UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN MATHÉMATIQUES ET INFORMATIQUE APPLIQUÉES

PAR
GALASS NDOUR

Une revue de littérature exhaustive, une librairie Matlab concernant les copules pour l'analyse de données spatiales

Août 2022

AVANT-PROPOS

Les mathématiques, de par leur richesse et complexité, m'ont toujours fasciné et, par ricochet, sont vite devenues mon domaine de prédilection. En particulier, j'ai trouvé le domaine des statistiques captivant offrant ainsi une panoplie d'applications concrètes. C'est dans ce cadre que le présent mémoire s'appesantit sur la modélisation des données spatiales à l'aide des copules. Plus spécifiquement, il s'agit ici de présenter une revue de littérature exhaustive sur ce champ d'analyse ainsi que les perspectives, tout en proposant une librairie Matlab basée sur l'analyse spatiale avec les copules. Ce travail a été réalisé grâce au soutien de plusieurs personnes que je tiens à remercier.

Je tiens à témoigner mon profond respect et ma reconnaissance à Monsieur Jean-François Quessy, mon directeur de recherche, pour sa disponibilité, son ouverture d'esprit, son écoute attentive et son soutien infaillible aussi bien financier que moral tout au long de ce travail. La qualité de ce travail ne saurait être améliorée sans sa grande sagesse, sa magnanimité et son œil critique.

Ensuite, je veux remercier les professeurs Alain Goupil et Fathallah Nouboud, du Département de mathématiques et d'informatique de l'UQTR, pour avoir gracieusement accepté de lire et de commenter ce travail de recherche. Leurs remarques judicieuses ont permis d'améliorer la version finale de mon travail.

Le financement durant mes études a été fourni par un octroi individuel à M. Jean-François Quessy dans le cadre du programme des Subventions à la Découverte du Conseil de Recherche en Sciences Naturelle et en Génie (CRSNG) du Canada, ainsi que par l'Institut des Sciences Mathématiques du Québec.

Enfin, je tiens à exprimer ma reconnaissance et ma gratitude à ma famille, ainsi qu'à toutes les personnes qui m'ont entourées, soutenues et encouragées durant cette expérience. Papa et maman, je vous remercie pour m'avoir inculqué la discipline, l'amour

du travail et la culture de l'excellence. Votre sagesse, votre affection et votre rigueur ont fait de moi la personne que vous aimez si tendrement. Ma réussite est la vôtre. Je remercie mon frère Khadim, mes tantes, mes tontons, ma grand-mère pour leur présence, leur soutien inconditionnel et l'ambiance familiale qu'ils ont toujours cultivée. Seynabou, je te remercie pour ton amour sincère et ton soutien inconditionnel. Je remercie Dame, Pathé, Mouha, Latyr, Rokhaya, Momo, Alassane, Abib, Lucas, Alice pour leur amitié grandissante et inconditionnelle qu'il ont toujours soutenue et entretenue. Votre sagesse et vos conseils ont largement contribué à la réalisation de ce travail..

Table des matières

Liste des tableaux

Table des figures

Introduction

La théorie des copules est relativement nouvelle dans le monde des probabilités et statistique. En effet, le concept à été introduit, pour la première fois, en 1959, par Abe Sklar, professeur de mathématiques appliquées à l'Institut des technologies de l'Illinois à l'époque. Pour modéliser le comportement conjoint de plusieurs variables aléatoires, les copules demeurent depuis plusieurs années une alternative intéressante et fructueuse. Ainsi, cette méthode procure une grande flexibilité par rapport à la construction de modèles multivariés en rendant possible le choix des lois marginales indépendamment de la structure de dépendance qui est gérée intégralement par la copule. Depuis le début du XXI^{e} siècle, on assiste au pullulement des applications de la théorie des copules. En particulier, les copules on été récemment utilisées comme alternative à la modélisation spatiale traditionnelle.

La modélisation de données spatiales suscite un intérêt particulier et est devenue un sujet attractif dans la littérature scientifique. Dans ce cas, un phénomène continu, est observé à des endroits de l'espace géographique, appelé champ aléatoire. On trouve des données spatiales dans plusieurs domaines tels que la météorologie, la démographie, l'épidémiologie, la foresterie, l'écologie, les sciences du sol et la géologie. Les méthodes traditionnelles de statistique spatiale utilisent le variogramme pour modéliser la dépendance. Toutefois, cette méthode dépend fortement des lois marginales et son estimation est influencée par les valeurs extrêmes observées. C'est pourquoi le recours aux copules demeure intéressant dans la mesure où leur utilisation permet la construction de modèles flexibles. Bàrdossy [2] a été le pionnier dans l'utilisation des copules comme alternative aux méthodes classiques de modélisation spatiale.

Par ailleurs, l'objet de la statistique spatiale étant généralement l'interpolation spatiale, c'est-à-dire le fait de prévoir la valeur du phénomène à l'étude en une position non observée, les modèles basés sur des extensions de la famille des copules normales permettent le calcul de la prévision optimale qui est donnée par une espérance conditionnelle. C'est une alternative à la méthode de krigeage populaire classique consistant en la meilleure prédiction linéaire.

Dans ce présent travail, nous proposons de faire une revue de littérature exhaustive sur l'analyse des données spatiales par l'approche des copules. Plus spécifiquement, après avoir fait l'état des lieux de l'analyse spatiale, nous allons proposer une librairie Matlab pour la modélisation et l'interpolation spatiale ainsi que la visualisation. Les perspectives de projets futurs relatives à l'analyse des données spatiales seront élaborées à la fin de ce travail.

Le mémoire est organisé comme suit. Le chapitre 1 traite de la modélisation de données spatiales ; un accent particulier sera mis sur les approches classiques. Dans le chapitre 2, on s'intéresse à la modélisation de la dépendance spatiale avec les copules. Il s'agira de rappeler certains concepts relatifs aux copules, en s'attardant à leurs principales propriétés et en décrivant quelques familles de modèles. L'analyse spatiale avec les copules sera passée au peigne fin en décrivant les principaux modèles. Le chapitre 3 est dédié à la présentation de la librairie Matlab basée sur l'analyse des données spatiales avec les copules. Il sera question de décrire les programmes ayant servi à la modélisation et l'interpolation spatiale ainsi qu'à la visualisation. Le chapitre 4 est consacré à l'illustration des méthodologies proposées dans ce mémoire sur de vraies données spatiales. Ces dernières, recueillies à 155 points d'échantillonnage, concernent la concentration dans le sol d'un polluant, c'est-à-dire le cadmium, près des rives néerlandaises du fleuve Meuse.

Chapitre 1

Modélisation de données spatiales : les approches classiques

Ce chapitre traite des approches classiques de la modélisation de données spatiales. Il s'intéresse, en particulier, à la présentation des différents types de données spatiales ainsi qu'aux différentes méthodes traditionnelles utilisées dans l'analyse spatiale.

1.1 Définition et types de données spatiales

Les données relatives aux informations géospatiales (ou encore géographiques) sont utilisées dans notre vie quotidienne. Selon Yamagata et Seya [26], par exemple, l' information géospatiale est définie comme :

(1) de l'information représentant la position d'un point ou d'une zone spécifique dans l'espace (y compris les informations temporelles se rapportant à ces informations) ; et/ou

(2) toute information associée à cette information géographique.

Les données spatiales sont celles qui sont relatives aux informations géospatiales. Elles apparaissent dans plusieurs domaines d'applications tels que l'exploitation minière, les sciences de l'environnement et de la Terre, l'écologie, la biologie, la géographie, l'épidémiologie, l'agronomie, la foresterie, le traitement d'image, etc. C'est pourquoi il existe une myriade de types de données spatiales.

Le livre de référence sur les statistiques spatiales reste celui de Cressie [5]. Ce grand ouvrage de 900 pages a fait office de "dictionnaire" dans ce domaine pendant de nombreuses années. Ainsi, le premier chapitre a distingué les types de données spatiales selon la nature du domaine spatial et sont regroupés en trois catégories, que nous appellerons (1) données géostatistiques, (2) données latticielles, et (3) données ponctuelles. Nous commençons par décrire ces données.

Soit, $\mathbb{R}$ l'ensemble des nombres réels, et soit $s \in \mathbb{R}^d$ une position spatiale dans l'espace euclidien de dimension d (généralement $d = 2$ ou 3) [1]. Nous désignons un processus spatial [2] en d dimensions par $\left\{Z(s) : s \in D \subset \mathbb{R}^d\right\}$, où Z désigne l'attribut que nous observons, par exemple, le rendement ou la concentration d'un élément chimique dans le sol.

1.1.1 Les données géostatistiques

Le domaine D est un sous-espace continu de $\mathbb{R}^d$, le champ $Z(s)$ est observé en n sites fixés $\{s_1, ..., s_n\} \subset D$. En raison de la continuité de D, les données géostatistiques sont également appelées *données spatiales à variation continue*. Il est important d'associer la continuité au domaine, et non à l'attribut mesuré. Le fait que l'attribut Z soit continu ou discret n'a aucune incidence sur le caractère géostatistique ou non des données.

1. $d > 3$ est souvent utilisé dans le domaine des expériences informatiques.
2. Souvent appelé aussi champ aléatoire.

Le niveau de pollution dans une ville, les mesures de précipitations ou de la température de l'air dans un pays, les concentrations de métaux lourds dans la couche arable d'une région sont des exemples de données géostatistiques. La géostatistique traite, entre autres, les questions de modélisation, de prédiction (ou krigeage) en un site non-observé et de reconstruction de Z partout sur D.

1.1.2 Les données laticielles ou régionales

Les données latticielles se présentent lorsque, d'une part, le domaine D est discret, c'est-à-dire que $Z(s)$ peut être observé dans un certain nombre d'emplacements fixes dénombrables. Ces derniers peuvent être des points ou des régions, mais il s'agit généralement de codes postaux, de zones de recensement, de provinces, etc. et, dans la plupart des cas, les données sont spatialement agrégées sur ces régions. Elles sont également appelées données régionales. Et d'autre part, lorsque les emplacements dans D sont non-stochastiques, c'est-à-dire fixés. Il va de soi que le voisinage est un concept central de l'analyse des données latticielles.

Le taux de chômage par ville, le prix moyen des logements par province et les rendements agricoles par parcelle sont des exemples de données latticielles. À la différence des données géostatistiques, elles peuvent être observées de manière exhaustive et, dans ce cas, la prédiction n'a aucun sens.

1.1.3 Les données ponctuelles

Contrairement aux données géostatistiques et latticielles où le domaine D est fixe, dans les données ponctuelles, D est discret ou continu, mais aléatoire. Ces données apparaissent lorsque l'attribut étudié est l'emplacement des événements (observations).

Comme exemples de données ponctuelles, on peut citer la position des arbres dans une forêt ou celle des nids dans une colonie de reproduction d'oiseaux. L'objectif principal de l'analyse des données ponctuelles est de déterminer si la répartition des points est plutôt régulière ou bien si elle est due au hasard, ou encore si elle présente des agrégats.

1.2 Covariance et variogramme

1.2.1 Rappels sur les champs aléatoires

Soit Ω, un espace des observables, $\mathcal{F}$, la tribu des événements et $\mathbb{P}$ une probabilité. $(\Omega, \mathcal{F}, \mathbb{P})$ est appelé espace de probabilité. Soit encore, S un ensemble de sites et $(E, \mathcal{E})$ un espace d'état mesurable.

Définition 1.1 (Champ aléatoire)**.** Un champ aléatoire (ou processus stochastique) à valeur dans E est une famille $Z = \{Z(s) : s \in S\}$ de variables aléatoires définies sur $(\Omega, \mathcal{F}, \mathbb{P})$ et à valeurs dans l'espace d'état $(E, \mathcal{E})$ du processus. S est l'ensemble (spatial) des sites sur lequel est défini le processus. Si $Z(s)$ est un processus stochastique, alors pour tout k-uplet $(s_1, ..., s_k) \in S^k, k \geq 1$, la loi de $\{Z_{s_1}, ..., Z_{s_k}\}$ est appelée la distribution marginale à k dimensions du processus. La série de toutes ces lois marginales, dénotée par

$$\{p_{s_1,...,s_k} : s_i \in S, 1 \leq i \leq k, k \geq 1\},$$

constitue la famille de toutes les distributions fini-dimensionnelles de Z et s'appelle la loi spatiale du processus ; si $S \subseteq \mathbb{R}$, on parle de loi temporelle. Dans la suite de ce chapitre, les processus considérés seront à valeurs réelles, $E \subseteq \mathbb{R}$ étant muni de sa tribu borélienne, $E = \mathcal{B}(E)$.

Définition 1.2 (Processus du second ordre)**.** Z est un processus du second ordre si, pour tout $s \in S$, $E(Z_s^2) < \infty$. La moyenne de Z qui existe alors est la fonction $m : S \to \mathbb{R}$ définie par $m(s) = E(X_s)$. La covariance de Z est la fonction $c : S \times S \to \mathbb{R}$ définie, pour tout s, t, par $c(s,t) = \text{cov}(Z_s, Z_t)$.

Soit $L^2 = L^2(\Omega, \mathcal{F}, \mathbb{P})$ l'ensemble des variables aléatoires sur $(\Omega, \mathcal{F})$ à valeurs réelles et de carré intégrable. On notera $Z \in L^2$ si Z est un processus du second ordre. Les processus gaussiens constituent une sous-classe importante des processus dans L^2.

1.2.2 Processus stationnaire

Soit Z, un champ du second ordre de moyenne m et de covariance c sur $S = \mathbb{R}^d$ ou $\mathbb{Z}^d$.

Définition 1.3 (Processus stationnaire du second ordre)**.** On dit que Z est un processus stationnaire du second ordre sur S si sa moyenne est constante et si sa covariance est invariante par translation, c'est-à-dire que

$$E\{X(s)\} = m(s) \text{ et } c(s,t) = C(t-s).$$

$C : S \to \mathbb{R}$ est alors la fonction de covariance stationnaire de Z.

Le champ Z est strictement stationnaire si pour tout entier $k \in \mathbb{N}$, tout k-uplet $(t_1, ..., t_k) \in S^k$ et tout $h \in S$, la loi de $(X_{t_1+h}, X_{t_2+h}, ..., X_{t_k+h})$ ne dépend pas de h. Plus formellement, Z est stationnaire au sens strict si la loi spatiale du processus est invariante par translation.

Hypothèses de stationnarité

La stationnarité dans l'espace est l'un des concepts clés de la géostatistique. Elle signifie que la distribution du processus aléatoire possède certains attributs invariants. En considérant le moment d'ordre 1, nous supposons que la moyenne $\mu = E[Z(x)]$ est constante pour tout x.

Par ailleurs, considérons les moments d'ordre 2. L'équation (1.3) est restreinte seulement aux deux points particuliers x_1 et x_2, ce qui ne décrit pas le processus. Ainsi pour généraliser l'équation, deux hypothèses sont nécessaires :

— Cas où $x_1 = x_2$. L'équation (1.3) définit alors la variance $\sigma^2 = E\left[\{Z(x) - \mu\}^2\right]$ parfois appelée variance a priori du processus. Nous supposons que celle-ci est finie et, comme la moyenne, qu'elle est la même partout.

— Cas où $x_1 \neq x_2$. Leur covariance dépend alors de leur distance et non de leur position absolue : ceci s'applique à toute paire de points x_i, x_j séparés par le vecteur $h = x_i - x_j$, de sorte que nous avons

$$C(x_i, x_j) = E\left[\{Z(x_i) - \mu\}\{Z(x_j) - \mu\}\right] , \tag{1.1}$$

qui est constante pour tout h. La constance des moments d'ordre 1 et 2 de l'ensemble du processus constitue une stationnarité de second ordre ou une stationnarité faible.

1.2.3 Covariance spatiale

La covariance permet de déterminer la relation entre deux variables pour des observations appariées. Pour n paires d'observations, $z_{i,1}$; $z_{i,2}$, $i = 1, 2, ..., n$ de deux variables,

z_1 et z_2, la covariance empirique est donnée par

$$\widehat{C}(z_1, z_2) = \frac{1}{n} \sum_{i=1}^{n} (z_{i,1} - \bar{z_1}) (z_{i,2} - \bar{z_2}) \ . \tag{1.2}$$

où $\bar{z_1}$ et $\bar{z_2}$ représentent respectivement les moyennes de z_1 et z_2. Si les unités $i = 1, ..., n$ ont été tirées au hasard alors $\widehat{C}(z_1, z_2)$ estime sans biais la covariance de la population.

Cette définition de la covariance peut être appliquée pour mettre en relation deux variables aléatoires. Le concept est utilisé dans plusieurs domaines, notamment dans l'espace et Yaglom [25] le présente comme étant à la base de la prédiction spatiale. Dans notre cadre spatial, z_1 et z_2 deviennent $Z(x_1)$ et $Z(x_2)$, qui sont les ensembles de valeurs de la même propriété Z aux points $x_1$et x_2 ; la notation en Z majuscule faisant référence aux variables aléatoires. Ainsi, leur covariance est

$$C(z_1, z_2) = E\left[(Z(x_1) - \mu(x_1)) (Z(x_2) - \mu(x_2))\right] \ . \tag{1.3}$$

où $\mu(x_1)$ et $\mu(x_2)$ sont les moyennes de Z en x_1 et x_2. L'équation (1.3) est analogue à l'équation (1.2) mais sa solution reste indisponible car, pour chaque point, nous avons seulement une réalisation de Z et les moyennes ne sont pas connues. D'où l'importance de faire appel aux hypothèses de stationnarité permettant de traiter les valeurs à différents endroits comme différentes réalisations de Z.

1.2.4 La fonction de covariance

L'équation (1.1) peut être réécrite

$$\begin{aligned} C\left[Z(x), Z(x+h)\right] &= E\left[(Z(x) - \mu) (Z(x+h) - \mu)\right] \\ &= E\left[(Z(x)) (Z(x+h)) - \mu^2\right] \\ &= C(h) \, . \end{aligned} \tag{1.4}$$

Ainsi, la fonction de covariance dépend uniquement du retard h et elle est également appelée fonction d'autocovariance car elle représente la covariance de Z avec lui-même. L'autocovariance dépend de l'échelle à laquelle Z est mesuré, et elle est souvent plus pratique à évaluer si elle est sans dimension grâce à l'autocorrélation, c'est-à-dire

$$\rho(h) = \frac{C(h)}{C(0)}\,, \tag{1.5}$$

$C(0)$ étant la covariance en $h = 0$, c'est-à-dire la variance σ^2.

1.2.5 La variation intrinsèque et le variogramme

Un processus aléatoire stationnaire peut être représenté par le modèle :

$$Z(x) = \mu + \epsilon(x)\,, \tag{1.6}$$

où μ est la moyenne du processus et ϵ une composante aléatoire provenant d'une distribution de moyenne nulle et une fonction de covariance donnée par

$$E\left[\epsilon(x)\epsilon(x+h)\right]. \tag{1.7}$$

Le problème le plus préoccupant découlant de la stationnarité faible est que la moyenne semble changer dans une région et que la variance augmente indéfiniment à mesure que la zone d'intérêt s'accroît. Dans ces circonstances, la covariance ne peut être définie.

Matheron [14] s'est penché sur le problème et a proposé une solution ayant contribué significativement à la géostatistique. Il a remarqué que, bien qu'en général la moyenne ne soit pas constante, elle le serait au moins pour les petits $|h|$, de sorte que les

différences attendues seraient nulles, à savoir que

$$E\left[Z(x) - Z(x+h)\right] = 0\,. \tag{1.8}$$

De surcroit, il a remplacé les covariances par les variances des différences comme mesures de la relation spatiale qui, comme la covariance, dépendent du décalage et non de la position absolue. Cela a conduit à

$$var\left[Z(x) - Z(x+h)\right] = E\left[\left(Z(x) - Z(x+h)\right)^2\right] = 2\gamma(h)\,. \tag{1.9}$$

Les équations (1.8) et (1.9) constituent l'hypothèse intrinsèque de Matheron qui a levé les contraintes de la stationnarité de second ordre. La quantité $\gamma(h)$, connue sous le nom de semivariance au retard h, est aussi appelée semivariogramme du fait de la fonction de h. La fonction $2\gamma(h)$ est par conséquent appelée variogramme.

1.2.6 Équivalence avec la covariance

Pour les processus stationnaires du second ordre, le variogramme et la covariance sont équivalents. En effet, partant de leurs définitions aux équations (1.4) et (1.9),

$$\gamma(h) = C(0) - C(h)\,. \tag{1.10}$$

Ainsi, un graphique du variogramme est simplement une image inversée de la fonction de covariance autour d'une ligne ou d'un plan parallèle à l'abscisse.

1.2.7 Estimation des variogrammes et des covariances

Le variogramme est la pierre angulaire de la géostatistique. Il est donc essentiel de l'estimer, de l'interpréter et de le modéliser correctement. Cette partie concerne son estimation à l'aide de l'estimateur de la méthode des moments de Matheron.

L'estimateur du semivariogramme de l'équation (1.9) est défini par

$$\widehat{\gamma}(h) = \mathrm{E}\left[(z(x) - z(x+h))^2\right] , \tag{1.11}$$

où $z(x)$ et $z(x+h)$ représentent les valeurs réelles de Z aux endroits distants de h. Pour un ensemble de données $z(x_i)$, $i = 1, 2, ...$, on peut calculer (Richard and Margaret [20])

$$\widehat{\gamma}(h) = \frac{1}{2m(h)} \sum_{i=1}^{m(h)} (z(x) - z(x+h))^2 , \tag{1.12}$$

où $m(h)$ est le nombre de paires de points de données séparées par h. En changeant h, nous obtenons un ensemble ordonné de semi-variances qui constituent le variogramme expérimental ou variogramme d'échantillon. L'équation (1.11) est la formule habituelle de calcul des semivariances ; elle est communément appelée estimateur par la méthode des moments de Matheron.

1.3 Simulations de champs aléatoires

1.3.1 Simulation d'un processus aléatoire

Le terme *simulation* est utilisé, en statistique, pour désigner la génération de valeurs d'une ou plusieurs variables qui reproduisent les caractéristiques générales de celles que nous observons dans la réalité. Les variables peuvent être qualitatives ou continues.

La simulation de données spatiales est importante à plusieurs égards :

- une méthode statistique pour les données géoréférencées ne peut souvent être validée que si elle est satisfaisante sur le long terme ;
- l'inférence statistique pour les données spatiales repose souvent sur des tests de randomisation. La capacité de simuler rapidement et efficacement des réalisations d'un processus hypothétique est importante pour permettre la production d'un nombre suffisant de réalisations ;
- l'absence de répliques dans la plupart des ensembles de données spatiales nécessite l'observation répétée d'un phénomène pour obtenir des estimations empiriques de la moyenne, de la variance et de la covariance.

Générer une réalisation d'un champ aléatoire $\left\{Z(s) : s \in D \subset \mathbb{R}^d\right\}$ n'est pas une tâche triviale, car elle nécessite la connaissance de la distribution spatiale

$$\mathbb{P}\left\{Z(s_1) < z_1, Z(s_2) < z_2, ..., Z(s_n) < z_n\right\}.$$

À partir d'un ensemble particulier de données et sous des hypothèses de stationnarité, nous pouvons seulement inférer la structure du premier et du second ordre du champ. L'inférence de la distribution conjointe de $Z(s_i)$ à partir de la moyenne et de la fonction de covariance n'est possible que si le champ aléatoire est gaussien. Même si la distribution spatiale était connue, il n'est généralement pas possible de construire une réalisation par simulation à partir de celle-ci. Le mieux que nous puissions faire - à

moins que le champ aléatoire ne soit gaussien - est de produire des réalisations dont les distributions marginales correspondent aux distributions respectives du champ cible.

Dans les deux prochaines sous-sections, il s'agira de passer en revue quelques méthodes permettant de générer des données spatiales. Certaines méthodes génèrent des données spatiales dont la distribution spatiale est connue, par exemple les méthodes de simulation de champs aléatoires gaussiens. D'autres génèrent des données qui ne respectent que les hypothèses de premier et de second ordre, c'est-à-dire des données dont la moyenne, la variance et la fonction de covariance sont connues.

1.3.2 Simulation inconditionnelle de champs gaussiens

De la même manière que la distribution gaussienne est la clé de nombreuses approches des inférences statistiques, le champ aléatoire gaussien occupe une position centrale dans l'analyse de données spatiales. Les meilleurs prédicteurs linéaires sans biais de krigeage (estimation d'une variable aléatoire en un point arbitraire) correspondent aux moyennes conditionnelles dans les champs aléatoires gaussiens. À cet effet, Matheron [14] a développé la théorie de la méthode de krigeage au début des années 1960 en tant que meilleur prédicteur linéaire sans biais de données spatiales.

Par ailleurs, dans un champ aléatoire gaussien, la stationnarité de second ordre implique une stationnarité stricte, les propriétés statistiques des estimateurs dérivés des données gaussiennes sont faciles à examiner et les statistiques de test ont généralement une distribution connue et simple. Ainsi la simulation de réalisations à partir d'un champ aléatoire gaussien devient une tâche importante.

Il existe une différence entre les simulations inconditionnelle et conditionnelle. Dans une simulation inconditionnelle, les réalisations générées respectent les contraintes statistiques globales (par exemple, la loi de probabilité marginale) sans être localement

restreintes par les observations. Les simulations de ce type peuvent reproduire les modèles globaux observés et sont souvent utilisées à des fins exploratoires.

D'autre part, la simulation conditionnelle génère des situations respectant à la fois les contraintes globales et les contraintes locales imposées par les observations. Par conséquent, les simulations conditionnelles sont plus représentatives de la réalité si des données spatiales sont disponibles.

Il existe plusieurs méthodes pour la simulation inconditionnelle des champs aléatoires. La méthode la plus simple - et probablement la plus brute - repose sur la propriété de reproduction de la distribution gaussienne (multivariée) et sur le fait qu'une matrice Σ définie positivement peut être représentée par

$$\Sigma = \Sigma^{1/2}\Sigma'^{1/2}.$$

Ainsi, si $X \sim \mathcal{N}(0,1)$, alors

$$\mu + \Sigma^{1/2}X$$

a une distribution $\mathcal{N}(\mu, \Sigma)$. Deux méthodes élémentaires pour générer une matrice racine carrée de la matrice de variance-covariance Σ sont la décomposition de Cholesky et la décomposition spectrale.

(i) Décomposition de Cholesky (LU). Étant donné $\Sigma_{n\times n}$ une matrice définie positive, alors il existe une matrice triangulaire supérieure $U_{n\times n}$ telle que $\Sigma = U'U$. La matrice U est appelée la racine de Cholesky de Σ. Puisque U' et U sont respectivement triangulaire inférieure et triangulaire supérieure, la décomposition est souvent appelée décomposition inférieure-supérieure ou décomposition LU.

(ii) Décomposition spectrale. Une deuxième méthode pour générer une matrice racine carrée de Σ s'appuie sur la décomposition spectrale d'une matrice symétrique réelle. Si $A_{p\times p}$ est une matrice symétrique réelle, alors il existe une matrice orthogonale

P de format $(p \times p)$ telle que

$$A = P\Delta P',$$

où Δ est une matrice diagonale contenant les valeurs propres de A. Puisque $PP' = I$, la matrice

$$\Sigma^{1/2} = P\Delta^{1/2}P'$$

possède les propriétés nécessaires pour se comporter comme la matrice de racine carrée et générer $\mathcal{N}(\mu, \Sigma)$ par :

$$y = \mu + \Sigma^{1/2}x$$

1.3.3 Simulation conditionnelle d'un champ aléatoire gaussien

Une simulation conditionnelle est une réalisation $S(s)$ d'un champ aléatoire $Z(s)$ reflétant les valeurs observées de $Z(s)$. Soient, $Z(s_1), Z(s_2), ..., Z(s_m)$, les valeurs observées d'un champ aléatoire. Une simulation conditionnelle produit $n = m + k$ valeurs

$$S(s) = [Z(s_1), Z(s_2), ..., Z(s_m), S(s_{m+1}), ..., S(s_{m+k})]'.$$

Si $m = 0$, la simulation devient inconditionnelle. Certaines méthodes de simulation conditionnelle conditionnent directement les données, tandis que d'autres commencent par une simulation inconditionnelle qui est ensuite conditionnée.

L'approche séquentielle est la méthode la plus simple pour simuler un champ gaussien multivarié. L'idée de la simulation séquentielle est simple. Pour le cas général, considérons la simulation d'un vecteur aléatoire Y avec une distribution connue

$$F(y_1, ..., y_n) = Pr(Y_1 \leq y_1, ..., Y_n \leq y_n).$$

La distribution de probabilité conjointe peut être décomposée en distributions condi-

tionnelles

$$F(y_1, ..., y_n) = F(y_1)F(y_2|y_1) \times ... \times F(y_n|y_1, ..., y_{n-1}).$$

La distribution conjointe peut être générée à partir des distributions conditionnelles lorsqu'elles sont accessibles. Le nom de la méthode provient de la nature séquentielle de la décomposition. Les simulations séquentielles ont l'avantage de produire un champ aléatoire avec la structure de covariance et la distribution spatiale adéquates. L'inconvénient est de devoir calculer les distributions conditionnelles.

Dans le cas particulier où $Z(s)$ est un champ aléatoire gaussien, les distributions conditionnelles sont simples. Ainsi, si

$$\begin{bmatrix} Z(s_0) \\ Z(s) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_0 \\ \mu \end{bmatrix}, \begin{bmatrix} \sigma^2 & c_0' \\ c_0 & \Sigma \end{bmatrix} \right),$$

alors $Z(s_0)|Z(s)$ est une distribution gaussienne de moyenne (Oliver and Carol [17])

$$\mathrm{E}[Z(s_0)|Z(s)] = \mu_0 + c'\Sigma^{-1}(Z(s) - \mu)$$

et de variance

$$\mathrm{var}[Z(s_0)|Z(s)] = \sigma^2 - c'\Sigma^{-1}c.$$

Si la moyenne du champ aléatoire est connue, il s'agit du prédicteur de krigeage simple et de la variance de krigeage correspondante. Nous pouvons donc calculer les distributions conditionnelles de $S(s_{m+i})$ étant donné $z(s_1), ..., z(s_m), s(s_{m+1}), ..., s(s_{m+i-1})$ comme gaussiennes. La moyenne de la distribution est le prédicteur de krigeage simple de $S(s_{m+i})$ basé sur les données $Z(s_1), ..., Z(s_m)$ et $S(s_{m+1}), ..., S(s_{m+i-1})$.

1.3.4 Recuit simulé (simulated annealing)

Le *Simulated annealing* est un terme générique pour un ensemble d'algorithmes qui optimisent plutôt que de simuler strictement. Le concept découle de la manière dont un métal en fusion se refroidit. Si un métal en fusion est refroidi lentement, les molécules peuvent se déplacer librement, atteignant finalement un état de solide avec une faible énergie et peu de tension. En revanche, si le refroidissement est rapide, le système ne pourra pas trouver l'état de basse énergie car le mouvement des molécules est entravé. Le solide n'atteindra pas l'équilibre thermique à une température donnée, les déficits sont gelés dans le solide. C'est le processus de recuit.

Clayton et André [4] ont introduit le recuit simulé en géostatistique pour créer des champs aléatoires avec des caractéristiques spécifiques. En géostatistique, les valeurs d'une variable régionalisée sont équivalentes aux molécules. Ces valeurs peuvent être déplacées ou remplacées par la méthode de manière à minimiser une certaine fonction objective qui mesure l'écart entre les caractéristiques cibles et actuelles de la réalisation à chaque i^{me} perturbation des données. La fonction objective ([4]) consiste à reproduire le modèle du variogramme,

$$G_i = \sum_{m=1}^{M} \left(\widehat{\gamma}_i(h)_m - \gamma(h)_m\right)^2 , \tag{1.13}$$

où $\gamma(h)_m$, $m = 1, ..., M$ sont les valeurs du modèle empirique, et les $\widehat{\gamma}_i(h)_m$ sont les valeurs calculées pour la réalisation courante sur la grille complète et donnent la valeur G. La quantité M définit la limite dans laquelle G doit être calculé. Si la variation est isotrope, c'est-à-dire, si la structure du variogramme et de la fonction de covariance reste identique en fonction de la diection, alors M est le nombre maximum d'intervalles sur la grille.

Les étapes du recuit simulé sont les suivantes (Richard and Margaret [20]) :

1. Le processus commence par des données, c'est-à-dire des valeurs observées de $Z(x)$ pour lesquelles nous disposons d'un modèle empirique du variogramme $\gamma(h)$, et la simulation est conditionnée par ces données. Le processus génère des valeurs supplémentaires sur une grille fine.

2. Calculer la valeur initiale de la fonction objective à partir de la réalisation initiale.

$$G(0) = \sum_{m=1}^{M} \left(\widehat{\gamma}_0(h)_m - \gamma(h)_m\right)^2 , \tag{1.14}$$

3. Perturber la réalisation en échangeant des paires de valeurs, ou par remplacement.

   Échange. Les valeurs de deux nœuds, disons $z(x_i)$ et $z(x_j)$, sont choisies au hasard et échangées, et G est recalculé. Si G est diminué, cela signifie que le nouveau variogramme expérimental, $\widehat{\gamma}_i(h)$, est plus proche de $\gamma(h)$ que le premier, et l'échange est donc conservé. Le processus se poursuit avec un échange de deux autres valeurs, un nouveau calcul de G, et le maintien de l'échange si G est diminué.

   Remplacement. La valeur d'un nœud choisi au hasard est remplacée par une autre valeur tirée au hasard de la même distribution que celle des valeurs originales, et G est recalculé. Comme dans le mode de permutation, la nouvelle valeur est conservée si G est diminué.

4. Le processus se termine lorsque G est devenu suffisamment petit ou que le nombre d'échanges a atteint une limite prédéfinie.

1.4 Méthodes de Krigeage

1.4.1 La prédiction spatiale

En raison de contraintes budgétaires ou technologiques, ou même de politiques de protection de la vie privée, les données spatiales ne peuvent être mesurées qu'à un nombre limité d'endroits. C'est pourquoi, dans de nombreux cas, il est nécessaire d'effectuer une sorte de prédiction, par exemple en estimant la distribution géographique des données à partir de données d'observation discrètes. En géostatistique, le terme "prédiction" désigne l'estimation de variables aléatoires à des points arbitraires autres que les points d'observation. La prédiction spatiale consiste en une interpolation et une extrapolation spatiales (Figure 1.1).

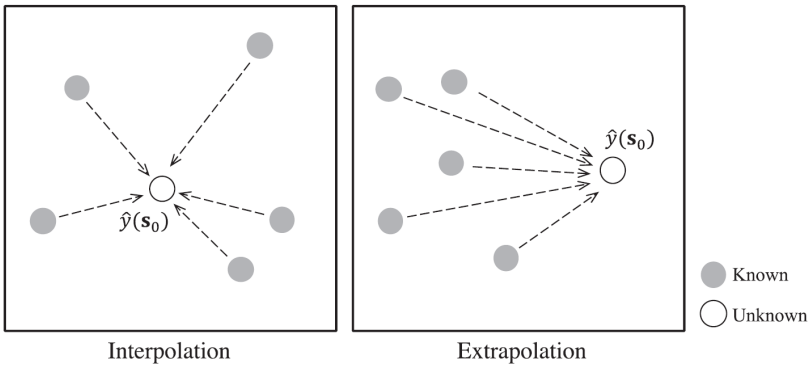


FIGURE 1.1 – Interpolation et extrapolation spatiales

L'interpolation prédit les valeurs numériques à l'intérieur d'une plage géographique contenant les valeurs observées, tandis que l'extrapolation prédit les valeurs numériques externes. Il existe plusieurs méthodes de prédiction spatiale. La méthode de pondération de la distance inverse, une méthode classique de prédiction spatiale, utilise l'inverse de la distance comme poids pour déterminer la valeur prédite des données en un point arbitraire de manière déterministe comme une somme linéaire des valeurs observées. Cette méthode peut être considérée comme une méthode qui tente de faire une prédiction basée uniquement sur l'information d'autocorrélation spatiale. Cepen-

dant, elle ne tient pas compte de la distribution locale entre les points d'observation (par exemple, dans la figure 1.1, si les points d'observation sont concentrés sur le côté supérieur du diagramme ou bien équilibrés entre les côtés supérieur et inférieur).

D'autre part, dans les modèles de régression, les valeurs prédites sont déterminées de manière probabiliste à l'aide des informations sur les attributs Z. Il est ainsi possible de déterminer la variance des erreurs de prédiction, ce qui permet d'évaluer objectivement la fiabilité des valeurs d'interpolation et d'extrapolation. Cependant, dans les prédictions avec le modèle de base, l'autocorrélation spatiale n'est pas prise en compte.

1.4.2 Le Krigeage

Le krigeage universel, l'une des méthodes de krigeage, prend en compte les deux aspects susmentionnés (relation mutuelle entre les points d'observation et l'autocorrélation spatiale). Ayant résumé les méthodes de prédiction spatiale les plus typiques et présenté 18 études comparatives sur la précision et l'exactitude de la prédiction spatiale, Jin et Andrew [10] ont conclu que les méthodes de krigeage sont plus performantes que les méthodes déterministes.

Dévelopé par D. G. Krige, un ingénieur minier sud-africain, le krigeage est une méthode statistiquement supérieure qui donne la meilleure prédiction linéaire sans biais des variables aléatoires en des points arbitraires. Ainsi, elle tient compte de l'autocorrélation spatiale et de la relation positionnelle entre les valeurs observées ; elle peut être combinée à un modèle de régression et prendre en compte divers facteurs de tendance. S'il n'y a pas d'erreurs d'observation, les valeurs observées correspondent aux valeurs prédites (c'est-à-dire interpolateur exact), et les erreurs de prédiction peuvent être calculées.

Pour définir la qualité de la prédiction, la fonction de perte entre la prédiction $\widehat{Z}(s_0)$ au point de prédiction s_0 et la vraie valeur $Z(s_0)$ est définie par

$$Fonction\ de\ perte = \left(Z(s_0) - \widehat{Z}(s_0)\right)^2 . \tag{1.15}$$

Puisque cette dernière varie de manière aléatoire, il est nécessaire de déterminer une prédiction qui minimise sa valeur attendue, à savoir

$$E\left[\left(Z(s_0) - \widehat{Z}(s_0)\right)^2\right] . \tag{1.16}$$

Cette valeur est appelée l'erreur quadratique moyenne de prédiction et sa minimisation est l'objet des méthodes de krigeage.

1.4.3 Le krigeage universel

Supposons les observations $Z(s_1), ..., Z(s_n)$ à des positions spatiales $s_1, ..., s_n$, et l'on souhaite prédire $Z(s_0)$ à la position s_0 où nous n'avons pas d'observation. Supposons en outre que la forme du modèle linéaire général soit valable à la fois pour les données et les valeurs latentes, à savoir

$$Z(s) = X(s)\beta + e(s), \text{ et } Z(s_0) = x(s_0)\beta + e(s_0),$$

où $x(s_0)$ est un vecteur de variables explicatives de dimension (p) associées à la position s_0. Nous supposons une matrice de variance-covariance générale pour les données, $\Sigma_{jk} = \text{var}\left(Z(s_j), Z(s_k)\right)$, et également que les données et les inobservables sont spatialement corrélées, de sorte que $Cov[Z(s), Z(s_0)] = \sigma$, un vecteur de format $(n \times 1)$, et $Var[Z(s_0)] = \sigma_0$.

Le Krigeage universel (Ku) a pour but de trouver le prédicteur linéaire optimal, c'est-à-dire à variance minimale dans la classe des prédicteurs linéaires sans biais. Ainsi,

nous considérons des prédicteurs de la forme $\chi' Z(s)$, et nous trouvons le vecteur χ pour que $\chi' Z(s)$ soit le meilleur prédicteur linéaire sans biais de $Z(s_0)$. Statistiquement, ce problème se pose : trouver le vecteur χ qui minimise l'erreur quadratique moyenne de prédiction (Oliver and Carol [17])

$$\begin{aligned} E\left[\{\chi' Z(s) - Z(s_0)\}^2\right] &= Var\left[\chi' Z(s)\right] + Var\left[Z(s_0)\right] - 2Cov\left[\chi' Z(s), Z(s_0)\right] \\ &= \chi' \Sigma \chi + \sigma_0 - 2\chi' \sigma\,, \end{aligned} \tag{1.17}$$

sous la contrainte $E[\chi Z(s)] = E[Z(s_0)]$. Cette condition d'absence de biais implique

$$\chi' X(s)\beta = x(s_0)'\beta \ \forall \beta$$

ce qui donne $\chi' X(s) = x(s_0)'$. Pour minimiser cette fonction sous la contrainte, nous utilisons la méthode des multiplicateurs de Lagrange. Le Lagrangien est

$$\mathcal{L} = \chi' \Sigma \chi + \sigma_0 - 2\chi' \sigma + 2m' \left(X(s)'\chi - x(s_0)\right),$$

où m est un vecteur de multiplicateur de Lagrange de dimension $p \times 1$. En dérivant par rapport à χ et m, on obtient

$$\frac{\mathcal{L}}{\chi} = 2\Sigma\chi - 2\sigma + 2X(s)m \text{ et } \frac{\mathcal{L}}{m} = 2\left(X(s)'\chi - x(s_0)\right).$$

En égalisant chaque terme à zéro et en réarrangeant, on obtient

$$\Sigma\chi + X(s)m = \sigma \text{ et } X(s)'\chi = x(s_0).$$

En résolvant ces équations pour χ, on obtient

$$\begin{aligned} \chi &= \Sigma^{-1}\left(\sigma - X(s)(X(s)'\Sigma^{-1}X(s))^{-1}(X(s)'\Sigma^{-1}\sigma - x(s_0))\right) \\ &= \Sigma_X^{-}\sigma + \Sigma^{-1}X(s)\left(X(s)'\Sigma^{-1}X(s)\right)^{-1}x(s_0)\,, \end{aligned} \tag{1.18}$$

avec $\Sigma_X^{-} = \Sigma^{-1} - \Sigma^{-1}X(s)\left(X(s)'\Sigma^{-1}X(s)\right) - 1X(s)'\Sigma^{-1}$.

Remarquons que $(X(s)\Sigma^{-1}X(s))^{-1}X\Sigma^{-1}Z(s) = \widehat{\beta}_{mcg}$ [3], et que le meilleur prédicteur linéaire sans biais de $Z(s_0)$ peut être écrit par

$$\chi' Z(s) \;=\; p_{ku}{}^{4}(Z;s_0) = x(s_0)'\widehat{\beta}_{mcg} + \sigma'\Sigma^{-1}\left(Z(s) - X(s)\widehat{\beta}_{mcg}\right). \qquad (1.19)$$

L'erreur quadratique moyenne de prédiction minimale fournit une mesure de l'incertitude associée au prédicteur universel de krigeage. Elle est, en substituant les poids optimaux, χ, donnée dans l'équation (1.18), dans l'équation (1.17), ce qui aboutit à la variance de krigeage

$$\begin{aligned}\sigma^2_{ku}(s_0) &= \chi'\Sigma\chi + \sigma_0 - 2\chi'\sigma \\ &= \sigma_0 - \sigma'\Sigma^{-1}\sigma \\ &\quad + \left(x(s_0)' - \sigma'\Sigma^{-1}X(s)\right)\left(X(s)'\Sigma^{-1}X(s)\right) - 1 \\ &\quad \times \left(x(s_0)' - \sigma'\Sigma^{-1}X(s)\right)'. \qquad (1.20)\end{aligned}$$

Pour prédire r variables, $Z(s_0)$, simultanément, nous étendons le modèle ci-dessus à

$$E\begin{bmatrix} Z(s) \\ Z(s_0)\end{bmatrix} = \begin{bmatrix} X(s)\beta \\ X(s_0)\beta\end{bmatrix} \qquad (1.21)$$

$$Var\begin{bmatrix} Z(s) \\ Z(s_0)\end{bmatrix} = \begin{bmatrix} \Sigma_{ZZ} & \Sigma_{Z0} \\ \Sigma_{0Z} & \Sigma_{00}\end{bmatrix}, \qquad (1.22)$$

avec Σ_{ZZ} étant la matrice de variance-covariance de dimension $n \times n$ des données, Σ_{00} celle de dimension $r \times r$ parmi les valeurs latentes, et Σ_{0Z} représente celle de dimension $r \times n$ entre les données et les inobservables. Avec ce modèle, la meilleure

3. Moindres carrés généralisés

4. "KU" est utilisé pour désigner le prédicteur universel de krigeage, afin de le distinguer des prédicteurs de krigeage simple et ordinaire.

prédiction linéaire sans biais (Goldberger [8] ; Gotway and Cressie [9]) est

$$\widehat{Z}(s_0) = X(s_0)\widehat{\beta}_{mcg} + \Sigma_{0Z}\Sigma_{ZZ}^{-1}\left(Z(s) - X(s)\widehat{\beta}_{mcg}\right), \tag{1.23}$$

et l'erreur quadratique moyenne de prédiction associée est donnée par

$$\begin{aligned} \mathrm{E}\left[\left(Z(s_0) - \widehat{Z}(s_0)\right)^2\right] &= \Sigma_{00} - \Sigma_{0Z}\Sigma_{ZZ}^{-1}\Sigma_{Z0} \\ &\quad + \left(X(s_0) - \Sigma_{0Z}\Sigma_{ZZ}^{-1}X(s)\right)\left(X(s)'\Sigma_{ZZ}^{-1}X(s)\right)^{-1} \\ &\quad \times \left(X(s_0) - \Sigma_{0Z}\Sigma_{ZZ}^{-1}X(s)\right)'. \end{aligned} \tag{1.24}$$

1.4.4 Krigeage en présence d'une tendance

Non-stationnarité de la moyenne

La plupart des méthodes de krigeage concernent des réalisations de processus stationnaires. Elles sont basées sur le modèle simple donné par l'équation (1.6) :

$$Z(x) = \mu + \epsilon(x), \tag{1.25}$$

dans laquelle μ, constante, est la moyenne, et $\epsilon(x)$ est une variable aléatoire de moyenne nulle et de variogramme $\gamma(h)$. Si le processus est stationnaire du second ordre, $\epsilon(x)$ a également une fonction de covariance $C(h)$, donnée par l'équation (1.7)). Nous nous appesantissons maintenant aux processus spatiaux dans lesquels μ varie. Il est à rappeler que certains processus spatiaux comportent une tendance, ou une " drift " comme on l'appelle communément en géostatistique ; ils ne sont pas stationnaires en général. La variation dans $Z(x)$ contient alors une composante systématique en plus de la composante aléatoire :

$$Z(x) = \mu(x) + \epsilon(x), \tag{1.26}$$

où $\mu(x)$, qui varie de façon régulière et est déterministe, remplace la moyenne, μ, dans l'équation (1.25). Dans ces conditions,

$$E\left[(Z(x) - Z(x+h))^2\right] \neq E\left[(\epsilon(x) - \epsilon(x+h))^2\right],$$

et les semi-variances brutes calculées par l'équation (1.12) seront des estimations biaisées de $\gamma(h)$, le variogramme des résidus de la tendance, c'est-à-dire de

$$\epsilon(x) = \mu(x) - Z(x), \tag{1.27}$$

Pour estimer $\gamma(h)$, sans biais, nous devons distinguer $\mu(x)$ de $\epsilon(x)$ qui sont inconnus. Nous ne disposons uniquement que des données $z(x_i)$, $i = 1, 2, ..., N$.

La tendance, $\mu(x)$, peut généralement être exprimée sous une forme fonctionnelle simple

$$\mu(x) = \sum_{k=0}^{K} \beta_k f_k(x), \tag{1.28}$$

dans laquelle les β_k, $k = 1, 2, ..., K$ sont des coefficients inconnus, et les $f_k(x)$ des fonctions choisies arbitrairement. Si nous combinons les équations (1.26) et (1.28), nous pouvons représenter un processus avec tendance par le modèle suivant

$$Z(x) = \mu(x) + \epsilon(x) = \sum_{k=0}^{K} \beta_k f_k(x) + \epsilon(x). \tag{1.29}$$

En général, une tendance spatiale peut être modélisée comme un polynôme d'ordre inférieur dans les coordonnées géographiques. Ainsi, dans le cas le plus simple d'une tendance linéaire, nous pouvons développer l'équation (1.29) comme suit

$$Z(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon(x), \tag{1.30}$$

dans laquelle x_1 et x_2 sont les coordonnées spatiales, et pour laquelle $K = 2$.

Si $K = 0$ alors $f_0 = 1$, $\mu(x) = \beta_0 = \mu$, et nous avons un processus stationnaire tel que représenté par l'équation (1.25) avec le variogramme habituel, qui est sans biais et que nous pouvons utiliser pour le krigeage ordinaire ou simple. Si $K > 0$, alors nous avons un problème plus complexe auquel nous devons trouver une solution.

Chapitre 2

Modélisation de la dépendance spatiale avec les copules

Dans ce chapitre, il sera question de passer en revue la modélisation de la dépendance spatiale avec les copules. En particulier, le concept de copule va être introduit et son utilisation dans l'analyse spatiale sera décrite.

2.1 Copules : bref survol

2.1.1 Définition et propriétés des copules

Le concept de copule est inhérent aux fonctions de répartition multidimensionnelles. Son importance dans la modélisation multivariée est mise en évidence dans un célèbre théorème dû à Sklar [22].

Théorème de Sklar

Soient $X_1, ..., X_d$ des variables aléatoires dont les lois marginales sont données par

$$F_i(x_i) = \mathbb{P}(X_i \leqslant x_i), \text{ pour } i = 1, ..., d,$$

et dont la loi jointe est définie par

$$F(x_1, ..., x_d) = \mathbb{P}(X_1 \leqslant x_1, ..., X_d \leqslant x_d).$$

Alors il existe une fonction $C : [0,1]^d \to [0,1]$ telle que, pour tout $(x_1, ..., x_d) \in \mathbb{R}^d$,

$$F(x_1, ..., x_d) = C\left\{F_1(x_1), ..., F_d(x_d)\right\}.$$

La fonction C est unique lorsque les lois marginales $F_1, ..., F_d$ sont continues.

La fonction C s'appelle la *copule* du vecteur $(X_1, ..., X_d)$ et contient toute l'information relative à la dépendance entre les composantes de ce vecteur. On peut donc extraire la copule d'une loi multivariée continue à l'aide de la formule

$$C(u_1, ..., u_d) = F\left\{F_1^{-1}(u_1), ..., F_d^{-1}(u_d)\right\}, \tag{2.1}$$

où $(u_1, ..., u_d) \in [0,1]^d$. Le théorème de Sklar révèle qu'une fonction de répartition se décompose en deux composantes : la copule et les lois marginales. La structure de dépendance, représentée par la copule, peut donc être étudiée indépendamment des marges.

Définition d'une copule

Définition 2.1. Une copule $C : [0,1]^d \to [0,1]$ est une fonction de répartition d-dimensionnelle dont les lois marginales sont uniformes sur $[0,1]$. En effet, supposons que les distributions marginales de $(X_1, ..., X_d)$ sont continues. L'équation 2.1 permet alors d'obtenir, pour tout $(u_1, ..., u_d) \in [0,1]^d$,

$$\begin{aligned} C(u_1, ..., u_d) &= \mathbb{P}\left(X_1 \leqslant F_1^{-1}(u_1), ..., X_d \leqslant F_d^{-1}(u_d)\right) \\ &= \mathbb{P}(F_1(X_1) \leqslant u_1, ..., F_d(X_d) \leqslant u_d), \end{aligned}$$

où $F_i(X_i)$ est une variable uniforme sur $[0,1]$, pour $i = 1, 2, ..., d$. Il en découle qu'une copule C est une fonction de répartition multivariée dont les marges sont uniformes sur $[0,1]$. Sa densité c, si elle existe, peut être obtenue en dérivant C successivement par rapport à $u_1, ..., u_d$ et elle peut également s'écrire

$$c(u_1, ..., u_d) = \frac{f\{F_1^{-1}(u_1), ..., F_d^{-1}(u_d)\}}{f_1\{F_1^{-1}(u_1)\} \times ... \times \{F_d^{-1}(u_d)\}},$$

où $f_i = dF_i$ est la densité marginale de X_i, $i = 1, ..., d$, et $f = dF$ la densité conjointe de $(X_1, ..., X_d)$.

La définition suivante découle du fait qu'une copule est une fonction de répartition multivariée.

Définition 2.2. Soient $(a_1, ..., a_d) \in [0,1]^d$ et $(b_1, ..., b_d) \in [0,1]^d$ vérifiant $a_i \leqslant b_i$ pour tout $i \in \{1, ..., d\}$. D'autre part, soit

$$\Delta_{a_k}^{b_k} C(u_1, ..., u_d) = C(u_1, ..., u_{k-1}, b_k, u_{k+1}, ..., u_d) - C(u_1, ..., u_{k-1}, a_k, u_{k+1}, ..., u_d).$$

Pour être une copule, une fonction $C : [0,1]^d \to [0,1]$ doit satisfaire les trois propriétés suivantes :

— Si $u_i = 0$ pour au moins un $i \in \{1, ..., d\}$, alors $C(u_1, ..., u_d) = 0$;

— $C(1, ..., 1, u_k, 1, ..., 1) = u_k$;

— $\Delta_{a_d}^{b_d}...\Delta_{a_1}^{b_1}C(u_1,...,u_d) \geqslant 0$.

Lorsque la copule de $(U_1,...,U_d)$ est C, la dernière condition peut s'interpréter comme

$$\Delta_{a_d}^{b_d}...\Delta_{a_1}^{b_1}C(u_1,...,u_d) = \mathbb{P}\left(a_1 \leqslant U_1 \leqslant b_1,...,a_d \leqslant U_d \leqslant b_d\right) \geqslant 0.$$

Dans le cas bidimensionnel, cette condition revient à

$$C(b_1,b_2) - C(b_1,a_2) - C(a_1,b_2) + C(a_1,a_2) \geqslant 0.$$

La proposition suivante établit qu'une copule est toujours comprise entre deux bornes.

Copule d'indépendance et bornes de Fréchet–Hoeffding

Si C est une copule, alors pour tout $(u_1,...,u_d) \in [0,1]^d$,

$$\max\left(\sum_{i=1}^{d} u_i - d + 1, 0\right) \leqslant C(u_1,...,u_d) \leqslant \min(u_1,...,u_d).$$

Les bornes supérieure et inférieure de cette proposition sont appelées les bornes de Fréchet- Hoeffding. La borne supérieure, donnée par $M(u_1,...,u_d) = \min(u_1,...,u_d)$, est elle-même une copule et représente la dépendance positive parfaite, c'est-à-dire la copule issue du cas extrême $U_1 \overset{d}{=} ... \overset{d}{=} U_d$. Dans le cas bidimensionnel, la borne inférieure, c'est-à-dire

$$W(u_1,u_2) = \max(u_1 + u_2 - 1, 0),$$

est également une copule et est associée à la dépendance négative parfaite, correspondant au cas où $U_1 \overset{d}{=} 1 - U_2$. A noter que lorsque $d > 2$, la borne inférieure n'est pas une copule.

Une caractéristique importante concernant les copules est leur invariance sous des transformations monotones croissantes.

Invariance des copules

Proposition 2.1. *Soient $\eta_1, ..., \eta_d$, des fonctions strictement croissantes. Si C est la copule de $(X_1, ..., X_d)$, alors la copule de $(\eta_1(X_1), ..., \eta_d(X_d))$ est également C.*

Démonstration. Soit $Y_i = \eta_i(X_i)$, pour $i = 1, ..., d$. La loi marginale de Y_i est donnée par $G_i(x) = F_i\left(\eta_i^{-1}(x)\right)$, où F_i est la loi marginale de X_i. La distribution jointe de $(Y_1, ..., Y_d)$ est telle que

$$G(x_1, ..., x_d) = \mathbb{P}\{\eta_1(X_1) \leqslant x_1, ..., \eta_d(X_d) \leqslant x_d\} = F\{\eta_1^{-1}(x_1), ..., \eta_d^{-1}(x_d)\},$$

où F est la loi jointe de $(X_1, ..., X_d)$. Suivant l'équation (2.1), la copule D de $(Y_1, ..., Y_d)$ est telle que, pour tout $(u_1, ..., u_d) \in [0, 1]^d$,

$$\begin{aligned} D(u_1, ..., u_d) &= G\{G_1^{-1}(u_1), ..., G_d^{-1}(u_d)\} \\ &= F\{\eta_1^{-1}\left(G_1^{-1}(u_1)\right), ..., \eta_d^{-1}\left(G_d^{-1}(u_d)\right)\} \\ &= F\{F_1^{-1}(u_1), ..., F_d^{-1}(u_d)\} \\ &= C(u_1, ..., u_d), \end{aligned}$$

ce qui complète la preuve. □

Symétrie radiale

Soit $(U_1, ..., U_d)$ un vecteur de variables aléatoires uniformes sur $[0, 1]$. Une copule C est dite symétrique radiale si pour $(U_1, ..., U_d) \sim^1 C$, on a également $(1-U_1, ..., 1-U_d) \sim C$.

La loi jointe de $(1 - U_1, ..., 1 - U_d)$, notée $\widehat{C}$, est appelée la copule de survie associée à C. Ainsi, une copule C est symétrique radiale si et seulement si $C = \widehat{C}$. Dans le cas où $d = 2$, on peut montrer que la copule de survie de C est donnée par

$$\widehat{C}(u_1, u_2) = u_1 + u_2 - 1 + C(1 - u_1, 1 - u_2).$$

Échangeabilité

Soient $\pi(1), ..., \pi(d)$, des permutations arbitraires des nombres entiers de 1 à d. Une copule C est dite échangeable si $C(u_1, ..., u_d) = C(u_{\pi(1)}, ..., u_{\pi(d)})$. Dans le cas bidimensionnel, l'échangeabilité est traduite par $C(u_1, u_2) = C(u_2, u_1)$ pour tout $(u_1, u_2) \in [0, 1]^2$ et on dit que C est symétrique par rapport à sa diagonale principale.

Si C est la copule d-dimensionnelle de $(X_1, ..., X_d)$, alors la copule de $(X_1, ..., X_{d-1})$ est donnée par $C(u_1, ..., u_{d-1}, 1)$. À partir de C, on peut donc obtenir de façon inductive l'expression de la copule de n'importe quel sous-ensemble de variables de $(X_1, ..., X_d)$. Ces copules de dimension inférieure à d sont appelées les copules sous-jacentes à C dans la définition suivante.

1. $\sim$ signifie ici 'même loi'

Fermeture sous les marges

Soit C, une copule d-dimensionnelle de $(X_1, ..., X_d)$ appartenant à une certaine famille de copules $\mathcal{F}$. La copule de $(X_1, ..., X_{d-1})$ est alors donnée par $C(u_1, ..., u_{d-1}, 1)$. À partir de C, on peut donc obtenir de façon inductive l'expression de la copule de n'importe quel sous-ensemble de variables de $(X_1, ..., X_d)$. Ces copules de dimension inférieure à d sont appelées les copules sous-jacentes à C. Lorsqu'elles appartiennent toutes à la famille $\mathcal{F}$, alors $\mathcal{F}$ est dite fermée sous les marges.

2.1.2 Mesures de dépendance

Puisqu'une copule C renferme toute l'information au sujet de la dépendance d'un vecteur aléatoire, une mesure qui tente de capter l'étendue de cette association devrait dépendre uniquement de la copule. Dans cette section, il s'agira de décrire quelques-unes de ces mesures dans le cas bidimensionnel ; les deux plus usuelles sont probablement le tau de Kendall et le rho de Spearman.

Le tau de Kendall et le rho de Spearman

Soit (X, Y), un vecteur aléatoire de distribution jointe F, de lois marginales continues F_X et F_Y et dont la copule sous-jacente à F est C. Soient également (X_1, Y_1) et (X_2, Y_2), des copies indépendantes et identiquement distribuées de (X, Y). Le tau de Kendall mesure la différence entre les probabilités de concordance et de discordance de (X_1, Y_1) et (X_2, Y_2). De manière formelle, il est défini par

$$\tau(X, Y) = \mathbb{P}\left\{(X_1 - X_2)(Y_1 - Y_2) > 0\right\} - \mathbb{P}\left\{(X_1 - X_2)(Y_1 - Y_2) < 0\right\}.$$

Le tau de Kendall peut s'écrire uniquement en terme de la copule C par

$$\tau(C) = 4 \int_0^1 \int_0^1 C(u,v)\, dC(u,v) - 1.$$

Le rho de Spearman, quant à lui, est défini par la corrélation entre les variables uniformisées $U = F_X(X)$ et $V = F_Y(Y)$, c'est-à-dire

$$\rho_S(X,Y) = \text{corr}\{F_X(X), F_Y(Y)\} = 12\text{E}(UV) - 3.$$

Cette mesure s'exprime également en terme de la copule de (X,Y) par

$$\rho_S(C) = 12 \int_0^1 \int_0^1 C(u,v)\, du\, dv - 3.$$

Sous la copule d'indépendance, on établit facilement que $\tau(\Pi) = \rho_S(\Pi) = 0$. De plus, en présence de dépendance positive parfaite, on a $\tau(M) = \rho_S(M) = 1$, tandis qu'en présence de dépendance négative parfaite, on obtient $\tau(W) = \rho_S(W) = -1$.

Le bêta de Blomqvist

Soient $\tilde{x}$ et $\tilde{y}$, les médianes des distributions marginales F_X et F_Y, respectivement. Le beta de Blomqvist est associé à la probabilité de concordance et de discordance entre (X,Y) et $(\tilde{x},\tilde{y})$. Sa définition exacte est

$$\beta(X,Y) = \mathbb{P}\left\{(X-\tilde{x})(Y-\tilde{y}) > 0\right\} - \mathbb{P}\left\{(X-\tilde{x})(Y-\tilde{y}) < 0\right\}.$$

Le beta de Blomqvist est une autre mesure de dépendance qui s'écrit en terme de la copule C. En effet, on peut montrer que

$$\beta(C) = 4C\left(\frac{1}{2}, \frac{1}{2}\right) - 1.$$

Indices de dépendance caudale

Les mesures de dépendance caudale sont utilisées afin de capturer la dépendance dans les queues des distributions bivariées. Plus spécifiquement, l'indice de dépendance caudale inférieur λ_L quantifie la probabilité d'observer une petite valeur de Y sachant que l'on a observé une petite valeur de X tandis que l'indice de dépendance caudale supérieure λ_U est associé à la probabilité d'observer une grande valeur de X sachant que l'on a observé une grande valeur de Y. Plus spécifiquement,

$$\lambda_L(X,Y) = \lim_{u\to 0^+} \mathbb{P}\left\{(X \leq F_X^{-1}(u) | Y \leq F_Y^{-1}(u)\right\}$$

et

$$\lambda_U(X,Y) = \lim_{u\to 1^-} \mathbb{P}\left\{(X > F_X^{-1}(u) | Y > F_Y^{-1}(u)\right\}.$$

On peut établir facilement que ces deux mesures s'écrivent en fonction de la copule C via

$$\lambda_L(C) = \lim_{u\to 0^+} \frac{C(u,u)}{u} \text{ et } \lambda_U(C) = 2 - \lim_{u\to 1^-} \frac{1-C(u,u)}{1-u}$$

2.1.3 Quelques familles de copules

La copule normale et autres modèles elliptiques

Soit ϕ_Σ, la densité de la loi normale d-dimensionnelle de moyennes nulles, de variances unitaires et de matrice de corrélation $\Sigma \in \mathbb{R}^{d\times d}$, c'est-à-dire

$$\phi_\Sigma(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}\mathbf{x}\Sigma^{-1}\mathbf{x}^\top\right), \text{ où } \mathbf{x} = (x_1, ..., x_d) \in \mathbb{R}^d.$$

La copule normale, encore appelée copule gaussienne, est la structure de dépendance extraite de la distribution normale multivariée de fonction de répartition donnée par

$$\Phi_\Sigma(\mathbf{x}) = \int_{-\infty}^{x_1} ... \int_{-\infty}^{x_d} \phi_\Sigma(s_1, ..., s_d)\, ds_d ... ds_1.$$

Dans ce cas, les distributions marginales sont toutes normales standardisées. Suivant le théorème de Sklar, la copule normale est alors donnée implicitement par

$$\begin{aligned} C_\Sigma^N(u_1, ...u_d) &= \Phi_\Sigma\left\{\Phi^{-1}(u_1), ..., \Phi^{-1}(u_d)\right\} \\ &= \int_{-\infty}^{\Phi^{-1}(u_1)} ... \int_{-\infty}^{\Phi^{-1}(u_d)} \phi_\Sigma(s)\, ds. \end{aligned} \tag{2.2}$$

où Φ est la fonction de répartition de la loi normale standardisée.

La copule normale est un cas particulier d'une classe de copule plus générale que sont les copules elliptiques (voir Embrechts et al. [6]). Ces dernières sont les copules sous-jacentes aux distributions elliptiques multidimensionnelles dont la densité, si elle existe, peut être écrite sous la forme

$$f_{\mu,\Sigma,h}(x) = \frac{1}{|\Sigma|^{1/2}} h\left\{\frac{1}{2}(x-\mu)\Sigma^{-1}(x-\mu)^\top\right\},$$

où $\mu \in \mathbb{R}^d$ est un vecteur de moyennes, $\Sigma \in \mathbb{R}^{d\times d}$ est une matrice de corrélation et $h : \mathbb{R}^+ \to \mathbb{R}^+$ est une fonction standardisée de telle sorte que

$$\int_0^\infty t^{\frac{d}{2}-1} h(t)\, dt = \frac{\Gamma(d/2)}{(2\pi)^{d/2}},$$

où Γ représente la fonction Gamma définie par $\gamma(s) = \int_0^\infty t^{s-1}\exp(-t)\, dt$. Il est important de noter que si $(X_1, ..., X_d)$ suit une loi elliptique multidimensionnelle, alors les lois marginales sont elliptiques univariées de la même famille. Ainsi, on peut donc invoquer le théorème de Sklar pour retrouver la forme de la copule elliptique associée à n'importe quelle densité de type $f_{\mu,\Sigma,h}$.

Un autre exemple de copule elliptique est la copule de Student à v degrés de liberté. Celle-ci est issue de la distribution de Student centrée multidimensionnelle dont la densité est

$$f_{v,\Sigma}(x) = \frac{\Gamma(\frac{v+2}{2})}{\Gamma(\frac{v}{2})(\pi v)^{d/2}|\Sigma|^{1/2}} \left(1 + \frac{x\Sigma^{-1}x^{\top}}{v}\right)^{-\frac{v+2}{2}}, \ v > 2.$$

Les lois marginales de cette distribution étant des lois de Student univariées à v degrés de liberté, alors la copule de Student est donnée par

$$C_{v,\Sigma}(u_1, ..., u_d) = \int_{-\infty}^{F_v^{-1}(u_1)} ... \int_{-\infty}^{F_v^{-d}(u_d)} f_{v,\Sigma}(s_1, ..., s_d)\, ds_d ... ds_1,$$

où F_v est la fonction de répartition de la distribution de Student univariée à v degrés de liberté.

Dans le cas bidimensionnel, les copules elliptiques sont indexées par le coefficient de corrélation $\rho = \Sigma_{12} = \Sigma_{21}$, $\rho \in (-1, 1)$. Si $C = C_\rho$ est une copule elliptique bidimensionnelle, alors le tau de Kendall est directement relié au coefficient de corrélation via l'expression (Alexander et al. [1])

$$\tau(C_\rho) = \frac{2}{\pi} \sin^{-1}(\rho).$$

Dans le cas de la copule normale de corrélation ρ, notée C_ρ^N , le rho de Spearman est donné par

$$\rho_S\left(C_\rho^N\right) = \frac{6}{\pi} \sin^{-1}(\frac{\rho}{2}).$$

Dans le cas de la copule de Student, les indices de dépendance caudale sont

$$\lambda_L(C_{v,\rho}) = \lambda_U(C_{v,\rho}) = 2F_{v+1}\left(-\sqrt{\frac{(v+1)(1-\rho)}{(1+\rho)}}\right).$$

Lorsque $v \to \infty$, la copule de Student correspond à la copule normale. Dans ce cas, les indices de dépendance caudale sont nuls.

Les copules Archimédiennes

Les copules *archimédiennes* forment une des classes de copules paramétriques les plus populaires. Une copule est dite *archimédienne* si elle peut être écrite sous la forme

$$C(u_1, ..., u_d) = \Psi\left\{\Psi^{-1}(u_1) + ... + \Psi^{-1}(u_d)\right\}, \tag{2.3}$$

où Ψ est une fonction appelée le générateur de la copule Archimédienne. Une condition suffisante pour que la forme (2.3) donne une copule, pour tout entier $d \geq 2$, est que la fonction Ψ soit complètement monotone, c'est-à-dire

$$(-1)^i \frac{d^i}{dt^i}\Psi(t) \geq 0 \text{ pour tout } t \in [0, \infty) \text{ et } i \in \{1, 2...\}.$$

Toutes les copules *archimédiennes* sont échangeables et fermées sous les marges. Si C en est une dont le générateur est Φ, alors le tau de Kendall s'exprime en fonction de son générateur par

$$\tau(C) = 1 - 4\int_0^{\Psi^{-1}(0)} t\left\{\Psi^{'}(t)\right\}^2 dt.$$

Les indices de dépendance caudale s'écrivent également en fonction du générateur par les expressions

$$\lambda_L(C) = \lim_{x\to\infty} \frac{\Psi(2x)}{\Psi(x)} \text{ et } \lambda_U(C) = 2 - \lim_{x\to 0^+} \frac{1-\Psi(2x)}{1-\Psi(x)}$$

Comme exemples de copules *archimédiennes* , on peut citer la copule de Clayton et celle de Frank. La première est définie par

$$C_\alpha(u_1, ..., u_d) = \left(\sum_{i=1}^{d} u_i^{-\alpha} - d + 1\right)^{-1/\alpha}, \alpha > 0.$$

Elle a pour générateur $\Psi_\alpha(t) = (1 + \alpha t)^{-1/\alpha}$, un tau de Kendall donné par $\tau(C_\alpha) = \alpha/(\alpha + 2)$. Son indice de dépendance caudale inférieure est $\lambda_L(C_\alpha) = 2^{-1/\alpha}$ et supérieure $\lambda_U(C_\alpha) = 0$. Cette copule n'est pas symétrique radiale.

La copule de Frank, quant à elle, est donnée par

$$C_\alpha(u_1,...,u_d) = -\frac{1}{\alpha}\ln\left[1+\frac{\{\exp(-\alpha u_1)-1\}\times...\times\{\exp(-\alpha u_d)-1\}}{\exp(-\alpha)-1}\right], \alpha>0,$$

et son générateur est

$$\Psi_\alpha(t) = -\frac{1}{\alpha}\log\left[\exp(-t)\{\exp(-\alpha)+1\}+1\right].$$

En dimension 2, la copule de Frank est symétrique radiale et ses deux indices de dépendance caudale sont nuls. Son tau de Kendall est donné par

$$\tau(C_\alpha) = 1-\frac{4\{D_1(\alpha)-1\}}{\alpha}, \text{ où } D_1(\alpha) = \frac{1}{\alpha}\int_0^\alpha \frac{x}{\exp(x)-1}\,dx.$$

2.2 Premiers pas : Bàrdossy (2006)

Bàrdossy (Bàrdossy [2]) est un pionnier quant à l'utilisation des copules en statistique spatiale comme alternative aux méthodes basées sur le variogramme. Soit $\{Z(\mathbf{x}) : \mathbf{x}\in S\}$, un champ aléatoire continu, stationnaire et isotrope. L'hypothèse de stationnarité implique que la loi marginale de la variable aléatoire $Z(\mathbf{x})$, qu'on notera F_Z, est la même en tout point $\mathbf{x}\in S$. D'après le Théorème de Sklar, l'unique copule C du vecteur de variables aléatoires $Z(\mathbf{x}_1),...,Z(\mathbf{x}_n)$ est telle que

$$\mathbb{P}\left(Z(\mathbf{x}_1)\le z_1,...,Z(\mathbf{x}_n)\le z_n\right) = C\left\{F_Z(z_1),...,F_Z(z_n)\right\}, \text{ où } (z_1,...,z_n)\in\mathbb{R}^n. \quad (2.4)$$

L'isotropie, quant à elle, révèle que le niveau de dépendance entre $Z(\mathbf{x}_i)$ et $Z(\mathbf{x}_j)$ est imputé à la distance entre $\mathbf{x}_i$ et $\mathbf{x}_j$, notée $\delta_{ij} = \|\mathbf{x}_i-\mathbf{x}_j\|, i\neq j\in\{1,...,n\}$. Ainsi, la copule spatiale C , telle qu'exprimée en (2.4), dépend des distances entre les postions d'observation $\mathbf{x}_i$; $i=1,...,n$. Une famille de copules permettra de modéliser la dépendance spatiale si elle est indexée par l'ensemble des matrices de corrélation

$\Sigma \in \mathbb{R}^{n\times n}$. Ainsi, on pose $\Sigma_{ij} = \Sigma_{ji} = g(\delta_{ij})$, où $g : [0,\infty) \to [0,1]$ est une fonction de lien telle que

$$g(\delta_{ij}) = \text{corr}\left\{Z(\mathbf{x}_i), Z(\mathbf{x}_j)\right\} = \frac{K(\delta_{ij})}{K(0)},$$

où K est la fonction de covariance définie par $K(\mathbf{x},\mathbf{y}) = \text{cov}\{Z(\mathbf{x}), Z(\mathbf{y})\}$, pour tout $\mathbf{x},\mathbf{y} \in S$. La fonction g retourne la corrélation attendue entre $Z(\mathbf{x}_i)$ et $Z(\mathbf{x}_j)$ selon la distance δ_{ij} entre $\mathbf{x}_i$ et $\mathbf{x}_j$. Étant donné que $K(.)$ est une fonction semi-définie positive, la fonction de lien doit également être semi-définie positive, c'est-à-dire

$$\sum_{i=1}^{n}\sum_{j=1}^{n} r_i r_j g(\|\mathbf{x}_i - \mathbf{x}_j\|) \geq 0,$$

pour tout $n \in \mathbb{N}$, tout $\mathbf{x}_1, ..., \mathbf{x}_n \in S$ et tous nombres réels $r_1, ..., r_n$; voir Cressie [5] pour plus de détails. Par conséquent, la matrice de corrélation Σ sera définie positive.

Une fonction de lien g est toujours continue sur $(0,\infty)$ et $g(0) = 1$. À l'origine, elle peut être soit continue ou discontinue. Lorsque

$$\lim_{\delta\to 0^+} g(\delta) = 1 - \epsilon, \text{ où } \epsilon > 0,$$

alors la fonction de lien est discontinue à l'origine et $\epsilon \in [0,1)$ est appelé l'effet pépite. Un grand effet pépite, c'est-à-dire un saut abrupt à l'origine, traduit une faible ressemblance entre des valeurs très rapprochées.

Il existe une panoplie de fonctions de lien proposées dans la littérature. La fonction de Matérn, dû à Matérn [15], est sans doute la plus populaire. Elle est donnée par

$$g_\nu(x) = \frac{x^\nu \mathcal{K}_\nu(x)}{2^{\nu-1}\Gamma(\nu)}, \nu > 0,$$

où $\mathcal{K}_\nu$ est la fonction de Bessel modifiée de deuxième espèce. Lorsque $\nu = 1/2$, on retrouve la fonction de lien exponentielle, c'est-à-dire $g_{1/2}(x) = \exp(-x)$. L'utilisation de la classe de fonctions Matérn a été fortement recommandée par Stein [23] car elle

permet une grande flexibilité quant à la modélisation de champs aléatoires, tout en conservant un nombre raisonnable de paramètres.

Soit $C_{g(\delta_{ij})}$, la copule de $Z(\mathbf{x}_i)$ et $Z(\mathbf{x}_j)$, $i \neq j \in \{1, ..., n\}$. Lorsque la distance δ_{ij} est infinitésimale, on s'attend à ce que $Z(\mathbf{x}_i)$ et $Z(\mathbf{x}_j)$ soient dépendantes positives parfaites. À cette fin, on impose que $C_{g(0)} = C_1 = M$, où M est la borne supérieure de Fréchet–Hoeffding bivariée. De surcroit, $Z(\mathbf{x}_i)$ et $Z(\mathbf{x}_j)$ devraient être indépendantes lorsque δ_{ij} est très grande. Ainsi, on impose également que

$$\lim_{\delta_{ij} \to \infty} C_{g(\delta_{ij})} = \Pi$$

où Π est la copule d'indépendance bidimensionnelle, pour tout $i \neq j \in \{1, ..., n\}$.

D'une part, les copules *archimédiennes* ne peuvent être utilisées pour la modélisation de la dépendance spatiale. Car, pour ces modèles, les niveaux de dépendance sont les mêmes pour chacune des distributions marginales bivariées. Les copules elliptiques, quant à elles, permettent une modélisation flexible des lois marginales bidimensionnelles par le biais d'une matrice de corrélation. Néanmoins, seule la copule normale atteint l'indépendance lorsque la corrélation est nulle. À cette fin, seule la copule normale est considérée parmi la famille des copules elliptiques pour modéliser la dépendance spatiale.

Par ailleurs, Bárdossy (2006) a introduit une autre copule possédant toutes les qualités requises afin de modéliser la dépendance spatiale. Il s'agit de la copule khi-deux. Elle est obtenue en élevant au carré les composantes d'un vecteur aléatoire issu de la distribution normale multidimensionnelle. Plus formellement, soit $(Z_1, ..., Z_n)$ un vecteur de variables aléatoires normales de moyennes nulles, de variances unitaires et de matrice de corrélation $\Sigma \in \mathbb{R}^{n \times n}$. La copule khi-carré est la structure de dépendance du vecteur aléatoire $(Y_1, ..., Y_n)$, où $Y_i = (Z_i + a)^2$, $i = 1, ..., n$, et $a \in \mathbb{R}$. Cette copule est donc indexée par une matrice de corrélation Σ et par un paramètre de forme a

que nous appelons paramètre de non-centralité.

2.3 Formalisation des copules spatiales : Quessy, Rivest & Toupin (2015)

Mise en contexte

Dans l'article de Bárdossy (2006), des modèles basés sur les copules sont construits pour les distributions finies associées à un champ aléatoire. Cette idée a été le point de départ de certains travaux dont Bárdossy et Li (2008), Li et al. (2011) et Kazianka et Pilz (2010), qui ont réalisé l'interpolation spatiale en utilisant la distribution conditionnelle complète. D'autres se sont appesanti sur l'estimation des paramètres, par exemple Kazianka et Pilz (2011) qui ont proposé une approche bayésienne et Bai et al. (2014) ayant utilisé la vraisemblance par paire dans le cas des données spatialement groupées. Plus récemment, des méthodes de prédiction basées sur des copules de vigne ont été proposées par Gräler et Pebesma (2011), Erhardt et al. (2015a), Erhardt et al. (2015b) et Gräler (2014). Toutefois, dans tous ces travaux, les distributions marginales sont spécifiées avec des paramètres inconnus.

C'est pour aller vers la formalisation que Toupin et al. (2015) ont proposé un article dont l'objectif principal est l'adoption d'une approche semi-paramétrique moins restrictive où les distributions marginales ne sont pas spécifiées. Dans ce cadre, Les marginaux sont considérés comme des paramètres de nuisance infinitésimaux, ce qui nécessite l'utilisation des rangs au lieu des observations originales. D'autre part, on note l'utilisation systématique d'une approche par paire pour l'inférence ; en d'autres termes, les procédures statistiques proposées n'utilisent que les informations fournies par les paires de localisation. Cela a pour avantage d'éviter le calcul de la distribution

multivariée, souvent difficile à réaliser.

Pour atteindre cet objectif, l'article propose (*i*) de décrire un cadre général pour les modèles de copules spatiales, (*ii*) d'étudier la performance des estimateurs de vraisemblance par paire basés sur le rang et (*iii*) d'effectuer une interpolation spatiale à des endroits non échantillonnés.

2.3.1 Un cadre général pour les champs aléatoires spatiaux basé sur les copules

Configuration générale

Soit $\{Z(\mathbf{x}) \mid \mathbf{x} \in S\}$ un champ aléatoire continu défini sur un sous-ensemble S de $\mathbb{R}^m$. Soit $\mathbf{x}_i \in S, i = 1, ..., n$ un treillis, c'est-à-dire une série d'emplacements dans S dont le champ aléatoire associé est noté $Z_1, ..., Z_n$ où $Z_i = Z(x_i), i \in 1, ..., n$. De surcroît, on suppose être en présence d'un champ aléatoire stationnaire, ce qui signifie que les distributions marginales sont les mêmes à chaque emplacement. Plus formellement, il existe une fonction de distribution F_Z telle que $\mathbb{P}(Z(\mathbf{z}_i) \leq z) = F_Z(\mathbf{z})$ pour chaque $i \in 1, ..., n$.

Un objectif crucial de la la modélisation des données reste la caractérisation de la distribution conjointe des données spatiales $Z = (Z_1, ..., Z_n)$, à savoir $H(z_1, ..., z_n) = \mathbb{P}(Z_1 \leq z_1, ..., Z_n \leq z_n)$. L'existence d'une unique copule C (voir (2.4)) a été établie par Bárdossy (2006) dans la section précédente. La forme de C caractérise complètement la dépendance entre un ensemble fini de lieux dans S. Tous les modèles multivariés ne sont pas susceptibles d'être des copules spatiales correctement définies. D'où l'objet de la sous-section suivante dans laquelle les copules spatiales sont définies en fonction d'une structure fonctionnelle générale indexée par une matrice de corrélation

qui caractérise la forme de la dépendance, et d'une fonction de liaison qui contrôle la force de la dépendance entre les paires de lieux en fonction de leur distance de séparation.

Une famille générale de copules spatiales

Dans la suite, considérons $\mathbb{N}^* = \mathbb{N} \setminus \{1\}$. Afin de modéliser la dépendance spatiale, il convient de disposer d'une famille de copules $F = \{C_{n,\Gamma_n} : n \in \mathbb{N}^*\}$ définie sur l'ensemble des matrices de corrélation $\{\Gamma_n\}$ de taille $n \in \mathbb{N}^*$. Cette exigence permettra d'inclure les distances entre les paires de lieux. De plus, afin de pouvoir éventuellement effectuer une interpolation spatiale, les modèles de cette famille doivent permettre de passer de $n-1$ à n dimensions ; pour cette raison, on suppose que cette famille est fermée sous les marges, soit

$$C_{n,\Gamma_n}(u_1, ..., u_{n-1}, 1) = C_{n-1,\Gamma_{n-1}^{-n}}(u_1, ..., u_{n-1}),$$

où $\Gamma_{n-1}^{-n} \in \mathbb{R}^{(n-1)\times(n-1)}$ est la matrice de corrélation des $n-1$ premières lignes et colonnes de Γ_n. La matrice de corrélation identité I_n donne la copule d'indépendance, c'est-à-dire $C_{n,I_n}(u_1, ..., u_n) = u_1 \times ... \times u_n$ pour chaque $n \in \mathbb{N}^*$. La famille $\mathcal{F}$ est supposée invariante sous les permutations, c'est-à-dire que si la copule de $Z = (Z_1, ..., Z_n)^\top$ est C_{n,Γ_n} , alors pour toute matrice de permutation $\Pi \in \mathbb{N}$, la copule de ΠZ est $C_{n,\Pi\Gamma_n\Pi^\top}$. Lorsque $n = 2$, cette propriété d'invariance de permutation implique que la copule bivariée $C_\rho = C_{2,\rho}$, où ρ est l'élément hors-diagonale de Γ_2, est symétrique, c'est-à-dire. $C_\rho(u, v) = C_\rho(v, u)$ pour tous les $(u, v) \in [0, 1]^2$. En d'autres termes, la structure de dépendance de (Z_i, Z_j) est la même que celle de (Z_j, Z_i) pour tout $i < j \in 1, ..., n$.

Une hypothèse naturelle en modélisation spatiale est de considérer que la force de la dépendance entre deux lieux est une fonction de leur distance ; c'est ce qu'on appelle

l'hypothèse d'isotropie. Soit $\Delta_n \in \mathbb{R}^{n\times n}$ la matrice de toutes les distances, c'est-à-dire que pour tous les $i, j \in 1, ..., n$, $(\Delta_n)_{ij} = \delta_{ij}$, où $\delta_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$ est la distance euclidienne entre $\mathbf{x}_i$ et $\mathbf{x}_j$. Dans la suite, l'intensité de la dépendance entre les lieux sera déterminée par une fonction de liaison, c'est-à-dire une fonction positive décroissante g définie sur $[0, \infty)$ qui satisfait $\lim_{x\to\infty} g(x) = 0$. Plus précisément, soit $\Gamma_n = \Gamma_{\theta,n}(\Delta_n) \in \mathbb{R}^{n\times n}$ telle que pour chaque $i, j \in \{1, ..., n\}$, $(\Gamma_{\theta,n})_{ij}$ pour un paramètre de plage donné $\theta > 0$ qui contrôle la vitesse à laquelle la corrélation décroît lorsque la distance entre les emplacements devient grande.

Mesure de la dépendance dans les modèles spatiaux

Une manière pratique et populaire de mesurer la dépendance dans une paire aléatoire à deux variables est de considérer la mesure d'association de Kendall. Plus précisément, si (X_1, Y_1) et(X_2, Y_2) sont des paires de variables aléatoires indépendantes à partir d'une distribution bivariée continue dont la copule sous-jacente est C, le tau de Kendall est défini comme suit

$$\tau(C) = 2\mathbb{P}\{(X_1 - X_2)(Y_1 - Y_2) > 0\} - 1.$$

La situation est plus compliquée dans les modèles de copule spatiale car le niveau de dépendance entre deux lieux dépend de leur distance. Afin de capturer cette caractéristique, définissons la portée effective associée à un modèle spatial donné comme étant la distance $\mathcal{D}$ entre deux lieux telle que la valeur de tau de Kendall soit égale à $0, 2$. En d'autres termes, pour un modèle spatial avec une copule sous-jacente $C_n\Gamma_{n,\theta}$, la portée effective $\mathcal{D}$ satisfait à la condition suivante

$$\tau\left(C_{g(\mathcal{D}/\theta)}\right) = 0, 2. \tag{2.5}$$

L'avantage majeur de $\mathcal{D}$ est qu'il peut être interprété indépendamment du modèle de copule. Ainsi, on peut dire que deux lieux séparés par une distance inférieure (resp. supérieure) à $\mathcal{D}$ ont un tau de Kendall supérieur (resp. inférieur) à $0,2$.

2.3.2 Fonctions de lien

On souhaite mesurer le niveau de dépendance entre deux locations en tenant en considération la distance qui les sépare, sous l'hypothèse d'isotropie. Pour cela, il convient d'utiliser une fonction de lien $g : [0, \infty[\to [0, 1[$ satisfaisant les conditions suivantes :

- $(\mathcal{C}_1)$ g est décroissante ;
- $(\mathcal{C}_2)$ $g(0) = 1$;
- $(\mathcal{C}_3)$ $g(x) \to 0$ quand $x \to \infty$.

On définit ensuite une matrice de corrélation Γ_θ telle que pour un certain $\theta > 0$, son élément à la position (j, j') est

$$(\Gamma_\theta)_{j,j'} = g\left(\frac{\Delta_{j,j'}}{\theta}\right).$$

Ici, θ est appelé le *paramètre de portée* dont le rôle est de contrôler la vitesse à laquelle le lien de dépendance fléchit à mesure que la distance entre deux sites devient grande. Pour plus de détails sur ce paramètre, voir [11]. Pour que la matrice Γ_θ soit définie positive pour chaque entier d, il faut que, selon Cressie [5], la fonction de lien g respecte la condition que pour tout $s_1, ..., s_d \in S$ et tout nombres réels $r_1, ..., r_d$,

$$\sum_{j,j'=1}^{d} r_j r'_j g\left(\Delta_{j,j'}\right) > 0.$$

Il existe une panoplie de fonctions de lien qui satisfont ces conditions. La plus usuelle

est probablement la fonction Matérn définie par

$$g_\nu^{Mat(x)} = \frac{x^\nu \mathcal{K}_\nu(x)}{2^{\nu-1}\Gamma(\nu)},$$

où $\mathcal{K}_\nu(x)$ est la fonction modifiée de Bessel de second type (voir Watson [24]) et $\Gamma(x) = \int_0^\infty t^{x-1}dt$ est la fonction Gamma. On obtient la fonction de lien Exponentielle $g^{Exp}(x) = \exp(-x)$ lorsque $\nu = 0,5$. Dans le présent mémoire, on va aussi considérer la fonction de lien gaussienne $g^{Gau}(x) = \exp(-x^2)$ ainsi que la fonction *relationnelle quadratique* $g_\nu^{RQ}(x) = (1+x^2)^{-\nu}$.

Enfin, on peut introduire un paramètre d'*effet de pépite*, à partir d'une fonction de lien g donnée. Il permet de tenir compte des variations à petite échelle ou encore des erreurs de mesure. Pour ce faire, on définit $\epsilon \in [0,1)$ comme le paramètre de pépite et on travaille avec la fonction de lien modifiée

$$g_\epsilon(x) = \epsilon \mathbb{1}(x = 0) + (1-\epsilon)g(x).$$

Ainsi la fonction g_ϵ présente un saut à $x = 0$ ($g\epsilon(0) = 1$), tandis que, pour $x > 0$ petit choisi arbitrairement, $g_\epsilon(x) = (1-\epsilon)g(x) \leqslant (1-\epsilon)g(0) = 1-\epsilon$.

2.3.3 Quelques modèles de copules spatiales

La copule spatiale normale

Pour tout $n \in \mathbb{N}^*$, soit Φ_{n,Γ_n} la fonction de densité normale standard à n dimensions avec la matrice de corrélation $\Gamma_n \in \mathbb{R}^{n\times n}$. Puisque les distributions marginales unidimensionnelles dans ce modèle sont Φ, c'est-à-dire la fonction densité de la distribution normale standard univariée, la copule spatiale normale à n dimensions est

donnée implicitement par

$$C^N_{n,\Gamma_n}(u_1, ..., u_n) = \Phi_{n,\Gamma_n}\left\{\Phi^{-1}(u_1), ..., \Phi^{-1}(u_n)\right\}.$$

La copule bivariée associée à une paire arbitraire est alors $C^N_{n,\rho}(u, v) = \Phi_\rho\{\Phi^{-1}(u), \Phi^{-1}(v)\}$, où, Φ_ρ désigne la densité de la distribution normale bivariée. Il est établi que le tau de Kendall dans ce modèle est $\tau(C^N_\rho) = (2/\pi)\sin^{-1}\rho$. Enfin, notons que de la définition de la portée effective dans (2.5), il en découle dans le modèle de copule spatiale normale avec fonction de liaison g,

$$\mathcal{D} = \theta g^{-1}\{\sin\Big(\frac{\pi}{10}\Big)\}. \tag{2.6}$$

Copules spatiales de khi-deux

La copule normale bivariée est radialement symétrique, c'est-à-dire que pour tout $\rho \in [-1, 1]$, on a $c^N_\rho(u, v) = c^N_\rho(1 - u, 1 - v)$ pour tout $u, v \in [0, 1]$. Cette propriété peut ne pas être valable pour certains ensembles de données spatiales, c'est-à-dire que les queues supérieure et inférieure de la structure de dépendance bivariée peuvent être différentes. On a alors besoin des modèles pour ces situations d'asymétrie de queue.

Afin de créer une sorte d'asymétrie à partir de la copule spatiale normale, Toupin (2015) et al. ont pris $X = (X_1, ..., X_n) \sim \Phi_{n,\Gamma_n}$ et défini, comme dans Bàrdossy (2006), la distribution du chi-deux comme la loi conjointe de

$$\mathbf{Y} = \big((X_1 + a)^2), ..., (X_n + a)^2)\big)$$

pour $a > 0$. Dans ce cas, le modèle n'est pas radialement symétrique. La copule associée à cette distribution a une forme non descriptible dans le cas général à n variables, ce qui motive l'utilisation de méthodes basées sur la dépendance par paire uniquement. Dans l'article de Toupin et Quessy (2015), il a été établi que la copule

Khi-deux à deux variables est

$$\begin{aligned} C^{\chi}_{\rho,a}(u,v) &= C^N_\rho\{u+h_a(u), v+h_a(v)\} + C^N_\rho\{h_a(u), h_a(v)\} \\ &\quad -C^N_\rho\{h_a(u), v+h_a(v)\} - C^N_\rho\{u+h_a(u), h_a(v)\}, \end{aligned} \tag{2.6}$$

où $h_a(u) = 1-\Phi\{\Psi_a^{-1}(1+u)+a\}$ et $\Psi_a(x) = \Phi(x+a)+\Phi(x-a)$. La copule normale apparaît comme un cas particulier de ce modèle lorsque $a \to \infty$, car alors $h_a(u) \to 0$ pour tout $u \in [0,1]$; ceci est vrai puisque $\Psi_a^{-1}(1+u)+a > a \to \infty$. Lorsque $a = 0$, on retrouve la copule khi-deux centrée dont l'expression peut être considérée comme

$$C^{\chi}_{\rho,0}(u,v) = 2C^N_\rho\left(\frac{1+u}{2}, \frac{1+v}{2}\right) - 2C^N_\rho\left(\frac{1-u}{2}, \frac{1+v}{2}\right) - u \tag{2.7}$$

Le tau de Kendall pour la copule khi-deux est donné par

$$\tau\left(C^{\chi}_{\rho,a}\right) = \tau\left(C^N_\rho\right)\left\{4\Phi_\rho(\sqrt{2}a, \sqrt{2}a) - 4\Phi(\sqrt{2}a) + 1\right\},$$

où $\tau\left(C^N_\rho\right) = \frac{2}{\pi}\sin^{-1}(\rho)$ est le tau de Kendall de la copule normale. Étant donné que $4\Phi_\rho(0,0)-1 = \tau\left(C^N_\rho\right)$, un cas particulier lorsque $a = 0$ entraîne que le tau de Kendall pour la copule khi-deux centrée est

$$\tau\left(C^{\chi}_{\rho,0}\right) = \left\{\tau\left(C^N_\rho\right)\right\}^2 = \left(\frac{2}{\pi}\sin^{-1}\rho\right)^2.$$

Ainsi, la portée effective pour les modèles de copules spatiales khi-deux centrées avec fonction de liaison g est liée au paramètre de portée θ par

$$\mathcal{D} = \theta g^{-1}\left(\sin\left(\frac{\pi}{2\sqrt{5}}\right)\right).$$

2.3.4 Un estimateur de vraisemblance par paire basé sur le rang

Description de l'estimateur

Considérons une famille de modèles de copules spatiales C_{n,Γ_n} qui satisfont aux conditions de la section 2.3.1. Ici, les éléments de la matrice de corrélation sont implicitement déterminés par une fonction de liaison paramétrique, à savoir $(\Gamma_n)_{ij} = g_\kappa(\delta_{ij}/\theta)$, où $\kappa = (\nu, \epsilon) \in A = (0, \infty) \times [0, 1)$ est un vecteur de paramètres qui comprend le paramètre de lissage et l'effet pépite, et $\theta \in \mathbb{R}^+$ est le paramètre de portée. De plus, la structure de dépendance qui sert de base à ces copules spatiales est également paramétrable, c'est-à-dire que $C_{n,\Gamma_n} = C_{\beta,n,\Gamma_n}$ pour $\beta \in B$. Dans la suite, ces copules sont notées $C_{\theta,\kappa,\beta}$. Le but ici est d'estimer le vecteur de paramètres inconnus $(\theta, \kappa, \beta) \in \mathbb{R}^+ \times A \times B$.

Pour un ensemble de lieux fixes $x_1, ..., x_n \in S$, soit $Z = (Z_1, ..., Z_n)$, avec $Z_i = Z(x_i)$. La densité conjointe de Z peut être écrite de la sorte $H_{\theta,\kappa,\beta}(z_1, ..., z_n) = C_{\theta,\kappa,\beta}\{F_Z(z_1), ..., F_Z(z_n)\}$. Une expression pour la densité associée à $H_{\theta,\kappa,\beta}$ s'obtient facilement par une différenciation directe en supposant que $f_Z = F'Z$ existe. Cependant, au lieu de maximiser la fonction de vraisemblance complète, l'article de Toupin (2015) propose de travailler avec ce qu'on appelle une vraisemblance par paire impliquant uniquement les densités bivariées. Pour ce faire, désignons $c_{\beta,g_\kappa(\delta/\theta)}$ comme étant la copule à deux variables associée à une paire donnée de Z dont la distance est δ. Alors une log-vraisemblance par paire basée sur le rang est donnée par

$$\mathcal{L}(\theta, \kappa, \beta) = \sum_{1 \le i < j \le n} \ln c_{\beta,g_\kappa(\delta/\theta)} \left\{F_n(Z_i), F_n(Z_j)\right\}, \tag{2.8}$$

où $F_n(z) = (n+1)^{-1} \sum_{i=1}^n \mathbb{I}(Z_i \le z)$ est la fonction de distribution empirique remise à l'échelle.

Un estimateur semi-paramétrique de maximum de vraisemblance de θ, κ, β basé sur $\mathcal{L}$ défini dans l'équation 2.8 est défini comme suit

$$\left(\widehat{\theta}_n, \widehat{\kappa}_n, \widehat{\beta}_n\right) = \underset{(\theta,\kappa,\beta)\in\mathbb{R}^+\times A\times B}{\arg\max} \mathcal{L}(\theta, \kappa, \beta). \tag{2.9}$$

Un autre avantage de travailler avec une vraisemblance par paire est la possibilité d'exclure les paires qui sont trop éloignées. L'idée est que l'information fournie par ces dernières sur les paramètres inconnus est limitée. A cet effet, soit $\xi \in (0, 1]$ un centile donné de l'ensemble des $n(n-1)/2$ distances δ_{ij}, $i < j \in 1, ..., n$. Définissons D_ξ comme la distance maximale que l'on veut prendre en compte. Ensuite, on considère la vraisemblance restreinte par paire

$$\mathcal{L}^\xi(\theta, \kappa, \beta) = \sum_{\substack{1\le i<j\le n:\\ \delta_{ij}\le D_\xi}} \ln c_{\beta, g_\kappa(\delta/\theta)} \left\{F_n(Z_i), F_n(Z_j)\right\},$$

et l'estimateur de vraisemblance semi-paramétrique correspondant

$$\left(\widehat{\theta}_n^\xi, \widehat{\kappa}_n^\xi, \widehat{\beta}_n^\xi\right) = \underset{(\theta,\kappa,\beta)\in\mathbb{R}^+\times A\times B}{\arg\max} \mathcal{L}^\xi(\theta, \kappa, \beta). \tag{2.10}$$

Dans la suite de l'article, les auteurs ont étudié l'efficacité des estimateurs définis dans 2.8 et 2.10 pour différents choix de $\xi \in (0, 1)$.

2.3.5 Interpolation spatiale par paires

L'objectif principal de la modélisation spatiale est la prédiction à des endroits où les données ne sont pas disponibles. Cette étape est appelée interpolation spatiale. Les méthodes d'interpolation sont généralement basées sur le champ aléatoire dit gaussien. Cela signifie que la distribution conjointe sur un ensemble fini d'emplacements est supposée être normale multivariée. Par conséquent, la structure de dépendance et les distributions marginales appartiennent toutes à la famille normale. Ces hypothèses

peuvent ne pas être satisfaites pour certains ensembles de données spatiales. La méthode proposée par Toupin offre plus de souplesse. C'est l'objet de cette sous-section.

La solution proposée : une approche par paire basée sur le rang

Soit $Z_1, ..., Zn$ les variables aléatoires correspondant à un champ aléatoire Z évalué aux emplacements $x_1, ..., x_n$. Supposons que la copule de $(Z_1, ..., Z_n)$ appartienne à une famille $\mathcal{F} = \{C_{n,\Gamma_n} : n \in \mathbb{N}^*\}$, où $(\Gamma_n)_{ij} = g_\kappa(\delta_{ij}/\theta)$ pour une certaine fonction de liaison g_κ. On note les densités de la copule bivariée dans ce modèle par $c_\beta, g_\kappa(\delta/\theta)$, où δ est la distance entre deux emplacements.

Le but ici est d'estimer la valeur Z_0 du champ aléatoire Z à l'endroit non échantillonné $x_0 \in S$. Pour cela, définissons pour chaque $i \in 1, ..., n$ la distance euclidienne $\delta_{0i} = \delta_{i0}$ entre les emplacements x_0 et x_i. Pour l'instant, supposons que la distribution marginale F_Z et sa densité associée f_Z sont connues. Pour chaque $i \in 1, ..., n$, la densité conditionnelle de Z_0 étant donné $Z_i = z$ est donc

$$f_{Z_0|Z_i}(z_0|z) = c_\beta, g_\kappa(\delta_{0i}/\theta)\{F_Z(z_0), F_Z(z)\}f_Z(z_0).$$

La valeur prédite de Z_0 basée sur la moyenne de cette distribution conditionnelle est la suivante

$$\mathbb{E}\left(Z_0|Z_i = z\right) = E_Z\left\{Z_0 c_\beta, g_\kappa(\delta_{0i}/\theta)\{F_Z(z_0), F_Z(z)\}\right\}.$$

Puisque F_Z n'est absolument pas spécifiée, on procède comme pour l'estimation des paramètres et on la remplace par la fonction de distribution empirique F_n remise à l'échelle. Une prédiction par rang de Z_0 basée sur la valeur prise par Z à l'emplacement x_i est alors la suivante

$$\widehat{Z}_0^{(i)} = \frac{1}{n}\sum_{j=1}^{n} Z_j c_\beta, g_\kappa(\delta_{0i}/\theta)\{F_n(Z_j), F_n(Z_i)\} = \sum_{j=1}^{n} \omega_j^{(i)} Z_j,$$

où $\omega_j^{(i)} Z_j = c_\beta, g_\kappa(\delta_{0i}/\theta)\{F_n(Z_j), F_n(Z_i)\}/n$. Afin que la somme des poids soit égale à l'unité, une version normalisée est utilisée à la place, à savoir

$$\widehat{Z}_0^{(i)} = \sum_{j=1}^{n} \tilde{\omega}_j^{(i)} Z_j, \tag{2.11}$$

où $\tilde{\omega}_j^{(i)} = \frac{\omega_j^{(i)}}{\omega_1^{(i)} + ... + \omega_n^{(i)}}$. Par conséquent, les prédictions se situent toujours entre le minimum et le maximum des valeurs observées. En pratique, le vecteur de paramètres (θ, κ, β) sera estimé par l'estimateur de vraisemblance par paires $\left(\widehat{\theta}_n^\xi, \widehat{\kappa}_n^\xi, \widehat{\beta}_n^\xi\right)$ pour un certain $\xi \in (0, 1]$.

Comme les observations trop éloignées de x_0 fournissent peu d'informations sur Z_0, ce ne sont pas toutes les observations qui seront utilisées pour l'interpolation spatiale. Si l'on considère que $m < n$ est le nombre d'observations dans un voisinage donné de x_0 avec un rayon r, une prédiction de Z_0 est

$$\widehat{Z}_0^m = \frac{1}{m} \sum_{i=1}^{n} \widehat{Z}_0^{(i)} \mathbb{I}(\delta_{0i} \leq r).$$

Un intervalle de prédiction peut facilement être construit pour $\widehat{Z}_0^m$. Pour ce faire, soit $L_v(u) = \partial C_\beta, g_\kappa(\delta/\theta)(u, v)/\partial v$ la distribution conditionnelle de $(U|V = v)$, où $(U, V) \to C_\beta, g_\kappa(\delta/\theta)$. Ensuite, on considère pour chaque $i \in 1, ..., n$ le centile conditionnel d'ordre $q \in (0, 1)$ donné par

$$\widehat{Z}_0^{(i),q} = F_n^{-1}\{L_{F_n(Z_i)}(q)\}.$$

Une version utilisant les m observations dans un voisinage de x_0 est alors

$$\widehat{Z}_0^{m,q} == \widehat{Z}_0^{(i),q} \mathbb{I}(\delta_{0i} \leq r).$$

Finalement, un intervalle de prédiction de $100 \times (1-\alpha)\%$ pour Z_0 est

$$\left[\widehat{Z}_0^{m,\alpha/2}, \widehat{Z}_0^{m,1-\alpha/2}\right].$$

En définitive, notons que Toupin a également utilisé une méthode alternative utilisant la distribution multivariée.

2.4 Copules factorielles spatiales

La modélisation des données spatiales requiert souvent des modèles flexibles, simples et faciles à interpréter. Ceux basés sur la normalité multivariée ont été largement utilisés pour modéliser lesdites données et pour l'interpolation à de nouveaux emplacements, ainsi que pour l'analyse d'incertitude. Toutefois, les modèles gaussiens ne tiennent pas compte des fortes dépendances dans les queues et des dépendances asymétriques entre les queues gauche et droite, que l'on trouve souvent dans les données réelles. Il est donc nécessaire d'élaborer des modèles plus flexibles qui conservent l'aspect attrayant des champs aléatoires gaussiens.

Pour surmonter ce problème, les copules peuvent être utilisées pour construire des distributions flexibles et multivariées. Néanmoins, la plupart des copules proposées dans la littérature ne conviennent pas à la modélisation des données spatiales. Par exemple, les copules *archimédiennes*, vues dans la sous-section 2.1.3, ont une structure de dépendance échangeable, tandis que les processus spatiaux ont généralement des dépendances plus fortes à de plus petites distances. La copule gaussienne $C(\mathbf{u}) = \Phi_\Sigma\{\Phi^{-1}(u_1), ..., \Phi^{-1}(u_n)\}$, bien que maniable et facile à interpréter, souffre de dépendance à la queue et est symétrique à la réflexion. La copule t de Student peut gérer la dépendance de la queue, mais est symétrique par réflexion, comme la copule

gaussienne. Par ailleurs, les copules de valeurs extrêmes sont dépendantes de la queue supérieure et asymétriques, mais indépendantes de la queue inférieure (Segers [21]). Elles sont difficiles à résoudre dans des dimensions élevées et se justifient théoriquement pour les extrêmes, et non pour l'ensemble des données (Castruccio et al. [3]). Quant aux copules de vigne (voir Nikoloulopoulos et al. [16] pour plus de détails), dont la distribution conjointe est construite à l'aide de copules bivariées à liaison conditionnelle, elles sont flexibles mais manquent d'interprétabilité et sont complexes à ajuster.

Les modèles de copules factorielles ont l'avantage de combiner flexibilité, parcimonie, interprétabilité, et maniabilité. Ils peuvent aussi capturer la dépendance de la queue et l'asymétrie de la queue (selon le facteur aléatoire latent), et sont basés sur des modèles géostatistiques classiques. C'est l'objet de cette section. Plus spécifiquement, il convient d'étudier deux modèles récents proposés par Krupski et al. [13] et Krupski et Genton[7] dont le dernier est l'extension du premier modèle.

2.4.1 Modèles de copules factorielles pour les données spatiales répliquées

Krupskii et al. (2017) ont introduit un modèle de copule pour les données spatiales avec des répliques et sans dépendance temporelle. Ce modèle combine la flexibilité d'une approche de modélisation par copules, l'interprétabilité et la parcimonie des modèles factoriels, la facilité d'utilisation de la copule gaussienne dans des dimensions élevées.

Le modèle et la copule correspondante sont basés sur le processus aléatoire suivant :

$$W(\mathbf{s}) = Z(\mathbf{s}) + V_0, \ \mathbf{s} \in \mathbb{R}^d, \tag{2.12}$$

où $Z(s)$ est un processus gaussien et $V_0 \sim F_{V_0}$ un facteur commun, qui ne dépend pas de l'emplacement spatial s. Lorsqu'il est observé à n emplacements spatiaux $s_1, ..., s_n \in S$, on peut poser $Z_j = Z(\mathbf{s}_j)$, $W_j = W(\mathbf{s}_j)$, $j = 1, ..., n$, et $Z = (Z_1, ..., Z_n)^T$, $W = (W_1, ..., W_n)^T$. Alors le modèle restreint de dimension finie peut être réécrit de la sorte

$$W_j = Z_j + V_0,\ j = 1, ..., n,\ Z \sim N(0, \Sigma_Z), \tag{2.13}$$

où Σ_Z est une matrice de corrélation et $V_0 \sim F_{V_0}$ est un facteur commun qui est indépendant de Z.

On peut vérifier que la matrice de corrélation de W est

$$\Sigma_W := cor(W) = (\Sigma_Z + \sigma_0^2)/(1 + \sigma_0^2),$$

où $\sigma_0^2 = var(V_0) < \infty$. Ainsi, le facteur aléatoire V_0 augmente la corrélation spatiale de manière uniforme ; ce qui peut ne pas être réaliste dans une grande région. V_0 peut être interprété comme un facteur sous-jacent affectant tous les emplacements spatiaux simultanément.

Distribution, densité et copule

En dimension n, la distribution, la densité, la copule et la densité de la copule de $(W_1, ..., W_n)^T$ sont respectivement

$$F_n^W(w_1, ..., w_n) = \int_{-\infty}^{\infty} \Phi_\Sigma(w_1 - v_0, ..., w_n - v_0) dF_{V_0}(v_0),$$

$$f_n^W(w_1, ..., w_n) = \int_{-\infty}^{\infty} \phi_\Sigma(w_1 - v_0, ..., w_n - v_0) dF_{V_0}(v_0),$$

$$C_n^W = (u_1, ..., u_n) = F_n^W\{(F_1^W)^{-1}(u_1), ..., (F_1^W)^{-1}(u_n)\},$$

$$C_n^W = (u_1, ..., u_n) = \frac{f_n^W\{(F_1^W)^{-1}(u_1), ..., (F_1^W)^{-1}(u_n)\}}{\prod_{j=1}^n f_1\{(F_1^W)^{-1}(u_j)\}}.$$

Pour certains choix de distribution de facteurs aléatoires F_{V_0} (par exemple, des facteurs exponentiels), les intégrales ci-dessus peuvent être calculées en forme fermée. Ce qui est pratiquement aussi rapide que le calcul d'une densité gaussienne. En général, ces intégrales unidimensionnelles peuvent être approchées de manière efficace et précise à l'aide de méthodes de Monte Carlo ou d'intégration finie.

Comportement de la queue des copules de facteurs

La distribution F_{V_0} détermine le comportement de la queue des copules de facteurs :

- Si $V_0 = v_0 \in \mathbb{R}$ presque sûrement, alors $W(s)$ est gaussien $\Rightarrow$ indépendance de la queue.
- Si $V_0 \sim \mathcal{N}(\mu_0, \sigma_0^2)$, si $W(s)$ est gaussien on a alors une indépendance de la queue.
- Si $var(V_0) >> 0$, $W(s)$ est dominé par $V_0 \Rightarrow$ dépendance parfaite.

Pour générer une dépendance de queue, nous devons utiliser une distribution pour V_0 qui a des queues plus fortes que la distribution normale. La densité normale a un ordre exponentiel quadratique de décroissance de queue, et la dépendance de queue peut être obtenue en utilisant une variable aléatoire, V_0, avec un ordre exponentiel sous-linéaire de décroissance de queue avec un ordre de décroissance exponentielle sous-linéaire, comme le montre la proposition 2.2.

Proposition 2.2. *Soit* $1 - F_{V_0}(v_0) \sim K v_0^\beta \exp(-\theta v_0^\alpha)$, $\alpha \geq 0$, $\beta \in \mathbb{R}$, $\theta > 0$, $K > 0$, $v_0 \to \infty$. *Soit* $\rho = \Sigma_{Z,1,2} < 1$.

- *Si* $0 < \alpha < 1$ *ou* $\alpha = 0$, $\beta < 0$, *alors la copule bivariée* $C_2^{\boldsymbol{W}}(u_1, u_2)$ *a une dépendance parfaite de la queue supérieure, c'est-à-dire,* $\lambda_U = 1$.
- *Si* $\alpha = 1$, *la copule* $C_2^{\boldsymbol{W}}(u_1, u_2)$ *a une dépendance de la queue supérieure avec* $\lambda_U = 2\Phi\left[-\theta\{(1-\rho)/2\}^{1/2}\right]$.

— *Si* $\alpha > 1$*, la copule* $C_2^{\boldsymbol{W}}(u_1, u_2)$ *est indépendante de la queue, c'est-à-dire que* $\lambda_U = 0$*. Des résultats similaires s'appliquent à la queue inférieure.*

Modèles spécifiques

Facteur de Weibull avec $F_{V_0}(v_0) = 1 - \exp(-\theta v_0^{\alpha})$, $v_0 > 0$: ce modèle interpole de la structure de dépendance gaussienne (indépendante de la queue) ($\rightarrow \infty$) à une dépendance parfaite de la queue supérieure ($\alpha < 1$), en passant par une dépendance non triviale de la queue supérieure ($\alpha = 1$).

Facteur exponentiel avec $F_{V_0}(v_0) = 1 - \exp(-\theta v_0)$, $v_0 > 0$: on a une dépendance de la queue supérieure ; la densité est disponible en forme fermée.

$V_0 = V_1 - V_2$, avec $V_1 \sim Exp(\theta_1) \perp\!\!\!\perp V_2 \sim Exp(\theta_2)$: il y a dépendance de la queue supérieure et inférieure, une asymétrie de la queue et la densité est disponible en forme fermée.

Facteur de Pareto avec $F_{V_0}(v_0) = 1 - (v_0/v_\star)^{\beta}$, $v_0 > v_\star$, $\beta < 0$: dépendance parfaite de la queue supérieure.

Inférence

Les marges peuvent être estimées de manière non paramétrique ou paramétrique (à condition de trouver un bon modèle paramétrique). Une famille de copules paramétrique peut être ajustée en utilisant l'inférence du maximum de vraisemblance (qui est très efficace pour les modèles à facteurs exponentiels). Les marges et la copule peuvent être estimées en deux étapes ou en une seule étape.

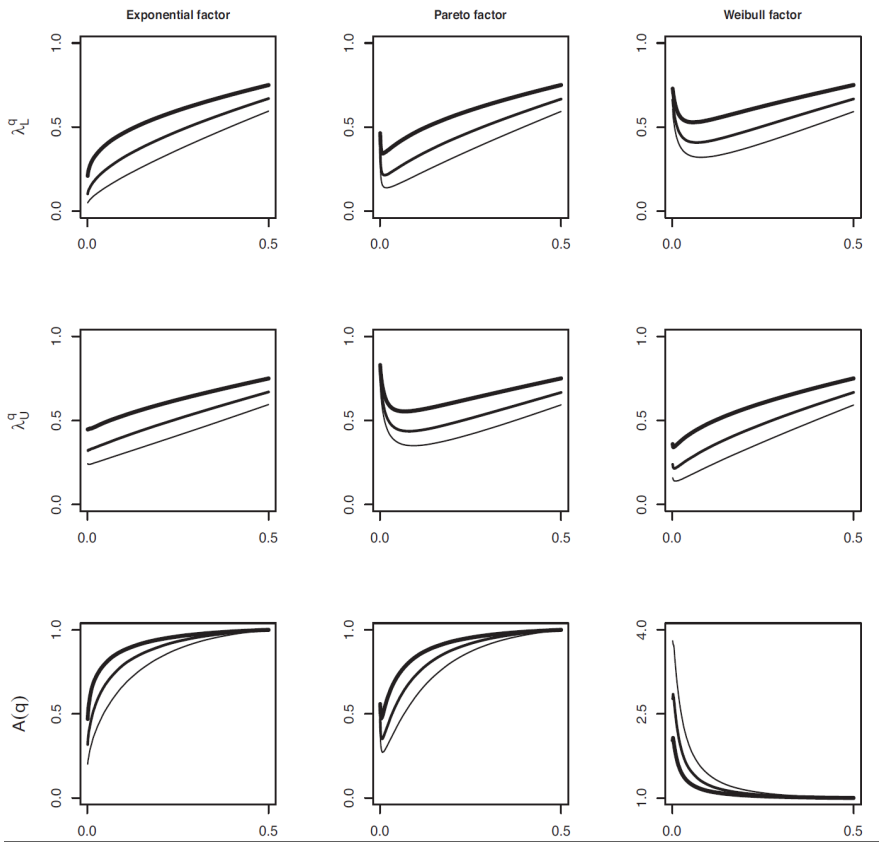


FIGURE 2.1 – λ_L^q(haut), λ_U^q (milieu) et $A(q)$ (bas), $0,001 \leq q \leq 0,5$, pour C_2^W dans les modèles 1 (gauche), 2 (milieu) et 3 (droite) ; ρ de Spearman $= 0,3$ (faible), $0,5$ (normal), $0,7$ (large).

L'estimation marginale non paramétrique offre une approche robuste lorsqu'il est difficile de choisir un modèle paramétrique. Cependant, il existe un léger biais asymptotique pour les échantillons finis. Quant à la procédure d'estimation en une étape, elle fournit une évaluation plus réaliste de l'incertitude globale des paramètres de la copule, mais l'optimisation est plus difficile et peut parfois être instable.

En définitive, l'article Krupskii et al. (2017) a proposé des modèles de copules factorielles pour les données spatiales répliquées et exploré leurs propriétés de dépendance de queue. Ces modèles permettent de remédier à certains des inconvénients des modèles spatiaux géostatistiques classiques. Les modèles de copules factorielles combinent flexibilité, parcimonie, interprétabilité, maniabilité, et peuvent capturer la dépendance de la queue et l'asymétrie de la queue (selon le facteur aléatoire latent), et sont basés

sur des modèles géostatistiques classiques. Cependant, ils peuvent ne pas être valables pour de grandes régions, car la non-corrélation complète ne peut pas être capturée à de grandes distances.

2.4.2 Modèles de copules factorielles pour les données avec dépendance spatio-temporelle

Pour combler les limites du modèle précédent, Krupski et Genton[7] en ont proposé une extension qui est basée sur le processus W mesuré dans l'espace et dans le temps :

$$W(s,t) = Z(s,t) + \alpha(s,t)\mathcal{E}_{\mathcal{P}(t)}, s \in \mathbb{R}^d, t \in \mathbb{R}_+. \tag{2.14}$$

Ici $Z(s,t)$ est un processus gaussien dans l'espace et dans le temps avec une moyenne nulle, une variance unitaire et une matrice de covariance Σ_Z ; $\alpha(s,t)$ est une fonction non aléatoire de l'espace (s) et du temps (t) ; $\mathcal{P}(t)$ est un processus de Poisson avec une fonction d'intensité $\Lambda(t)$ et $\mathcal{E} \sim_{i.i.d.} Exp(1)$ sont des facteurs exponentiels qui ne dépendent pas de $Z(s,t)$ ou de l'emplacement s.

Les facteurs $\mathcal{E}_{\mathcal{P}(t)}$ permettent de tenir compte de la dépendance de la queue pour la copule correspondant à la distribution conjointe du processus $W(s,t)$ mesurée à différents emplacements spatiaux et à différents moments. La fonction d'intensité, $\Lambda(t)$, du processus de Poisson $\mathcal{P}(t)$ contrôle le taux de décroissance de la dépendance dans le temps. La distribution exponentielle de $\mathcal{E}_{\mathcal{P}(t)}$ permet d'obtenir la densité de la copule conjointe dans ce modèle (2.14) sous une forme fermée, de sorte que les paramètres du modèle peuvent être estimés efficacement en utilisant l'approche du maximum de vraisemblance.

Considérons le processus $W(s,t)$, tel que défini dans (2.14), mesuré à n endroits différents $s_1, ..., s_n$ et à T instants différents $t_1 < t_2 < ... < t_T$. Pour simplifier, on

pose $W_{i,j} := W(s_i, t_j)$, $Z_{i,j} := Z(s_i, t_j)$ et $\alpha_{i,j} := \alpha(s_i, t_j)$. De (2.14), on a :

$$W_{i,j} = Z_{i,j} + \alpha_{i,j}\mathcal{E}_{\mathcal{P}(t_j)}, i = 1, ..., n, j = 1, ..., T. \tag{2.15}$$

D'après la définition du processus de Poisson P(t), nous voyons que

$$\lambda_{j_1,j_2} := pr\{\mathcal{P}(t_{j_1}) = \mathcal{P}(t_{j_2})\} = \exp\{-\int_{t_{j_1}}^{t_{j_2}} \Lambda(t)dt\}.$$

Soit Σ_Z et Σ_W les matrices de covariance des vecteurs respectives

$$Z = (Z_{1,1}, ..., Z_{n,1}, ..., Z_{1,T}, ..., Z_{n,T})^\top \text{ et } W = (W_{1,1}, ..., W_{n,1}, ..., W_{1,T}, ..., W_{n,T})^\top.$$

Il s'ensuit (avec $\lambda_{j,j} = 1$) que

$$\Sigma_W^{j_1,j_2} = \{\Sigma_Z^{j_1,j_2} + \lambda_{j_1,j_2}\alpha_1(t_{j_1})\alpha_1(t_{j_2})^\top\}/\{\alpha_2(t_{j_1})\alpha_2(t_{j_2})^\top\},\ 1 \leq j_1, j_2 \leq T,$$

où $\alpha_1(t) = \{\alpha_{s_1,t}, ..., \alpha_{s_n,t}\}$, $\alpha_2(t) = \{\mathbf{1}_n + \alpha_1(t)^2\}^{1/2}$, $\mathbf{1}_n$ est un vecteur unitaire de longueur n, et l'exposant j_1, j_2 désigne le bloc $n \times n$ de la matrice de covariance (Σ_Z ou Σ_W) correspondant aux covariances croisées aux instants t_{j_1} et t_{j_2}, pour différents emplacements.

La corrélation pour le processus $W(s, t)$ est plus grande pour des distances et des décalages temporels plus petits car $\left(\Sigma_Z^{j_1,j_2}\right)_{s_1,s_2}$ et λ_{j_1,j_2} deviennent grands lorsque les quantités $\|s_1 - s_2\|$ et $|t_{j_1} - t_{j_2}|$ sont petites. Il est supposé ici que la matrice de covariance Σ_Z est paramétrée de telle sorte que les corrélations sont plus faibles pour les paires d'observations présentant des distances et des décalages temporels plus importants. La plupart des modèles classiques de covariance spatio-temporelle satisfont à cette propriété.

Distribution, densité et copule

La copule C^W et sa densité, c_W, qui correspondent à la distribution conjointe, F^W, peuvent s'écrire comme suit :

$$C^W(u) = F^W(w),\ c^W(u) = f^W(w) \Big/ \prod_{i=1,j=1}^{n,T} f_{i,j}^W(w_{i,j}),$$

où $w = (w_{1,1}, ..., w_{n,1}, ..., w_{1,T}, ..., w_{n,T}) \top$, $u = (u_{1,1}, ..., u_{n,1}, ..., u_{1,T}, ..., u_{n,T}) \top$ et $w_{i,j} = (F_{i,j}^W)^{-1}(u_{i,j})$.

Comportement de la queue des copules de facteurs

La copule proposée a des propriétés de queue attrayantes car elle permet de contrôler la force de la dépendance dans la queue supérieure. Les mesures classiques de la dépendance de la queue sont les coefficients de dépendance de la queue inférieure et supérieure, λ_L et λ_U, définis pour une copule bivariée, $C_{1,2}$:

$$\lambda_L := \lim_{q \to 0} C_{1,2}(q, q)/q \in [0, 1]\ \lambda_U := \lim_{q \to 0} \bar{C}_{1,2}(1 - q, 1 - q)/q \in [0, 1],$$

où $\bar{C}_{u_1,u_2} := 1 - u_1 - u_2 + C_{1,2}(u_1, u_2)$ est la copule de survie. Si $\lambda_L > 0$ ($\lambda_U > 0$), on dit que la copule $C_{1,2}$ présente une dépendance dans la queue inférieure (supérieure). Pour la copule gaussienne, $\lambda_L = \lambda_U = 0$, et donc cette copule peut ne pas convenir pour modéliser des données avec une forte dépendance dans les queues. En même temps, la copule C^W permet une dépendance de la queue supérieure, et cette dépendance est plus faible avec une plus grande distance ou avec un plus grand décalage temporel.

Discussion

Avec ce modèle, des études de simulation ont montré que différents types de dépendance spatio-temporelle peuvent être obtenus, et que le taux de décroissance de la dépendance dans le temps peut être contrôlé à l'aide de la fonction d'intensité, $\Lambda(t)$, ainsi que des paramètres de la matrice de covariance Z. La fonction de vraisemblance pour le modèle proposé ne nécessite pas d'intégration numérique multivariée et l'estimation est donc assez simple à moins que le nombre de décalages temporels soit très important. En outre, ce modèle de copule peut être utilisé avec des marges univariées arbitraires ; ce qui permet une plus grande flexibilité dans la modélisation des données spatio-temporelles.

Malgré sa flexibilité, le modèle proposé nécessite des répliques pour estimer ses paramètres si le nombre de décalages temporels est faible. Avec un grand nombre de décalages temporels, l'estimation de ce modèle devient très difficile en raison du nombre exponentiellement croissant de termes à calculer pour la fonction de vraisemblance. Il est aussi à noter que ce modèle est ergodique dans le temps mais pas dans l'espace. Étant donné que dans les applications géostatistiques, les données n'ont souvent qu'une seule réplique, il peut être intéressant donc de trouver un modèle de copule pour les données spatio-temporelles qui ne nécessite pas de réplication pour l'estimation.

Chapitre 3

Une librairie Matlab pour l'analyse de données spatiales avec les copules

Ce chapitre revêt une importance capitale pour la suite dans la mesure où il y est décrit les différents programmes qui vont servir à l'analyse de données spatiales avec les copules. Il s'agit de présenter la librairie *Tools in spatial statistics* qui est un ensemble de programmes permettant d'effectuer une analyse spatiale basée sur les copules. Elle est inspirée des articles de Kazianka et Pilz [12] et Quessy et al. [19] et est implémentée dans le langage de programmation Matlab dont le choix repose sur son environnement flexible pour le calcul mathématique et technique.

L'objectif principal de cette librairie est de répondre aux besoins des chercheurs analysant des données spatiales ayant notamment une distribution non gaussienne, et pouvant présenter une absence d'isotropie.

3.1 Programmes utilitaires

Le tableau A.1 de l'annexe A dresse la liste des fonctions se trouvant dans la librairie *Tools in spatial statistics*. Cette panoplie de programmes permet d'effectuer une modélisation et une interpolation spatiales à l'aide de copules. De surcroit, on y trouve des fonctions pour effectuer une analyse exploratoire des données ainsi que celles qui permettent la validation croisée des modèles. Toutefois, seules les fonctions principales d'estimation des paramètres et de prédiction spatiale seront décrites dans ce chapitre, le reste se trouvant en Annexe.

3.1.1 SpatialTools_CreationOfAGrid

Le programme génère des positions sur le carré de taille 100×100 et calcule la matrice de toutes les distances entre ces positions. Cette démarche est mise en œuvre dans la fonction *SpatialTools_ CreationOfAGrid()*. Étant donné un nombre d d'emplacements choisi arbitrairement, on aura deux outputs : *Coord* qui est une matrice $d \times 2$ des coordonnées des d emplacements ; et D, une matrice symétrique $d \times d$ des distances entre les d emplacements. Pour ce faire, le programme propose trois étapes :

1. Construction d'une grille 67×67 de $0,5$ à $99,5$.
2. Sélection des d emplacements et ajout d'une perturbation
3. Calcul des distances entre les lieux sélectionnés.

```
1 function [Coord,D] = SpatialTools_CreationOfAGrid(d)
2
3 % Construction of a 67x67 grid from 0.5 to 99.5
4 div = 67;
5 x = zeros(div,1); y = zeros(div,1);
6 for j=1:div
```

```
    x(j) = 0.5 + 1.5*(j-1);
    y(j) = x(j);
end

P = zeros(div^2,2);
ell = 1;
for j=1:div
    for k=1:div
        P(ell,:) = [x(j),y(k)];
        ell = ell + 1;
    end
end

% Selection of the d locations and addition of a perturbation
a = randperm(div^2);
Coord = zeros(d,2);
for j=1:d
    Coord(j,1) = P(a(j),1) + (rand - 0.5);
    Coord(j,2) = P(a(j),2) + (rand - 0.5);
end

% Calculation of the distances between the selected locations
D = zeros(d,d);
for j=1:d-1
    for k=j+1:d
        D(j,k) = norm(Coord(j,:)-Coord(k,:));
        D(k,j) = D(j,k);
    end
end
```

Si on choisit $d = 5$, par exemple, on obtient les matrices *Coord* et D suivantes :

```
>> [Coord,D] = SpatialTools_CreationOfAGrid(5)

Coord =

   60.2353    7.6810
   24.1478   90.2244
   90.9081   26.3171
   51.3015   33.9837
   76.6108    7.5847


D =

         0   90.0872   35.8904   27.7784   16.3758
   90.0872         0   92.4179   62.4527   97.8861
   35.8904   92.4179         0   40.3417   23.5652
   27.7784   62.4527   40.3417         0   36.5714
   16.3758   97.8861   23.5652   36.5714         0
```

FIGURE 3.1 – Illustration du programme *SpatialTools_ CreationOfAGrid()*, pour $d = 5$ stations.

3.1.2 SpatialTools_Link

Le programme renvoie la valeur d'une fonction de lien calculée en x. Ainsi, la fonction *SpatialTools_ Link()* fait appel à quatre (4) arguments :

- **Link** : Either Matérn ('Mat'), Rational quadratic ('RQ') ou Exponential ('Exp').
- **nu** : Paramètre de lissage spécifique à la fonction.
- **nugget** : effet de pépite [1].
- **x** : Argument de la fonction de lien sélectionnée.

```
1 function v = SpatialTools_Link(Link,nu,nugget,x)
2
3 x = max(x,0);
4
5 if strcmp(Link,'Mat')
```

1. L'effet pépite est un terme géostatistique utilisé pour décrire la variabilité observée entre des échantillons très proches les uns des autres.

```
    f = (1/(2^(nu-1)*gamma(nu)))*(x^nu)*besselk(nu,x);
end
if strcmp(Link,'RQ')
    f = (1 + x^2)^(-nu);
end

if strcmp(Link,'Exp')
    f = exp(-x);
end

v = (1-nugget)*f;
```

Pour $Link =' Mat'$, $nu = 100$, $nugget = 0.5$ et $x = 10$, on obtient la valeur de $v = 0,3885$:

```
>> v = SpatialTools_Link('Mat',100,0.5,10)

v =

    0.3885
```

FIGURE 3.2 – Illustration du programme *SpatialTools_ Link()*.

3.1.3 SpatialTools_PositiveDefinite

Le programme indique si une matrice symétrique est définie positive, et si non, la transforme en une matrice définie positive selon Higham (2002).

La fonction *SpatialTools_ PositiveDefinite()* fait appel à un seul argument **Sigma** qui est une matrice symétrique de taille $d \times d$. Elle a deux sorties : PD indiquant si la matrice $Sigma$ est positive définie ou non ; et $SigmaMod$ qui une transformation de $Sigma$ en une matrice positive définie dans le cas où on est en face d'une matrice

négative définie.

```
function [PD,SigmaMod] = SpatialTools_PositiveDefinite(Sigma)

d = length(Sigma);
eigen = eig(Sigma);

NegativeEigen = sum(eigen<0);

if NegativeEigen == 0
    PD = 1;
    SigmaMod = Sigma;
else
    PD = 0;
    SigmaMod = Sigma + abs(eigen(1))*eye(d);
end
```

Par exemple, pour une matrice symétrique *Sigma* de taille 4×4 donnée, on obtient le résultat suivant :

```
>> Sigma = toeplitz(randi([0 9],4,1))

Sigma =

     1     0     5     1
     0     1     0     5
     5     0     1     0
     1     5     0     1

>> [PD,SigmaMod] = SpatialTools_PositiveDefinite(Sigma)

PD =

     0


SigmaMod =

    5.5249         0    5.0000    1.0000
         0    5.5249         0    5.0000
    5.0000         0    5.5249         0
    1.0000    5.0000         0    5.5249
```

FIGURE 3.3 – Illustration du programme *SpatialTools_ PositiveDefinite()*.

3.1.4 TestOptions

Cette procédure permet la mise à jour des options utilisées dans toute la suite.

```
function options=testOptions(options)

% Updates the options structure array which is used throughout the
    code.
%
% options.tol........the tolerance level at which to stop the
    optimization
%                    procedures
% options.print......intensity of the notification about progam
    status
%                    (1,2,3)
% options.optim......optimset array for tuning optimization
    procedures
% options.parallel...whether or not to use the Parallel Computing
    Toolbox
%
% Inputs:
%
% options.......options structure array to check
%
% Outputs:
%
% options......updated options
%

    if isempty(options) || ~isstruct(options)
        options.tol=0.001;
        options.print=2;
        options.parallel=0;
        options.optim=optimset('LargeScale','off','MaxIter',1000,'
    MaxFunEvals',5000,'Display','off','Algorithm','active-set');
    else
```

```
27          if ~isfield(options,'tol') || ~isnumeric(options.tol) ||
    options.tol<=0
28              options.tol=0.001;
29          end
30          if ~isfield(options,'print') || ~isnumeric(options.print)
31              options.print=2;
32          end
33          if ~isfield(options,'optim')
34              options.optim=optimset('LargeScale','off','MaxIter',1000,
    'MaxFunEvals',5000,'Display','off','Algorithm','active-set');
35          end
36          if ~isfield(options,'parallel')
37              options.parallel=0;
38          end
39       end
```

3.1.5 Initialize

La fonction *initialize()* est la pierre angulaire de la modélisation spatiale proposée ici. Elle permet de proposer automatiquement une distribution marginale appropriée, une fonction de corrélation adéquate ainsi que les paramètres initiaux pour l'estimation par maximum de vraisemblance. Plus formellement, la fonction met en place le modèle de copule spatiale en estimant d'emblée les marges et les paramètres de corrélation :

```
1 function [model options]= initialize(data,options)
2
3 % Inputs:
4 %
5 % data.........the data set as a structure array of the format data.x
    (x-coordinates), data.y
6 %              (y-coordinates), data.z (measured values)
7 %
8 % options.....options from testOptions
```

```
9  %
10 % Outputs:
11 %
12 % model.......contains the parameters of the spatial copula model
13 %             model.anisotropy.params.....angle and ratio for
    geometric anisotropy
14 %             model.anisotropy.upper......upper bounds for
    optimization
15 %             model.anisotropy.lower......lower bounds for
    optimization
16 %             model.anisotropy.const......0,1 depending on whether or
     not optimization should be performed
17 %
18 %             model.trend.params.....parameters for spatial trend
19 %             model.trend.upper......upper bounds for optimization
20 %             model.trend.lower......lower bounds for optimization
21 %             model.trend.F..........design matrix
22 %             model.trend.const......0,1 depending on whether or not
    optimization should be performed
23 %
24 %             model.correlation.params.....parameters of the
    correlation function
25 %             model.correlation.model......name of the correlation
    model as specified in the function correlCovMat
26 %             model.correlation.upper......upper bounds for
    optimization
27 %             model.correlation.lower......lower bounds for
    optimization
28 %             model.correlation.const......0,1 depending on whether
    or not optimization should be performed
29 %
30 %             model.margin.params.....parameters of the marginal
    distribution excluding the trend parameter
31 %             model.margin.name.......margin name e.g. 'norm', '
    gevMod', 'logn', 'boxcox', 'gamMod', ...
32 %             model.margin.upper......upper bounds for optimization
```

```
33 %             model.margin.lower......lower bounds for optimization
34 %             model.margin.const......0,1 depending on whether or not
       optimization should be performed
35 %
36 %             model.copula.params.....copula specific parameter
37 %             model.copula.name.......copula name: 'Gaussian', 'Chi2
      ', 'Student'
38 %             model.copula.upper......upper bounds for optimization
39 %             model.copula.lower......lower bounds for optimization
40 %             model.copula.const......0,1 depending on whether or not
       optimization should be performed
41 %
42 % options.....options from testOptions
43 %
44 %
45
46     if nargin<2
47         options=testOptions([]);
48     else
49         options=testOptions(options);
50     end
51
52     copula=[];
53
54 % decide whether to include geometric anisotropy into the analysis
55     [anisotropy.params do]=estimateAnisotropy(data);
56     anisotropy.lower=[0 1];
57     anisotropy.upper=[pi Inf];
58     anisotropy.const=0;
59
60     tempData=data;
61     if do==1
62         xy=[cos(anisotropy.params(1)),sin(anisotropy.params(1));-
      anisotropy.params(2)*sin(anisotropy.params(1)),anisotropy.params
      (2)*cos(anisotropy.params(1))]*[data.x';data.y'];
63         tempData.x=xy(1,:)';
```

```
64          tempData.y=xy(2,:)';
65      else
66          anisotropy.params=[0 1];
67      end
68
69      empvario=empiricalVariogram(tempData);
70
71  % Select an appropriate correlation function according to a mean
      squared error (MSE) criterion
72      models=cell(0);
73      models{1}='Gau';
74      models{2}='Exp';
75      models{3}='Sph';
76      models{4}='Mat';
77      MSEopt=Inf;
78
79      range=0.1*sqrt((max(tempData.x) - min(tempData.x))^2 + (max(
      tempData.y) - min(tempData.y))^2);
80      sill=mean([max(empvario.v),median(empvario.v)]);
81      nugget=min(empvario.v)/sill;
82      correlation=[];
83      for i=1:length(models)
84          temp.model=models{i};
85          if strcmp(temp.model,'Mat')
86                  if ~isempty(correlation.params)
87                      temp.params=[correlation.params 0.5];
88                  else
89                      temp.params=[nugget range 0.5];
90                  end
91                  temp.upper=[0.99 Inf 10];
92                  temp.lower=[0.001 0.001 0.1];
93          else
94              temp.params=[nugget range];
95              temp.upper=[1 Inf];
96              temp.lower=[0.001 0.001];
97
```

```
98          end
99          [params MSE]=fitVariogram(empvario,temp,options);
100         if MSE<MSEopt
101             correlation=temp;
102             correlation.params=params;
103             MSEopt=MSE;
104         end
105     end
106     correlation.const=0;
107
108 % find an appropriate univariate marginal distribution function
    family
109     [model.trend model.margin]=findmargins(data,options);
110     model.correlation=correlation;
111     model.copula=copula;
112     model.anisotropy=anisotropy;
113
114 function [trend margin]=findmargins(data,options)
115
116     margins=cell(0);
117    if sum(data.z<=0)>0
118         margins{1}='norm';
119         margins{2}='gev';
120    else
121         margins{1}='norm';
122         margins{2}='gev';
123         margins{3}='logn';
124         margins{4}='gam';
125         margins{5}='boxcox';
126    end
127
128   likeOpt=-Inf;
129     for i=1:length(margins)
130     [params like]=eval([margins{i} 'testing(data,options);']);
131         if(like>likeOpt)
132       margin.name=margins{i};
```

```
133             margin.params=params(2:end);
134             bounds=findmarginbounds(margin.name);
135             margin.upper=bounds.upper(2:end);
136             margin.lower=bounds.lower(2:end);
137             trend.upper=bounds.upper(1);
138             trend.lower=bounds.lower(1);
139             trend.params=params(1);
140             likeOpt=like;
141         end
142     end
143     trend.F=ones(length(data.z),1);
144     trend.const=0;
145     margin.const=0;
146
147
148 function bounds = findmarginbounds(name)
149   switch name
150         case 'norm'
151             bounds.upper=[Inf Inf];
152             bounds.lower=[-Inf 0.001];
153         case 'logn'
154             bounds.upper=[Inf Inf];
155             bounds.lower=[-Inf 0.001];
156         case 'gam'
157             bounds.upper=[Inf Inf];
158             bounds.lower=[0.001,0.001];
159         case 'gev'
160             bounds.upper=[Inf Inf Inf];
161             bounds.lower=[-Inf 0.001 -Inf];
162         case 'boxcox'
163             bounds.upper=[Inf Inf 10];
164             bounds.lower=[-Inf 0.001 0.001];
165   end
166
167 function [params like]=normtesting(data,options)
168     params=[mean(data.z) std(data.z)];
```

```
169     like=sum(log(normpdf(data.z,params(1),params(2))));
170
171 function [params like]=logntesting(data,options)
172     params=[mean(log(data.z)) std(log(data.z))];
173     like=sum(log(lognpdf(data.z,params(1),params(2))));
174
175     function [params like]=gamtesting(data,options)
176     params=gamfit(data.z);
177     like=sum(log(gampdf(data.z,params(1),params(2))));
178     params=[0 params];
179
180 function [params like]=gevtesting(data,options)
181     params=gevfit(data.z);
182     like=sum(log(gevpdf(data.z,params(1),params(2),params(3))));
183     params=[params(3) params(2) params(1)];
184
185 function [params like]=boxcoxtesting(data,options)
186     [lambda,like]=fmincon(@criteriumBoxCox,1,[],[],[],[],0.001,10,[],
    options.optim,data.z,options);
187     params=[mean((data.z.^lambda-1)./lambda) std((data.z.^lambda-1)./
    lambda) lambda];
188     like=-like;
189
190 function val=criteriumBoxCox(lambda,z,options)
191     mu=mean((z.^lambda-1)./lambda);
192     sd=std((z.^lambda-1)./lambda);
193     val=-sum(log(boxcoxpdf(z,mu,sd,lambda)));
194     if options.print>=2
195         disp(num2str([val mu sd lambda]));
196     end
```

3.2 Simulation de champs aléatoires à base de copules

Générer des champs aléatoires à partir d'une copule spatiale donnée est crucial pour leur visualisation. C'est l'objet de cette section.

3.2.1 SpatialTools_SimulationChi2

Le programme a pour but de générer des champs aléatoires répliqués à partir de la copule du Chi-deux. Ainsi, la fonction *SpatialTools_ SimulationChi2()* fait appel à six (6) arguments :

- **n** : Nombre de distributions identiquement indépendantes (i.i.d) répliquées.
- **a** : Valeur du paramètre de non-centralité.
- **Link** : Either Matérn ('Mat'), Rational quadratic ('RQ') ou Exponential ('Exp').
- **LinkParam** :
  - 1ıère entrée : **nu**, paramètre de lissage spécifique à la fonction (valeur dans $(0, Inf)$).
  - 2ıème entrée : **nugget**, effet de pépite(valeur dans $[0, 1)$).
- **ER** : Valeur de la portée effective.
- **D** : Matrice symétrique $d \times d$ des distances entre les d stations.

```
function Z = SpatialTools_SimulationChi2(n,a,Link,LinkParam,ER,D)

% Necessary procedures
%    BivCopula_InversionKendall
%    SpatialTools_Link
%    PairwiseCopulaNonCentralSquared_Simulation

TauInv = BivCopula_InversionKendall(2,.2,0);
theta = ER / TauInv;

d = length(D);
nu = LinkParam(1);
nugget = LinkParam(2);

Sigma = ones(d,d);
for j=1:d-1
    for k=j+1:d
        x = D(j,k) / theta;
        Sigma(j,k) = SpatialTools_Link(Link,nu,nugget,x);
        Sigma(k,j) = Sigma(j,k);
    end
end
Z = PairwiseCopulaNonCentralSquared_Simulation(n,2,Sigma,0,a*ones(1,d
   ),'Norm');
```

Par exemple, pour une matrice symétrique D de taille 5×5 donnée, on obtient le champ aléatoire simulé Z de taille $n(= 10) \times 5$ suivant :

```
>> D

D =

         0   65.1233  112.3458   66.6308   30.0174
   65.1233         0   47.6920   13.1254   48.5239
  112.3458   47.6920         0   46.8815   95.3795
   66.6308   13.1254   46.8815         0   55.6505
   30.0174   48.5239   95.3795   55.6505         0

>> Z = SpatialTools_SimulationChi2(10,0,'Mat',[1,0],10,D)

Z =

    0.5349    0.5342    0.4660    0.0083    0.9865
    0.5044    0.8033    0.7294    0.2141    0.0508
    0.9586    0.6348    0.0232    0.6056    0.7587
    0.4099    0.6741    0.3261    0.9735    0.4756
    0.6440    0.3518    0.0029    0.8245    0.3206
    0.4555    0.0647    0.5494    0.1734    0.6354
    0.5595    0.2362    0.6756    0.0339    0.9549
    0.1922    0.9737    0.4337    0.1710    0.6841
    0.9996    0.1399    0.0154    0.8080    0.8494
    0.3528    0.2367    0.6426    0.2657    0.0959

fx >>
```

FIGURE 3.4 – Illustration du programme *SpatialTools_ SimulationChi2()*.

3.2.2 SpatialTools_SimulationPairwiseCopula

Le programme génère un champ aléatoire à partir d'une structure de dépendance sélectionnée (les marginaux sont uniformes). La fonction *SpatialTools_ SimulationPairwiseCopula* nécessite six (7) entrées :

- **n** : Nombre de distributions identiquement indépendantes (i.i.d) répliquées.
- **C** : Une famille de copule bivariée.
- **BetaC** : Valeur du paramètre auxiliaire.
- **Link** : Either Matérn ('Mat'), Rational quadratic ('RQ') ou Exponential ('Exp').
- **LinkParam** :
  - 1ıère entrée : **nu**, paramètre de lissage spécifique à la fonction (valeur dans $(0, Inf)$).

  - 2ième entrée : **nugget**, effet de pépite(valeur dans $[0, 1)$).
- **ER** : Valeur de la portée effective.
- **D** : Matrice symétrique $d \times d$ des distances entre les d stations.

```
function Z = SpatialTools_SimulationPairwiseCopula(n,C,BetaC,Link,
    LinkParam,ER,D)

% Necessary procedures
%     BivCopula_InversionKendall
%     SpatialTools_Link
%     PairwiseCopula_Simulation

TauInv = BivCopula_InversionKendall(C,.2,BetaC);
theta = ER / TauInv;

d = length(D);
nu = LinkParam(1);
nugget = LinkParam(2);

SigmaC = ones(d,d);
for j=1:d-1
    for k=j+1:d
        x = D(j,k) / theta;
        SigmaC(j,k) = SpatialTools_Link(Link,nu,nugget,x);
        SigmaC(k,j) = SigmaC(j,k);
    end
end
Z = PairwiseCopula_Simulation(n,C,SigmaC,BetaC);
```

Par exemple, pour une matrice symétrique D de taille 5×5 donnée, on obtient le champ aléatoire simulé Z de taille $n(= 10) \times 5$ suivant :

```
>> Z = SpatialTools_SimulationPairwiseCopula(10,2,0,'Mat',[1,0],10,D)

Z =

    0.7477    0.4479    0.7605    0.1690    0.1922
    0.2943    0.0087    0.8370    0.4326    0.0560
    0.0117    0.1037    0.2312    0.4711    0.7936
    0.6296    0.4134    0.9551    0.4693    0.8940
    0.2753    0.4094    0.1097    0.4448    0.4601
    0.1146    0.8657    0.2507    0.3316    0.6075
    0.2056    0.3240    0.1510    0.3579    0.2067
    0.6381    0.4930    0.1209    0.2623    0.4695
    0.9789    0.3751    0.9836    0.4517    0.6192
    0.6034    0.5732    0.6143    0.4610    0.2993

fx >>
```

FIGURE 3.5 – Illustration du programme *SpatialTools_SimulationPairwiseCopula()*.

3.3 Visualisation 3d d'un champ aléatoire

Dans cette section, il est question de présenter un programme de visualisation 3d d'un champ aléatoire et de faire quelques illustrations graphiques.

3.3.1 SpatialTools_PlotRandomField3d

Le programme propose une représentation graphique d'un champ aléatoire Z observé à d endroits. La fonction *SpatialTools_PlotRandomField3d()* nécessite deux entrées :

- **Z** : Valeurs observées du champ aléatoire à d emplacements.
- **Coord** : (x, y) - coordonnées où le champ aléatoire a été observé.

```
1 function SpatialTools_PlotRandomField3d(Z,Coord,Scale)
2
3 m = floor(min(Coord));
4 M = floor(max(Coord))+1;
5 [XI,YI] = meshgrid(m(1):1:M(1),m(2):1:M(2));
6
```

```
if strcmp(Scale,'Uniform')
    Z = Z;
elseif strcmp(Scale,'Normal')
    Z = norminv(Z);
end

ZI = griddata(Coord(:,1),Coord(:,2),Z.',XI,YI);
mesh(XI,YI,ZI); hold on;
plot3(Coord(:,1),Coord(:,2),Z.','.','color','black');
colorbar;
hold off;
```

3.3.2 Illustrations graphiques

Illustration 1 :

Par le moyen du programme *SpatialTools_SimulationChi2()*, il est généré un champ alétoire Z (figure 3.6) observé dans $d = 10$ stations qui va être soumis à une visualisation graphique (figure 3.7).

```
Z =

    0.1695    0.0993    0.6764    0.6287    0.2038    0.3919    0.1191    0.7969    0.5823    0.9674
    0.7459    0.7525    0.4527    0.0072    0.7781    0.6062    0.4542    0.2769    0.8882    0.9142
    0.5090    0.4613    0.2817    0.6506    0.0513    0.3009    0.1079    0.4961    0.7802    0.5623
    0.1696    0.5859    0.2424    0.8323    0.5569    0.4585    0.2530    0.4603    0.3200    0.2263
    0.5240    0.6252    0.7938    0.0989    0.9705    0.4253    0.4246    0.6260    0.4899    0.3366
    0.7110    0.6881    0.3967    0.1944    0.5912    0.2362    0.0331    0.0036    0.6436    0.3194
    0.6619    0.5472    0.7791    0.7818    0.7319    0.3394    0.2173    0.3738    0.0416    0.5050
    0.4509    0.7319    0.5560    0.7800    0.3187    0.4110    0.5525    0.4112    0.8681    0.8834
    0.9605    0.4929    0.9292    0.9395    0.8620    0.9827    0.7862    0.1521    0.3106    0.6713
    0.9583    0.6123    0.7385    0.3045    0.5949    0.4098    0.3757    0.3315    0.1859    0.2072

>> Coord

Coord =

   90.7332   47.0582
   63.9939   81.3893
   54.8817   92.3933
   40.8860   15.7276
   53.4126   21.9041
   39.0815    9.0333
   78.5724    7.5093
   22.6983   16.5207
    9.7923   39.1908
   20.4521   89.2702

>> SpatialTools_PlotRandomField3d(Z,Coord,'Normal')
```

FIGURE 3.6 – Illustration du programme *SpatialTools_PlotRandomField3d* avec $d = 10$.

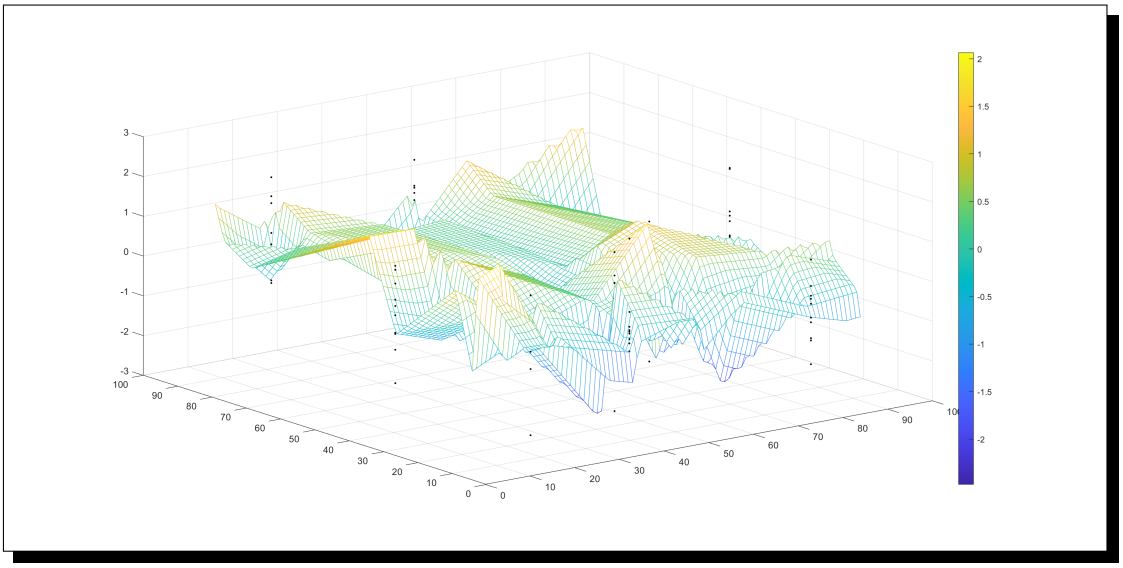


FIGURE 3.7 – Représentation graphique de Z obtenu à partir de *SpatialToolsSimulationChi2()*

Illustration 2 :

Ici, le programme *SpatialTools_ SimulationPairwiseCopula()* est utilisé pour générer le champ aléatoire Z observé dans $d = 5$ stations.

```
Coord =

   16.9114    2.3700
   50.2100   63.9424
   61.9802   99.4346
    0.5121   48.5245
   36.4691   77.4585

>> Z = SpatialTools_SimulationPairwiseCopula(5,2,0,'Mat',[1,0],20,D)

Z =

    0.5851    0.7345    0.9873    0.5510    0.8004
    0.9250    0.8791    0.1095    0.6840    0.9407
    0.0515    0.1414    0.5224    0.4167    0.1272
    0.2453    0.7324    0.3953    0.4425    0.7846
    0.5784    0.9967    0.3903    0.5133    0.9721

>> SpatialTools_PlotRandomField3d(Z,Coord,'Uniform')
>>
```

FIGURE 3.8 – Illustration du programme *SpatialTools_ PlotRandomField3d* avec $d = 5$.

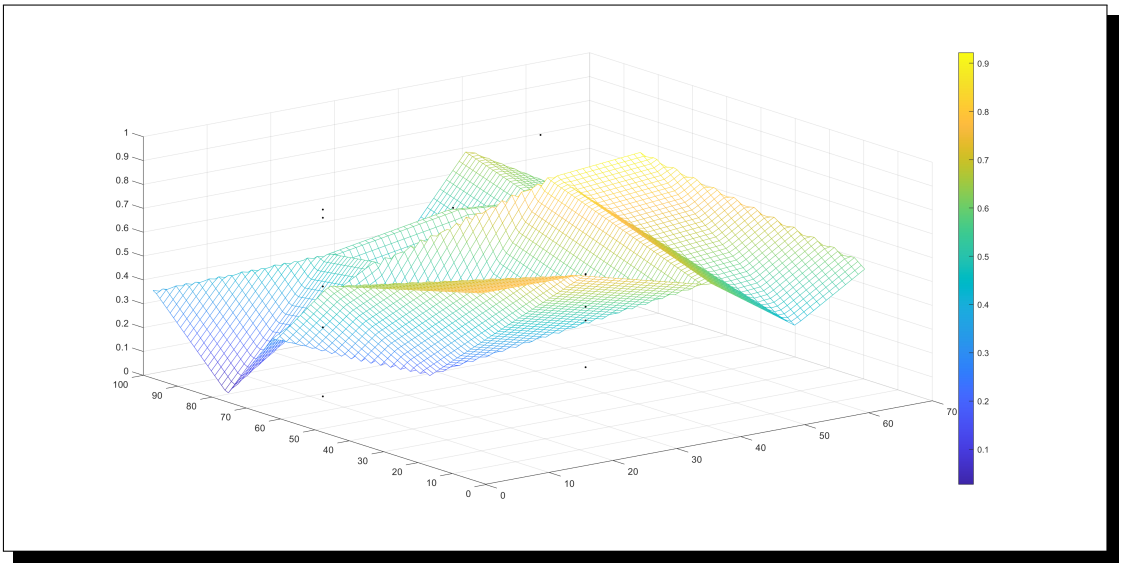


FIGURE 3.9 – Représentation graphique de Z obtenu à partir de *SpatialToolsSimulationPairwiseCopula()*

3.4 Estimation des paramètres d'une copule spatiale

Il convient de remarquer que plusieurs paramètres d'une copule sont inconnus. Ainsi, il est primordial de les estimer afin d'utiliser la copule dans une situation concrète, notamment pour la modélisation spatiale. En raison de la nuée de variables devant être optimisées, une approche de profil de vraisemblance est utilisée, c'est-à-dire que les paramètres sont optimisés tour à tour. Cette section vient présenter deux programmes pour l'estimation des paramètres. Le premier estime par vraisemblance les paires de (θ,a) pour la copule Khi-deux et le second généralise pour les copules Khi-deux, normale et Student.

3.4.1 Estimation par vraisemblance des paires de (θ,a) pour la copule Khi-deux

L'estimation par vraisemblance des paires pour la copule de Khi-deux présentée dans l'équation 2.10 est réalisée grâce au programme *SpatialChi2_ FullMLE* s'appuyant sur *Optimization Toolbox* de Matlab.

```
function MLE = SpatialChi2_FullMLE(Z,D,link,nu)

% Estimation of:
% - the range parameter theta = x(1)
% - the nugget effect epsilon = x(2)
% - the non-centrality parameter a = x(3)

% Uniformiser les donnees basees sur les rangs
n = size(Z,1);
U = VectorOfRanks(Z.') / (n+1);

x0 = [250 .05 .5];
options = optimset('MaxFunEvals',100);
MLE = fminsearch(@(x)(-SpatialChi2_Likelihood(U,D,x(3),link,nu,x(2),x
    (1))),x0,options);
```

```
function L = SpatialChi2_Likelihood(U,D,a,link,nu,nugget,theta)

Neighborhood = 100;

[n,d] = size(U);

if( theta<=0 || nugget<0 )
    L = -10000000000;
else
    L = 0;
    for j=1:d-1
        Delta = prctile(D(:,j),Neighborhood);
        for k=j+1:d
```

```
            if (D(j,k)<=Delta)
                x = D(j,k)/theta;
                rho = SpatialTools_Link(link,nu,nugget,x);
                Uj = U(:,j); Uk = U(:,k);
                parfor i=1:n
                    L = L + log(SpatialTools_Chi2Features(a,rho,Uj(i)
   ,Uk(i),[1 1]));
                end
            end
        end
    end
end
```

```
function C = SpatialTools_Chi2Features(a,theta,u,v,diff)


% Description
%    Computes C, the partial derivatives of C, and the density of C
%    for the non-central chi-square copula
% Input
%    a: Value of the non-centrality parameter
%    theta: Correlation parameter
%    (u,v): Arguments of the function
%    diff: Either [0 0] for C, [1 0] for the 1st partial derivative,
%          [0 1] for the 2nd partial derivative or [1 1] for the
   copula density
%
% Necessary procedures
%    BivCopula_Copula
%    BivCopula_Density

if diff(1)==0 && diff(2)==0
    C = BivCopula_Copula(2,theta,a,u,v);

elseif diff(1)==1 && diff(2)==1
    C = BivCopula_Density(2,theta,a,u,v);
```

```
23
24 elseif diff(1)==1 && diff(2)==0
25     if (a>=10)
26         x = norminv(u); y = norminv(v);
27         C = normcdf((y - theta*x)/sqrt(1-theta^2));
28     else
29         s = fzero(@(x)(normcdf(sqrt(x) + a) + normcdf(sqrt(x) - a) -
    1 - u),[0,5000]);
30         t = fzero(@(x)(normcdf(sqrt(x) + a) + normcdf(sqrt(x) - a) -
    1 - v),[0,5000]);
31         x1 = sqrt(s) - a; x2 = -sqrt(s) - a; y1 = sqrt(t) - a; y2 = -
    sqrt(t) - a;
32         u1 = normcdf(x1); u2 = normcdf(x2);  v1 = normcdf(y1); v2 =
    normcdf(y2);
33         C1 = SpatioTempChi2_Features(10,theta,u1,v1,diff)*normpdf(x1)
    ;
34         C2 = SpatioTempChi2_Features(10,theta,u1,v2,diff)*normpdf(x1)
    ;
35         C3 = SpatioTempChi2_Features(10,theta,u2,v1,diff)*normpdf(x2)
    ;
36         C4 = SpatioTempChi2_Features(10,theta,u2,v2,diff)*normpdf(x2)
    ;
37         C  = (C1-C2+C3-C4) / (normpdf(x1) + normpdf(x2));
38     end
39
40 elseif diff(1)==0 && diff(2)==1
41     C = SpatialTools_Chi2Features(a,theta,v,u,[1 0]);
42
43 end
```

3.4.2 Estimation des paramètres pour les copules de Khi-Deux, Normale et Student

L'estimation des paramètres est effectuée à l'aide de *CopulaEstimation()* qui nécessite également *Optimization Toolbox* de Matlab. Cette fonction effectue une estimation par maximum de vraisemblance de profil en maximisant l'équation 2.9 dans le cas de la copule normale et de la copule de Student. Le programme se trouve dans l'Annexe A (A.1).

3.5 Interpolation spatiale

Après avoir modélisé les données spatiales, nous nous intéressons à la prédiction des valeurs du champ aléatoire à des emplacements inconnus. Nous allons appliquer, dans la présente étude, la méthode d'interpolation spatiale appelée prédiction du *plug-in* qui est présentée par Kazianka et Pilz [12]. De manière laconique, la prédiction du *plug-in* consiste à considérer les estimations du maximum de vraisemblance des paramètres du modèle comme les valeurs réelles et calcule la distribution prédictive du *plug-in* à chaque endroit inconnu. La librairie fournit deux fonctions différentes (se trouvant dans l'Annexe A) pour l'interpolation spatiale à des emplacements non observés. La première, *CopulaPredict()* (voir (A.2)), effectue une interpolation spatiale basée sur les copules à des endroits donnés, en calculant la moyenne, la variance et les quantiles prédictifs du *plug-in* tandis que *CopulaPredictDensity()* (voir (A.3)) consiste à calculer la densité prédictive à des endroits donnés.

3.6 Validations croisées quantitative et qualitative

Dans cette section, nous allons présenter les deux validations croisées quantitative et qualitative utilisées dans la présente étude pour le choix du modèle. Elles ont été mises en évidence par Kazianka et Pilz [12].

3.6.1 Validation quantitative

La validation croisée *leave-one-out* est utilisée comme méthode quantitative pour évaluer la performance des modèles. Elle est obtenue par :

$$MSPE = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{Z}_{-i}(x_i) - z(x_i)\right)^2,$$

où $\hat{Z}_{-i}(x_i)$ est la prédiction à l'emplacement x_i qui est calculée en utilisant toutes les valeurs observées sauf $z(x_i)$. Cette validation peut être réalisée à l'aide de deux fonctions *copulaCV()* (A.4) et *copulaCVDensity()* (A.5) qui se trouvent également dans l'Annexe A. La première calcule les statistiques associées telles que, entre autres, les prédictions de la validation croisée, l'erreur quadratique moyenne, la corrélation entre les vraies valeurs et celles prédites. Quant à la seconde fonction, elle effectue une validation croisée (*leave one out*) des densités prédictives.

3.6.2 Validation qualitative

La validation croisée par les intervalles de confiance sert d'outil de comparaison qualitative des performances prédictives. La fonction *computeCoverage()* (voir (A.6)) examine la largeur moyenne et la couverture des intervalles de prédiction par validation croisée centrée sur la médiane qui sont délimités par les quantiles $\frac{\alpha_L(1-p)}{2}$ et $\frac{\alpha_U(1+p)}{2}$.

Si $F_{Z(x_i)\setminus z(-i)}, \Theta$ désigne la fonction de distribution prédictive (plug-in) basée sur la suppression, la fraction des valeurs réelles tombant dans l'intervalle de prédiction correspondant à $p \in [0,1]$ est donnée par :

$$\bar{\xi}(p) = \frac{1}{n} \sum_{i=1}^{n} \xi_i(p),$$

$$\xi_i(p) = \mathbb{1}\left(F^{-1}_{Z(x_i)\setminus z(-i)}, \Theta(\alpha_L) \leq z(x_i) \leq F^{-1}_{Z(x_i)\setminus z(-i)}, \Theta(\alpha_U) \right),$$

où $\mathbb{1}$ désigne la fonction indicatrice. La largeur moyenne de l'intervalle de prédiction peut être calculée comme suit

$$\bar{W}(p) = \frac{\sum_{i=1}^{n} \xi_i(p) \left(F^{-1}_{Z(x_i)\setminus z(-i)}, \Theta(\alpha_U) - F^{-1}_{Z(x_i)\setminus z(-i)}, \Theta(\alpha_L) \right)}{n\bar{\xi}(p)}.$$

Chapitre 4

Illustration sur le jeu de données « Meuse »

Dans ce chapitre, il sera question d'illustrer, sur de vraies données la fonctionnalité des programmes introduits dans le chapitre 3.

4.1 Présentation des données et analyses préliminaires

Dans cette section, il convient de décrire, de manière succincte, le jeu de données sur lequel est portée notre analyse.

Le choix sur le jeu de données « Meuse » réside dans le fait qu'il est assez connu en analyse spatiale. De surcroit, il est gratuitement disponible dans la bibliothèque *sp* du logiciel *R* produit par Pebesma et Bivand [18]. Ainsi, il suffit de charger la bibliothèque pour pouvoir récupérer les données associées.

La Meuse est un fleuve européen qui prend sa source en France et se jette dans la mer du Nord à travers la France, la Belgique et les Pays-Bas sur une longueur d'environ 950 kilomètres. Le jeu de données « Meuse » contient des concentrations de métaux lourds (zinc, cuivre, plomb et cadmium) dans le sol supérieur de la plaine inondable le long de la Meuse aux Pays-Bas, près du village de Stein.

Une des hypothèses est que ces sédiments contaminés par des métaux lourds sont transportés par la rivière et se déposent principalement près des côtes et à des altitudes plus basses. C'est ce que nous allons essayer de modéliser dans ce chapitre, en s'appesantissant uniquement sur la concentration en cadmium.

Plus précisément, le jeu de données considéré comprend les valeurs $z(x_i)$ de concentrations de cadmium, observées dans $n = 155$ locations $x_i = (x_{1i}, x_{2i})^T$, $i = 1, ..., 155$. L'emplacement de ces dernières est identifié sur la figure 4.1, obtenue grâce au programme *SpatialTools_ PlotRandomField3d()*, tandis que la figure 4.2 combine l'emplacement des valeurs observées et la grille d'interpolation sur laquelle la prédiction va être faite.

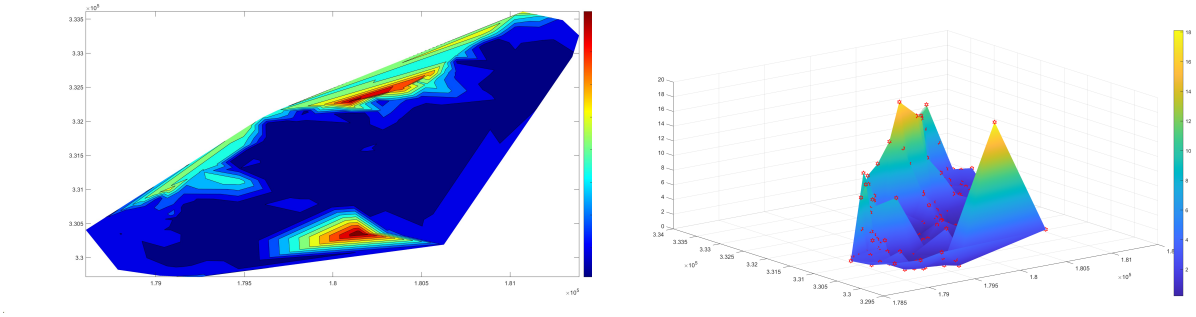


FIGURE 4.1 – Emplacement des $n = 155$ points d'observations (points rouges sur la figure de droite) autour de la Meuse.

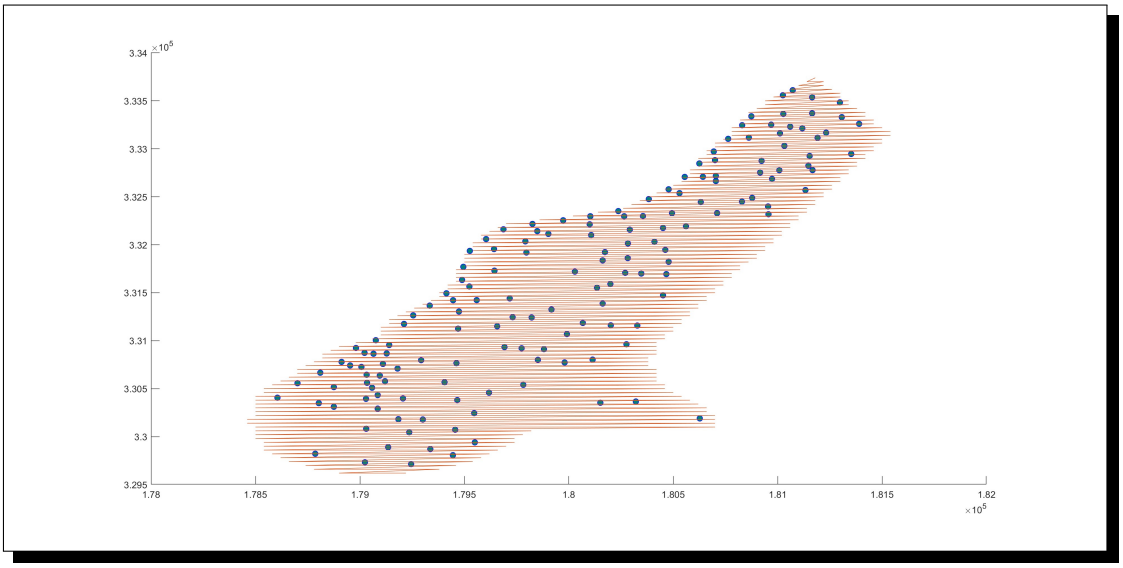


FIGURE 4.2 – Concentration de cadmium : valeurs observées (points bleus) et la grille d'interpolation (traits rouges).

Une analyse rapide des statistiques sommaires de la table 4.1 révèle la présence d'asymétrie dans les données, corroborée par la valeur (1,779) significativement plus grande que zéro du coefficient d'asymétrie (skewness). Un de nos objectifs étant de travailler sur des données non gaussiennes, la distribution des données de concentration de cadmium cadre bien avec la présente étude.

TABLE 4.1 – Statistiques sommaires des valeurs observées de Z.

n	Min.	Moy.	Médiane	Mode	Max.	Écart type	Skewness
155	0,200	3,246	2,100	0,200	18,100	3,524	1,779

4.2 Modélisation des marges

L'avantage majeur de l'utilisation des copules pour la modélisation multidimensionnelle demeure la possibilité de choisir des marges adéquates, indépendamment de la structure de dépendance. Ainsi, par ricochet, l'étape première de la modélisation consiste à ajuster des modèles pour les lois marginales.

De ce fait, la procédure *Initialize()* propose d'appliquer des marginaux gaussiens par la transformée de Box-Cox avec un paramètre de transformation $\lambda = 0,1314$. Plus formellement, ladite transformée est une transformation non linéaire, définie, pour toute $x \geqslant 0$, de la manière suivante :

$$B(x,\lambda) = \begin{cases} \frac{x^\lambda - 1}{\lambda} & \text{si } \lambda \neq 0 \\ \log(x) & \text{si } \lambda = 0 \end{cases}.$$

Ainsi, si $\lambda > 1$, x voit ses grandes valeurs amplifiées, sinon ses grandes valeurs seront réduites. La puissance de *Initialize()* sur le choix des lois marginales a été vérifiée en considérant différents modèles de copules spatiales avec quelques lois marginales (log-normale, gaussienne, Box-Cox etc.) et ceux dont la transformation de Box-Cox est choisie, ont été avérés les meilleurs modèles.

4.3 Estimation des paramètres

Sept modèles spatiaux ont été considérés pour modéliser la dépendance spatiale, à savoir les copules Normale et de Student, couplées aux fonctions de lien Matérn lorsque $\nu = 0,5$ et exponentielle, et la copule de Khi-deux couplée avec la fonction de lien Matérn. La compilation des résultats d'estimation de la méthode que nous proposons ici prend grossièrement du temps dans le cas de la copule Khi-deux. C'est pour cette raison que nous avons limité arbitrairement la modélisation à l'aide de cette copule. Ici, le paramètre de l'effet de pépite ϵ et le paramètre de la portée θ seront estimés à partir des données à l'aide de l'estimateur de pseudo-vraisemblance par paire ; le paramètre de non-centralité a de la copule de Khi-deux sera également estimé.

Les résultats de l'estimation des paramètres pour chacun des modèles considérés sont présentés dans le tableau 4.2.

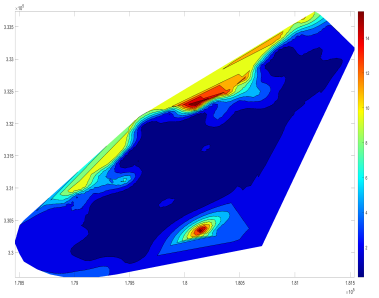
TABLE 4.2 – Estimation des paramètres pour différents modèles de copules spatiales.

Copule	Loi marginale	Fonction de lien	$\hat{\theta}$	$\hat{\epsilon}$	$\hat{a}$	MSPE
Normale	Logn	Exponentielle	655,398	0,01	-	5,923
	Logn	Matérn$^{1/2}$	696,197	0,001	-	6,0661
	Boxcox	Matérn$^{1/2}$	864,392	0,001	-	5,510
Student	Logn	Exponentielle	1036,532	0,01	-	5,534
	Logn	Matérn$^{1/2}$	957,914	0,001	-	5,519
	Boxcox	Matérn$^{1/2}$	883,475	0,001	-	5,511
Khi-deux	Boxcox	Matérn$^{1/2}$	975,230	0,00012	2,110	5,469

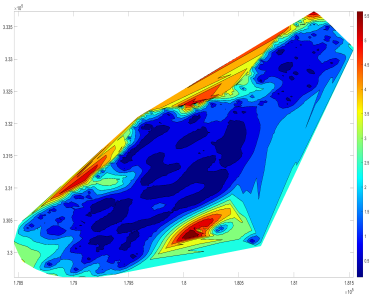
4.4 Interpolation spatiale et validations croisées

Après avoir estimé les paramètres du modèle, nous nous sommes appesanti sur la validation du modèle prédictif et par la prédiction des valeurs du champ aléatoire sous-jacent aux 1303 endroits de la grille d'interpolation illustrés par des traits rouges sur la figure 4.2. Cette grille est également fournie dans le package *sp* du logiciel R.

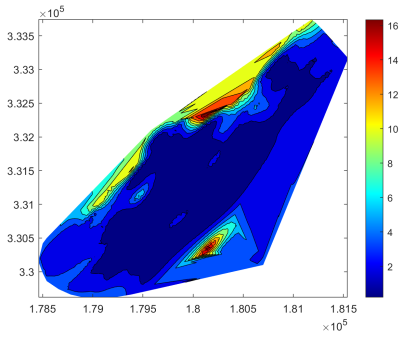
La figure 4.3 met en évidence les prédictions du *plug-in* des moyennes et des écarts types pour les modèles de copules spatiales Khi-deux et Normale. L'analyse des cartes des moyennes révèle que les deux modèles conduisent à des résultats légèrement différents dans la partie nord et sud du fleuve Meuse. Toutefois, le modèle de la copule Normale débouche sur des écarts types plus élevés dans le sud (légèrement) et dans l'est du fleuve, tandis que la copule de Khi-Deux supplante au nord. Cela peut être expliqué par le plus grand nombre d'observations de cadmium dans le nord, tel que corroboré par la figure 4.1.



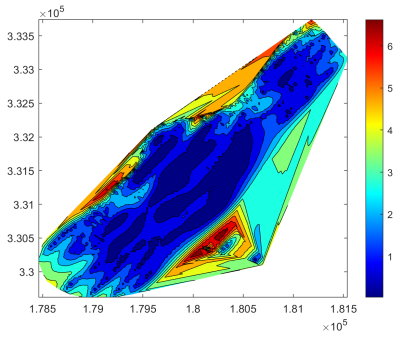
(a) Copule Khi-deux : moyenne prédite



(b) Copule Khi-deux : variance prédite



(c) Copule Normale : moyenne prédite



(d) Copule Normale : variance prédite

FIGURE 4.3 – Concentration de cadmium : moyenne et variance prédites pour les couples Khi-deux et Normale utilisant les marginaux de Box-cox calculées sur une grille régulière.

En l'absence de données de test, nous utilisons la validation croisée *leave-one-out* présentée dans le chapitre 3 comme méthode quantitative et celle des intervalles de confiance, pour évaluer la performance du modèle. La dernière colonne du tableau 4.2 résume les résultats de la validation croisée. Ainsi, il en découle que le modèle de Khi-deux conduit à la MSPE (5,469) la plus faible, et par ricochet, peut être considéré comme le mieux adapté.

L'analyse de la figure 4.4 renseigne que le modèle basé sur la copule de Khi-deux (courbe bleue) conduit aux intervalles de confiance les plus étroits. En effet, le pourcentage de couverture du modèle est toujours très proche du niveau nominal (ligne rouge pointillée). Par ailleurs, le modèle basé sur la copule Normale (courbe verte) débouche sur des intervalles de confiance relativement grands et surestime donc l'incertitude de l'interpolation.

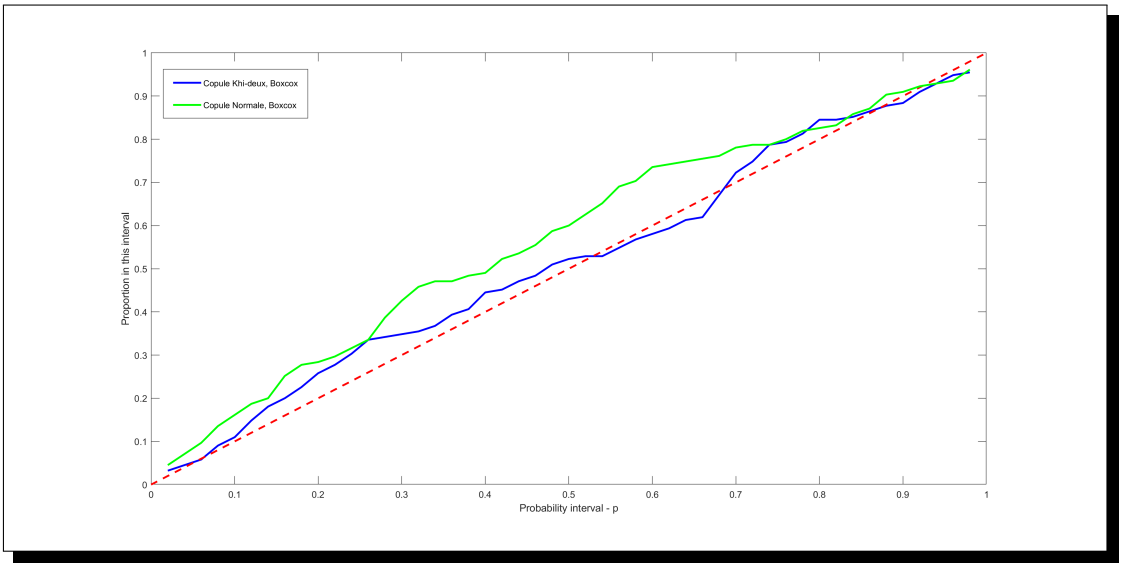


FIGURE 4.4 – Couverture des intervalles de confiance par validation croisée

En définitive, il émane de cette étude que les modèles de copules spatiales Normale et Khi-deux s'adaptent bien aux données. Toutefois, celui de la copule de Khi-deux est nettement plus performant, en raison notamment du comportement radialement asymétrique des données détecté par l'analyse exploratoire.

Conclusion

En définitive, nous avons vu que la modélisation de la dépendance spatiale à l'aide des copules offre une myriade d'avantages par rapport aux méthodes traditionnelles. D'un côté, elle offre une grande flexibilité dans la construction du modèle et peut assouplir l'hypothèse traditionnelle de champ aléatoire gaussien, qui suppose implicitement une structure de dépendance normale. D'un autre côté, elle fournit une meilleure compréhension du champ aléatoire à l'étude, notamment avec la possibilité de quantifier le niveau de dépendance d'un champ aléatoire via des mesures non paramétriques de dépendance. De surcroit, la modélisation de la dépendance spatiale à l'aide de copules permet de calculer les meilleures prédictions de champs aléatoires à des emplacements non observés. Cette procédure est généralement appelée interpolation spatiale et s'appuie fortement sur les hypothèses de distribution du phénomène à variation spatiale.

L'atout majeur de la méthodologie basée sur les copules repose précisément sur l'analyse des données non gaussiennes. C'est dans cette optique, qu'une librairie pour l'analyse des données spatiales à l'aide des copules a été mise en place. Elle contient diverses fonctions permettant d'effectuer une modélisation et une interpolation spatiales à l'aide de copules. Outre les fonctions principales d'estimation des paramètres et de prédiction spatiale, diverses fonctions d'analyse exploratoire des données et de validation quantitative et qualitative des modèles sont fournies. En plus, il existe une fonction permettant de proposer une distribution marginale appropriée, une fonction de corrélation adéquate et les paramètres de départ correspondants pour l'estimation du maximum de vraisemblance sans interaction avec l'utilisateur.

L'utilité de la librairie a été mise en exergue en analysant les données de Meuse, plus précisément les mesures de concentration du cadmium qui sont modérément asymétriques. La copule de Khi-deux s'est avérée un outil de choix pour la modélisation de la dépendance spatiale.

Les travaux effectués dans le cadre de ce mémoire peuvent être étendus dans plusieurs directions. En effet, le domaine étudié est si vaste et rempli de possibilités qu'il pourrait facilement faire l'objet de recherches subséquentes. Il serait intéressant, alternativement à la méthode de la validation croisée, de développer des outils d'adéquation formels pour le choix d'une copule et d'une fonction de lien dans un contexte spatial. De plus, étudier les copules *squared* pour l'analyse de données répliquées serait un grand intérêt.

Bibliographie

[1] M. Alexander, J., R. Frey, and P. Embrechts. Quantitative risk management : Concepts, techniques and tools. *Princeton University Press, Princeton, NJ, 2005*, 2005.

[2] A. Bàrdossy. Copula-based geostatistical models for groundwater quality parameters. *Water Resources Research*, 42(11) :1–12, 2006. URL `http://www.agu.org/pubs/crossref/2006/2005WR004754.shtml`.

[3] S. Castruccio, R. Huser, and M. G. Genton. High-order composite likelihood inference for max-stable distributions and processes. *Journal of Computational and Graphical Statistics 25, 1212–1229*, 2016.

[4] D. Clayton and J. André. *GSLIB : Geostatistical Software Library and User's Guide (Applied Geostatistics Series)*. Applied Geostatistics Series. Oxford University Press, USA, 2 edition, 1992. ISBN 0195100158,9780195100150.

[5] N. Cressie. *Statistics for spatial data*. John Wiley and Sons, Inc, 1993.

[6] P. Embrechts, F. Lindskog, and A. McNeil. Modelling dependence with copulas and applications to risk management. *In Handbook of Heavy Tailed Distributions in Finance*, 2003.

[7] G. Genton, Marc and P. Krupskii. Factor copula models for data with spatio-temporal dependence. *Spatial Statistics*, 2017.

[8] A. Goldberger. Best linear unbiased prediction in the generalized linear regression model. *Journal of the American Statistical Association*, 57 :69–375, 1963.

[9] C. Gotway and N. Cressie. Improved multivariate prediction under a general linear model. *Journal of Multivariate Analysis*, 45 :56–72, 1993.

[10] L. Jin and H. Andrew. Spatial interpolation methods applied in the environmental sciences : A review. *Environmental Modelling & Software*, 53(4) :173–189, 2011.

[11] Kaufman and Shaby. The role of the range parameter for estimation and prediction in geostatistics. *Biometrika, 100(2) :473-484*, 2013.

[12] Kazianka and Pilz. Bayesian spatial modeling and interpolation using copulas. *Comp Geosci 37 :310–319*, 2011.

[13] P. Krupski, R. Huser, and M. G. Genton. Factor copula models for replicated spatial data. *Journal of the American Statistical Association*, 2017.

[14] G. Matheron. Principles of geostatistics. *Economic geology*, 58(8) :1246–1266, 1963.

[15] B. Matérn. Spatial variation : Stachastic models and their application to some problems in forest surveys and other sampling investigations. *Statens skogsforskningsinstitut*, 1960.

[16] A. K. Nikoloulopoulos, H. Joe, and H. Li. Vine copulas with asymmetric tail dependence and applications to financial return data. *Computational Statistics & Data Analysis 56, 3659–3673*, 2012.

[17] S. Oliver and G. Carol, A. *Statistical methods for spatial data analysis*. Texts in statistical science. Chapman & Hall/CRC, 2005. ISBN 9781584883227,1584883227.

[18] E. Pebesma and R. Bivand. Classes and methods for spatial data in r. *R News*, 5, 01 2005.

[19] J.-F. Quessy, L.-P. Rivest, and M.-H. Toupin. On the family of multivariate chi-square copulas. *J. Multivariate Anal.*, 152 :40–60, 2016.

[20] W. Richard and O. Margaret, A. *Geostatistics for environmental scientists*. Statistics in Practice. Wiley, 2 edition, 2007. ISBN 9780470028582,0470028580.

[21] J. Segers. Max-stable models for multivariate extremes. *Statistical Journal 10, 61–82*, 2012.

[22] A. Sklar. Fonctions de répartition à n dimensions et leurs marges. *Publ. Inst. Statist. Univ. Paris*, 8 :229–231, 1959.

[23] L. Stein. *Interpolation of spatial data*. Some theory for Kriging. Springer Series in Statistics, springer-verlag, new york edition, 1999.

[24] Watson. A treatise on the theory of bessel functions. *Cambridge Mathematical Library*, 1995.

[25] Yaglom. *Correlation Theory of Stationary and Related Random Functions : Volume I : Basic Results*. Springer, 1 edition, 1987. ISBN 9780387962689,0387962689.

[26] Y. Yamagata and H. Seya. *Spatial analysis using big data : methods and urban applications*. Academic Press ; Elsevier, 2020. ISBN 9780128131275, 0128131276. Spatial econometrics and spatial statistics.

Annexe A

Fonctions de la librairie Matlab pour l'analyse des données spatiales avec les copules

A.1 CopulaEstimation

```
function optimized=copulaEstimation(data,model,options)

% Performs maximum (composite) likelihood estimation for the model parameters based on
% a profile likelihood approach.
%
% Inputs:
%
% data.....the data set as a structure array of the format data.x (x-coordinates), data.y
%             (y-coordinates), data.z (measured values)
%
% model.....contains the parameters of the spatial copula model (
```

```
    see
% function initialize)
%
% options.....options structure array (see function testOptions)
%
% Outputs:
%
% optimized...contains the optimized parameters of the spatial copula
    model
%

    options=testOptions(options);

    if isempty(model.copula) || strcmp(model.copula.name,'Gaussian')
      optimized=profilelikelihoodGaussian(model.margin,model.
    correlation,model.trend,model.anisotropy,data,options);
    elseif strcmp(model.copula.name,'Chi2')
        optimized=profilelikelihoodChi2(model.margin,model.
    correlation,model.trend,model.anisotropy,model.copula,data,
    options);
    elseif strcmp(model.copula.name,'Student')
      optimized=profilelikelihoodStudent(model.margin,model.
    correlation,model.trend,model.anisotropy,model.copula,data,
    options);
    end

%%
%GAUSSIAN COPULA

function optimized= profilelikelihoodGaussian(margin,correlation,
    trend,anisotropy,data,options)

    if isempty(anisotropy.params)
        h=pdist([data.x data.y]);
        h=squareform(h);
    elseif anisotropy.const==1
```

```
40      ab=[cos(anisotropy.params(1)),sin(anisotropy.params(1));-
    anisotropy.params(2)*sin(anisotropy.params(1)),anisotropy.params
    (2)*cos(anisotropy.params(1))]*[data.x';data.y'];
41          x=ab(1,:)';
42          y=ab(2,:)';
43          h=pdist([x y]);
44          h=squareform(h);
45      end
46
47      len=length(margin.params);
48      lik=0;
49      oldlik=1;
50
51      if options.print>=1
52          disp('Starting Profile Likelihood Approach');
53      end
54
55      while abs(lik-oldlik)>options.tol
56          oldlik=lik;
57
58          if ~isempty(anisotropy.params) && anisotropy.const==0
59              ab=[cos(anisotropy.params(1)),sin(anisotropy.params(1));-
    anisotropy.params(2)*sin(anisotropy.params(1)),anisotropy.params
    (2)*cos(anisotropy.params(1))]*[data.x';data.y'];
60              x=ab(1,:)';
61              y=ab(2,:)';
62              h=pdist([x y]);
63              h=squareform(h);
64          end
65
66          cov=correlCovMat(h,correlation.params,correlation.model);
67          R = chol(cov,'upper');
68          Rinv=eye(size(cov)) / R;
69          logSqrtDetSigma = sum(log(diag(R)));
70
71          if ~isempty(margin.params) && ~isempty(trend.params) && (
```

```
    margin.const==0 || trend.const==0)
72          [par lik]=fmincon(@optimfun1,[margin.params trend.params
    ],[],[],[],[],[margin.lower trend.lower],[margin.upper trend.
    upper],[],options.optim,margin,trend,correlation,anisotropy,data,
    h,len,Rinv,logSqrtDetSigma,options);
73          margin.params=par(1:len);
74          trend.params=par((len+1):end);
75      end
76
77      if ~isempty(correlation.params) && correlation.const==0
78          [correlation.params lik]=fmincon(@optimfun2,correlation.
    params,[],[],[],[],correlation.lower,correlation.upper,[],options
    .optim,margin,trend,correlation,anisotropy,data,h,options);
79      end
80
81      if ~isempty(anisotropy.params) && anisotropy.const==0
82          [anisotropy.params lik]=fmincon(@optimfun3,anisotropy.
    params,[],[],[],[],anisotropy.lower,anisotropy.upper,[],options.
    optim,margin,trend,correlation,anisotropy,data,h,options);
83      end
84      if options.print>=1
85        disp(['Profile Likelihood increase: ' num2str(abs(lik-
    oldlik))]);
86          disp([num2str(lik) '    ' num2str(round([margin.params
    trend.params correlation.params anisotropy.params]*100)/100)]);
87      end
88  end
89
90  if options.print>=1
91      disp('Final result:');
92      disp([num2str(lik) '    ' num2str(round([margin.params trend.
    params correlation.params anisotropy.params]*100)/100)]);
93  end
94
95  optimized.margin=margin;
96  optimized.correlation=correlation;
```

```
97      optimized.anisotropy=anisotropy;
98      optimized.trend=trend;
99      optimized.options=options;
100
101
102 function val =optimfun1(params,margin,trend,correlation,anisotropy,
    data,h,len,Rinv,logSqrtDetSigma,options)
103         margin.params=params(1:len);
104         trend.params=params((len+1):end);
105         val=loglikelihoodGaussian(margin,trend,correlation,anisotropy
    ,data,h,Rinv,logSqrtDetSigma,0,options);
106
107 function val =optimfun2(params,margin,trend,correlation,anisotropy,
    data,h,options)
108         correlation.params=params;
109         val=loglikelihoodGaussian(margin,trend,correlation,anisotropy
    ,data,h,[],[],0,options);
110
111 function val =optimfun3(params,margin,trend,correlation,anisotropy,
    data,h,options)
112         anisotropy.params=params;
113         val=loglikelihoodGaussian(margin,trend,correlation,anisotropy
    ,data,h,[],[],1,options);
114
115
116 function like=loglikelihoodGaussian(margin,trend,correlation,
    anisotropy,data,h,Rinv,logSqrtDetSigma,anisomode,options)
117
118     if anisomode
119       xy=[cos(anisotropy.params(1)),sin(anisotropy.params(1));-
    anisotropy.params(2)*sin(anisotropy.params(1)),anisotropy.params
    (2)*cos(anisotropy.params(1))]*[data.x';data.y'];
120       x=xy(1,:)';
121       y=xy(2,:)';
122     end
123
```

```
124     expected=trend.F*trend.params';
125     if length(margin.params)==1
126         z=feval([margin.name 'cdf'],data.z,expected,margin.params);
127         zz=feval([margin.name 'pdf'],data.z,expected,margin.params);
128     elseif length(margin.params)==2
129         z=feval([margin.name 'cdf'],data.z,expected,margin.params(1),
    margin.params(2));
130         zz=feval([margin.name 'pdf'],data.z,expected,margin.params(1)
    ,margin.params(2));
131     end
132 
133     ok=sum(z~=1)==length(z);
134     ok=ok*(sum(z~=0)==length(z));
135 
136     if ok==1
137         z=norminv(z,0,1);
138         if isempty(Rinv) || isempty(logSqrtDetSigma)
139             if anisomode
140                 h=pdist([x y]);
141                 h=squareform(h);
142             end
143             cov=correlCovMat(h,correlation.params,correlation.model);
144             like=-mvnlogpdf_noconst(z',cov)-0.5*(z'*z)-sum(log(zz));
145         else
146             xRinv = z'*Rinv;
147             quadform = sum(xRinv.^2, 2);
148             like=0.5*quadform+logSqrtDetSigma-0.5*(z'*z)-sum(log(zz))
    ;
149         end
150 
151         if isnan(like)
152             like=10^100;%a large value. This is to continue the
    optimization procedure if inappropriate values for the parameters
     are chosen by the fmincon-algorithm
153         end
154     else
```

```
155         like=10^100;
156     end
157     if options.print>=2
158         disp([num2str(like) ' ' num2str(round([margin.params trend.
    params correlation.params anisotropy.params]*1000)/1000)])
159     end
160
161 %%
162 %CHI^2-COPULA
163
164
165 function optimized= profilelikelihoodChi2(margin,correlation,trend,
    anisotropy,copula,data,options)
166
167     if isempty(anisotropy.params)
168         h=pdist([data.x data.y]);
169         h=squareform(h);
170     elseif anisotropy.const==1
171       ab=[cos(anisotropy.params(1)),sin(anisotropy.params(1));-
    anisotropy.params(2)*sin(anisotropy.params(1)),anisotropy.params
    (2)*cos(anisotropy.params(1))]*[data.x';data.y'];
172         x=ab(1,:)';
173         y=ab(2,:)';
174         h=pdist([x y]);
175         h=squareform(h);
176     end
177
178
179     len=length(margin.params);
180     lik=0;
181     oldlik=1;
182
183
184     while abs(lik-oldlik)>options.tol
185       oldlik=lik;
186
```

```
187         if ~isempty(anisotropy.params) && anisotropy.const==0
188             ab=[cos(anisotropy.params(1)),sin(anisotropy.params(1));-
    anisotropy.params(2)*sin(anisotropy.params(1)),anisotropy.params
    (2)*cos(anisotropy.params(1))]*[data.x';data.y'];
189             x=ab(1,:)';
190             y=ab(2,:)';
191             h=pdist([x y]);
192             h=squareform(h);
193         end
194 
195         if ~isempty(margin.params) && ~isempty(trend.params) && (
    margin.const==0 || trend.const==0)
196             [par lik]=fmincon(@optimfun4,[margin.params trend.params
    ],[],[],[],[],[margin.lower trend.lower],[margin.upper trend.
    upper],[],options.optim,margin,trend,correlation,anisotropy,
    copula,data,h,len,options);
197             margin.params=par(1:len);
198             trend.params=par((len+1):end);
199         end
200 
201         if ~isempty(copula.params) && copula.const==0
202             [copula.params lik]=fmincon(@optimfun7,copula.params
    ,[],[],[],[],copula.lower,copula.upper,[],options.optim,margin,
    trend,correlation,anisotropy,copula,data,h,options);
203         end
204 
205         if ~isempty(correlation.params) && correlation.const==0
206            [correlation.params lik]=fmincon(@optimfun5,correlation.
    params,[],[],[],[],correlation.lower,correlation.upper,[],options
    .optim,margin,trend,correlation,anisotropy,copula,data,h,options)
    ;
207         end
208 
209         if ~isempty(anisotropy.params) && anisotropy.const==0
210            [anisotropy.params lik]=fmincon(@optimfun6,anisotropy.
    params,[],[],[],[],anisotropy.lower,anisotropy.upper,[],options.
```

```
    optim,margin,trend,correlation,anisotropy,copula,data,h,options);
211         end
212
213         if options.print>=1
214           disp(['Profile Likelihood increase: ' num2str(abs(lik-
    oldlik))]);
215             disp([num2str(lik) '    ' num2str(round([margin.params
    trend.params correlation.params anisotropy.params copula.params
    ]*1000)/1000)])
216         end
217     end
218
219     if options.print>=1
220         disp('Final result:');
221         disp([num2str(lik) '    ' num2str(round([margin.params trend.
    params correlation.params anisotropy.params copula.params]*100)
    /100)])
222     end
223
224     optimized.margin=margin;
225     optimized.correlation=correlation;
226     optimized.anisotropy=anisotropy;
227     optimized.trend=trend;
228     optimized.copula=copula;
229     optimized.options=options;
230
231
232 function val =optimfun4(params,margin,trend,correlation,anisotropy,
    copula,data,h,len,options)
233   margin.params=params(1:len);
234   trend.params=params((len+1):end);
235   val=loglikelihoodChi2(margin,trend,correlation,anisotropy,copula,
    data,h,0,options);
236
237 function val =optimfun5(params,margin,trend,correlation,anisotropy,
    copula,data,h,options)
```

```
238     correlation.params=params;
239     val=loglikelihoodChi2(margin,trend,correlation,anisotropy,copula,
    data,h,0,options);
240
241 function val =optimfun6(params,margin,trend,correlation,anisotropy,
    copula,data,h,options)
242     anisotropy.params=params;
243     val=loglikelihoodChi2(margin,trend,correlation,anisotropy,copula,
    data,h,1,options);
244
245 function val =optimfun7(params,margin,trend,correlation,anisotropy,
    copula,data,h,options)
246     copula.params=params;
247     val=loglikelihoodChi2(margin,trend,correlation,anisotropy,copula,
    data,h,0,options);
248
249
250
251 function like = loglikelihoodChi2(margin,trend,correlation,anisotropy
    ,copula,data,h,anisomode,options)
252
253     if anisomode
254             xy=[cos(anisotropy.params(1)),sin(anisotropy.params(1));-
    anisotropy.params(2)*sin(anisotropy.params(1)),anisotropy.params
    (2)*cos(anisotropy.params(1))]*[data.x';data.y'];
255             x=xy(1,:)';
256             y=xy(2,:)';
257     end
258
259     expected=trend.F*trend.params';
260     if length(margin.params)==1
261         z=feval([margin.name 'cdf'],data.z,expected,margin.params);
262         zz=feval([margin.name 'pdf'],data.z,expected,margin.params);
263     elseif length(margin.params)==2
264         z=feval([margin.name 'cdf'],data.z,expected,margin.params(1),
    margin.params(2));
```

```
265         zz=feval([margin.name 'pdf'],data.z,expected,margin.params(1)
    ,margin.params(2));
266     end
267 
268     n=length(data.z);
269     ok=sum(z~=1)==n;
270     ok=ok*(sum(z~=0)==n);
271     lambda=copula.params;
272 
273     if ok==1
274         if anisomode
275             h=pdist([x y]);
276             h=squareform(h);
277         end
278 
279         if lambda<0
280             like=10^100;
281         else
282             z=ncx2inv(z,1,lambda);
283             x=log(ncx2pdf(z,1,lambda));
284             z=sqrt(z);
285             logz=log(z);
286             lambda=sqrt(lambda);
287             %m=[lambda,lambda];
288 
289             zz=log(zz);
290             like=0;
291             cov=correlCovMat(h,correlation.params,correlation.model);
292             for i=1:n
293                 for j=(i+1):n
294                     like=like-(log(sum(exp(mvnlogpdf_noconst([z(i),z(
    j);-z(i),z(j);z(i),-z(j);-z(i),-z(j)]-lambda,cov([i,j],[i,j])))))
    /(2*pi))-logz(i)-logz(j)-x(i)-x(j))-zz(i)-zz(j);
295                 end
296             end
297             if isnan(like) || isinf(like)
```

```
298                 like=10^100;
299             end
300         end
301     else
302         like=10^100;
303     end
304
305     if options.print>=2
306         disp([num2str(like) ' ' num2str(round([margin.params trend.
    params correlation.params anisotropy.params copula.params]*1000)
    /1000)])
307     end
308
309 %%
310 %Student-t copula
311 function optimized= profilelikelihoodStudent(margin,correlation,trend
    ,anisotropy,copula,data,options)
312
313     if isempty(anisotropy.params)
314         h=pdist([data.x data.y]);
315         h=squareform(h);
316     elseif anisotropy.const==1
317       ab=[cos(anisotropy.params(1)),sin(anisotropy.params(1));-
    anisotropy.params(2)*sin(anisotropy.params(1)),anisotropy.params
    (2)*cos(anisotropy.params(1))]*[data.x';data.y'];
318         x=ab(1,:)';
319         y=ab(2,:)';
320         h=pdist([x y]);
321         h=squareform(h);
322     end
323
324     len=length(margin.params);
325     lik=0;
326     oldlik=1;
327
328     while abs(lik-oldlik)>options.tol
```

```
329         oldlik=lik;
330 
331         if ~isempty(anisotropy.params) && anisotropy.const==0
332             ab=[cos(anisotropy.params(1)),sin(anisotropy.params(1));-
    anisotropy.params(2)*sin(anisotropy.params(1)),anisotropy.params
    (2)*cos(anisotropy.params(1))]*[data.x';data.y'];
333             x=ab(1,:)';
334             y=ab(2,:)';
335             h=pdist([x y]);
336             h=squareform(h);
337         end
338 
339         cov=correlCovMat(h,correlation.params,correlation.model);
340         R = chol(cov,'upper');
341         Rinv=eye(size(cov)) / R;
342         logSqrtDetSigma = sum(log(diag(R)));
343 
344         if ~isempty(margin.params) && ~isempty(trend.params) && (
    margin.const==0 || trend.const==0)
345             [par lik]=fmincon(@optimfun8,[margin.params trend.params
    ],[],[],[],[],[margin.lower trend.lower],[margin.upper trend.
    upper],[],options.optim,margin,trend,correlation,anisotropy,
    copula,data,h,len,Rinv,logSqrtDetSigma,options);
346             margin.params=par(1:len);
347             trend.params=par((len+1):end);
348         end
349 
350         if ~isempty(copula.params) && copula.const==0
351             [copula.params lik]=fmincon(@optimfun9,copula.params
    ,[],[],[],[],copula.lower,copula.upper,[],options.optim,margin,
    trend,correlation,anisotropy,copula,data,h,Rinv,logSqrtDetSigma,
    options);
352         end
353 
354         if ~isempty(correlation.params) && correlation.const==0
355             [correlation.params lik]=fmincon(@optimfun10,correlation.
```

```
    params,[],[],[],[],correlation.lower,correlation.upper,[],options
    .optim,margin,trend,correlation,anisotropy,copula,data,h,options)
    ;
356         end
357
358         if ~isempty(anisotropy.params) && anisotropy.const==0
359             [anisotropy.params lik]=fmincon(@optimfun11,anisotropy.
    params,[],[],[],[],anisotropy.lower,anisotropy.upper,[],options.
    optim,margin,trend,correlation,anisotropy,copula,data,h,options);
360         end
361
362         if options.print>=1
363           disp(['Profile Likelihood increase: ' num2str(abs(lik-
    oldlik))]);
364             disp([num2str(lik) '    ' num2str(round([margin.params
    trend.params correlation.params anisotropy.params copula.params
    ]*1000)/1000)])
365         end
366      end
367
368      if options.print>=1
369         disp('Final result:');
370         disp([num2str(lik) '    ' num2str(round([margin.params trend.
    params correlation.params anisotropy.params copula.params]*100)
    /100)])
371      end
372
373      optimized.margin=margin;
374      optimized.correlation=correlation;
375      optimized.anisotropy=anisotropy;
376      optimized.trend=trend;
377      optimized.copula=copula;
378      optimized.options=options;
379
380
381 function val =optimfun8(params,margin,trend,correlation,anisotropy,
```

```
    copula,data,h,len,Rinv,logSqrtDetSigma,options)
382         margin.params=params(1:len);
383         trend.params=params((len+1):end);
384         val=loglikelihoodStudent(margin,trend,correlation,anisotropy,
    copula,data,h,Rinv,logSqrtDetSigma,0,options);
385
386 function val =optimfun9(params,margin,trend,correlation,anisotropy,
    copula,data,h,Rinv,logSqrtDetSigma,options)
387         copula.params=params;
388         val=loglikelihoodStudent(margin,trend,correlation,anisotropy,
    copula,data,h,Rinv,logSqrtDetSigma,0,options);
389
390
391 function val =optimfun10(params,margin,trend,correlation,anisotropy,
    copula,data,h,options)
392         correlation.params=params;
393         val=loglikelihoodStudent(margin,trend,correlation,anisotropy,
    copula,data,h,[],[],0,options);
394
395 function val =optimfun11(params,margin,trend,correlation,anisotropy,
    copula,data,h,options)
396         anisotropy.params=params;
397         val=loglikelihoodStudent(margin,trend,correlation,anisotropy,
    copula,data,h,[],[],1,options);
398
399 function like = loglikelihoodStudent(margin,trend,correlation,
    anisotropy,copula,data,h,Rinv,logSqrtDetSigma,anisomode,options)
400
401     if anisomode
402             xy=[cos(anisotropy.params(1)),sin(anisotropy.params(1));-
    anisotropy.params(2)*sin(anisotropy.params(1)),anisotropy.params
    (2)*cos(anisotropy.params(1))]*[data.x';data.y'];
403             x=xy(1,:)';
404             y=xy(2,:)';
405     end
406
```

```
407
408     expected=trend.F*trend.params';
409     if length(margin.params)==1
410         z=feval([margin.name 'cdf'],data.z,expected,margin.params);
411         zz=feval([margin.name 'pdf'],data.z,expected,margin.params);
412     elseif length(margin.params)==2
413         z=feval([margin.name 'cdf'],data.z,expected,margin.params(1),
    margin.params(2));
414         zz=feval([margin.name 'pdf'],data.z,expected,margin.params(1)
    ,margin.params(2));
415     end
416
417     n=length(z);
418     ok=sum(z~=1)==n;
419     ok=ok*(sum(z~=0)==n);
420     nu=copula.params;
421
422     if ok==1
423         z=tinv(z,nu);
424         zzz=tpdf(z,nu);
425         if isempty(Rinv) || isempty(logSqrtDetSigma)
426             if anisomode
427                 h=pdist([x y]);
428                 h=squareform(h);
429             end
430             cov=correlCovMat(h,correlation.params,correlation.model);
431             R = chol(cov,'upper');
432             xRinv = z' / R;
433             logSqrtDetSigma = sum(log(diag(R)));
434             quadform = sum(xRinv.^2, 2);
435             like=gammaln(nu/2)-gammaln((nu+n)/2)+n/2*log(nu)+0.5*(nu+
    n)*log(1+1/nu*quadform)+logSqrtDetSigma-sum(log(zz))+sum(log(zzz)
    );
436         else
437             xRinv = z'*Rinv;
438             quadform = sum(xRinv.^2, 2);
```

```
439             like=gammaln(nu/2)-gammaln((nu+n)/2)+n/2*log(nu)+0.5*(nu+
    n)*log(1+1/nu*quadform)+logSqrtDetSigma-sum(log(zz))+sum(log(zzz)
    );
440         end
441
442         if isnan(like)
443             like=10^100;
444         end
445     else
446         like=10^100;
447     end
448
449     if options.print>=2
450         disp([num2str(like) ' ' num2str(round([margin.params trend.
    params correlation.params anisotropy.params copula.params]*1000)
    /1000)])
451     end
```

A.2 CopulaPredict

```
1  function prediction = copulaPredict(locations,data,model,search,calc,
       options)
2
3  %
4  % Inputs:
5  %
6  % locations...structure array representing the x-coordinates (
       locations.x),
7  %             y-coordinates (locations.y) and design matrix (
       locations.F)
8  %             of locations where prediction is desired
9  %
10 % data.........the data set as a structure array of the format data.x
       (x-coordinates), data.y
11 %             (y-coordinates), data.z (measured values)
```

```
12 %
13 % model.......contains the parameters of the spatial copula model (
   see
14 %             function initialize)
15 %
16 % search......search radius for interpolation
17 %
18 % calc........structure array indicating which predictive quantities
   to compute
19 %             calc.mean.....either 1 or 0 depending on whether to
   compute
20 %                           the mean of the predictive distribution
   or not
21 %             calc.std.....either 1 or 0 depending on whether to
   compute
22 %                           the standard deviation of the predictive
23 %                           distribution or not
24 %             calc.quantiles....vector of quantiles to compute (empty
    if no
25 %                               quantiles are sought)
26 %
27 % options.....options structure array (see function testOptions)
28 %
29 % Outputs:
30 %
31 % prediction...structure array consisting of the prediction results
32 %              prediction.mean...vector of predictive means
33 %              prediction.std....vector of predictive standard
   deviations
34 %              prediction.quantiles...matrix (number of locations x
   number
35 %                                     of quantiles) consisting of
   predictive
36 %                                     quantiles
37 %
38
```

```
39
40
41      options=testOptions(options);
42
43      meanValue=NaN.*zeros(length(locations.x),1);
44      varValue=NaN.*zeros(length(locations.x),1);
45      if ~isempty(calc.quantiles)
46          quantilesValue=NaN.*zeros(length(locations.x),length(calc.
    quantiles));
47      else
48          quantilesValue=NaN.*zeros(length(locations.x),1);
49      end
50
51      if ~isempty(model.anisotropy)
52          xy=[cos(model.anisotropy.params(1)),sin(model.anisotropy.
    params(1));-model.anisotropy.params(2)*sin(model.anisotropy.
    params(1)),model.anisotropy.params(2)*cos(model.anisotropy.params
    (1))]*[data.x';data.y'];
53          data.x=xy(1,:)';
54          data.y=xy(2,:)';
55          xy=[cos(model.anisotropy.params(1)),sin(model.anisotropy.
    params(1));-model.anisotropy.params(2)*sin(model.anisotropy.
    params(1)),model.anisotropy.params(2)*cos(model.anisotropy.params
    (1))]*[locations.x';locations.y'];
56          locations.x=xy(1,:)';
57          locations.y=xy(2,:)';
58      end
59
60      expected=model.trend.F*model.trend.params';
61      if length(model.margin.params)==1
62          newdata=feval([model.margin.name 'cdf'],data.z,expected,model
    .margin.params);
63      elseif length(model.margin.params)==2
64          newdata=feval([model.margin.name 'cdf'],data.z,expected,model
    .margin.params(1),model.margin.params(2));
65      end
```

```
66
67      multeps=[];
68      if isempty(model.copula) || strcmp(model.copula.name,'Gaussian')
69          newdata=norminv(newdata,0,1);
70      else
71          switch model.copula.name
72              case 'Chi2'
73                  ijmatrix=zeros(2^(search+1),search+1);
74                  for j=1:(search+1)
75                      ijmatrix(:,j)=kron(mod(0:(2^(search-j+2)-1),2)',
    ones(2^(j-1),1));
76                  end
77                  multeps=(-1).^ijmatrix;
78                  newdata=ncx2inv(newdata,1,model.copula.params);
79              case 'Student'
80                  newdata=tinv(newdata,model.copula.params);
81          end
82      end
83
84      h=pdist([data.x data.y]);
85      h=squareform(h);
86
87       if options.parallel==1
88              parpool
89          options.parallel=0;
90          parfor i=1:length(locations.x)
91              [meanValue(i),varValue(i),quantilesValue(i,:)]=
    copulaPredictLoop(i,locations,data,model,search,calc,options,
    newdata,h,multeps);
92          end
93      else
94          for i=1:length(locations.x)
95              [meanValue(i),varValue(i),quantilesValue(i,:)]=
    copulaPredictLoop(i,locations,data,model,search,calc,options,
    newdata,h,multeps);
96          end
```

```
97      end
98
99      prediction=[];
100     if  calc.mean==1
101         prediction.mean=meanValue;
102     end
103     if calc.std==1
104         prediction.std=sqrt(varValue);
105     end
106     if ~isempty(calc.quantiles)
107         prediction.quantiles=quantilesValue;
108     end
109
110 function [meanValue varValue quantilesValue]=copulaPredictLoop(i,
    locations,data,model,search,calc,options,newdata,h,multeps)
111
112         meanValue=NaN;
113         varValue=NaN;
114         quantilesValue=NaN;
115         if ~isnan(locations.x(i))
116             distances=sqrt((locations.x(i)-data.x).^2+(locations.y(i)
    -data.y).^2);
117             [distances index]=sort(distances);
118             distances=distances(1:(search))';
119             index=index(1:(search));
120             F=locations.F(i,:)*model.trend.params';
121             sigma=correlCovMat([0 distances;[distances' h(index,index
    )]],model.correlation.params,model.correlation.model);
122
123
124             if calc.mean==1
125                 if isempty(model.copula) || strcmp(model.copula.name,
    'Gaussian')
126                     temp=sigma(1,2:end)/sigma(2:end,2:end);
127                     m=temp*newdata(index);
128                     v=sqrt(1-temp*sigma(2:end,1));
```

```
129                     meanValue=quad(@meanintegrateGaussian,0,1,1e-9,0,
    m,v,model.margin,F);
130                 else
131                     switch model.copula.name
132                         case 'Chi2'
133                             sigmaChol=chol(sigma,'upper');
134                             sigma22Chol=chol(sigma(2:end,2:end),'
    upper');
135                             eps=multeps(1:2^search,1:search).*kron(
    ones(2^search,1),sqrt(newdata(index))');
136                             c=sum(mvnpdf_noinvert(eps,sqrt(model.
    copula.params)*ones(1,search),sigma22Chol));
137                             meanValue=quad(@meanintegrateChi2,0,1,1e
    -5,0,model.copula.params,sigmaChol,multeps,newdata(index),search,
    model.margin,F,c);
138                         case 'Student'
139                             temp=sigma(1,2:end)/sigma(2:end,2:end);
140                             m=temp*newdata(index);
141                             v=sqrt(model.copula.params/(model.copula.
    params+search)*(1+newdata(index)'*(sigma(2:end,2:end)\newdata(
    index))/model.copula.params)*(1-temp*sigma(2:end,1)));
142                             meanValue=quad(@meanintegrateStudent
    ,0,1,1e-9,0,m,v,model.margin,F,model.copula.params,search);
143                     end
144                 end
145             end
146
147             if calc.std==1 && calc.mean==1
148                 if isempty(model.copula) || strcmp(model.copula.name,
    'Gaussian')
149                   varValue=quad(@varintegrateGaussian,0,1,1e-9,0,m,v,
    model.margin,F,meanValue);
150                 else
151                     switch model.copula.name
152                         case 'Chi2'
153                             varValue=quad(@varintegrateChi2,0,1,1e
```

```
-5,0,model.copula.params,sigmaChol,multeps,newdata(index),search,
model.margin,F,meanValue,c);
                    case 'Student'
                        varValue=quad(@varintegrateStudent,0,1,1e
-9,0,m,v,model.margin,F,meanValue,model.copula.params,search);%%
                end
            end
        end

        if ~isempty(calc.quantiles) && (isempty(model.copula) ||
strcmp(model.copula.name,'Gaussian') || strcmp(model.copula.name,
'Student'))
            test=0.1:0.1:0.9;
            integr=zeros(length(test),1);
            quantilesValue=zeros(1,length(calc.quantiles));
            if isempty(model.copula) || strcmp(model.copula.name,
'Gaussian')
                for zz=1:length(test)
                    integr(zz)=quad(@condcopuladensGaussian,0,
test(zz),1e-5,0,m,v);
                end
            else
                switch model.copula.name
                    case 'Student'
                        for zz=1:length(test)
                            integr(zz)=quad(
@condcopuladensStudent,0,test(zz),1e-5,0,m,v,model.copula.params,
search);%%
                        end
                end
            end
            for j=1:length(calc.quantiles)
                for zz=1:length(test)
                    val(zz)=calc.quantiles(j)-integr(zz);
                end
                zz=find(val.^2==min(val.^2));
```

```
181                     q=test(zz);
182
183                     if length(test)==1
184                         if val(zz)>=0
185                             q1=q;
186                             f1=val(zz);
187                             q2=0.9999999;
188                             f2=calc.quantiles(j)-1;
189                         else
190                             q1=0.0000001;
191                             f1=calc.quantiles(j);
192                             q2=q;
193                             f2=val(zz);
194                         end
195                     elseif zz~=length(test) && zz~=1
196                         if val(zz)>=0
197                             q1=q;
198                             f1=val(zz);
199                             q2=test(zz+1);
200                             f2=val(zz+1);
201                         else
202                             q1=test(zz-1);
203                             f1=val(zz-1);
204                             q2=q;
205                             f2=val(zz);
206                         end
207                     elseif zz==1
208                         if val(zz)>=0
209                             q1=q;
210                             f1=val(zz);
211                             q2=test(zz+1);
212                             f2=val(zz+1);
213                         else
214                             q1=0.0000001;
215                             f1=calc.quantiles(j);
216                             q2=q;
```

```
217                             f2=val(zz);
218                         end
219                     else
220                         if val(zz)>=0
221                             q1=q;
222                             f1=val(zz);
223                             q2=0.9999999;
224                             f2=calc.quantiles(j)-1;
225                         else
226                             q1=test(zz-1);
227                             f1=val(zz-1);
228                             q2=q;
229                             f2=val(zz);
230                         end
231                     end
232
233                     if sign(f1)~=-sign(f2)
234                         q1=0;
235                         q2=1;
236                     end
237
238                     if isempty(model.copula) || strcmp(model.copula.
    name,'Gaussian')
239                         [quantilesValue(j) val]=fzero(
    @quantilesGaussian,[q1,q2],[],m,v,calc.quantiles(j));
240                     else
241                         switch model.copula.name
242                             case 'Student'
243                                 [quantilesValue(j) val]=fzero(
    @quantilesStudent,[q1,q2],[],m,v,calc.quantiles(j),model.copula.
    params,search);%%
244                         end
245                     end
246                 end
247
248                 if length(model.margin.params)==1
```

```
249                     quantilesValue=feval([model.margin.name 'inv'],
    quantilesValue,F,model.margin.params);
250                 elseif length(model.margin.params)==2
251                     quantilesValue=feval([model.margin.name 'inv'],
    quantilesValue,F,model.margin.params(1),model.margin.params(2));
252                 end
253             end
254         end
255         if options.print>=1
256             disp(['Prediction: Location ' num2str(i)]);
257         end
258
259 %%
260 %GAUSSIAN COPULA
261
262 function val = quantilesGaussian(loc,m,v,quant)
263     if loc<1 && loc >0
264         val=quant-quad(@condcopuladensGaussian,0,loc,1e-9,0,m,v);
265     else
266         val=1e10;
267     end
268
269
270
271 function d=meanintegrateGaussian(loc,m,v,margin,expected)
272     if length(margin.params)==1
273         locx=feval([margin.name 'inv'],loc,expected,margin.params);
274     elseif length(margin.params)==2
275         locx=feval([margin.name 'inv'],loc,expected,margin.params(1),
    margin.params(2));
276     end
277     d=locx.*condcopuladensGaussian(loc,m,v);
278     if sum(isnan(d))>0
279         d(isnan(d))=0;
280     end
281
```

```
282
283 function d=varintegrateGaussian(loc,m,v,margin,expected,mean1)
284     if length(margin.params)==1
285         locx=feval([margin.name 'inv'],loc,expected,margin.params);
286     elseif length(margin.params)==2
287         locx=feval([margin.name 'inv'],loc,expected,margin.params(1),
    margin.params(2));
288     end
289     d=(locx-mean1).^2.*condcopuladensGaussian(loc,m,v);
290     if sum(isnan(d))>0
291         d(isnan(d))=0;
292     end
293
294
295 function distrib=condcopuladensGaussian(loc,m,v)
296     loc=norminv(loc,0,1);
297     div=normpdf(loc,0,1);
298     distrib=normpdf(loc,m,v)./div;
299
300
301 %%
302 %STUDENT COPULA
303 function val = quantilesStudent(loc,m,v,quant,nu,search)
304     if loc<1 && loc >0
305         val=quant-quad(@condcopuladensStudent,0,loc,1e-9,0,m,v,nu,
    search);
306     else
307         val=1e10;
308     end
309
310
311
312 function d=meanintegrateStudent(loc,m,v,margin,expected,nu,search)
313     if length(margin.params)==1
314         locx=feval([margin.name 'inv'],loc,expected,margin.params);
315     elseif length(margin.params)==2
```

```
316         locx=feval([margin.name 'inv'],loc,expected,margin.params(1),
    margin.params(2));
317     end
318     d=locx.*condcopuladensStudent(loc,m,v,nu,search);
319     if sum(isnan(d))>0
320         d(isnan(d))=0;
321     end
322
323
324 function d=varintegrateStudent(loc,m,v,margin,expected,mean1,nu,
    search)
325     if length(margin.params)==1
326         locx=feval([margin.name 'inv'],loc,expected,margin.params);
327     elseif length(margin.params)==2
328         locx=feval([margin.name 'inv'],loc,expected,margin.params(1),
    margin.params(2));
329     end
330     d=(locx-mean1).^2.*condcopuladensStudent(loc,m,v,nu,search);
331     if sum(isnan(d))>0
332         d(isnan(d))=0;
333     end
334
335
336 function distrib=condcopuladensStudent(loc,m,v,nu,search)
337     loc=tinv(loc,nu);
338     div=tpdf(loc,nu);
339     distrib=(tpdf((loc-m)./v,nu+search)./div)./v;
340
341
342 %%
343 %CHI^2-COPULA
344
345 function distrib=condcopuladensChi2(loc,lambda,sigmaChol,multeps,
    newdata,search,c)
346     loc=ncx2inv(loc,1,lambda);
347     div=ncx2pdf(loc,1,lambda);
```

```
348
349     distrib=zeros(length(loc),1);
350     for i=1:length(loc)
351         eps=multeps.*kron(ones(2^(search+1),1),[sqrt(loc(i));sqrt(
    newdata)]');
352         distrib(i)=sum(mvnpdf_noinvert(eps,sqrt(lambda)*ones(1,search
    +1),sigmaChol))/(2*sqrt(loc(i))*div(i)*c);
353     end
354
355
356 function d=meanintegrateChi2(loc,lambda,sigmaChol,multeps,newdata,
    search,margin,expected,c)
357     if length(margin.params)==1
358         locx=feval([margin.name 'inv'],loc,expected,margin.params);
359     elseif length(margin.params)==2
360         locx=feval([margin.name 'inv'],loc,expected,margin.params(1),
    margin.params(2));
361     end
362
363     d=locx.*condcopuladensChi2(loc,lambda,sigmaChol,multeps,newdata,
    search,c)';
364     if sum(isnan(d))>0
365         d(isnan(d))=0;
366     end
367
368
369 function d=varintegrateChi2(loc,lambda,sigmaChol,multeps,newdata,
    search,margin,expected,mean1,c)
370     if length(margin.params)==1
371         locx=feval([margin.name 'inv'],loc,expected,margin.params);
372     elseif length(margin.params)==2
373         locx=feval([margin.name 'inv'],loc,expected,margin.params(1),
    margin.params(2));
374     end
375     d=(locx-mean1).^2.*condcopuladensChi2(loc,lambda,sigmaChol,
    multeps,newdata,search,c)';
```

```
376     if sum(isnan(d))>0
377         d(isnan(d))=0;
378     end
```

A.3 CopulaPredictDensity

```
1  function dens = copulaPredictDensity(locations,data,model,search,grid
       ,options)
2
3  %
4  % Inputs:
5  %
6  % locations...structure array representing the x-coordinates (
       locations.x),
7  %             y-coordinates (locations.y) and design matrix (
       locations.F)
8  %             of locations where prediction is desired
9  %
10 % data........the data set as a structure array of the format data.x
       (x-coordinates), data.y
11 %             (y-coordinates), data.z (measured values)
12 %
13 % model.......contains the parameters of the spatial copula model (
       see
14 %             function initialize)
15 %
16 % search......search radius for interpolation
17 %
18 % grid........the grid on which the density is computed (
       discretization of
19 %             the support of the predictive distribution)
20 %
21 % options.....options structure array (see function testOptions)
22 %
23 % Outputs:
```

```
%
% dens.....matrix (number of observations x grid size) consisting of the
%             predictive densities
%


    options=testOptions(options);

    if ~isempty(model.anisotropy)
        xy=[cos(model.anisotropy.params(1)),sin(model.anisotropy.params(1));-model.anisotropy.params(2)*sin(model.anisotropy.params(1)),model.anisotropy.params(2)*cos(model.anisotropy.params(1))]*[data.x';data.y'];
        data.x=xy(1,:)';
        data.y=xy(2,:)';
        xy=[cos(model.anisotropy.params(1)),sin(model.anisotropy.params(1));-model.anisotropy.params(2)*sin(model.anisotropy.params(1)),model.anisotropy.params(2)*cos(model.anisotropy.params(1))]*[locations.x';locations.y'];
        locations.x=xy(1,:)';
        locations.y=xy(2,:)';
    end

    expected=model.trend.F*model.trend.params';
    if length(model.margin.params)==1
        newdata=feval([model.margin.name 'cdf'],data.z,expected,model.margin.params);
    elseif length(model.margin.params)==2
        newdata=feval([model.margin.name 'cdf'],data.z,expected,model.margin.params(1),model.margin.params(2));
    end

    multeps=[];
    if isempty(model.copula) || strcmp(model.copula.name,'Gaussian')
        newdata=norminv(newdata,0,1);
```

```
    else
        switch model.copula.name
            case 'Chi2'
                ijmatrix=zeros(2^(search+1),search+1);
                for j=1:(search+1)
                    ijmatrix(:,j)=kron(mod(0:(2^(search-j+2)-1),2)',
   ones(2^(j-1),1));
                end
                multeps=(-1).^ijmatrix;
                newdata=ncx2inv(newdata,1,model.copula.params);
            case 'Student'
                newdata=tinv(newdata,model.copula.params);
        end
    end


    h=pdist([data.x data.y]);
    h=squareform(h);
    dens=NaN.*zeros(length(locations.x),length(grid));

    if options.parallel==1
            parpool
        options.parallel=0;
        parfor i=1:length(locations.x)
            dens(i,:)=copulaPredictDensityLoop(i,locations,data,model
   ,search,grid,options,newdata,h,multeps);
        end
    else
        for i=1:length(locations.x)
            dens(i,:)=copulaPredictDensityLoop(i,locations,data,model
   ,search,grid,options,newdata,h,multeps);
        end
     end


function dens=copulaPredictDensityLoop(i,locations,data,model,search,
```

```
    grid,options,newdata,h,multeps)
84
85          if ~isnan(locations.x(i))
86              F=locations.F(i,:)*model.trend.params';
87              if length(model.margin.params)==1
88                  gridx=feval([model.margin.name 'cdf'],grid,F,model.
    margin.params);
89                  jacobi=feval([model.margin.name 'pdf'],grid,F,model.
    margin.params);
90              elseif length(model.margin.params)==2
91                  gridx=feval([model.margin.name 'cdf'],grid,F,model.
    margin.params(1),model.margin.params(2));
92                  jacobi=feval([model.margin.name 'pdf'],grid,F,model.
    margin.params(1),model.margin.params(2));
93              end
94
95              if isempty(model.copula) || strcmp(model.copula.name,'
    Gaussian')
96                  gridx=norminv(gridx);
97                  div=normpdf(gridx);
98              else
99                  switch model.copula.name
100                     case 'Chi2'
101                         gridx=ncx2inv(gridx,1,model.copula.params);
102                         div=ncx2pdf(gridx,1,model.copula.params);
103                     case 'Student'
104                         gridx=tinv(gridx,model.copula.params);
105                         div=tpdf(gridx,model.copula.params);
106                 end
107             end
108
109             distances=sqrt((locations.x(i)-data.x).^2+(locations.y(i)
    -data.y).^2);
110             [distances index]=sort(distances);
111             distances=distances(1:(search))';
112             index=index(1:(search));
```

```
113
114
115             covi=correlCovMat([0 distances;[distances' h(index,index)
    ]],model.correlation.params,model.correlation.model);
116             if isempty(model.copula) || strcmp(model.copula.name,'
    Gaussian')
117                 u=covi(1,2:end)/covi(2:end,2:end);
118                 dens=(normpdf(gridx,u*newdata(index),sqrt(1-u*covi(2:
    end,1))))./div).*jacobi;
119             else
120                 switch model.copula.name
121                     case 'Chi2'
122                         dens=zeros(1,length(grid));
123                         newdataindex=sqrt(newdata(index));
124                         eps=multeps(1:2^search,1:search).*kron(ones
    (2^search,1),newdataindex');
125                         c=sum(mvnpdf(eps,sqrt(model.copula.params)*
    ones(1,search),covi(2:end,2:end)));
126                         R = chol(covi,'upper');
127                         d=size(covi,1);
128                         logSqrtDetSigma = sum(log(diag(R)));
129                         mu=repmat(sqrt(model.copula.params)*ones(1,
    search+1),size(multeps,1),1);
130                         Rinv=eye(size(covi)) / R;
131                         for j=1:length(grid)
132                             eps=multeps.*kron(ones(2^(search+1),1),[
    sqrt(gridx(j));newdataindex]');
133                             xRinv=(eps - mu)*Rinv;
134                             quadform = sum(xRinv.^2, 2);
135                             dens(j)=sum(exp(-0.5*quadform -
    logSqrtDetSigma - d*log(2*pi)/2))/(2*sqrt(gridx(j))*div(j)*c);
136                         end
137                         dens=dens.*jacobi;
138                     case 'Student'
139                         u=covi(1,2:end)/covi(2:end,2:end);
140                         m=u*newdata(index);%m=covi(1,2:end)*
```

```
    sigma22inv*newdata(index);
141                         v=sqrt(model.copula.params/(model.copula.
    params+search)*(1+newdata(index)'*(covi(2:end,2:end)\newdata(
    index))/model.copula.params)*(1-u*covi(2:end,1)));
142                         dens=((tpdf((gridx-m)./v,model.copula.params+
    search)./div)./v).*jacobi;
143                 end
144             end
145
146         end
147         if options.print>=1
148             disp(['Prediction: Location ' num2str(i)]);
149         end
```

A.4 CopulaCV

```
1  function [cv stat]=copulaCV(data,model,search,options)
2
3
4  %
5  % See: copulaPredict
6  %
7  % Inputs:
8  %
9  % data........the data set as a structure array of the format data.x
       (x-coordinates), data.y
10 %              (y-coordinates), data.z (measured values)
11 %
12 % model.......contains the parameters of the spatial copula model (
       see
13 %              function initialize)
14 %
15 % search......search radius for interpolation
16 %
17 % options.....options structure array (see function testOptions)
```

```
%
% Outputs:
%
% cv..........cross validation predictions
%
% stat........cross validation summary statistics
%              stat.mse.....mean squared error
%              stat.corr....correlation between true and predicted
    values
%              stat.mae.....mean absolute error
%              stat.mad.....median absolute deviation
%              stat.bias....prediction bias
%

    options=testOptions(options);

    n=length(data.x);
    cv=zeros(n,1);
    if options.parallel==1
           parpool
        options.parallel=0;
        parfor i=1:n
            cv(i)=copulaCVLoop(i,data,model,search,n,options);
        end
    else
        for i=1:n
            cv(i)=copulaCVLoop(i,data,model,search,n,options);
        end
    end

    stat.mse=mean((data.z-cv).^2);
    stat.corr=corr(data.z,cv);
    stat.mae=mean(abs(data.z-cv));
    stat.mad=median(abs(data.z-cv));
    stat.bias=mean(cv-data.z);

```

```
function cv=copulaCVLoop(i,data,model,search,n,options)
    locationsnew.x=data.x(i);
    locationsnew.y=data.y(i);
    locationsnew.z=data.z(i);
    locationsnew.F=model.trend.F(i,:);
    data.z=data.z(setdiff(1:n,i));
    data.x=data.x(setdiff(1:n,i));
    data.y=data.y(setdiff(1:n,i));
    model.trend.F=model.trend.F(setdiff(1:n,i),:);
    calc.mean=1;
    calc.quantiles=[];
    calc.std=0;
    temp=options.print;
    options.print=0;
    prediction=copulaPredict(locationsnew,data,model,search,calc,
   options);
    options.print=temp;
    if options.print>=1
        disp(['Prediction: Location ' num2str(i)]);
    end
    if options.print>=2
        disp(['True: ' num2str(locationsnew.z) ', Predicted: '
   num2str(prediction.mean)]);
    end
    cv=prediction.mean;
```

A.5 CopulaCVDensity

```
function dens=copulaCVDensity(data,model,search,grid,options)


%
% See: copulaPredictDensity
%
% Inputs:
```

```
8  %
9  % data........the data set as a structure array of the format data.x
       (x-coordinates), data.y
10 %              (y-coordinates), data.z (measured values)
11 %
12 % model.......contains the parameters of the spatial copula model (
       see
13 %              function initialize)
14 %
15 % search......search radius for interpolation
16 %
17 % grid........the grid on which the density is computed (
       discretization of
18 %              the support of the predictive distribution)
19 %
20 % options.....options structure array (see function testOptions)
21 %
22 % Outputs:
23 %
24 % dens........matrix (number of observations x grid size) consisting
       of the
25 %              cross validation predictive densities
26 %
27
28     options=testOptions(options);
29
30     n=length(data.x);
31     dens=zeros(n,length(grid));
32     if options.parallel==1
33             parpool
34         options.parallel=0;
35         parfor i=1:n
36             dens(i,:)=copulaCVDensityLoop(i,data,model,search,grid,n,
       options);
37         end
38     else
```

```
        for i=1:n
            dens(i,:)=copulaCVDensityLoop(i,data,model,search,grid,n,
    options);
        end
    end

function dens=copulaCVDensityLoop(i,data,model,search,grid,n,options)
  locationsnew.x=data.x(i);
  locationsnew.y=data.y(i);
  locationsnew.z=data.z(i);
    locationsnew.F=model.trend.F(i,:);
    data.z=data.z(setdiff(1:n,i));
  data.x=data.x(setdiff(1:n,i));
  data.y=data.y(setdiff(1:n,i));
    model.trend.F=model.trend.F(setdiff(1:n,i),:);
    temp=options.print;
    options.print=0;
    dens=copulaPredictDensity(locationsnew,data,model,search,grid,
    options);
    options.print=temp;
    if options.print>=1
        disp(['Prediction: Location ' num2str(i)]);
    end
```

A.6 ComputeCoverage

```
function [coverage width]=computeCoverage(statistics,probabilities,
    dens,grid,z)

% Computes coverage and average width of confidence bounds.
%
% See: computeStatistics, copulaPredictDensity, copulaCVDensity
%
% Inputs:
%
```

```
% statistics......the output of computeStatistics(grid,dens,[],z)
%
% probabilities...the confidence level, e.g. if probabilities=97.5, then
%                 the 97.5% confidence interval is considered
%
% dens............(predictive) density computed on a grid e.g. by
%                  copulaPredictDensity
%
% grid............the grid on which the density is available
%
% z...............observations (z-values, data.z)
%
% Outputs:
%
% coverage........percentages of coverage of the true values by the
%                 confidence intervals
%
% width...........average width of probability intervals that include the
%                 true values "z"
%


    if nargin<5
        z=[];
        grid=[];
        dens=[];
        width=[];
    end
    if nargin<2
        probabilities=0.02:0.02:0.98;
    end

    n=length(statistics.PIT);
    coverage=zeros(1,length(probabilities));
```

```
43
44      for i=1:n
45          coverage=coverage+(((1-probabilities)/2)<statistics.PIT(i) &
    statistics.PIT(i)<=((1+probabilities)/2));
46      end
47      coverage=coverage./n;
48
49      if nargout==2 && ~isempty(z)
50          statistics=computeStatistics(dens,grid,[(1-probabilities)/2
    (1+probabilities)/2],z);
51          width=zeros(length(probabilities),1);%average width of
    probability intervals that include the true values (should be as
    small as possible)
52          for i=1:length(probabilities)
53              width(i)=1/(n*coverage(i))*sum(((((1-probabilities(i))/2)
    <statistics.PIT & statistics.PIT<=((1+probabilities(i))/2))).*(
    statistics.quantiles(:,length(probabilities)+i)-statistics.
    quantiles(:,i)));
54          end
55      end
56
57
58      plot(probabilities,coverage,'b')
59      hold on;plot([0 1],[0 1],'r')
60      xlabel('Probability interval - p')
61      ylabel('Proportion in this interval')
```

A.7 Résumé des fonctions de la librairie Matlab

TABLE A.1 – Tableau récapitulatif des fonctions de la librairie Matlab.

	Fonction	Description
Analyse exploratoire	ScatterPlotRanks	Produit des diagrammes de dispersion par paire basés sur des observations multivariées et les rangs standardisés associés.
	BivCopula_Density	Calcule la densité de la copule.
	SpatialTools_KendallOgram	Calcule la valeur du tau de Kendall en fonction de la distance pour chaque paire d'un ensemble de données spatiales avec des répliques.
	Plot_Lags	Trace la denstié de la copule empirique et théorique.
	Empirical_Copula	Calcule la copule empirique.
Initialisation	Initialize	Définit le modèle de copule spatiale.
	SpatialTools_CreationOfAGrid	Génère des positions sur le carré de taille 100×100 et calcule la matrice de toutes les distances entre ces positions.
	SpatialTools_Link	Renvoie la valeur d'une fonction de lien calculée en x.
	SpatialTools_PositiveDefinite	Indique si une matrice symétrique est définie positive, et si non, la transforme en une matrice définie positive.
	SpatialTools_SimulationChi2	Génére des champs aléatoires répliqués à partir de la copule du Chi-deux.
	SpatialTools_SimulationPairwiseCopula	Génère un champ aléatoire à partir d'une structure de dépendance sélectionnée (les marginaux sont uniformes).
	SpatialTools_PlotRandomField3d	Propose une représentation graphique d'un champ aléatoire Z observé à d endroits.
	Test_Options	Définit certaines options, par exemple l'affichage.
	Empirical_Variogram	Calcule le variogramme empirique.
	Estimate_Anisotropy	Estimation des paramètres d'anisotropie.
	VectorOfRanks	Retourne les vecteurs de rangs d'un ensemble de données à d-variables de taille n.
Estimation des paramètres	SpatialChi2_FullMLE	Estime les paramètres (theta, epsilon et a) d'une copule Khi-deux en utilisant les paires de ML.
	SpatialChi2_thetaMLE	Estime le paramèt theta d'une copule Khi-deux.
	Copula_Estimation	Estime les paramètres du modèle par le ML (Pour les copules Khi-deux, normale et Student).
	Correl_CovMat	Calcule la matrice de corrélation/covariance.
Interporalation spatiale	Copula_Predict	Réalise l'interpolation spatiale.
	Copula_PredictDensity	Calcule la densité prédite.
	Copula_CV	Effectue la validation croisée.
	Compute_Coverage	Calcule la largeur et la précision des intervalles de confiance.
	Copula_CVDensity	Effectue la validation croisée de la densité.
	Plot_CI	Trace des intervalles de confiance pour chaque endroit.
	Plot_PIT	Visualise les valeurs de PIT.
Jeu de données	MeuseCadmium	Base de données de meuse pour le métal Cadmium (Structure : (X,Y) = Coordonnées des valeurs observées et Z = Valeurs observées) .
	MeuseCadmiumGrid	Une grille d'interpolation pour les données MeuseCadmium