université
angers

MASTER 2 DATA SCIENCES
2022 -2023
PROJET ANNUEL

KRIGEAGE

Auteurs :
Vincent LEVENT
Vahia RABARIJAONA

Encadrant :
Frédéric PROIA

25 février 2023

Table des matières

Chapitre 1

Introduction

Lorsqu'un processus stochastique est indexé par un ensemble S qui représente l'espace, on parle de statistique spatiale. Les données à référence spatiale prennent de plus en plus d'importance dans différents domaines. Les illustrations les plus simples concernent les problèmes tels que l'interpolation des températures ou des précipitations. Mais il peut aussi s'agir de problèmes touchant le domaine de la santé et de l'épidémiologie en passant par le domaine de l'économie ou du géomarketing. Les méthodes associées à la statistique spatiale se concentrent donc sur l'étude de données distribuées sur un espace géographique. L'interpolation spatiale est un des outils qui permet à l'aide de fonctions mathématiques de traiter ces données.

L'interpolation spatiale se déploie sur une région d'étude appelée « champ », qui définit un ensemble noté D. La variable que l'on étudie sur le champ est nommé « variable régionalisé », et se définit comme une fonction notée Z(s) appartenant à l'ensemble D, où s désigne la position dans l'espace. Étant donné qu'il est infaisable de collecter tous les points d'un champ D, le but de l'interpolation spatiale sera de prédire la valeur d'une variable régionalisée Z(s0) en un site où elle n'a pas été mesuré à partir des valeurs régionalisées observées Z(s1) à Z(sn). Quand l'interpolation se base uniquement sur la variable régionalisée, on parle de méthode déterministe, car aucune notion probabiliste n'intervient. Lorsque la variable régionalisée est vue comme une réalisation d'une fonction aléatoire, on parle de méthode stochastique. Il existe plusieurs méthodes d'interpolation spatiale stochastique comme les techniques de régression classique ou de régression locale. Quant à notre étude, il portera principalement sur une de ces méthodes, qui est le krigeage.

Le krigeage est une méthode d'interpolation spatiale stochastique qui prend en compte la structure de dépendance spatiale entre les différents points des données. Le nom de cette méthode vient de son précurseur, un ingénieur minier sud-africain qui s'appelle Krige (1951). Le terme « krigeage » a été formalisé par le français Matheron (1962), qui est un des pionniers ayant grandement contribué au développement de cette méthode.

Le projet sera structuré en plusieurs parties : nous introduirons d'abord les bases du krigeage ainsi que ses principes de bases. Nous aborderons ensuite l'étude du semi-variogramme, la fonction qui joue le plus grand rôle dans l'estimation du krigeage. Enfin, nous nous concentrerons sur deux types de krigeage : le krigeage simple et le krigeage ordinaire, avant de les appliquer à données réelles.

Chapitre 2

Présentation du krigeage

Le krigeage a pour but de prévoir la valeur de la variable régionalisée en un site non échantilloné s_0 par une combinaison linéaire des données connues. Le modèle du krigeage s'écrit :

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

où $\mu : S \to \mathbb{R}$ est une fonction déterministe donnant la valeur moyenne du processus, et $\delta : S \times S \to \mathbb{R}$ une fonction de covariance aléatoire stationnaire, d'espérance nulle et de structure de dépendance connue. Le modèle du krigeage a la même forme que le modèle de régression classique, mais les erreurs sont maintenant supposées dépendantes spatialement.

Pour la formulation du modèle, il faut spécifier la forme du tendance $\mu(s)$, car le type de krigeage effectué en dépend. Les différents types de krigeage classique sont :

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

Chapitre 3

Analyse variographique

L'analyse variographique constitue un point central dans le krigeage. Nous verrons par la suite que la valeur des estimateurs du krigeage dépend des observations et de la structure de la dépendance spatiale de la variable régionalisée qui est appréhendée par une fonction appelée semi-variogramme. Cette fonction estime les valeurs inconnues de la fonction aléatoire $\delta(.)$.

Sachant que $\delta(.)$ est une fonction aléatoire, il est nécessaire de limiter ses paramètres afin de faciliter l'inférence statistique en lui émettant une hypothèse de stationnarité. Cette hypothèse stipule qu'une fonction stationnaire se répète dans l'espace, permettant ainsi de faire une inférence à partir de la seule réalisation de la fonction $Z(.)$ dont on dispose.

L'objectif du chapitre est d'abord d'introduire et d'expliquer le concept de stationnarité avant de se concentrer sur le semi-variogramme, ainsi que ses différentes propriétés.

3.1 Stationnarité

Stationnarité de second ordre

La stationnarité du second ordre impose une invariance de la moyenne et de la covariance par translation. Elle requiert les conditions suivantes :

1. $E(\delta(s)) = m = 0 \ \forall \ s \in D$.
2. $Cov(\delta(s), \delta(s+h)) = \mathbb{E}[(Z(s+h) - m)(Z(s) - m)] = C(h)$.

La fonction $C(h)$ est la "covariogramme". La moyenne est constante et la covariance ne dépend seulement que du décalage spatial entre les points. En pratique, cette hypothèse s'avère forte, car la moyenne peut évidemment changer à différents endroits et la variance peut-être non-définie dans le cas où certaines phénomènes régionalisées présentent une dispersion infinie. Pour palier ces limites, on introduit la stationnarité intrinsèque.

Stationnarité intrinsèque

La stationnarité intrinsèque impose les conditions suivantes :

— $E(\delta(s+h) - \delta(s)) = 0 \forall s \in D$

— $Var(\delta(s+h) - \delta(s) = 2\gamma(h)$

La variance dépend uniquement de h, la fonction de variance $2\gamma(h)$ est appelé "variogramme". Néanmoins, on ne travaillera que sur la demie du variogramme : la "semi-variogramme".

La stationnarité du second ordre entraîne donc une stationnarité intrinsèque, contrairement à l'implication inverse. Il est alors possible de calculer un semi-variogramme d'une fonction aléatoire sans forcément passer par le calcul de sa covariogramme.

3.2 Semi-variogramme

Rappelons que le semi-variogramme s'écrit :

$$\gamma(h) = \frac{1}{2}\mathbb{V}\text{ar}(\delta(s+h) - \delta(s))$$

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

Le semi-variogramme $\gamma(.)$ possède ces propriétés :

$$\begin{cases} \gamma(h) = \gamma(-h) \ \forall \ h \in D \\ \gamma(0) = 0 \text{ et } \gamma(h) > 0 \text{ si } h \neq 0 \end{cases}$$

Le semi-variogramme est donc une fonction paire. Par la suite, il sera considéré commme isotrope, c'est à-dire qui ne dépend que de la norme de h, la distance entre les points s et $s+h$. On a donc :

$$\gamma(h) = \gamma(||h||)$$

3.2.1 Effet pépite

L'effet pépite se réfère à la variance non-détectée à une micro-échelle ou à une échelle plus petite que l'unité de mesure. Il se caractérise par un saut abrupt à l'origine, qui se dénote par une constante c_0 et qui s'obtient par :

$$\lim_{||h|| \to 0^+} \gamma(h) = c_0$$

L'effet pépite est donc un témoin d'une faible ressemblance entre des valeurs régionalisées voisines. S'il est important, cela signifie qu'il existe une grande variabilité à petite échelle, que les points les plus proches ne fournissent pas beaucoup d'informations sur les valeurs des autres points. S'il est négligeable, cela signifie que la variabilité est relativement homogène.

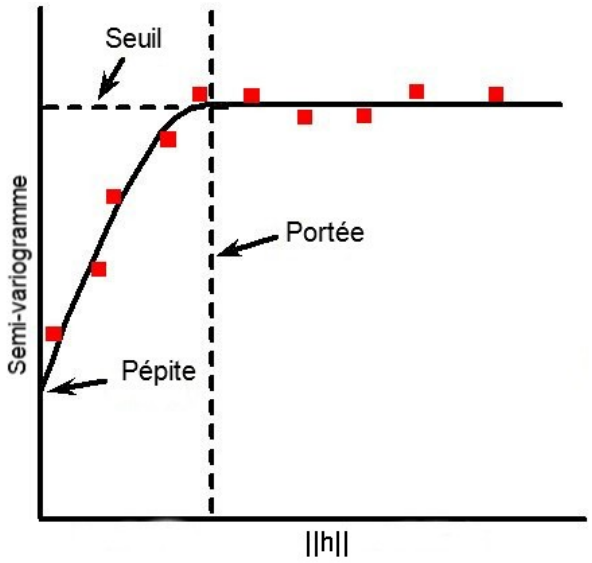
3.2.2 Portée et palier

Les modèles à seuil se concentrent sur le comportement de la covariogramme lorsque h augmente. Ils vérifient :

$$\lim_{||h|| \to \infty} \gamma(h) = \gamma_s$$

À une certaine valeur de $||h||$ nommée portée, le semi variogramme peut atteindre un seuil nommé palier, à partir duquel il n'y a plus de dépendance spatiale entre les données. Le palier peut être asymptotique ; dans ce cas-là, la portée est infinie. Sinon, la portée est définie par la distance à laquelle le covariogramme atteint 95% de la valeur de son palier.

Pour illustrer ces caractéristiques, voici une représentation du semi-variogramme :



3.2.3 Estimation et modélisation

Rappelons que le semi-variogramme est donné par :

$$\gamma(h) = \frac{1}{2}\mathbb{V}\text{ar}(\delta(s) - \delta(s+h)$$

En développant l'expression, on a :

$$\begin{aligned}
\gamma(h) &= \frac{1}{2}\mathbb{V}\text{ar}(\delta(s) - \delta(s+h)) \\
&= \frac{1}{2}\mathbb{V}\text{ar}[(Z(s) - \mu(s)) - (Z(s+h) + \mu(s+h))] \\
&= \frac{1}{2}\mathbb{V}\text{ar}(Z(s) - Z(s+h)) \\
&= \frac{1}{2}\mathbb{E}[(Z(s) - Z(s+h))^2] - \frac{1}{2}\mathbb{E}[Z(s) - Z(s+h)]^2 \\
&= \frac{1}{2}\mathbb{E}[(Z(s) - Z(s+h))^2] - \frac{1}{2}(\mu(s) - \mu(s+h))^2
\end{aligned}$$

Lorsque $\mu(.)$ est une fonction constante, comme dans le cas du krigeage simple et ordinaire, le deuxième terme s'annule. Donc il est possible d'estimer le semi-variogramme directement à partir des observations $z(s_i)$. L'estimation sera de type expérimental. L'estimation expérimental du semi-variogramme le plus commun est celui des moments :

$$\hat{\gamma}(h) = \frac{1}{2N(h)} \sum_{i=1}^{n} \sum_{j=1}^{n} [z(x_i) - z(x_j + h)]^2$$

où :

— $\hat{\gamma}(h)$ est l'estimation du semi-variogramme pour une distance de séparation h

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

— $z(s_i)$ et $z(s_j + h)$ sont les valeurs de l'attribut étudié aux points s_i et s_{j+h}, respectivement ; N est le nombre total de points dans l'échantillon.

Introduisons maintenant des modèles de semi-variogramme théorique, qui sont plus simples que l'estimation expérimental. On peut en citer plusieurs dont les modèles avec palier et sans palier, de portée asymptotique ou non.

Modèle sphérique

Le modèle sphérique est un modèle avec palier et avec portée exacte d'effet pépite c_0, de palier $c_0 + c$ et de portée a :

$$\gamma(h) = \begin{cases} 0 \text{ pour } h = 0 \\ c_0 \text{ pour } h > 0 \end{cases}$$

Modèle exponentiel

Le modèle exponentiel est un modèle avec palier avec une portée asymptotique doté d'un effet pépite c_0, d'un palier $c_0 + c$ et de portée pratique égale à $3a$:

$$\gamma(h) = c_0 + c(1 - exp(-\frac{h}{a}))\text{pour}h \leq 0$$

Modèle gaussien

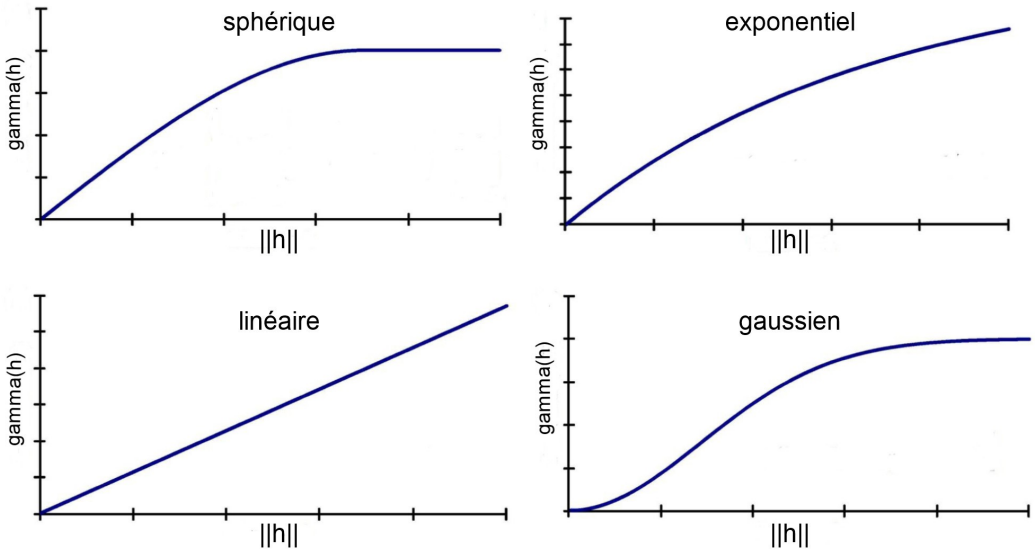
Le modèle gaussien est aussi un modèle de semi-variogramme avec palier et avec une portée asymptotique, de portée pratique $3a$.

$$\gamma(h) = c_0 + c(1 - exp(-\frac{h^2}{a^2}))\text{pour}h \leq 0$$

Modèle linéaire

Le modèle linéaire peut avoir ou non un palier. Dans le cas où il l'a, le palier s'exprime par $c_0 + c$ avec une portée tel que :

$$\gamma(h) = \begin{cases} c_0 + \frac{c}{a}h \text{ pour } 0 \leq h \leq a \\ c_0 + c \text{ pour } h > a \end{cases}$$



Chapitre 4

Krigeage

Dans ce chapitre, on se concentrera sur la partie mathématique de l'obtention de la forme d'une prévision en krigeage simple et en krigeage ordinaire. D'abord, le krigeage simple est le modèle de krigeage le plus simple, il suppose que l'espérance du processus est constante sur tout le champ, et que la variance est homogène. Le krigeage ordinaire est une généralisation du krigeage simple dans le cas où l'espérance est inconnue.

Ayant observé $(Z(s_1), \ldots, Z(s_n)$ pour $s_1, \ldots, s_n = S_obs$, le krigeage consiste à reconstruire la valeur du processus en tout point $s_0 \in S$ selon les quatre contraintes suivantes :

1. Linéarité : la prédiction est une combinaison linéaire des observations.
2. Autorisation : la variance de l'erreur de prédiction existe.
3. Non biais : la prédiction est sans biais.
4. Optimalité : la variance de l'erreur de prédiction est minimale

4.1 Krigeage simple

Contrainte de linéarité

La prévision $\hat{Z}_(s_0)$ prend la forme d'une combinaison linéaire et s'écrit :

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

Contrainte d'autorisation

Cet hypothèse n'intervient pas dans le krigeage simple car on fait l'hypothèse d'une stationnarité intrinsèque, ce qui implique l'existence de l'espérance et de la variance.

Contrainte de non-biais

L'absence de biais est nécessaire pour la prévision par le krigeage, on doit satisfaire l'égalité :

$$\begin{aligned}\mathbb{E}[\hat{Z}(s_0) - Z(s_0)] &= \mathbb{E}[a + \sum_{i=1}^{n} \lambda_i Z(s_i) - Z(s_0)] \\ &= \mathbb{E}[a] + \Lambda^t \mathbb{E}[\sum_{i=1}^{n} Z(s_i)] + \mathbb{E}[Z(s_0)] \\ &= a + \Lambda^t m \mathbb{1}_n - m \qquad\qquad = 0\end{aligned}$$

Donc, il faut que :

$$\begin{aligned}a &= m - \Lambda^t m \mathbb{1}_n \\ a &= m(1 - \Lambda^t \mathbb{1}_n)\end{aligned}$$

En remplaçant a dans l'expression de base, on obtient finalement :

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

Contrainte d'optimalité

La contrainte d'optimalité a pour but de minimiser $Var[\hat{Z}(s_0] - Z(s_0)$ sous les trois contraintes citées au-dessus On a :

$$Var[\hat{Z(}s_0 - Z(s_0)] = Var[m + \Lambda^t(Z - m\mathbb{1}) - Z(s_0)]$$

$$Var[\hat{Z(}s_0 - Z(s_0)] =$$

4.2 Krigeage ordinaire

Le krigeage ordinaire est une version généralisée du krigeage simple. En effet, il prend en compte l'aspect aléatoire de la moyenne dans l'espace contrairement au krigeage simple, qui suppose que la moyenne est la même partout dans le champ. Ce type de krigeage se développe sous l'hypothèse de la stationnarité intrinsèque et s'écrit :

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

où μ est une constante inconnue et $\delta(.)$ une fonction aléatoire stationnaire intrinsèque d'espérance nulle et de structure de dépendance connue. Le krigeage ordinaire est donc plus adapté aux jeux de données réelles plus grands et plus hétérogènes.

Contrainte de linéarité

La prévision $\hat{Z}(s_0)$ prend la forme d'une combinaison linéaire des variables aléatoires $Z(s_1), \ldots, Z(s_n)$ et s'écrit :

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

Contrainte d'autorisation

$$\begin{aligned}\hat{Z}(s_0) - Z(s_0) &= a + \sum_{i=1}^{n} \lambda_i Z(s_i) - Z(s_0) \\ &= a + \sum_{i=1}^{n} \lambda_i (\mu + \delta(s_i)) - (\mu + \delta(s_0)) \\ &= a + \mu \sum_{i=1}^{n} \lambda_i - \mu + \sum_{i=1}^{n} \lambda_i \delta(s_i) - \delta(s_0)\end{aligned}$$

Contrainte de non-biais

Contrainte d'optimalité

Chapitre 5

Mise en application

La suite de notre projet se concentre sur la mise en application des différents types de krigeage. L'implementation du krigeage ordinaire se fera à l'aide du package gstats de python tandis que l'implémentation du krigeage simple se fera à la main. Nous illustrerons ces implémentatios sur des données issues de *Geostatistics Resources*. Ces données contiennent des informations sur des mesures de profondeur, de porosité et de perméabilité d'une zone géologique. Ces données sont utilisées pour des études géologiques, notamment pour comprendre la structure et la qualité des réservoirs de pétrole et de gaz, ainsi que pour prédire la production de ces réservoirs.

5.1 SciKit-GStat

SciKit-GStat est un module d'analyse semblable à SciPy qui est spécialisé dans l'analyse de variogramme. La première étape consiste à calculer le semi-variogramme expérimental, qui rappelons-le, est utilisé pour modéliser la variabilité spatiale des données. Pour le calculer, SciKit-GStat utilise la classe la plus basique et la importante du module qui est skgstat.Variogram.

Cette classe prend en entrée plusieurs paramètres : les coordonnées et les valeurs de chaque point, le modèle de semi-variogramme théorique qui décrit le semi-variogramme expérimental, ainsi que la fonction de distance pour calculer la distance entre chaque point. Le nombre de lags, qui correspond au nombre d'intervalles dans lesquels on a divisé la plage de valeurs, est également important. Dans notre exemple, la classe skgstat.Variogram. sera nommée vg.

```
vg = skg.Variogram(coords, vals, model='exponential', dist_func='euclidean', n_lags=10)
```

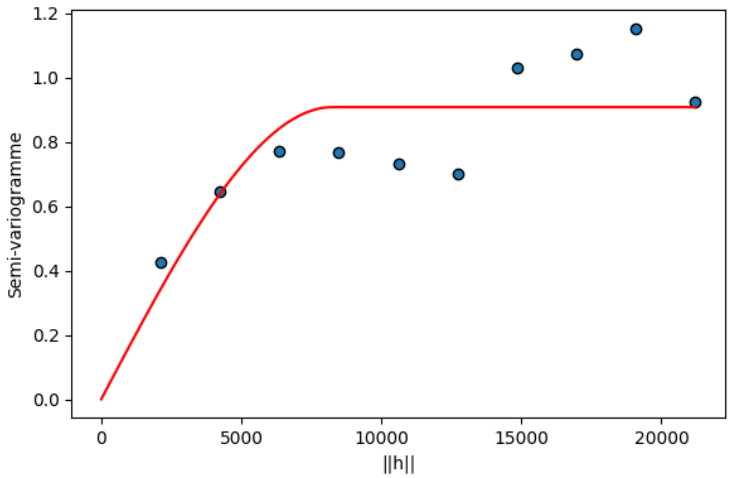
Lorsque la classe Variogram est compilée, elle retourne un objet < variogram > qui propose plusieurs méthodes pour décrire et analyser le variogramme.

```
< gaussian Semivariogram fitted to 10 bins >
```

Les méthodes qui nous intéressent sont :

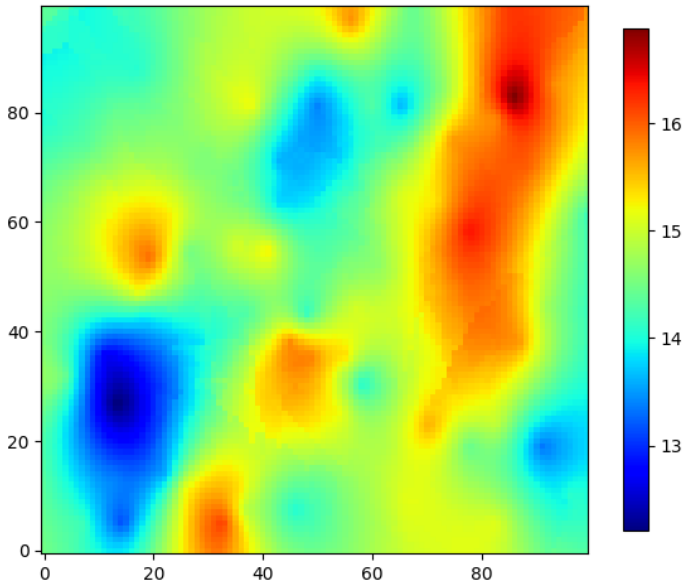
- variogram.data(), qui permet d'extraire les valeurs du variogramme théorique.
- variogram.bins, qui extrait les lags, les distances dans lesquelles les valeurs de semi-variogrammes ont été réparties.
- variogram.experimental, qui retourne les valeurs du semi-variogramme expérimental.

Ces méthodes nous permettent de tracer le semi-variogramme.

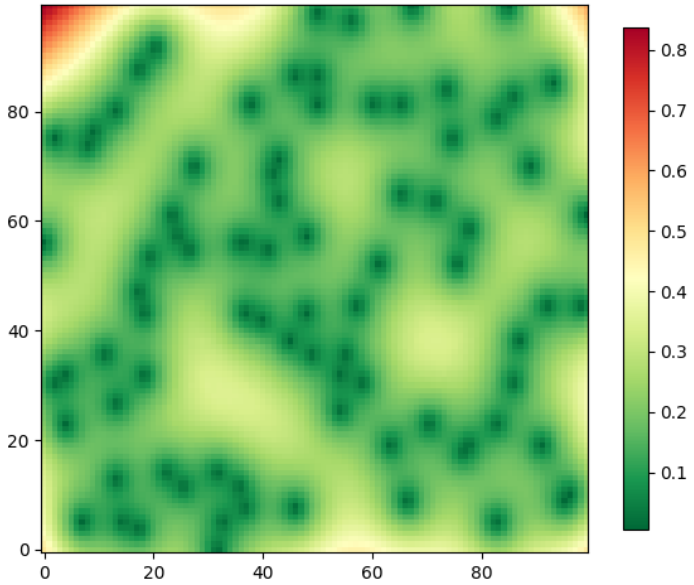


Une fois que le variogramme est calculé, on interpole les données inconnues en utilisant le krigeage ordinaire. Pour cela, gstat propose la classe skgstat.OrdinaryKriging, qui prend en argument l'objet variogramme précédemment calculé.

Ensuite, nous créons une grille sur laquelle nous allons interpoler les valeurs pour calculer les estimations dans les zones inconnues. La dernière étape consiste à utiliser l'attribut transform de l'objet OrdinaryKriging pour calculer les estimations sur chaque point de la grille. Pour obtenir la carte d'interpolation, les résultats sont visualisés à l'aide de matplotlib.



Par la suite, il est intéressant de se pencher sur la carte de la variance des erreurs. Le traçage de cette carte se fait à l'aide de l'attribut sigma de la classe ordinary krigeage. Il s'obtient ensuite exactement de la même manière que l'interpolation :



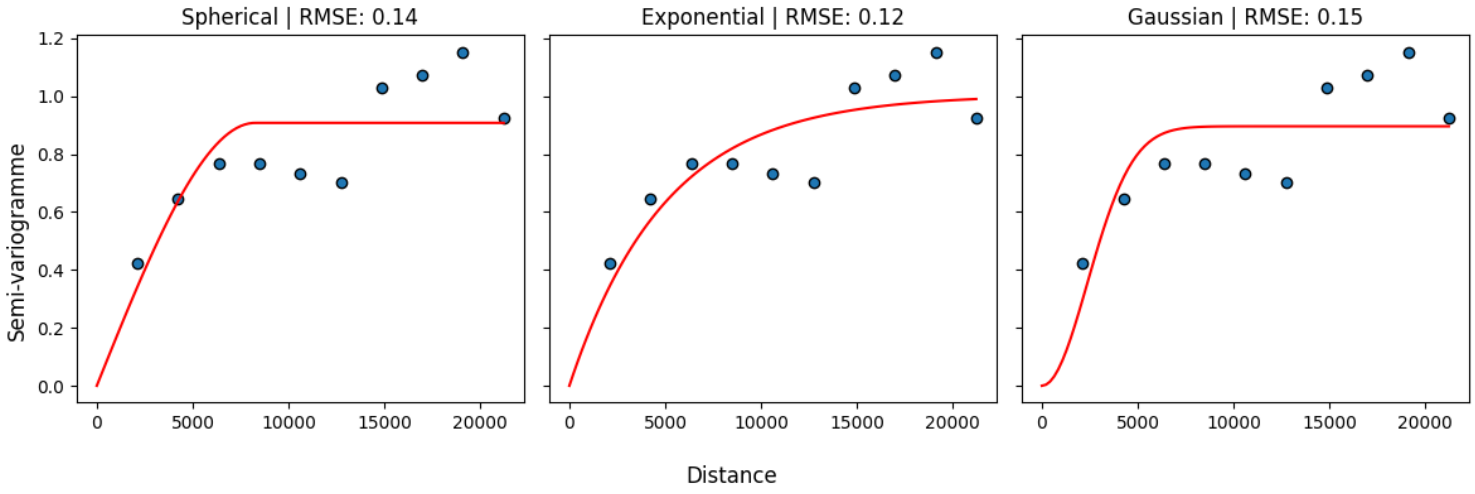
On remarque que la variance de l'interpolation est très faible lorsqu'on se trouve au niveau des points d'observations. Tandis que cette variance est plutôt élevé dans les zones non couverts. Il pourrait donc être intéressant de définir un seuil de variance maximal accepté de l'erreur de Krigeage pour identifier les régions où l'interpolation est plus ou moins exacte.

5.1.1 Comparaison de plusieurs modèles selon différents paramètres

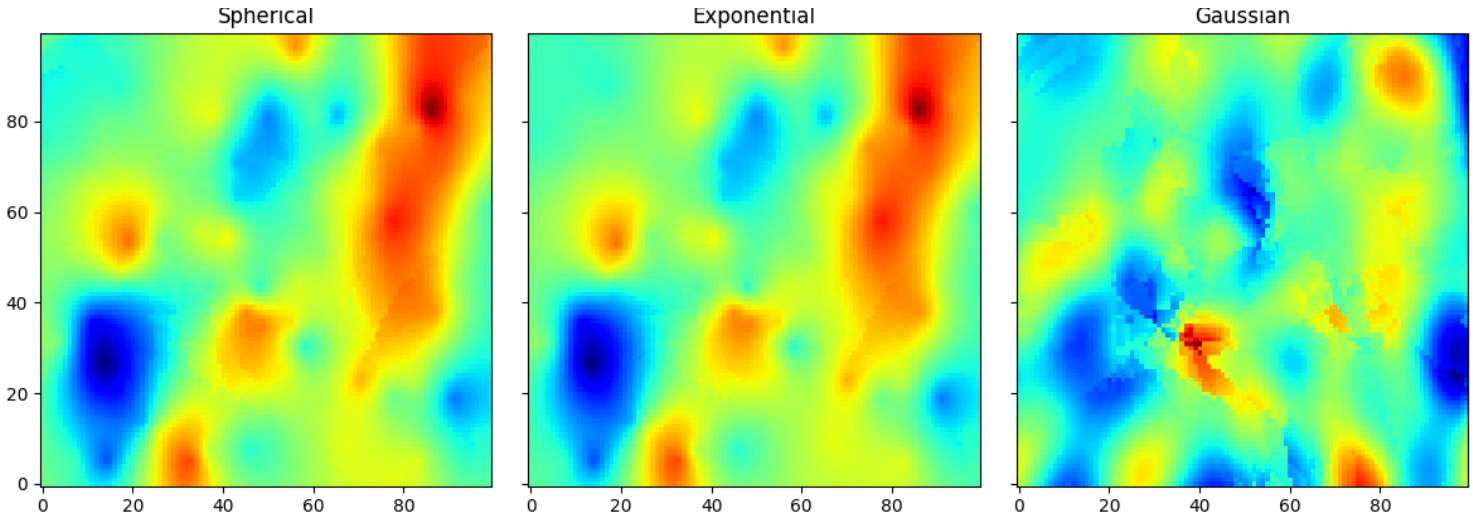
Il serait intéressant de voir l'influence de chaque paramètre en implémentant plusieurs variogrammes à partir de 3 modèles différentes :

— le modèle sphérique
— le modèle exponentiel
— le modèle gaussien

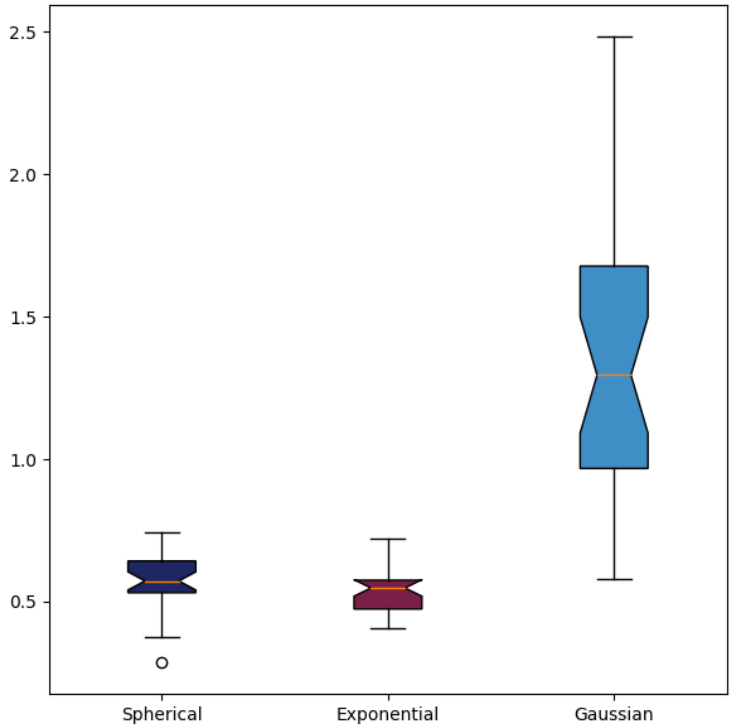
Pour chaque modèle, l'allure du semi-variogramme ainsi que le RMSE entre le semi-variogramme expérimental et semi-variogramme théorique est calculé. Nous obtenons alors :



A vue d'oeil, le modèle sphérique et le modèle gaussien semblent être très proches. De plus, leur RMSE ne diffèrent presque pas. Voyons ce qu'il en est de l'interpolation de ces modèles.



Cette fois-ci, il est devenu difficile de voir la différence entre le modèle sphérique et le modèle exponentiel. Par contre, l'interpolation par le modèle gaussien est assez mauvaise, elle est caractérisée par des flous redondants surtout en bas à gauche de la carte. Faisons une validation croisée de l'estimation du krigeage pour chaque modèle.



Une fois de plus, le modèle sphérique et le modèle exponentiel sont très proches, contrairement au modèle gaussien. Afin de mieux voir les différences, prenons en compte l'effet pépite dans nos paramètres.

5.2 Implementation du krigeage simple

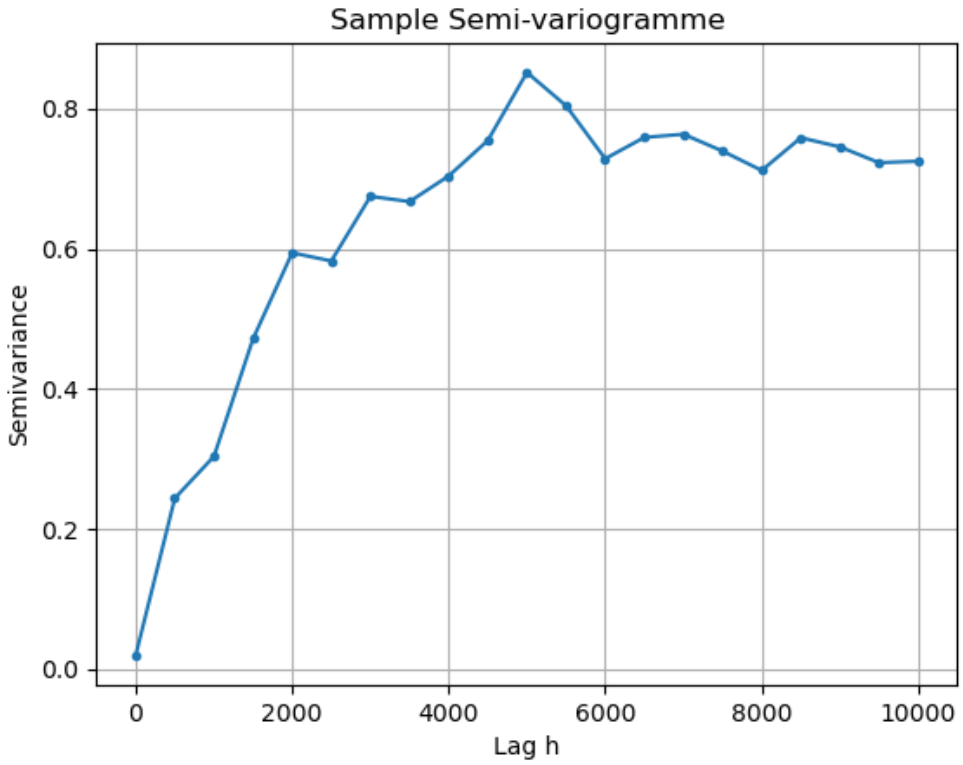
Dans cette partie, nous allons programmé à la main un krigeage simple. Comme vue dans les démonstrations précèdentes, le krigeage simple nous donne une expression directe de nos estimations via 3 éléments : la matrice de variance entre toutes les points ayant une valeur connus, un vecteur de variance entre le nouveau point considéré et les points connus et la moyenne que l'on suppose constant sur l'ensemble de l'espace étudié. Pour commencer, nous allons déjà devoir modéliser notre semi-variogramme et crée le modèle, ici nous allons utilisé le modèle exponentiel, qui fit le mieux notre semi-variogramme.

5.2.1 Création du Variogramme :

Pour commencer, nous devons crée un semi-variogramme directement via les données connues. J'ai décidé de garder une distance maximum entre les données de 10000 et de couper cette intervalle en 21 morceaux mesurant 500 chacun. Il suffit alors de prendre tous les points ayant une distance comprise dans chaque intervalle et de calculer l'estimation du semi-variogramme, ici le plus commun sera calculé, celui des moments :

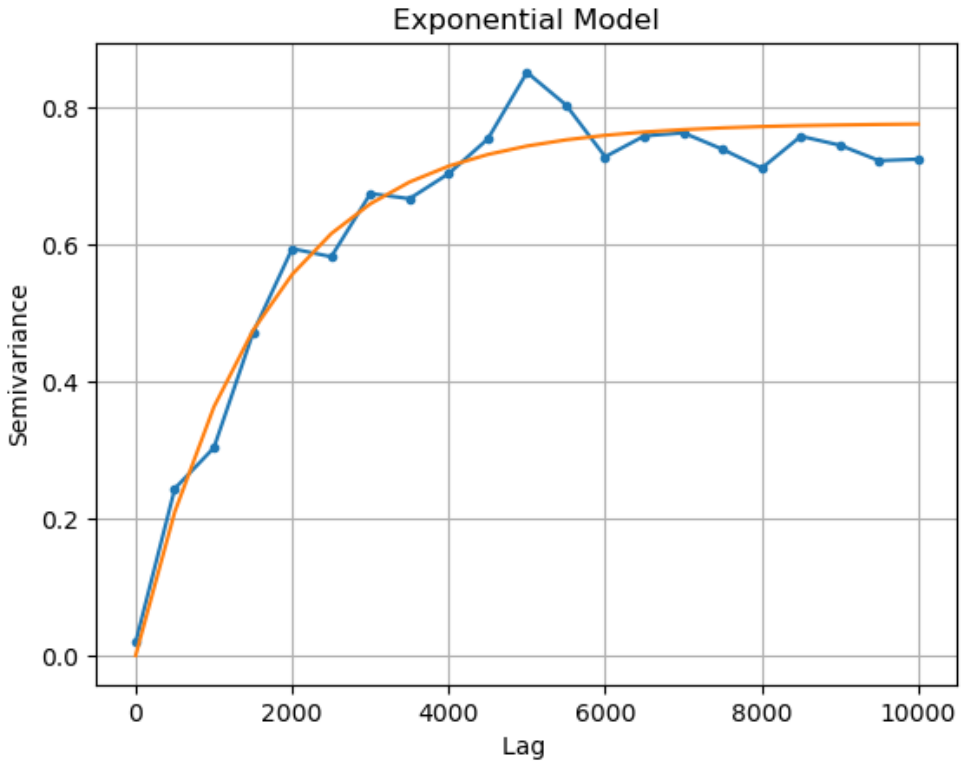
$$\hat{\gamma}(r) = \frac{1}{2\,|N(r)|} \sum_{N(r)} (z(s_i) - z(s_j))^2$$

Cela nous permet d'avoir le graphique suivant :



5.2.2 Interpolation via un modèle exponentiel :

Maintenant que nous avons notre modèle expérimental de semi-variogramme, il faut l'interpoler via un modèle dit classiquement utilisé en geostats. Nous allons utilisé le modèle exponentiel. Ce modèle demande cependant à paramétrer certains hyper-paramètre, nous allons ainsi optimiser notre hyper-paramètre "a" et nous allons garder le "a" qui minimise le MSE du modèle exponentiel et le semi-variogram des moments. Une fois optimiser, nous avons le modèle suivant :

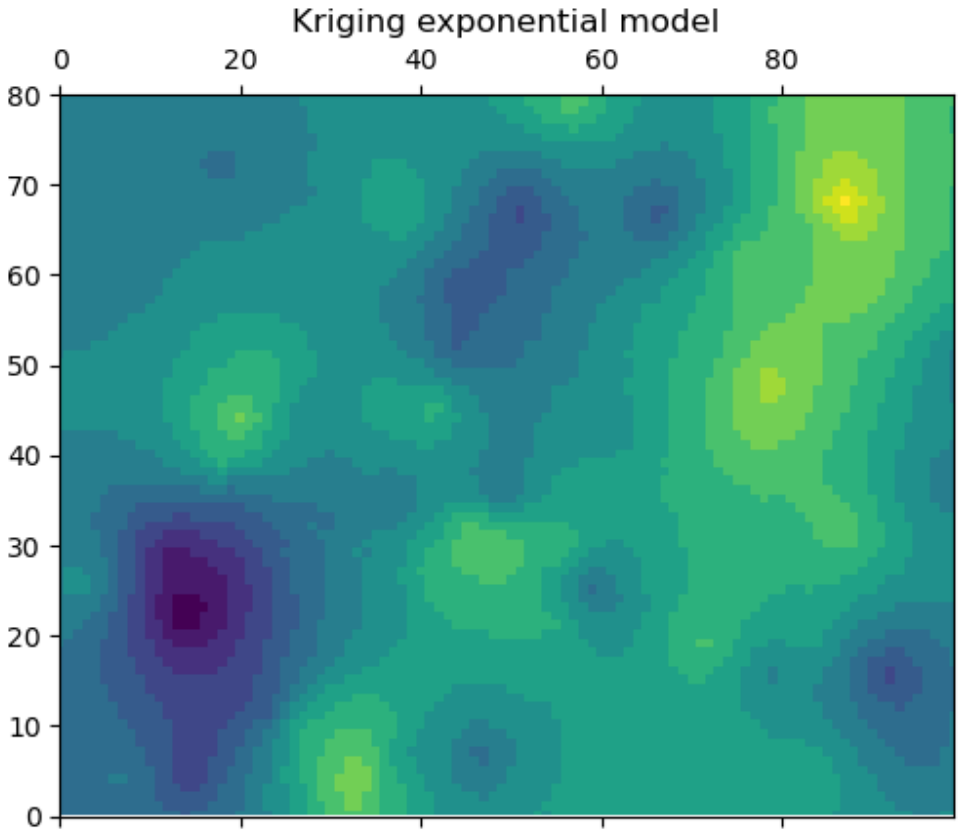


5.2.3 Krigeage simple :

Nous pouvons désormais faire le krigeage grâce à notre modèle calculer précèdement. Pour cela, nous allons délimiter un nombre de voisins qui vont nous servir à calculer les matrices de variance nous permettant de calculer l'estimation pour chaque point de la carte. Plus précisement, nous allons résoudre l'équation pour trouver le poinds représentant l'influence des données sur notre point considéré.

$$\lambda = (\sum)^{-1} * \sigma_{s_0}$$

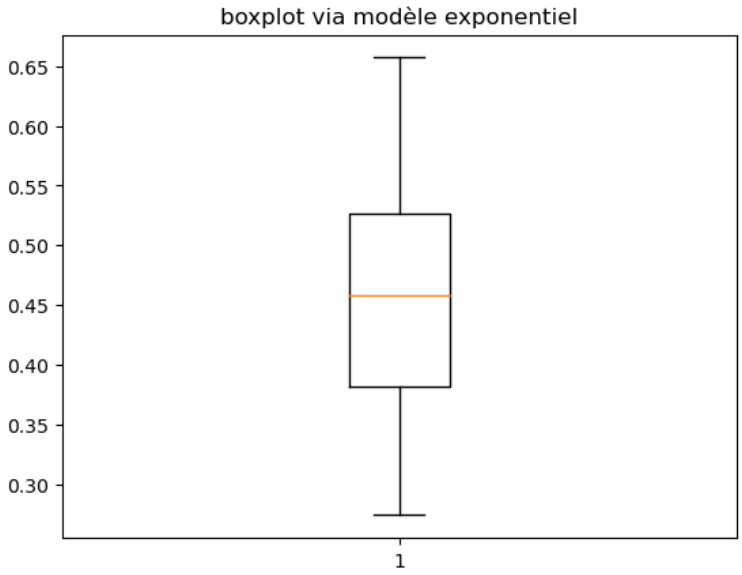
En faisant un recadrage sur une fenêtre [80*100], nous obtenons le graphique suivant :



5.2.4 Validation croisées :

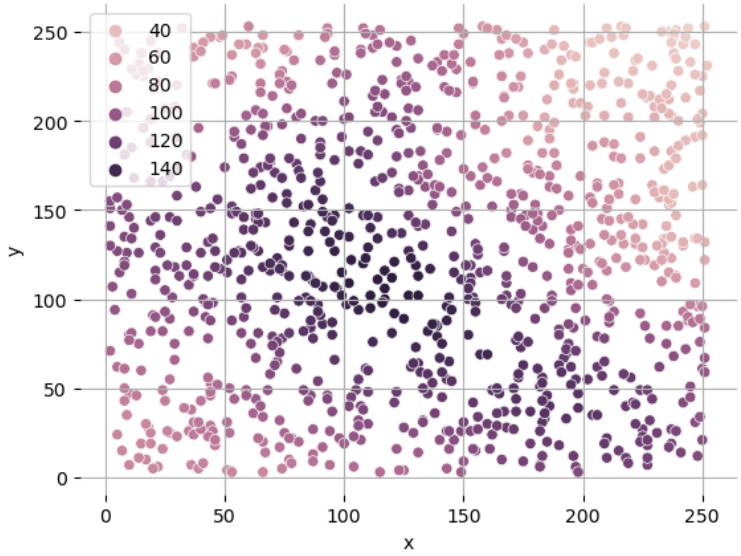
Afin de vérifier la robustesse de l'algorithme, nous allons utilisé la méthode de la validation croisée afin de mesurer l'écart entre des valeurs connus avec les valeurs simulés avec les même coordonnées. Ici, je vais garder 75 points pour l'apprentissage et 10 points pour la mesure d'erreur. Nous pouvons voir via l'écart type d'un boxplot la robustesse de l'algorithme.

Nous obtenons la boxplot suivante :

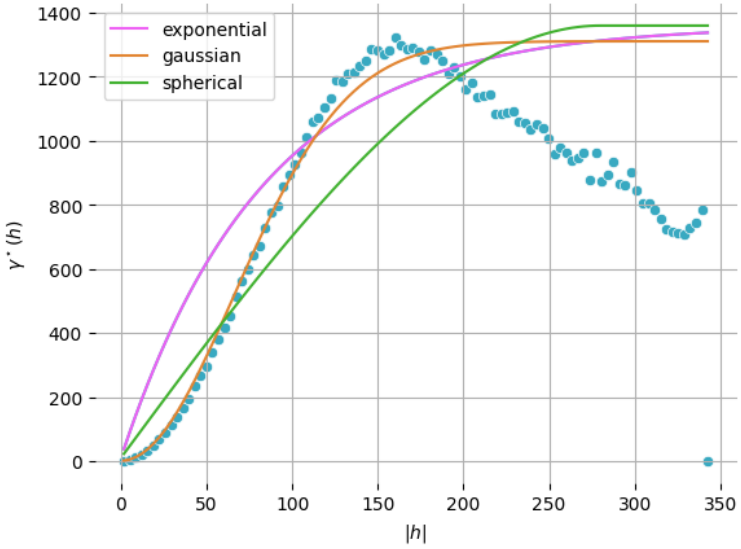


5.3 Problème d'interpolation :

Lors de la partie précèdente, nous partons du principe que notre modèle de semi-variogramme sera interpolable via un modèle classique tel que exponentiel, sphérique, gaussien ,... . Ces modèles sont souvent utilisés en geostats car l'espace étudié respecte l'idée de l'existence d'un pallier et donc d'une certaine croissance de variance plus la distance ente les points est grande. Or il est possible de trouver un problème de krigeage ayant une disposition ne vérifiant pas cette propriété. Regardons le cas suivant :



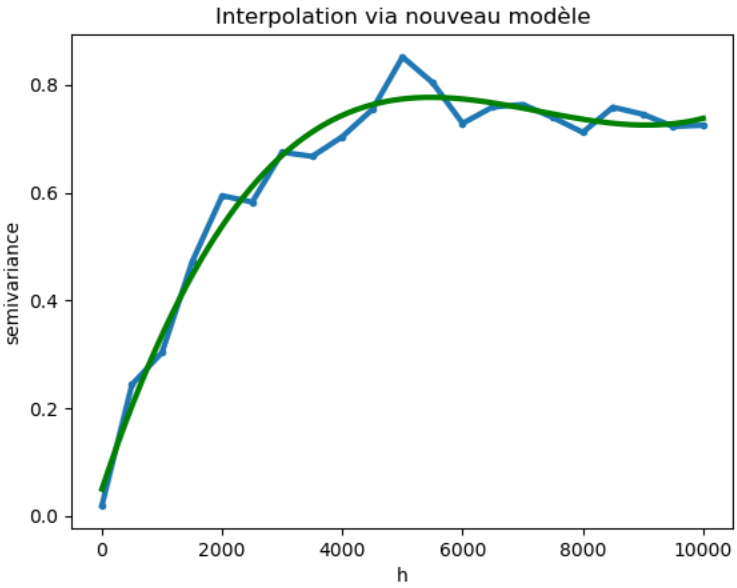
On voit ici que les données les plus distants auront un lieu plus fort qu'avec les données plus proches d'elles. Nous pouvons voir que les modèles classiques ne sont vraiment pas utilisable directement avec un graphe de semi-variogramme.



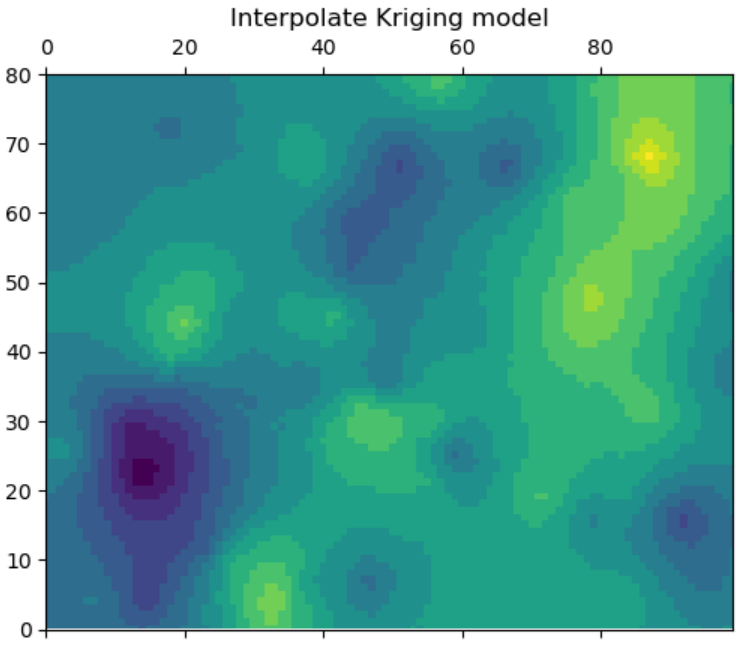
Nous pouvons donc nous poser la question suivante : L'étude du semi-variogramme par interpolation autre que cas classique nous donnera un meilleur résultat avec les données initial ?

5.3.1 Interpolation via package

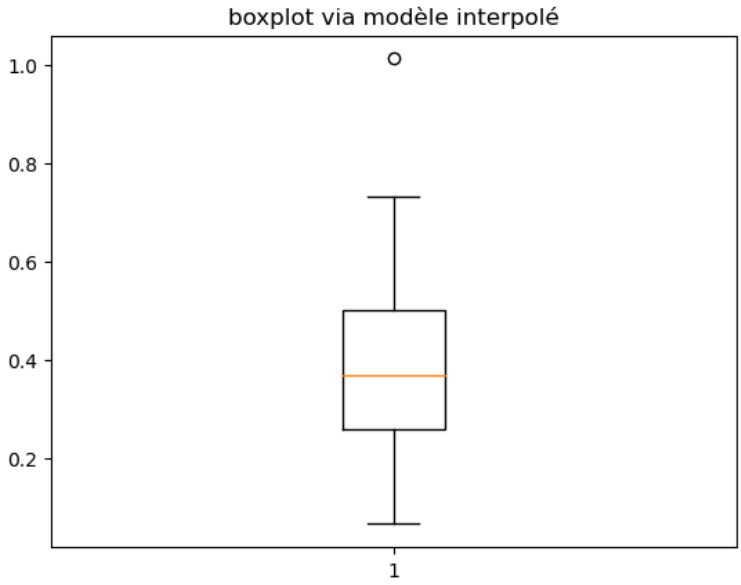
Pour interpoler les données sans overfitter, on utilise le package : "scipy.interpolate". Nous utilisé le paramètre "smooth" pour éviter l'overfit. Cela nous donne le nouveau modèle suivant :



De même que précèdemment, on crée une fonction krige permettant d'estimer les points de notre espace. Cela donne ce nouveau krigeage :



Nous allons réévalué la robustesse de notre kriging avec notre nouveau modèle. Cela donne la boxplot suivant :


boxplot via modèle interpolé
1.0
0.8
0.6
0.4
0.2
1

Bibliographie

[1] F. PROIA, *S*éries chronologiques, Université d'ANGERS, Master 2 Data Sciences, 2023

[2] S. BAILLERGEON, *L*e krigeage : revue de la théorie et application à l'interpolation spatiale de données de précipitations, 2005

[3] J.M. FLOCH, *C*hapitre 5. Géostatique, Manuel d'analyse spatiale Théorie et mise en oeuvre pratique avec R Insee Méthodes n° 131 - octobre 2018

[4] Mirko Mälicke, Egil Möller, Helge David Schneider, & Sebastian Müller. (2021). mmaelicke/scikit-gstat : A scipy flavoured geostatistical variogram analysis toolbox (v0.6.0). Zenodo. https://doi.org/10.5281/zenodo.4835779

[5] J. MAURER, 2022, https://youtu.be/HDW-HLMxcnE

Annexes

```
import numpy as np
import skgstat as skg
import matplotlib.pyplot as plt
from scipy.spatial.distance import pdist, squareform
from skgstat.models import spherical


class gstats:
    def __init__(self, coords, vals, model, lags):
        self.model = model
        self.lags = lags
        self.coords = coords
        self.vals = vals

        # create the variogram
        self.vg = skg.Variogram(
            self.coords, self.vals, model=self.model, n_lags=self.lags
        )

        # creation of a grid to interpolate
        self.x = self.coords[:, 0]
        self.y = self.coords[:, 1]
        self.xx, self.yy = np.mgrid[
            self.x.min() : self.x.max() : 100j, self.y.min() : self.y.max() : 100j
        ]

        # application of ordinary kriging
        self.ok = skg.OrdinaryKriging(self.vg)

    def interpolate(self):
        """
        Fonction qui:
            - trace le semi-variogramme
            - trace l'interpolation par le krigeage ordinaire
        """

        # affichage du variogramme
        fig = plt.figure(figsize=(6, 4))
        plt.scatter(self.vg.bins, self.vg.experimental, edgecolor="black")
        plt.plot(self.vg.data()[0], self.vg.data()[1], c="r")
        #plt.title("semi-variogramme", fontsize=15)
```

```
    plt.xlabel("||h||")
    plt.ylabel("Semi-variogramme")

    plt.tick_params(
        axis="x", bottom=True, top=False, labelbottom=True, labeltop=False
    )
    plt.tight_layout()
    plt.savefig("variogram.png")


    # interpolation of data over the grid
    field = self.ok.transform(self.xx.flatten(), self.yy.flatten()).reshape(self.xx.

    # affichage de l'interpolation
    fig, ax = plt.subplots(1, 1, figsize=(6, 6))
    m = ax.matshow(field.T, cmap="jet")
    plt.colorbar(m, shrink=0.7)
    plt.gca().invert_yaxis()
    plt.tick_params(
        axis="x", bottom=True, top=False, labelbottom=True, labeltop=False
    )
    plt.tight_layout()
    plt.savefig("interpolation.png")


def compare_variogram(self):
    """
    compare_variogram() est une fonction qui compare 3 modeles selon le semi-variogr
    La fonction compare:
        - le modele exponential
        - le modele spherical
        - le modele gaussien

    Il retourne un graphique qui affiche le semi-variogramme et le RMSE de chacun de

    """
    fig, axes = plt.subplots(1, 3, figsize=(12, 4), sharey=True)
    axes = axes.flatten()

    for i, model in enumerate(("spherical", "exponential", "gaussian")):
        self.vg.model = model   #choice of a model
        axes[i].scatter(self.vg.bins, self.vg.experimental, edgecolor="black")
        axes[i].plot(self.vg.data()[0], self.vg.data()[1], c="r")
        axes[i].set_title("%s | RMSE: %.2f" % (model.capitalize(), self.vg.rmse))
    fig.supxlabel("Distance")
    fig.supylabel("Semi-variogramme")
    plt.tight_layout()
    plt.savefig("compare_variogram.png")


def compare_interpolation(self):
    """
```

```
    fonction qui compare l'interpolation par le krigeage ordinaire des modeles expo

    """
    fig, _a = plt.subplots(1, 3, figsize=(12, 4), sharex=True, sharey=True)
    axes = _a.flatten()

    for i, model in enumerate(("spherical", "exponential", "gaussian")):
        vg = skg.Variogram(self.coords, self.vals, model=model, n_lags=self.lags)
        ok = skg.OrdinaryKriging(vg)
        field = ok.transform(self.xx.flatten(), self.yy.flatten()).reshape(self.xx.s
        m = axes[i].matshow(field.T, cmap="jet")
        axes[i].set_title(f"{model.capitalize()}")
        axes[i].invert_yaxis()
        axes[i].tick_params(
            axis="x", bottom=True, top=False, labelbottom=True, labeltop=False
        )
    plt.tight_layout()
    plt.savefig("compare_interpolation.png")


def interpolate_error(self):
    """
    fonction qui retourne l'interpolation des erreurs du modele choisi de base
    """
    s2 = self.ok.sigma.reshape(self.xx.shape)
    fig, ax = plt.subplots(1, 1, figsize=(6, 6))
    m = ax.matshow(s2.T, cmap="RdYlGn_r")
    plt.colorbar(m, shrink=0.7)
    plt.gca().invert_yaxis()
    plt.tick_params(
        axis="x", bottom=True, top=False, labelbottom=True, labeltop=False
    )
    plt.tight_layout()
    plt.savefig("error.png")


def compare_boxplot(self):
    """
    fonction qui compare le boxplot des cross-validations des modeles spherical, ex
    """
    box1 = []
    box2 = []
    box3 = []
    for i, model in enumerate(("spherical", "exponential", "gaussian")):
        self.vg.model = model
        for j in range(30):
            if model == "spherical":
                box1.append(self.vg.cross_validate(method="jacknife", n=10))
            elif model == "exponential":
                box2.append(self.vg.cross_validate(method="jacknife", n=10))
            elif model == "gaussian":
                box3.append(self.vg.cross_validate(method="jacknife", n=10))
```

```
        boxes = np.stack([box1, box2, box3], axis=1)
        colors = ["#1e2761", "#7a2048", "#408ec6"]
        plt.figure(figsize=(6, 6))
        bxplot = plt.boxplot(boxes, patch_artist=True, notch=True)
        for patch, color in zip(bxplot["boxes"], colors):
            patch.set_facecolor(color)
        plt.xticks([1, 2, 3], ["Spherical", "Exponential", "Gaussian"])
        plt.tight_layout()
        plt.savefig("boxplots.png")


if __name__ == "__main__":
    #import data
    data = open("ZoneA.dat", "r").readlines()
    data = [i.strip().split() for i in data[10:]]
    data = np.array(data, dtype=np.float32)
    coords = np.array(list(zip(data[:, 0], data[:, 1])))
    vals = data[:, 3]

    #choice of parameters
    model = "spherical"
    n_lags = 10

    #compile all the functions
    krige = gstats(coords, vals, model, n_lags)
    krige.interpolate()
    krige.compare_variogram()
    krige.compare_interpolation()
    krige.interpolate_error()
    krige.compare_boxplot()
```