Université d'Angers

Master Mathématiques et Applications

M1 DATA SCIENCE

Année académique 2022-2023

Travail Encadré de Recherche

Krigeage

Etudiants :
Paul-Antoine RICHARD
Clément TERNAT

Tuteur Enseignant :
Frédéric PROIA

23 mai 2023

Engagement de non plagiat

Nous, soussignés RICHARD Paul-Antoine et TERNAT Clément déclaront être pleinement conscients que le plagiat de documents ou d'une partie d'un document publiés sur toutes formes de support, y compris l'internet, constitue une violation des droits d'auteur ainsi qu'une fraude caractérisée. En conséquence, nous nous engageons à citer toutes les sources que nous avons utilisées pour écrire ce rapport.

Signature : RICHARD Paul-Antoine , TERNAT Clément

Table des matières

Chapitre 0

Introduction

Le krigeage est une méthode d'interpolation spatiale stochastique qui permet de prévoir la valeur d'un phénomène en un site non échantillonné par une combinaison linéaire des observations en des sites voisins. Il s'agit de la méthode la plus répandue et la plus simple à manoeuvrer pour ce qui concerne l'interpolation spatiale. Il est utilisé par exemple pour représenter des cartes météorologiques ou alors la concentration en minerais d'un sol entre autres. Cette méthode a l'avantage d'être assez visuelle et est donc par conséquent compréhensible assez facilement.

L'interpolation se déploie sur une certaine région étudiée que l'on appelere 'champ', notée D dans ce rapport. La variable que l'on cherchere à étudier dans le champ D est nommée 'variable régionalisée', on l'écrira $Z(s)$, $\forall Z \in D$, s étant la position dans l'espace de ce point. Notre objectif via le krigeage est de trouver la valeur d'un phénomène en un point que l'on nomme s_0. Pour trouver $Z(s_0)$ on utilise alors une combinaison linéaire de toutes les valeurs observées voisines $Z(s_1)$ à $Z(s_n)$ d'après les principes du krigeage.

Notre projet est découpé en 4 chapitres, le premier porte sur les différentes méthodes d'interpolation spatiales et le second porte sur l'analyse variographique. Ces deux premières parties nous permettent d'expliquer ensuite deux méthodes de krigeage dans un troisième chapitre. Enfin, dans le dernier chapitre nous avons mis en place plusieurs applications des méthodes de krigeage.

Chapitre 1

Méthodes d'interpolation spatiale

Dans ce premier chapitre, nous allons pouvoir présenter et expliciter les deux principaux groupes de méthodes d'interpolation spatiale, les méthodes déterministes et les méthodes stochastiques. Etant donné que la méthode qui nous intéresse pour ce projet est le Krigeage, nous ne rentrerons pas dans le détail des autres méthodes.

1.1 Méthodes d'interpolation déterministes

Les 3 principales méthodes d'interpolation déterministes sont les méthodes barycentriques, les méthodes d'interpolation par partitionnement de l'espace et les splines. Le terme 'déterministe' signifie ici que ce sont des méthodes qui ne prennent pas en compte d'aléatoire.

1.1.1 Méthodes barycentriques

Les méthodes barycentriques sont certainement les méthodes les plus intuitives et les plus simples à expliquer. En effet, leur objectif est de prévoir la variable régionalisée s_0 par une moyenne pondérée des valeurs régionalisées observées :

$$\hat{z}(s_0) = \Sigma_{i=1}^{n} \lambda_i z(s_i) \quad avec \quad \Sigma_{i=1}^{n} = 1 \tag{1.1}$$

On notifiera que la somme des poids λ_i est égale à 1 afin de ne pas déformer la réalité. Ces poids dépendent donc de la distance euclidienne entre le site à prédire s_0 et le site d'observation s_i. On a donc $|s_i - s_0|$, construit de manière à ce que les points des sites les plus proches aient une influence forte dans l'interpolation, alors qu'à l'inverse les points les plus éloignés auront une influence très faible voire nulle. C'est la raison pour laquelle une notion de voisinage 'V' est mise en place. Cette notion est très importante dans le krigeage, elle nous permet de sélectionner uniquement les points les plus proches dans notre interprétation de l'interpolation spatiale. Il faut noter que dans ce rapport, étant donné que nos échantillons sont de petites tailles dans les applications, le voisinage sera composé de l'ensemble des points.

L'exemple le plus populaire de méthode barycentrique est la méthode de l'inverse de la distance à une certaine puissance d. On aura alors pour cette méthode une prévision en s_0 de la forme suivante :

$$\hat{z}(s_0) = \Sigma_{i \in V(s_0)} \frac{1/|s_i - s_0|^d}{\Sigma_{i \in V(s_0)} 1/|s_i - s_0|^d} z(s_i) \quad , \quad d > 0 \tag{1.2}$$

1.1.2 Méthodes d'interpolation par partitionnement de l'espace et Splines

Ces deux méthodes étant plus complexes et peu importantes pour la compréhension du krigeage, nous avons fait le choix de ne pas les approfondir mais de simplement expliquer leurs fonctionnements.

Tout, d'abord, les méthodes d'interpolation par partitionnement de l'espace tiennent leur nom du fait qu'elles divisent et découpent l'espace en sous-parties. On peut donc découper le champ D de différentes manières, que ce soit par polygones ou par triangles. Les méthodes les plus couramment utilisées sont les 'polygones de Thiessen', le 'diagramme de Voronoi' ou encore la 'mosaïque de Dirichlet"'.

Via ces méthodes nous pouvons ainsi calculer la valeur d'un s_0 comme une moyenne des observations s_i appartenant au même polygone créé à partir des observations connues. D'autres méthodes auront elles comme objectif de créer de nouveaux polygones avec s_0.

Pour ce qui concerne les Splines, il s'agit d'une méthode d'interpolation bien plus complexe. Il ne s'agit pas d'une méthode qualifiée de 'point par point' comme peut l'être la méthode barycentrique par exemple mais plutôt d'une méthode dite soit 'de lissage' ou alors 'd'interpolation contrainte'. La différence entre les deux types de splines vient du fait que avec la méthode des splines de lissage, plutôt que de passer en tous points s_i, on tracera des courbes qui passent seulement à proximité de ces points, contrairement aux méthodes d'interpolations contraintes, qui comme leurs noms l'indiquent, doivent passer en tous points.

1.2 Methodes d'interpolation stochastiques

Les 3 principales méthodes d'interpolation stochastiques sont composées des méthodes classiques de régression, les régressions classiques et locales, ainsi que de la méthode du krigeage. Il est important de noter que ce qui les différencie des méthodes déterministes est le fait que ces méthodes sont dites 'stochastiques'. Cela signifie qu'elles ont une partie aléatoire, c'est à dire qu'une partie de l'analyse variographique n'est pas fixée.

1.2.1 Régressions classique et ordinaire

La régression classique permet de réaliser une interpolation en ajustant une surface aux valeurs observées. Elle suppose que la variable régionalisée étudiée est une fonction aléatoire s'écrivant : $Z(s) = \mu(s) + \varepsilon(s), s \in D$, avec $\mu(.)$ la structure déterministe et $\varepsilon(.)$ la structure aléatoire normale d'espérance nulle, de variance homogène et qui ne présente pas de structure de dépendance spatiale. $\mu(.)$ peut prendre différentes formes mais celle qui est la plus utilisée est un polynôme de degré d (généralement inférieur ou égal à 3), tel que :

$$\mu(s) = \mu(x, y) = \Sigma_{l+k \leq d}\, \beta_{lk} x^l y^k \tag{1.3}$$

Les coefficients β_{lk} peuvent être ajustés de différentes manières, par exemple, par la méthode du maximum de vraisemblance ou par la méthode des moindres carrés, cette dernière ajuste β_{lk} en minimisant :

$$\Sigma_{i=1}^{n} [\hat{z}(s_i - z(s_i)]^2 \tag{1.4}$$

La régression locale est une extension de la régression classique, elle lui ressemble à l'exception que les coefficients de $\mu(.)$ sont estimés par une méthode locale d'estimation comme par la

méthode du maximum de vraissemblance pondérée ou celle de la méthode des moindres carrés pondérés. Nous aurions donc par la méthode des moindres carrés pondérés, la surface estimée en un point s_0 qui minimise :

$$\Sigma_{i=1}^{n} \lambda_i(s_0)[\hat{z}(s_i|_{s_0} - z(s_i)]^2 \tag{1.5}$$

où les poids du modèle aux points s_i prennent la forme :

$$\lambda_i(s_0) = K(\frac{|s_0 - s_i|}{h(s_0)}) \tag{1.6}$$

avec $K(.)$ une fonction de poids.

1.2.2 Le Krigeage

Le krigeage, contrairement aux méthodes déterministes et aux méthodes stochastiques que nous avons pu voir précedemment, est la première méthode d'interpolation spatiale à prendre en considération la structure de dépendance spatiale des données. Elle tient son nom de l'ingénieur minier sud-africain Krige. Le concept du krigeage est de prévoir la valeur de la variable généralisée étudiée en un site non-échantilloné s_0 par une combinaison linéaire des données adjacentes :

$$\hat{z}(s_0) = a + \Sigma_{V_i} \lambda_i z(s_i) \quad , \ i = 1, ..., n \tag{1.7}$$

Les poids λ_i sont choisis de manière à obtenir une prévision non biaisée et de variance minimale. Ils dépendent donc de la localisation de l'observation qui leur est associée et de la structure de la dépendance spatiale générale. En effet un point plus éloigné provoquera une dépendance moindre et donc un poids plus faible.

Le modèle du krigeage est donc le suivant :

$$Z(s) = \mu(s) + \delta(s), \ \forall s \in D \tag{1.8}$$

On peut alors remarquer qu'il a la même forme que le modèle de régression classique, mais que cette fois les erreurs sont supposées dépendantes spatialement. Dans ce modèle, $\mu(.)$ est la structure déterministe pour l'espérance de $Z(s)$, et $\delta(.)$ est une fonction aléatoire dont nous étudierons la stationnarité plus tard dans cette étude.

Etant donné qu'il existe plusieurs formes de krigeage, il est important de définir la forme de $\mu(s)$, car à chaque forme son modèle associé. Il existe donc trois types de krigeage classique :

- Le krigeage simple : $\mu(s) = m$ est une constante connue
- Le krigeage ordinaire : $\mu(s) = \mu$ est une constante inconnue
- Le krigeage universel : $\Sigma_i^p \alpha_s f_i(s)$ est une combinaison linéaire de fonctions de base f_i.

Il est important d'ajouter qu'il existe d'autres méthodes de krigeage comme le modèle bayésien. Dans ce rapport, nous ne nous intéresserons qu'aux deux premières méthodes, c'est-à-dire au krigeage simple et au krigeage universel. Dans le chapitre suivant, nous nous focaliserons sur la partie stochastique du krigeage, afin de la définir et donc de mieux la comprendre.

Chapitre 2

Analyse Variographique

L'objectif de ce second chapitre est de présenter plusieurs méthodes d'interpolation spatiale, autrement dit, de définir le $\delta(.)$ du chapitre précédent, c'est à dire la partie aléatoire du krigeage. L'analyse variographique étant l'un des points centraux du krigeage, nous allons donc l'introduire dans un premier temps, puis expliquer le concept de stationnarité de ce facteur, puis enfin nous parlerons du semi-variogramme et de ses propriétés.

2.1 Stationnarité

Dans l'étude de l'analyse variographique, il est important de poser des conditions. C'est pourquoi une hypothèse de stationnarité est appliquée à la partie aléatoire, en effet elle permet de limiter les paramètres du $\delta(s)$. Cette limitation vient du fait que étant donné qu'il s'agit d'une fonction aléatoire, on ne veut pas qu'il y ait de trop grandes variations au sein de l'aléa et donc de la variance.

2.1.1 Stationnarité de second ordre

La stationnarité de second ordre impose une invariance de la moyenne et de la covariance par translation. Ses conditions sont alors :

— $\mathrm{E}(\delta(s)) = m = 0 \ , \forall s \in D$

— $\mathrm{Cov}(\delta(s), \delta(s+h)) = \mathrm{E}[(Z(s+h) - m)(Z(s) - m)] = C(h) \ , \ \forall s, \ s+h \in D$

Le $C(h)$ obtenu est la fonction que l'on appelle 'covariogramme'. Cette fonction dépend de h, le vecteur de translation entre les deux points s et $s+h$. On peut notifier que sous ces conditions, l'espérance de $\delta(s)$ existe et est supposée nulle, et que la covariance de $\delta(s)$ existe elle aussi.

On peut ajouter que la stationnarité de second ordre est une hypothèse de stationnarité forte qui implique la stationnarité intrinsèque. Cette affirmation sera justifiée par la suite.

2.1.2 Stationnarité intrinsèque

Les conditions imposées à la stationnarité intrinsèque sont les suivantes :

— $\mathrm{E}(\delta(s+h)) = 0 \ , \forall s \in D$

— $\mathrm{Var}(\delta(s+h) - \delta(s)) = 2\gamma(h) \ , \forall s, \ s+h \in D$

Cette fois si nous obtenons la fonction $2\gamma(h)$ qui est appelée 'variogramme'. On remarquera que contrairement à la stationnarité de second ordre, l'espérance de $\delta(s)$ sous ces conditions n'existe pas. Sa variance elle existe bien, et ne dépend que de h.

Nous allons montrer que la stationnarité de second ordre implique la stationnarité intrinsèque. Pour cela nous allons relier le covariogramme et le variogramme :

$$2\gamma(h) = 2(C(0) - C(h)) \tag{2.1}$$

L'existence du covariogramme implique donc l'existence du variogramme. Par contre, l'implication inverse n'est vraie que sous certaines conditions. C'est la raison pour laquelle nous prioriserons l'utilisation de la stationnarité intrinsèque par la suite, ce qui est d'ailleurs le cas dans la majorité des rapports et ouvrages sur ce sujet. Ainsi nous privilégierons l'étude du variogramme et même plus précisémment du semi-variogramme $\gamma(.)$

2.2 Caractéristiques du semi-variogramme

Le semi-variogramme s'écrit de la manière suivante :

$$\gamma(h) = \frac{1}{2}\text{Var}(\delta(s+h) - \delta(s)) \tag{2.2}$$

où $\delta(.)$ est une fonction aléatoire d'espérance égale à 0.

2.2.1 Isotropie

Pour l'intégralité de ce rapport nous allons considérer que le covariogramme étudié est isotrope. Cela signifie que que la norme de h ne dépend pas de la direction d'un point par rapport à un autre mais seulement de la distance entre ces points. Si nous n'appliquons pas cette hypothèse au covariogramme, il serait donc anisotrope, et ces caractéristiques ne seraient pas uniformes en fonction de la direction. En effet, plutôt que de nous concentrer sur la simple distance h nous aurions à nous préoccuper d'un vecteur avec deux coordonnées.

Dans notre cas donc, le semi-variogramme ne dépend que de la norme de h, tel que :

$$\gamma(h) = \gamma(\parallel h \parallel) \tag{2.3}$$

2.2.2 Effet pépite

L'effet pépite est caractérisé par un saut à l'origine du semi-variogramme. Si $lim_{\|h\|\to 0^+}\ \gamma(h) = c_0 > 0$, alors c_0 est appelé effet pépite. Cela signifie que la variance est plus élevée et donc que les valeurs régionaliées voisines ont une faible ressemblance. Il peut aussi très bien être négligeable dans un champ ou la variablilité est relativement homogène.

2.2.3 Portée et Palier

Lorsqu'on étudie l'évolution de la norme de h en fonction du semi-variogramme, on étudie les variations de $\gamma(\parallel h \parallel)$. Lors de sa croissance, le semi-variogramme peut atteindre ou non un plateau. Lorsqu'un plateau est atteint, cela signifie qu'il n'y a plus de dépendance spatiale entre les données. Cette distance est notée la portée. Dans le cas où le palier est asymptotique, la portée est donc définie par la distance à laquelle le covariogramme atteint 95% de la valeur de son palier. Ci-après une image pour illustrer la représentation du semi-variogramme :

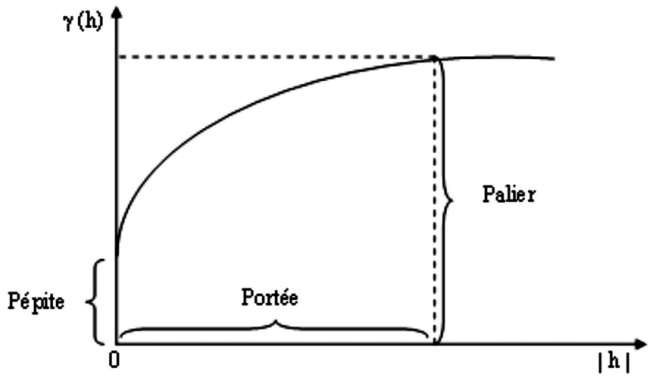


FIGURE 2.1 – Représentation des caractéristiques d'un semi-variogramme

2.2.4 Estimation

Dans cette sous-partie nous allons chercher une estimation expérimentale du semi-variogramme tel que, en développant :

$$\gamma(h) = \frac{1}{2}\mathrm{Var}(\delta(s) - \delta(s+h)) \tag{2.4}$$
$$= \frac{1}{2}\mathrm{Var}[(Z(s) - \mu(s)) - (Z(s+h) + \mu(s+h))] \tag{2.5}$$
$$= \frac{1}{2}\mathrm{Var}[(Z(s) - Z(s+h)] \tag{2.6}$$
$$= \frac{1}{2}\mathrm{E}[(Z(s) - Z(s+h))^2] - \frac{1}{2}\mathrm{E}[Z(s) - Z(s+h)]^2 \tag{2.7}$$
$$= \frac{1}{2}\mathrm{E}[(Z(s) - Z(s+h))^2] - \frac{1}{2}(\mu(s) - \mu(s+h))^2 \tag{2.8}$$

Etant donné que $\mu(.)$ est une fonction constante, le deuxième terme s'annule et le semi-variogramme est estimable directement à partir des observations $z(s_i)$. L'estimateur expérimental du semi-variogramme le plus commun est celui des moments, il est de la forme suivante :

$$\hat{\gamma}(h) = \frac{1}{2N(h)}\Sigma_{i=1}^{n}[z(s_i) - z(s_i+h)]^2 \tag{2.9}$$

Avec :

— $\hat{\gamma}(h)$ est l'estimation du semi-variogramme en fonction de la distance h

— $N(h)$ est le nombre de paires distinctes de distances de l'ensemble

Ce modèle n'est que très peu utilisé étant donné que sa mise en place est complexe à cause de $N(h)$. En effet il est très rare que lors d'une utilisation réelle d'une méthode de krigeage les points soient tous parfaitement espacés d'une même distance. C'est la raison pour laquelle nous allons voir différents modèles qualifiés de 'théoriques' par la suite, ces modèles étant plus viable dans des cas concrets.

2.2.5 Modélisation

Comme présenté à la fin de la section précédente, nous allons voir ici plusieurs estimations théoriques du semi-variogramme. On peut ajouter qu'il existe beaucoup de types de modèles différents, par exemple des modèles avec ou sans palier, ou alors des modèles avec une portée asymptotique ou non. Nous allons regarder en détail plusieurs de ces estimations puis nous tracerons leurs graphes :

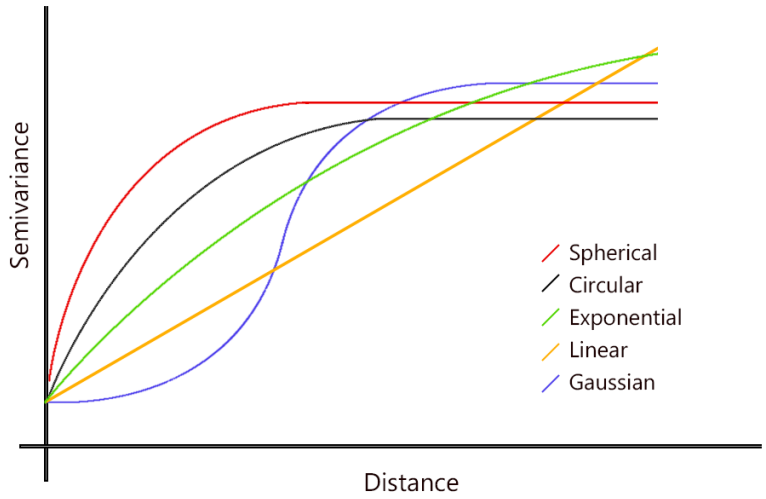


FIGURE 2.2 – Représentation de plusieurs semi-variogrammes

Modèle linéaire avec palier : Ce modèle est d'effet de pépite c_0, de palier $c_0 + c$ et de portée a :

$$\gamma(r) = \begin{cases} c_0 + \dfrac{c}{a} r & , \quad \text{pour } 0 \leq r \leq a \\ c_0 + c & , \quad \text{pour } r > a \end{cases}$$

Modèle gaussien : Ce modèle est d'effet de pépite c_0, de palier $c_0 + c$ et de portée pratique égale à $a\sqrt{3}$:

$$\gamma(r) = c_0 + c(1 - exp(-\frac{r^2}{a^2})) \quad pour\ r \geq 0 \tag{2.10}$$

Modèle exponentiel : Ce modèle est d'effet de pépite c_0, de palier $c_0 + c$ et de portée pratique égale à $3a$:

$$\gamma(r) = c_0 + c(1 - exp(-\frac{r}{a})) \quad pour\ r \geq 0 \tag{2.11}$$

Modèle sphérique : Ce modèle est d'effet de pépite c_0, de palier $c_0 + c$ et de portée a :

$$\gamma(r) = \begin{cases} c_0 + c\left(\dfrac{3}{2}\dfrac{r}{a} - \dfrac{1}{2}\dfrac{r^3}{a^3}\right) & , \quad \text{pour } 0 \leq r \leq a \\ c_0 + c & , \quad \text{pour } r > a \end{cases}$$

On peut alors observer les courbes des différents semi-variogrammes :
Variogram-models

Dans ce chapitre, nous avons donc pu définir un nouvel outils d'analyse variographique, le semi-variogramme, qui nous permet de représenter une dépendance spatiale. Nous allons donc pouvoir l'utiliser dans le prochain chapitre, qui porte sur deux méthodes de krigeage, le krigeage simple et le krigeage oridnaire.

Chapitre 3

Théorie du krigeage

Dans ce troisème chapitre, nous allons nous focaliser sur la partie mathématiques de l'obtention de la forme d'une prévision de deux types de krigeage, le krigeage simple dans un premier temps puis le krigeage ordinaire. On rappelle que le krigeage simple est un cas particulier du krigeage oridnaire où l'espérance est connue. Nous allons voir la démarche de résolution des équations du krigeage dans la première sous-partie, puis nous appliquerons cette méthode aux deux types de krigeage.

3.1 Méthode générale

La démarche du krigeage suit les 4 contraintes suivantes :

Contrainte de linéarité :
Il s'agit de la première contrainte, la prévision du krigeage doit prendre la forme d'une combinaison linéaire de données telle que :

$$\hat{Z}(s_0) = a + \Sigma_{i=1}^{n} \lambda_i Z(s_i) \tag{3.1}$$

Les poids λ_i et la constante a sont les inconnus du problème.

Contrainte d'autorisation :
Cette contrainte stipule que l'espérance et la variance de l'erreur de prévision existent, ce que nous avons vérifié dans le chapitre précédent, sous les hypothèses de stationnarité de second ordre. Cette contrainte n'intervient donc pas.

Contrainte de non-biais :
Comme son nom l'indique, il faut une absence de biais tel que : $\mathrm{E}[\hat{Z}(s_0) - Z(s_0)] = 0$

Contrainte d'optimalité :
Les poids λ_i et la constante a sont déterminés de facon à minimiser : $\mathrm{Var}[\hat{Z}(s_0) - Z(s_0)]$

Nous serons donc amenés à résoudre des équations, et pour faciliter leurs écritures sous forme matricielle nous utiliserons les notations suivantes :

- $Z = (Z(s_1), ..., Z(s_n))^t$
- $\Lambda = (\lambda_1, ..., \lambda_n)^t$
- $\Delta = (\delta_1, ..., \delta_n)^t$
- $\Sigma = \mathrm{Var}(Z)$ est la matrice de variances-covariances
- $c_0 = \mathrm{Cov}(Z, Z(s_0)) = (c_{0_1}, ..., c_{0_n})$

3.2 Krigeage simple

Nous allons appliquer ces contraintes au krigeage simple.

Contrainte de linéarité : La prévision $\hat{Z}(s_0)$ prend la forme de la combinaison linéaire suivante :

$$\hat{Z}(s_0) = a + \Sigma_{i=1}^{n} \lambda_i Z(s_i) = a + \Lambda^t Z$$

Contrainte d'autorisation : Comme expliqué précédemment, cette hypothèse n'intervient pas.

Contrainte de non-biais : Nous devons satisfaire l'égalité suivante :

$$\begin{aligned} \mathrm{E}[\hat{Z}(s_0) - Z(s_0)] = 0 \Leftrightarrow \mathrm{E}[a + \Sigma_{i=1}^{n} \lambda_i Z(s_i) - Z(s_0)] = 0 & \quad (3.2) \\ \Leftrightarrow \mathrm{E}[a] + \Lambda^t \mathrm{E}[Z] + \mathrm{E}[Z(s_0)] = 0 & \quad (3.3) \\ \Leftrightarrow a + \Lambda^t m 1_n - m = 0 & \quad (3.4) \end{aligned}$$

Cela signifie donc que :

$$a = m - \Lambda^t m 1_n = m(1 - \Lambda^t 1_n)$$

On remplace alors a dans l'expression de base et nous obtenons :

$$\hat{Z}(s_0) = m(1 - \Lambda^t 1_n) + \Lambda^t Z$$

Contrainte d'optimalité : Nous devons minimiser $\mathrm{Var}[\hat{Z}(s_0) - Z(s_0)]$ sous les trois contraintes précédentes :

$$\begin{aligned} \mathrm{Var}[\hat{Z}(s_0) - Z(s_0)] &= \mathrm{Var}[m + \Lambda^t(Z - m1) - Z(s_0)] & (3.5) \\ &= \mathrm{Var}[m + \Lambda^t \delta - m - \delta(s_0)] & (3.6) \\ &= \mathrm{Var}[\Lambda^t \delta] + \mathrm{Var}[\delta(s_0)] - 2\mathrm{Cov}[\Lambda^t \delta, \delta(s_0)] & (3.7) \\ &= \Lambda^t \mathrm{Var}[\delta]\Lambda + \mathrm{Var}[\delta(s_0)] - 2\Lambda^t \mathrm{Cov}[\delta, \delta(s_0)] & (3.8) \\ &= \Lambda^t \Sigma \Lambda + \sigma^2 - 2\Lambda^t c_0 & (3.9) \end{aligned}$$

Nous pouvons donc voir cette expression comme une fonction des poids λ_i. On calcule alors le gradient de cette fonction et nous obtenons :

$$\frac{\theta}{\theta \Lambda} f(\Lambda) = \frac{\theta}{\theta \Lambda}(\Lambda^t \Sigma \Lambda + \sigma^2 - 2\Lambda^t c_0) = 2\Sigma\Lambda - 2c_0$$

On cherche alors le minimum, pour cela on résout :

$$2\Sigma\Lambda - 2c_0 = 0 \Leftrightarrow \hat{\Lambda} = \Sigma^{-1} c_0$$

On peut donc estimer le krigeage simple :

$$\begin{aligned} \hat{Z}(s_0) &= m(1 - \Sigma^{-1} c_0 1_n) + \Sigma^{-1} c_0 Z & (3.10) \\ &= m + c_0^t \Sigma^{-1}(Z - m1_n)) & (3.11) \end{aligned}$$

La valeur minimale de la variance de l'erreur de prévision, que l'on peut aussi appeler 'variance de krigeage' est alors :

$$\begin{aligned} \sigma^2(s_0) &= \mathrm{Var}[\hat{Z}(s_0) - Z(s_0)] & (3.12) \\ &= c_0^t \Sigma^{-1} \Sigma \Sigma^{-1} = c_0 + \sigma^2 - 2c_0^t \Sigma^{-1} c_0 & (3.13) \end{aligned}$$

3.3 Krigeage ordinaire

Comme indiqué au début du chapitre, le krigeage ordinaire est une forme de krigeage simple généralisée car il prend en compte l'aspect stochatique de l'espérance. Nous allons développer ce krigeage sous l'hypothèse de stationnarité intrinsèque. On peut donc l'écrire de la manière suivante :

$$Z(s) = \mu + \delta(s) \ , \ \forall s \in D$$

avec μ une constante inconnue et $\delta(.)$ une fonction aléatoire d'espérance nulle et de stucture de dépendance connue. C'est la raison pour laquelle cette forme de krigeage est plus adaptée à des jeux de données plus grands. Elle est d'ailleurs la technique de krigeage la plus souvent utilisée. Nous allons donc pouvoir lui appliquer les quatres contraintes :

Contrainte de linéarité : La prévision $\hat{Z}(s_0)$ prend la forme de la combinaison linéaire suivante :

$$\hat{Z}(s_0) = a + \Sigma_{i=1}^{n} \lambda_i Z(s_i) = a + \Lambda^t Z$$

Contrainte d'autorisation : Comme expliqué précédemment, cette hypothèse n'intervient pas car la stationnarité de second ordre implique la stationnarité intrinsèque.

Contrainte de non-biais : Comme pour le krigeage simple, nous devons satisfaire l'égalité suivante :

$$\begin{aligned} \mathrm{E}[\hat{Z}(s_0) - Z(s_0)] = 0 \Leftrightarrow \mathrm{E}[a + \Sigma_{i=1}^{n} \lambda_i Z(s_i) - Z(s_0)] = 0 & \quad (3.14) \\ \Leftrightarrow a + \mu(\Sigma_{i=1}^{n} \lambda_i - 1) = 0 & \quad (3.15) \end{aligned}$$

Etant donné qu'une combinaison linéaire d'accroissements est équivalente à une combinaison linéaire de variables aléatoires avec des poids de somme nulle, cela signifie que : $\Sigma_{i=1}^{n} \lambda_i = 1$, sachant que les poids sont associés aux $\delta(s_i)$ et 1 est le poids associé à $\delta(s_0)$. Sous cette contrainte a est donc égal à 1. Donc :

$$\hat{Z}(s_0) = \Sigma_{i=1}^{n} \lambda_i Z(s_i) \quad avec \ \ \Sigma_{i=1}^{n} \lambda_i = 1$$

Contrainte d'optimalité : Comme pour le krigeage simple nous devons trouver les poids qui minimisent la valeur de l'erreur de prévision tel que :

$$\begin{aligned} \mathrm{Var}[\hat{Z}(s_0) - Z(s_0)] &= \mathrm{E}[(\hat{Z}(s_0) - Z(s_0))^2] & (3.16) \\ &= \mathrm{E}[(\Sigma_{i=1}^{n} \lambda_i Z(s_i) - Z(s_0))^2] & (3.17) \\ &= \mathrm{E}[(\Sigma_{i=1}^{n} \lambda_i \mu - \Sigma_{i=1}^{n} \lambda_i \delta(s_i) - \mu - \delta(s_0))^2] & (3.18) \\ &= \mathrm{E}[(\Sigma_{i=1}^{n} \lambda_i \delta(s_i) - \delta(s_0))^2] & (3.19) \\ &= \mathrm{E}[(\Sigma_{i=1}^{n} \lambda_i \delta(s_i))^2 - 2\delta(s_0)\Sigma_{i=1}^{n} \lambda_i \delta(s_i) + \delta(s_0))^2] & (3.20) \\ &= \mathrm{E}[\Sigma_{i=1}^{n}\Sigma_{j=1}^{n} \lambda_i \lambda_j \delta(s_i)\delta(s_j) - 2\delta(s_0)\Sigma_{i=1}^{n} \lambda_i \delta(s_i) + \delta(s_0))^2] & (3.21) \\ &= \mathrm{E}[\Sigma_{i=1}^{n}\Sigma_{j=1}^{n} \lambda_i \lambda_j \delta(s_i)\delta(s_j) - \Sigma_{i=1}^{n} \lambda_i \delta(s_i)^2 & (3.22) \\ &\quad + \Sigma_{i=1}^{n} \lambda_i \delta(s_i)^2 - 2\delta(s_0)\Sigma_{i=1}^{n} \lambda_i \delta(s_i) + \delta(s_0))^2] & (3.23) \end{aligned}$$

On décompose l'égalité en deux parties pour simplifier les calculs. Pour la première partie nous

avons donc :

$$\Sigma_{i=1}^{n}\Sigma_{j=1}^{n}\lambda_i\lambda_j\delta(s_i)\delta(s_j) - \Sigma_{i=1}^{n}\lambda_i\delta(s_i)^2 \tag{3.24}$$
$$= \Sigma_{i=1}^{n}\Sigma_{j=1}^{n}\lambda_i\lambda_j\delta(s_i)\delta(s_j) - \Sigma_{j=1}^{n}\lambda_j\Sigma_{i=1}^{n}\lambda_i\delta(s_i)^2 \tag{3.25}$$
$$= \Sigma_{i=1}^{n}\Sigma_{j=1}^{n}\lambda_i\lambda_j\delta(s_i)\delta(s_j) - \frac{1}{2}\Sigma_{i=1}^{n}\Sigma_{j=1}^{n}\lambda_i\lambda_j\delta(s_i)^2 - \frac{1}{2}\Sigma_{i=1}^{n}\Sigma_{j=1}^{n}\lambda_i\lambda_j\delta(s_j)^2 \tag{3.26}$$
$$= -\frac{1}{2}\Sigma_{i=1}^{n}\Sigma_{j=1}^{n}\lambda_i\lambda_j(\delta(s_i)^2 + \delta(s_j)^2 - 2\delta(s_i)\delta(s_j)) \tag{3.27}$$
$$= -\frac{1}{2}\Sigma_{i=1}^{n}\Sigma_{j=1}^{n}\lambda_i\lambda_j(\delta(s_i) - (\delta(s_j))^2 \tag{3.28}$$

Et pour la deuxième partie nous avons donc :

$$\Sigma_{i=1}^{n}\lambda_i\delta(s_i)^2 - 2\delta(s_0)\Sigma_{i=1}^{n}\lambda_i\delta(s_i) + \delta(s_0)^2 \tag{3.29}$$
$$= \Sigma_{i=1}^{n}\lambda_i\delta(s_i)^2 - 2\delta(s_0)\Sigma_{i=1}^{n}\lambda_i\delta(s_i) + \Sigma_{i=1}^{n}\lambda_i\delta(s_0)^2 \tag{3.30}$$
$$= \Sigma_{i=1}^{n}\lambda_i(\delta(s_i)^2 - 2\delta(s_0)\delta(s_i) + \delta(s_0)^2) \tag{3.31}$$
$$= \Sigma_{i=1}^{n}\lambda_i(\delta(s_i) - \delta(s_0))^2 \tag{3.32}$$

Nous pouvons alors lier les deux parties, ce qui nous donne :

$$\mathrm{Var}[\hat{Z}(s_0) - Z(s_0)] = \mathrm{E}[-\frac{1}{2}\Sigma_{i=1}^{n}\Sigma_{j=1}^{n}\lambda_i\lambda_j(\delta(s_i) - (\delta(s_j))^2] + \mathrm{E}[\Sigma_{i=1}^{n}\lambda_i(\delta(s_i) - \delta(s_0))^2] \tag{3.33}$$
$$= -\frac{1}{2}\Sigma_{i=1}^{n}\Sigma_{j=1}^{n}\lambda_i\lambda_j\mathrm{E}[(\delta(s_i) - (\delta(s_j))^2] + \Sigma_{i=1}^{n}\lambda_i\mathrm{E}[(\delta(s_i) - \delta(s_0))^2] \tag{3.34}$$
$$= -\Sigma_{i=1}^{n}\Sigma_{j=1}^{n}\lambda_i\lambda_j\frac{1}{2}\mathrm{Var}[(\delta(s_i) - (\delta(s_j))] + 2\Sigma_{i=1}^{n}\lambda_i\frac{1}{2}\mathrm{Var}[\delta(s_i) - \delta(s_0)] \tag{3.35}$$
$$= -\Sigma_{i=1}^{n}\Sigma_{j=1}^{n}\lambda_i\lambda_j\gamma(s_i - s_j) + 2\Sigma_{i=1}^{n}\lambda_i\gamma(s_0 - s_i) \tag{3.36}$$
$$= -\Lambda^t\Sigma\Lambda + 2\Lambda^t c_0 \tag{3.37}$$

Il faut alors minimiser cette expression sous la contrainte $\Lambda^t 1_{n_0} = 1$. Nous utiliserons un lagrangien que l'on notera l. On doit alors optimiser la fonction suivante :

$$f(\Lambda, l) = -\Lambda^t\Sigma\Lambda + 2\Lambda^t c_0 + 2l(\Lambda^t 1_{n_0} - 1)$$

On dérive $f(\Lambda, l)$ par rapport aux poids λ_i :

$$\frac{\theta}{\theta\Lambda}f(\Lambda, l) = -2\Sigma\Lambda + 2c_0 + 2l1_{n_0}$$

On a donc pour minimum :

$$\hat{\Lambda} = \Sigma^{-1}(c_0 + l1_{n_0})$$

Sous contrainte, on peut donc estimer le lagrangien :

$$\hat{l} = \frac{1 - 1_{n_0}^t\Sigma^{-1}c_0}{1_{n_0}^t\Sigma^{-1}1_{n_0}}$$

On peut alors estimer les poids en fonction de l'estimation du lagrangien précédent :

$$\hat{\Lambda} = \Sigma^{-1}(c_0 + \frac{1 - 1_{n_0}^t\Sigma^{-1}c_0}{1_{n_0}^t\Sigma^{-1}1_{n_0}}1_{n_0})$$

Matheron a prouvé qu'il s'agit bien de l'unique minimum de $f(\Lambda, l)$.
Ainsi, $Z(s_0)$ est donné par l'expression suivante :

$$\hat{Z}(s_0) = (c_0 + \frac{1 - 1_{n_0}^t \Sigma^{-1} c_0}{1_{n_0}^t \Sigma^{-1} 1_{n_0}} 1_{n_0}) \Sigma^{-1} Z$$

La variance du krigeage ordinaire s'écrit alors :

$$\sigma^2(s_0) = \mathrm{Var}[\hat{Z}(s_0) - Z(s_0)] = c_0^t \Sigma^{-1} c_0 - \frac{(1 - 1_{n_0}^t \Sigma^{-1} c_0)^2}{1_{n_0}^t \Sigma^{-1} 1_{n_0}}$$

Nous avons donc la formule du krigeage ordinaire.

Dans ce chapitre nous avons donc pu facilement obtenir les prévisions de deux méthodes de krigeage. Il faut préciser qu'il existe beaucoup d'autres méthodes de krigeage possible, telles que le krigeage universel que nous avons évoqué dans le chapitre précédent ou encore le krigeage bayésien. Dans le prochain chapitre nous allons donc commencer les mises en application du krigeage avec plusieurs exemples.

Chapitre 4

Application au krigeage ordinaire

Afin de mettre en application le krigeage ordinaire en Python, nous avons réalisé deux méthodes :

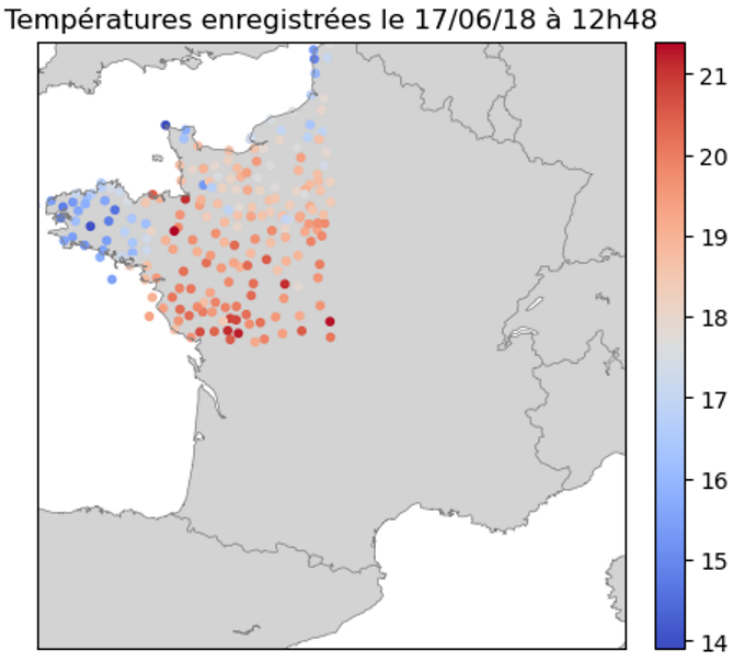
- Méthode 'PyKrige' - Module de Krigeage
- Méthode 'Maison' - Algorithme programmé à la main

Les données que nous utilisons sont des mesures de températures effectuées le 17/06/18 à 12h48 par Météo-France provenant de 211 stations météo situées dans le nord-ouest de la France.

	lat	lon	t
0	49.334	-0.431	18.7
1	49.180	-0.456	18.2
2	48.928	-0.149	19.0
3	48.795	-1.037	15.3
4	48.926	-0.693	17.6
...	...	...	...
206	46.321	0.410	19.9
207	47.035	0.098	20.3
208	46.412	0.841	19.4
209	48.526	1.993	18.4
210	49.108	1.831	17.4

211 rows × 3 columns

(a) Jeu de données



(b) Graphe des températures relevées

FIGURE 4.1 – Représentation des données utilisées

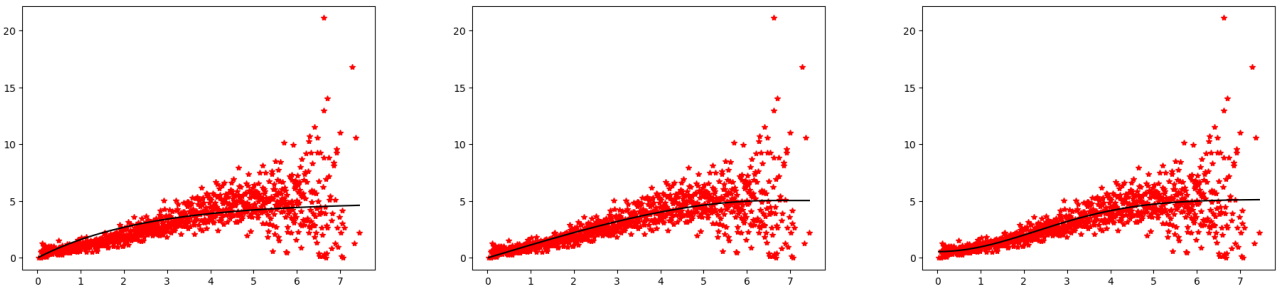
4.1 Méthode PyKrige

Grâce à la fonction OrdinaryKriging du module PyKrige, nous allons pouvoir obtenir les paramètres offrant les meilleures estimations de température selon un modèle de variogramme donné. L'avantage d'utiliser cette méthode est qu'elle est facile d'accès et qu'elle permet d'obtenir des estimations en un temps réduit. Nous rentrerons dans les détails de cette méthode

lorsque nous la programmerons dans la méthode 'Maison'.

Comparaison de différents modèles

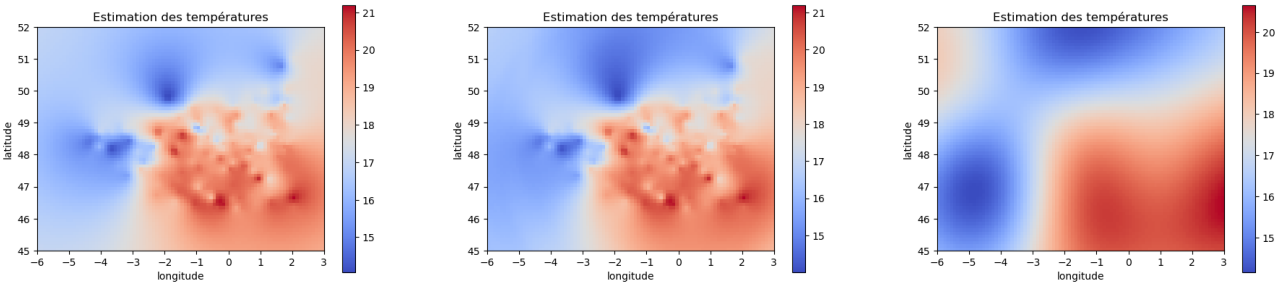
L'interpolation des points est différente selon le modèle de variogramme choisi. Ici, nous allons utiliser les modèles exponentiel, sphérique et gaussien.
On représente les semi-variogrammes associés :



(a) Modèle Exponentiel (b) Modèle Sphérique (c) Modèle Gaussien

FIGURE 4.2 – Comparaison des semi-variogrammes

On remarque que les semi-variogrammes exponentiel et sphérique se ressemblent, on peut espérer donc des interpolations qui seront très proches entre les deux modèles.

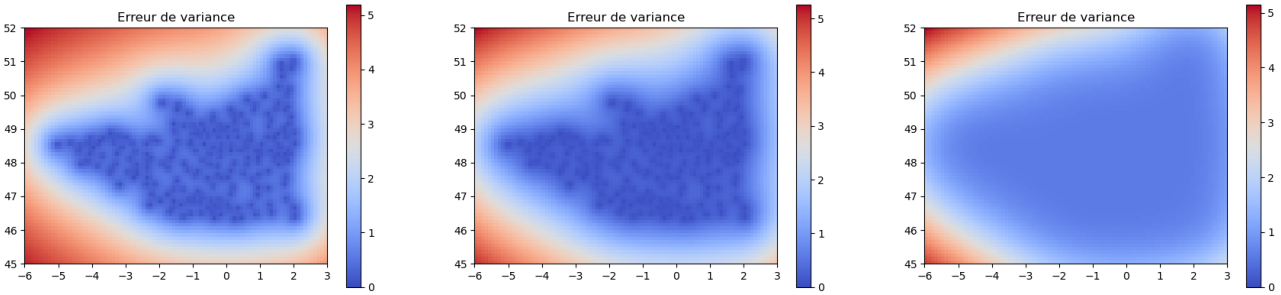


(a) Modèle Exponentiel (b) Modèle Sphérique (c) Modèle Gaussien

FIGURE 4.3 – Comparaison des interpolations

Comme attendu, les modèles exponentiel et sphérique sont très proches. Le modèle gaussien n'est pas pertinent dans notre cas, car il comporte de grandes zones de températures maximales et minimales.

On observe les graphes des erreurs de variance de chaque modèle :



(a) Modèle Exponentiel (b) Modèle Sphérique (c) Modèle Gaussien

FIGURE 4.4 – Comparaison des erreurs de variance

On constate que les graphes des modèles exponentiel et sphériques se ressemblent fortement. Plus on s'éloigne des points connus, plus la variance augmente, avec une faible variance près des points connus. Dans le cas du modèle gaussien, la variance autour des points connus est constante, confirmant une fois de plus que ce modèle n'est pas approprié dans notre cas.

Pour choisir le meilleur modèle, nous effectuons une validation-croisée sur chaque modèle, en gardant 60% des données pour l'entraînement des modèles, et 40% pour l'ensemble de test. On affiche le boxplot en sortie :

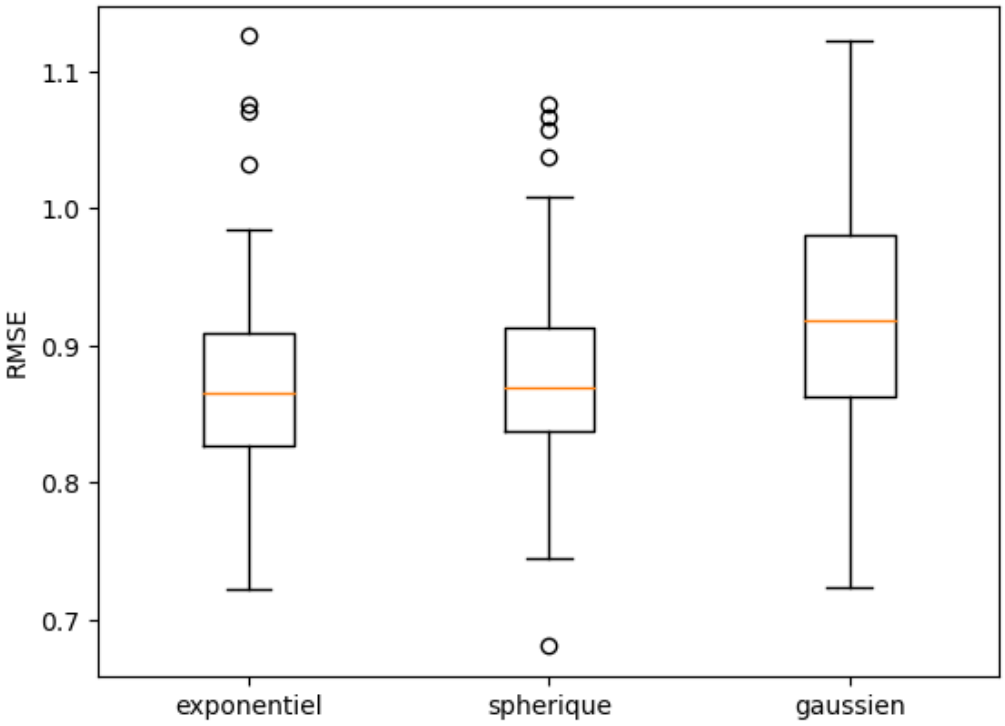


FIGURE 4.5 – Représentation des boxplots

On constate encore une fois que les modèles exponentiel et sphérique sont très ressemblants, avec chacun un RMSE qui varie moins que pour le modèle gaussien.

On peut donc conclure que les modèles exponentiel et sphérique ont quasiment la même performance dans notre cas. Le choix de l'un où l'autre a peu d'importance.

Dans la suite, on se concentrera donc uniquement sur les modèles exponentiel et sphériques.

4.2 Méthode Maison

L'objectif de cette méthode est de construire un algorithme de krigeage ordinaire en partant de zéro, dans le but d'obtenir des résultats qui se rapprochent de ceux des algorithmes du module PyKrige.

La difficulté principale va être d'estimer les meilleurs paramètres (pépite, palier, portée) de manière à, par validation-croisée, avoir le meilleur taux de prévision.

On peut décomposer cet algorithme en plusieurs étapes clés :

- Séparation aléatoire des données en un ensemble d'entraînement (60%) et un ensemble de test (40%).
- Calcul de la matrice des distances entre les points de l'ensemble d'entraînement.
- Application à la matrice de la formule du variogramme selon le modèle choisi.
- Pour chaque point de l'ensemble de test :
  - Calcul de la matrice des distances entre chaque point et le point à estimer.

— Application à la matrice de la formule du variogramme selon le modèle choisi.
— Calcul des poids de chaque point de l'ensemble d'entraînement à l'aide des matrices obtenues.
— Estimation de la température au point étudié.

– Validation Croisée : A l'aide des températures réelles et des températures estimées de l'ensemble de test, on calcule la Root Mean Square Error (RMSE).

On répète 100 fois cet algorithme afin de réduire l'effet aléatoire de la séparation des données sur le RMSE puis on calcule la moyenne des RMSE.

Enfin, on répète pour chaque combinaison pépite/palier/portée, et on garde celle ayant la plus faible moyenne des RMSE.

Maintenant que nous avons défini et compris le fonctionnement de notre algorithme, nous allons l'appliquer aux cas d'un variogramme exponentiel et d'un variogramme sphérique.

4.2.1 Modèle Exponentiel

Estimation des paramètres

Avec la méthode 'PyKrige', on obtenait les estimations des paramètres suivantes :

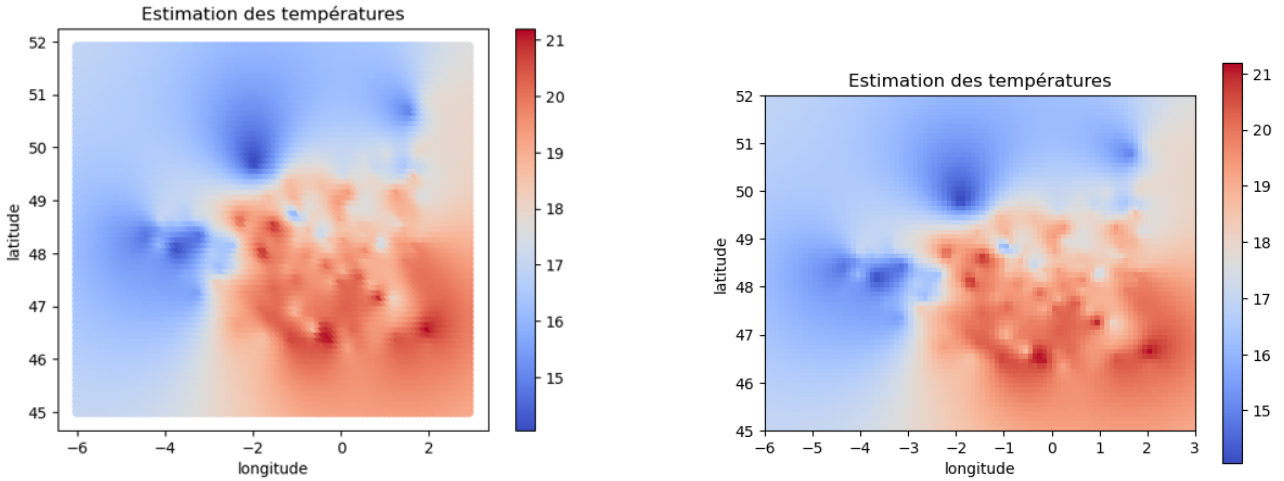
- pepite = 1.216e-14 / palier = 4.872 / portee = 7.441

A l'aide de notre algorithme, on estime les paramètres suivants :

- pepite = 0.01 / palier = 4.8 / portee = 7.4

On constate que notre estimation est très proche de celle de PyKrige, on peut désormais effectuer l'interpolation des points, connaissant les paramètres optimaux.

Comparaison des interpolations



(a) Interpolation par la méthode Maison

(b) Interpolation par la méthode PyKrige

FIGURE 4.6 – Comparaison des méthodes d'interpolation

Il est quasiment impossible de discerner notre interpolation de celle de PyKrige. Notre algorithme effectue donc un excellent travail dans le cas d'un modèle exponentiel.
On peut désormais représenter nos résultats sur une carte, à l'aide du module 'Basemap' de Python :

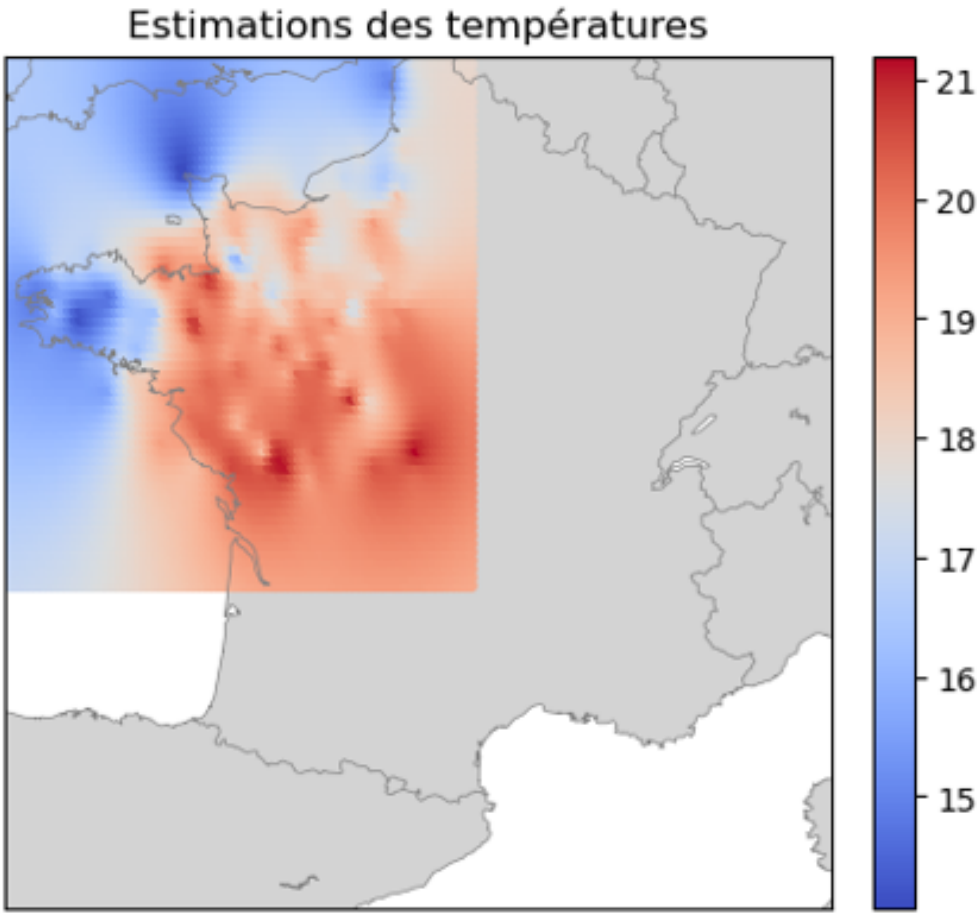


FIGURE 4.7 – Interpolation par la méthode Maison dans le cas d'un modèle exponentiel

4.2.2 Modèle Sphérique

Estimation des paramètres

Avec la méthode 'PyKrige', on obtenait :

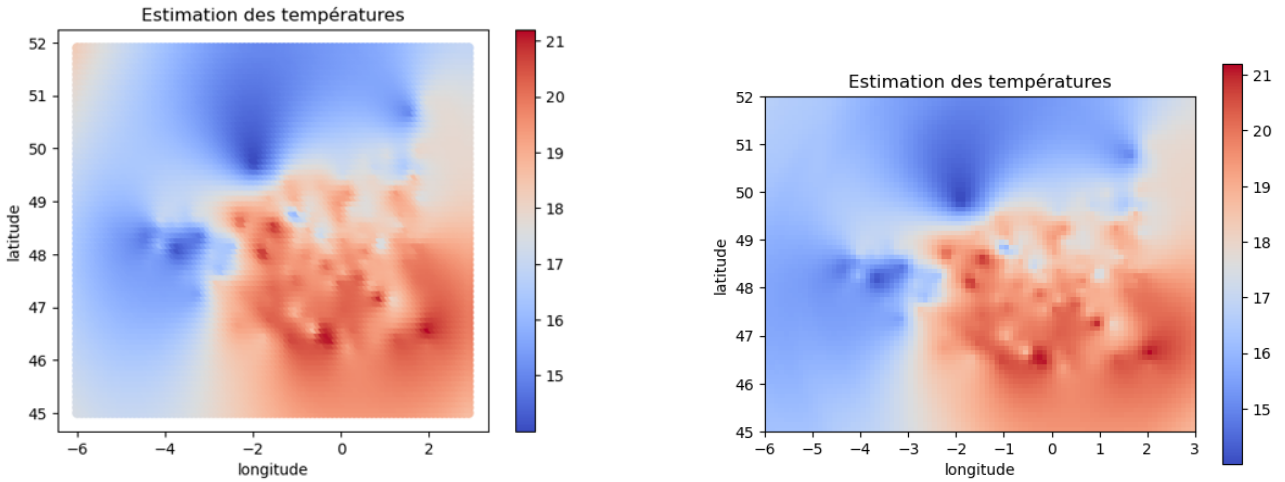
- pepite = 5.966e-12 / palier = 5.052 / portee = 6.593

Grâce à notre algorithme, on estime les paramètres suivants :

- pepite = 0.01 / palier = 5.1 / portee = 6.5

Une fois de plus, notre estimation des paramètres est très proche de celle de PyKrige, et on peut s'attendre à obtenir la même conclusion que pour le modèle exponentiel.

Comparaison des interpolations



(a) Interpolation par la méthode Maison

(b) Interpolation par la méthode PyKrige

FIGURE 4.8 – Comparaison des méthodes d'interpolation

Comme attendu, il est quasiment impossible de discerner notre interpolation de celle de PyKrige. On représente nos résultats sur une carte :

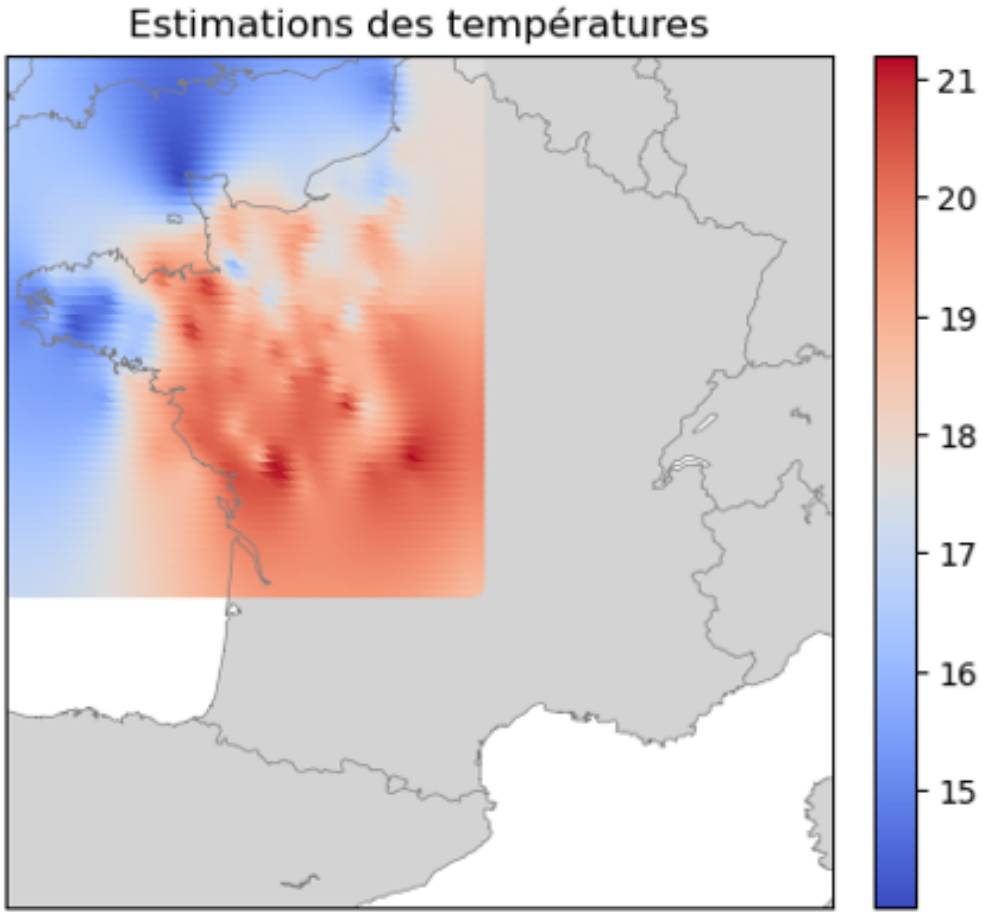


FIGURE 4.9 – Interpolation par la méthode Maison dans le cas d'un modèle sphérique

4.2.3 Choix du meilleur modèle

Afin de connaître le modèle à garder, on compare les RMSE obtenus lors des estimations des paramètres optimaux :

— Modèle exponentiel : RMSE = 0.8533813571393817

— Modèle sphérique : RMSE = 0.8661660213426388

Les deux RMSE étant quasiment égales, on arrive à la même conclusion que dans la méthode 'PyKrige', le choix de l'un ou l'autre des modèles a peu d'importance tant les prévisions obtenues sont proches.

Chapitre 5

Conclusion

L'objectif de ce rapport était de présenter de manière simple et efficace le fonctionnement du krigeage. Pour cela, nous avons suivi plusieurs étapes préliminaires visant à acquérir une meilleure compréhension des outils nécessaires tels que l'analyse variographique et le semi-variogramme. Ensuite, nous nous sommes concentrés sur l'explication théorique de deux méthodes de krigeage, ce qui nous a permis de mettre en place une application fonctionnelle dans la dernière partie.

En conclusion, le krigeage apparaît comme la méthode d'interpolation la plus optimale et largement utilisée de nos jours. Cependant, notre travail s'est limité aux méthodes du krigeage simple et du krigeage ordinaire. Il serait donc très intéressant d'étudier de nouvelles approches telles que le krigeage universel ou bayésien.

Annexes

Python

Importation des modules

```
from math import sqrt
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pykrige.kriging_tools as kt
from pykrige.ok import OrdinaryKriging
from sklearn.model_selection import train_test_split
from mpl_toolkits.basemap import Basemap
```

Création des listes à utiliser

```
data = pd.read_csv('C://Users//terna//Desktop//TER//datameteo.csv')
X = data['lon'].tolist()
Y = data['lat'].tolist()
Temp = data['t'].tolist()
Temp = [temp - 273.15 for temp in Temp]
```

Affichage des points sur une map avec les températures.

```
m = Basemap(llcrnrlon=-5, llcrnrlat=41, urcrnrlon=9, urcrnrlat=51, resolution='i',
↪  projection='merc')
m.drawcountries(linewidth=0.5, linestyle='solid', color='gray')
m.drawcoastlines(linewidth=0.5, linestyle='solid', color='gray')
m.fillcontinents(color='lightgray', lake_color='white')

x, y = m(X, Y)
sc = m.scatter(x, y, c=Temp, cmap='coolwarm', s=10)
cbar = m.colorbar(sc, location='right', pad='5%')
plt.title('Températures enregistrées le 17/06/18 à 12h48')
plt.show()
```

Fonction krigeage ordinaire de PyKrige. On peut paramétrer variogram-model en exponential, spherical ou gaussian selon le modèle choisi.

```
OK = OrdinaryKriging(
    X,
    Y,
    Temp,
    variogram_model='spherical',
    verbose=True,
    enable_plotting=True,
    nlags=1000,
)
```

Application du krigeage sur une grille de points.

```
gridx = np.arange(-6, 3, 0.1, dtype='float64')
gridy = np.arange(45, 52, 0.1, dtype='float64')
```

```
predi_val, predi_var = OK.execute("grid", gridx, gridy)
predi_val_plot = plt.imshow(predi_val, extent=(-6, 3, 45, 52), origin='lower',
↪  cmap='coolwarm')
colorbar = plt.colorbar(predi_val_plot)
plt.xlabel('longitude')
plt.ylabel('latitude')
plt.title('Estimation des températures')
plt.show()
```

Calcul des RMSE des modèles.

```
def pykrige_rmse(model,X,Y,Temp):
    L=[]
    for i in range(100):
        X_train, X_test, Y_train, Y_test, Temp_train, Temp_test = train_test_split(X, Y,
        ↪  Temp, test_size=0.4)
        OK = OrdinaryKriging(X_train, Y_train, Temp_train, variogram_model=f"{model}",
        ↪  nlags=1000)
        pred, ss = OK.execute('points', X_test, Y_test)
        rmse = calc_rmse(Temp_test, pred)
        L.append(rmse)
    return L
```

Création et affichage des boxplots des RMSE.

```
def plot_boxplots(L1, L2, L3):
    data = [L1, L2, L3]
    fig, ax = plt.subplots()
    ax.boxplot(data)
    ax.set_xticklabels(['exponentiel', 'spherique', 'gaussien'])
    ax.set_ylabel('RMSE')
    plt.show()
```

Calcul de la matrice de distance des points.

```
def mat_distance(X,Y):
    Lmat = []
    for x,y in zip(X,Y):
        L = []
        for i,j in zip(X,Y):
            dist = sqrt((x-i)2+(y-j)2)
            L.append(dist)
        Lmat.append(L)
    mat_dist = np.array(Lmat)
    return mat_dist
```

Calcul de la matrice des distances d'un point (i,j) aux points de (X,Y).

```
def mat_distance_point(X,Y,i,j):
    L = []
    for x,y in zip(X,Y):
        dist = sqrt((x-i)2+(y-j)2)
        L.append(dist)
    mat_dist_pt = np.array(L)
    return mat_dist_pt
```

On définit la formule du variogramme sphérique.

```
def spherical(pepite, palier, portee, h):
    return pepite + palier(1.5(h/portee)-0.5*((h/portee)3))
```

On définit la formule du variogramme exponentiel.

```
def exponential(pepite, palier, portee, h):
    return palier*(1.0-np.exp(-h/(portee/3.0)))+pepite
```

On applique le variogramme choisi à la matrice des distances de (X,Y). On peut changer spherical en exponential.

```
def sv_mat_dist(mat_dist):
    nv_mat = spherical(pepite, palier, portee, mat_dist)
    L1 = np.ones((1, nv_mat.shape[1]))
    nv_mat = np.vstack((nv_mat, L1))
    C1 = np.ones((nv_mat.shape[0], 1))
    nv_mat = np.hstack((nv_mat, C1))
    nv_mat[-1,-1]=0
    return nv_mat
```

On applique le variogramme choisi à la matrice des distances de (i,j) à (X,Y). On peut changer spherical en exponential.

```
def sv_mat_dist_pt(mat_dist_pt):
    nv_mat_pt = spherical(pepite, palier, portee, mat_dist_pt)
    nv_mat_pt = np.hstack((nv_mat_pt, 1))
    return nv_mat_pt
```

Fonction qui calcule les prévisions ainsi que le RMSE d'un ensemble de points en fonction d'un ensemble d'entraînement.

```
def KO(X_train, X_test, Y_train, Y_test, Temp_train, Temp_test, pepite, palier, portee):
    mat_dist = mat_distance(X_train,Y_train)
    mat_dist_vario = sv_mat_dist(mat_dist)
    TempPrevision = []
    for x,y in zip(X_test,Y_test):
        mat_dist_pt = mat_distance_point(X_train,Y_train,x,y)
        mat_dist_pt_vario = sv_mat_dist_pt(mat_dist_pt)
        mat_dist_vario_inv = np.linalg.inv(mat_dist_vario)
        poids = np.dot(mat_dist_vario_inv, mat_dist_pt_vario)
        poids = poids[:-1]
        prevision = np.dot(Temp_train, poids)
        TempPrevision.append(prevision)
    ErrQuadra = [(estimation - valeur_reelle)2 for estimation, valeur_reelle in
    ↪  zip(TempPrevision, Temp_test)]
    mse = sum(ErrQuadra) / len(ErrQuadra)
    rmse = sqrt(mse)
    return rmse, TempPrevision
```

Fonction qui calcule la prévision d'un point unique.

```
def KO_un_seul_point(X,Y,i,j, pepite, palier, portee):
    mat_dist = mat_distance(X,Y)
    mat_dist_vario = sv_mat_dist(mat_dist)
    mat_dist_pt = mat_distance_point(X,Y,i,j)
    mat_dist_pt_vario = sv_mat_dist_pt(mat_dist_pt)
    mat_dist_vario_inv = np.linalg.inv(mat_dist_vario)
    poids = np.dot(mat_dist_vario_inv, mat_dist_pt_vario)
    poids = poids[:-1]
    prevision = np.dot(Temp, poids)
    return prevision
```

Programme qui estime la meilleure combinaison des paramètres pépite/portée/palier. On peut changer les intervalles et les pas dans les boucles for.

```
precision = 100
for pepite in np.arange(0, 0.1, 0.01):
    for palier in np.arange(4.8, 5.3, 0.1):
        for portee in np.arange(6.2, 6.7, 0.1):
            Lprecision = []
            for i in range(0,500):
                X_train, X_test, Y_train, Y_test, Temp_train, Temp_test =
                ↪  train_test_split(X, Y, Temp, test_size=0.4)
                rmse, prev = KO(X_train, X_test, Y_train, Y_test, Temp_train, Temp_test,
                ↪  pepite, palier, portee)
                Lprecision.append(rmse)
            if np.mean(Lprecision)<precision:
                precision = np.mean(Lprecision)
                print(np.mean(Lprecision), pepite, palier, portee)
```

On calcule l'estimation de chaque point d'une grille.

```
Tempgrid = []
gridx = list(np.arange(-6, 3, 0.1, dtype='float64'))
gridy = list(np.arange(45, 52, 0.1, dtype='float64'))
Xgrid, Ygrid = np.meshgrid(gridx, gridy)

for xx, yy in np.nditer([Xgrid, Ygrid]):
    prevision = KO_un_seul_point(X,Y,xx,yy, pepite, palier, portee)
    Tempgrid.append(prevision)
```

On affiche la grille des températures sur un graphe.

```
plt.scatter(Xgrid, Ygrid, c=Tempgrid, cmap='coolwarm')
plt.colorbar()
plt.xlabel('longitude')
plt.ylabel('latitude')
plt.title('Estimation des températures')
plt.show()
```

Affichage de la grille des températures de chaque point sur une carte de la France.

```
m = Basemap(llcrnrlon=-5, llcrnrlat=41, urcrnrlon=9, urcrnrlat=51, resolution='i',
↪  projection='merc')
m.drawcountries(linewidth=0.5, linestyle='solid', color='gray')
m.drawcoastlines(linewidth=0.5, linestyle='solid', color='gray')
m.fillcontinents(color='lightgray', lake_color='white')
xmap, ymap = m(Xgrid, Ygrid)
sc = m.scatter(xmap, ymap, c=Tempgrid, cmap='coolwarm', s=10)
cbar = m.colorbar(sc, location='right', pad='5%')
plt.title('Estimations des températures')
plt.show()
```

Bibliographie

[1] S.BAILLERGEON, *Le Krigeage : revue de la théorie et application à l'interpolation spatiale de données de précipitations.* , 2005

[2] Documentation du module de krigeage : `https://geostat-framework.readthedocs.io/projects/pykrige/en/stable/index.html`

[3] Source des données météorologiques : `https://meteonet.umr-cnrm.fr/dataset/data/NW/ground_stations/`