Simulation conditionnelle du processus de Schlather

Christian Lantuéjoul[1] & Liliane Bel[2]

[1] *MinesParisTech, Centre de Géosciences, F-77305 Fontainebleau, France*
christian.lantuejoul@mines-paristech.fr
[2] *AgroParisTech/INRA, UMR 518 Math. Info. Appli., F-75005 Paris, France*
liliane.bel@agroparistech.fr

Résumé. Les champs aléatoires max-stables permettent de modéliser des phénomènes extrêmes dans de nombreux domaines de l'environnement ou du climat. Leur distribution multivariée étant peu manipulable, il est utile de recourir à des simulations pour évaluer des risques ou envisager les conséquences de divers scénarios. Partant de l'algorithme de simulation conditionnelle mis au point par Dombry, Eyi-Minko et Ribatet (2012), nous proposons un nouvel algorithme qui s'appuie sur des variables latentes et qui s'avère plus rapide, plus précis et plus apte à traiter de grands jeux de données. Son efficacité est montrée sur le processus de Schlather.

Mots-clés. Max-stabilité, processus de Schlather, simulation conditionnelle, krigeage.

Abstract. Numerous extreme phenomena occuring in various environmental or climate frameworks can be modeled by max-stable processes. Because their multivariate distribution is generally not tractable, it is convenient to resort to simulations for planning hazard management or testing the consequences of diverse scenarios. Starting from conditional simulation algorithm designed by Dombry, Eyi-Minko and Ribatet (2012), we propose a new algorithm that is based on latent variables, and that turns out to be faster, more accurate and apter to handle large datasets. Its efficiciency is demonstrated on the Schlather process.

Keywords. Max-stability, Schlather process, conditional simulation, kriging.

1 Introduction

On s'intéresse à des processus max-stables s'écrivant

$$Z(s) = \bigvee_{n=1}^{\infty} \zeta_n Y_n(s) \qquad s \in I\!R^d \tag{1}$$

où les ζ_n sont les points d'un processus de Poisson sur $[0, \infty[$ et les Y_n sont des copies indépendantes d'un même champ aléatoire Y à réalisations continues. Des exemples classiques sont les processus de Schlather, t-extrêmaux et de Brown-Resnick. Une attention

plus particulière sera consacrée au processus de Schlather (2002) pour lequel les ζ_n ont la densité $\mu\zeta^{-2}d\zeta$ et Y est un champ gaussien standard de fonction de covariance C. La figure 1 montre une simulation non conditionnelle d'un tel processus ($\mu = 1$ et C est exponentielle de facteur d'échelle 15) dans un domaine rectangulaire 600×400. Les valeurs s'y échelonnent entre 0.03 et 7.66.

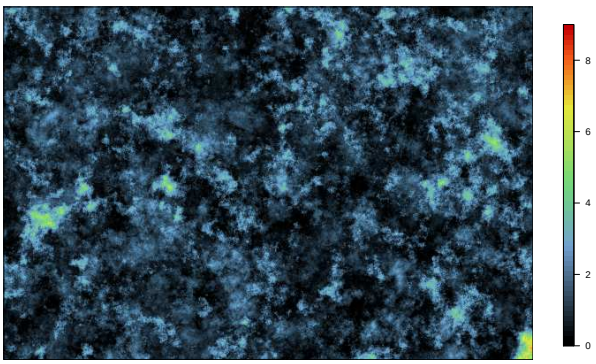

Figure 1: Simulation d'un processus de Schlather

On cherche à générer des réalisations de Z dans un domaine D, qui prennent les valeurs $z_{1:k} = (z_1, ..., z_k)$ aux points $s_{1:k} = (s_1, ..., s_k)$.

2 L'algorithme de Dombry, Eyi-Minko et Ribatet

Cet algorithme repose sur un certain nombre de propriétés, obtenues par Dombry et Eyi-Minko (2013). Tout d'abord, chaque point conditionnant est respecté par une fonction de base $\zeta_n Y_n$ et une seule. Il s'ensuit que le conditionnement est assuré par au plus k fonctions de base. Ces fonctions sont dites *actives* (ou extrêmales). Elles créent une partition de l'ensemble des points conditionnants, deux points appartenant au même bloc si et seulement s'ils sont respectés par la même fonction de base (cf. Fig. 2). De plus, une formule explicite a été obtenue pour la loi des partitions.
Par opposition, les fonctions de base qui ne jouent aucun rôle dans le conditionnement sont dites *passives* (ou non extrêmales). Les fonctions actives et le maximum des fonctions passives sont conditionnellement indépendantes.
La mise en œuvre de tous ces éléments conduit à l'algorithme suivant, proposé par Dombry, Eyi-Minko et Ribatet (2012):

(i) générer une partition $\mathcal{J} = (J_1, ..., J_n)$ *de* $s_{1:k}$ *sachant* $Z(s_{1:k}) = z_{1:k}$*;*
(ii) pour tout $i \in 1{:}n$ *générer* $Z_i^{(a)}$ *tel que* $Z_i^{(a)}(s_{J_i}) = z_{J_i}$ *et* $Z_i^{(a)}(s_{J_i^c}) < z_{J_i^c}$*;*
(iii) générer le maximum $Z^{(p)}$ *des champs passifs sachant* $Z(s_{1:k}) = z_{1:k}$*;*
(iv) poser $Z^{CS} = Z_1^{(a)} \vee \cdots \vee Z_n^{(a)} \vee Z^{(p)}$.

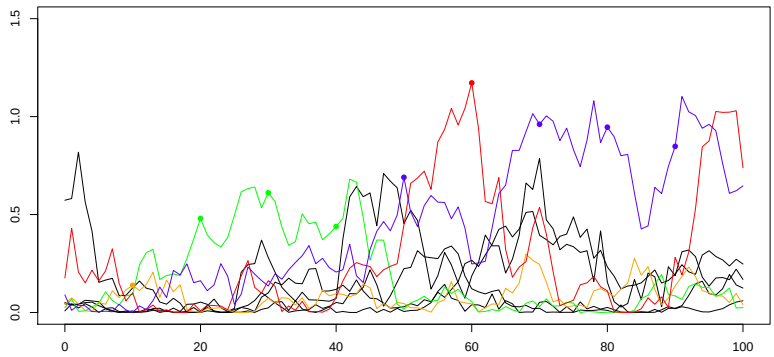

Figure 2: Partition des points conditionnants dessinée par les fonctions actives.

La mise en œuvre pratique de cet algorithme soulève plusieurs questions:

– L'étape *(i)* est plus délicate qu'il n'y parait. Il se trouve que l'espace des partitions est de dimension considérable (100 points conditionnants donnent lieu à 10^{116} partitions). De plus, la loi des partitions s'exprime comme un produit d'intégrales multidimensionnelles

$$p(\mathcal{J}) \propto \prod_{i=1}^{n} \int_{v_{i,J_i^c} < z_{J_i^c}} \theta(v_{i,J_i^c}, z_{J_i}) \, dv_{i,J_i^c}, \tag{2}$$

où $\theta(v_1 : k)$ représente la densité poissonnienne des valeurs prises par les fonctions de base $\zeta_n Y_n$ du processus sur l'ensemble des points conditionnants. L'évaluation de ces intégrales peut être une source importante d'imprécisions et d'instabilités numériques. Le recours à un algorithme itératif (l'échantillonneur de Gibbs) pour simuler la loi des partitions ne lève pas cette difficulté.

– L'étape *(ii)* peut se ramener à la simulation de chaque fonction active $\zeta_n Y_n$ sous contrainte d'inégalités. Plutôt qu'un algorithme de rejet qui perd en efficacité au fur et à mesure que le nombre de points conditionnants augmente, un algorithme à la Geweke (1991) peut être préconisé. Il s'agit d'un échantillonneur de Gibbs à deux états, chaque itération consistant à simuler Y_n à ζ_n connu, puis ζ_n à Y_n connu.

– L'étape *(iii)* est de loin la plus facile. Il suffit de générer une copie de Z en effaçant toutes les fonctions de base qui ne sont pas compatibles avec le conditionnement. Un problème mineur se pose qui est le choix d'un critère d'arrêt.

Dans ce qui suit, nous montrons comment s'affranchir du calcul des intégrales.

3 Un nouvel algorithme

Les intégrales de la formule (2) portent sur les valeurs prises par les fonctions actives aux points conditionnants qu'elles ne respectent pas. Si l'ensemble de ces valeurs était

simulé d'emblée, alors le calcul de ces intégrales ne serait pas nécessaire. C'est la raison qui nous amène à rechercher un algorithme qui commence par simuler non seulement la partition, mais aussi l'ensemble des valeurs prises par les fonctions actives *en tout point conditionnant*.

3.1 Un espace d'états matriciel

Dans ce nouveau contexte, l'objet à simuler n'est plus une partition, mais une matrice à n lignes (nombre - inconnu - de fonctions actives) et k colonnes (nombre de points conditionnants. Si v désigne une telle matrice, alors $v_{i,j}$ est la valeur prise par la fonction active i au point conditionnant s_j. La ligne v_i fournit les valeurs de la fonction active i aux points conditionnants, tandis que la colonne $v_{\cdot,j}$ fournit les valeurs prises par l'ensemble des fonctions actives au point conditionnant s_j. Plus généralement, $v_{i,J}$ contient les valeurs de la ligne i situées aux colonnes indexées par J.
Pour éviter de manipuler des matrices de dimension variable, on travaillera sur des matrices de k lignes. Cela nous amène à considérer l'ensemble $\mathcal{V}_k$ des matrices $k \times k$ qui vérifient les 3 propriétés suivantes:
– $v_{i,j} \leq z_j$;
– pour tout $j \in 1\!:\!k$, il existe un unique indice i tel que $v_{i,j} = z_j$;
– si $v_{i,j} = z_j$, alors $v_{i',j+1} = z_{j+1}$ pour un indice $i' \leq i+1$.
Les deux premières propriétés expriment que chaque point conditionnant est respecté par une fonction active et une seule. La troisième propriété ordonne les fonctions actives selon l'ordre attribué aux points conditionnants. Cette procédure, due à Orlov (2002) et reprise de Dombry et al. (2012), a l'avantage d'éliminer tout problème combinatoire que pourrait entrainer un positionnement arbitraire des fonctions actives dans la matrice.

3.2 Densité matricielle

Soit $v \in \mathcal{V}_k$, et soit $\mathcal{J}(v) = \{J_1, ..., J_n\}$ la partition qui lui est associée. On pose

$$f(v) \propto \prod_{i=1}^{n} \theta\big(v_{i,J_i^c}, z_{J_i}\big) \prod_{i=n+1}^{k} \tau(v_i) \tag{3}$$

où τ est une densité de probabilité sur $\prod_{j=1}^{k}]-\infty, z_j[$ que l'on sait facilement simuler. Par exemple, on pourra prendre τ à composantes indépendantes, i.e. $\tau(v_i) = \prod_{j\in 1:k} \tau_j(v_{ij})$, τ_j étant une densité de Cauchy tronquée à z_j. On vérifie facilement que la densité f est compatible avec la loi des partitions

$$\int_{\mathcal{J}(v)=\mathcal{J}} f(v)\, dv \;\propto\; \prod_{i=1}^{n} \int_{v_{i,J_i^c} < z_{J_i^c}} \theta(v_{i,J_i^c}, z_{J_i})\, dv_{i,J_i^c} \;\propto\; p(\mathcal{J})$$

3.3 Algorithme de simulation

Il s'agit d'un algorithme itératif. A chaque itération, un point conditionnant j est sélectionné au hasard et la valeur z_j de l'observation correspondante est relocalisée sur une autre ligne de la matrice v tandis qu'une valeur est générée selon la loi τ_j pour la remplacer. Cette matrice est ensuite réordonnée selon le critère d'Orlov pour servir de matrice propositionnelle. L'exemple qui suit illustre ce principe de contruction. La valeur sélectionnée z_1 est déplacée en ligne 3 et remplacée par une autre valeur $w_{1,1}$ tirée selon la loi τ_1.

$$v = \begin{pmatrix} z_1 & v_{1,2} & z_3 \\ v_{2,1} & z_2 & v_{2,3} \\ v_{3,1} & v_{3,2} & v_{3,3} \end{pmatrix} \longrightarrow \begin{pmatrix} w_{1,1} & v_{1,2} & z_3 \\ v_{2,1} & z_2 & v_{2,3} \\ z_1 & v_{3,2} & v_{3,3} \end{pmatrix} \longrightarrow \begin{pmatrix} z_1 & v_{3,2} & v_{3,3} \\ v_{2,1} & z_2 & v_{2,3} \\ w_{1,1} & v_{1,2} & z_3 \end{pmatrix} = w$$

La matrice propositionnelle w ainsi obtenue est acceptée au dépens de la matrice courante v, ou bien rejetée selon un critère de Metropolis-Hastings. Voici l'algorithme en question:
(i) initialiser v;
(ii) générer $j \sim \mathcal{U}(1:k)$, et soit i tel que $j \in J_i$;
(iii) générer $i' \sim \mathcal{U}\big(1:|\mathcal{J}| + 1_{|J_i|>1}\big)$ tant que $i' = i$;
(iv) poser $w = v$, $w_{i',j} = z_j$, et générer $w_{i,j} \sim \tau_j$ ainsi que $u \sim \mathcal{U}$;
(v) poser $v = w$ si $u < \dfrac{f(w)}{f(v)} \dfrac{\tau_j(v_{i,j})}{\tau_j(w_{i',j})}$;
(vi) aller en (ii).
On remarquera que le taux d'acceptation peut être réduit à une combinaison de 4 termes

$$\frac{f(w)}{f(v)} = \frac{\theta(w_{i'})}{\theta(v_i)} \frac{\theta(w_i)1_{|J_i|>1} + \tau(w_i)1_{|J_i|=1}}{\theta(v_{i'})1_{|J_{i'}|>0} + \tau(v_{i'})1_{|J_{i'}|=0}}$$

et cela quelque soit le nombre de points conditionnants considérés.

Un autre avantage de cette approche est qu'elle simplifie la simulation conditionnelle des fonctions actives dans le domaine D. En ce qui concerne le processus de Schlather, la loi conditionnelle obtenue par Dombry et Eyi-Minko (2013) peut s'interpréter de la façon suivante

$$Z_i^{(a)}(s_D) \,|\, Z_i^{(a)}(s_{1:k}) = v_{i,i:k} \quad \overset{\mathcal{L}}{\equiv} \quad Z_i^{(a*)}(s_D) + \kappa\big(Y(s_D) - Y^*(s_D)\big),$$

où $Z_i^{(a*)}(s_D)$ et $Y^*(s_D)$ sont les krigeages simples respectifs de $Z_i^{(a)}(s_D)$ sur les données $Z_i^{(a)}(s_{1:k})$, et d'un champ gaussien simulé $Y(s_D)$ sur ses valeurs $Y(s_{1:k})$ simulées aux points de données. La fonction de covariance utilisée pour ces deux krigeages n'est autre que C. De son coté, la constante κ vaut $\sqrt{v_{i,1:k}^t \Sigma^{-1} v_{i,1:k}/X}$, Σ étant la matrice de covariance induite par C aux points de données, et X est une variable suivant une loi du χ^2 à $k+1$ degrés de liberté.

4 Illustration

Nous reprenons les paramètres du processus de Schlather introduit en section 1. Un jeu de données de 100 points a été prélevé au hasard dans la simulation de la Figure 1 et utilisé pour générer les quatre simulations conditionnelles de la figure 3. Les maxima sont assez fluctuants (11.44, 9.35, 7.54 et 10.66).

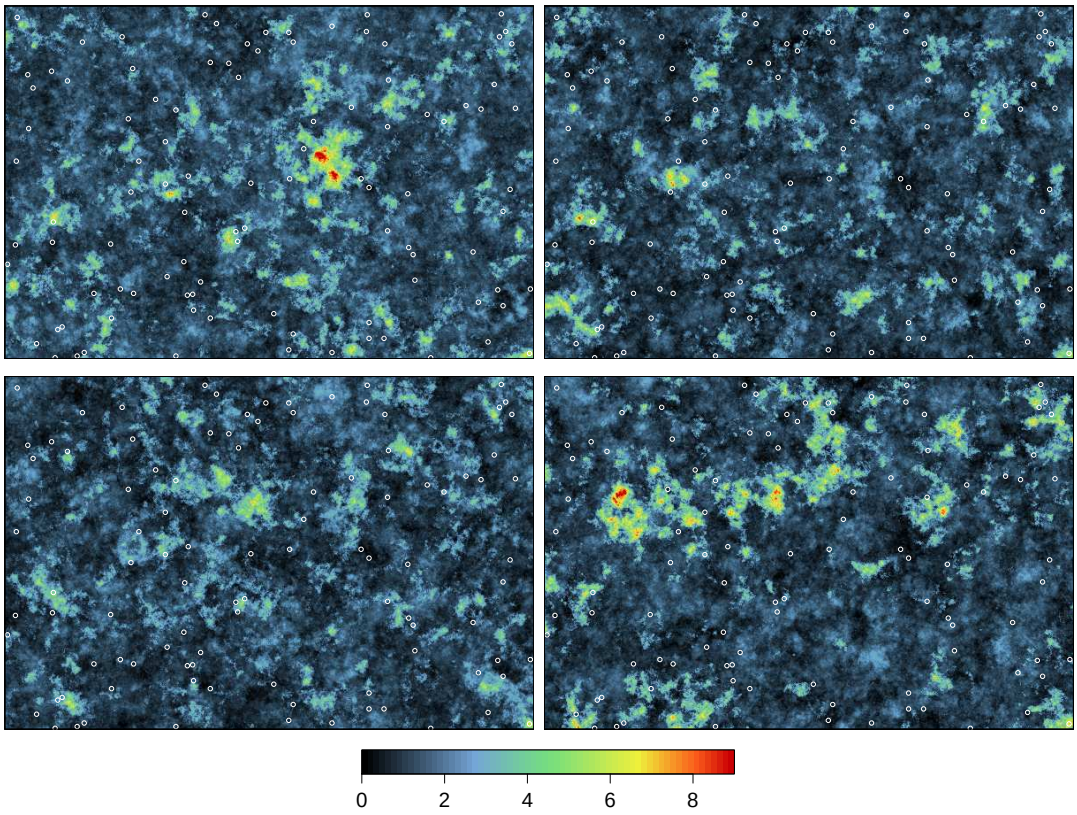

Figure 3: Quatre simulations conditionnelles d'un processus de Schlather

Remerciement: Ce travail a été financé par l'ANR Mc Sim.

Bibliographie

[1] Dombry, C. et Eyi-Minko, F. (2013), Regular conditional distributions of continuous max-infinitely divisible random fields, Electron. J. Probab., 7, 1-21.
[2] Dombry, C., Eyi-Minko, F. et Ribatet, M. (2012), Conditional simulation of max-stable processes, Biometrika, 20, 1-23.
[3] Geweke, J. (1991), Efficient simulation for the multivariate normal and Student-t distributions subject to linear constraints, Comp. Sci. and Stat., 571-578.
[4] Orlov, M. (2002), Efficient generation of set partitions, http://www.cs.bgu.ac.il/ orlovm-/papers/partitions.pdf.