StrathE2E2 User Manual (Version 3.3.1)

Michael R. Heath (m.heath@strath.ac.uk) and Ian Thurlbeck (i.thurlbeck@strath.ac.uk)
University of Strathclyde, Department of Mathematics and Statistics,
Glasgow, Scotland, UK

February 14, 2021

Contents

Summary

- **StrathE2E2** is an **R** package for modelling the 'big-picture', whole ecosystem effects of hydrodynamics, temperature, nutrient additions, and fishing on continental shelf marine food webs.
- **StrathE2E2** has two linked parts - a fishing fleet model and an ecology model. The fishing model integrates harvesting, discarding and seabed disturbance rates across a range of gears and passes the results into the ecology model.
- The ecology model is a network of coupled ordinary differential equations representing the rates of change in organic detritus, dissolved inorganic nutrient, and coarse guilds of living biomass spanning microbes to megafauna. The equations include representations of feeding, metabolism, reproduction, active migrations, advection and mixing. Environmental driving data include temperature, irradiance, hydrodynamics, and nutrient inputs from rivers, atmosphere and ocean boundaries.
- The package includes functions for parameter optimization, global sensitivity analysis, and Monte Carlo estimation of credible intervals for model outputs.

- A fully developed and documented implementation of StrathE2E2 for the North Sea is included with the package.

Package website

StrathE2E GitLab site and repository

Documentation

To see an overview of the package, launch an R-session and then:

```
library(StrathE2E2)      # Load the package
help(StrathE2E2)         # Launch the overview document in a browser window
```

Reference Manual - compilation of help pages for all the package functions generated by CRAN-check

Cheatsheet - quick-reference guide to StrathE2E2 functions

Technical Manual - documentation on the structure of input and output R-objects and files

Download the following supporting documents:

- Origin of the model (previous versions)
- Overview of the model concepts and design
- Ecology model description
- Fishing fleet model description
- Parameter optimization, sensitivity and Monte Carlo methodology
- Download documentation for the **North Sea model** which is provided in the package
- **Publications inventory** for StrathE2E models and applications

Funding

Development of the StrathE2E2 package was supported by the UK NERC Marine Ecosystems Research Programme (NE/L003120/1)
Other projects which have supported aspects of the model development are:
EU-Horizon 2020 (DISCARDLESS project)
Fisheries Innovation Scotland (project FIS003).

User community

A repository for contributed implementations of StrathE2E2 is available here. Users developing implementations which they are willing to share are encouraged to contact us to discuss inclusion in the repository.

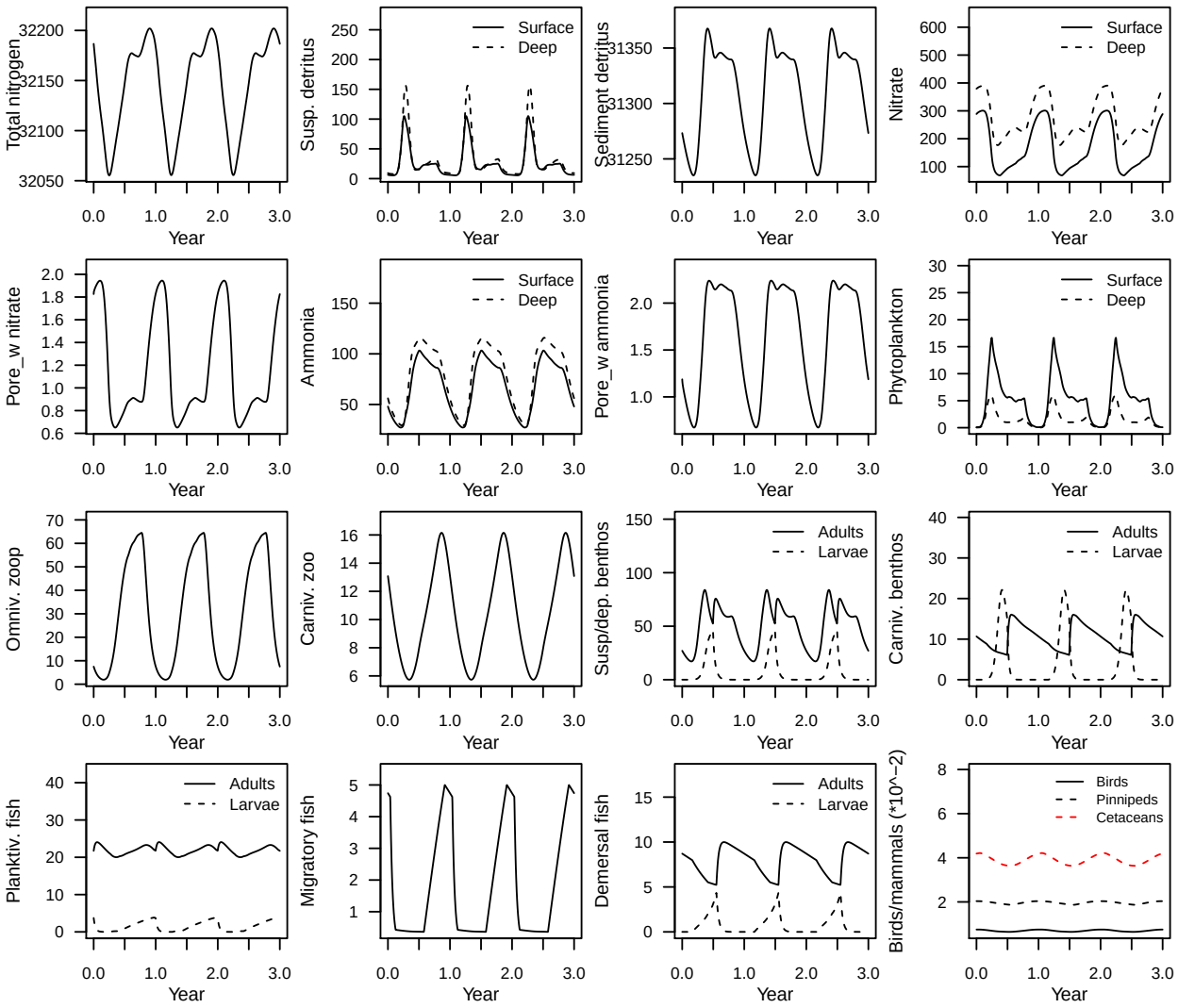
How to cite the package:

```
citation("StrathE2E2")

To cite package 'StrathE2E2' in publications use:

  Heath, M.R., Speirs, D.C., Thurlbeck, I. and Wilson, R. (2020).
```

```
  StrathE2E2: an R package for modelling the dynamics of marine food
  webs and fisheries. Methods in Ecology and Evolution, published
  online 11 October 2020. 8pp.

A BibTeX entry for LaTeX users is

  @Article{,
    title = {StrathE2E2: an R package for modelling the dynamics of marine food webs
          and fisheries},
    author = {Michael R. Heath and Douglas C. Speirs and Ian Thurlbeck and Robert Wilson},
    journal = {Methods in Ecology and Evolution},
    volume = {Published online 11 October 2020},
    year = {2020},
    issn = {2041-210X},
    url = {https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210X.13510},
    keywords = {food web, dynamic model, ecosystem, fisheries,
                 ordinary differential equations, optimization,
                 sensitivity analysis, Monte-Carlo, R},
  }
```

Quick start

```
library(StrathE2E2)                          # Load the package
model <- e2e_read("North_Sea", "1970-1999")  # Read the 1970-1999 North Sea model
Current working directory is...
'C:/Users/Public/Documents/StrathE2E2'
No 'results.path' specified so any csv data requested
will be directed to/from the temporary directory...
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ'

Model setup and parameters gathered from ...
StrathE2E2 package folder
Model results will be directed to/from ...
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ/North_Sea/1970-1999/'
results <- e2e_run(model,nyears=3)           # Run the model for 3 years
e2e_plot_ts(model,results,"ECO")             # Plot time series of the variables
```



1 Package structure

The package contains 8 families of functions:

- Model house-keeping (**Table 1**)
- Basic model operations (**Table 2**)
- Parameter estimation (**Table 3**)
- Sensitivity and Monte Carlo analyses (**Table 4**)
- Comparing model runs and observations (**Table 5**)
- Fishery yield curves (**Table 6**)
- Visualizing model inputs (**Table 7**)
- Visualizing model outputs (**Table 8**)

Table 1. Model house-keeping

Function name	Description
e2e_ls()	List the available models
e2e_copy()	Copy models between workspaces
e2e_get_parmdoc()	Download parameter documentation as a dataframe

Table 2. Basic model operations

Function name	Description
e2e_read()	Read a model setup
e2e_run()	Run StrathE2E for a prescribed number of years with a given setup
e2e_extract_start()	Create a new initial values file from the end-state of a model run
e2e_extract_hr()	Extract the values of harvest ratios generated by the fleet model
e2e_plot_ts()	Plot time series of model outputs for the full duration of a model run

Table 3. Parameter estimation

Function name	Description
e2e_optimize_eco()	Optimize ecology model parameters
e2e_optimize_hr()	Optimize harvest ratio scaling parameters
e2e_optimize_act()	Optimize fishing gear activity rates
e2e_plot_opt_diagnostics()	Plot diagnostic data for optimization runs
e2e_calculate_hrscale()	Calculate initial values of parameters linking effort to harvest ratios

Table 4. Senstivity and Monte Carlo analyses

Function name	Description
e2e_run_sens()	Run a global parameters sensitivity analysis with a given setup
e2e_run_mc()	Run a Monte Carlo analysis with a given setup
e2e_merge_sens_mc()	Merge parallel processing files from sensitivity or Monte Carlo runs
e2e_process_sens_mc()	Post-process raw output data from sensitivity or Monte Carlo runs
e2e_plot_sens_mc()	Plot diagnostic results from sensitivity or Monte Carlo runs
e2e_get_senscrit()	List the model outputs available as the basis for sensitivity analysis

Table 5. Compare model runs and observations

Function name	Description
e2e_compare_obs()	Box-plot comparisons between observations and model outputs
e2e_compare_runs_box()	Box-plot comparisons between two different model runs
e2e_compare_runs_bar()	Tornado bar-plots comparing two different model runs

Table 6. Fishery yield analysis

Function name	Description
e2e_run_ycurve()	Perform a set of model runs to generate fishery yield curve data
e2e_plot_ycurve()	Plot fishery yield curve data

Table 7. Visualize model inputs

Function name	Description
e2e_plot_edrivers()	Plot a climatological year of environmental driving data
e2e_plot_fdrivers()	Plot distributions of fishery-related driving data

Table 8. Visualize model outputs from the final year of a run

Function name	Description
e2e_plot_eco()	Plot annual cycles of ecology model variables
e2e_plot_catch()	Plot annual landings and discards
e2e_plot_migration()	Plot annual cycles of active migration fluxes
e2e_plot_trophic()	Plot mean trophic level and omnivory indices
e2e_plot_biomass()	Plot zonal distributions of annual average biomass densities

2 Getting started and model house-keeping

2.1 Installing the package

Download the installation zip file for your operating system from https://gitlab.com/MarineResourceModelling/StrathE2E and install into your R version. Note that **Strathe2E2** requires installation of the following additional packages which are available from any CRAN site:

- deSolve
- NetIndices

To start a session, begin by loading the package:

```
library(StrathE2E2)
```

2.2 Installing the supplementary data-package

The model package has a supplementary data-package - Strathe2E2examples - which contains example outputs from the more computationally intensive functions in the package. These are provided to illustrate the performance of various post-processing and plotting functions for these analyses that are available.

The first time that example data are called from a StrathE2E2 function, the data-package is downloaded and installed from the GitLab server https://marineresourcemodelling.gitlab.io/sran/index.html. Thereafter the example data are available offline.

Alternatively, users can manually install the package using:

```
install.packages("StrathE2E2examples",
                 repos="https://marineresourcemodelling.gitlab.io/sran")
```

Once the data-package has been installed, documentation on the data sets is available by typing:

```
library(StrathE2E2examples)
help(StrathE2E2examples)
```

2.3 Folder structure for model input and output files

All paths for reading model setup and paramater data and writing model results (when requested) are set using the **e2e_read()** function. This function creates a list object which contains all of the data required to completely define a given model. Read and write paths are defined RELATIVE TO THE CURRENT WORKING DIRECTORY through function arguments. The current working directory can always be obtained by the R command **getwd()**, or **e2e_read()** routinely returns the current working directory to the screen as a reminder.

The data which define a model (parameters and driving data) are held in a specific folder structure for each implementation. The location of the North Sea demonstration model data within the folder structure of the R-package is shown in **Figure 1**.

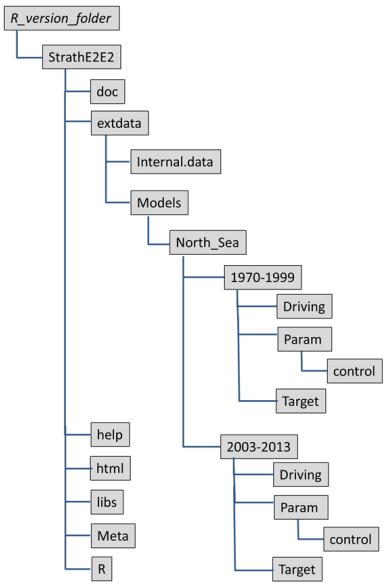


Figure 1. Locations of the definition data folders for the North Sea demonstration model and its two variants in the structure of the R-package.

The model folder structure within the model package is read-only, so users developing their own models will need to create an editable folder structure in their own accessible workspace to hold their model configuration data. In addition, users of the North Sea demonstration model wishing to use the more advanced optimization functions in the package will need to create a copy of the demonstration models in their own workspace, because these functions require to write data back into the **/Param** sub-folders on completion of the optimization. This can be done using using any file manager or the **e2e_copy()** function provided with the package. The structure expected for any user-defined model folders is shown in **Figure 2**.

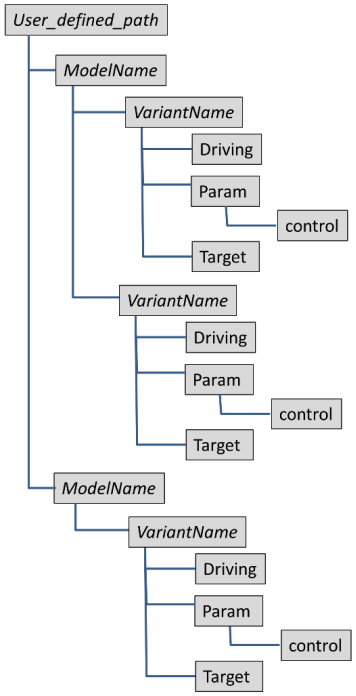


Figure 2. Organisation of model definition data folders in any user-defined work-space. Italicised folder

names are free for user specification.

Most of the model functions output R-data objects which are held in memory and available for user-analysis. In addition, many of the contents of the list objects can, on request, be output to comma-separated-variable (csv) files in a folder structure in the current user workspace (**Figure 3**). Requests to generate csv ouputs are issued through an argument **csv.output** in each relevant function call, with the dafult being FALSE in each case. The results folder path must be specified by an argument of an **e2e_read()** function call which preceeds any model operation. The package will auto-create modelname and modelvariant sub-folders within the designated results folder if none already exist. If no results filepath is been specified then any output subsequently requested is directed to a temporary system folder.

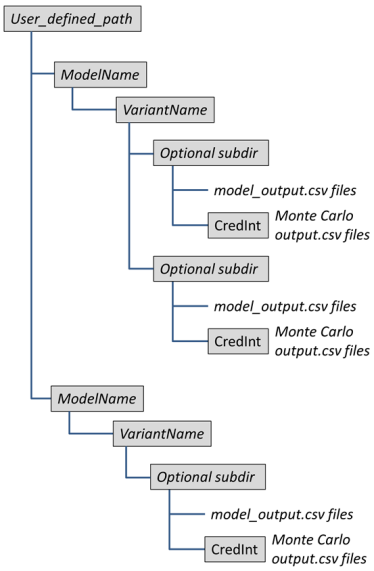


Figure 3. Organisation of model results folders. Italicised folder names are free for user specification.

2.4 Listing and copying model folders

2.4.1 Listing the available models e2e_ls()

List the available models in a designated workspace.

Table 9. Arguments of the **e2e_ls()** function

Argument	Description
models.path	Relative path from the current working directory to a folder containing a library of model configurations (typically "Folder/Models"). Setting models.path="" is valid. Default models.path=NULL, meaning read a North Sea model setup from the package folder extdata/Models

```
library("StrathE2E2")
e2e_ls()
Current working directory is 'C:/Users/Public/Documents/StrathE2E2'
List of package models in system folder "extdata/Models", with examples of how to read them:
 Model: "North_Sea"
  Variant: "1970-1999"  model <- e2e_read("North_Sea", "1970-1999")
  Variant: "2003-2013"  model <- e2e_read("North_Sea", "2003-2013")
```

2.4.2 Copying a named model

Make a copy of a named model/variant and documentation in a user defined workspace.

Table 10. Arguments of the **e2e_ls()** function

Argument	Description
model.name	Name of a model to copy
model.variant	Name of a model variant to copy
source.path	Relative path from the current working directory to the source model to be copied. Setting source.path="" is valid. Default source.path=NULL, meaning read a North Sea model setup from the package folder extdata/Models
dest.path	Relative path from the current working directory to a destination address in which to create a 'Models' folder if necessary, and then copy the model files. Setting dest.path="" is valid. Default dest.path=NULL in which case the Models folder will be created in a temporary directory

```
library("StrathE2E2")
# Copy the 2003-2013 version of the North Sea model supplied with the package into a
# temporary folder:
e2e_copy("North_Sea", "2003-2013")
#
# Copy the 2003-2013 version of the North Sea model supplied with the package into a
# used-defined folder (edit "Folder/Models to your own relative path):
#   e2e_copy("North_Sea", "2003-2013",dest.path="Folder/Models")
#'
# Copy a user model into a user workspace:
# Replace "Folder1/Models" and "Folder2/Models" with your own source.path and dest.path
# remembering that these are relative to the current working directory.
# e.g.... e2e_copy("Modelname", "Modelvariant",
#         source.path="Folder1/Models",
#         dest.path="Folder2/Models")
```

Copies of the North Sea model make a useful template for development of new models.

2.5 Downloading parameter documentation e2e_get_parmdoc()

Get documentation on the parameters inthe model in a dataframe format

Table 11. Arguments of the **e2e_ls()** function

Argument	Description
id	Integer value denoting the class of parameter details to be downloaded

The full range of inputs to the model includes both numeric values which are constant over the during a run, and values which vary over time during a run according to an externally provided schedule. We refer to the latter as 'drivers', but they are all parameters nontheless. The constant parameters are divided into categories according to whether they are associated with the ecology model, the fishing fleet model, and whether they are available for optimization or not.

This function downloads documentation on the full range of parameters in the form of a dataframe which can be used to form labels in user-generated code. The parameters are identified by a numeric value linked to 6 different classes of parameter. Choose from: 0 = fitted ecology, 1 = fixed ecology, 2 = fishing fleet, 3 =

harvest ratios, 4 = environmental drivers, 5 = physical configuration, 9 = all (default = 9).

Note that the function does not supply the numeric values of parameters for any given model setup. These are available in the model definition object generated by the function e2e_read().

```
parm_list <- e2e_get_parmdoc(0)       # Get documentation on the fitted ecology parameters
parm_list <- e2e_get_parmdoc(2)       # Get documentation on the fleet model parameters
parm_list <- e2e_get_parmdoc()        # Get documentation all parameters
head(parm_list)
  parametername parameterid fixfit        Parameter.class Parameter.description
1       thik_so           1      5 Physical configuration    Vertical thickness
2        thik_d           2      5 Physical configuration    Vertical thickness
3       thik_si           3      5 Physical configuration    Vertical thickness
4        thik_b           4      5 Physical configuration    Vertical thickness
5   porosity_s1          28      5 Physical configuration     Sediment porosity
6   porosity_s2          29      5 Physical configuration     Sediment porosity
         Model.guild.or.feature
1     Offshore zone upper layer
2     Offshore zone lower layer
3                  Inshore zone
4 Benthic boundary feeding layer
5       Inshore muddy sediments
6       Inshore sandy sediments
```

3 Basic model operations

3.1 Reading a model configuration e2e_read()

Read all the configuration, driving and parameter files for a designated model and compile into a list object.

Table 12. Arguments of the **e2e_read()** function

Argument	Description
model.name	Name of model to read (no default)
model.variant	Variant of the model to be read (no default)
models.path	Relative path from the current working directory to a folder containing a library of model configurations (typically "Folder/Models"). Setting models.path="" is valid. Default models.path=NULL, meaning read a North Sea model setup from the package folder extdata/Models
results.path	Relative path from the current working directory to a folder for writing and reading model output files (e.g. "E2E_results"). Setting results.path="" is valid. Model-specific sub-folders will be assigned and if necessary auto-created according to the model name and variant. Default results.path=NULL, meaning write/read to/from a temporary directory
results.subdir	Subdirectory of "working_directory/results.path/model_name/model_variant" to be created if required. (Default="", meaning no subdirectory will be created)
model.ident	Identifier text appended to output file names (e.g. OFFSHORE_model_annualresults-TEXT.csv instead of just OFFSHORE_model_annualresults.csv). (Default="base")
quiet	Logical. If FALSE then see on-screen information on indvidual parameter files as they are read (default=TRUE)
silent	Logical. If FALSE then see on-screen information on model and results paths (default=FALSE). If set TRUE then this over-rides any quiet=FALSE setting and forces quiet=TRUE

The **e2e_read()** function is the foundation for all StrathE2E2 model runs. This function loads all the data required to configure a model into an R-list object. The function is pointed to each of the required input files by the file **MODEL_SETUP.csv** which resides in the **/ModelName/VariantName** folder for the relevant model. **MODEL_SETUP.csv** points to specific files in the **/Param**, **/Target** and **/Driving** subfolders.

The default **models.path=NULL** for **e2e_read()** is to load one of the variants of the North Sea model supplied with the package. The North Sea models are configured so that the initial conditions which are loaded for a default run launch the system in a stationary state - that is, the model produces repeating annual cycles of output given repeating annual cycles of the time-dependent driving data (e.g. temperature, hydrodynamics etc).

The default **results.path=NULL** for **e2e_read()** is to write output files to a temporary system folder, which is reported by the **e2e_read()** function. However, instigating csv outputs, either to the temporary folder or to a path controlled by the user is determined by the logical argument **csv.output** in each of the relevant package functions, the default being FALSE.

```
library("StrathE2E2")
# Load the 1970-1999 version of the North Sea model supplied with the package; outputs go
# to a temporary results folder; default model.ident setting.
# Default screen information settings - on-screen parameter file information suppressed but
# path information enabled:
model <- e2e_read("North_Sea", "1970-1999")
Current working directory is...
'C:/Users/Public/Documents/StrathE2E2'
No 'results.path' specified so any csv data requested
will be directed to/from the temporary directory...
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ'

Model setup and parameters gathered from ...
StrathE2E2 package folder
Model results will be directed to/from ...
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ/North_Sea/1970-1999/'
#
# User-specified model.ident; on-screen parameter file and path information enabaled:
model <- e2e_read("North_Sea", "1970-1999",model.ident="baseline",quiet=FALSE)
Current working directory is...
'C:/Users/Public/Documents/StrathE2E2'
No 'results.path' specified so any csv data requested
will be directed to/from the temporary directory...
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ'

Loading model   : C:/Users/ais04103/OneDrive - University of Strathclyde/Documents/R/win-library/4.0/St
 Reading CSV file: MODEL_SETUP.csv
 Reading CSV file: physical_parameters_NORTH_SEA.csv
 Reading CSV file: fixed_miscellaneous.csv
 Reading CSV file: fixed_consumers_NORTH_SEA_1970-1999.csv
 Reading CSV file: physics_NORTH_SEA_1970-1999.csv
 Reading CSV file: chemistry_NORTH_SEA_1970-1999.csv
 Reading CSV file: event_timing_NORTH_SEA_1970-1999.csv
 Reading CSV file: fitted_preference_matrix_NORTH_SEA.csv
 Reading CSV file: fitted_uptake_mort_rates_NORTH_SEA.csv
 Reading CSV file: fitted_microbiology_others_NORTH_SEA.csv
 Reading CSV file: initial_values_NORTH_SEA_1970-1999.csv
 Reading CSV file: fishing_activity_NORTH_SEA_1970-1999.csv
```

```
 Reading CSV file: fishing_power_NORTH_SEA_1970-1999.csv
 Reading CSV file: fishing_discards_NORTH_SEA.csv
 Reading CSV file: fishing_distribution_NORTH_SEA.csv
 Reading CSV file: fishing_fleet_NORTH_SEA_1970-1999.csv
 Reading CSV file: fishing_processing_NORTH_SEA.csv
 Reading CSV file: fishing_gear_multiplier.csv
 Reading CSV file: harvest_ratio_multiplier.csv
Model setup and parameters gathered from ...
StrathE2E2 package folder
Model results will be directed to/from ...
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ/North_Sea/1970-1999/'
#
# All on-screen information suppressed even though quiet=FALSE:
model <- e2e_read("North_Sea", "1970-1999",model.ident="baseline",quiet=FALSE,silent=TRUE)
# View the top 2 levels of input list object
str(model,max.level=2)
List of 2
 $ setup:List of 7
  ..$ read.only    : logi TRUE
  ..$ model.name   : chr "North_Sea"
  ..$ model.variant: chr "1970-1999"
  ..$ model.ident  : chr "baseline"
  ..$ model.subdir : chr ""
  ..$ model.path   : chr "C:/Users/ais04103/OneDrive - University of Strathclyde/Documents/R/win-library
  ..$ resultsdir   : chr "C:\\Users\\ais04103\\AppData\\Local\\Temp\\RtmpiySPHQ/North_Sea/1970-1999/"
 $ data :List of 8
  ..$ fixed.parameters   :List of 51
  ..$ fitted.parameters  :List of 171
  ..$ physical.parameters:List of 63
  ..$ physics.drivers    :'data.frame': 12 obs. of  23 variables:
  ..$ chemistry.drivers  :'data.frame': 12 obs. of  22 variables:
  ..$ biological.events  :'data.frame': 26 obs. of  2 variables:
  ..$ fleet.model        :List of 19
  ..$ initial.state      :List of 403
```

The object **model** is a list containing all of the input data and parameters gathered from the csv files in the folder structure:

```
# Load a user defined model from a user workspace
# ... substitute your own path details for e.g. "Folder/Models":
#     model<-e2e_read("Modelname", "Variantname",
#                     models.path="Folder/Models",
#                     model.ident="demo")
#'
# Load a user defined model from a user workspace and send the model outputs to
# a specified folder ... substitute your own path details for the examples shown here
# for "Folder/Models" and "Folder/results":
#     model<-e2e_read("Modelname", "Variantname",
#                     models.path="Folder/Models",
#                     results.path="Folder/results",
#                     model.ident="demo")
```

Full documentation of the input csv files and the contents of the list object produced by **e2e_read()** is provided in the Technical Manual.

3.2 Running a model e2e_run()

Run the StrathE2E model with a given configuration.

Table 13. Arguments of the **e2e_run()** function

Argument	Description
model	R-list object defining the model configuration compiled by the e2e_read() function
nyears	Number of years (integer) to run the model (default=20)
quiet	Logical. If FALSE then see status outputs during the model run (default=TRUE)
csv.output	Logical. If TRUE then enable writing of csv output files (default=FALSE)

The function **e2e_run()** runs a single instance of the model, and generates another list object containing all of the outputs. In addition a range of derived outputs for the final year of the run are exported as csv files to a **/results** folder in the users local workspace.

```
results <- e2e_run(model, nyears=3)          # Run the model for 3 years
str(results,max.level=1)                     # View the contents of the output list object
List of 7
 $ build               :List of 4
 $ output              :'data.frame':   1081 obs. of  404 variables:
 $ aggregates          :List of 130
 $ fleet.output        :List of 7
 $ total.annual.catch  :List of 2
 $ annual.catch.by.gear:List of 2
 $ final.year.outputs  :List of 27
```

Full documentation of the output csv files and the contents of the list object produced by **e2e_run()** is provided in the Technical Manual.

3.3 Setting initial conditions e2e_extract_start()

Extract the values of all state variables at the end of a model run and format for use as new initial conditions.

Table 14. Arguments of the **e2e_extract_start()** function

Argument	Description
model	R-list model object for the model run generated by the function **e2e_read()**
results	R-list results object generated by the function **e2e_run()**
csv.output	Logical. If TRUE then enable writing of csv file (default = FALSE)

The function **e2e_extract_start()** saves the state of the model at the end of a run (using the **e2e_run()** function) for use as initial conditions in future runs. This enables, for example, the model to be run for a long time to attain a stationary state, and then restarted in that state.

Initial conditions for a model run are held in a csv file with the prefix **initial_values** in the **/Param** folder of the **ModelName/VariantName** path specified in the **e2e_read()** function call used to define a run. By default, the function simply returns the new initial conditions as a data-object. Set **csv.output=TRUE** to enable writing of the model end-state file back to the **/ModelName/VariantName/Param** folder. However, the package folders are read-only so if **e2e_read()** has been specified to load an internally provided **ModelName/VariantName** then the output will revert to the currently specified results folder instead. To fix this, copy the required package model to a user workspace using the **e2e_copy()** function and re-run.

The internal North Sea models are both supplied with initial conditions which are consistent with a stationary

state, so they should both show repeating annual cycles of output from time = 0, unless the parameters or driving conditions are altered to create some new environmental scenario.

```
model<-e2e_read("North_Sea", "2003-2013",silent=TRUE) # Read the 2003-2013 internal model
results <- e2e_run(model, nyears=3)                   # Run the model for 3 years
new_start_data <- e2e_extract_start(model,results)
# This will produce the new data only as a dataframe
head(new_start_data)
#
#
# Copy the 2003-2013 version of the North Sea model supplied with the package into a
#    user workspace (Windows OS):
e2e_copy("North_Sea", "2003-2013",
            dest.path="Documents/Models")
# Load the 2003-2013 version of the North Sea model from the user workspace:
model<-e2e_read(model.name="North_Sea",
                  model.variant="2003-2013",
                  models.path="Folder/Models",
                  model.ident="TEST")
results <- e2e_run(model, nyears=3)                   # Run the model for 3 years
new_start_data <- e2e_extract_start(model,results,csv.output=TRUE)
# Extracts new initial conditions file
# This will be written to the user model /Param folder)
```

3.4 Extracting harvest ratios from a model run e2e_extract_hr()

Extract the values of harvest ratios generated by the fleet model.

Table 15. Arguments of the **e2e_extract_hr()** function

Argument	Description
model	R-list model object for the model run generated by the function **e2e_read()**
results	R-list results object generated by the function **e2e_run()**
csv.output	Logical. If TRUE then enable writing of csv file (default = FALSE)

Harvest ratios for each guild in each zone (proportion of biomass captured per day) are a key input to the ecology model. They are generated by the fleet model based on inputs of fishing gear activity rates, distribution in space, and catching power for each guild.

The function **e2e_extract_hr()** returns a dataframe, and optionally a csv file, of harvest ratios in each zone and the domain as a whole, for each guild in the ecology model. These are extracted from the results object generated by running the **e2e_run()** function.

```
# Load the 2003-2013 version of the North Sea model from the user workspace:
model<-e2e_read(model.name="North_Sea",
                  model.variant="2003-2013")
Current working directory is...
'C:/Users/Public/Documents/StrathE2E2'
No 'results.path' specified so any csv data requested
will be directed to/from the temporary directory...
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ'

Model setup and parameters gathered from ...
StrathE2E2 package folder
```

```
Model results will be directed to/from ...
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ/North_Sea/2003-2013/'
results <- e2e_run(model, nyears=3)                    # Run the model for 3 years
harvest_ratio_data <- e2e_extract_hr(model,results)    # Extract the harvest ratio data
harvest_ratio_data
               Guilds Inshore_harvest_ratio Offshore_harvest_ratio
1  Planktivorous_fish          7.116297e-04           5.534445e-04
2       Demersal_fish          3.075409e-04           2.024661e-04
3      Migratory_fish          1.063235e-03           9.874583e-04
4    Benthos_susp-dep          3.319640e-04           1.955593e-04
5   Benthos_carn-scav          9.567688e-04           1.155176e-04
6    Zooplankton_carn          3.755834e-04           6.034284e-04
7               Birds          2.012939e-06           1.530527e-06
8           Pinnipeds          4.750955e-05           6.215574e-06
9           Cetaceans          7.860659e-04           1.614313e-04
10        Macrophytes          0.000000e+00           0.000000e+00
   Whole_domain_harvest_ratio
1                5.929228e-04
2                2.286897e-04
3                1.006370e-03
4                2.296018e-04
5                3.254689e-04
6                5.465650e-04
7                1.650923e-06
8                1.652132e-05
9                3.173215e-04
10               0.000000e+00
```

3.5 Plotting time-series of model outputs e2e_plot_ts()

Plot ecology model state variables or fishery catches for the full duration of a model run.

Table 16. Arguments of the function for plotting full-length time-series data from model outputs

Argument	Description
model	R-list model object for the model run generated by the function **e2e_read()**
results	R-list results object generated by the function **e2e_run()**
selection	Text string from a list identifying whether ecology state variable or fishery catches are to be plotted. Select from: "ECO", "CATCH", default = "ECO". Remember to include the phrase within "" quotes.

The function **e2e_plot_ts()** plots a multi-panel page of time series plots of either a) daily ecology state variable values aggregated over the inshore and offshore zones of the domain and over sediment classes, or b) annual landings and discards by zone for each model guild, for the full duration of a model run. Currently the masses of macrophytes, corpses and discards are not included in the state varibale plots due to space constraints.

Be warned that if the run is more than about 10 years then the plot becomes extremely compressed and messy. It is not intended to be of publication quality. The intention is to provide a quick-look diagnostic of trends in the state variables. This is useful to assess whether the model is close to stationary state or not.

Units of the plotted state varaiables are mMN i.e. mass in the model domain without any scaling to zone-area or layer thickness. Similarly, units of catches are mMN.y^{-1} from the whole model domain without any scaling to zone-area. The assumed area of the whole model domain is 1 m^2.

Selection of either ecology state variable or catches to plot is by a function argument.

A separate set of functions is provided for plotting more detailed visualizations of just the final year from a model run, e.g. **e2e_plot_eco()**, **e2e_plot_catch()**.

```
model<-e2e_read(model.name="North_Sea",
                model.variant="1970-1999")
Current working directory is...
'C:/Users/Public/Documents/StrathE2E2'
No 'results.path' specified so any csv data requested
will be directed to/from the temporary directory...
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ'

Model setup and parameters gathered from ...
StrathE2E2 package folder
Model results will be directed to/from ...
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ/North_Sea/1970-1999/'
results <- e2e_run(model, nyears=3)
e2e_plot_ts(model, results,selection="ECO")
```

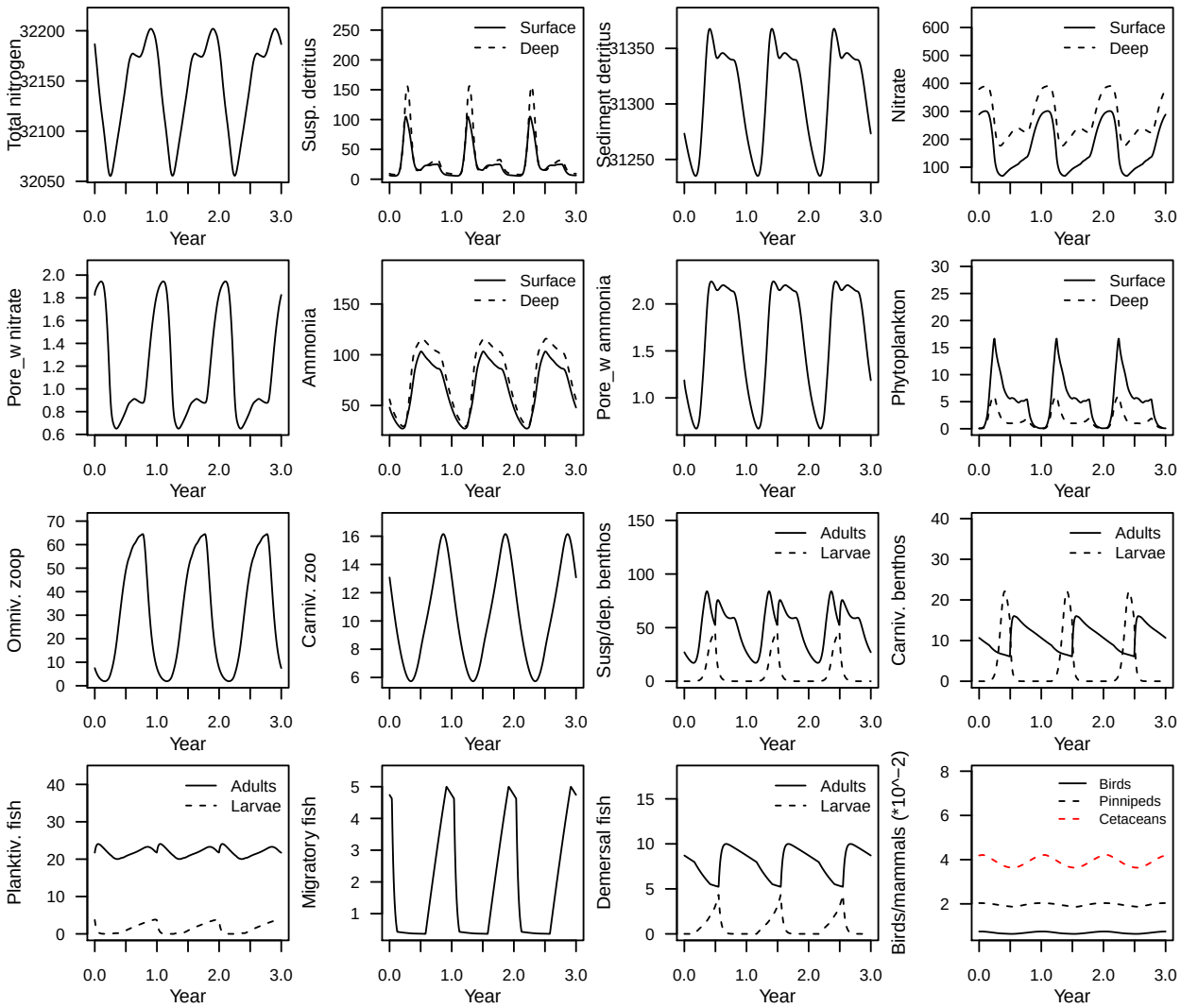


Figure 4. Time-series plot of ecology model output, units mMN in the model domain as a whole (1 m^{-2} sea surface).

```
e2e_plot_ts(model, results,selection="CATCH")
```

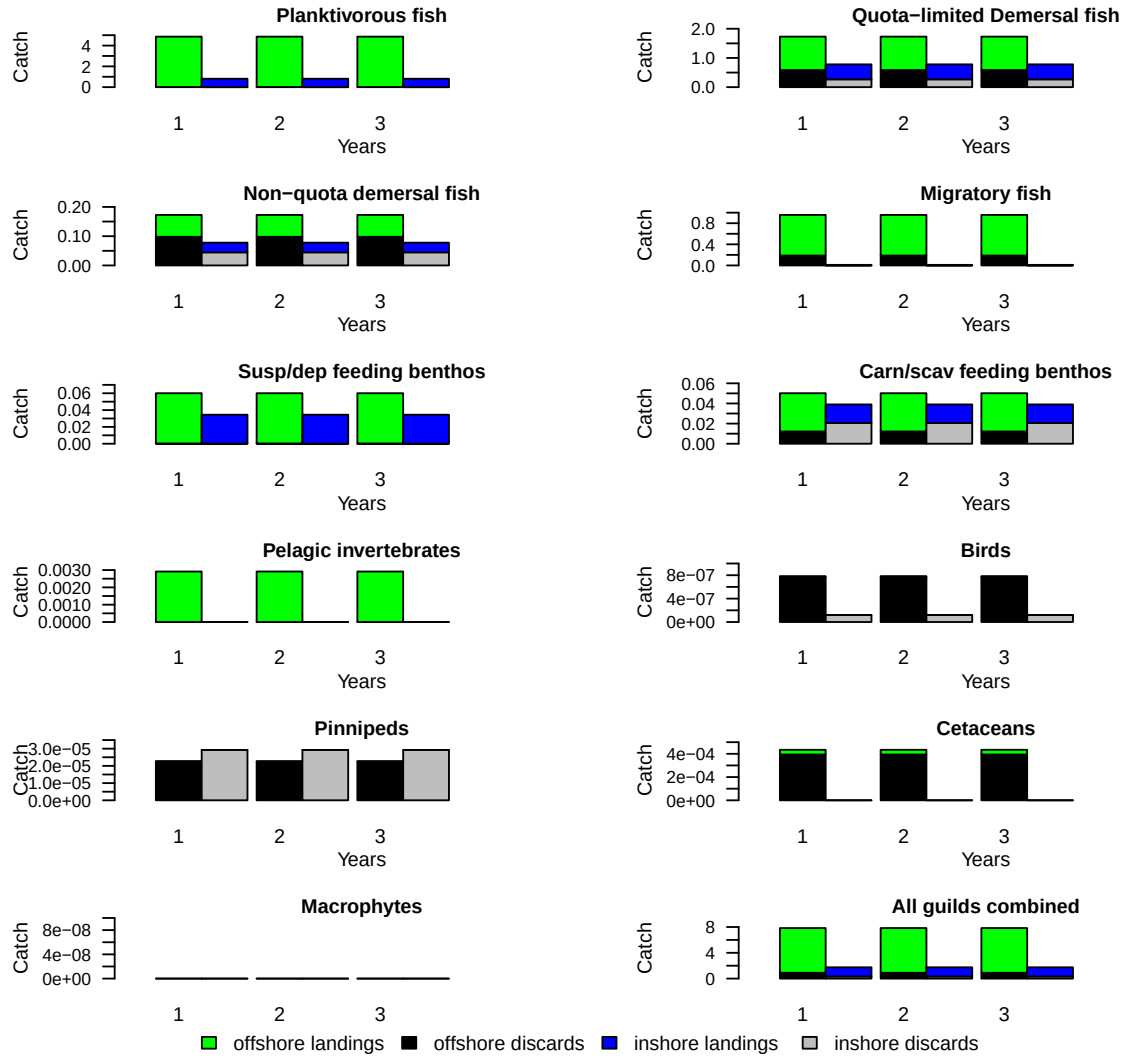


Figure 5. Time-series plot of catch data output, units mMN.y^{-1} in the model domain as a whole (1m^{-2} sea surface).

All of the plotting functions provide with the package produce output to a screen window. To divert output to a graphic file wrap the plotting function call inside graphics device open and close statememts as follows:

```
pdf("plot.pdf",width=7,height=6)                    # or jpeg("plot.jpg"), png("plot.png")
e2e_plot_ts(model, results,selection="CATCH")
dev.off()
```

4 More advanced use of the model

4.1 Writing your own code to create model scenarios

Typical use of the model might involve

- comparing a baseline run with a scenario run involving some changes in driving data (e.g. different temperature conditions or different activity rates of selected gears),
- conducting an analysis of the sensitivity to systematic changes in driving data (e.g. incrememts in river nutrient inputs over some range of values).

Users have two options for configuring such experiments

- manually edit the .csv input files as required and rerun the model,
- use coding to alter the required elements of the R list object created by the **e2e_read()** function, assigning a unique identifier to the outputs from each run. This is the most efficient way to create model scenarios

The scope for developing fishing and/or environmemtal scenarios by coding to modify a default model object loaded by **e2e_read()** is limitless, but requires and understanding of the structure and content of the list object produced by the **e2e_read()** function. Use the **str()** function in R to explore the list and its sub-components, and consult the Technical Manual.

```
model<-e2e_read("North_Sea", "2003-2013",silent=TRUE)
str(model,max.level=2)                    # View the top 2 levels of input list object
List of 2
 $ setup:List of 7
  ..$ read.only    : logi TRUE
  ..$ model.name   : chr "North_Sea"
  ..$ model.variant: chr "2003-2013"
  ..$ model.ident  : chr "base"
  ..$ model.subdir : chr ""
  ..$ model.path   : chr "C:/Users/ais04103/OneDrive - University of Strathclyde/Documents/R/win-library
  ..$ resultsdir   : chr "C:\\Users\\ais04103\\AppData\\Local\\Temp\\RtmpiySPHQ/North_Sea/2003-2013/"
 $ data :List of 8
  ..$ fixed.parameters   :List of 51
  ..$ fitted.parameters  :List of 171
  ..$ physical.parameters:List of 63
  ..$ physics.drivers    :'data.frame': 12 obs. of  23 variables:
  ..$ chemistry.drivers  :'data.frame': 12 obs. of  22 variables:
  ..$ biological.events  :'data.frame': 26 obs. of  2 variables:
  ..$ fleet.model        :List of 19
  ..$ initial.state      :List of 403
```

To mine deeper into the model object, for example, to show the fleet model inputs:

```
model<-e2e_read("North_Sea", "2003-2013",silent=TRUE)
model$data$fleet.model
$gear_labels
 [1] "Pelagic_Trawl+Seine"
 [2] "Sandeel+sprat_trawl(Otter30-70mm+TR3)"
 [3] "Longline_mackerel"
 [4] "Beam_Trawl_BT1+BT2"
 [5] "Demersal_Seine"
 [6] "Demersal_Otter_Trawl_TR1"
 [7] "Gill_Nets+Longline_demersal"
 [8] "Beam_Trawl_shrimp"
 [9] "Nephrops_Trawl_TR2"
[10] "Creels"
[11] "Mollusc_Dredge"
[12] "Whaler"
```

```
$gear_codes
 [1] "PTS" "SST" "LLm" "BTf" "DS"  "OT"  "LLd" "BTs" "NT"  "CR"  "MD"  "Wh"

$gear_activity
 [1] 2.17000e-06 4.23000e-06 1.68000e-06 1.15000e-05 1.72000e-08 2.16000e-05
 [7] 7.92000e-06 1.27000e-05 1.72000e-05 2.40000e-05 3.11000e-06 1.27413e-08

$gear_group_rel_power
       pelagic    demersal    migratory     filtben      carnben  carnzoo
1  1.932115e+03  0.89347662 1.077415e+03 0.000000000  0.002671044 0.000000
2  1.013869e+03  2.12168553 3.843438e+01 0.000884645  0.767630355 0.000000
3  1.218981e-02  0.04958782 7.381184e+00 0.000049300  0.025221014 0.000000
4  1.209108e-03 80.89547724 1.613851e-01 0.001074415  0.389642624 0.000000
5  0.000000e+00 10.64039678 1.094104e+01 0.000000000  0.015250346 5.090975
6  8.489664e-01 77.63191616 9.424984e+00 0.001409053  0.699460502 1.496856
7  2.615744e-01  8.66490093 5.601849e-02 0.000892911  0.069121787 0.000000
8  2.478026e-01  5.17516220 2.046550e-02 0.000009520 27.908549830 0.000000
9  6.521403e-03 16.06754050 1.646958e-01 0.000232275  5.723924492 0.137040
10 3.902940e-04  0.02007798 3.983892e-02 0.001443302  1.676332703 0.000000
11 1.231062e-01  0.04814089 7.589362e-03 5.800854733  0.005832199 0.000000
12 0.000000e+00  0.00000000 0.000000e+00 0.000000000  0.000000000 0.000000
       bird     seal        ceta kelp
1  0.175037 0.000000   0.3514017    0
2  0.000000 0.000000   0.0000000    0
3  0.085223 0.000000   0.0000000    0
4  0.000000 0.000000   0.0000000    0
5  0.000000 0.000000   0.0000000    0
6  0.000000 0.000000   0.0000000    0
7  0.010659 0.749952   1.8122620    0
8  0.000000 0.000000   0.0000000    0
9  0.000000 0.000000   0.0000000    0
10 0.000000 0.000000   0.0326668    0
11 0.000000 0.000000   0.0000000    0
12 0.000000 0.000000 127.4624000    0

$gear_group_discard
       pelagic    demersal   migratory     filtben    carnben carnzoo bird seal
1  0.002940862 0.111955218 0.052261365 0.000000000 0.00000000       0    1    1
2  0.000117639 0.537722720 0.008721464 0.000409807 0.17253984       0    1    1
3  0.000000000 0.004905382 0.000000000 0.000000000 0.00002210       0    1    1
4  0.000000000 0.544527877 0.670642427 0.013789586 0.14986397       0    1    1
5  0.000000000 0.000000000 0.000000000 0.000000000 0.00000000       0    1    1
6  0.125586035 0.473864510 0.769790851 0.021689801 0.31255895       0    1    1
7  0.000013000 0.027499596 0.231375133 0.000488107 0.49755816       0    1    1
8  0.422012971 0.970253908 0.365643786 0.000000000 0.57604018       0    1    1
9  0.379740296 0.684241519 0.439291664 0.000000000 0.06020828       0    1    1
10 0.056057846 0.008395374 0.000584674 0.000000000 0.00000177       0    1    1
11 0.000000000 0.476891130 0.000000000 0.005634317 0.00000000       0    1    1
12 0.000000000 0.000000000 0.000000000 0.000000000 0.00000000       0    0    0
   ceta kelp
1     1    0
2     1    0
3     1    0
```

```
4     1    0
5     1    0
6     1    0
7     1    0
8     1    0
9     1    0
10    1    0
11    1    0
12    0    0

$gear_group_gutting
   pelagic demersal migratory filtben carnben carnzoo bird seal ceta kelp
1        0      0.0         0       0     0.0       0    0    0    0    0
2        0      0.0         0       0     0.0       0    0    0    0    0
3        0      0.0         0       0     0.0       0    0    0    0    0
4        0      0.5         0       0     0.0       0    0    0    0    0
5        0      0.5         0       0     0.0       0    0    0    0    0
6        0      0.5         0       0     0.0       0    0    0    0    0
7        0      0.0         0       0     0.0       0    0    0    0    0
8        0      0.0         0       0     0.0       0    0    0    0    0
9        0      0.0         0       0     0.8       0    0    0    0    0
10       0      0.0         0       0     0.0       0    0    0    0    0
11       0      0.0         0       0     0.0       0    0    0    0    0
12       0      0.0         0       0     0.0       0    0    0    0    0

$gear_ploughing_rate
 [1]  0.0  8.8  0.0 54.1 22.4 17.1  0.0 13.5 78.9  0.0 22.4  0.0

$gear_habitat_activity
            s0          s1         s2          s3          d0         d1
1  0.002306225 0.022318976 0.18565659 0.056603194 0.004080129 0.36177475
2  0.001633992 0.041936670 0.18845895 0.099262077 0.003321046 0.22028780
3  0.002585154 0.019179084 0.13339978 0.004641906 0.002198504 0.04811572
4  0.000000000 0.010312589 0.55581385 0.051585249 0.000000000 0.15791381
5  0.010771788 0.041737583 0.24808350 0.039262834 0.015696502 0.23161262
6  0.003831778 0.005541356 0.07911069 0.086768094 0.009586108 0.30495406
7  0.017572358 0.011517676 0.38187595 0.306710860 0.009678597 0.05754839
8  0.000000000 0.041352126 0.86793552 0.061227227 0.000000000 0.01897104
9  0.000000000 0.113741270 0.00000000 0.000000000 0.000000000 0.88625873
10 0.014753902 0.048364874 0.18995648 0.166975262 0.023811479 0.05029062
11 0.000000000 0.011387932 0.25645298 0.092935228 0.000000000 0.05084907
12 0.000000000 0.000000000 0.00000000 0.000000000 0.000000000 0.20000000
           d2          d3
1  0.33238459 0.034875556
2  0.42221147 0.022887992
3  0.78622360 0.003656248
4  0.21831786 0.006056644
5  0.36953715 0.043298018
6  0.47351519 0.036692729
7  0.20548174 0.009614425
8  0.01021216 0.000301922
9  0.00000000 0.000000000
10 0.42136843 0.084478943
```

```
11 0.49148021 0.096894590
12 0.80000000 0.000000000

$HRscale_vector
PF_HR_scale DF_HR_scale MF_HR_scale SB_HR_scale CB_HR_Scale CZ_HR_Scale
 0.06971091  0.07548334  0.36956660 12.66202800  0.63005471 15.71639518
BD_HR_Scale SL_HR_Scale CT_HR_Scale KP_HR_Scale
 2.71790780  2.78154519 18.10813690  1.00000000

$HRscale_vector_multiplier
 [1] 1 1 1 1 1 1 1 1 1 1

$offal_prop_live_weight
[1] 0.1

$gear_mult
 [1] 1 1 1 1 1 1 1 1 1 1 1 1

$quota_nonquota_parms_vector
[1] 0.16 0.07 0.67 0.02 0.40 0.02

$DFsize_SWITCH
[1] 0

$DFdiscard_SWITCH
[1] 1

$plough_thickness
[1] 0.05

$plough_depth_vector
[1] 0.00000000 0.43160013 0.05820558 0.04545455 0.00000000 0.50000000 0.07243667
[8] 0.04545455

$BSmort_gear
[1] 0.2

$BCmort_gear
[1] 0.2
```

Below are two examples of code to configure and run scenario cases of the the 2003-2013 North Sea model provided with the package. The first example runs 2003-2013 North Sea model as a baseline, and then adds 2 deg-C to the temperatures in all spatial zones and reruns the model. The second example iterates through seven levels of demersal fish harvest ratio ranging from 0 to 3-times the value in the 1970-1999 model (in steps of 0.5-times). Note that the changes in harvest ratio are not accompanied by any changes in fishing gear activity. The package includes a dedicated function for performing this task and generating plots (**e2e_run_ycurve()**.

NOTE - The model is run for only 10 years in Example 1 to create the output shown here - in an actual working example it would be recommended to run for at least 40 years in each case to ensure that the model is close to a stationary state by the end of each run.

Model scenario Example 1

```
# Example of code to run a baseline case of the North Sea model with 2003-2013 conditions,
# and then edit the model list object to create a scenario run with the temperature in
# all zones increased by 2 deg-C
#-----------------------------------------------------------
# Read the embedded North Sea 2003-2013 model and assign an identifier for the results
baseTemp_model<-e2e_read("North_Sea", "2003-2013",model.ident="baseTemp",silent=TRUE)
# Run the model for 10 years and save the results to a named list object
baseTemp_results<-e2e_run(baseTemp_model,nyears=10)
# Visualize the output from the run (should show a repeating annual cycle with no trend)
e2e_plot_ts(baseTemp_model,baseTemp_results,"ECO")
```

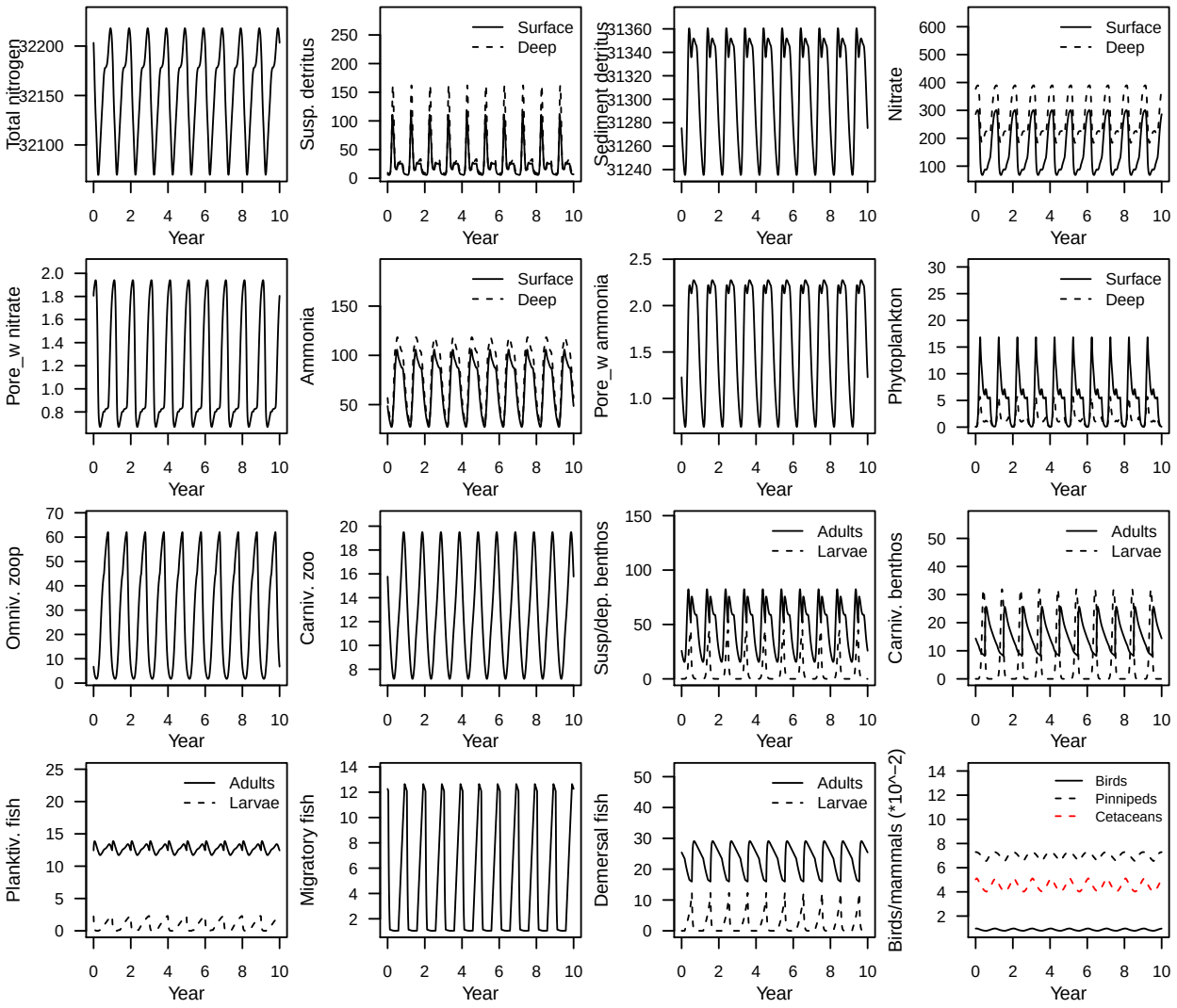


Figure 6. Baseline model run for Example 1 - 2003-2013 conditions in the North Sea

```
dT<-2    # temperature increase to add to all temperatures in the model drivers
# copy the baseline model list object to a new model list
plusTC_model<-baseTemp_model
# add temperature increase to upper layer offshore temperatures
plusTC_model$data$physics.drivers$so_temp <- baseTemp_model$data$physics.drivers$so_temp+dT
# add temperature increase to inshore temperatures
```

```
plusTC_model$data$physics.drivers$si_temp <- baseTemp_model$data$physics.drivers$si_temp+dT
# add temperature increase to lower layer offshore temperatures
plusTC_model$data$physics.drivers$d_temp  <- baseTemp_model$data$physics.drivers$d_temp+dT
# Assign a unique identifier for the .csv outputs
plusTC_model$setup$model.ident <- "baseTemp_plusTC"
# Run the model for 10 years and save the results to a named list object
plusTC_results<-e2e_run(plusTC_model,nyears=10)
# Visualize the output from the run (should show trends in outputs due to change in T)
e2e_plot_ts(plusTC_model,plusTC_results,"ECO")
```

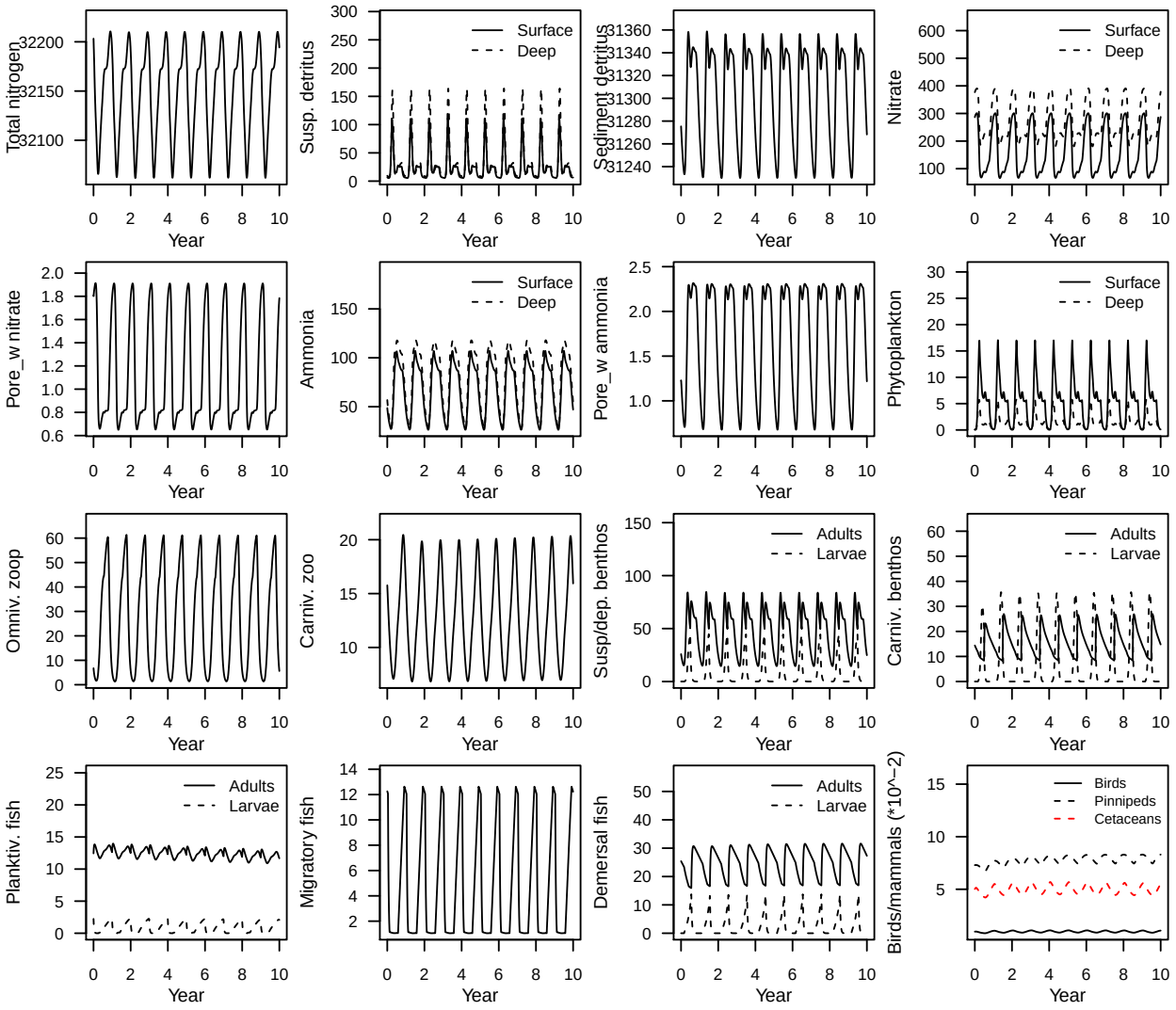


Figure 7. Scenario model run for Example 1 - 2003-2013 plus 2°C conditions in the North Sea.

Model scenario Example 2

```
# Example of code to loop through a set of seven levels of demersal fish harvest ratio
# ranging from 0 to 3-times the baseline value for the 1970-1999 North Sea model. The
# .csv outputs for each level are saved to unique filenames in a temporary folder since
# results.path is not set in the ee2_read() functioncall. The results list-object
# is discarded each time. The baseline case is the third level (HScale=1). Each level is run
```

```
# for 40 years.
#-------------------------------------------------------
model<-e2e_read("North_Sea", "1970-1999",silent=TRUE)
for(i in 1:7) {                                  # Start loop through seven model scenarios
  HScale <- (i-1)*0.5                            # HScale becomes 0 to 3 in steps of 0.5
      # Assign a unique identifier for each set of .csv outputs
  model$setup$model.ident <- paste("Dem_HR_scale_",HScale,sep="")
      # Set the demersal fish harvest ratio multipler to HScale
  model$data$fleet.model$HRscale_vector_multiplier[2] <- HScale
  results<-e2e_run(model,nyears=40,         # Run model, overwrite previous list-object,
                   csv.output=TRUE)             # save the .csv results
  cat("Harvest Ratio scaling = ",HScale,"\n")  # Print a screen message to monitor progress
  e2e_plot_ts(model,results,"ECO")          # Visualize the output from each run
}                            # End loop through scenarios
```

4.2 Writing your own code to work with the model output

The first step to working with the model outputs is understanding their structure. This is described in the following section, but more details are available in a Technical Manual.

The outputs from running the basic model are an R list object (**results** in the examples above) and a set of csv files. The list object contains 7 primary data structures. The csv file outputs are generated in the **/results** folder, either in the user home workspace by default, or in a workspace defined by the arguments of the **e2e_read()** function. Each csv file has a generic file name plus an alpha-numeric identifier (**model.ident**) assigned by the user as an argument of the **e2e_read()** function (default = "base"). In the following descriptions this is represented by "TEXT". The csv files replicate elements of the data contained in the R list object generated by **e2e_run()**. Their purpose is to preserve the most useful outputs for future use e.g. in comparisons of baseline and scenario model runs.

An abbreviated view of the contents of the results list object is shown below.

```
model<-e2e_read("North_Sea", "2003-2013",silent=TRUE)
results <- e2e_run(model, nyears=3)          # Run the model for 3 years
str(results,max.level=1)                     # View the contents of the output list object
List of 7
 $ build               :List of 4
 $ output              :'data.frame':   1081 obs. of  404 variables:
 $ aggregates          :List of 130
 $ fleet.output        :List of 7
 $ total.annual.catch  :List of 2
 $ annual.catch.by.gear:List of 2
 $ final.year.outputs  :List of 27
```

A brief description of the contents of each of the elements of the results object, and correspondence with csv output files is shown below. Full details are provided in the Technical Manual.

The object **build** is a list containing a copy of the model input data imported from the **model** object created by **e2e_read()**. This is included in the **results** object since some of these data are required for post-processing of results, e.g. to convert state variable mass to area or volume densities (see examples later):

```
str(results$build,max.level=1)
List of 4
 $ model.parameters: Named num [1:585] 30 50.04 24.16 1 0.25 ...
  ..- attr(*, "names")= chr [1:585] "thik_so" "thik_d" "thik_si" "thik_b" ...
 $ run             :List of 7
 $ drivers         :List of 51
```

```
 $ forcings        :List of 53
```

The object **output** is a dataframe containing the raw daily-interval output from the ODE solving computations on the masses of all of the state varaiables. This dataframe is the raw material for all subsequent post-processing of the model results. The example below views just the top and bottom few rows, and the first 6 columns of the dataframe. The prefix x_ in front of the detritus and other terms signifies a sediment property, with the extension _s1, _s2, or _s3 indicating inshore sediment habitat, and _d1, _d2, or _d3 indicating offshore sediment habitat. For water column components the extension _so, _d, and _si indicate offshore upper and lower, and inshore zones/layers.

```
head(results$output[,1:6])
  time detritus_so detritus_d x_detritus_s1 x_detritus_s2 x_detritus_s3
1    0    5.220919   9.623444     1.0235378      17.02020      4.675414
2    1    5.152036   9.465099     1.0155082      16.87355      4.637773
3    2    5.079553   9.327179     1.0075133      16.72756      4.600269
4    3    5.009636   9.202132     0.9995546      16.58226      4.562915
5    4    4.943984   9.087204     0.9916338      16.43772      4.525724
6    5    4.882914   8.981162     0.9837528      16.29397      4.488707
tail(results$output[,1:6])
     time detritus_so detritus_d x_detritus_s1 x_detritus_s2 x_detritus_s3
1076 1075    5.229323   9.617508      1.069287      17.85918      4.884862
1077 1076    5.225330   9.614772      1.059991      17.68841      4.842342
1078 1077    5.222572   9.614131      1.050767      17.51912      4.800133
1079 1078    5.220968   9.615449      1.041617      17.35132      4.758240
1080 1079    5.220441   9.618595      1.032541      17.18502      4.716668
1081 1080    5.220923   9.623452      1.023540      17.02023      4.675421
```

The object **aggregates** is a list containing vectors of 124 derived model variable which are aggregations of selected columns in the raw model output in the **output** object. Aggregations are created by summing, for example, across inshore and offshore values of the same food web component to produce a whole-domain value. Each numeric vector has the same length as the number of rows in dataframe **results$output**, in other words a value for each daily output step. In addition, the **aggregates** list contains six single numeric values of zonal layer thicknesses and sediment porosities needed to convert aggregated masses into mass densities (m^{-2} or m^{-3}).

The example below views the names of just the first few variables in the **aggregates** list, extracts two variables and makes plot of their time-series. More details are provided in the Technical Manual.

```
# List the names of the first 40 vectors in the results$aggregates list:
names(results$aggregates[1:40])
 [1] "totalN"         "totalN_o"       "totalN_i"       "x_detritus"
 [5] "x_detritus_o"   "x_detritus_i"   "corpse"         "corpse_o"
 [9] "corpse_i"       "x_ammonia"      "x_ammonia_o"    "x_ammonia_i"
[13] "x_nitrate"      "x_nitrate_o"    "x_nitrate_i"    "s_detritus"
[17] "s_ammonia"      "s_nitrate"      "s_phyt"         "benthslar"
[21] "benthclar"      "benths"         "benthc"         "discard"
[25] "omni"           "carn"           "fishp"          "fishd"
[29] "fishm"          "bird"           "seal"           "ceta"
[33] "fishplar"       "fishdlar"       "NNCP"           "netpprod"
[37] "fluxwcamm_phyt" "fluxwcnit_phyt" "phytgrossprod"  "omnigrossprod"
```

```
# Plot the time series of the sedient nitrate mass in the inshore and offshore zones
# i.e. aggregated across the sediment types in each zone.
# Note that this is the mass not the porewater concentration so uncorrected for
# zonal area or sediment porosity or layer thickness.
plot(seq(1,nrow(results$output)), results$aggregates$x_nitrate_o, type="l",col="black",
```

```
     ylim=( c( 0,(1.5*max(results$aggregates$x_nitrate_o)) ) ),
     xlab="Days", ylab="Sediment nitrate mass (mMN)")
lines(seq(1,nrow(results$output)), results$aggregates$x_nitrate_i, type="l",col="red")
legend("topleft",  bg="transparent", c("offshore","inshore"), lty=c(1,1),col=c("black","red"))
```

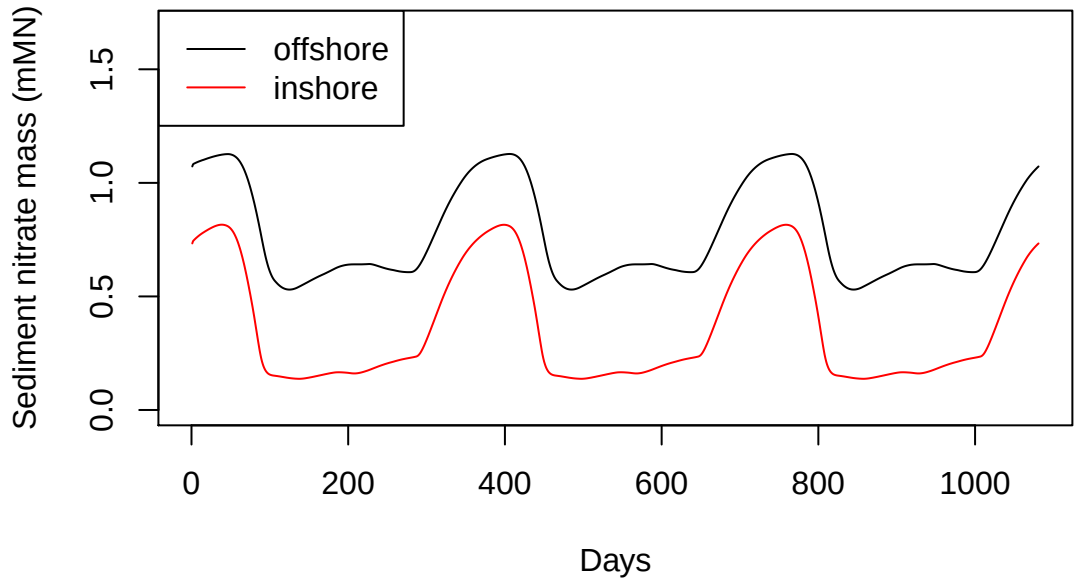


Figure 8. Plot of the time series of sediment nitrate mass aggregated over the offshore and inshore zones.

The object **fleet.output** is a list of dataframes containing outputs from the fleet model which are either piped into the ecology model, or saved internally for use in post-processing the model outputrs - in particular for disaggregating the total catch of each good web guild into the fractions attributable to each fishing gear. This post processing is carried out automatically by the **e2e_run()** function (see below) so under most circumatances the user would not require access to these internal data:

```
str(results$fleet.output,max.level=1)
List of 7
 $ fleet_vector                              : Named num [1:250] 0.000712 0.000553 0.000308 0.000202 0.
  ..- attr(*, "names")= chr [1:250] "F_inshore_pelagic" "F_offshore_pelagic" "F_inshore_demersal" "F_of
 $ offshore_gear_group_prop_effort           :'data.frame': 12 obs. of  10 variables:
 $ inshore_gear_group_prop_effort            :'data.frame': 12 obs. of  10 variables:
 $ offshore_gear_to_region_discard_rate_ratio:'data.frame': 12 obs. of  10 variables:
 $ inshore_gear_to_region_discard_rate_ratio :'data.frame': 12 obs. of  10 variables:
 $ offshore_gear_group_props                 :'data.frame': 3 obs. of  241 variables:
 $ inshore_gear_group_props                  :'data.frame': 3 obs. of  241 variables:
```

The object **total.annual.catch** is a list comprising two dataframes - **offshore_annual_group_gear_land_disc** and **inshore_annual_group_gear_land_disc**. These dataframes contain the annual landings and discards of each harvestable guild in the model for each year of the model run (units mMN.y^{-1}, from a model domain of 1m^2 sea surface area). These data correspond to the files **model_offshore_annual_landings_discards-TEXT.csv** and **model_inshore_annual_landings_discards-TEXT.csv** which are generated in the **/results** folder.

```
str(results$total.annual.catch,max.level=1)
List of 2
 $ offshore_annual_group_land_disc:'data.frame':    3 obs. of  23 variables:
 $ inshore_annual_group_land_disc :'data.frame':    3 obs. of  23 variables:
```

The object **annual.catch.by.gear** is a list comprising two dataframes - **offshore_annual_group_land_disc** and **inshore_annual_group_land_disc**. These dataframes contain the same data as in **total.annual.catch** but disaggregataed into the fractions attributable to the (up to) 12 gears. The data are not output to csv files.

```
str(results$annual.catch.by.gear,max.level=1)
List of 2
 $ offshore_annual_group_gear_land_disc:'data.frame':   3 obs. of  265 variables:
 $ inshore_annual_group_gear_land_disc :'data.frame':   3 obs. of  265 variables:
```

The object **final.year.output** is a list of arrays and dataframes containing a wide range of properties derived from the final year of the model run. The majority of these are also output as csv files in the **/results** folder.

```
str(results$final.year.output,max.level=1)
List of 27
 $ inshore_catchmat               : num [1:11, 1:12] 1.35e-01 2.73e-04 8.59e-06 1.62e-02 0.00 ...
  ..- attr(*, "dimnames")=List of 2
 $ inshore_discmat                : num [1:11, 1:12] 3.96e-04 3.07e-05 7.79e-07 8.47e-04 0.00 ...
  ..- attr(*, "dimnames")=List of 2
 $ inshore_landmat                : num [1:11, 1:12] 1.34e-01 2.42e-04 7.81e-06 1.54e-02 0.00 ...
  ..- attr(*, "dimnames")=List of 2
 $ offshore_catchmat              : num [1:11, 1:12] 1.013477 0.000793 0.000025 0.949236 0 ...
  ..- attr(*, "dimnames")=List of 2
 $ offshore_landmat               : num [1:11, 1:12] 1.01 7.03e-04 2.27e-05 9.00e-01 0.00 ...
  ..- attr(*, "dimnames")=List of 2
 $ offshore_discmat               : num [1:11, 1:12] 2.98e-03 8.93e-05 2.27e-06 4.96e-02 0.00 ...
  ..- attr(*, "dimnames")=List of 2
 $ monthly.averages               :'data.frame':    12 obs. of  9 variables:
 $ mass_results_inshore           :'data.frame':    55 obs. of  3 variables:
 $ maxmass_results_inshore        :'data.frame':    55 obs. of  3 variables:
 $ minmass_results_inshore        :'data.frame':    55 obs. of  3 variables:
 $ annual_flux_results_inshore    :'data.frame':    240 obs. of  3 variables:
 $ mass_results_offshore          :'data.frame':    55 obs. of  3 variables:
 $ maxmass_results_offshore       :'data.frame':    55 obs. of  3 variables:
 $ minmass_results_offshore       :'data.frame':    55 obs. of  3 variables:
 $ annual_flux_results_offshore   :'data.frame':    240 obs. of  3 variables:
 $ mass_results_wholedomain       :'data.frame':    55 obs. of  3 variables:
 $ maxmass_results_wholedomain    :'data.frame':    55 obs. of  3 variables:
 $ minmass_results_wholedomain    :'data.frame':    55 obs. of  3 variables:
 $ annual_flux_results_wholedomain:'data.frame':    240 obs. of  3 variables:
 $ flow_matrix_all_fluxes         :'data.frame':    30 obs. of  32 variables:
 $ flow_matrix_excl_spawn_recruit :'data.frame':    30 obs. of  32 variables:
 $ NetworkIndexResults            :'data.frame':    99 obs. of  1 variable:
 $ annual.target.data             :'data.frame':    84 obs. of  9 variables:
 $ monthly.target.data            :'data.frame':    108 obs. of  9 variables:
 $ annual_obj                     : num 0.496
 $ partial_chi                    :'data.frame':    85 obs. of  1 variable:
 $ opt_results                    :'data.frame':    84 obs. of  8 variables:
```

The first six entries in the **final.year.output** list object are arrays which all have the same format

- the catch, landings or discard mass of each guild (rows) by each gear (columns) in a given spatial zone, over the final year (mMN.y^{-1} from a model domain of 1m^2 surface area). These data correspond with csv files in the **/results** folder, for example: **final.year.output$inshore_catchmat** becomes **INSHORE_catchcomposition_by_gear-TEXT.csv**. For example, the first two columns of the array containing inshore catch data are given by:

```
# List the first six entries
str(results$final.year.output[1:6],max.level=1)
List of 6
 $ inshore_catchmat : num [1:11, 1:12] 1.35e-01 2.73e-04 8.59e-06 1.62e-02 0.00 ...
  ..- attr(*, "dimnames")=List of 2
 $ inshore_discmat  : num [1:11, 1:12] 3.96e-04 3.07e-05 7.79e-07 8.47e-04 0.00 ...
  ..- attr(*, "dimnames")=List of 2
 $ inshore_landmat  : num [1:11, 1:12] 1.34e-01 2.42e-04 7.81e-06 1.54e-02 0.00 ...
  ..- attr(*, "dimnames")=List of 2
 $ offshore_catchmat: num [1:11, 1:12] 1.013477 0.000793 0.000025 0.949236 0 ...
  ..- attr(*, "dimnames")=List of 2
 $ offshore_landmat : num [1:11, 1:12] 1.01 7.03e-04 2.27e-05 9.00e-01 0.00 ...
  ..- attr(*, "dimnames")=List of 2
 $ offshore_discmat : num [1:11, 1:12] 2.98e-03 8.93e-05 2.27e-06 4.96e-02 0.00 ...
  ..- attr(*, "dimnames")=List of 2
#
# View the first two column of the inshore_catchmat array
results$final.year.output$inshore_catchmat[,1:2]
                           Pelagic_Trawl+Seine
Planktivorous fish                  1.345671e-01
Quota-limited demersal fish         2.725820e-04
Non-quota demersal fish             8.591024e-06
Migratory fish                      1.621141e-02
Susp/deposit feeding benthos        0.000000e+00
Carn/scavenge feeding benthos       5.468340e-07
Pelagic invertebrates               0.000000e+00
Birds                               2.591004e-08
Pinnipeds                           0.000000e+00
Cetaceans                           0.000000e+00
Macrophytes                         0.000000e+00
                           Sandeel+sprat_trawl(Otter30-70mm+TR3)
Planktivorous fish                                  1.708657e-01
Quota-limited demersal fish                         1.566254e-03
Non-quota demersal fish                             4.936394e-05
Migratory fish                                      1.399345e-03
Susp/deposit feeding benthos                        5.461389e-06
Carn/scavenge feeding benthos                       3.802714e-04
Pelagic invertebrates                               0.000000e+00
Birds                                               0.000000e+00
Pinnipeds                                           0.000000e+00
Cetaceans                                           0.000000e+00
Macrophytes                                         0.000000e+00
```

The object **final.year.output$monthly_averages** is a dataframe of monthly averaged nutrient and plankton data (units: mMN.m^{-3}, except for chlorophyll: mg.m^{-3}) corresponding to the observational data available in the **/Target** folder of model stup data. Chlorophyll is estimated assuming Redfield C:N (molar ratio 106:16), and a carbon:chlorophyll weight ratio of 20. The data correspnd to the csv output file **model_monthlyresults-TEXT.csv**. For example, the first two columns of the array are given by:

```
results$final.year.output$monthly.averages[,1:2]
   surfnitratemMm3 deepnitratemMm3
1        10.299030       10.207210
2        10.413101       10.330094
3         7.602763        8.718036
4         3.037916        5.425206
5         2.452920        5.017180
6         2.878804        5.508646
7         3.056570        5.923631
8         3.308574        6.001262
9         4.018523        5.824872
10        4.915528        5.974877
11        7.271521        7.712605
12        9.335612        9.432706
```

The next four entries in the **final.year.output** list object are dataframes containing annual averages, maxima and minima of each state variable in the inshore zone (mMN), and a set of annually integratred flux terms (mMN.y^{-1}) during the final year. The fluxes include both physical flows due to advection, mixing, and inflows and outflows, and ecological fluxes between state variables. The food web fluxes are not calculated for the inshore zone, but are provided in other data frames for the model domain as a whole. In addition the dataframe contains the zonal area proportions and layer thicknesses needed to convert the data to mass or flux densities (m^{-2} or m^{-3}). The formats of the average, maxima and minima files are the same. The data correspond to csv files output to the */results* folder with names: **INSHORE_model_anav_biomass-TEXT.csv**, **INSHORE_model_maximum_biomass-TEXT.csv**, **INSHORE_model_minimum_biomass-TEXT.csv**, **INSHORE_model_annualresults-TEXT.csv**. The dataframes contain 3 columns: the numeric values, units and a descriptive text field. For example, list these dataframes in the list, and view the top few rows of the annual average dataframe, and the annual fluxes dataframe:

```
str(results$final.year.output[8:11],max.level=1)
List of 4
 $ mass_results_inshore       :'data.frame':    55 obs. of  3 variables:
 $ maxmass_results_inshore    :'data.frame':    55 obs. of  3 variables:
 $ minmass_results_inshore    :'data.frame':    55 obs. of  3 variables:
 $ annual_flux_results_inshore:'data.frame':    240 obs. of  3 variables:
#
head(results$final.year.output$mass_results_inshore)
  Model_annual_mean                                Units
1      4.475829e+00 mMN_in_the_whole model_domain_(1m2)
2                NA mMN_in_the_whole model_domain_(1m2)
3      1.221649e+04 mMN_in_the_whole model_domain_(1m2)
4      1.218173e+04 mMN_in_the_whole model_domain_(1m2)
5      5.159798e-02 mMN_in_the_whole model_domain_(1m2)
6      1.248255e+00 mMN_in_the_whole model_domain_(1m2)
                            Description
1                   Surface_layer_detritus
2                      Deep_layer_detritus
3 Sediment_labile_plus_refractory_detritus
4             Sediment_refractory_detritus
5                         Fishery_discards
6                                  Corpses
#
head(results$final.year.output$annual_flux_results_inshore)
  Model_annual_flux                          Units          Description
1        137.637476 mMN/whole_model_domain_(1m2)/y           DIN_inflow
```

```
2         95.026438 mMN/whole_model_domain_(1m2)/y          DIN_outflow
3         69.075865 mMN/whole_model_domain_(1m2)/y   Particulate_inflow
4          9.793046 mMN/whole_model_domain_(1m2)/y  Particulate_outflow
5         17.669722 mMN/whole_model_domain_(1m2)/y Atmosphere_DIN_input
6        104.700823 mMN/whole_model_domain_(1m2)/y     River_DIN_inflow
```

The next four entries in the **final.year.output** list object are dataframes containing annual averages, maxima and minima of each state variable, and annual fluxes in the offshore zone (mMN) during the final year, exactly corresponding to the previous four entries for the inshore zone. As for the inshore zone, the data are output to corresponding csv files.

```
str(results$final.year.output[12:15],max.level=1)
List of 4
 $ mass_results_offshore       :'data.frame':   55 obs. of  3 variables:
 $ maxmass_results_offshore    :'data.frame':   55 obs. of  3 variables:
 $ minmass_results_offshore    :'data.frame':   55 obs. of  3 variables:
 $ annual_flux_results_offshore:'data.frame':   240 obs. of  3 variables:
```

The next four entries in the **final.year.output** list object are dataframes containing annual averages, maxima and minima of each state variable, and annual fluxes for the model domain as a whole (mMN) during the final year, exactly corresponding to the previous four entries for the offshore zone, except that in this case all of the flows between state variables in the food web are evaluated. In following outputs these flows are assembled into the flow matrix format required by the NetIndices package to calculate network indices. As for the offshore zone, the data are output to corresponding csv files.

```
str(results$final.year.output[16:19],max.level=1)
List of 4
 $ mass_results_wholedomain       :'data.frame':    55 obs. of  3 variables:
 $ maxmass_results_wholedomain    :'data.frame':    55 obs. of  3 variables:
 $ minmass_results_wholedomain    :'data.frame':    55 obs. of  3 variables:
 $ annual_flux_results_wholedomain:'data.frame':    240 obs. of  3 variables:
```

The next three entries in the **final.year.output** list object are dataframes containing the flow matrices required for input to the **NetIndices** package, and the outputs from NetIndices. Two version of the flow matrix are included - one which includes spawning and recruitment flows - from adults to larvae and larvae to adults, and one in which these flows are excluded. In each case the rows names of the matrix indicate the "flow-from", and the columns indicatethe "flow-to". All three of these data sets are exported to csv files in the **/results** folder: **flow_matrix_all_fluxes-TEXT.csv**, **flow_matrix_excl_spawn_recruit-TEXT.csv**, and **Network_indices_output-TEXT.csv**.

```
str(results$final.year.output[20:22],max.level=1)
List of 3
 $ flow_matrix_all_fluxes        :'data.frame': 30 obs. of  32 variables:
 $ flow_matrix_excl_spawn_recruit:'data.frame': 30 obs. of  32 variables:
 $ NetworkIndexResults           :'data.frame': 99 obs. of  1 variable:
#
#Show the first 5 rows and columns of the matrix of all flows:
results$final.year.output$flow_matrix_all_fluxes[1:5,1:5]
             wcammonia sedammonia wcnitrate  sednitrate wcdetritus
wcammonia    0.0000000          0  1085.117   0.0000000          0
sedammonia 380.0737013          0     0.000   0.0372424          0
wcnitrate    0.0000000          0     0.000 205.3809451          0
sednitrate   0.0000000          0     0.000   0.0000000          0
wcdetritus   0.3577377          0     0.000   0.0000000          0
#
#Show the first few rows of the dataframe containing outputs from the NetIndices package:
```

```
head(results$final.year.output$NetworkIndexResults)
                         NetworkData
wcammonia_trophiclevel             1
sedammonia_trophiclevel            1
wcnitrate_trophiclevel             1
sednitrate_trophiclevel            1
wcdetritus_trophiclevel            1
seddetritus_trophiclevel           1
```

The next two entries in the **final.year.output** list object are dataframes containing annual observational data on the state of the ecosystem, and the monthly observational data. These are exact copies of the input data from the **/Target** folder of input files, and are included in the **results** object merely for convenience so that they are easily available for comparison with model outputs:

```
str(results$final.year.output[23:24],max.level=1)
List of 2
 $ annual.target.data :'data.frame':    84 obs. of  9 variables:
 $ monthly.target.data:'data.frame':    108 obs. of  9 variables:
```

The final three elements of the **final.year.output** list object are associated with the calulation of the likelihood of the observed annual target data on the state of the ecosystem, given the model drivers and parameters. The first is a single numeric value - the overall likelihood of the observed target data. The second is a dataframe of the partial likelihoods for each aspect of the observational dataset. Finally, a dataframe of the derived model outouts for the final year of the run, exactly corresponding to the set of observational measures. The two dataframes are also output as csv files to the **/results** folder with names **model__likelihood__results-TEXT.csv** and **model__target__annualresults__plus__chi-TEXT.csv**.

```
str(results$final.year.output[25:27],max.level=1)
List of 3
 $ annual_obj : num 0.496
 $ partial_chi:'data.frame':    85 obs. of  1 variable:
 $ opt_results:'data.frame':    84 obs. of  8 variables:
#
# Show the overall likelihood value:
results$final.year.output$annual_obj
[1] 0.4963685
#
#Show the first few rows of the partial likelihood data:
head(results$final.year.output$partial_chi)
             Likelihood
Obs_TAPP     0.16250081
Obs_NP       0.03332576
Obs_KelpP    1.00000000
Obs_OmnizooP 0.72339227
Obs_CarnzooP         NA
Obs_PFishP           NA
#
# Show the first few rows of the dataframe containing the model outputs corresponding to
# the annual observational target data:
head(results$final.year.output$opt_results)
  Annual_measure SD_of_measure Model_data Use1_0          Chi         Name
1        1522.00       150.940 1234.25655      1 1.817072e+00     Obs_TAPP
2         672.76        73.000  482.35941      1 3.401425e+00       Obs_NP
3         600.00       100.000  600.00019      1 1.789660e-12    Obs_KelpP
4         339.60        25.157  319.35514      1 3.238036e-01 Obs_OmnizooP
```

```
5             NA            NA   45.58046      0           NA Obs_CarnzooP
6             NA            NA   13.67865      0           NA   Obs_PFishP
     Units
1 mMN/m2/y
2 mMN/m2/y
3  gC/m2/y
4 mMN/m2/y
5 mMN/m2/y
6 mMN/m2/y
                                                                        Description
1                                                          Annual_total_primary_production
2 Annual_new_production_from_depth_integrated_nitrate_plus_summer_river_and_atmos_inputs
3                                      Annual_WITHIN_FOREST_NET_production_of_macrophytes
4                                             Annual_ominiv_zooplankton_gross_production
5                                        Annual_carnivorous_zooplankton_gross_production
6                                             Annual_planktivorous_fish_gross_production
```

While the **results** object generated by the **e2e_run()** function contains a wide range of derived model outputs, users may also need to generate their own. The following examples illustrate how to extract and process the raw output held in the **results$output** dataframe.

For a set of results created using the **e2e_read()** and **e2e_run()** functions, the raw data from the model run are contained in the object dataframe **results$output**. The first row of the **output** dataframe corresponds to time=0, i.e. the initial conditions passed to the differential equation solver at the start of the run. A calendar year of output corresponds to a time interval of 360 days, i.e. 361 rows of output data (time = 0:360), so the total number of rows of output will be ((360 * nyears)+1), where nyears is the specified length of the model run in years.

Assuming that the **output** dataframe has been extracted from the results object into a new dataframe **out** as follows:

```
out <- results$output
```

Data for any one year specified by Y (Y <= nyears) can be extracted by the R statement:

```
# All the output corresponding to year Y
Y <- 3                    # select year 3
yearY_data <- out[(((Y-1)*360)+1) : ((Y*360)+1),]
#
#List the first few rows and columns of these extracted data
head(yearY_data[,1:6])
    time detritus_so detritus_d x_detritus_s1 x_detritus_s2 x_detritus_s3
721  720    4.990289   9.184494     1.0265906      17.07606      4.686113
722  721    4.964150   9.139898     1.0180237      16.91927      4.646417
723  722    4.937842   9.094908     1.0095455      16.76422      4.607091
724  723    4.911354   9.049509     1.0011556      16.61090      4.568132
725  724    4.884677   9.003687     0.9928533      16.45931      4.529541
726  725    4.857802   8.957430     0.9846381      16.30942      4.491315
```

```
# All the output for a given state variable (denoted by "column_name") in year Y:
Y <- 3                    # select year 3
column_name<- "omni_o"    # Select offshore omnivorous zooplankton
col2get <- which(colnames(out)==column_name)
column_data_yearY <- out[(((Y-1)*360)+1) : ((Y*360)+1),col2get]
```

```
# Calculate the annual average in year Y of the variable extracted in the previous step:
    mean(column_data_yearY)
```

```
[1] 24.52889
```

The flux terms in the **output** dataframe are cumulative over the duration of the model run, beginning at 0 at time = 0. Hence, the time-series of instantaneous flux rates (d^{-1}) is the sequence of increments between successive rows of output. In R this can be generated by, for example:

```
column_name<- "phytgrossprod_o"            # Select for example offshore gross production
                                           # by phytoplankton
col2get <- which(colnames(out)==column_name)
J <- nrow(out)                      # number of rows of data in the "out" dataframe
temp <- rep(0,J)                           # creates a vector to hold temporary data
temp[1: (J-1)] <- out[2 : J, col2get]      # copies data from the "out" into "temp" offset
                                           # by 1 time increment
temp[J] <- 2*(out[J,col2get]) - out[J-1,col2get]   # fills in the terminal value of the "temp"
                                           # vector by extrapolation
rate <- temp - out[,col2get]               # "rate" is a vector of the increments between
                                           # successive time steps

#Plot the original cumulative data and the derived rate
par(mfrow=c(2,1))
par(mar=c(4,4,1,1))
plot(seq(1,length(out[,col2get])),out[,col2get],type="l",xlab="Days",ylab="Cumulative flux")
plot(seq(1,length(rate)),rate,type="l",xlab="Days",ylab="Daily flux")
```

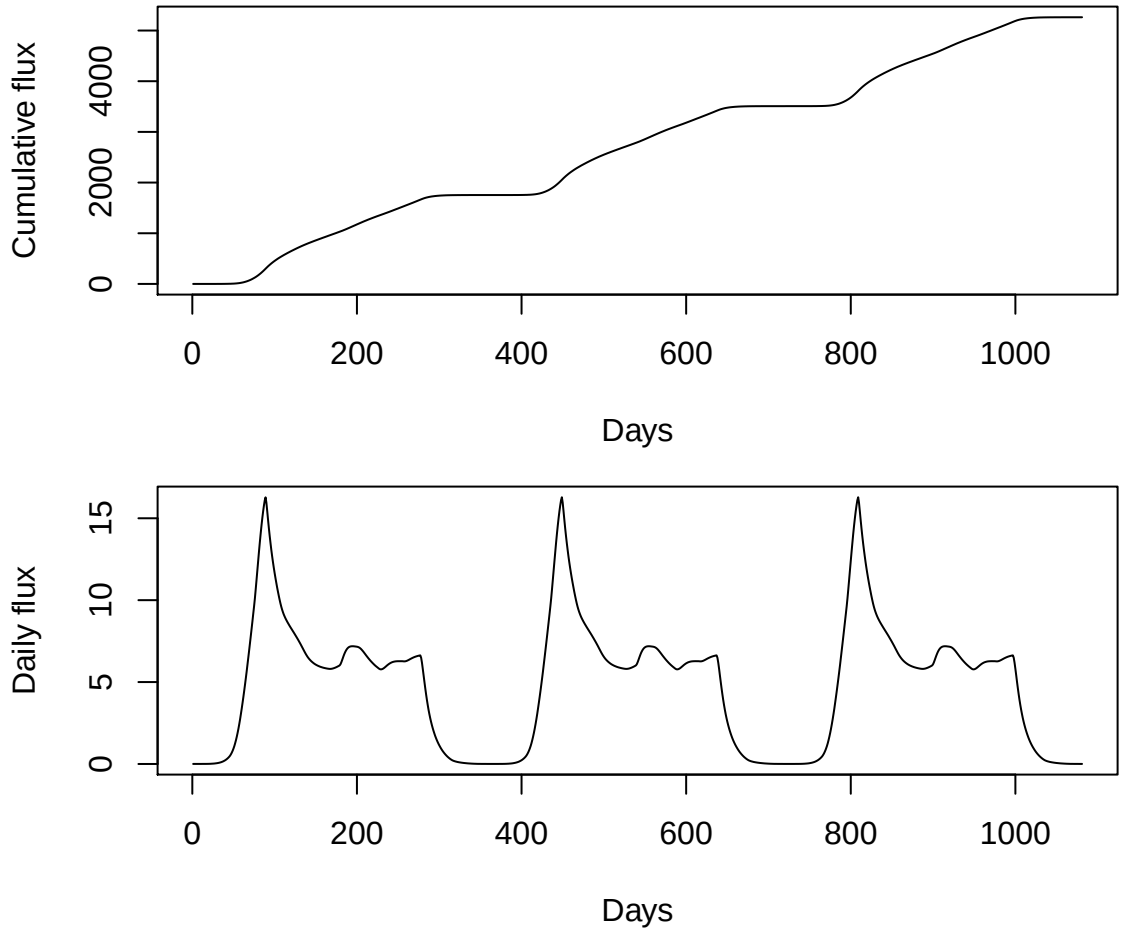


Figure 9. Upper panel: time series of the cumulative phytoplankton gross production as output from the model. Lower panel: same data converted into daily gross production rates.

To obtain the annual integral of a flux variable over a given year, the R statement would be:

```
Y <- 3                                          # select year 3
column_name<- "phytgrossprod_o"                 # Select for example offshore gross production
                                                # by phytoplankton
col2get <- which(colnames(out)==column_name)
out[((Y*360)+1),col2get] - out[(((Y-1)*360)+1),col2get] # integrated flux of the variable
[1] 1754.116
                                                        # denoted by "column_name" over year Y
```

All of the terms in the **output** dataframe are in mass units (mMN or mMC) in the model domain which is is scaled to a sea-surface area of 1 m^2. To express the outputs as area densities in the inshore or offshore zones (e.g. $mMN.m^{-2}$) or layer concentrations (e.g. $mMN.m^{-3}$), the mass values need to be re-scaled to the area-proportion of the relevant zone and/or layer thickness. In addition, terms relating to sediment or sediment porewater require to be scaled by the area-proportions of seabed habitats, sediment layer thicknesses and sediment porosity. These scaling parameters are accessible to the user as elements of the object **build$model.parameters** which forms part of the list-object generated by a model run. Examples of R statements for converting mass outputs into area-densities and layer concentrations are shown below.

```
# Extract the relevant area and volumetric parameters from the $build$model.parameters object
inshore_area         <- as.numeric(results$build$model.parameters["shallowprop"])
offshore_upper_thick <- as.numeric(results$build$model.parameters["thik_so"])
offshore_lower_thick <- as.numeric(results$build$model.parameters["thik_d"])
inshore_sed1_area    <- as.numeric(results$build$model.parameters["area_s1"])
inshore_sed1_thick   <- as.numeric(results$build$model.parameters["thik_x_s1"])
inshore_sed1_poros   <- as.numeric(results$build$model.parameters["porosity_s1"])
#
# Time series of suspended detritus and bacteria concentration in the offshore upper
# layer (mMN.m-3)
detr_bact_conc_so<- out$detritus_so/((1-inshore_area)*offshore_upper_thick)
#
# Time series of the area-density of omnivorous zooplankton in the inshore zone (mMN.m-2)
oz_dens_i<- out$omni_i/inshore_area
#
# Time series of the depth averaged concentration of omnivorous zooplankton in the
# offshore zone (mMN.m-3)
oz_conc_o<- out$omni_o /((1-inshore_area)*(offshore_upper_thick+offshore_lower_thick))
#
# Time series of the porewater concentration of ammonia in inshore sediment
# habitat 1 (mMN.m-3):
xamm_conc_s1<- out$x_ammonia_s1/(inshore_sed1_area*inshore_sed1_thick*inshore_sed1_poros)
#
# Time series of labile detritus & bacterial in inshore sediment habitat 1 as a % of
# sediment dry weight (%gN.g-1):
xdet_conc_s1<- 100*(out$xR_detritus_s1*(14/1000))/(inshore_sed1_area*
               inshore_sed1_thick*(1-inshore_sed1_poros)*2650000 )
# (nitrogen atomic weight = 14 g.mole-1;
# dry sediment density = quartz density = 2.65 x 1e6 g.m-3)
```

5 Parameter estimation

5.1 Background

Three groups of parameters in the combined ecology and fishing fleet model system might be considered as candidates for estimation, depending on the availability of external data:

- Ecology model parameters
- Scaling parameters linking fishing effort to harvest ratios
- Fishing gear activity rates

It is clearly unrealistic to expect to be able to optimize all of these sets of parameters concurrently since there is a risk that some may be confounded. So, the R package provides several options for parameter estimation, each of which optimizes one of these groups of parameters, given knowledge on the other two.

The parameter estimation schemes are configured as a stationary state fitting method, though there is no intrinsic reason why they could not be adapted for time-series fitting. For the most part, the methodology involves likelihood estimation and a simulated annealing scheme using the Metropolis-Hastings iterative algorithm (Bertsimas & Tsitsiklis, 1993; Cerny, 1985; Kirkpatrick et al., 1983).

5.2 Implementation of the Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm implemented with StrathE2E2 automates the acceptance or rejection of proposed new randomly generated sets of model parameters (**Figure 10**). The objective is to maximize the likelihood of the set of observed (target) indices given a vector of model parameter values. Starting with an initial informed-guess parameter vector, the procedure iterates through model runs, and at each iteration the parameters are independently 'jiggled' by proposing new values drawn from gaussian distributions of fixed coefficient of variation (standard deviation divided by the mean) around the current accepted values. There are some special-case parameters in StrathE2E2 which are bounded by known limits, and the preference parameters which require to be renormalized to sum to 1.0 for each predator after the addition of random 'jiggles'.

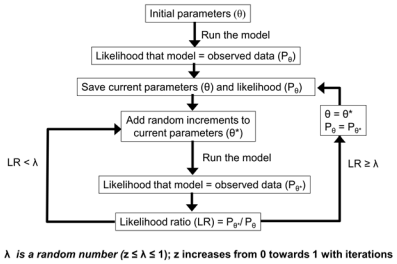


Figure 10. Flow diagram illustrating the operation of the simulated annealing scheme for parameter optimization.

After running the model to a stationary state with the new parameter vector, and extracting simulated versions of the (i) observed ecosystem state indices $(1 < i < I)$ (where the observed indices each have a standard deviation σ_i), an overall error term $(\chi^2_{\theta^*})$ is calculated for the model as follows:

$$\chi^2_{\theta^*} = \frac{\sum_{i=1}^{i=I} \frac{(observed_i - simulated_{\theta^*,i})^2}{2\sigma_i^2}}{I}$$

The likelihood of the overall set of indices $P(observation|\theta^*)$ is then $exp(-\chi^2_{\theta^*})$.

The change in performance of the model due to the jiggling of the proposed parameter set is measured by $(\chi^2_{\theta^*} - \chi^2_{\theta_{current}})$. A simple 'hill-climbing' scheme would accept the proposed parameter set θ^* as a new version of the parameter set $\theta_{current}$ for the next iteration only if the likelihood ratio $LR = exp(\chi^2_{\theta^*} - \chi^2_{\theta_{current}}) > 1$.

Over many iterations the accepted parameter set should then migrate towards a maximum likelihood fit. However, such schemes are notorious for becoming trapped in local optima. Simulated annealing attempts to mitigate this risk by accepting a proportion of instances of proposed sets which produce worse results than current accepted (i.e. $LR < 1$), thereby exploring a wider range of the parameter space. The process is an analogy of the way in which crystals form in a liquid metal as it cools. Taking $T[k]$ to represent the 'temperature' of the system at iteration k, the probability of accepting a proposal as the new current is equal to the modified likelihood ratio:

$$LR_m = exp((\chi^2_{\theta^*} - \chi^2_{\theta_{current}})/T[k])$$

The value of $T[k]$ is allowed to 'cool' as the iterations progress (increasing k). As $T[k]$ decreases towards 0 the system assumes a simple hill-climbing mode (i.e. only vectors θ^* producing likelihood ratios > 1 are accepted). In the early stages, when $T(k) > 0$, a proportion of θ^* producing likelihood ratios < 1 are accepted. Practically, after each model run $\theta[k] = \theta^*$ only if $LR_m \geq \lambda$, where λ is drawn from a uniform random distribution between 0 and 1, otherwise $\theta[k] = \theta[k-1]$ for the next iteration. At each iteration, both the currently accepted and the proposed parameter vectors are saved, along with the resulting likelihood of the observations given the parameters.

The sequence of (P(observation$|\theta^*$)[k]) should converge to a maximum value representing the best attainable fit between the model and the observations given the model structure and driving data. However, the rate at which convergence is achieved, and potentially the converged vector itself, are dependent on the cooling schedule and the magnitude of the jiggling coefficient of variation (cv) used to generate $\theta[k]$. A geometrically decreasing temperature is applied ($T[k] = r.T[k-1]$), and after experimentation with the value of r and cv, the finally adopted values for fitting StrathE2E2 models to achieve consistency of convergence within a sensible number of iterations (<10000) are: r = 0.975, cv = 0.005. The best-fit value of $\theta[k]$ is regarded as being that remaining when no new parameter vectors are accepted within 200 iterations.

5.3 Target data for optimization

The target data for optimizing a particular model should all be located in the **/Models/Variant/Target** folder of the model definition. The individual files names must all contain a standard text string plus a free text string for identification of different versions. The particular version required to be loaded for a given optimization run should be identified inthe **MODEL_SETUP.csv** file in the **/ModelName/VariantName** folder.

In most instances, the target data to which the model is optimized during the simulated annealing process are a set of up to 84 indices of the real-world state of the ecosystem and their associated uncertainties (standard deviations). This file is named **annual_observed*.csv** (where * is a free text field for identification). The ecosystem state indices include annual average biomasses or concentrations, annual integrated production rates of food web components, annual integrated dietary intakes, fishery landings and discards, denitrification rate, and other properties. The file also contains text fields including a description, units, geographic region and time period of applicability, and the origins of the data that were used to generate the index. For each index, the code calculates an equivalent value for the final year of each model run as the basis for the likelihood estimation. Individual indices can be included or excluded from the likelihood calculation by means of a switch (value 1 or 0) in the input file.

One of the available methods for estimating fishing gear activity rates where these are unknown relies on target data concerning the harvest ratios on each guild in each zone of the model. These are contained in a file **zonal_harvest_r*.csv** (where * is a free text field for identification).

Finally, calculation of effort-to-harvest ratio scaling parameter values when activity, fishing power and harvest ratios are all known, requires the target data on harvest ratios. to be located in the file **region_harvest_r*.csv** (where * is a free text field for identification).

5.4 Estimating of ecology model parameters e2e_optimize_eco()

Optimize StrathE2E ecology model parameters to maximize the likelihood of observed ecosystem target data.

Table 17. Arguments of the function **e2e_optimize_eco()**.

Argument	Description
model	R-list object generated by the e2e_read() function which defines the model configuration
nyears	Number of years to run the model in each iteration (default=40)
n_iter	Number of iterations of the model (default=500)
start_temperature	Initial value of the simulated annealing temperature parameter (default=0.5). Suggested values in the range 0.0005 - 5. Higher values increase the probability of rejecting parameter combinations producing an improvement in likelihood
cooling	Rate at which the simulated annealing temperature declines with iterations (default=0.975). Suggested values in the range 0.9 - 0.985
toppredlock	Logical. If TRUE then locks-down the uptake parameters of the birds pinnipeds and cetaceans as these are hard to fit alongside the other parameters (default=TRUE)
quiet	Logical. If TRUE then suppress informational messages at the start of each iteration (default=TRUE)
csv.output	Logical. If TRUE then enable writing of csv output files (default=FALSE)
runtime.plot	Logical. If FALSE then disable runtime plotting of the progress of the run - useful for testing (default=TRUE)

The function **e2e_optimize_eco()** launches a StrathE2E simulated annealing process to find the set of ecology model parameters producing the maximum likelihood of observed target data on the state of the ecosystem, given specified environmental driving data and fishing fleet parameters.

Certain parameters of the ecology model are excluded from the optimization process because they are either confounded with other parameters in some way or there is a clear route to establishing values from external evidence (**Table 18**). All other parameters, defining the prey preferences of all the living components of the model, the uptake and mortality rate coefficients, and the rate coefficients for the biogeochemical processes, are eligible for optimization. A subset of the feeding rate parameters for the top-predators can be optionally excluded from the optimization process if reasonable values can be independently estimated. Separate coefficients of variation can be set for different classes of parameters (preference parameters, maximum uptake rates, half saturation coefficients, microbial rates, density dependent mortality rates) via a setup text file.

Table 18. Background to the parameters of the ecological model which require to be fixed from external evidence and are excluded from the simulated annealing process. These parameters are contained in two files **fixed_consumers*.csv** and **fixed_miscellaneous*.csv** (where * is a free text identifier) in the **/Param** folder of each model definition.

Parameter type	Description
Assimilation efficiencies for each living guild in the model	Proportion of the ingested mass of food that contributes to new body tissue after subtracting defecation and the metabolic costs of digestion and synthesis
Biomass loss rates due to temperature-dependent metabolism for each living resource guild	Proportion of biomass lost to ammonia per day due to non-feeding related metabolism at a given Q10 reference temperature
Q10 values for temperature dependent processes and the Q10 reference temperature	Separate Q10 values for autotrophic uptake of nutrient, heterotrophic feeding, and heterotrophic metabolism based on literature data
Light intensity required to saturate autotrophic nutrient uptake	Light saturation intensity for nutrient uptake, estimated from survey of field and laboratory experiments
Annual weight specific fecundities of fish and benthos guilds	Guild-level values of the annual proportion of biomass which is shed as eggs

Parameter type	Description
Spawning start and end dates for fish and benthos guilds	Day-number (assuming 360-day year) of the onset and end of spawning to be based on such empirical data as are available
Recruitment start and end dates for fish and benthos guilds	Day-number (assuming a 360-day year) of the onset and end of recruitment from the larval stage to the main guild biomass
Extra-domain biomass of migratory fish and the proportion invading the domain each year	Biomass of the wider stock contributing to the transient population in the model domain, extracted from fisheries stock assessments
Start and end dates for the annual invasion and emigration of migratory fish	Day-number (assuming a 360-day year) of the onset and end of immigration and emigration phases
Harvestable biomass density threshold for each resource guild	A limit for each guild below which the harvestable species are assumed to be exhausted. Values based on data from surveys, fisheries and stock assessments
Minimum inedible biomass of carnivorous zooplankton	A minimum edible threshold is set to ensure that the guild as a whole cannot be extirpated by predation

Parameters elligible for fitting are contained in the files **fitted_preference_matrix*.csv**, **fitted_uptake_mort_rates*.csv**, and **fitted_microbiology_others*.csv** (where * is a free text identifier) in the **/Param** folder of each model definition. At the end of an optimization run, provided that **csv.output=TRUE**, the code exports a new set of these three files containing the accepted values for the maximum likelihood model, back to the **/Param** folder with a new text identifier which is set by the **model.ident** argument in the **e2e_read()** function.

Configuration of a simulated annealing run to estimate ecology model parameters must be preceded by a **e2e_read()** function call to load the model setup which includes the initial values for all of the parameters to be fitted. It is important to note that the **models.path** argument in the **e2e_read()** function call needs to point to a user workspace folder, not the default North Sea model provided with the package. This is because the annealing function needs write-access to the model **/Param** folder, but the **/extdata/Models** folder in the package installation is read-only. To use the annealing function on the North Sea model, use the **e2e_copy()** function to make a copy of the North Sea model in the user workspace.

The coefficients of variation for 'jiggling' the parameter values during an optimization run are held in the file **optimize_ecology.csv** in the **/Param/control** folder of each model definition. These can be changed in real-time during a run, to enable exploration of the effects of parameter search patterns.

The function produces a real-time graphical summary of the progress of the fitting procedure, displaying the likelihoods of the proposed and accepted parameter sets at each iteration. The Y-axis (likelihood) range of the real time plot can be varied during the run by editing the setup file **optimize_ecology.csv** at any time during the run.

At the end of the procedure the function returns a list object containing the histories of proposed and accepted parameters and the final accepted parameter values. Optionally (if csv.output=TRUE), new versions of the three ecology model **fitted_*.csv** files are exported to the folder **/Param** of the model version, with a user defined identifier specified by the **model.ident** argument in the **e2e_read()** function. The histories of proposed and accepted parameter combinations are saved as csv files in the current results folder.

In order to use the new fitted parameter values in a subsequent run of the StrathE2E model (using the **e2e_run()** function) it will be necessary to edit the **MODEL_SETUP.csv** file in the relevant **/Models/variant** folder to point to the new files.

```
# Copy the 2003-2013 version of the North Sea model into a user workspace:
    e2e_copy("North_Sea", "2003-2013",
    dest.path="Folder/Models")
#
```

```
# Load the 2003-2013 version of the North Sea model from the user workspace:
    model<-e2e_read(model.name="North_Sea",
                            model.variant="2003-2013",
                            models.path="Folder/Models",  # insert your own path here
                            model.ident="TEST")
# Since results.path is not defined here the histories of proposed and accepted
# parameter values will be written to a temporary folder if csv.output=TRUE.
#
# Quick demonstration of the optimization function in operation:
e2e_optimize_eco(model, nyears=5, n_iter=10, start_temperature=0.5,csv.output=TRUE)
#
# More realistic configuration would be as follows :
e2e_optimize_eco(model, nyears=50, n_iter=1000, start_temperature=1,csv.output=TRUE)
# (WARNING - this will take about 26 hours to run)
#
```

5.5 Estimating of harvest ratio scaling parameters e2e_optimize_hr()

Optimize StrathE2E harvest ratio multipliers to maximize the likelihood of observed ecosystem target data.

Table 19. Arguments of the function **find_harvest_ratio_mult()**.

Argument	Description
model	R-list object generated by the e2e_read() function which defines the model configuration
nyears	Number of years to run the model in each iteration (default=40)
n_iter	Number of iterations of the model (default=500)
start_temperature	Initial value of the simulated annealing temperature parameter (default=0.5). Suggested values in the range 0.0005 - 5. Higher values increase the probability of rejecting parameter combinations producing an improvement in likelihood
cooling	Rate at which the simulated annealing temperature declines with iterations (default=0.975). Suggested values in the range 0.9 - 0.985
quiet	Logical. If TRUE then suppress informational messages at the start of each iteration (default=TRUE)
csv.output	Logical. If TRUE then enable writing of csv output files (default=FALSE)
runtime.plot	Logical. If FALSE then disable runtime plotting of the progress of the run - useful for testing (default=TRUE)

The function **e2e_optimize_hr()** launches a StrathE2E simulated annealing process to find the set of fishing fleet model harvest ratio multipliers producing the maximum likelihood of observed target data on the state of the ecosystem, given specified environmental driving data and ecology model parameters.

Proposed values of the scaling parameters are contained in the configuration file for the fishing fleet model. In addition, the model setup includes a separate csv file (**harvest_ratio_multiplier.csv**) containing multipliers to be applied to each of these scaling parameters. These are useful when conducting scenario experiments, but would ordinarily be set to 1.0. The simulated annealing code for optimizing the scaling values iterates the multiplier values to seek the combination producing the best fit of the overall model to the set of target data. At the end of the process, the code optionally (if csv.output=TRUE) saves the history of proposed and accepted multiplier values to the current results folder, and the final accepted version of the multipliers back to the **/ModelName/VariantName/Param** folder with the model.itent text identifier assigned in the **e2e_read()** function call which must precede the run. If deemed worthy, these multipliers can then be manually applied to the corresponding parameter values in the fishing fleet configuration to produce an updated version. The effort-harvest ratio multipliers associated with birds, pinnipeds and cetaceans are

likely to be poorly constrained by observational data and may need to be set independently. Nevertheless thay are included in the process here.

Configuration of a simulated annealing run to estimate harvest ratio multipliers must be preceded by a **e2e_read()** function call to load the model setup which includes the initial values of the parameters to be fitted. It is important to note that the **models.path** argument in the **e2e_read()** function call needs to point to a user workspace folder, not the default North Sea model provided with the package. This is because the annealing function needs write-access to the model **/Param** folder, but the **/extdata/Models** folder in the package installation is read-only. To use the annealing function on the North Sea model, use the **e2e_copy()** function to make a copy of the North Sea model in the user workspace.

The coefficient of variation for 'jiggling' the parameter values during an optimization run is held in the file **optimize_fishing.csv** in the **/Param/control** folder of each model definition. This can be changed in real-time during a run, to enable exploration of the effects of parameter search patterns.

The function produces a real-time graphical summary of the progress of the fitting procedure, displaying the likelihoods of the proposed and accepted parameter sets at each iteration. The Y-axis (likelihood) range of the real time plot can be varied during the run by editing the setup file **optimize_fishing.csv** at any time during the run.

At the end of the procedure (provided csv.output=TRUE) new versions of the harvest ratio multiplier file are exported to the folder /Param of the model version, with a user defined identifier specified by the **model.ident** argument in the **e2e_read()** function. The histories of proposed and accepted parameter combinations are saved as csv files in the results folder and returned as a list object. The two csv files generated by the procedure have names:
annealing_HRmult_proposalhistory-*, annealing_HRmult_acceptedhistory-*, where * denotes the value of model.ident defined in the preceding e2e_read() function call.

In order to use the new fitted parameter values in a subsequent run of the StrathE2E model (using the **e2e_run()** function) it will be necessary to edit the **MODEL_SETUP.csv** file in the relevant **/Models/variant** folder to point to the new harvest ratio multiplier file.

Alternatively, to preserve the new harvest ratio multipliers and incorporate them into the fishing fleet model parameterisation the multiplier values need to be applied to the scaling parameters which link the integrated effort by each gear to the harvest ratio value which gets piped into the ecology model. Manually update the values in rows 12-21 (excluding the header row) of the file **/Param/fishing_fleet*.csv**, by multiplying the existing values by the new multipliers emerging from the annealing process. Then make sure to use a version of the multiplier file with all values set to 1.0 for all future runs.

If the edited file **fishing_fleet*.csv** is saved with a new identifier (*) then in order to use it in a subsequent run of the StrathE2E model (using the e2e_run() function) it will be necessary to edit the MODEL_SETUP.csv file in the relevant /Models/variant folder to point to the new file.

The returned list object contains three dataframes: parameter_proposal_history, parameter_accepted_history, new_parameter_data (a list of three). The proposal and accepted histories can be further analysed with the function e2e_plot_opt_diagnostics() to assess the performance of the optimization process.

```
# Copy the 2003-2013 version of the North Sea model supplied with the package into a user
# workspace (Windows OS):
    e2e_copy("North_Sea", "2003-2013",
    dest.path="Folders/Models")
#
# Load the 2003-2013 version of the North Sea model from the user workspace:
    model<-e2e_read(model.name="North_Sea",
                            model.variant="2003-2013",
                            models.path="Folder/Models",
                            model.ident="TEST")
# Since results.path is not defined here the histories of proposed and accepted
```

```
# parameter values will be written to a temporary folder if csv.output=TRUE.
#
# Quick demonstration of the annealing function in operation:
e2e_optimize_hr(model, nyears=5, n_iter=10, start_temperature=0.5, csv.output=TRUE)
#
# More realistic configuration would be as follows :
e2e_optimize_hr(model, nyears=50, n_iter=1000, start_temperature=1, csv.output=TRUE)
# (WARNING - this will take about 26 hours to run)
#
```

5.6 Estimating of fishing activity rates e2e_optimize_act()

Optimize Strathe2E fishing gerar activity multipliers to maximize the likelihood of either observed ecosystem target data or known harvest ratios.

Table 20. Arguments of the function **e2e_optimize_act()**.

Argument	Description
model	R-list object generated by the e2e_read() function which defines the model configuration
selection	Text string from a list to select the method for optimization. Select from: "HR", "ECO", Remember to include the phrase within "" quotes (default = "HR"). "HR" selects a scheme for optimizing the activity multipler to known harvest ratios and uses just the fishing fleet model. "ECO" selects a scheme for optimizing to the observational data in the state of the ecosystem and so uses both the fishing fleet and ecology models
n_iter	Number of iterations of the model (default=500)
start_temperature	Initial value of the simulated annealing temperature parameter (default=0.5). Suggested values in the range 0.0005 - 5. Higher values increase the probability of rejecting parameter combinations producing an improvement in likelihood
cooling	Rate at which the simulated annealing temperature declines with iterations (default=0.975). Suggested values in the range 0.9 - 0.985
csv.output	Logical. If TRUE then enable writing of csv output files (default=FALSE)
runtime.plot	Logical. If FALSE then disable runtime plotting of the progress of the run - useful for testing (default=TRUE)
n_traj	Specific to selection="HR" : Number of repeats (trajectories) of the simulated annealing process (default=100)
deltaHi	Specific to selection="HR" : Initial coefficient of variation for jiggling the activity multiplier values (default=0.2). This is set in the parameter file "annealing_SD_fishing.csv" in the case where selection="ECO" and can be manually varied during a run
attenuationstep	Specific to selection="HR" : Number of itereations between down-steps of the jiggling factor applied to the multiplier values. The jiggling rate is attenuated by a factor of 2 every xx iterations (default=500)
deltaGrat	Specific to selection="HR" : Coefficient of variation for jiggling the gear linkage values (default=0.25)
nyears	Specific to selection="ECO" : Number of years to run the ecology model in each iteration (default=40)
quiet	Logical. Specific to selection="ECO" : If TRUE then suppress informational messages at the start of each iteration (default=TRUE)

Usually, the fishing gear activity rates are a known input to the model. Sometimes, however, it may be necessary to use the model to hindcast the likely rates of activity given established values for the ecology

model parameters, the scaling parameters linking effort to harvest ratios, and the relative spatial distribution of fioshing. This is a situation that might arise when the coupled fishing and ecology model system has been comprehensively fitted for a data-rich period of time, and is then being applied to a different data-poor period where the fishing gear activity rates are unknown or only approximately known.

Proposed values of the gear activity rates need to be contained in the standard **fishing_activity*.csv** input file for the fishing fleet model. In addition, the model setup includes a separate csv file (**fishing_gear_multiplier.csv**) containing multipliers to be applied to each of these rates. These are useful when conducting scenario experiments, but would ordinarily be set to 1.0. In the same way as for the effort-harvest ratio scaling value fitting, the simulated annealing code for optimizing the gear activity rates iterates the multiplier values to seek the combination producing the best fit of the overall model to the set of target data. At the end of the process, the code saves the history of proposed and accepted multiplier values. If deemed reasonable, the final accepted version of the multipliers can then be manually applied to the corresponding rates in the initial input file to produce an updated version.

It is important to note that the **models.path** argument in the **e2e_read()** function call needs to point to a user workspace folder, not the default North Sea model provided with the package. This is because the annealing function needs write-access to the model **/Param** folder, but the **/extdata/Models** folder in the package installation is read-only. To use the annealing function on the North Sea model, use the **e2e_copy()** function to make a copy of the North Sea model in the user workspace.

There are two alternative ways to optimize fishing activity rates. The first is conceptually equivalent to the simulated annealing functions for optimizing the ecology model parameters, in that the target for fitting is the same database of observational indices of ecosystem status. This uses the fully coupled fishing fleet and ecology model system and hence is costly to run.

The alternative function for estimating the activity rate of gears is much faster and relies only on the fishing fleet model, but here the target for optimization is a set of known harvest ratios. The circumstances under which the past harvest ratios for each guild are sufficiently well known to enable this fitting approach are likely to be rather restricted, but this method is probably the more robust. In both approaches the chief problem is that the selectivity patterns of the fishing gears are likely to be highly overlapping so that it will be difficult to identify a realistic maximum likelihood set of activity rates.

The target data for the second method for estimating activity rates are contained in the file **zonal_harvest_r*.csv** (where * is a free text identifier) located in the **/ModelName/VariantName/Target** folder of the model definition data.

The implementation of simulated annealing in this function is slightly different to the other optimization functions in the package. In this case, because the problem of local maxima in the likelihood response surface is particularly acute (due to the potential overlap in the selectivities of different gears), so the annealing process is replicated many times from different randomly selected initial conditions. We refer to these replicates as 'trajectories'. Each trajectory follows a different pathway through the parameter space. At the end of the process, the best-fit set of activity multipliers is selected from across all the trajectories. In addition, the coefficient for jiggling the parameters is systematically attenuated with increasing iterations rather than remaining constant or potentailly being manually attenuated during the run my the user. Because the process uses only the fishing fleet model it is relatively fast so many trajectories can be run in a moderate time span.

To cope with the overlap in selectivity of the fishing gears, the function uses a set of linkage parameters specified in the the file **/Param/fishing_gear_linkages.csv** These parameters force selected gears activities to vary in concert ± some variation, rather than independently. The parameters for the gear linkages are located in the file ../Modelname/Variantname/Param/fishing_gear_linkages.csv. The table of linkages specifies which gear activity rates are forced to vary in concert during the fitting process, as opposed to varying independently. The value of the linkage coefficient defines the scaling of changes in the activity rate of a dependent gear relative to its linked independent gear. For example, if gear 8 is permitted to vary independently (value in column "Gear to which linked" = NA and "Linkage coefficient" = NA). If gear 9 is dependent on gear 8 then the activity rate of gear 9 this would be specified by e.g. "Gear to which linked" = 8 and "Linkage coefficient"

= 0.645. This would force the activity of gear 9 to be set at gear8_activity * (0.645 ± a random variation defined by the function argument **deltaGrat**).

The function produces a real-time graphical summary of the progress of the fitting procedure, displaying the likelihoods of the proposed and accepted parameter sets at each iteration. Y-axis (likelihood) range of the real time plot can be varied during the run by editing the control file **optimize_fishing.csv**

At the end of the procedure, provided that csv.output=TRUE, a new version of the gear multipliers file is exported to the folder **/Param** of the model version, with a user defined identifier specified by the **model.ident** argument in the **e2e_read()** function. The multiplier data from the end of each trajectory, the emergent harvest ratios, and the associated likelihood are returned as a list object and optionally saved as csv files in the results folder. The list and csv outputs also includes the same data on multipliers expressed relative to the initial values, and the emergent harvest ratios expressed relative to the target values. These data can be used in the function **e2e_plot_opt_diagnostics()** to assess the performance of the optimization process. File names for the csv outputs are as follows: **activity_optim_gearmult_history*.csv**, **activity_optim_gearmult_relinitial_history*.csv**, **activity_optim_harvestratio_history*.csv**, **activity_optim_harvestratio_reltarget_history*.csv**

In order to use the new fitted parameter values in a subsequent run of the StrathE2E2 model (using the **e2e_run()** function) it will be necessary to edit the **MODEL_SETUP.csv** file in the relevant **/Models/Variant** folder to point to the new harvest ratio multiplier file. Or, to preserve the new activity rate multipliers and incorporate them into the fishing fleet model parameterisation, manually update the values in the file **/Param/fishing_activity*.csv**, by multiplying the existing values by the new multipliers emerging from the annealing process. If using Excel to do this task beware of the number formatting settings - specifically the number of decimal places displayed with scientific notation. Increase the number of decimal places before saving as a csv file or precision of the activity rates will be lost.

If the edited file **fishing_activity*.csv** is saved with a new identifier (*) then in order to use it in a subsequent run of the StrathE2E2 model (using the **e2e_run()** function) it will be necessary to edit the **MODEL_SETUP.csv** file in the relevant **/Models/Variant** folder to point to the new file. Then make sure to use a version of the multiplier file with all values set to 1.0 for all future runs.

```
# Copy the 1970-1999 version of the North Sea model supplied with the package into a
# user workspace (Windows OS):
e2e_copy("North_Sea", "1970-1999",
            dest.path="Folder/Models")
#
# Load the 1970-1999 version of the North Sea model from a user workspace:
model<-e2e_read(model.name="North_Sea",
                  model.variant="1970-1999",
                  models.path="Folder/Models",
                  model.ident="TEST")
# Since results.path is not defined here the histories of proposed and accepted
# parameter values will be written to a temporary folder if csv.output=TRUE.
#
# Quick Demonstration of optimizing to ecosystem data:
test_run <- e2e_optimize_act(model, selection="ECO", n_iter=10, start_temperature=0.4,
                             cooling=0.975, csv.output=FALSE, nyears=5 )
str(test_run,max.level=1)                          # View the structure of the returned list
#
# More realistic configuration would be (WARNING - this will take about 14 hours to run) :
fitting_data <- e2e_optimize_act(model, selection="ECO", n_iter=500, start_temperature=0.4,
                                 cooling=0.975, csv.output=TRUE, nyears=40 )
#
#
# Quick Demonstration of optimizing to harvest ratios:
```

```
test_run <- e2e_optimize_act(model, selection="HR", n_iter=100, start_temperature=1.0,
                             cooling=0.985, csv.output=FALSE, n_traj=5 )
str(test_run,max.level=1)                          # View the structure of the returned list
str(test_run$new_parameter_data,max.level=1)       # View the structure of the returned list
                                                   # element containing parameter objects
test_run$new_parameter_data                        # View the new, final accepted parameter data#

# More realistic configuration would be (WARNING - this will take about 6 hours to run) :
model<-e2e_read(model.name="North_Sea",
                model.variant="1970-1999",
                models.path="Folder/Models",
                model.ident="fittingrun")
fitting_data<-e2e_optimize_act(model, selection="HR", n_iter=3000, start_temperature=1.0,
                               cooling=0.985, csv.output=TRUE, n_traj=100 )
```

5.7 Diagnostics for optimization processes e2e_plot_opt_diagnostics()

Plot diagnostic data on the performance of parameter optimization functions.

Table 21. Arguments of the function **e2e_plot_opt_diagnostics()**.

Argument	Description
model	R-list object generated by the **e2e_read()** function which defines the model configuration
selection	Text string from a list to select the method for optimization. Select from: "ECO", "HR", "ACT", corresponding to output generated by the e2e_optomize_eco(), e2e_optomize_hr(), or e2e_optomize_act() function, Remember to include the phrase within "" quotes (default = "HR"). "HR" selects a scheme for optimizing the activity multipler to known harvest ratios and uses just the fishing fleet model. "ECO" selects a scheme for optimizing to the observational data in the state of the ecosystem and so uses both the fishing fleet and ecology models
fitted.to	Specific to the case where selection="ACT"; text string from a list identifying which version of activity optimization procedure generated the data to be plotted. Select from: "ECO", "HR", corresponding to maximising the likelihod of ecosystem state data, or zonal harvest ratios respectively. Default="". Remember to include the phrase within"" quotes
use.saved	Logical. If TRUE use data from a prior user-defined run held as csv files data in the current results folder (default=FALSE).
use.example	Logical. If TRUE use pre-computed example data from the internal North Sea model rather than user-generated data (default=FALSE).
results	R-list object of output from an optimization process generated by the e2e_optomize_eco(), e2e_optomize_hr(), or e2e_optomize_act() function. Only needed if use.saved1=FALSE and use.example1=FALSE. (Default=NULL).

Creates diagnostic plots from outputs of the functions for optimizing ecology, harvest ratio, and fishing activity parameters: e2e_optimize_eco(), e2e_optimize_hr(), and e2e_optimize_act() respectively.

The function takes the history of parameter values proposed during each simulated annealing process, expresses these relative to the initial value of each parameter, generates qunatiles (0.005, 0.25, 0.5, 0.75 and 0.995) of the distribution for each parameter, and plots the results as box-and-whisker diagrams. The relativity to initial values is expressed as (proposed - initial)/initial so the initial is represented by a relative value of zero. For each parameter, the final accepted value is over-plotted onto the box-and-whisker as a red bar. The resulting diagram, with a separate box-and-whisker for each parameter shows the extent to which each

parameter has migrated from its initial value en-route to the final accepted state.

The format of the parameter diagram is the same for each type of optimization scheme: ecology, harvest ratio, and fishing activity parameters. However, in the case where activity parameters are optimized to zonal harvest ratios rather than ecosystem state data, the diagram has an additional panel showing the distribution of proposal harvest ratios relative to the target, and (in red) the harvest ratios corresponding to the final accepted activity parmeters (relative to the targets).

Arguments for this function permit the input data to be drawn from an existing data object generated by the various optimization functions, previously genearted csv files, or example data provided with as a supplementary data-package for versions of the internal North Sea models. The first time that example data are called from a StrathE2E2 function, the data-package is downloaded and installed from the GitLab server https://gitlab.com/MarineResourceModelling/StrathE2E/strathe2e2examples. Thereafter the example data are available offline.

Documentation in a dataframe format on each of the classes of parameters in the model can be obtained with the function **e2e_get_parmdoc()**. This provides a key to the abbreviated parameter names especially in the diagnostic plots for **e2e_optimize_eco()**.

As well as drawing the plot the function returns a list object containing a) an array of the quantiles of the proposal distribution for each parameter, and b) an array of one row, of the final accepted parameter set from the procedure.

```
# Load the 1970-1999 version of the North Sea model supplied with the package and generate
# a quick test data object with only 10 itereations and running the model for 5 years.
# More realistic would be at least 500 iterations and running for 50 years. Even so this
# example will take a few minutes to run:
model<-e2e_read(model.name="North_Sea",
                model.variant="1970-1999",
                model.ident="test")
# This model is already optimized to the observed ecosystem data supplied with the package
# Perturb the temperature driving to knock the model away from its maximum likelihood state
# relative to the target data:
model$data$physics.drivers$so_temp <- model$data$physics.drivers$so_temp+3
                                      # add 3 degC to upper layer offshore temperatures
model$data$physics.drivers$si_temp <- model$data$physics.drivers$si_temp+3
                                      # add 3 degC to inshore temperatures
model$data$physics.drivers$d_temp  <- model$data$physics.drivers$d_temp+3
                                      # add 3 degC to lower layer offshore temperatures
test_run  <- e2e_optimize_eco(model, nyears=5, n_iter=10, start_temperature=0.4,
                              csv.output=FALSE,runtime.plot=FALSE)
plot_data <- e2e_plot_opt_diagnostics(model,selection="ECO",results=test_run)
str(plot_data,max.level=1)  # show the structure of the list object plot_data
```

```
# Or... plot example date supplied with the package showing some data generated during the
# process of optimizing the North Sea model:
model     <- e2e_read(model.name="North_Sea", model.variant="1970-1999")
Current working directory is...
'C:/Users/Public/Documents/StrathE2E2'
No 'results.path' specified so any csv data requested
will be directed to/from the temporary directory...
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ'

Model setup and parameters gathered from ...
StrathE2E2 package folder
Model results will be directed to/from ...
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ/North_Sea/1970-1999/'
```

```
plot_data <- e2e_plot_opt_diagnostics(model,selection="ECO",use.example=TRUE)
Reading example results from StrathE2E2examples data package for the North_Sea 1970-1999 model
Reading example results from StrathE2E2examples data package for the North_Sea 1970-1999 model
```

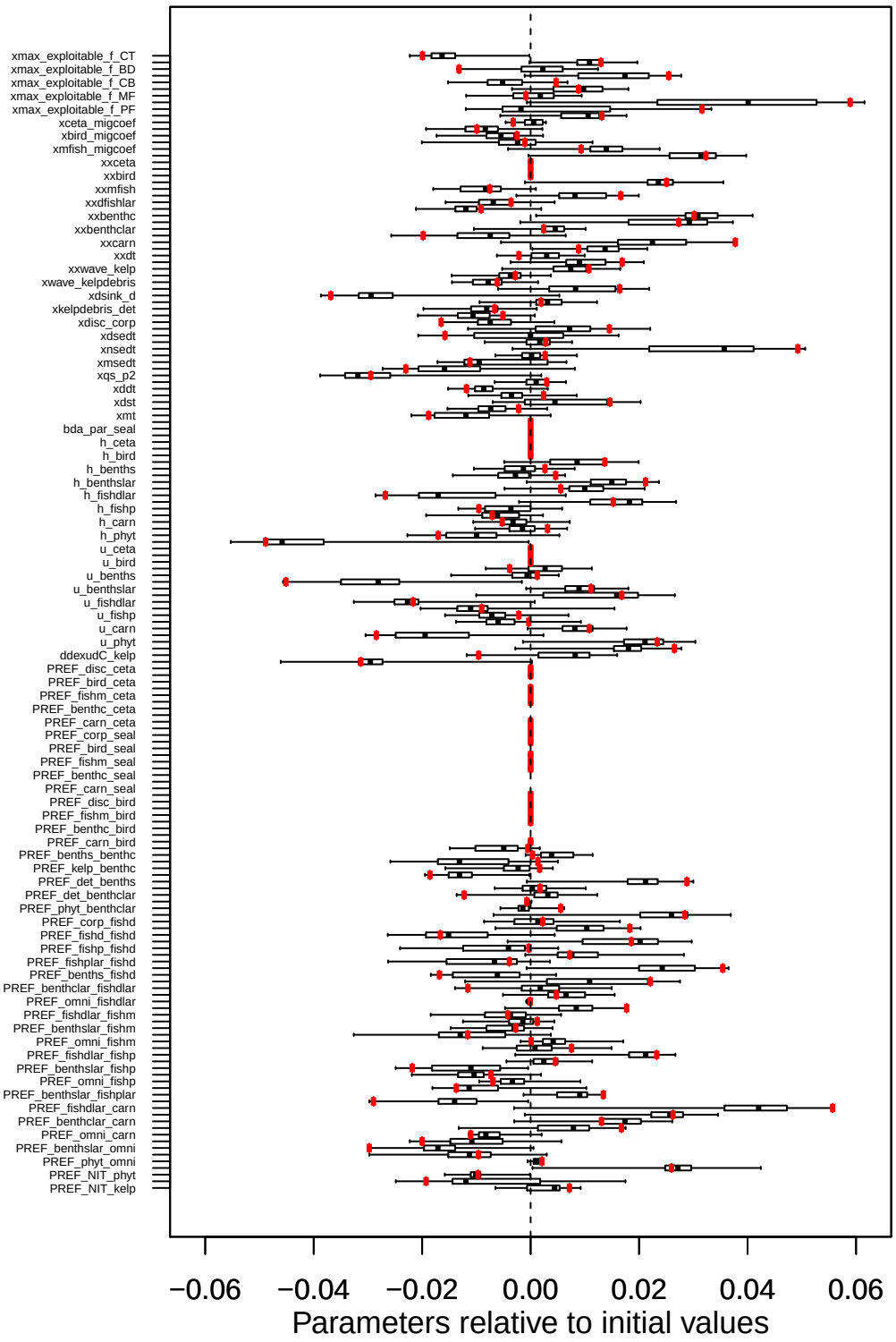


Figure 11. Diagnostic plot of the procedure from optimizing ecology model parameters for the North Sea

1970-1999 model.

```
# Same for harvest ratio optimization...
model<-e2e_read(model.name="North_Sea",
                model.variant="1970-1999",
                model.ident="test")
# This model is already optimized to the observed ecosystem data supplied with the package
# Perturb the temperature driving to knock the model away from its maximum likelihood
# state relative to the target data:
model$data$physics.drivers$so_temp <- model$data$physics.drivers$so_temp+3
                                      # add 3 degC to upper layer offshore temperatures
model$data$physics.drivers$si_temp <- model$data$physics.drivers$si_temp+3
                                      # add 3 degC to inshore temperatures
model$data$physics.drivers$d_temp  <- model$data$physics.drivers$d_temp+3
                                      # add 3 degC to lower layer offshore temperatures
test_run  <- e2e_optimize_hr(model, nyears=5, n_iter=10, start_temperature=0.4,
                             csv.output=FALSE,runtime.plot=FALSE)
plot_data <- e2e_plot_opt_diagnostics(model,selection="HR",results=test_run)
str(plot_data,max.level=1)  # show the structure of the list object plot_data
```

```
# Or... plot example date supplied with the package showing some data generated during
# the process of optimizing the North Sea model:
model     <- e2e_read(model.name="North_Sea", model.variant="2003-2013")
Current working directory is...
'C:/Users/Public/Documents/StrathE2E2'
No 'results.path' specified so any csv data requested
will be directed to/from the temporary directory...
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ'

Model setup and parameters gathered from ...
StrathE2E2 package folder
Model results will be directed to/from ...
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ/North_Sea/2003-2013/'
plot_data <- e2e_plot_opt_diagnostics(model,selection="HR",use.example=TRUE)
Reading example results from StrathE2E2examples data package for the North_Sea 2003-2013 model
Reading example results from StrathE2E2examples data package for the North_Sea 2003-2013 model
```

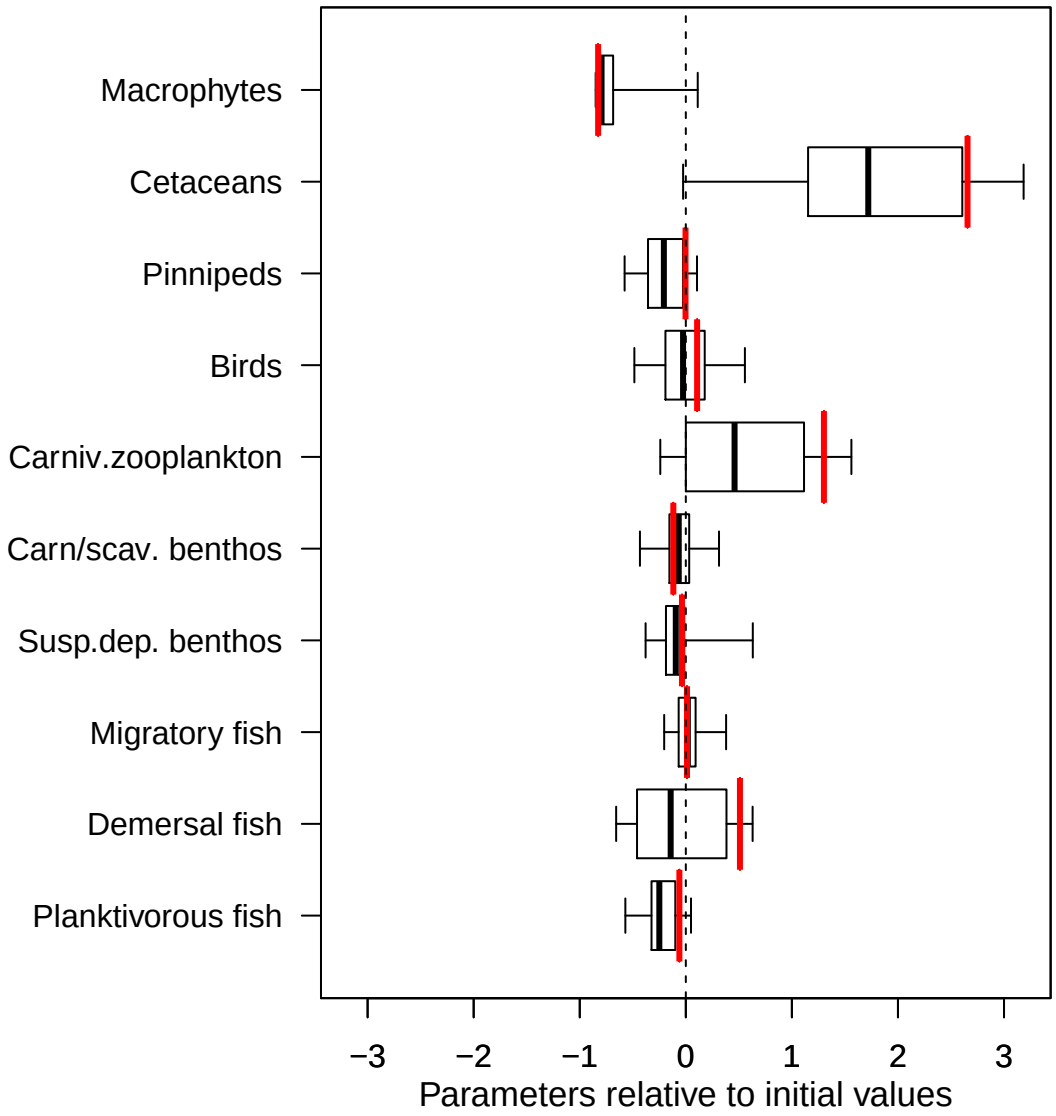


Figure 12. Diagnostic plot of the procedure for optimizing harvest ratio multipliers parameters for the North Sea 2003-2013 model.

```
# For activity rate optimization relative to ecosystem data:
model<-e2e_read(model.name="North_Sea",
                model.variant="1970-1999",
                model.ident="test")
Current working directory is...
'C:/Users/Public/Documents/StrathE2E2'
No 'results.path' specified so any csv data requested
will be directed to/from the temporary directory...
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ'

Model setup and parameters gathered from ...
StrathE2E2 package folder
Model results will be directed to/from ...
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ/North_Sea/1970-1999/'
# This model is already optimized to the observed ecosystem data supplied with the package,
```

```
# but not by optimizing gear activity rates
# The e2e_optimize_eco() function was used in this case.
# Perturb the temperature driving to knock the model away from its maximum likelihood
# state relative to the target data:
model$data$physics.drivers$so_temp <- model$data$physics.drivers$so_temp+3
                                        # add 3 degC to upper layer offshore temperatures
model$data$physics.drivers$si_temp <- model$data$physics.drivers$si_temp+3
                                        # add 3 degC to inshore temperatures
model$data$physics.drivers$d_temp  <- model$data$physics.drivers$d_temp+3
                                        # add 3 degC to lower layer offshore temperatures
test_run  <- e2e_optimize_act(model, selection="ECO", n_iter=10, start_temperature=0.4,
                              cooling=0.975, csv.output=FALSE, nyears=5 )
[1] "Sun Feb 14 17:32:59 2021"
Iteration: 1; proposal likelihood: 0.2167900;  accepted: YES
Iteration: 2; proposal likelihood: 0.2274985;  accepted: YES
Iteration: 3; proposal likelihood: 0.2181817;  accepted: YES
Iteration: 4; proposal likelihood: 0.2318373;  accepted: YES
Iteration: 5; proposal likelihood: 0.2477961;  accepted: YES
Iteration: 6; proposal likelihood: 0.2248988;  accepted: YES
Iteration: 7; proposal likelihood: 0.2168564;  accepted: YES
Iteration: 8; proposal likelihood: 0.2208385;  accepted: YES
Iteration: 9; proposal likelihood: 0.1914068;  accepted: NO
Iteration: 10; proposal likelihood: 0.2117411;  accepted: YES
Model is read-only: accepted gear multiplier values have not been written to model parameters folder
csv.output is FALSE: accepted gear multiplier values have not been written to model parameters folder
plot_data <- e2e_plot_opt_diagnostics(model,selection="ACT",fitted.to="ECO",results=test_run)
str(plot_data,max.level=1)  # show the structure of the list object plot_data
List of 2
 $ proposed_value_disribution_rel_initial: num [1:5, 1:12] -0.00398 0.00995 0.02263 0.03729 0.0578 ...
  ..- attr(*, "dimnames")=List of 2
 $ finalvalue_rel_initial                : Named num [1:12] 0.0317 0.1373 -0.0534 0.0793 -0.0668 ...
  ..- attr(*, "names")= chr [1:12] "Pelagic_Trawl+Seine" "Sandeel+sprat_trawl(Otter30-70mm+TR3)" "Longl

# There are no example data available in the package for this function
```

```
# For activity rate optimization relative to zonal harvest ratios:
model<-e2e_read(model.name="North_Sea",
                model.variant="1970-1999",
                model.ident="test")
Current working directory is...
'C:/Users/Public/Documents/StrathE2E2'
No 'results.path' specified so any csv data requested
will be directed to/from the temporary directory...
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ'

Model setup and parameters gathered from ...
StrathE2E2 package folder
Model results will be directed to/from ...
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ/North_Sea/1970-1999/'
# Activity rates in this model are already optimized to the target harvest ratios supplied
# with the package but we woud not expect
# to recover these values in this short demonstration run
test_run  <- e2e_optimize_act(model, selection="HR", n_iter=100, start_temperature=1.0,
                              cooling=0.985, csv.output=FALSE, n_traj=5 )
```

```
[1] "Sun Feb 14 17:34:25 2021"
Trajectory: 1 of 5; likelihood: 0.6170642; global max likelihood: 0.6170642
Trajectory: 2 of 5; likelihood: 0.2208441; global max likelihood: 0.6170642
Trajectory: 3 of 5; likelihood: 0.4094948; global max likelihood: 0.6170642
Trajectory: 4 of 5; likelihood: 0.2037387; global max likelihood: 0.6170642
Trajectory: 5 of 5; likelihood: 0.6170572; global max likelihood: 0.6170642
Model is read-only: accepted gear multiplier values have not been written to model parameters folder
csv.output is FALSE: accepted gear multiplier values have not been written to model parameters folder
plot_data <- e2e_plot_opt_diagnostics(model,selection="ACT",fitted.to="HR",results=test_run)
str(plot_data,max.level=1)  # show the structure of the list object plot_data
List of 4
 $ gearmult_distrubution_rel_initial : num [1:5, 1:12] -0.653 -0.336 -0.186 -0.122 0.103 ...
  ..- attr(*, "dimnames")=List of 2
 $ gearmult_maxlikelihood_rel_initial: Named num [1:12] 0.108 -0.237 0.336 -0.275 -0.152 ...
  ..- attr(*, "names")= chr [1:12] "PTS" "SST" "LLm" "BTf" ...
 $ HRvalues_distrubution_rel_initial : num [1:5, 1:20] -0.5105 -0.2708 0.0326 0.033 0.2139 ...
  ..- attr(*, "dimnames")=List of 2
 $ HRvalues_maxlikelihood_rel_initial: Named num [1:20] 0.0326 0.0394 -0.207 -0.0931 0.0562 ...
  ..- attr(*, "names")= chr [1:20] "Plank.fish_i" "Plank.fish_o" "Dem.fish_i" "Dem.fish_o" ...
```

```
# Or... plot example date supplied with the package showing some data generated during the
# process of optimizing the North Sea model:
model     <- e2e_read(model.name="North_Sea", model.variant="1970-1999")
Current working directory is...
'C:/Users/Public/Documents/StrathE2E2'
No 'results.path' specified so any csv data requested
will be directed to/from the temporary directory...
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ'

Model setup and parameters gathered from ...
StrathE2E2 package folder
Model results will be directed to/from ...
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ/North_Sea/1970-1999/'
plot_data <- e2e_plot_opt_diagnostics(model,selection="ACT",fitted.to="HR",use.example=TRUE)
Reading example results from StrathE2E2examples data package for the North_Sea 1970-1999 model
Reading example results from StrathE2E2examples data package for the North_Sea 1970-1999 model
```

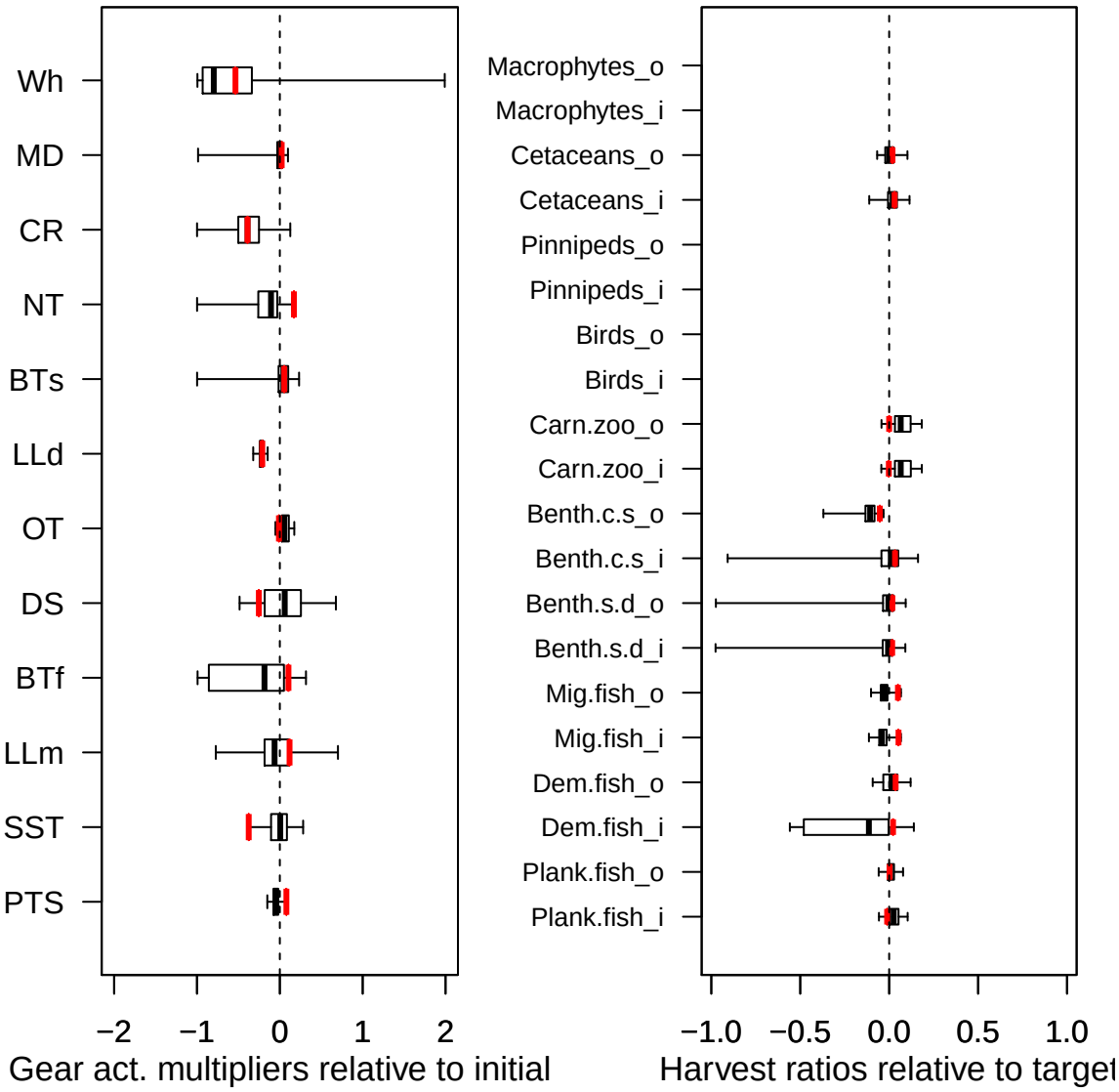


Figure 13. Diagnostic plot of the procedure or optimizing gear activity multiplers parameters for the North Sea 1970-1999 model.

5.8 Calculating initial values for harvest ratio scaling parameters e2e_calculate_hrscale()

Calculate initial values of the scaling parameters for the fishing fleet model which link effort to harvest ratios.

Table 22. Arguments of the function **calculate_hr_scale_values()**.

Argument	Description
model	R-list object generated by the e2e_read() function which defines the model configuration

The scaling parameters convert the effort applied by each gear to each living resource guild in the model, into a harvest ratio (proportion of mass captured per day). Effort is the product of activity and the relative power of each gear with respect to each guild.

In order to estimate these scaling parameters, data are needed for a 'calibration' time period when activity, catch and harvest ratio are all known.

The function assumes that the relevant model configuration has already been loaded using the **e2e_read()** function. It is expected that this has loaded the array of power parameters (catch per unit activity by gear and guild during the calibration period) given in the file **/Param/fishing_power*.csv**, the corresponding activity in the file **/Param/fishing_activity*.csv**, and the known harvest ratios of each guild in the file **/Target/region_harvest_r*.csv**

The function returns rough estimates of the scaling parameters. These are only rough because they assume that the effort and resource biomass are distributed uniformly over the model domain. Hence they should be used as initial estimates to be refined by optimization.

The scaling parameters are dispayed on the screen and saved in a dataframe. The values then need to be manually edited into rows 12-21 (excluding the header row) of the file **/Param/fishing_fleet*.csv**

```
# Load the 2003-2013 version of the North Sea model supplied with the package and calculate
# scaling parameter values:
model <- e2e_read("North_Sea", "2003-2013",silent=TRUE)
#
scale_values <- e2e_calculate_hrscale(model)
Manually edit these values into the 'fishing_fleet_*.csv' file, rows 12-21, in the 'Param' folder
scale_values
                Guild     HRscale
1  Planktivorous_fish  0.03102313
2       Demersal_fish  0.10674489
3      Migratory_fish  0.05912854
4    Benthos_susp-dep  5.86628271
5   Benthos_carn-scav  0.78986760
6    Zooplankton_carn 14.22221473
7               Birds  2.71513873
8           Pinnipeds  4.38631271
9           Cetaceans  6.87913648
10        Macrophytes  0.00000000
#
```

6 Sensitivity and Monte Carlo analysis

6.1 Background

6.1.1 Sensitivity analysis

Sensitivity analysis helps to highlight the parameters which have the most influence in the simulated annealing optimization process. In addition the analysis can help to identify parameters for which constraint by external information would be most beneficial.

Simple one-at-a-time sensitivity analysis of parameters in a model is a straightforward process when there are few parameters and relatively little interaction between parameters. However, StrathE2E2 include many parameters and there is a strong possibility of interactions, so a more sophisticated scheme is required.

The R package includes a function to perform global analysis of parameter sensitivity using the Morris Method (Morris, 1991). This involves a factorial sampling scheme to generate parameter values for replicate model runs and generate values for the 'Elementary Effect' (EE) of each parameter. The mean and standard deviation of EE values over many runs allows ranking of the parameters in terms of the sensitivity of their effects on the model, and the strength of interactions with other parameters. For a review of global sensitivity analysis methods, including the Morris Method, see Wu et al. (2013).

The parent parameter set for the sensitivity analysis (θ) incorporates all of the parameters and inputs to the model, ideally. the maximum-likelihood set of fitted ecology model parameters produced by the simulated annealing scheme, plus the fixed ecological parameters, the physical configuration parameters of the model (layer thicknesses, inshore/offshore areas), the parameters passed to the ecology model from the fishing fleet model, and the fishing activity rates and environmental driving data. The sensitivity to the environmental driving data is investigated by applying a scaling factor uniformly to the annual cycle of environmental data values.

From this parent set, a series of child-sets (θ^*_k where $1 \geq k \geq r$) is generated by applying a separate random increment to each parameter; $\theta^*_k = \theta + \delta[k]$ where $\delta[k]$ is a vector of random values from a gaussian distribution of mean 0 and standard deviation given by a fixed coefficient of variation applied to the parent-set value of each parameter. This process is equivalent to that for generating parameter proposals in the simulated annealing scheme, except that all of the parameters and model inputs are varied in the sensitivity analysis, rather than just the subset of parameters to be fitted.

For each of the child-sets of parameters, either the likelihood of the suite of observed data indices $P(observations|\theta^*_{k,0})$ or one of a range of possible outputs from the model $Y|\theta^*_{k,0}$ is calculated following runs of the model to stationary state. We refer to these as trajectories. The model output selected as the subject of the sensitivity analysis is referred to as the criterion.

Then, for each trajectory, the parameters ($1 \geq i \geq n$) are varied in turn, one at a time, by adding a fixed proportionality increment, the model re-run, and the likelihood computed ($P(observations|\theta^*_{k,i})$). The proportionality increment for a given trajectory is drawn at random from a set of four fixed levels in the range $\pm 10\%$ of the child parameter set (-10%, -5%, +5%, +10%, i.e. $\Delta = 0.9, 0.95, 1.05, 1.10$), so that:

$$\theta[i]^*_{k,i} = \theta[i]^*_{k,0}.\Delta[z]$$

where $\theta[i]^*$ is the ith element of θ^*, $\Delta[z]$ is the zth element of Δ, and z is drawn at random from the series (1, 2, 3, 4) for each value of k. Hence, for each trajectory the model runs are repeated $n + 1$ times, where i=0 corresponds to the baseline run for each trajectory. The total number of nested runs to support the sensitivity analysis is thus $r * (n + 1)$.

From each level-run of the model (in which a single parameter is perturbed), and its corresponding likelihood, the Elementary Effect of the parameter is calculated as (illustrated for the case where the criterion is the likelihood of the observed data):

$$EE_{k,i} = \frac{(P(observations|\theta^*_{k,i}) - P(observations|\theta^*_{k,0}))}{\sqrt{(\Delta[z] - 1)^2}}$$

On completion of the runs for all trajectories, the mean (μ_i) and standard deviation (S_i) of the r Elementary Effects for each parameter (i) are calculated. For the mean,

$$\mu_i = \frac{1}{r} \sum_{k=1}^{r} EE_{k,i}$$

The magnitude of the mean represents the sensitivity of each parameter, and the corresponding standard deviation indicates the degree of non-linearity in the response or interaction with other parameters. The standard error of the mean for each parameter (SEM_i) is given by:

$$SEM_i = \frac{S_i}{\sqrt{r}}$$

If $S_i > |\mu_i|.\frac{\sqrt{r}}{2}$ then we can approximately conclude that μ_i is significantly greater than zero.

6.1.2 Monte Carlo analysis

The Monte Carlo scheme provided with the StrathE2E2 package has two modes of operation referred to as "baseline-mode", and "scenario-mode".

Baseline-mode involves generating a list of ecology model parameter sets and, for each set, determining the likelihood of observational data consistent with the environmental and fishery drivers. The model outputs generated with each set are then weighted by the likelihood before deriving quantiles of their distribution. The quantile ranges then represent credible intervals of the model outputs.

The ensemble of baseline-mode parameter sets is generated by sampling the ecology model parameters from uniform distibutions centred on an initial (baseline) parameter set. Ideally, this should be the maximum likelihood parameter set arrived at by prior application the various optimization functions in the package to 'fit' the model to a suite of observational data on the state of ecosystem (target data for fitting are in /Modelname/Variantname/Target/annual_target_data_*.csv). Each iteration in the Monte Carlo scheme then generates a unique time series of model outputs at daily intervals, together with the overall likelihood of the obervational target data. From these data, the function then derives credible intervals of each output by weighting the values from each parameter set by the associated likelihood.

For each of the new parameter sets, the likelihood of the suite of observed data indices $P(observation|\theta_k^*)$ is calculated following a run of the model to stationary state (as described in section 5 on parameter optiization), and all the output from the final, stationary year of each run is saved ($k = 0$ to r, where $k = 0$ corresponds to the maximum likelihood (baseline) parameter set).

To calculate the credible interval for any direct or derived model output variable (e.g. annual average mass density of a state variable, or the mass density on a given day in the final year), the values from the individual model runs (V_k) and the associated likelihoods (P_k) are assembled as a list of ($r + 1$; $k = 0$ to r) pairs (a_k = (V_k,P_k)). The list for each variable is then sorted by ascending values V (a_j ; $j = 1$ to $(r + 1)$ such that $V_j \geq V_{(j-1)}$). Then, the cumulative likelihood and the cumulative likelihood with increasing variable value is calculated as follows

$$C_j = (P_{j=1}, \sum_{j=1}^{j=2} P_j, \sum_{j=1}^{j=3} P_j, \sum_{j=1}^{j=4} P_j ... \sum_{j=1}^{j=(r+1)} P_j)$$

Proportions of the maximum cumulative likelihood are then caclulated as $Q_j = C_j/C_{(r+1)}$

Finally, values of V corresponding to discrete values of Q = (0.005, 0.25, 0.5, 0.75, 0.995) are extracted by interpolation. These values span the 0.5% and 99.5% credible intervals of the model output given the observed target data and uncertainty in the fitted ecology parameters. Note that uncertainties in the fixed ecology parameters, fishing fleet parameters, and environmental driving data are not reflected in these credible intervals.

In the "scenario-mode" of operation, there are no target data against which the outpiuts from each member of the ensemble can be compared. In this case, the parameter sets and their associated likelihoods are taken from a pre-existing **baseline** mode simulation. The parameter sets are used to generate model outputs for a scenarios of environmental or fishing drivers - e.g. increased activity by selected gears, or warmer sea temperatures. These scenario outputs are then weighted by the **baseline** mode likelihoods before derivation of quantiles and credible intervals.

6.2 Performing a sensitivity analysis e2e_run_sens()

Perform a parameter sensitivity analysis on the StrathE2E model.

Table 23. Arguments of the function **e2e_run_sens()**.

Argument	Description
model	R-list object generated by the e2e_read() function which defines the model configuration
nyears	Number of years to run the model in each iteration (default=50)
n_traj	Number of trajectories of parameter sets (default=16)
trajsd	Coefficient of variation used to set the standard deviation for the gaussian distribution from which new paramater values are drawn to create each trajectoy baseline from the initial parent values (default=0.0075)
n_setoflevels	Number of fixed levels of coefficient of variation used to generate the individual parameter values in each level-run. Must be an even number (default=4)
v_setoflevels	Maximum coefficient of variation for the set of levels (default=0.1, i.e. -10 percent to +10 percent)
coldstart	Logical. If TRUE then the run is starting from cold - which means that the first trajectory baseline is the parent configuration as specified in the 'model' list object. If FALSE then signifies that this is a parallel run whuch will later be merged with the 'coldstart=TRUE' run. In this case the first trajectory baseline is a derivative of the parent. Default=TRUE
quiet	Logical. If TRUE then suppress informational messages at the start of each iteration (default=TRUE)
postprocess	Logical. If TRUE then process the results through to a final sorted list of parameter sensitivies for plotting. If FALSE just produce the raw results. The reason for NOT processing would be if the job has been shared across multiple machines/processors and several raw result files need to be merged before processing. Default=TRUE
csv.output	Logical. If TRUE then enable writing of csv output files (default=FALSE)
runtime.plot	Logical. If FALSE then disable runtime plotting of the progress of the run - useful for testing (default=TRUE)
outID	Numeric value in the range 0 to 247. Selects the output criterion to be used as the basis for the analysis. Default=0 corresponds to the likelihood of observed target data. Other values obtainable by running e2e_get_senscrit()

Performs a one-at-a-time parameter sensitivity analysis on the StrathE2E model using the Morris Method for factorial sampling of the physical configuration parameters, ecology model parameters, the fishing fleet model parameters, and the environmental forcings (see section 6.1.1. above for details).

The basis for the method is a scheme for sampling the model parameters, one at a time, from distributions around baseline sets, and testing the effects on the performance of the model against some criterion. The criterion can be the likelihood of the observational data on the state of the ecosystem that are used as the target for parameter optimization in the various simulated annealing functions supplied with the package, or any one of a range of model outputs (selected by the argument outID).

The process requires an initial set of parameters for the model. We refer to this as the 'parent' parameter set. It is recommended that this should be the parameters producing the maximum likelihood of the observational target data (as estimated by e.g. the e2e_optimize_eco() function). The MODEL_SETUP.csv file in the folder /Models/Modelname/Modelvariant/ should be configured to point to the relevant files, and then loaded with the e2e_read() function.

From this parent set, a series of 'child' parameter sets are generated by applying a separate random increment to each parameter drawn from a gaussian distribution of mean 0 and standard deviation given by a fixed coefficient of variation applied to the parent-set value of each parameter.

For each of the child-sets of parameters, the criterion is calculated following runs of StrathE2E to stationary state. We refer to these as trajectory baselines.

Then, for each trajectory, the parameters are varied in turn, one at a time, by adding a fixed proportionality increment to the trajectory baseline values, the model re-run, and the criterion computed. We refer to these

as 'level runs'. The proportionality increment is the same for all of the level runs within a given trajectory, and is drawn at random from a set of fixed levels distributed symetrically around 0 (e.g. -10, -5, +5, +10 percent, i.e. proportions of the trajectory baseline values = 0.9, 0.95, 1.05, 1.10).

For each level run, the 'Elementary Effect (EE)' of the given parameter is calculated from the difference between the level run value of the criterion and the corresponding trajectory baseline criterion value.

On completion of all the trajectories, the raw results are (optionally) post-processed to generate the mean and standard deviations of all the EE values for each parameter. EE_mean is an index of the magnitude of the sensitivity, and EE_sd is an index of the extent of interaction with other parameters.

During the run the function produces a real-time plot for each trajectory, in which the x-axis represents the sequence of parameters, and the y-axis is the criterion for the analysis. A horizontal red line indicates the likelihood of the parent parameter set, horizontal grey line indicates the likelihood for each trajectory baseline and each level-run likelihood is shown by a symbol. The y-axis range can be changed in real-time by editing the setup file **/Models/Modelname/Modelvariant/Param/control/sensitivity.csv**

The outputs from the function are directed to csv files in the current "results" folder, provided that csv.output=TRUE. The outputs are: a) Table of parameter values applied in each run of the model (OAT_parameter_values-*.csv, where* = model.ident as defined by e2e_read()) b) Table of the criterion value and EE value for each trajectory/level run (OAT_results-*.csv) c) If post-processing is selected, then a table of Mean EE and standard deviation of EE for each parameter, sorted by the absolute value of EE_mean (sorted_parameter_elemental_effects-*.csv)

WARNING - This function will take several days to run to completion on a single processor with even a modest number of iteration. The total number of model runs required to support the analysis is r*(n+1) where r is the number of trajectories and n is the number of parameters. The function incorporates all of the physical configuration parameters, fixed and fitted ecology model parameters, the fishing fleet model parameters, and the environmental forcings into the analysis, so n = 450. Each model run needs to be sufficiently long to achieve a stationary state and as a consequnce a typical runtime of around 10h per trajectory. The mininimum recommended number of trajectories is 15, to the function can take several days to complete.

However, it is possible to spread the load over multiple processor/machines with arguments in the function allowing for management of this parallelisation. Afterwards, the raw results files are combined into a single data set using the e2e_merge_sens_mc() function, and then processed using the function e2e_process_sens_mc().

A separate function e2e_plot_sens_mc() produces a graphical representation of the EE_mean and EE_sd results.

Example results data for both variants of the North Sea model generated with the default function settings are included in the package and can be plotted by setting an argument in the **e2e_plot_sens()** function.

```
# Load the 2003-2013 version of the North Sea model supplied with the package:
model <- e2e_read("North_Sea", "1970-1999",silent=TRUE)
# Since results.path is not defined here the results will be written
# to a temporary folder if csv.output=TRUE.
#
# Run the sensitivity analysis process:
# WARNING - Running a full sensitivity analysis takes days of computer time on a single
# machine/processor because it involves a huge number of model runs.
# The example below is just a (relatively) quick minimalist demonstration and should NOT be
# taken as the basis for any analysis or conclusions.
# Even so, this minimalist demo could take 45 min to run to completion because it
# involves 1353 model runs.
# This example uses the likelihood of observed data at the analysis criterion since
# the argument outID is not set and hence adopts the default value 0.
sens_results <- e2e_run_sens(model, nyears=1, n_traj=3, csv.output=FALSE)
```

```
head(sens_results)

# A more realistic sensitivity analysis would be something like:
sens_results <- e2e_run_sens(model, nyears=50, n_traj=16, postprocess=TRUE, csv.output=TRUE)
# DO NOT launch this configuration unless you are prepared to wait many days for the results
#
# Example of parallelising the process:
# Launch two (or more) runs separately on different processors...
# Launch batch 1:
model1 <- e2e_read("North_Sea", "1970-1999", model.ident="BATCH1",silent=TRUE)
sens_results1 <- e2e_run_sens(model1, nyears=50, n_traj=10, coldstart=TRUE, postprocess=FALSE,
                              csv.output=TRUE)
     # On machine 1

model2 <- e2e_read("North_Sea", "1970-1999", model.ident="BATCH2",silent=TRUE)
sens_results2 <- e2e_run_sens(model2, nyears=50, n_traj=10, coldstart=FALSE, postprocess=FALSE,
                              csv.output=TRUE)
    # On machine 2
# Note that these two runs return only raw data since postprocess=FALSE
#
# Then, afterwards, merge the two raw results files with text-tags BATCH1 and BATCH2,
# and post process the combined file:
model3 <- e2e_read("North_Sea", "1970-1999", model.ident="COMBINED",silent=TRUE)
processed_data <- e2e_merge_sens_mc(model3, selection="SENS", ident.list<-c("BATCH1","BATCH2"),
                                    postprocess=TRUE, csv.output=TRUE)

# Plot a diagram of parameter sensitivities from the combined data
e2e_plot_sens_mc(model3, selection="SENS", use.example=FALSE)
```

The default numeric value of the argument outID = 0, corresponding to the analysis criterion being the likelihood of observed data given a parameter vector. To find the value of outID corresponding to some other criterion use the function e2e_get_senscrit(), which returns a dataframe object.

Table 24. Arguments of the function **e2e_get_senscrit()**.

Argument	Description
id	Choose single value from 0 : 247 = unique criterion, A = All (default), L = Likelihood, M = Annual average mass, F = Annual integrated fluxes

6.3 Performing a Monte Carlo analysis e2e_run_mc()

Run a Monte Carlo simulation with StrathE2E and derive centiles of credible values of model outputs.

Table 25. Arguments of the function **e2e_run_mc()**.

Argument	Description
model	R-list object generated by the e2e_read() function which defines the model configuration
nyears	Number of years to run each instance of the model. Needs to be long enough to allow the model to attain a stationary state (default=50)
baseline.mode	Logical. If TRUE then the simulation adopts a baseline-mode. If FALSE then scenario-mode (default=TRUE)

Argument	Description
use.example.baseparms	Logical. Value required only if baseline.mode=FALSE. If TRUE then the baseline parameters set is drawn from example data provided with the package. If FALSE then a user generated baseline parameter set is expected
baseparms.ident	Default = "". Value required if baseline.run=FALSE and use.example.baseparms=FALSE, in which case this is the model.ident string of the a user generated baseline parameter set. Remember to include the phrase within"" quotes
begin.sample	Default = 1. Value required if baseline.mode=FALSE. Value sets the first row in the file of baseline parameter data from which the sequence of parameters sets to be used in this run will be taken. Set a value > 1 if this run is part of a parallel set of runs
n_iter	Number of iterations of the model (default=1000). If baseline.mode=FALSE and the number of sets in the supplied baseline parameter file beyond the value of begin.sample is less than n_iter, then n_iter is reset to the remaining number available
csv.output	Logical. If TRUE then enable writing of csv output files (default=FALSE)
runtime.plot	Logical. If FALSE then disable runtime plotting of the progress of the run - useful for testing (default=TRUE)
postprocess	Logical. If FALSE then disable post-processing of the cumulative outputs, for example in parallel mode where multiple runs are going to be combined later (default=TRUE)

The Monte Carlo scheme provided with the StrathE2E2 package has two modes of operation. The first is referred to as "baseline-mode". This involves generating a list of ecology model parameter sets and, for each set, determining the likelihood of observational data consistent with the environmental and fishery drivers. The model outputs generated with each set are then weighted by the likelihood before deriving quantiles of their distribution. The quantile ranges then represent credible intervals of the model outputs.

The second mode of operation is referred to as the "scenario-mode". In this case, the parameter sets and their associated likelihoods from a baseline-mode simulation, are used to generate model outputs for scenarios of environmental or fishing drivers - e.g. increased activity by selected gears, or warmer sea temperatures. These scenario outputs are then weighted by the baseline-mode likelihoods before derivation of quantiles and credible intervals.

In baseline-mode, the Monte Carlo scheme generates an ensemble of model runs by sampling the ecology model parameters from uniform distibutions centred on an initial (baseline) parameter set. Ideally, this should be the maximum likelihood parameter set arrived at by prior application the various optimization functions in the package to 'fit' the model to a suite of observational data on the state of ecosystem (target data for fitting are in **/Modelname/Variantname/Target/annual_observed*.csv**). Each iteration in the Monte Carlo scheme then generates a unique time series of model outputs at daily intervals, together with the overall likelihood of the obervational target data. From these data, the function then derives credible intervals of each output by weighting the values from each parameter set by the associated likelihood.

Running a scenario-mode simulation requires a baseline model to have been previously completed, in order to generate a list of parameter sets and associated likelihoods. In scenario-mode, model driving data (e.g. temperatures or fishing activities) are varied from the baseline configuration by editing the model object generated by a **e2e_read()** function call, and the model is then run with the existing baseline model parameter sets, and the results weighted by the baseline-mode likelihoods in order to derive the credible intervals.

The choice of baseline vs scenario-mode of simulations is determined in the function by a logical argument setting 'baseline.mode'.

In baseline-mode, the coefficients of variation for jiggling the ecology parameter can be varied in real-time

during the run by editing the file **monte_carlo.csv** in the folder **/Param/control/** of the model version. However, it is recommended to leave the setting constant for the duration of a run. A CV of 0.10 to 0.15 is recommended. If comparing the credible intervals for two different baseline models then it is important to use the same CV in both cases.

In scenario model, the parameter sets are pre-ordained by a prior baseline simulation, so the CV setting is ineffective.

On completion of all the iterations of the model, whether in baseline or scenarion mode, for each model output variable in turn, values from the individual runs and their associated model likelihoods are assembled as a list of paired values. The list is then sorted by ascending values of the model variable, and the cumulative likelihoods with increasing value of model variable is calculated. Values for the model variable at standard proportions (0.005, 0.25, 0.5, 0.75, 0.995) of the maximum cumulative likelhood are then extracted by interpolation. These represent the credible intervals of model output given uncertainty in the ecology model parameters.

Outputs from the **e2e_run_mc()** function depend on whether post-processing is selected. If not, then raw data are saved as csv files (if csv.output=TRUE) and the function returns a dataframe of the parameter sets used in the run and their corresponding likelihoods. In scenario-mode this is just a replica of the baseline-mode data that have been used to run teh simulation. If post-processing is selected, then both the raw and processed credible interval data are saved to csv files (by dafault), and the function returns a list opject which includes the parameter dataframe plus all the processed credible interval data structures. csv files are saved in the folder /results/Modelname/Variantname/CredInt/ with an identifier for the simulation (model.ident) created by the **e2e_read()** function. There are two types of output. First is simply an accumulation of all the standard outputs from StrathE2E on a run-by-run basis. The second is a set of files containing the derived credible intervals.

The function displays a real-time graphic to show the progress of the simulation. During the StrathE2E-running phase of the process an x-y graph is updated after each iteration (x-axis=iterations, y-axis=Likelihood of the target data), with a horizontal grey line showing the baseline (maximum liklihood) result, and black symbols showing the likelihood for each iteration based on the parameter values sampled from the baseline. The y-axis limits for the graph can be varied in real-time during the run by editing the file **monte_carlo.csv** in the folder **/Param/control/** of the model version.

During the post-processing phase, a message is displayed as each output variable is processed.

WARNING - the scheme can take a long time to run (~2 days with the default settings), and generate large output files (11 files total 3.2 Mb per iteration). Check the memory capacity of you machine before starting a long run. It can make sense to parallelise the process by splitting across different processors and merging the results afterwards. The package contains a function for merging cumulative output files from multiple runs and post-processing the combined data.

In baseline-mode, parallel instances of the function can be implemented on different processors, all starting from the same initial parameter set. During the subsequent merging process, the outputs from the first parameter set, of all but the first instance, are stripped away and discarded.

Impementing parallel runs in scenario-mode requires a different approach. Here, the parameter sets are pre-ordained and it is important to avoid using duplicate sets in the simulations. Hence, the function includes an argument 'begin.sample' to select a pointer in the baseline parameter set to begin sampling in any scenario-mode run. For example, in the first instance, begin.sample=1, and the number of iterations (n_iter) might be set to e.g. 200. For the second instance the pointer (begin.sample) would then be set to 201, and so on. If begin.sample > 1, the function runs the first baseline parameter set (sample 1) first before proceeding to the pointer begin.sample and completing the requested number of iterations, so there will be one extra iteration in the output files. If the requested number of iterations means that sampling would over the end of available number of parameter sets then the number of iterations is reduced accordingly.

Although the **e2e_run_mc()** function generates large raw data files (11 files total 3.2 Mb per iteration), the processed data are much smaller (13 files total ~4 Mb regardless of the nunber of iterations).

```
# Load the 1970-1999 version of the North Sea model supplied with the package:
   model <- e2e_read("North_Sea", "1970-1999",silent=TRUE)
# Since results.path is not defined here the results will be written
# to the current temporary folder if csv.output=TRUE.
#
# Run the Monte Carlo process and generate some data but disable csv output
# A quick demonstration of baseline-mode with postprocessing and csv output disabled:
   demo <- e2e_run_mc(model,nyears=2,baseline.mode=TRUE,n_iter=10,csv.output=FALSE)
      str(demo,max.level=1)                    # View the structure of the returned list
      str(demo$CI_annual_avmass,max.level=1)   # View the structure of a returned list
                                               # element containing sub-sets
      str(demo$CI_annual_fluxes,max.level=1)   # View the structure of a returned list element
                                               # containing sub-sets
      head(demo$CI_annual_avmass$whole)        # View the top few rows on the whole domain data
                                               # on annual averge mass
#
# ------------------------------------------------------------------------------------
#
# A more meaningful run would be:
# WARNING: This run will take about 48h to complete, much better to split up and spread
# across multiple processors !
   mc_results <- e2e_run_mc(model,baseline.mode=TRUE,nyears=50,n_iter=1000,csv.output=TRUE)
#
# ------------------------------------------------------------------------------------
#
# A quick demonstration run in baseline-mode and save the data to csv files:
   basemodel <- e2e_read("North_Sea", "1970-1999",model.ident="mcbaseline",silent=TRUE)
   basedemo <- e2e_run_mc(basemodel,nyears=2,baseline.mode=TRUE,
                          n_iter=10,csv.output=TRUE)
#
# ------------------------------------------------------------------------------------
#
# Then a quick demonstration run in scenario-mode using the saved baseline parameter
# data, and save to csv:
# First create an extreme fishing scenario - quadruple some gear activities, run for 10 year
   scenariomodel<-basemodel
   scenariomodel$setup$model.ident <- "mcscenario"
   scenariomodel$data$fleet.model$gear_mult[1] <- 4
                              # Gear 1 (Pelagic trawls) activity rate rescaled to 4*baseline
   scenariomodel$data$fleet.model$gear_mult[4] <- 4
                              # Gear 4 (Beam_Trawl_BT1+BT2) activity rate rescaled to 4*baseline
   scendemo <- e2e_run_mc(scenariomodel,nyears=10,baseline.mode=FALSE,
                          use.example.baseparms=FALSE, baseparms.ident="mcbaseline",
                          begin.sample=1, n_iter=10,csv.output=TRUE)

# Compare the results of the baseline and scenario:
   basemodel <- e2e_read("North_Sea", "1970-1999", model.ident="mcbaseline",silent=TRUE)
   scenariomodel <- e2e_read("North_Sea", "1970-1999",model.ident="mcscenario",silent=TRUE)
   e2e_compare_runs_box(selection="ANNUAL", model1=basemodel, ci.data1=TRUE, use.saved1=TRUE,
#                          model2=scenariomodel, ci.data2=TRUE, use.saved2=TRUE )
   dev.new()
   e2e_compare_runs_box(selection="MONTHLY", model1=basemodel, ci.data1=TRUE, use.saved1=TRUE,
                        model2=scenariomodel, ci.data2=TRUE, use.saved2=TRUE )
```

```
#
# ------------------------------------------------------------------------------------
#
# Example of parallelising the process in baseline-mode:
# Launch batch 1:
   model1 <- e2e_read("North_Sea", "1970-1999", model.ident="BATCH1",silent=TRUE)
   results1 <- e2e_run_mc(model1,nyears=2,baseline.mode=TRUE,
                          n_iter=10,csv.output=TRUE,postprocess=FALSE)
# Launch batch 2 (on a different processor):
   model2 <- e2e_read("North_Sea", "1970-1999", model.ident="BATCH2",silent=TRUE)
   results2 <- e2e_run_mc(model2,nyears=2,baseline.mode=TRUE,
                          n_iter=10,csv.output=TRUE,postprocess=FALSE)
# Note that these two runs return only raw data since postprocess=FALSE
# Note that Batch 2 requests 11 iterations, rather that 10 in Batch 1. The number of
# iterations do not have to be the same in each batch. However, the first in each batch
# has to use the initial parameter set from the model setup, as this is the parent for
# all the subsequent samples generated during the run. When we come to merge the data
# from separate batches, data from the first iteration are stripped off and discarded
# for all but the first batch as we do not want to include duplicate data in the combined
# files. Hence we choose 11 iterations here in Batch 2 to make the point, and we expect
# the combined data to include 20 iterations.

# Then, afterwards, merge the two raw results files with text-tags BATCH1 and BATCH2,
# and post process the combined file:
   model3 <- e2e_read("North_Sea", "1970-1999", model.ident="COMBINED",silent=TRUE)
   processed_data <- e2e_merge_sens_mc(model3, selection="MC", ident.list<-c("BATCH1","BATCH2"),
                                       postprocess=TRUE, csv.output=TRUE)
# or...
   combined_data <- e2e_merge_sens_mc(model3, selection="MC", ident.list<-c("BATCH1","BATCH2"),
                                      postprocess=FALSE, csv.output=TRUE)
   processed_data<-e2e_process_sens_mc(model3, selection="MC", use.example=FALSE,csv.output=TRUE)
#
# Plot the parameter likelihood distributions from the combined data
   e2e_plot_sens_mc(model3, selection="MC")

# ------------------------------------------------------------------------------------

# Example of parallelising the process in scenario-mode, using the baseline parameter
# sets 'COMBINED' from above:
# The activity of all fishing gears is reduced to zero to create a no-fishing scenario.
# Run each batch for 10 years as a relatively quick demo - a real run would need to run
# for at least 40 year.
# Launch batch 1:
   model1s <- e2e_read("North_Sea", "1970-1999", model.ident="BATCH1_S",silent=TRUE)
   model1s$data$fleet.model$gear_mult[1:12] <- 0
                                      # Activity rates of all 12 gears rescaled to 0*baseline
   results1s <- e2e_run_mc(model1s,nyears=10,baseline.mode=FALSE, baseparms.ident="COMBINED",
                           begin.sample=1, n_iter=10,csv.output=TRUE,postprocess=FALSE)
# Launch batch 2 (on a different processor):
   model2s <- e2e_read("North_Sea", "1970-1999", model.ident="BATCH2_S",silent=TRUE)
   model2s$data$fleet.model$gear_mult[1:12] <- 0
                                      # Activity rates of all 12 gears rescaled to 0*baseline
   results2s <- e2e_run_mc(model2s,nyears=10,baseline.mode=FALSE,, baseparms.ident="COMBINED",
```

```
                              begin.sample=11, n_iter=10,csv.output=TRUE,postprocess=FALSE)
# Note that Batch 1 samples rows 1:10 of the baseline-mode parameter set archive "COMBINED"
# (begin.sample=1, n_iter=10), so Batch 2 needs to start sampling at row 11 (begin.sample=11).
# The baseline archive contains 20 rows, so Batch 2 has the capacity to undertake up to
# 10 sample iterations (rows 11:20). If we select more than 10 iterations (e.g. n_iter=15)
# then the code will automatically restrict to 10. Note that in fact, to be consistent with the
# format of output files from the baseline-mode, each scenario-mode run where
# 'begin.sample' > 1 will complete n_iter+1 iterations, by adding an initial run using the
# parameter values from row 1 of the baseline parameter # set - which is then stripped off
# during merging.

# Then, merge the two raw results files with text-tags BATCH1_S and BATCH2_S, and post
# process the combined file:
   model3s <- e2e_read("North_Sea", "1970-1999", model.ident="COMBINED_S",silent=TRUE)
   processed_datas <- e2e_merge_sens_mc(model3s, selection="MC",
                                        ident.list<-c("BATCH1_S","BATCH2_S"),
                                        postprocess=TRUE, csv.output=TRUE)

# Finally plot comparisons of the baseline and scenario model runs:
   e2e_compare_runs_box(selection="ANNUAL", model1=model3, ci.data1=TRUE, use.saved1=TRUE,
                        model2=model3s, ci.data2=TRUE, use.saved2=TRUE )
   dev.new()
   e2e_compare_runs_box(selection="MONTHLY", model1=model3, ci.data1=TRUE, use.saved1=TRUE,
                        model2=model3s, ci.data2=TRUE, use.saved2=TRUE )
#
```

6.4 Parallelisation of sensitivity and Monte Carlo analyses

Both the sensitivity and Monte Carlo analyses are extremely time consuming, especailly the sensitivity analysis which can take several day to complete a meaningful number of trajectories. This is beacause of the large number of replicate simulations that are required. However, the process is straightforward to parallelise on on multiple processors with the results being merged afterwards for processing.

6.4.1 Merging data generated on parallel processors e2e_merge_sens_mc()

Combine two or more sets of raw output data from parallel sensitivity or Monte Carlo analysis runs performed on separate machines/processors.

Table 26. Arguments of the function **e2e_merges_sens_mc()**.

Argument	Description
model	R-list object generated by the **e2e_read()** function which defines the model configuration
selection	Text string from a list to select the method for optimization. Select from: "SENS" or "MC", corresponding to output generated by the e2e_optomize_eco(), e2e_optomize_hr(), or e2e_optomize_act() function, Remember to include the phrase within "" quotes (default = "HR"). "HR" selects a scheme for optimizing the activity multipler to known harvest ratios and uses just the fishing fleet model. "ECO" selects a scheme for optimizing to the observational data in the state of the ecosystem and so uses both the fishing fleet and ecology models
ident.list	A vector of text variables corresponding to the "model.ident" identifiers for each of the files to merged (list must be length 2 or greater)
postprocess	Logical. if TRUE then process the results through to final data; if FALSE just produce the combined raw results (default=TRUE). The reason for NOT processing would be if there are further run results stil to be combined with the set produced by this function

Argument	Description
csv.output	Logical. If TRUE then enable writing of csv output files (default=FALSE)

The process of merging results files generated on parallel processors is not as simple as merely concatenating the files as it is necessary to keep track of the unique identities of the sets of trajectories.

The files to be combined must be transferred into the same folder, and this is where the new combined files will be placed. The path to locate the files is set in a e2e_read() function call. If not specified it is assumed that the files are located in the current temporary folder.

An identifying text string for the new combined files is set by the 'model.ident' argument in a **e2e_read()** function call.

The list of files to be combined (any number > 1) is defined by a vector of their individual 'model.ident' identifiers ('ident.list' argument).

When combining the files, the function creates a seamless sequence of trajectory identifiers through the combined data, beginning from 1 for the first baseline (maximum likelihood) trajectory of the first set.

If for any reason there is a need to combine separate batches of multiple run results, then post-processing can be delayed with the 'postprocess' argument until the last merge when all the data have been gathered together. Stand-alone postprocess can be performed using the function **e2e_process_sens_mc()**.

Provided that csv.output=TRUE, the function returns csv files of the merged data in with thw identifier model.ident as specified in a prior call of the **e2e_real()** function. If the argument postprocess=TRUE then a dataframe of processed output (for sensitivity analysis) or a list object of dataframes (for Monte Carlo analysis) is also returned.

Details relating to merging sensitivity analysis files:

e2e_run_sens() generates two output files per run - **OAT_results*.csv**, and **OAT_parameter_values*.csv**, where * is a model.ident text string set as an argument of the **e2e_read()** function. Thus function merges both of these types of files.

When combining sensitivity analysis data the first-named model.ident in the ident.list vector MUST correspond to a run of the **e2e_run_sens()** function with the argument coldstart=TRUE, and all others with coldstart=FALSE. This forces the first trajectory of sensitivity test to be performed on the baseline (e.g. maximum-likelihood) parameter set loaded with the initial **e2e_read()** function call. Thus is important for the post-processing stage of the analysis which needs to be performed on the combined results.

Details relating to merging Monte Carlo analysis files:

e2e_run_mc() generates 11 different output files of accumulated outputs from the iterations (total 3.2 Mb per iteration) covering the range of model outputs. This merging function combines batches of each of these types of files. Check the memory capacity of you machine before starting a long run or before merging runs. The processed output consists of 13 files total ~4 Mb regardless of the number of iterations.

```
# Example of parallelising the sensitivity analysis process:
# Launch two (or more) runs separately on different processors...
# Launch batch 1:
   model1 <- e2e_read("North_Sea", "1970-1999", model.ident="BATCH1",silent=TRUE)
# Since results.path is not defined here the results will be written
# to the current temporary folder if csv.output=TRUE.
   sens_results <- e2e_run_sens(model1, nyears=50, n_traj=10, coldstart=TRUE,
                           postprocess=FALSE,csv.output=TRUE)    ### On processor 1
# The output files are sent to the temporary folder.
# Note that coldstart=TRUE for the first batch only.
# Launch batch 2 (on a different processor):
```

```
   model2 <- e2e_read("North_Sea", "1970-1999", model.ident="BATCH2",silent=TRUE)
   sens_results <- e2e_run_sens(model2, nyears=50, n_traj=10, coldstart=FALSE,
                            postprocess=FALSE,csv.output=TRUE)    ### On processor 2
# Note that these two runs return only raw data since postprocess=FALSE

# Then, afterwards, merge the two raw results files with text-tags BATCH1 and BATCH2,
# and post process the combined file:
   model3 <- e2e_read("North_Sea", "1970-1999", model.ident="COMBINED",silent=TRUE)
   processed_data <- e2e_merge_sens_mc(model3, selection="SENS",
                                       ident.list<-c("BATCH1","BATCH2"),
                                       postprocess=TRUE, csv.output=TRUE)
# or...
   combined_data <- e2e_merge_sens_mc(model3, selection="SENS",
                                      ident.list<-c("BATCH1","BATCH2"),
                                      postprocess=FALSE, csv.output=TRUE)
   processed_data <- e2e_process_sens_mc(model3, selection="SENS",
                                         use.example=FALSE,csv.output=TRUE)


# Example of parallelising the Monte Carlo process:
# Launch batch 1:
   model1 <- e2e_read("North_Sea", "1970-1999", model.ident="BATCH1",silent=TRUE)
   results1 <- e2e_run_mc(model1,nyears=2,baseline.mode=TRUE,
                          n_iter=10,csv.output=TRUE,postprocess=FALSE)
# Launch batch 2 (on a different processor):
   model2 <- e2e_read("North_Sea", "1970-1999", model.ident="BATCH2",silent=TRUE)
   results2 <- e2e_run_mc(model2,nyears=2,baseline.mode=TRUE,
                          n_iter=11,csv.output=TRUE,postprocess=FALSE)
# Note that these two runs return only raw data since postprocess=FALSE
# Note 11 iterations in batch 2 - the first iteration wil be stripped off at merging so the
# combined data should include only 20 iterations.

# Then, afterwards, merge the two raw results files with text-tags BATCH1 and BATCH2,
# and post process the combined file:
   model3 <- e2e_read("North_Sea", "1970-1999", model.ident="COMBINED",silent=TRUE)
   processed_data <- e2e_merge_sens_mc(model3, selection="MC",
                                       ident.list<-c("BATCH1","BATCH2"),
                                       postprocess=TRUE, csv.output=TRUE)
# or...
   combined_data <- e2e_merge_sens_mc(model3, selection="MC",
                                      ident.list<-c("BATCH1","BATCH2"),
                                      postprocess=FALSE, csv.output=TRUE)
   processed_data<-e2e_process_sens_mc(model3, selection="MC",
                                       use.example=FALSE,csv.output=TRUE)
```

6.4.2 Processing data generated on parallel processors e2e_process_sens_mc()

Post-process already-created data from sensitivity or Monte Carlo analysis runs.

Table 27. Arguments of the function **e2e_merges_sens_mc()**.

Argument	Description
model	R-list object generated by the **e2e_read()** function which defines the model configuration

Argument	Description
selection	Text string from a list to select the method for optimization. Select from: "SENS" or "MC", corresponding to output generated by the e2e_optomize_eco(), e2e_optomize_hr(), or e2e_optomize_act() function, Remember to include the phrase within "" quotes (default = "HR"). "HR" selects a scheme for optimizing the activity multipler to known harvest ratios and uses just the fishing fleet model. "ECO" selects a scheme for optimizing to the observational data in the state of the ecosystem and so uses both the fishing fleet and ecology models
ident.list	A vector of text variables corresponding to the "model.ident" identifiers for each of the files to merged (list must be length 2 or greater)
use.example	Logical. If TRUE use pre-computed example data from the internal North Sea model rather than user-generated data (default=FALSE)
csv.output	Logical. If TRUE then enable writing of csv output files (default=FALSE)

The functions **e2e_run_sens()** and **e2e_run_mc()** are extremely time consuming so it makes sense to share the load across multiple processors in parallel and combine the results afterwards using the function **e2e_merge_sens_mc()**. Both the original run functions and the merging function have post-processing options in their workflow. Nevertheless, there may be occasions where it is necessary to simply post-process an already created raw data set. Hence the need for this function.

The model.ident identifier and folder path for input files must be set in a **e2e_read()** function call. If enabled, output files will be directed to the same folder.

Details relating to sensitivity analysis processing:

The function reads a file of raw sensitivity analysis data (OAT_results-*.csv, where* refers to the identifier model.ident set in the **e2e_read()** function) from a /results folder and performs the post-processing that would ordinarily be done automatically within the **e2e_run_sens()** or **e2e_merge_sens_mc()** functions. The output of the processing is a dataframe (and optionaly a csv file) containing the mean Elementary Effect (EE_mean) and the standard deviation of EE (EE_sd) for each parameter, sorted by the absolute value of EE_mean. EE_mean is an index of the magnitude of the sensitivity of a parameter, and EE_sd is an index of the extent of interaction with other parameters.

csv output from the function is a file named **sorted_parameter_elementary_effects*.csv** in the current folder **/results/Modelname/Variantname/**

Optionally, the function can read example raw sensitivity data for one of the two North Sea model variants supplied with the package.

For details of how the Elementary Effect values are derived for each parameter see help(e2e_run_sens)

Details relating to Monte Carlo analysis processing:

The function **e2e_run_mc()** generates 11 different output files covering the range of data generated by StrathE2E. The post processing reads all of these and generates quantiles of the likelihood weighted distributions of all of the outputs and saves these as credible interval files.

By default, the function writes output to csv files, but this can be disabled by a function argument. The function always returns all of the processed data as a list object. The function takes a few minutes to run as it has a lot of work to do, especially in processing all of the daily output variables.

Although the **e2e_run_mc()** function generates large raw data files (11 files total 3.2 Mb per iteration), the processed data are much smaller (13 files total ~4 Mb regardless of the nunber of iterations.

```
# Sensitivity analysis processing:
# Load details of the 1970-1999 version of the North Sea model supplied with the package:
model <- e2e_read("North_Sea", "1970-1999",silent=TRUE)
# Process the example data for this model variant provided with the package
```

```
   sens_results <- e2e_process_sens_mc(model, selection="SENS", use.example=TRUE)
   head(sens_results)
#'
# Monte Carlo analysis processing
# The example here assumes that raw results data have been previously
# generated by a of the e2e_run_mc() function, with the model.ident value
# assigned as "testdata", or that data from parallel runs have been gathered
# together in the same folder with model.ident="testdata".
# Load the model name and version for the data to be processed :
# e.g. ... model <- read_model("North_Sea", "2003-2013", model.ident="testdata")
   mc_results <- e2e_process_sens_mc(model, selection="MC", csv.output=TRUE)
   str(mc_results,max.level=1)
#
```

6.5 Diagnostics for sensitivity and Monte Carlo analyses e2e_plot_sens_mc()

Plot diagnostic data from sensitivity and Monte Carlo analyses.

Table 28. Arguments of the function **e2e_plot_sens_mc()**.

Argument	Description
model	R-list object generated by the **e2e_read()** function which defines the model configuration
selection	Text string from a list to select the method for optimization. Select from: "SENS" or "MC", corresponding to output generated by the e2e_optomize_eco(), e2e_optomize_hr(), or e2e_optomize_act() function, Remember to include the phrase within "" quotes (default = "HR"). "HR" selects a scheme for optimizing the activity multipler to known harvest ratios and uses just the fishing fleet model. "ECO" selects a scheme for optimizing to the observational data in the state of the ecosystem and so uses both the fishing fleet and ecology models
use.example	Logical. If TRUE use pre-computed example data from the internal North Sea model rather than user-generated data (default=FALSE)

Generates diagnostic plots from either a parameter sensitivity analysis or a Monte Carlo simulation of credible intervals of model outputs.

For both sensitivity and Monte Carlo analyses the inputs required by the function are a model list object created by the **e2e_read()** function defining the model configuration and relevant model.ident argument to point to the required files, and a logical argument to determne the origin of data files (either saved as csv files in the user workspace, or example data for the North Sea model provided with the package.

Details relating to sensitivity analysis plot:

The function reads processed results generated by the function **e2e_run_sens()**, or the function **e2e_process_sens_mc(...,selection="SENS")**, and plots the Elementary Effect mean (x-axis; magnitude of sensitivity) against Elementary Effect standard deviation (y-axis; strength of interactions between parameters).

Processed results from the function **e2e_run_sens()**, or the function **e2e_process_sens_mc(...,selection="SENS")**, are stored in the csv file /results/Modelname/Variantname/sorted_parameter_elementary_effects-*.csv* *where* represents the model run identifier (model.ident) text embedded in the R-list object created by the **e2e_read()** function.

Each symbol in the plot represents a single parameter in the model. The parameters are colour-coded to indicate 6 different types - fitted and fixed parameters of the ecology model, fishing fleet model parameters, fishery harvest ratios, environmental drivers, and physical configuration parameters.

The plot also shows a wedge formed by the two dashed lines. These correspond to ± 2 standard errors of the mean, so for points falling outside of the wedge there is a significant expectation that the distribution of elementary effects is non-zero. For points falling within the wedge the distribution of elementary effects is not significantly different from zero.

For details of how the Elementary Effect values are derived for each parameter see help(e2e_run_sens)

Details relating to Monte Carlo analysis plot:

The function creates a plot showing the credible distributions of ecology model parameters based on the results from the e2e_run_mc() function. These distributions are formed from the input distributions to the Monte Carlo process, weighted by the likelihood of observed target data on the state of the ecosystem given each combination of parameter values.

Post-processed data from the **e2e_run_mc()** function are stored in the file **/results/Modelname/Variantname/CredInt** where * represents the model run identifier (model.ident) text embedded in the R-list object created by the **e2e_read()** function.

Each parameter in the plot is scaled to its baseline value as (value - baseline)/baseline. Ideally, the baseline is the maximum likelihood model developed by application of the optimization functions available in the package (e.g. **e2e_optimize_eco()**). Each parameter is then represented by a box and whisker plot which shows the distribution of credible parameter values around zero, i.e. around the baseline. The median of the credible values distribution for each parameter is shown my a black tick-mark. The box spans the 50% credible interval (quartiles of the cumulative likelihood of simulated values). Whisker lines span the 99% credible interval.

The individual parameters are identified by numbers (rather than text names). These numbers correspond to the column numnbers in the file **/results/Modelname/Variantname/CredInt/CredInt_processed_parameters*.csv** Details of the parameters associated with each identification number are available as a dataframe by using the function **e2e_get_parmdoc()**.

The input distribution of parameter values to the Monte Carlo process is drawn from a random uniform distribution with limits specified in the CredIntSim_SD control file for the model setup (located in a sub-folder of the **/Param** folder). This distribution is shown by a red box and whisker at the bottom of the plot. Given the random uniform input we expect the quartiles (limits of the box) to be symetrical and located mid-way between zero and the upper and lower extremes. Vertical red lines show the expected limits of the quartiles boxes if model results were completely insensitive to individual parameter values.

The extent to which individual parameter distributions deviate from the random uniform input is an indication of their sensitivity in the model. Parameters whose distributions are tightly confined around zero (the baseline value) are highly sensitive.

For some parameters, in particular the preference parameters, their credible distributions may span a wider range than the inputs. This may seem unexpected, but arises because within each feeding guild the preference parameters are not independent of each other. The preferences within each guild must sum to 1. Hence, during the Monte Carlo process, after drawing new values of the preference values they are all rescaled to sum to 1, which may mean that some of them will have been varied by more than the original coefficient of variation of the input random uniform.

For details of how the distribution of credible output values from StrathE2E are calculated see the help information for the **e2e_run_mc()** function.

```
# Load the 1970-1999 version of the North Sea model supplied with the package:
   model <- e2e_read("North_Sea", "1970-1999",silent=TRUE)
#
# Either run the functions e2e_run_sens() to generate some results, or use the
# internal example data
#
# Plot the ecology model parameter distributions:
```

```
   e2e_plot_sens_mc(model, selection="SENS", use.example=TRUE)
Reading example results from StrathE2E2examples data package for the North_Sea 1970-1999 model
```

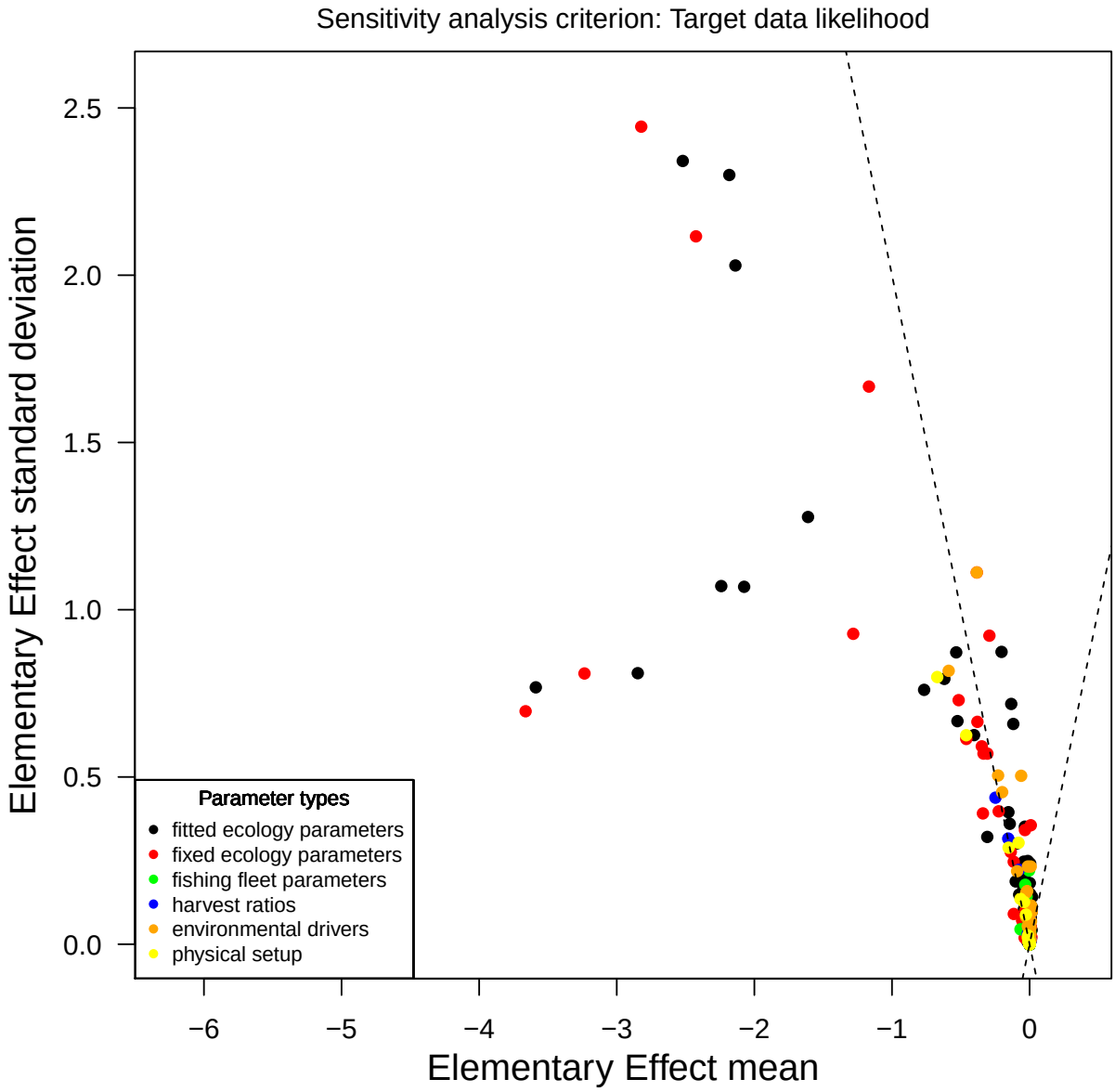


Figure 14. Sensitivity analysis plot for the 1970-1999 North Sea model.

```
# Load the 1970-1999 version of the North Sea model supplied with the package:
   model <- e2e_read("North_Sea", "1970-1999",silent=TRUE)
#
# Either run the functions e2e_run_mc() to generate some results, or use the
# internal example data
#
# Plot the ecology model parameter distributions:
   e2e_plot_sens_mc(model, selection="MC", use.example=TRUE)
```

```
Reading example results from StrathE2E2examples data package for the North_Sea 1970-1999 model
Reading example results from StrathE2E2examples data package for the North_Sea 1970-1999 model
```

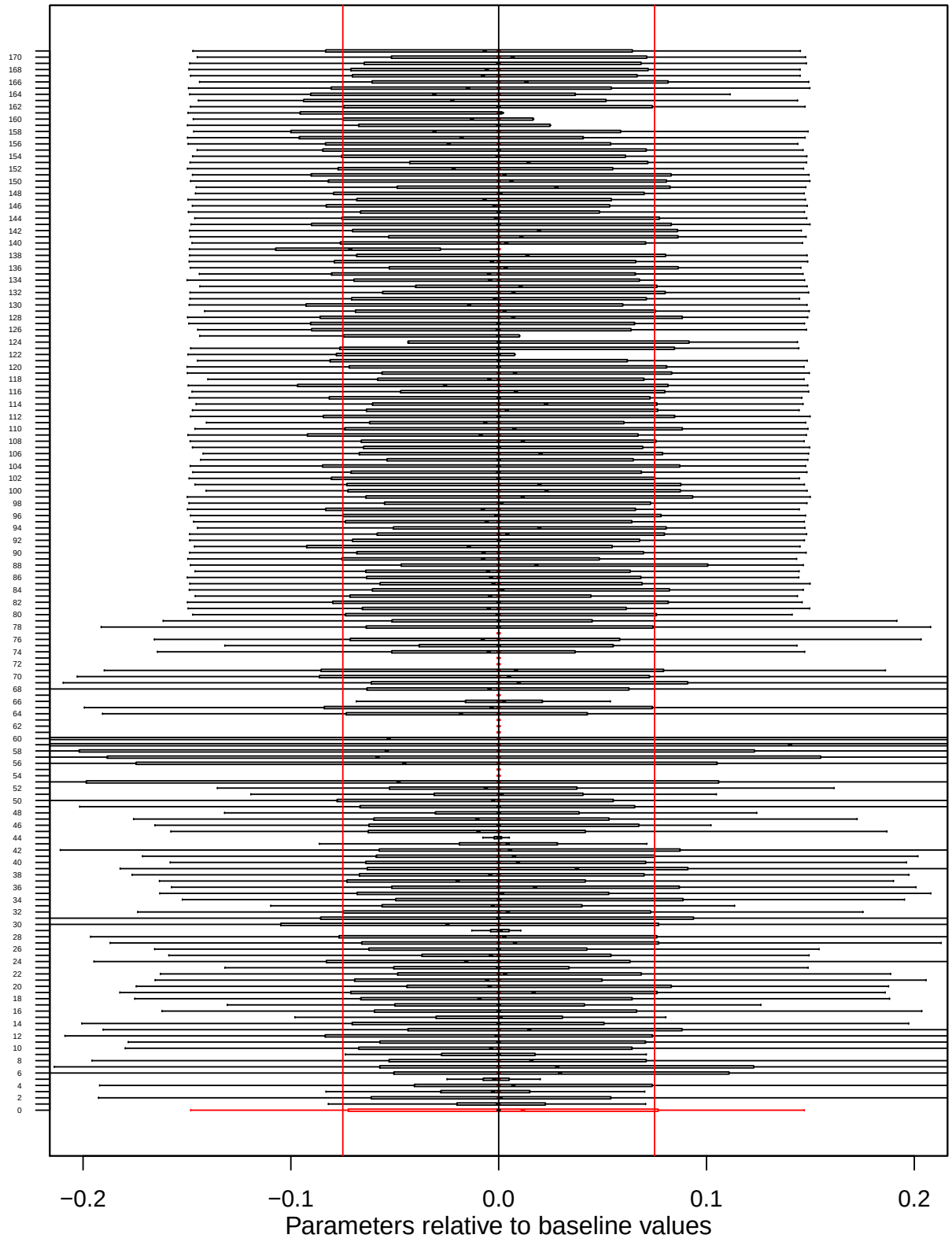


Figure 15. Parameter likelihood distributions from a baseline-mode Monte Carlo analysis with the 1970-1999 North Sea model.

7 Comparing model runs and observations

Comparing model results with observational data is a key part of the process of validation. Similarly, comparing outputs from different model runs is a key aspect of experimenting with the model.

The extensive model outputs provide scope for many creative ways of visualizing the relationships between models and observations and between models. Section 4 provides guidance of accessing the model outputs for users to develop their own graphics. This section describes a set of standard functions for displaying model and observational data that have proved useful.

7.1 Comparing model results with obseravtions e2e_compare_obs()

Box and whisker plots comparing annual or monthly observational data with corresponding model outputs.

Table 29. Arguments of the function **e2e_compare_obs()**.

Argument	Description
model	R-list object generated by the e2e_read() function which defines the model configuration
selection	Text string from a list to select comparison with annual or monthly observations. Select from: "ANNUAL", "MONTHLY". Remember to include the phrase within "" quotes
ci.data	Logical. If TRUE plot credible intervals around model results based on Monte Carlo simulation with the e2e_run_mc() function (default=FALSE)
use.saved	Logical. If TRUE use data from a prior user-defined run held as csv files data in the current results folder (default=FALSE)
use.example	Logical. If TRUE use pre-computed example data from the internal North Sea model rather than user-generated data (default=FALSE)
results	R-list object of model output generated by the e2e_run() function (default=NULL)

Generate a multi-panel set of box and whisker diagrams comparing annual or monthly averaged or integrated observational data on the state of the ecosystem with corresponding model outputs. Each panel displays a different category of observational and model data.

The observational data are read from the file **annual_observed*.csv** or **monthly_observed*.csv**in the /Targeta folder of the model setup defined by a **e2e_read()** function call. Column 3 of annual_observed* (header "Use_1.yes_0.no") is a flag to set whether any given row is used in calculating the likelihood of the observed data given the model setup by functions such as **e2e_optimize_eco()**. Un-used rows of data are omitted from the box and whisker plotting panels.

Arguments determine the source of model data for comparison with the observations. These can be outputs from a single model run with data held in memory as a list object or in a saved csv file, or from a Monte Carlo simulation (using the function e2e_run_mc()) to estimate credible intervals of model outputs. Generation of credible interval data is a long computing task, so example data for the North Sea model provided with the package for illustration.

In each plot panel, the distribution of observation data is shown by a black box plot (box spans 50% of obervations, whiskers span 99%, median indicated by a black tick mark). Corresponding data derived from model outputs are always shown in red - either a red tick mark for data from a single model run, or a red boxplot for likelihood weighted data from a Monte Carlo analysis.

```
# Load the 1970-1999 version of the North Sea model supplied with the package, run, and
# generate a plot. Note that these are the observational data that were used as the target
# for optimizing the model parameters
model <- e2e_read("North_Sea", "1970-1999",silent=TRUE)
results <- e2e_run(model, nyears=3)
e2e_compare_obs(selection="ANNUAL", model, results=results)
```

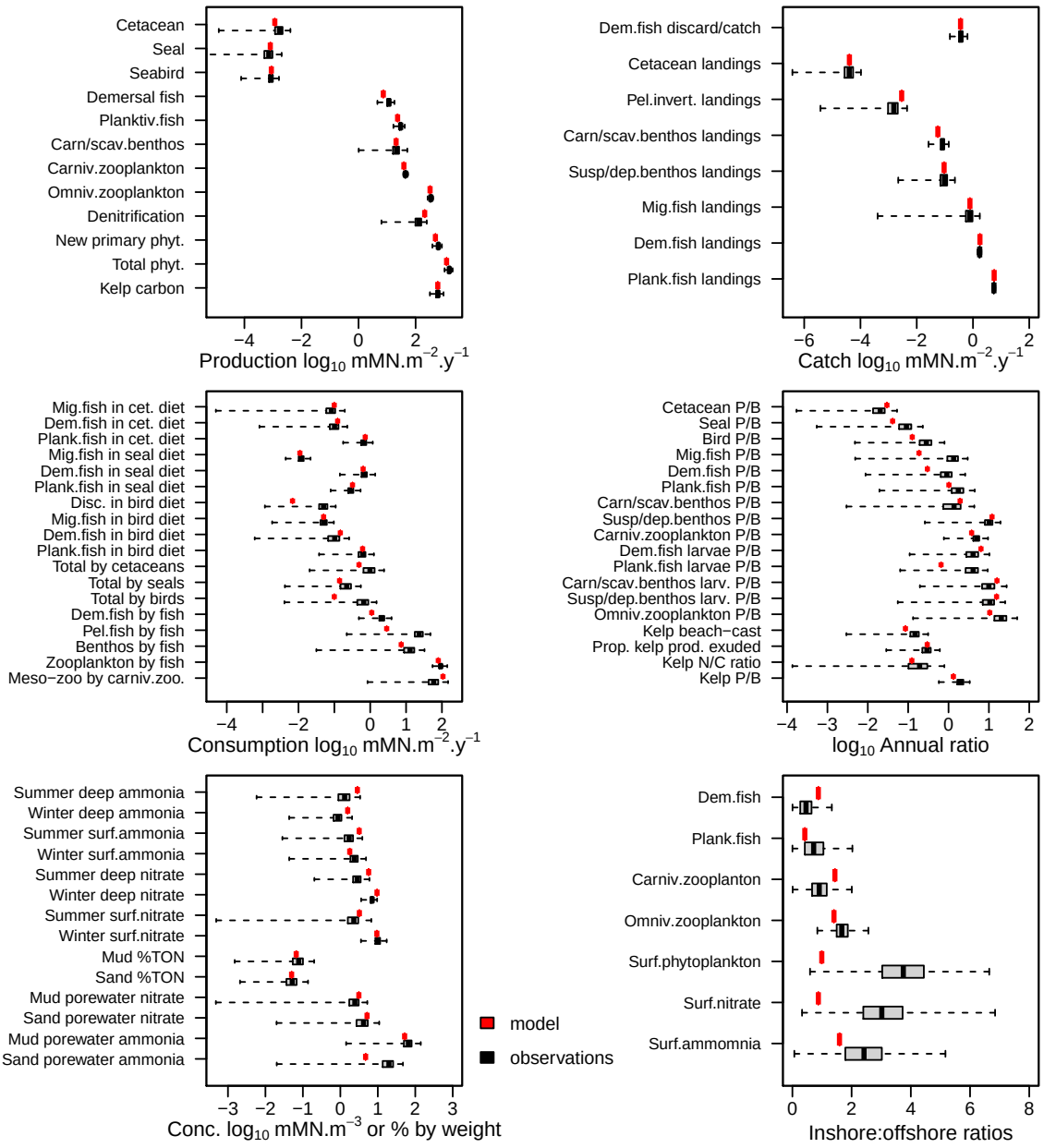


Figure 16. Comparison of 1970-1999 annual integrated or averaged observational data from the North Sea (black box-plots) with corresponding values derived from a single run of the model (red symbols). For the observations, the box spans 50% of the distribution of observations and the whiskers 99%. The median is shown by a black tick mark.

```
# Compare results from the 1970-1999 version of the North Sea model with monthly averaged
# data. Note that these are the observational data that were NOT used for optimizing the model
# parameters
e2e_compare_obs(model, selection="MONTHLY",results=results)
```

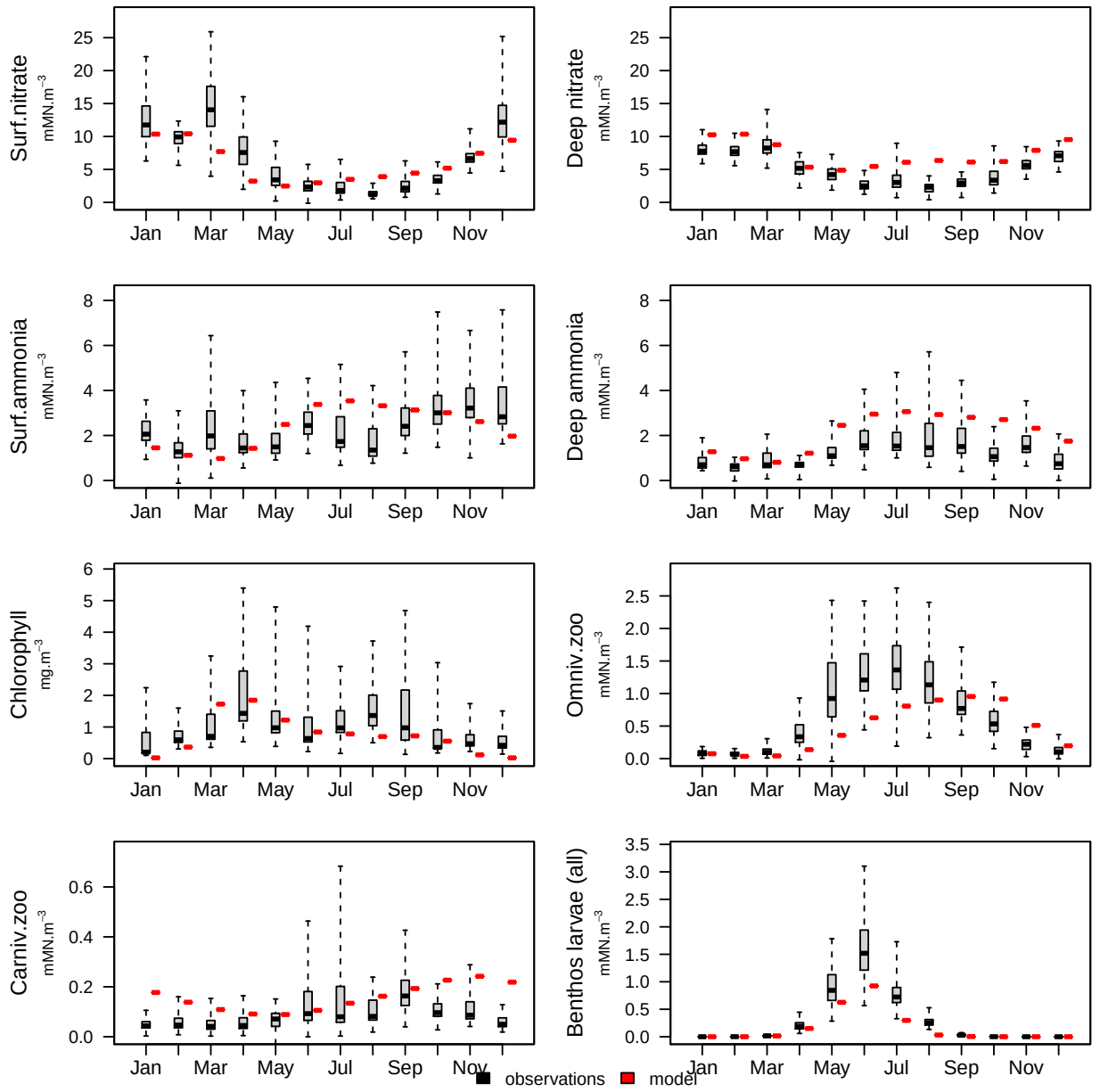


Figure 17. Comparison of 1970-1999 monthly averaged observational data from the North Sea (black boxplots) with corresponding values derived from a single run of the model (red symbols). For the observations, the box spans 50% of the distribution of observations and the whiskers 99%. The median is shown by a black tick mark.

```
# As above, but plotting credible intervals of model data derived from a baseline-mode
# Monte Carlo analysis:
model <- e2e_read("North_Sea", "1970-1999",silent=TRUE)
e2e_compare_obs(model, selection="ANNUAL",ci.data=TRUE,use.example=TRUE)
Reading example results from StrathE2E2examples data package for the North_Sea 1970-1999 model
```

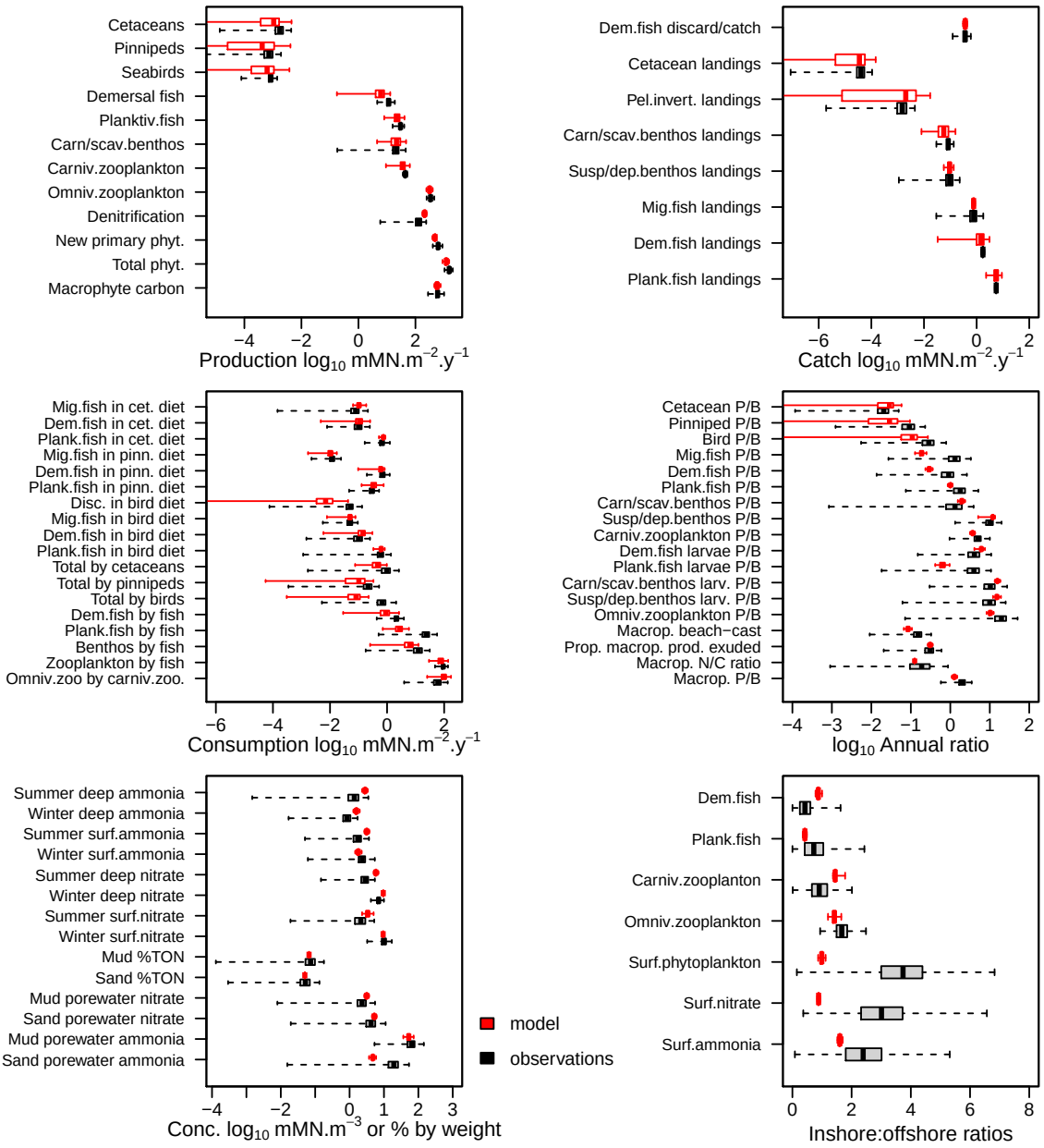


Figure 18. Comparison of 1970-1999 annual integrated or averaged observational data from the North Sea (black box-plots) with credible intervals of corresponding values derived from a Monte Carlo analysis of the model (red boxplots). For both observations and model outputs, the box spans 50% of the distribution of observations and the whiskers 99%. The median is shown by a tick mark.

```
e2e_compare_obs(selection="MONTHLY", model, ci.data=TRUE,use.example=TRUE)
Reading example results from StrathE2E2examples data package for the North_Sea 1970-1999 model
```

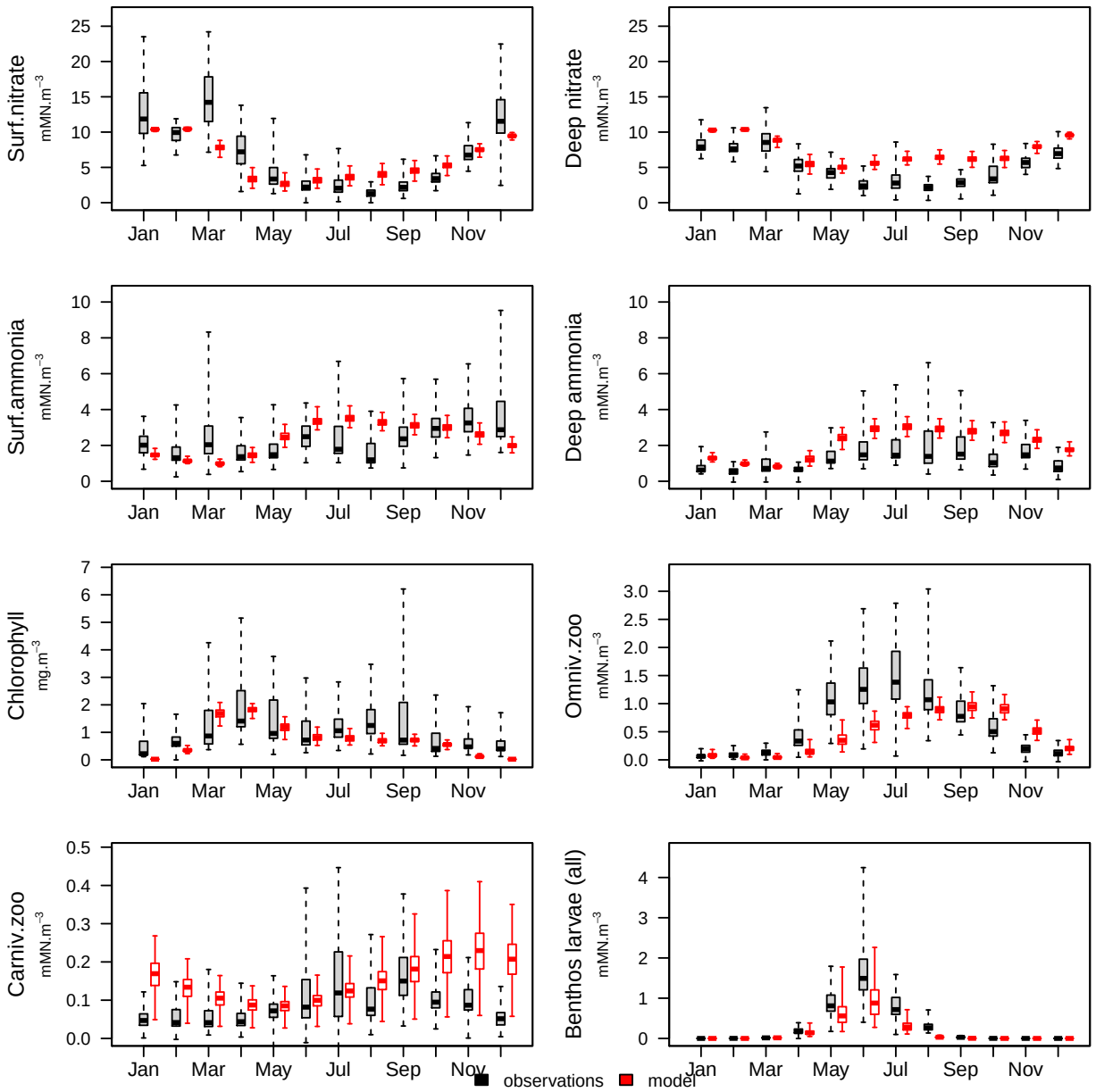


Figure 19. Comparison of 1970-1999 monthly averaged observational data from the North Sea (black box-plots) with credible intervals of corresponding values derived from a Monte Carlo analysis of the model (red boxplots). For both observations and model outputs, the box spans 50% of the distribution of observations and the whiskers 99%. The median is shown by a tick mark.

7.2 Comparing baseline and scenario model results

Two functions are provided for comapring baseline and scenario model runs. The first relies on a boxplot format similar to that used for comapring model runs with observations. The second utilizes tornado barplots which are widely used in sensitivity analysis.

7.2.1 Compare baseline and scenario models with boxplots e2e_compare_runs_box()

Box and whisker plots comparing annual or monthly outputs from single or Monte Carlo baseline and scenario model runs.

Table 30. Arguments of the function **e2e_compare_runs_box()**.

Argument	Description
selection	Text string from a list to select comparison with annual or monthly observations. Select from: "ANNUAL", "MONTHLY". Remember to include the phrase within "" quotes
model1	R-list object defining the baseline model configuration compiled by the e2e_read() function
ci.data1	Logical. If TRUE plot credible intervals around baseline model results based on Monte Carlo simulation with the e2e_run_mc() function (default=FALSE)
use.saved1	Logical. If TRUE use baseline data from a prior user-defined run held as csv files data in the current results folder (default=FALSE)
use.example1	Logical. If TRUE use pre-computed example data from the internal North Sea model as the baseline rather than user-generated data (default=FALSE)
results1	R-list object of baseline model output generated by the e2e_run() function. Only needed if ci.data1=FALSE, use.saved1=FALSE and use.example1=FALSE. (Default=NULL)
model2	R-list object defining the scenario model configuration compiled by the e2e_read() function
ci.data2	Logical. If TRUE plot credible intervals around scenario model results based on Monte Carlo simulation with the e2e_run_mc() function (default=FALSE)
use.saved2	Logical. If TRUE use scenario data from a prior user-defined run held as csv files data in the current results folder (default=FALSE)
use.example2	Logical. If TRUE use pre-computed example data from the internal North Sea model as the scenario rather than user-generated data (default=FALSE)
results2	R-list object of scenario model output generated by the e2e_run() function. Only needed if ci.data1=FALSE, use.saved1=FALSE and use.example1=FALSE. (Default=NULL)

Generate a multi-panel set of box and whisker diagrams comparing annual or monthly averaged or integrated model outputs from single or Monte Carlo baseline (black) and scenario (red) model runs. Each panel of annual data displays a different category of model data; each panel of monthly data shows a different nutrient or plankton guild.

Arguments determine the source of model data for comparison. These can be outputs from a Monte Carlo simulations (using the function e2e_run_mc()) to estimate credible intervals of model outputs, or from single model runs using e2e_run(). Generation of credible interval data is a long computing task, so example data for the North Sea model provided with the package are available as an illustration.

In each plot panel, where credible intervals of model outputs (from Monte Carlo analysis) have been selected the box spans 50% of the lielihood distribution of data, whiskers span 99%, and the median is indicated by tick mark). Baseline model results are always shown in black, scenario results in red.

```
# Load the 1970-1999 version of the North Sea model supplied with the package and
# run for 3 years
m1 <- e2e_read("North_Sea", "1970-1999",silent=TRUE)
r1<-e2e_run(m1,nyears=3)
# Load the 2003-2013 version of the North Sea model supplied with the package and
# run for 3 years
m2 <- e2e_read("North_Sea", "2003-2013",silent=TRUE)
r2<-e2e_run(m2,nyears=3)
# Compare 1970-1999 as baseline (from example data with cred.int.), with 2003-2013 as
# sceanrio (from single model run)
e2e_compare_runs_box(selection="ANNUAL",  model1=m1, ci.data1=TRUE, use.example1=TRUE,
                     model2=m2, ci.data2=FALSE, results2=r2)
```

```
Reading example results from StrathE2E2examples data package for the North_Sea 1970-1999 model
```

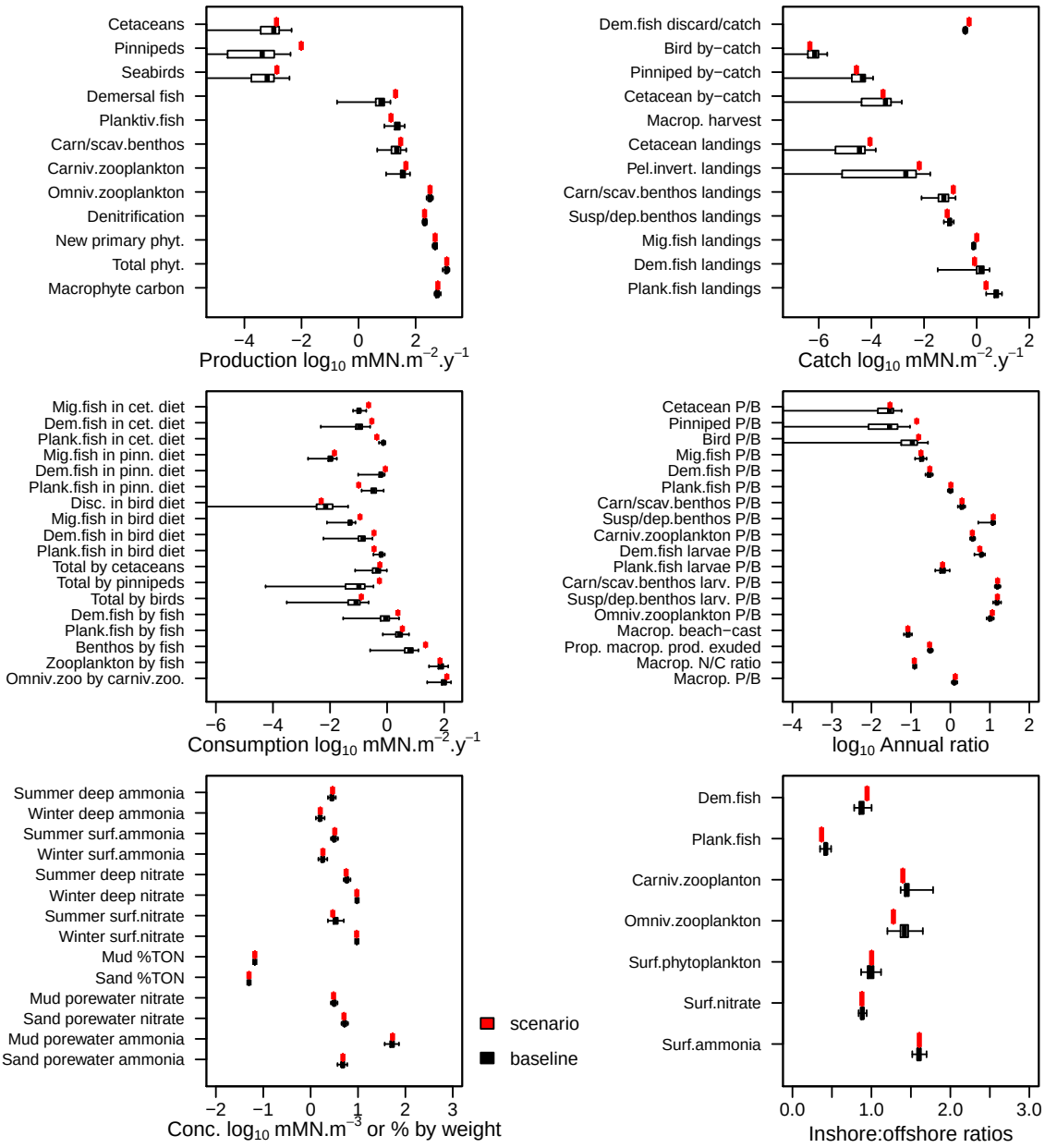


Figure 20. Comparison of 1970-1999 annual integrated or averaged data from a single run of the North Sea (baseline model, black symbols) with corresponding values derived from a single run of the 2003-2013 model (scenario model, red symbols).

```
e2e_compare_runs_box(selection="MONTHLY", model1=m1, ci.data1=TRUE, use.example1=TRUE,
                     model2=m2, ci.data2=FALSE, results2=r2)
Reading example results from StrathE2E2examples data package for the North_Sea 1970-1999 model
```

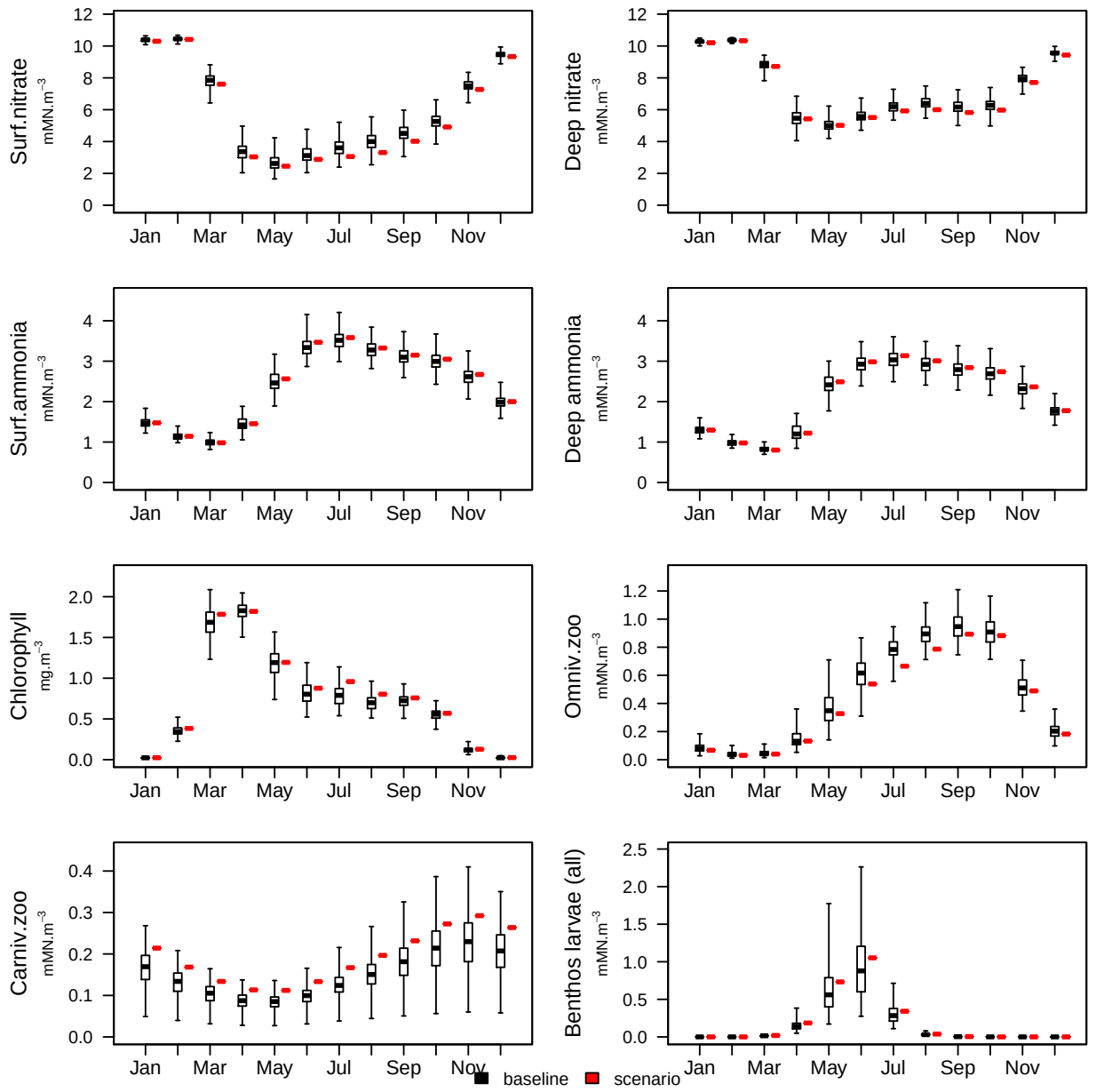


Figure 21. Comparison of 1970-1999 monthly averaged data from a single run of the North Sea (baseline model, black symbols) with corresponding values derived from a single run of the 2003-2013 model (scenario model, red symbols).

```
# Compare 1970-1999 as baseline (from single model run), with 2003-2013 as
# sceanrio (from example data with cred.int.)
e2e_compare_runs_box(selection="ANNUAL",  model1=m1, ci.data1=FALSE, results1=r1,
                     model2=m2, ci.data2=TRUE, use.example2=TRUE)
Reading example results from StrathE2E2examples data package for the North_Sea 2003-2013 model
```

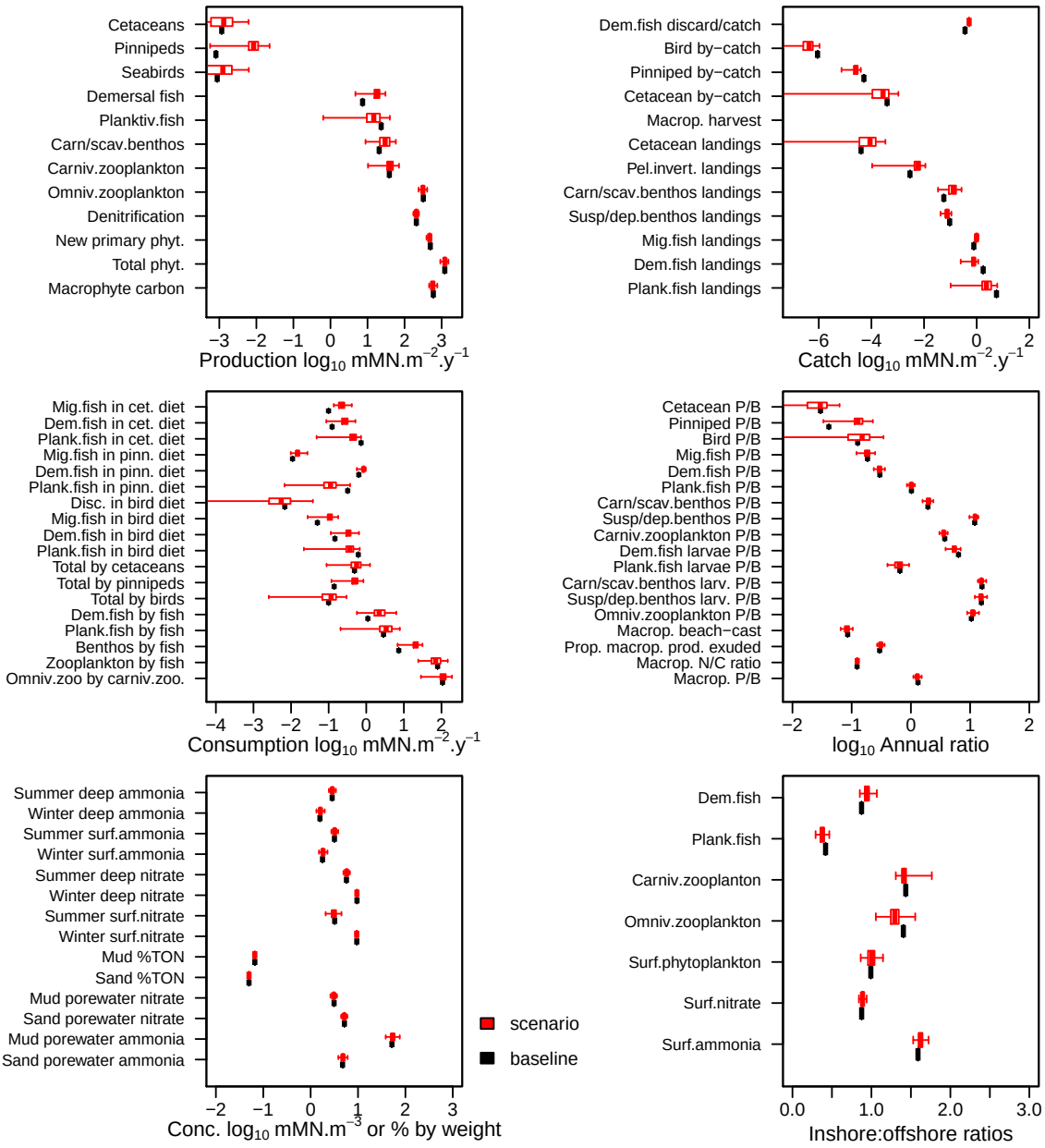


Figure 22. Comparison of 1970-1999 annual integrated or averaged data from a single run of the North Sea (baseline model, black symbols) with corresponding values derived from a Monte Carlo analysis of the 2003-2013 model (scenario model, red boxplots). The scenario model boxes span 50% of the distribution of likelihood weighted model results and the whiskers 99%. The median is shown by a tick mark.

```
e2e_compare_runs_box(selection="MONTHLY", model1=m1, ci.data1=FALSE, results1=r1,
                     model2=m2, ci.data2=TRUE, use.example2=TRUE)
Reading example results from StrathE2E2examples data package for the North_Sea 2003-2013 model
```

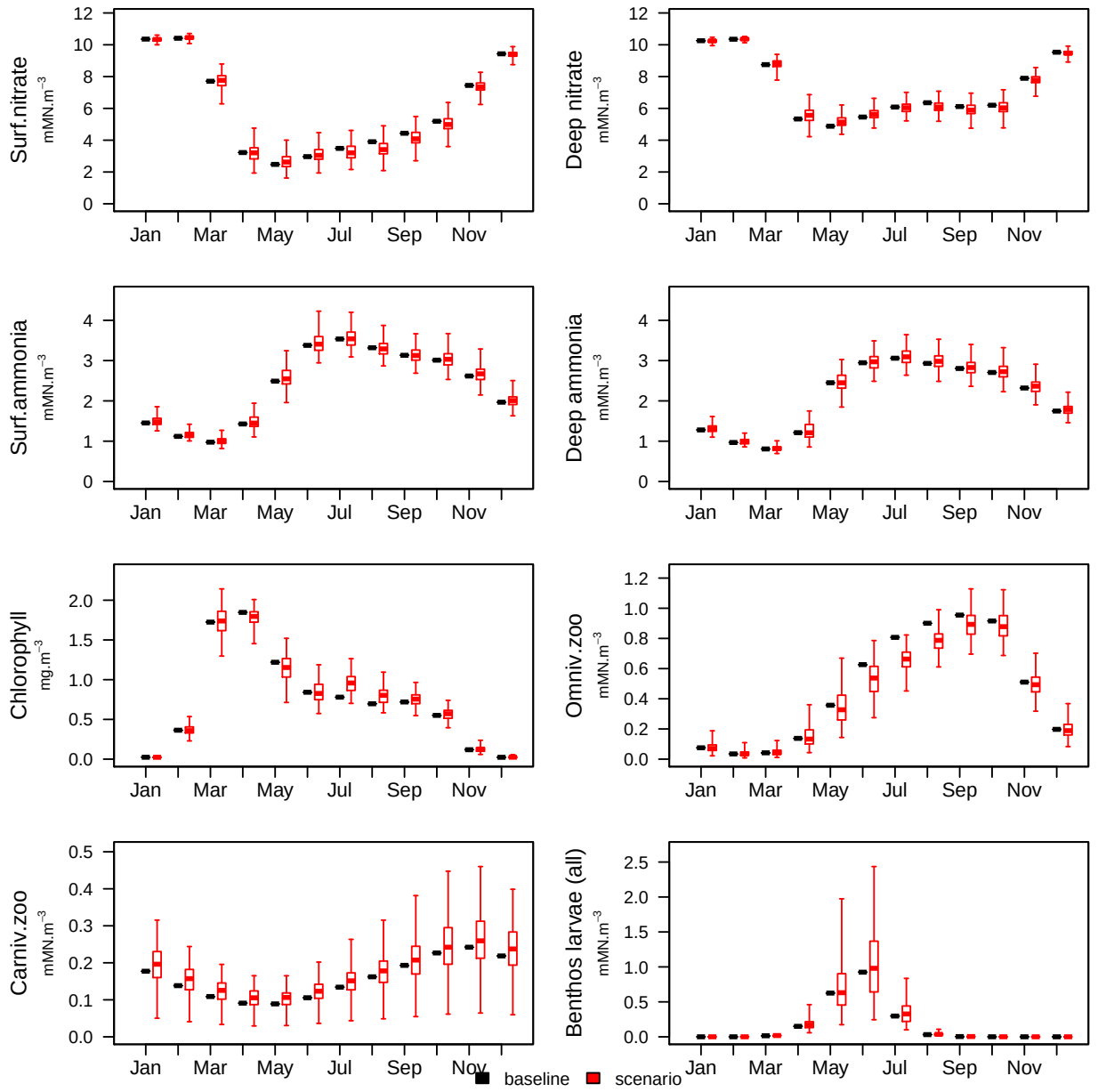


Figure 23. Comparison of 1970-1999 monthly averaged data from a single run of the North Sea (baseline model, black symbols) with corresponding values derived from a Monte Carlo analysis of the 2003-2013 model (scenario model, red boxplots). The scenario model boxes span 50% of the distribution of likelihood weighted model results and the whiskers 99%. The median is shown by a tick mark.

```
# Compare 1970-1999 as baseline (from example data with cred.int.), with 2003-2013
# as sceanrio (from example data with cred.int.)
e2e_compare_runs_box(selection="ANNUAL", model1=m1, ci.data1=TRUE, use.example1=TRUE,
                     model2=m2, ci.data2=TRUE, use.example2=TRUE )
Reading example results from StrathE2E2examples data package for the North_Sea 1970-1999 model
Reading example results from StrathE2E2examples data package for the North_Sea 2003-2013 model
```

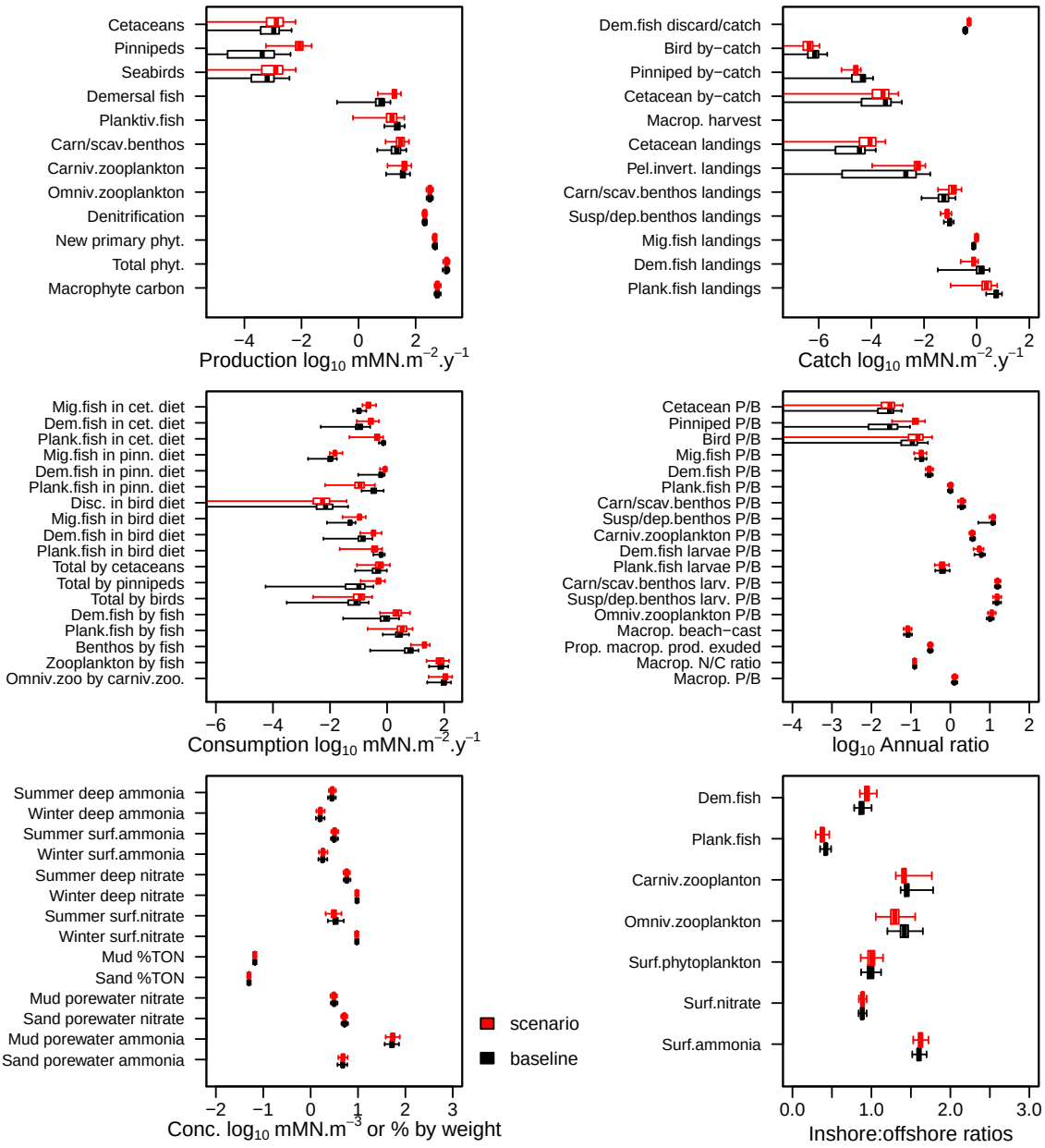


Figure 24. Comparison of 1970-1999 annual integrated or averaged data from a Monte Carlo analysis of the North Sea (baseline model, black boxplots) with corresponding values derived from a Monte Carlo analysis of the 2003-2013 model (scenario model, red boxplots). In both cases the boxes span 50% of the distribution of likelihood weighted model results and the whiskers 99%. The median is shown by a tick mark.

```
e2e_compare_runs_box(selection="MONTHLY", model1=m1, ci.data1=TRUE, use.example1=TRUE,
                     model2=m2, ci.data2=TRUE, use.example2=TRUE )
Reading example results from StrathE2E2examples data package for the North_Sea 1970-1999 model
Reading example results from StrathE2E2examples data package for the North_Sea 2003-2013 model
```

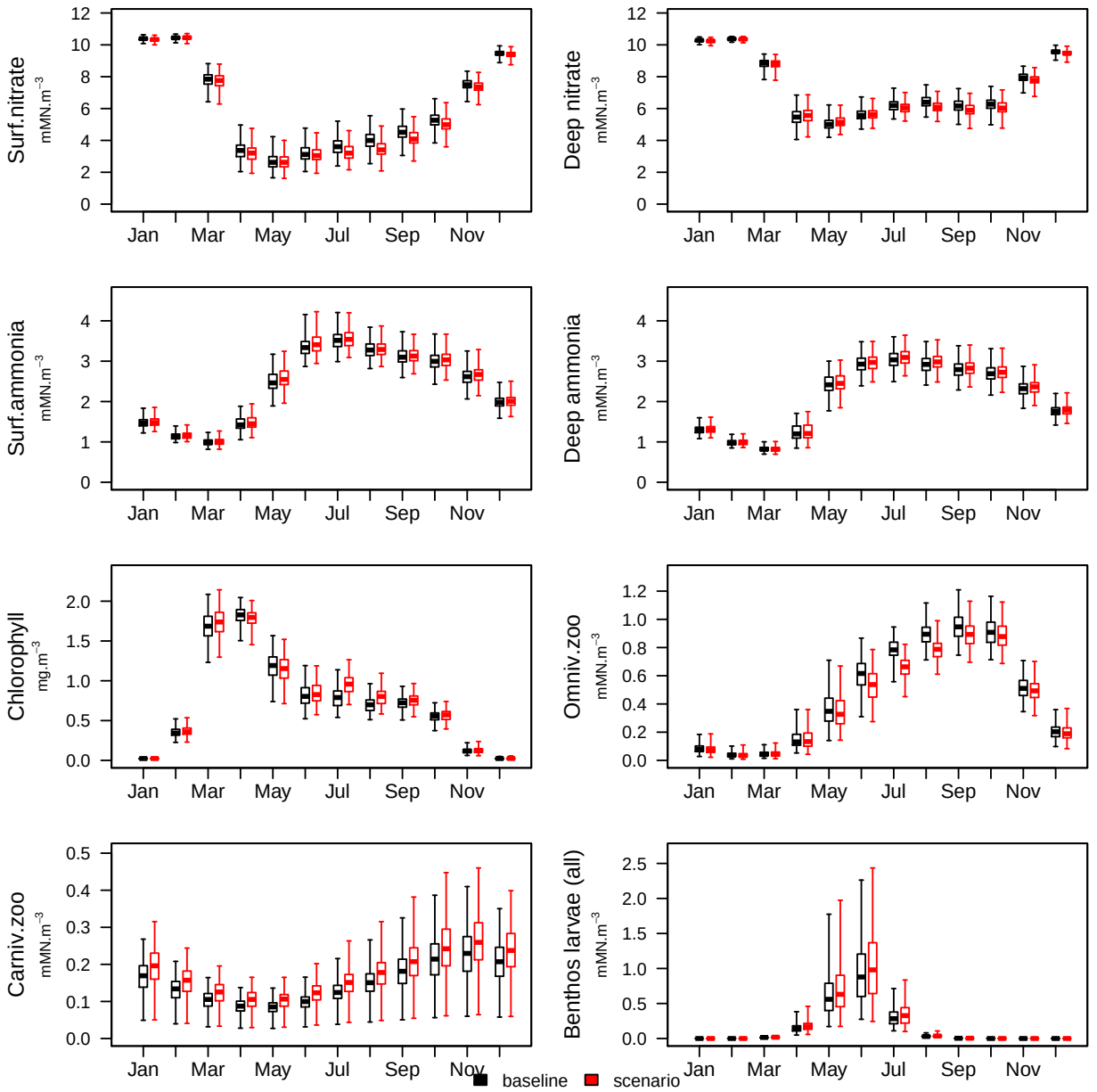


Figure 25. Comparison of 1970-1999 monthly averaged data from a Monte Carlo analysis of the North Sea (baseline model, black boxplots) with corresponding values derived from a Monte Carlo analysis of the 2003-2013 model (scenario model, red boxplots). In both cases the boxes span 50% of the distribution of likelihood weighted model results and the whiskers 99%. The median is shown by a tick mark.

7.2.2 Compare baseline and scenario models using barplots e2e_compare_runs_bar()

Create a tornado bar-plots of the differences between two model runs.

Table 31. Arguments of the function **e2e_compare_runs_bar()**.

Argument	Description
selection	Text string from a list identifying which type of data are to be plotted. Select from: "AAM", "CATCH", corresponding to annual average mass data, and catch data respectively (default = "AAM"). Remember to include the phrase within "" quotes

Argument	Description
model1	R-list object defining the baseline model configuration compiled by the e2e_read() function
use.saved1	Logical. If TRUE use baseline data from a prior user-defined run held as csv files data in the current results folder (default=FALSE)
results1	R-list object of baseline model output generated by the e2e_run() function. Only needed if ci.data1=FALSE, use.saved1=FALSE and use.example1=FALSE. (Default=NULL)
model2	R-list object defining the scenario model configuration compiled by the e2e_read() function
use.saved2	Logical. If TRUE use scenario data from a prior user-defined run held as csv files data in the current results folder (default=FALSE)
results2	R-list object of scenario model output generated by the e2e_run() function. Only needed if ci.data1=FALSE, use.saved1=FALSE and use.example1=FALSE. (Default=NULL)
log.pc	Value="LG" for data to be plotted on a log10 scale, value = "PC" for data to be plotted on a percentage difference scale (default = "PC")
zone	Value = "O" for offshore, "I" for inshore, or "W" for whole model domain (all upper case) (default = "W")
bpmin	Axis minimum for plot - i.e. the maximum NEGATIVE value of (scenario-baseline). Default = -50, given log.pc="PC" (percentage differences). Needs to be reset to e.g. -0.3 if log.pc="LG" (log scale)
bpmax	Axis maximum for plot - i.e. the maximum POSITIVE value of (scenario-baseline). Default = +50, given log.pc="PC" (percentage differences). Needs to be reset to e.g. +0.3 if log.pc="LG" (log scale)
maintitle	An optional descriptive text field (in quotes) to be added above the plot. Keep to 45 characters including spaces (default="")

Create a tornado bar-plot diagram of the differences in either annual average masses of ecology model varibles, or annual fishery catches, between two different runs of the StrathE2E model, referred to as baseline and sceanrio runs.

A tornado plot is a horizontal barplot which shows the difference between baseline and sceanrio runs. Bars to the right indicate scenario values greater than the baseline, bars to the left indicate scenario values less than baseline. In this function the difference between scenario and baseline data is calculated as either ((scenario - baseline)/baseline) and plotted on a linear (percentage) scale, or (scenario/baseline) and plotted on a log10 scale. In both cases, zero represents scenario = baseline.

The choice of whether to plot data on the masses of ecosystem components, or fishery catches, is determined by an argument setting in the function call. In either case two panels of tornado diagrams are plotted. In the case of mass data, the panels show data on water column components of the ecoststem (upper panel) and seabed components of the system (lower panel). In the case of catch data, the upper panel shows landings, the lower shows discards. For mass data and landings, positive values (scenario > baseline) are indicated by green bars to the right, negative values (scenario < baseline) by red bars to the left. For discards, positive values are in black, negative in grey. For any bars extending outsided the selected plotting range, the numeric value is displayed inside the relevant bar.

The function accepts data from either saved csv files created by the e2e_run() function, or directly from e2e_run() results objects. The relevant csv files are: **/results/Modelname/Variantname/ZONE_model_anav_bioma** where ZONE is INSHORE, OFFSHORE or WHOLEDOMAIN and * represents the model run identifier (model.ident) embedded in the R-list object created by the e2e_read() function.

In addition to generating graphics output the function returns a list object of the data presented in the plot. The object comprises two dataframes (changewater and changeseabed where annual average mass data are selected; changeland and changedisc where catch data are selected). The first column in each dataframe is the proportional difference expressed on a log10 scale, the second column as a percentage.

```
# Load the 1970-1999 version of the North Sea model supplied with the package.
# Run and save csv outputs to a temporary folder:
```

```
base_model   <- e2e_read("North_Sea", "1970-1999",model.ident="baseline",silent=TRUE)
base_results <- e2e_run(base_model,nyears=3, csv.output=TRUE)

# Create a scenario run from 1970-1999 baseline:
scen1_model   <- base_model   # Copies the baseline configuration into a new model object
scen1_model$setup$model.ident <- "scenario1"
scen1_model$data$fleet.model$gear_mult[4] <- 0.5
                              # Gear 4 (Beam_Trawl_BT1+BT2) activity rescaled to 0.5*baseline
scen1_results <- e2e_run(scen1_model,nyears=30)
#
#Compare the annual average mass scenario data with the baseline
mdiff_results1 <- e2e_compare_runs_bar(selection="AAM",
                                       model1=NA, use.saved1=FALSE, results1=base_results,
                        model2=NA, use.saved2=FALSE, results2=scen1_results,
                                    log.pc="PC", zone="W",
                                bpmin=(-40),bpmax=(+40),
                                    maintitle="Beam Trawl activity reduced by half")
[1] "Using baseline data held in memory from an existing model run"
[1] "Using scenario data held in memory from an existing model run"
```

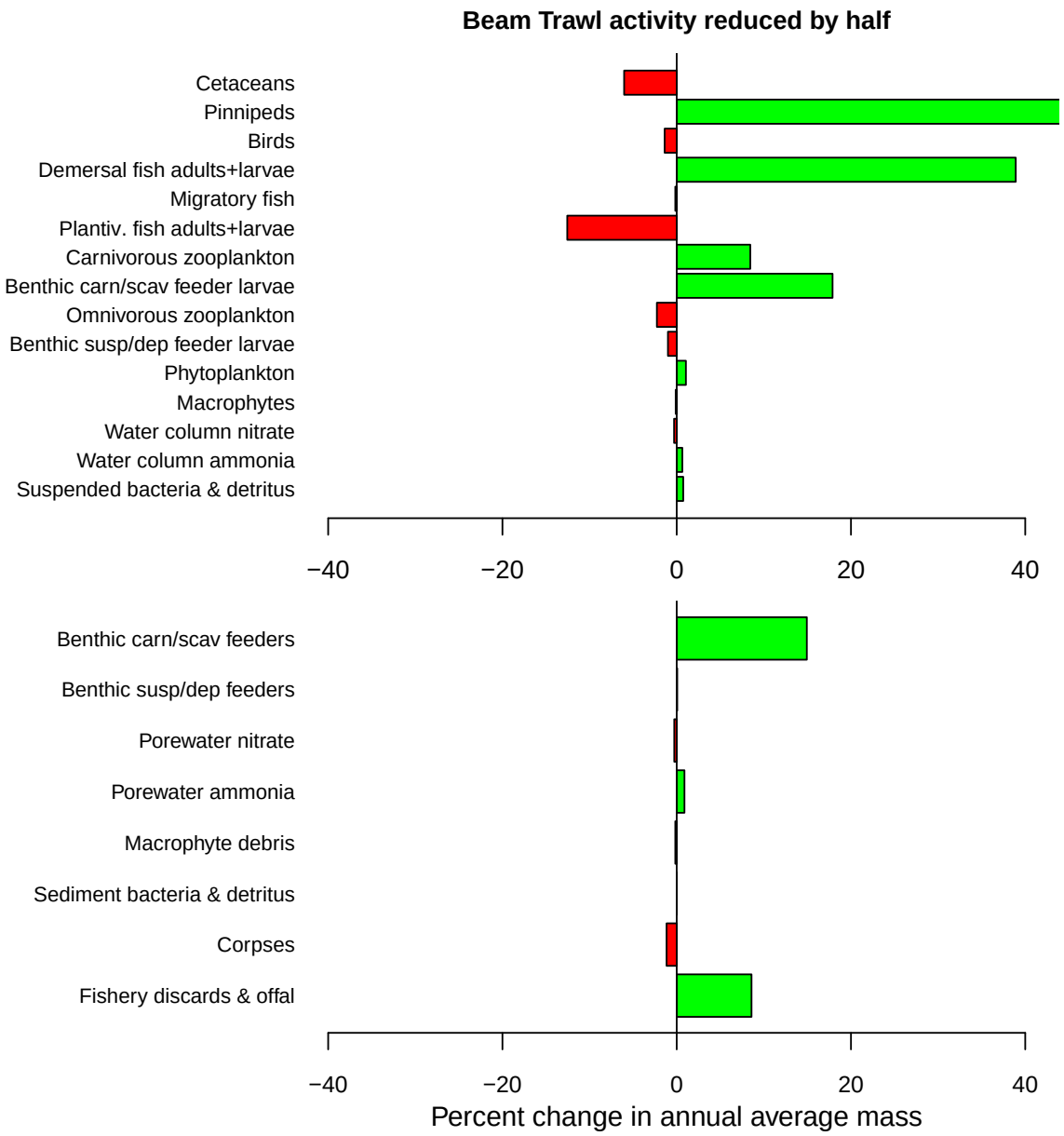


Figure 26. Tornado plot comparison of annual average masses of ecosystem components in a baseline model (North Sea, 1970-1999) with a scenario model in which beam trawl activity is reduced by half. Red bars to the left indicate that the mass was lower in the scenario model than the baseline; green bars to the right indicate higher in the scenario model.

```
str(mdiff_results1,max.level=1)   # View the structure of the returned list object
List of 2
 $ changewater :'data.frame':   15 obs. of  2 variables:
 $ changeseabed:'data.frame':   8 obs. of  2 variables:
```

```
#Create a second sceanario run
scen2_model   <- base_model     # Copies the baseline configuration into a new model object
scen2_model$setup$model.ident <- "scenario2"
scen2_model$data$fleet.model$gear_mult[1] <- 0.5
                                # Gear 1 (Pelagic_Trawl+Seine) activity rescaled to 0.5*baseline
```

```
scen2_results <- e2e_run(scen2_model,nyears=30)

#Compare the annual catches in the new scenario with the baseline previously saved in csv files
mdiff_results2 <- e2e_compare_runs_bar(selection="CATCH",
                          model1=base_model, use.saved1=TRUE, results1=NA,
                                       model2=NA,use.saved2=FALSE, results2=scen2_results,
                                       log.pc="LG", zone="W",
                                       bpmin=(-0.4),bpmax=(+0.6),
                                       maintitle="Pelagic Trawl/Seine activity reduced by half")
[1] "Using scenario data held in memory from an existing model run"
[1] "Using baseline data held in a csv files from a past model run"
```

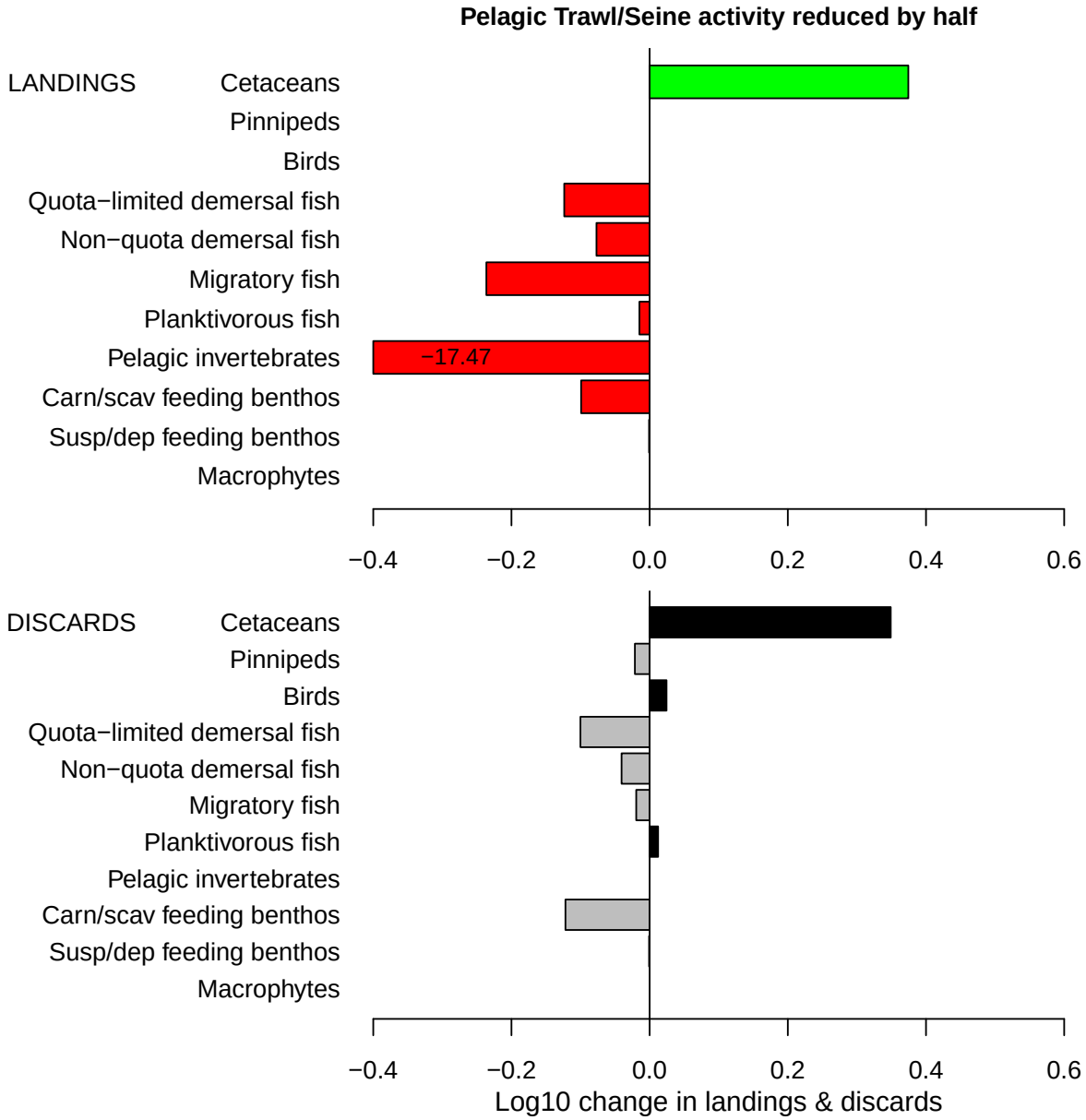


Figure 27. Tornado plot comparison of annual catches and discards in a baseline model (North Sea,

1970-1999) with a scenario model in which pelagic trawl activity is reduced by half. Upper panel (landings): red bars to the left indicate that the landed live weight was lower in the scenario model than the baseline; green bars to the right indicate higher in the scenario model. Lower panel (discards): grey bars to the left indicate that the discarded live weight was lower in the scenario model than the baseline; black bars to the right indicate higher in the scenario model.

8 Generating fisheries yield curves

Fisheries yield curves are a key product that users will want to generate with the model. The yield curve relates stationary state catch of a guild (the catch that can be taken sustainable in the long term under stationary environmental and fishery conditions) to its harvest ratio. We expect the yield curve to be dome-shaped - at zero harvest ratio there is no catch; at very high harvest ratio the stock is depleted so there is little or no catch. At intermediate harvest ratios we expect a sustainable catch. The peak catch is referred to as the Maximum Sustainable Yield (MSY), and this is generated at a harvest ratio which is equivalent to the term F_MSY in the fisheries literature. One of the most important tasks in fish stock assessment is to determine where the current fishing mortality rate of a species sits in relation to F_MSY.

Traditionally, MSY and F_MSY are regarded as constant characteristics for each harvested species, or at least their potential for dependency on the status of other species is neglected. However, this is plainly not the case since changes in predation mortality must affect both MSY and F_MSY. This interdependency of yield properties of guilds is one the key features that StrathE2E was intended to resolve,

The function **e2e_run_ycurve()** runs a set of model scenarios defined by a vector of values of the harvest ratio multipler for either planktivorous or demersal fish. The baseline in each case is defined by a harvest ratio multipler vaklue of 1.0, i.e. the 'present-day' state of the fishery. For each muliplier in the vector, the model is run to a stationary state and the whole-domain annual average mass, annual landings and annual discards of planktivorous and demersal fish are extracted from the final year outputs. In addition the whole-domain annual average masses of all o fthe other living components of the food web are saved. The data are returned as a dataframe and saved to .csv files.

A separate function is provided to plot the yield curve data (**e2e_plot_ycurve()**).

8.1 Generating yield data e2e_run_ycurve()

Run a set of models to generate fishery yield curve data for either planktivorous or demersal fish.

Table 32. Arguments of the functions **e2e_run_ycurve()**

Argument	Description
model	R-list object defining a baseline model and configuration compiled by the e2e_read() function
selection	Text string from a list identifying the fish guild for which a yield curve is to be generated. Select from: "PLANKTIV", "DEMERSAL", Remember to include the phrase within "" quotes
nyears	Number of years to run each model scenario (default = 50)
HRvector	A vector of ascending order unique multiplier values to be applied to the baseline fish harvest ratio of the guild to be analysed. The values do not have to be evenly spaced (default = c(0,0.5,1.0,1.5,2.0,2.5,3.0))
HRfixed	Single value of the multiplier to be applied as the alternative (planktivorous or demersal) fish harvest ratio for all runs (default=1.0)
csv.output	Logical. If TRUE then enable writing of csv output files (default=FALSE)

Perform a set of StrathE2E model runs along a sequence of values of either planktivorous or demersal fish harvest ratio multiplier, saving the annual average whole-domain biomass, annual landings and annual discards of all exploitable food web guilds from each run plus the annual average biomasses of all the other living components of the food web.

The baseline for the sequence of runs (harvest ratio multiplier = 1.0) is a model name and variant as loaded by the e2e_read() function.

The planktivorous or demersal fish yield curve can be generated for a given fixed setting of the other (demersal or planktivorous) fish harvest ratio multiplier (default = 1.0). All other conditions are held constant as in the baseline model configuration.

The yield curve represents the catch that would be generated from the stationary state of the model attained with long-term repeating annual cycles of all driving data. Hence it is important that each simulation is run for long enough that the model attains its stationary state, which may be some distance from the baseline model initial conditions. It is recommended that each run is at least 50 years.

The data on annual average biomass and annual integrated catches stored in the returned dataframe object (and optionally a csv file) can subsequently be plotted using the function e2e_plot_ycurve(). Users can easily plot any of the other saved data using their own plotting code.

If csv output is selected then the resulting files in the current user results folder have names Yield_curve_data_PFHRmult-*.csv or Yield_curve_data_DFHRmult-*.csv, depending on the 'selection' argument, and * represents the model.ident text set in the prior e2e_read() function call.

```
# Load the 1970-1999 version of the North Sea model supplied with the package :
model <- e2e_read("North_Sea", "1970-1999",model.ident="70-99base",silent=TRUE)
#
# In this illustrative example the StrathE2E() model is run for only 5 years to enable
# quick return of results. In a real simulation nyear would be at least 50.
# This example illustrates that the vector of planktivorous fish harvest ratio multipliers
# does not have to be evenly spaced.
hr=c(0,0.5,0.75,1.0,1.25,2.0,3.0)
pf_yield_data <- e2e_run_ycurve(model,selection="PLANKTIV", nyears=5,HRvector=hr,
                                HRfixed=1,csv.output=TRUE)
#
# View the column names of the results dataframe:
names(pf_yield_data)
#
```

8.2 Plotting yield data e2e_plot_ycurve()

Plot fishery yield curve data for planktivorous or demersal fish.

Table 33. Arguments of the functions **e2e_plot_ycurve()**

Argument	Description
model	R-list object defining the baseline model configuration used to generate the data and compiled by the e2e_read() function
selection	Text string from a list identifying the fish guild for which a yield curve is to be generated. Select from: "PLANKTIV", "DEMERSAL". Remember to include the phrase within "" quotes
use.saved	Logical. If TRUE use data from a prior user-defined run held as csv files data in the current results folder (default=FALSE)
use.example	Logical. If TRUE use pre-computed example data from the internal North Sea model rather than user-generated data (default=FALSE)
results	Dataframe generated by the function e2e_run_ycurve(). Only needed if use.saved2=FALSE and use.example2=FALSE. (Default=NULL)
title	Optional free text (enclosed in "") to be added as a header for the plot (default ="")

Plot planktivorous or demersal fish yield curve data generated by the function e2e_run_ycurve().

In the function e2e_run_ycurve(), the baseline for the sequence of runs (harvest ratio multiplier = 1.0) is a model name and variant as loaded by the e2e_read() function. The function then generates a set of biomass, landings and discards data for multiples of the target fish (planktivorous or demersal) harvest ratios relative to this baseline. This is done for a given value of the alternative (demersal or planktivorous) harvest ratio (also a multiple of the the baseline).

This function plots two graphs - the annual average fish biomass in the whole model domain ($mMN.m^{-2}$) as a function of harvest ratio multiplier, and the yield curve, i.e. the annual catch (and discards) ($mMN.m^{-2}.y^{-1}$ from the whole model domain) as functions of the multiplier.

The yield curve represents the catch that would be generated from the stationary state of the model attained with long-term repeating annual cycles of all driving data including fishing.

Arguments for this function permit the input data to be drawn from an existing data object generated by the function e2e_run_ycurve(), a previously generated csv file, or example data provided with the package for versions of the internal North Sea models.

The function returns a dataframe containing the data which have been plotted plus the annual average mass density and where appropritae the landings and discards of all the other living food web components in the model.

```
# Load the 1970-1999 version of the North Sea model supplied with the package
# and generate a yield data object:
# WARNING - this example will take about several minutes to run
model <- e2e_read("North_Sea", "1970-1999",silent=TRUE)
pfhr=c(0,0.5,0.75,1.0,1.25,1.5,2.0,2.5,3.0)
pf_yield_data <- e2e_run_ycurve(model,selection="PLANKTIV",nyears=50,HRvector=pfhr,
                                HRfixed=1,csv.output=FALSE)
# Then plot the results as follows:
data <- e2e_plot_ycurve(model,selection="PLANKTIV", results=pf_yield_data,
                        title="Planktivorous yield with baseline demersal fishing")
```

```
# Using examaple data generated with selection="PLANKTIV" ...
# Plot example data for one of the North Sea model versions internal to the package
model <- e2e_read("North_Sea", "1970-1999",silent=TRUE)
pf_yield_data<-e2e_plot_ycurve(model, selection="PLANKTIV", use.example=TRUE)
Reading example results from StrathE2E2examples data package for the North_Sea 1970-1999 model
```

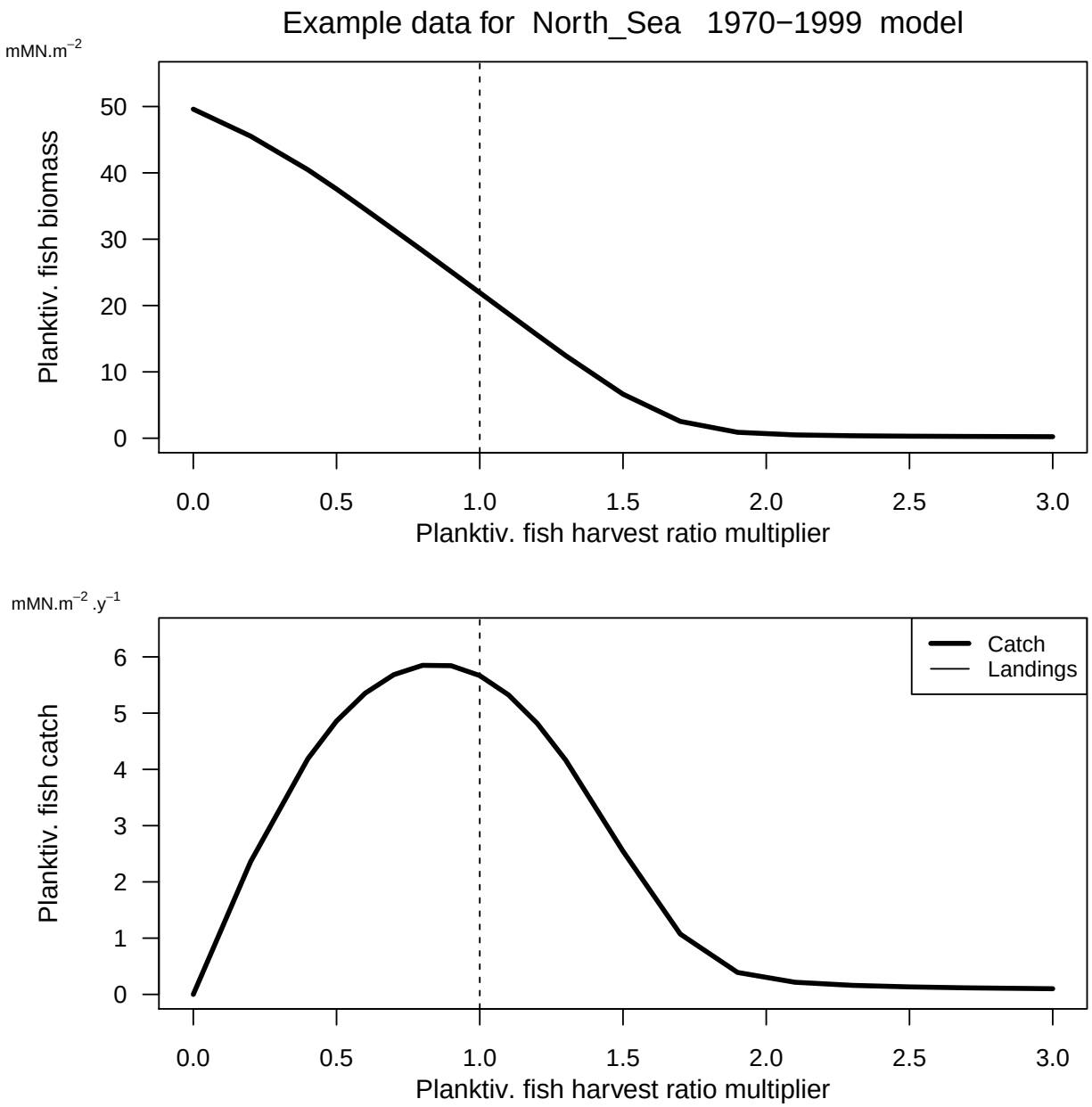
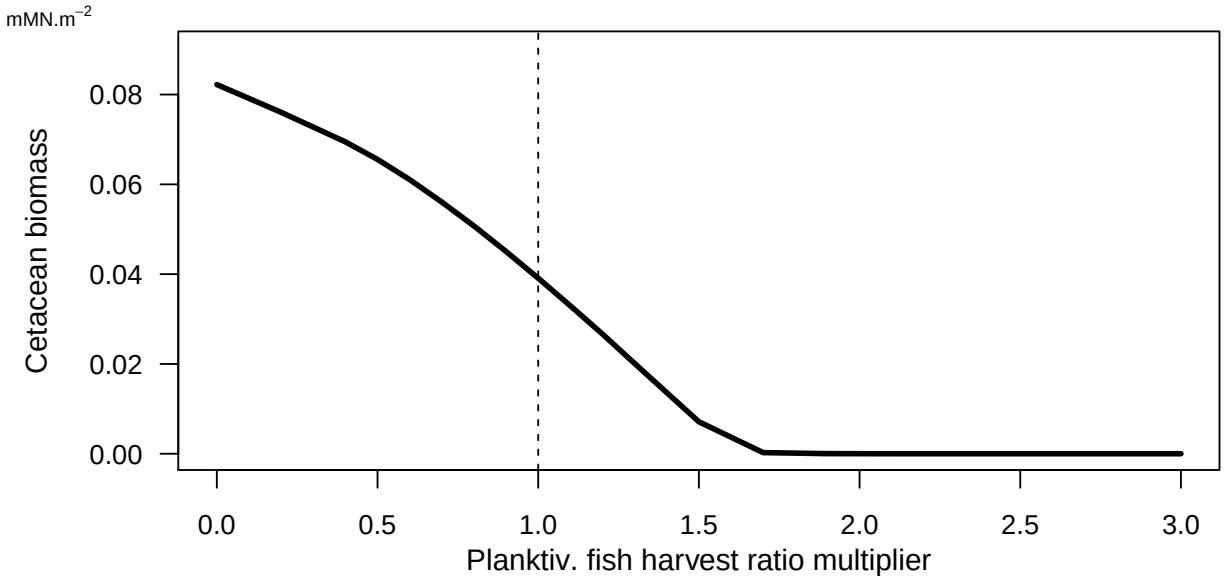


Figure 28. Yield data plot for planktivoirous fish in the 1970-1999 North Sea model with baseline demersal harvest ratios. Upper panel, annual average biomass of planktivorous fish (mMN.m^{-2} in the whole modle domain) as a function of harvest ratio. Lower panel: catch divided into landings and discards of planktivorous fish (mMN.m^{-2}.y^{-1}) as a function of harvest ratio.

```
# Users can then plot other biomass, landings and discards data in the results
# object by, for example:
par(mfrow=c(2,1))
par(mar=c(3.2,5,2,0.8))
ym<-1.1*max(pf_yield_data$Cetaceanbiom)
plot(pf_yield_data$PlankFishHRmult,pf_yield_data$Cetaceanbiom,ylim=c(0,ym),type="l",
     lwd=3,yaxt="n",xaxt="n",ann=FALSE)
   abline(v=1,lty="dashed")
   axis(side=1,las=1,cex.axis=0.9)
   axis(side=2,las=1,cex.axis=0.9)
```

```
   mtext("Planktiv. fish harvest ratio multiplier",cex=1,side=1,line=2)
   mtext("Cetacean biomass",cex=1,side=2,line=3.5)
   mtext(bquote("mMN.m"^-2),cex=0.7,side=3,line=-0.05,adj=-0.18)
ym<-1.1*max(pf_yield_data$Cetaceandisc)
plot(pf_yield_data$PlankFishHRmult,pf_yield_data$Cetaceandisc,ylim=c(0,ym),type="l",
     lwd=3,yaxt="n",xaxt="n",ann=FALSE)
   abline(v=1,lty="dashed")
   axis(side=1,las=1,cex.axis=0.9)
   axis(side=2,las=1,cex.axis=0.9)
   mtext("Planktiv. fish harvest ratio multiplier",cex=1,side=1,line=2)
   mtext("Cetacean by-catch",cex=1,side=2,line=3.5)
   mtext(bquote("mMN.m"^-2 ~ ".y"^-1),cex=0.7,side=3,line=-0.05,adj=-0.18)
```



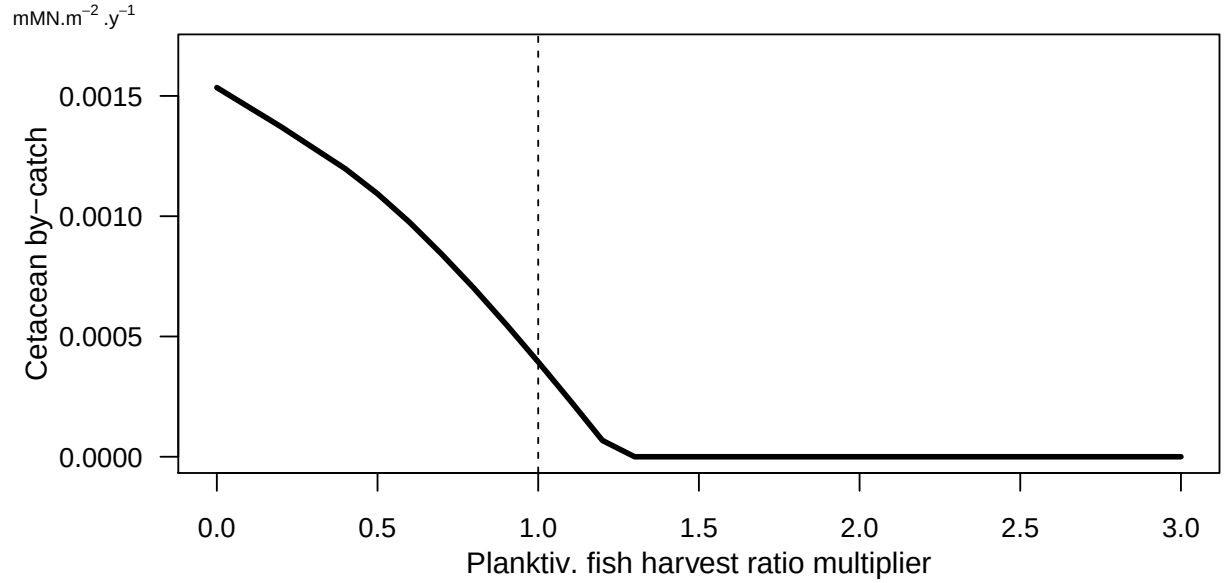


Figure 29. Example of a possible user-generated yield data plot for cetaceans in the 1970-1999 North Sea model as a function of planktivorous harvest ratio, with baseline demersal harvest ratios. Upper panel, annual average biomass of cetaceans (mMN.m^{-2} in the whole modle domain) as a function of planktivorous fish harvest ratio. Lower panel: catch divided into targeted landings and by-catch (discards) of cetaceans (mMN.m^{-2}.y^{-1}) as a function of harvest ratio.

```
# Using example data generated with selection="DEMERSAL"...
model <- e2e_read("North_Sea", "1970-1999",silent=TRUE)
df_yield_data<-e2e_plot_ycurve(model, selection="DEMERSAL", use.example=TRUE)
Reading example results from StrathE2E2examples data package for the North_Sea 1970-1999 model
```

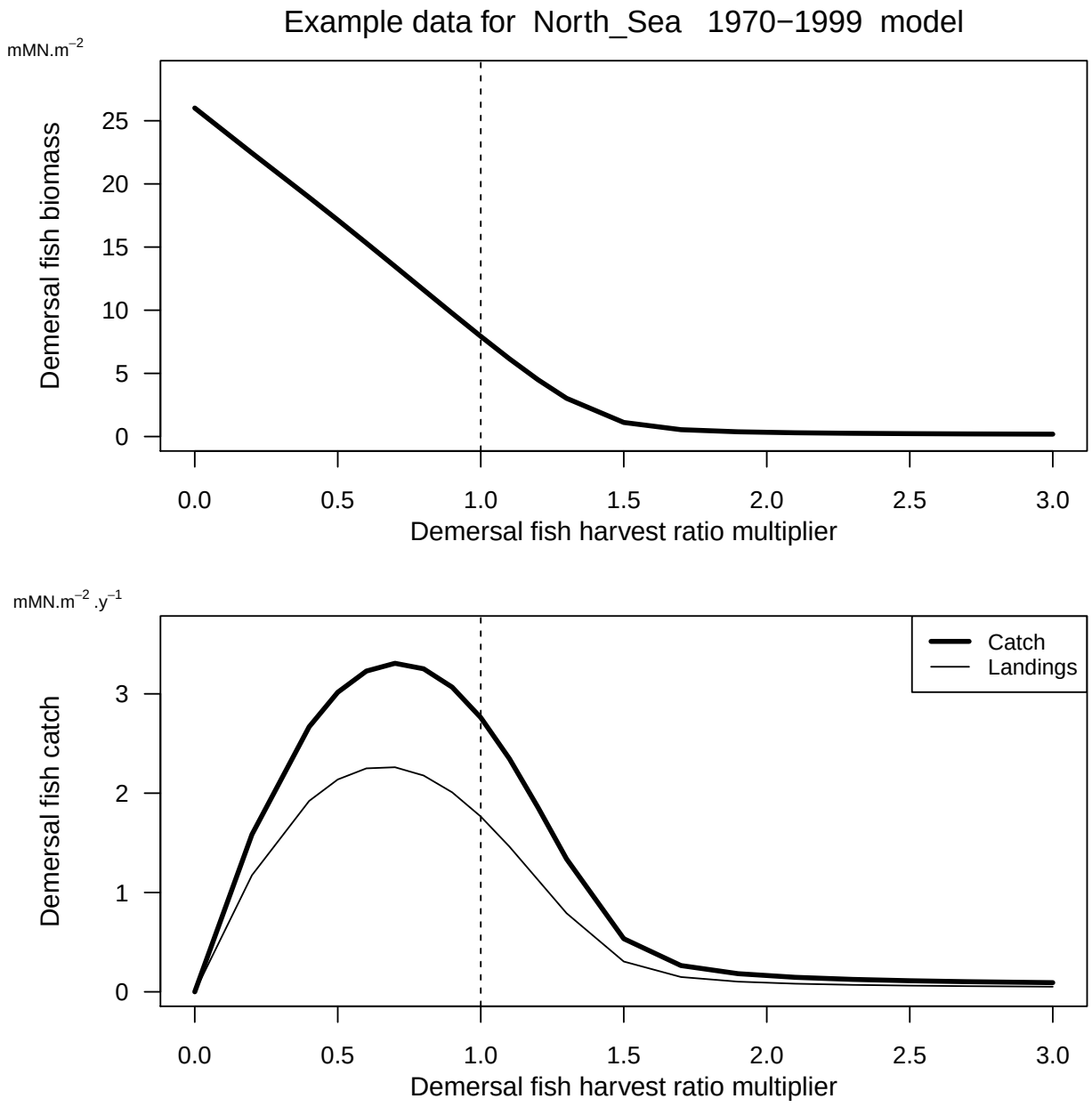


Figure 30. Yield data plot for demersal fish in the 1970-1999 North Sea model with baseline planktivorous harvest ratios. Upper panel, annual average biomass of demersal fish (mMN.m^{-2} in the whole modle domain) as a function of harvest ratio. Lower panel: catch divided into landings and discards of demersal fish

($mMN.m^{-2}.y^{-1}$) as a function of harvest ratio.

```
# Users can then plot other biomass, landings and discards data in the results object
# by, for example:
par(mfrow=c(2,1))
par(mar=c(3.2,5,2,0.8))
ym<-1.1*max(df_yield_data$Pinnipedbiom)
plot(df_yield_data$DemFishHRmult,df_yield_data$Pinnipedbiom,ylim=c(0,ym),type="l",
     lwd=3,yaxt="n",xaxt="n",ann=FALSE)
    abline(v=1,lty="dashed")
    axis(side=1,las=1,cex.axis=0.9)
    axis(side=2,las=1,cex.axis=0.9)
    mtext("Demersal fish harvest ratio multiplier",cex=1,side=1,line=2)
    mtext("Pinniped biomass",cex=1,side=2,line=3.7)
    mtext(bquote("mMN.m"^-2),cex=0.7,side=3,line=-0.05,adj=-0.18)
ym<-1.1*max(df_yield_data$Pinnipeddisc)
plot(df_yield_data$DemFishHRmult,df_yield_data$Pinnipeddisc,ylim=c(0,ym),type="l",
     lwd=3,yaxt="n",xaxt="n",ann=FALSE)
    abline(v=1,lty="dashed")
    axis(side=1,las=1,cex.axis=0.9)
    axis(side=2,las=1,cex.axis=0.9)
    mtext("Demersal fish harvest ratio multiplier",cex=1,side=1,line=2)
    mtext("Pinniped by-catch",cex=1,side=2,line=3.7)
    mtext(bquote("mMN.m"^-2 ~ ".y"^-1),cex=0.7,side=3,line=-0.05,adj=-0.18)
```

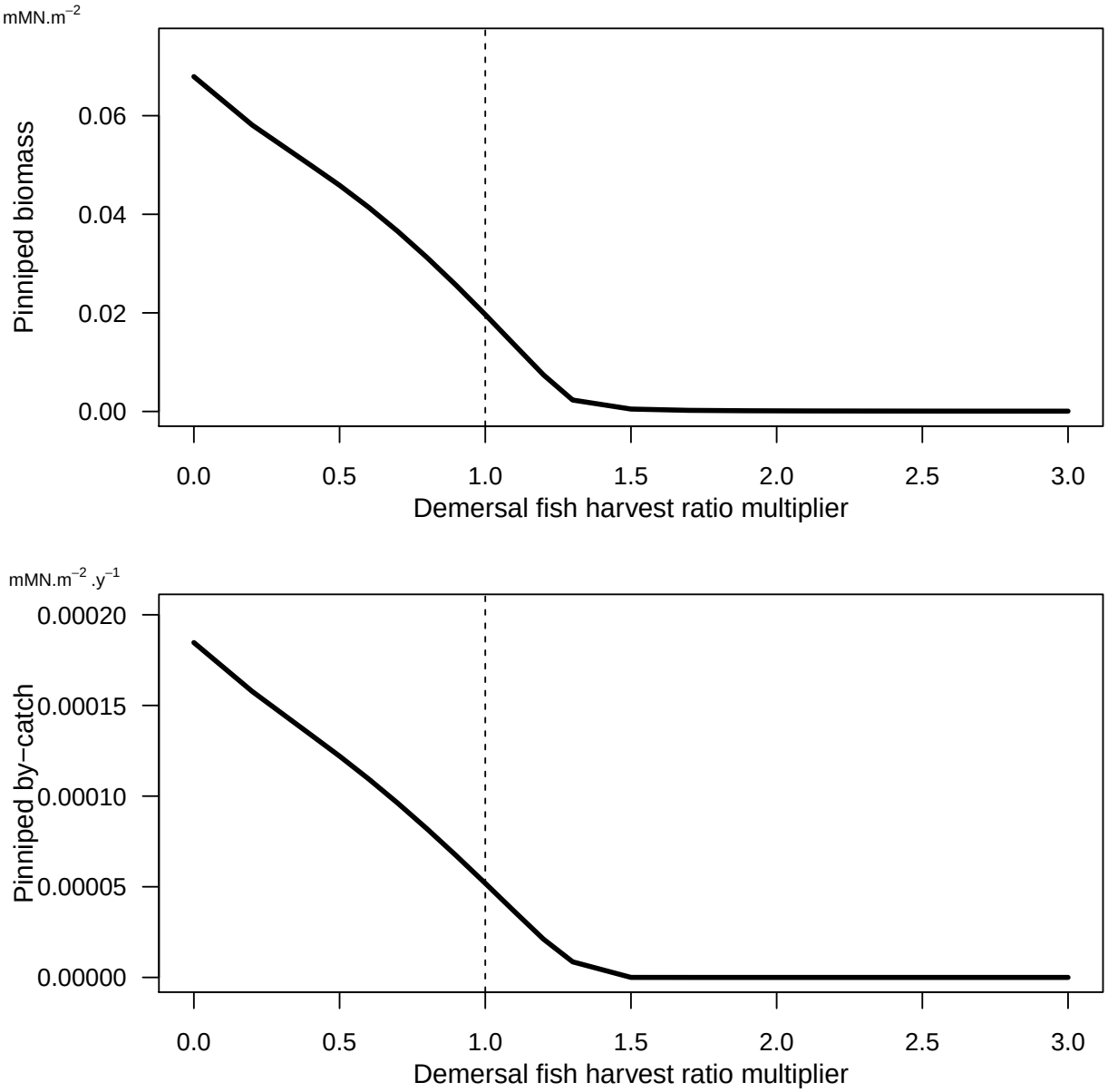


Figure 31. Example of a possible user-generated yield data plot for pinnipeds in the 1970-1999 North Sea model as a function of demersal harvest ratio, with baseline planktivorous harvest ratios. Upper panel, annual average biomass of pinnipeds (mMN.m^{-2} in the whole modle domain) as a function of demersal fish harvest ratio. Lower panel: by-catch (discards) of pinnipeds (mMN.m^{-2}.y^{-1}) as a function of harvest ratio. In this case there are no targeted landings.

9 Plotting functions for visualizing model inputs

There are two main functions for plotting the input data to the model - **e2e_plot_edrivers()** and e2e_plot_fdrivers()** - these visualize the environmental inputs and fishing-related inputs respectively.

9.1 Visualizing environmental drivers e2e_plot_edrivers()

Plot climatological year of environmental driving data.

Table 34. Arguments of the functions **e2e_plot_edrivers()**

Argument	Description
model	R-list object defining a model configuration and compiled by the e2e_read() function

Create a multi-panel page of time series plots of climatological annual cycles of driving data as provided in the input csv files.

The variables to be plotted and their units are shown in Table 35.

Table 35. Environmental input data dispalayed in each panel of the plot produced by **e2e_plot_edrivers()**

Data	Units	Comments
Sea surface irradiance	μE.m^{-2}.d^{-1}	
Suspended particulate matter	g.m^{-3}	
Temperature	deg-C	
Vertical diffusivity gradient	m.d^{-1}	derived from the vertical diffusivity (m^2.s^{-1}) and mixing length scale (m)
External inflows	m^3 per m^2 sea surface of model domain	derived from proportion input per layer volume, layer thicknesses and areas
River discharge	m^3 per m^2 sea surface of model domain	derived from proportion input to inshore volume, and inshore layer thickness and area
Inshore significant wave height	m	
Proportion of seabed disturbed	d^{-1}	aggregated over the three sediment classes in each zone
External boundary nitrate concentration	mMN.m^{-3}	
External boundary ammonia concentration	mMN.m^{-3}	
External boundary phytoplankton concentration	mMN.m^{-3}	
External boundary detritus concentration	mMN.m^{-3}	
River nitrate concentration	mMN.m^{-3}	
River ammonia concentration	mMN.m^{-3}	
Atmospheric nitrate deposition flux	mMN.m^{-2}.d^{-1}	area averaged flux density over each zone
Atmospheric ammonia deposition flux	mMN.m^{-2}.d^{-1}	area averaged flux density over each zone

```
model<-e2e_read(model.name="North_Sea", model.variant="2003-2013")
Current working directory is...
'C:/Users/Public/Documents/StrathE2E2'
No 'results.path' specified so any csv data requested
will be directed to/from the temporary directory...
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ'

Model setup and parameters gathered from ...
StrathE2E2 package folder
Model results will be directed to/from ...
```

```
'C:\Users\ais04103\AppData\Local\Temp\RtmpiySPHQ/North_Sea/2003-2013/'
e2e_plot_edrivers(model)
```

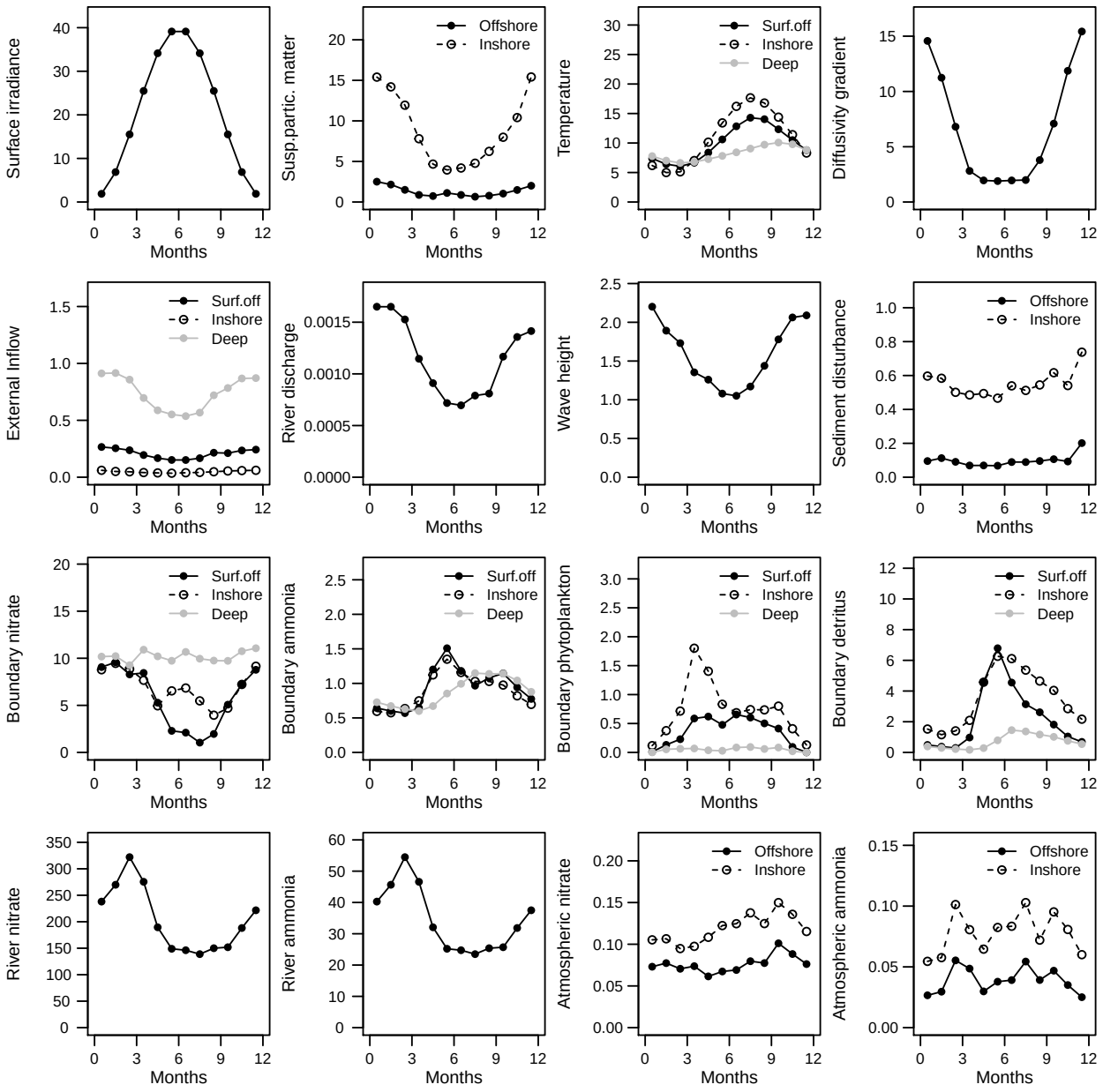


Figure 32. Plot of climatological annual cycles of the environmental drivers of the 2003-2013 North Sea model

9.2 Visualizing fishing-related drivers e2e_plot_fdrivers()

Plot shaded-grid-cell diagrams of the distributions of fishing-related drivers across gears, guilds and seabed habitats.

Table 36. Arguments of the functions **e2e_plot_fdrivers()**

Argument	Description
model	R-list object defining a model configuration and compiled by the e2e_read() function
selection	Text string from a list identifying the class of fishing-related driving data to be plotted. Select from: "ACTIVITY", "ABRASION", "HARVESTR", "DISCARDS", "OFFAL". Remember to include the phrase within "" quotes

Select from a list of fishing-related drivers of the model (gear activity rates, seabed abrasion rates, harvest ratios, discard rates and offal generation rates), and generate a shaded-grid-cell diagram showing the distribution across gears, guilds and seabed habitats.

In each case, regardless of the vaviable selected, the function returns both a graphical image and a list object containing the matix of data plotted and vectors of the axis labels, which uses can use to generate their own plots if desired.

Details relating to fishing activity distributions (selection = "ACTIVITY"):

Plot a matrix of seabed habitats vs fishing gears with each cell shaded to indicate log_e transformed activity density (white = 0, purple = high).

The spatial distribution of fishing gear activity in the model is defined by two input data sets: * Vector of whole domain activity density of each fishing gear ($s.d^{-1}$ per m^2 of whole model domain) * Matrix of the proportional distribution of whole domain activity density of each gear (rows) across seabed habitats (columns)

The activity density of each gear in each habitat is then obtained by multiplying the whole domain activity density into the proportional distribution matrix, and dividing by the area-proportions of habitats in the domain. The units of habitat-specific activity density are then ($s.d^{-1}.m^{-2}$).

The vector of area-proportions of each habitat in the model domain is part of the moldel configuration parameter set.

The calculation take account of any activity multipliers set in the csv inputs to be applied to the activity density.

When interpreting patterns of activity density, bear in mind that the impact (harvest ratio, seabed abrasion) per unit activity can vary considerably between gears.

```
# Load the 1970-1999 version of the North Sea model supplied with the package,
# run, and generate a plot:
model <- e2e_read("North_Sea", "2003-2013",silent=TRUE)
plotted_data<-e2e_plot_fdrivers(model, selection="ACTIVITY")
```

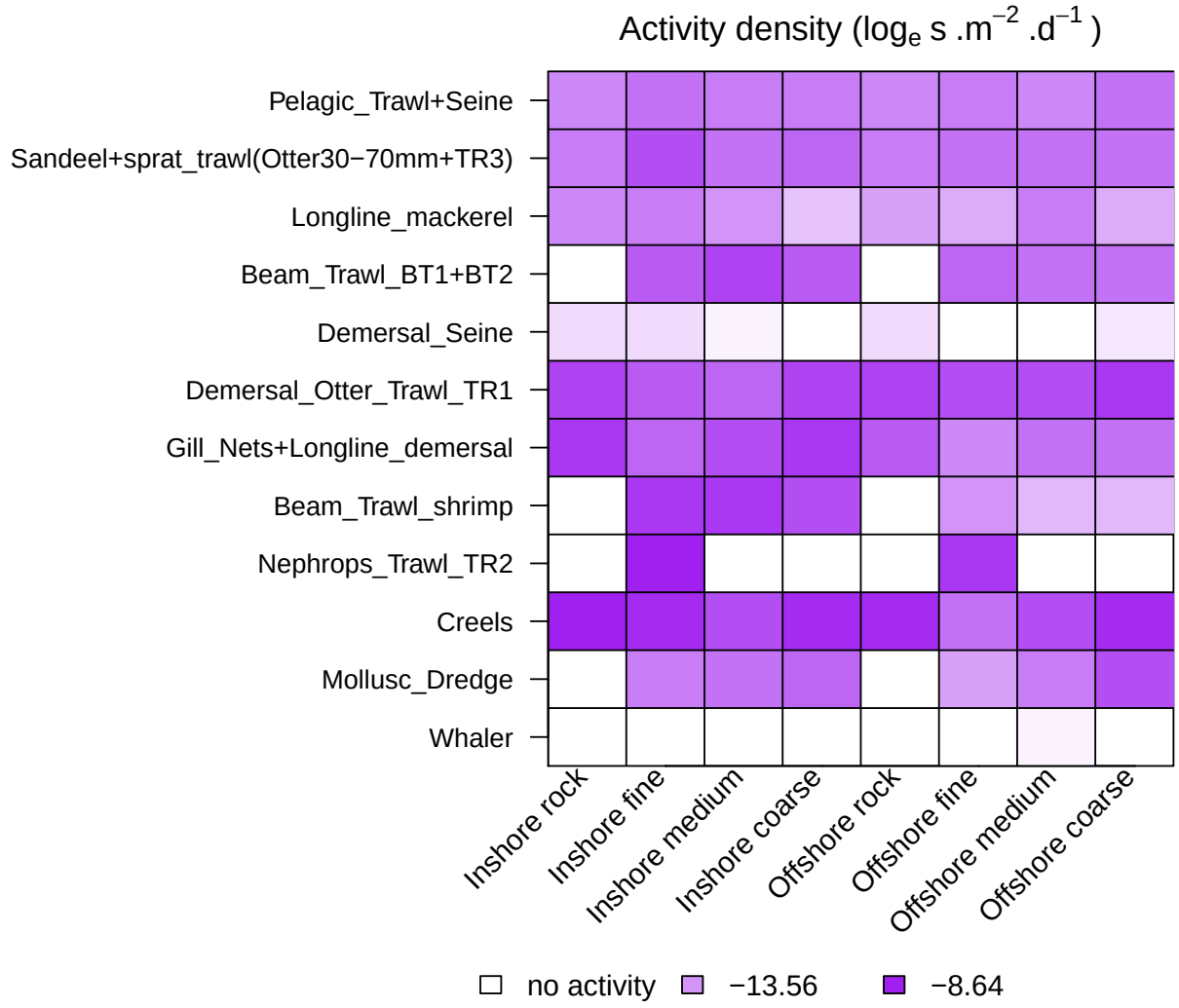


Figure 33. Distribution of gear activity rates across habitats in the 2003-2013 North Sea model.

Details relating to seabed abrasion rate distributions (selection = "ABRASION"):

The function plots a matrix of seabed habitats vs fishing gears with each cell shaded to indicate abrasion area ratio (log_e transformed proportion of habitat area abraded per day; white = 0, purple = high).

The spatial distribution of seabed abrasion in the model is defined by three input data sets: * Vector of whole domain activity density of each fishing gear ($s.d^{-1}$ per m^2 of whole model domain) * Matrix of the proportional distribution of whole domain activity density of each gear (rows) across seabed habitats (columns) * Vector of seabed abrasion rate of each fishing gear ($m^2.s^{-1}$)

The activity density of each gear in each habitat is then obtained by multiplying the whole domain activity density into the proportional distribution matrix, and dividing by the area-proportions of habitats in the domain. The units of habitat-specific activity density are then ($s.d^{-1}.m^{-2}$).

The product of activity density per gear and habitat and gear abrasion rate is then the abrasion area ratio, or abrasion rate ($m^2.m^{-2}.d^{-1}$)

The vector of area-proportions of each habitat in the model domain is part of the moldel configuration parameter set.

The calculation take account of any activity multipliers set in the csv inputs to be applied to the activity

density.

```
# Load the 1970-1999 version of the North Sea model supplied with the package,
# run, and generate a plot:
model <- e2e_read("North_Sea", "2003-2013",silent=TRUE)
plotted_data<-e2e_plot_fdrivers(model, selection="ABRASION")
```

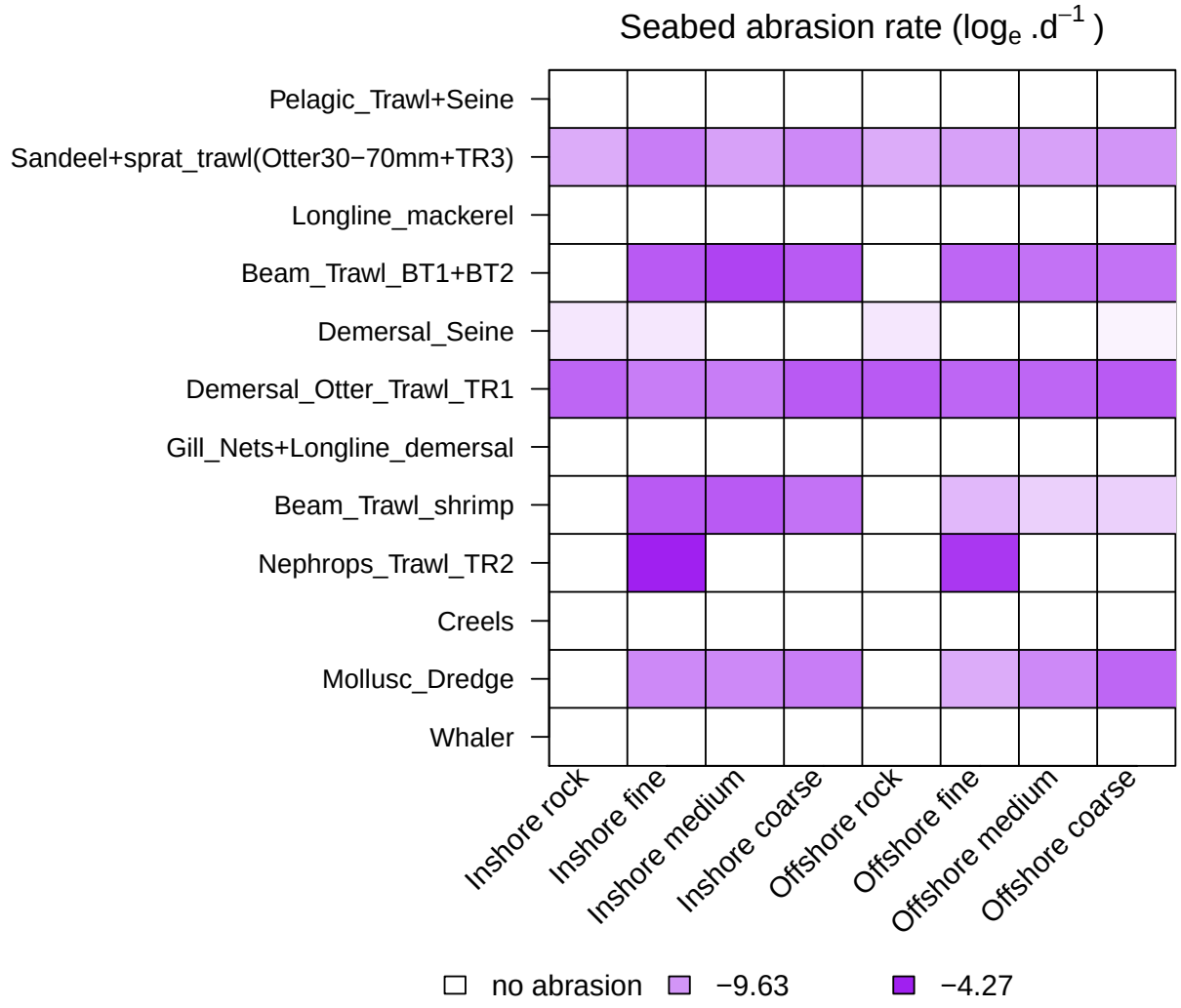


Figure 34. Distribution of abrasion area ratio across habitats in the 2003-2013 North Sea model.

Details relating to harvest ratio distributions (selection = "HARVSTR"):

The function plots a matrix of the values of harvest ratio (proportion of exploitable biomass of each guild capatured per day) for each guild in the ecology model (columns) arising from the activity of each gear (rows). Hence each row represents the selectivity pattern of each gear with respect to guilds. Cells in the matrix are shaded to indicate log_e transformed harvest ratio (colour gradient: white = 0, purple = high).

Harvest ratio is calculated by the fishing fleet model and piped into the ecology model. The fleet model takes inputs of activity density by each gear, and a matrix of fishing power, to calculate effort (effort = activity * power). Harvest ratio is assumed to be lineally related to effort by a scaling coefficient which is input as a parameter to the fleet model.

Note that the visualization of harvest ratio generated by this function is based on the csv input data to

the fleet model (including any multipliers to be applied to either fishing activity or harvest ratio scaling parameters). However, it does not reflect any effects on harvest ratio arising from discarding scenarios since these are dynamic and generated at run-time within the ecology model.

```
# Load the 1970-1999 version of the North Sea model supplied with the package,
# run, and generate a plot:
model <- e2e_read("North_Sea", "2003-2013",silent=TRUE)
plotted_data<-e2e_plot_fdrivers(model, selection="HARVESTR")
```

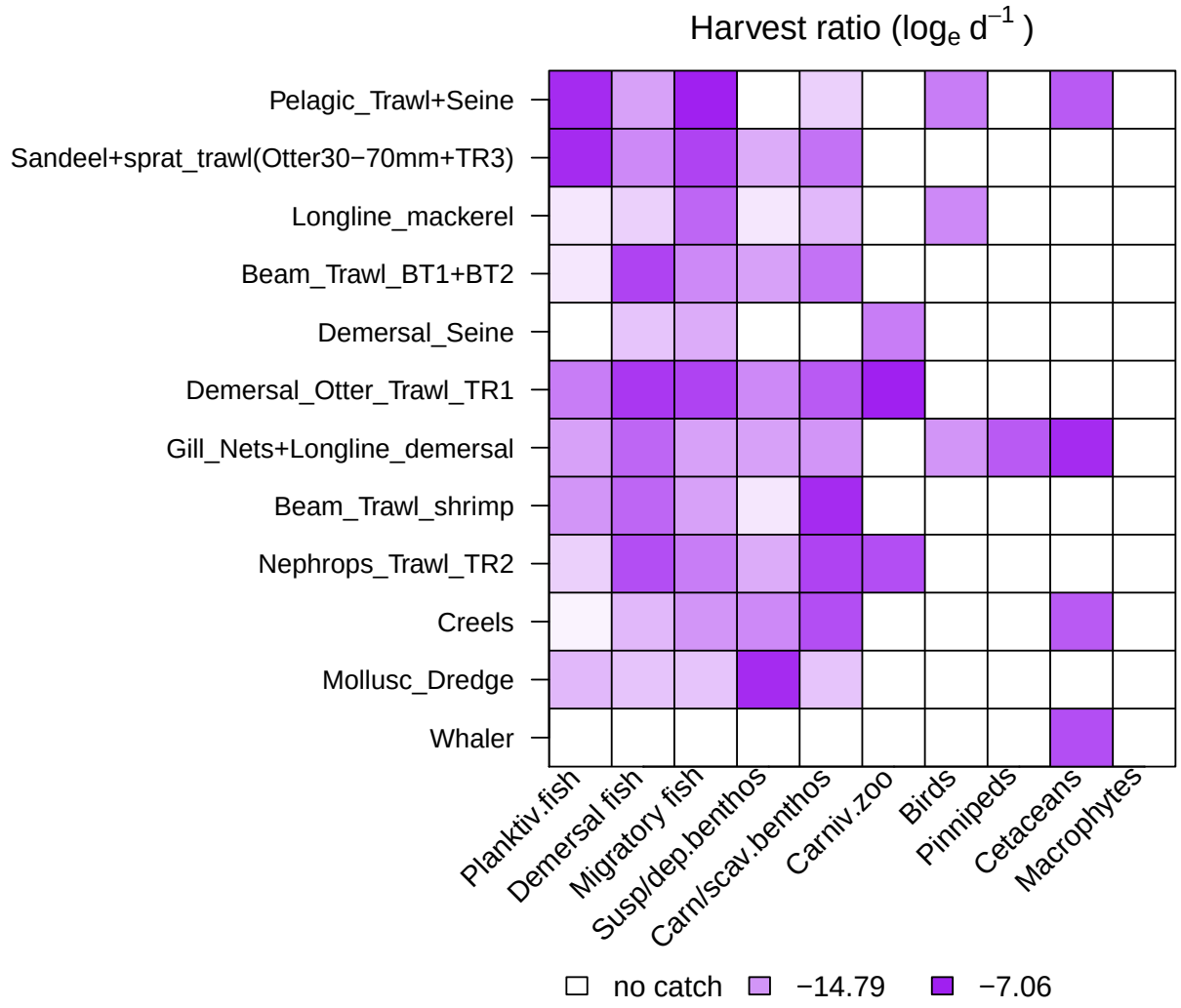


Figure 35. Distribution of harvest ratios by gears across guilds in the 2003-2013 North Sea model.

Details relating to discard rate distributions (selection = "DISCARDS"):

The function plots a matrix of the values of discard rate (proportion by weight of catch discarded at sea prior to any at-sea processing (evisceration)) for each guild in the ecology model (columns) arising from the activity of each gear (rows). Cells in the matrix are shaded to indicate discard rate (range 0 - max; colour gradient: white = 0, purple = high) on a linear scale.

It is possible (and allowable) for the discard rate of a gear-guild combination to be set as a positive number in the 'fishing_discard_parameters' csv input file, but nevertheless the the catching power settings in the corresponding 'fishing_power_parameters' file to be set to zero - in other words the given gear does not

actually catch the given guild. In such cases this function resets the discard rate to NA for plotting purposes.

Note that the visualization of discard rate generated by this function is based on the csv input data to the fleet model. Hence, it does not reflect any effects on discard rate arising from discarding scenarios configured in the fleet model input since these are dynamic and generated during run-time within the ecology model.

```
# Load the 1970-1999 version of the North Sea model supplied with the package,
# run, and generate a plot:
model <- e2e_read("North_Sea", "2003-2013",silent=TRUE)
plotted_data<-e2e_plot_fdrivers(model, selection="DISCARDS")
```

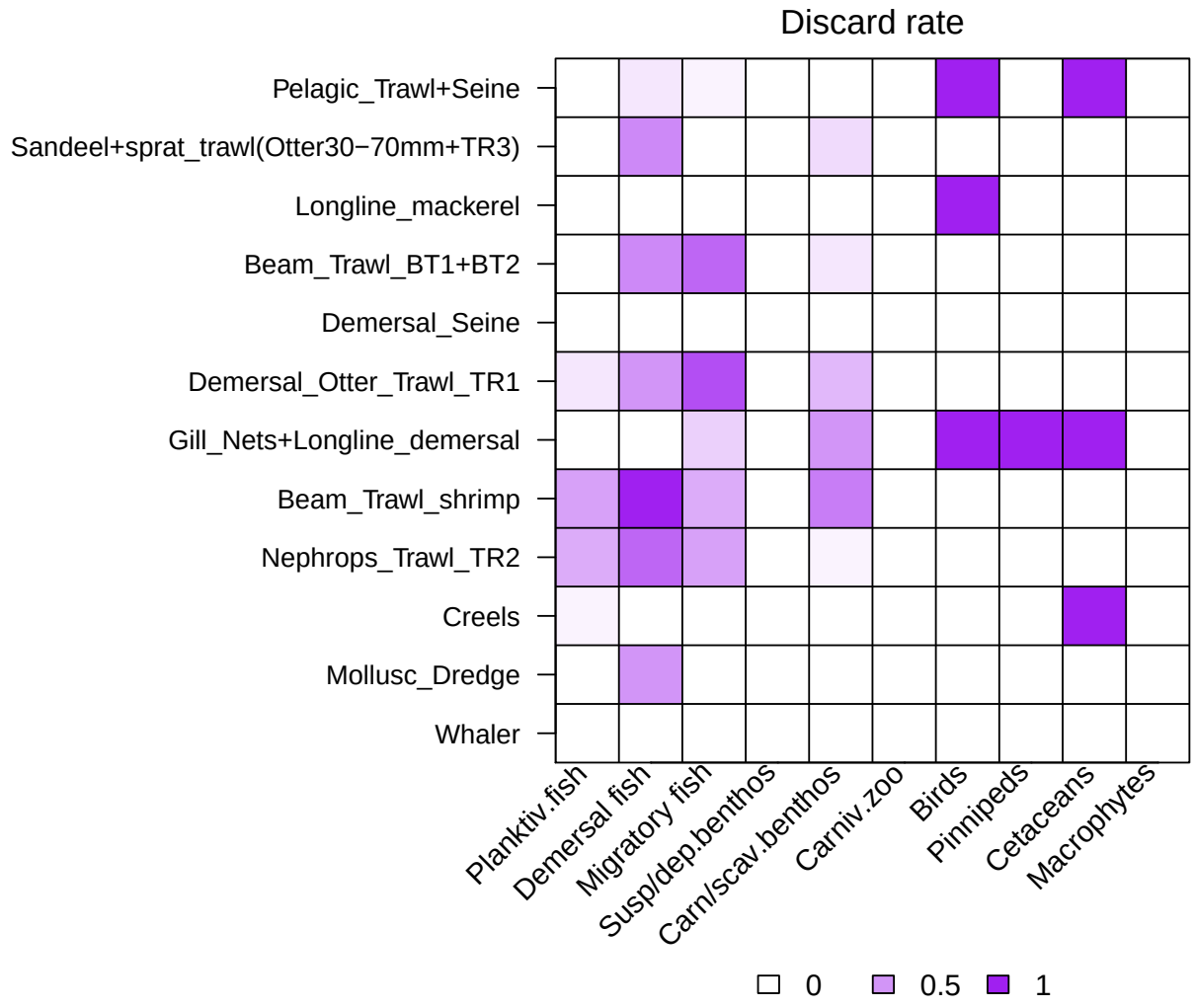


Figure 36. Distribution of discard rates by gears across guilds in the 2003-2013 North Sea model.

Details relating to offal generation rate distributions (selection = "OFFAL"):

The function plots a matrix of the values of offal generation rate (i.e. the proportion by weight of total catch which is returned to the sea as viscera after processing) for each guild (columns) by each gear (rows). Cells in the matrix are shaded to indicate offal generation rate (range 0 - max; colour gradient: white = 0, purple = high) on a linear scale.

The offal generation rate is a quantity derived from three input variables to the fishing fleet model: * Matrix of the discard rate (proportion of catch weight discarded without being processed) for each

guild and gear; (D); (input file '/Param/fishing_discard_parameters.*csv')* Matrix of the processing-at-sea rate (proportion of retained catch which is processed) for each guild and gear; (P); (input file'/Param/fishing_processing_at_sea_parameters.*csv')* Proportion by weight of viscera in the retained catch (assumed constant across all guilds); (V); (value in input file '/Param/fishing_fleet_parameters*.csv')

The offal generation rate is then given by (1 - D).P.V

It is possible (and allowable) for the discard rate and processing-at-sea rate of a gear-guild combination to be set as positive numbers in the relevant input files, but nevertheless the the catching power settings in the corresponding 'fishing_power_parameters' file to be set to zero - in other words the given gear does not actually catch the given guild. In such cases this function resets the discard rate and processing-at-sea rate to NA for plotting purposes.

Note that the visualization of offal rate generated by this function is based on the csv input data to the fleet model. Hence, it does not reflect any effects on discard rate arising from discarding scenarios configured in the fleet model input since these are dynamic and generated during run-time within the ecology model.

```
# Load the 1970-1999 version of the North Sea model supplied with the package,
# run, and generate a plot:
model <- e2e_read("North_Sea", "2003-2013",silent=TRUE)
plotted_data<-e2e_plot_fdrivers(model, selection="OFFAL")
```

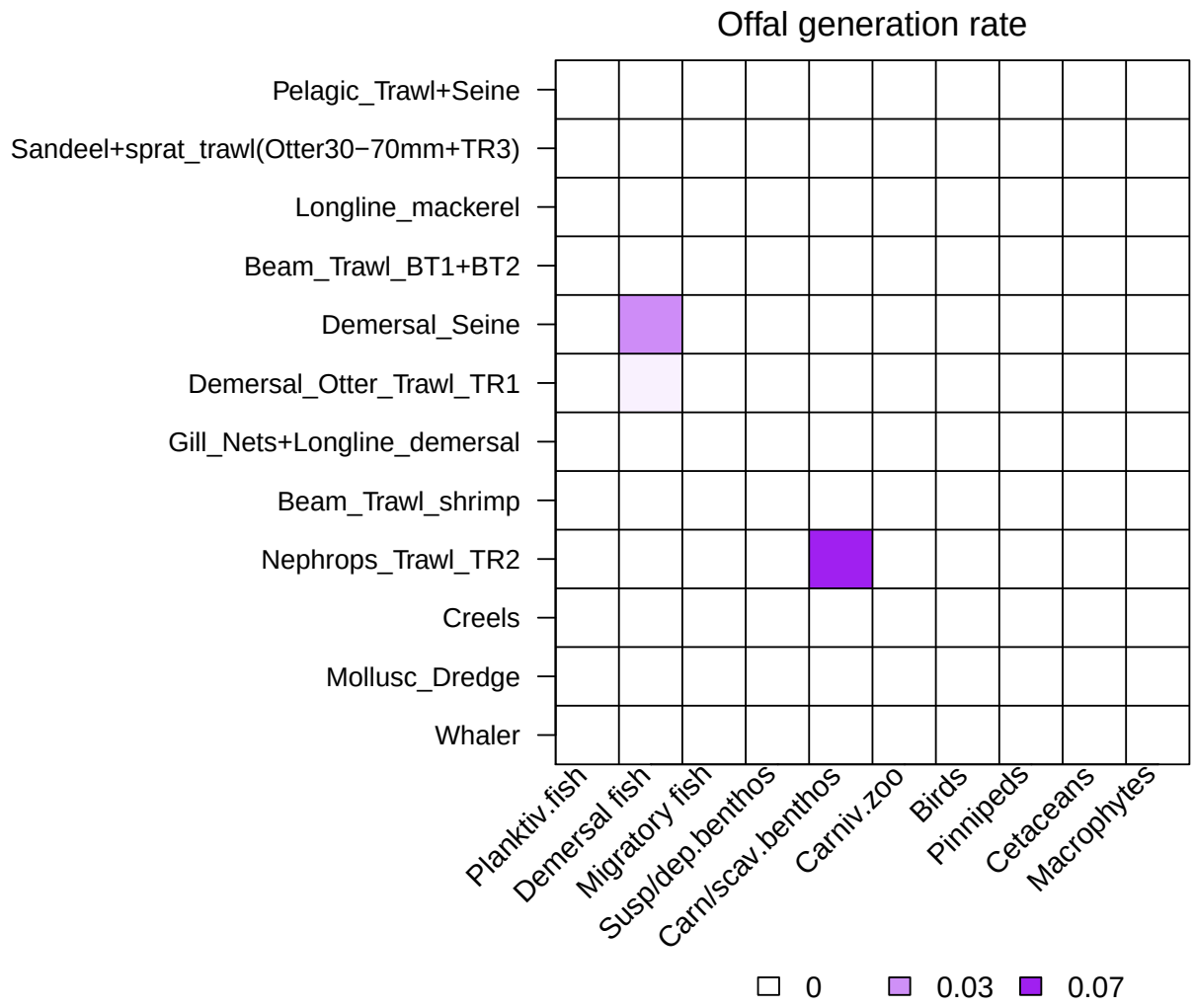


Figure 37. Distribution of offal generation rates by gears across guilds in the 2003-2013 North Sea model.

10 Plotting functions for visualizing model outputs from the final year of a run

The model generates a wide range of outputs from each run. Full details are given in the Technical Manual, and guidance on how to unpack the list object returned at the end of a run is given in Section 4. There is considerable scope for creativity in generating graphics to visualize the data.

The package contains 5 main functions for visualizing model output. They do not provide comprehensive coverage of all the available outputs, more an entry point for users.

Variables to be plotted are selected by argument settings in the functions. In addition, the functions are configured to plot either data from single-runs of the model, or outputs from Monte Carlo simulations showing credible intervals of model outputs.

All of the functions in this section (**Table 37**) show outputs from the final year of any model runs, not transients. The philosophy of the model is focussed on stationary state solutions, and less on transient behaviours. This is because transients are dependent on initial conditions which are unknown in most cases, whereas stationary states are independent of initial conditions.

Table 37. Functions available for visualizing model outputs from the final year of a run

Function	Description
e2e_plot_eco()	Plot annual cycles of ecology model variables
e2e_plot_migration()	Plot annual cycles of active migration fluxes
e2e_plot_catch()	Plot distributions of catches by gear and guild
e2e_plot_trophic()	Plot mean trophic level and omnivory indices
e2e_plot_biomass()	Plot zonal distributions of annual average biomass densities

10.1 Plotting annual cycles of ecology model variables e2e_plot_eco()

Plot daily data on ecological outputs from the model over the final year of a run, optionally with credible intervals.

Table 38. Arguments of the functions **e2e_plot_eco()**

Argument	Description
model	R-list object defining the model configuration compiled by the e2e_read() function
selection	Text string from a list identifying the group of model output variables to be plotted. Select from: “NUT_PHYT”, “SEDIMENT”, “ZOOPLANKTON”, “FISH”, “BENTHOS”, “PREDATORS”, “CORP_DISC”, “MACROPHYTE”, Remember to include the phrase within "" quotes
ci.data	Logical. If TRUE plot credible intervals around baseline model results based on Monte Carlo simulation with the e2e_run_mc() function (default=FALSE)
use.saved	Logical. If TRUE use baseline data from a prior user-defined run held as csv files data in the current results folder (default=FALSE)
use.example	Logical. If TRUE use pre-computed example data from the internal North Sea model as the baseline rather than user-generated data (default=FALSE)
results	R-list object of model output generated by the e2e_run() function. Only needed if ci.data=FALSE, use.saved=FALSE and use.example=FALSE. (Default=NULL)

Generate a multi-panel set of one-year time series plots of selected outputs on the concentrations of ecology

model state variables in each spatial zone (**Table 39**). The default is to plot data from a single model run but, if available, credible intervals of model output from a Monte Carlo analysis can be plotted instead.

Table 39. Variables plotted (and units) given values of the argument "selection"

selection	Variables plotted	Units
NUT_PHYT	Water column nitrate, ammonia, detritus and phytoplankton	$mMN.m^{-3}$
SEDIMENT	Sediment porewater nitrate and ammonia, detritus and corpses	$mMN.m^{-3}$ or $gN.gDW^{-1}$
ZOOPLANKTON	Omnivorous and carnivorous zooplankton	$mMN.m^{-2}$
FISH	Planktivorous and demersal fish and fish larvae	$mMN.m^{-2}$
BENTHOS	Susp/dep and carn/scav benthos and benthos larvae	$mMN.m^{-2}$
PREDATORS	Birds, pinnipeds, cetaceans and migratory fish	$mMN.m^{-2}$
CORP_DISC	Corpses and discards	$mMN.m^{-2}$
MACROPHYTE	Macrphytes and macrophyte debris	$mMN.m^{-2}$

Arguments determine the source of model data to be plotted. These can be outputs from a single model run with data held in memory as a list object or in a saved csv file, or from a Monte Carlo simulation (using the function **e2e_run_mc()**) to estimate credible intervals of model outputs. Generation of credible interval data is a long computing task, so example data for the North Sea model provided with the package are available as an illustration.

If plotting of credible intervals is selected, results from the maximum likelihood model are shown by a red line. The median of the credible values distribution is shown my a solid black line. A grey-shaded area indicates the 50% credible interval (spanning quartiles of the cumulative likelihood of simulated values). Black dashed lines span the 99% credible interval.

To direct the graph output to a file rather than the screen, wrap the e2e_plot_eco() function call in a graphical device call: Since the plot pages contain different numbers of panels the recommended width:height ratios are as follows:

Table 40. Suggested width:height ratios for graphical output to a file depending on the argument "selection"

selection	width	height
NUT_PHYT	1.5	1
SEDIMENT	0.67	1
ZOOPLANKTON	1	1
FISH	2	1
BENTHOS	2	1
PREDATORS	2	1
CORP_DISC	1	1
MACROPHYTE	2	1

```
# Load the 1970-1999 version of the North Sea model supplied with the package,
# run, and generate a plot:
model <- e2e_read("North_Sea", "1970-1999",silent=TRUE)
results <- e2e_run(model, nyears=3)
e2e_plot_eco(model, selection="NUT_PHYT",results=results)
```

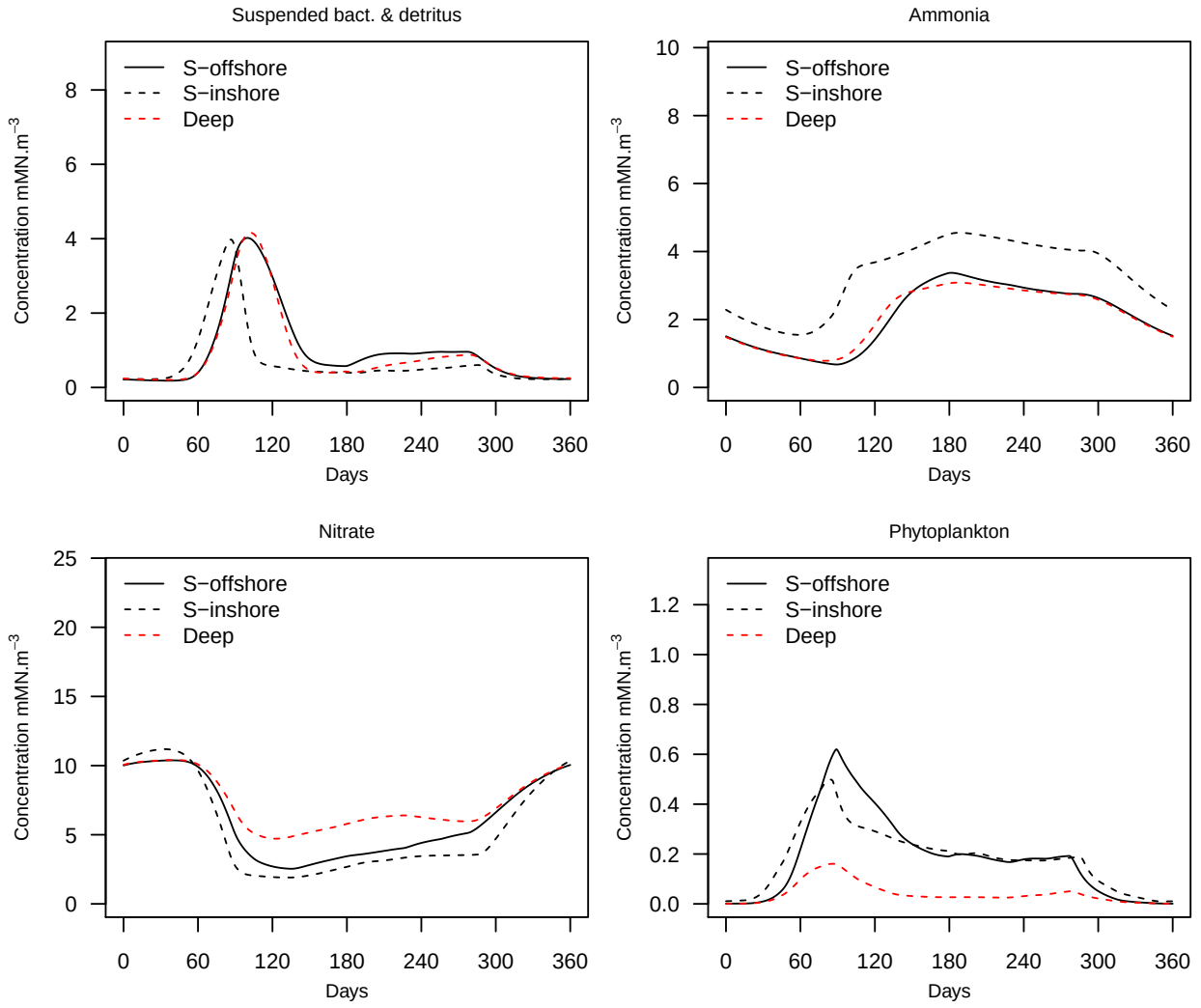


Figure 38. Annual cycles of nutrients and phytoplankton in the final year of a 1970-1999 North Sea model.

```
# Load the 1970-1999 version of the North Sea model supplied with the package and
# plot example credible interval data:
model <- e2e_read("North_Sea", "1970-1999",silent=TRUE)
e2e_plot_eco(model, selection="NUT_PHYT", ci.data=TRUE, use.example=TRUE)
Reading example results from StrathE2E2examples data package for the North_Sea 1970-1999 model
PlotOffshore zone upper layerandSuspended bact & detritus
PlotOffshore zone upper layerandAmmonia
PlotOffshore zone upper layerandNitrate
PlotOffshore zone upper layerandPhytoplankton
PlotOffshore zone lower layerandSuspended bact & detritus
PlotOffshore zone lower layerandAmmonia
PlotOffshore zone lower layerandNitrate
PlotOffshore zone lower layerandPhytoplankton
PlotInshore zoneandSuspended bact & detritus
PlotInshore zoneand Ammonia
PlotInshore zoneandNitrate
PlotInshore zoneandPhytoplankton
```

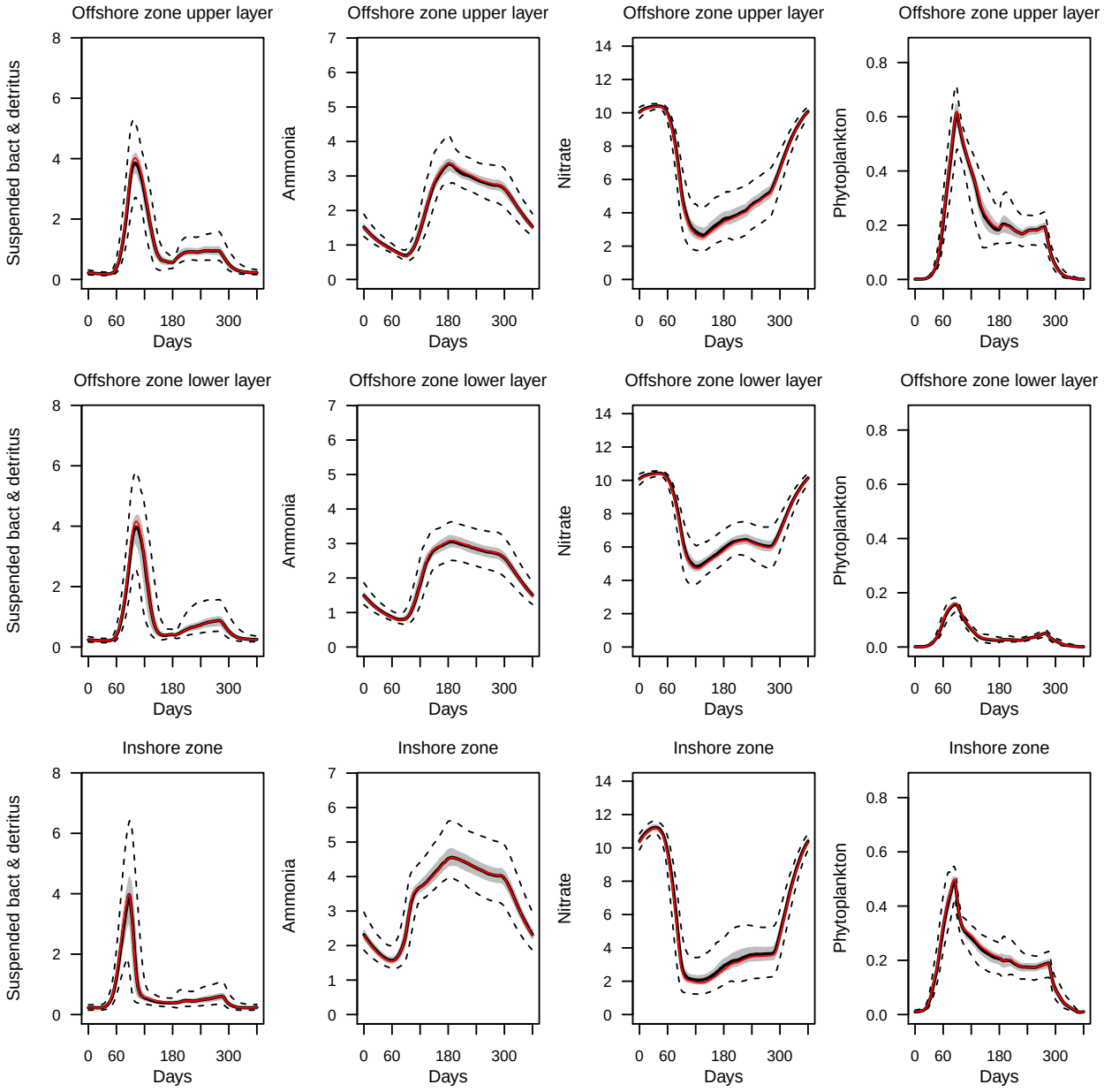


Figure 39. Annual cycles of zooplankton with credible intervals in the final year of a 1970-1999 North Sea model.

Example of directing graphical output to a pdf file

```
pdf("C:/Users/username/Documents/Foldername/plot.pdf",width=7,height=4.667)
                                                # or jpeg("plot.jpg"), png("plot.png")
     e2e_plot_eco(model, "NUT_PHYT", ci.data=TRUE, use.example=TRUE)
dev.off()
```

10.2 Plotting annual cycles of active migration fluxes e2e_plot_migration()

Plot daily data on migration fluxes by actively mobile guilds during the final year of a run, optionally with credible intervals.

Table 41. Arguments of the functions **e2e_plot_migration()**

Argument	Description
model	R-list object defining the model configuration compiled by the e2e_read() function
selection	Text string from a list identifying the group of model output variables to be plotted. Select from: "NUT_PHYT", "SEDIMENT", "ZOOPLANKTON", "FISH", "BENTHOS", "PREDATORS", "CORP_DISC", "MACROPHYTE", Remember to include the phrase within "" quotes
ci.data	Logical. If TRUE plot credible intervals around baseline model results based on Monte Carlo simulation with the e2e_run_mc() function (default=FALSE)
use.saved	Logical. If TRUE use baseline data from a prior user-defined run held as csv files data in the current results folder (default=FALSE)
use.example	Logical. If TRUE use pre-computed example data from the internal North Sea model as the baseline rather than user-generated data (default=FALSE)
results	R-list object of model output generated by the e2e_run() function. Only needed if ci.data=FALSE, use.saved=FALSE and use.example=FALSE. (Default=NULL)

Generate a multi-panel set of one-year time series plots of the mass fluxes between inshore and offshore zones due to migration by actively mobile guilds in the ecology model: all three fish guilds, birds, pinnipeds and cetaceans. The default is to plot data from a single model run but if available, credible intervals of model output from a Monte Carlo analysis can be plotted instead.

Daily interval post-processed data from the Monte Carlo **e2e_run_mc()** function are stored in the file **/results/Modelname/Variantname/CredInt/CredInt_processed_daily_migrations*.csv**, where * represents the model run identifier (model.ident) text embedded in the R-list object created by the **e2e_read()** function.

Arguments determine the source of model data to be plotted. These can be outputs from a single model run with data held in memory as a list object or in a saved csv file, or from a Monte Carlo simulation (using the function **e2e_run_mc()**) to estimate credible intervals of model outputs. Generation of credible interval data is a long computing task, so data for the North Sea model provided with the package are available as example data sets.

Each panel of the plot shows a time-series of the net flux densities (mMN.d^{-1} in the model domain as a whole, assumed to be 1 m^2 sea surface area) of one of the migratory guilds in the model (all three guilds of fish, birds, pinnipeds and cetaceans) between the inshore and offshore zones of the model, over the final year of a run. These migration fluxes are the dynamic product of gradients in the ratio of food concentration to predator concentration across the inshore-offshore boundary. Positive values of the net migration flux indicate net movement from the offshore to inshore zone. Negative values indicate net movement from inshore to offshore.

If plotting of credible intervals is selected, results from the maximum likelihood model are shown by a red line. The median of the credible values distribution is shown my a solid black line. A grey-shaded area indicates the 50% credible interval (spanning quartiles of the cumulative likelihood of simulated values). Black dashed lines span the 99% credible interval.

For details of how the distribution of credible output values from StrathE2E are calculated see help(e2e_run_mc).

```
# Load the 1970-1999 version of the North Sea model supplied with the package,
# run, and generate a plot:
model <- e2e_read("North_Sea", "1970-1999",silent=TRUE)
results <- e2e_run(model, nyears=3)
e2e_plot_migration(model, results=results)
```

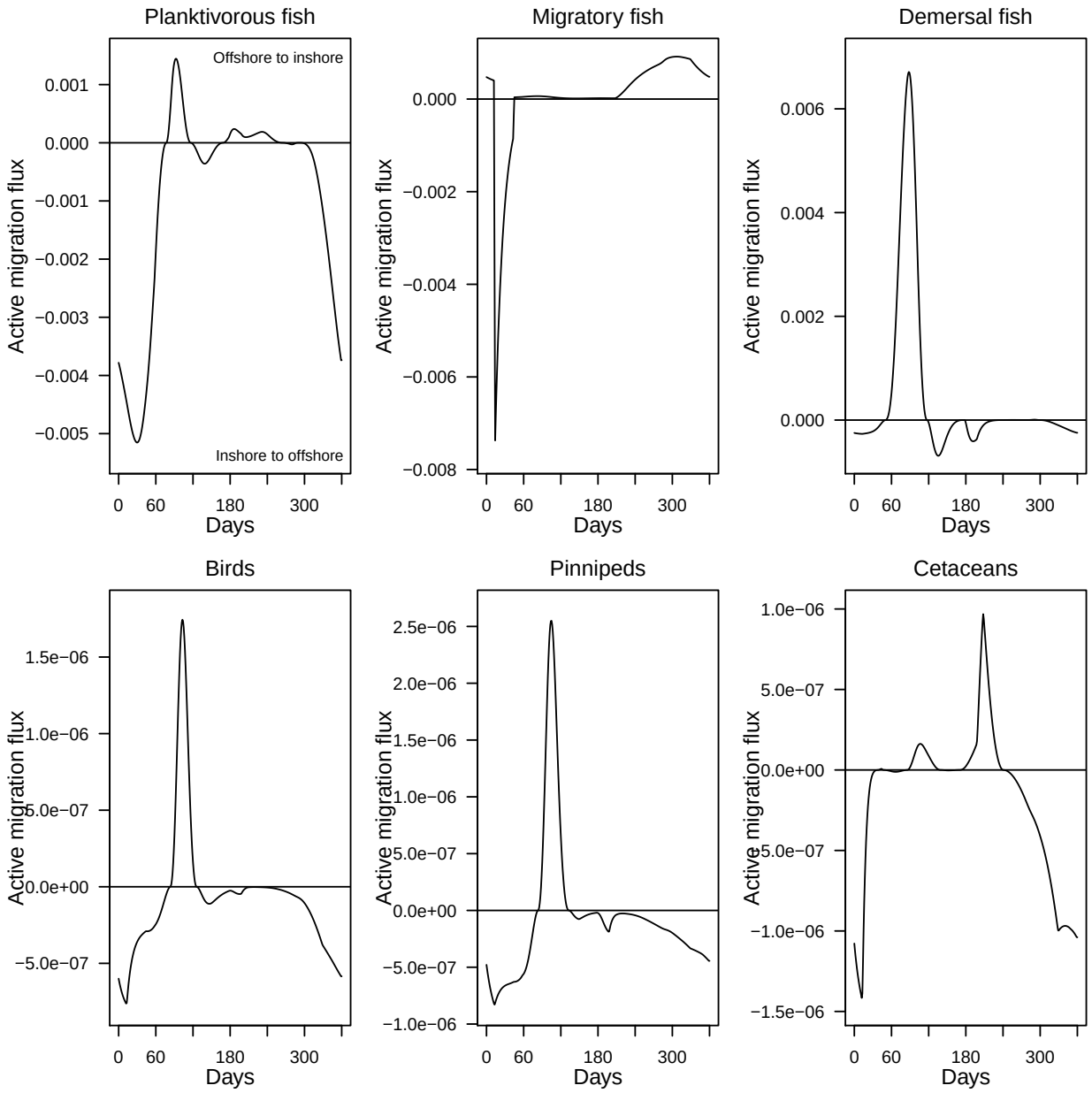


Figure 40. Annual cycles of migration fluxes between the offshore and inshore zones in the final year of a 1970-1999 North Sea model.

```
# For the same model as supplied with the package, plot the example data
# with credible intervals:
model <- e2e_read("North_Sea", "1970-1999",silent=TRUE)
e2e_plot_migration(model, ci.data=TRUE, use.example=TRUE)
Reading example results from StrathE2E2examples data package for the North_Sea 1970-1999 model
```

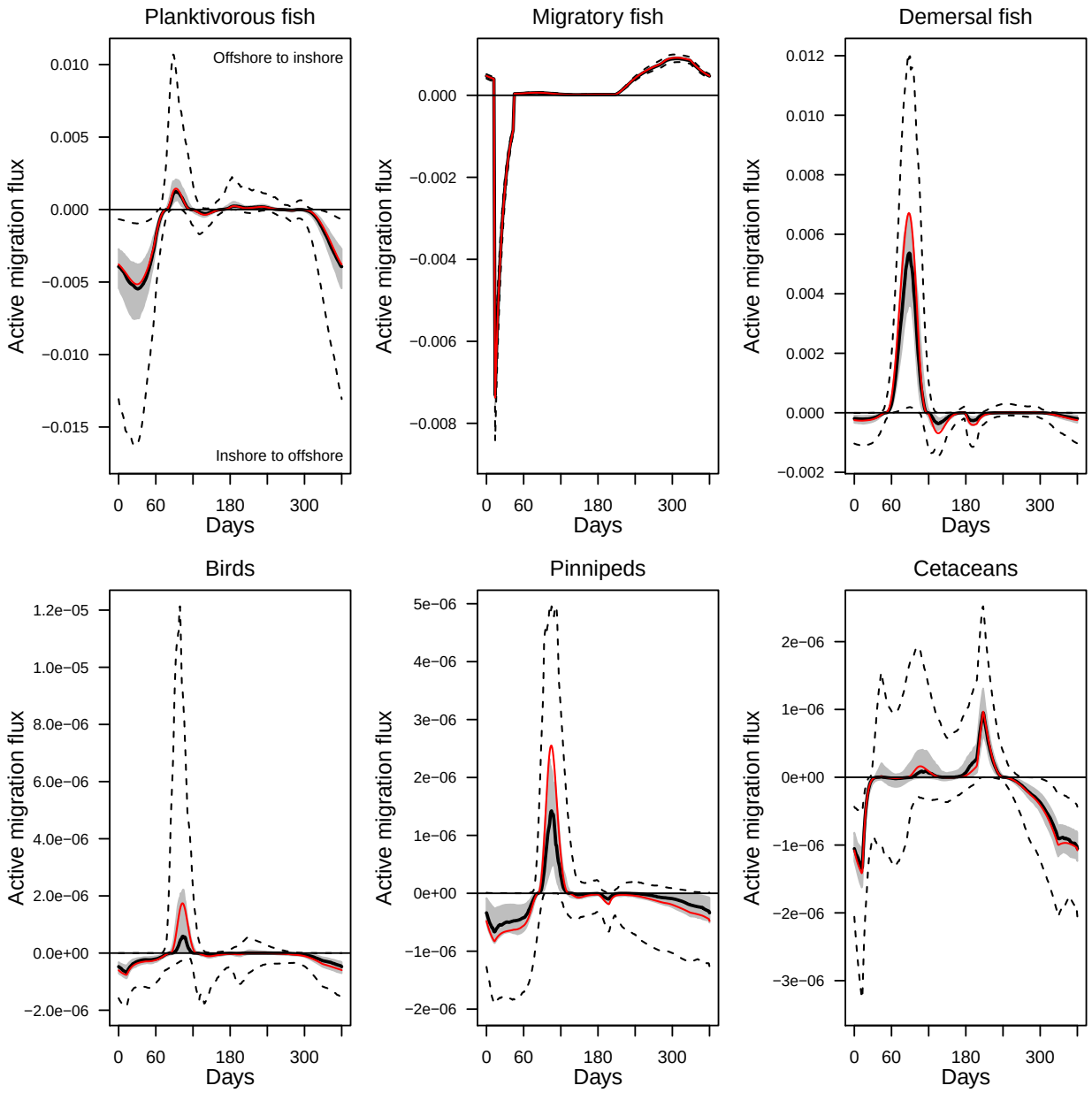


Figure 41. Annual cycles of migration fluxes with credible intervals between the offshore and inshore zones in the final year of a 1970-1999 North Sea model.

10.3 Plotting distributions of catches by gear and guild e2e_plot_catch()

Plot the distribution and composition of annual catches across guild, zones and gears in the final year of a model run.

Table 42. Arguments of the functions **e2e_plot_catch()**

Argument	Description
model	R-list object defining the model configuration compiled by the e2e_read() function
results	List object of single-run model output generated by running the function e2e_run() function

Argument	Description
selection	Text string from a list identifying the group of model output variables to be plotted. Select from: “BY_GUILD”, “BY_GEAR”. With the former, each panel represents a different guild; with the latter, each panel represnts a different gear. Remember to include the phrase within "" quotes

Create stacked barplots of the distribution of offshore and inshore landings and discards across gears by guild, or across guilds by gears, in the final year of a model run.

Data in zonal landings and discards by guild and gear in the final year of a model run are generated as a standard output from the model, and saved both as csv files and in the results object returned by the e2e_run() function. This function organises these data in two different ways to display as barplots.

The first display is a multi-panel plot in which each panel represents a different guild, and the bars show the zonal landings by each gear. The alternative display has each panel as a different gear, and the bars show the landings and discards of each guild.

The unit of the displayed data are mMN.y^{-1} from the model domain as a whole, which is taken as being 1 m^2.

```
# Load the 1970-1999 version of the North Sea model supplied with the package,
# run, and generate a plot:
model <- e2e_read("North_Sea", "1970-1999",silent=TRUE)
results <- e2e_run(model, nyears=3)
e2e_plot_catch(model, results, selection="BY_GEAR")
```

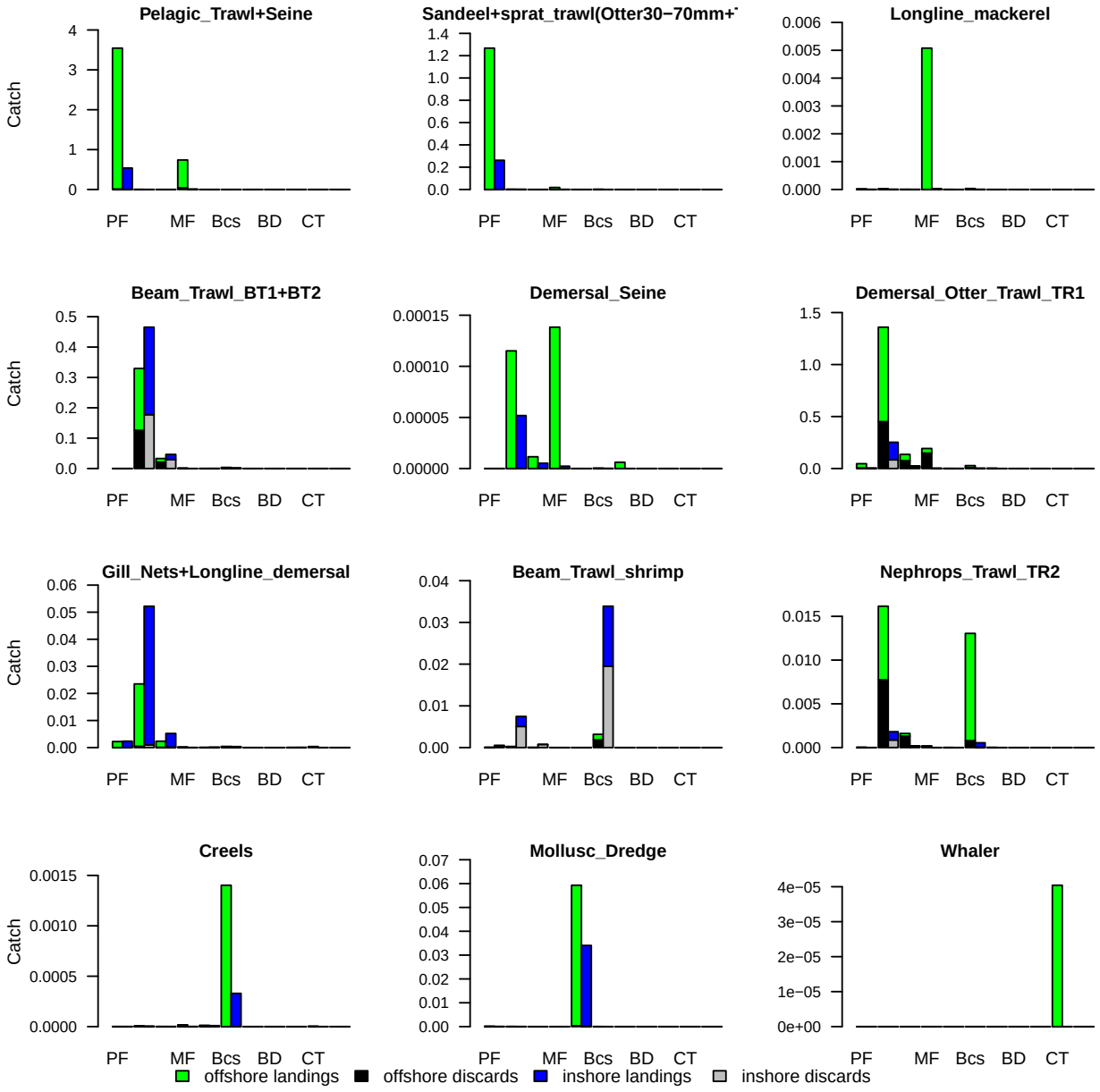


Figure 42. Annual landings and discards of each guild, by gear, in the final year of a run of the 1970-1999 North Sea model.

```
# Load the 1970-1999 version of the North Sea model supplied with the package,
# run, and generate a plot:
e2e_plot_catch(model, results, selection="BY_GUILD")
```

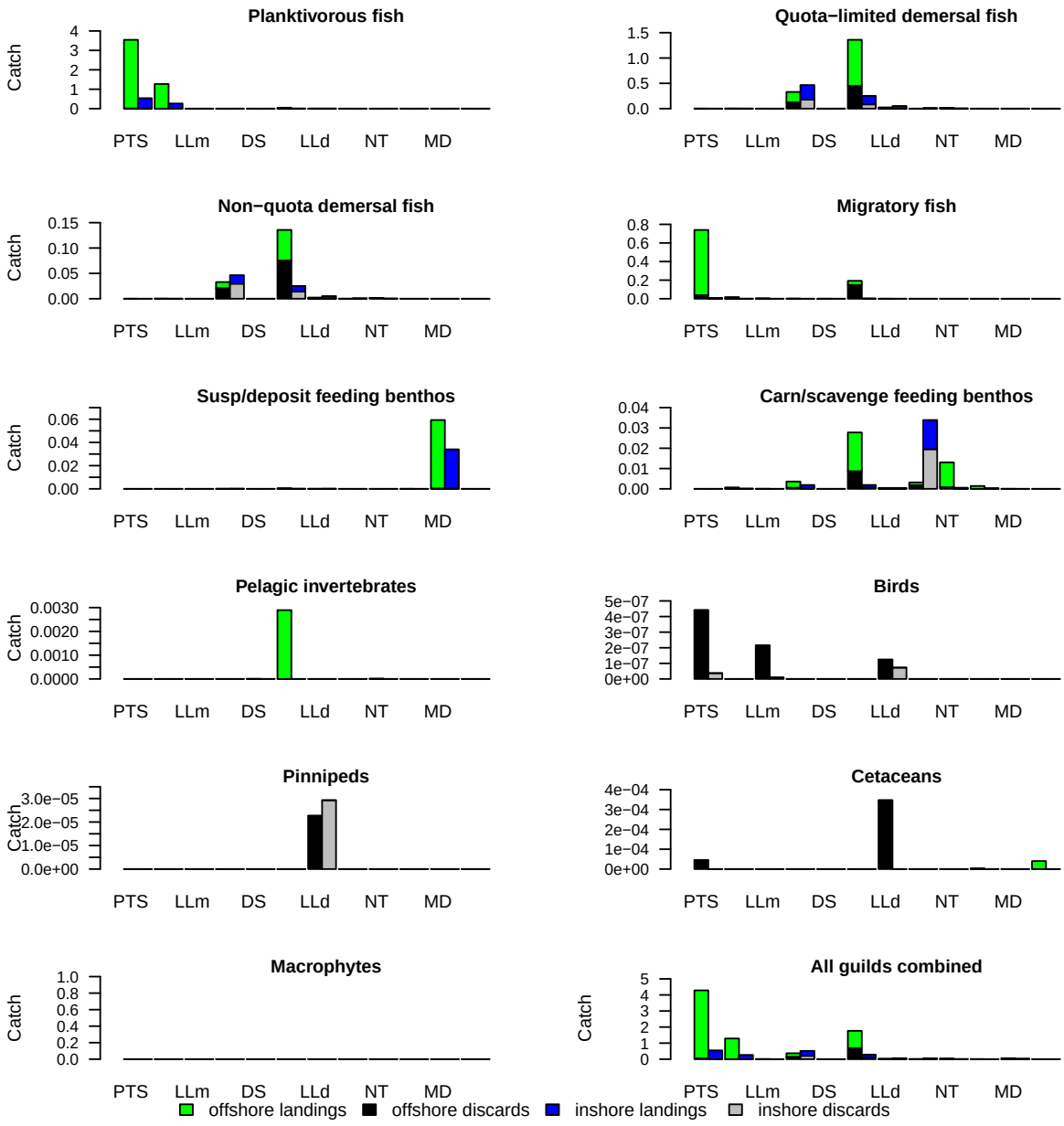


Figure 43. Annual landings and discards by each gear, by guild, in the final year of a run of the 1970-1999 North Sea model.

10.4 Plotting mean trophic level and omnivory indices e2e_plot_trophic()

Plot showing the annual mean trophic level index, and the omnivory index of each guild during the final year of a model run, optionally with credible intervals.

Table 43. Arguments of the functions **e2e_plot_trophic()**

Argument	Description
model	R-list object defining the model configuration compiled by the e2e_read() function
ci.data	Logical. If TRUE plot credible intervals around baseline model results based on Monte Carlo simulation with the e2e_run_mc() function (default=FALSE)

Argument	Description
use.saved	Logical. If TRUE use baseline data from a prior user-defined run held as csv files data in the current results folder (default=FALSE)
use.example	Logical. If TRUE use pre-computed example data from the internal North Sea model as the baseline rather than user-generated data (default=FALSE)
results	R-list object of model output generated by the e2e_run() function. Only needed if ci.data=FALSE, use.saved=FALSE and use.example=FALSE. (Default=NULL)

Generate a two-panel plot showing: (upper panel) the mean trophic level of each guild in the ecology model, and (lower panel) the omnivory index of each guild. The data are generated by the NetIndices package from a flow matrix of nutrient fluxes through, into and out of the ecosystem during the final year of a run. The data are generated automatically as part of the output from every call of the **e2e_run()** function. The default is to plot data from a single model run but if available, credible intervals of model output from a Monte Carlo analysis can be plotted instead.

If credible intervals are plotted these are displayed as box-and-whiskers. The box spans 50% of the likelihood distribution of values and the whiskers 99%. The median is shown by a black tick mark and the maximum likelihhod model by a red tick mark.

Note the the NetINdices package assigns trophic level 1 to inorganic nutrients and detritus, to autotrophs and detritivores are ranked as trophic level 2.

Arguments determine the source of model data to be plotted. These can be outputs from a single model run with data held in memory as a list object or in a saved csv file, or from a Monte Carlo simulation (using the function **e2e_run_mc()**) to estimate credible intervals of model outputs. Generation of credible interval data is a long computing task, so example data for the North Sea model provided with the package are available as illustration.

```
# Load the 1970-1999 version of the North Sea model supplied with the package,
# run, and generate a plot:
model <- e2e_read("North_Sea", "1970-1999",silent=TRUE)
results <- e2e_run(model, nyears=3)
e2e_plot_trophic(model, results=results)
```

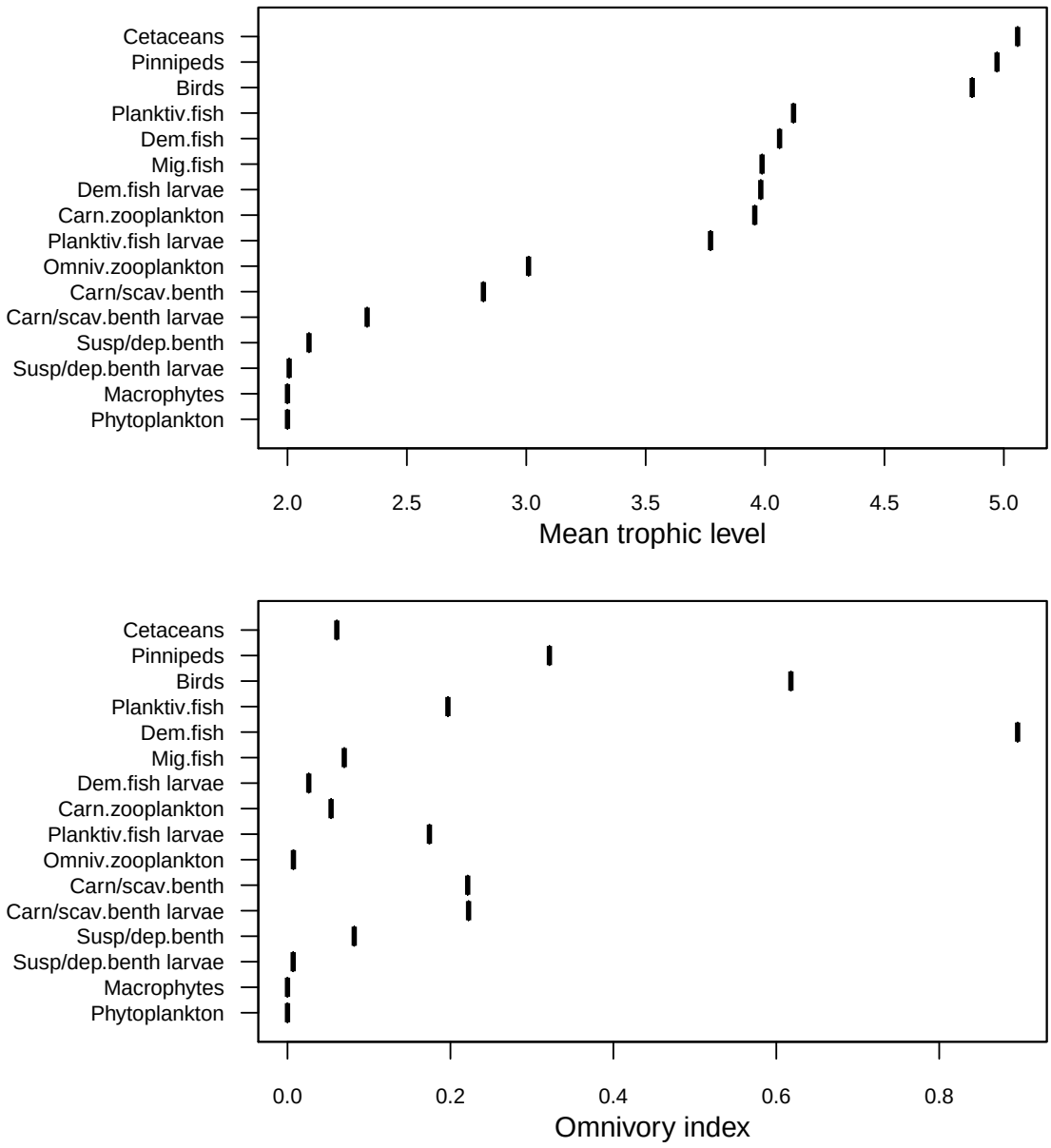


Figure 44. Annual mean trophic level and omnivory indices for the final year of a single run of the 1970-1999 North Sea model.

```
# For the same model as supplied with the package, plot the example data with
# credible intervals:
e2e_plot_trophic(model, ci.data=TRUE, use.example=TRUE)
Reading example results from StrathE2E2examples data package for the North_Sea 1970-1999 model
```

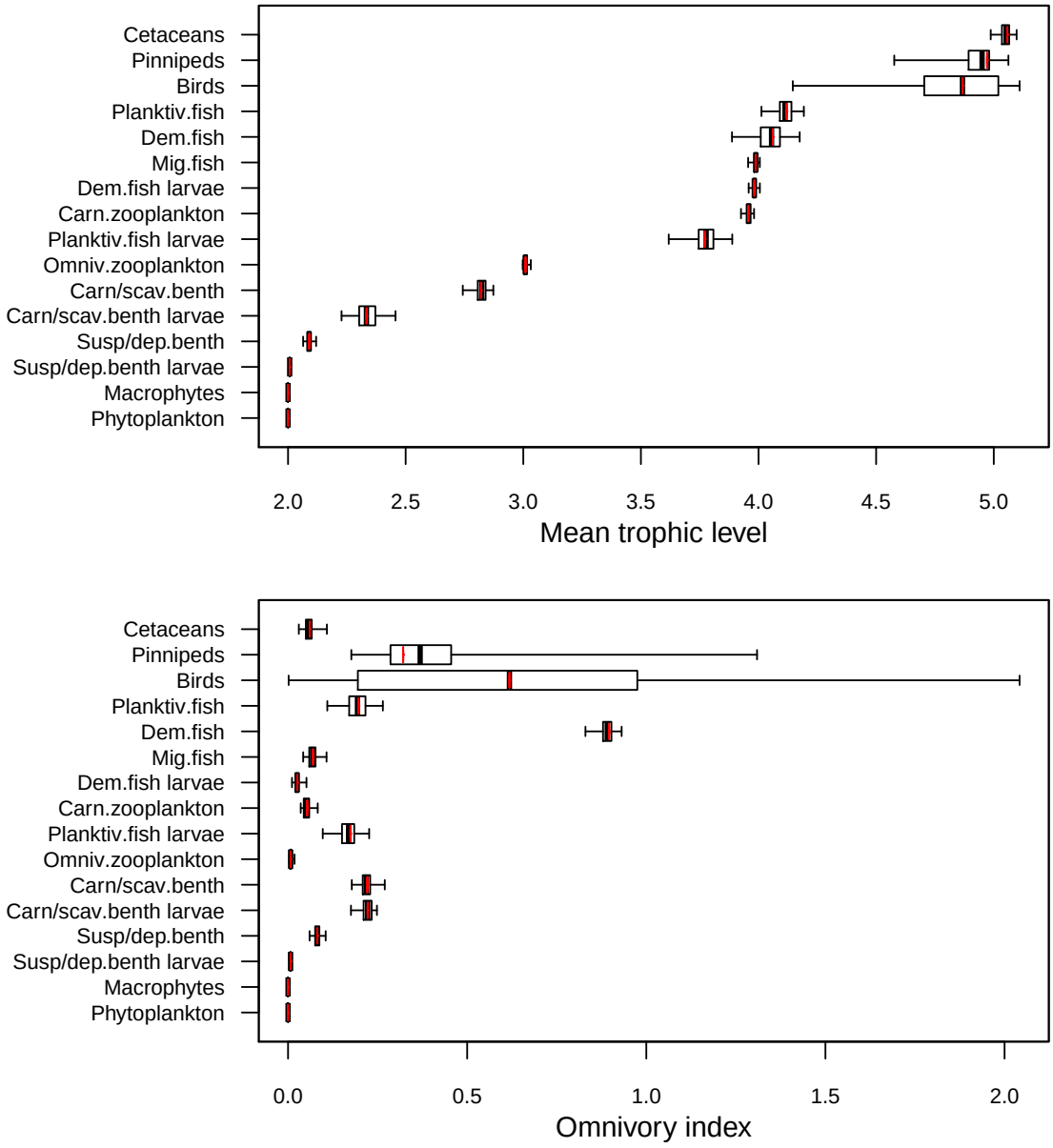


Figure 45. Annual mean trophic level and omnivory indices with credible intervals, for the final year of the 1970-1999 North Sea model.

10.5 Plotting zonal distributions of annual average biomass e2e_plot_biomass()

Plot showing the annual annual average biomass densities of each guild in the inshore and offshore zones, optionally with credible intervals.

Table 44. Arguments of the functions **e2e_plot_biomass()**

Argument	Description
model	R-list object defining the model configuration compiled by the e2e_read() function
ci.data	Logical. If TRUE plot credible intervals around baseline model results based on Monte Carlo simulation with the e2e_run_mc() function (default=FALSE)

Argument	Description
use.saved	Logical. If TRUE use baseline data from a prior user-defined run held as csv files data in the current results folder (default=FALSE)
use.example	Logical. If TRUE use pre-computed example data from the internal North Sea model as the baseline rather than user-generated data (default=FALSE)
results	R-list object of model output generated by the e2e_run() function. Only needed if ci.data=FALSE, use.saved=FALSE and use.example=FALSE. (Default=NULL)

Generate plots showing the annual average biomass densities of each guild in the ecology model in the inshore (blue) and offshore (red) zones during the final year of a run. The default is to plot data from a single model run but if available, credible intervals of model output from a Monte Carlo analysis can also be plotted.

If credible intervals are plotted these are displayed as box-and-whiskers. The box spans 50% of the likelihood distribution of values and the whiskers 99%. The median is shown by a tick mark.

Note that in this plot the biomass are expressed as $mMN.m^{-2}$, meaning that the mass in each zone has been scaled to the zonal sea surface area. So the data in this plot are area densities and not mass.

Arguments determine the source of model data to be plotted. These can be outputs from a single model run with data held in memory as a list object or in a saved csv file, or from a Monte Carlo simulation (using the function **e2e_run_mc()**) to estimate credible intervals of model outputs. Generation of credible interval data is a long computing task, so example data for the North Sea model provided with the package are available as example data sets.

```
# Load the 1970-1999 version of the North Sea model supplied with the package,
# run, and generate a plot:
model <- e2e_read("North_Sea", "1970-1999",silent=TRUE)
results <- e2e_run(model, nyears=3)
e2e_plot_biomass(model, results=results)
```

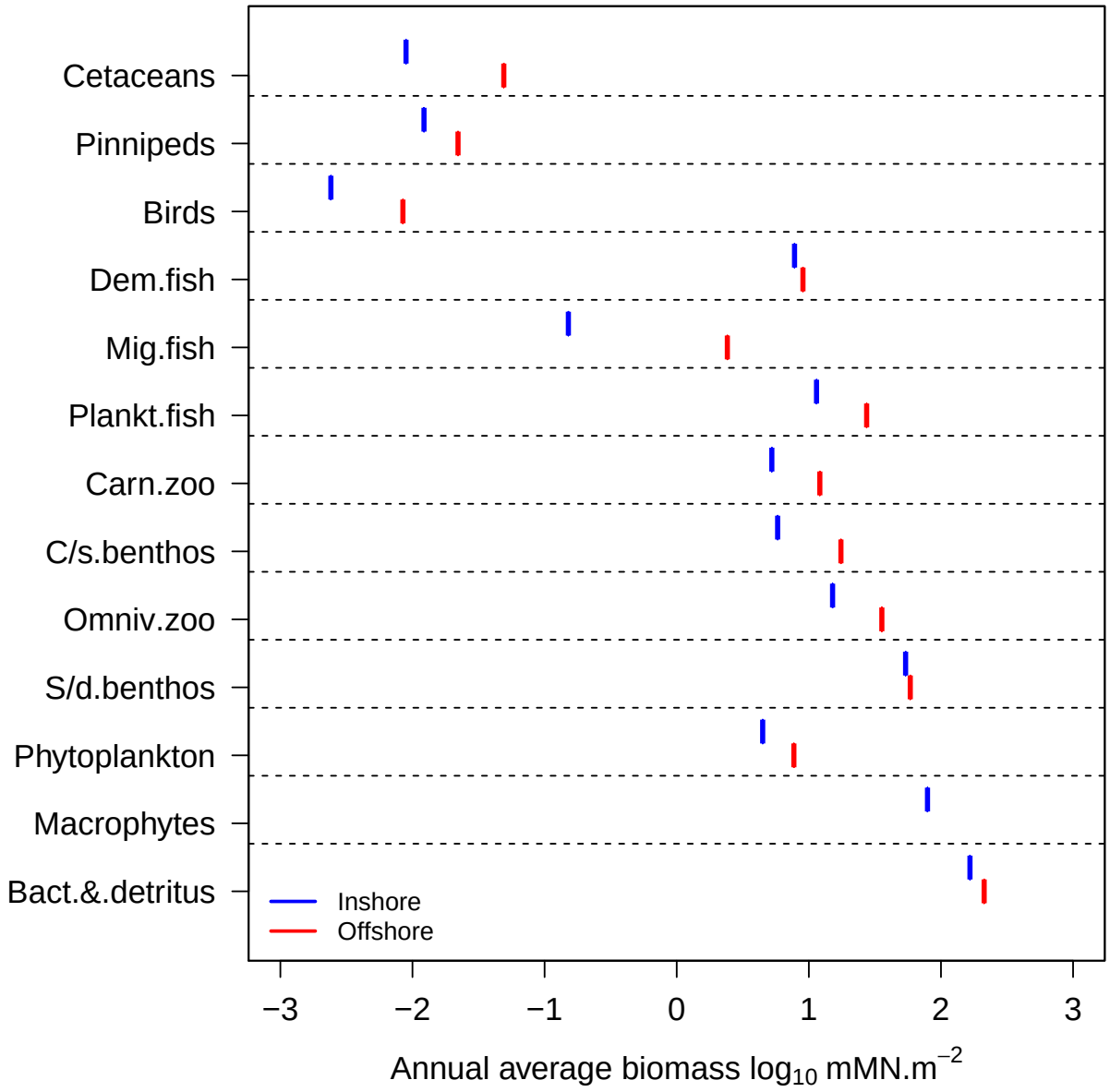


Figure 46. Annual average biomass densities for the final year of a single run of the 1970-1999 North Sea model.

```
# For the same model as supplied with the package, plot the example data
# with credible intervals:
e2e_plot_biomass(model, ci.data=TRUE, use.example=TRUE)
Reading example results from StrathE2E2examples data package for the North_Sea 1970-1999 model
Reading example results from StrathE2E2examples data package for the North_Sea 1970-1999 model
```

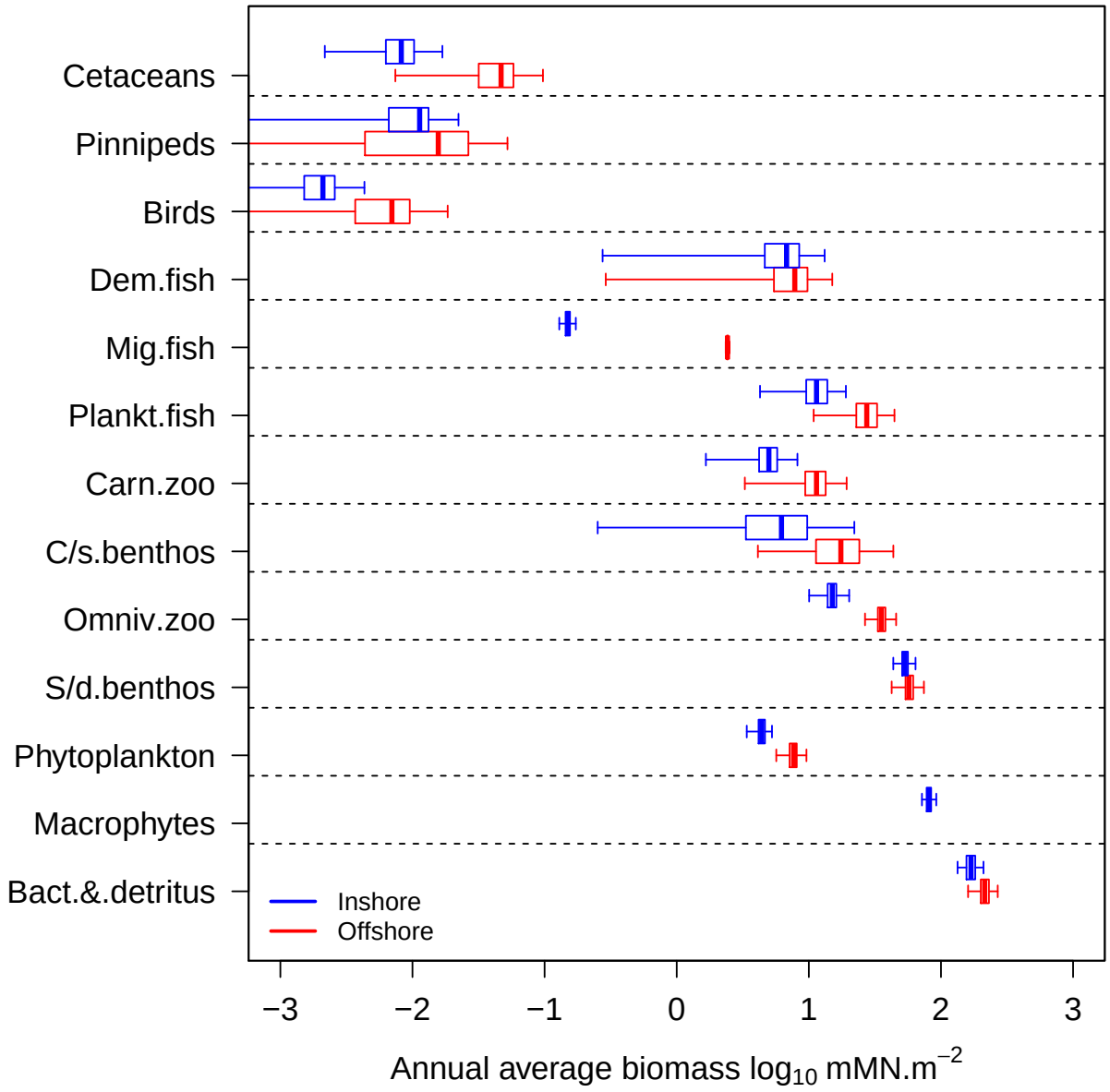


Figure 47. Annual average biomass densities with credible intervals for the final year of a single run of the 1970-1999 North Sea model.

References

Beddington, J.R. (1975). Mutual interference between parasites or predators and its effect on searching efficiency. *Journal of Animal Ecology*, **51**, 331-340.

Bertsimas, D. & Tsitsiklis, J. (1993). Simulated annealing. *Statistical Science*, **8**, 10-15.

Cerny, V. (1985). A thermodynamic approach to the travelling salesman problem: An efficient simulation. *Journal of Optimization Theory and Applications*, **54**, 41-51.

DeAngelis, D.L., Goldstein, R.A.. & O'Neill, R.V. (1975). A model for trophic interaction. *Ecology*, **56**, 881-892.

Heath, M.R., Speirs, D.C. & Steele, J.H. (2014). Understanding patterns and processes in models of trophic cascades. *Ecology Letters*, **17**, 101-114.

Kirkpatrick, S., Gelett, C.D. & Vecchi, M.P. (1983). Optimisation by simulated annealing. *Science*, **220**, 621-630.

Morris, M.D. (1991). Factorial sampling plans for preliminary computational experiments. *Technometrics*, **33**, 161-174.

Pace, M.L., Cole, J.J., Carpenter, S.R. & Kitchell, J.F. (1999) Trophic cascades revealed in diverse ecosystems. *Trends in Ecology and Evolution*, **14**, 483-488.

Wu, J., Dhingra, R., Gambhir, M. & Remais, J.V. (2013). Sensitivity analysis of infectious disease models: methods, advances and their application. *Journal of the Royal Society Interface*, **10**: 20121018, 14pp.