THE SMARTR PACKAGE

1
Introduction

The package `smartR` is a tool for assessing bio-economic feedback in different management scenarios. `smartR` (Spatial Management and Assessment of demersal Resources for Trawl fisheries) combines information from different tasks gathered within the European Data Collection Framework for the fishery sector. The `smartR` package implements the SMART model[1], through the object-oriented programming paradigm, and within this package, it is possible to achieve the complete set of analyses required by the SMART approach: editing and formatting of the raw data; construction and maintenance of coherent datasets; numerical and visual inspection of the generated metadata; simulation of management scenarios and forecast of their effects. The interaction between the user and the application could take place by calling of methods via the command line or could be entirely operated from the graphical user interfaces (GUI). In this short guide, you will find instructions for operation via both the command line and the GUI.

[1] "SMART: A Spatially Explicit Bio-Economic Model for Assessing and Managing Demersal Fisheries, with an Application to Italian Trawlers in the Strait of Sicily."

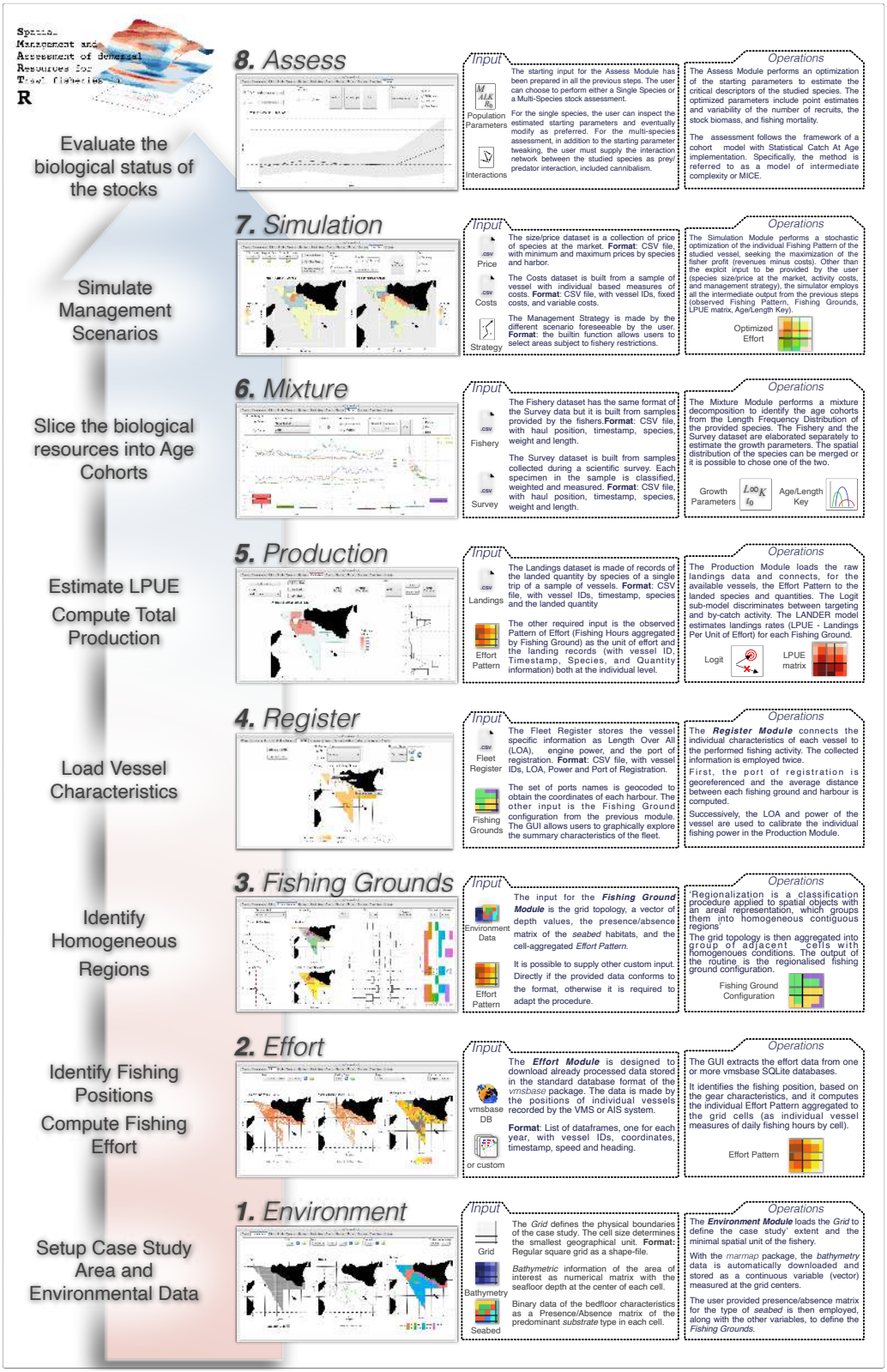

Figure 1.1: Complex Workflow

2
Overview

This vignette is meant to give a jumpstart to new users of the `smartR` package. The following text illustrates the series of instructions needed to start using the package. Further information and references are provided throughout the text, however, it is advisable to have a quick look at the foundational paper that inspired the development of `smartR`. The initial description of the basic concepts and first implementation of the model is in Russo et al. [2014]. However, to get a broader view of a SMART project, Russo et al. [2019][1] shows a practical application of the `smartR` package to two case studies within the context of the MANTIS EU project.

Finally, for more details, in Russo et al. [2018][2] a novel technique for the estimation of the Landings Per Unit of Effort (LPUE) has been proposed and adopted by the SMART model.

[1] Russo et al. 2019. "Simulating the Effects of Alternative Management Measures of Trawl Fisheries in the Central Mediterranean Sea. Frontiers in Marine Science

[2] Russo et al. 2018. "A Model Combining Landings and VMS Data to Estimate Landings by Fishing Ground and Harbor." Fisheries Research

3
Installation

It is possible to install the last stable version of the `smartR` package from the CRAN with:

```
# Install `smartR` package from CRAN
install.packages("smartR")
```

Alternatively, it is possible to install the developing version from GitHub with:

```
# Load devtools
library(devtools)

# Install `smartR` package from github
install_github(repo = "d-lorenz/smartR")
```

Be sure of having all the dependencies already available when performing the installation of `smartR` from GitHub, otherwise you can get some errors/warnings.

4
GUIs & Modules

EIGHT main (and one accessory) Graphical User Interfaces (GUI), or Modules (see Fig. 1), guide the smartR workflow: 1) Environment configures the case study area with three environmental layers (grid, bathymetry, and seabed); 2) Effort loads the fishing effort database, assigns fishing locations, and aggregates the data to the grid (as Fishing Hours); 3) Fishing Grounds subdivides the study area into homogeneous regions; 4) Register loads fleet register data (Vessel IDs, length, power, and registration port); 5) Production reconstructs the spatial origin of the catches and estimates the Landings (or Catches) per Unit of Effort (i.e. LPUE as Kgs * Fishing Hours * vessel length) for each fishing ground; 6) Mixture and Cohorts (cohorts is the accessory GUI) loads georeferenced Length Frequency Distributions (LFD) from the survey and fishery datasets, determines growth parameters, subdivides the studied stocks into cohorts, and visualizes the spatial distribution of the cohorts; 7) Simulation estimates costs and revenues, and simulates different management scenarios; 8) Assess evaluates the biological status of the studied stocks.

SMARTR adopts the object-oriented framework provided by the `R6` Chang [2019][1] package. Even if `smartR` is composed of eight functional modules, we have structurally separated the `smartR` package in three distinct classes: the 'Environment' class, the 'Fleet' class and the 'Resource' class enclosed within a main 'Project' class. Each case study is initialized as a new `smartR` project: At the beginning of the workflow, these entities are clearly distinct, while they blend together in the later steps (i.e. production is a mix of effort data and resource data).

[1] Winston 2019. R6: Encapsulated Classes with Reference Semantics

HERE are listed the most important elements of a `smartR` project. The first, smartRgui, is the GUI developed to guide and assist the user, while the other four are the classes that make up the `smartR` package:

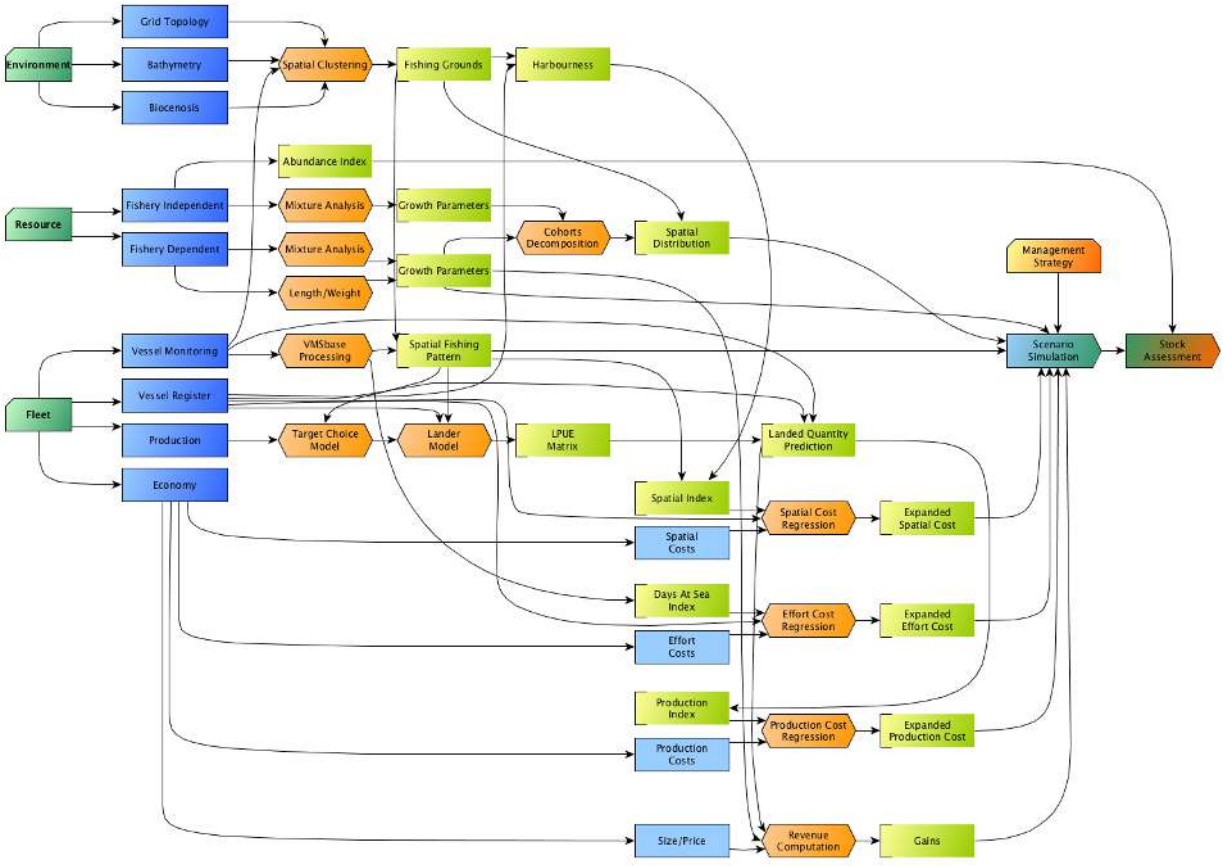

Figure 4.1: Complex Workflow

- smartRgui GUI to assist the analysis
- SmartProject main project class * Environment class * Fleet class * Resource class

The GUIs are provided by the `gWidgets2` R package Verzani [2019][2] and the `GTK` library.

The following sections will illustrate the GUIs usage and functionalities, along with the command-line instruction to follow the common workflow of a smartR project. Each section will describe the format of the required input and the structure of the resulting output.

[2] Verzani 2019. gWidgets API for Building Toolkit-Independent, Interactive GUIs

GTK: Library containing a set of graphical control elements (called widgets) used to construct the GUI of programs. Wikipedia

4.1 Environment

The Environment Module collects, process and stores the environmental information required for a `smartR` case study analysis.

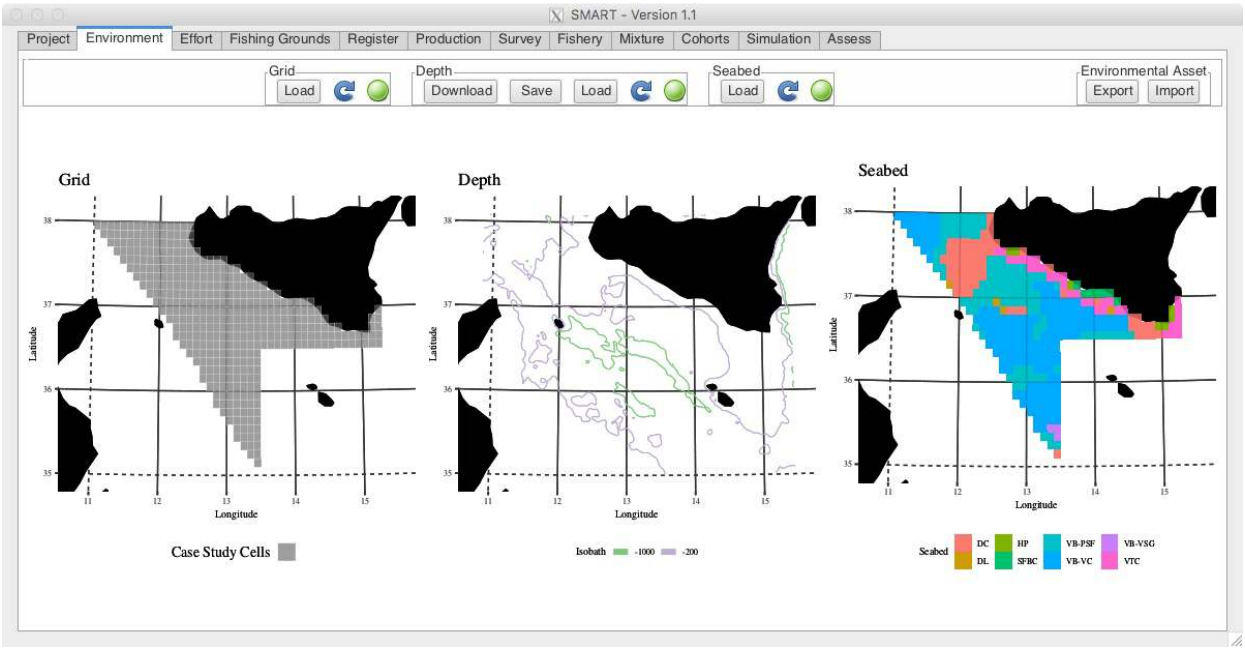

Figure 4.2: Screenshot of Environment GUI

```
# Locate the example environment asset' file
envAssetPath <- system.file("extdata/mapAsset.RDS", package = "smartR")
```

```
# Load environment asset' data
yourSmartRstudy$importEnv(readRDS(envAssetPath))

# Setup case study' map
yourSmartRstudy$sampMap$getGooMap()
yourSmartRstudy$sampMap$setGooGrid()
yourSmartRstudy$sampMap$setGooBbox()
yourSmartRstudy$sampMap$setGgDepth()
yourSmartRstudy$sampMap$setGgBioDF()

# View case study' grid
print(yourSmartRstudy$sampMap$gooGrid)
```

Table 4.1: Widgets of the Environment GUI

Widget	Function
Grid	
Load	Opens a file selection window to choose a shapefile and set up the graphical output
Depth	
Download	Downloads the bathymetry of the area of interest
Save	Stores the bathimetric data as an XYZ matrix
Load	Loads an XYZ matrix as bathymetric data
Seabed	
Load	Opens a file selection window to choose a seabed shapefile and set up the graphical output
Asset	
Export	Stores grid, depth and seabed data as an RDS object
Import	Loads the environmental RDS object with grid, depth and seabed data

The grid input

Shapefile: a geospatial vector data format for geographic information system (GIS) software. Wikipedia

The topology of the area of study should be provided as a grid of rectangular polygons in the shapefile format. The following code chunk shows the typical instructions to load the grid topology and set up the `smartR` case study.

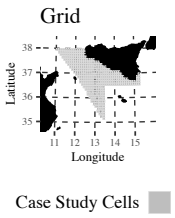

Figure 4.3: Loaded Grid

```
# Load the shapefile
yourSmartRstudy$loadMap("Data/Environment/
                         Grid/areaOfStudy.shp")

# Setup the graphical device
yourSmartRstudy$sampMap$getGooMap()

# Setup the `smartR` grid object
yourSmartRstudy$sampMap$setGooGrid()

# Setup the bounding box for the case study
yourSmartRstudy$sampMap$setGooBbox()
```

```
## Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
##   ..@ data       :'data.frame':  394 obs. of  1 variable:
##   ..@ polygons   :List of 394
##   ..@ plotOrder  : int [1:394] 197 296 291 293 175 366 373 298 66 72 ...
##   ..@ bbox       : num [1:2, 1:2] 11 35.1 15.3 38
##   .. ..- attr(*, "dimnames")=List of 2
##   ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
## NULL
```

The seabed input

The sea bottom characteristic (EUNIS classification scheme for marine habitats by Davies et al. [2004][3]) of each cell of the topological grid can be loaded as a presence/absence matrix annotated according to the prevalent substrate type.

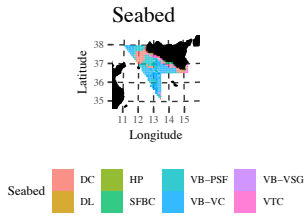

Figure 4.4: Loaded Seabed

[3] Davies et al. 2004. EUNIS Habitat Classification

```
# Load the seabed matrix
yourSmartRstudy$sampMap$loadBioDF("Data/Environment/
                                   Seabed/seabed.RDS")
```

DC	VB-PSF	VB-VSG	VB-VC	VTC	HP	DL	SFBC
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0

Table 4.2: A subset of the seabed input matrix.

The bathymetry

```
## Warning: Computation failed in `stat_contour()`:
## could not find function "contour_lines"
```

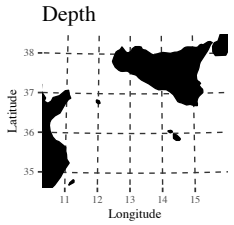

Figure 4.5: Loaded Bathymetry

THROUGH the functionalities provided by the `marmap`[4] package, `smartR` automatically downloads the bathymetry of the area of interest from the ETOPO1 database Amante and Eakins [2009][5] through the NOAA servers.

[4] Pante & Simon-Bouhet 2013. marmap: A Package for Importing, Plotting and Analyzing Bathymetric and Topographic Data in R. PLoS ONE

[5] ETOPO1 1 Arc-Minute Global Relief Model

```
# Download the bathimetry from NOAA with marmap
yourSmartRstudy$sampMap$getGridBath()

# Save the bathimetry matrix as RDS
saveRDS(object = yourSmartRstudy$sampMap$gridBathy,
        file = "Data/Environment/Bathymetry/bathymetry.RDS")

# Load the bathymetry matrix
yourSmartRstudy$sampMap$loadGridBath("Data/Environment/
                                     Bathymetry/bathymetry.RDS")

## Bathymetric data of class 'bathy', with 277 rows and 185 columns

## Latitudinal range: 35.02 to 38.08 (35.02 N to 38.08 N)

## Longitudinal range: 10.85 to 15.45 (10.85 E to 15.45 E)

## Cell size: 1 minute(s)

##

## Depth statistics:

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2740.0  -472.0  -145.0  -217.1   -39.0  3090.0

##

## First 5 columns and rows of the bathymetric matrix:

##                  35.0183333333333 35.034981884058 35.0516304347826
## 10.8483333333333               44              55               62
## 10.8650120772947               36              45               56
## 10.881690821256                30              39               47
## 10.8983695652174               27              35               40
## 10.9150483091787               28              32               33
##                  35.0682789855072 35.0849275362319
```

```
## 10.8483333333333              69              79
## 10.8650120772947              71              79
## 10.881690821256               56              64
## 10.8983695652174              44              52
## 10.9150483091787              33              41
```

4.2 Effort

The Fleet Module collects, process and organizes all the information about the studied fleet such as:

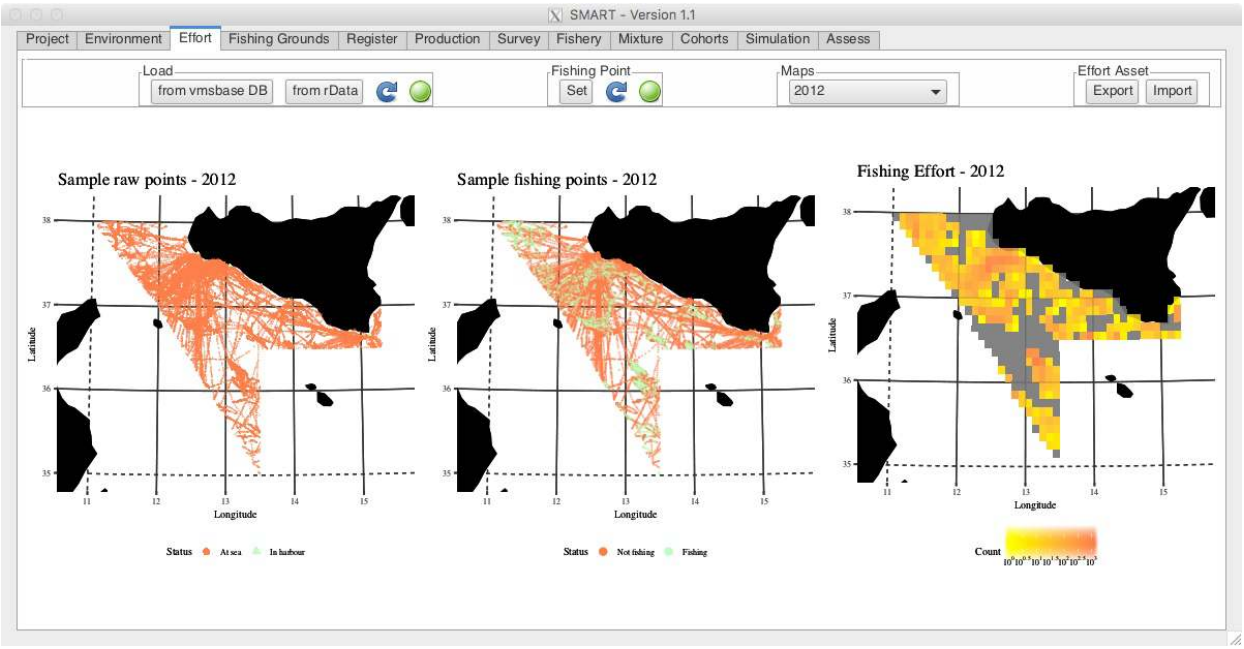

Figure 4.6: smartR GUI Fleet

- VMS/AIS positions
- Fleet Register
- Landings

Widget	Function
Load	
from vmsbase DB	Opens a file selection window to choose a proper vmsbase DB
from rData	Opens a file selection window to choose an already exported rData file with vms data
Fishing Point	
Set	Opens a new window to set the speed and depth parameters to filter the fishing points
Maps	
Droplist	Draws the raw points, fishing points and gridded data for the selected temporal frame
Asset	
Export	Stores raw points, fishing points and gridded data as an RDS object
Import	Loads the effort RDS object with raw points, fishing points and gridded data

VMS/AIS positions

The VMS/AIS position records brings information about the spatial and temporal activity of the fishing fleet. The `smartR` package can directly query a `vmsbase` database and process the information to obtain the gridded fishing hours.

VMS: Vessel Monitoring Systems. Wikipedia

AIS: Automatic Identification System. Wikipedia

```
# Initialize Fleet class
yourSmartRstudy$createFleet()

# Extract data from vmsbase DBs
yourSmartRstudy$loadFleeEffoDbs(
               effort_path = "Path/to/VMS_AIS_DBs",
               met_nam = "Selected Fishing Gear",
               onBox = TRUE,
               perOnBox = 0.9))

# Setup fishing vessel ids
yourSmartRstudy$fleet$setEffortIds()
```

The raw positions are mandatory for all the subsequent analyses. It is possible to load this data without having it stored into one or more `vmsbase` DBs assuming that the format is correct. The custom data

must be loaded into the `rawEffort` attribute of the `fleet` class and it should be structured as a list of data.frames with one data.frame for each year of data. Each data.frame should have the following fields:

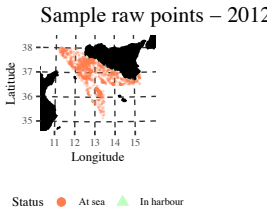

Figure 4.7: Raw Points

```
##         I_NCEE      LAT      LON     DATE       SPE       HEA
## 450513  24713 36.61616 15.15775 15591.40 16.239317  98.02264
## 435620  17930 36.59722 15.08722 15687.64 15.186400  14.00000
## 376449  25310 36.57085 13.52204 15430.88  6.434147 122.24989
## 255314   7908 37.49106 12.84213 15518.58 17.594000 100.40000
## 83546    5471 37.13960 12.45873 15413.17  8.182793 318.14045
## 265272   5471 37.50181 12.91483 15555.01  7.694736 235.10703
```

It is extremely important to have the individual based information to proceed further in the analyses. It is not possible to load and analyse aggregated data, such as already gridded information at the whole fleet level without the single vessel records.

Fishing positions

The dataset with the raw effort is split between steaming and fishing positions. The determination of the fishing positions is performed using a combined speed and depth filter, characterising the fishing operations of the studied metier (i.e. required depth range and technical speed range of the fishing gear).

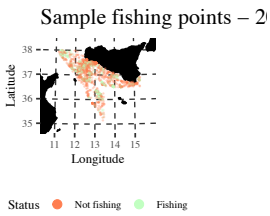

Figure 4.8: Fishing Points

```
# View speed distribution to setup fishing point filter
yourSmartRstudy$fleet$plotSpeedDepth(which_year = "2012",
                                     speed_range = c(2, 8),
                                     depth_range = c(-20, -600) )

# Setup fishing points' filter
yourSmartRstudy$fleet$setFishPoinPara(speed_range = c(2, 8),
                                      depth_range = c(-20, -600))

# Compute fishing points
yourSmartRstudy$fleet$setFishPoin()
```

Gridded Effort

```
## Warning: Transformation introduced infinite values in discrete y-axis
```

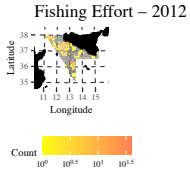

Figure 4.9: Gridded Effort

The fishing positions are then spatially aggregated to the environmental grid and temporally subdivided in daily records. The sum of fishing positions within each cell is then divided by the interpolation frequency of the native pings to obtain the individual pattern of effort of each vessel in each cell/fishing ground of the case study. The unit of effort, subsequently employed in the following sections, is then defined as the Hours of Fishing.

```
# Assign cell id to each fishing point
yourSmartRstudy$setCellPoin()

# Add week and month number to each point
yourSmartRstudy$fleet$setWeekMonthNum()
```

4.3 Fishing Ground

The grid topology can be aggregated into groups of adjacent cells with homogeneous conditions. Spatial clustering analysis is performed with the skater package which regionalizes the grid.

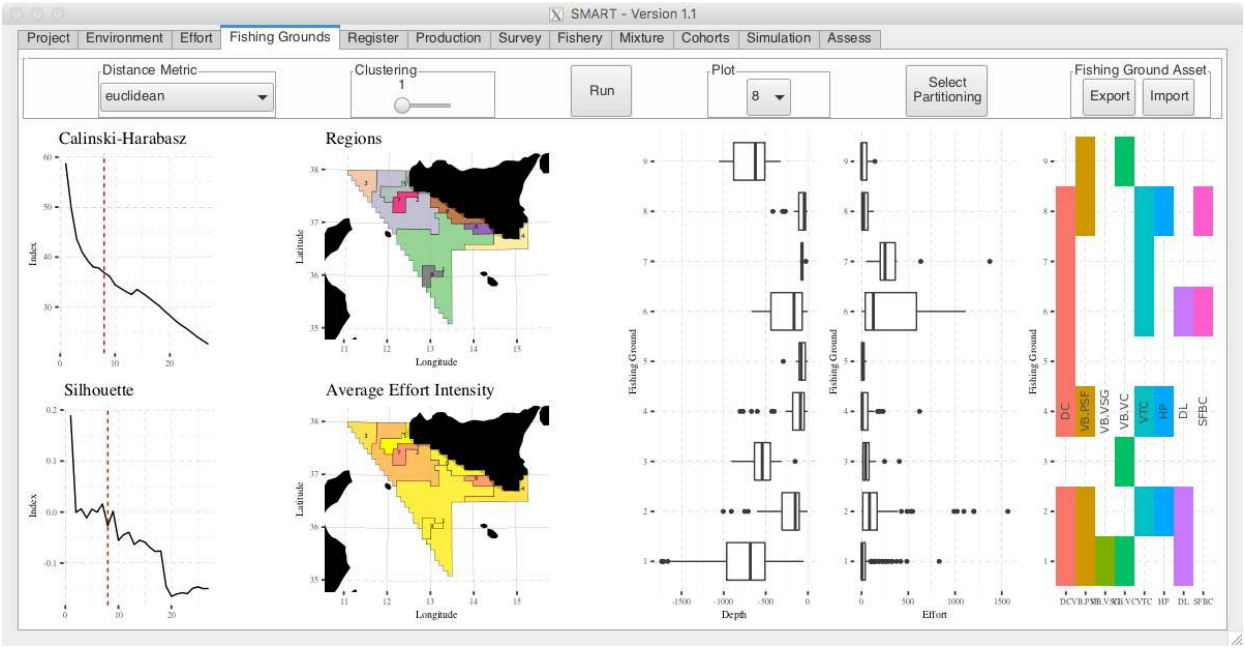

Figure 4.10: smartR GUI Fleet

From the `skater` package description:

> *Regionalization is a classification procedure applied to spatial objects with an areal representation, which groups them into homogeneous contiguous regions*

The input is the grid, and a vector of bathymetry, the absence/presence matrix of the seabed habitats, and the cell-aggregated fishing effort. After applying the SKATER procedure, the contiguous cells of the grid sharing common characteristics are assembled to get larger polygons based on the number of clusters selected by the user. The resulting configuration is treated as the fundamental spatial partition for the subsequent analyses.

```
# Load available data
my_sampling$setAvailData()
```

Widget	Function
Distance Metric	
Droplist	Select different distance or similarity measures
Clustering	
Slider	Select the maximum number of clusters to test
Run	
Button	Start the SKATER clustering routine
Plot	
Droplist	Show the output of the slicing with different number of clusters
Select Partitioning	
Button	Select the current numbers of clusters
Asset	
Export	Stores the regionalisation result as an RDS object
Import	Loads the previously saved regionalisation result from an RDS object

```
# Setup clustering input
my_sampling$sampMap$setClusInpu()

# Fishign ground clustering
my_sampling$sampMap$calcFishGrou(numCuts = my_sampling$sampMap$nCells-1 ,
                                 minsize = 1,
                                 modeska = "S",
                                 skater_method = "manhattan")

# Setup selected regionalization
my_sampling$sampMap$setCutResult(ind_clu = my_sampling$sampMap$nCells-1 )
```

4.4 Register

LOA: Length OverAll. Wikipedia

The Fleet register data stores the structural information of each vessel. The required fields are the vessel ID, LOA, engine power, and the port of registration.

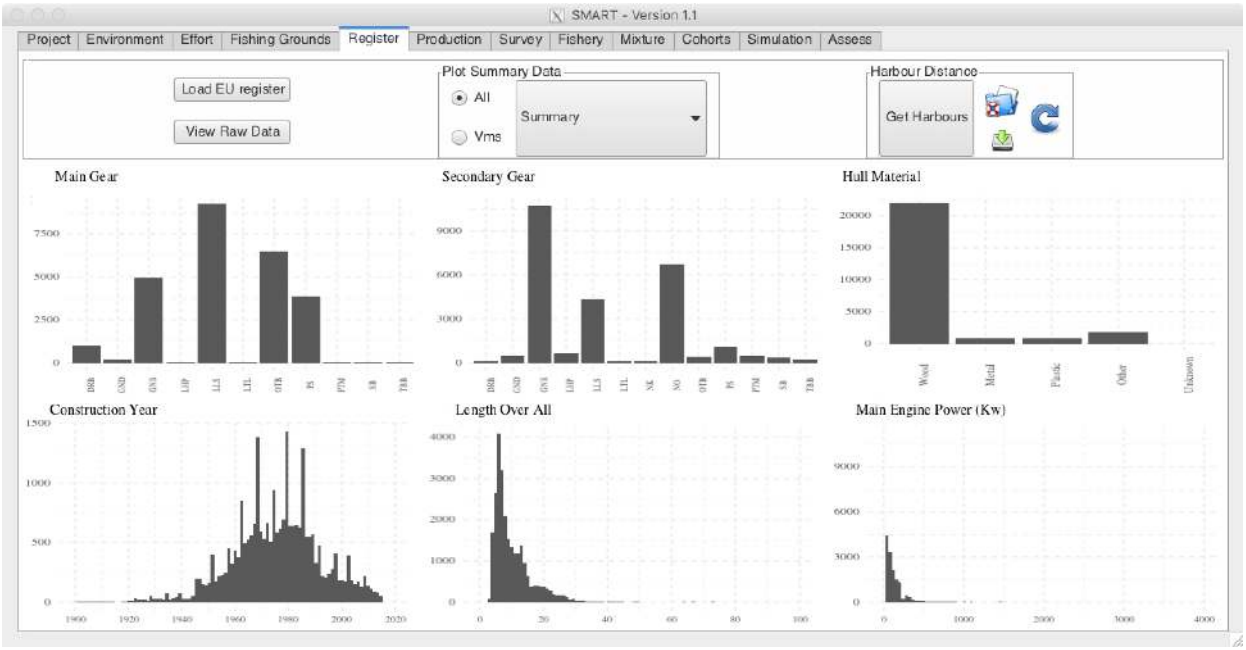

Figure 4.11: smartR GUI Fleet

Usually, this information is collected by the local port authorities. For example, the EU commission database share this information through a centralized registry available on their website

```
# Load the fleet register
my_sampling$fleet$loadFleetRegis(register_path = "path/to/fleetRegister.csv")

# Remove unused fields and records
my_sampling$fleet$cleanRegister()

# Subset the total fleet register to get only the vessel on the effort dataset
my_sampling$fleet$setVmsRegister()

# Visualize summary data
ggplot_registerDispatch(curRegister = myFishery$fleet$rawRegister,
                        selPlot = "Summary")
```

Widget	Function
Load EU register	Opens a file selection window to choose a csv file with the standard output provided the FAO
View Raw Data	Opens a new window to show the loaded raw data
Summary	
Radio Buttons	Shows summary statistics for all the loaded register data or for only the vms equipped vessels
Harbour	
Get Harbours	Retrieves the geographical coordinates for the harbours in the fleet register field
Folder Icon	Opens a file selection window to choose the RDS file of harbours' coordinates
Save Icon	Stores the harbours' coordinates as an RDS file

```
# Setup the harbours of registration
my_sampling$fleet$setRegHarbs()

# Compute average distance between fishing grounds and harbours
my_sampling$getHarbFgDist()

# Visualize the distance matrix
my_sampling$ggplotFgWeigDists()
```

4.5 Production

The landings dataset is employed to get an estimate of the spatial and temporal distribution of LPUEs (Landings Per Unit of Effort) in the area of study. The required input is a `list` of `data.frames` with vessel ID, Start and End Timestamp, Species (scientific name or acronym), and Quantity (Kg).

```
# Load the landings dataset
my_sampling$fleet$loadProduction(production_path = "path/2/production")

# Extract the vessel IDs from the landings dataset
my_sampling$fleet$setProdIds()

# Match the vesseld IDs between the landings and the effort dataset
my_sampling$fleet$setIdsEffoProd()

# Visualize the IDs counts on each dataset and the overlap
my_sampling$fleet$plotCountIDsEffoProd()

# Reshape the landings dataset from long to wide
my_sampling$fleet$setProdMatr()

# Aggregate the gridded effort into daily records
my_sampling$fleet$setDayEffoMatrGround()

# Join the landings and the daily effort
my_sampling$fleet$setEffoProdMatr()

# Reshape the daily effort*landings into monthly records
my_sampling$fleet$setEffoProdMont()

# Get the species IDs
my_sampling$fleet$setProdSpec()

# Reshape the monthly effort*landings list into a single data.frame
my_sampling$fleet$setEffoProdAll()

# Join the effort*landings data.frame with vessel' LOA
my_sampling$fleet$setEffoProdAllLoa()
```

For each species to be analyzed, the user must provide a threshold value for the minimum quantity to be landed to consider a catch to be intentional. The landing threshold is employed to train a Logit model

that classifies each Effort Pattern as 'Targeting' or 'Not-Targeting' the species. Then, the targeting patterns are used to estimate the rate of LPUEs in each time-space frame (i.e. fishing ground by month).

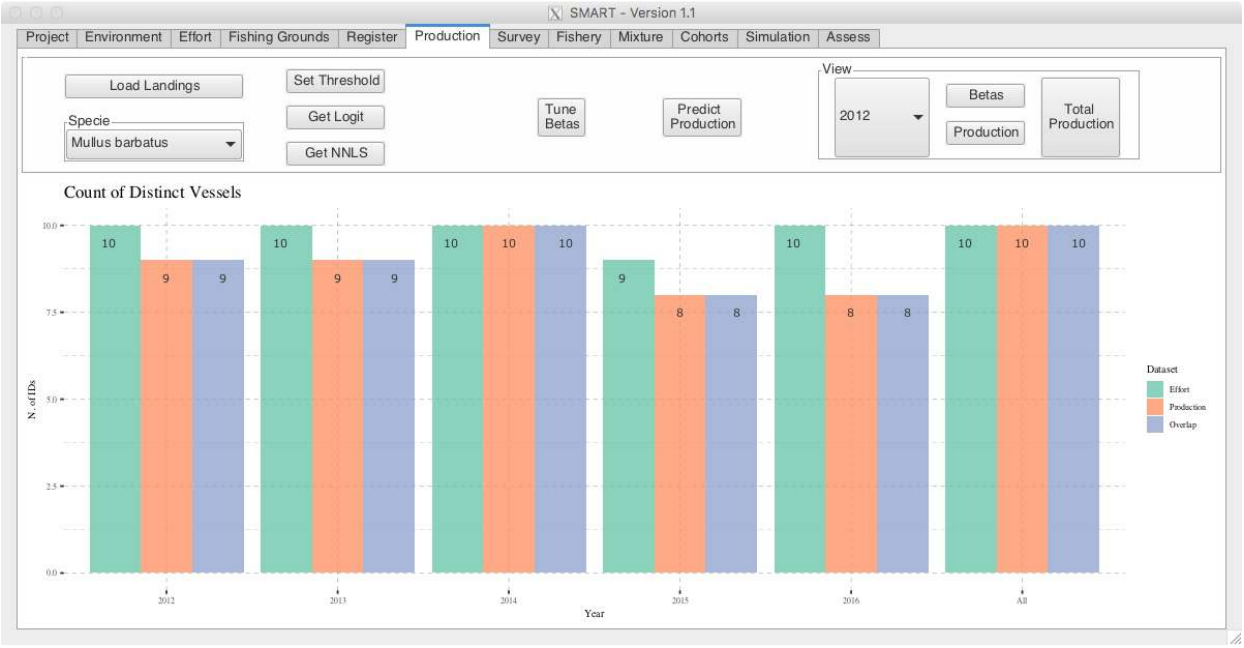

Figure 4.12: smartR GUI Fleet

The landed quantity of a single trip of each vessel is joined to the corresponding spatial fishing pattern prepared in the fleet dataset processing. The resulting structure is a `matrix` where each row contains the vessel identifier ($vessel_v$), timestamp ($time_t$), total hours of fishing in each fishing ground ($FT_{vtg^1}..FT_{vtg^n}$) and the landed quantity ($landedQty_{vt}$).

```
# Setup the Logit landing' threshold
my_sampling$fleet$setSpecSettItm(species = "specieName",
                                 thresh = 100,
                                 brea = 99,
                                 max_xlim = 1000)

# Train the Logit model
my_sampling$fleet$setSpecLogit(selspecies = "speciesName",
                               selModel = c("GLM", "CART", "RF", "NN")[1],
                               cp = 0.01,
                               cv = 2)

# Visualize the Logit ROC
```

Widget	Function
Load Landings	Opens a file selection window to choose a proper file of landings data
Species	Select the current species to analyse
View	
Set Threshold	Opens a new window with a GUI to select a weight threshold as input for the Logit
Get Logit	Opens a new window with a GUI to setup the input parameters for the Logit model
Get NNLS	Opens a new window to filter the results of the NNLS model
Tune Betas	Stores the regionalisation result as an RDS object
Predict Production	Predicts the landings of the current species
Year	Droplist to select the time frame of the statistical summary
Betas	Shows the spatial pattern of the beta values for the selected year
Production	Shows the spatial pattern of the production values for the selected year
Total Production	Shows the summary statistics of total production and beta values

```
my_sampling$fleet$plotLogitROC(selspecies = "speciesName")

# Setup the Logit confusion matrix
my_sampling$fleet$setSpecLogitConf(selspecies = "speciesName",
                                   cutoff = 0.73)
```

The landed quantity value is given by the sum of all landed quantities, across all the fishing grounds g of the $v - eth$ vessel within the $t - eth$ time frame. A discrimination threshold is set to distinguish the quantity of landed catch obtained with a purposeful targeting from not purposefully targeting of the studied species.

$$Targeting : T_{vts} = \begin{cases} 1 = target, & \text{if } landedQty_{vt} \geq threshold_s \\ 0 = notarget, & \text{otherwise} \end{cases}$$

Accordingly, the dataset is divided into two groups. The records of the vessel v during the time interval t with a landed quantity of the species s above the threshold are classified as targeting the considered species, while the records with a landed quantity below the

threshold are considered as by-catch. Three different predictors for the binary choice model are currently implemented: `GLM` (generalized linear model); `CART` (classification and regression tree); `RF` (random forest). All the three models go through the same phases of a standard estimation process of training, prediction, and tuning.

```
# Train the NNLS model
my_sampling$getNnlsModel(species = "specieName",
                         minobs = 10,
                         thr_r2 = 0.85)

# Visualize the NNLS observed/estimated scatterplot
my_sampling$fleet$plotNNLS(species = "specieName",
                           thresR2 = 0.85)

# Reshape the effort dataset into monthly records
my_sampling$fleet$setEffoMont()

# Convert the effort yearly list into a single data.frame
my_sampling$fleet$setEffoAll()

# Join the effort data.frame with the LOA
my_sampling$fleet$setEffoAllLoa()
```

The NNLS[6] model is implemented with the `nnls()` function of the `nnle` package (Mullen and van Stokkum 2012) which performs a least square regression with a positive coefficient constraint. The resulting coefficients of the NNLS regression are arranged in the matrix of $LPUE_s$ estimates for the species s on every time interval t in every fishing ground g. Finally, using the estimated $LPUE_s$ matrix as a 'Production Model', which is assumed to be constant within the considered time-frame, it is possible to predict the landed quantity of the set of fishing vessels with an unobserved landing data.

[6] "In mathematical optimization, the problem of non-negative least squares (NNLS) is a type of constrained least squares problem where the coefficients are not allowed to become negative" Wikipedia

```
# Compute the total production
my_sampling$predictProduction(species = "specieName")

# Setup the LPUEs plots
my_sampling$setPlotBetaMeltYear(species = "specieName",
                                year = "2014")

# Visualize the LPUEs on the map
print(myFishery$sampMap$ggBetaFGmap)

# Visualize the LPUEs boxplot
```

```
print(myFishery$sampMap$ggBetaFGbox)

# Setup the total production plots
my_sampling$setPlotProdMeltYear(species = "specieName",
                                year = "2014")

# Visualize the total production on the map
print(myFishery$sampMap$ggProdFGmap)

# Visualize the total production boxplot
print(myFishery$sampMap$ggProdFGbox)
```

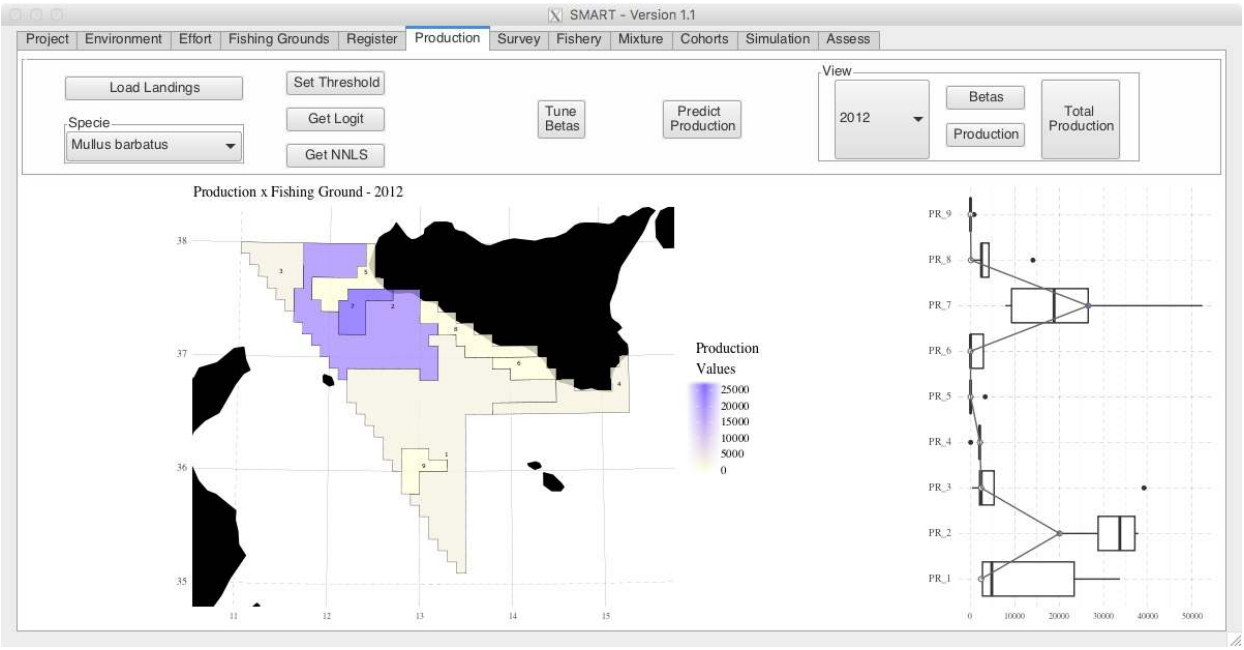

Figure 4.13: smartR GUI Fleet

4.6 Mixture & Cohorts

```
# Load the commercial fishery LFD sampling
my_sampling$loadFisheryLFD(csv_path = pathFishery)

# Match the sampling coordinates with the grid' cells
my_sampling$addFg2Fishery()

# Convert the aggregated numbers into single specimen
my_sampling$setSpreaFishery()

# Setup reporting statistics
my_sampling$setSpatFishery()

# Setup the visualization device
my_sampling$sampMap$set_ggMapFgFishery(rawSampCoo = myFishery$rawDataFishery)
```

Providing the maximum supposed number of components (ages) of the mixture, the routine implemented in `smartR` links, for each specimen, the length attribute to a corresponding age. The implemented growth model runs three parallel chains with a custom number of sampled specimens, a number of adaptation and sampling cycles according to the dimensions selected by the user. At the end of the simulation, the MCMC outputs are used to split the observed catches/landings by ages, to estimate the LFD characterizing every cohort in each fishing ground for each time frame.

```
# Setup the maximum age cohort
my_sampling$fisheryBySpecie[[i]]$setNCoho(num_coh = 5)

# Start the MCMC simulation
my_sampling$fisheryBySpecie[[i]]$calcMixDate(
  nAdap = 1000000,
  nSamp = 20000,
  nIter = 500000,
  sexDrop = "Female",
  curveSel = "von Bertalanffy")

# Visualize the results
my_sampling$fisheryBySpecie[[i]]$ggplotMcmcOut(
  selCompo = c("MCMC", "Key", "Birth")[1],
  selSex = c("Female", "Male", "Unsex")[1])
```

The depth-stratified data from a time series collected through a stan-

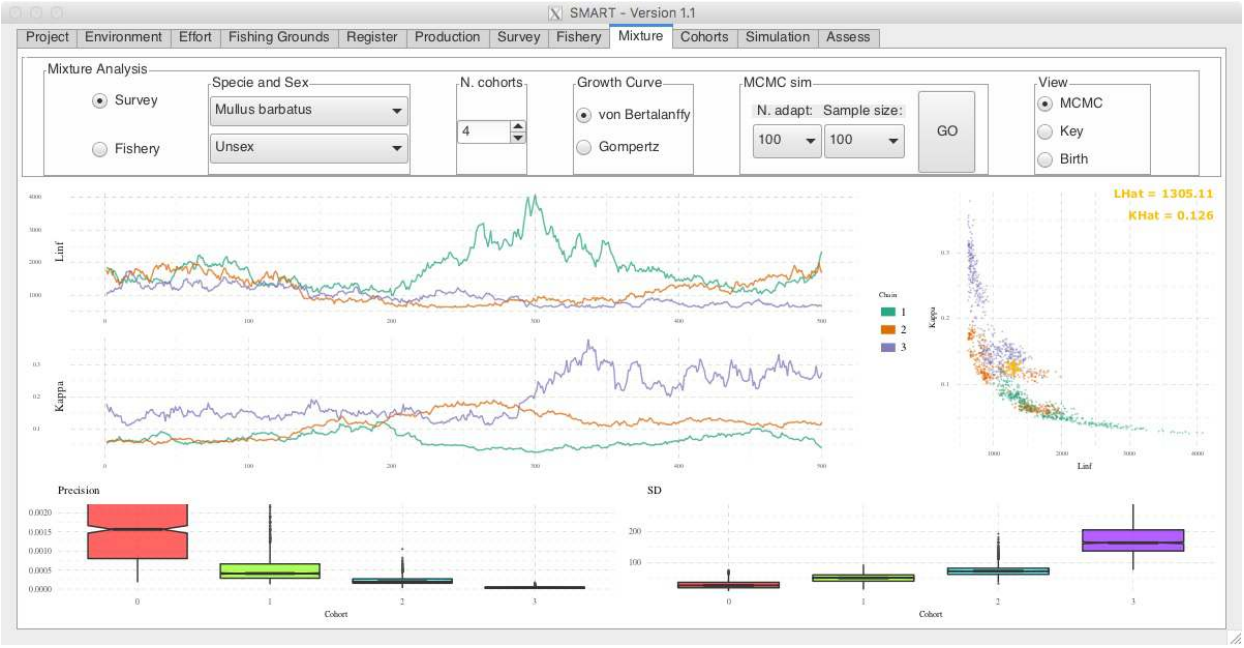

Figure 4.14: smartR GUI Fleet

dardized survey is also employed to assess the abundance index of each length class by the computation of the average number of individuals by length and depth stratum. The routine implemented in `smartR` follows the data format and algorithmic process developed for the 'MEDiterranean International bottom Trawl Survey' (MEDITS) campaigns and shared as the MEDITS protocol Bertrand et al. [2000][7].

[7] Bertrand et al. 2000. "An international bottom trawl survey in the Mediterranean" Actes de Colloques-IFREMER

```
# Load the scientific survey with LFD sampling
my_sampling$loadSurveyLFD(csv_path = pathSurvey)

# Same procedure as the fishery dataset
my_sampling$addFg2Survey()
my_sampling$setSpreaSurvey()
my_sampling$setSpatSurvey()

# Setup of the depth-stratified biomass index
my_sampling$setDepthSurvey()

# Define the Depth Strata
strataVec <- c(0, 10, 50, 100, 200, 500, 800, Inf)

# Compute the total area of the studied area
my_sampling$sampMap$setAreaGrid()
```

Widget	Function
Mixture Analysis	
Radio Buttons	Select the input data between the survey or the fishery dataset
Species & Sex	
Species	Droplist to select the species to analyse in the chosen dataset
Sex	Droplist to select the sex to analyse for chosen species/dataset
N. cohorts	
Slider	Select the maximum number of cohort to test in the mixture analysis
Growth Curve	
Radio Buttons	Select the growth curve to employ between 'von Bertalanffy' and 'Gompertz'
MCMC sim	
N. Adapt	Number of adaptation steps in the MCMC simulation
Sample Size	Number of samples to employ in the MCMC simulation
GO	Button to start the MCMC simulation
View	
Radio Buttons	Show the main graphical output between the MCMC diagnostics, Age/Length key and the Birth graph

```
# Compute the area of each depth stratum
my_sampling$sampMap$setAreaStrata(vectorStrata = strataVec)

# The proportion of each strata in the area of study
my_sampling$sampMap$setWeightStrata()

# Compute the relative abundances of specimen in each stratum
my_sampling$setStratumSurvey(vectorStrata = strataVec)

# Compute the average abundances
my_sampling$setAbuAvgAll()

# Expand the average abundances to the total area
my_sampling$setStrataAbu()

# Aggregate the results to get the MEDITS index
```

```
my_sampling$setMeditsIndex()
```

Lastly, a length/weight relationship, like Cren [1951][8] is employed to convert the length observation into weight estimates to get the biomass estimates Froese [2006][9] required in subsequent phases. The classical equation defines the exponential relation between the weight (W) and the length (L) of fishes:

$$W = \alpha L^{\beta} \tag{4.1}$$

Within the smartR package, the values for the α and β parameters can be either placed manually, e.g. through literature review or FishBase lookup (Froese and Pauly [2017][10]), or imputed with non-linear least-squares regression. In the latter case, having a suitable dataset of length-weight observation pairs, it is possible to estimate the parameters of the non-linear model applying the default Gauss-Newton algorithm implemented in the nls() function of the stats package.

[8] Cren 1951. "The Length-Weight Relationship and Seasonal Cycle in Gonad Weight and Condition in the Perch (Perca fluviatilis)" The Journal of Animal Ecology

[9] Froese 2006. "Cube law, condition factor and weight-length relationships: history, meta-analysis and recommendations" Journal of Applied Ichthyology

[10] Froese & Pauly 2017. "Fishbase" Technical Report

```
# Load a dataset with Length and Weight measures
lw_data <- read.csv(pathLWrel)

# Perform the nonlinear least-square estimation of the Length Weight Relationship
lw_fit <- nls(Weight ~ I(alpha * Length ^ beta),
              data = lw_data[,c("Length", "Weight")],
              start = list(alpha = 1, beta = 1))

# Extract the alpha and beta parameters
alpha <- round(summary(lw_fit)$coefficients[1,1], 5)
beta <- round(summary(lw_fit)$coefficients[2,1], 5)

# Load the alpha and beta parameters into the smartR project
my_sampling$fisheryBySpecie[[1]]$setLWpar(alphaVal = alpha,
                                          betaVal = beta,
                                          sex = "Female")

# Compute the weight of all the length-measured specimens
my_sampling$fisheryBySpecie[[1]]$setWeight(sexVal = "Unsex")
```

4.7 Simulation

> "The economic performance of the fleet results from the balance between total cost and revenues of every vessel actively involved in the fishery"

THE FIRST STEP is the computation of a set of three economic indicators (spatial index, number of days at sea and production index).

THE SECOND STEP is the estimate of the costs relative to each one of the three indicators. The third step is the computation of the revenues from the total landed quantity for each species and, lastly, the subtraction of the costs from the revenue to get the net gains.

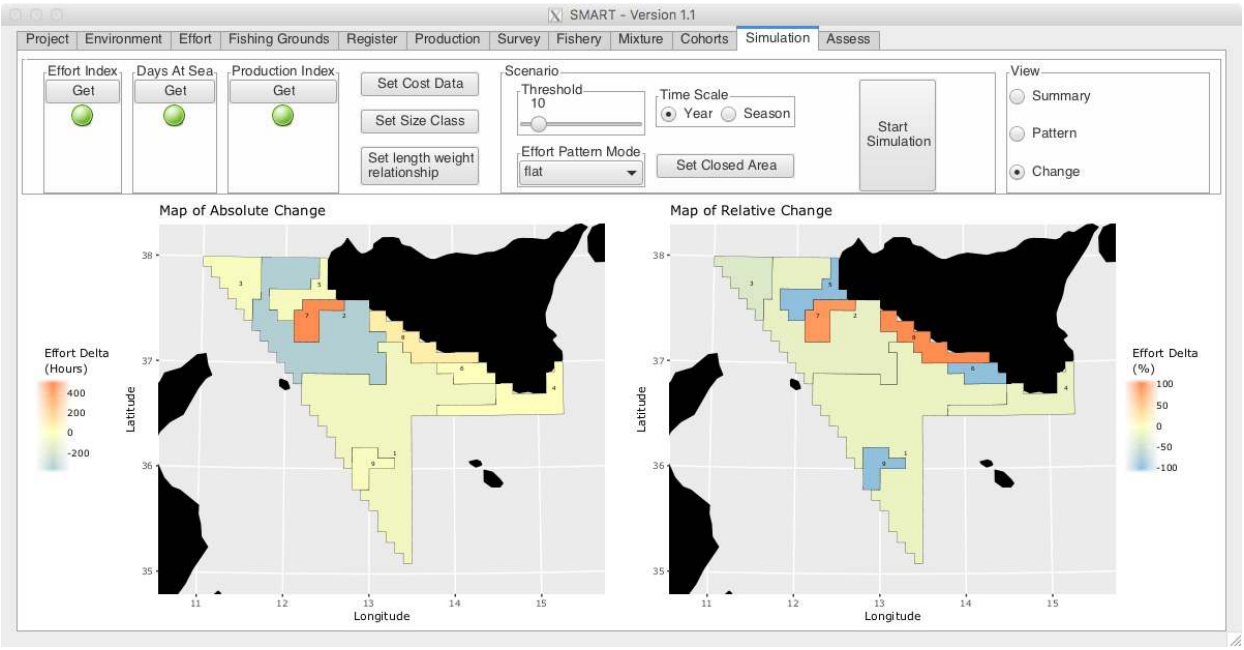

Figure 4.15: smartR GUI

THE SPATIAL INDEX represents a measure of the variable costs associated with the spatial choices of the fisher. It is computed as the weighted arithmetic mean of the cumulative yearly spatial fishing pattern in hours (CFT_g^v the cumulative yearly fishing times $FT_{vtg^1}..FT_{vtg^n}$ of the $v-eth$ vessel), weighted by the average distance of the harbour from the corresponding fishing grounds (harbour average distance w_g).

$$spatialIndex^v = \frac{\sum_{g=1}^{G} w_g CFT_g^v}{\sum_{g=1}^{G} w_g} \tag{4.2}$$

Widget	Function
Effort Index	
Set	Start the computation of the values for the Effort Index
Days At Sea	
Set	Start the computation of the Days At Sea
Production Index	
Set	Start the computation of the values for the Production Index
Set Cost Data	Opens a new GUI window to load the economic data and setup the regression models for the economic performance
Set Size Class	Opens a new GUI window to setup the size/price class for each species
Set LW relationship	Opens a new GUI window to compute the length/weight relationship for each species
Scenario	
Threshold	Slider to setup the optimization threshold for the scenario simulation
Time Scale	Radio button to select the time scale of the scenario simulation between yearly or seasonal
Effort Pattern Mode	Set the optimization modality
Set Closed Area	Opens a new GUI windows to select the closed areas for the spatial restiction in the scenario simulation
Start Simulation	Button to begin the scenario simulation
View	
Radio Buttons	Show the output of the simulation

The Days at Sea index is linked to the amount of time spent by a fishing vessel (and its crew) at sea, independent of the fishing location. It indicates the raw value of the number of days per month where, for each vessel, the tracking device has recorded at least one position outside the harbour.

$$daysAtSea^{v} = \sum_{d=1}^{D} Out_{d}^{v} \tag{4.3}$$

$$At\ Sea\ Status : Out_{vd} = \begin{cases} 1 = at\ sea \\ 0 = in\ harbour \end{cases}$$

where the '*at sea*' status is obtained when at least one ping of the

$v - eth$ vessel, in the $d - eth$ day, is classified outside the harbour' buffer area by the standard vmsbase processing.

The production index acts as a proxy for the variable costs tied to the landed quantity of fish (as taxes or commercialization costs). It is obtained by summing the landed quantity, of every single species separately, by year for each vessel independently of the fishing ground. The total yearly production index of the v - eth vessel is given by the equation:

$$productionIndex^v = \sum_{t=1}^{T} \sum_{s=1}^{S} L_{vst} \tag{4.4}$$

```
my_sampling$setEffortIndex()
print(ggplot_effoIndBoxplot(df_EffoInde = myFishery$fleet$effortIndex))
my_sampling$setDaysAtSea()
print(ggplot_seaDaysBoxplot(df_seaDays = myFishery$fleet$daysAtSea))
my_sampling$setProductionIndex()
print(ggplot_prodIndBoxplot(df_ProdInde = myFishery$fleet$prodIndex))
```

The cost data is employed, within three sets of linear regressions (spatial-, effort- and production-based), to assess the financial investment of each vessel.

The spatial-based regression aims at predicting the **costs** depending on the location choice relative to the spatial index. The linear regression is performed with $spatialIndex^v$ and LOA^v of the $v - eth$ vessel respectively from the previously computed index and the fleet register data as explanatory variables, β_1^{sc} and β_2^{sc} as the regression coefficients for the $spatialIndex$ and LOA respectively. The spatial cost regression of the v - eth vessel is given by the equation:

$$spatialCost^v = \beta_1^{sc} spatialIndex^v + \beta_2^{sc} LOA^v \tag{4.5}$$

The effort-based regression delivers the fixed costs relative to the fishing activity independently of the location choice. The regression is performed with $daysAtSea^v$, LOA^v and Kw^v of the $v - eth$ vessel respectively from the $daysAtSea\ Index$ and the fleet register, β_1^{ec}, β_2^{ec} and β_3^{ec} the regression coefficients for the independent variables $daysAtSea$, LOA and Kw. The effort cost regression of the v - eth vessel is given by the equation:

$$effortCost^v = \beta_1^{ec} daysAtSea^v + \beta_2^{ec} LOA^v + \beta_3^{ec} Kw^v \tag{4.6}$$

The production-based regression relates directly the production costs to the production index. The regression is performed with $productionIndex^v$ of the $v - eth$ vessel as explanatory variables and β_1^{pc} as the regression coefficient. The production cost regression of the v - eth vessel is given by the equation:

$$productionCost^v = \beta_1^{pc} productionIndex^v \tag{4.7}$$

```
# Import the raw costs dataset
my_sampling$fleet$loadRawEconomy(economic_path = pathCosts)

# Match the Effort Pattern with the economic costs
my_sampling$fleet$setYearEconomy()

# Setup the three cost models
my_sampling$setCostInput()

# Get the coefficients for each regession
my_sampling$fleet$getCostOutput()

# Setup the graphical decice
my_sampling$fleet$setCostPlot()

# Visualize the regressions
print(myFishery$fleet$plotSpatialReg)
print(myFishery$fleet$plotEffortReg)
print(myFishery$fleet$plotProductionReg)
```

Starting from the weight at length relationship computed in the Mixture module, a reference table is built to collect, for each weight class in each fishing ground, four main measures: the average length, the standard deviation and the relative and absolute specimen abundances. Then, given the production pattern, the estimated revenues are obtained, from the landed quantity of a vessel in the specific month and fishing ground, computing the following algorithmic procedure for each vessel in each month in each fishing ground:

1. take the average length and absolute frequency for each weight class
2. spread the monthly production across all weight classes proportionally to their absolute frequencies
3. aggregate the weight spread according to the size/price classes
4. multiply the total weight of each size class by the price of each class
5. sum the gains for each class

```
# Standard format for the size/price information
sizePrice <- data.frame(Class = c("small", "medium", "large"),
                        Units = c("Length", "Length", "Length"),
                        LowerBound = c(1, 5, 10),
                        UpperBound = c(5, 10, 15),
                        Price = c(2, 5, 10),
                        stringsAsFactors = FALSE, row.names = NULL)

# Load the data into the smartR project
my_sampling$fleet$setEcoPrice(sel_specie = "specieName",
                              price_df = sizePrice)
```

The monthly revenues by fishing ground are stored after being summed at the yearly scale to get a total revenue value for each vessel for each year. Finally, the gains pattern is easily generated, for each month t in each fishing ground g for every fishing unit v, as the resulting amount after the arithmetic subtraction of the cost to the revenues at the individual level.

```
# Setup the demographic Length Weigth distribution
my_sampling$getLWstat()

# Simulate the selected management scenario
my_sampling$genSimEffo()

# Compute the final total production of the optimized system
my_sampling$simProdAll()

# Get the final total costs
my_sampling$getSimTotalCost()

# Get the final total revenues
my_sampling$getSimRevenue(timeScale = "Year")

# Get the final difference betwen costs and revenues
my_sampling$getCostRevenue()
```

All the previously computed outputs are an indispensable input of the simulation phase. Thus, the required parameters are received directly from the previous computations. The main requirements already available are:

- the topology of the area of study and the fishing grounds configuration;

- the recent observed spatial effort pattern (last year of the time-series);
- the target choice model;
- the production model;
- the economic models (to compute the costs and revenues of each fishing pattern);
- the growth model (resource' growth and spatial distribution);

while the last needed component is the management strategy to experiment.

Two main strategies are currently available to gauge the corresponding effects on resource abundance. In the first scenario, named Optimized Status Quo (OSQ), the simulator works without spatial constraints, acting solely as a gain optimizer. In the second scenario, named Area Ban (AB), the spatial constraints are implemented setting the fishing effort to zero in one or more fishing grounds selected by the user. Once all the inputs are specified, the simulator operates stochastically on the observed pattern, at the single vessel level, seeking the maximization of the fishing activity gains. The operations performed, at each iteration for each fishing vessel, are:

1. Generate a new monthly effort pattern for each vessel;
2. Compute the costs and revenues of the new pattern;
3. Evaluate the new gains against the gains of the previous iteration;
4. Keep the patterns with improved performance.

The monthly effort patterns generated by the simulator are constructed, at the individual level, for every single vessel. The simulated patterns are obtained starting from the observed probability distribution of fishing effort deployed in each fishing ground. The new patterns are created maintaining the observed proportion of effort allocation while keeping the total amount (of cumulative annual fishing effort computed in the previous year) fixed. Since it is unreasonable that the total monthly fishing effort for each fishing ground exceed an ideal threshold (the crowd of many fishing vessels is likely to produce density effects), the probability distribution used to generate the candidate effort pattern is also inversely dependent from the cumulative effort for each fishing ground.

4.8 Assessment

The Stock Assessment procedure implemented in smartR, is a MICE[11] model. This framework models a simple Statistical Catch At Age (SCAA) with a basic population dynamic which follows the classical approach of Doubleday [1976][12] where the catch-at-age datasets are fitted for multiple cohorts simultaneously and the fishing mortality is split into age and year components.

[11] Punt et al. 2016. "Exploring the Implications of the Harvest Control Rule for Pacific Sardine, Accounting for Predator Dynamics" Ecological Modelling

[12] Doubleday 1976. "A Least Squares Approach to Analyzing Catch at Age Data" Int. Comm. Northwest Atl. Fish. Res. Bull

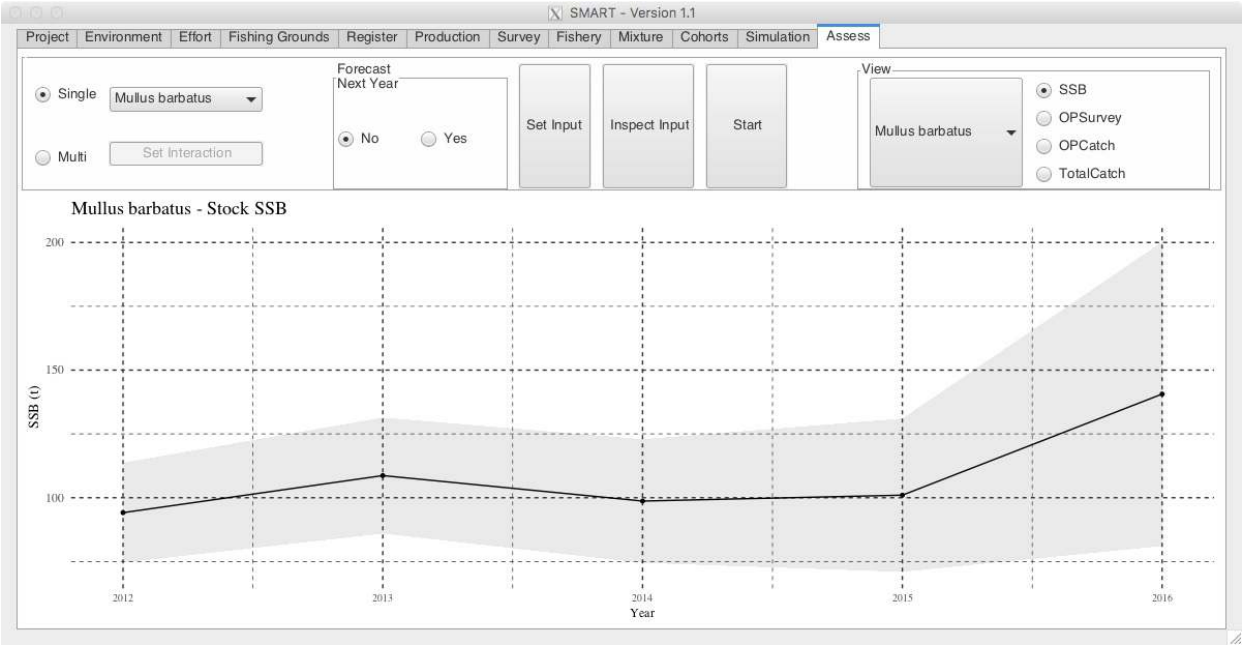

Figure 4.16: smartR GUI Fleet

```
# Setup initial parameters
my_sampling$setAssessData(species = "Species name", forecast = TRUE)

# Inspect the initial parameters
my_sampling$assessData$`Species name`
```

Defined the total mortality Z of the age group a during year y as:

$$Z_{ya} = M_a + S_a F_y \tag{4.8}$$

where M_a is the natural mortality rate at age a, S_a is the fishery selectivity at age a and F_y is the fishing mortality of the year y. The age-structured population dynamic is designed with a forward projec-

Widget	Function
Single	
Species	Select a species to analyse with a single species approach
Multi	
Set Interaction	Opens a new GUI window to setup the interaction between species with a multi-species approach
Forecast	
Set Input	Button to setup the initial parameters for the stock assessment
Inspect Input	Opens a new GUI window to show the input parameter computed and to be used in the stock assessment
Start	Begin the optimization
View	
Droplist	Show the results of the stock assessment

tion method, and it is modelled as:

$$N_{ya} = \begin{cases} R_0 e^{\varepsilon_y}, & \text{if } a = 0 \\ N_{y-1a-1} e^{-Z_{y-1a-1}}, & \text{if } 1 \leq a \leq x \\ N_{y-1x-1} e^{-Z_{y-1x-1}} + N_{y-1x} e^{-Z_{y-1x}}, & \text{if } a = x \end{cases}$$

where N_{ya} is the number of individual of age a in the year y, R_0 is the median recruitment with a yearly deviation of e^{ε_y} and x as the maximum age class. The catch at age in numbers for each year C_{ya} is estimated as:

$$C_{ya} = \frac{S_a F_y}{Z_{ya}} N_{ya}(1 - e^{-Z_{ya}}) \tag{4.9}$$

```
# Start the optimization of a multispecies assessment
my_sampling$assMulti()

# Configure the output devices
my_sampling$setPlotMulti()
```

The catch-at-age datasets are assumed to be multinomially distributed, while the survey estimates of abundance by age-class are assumed to be log-normally distributed with a standard error of the log that is independent of age and year. The spawning biomass is

calculated in the middle of the year as modeled by the expression:

$$SSB_y = \sum_{a=0}^{x} w_a m_a N_{ya} e^{-0.5 Z_{ya}} \tag{4.10}$$

The main output of the stock assessment procedure integrated into the `smartR` package is represented by the estimated SSB, resulting from the analysis of the recorded time-series of landings and scientific surveys. After the simulation of alternative management scenarios, the simulated landings are added at the end of the real time-series, as a new year of activity of the fishing fleet on the studied stock. The SSB values from the OSQ and AB scenario are then evaluated against each other.

```
# Perform a forward projection with the estimated parameters
forwPop(Pars = my_sampling$assMultiRes$par,
        SpeciesData = my_sampling$assessData,
        Nspecies = length(my_sampling$assessData),
        PredationPars = my_sampling$assessInteract,
        Nproj = 10, Nsim = 50, SigmaR = 0.5)
```

5
FAQs

QUESTION: Could a future iteration of this model include simulation of changing temperatures/climate velocities (i.e., a simulation species distribution shifts)?

> *Yes, it is already possible to include other environmental dynamics in the standard format. While it is possible, it would require some customization of the algorithms*

Q: Except the grid topology, bathymetry, and seabed categories, is it possible to load other important parameters of bottom temperature and salinity?

> *Yes, as for some custom simulation strategies, it is already possible to include other environmental variables in the standard format. For different data structures, it is necessary to customize some algorithms but it is still feasible*

Q: Is it possible to read and use an existing grid containing the estimated fishing effort values?

Yes, it is possible to use an existing grid if the estimated fishing effort values are still disaggregated at the individual level

---

Q: Why the need to discriminate between the landed catch (landings) obtained with a "purposeful" targeting from the "not purposeful"?

The LOGIT model discriminates between the targeted landings ("purposeful") from the accessory/unwanted ("not purposeful targeted") landings. Once these two subgroups are identified, it is possible to estimate the LPUE rates with the NNLS model. This is required for complex fishery systems in particular when the landings are characterized by a mixed pool of species or if the studied fleet is engaged in several different metiers

---

Q: Where do the data to construct the length-frequency for the landings come from?

The data required to construct the length-frequency distributions of the resources at sea is collected within scientific surveys and commercial samplings where a subset of the total catch is measured (length and weight). This dataset is usually independent of the landings dataset.

6
Links

AIS https://en.wikipedia.org/wiki/Automatic_identification_system

GUI https://en.wikipedia.org/wiki/Graphical_user_interface

LOA https://en.wikipedia.org/wiki/Length_overall

NNLS https://en.wikipedia.org/wiki/Non-negative_least_squares

Shapefile https://en.wikipedia.org/wiki/Shapefile

VMS https://en.wikipedia.org/wiki/Vessel_monitoring_system

7
Bibliography

C. Amante and B. W. Eakins. ETOPO1 1 Arc-Minute Global Relief Model: Procedures, Data Sources and Analysis, 2009. URL `https://doi.org/10.7289/V5C8276M`.

J. A. Bertrand, L. Gil De Sola, Costas Papaconstantinou, G. Relini, and Arnauld Souplet. An international bottom trawl survey in the Mediterranean: the MEDITS programme. *Actes de Colloques-IFREMER*, pages 76–96, 2000.

Winston Chang. *R6: Encapsulated Classes with Reference Semantics*, 2019. URL `https://CRAN.R-project.org/package=R6`. R package version 2.4.1.

E. D. Le Cren. The Length-Weight Relationship and Seasonal Cycle in Gonad Weight and Condition in the Perch (Perca fluviatilis). *The Journal of Animal Ecology*, 1951. DOI: 10.2307/1540.

Cynthia Davies, Dorian Moss, and Mark Hill. EUNIS Habitat Classification - Revised. Technical report, 2004. URL `http://www.eea.europa.eu/themes/biodiversity/eunis/ds_resolveuid/6RGHTVEOZK`.

WG Doubleday. A least squares approach to analyzing catch at age data. *Int. Comm. Northwest Atl. Fish. Res. Bull*, 12:69–81, 1976.

R. Froese. Cube law, condition factor and weight-length relationships: history, meta-analysis and recommendations. *Journal of Applied Ichthyology*, 2006. DOI: 10.1111/j.1439-0426.2006.00805.x.

R. Froese and D. Pauly. Fishbase. Technical report, 2017. URL `www.fishbase.org`.

T. Russo, A. Parisi, G. Garofalo, M. Gristina, S. Cataudella, and F. Fiorentino. SMART: A Spatially Explicit Bio-Economic Model for Assessing and Managing Demersal Fisheries, with an Application

to Italian Trawlers in the Strait of Sicily. *PLOS ONE*, (1), 2014. DOI: 10.1371/journal.pone.0086222. URL http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0086222.

T. Russo, E. B. Morello, A. Parisi, G. Scarcella, S Angelini, L. Labanchi, M. Martinelli, L. D'Andrea, A Santojanni, E. Arneri, and S. Cataudella. A model combining landings and VMS data to estimate landings by fishing ground and harbor. *Fisheries Research*, March 2018. DOI: 10.1016/j.fishres.2017.11.002. URL http://www.sciencedirect.com/science/article/pii/S0165783617303107.

T. Russo, L. D'Andrea, S. Franceschini, P. Accadia, A. Cucco, G. Garofalo, M. Gristina, A. Parisi, G. Quattrocchi, R. F. Sabatella, M. Sinerchia, D. M. Canu, S. Cataudella, and F. Fiorentino. Simulating the Effects of Alternative Management Measures of Trawl Fisheries in the Central Mediterranean Sea: Application of a Multi-Species Bio-economic Modeling Approach. *Frontiers in Marine Science*, 2019. DOI: 10.3389/fmars.2019.00542. URL https://www.frontiersin.org/articles/10.3389/fmars.2019.00542/full.

John Verzani. *gWidgets: gWidgets API for Building Toolkit-Independent, Interactive GUIs*, 2019. URL https://CRAN.R-project.org/package=gWidgets. R package version 0.0-54.1.