Woods Hole

Getting data in and getting results out

Anders Nielsen

Getting data in

- We need to get data into R first — OK, or look into?
- We further need to pass data to the cpp program
- TMB uses 'computer memory'
- In other words no need to write stuff to files on disk
- A simple example:

```
42
```

```
library(TMB)
compile("scalar.cpp")
dyn.load(dynlib("scalar"))

Y<-scan("scalar.dat")

data <- list()
data$Y <- Y

param <- list()
param$mu <- 0

obj <- MakeADFun(data, param, DLL="scalar")
opt <- nlminb(obj$par, obj$fn, obj$gr)
opt$par
```

```
#include <TMB.hpp>
template<class Type>
Type objective_function<Type>::operator()()
{
  DATA_SCALAR(Y);
  PARAMETER(mu);
  Type nll = pow(Y-mu,2);
  return nll;
}
```

Transfers basic objects

What	R side (example)	c++ side
Number	`1`	`DATA_SCALAR`
Vector	`c(1,2,3)`	`DATA_VECTOR`
Matrix	`matrix(c(1,2,3,4), nrow=2, ncol=2)`	`DATA_MATRIX`
Array	`matrix(c(1,2,3,4), nrow=2, ncol=2)`	`DATA_ARRAY`
Integer	`1`	`DATA_INTEGER`
Integer Vector	`c(1,2,3)`	`DATA_IVECTOR`
Integer Matrix	`matrix(c(1,2,3,4), nrow=2, ncol=2)`	`DATA_IMATRIX`
Integer Array	`matrix(c(1,2,3,4), nrow=2, ncol=2)`	`DATA_IARRAY`
Factor	`factor(c("a","b"))`	`DATA_FACTOR`
String	`"a"`	`DATA_STRING`

Dimensions set by data

```
library(TMB)
compile("size.cpp")
dyn.load(dynlib("size"))
data <- list()
data$V <- 1:3
data$M <- matrix(1:6, nrow=2, ncol=3)
data$A <- array(1:6, dim=c(1,2,3))

param <- list()
param$mu <- 0
obj <- MakeADFun(data, param)
obj$report()

#Rout>  $Ad
#Rout>  [1] 1 2 3
#Rout>
#Rout>  $Vs
#Rout>  [1] 3
#Rout>
#Rout>  $Ms
#Rout>  [1] 6
#Rout>
#Rout>  $Mnrow
#Rout>  [1] 2
#Rout>
#Rout>  $As
#Rout>  [1] 6
#Rout>
#Rout>  $Mncol
#Rout>  [1] 3
#Rout>
```

```
#include <TMB.hpp>
template<class Type>
Type objective_function<Type>::operator()()
{
  DATA_VECTOR(V);
  DATA_MATRIX(M);
  DATA_ARRAY(A);

  int Vs=V.size();
  REPORT(Vs);

  int Ms=M.size();
  REPORT(Ms);
  int Mnrow=M.rows();
  REPORT(Mnrow);
  int Mncol=M.cols();
  REPORT(Mncol);

  int As=A.size();
  REPORT(As);
  vector<int> Ad(3);
  //Ad(0)=A.dim[0]; Ad(1)=A.dim[1]; Ad(2)=A.dim[2];
  Ad=A.dim;
  REPORT(Ad);

  PARAMETER(mu);
  Type nll = pow(mu,2);
  return nll;
}
```

Checking what is read in

- Useful to verify when developing models
- Can use `std::cout<<`, but `REPORT` is often easier:

```
library(TMB)
compile("verify.cpp")
dyn.load(dynlib("verify"))
data <- list()
data$V <- 1:3
data$M <- matrix(1:6, nrow=2, ncol=3)
data$A <- array(1:6, dim=c(1,2,3))

param <- list()
param$mu <- 0
obj <- MakeADFun(data, param)
out <- obj$report()
out$M==data$M

#Rout>      [,1] [,2] [,3]
#Rout> [1,] TRUE TRUE TRUE
#Rout> [2,] TRUE TRUE TRUE
```

```
#include <TMB.hpp>
template<class Type>
Type objective_function<Type>::operator()()
{
  DATA_VECTOR(V);
  DATA_MATRIX(M);
  DATA_ARRAY(A);
  REPORT(V);
  REPORT(M);
  REPORT(A);

  PARAMETER(mu);
  Type nll = pow(mu,2);
  return nll;
}
```

Indexing from 0

- In C^{++} the first element is **number 0**
- Different from R, so difficult to remember in the beginning

In R

```
data <- list()
data$y <- c(1.1, 2.2)
data$z <- mymatrix

y[1] ... y[n]
z[1,1] ... z[m,n]
```

In C^{++}-part

```
DATA_VECTOR(y)
DATA_ARRAY(z)

y(0) ... y(n-1)
z(0,0) ... z(m-1,n-1)
```

Exercise: No ragged arrays

- TMB has no ragged arrays.
- Consider how we would represent this data set in TMB?

```
> set.seed(123)
> ex <- lapply(c(7,9,13), function(n)round(rnorm(n),3))
> ex

[[1]]
[1] -0.560 -0.230  1.559  0.071  0.129  1.715  0.461

[[2]]
[1] -1.265 -0.687 -0.446  1.224  0.360  0.401  0.111 -0.556  1.787

[[3]]
 [1]  0.498 -1.967  0.701 -0.473 -1.068 -0.218 -1.026 -0.729 -0.625 -1.687
[11]  0.838  0.153 -1.138
```

Getting results out

- If estimated standard errors are not needed, then the
  - `REPORT(X)` in the C^{++} file
  - `obj$report()$X` in the R file
- If estimated standard errors are needed, then use
  - `ADREPORT(X)` in the C^{++} file (for derived quantities)
  - `summary(sdreport(obj))` in the R file
- To get estimates and standard deviations in the same format as they entered the parameter list, try:
  - Parameter list: `pl <- as.list(sdreport(obj),"Est")`
  - Parameter Sd list: `plsd <- as.list(sdreport(obj),"Std")`

Exercise: Beverton-Holt stock recruitment model

- The Beverton-Holt model can be written (slightly re-parametrized) as:

  $\log R_i = \log(a) + \log(\text{ssb}_i) - \log(1 + \exp(\log(b))\text{ssb}_i) + \varepsilon_i$ where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ independent

- Estimate the model parameters $\log(a)$ and $\log(b)$ and $\log(\sigma)$.
- A data set of SSB and $\log(R)$ is found in the shared folder