Call MARE/2020/08

Agreement reference: SI2.839444
European Maritime and Fisheries Fund (EMFF)

Development of the Regional Database for the Mediterranean & Black Seas

Work-package 4 - Deliverable 4.4.1

R package containing the new data processing tools with pertinent documentation

I. Bitetto, W.Zupa

Partners involved:

COISPA, HCMR

Table of Contents

Acronyms

CRAN	Comprehensive R Archive Network
DCRF	Data Collection Reference Framework
GFCM	General Fisheries Commission for the Mediterranean
MED&BS	Mediterranean and Black Sea
RCG	Regional Coordination Group

Executive summary

The availability of common, freely accessible and agreed data processing tools is of paramount importance to standardize the data preparation for the different data calls at regional level, starting from a common format.

Auxiliary tools to standardize and ease the procedures for data processing and management weredeveloped in STREAM[1] project (MARE/2016/22 – SI2.770115) through ad hoc scripts for the conversion of primary data into the formats for data transmission to specific data calls (e.g. MED&BS, FDI, GFCM DCRF) using existing tools (e.g. COST) for data analyses.

The aim of Task 4.4 - Developing data processing tools – was to consolidate these existing tools and to include them in an R package to be easily integrated in RDBFIS as well as used as a standalone application to produce the necessary tables in the different data calls formats.

This task developed an open-access (available on GitHub repository) and well documented tools for fishery data processing, data management and conversion into different formats required by several end-users to dramatically reduce the risk of data failure of the MS and CPs in the data submission to the three main data calls present in Mediterranean and Black Sea: MED&BS, FDI, GFCM and DCRF.

As these tools have been implemented in R language, they can be easily adapted to future new formats and generalized to be fed by any type of input format. Moreover, these tools have been incorporated in RDBFIS, to allow the production of the aggregated data in the format required by the specific datacalls, starting from the same dataset, thus ensuring a higher consistency of the data required by the different datacalls.

The storing on the GitHub repository and the development in R language will facilitate this code adaptation and improvement, allowing the collaboration among the experts involved in the data calls.

This is also expected to contribute to ease the work of data processing at regional level, improving the data quality.

[1] *https://datacollection.jrc.ec.europa.eu/documents/10213/1239605/2019-12_16th_Liaison_Meeting.pdf/c59331f6-3047-4f25-a0b0-89945475a174?version=1.2*

1. Introduction

The availability of common, freely accessible and agreed data processing tools is of paramount importance to standardize the data preparation for the different data calls at regional level, starting from a common format.

Auxiliary tools to standardize and ease the procedures for data processing and management weredeveloped in STREAM[2] project(MARE/2016/22 – SI2.770115) through *ad hoc* scripts for the conversion of primary data into the formats for data transmission to specific data calls (e.g. MED&BS, FDI, GFCM DCRF) using existing tools (e.g. COST) for data analyses.

The aim of Task 4.4 - Developing data processing tools – was to consolidate these existing tools and to include them in an R package to be easily integrated in RDBFIS as well as used as a standalone application to produce the necessary tables in the different data calls formats.

This is also expected to contribute to ease the work of data processing at regional level, improving the data quality.

[2] *https://datacollection.jrc.ec.europa.eu/documents/10213/1239605/2019-12_16th_Liaison_Meeting.pdf/c59331f6-3047-4f25-a0b0-89945475a174?version=1.2*

2. Working method

In Task 4.4 - Developing data processing tools - new functionalities, through R scripts, have been developed to support users to check and prepare input data to be ready for the upload process in the database, focusing on the correctness of the data formats and the presence of the expected values for each field and type of table.

This task took advantage from the scripts developed in STREAM project that produce the fishery biological aggregated data in the formats required by the three relevant data calls of Mediterranean and Black Sea: DGMARE MED&BS, FDI and GFCM DCRF. Indeed, STREAM project developed a set of R auxiliary scripts for the conversion of dataset into the relevant formats for the data transmission and specifically:

- from RCG Med&BS Data Call format to the SDEF (COST)
- from the SDEF (COST) format to the DG MARE MED&BS Data Call
- from the SDEF (COST) format to the GFCM/DCRF Data Call
- from the SDEF (COST) format into DG MARE FDI Data Call format (using DG MARE Med&BS Data Call format).

The aim of task 4.4 has been to update the relevant scripts according to the last specifications of DGMARE MED&BS and FDI datacalls (*https://datacollection.jrc.ec.europa.eu/dc/medbs* and *https://datacollection.jrc.ec.europa.eu/dc/fdi*).

The latest version of the package can be found on a publicly accessible GitHub repository: *https://github.com/COISPA/RDBprocessing*.

All pertinent documentation for the package is included as interactive help and can be found in Annex II.

2.1 MED&BS datacall

As a first step, the conversion scripts from the RCG CS and CL tables to the COST tables have been transformed in R functions, in order to be easily recalled within the MED&BS processing functions.

The first function, **RCGtoCOST_CS**, allows to convert the RCG data call format into the SDEF format as a CS COST object (saved both as TR-HH-SL-HL-CA .csv tables and an R object).

The function allows the population of the following variables in the CS (Commercial Sampling) tables:

- Table TR (trip): Harbour
- Table HH (Fishing station): Fishing duration, latitude and longitude for the fishing operation, main fishing depth and main water depth.
- Table SL (Species List): Sex, commercial category scale
- Table HL (length): Sex, commercial category scale
- Table CA (sex- maturity- age- weight- length): if data to the individual level are provided, the CA fields “commercial size category scale”, “single fish number”, “age”, “ageing method”, “sex”, “individual weight”, “maturity stage”, “maturity scale”, “maturity scaling method” are also populated. In case aggregated data are provided, the script allows the CA table to be populated through the SL (Species List) and the HL (Length) tables for all aforementioned variables, excluding “individual weight” (as it cannot be provided in the aggregated data).

The function includes also integrity checks on the tables, producing warnings in order to perform any necessary corrections.

The second function, **RCGtoCOST_CL**, allows to convert the RCG CL table in the CL COST object.
Those functions can be used as starting point for the implementation of functions allowing the conversion of dataset into RDBES format, although most of the information related to the sampling (design, scheme, stratification) is not codified and then should be compiled by the person in charge of the sampling.
For further details on the functions see **ANNEX XVI RDBprocessing_0.0.1** and APPENDIX - R code of the functions.

2.1.1 Landing table - LAND_MEDBS

The script developed in STREAM project has been transformed in a function and adapted in order to receive as **input** directly the **RCG CS and CL tables** (not the COST objects); the conversion in COST objects, indeed, has been included in the function.

Moreover, additional tables that were provided as csv files in the STREAM script have been derived directly from the detailed data (e.g. the list of the species to be included in the LANDING table) and other tables containing codes (e.g. communicationTable_for_fishery.csv including metier codifications) have been included in the data folder of the package.

The default stratification implemented in the function is the area-quarter-metier; however, additional options could be easily included in the code.

The *RaiseLgth*() function of the COST library are used, with analytical method as default. The function estimates the total weight by each combination of the spatial, temporal and technical strata, and also the numbers at each length class for each combination of the strata.

Conversion of the technical field "metier" into the field GEAR_FISHERY_MESH_SIZE_RANGE according to the codification in the Data Call specifications is also made.

The **output** data frame is in accordance with the template of DG MARE Med&BS datacall.

The function works by area and species.

For further details on the scripts see **ANNEX XVI RDBprocessing_0.0.1** and APPENDIX - R code of the functions.

2.1.2 Discard table - DISC_MEDBS

The script developed in STREAM project has been transformed in a function and adapted in order to receive as **input** directly the **RCG CS and CL tables** (not the COST objects); the conversion in COST objects, indeed, has been included in the function. In this case the effort table (**CE**) in the COST format is also required to derive the discard volume and raised discard length-frequency distributions.

Also in this case, the default stratification implemented in the function is the area-quarter-metier; however, additional options could be easily included in the code.

Moreover, additional tables required by the STREAM scripts were included in the package also for this function.

The *totVolume*() function of the COST package is used, implementing the raising method by trip as default; however, additional options could be easily included in the code. The function estimates the total weight by each combination of the spatial, temporal and technical strata, and also the numbers at each length class for each combination of the strata.

The output data frame is in accordance with the DG MARE Med&BS Data Call template.

The function works by area and species.

For further details on the scripts see **ANNEX XVI RDBprocessing_0.0.1** and APPENDIX - R code of the functions.

2.1.3 Catch table - CATCH_MEDBS

The script developed in STREAM project has been transformed in a function and adapted in order to receive as **input** directly the **RCG CS and CL tables** (not the COST objects); the conversion in COST objects, indeed, has been included in the function. In this case it is also required the effort table (**CE**) in the COST format to derive the discard volume and raised discard length-frequency distributions.

Also for Landings and Discards tables the default stratification implemented in the function is the area-quarter-metier; additional tables required by the STREAM scripts were included in the package also for this function.

The ALK type is set as "fixed" by default (otoliths collection stratified by length class); however, additional options could be easily included in the code. The method used for the weight/length at age estimation for Landings/Discard was set as “analytical” by default”; also in this case, generalizations are easy to be made. The adjustment settings are all set as TRUE, to include the information of HL tables in the weight at age and length at age for discard and landing.

The *RaiseLgth()* function of the COST library is used, with analytical method as default, as it was done in the case of Landings table. Subsequently, the *RaiseAge()* function is used to transform the numbers at length to numbers at age for Landings.

The *totVolume()* function of the COST package is used, implementing the raising method by trip as default. Subsequently, the *RaiseAge()* function is used to transform the numbers at length to numbers at age for Discards.

For estimating mean length at age for discards and for landings, the *bpEstim()* function of the COST library is used, with “analytical” method as default. Similarly, the weight at age is estimated for both the Landings and the Discards.

The output data frame is in accordance with the DG MARE Med&BS Data Call template.

The function works by area and species.

For further details on the scripts see **ANNEX XVI RDBprocessing_0.0.1** and APPENDIX - R code of the functions.

2.1.4 ALK table - ALK_MEDBS

The script developed in STREAM project has been transformed in a function and adapted in order to receive as **input** directly the **RCG CS table** (not the COST objects); the conversion in COST objects, indeed, has been included in the function.

Moreover, additional tables that were provided as csv files in the STREAM script have been derived directly from the detailed data (e.g. the list of the species to be included in the LANDING table).

The *alkLgthRec()* function of the COST library is used, applying the ALK method “stepIncr” as default; however, additional options could be easily included in the code. The function estimates the number of samples (age readings) used in the analysis from the detailed data.

The CV estimation is based on the length-at-age estimation, using the "analytical" method as default (also this setting can be generalized).

The value for the field "APPLY_TO_CATCHES_FILE" of the table ALK is set as "Y" by default; a consolidated COST object is exported for the selected species, area, sex and time-period updated in accordance with the "stepIncr" method for filling in the gaps of the ALK table.

The output data frame is in accordance with the template of DG MARE Med&BS Data Call .

For further details on the scripts see **ANNEX XVI RDBprocessing_0.0.1** and APPENDIX - R code of the functions.

2.1.5 ML, MA, SRL, SRA tables: ML_MEDBS, MA_MEDBS, SRL_MEDBS, SRA_MEDBS

ML, MA, SRL and SRA tables functions have been completely re-written in order to disentangle from the COST functions and make them, hence, linked directly to the detailed data in RCG CS format. The compilation of the MED&BS tables is made basically through synthetic tables of the detailed data.

For the four tables the main **input** is the **RCG CS table**. For MA and ML tables the stages to be considered as immature should be also indicated. The default values are "1","2","2a". In this case the METHOD_USED is set as default "Observed proportions (immatures 1, 2, 2a)". In the MA and SRA tables only the species with age data are included. In SRL and SRA tables in the COMMENTS column is reported for completeness the string "Observed F/(F+M)".

The output data frame is in accordance with template of the DG MARE Med&BS Data Call.

For further details on the scripts see **ANNEX XVI RDBprocessing_0.0.1** and APPENDIX - R code of the functions.

2.2 FDI datacall - CATCH_FDI

STREAM developed R scripts that allows to create the following tables:

- **Table C**. Discard biological data (age based);
- **Table D**. Discard biological data (length based);
- **Table E**. Landings biological data (age based);
- **Table F**. Landings biological data (length based).

Nevertheless, these four tables are no longer required by Mediterranean and Black Sea, and thus have not been included in the package.

The only table required is the Catch table for which no script was available from STREAM project.

A new function has been, thus, developed from scratch to be integrated in the RDBprocessing package and in the RDBFIS platform.

The function uses as input the RCG CL landing table, where the information on landing and landing value are available by metier but not by vessel length; thus, the vessel length column is reported as "NK". Information on the discard cannot be automatically reported, but needs to be entered a posteriori, manipulating off-line the information from MED&BS data call that not always is reported at the same aggregation level.

The output data frame is in accordance with the FDI Data Call template.

For further details on the scripts see **ANNEX XVI RDBprocessing_0.0.1** and APPENDIX - R code of the functions.

2.3 GFCM DCRF datacall: TableVII31 and TabII2_GFCM

The scripts developed in STREAM were transformed in R functions and included in the RDBprocessing R package. Specifically, the functions included in the package allow to create:

- Table II.2 - Catch data per species: the total catch in weight, including landing and discards (if present and recorded), by main commercial species, fleet segment (Appendix B of GFCM DCRF) and area (GSA);
- Tables VII.3.1 - Other biological data: *Maturity data* table (species of Group 1 Appendix A of GFCM DCRF).

The DCRF requires the fishery data aggregated according to the GFCM's fleet segments definition, while the Data Collection framework requires the data aggregated by metier and LOA.

For this reason, a communication table between GFCM fleet segments and metier-LOA defined within DCF is necessary to provide the sampled and raised data consistently with the DG MARE Med&BS Data Call. The table has been included in the package (CT table, Figure 2.3-1).

	Fleet_segment	Description	METIER	LOA	GEAR_ACatch	MESH_SIZE_RANGE_ACatch	FISHERY_ACatch
1	D-01	Dredgers (< 6)	DRB_MOL_0_0_0	VL0006	DRB	NK	MOL
2	D-02	Dredgers (6 - 12)	DRB_MOL_0_0_0	VL0612	DRB	NK	MOL
3	D-03	Dredgers (12 - 24)	DRB_MOL_0_0_0	VL1218	DRB	NK	MOL
4	D-03	Dredgers (12 - 24)	DRB_MOL_0_0_0	VL1824	DRB	NK	MOL
5	D-04	Dredgers (> 24)	DRB_MOL_0_0_0	VL2440	DRB	NK	MOL
6	D-04	Dredgers (> 24)	DRB_MOL_0_0_0	VL40XX	DRB	NK	MOL
7	L-01	Longliners (< 6)	LLS_DEF_0_0_0	VL0006	LLS	NK	DEF
8	L-02	Longliners (6 - 12)	LLS_DEF_0_0_0	VL0612	LLS	NK	DEF
9	L-03	Longliners (12 - 24)	LLS_DEF_0_0_0	VL1218	LLS	NK	DEF
10	L-03	Longliners (12 - 24)	LLS_DEF_0_0_0	VL1824	LLS	NK	DEF
11	L-04	Longliners (> 24)	LLS_DEF_0_0_0	VL2440	LLS	NK	DEF
12	L-04	Longliners (> 24)	LLS_DEF_0_0_0	VL40XX	LLS	NK	DEF
13	P-05	Small-scale vessels with engine using passive gears (< 6)	FPO_DEF_0_0_0	VL0006	FPO	NK	DEF
14	P-05	Small-scale vessels with engine using passive gears (< 6)	FYK_CAT_0_0_0	VL0006	FYK	NK	CAT
15	P-05	Small-scale vessels with engine using passive gears (< 6)	FYK_DEF_0_0_0	VL0006	FYK	NK	DEF
16	P-05	Small-scale vessels with engine using passive gears (< 6)	GND_SPF_0_0_0	VL0006	GND	NK	SPF
17	P-05	Small-scale vessels with engine using passive gears (< 6)	GNS_DEF_>=16_0_0	VL0006	GNS	16D20	DEF
18	P-05	Small-scale vessels with engine using passive gears (< 6)	GNS_SLP_>=16_0_0	VL0006	GNS	16D20	SLP
19	P-05	Small-scale vessels with engine using passive gears (< 6)	GTR_DEF_>=16_0_0	VL0006	GTR	16D20	DEF

Figure 2.3-1 CT (communication table) between GFCM fleet segments and DCF metier-vessel length.

For further details on the scripts see **ANNEX XVI RDBprocessing_0.0.1** and APPENDIX - R code of the functions.

3 Technical features and testing

The RDBprocessing package has been created with the R version 4.1.2 (R Core Team, 2021), within R studio environment (version 2022.12.0 Build 353). The roxygen2 library was used to automatically generate the function documentation in-line with the code.

Devtools (R CMD check) (Wickham et al., 2021) has been used to build the package in Rstudio, running all sorts of checks on the contents of the R package. This tool gives warning and error messages when it finds things that are not properly specified within the package. It also runs the examples reported in the .Rd files for each of the functions, as well as other automated included tests.

Before submitting a package to the R CRAN, R CMD check (including the option --as-cran) needs to be carried out to verify that in the package there are no warnings or errors and other incompatibilities with the CRAN policies.

The *dependencies* of the RDBprocessing package are: dplyr, COSTeda, COSTcore, COSTdbe, magrittr, methods, data.table, tidyr, lubridate, plyr, reshape2; suggested: rmarkdown, knitr, markdown (Hadley et al, 2021; COST Team et al., 2017a; COST Team et al., 2017b; COST Team et al., 2017c; Bache and Wickham, 2020; R Core Team, 2021; Dowle and Srinivasan, 2021; Wickham, 2021; Grolemund and Wickham, 2011; Wickham, 2011; Wickham, 2007; Allaire et al., 2022; Xie, 2022;Allaire et al., 2019).

The structure followed in the package is shown in Figure 3.1 and is in line with the structure established by R CRAN. In R folder all the R scripts corresponding to processing functions are contained; in “man” folder the code related to the documentation of the functions and of data examples are stored. In data folder the example data and other useful tables are contained.

The COST packages, that are needed to use the RDBprocessing functions are store here:

https://github.com/COISPA/RDBprocessing/tree/main/COSTpackages.

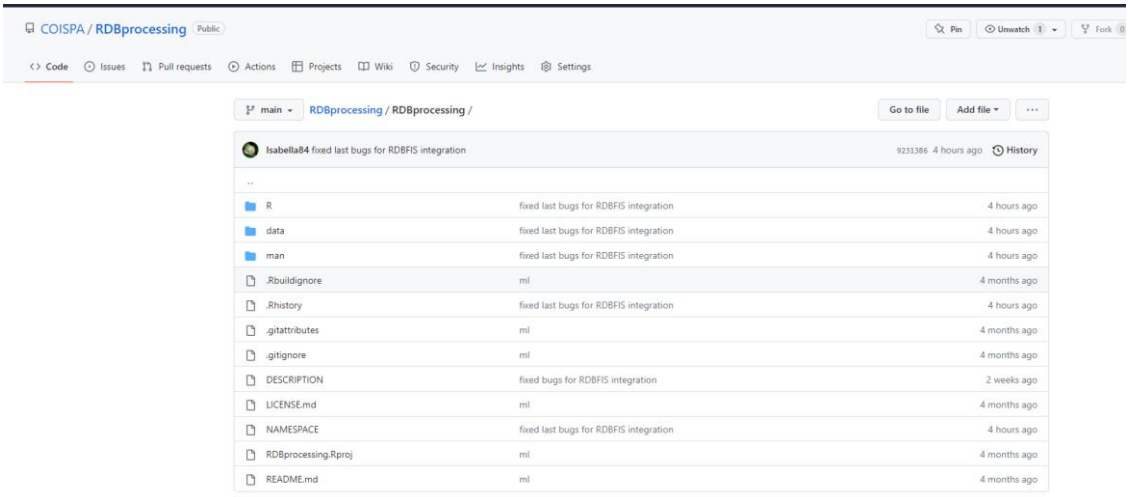

Figure 3-1 Example of structure followed in the creation of RDBprocessing packages.

RDBrpocessing package was tested by several COISPA experts on GSA 18 data as standalone application; moreover, during the “Med&BS RDBFIS WP5 Meetings, testing phase - 2nd WORKSHOP” held the 23rd January 2023, a real time demonstration of the functioning of the package was made.

Finally, the RDBprocessing package was tested through RDBFIS web application on the GSA 18 data.

4 Conclusions

The aim of Task 4.4 - Developing data processing tools – was to consolidate the auxiliary tools developed in STREAM and to include them in an R package to be easily integrated in RDBFIS as well as used as a standalone application to produce the necessary tables in the different data calls formats.

This task developed an open-access (available on GitHub repository) and well documented tools for fishery data processing, data management and conversion into different formats required by several end-users to dramatically reduce the risk of data failure of the MS and CPs in the data submission to the three main data calls present in Mediterranean and Black Sea: MED&BS, FDI, GFCM DCRF and RCG.

As these tools have been implemented in R language, they can be easily adapted to future new formats and generalized to be fed by any type of input format. Moreover, these tools have been incorporated in RDBFIS, to allow the production of the aggregated data in the format required by the specific datacalls, starting from the same dataset, thus ensuring a higher consistency of the data required by the different datacalls.

The storing on the GitHub repository and the development in R language will facilitate this code adaptation and improvement, allowing the collaboration among the experts involved in the data calls. This is also expected to contribute to ease the work of data processing at regional level, improving the data quality.

Further generalization of the processing functions would be needed to easily adapt the algorithms already implemented in STREAM (e.g. raising from detailed data to the length and age structures of the landing and to the discard volume) according to different sampling designs, defined and tracked in a data base structure in line with RDBES concept. This generalization of the functions could be conceived making explicit the sampling hierarchy applied to collect the stored detailed data.

The idea could be to develop, for each data call, different functions depending on the sampling scheme and the corresponding sampling frame strata of the commercial data.

A communication with RDBES working group has been established during the project and is expected to be maintained in the future. This collaboration is expected to be beneficial for the generalization of the processing methodology following the RDBES concept.

5 References

Allaire J.J., Horner J., Xie Y., Marti V., Porte N. (2019). markdown: Render Markdown with the C Library 'Sundown'. R package version 1.1

Allaire,J.J., Xie, Y., McPherson, J., Luraschi, J.,Ushey, K., Atkins, A., Wickham, H.,Cheng, J., Chang, W., Iannone, R. (2022). rmarkdown: Dynamic Documents for R. R package version 2.17. URL https://rmarkdown.rstudio.com

Bache S.M., Wickham H. (2020). magrittr: A Forward-Pipe Operator for R. R package version 2.0.1

COST Team et al. (2017a). COSTeda: Exploratory Data Analysis methods for COST. R package version 1.4.0.

COST Team et al. (2017b). COSTcore: Core package of COST. R package version 1.4-0.

COST Team et al. (2017c). COSTdbe: Design-based estimates package for COST. R package version 1.4-1.

Dowle, M, Srinivasan, A. (2021). data.table: Extension of `data.frame`. R package version 1.14.2. https://CRAN.R-project.org/package=data.table

Grolemund G., Wickham H. (2011). Dates and Times Made Easy with lubridate. Journal of Statistical Software, 40(3), 1-25.

Hadley Wickham, Romain François, Lionel Henry and Kirill Müller (2021). dplyr: A Grammar of Data Manipulation. R package version 1.0.7. https://CRAN.R-project.org/package=dplyr

R Core Team (2021). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria.

Wickham H. (2007). Reshaping Data with the reshape Package. Journal of Software, 21(12), 1-20.

Wickham H. (2011). The Split-Apply-Combine Strategy for Data Analysis. Journal of Statistical Software, 40(1), 1-29.

Wickham, H. (2021). tidyr: Tidy Messy Data. R package version 1.1.4. https://CRAN.R-project.org/package=tidyr

Wickham H., Hester J.,Chang W., Bryan J. (2021). devtools: Tools to Make Developing R Packages Easier. R package version 2.4.3.

Xie, Y. (2022). knitr: A General-Purpose Package for Dynamic Report Generation in R. R package version 1.40

6 APPENDIX - R code of the functions

```
#' Function converting RCG CS in COST CS object
#' @param data Detailed data in RCG CS format
#' @param verbose boolean. If TRUE a message is printed.
#' @return COST CS object
#' @export
#' @examples RCGtoCOST_CS(RDBprocessing::data_ex)
#' @importFrom methods new
#' @importFrom dplyr mutate
#' @importFrom magrittr %>%
#' @importFrom COSTcore csData
#' @importFrom lubridate quarter month

RCGtoCOST_CS<-function(data, verbose = FALSE){

#path.data=getwd()

    data=check_cs_header(data)

CS=data
error <- FALSE
dataset_proj <- ""
bad.rm <- FALSE
log_var  <- TRUE
lenNum <- log_varMsg<- subSampWt<- vslFlgCtry<- log_varCs<-NULL

Date=format(as.Date(CS$Date, format = "%d/%m/%Y"), "%Y-%m-%d")
if (!all(is.na(Date))) {
    CS$Date=Date

}

    names(CS)[which(tolower(names(CS)) == "sampling.type")] <- "sampType"
    names(CS)[which(tolower(names(CS)) == "sampling_type")] <- "sampType"
    names(CS)[which(tolower(names(CS)) == "samplingtype")] <- "sampType"
    names(CS)[which(tolower(names(CS)) == "flag.country")] <- "vslFlgCtry"
    names(CS)[which(tolower(names(CS)) == "flag_country")] <- "vslFlgCtry"
    names(CS)[which(tolower(names(CS)) == "flagcountry")] <- "vslFlgCtry"
    names(CS)[which(tolower(names(CS)) == "year")] <- "year"
    names(CS)[which(tolower(names(CS)) == "trip.code")] <- "trpCode"
    names(CS)[which(tolower(names(CS)) == "trip_code")] <- "trpCode"
    names(CS)[which(tolower(names(CS)) == "tripcode")] <- "trpCode"
    names(CS)[which(tolower(names(CS)) == "number.of.sets...hauls.on.trip")] <- "foNum"
    names(CS)[which(tolower(names(CS)) == "number_of_sets_hauls_on_trip")] <- "foNum"
    names(CS)[which(tolower(names(CS)) == "number_of_setshauls")] <- "foNum"
    names(CS)[which(tolower(names(CS)) == "nsets")] <- "foNum"
    names(CS)[which(tolower(names(CS)) == "days.at.sea")] <- "daysAtSea"
    names(CS)[which(tolower(names(CS)) == "days_at_sea")] <- "daysAtSea"
    names(CS)[which(tolower(names(CS)) == "daysatsea")] <- "daysAtSea"
    names(CS)[which(tolower(names(CS)) == "sampling.method")] <- "sampMeth"
    names(CS)[which(tolower(names(CS)) == "sampling_method")] <- "sampMeth"
    names(CS)[which(tolower(names(CS)) == "aggregation.level")] <- "aggLev"
    names(CS)[which(tolower(names(CS)) == "aggregation_level")] <- "aggLev"
    names(CS)[which(tolower(names(CS)) == "aggregationlevel")] <- "aggLev"
    names(CS)[which(tolower(names(CS)) == "station.number")] <- "staNum"
    names(CS)[which(tolower(names(CS)) == "station_number")] <- "staNum"
    names(CS)[which(tolower(names(CS)) == "stationnumber")] <- "staNum"
    names(CS)[which(tolower(names(CS)) == "catch.registration")] <- "catReg"
    names(CS)[which(tolower(names(CS)) == "catch_registration")] <- "catReg"
    names(CS)[which(tolower(names(CS)) == "catchregistration")] <- "catReg"
    names(CS)[which(tolower(names(CS)) == "species.registration")] <- "sppReg"
    names(CS)[which(tolower(names(CS)) == "species_registration")] <- "sppReg"
    names(CS)[which(tolower(names(CS)) == "speciesregistration")] <- "sppReg"
    names(CS)[which(tolower(names(CS)) == "date")] <- "date"
    names(CS)[which(tolower(names(CS)) == "area")] <- "area"
    names(CS)[which(tolower(names(CS)) ==
                        "fishing.activity.category.national")] <- "foCatNat"
    names(CS)[which(tolower(names(CS)) ==
                        "fishing_activity_category_european_lvl_6")] <- "foCatEu6"

    names(CS)[which(tolower(names(CS)) ==
                        "fishing_activity_category_national")] <- "foCatNat"
    names(CS)[which(tolower(names(CS)) ==
                        "fishingactivitycategorynational")] <- "foCatNat"
    names(CS)[which(tolower(names(CS)) == "fac_national")] <- "foCatNat"
    names(CS)[which(tolower(names(CS)) ==
                        "fishing.activity.category.european.lvl.6")] <- "foCatEu6"
    names(CS)[which(tolower(names(CS)) == "metier")] <- "foCatEu6"
    names(CS)[which(tolower(names(CS)) == "fac_ec_lvl6")] <- "foCatEu6"
    names(CS)[which(tolower(names(CS)) == "species")] <- "spp"
    names(CS)[which(tolower(names(CS)) == "catch.category")] <- "catchCat"
    names(CS)[which(tolower(names(CS)) == "catch_category")] <- "catchCat"
    names(CS)[which(tolower(names(CS)) == "catchcategory")] <- "catchCat"
    names(CS)[which(tolower(names(CS)) == "commercial.size.category")] <- "commCat"
    names(CS)[which(tolower(names(CS)) == "commercial_size_category")] <- "commCat"
    names(CS)[which(tolower(names(CS)) == "weight")] <- "wt"
    names(CS)[which(tolower(names(CS)) == "subsample.weight")] <- "subSampWt"
```

```
names(CS)[which(tolower(names(CS)) == "subsample_weight")] <- "subSampWt"
names(CS)[which(tolower(names(CS)) == "subsampleweight")] <- "subSampWt"
names(CS)[which(tolower(names(CS)) == "length.code")] <- "lenCode"
names(CS)[which(tolower(names(CS)) == "length_code")] <- "lenCode"
names(CS)[which(tolower(names(CS)) == "lengthcode")] <- "lenCode"
names(CS)[which(tolower(names(CS)) == "length.class")] <- "lenCls"
names(CS)[which(tolower(names(CS)) == "length_class")] <- "lenCls"
names(CS)[which(tolower(names(CS)) == "lengthclass")] <- "lenCls"
names(CS)[which(tolower(names(CS)) == "number.at.length")] <- "lenNum"
names(CS)[which(tolower(names(CS)) == "number_at_length")] <- "lenNum"
names(CS)[which(tolower(names(CS)) == "numberatlength")] <- "lenNum"


# new Fields RCG 2018 ----------------------------------------------------

names(CS)[which(tolower(names(CS)) == "commercial.size.category.scale")] <- "commCatScl"
names(CS)[which(tolower(names(CS)) == "commercial_size_category_scale")] <- "commCatScl"
names(CS)[which(tolower(names(CS)) == "commercialsizecategoryscale")] <- "commCatScl"

names(CS)[which(tolower(names(CS)) == "fish.id")] <- "fishId"
names(CS)[which(tolower(names(CS)) == "fish_id")] <- "fishId"
names(CS)[which(tolower(names(CS)) == "fishid")] <- "fishId"

names(CS)[which(tolower(names(CS)) == "individual.weight")] <- "indWt"
names(CS)[which(tolower(names(CS)) == "individual_weight")] <- "indWt"
names(CS)[which(tolower(names(CS)) == "individualweight")] <- "indWt"

names(CS)[which(tolower(names(CS)) == "harbour")] <- "harbour"

names(CS)[which(tolower(names(CS)) == "duration.of.fishing.operation")] <- "foDur"
names(CS)[which(tolower(names(CS)) == "duration_of_fishing_operation")] <- "foDur"
names(CS)[which(tolower(names(CS)) == "durationoffishing_operation")] <- "foDur"

names(CS)[which(tolower(names(CS)) == "initial.latitude")] <- "latIni"
names(CS)[which(tolower(names(CS)) == "initial_latitude")] <- "latIni"
names(CS)[which(tolower(names(CS)) == "initiallatitude")] <- "latIni"

names(CS)[which(tolower(names(CS)) == "initial.longitude")] <- "lonIni"
names(CS)[which(tolower(names(CS)) == "initial_longitude")] <- "lonIni"

names(CS)[which(tolower(names(CS)) == "final.latitude")] <- "latFin"
names(CS)[which(tolower(names(CS)) == "final_latitude")] <- "latFin"
names(CS)[which(tolower(names(CS)) == "finallatitude")] <- "latFin"

names(CS)[which(tolower(names(CS)) == "final.longitude")] <- "lonFin"
names(CS)[which(tolower(names(CS)) == "final_longitude")] <- "lonFin"
names(CS)[which(tolower(names(CS)) == "finallongitude")] <- "lonFin"

names(CS)[which(tolower(names(CS)) == "depth.of.fishing.operation")] <- "foDep"
names(CS)[which(tolower(names(CS)) == "depth_of_fishing_operation")] <- "foDep"
names(CS)[which(tolower(names(CS)) == "depthoffishingoperation")] <- "foDep"

names(CS)[which(tolower(names(CS)) == "water.depth")] <- "waterDep"
names(CS)[which(tolower(names(CS)) == "water_depth")] <- "waterDep"
names(CS)[which(tolower(names(CS)) == "waterdepth")] <- "waterDep"

names(CS)[which(tolower(names(CS)) == "sex")] <- "sex"

names(CS)[which(tolower(names(CS)) == "maturity.method")] <- "matMeth"
names(CS)[which(tolower(names(CS)) == "maturity_method")] <- "matMeth"
names(CS)[which(tolower(names(CS)) == "maturitymethod")] <- "matMeth"

names(CS)[which(tolower(names(CS)) == "maturity.scale")] <- "matScale"
names(CS)[which(tolower(names(CS)) == "maturity_scale")] <- "matScale"
names(CS)[which(tolower(names(CS)) == "maturityscale")] <- "matScale"


names(CS)[which(tolower(names(CS)) == "maturity.stage")] <- "matStage"
names(CS)[which(tolower(names(CS)) == "maturity_stage")] <- "matStage"
names(CS)[which(tolower(names(CS)) == "maturitystage")] <- "matStage"

names(CS)[which(tolower(names(CS)) == "ageing.method")] <- "ageMeth"
names(CS)[which(tolower(names(CS)) == "ageing_method")] <- "ageMeth"
names(CS)[which(tolower(names(CS)) == "ageingmethod")] <- "ageMeth"

names(CS)[which(tolower(names(CS)) == "age")] <- "age"


CS[is.na(CS[,])] <- -1


CS$aggLev <- as.character(CS$aggLev)
CS$aggLev[ CS$aggLev %in% c("t","TRUE",TRUE)] <- "T" ##!!

CS$spp <- unlist(lapply(CS$spp, function(x) paste(toupper(substring(x,1, 1)),
                                                  tolower(substring(x, 2)), sep = "")))

CS <-CS %>% mutate(proj=dataset_proj,landCtry=vslFlgCtry) ##!!
```

```
trPk <- c("sampType", "vslFlgCtry", "year", "trpCode","proj","landCtry") ##!!
trOther <- c("foNum", "daysAtSea", "sampMeth",
             "harbour" ) # new : harbour

hhPk <- c(trPk, "staNum")
hhOther <- c("aggLev", "catReg", "sppReg", "date", "area",
             "foCatNat", "foCatEu6",
             "foDur", "latIni", "lonIni", "latFin", "lonFin", "foDep", "waterDep") # new

# modified respect to the previous version
slPk <- c(hhPk, "spp", "catchCat", "commCat", "commCatScl","sex") # new
slOther <- c("wt", "subSampWt", "lenCode")

#slPk <- c(hhPk, "spp", "catchCat", "commCat", "commCatScl","sex", "wt", "subSampWt") # new
#slOther <- c( "lenCode")

hlPk <- c(slPk, "lenCls")
hlOther <- c("lenNum")

# modified respect to the previous version
#caPk <- c(hlPk, "age","area","fishId")
#caOther <- c( "matMeth",  "matScale" ,"matStage", "ageMeth" , "indWt", "lenCode")

caPk <- c(hlPk, "age","area","fishId", "matStage")
caOther <- c( "matMeth",  "matScale" , "ageMeth" , "indWt", "lenCode")

allFields <- c(caPk, trOther, hhOther, slOther, hlOther,caOther)
missingFields <- allFields[!allFields %in% names(CS)]

# check if all fields are used

# names(CS)[!names(CS) %in% allFields]

if (length(missingFields) > 0) {
    stop("Missing fields : ", paste(missingFields, collapse = ", ",
                                    sep = ""))
}

csTr <- unique(CS[, c(trPk, trOther)])

trPkV <- fpKey(csTr, trPk)

trPkVDup <- trPkV %in% trPkV[duplicated(trPkV)]
if (any(trPkVDup)) {
    print(trPkVDup)
    if (log_var) {
        log_varCs$duplicated_TR <- fpKey(CS, trPk) %in% trPkV[trPkVDup]
    }
    if (bad.rm) {
        #message("Integrity problem for CS/TR, ", sum(trPkVDup),
              #  " row(s) removed.", log_varMsg)
        print("Removed following row(s):", quote = F)
        print(csTr[trPkVDup, ])
        CS <- merge(CS, csTr[!trPkVDup, ])
    }    else {
        error <- TRUE
        message("Integrity problem for CS/TR, ", sum(trPkVDup),
                " row(s) concerned.", log_varMsg)
    }
}
print(paste("No rows TR =",  nrow(csTr)), quote=F)


csHh <- unique(CS[, c(hhPk, hhOther)])
hhPkV <- fpKey(csHh, hhPk)
hhPkVDup <- hhPkV %in% hhPkV[duplicated(hhPkV)]
if (any(hhPkVDup)) {
    if (log_var) {
        log_varCs$duplicated_HH <- fpKey(CS, hhPk) %in% hhPkV[hhPkVDup]
    }
    if (bad.rm) {
        message("Integrity problem for CS/HH, ", sum(hhPkVDup),
                " row(s) removed.", log_varMsg)
        print("Removed following row(s):", quote = F)
        print(csHh[hhPkVDup, ])
        CS <- merge(CS, csHh[!hhPkVDup, ])
    }
    else {
        error <- TRUE
        message("Integrity problem for CS/HH, ", sum(hhPkVDup),
                " row(s) concerned.", log_varMsg)
    }
}
print(paste("No rows HH =",  nrow(csHh)), quote=F)


csSl <- unique(CS[, c(slPk, slOther)])
slPkV <- fpKey(csSl, slPk)
slPkVDup <- slPkV %in% slPkV[duplicated(slPkV)]
if (any(slPkVDup)) {
```

```
        if (log_var) {
            log_varCs$duplicated_SL <- fpKey(CS, slPk) %in% slPkV[slPkVDup]
        }
        if (bad.rm) {
            message("Integrity problem for CS/SL, ", sum(slPkVDup),
                    " row(s) removed.", log_varMsg)
            print("Removed following row(s):", quote = F)
            print(csSl[slPkVDup, ])
            CS <- merge(CS, csSl[!slPkVDup, ])
        }
        else {
            error <- TRUE
            message("Integrity problem for CS/SL, ", sum(slPkVDup),
                    " row(s) concerned.", log_varMsg)
        }
    }
    print(paste("No rows SL =",  nrow(csSl)), quote=F)


    CS_aggregated_by_length <-  aggregate(CS$lenNum, by=list(CS$sampType , CS$vslFlgCtry ,
                                                             CS$year,  CS$trpCode, CS$proj, CS$landCtry,
CS$harbour,
                                                             CS$foNum, CS$daysAtSea ,  CS$sampMeth, CS$aggLev,
CS$staNum,
                                                             CS$foDur, CS$foDep, CS$catReg, CS$sppReg, CS$date,
CS$area,
                                                             CS$foCatNat, CS$foCatEu6, CS$spp,  CS$catchCat,
CS$wt,
                                                             CS$subSampWt, CS$sex, CS$lenCode, CS$lenCls,
                                                             CS$commCatScl, CS$commCat), FUN="sum")

    colnames(CS_aggregated_by_length) <- c("sampType" , "vslFlgCtry" , "year" , "trpCode" ,
                                           "proj" , "landCtry","harbour" , "foNum" , "daysAtSea" ,
                                           "sampMeth" ,  "aggLev" , "staNum" , "foDur" , "foDep" ,
                                           "catReg", "sppReg" , "date" , "area" , "foCatNat" ,
                                           "foCatEu6",  "spp" , "catchCat" , "wt" , "subSampWt" ,
                                           "sex" , "lenCode" , "lenCls" , "commCatScl" , "commCat",
                                           "lenNum")

    # sum(CS$lenNum)
    # sum(CS_aggregated_by_length$lenNum)

    csHl <- unique(CS_aggregated_by_length[, c(hlPk, hlOther)])
    hlPkV <- fpKey(csHl, hlPk)
    hlPkVDup <- hlPkV %in% hlPkV[duplicated(hlPkV)]
    if (any(hlPkVDup)) {
        if (log_var) {
            log_varCs$duplicated_HL <- fpKey(CS, hlPk) %in% hlPkV[hlPkVDup]
        }
        else {
            error <- TRUE
            message("Integrity problem for CS/HL, ", sum(hlPkVDup),
                    " row(s) concerned.", log_varMsg)
            print("Check the following row(s):", quote = F)
            csHl[hlPkVDup, ]
        }
    }
    print(paste("No rows HL =",  nrow(csHl)), quote=F)

    #   # check CA ------------------------------------------------------------

    csCa <- unique(CS[, c(caPk, caOther)])
    caPkV <- fpKey(csCa, caPk)
    caPkVDup <- caPkV %in% caPkV[duplicated(caPkV)]
    if (any(caPkVDup)) {
        if (log_var) {
            log_varCs$duplicated_CA <- fpKey(CS, caPk) %in% caPkV[caPkVDup]
        }
        if (bad.rm) {
            message("Integrity problem for CS/CA, ", sum(caPkVDup),
                    " row(s) removed.", log_varMsg)
            print("Removed following row(s):", quote = F)
            print(csCa[caPkVDup, ])
            CS <- merge(CS, csCa[!caPkVDup, ])
        } else {
            error <- TRUE
            message("Integrity problem for CS/CA, ", sum(caPkVDup),
                    " row(s) concerned.", log_varMsg)
            print("Check the following row(s):", quote = F)
            csCa[caPkVDup, ]
        }
    }


    # All CS tables names -------------------------------------------------

    ##!! Simpler way to define names for the CS tables
    obj <- new("csData")
    tr0 <- obj@tr
    hh0 <- obj@hh
```

```
    sl0 <- obj@sl
    hl0 <- obj@hl
    ca0 <- obj@ca

    TR.col <- names(tr0)
    HH.col <- names(hh0)
    SL.col <- names(sl0)
    HL.col <- names(hl0)
    CA.col <- names(ca0)

    ## end names ##!!

    if (!error) {

        missTR<-TR.col[!TR.col %in% names(csTr)]
        costCS.TR <- csTr
        costCS.TR[ ,missTR] <- NA
        costCS.TR<-  costCS.TR[,TR.col]

        #write.table(costCS.TR, file.path(path.data, "SDEF CS-TR data.csv"),
                   # sep = ";", row.names = FALSE)

        missHH<-HH.col[!HH.col %in% names(csHh)]
        costCS.HH <- csHh
        costCS.HH[ , missHH] <- NA

        costCS.HH<-  costCS.HH[,HH.col]

        #write.table(costCS.HH, file.path(path.data, "SDEF CS-HH data.csv"),
                   # sep = ";", row.names = FALSE)


        missSL<-SL.col[!SL.col %in% names(csSl)]
costCS.SL <- csSl
        costCS.SL[ , missSL] <- NA

        costCS.SL<-  costCS.SL[,SL.col]

#write.table(costCS.SL, file.path(path.data, "SDEF CS-SL data.csv"),
                    #sep = ";", row.names = FALSE)


        missHL<-HL.col[!HL.col %in% names(csHl)]
        costCS.HL <- csHl
        costCS.HL[ , missHL] <- NA

        costCS.HL <-  costCS.HL[,HL.col]

        #write.table(costCS.HL, file.path(path.data, "SDEF CS-HL data.csv"),
                    #sep = ";", row.names = FALSE)

        if (all(CS$fishId == -1)) {

            ## add subSampWt (SL) and lenNum (HL) to CA

            ca.sl.hl.COL<-c(caPk, caOther,"subSampWt","lenNum")
            ca.sl.hl <- unique(CS[, ca.sl.hl.COL])

            # indWt
            ca.sl.hl<-ca.sl.hl %>% mutate(indWt=subSampWt/lenNum)
            ca.sl.hl1 <- ca.sl.hl[rep(row.names(ca.sl.hl), ca.sl.hl$lenNum), 1:ncol(ca.sl.hl)]
            ca.sl.hl1$fishId <- 1:nrow(ca.sl.hl1)
            costCS.CA <- ca.sl.hl1[, names(  ca.sl.hl1) %in% CA.col]

missCA<-CA.col[!CA.col %in% names(costCS.CA)]
            costCS.CA[ , missCA] <- NA
            costCS.CA<-  costCS.CA[,CA.col]
costCS.CA$indWt <- -1

            merge=merge(costCS.CA,costCS.HH,by="trpCode")

            merge$quarter=quarter(as.Date(merge$date))
            merge$month=month(as.Date(merge$date))



            trip= unique(merge$trpCode)

            for (tr in trip){
costCS.CA[costCS.CA$trpCode==tr,]$quarter=merge[merge$trpCode==tr,]$quarter
costCS.CA[costCS.CA$trpCode==tr,]$month=merge[merge$trpCode==tr,]$month
            }



        } else {

            missCA <- CA.col[!CA.col %in% names(csCa)]
costCS.CA <- csCa
            costCS.CA[ , missCA] <- NA
```

```
merge=merge(costCS.CA,costCS.HH,by="trpCode")

            merge$quarter=quarter(as.Date(merge$date))
            merge$month=month(as.Date(merge$date))



            trip= unique(merge$trpCode)

            for (tr in trip){
costCS.CA[costCS.CA$trpCode==tr,]$quarter=merge[merge$trpCode==tr,]$quarter
costCS.CA[costCS.CA$trpCode==tr,]$month=merge[merge$trpCode==tr,]$month
            }


            costCS.CA$proj<- dataset_proj
            costCS.CA$landCtry<-costCS.CA$vslFlgCtry
costCS.CA<-  costCS.CA[,CA.col]

costCS.CA$fishId=seq(1,dim(costCS.CA)[1],by=1)

        }

        print(paste("No rows CA =",  nrow(costCS.CA)), quote=F)

       # write.table(costCS.CA, file.path(path.data, "SDEF CS-CA data.csv"),
                    #sep = ";", row.names = FALSE)

        costCS.CA$quarter<- factor(costCS.CA$quarter, levels = c(1,2,3,4))

        costCS.CA$month<- factor(costCS.CA$month, levels = c(1,2,3,4,5,6,7,8,9,10,11,12))

        costCS = csData(tr =costCS.TR, hh = costCS.HH, sl = costCS.SL,
hl = costCS.HL, ca=costCS.CA)

#saveRDS(costCS, "costCS.rds")

    } else {

        print("An error occurred in the trasformation.
        Impossible to create the CS COST object!")
    }

    return(costCS)
}


#' Function converting RCG CL in COST CL object
#' @param data Landing data in RCG CL format
#' @param verbose boolean. If TRUE a message is printed.
#' @return COST CL object
#' @export
#' @examples RCGtoCOST_CL(RDBprocessing::data_exampleCL)
#' @importFrom methods new
#' @importFrom dplyr mutate
#' @importFrom magrittr %>%
#' @import COSTcore

RCGtoCOST_CL<-function(data, verbose = FALSE){


CL=data
    error <- FALSE
    dataset_proj <- ""
bad.rm <- FALSE
    log_var  <- TRUE
    #lenNum <- log_varMsg<- vslFlgCtry
    logMsg<- NULL

    names(CL)[which(tolower(names(CL)) == "flag.country")] <- "vslFlgCtry"
    names(CL)[which(tolower(names(CL)) == "flag_country")] <- "vslFlgCtry"

    names(CL)[which(tolower(names(CL)) == "year")] <- "year"
    names(CL)[which(tolower(names(CL)) == "quarter")] <- "quarter"
    names(CL)[which(tolower(names(CL)) == "month")] <- "month"
    names(CL)[which(tolower(names(CL)) == "area")] <- "area"
    names(CL)[which(tolower(names(CL)) == "species")] <- "taxon"

    names(CL)[which(tolower(names(CL)) ==
                        "fishing.activity.category.national")] <- "foCatNat"
    names(CL)[which(tolower(names(CL)) == "fac_national")] <- "foCatNat"
    names(CL)[which(tolower(names(CL)) ==
                        "fishing_activity_category_national")] <- "foCatNat"
```

```
names(CL)[which(tolower(names(CL)) ==
                    "fishing.activity.category.european.lvl.6")] <- "foCatEu6"
names(CL)[which(tolower(names(CL)) ==
                    "fishing_activity_category_eu_l6")] <- "foCatEu6"
names(CL)[which(tolower(names(CL)) == "fac_ec_lvl6")] <- "foCatEu6"

names(CL)[which(tolower(names(CL)) == "harbour")] <- "harbour"

names(CL)[which(tolower(names(CL)) == "official.landings.weight")] <- "landWt"
names(CL)[which(tolower(names(CL)) == "official_landings_weight")] <- "landWt"

names(CL)[which(tolower(names(CL)) == "official.landings.value")] <- "landValue"
names(CL)[which(tolower(names(CL)) == "official_landings_value")] <- "landValue"


## primary keys & fields

clPk <- c("vslFlgCtry", "year", "quarter", "month", "area", "taxon",
          "foCatNat", "foCatEu6")
clOther <- c("landWt", "landValue")

# check fields
allFields <- c(clPk, clOther)
missingFields <- allFields[! allFields %in% names(CL)]
if (length(missingFields) > 0) {
    stop("Missing fields : ", paste(missingFields, collapse = ", ", sep=""))
}

clPkV <- fpKey(CL, clPk)
clPkVDup <- clPkV %in% clPkV[duplicated(clPkV)]

# test integrity
if (any(clPkVDup)) {
    if (log) {
        logCl$duplicated <-  fpKey(CL, clPk)
    }


    if (bad.rm) {
        message("Integrity problem for CL, ", sum(clPkVDup),
                " row(s) removed.", logMsg)
        CL <- CL[! clPkVDup,]
    } else {
        error <- TRUE
        message("Integrity problem for CL, ", sum(clPkVDup),
                " row(s) concerned.", logMsg)
    }
}



if (error) {
    stop("Stop on reported errors.")
}


# formating

CL$taxon <- unlist(lapply(CL$taxon, function(x)
    paste(toupper(substring(x, 1, 1)), tolower(substring(x, 2)), sep="")))


## df
clDf <- data.frame(
    landCtry=NA,
    vslFlgCtry=CL$vslFlgCtry,
    year=CL$year,
    quarter=CL$quarter,
    month=CL$month,
    area=CL$area,
    rect=NA,
    subRect=NA,
    taxon=CL$taxon,
    landCat=NA,
    commCatScl=NA,
    commCat=NA,
    foCatNat=CL$foCatNat,
    foCatEu5=NA,
    foCatEu6=CL$foCatEu6,
    harbour=CL$harbour,
    vslLenCat=NA,
    unallocCatchWt=NA,
    misRepCatchWt=NA,
    landWt=CL$landWt,
    landMult=NA,
    landValue=CL$landValue,
    stringsAsFactors=FALSE)

#write.table(clDf, file.path(path.data,  "SDEF CL data.csv"), sep=";", row.names = FALSE)
```

```
    costCL = clData(cl=clDf)
    #saveRDS(costCL, "costCL.rds")

return(costCL)
    }


#' Landing by length (LANDINGS) table - MED & BS data call
#'
#' @param datacs Detailed data in RCG CS format
#' @param datacl Landings aggregated data in RCG CL format
#' @param verbose boolean. If TRUE a message is printed.
#' @return LANDINGS table
#' @export
#' @examples LAND_MEDBS(RDBprocessing::data_ex,RDBprocessing::data_exampleCL)
#' @importFrom stats complete.cases
#' @import COSTeda
#' @importFrom COSTdbe dbeObject RaiseLgth
#' @import COSTcore
#' @importFrom dplyr rename left_join bind_rows vars funs
#' @importFrom plyr round_any
#' @importFrom data.table as.data.table
#' @importFrom tidyr separate
#' @importFrom magrittr %>%
#' @importFrom stats time
#' @import reshape2
#'
LAND_MEDBS<-function(datacs,datacl,verbose=FALSE){

    datacs=check_cs_header(datacs)
FISHERY<- GEAR<- ID<- LENGTHCLASS100_PLUS<- LENGTHCLASS99<- MESH_SIZE_RANGE<-QUARTER<- SPECIES<- VESSEL_LENGTH<-
VL<- Year<- fishery<- gear<- id <- space<- stock<- technical<- value<-.<-NULL



    # datacs=data_ex
    # datacl=data_exampleCL


    fri_cs1<-RCGtoCOST_CS(datacs)

    fri_cl1<-RCGtoCOST_CL(datacl)


    fri_strD1 <- strIni(timeStrata="quarter", techStrata = "foCatEu6",
                        spaceStrata = "area")

    fri_csv <- csDataVal(fri_cs1)


    #fri_csv@ca$quarter<- factor(fri_csv@ca$quarter, levels = c(1,2,3,4))

    #fri_csv@ca$month<- factor(fri_csv@ca$month, levels = c(1,2,3,4,5,6,7,8,9,10,11,12))


    #fri_cs1@hh$time="00:00-23:59"

     fri_csc <- suppressWarnings(csDataCons(fri_csv, fri_strD1))


fri_clv <- clDataVal(fri_cl1)
     fri_clc <- clDataCons(fri_clv, fri_strD1)

# extract COUNTRY  and YEAR
    COUNTRY<-unique(fri_cs1@tr$landCtry)
    YEAR=unique(fri_cs1@tr$year)

    header=c("ID","COUNTRY","YEAR","QUARTER","VESSEL_LENGTH","GEAR","MESH_SIZE_RANGE","FISHERY","AREA","SPECON",
"SPECIES","LANDINGS","UNIT",paste("LENGTHCLASS",seq(0,99),sep=""),"LENGTHCLASS100_PLUS")


    mat=
aggregate(datacs$Length_class,by=list(datacs$Flag_country,datacs$Year,datacs$Area,datacs$Species),FUN="length")
    colnames(mat)=c("COUNTRY","YEAR","AREA","SPECIES","NB")
    tab1=merge(mat,RDBprocessing::Annex17,by.x="SPECIES",by.y="Scientific_name")


    colnames(tab1)[6]="SPE"
    sel_spe<-tab1
    sel_spe$lanEstim_methodDesc<-"analytical"

    colnames(sel_spe)[4]="GSA"
    sel_spe$SPECON<-""
    sel_spe$LC_RANGE=""
    if(nrow(sel_spe[sel_spe$UNIT=="cm",])>0) {    sel_spe[sel_spe$UNIT=="cm",]$LC_RANGE=10} else if
(nrow(sel_spe[sel_spe$UNIT=="mm",])>0){
    sel_spe[sel_spe$UNIT=="mm",]$LC_RANGE=1
}

i=1
```

```
for (i in 1:dim(sel_spe)[1]) {

    STK<- sel_spe$SPECIES[i]

    AREA <- sel_spe$GSA[i]

    fri_csc1<- COSTcore::subset(fri_csc, space==sel_spe$GSA[i],table="ca",link=T)
    fri_clc1<- COSTcore::subset(fri_clc, space==sel_spe$GSA[i],table="cl")

    # The first step is to create the empty object, that will be given
    # the appropriate values for the descritor  fields.

    lanEstim <-
        dbeObject(
            desc = paste(STK, AREA,"Landings", sep="_"),
            species = STK,
            catchCat = "LAN",
            strataDesc = fri_strD1,
            methodDesc = sel_spe$lanEstim_methodDesc[i]
        )

    # the only arguments to pass to the function are the dbe object,
    # the consolidated cs and cl datasets.

    if ( sel_spe$lanEstim_methodDesc[i]=="analytical"){
        lanEstim <- RaiseLgth(lanEstim, fri_csc1, fri_clc1,incl.precision =F)
    } else {
        lanEstim <- RaiseLgthBoot(lanEstim, fri_csc1, fri_clc1,
                                  incl.precision =F,B=15)
    }


    # totalW\$estim : total weight,
    aa <-lanEstim@totalW$estim

    aa$value<- aa$value/1000 # tons

    aa<- dplyr::rename(aa, "totalW"=value)

    # lenStruc\$estim : numbers-at-length estimates

    bb<- lanEstim@lenStruc$estim

    # define LCs and UNIT len
    UNIT <- as.character( unique(fri_csc@ca$lenCode[fri_csc@ca$spp==STK]) )

    if (UNIT %in% c("mm", "MM")& sel_spe$LC_RANGE[i]==10) {
        bb$length<-as.numeric(bb$length)/10
        UNIT1<-"CM"
    }

    if (UNIT %in% c("mm", "MM") & sel_spe$LC_RANGE[i]==1) {
        bb$length<-as.numeric(bb$length)
        UNIT1<- "MM"
    }

    if (UNIT %in% c("mm", "MM") & sel_spe$LC_RANGE[i]==5) {
        bb$length<-as.numeric(bb$length)/10
        UNIT1<-"CM"
    }

    if (UNIT %in% c("cm", "CM") ) {
        bb$length<-as.numeric(bb$length)
        UNIT1<- "CM"
    }

    if (UNIT %in% c("scm", "SCM") & sel_spe$LC_RANGE[i]==5) {
        bb$length<-as.numeric(bb$length)/10
        UNIT1<-"CM"
    }

    if (UNIT %in% c("scm", "SCM") & sel_spe$LC_RANGE[i]==10) {
        bb$length<-as.numeric(bb$length)/10
        UNIT1<-"CM"
    }



    bb$length<- plyr::round_any(as.numeric(bb$length),1,floor)

    bb$value<- bb$value/1000 # '000 ind

    bb<-aggregate(bb$value,by=list(bb$time,bb$space,bb$technical,bb$length),FUN="sum")

    colnames(bb)=c("time","space","technical","length","value")

    ab=  suppressMessages(dplyr::left_join(bb,aa ,by = c("time", "space", "technical")))

    ab<- suppressWarnings(ab %>% tidyr::separate(technical, c("gear","FISHERY","MESH_SIZE_RANGE"),
```

```
                        sep = "_",remove=F))

        ab<-cbind(ab[,c(1:5)],rep(NA,nrow(ab)),ab[,c(6:9)])
        #, "VL"
        colnames(ab)[6]="VL"

        ab$length<- as.numeric(as.character(ab$length))

        ab<- ab%>% dplyr::group_by(time,   space , gear  ,FISHERY, VL,MESH_SIZE_RANGE  ) %>%
            dplyr::mutate(minlc=min(length,na.rm=T),maxlc=max(length,na.rm=T))

        # matrix with all combinations of "time"   "space"  "gear"   "VL"
        # "length" ,"MESH_SIZE_RANGE"

        dt <- data.table::as.data.table(ab)

        dt[,c(1:7)][is.na(dt[,c(1:7)])]<- -1

        seq_l <-  seq(0, 99, by = 1) #

        dt$id<- paste(dt$time,dt$space,dt$gear,dt$FISHERY,dt$VL,
                      dt$MESH_SIZE_RANGE,sep=":")

        dt1<- dt[, list(length = seq_l), by = id]

        dt1<- suppressWarnings(dt1 %>% tidyr::separate(id, c("time", "space", "gear", "FISHERY","VL",
                                    "MESH_SIZE_RANGE"), sep = ":"))


        ab[is.na(ab)]<- -1
        class(dt1$VL)<-"numeric"

        dt2<- suppressMessages(dplyr::left_join(dt1,ab))

        dt2$stock<- STK

        ##

        dt3 <- reshape2::dcast(dt2,as.formula(paste(paste(names(dt2)[! names(dt2) %in%
                                                                    c("length","value")], collapse='+'),
"length", sep="~")),
                               value.var = "value")


        dt3=dt3[complete.cases(dt3[,c(7:9)]), ]

        dt3 <- suppressWarnings(dt3 %>% tidyr::separate(time, c("Year","Quarter")," - "))

        dt3$MESH_SIZE_RANGE<-as.character(dt3$MESH_SIZE_RANGE)


        # numbers at LC : NA-->0
        dt3<- dt3 %>% dplyr::mutate_at(dplyr::vars( -(Year:stock) ),
                                list(~ dplyr::if_else( is.na(.), 0, .) ) )


        LANDINGS <- data.frame(

            ID = NA ,
            COUNTRY = COUNTRY ,
            YEAR = YEAR ,
            QUARTER =dt3$Quarter,
            VESSEL_LENGTH = dt3$VL,
            GEAR = dt3$gear,
            MESH_SIZE_RANGE = dt3$MESH_SIZE_RANGE,
            FISHERY =  dt3$FISHERY ,
            AREA = sel_spe$GSA[i],
            SPECON = -1 ,
            SPECIES = STK ,
            LANDINGS = dt3$totalW ,
            UNIT = UNIT1
        )



        LANDINGS<- suppressMessages(dplyr::left_join(LANDINGS,dt3[,-c(1,3,8:11)],by=c( "QUARTER"  ="Quarter" ,
                                                              "GEAR"="gear" ,  "VESSEL_LENGTH" = "VL"  ,
                                                              "MESH_SIZE_RANGE","FISHERY" ))
)
        LANDINGS<-LANDINGS[,-14]
        # take care of number of Length classes (max is 100 acc. to DG MARE Med&BS template)
        zz<-dim(LANDINGS[-c(1:13)])[2]
        names(LANDINGS)[-c(1:13)]<- paste("LENGTHCLASS",seq(0,zz-1,1),sep="")


        if(zz>=100){

            LANDINGS$LENGTHCLASS100_PLUS<- rowSums(LANDINGS[,!1:113])
```

```
            LANDINGS<-LANDINGS %>% dplyr::select(ID:LENGTHCLASS99,LENGTHCLASS100_PLUS)
        }



        # FISHERY to DG MARE Med&BS codification
        LANDINGS$FISHERY <- RDBprocessing::fishery$SDEF_codification[match(LANDINGS$FISHERY ,
                                                    RDBprocessing::fishery$DGMARE_Med_BS_codification)]



        LANDINGS$SPECIES<-RDBprocessing::Annex17$Species[match(LANDINGS$SPECIES ,
                                                   RDBprocessing::Annex17$Scientific_name)]

        LANDINGS$MESH_SIZE_RANGE<-RDBprocessing::msr$DGMARE_Med_BS_codification_MSR[match(LANDINGS$MESH_SIZE_RANGE
,
                                                        RDBprocessing::msr$SDEF_codification_MSR)]

        # species to FAO three alpha code and set ID (COUNTRY, AREA, GEAR, VESSEL_LENGTH,
        # MESH_SIZE_RANGE,QUARTER, SPECIES)
        land.tab <-LANDINGS %>% dplyr::mutate(ID = paste(COUNTRY, AREA, GEAR,FISHERY, VESSEL_LENGTH,
                                            MESH_SIZE_RANGE,YEAR, QUARTER, SPECIES, sep = "_"))

        lan.temp2<-data.frame(matrix(nrow=0,ncol=length(header)))
        colnames(lan.temp2)=as.vector(header)

        lan.temp2<-rbind(lan.temp2,land.tab)


        lan.temp2[,-c(1:13)][is.na(lan.temp2[,-c(1:13)])] <- 0

    }


    #lan.temp2 <- lan.temp2 #[, 2:ncol(lan.temp2)]

if(nrow(lan.temp2[is.na(lan.temp2$VESSEL_LENGTH),])>0)
lan.temp2[is.na(lan.temp2$VESSEL_LENGTH),]$VESSEL_LENGTH="NA"


    return(lan.temp2)


    }



#' Discard by length (DISCARDS) table - MED & BS data call
#'
#' @param datacs Detailed data in RCG CS format
#' @param datacl Landings aggregated data in RCG CL format
#' @param datace Effort aggregated data in RDB CE format
(https://www.ices.dk/data/Documents/RDB/RDB%20Exchange%20Format.pdf)
#' @param verbose boolean. If TRUE a message is printed.
#' @return DISCARDS table
#' @export
#' @examples DISC_MEDBS(RDBprocessing::data_ex,RDBprocessing::data_exampleCL,RDBprocessing::ce_example)
#' @importFrom stats complete.cases
#' @import COSTeda
#' @importFrom COSTdbe dbeObject RaiseLgth
#' @importFrom COSTcore subsetSpp
#' @importFrom dplyr rename left_join bind_rows ungroup
#' @importFrom plyr round_any
#' @importFrom data.table as.data.table set setDF setDT
#' @importFrom tidyr separate
#' @importFrom magrittr %>%
#' @import reshape2
DISC_MEDBS<-function(datacs,datacl, datace, verbose=FALSE){

    # datacs=data_ex
    # datacl=data_exampleCL
# datace=ce_example

    datacs=check_cs_header(datacs)

    FISHERY<- GEAR<- ID<- LENGTHCLASS100_PLUS<- LENGTHCLASS99<- MESH_SIZE_RANGE<-QUARTER<- SPECIES<-
VESSEL_LENGTH<- VL<- Year<- fishery<- gear<- id <- space<- stock<- technical<- value<-.<-NULL

    fri_cs1<-RCGtoCOST_CS(datacs)
    fri_cl1<-RCGtoCOST_CL(datacl)
fri_ce1 = ceData(ce=datace)

    COUNTRY<-unique(fri_cs1@tr$landCtry)
    YEAR=unique(fri_cs1@tr$year)

    fri_strD1 <- strIni(timeStrata="quarter", techStrata = "foCatEu6",
spaceStrata = "area")

    fri_csv <- suppressWarnings(csDataVal(fri_cs1))
    fri_clv <- suppressWarnings(clDataVal(fri_cl1))
```

```
    fri_cev <- suppressWarnings(ceDataVal(fri_ce1))


    fri_csc <- suppressWarnings(csDataCons(fri_csv, fri_strD1))
    fri_clc <- suppressWarnings(clDataCons(fri_clv, fri_strD1))
    fri_cec <- suppressWarnings(ceDataCons(fri_cev, fri_strD1))


    header=c("ID","COUNTRY","YEAR","QUARTER","VESSEL_LENGTH","GEAR","MESH_SIZE_RANGE","FISHERY","AREA","SPECON",
"SPECIES","DISCARDS","UNIT",paste("LENGTHCLASS",seq(0,99),sep=""),"LENGTHCLASS100_PLUS")


    mat=
aggregate(datacs$Length_class,by=list(datacs$Flag_country,datacs$Year,datacs$Area,datacs$Species),FUN="length")
    colnames(mat)=c("COUNTRY","YEAR","AREA","SPECIES","NB")
    tab1=merge(mat,RDBprocessing::Annex17,by.x="SPECIES",by.y="Scientific_name")


    colnames(tab1)[6]="SPE"
    sel_spe<-tab1
    sel_spe$type<-"trip"

    colnames(sel_spe)[4]="GSA"
    sel_spe$FISHERY<-"-1"
    sel_spe$LC_RANGE=""
    sel_spe$landSpp=""

    if(nrow(sel_spe[sel_spe$UNIT=="cm",])>0) {    sel_spe[sel_spe$UNIT=="cm",]$LC_RANGE=10} else if
(nrow(sel_spe[sel_spe$UNIT=="mm",])>0){
        sel_spe[sel_spe$UNIT=="mm",]$LC_RANGE=1
    }

    i=1
        for (i in 1:dim(sel_spe)[1]) {

            STK<- sel_spe$SPECIES[i]

            AREA <- sel_spe$GSA[i]

            fri_csc1<- subset(fri_csc, space==sel_spe$GSA[i],table="ca",link=T)
            fri_clc1<- subset(fri_clc, space==sel_spe$GSA[i],table="cl")

            fri_cec1<- subset(fri_cec, space==sel_spe$GSA[i],table="ce")

            # The first step is to create the empty object, that will be given
            # the appropriate values for the descritor  fields.

            DIS_dbe <- dbeObject(desc= paste(STK, AREA,"Discards", sep="_"),
                                 species=STK,
                                 catchCat="DIS",
                                 strataDesc=fri_strD1,
                                 methodDesc="analytical"
            )

            if (sel_spe$type[i]=="landings" ) {

                DIS_dbe <- totVolume(DIS_dbe,fri_csc1,fri_cec1, fri_clc1,
                                     type=sel_spe$type[i],val="nAtLength",landSpp=sel_spe$landSpp[i])
            } else {
                DIS_dbe <- totVolume(DIS_dbe,fri_csc1,fri_cec1, type=sel_spe$type[i],
                                     val="nAtLength")
            }


            # totalW\$estim : total weight,
            aa <-DIS_dbe@totalW$estim

            aa$value<- aa$value/1000 # tons

            aa<- rename(aa, "totalW"=value)
            # lenStruc\$estim : numbers-at-length estimates,

            bb<- DIS_dbe@lenStruc$estim

            bb$length=as.numeric(bb$length)

            # define LCs and UNIT len
            UNIT <- as.character( unique(fri_csc@ca$lenCode[fri_csc@ca$spp==STK]) )

            if (UNIT %in% c("mm", "MM")& sel_spe$LC_RANGE[i]==10) {
                bb$length<-as.numeric(bb$length)/10
                UNIT1<-"CM"
            }

            if (UNIT %in% c("mm", "MM") & sel_spe$LC_RANGE[i]==1) {
                bb$length<-as.numeric(bb$length)
                UNIT1<- "MM"
            }

            if (UNIT %in% c("mm", "MM") & sel_spe$LC_RANGE[i]==5) {
```

```
            bb$length<-as.numeric(bb$length)/10
            UNIT1<-"CM"
        }

        if (UNIT %in% c("cm", "CM") ) {
            bb$length<-as.numeric(bb$length)
            UNIT1<- "CM"
        }

        if (UNIT %in% c("scm", "SCM") & sel_spe$LC_RANGE[i]==5) {
            bb$length<-as.numeric(bb$length)/10
            UNIT1<-"CM"
        }

        if (UNIT %in% c("scm", "SCM") & sel_spe$LC_RANGE[i]==10) {
            bb$length<-as.numeric(bb$length)/10
            UNIT1<-"CM"
        }

        bb$length<- plyr::round_any( bb$length,1,floor)

        bb$value<- bb$value/1000 # '000 ind

        bb<-aggregate(bb$value,by=list(bb$time,bb$space,bb$technical,bb$length),FUN="sum")
        colnames(bb)=c("time","space","technical","length","value")

        ab= suppressMessages(left_join(bb,aa ,by = c("time", "space", "technical")))

        ab<- suppressWarnings(ab %>% separate(technical, c("gear","FISHERY", "MESH_SIZE_RANGE"),
                              sep = "_",remove=T))


        ab<-cbind(ab[,c(1:5)],rep(NA,nrow(ab)),ab[,c(6:8)])
        #, "VL"
        colnames(ab)[6]="VL"

        ab$length<- as.numeric(as.character(ab$length))

        ab<- ab%>% group_by(time,   space , gear  ,FISHERY, VL,MESH_SIZE_RANGE  ) %>%
            mutate(minlc=min(length,na.rm=T),maxlc=max(length,na.rm=T))

        # matrix with all combinations of "time"   "space"  "gear"   "VL"
        # "length" ,"MESH_SIZE_RANGE"

        dt <- as.data.table(ab)

        dt[,c(1:7)][is.na(dt[,c(1:7)])]<- -1

        seq_l <-  seq(0, 99, by = 1) #

        dt$id<- paste(dt$time,dt$space,dt$gear,dt$FISHERY,dt$VL,
                      dt$MESH_SIZE_RANGE,sep=":")

        dt1<- dt[, list(length = seq_l), by = id]

        dt1<- suppressMessages(dt1 %>% separate(id, c("time", "space", "gear", "FISHERY","VL",
                                        "MESH_SIZE_RANGE"), sep = ":"))

        ab[is.na(ab)]<- -1
        class(dt1$VL)<-"numeric"

        # ab[,`1:6]: NA-->-1
        ab<- suppressWarnings(ab %>%ungroup()%>% mutate_at(vars(c(time:MESH_SIZE_RANGE) ),
                                           funs( ifelse( is.na(.), -1, .)) ) )


        dt2<- suppressWarnings(dplyr::left_join(dt1,ab))
        dt2$stock<- STK

        ##

        dt3 <- suppressWarnings(reshape2::dcast(dt2,as.formula(paste(paste(names(dt2)[! names(dt2) %in%
                                                                             c("length","value")],
collapse='+'), "length", sep="~")),
                                        value.var = "value"))

        dt3=dt3[complete.cases(dt3[,c(7:9)]), ]

        dt3 <- suppressWarnings(dt3 %>% separate(time, c("Year","Quarter")," - "))

        dt3$MESH_SIZE_RANGE<-as.character(dt3$MESH_SIZE_RANGE)


        # numbers at LC : NA-->0
        dt3<- suppressWarnings(dt3 %>% mutate_at(vars( -(Year:stock) ),
                                    funs( if_else( is.na(.), 0, .) ) ))

        DISCARDS <- data.frame(

            ID = NA ,
```

```
                COUNTRY = COUNTRY ,
                YEAR = YEAR ,
                QUARTER =dt3$Quarter,
                VESSEL_LENGTH = dt3$VL,
                GEAR = dt3$gear,
                MESH_SIZE_RANGE = dt3$MESH_SIZE_RANGE,
                FISHERY= dt3$FISHERY  ,
                AREA = sel_spe$GSA[i],
                SPECON = "",
                SPECIES = STK ,
                DISCARDS = dt3$totalW ,
                UNIT = UNIT1
            )

            DISCARDS<- suppressMessages(left_join(DISCARDS,dt3[,-c(1,3,8:11)],by=c( "QUARTER"  ="Quarter" ,
                                                                    "GEAR"="gear" ,  "VESSEL_LENGTH" = "VL"
,"MESH_SIZE_RANGE","FISHERY" )))

            # take care of number of Length classes (max is 100 acc. to JRC template)
            zz<-dim(DISCARDS[-c(1:13)])[2]
            names(DISCARDS)[-c(1:13)]<- paste("LENGTHCLASS",seq(0,zz-1,1),sep="")

            if(zz>=100){
                DISCARDS$LENGTHCLASS100_PLUS<- rowSums(DISCARDS[,!1:113],na.rm = T)
                DISCARDS<- suppressWarnings(DISCARDS %>% select(ID:LENGTHCLASS99,LENGTHCLASS100_PLUS))
            }

            # FISHERY to DG MARE Med&BS specification
            DISCARDS$FISHERY <- RDBprocessing::fishery$SDEF_codification[match(DISCARDS$FISHERY ,

RDBprocessing::fishery$DGMARE_Med_BS_codification)]
            DISCARDS$SPECIES<-RDBprocessing::Annex17$Species[match(DISCARDS$SPECIES ,
                                                                   RDBprocessing::Annex17$Scientific_name)]
            DISCARDS$VESSEL_LENGTH<-RDBprocessing::msr$SDEF_codification_MSR[match(DISCARDS$VESSEL_LENGTH ,

RDBprocessing::msr$DGMARE_Med_BS_codification_MSR)]

            # species to FAO three a?pha code and set ID (COUNTRY, AREA, GEAR,
            # VESSEL_LENGTH,  MESH_SIZE_RANGE,QUARTER, SPECIES)

            dis.tab <-DISCARDS %>% mutate(ID = paste(COUNTRY, AREA, GEAR,FISHERY, VESSEL_LENGTH,
                                                     MESH_SIZE_RANGE,YEAR, QUARTER, SPECIES, sep = "_"))

            dis.temp2<-data.frame(matrix(nrow=0,ncol=length(header)))
            colnames(dis.temp2)=as.vector(header)

           # lan.temp2<-rbind(lan.temp2,land.tab)

            dis.temp2<-rbind(dis.temp2,dis.tab)

            # col after 13: set -1 or NA to 0
            dis.temp2[,-c(1:13)][is.na(dis.temp2[,-c(1:13)])] <- 0

            dis.temp2<-setDT(dis.temp2)
            for (jj in c(14:114)) data.table::set(dis.temp2, i = which(dis.temp2[[jj]]==-1), j = jj, v = 0)

            dis.temp2<-setDF(dis.temp2)

        }
    dis.temp2[is.na(dis.temp2$VESSEL_LENGTH),]$VESSEL_LENGTH="NA"


    return(dis.temp2)

}


#' Catch at age (CATCH) table - MED & BS data call
#'
#' @param datacs Detailed data in RCG CS format
#' @param datacl Landings aggregated data in RCG CL format
#' @param datace Effort aggregated data in RDB CE format
(https://www.ices.dk/data/Documents/RDB/RDB%20Exchange%20Format.pdf)
#' @param verbose boolean. If TRUE a message is printed.
#' @return CATCH table
#' @export
#' @examples CATCH_MEDBS(RDBprocessing::data_ex,RDBprocessing::data_exampleCL,RDBprocessing::ce_example)
#' @importFrom stats complete.cases
#' @import COSTeda
#' @importFrom COSTdbe dbeObject RaiseLgth
#' @importFrom COSTcore subsetSpp
#' @importFrom dplyr rename left_join bind_rows ungroup distinct everything n_distinct
#' @importFrom plyr round_any
#' @importFrom data.table as.data.table set setDF setDT
#' @importFrom tidyr separate spread
#' @importFrom magrittr %>%
#' @import reshape2

CATCH_MEDBS<-function(datacs,datacl, datace, verbose=FALSE){
```

```
    datacs=check_cs_header(datacs)

    . <-.id <- id<- Dis<- FISHERY<- GEAR<- ID<- LANDINGS<- Lan<- MESH_SIZE_RANGE<- NO_AGE_MEASUREMENTS_DISCARDS<-
NO_AGE_MEASUREMENTS_LANDINGS <-   NO_LENGTH_MEASUREMENTS_DISCARDS <- NO_LENGTH_MEASUREMENTS_LANDINGS<-
NO_SAMPLES_DISCARDS <- NO_SAMPLES_LANDINGS<- QUARTER<- SPECIES<- VESSEL_LENGTH<- VL<- Year<- age<- area<-
catchCat<- foCatEu6<- funs<- gear<- id<-   lenCls<- meanW.at.age<- n <-n.at.age<-  space<- spp<- stock<-
technical<-    trpCode<- value<- vars<- year <- NULL

    if (nrow(datacs[which(as.numeric(datacs$Age)>=20),])>0)   datacs[datacs$Age>=20,]$Age=20


    datacs$Age=round(as.numeric(datacs$Age),0)


    #if(any(is.na(datacs[datacs$Age=="",]$Age))){
    datacs[is.na(datacs)]<-""
    #}

    datacs$Individual_weight=""
    datacs$fish_ID=""
    datacs$Maturity_Stage=""

    datacs2=aggregate(datacs$Number_at_length,
by=list(datacs$Sampling_type,datacs$Flag_country,datacs$Year,datacs$Trip_code,datacs$Harbour,datacs$Number_of_sets
_hauls_on_trip,datacs$Days_at_sea,datacs$Sampling_method,datacs$Aggregation_level,datacs$Station_number,datacs$Dur
ation_of_fishing_operation,datacs$Initial_latitude,datacs$Initial_longitude,datacs$Final_latitude,datacs$Final_lon
gitude,datacs$Depth_of_fishing_operation,datacs$Water_depth,datacs$Catch_registration,datacs$Species_registration,
datacs$Date,datacs$Area,datacs$Fishing_activity_category_National,datacs$Fishing_activity_category_European_lvl_6,
datacs$Species,datacs$Catch_category,datacs$Weight,datacs$Subsample_weight,datacs$Sex,datacs$Maturity_method,datac
s$Maturity_scale,datacs$Maturity_Stage,datacs$Ageing.method,datacs$Age,datacs$Length_code,datacs$Length_class,data
cs$Commercial_size_category_scale,datacs$Commercial_size_category,datacs$fish_ID,datacs$Individual_weight),
FUN="sum")


    datacs=datacs2[,c(1:35,40,36:39)]
    colnames(datacs)=colnames(RDBprocessing::data_ex)
    datacs=check_cs_header(datacs)


    if(any(is.na(datacs[datacs$Age=="",]$Age))){
    datacs[datacs$Age=="",]$Age<-NA
    }



    header=c("ID","COUNTRY","YEAR","QUARTER","VESSEL_LENGTH","GEAR","MESH_SIZE_RANGE","FISHERY","AREA","SPECON",
"SPECIES","LANDINGS", "DISCARDS", "NO_SAMPLES_LANDINGS","NO_LENGTH_MEASUREMENTS_LANDINGS",
"NO_AGE_MEASUREMENTS_LANDINGS", "NO_SAMPLES_DISCARDS","NO_LENGTH_MEASUREMENTS_DISCARDS",
"NO_AGE_MEASUREMENTS_DISCARDS", "NO_SAMPLES_CATCH", "NO_LENGTH_MEASUREMENTS_CATCH", "NO_AGE_MEASUREMENTS_CATCH",
"MIN_AGE", "MAX_AGE" ,
"AGE_0", "AGE_0_NO_LANDED", "AGE_0_MEAN_WEIGHT_LANDED", "AGE_0_MEAN_LENGTH_LANDED", "AGE_0_NO_DISCARD",
"AGE_0_MEAN_WEIGHT_DISCARD", "AGE_0_MEAN_LENGTH_DISCARD",
"AGE_1", "AGE_1_NO_LANDED", "AGE_1_MEAN_WEIGHT_LANDED", "AGE_1_MEAN_LENGTH_LANDED", "AGE_1_NO_DISCARD",
"AGE_1_MEAN_WEIGHT_DISCARD", "AGE_1_MEAN_LENGTH_DISCARD",
"AGE_2", "AGE_2_NO_LANDED", "AGE_2_MEAN_WEIGHT_LANDED", "AGE_2_MEAN_LENGTH_LANDED", "AGE_2_NO_DISCARD",
"AGE_2_MEAN_WEIGHT_DISCARD", "AGE_2_MEAN_LENGTH_DISCARD",
"AGE_3", "AGE_3_NO_LANDED", "AGE_3_MEAN_WEIGHT_LANDED", "AGE_3_MEAN_LENGTH_LANDED", "AGE_3_NO_DISCARD",
"AGE_3_MEAN_WEIGHT_DISCARD", "AGE_3_MEAN_LENGTH_DISCARD",
"AGE_4", "AGE_4_NO_LANDED", "AGE_4_MEAN_WEIGHT_LANDED", "AGE_4_MEAN_LENGTH_LANDED", "AGE_4_NO_DISCARD",
"AGE_4_MEAN_WEIGHT_DISCARD", "AGE_4_MEAN_LENGTH_DISCARD",
"AGE_5", "AGE_5_NO_LANDED", "AGE_5_MEAN_WEIGHT_LANDED", "AGE_5_MEAN_LENGTH_LANDED", "AGE_5_NO_DISCARD",
"AGE_5_MEAN_WEIGHT_DISCARD", "AGE_5_MEAN_LENGTH_DISCARD",
"AGE_6", "AGE_6_NO_LANDED", "AGE_6_MEAN_WEIGHT_LANDED", "AGE_6_MEAN_LENGTH_LANDED", "AGE_6_NO_DISCARD",
"AGE_6_MEAN_WEIGHT_DISCARD", "AGE_6_MEAN_LENGTH_DISCARD",
"AGE_7", "AGE_7_NO_LANDED", "AGE_7_MEAN_WEIGHT_LANDED", "AGE_7_MEAN_LENGTH_LANDED", "AGE_7_NO_DISCARD",
"AGE_7_MEAN_WEIGHT_DISCARD", "AGE_7_MEAN_LENGTH_DISCARD",
"AGE_8", "AGE_8_NO_LANDED", "AGE_8_MEAN_WEIGHT_LANDED", "AGE_8_MEAN_LENGTH_LANDED", "AGE_8_NO_DISCARD",
"AGE_8_MEAN_WEIGHT_DISCARD", "AGE_8_MEAN_LENGTH_DISCARD",
"AGE_9", "AGE_9_NO_LANDED", "AGE_9_MEAN_WEIGHT_LANDED", "AGE_9_MEAN_LENGTH_LANDED", "AGE_9_NO_DISCARD",
"AGE_9_MEAN_WEIGHT_DISCARD", "AGE_9_MEAN_LENGTH_DISCARD",
"AGE_10", "AGE_10_NO_LANDED", "AGE_10_MEAN_WEIGHT_LANDED", "AGE_10_MEAN_LENGTH_LANDED", "AGE_10_NO_DISCARD",
"AGE_10_MEAN_WEIGHT_DISCARD", "AGE_10_MEAN_LENGTH_DISCARD",
"AGE_11", "AGE_11_NO_LANDED", "AGE_11_MEAN_WEIGHT_LANDED", "AGE_11_MEAN_LENGTH_LANDED", "AGE_11_NO_DISCARD",
"AGE_11_MEAN_WEIGHT_DISCARD", "AGE_11_MEAN_LENGTH_DISCARD",
"AGE_12", "AGE_12_NO_LANDED", "AGE_12_MEAN_WEIGHT_LANDED", "AGE_12_MEAN_LENGTH_LANDED", "AGE_12_NO_DISCARD",
"AGE_12_MEAN_WEIGHT_DISCARD", "AGE_12_MEAN_LENGTH_DISCARD",
"AGE_13", "AGE_13_NO_LANDED", "AGE_13_MEAN_WEIGHT_LANDED", "AGE_13_MEAN_LENGTH_LANDED", "AGE_13_NO_DISCARD",
"AGE_13_MEAN_WEIGHT_DISCARD", "AGE_13_MEAN_LENGTH_DISCARD",
"AGE_14", "AGE_14_NO_LANDED", "AGE_14_MEAN_WEIGHT_LANDED", "AGE_14_MEAN_LENGTH_LANDED", "AGE_14_NO_DISCARD",
"AGE_14_MEAN_WEIGHT_DISCARD", "AGE_14_MEAN_LENGTH_DISCARD",
"AGE_15", "AGE_15_NO_LANDED", "AGE_15_MEAN_WEIGHT_LANDED", "AGE_15_MEAN_LENGTH_LANDED", "AGE_15_NO_DISCARD",
"AGE_15_MEAN_WEIGHT_DISCARD", "AGE_15_MEAN_LENGTH_DISCARD",
"AGE_16", "AGE_16_NO_LANDED", "AGE_16_MEAN_WEIGHT_LANDED", "AGE_16_MEAN_LENGTH_LANDED", "AGE_16_NO_DISCARD",
"AGE_16_MEAN_WEIGHT_DISCARD", "AGE_16_MEAN_LENGTH_DISCARD",
"AGE_17", "AGE_17_NO_LANDED", "AGE_17_MEAN_WEIGHT_LANDED", "AGE_17_MEAN_LENGTH_LANDED", "AGE_17_NO_DISCARD",
"AGE_17_MEAN_WEIGHT_DISCARD", "AGE_17_MEAN_LENGTH_DISCARD",
"AGE_18", "AGE_18_NO_LANDED", "AGE_18_MEAN_WEIGHT_LANDED", "AGE_18_MEAN_LENGTH_LANDED", "AGE_18_NO_DISCARD",
"AGE_18_MEAN_WEIGHT_DISCARD", "AGE_18_MEAN_LENGTH_DISCARD",
```

```
"AGE_19", "AGE_19_NO_LANDED", "AGE_19_MEAN_WEIGHT_LANDED", "AGE_19_MEAN_LENGTH_LANDED", "AGE_19_NO_DISCARD",
"AGE_19_MEAN_WEIGHT_DISCARD", "AGE_19_MEAN_LENGTH_DISCARD",
"AGE_20_PLUS", "AGE_20_PLUS_NO_LANDED", "AGE_20_PLUS_MEAN_WEIGHT_LANDED", "AGE_20_PLUS_MEAN_LENGTH_LANDED",
"AGE_20_PLUS_NO_DISCARD", "AGE_20_PLUS_MEAN_WEIGHT_DISCARD", "AGE_20_PLUS_MEAN_LENGTH_DISCARD")

    header2=c("ID","COUNTRY","YEAR","QUARTER","VESSEL_LENGTH","GEAR","MESH_SIZE_RANGE","FISHERY","AREA","SPECON",
"SPECIES","LANDINGS", "DISCARDS", "NO_SAMPLES_LANDINGS","NO_LENGTH_MEASUREMENTS_LANDINGS",
"NO_AGE_MEASUREMENTS_LANDINGS", "NO_SAMPLES_DISCARDS","NO_LENGTH_MEASUREMENTS_DISCARDS",
"NO_AGE_MEASUREMENTS_DISCARDS", "NO_SAMPLES_CATCH", "NO_LENGTH_MEASUREMENTS_CATCH", "NO_AGE_MEASUREMENTS_CATCH",
"MIN_AGE", "MAX_AGE" ,
             "AGE_0", "AGE_0_NO_LANDED", "AGE_0_MEAN_WEIGHT_LANDED", "AGE_0_MEAN_LENGTH_LANDED",
"AGE_0_NO_DISCARD", "AGE_0_MEAN_WEIGHT_DISCARD", "AGE_0_MEAN_LENGTH_DISCARD",
             "AGE_1", "AGE_1_NO_LANDED", "AGE_1_MEAN_WEIGHT_LANDED", "AGE_1_MEAN_LENGTH_LANDED",
"AGE_1_NO_DISCARD", "AGE_1_MEAN_WEIGHT_DISCARD", "AGE_1_MEAN_LENGTH_DISCARD",
             "AGE_2", "AGE_2_NO_LANDED", "AGE_2_MEAN_WEIGHT_LANDED", "AGE_2_MEAN_LENGTH_LANDED",
"AGE_2_NO_DISCARD", "AGE_2_MEAN_WEIGHT_DISCARD", "AGE_2_MEAN_LENGTH_DISCARD",
             "AGE_3", "AGE_3_NO_LANDED", "AGE_3_MEAN_WEIGHT_LANDED", "AGE_3_MEAN_LENGTH_LANDED",
"AGE_3_NO_DISCARD", "AGE_3_MEAN_WEIGHT_DISCARD", "AGE_3_MEAN_LENGTH_DISCARD",
             "AGE_4", "AGE_4_NO_LANDED", "AGE_4_MEAN_WEIGHT_LANDED", "AGE_4_MEAN_LENGTH_LANDED",
"AGE_4_NO_DISCARD", "AGE_4_MEAN_WEIGHT_DISCARD", "AGE_4_MEAN_LENGTH_DISCARD",
             "AGE_5", "AGE_5_NO_LANDED", "AGE_5_MEAN_WEIGHT_LANDED", "AGE_5_MEAN_LENGTH_LANDED",
"AGE_5_NO_DISCARD", "AGE_5_MEAN_WEIGHT_DISCARD", "AGE_5_MEAN_LENGTH_DISCARD",
             "AGE_6", "AGE_6_NO_LANDED", "AGE_6_MEAN_WEIGHT_LANDED", "AGE_6_MEAN_LENGTH_LANDED",
"AGE_6_NO_DISCARD", "AGE_6_MEAN_WEIGHT_DISCARD", "AGE_6_MEAN_LENGTH_DISCARD",
             "AGE_7", "AGE_7_NO_LANDED", "AGE_7_MEAN_WEIGHT_LANDED", "AGE_7_MEAN_LENGTH_LANDED",
"AGE_7_NO_DISCARD", "AGE_7_MEAN_WEIGHT_DISCARD", "AGE_7_MEAN_LENGTH_DISCARD",
             "AGE_8", "AGE_8_NO_LANDED", "AGE_8_MEAN_WEIGHT_LANDED", "AGE_8_MEAN_LENGTH_LANDED",
"AGE_8_NO_DISCARD", "AGE_8_MEAN_WEIGHT_DISCARD", "AGE_8_MEAN_LENGTH_DISCARD",
             "AGE_9", "AGE_9_NO_LANDED", "AGE_9_MEAN_WEIGHT_LANDED", "AGE_9_MEAN_LENGTH_LANDED",
"AGE_9_NO_DISCARD", "AGE_9_MEAN_WEIGHT_DISCARD", "AGE_9_MEAN_LENGTH_DISCARD",
             "AGE_10", "AGE_10_NO_LANDED", "AGE_10_MEAN_WEIGHT_LANDED", "AGE_10_MEAN_LENGTH_LANDED",
"AGE_10_NO_DISCARD", "AGE_10_MEAN_WEIGHT_DISCARD", "AGE_10_MEAN_LENGTH_DISCARD",
             "AGE_11", "AGE_11_NO_LANDED", "AGE_11_MEAN_WEIGHT_LANDED", "AGE_11_MEAN_LENGTH_LANDED",
"AGE_11_NO_DISCARD", "AGE_11_MEAN_WEIGHT_DISCARD", "AGE_11_MEAN_LENGTH_DISCARD",
             "AGE_12", "AGE_12_NO_LANDED", "AGE_12_MEAN_WEIGHT_LANDED", "AGE_12_MEAN_LENGTH_LANDED",
"AGE_12_NO_DISCARD", "AGE_12_MEAN_WEIGHT_DISCARD", "AGE_12_MEAN_LENGTH_DISCARD",
             "AGE_13", "AGE_13_NO_LANDED", "AGE_13_MEAN_WEIGHT_LANDED", "AGE_13_MEAN_LENGTH_LANDED",
"AGE_13_NO_DISCARD", "AGE_13_MEAN_WEIGHT_DISCARD", "AGE_13_MEAN_LENGTH_DISCARD",
             "AGE_14", "AGE_14_NO_LANDED", "AGE_14_MEAN_WEIGHT_LANDED", "AGE_14_MEAN_LENGTH_LANDED",
"AGE_14_NO_DISCARD", "AGE_14_MEAN_WEIGHT_DISCARD", "AGE_14_MEAN_LENGTH_DISCARD",
             "AGE_15", "AGE_15_NO_LANDED", "AGE_15_MEAN_WEIGHT_LANDED", "AGE_15_MEAN_LENGTH_LANDED",
"AGE_15_NO_DISCARD", "AGE_15_MEAN_WEIGHT_DISCARD", "AGE_15_MEAN_LENGTH_DISCARD",
             "AGE_16", "AGE_16_NO_LANDED", "AGE_16_MEAN_WEIGHT_LANDED", "AGE_16_MEAN_LENGTH_LANDED",
"AGE_16_NO_DISCARD", "AGE_16_MEAN_WEIGHT_DISCARD", "AGE_16_MEAN_LENGTH_DISCARD",
             "AGE_17", "AGE_17_NO_LANDED", "AGE_17_MEAN_WEIGHT_LANDED", "AGE_17_MEAN_LENGTH_LANDED",
"AGE_17_NO_DISCARD", "AGE_17_MEAN_WEIGHT_DISCARD", "AGE_17_MEAN_LENGTH_DISCARD",
             "AGE_18", "AGE_18_NO_LANDED", "AGE_18_MEAN_WEIGHT_LANDED", "AGE_18_MEAN_LENGTH_LANDED",
"AGE_18_NO_DISCARD", "AGE_18_MEAN_WEIGHT_DISCARD", "AGE_18_MEAN_LENGTH_DISCARD",
             "AGE_19", "AGE_19_NO_LANDED", "AGE_19_MEAN_WEIGHT_LANDED", "AGE_19_MEAN_LENGTH_LANDED",
"AGE_19_NO_DISCARD", "AGE_19_MEAN_WEIGHT_DISCARD", "AGE_19_MEAN_LENGTH_DISCARD",
             "AGE_20", "AGE_20_NO_LANDED", "AGE_20_MEAN_WEIGHT_LANDED", "AGE_20_MEAN_LENGTH_LANDED",
"AGE_20_NO_DISCARD", "AGE_20_MEAN_WEIGHT_DISCARD", "AGE_20_MEAN_LENGTH_DISCARD")


    mat=
aggregate(datacs$Length_class,by=list(datacs$Flag_country,datacs$Year,datacs$Area,datacs$Species),FUN="length")

    colnames(mat)=c("COUNTRY","YEAR","AREA","SPECIES","NB")
    tab1=merge(mat,RDBprocessing::Annex17,by.x="SPECIES",by.y="Scientific_name")

    colnames(tab1)[6]="SPE"
    sel_spe<-tab1
    sel_spe$type<-"trip"

    colnames(sel_spe)[4]="GSA"
    sel_spe$FISHERY<-"-1"
    sel_spe$LC_RANGE=""
    sel_spe$landSpp=""

    if(nrow(sel_spe[sel_spe$UNIT=="cm",])>0) {    sel_spe[sel_spe$UNIT=="cm",]$LC_RANGE=10} else if
(nrow(sel_spe[sel_spe$UNIT=="mm",])>0){
        sel_spe[sel_spe$UNIT=="mm",]$LC_RANGE=1
    }
    sel_spe$typeALK="fixed"
    sel_spe$type="trip"
    sel_spe$lanEstim_methodDesc="analytical"
    sel_spe$methodDesc_LAN.age.wght="analytical"
    sel_spe$methodDesc_LAN.len.age="analytical"
    sel_spe$methodDesc_LAN.age.wght="analytical"
    sel_spe$methodDesc_DIS.age.wght="analytical"
    sel_spe$methodDesc_DIS.len.wght="analytical"
    sel_spe$methodDesc_DIS.len.age<-"analytical"
    sel_spe$mcrs="."
    sel_spe$specon_catch=""
    sel_spe$landSpp=""

  sel_spe$adjust_L.w.a <-sel_spe$adjust_D.w.a<-sel_spe$adjust_L.len.a<-sel_spe$adjust_D.len.a<-TRUE

  # get col w. all=NA by spp
  #aa=fri_csc@ca %>% group_by(spp) %>% summarize_all(~all(is.na(.)))
```

```
  # to make it run for len if age=NA, impute age =="-1"
  #fri_csc@ca$age[fri_csc@ca$spp %in% aa$spp[aa$age==TRUE]]=-1


  ### error if sp-quarter-gear-VL not found in both CS and CL

  i=1
  for (i in 1:dim(sel_spe)[1]) {


      STK<- sel_spe$SPECIES[i]

      AREA <- sel_spe$GSA[i]

      datacs_temp=datacs[datacs$Area==AREA & datacs$Species==STK,]
      datacl_temp=datacl[datacl$area==AREA & datacl$species==STK,]
      datace_temp=datace[datace$area==AREA,]


      fri_cs1<-RCGtoCOST_CS(datacs_temp)
fri_cl1<-RCGtoCOST_CL(datacl_temp)
      fri_ce1 = ceData(ce=datace_temp)


      COUNTRY<-unique(fri_cs1@tr$landCtry)
      YEAR=unique(fri_cs1@tr$year)

      fri_strD1 <- strIni(timeStrata="quarter", techStrata = "foCatEu6",
                          spaceStrata = "area")

      fri_csv <- csDataVal(fri_cs1)
      fri_clv <- clDataVal(fri_cl1)
      fri_cev <- ceDataVal(fri_ce1)

fri_csc1 <- suppressWarnings( csDataCons(fri_csv, fri_strD1) )
      fri_clc1 <- suppressWarnings( clDataCons(fri_clv, fri_strD1) )
      fri_cec1 <- suppressWarnings( ceDataCons(fri_cev, fri_strD1) )

      fri_csc1@ca$age[fri_csc1@ca$age==-1]=NA
# da sostituire
      #fri_csc1<- subset(fri_csc, space==sel_spe$GSA[i],table="ca",link=T)
      #fri_clc1<- subset(fri_clc, space==sel_spe$GSA[i],table="cl")
      #fri_cec1<- subset(fri_cec, space==sel_spe$GSA[i],table="ce")


      str_ca=unique(fri_csc1@ca$time)
      str_cl=unique(fri_clc1@cl$time)
      str_ce=unique(fri_cec1@ce$time)

intersection=intersect(intersect(str_ca,str_cl),str_ce)
      years=substr(intersection,1,4)

if (as.character(sel_spe$YEAR[i]) %in% years){

      #fri_csc1<- subset(fri_csc, space==sel_spe$GSA[i], time==intersection,table="ca",link=T)
      #fri_clc1<- subset(fri_clc, space==sel_spe$GSA[i],time== intersection,table="cl")

      #fri_cec1<- subset(fri_cec, space==sel_spe$GSA[i],time== intersection,table="ce")

      # Estimating age structure LAN -  ----------------------------------


      lanEstim <-
          dbeObject(
              desc = paste(STK, "Landings.str", sep="_"),
              species = STK,
              catchCat = "LAN",
              strataDesc = fri_strD1,
              methodDesc = sel_spe$lanEstim_methodDesc[i]
          )

      # length str LAN

      if ( sel_spe$lanEstim_methodDesc[i]=="analytical"){
          lanEstim <- RaiseLgth(lanEstim, fri_csc1, fri_clc1,incl.precision =F)
      } else {
          lanEstim <- RaiseLgthBoot(lanEstim, fri_csc1, fri_clc1,incl.precision =F,B=15)
      }

      # subset lenstruc for MCRS

      if (!is.na(sel_spe$mcrs[i]) & sel_spe$mcrs[i]!=".") {
          lanEstim@lenStruc$estim =
              lanEstim@lenStruc$estim[as.numeric(lanEstim@lenStruc$estim$length) <= sel_spe$mcrs[i],]
          lanEstim@lenVar = lanEstim@lenVar[as.numeric(lanEstim@lenVar$length) <= sel_spe$mcrs[i],]
      }

      #########>>>>>>>>>>>>>> Estimation of total numbers-at-age from market sampling
```

```
      lanEstim <- RaiseAge(lanEstim, csObject=fri_csc1, fri_clc1, type = sel_spe$typeALK[i],
                           strataDesc = fri_strD1)

      # Estimating age structure DIS - ------------------------------------------------

      DIS_dbe <- dbeObject(desc= paste(STK, AREA,"Discards", sep="_"),
                           species=STK,
                           catchCat="DIS",
                           strataDesc=fri_strD1,
                           methodDesc="analytical")

      # discards raising

      if (sel_spe$type[i]=="landings" ) {
          DIS_dbe <- totVolume(DIS_dbe,fri_csc1,fri_cec1, fri_clc1,
                               type=sel_spe$type[i],landSpp=sel_spe$landSpp[i])
      } else {
          DIS_dbe <- totVolume(DIS_dbe,fri_csc1,fri_cec1, type=sel_spe$type[i])
      }

      DIS_dbe <- suppressWarnings(RaiseAge(DIS_dbe, fri_csc1, fri_clc1,type = sel_spe$typeALK[i],
                           strataDesc = fri_strD1))

      # CATCH1 :  cols ID: MAX_AGE ----------------------------------------------------------

      # NO SAMPLES ------------------------------------------------------------------

      ### No Samples == No trips (see ANNEX2- DG MARE Med&BS data call spec.).
      # Note: dbe estimates n.samples as trips*fo , thus the estimation must be based on HL

       newhl<-mergecsData(fri_cs1)@hl %>%
            #rename("space"=area, "technical"=foCatEu6) %>%
            mutate(time=paste(year, quarter, sep=" - ")) %>% filter( spp==STK)
      colnames(newhl)[18]="space"
      colnames(newhl)[22]="technical"
      # newhl<-mergecsData(fri_cs1)@tr %>%
      #     rename("space"=area, "technical"=foCatEu6) %>%
      #     mutate(time=paste(year, quarter, sep=" - ")) #%>% filter( spp==STK)
      #

      if (!is.na(sel_spe$mcrs[i]) & sel_spe$mcrs[i]!=".") { # MCRS

          no.samples<- suppressWarnings( data.frame(newhl) %>% filter(lenCls<=sel_spe$mcrs[i]) %>%
              dplyr::group_by(time,space,technical,catchCat)%>%
              summarize(value=n_distinct(trpCode)))

          no.samples<- no.samples%>% tidyr::spread(catchCat,value)

          L.no.samples<- suppressWarnings(no.samples%>% select(-Dis) ) #%>% rename("value"=Lan))
          D.no.samples<- suppressWarnings(no.samples%>% select(-Lan)%>% rename("value"=Dis))

      } else {

           no.samples<- suppressMessages( data.frame(newhl) %>%
               dplyr::group_by(time,space,technical,catchCat)%>%
                   dplyr::summarize(value=n_distinct(trpCode)) )


          no.samples<- no.samples%>% tidyr::spread(catchCat,value)

          L.no.samples<- suppressWarnings(no.samples%>% dplyr::select(-Dis) )
          colnames(L.no.samples)[4]="value"
                                       #%>% rename("value"=Lan))
          D.no.samples<- suppressWarnings(no.samples%>% select(-Lan))
colnames(D.no.samples)[4]="value"
#rename("value"=Dis))
      }

      # MCRS : no age and len measur. for LAN < MCRS----------------------

      if (!is.na(sel_spe$mcrs[i]) & sel_spe$mcrs[i]!="."){

          # No age measurements LAN

          newca <- mergecsData(fri_cs1)@ca %>%
              rename("space"=area, "technical"=foCatEu6) %>%
              mutate(time=paste(year, quarter, sep=" - ")) %>% filter( spp==STK)

          no.age.meas.lan.mcrs<- suppressWarnings(  data.frame(newca) %>%
              filter(lenCls<=sel_spe$mcrs[i]&!is.na(age)&catchCat=="LAN" )%>%
              dplyr::group_by(time,space,technical)%>% summarize(value=n()) )

          # remove technical if all ==NA
          no.age.meas.lan.mcrs<- no.age.meas.lan.mcrs[,!apply(is.na(no.age.meas.lan.mcrs), 2, all)]

          # No len measurements LAN

          no.len.meas.lan.mcrs<- suppressWarnings(  data.frame(newca) %>%
              filter(lenCls<=sel_spe$mcrs[i]&!is.na(lenCls)&catchCat=="LAN")%>%
              dplyr::group_by(time,space,technical)%>% summarize(value=n()) )
```

```
      }

      ### end

      # separate merge for age samples: may not have technical strata (use : space, time)

      if (!is.na(sel_spe$mcrs[i]) & sel_spe$mcrs[i]!=".") {
          # if mcrs use no.age.meas.lan.mcrs
          list.age.smp<-list(no.age.meas.lan.mcrs,DIS_dbe@nMeas$age)
      } else{
          list.age.smp<-list(lanEstim@nMeas$age,DIS_dbe@nMeas$age)
      }

      names(list.age.smp)<-c("L.nmeas.age", "D.nmeas.age")
      list.age.smp.merge = data.table::rbindlist(list.age.smp,id=T)
      age.smp= tidyr::spread(list.age.smp.merge,key=.id,value=value)

      # list all remaining output tables (excl. age meas & no samples)

      if (!is.na(sel_spe$mcrs[i]) & sel_spe$mcrs[i]!=".") { # if mcrs use no.len.meas.lan.mcrs

          list2<- list(no.len.meas.lan.mcrs,DIS_dbe@nMeas$len,lanEstim@totalN$estim,
                       lanEstim@totalW$estim ,DIS_dbe@totalW$estim ,DIS_dbe@totalN$estim,
                       L.no.samples,D.no.samples)

      } else {

          list2<- list(lanEstim@nMeas$len,DIS_dbe@nMeas$len,
                       lanEstim@totalN$estim,lanEstim@totalW$estim,
                       DIS_dbe@totalW$estim ,DIS_dbe@totalN$estim,
                       L.no.samples,D.no.samples)
      }

      names(list2)<-c("LnMeas.len","DnMeas.len","LtotalN","LtotalW","totalWDIS",
                      "totalNDIS","L.no.samples","D.no.samples")

      list3=append(list.age.smp,list2)
      list..merge = data.table::rbindlist(list3,id=T,fill=T)

      all.merge= tidyr::spread(list..merge,key=.id,value=value)

      all.merge <- suppressWarnings(all.merge%>% mutate( "stock"=STK) %>% select(stock,everything()))

      # delete rows (age...) with no "technical": a number should be given only
      # if it relates to this fishery only
      # (see DG MARE Med&BS Data Call specificatin - Annex 2 - Catch)

      all.merge<-  all.merge[complete.cases(all.merge$technical), ]

      aa.len <- all.merge
      aa.len$totalN=all.merge$LtotalN/1000 # '000 ind
      aa.len$totalW=all.merge$LtotalW/1000 # tons

      aa.len$totalNDIS=all.merge$totalNDIS/1000 # '000 ind
      aa.len$totalWDIS=all.merge$totalWDIS/1000 # tons

      # AgeStruc : n.at.age LAN / DIS ----------------------------------------

      # landings
      bb<-lanEstim@ageStruc$estim

      bb$value=bb$value/1000 # '000 ind
      colnames(bb)[5]="n.at.age"
      #bb<-rename(bb, "n.at.age"=value)

      # discards
      bbd<-DIS_dbe@ageStruc$estim

      bbd$value=bbd$value/1000 # '000 ind
colnames(bbd)[5]="DIS.n.at.age"
      #bbd<-rename(bbd, "DIS.n.at.age"=value)

      ab<- suppressMessages(suppressMessages(left_join(bb,bbd) %>% left_join(aa.len)))


      ab<- suppressWarnings(ab %>% tidyr::separate(technical, c("gear","FISHERY","MESH_SIZE_RANGE"),
                           sep = "_"))

      # min age/ max age-------------------------------------------------------------
      ab<-cbind(ab[,c(1:5)],rep(NA,nrow(ab)),ab[,c(6:21)])
      #, "VL"
      colnames(ab)[6]="VL"

      ab$age <- as.numeric(as.character(ab$age))

      ab$age<- as.numeric(as.character(ab$age))
      ab <- ab%>% group_by(time,   space , gear,FISHERY  , VL ,MESH_SIZE_RANGE ) %>%
          mutate(minage=min(age,na.rm=T),maxage=max(age,na.rm=T))
```

```
      ab <- ab %>% tidyr::separate(time, c("Year","Quarter")," - ",remove=F)

      # ### >>>>>>>>>. catch1: info by row -------------------------------------


      catch1= data.frame(
          ID = NA ,
          COUNTRY =COUNTRY ,
          YEAR = sel_spe$YEAR[i] ,
          QUARTER = ab$Quarter ,
          VESSEL_LENGTH = ab$VL ,
          GEAR = ab$gear ,
          MESH_SIZE_RANGE = ab$MESH_SIZE_RANGE ,
          FISHERY = ab$FISHERY[i] ,
          AREA = AREA ,
SPECON = sel_spe$specon_catch[i] ,
SPECIES = STK ,
          LANDINGS = ab$totalW , # MCRS: meanW.at.age * n.at.age
          DISCARDS = ab$totalWDIS,
          NO_SAMPLES_LANDINGS =ab$L.no.samples, # = TRIPS
          NO_LENGTH_MEASUREMENTS_LANDINGS = ab$LnMeas.len ,
          NO_AGE_MEASUREMENTS_LANDINGS = ab$L.nmeas.age ,
          NO_SAMPLES_DISCARDS = ab$D.no.samples , # = TRIPS
          NO_LENGTH_MEASUREMENTS_DISCARDS = ab$DnMeas.len ,
          NO_AGE_MEASUREMENTS_DISCARDS = ab$D.nmeas.age,
          NO_SAMPLES_CATCH = 0,
          NO_LENGTH_MEASUREMENTS_CATCH = 0,
          NO_AGE_MEASUREMENTS_CATCH = 0 ,
          MIN_AGE = ab$minage ,
          MAX_AGE = ab$maxage
      )

      # if mcrs delete LANDINGS: estimated below
      if (!is.na(sel_spe$mcrs[i]) & sel_spe$mcrs[i]!="."){
          catch1<- suppressWarnings(catch1 %>% select(-LANDINGS))
      }

      # ### >>>>>>>>>. catch1: no samples & no meas.  ----------------------------

      # NO_SAMPLES_CATCH - NO_LENGTH_MEASUREMENTS_CATCH -NO_AGE_MEASUREMENTS_CATCH

      catch1 <- catch1 %>% mutate(
          NO_SAMPLES_CATCH = rowSums( cbind (NO_SAMPLES_LANDINGS , NO_SAMPLES_DISCARDS),
                                      na.rm=TRUE),
          NO_LENGTH_MEASUREMENTS_CATCH = rowSums( cbind (NO_LENGTH_MEASUREMENTS_DISCARDS ,
                                                         NO_LENGTH_MEASUREMENTS_LANDINGS),
                                                  na.rm=TRUE),

          NO_AGE_MEASUREMENTS_CATCH=rowSums( cbind (NO_AGE_MEASUREMENTS_DISCARDS,
                                                    NO_AGE_MEASUREMENTS_LANDINGS),
                                             na.rm=TRUE)) %>% distinct()

      # ############ >>>>>> LAN- DIS n.at.age -------------------------------------


      # matrix with all combinations of "time" "space"  "gear" "VL"
      # "MESH_SIZE_RANGE" "length"

      ab[,c(1:8)][is.na(ab[,c(1:8)])]<--1

      dt <- data.table::as.data.table(ab )

      seq_l <-  try(seq(0, 20, by = 1),silent=T)
      if(class(seq_l)=="try-error"){seq_l=-1}

      dt$id<- paste(dt$time,dt$space,dt$gear,dt$FISHERY,dt$VL,dt$MESH_SIZE_RANGE,
                    sep=":")

      dt1<- dt[, list(age = seq_l), by = id]
      dt1<- dt1 %>% tidyr::separate(id, c("time", "space", "gear", "FISHERY",
                                    "VL","MESH_SIZE_RANGE"), sep = ":")
      class(dt1$VL)<-"numeric"
      #dt1[is.na(dt1$VL),]$VL=-1
      dt2=suppressMessages(suppressWarnings(left_join(dt1,ab %>% select("time" , "space", "gear" , "FISHERY",
                                     "VL" , "MESH_SIZE_RANGE","age","n.at.age",
                                     "DIS.n.at.age","stock" ))))

      dt2$stock=STK

      #  MEAN LENGTH LAND, MEAN WEIGHT LAND -------------------------------------

      #####>>>>>>>>>>> LAN weight at age!!

      wtEstim_An <-
          dbeObject(
              desc = "Weights at age",
              species = STK,
              catchCat = "LAN",
              param = "weight",
```

```
            strataDesc = fri_strD1, # strBP,
            methodDesc = sel_spe$methodDesc_LAN.age.wght[i]
        )



    if(sel_spe$methodDesc_LAN.age.wght[i]=="analytical") {
        wtEstim_An <- bpEstim(wtEstim_An, fri_csc1,adjust = sel_spe$adjust_L.w.a[i])
    } else{
        wtEstim_An <- bpBoot(wtEstim_An, fri_csc1, adjust = sel_spe$adjust_L.w.a[i])
    }

    # # LAN mean weight at age ------------------------------------------------

    cc=wtEstim_An@ageStruc$estim

    cc$value<- cc$value/1000 # g ->kg
colnames(cc)[5]="meanW.at.age"
    cc$age=as.numeric(as.character(cc$age))

    #cc=rename(cc, "meanW.at.age"=value)
    cc= suppressWarnings( cc %>% tidyr::separate(technical, c("gear", "FISHERY","MESH_SIZE_RANGE"),
                      sep = "_"))
    cc<-cbind(cc[,c(1:5)],rep(NA,nrow(cc)),cc[,c(6:7)])
    #, "VL"
    colnames(cc)[6]="VL"

    # LAN length at age -----------------------------------------------------

    LEstim_An <-
        dbeObject(
            desc = "Length at age",
            species = STK,
            catchCat = "LAN",
            param = "length",
            strataDesc = fri_strD1, # strBP,
            methodDesc = sel_spe$methodDesc_LAN.len.age[i]
        )


    if(sel_spe$methodDesc_LAN.len.age[i]=="analytical") {
        LEstim_An <-  bpEstim(LEstim_An, fri_csc1, adjust = sel_spe$adjust_L.len.a[i])

    } else{
        LEstim_An <-  bpBoot(LEstim_An, fri_csc1, adjust = sel_spe$adjust_L.len.a[i])
    }

    # LAN mean length at age ------------------------------------------------

    ff=LEstim_An@ageStruc$estim

    UNIT <- unique(fri_csc1@ca$lenCode[fri_csc1@ca$spp==STK])
    if (UNIT=="mm" | UNIT=="MM"){
        ff$value <- ff$value/10  # mm-> cm
    }


    ff$age=as.numeric(as.character(ff$age))
    colnames(ff)[5]="meanL.at.age"
    #ff=rename(ff, "meanL.at.age"=value)
    ff= suppressWarnings( ff %>% tidyr::separate(technical, c("gear","FISHERY","MESH_SIZE_RANGE"),
                      sep = "_"))
    ff<-cbind(ff[,c(1:5)],rep(NA,nrow(ff)),ff[,c(6:7)])
    #, "VL"
    colnames(ff)[6]="VL"

    # MEAN LENGTH DISCARD, MEAN WEIGHT DISCARD ------------------------------

    # DIS weight at age -----------------------------------------------------


    DwtEstim_An <-
        dbeObject(
            desc = "Weights at age",
            species = STK,
            catchCat = "DIS",
            param = "weight",
            strataDesc = fri_strD1, # strBP,
            methodDesc = sel_spe$methodDesc_DIS.age.wght[i]
        )


    if(sel_spe$methodDesc_DIS.age.wght[i]=="analytical") {
        DwtEstim_An <- bpEstim(DwtEstim_An, fri_csc1, adjust = sel_spe$adjust_D.w.a[i])
    } else{
        DwtEstim_An <- bpBoot(DwtEstim_An, fri_csc1, adjust = sel_spe$adjust_D.w.a[i])
    }

    # DIS mean weight at age
    ccD=DwtEstim_An@ageStruc$estim
```

```
      ccD$value <- ccD$value/1000 # g -> kg

      ccD$age=as.numeric(as.character(ccD$age))
      colnames(ccD)[5]="DmeanW.at.age"
      #ccD=rename(ccD, "DmeanW.at.age"=value)
      ccD=suppressWarnings(  ccD %>% tidyr::separate(technical, c("gear", "FISHERY","MESH_SIZE_RANGE"),
                           sep = "_") )

      ccD<-cbind(ccD[,c(1:5)],rep(NA,nrow(ccD)),ccD[,c(6:7)])
      #, "VL"
      colnames(ccD)[6]="VL"
      # DIS length at age ------------------------------------------------------------

      DLEstim_An <-
          dbeObject(
              desc = " Length at age",
              species = STK,
              catchCat = "DIS",
              param = "length",
              strataDesc = fri_strD1, # strBP,
              methodDesc = sel_spe$methodDesc_DIS.len.age[i]
          )

      if(sel_spe$methodDesc_DIS.len.age[i]=="analytical") {
          DLEstim_An <- bpEstim(DLEstim_An, fri_csc1, adjust = sel_spe$adjust_D.len.a[i])
      } else{
          DLEstim_An <- bpBoot(DLEstim_An, fri_csc1, adjust = sel_spe$adjust_D.len.a[i])
      }

      ####### >> DIS mean length at age
      ffD=DLEstim_An@ageStruc$estim

      UNIT <- unique(fri_csc1@ca$lenCode[fri_csc1@ca$spp==STK])
      if (UNIT=="mm" |UNIT== "MM"){
          ffD$value <- ffD$value/10  # mm-> cm
      }

      ffD$age=as.numeric(as.character(ffD$age))
      colnames(ffD)[5]="DmeanL.at.age"
      #ffD=rename(ffD, "DmeanL.at.age"=value)
      ffD=suppressWarnings(  ffD %>% tidyr::separate(technical, c("gear", "FISHERY","MESH_SIZE_RANGE"),
                           sep = "_") )

      ffD<-cbind(ffD[,c(1:5)],rep(NA,nrow(ffD)),ffD[,c(6:7)])
      #, "VL"
      colnames(ffD)[6]="VL"

      # ## combine: w.age, l.age, no.age (LAN- DIS)... to get CATCH cols --------

      dt2[dt2$VL==-1,]$VL=NA


      dt2[which(is.na(dt2[,c("n.at.age")])),c("n.at.age")]<--1
      dt2[which(is.na(dt2[,c("DIS.n.at.age")])),c("DIS.n.at.age")]<--1

      cc[which(is.na(cc[,c("meanW.at.age")])),c("meanW.at.age")]<--1

      ff[which(is.na(ff[,c("meanL.at.age")])),c("meanL.at.age")]<--1

      ccD[which(is.na(ccD[,c("DmeanW.at.age")])),c("DmeanW.at.age")]<--1

      ffD[which(is.na(ffD[,c("DmeanL.at.age")])),c("DmeanL.at.age")]<--1

      l3=list(dt2,cc,ff,ccD,ffD)



      l3=lapply(l3, function(x){ x[,c(1:8)][is.na(x[,c(1:8)])]<- -1;return(x)})


      cfdt2=Reduce(function(x, y) merge(x, y, by = c("time", "space",  "gear" ,
                                                     "FISHERY","VL","MESH_SIZE_RANGE" ,"age"  ),all.x=T), l3)

# c'era un try
      dt3 = try(data.table::dcast(data.table::setDT(distinct(cfdt2)),
                                  time + space + gear +FISHERY+ VL+MESH_SIZE_RANGE ~ age,
                                  value.var = c("n.at.age","DIS.n.at.age","meanW.at.age",
                                                "meanL.at.age","DmeanW.at.age","DmeanL.at.age",
                                                "age" )),silent=T)

      # if mcrs
      # Landings= meanW.at.age*n.at.age
      if (!is.na(sel_spe$mcrs[i]) & sel_spe$mcrs[i]!="."){

          # tonnes
          landings <- suppressWarnings(cfdt2 %>% mutate(LANDINGS= meanW.at.age*n.at.age) %>%
              select(time, space, gear, FISHERY, VL, MESH_SIZE_RANGE,LANDINGS))

          landings <- landings[!is.na(landings$LANDINGS ), ]
```

```
          dt3<- suppressMessages(left_join(dt3,landings))

      }

      ## rename col to match CATCH

      names(dt3)[grep("DIS.n.at.age", names(dt3)) ]<-paste("AGE",
                                                           0:(length(grep("DIS.n.at.age", names(dt3)))-
1),"NO_DISCARD",sep="_")

      names(dt3)[grep("n.at.age", names(dt3)) ]<-paste("AGE",
                                                       0:(length(grep("n.at.age", names(dt3)))-
1),"NO_LANDED",sep="_")

      names(dt3)[grep("DmeanW.at.age", names(dt3)) ]<-paste("AGE",
                                                            0:(length(grep("DmeanW.at.age", names(dt3)))-
1),"MEAN_WEIGHT_DISCARD",
                                                            sep="_")

      names(dt3)[grep("DmeanL.at.age", names(dt3)) ]<-paste("AGE",
                                                            0:(length(grep("DmeanL.at.age", names(dt3)))-
1),"MEAN_LENGTH_DISCARD",
                                                            sep="_")

      names(dt3)[grep("meanW.at.age", names(dt3)) ]<-paste("AGE",
                                                           0:(length(grep("meanW.at.age", names(dt3)))-
1),"MEAN_WEIGHT_LANDED",
                                                           sep="_")

      names(dt3)[grep("meanL.at.age", names(dt3)) ]<-paste("AGE",
                                                           0:(length(grep("meanL.at.age", names(dt3)))-
1),"MEAN_LENGTH_LANDED",
                                                           sep="_")

      names(dt3)[grep("age.1", names(dt3))]<-paste("AGE",
                                                   0:(length(grep("age.1", names(dt3)))-1),sep="_")

      dt3 <- dt3 %>% tidyr::separate(time, c("Year","Quarter")," - ",remove=T)

      dt3<-data.frame(dt3)

      dt3[is.na(dt3)]<--1

      ###########

      # FINAL CATCH TAB ------------------------------------------------------

      catch1<- catch1 %>% mutate_at(vars( c(ID:SPECIES) ),
                                    list(~ ifelse( is.na(.), -1, .) ) )


      dt3<- dt3 %>% mutate_at(vars( c(Year:MESH_SIZE_RANGE) ),
                              list(~ ifelse( is.na(.), -1, .) ) )

      #which(is.na(dt3))<--1


      catch1$YEAR=as.character(catch1$YEAR)
      catch1$FISHERY=as.character(catch1$FISHERY)

colnames(dt3)[c(1,2,3,4,6)]=c("YEAR","QUARTER","AREA","GEAR","VESSEL_LENGTH")

      catch.tab <- suppressMessages( left_join(catch1,dt3,by=c( "QUARTER" ,  "YEAR",  "AREA", "GEAR" ,
"VESSEL_LENGTH", "FISHERY", "MESH_SIZE_RANGE")) )

      # FISHERY to DG MARE Med&BS codification
      catch.tab$FISHERY <- RDBprocessing::fishery$SDEF_codification[match(catch.tab$FISHERY ,
                                              RDBprocessing::fishery$DGMARE_Med_BS_codification)]

      # species to FAO three alpha code and set ID (COUNTRY, AREA, GEAR, VESSEL_LENGTH,
      # MESH_SIZE_RANGE,QUARTER, SPECIES)
      catch.tab <- catch.tab %>%
          mutate(SPECIES=sel_spe$SPE[match(SPECIES,sel_spe$SPECIES)],
                 ID = paste(COUNTRY, AREA, GEAR,FISHERY, VESSEL_LENGTH, MESH_SIZE_RANGE,
                            YEAR, QUARTER, SPECIES, sep = "_"))

      catch.tab$YEAR=as.numeric(catch.tab$YEAR)

      catch.temp2<-data.frame(matrix(nrow=0,ncol=length(header)))
      colnames(catch.temp2)=as.vector(header)

      # lan.temp2<-rbind(lan.temp2,land.tab)

      catch.temp2<-rbind(catch.temp2,catch.tab)

      #catch.temp2<-bind_rows(catch.temp2,catch.tab)
}
  }
```

```
  #catch.temp2[,-c(1:11)][is.na(catch.temp2[,-c(1:11)])] <- 0

  catch.temp2<-data.table::setDT(catch.temp2)


  for (jj in c(12:171)) set(catch.temp2, i = which(catch.temp2[[jj]]==-1),
                            j = jj, v = 0)

  catch.temp2<-setDF(catch.temp2)



  catch.temp2<-  suppressWarnings(catch.temp2 %>% select(dplyr::all_of(header2)))

  colnames(catch.temp2)=header


return(catch.temp2)
    }

#' Age Length Key (ALK) table - MED & BS data call
#'
#' @param data Detailed data in RCG CS format
#' @return ALK table
#' @export
#' @examples
#' library(COSTcore)
#' ALK_MEDBS(RDBprocessing::data_ex)
#' @importFrom data.table last first between as.data.table
#' @importFrom stats as.formula
#' @import COSTdbe COSTeda COSTcore
#' @importFrom dplyr group_by filter select summarize mutate_at one_of if_else n
#' @importFrom tidyr gather
#' @importFrom magrittr %>%
#' @importFrom plyr .

ALK_MEDBS<-function(data) {


    data=check_cs_header(data)
. <- year<-n<-Start<-End<-NULL  # vars<-.<-funs<-

    data$Age=round(data$Age,0)
data[is.na(data)]<-""
    data$Individual_weight=""
data$fish_ID=""
    data$Maturity_Stage=""

    data2=aggregate(data$Number_at_length,
by=list(data$Sampling_type,data$Flag_country,data$Year,data$Trip_code,data$Harbour,data$Number_of_sets_hauls_on_tr
ip,data$Days_at_sea,data$Sampling_method,data$Aggregation_level,data$Station_number,data$Duration_of_fishing_opera
tion,data$Initial_latitude,data$Initial_longitude,data$Final_latitude,data$Final_longitude,data$Depth_of_fishing_o
peration,data$Water_depth,data$Catch_registration,data$Species_registration,data$Date,data$Area,data$Fishing_activ
ity_category_National,data$Fishing_activity_category_European_lvl_6,data$Species,data$Catch_category,data$Weight,d
ata$Subsample_weight,data$Sex,data$Maturity_method,data$Maturity_scale,data$Maturity_Stage,data$Ageing.method,data
$Age,data$Length_code,data$Length_class,data$Commercial_size_category_scale,data$Commercial_size_category,data$fis
h_ID,data$Individual_weight), FUN="sum")
    data2=data2[,c(1:35,40,36:39)]
    colnames(data2)=colnames(RDBprocessing::data_ex)

if(any(is.na(data2[data2$Age=="",]$Age))){
data2[data2$Age=="",]$Age<-NA
}

    #data=data[!is.na(data$Age),]


LC<- age<- area<- logMsg<- n.at.len<- sex<- spp<- value<-NULL

  mat=  aggregate(data$Age,by=list(data$Flag_country,data$Year,data$Sex,data$Area,data$Species),FUN="length")
  colnames(mat)=c("COUNTRY","START_YEAR","SEX","AREA","SPECIES","NB")
  tab1=merge(mat,RDBprocessing::Annex17,by.x="SPECIES",by.y="Scientific_name")
  tab1=tab1[tab1$SEX=="F" | tab1$SEX=="M",]
  tab2=tab1[tab1$SEX=="F",]
  tab2$SEX="C"
  tab1=rbind(tab1,tab2)

  colnames(tab1)[7]="SPE"


  sel_spe<-tab1
  sel_spe$END_YEAR<-sel_spe$START_YEAR
  sel_spe$methodDesc_LAN.len.age<-"analytical"
  sel_spe$adjust<-"FALSE"
  colnames(sel_spe)[5]="GSA"

  sel_spe$typeALK<- "stepIncr"
  sel_spe$valueALK<-10
```

```
  sel_spe$APPLY_TO_CATCHES_FILE <- "Y"
  sel_spe$COMMENTS <-""
  sel_spe$LC_RANGE <-10
  header<-
c("COUNTRY","AREA","START_YEAR","END_YEAR","SPECON","SPECIES","SEX","APPLY_TO_CATCHES_FILE","TOTAL_NUMBER_OF_HARD_
STRUCTURE_READ_BY_AGE","CV","UNIT","AGE" ,paste("LENGTHCLASS",seq(0,99),sep=""),"LENGTHCLASS100_PLUS","COMMENTS")

  i=1
  for (i in c(1:dim(sel_spe)[1])) {

      STK<- sel_spe$SPECIES[i]

      #Start<<-as.numeric()
      #End<<-as.numeric(sel_spe$END_YEAR[i])

      # filter year
       #     fri_cs1<- subset(fri_cs, year %in% seq(Start,
                                          # End,by=1),table="ca",link=T)

      data2_temp=data2[as.numeric(data2$year) %in% c(sel_spe$START_YEAR[i]:sel_spe$END_YEAR[i]),]



      #  estimate sample size (number of otoliths per stock, sex and age)
      if (sel_spe$SEX[i]=="C"){

fri_cs1<-RCGtoCOST_CS(data2_temp)

          fri_strD <- strIni(spaceStrata="area")
          COUNTRY<-unique(fri_cs1@ca$landCtry)

          fri_csv <- COSTcore::csDataVal(fri_cs1)

nml<- suppressMessages ( data.frame(fri_cs1@ca) %>% dplyr::filter(!is.na(age))%>%
              dplyr::group_by(area,spp,age)%>%
              dplyr::summarize(TOTAL_NUMBER_OF_HARD_STRUCTURE_READ_BY_AGE=n()) )

      } else { # ALK for selected sex

         data2_temp= data2_temp[data2_temp$sex==sel_spe$SEX[i],]
          fri_cs1<-RCGtoCOST_CS(data2_temp)
          # header<-
c("COUNTRY","AREA","START_YEAR","END_YEAR","SPECON","SPECIES","SEX","APPLY_TO_CATCHES_FILE","TOTAL_NUMBER_OF_HARD_
STRUCTURE_READ_BY_AGE","CV","UNIT","AGE" ,paste("LENGTHCLASS",seq(0,99),sep=""),"LENGTHCLASS100_PLUS","COMMENTS")

fri_strD <- strIni(spaceStrata="area")
          COUNTRY<-unique(fri_cs1@ca$landCtry)


#fri_cs1=COSTcore::subset(fri_cs1,sex==sel_spe$SEX[i],table="ca",link=T)

fri_csv <- COSTcore::csDataVal(fri_cs1)


# get sample size: number of otoliths
          nml<- suppressMessages ( data.frame(fri_cs1@ca) %>% dplyr::filter(!is.na(age))%>%
              dplyr::group_by(area,spp,age,sex)%>%
            dplyr::summarize(TOTAL_NUMBER_OF_HARD_STRUCTURE_READ_BY_AGE=n()) )


      }


      fri_csv1<- fri_csv #subset(fri_csv, spp == STK,table="ca",link=T) # COSTeda::subSetSpp(fri_csv, STK)


      fri_csc1 <- suppressWarnings(csDataCons(fri_csv1, fri_strD))


       ## ### CV from individual length-at-age
      LEstim_An <-
          dbeObject(
              desc = "Length at age",
              species = STK,
              catchCat = "LAN",
              param = "length",
              strataDesc = fri_strD, # ,
              methodDesc = "analytical"   #sel_spe$methodDesc_LAN.len.age[i]
          )


      if(sel_spe$methodDesc_LAN.len.age[i]=="analytical") {
          LEstim_An <-  bpEstim(LEstim_An, fri_csc1, adjust = sel_spe$adjust[i])

      } else{
          LEstim_An <-  bpBoot(LEstim_An, fri_csc1, adjust = sel_spe$adjust[i])

      }
```

```
      ## ALK

      res1 <- COSTdbe::alkLgthRec(fri_csc1,type=sel_spe$typeALK[i],value=sel_spe$valueALK[i],update=F,
preview=F,postview = F)



      if (sel_spe$typeALK[i]=="fillALKmult"){
          fri_csc2 <- fillALKmult(fri_csc1,STK,p=10,trace=T)
          res1 <- alkLgthRec(fri_csc2,update=F, preview=F,postview = F,
                             value=sel_spe$valueALK[i])
      }

      dfALK <-
          data.frame(
              COUNTRY =COUNTRY ,
              AREA =sel_spe$GSA[i],
              START_YEAR = unique(sel_spe$START_YEAR[i]) ,
              END_YEAR = unique( sel_spe$END_YEAR[i]) ,
              SPECIES = STK ,
              SEX = sel_spe$SEX[i],
              UNIT = unique(fri_cs1@ca$lenCode[fri_cs1@ca$spp==STK]) ,
              SPECON=  -1,
              APPLY_TO_CATCHES_FILE= unique(sel_spe$APPLY_TO_CATCHES_FILE[i]),
              CV=NA,
              AGE = as.numeric(colnames(res1$alk)),
              COMMENTS= sel_spe$COMMENTS[i],
              LENGTHCLASS100_PLUS=0
          )

      # get sample size

      if (sel_spe$SEX[i]=="C"){

          dfALK <- suppressMessages( dfALK %>%
              left_join(nml, by = c("AREA" = 'area', 'SPECIES' = 'spp',"AGE"="age")) %>%
              dplyr::mutate(SPECIES =  sel_spe$SPE[i]))
          # FAO Three alpha code
      } else{

          dfALK <- suppressMessages( dfALK %>%
              left_join(nml, by = c("AREA" = 'area', 'SPECIES' = 'spp',"AGE"="age",
                                    "SEX"="sex")) %>%
              dplyr::mutate(SPECIES =  sel_spe$SPE[i]) )
          # FAO Three alpha code

      }


      aa=data.frame(res1$alk)
      names(aa)=colnames(res1$alk)

      aa=aa%>% dplyr::mutate(LC=rownames(res1$alk))

      ## fix LC
      UNIT <- as.character( unique(fri_csc1@ca$lenCode[fri_csc1@ca$spp==STK]) )

      if (UNIT %in% c("mm", "MM")& sel_spe$LC_RANGE[i]==10) {
          aa$LC<-as.numeric(aa$LC)/10
          UNIT1<-"cm"
      }

      if (UNIT %in% c("mm", "MM") & sel_spe$LC_RANGE[i]==1) {
          aa$LC<-as.numeric(aa$LC)
          UNIT1<- "mm"
      }

      if (UNIT %in% c("mm", "MM")& sel_spe$LC_RANGE[i]==5) {
          aa$LC<-as.numeric(aa$LC)/10
          UNIT1<-"cm"
      }

      if (UNIT %in% c("cm", "CM") ) {
          aa$LC<-as.numeric(aa$LC)
          UNIT1<- "cm"
      }

      if (UNIT %in% c("scm", "SCM")& sel_spe$LC_RANGE[i]==10) {
          aa$LC<-as.numeric(aa$LC)/10
          UNIT1<-"cm"
      }

      if (UNIT %in% c("scm", "SCM")& sel_spe$LC_RANGE[i]==5) {
          aa$LC<-as.numeric(aa$LC)/10
          UNIT1<-"cm"
      }
        round_any = function(x, accuracy, f=round){f(x/ accuracy) * accuracy}

      aa$LC<- round_any( aa$LC,1,floor)
```

```
      ###

      dfALK$UNIT<-UNIT1

      aa1=aa %>%  tidyr::gather(age, n.at.len, -LC)

aa1 <- as.data.table(aa1)
#aa$LC
seq_l <-  seq(0, 99, by = 1) #
      dt1<- aa1[, list(LC = seq_l), by = age]


      dt2<- suppressMessages (left_join(dt1,aa1))

      dt3 <- data.table::dcast(dt2,as.formula(paste(paste(names(dt2)[! names(dt2)
                                                                    %in% c("LC","n.at.len")], collapse='+'),
"LC", sep="~")),
value.var = "n.at.len")

dt3$age=as.numeric(dt3$age)


      dt3<- suppressWarnings(dt3 %>% mutate_at(vars( -(age) ),
                           funs( if_else( is.na(.), 0, .) ) ))

      dfALK<- suppressMessages (left_join(dfALK,dt3,by=c("AGE"="age")))

      ## CV
      # LEstim_An @ageNum$ cv

      LEstim_An@ageNum[["cv"]]$age=as.numeric(LEstim_An@ageNum[["cv"]]$age)

      dfALK<- suppressMessages( dfALK%>% left_join(LEstim_An@ageNum[["cv"]]%>% select(age,value),
                                by=c( "AGE"="age"))%>% dplyr::mutate(CV=value)%>% select(-c(value)))

      # take care of number of Length classes (max is 100 acc. to DG MARE Med&BS template)
      zz<-dim(dfALK[-c(1:14)])[2]
      names(dfALK)[-c(1:14)]<- paste("LENGTHCLASS",seq(0,zz-1,1),sep="")


      if(zz>=100){
          dfALK$LENGTHCLASS100_PLUS<- rowSums(dfALK[,-c(1:114)])
      }


      alk.temp2<-data.frame(matrix(nrow=0,ncol=114))
colnames(alk.temp2)=as.vector(header)

      dfALK<-dfALK%>% dplyr::select(one_of(as.vector(names(alk.temp2))))


     # dfALK


      alk.temp2<- rbind(alk.temp2,(dfALK))
      #alk.temp2<-dfALK
      #alk.temp2[,-c(1:13,114)][is.na(alk.temp2[,-c(1:13,114)])] <- 0


      # export updated CsDataCons

      if (sel_spe$APPLY_TO_CATCHES_FILE[i] == "Y") {
          if (sel_spe$typeALK[i] == "fillALKmult") {
              fri_csc2 <- fillALKmult(fri_csc1, STK, p = 10, trace = T)

              save(fri_csc2,
                   file = paste("upd", STK, sel_spe$SEX[i], sel_spe$GSA[i], ".Rdata", sep ="_"))

          } else{
              res1 <-
                  alkLgthRec(
                      fri_csc1,
                      type = sel_spe$typeALK[i],
                      value = sel_spe$valueALK[i],
                      update = T,
                      preview = F,
                      postview = F
                  )

              #save(res1,
                  # file = paste("upd", STK, sel_spe$SEX[i], sel_spe$GSA[i], ".Rdata", sep =
                                 #  "_"))
          }

      }

      if (i ==1) {
        alk.temp3 <-  alk.temp2
      } else {
```

```
        alk.temp3 <- rbind( alk.temp3,  alk.temp2)
      }

  }

  alk.temp3=alk.temp3[alk.temp3$AGE!=-1,]

  alk.temp3[is.na(alk.temp3$CV),]$CV=-1

  return(alk.temp3)

}

#' Age Length Key (ALK) table - MED & BS data call
#'
#' @param data Detailed data in RCG CS format
#' @return ALK table
#' @export
#' @examples
#' library(COSTcore)
#' ALK_MEDBS(RDBprocessing::data_ex)
#' @importFrom data.table last first between as.data.table
#' @importFrom stats as.formula
#' @import COSTdbe COSTeda COSTcore
#' @importFrom dplyr group_by filter select summarize mutate_at one_of if_else n
#' @importFrom tidyr gather
#' @importFrom magrittr %>%
#' @importFrom plyr .

ALK_MEDBS<-function(data) {


    data=check_cs_header(data)
. <- year<-n<-Start<-End<-NULL  # vars<-.<-funs<-

    data$Age=round(data$Age,0)
data[is.na(data)]<-""
    data$Individual_weight=""
data$fish_ID=""
    data$Maturity_Stage=""

    data2=aggregate(data$Number_at_length,
by=list(data$Sampling_type,data$Flag_country,data$Year,data$Trip_code,data$Harbour,data$Number_of_sets_hauls_on_tr
ip,data$Days_at_sea,data$Sampling_method,data$Aggregation_level,data$Station_number,data$Duration_of_fishing_opera
tion,data$Initial_latitude,data$Initial_longitude,data$Final_latitude,data$Final_longitude,data$Depth_of_fishing_o
peration,data$Water_depth,data$Catch_registration,data$Species_registration,data$Date,data$Area,data$Fishing_activ
ity_category_National,data$Fishing_activity_category_European_lvl_6,data$Species,data$Catch_category,data$Weight,d
ata$Subsample_weight,data$Sex,data$Maturity_method,data$Maturity_scale,data$Maturity_Stage,data$Ageing.method,data
$Age,data$Length_code,data$Length_class,data$Commercial_size_category_scale,data$Commercial_size_category,data$fis
h_ID,data$Individual_weight), FUN="sum")
    data2=data2[,c(1:35,40,36:39)]
    colnames(data2)=colnames(RDBprocessing::data_ex)

if(any(is.na(data2[data2$Age=="",]$Age))){
data2[data2$Age=="",]$Age<-NA
}

    #data=data[!is.na(data$Age),]


LC<- age<- area<- logMsg<- n.at.len<- sex<- spp<- value<-NULL

  mat=  aggregate(data$Age,by=list(data$Flag_country,data$Year,data$Sex,data$Area,data$Species),FUN="length")
  colnames(mat)=c("COUNTRY","START_YEAR","SEX","AREA","SPECIES","NB")
  tab1=merge(mat,RDBprocessing::Annex17,by.x="SPECIES",by.y="Scientific_name")
  tab1=tab1[tab1$SEX=="F" | tab1$SEX=="M",]
  tab2=tab1[tab1$SEX=="F",]
  tab2$SEX="C"
  tab1=rbind(tab1,tab2)

  colnames(tab1)[7]="SPE"


  sel_spe<-tab1
  sel_spe$END_YEAR<-sel_spe$START_YEAR
  sel_spe$methodDesc_LAN.len.age<-"analytical"
  sel_spe$adjust<-"FALSE"
  colnames(sel_spe)[5]="GSA"

  sel_spe$typeALK<- "stepIncr"
  sel_spe$valueALK<-10
  sel_spe$APPLY_TO_CATCHES_FILE <- "Y"
  sel_spe$COMMENTS <-""
  sel_spe$LC_RANGE <-10
  header<-
c("COUNTRY","AREA","START_YEAR","END_YEAR","SPECON","SPECIES","SEX","APPLY_TO_CATCHES_FILE","TOTAL_NUMBER_OF_HARD_
STRUCTURE_READ_BY_AGE","CV","UNIT","AGE" ,paste("LENGTHCLASS",seq(0,99),sep=""),"LENGTHCLASS100_PLUS","COMMENTS")

  i=1
  for (i in c(1:dim(sel_spe)[1])) {
```

```
      STK<- sel_spe$SPECIES[i]

      #Start<<-as.numeric()
      #End<<-as.numeric(sel_spe$END_YEAR[i])

      # filter year
       #     fri_cs1<- subset(fri_cs, year %in% seq(Start,
                                          # End,by=1),table="ca",link=T)

      data2_temp=data2[as.numeric(data2$year) %in% c(sel_spe$START_YEAR[i]:sel_spe$END_YEAR[i]),]



      #  estimate sample size (number of otoliths per stock, sex and age)
      if (sel_spe$SEX[i]=="C"){

fri_cs1<-RCGtoCOST_CS(data2_temp)

          fri_strD <- strIni(spaceStrata="area")
          COUNTRY<-unique(fri_cs1@ca$landCtry)

          fri_csv <- COSTcore::csDataVal(fri_cs1)

nml<- suppressMessages ( data.frame(fri_cs1@ca) %>% dplyr::filter(!is.na(age))%>%
              dplyr::group_by(area,spp,age)%>%
              dplyr::summarize(TOTAL_NUMBER_OF_HARD_STRUCTURE_READ_BY_AGE=n()) )

      } else { # ALK for selected sex

         data2_temp= data2_temp[data2_temp$sex==sel_spe$SEX[i],]
          fri_cs1<-RCGtoCOST_CS(data2_temp)
          # header<-
c("COUNTRY","AREA","START_YEAR","END_YEAR","SPECON","SPECIES","SEX","APPLY_TO_CATCHES_FILE","TOTAL_NUMBER_OF_HARD_
STRUCTURE_READ_BY_AGE","CV","UNIT","AGE" ,paste("LENGTHCLASS",seq(0,99),sep=""),"LENGTHCLASS100_PLUS","COMMENTS")

fri_strD <- strIni(spaceStrata="area")
          COUNTRY<-unique(fri_cs1@ca$landCtry)


#fri_cs1=COSTcore::subset(fri_cs1,sex==sel_spe$SEX[i],table="ca",link=T)

fri_csv <- COSTcore::csDataVal(fri_cs1)


# get sample size: number of otoliths
          nml<- suppressMessages ( data.frame(fri_cs1@ca) %>% dplyr::filter(!is.na(age))%>%
              dplyr::group_by(area,spp,age,sex)%>%
            dplyr::summarize(TOTAL_NUMBER_OF_HARD_STRUCTURE_READ_BY_AGE=n()) )


      }


      fri_csv1<- fri_csv #subset(fri_csv, spp == STK,table="ca",link=T) # COSTeda::subSetSpp(fri_csv, STK)


      fri_csc1 <- suppressWarnings(csDataCons(fri_csv1, fri_strD))


       ## ### CV from individual length-at-age
      LEstim_An <-
          dbeObject(
              desc = "Length at age",
              species = STK,
              catchCat = "LAN",
              param = "length",
              strataDesc = fri_strD, # ,
              methodDesc = "analytical"   #sel_spe$methodDesc_LAN.len.age[i]
          )


      if(sel_spe$methodDesc_LAN.len.age[i]=="analytical") {
          LEstim_An <-  bpEstim(LEstim_An, fri_csc1, adjust = sel_spe$adjust[i])

      } else{
          LEstim_An <-  bpBoot(LEstim_An, fri_csc1, adjust = sel_spe$adjust[i])

      }


      ## ALK

      res1 <- COSTdbe::alkLgthRec(fri_csc1,type=sel_spe$typeALK[i],value=sel_spe$valueALK[i],update=F,
preview=F,postview = F)



      if (sel_spe$typeALK[i]=="fillALKmult"){
          fri_csc2 <- fillALKmult(fri_csc1,STK,p=10,trace=T)
```

```
        res1 <- alkLgthRec(fri_csc2,update=F, preview=F,postview = F,
                           value=sel_spe$valueALK[i])
    }

    dfALK <-
        data.frame(
            COUNTRY =COUNTRY ,
            AREA =sel_spe$GSA[i],
            START_YEAR = unique(sel_spe$START_YEAR[i]) ,
            END_YEAR = unique( sel_spe$END_YEAR[i]) ,
            SPECIES = STK ,
            SEX = sel_spe$SEX[i],
            UNIT = unique(fri_cs1@ca$lenCode[fri_cs1@ca$spp==STK]) ,
            SPECON=  -1,
            APPLY_TO_CATCHES_FILE= unique(sel_spe$APPLY_TO_CATCHES_FILE[i]),
            CV=NA,
            AGE = as.numeric(colnames(res1$alk)),
            COMMENTS= sel_spe$COMMENTS[i],
            LENGTHCLASS100_PLUS=0
        )

    # get sample size

    if (sel_spe$SEX[i]=="C"){

        dfALK <- suppressMessages( dfALK %>%
            left_join(nml, by = c("AREA" = 'area', 'SPECIES' = 'spp',"AGE"="age")) %>%
            dplyr::mutate(SPECIES =  sel_spe$SPE[i]))
        # FAO Three alpha code
    } else{

        dfALK <- suppressMessages( dfALK %>%
            left_join(nml, by = c("AREA" = 'area', 'SPECIES' = 'spp',"AGE"="age",
                                  "SEX"="sex")) %>%
            dplyr::mutate(SPECIES =  sel_spe$SPE[i]) )
        # FAO Three alpha code

    }


    aa=data.frame(res1$alk)
    names(aa)=colnames(res1$alk)

    aa=aa%>% dplyr::mutate(LC=rownames(res1$alk))

    ## fix LC
    UNIT <- as.character( unique(fri_csc1@ca$lenCode[fri_csc1@ca$spp==STK]) )

    if (UNIT %in% c("mm", "MM")& sel_spe$LC_RANGE[i]==10) {
        aa$LC<-as.numeric(aa$LC)/10
        UNIT1<-"cm"
    }

    if (UNIT %in% c("mm", "MM") & sel_spe$LC_RANGE[i]==1) {
        aa$LC<-as.numeric(aa$LC)
        UNIT1<- "mm"
    }

    if (UNIT %in% c("mm", "MM")& sel_spe$LC_RANGE[i]==5) {
        aa$LC<-as.numeric(aa$LC)/10
        UNIT1<-"cm"
    }

    if (UNIT %in% c("cm", "CM") ) {
        aa$LC<-as.numeric(aa$LC)
        UNIT1<- "cm"
    }

    if (UNIT %in% c("scm", "SCM")& sel_spe$LC_RANGE[i]==10) {
        aa$LC<-as.numeric(aa$LC)/10
        UNIT1<-"cm"
    }

    if (UNIT %in% c("scm", "SCM")& sel_spe$LC_RANGE[i]==5) {
        aa$LC<-as.numeric(aa$LC)/10
        UNIT1<-"cm"
    }
      round_any = function(x, accuracy, f=round){f(x/ accuracy) * accuracy}

    aa$LC<- round_any( aa$LC,1,floor)


    ###

    dfALK$UNIT<-UNIT1

    aa1=aa %>%  tidyr::gather(age, n.at.len, -LC)

aa1 <- as.data.table(aa1)
#aa$LC
```

```
seq_l <-  seq(0, 99, by = 1) #
      dt1<- aa1[, list(LC = seq_l), by = age]


      dt2<- suppressMessages (left_join(dt1,aa1))

      dt3 <- data.table::dcast(dt2,as.formula(paste(paste(names(dt2)[! names(dt2)
                                                                    %in% c("LC","n.at.len")], collapse='+'),
"LC", sep="~")),
value.var = "n.at.len")

dt3$age=as.numeric(dt3$age)


      dt3<- suppressWarnings(dt3 %>% mutate_at(vars( -(age) ),
                           funs( if_else( is.na(.), 0, .) ) ))

      dfALK<- suppressMessages (left_join(dfALK,dt3,by=c("AGE"="age")))

      ## CV
      # LEstim_An @ageNum$ cv

      LEstim_An@ageNum[["cv"]]$age=as.numeric(LEstim_An@ageNum[["cv"]]$age)

      dfALK<- suppressMessages( dfALK%>% left_join(LEstim_An@ageNum[["cv"]]%>% select(age,value),
                                by=c( "AGE"="age"))%>% dplyr::mutate(CV=value)%>% select(-c(value)))

      # take care of number of Length classes (max is 100 acc. to DG MARE Med&BS template)
      zz<-dim(dfALK[-c(1:14)])[2]
      names(dfALK)[-c(1:14)]<- paste("LENGTHCLASS",seq(0,zz-1,1),sep="")


      if(zz>=100){
          dfALK$LENGTHCLASS100_PLUS<- rowSums(dfALK[,-c(1:114)])
      }


      alk.temp2<-data.frame(matrix(nrow=0,ncol=114))
colnames(alk.temp2)=as.vector(header)

      dfALK<-dfALK%>% dplyr::select(one_of(as.vector(names(alk.temp2))))


     # dfALK


      alk.temp2<- rbind(alk.temp2,(dfALK))
      #alk.temp2<-dfALK
      #alk.temp2[,-c(1:13,114)][is.na(alk.temp2[,-c(1:13,114)])] <- 0


      # export updated CsDataCons

      if (sel_spe$APPLY_TO_CATCHES_FILE[i] == "Y") {
          if (sel_spe$typeALK[i] == "fillALKmult") {
              fri_csc2 <- fillALKmult(fri_csc1, STK, p = 10, trace = T)

              save(fri_csc2,
                   file = paste("upd", STK, sel_spe$SEX[i], sel_spe$GSA[i], ".Rdata", sep ="_"))

          } else{
              res1 <-
                  alkLgthRec(
                      fri_csc1,
                      type = sel_spe$typeALK[i],
                      value = sel_spe$valueALK[i],
                      update = T,
                      preview = F,
                      postview = F
                  )

              #save(res1,
                  # file = paste("upd", STK, sel_spe$SEX[i], sel_spe$GSA[i], ".Rdata", sep =
                                 #  "_"))
          }

      }

      if (i ==1) {
        alk.temp3 <-  alk.temp2
      } else {
        alk.temp3 <- rbind( alk.temp3,  alk.temp2)
      }

  }

  alk.temp3=alk.temp3[alk.temp3$AGE!=-1,]

  alk.temp3[is.na(alk.temp3$CV),]$CV=-1
```

```
  return(alk.temp3)

}

#' Maturity at length (ML) table - MED & BS data call
#' @param data Detailed data in RCG CS format
#' @param imm maturity stages to be considered as immature
#' @param verbose boolean. If TRUE a message is printed.
#' @return ML table
#' @export
#' @examples ML_MEDBS(RDBprocessing::data_ex)
#' @importFrom stats aggregate

ML_MEDBS<-function(data,imm=c("1","2","2a"), verbose = FALSE) {

    data=check_cs_header(data)


header=c("COUNTRY","AREA","START_YEAR","END_YEAR","SPECIES","SEX","LENGTHCLASS","UNIT","SAMPLE_SIZE","PRM","METHOD
_USED")

data=data[-which(data$Maturity_Stage %in%c(NA,-1,0,"")),]

data$mat=ifelse(data$Maturity_Stage %in% imm,0,1)

data2=merge(data,RDBprocessing::Annex17,by.x="Species",by.y="Scientific_name")

data=data2 #[-which(as.character(data2$Species.y)==""),]

matrix=aggregate(data$Number_at_length,by=list(data$Flag_country,
data$Year,data$Area,data$Species.y,data$Sex,data$Length_class,data$mat,data$Length_code),FUN="sum")
colnames(matrix)=c("COUNTRY", "START_YEAR","AREA","SPECIES","SEX","LENGTHCLASS","mat","UNIT","SAMPLE_SIZE")
immat=matrix[matrix$mat==0,]
mat=matrix[matrix$mat==1,]

tab1=merge(immat,mat,by=c("COUNTRY", "START_YEAR","AREA","SPECIES","SEX","LENGTHCLASS","UNIT") )
tab1=tab1[,c(1:7,9,11)]
colnames(tab1)[c(8,9)]=c("imm","mat")

tab1=merge(tab1,RDBprocessing::Annex17,by.x="SPECIES",by.y="Species")


tab1$LENGTHCLASS=ifelse(tab1[,11]=="cm",tab1$LENGTHCLASS/10,tab1$LENGTHCLASS)
tab1$LENGTHCLASS=trunc(tab1$LENGTHCLASS)
colnames(tab1)[11]="UNIT"

tab1=aggregate(c(tab1[,c(8,9)]),by=list(tab1$SPECIES,tab1$COUNTRY, tab1$START_YEAR, tab1$AREA, tab1$SEX,
tab1$LENGTHCLASS,tab1$UNIT),FUN="sum")
colnames(tab1)=c("SPECIES","COUNTRY","START_YEAR","AREA","SEX","LENGTHCLASS","UNIT","imm","mat")
tab1$PRM=tab1$mat/(tab1$imm+tab1$mat)
tab1$SAMPLE_SIZE=(tab1$imm+tab1$mat)

ML=tab1[,c(2,4,3,3,1,5,6,7,11,10)]

ML$METHOD_USED="Observed proportions (immatures 1,2,2a)"
colnames(ML)[4]="END_YEAR"

return(ML)

}

#' Maturity at age (MA) table - MED & BS data call
#' @param data Detailed data in RCG CS format
#' @param imm maturity stages to be considered as immature
#' @param verbose boolean. If TRUE a message is printed.
#' @return MA table
#' @export
#' @examples MA_MEDBS(RDBprocessing::data_ex)
#' @importFrom stats aggregate

MA_MEDBS<-function(data,imm=c("1","2","2a"), verbose = FALSE) {

    data=check_cs_header(data)

header=c("COUNTRY","AREA","START_YEAR","END_YEAR","SPECIES","SEX","AGECLASS","SAMPLE_SIZE","PRM","METHOD_USED")

    data=data[-which(data$Maturity_Stage %in% c(NA,-1,0,"") | data$Age %in% c(NA,-1,"")),]

    data$mat=ifelse(data$Maturity_Stage %in% imm,0,1)

    data2=merge(data,RDBprocessing::Annex17,by.x="Species",by.y="Scientific_name")

    data=data2 #[-which(as.character(data2$Species.y)==""),]

    matrix=aggregate(data$Number_at_length,by=list(data$Flag_country,
data$Year,data$Area,data$Species.y,data$Sex,data$Age,data$mat),FUN="sum")
    colnames(matrix)=c("COUNTRY", "START_YEAR","AREA","SPECIES","SEX","AGECLASS","mat","SAMPLE_SIZE")
    immat=matrix[matrix$mat==0,]
    mat=matrix[matrix$mat==1,]
```

```
    tab1=merge(immat,mat,by=c("COUNTRY", "START_YEAR","AREA","SPECIES","SEX","AGECLASS") )
    tab1=tab1[,c(1:6,8,10)]
    colnames(tab1)[c(7,8)]=c("imm","mat")

    tab1=merge(tab1,RDBprocessing::Annex17,by.x="SPECIES",by.y="Species")


   # tab1$LENGTHCLASS=ifelse(tab1[,11]=="cm",tab1$LENGTHCLASS/10,tab1$LENGTHCLASS)
    tab1$AGECLASS=trunc(tab1$AGECLASS)
    #colnames(tab1)[11]="UNIT"

    tab1=aggregate(c(tab1[,c(7,8)]),by=list(tab1$SPECIES,tab1$COUNTRY, tab1$START_YEAR, tab1$AREA, tab1$SEX,
tab1$AGECLASS),FUN="sum")

    colnames(tab1)=c("SPECIES","COUNTRY","START_YEAR","AREA","SEX","AGECLASS","imm","mat")
    tab1$PRM=tab1$mat/(tab1$imm+tab1$mat)
    tab1$SAMPLE_SIZE=(tab1$imm+tab1$mat)

    MA=tab1[,c(2,4,3,3,1,5,6,10,9)]

    MA$METHOD_USED="Observed proportions (immatures 1,2,2a)"
    colnames(MA)[4]="END_YEAR"

    return(MA)

}

#' Sex ratio at length (SRL) table - MED & BS data call
#' @param data Detailed data in RCG CS format
#' @param verbose boolean. If TRUE a message is printed.
#' @return SRL table
#' @export
#' @examples SRL_MEDBS(RDBprocessing::data_ex)
#' @importFrom stats aggregate
SRL_MEDBS<-function(data, verbose = FALSE) {
    data=check_cs_header(data)
    header=c("COUNTRY","AREA","START_YEAR","END_YEAR","SPECIES","LENGTHCLASS","UNIT","SEX_RATIO","COMMENTS")

    data=data[-which(data$Sex %in%c(NA,-1,0,"","U","C","N","I")),]

    data$fem=ifelse(data$Sex== "F",1,0)

    data2=merge(data,RDBprocessing::Annex17,by.x="Species",by.y="Scientific_name") #

    data=data2 #[-which(as.character(data2$Species.y)==""),]

    matrix=aggregate(data$Number_at_length,by=list(data$Flag_country,
data$Year,data$Area,data$Species.y,data$Length_class,data$fem,data$Length_code),FUN="sum")

    colnames(matrix)=c("COUNTRY", "START_YEAR","AREA","SPECIES","LENGTHCLASS","fem","UNIT","SAMPLE_SIZE")
    fem=matrix[matrix$fem==1,]
    mal=matrix[matrix$fem==0,]

    tab1=merge(fem,mal,by=c("COUNTRY", "START_YEAR","AREA","SPECIES","LENGTHCLASS","UNIT") )
    tab1=tab1[,c(1:6,8,10)]
    colnames(tab1)[c(7,8)]=c("fem","mal")

    tab1=merge(tab1,RDBprocessing::Annex17,by.x="SPECIES",by.y="Species") #


    tab1$LENGTHCLASS=ifelse(tab1[,10]=="cm",tab1$LENGTHCLASS/10,tab1$LENGTHCLASS)
    tab1$LENGTHCLASS=trunc(tab1$LENGTHCLASS)
    colnames(tab1)[10]="UNIT"

    tab1=aggregate(c(tab1[,c(7,8)]),by=list(tab1$SPECIES,tab1$COUNTRY, tab1$START_YEAR, tab1$AREA,
tab1$LENGTHCLASS,tab1$UNIT),FUN="sum")
    colnames(tab1)=c("SPECIES","COUNTRY","START_YEAR","AREA","LENGTHCLASS","UNIT","fem","mal")
    tab1$SEX_RATIO=tab1$fem/(tab1$fem+tab1$mal)
    #tab1$SAMPLE_SIZE=(tab1$imm+tab1$mat)

    SRL=tab1[,c(2,4,3,3,1,5,6,9)]

    SRL$COMMENTS="Observed F/(F+M)"
    colnames(SRL)[4]="END_YEAR"

    return(SRL)

}


#' Sex ratio at age (SRA) table - MED & BS data call
#' @param data Detailed data in RCG CS format
#' @param verbose boolean. If TRUE a message is printed.
#' @return SRA table
#' @export
#' @examples SRA_MEDBS(RDBprocessing::data_ex)
#' @importFrom stats aggregate

SRA_MEDBS<-function(data, verbose = FALSE) {
    data=check_cs_header(data)
```

```
    header=c("COUNTRY","AREA","START_YEAR","END_YEAR","SPECIES","AGECLASS","SEX_RATIO","COMMENTS")

    data=data[-which(data$Sex %in%c(NA,-1,0,"","U","C","N","I") | data$Age %in% c(NA,-1,"")),]

    data$fem=ifelse(data$Sex== "F",1,0)

    data2=merge(data,RDBprocessing::Annex17,by.x="Species",by.y="Scientific_name") #

    data=data2 #[-which(as.character(data2$Species.y)==""),]

    matrix=aggregate(data$Number_at_length,by=list(data$Flag_country,
data$Year,data$Area,data$Species.y,data$Age,data$fem),FUN="sum")

    colnames(matrix)=c("COUNTRY", "START_YEAR","AREA","SPECIES","AGECLASS","fem","SAMPLE_SIZE")
    fem=matrix[matrix$fem==1,]
    mal=matrix[matrix$fem==0,]

    tab1=merge(fem,mal,by=c("COUNTRY", "START_YEAR","AREA","SPECIES","AGECLASS") )
    tab1=tab1[,c(1:5,7,9)]
    colnames(tab1)[c(6,7)]=c("fem","mal")

    tab1=merge(tab1,RDBprocessing::Annex17,by.x="SPECIES",by.y="Species") #


    #tab1$LENGTHCLASS=ifelse(tab1[,10]=="cm",tab1$LENGTHCLASS/10,tab1$LENGTHCLASS)
    tab1$AGECLASS=trunc(tab1$AGECLASS)
    #colnames(tab1)[10]="UNIT"

    tab1=aggregate(c(tab1[,c(6,7)]),by=list(tab1$SPECIES,tab1$COUNTRY, tab1$START_YEAR, tab1$AREA,
tab1$AGECLASS),FUN="sum")
    colnames(tab1)=c("SPECIES","COUNTRY","START_YEAR","AREA","AGECLASS","fem","mal")
    tab1$SEX_RATIO=tab1$fem/(tab1$fem+tab1$mal)
    #tab1$SAMPLE_SIZE=(tab1$imm+tab1$mat)

    SRA=tab1[,c(2,4,3,3,1,5,8)]

    SRA$COMMENTS="Observed F/(F+M)"
    colnames(SRA)[4]="END_YEAR"

    return(SRA)

}

#' CATCH table  - FDI data call
#'
#' @param datacl Landing data in RCG CL format
#' @param verbose boolean. If TRUE a message is printed.
#'
#' @return CATCH FDI table
#' @export
#'
#' @examples CATCH_FDI(RDBprocessing::data_exampleCL)
#'@import tidyr

CATCH_FDI<-function(datacl,verbose=FALSE){

header=c("COUNTRY",
"YEAR",
"QUARTER",
"VESSEL_LENGTH",
"FISHING_TECH",
"GEAR_TYPE",
"TARGET_ASSEMBLAGE",
"MESH_SIZE_RANGE",
"METIER",
"METIER_7",
"DOMAIN_DISCARDS",
"DOMAIN_LANDINGS",
"SUPRA_REGION",
"SUB_REGION",
"EEZ_INDICATOR",
"GEO_INDICATOR",
"NEP_SUB_REGION",
"SPECON_TECH",
"DEEP",
"SPECIES",
"TOTWGHTLANDG",
"TOTVALLANDG",
"DISCARDS",
"CONFIDENTIAL")

#datacl=RDBprocessing::data_exampleCL

DF=data.frame(COUNTRY=datacl$flag_country,
              YEAR=datacl$year,
              QUARTER=datacl$quarter,
              VESSEL_LENGTH=ifelse(any(!is.na(datacl$vslLenCat)),datacl$vslLenCat,"NK"),
              FISHING_TECH="",
              GEAR_TYPE="",
              TARGET_ASSEMBLAGE="",
```

```
              MESH_SIZE_RANGE="NK",
              METIER=datacl$fishing_activity_category_eu_l6,
              METIER_7="NA",
              DOMAIN_DISCARDS="NK",
              DOMAIN_LANDINGS="NK",
              SUPRA_REGION="MBS",
              SUB_REGION=datacl$area,
              EEZ_INDICATOR="NA",
              GEO_INDICATOR="NK",
              NEP_SUB_REGION="NA",
              SPECON_TECH="NK",
              DEEP="NA",
              SPECIES=datacl$species,
              TOTWGHTLANDG=round(datacl$official_landings_weight,3)/1000, # weihgts to be reported in tons
              TOTVALLANDG=round(datacl$official_landings_value,3),
              DISCARDS="NK",
              CONFIDENTIAL="N")

DF$VESSEL_LENGTH[is.na(DF$VESSEL_LENGTH)]="NK"
# update FISHING_TECH and GEAR

# extraction of the gear from the metier
DF= DF %>% tidyr::separate(col="METIER", into=c("GEAR_TYPE","TARGET_ASSEMBLAGE",NA,NA,NA),sep="_",remove=FALSE)

DF$FISHING_TECH <- RDBprocessing::FT_GEAR$FISHING_TECH[match(DF$GEAR_TYPE ,
                                                             RDBprocessing::FT_GEAR$GEAR)]

DF$SPECIES<-RDBprocessing::Annex17$Species[match(DF$SPECIES ,
                                                 RDBprocessing::Annex17$Scientific_name)]



DF$TARGET_ASSEMBLAGE = RDBprocessing::fishery$SDEF_codification[match(DF$TARGET_ASSEMBLAGE ,
                                           RDBprocessing::fishery$DGMARE_Med_BS_codification)]
return(DF)

    }


#' Table II.2  - GFCM DCRF data call
#'
#' @param CatchFDI Catch table FDI
#' @param verbose boolean. If TRUE a message is printed.
#' @return Table II.2 GFCM DCRF table
#' @export
#'
#' @examples TabII2_GFCM(RDBprocessing::CATCH_FDIex)
#'@import reshape
#'@import reshape2
#'@import dplyr


TabII2_GFCM <- function(CatchFDI,verbose=FALSE) {


    species=unique(ACatch$SPECIES)

#ACatch=ACatch[as.character(ACatch$SPECIES) %in% as.character(species[,1]),]

    #ACatch=ACatch[,c(2:9,11:13)]

ACatch_L=aggregate(ACatch$TOTWGHTLANDG,by=list(ACatch$COUNTRY, ACatch$YEAR,
                                        ACatch$VESSEL_LENGTH, ACatch$GEAR_TYPE,
                                        ACatch$TARGET_ASSEMBLAGE, ACatch$SUB_REGION, ACatch$SPECIES),FUN="sum")

class(ACatch$DISCARDS)="numeric"
#ACatch$DISCARDS[ACatch$DISCARDS=="NK"]=0

ACatch_D=aggregate(ACatch$DISCARDS,by=list(ACatch$COUNTRY, ACatch$YEAR,
                                        ACatch$VESSEL_LENGTH, ACatch$GEAR_TYPE,
                                        ACatch$TARGET_ASSEMBLAGE, ACatch$SUB_REGION,
ACatch$SPECIES),FUN="sum",na.rm=F)


Merge=merge(ACatch_L,ACatch_D,by=c("Group.1","Group.2","Group.3","Group.4","Group.5",
                                   "Group.6","Group.7"))

colnames(Merge) =c("COUNTRY", "YEAR", "VESSEL_LENGTH","GEAR", "FISHERY",
                   "AREA", "SPECIES","LANDINGS","DISCARDS")
Merge$DISCARDS[Merge$DISCARDS<0]=NA
Merge$GFCM_fleetsegment=as.character(Merge$GEAR)
Merge$GFCM_fleetsegment=""

CT=RDBprocessing::CT

for (i in 1:nrow(Merge)){
    if (nrow(CT[as.character(CT$LOA)== as.character(Merge$VESSEL_LENGTH[i]) &
                as.character(CT$GEAR_ACatch)== as.character(Merge$GEAR[i]) &
                as.character(CT$FISHERY_ACatch)== as.character(Merge$FISHERY[i]),])>0) {
```

```
        Merge$GFCM_fleetsegment[i]= as.character(CT[as.character(CT$LOA)==
                                                   as.character(Merge$VESSEL_LENGTH[i]) &
                                                   as.character(CT$GEAR_ACatch)== as.character(Merge$GEAR[i])
&
                                                   as.character(CT$FISHERY_ACatch)==
                                                   as.character(Merge$FISHERY[i]) ,]$Fleet_segment)
    } else {
        Merge$GFCM_fleetsegment[i]=""}
}

Merge_noempty=Merge[Merge$GFCM_fleetsegment!="",]

Merge_noempty_L=aggregate(Merge_noempty$LANDINGS,by=list(Merge_noempty$COUNTRY,
                                                         Merge_noempty$YEAR, Merge_noempty$GFCM_fleetsegment,
                                                         Merge_noempty$SPECIES),FUN="sum")
Merge_noempty_D=aggregate(Merge_noempty$DISCARDS,by=list(Merge_noempty$COUNTRY,
                                                         Merge_noempty$YEAR, Merge_noempty$GFCM_fleetsegment,
                                                         Merge_noempty$SPECIES),FUN="sum")

Merge=merge(Merge_noempty_L,Merge_noempty_D,by=c("Group.1","Group.2","Group.3",
                                                "Group.4"))
Merge$GSA=ACatch$SUB_REGION[1]

Merge=Merge[,c(1,2,7,3,4,5,6)]

Merge$Catch= rowSums(data.frame(col1=Merge[,6],col2=Merge[,7]),na.rm=T)
colnames(Merge) =c("Country","Reference_year","GSA","Fleet_segment","Species",
                   "Total_landing_per_species_(tons)","Total_discards_per_species_(tons)",
                   "Total_catch_per_species")

return(Merge)

}

#' TableVII.3.1 - GFCM DCRF datacall
#'
#' @param datacs CS table in RCG format
#' @param verbose boolean. If TRUE a message is printed.
#' @return TableVII31 (Biological information: Size at first maturity)
#' @export
#'
#' @examples TableVII31(RDBprocessing::data_ex)
#'
TableVII31<-function(datacs,verbose=F){


    datacs=check_cs_header(datacs)

ML_tab=ML_MEDBS(datacs,verbose=F)
species=unique(ML_tab$SPECIES)

ML_tab=ML_tab[as.character(ML_tab$SPECIES) %in% as.character(species),]
ML_tab_50=ML_tab[ML_tab$PRM<0.60 & ML_tab$PRM>0.4,]

L50=aggregate(ML_tab_50$LENGTHCLASS,
              by=list(ML_tab_50$COUNTRY,ML_tab_50$START_YEAR,ML_tab_50$END_YEAR,
                      ML_tab_50$SPECIES,ML_tab_50$SEX,ML_tab_50$AREA),FUN="mean")

#L50=L50[L50$Group.2<=YEAR & L50$Group.3>=YEAR,]    # selection on the year

#L50$AREA=ML_tab_50$AREA
#L50$YEAR=L50$START_YEAR
L50$Reference = "MEDBS"

#Final=L50[,c(1,8,7,4,5,6,9)]

Final=L50[,c(1,2,6,4,5,7,8)]

colnames(Final)=c("Country","Reference_year","GSA","Species","Sex","L50","Reference")

return(Final)

}
```