Ré-échantillonage en apprentissage automatique

Fabrice Rossi

Université Paris 1 Panthéon Sorbonne

2018

Plan

Introduction

Compromis biais variance

Leave-one-out

Validation croisée

Bootstrap

Données structurées

Plan

Introduction

Compromis biais variance

Leave-one-out

Validation croisée

Bootstrap

Données structurées

Estimation de performances

Problème fondamental de l'apprentissage

- détermination des paramètres d'un modèle
- comparaison de modèles (méta-paramètres)
- prévision du comportement futur

Difficile

- estimation biaisée : le meilleur modèle a toujours les meilleures performances
- données = vrai modèle + bruit : un modèle « trop bon » apprend le bruit
- garanties mathématiques trop lâches

Solution simple

Découpage des données

- ensemble d'apprentissage : estimation des paramètres
- ensemble de validation : comparaison de modèles (réglage des méta-paramètres)
- ensemble de test : évaluation finale du modèle retenu

Limitations

- nécessite d'être riche en données
- compromis entre les qualités des différentes estimations (paramètres vs méta-paramètres vs prévision des performances futures)
- effets du découpage

Solutions plus efficaces

Ré-échantillonage

- classe de méthodes basées sur des tirages aléatoires dans les données
- généralisation du principe de découpage (notamment)
- richesse en données remplacée par richesse en puissance de calcul

Nombreuses variantes et extensions

- cas extrême : *leave-one-out* (et *Jackknife*)
- *bootstrap*
- validation croisée (et ses variantes)
- tests empiriques (*permutation test*)

Plan

Introduction

Compromis biais variance

Leave-one-out

Validation croisée

Bootstrap

Données structurées

Cadre mathématique

Hypothèses classiques

- variables explicatives à valeurs dans $\mathcal{X}$
- variable à expliquer à valeurs dans $\mathcal{Y}$
- phénomène sous-jacent : un couple de variables aléatoires (X, Y) à valeurs dans $\mathcal{X} \times \mathcal{Y}$, de loi $\mathcal{P}$
- un jeu de données : $\mathcal{D} = (X_i, Y_i)_{1 \leq i \leq N}$, avec $(X_i, Y_i) \sim \mathcal{P}$
- les observations sont supposées indépendantes (en plus d'être identiquement distribuées)

Qualité d'un modèle

- fonction de perte l de $\mathcal{Y}^2$ dans $\mathbb{R}^+$
- risque $L(g) = E_{(X,Y)\sim\mathcal{P}}\{l(g(X), Y)\}$
- modèle optimal $g^* = \arg\min_g L(g)$ et risque optimal $L^* = \inf_g L(g)$

Approximation et bruit

Cas de la régression

- perte quadratique : $l_2(p,t) = (p-t)^2$
- modèle optimal : $g^*(x) = E_{(X,Y)\sim\mathcal{P}}\{Y|X=x\}$
- décomposition de l'erreur

$$L_2(g) = \underbrace{E_{(X,Y)\sim\mathcal{P}}\{(g(X)-g^*(X))^2\}}_{\text{approximation}} + \underbrace{E_{(X,Y)\sim\mathcal{P}}\{(Y-g^*(X))^2\}}_{\text{bruit}}$$

Discrimination

- perte 0/1 : $l(p,t) = \mathbb{I}_{p\neq t}$
- modèle optimal : $g^*(x) = \arg\min_{y\in\mathcal{Y}} \mathbb{P}_{(X,Y)\sim\mathcal{P}}(Y=y|X=x)$
- résultat similaire avec une mesure d'approximation plus complexe

Difficulté : coller au modèle optimal sans coller au bruit

Compromis biais/variance

Plusieurs jeux de données

- $\mathcal{D}$ distribué selon $\mathcal{P}^N$
- un modèle par jeu de données : $g_{\mathcal{D}}$
- décomposition ponctuelle :

$$E_{\mathcal{D}\sim\mathcal{P}^N}\{l_2(g_{\mathcal{D}}(x),g^*(x))\} = \underbrace{(E_{\mathcal{D}'\sim\mathcal{P}^N}\{g_{\mathcal{D}'}(x)\} - g^*(x))^2}_{\text{biais}^2}$$
$$+ \underbrace{E_{\mathcal{D}\sim\mathcal{P}^N}\{(g_{\mathcal{D}}(x) - E_{\mathcal{D}'\sim\mathcal{P}^N}\{g_{\mathcal{D}'}(x)\})^2\}}_{\text{variance}}$$
$$+ \underbrace{E_{(X,Y)\sim\mathcal{P}}\{(Y - g^*(X))^2|X = x\}}_{\text{bruit}}$$

En pratique

- modèle simple : fort biais et faible variance
- modèle complexe : faible biais et forte variance

Illustration numérique

Données

- x_i déterministe : grille régulière sur $[-\pi,\pi]$, 51 points
- $Y_i = \sin(x_i) + \varepsilon_i$, avec les ε_i i.i.d. $\mathcal{N}(0,\sigma^2)$, avec $\sigma = 0.2$
- 50 répétitions de l'expérience

Exemple

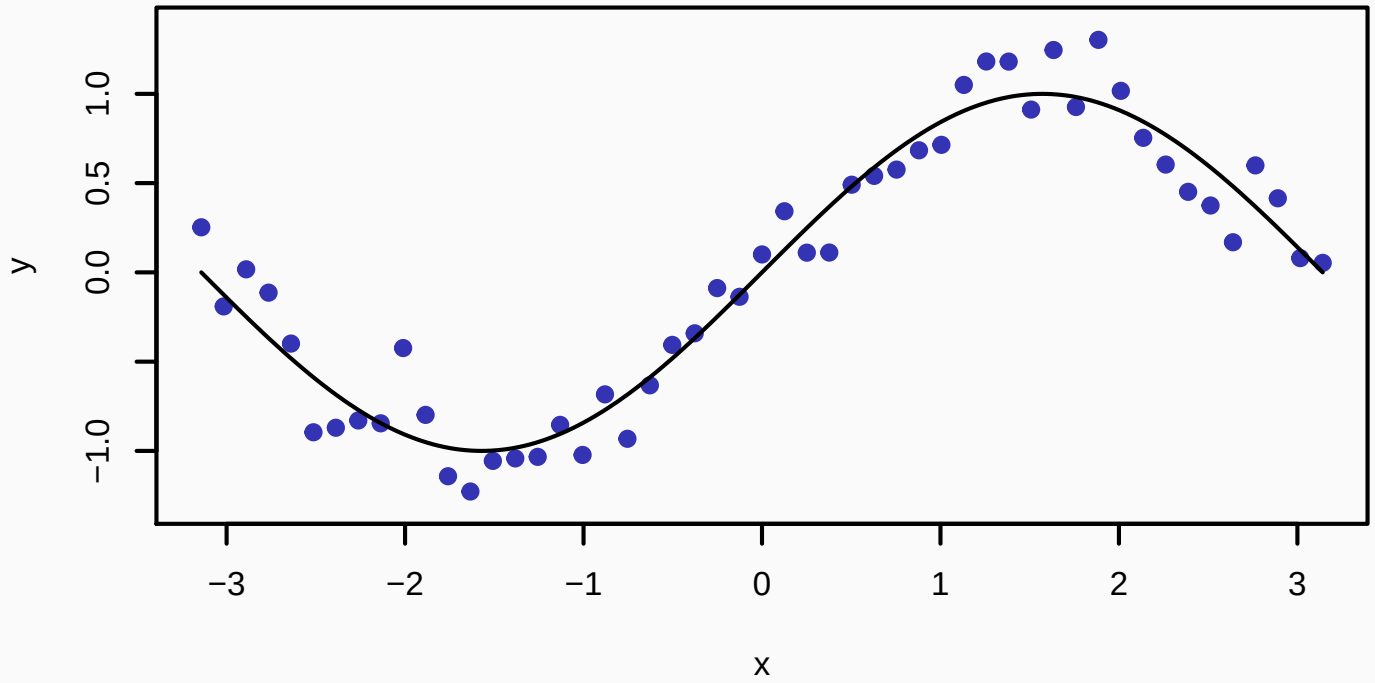
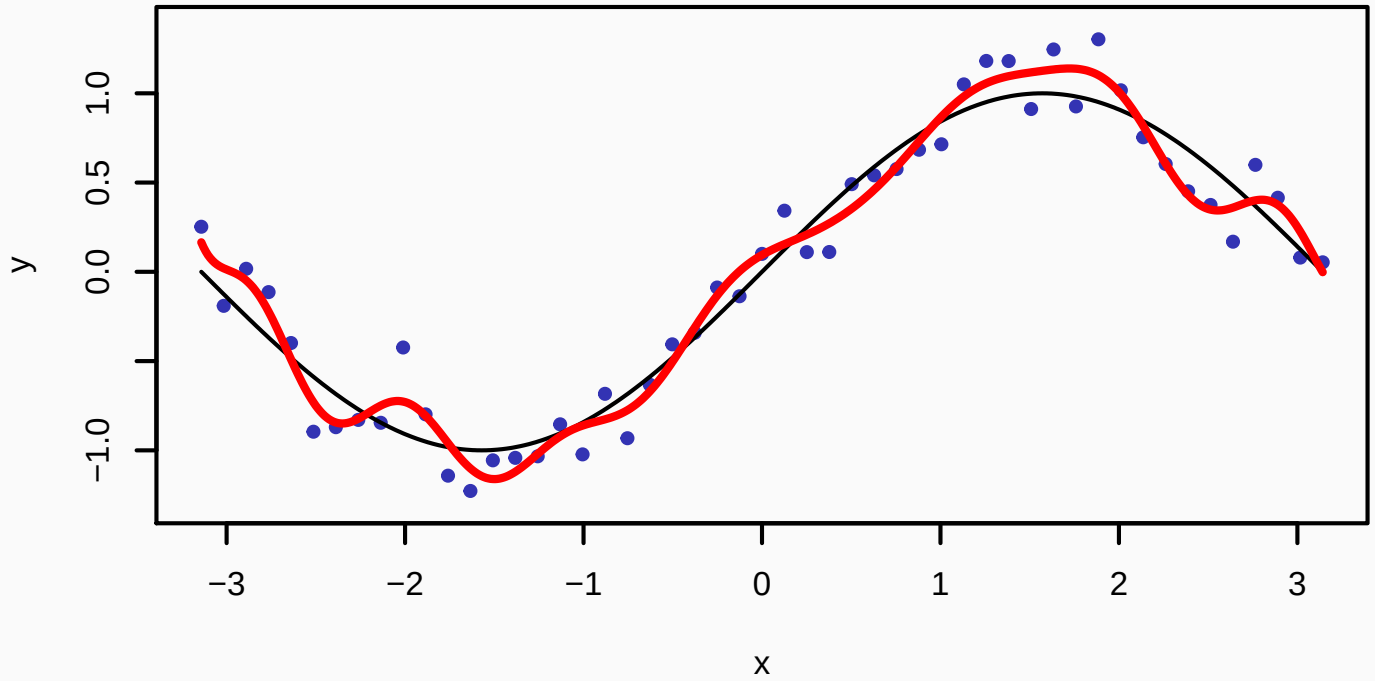


Illustration numérique

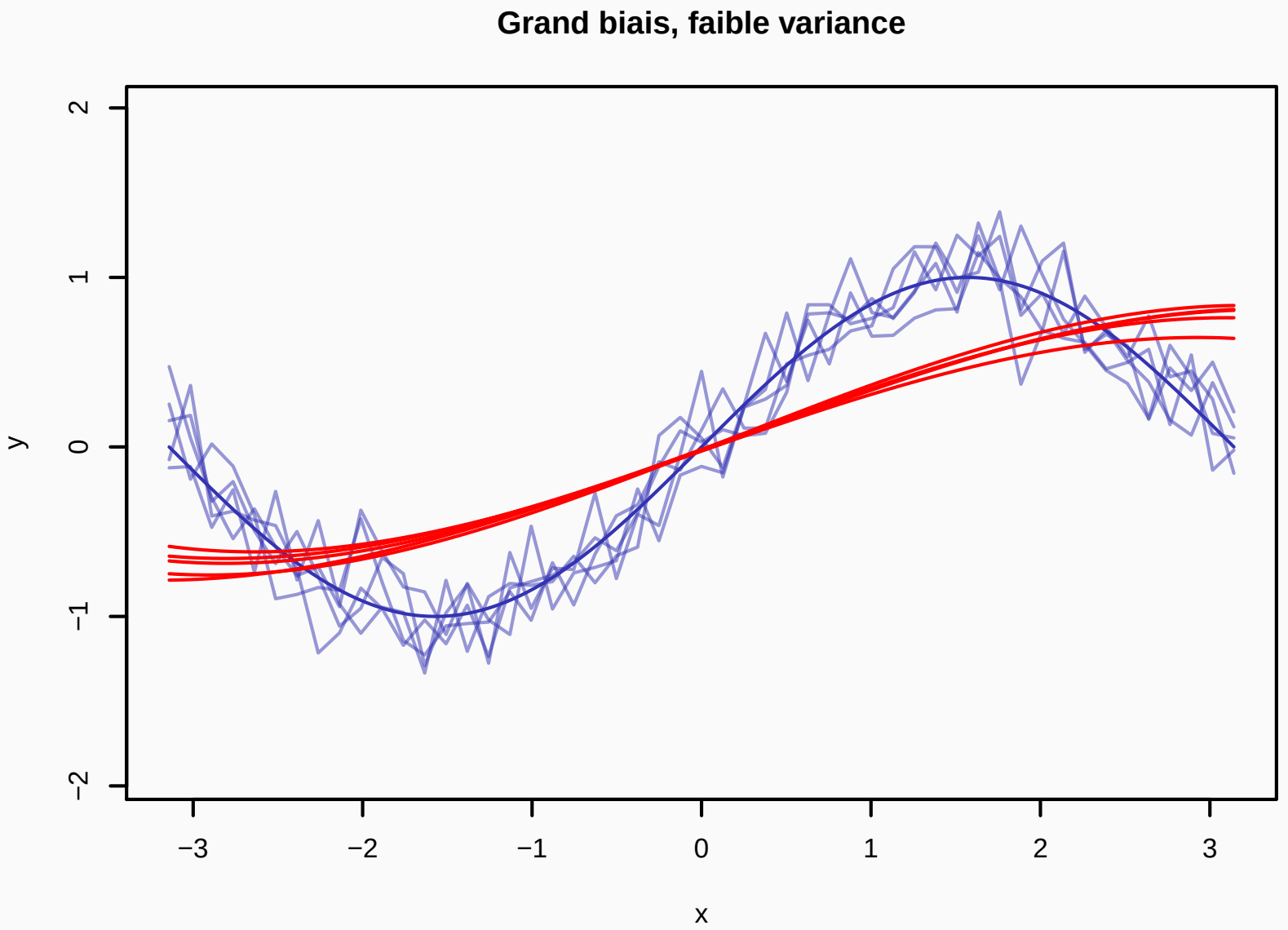
Modèle

- *Kernel ridge regression* avec un noyau gaussien
- $g(x) = \sum_{i=1}^{N} \alpha_i \exp(-\sigma(x - x_i)^2)$
- paramètre de régularisation fixé à une valeur relativement faible
- on montre l'effet de σ

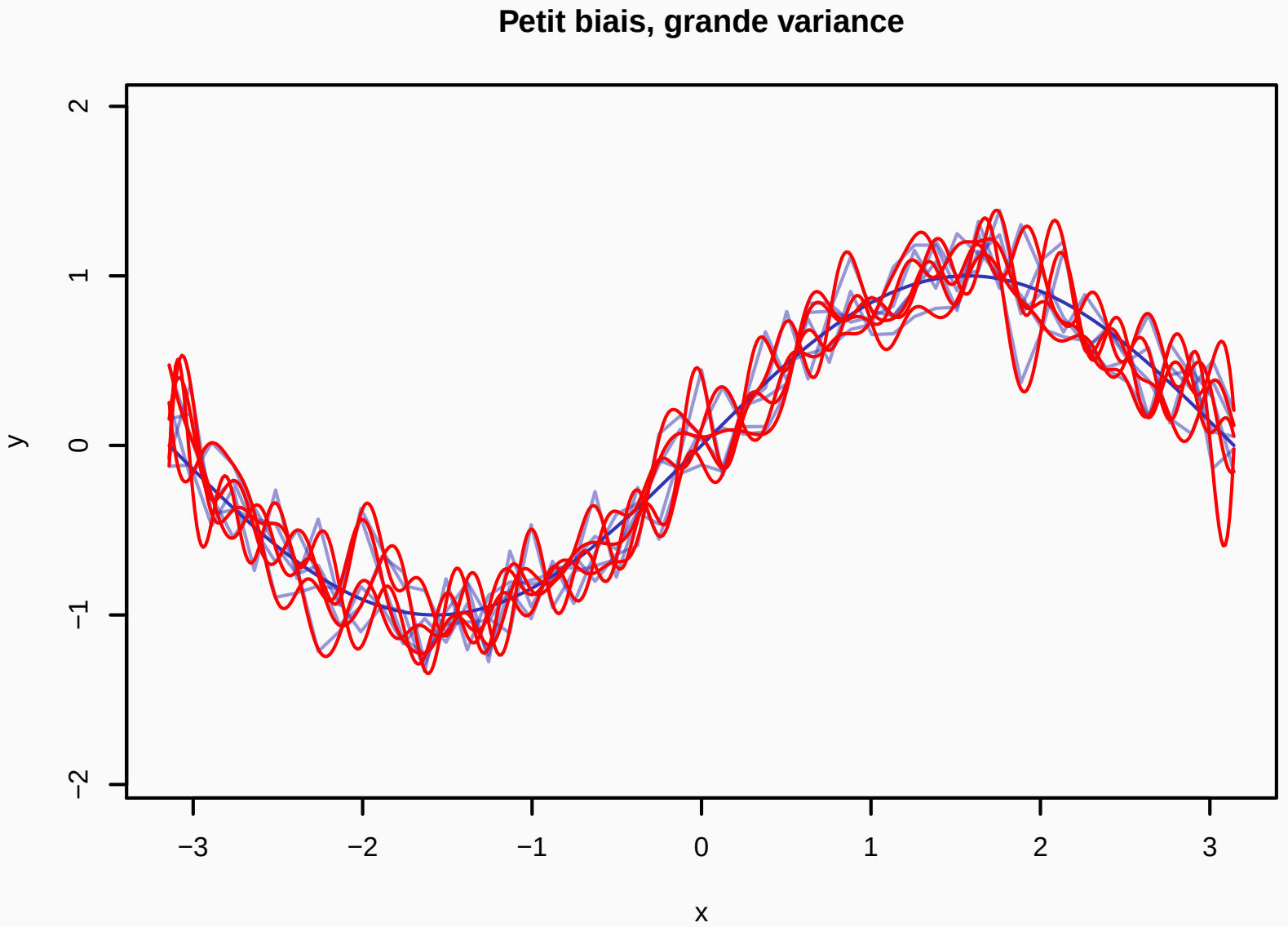
Example



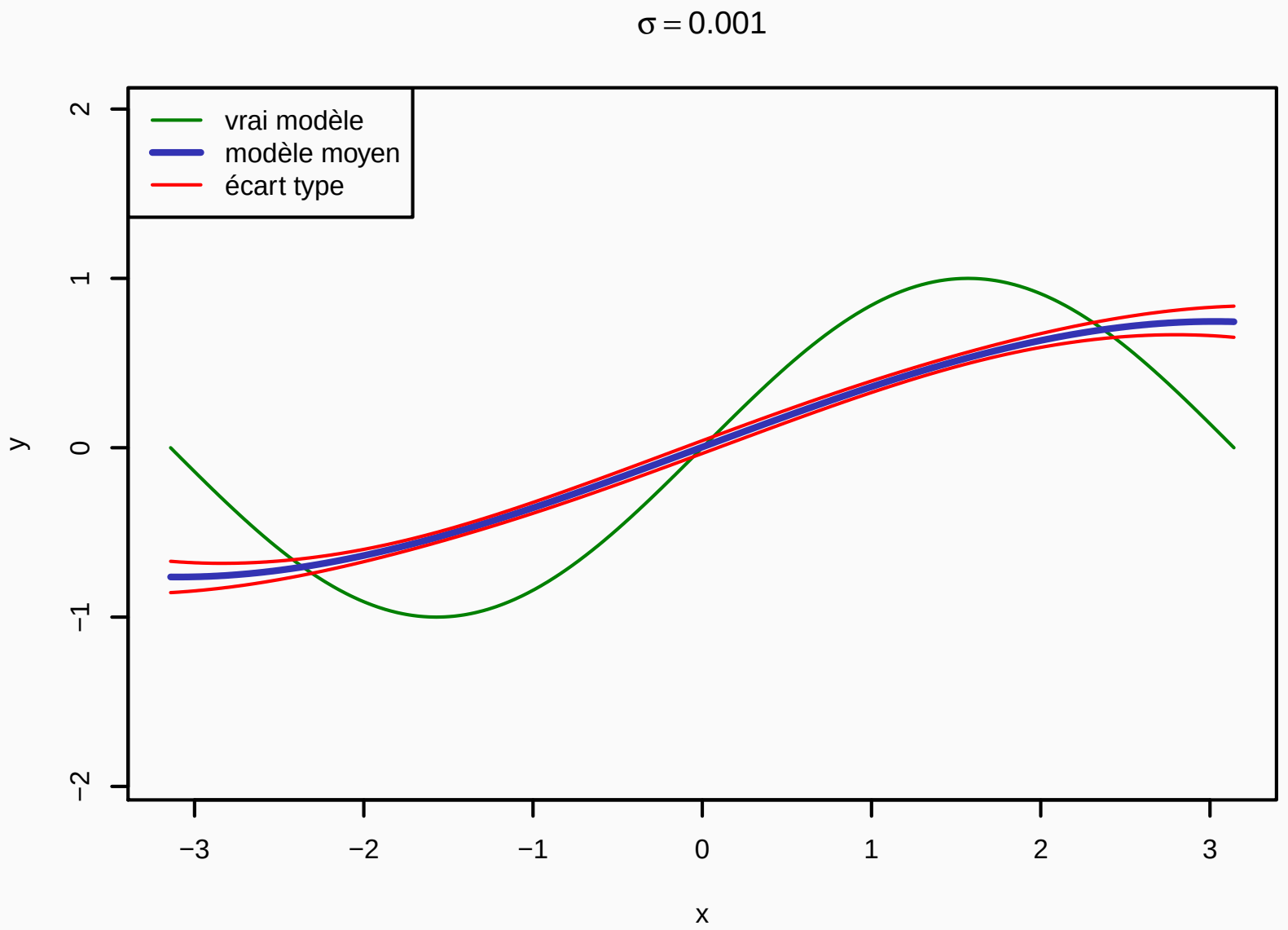
Exemple



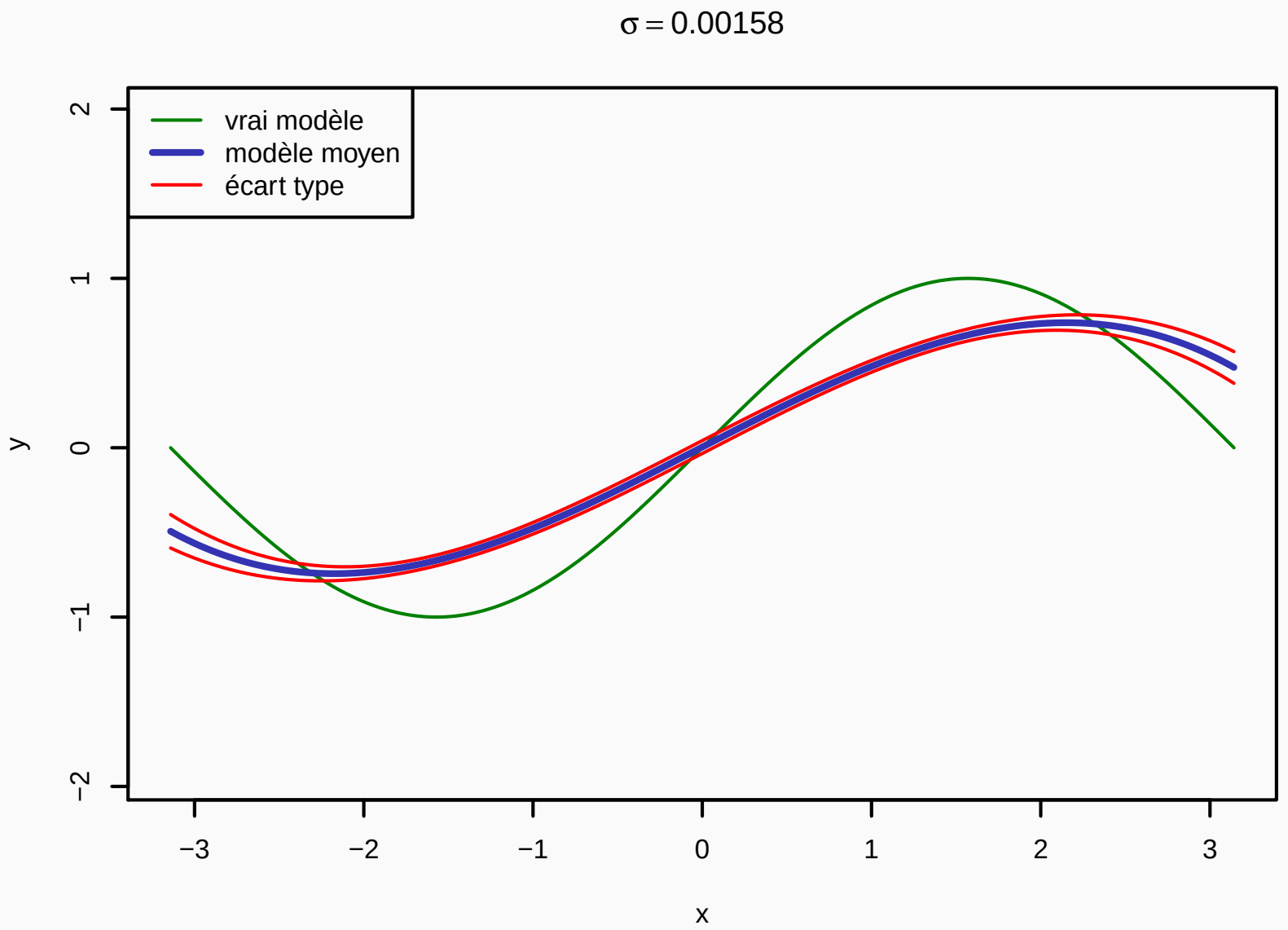
Exemple



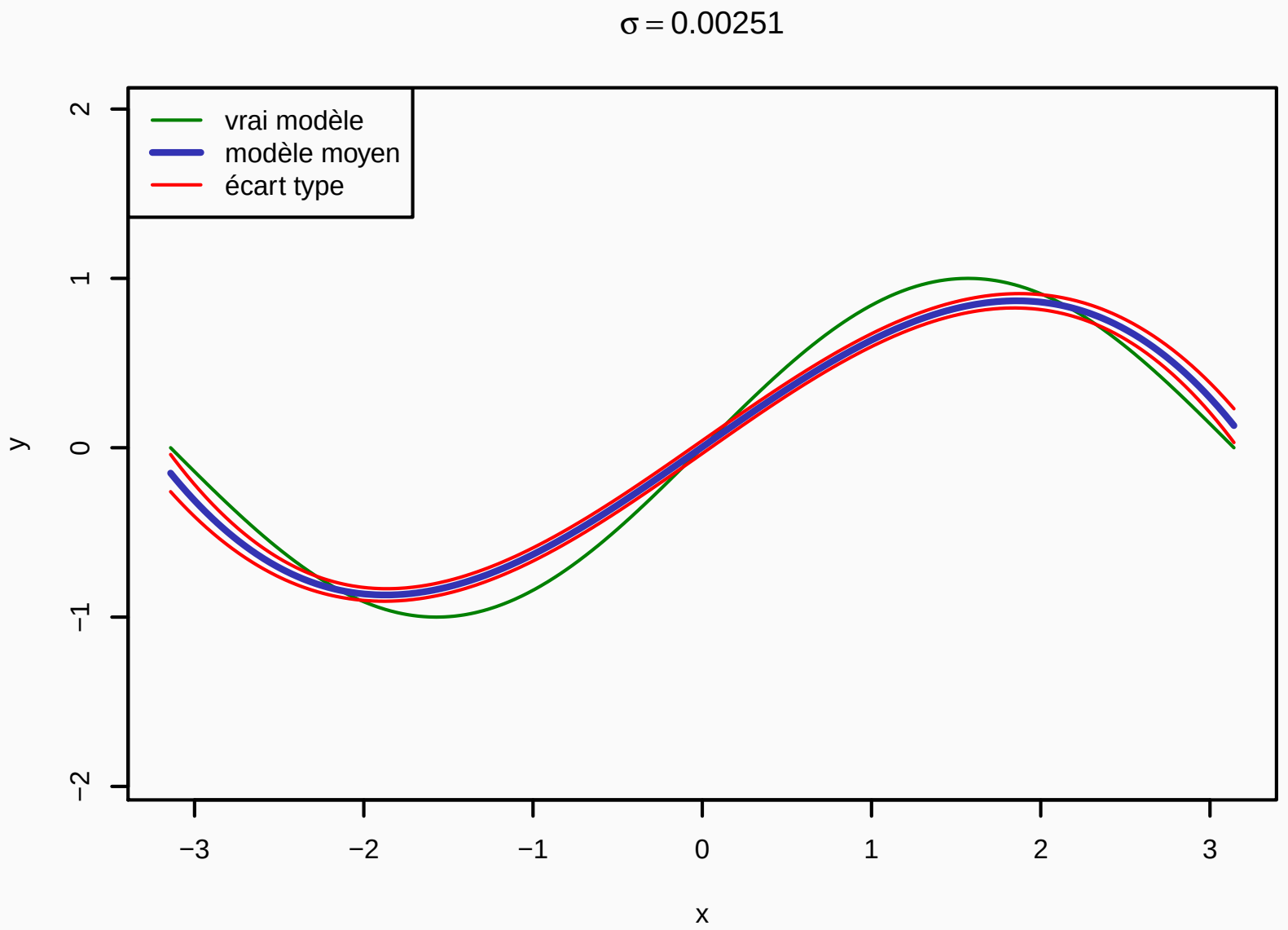
Résultats



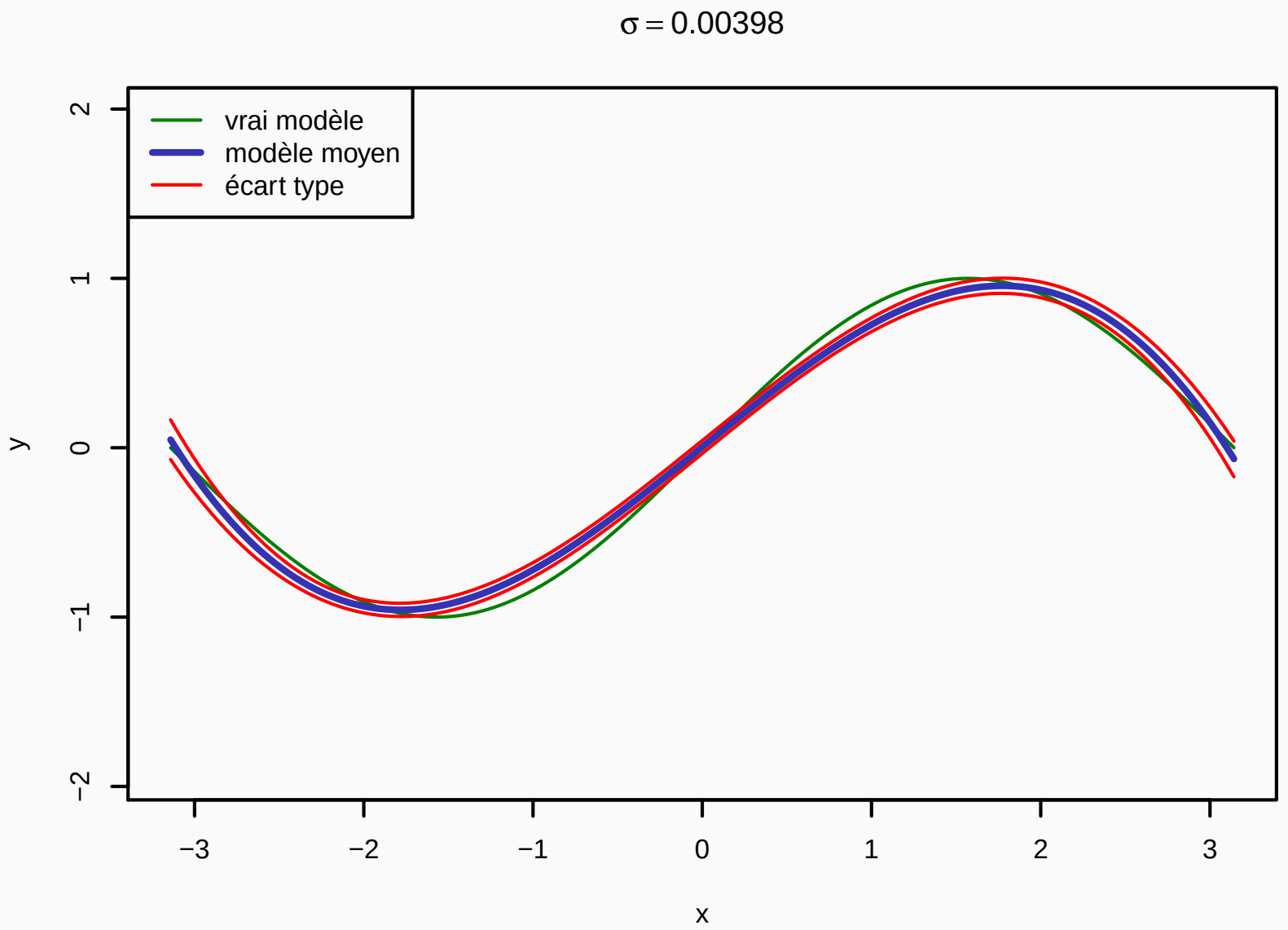
Résultats


σ = 0.00158
vrai modèle
modèle moyen
écart type
y
x
2
1
0
−1
−2
−3
−2
−1
0
1
2
3

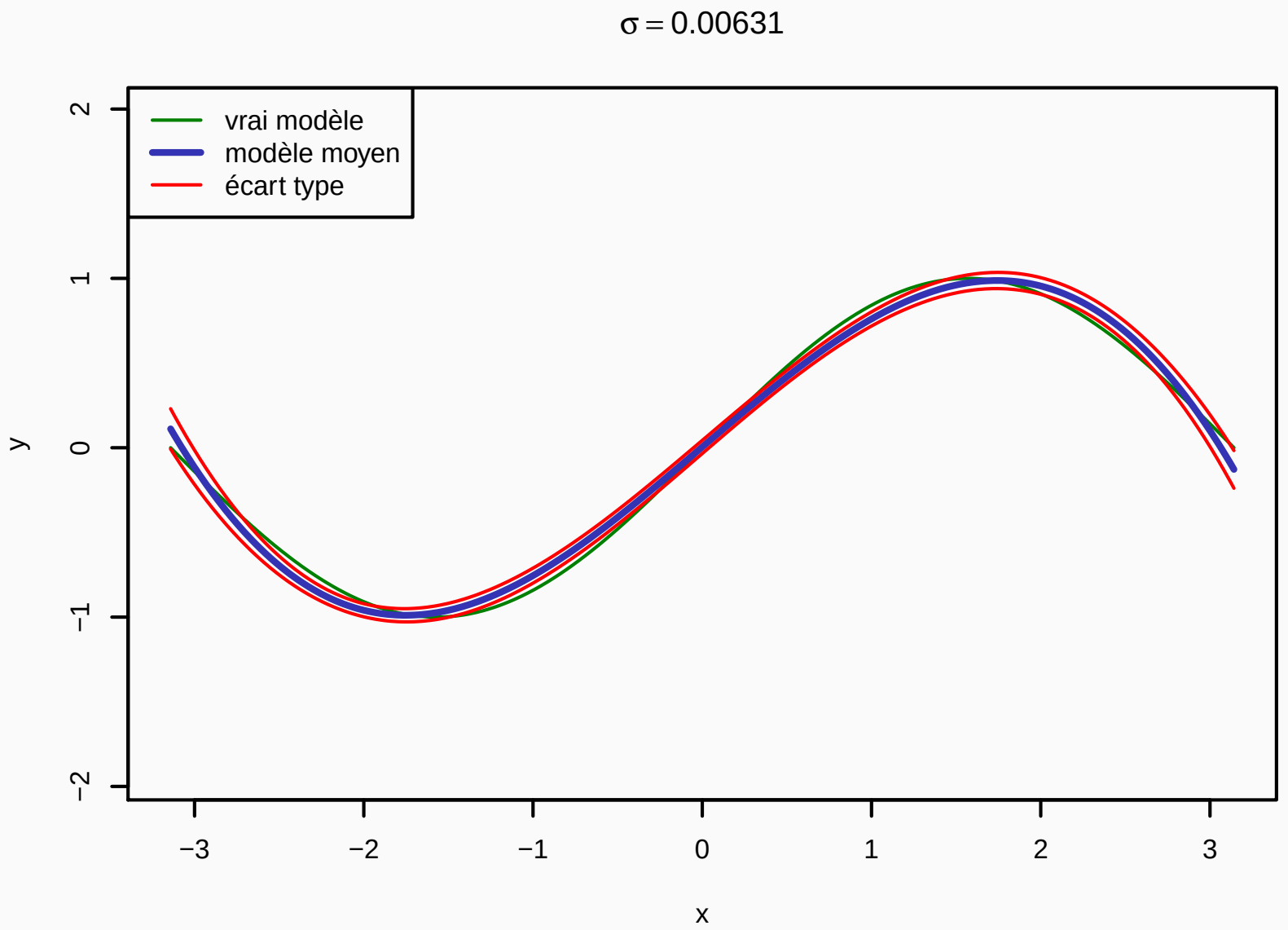
Résultats


σ = 0.00251
vrai modèle
modèle moyen
écart type
y
x
2
1
0
−1
−2
−3
−2
−1
0
1
2
3

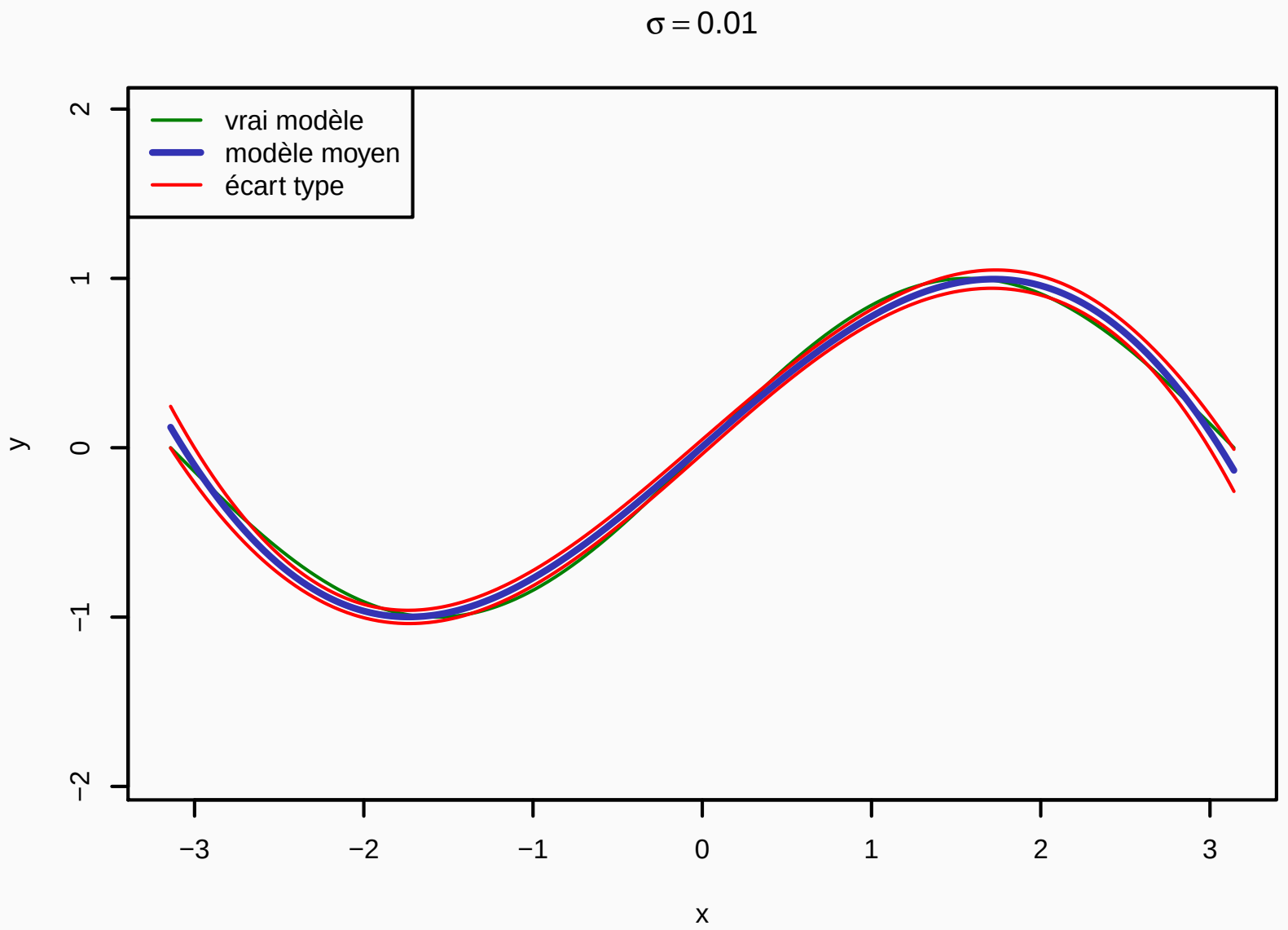
Résultats


σ = 0.00398
vrai modèle
modèle moyen
écart type
y
x
2
1
0
−1
−2
−3
−2
−1
0
1
2
3

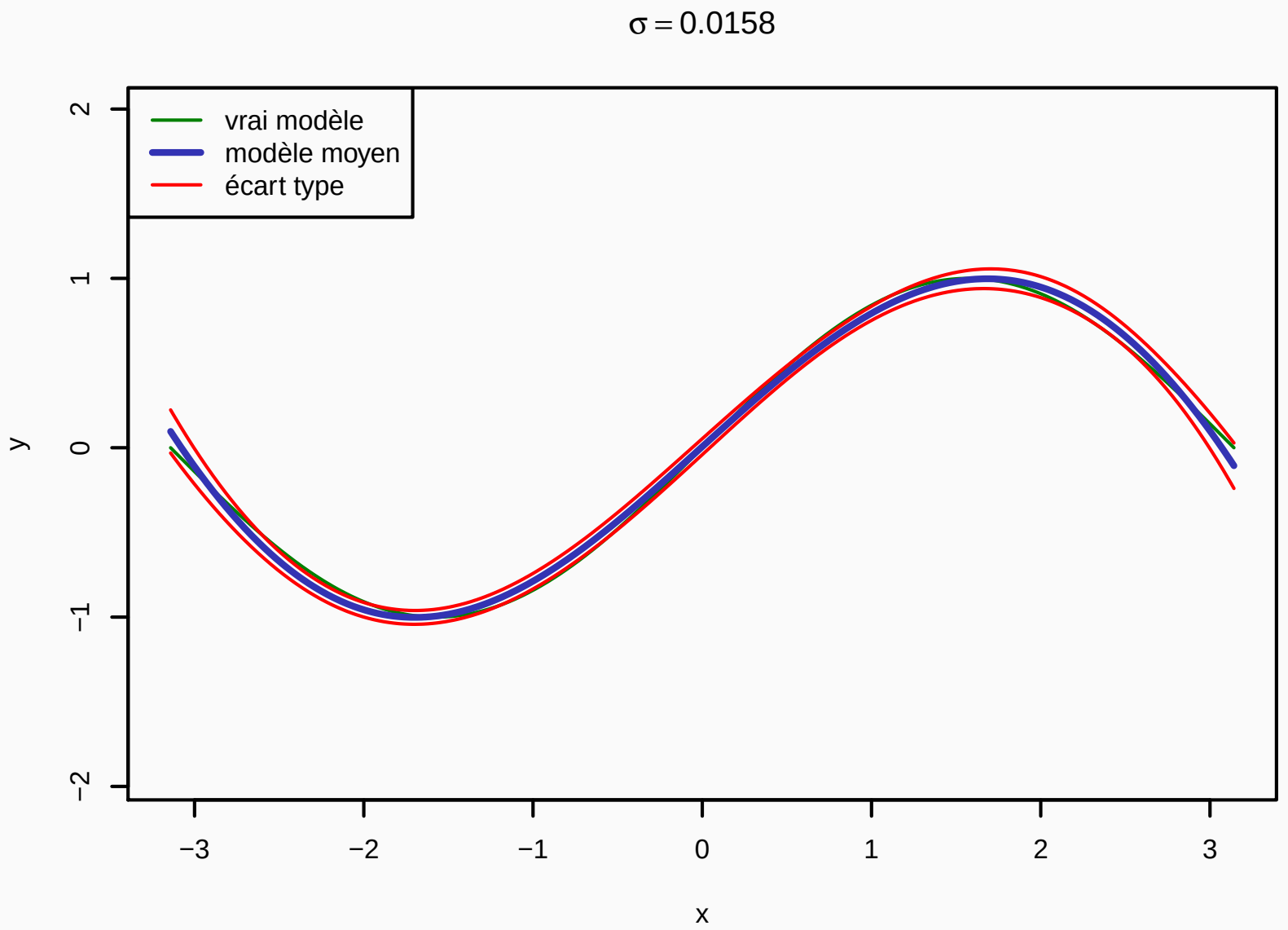
Résultats


σ = 0.00631
vrai modèle
modèle moyen
écart type
y
x
2
1
0
−1
−2
−3
−2
−1
0
1
2
3

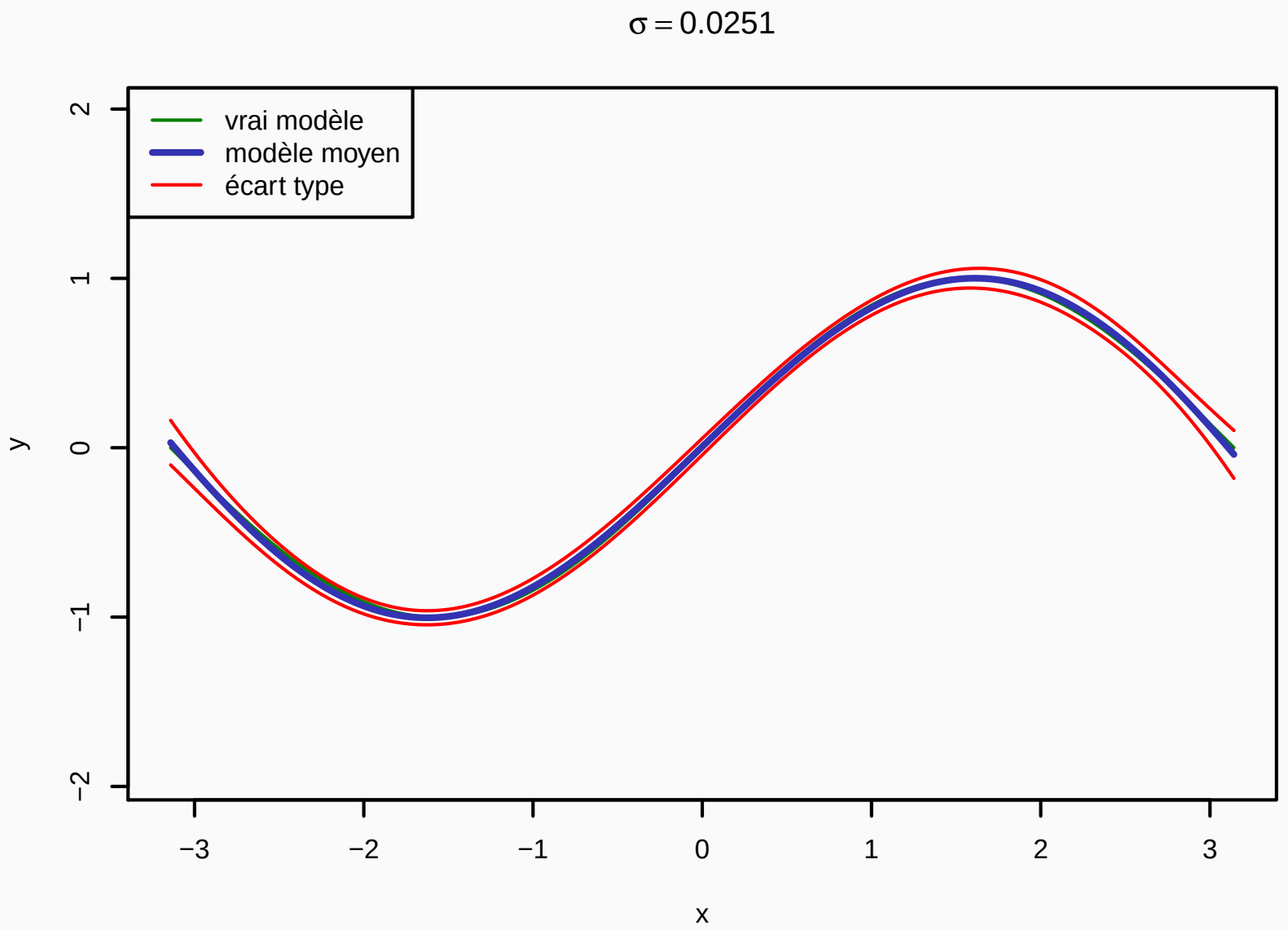
Résultats



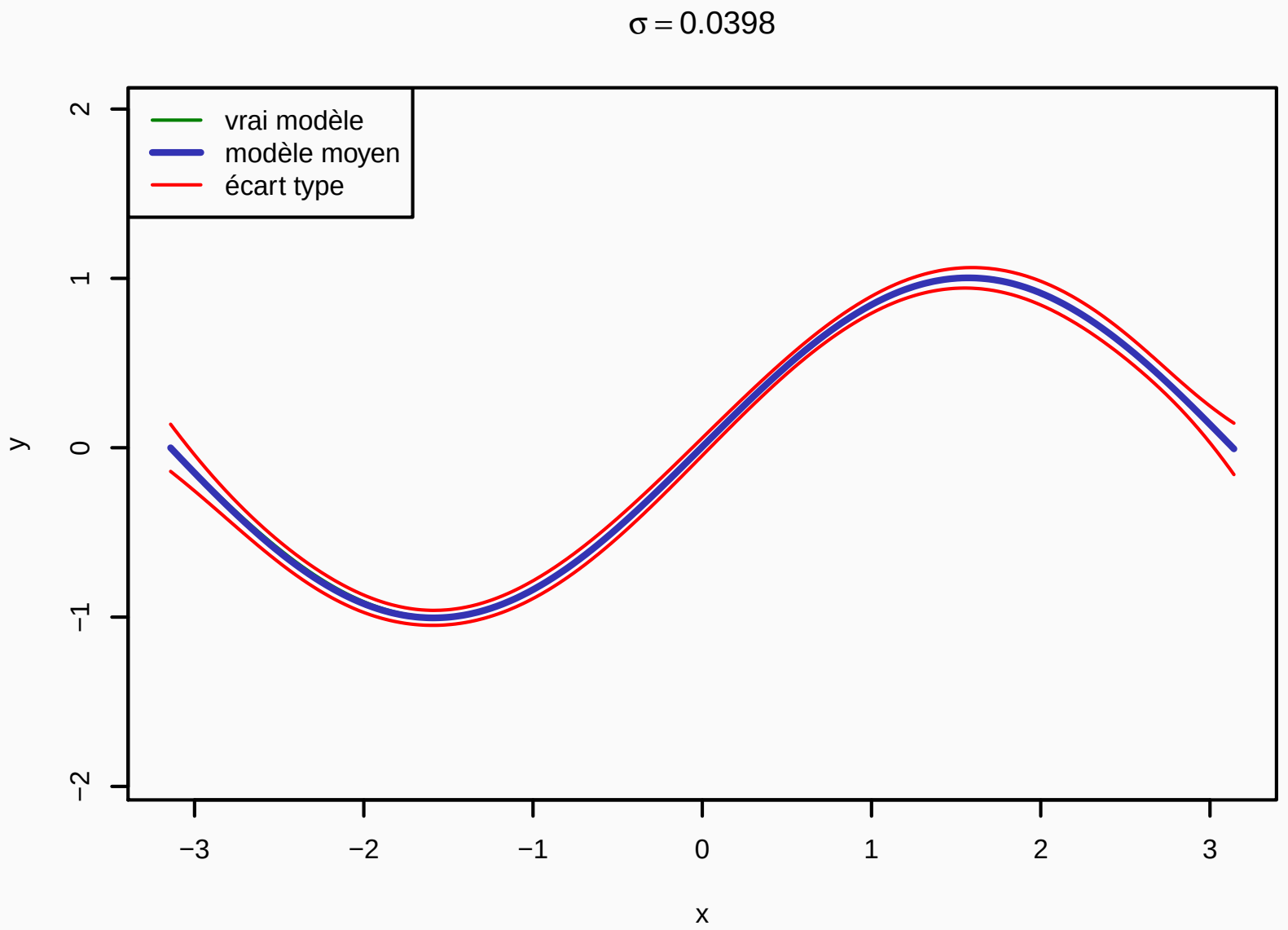
Résultats


σ = 0.0158
vrai modèle
modèle moyen
écart type
y
x
2
1
0
−1
−2
−3
−2
−1
0
1
2
3

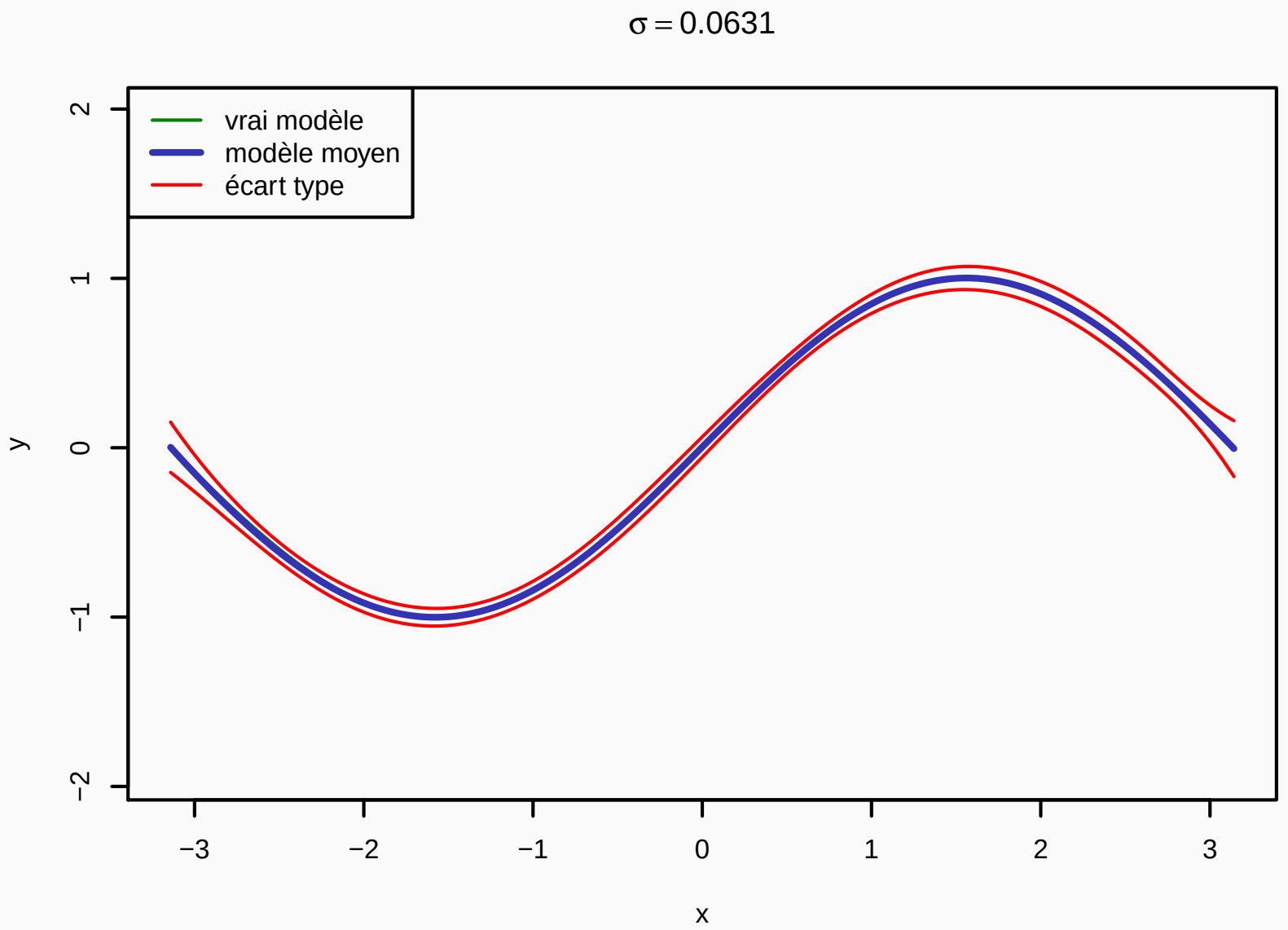
Résultats


σ = 0.0251
vrai modèle
modèle moyen
écart type
y
x
2
1
0
−1
−2
−3
−2
−1
0
1
2
3

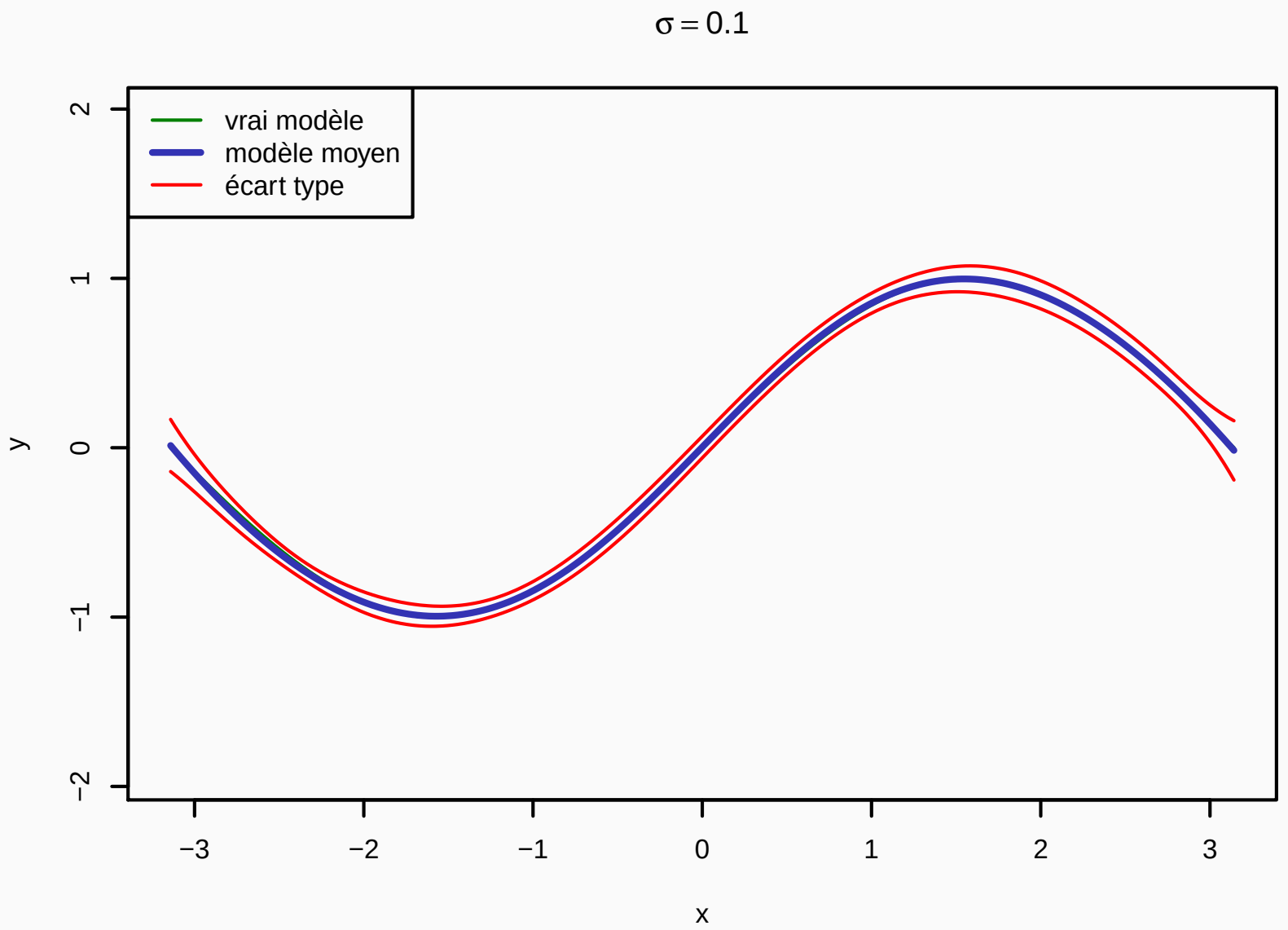
Résultats


σ = 0.0398
vrai modèle
modèle moyen
écart type
y
x
2
1
0
−1
−2
−3
−2
−1
0
1
2
3

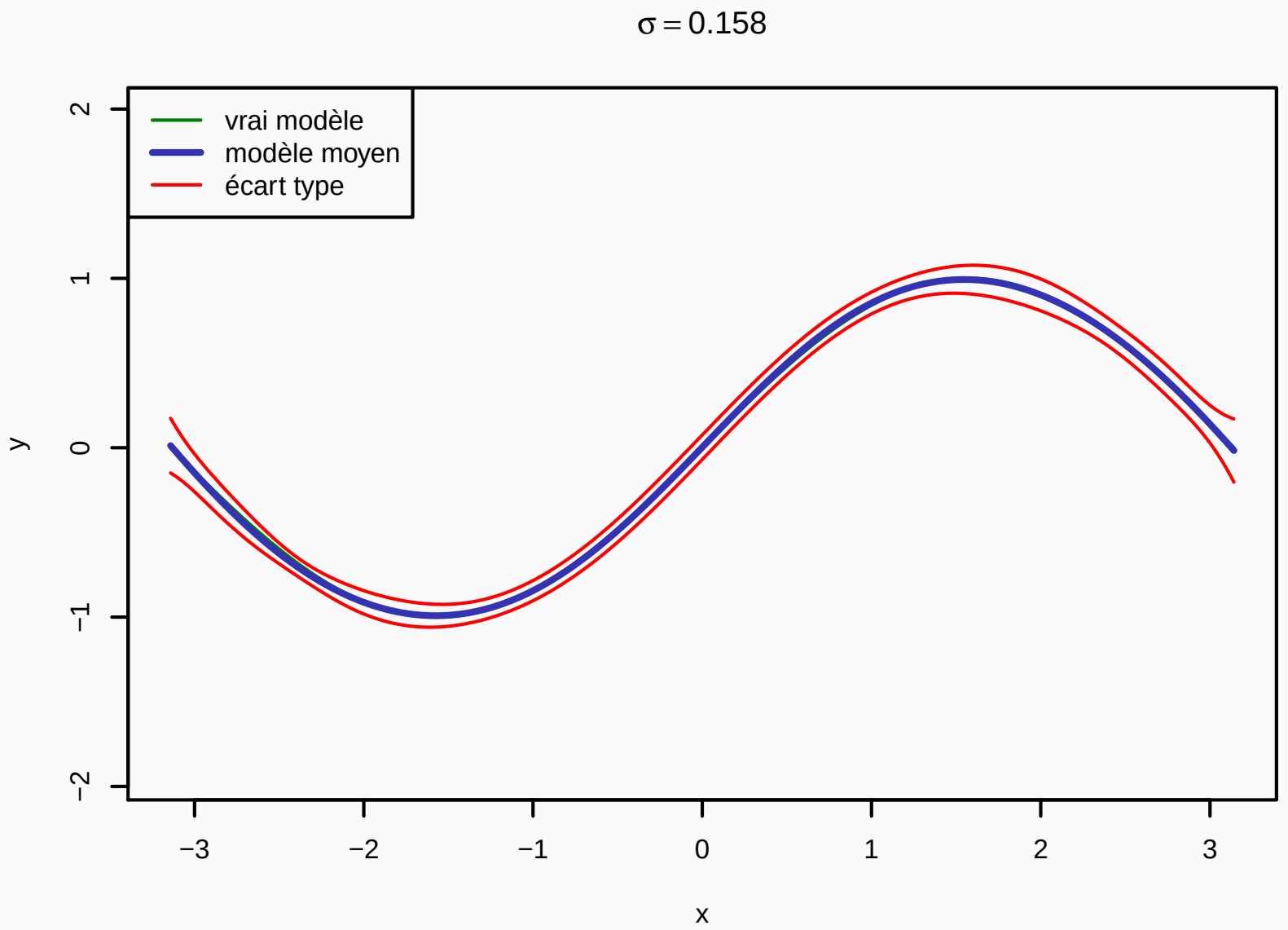
Résultats


σ = 0.0631
vrai modèle
modèle moyen
écart type
y
x
2
1
0
−1
−2
−3
−2
−1
0
1
2
3

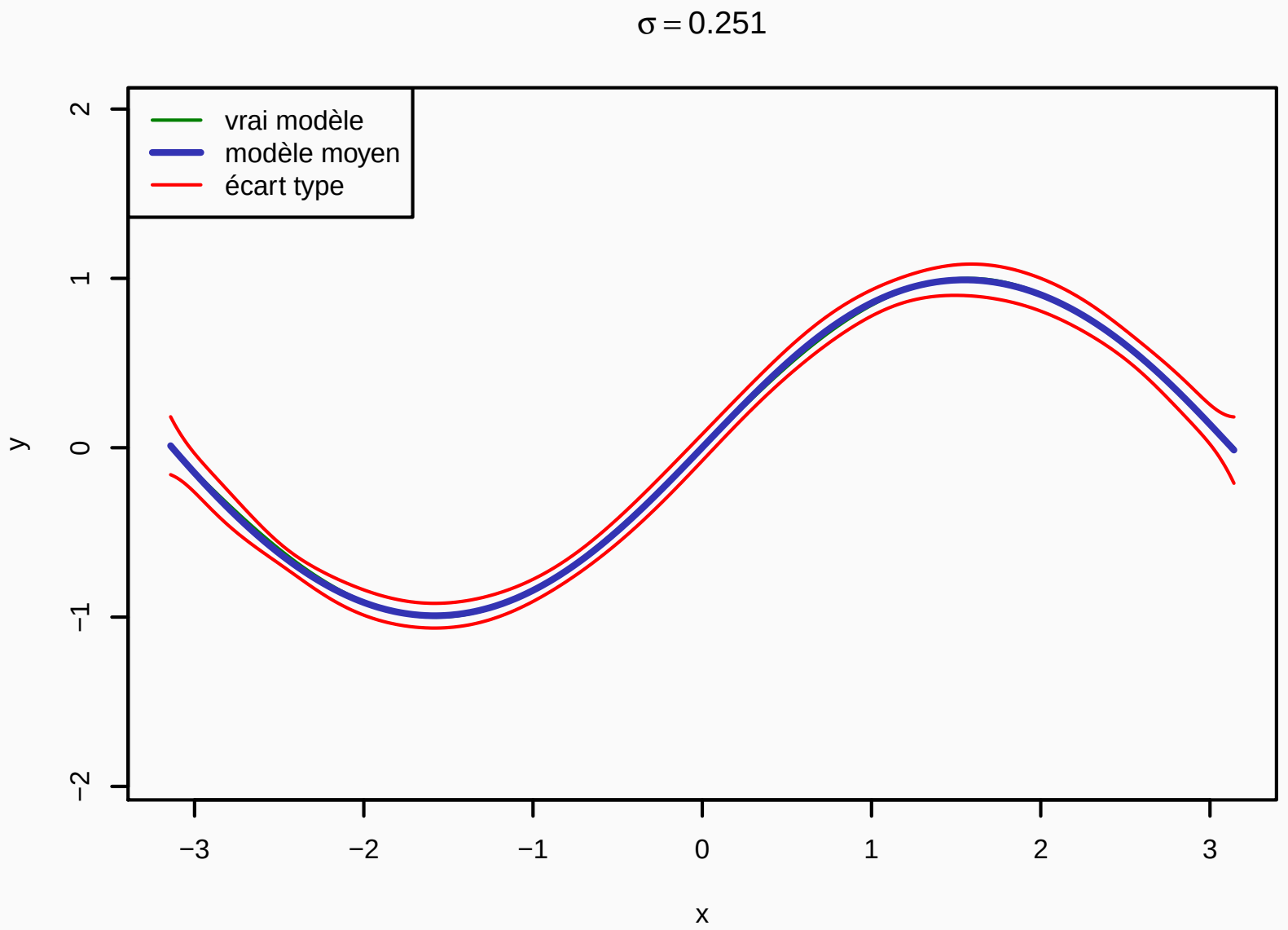
Résultats



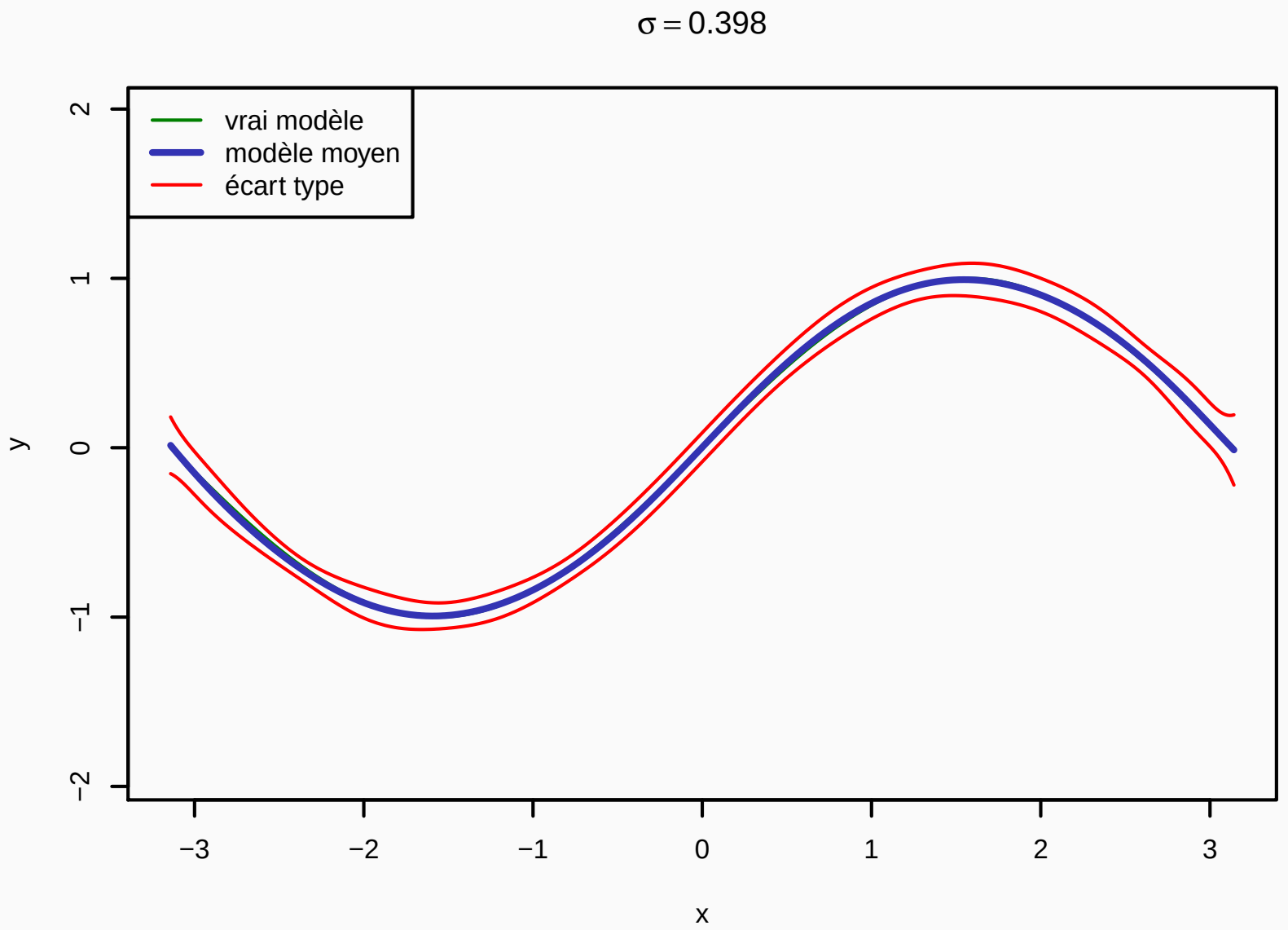
Résultats



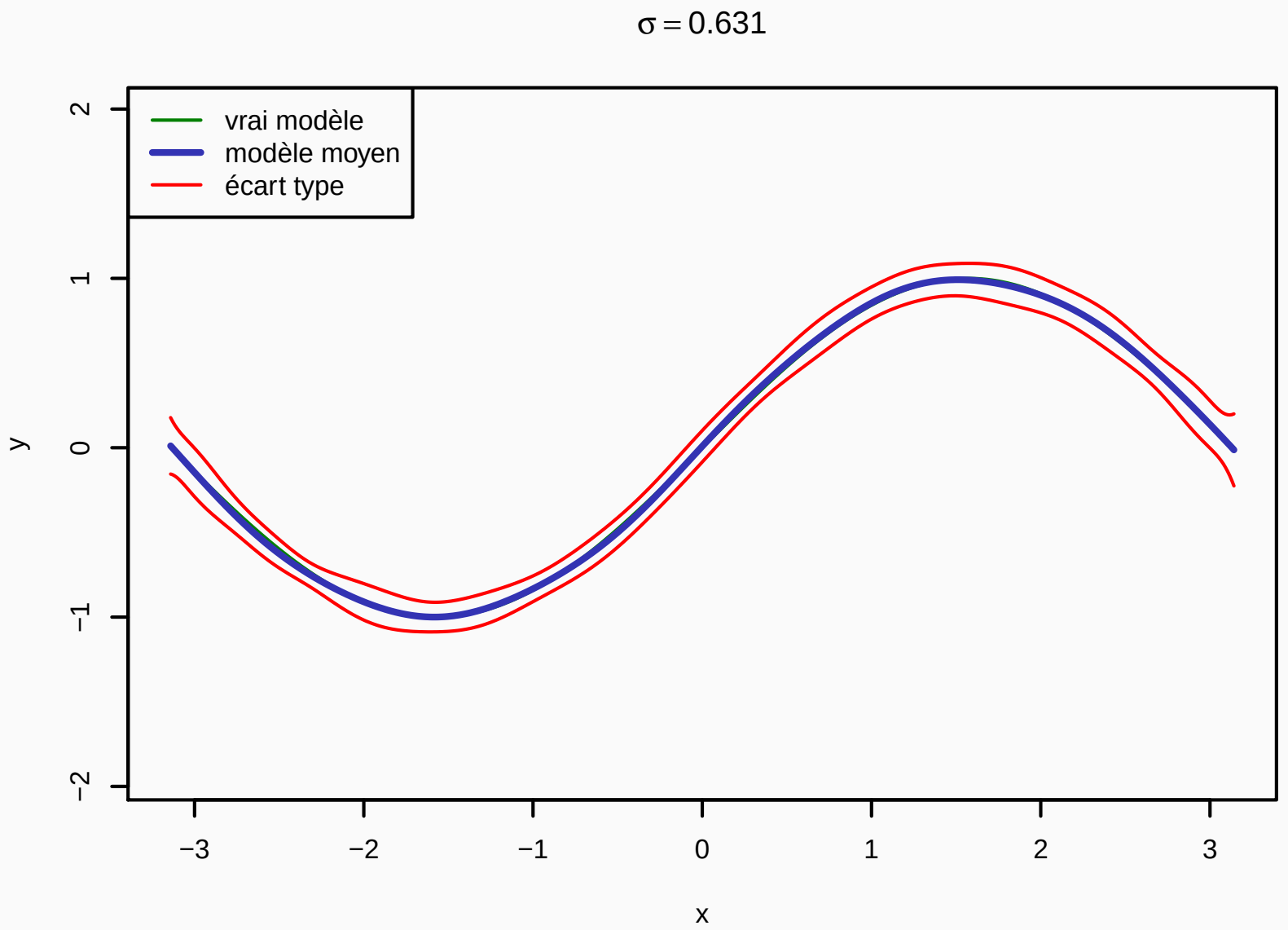
Résultats



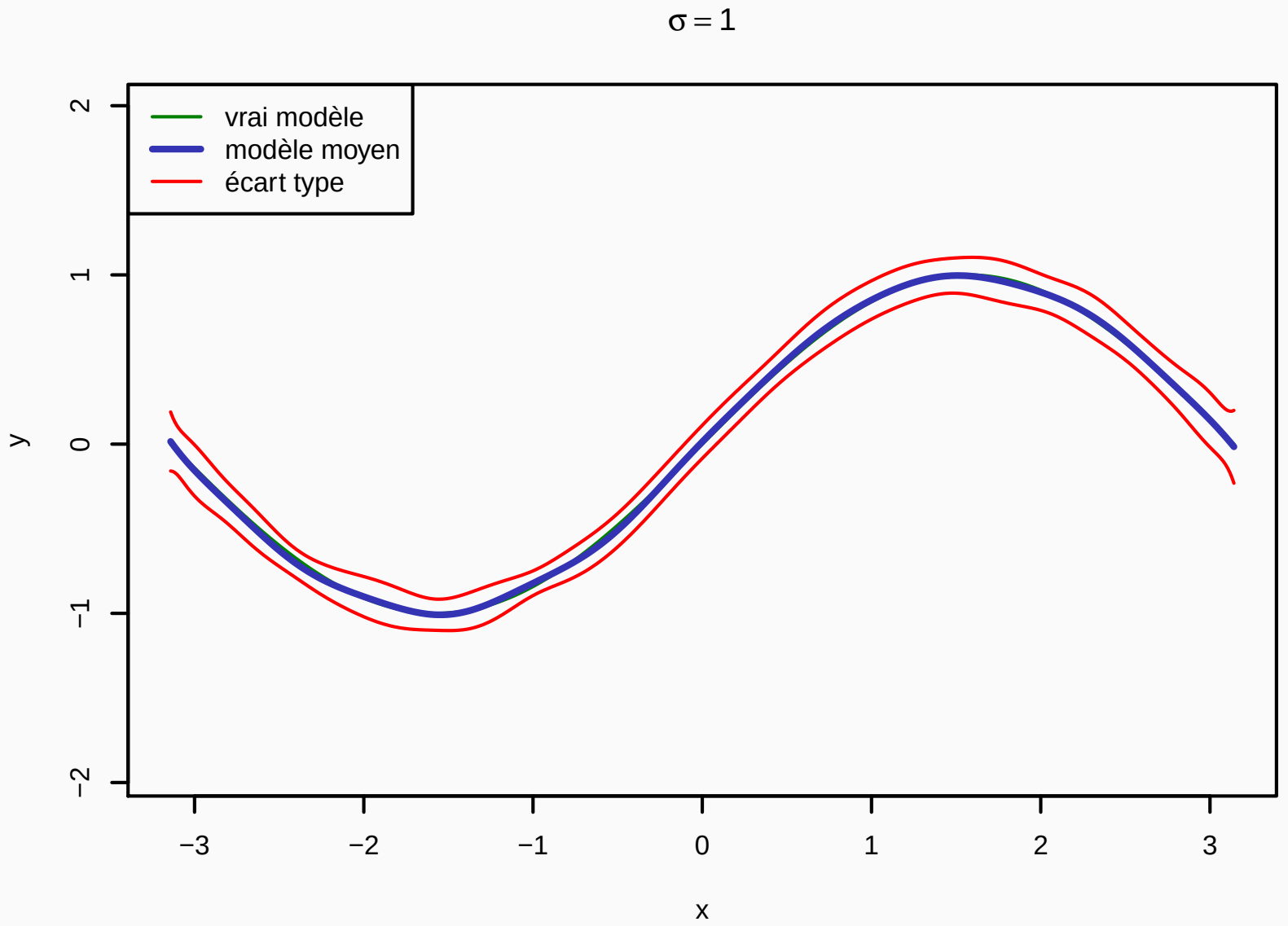
Résultats



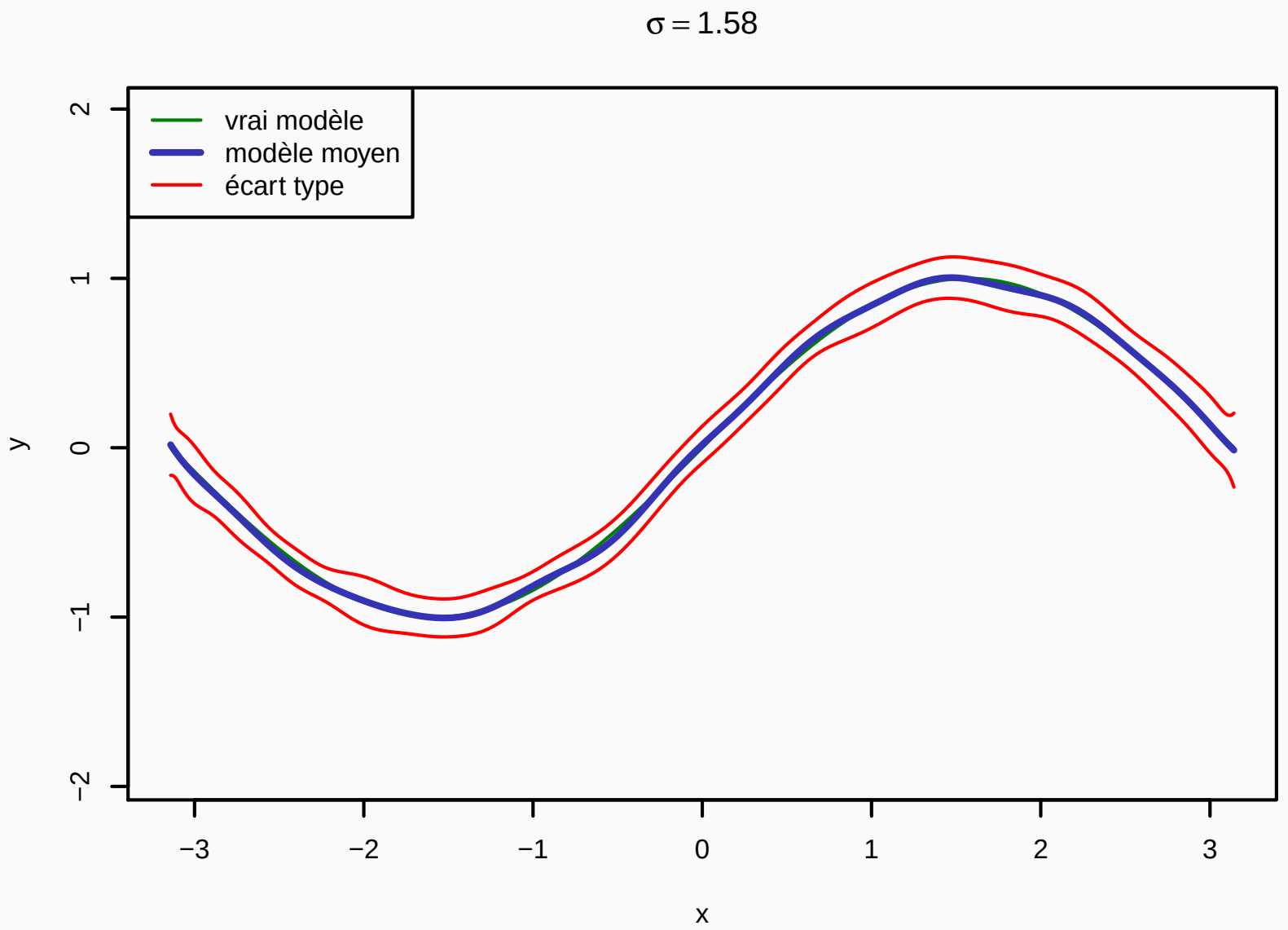
Résultats



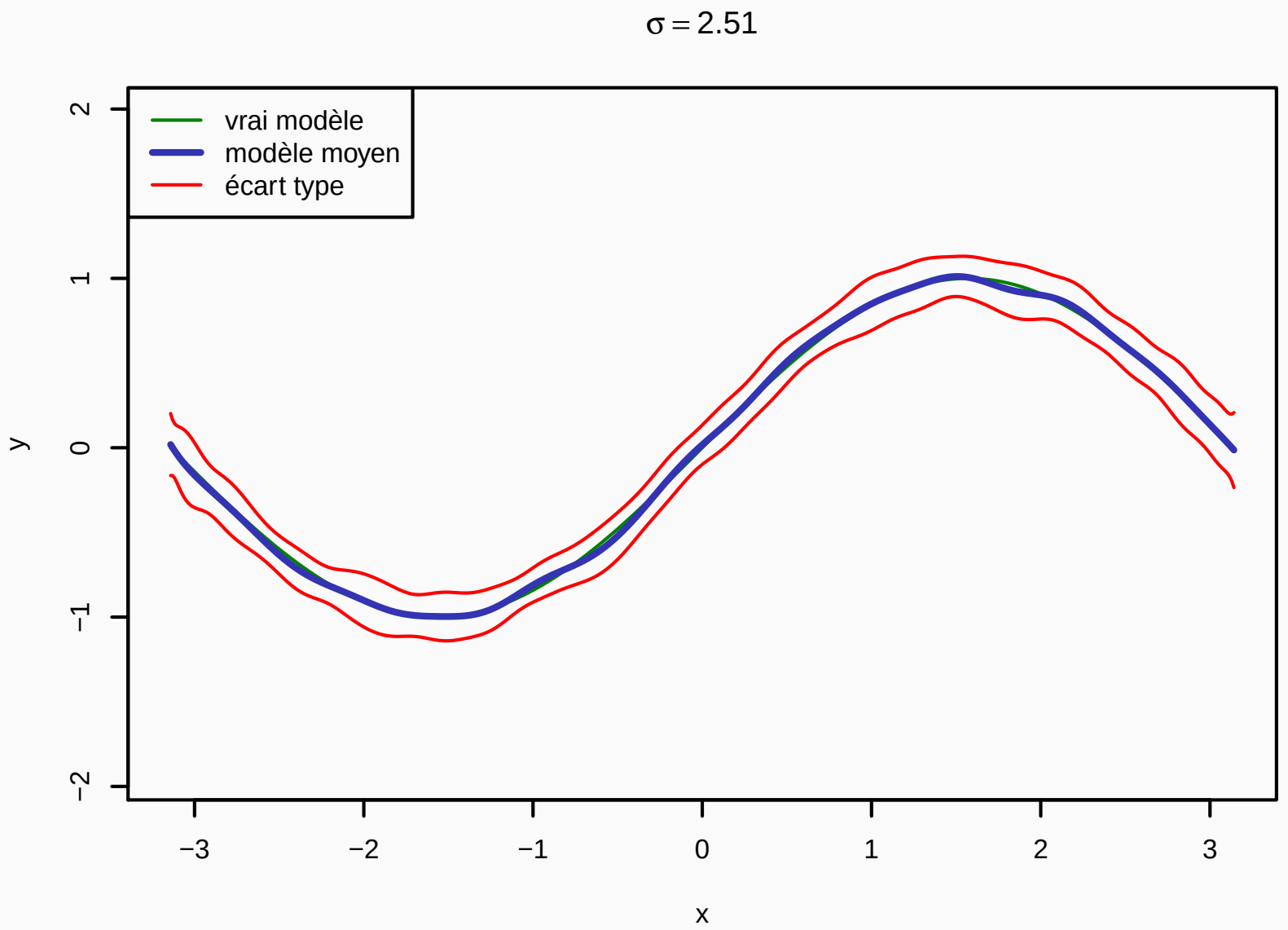
Résultats



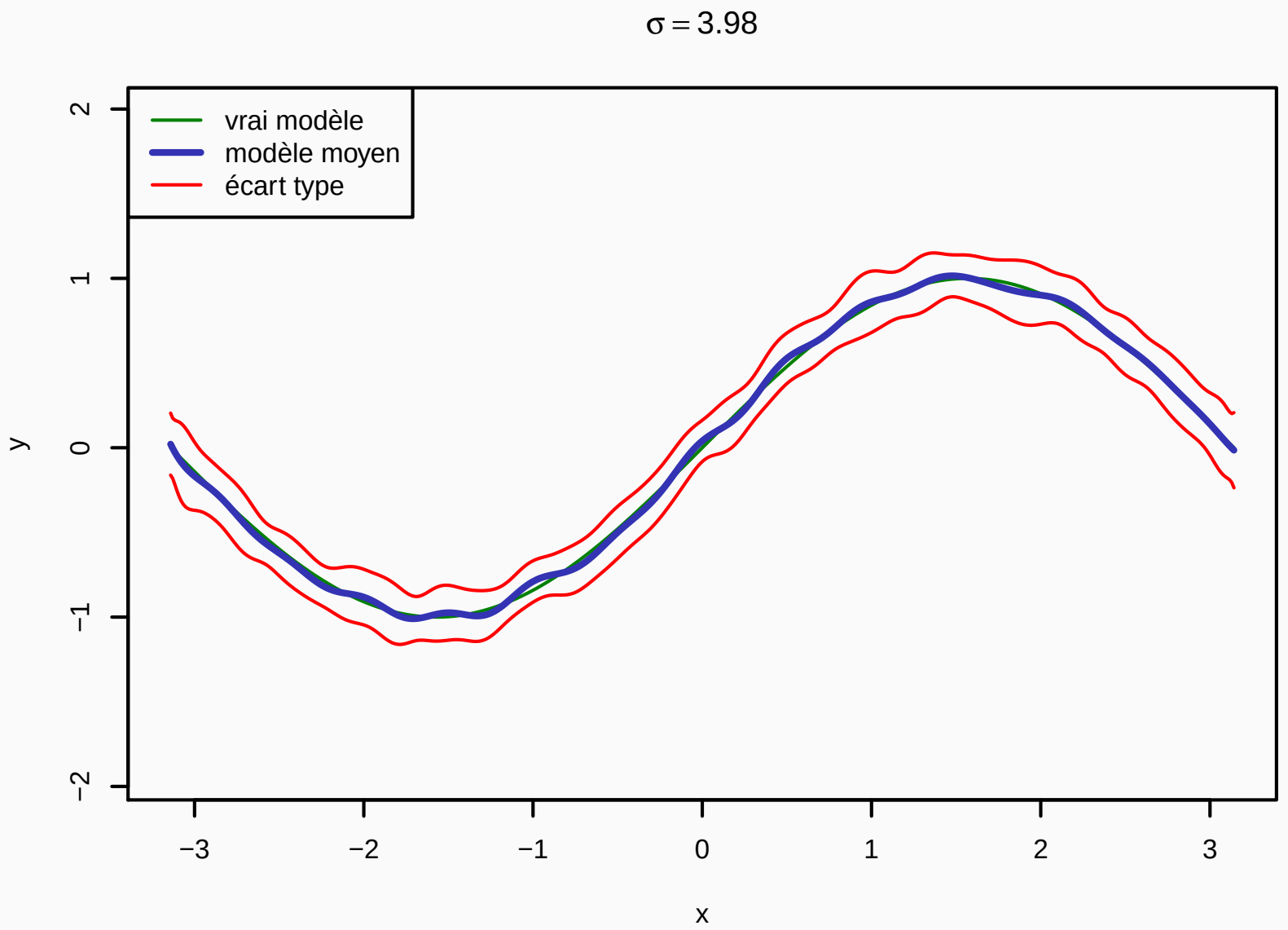
Résultats



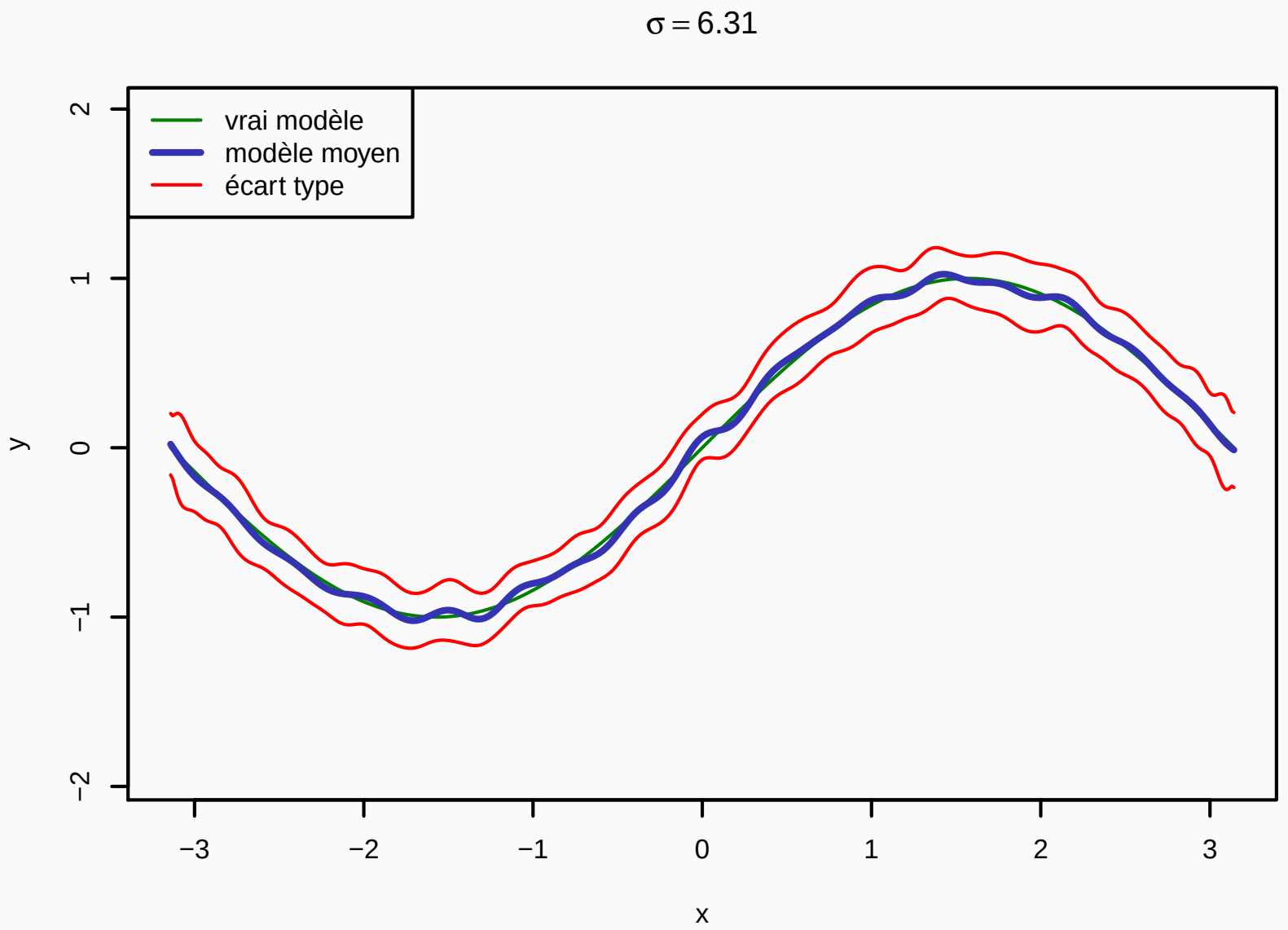
Résultats



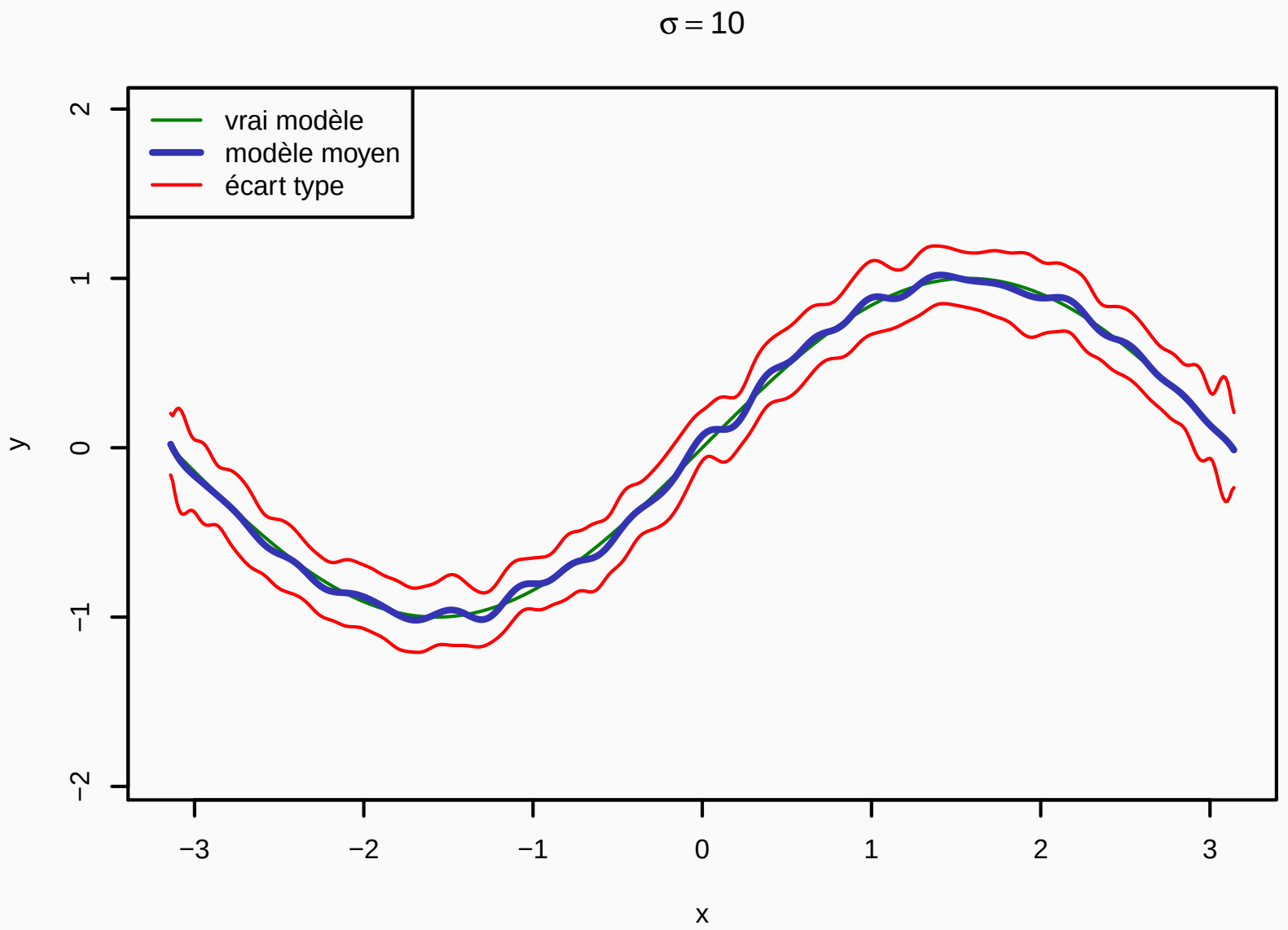
Résultats



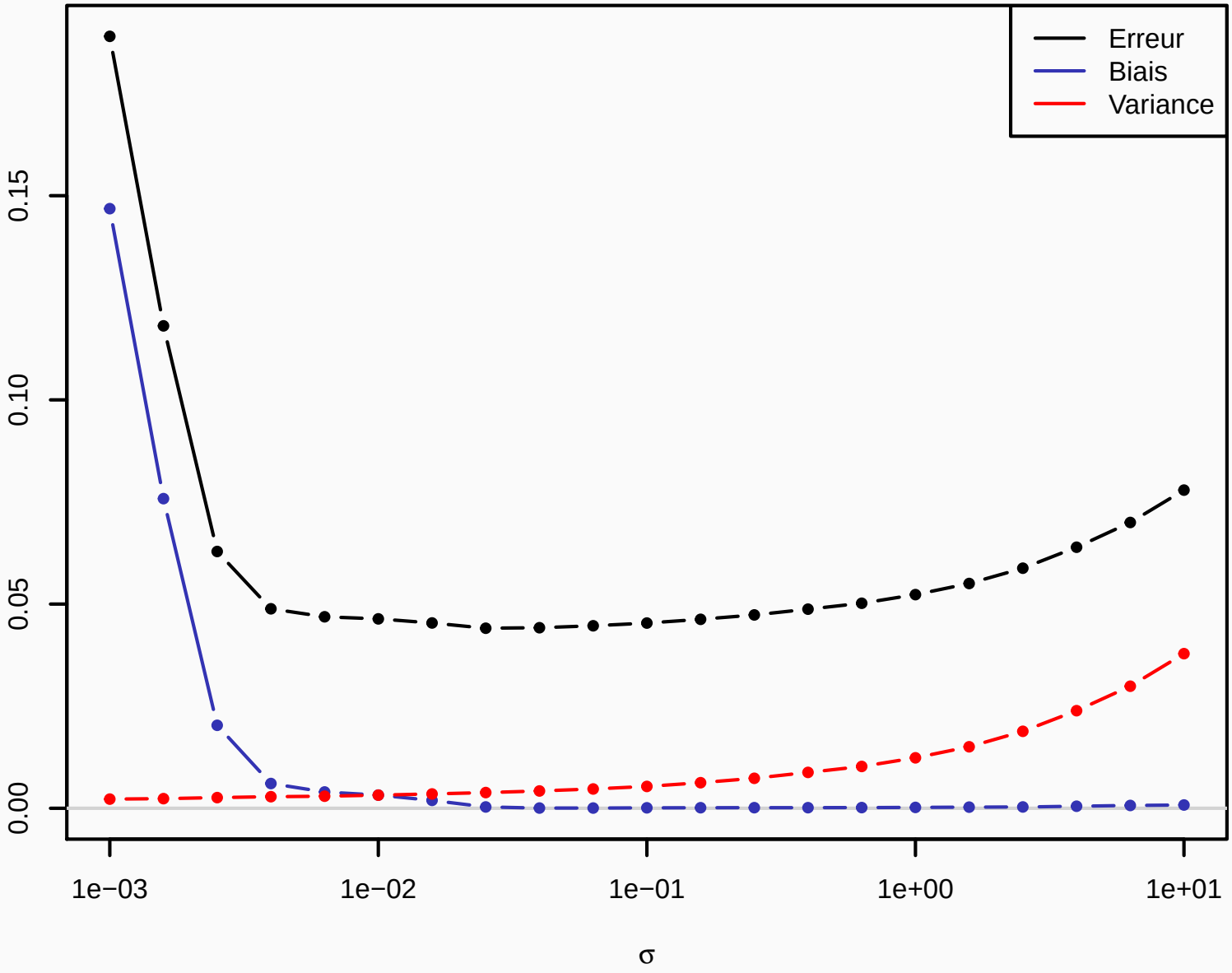
Résultats



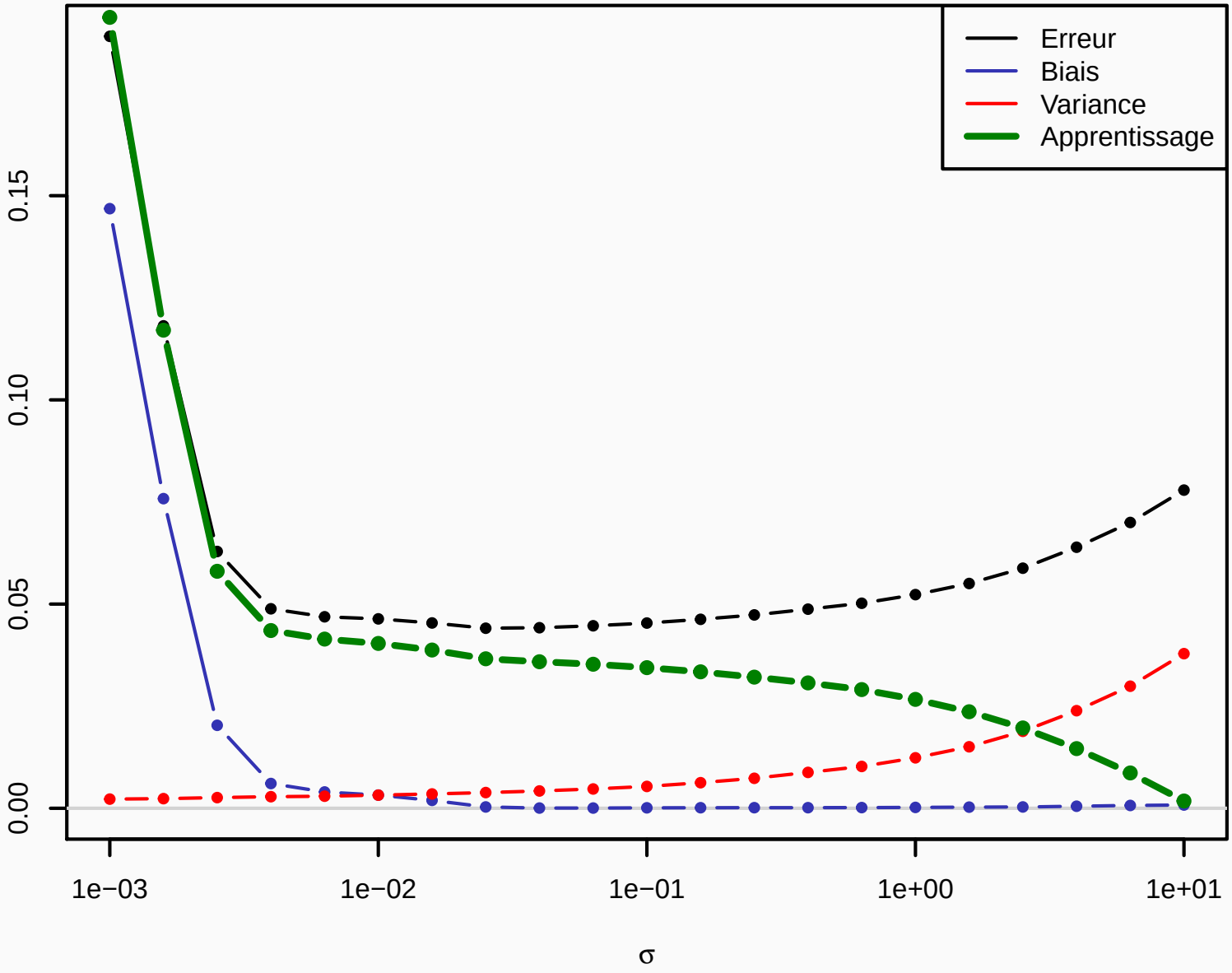
Résultats



En résumé


Erreur
Biais
Variance
0.00
0.05
0.10
0.15
1e−03
1e−02
1e−01
1e+00
1e+01
σ

En résumé


Erreur
Biais
Variance
Apprentissage
0.00
0.05
0.10
0.15
1e−03
1e−02
1e−01
1e+00
1e+01
σ

En résumé

Comportement général

- la variance du modèle est directement liée au sur-apprentissage
- quand le modèle colle aux données, sa variance explose
- un bon modèle est à l'équilibre : le prix à payer pour un faible biais est un niveau minimal de variance

Deux idées majeures

- pour étudier la variance d'un modèle, on peut simuler des ensembles d'apprentissage : techniques de ré-échantillonage
- si on sait simuler plusieurs ensembles d'apprentissage, on peut utiliser un modèle moyen : techniques de combinaison de modèles

Procédure générale

Objectifs

- choix de modèle (méta-paramètres)
- évaluation des performances du modèle choisi

Procédure

- deux niveaux de ré-échantillonage (ou de partition)
- un niveau externe d'évaluation
- un niveau interne de choix de modèle
- construction de plusieurs modèles

Exemple

- un ensemble de test et un ensemble d'apprentissage
- ré-échantillonage sur l'ensemble d'apprentissage
- évaluation directe sur l'ensemble de test

Dans la suite

Point principal

- l'estimation des performances par ré-échantillonage
- avantages/inconvénients des différentes méthodes
- bonnes pratiques

Point secondaire

- les procédures complètes
- cas particuliers intéressants

Statistiques et apprentissage

Statistiques

- un estimateur quelconque $\hat{\theta}$
- caractérisé par son biais et sa variance : espérances sous la distribution des données
- on estime ces quantités par ré-échantillonage

Apprentissage

- un modèle quelconque $\hat{g}$
- caractérisé par son risque $L(\hat{g})$: espérance sous la distribution d'une observation
- on estime le risque par ré-échantillonage

$$E_{\mathcal{D}\sim\mathcal{P}^N}\{\hat{\theta} - \theta\} \qquad E_{(X,Y)\sim\mathcal{P}}\{l(\hat{g}(X), Y)\}$$

Plan

Introduction

Compromis biais variance

Leave-one-out

Validation croisée

Bootstrap

Données structurées

Jackknife

Méthode statistique

- Estimateur $\hat{\theta} = f(X_1, \ldots, X_N)$
- Estimation partielle $\hat{\theta}_{-k} = f(X_1, \ldots, X_{k-1}, X_{k+1}, \ldots, X_N)$
- Estimateur Jackknife

$$\hat{\theta}^* = N\hat{\theta} - \frac{N-1}{N} \sum_{k=1}^{N} \hat{\theta}_{-k}$$

Justification

- Pseudo valeur $\hat{\theta}_k^* = N\hat{\theta} - (N-1)\hat{\theta}_{-k}$
- $\hat{\theta}^*$ est la moyenne des pseudo valeurs
- vient du cas de l'estimateur de l'espérance par la moyenne empirique

$$x_k = N \left(\frac{1}{N} \sum_{j=1}^{N} x_j \right) - (N-1) \left(\frac{1}{N-1} \sum_{j=1,\, j \neq k}^{N} x_j \right)$$

Estimation du biais

- estimation jackknife du biais

$$B_{jack} = (N-1)\left(\frac{1}{N}\sum_{k=1}^{N}\hat{\theta}_{-k} - \hat{\theta}\right)$$

- le facteur $(N-1)$ permet une correction exacte à l'ordre 1 (bias en $\frac{1}{N}$)
- l'estimateur $\hat{\theta}^*$ est débiaisé

$$\begin{aligned}\hat{\theta}^* &= N\hat{\theta} - \frac{N-1}{N}\sum_{k=1}^{N}\hat{\theta}_{-k} \\ &= \hat{\theta} - B_{jack}\end{aligned}$$

Leave-one-out (LOO)

Utilisation en apprentissage

- on ne conserve que l'idée de base : estimation avec une observation en moins
- modèle $g = A(Z_1, \ldots, Z_n)$ où A est un algorithme d'apprentissage (avec $Z_i = (X_i, Y_i)$)
- modèle *loo* $g_{-k} = A(Z_1, \ldots, Z_{k-1}, Z_{k+1}, \ldots, Z_n)$
- estimation de la performance

$$\hat{L}_{loo}(g) = \frac{1}{N} \sum_{k=1}^{N} l(g_{-k}(X_k), Y_k)$$

Procédure très différente

- chaque terme $l(g_{-k}(X_k), Y_k)$ utilise toutes les données
- on n'estime donc pas le biais d'un estimateur

En pratique

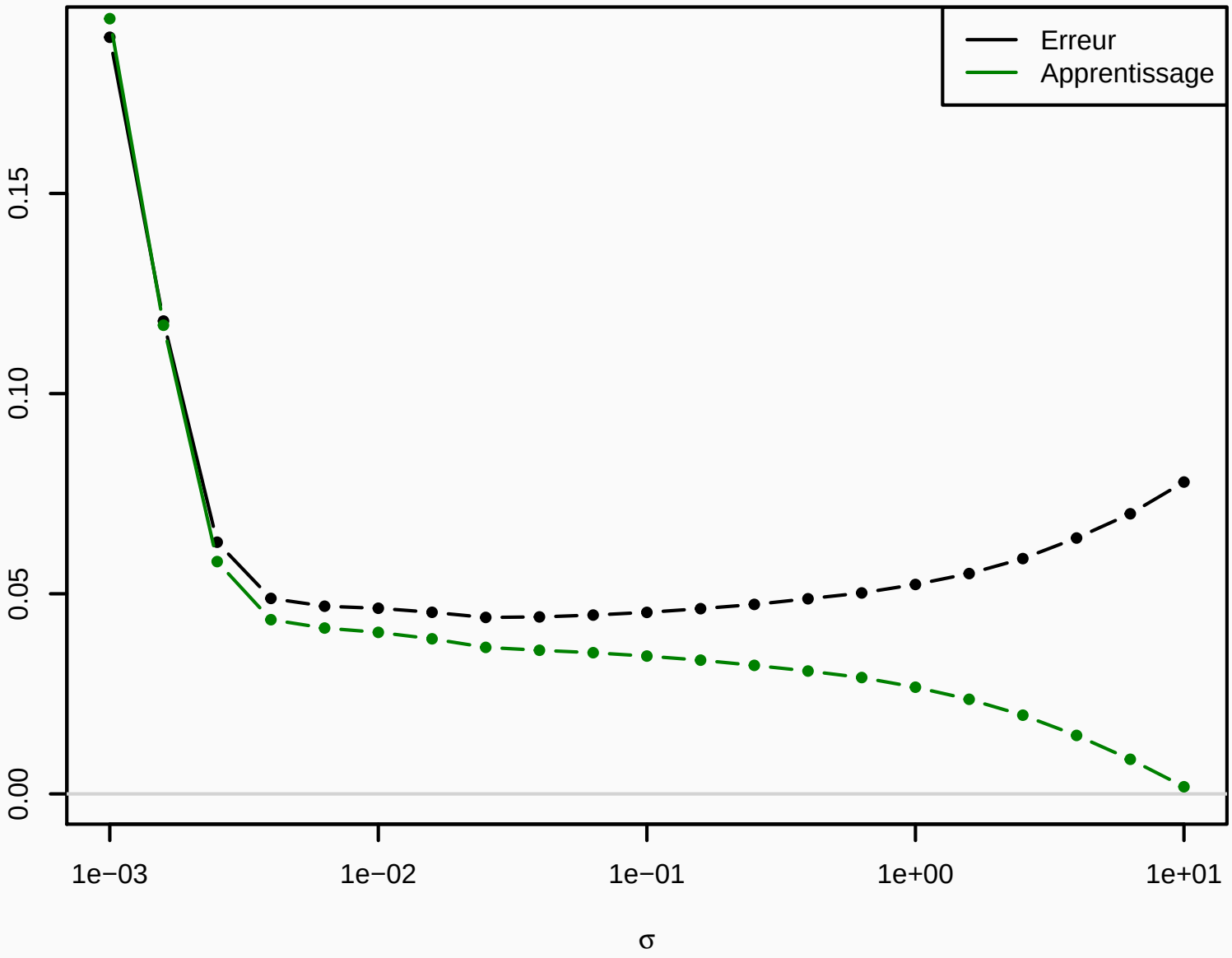
Avantages

- $\hat{L}_{loo}(g)$ a en général un biais faible
- aucun phénomène aléatoire : chaque observation est utilisée de la même façon ($N-1$ en apprentissage, 1 fois en évaluation)
- parallélisation évidente

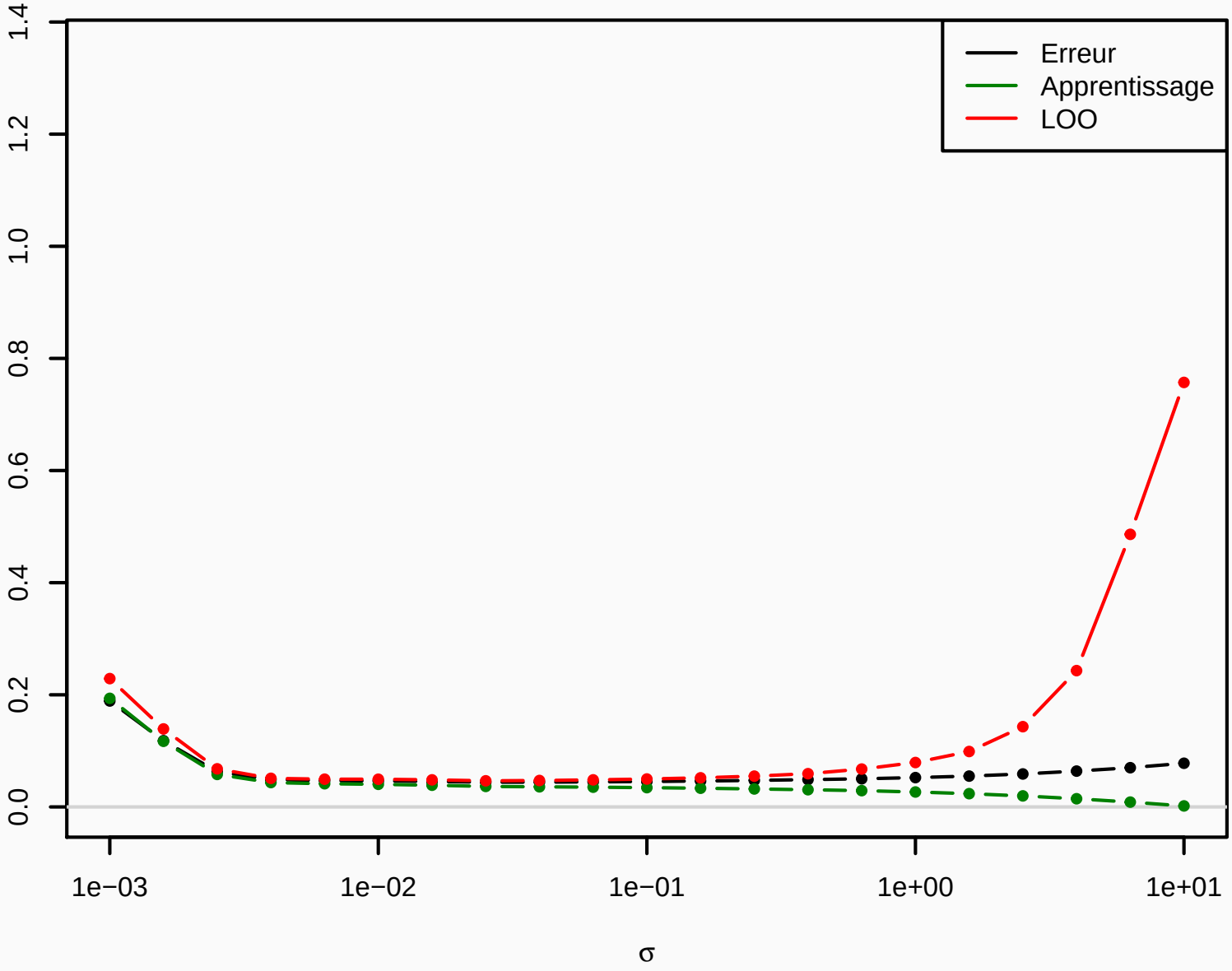
Inconvénients

- $\hat{L}_{loo}(g)$ a en général une variance élevée
- temps de calcul très élevé : N modèles à ajuster sur presque toutes les données

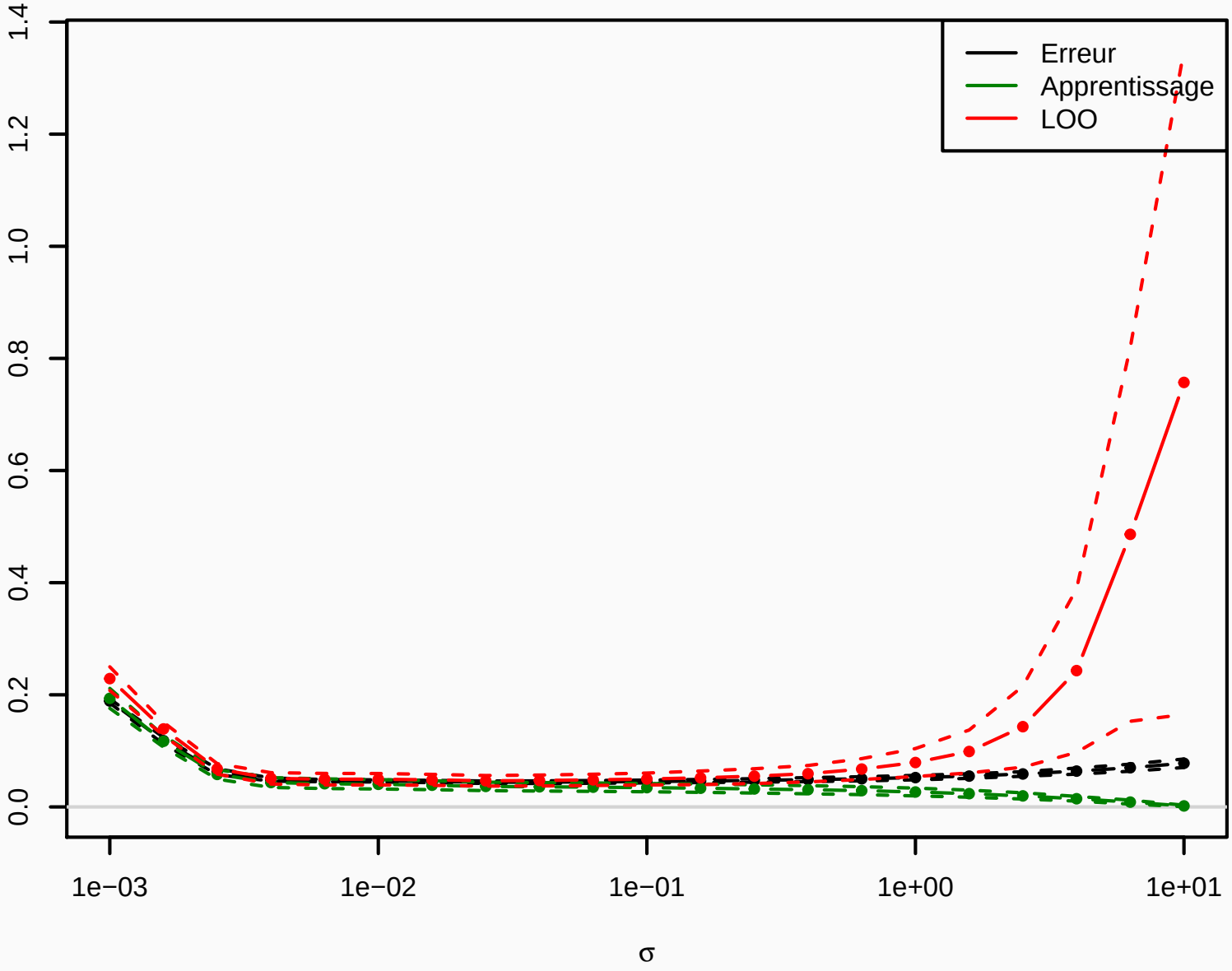
Retour sur l'exemple



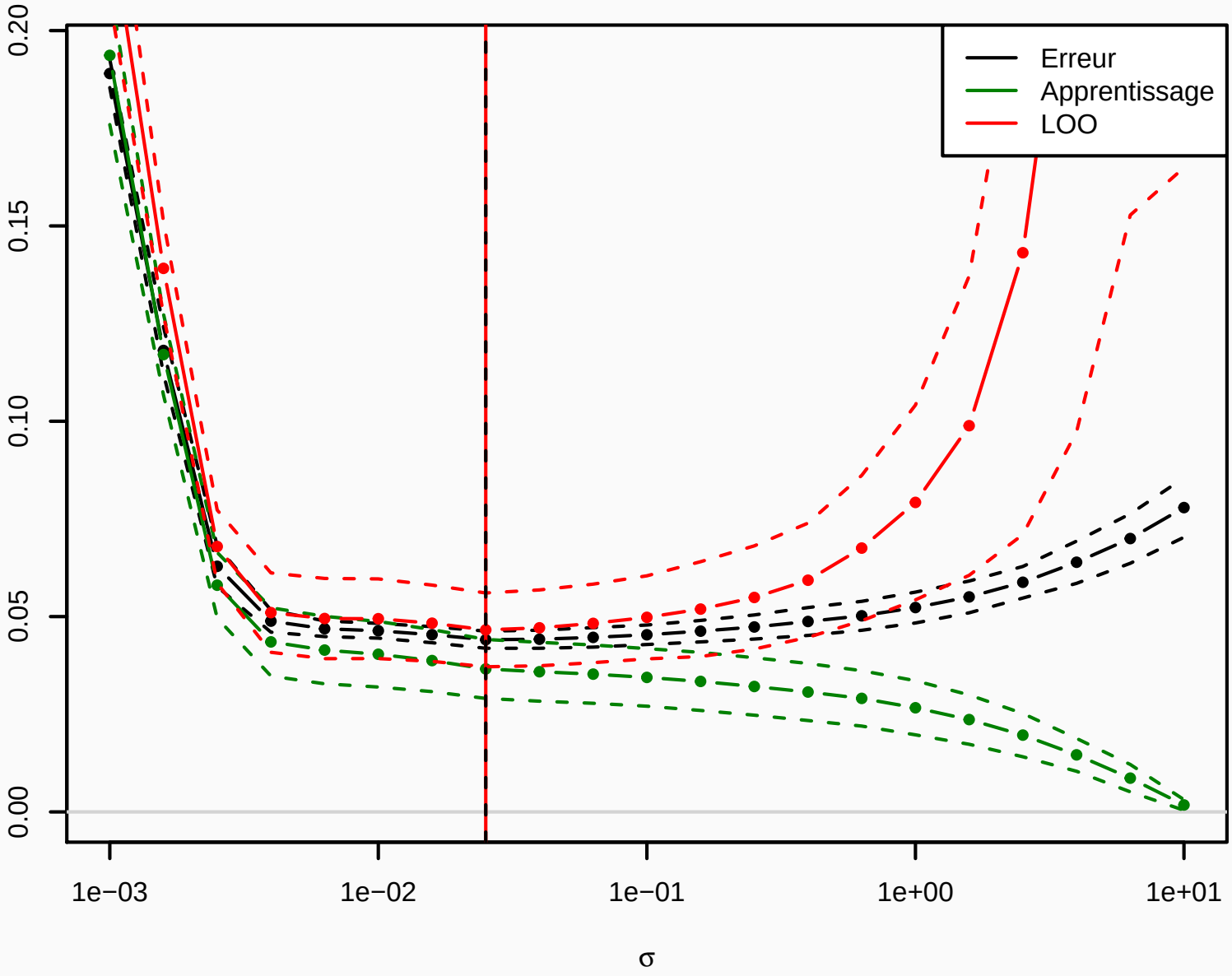
Retour sur l'exemple



Retour sur l'exemple



Retour sur l'exemple



Version efficace

Coût algorithmique

- le calcul de $\hat{L}_{loo}(g)$ demande l'estimation de N modèles
- le leave-one-out est inutilisable en général avec des données volumineuses et/ou des modèles complexes
- sauf pour certains modèles linéaires

Modèles linéaires

- $\mathbf{Y}$: vecteur des Y_i
- $\hat{\mathbf{Y}}$: ensemble des prévisions du modèle
- dans certains modèles « linéaires », on $\hat{\mathbf{Y}} = \mathbf{SY}$ pour une matrice $\mathbf{S}$ de lissage
- dans ce cas, on a souvent

$$\hat{L}_{loo}(g) = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{y_i - \mathbf{SY}_i}{1 - S_{ii}} \right)^2$$

Régression linéaire

Rappels

- $\mathbf{X}$: matrice des X_i en ligne (avec un 1 en dernière position)
- $\mathbf{Y}$: vecteur des Y_i
- régression linéaire : $\mathbf{Y} \simeq \mathbf{X}\beta$
- β optimal au sens des moindres carrés : $\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T Y$

Leave-one-out

- on a alors $\hat{\mathbf{Y}} = \underbrace{\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T}_{\mathbf{S}} Y$
- et dans cette situation

$$\hat{L}_{loo}(g) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - \mathbf{S}\mathbf{Y}_i}{1 - S_{ii}}\right)^2$$

Conséquences

Coût

- le surcoût algorithme du calcul de $\hat{L}_{loo}(g)$ est essentiellement négligeable
- l'estimation leave-one-out est donc « gratuite » pour ces modèles

Applications

- pour le modèle linéaire : sélection de variables (mais il existe de nombreuses autres solutions)
- pour les modèles plus généraux :
  - régression ridge : sélection du paramètre λ de régularisation
  - *kernel ridge régression* : idem + sélection du noyau

Kernel ridge regression

Noyau

Un noyau K de $\mathcal{X}^2$ dans $\mathbb{R}$ est une fonction qui vérifie

- $K(x,y) = K(y,x)$
- $\sum_i \sum_j \lambda_i \lambda_j K(x_i,x_j) \geq 0$

Kernel ridge regression

- modèle de la forme $g(x) = \sum_{j=1}^{N} \alpha_j K(x_j,x)$
- $\hat{\alpha}$ optimal solution de

$$\min_{\alpha \in \mathbb{R}^N} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{N} \alpha_j K(x_j,x_i) \right)^2 + \lambda \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j K(x_i,x_j)$$

- extension non linéaire de la régression ridge

Kernel ridge regression

Leave-one-out

- on montre que $\hat{\alpha} = (K + \lambda I_N)^{-1} Y$
- et que

$$\hat{\mathbf{Y}} = \underbrace{K(K + \lambda I_N)^{-1}}_{\mathbf{S}} Y$$

- la formule du leave-one-out s'applique

Astuce algorithmique

1. calcul de la SVD de $\mathbf{K} = (K(x_i, x_j))_{i,j}$, $\mathbf{K} = \mathbf{U}\mathbf{D}\mathbf{V}^T$
2. calcul de la matrice diagonale $\mathbf{W}(\lambda)$ de termes diagonaux $W(\lambda)_{ii} = \frac{1}{D_{ii} + \lambda}$
3. calcul de $\mathbf{S}(\lambda) = \mathbf{V}\mathbf{W}(\lambda)\mathbf{U}^T$

TP LOO

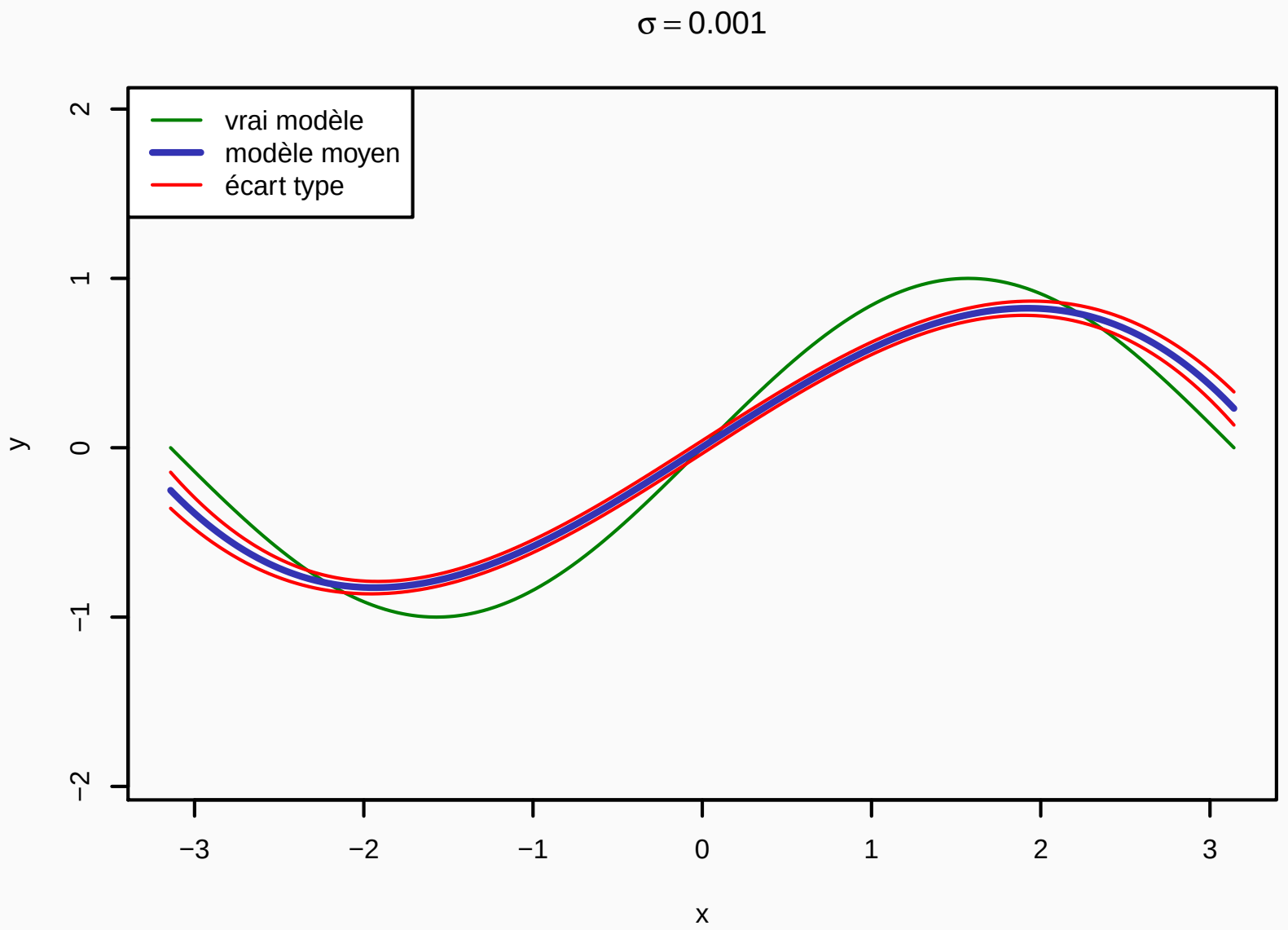
Objectifs

- prise en main du leave-one-out
- expérimentation biais versus variance

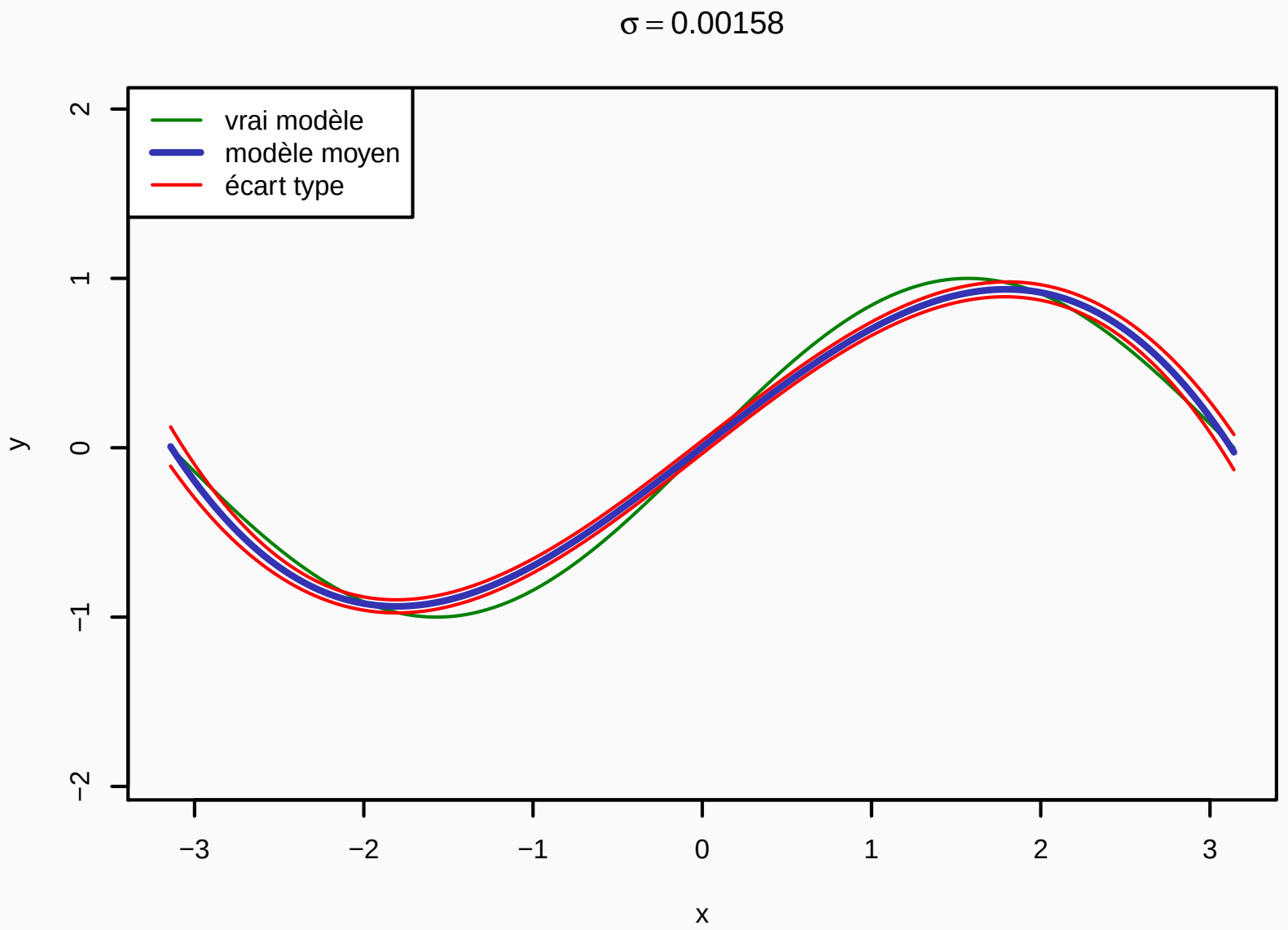
Mise en œuvre

1. engendrer des données artificielles (plusieurs jeux)
2. programmer la kernel ridge regression avec sélection automatique de λ par leave-one-out
3. sélection des paramètres du noyau de la même façon
4. courbes erreur/biais/variance/etc.

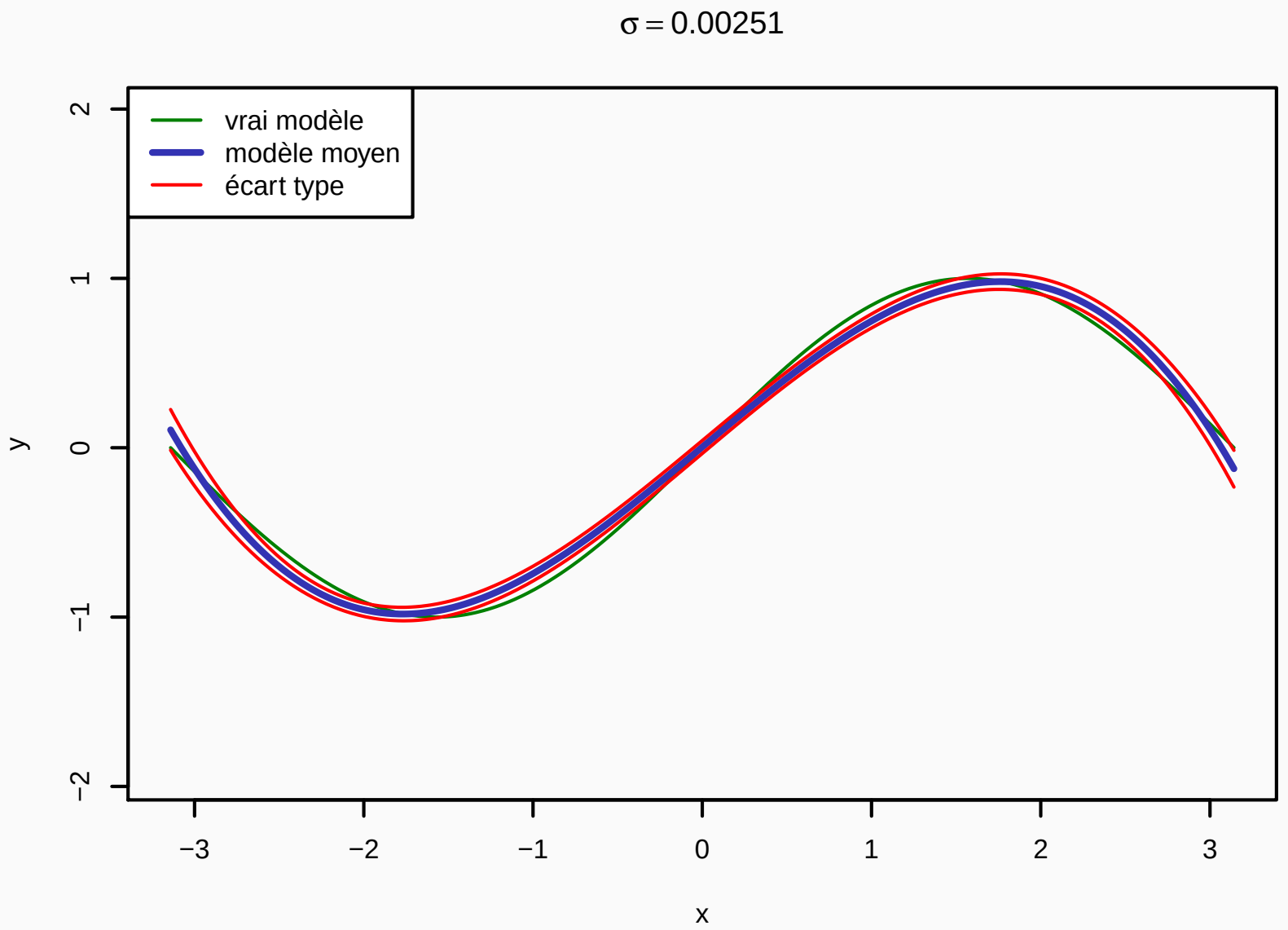
Choix automatique de λ



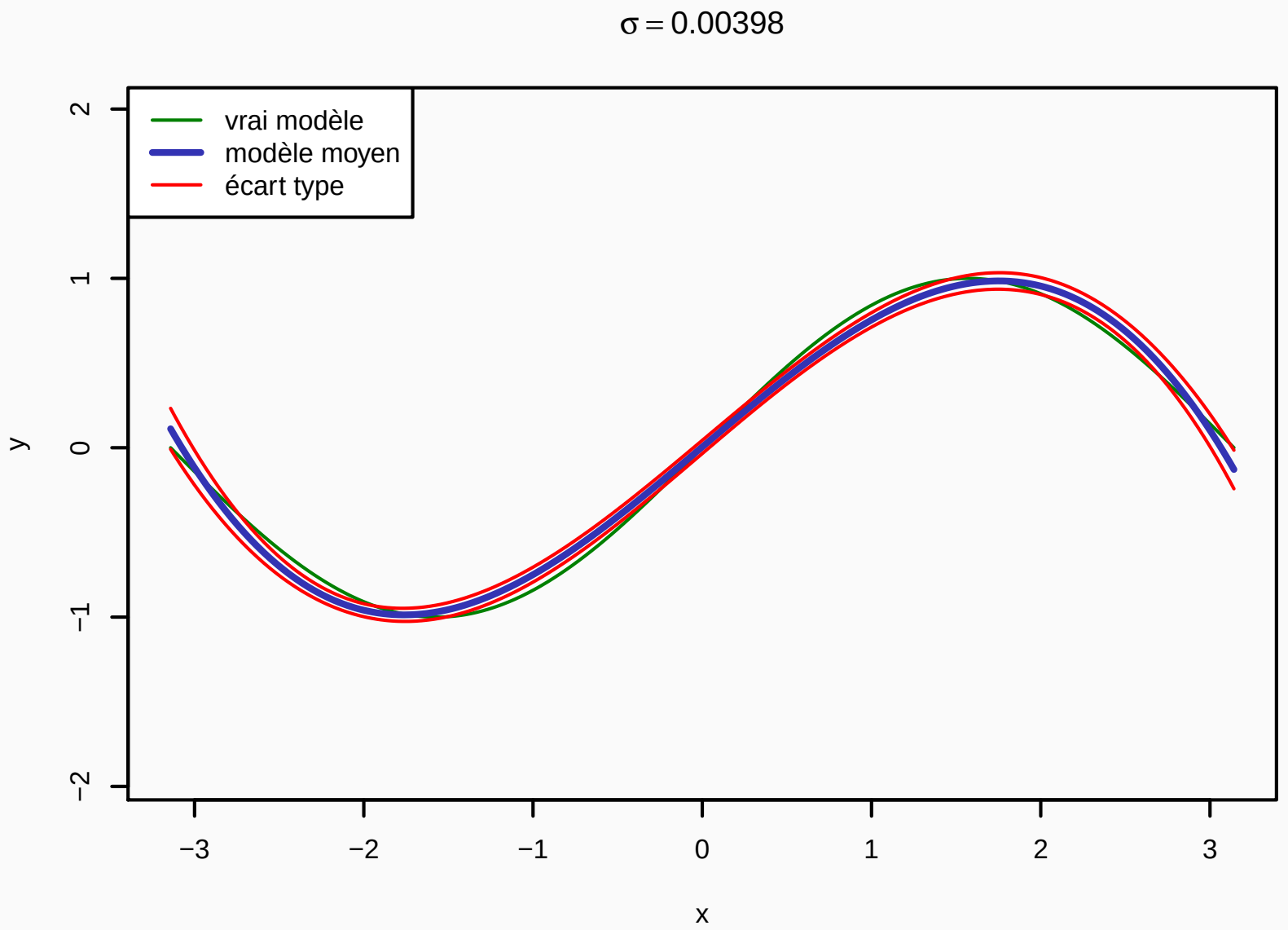
Choix automatique de λ



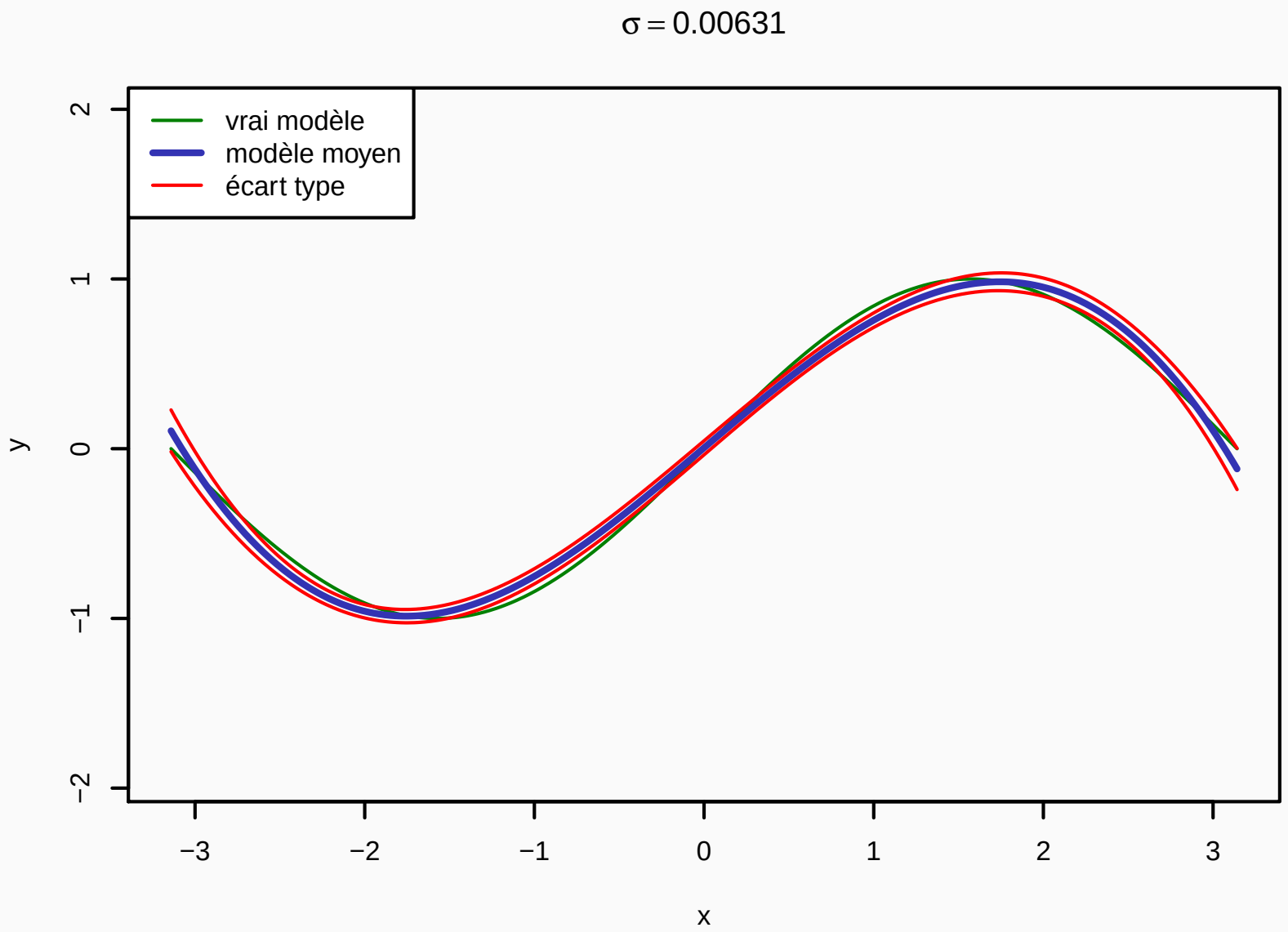
Choix automatique de λ



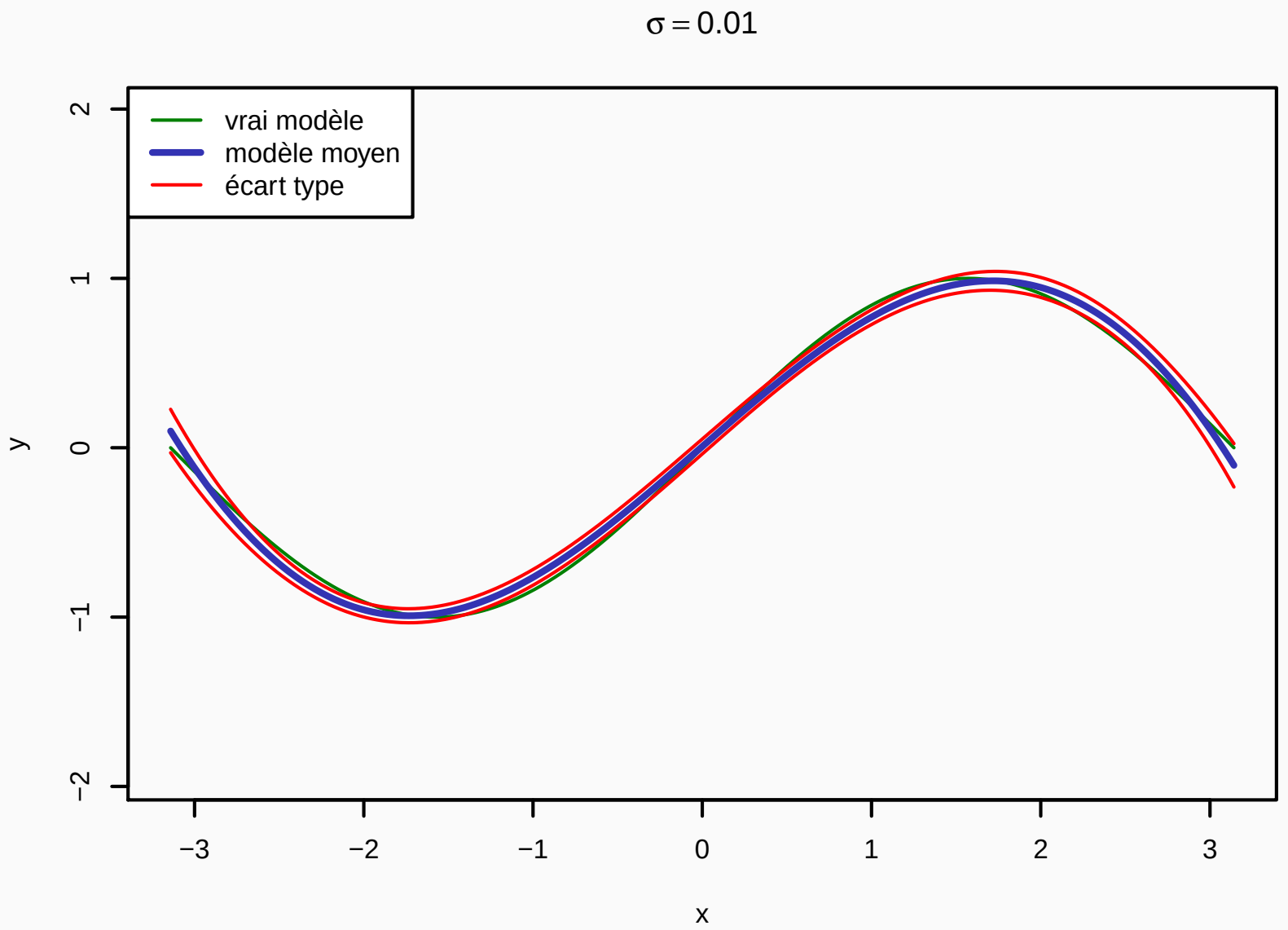
Choix automatique de λ



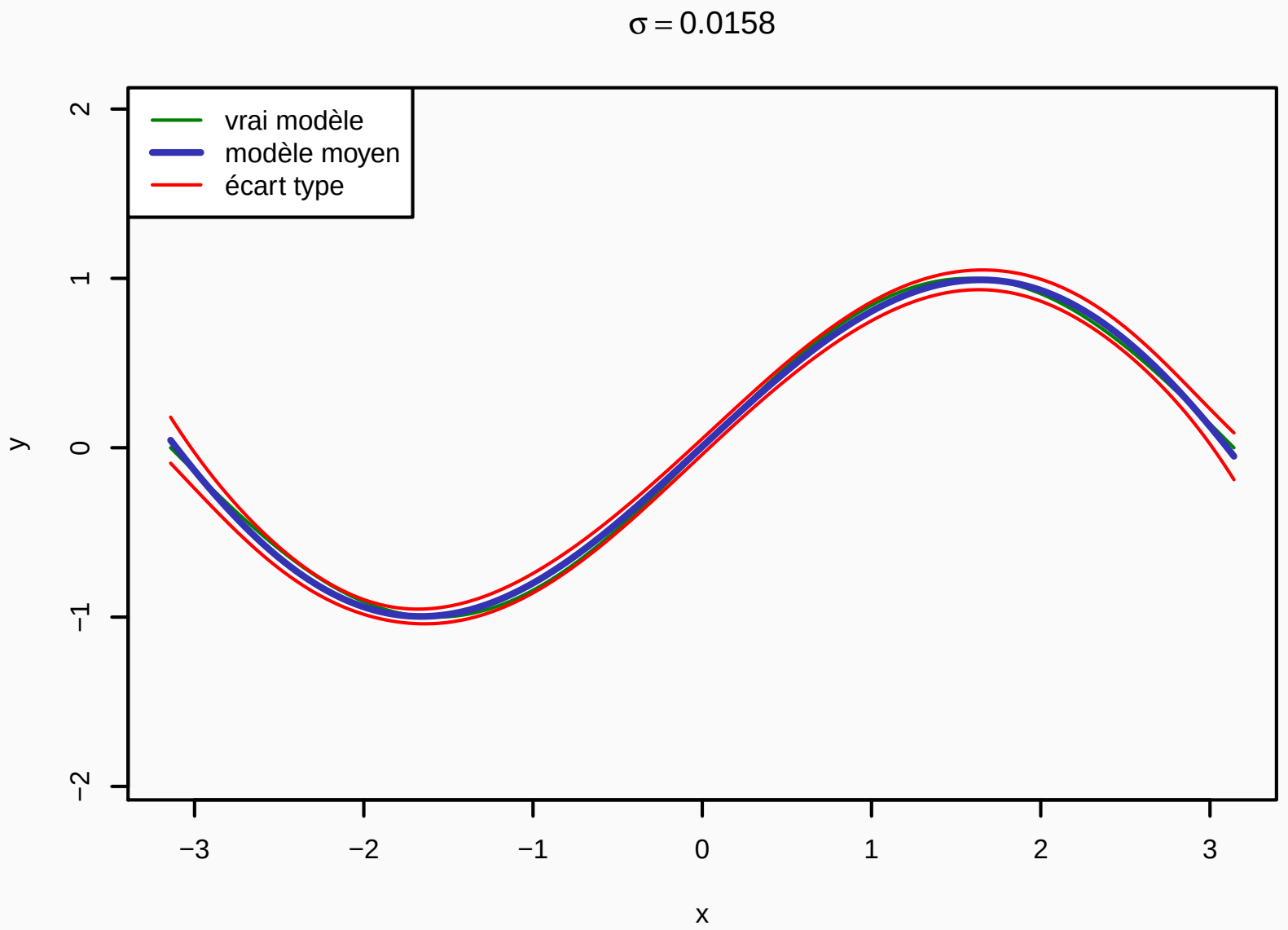
Choix automatique de λ



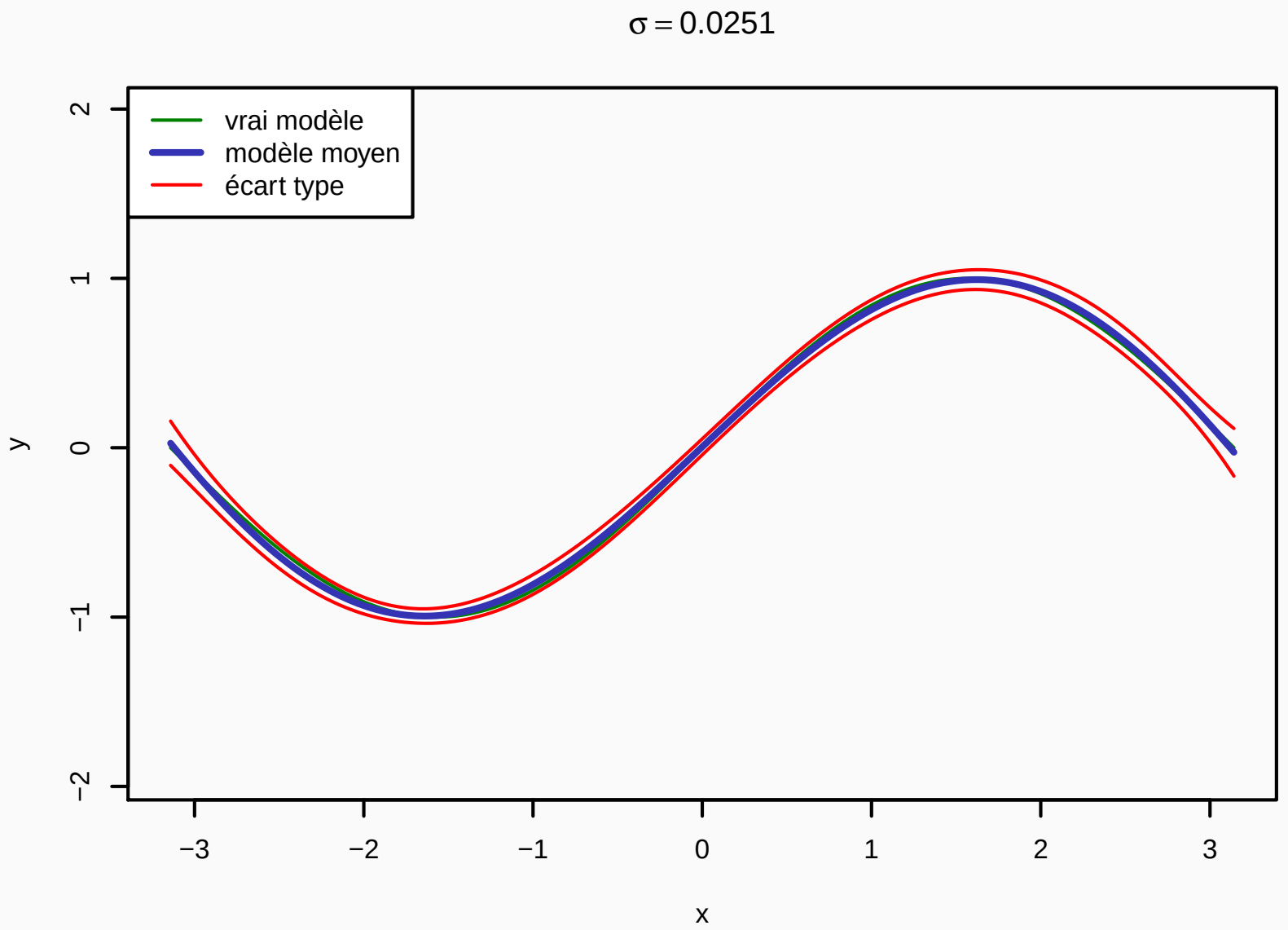
Choix automatique de λ



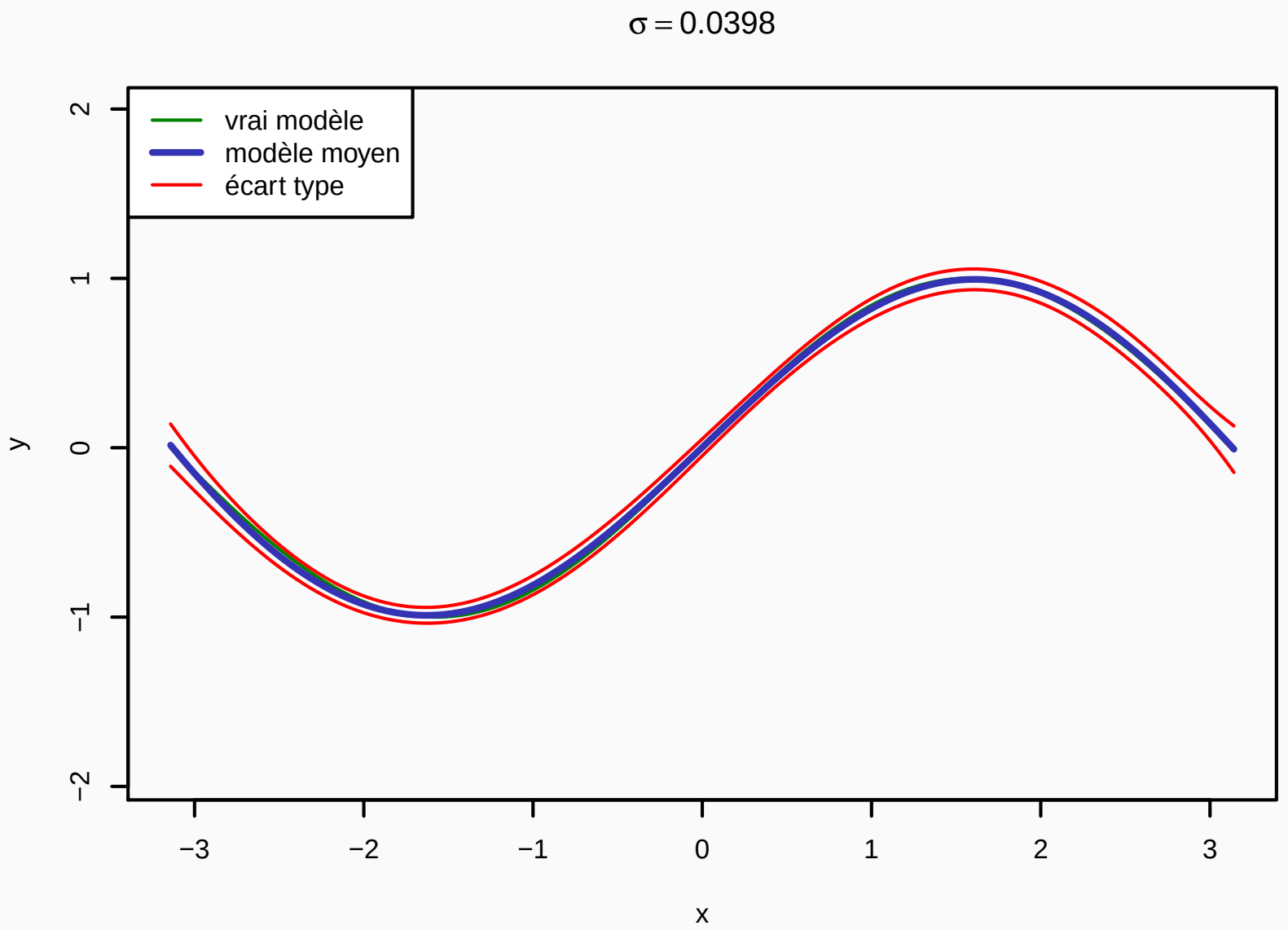
Choix automatique de λ



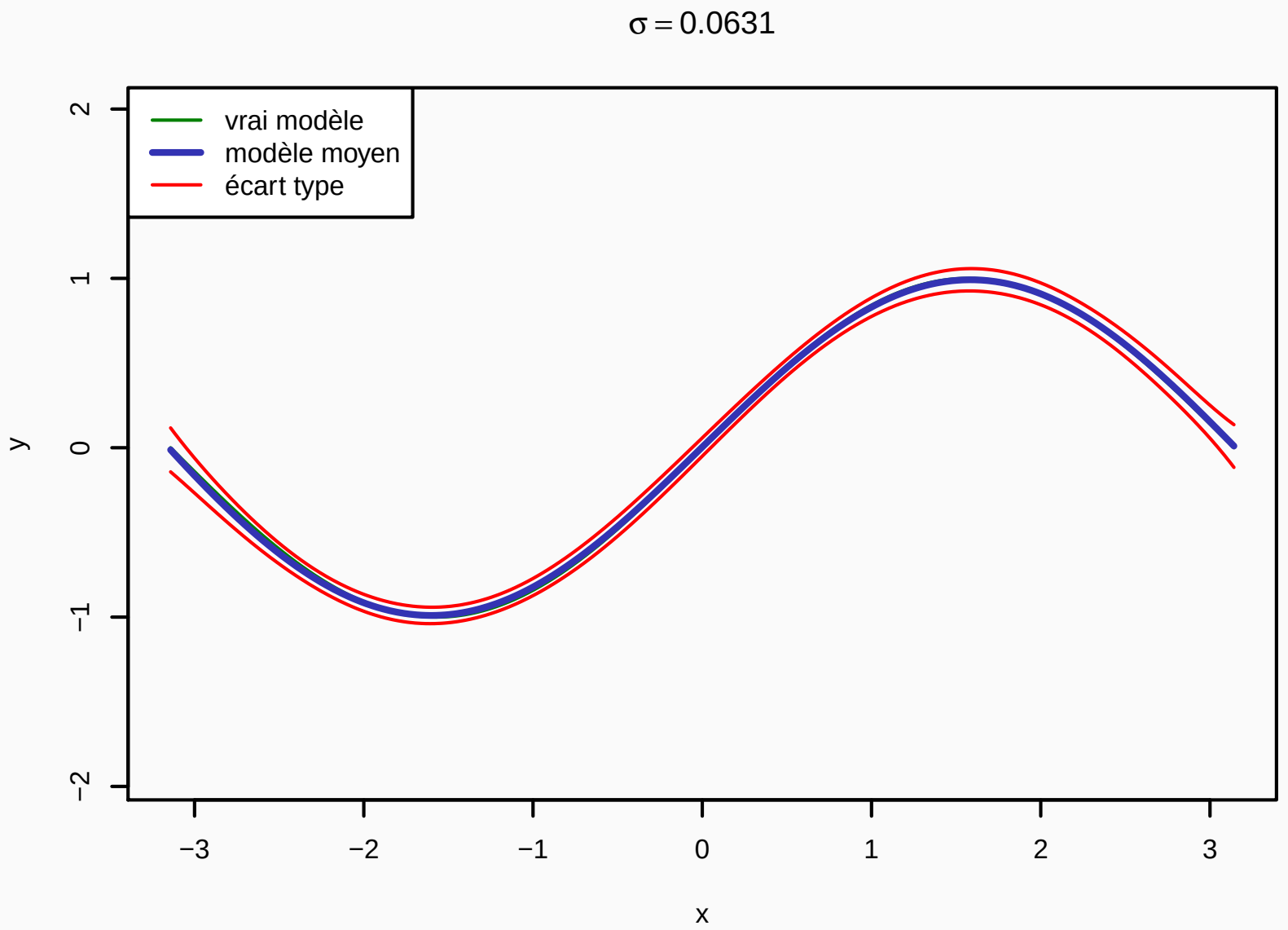
Choix automatique de λ



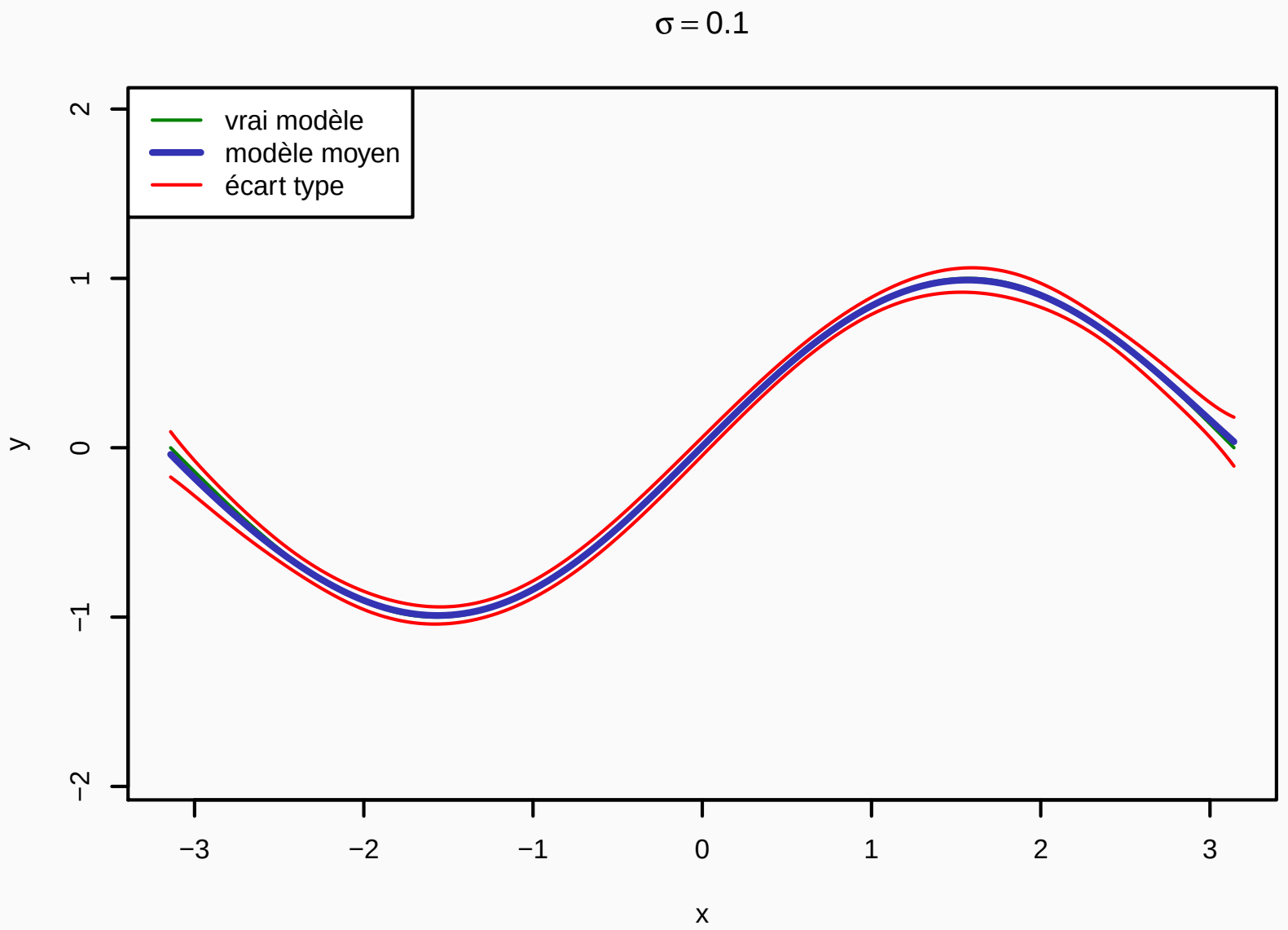
Choix automatique de λ



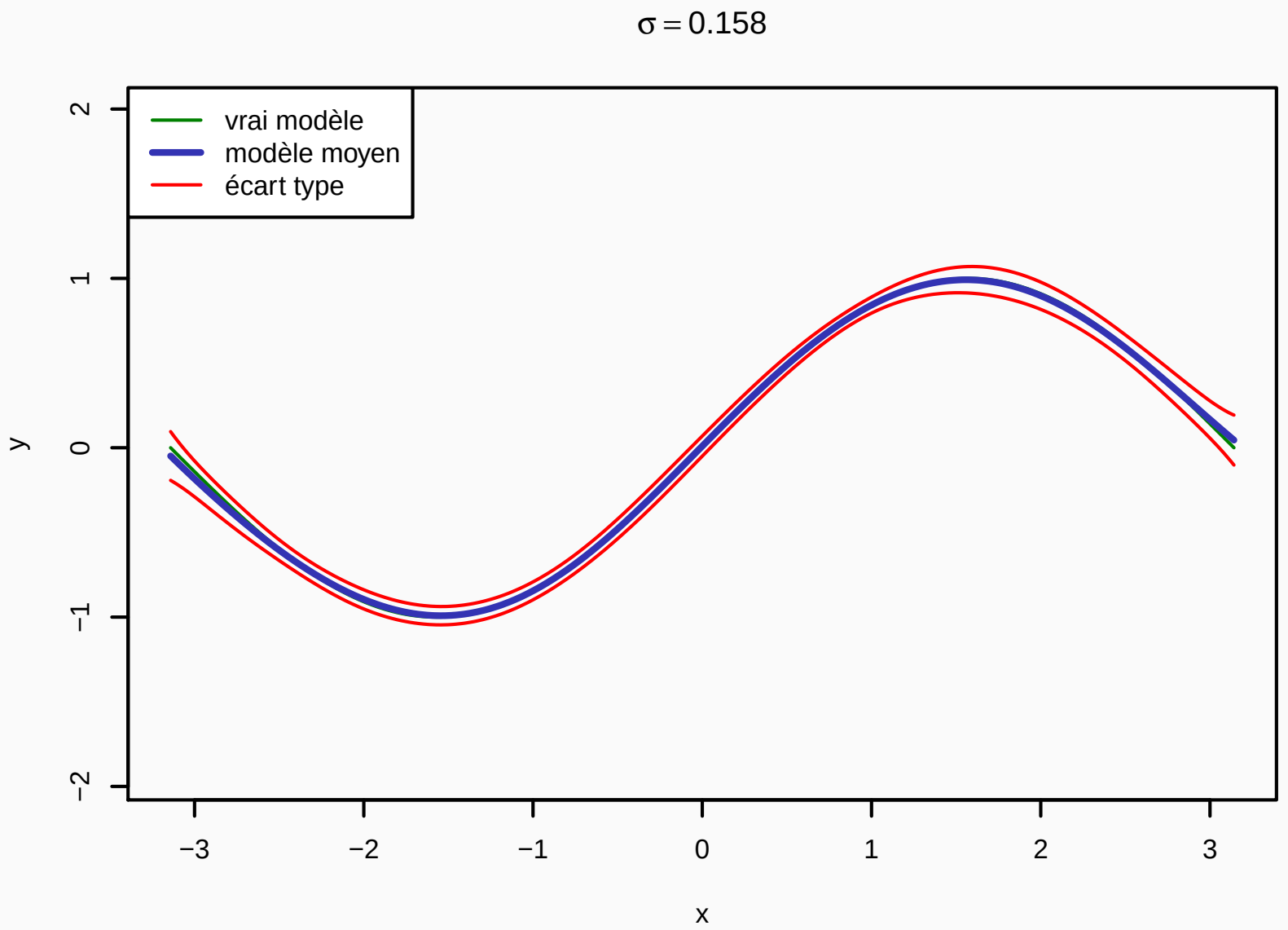
Choix automatique de λ



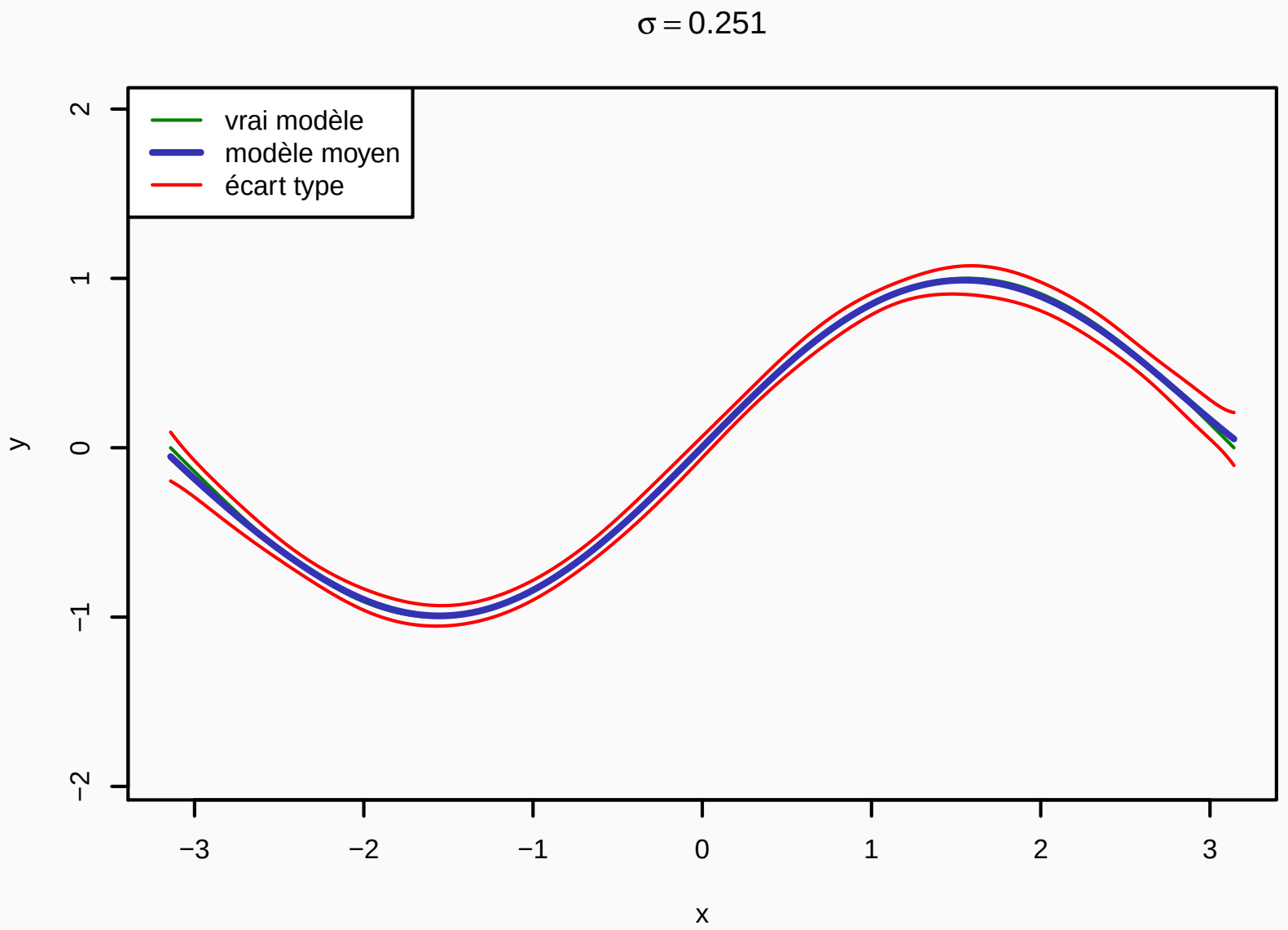
Choix automatique de λ



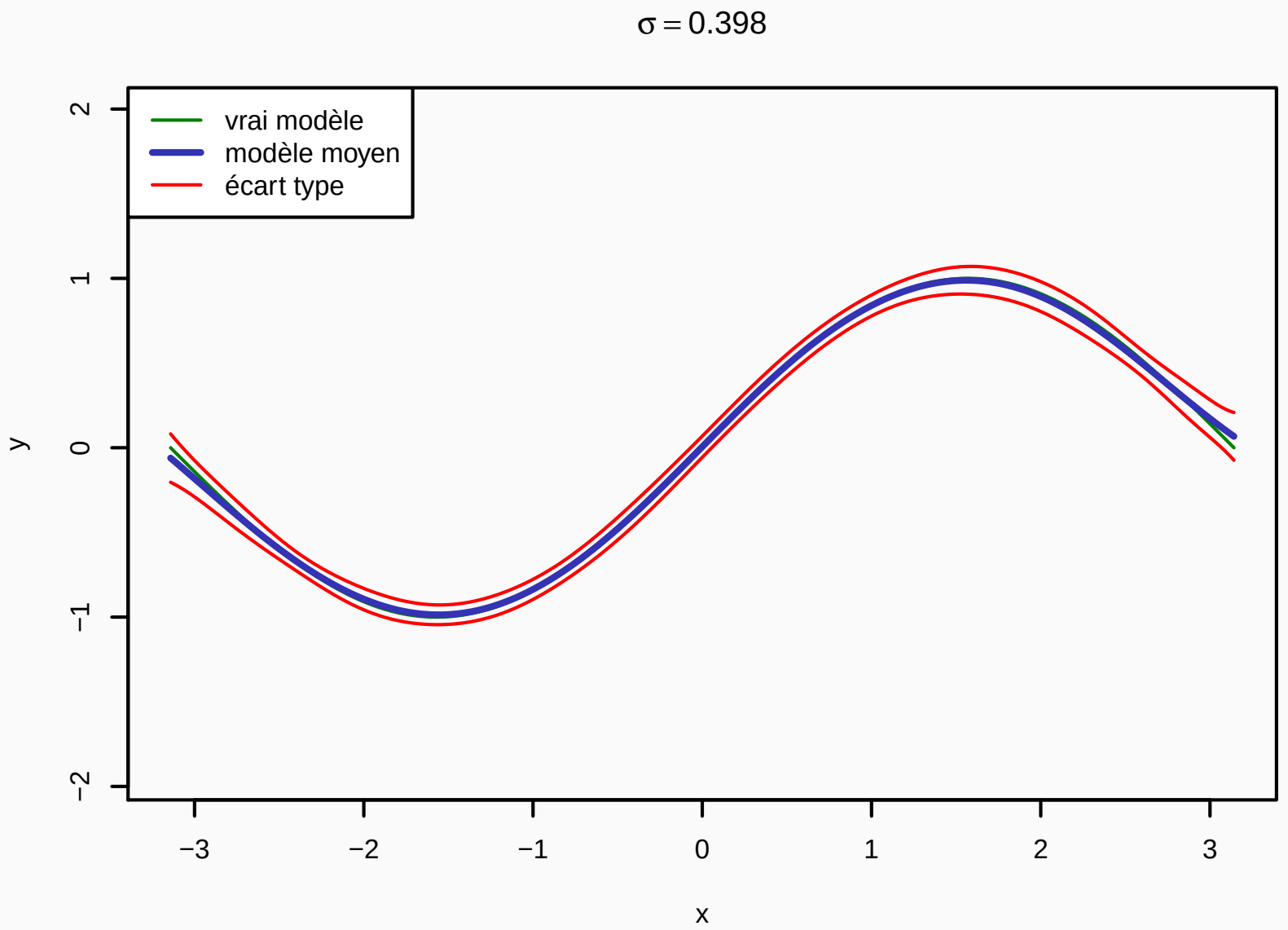
Choix automatique de λ



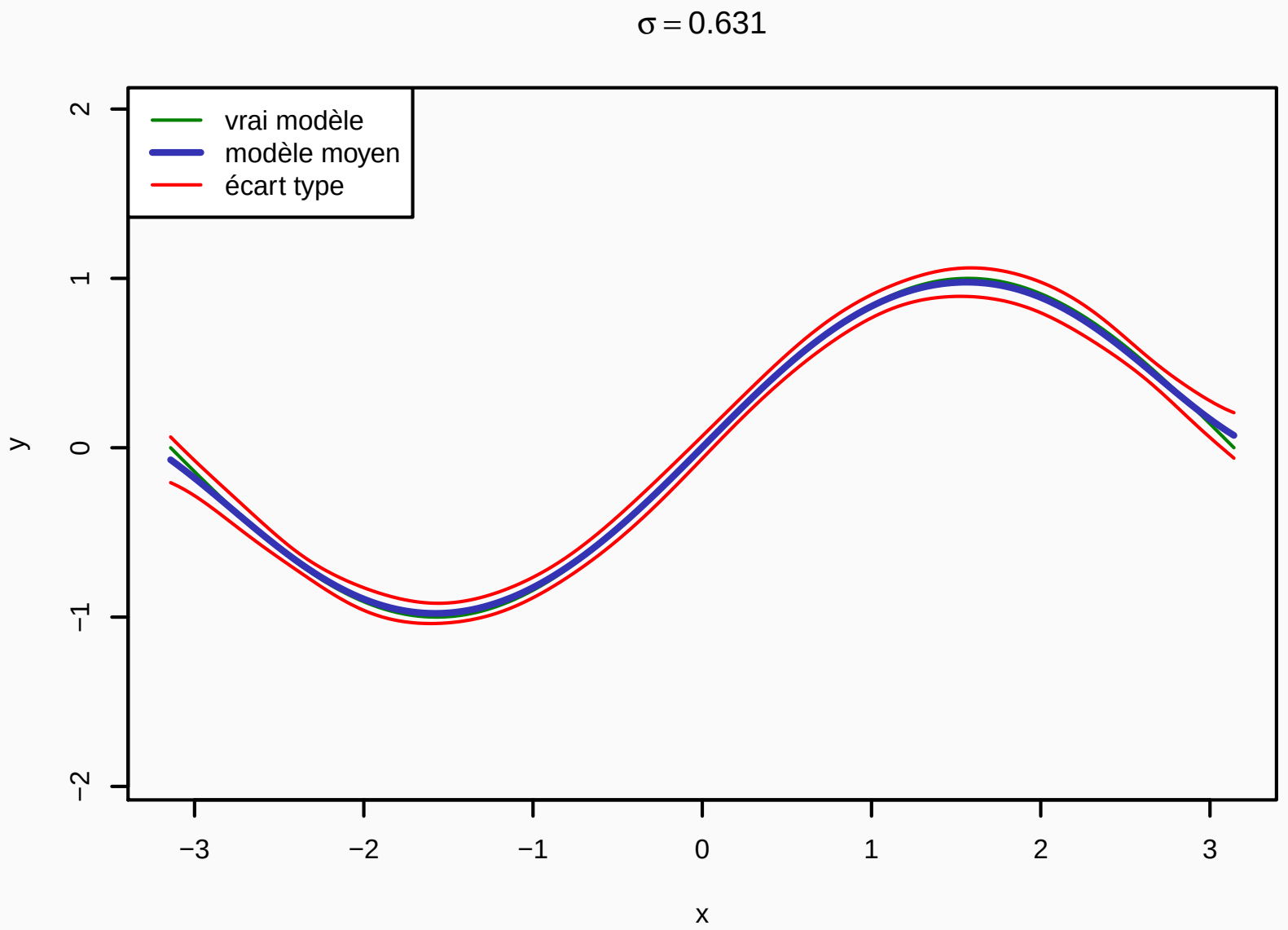
Choix automatique de λ



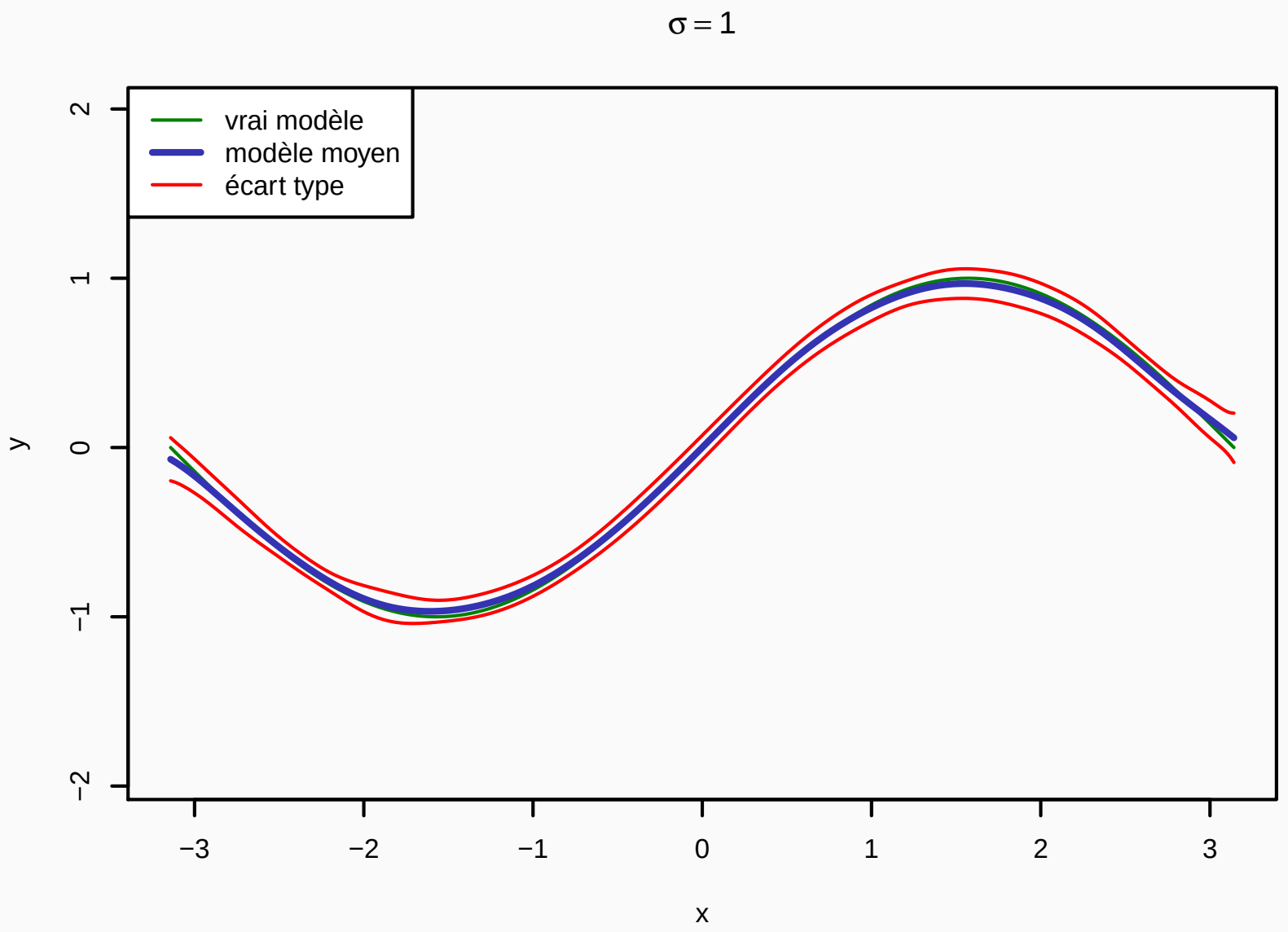
Choix automatique de λ



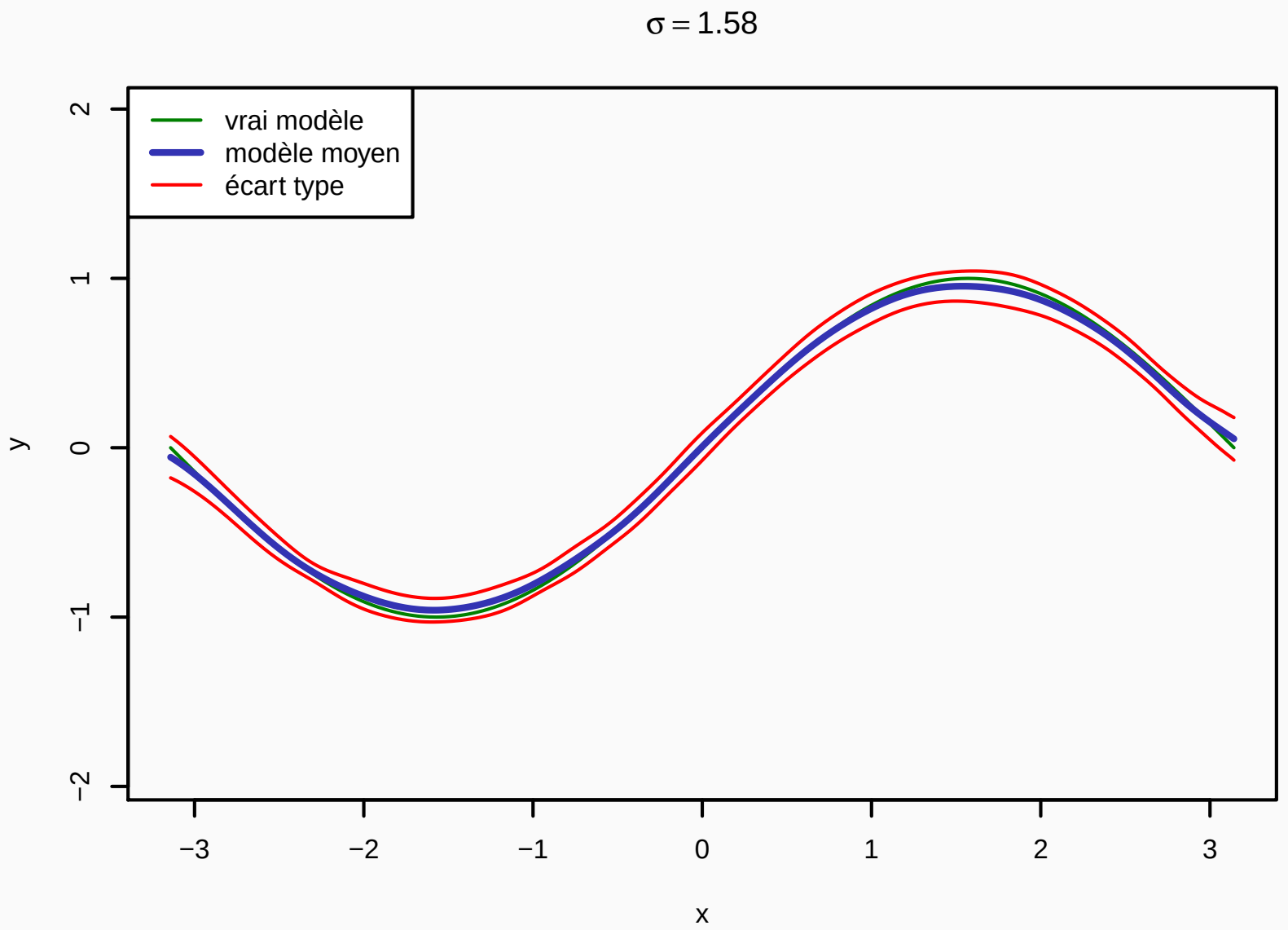
Choix automatique de λ



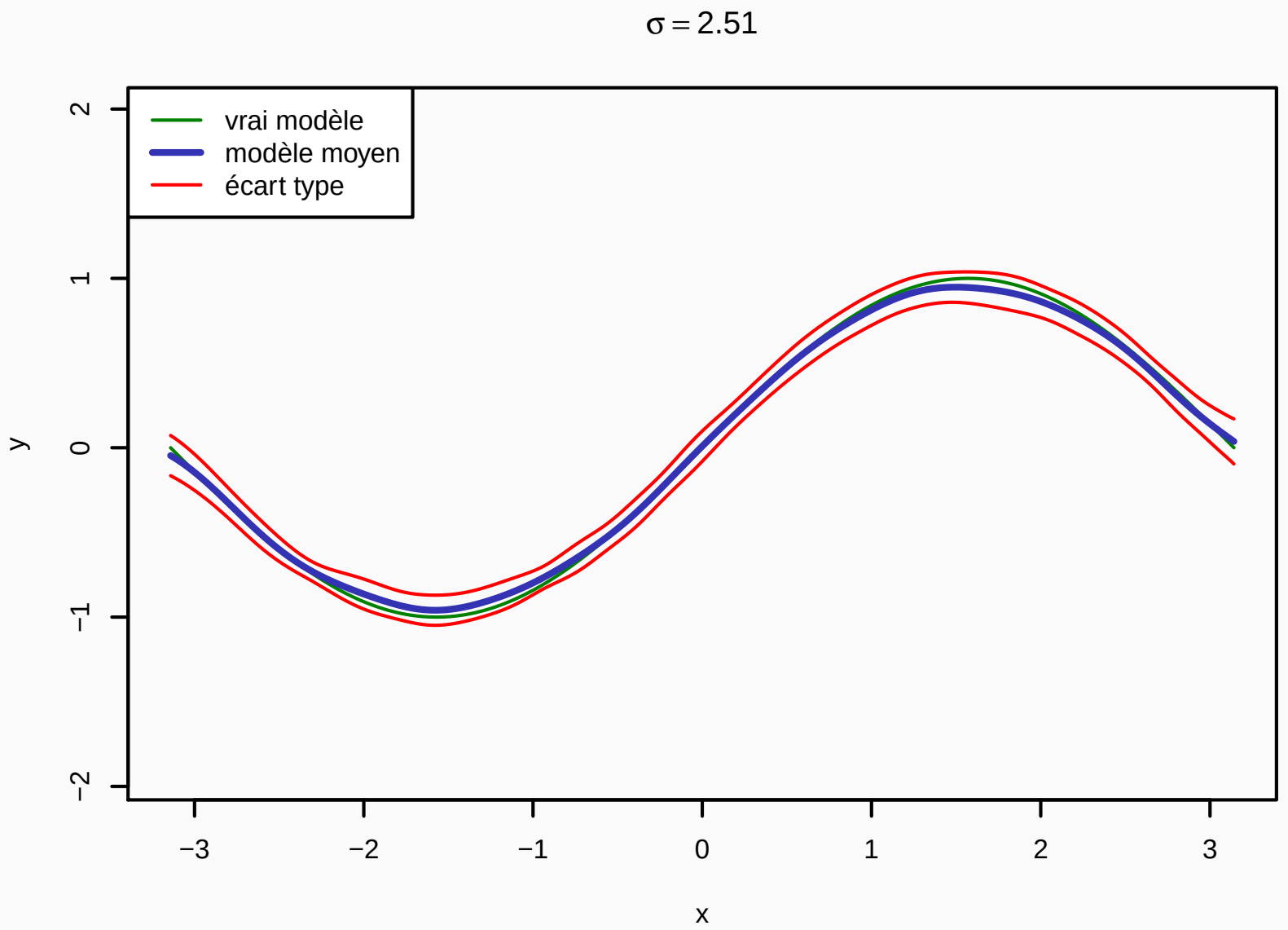
Choix automatique de λ



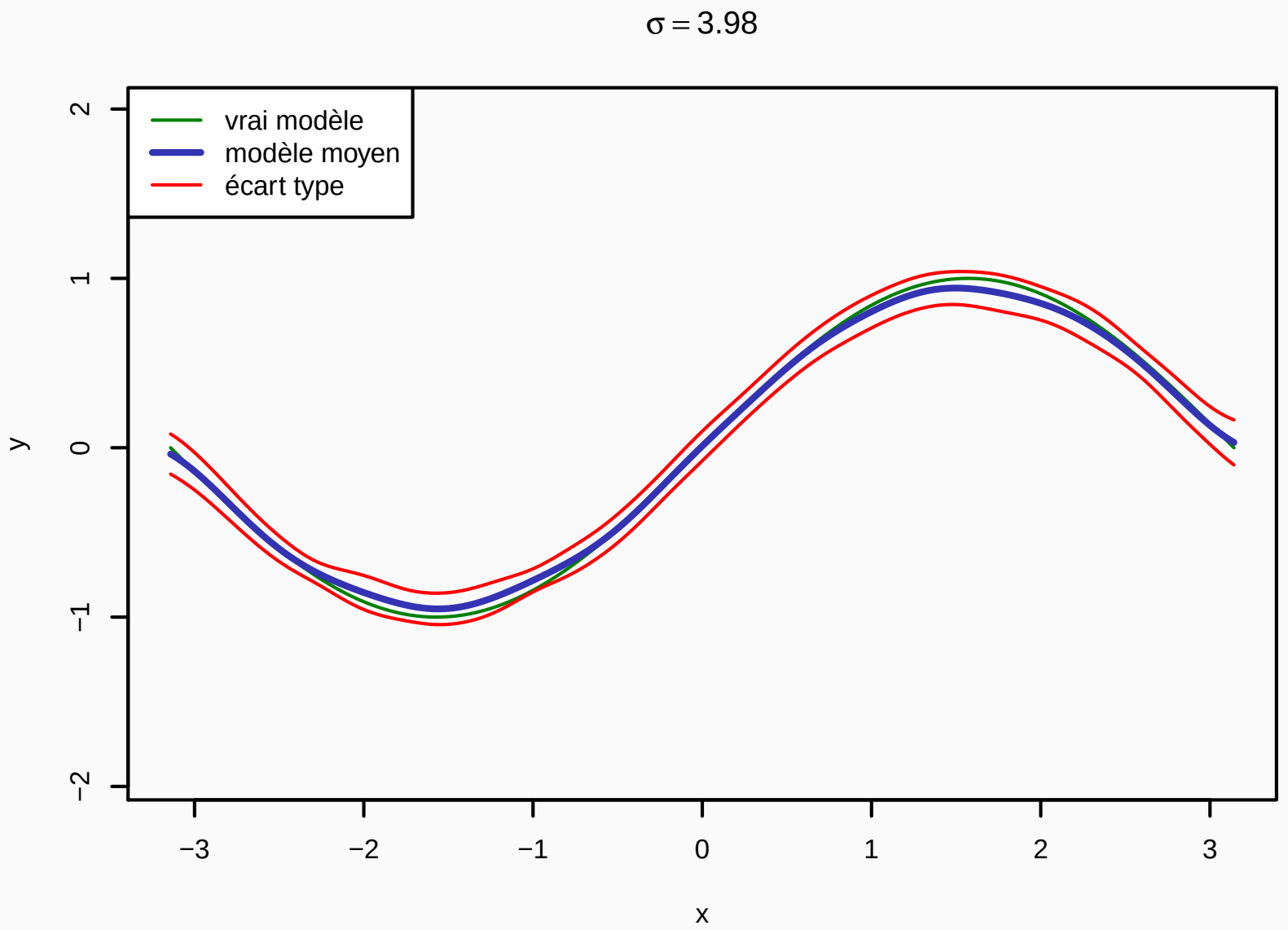
Choix automatique de λ



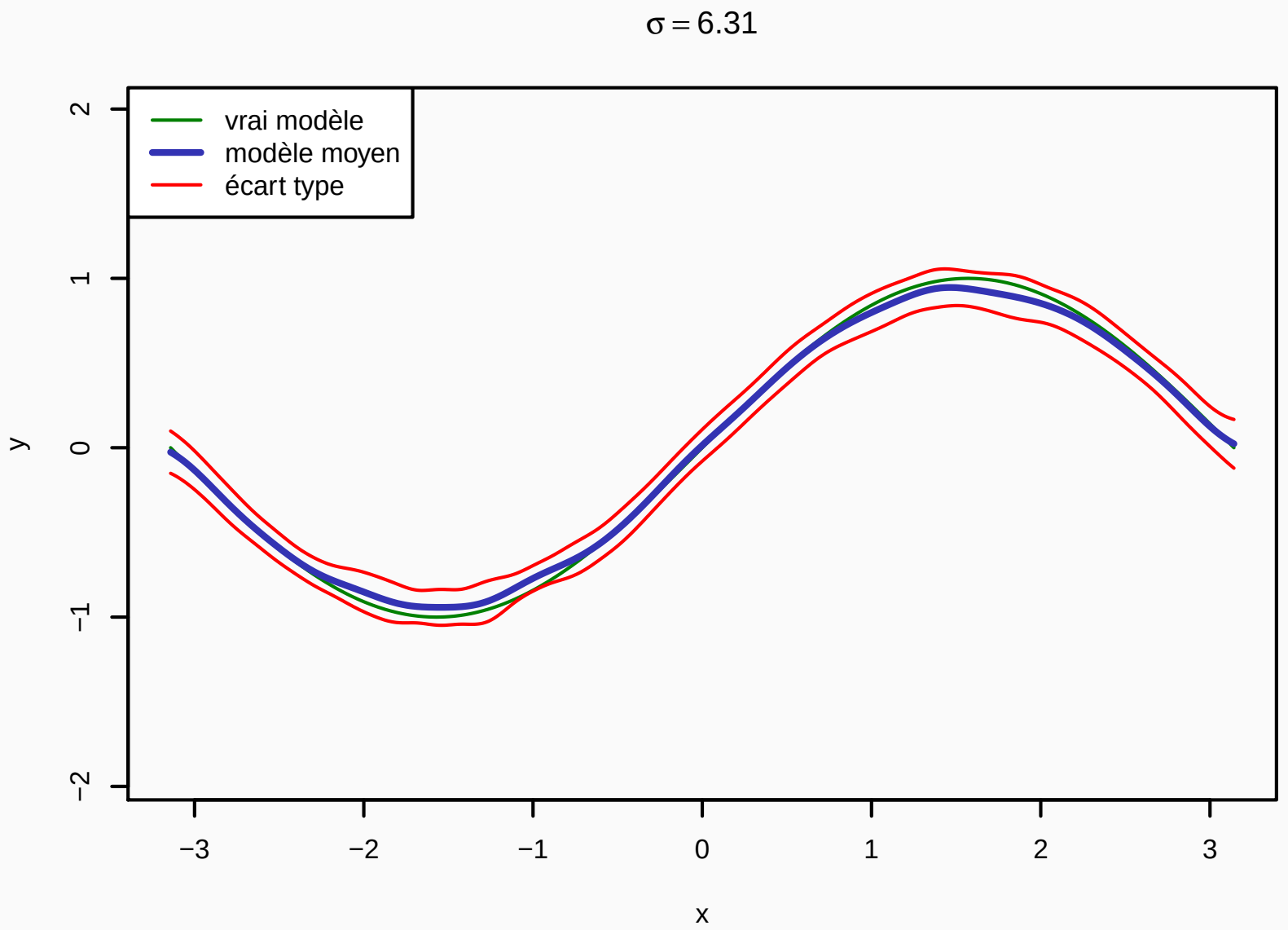
Choix automatique de λ



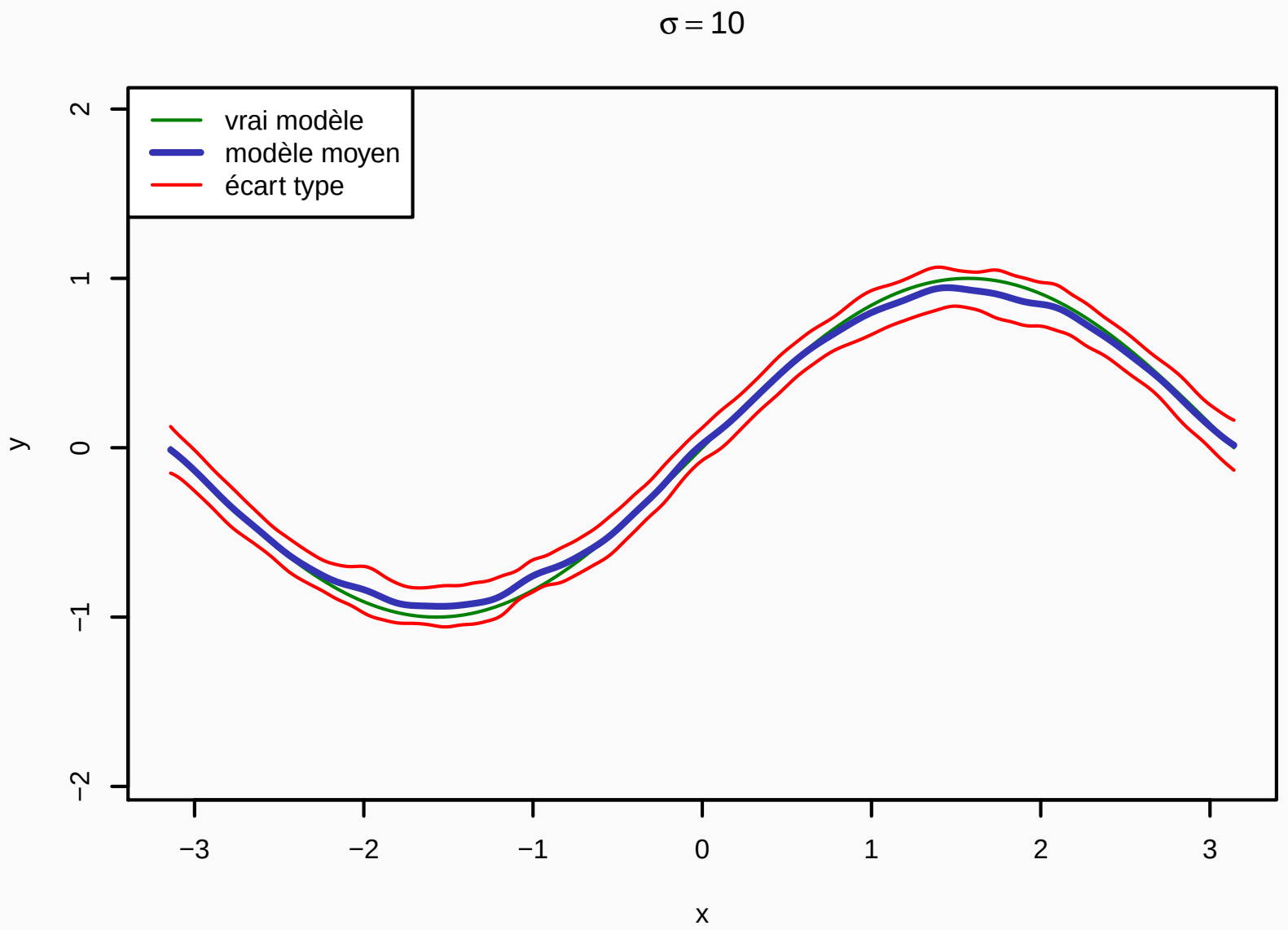
Choix automatique de λ



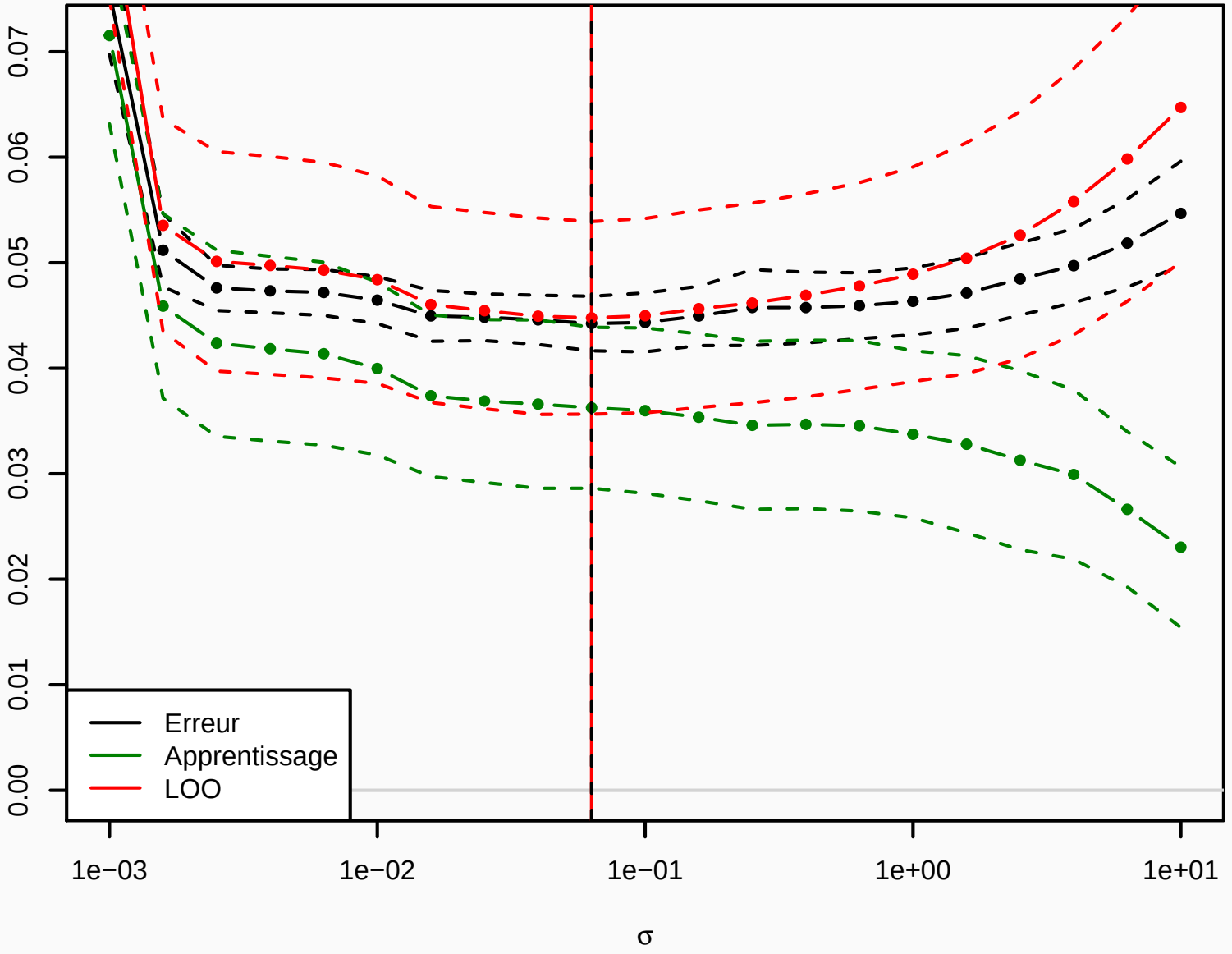
Choix automatique de λ



Choix automatique de λ



Résultats


0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
Erreur
Apprentissage
LOO
1e−03
1e−02
1e−01
1e+00
1e+01
σ

Plan

Introduction

Compromis biais variance

Leave-one-out

Validation croisée

Bootstrap

Données structurées

Validation croisée

Objectifs

- réduire la variance du leave-one-out : plus d'exemples en évaluation
- réduire le temps de calcul du leave-one-out : moins de modèles à construire
- conserver un rôle identique à chaque observation

Solution

- partition aléatoire de $\{1,\ldots,N\}$ en K blocs (les *folds*), $C_1,\ldots,C_K$
- $\kappa(i)$: numéro du bloc de l'observation (X_i, Y_i)
- g_{-k} : modèle appris sur les données dans tous les blocs *sauf* C_k
- estimateur du risque

$$\hat{L}_{cv}(g) = \frac{1}{N}\sum_{i=1}^{N} l(g_{-\kappa(i)}(X_i), Y_i)$$

Conséquences

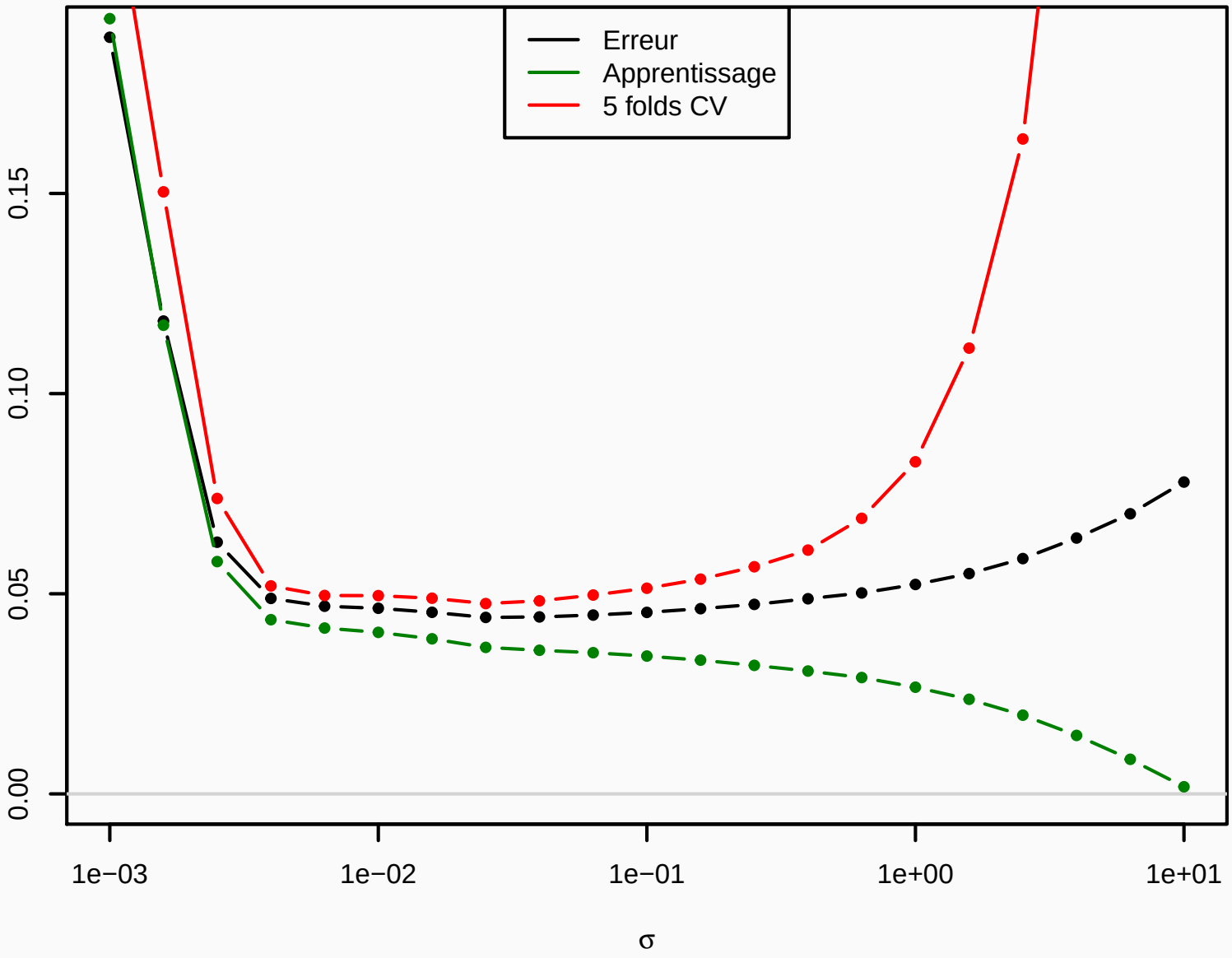
Compromis

- plus K est grand
  - plus le temps de calcul est élevé (attention, c'est plus subtil que ça !)
  - plus le biais diminue : les données d'apprentissage grossissent avec K
  - plus la variance augmente : moins de données d'évaluation
- on considère qu'un bon compromis est K entre 5 et 10

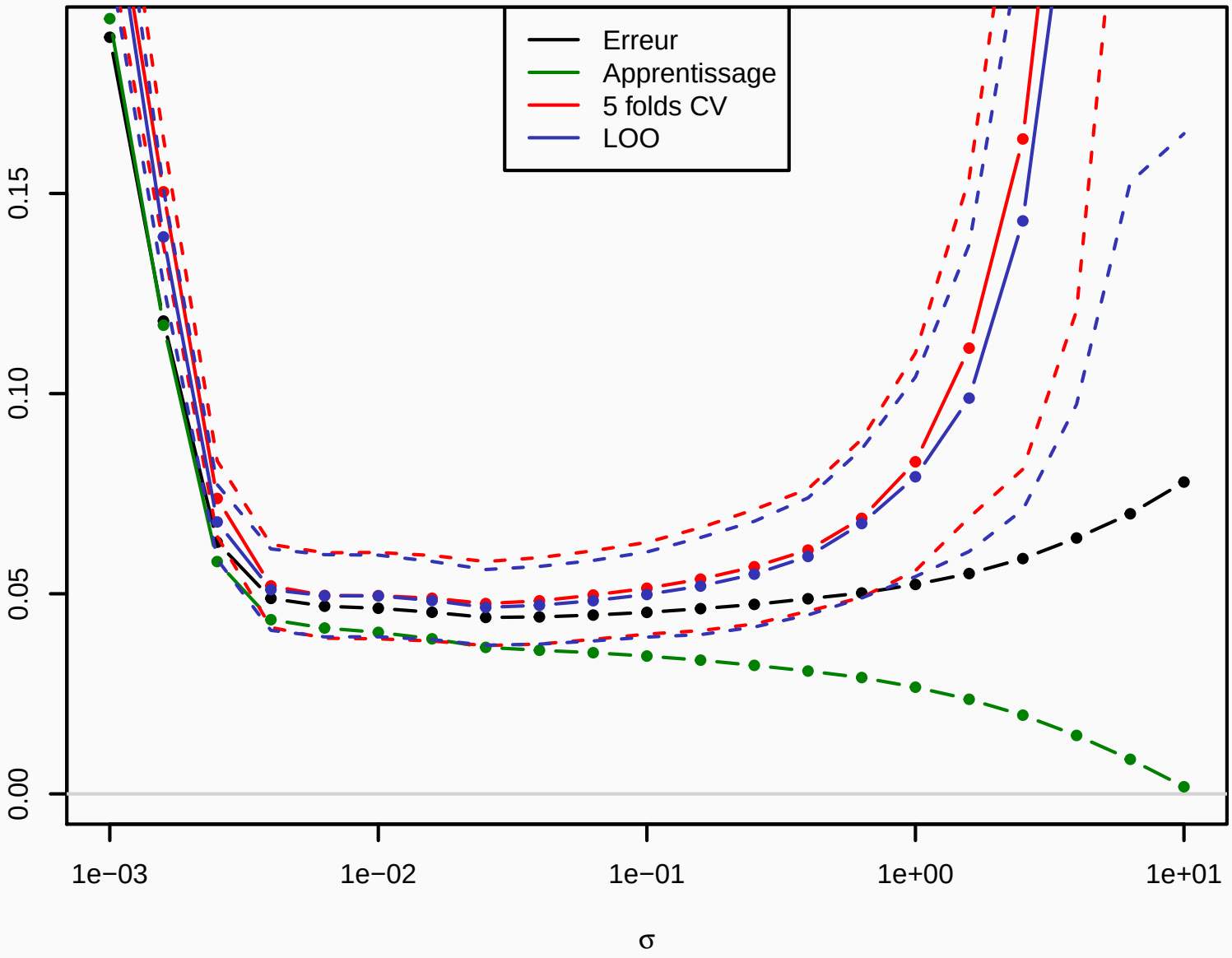
Difficultés

- pas de version rapide
- effet du choix de K ?
- effet du choix des blocs ?

Validation croisée (λ fixé)



Validation croisée vs LOO



Stratification

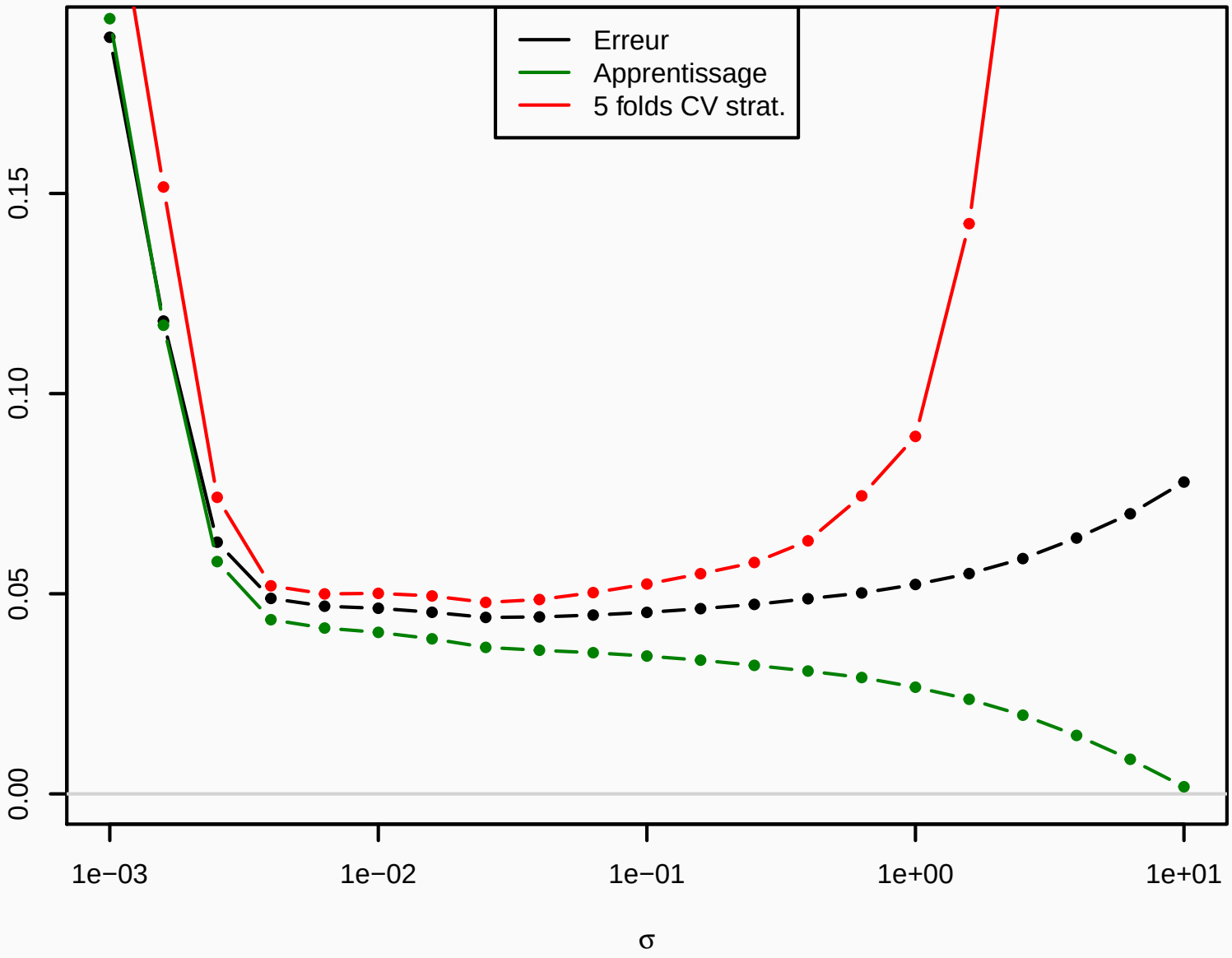
Problème

- les blocs sont déterminés aléatoirement
- la distribution des données dans un bloc peut être « trop différente » de la distribution globale

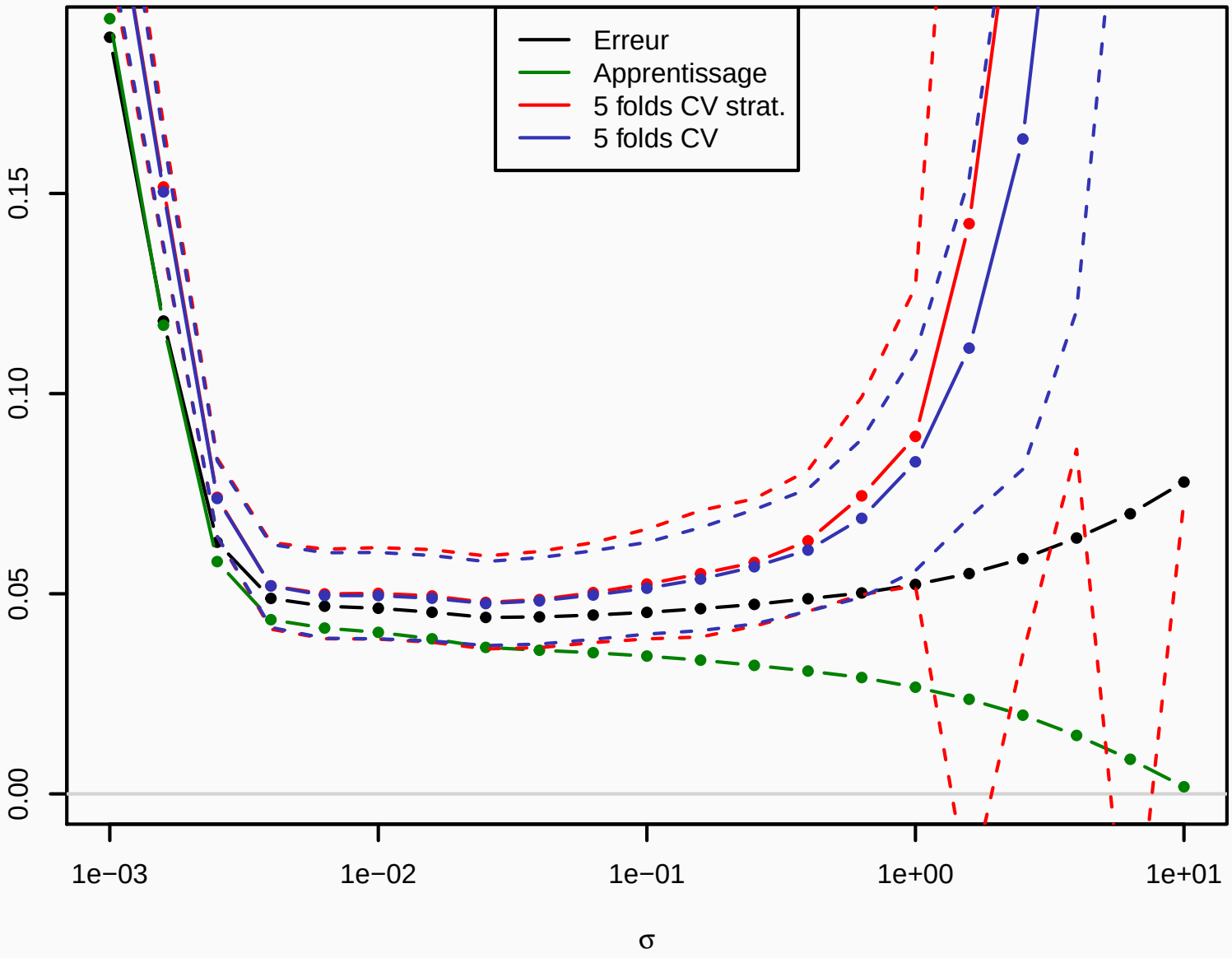
Solution : stratification

- méthode générale : on s'arrange pour que la variable Y soit distribuée de façon identique dans chaque bloc
- en pratique :
  - cas Y discrète : on cherche à avoir $\mathbb{P}(Y = y|C_k) = \mathbb{P}(Y = y)$
  - cas Y continue : on cherche à avoir $\mathbb{P}(Y \in U|C_k) = \mathbb{P}(Y \in U)$ pour quelques U bien choisis (par exemple un découpage basé sur les percentiles de Y)
- systématique pour Y discrète, plus controversée pour Y continue
- extension à X, mais controversée

Validation croisée stratifiée



Validation croisée stratifiée vs de base



Piège classique

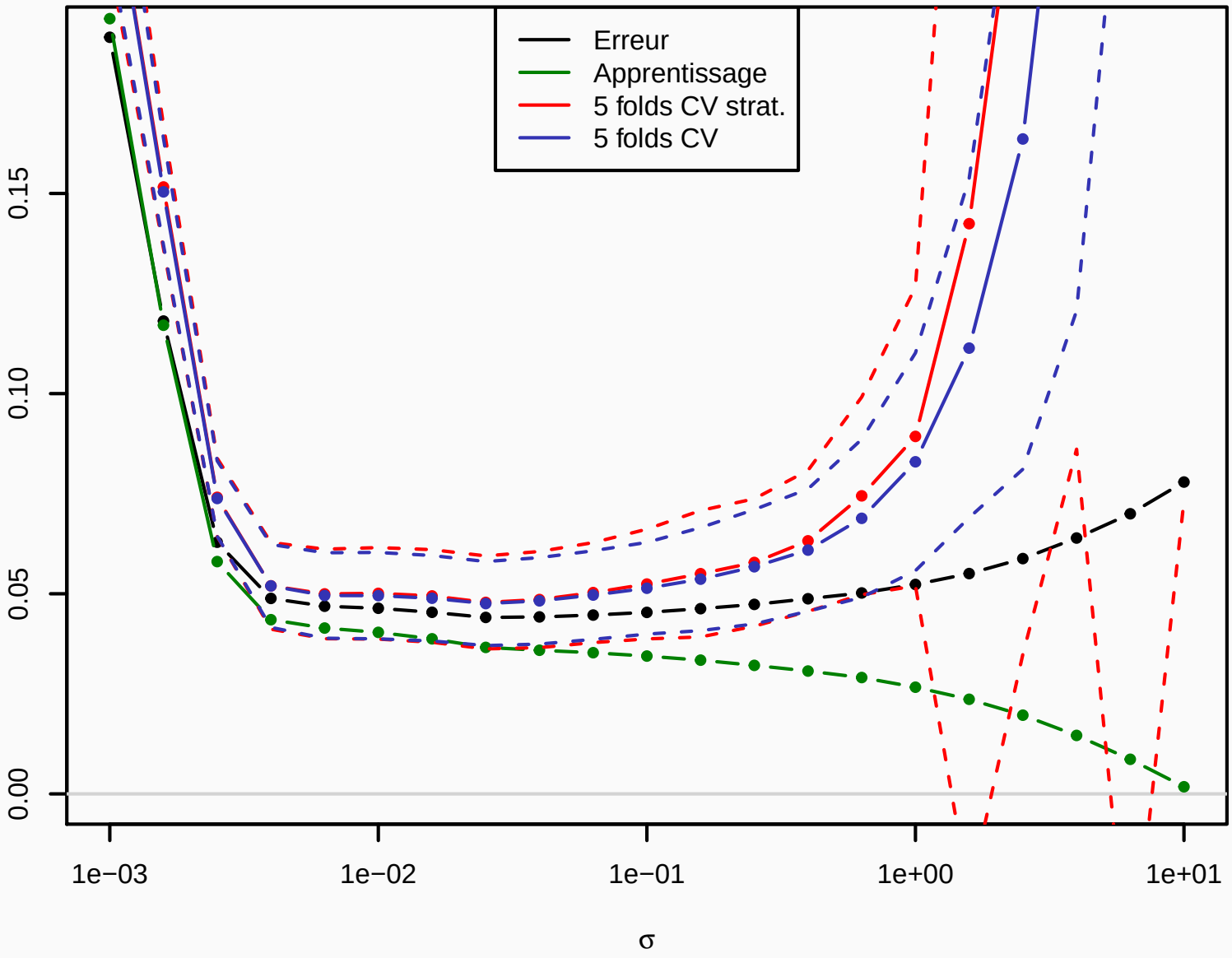
Effets de la variance de la validation croisée

- les estimateurs $\hat{L}_{cv}(g)$ et $\hat{L}_{loo}(g)$ ont une variance non négligeable (comparativement à la variance naturelle induite par les données)
- peut-on comparer $\hat{L}_{cv}(g_1)$ et $\hat{L}_{cv}(g_2)$?
- en théorie : pas directement, il faut tenir compte de la variance

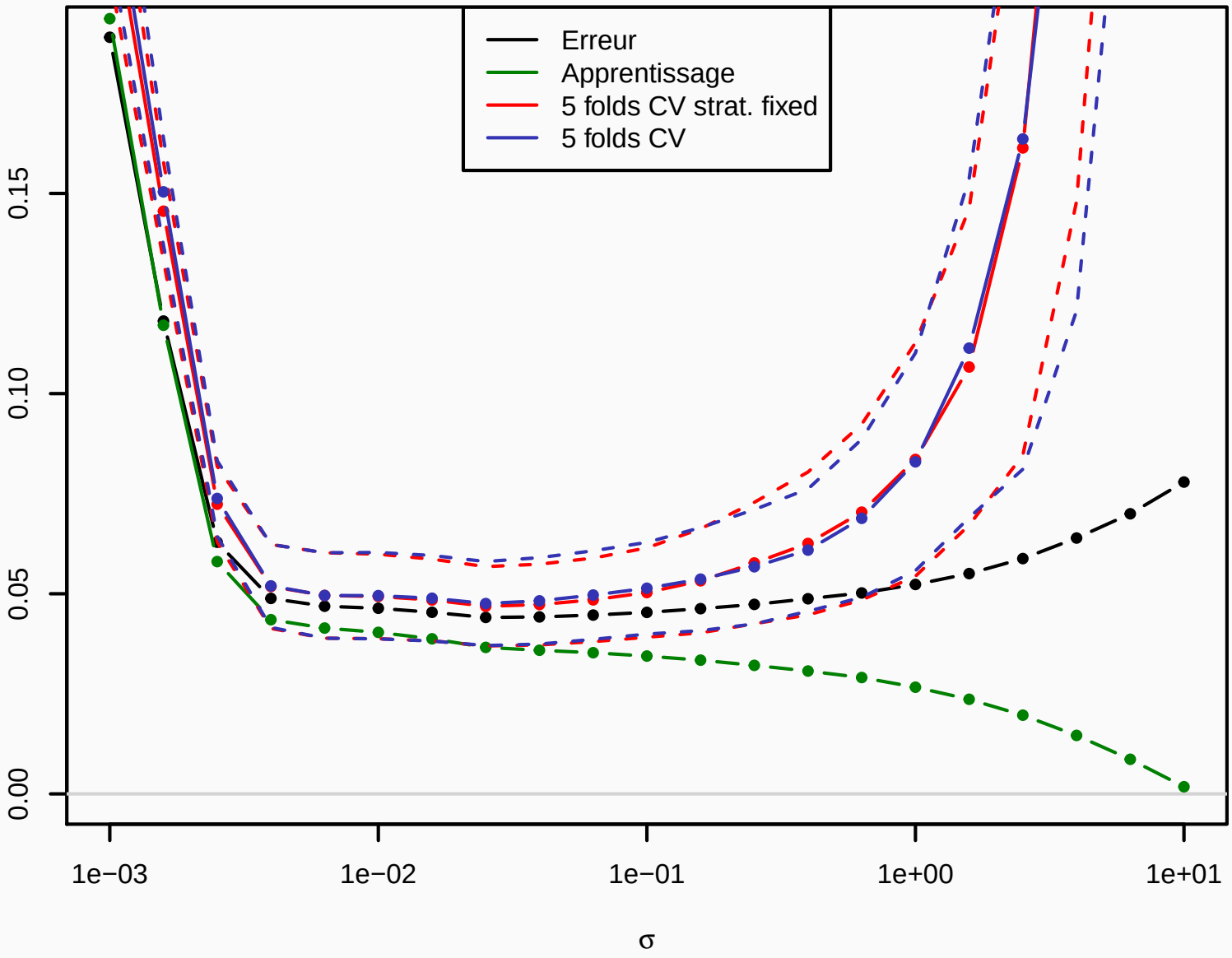
Solutions

1. technique coûteuse :
   - plusieurs validations croisées : $\hat{L}_{cv,1}(g), \ldots, \hat{L}_{cv,p}(g)$
   - tests appariés (Wilcoxon) : on compare $\hat{L}_{cv,j}(g_1)$ et $\hat{L}_{cv,j}(g_2)$ sur les mêmes blocs
2. technique plus simple : on estime la variance par $\widehat{\mathrm{Var}}_{cv}(g) = \frac{1}{N}\sum_{i=1}^{N}\left(l(g_{-\kappa(i)}(X_i), Y_i) - \hat{L}_{cv}(g)\right)^2$ (test t de Welch)
3. variante basique : une seule validation croisée à blocs identiques pour tous les modèles

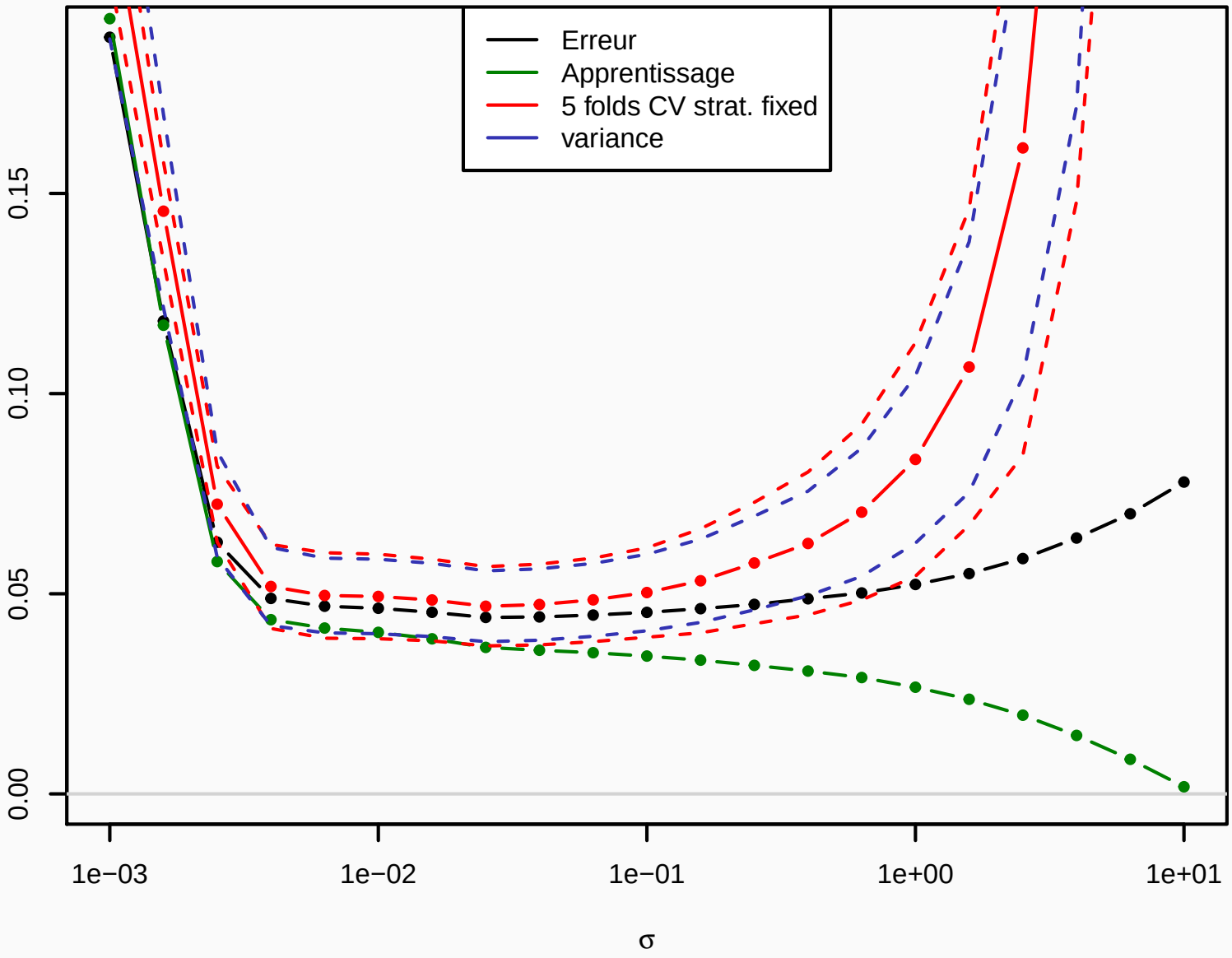
Validation croisée stratifiée



Validation croisée stratifiée unique



Estimation basique de la variance

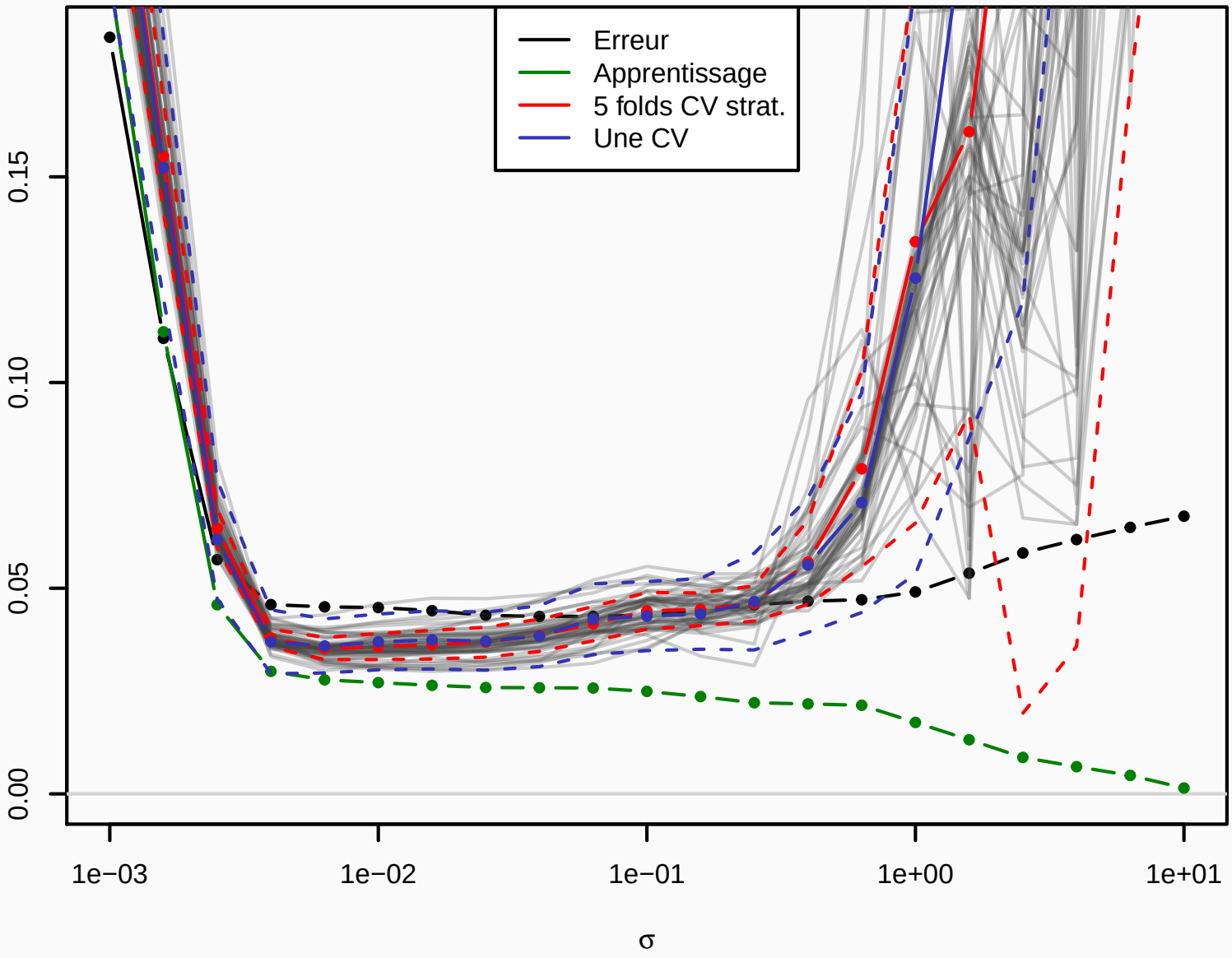


Validations croisées

Exemple

- on reprend le même exemple
- un jeu de données unique
- 50 validations croisées stratifiées

Validations croisées



Bilan

Comportement général

- plus le modèle est complexe plus le tirage des blocs a d'effets
- la variance de la perte estime plutôt bien la variance induite par le tirage des blocs
- la stratification a des effets complexes sur la variance de l'estimateur

Bonnes pratiques

- à minima, comparer les modèles sur les mêmes blocs (une seule partition fixée)
- si possible, étudier l'effet du tirage des blocs, au moins sur quelques modèles importants

Concrètement

K plus proches voisins

1. tirer un découpage en P blocs, $C_1, \ldots, C_P$
2. pour k allant de 1 à N (impair seulement)
   - 2.1 pour i allant de 1 à p
     - 2.1.1 « calculer » le classifieur $g_{-i,k}$ des k plus proches voisins sur tous les blocs sauf C_i
     - 2.1.2 calculer ses prévisions sur C_i
   - 2.2 calculer $\hat{L}_{cv}(g_k)$ à partir des prévisions sur les C_i
3. $k^* = \arg\min_k \hat{L}_{cv}(g_k)$
4. « calculer » le classifieur g_{k^*} des k^* plus proches voisins sur l'ensemble des données

Attention

- $\hat{L}_{cv}(g_{k^*})$ est un estimateur biaisé de $L(g_{k^*})$
- en théorie, il faudrait utiliser un nouveau jeu de données pour estimer $L(g_{k^*})$
- ou une nouvelle validation croisée...

Validation croisée aléatoire

Objectifs

- séparer la proportion apprentissage/validation du nombre de modèles construits
- réduire l'impact du tirage des blocs

Méthode

- paramètres : une proportion p et un nombre de tirages B
- on tire aléatoirement B sous-ensembles de $\{1,\ldots,N\}$, $A_1,\ldots,A_B$ avec $|A_k| = Np$
- g_b : modèle appris sur A_b
- estimateur du risque

$$\hat{L}_{rrcv}(g) = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{N-|A_b|}\sum_{i\notin A_b} l(g_b(X_i),Y_i)$$

En pratique

Paramètres

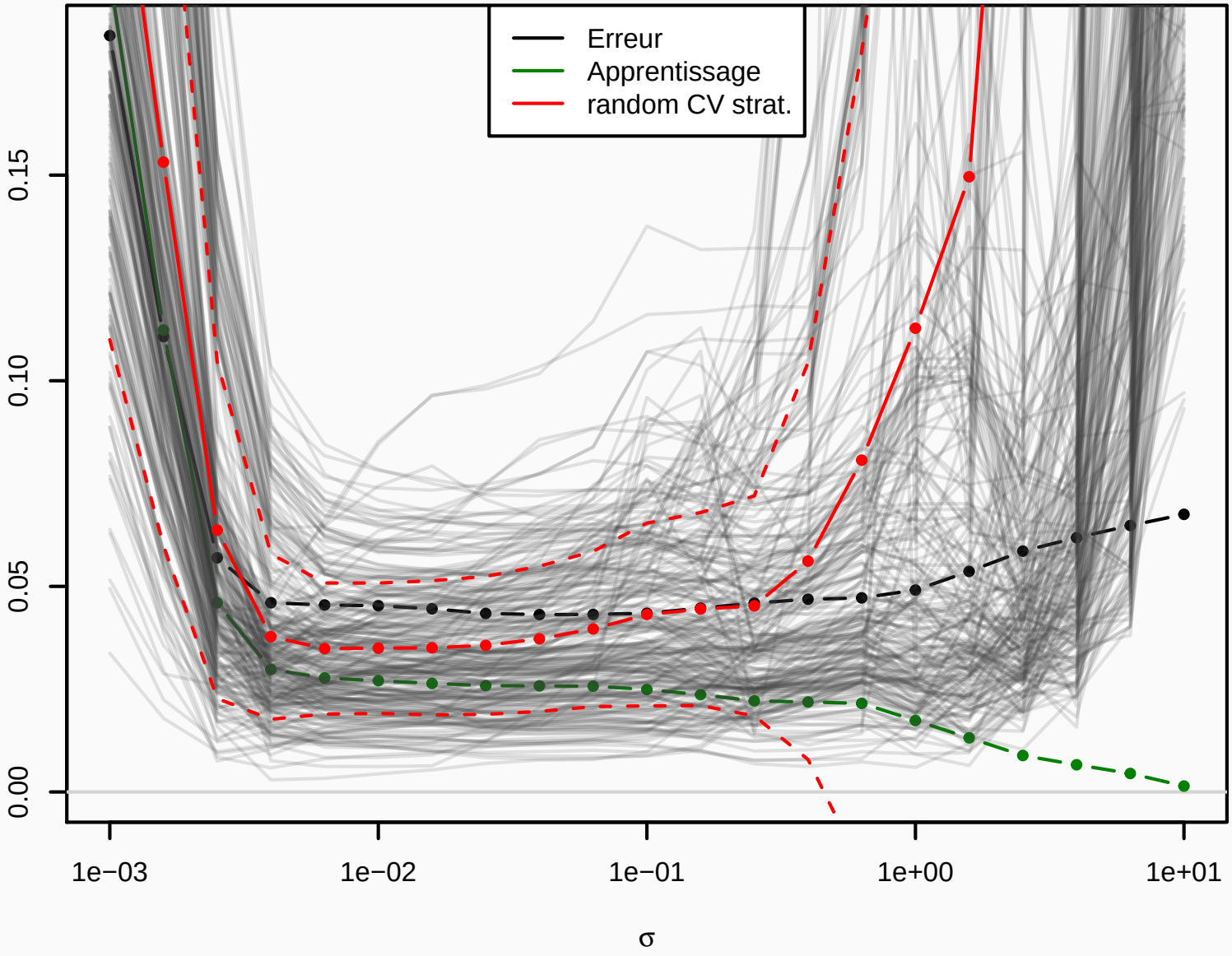
- B :
  - considération de coût en temps de calcul
  - un grand B réduit la variance induite par les tirages
- p :
  - rôle comparable à K pour la validation croisée classique
  - valeurs typiques entre 0,5 et 0,8

Autres aspects

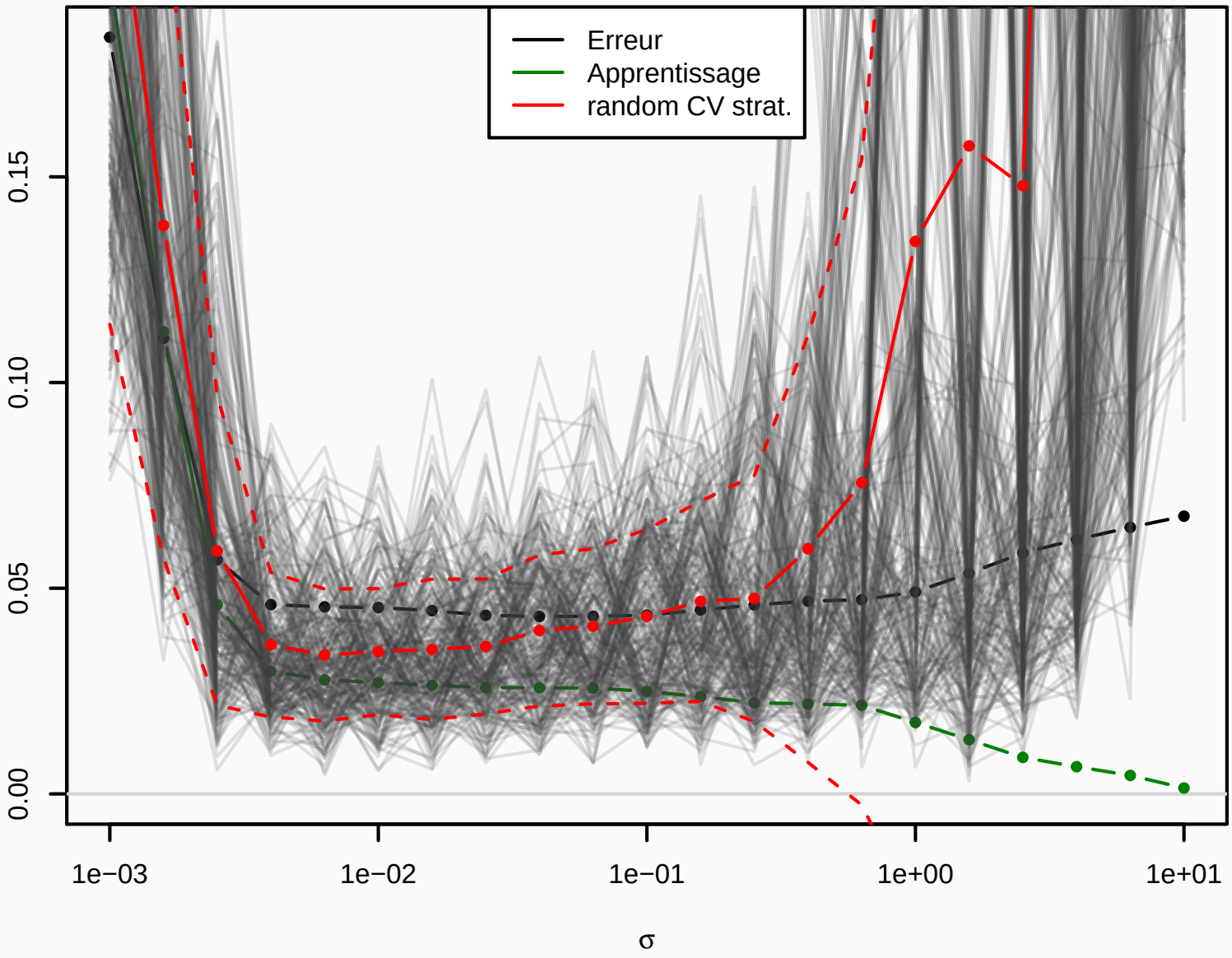
- stratification possible
- appariement conseillé (mêmes tirages pour un ensemble de modèles)

attention très grande variance

Validations croisées aléatoires



Validations croisées aléatoires sans appariement



TP `caret`

Objectifs

- prise en main de `caret`
- expérimentation de la variabilité des résultats

Mise en œuvre

1. installer le *package* `caret` de R
2. sélection automatique des paramètres de la *kernel ridge regression* par validation croisée :
   - 2.1 à partir d'une liste de vecteurs d'indices (les C_k)
   - 2.2 liste engendrée par `createFolds` de `caret`
3. prise en main de la fonction `train` de `caret` :
   - 3.1 fonctionnement « automatique »
   - 3.2 contrôle fin pour les appariements par exemple

Plan

Introduction

Compromis biais variance

Leave-one-out

Validation croisée

Bootstrap

Données structurées

Bootstrap

Méthode statistique

- Estimateur $\hat{\theta} = f(X_1, \ldots, X_N) = f(\mathcal{D})$
- Échantillon bootstrap $\mathcal{D}^b = (X'_1, \ldots, X'_N)$: tirage uniforme avec remise dans $\mathcal{D}$
- on approche la distribution de $\hat{\theta}$ (induite par $\mathcal{D}$) par celle de $\hat{\theta}(\mathcal{D}^b)$

Application

- application canonique : estimation de la variance d'un estimateur
- pour B échantillons (de l'ordre de 500 au moins) :

$$\widehat{\sigma^2}_{boot}(\hat{\theta}) = \frac{1}{B-1} \sum_{b=1}^{B} \left(\hat{\theta}(\mathcal{D}^b) - \hat{\theta}_{boot}(\mathcal{D}) \right)^2,$$

avec

$$\hat{\theta}_{boot}(\mathcal{D}) = \frac{1}{B} \sum_{b=1}^{B} \hat{\theta}(\mathcal{D}^b)$$

Utilisation en apprentissage

Approche originale (Efron 1979 [Efr79])

- estimateur étudié : $\hat{\theta} = L(g) - \hat{L}(g)$ où $\hat{L}(g)$ est le risque empirique

$$\hat{L}(g) = \frac{1}{N} \sum_{i=1}^{N} l(g(X_i), Y_i)$$

- estimation de l'espérance de $\hat{\theta}$ par bootstrap
- on a

$$\hat{\theta}_{boot}(\mathcal{D}) = \frac{1}{B} \sum_{b=1}^{B} \left(\hat{L}(g_b) - \hat{L}_b(g_b) \right),$$

où g_b désigne le modèle appris sur l'échantillon bootstrap $\mathcal{D}^b$ et $\hat{L}_b$ le risque empirique sur $\mathcal{D}^b$

- risque empirique corrigé : $\hat{L}_{boot}(g) = \hat{L}(g) + \hat{\theta}_{boot}(\mathcal{D})$

Utilisation en apprentissage

Estimation directe : *leave-one-out bootstrap* (Efron 1983 [Efr83])

- point de vue proche de la validation croisée aléatoire
- on sait que $\mathcal{D}^b$ contient en moyenne 63,2 % de $\mathcal{D}$
- on peut donc estimer l'erreur sur $\overline{\mathcal{D}^b}$ (les 36,8 % restant)

$$\hat{L}_{loob}(g) = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{|\overline{\mathcal{D}^b}|}\sum_{i\in\overline{\mathcal{D}^b}} l(g_b(X_i), Y_i)$$

- biais parfois important (équivalent à la validation croisée à $K = 2$ blocs)

Techniques de correction du biais

Bootstrap .632 (Efron 1983 [Efr83])

- le *leave-one-out bootstrap* surestime en général le risque
- compensation en combinant avec le risque empirique
- estimateur .632

$$\hat{L}_{.632}(g) = 0{,}368 \times \hat{L}(g) + 0{,}623 \times \hat{L}_{loob}(g)$$

Bootstrap .632+ (Efron & Tibshirani 1997 [ET97])

- amélioration de l'estimateur .632 en cas de sur-apprentissage massif
- taux d'erreur sans information

$$\hat{\gamma} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} l(g(X_i), Y_j)$$

- taux de sur-apprentissage relatif

$$\hat{R} = \frac{\hat{L}_{loob}(g) - \hat{L}(g)}{\hat{\gamma} - \hat{L}(g)}$$

- estimateur .632+

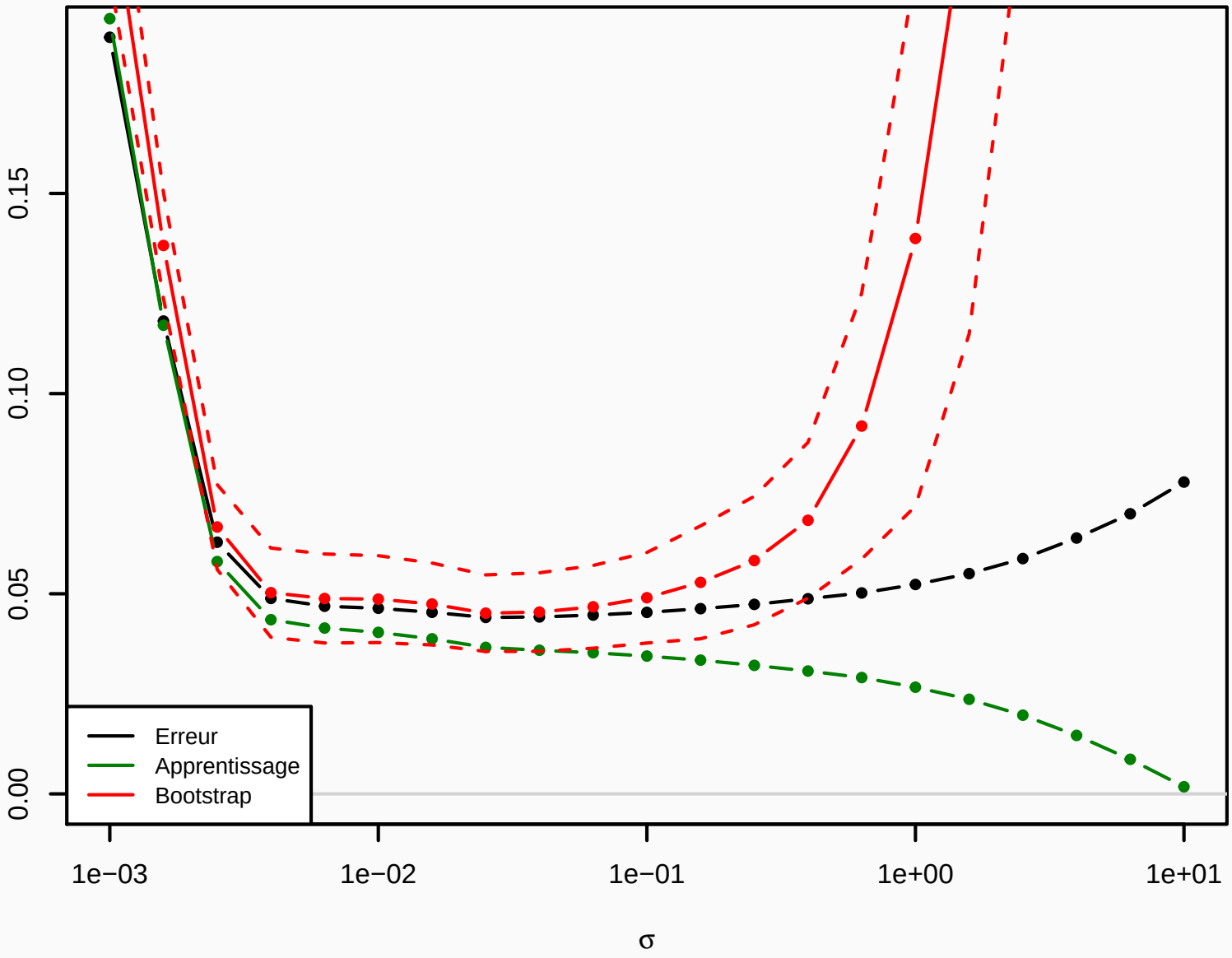
$$\hat{L}_{.632+}(g) = \frac{0{,}368 \times (1 - \hat{R})\hat{L}(g) + 0{,}632 \times \hat{L}_{loob}(g)}{1 - 0{,}368 \times \hat{R}}$$

Loo Bootstrap

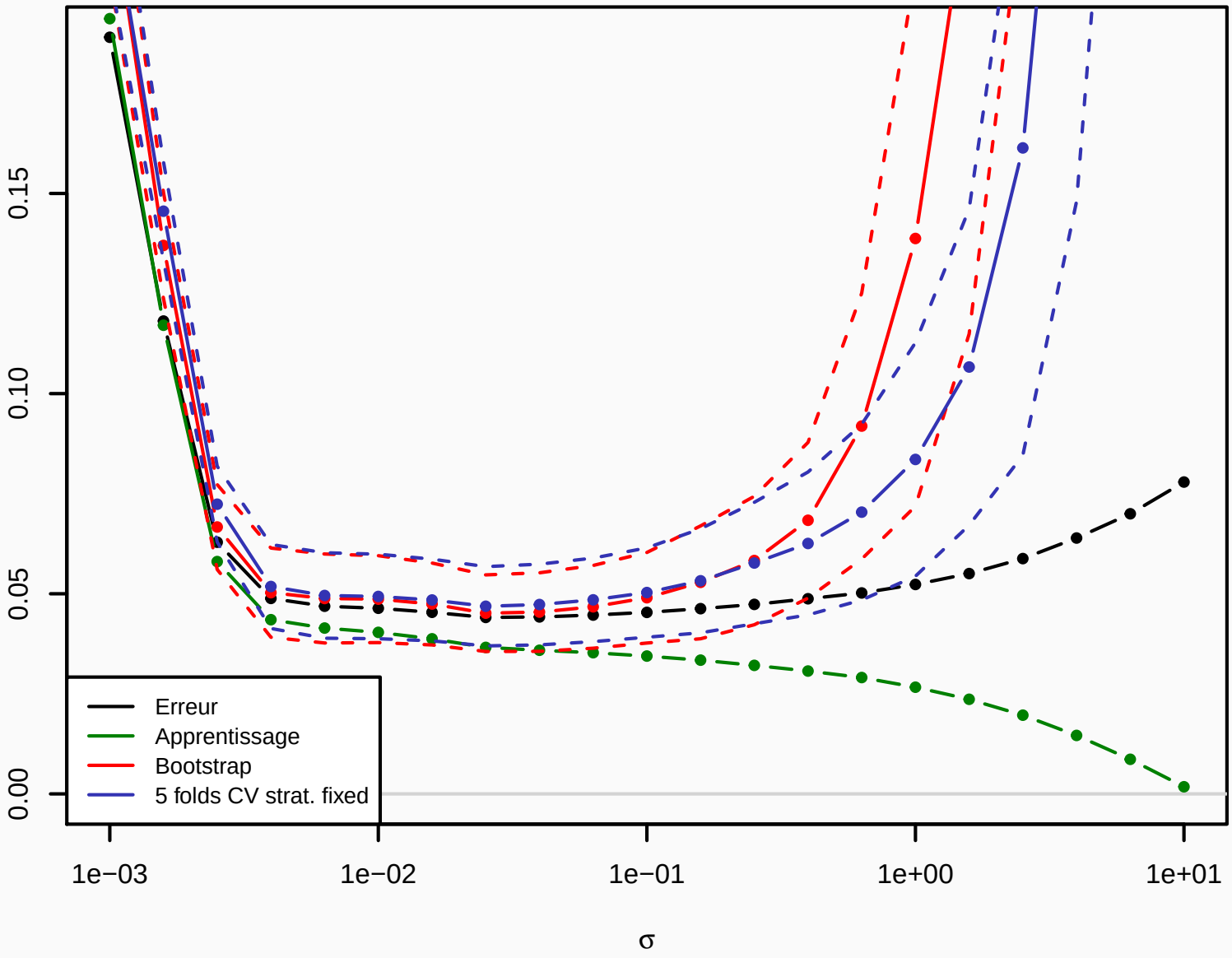
Exemple

- toujours les mêmes données
- échantillons bootstrap identiques pour toutes les données
- versions testées :
  - correction de biais
  - leave-one-out
  - .632
- λ fixé
- $B = 100$ (relativement petite valeur)

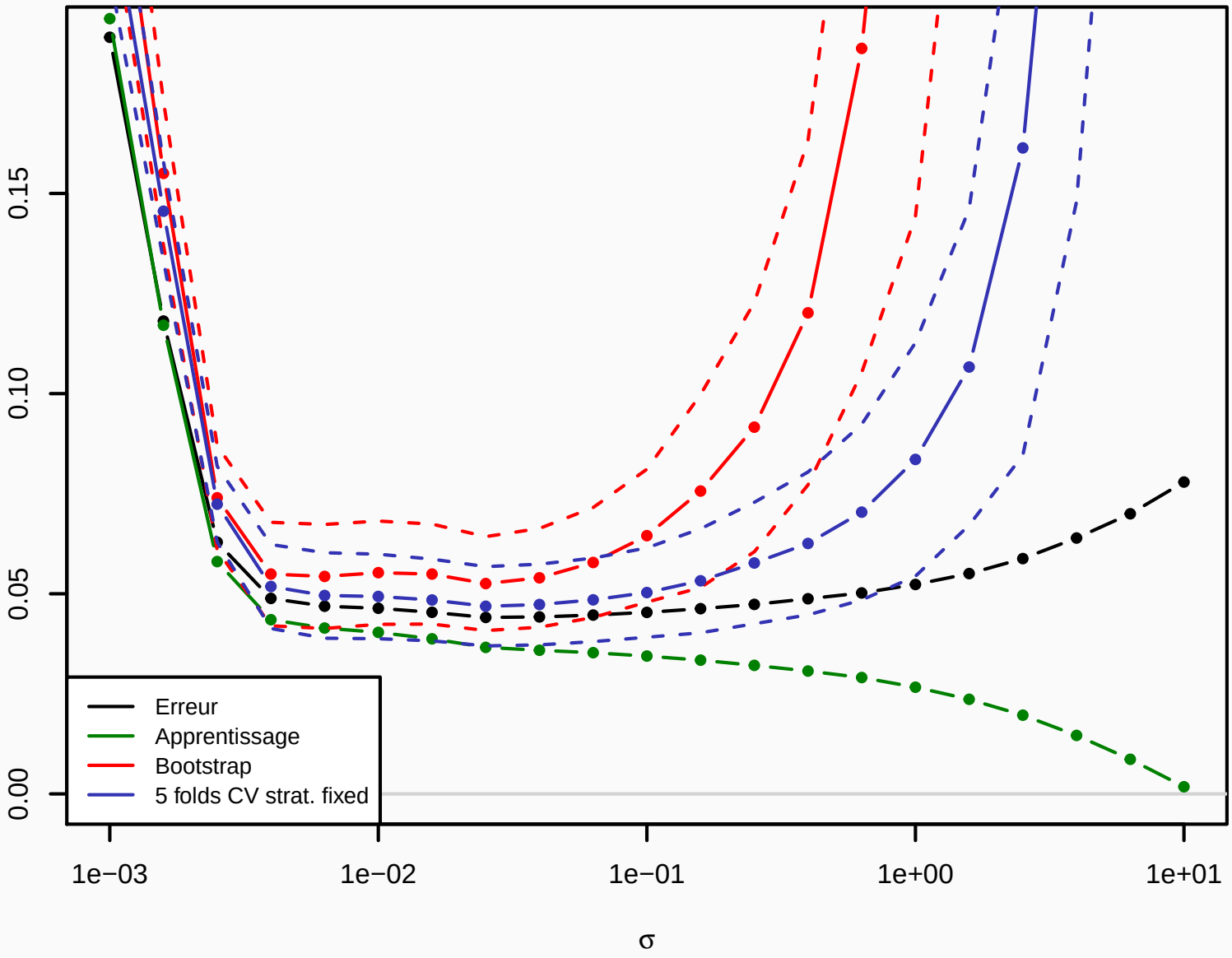
Estimation de biais



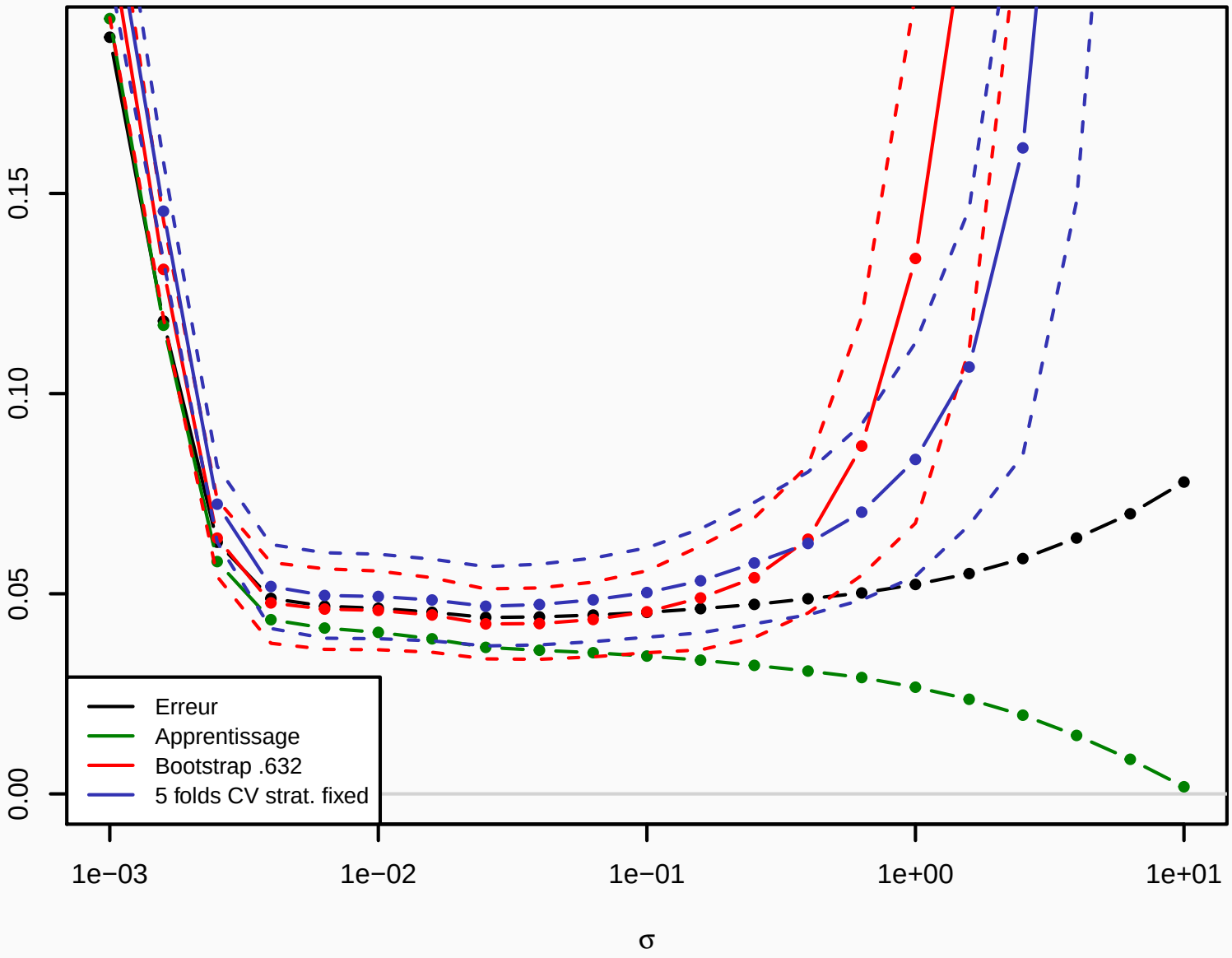
Estimation de biais



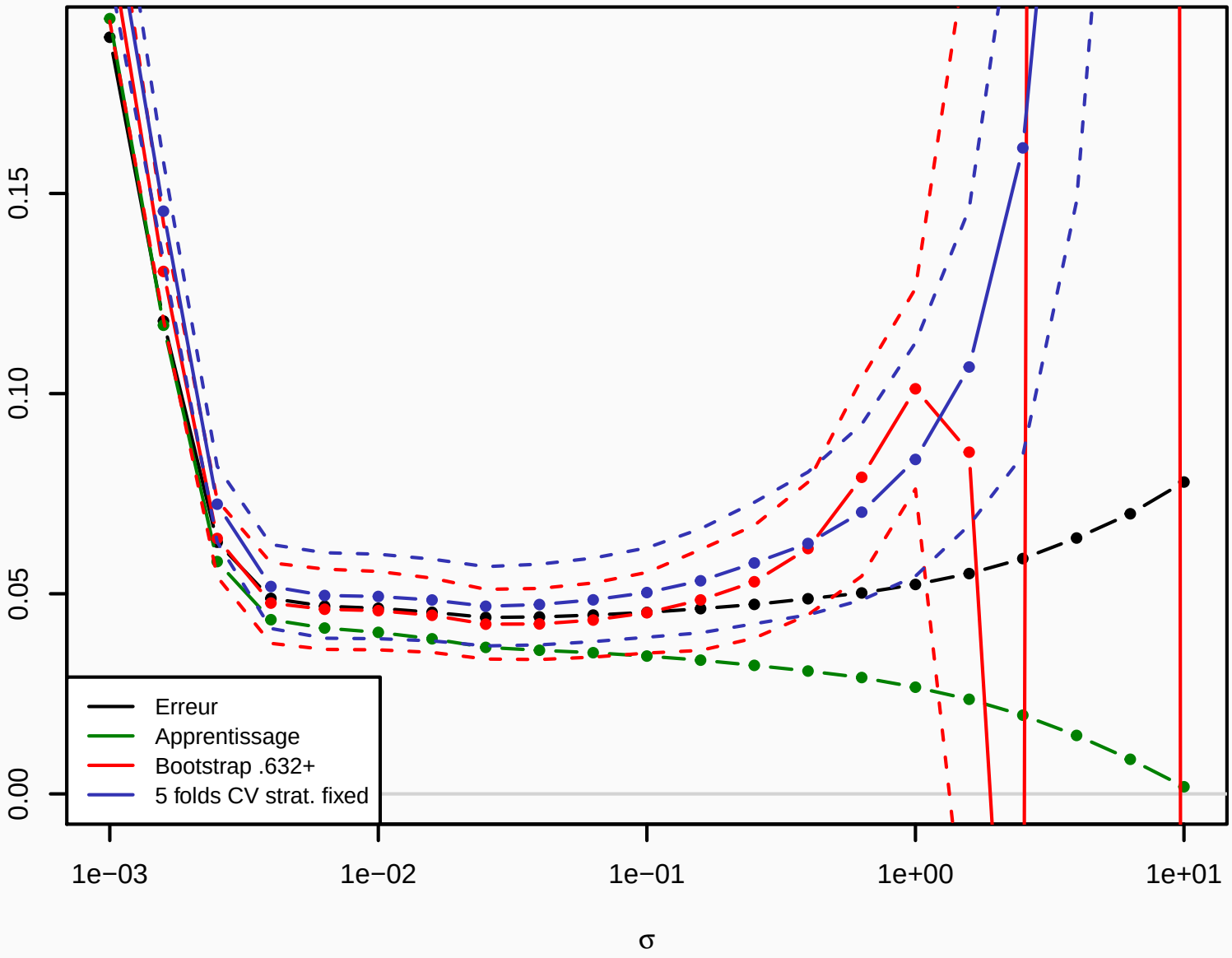
Loo Bootstrap



Bootstrap .632



Bootstrap .632+



Bootstrap en pratique

En statistiques

- très nombreuses variantes
  - version paramétrique
  - version « régularisée »
  - etc.
- en général éloignées des considérations de l'apprentissage

En apprentissage

- bénéficie des mêmes bonnes pratiques que la validation croisée :
  - stratification
  - structuration
  - *bootstrapping* de toute la chaîne d'apprentissage
- généralement très coûteux
- bonne correction du biais avec les versions améliorées (sauf dans des cas extrêmes)

Bootstrap stratifié

Principe

- même idée générale que pour la validation croisée :
  1. découpage $\mathcal{Y}$ en sous-ensembles (naturellement les classes pour $|\mathcal{Y}| < \infty$)
  2. un échantillon bootstrap par sous-ensemble
  3. combinaison des sous-ensembles
- très classique en statistiques
- ne change rien aux procédures décrites jusqu'à présent

Bagging

Principe

- ne pas jeter les modèles g_b construits sur les échantillons bootstrap
- construire un modèle moyen $g = \frac{1}{B} \sum_{b=1}^{B} g_b$
- profiter du biais faible d'un modèle à grand variance
- composant essentiel des forêts aléatoires

Estimation *Out-of-bag*

- O_i : ensemble des $\mathcal{D}^b$ qui ne contiennent pas i
- conduit à un estimateur $\hat{L}_{oob}(g)$

$$\hat{L}_{oob}(g) = \frac{1}{N} \sum_{i=1}^{N} I\left(\frac{1}{|O_i|} \sum_{b \in O_i} g_b(X_i), Y_i \right)$$

Bagging

Bonnes pratiques

- assez robuste au sur-apprentissage (par rapport au nombre d'échantillons)
- peu de paramètres à optimiser dans certains cas
  - il suffit de prendre B « grand » (500 à 5000)
  - pour les forêts aléatoires, on peut éventuellement optimiser le taux de sélection des variables

Stratification

- indispensable pour Y discret
- à intégrer dans le *bootstrap*
- n'est pas mise en œuvre par défaut dans certains packages (par exemple randomForest)
- cf la suite du cours

Plan

Introduction

Compromis biais variance

Leave-one-out

Validation croisée

Bootstrap

Données structurées

Données structurées

Hypothèse fondamentale

- les données sont indépendantes et identiquement distribuées
- le ré-échantillonnage s'appuie largement sur cette hypothèse
- qui est fausse dans certains situations :
  - données temporelles
  - données spatiales
  - plus généralement : données structurées
- cœur du problème pour la validation croisée : la dépendance entre les blocs induit une sous-estimation des erreurs
- pour l'estimation de type loo bootstrap

Illustration

Données temporelles

- modélisation auto-régressive $Y_i \simeq f(Y_{i-k}, Y_{i-k+1}, \ldots, Y_{i-1})$
- mauvaise idée :
  - considérer les couples (X_i, Y_i) avec $X_i = (Y_{i-k}, Y_{i-k+1}, \ldots, Y_{i-1})$
  - utiliser une validation croisée classique : l'algorithme voit le futur !

Solutions ad hoc

- mécanisme *rolling* :
  - on apprend sur $Y_1, \ldots, Y_{i-1}$ et on évalue sur Y_i
  - en faisant varier i de $k+1$ à N
- variantes possibles :
  - prévisions à horizon supérieur à 1
  - origine mouvante (ne pas partir de 1 mais de $i - T$)

Validation structurée

Solution générale (Roberts et al 2016 [RBC+17])

- objectif : « garantir » l'indépendance entre les blocs
- solution : validation croisée structurée (*block cross-validation*)

Structures

- idée de base : découper les données en structures *indépendantes* (*block* par opposition aux *folds*)
- construire les blocs aléatoirement à partir des structures
- difficulté : déterminer les structures

Cas temporel

- il suffit de prendre des observations contiguës temporellement
- on peut éventuellement supprimer les observations qui induisent des superpositions

Exemple

- modélisation auto-régressive d'ordre k
- données mise sous la forme $X_i = (Y_{i-k}, Y_{i-k+1}, \ldots, Y_{i-1})$
- structures de la forme $S_p = ((X_p, Y_p), \ldots, (X_{p+l}, Y_{p+l}))$
- on garde un écart de $l + k + 1$ entre les structures :
  - S_p puis $S_{p+l+k+1}$
  - cela revient à découper la série en tranches de la forme Y_{p_k} à Y_{p+l} et à supprimer les superpositions
- classiquement blocs = structures

Bonnes pratiques

État de l'art

- les publications ne sont pas très concluantes sur le sujet :
  - exemples connus où une validation croisée naïve ne fonctionne pas
  - mais aussi des études dans lesquelles la situation est moins claire
- le cas temporel est le plus étudié, mais les autres structures sont importantes aussi (effets géographiques, notamment)

Que retenir ?

- pour les données temporelles, il semble que même une VC naïve soit plus efficace que l'approche *rolling*
- compromis délicat entre la gestion des dépendances et la bonne utilisation des données
- éviter le super naïf : ne jamais évaluer sur des données déjà vues !

Plan

Introduction

Compromis biais variance

Leave-one-out

Validation croisée

Bootstrap

Données structurées

Que faut-il valider/bootstrapper ?

Processus d'apprentissage

- pré-traitement (normalisation, ACP, etc.)
- équilibrage des données (cf le cours spécifique)
- sélection de variables
- choix des méta-paramètres (ou du modèle)

À valider/boostrapper

- Tout !
- plus sérieusement :
  - la question se pose surtout pour les pré-traitements
  - tout le reste a une influence très importante sur le résultat final
  - on risque le sur-apprentissage et/ou une sous-estimation des erreurs futures en ne validant pas certains aspects
- principe : la validation croisée doit reproduire l'intégralité du processus d'apprentissage

Validation à deux niveaux

Sur-apprentissage de « second ordre »

Dans certaines situations, le processus d'apprentissage est trop complexe pour que les estimations de performances utilisées pour l'ajuster restent valides après cet ajustement

Validation croisée à deux niveaux

1. découper aléatoirement les données en K blocs $C_1, \ldots, C_K$
2. pour e allant de 1 à K
   - 2.1 pour i dans $\{1, \ldots, K\} \setminus \{e\}$
     - 2.1.1 apprendre les modèles étudiés sur les C_j avec $j \neq e$ et $j \neq i$
     - 2.1.2 calculer les prévisions sur des modèles sur le bloc C_i
   - 2.2 calculer $\hat{L}_{cv}^{-e}(g)$ pour chaque modèle g
   - 2.3 apprendre le meilleur modèle au sens de $\hat{L}_{cv}^{-e}(g)$ sur les C_j avec $j \neq e$
   - 2.4 calculer les prévisions du modèle sur C_e
3. calculer $\hat{L}_{cv}(g)$
4. appliquer une validation croisée classique (sur les mêmes blocs) pour obtenir le modèle final

Conclusion

Message principal

Toujours utiliser une méthode valide d'estimation du risque d'un modèle

Mais aussi...

- valider/boostrapper l'intégralité de la chaîne d'apprentissage
- attention aux dépendances cachées entre blocs : structurer le re-échantillonnage
- stratifier et équilibrer
- tout ceci coûte cher !

Bibliographie I

B. Efron.

Bootstrap methods : Another look at the jackknife.
Ann. Statist., 7(1) :1–26, 01 1979.

Bradley Efron.

Estimating the error rate of a prediction rule : Improvement on cross-validation.
Journal of the American Statistical Association, 78(382) :316–331, 1983.

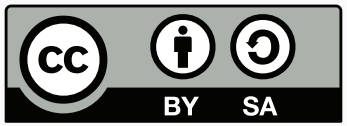
Bradley Efron and Robert Tibshirani.

Improvements on cross-validation : The .632+ bootstrap method.
Journal of the American Statistical Association, 92(438) :548–560, 1997.

David R. Roberts, Volker Bahn, Simone Ciuti, Mark S. Boyce, Jane Elith, Gurutzeta Guillera-Arroita, Severin Hauenstein, José J. Lahoz-Monfort, Boris Schröder, Wilfried Thuiller, David I. Warton, Brendan A. Wintle, Florian Hartig, and Carsten F. Dormann.

Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure.
Ecography, 2017.

Licence



Version

Dernier commit git : 2018-09-05
Auteur : Fabrice Rossi (Fabrice.Rossi@apiacoa.org)
Hash git : 1726ab8d06f33e54d8b61a3c473b2724ef18d86c