Getting started with FSmod

Aaron Springford

May 1, 2012

Abstract

The FSmod framework simulates directional migration of co-migrating populations (e.g. Fraser Sockeye salmon populations) in the presence of varying environmental conditions. Effects on the movement and survival of the populations as they migrate can be both acute and cumulative thanks to the use of BOTS (BOTS are Objects for Tracking States). The model can be configured to be stochastic or deterministic. It employs discrete timesteps and spatial increments that can be arbitrarily large or small (within computing limits). The framework is driven in large part by a series of configuration files that are read in at run-time and define the characteristics of the system. Due to the computational requirements typical of such large problems, the package can make use of a computing cluster, which greatly speeds computation.

The purpose of this document is to guide new users in installation and first use of the FSmod framework.

1 Installation

The FSmod R package can be downloaded from ----------. If you do not have R installed, it can be downloaded from `http://cran.us.r-project.org/`. You should have a relatively recent version of R installed, otherwise you may have issues with compatilibilty. For additional information, see `http://cran.r-project.org/doc/manuals/R-admin.html`.

Windows Installation If you are using Microsoft Windows, then you will want to download FSmod_1.0.zip. Then, open R and choose `Packages > Install package(s) from local .zip files...` Then choose the FSmod_1.0.zip file that you downloaded previously and follow the prompts (if any) that follow. If asked whether to create a local library (or something similar), then it is safe to say *yes*.

Mac/Linux/Unix Installation If you are using Mac, Linux, or other Unix-like operating systems, you will want to download the source version of the package FSmod_1.0.tar.gz.

If you are using Linux or Unix, then your distribution probably includes R, although you may have to use your package management tool to install it. This tool varies depending on the distribution of Linux (e.g. in Fedora, use `yum install R`). Any Linux or Unix distribution will have the necessary compilers to build and install FSmod from source (FSmod_1.0.tar.gz).

If you are using a Mac, then you will have to install the necessary compilers, typically by installing the Xcode Mac developers toolkit `http://developer.apple.com/`. More information is available at `http://r.research.att.com/tools/`.

Assuming that R and the necessary compilers are installed correctly, to build and install the package in the terminal, ensure that you are in the same directory as the downloaded FSmod_1.0.tar.gz and use

```
R CMD INSTALL FSmod_1.0.tar.gz
```

2 First use

In the code examples, I will denote the R command prompt with a > symbol, as it will appear when actually typing the commands into R.

2.1 Loading FSmod and retrieving input files

Once installed, start R and begin by loading the package with

```
> library(FSmod)
```

Help in R can be accessed using the `help()` function or the shorthand `?` version of help. Start by taking a look at the help for the FSmod package with either of the following commands:

```
> help(FSmod)
> ?FSmod
```

The help documents in FSmod are cross-referenced in order to help with discovering the various functions available. A quick read of the package help file, for instance, will point you towards the main simulation function `FSmodRun()` as well as the function `getFSmodFiles()`.

FSmodRun() is the main simulation function. It takes a long list of arguments (see the help file). We designed the software package with the following principles in mind:

1. It should be flexible. This means that it should be able to accomodate changes to things like the migration map, the timestep, the populations, the movement rates, the sensitivity to environmental conditions, etc. There should be very few (if any) things that are hard-coded and that can't be changed.

2. It should be extensible. This means that it should be able to be built upon / expanded easily and without having to modify the source code.

3. It should attempt to minimize execution time as much as possible. There is a tradeoff between this point and the first two. We have tried to keep the simulation time down by taking advantage of modern computing architectures such as computing clusters and using simpler data structures where possible.

These design principles led us to the current argument structure, which includes text files for the arguments that are data-like, and standard inputs for arguments that are parameter-like. For example, the migration map is read in (and then built) from the `reachNodes.file` and `reachDefs.file`. The number of timesteps to simulate is read directly from the argument `nT`. Building these structures at runtime addresses the flexibility principle.

The software is also highly extensible. The design allows GUI-based tools, for example, to be used easily by modification of the input files and subsequent execution controlled by the GUI. In addition, R provides a very rich set of tools for data manipulation, display, and so on.

The package ships with a number of input files to get started. These can be retrieved directly from the installation by using the helper function `getFSmodFiles()`. Ensure that the working directory for your R session is where you would like it to be. You can check the working directory with

```
> getwd()
```

and change it with

```
> setwd("/path/to/my/working/directory")
```

Then retrieve the input files using

```
> getFSmodFiles()
```

Note that by default, getFSmodFiles() will overwrite files with the same name in your working directory, so it can be used to *refresh* these files if you have changed them and want the defaults back. This behaviour can be changed by setting `overwrite=FALSE`.

This might be a good time to examine the help file for FSmodRun() along with the input files retrieved by getFSmodFiles(). Or, you can go directly to the next section and run your first simulation.

2.2 A first simulation

Once you have retrieved the default input files using getFSmodFiles(), you are ready to perform your first simulation using FSmodRun(). For your first run, consider changing the following default arguments:

- Set `nProcs` to the number of (logical) processors on your machine.
- Set `loadY=TRUE`. When set to FALSE, a large data.frame with timestep, reach, CU, environment variables, and the movement rate variable must be built from input files. This can be skipped for the default input files and saves simulation setup time. If the number of timesteps, the CUs, the environment variables, or movement rate variables are changed, then loadY should be set to FALSE.
- Optional: Set `maxrows=1e9`. When set to the default value of 1e6 (that is, 1 million rows), the `out.df` matrix containing the simulation results is written to a file if it exceeds 1 million rows. This is done to reduce memory requirements, but many modern computers (and especially those running a 64-bit operating system) can handle larger objects in memory. If the `out.df` matrix is written to file, it will not be returned by the FSmodRun function at the end of the simulation - the results will have to be retrieved from the file specified in `out.file`.

Go ahead and run your first simulation:

```
> res <- FSmodRun( nProcs = 4, loadY = TRUE )
```

The simulation will take several minutes to run, during which time status messages describing the simulation progress may be displayed. Leaving `plotTF=TRUE` and `plotTFinteract=TRUE` will cause plots and interactive plots with simulation results to be generated at the end of the simulation.

The FSmodRun function returns a list with the simulation results. See `?FSmodRun` for a description.