# How to use the `ss3sim` package to run simulations in SS3

First start by installing the latest version of `ss3sim` and loading the package:

```
> # install.packages("devtools")
> # devtools::install_github("ss3sim", username="seananderson")
> library(ss3sim)
```

## Setting up the file structure

We are assuming there are a series of operating models in a folder and a series of estimation models in another folder. Within each folder, the models should be named according to whatever you would like the scenario ID to be. For our purposes, I suggest we use a brief identifier made up of lower-case letters and numbers followed by a dash followed by the species name. For example for a scenario with a block change in natural mortality you might have these folders:

```
blockm-cod
blockm-flat
blockm-sardine
```

It is up to the various groups to come up with these operating models and estimation models. There are a number of functions in this R package to facilitate this. We will come back to this.

Once you have these folders set up you can move them into the simulation folder structure with the `copy_model` function. Assuming you've put these in folders called `operating-models` and `estimation-models` you can copy the models over like this:

```
> copy_models(model_dir = "operating-models", type = "om")
> copy_models(model_dir = "estimation-models", type = "em")
```

or if you were only responsible for 1:50:

```
> copy_models(model_dir = "operating-models", type = "om",
+   iterations = 1:50)
```

This creates the structure:

```
blockm-cod/1/om
blockm-cod/1/em
blockm-cod/2/om
blockm-cod/2/em
...
```

Note that the operating and estimating model folders have been renamed `om` and `em` within each iteration.

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. The folders containing the operating and assessment scenarios should also be in this same base folder.

# Running the models

The `run_scenario` function is a wrapper function. It calls `run_model` to run the operating model, adds the recruitment deviations, samples various survey estimates from the operating model, copies and renames files as necessary, and calls `run_model` again to run the estimation model.

Say you have a text files of scenarios to run and you want to run the first 50 iterations of those scenarios. You could run them like this:

```
> scenarios <- scan("mysenarios.txt", what = "character")
> run_scenario(scenarios, iterations = 1:50)
```

Or, to test the operating model for the first scenario only:

```
> run_scenario(scenarios[1], iterations = 1, type = "om")
```

## The flat scenario ID structure

There are many advantages to this flat scenario ID fold setup:

1. It makes it easier for multiple papers to share scenarios.
2. It makes it easier for papers to change which scenarios to compare after.
3. It avoids unnecessary nested folder structure.
4. It's easier to distribute the model runs across people and computers.
5. The functions are more general and applicable to future research.
6. Since each folder represents a unique scenario run, it's simple to keep track of progress on model runs in a spreadsheet

Cole suggested we have a spreadsheet with the following columns:

`Scenario ID, Scenario description, Control modifications, Model status`

Then, groups can compile a list of scenario IDs they want to extract and compare.

## Setting up the models

The main functions to work with are:

`change_f` A function to alter fishing mortality values in the `.par` file

`add_time_varying_features` Adds time-varying natural mortality either through the environment, block, or deviation methods.

ADD EXAMPLES OF USING THESE FUNCTIONS FOR COMMON SCENARIOS

## What `run_scenario` does

Between running the operating and estimation models, `run_scenario` performs a number of tasks that are needed across all scenarios:

1. Takes the appropriate column of recruitment deviations from `data(recdevs)`, scales them by the appropriate standard deviation using `TODO`, and adds them to the `.par` file using `change_rec_devs`.
2. Samples the length and age composition data and adds them to the data file using `change_lcomp` and `change_agecomp`.
3. Jitters the index of abundance based on the reported biomass for each fleet using `jitter_index`.
4. TODO renames and moves files as needed