Computational Statistics

A short introduction to Template Model Builder (TMB)

Joaquin Cavieres

Georg-August-Universität Göttingen

Content

UNIVERSITÄT GÖTTINGEN

1. Introduction

Normally, we are interested in proposing a statistical model to explain certain phenomena. To achieve this, we need to be clear about the following:

- Propose a methodology to solve the problem.
- Implementation of the statistical model.
- Evaluation of the results.

To propose the methodology we need:

- Consider all the characteristics of the problem
- Evaluate the best methodology to approximate the solution

For the implementation of the statistical model we have to:

- Building models based on the characteristics already analyzed
- Propose a Likelihood function for the response variable

To evaluate the results of the model:

- Consider the Likelihood function used for fit the model
- Diagnostics
- Compute the uncertainty of the estimations

How we can do that?

Template Model Builder (TMB)

How we can do that?

Template Model Builder (TMB)

What is TMB?

- Template Model Builder (TMB) is an open source R package that enables quick implementation of complex nonlinear random effects (Kristensen et al. (2015))
- TMB is a frequentist statistical platform that works with Automatic differentiation (AD) to obtain the first and second derivatives of a function.
- It enables easy parallel computations, with users defining joint likelihoods in C++ while other operations are managed in R.

What is TMB?

- Template Model Builder (TMB) is an open source R package that enables quick implementation of complex nonlinear random effects (Kristensen et al. (2015))
- TMB is a frequentist statistical platform that works with Automatic differentiation (AD) to obtain the first and second derivatives of a function.
- It enables easy parallel computations, with users defining joint likelihoods in C++ while other operations are managed in R.

Main features of TMB:

- TMB is designed for complex hierarchical models
- It uses maximum marginal likelihood
- Compatible with Stan to make Bayesian inference

Main features of TMB:

- TMB is designed for complex hierarchical models
- It uses maximum marginal likelihood
- Compatible with Stan to make Bayesian inference

Why use TMB:

- The objective function (and its derivatives) can be called from R, so parameter optimization can be done via e.g. `nlminb()`.
- The user can specify the use of Laplace approximation to obtain the marginal likelihood of the latent variables (random effects)
- Compute the standard deviations of any parameter, or derived parameters by the Delta method.

Why use TMB:

- The objective function (and its derivatives) can be called from R, so parameter optimization can be done via e.g. `nlminb()`.
- The user can specify the use of Laplace approximation to obtain the marginal likelihood of the latent variables (random effects)
- Compute the standard deviations of any parameter, or derived parameters by the Delta method.

Installing TMB:

1. From Github: `https://github.com/kaskr/adcomp`.
   - git clone https://github.com/kaskr/adcomp

2. From R:

```
install.packages('TMB')
library(TMB)
#test to know if TMB is correctly installed:
runExample(all=TRUE)
```

Note: Maybe you need to install Rtools as well

- `https://cran.r-project.org/bin/windows/Rtools/`

Restart R after installing Rtools and run: library(devtools)

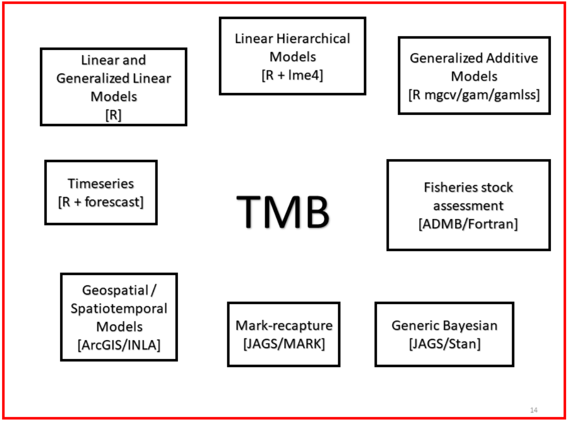

Figure 1: Summary of models which can be implemented in TMB (by Cole. C Monnahan) (by Cole. C Monnahan)

TMB workflow

- Propose a statistical model
- Write a C++ template within R to calculate the negative log-likelihood given parameters
- Compile the model and load it into R
- Declare which parameters are “ random"
- Fit the model using R and minimize the objective function returned by TMB
- Perform inference using the fitted model

2. TMB for modelling

Steps

- Compile the C++ file (template)
- Load the model in R (`dyn.load`)
- Build a TMB object (`MakeADFun`) function based on a list of data and parameters
- The returned object is a list containing various elements, including objective function (`fn`) and its gradient (`gr`)

TMB sections

The model is written in C++, and the structure of the TMB model follows this format

- Read the data from R
- Set the parameters
- Compute:
  1. Model expectation (given the parameters).
  2. Negative log-likelihood (NLL)
- Report the results back to R
- Return the NLL

DATA section

Importing the data from R

TMB syntax	C++ syntax	R syntax
`DATA_VECTOR(x)`	`vector<Type>`	`vector()`
`DATA_MATRIX(x)`	`matrix<Type>`	`matrix()`
`DATA_SCALAR(x)`	`Type`	`numeric()`
`DATA_INTEGER(x)`	`int`	`integer()`
`DATA_FACTOR(x)`	`vector<int>`	`factor()`
`DATA_ARRAY(x)`	`array<Type>`	`array()`
`DATA_SPARSE_MATRIX(x)`	`Eigen::SparseMatrix<Type>`	`dgTMatrix()`

PARAMETER section

TMB syntax	C++ syntax	R syntax
PARAMETER_MATRIX(x)	matrix<Type>	matrix()
PARAMETER_VECTOR(x)	vector<Type>	vector()
PARAMETER_ARRAY(x)	array<Type>	array()
PARAMETER(x)	Type	numeric()

REPORT section

Reporting objects back to R

- Return objects via `REPORT()`, for example:

```
REPORT(predict);
```

- In R you must do

```
obj$report()
```

- Report parameters from the fitted model

```
obj$report(par)
```

Calculating the -log-likelihood

- Calculate likelihood function using e.g,. `dnorm()`

```
nll= -dnorm(y(i), mu(i), sigma, true);
```

- For class type vectors, you need to do

```
nll= -dnorm(y, mu, sigma, true).sum();
```

- In the last line of the C++ template, you must write `return nll;`
- The `nll` must be a scalar Type variable

Calculating the variances

TMB returns the asymptotic variances for the parameters using the function `sdreport(obj)` in R. Also we can calculate the standard errors from the derived quantities such as

```
Type nu = exp(eta);
ADREPORT(nu);
```

3. Some theory behind of TMB

UNIVERSITÄT GÖTTINGEN

Consider the following hierarchical model (mixed effects)

$$\begin{aligned} \mathbf{y} \mid \boldsymbol{\beta}, \mathbf{u} &\sim f(\mathbf{y} \mid \boldsymbol{\beta}, \mathbf{u}) \\ \mathbf{u} &\sim h(\mathbf{u} \mid \vartheta) \end{aligned}$$

We will denote the conditional density of $\mathbf{y}$ given $\mathbf{u}$ as $f(\mathbf{y} \mid \mathbf{u})$, and the marginal density of $\mathbf{u}$ as $h(\mathbf{u})$. Since f and h depends on unknown parameters $\boldsymbol{\theta} = (\boldsymbol{\beta}, \boldsymbol{\vartheta})$, their dependency is expressed as f_{β} and h_{ϑ} respectively.

Hence the joint likelihood is

$$\mathcal{L}(\beta, \vartheta) = f_\beta(\mathbf{y} \mid \mathbf{u}) h_\vartheta(\mathbf{u}) \tag{1}$$
$$\mathcal{L}(\boldsymbol{\theta}) = f_\beta(\mathbf{y} \mid \mathbf{u}) h_\vartheta(\mathbf{u}) \tag{2}$$
$$\mathcal{L}(\boldsymbol{\theta}) = \int f_\beta(\mathbf{y} \mid \mathbf{u}) h_\vartheta(\mathbf{u}) d\mathbf{u} \tag{3}$$

Integrating out the random effects $\mathbf{u}$ from the joint density $f_\beta(\mathbf{y} \mid \mathbf{u}) h_\vartheta(\mathbf{u})$, the marginal likelihood can be written as

$$\mathcal{L}(\boldsymbol{\theta}) = \int \exp\{g(\boldsymbol{u}, \boldsymbol{\theta})\} \tag{4}$$

where $g(\boldsymbol{u}, \boldsymbol{\theta}) = \log\{f_\beta(\mathbf{y} \mid \mathbf{u})\} + \log\{h_\vartheta(\mathbf{u})\}$.

Hence the joint likelihood is

$$\mathcal{L}(\beta, \boldsymbol{\vartheta}) = f_\beta(\mathbf{y} \mid \mathbf{u}) h_\vartheta(\mathbf{u}) \tag{5}$$

$$\mathcal{L}(\boldsymbol{\theta}) = f_\beta(\mathbf{y} \mid \mathbf{u}) h_\vartheta(\mathbf{u}) \tag{6}$$

$$\mathcal{L}(\boldsymbol{\theta}) = \int f_\beta(\mathbf{y} \mid \mathbf{u}) h_\vartheta(\mathbf{u}) d\mathbf{u} \tag{7}$$

Integrating out the random effects $\mathbf{u}$ from the joint density $f_\beta(\mathbf{y} \mid \mathbf{u}) h_\vartheta(\mathbf{u})$, the marginal likelihood can be written as

$$\mathcal{L}(\boldsymbol{\theta}) = \int \exp\{g(\boldsymbol{u}, \boldsymbol{\theta})\} \tag{8}$$

where $g(\boldsymbol{u}, \boldsymbol{\theta}) = \log\{f_\beta(\mathbf{y} \mid \mathbf{u})\} + \log\{h_\vartheta(\mathbf{u})\}$.

Solving this integral presents a challenge and requires intensive computational work aimed at maximizing $\mathcal{L}$. Hence, numerical methods can be employed to approximate the solution, one of which is the Laplace approximation (we will see this within few weeks more..).

See next example in R using TMB...

See you next week!

References I

Kristensen, K., Nielsen, A., Berg, C. W., Skaug, H., and Bell, B. (2015). Tmb: automatic differentiation and laplace approximation. *arXiv preprint arXiv:1509.00660*.