How to use the ss3sim package to run simulations in SS3

Sean C. Anderson, ...

1 Installing the ss3sim R package

The package can be installed and loaded with:

```
# install.packages(c("r4ss", "plyr", "MCMCpack")) # dependencies, if needed
# install.packages("devtools")
# devtools::install_github("ss3sim", username="seananderson")
library(ss3sim)
```

While the package is under active development, it's a good idea to install a new version every day or so using the `install_github` command:

```
devtools::install_github("ss3sim", username="seananderson")
```

You can read the help files and access this vignette again with:

```
help(package = "ss3sim")
vignette("ss3sim")
```

2 Shortcut to running FISH 600 models

Assuming you've set up SS3 in your path and are familiar with how the package works (as explained below), you can run FISH 600 models as follows. Note that this example runs 1 iteration and 2 bias adjustment runs.

```
run_fish600(iterations = 1, scenarios = "D0-E0-F0-G0-M0-R0-S0-cod",
bias_adjust = TRUE, bias_nsim = 2)
```

Typically we will run 100 iterations and 10 bias adjustment runs.

```
run_fish600(iterations = 1:100, scenarios = "D0-E0-F0-G0-M0-R0-S0-cod",
bias_adjust = TRUE, bias_nsim = 10)
```

You may want to run the iterations in chunks. In that case you'll want to tell `run_ss3sim` that you've already run the bias correction by setting the `bias_already_run` argument to `TRUE` and the `bias_adjust` argument to `FALSE`. When you run the bias correction runs you need to run at least one regular run. Therefore, you might do something like this:

```
run_fish600(iterations = 1, scenarios = "D0-E0-F0-G0-M0-R0-S0-cod",
bias_adjust = TRUE, bias_nsim = 10)

run_fish600(iterations = 2:100, scenarios = "D0-E0-F0-G0-M0-R0-S0-cod",
bias_adjust = FALSE, bias_already_run = TRUE)
```

Through this strategy you could run the second set (after running bias adjustment) using parallel processing or on separate machines. (Separate machines would require copying the existing folder structure to the second machine first.)

You can run multiple scenarios in sequence by expanding the vector of values passed to `scenarios`. For example,

```
run_fish600(iterations = 1:100,
scenarios = c("D0-E0-F0-G0-M0-R0-S0-cod", "D0-E0-F0-G0-M1-R0-S0-cod"),
bias_adjust = TRUE, bias_nsim = 10)
```

3 Putting SS3 in your path

`SS3` must be in your path for the `ss3sim` package to work. Your "path" is a list of folders that your operating system looks in whenever you type the name of a program on the command line. Having a binary in your path means that your operating system knows where to look for the file regardless of what folder you're working in.

3.1 For Unix (Linux and OS X)

To check if `SS3` is in your path: open a Terminal window and type `which SS3` and hit enter. If you get nothing returned then SS is not in your path. The easiest way to fix this is to move the `SS3` binary to a folder that's already in your path. To find existing path folders type `echo $PATH` in the terminal and hit enter. Now move the `SS3` binary to one of these folders. For example, in a Terminal window type:

```
sudo cp ~/Downloads/SS3 /usr/bin/
```

You will need to use `sudo` and enter your password after to have permission to move a file to a folder like `/usr/bin/`.

If you've previously modified your path to add a non-standard location for the `SS3` binary, you may need to also tell `R` about the new path. The path that `R` sees may not include additional paths that you've added through a configuration file like `.bash_profile`. You can add to the path that `R` sees by including a line like this in your `.Rprofile` file. (This is an invisible file in your home directory.)

```
Sys.setenv(PATH=paste(Sys.getenv("PATH"),"/my/non-standard/folder",sep=":"))
```

3.2 For Windows

To check if SS is in your path: open a DOS prompt and type `ss3 -?` and hit enter. If you get a line like "ss3 is not recognized . . . " then SS is not in your path. To add it to your path:

1. Find the latest version of the `ss3.exe` binary on your computer
2. Record the folder location. E.g. `C:/SS3.24o/`
3. Click on the start menu and type "environment"
4. Choose "Edit environment variables for your account" under Control Panel
5. Click on `PATH` if it exists, create it if doesn't exist
6. Choose `PATH` and click edit
7. In the "Edit User Variable" window add to the **end** of the "Variable value" section a semicolon and the `SS3` folder location you recorded earlier. E.g. `;C:/SS3.24o/`
8. Restart your computer
9. Go back to the DOS prompt and try typing `ss3 -?` and hitting return again.

4 Setting up the file structure

This package is set up assuming that you have an established base case operating model and estimation model to work with. Each operating model and estimation model should be in their own folder. The operating model folder should have the files:

```
yourmodel.ctl
yourmodel.dat
ss3.par
starter.ss
forecast.ss
```

The estimation model folder should have:

```
yourmodel.ctl
yourmodel.dat # optional; not used
starter.ss
forecast.ss
```

In both cases, nothing more and nothing less. The names of the `.ctl` and `.dat` files are not important. The package functions will rename them after they are copied to appropriate folders. These files should be derived from the `.ss_new` files but named as listed above. It's important to use these `.ss_new` files so they have consistent formatting. Many of the functions in this package depend on that formatting.

For the purposes of the FISH 600 project, we have unique case identifiers which combine to create unique scenarios. The types of cases are: natural mortality (M), fishing mortality (F), data quality (D), and retrospective (R). And the species are cod (cod), sardine-like (sar), and flatfish (fla). It's important to use these three letter abbreviations for the species since the functions assume that the last three letters represent a species (or some other identifier for a different project).

The different version of each case are identified with numbers. So, for example, the base case scenario for cod is identified as `M1-F1-D1-R1-cod`. We will have a spreadsheet that describes each of these in plain language.

The function `copy_ss3models` creates a folder structure and copies over the operating and estimation models. The folder structure looks like:

```
M1-F1-D1-R1-cod/1/om
M1-F1-D1-R1-cod/1/em
```

```
M1-F1-D1-R1-cod/2/om
M1-F1-D1-R1-cod/2/em
...
```

If you are using bias adjustment (`bias_adjust = TRUE`) then there will be some additional folders. In that case the folders will look like:

```
M1-F1-D1-R1-cod/bias/1/om
M1-F1-D1-R1-cod/bias/1/em
M1-F1-D1-R1-cod/bias/2/om
M1-F1-D1-R1-cod/bias/2/em
...
M1-F1-D1-R1-cod/1/om
M1-F1-D1-R1-cod/1/em
M1-F1-D1-R1-cod/2/om
M1-F1-D1-R1-cod/2/em
...
```

Note that the operating and estimating model folders have been renamed `om` and `em` within each iteration, the operating and estimation models have been checked to make sure they contain the minimal files (as listed above), the filenames have been made all lowercase, the data file has been renamed `data.dat`, the control files have been renamed `om.ctl` or `em.ctl`, and the starter and control files have been adjusted to reflect these new file names.

The functions in this package assume you've set your working directory in `R` to be the base folder where you will store the scenario folders.

5 Creating the input configuration files

You will need to have a folder containing "case" argument definitions. These plain text files are read by `get_caseval` and turned into argument lists that are passed to `run_ss3sim`. You can create template input files by running `create_argfiles`. It reads the various functions and parses the arguments and default values into plain text files. The default settings create these files:

1. `M0-spp.txt`
2. `F0-spp.txt`

3. `index0-spp.txt`
4. `agecomp0-spp.txt`
5. `lcomp0-spp.txt`
6. `R0-spp.txt`
7. `S0-spp.txt`
8. `G0-spp.txt`
9. `E0-spp.txt`

Look in your working directory for the template files. Change the case ID number (defaults to `0`) and the species identifier to a three letter identifier. For the FISH 600 project use one of `cod`, `sar`, or `fla` for cod, sardine, or flatfish. An example filename would be `M1-sar.txt` or `lcomp2-fla.txt`. The case `D1` corresponds to the files `index1-spp.txt`, `agecomp1-spp.txt`, and `lcomp0-spp.txt`. The other case types have single argument files.

The first column in the text files denotes the argument to be passed to a function. The second argument denotes the value to be passed. You can use any simple `R` syntax. For example: `c(1, 2, 4)`, or `seq(1, 100)` or `1:100` or `matrix()`. Character objects don't need to be quoted, but can be if you'd like. However, be careful not to use the delimiter (set up as a semicolon) anywhere else in the file besides to denote columns. You can add comments after any `#` symbols. Internally, the functions evaluate in `R` any entries that have no character values (e.g. `1:100`) or have an alphanumeric character followed by a `(`. Anything that is character only or has character mixed with numeric but doesn't have the regular expression `"[A-Za-z0-9]("` gets turned into a character argument.

Putting that all together, here's what an example `F1-cod.txt` file might look like:

```
years; 1:100
years_alter; NA
fvals; NA
```

6 Running the models

The `run_ss3sim` function is a wrapper function. It adjusts the natural mortality, adjusts the fishing mortality, adds recruitment deviations, calls `run_ss3model` to run the operating model, samples various survey estimates from the operating model, changes the age composition data as specified, changes the length composition data as specified, copies and renames files as necessary, and calls `run_ss3model` again to run the estimation model.

The `run_fish600` function is a higher-level wrapper that deals with parsing the case input files and then passes these arguments on to `run_ss3sim`. This is what we will use for the FISH 600 project.

See the PDF version of this vignette for a flow chart illustrating how `run_fish600` and `run_ss3sim` work:

```
vignette("ss3sim")
```

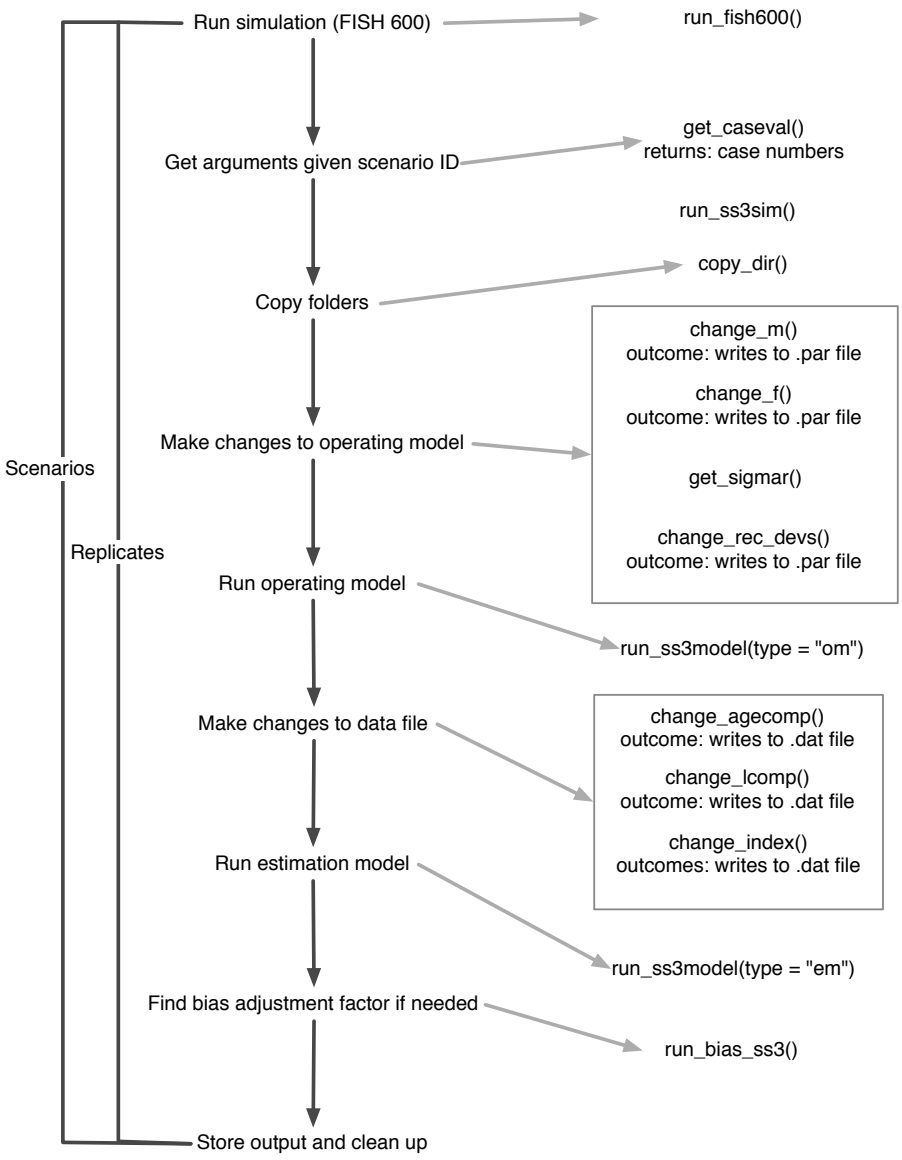

Figure 1: Simulation steps. Higher-level function calls are shown on the right.