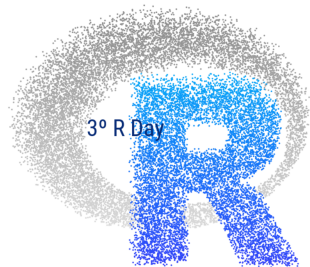


Implementando modelos estatísticos de maneira eficiente com o TMB

Um tutorial

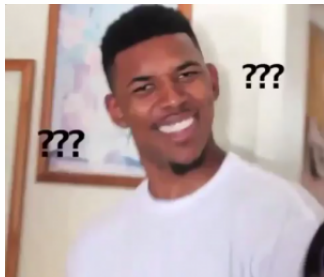


Henrique Laureano, Ricardo Petterle & Wagner Bonat

LEG @ UFPR

9 de setembro, 2021

TMB: Template Model Builder



O quê?



Por quê?



Como?

O que é o Template Model Builder?



Pelas palavras dos autores:



Kristensen et al. (2016).

Um pacote R (R Core Team 2021) para a rápida implementação de complexos modelos de efeitos aleatórios através de simples templates C++.

Complexos modelos de efeitos aleatórios? Do simples ao complicado.

De modelos simples como um

- um modelo linear (**LM**) ou
- um modelo linear generalizado (**GLM**),

até

- **modelos não-lineares com efeitos aleatórios e**
- complexos **modelos espaço-temporais.**

Inúmeras possibilidades...



- 1 Estudar o efeito de características numa certa variável? **Modelos lineares** (LM);
- 2 A resposta é não-Normal/Gaussiana? **Modelos lineares generalizados** (GLM);
- 3 Função não-linear nos parâmetros? **Modelos não-lineares**;
- 4 Múltiplas respostas/variáveis? **Modelos multivariados**;
- 5 Presença de dependência não-observada/latente?
Modelos de efeito aleatório/latente/misto.
 - 1 Modelos para **dados longitudinais** (medidas repetidas, séries temporais);
 - 2 Modelos para **dados espaciais** e **espaço-temporais**;
- 6 ...

O TMB possibilita o ajuste de todos esses modelos.

Características-chave:

- 1 Diferenciação automática;
o estado-da-arte na computação de derivadas
- 2 Aproximação de Laplace.
Uma maneira eficiente de aproximar as integrais do efeito aleatório

Um pouco de matemática para justificar as coisas...

Considere que $f(u, \theta)$ seja o

negativo da sua função de log-verossilhança **conjunta nll**),

em que $u \in \mathbb{R}^n$ são os **efeitos aleatórios desconhecidos** e $\theta \in \mathbb{R}^m$ são os **parâmetros**.

Conjunta? Num modelo estatístico especificamos uma distribuição de probabilidade para o que observamos (**dados**) e outra para o que não observamos (**efeito aleatório**). **E é aí que mora o problema!**

TMB: Lidando com efeitos aleatórios



Paradigmas: Verossimilhancista e Bayesiano.

Bayesiano: Atribuição de distribuições *à priori* para os parâmetros, que passam a serem vistos como variáveis. Não mais estimamos os "parâmetros", e sim amostramos de sua distribuição *a posteriori*. Funciona, mas é **computacionalmente intensivo**.

Verossimilhancista: Temos um problema, já que o efeito aleatório é **não observável**. Contudo, da estatística básica: se temos uma **conjunta**, basta **integrarmos** na variável que não queremos mais. Resultando numa

Função de verossimilhança **marginal** :

$$L(\theta) = \int_{\mathbb{R}^n} \exp(-f(u, \theta)) \, du.$$

Bem, essa é a ideia básica. Na prática não é bem assim...

Quando a distribuição que especificamos pro **dado** não é Gaussiana, não conseguimos resolver aquela integral analiticamente.

Aí que entra a **aproximação de Laplace** : $L^*(\theta) = \sqrt{2\pi}^n \det(H(\theta))^{-1/2} \exp(-f(\hat{u}, \theta))$, com

- $H(\theta) = f''_{uu}(\hat{u}(\theta), \theta)$ sendo o Hessiano de $f(u, \theta)$ w.r.t. u e avaliado em $\hat{u}(\theta)$;
- $\hat{u}(\theta) = \arg \min_u f(u, \theta)$ sendo o minimizador de $f(u, \theta)$ w.r.t. u .

Finalmente,

a função objetivo final a ser minimizada em termos de θ é

$$-\log L^*(\theta) = -n \log \sqrt{2\pi} + \frac{1}{2} \log \det(H(\theta)) + f(\hat{u}, \theta),$$

i.e., o **log negativo da aproximação de Laplace**.

Uncertainty quantification: Método- δ

$$\text{VAR}(\phi(\hat{\theta})) = -\phi'_{\theta}(\hat{\theta})(\nabla^2 \log L^*(\hat{\theta}))^{-1} \phi'_{\theta}(\hat{\theta})^{\top},$$

ou seja,

conseguimos quantificar a incerteza da estimativa $\hat{\theta}$, e de qualquer função diferenciável da estimativa $\phi(\hat{\theta})$.

Em todas as etapas

- Aproximação de Laplace (*otimização interna*);
- Log negativo da aproximação de Laplace (*otimização externa*);
- Quantificação da incerteza,

o cálculo de derivadas (1a e 2a ordem) é essencial.

Portanto, sermos capazes de usar a **maneira mais eficiente existente hoje** pra calcular essas derivadas, é uma tremenda **qualidade**.

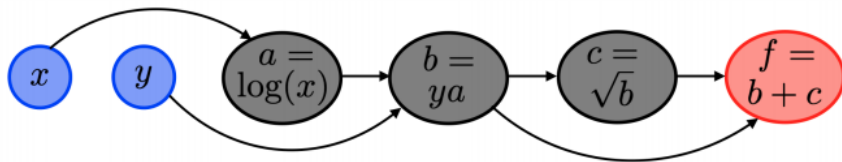
Diferenciação automática (AD) em um exemplo



Considere a função (📖 exemplo tirado de Peyré (2020, página 33))

$$f(x, y) = y \log(x) + \sqrt{y \log(x)}$$

O que a **AD** internamente faz é decompor a função em *nodos*, construir um **grafo**



e trabalhar em cima do mesmo.

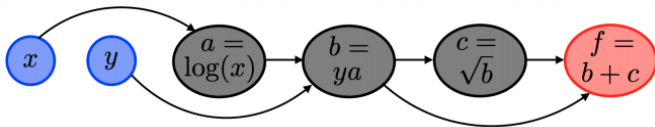
Existem dois **modos** de avaliar a função via **AD**. O modo **forward** e o modo **reverse**. O que o **TMB** faz é a **reverse**, **computacionalmente mais eficiente**.

AD: modo reverso



Função:

$$f(x, y) = y \log(x) + \sqrt{y \log(x)}$$



$$\frac{\partial f}{\partial f} = 1$$

$$\frac{\partial f}{\partial c} = \frac{\partial f}{\partial f} \frac{\partial f}{\partial c} = \frac{\partial f}{\partial f} 1$$

$$\{c \mapsto f = b + c\}$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial c} \frac{\partial c}{\partial b} + \frac{\partial f}{\partial f} \frac{\partial f}{\partial b} = \frac{\partial f}{\partial c} \frac{1}{2\sqrt{b}} + \frac{\partial f}{\partial f} 1$$

$$\{b \mapsto c = \sqrt{b}, b \mapsto f = b + c\}$$

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial b} \frac{\partial b}{\partial a} = \frac{\partial f}{\partial b} y$$

$$\{a \mapsto b = ya\}$$

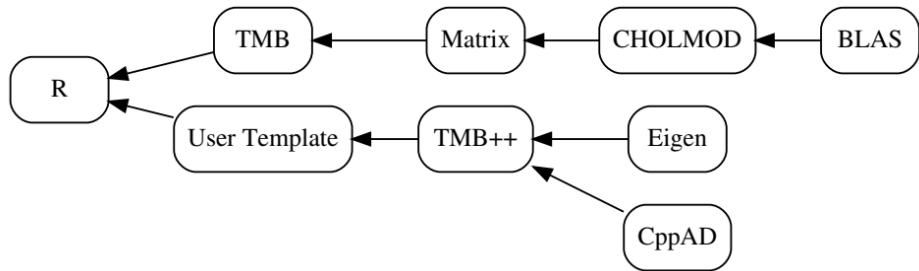
$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial b} \frac{\partial b}{\partial y} = \frac{\partial f}{\partial b} a$$

$$\{y \mapsto b = ya\}$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial a} \frac{\partial a}{\partial x} = \frac{\partial f}{\partial a} \frac{1}{x}$$

$$\{x \mapsto a = \log(x)\}$$

TMB package design



Combinação de alguns softwares *estado-da-arte*:

- CppAD, um pacote C++ para AD <https://coin-or.github.io/CppAD/>;
- Bibliotecas de álgebra como a Eigen (Guennebaud, Jacob, and others (2010), em C++), a CHOLMOD (<https://developer.nvidia.com/cholmod>, em C) e a Matrix (Bates and Maechler (2019), usando a LAPACK <http://www.netlib.org/lapack/> e a SuiteSparse <https://sparse.tamu.edu/>);
- Paralelismo através da BLAS (<http://www.netlib.org/blas/>, em Fortran).

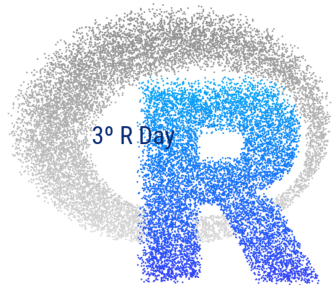
Como usar?

Workflow

- 1 Escreva sua função objetivo em um .cpp com `#include <TMB.hpp>;`
- 2 Compile e carregue seu arquivo .cpp em R via `TMB::compile()` e `base::dyn.load(TMB::dynlib());`
- 3 Compute as derivadas da sua função objetivo com `obj <- TMB::MakeADFun();`
- 4 Faça o ajuste do modelo, `opt <- base::nlminb(objpar, objfn, obj$gr);`
- 5 Quantifique a incerteza dos parâmetros, `TMB::sdreport(obj)`.

(Extra) Funcionalidades

- Paralelização;
- Simulação;
- Esparsidade;
- Perfis de verossimilhança;
- ...



TMB: Template Model Builder, Exemplos



Lista online dos usuários de TMB: <https://groups.google.com/g/tmb-users>;



Tutorial online: https://kaskr.github.io/adcomp/_book/Tutorial.html;



Para mais detalhes sobre TMB, AD e aproximação de Laplace: Laureano (2021).



Tutorial completo disponível em <https://github.com/henriquelaureano>



Bates, D., and M. Maechler. 2019. *Matrix: Sparse and Dense Matrix Classes and Methods*. R Foundation for Statistical Computing. Vienna, Austria.

Guennebaud, G., B. Jacob, and others. 2010. “Eigen V3.” <http://eigen.tuxfamily.org>.

Kristensen, K., A. Nielsen, C. W. Berg, H. J. Skaug, and B. M. Bell. 2016. “TMB: Automatic Differentiation and Laplace Approximation.” *Journal of Statistical Software* 70 (5): 1–21.

Laureano, H. A. 2021. “Modeling the Cumulative Incidence Function of Clustered Competing Risks Data: A Multinomial Glmm Approach.” Master’s thesis, Federal University of Paraná (UFPR).

Peyré, G. 2020. “Course Notes on Optimization for Machine Learning.” May 10, <https://mathematical-tours.github.io/book-sources/optim-ml/OptimML.pdf>.

R Core Team. 2021. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria.