Preparing a process-based model for parameter estimation in R

FlorianHartig

25 Apr 2015

Synopsis: this short tutorial discusses how to link an existing process-based model that is written in C/C++ or another programming language to R, with the goal of doing automatic calibrations, parameter variation, etc.

Step1: making your model callable from R

To make parameter variation of calibrations from R, the first thing thing we need to do is to be able to execute the model with a given set of parameters. We will assume that we want to run the model with parameters coded in an object called *par*, which should be either a (named) vector or a list.

What happens now depends on how your model is programmed

1. Model in R, or R interface existing
2. Model accepts parameters via the command line
3. Model can be linked as dll
4. Model reads parameters only via parameter file

a) If your model is already coded in R Usually, you will have to do nothing. Just make sure you can call your model like that

```
runMyModel(par)
```

b) If you have a compiled model, parameters set via command line If your model can receive parameters via the command line, try something like that

```
runMyModel(par){

  # Create here a string with what you would write to call the model from the command line
  systemCall <- paste("model.exe", par[1], par[2])

  # this
  system(systemCall)

}
```

c) If you have a compiled dll, parameters are set via dll interface Use the dyn.load function to link to your model

```
dyn.load(model)

runMyModel(par){
```

```
  # model call here

}
```

d) If you have a compiled model, parameters set via parameter file Many models read parameters with a parameter file. In this case you want to do something like this

```
runMyModel(par){

  writeParameters(par)

  system("Model.exe")

}


writeParameters(par){

  # e.g.
  # read template parameter fil
  # replace strings in template file
  # write parameter file
}
```

How you do the write parameter function depends on the file format you use for the parameters. In general, you probably want to create a template parameter file that you use as a base and from which you change parameters

1. If your parameter file is in an *.xml format*, check out the xml functions in R
2. If your parameter file is in a *general text format*, the best option may be to create a template parameter file, place a unique string at the locations of the parameters that you want to replace, and then use string replace functions in R, e.g. grep to replace this string.

Step 2: Reading in data

For simple models, you might consider returning the model output directly with the runMyModel function. This is probably so for cases a) and b) above, i.e. model is already in R, or accepts parameters via command line.

More complicated models, however, produce a large number of outputs and you typically don't need all of them. It is therefore more useful to make on or several separate readData or getDate function. The only two different cases I will consider here is

1. via dll
2. via file ouputs

a) Model is a dll If the model is a dll file, the best thing would probably be to implement appropriate getData functions in the source code that can then be called from R

b) Model writes file output If the model writes file output, write a getData function that reads in the model outputs and returns the data in the desired format, typically the same that you would use to represent your field data.

```
getData(type = X){

  read.csv(xxx)

  # do some transformation

  # return data in desidered format
}
```

Result

From R, you shouls now be able to do something like that

```
par = c(1,2,3,4 ..)

runMyModel(par)

output <- getData(type = DesiredType)

plot(output)
```

Further readings

As introductions to what can be done AFTER the model is wrapped in R

- Tutorial by Marcel van Oijen on fitting forest models with Bayes here
- Hartig et al. (2012): Connecting dynamic vegetation models to data - an inverse perspective. J. Biogeogr., 39, 2240-2252. journal, pdf
- Van Oijen, M.; Rougier, J. & Smith, R. (2005) Bayesian calibration of process-based forest models: bridging the gap between models and data Tree Physiol., 25, 915-927