Génie Mathématiques et Modélisation
INSA Toulouse
135 Avenue de Rangueil
31400 Toulouse

Unité MIAT
INRA
24 Chemin de Borde-Rouge
31326 Castanet-Tolosan Cedex

Analyse de sensibilité et planification d'expériences

Marjory Coustaroux

Stage de 4ème année - Génie Mathématiques et Modélisation
option Statistiques - 2012/2013

Maîtres de stage : Robert FAIVRE & Victor PICHENY
Période de stage : Du 1er juillet 2013 au 20 septembre 2013

Remerciements

Je tiens tout d'abord à remercier mes encadrants Robert Faivre et Victor Picheny pour le temps qu'ils m'ont consacré et l'aide qu'ils ont pu m'apporter tout au long de mon stage.

Je remercie également Régis Sabbadin et Fabienne Ayrignac, pour m'avoir accueilli et introduit au sein de l'unité MIAT.

Enfin, je remercie également l'ensemble du personnel de l'INRA et particulièrement les membres de l'unité MIAT pour leur accueil et le très bon environnement dans lequel ils m'ont permis d'effectuer ce stage.

Table des matières

Introduction

De nos jours, beaucoup de modèles sont développés afin de décrire de nombreux phénomènes dans le domaine de la biologie ou de l'agronomie par exemple. Cependant, ces modèles, de plus en plus complexes, sont difficiles à appréhender. En effet, ils prennent en compte un grand nombre de facteurs d'entrée. L'unité MIAT de l'INRA de Toulouse, au sein de laquelle j'ai pu travailler tout au long de mon stage, s'intéresse à caractériser l'influence de ces paramètres du modèle sur les sorties obtenues par simulateur. Mon stage s'insère ainsi dans la thématique d'exploration de modèles par simulation.

Lorsque nous souhaitons analyser plus en détail un modèle ou le simplifier, nous pouvons avoir recours à de l'analyse de sensibilité. Ceci permet de savoir à quels paramètres d'entrée, ou facteurs, la sortie du modèle va être la plus sensible. Pour cela, nous utilisons des indices afin de déterminer la sensibilité de la sortie à une entrée donnée. Dans la plupart des cas, les modèles étant complexes, nous ne pouvons pas connaître la valeur théorique de ces indices : nous avons donc recours à leur estimation de ces indices. Différentes méthodes ont été développées afin de les estimer.

Les modèles étant aussi très coûteux en temps de calcul, nous utilisons alors des techniques parcimonieuses afin d'estimer ces indices : la planification d'expérience et la métamodélisation.

L'analyse du modèle est menée en général selon quatre étapes :

1. définition de l'espace dans lequel se situe l'analyse : valeurs possibles ou distributions des facteurs
2. planification d'expériences : choix du plan d'expérience (choix des valeurs d'un certain nombre de facteurs d'entrée) en essayant de choisir celui qui permettra de recouvrir le mieux possible l'espace des facteurs, et donc l'ensemble des combinaisons des valeurs des facteurs possibles
3. calcul des sorties du modèle, soit à partir du "vrai" modèle, soit à la suite d'une métamodélisation
4. estimation des indicateurs qui nous intéressent afin d'analyser le modèle, comme par exemple estimation des indices de sensibilité des facteurs

En premier lieu, l'objectif de mon stage a été de comparer différents plans d'expérience afin de voir si un des plans - que nous expliciterons dans la suite - donne de meilleurs résultats que les autres, en terme d'estimation des indices de sensibilité notamment. Nous nous demandions aussi si une méthode de métamodélisation, les processus gaussiens, donnerait de meilleures estimations des indices de sensibilité, en terme de biais et de variance des estimateurs, selon le plan d'expérience choisi pour effectuer cette métamodélisation.

Pour ce type de problème, il n'existe pas de résultats théoriques connus, et les calculs analytiques sont compliqués à effectuer. Nous avons donc utilisé une approche empirique, par expérimentation, en simulant un grand nombre de calculs à l'aide du logiciel **R**.

Dans une première partie, je vais vous présenter l'organisme au sein duquel j'ai travaillé. Puis j'exposerai les différents résultats théoriques et analytiques à partir desquels j'ai mené mon étude. Enfin, je parlerai des expériences que j'ai réalisées et des différentes analyses que nous avons pu faire pour le moment.

Abréviations

Tout au long de ce rapport, nous allons utiliser les abréviations suivantes :

FAST	Fourier Amplitude Sensitivity Testing
LHS	Latin Hypercube Sampling
INRA	Institut National de la Recherche Agronomique
MIAT	Mathématiques et Informatique Appliquées de Toulouse
MAD	Modélisation des Agro-écosystèmes et Décision
MC	plan de Monte-Carlo
OA-LHS	Orthogonal Array-based Latin hypercube
RMSE	Root-Mean-Square Error

Chapitre 1

Présentation de l'entreprise

1.1 L'INRA

L'INRA [1], Institut National de la Recherche Agronomique, est un Établissement Public à caractère Scientifique et Technologique. Son financement est principalement dû à des fonds publics.

En 1946, lors de sa fondation, il devait aider aux différents changements du monde agricole, ainsi que des filières alimentaires dans l'objectif d'évoluer selon les changements de la société et des problématiques de suffisance alimentaire. Depuis ce jour, les objectifs de l'INRA ont évolué, ainsi que les domaines d'étude tels que les domaines climatique, démographique et énergétique. L'INRA se concentre aussi bien sur des missions permettant de limiter les gaz à effet de serre d'origine agricole, que des missions favorisant l'adaptation de l'agriculture et des forêts au changement climatique non réversible. Pour ce faire, il faut par exemple connaître les comportements des individus à l'échelle des territoires ou des marchés, rechercher de nouvelles techniques pour la production énergétique, étudier les liens entre plantes et hommes ...

L'INRA incite le développement des connaissances scientifiques et l'innovation économique et sociale dans les domaines de l'alimentation, de l'agriculture et de l'environnement. Pour cela, l'INRA est présent au niveau mondial et est en permanence aux contacts des acteurs académiques, économiques, associatifs et territoriaux. Tous ces différents acteurs agissent au travers de différentes branches scientifiques : les sciences de la vie en majorité (68% des compétences scientifiques de l'INRA), les sciences des milieux et des procédés (12%), l'ingénierie écologique, les écotechnologies et les biotechnologies (8%), de même que les sciences économiques et sociales (8%) et les sciences du numérique (4%).

L'INRA est divisée, afin d'atteindre ces objectifs à échelle nationale et internationale, en 400 unités de recherche environ, réparties dans 17 centres (localisations) et 14 départements (grandes thématiques) de recherche.

1. http://www.inra.fr

1.2 Le centre de recherche de Toulouse Midi-Pyrénées

Le centre INRA de Toulouse Midi-Pyrénées est en partenariat avec de nombreuses académies, de domaines bien différents. Il possèdent des unités de tous les départements scientifiques de l'INRA. Il est associé aux universités de Toulouse (UT1 Capitole, UT3 Paul Sabatier), à l'école vétérinaire (INP-ENVT) et à des écoles d'ingénieurs (INP-ENSAT, INP-EIP, INP-ENSIACET, INSA). De plus, il est aussi en collaboration avec le CNRS.

Les activités de recherche sont en relation avec de nombreuses unités expérimentales (grandes cultures, élevages ovins et cunicole). L'INRA est fondateur du pôle de compétences en sciences agronomiques et vétérinaires « TOULOUSE AGRI CAMPUS » et est aussi un acteur du pôle de compétitivité sur les agro-chaînes « Agrimip Sud-Ouest Innovation ».

L'INRA Toulouse Midi-Pyrénées, au travers de toutes ses équipes, se concentre sur trois grands enjeux, de part ses activités de recherche et d'innovation :

1. des systèmes de production agricoles (végétaux, animaux) et forestiers plus durables et adaptés au changement climatique,
2. une alimentation attentive aux questions de santé,
3. de nouvelles filières de transformation des agro-ressources en faveur d'une valorisation du carbone renouvelable.

Ces recherches transdisciplinaires se divisent en 7 axes majeurs :

1. Génétique et biologie animale intégrative, maladies animales infectieuses et santé publique, conception des systémes d'élevage durable
2. Biologie intégrative des interactions plantes-environnement
3. Nutrition et prévention : toxicologie, biomarqueurs
4. Biotechnologies industrielles
5. Méthode et plate-forme pour la biologie intégrative animale, végétale et microbienne
6. Agro-écologie dans les territoires agricoles et forestiers
7. Économie de l'environnement et des marchés

Le centre INRA Toulouse Midi-Pyrénées est un des 5 plus grands centres INRA (au niveau des effectifs). C'est un axe de référence en toxicologie alimentaire et pour les recherches sur le tournesol et le lapin. Avec le renforcement de la génomique, des biotechnologies vertes et blanches, de la modélisation et avec la construction future d'un data center, le centre de Toulouse conserve un niveau d'excellence de ses programmes technologiques.

1.3 L'unité MIAT

L'Unité de Mathématiques et Informatique Appliquées de Toulouse (MIAT, anciennement Unité de Biométrie et Intelligence Artificielle) fait partie du département de

Mathématiques et Informatique Appliquées (MIA) de l'INRA. L'unité MIAT a pour objectif le développement et le mise en œuvre de méthodes mathématiques et/ou informatiques appropriées à la résolution de problèmes particuliers que peuvent avoir, en général, d'autres départements de l'INRA. Depuis janvier 2011, l'unité possède deux équipes de recherche (MAD et SaAB) et trois équipes de service (plateformes Bioinformatique du GIS Genotoul, RECORD et SIGENAE) :

- MAD (Modélisation des Agro-écosystémes et Décision)
- SaAB (Statistiques et Algorithmique pour la Biologie)
- plateforme Bioinformatique du GIS GENOTOUL - Génopole Toulouse Midi-Pyrénées
- RECORD (plateforme de modélisation et de simulation des agro-écosystèmes)
- SIGENAE (plateforme « Systèmes d'information des génômes des animaux d'élevage »)

Lors de mon stage, j'ai été affectée à l'unité MIAT, au sein de l'équipe MAD.

1.4 L'équipe MAD

Les activités de recherche de l'équipe MAD concernent le développement de modèles et méthodes mathématiques et informatiques pour l'analyse et la conduite des agro-écosystèmes. Ces systèmes sont définis par l'interaction entre des activités agricoles et des ressources naturelles ou biologiques, aux échelles de la parcelle, de l'exploitation agricole ou du territoire, de la décision individuelle à la gestion collective, pour des horizons temporels allant de la journée à plusieurs décennies.

Ces travaux sont basées sur des collaborations finalisées avec des équipes de recherche en agronomie, épidémiologie, sylviculture, écologie... Les problématiques portent sur les organisations et modes de gestion innovants des agro-écosystèmes face au contexte actuel de changement global (climat, démographie...).

Ses productions prennent la forme de publications scientifiques, d'ouvrages de synthèse, de supports de cours ou encore de logiciels.

Les méthodologies utilisées et développées par l'équipe MAD concernent la modélisation, la simulation, l'exploration, l'optimisation de systèmes dynamiques représentant des agro-écosystème pilotés.

Chapitre 2

Description des méthodes utilisées

2.1 Analyse de sensibilité et estimation des indices

L'analyse de sensibilité consiste à regarder sur un modèle donné à quelles entrées la ou les sortie(s) est (sont) la (les) plus sensible(s). Pour cela, nous utilisons des indices de sensibilité, compris entre 0 et 1. On distingue deux indices importants : l'indice principal et l'indice total.

- L'indice principal (ou indice du premier ordre) d'une variable x_i ne prend en compte que la sensibilité de la sortie à l'effet principal de la variable x_i seule. Sa formule est la suivante :

  $$SI_i = \frac{\mathrm{Var}(\mathrm{E}[Y|X_i])}{\mathrm{Var}(Y)}$$

  Cet indice va permettre de savoir quelles sont les entrées dont il faut avoir une parfaite connaissance, c'est-à-dire celles pour lesquelles l'erreur d'incertitude doit être diminuée afin de diminuer la variance de Y.

- L'indice total d'une variable x_i prend en compte l'influence venant de la variable x_i seule mais aussi de toutes les interactions avec les autres variables $x_j, j \neq i$, quel que soit le degré d'interaction (une interaction d'ordre 3 étant une interaction entre 3 variables). Cet indice s'écrit :

  $$TSI_i = \frac{\mathrm{E}(\mathrm{Var}[Y|X_{-i}])}{\mathrm{Var}(Y)}$$

  où X_{-i} est l'ensemble des facteurs sauf X_i.

  Si le TSI_i d'une variable X_i est faible, nous pouvons fixer la valeur de X_i (arbitrairement) sans pour autant changer considérablement la valeur de la sortie.

Ces indices sont souvent difficiles à calculer analytiquement et nécessitent d'être estimés numériquement. Les méthodes d'estimation les plus connues sont les méthodes de Sobol, FAST et de Morris.

2.1.1 Méthode de Sobol

Cette méthode ne repose sur aucune hypothèse, si ce n'est que l'espérance et la variance de Y sont finies. Cependant, en contre-partie du peu d'hypothèses requises, elle exige un grand nombre de simulations. Elle est donc très intéressante pour des modèles peu coûteux.

Cette méthode est basée sur une analyse fonctionnelle de la variance.

Proposition : Pour tout modèle G(x) défini et intégrable sur le domaine $D = D_1 \times ... \times D_d$, il existe une décomposition unique de la forme :

$$G(x) = f_0 + f_1(x_1) + ... + f_d(x_d) + f_{1,2}(x_1, x_2) + ... + f_{d-1,d}(x_{d-1}, x_d) + ... + f_{1,...,d}(x_1, ..., x_d)$$

où toutes les fonctions f_U sont mutuellement orthogonales : $\langle f_U, f_V \rangle = 0, \forall U \neq V$.

On peut réécrire cela de la forme : $G(x) = \sum_{U \in \{1,...,d\}} f_U(x_U)$.

De plus, on peut écrire l'espérance et la variance d'une variable $Y = G(X)$:

$$\mathrm{E}(Y) = \int_D G(x) f(x) dx$$

où $f(x)$ est la densité de probabilité de X et D le domaine de définition.

$$\mathrm{Var}(Y) = \mathrm{E}(Y - f_0)^2 = \int_D (G(x) - f_0)^2 f(x) dx$$

où $f_0 = \mathrm{E}(Y)$ est la moyenne de Y.

En appliquant la proposition précédente à la variance de $Y = G(X)$, elle peut se décomposer de façon unique :

$$\mathrm{Var}(Y) = \sum_{U \in \{1,...,d\}} V_U$$

où $V_U = \int_{D_U} f_U(x_U)^2 \pi_U(x_U) dx_U$.

Pour estimer les indices de sensibilité vus en introduction par la méthode de Sobol, il faut choisir aléatoirement (par exemple par loi $U(0, 1)$, par suite de Sobol ou par un autre plan d'expérience) deux matrices A et B de dimension $N \times d$. d nouvelles matrices sont construites par mélange des deux matrices initiales. La sortie Y est alors calculée pour chacune des $(d + 2)$ matrices précédentes. Cela requiert $N \times (d + 2)$ simulations du modèle et on obtient le même nombre de sorties. Ensuite, à partir de cet échantillon, les indices sont estimés par évaluation d'intégrales en raison des expressions de l'espérance et de la variance de Y vues précédemment.

Pour pouvoir estimer la précision de l'estimation, la technique du bootstrap peut être utilisée. Elle fournit un intervalle de confiance pour un échantillon dont la loi n'est pas connue, en estimant la variance avec le même échantillon, sans évaluations supplémentaires du modèle G, mais avec des tirages aléatoires et avec remise de cet échantillon.

2.1.2 Méthode FAST

La méthode FAST (Fourier Amplitude Sensitivity Testing) est basée sur les principes de l'analyse de Fourier. Elle permet d'écrire la variance de Y selon une décomposition de Fourier :

$$\mathrm{Var}(Y) = \sum_{U\in\{1,\dots,d\}} V_U \text{ où } V_U = \sum_{\nu} |c_\nu(G)|$$

avec $|c_\nu(G)| = \int_D G(x)\exp(-i2\pi\sum_{j=1}^{d}\nu_j x_j)dx$.

L'échantillonnage est effectuée dans $D = [0,1]^d$ en parcourant le champ des fréquences possibles, avec $\nu = (\nu_1, ..., \nu_d)$ vecteur de fréquences, pour les entrées X_j. On évalue en fait $|c_\nu(G)| = \mathrm{E}[G(x)\exp(-i2\pi\nu.x)]$.
Pour un ν donné, on définit la composante spectrale d'une trajectoire, composée d'une suite de points $x_j = (x_{j,1}, ...x_{j,d})$, $j = 1, ..., N$:

$$D_\nu = \mid \frac{1}{N}\sum_{j=1}^{N} G(x_j)\exp(-i\nu u_j) \mid^2$$

où :
- $u_j = -\pi + 2\pi\frac{j-\frac{1}{2}}{N}$: suite de valeurs dans $[-\pi, \pi]$ équi-espacées
- les composantes des points x_j sont choisies telles que $x_{j,i} = g(\sin(\omega_i u_j + \phi_i))$
- g : fonction à déterminer selon la distribution. Pour la loi $U(0,1)$, g est la fonction : $x \longrightarrow \frac{1}{2} + \frac{1}{\pi}\arcsin(x)$.
- ω_i : fréquence des répétitions des $x_{j,i}$
- ϕ_i : déphasage permettant de varier les trajectoires

On peut écrire aussi : $D_\nu = A_\nu^2 + B_\nu^2$ où :

$$A_\nu = \frac{1}{N}\sum_{j=1}^{N} G(x_j)\cos(\nu u_j)$$
$$B_\nu = \frac{1}{N}\sum_{j=1}^{N} G(x_j)\sin(\nu u_j)$$

On obtient comme estimateur des indices de sensibilité :

$$\hat{SI}_i = \frac{\sum_{m=1}^{M} D_{m\omega_k}}{\mathrm{Var}(Y)} \text{ avec } M\text{=4 ou 6 en général}$$
$$T\hat{S}I_i = 1 - \frac{\sum_\nu D_\nu}{\mathrm{Var}(Y)}$$

avec des fréquences ν de la forme $\sum_{i'\neq i} m_{i'}\omega_{i'} < \omega_i$

Cette méthode est plus rapide que celle de Sobol mais elle reste tout de même assez coûteuse en temps de calcul. Elle nécessite $N \times d$ simulations. Elle est aussi souvent plus stable que la méthode de Sobol.

2.1.3 Méthode de Morris

Cette méthode s'appuie sur une discrétisation de l'espace des variables, c'est-à-dire qu'il n'y a qu'un certain nombre de points qui peuvent être échantillonnés. La sensibilité de la sortie à un des facteurs X_i est mesurée en comparant des résultats où seul ce paramètre X_i aura varié. C'est pour cela que cette méthode est classée parmi les méthodes *One At a Time* (OAT). Elle est robuste si le modèle possède une certaine régularité.

Considérons un modèle avec d facteurs continus à valeurs dans $[0,1]^d$ et

une discrétisation avec p niveaux. Ces facteurs peuvent prendre les niveaux : $\{0, \frac{1}{p-1}, \frac{2}{p-1}, ..., 1\}$. L'échantillonnage implique la construction d'une suite de r trajectoires pour lesquelles chaque facteur varie une seule fois par trajectoire, et ainsi chaque trajectoire passe par $d+1$ nœuds différents (en comptant le nœud de départ). Le nœud de départ est choisi aléatoirement ainsi que la direction de chaque pas entre deux nœuds successifs. La longueur de chaque pas est égale à $\frac{1}{p-1}$.

Le nombre de simulations nécessaire est alors $r \times (p+1)$, ce qui la rend beaucoup plus rapide que les méthodes de Sobol et FAST. Elle permet principalement un premier classement des facteurs :

- facteurs avec des effets négligeables
- facteurs ayant des effets linéaires et sans interaction
- facteurs ayant des effets non linéaires

L'indice de sensibilité de X_k est estimé par la somme des rapports entre la différence des valeurs en sortie entre deux entrées successives où seul X_k varie et le pas δ de l'échantillonnage. En notant :

$$\Delta_k^i = \epsilon \, \frac{G(x_1^i,...,x_k^i+\epsilon\delta,...,x_d^i) - G(x_1^i,...,x_k^i,...,x_d^i)}{\delta}$$

où $\epsilon = \pm 1$, G est la fonction du modèle et x_k^i la coordonnée de X_k au premier nœud sur l'axe associé.

La somme des valeurs absolues de ces termes est ensuite calculée :

$$\mu_k^* = \frac{1}{r} \sum_{i=1}^{r} |\Delta_k^i|$$

Plus cette valeur est grande, et plus la sortie est sensible à une variation du facteur X_k.

Nous pouvont aussi calculer un second indice, appelé variance des effets élémentaires :

$$\sigma_k = \sqrt{\frac{1}{r-1} \sum_{i=1}^{r} (\Delta_k^i - \mu_k)}$$

où $\mu_k = \frac{1}{r} \sum_{i=1}^{r} \Delta_k^i$.

Cet indice permet de mesurer les interactions entre les facteurs et la linéarité de notre modèle. Plus σ_k est grand, plus la probabilité que le modèle ne soit pas linéaire et qu'il y ait des interactions est grande. En effet, σ_k est nul lorsque les effets élémentaires sont égaux et donc lorsque l'effet d'une variation de X_k est identique quel que soit l'endroit, ceci impliquant que le modèle est linéaire.

Cette méthode permet ainsi une première approche de l'analyse de sensibilité d'un modèle, mais elle a besoin d'être complétée par une des méthodes précédentes.

Afin d'estimer ces indices de sensibilité, il faut déterminer un nombre fixé N d'expériences qui permettront le calcul des estimateurs. Pour cela, nous aurons recours dans un premier lieu à de la planification d'expériences.

2.2 Plans d'expérience

Un plan d'expérience est un ensemble de N vecteurs d'entrée ou facteurs x_i. Considérons un espace de dimension d et un plan de N expériences. Le plan d'expérience X est noté de la façon suivante :

$$X = \begin{bmatrix} x_{11} & \dots & x_{1d} \\ \vdots & \ddots & \vdots \\ x_{N1} & \dots & x_{Nd} \end{bmatrix}$$

Ces plans d'expérience sont utilisés pour toute analyse de modèle. En effet, nous ne pouvons pas réaliser toutes les expériences parmi le domaine de définition des facteurs d'entrée. Il faut choisir un nombre N de points couvrant le mieux possible le champ de valeurs que peuvent prendre les entrées, afin que l'ensemble des valeurs obtenues pour la sortie y soit le plus représentatif de tous les cas possibles. Ainsi, différents plans d'expérience existent, de propriétés différentes, chacun utilisant une façon particulière pour couvrir du mieux possible l'espace de définition des facteurs d'entrée.

Le plan d'expérience le plus simple est le plan de Monte Carlo dont les points x_i sont répartis suivant une loi prédéfinie (dans notre cas, la loi $U(0,1)$) et réajustés selon les bornes inférieure et supérieure des entrées du modèle. Ce plan sert de référence au cours de l'analyse des résultats obtenus à partir de différents plans d'expérience.

D'autres plans d'expérience ont été développés, chacun étant basé sur une répartition spécifique des points dans l'espace. Parmi ces plans, nous avons notamment utilisé les plans LHS ou Latin Hypercube Sampling, quelques suites à faible discrépance ainsi que les plans OA-LHS et tms-net, explicités dans la suite.

2.2.1 Latin Hypercube Sampling (LHS)

Définition : Un hypercube latin à n points et d variables est une matrice $n \times d$ dont chacune des colonnes est une permutation de l'ensemble $\{1, 2, \dots, n\}$. On le note $LHD(n, d)$.

Ainsi, pour un plan LHS à n observations, chaque dimension de l'espace sera découpée en n intervalles et un point par intervalle sera choisi (et ce pour chaque dimension). Par exemple, sur $\mathbb{R}^2$, un LHS à 5 points et un LHS à 10 points peuvent donner les plans suivants :

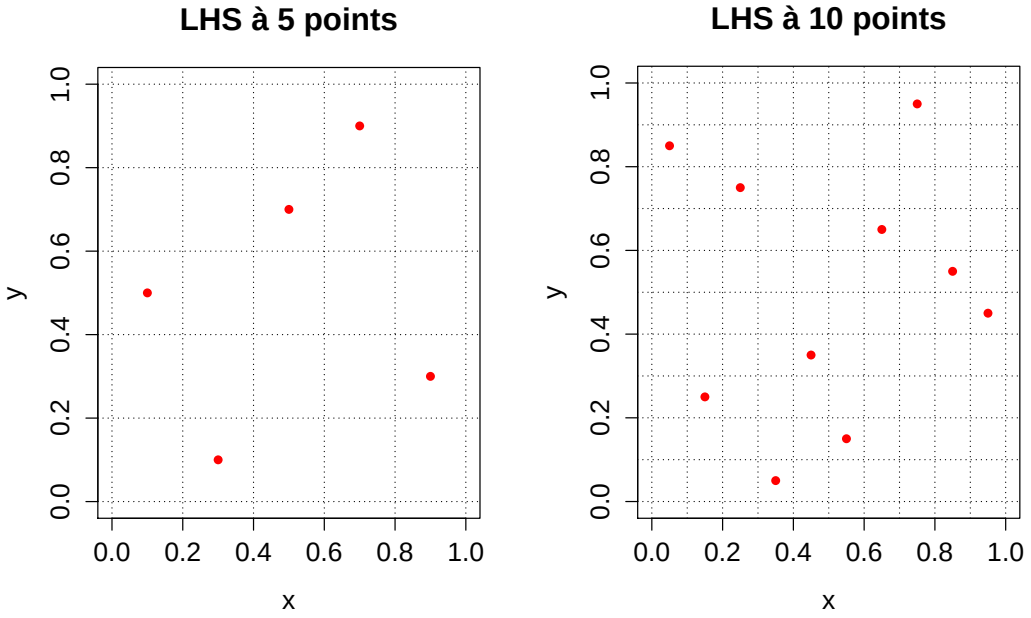


FIGURE 2.1 – Plans LHS avec 5 et 10 points

Ces plans sont très facile à générer et permettent une projection uniforme sur les marginales.

Nous pouvons aussi rajouter du brouillage (Scrambling) qui induit que chaque point est tiré selon la loi $U(0,1)$ dans son intervalle afin de ne pas avoir que des points centrés dans chaque intervalle.

Ces plans ne sont pas uniques car il existe de nombreuses combinaisons possibles pour les plans LHS "simples". Ils peuvent aussi être de très mauvaise qualité, ce qui est le cas quand nous obtenons un plan LHS composé de tous les points d'une des diagonales.

Ainsi, pour pallier à ce problème, des plans LHS optimisés ont été mis en place. Plusieurs critères d'optimisation existent et les deux plus connus sont le minimax et le maximin :

- minimax : la distance maximale entre un point x du domaine et le point de notre plan d'expérience le plus proche de x est minimisée, et ce pour tous les points du domaine D. Le minimax s'écrit de la façon suivante :

$$\min_{x_1,\ldots,x_N}[\max_{x\in D}(\min_i d(x_i,x))]$$

  où D est le domaine de l'ensemble des valeurs d'entrées possibles
- maximin : on va chercher à maximiser la distance minimale entre 2 points de notre plan d'expérience, soit à effectuer :

$$\max_{x_1,\ldots,x_N}(\min_{i\neq j} d(x_i,x_j))$$

Dans la réalité, ces méthodes d'optimisation, et notamment le minimax, sont très coûteuses. En général, nous nous contentons donc de choisir le meilleur plan LHS (au sens d'un des critères comme ceux déjà exposés) parmi un certain nombre de plans qui auront été générés.

En **R**, ces plans sont générés grâce au package *lhs*. J'ai notamment utilisé les fonctions *randomLHS* pour générer des plans LHS de "base" et *improvedLHS* pour des plans LHS optimisés.

2.2.2 Suites à faible discrépance

La discrépance mesure la déviation de la répartition dans l'espace d'un échantillon de points par rapport à celle d'un échantillon de même taille tiré par loi uniforme. Plusieurs définitions de la discrépance existent et, pour une suite $(x^{(i)})_{i=1...N}$, elle peut notamment s'écrire :

$$D(x) = \sup_{y\in[0,1[^K} \mid \frac{1}{N}\sum_{k=1}^{N} \mathbb{1}_{(x^{(k)}\in[0,y])} - \text{Volume}([0,y]) \mid$$

Les suites à faible discrépance sont des suites quasi-aléatoires ayant une faible valeur de discrépance. Ce sont potentiellement des plans d'expérience très satisfaisants, car elles permettent un bon remplissage de l'espace. Plusieurs suites à faible discrépance existent et nous avons notamment utilisé les suites quasi-aléatoires de Sobol et les suites de Halton.

Les Suites de Sobol

Les suites de Sobol permettent de répartir les différents points dans l'espace en minimisant la distance entre chaque observation. Leur construction est assez compliquée, et en arithmétique modulo 2 elle s'obtiennent par des récurrences linéaires à partir de polynômes primitifs sur le corps $\mathbb{Z}_2 = \{0, 1\}$. Elles sont nommées quasi-aléatoires car on peut toujours trouver les coordonnées du deuxième point à partir du premier et ainsi de suite.

Pour expliquer leur construction, il faut d'abord rappeler la notion de polynôme primitif :

Définition : Un polynôme $p(t)$ de degré m de la forme $t^m + a_1 t^{m-1} + ... + a_{m-1}t + a_m$ est dit primitif sur $\mathbb{Z}_2$ s'il est irréductible sur $\mathbb{Z}_2$ et si le plus petit entier i, tel qu'il divise $t^i - 1$ ou $t^i + 1$, est égal à $2^m - 1$.
Ceci revient à regarder si aucun polynôme de degré inférieur à m ne divise $p(t)$.

Ainsi une suite de Sobol $u = \{u^0, u^1, ..., u^{n-1}\}$ est construite, en dimension 1, de la façon suivante :

$$u^i = \frac{1}{2^s}\left(\oplus_{k=1}^{m} p_k l_k\right)$$

où :
- $(p_k,\ 1 \leq k \leq s)$ est la représentation binaire de i

- $s = \{ \begin{array}{ll} 1 & \text{si} \quad i = 0 \\ 1 + \lfloor \ln i \rfloor & \text{sinon} \end{array}$
- $l_k, 1 \leq k \leq m$ sont des entiers impairs tels que $1 \leq l_k \leq 2^k$
- $l_k, k > m$ se calculent par la récurrence :

$$l_k = 2a_1 l_{k-1} \oplus 2^2 a_2 l_{k-2} \oplus ... \oplus 2^{m-1} a_{m-1} l_{k-m+1} \oplus (2^m l_{k-m} \oplus l_{k-m})$$

où a_k sont les coefficients d'un polynôme primitif de degré m sur $\mathbb{Z}_2$.

Pour construire cette suite dans un espace de dimension d, il faut alors prendre d polynômes primitifs distincts.

Rapides à construire, ces suites permettent aussi de garder une bonne disposition dans l'espace malgré l'augmentation en dimension. Un des inconvénients réside dans l'initialisation des entiers l_k. Ces derniers étant aléatoire, la répartition des points dans l'hypercube unité peut être mal faite. Afin de pallier cet inconvénient, une technique de brouillage peut être utilisée.

Les Suites de Halton

Les suites de Halton sont construites à partir d'une base différente par dimension de l'espace, ces bases étant construites à partir de nombres premiers. Ce choix permet de minimiser la discrépance en réduisant le terme dominant de la majoration de la discrépance, ainsi que d'avoir des suites uniformes, c'est-à-dire uniformément distribuées dans le cube unité.
En posant $p \geq 2$ un nombre premier, un entier n se décompose de façon unique :

$$n = a_0^{n,p} + a_1^{n,p} p + ... + a_{k_{n,p}}^{n,p} p^{k_{n,p}}$$

avec :
- $k_{n.p}$ un entier tel que $k_{n,p} = \min\{k \geq 1, r_{k+1}^{n,p} = 0\}$
- $r_0^{n,p} = n$
- $a_k^{n,p} = r_k^{n,p} \mod p$ où $r_k^{n,p} = \frac{r_{k-1}^{n,p} - a_{k-1}^{n,p}}{p}$ et $1 \leq k \leq k_{n,p}$
  D'où, $1 \leq a_k^{n,p} \leq p-1$ et $a_{k_{n,p}}^{n,p} \neq 0$.

En posant $\Phi_p(n) = \frac{a_0^{n,p}}{p} + ... + \frac{a_{k_{n,p}}^{n,p}}{p^{k_{n,p}+1}}$, on peut alors définir la suite de Halton $u = (u_n)_{n \geq 1}$ de la façon suivante :

$$u_n^i = \Phi_{p_i}(n) \quad \text{avec} \quad p_i, 1 \leq i \leq d, \text{ les } d \text{ premiers nombres premiers.}$$

La discrépance des suites de Halton est alors : $D(x) \leq \bigcirc \left(\frac{\ln(N)^d}{N} \right)$ avec N la taille de la suite.

Ces suites sont beaucoup moins robustes que les suites de Sobol, notamment dès que la dimension est assez grande.

J'ai aussi pu comparer les résultats de ces deux types de suites avec d'autres suites quasi-aléatoires comme les suites de Torus.

En **R**, ces suites sont générées grâce au package *randtoolbox*. Afin de varier les résultats, le paramètre *scrambling* peut être utilisé, permettant un brouillage des points, ce qui peut parfois être une solution aux problèmes liés à la montée en dimension.

2.2.3 Plans OA-LHS et tms-net

Ces deux plans sont basés sur le même principe de répartition dans l'espace que les plans LHS de base. Cependant, ils possèdent une ou plusieurs contraintes supplémentaires.

Plan OA-LHS

Le plan OA-LHS (Orthogonal Array-based Latin hypercube) peut être décrit de deux façons :

- un plan LHS de "base" auquel une propriété supplémentaire est ajoutée : en découpant en plus l'espace selon des "cubes", il faut que dans chaque hypercube il y ait un point et un seul par niveau que peut prendre ces points. Cependant, pour que cette répartition fonctionne, il faut avoir un nombre de points particulier, fonction du nombre de niveaux (ou découpages). Par exemple, pour 3 facteurs et 3 niveaux par facteurs, avec 27 points nous pouvons obtenir :

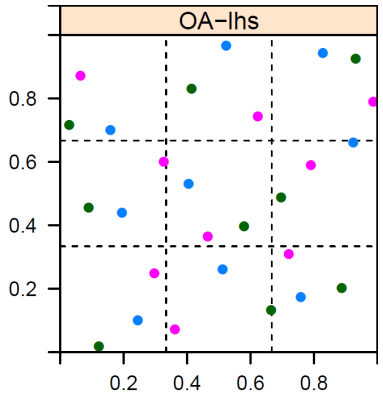


FIGURE 2.2 – Oalhs avec 3 facteurs pour 27 observations

- un plan orthogonal dans le sous espace de dimension "force" (paramètre à donner en argument à la fonction permettant de générer ces plans sous **R**).

Plan tms-net

Le plan tms-net est un cas particulier du plan OA-LHS avec une contrainte supplémentaire. En découpant à nouveau les colonnes, nous devons avoir un point par niveau dans chaque colonne. Sur le même exemple que précédemment cela donne :

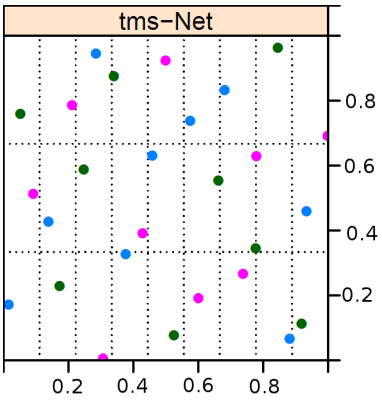


FIGURE 2.3 – Tms avec 3 facteurs pour 27 observations

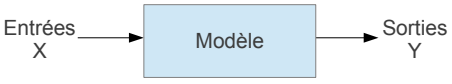
L'intérêt de ces plans réside dans la stratification supplémentaire par rapport au plan LHS de "base", présenté au début. Cela permet de garantir que chaque région de l'espace possède bien un point au sein du plan d'expérience, et ainsi la bonne répartition dans tout l'espace de définition.

2.3 Métamodélisation

La modélisation consiste à représenter ou reproduire un phénomène réel par un modèle mathématique. Cependant, dans la plupart des cas, le modèle obtenu est complexe et très coûteux en temps de calcul. Ainsi, afin d'analyser plus en détail les facteurs et sorties de ce modèle, les calculs nécessaires à l'obtention de résultats pertinents et à une bonne analyse peuvent être assez long.

Nous pouvons alors avoir recours à de la métamodélisation, c'est-à-dire à la modélisation du premier ou "gros" modèle. Nous obtenons un "modèle de modèle". Pour cela, un premier plan d'expérience est généré, puis les sorties du "gros" modèle sont simulés à partir de ce plan. Grâce à cet échantillon Y de valeurs du "gros" modèle, nous pouvons générer le métamodèle, plus simple et moins coûteux que le "gros" modèle. Ces étapes peuvent être schématisées de la façon suivante :

1. Réalisation du plan d'expérience : $X = \begin{bmatrix} x_{11} & \dots & x_{1d} \\ \vdots & \ddots & \vdots \\ x_{N1} & \dots & x_{Nd} \end{bmatrix}$ par une des méthodes vues précédemment.
2. A partir du modèle



nous obtenons $Y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$.

3. Grâce à X et Y et à une technique de métamodélisation, nous générons le nouveau métamodèle.
4. Enfin, à partir du métamodèle et d'un nouveau plan d'expérience X', tel que, $X' = \begin{bmatrix} x'_{11} & ... & x'_{1d} \\ \vdots & \ddots & \vdots \\ x'_{M1} & ... & x'_{Md} \end{bmatrix}$ avec $M \gg N$ (soit un nombre d'observations bien plus grand que celui nous ayant permis de simuler le métamodèle), les valeurs estimées de la sortie, nécessaires à une future analyse (de sensibilité, d'optimisation ...), sont générées :

Différentes techniques existent et la plus connue est la régression linéaire multiple.

2.3.1 Régression linéaire

La régression linéaire consiste à étudier l'influence des entrées X_i, $i \in [1...n]$, appelées variables explicatives, sur la sortie Y, variable à expliquer, en écrivant Y comme fonction linéaire des X_i. Cette influence est déterminée par le calcul d'un coefficient devant chaque fonction des X_i. Le modèle linéaire s'écrit :

$$Y = \sum \beta_I f_I(X) + \epsilon$$

où $I \subset \{1, ..., n\}$, les coefficients β_I sont les coefficients de la régression et les $f_I(X)$ des fonctions linéaires des X_i.

Par exemple, nous utilisons souvent la forme suivante :

$$Y = \beta_0 + \beta_1 X_1 + ... + \beta_n X_n + ... + \beta_{ij} X_i X_j + ... + \epsilon$$

où nous cherchons à déterminer tous les coefficients β_i, coefficient des effets propres de X_i sur Y, ainsi que β_{ij} qui est le coefficient de l'interaction (ordre 2) entre X_i et X_j et ainsi de suite pour les interactions d'ordre supérieur.

Sous **R**, la fonction *lm* permet d'estimer les coefficients de la régression linéaire. Pour cela, les variables explicatives du modèle de régression seront données en argument, ainsi les valeurs simulées de la variable à expliquer et le degré d'interaction.

2.3.2 Processus Gaussiens

Les processus gaussiens sont une autre façon de faire de la métamodélisation. Le principe consiste à considérer la valeur d'intérêt Y comme la réalisation d'un processus gaussien de moyenne et covariance connues. Son noyau de covariance est noté k, ou $k[Y(x), Y(x')]$ pour la covariance entre le processus au point x et au point x'. Dans toute la suite, Y sera le processus gaussien et y la sortie du vrai modèle.

Pour un plan d'expérience $(x_1, ..., x_n)$ et une entrée x, le vecteur $\begin{bmatrix} Y(x_1) \\ \vdots \\ Y(x_n) \\ Y(x) \end{bmatrix}$ est un vecteur gaussien où $Y(x_1) = y_1$ avec y_i la sortie du vrai modèle au point x_i $\forall i \in [1, ..., n]$. Nous cherchons à connaître la loi de $Y(x)|y_1, ..., y_n$ soit la loi de $Y(x)|Y(x_1), ..., Y(x_n)$.

La matrice de covariance du vecteur gaussien vu au-dessus s'écrit :

$$\Sigma = \left(\begin{array}{ccc|c} \mathrm{Cov}[Y(x_1), Y(x_1)] & ... & \mathrm{Cov}[Y(x_1), Y(x_n)] & \mathrm{Cov}[Y(x_1), Y(x)] \\ \vdots & \ddots & \vdots & \vdots \\ \mathrm{Cov}[Y(x_n), Y(x_1)] & ... & \mathrm{Cov}[Y(x_n), Y(x_n)] & \mathrm{Cov}[Y(x_n), Y(x)] \\ \hline \mathrm{Cov}[Y(x), Y(x_1)] & ... & \mathrm{Cov}[Y(x), Y(x_n)] & \mathrm{Cov}[Y(x), Y(x)] \end{array}\right)$$

Ceci peut aussi se décomposer de la façon suivante :

$$\Sigma = \left(\begin{array}{c|c} K_0 & k_n(x) \\ \hline k_n^T(x) & k(x,x) \end{array}\right)$$

où K_0 est la matrice de covariance de $\begin{bmatrix} Y(x_1) \\ \vdots \\ Y(x_n) \end{bmatrix}$, $k_n(x) = \begin{pmatrix} \mathrm{Cov}[Y(x_1), Y(x)] \\ \vdots \\ \mathrm{Cov}[Y(x_n), Y(x)] \end{pmatrix}$ et $k(x,x) = \mathrm{Cov}[Y(x), Y(x)]$.

Ainsi, l'espérance et la variance de $Y(x)|y_1, ..., y_n$, par hypothèse du vecteur gaussien précédent, vont avoir la forme :

$$\mathrm{E}[Y(x)|y_1, ..., y_n] = k_n^T(x) K_0^{-1} \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$
$$\mathrm{Var}[Y(x)|y_1, ..., y_n] = k(x,x) - k_n^T(x) K_0^{-1} k_n(x)$$

De même, la covariance entre $Y(x)$ et $Y(x')|y_1, ..., y_n$ s'écrit :

$$\mathrm{Cov}[Y(x), Y(x')|y_1, ..., y_n] = k(x,x') - k_n^T(x) K_0^{-1} k_n(x').$$

La covariance peut prendre plusieurs formes différentes, et une des formes possibles est :

$$\mathrm{Cov}[Y(x), Y(x')] = \sigma^2 \exp[-\tfrac{\|x-x'\|^2}{2\theta^2}]$$

où $\sigma^2 = k(x,x)$, appelée amplitude, est la variance du processus gaussien et θ , appelée portée, est homogène à une distance.
Cette covariance est de type stationnaire, car elle ne dépend de x et de x' qu'à travers la distance $\|x - x'\|$.

Ainsi, pour générer un processus gaussien, il faut choisir la forme de la covariance $\mathrm{Cov}[Y(x), Y(x')]$ car plusieurs formes existent, ainsi que différentes valeurs des paramètres σ^2 et θ :

- Pour le choix de la covariance, il faut qu'elle ne dépende pas de l'endroit où se situe x et qu'elle diminue quand la distance entre x et x' augmente. Lors des différentes simulations sous **R**, j'ai étudié deux covariances particulières :
  1. $\mathrm{Cov}[Y(x), Y(x')] = \left[1 + \sqrt{3}\frac{\|x-x'\|}{\theta}\right] \exp\left(-\sqrt{3}\frac{\|x-x'\|}{\theta}\right)$ nommé *matern3_2* (dérivable 1 fois).
  2. $\mathrm{Cov}[Y(x), Y(x')] = \left[1 + \sqrt{5}\frac{\|x-x'\|}{\theta} + \frac{5}{3}\left(\frac{\|x-x'\|}{\theta}\right)^2\right] \exp\left(-\sqrt{5}\frac{\|x-x'\|}{\theta}\right)$ nommé *matern5_2* (dérivable 2 fois).
- Pour le choix des paramètres σ^2 et θ, une estimation par maximum de vraisemblance est utilisée. Il faut savoir qu'il peut y avoir plusieurs termes de portée θ_1,...,θ_n quand les dimensions ne sont pas du même ordre de grandeur, mais qu'ils sont tous estimés par maximum de vraisemblance. Le problème pouvant survenir, en plus du fait que cette estimation est longue en temps de calcul, est qu'elle n'est pas très robuste. Ceci est une des failles des processus gaussiens.

Lorsque la métamodélisation du "gros" modèle par processus gaussien sera générée sur **R**, nous devrons aussi choisir un paramètre appelé la tendance. C'est la valeur autour de laquelle le processus va osciller. Lors de notre étude, nous allons étudier deux types de krigeage, ou tendance :
- krigeage ordinaire : la tendance est égale à une constante β_0.
- krigeage linéaire (sans interactions) : la tendance est par exemple égale en dimension 3 (entrées X_1, X_2 et X_3) à $X_1 + X_2 + X_3$.

Le package de **R** permettant de générer des processus gaussiens, puis de prédire la sortie du modèle, est *DiceKriging*. Pour cela, nous utilisons deux fonctions, dans l'ordre suivant :

1. *km()* dont les arguments sont :
   - la plan d'expérience X (plutôt de petite taille)
   - la réponse y du "gros" modèle aux entrées correspondant au plan d'expérience X
   - la tendance
   - la covariance
   - les bornes inférieure et supérieure pour le calcul de la covariance : la borne inférieure permet d'éviter les bruits blanc, et la borne supérieure permet d'éviter qu'il y ait trop de rigidité dans le modèle et donc des erreurs de stockage.
2. *predict()* : à partir du métamodèle généré avec *km()*, cette fonction permet de prédire la sortie du "gros" modèle pour une entrée x donnée

Chapitre 3

Description des travaux réalisés

Dans un premier temps, j'ai été amenée à maîtriser les éléments théoriques vus, dans le chapitre précédent, et ce, en comparant les différents résultats pouvant être obtenus à partir de différents tests effectués avec le logiciel libre **R**. J'ai notamment commencé à manipuler les fonctions des packages *lhs* et *randtoolbox* permettant de générer un grand nombre de plans d'expérience, ainsi que d'observer la répartition des points dans l'espace. J'ai aussi étudié en première approche les méthodes d'analyse de sensibilité de Sobol, FAST et Morris sur des modèles, ou fonctions, préexistants. J'ai aussi cherché à retrouver la forme théorique de ces modèles (connue) comme si elle était inconnue, en effectuant une planification d'expériences puis une métamodélisation par régression linéaire.

J'ai aussi découvert un modèle utilisé par les agronomes : le modèle Sunopt. C'est un modèle concernant la maturation d'une culture de tournesol, dont les deux sorties que j'ai pu observer sont le rendement et la teneur en huile des tournesols, à date de maturation. Les agronomes s'intéressent à l'analyse de sensibilité d'un tel modèle, afin de pouvoir améliorer, par exemple, le rendement d'une culture de tournesol. Cependant, ce modèle est très coûteux en temps de calcul. Nous ne pouvons donc pas effectuer un grand nombre de tests et nous avons besoin d'un "bon" plan d'expérience, déterminé en fonction de certains critères, afin de bien choisir les valeurs d'entrées permettant d'analyser la sortie de ce modèle. Il peut aussi avoir besoin d'être métamodélisé afin d'accélérer le temps de calcul des estimateurs des différents indices.

Avant de commencer l'analyse d'un tel modèle, nous devons bien choisir le plan d'expérience, ainsi que la méthode d'analyse de sensibilité, qui nous fournira les estimateurs désirés. Pour cela, nous devons mener une étude en amont, sur des modèles dont les résultats sont connus, afin d'analyser les estimateurs donnés selon le plan, et pouvoir choisir celui qui conviendra le mieux à un modèle donné.

Problématiques posées

Ainsi, les principales thématiques de mon stage ont été :

- Comparer la précision des estimateurs d'indice de sensibilité en fonction du plan d'expérience utilisé. J'ai ainsi développé une étude empirique basée sur un très grand nombre d'essais avec deux difficultés principales :
  1. Une première difficulté technique : j'ai dû développer un certain nombre de programmes, grâce au logiciel **R**, effectuant de nombreux calculs, et ainsi devant

être efficaces afin de minimiser le temps de calcul.

2. Un deuxième enjeu pour l'analyse des résultats : pour ce faire, j'ai pu comparer les résultats obtenus aux indices théoriques, en analysant le biais et la variance des différents estimateurs.

– Améliorer les indices de sensibilité donnés par la fonction *plmm* (que je présenterai plus tard). Pour répondre à cet objectif, nous avons développé des méthodes basées sur une analyse plus théorique et mathématique.

Fonctions test

Tout au long de ce stage, j'ai effectué de nombreuses simulations sur trois modèles dont je connaissais la forme théorique, ainsi que les indices de sensibilité :

– **exemple3.fun** : fonction, avec 3 facteurs d'entrée X_1, X_2 et X_3, qui possède une très grande régularité et dont la sortie est :

$$Y = 1 + X_1 - 0.1X_1^2 + 0.2X_1^3 + X_2 + 0.3X_2^2 - X_1X_2 - 2X_3 + 3X_2X_3^2$$

– **ishigami.fun** : fonction du package *sensitivity* du logiciel **R**, avec 3 facteurs d'entrées, dont la sortie est définie de la façon suivante :

$$Y = \sin(X_1) + 7\sin(X_2)^2 + 0.1X_3^4\sin(X_1) \quad \text{où} \quad X_j \sim U(-\pi, \pi)$$

– **sobol.fun** : fonction du package *sensitivity*, avec 8 facteurs d'entrées, dont la sortie est :

$$Y = \prod_{j=1}^{k} g_j(X_j) \quad \text{où} \quad g_j(X_j) = \frac{|X_j - 2| + a_j}{1 + a_j}$$

avec $a_j \geq 0$, $k = 8$, $a_j = \{0, 1, 4.5, 9, 99, 99, 99, 99\}$ et $X_j \sim U(0, 1)$.

Plans d'expérience comparés

Les plans d'expérience utilisés tout au long de ce stage ont été :

– plan de Monte-Carlo, *mc* dans mes scripts **R**, qui simule les points par loi $U(0, 1)$
– plans LHS, du package *lhs* :
  – fonction *randomLHS* simulant des plans lhs de base avec tirage aléatoire
  – fonction *improvedLHS* simulant des plans lhs optimisés, appelés *optimlhs* dans mes scripts
– plans oalhs et soalhs (oalhs avec brouillage)
– les suites quasi-aléatoires, du package *randtoolbox* :
  – suites de Sobol avec ou sans brouillage
  – suites de Halton
  – suites de Torus
– plans tms et stms (tms avec brouillage)

3.1 Une première approche de l'analyse de sensibilité

L'analyse de sensibilité permet de déterminer à quelles variables explicatives est sensible une variable réponse Y. Cependant, déterminer ces indices et les expliquer peut être assez complexe car les variations de la réponse d'un modèle ne sont pas dues qu'à une variable explicative, et l'influence d'une variable X_i peut intervenir dans les indices d'une autre variable explicative X_j. En effet, des phénomènes d'interaction et de confusion existent, pouvant perturber la bonne estimation et analyse des indices de sensibilité.

3.1.1 Interaction et Confusion

Interaction

L'interaction apparaît quand plusieurs variables explicatives agissent ensemble afin de faire varier la valeur de la variable réponse Y. Par exemple, pour un modèle à deux variables explicatives X_1 et X_2 et une sortie Y, on identifiera une interaction dès lors que l'effet de X_1 et X_2 sur Y ne sera pas additif. En effet, en effectuant la régression linéaire de Y sur X_1 et X_2, celle-ci sera différente de la somme des simples régressions de Y sur X_1 puis de Y sur X_2 si le modèle contient un terme multiplicatif $X_1 \times X_2$ par exemple. Il faut cependant ne pas confondre l'interaction de la corrélation ; deux variables peuvent être corrélées sans interagir et inversement. Un exemple d'interaction mais de non corrélation peut être en posant X_1 la vitesse d'un automobiliste, X_2 l'utilisation du téléphone pendant la conduite et Y le risque d'accident. Les deux variables ne sont pas corrélées (non liées) mais leur association augmente considérablement la variable réponse, plus que la simple somme des deux effets séparés sur le risque d'accident d'un automobiliste. Elles interagissent donc.

Lorsque deux variables interagissent, l'effet d'un facteur sur la sortie va dépendre de l'autre facteur, ce qui va donc influencer les indices de sensibilité de ces deux variables.

Confusion

La confusion peut exister à différents niveaux :

- **Confusion due à une corrélation** : lorsque 2 variables sont liées, l'indice de sensibilité dû à une des variables peut être mal estimé en raison de la confusion que la corrélation entraîne : l'influence de la deuxième variable sur la première sera prise en compte dans l'indice. Ceci peut aussi être le cas lorsqu'une variable "cachée", influençant la sortie, n'est pas prise en compte dans le modèle, et qu'elle est liée à un des facteurs du modèle. Par exemple, si on régresse chez une classe de primaires les fautes d'orthographe en fonction de la taille des pieds nous pouvons trouver une relation. Or, rien n'indique qu'il y en ait une. Cependant, cette relation vient d'une variable "cachée", l'âge, qui influence le nombre de fautes mais aussi sur la taille des pieds des enfants.

- **Confusion due au sur-ajustement** : lorsque nous n'avons pas assez de points afin d'effectuer l'analyse, nous pouvons trouver une relation qui est en fait inexistante ou qui est bien moins importante. Ceci est causé par l'analyse qui tente de trop expliquer certains effets ; par exemple l'effet principal d'un facteur peut être

confondu avec un autre effet principal ou un effet d'interaction. Pour éviter ce genre de confusion, il faut analyser le coefficient de détermination ajusté noté $R^2_{\text{ajusté}}$. Nous pouvons définir le coefficient de détermination R^2 comme un indice compris entre 0 et 1 permettant de savoir quelle est la qualité de la régression effectuée. Pour l'estimation d'une sortie y, ce coefficient est définit comme :

$$R^2 = \frac{\sum_{i=1}^{n}(\widehat{y_i}-\overline{y})^2}{\sum_{i=1}^{n}(y_i-\widehat{y_i})^2}$$

où $\widehat{y_i}$ est la valeur estimée de y_i et $\overline{y}$ la moyenne des y_i.
Ce coefficient mesure ainsi la part de variance expliquée par le modèle par rapport à la variance totale. Ainsi, plus il est proche de 1, plus le modèle explique la variance de la sortie. Cependant, cet estimateur ne prend pas en compte le nombre d'observations. Ainsi, le coefficient de détermination ajusté est défini par :

$$R^2_{\text{ajusté}} = 1 - \frac{(1-R^2)(n-1)}{n-k-1}$$

où n est le nombre d'observations et k le nombre de facteurs.
Cet indice prend donc en compte le nombre d'observations utilisé pour la régression et peut ainsi indiquer, s'il est faible, un problème lié à un faible nombre d'observations.

– **Confusion d'effets** : quand le plan d'expérience n'est pas très bon, il ne permet pas d'identifier certains effets. C'est la confusion d'effets.
Nous utilisons en général des plans fractionnaires car les plans complets (plans permettant tous les croisements des valeurs du domaine de définition des facteurs) ne sont pas réalisables. Ces plans ne sont constitués que d'une partie des expériences du domaine de définition mais ont des propriétés impliquant que les effets principaux et les effets d'interaction faible ne sont pas confondus. Ceci repose sur l'hypothèse que les termes d'interaction d'ordre élevé sont négligeables. Cependant, lorsque le modèle possède des termes d'interaction d'ordre élevé, ces derniers peuvent être confondus avec d'autres effets, et les coefficients de régression de ces effets sont augmentés. Par exemple, au lieu d'avoir dans la régression le coefficient β_1 qui correspond juste à l'effet de X_1, soit $\beta_1 X_1$, nous allons en fait régresser par exemple $\beta_1(X_1 + X_1^3 X_2^2 X_3)$ et l'effet principal de X_1 est confondu avec l'interaction entre X_1^3, X_2^2 et X_3.
Ceci est le cas lorsque, par exemple, les observations dont le facteur X_1 est égal au niveau i sont les mêmes que celles dont le facteur X_2 est de niveau j. Les effets dus à ces deux niveaux peuvent être confondus (nous ne pouvons pas les distinguer ou les identifier). En général, les plans orthogonaux permettent d'éviter ce type de confusion.

Nous avons commencé à trouver un moyen de différencier cette confusion, plutôt de type confusion d'effets, de l'interaction dans les indices, notamment dans ceux que la fonction *plmm* nous fournit.

3.1.2 La fonction *plmm*

Cette fonction a été créée par mon maître de stage. Elle permet de faire de la régression linéaire à partir du modèle de départ, puis d'estimer les indices de sensibilité à partir de cette régression. La régression peut être faite du degré 1 au degré 4, c'est-à-dire qu'elle peut comprendre jusqu'à des interactions de degré 4, si nous le précisons en argument. Elle permet de connaître quatre estimateurs d'indices :

- **Alone** : correspond à ce qu'explique la variable seule, sans prendre en compte les termes d'interaction. Cependant, cet indice peut être augmenté par de la confusion, par exemple si cette variable même seule dépend quand même d'une autre variable.
- **Specific** : correspond aux seules parties expliquées par la variable dans la sortie, lorsque les autres variables sont tenues constantes par rapport à la variable explicative. Quand il n'y a pas de confusion entre les variables, cette valeur est égale à la valeur de l'indice précédent.
- **Total** : indice correspondant à tout ce que peut expliquer la variable, qu'elle soit seule ou en interaction avec d'autres variables.
- **Interaction** : partie correspondant à la proportion d'explication provenant de la variable quand elle interagit avec d'autres variables. Cet indice peut être négatif, quand il correspond à de la confusion.

Par exemple, pour la fonction *exemple3.fun* et un plan randomLHS avec 1000 observations, nous obtenons le résumé suivant, par *plmm* de degré 2 :

```
Percentage of factor contributions
 Polynomial Linear MetaModel of degree 2

Y ~ polym(V1, V2, V3, degree = 2)

R-squared (percentage) of the complete metamodel: 98.96259

Initial  standard error of 0.6388513 with  degrees of freedom

Residual standard error of 0.06536417 with 990 degrees of freedom

       Alone  Specific    Total Interaction
V1  6.872213  6.872213  8.78789    1.915677
V2 66.708606 66.708606 83.74862   17.040010
V3  8.626712  8.626712 23.48271   14.855999
```

Listing 3.1 – Résumé de l'analyse de sensibilité par plmm en degré 2

En utilisant la fonction *plot* sur le résultat du plmm, on obtient le graphique suivant :

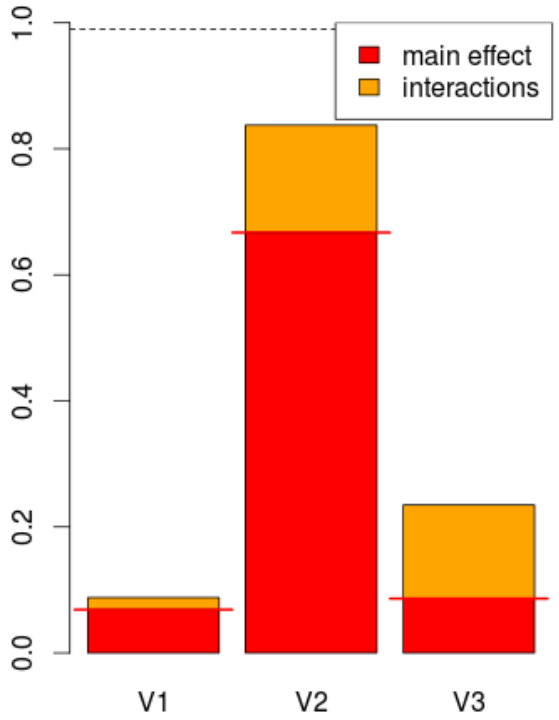


FIGURE 3.1 – Indices de sensibilité par plmm en degré 2

Nous pouvons observer en jaune la partie **Interaction** et en rouge la partie **Specific**. Le trait rouge correspond à la limite de la partie **Alone**. Lorsqu'il ne délimite pas la partie jaune de la rouge, il y a alors de la confusion.

3.1.3 Amélioration de la fonction *plmm*

Nous avons cherché différents moyens afin de différencier cette interaction des parties correspondant à de la confusion pour mieux analyser les indices de sensibilité générés par la fonction *plmm*. Pour cela, nous nous sommes particulièrement appuyés sur un document de l'université de Montréal [**?**], définissant 3 indicateurs différents, permettant de différencier les niveaux d'influence d'une variable explicative sur une variable réponse :

- **Contribution** d'une variable explicative X_i sur la valeur d'une variable réponse Y : elle peut être positive où négative, et est égale à $a'_i r_{yX_i}$ où r_{yX_i} est le coefficient de corrélation de Pearson et a'_i est le coefficient de régression centré-réduit de X_i. La somme des contributions des variables doit être égale au coefficient de régression multiple, R^2 qui permet de savoir quelle part de la variance de la sortie Y est expliquée par le modèle, soit par toutes les variables explicatives. Ainsi,

$$R^2 = \sum_{i=1}^{d} a'_i r_{yX_i} \text{ avec } d \text{ variables explicatives, ou facteurs.}$$

- **Fraction** : égale à la partie Specific générée par la fonction *plmm*, elle mesure la partie explicative dû à une variable X_i dans les variations de la sortie Y, quand toutes les autres variables sont maintenues constantes par rapport à X_i, c'est-à-dire quand toutes les influences des autres variables sont enlevées de X_i. Pour cela, on régresse en premier X_i par rapport à toutes les autres variables. Puis on prend le résidu de cette première régression, et on régresse Y par rapport à ce résidu. L'influence des autres variables est alors bien enlevée de X_i.
- **Coefficient de corrélation multiple** : il permet de savoir quelle part de la variance de Y est expliquée par un facteur X_i quand toutes les influences des autres variables ont été enlevées de X_i mais aussi de Y. Pour cela on régresse tout d'abord X_i en fonction des autres facteurs, puis Y en fonction de tous les facteurs sauf X_i. Enfin, on régresse le résidus de la régression de Y par le résidu de la régression de X_i. Pour deux variables explicatives X_1 et X_2, nous pouvons aussi l'obtenir par la formule :

$$r_{Y,X_1|X_2} = \frac{r_{YX_1} - r_{YX_2} r_{X_1X_2}}{\sqrt{(1-r^2_{YX_2})(1-r^2_{X_1X_2})}}$$

  où r_{YX_1} est le coefficient de corrélation de Pearson entre Y et X_1. Ceci est aussi égal, en degré 1, à : $\frac{[a]}{[a]+[d]}$ où $[a]$ est la fraction de X_1 et $[d] = 1-R^2$ est la partie résiduelle.

En effectuant quelques tests avec la fonction *plmm*, nous avons pu vérifier cette égalité en degré 1. Cependant, avec un degré supérieur à 1, les valeurs sont différentes entre la division avec les fractions, ou indices "Specific", et la régression du résidu de Y sur le résidu de X_i vu au-dessus. En effet, pour cette deuxième valeur, en dimension 3, en degré 3, nous effectuons par exemple la commande suivante :

```
lm ( resid( lm(Y ~ polym(X2,X3, degree=3))) ~ resid(lm(poly(X1,degree=3) ~ polym(X2,X3,
    degree=3))))
```

Listing 3.2 – Commande **R** pour générer l'indice *IntR2p* de X_1 en degre 2

Cette valeur est différente de la division de l'indice "Specific" de X_1 par la somme de cet indice plus la valeur $(1 - R^2)$ obtenu par la fonction *plmm*. En effet, l'indice obtenu par régression prend en compte les interactions d'ordres inférieurs ou égal à 3 entre X_2 et X_3 et enlève leurs influences sur X_1 et Y, ce que ne fait pas le premier indice. Ceci peut permettre une première connaissance et analyse des interactions et de leur influence sur Y.

Pour analyser dans un second temps ces indices, la fonction *plmm* a été modifiée pour y ajouter les deux indices suivants :

- **Partiel** : coefficient de corrélation multiple
- **IntR2p** : indice obtenu par la régression du résidu de Y, obtenu par régression polynomiale (de degré égal à celui donné en argument à la fonction *plmm*) de Y sur toutes les variables sauf X_i, par le résidu de X_i, obtenu par régression polynomiale de X_i sur toutes les autres variables.

3.2 Effets des différents plans d'expérience sur la qualité d'estimation des indices de sensibilité

Nous nous demandions si un des plans d'expérience vus précédemment donnerait de meilleurs résultats que les autres, en terme d'estimation des indices sensibilité notamment. Nous nous sommes aussi interrogés sur la métamodélisation par processus gaussiens : nous voulions savoir si un des plans d'expérience permettrait d'avoir un meilleur modèle, plus proche de la réalité, afin d'obtenir de meilleurs estimateurs des indices de sensibilité, par la suite. Pour cela, j'ai notamment comparé la précision des résultats selon les plans, grâce à de nombreuses sorties graphiques tels que les boxplots par exemple.

3.2.1 Protocole expérimental

Afin de comparer ces plans d'expérience, j'ai implémenté un script avec le logiciel **R**, prenant en argument le modèle testé, le nombre d'observations, les plans testés ainsi que des booléens permettant de choisir les méthodes d'analyse de sensibilité pour lesquelles nous souhaitons comparer les résultats. Cette fonction est présentée en annexe. L'en-tête de la fonction est :

```
my.samo.model <- function(model, INFOmodel = NULL, p = 3, d = 3, force = 3, N = 27,
                          plans=c("sobol","sobol.scr","halton","torus","oalhs",
                                  "soalhs","randomlhs","optimlhs","mc") ,
                          amorce = 2013, replhs=99, outplan = FALSE,
                          replicate = 1, repli.N=N/replicate,
                          outplmm=TRUE, outsob=TRUE, outPG=TRUE)
```

Listing 3.3 – En-tête de la fonction my.samo.model

Les paramètres "force" et "p" (nombre de niveaux par facteur) sont exclusivement utilisés pour les plans oalhs et tms-net.

Pour générer des plans tms, les paramètres "force" et "p" interviennent dans la division de chaque dimension. Ainsi, pour un paramètre "force" égal à 3 et un paramètre "p" égal à 3 aussi, dans $\mathbb{R}^2$, les ordonnées seront divisées selon les 3 niveaux, et les abscisses en 9, afin d'avoir une division en $p^{force} = 27$. Le nombre de points doit alors être soit 27, soit un multiple de 27. Dans ce dernier cas, le paramètre *replicate* sera égal à ce multiple, et il y aura non pas un point par case mais ce même nombre *replicate* de points par case.

J'ai effectué les premières simulations sur les trois modèles présentés au début : **exemple3.fun**, **ishigami.fun** et **sobol.fun**. Pour chaque modèle, j'ai effectué les simulations avec les paramètres suivant :

- **exemple3.fun** et **ishigami.fun** : simulations pour 27, 81, 125 et 243 observations pour tous les plans vu précédemment. Pour les plans *tms*, j'ai testé avec différentes forces par nombre d'observations :
  - N=27 avec $force = 3$ et $p = 3$
  - N=81 avec $force = 3$ ou $force = 4$ et $p = 3$
  - N=125 avec $force = 3$ et $p = 5$
  - N=243 avec $force = 3$, $force = 4$ ou $force = 5$ et $p = 3$

– **sobol.fun** : simulations pour 81, 162 et 243 observations pour tous les plans sauf les plans tms-net.

Concernant la méthode d'estimation de Sobol, plusieurs fonctions ont été programmées sous **R** :

– sobol2002
– sobol2007
– soboljansen ...

Les différences entre ces fonctions résident dans la formation des différentes matrices à partir des deux matrices données en paramètre. Les deux premières fonctions étant très similaires, nous avons effectué les tests à partir des deux fonctions *sobol2002* et *soboljansen*.

Pour la méthode d'analyse par processus gaussiens, les estimations à partir du métamodèle ont toujours été effectuée avec les fonctions *sobol2002* et *soboljansen*.

Les deux tableaux suivants résument tous les tests effectués, avec les méthodes et les différents plans testés :

Méthodes d'estimation		
sobol2002		
soboljansen		
sobol2002 .PG	matern3_2	krigeage ordinaire
		krigeage universel
	matern5_2	krigeage ordinaire
		krigeage universel
soboljansen .PG	matern3_2	krigeage ordinaire
		krigeage universel
	matern5_2	krigeage ordinaire
		krigeage universel
plmm	degre1	
	degre2	
	degre3	
	degre4	

FIGURE 3.2 – Tableau des différentes méthodes d'analyse de sensibilité utilisées

			Plans											
			Sobol	Sobol .scr	Hal ton	Torus	oalhs	soalhs	lhs	optim lhs	mc	tms & stms		
												f=3	f=4	f=5
M O D E L E	ex3 .fun	N=27	√	√	√	√	√	√	√	√	√	√	×	×
		N=81	√	√	√	√	√	√	√	√	√	√	√	×
		N=125	√	√	√	√	√	√	√	√	√	√	×	×
		N=243	√	√	√	√	√	√	√	√	√	√	√	√
	ishi gami .fun	N=27	√	√	√	√	√	√	√	√	√	√	×	×
		N=81	√	√	√	√	√	√	√	√	√	√	√	×
		N=125	√	√	√	√	√	√	√	√	√	√	×	×
		N=243	√	√	√	√	√	√	√	√	√	√	√	√
	sobol .fun	N=81	√	√	√	√	√	√	√	√	√	×	×	×
		N=162	√	√	√	√	√	√	√	√	√	×	×	×
		N=243	√	√	√	√	√	√	√	√	√	×	×	×

FIGURE 3.3 – Résumé des différents tests effectués

Nous avons généré un certain nombre de fois la même expérience (même famille de plan, même fonction, même nombre d'observations...) afin de pouvoir analyser la moyenne et la variance d'un estimateur pour un test donné, en effectuant plusieurs fois ce même test. En effet, au sein d'une même famille de plans, les solutions possibles pour générer un plan sont multiples. Il fallait alors effectuer plusieurs simulations pour le même test, afin d'analyser les divers résultats que nous obtenions selon les plans d'une même famille. L'analyse a ainsi été faite sur 101 répétitions, pour chaque méthode et chaque plan. En effet, nous avons choisi ce nombre de répétitions car analyser la moyenne et la variance des estimateurs d'une même valeur à partir de 101 estimations nous semblait suffisant. De plus, nous avons choisi un nombre impair afin que la médiane observée soit exactement une des estimations calculées. Nous avons donc obtenu 101 estimations différentes pour une même valeur à estimer.

Ainsi, le nombre d'expériences effectuées par modèle est égal à :

$$\text{nombre de plans} \times \text{les différents nombres d'observations} \times \text{nombre de méthodes} \times \text{nombre de répétitions}$$
$$= \begin{cases} 15 \times 4 \times 14 \times 101 = 84840 & \text{pour exemple3.fun et ishigami.fun} \\ 9 \times 3 \times 14 \times 101 = 38178 & \text{pour sobol.fun} \end{cases}$$

Nous avons donc fait au total $84840 \times 2 + 38178 = 207858$ expériences. Pour les 84840×2 premières expériences, nous avons obtenu 6 estimateurs d'indices de sensibilité à comparer, plus l'estimateur de la variance, et ce par expérience réalisée. Pour les 38178 expériences réalisées pour la fonction sobol.fun, nous avons 16 estimateurs d'indices, plus celui de la variance, à analyser par expérience.

Nous devions générer un très grand nombre d'expériences et estimer beaucoup d'indicateurs différents. J'ai donc réfléchi à une façon de générer ces expériences de façon optimale et à une méthode afin de pouvoir analyser efficacement tous les estimateurs que j'allais obtenir. Pour cela, j'ai procédé de manière progressive, et j'ai conçu plusieurs outils graphiques d'analyse.

3.2.2 Développement des différents programmes

Fonction my.samo.model.R

J'ai commencé à développer cette fonction progressivement, en faisant une première version d'essai. Après avoir pallié à tous les problèmes liés à la programmation de cette première version de la fonction, je l'ai optimisée, en la testant sur 3 ou 4 répétitions seulement. En effet, la première version de cette fonction effectuait les tests pour tous les plans sauf tms, et si nous voulions refaire certains tests sur quelques plans, nous ne pouvions pas.

J'ai alors développé une nouvelle version avec les plans que nous souhaitions comparer donnés en arguments. J'ai aussi ajouté plusieurs booléens en paramètres permettant de choisir si nous voulions effectuer des simulations avec la méthode plmm, avec les deux fonctions générant les estimateurs par méthode de Sobol, ou en utilisant les processus gaussiens. Afin de permettre à l'utilisateur de choisir exactement les tests qu'il voudrait faire, j'ai donc changé ma fonction en générant les estimateurs dans des boucles génériques.

Les programmes étant très coûteux en temps de calcul, j'ai aussi dû réfléchir au stockage de certains outils dans la fonction, dans un but d'optimisation.

J'ai ainsi pu apprendre comment développer un programme en plusieurs étapes. J'ai appris comment localiser et détecter les erreurs de programmation. Enfin, j'ai aussi découvert comment développer des méthodes permettant de vérifier le bon fonctionnement d'un programme, sans erreurs de stockage (vérification que les stockages sont faits aux bons endroits).

Une fois la dernière version de la fonction **my.samo.model** terminée (version jointe en annexes), j'ai effectué les différents tests vus dans la partie précédente. Les 101 répétitions, pour un modèle et un nombre d'observations donnés, prenaient entre un à deux jours de temps d'exécution chacun. En attendant les résultats finaux, j'ai alors commencé à réfléchir à une façon d'organiser et d'analyser le nombre important d'estimateurs que j'allais obtenir. Pour cela, j'ai développé des scripts permettant leur analyse sous forme graphique.

Fonctions d'affichage sous forme graphique des résultats

Suite à l'obtention des différents résultats, nous avons alors comparé trois types d'estimateurs différents :

- les estimateurs des indices de sensibilité du premier ordre et totaux, pour chacune des variables. Pour cela, j'ai tracé sous forme de boxplots ces estimateurs afin d'analyser leur biais et variance selon le plan, la méthode utilisée et les différentes caractéristiques vues précédemment.
- l'estimateur de la variance pour chaque méthode
- le RMSE (Root-mean-square error) de chacun des estimateurs des indices. Le RMSE d'une suite de valeurs estimées $\widehat{\theta}_t, t \in [1, .., n]$ est défini comme :

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(\theta - \widehat{\theta}_t)^2}{n}} \text{ où } \theta \text{ est la vrai valeur}$$

Dans le cas de notre étude, θ est égal à l'indice de sensibilité étudié.

Cette statistique permet de visualiser d'une deuxième manière l'écart des estimateurs à la vrai valeur de l'indice.

Pour ce faire, j'ai alors développé deux programmes :

- Un premier script qui permet de comparer la qualité de l'estimation des indices de sensibilité (ou de la variance par le même principe) en traçant les estimateurs sous forme de boxplot.

- Un deuxième script qui permet de faire les mêmes comparaisons mais sous une forme plus synthétique grâce à des diagrammes en barre représentant les RMSE des indices de sensibilité à partir d'une autre fonction que j'ai créée, qui calcule le RMSE de l'estimation d'un indice de sensibilité donné.

Pour implémenter ces deux fonctions, j'ai notamment appris à utiliser la librairie *Lattice*, permettant de tracer plusieurs mêmes types de graphiques sur une page, ceci étant très utile pour effectuer plusieurs comparaisons. J'ai procédé de la même manière que pour développer la fonction d'estimation des différents indices : j'ai tout d'abord implémenté une première version de chacun des deux outils graphiques, puis j'ai amélioré mes deux programmes afin qu'ils gagnent en rapidité et efficacité. Les versions finales de ces deux fonctions, **bwplot.my.samo.model.R** et **my.barchart.rmse.R** sont disponibles en annexes.

Après la première phase d'analyse que j'exposerai dans la partie suivante, j'ai à nouveau implémenté, sur le même procédé que les deux fonctions énoncées au-dessus, une nouvelle façon de présenter graphiquement les estimateurs des indices de sensibilité sous forme de boxplots, ainsi que les RMSE mais cette fois-ci sous forme de graphiques appelés "dotplot".

3.2.3 Nature des analyses effectuées

Estimateurs des indices de sensibilité

Nous avons obtenu un très grand nombre d'informations et beaucoup de sorties graphiques représentant les différents estimateurs des indices de sensibilité. Afin de simplifier l'interprétation de ces résultats, ces estimateurs ont été tracés sous forme de boxplots. Voici un exemple d'un de ces boxplots pour la fonction ishigami.fun, avec 125 observations et par la méthode d'estimation de Sobol, codée grâce à la fonction *soboljansen* :

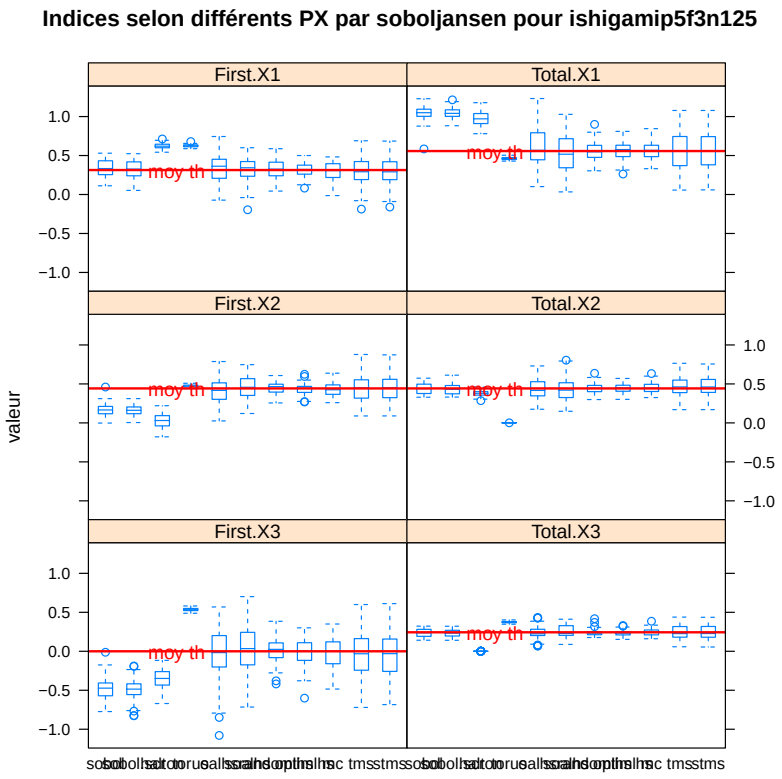


FIGURE 3.4 – Estimateurs des indices de sensibilité du modèle ishigami.fun, avec 125 observations et générés grâce à *soboljansen*

Nous pouvons voir la répartition des estimateurs des 6 indices de sensibilité de cette fonction (un indice principal et un indice total pour chacune des 3 variables). La valeur théorique de ces indices est tracée en rouge. Nous pouvons donc observer l'écart à la moyenne de ces estimateurs, soit leur biais. De plus, grâce à ces graphiques nous pouvons aussi voir les estimateurs les plus précis, soit ceux ayant la plus petite variance, c'est-à-dire les boîtes les plus petites. Les résultats sont tracés par estimateur de l'indice (par exemple First.X1 représente l'estimateur de l'indice principal de X_1) et chaque boxplot est spécifique à un plan donné. Les boîtes sont tracées en fonction des plans selon l'ordre suivant : suites de Sobol sans puis avec brouillage, suites de Halton puis de Torus, plan OA-LHS sans et avec brouillage, plan LHS de "base", optimlhs, plan de Monte-Carlo et enfin tms sans puis avec brouillage.

Pour cet exemple, nous pouvons remarquer que certains des estimateurs sont négatifs (alors qu'ils sont compris, théoriquement, entre 0 et 1). Ceci vient du fait que nous n'avons que 125 observations, alors que pour la méthode d'estimation de Sobol nous avons vu qu'il faut un nombre important d'observations afin d'obtenir des résultats précis.

Nous pouvons aussi remarquer que les indices estimés avec les suites quasi-aléatoires sont fortement biaisés (par exemple, l'indice total de X_1 appelé Total.X1), mais que leur variance est cependant très faible. Ceci est problématique car dans le cas de l'indice principal de X_1, ces suites donnent forcément un résultat erroné.

RMSE des indices de sensibilité

Nous nous sommes aussi intéressés au RMSE des estimateurs vus précédemment. Les résultats graphiques obtenus sont représentés sous forme de diagrammes en barre. Par exemple pour la fonction exemple3.fun avec 81 observations et une estimation avec la fonction plmm de degré 3, nous obtenons :

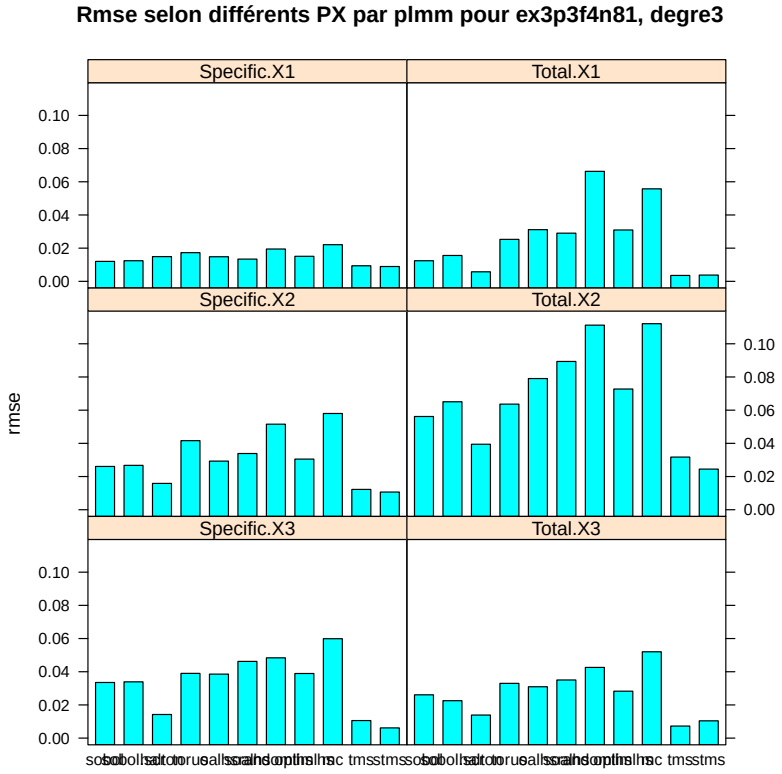


FIGURE 3.5 – RMSE des estimateurs des indices du modèle exemple3.fun, avec 81 observations et générés grâce à *plmm* de degré 3

Nous pouvons voir en bleu les valeurs des différents RMSE pour l'estimation de chaque indice en fonction du plan d'expérience utilisé. Par exemple, pour l'indice total de X_1 nous pouvons voir que les RMSE générés suite à l'utilisation de plans tms (les deux dernières barres) sont nettement meilleurs que ceux obtenu par plans Monte-Carlo (entre 0.05 et 0.06) ou par plans LHS de "base" ($RMSE \simeq 0.6$). Ainsi, nous avons directement une information sur la précision des différents estimateurs. En effet, si l'estimateur est biaisé ou si sa variance est grande, son écart à la valeur théorique sera comptabilisé dans le RMSE.

Estimateur de la variance

Le dernier estimateur étudié est celui de la variance du modèle testé. En effet, pour les fonctions sobol.fun et ishigami.fun, nous avons pu trouver les valeurs théoriques de cette variance. Concernant la variance du modèle exemple3.fun, j'ai effectué le calcul théorique de cette variance et j'ai obtenu 0.4057. J'ai ainsi pu observer que les calculs de variance

devenaient complexes lorsque nous souhaitons obtenir la valeur théorique. Pour remédier à ce problème, nous aurions aussi pu simuler plusieurs calculs de variance de ce modèle avec un très grand nombre de points et en prendre la moyenne. Les estimateurs de cette variance ont été tracés sous forme de boxplot, de la même manière que ceux des indices de sensibilité. Pour le modèle exemple3.fun avec 243 observations et la méthode plmm, nous obtenons :

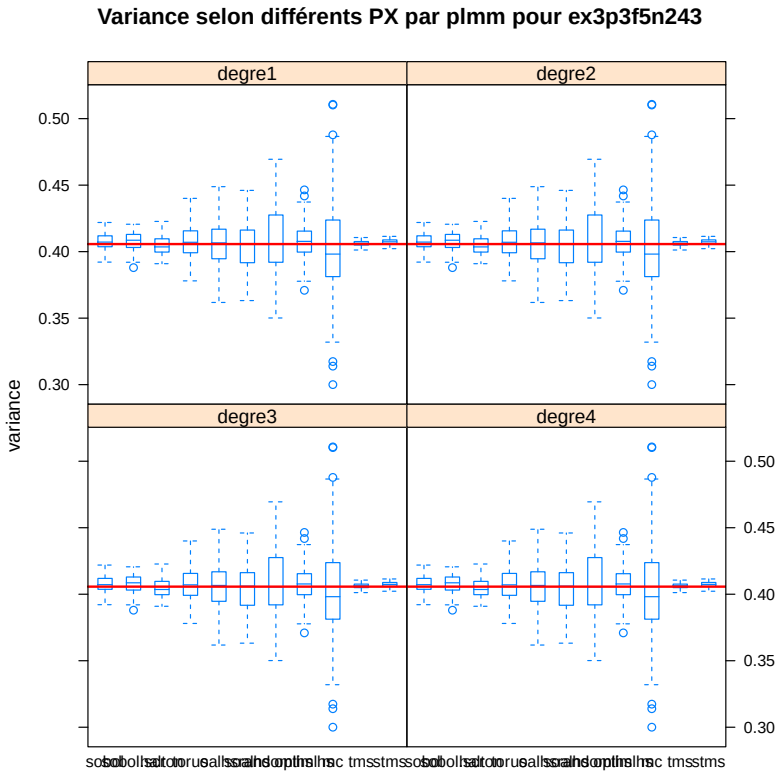


FIGURE 3.6 – Estimateurs de la variance du modèle exemple3.fun, avec 243 observations et générés grâce à *plmm*

Nous pouvons ici voir qu'en terme d'estimation de la variance, les plans tms sont bien meilleurs que les autres. Les boîtes étant très petites et le milieu de la boîte étant au même niveau que la valeur théorique, l'estimateur obtenu par plans tms est de très petite variance et non biaisé. Nous pouvons notamment le comparer à ceux des plans de Monte-Carlo qui sont bien plus variables et donc de moins bonnes performances.

3.2.4 Premières conclusions

Dans un premier temps, nous avons fait un tri de l'information, les graphiques étant riches en information, ils étaient aussi compliqués à analyser. Nous avons sélectionné les plans qui sont les plus intéressants à analyser et nous avons réorganisé l'information de manière plus simple, afin de simplifier les analyses. Nous avons remarqué que les processus gaussiens fournissent de très bonnes estimations des indices de sensibilité, c'est-à-dire avec peu de biais et de variance. Ceci est vrai quel que soit le plan d'expérience utilisé pour la métamodélisation. Par exemple, pour ishigami.fun avec 125 observations, nous obtenons les graphiques suivants :

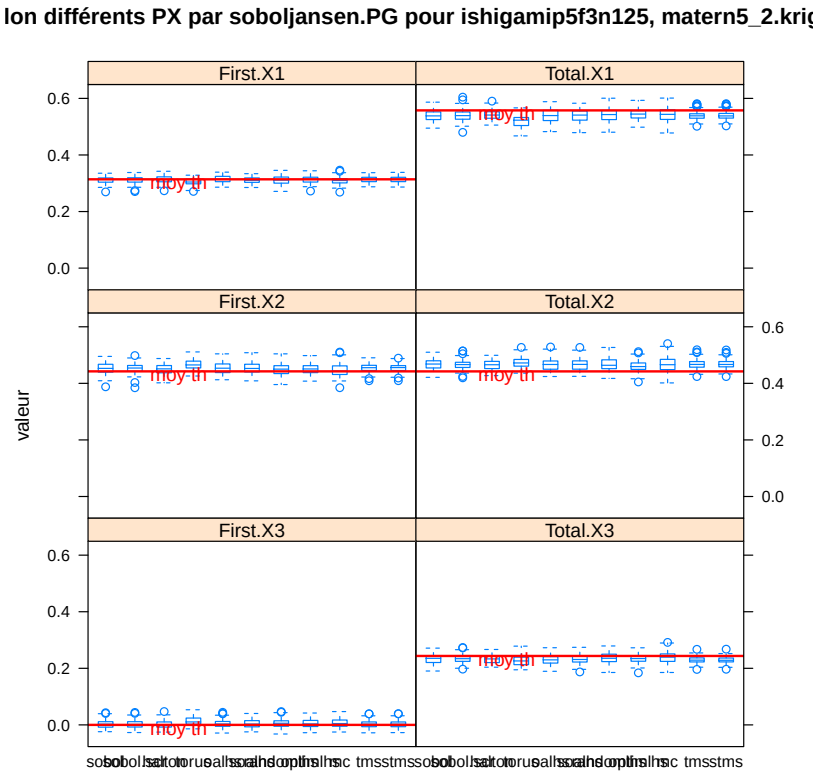


FIGURE 3.7 – Estimateurs des indices de sensibilité du modèle ishigami.fun, avec 125 observations et générés suite à une métamodélisation par processus gaussiens

La valeur théorique des estimateurs des indices principaux est connue. Celle des indices totaux a été estimée sous **R**. Les résultats étant très similaires, nous ne pouvons que conclure que cette méthode est très efficace et très précise quel que soit le plan choisi pour effectuer cette métamodélisation. De plus, les résultats sont aussi semblables entre les différents tests utilisant différentes covariances ou différentes tendances, et entre les méthodes utilisant *sobol2002* ou *soboljansen*. Pour une future analyse, nous n'allons conserver qu'un seul des huit cas : celui avec le paramètre de covariance égal à *matern5_2*, le krigeage linéaire et utilisant la fonction *sobol2002*.

Concernant la méthode d'estimation de Sobol, nous avons fait le tri dans les plans afin de diminuer la quantité d'information présente au sein de chacun des graphiques :

- les suites de Sobol avec ou sans brouillage donnent des résultats similaires, nous les enlevons des graphiques simplifiés
- les estimateurs obtenus à partir de suites de Torus sont généralement mauvais, nous les enlevons donc de l'analyse graphique qui suivra
- les résultats entre OA-LHS avec ou sans brouillage sont similaires, de même que ceux des tms avec ou sans brouillage. Nous conservons donc les résultats des plans OA-LHS sans brouillage et ceux des tms avec brouillage.

Les résultats obtenus par *plmm* en degré 1 sont souvent assez mauvais, nous ne conservons que les résultats selon le degré 2 et le degré 3.

3.2.5 Analyses finales

Dans un deuxième temps, nous avons réorganisé les résultats, toujours sous forme de boxplots mais présentés différemment afin de pouvoir les comparer plus en détail.

Réorganisation graphique des estimateurs des indices de sensibilité

Nous avons ainsi tracé pour un même indice, la répartition des estimations de cet indice en fonction des différents nombres d'observations et selon 9 plans, en comptant les 3 forces possibles pour les plans tms comme différents plans. Par exemple, nous obtenons pour l'indice principal de X_1 du modèle exemple3.fun estimé avec la fonction sobol2002 le graphique suivant :

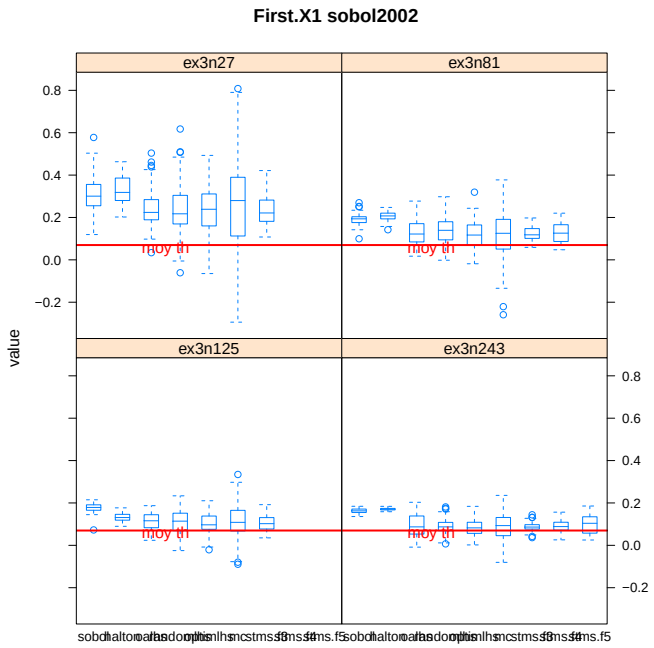


FIGURE 3.8 – Boxplot de l'indice principal de X_1 du modèle exemple3.fun, générés par la fonction *sobol2002*

Les plans d'expérience sont classés de gauche à droite dans l'ordre suivant : Sobol,

Halton, OA-LHS, LHS de “base”, optimLHS, Monte-Carlo (MC), tms avec brouillage force 3 puis force 4 et enfin force 5.

Pour N=27 et N=125, nous n’avons que 7 plans à comparer, les résultats pour tms n’ayant été générés qu’avec une force égale à 3. Avec 81 observations, nous avons 8 plans à comparer puisque les résultats à partir de plans tms avec force 4 ont été générés, et avec 243 observations nous pouvons analyser les effets des 9 plans. Ce graphique s’analyse de la façon suivante :

- Les suites quasi-aléatoires sont toujours biaisées et donnent des résultats toujours erronés en raison de leur biais et de leur faible variance.
- Avec N=243, nous voyons que les tms avec une force égale à 3 sont meilleurs que ceux avec une force égale à 4 ou 5.
- Les boîtes dont la médiane est égale à la moyenne théorique et de petite taille (donc de petite variance) sont celles générées par les plans tms force 3 et optimlhs.

Analyse des estimateurs des indices du modèle exemple3.fun avec plmm en degré 3

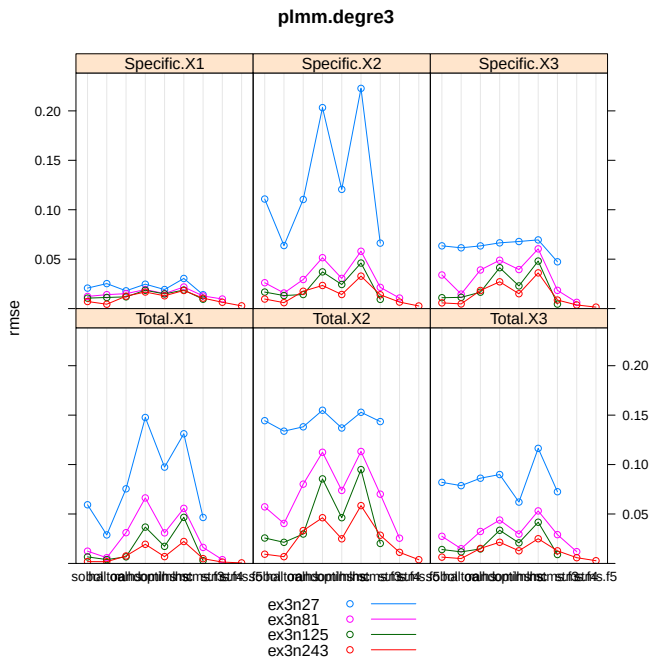


FIGURE 3.9 – RMSE des estimateurs des indices, modèle exemple3.fun, générés grâce à *plmm* de degré 3

Ce graphique représente les RMSE de chaque indice de sensibilité, en fonction du plan d’expérience utilisé (en abscisses) et selon la taille du plan considéré (pour changer de la représentation graphique vue précédemment). Comme dans l’exemple précédent, les plans d’expérience sont classés dans l’ordre suivant (de gauche à droite) : Sobol, Halton, OA-LHS, LHS de “base”, optimLHS, MC, tms avec brouillage force 3 puis force 4 et enfin force 5.

Pour l’estimateur de l’indice principal de X_2 (nommé Specific.X2 dans les

représentations graphiques), nous voyons que le RMSE est mauvais pour N=27 et qu'il diminue considérablement en passant à 81 observations quel que soit le plan. Nous pouvons observer plusieurs tendances :

- Les plus mauvais résultats sont obtenus pour les plans de Monte-Carlo et les plans LHS de "base", quel que soit le nombre d'observations ($RMSE \geq 0.05$ pour 81 observations).
- Les plans optimLHS et OA-LHS sont meilleurs que les précédents, mais moins que les suites quasi-aléatoires de Sobol et de Halton ($0.03 \leq RMSE \leq .04$ pour 81 observations).
- Ce sont les plans tms qui donnent les RMSE les plus petits. Notamment, le plan tms avec force 5 apparaît le meilleur avec cette statistique. En effet, avec 81 observations, le $RMSE$ est inférieur à 0.01 pour le plan tms (force=4) et avec 243 observation le tms, force 5, fournit un $RMSE$ quasiment nul.

En observant les autres estimateurs, notamment les indices totaux, nous observons toujours ce même classement des différents plans. Pour analyser cela plus en détail, nous pouvons nous intéresser aux boxplots des estimateurs de ces indices.

Pour l'indice principal de X_2, nous obtenons le boxplot suivant :

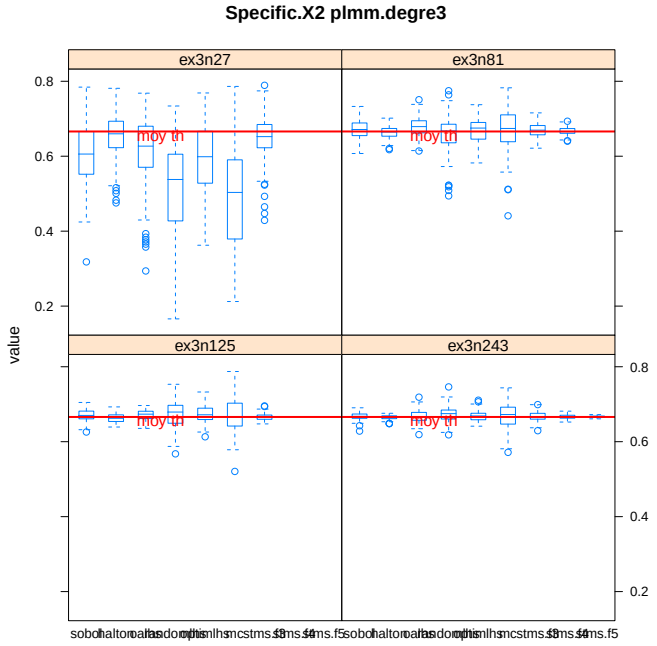


FIGURE 3.10 – Boxplot de l'indice principal de X_2, modèle exemple3.fun, générés grâce à *plmm* de degré 3

Avec 27 observations, les estimateurs sont quasiment tous mauvais : ils sont biaisés et possèdent une très grande variance, avec tout de même quelques valeurs extrêmes (points isolés, en dehors de la boîte et ses moustaches). Il n'y a que les estimateurs obtenus avec les suites de Halton et les plans tms qui donnent de bons résultats.

Avec 81 observations, les estimations sont nettement meilleures. Nous observons que le plan de Monte-Carlo est celui possédant la boîte la plus grande et donc les valeurs des estimateurs les plus dispersées. Il n'est pas biaisé, ceci étant une de ses principales propriétés. Ce sont les tms qui donnent les boîtes les plus petites, et ainsi qui ont la plus petite variance des estimations. De plus, la médiane est situé au même niveau que la valeur théorique de l'indice : ce plan fournit donc un estimateur non biaisé.

En regardant les résultats obtenus avec les plus grands nombres d'observations, nous observons toujours la même tendance : les plans tms sont les meilleurs, et leur erreur est presque nulle avec 243 observations. Avec cette méthode, ce sont les plans tms de force égale à 5 qui sont les plus performants.

Les résultats les plus intéressant à analyser avec la méthode plmm sont ceux pour 81 observations car cette méthode est sensé être performante pour un faible nombre d'observations, à l'opposé des méthodes de Sobol.

En observant l'estimateur de l'indice principal de X_3, Specific.X3, nous pouvons faire les mêmes conclusions :

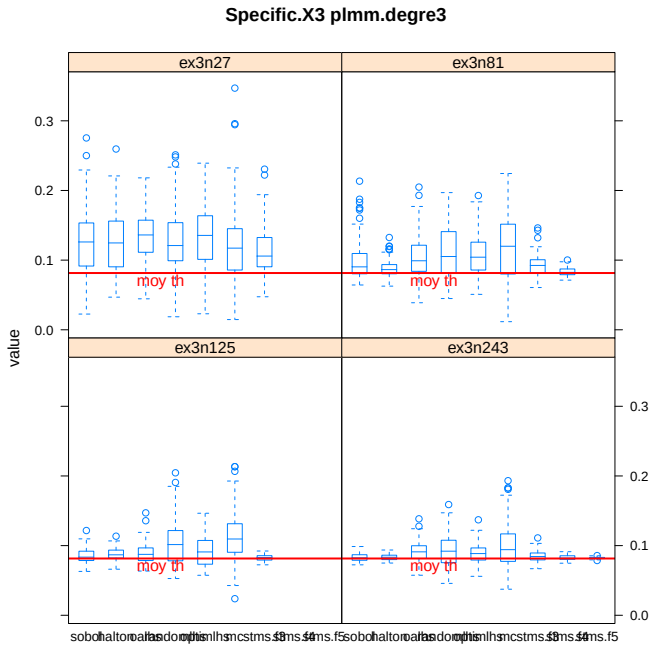


FIGURE 3.11 – Boxplot de l'indice principal de X_3, modèle exemple3.fun, générés grâce à *plmm* de degré 3

Pour N=81, le plan de Monte-Carlo n'est pas très bon, il est biaisé et les valeurs sont très variables. Le plan tms est toujours celui qui donne les résultats les plus précis, environ 5 à 6 fois moins variable que le plan de Monte-Carlo. En effet, la taille de l'inter-quartile du boxplot est inférieure à 0.05 (pour des indices de valeur comprise entre 0 et 1). Même si les suites quasi-aléatoires de Halton sont très performantes, elles possèdent plus de

valeurs extrêmes que les plans tms.

Avec 243 observations, la précision des plans tms force 5 est tellement grande que la variance de la boîte est presque nulle : nous ne discernons presque pas la boîte du trait représentant la valeur théorique de l'indice.

L'analyse des estimateurs des indices totaux nous fournit les mêmes conclusions. Les boxplots des estimateurs de ces indices, ainsi que de l'estimateur de l'indice principal de X_1, par plmm en degré 3 sont donnés en annexes.

Analyse des estimateurs des indices du modèle ishigami.fun avec la fonction *sobol2002*

Voici le graphique représentant les RMSE des estimateurs des indices du modèle ishigami que nous obtenons :

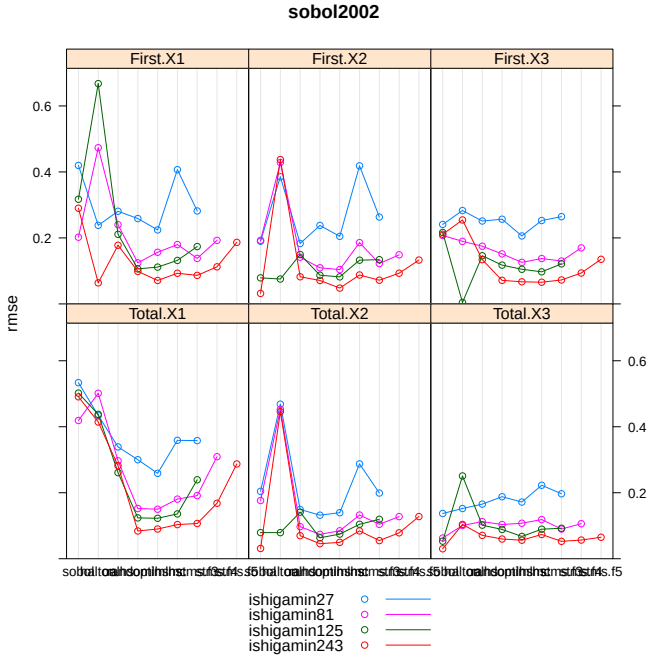


FIGURE 3.12 – RMSE des estimateurs des indices, modèle ishigami.fun, générés grâce à *sobol2002*

Nous pouvons voir que l'ordre de grandeur des RMSE est bien plus grand que lors de l'étude faite précédemment : il est entre 0.1 et 0.5 voire plus grand alors que précédemment il était plutôt inférieur à 0.1. Ceci provient de deux raisons :

- Le modèle d'ishigami est bien plus compliqué à estimer en raison de sa forme théorique en sinus et cosinus, alors que le modèle exemple3.fun est beaucoup plus régulier, et donc les estimateurs vont plus facilement être proches de la valeur théorique des indices
- La fonction sobol2002 nécessite un plus grand nombre de points afin de fournir une bonne estimation que la méthode plmm

Lorsque nous analysons la méthode de Sobol, nous ne nous intéressons qu'aux résultats obtenus pour 125 et 243 observations, les deux autres nombres d'observations étant trop

faibles pour que l'analyse soit pertinente avec la méthode de Sobol. En effet, nous avons vu que pour que cette méthode donne des résultats pouvant être analysés, il fallait un minimum de points.

- Les suites de Halton peuvent donner des résultats très variables : en observant l'indice principal de X_1 nous voyons qu'avec 125 observations, le RMSE est supérieur à 0.6, erreur très importante pour l'estimation d'un indice compris entre 0 et 1. Pour l'estimation de l'indice total de X_1, les suites de Sobol et de Halton ont un RMSE très mauvais.
- Avec 243 observations, les plans tms avec force 3 et lhs semblent être les plus performants. Nous pouvons remarquer que les plans tms force 3 donnent des résultats assez satisfaisants (même si ils pourraient être bien meilleurs) mais que ceux avec force 4 ou force 5 sont bien moins bons. Ceci est dû soit à la méthode utilisée soit au modèle.

Nous pouvons aussi analyser directement l'estimateur de l'indice principal de X_2 :

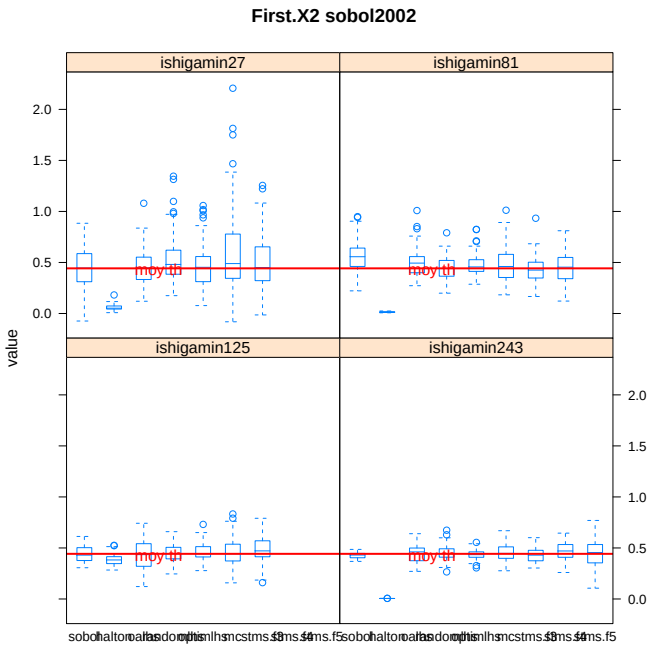


FIGURE 3.13 – Boxplot de l'indice principal de X_2, modèle ishigami.fun, générés grâce à *sobol2002*

Nous voyons donc bien que les suites de Halton donnent des résultats très biaisés (sauf pour 125 observations) et peu variables : ces suites fournissent tout le temps un estimateur qui n'est pas bon.

Comme nous l'avons dit précédemment, les tms donnent de bons résultats uniquement avec la force égale à 3 ; en force 4 et 5 les résultats ne sont pas biaisés mais ils sont beaucoup plus variables.

Le plan optimlhs est celui qui fournit les meilleurs résultats, non biaisés et avec la plus petite variance.

Les conclusions ne sont pas les mêmes que lors de l'étude précédente. Deux pistes sont possibles pour justifier cela :

- les plans tms avec force 3 sont très performants, surtout lorsqu'ils sont utilisés avec l'estimation par plmm. S'ils sont utilisés avec la méthode de Sobol, il faut peut être privilégier une force égale à 5.
- la complexité de la fonction ishigami.fun peut impliquer que ces conclusions sont peut être particulières à une fonction avec peu de régularité

Ces débuts d'analyse doivent être approfondis et renforcés par de nouveaux tests, permettant de valider ou non une des suppositions précédentes.

Commentaires sur les processus gaussiens

Les graphiques obtenus pour les estimations des indices, suite à une métamodélisation par processus gaussiens, sont très similaires. Nous obtenons les graphiques des RMSE suivants :

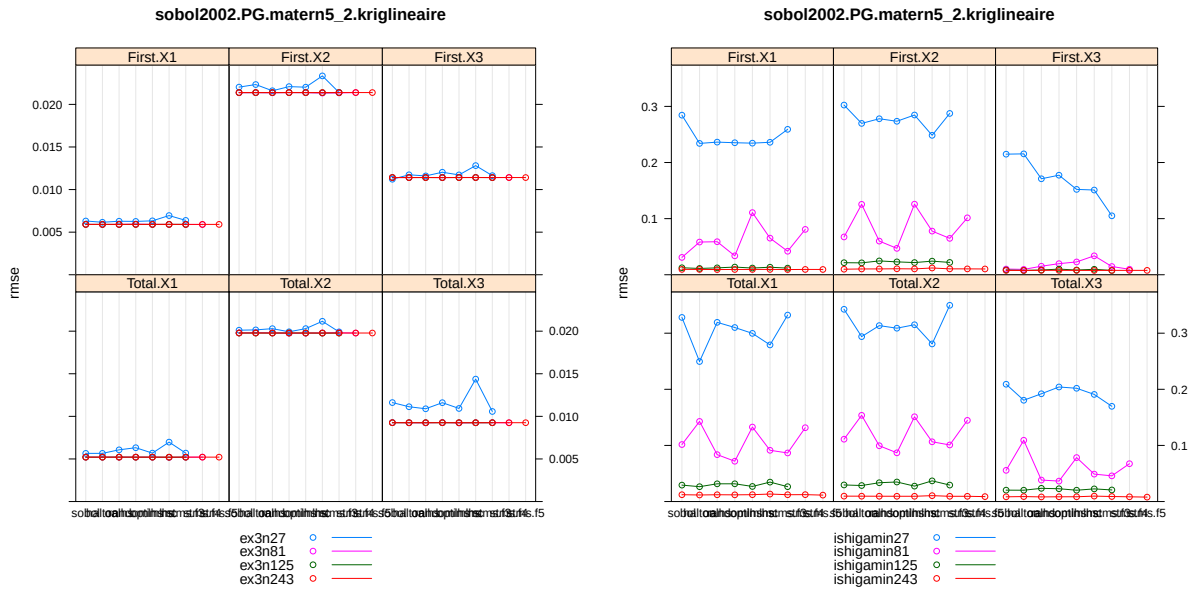


FIGURE 3.14 – RMSE des estimateurs des indices, modèles exemple3.fun et ishigami.fun, générés suite à une métamodélisation par processus gaussiens

Les résultats obtenus par processus gaussiens sont très bons, même avec peu de points, et quel que soit le plan d'expérience. Pour le modèle exemple3.fun, ils sont de très bonne qualité dès 27 ou 81 points, et pour le modèle ishigami, qui est très peu régulier, dès 125 points le RMSE est inférieur à 0.1.

Par exemple, pour le modèle ishigami.fun, l'estimation de l'indice principal de X_2 donne :

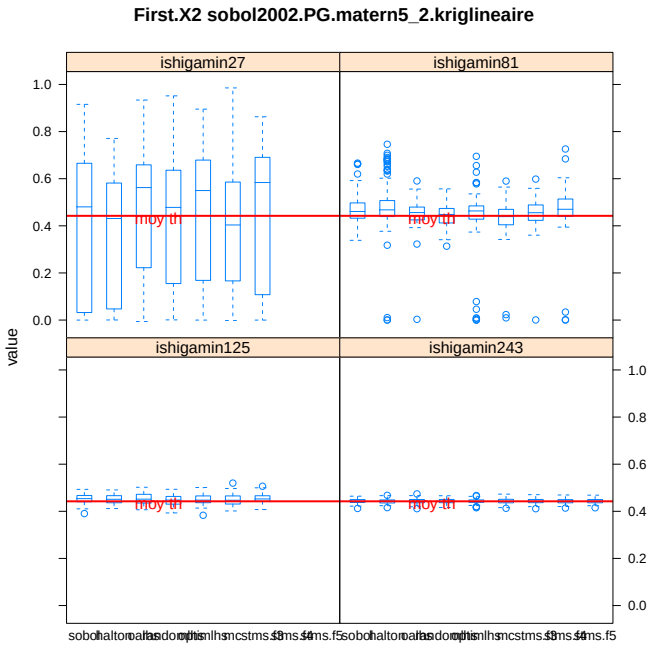


FIGURE 3.15 – Boxplot de l'indice principal de X_2, modèle ishigami.fun, générés suite à une métamodélisation par processus gaussiens

Dès que le nombre d'observations dépasse 125 points, nous ne voyons aucune différence due au plan d'expérience utilisé lors de la métamodélisation. Les petites variations qui peuvent exister sont causées par les deux matrices utilisées comme deuxième plan par la fonction sobol2002. Pour approfondir l'analyse et obtenir des résultats dont les différences pourraient provenir du plan utilisé lors de la métamodélisation, il faudrait augmenter la taille des deux matrices utilisées par sobol2002 (qui étaient de 20 000 points et simulées par loi uniforme dans mes premières expériences). Ceci pourrait être fait lors d'une future analyse.

3.3 Perspectives

Ces analyses sont à approfondir par des tests futurs. En effet, nous avons vu que selon la méthode utilisée les conclusions ne sont pas toujours les mêmes, et que selon la régularité de la fonction les résultats varient aussi. Ainsi, il faut renforcer les tests précédents par de nouveaux tests afin de donner plus de confiance aux conclusions précédentes. Pour cela, une analyse de variance regroupant les différents éléments d'analyse peut être mise en place. Il faudra prendre en compte qu'elle sera partiellement incomplète car la particularité des plans tms et de leur utilisation sous certaines conditions fait qu'ils n'ont pas tous été testés sur toutes les expériences.

De plus, concernant les processus gaussiens, nous avons vu que les modèles obtenus sont très proches quel que soit le plan d'expérience utilisé. Ainsi, comme nous l'avons dit précédemment, de nouvelles expériences, avec les matrices données en argument à la fonction *sobol2002* de taille plus grande, peuvent être effectuées afin d'analyser plus précisément les effets des plans sur la qualité de détermination des indices de sensibilité par le biais d'un processus gaussiens.

Enfin, les estimateurs que nous avons développés pour la fonction *plmm* et qui permettent de déceler les effets de certaines interactions sont à approfondir. En effet, nous avons trouvé le début de la forme mathématique quantifiant certaines interactions et confusions, mais nous ne savons pas encore précisément quelle est la quantité de l'interaction fournie par l'estimateur ni quels sont les types de confusions que ces indicateurs nous permettent de trouver. Il faut donc effectuer de nouveaux tests sur des modèles dont nous pouvons quantifier les interactions afin de valider et d'approfondir ces estimateurs.

Conclusion

Au cours de ce stage, j'ai acquis de nombreuses connaissances sur l'analyse de sensibilité et la planification d'expériences. Ce travail de recherche m'a permis de développer des scripts **R**, afin de générer les résultats nécessaires pour répondre aux problématiques posées. Il m'a aussi aidé à développer de nouvelles méthodes afin d'organiser, d'une manière plus synthétique, un grand nombre de résultats. J'ai aussi apprécié la découverte des enjeux d'une étude de modèle pour une application concrète qui est l'agronomie.

Au niveau professionnel, ce stage m'a permis de découvrir le domaine de la recherche et ses nombreux enjeux. J'ai appris de nouvelles méthodologies de travail, ainsi que d'analyse. J'ai acquis de nouvelles connaissances statistiques et découvert leurs applications à un cas d'étude plus concret. J'ai appris à développer et organiser les programmes avec plus d'efficacité, de rapidité et de rigueur. De même, j'ai appris à développer des méthodes permettant d'optimiser ces programmes.

Je regrette de ne pas avoir eu un peu plus de temps afin d'approfondir mes recherches et afin d'appliquer les méthodes d'analyse de sensibilité apprises au modèle d'une culture de tournesols testé au début de mon stage.

En conclusion, ce stage a été une expérience très enrichissante, que ce soit au niveau personnel ou professionnel. J'ai amélioré ma façon de travailler, ainsi que la manière d'appréhender un problème quel que soit son type. J'ai découvert des cas d'application concrets des éléments théoriques que j'avais appris ou que j'ai pu apprendre tout au long de ce stage. J'ai appris la manière d'aborder un sujet de recherche, dont les solutions aux problématiques posées sont incertaines.

Annexes

Annexe A

Principaux programmes

A.1 Fonction my.samo.model

```
library(akima)
library(rgl)
library(randtoolbox)
library(DiceKriging)
library(planor)
library(sensitivity)
library(lhs)
source('~/R/Srcipts Robert/newmaketms.R')
source('~/R/Srcipts Robert/newoalhs.R')
source('~/ECaspen2013/R/newplmm.mtk.R')

my.samo.model <- function(model, INFOmodel = NULL, p = 3, d = 3, force = 3, N = 27,
                          plans=c("sobol","sobol.scr","halton","torus","oalhs",
                                  "soalhs","randomlhs","optimlhs","mc") ,
                          amorce = 2013, replhs=99, outplan = FALSE,
                          replicate = 1, repli.N=N/replicate,
                          outplmm=TRUE, outsob=TRUE, outPG=TRUE)
{
  ## ARGUMENTS:
  ## model : la fonction modele prenant en entree une matrice (parametres en ligne)
  ##       : la sortie sera appelee Y
  ## INFOmodel : un data frame avec en colonnes name, binf et bsup pour chaque parametre
  ##           : si NULL, on la genere avec les LETTRES et entre 0 et 1
  ## p : nombre de niveaux par facteur (en principe un premier ou puissance de premier)
  ## d : nombre de facteurs
  ## force : force du tableau orthogonal servant de base a la construction
  ## N : taille du plan
  ## amorce : amorce du generateur de nombres aleatoires (increment de 1 par repetition)
  ## replhs : nombre de repetitions de plans lhs ou oalhs avec scambling (99 par defaut)
  ## outplan = TRUE pour stocker tous les plans generes (defaut = FALSE)
  ## outplmm  = TRUE si on veut faire une analyse avec plmm
  ## outsob  = TRUE si on veut faire une analyse avec sobol2002 et soboljansen
    directement
  ## outPG  = TRUE si on veut faire une meta-modelisation par processus gaussiens
  ##          avant l'analyse avec sobol2002 et soboljansen

  ## SORTIE : une structure liste contenant toutes les modalites de plan d'
    echantillonnage
  ## DETAILS : Pas de tableaux apparies

  ## Attention aux taille N en fonction de p, d et force
  ## OALHS : oalhs(p=3, d=13, force=2, N=27, pair = TRUE, scramble = FALSE)
  ## Avec force = 2 au moins jusqu'  d = 13 .  a  coince    14.
  ## Au dessus de d=13, il faut au moins N = 81.
  ## Avec force = 3 au moins jusqu'  d = 10 avec N = 81.  a  coince    11.
  ## Avec force = 3 au moins    partir de d = 11 avec N = 243.
  ## Avec force = 4 au moins    partir de d = 5 avec N = 243.
```

```
## version avec en entree le modele
# Le modele doit prendre en entree une matrice (PLAN) et donner en sortie une matrice PLAN + Ysimule
model.simule <- function (X, tout = FALSE, transfo = FALSE, b1 = INFOmodel$binf,
                          b2 = INFOmodel$bsup, monmodel = model)
{
  if (transfo) {
    Nbfac <- ncol(X)
    X <- t(b1 + t(X) * (b2 - b1))
  }
  sortie <- monmodel(X)
  if (tout)
    sortie <- cbind(X, Y = sortie)
  return(sortie)
}

## On simplifie le summary.plmm pour eviter les affichages
resume.plmm = function (object)
{
  specific <- apply(cbind(object$main, object$total), 1, min)
  total <- apply(cbind(object$main, object$total), 1, max)
  confusion <- (object$total - object$main)
  invisible(100 * cbind(Alone = object$main, Specific = specific,
                        Total = total, Interaction = confusion,
                        Partiel = object$partiel, IntR2p = total*(1- object$partiel)))
}

## Fonction d'analyse par plmm (sur tous les parametres de model)
analyse.plmm = function(SORTIE, DEGRE = 4, NOMfac = paste("X",1:d,sep="") )
{
  ## Fonction qui stocke les resultats de l'analyse de la simulation sur un plan
  ## pour les metamodeles de degre 1 a DEGRE
  sortie = list( general = array(NA, dim=c(DEGRE, 4),
                                 dimnames=list(paste("degre",1:DEGRE,sep=""),
                                               c("residual","dlib", "rse", "se.base"))),
                 summary = array(NA, dim=c(DEGRE, length(NOMfac),6),
                                 dimnames = list(paste("degre",1:DEGRE,sep=""),
                                                 NOMfac,
                                                 c("Alone", "Specific", "Total",
                                                   "Interaction", "Partiel", "IntR2p")
  ) ))
  ## Debut de boucle sur les degres
  for (degre in 1:DEGRE) {
    RES = newplmm.mtk(as.data.frame(SORTIE), deg=degre)
    sortie$general[degre,"residual"] = RES$residual
    sortie$general[degre,"dlib"] = RES$dlib
    sortie$general[degre,"rse"] = RES$rse
    sortie$general[degre,"se.base"] = RES$se.base
    sortie$summary[degre, ,] = resume.plmm(RES) /100 # on renormalise (plmm est en pourcentage)
  }
  ## Fin de boucle sur les degres
  return(sortie)
}
## Fin de la la fonction d'analyse

# Analyse avec la fonction sobol2002
analyse.sobol2002 = function(monmodel, A, B, NOMfac = paste("X",1:d,sep="") )
{
  # d = nbre de param tres
  # on stocke les indices V (qui a 2d+1 colonnes), S (d colonnes) et T(d colonnes)
  #    la suite dans un tableau
  ncol = length(NOMfac)*4+1
  NOMfacV=c("global", paste(NOMfac,".V",sep=""),paste("-",NOMfac,".V",sep=""))
  NOMfacS=paste("First",NOMfac,sep=".")
  NOMfacT=paste("Total",NOMfac,sep=".")
  sortie = list(general = array(NA, dim=c(ncol,5)))

```

```
    RES = sobol2002(monmodel, as.data.frame(A), as.data.frame(B), nboot=100)
    sortie$general=rbind(RES$V, RES$S, RES$T)
    # on renomme les lignes
    rownames(sortie$general)=c(NOMfacV, NOMfacS, NOMfacT)
    sortie$general=as.matrix(sortie$general)
    return(sortie)
  }
  # Fin de l'analyse avec sobol2002

  # Analyse avec la fonction soboljansen
  analyse.soboljansen = function(monmodel, A, B, NOMfac = paste("X",1:d,sep="") )
  {
    # d = nbre de param tres
    # on stocke les indices V (qui a 2d+1 colonnes), S (d colonnes) et T(d colonnes)
    #    la suite dans un tableau
    ncol = length(NOMfac)*4+1
    NOMfacV=c("global", paste(NOMfac,".V",sep=""),paste("-",NOMfac,".V",sep=""))
    NOMfacS=paste("First",NOMfac,sep=".")
    NOMfacT=paste("Total",NOMfac,sep=".")
    sortie = list(general = array(NA, dim=c(ncol,5)))

    RES = soboljansen(monmodel, as.data.frame(A), as.data.frame(B), nboot=100)
    sortie$general=rbind(RES$V, RES$S, RES$T)
    # on renomme les lignes
    rownames(sortie$general)=c(NOMfacV, NOMfacS, NOMfacT)
    sortie$general=as.matrix(sortie$general)
    return(sortie)
  }
  # Fin de l'analyse avec soboljansen

  # Fonction de Krigeage pour les processus gaussiens
  kriging.mean <- function(Xnew, mod)
  {
    predict(object=mod, newdata=Xnew, type="UK", se.compute = FALSE,
            checkNames = FALSE, light.return=TRUE)$mean
  }

  # Analyse avec la fonction sobol2002 avec un mod le de processus gaussiens
  analyse.sobol2002.PG = function(monmodelPG, A, B, NOMfac = paste("X",1:d,sep="") )
  {
    # monmodelPG = le meta-modele par processus gaussiens obtenu    partir de model
    # en param tre de la fonction generale
    # d = nbre de param tres

    # on stocke les indices V (qui a 2d+1 colonnes), S (d colonnes) et T(d colonnes)
    #    la suite dans un tableau
    ncol = length(NOMfac)*4+1
    NOMfacV=c("global", paste(NOMfac,".V",sep=""),paste("-",NOMfac,".V",sep=""))
    NOMfacS=paste("First",NOMfac,sep=".")
    NOMfacT=paste("Total",NOMfac,sep=".")
    sortie = list(general = array(NA, dim=c(ncol,5)))

    RES = sobol2002(model = kriging.mean, X1=data.frame(A), X2=data.frame(B),
                    nboot=100, mod=monmodelPG)[9:11]
    sortie$general=rbind(RES$V, RES$S, RES$T)
    # on renomme les lignes
    rownames(sortie$general)=c(NOMfacV, NOMfacS, NOMfacT)
    sortie$general=as.matrix(sortie$general)
    return(sortie)
  }
  # Fin de l'analyse avec sobol2002 avec un mod le de processus gaussiens


  # Analyse avec la fonction soboljansen avec un mod le de processus gaussiens
  analyse.soboljansen.PG = function(monmodelPG, A, B, NOMfac = paste("X",1:d,sep="") )
  {
    # monmodelPG = metamod le par processus gaussiens obtenu    partir de model
    # en param tre de la fonction generale
    # d = nbre de param tres

```

```
    # on stocke les indices V (qui a 2d+1 colonnes), S (d colonnes) et T(d colonnes)
    #    la suite dans un tableau
    ncol = length(NOMfac)*4+1
    NOMfacV=c("global", paste(NOMfac,".V",sep=""),paste("-",NOMfac,".V",sep=""))
    NOMfacS=paste("First",NOMfac,sep=".")
    NOMfacT=paste("Total",NOMfac,sep=".")
    sortie = list(general = array(NA, dim=c(ncol,5)))

    RES = soboljansen(model = kriging.mean, X1=data.frame(A), X2=data.frame(B),
                      nboot=100, mod=monmodelPG)[9:11]
    sortie$general=rbind(RES$V, RES$S, RES$T)
    # on renomme les lignes
    rownames(sortie$general)=c(NOMfacV, NOMfacS, NOMfacT)
    sortie$general=as.matrix(sortie$general)
    return(sortie)
  }
  # Fin de l'analyse avec soboljansen avec un mod le de processus gaussiens

  ## Calcule le PX qui doit  tre  un des plans de allplan
  run.plan <- function(plan, plan.base, replic=replicate, P=p, f=force, n=N,
                       dim=d, AMORCE, ini=FALSE)

    # fonction qui calcule un plan d'experiences pour n observations
    # valide pour les plans : sobol, sobol.scrambling, halton, torus, random, randomlhs
    et optimLhs
  {
    allplan=c("sobol","sobol.scr","halton","torus","optimlhs","oalhs","soalhs",
              "randomlhs","mc","tms","stms")

    if (plan %in% allplan) {}
    else
      stop("le plan donne en argument n'est pas un plan possible")

    set.seed(AMORCE)
    # on regarde d'abord quel est le plan d'experience utilise
    # soit les suites de Sobol, d'Halton ou de Torus ou sinon un plan aleatoire suivant
    la loi uniforme

    PLAN<-switch(plan,
                 sobol = randtoolbox::sobol(n,dim,init=ini),
                 sobol.scr = randtoolbox::sobol(n=n,dim,init=ini,scrambling=1,seed=AMORCE
    ),
                 halton = randtoolbox::halton(n=n,dim,init=ini),
                 torus = randtoolbox::torus(n=n,dim,init=ini),
                 optimlhs = improvedLHS(n,dim) ,
                 oalhs = oalhs(P, dim, f, n, scramble = FALSE, pair=FALSE,
                               design = plan.base) ,
                 soalhs = oalhs(P, dim, f, n, scramble = TRUE, pair=FALSE,
                                design = plan.base) ,
                 randomlhs = randomLHS(n,dim) ,
                 mc = matrix(runif(n*dim), nrow=n, ncol=dim) )

    if(plan == "tms") {
      ## Simulations avec un plan tms
      ##  Avec les tms, on n'a pas de regle pour optimiser les tms
      ## Par exemple : hslhs(p=3, d=13, force=2, N=27) ou hslhs(p=3, d=13, force=2, N=81)
      ## Dans ce cas, replicate, repete la procedure de N=27 trois fois.
      set.seed(AMORCE)
      ## Les plans pour scramble = FALSE
      if(replic==1) PLAN = hslhs(P, dim, f, N, scramble = FALSE,
                                 design = plan.base)
      else {
        repli.N=n/replic
        ppp = list()
        for(i in 1:replic) ppp[[i]] = hslhs(P, dim, f, repli.N,
                                            scramble = FALSE,
                                            design = plan.base)
        PLAN = ppp[[1]]
        for(i in 2:replic) PLAN = rbind( PLAN, ppp[[i]])
      }
```

```
    }

    if (plan == "stms") {
      ## Les plans pour scramble = TRUE
      set.seed(AMORCE)
      if(replic==1) PLAN = hslhs(P, dim, f, n, scramble = TRUE,
                                 design = plan.base)
      else {
        repli.N=n/replic
        ppp = list()
        for(i in 1:replic) ppp[[i]] = hslhs(P, dim, f, repli.N, scramble = TRUE,
                                            design = plan.base)
        PLAN = ppp[[1]]
        for(i in 2:replic) PLAN = rbind(PLAN, ppp[[i]])
      }
    }

    return(PLAN)
  }
  # Fin de la fonction run.plan

  ## Attention, les structures d'entree doivent etre des data.frame
  ## Les analyses sont faites selon plusieurs options ou identificateurs.
  ## Le degre du polynome du plmm varie de 1 a 4
  ## Chacune de ces options s'analysent sur les N valeurs

  ## Les analyses sont faites sur differents plans, tous de taille N :
  ## suites quasi-aleatoires de sobol (sobol)
  ## suites quasi-aleatoires de sobol scrambled (sobol.scr)
  ## suites quasi-aleatoires de halton (halton)
  ## suites quasi-aleatoires de torus (torus)
  ## oalhs
  ## oalhs scrambled (soalhs)
  ## lhs (randomlhs)
  ## lhs optimise avec la fonction improvedLHS (optimlhs)
  ## runif (mc)
  ## si option TMS=TRUE :
  ## tms fixe (tms)
  ## tms scrambled (stms)

  if (is.null(INFOmodel)) {
    INFOmodel = data.frame(name = paste("X",1:d,sep=""), binf = rep(0, d), bsup= rep(1,d)
    )
  }

  OA=FALSE
  SOA=FALSE
  TMS=FALSE
  STMS=FALSE

  if ("oalhs" %in% plans) {
    OA=TRUE
  }

  if ("soalhs" %in% plans) {
    SOA=TRUE
  }

  if ("tms" %in% plans) {
    TMS=TRUE
  }

  if ("stms" %in% plans) {
    STMS=TRUE
  }

  ## on definit la structure finale
  ## une partie analyse par plmm si outplmm = TRUE
  if (outplmm) {
    analyseS.plmm  = list( general = array(NA, dim=c(length(plans),replhs,4, 4),
```

```
                                                dimnames=list(plans,
                                                              paste("rep",1:replhs,sep=""),
                                                              paste("degre",1:4,sep=""),
                                                              c("residual","dlib", "rse", "se.base"))),
                                summary = array(NA, dim=c(length(plans),replhs, 4, d, 6),
                                                dimnames = list(plans,
                                                                paste("rep",1:replhs,sep=""),
                                                                paste("degre",1:4,sep=""),
                                                                INFOmodel$name,
                                                                c("Alone", "Specific", "Total",
                                                                  "Interaction", "Partiel", "IntR2p") ) ) )
  }

  NOMfacV=c("global", paste(INFOmodel$name,".V",sep=""),paste("-",INFOmodel$name,".V",sep=""))
  NOMfacS=paste("First",INFOmodel$name,sep=".")
  NOMfacT=paste("Total",INFOmodel$name,sep=".")

  ## une partie analyse par sobol2002 et soboljansen (sans simplifaction)
  ## si outsob = TRUE
  if (outsob) {
    analyseS.sobol2002  = list(general = array(NA, dim=c(length(plans),replhs,4*d+1,5),
                                               dimnames=list(plans, paste("rep",1:replhs,sep=""),
                                                             c(NOMfacV, NOMfacS, NOMfacT),
                                                             c("original","bias","std.error",
                                                               "min. c.i.","max. c.i.") )
     ) )

    analyseS.soboljansen  = list(general = array(NA, dim=c(length(plans),replhs,4*d+1,5),
                                                 dimnames=list(plans, paste("rep",1:replhs,sep=""),
                                                               c(NOMfacV, NOMfacS, NOMfacT),
                                                               c("original","bias","std.error",
                                                                 "min. c.i.","max. c.i.")
     ) ) )

  }

  ## une partie analyse par sobol2002 et soboljansen apr s processus gaussiens
  ## si outPG = TRUE
  if (outPG) {
    cov=c("matern3_2","matern5_2")
    formula=c(~1,~.)
    names_formula=c("krig ordinaire","krig lineaire")

    analyseS.sobol2002.PG  = list(general = array(NA, dim=c(length(plans),replhs,4*d+1,5,
                                                            length(cov),length(formula)),
                                                  dimnames=list(plans, paste("rep",1:replhs,sep=""),
                                                                c(NOMfacV, NOMfacS, NOMfacT),
                                                                c("original","bias","std.error",
                                                                  "min. c.i.","max. c.i."),
                                                                cov,
                                                                names_formula) ) )

    analyseS.soboljansen.PG  = list(general = array(NA, dim=c(length(plans),replhs,4*d+1,5,
```

```
                                                                length(cov),length(formula)
    ),
                                                        dimnames=list(plans, paste("rep",1:
    replhs,sep=""),
                                                                c(NOMfacV, NOMfacS,
    NOMfacT),
                                                                c("original","bias","
    std. error",
                                                                  "min. c.i.","max. c.i
    ."),
                                                                cov,
                                                                names_formula) ) )
  }

  ## Attention, les permutations etant aleatoires, il faut gerer l'amorce.
  ## L'analyse est repetee pour chacune des options de plan

  ## Initialisation de la sortie des plans
  names_plans=paste("PLAN",plans,sep=".")
  names_plans2=paste(names_plans,"2",sep="")

  # On stocke les plans si outplan=TRUE
  if (outplan) {
    PLANS.tout = list()

    for (pp in 1:length(plans)) {
      PLANS.tout[[names_plans[pp]]]= array(NA, dim=c(replhs,N, d))
    }

    ## si outsob = TRUE il faut un deuxi me plan car sobol2002 et jansen
    ## prenent deux matrices X et X en entree
    if (outsob) {
      for (pp2 in 1:length(plans)) {
        PLANS.tout[[names_plans2[pp2]]]= array(NA, dim=c(replhs,N, d))
      }
    }
  }

  ## Calculs des plans de base pour les OA et pour les HS (tms-net)
  if(OA|SOA) plan.OAbase = oalhs.base(p, d, force, N)
  if(TMS|STMS) plan.TMSbase = hslhs.base(p, d, force, repli.N)

  ## Debut de la boucle sur les repetitions (replhs)
  amorce=amorce-1
  for(rep in 1:replhs) {
    amorce=amorce+1

    # lors de la 1ere boucle init = True et apr s False pour changer l'init
    #  des suites quasi-aleatoires
    init=(rep==1)

    PLANS=list()
    names_sorties=paste("SORTIE",plans,sep=".")
    sortie=list()

    for(sim in 1:length(plans)) {
      set.seed(amorce)
      if ((plans[sim]=="oalhs")|(plans[sim]=="soalhs")) {
        PLANS[[names_plans[sim]]]= try(run.plan(plan=plans[sim],
                                                plan.base=plan.OAbase,
                                                P=p,
                                                AMORCE=amorce))
      }

      else if ((plans[sim]=="tms")|(plans[sim]=="stms")){
        PLANS[[names_plans[sim]]]= try(run.plan(plan=plans[sim],
                                                plan.base=plan.TMSbase,
                                                P=p,
                                                AMORCE=amorce))
      }
```

```
      else
      {
        PLANS[[names_plans[sim]]]=try(run.plan(plan =plans[sim],ini=init,
                                                AMORCE=amorce))
      }

      dimnames(PLANS[[names_plans[sim]]])[[2]] <- as.character(INFOmodel$name)

      if (typeof(PLANS[[names_plans[sim]]]) != "character")
        sortie[[names_sorties[sim]]]= try(model.simule(PLANS[[names_plans[sim]]],
                                                        tout =TRUE, transfo=TRUE,
                                                        monmodel = model))
    }

    if (outsob) {
      ## Simulation du 2eme plan pour la methode Sobol sans PG si outsob = TRUE
      amorce=amorce+1
      init=FALSE
      PLANS2=list()

      for(sim2 in 1:length(plans)) {
        set.seed(amorce)
        if ((plans[sim2]=="oalhs")|(plans[sim2]=="soalhs")) {
          PLANS2[[names_plans2[sim2]]]=try(run.plan(plan=plans[sim2],
                                                    plan.base=plan.OAbase,
                                                    P=p,
                                                    AMORCE=amorce))
        }

        else if ((plans[sim2]=="tms")|(plans[sim2]=="stms")){
          PLANS2[[names_plans2[sim2]]]=try(run.plan(plan=plans[sim2],
                                                    plan.base=plan.TMSbase,
                                                    P=p,
                                                    AMORCE=amorce))
        }
        else
        {
          PLANS2[[names_plans2[sim2]]]=try(run.plan(plan =plans[sim2],ini=init,
                                                    AMORCE=amorce))
        }
        dimnames(PLANS2[[names_plans2[sim2]]])[[2]] <- as.character(INFOmodel$name)
      }
    }

    ## On stocke les plans
    if (outplan) {
      for(pl in 1:length(plans)) {
        name=names_plans[pl]
        if (typeof(PLANS[[name]]) != "character")
          PLANS.tout[[name]][rep, , ] = as.matrix(PLANS[[name]])
      }
      if(outsob) {
        for(pl2 in 1:length(plans)) {
          name2=names_plans2[pl2]
          if (typeof(PLANS2[[name2]]) != "character")
            PLANS.tout[[name2]][rep, , ] = as.matrix(PLANS2[[name2]])
        }
      }
    }

    ## ANALYSES sur les differents plans

    if (outplmm) {
      for (aplmm in 1:length(plans)) {
        name_sort=names_sorties[aplmm]

        res = try(analyse.plmm(sortie[[name_sort]]))
        if (typeof(res) != "character") {
          analyseS.plmm$general[plans[aplmm], rep, , ] = res$general
          analyseS.plmm$summary[plans[aplmm], rep, , ,] = res$summary
```

```
        }
        rm(res)
      }
    }

    if (outsob) {
      for (asob in 1:length(plans)) {
        name_pl=names_plans[asob]
        name_pl2=names_plans2[asob]

        sob= try(analyse.sobol2002(model, A=PLANS[[name_pl]], B=PLANS2[[name_pl2]]))
        if (typeof(sob) != "character")
          analyseS.sobol2002$general[plans[asob], rep, , ] = sob$general
        rm(sob)

        sobjansen=try(analyse.soboljansen(model, A=PLANS[[name_pl]], B=PLANS2[[name_pl2
    ]]))
        if (typeof(sobjansen) != "character")
          analyseS.soboljansen$general[plans[asob], rep, , ] = sobjansen$general
        rm(sobjansen)
      }
    }

    ## Si on utilise un meta-mod le par processus gaussiens
    ## pour ensuite faire l'analyse avec sobol2002 puis soboljansen
    if (outPG) {

      # taille des matrices A et B utilisees apr s le calcul du meta-mod le PG
      # pour analyser par sobol ! taille bien plus grande que le 1er PX utilise pour
      # le calcul du meta-mod le par PG
      NPG=20000

      set.seed(amorce+100)

      A=matrix(runif(NPG*d), nrow=NPG, ncol=d)
      B=matrix(runif(NPG*d), nrow=NPG, ncol=d)

      dimnames(A)[[2]] <- as.character(INFOmodel$name)
      dimnames(B)[[2]] <- as.character(INFOmodel$name)

      ### RAPPEL :
      # cov=c("matern3_2","matern5_2") et formula=c(~1,~.)
      # names_formula=c("krig ordinaire","krig lineaire")
      lower=rep(sqrt(d)/100,d)
      upper=rep(2*sqrt(d),d)

      # boucle sur les differentes formules pour les tendances utilisees pour le calcul
      # du PG
      for (ff in 1:length(formula)) {

        # boucle sur les differentes covariances utilisees pour le PG
        for (cc in 1:length(cov)) {
          # SORTIE.sobol est obtenue avec la fonction model.simule qui concatene
          # le plan d'experiences X et Y donc pour metter dans response de km
          # on ne recup re que la sortie Y

          for (aPG in 1:length(plans)) {
            name_pl=names_plans[aPG]
            name_sort=names_sorties[aPG]

            set.seed(amorce)
            mod <- try( km(formula=formula[[ff]], design=data.frame(PLANS[[name_pl]]) ,
                           response=sortie[[name_sort]][,"Y"], covtype=cov[cc],
                           lower=lower,upper=upper) )

            if (typeof(mod)=="character") {
              set.seed(amorce+1)
              mod <- try( km(formula=formula[[ff]], design=data.frame(PLANS[[name_pl]]) ,
                             response=sortie[[name_sort]][,"Y"], covtype=cov[cc],
                             lower=lower,upper=upper) )
```

```
575             }

577             if (typeof(mod)=="character"){
                  set.seed(amorce+2)
579               mod <- try(km(formula=formula[[ff]], design=data.frame(PLANS[[name_pl]]) ,
                                response=sortie[[name_sort]][,"Y"], covtype=cov[cc],
581                             lower=lower/2, upper=upper/2))
                }
583
                if (typeof(mod)=="character"){
585               name_fich=paste("erreurkm_","am",amorce-1,"rep",rep,name_pl, sep="")
                  fich=paste("/home/mcoustaroux/R/erreur/",name_fich,sep="")
587
                  write.table(sortie[[name_sort]],file=fich)
589             }

591             else {
                  sob=try(analyse.sobol2002.PG(mod,A=A,B=B))
593               if (typeof(sob) != "character")
                    analyseS.sobol2002.PG$general[plans[aPG], rep, ,
595                                                 ,cov[cc],names_formula[ff]] = sob$general
                  rm(sob)
597
                  sobjansen=try(analyse.soboljansen.PG(mod,A=A,B=B))
599               if (typeof(sobjansen) != "character")
                    analyseS.soboljansen.PG$general[plans[aPG], rep, , ,
601                                                   cov[cc],names_formula[ff]] = sobjansen$
    general
                  rm(sobjansen)
603             }
                rm(mod)
605           }
            }
607         # fin de la boucle sur les types de covariance pour km()
          }
609       # fin de la boucle sur les types de formula pour la tendance du km()
        }
611     # fin de la boucle pour les PG
      }
613   ## Fin de la boucle sur les repetitions (replhs)

615   result=list()

617   ## on stocke tous les resultats dans la variable result que la fonction va renvoyer
      if (outplmm)
619     result$analyses.plmm = analyseS.plmm

621   if (outsob) {
        result$analyses.sobol2002 = analyseS.sobol2002
623     result$analyses.soboljansen = analyseS.soboljansen
      }
625   if (outPG) {
        result$analyses.sobol2002.PG = analyseS.sobol2002.PG
627     result$analyses.soboljansen.PG = analyseS.soboljansen.PG
      }
629   if (outplan) {
        result$plans = PLANS.tout
631   }
      result$appel=match.call()
633   return (result)
    }
```

Listing A.1 – Fonction my_samo_model_function.R

A.2 Fonction permettant de tracer les premiers boxplot

```
library(lattice)
source('~/R/my_samo_model_function.R')

my.bwplot.px <- function(test,real_mean=NULL,indices="",methode="sobol2002",plans="",name
    ="")
{ ## on donne l'objet test qui vient d'une realisation de la fonction my.samo.model
  ## si on veut representer sur le boxplot de la vrai valeur
  ## des indices First et Totaux on les passe en param tre dans l'ordre
  ## dans le vecteur "real_mean" dans l'ordre : First.X1 Total.X1 First.X2 Total.X2 ...
  ## Si on ne veut representer qu'une partie des indices pour certains facteurs et
    indices
  ## on les passe en param tre dans la variable "indice" sous la bonne forme
  ## ex: First.X2, Total.X3 ... sinon par defaut on prend tous les indices

  allmethod=c("sobol2002","soboljansen","plmm","sobol2002.PG","soboljansen.PG")

  if (methode %in% allmethod) {}
  else
    stop("la methode donne en argument n'est pas une methode possible")

  method=paste("analyses",methode,sep=".")

  ## Si on n'a passe aucun plan en arguments, on les trace tous
  if (plans[1]=="")
    plans=dimnames(test[[method]][["general"]])[1][[1]]

  ## Si on veut tracer avec methode plmm
  if (methode=="plmm")
  {
    ##Si on n a pas donne les indices en argument
    ## Cre un vecteur : Specific.X1 Total.X1 Specific.X2 Total.X2 ...
    if(indices[1] == "")
    {
      #fac est le nom de tous les facteurs
      fac=dimnames(test[[method]][["summary"]])[[4]]
      n=length(fac)
      indices=NULL
      for (nnn in 1:n)
        indices=c(indices,paste(c("Specific","Total"),fac[nnn],sep="."))
    }

    deg=dimnames(test[[method]][["general"]])[3][[1]]
    # on cre un data.frame avec toutes les donnees pour ensuite en faire le bwplot
    table=expand.grid(method=method,
                      plan=plans,
                      rep=dimnames(test[[method]][["general"]])[2][[1]],
                      degre=deg,
                      indice=indices,
                      valeur=NA)
    # boucle qui met les valeurs des indice    l'interieur de "valeur" du data.frame
    for (i in 1:dim(table)[1])
    {
      plani=as.character(table[i,"plan"])
      indi=as.character(table[i,"indice"])
      repi=as.character(table[i,"rep"])
      degi=as.character(table[i,"degre"])

      # indi est egal    Specifique.Xi ou Total.Xi et on doit separer "Specific" de "Xi"
      # pour obtenir la valeur de l'indice dans le summary de notre objet
      indi=unlist(strsplit(indi,"[.]"))
      # indi est un vecteur contenant Specific ou Total dans la 1 re  colonne
      # et le facteur Xi dans la 2 me

      table[i,"valeur"]=test[[method]][["summary"]][plani,repi,degi,indi[2],indi[1] ]
    }
```

```
    if (name!="")
      main=paste("Indices selon differents PX par",methode,"pour",name,sep=" ")
    else
      main=paste("Indices selon differents PX par",methode,sep=" ")

    bxpl=list()
    n=length(indices)

    for (j in 1:length(deg))
    {
      # on trace le boxplot des valeurs en fonction du plan et de l'indice,
      # c'est un boxplot sur les repetitions rep
      table_deg=subset(table, degre==deg[j])
      bxpl[[deg[j]]]=bwplot(valeur ~ plan | indice, data=table_deg,
                            panel=function(panel.number,...){
                              panel.bwplot(..., pch='|')
                              panel.abline(h=real_mean[which.packet()],col="red",lwd=2)
                              panel.text(3.5,real_mean[which.packet()]-0.01,lab="moy th",
    col="red")
                            },
                            main=paste(main,deg[j],sep=", "),
                            layout=c(2,ceiling(n/2)), as.table=TRUE)
    }
  }
  ## Si on veut tracer avec une methode Sobol avec ou sans PG
  else
  {
    ##Si on n a pas donne les indices en argument
    ## Cre un vecteur : First.X1 Total.X1 First.X2 Total.X2 ...
    if(indices[1] == "")
    {
      #nn est la taile du vecteur de tous les noms des indices, du "global",    V, First
    et Total
      nn=length(dimnames(test[[method]][["general"]])[[3]])
      # n=au nombre d'indices First + Total
      n=(nn-1)/4  #on decompte le "global" avec -1
      fac=paste("X",1:n,sep="")
      indices=NULL
      for (nnn in 1:n)
        indices=c(indices,paste(c("First","Total"),fac[nnn],sep="."))
    }
    n=length(indices)

    if (methode %in% c("sobol2002.PG","soboljansen.PG"))
    {
      cov=dimnames(test[[method]][["general"]])[5][[1]]
      formul=dimnames(test[[method]][["general"]])[6][[1]]

      # on cre un data.frame avec toutes les donnees pour ensuite en faire le bwplot
      table=expand.grid(method=method,
                        plan=plans,
                        rep=dimnames(test[[method]][["general"]])[2][[1]],
                        indice=indices,
                        covariance=cov, formula=formul,
                        valeur=NA)

      # boucle qui met les valeurs des indice    l'interieur de "valeur" du data.frame
      for (i in 1:dim(table)[1])
      {
        plani=as.character(table[i,"plan"])
        indi=as.character(table[i,"indice"])
        repi=as.character(table[i,"rep"])
        covi=as.character(table[i,"covariance"])
        formulai=as.character(table[i,"formula"])

        table[i,"valeur"]=test[[method]][["general"]][plani,repi,indi,1,covi,formulai]
      }

      if (name!="")
```

```
        main=paste("Indices selon differents PX par",methode,"pour",name,sep=" ")
      else
        main=paste("Indices selon differents PX par",methode,sep=" ")

      bxpl=list()

      for (cc in 1:length(cov))
      {
        # on trace le boxplot des valeurs en fonction du plan et de l'indice,
        # c'est un boxplot sur les repetitions rep
        for (ff in 1:length(formul))
        {
          table_cov_formula=subset(table, formula==formul[ff] & covariance==cov[cc])
          name=paste(cov[cc],formul[ff],sep=".")
          bxpl[[name]]=bwplot(valeur ~ plan | indice, data=table_cov_formula,
                              panel=function(panel.number,...){
                                panel.bwplot(..., pch='|')
                                panel.abline(h=real_mean[which.packet()],col="red",lwd=2)
                                panel.text(3.5,real_mean[which.packet()]-0.01,lab="moy th
    ",col="red")
                              },
                              main=paste(main,name,sep=", "),
                              layout=c(2,ceiling(n/2)), as.table=TRUE )
        }
        # fin de la boucle pour les formula
      }
      # fin de la boucle sur les covariances
    }
    # fin de la boucle pour la methode PG

    else
    {
      # on cre un data.frame avec toutes les donnees pour ensuite en faire le bwplot
      table=expand.grid(method=method,
                        plan=plans,
                        rep=dimnames(test[[method]][["general"]])[2][[1]],
                        indice=indices,
                        valeur=NA)

      # boucle qui met les valeurs des indice    l'interieur de "valeur" du data.frame
      for (i in 1:dim(table)[1])
      {
        plani=as.character(table[i,"plan"])
        indi=as.character(table[i,"indice"])
        repi=as.character(table[i,"rep"])

        table[i,"valeur"]=test[[method]][["general"]][plani,repi,indi,1]
      }

      if (name!="")
        main=paste("Indices selon differents PX par",methode,"pour",name,sep=" ")
      else
        main=paste("Indices selon differents PX par",methode,sep=" ")

      # on trace le boxplot des valeurs en fonction du plan et de l'indice,
      # c'est un boxplot sur les repetitions rep
      bxpl=list(bx=bwplot(valeur ~ plan | indice, data=table,
                          panel=function(panel.number,...){
                            panel.bwplot(..., pch='|')
                            panel.abline(h=real_mean[which.packet()],col="red",lwd=2)
                            panel.text(3.5,real_mean[which.packet()]-0.01,lab="moy th",
    col="red")
                          },
                          main=main, layout=c(2,ceiling(n/2)), as.table=TRUE ))
    }
  }
  box=lapply(bxpl,print)
}
```

Listing A.2 – Fonction bwplot.my.samo.model.R

A.3 Fonction permettant de tracer les RMSE

```
library(lattice)
source('~/R/my_samo_model_function.R')

# fonction qui calcule le RMSE pour un vecteur d'observations donne
# et la valeur theorique de ces observations
rmse <- function(obs,pred)
{
  diff=obs-pred
  diff=diff[!is.na(diff)]  #to remove NA values
  return (sqrt(mean((diff)^2)))
}

my.barchart.rmse <- function(test,real_mean=NULL,indices="",methode="sobol2002",name="")
{
  ## on donne l'objet test qui vient d'une realisation de la fonction my.samo.model
  ## Si on ne veut representer qu'une partie des rmse des indices
  ## pour certains facteurs et indices
  ## on les passe en param tre dans la variable "indices" sous la bonne forme
  ## ex: First.X2, Total.X3 ... sinon par defaut on prend tous les indices

  allmethod=c("sobol2002","soboljansen","plmm","sobol2002.PG","soboljansen.PG")

  if (methode %in% allmethod) {}
  else
    stop("la methode donne en argument n'est pas une methode possible")

  method=paste("analyses",methode,sep=".")

  ## Genere un vecteur du nom des indices si rien n'est donne en parametre
  if(indices[1] == "") {
    ## Si on veut tracer avec methode plmm
    if (methode=="plmm")
    {
      #fac est le nom de tous les facteurs
      fac=dimnames(test[[method]][["summary"]])[[4]]
      nn=length(fac)
      indices=NULL
      for (nnn in 1:nn)
        indices=c(indices,paste(c("Specific","Total"),fac[nnn],sep="."))
    }
    else
    {
      #nn est la taile du vecteur de tous les noms des indices, du "global",    V, First et Total
      nn=length(dimnames(test[[method]][["general"]])[[3]])
      # n=au nombre d'indices First + Total
      n=(nn-1)/4  #on decompte le "global" avec -1
      fac=paste("X",1:n,sep="")
      indices=NULL
      for (nnn in 1:n)
        indices=c(indices,paste(c("First","Total"),fac[nnn],sep="."))
    }
  }

  if (length(real_mean)!=length(indices))
    stop("le nombre d'indices est differenct du nombre de valeurs theoriques dans real_mean")

  ## Si on veut tracer avec methode plmm
  if (methode=="plmm")
  {
    deg=dimnames(test[[method]][["general"]])[3][[1]]
    # on cre un data.frame avec toutes les donnees pour ensuite en faire le barchart
    table=expand.grid(method=method,
                      plan=dimnames(test[[method]][["general"]])[1][[1]],
                      degre=deg,
                      indice=indices,
```

```
                          rmse=NA)
    # boucle qui met les valeurs des indice    l'interieur de "valeur" du data.frame
    for (i in 1:dim(table)[1])
    {
      plani=as.character(table[i,"plan"])
      indi=as.character(table[i,"indice"])
      degi=as.character(table[i,"degre"])

      for (j in (1:length(indices)))
      {
        if (indices[j]==indi)
          n=real_mean[j]
      }
      # indi est egal    Specifique.Xi ou Total.Xi et on doit separer "Specific" de "Xi"
      # pour obtenir la valeur de l'indice dans le summary de notre objet
      indi=unlist(strsplit(indi,"[.]"))
      # indi est un vecteur contenant Specific ou Total dans la 1 re  colonne
      # et le facteur Xi dans la 2 me

      all_ind=test[[method]][["summary"]][plani,,degi,indi[2],indi[1] ]
      table[i,"rmse"]=rmse(all_ind,n)
    }

    if (name!="")
      main=paste("Rmse selon differents PX par",methode,"pour",name,sep=" ")
    else
      main=paste("Rmse selon differents PX par",methode,sep=" ")

    barx=list()

    for (j in 1:length(deg))
    {
      table_deg=subset(table, degre==deg[j])
      # on trace le barchart des valeurs en fonction du plan et de l'indice
      barx[[deg[j]]]=barchart(rmse ~ plan | indice, data=table_deg,
                              main=paste(main,deg[j],sep=", "),
                              layout=c(2,length(indices)/2), as.table=TRUE )
    }
  }

  else
  {
    if (methode %in% c("sobol2002.PG","soboljansen.PG"))
      # m thode sobol avec PG
    {
      cov=dimnames(test[[method]][["general"]])[5][[1]]
      formul=dimnames(test[[method]][["general"]])[6][[1]]

      # on cre un data.frame avec toutes les donnees pour ensuite en faire le barchart
      table=expand.grid(method=method,
                        plan=dimnames(test[[method]][["general"]])[1][[1]],
                        indice=indices,
                        covariance=cov, formula=formul,
                        rmse=NA)

      # boucle qui met les valeurs des indice    l'interieur de "valeur" du data.frame
      for (i in 1:dim(table)[1])
      {
        plani=as.character(table[i,"plan"])
        indi=as.character(table[i,"indice"])
        covi=as.character(table[i,"covariance"])
        formulai=as.character(table[i,"formula"])

        for (j in (1:length(indices)))
        {
          if (indices[j]==indi)
            n=real_mean[j]
        }

        all_ind=test[[method]][["general"]][plani,,indi,1,covi,formulai]
```

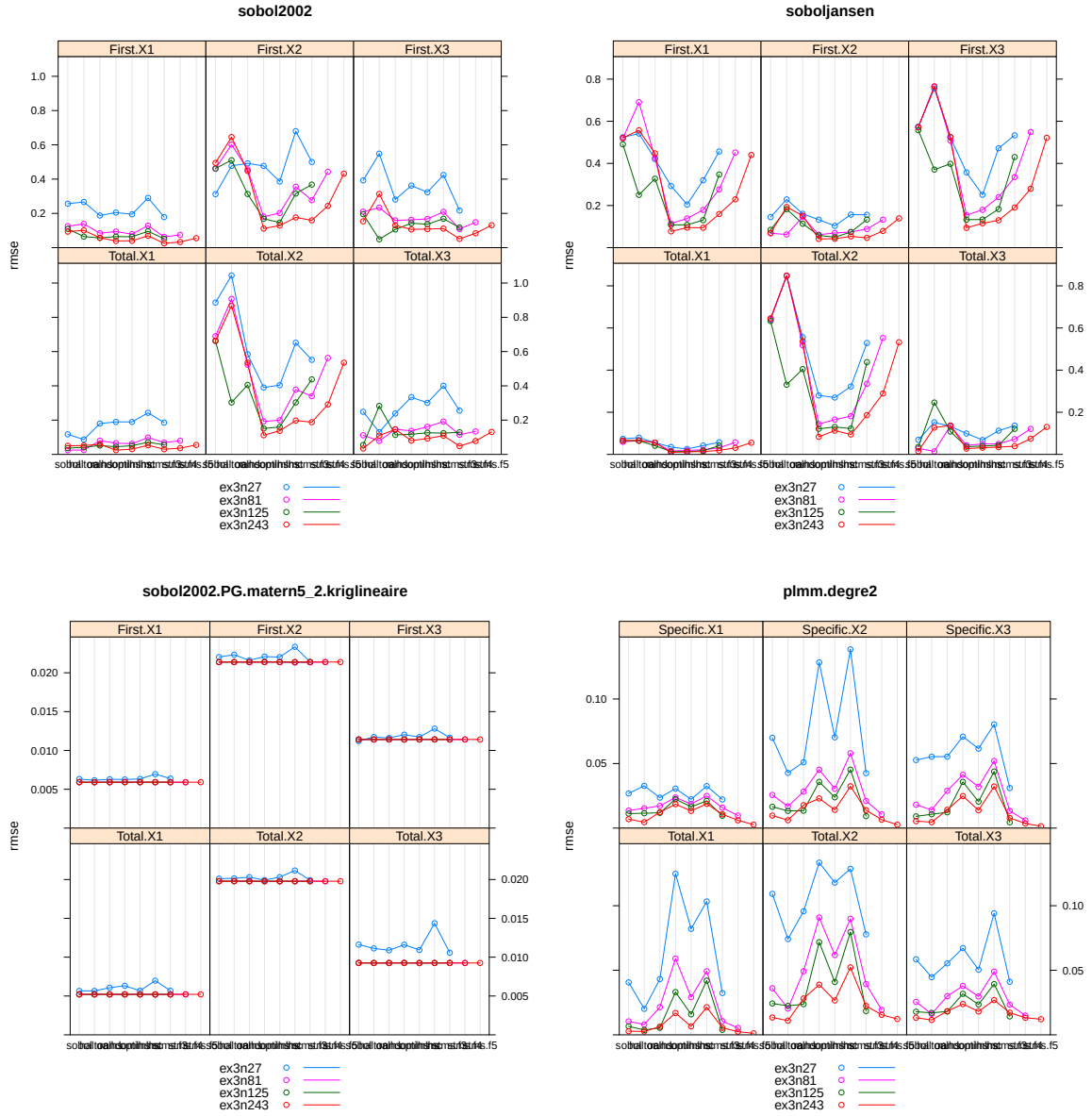
```
        table[i,"rmse"]=rmse(all_ind,n)
      }

      if (name!="")
        main=paste("Rmse selon differents PX par",methode,"pour",name,sep=" ")
      else
        main=paste("Rmse selon differents PX par",methode,sep=" ")

      barx=list()

      for (cc in 1:length(cov))
      {
        for (ff in 1:length(formul))
        {
          table_cov_formula=subset(table, covariance==cov[cc] & formula==formul[ff])
          name=paste(cov[cc],formul[ff],sep=".")

          # on trace le barchart des valeurs en fonction du plan et de l'indice
          barx[[name]]=barchart(rmse ~ plan | indice, data=table_cov_formula,
                                main=paste(main,name,sep=", "),
                                layout=c(2,length(indices)/2), as.table=TRUE )
        }
        # fin de la boucle sur les formula
      }
      # fin de la boucle sur les covariances
    }
    # fin de la boucle pour la methode PG

    else
      # methode sobol sans PG
    {
      # on cre un data.frame avec toutes les donnees pour ensuite en faire le barchart
      table=expand.grid(method=method,
                        plan=dimnames(test[[method]][["general"]])[1][[1]],
                        indice=indices,
                        rmse=NA)

      # boucle qui met les valeurs des indice    l'interieur de "valeur" du data.frame
      for (i in 1:dim(table)[1])
      {
        plani=as.character(table[i,"plan"])
        indi=as.character(table[i,"indice"])

        for (j in (1:length(indices)))
        {
          if (indices[j]==indi)
            n=real_mean[j]
        }

        all_ind=test[[method]][["general"]][plani,,indi,1]
        table[i,"rmse"]=rmse(all_ind,n)
      }

      if (name!="")
        main=paste("Rmse selon differents PX par",methode,"pour",name,sep=" ")
      else
        main=paste("Rmse selon differents PX par",methode,sep=" ")

      # on trace le barchart des valeurs en fonction du plan et de l'indice
      barx=list(barchart=barchart(rmse ~ plan | indice, data=table, main=main,
                                  layout=c(2,length(indices)/2), as.table=TRUE ) )
    }
    # fin de la boucle sur la methode Sobol sans PG
  }
  # fin de la boucle sur la methode Sobol (avec ou sans PG)
  bar=lapply(barx,print)
}
```

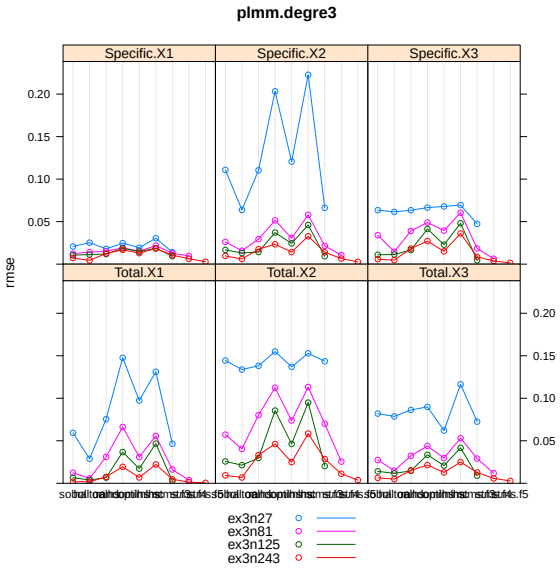
Listing A.3 – Fonction rmse.R

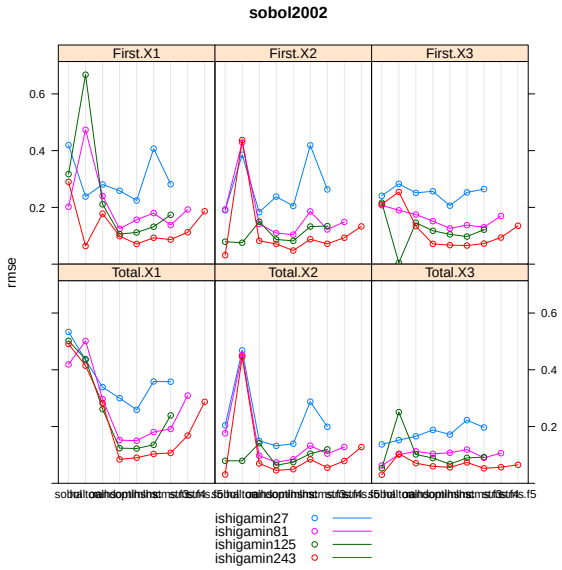
Annexe B

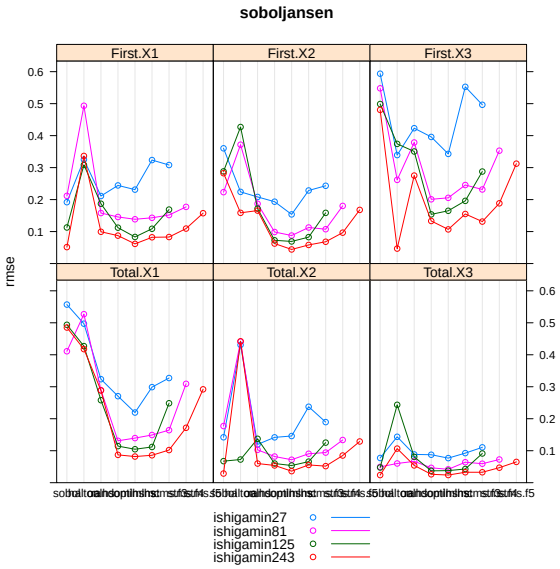
RMSE

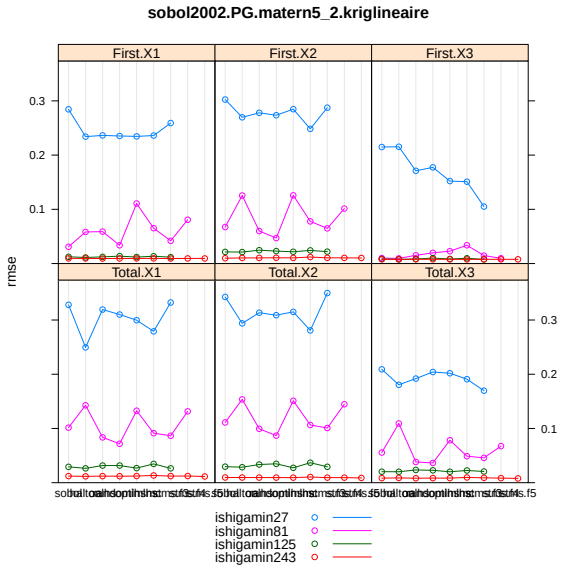
B.1 RMSE des estimateurs des indices

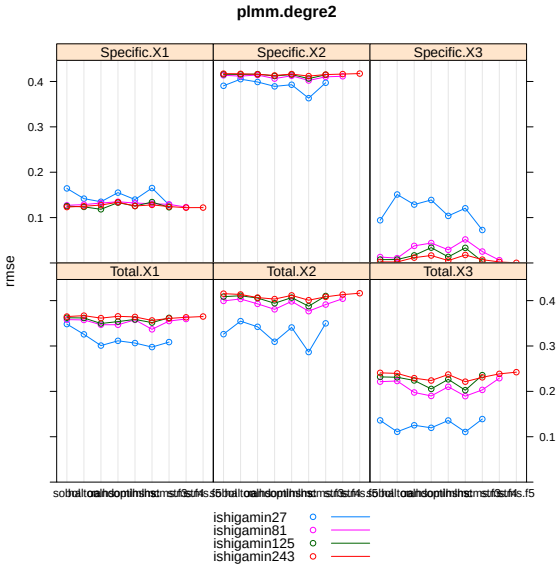


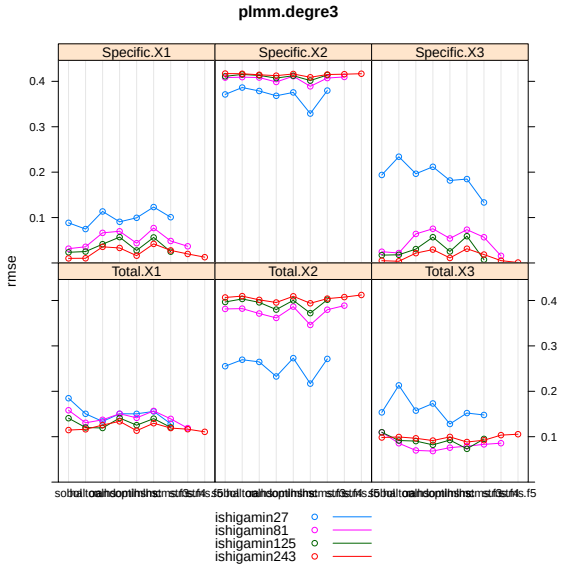

plmm.degree3
Specific.X1
Specific.X2
Specific.X3
Total.X1
Total.X2
Total.X3
rmse
ex3n27
ex3n81
ex3n125
ex3n243


sobol2002
First.X1
First.X2
First.X3
Total.X1
Total.X2
Total.X3
rmse
ishigamin27
ishigamin81
ishigamin125
ishigamin243


soboljansen
First.X1
First.X2
First.X3
Total.X1
Total.X2
Total.X3
rmse
ishigamin27
ishigamin81
ishigamin125
ishigamin243


sobol2002.PG.matern5_2.kriglineaire
First.X1
First.X2
First.X3
Total.X1
Total.X2
Total.X3
rmse
ishigamin27
ishigamin81
ishigamin125
ishigamin243


plmm.degree2
Specific.X1
Specific.X2
Specific.X3
Total.X1
Total.X2
Total.X3
rmse
ishigamin27
ishigamin81
ishigamin125
ishigamin243

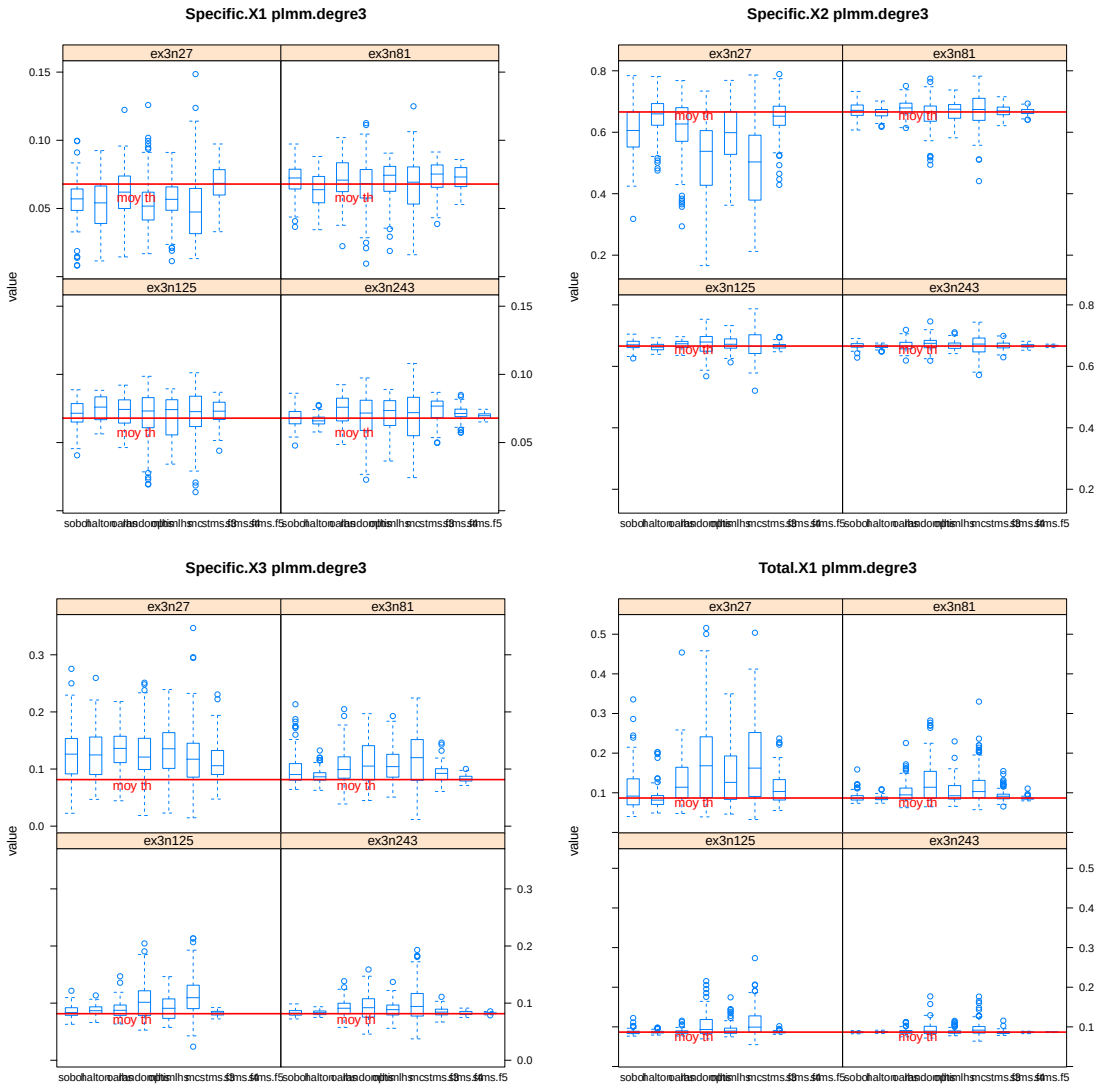

plmm.degree3
Specific.X1
Specific.X2
Specific.X3
Total.X1
Total.X2
Total.X3
rmse
ishigamin27
ishigamin81
ishigamin125
ishigamin243

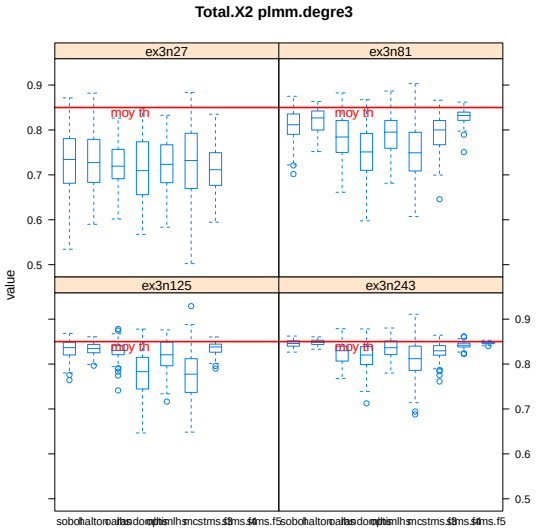
B.2 RMSE des estimateurs de la variance

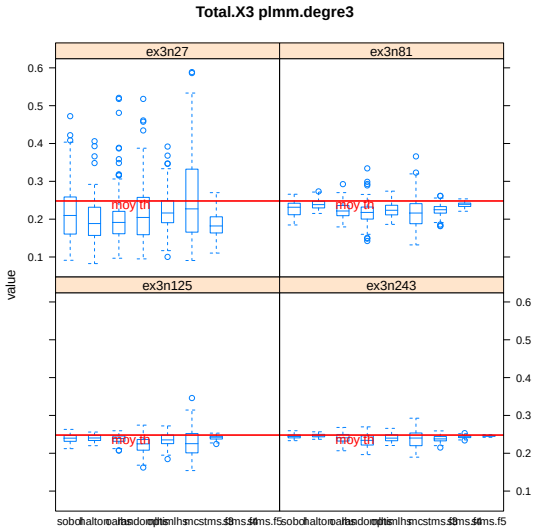
Annexe C

Quelques Boxplot & Diagrammes en barres

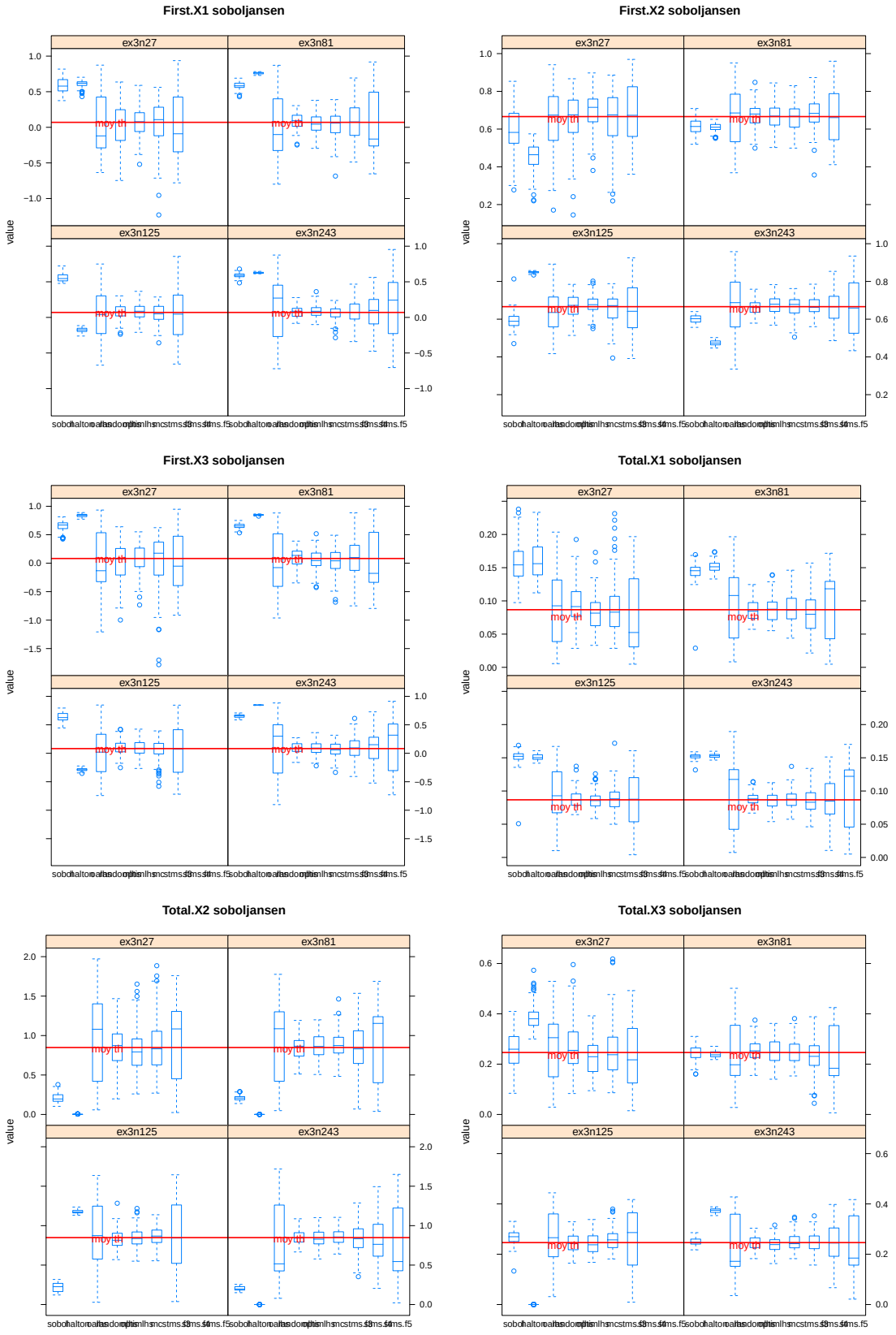
C.1 Modèle exemple3.fun, fonction plmm en degré 3



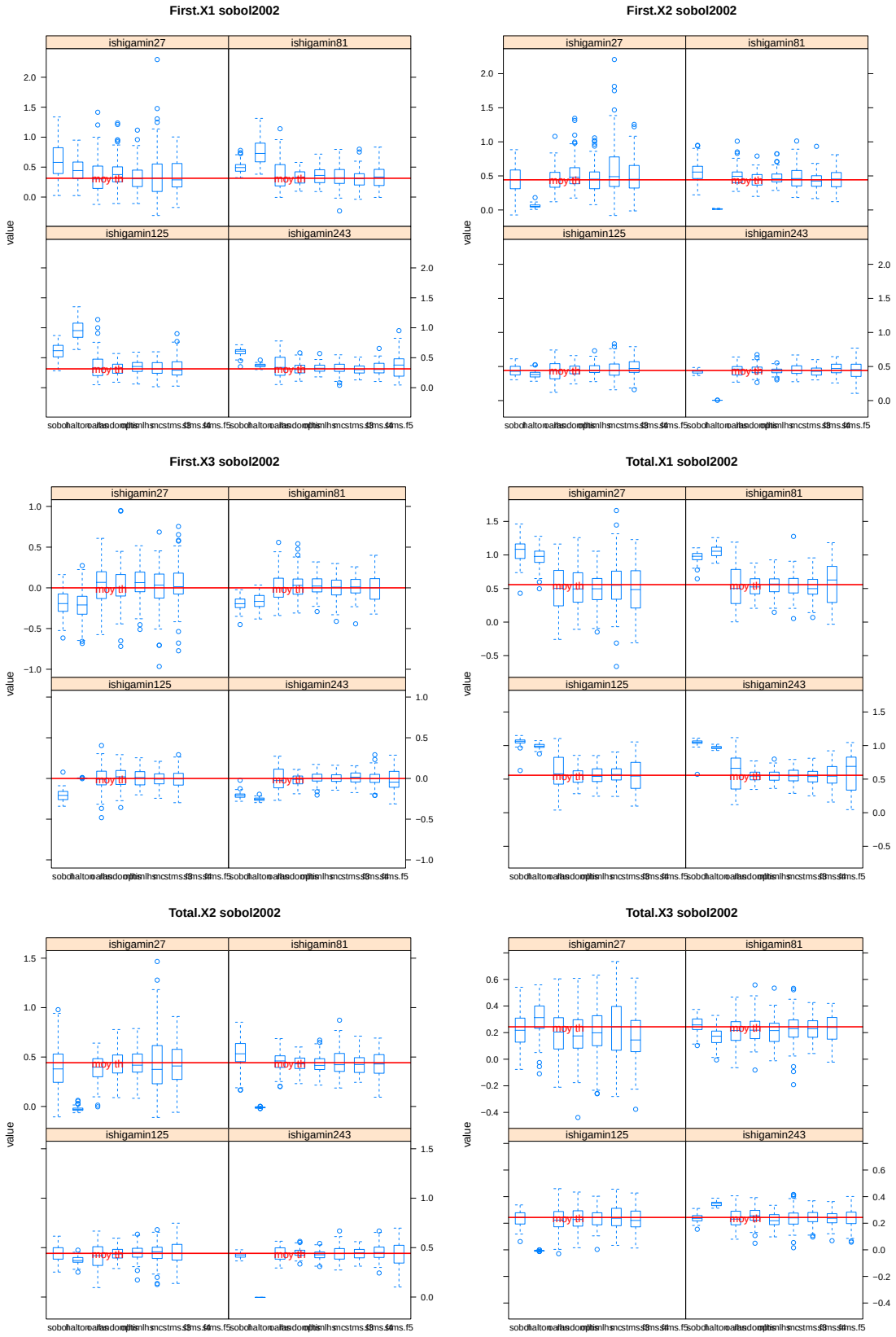

Total.X2 plmm.degree3
ex3n27
ex3n81
ex3n125
ex3n243
moy th
value
sobol halton mcs tms.f3 tms.f4 tms.f5


Total.X3 plmm.degree3
ex3n27
ex3n81
ex3n125
ex3n243
moy th
value
sobol halton mcs tms.f3 tms.f4 tms.f5

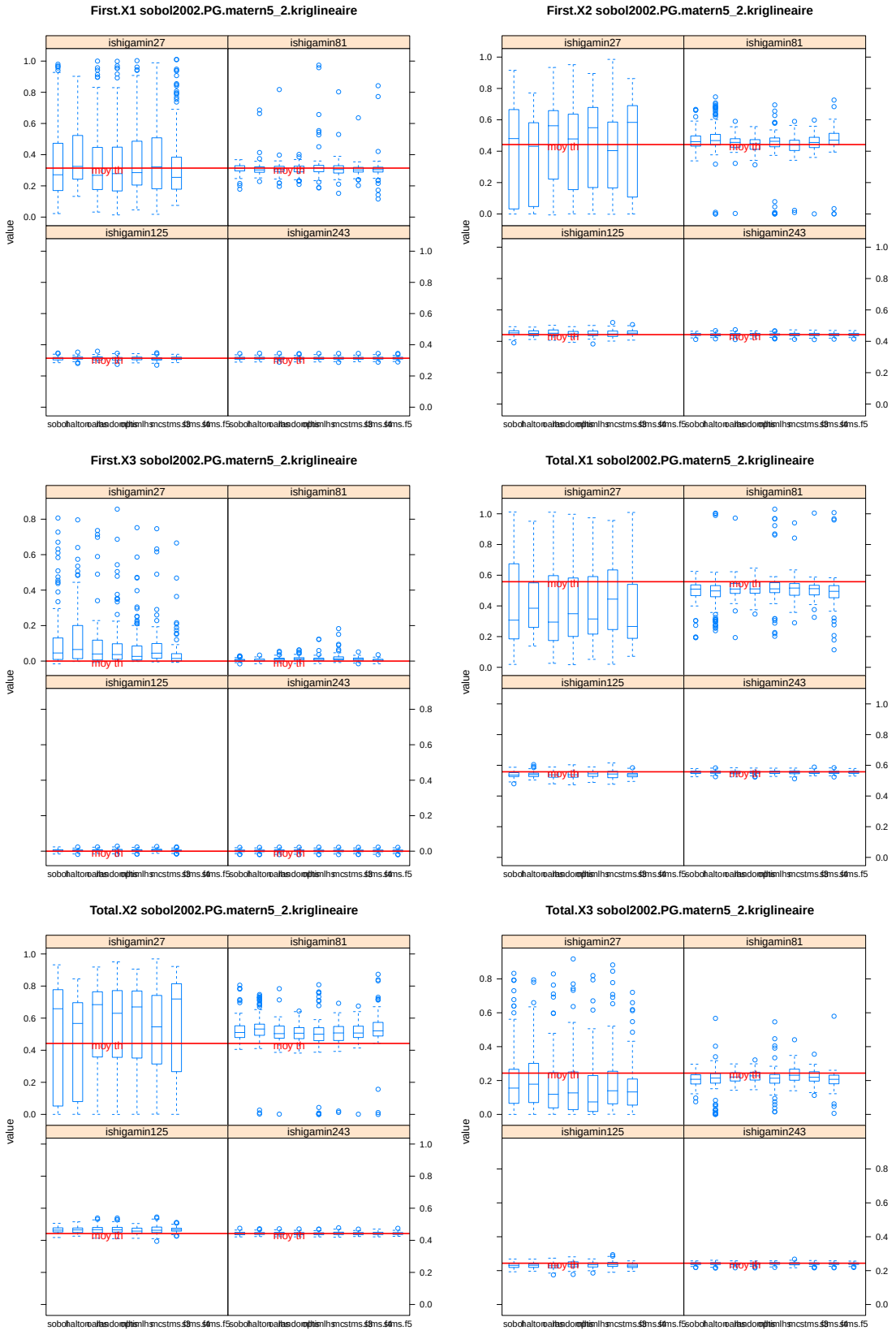
C.2 Modèle exemple3.fun, fonction soboljansen



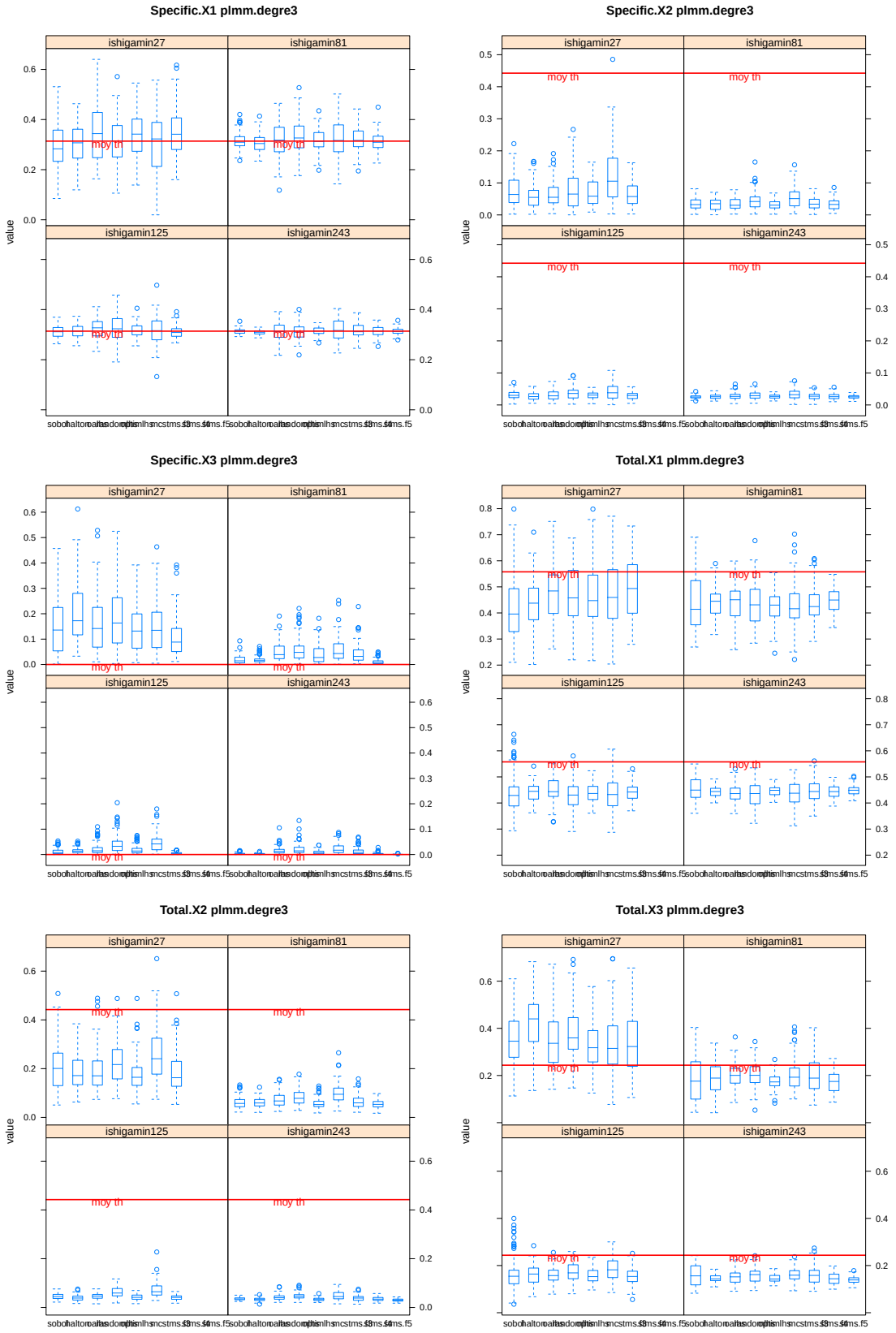
C.3 Modèle ishigami.fun, fonction sobol2002



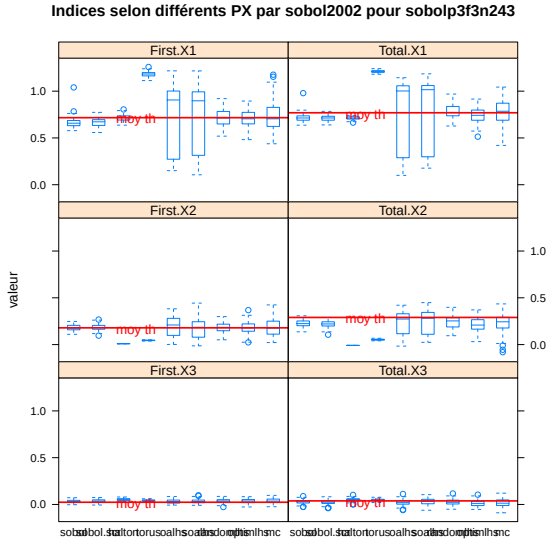
C.4 Ishigami.fun, sobol2002 & processus gaussiens

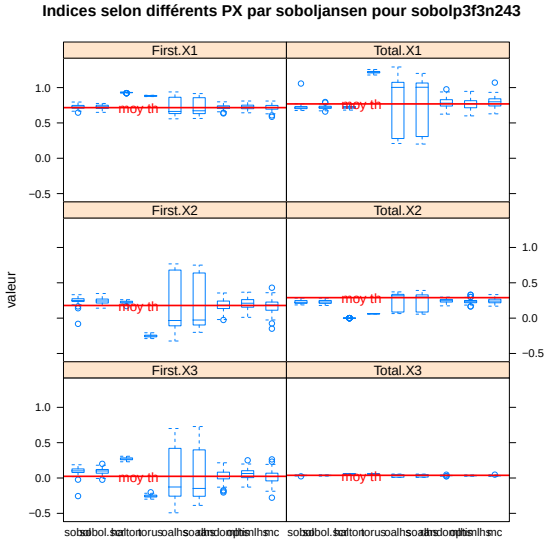


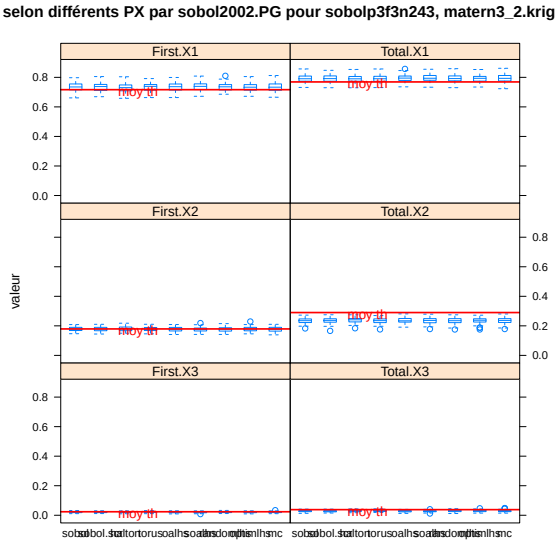
C.5 Modèle ishigami.fun, fonction plmm en degré 3

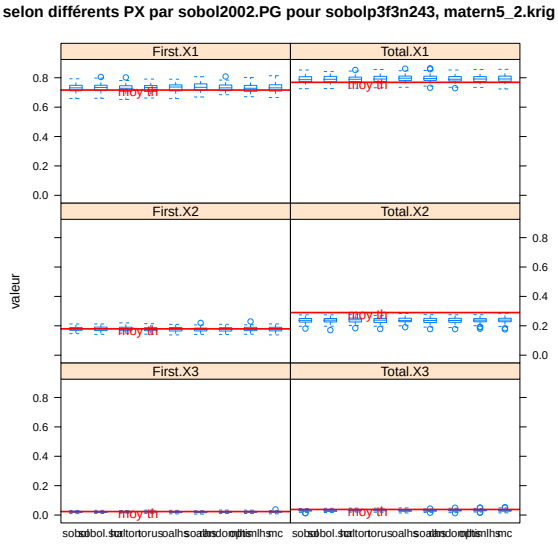


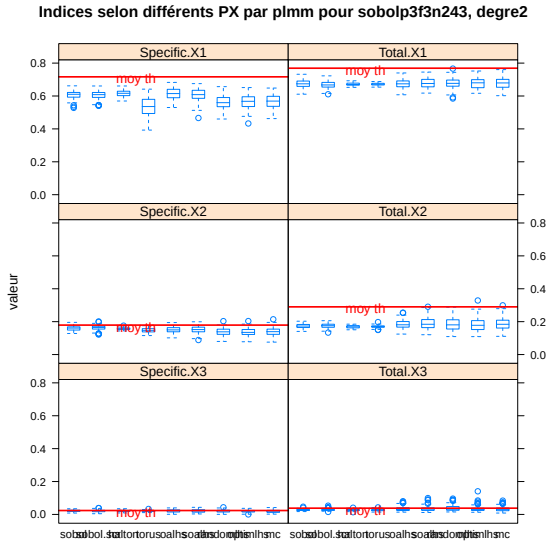
C.6 Sobol.fun, premiers boxplots, 243 observations

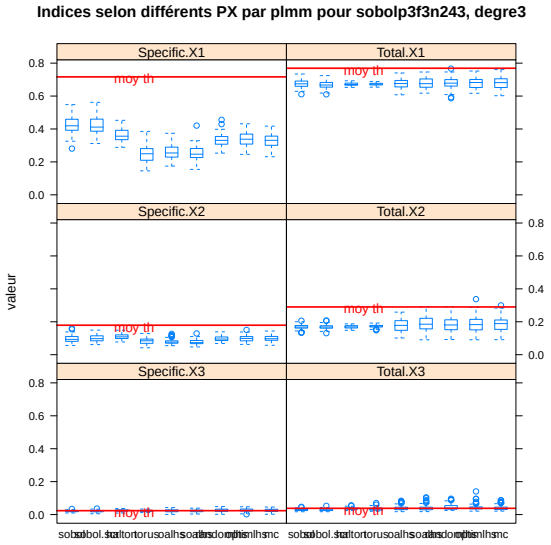




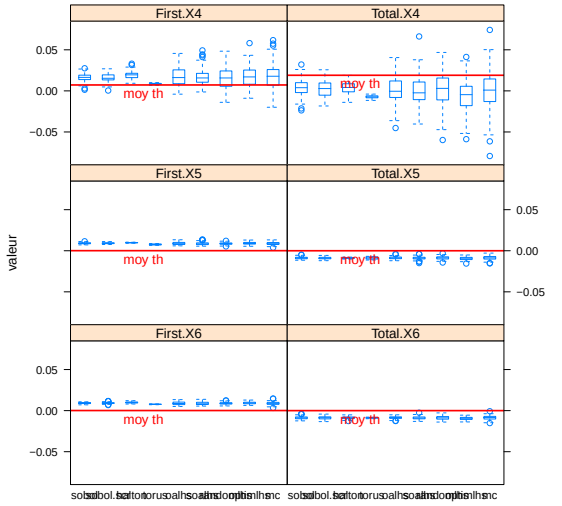




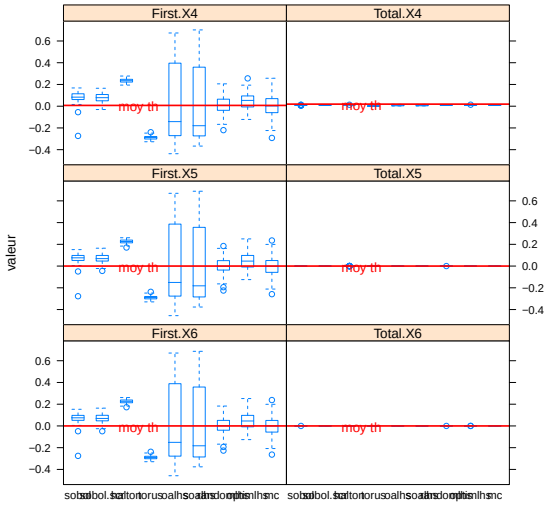




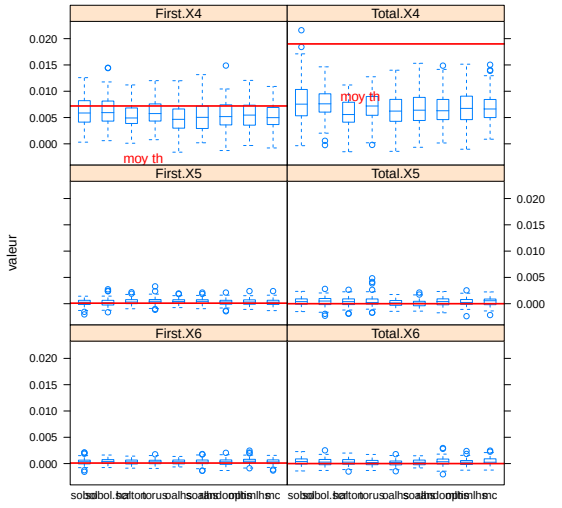
Indices selon différents PX par sobol2002 pour sobolp3f3n243



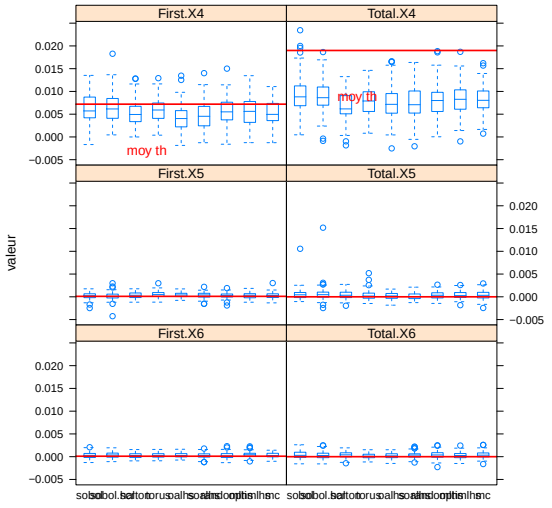
Indices selon différents PX par soboljansen pour sobolp3f3n243



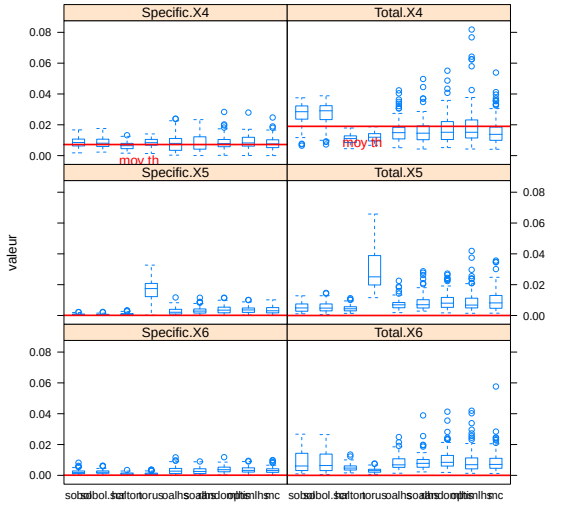
selon différents PX par sobol2002.PG pour sobolp3f3n243, matern3_2.krig o



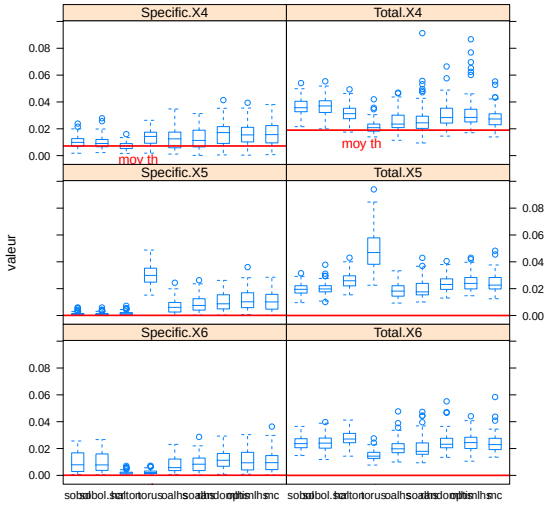
selon différents PX par sobol2002.PG pour sobolp3f3n243, matern5_2.krig o



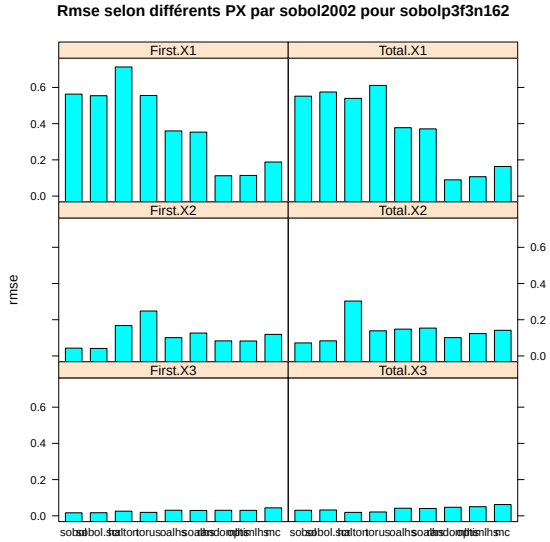
Indices selon différents PX par plmm pour sobolp3f3n243, degre2

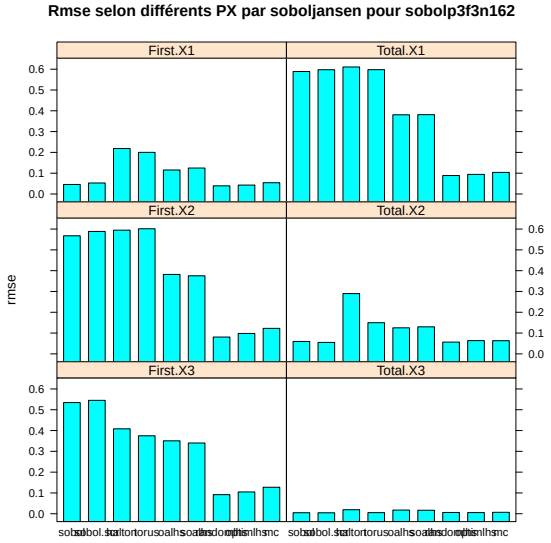


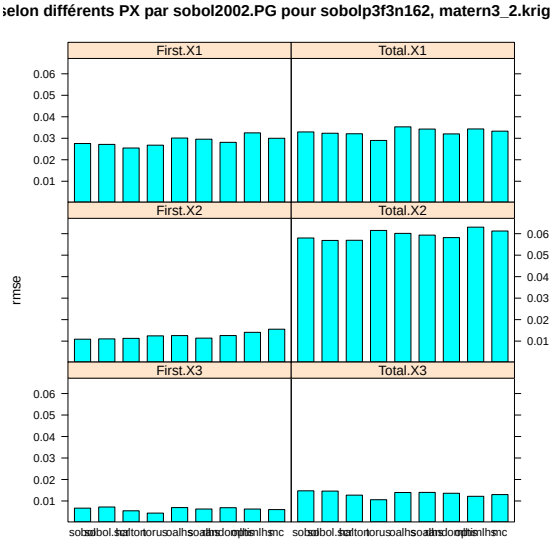
Indices selon différents PX par plmm pour sobolp3f3n243, degre3

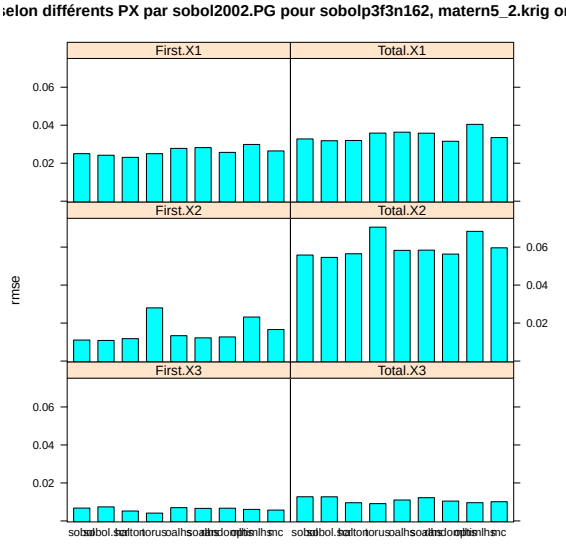


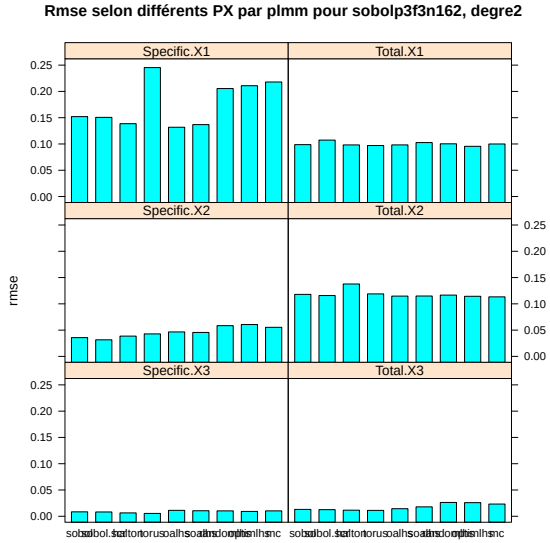
C.7 Sobol.fun, premiers RMSE, 162 observations

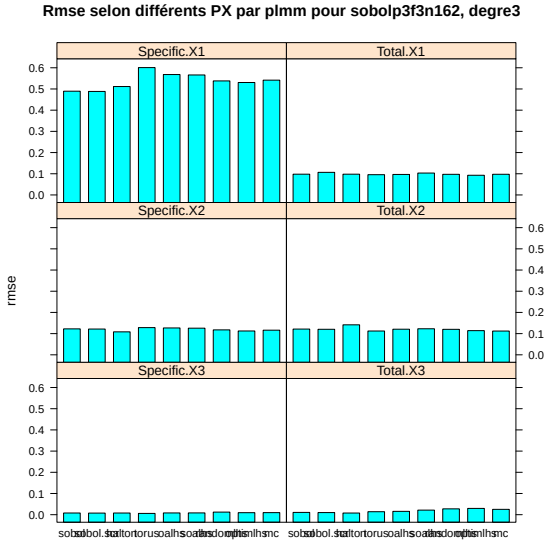


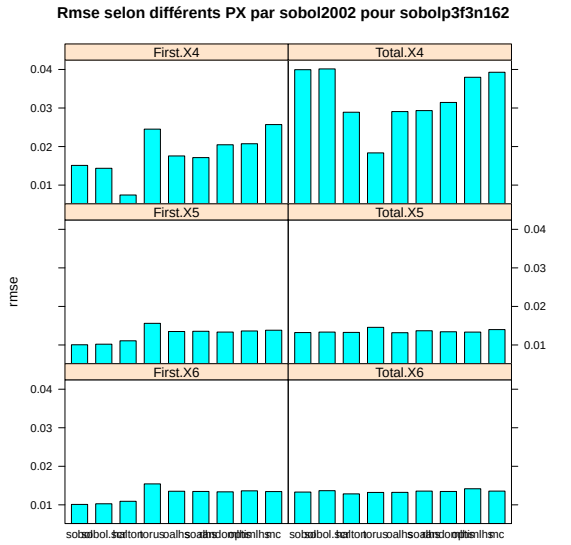


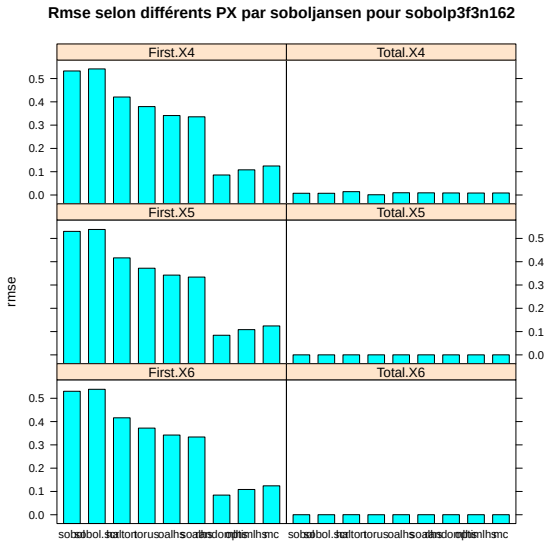


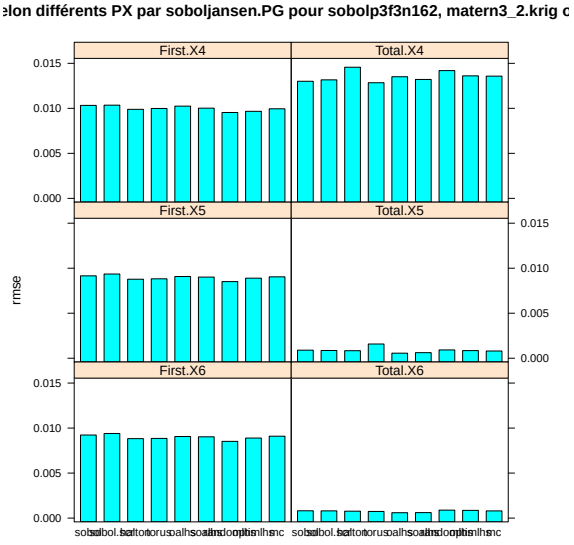


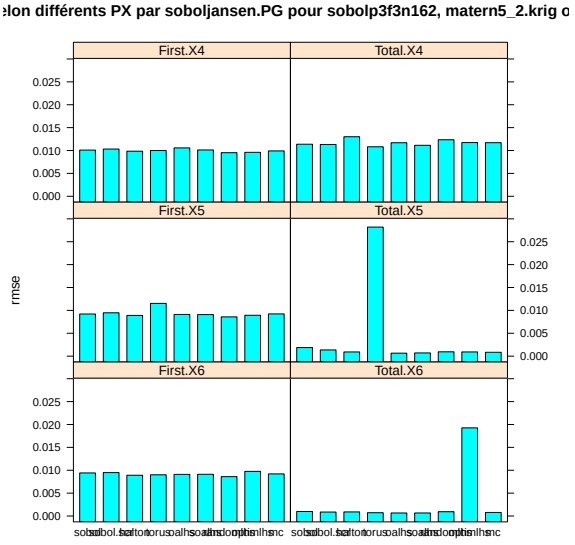


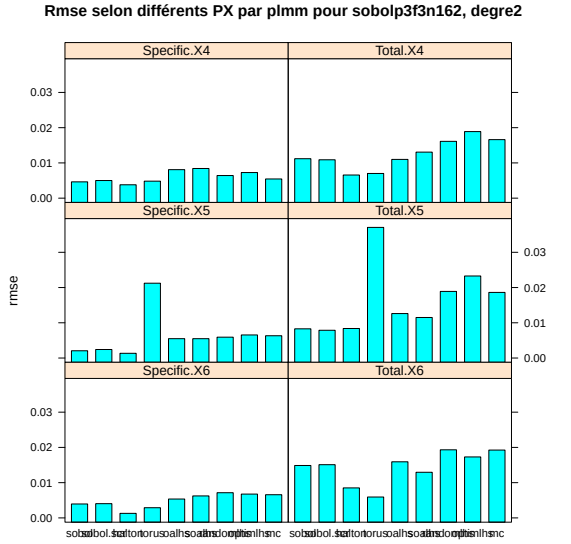


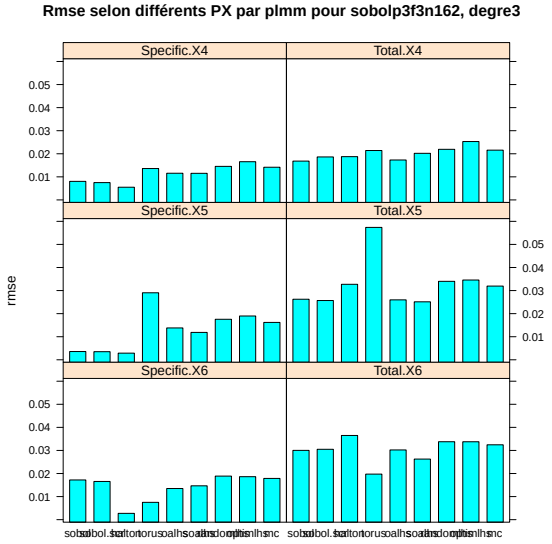

Rmse selon différents PX par sobol2002 pour sobolp3f3n162
First.X4
Total.X4
First.X5
Total.X5
First.X6
Total.X6
rmse


Rmse selon différents PX par soboljansen pour sobolp3f3n162
First.X4
Total.X4
First.X5
Total.X5
First.X6
Total.X6
rmse


elon différents PX par soboljansen.PG pour sobolp3f3n162, matern3_2.krig c
First.X4
Total.X4
First.X5
Total.X5
First.X6
Total.X6
rmse


elon différents PX par soboljansen.PG pour sobolp3f3n162, matern5_2.krig c
First.X4
Total.X4
First.X5
Total.X5
First.X6
Total.X6
rmse


Rmse selon différents PX par plmm pour sobolp3f3n162, degre2
Specific.X4
Total.X4
Specific.X5
Total.X5
Specific.X6
Total.X6
rmse


Rmse selon différents PX par plmm pour sobolp3f3n162, degre3
Specific.X4
Total.X4
Specific.X5
Total.X5
Specific.X6
Total.X6
rmse

Résumé :

Ce stage d'une durée de 12 semaines s'est déroulé au sein de l'INRA de Toulouse, un organisme public de recherche agronomique. J'ai été affectée à l'unité Mathématiques et Informatique Appliquées de Toulouse, dont l'objectif est de permettre à l'INRA de disposer des compétences et des outils en mathématiques et informatique appliquées, notamment en collaborant avec les autres départements de recherche.

De plus en plus de modèles sont développés mais ils sont complexes et coûteux en tant de calcul. Afin de les analyser, nous pouvons avoir recours à l'analyse de sensibilité, ainsi qu'au préalable à la métamodélisation. Pour effectuer ces études, il est souvent nécessaire de disposer d'un plan d'expérience, c'est-à-dire d'un petit nombre d'observations à partir desquelles les sorties du modèle sont simulées.

Un des objectifs de mon stage a été d'améliorer les estimations des indices de sensibilité des effets d'interaction entre facteurs d'entrée d'un modèle. J'ai comparé la précision des estimateurs d'indice de sensibilité en fonction du plan d'expérience utilisé. Pour cela, j'ai analysé l'efficacité du plan d'expérience choisi sur la qualité des estimateurs obtenus à partir de ce plan. Enfin, j'ai analysé une technique de métamodélisation : les processus gaussiens. Toutes ces analyses ont été effectuées à partir de résultats générés avec le logiciel libre **R**. J'ai développé de nombreux scripts permettant de générer les différents indicateurs nécessaires aux analyses, ainsi que les scripts permettant d'organiser ces résultats sous une forme graphique, bien plus facile à analyser.

Ce stage a ainsi permis de commencer à développer un nouvel estimateur des effets d'interaction d'un modèle complexe. De plus, nous avons découvert l'efficacité des processus gaussiens, quel que soit le plan d'expérience utilisé. Enfin, nous avons commencé à analyser l'influence des différents plans d'expérience utilisés sur la précision des estimateurs des indices de sensibilité, indépendamment du métamodèle utilisé ; notamment nous avons constaté l'efficacité d'un plan particulier : le plan tms-net.

Grâce à ce stage, j'ai découvert le domaine de la recherche. Il m'a permis d'apprendre à développer une méthodologie de programmation, avec plus d'efficacité et de rigueur, ainsi que des méthodes afin d'analyser des résultats graphiques. J'ai amélioré ma façon de travailler, ainsi que d'appréhender tout type de problème.

Mots-clés : analyse de sensibilité, krigeage, LHS, métamodèle, plan d'expérience, planification d'expériences, processus gaussiens, Sobol, tms-net.