Technical University of Denmark

Exercises for Template Model Builder (TMB)

Christoffer Moesgaard Albertsen

cmoe@aqua.dtu.dk

July 29, 2014

DTU Aqua
National Institute of Aquatic Resources

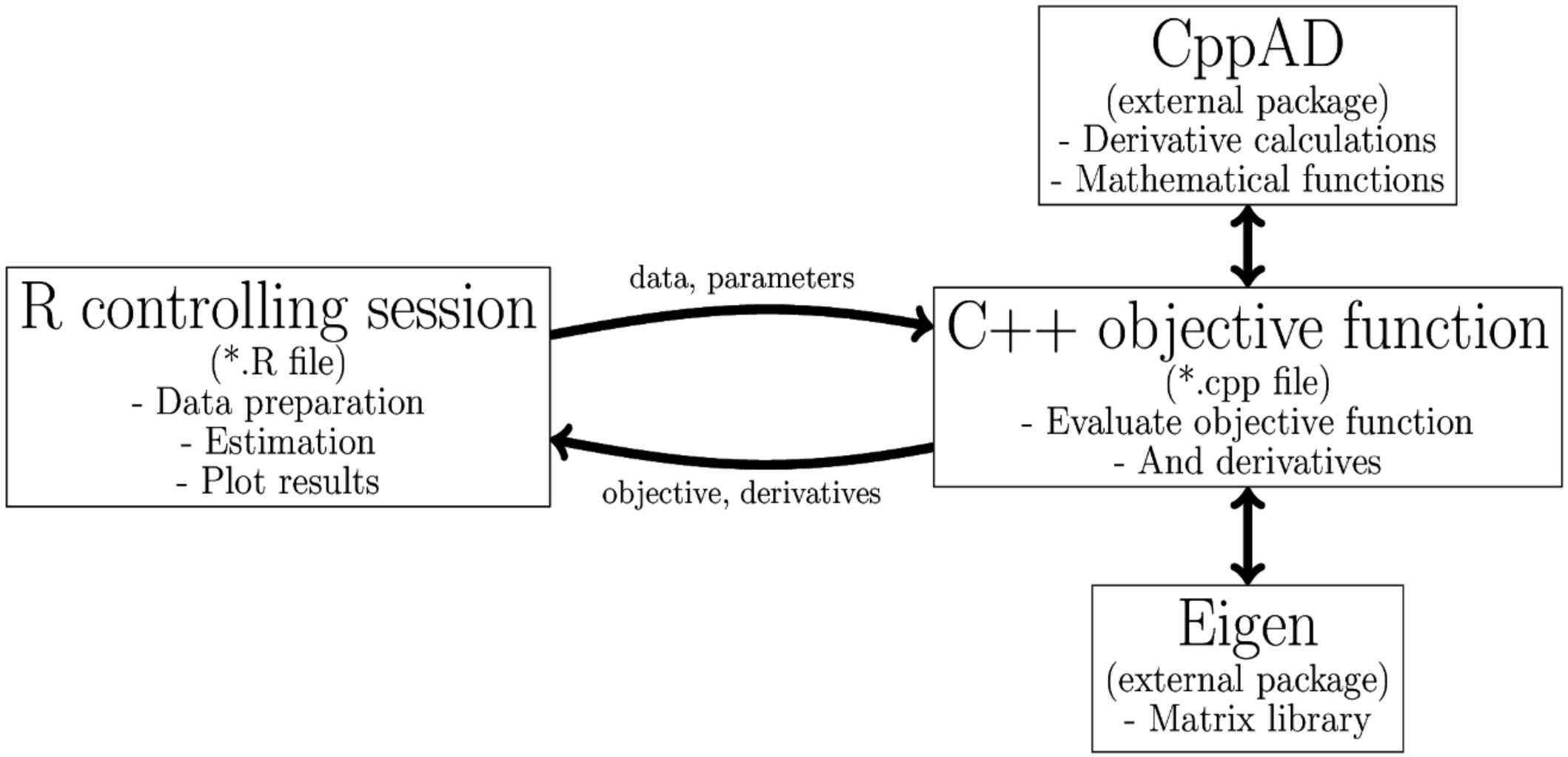
Outline of today

- Recap from yesterday (building models in TMB)
- Exercises in TMB - "regular" models
- OTN session
- Exercises in TMB - state-space models

Outline of today

- Download data files from http://www.staff.dtu.dk/cmoe/tmb-dal-2014
- I will show a slide with questions
- You will have some time to answer them
- We will talk about the solutions
- I will present a new slide with questions
- ...
- Feel free to ask if you have any questions

Building models in TMB



http://folk.uib.no/hsk021/tmbdoc/group__Structure__TMB.html

Models in TMB - C++

- Include TMB specific macros and functions

```
#include <TMB.hpp>
```

- Start the definition of the objective function

```
template<class Type>
Type objective_function<Type>::operator() ()
{
```

Models in TMB - C++

- Define data and parameters

Data	Parameters	R equivalent
DATA_SCALAR	PARAMETER	numeric(1)
DATA_VECTOR	PARAMETER_VECTOR	vector
DATA_MATRIX	PARAMETER_MATRIX	matrix
DATA_ARRAY	PARAMETER_ARRAY	array
DATA_SPARSE_MATRIX	-	dgTMatrix
DATA_INTEGER	-	integer(1)
DATA_FACTOR	-	factor

Models in TMB - C++

- Code the objective function, e.g.,

```
  Type f;
  f = -sum(dnorm(x,mu,sigma,true));
```

- Report calculations back to R

```
REPORT(x-mu);
ADREPORT(pow(sigma,2));
```

- Return the objective function value

```
  return f;
```

- End the function definition

```
}
```

Models in TMB - C++

- You may need to include

```
using namespace tmbutils;
```

See http://folk.uib.no/hsk021/tmbdoc/modules.html for available distributions (or search the GitHub page)

Models in TMB - R

- load the package

```
library(TMB)
```

- Compile the C++ code (when changed)

```
compile("mycode.cpp")
```

- Load the C++ part into R

```
dyn.load(dynlib("mycode"))
```

Models in TMB - R

- Define a data, a parameter, and a map list, e.g. dat, param, map.
    - dat is a list corresponding to the DATA_ variables
    - param is a list corresponding to the PARAMETER_ variables
    - map is a list of factors fixing parameter values

- Create the TMB object

```
obj <- MakeADFun(data=dat,parameters=param,map=map,DLL="mycode")
```

Models in TMB - R

- Use the many functions and variables in the object for inference and calculations

Function/variable	Description
obj$par	Vector of initial parameters to estimate
obj$fn()	Objective function
obj$gr()	Objective gradient
obj$he()	Objective hessian (only fixed effects)
obj$report()	List of values reported by REPORT(...)
objenvparList()	List of all parameters
objenvlast.par	Last evaluated parameters
objenvlast.par.best	Best evaluated parameters
objenvvalue.best	Lowest function value encountered
sdreport(obj)	Estimates and st.dev. for parameters and ADREPORT values

E1 - Rats' weight

1. Fit the following linear model to the data
   - $X_i \sim \mathcal{N}(\alpha + \beta \cdot t_i, \sigma^2)$
2. What is the negative log-likelihood and its gradient for:
   - $X_i \sim \mathcal{N}(160 + 40 \cdot t_i, 7^2)$
   - $X_i \sim \mathcal{N}(130 + 50 \cdot t_i, 9^2)$
3. What are the estimates and standard errors of $\sigma, \log(\sigma)$ and σ^2
4. Plot the regression line with 95% confidence interval
5. Add a 95% prediction interval
6. Plot the residuals
7. Now use a Laplace distribution for the errors
   - $f(x) = \frac{1}{2b} \exp\left(\frac{-|x-\mu|}{b}\right)$
8. Which fits better?

E2 - Count data

1. Fit a Poisson distribution to the data
2. Fit a Poisson distribution with gamma distributed random effects for each observation as the mean
    - $\lambda_i \sim \Gamma(\alpha, \beta)$
    - $X_i \sim Pois(\lambda_i)$,
3. Compare the Gamma-Poisson model to a Negative Binomial model

E3 - Beaver temperature

1. Fit an AR(1) model to the data
   - $X_i \sim \mathcal{N}(c_{a(i)} + \phi \cdot X_{i-1}, \sigma^2)$ for $i > 0, |\phi| < 1$
   - $X_0 \sim \mathcal{N}(c_{a(i)}/(1-\phi), \sigma^2/(1-\phi^2))$
   - $a(i)$ is the activity code (0 or 1) for observation i
2. Configure your code to use conditional maximum likelihood (i.e. assume X_0 is fixed)
3. Compare the results
4. Configure your code to fit an AR(p) model using conditional maximum likelihood
   - $X_i \sim \mathcal{N}(c_a + \phi_1 \cdot X_{i-1} + \cdots + \phi_p \cdot X_{i-p}, \sigma^2)$ for $i > p, |\phi| < 1$
   - Hint: v.segment(i,n) will give you a subset of vector v with length n, starting from i.
5. Which value would you choose for p?

E4 - Random walks

1. Fit a random walk model to the data
   - $X_i \sim \mathcal{N}(X_{i-1}, \sigma^2)$ for $i > 0$
   - $X_0 \sim \mathcal{N}(0, \sigma^2)$
2. Plot the data together with the predicted values.
3. Add a 95% prediction interval
4. Extend your random walk model to include measurement error
   - $Y_i \sim \mathcal{N}(X_i, \nu^2)$
5. Plot the data and the predicted states
6. Add a 95% confidence interval for the states
7. Add a 95% prediction interval

Multivariate normals in TMB

- Multivariate normal with mean zero and user specified covariance
- Need to include

```
using namespace density;
```

- Setup the covariance matrix, e.g.

```
matrix<Type> cov(2,2);
cov << 2, 0, 0, 1;
cov(0,0) = 2; cov(0,1) = 0;
cov(1,0) = 0; cov(1,1) = 1;
```

- Declare the multivariate normal

```
MVNORM_t nll_dens(cov);
```

Multivariate normals in TMB

- Alternatively, declare without covariance matrix

```
MVNORM_t nll_dens;
```

- And set the covariance afterwards

```
nll_dens.setSigma(cov);
```

- Evaluate the negative log-likelihood at x

```
nll += nll_dens(x);
```

E5 - Correlated random walk

1. Fit a two dimensional random walk model to the data
   - $X_i \sim \mathcal{N}_2(X_{i-1}, \Sigma)$ for $i > 0, |\phi| < 1$
   - $X_0 \sim \mathcal{N}_2(0, \Sigma)$
2. What is the estimated correlation between the two dimensions?
3. Extend your random walk model to include measurement error
   - $Y_i \sim \mathcal{N}_2(X_i, \Sigma_Y)$
4. How did this affect your correlation estimates from before?
5. Interpret the difference in an animal movement framework

E6 - Geolocation - step 1

1. Fit a state-space model to the data with (c = latitude,longitude; $\Delta_i = 10$):
   - States: $\nu_c(t_{i+1}) - \gamma_c = e^{-\beta_c \Delta_i}(\nu_c(t_i) - \gamma_c) + \eta_{ci}$
   - $\eta_{ci} \sim \mathcal{N}\left(0, \sigma_c^2(1 - e^{-2\beta_c \Delta_i})/(2\beta_c)\right)$
   - Measurement: $\mu_c(t_{i+1}) = \mu_c(t_i) + \nu_c(t_i)\left(1 - e^{-\beta_c \Delta_i}\right)/\beta_c + \zeta_{ci}$.
   - Assume $\mu_c(t_0)$ is known.
   - $\zeta_{ci} \sim \mathcal{N}\left(0, \frac{\sigma_c^2}{\beta_c^2}\left(\Delta_i - 2\left(1 - e^{-\beta_c \Delta_i}\right)/\beta_c + \left(1 - e^{-2\beta_c \Delta_i}\right)/(2\beta_c)\right)\right)$
2. Compare the predicted measurements with the observations in a plot

E7 - Geolocation

1. Change your code from E6 to make μ to states such that
   - $Cov(\eta_{ci}, \zeta_{ci}) = \frac{\sigma_c^2}{2\beta_c^2}\left(1 - 2e^{-\beta_c \Delta_i} + e^{-2\beta_c \Delta_i}\right)$
   - Measurements: $y_c(t_i) = \mu_c(t_i) + \epsilon_{ci}$
   - Where $\epsilon_{ci} \sim \mathcal{N}(0, \sigma^2_{c,g(i)})$
   - With $\sigma_{c,g(i)}$ fixed at 0.005, 0.03, 0.5 for classes $g(i) = 1$, 2, and 3 respectively in both coordinates.
2. Compare the results with E6
3. Extend your model to include the measurement standard errors as parameters
4. Compare the results and time to estimate
5. Are the errors significantly different in the two coordinates?