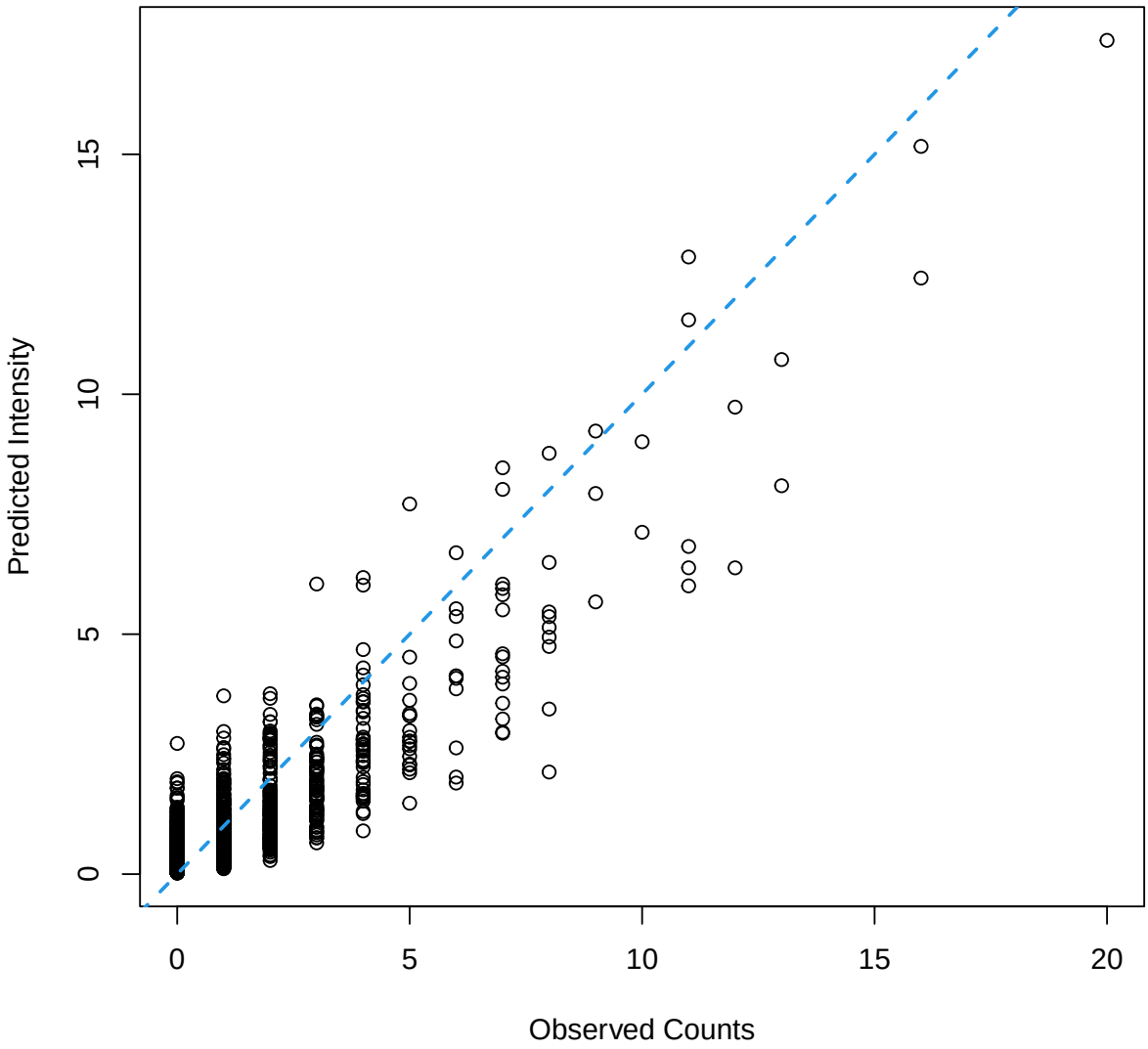
Prediction and Evaluation of LGCP Model in INLA

```
# Compute Area of Intersection Between Polygons and Study Domain
polygon_areas <- as.vector(st_area(st_intersection(ENP_grid, ENP_domain)))

# Project Model Predictions onto Centroids of Polygons
#Creates a projection object to interpolate model predictions at specified locations
projector <- inla.mesh.projector(mesh, st_coordinates(st_centroid(ENP_grid)))

# Extract Model Predictions at Polygon Centroids
#Interpolates the model's median predicted values (`0.5quant`) onto polygon centroids
predicted_intensity <- inla.mesh.project(projector,
   fit$summary.fitted.values$`0.5quant`[1:mesh$n])

# Compute Final Prediction Values
pred_val <- predicted_intensity * polygon_areas
pred1_val <- 1 - exp(-as.numeric(predicted_intensity * polygon_areas))
# - `pred_val`: Multiplies intensity predictions by polygon areas to scale predictions.
# - `pred1_val`: Converts intensity predictions into probabilities of presence using
#Poisson probability transformation.

# Visualize Predicted Intensity Surface
ggplot() +
  geom_sf(data=ENP_grid, aes(fill=log10(pred_val))) +
  geom_sf(data=sp_dat_sf,size=0.5) +
  scale_fill_viridis() +
  theme_minimal() +
  labs(title = "Estimated Intensity Surface")
```



Compare Predictions with Observed Counts

```
# Count the number of observed points in each polygon
point_in_poly <- st_intersects(ENP_grid, sp_dat_sf, sparse = FALSE)
point_counts <- rowSums(point_in_poly)

# Scatter plot comparison
plot(point_counts, pred_val, xlab = "Observed Counts", ylab = "Predicted Intensity")
abline(a=0, b=1, lwd=2, lty=2, col=4)  # Reference line
```




```
# Correlation coefficient
cor(point_counts, pred_val)
```

```
## [1] 0.891013
```

```
# Compute ROC Curve for Presence Prediction
test_roc <- roc(as.numeric(point_counts > 0), pred1_val)
print(paste0("AUC = ",test_roc$auc))
```

```
## [1] "AUC = 0.917986816121764"
```

```
print(paste0("TSS = ",max(test_roc$sensitivities + test_roc$specificities - 1)))
```

```
## [1] "TSS = 0.67388849646041"
```

```
# - `roc(as.numeric(point_counts > 0), pred1_val)`: Creates an ROC curve comparing
#presence/absence with predicted probabilities.
# - `test_roc$auc`: Computes Area Under the Curve (AUC).
# - `max(test_roc$sensitivities + test_roc$specificities - 1)`: Computes Youden's J statistic,
#a measure of optimal classification performance.
```

Fit a LGCP to the focal species occurrences corrected by the TG

Poisson Likelihood for LGCP

The **joint likelihood** is built from two stacked datasets:

$$Y_{\text{species},s} \sim \text{LGCP}(\lambda_s^{\text{species}} E_s)$$
$$Y_{\text{TG},s} \sim \text{LGCP}(\lambda_s^{\text{TG}} E_s)$$

Where:

$$\text{Species Model:} \quad \log \lambda_s^{\text{species}} = \beta_0 + \sum_{i=1}^{6} \beta_i \text{Axis}_i(s) + w_{\text{specific}}(s) + \beta_{\text{copy}} w_{\text{sharedTG}}(s)$$
$$\text{TG Model:} \quad \log \lambda_s^{\text{TG}} = w_{\text{sharedTG}}(s)$$

We define **three spatial effects**:

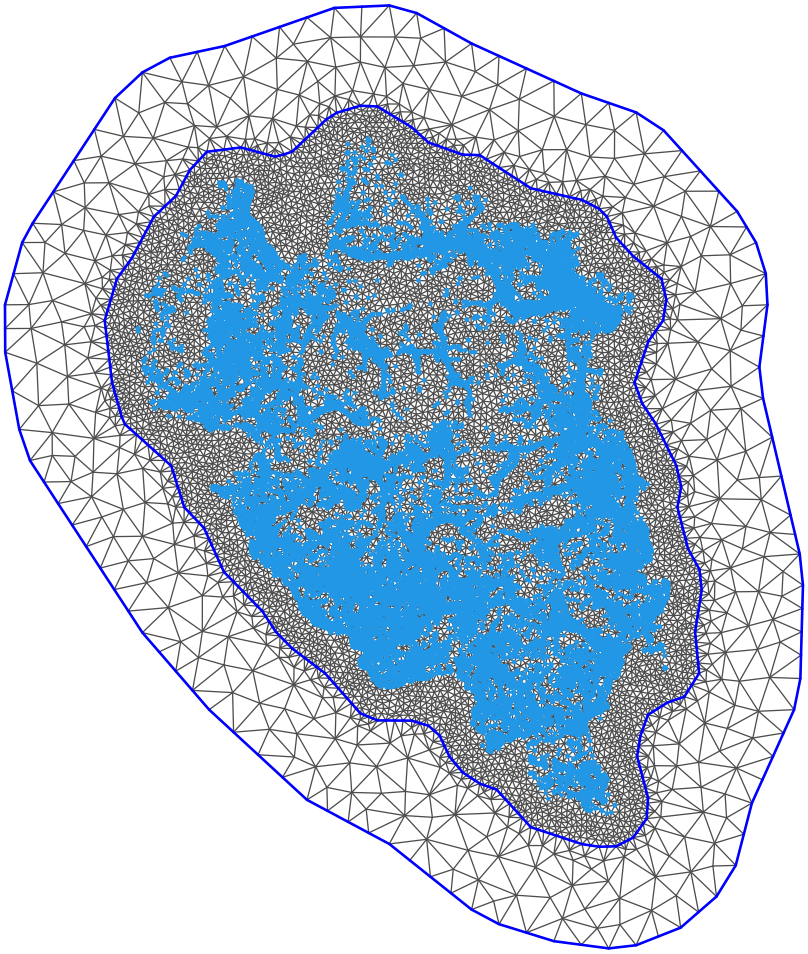
- $w_{\text{specific}}(s) \rightarrow$ Species-specific spatial effect.
- $w_{\text{sharedTG}}(s) \rightarrow$ Shared spatial structure linked to TG.
- $w_{\text{shared}}(s) = \beta_{\text{copy}} w_{\text{sharedTG}}(s) \rightarrow$ A **copied effect** derived from the TG effect with a fixed scaling factor. In our case β_{copy} is fixed to 1 as we want to model the relative intensity of the specific observations with respect to the TG.

Primary and dual meshes

```
# - get coordinates observation points
grid_coords=st_coordinates(TG_dat_sf)
# Define mesh boundaries:
# - Inner boundary with slight concavity to follow data
bndint = inla.nonconvex.hull(grid_coords, convex = -0.05)
# - Outer boundary with stronger concavity for extension
bndext = inla.nonconvex.hull(grid_coords, convex = -0.2)

# Build triangular mesh:
# - max.edge: Maximum allowed triangle edge length (inner, outer zones)
# - cutoff: Minimum distance between mesh vertices
meshTG = fm_mesh_2d_inla(grid_coords,
                         boundary = list(bndint, bndext),
                         max.edge = c(1000, 5000), cutoff = 500)
print(meshTG$n) # Number of mesh vertices
```

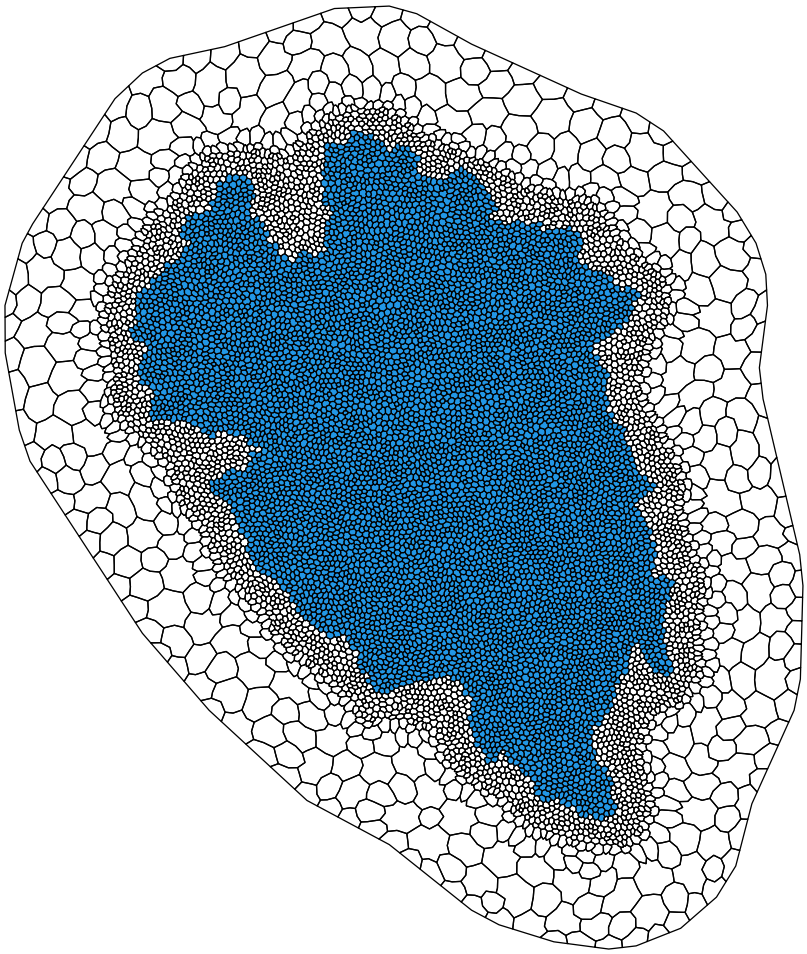
```
## [1] 10380
```

```
plot(meshTG) # the mesh !
points(st_coordinates(TG_dat_sf),col=4,pch=16,cex=0.5)
```



```
# Create dual mesh (Voronoi polygons around mesh vertices)
dmeshTG <- book.mesh.dual(meshTG)
dmeshTG = st_as_sf(dmeshTG)
st_crs(dmeshTG)=st_crs(ENP_domain)  # Set coordinate system

# Calculate integration weights:
# 1. Find which dual polygons intersect study domain
intersect_idx <- st_intersects(dmeshTG, ENP_domain, sparse = FALSE)
# 2. Compute intersection areas
intersections <- st_intersection(dmeshTG[intersect_idx, ], ENP_domain)
# 3. Create weight vector proportional to polygon area within domain
```

```
w <- numeric(nrow(dmeshTG))
w[intersect_idx] <- as.numeric(st_area(intersections))
#plot
plot(dmeshTG,col=(w>0)*4)
points(st_coordinates(TG_dat_sf),col=2,cex=0.5,pch=16)
```



Preparation of fixed effects

```
# Extract environmental covariates at mesh nodes:
# 1. Convert mesh vertices to spatial points
meshTG_sf <- st_as_sf(data.frame(meshTG$loc), coords = 1:2, crs = st_crs(ENP_domain))
# 2. Join with grid covariates using spatial join
meshTG_sf <- st_join(meshTG_sf, ENP_grid[, covarNames], join = st_within)
```

```r
# 3. Identify rows where a covariate is NA
na_rows <- which(is.na(meshTG_sf$Axis1))
# 4. Find the nearest neighbor and replace NA values
if (length(na_rows) > 0) {
  for (covar in covarNames) {
    # Find nearest neighbor index
    nearest_idx <- st_nearest_feature(meshTG_sf[na_rows, ], ENP_grid)
    meshTG_sf[na_rows, covar] <- ENP_grid[nearest_idx, covar, drop = TRUE]
  }
}

# Combine covariates from mesh nodes and observation points
# Intercept column
fixed_effects = data.frame(intercept = rep(1,meshTG$n+nrow(sp_dat_sf)))
for(cov in covarNames){
   fixed_effects[,cov]=c(st_drop_geometry(meshTG_sf)[,cov],
      st_drop_geometry(sp_dat_sf)[,cov])
}
```

SPDE prior configuration

```r
# Matérn covariance parameters:
nu = 1  # Smoothness parameter
d = 2    # Spatial dimension
alpha = nu + d/2  # SPDE parameterization

# Define PC priors for spatial field:
spde_sp = inla.spde2.pcmatern(
  mesh = meshTG,
  alpha = alpha,
  prior.range = c(10000, 0.5),  # P(range < 10km) = 0.5
  prior.sigma = c(1, 0.5)       # P(sigma > 1) = 0.5
)
```

Projection matrix construction

1. Define Number of Mesh Vertices and Observations

```r
nv = meshTG$n # Number of mesh vertices
nsp = nrow(sp_dat_sf) # Number of observations
nTG = nrow(TG_dat_sf) # Number of TG observations
```

2. Create the Response Vector (`y.pp`)

```r
ysp.pp <- rep(0:1, c(nv, nsp))
yTG.pp <- rep(0:1, c(nv, nTG))
```

3. Create the Exposure Vector (`e.pp`)

```r
esp.pp <- c(w, rep(0, nsp))
eTG.pp <- c(w, rep(0, nTG))
```

4. Create Projection Matrices for the Spatial Field

```
imat <- Diagonal(nv, rep(1, nv))  # Identity matrix for mesh nodes
lmatsp <- inla.spde.make.A(meshTG, st_coordinates(sp_dat_sf))
lmatTG <- inla.spde.make.A(meshTG, st_coordinates(TG_dat_sf))
Asp.pp <- rbind(imat, lmatsp)
ATG.pp <- rbind(imat, lmatTG)
```

Data Stacking

```
idx_specific_sp = inla.spde.make.index("specific_sp", n.spde = spde_sp$n.spde)
idx_shared_sp = inla.spde.make.index("shared_sp", n.spde = spde_sp$n.spde)
idx_shared_TG = inla.spde.make.index("shared_TG", n.spde = spde_sp$n.spde)

stksp.pp <- inla.stack(
  data = list(y = cbind(ysp.pp,NA), e = esp.pp),
  A = list(1, Asp.pp, Asp.pp),
  effects = list(fixed_effects,
                 idx_specific_sp,
                 idx_shared_sp),
  tag = 'sp')

stkTG.pp <- inla.stack(
  data = list(y = cbind(NA,yTG.pp), e = eTG.pp),
  A = list(ATG.pp),
  effects = list(idx_shared_TG),
  tag = 'TG')

stack_full = inla.stack(stksp.pp, stkTG.pp)
```

Model Formula Definition

```
formula = y ~ -1 +                  # Remove intercept
  intercept +                       # Explicit intercept
  Axis1 + Axis2 + Axis3 +           # Environmental covariates
  Axis4 + Axis5 + Axis6 +
  f(specific_sp, model = spde_sp) +
  f(shared_TG, model = spde_sp) +
  f(shared_sp, copy = "shared_TG", fixed = TRUE)
```

Model Fitting with INLA

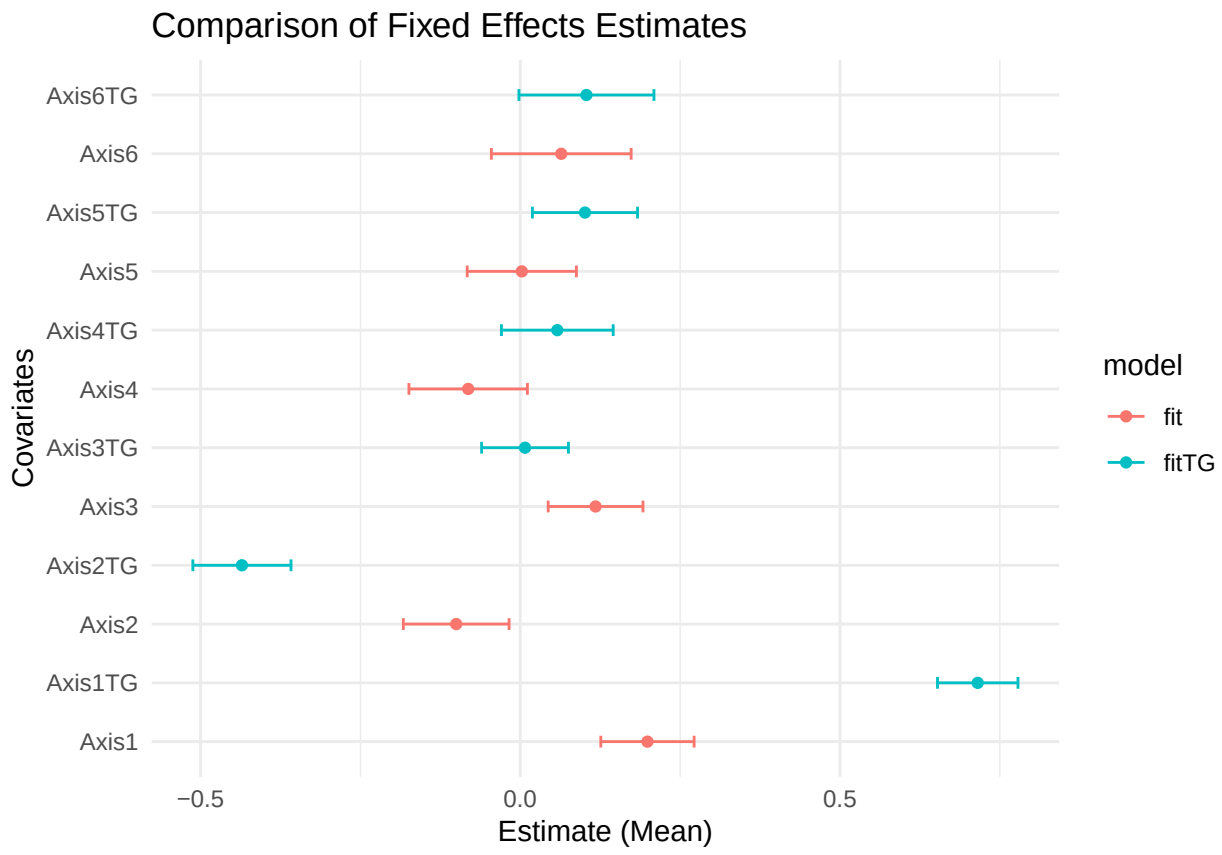
```
fitTG <- inla(
  formula,
  E = inla.stack.data(stack_full)$e,    # Exposure term
  data = inla.stack.data(stack_full),    # Stacked data
  family = c("poisson", "poisson"),                # LGCP as Poisson process
  control.compute = list(           # Compute model metrics
    cpo=T, dic=T, mlik=T, waic=T,
    return.marginals.predictor = T,
    config = TRUE),
  control.predictor = list(
```

```
    A = inla.stack.A(stack_full)),      # Projection matrix
  control.inla=list(
    strategy="adaptive",            # Optimization strategy
    int.strategy="eb"),
  num.threads = 30)
```

Predictions and estimates

Comparison of estimated effects between the two models

```
# Extract fixed effect estimates from both models
fixed_fit <- fit$summary.fixed %>% mutate(model = "fit")
fixed_fitTG <- fitTG$summary.fixed %>% mutate(model = "fitTG")

# Combine into one data frame
fixed_comparison <- bind_rows(fixed_fit[-1,], fixed_fitTG[-1,])
rownames(fixed_comparison) = c(paste0("Axis",1:6),paste0(paste0("Axis",1:6),"TG"))

ggplot(fixed_comparison, aes(x = rownames(fixed_comparison), y = mean, color = model)) +
  geom_point(position = position_dodge(width = 0.5)) +
  geom_errorbar(aes(ymin = `0.025quant`, ymax = `0.975quant`),
   width = 0.2, position = position_dodge(width = 0.5)) +
  theme_minimal() +
  labs(title = "Comparison of Fixed Effects Estimates",
       x = "Covariates", y = "Estimate (Mean)") +
  coord_flip()
```



Comparison of the spatial random effect

Projecting Spatial Effects from INLA Models

```
# model without TG
projector <- inla.mesh.projector(mesh, st_coordinates(st_centroid(ENP_grid)))
spatial_effect <- inla.mesh.project(projector, fit$summary.random$sp$`0.5quant`)

# model with TG
projector <- inla.mesh.projector(meshTG, st_coordinates(st_centroid(ENP_grid)))
spatial_effectsp <- inla.mesh.project(projector,
   fitTG$summary.random$specific_sp$`0.5quant`)
spatial_effectTG <- inla.mesh.project(projector,
   fitTG$summary.random$shared_TG$`0.5quant`)
```

- `inla.mesh.projector(mesh, coords)`: Projects the mesh nodes onto a grid of centroid locations.
- `inla.mesh.project(projector, fit$summary.random$sp$0.5quant)`: Extracts the median (`0.5quant`) of the spatial effect predictions.
- `meshTG` is used for the model **with TG correction**, while `mesh` is used for the **model without TG**.

Plot 1: Specific Spatial Effect Without TG

```
plot1 <- ggplot() +
  geom_sf(data=ENP_grid, aes(fill=spatial_effect)) +
  geom_sf(data=sp_dat_sf, size=0.5, color = "red", alpha = 1) +
  scale_fill_viridis() +
  theme_minimal() +
  labs(title = "Specific Spatial Effect Without TG")
```

Plot 2: Specific Spatial Effect With TG

```
plot2 <- ggplot() +
  geom_sf(data=ENP_grid, aes(fill=spatial_effectsp)) +
  geom_sf(data=sp_dat_sf, size=0.5, color = "red", alpha = 1) +
  scale_fill_viridis() +
  theme_minimal() +
  labs(title = "Specific Spatial Effect With TG")
```

Plot 3: TG Spatial Effect

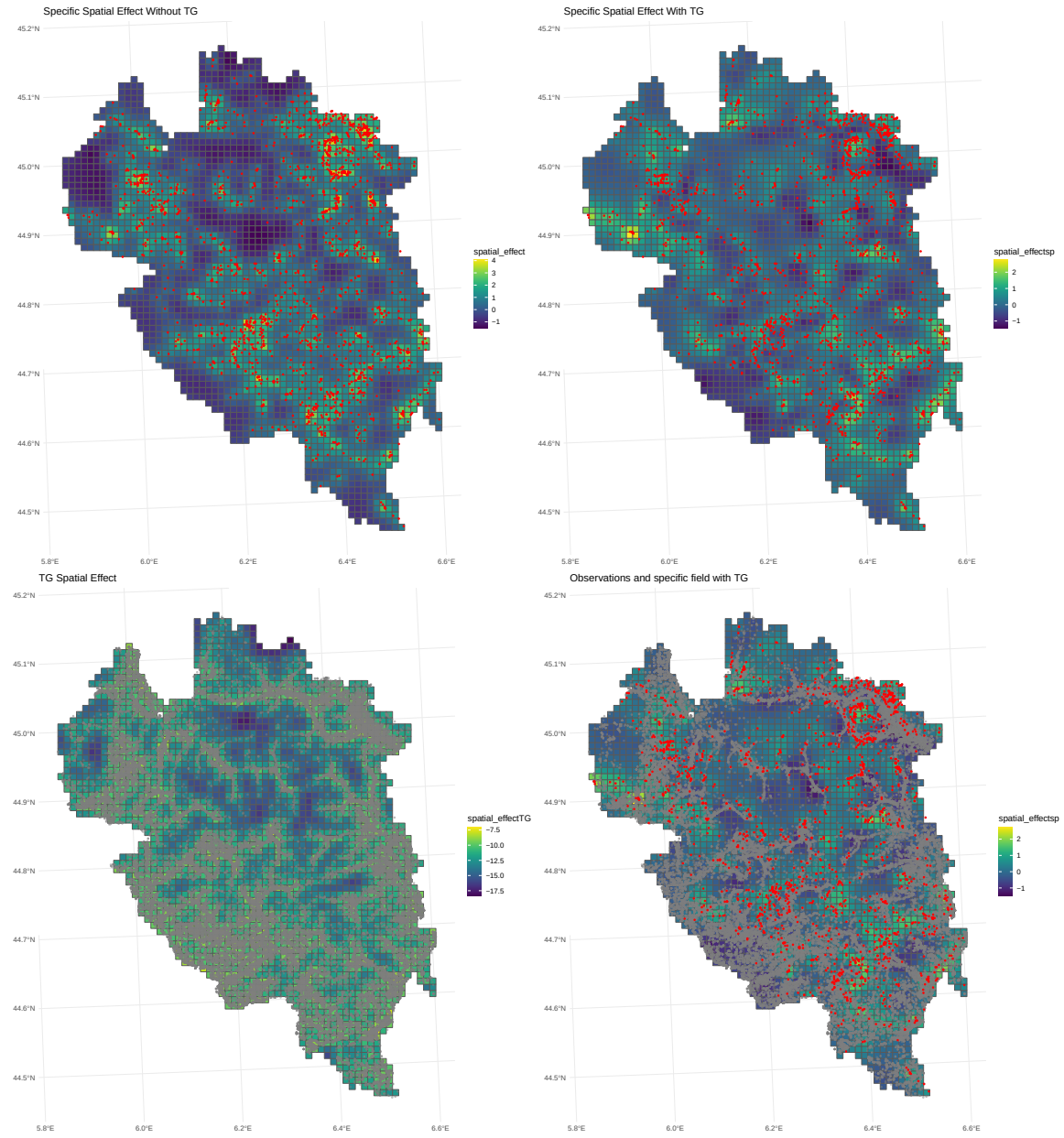
```
plot3 <- ggplot() +
  geom_sf(data=ENP_grid, aes(fill=spatial_effectTG)) +
  geom_sf(data=TG_dat_sf, size=0.5, color = "grey50", alpha = 0.5) +
  scale_fill_viridis() +
  theme_minimal() +
  labs(title = "TG Spatial Effect")
```

Plot 4: TG and specific observations with specific spatial fields

```
plot4 <- ggplot() +
  geom_sf(data=ENP_grid, aes(fill=spatial_effectsp)) +
  geom_sf(data=TG_dat_sf, size=0.5, color = "grey50", alpha = 0.5) +
  geom_sf(data=sp_dat_sf, size=0.5, color = "red", alpha = 1) +
  scale_fill_viridis() +
  theme_minimal() +
  labs(title = "Observations and specific field with TG")
```

Combine the Four Plots into One Layout

```
combined_plot <- (plot1 | plot2) / (plot3 | plot4)  # Arrange in a 2x2 grid
combined_plot
```



Predictions

Organization of `fit$summary.fitted.values` in INLA for bivariate model

This data frame contains posterior summaries of the fitted values:

Column Name	Description
`mean`	Posterior mean of the fitted value
`sd`	Posterior standard deviation
`0.025quant`	Lower bound of 95% credible interval
`0.5quant`	Median (posterior quantile 0.5)

Column Name	Description
`0.975quant`	Upper bound of 95% credible interval
`mode`	Posterior mode

The rows in `fit$summary.fitted.values` correspond to the response vector (`y`) as structured in `inla.stack()`. The organization follows this pattern:

Row Index	Data Type	Source Stack
1 → `nv`	Background points (integration)	`stksp.pp`
`nv+1` → `nv+nsp`	Presence points (species)	`stksp.pp`
`nv+nsp+1` → `nv+nsp+nv`	Background points (integration)	`stkTG.pp`
`nv+nsp+nv+1` → end	Presence points (TG dataset)	`stkTG.pp`

Where:

- **`nv`** = Number of mesh vertices (integration points)
- **`nsp`** = Number of species presence observations
- **`nTG`** = Number of TG presence observations

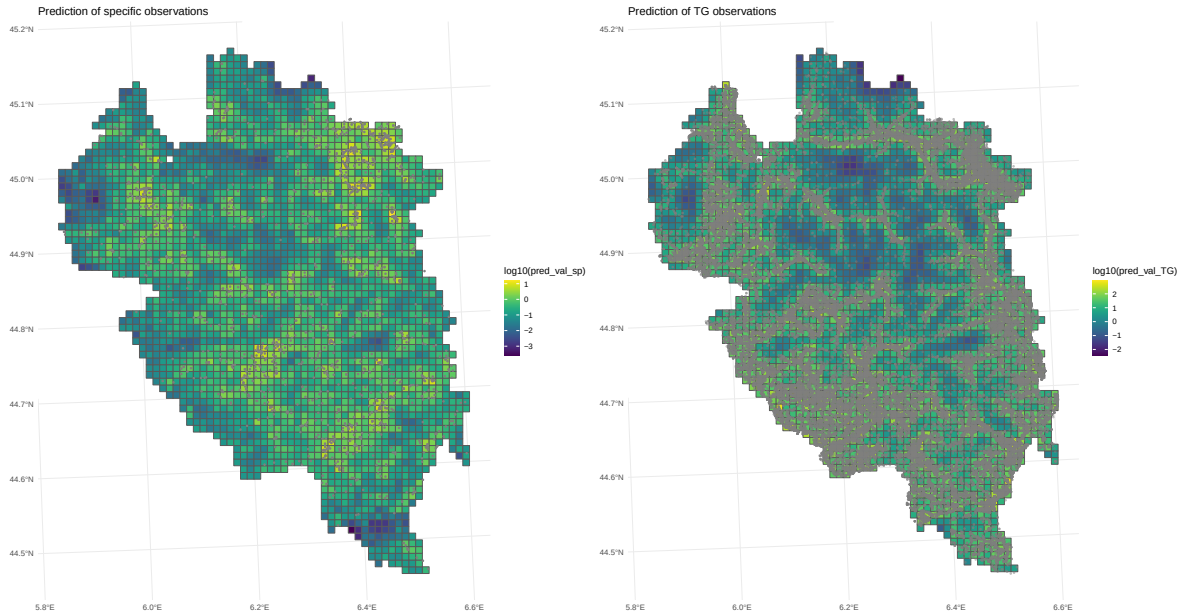
Compute predictions for specific and TG

```
# Compute Area of Intersection Between Polygons and Study Domain
polygon_areas <- as.vector(st_area(st_intersection(ENP_grid, ENP_domain)))

# Project Model Predictions onto Centroids of Polygons
#Creates a projection object to interpolate model predictions at specified locations
projector <- inla.mesh.projector(meshTG, st_coordinates(st_centroid(ENP_grid)))

# Extract Model Predictions at Polygon Centroids
#Interpolates the model's median predicted values (`0.5quant`) onto polygon centroids
predicted_intensity_sp <- inla.mesh.project(projector,
   fitTG$summary.fitted.values$`0.5quant`[1:nv])
predicted_intensity_TG <- inla.mesh.project(projector,
   fitTG$summary.fitted.values$`0.5quant`[(nv+nsp+1):(nv+nsp+nv)])

# Compute Final Prediction Values
pred_val_sp <- predicted_intensity_sp * polygon_areas
pred_val_TG <- predicted_intensity_TG * polygon_areas
```

Plots of predictions and observations

```
p1=ggplot() +
  geom_sf(data=ENP_grid, aes(fill=log10(pred_val_sp))) +
  geom_sf(data=sp_dat_sf,size=0.5, color = "grey50", alpha = 1) +
  scale_fill_viridis() +
  theme_minimal() +
  labs(title = "Prediction of specific observations")

p2=ggplot() +
  geom_sf(data=ENP_grid, aes(fill=log10(pred_val_TG))) +
  geom_sf(data=TG_dat_sf,size=0.5, color = "grey50", alpha = 0.5) +
```

```
  scale_fill_viridis() +
  theme_minimal() +
  labs(title = "Prediction of TG observations")

(p1 | p2)
```



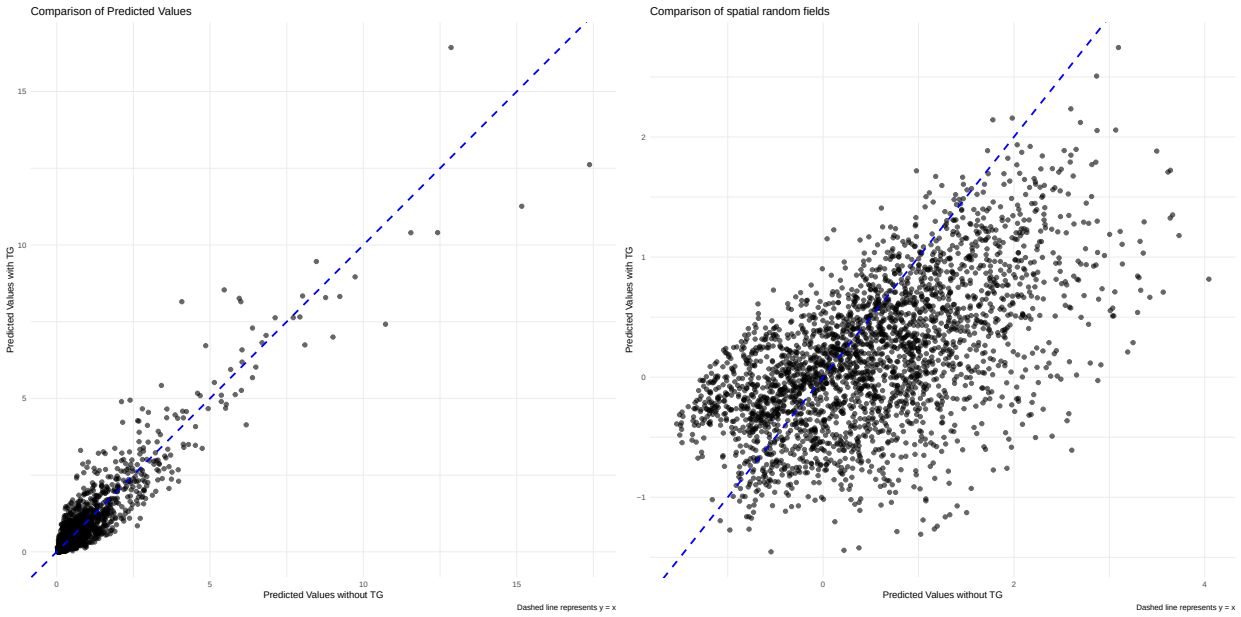
Comparison of predictions between simple and bivariate model

```
plot_data <- data.frame(pred_val = pred_val, pred_val_sp = pred_val_sp)
p1=ggplot(plot_data, aes(x = pred_val, y = pred_val_sp)) +
  geom_point(color = "black", alpha = 0.6, size = 2) +  # Scatter points
  geom_abline(intercept = 0, slope = 1, color = "blue",
   linetype = "dashed", linewidth = 1) +
  theme_minimal() +
  labs(title = "Comparison of Predicted Values",
       x = "Predicted Values without TG",
       y = "Predicted Values with TG",
       caption = "Dashed line represents y = x")
```

Comparison of the predicted specific spatial effects

```
plot_data <- data.frame(spatial_effect = spatial_effect, spatial_effect_sp = spatial_effectsp)
p2=ggplot(plot_data, aes(x = spatial_effect, y = spatial_effect_sp)) +
  geom_point(color = "black", alpha = 0.6, size = 2) +  # Scatter points
  geom_abline(intercept = 0, slope = 1, color = "blue",
   linetype = "dashed", linewidth = 1) +
  theme_minimal() +
  labs(title = "Comparison of spatial random fields",
       x = "Predicted Values without TG",
       y = "Predicted Values with TG",
       caption = "Dashed line represents y = x")
```

```
(p1 | p2)
```


Comparison of Predicted Values
Predicted Values with TG
Predicted Values without TG
Dashed line represents y = x
Comparison of spatial random fields
Predicted Values with TG
Predicted Values without TG
Dashed line represents y = x

Spatiotemporal models with INLA

Thomas Opitz, Christophe Botella, Pascal Monestiez, Julien Papaïx

2025-03-12

Load R packages and dataset

Our dataset notably contains:

- "ENP_domain": sf polygon delimitating the Écrins Park study area.
- "ENP_grid": sf data.frame of 500m cells inside the study area with various environmental covariates
- "list_of_species": Data.frame containing information for each bird species found in the Park
- "Passeriformes": Data.frame of bird species occurrences

```
dir="/mnt/c/Users/jpapaix/Documents/modelisation/atelierINLA2025/bon/"
setwd(dir)
#### Libraries ####
source("inlabookfunctions.R")
## Spatial data packages
library(sf) # For generic spatial data manipulation
## Data manipulation packages
library(tidyverse) # For generic data manipulation and visualization
library(viridis)
library(pROC)

## INLA
library(INLA) # For Bayesian inference using INLA
library(fmesher) # For building meshes
## Dataset
load(paste0(dir,"Data_for_modeling_fixed.RData"))
```

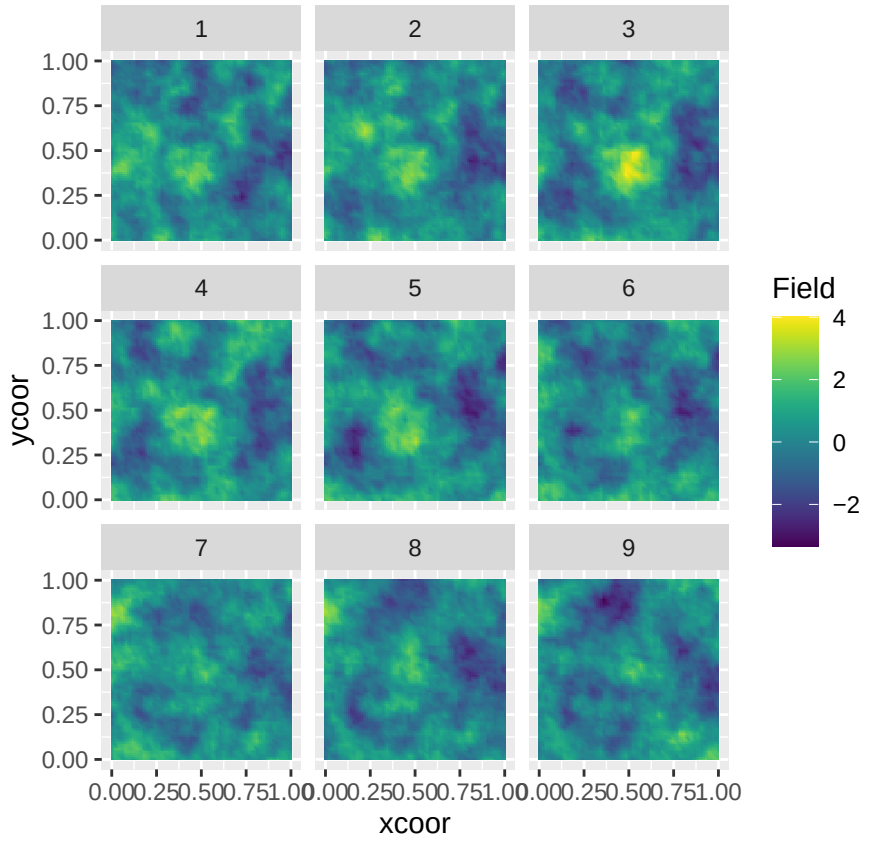
Simulation and visualisation of a Gaussian space-time field

To better understand spatiotemporal models, we first simulate a space-time Gaussian random field using the Matérn covariance function in space (SPDE model) and a first-order temporal autocorrelation with autocorrelation coefficient $\rho \in [-1, 1]$. The temporal dependence structure is defined as an `AR1` *group model* for the time-replicated spatial fields. The model is simulated on the unit square and has the following structure:

$$\begin{aligned} \varepsilon_t(s) &\overset{ind.}{\sim} \text{Matérn-SPDE}(\text{range}, \text{sigma}), \quad t = 1, 2, ... \\ W(s, 1) &= \varepsilon_1(s) \\ W(s, t+1) &= \rho W(s, t) + \sqrt{1-\rho^2}\varepsilon_{t+1}(s), \quad t = 1, 2, ... \end{aligned}$$

```
n_time = 9
rho = .8
# Define spatial domain and mesh
loc <- expand.grid(seq(0, 1, length.out = 100), seq(0, 1, length.out = 100))
mesh <- inla.mesh.2d(loc = loc, max.edge = c(0.05, 0.1), cutoff = 0.02)
```

```r
# Define SPDE model (range and sigma parameters are not yet important)
spde <- inla.spde2.pcmatern(mesh = mesh, alpha = 2, prior.range = c(1, NA),
    prior.sigma = c(1, NA))
# Extract precision matrix for spatial field
Q_spatial <- inla.spde2.precision(spde, theta = log(c(.25, 1)))
eps <- inla.qsample(n = n_time, Q_spatial)
proj <- fm_evaluator(mesh, loc = as.matrix(loc))
x <- cbind(fm_evaluate(proj, eps[,1]))
for(k in 2:n_time){
  x <- cbind(x, rho*x[,k-1] + sqrt(1-rho^2) * fm_evaluate(proj,  eps[,k]))
}

df = data.frame(Field = as.numeric(x), xcoor = rep(loc[,1], n_time),
    ycoor = rep(loc[,2], n_time), tcoor = rep(1:n_time, each = nrow(loc)))
pl <- ggplot(df) + coord_fixed()
pl <- pl + geom_tile(aes(x = xcoor, y = ycoor, fill = Field)) + facet_wrap(~tcoor)
pl <- pl + scale_fill_viridis()
print(pl)
```



Prepare species and background occurrences

We select a focal species in the list of bird species found in the Park.

```r
print(list_of_species$Name_fr)
```

```
##  [1] "Sizerin flammé"              "Rousserolle verderolle"
```

```
##  [3] "Orite à longue queue"        "Alouette des champs"
##  [5] "Pipit spioncelle"            "Pipit des arbres"
##  [7] "Jaseur boréal"               "Chardonneret élégant"
##  [9] "Venturon montagnard"         "Grimpereau des jardins"
## [11] "Grimpereau des bois"         "Rougequeue à front blanc"
## [13] "Cincle plongeur"             "Grosbec casse-noyaux"
## [15] "Corneille noire"             "Corneille mantelée"
## [17] "Choucas des tours"           "Mésange bleue"
## [19] "Hirondelle de fenêtre"       "Bruant fou"
## [21] "Bruant zizi"                 "Bruant jaune"
## [23] "Bruant ortolan"              "Rougegorge familier"
## [25] "Gobemouche noir"             "Pinson des arbres"
## [27] "Pinson du nord"              "Geai des chênes"
## [29] "Hirondelle rustique"         "Pie-grièche écorcheur"
## [31] "Linotte mélodieuse"          "Mésange huppée"
## [33] "Bec-croisé des sapins"       "Alouette lulu"
## [35] "Rossignol philomèle"         "Monticole de roche"
## [37] "Niverolle alpine"            "Bergeronnette grise"
## [39] "Bergeronnette des ruisseaux" "Bergeronnette printanière"
## [41] "Cassenoix moucheté"          "Traquet motteux"
## [43] "Mésange charbonnière"        "Moineau domestique"
## [45] "Moineau friquet"             "Mésange noire"
## [47] "Moineau soulcie"             "Rougequeue noir"
## [49] "Verdier d'Europe"            "Pouillot de Bonelli"
## [51] "Pouillot véloce"             "Pie bavarde"
## [53] "Mésange boréale"             "Mésange nonnette"
## [55] "Accenteur alpin"             "Accenteur mouchet"
## [57] "Hirondelle de rochers"       "Bouvreuil pivoine"
## [59] "Roitelet à triple bandeau"   "Roitelet huppé"
## [61] "Tarier des prés"             "Tarier pâtre"
## [63] "Serin cini"                  "Sittelle torchepot"
## [65] "Tarin des aulnes"            "Étourneau sansonnet"
## [67] "Fauvette à tête noire"       "Fauvette des jardins"
## [69] "Fauvette grisette"           "Fauvette babillarde"
## [71] "Tichodrome échelette"        "Troglodyte mignon"
## [73] "Merle noir"                  "Grive musicienne"
## [75] "Grive litorne"               "Merle à plastron"
## [77] "Grive draine"
```

```
species_name="Accenteur alpin"
## Number of occurrences for the corresponding species
print(paste0(length(Passeriformes$species[Passeriformes$species == species_name]),
    " observations of ", species_name, " in the Ecrins National Park between 1994 et 2021"))
```

```
## [1] "1775 observations of Accenteur alpin in the Ecrins National Park between 1994 et 2021"
```

We generate a sf data.frame for the occurrences of the focal species and the Target-Group Background (TGB, used for spatial or spatiotemporal sampling bias correction when fitting the intensity). This data.frame also gathers 6 covariates for each point.

```
Passeriformes=st_as_sf(Passeriformes)
covarNames=c("Axis1","Axis2","Axis3","Axis4","Axis5","Axis6")

# Focal species occurrences
# add cell ID, and covariates for each
sp_dat_sf = Passeriformes%>%
```

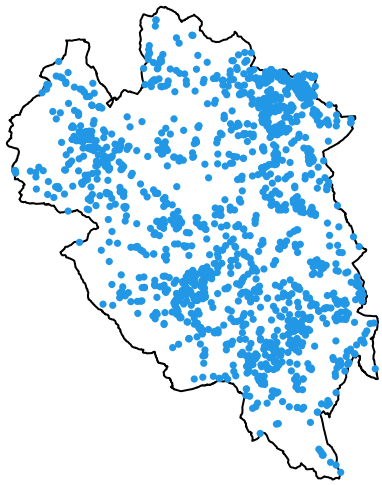
```
  filter(species==species_name)%>%
  select(Month,Time_period,Year,geometry)%>%
  st_join(ENP_grid[c('id_cell',covarNames)],join=st_within)

# Target-Group background occurrences
TG_dat_sf = Passeriformes%>%
  select(Month,Time_period,Year,geometry)%>%
  st_join(ENP_grid[c('id_cell',covarNames)],join=st_within)
```

We plot spatial maps of the study domain of the Park, including the points of the focal species, first for all points, then by time interval. We further define a group index *groups_time* (running from 1 to the number of time steps)

```
groups_time = 1:length(unique(sp_dat_sf$Time_period))
groups_time
```

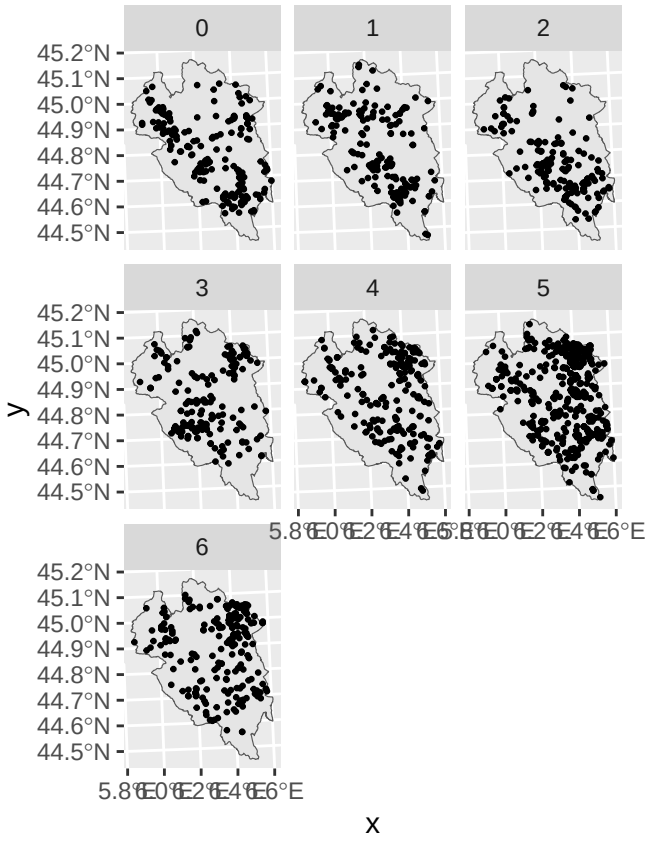
```
## [1] 1 2 3 4 5 6 7
```

```
plot(ENP_domain)
plot(sp_dat_sf,pch=16,col=4,add=T,cex=0.5, main = "Spatial")
```



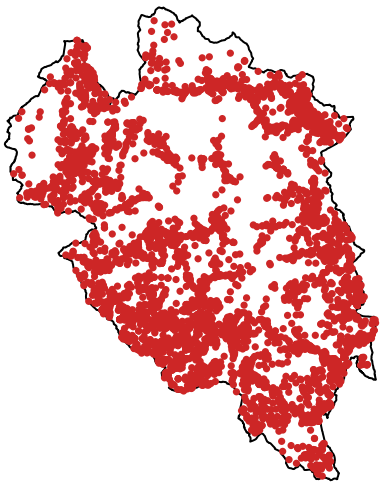
```
#for(k in groups_time){
#  data_k = sp_dat_sf[sp_dat_sf$Time_period == k,]
#  plot(ENP_domain)
#  plot(data_k,pch=16,col=4,add=T,cex=0.5, main = paste0("Time period ", k))
#}
xyt = cbind(st_coordinates(sp_dat_sf$geometry), sp_dat_sf$Time_period)
```

```
xyt = data.frame(xyt)
names(xyt) = c("x", "y", "time")
pl = ggplot(xyt) + geom_sf(data = ENP_domain)
pl = pl + geom_point(aes(x = x, y = y), size = .5) + facet_wrap(~time)
print(pl)
```

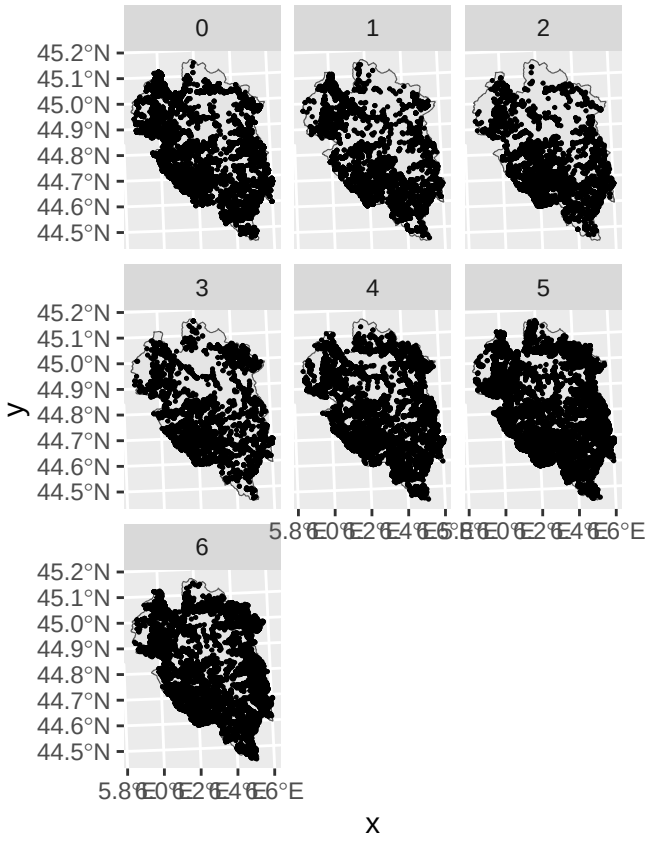


We can also show a subset of the larget set of TGB points, first aggregated over time,then by time period.

```
plot(ENP_domain)
plot(TG_dat_sf[sample(1:nrow(TG_dat_sf),5000),],
     pch=16,col='firebrick3',add=T,cex=0.5)
```



```
xyt = cbind(st_coordinates(TG_dat_sf$geometry), TG_dat_sf$Time_period)
xyt = data.frame(xyt)
names(xyt) = c("x", "y", "time")
pl = ggplot(xyt) + geom_sf(data = ENP_domain)
pl = pl + geom_point(aes(x = x, y = y), size = .25) + facet_wrap(~time)
print(pl)
```



Fit a LGCP to the focal species occurrences (without the TG)

The model we fit has the following structure on the study area:

$$
\begin{aligned}
\{s_1, \ldots, s_n\} &\sim \text{LGCP}(\{\lambda(s,t),\ s \in D,\, t = 1, \ldots, 7\}) \\
\log \lambda(s,t) \mid W(s,t) &= \beta_0 + \sum_{i=1}^{6} \beta_k z_k(s) + W(s,t) \\
\varepsilon_t(s) &\overset{ind.}{\sim} \text{Matérn-SPDE(range)}, \quad t = 1, 2, ... \\
W(s,1) &= \varepsilon_1(s) \\
W(s,t+1) &= \rho W(s,t) + \sqrt{1-\rho^2}\varepsilon_{t+1}(s), \quad t = 1, 2, ...
\end{aligned}
$$

Primary mesh of latent gaussian variables

We first build the spatial mesh for the SPDE approximation. Here, we include the observation points' coordinates when building the primary mesh, ensuring that each observation point corresponds to a dual mesh cell (see below). This is not mandatory, but can improve the model fit and numerical stability. When the number of points very large, this approach may become numerically infeasible, but we can tune the mesh to avoid having extremely small dual mesh cells, for example by using the *cutoff* parameter in the code below.

Why Should Observation Points Be Included in the Primary Mesh?

1. **Accurate Representation of the Intensity Function**

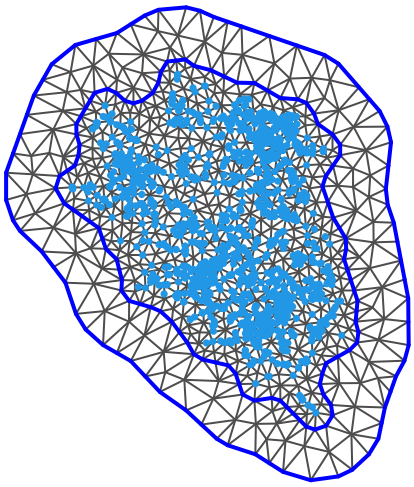
- The **LGCP model** estimates the intensity function at mesh nodes, which is then integrated over the **dual mesh cells**.
- If observation points are not included, they may fall into large, arbitrarily shaped cells, leading to a poor approximation of the intensity.

2. **Ensuring Each Observation Has a Corresponding Dual Cell**
   - The **dual mesh cells** are derived from the **primary mesh nodes**.
   - If an observation falls outside the **primary mesh nodes**, it might be poorly represented or assigned to an incorrect region.
   - By including observation locations in the **primary mesh**, you guarantee that each observation falls into a well-defined **dual cell**.
3. **Avoiding Numerical Instabilities and Biases**
   - Ignoring observation locations during mesh construction may cause **distorted or oversmoothed** spatial structures, particularly in regions with high observation densities.
   - This can lead to **biased intensity estimates** where many observations exist but the mesh lacks refinement.
4. **Better Integration for Likelihood Approximation**
   - The LGCP likelihood relies on **integration over dual mesh cells**.
   - If observation points are not used in mesh construction, some may fall into **overly large or irregular cells**, leading to **poor numerical approximations** of the likelihood.

```
# - get coordinates observation points
grid_coords=st_coordinates(sp_dat_sf)
# Define mesh boundaries:
# - Inner boundary with slight concavity to follow data
bndint = inla.nonconvex.hull(grid_coords, convex = -0.05)
# - Outer boundary with stronger concavity for extension
bndext = inla.nonconvex.hull(grid_coords, convex = -0.2)

# Build triangular mesh:
# - max.edge: Maximum allowed triangle edge length (inner, outer zones)
# - cutoff: Minimum distance between mesh vertices
mesh = fm_mesh_2d_inla(grid_coords,
                       boundary = list(bndint, bndext),
                       max.edge = c(2500, 8000), cutoff = 2000) #cutoff = 1000 for finer mesh
print(mesh$n) # Number of mesh vertices
```

```
## [1] 734
```

```
plot(mesh) # the mesh !
points(st_coordinates(sp_dat_sf),col=4,pch=16,cex=0.5)
```



Spatial dual mesh for integration areas

Here, we calculate the integration weights that we have to specify as *exposure* values for the Poisson likelihood.

1. What is a Dual Mesh?

A **dual mesh** is a structure derived from a **primary mesh (triangulation)**. The primary mesh consists of nodes connected by edges forming triangular elements, while the **dual mesh is constructed by associating each triangular element with a corresponding polygonal cell**.

- The dual mesh is typically a **Voronoi tessellation (Dirichlet cells)** of the mesh nodes.
- Each **dual cell** surrounds a **mesh node** and contains all the spatial locations that are closer to that node than to any other node.

Thus, the dual mesh represents a partition of space, where each cell corresponds to an area of influence of a particular node in the original triangulated mesh.

2. Why Do We Need the Dual Mesh for LGCP in INLA?

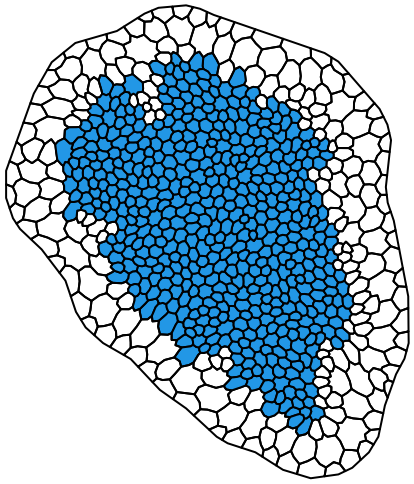
When modeling a **Log-Gaussian Cox Process (LGCP)**, the intensity function of the point process is driven by an underlying **spatial Gaussian random field (GRF)**. The **SPDE-INLA** approach discretizes the domain using a **finite element method (FEM) mesh**, but likelihood computation requires integration over spatial regions. This is where the **dual mesh** becomes essential:

1. **Computing Integrals Over the Study Region**
   - The LGCP model requires integration of the intensity function over space.
   - The **primary mesh** is used to approximate the spatial process, but **the integration needs to be performed over non-overlapping regions**.

- The **dual mesh** provides these non-overlapping integration regions.

2. **Assigning Areas to Mesh Nodes**
   - Each **node in the primary mesh** contributes to defining the spatial field.
   - The corresponding **dual mesh cell gives the area assigned to that node**.
   - This helps in approximating the integral of the **exponentiated Gaussian field (i.e., the intensity function of the LGCP)** over space.
3. **Avoiding Over- or Under-Representation of Areas**
   - Without the dual mesh, we might assign intensities incorrectly, especially in irregularly shaped or non-uniformly spaced meshes.
   - The dual mesh ensures that each node's contribution is weighted appropriately.
4. **Improving Computational Efficiency**
   - Instead of integrating over the fine-scale primary mesh, **integration over dual cells provides a computationally feasible approximation**.
   - This is particularly useful when using INLA, which requires efficient integration methods to compute the approximate posterior.

```r
# Create dual mesh (Voronoi polygons around mesh vertices)
dmesh <- book.mesh.dual(mesh)
dmesh = st_as_sf(dmesh)
st_crs(dmesh)=st_crs(ENP_domain)  # Set coordinate system

# Calculate integration weights:
# 1. Find which dual polygons intersect study domain
intersect_idx <- st_intersects(dmesh, ENP_domain, sparse = FALSE)
# 2. Compute intersection areas
intersections <- st_intersection(dmesh[intersect_idx, ], ENP_domain)
# 3. Create weight vector proportional to polygon area within domain
w <- numeric(nrow(dmesh))
w[intersect_idx] <- as.numeric(st_area(intersections))
#plot
plot(dmesh,col=(w>0)*4)
points(st_coordinates(sp_dat_sf),col=2,cex=0.5,pch=16)
```



Preparation of fixed effects

We prepare a data frame containing the fixed effects as its columns, including a column with only ones to model the intercept. Here, environmental covariates do not change over time, such that we can work only with the spatial coordinates to extract the covariates for the mesh nodes. We use the number of time steps (or intervals), `n_time`, to replicate the intercept and spatial covariates for each time step.

```
# Extract environmental covariates at mesh nodes:
# 1. Convert mesh vertices to spatial points
mesh_sf <- st_as_sf(data.frame(mesh$loc), coords = 1:2, crs = st_crs(ENP_domain))
# 2. Join with grid covariates using spatial join
mesh_sf <- st_join(mesh_sf, ENP_grid[, covarNames], join = st_within)
# 3. Identify rows where a covariate is NA
na_rows <- which(is.na(mesh_sf$Axis1))
# 4. Find the nearest neighbor and replace NA values
if (length(na_rows) > 0) {
  for (covar in covarNames) {
    # Find nearest neighbor index
    nearest_idx <- st_nearest_feature(mesh_sf[na_rows, ], ENP_grid)
    mesh_sf[na_rows, covar] <- ENP_grid[nearest_idx, covar, drop = TRUE]
  }
}

# Combine covariates from mesh nodes and observation points
n_time = length(unique(sp_dat_sf$Time_period)) # number of time periods
# Intercept column
```

```
fixed_effects = data.frame(intercept = rep(1,n_time*mesh$n+nrow(sp_dat_sf)))
for(cov in covarNames){
    fixed_effects[,cov]=c(rep(st_drop_geometry(mesh_sf)[,cov], n_time),
        st_drop_geometry(sp_dat_sf)[,cov])
}
```

Spatial SPDE prior configuration

Here we have to set the prior distribution for the two estimated parameters of the Matérn correlation function: the effective correlation range *range* (i.e., the distance at which the correlation attains a small value of approximately 0.1), and the standard deviation *sigma*. To define the priors we want, we set a threshold u and a probability level *alpha* such that $Pr(range < u_{range}) = alpha_{range}$ (for the range) and $Pr(sigma > u_{sigma}) = alpha_{sigma}$.

```
# Matérn covariance parameters:
nu = 1  # Smoothness parameter
d = 2    # Spatial dimension
alpha = nu + d/2  # SPDE parameterization

# Define PC priors for spatial field:
spde_sp = inla.spde2.pcmatern(
  mesh = mesh,
  alpha = alpha,
  prior.range = c(10000, 0.5),  # P(range < 10km) = 0.5
  prior.sigma = c(1, 0.5)       # P(sigma > 1) = 0.5
)
```

Spatiotemporal projection matrix construction

Next, we have to combine a matrix that maps the vector of latent variables (in particular, the Gaussian variables on the mesh nodes) to the linear predictor used for the observations. The latent Gaussian variables of the spatial field are replicated for each time step.

1. Define Number of Mesh Vertices and Observations

We first extract the number of **mesh vertices** (`nv`) and the **number of observed presence points** (`n`).

```
nv = mesh$n  # Number of mesh vertices
n = nrow(sp_dat_sf)  # Number of observations
```

- `nv`: Represents the **number of latent random field locations** (mesh nodes).
- `n`: Represents the **number of species presence observations**.

2. Create the Response Vector (`y.pp`)

The response vector encodes **presence-absence** information for the LGCP model. We combine responses of value zero for the pseudo-absences on the knots of the mesh for each time step, and responses of value one for the observed points.

```
# y.pp <- rep(0:1, c(nv, n))
y.pp <- rep(0:1, c(n_time*nv, n))
```

- The first `n_time*nv` elements are **0** (representing integration points/background locations for the `n_time` time periods).
- The last `n` elements are **1** (representing the observed presences).

3. Create the Exposure Vector (`e.pp`)

The exposure vector provides **weights** for background points and zeros for presence points. Again, for the background points, we have to take into account the temporal replication for the `n_time` time steps.

```
e.pp <- c(rep(w, n_time), rep(0, n))
```

- `w`: Contains **weights** for the integration points.
- `0` values are assigned to observed presences, as they do not need weights.

4. Create Projection Matrices for the spatiotemporal Field

Identity Matrix for Mesh Nodes (`imat`) Each mesh node is treated as a **latent basis function**, and the `mesh$n` spatial nodes are replicated `n_time` times. For the background/integration points, the locations of the responses are the same as the mesh nodes, so we can use a diagonal matrix.

```
imat <- Diagonal(n_time*nv, rep(1, n_time*nv))  # Identity matrix for mesh nodes
```

Observation to Mesh Projection (`lmat`) Observations are projected onto the spatial mesh using **barycentric interpolation**. We specify the time group of each observation when generating the observation matrix `A`.

```
lmat <- inla.spde.make.A(mesh, loc = st_coordinates(sp_dat_sf),
    group = sp_dat_sf$Time_period+1)
```

Combine Projection Matrices (`A.pp`) The final projection matrix links **background points** and **presence points** to the mesh.

```
A.pp <- rbind(imat, lmat)
```

We also generate a named index running through all the time-replicated integration nodes.

```
idx_sp = inla.spde.make.index("sp", n.group = n_time, n.spde = mesh$n)
names(idx_sp)
```

```
## [1] "sp"       "sp.group" "sp.repl"
```

```
head(idx_sp$sp)
```

```
## [1] 1 2 3 4 5 6
```

```
tail(idx_sp$sp)
```

```
## [1] 729 730 731 732 733 734
```

```
head(idx_sp$sp.group)
```

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

```
tail(idx_sp$sp.group)
```

```
## [1] 7 7 7 7 7 7
```

Summary of the Data Structures

Component	Purpose	Why is it Needed?
`nv, n`	Number of mesh vertices and presence points	Defines problem dimensions

Component	Purpose	Why is it Needed?
`n_time`	Number of time steps (i.e., number of temporal replicates)	Defines problem dimensions
`y.pp`	Binary response vector (`0` = background, `1` = observed presence)	Required for LGCP likelihood
`e.pp`	Exposure vector (weights for integration points, `0` for presences)	Corrects for intensity scaling
`imat`	Identity matrix for mesh vertices	Defines latent spatial field basis
`lmat`	Projection matrix (maps observations to mesh)	Ensures proper spatial interpolation
`A.pp`	Combined projection matrix (`imat + lmat`)	Links spatial process to both background & presence points

Data Stacking

Data stacking in **INLA** is a technique used to efficiently structure data for modeling, particularly when using the **SPDE approach** for spatial and spatiotemporal modeling. The goal is to combine multiple components of the data (observations, latent effects, exposure weights) into a single structure that INLA can process effectively.

```
stk.pp <- inla.stack(
  data = list(y = y.pp, e = e.pp),  # Response + exposure
  A = list(1, A.pp),                # Projection matrices
  effects = list(
    fixed_effects,                  # Fixed effects covariates
    idx_sp                 # Spatial random effect
  ),
  tag = 'pp')
```

Explanation of Components:

1. `data = list(y = y.pp, e = e.pp)`
   - `y.pp`: Response variable, combining background (`0`) and observed presences (`1`).
   - `e.pp`: Exposure term, used to scale the likelihood appropriately.
2. `A = list(1, A.pp)`
   - Defines **projection matrices** that link the different data components.
   - `1`: Represents an identity matrix for fixed effects.
   - `A.pp`: Links presence locations to the mesh through spatial interpolation.
3. `effects = list(fixed_effects, list(sp = 1:nv))`
   - `fixed_effects`: Environmental covariates.
   - `list(sp = 1:nv)`: The **spatial random effect**, which is linked to the mesh nodes.
4. `tag = 'pp'`
   - Provides a tag for identifying this stack in INLA later.

Model Formula Definition

We include an intercept and the covariates as fixed effects. For the space-time Gaussian field, we have to use the `f(...)`-function and give it the SPDE model and the group index of the time-replicated latent variables. Moreover, we have to specify the temporal correlation model. Here, we use a first-order autoregressive correlation model with name `ar1` in INLA.

```
c.g = list(model = "ar1")  #control.group argument
formula = y ~ -1 +                    # Remove intercept
  intercept +                         # Explicit intercept
  Axis1 + Axis2 + Axis3 +            # Environmental covariates
  Axis4 + Axis5 + Axis6 +
  f(sp, model = spde_sp, group = sp.group, ngroup = n_time, control.group = c.g)
  # Spatial random effect names(idx_sp)
```

Explanation of Components:

1. `y ~ -1 +`
   - `-1` removes the default intercept (explicitly defining it below).
2. `intercept +`
   - Adds a manually defined intercept to the model.
3. `Axis1 + Axis2 + Axis3 + Axis4 + Axis5 + Axis6 +`
   - Represents **environmental covariates** influencing species presence.
4. `f(sp, model = spde_sp, group = sp.group, ngroup = n_time, control.group = c.g)`
   - Defines the **spatiotemporal random effect** modeled using the **SPDE approach**.

Model Fitting with INLA

The `inla(...)`-function estimates the model.

```
fit <- inla(
  formula,
  E = inla.stack.data(stk.pp)$e,     # Exposure term
  data = inla.stack.data(stk.pp),     # Stacked data
  family = 'poisson',                # LGCP as Poisson process
  control.compute = list(            # Compute model metrics # TODO
    cpo=T, dic=T, mlik=T, waic=T,
    return.marginals.predictor = T,
    config = TRUE),
  control.predictor = list(
    A = inla.stack.A(stk.pp)),       # Projection matrix
  control.inla=list(
    strategy="adaptive",             # Optimization strategy
    int.strategy="eb"),
  num.threads = 2)
```

Explanation of Components:

1. `formula`
   - Uses the previously defined **spatial LGCP model**.
2. `E = inla.stack.data(stk.pp)$e`
   - Provides the **exposure term**, ensuring proper intensity scaling.
3. `data = inla.stack.data(stk.pp)`
   - Supplies the **stacked data** to INLA.
4. `family = 'poisson'`
   - LGCP is modeled as a **Poisson process**.
5. `control.compute`
   - Computes key model metrics:
     - **CPO** (Conditional Predictive Ordinate)
     - **DIC** (Deviance Information Criterion)
     - **MLIK** (Marginal Likelihood)

    - **WAIC** (Watanabe-Akaike Information Criterion)
    - **Return marginals for predictions**.
6. `control.predictor = list(A = inla.stack.A(stk.pp))`
    - Ensures correct **projection of predictions** using the stacked data.
7. `control.inla`
    - Uses an **adaptive strategy** for optimization.
    - **Empirical Bayes (EB)** integration strategy.
8. `num.threads = 2`
    - Utilizes **parallel computing** for efficiency. Each thread has memory requirement, so be careful if you have only limited memory ressources.

Next, we can show the summary of the fitted model, and more specifically of the hyperparameters of the model (here, the range, standard deviation and temporal autocorrelation coefficient of the Gaussian space-time model).

```
summary(fit)
```

```
## Time used:
##     Pre = 0.416, Running = 11.2, Post = 0.355, Total = 11.9
## Fixed effects:
##              mean    sd 0.025quant 0.5quant 0.975quant    mode kld
## intercept -17.655 0.690    -19.007  -17.655    -16.304 -17.655   0
## Axis1       0.186 0.044      0.100    0.186      0.272   0.186   0
## Axis2      -0.268 0.046     -0.358   -0.268     -0.178  -0.268   0
## Axis3       0.279 0.043      0.194    0.279      0.363   0.279   0
## Axis4      -0.325 0.052     -0.428   -0.325     -0.223  -0.325   0
## Axis5      -0.160 0.054     -0.266   -0.160     -0.053  -0.160   0
## Axis6       0.079 0.068     -0.054    0.079      0.212   0.079   0
##
## Random effects:
##   Name     Model
##     sp SPDE2 model
##
## Model hyperparameters:
##                      mean       sd 0.025quant 0.5quant 0.975quant     mode
## Range for sp    23201.87 3256.596   1.75e+04 2.30e+04   30320.35 2.24e+04
## Stdev for sp        2.02    0.267   1.55e+00 2.00e+00       2.60 1.96e+00
## GroupRho for sp     0.93    0.019   8.88e-01 9.33e-01       0.96 9.37e-01
##
## Deviance Information Criterion (DIC) ...............: 58657.90
## Deviance Information Criterion (DIC, saturated) ....: 950310.12
## Effective number of parameters .....................: 419.66
##
## Watanabe-Akaike information criterion (WAIC) ...: 59241.32
## Effective number of parameters .................: 856.03
##
## Marginal log-Likelihood:  -29516.16
## 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)')
```

```
fit$summary.hyperpar
```

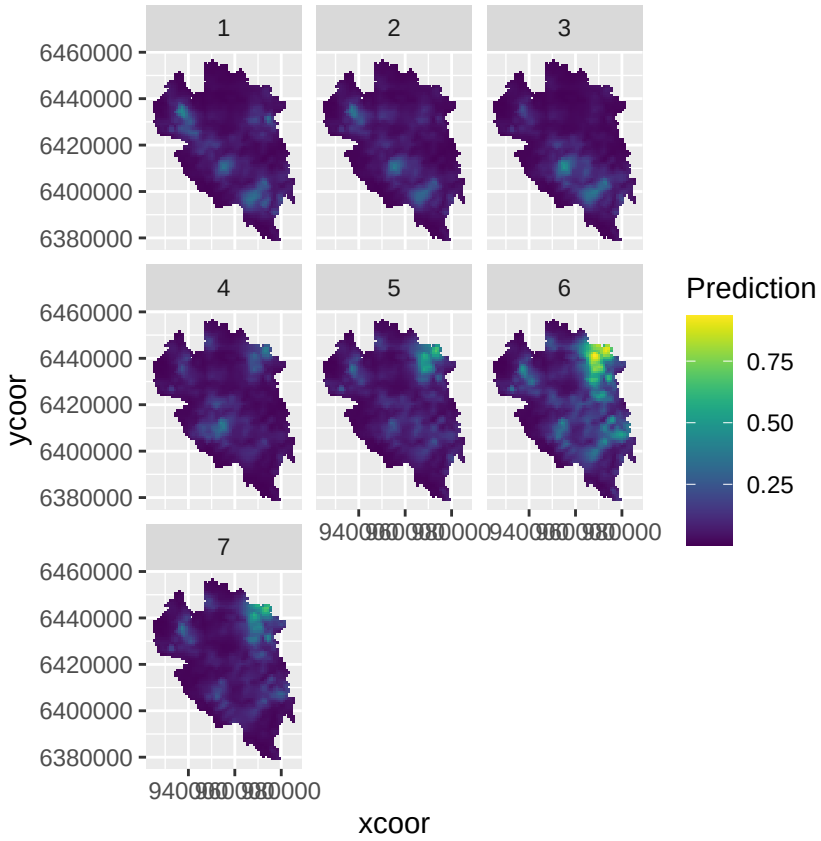
```
##                         mean           sd   0.025quant     0.5quant
## Range for sp    2.320187e+04 3.256596e+03 1.752330e+04 2.295287e+04
## Stdev for sp    2.022461e+00 2.674969e-01 1.551327e+00 2.003568e+00
```

```
## GroupRho for sp 9.304476e-01 1.864853e-02 8.875491e-01 9.326893e-01
##                   0.975quant         mode
## Range for sp    3.032035e+04 2.242011e+04
## Stdev for sp    2.602698e+00 1.963672e+00
## GroupRho for sp 9.604157e-01 9.369511e-01
```

Prediction and Evaluation of LGCP Model in INLA

We can retrieve the predicted fields for the different time steps and map them.

```
# Compute Area of Intersection Between Polygons and Study Domain
polygon_areas <- as.vector(st_area(st_intersection(ENP_grid, ENP_domain)))

# Spatially project Model Predictions onto Centroids of Polygons
#Creates a projection object to interpolate model predictions at specified locations
xy_grid = st_coordinates(st_centroid(ENP_grid))
projector <- inla.mesh.projector(mesh, xy_grid)

# Extract Model Predictions at Polygon Centroids
#Interpolates the model's median predicted values (`0.5quant`) onto polygon centroids
predicted_intensity <- inla.mesh.project(projector,
    fit$summary.fitted.values$`0.5quant`[1:mesh$n])
for(k in 2:n_time){
  predicted_intensity = cbind(predicted_intensity,
    inla.mesh.project(projector,
        fit$summary.fitted.values$`0.5quant`[(k-1)*mesh$n+1:mesh$n]))
}

# Compute Final Prediction Values
pred_val <- predicted_intensity * polygon_areas
pred1_val <- 1 - exp(-predicted_intensity * polygon_areas)
#pred1_val <- 1 - exp(-as.numeric(predicted_intensity * polygon_areas))
# - `pred_val`: Multiplies intensity predictions by polygon areas to scale predictions.
# - `pred1_val`: Converts intensity predictions into probabilities of presence using
#Poisson probability transformation.

df = data.frame(Prediction = as.numeric(pred1_val), xcoor = xy_grid[,1],
    ycoor = xy_grid[,2], tcoor = rep(1:n_time, each = nrow(xy_grid)))
pl <- ggplot(df) + coord_fixed()
pl <- pl + geom_tile(aes(x = xcoor, y = ycoor, fill = Prediction)) + facet_wrap(~tcoor)
pl <- pl + scale_fill_viridis()
print(pl)
```



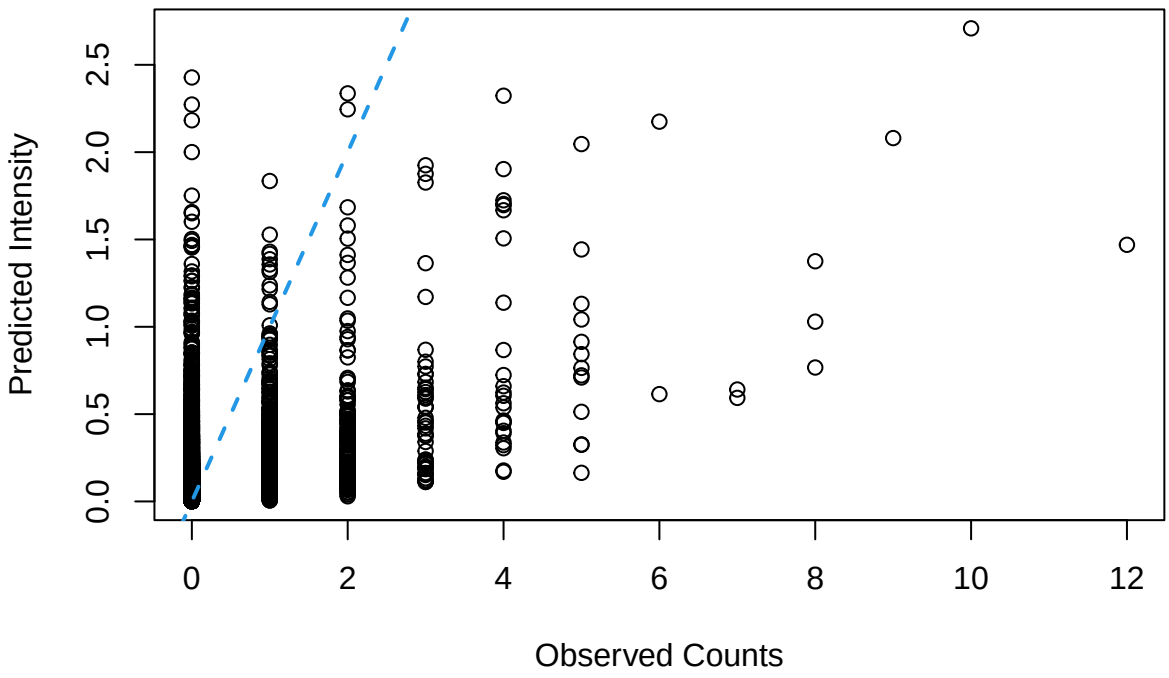

```
## Visualize Predicted Intensity Surface
#ggplot() +
#  geom_sf(data=ENP_grid, aes(fill=log10(pred_val))) +
#  geom_sf(data=sp_dat_sf) +
#  scale_fill_viridis() +
#  theme_minimal() +
#  labs(title = "Estimated Intensity Surface")
```

Compare Predictions with Observed Counts

```
# Count the number of observed points in each polygon
point_in_poly <- st_intersects(ENP_grid, sp_dat_sf, sparse = FALSE)
dim(point_in_poly)
```

```
## [1] 2707 1775
```

```
point_counts <- c()
for(k in sort(unique(sp_dat_sf$Time_period))){
  point_counts <- cbind(point_counts,  rowSums(point_in_poly[, sp_dat_sf$Time_period == k]))
}

# Scatter plot comparison
plot(as.numeric(point_counts), as.numeric(pred_val),
    xlab = "Observed Counts", ylab = "Predicted Intensity")
abline(a=0, b=1, lwd=2, lty=2, col=4)  # Reference line
```




```
# Correlation coefficient
cor(as.numeric(point_counts), as.numeric(pred_val))
```

```
## [1] 0.4355321
```

```
# Compute ROC Curve for Presence Prediction
test_roc <- roc(as.numeric(point_counts > 0), as.numeric(pred1_val))
print(paste0("AUC = ",test_roc$auc))
```

```
## [1] "AUC = 0.82225006460983"
```

```
print(paste0("TSS = ",max(test_roc$sensitivities + test_roc$specificities - 1)))
```

```
## [1] "TSS = 0.495591232716538"
```

```
# - `roc(as.numeric(point_counts > 0), pred1_val)`: Creates an ROC curve
#comparing presence/absence with predicted probabilities.
# - `test_roc$auc`: Computes Area Under the Curve (AUC).
# - `max(test_roc$sensitivities + test_roc$specificities - 1)`: Computes Youden's J
#statistic, a measure of optimal classification performance.
```

Fit a Poisson-GLMM model with a latent spatiotemporal field to the species occurrences

We note C the number of grid cells within our study area, and T the number of time periods considered. We aim at fitting the following Poisson-GLMM model.

[

$$
\begin{aligned}
Y_{i,t} &\sim \text{Poisson}(\lambda_{i,t}) \quad i \in [1, C], t \in [1, T] \\
\log(\lambda_{i,t}) &= \alpha + \log(nTG_{i,t}) + \beta^T Covar_i + W(z_i, t) \\
&\text{Where:} \\
nTG_i &: \text{Count of TGB occurrences in cell i for period t} \\
Covar_i &: \text{Covariate vector for cell i} \\
W(z_i, t) &: \text{Spatiotemporel Gaussian random effect at center } z_i \text{ of cell i}
\end{aligned}
\tag{1}
$$

]

For $W(s,t)$, we use a separable model with spatial SPDE (Gaussian field with Matérn covariance function) and temporal 1st order autoregression:

$$
\begin{aligned}
\varepsilon_t(s) &\overset{ind.}{\sim} \text{Matérn-SPDE(range)}, \quad t = 1, 2, ... \\
W(s, 1) &= \varepsilon_1(s) \\
W_(s, t+1) &= \rho W(s,t) + \sqrt{1-\rho^2}\varepsilon_{t+1}(s), \quad t = 1, 2, ...
\end{aligned}
$$

In this model, $nTG_{i,t}$ is an offset which can be interpreted as a multiplicative factor (varying across space and time) of the occurrence count, aiming at correcting spatio-temporal sampling bias.

Count TG and focal species occurrences per cell

```
tmp <- sp_dat_sf %>%
  st_drop_geometry()%>%
  select(id_cell, Time_period)%>%
  group_by(id_cell, Time_period)%>%
  summarize(nFocal=n())

Poisson_sf <- TG_dat_sf %>%
  group_by(id_cell, Time_period) %>%
  summarize(
    across(-c(Month,Year), \(x) mean(x, na.rm = TRUE)),
    nTG=n()) %>%
  merge(tmp,by=c('id_cell','Time_period'),all.x=T)%>%
  as.data.frame()
Poisson_sf$nFocal[is.na(Poisson_sf$nFocal)]=0

head(Poisson_sf)
```

```
##   id_cell Time_period       Axis1     Axis2     Axis3      Axis4      Axis5
## 1  cell_1           4 -0.08895126 0.7822310 0.3855413 0.06767522  0.1853005
## 2  cell_1           5 -0.08895126 0.7822310 0.3855413 0.06767522  0.1853005
## 3  cell_1           6 -0.08895126 0.7822310 0.3855413 0.06767522  0.1853005
## 4 cell_10           0 -0.69735524 0.7431553 0.3827320 0.74265032 -0.7415284
## 5 cell_10           4 -0.69735524 0.7431553 0.3827320 0.74265032 -0.7415284
## 6 cell_10           5 -0.69735524 0.7431553 0.3827320 0.74265032 -0.7415284
##       Axis6 nTG nFocal                       geometry
## 1 0.3435648   2      0       POINT (976247.9 6379313)
## 2 0.3435648  14      0 MULTIPOINT ((975984 6379653...
## 3 0.3435648   5      0 MULTIPOINT ((976254.8 63793...
## 4 0.6235026  11      0       POINT (977996.1 6380472)
## 5 0.6235026   1      0       POINT (978178.2 6380853)
## 6 0.6235026  10      0 MULTIPOINT ((977830 6380380...
```

Fixed random effects and time

Let's prepare fixed effects. We modify the Time_period column such that time periods run from 1 to 7.

```
fixed_effects = Poisson_sf[,c('nTG',paste0('Axis',1:6))]%>%
  mutate(intercept = 1,
         log_TG_Count=log(nTG)# linear predictor->log-scale
         )%>%
  select(-nTG)%>%st_drop_geometry()%>%as.data.frame()
fixed_effects$Intercept = 1
Poisson_sf$Time_period = 1 + Poisson_sf$Time_period
groups_time = sort(unique(Poisson_sf$Time_period))
```

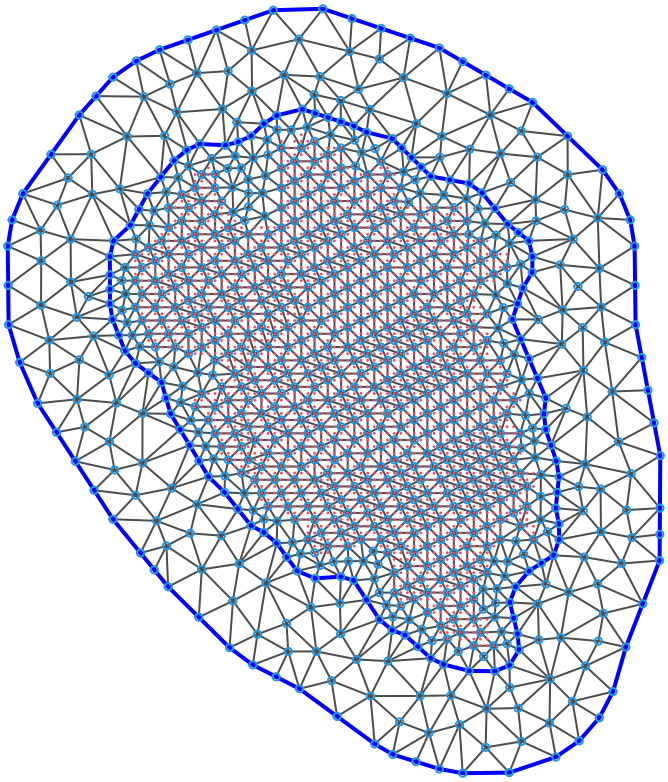
Set up components for the latent spatial fields

We first need to define the spatial mesh of latent gaussian variables. We get the coordinates of grid centroids and use them to build inner and outter boundaries for the mesh. The area between the inner and outter boundaries aims at minimizing boundary effects. Then, build the mesh of vertices at which the latent gaussian variables will be estimated.

```
grid_coords=as.matrix(ENP_grid[,c('x','y')]%>%st_drop_geometry())
## Compute the boundaries of the mesh
# - Inner boundary with slight concavity to follow data
bndint = inla.nonconvex.hull(grid_coords, convex = -0.05)
# - Outer boundary with stronger concavity for extension
bndext = inla.nonconvex.hull(grid_coords, convex = -0.25)
# Build triangular mesh:
# - max.edge: Maximum allowed triangle edge length (inner, outer zones)
# - cutoff: Minimum distance between mesh vertices
mesh = fm_mesh_2d_inla(grid_coords,
                       boundary = list(bndint, bndext),
                       max.edge = c(2500, 8000), cutoff = 2000)
print(mesh$n) # Number of mesh vertices
```

```
## [1] 897
```

```
par(mar = rep(0.5, 4))
plot(mesh, main = "", asp = 1)
points(grid_coords,col=2,cex=0.2,pch=16)
points(mesh$loc, col=4, cex=0.5)
```



We get the coordinates of cell centroids for each sampled spatio-temporal cell. Then, we compute the observation matrix A, which enables to linearly approximate latent gaussian field value in each cell from its value in surrounding mesh vertices.

```
# cell centroids coordinates per sampled spatio-temporal cell.
Poisson_sf=Poisson_sf %>%
merge(ENP_grid[,c('x','y','id_cell')]%>%st_drop_geometry(),
      by="id_cell",all.x=T)
coord_data = as.matrix(Poisson_sf[,c('x','y')])
# Compute the observation matrix A
A = inla.spde.make.A(mesh, loc = coord_data, group = Poisson_sf$Time_period)
# Indices for the total observation matrix A implemented with inla.stack.
idx_sp = inla.spde.make.index("sp", n.spde = mesh$n, n.group = max(groups_time))
```

Create the INLA stack

```
stk = inla.stack(
  data = list(y = as.numeric(Poisson_sf$nFocal)),  # Specify the response variable
  A = list(1, A),          # Vector of multiplication factors for random and fixed effects
  effects = list(fixed_effects, idx_sp),
  tag = "obs"  # Detail the random and fixed effects
)
```

Define the model and run the estimation

Define the 2D-SPDE priors for the spatial effect

```
nu = 1
d = 2 # dimension
alpha = nu + d / 2
spde_sp = inla.spde2.pcmatern(
  mesh = mesh, alpha = alpha,constr = T,
  prior.range = c(10000, 0.5), prior.sigma = c(1, 0.5)
)
```

Define the model formula with AR1 group model to link the spatial fields across time periods.

```
c.g = list(model = "ar1")  #control.group argument
## Express y as a function of the linear predictor in the formulate variable
formula = y ~ -1 + Intercept + Axis1 + Axis2 + Axis3 + Axis4 + Axis5 + Axis6 +
  f(sp, model = spde_sp, group = sp.group, ngroup = n_time, control.group = c.g)
```

Now we can call inla to fit the model.

```
fit = inla(
  formula, # Formula of the model
  offset = fixed_effects$log_TG_Count,
  data = inla.stack.data(stk), # data = Observation matrix
  family = "poisson",  # Distribution of the response variable
  control.compute = list(cpo=T,
                         dic=T,
                         mlik=T,
                         waic=T,
                         return.marginals.predictor = T,
                         config = TRUE),  # Statistics criteria
  control.predictor=list(compute = FALSE,
    A = inla.stack.A(stk), link = 1),
    # link = 1 to compute the fitted values with the links function
  control.inla = list(int.strategy = "eb",
    strategy = "adaptive"),  # Optimization settings
  verbose = F,#T, # Text option
  #inla.mode = "experimental",
  num.threads = 2)
summary(fit)
```

```
## Time used:
##     Pre = 0.357, Running = 30.7, Post = 0.559, Total = 31.6
## Fixed effects:
##             mean    sd 0.025quant 0.5quant 0.975quant   mode kld
## Intercept -4.228 0.064     -4.353   -4.228     -4.103 -4.228   0
## Axis1      0.797 0.031      0.736    0.797      0.858  0.797   0
## Axis2     -0.559 0.038     -0.633   -0.559     -0.484 -0.559   0
## Axis3     -0.083 0.033     -0.149   -0.083     -0.018 -0.083   0
## Axis4      0.040 0.043     -0.044    0.040      0.124  0.040   0
## Axis5      0.100 0.040      0.021    0.100      0.179  0.100   0
## Axis6      0.026 0.052     -0.075    0.026      0.128  0.026   0
##
## Random effects:
##   Name     Model
##     sp SPDE2 model
##
## Model hyperparameters:
```

```
##                      mean      sd 0.025quant 0.5quant 0.975quant     mode
## Range for sp    2685.862 492.570   1827.340 2648.271    3759.35 2585.067
## Stdev for sp       1.620   0.146      1.356    1.612       1.93    1.592
## GroupRho for sp    0.645   0.047      0.545    0.647       0.73    0.653
##
## Deviance Information Criterion (DIC) ...............: 9178.99
## Deviance Information Criterion (DIC, saturated) ....: 6365.35
## Effective number of parameters .....................: 794.28
##
## Watanabe-Akaike information criterion (WAIC) ...: 9460.42
## Effective number of parameters .................: 895.49
##
## Marginal log-Likelihood:  -4743.21
## 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)')
```

```
fit$summary.hyperpar
```

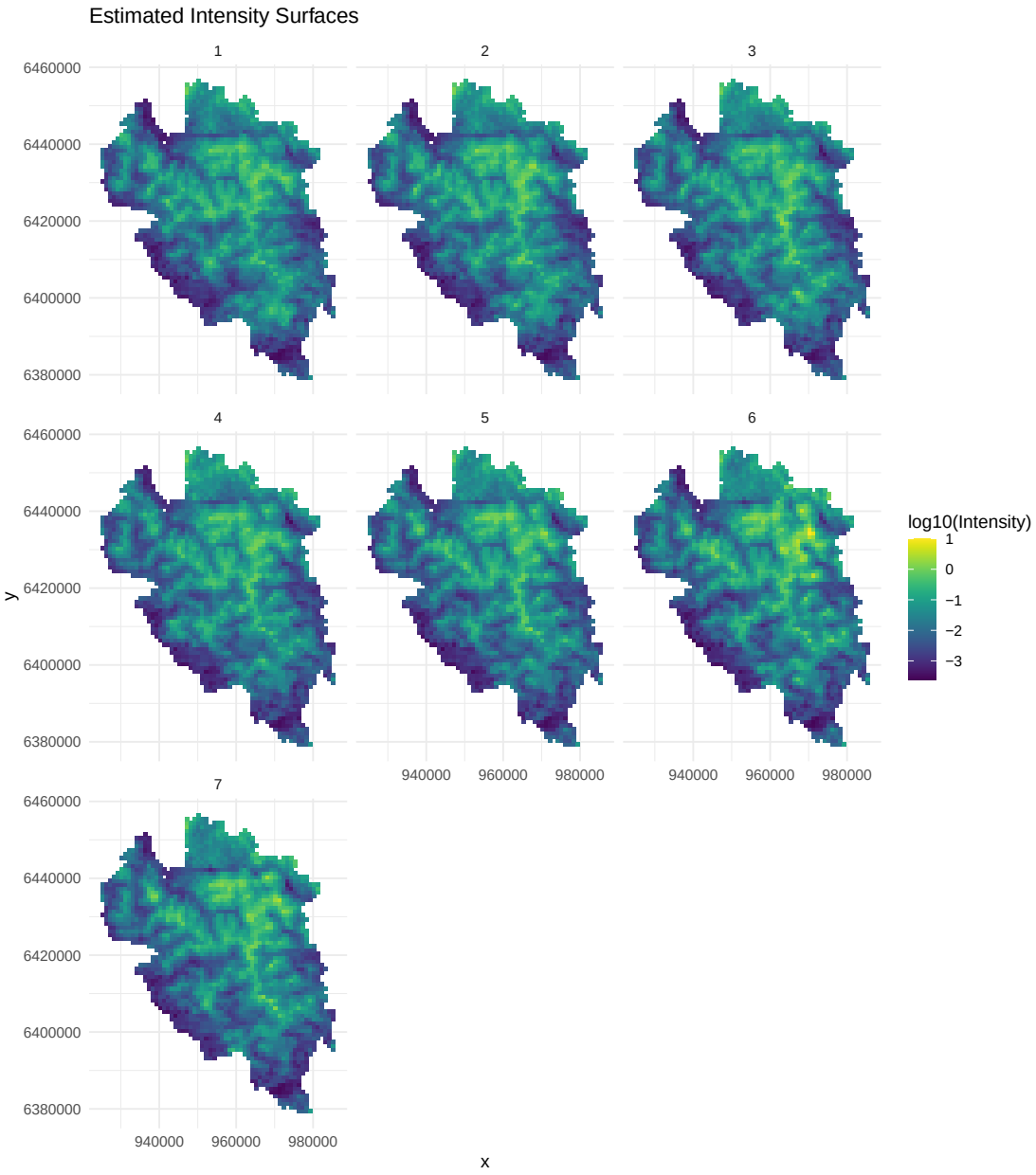
```
##                         mean           sd   0.025quant    0.5quant   0.975quant
## Range for sp    2685.8615608 492.57011170 1827.3402539 2648.271015 3759.3529684
## Stdev for sp       1.6201745   0.14629029    1.3564857    1.611871    1.9320245
## GroupRho for sp    0.6446838   0.04717259    0.5445866    0.647269    0.7299258
##                         mode
## Range for sp    2585.0667644
## Stdev for sp       1.5917748
## GroupRho for sp    0.6527552
```

Visualize the intensity surfaces by time period

We here compute the intensity over the grid for each time step and then plot it.We use the 500m grid. We can predict the TGB-corrected intensity surface on this grid (covariate fixed effects + Gaussian spatial fields).

```
# Calculate Fixed Effects
X=ENP_grid[,covarNames]%>%
  st_drop_geometry()%>%
  as.matrix()
X=cbind(Intercept=1,X)
# Get fixed effects coefficients
beta <- fit$summary.fixed$mean

# Calculate fixed effects component (intercept + covariates)
fixed_part <- X %*% beta

# Create projection matrix from mesh to prediction points
A_pred <- inla.spde.make.A(mesh = mesh, loc = grid_coords)

spatial_effects = c()
spatial_intensities = c()
for(k in groups_time){
  # Get estimated spatial field values (posterior mean)
  spatial_field <- fit$summary.random$sp$mean[(k-1)*mesh$n+1:mesh$n]
  # Calculate spatial component at prediction points
  spatial_effects <- cbind(spatial_effects, as.vector(A_pred %*% spatial_field))
  spatial_intensities <- cbind(spatial_intensities,
```

```
    exp(as.vector(A_pred %*% spatial_field) + fixed_part))
}

df = data.frame(Intensity = as.numeric(spatial_intensities),
    x = rep(grid_coords[,1], max(groups_time)),
    y = rep(grid_coords[,2], max(groups_time)),
    Time = rep(groups_time, each = nrow(spatial_intensities)))
ggplot(df) +
  geom_tile(aes(x = x, y = y, fill = log10(Intensity))) +
  coord_fixed()+
  scale_fill_viridis()+
  theme_minimal() +
  facet_wrap(~Time) +
  labs(title = "Estimated Intensity Surfaces")
```


Estimated Intensity Surfaces
1
2
3
4
5
6
7
6460000
6440000
6420000
6400000
6380000
940000
960000
980000
y
x
log10(Intensity)
1
0
−1
−2
−3