TLM Simulation Example

Raphaël McDonald

Contents

1 Simulation Example **1**

1.1 Step 1: Simulation Object 1

1.2 Step 2: Choosing Model Parameters 2

1.3 Step 3: Simulate Data 3

2 Simulation Study Example **4**

1 Simulation Example

This document aims to show a step by step example of how to run a simulation experiment using TLM, including all the decisions that are required to do this.

1.1 Step 1: Simulation Object

The first step is to create the object that is given to TMB in which data and random effects will be simulated. Many of the modelling decisions are made at this stage using the *simulate_obj()* function. Not all of this function's parameters are required by TLM (as the same function is used by SEBDAM), and those that are not will be ignored here.

- model: Choose between SEBDAM or TLM model
- n_years: The number of years in which to split the total observations simulated in this function.
- obs_mort: TRUE if we want to simulate observations of natural mortality as is possible in scallop fisheries with clappers, FALSE otherwise (and by default).
- n_obs: Number of observations desired.
- even_spread: TRUE to split the total number of observations evenly in each year, FALSE to introduce some variability in the number of observations per year
- fix_m: only required if obs_mort is FALSE, and is the value at which we want to fix the natural mortality
- gI: commercial size growth rates
- gR: recruit growth rates
- prior_q: TRUE if user desires to use a beta distribution to inform the commercial size catchability qI

```
simobj<-simulate_obj(model="TLM",n_years=10,obs_mort=F,n_obs=1000,even_spread=F,
                     gI=rep(1.1,10),gR=rep(1.4,10),prior_q=T)
```

This outputs the object that can be used to simulate new data, i.e. the skeleton of the upcoming simulations.

```
str(simobj)
```

```
## List of 13
##  $ model      : chr "TLM"
##  $ options_vec: num [1:2] 1 0
##  $ prior_pars : num [1:2] 10 12
##  $ logI       : num [1:1000] 1 1 1 1 1 1 1 1 1 1 ...
##  $ logIR      : num [1:1000] 1 1 1 1 1 1 1 1 1 1 ...
##  $ C          : num [1:11] 1 1 1 1 1 1 1 1 1 1 ...
##  $ g          : num [1:10] 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1
##  $ gR         : num [1:10] 1.4 1.4 1.4 1.4 1.4 1.4 1.4 1.4 1.4 1.4
##  $ n_tows     : num [1:10] 120 73 120 131 73 130 107 61 111 74
##  $ pos_tows_I : num [1:10] 1 1 1 1 1 1 1 1 1 1
##  $ pos_tows_IR: num [1:10] 1 1 1 1 1 1 1 1 1 1
##  $ t_i        : int [1:1000] 0 0 0 0 0 0 0 0 0 0 ...
##  $ set_m      : num 0.1
```

1.2 Step 2: Choosing Model Parameters

The second step involves choosing the values of the parameters used by TLM using the *simulate_TLM_params()* function.. These are the last preliminary decisions that are required before actually simulating data and random effects. Parameters are the following:

- simul_data_obj: simulation object obtained by step 1 using *simulate_obj()* function
- sigma_tau: underlying biomass process variance (default: 0.1).
- sigma_phi: underlying recruitment process variance (default: 0.1).
- sigma_epsilon: commercial biomass observation variance (default: 0.1).
- sigma_upsilon: recruit biomass observation variance (default: 0.1).
- qR: recruit catchability (default: 0.2).
- qI: commercial size catchability (default: 0.45).
- p_I: probability of positive commercial size biomass observations (default: 0.9).
- p_IR: probability of positive recruit biomass observations (default: 0.4).
- sigma_m: needed if obs_mort=TRUE, natural mortality process variance (default: 0.1).
- S: only required if using observations of natural mortality (obs_mort=TRUE), clapper catchability (default: 0.4).

```
#If defaults are good and using a single qI
# simpar<-simulate_TLM_params(simobj)
#Other examples:
simpar<-simulate_TLM_params(simobj,sigma_epsilon=0.5,sigma_upsilon=0.5)
```

1.3 Step 3: Simulate Data

Now that the modelling decisions have been made in step 1 and 2, the actual data can be simulated using the *simulate_data()* function which requires the following:

- simul_obj: object obtained from *simulate_TLM_params()* function in step 2.
- seed: to set the simulation seed for reproducibility.
- format: choose between default formatted data (data formatted as is necessary for model fitting functions, "formatted") or raw data ("raw") where everything is returned inside lists as understood by TMB.
- sim_obs_loc: used by SEBDAM, ignored by TLM.

```
simdata<-simulate_data(simpar,seed=20)
```

```
## Constructing atomic log_dbinom_robust
## Constructing atomic D_lgamma
## Constructing atomic log_dbinom_robust
## Constructing atomic D_lgamma
```

The raw format (not shown here) simply fills out the lists in the object obtained from simulate_TLM_params() with simulated data. The formatted data contains 4 lists:

- data: This contains the simulated observations formatted properly to fit TLM to:

```
str(simdata$data)
```

```
## 'data.frame':    1000 obs. of  3 variables:
##  $ I   : num  483 461 325 204 412 ...
##  $ IR  : num  52.1 0 54.3 0 0 ...
##  $ Year: num  1 1 1 1 1 1 1 1 1 1 ...
```

- growths: contains the commercial size and recruit growth rates as specified by the user in *simulate_obj()* in step 1.

```
str(simdata$growths)
```

```
## 'data.frame':    10 obs. of  2 variables:
##  $ g : num  1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1
##  $ gR: num  1.4 1.4 1.4 1.4 1.4 1.4 1.4 1.4 1.4 1.4
```

- processes: contains the simulated underlying processes. Not used in model fitting, but necessary to compare the model output at the end.

```
str(simdata$processes)
```

```
## 'data.frame':    11 obs. of  3 variables:
##  $ B: num  1000 1003 1194 1175 1183 ...
##  $ R: num  100 112 105 125 109 ...
##  $ m: num  0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 ...
```

- catch: contains appropriately formatted simulated landings

```
str(simdata$catch)
```

```
##  num [1:11] 99.6 88.4 88.1 142.7 116.3 ...
```

We now have data simulated from the TLM equations that can be fitted to it again or for any other purposes. For details of how to use the model fitting functions, see Fitting TLM to Data example.

2 Simulation Study Example

This is an example of how one could test out SEBDAM to see the impact of large observation variances on the outcome, from start to finish:

```
#Setting up to simulate data
simobj<-simulate_obj(model="TLM",n_years=10,obs_mort=F,n_obs=1000,even_spread=F,
                     gI=rep(1.1,10),gR=rep(1.4,10),prior_q=T)
simpar<-simulate_TLM_params(simobj,sigma_epsilon=0.5,sigma_upsilon=0.5)
#Setup for running the simulations multiple time
n_sim<-10
sim_data_list<-list()
mod_fit_list<-list()
parameters_list<-list()
pred_pro_list<-list()
#Running the simulations (should take a few seconds)
for (i in 1:n_sim){
  simdata<-simulate_data(simpar)
  sim_data_list[[i]]<-simdata
  dat_set<-data_setup(data=simdata$data,growths= simdata$growths,
                      catch=simdata$catch,model="TLM",
                      obs_mort=FALSE,prior=TRUE,prior_pars=c(10,12))
  mod_fit<-fit_model(dat_set,optim="optimr",
                       optim_method="nlminb",
                      control=list(eval.max=10000,iter.max=10000),
                      silent=T)
  mod_fit_list[[i]]<-mod_fit
  parameters<-get_parameters(mod_fit)
  parameters_list[[i]]<-parameters
  pred_processes<-get_processes(mod_fit)
  pred_pro_list[[i]]<-pred_processes
}
```

```
## [1] 0
## [1] "relative convergence (4)"
```

```
## Warning in sqrt(diag(cov)): NaNs produced
```

```
## [1] 0
## [1] "relative convergence (4)"
## [1] 0
## [1] "relative convergence (4)"
```

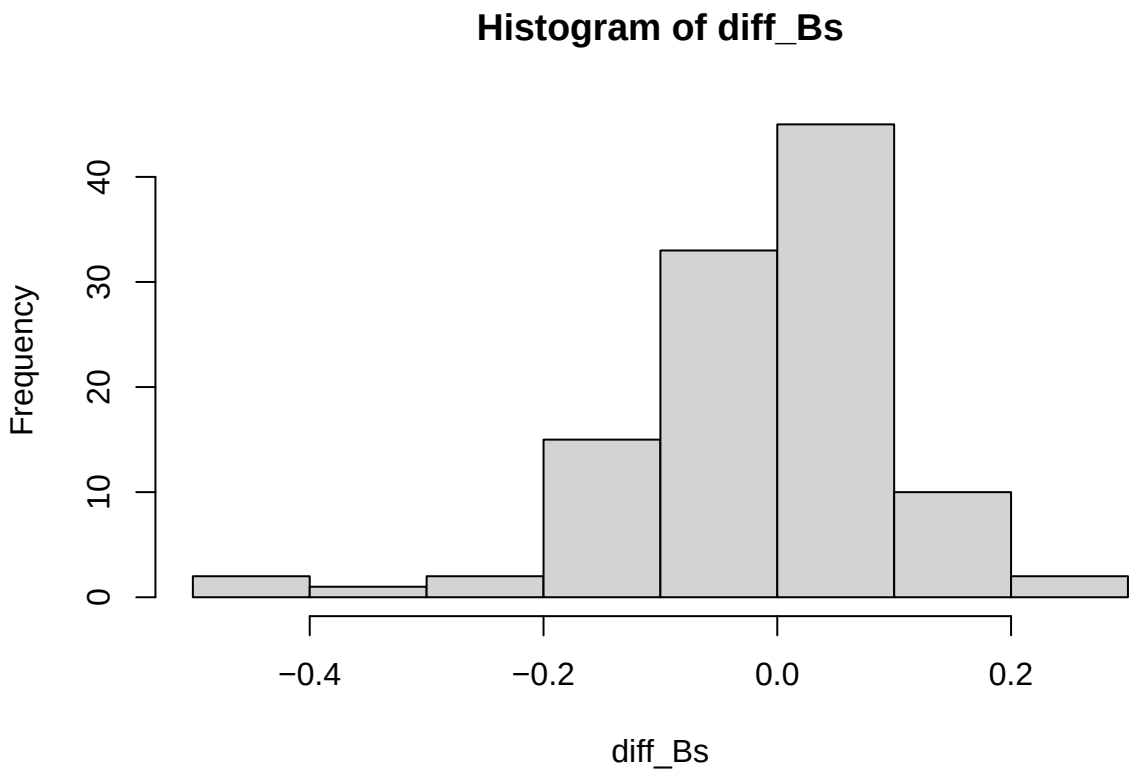
```
## [1] 0
## [1] "relative convergence (4)"
## [1] 0
## [1] "relative convergence (4)"
## [1] 0
## [1] "relative convergence (4)"
## [1] 0
## [1] "relative convergence (4)"
## [1] 0
## [1] "relative convergence (4)"
## [1] 0
## [1] "relative convergence (4)"
## [1] 0
## [1] "relative convergence (4)"
```

To see how the model is impacted by fitting an spde approach to data simulated with spde_aniso, we can see how well it captures the simulated biomass. However, it is important to only look at those fits that were successful, and remove those that did not converge or gave false convergence.

```
#ADD X AND relative convergence together
good_fits<-c()
for (i in 1:n_sim){
  if (mod_fit_list[[i]]$opt$message=="relative convergence (4)") {
    good_fits<-c(good_fits,i)
  }
}

diff_Bs<-matrix(rep(NA,n_sim*11),ncol=n_sim)
for (i in good_fits){
  sim_val<-sim_data_list[[i]]$processes$B
  pred_val<-pred_pro_list[[i]]$processes$B
  diff_Bs[,i]<-(sim_val-pred_val)/sim_val
}

hist(diff_Bs)
```




```
median(diff_Bs,na.rm=T)
```

```
## [1] 0.001534735
```

```
mean(diff_Bs,na.rm=T)
```

```
## [1] -0.01337263
```