Package 'marqLevAlg' - Algorithme de Marquardt-Levenberg en R :

Une alternative à 'optimx' pour des problèmes de minimisation

Mélanie Prague, Amadou Diakite et Daniel Commenges

ISPED Université Bordeaux Segalen
INSERM U897 Épidémiologie et Biostatistique

Rencontres R Bordeaux - 2/3 Juillet 2012

Problématique et état de l'art

Problème d'optimisation :

$$\underset{\theta \in \mathbb{R}^p}{\operatorname{argmin}} \quad f(\theta)$$

Méthode de Newton-Raphson :

- Approximation quadratique de la surface à optimiser.
- Peu robuste si la surface est loin d'une quadrique.

→ Robustification avec l'algorithme de Marquardt-Levenberg [1]

[1] Marquardt, D.W. (1963). *An algorithm for least-squares estimation of nonlinear parameters.* Journal of the society for Industrial and Applied Mathematics, 11(2), 431-441

Objectif et implémentation

Fonctions déjà disponibles dans R (*optimx*) [2]

- Agrégation d'outils d'optimisation
- "Rcgmin", "nlm", "nlminb", "spg", "CG", "bobyqa", "uobyqa" and "newuoa", ...
- **"Nelder-Mead", "BFGS"**

→ Implémenter Marquardt-Levenberg en R avec la même syntaxe.

→ Proposer un nouveau critère de convergence.

[2] Nash J.C. and Varadhan R. (2011). *Unifying Optimization Algorithms to Aid Software System Users : optimx for R.* Journal of Statistical Software, 49(3), 1-14

Robustification (1)

Newton-Raphson classique :

$$\begin{aligned} U &: \text{Gradient en } \theta \\ H &: \text{Hessienne en } \theta \\ \delta &= -\underbrace{H^{-1}}_{Existence(?)} U \end{aligned}$$

La matrice H n'est pas toujours définie positive :

$\rightarrow$ Amortissement : "gonfler" la diagonale.
$\rightarrow$ Adaptatif jusqu'à ce que H soit inversible.

$$H \simeq \begin{pmatrix} h_{11}+\lambda_1|h_{11}| + \lambda_2 Tr(|H|) & h_{12} & \dots & h_{1p} \\ h_{21} & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ h_{p1} & \dots & \dots & h_{pp}+\lambda_1|h_{pp}| + \lambda_2 Tr(|H|) \end{pmatrix}$$

Robustification (2)

Si l'optimisation globale n'améliore pas la valeur de la fonction on peut penser que δ est surestimé.

Recherche d'un meilleur déplacement :

- Calculer la fonction en $\delta^* = \alpha_1 \delta \quad (\alpha_1 < 1)$
- Calculer la fonction en $\delta^* = \alpha_2 \delta \quad (\alpha_2 = \alpha_1 + \epsilon < 1)$
- Itérativement, faire varier (α_1, α_2) pour trouver le meilleur déplacement $\alpha^* \neq 0$
- Poser $\delta^* = \alpha^* \delta$

L'algorithme peut éventuellement "reculer".

Algorithme de Marquardt-Levenberg

- Valeur initiale $\theta^{(0)}$
- A l'itération k, en $\theta^{(k)}$:
  1. Calcul de la fonction $f(\theta^{(k)})$
  2. Calcul pentes et courbure de $f(\theta^{(k)})$
     - Gradient : $U(\theta^{(k)})$
     - Hessienne : $H(\theta^{(k)})$
  3. Choix du déplacement
     - Newton-Raphson avec amortissement
     - Si $f(\theta^{(k)} + \delta) > f(\theta^{(k)})$:
       Recherche linéaire unidirectionnelle.
  4. Déplacement : $\theta^{(k+1)} = \theta^{(k)} + \delta^*$
  5. Test de Convergence
     - Si OK : optimisation réussie
     - Sinon : Retour en 1
- Compte rendu d'optimisation

Évaluation de la convergence

Stabilité entre $\theta^{(k)}$ et $\theta^{(k+1)}$

$$||\theta^{(k+1)} - \theta^{(k)}|| < \epsilon_a$$
$$|f(\theta^{(k+1)}) - f(\theta^{(k)})| < \epsilon_b$$

Distance relative au minimum en $\theta^{(k+1)}$ (RDM)

$$0 < \mathrm{RDM} = \frac{(\mathrm{U}(\theta^{(\mathrm{k+1})}))^{\mathrm{T}}\mathrm{H}^{-1}(\theta^{(\mathrm{k+1})})\mathrm{U}(\theta^{(\mathrm{k+1})})}{\mathrm{m}} < \epsilon_{\mathrm{d}}$$

m : Nombre de paramètres estimés

Le package "marqLevAlg"

Arguments obligatoires

b ou **m**	Valeurs initiales ou taille du vecteur
fn	Fonction à maximiser : f

Arguments facultatifs

gr	Fonction Gradient de f
hess	Fonction Hessienne de f

Arguments de convergence

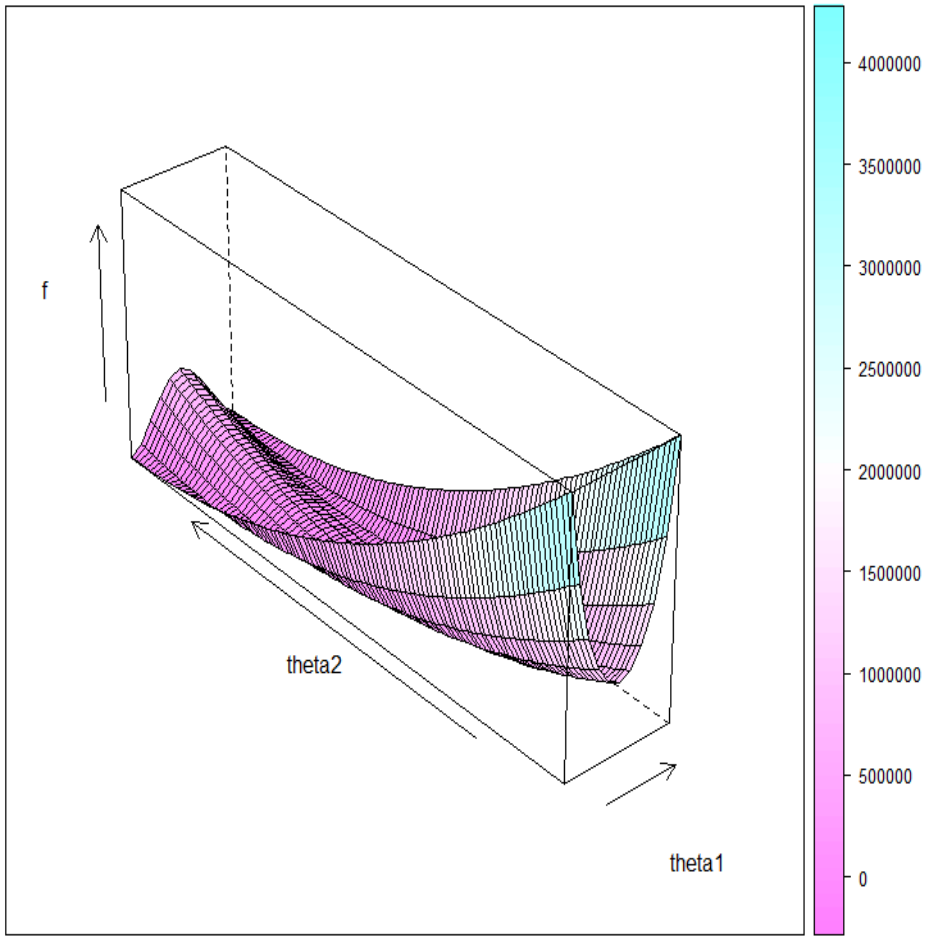
epbsa	Seuil de convergence pour $\|\|\theta^{(k+1)} - \theta^{(k)}\|\|$
epbsb	Seuil de convergence pour $\|f(\theta^{(k+1)}) - f(\theta^{(k)})\|$
epbsd	Seuil de convergence pour le RDM
maxiter	Nombre maximum d'itérations

Arguments d'information

digits	Nombre de chiffres après la virgule dans les sorties
print.info	Impression d'un résumé à chaque itération

Rosenbrock Banana function : Un exemple jouet

$$f(\theta_1, \theta_2) = 100 * (\theta_2 - \theta_1^2)^2 + (1 - \theta_1)^2$$



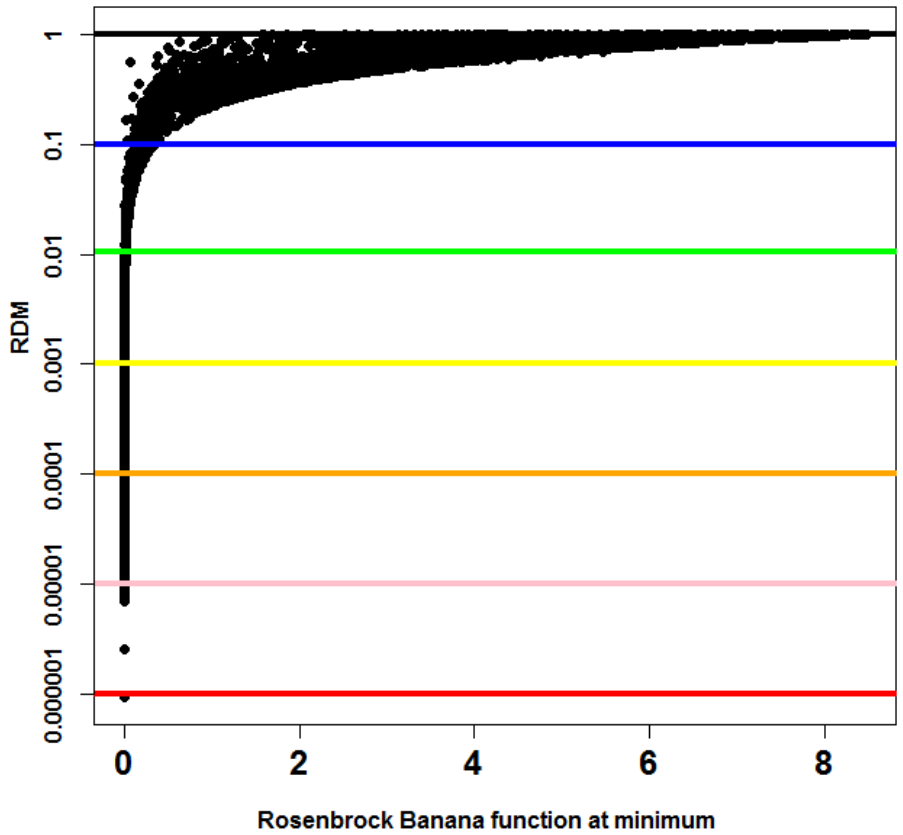
Sorties "marqLevAlg" :

- Statut de convergence,
- Nombre d'itérations,
- Estimations et variances,
- Valeur de f,
- critères de convergence.

Le critère de convergence principal (RDM)

Simulations :

- 15000 points de départ [$\theta_1 \sim \mathcal{N}(-1.2, 10.0)$ et $\theta_2 \sim \mathcal{N}(1, 10.0)$]
- 3000*5 optimisations avec rdm=1.0 ; 0.5 ; 0.1 ; 0.01 ; 0.001
- Pas de contrainte sur *epsa* et *epsb*



Comparaison moyenne des deux algorithmes (1)

Simulations :

- 5000 [$\theta_1 \sim \mathcal{N}(-1.2, 10.0)$ et $\theta_2 \sim \mathcal{N}(1, 10.0)$]
- Optimisation "par défaut"

	Options "par défaut"	
Critère	**optimx**	**marqLevAlg**
	$abstol = 10^{-8}$	$epsa = 0.001$
	$reltol = 10^{-8}$	$epsb = 0.001$
		$epsd = 0.001$
Méthode	14% N-M	-
	86% BFGS	-
Temps	0.001 sec.	0.07 sec.
% Réussite	93.7%	100%
f_{min}	0.55	3e-06
variance θ_1	0.54	3e-06
variance θ_2	8.6	1e-05

Comparaison moyenne des deux algorithmes (2)

Simulations :

- 5000 [$x_1 \sim \mathcal{N}(-1.2, 10.0)$ et $x_2 \sim \mathcal{N}(1, 10.0)$]
- Optimisation "boostée"

	Options "boostée"	
Critère	**optimx**	**marqLevAlg**
	$abstol = 10^{-16}$	$epsa = 10^{-16}$
	$reltol = 10^{-16}$	$epsb = 10^{-16}$
		$epsd = 10^{-16}$
Méthode	99% N-M	-
	1% BFGS	-
Temps	0.002 sec.	0.07 sec.
% Réussite	100%	100%
f_{min}	1e-10	4e-12
variance θ_1	7e-11	2e-25
variance θ_2	3e-10	9e-25

Conclusion

Avantages

- Modification des critères de convergence facile.
- Interprétabilité des critères de convergence.
- Récupération des variances des estimations facile.
- Convergence améliorée pour des points de départ éloignés.

Inconvénients

- Temps de calculs un peu supérieur à *optimx*.