Prédiction spatio-temporelle et validation de modèles

Liliane Bel, Thomas Romary

Atelier RESSTE

Avignon, 27-29 avril 2016

Prédiction spatio-temporelle

$Z(s,t)$ champ spatio-temporel, $s \in D$ site spatial, $t \in (0,\infty)$ indice de temps.
Mesures disponibles : $z(s_i,t_j)$, $i=1,ns$, $j=1,T$
Cas idéal : les mesures sont disponibles pour tous les instants à toutes les stations,
en général il y a des données manquantes.

Prédiction spatio-temporelle :

- $\widehat{Z}(s_0,t_j)$
- $\widehat{Z}(s_i,T+h)$
- $\widehat{Z}(s_0,T+h)$

Modèles avec tendance

$$Z(s,t) = \mu(s,t) + \nu(s,t)$$

$$\mu(s,t) = \sum_{\ell=1}^{L} \gamma_\ell \mathcal{M}_\ell(s,t) + \sum_{i=1}^{m} \beta_i(s) f_i(t)$$

$\mathcal{M}_\ell(s,t)$ covariables spatio-temporelles, γ_ℓ coefficients
$\{f_i(t)\}_i$ fonctions de base temporelles (décomposition en valeurs singulières)
$\beta_i(s)$ coefficients variant spatialement (krigeage universel, dépendent de covariables),indépendants
$\nu(s,t)$ champ spatial indépendant en temps, stationnaire en espace

Package *SpatioTemporal* (J. Lindstrom et al)

- choix de la base $\{f_i(t)\}_i$
- complétion de données manquantes
- estimation par maximum de vraisemblance (profilée, REML)
- prédiction, calcul de la variance
- validation croisée

Modèles avec tendance

$$Z_t = X_t\beta + KY_t + e_t \qquad e_t \sim \mathcal{N}(0, \Sigma_e)$$

$$Y_t = GY_{t-1} + \eta_t \qquad \eta_t \sim \mathcal{N}(0, \Sigma_\eta)$$

$Y_0 \sim \mathcal{N}(m_0, C_0)$
Z_t : $ns \times 1$ observations aux stations
Y_t : $p \times 1$ processus latent
X_t : $L \times ns$ covariables
e_t : champ aléatoire indépendant en temps, covariance spatiale

Package *Stem* (A. Cameletti)

- estimation paramètres par EM
- prédiction
- estimation de la variance d'erreur par bootstrap
- exemple PM10

Krigeage spatio-temporel

Covariance spatio-temporelle de Z : $C((s_1, t_1), (s_2, t_2))$
On veut prédire $Z(t_0, x_0)$ à partir des données Z sous la forme d'un prédicteur linéaire
Minimiser la variance de prédiction conduit au prédicteur de krigeage simple

$$Z^*(t_0, x_0) = c(x_0, t_0)\Sigma^{-1}\mathrm{Z}$$

où

- $\mathrm{Z} = (Z(t_j, x_i))$, $i = 1, \ldots, ns$, $j = 1, \ldots, T$,
- $\Sigma = \mathbb{V}(\mathrm{Z})$
- $c(x_0, t_0) = \mathrm{Cov}(Z(t_0, x_0), \mathrm{Z})$

La variance de krigeage simple s'écrit

$$\begin{aligned} \sigma^*(t_0, x_0) &= \mathbb{V}(Z^*(t_0, x_0) - Z(t_0, x_0))^{\frac{1}{2}} \\ &= (\mathbb{V}(Z(t_0, x_0)) - c(x_0, t_0)\Sigma^{-1}c(x_0, t_0))^{\frac{1}{2}} \end{aligned}$$

Complexité de calcul

En présence d'un grand nombre de données, l'inversion de Σ devient coûteuse, voire impossible
Solutions

- voisinage glissant
  - si le nombre de sites de prédiction n'est pas trop important
  - génère des discontinuités sur la carte prédite
  - choix de la forme du voisinage en fonction des paramètres du modèle de covariance
- tapering (cf. atelier 3)
  - on considère le produit de la fonction de covariance par une fonction de covariance à support compact de manière à obtenir une matrice creuse
  - fonctionne bien avec un jeu de données dense
  - effets de bords
- autres solutions : modèles de rang faible, spde

Schémas de validation

Validation externe (VE)

- on sépare le jeux de données en un ensemble d'apprentissage et de validation

Validation croisée (VC)

- Leave-one-out cross validation
- K-fold CV (moins coûteuse)

En VE et K-fold VC, on peut imaginer des sous-ensembles adaptés au contexte spatio-temporel pertinence ?

- spatial : on extrait un même sous-ensemble de stations en chaque pas de temps
- temporel : on extrait toutes les stations pour un sous-ensemble de dates

Quelques scores de prédiction

- Mean square error et Normalized mean square error

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\mathrm{Z}_i^* - Z_i)^2 \text{ et } \mathrm{NMSE} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\mathrm{Z}_i^* - Z_i}{\sigma_i^*}\right)^2$$

- Logarithmic score

$$\mathrm{LogS} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{2}\log(2\pi\sigma_i^{2*}) + \frac{1}{2}\left(\frac{\mathrm{Z}_i^* - Z_i}{\sigma_i^*}\right)^2\right)$$

- Continued Ranked Probability Score

$$\frac{1}{n}\sum_{i=1}^{n}\int_{-\infty}^{\infty}(F_i(y) - \mathbf{1}_{Z_i^* \leq y})dy$$

où $F_i(y) = \mathbb{P}(Z_i \leq y | Z_j, j \neq i)$

Implémentation en R

- gstat

  `krigeST(...)`

  voisinage éventuellement défini par un nombre maximum de voisins, unique par défaut
  pas de simulation

- CompRandFld

  `Kri(...)`

  permet en particulier d'utiliser le covariance tapering
  simulations non-conditionnelles via `RFism`, cf. ci-dessous

- RandomFields

  `RFinterpolate(...)`

  voisinage défini par nombre maximum de voisins
  simulations conditionnelles et non-conditionnelles via `RFismulate`

gstat

```
krigeST(formula, data, newdata, modelList, beta,
        nmax = Inf, computeVar = FALSE)
```

où

- `formula` permet de modéliser le terme de moyenne incluant éventuellement des covariables. ex : `PM10~1` pour un krigeage ordinaire
- `data` les données d'entrées, au format ST
- `newdata` les coordonnées des points cibles, au format ST
- `modelList` le modèle de variogramme au format vgmST
- `beta` la moyenne connue pour un krigeage simple
- `nmax` nombre maximum de voisins pour un krigeage en voisinage glissant. Voisinage unique par défaut
- `computeVar` calcul de la variance de krigeage
- et d'autres options moins utiles

Renvoie un objet de classe ST contenant les valeurs prédites

CompRandFld

```
Kri(data, coordx, coordy=NULL, coordt=NULL, corrmodel,
    distance="Eucl",grid=FALSE, loc, maxdist=NULL,
    maxtime=NULL, param, taper=NULL,tapsep=NULL, time=NULL,
    type="Standard",type_krig="Simple")
```

où

- `data` les données d'entrées, au format standard (matrix)
- `newdata` les coordonnées des points cibles, au format ST
- `beta` la moyenne connue pour un krigeage simple
- `nmax` nombre maximum de voisins pour un krigeage en voisinage glissant. Voisinage unique par défaut
- `computeVar` calcul de la variance de krigeage
- et d'autres options moins utiles

RandomField

```
RFinterpolate(model, x, y = NULL, z = NULL, T = NULL,
grid=NULL, distances, dim, data, given=NULL,
err.model, ignore.trend = FALSE, ...)
```

où

- `model` modèle de covariance avec paramètres
- `data` les données d'entrées, au format standard (matrix, x_i, y_i, z_i, T_i, régulier en T) ou RFsp
- `x`, `y`, `z` les coordonnées des points cibles, régulier en T
- `grid` les coordonnées des points cibles, sur une grille

sorties
$T vector de taille $nx \times ny \times nT$ ....