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 4.5.1

R Markdown automatic reporting tools

Walter Zupa, Isabella Bitetto

Partners involved:

COISPA, HCMR

Table of Contents

Acronyms

DCRF	Data Collection Reference Framework
FDI	Fisheries Dependent Information
GFCM	General Fisheries Commission for the Mediterranean
GSA	Geographical Subarea
HTML	hypertext markup language
LFD	Length Frequency Distribution
QC	Quality check
RCG	Regional Coordination Groups
RDBFIS	Regional Database and Fishery Information System
Rmd	R markdown
SOP	Sum of Products
YAML	YAML Ain't Markup Language

Executive summary

Task 4.5 is in charge of producing the Deliverable 4.5.1, whose objective is to develop R markdown tools to support automatic reporting for the RCG, MED & BS, FDI and GFCM data stored in the MED & BS RDBFIS database. The work done in the activities of this task takes advantage of the experience gained in the frame of the STREAM project (MARE/2016/22, *https://datacollection.jrc.ec.europa.eu/docs/regional-grants*) concerning data quality procedures to be applied both on detailed and aggregated data,and the work done in task 4.3 of this project for the creation of the RDBqc R package, collecting quality check functions for the above mentioned datacalls.

All the Rmd files were built to work both as standalone tools to check local data in csv format and as tools embedded in the MED & BSRDBFIS database's web application, working on data imported and fed by the database. The output of each of the above mentioned Rmd files is a detailed report in HTML format, to ease the visualization on a web browser of the reports generated by the web-based RDBFIS application.

1. Introduction

In the perspective of ensuring an acceptable data quality in the RDB data base and in accordance with the proposed functionalities for the Med&BS RDB included in the report of the ICES Steering Committee of the Regional Fisheries Database (ICES, 2020), specific software tools have been developed in the frame of the activities of the task 4.5 of Med & BS RDB project.

The work done in the activities of this project take advantage of the experiencegained in the frame of the STREAM project (MARE/2016/22, *https://datacollection.jrc.ec.europa.eu/docs/regional-grants*) concerning data quality procedures to be applied both on detailed and aggregated data. Indeed, STREAM project developed two different R markdown scripts to perform respectively *a priori* and *a posteriori* quality checks and to produce automatic reports. Most of the quality checks developed in the STREAM project have been coded in the form of the R functions and included, together with new *ad hoc* developed quality check functions, in the *RDBqc* R library, created under the activities of task 4.3 of the same project. The activity of task 4.5 described in this deliverable was focused on the development of a completely new set of R markdown scripts implementing the functions included in the above mentioned *RDBqc* R library, to perform *a priori* quality checks on data in the RCG data call formats and *a posteriori* quality checks on FDI, GFCM and DGMARE-MED datacall formats.

All the automatic reporting tools developed in the frame of the activities of task 4.5 and described in this deliverable will allow to facilitate, from a data-quality point of view, the cross-checks among the different formats, the harmonization of the data quality checks among the countries, and a more systematic and well-defined schedule for quality checks prior the relevant datacalls in order to reduce the risk of data failure. In turn, data quality will globally increase, facilitating the production of status reports, the awareness on fisheries data collected.

2. Working method

The aim of task 4.5 of the Med & BS RDB project was to produce scripts in R language to support data validation and quality checks in a structured way, creating automatic reports which collect the results of specific quality checks developed by the task 4.3 of the project. Such scripts, available in an open-access GitHub repository: *https://github.com/COISPA/RDBqc/tree/main/RMD%20reports* (**APPENDIX 1-4**), will use all the quality checks that are included in the *RDBqc* R library of functions (https://github.com/COISPA/RDBqc/tree/main/RDBqc).

The R Markdown files works in Rstudio environment. Markdown is a lightweight formatting language for authoring documents in HTML and other different file formats, such as PDF, and MS Word. This language was created by John Gruber (2004) as text-to-HTML conversion tool for web writers.

R Markdown provides an authoring framework for data science based on markdown language developed on the literature programming idea (Knuth, 1984) in which code and text are integrated in the same document. R markdown files are plain text fully reproducible documents with the extension ".Rmd" used to create dynamic documents with R language in which tables and graphs could be included. The files are structured in "chunks" containing R code that are interpreted by the *rmarkdown* library (Allaire et al., 2022). Rmarkdown uses the "*knitr*" library (Xie, 2022) to process the code included in the document, creates the output (with tables and plots too) and includes everything in the R markdown file dynamically generated. Afterward, the document is transformed in the final document format (e.g. HTML) by mean the *Pandoc* software.

Four specific R markdown automatic report files have been produced to perform both *a priori* and *a posteriori* checks to carry out data validation and quality checks on detailed data and output data, respectively, in different data calls. In particular, in the frame of task 4.5 the following completely new Rmd report files have been developed:

- Rmd R Markdown report file for the *a priori* QC on the RCG format,
- Rmd R Markdown report file for the *a posteriori* QC on DGMARE Med&BS format,
- Rmd R Markdown report file for the *a posteriori* QC on FDI format,
- Rmd R Markdown report file for the *a posteriori* QC on GFCM DCRF format.

The output of each of the above mentioned Rmd files is a detailed report in HTML format of the checks carried out on the selected tables (see also on **ANNEX XVII (Tutorial Rmarkdown)**).

The Rmd scripts for the automatic reporting working on data in the format of the four datacall were built with R software (R core Team, 2022) version 4.1.2 (64-bit) and tested in Rstudio software (Rstudio Team, 2020; version 2022.12.0 Build 353) that is an integrated development environment (IDE) for R. Thanks to the multi-platform user interface of R, all the Rmds have been successfully tested in both Windows and Linux (Debian) platforms.

All the Rmd files were built to work both as standalone tools, to check local data in csv format, and as tools embedded in the RDBFIS database's web application, working on data imported and fed by the database.

Finally, Rmd files were tested through RDBFIS web application on the GSA 18 data in occasion of the "Med&BS RDBFIS WP5 Meetings, testing phase - 2nd WORKSHOP" held the 23rd January 2023.

2.1.1 Structure of the Rmd files

The code included in the Rmd files is divided in different sections. The first one is the YAML (YAML Ain't Markup Language) section (Figure 1) which contains the metadata of the document. YAML is a human-friendly data serialization language for all programming languages. In this section the title of the document, the authors, the data of execution and other details are defined by key:value pairs. The first section of the document is also used to define the html template to be used for the document graphic layout. All the produced Rdm files use the same template. The YAML section was parametrized to generate of the table of content, included at the beginning of the report, in which 5 different levels of depth are defined. Each section included in the report is automatically generated.

```
---
title: "| ![](logo.png){width=6in} \\vspace{1in} \n\n\n\n`r format(params$term)`
Report\n\n\n\n
  \\vspace{0.1in} "

date: "Compiled on `r format(Sys.time(), '%d/%m/%Y, %H:%M')`"
subtitle: ''

output:
  html_document:
    template: RDBFIS_report_template.html
    toc: true
    toc-title: "Table of contents"
    toc_depth: 5
    number_sections: true
    collapsed: true
    df_print: paged
params:
  term: RCG datacall
---
```

Figure 1. Screenshot of an example of the YAML section of the RCG datacall Rmd file.

The rest of the document is composed by the alternation of textual parts coded in Markdown language and "chunks" of R code.

After the YAML section, the Rmd files include a section for the initialization (Figure 2) of the environment by mean the loading of all the required libraries needed to run the data analysis and the definition of other environmental parameters. The following libraries are required to run the Rmd scripts: *RDBqc, knitr, markdown, kableExtra, dplyr, ggplot2, rworldmap, sp, rworldxtra, pander, data.table, grDevices, magrittr, tictoc, tidyverse, fishmethods, tidyr, gridExtra, outliers*(Allaire et al., 2022; Auguie, 2017; Bitetto and Zupa, 2022; Daróczi and Tsegelskyi, 2021; Dowle and Srinivasan, 2021; Izrailev,2021; Komsta, 2022; Milton and Wickham, 2020; Nelson, 2021; Pebesma and Bivand, 2005; R core Team, 2022; South, 2011; South, 2012; Wickham, 2016; Wickham et al., 2019; Wickham, 2021; Wickham et al., 2021; Xie, 2022; Zhu, 2021).

```
```{r setup, include=FALSE}
knitr::opts_chunk$set(
    echo = FALSE,
    message = FALSE,
    warning = FALSE
)

# loading needed libraries
lib <- c("RDBqc","knitr","kableExtra","dplyr", "ggplot2", "rworldmap", "sp",
"rworldxtra", "pander", "data.table", "grDevices", "magrittr", "tictoc","tidyverse",
"fishmethods","tidyr","gridExtra","outliers")
lapply(lib, require, character.only = TRUE)
```
```

Figure 2. Screenshot of an example of the initialization section

Afterward, next section is the only one that could be modified by the user, in case the Rmd files are used as standalone tool to process local .csv files. User can define the pertinent source files for the data tables to be processed for the selected datacall. In Figure 3 is reported an example of the code included in the Rmd

file of the RCG datacall where the user can either select the input table/s (e.g. CS and CL tables). The user can also define to apply some filters to the source data files, such as a pool of species, the GSA, and other, according to the datacall file considered. All the Rmd files work on data of only one country per time.

```
<!-- USER DEFINED SOURCE FILES -->

```{r setup, include=FALSE}
# reading files
# UNCOMMENT  THE FOLLOWING LINES FOR STAN-ALONE USE
# SELECT THE APPROPRIATE FILE PATHS ON THE LOCAL FOLDERS
CS <- read.csv("./sampling.csv")
CL <- read.csv("./landings.csv")


SPs <- c("Merluccius merluccius")
MS <- "ITA"
GSAs <- c("GSA99")
stages_for_immatures <- c("1", "2a")
```
```

Figure 3. Example of the user defined section of the RCG datacall Rmd file.

The filters will be applied to all the selected data table. The user can choose also to perform the analysis applying either no filter or only some selected filters, commenting the relative code in the chunk.

All the code included in the Rmd file after the user defined chunk should not be modified. Before the data tables are processed by the Rmd, a dedicated chunk checks whether any filter was defined by the user. In case no filters were set, the unique combination of the main variables (e.g., GSA, species) are extracted by the pertinent defined tables.

All the tables of the selected datacall can be processed in a unique run of the script following a predefined order of tables and checks. In case for a given datacall a source table is not defined, all the checks available for that table are skipped and a message is reported in the final report. In this way the analysis is conducted only on the available data, avoiding the code crash.

2.1.2 RCG datacall

The Rmd file to analyse the RCG data (**APPENDIX I**) allows to apply checks on both sampling table (CS) and landing table (CL). The user can define the following filter: country (MS), the geographical subareas (GSA) and the species. Furthermore, the user can define the maturity stages to be considered as immature for the check of the maturity stages composition on CS table. The consistency of landing table (CL) is checked for the temporal, spatial and metier coverage for each species and GSA of the selected country. The consistency checks implemented for the sampling data (CS) are:

- Summary of the reported trips,
- Consistency of trip location,
- Summary of individual measurements,
- Consistency of LFD by year,
- Consistency of LFD by year and commercial category,
- Consistency of sex and maturity stage,
- Check of maturity ogive,

- Consistency of length-weight relationship,
- Consistency of age-length relationship.

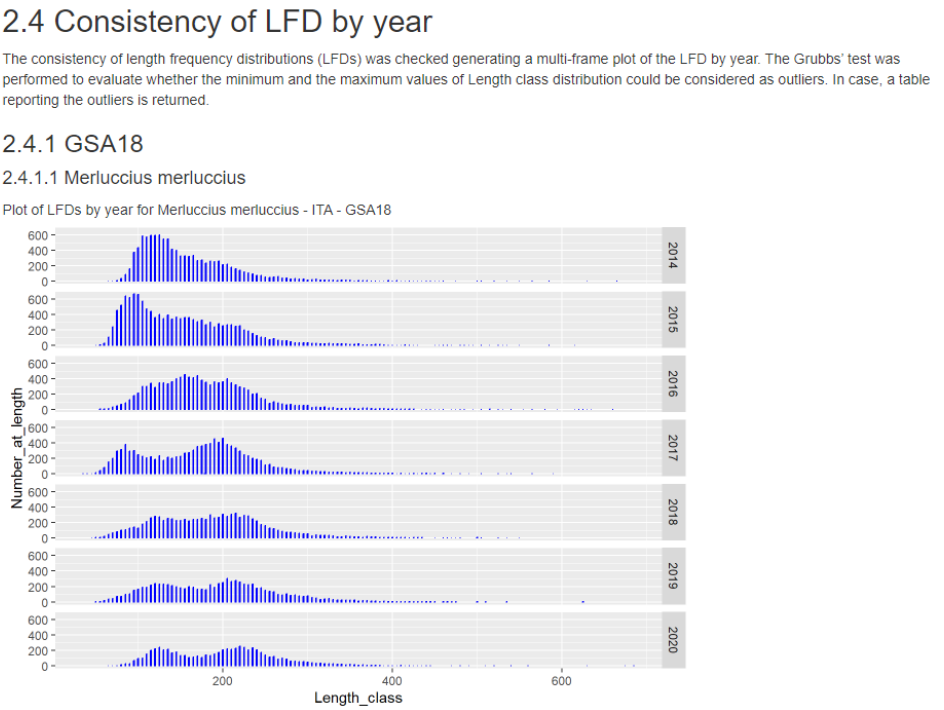

Figure 4. Example plot of LFDs by yearderived from RCG datacall CS table.

2.1.3 MED & BS datacall

The Rmd file to analyse the MED& BS data (**APPENDIX II**) allows to apply checks on 9 different type of tables. The user can define the following filter: country (MS), the geographical subareas (GSA) and the species. The list of the checks implemented for each table is here reported:

Catch table

- Presence of duplicated records,
- Coverage of catch data,
- Check of Sum of Products (SOP).

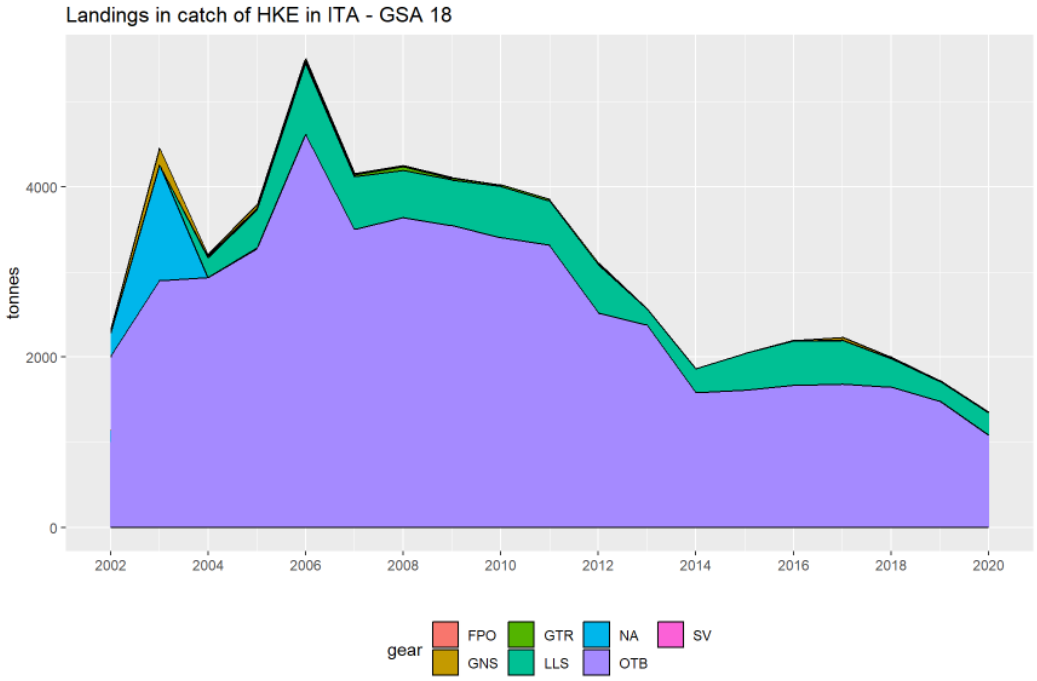

Figure 5. Example plot of landings coveragederived from Med&BS datacall catch table.

Landing table

- Presence of duplicated records,
- Coverage of landings data,
- Checks on landing volumes,
- Comparison of landing volumes aggregated by fishery accounting for time frame,
- Kolmogorov-Smirnov test on cumulative length distributions,
- Check the presence of length classes numbers with zeros values in landings,
- Consistency of main length size indicators,
- Check the presence of years with missing length distribution,
- Check the presence of landing weights with zero values in landings,
- Check the presence of landing weights with -1 values in landings,
- Consistency of mean weight by year, gear and fishery aggregation,
- Consistency of total landing time series by quarter,
- Consistency of total landing time series by gear and fishery.

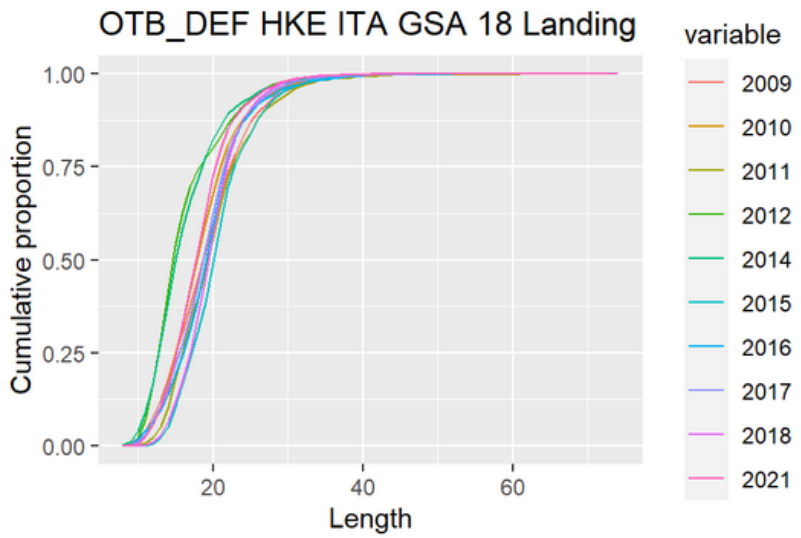

Figure 6. Example of cumulative length distributions derived fromthe landing table of MED & BS datacall.

Discard table

- Presence of duplicated records,
- Coverage of discards data,
- Comparison between discards in weight by quarter and year,
- Comparison between discards in weight by quarter and year, accounting for fishery,
- Consistency of mean weight by year, gear and fishery aggregation,
- Kolmogorov-Smirnov test on cumulative length distributions,
- Check the presence of length classes numbers with zero values in discards,
- Consistency of main length size indicators,
- Check the presence of years with missing length distribution,
- Check the presence of discard weights with zero values in discards,
- Check the presence of discard weights with -1 values in discards,
- Consistency of total discard time series by quarter,
- Consistency of total discard time series by gear and fishery.

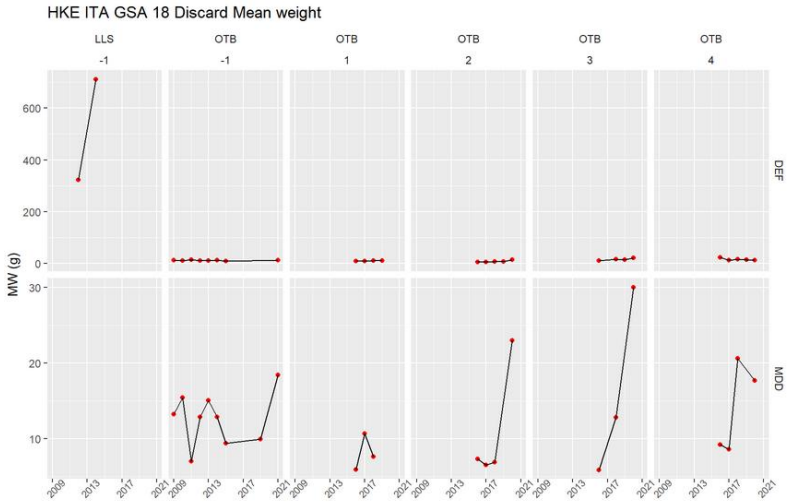

Figure 7. Example plot of mean landing weight by year, gear and fisheryderived fromdiscard table of MED & BS datacall

Maturity at age (MA) table

- Plots of Maturity at Age (MA) data

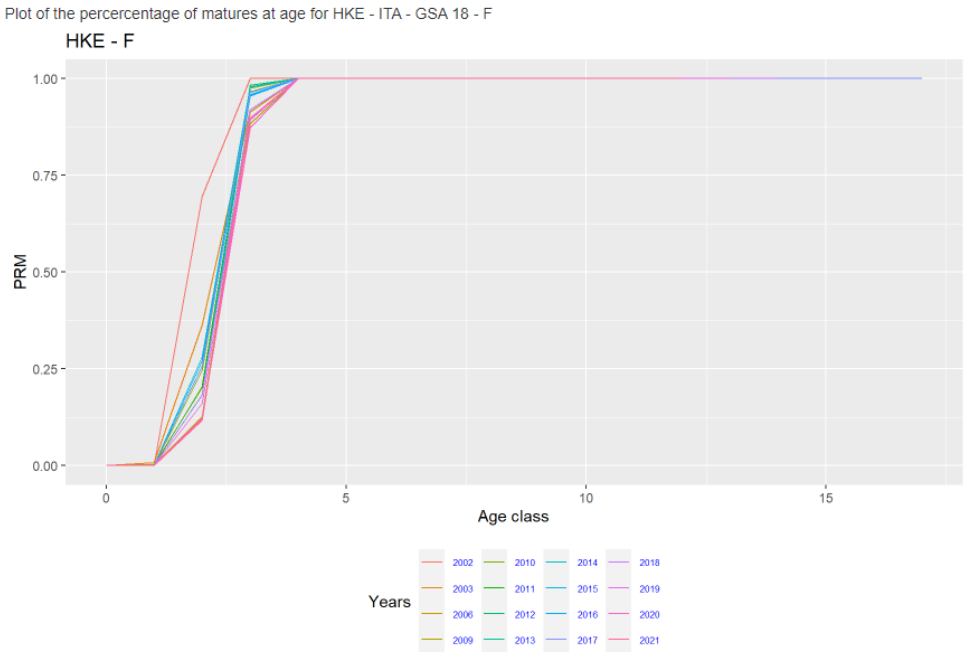

Figure 8. Example plot of percentage of maturity at age data derived fromMED & BS datacallmaturity at age table.

Maturity at Length (ML) table

- Plots of Maturity at Length (ML) data.

Sex Ratio at length (SRL) table

- Plots of Sex Ratio at length (SRL) data.

Sex Ratio at age (SRA) table

- Plots of Sex Ratio at age (SRA) data.

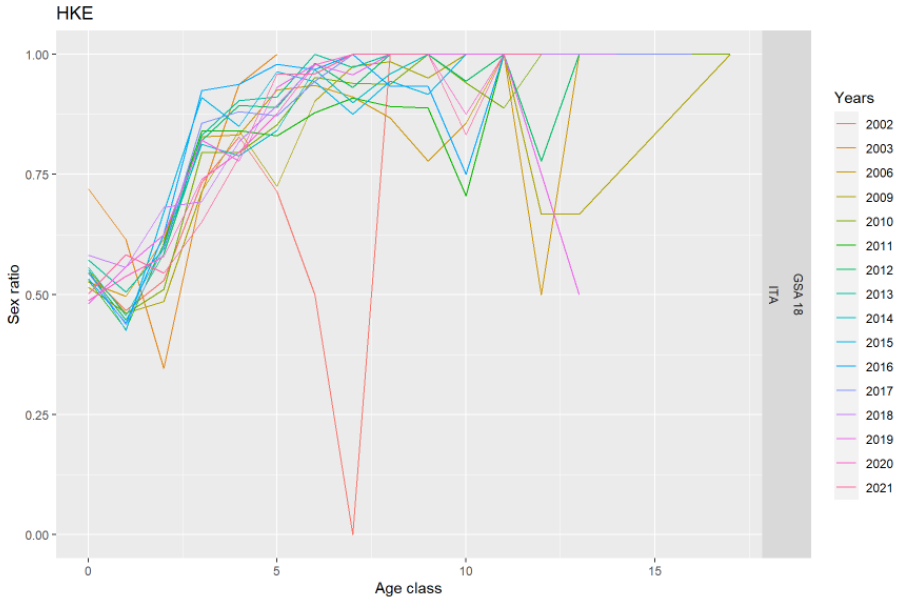

Figure 9. Example plot of sex-ratio data by year derived fromMED & BS datacall sex-ratio at age table.

Growth parameters (GP) table

- plots of the growth parameter,
- Plots of length-weight (LW) functions.

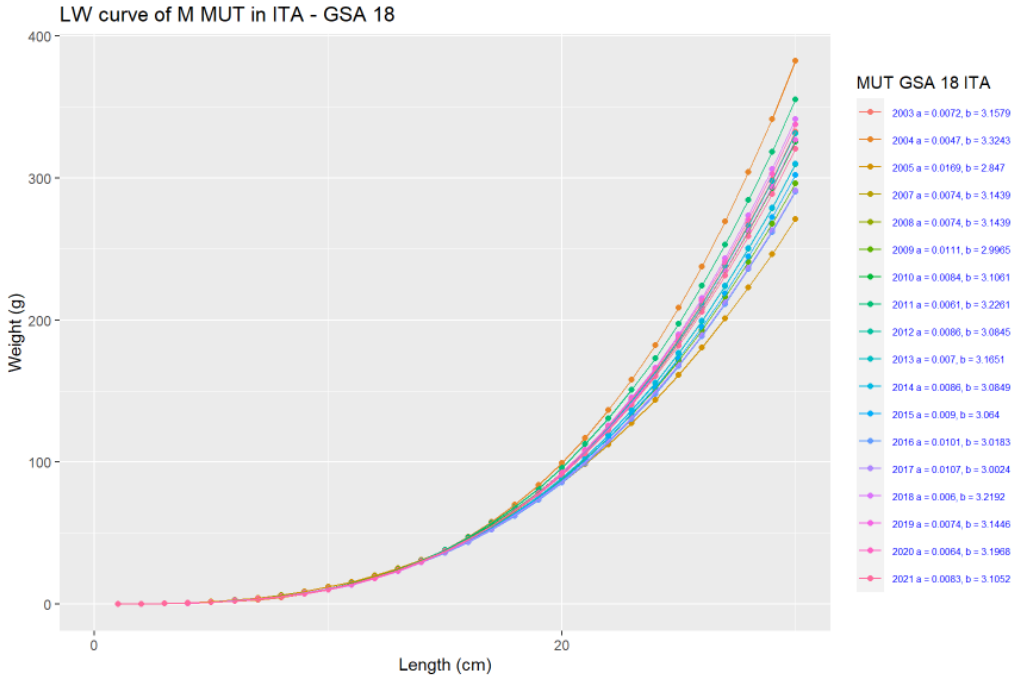

Figure 10. Example plot of length-weight curve derived fromMED & BS datacall GPtable.

Age-Length Keys (ALK) table

- Plots of Age-Length Keys (ALK) data.

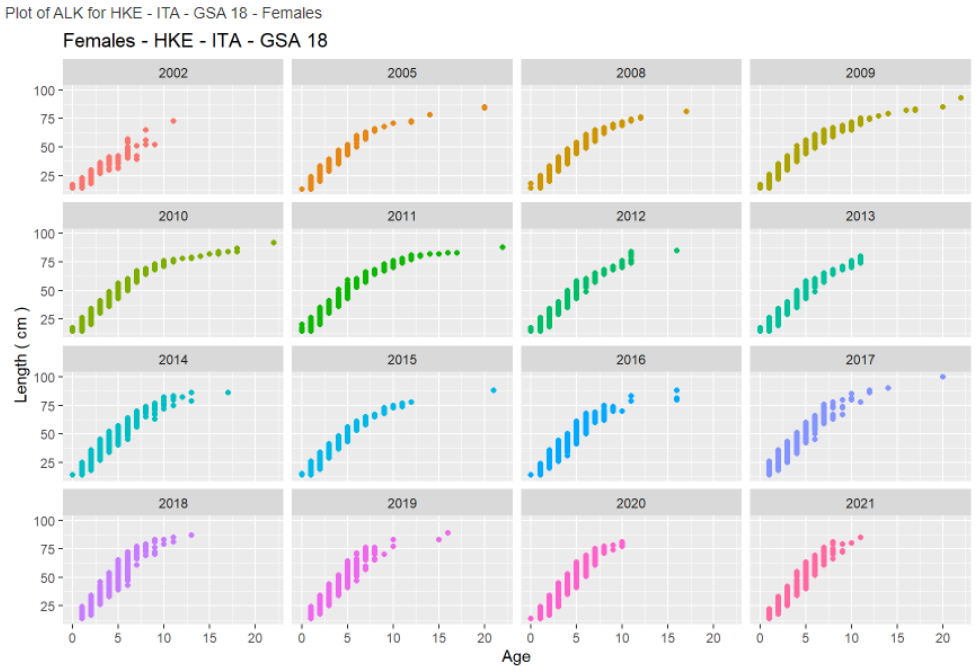

Figure 11. Example plot of age-length key data derived fromMED & BS datacall ALKtable.

2.1.4 FDI datacall

The Rmd file to analyse the FDI data (**APPENDIX III**) allows to apply checks on 5 different tables relevant for the Mediterranean and Black Sea and allows also to perform cross-checks between these tables (accordingto the checks described in the STECF EWG 21-12 report). The user can define the following filter: country (MS), the geographical subareas (GSA), the species, the fishing techniques, the vessel length categories and the metiers. The list of the checks implemented for each table is here reported:

Catch data (Table A)

- Check the presence of empty fields,
- Check the presence of duplicated records,
- Coverage of catch data by GSA and year,
- Cross-coverage of landing and discard data,
- Consistency of landing data and discard data availability by year,
- Check on species value,
- Check on species' prices trend.

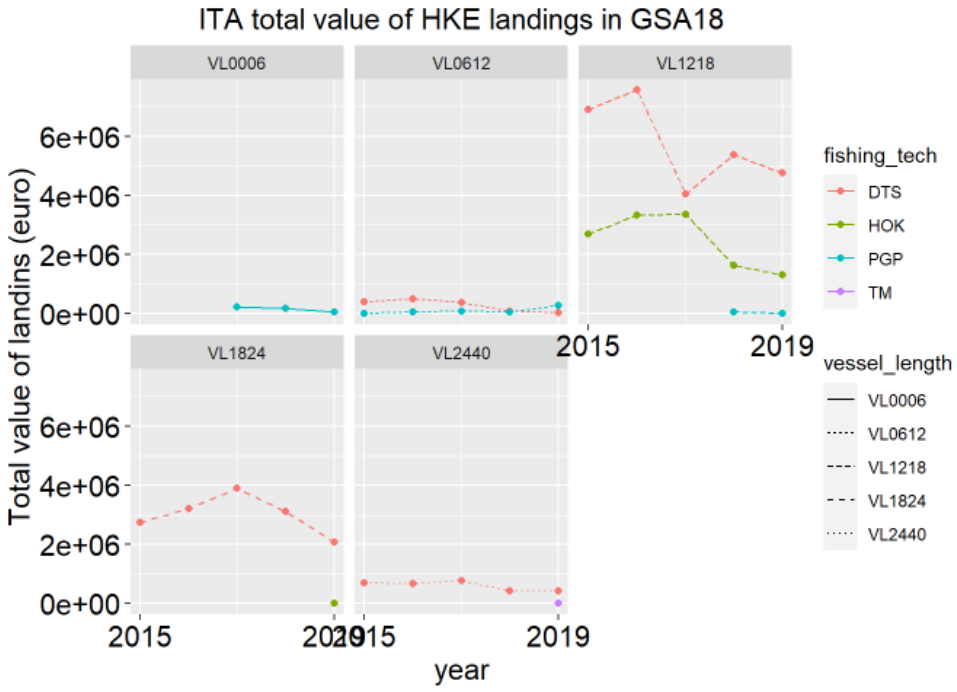

Figure 12. Example plot of landings value derived from FDI datacall catch data (Table A).

Effort data (table G)

- Check the presence of empty fields,
- Check the presence of duplicated records,
- Coverage of fishing effort data.

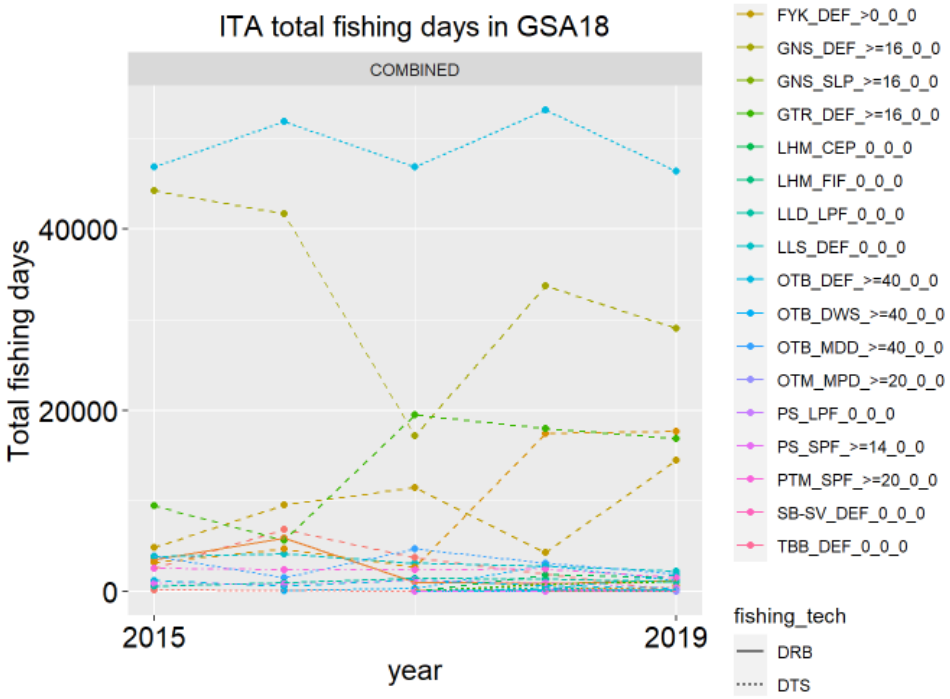

Figure 13.Example plot of total fishing days data derived from FDI datacall table G.

Landings by rectangle data (table H)

- Check the presence of empty fields,
- Check the presence of duplicated records,
- Coverage of landings data by rectangle, GSA and year,
- Consistency of spatial data by rectangle,
- Check of compatibility of the geographical coordinates with rectangle type.

Effort by rectangle data (table I)

- Check the presence of empty fields,
- Check the presence of duplicated records,
- Coverage of effort by rectangle, GSA and year,
- Consistency of spatial data by rectangle,
- Check of compatibility of the geographical coordinates with rectangle type.

Capacity and fleet segment effort data (table J)

- Check the presence of empty fields,
- Check the presence of duplicated records,
- Coverage of data in Table J,
- Consistency of vessel length with vessel length category.

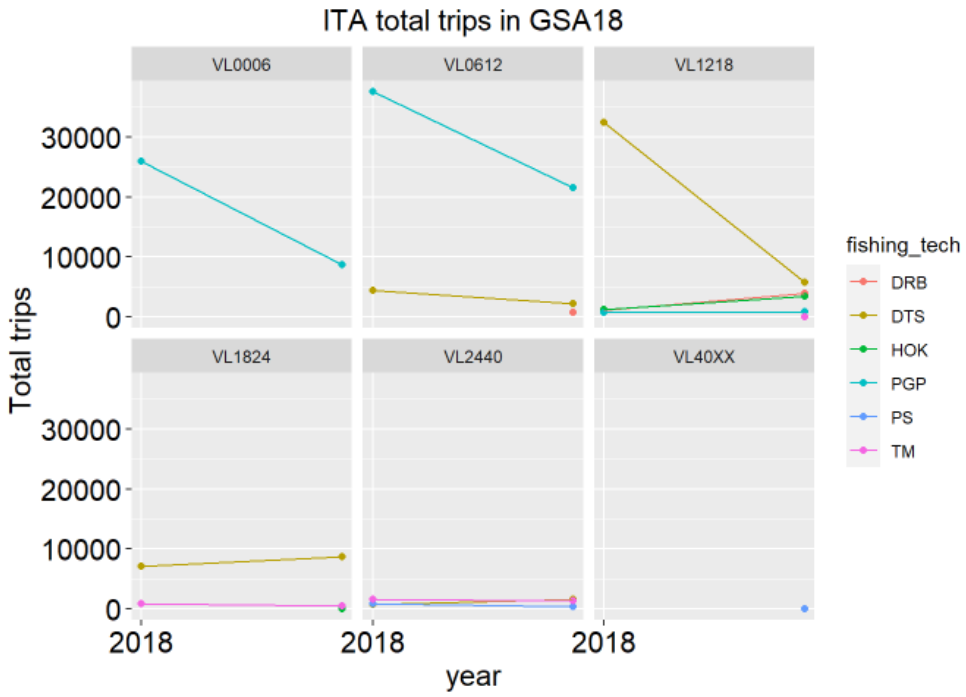

Figure 14. Example plot of total trip data derived from FDI datacall table J.

Cross-checks

- Comparison of total fishing days in tables G and I,
- Comparison of landing volumes in tables A and H,
- Consistency of number of vessels in table G and J,
- Consistency of landings and effort in tables A and G,
- Consistency of landings and spatial landings in table A and H,
- Consistency of effort and spatial effort in tables G and I.

2.1.5 GFCM datacall

The Rmd file to analyse the GFCM data (**APPENDIX IV**) allows to apply quality checks on 5 different type of tables relevant for the Mediterranean and Black Sea. The user can define the following filter: country (MS), the geographical subareas (GSA), the species, and the fleet segment. The list of the checks implemented for each table is here reported:

task II.2

- Presence of empty records,
- Coverage of GFCM Catch table,

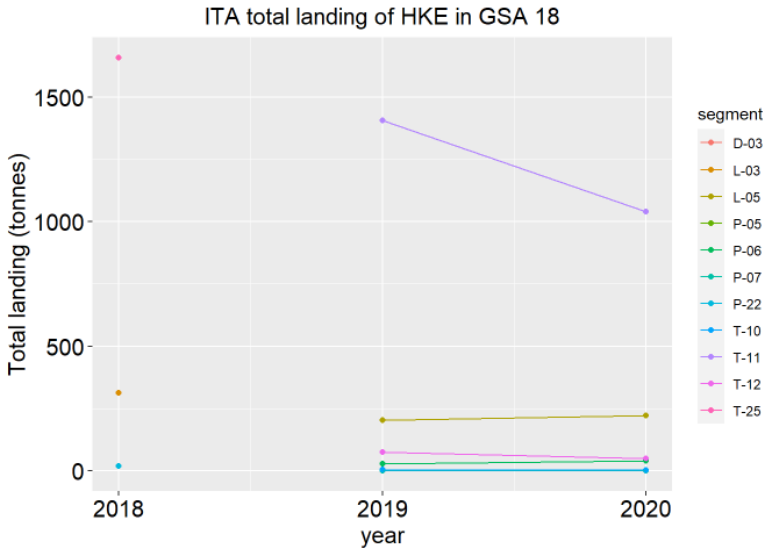

Figure 15. Example plot oftotal landing data by year and segment derived from GFCM catch Table II.2.

task III

- Presence of duplicated records,
- Presence of empty records,
- Coverage of GFCM Incidental catch of vulnerable species.

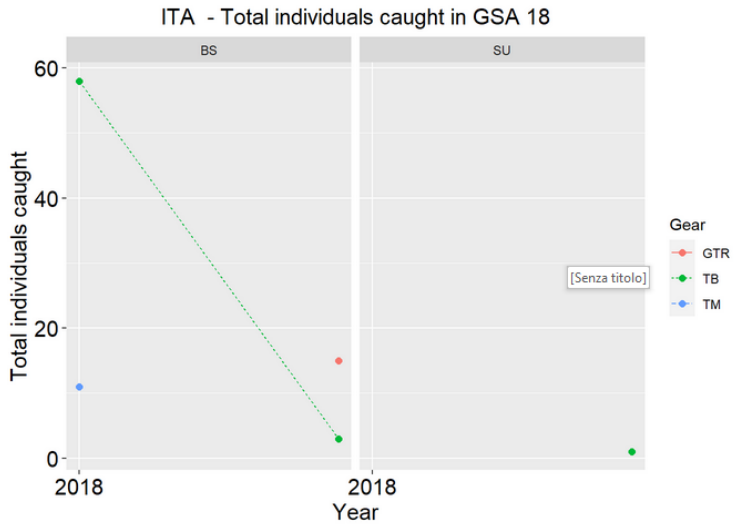

Figure 16. Example plot of the number of individuals caught by year and gear derived from GFCM incidental catch. Table III.

task VII.2

- Presence of duplicated records,
- Presence of empty records,
- Consistency of Length-Weight relationship,

- Consistency of Length Frequency Distribution in Biological information.

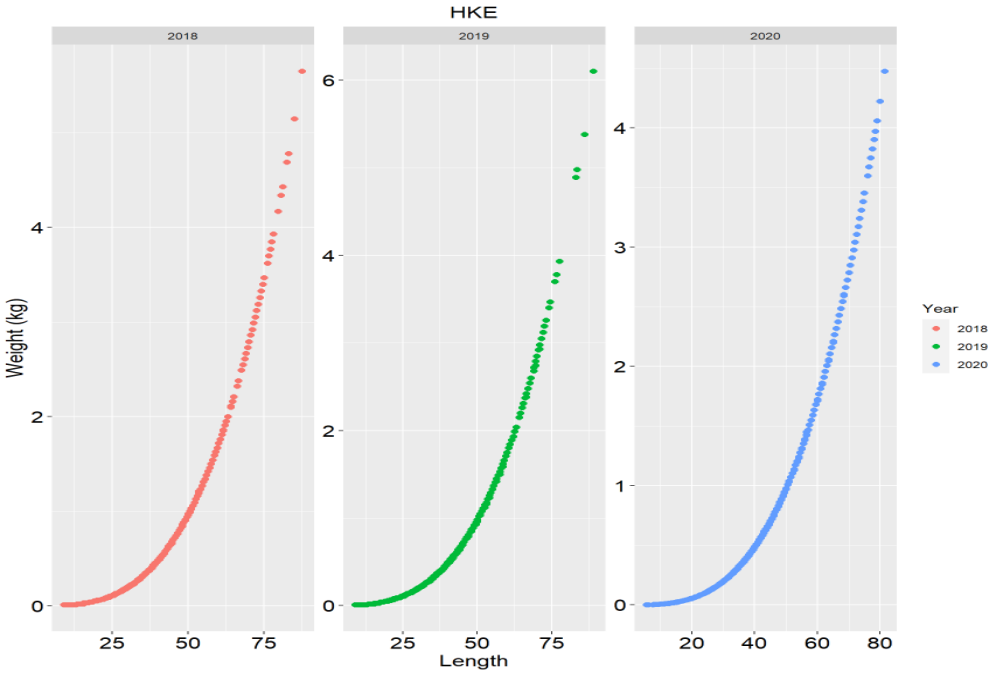

Figure 17. Example plot of length-weight relationships by year derived from GFCM length data Table VII.2

task VII.3.1

- Presence of duplicated records,
- Presence of empty records,
- Consistency of size at first maturity.

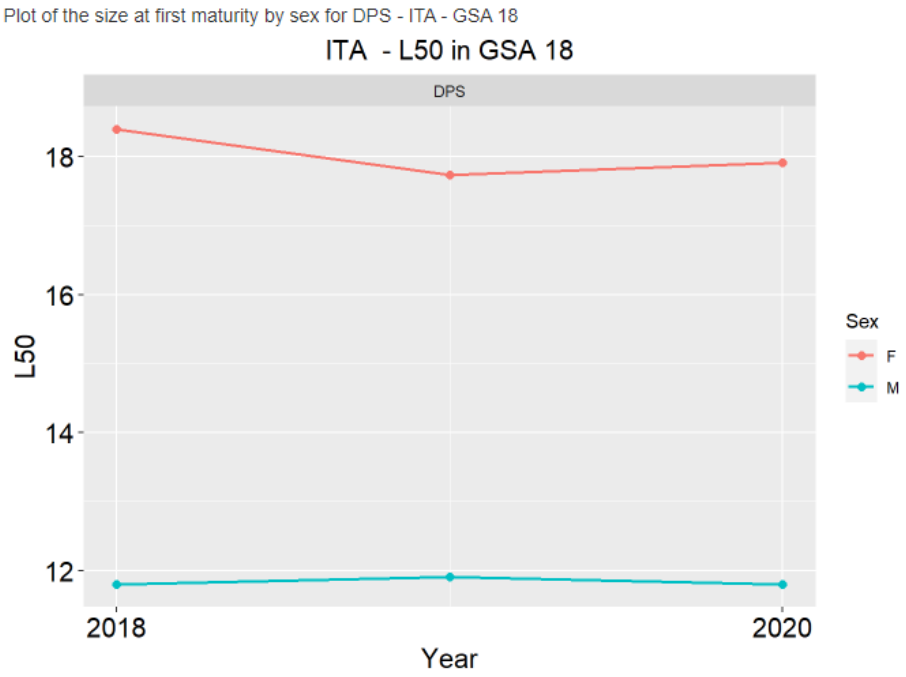

Figure 18. Example plot of the time series of size at first maturity by sex derived from GFCM size at first maturity Table VII.3.1.

task VII.3.2

- Presence of duplicated records,
- Presence of empty records,
- Consistency of sex and maturity stages,
- Consistency of Length-Weight relationship.

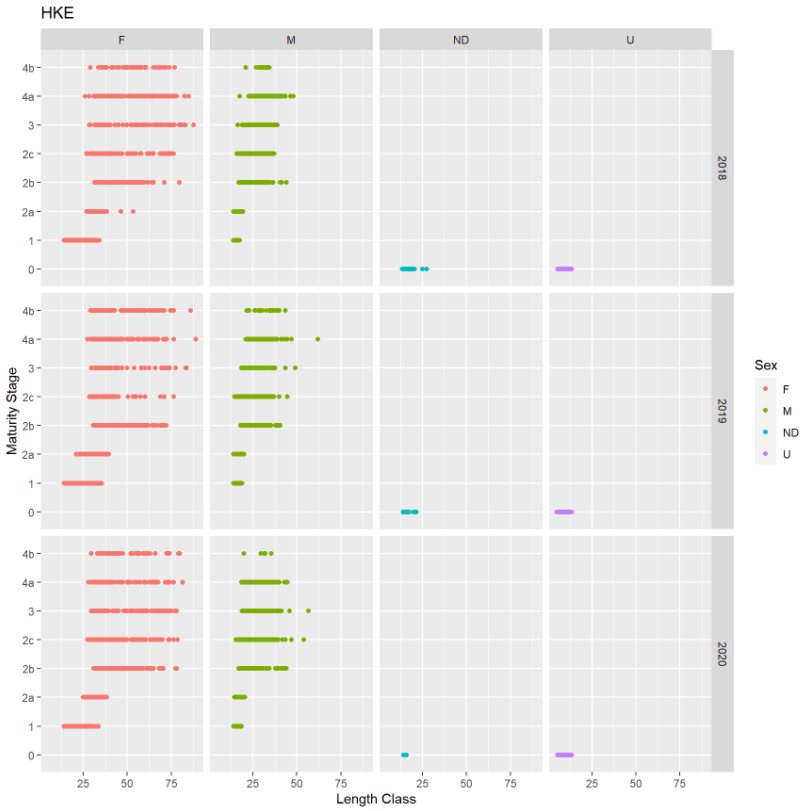

Figure 19. Example plot of the length distribution by sex, year and maturity stage derived from GFCM maturity data table VII.3.2

3. Conclusions

The activity conducted in Task 4.5allowed to produce four specific R markdown automatic report files to perform both *a priori* and *a posteriori* checks to carry out data validation and quality checks respectively on detailed data and output data in different data calls required by end-users. The work done in this task took advantage of the experience gained in the frame of the STREAM project (MARE/2016/22, *https://datacollection.jrc.ec.europa.eu/docs/regional-grants*) concerning data quality procedures to be applied both on detailed and aggregated data and of the work conducted in task 4.3, for the creation of the RDBqc R library.

The advantage of new tools supporting the production of status reports is expected to globally increase the awareness on fisheries data collected, facilitating the control of the consistency of data quality among the datacalls.

The great advantage provided by these new Rmd report files is the possibility to apply the quality checks offline, on data resident on a local PC, using exactly the same functions used to check data in the MED & BS RDBFIS. This will improve the harmonization of the data quality checks among the countries, and a more systematic and well-defined schedule for quality checks prior the relevant datacalls in order to reduce the risk of data failure.

A further advantage of these tools is represented by the ease of updating and modification which makes the implementation of new functions particularly versatile, allowing continuous improvement and expansion in the future.

4. References

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

Auguie, B. (2017). gridExtra: Miscellaneous Functions for"Grid" Graphics. R package version 2.3.https://CRAN.R-project.org/package=gridExtra

Bitetto I., Zupa W. (2022). RDBqc: Quality checkfunctions for RDBFIS. R package version 0.0.14.

Daróczi, G., Tsegelskyi, R. (2021). pander: An R 'Pandoc'Writer. R package version 0.6.4.https://CRAN.R-project.org/package=pander

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

Gruber, J. (2004). https://daringfireball.net/projects/markdown/

Izrailev, S. (2021). tictoc: Functions for Timing R Scripts, asWell as Implementations of Stack and List Structures. R package version 1.0.1. https://CRAN.R-project.org/package=tictoc

Knuth, D. E. (1984). "Literate Programming". The Computer Journal. British Computer Society. 27 (2): 97–111. doi:10.1093/comjnl/27.2.97

Komsta, L. (2022). outliers: Tests for Outliers. R package version 0.15. https://CRAN.R-project.org/package=outliers

Milton, S. B., Wickham, H. (2020). magrittr: AForward-Pipe Operator for R. R package version 2.0.1.https://CRAN.R-project.org/package=magrittr

Nelson, G. A. (2021). fishmethods: Fishery Science Methods andModels. R package version 1.11-2.https://CRAN.R-project.org/package=fishmethods

Pebesma, E.J., Bivand, R.S.(2005). Classes and methods for spatialdata in R. R News 5 (2), https://cran.r-project.org/doc/Rnews/.

R Core Team (2022). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria.URL https://www.R-project.org/

RStudio Team (2020). RStudio: Integrated Development for R. RStudio, PBC, Boston, MA URL http://www.rstudio.com/

South, A.(2011) rworldmap: A New R package for Mapping GlobalData. The R Journal Vol. 3/1 : 35-43.

South, A. (2012). rworldxtra: Country boundaries at highresolution. R package version 1.01. https://CRAN.R-project.org/package=rworldxtra

Wickham et al., (2019). Welcome to the tidyverse. Journal of Open Source Software, 4(43), 1686, https://doi.org/10.21105/joss.01686

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

Wickham, H.(2016). ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

Wickham, H., Romain, F., Lionel, H., Kirill, M. (2021). dplyr: A Grammar of Data Manipulation. R package version1.0.7. https://CRAN.R-project.org/package=dplyr

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

Zhu, H (2021). kableExtra: Construct Complex Table with 'kable'and Pipe Syntax. R package version 1.3.4.

5. APPENDIX I – RCG Rmd script

````
---
title: "| ![](logo.png){width=6in} \\vspace{1in} \n\n\n\n`r format(params$term)` Report\n\n\n\n\n
  \\vspace{0.1in} "

date: "Compiled on `r format(Sys.time(), '%d/%m/%Y, %H:%M')`"
subtitle: ''

output:
  html_document:
    template: RDBFIS_report_template.html
    toc: true
    toc-title: "Table of contents"
    toc_depth: 5
    number_sections: true
    collapsed: true
    df_print: paged
params:
  term: RCG datacall
---
<!-- INITIALIZATION -->

```{r inizialization, include=FALSE}
knitr::opts_chunk$set(
    echo = FALSE,
    message = FALSE,
    warning = FALSE
)

# loading needed libraries
lib <- c("RDBqc","knitr","kableExtra","dplyr", "ggplot2", "rworldmap", "sp", "rworldxtra", "pander", "data.table",
"grDevices", "magrittr", "tictoc","tidyverse", "fishmethods","tidyr","gridExtra","outliers")
lapply(lib, require, character.only = TRUE)
```

<!-- USER DEFINED SOURCE FILES -->

```{r setup, include=FALSE}
# reading files
# UNCOMMENT  THE FOLLOWING LINES FOR STAN-ALONE USE
# SELECT THE APPROPRIATE FILE PATHS ON THE LOCAL FOLDERS
CS <- read.csv("./sampling.csv")
CL <- read.csv("./landings.csv")

SPs <- c("Merluccius merluccius")
MS <- "ITA"
GSAs <- c("GSA99")
stages_for_immatures <- c("1", "2a")
```

<!-- DO NOT MODIFY THE FOLLOWING PART OF THE CODE -->

```{r setup2, include=FALSE}
# check the presence of user defined filter for species (SPs)
if (exists("SPs")) {
  if (length(SPs) == 1 & is.na(SPs[1])) {
    user_SP <- FALSE
  } else {
    user_SP <- TRUE
  }
} else {
  user_SP <- FALSE
}

# check the presence of user defined filter for member state (MS)
if (exists("MS")) {
if(length(MS)==1 & is.na(MS[1])) {
  user_MS = FALSE
} else {
  user_MS = TRUE
}
} else {
  user_MS <- FALSE
}

# check the presence of user defined filter for sub-area (GSAs)
if (exists("GSAs")) {
if(length(GSAs)==1 & is.na(GSAs[1])) {
  user_GSA = FALSE
} else {
  user_GSA = TRUE
}
} else {
  user_GSA <- FALSE
````
````

```
}

# check the presence of user defined filter for maturity stages for
# immature individuals (stages_for_immatures)
if (exists("stages_for_immatures")) {
  if (length(stages_for_immatures) == 1 & is.na(stages_for_immatures[1])) {
    user_stages_for_immatures <- FALSE
  } else {
    user_stages_for_immatures <- TRUE
  }
} else {
  user_stages_for_immatures <- FALSE
}

```

\newpage

<!-- CL -->

```{r check_CL, results = 'asis'}

if (exists("CL")) {
  if (class(CL)=="data.frame") {
    if(nrow(CL)>0){
      check_CL <- TRUE
      cat(paste0("\n# Landing data table\n"))
      if(!user_SP){
        SPs <- unique(CL$species)
      }
      if(!user_MS){
        MS  <- unique(CL$flag_country)
      }
      if(!user_GSA){
        GSAs <- unique(CL$area)
      }

    }
  }
} else {
  check_CL <- FALSE
  cat("\n No Landing data available \n")
}

```

```{r RCG_check_CL, results = 'asis', fig.height=6, fig.width=9}
if (check_CL) {

  cat("\n## Consistency of data in CL table\n")
  cat("\nThe consistency of CL table was checked plotting data for temporal, spatial and metier coverage for the
given species:\n")

for (g in 1:length(GSAs)) {
      cat(paste0("\n### ", GSAs[g], "\n"))
      s=1
      for (s in 1:length(SPs)) {
        cat(paste0("\n#### ", SPs[s], "\n"))
        suppressMessages(res <- RCG_check_CL(data=CL, MS = MS, GSA = GSAs[g], SP=SPs[s]))
        if (class(res)== "list" & length(res)>0){
          # temp_covL
            cat(paste0("\n_Sum of Landings by year, quarter and month for ",SPs[s]," in ",GSAs[g],"_"))
            print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width=TRUE)
)

          # temp_covLV
            cat(paste0("\n_Sum of Landing value by year, quarter and month for ",SPs[s]," in ",GSAs[g],"_"))
            print(res[[2]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width=TRUE)
)

          # spat_covL
            cat(paste0("\n_Sum of landings by LandCtry, VslFlgCtry,  Area, Rect, SubRect, Harbour for ",SPs[s],"
in ",GSAs[g],"_"))
            print(res[[3]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width=TRUE)
)

          # spat_covLV
            cat(paste0("\n_Sum of landing value by LandCtry, VslFlgCtry,   Area, Rect, SubRect, Harbour for
",SPs[s]," in ",GSAs[g],"_"))
            print(res[[4]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width=TRUE)
)

          # spe_cov_L
            cat(paste0("\n_Sum of landings by Year, Species, foCatEu5, foCatEu6 for ",SPs[s]," in ",GSAs[g],"_"))
            print(res[[5]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width=TRUE)
)
```

```
          # spe_cov_LV
            cat(paste0("\n_Sum of landing value by Year, Species, foCatEu5, foCatEu6 for ",SPs[s]," in
",GSAs[g],"_"))
            print(res[[6]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width=TRUE)
)

          # p
          cat("\n\n\n")
          cat(paste0("\nPlot of Sum of landings by Year, Species, foCatEu6 for ",SPs[s]," - ",MS," -
",GSAs[g],"\n"))
print(res[[7]])
          cat("\n\n\n")

} else {
          cat(paste0("\n\n\n- No data available for ",SPs[s]," - ",MS," - ",GSAs[g],"\n"))
        }
      }
}
} # check_CL
```

\newpage

```
<!-- CS -->

```{r check_CS, results = 'asis'}

if (exists("CS")) {
  if (class(CS)=="data.frame") {
    if(nrow(CS)>0){
      check_CS <- TRUE
      cat(paste0("\n# Detailed data table\n"))

      CS <- check_cs_header(CS)

      if(!user_SP){
        SPs <- unique(CS$Species)
      }
      if(!user_MS){
        MS  <- unique(CS$Flag_country)
      }
      if(!user_GSA){
        GSAs <- unique(CS$Area)
      }
      if(!user_stages_for_immatures){
        stages_for_immatures <- c("0", "1", "2a")
      }
    }
  }
} else {
  check_CS <- FALSE
  cat("\n No detailed data available \n")
}
```

```{r RCG_summarize_trips, results = 'asis', fig.height=6, fig.width=9}
if (check_CS) {
    cat("\n## Summary of trips\n")
    cat("\nBiological data were analysed to provide a summary table reporting the number of trips/hauls monitored
by year, port, metier and sampling method.\n")

for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
      s=2
      for (s in 1:length(SPs)) {
        cat(paste0("\n#### ", SPs[s], "\n"))
        res <- suppressMessages(RCG_summarize_trips(data = CS, MS = MS, GSA = GSAs[g], SP = SPs[s],verbose=FALSE))
        if (!is.null(res)){
        if (nrow(res)>0){
            cat(paste0("\n_Summary table reporting the number of trips/hauls monitored by year by port, metier,
sampling method for ",SPs[s]," in ",GSAs[g],"_"))
        print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width=TRUE) )
        } else {
            cat(paste0("\n- No data available for : ",SPs[s]," - ",MS," - ",GSAs[g], "\n"))
        }
        } else {
            cat(paste0("\n- No data available for : ",SPs[s]," - ",MS," - ",GSAs[g], "\n"))
        }
      }
}
} # check_Land
```

```{r RCG_check_loc, results = 'asis', fig.height=6, fig.width=9}
if (check_CS) {

  cat("\n## Consistency of trip location\n")
```
```

```
  cat("\nThe consistency of trip location was checked plotting the spatial distribution of data using the initial
and final coordinates, where available, and the ports position included in the data in case coordinates are not
available.\n")

for (g in 1:length(GSAs)) {
      s=1
      for (s in 1:length(SPs)) {
        suppressMessages(res <- RCG_check_loc(data=CS, MS = MS, GSA = GSAs[g]))
        if (!is.null(res)) {

        cat("\n\n\n")
        cat(paste0("\nMap of data spatial distribution.\n"))
print(res)
        cat("\n\n\n")

} else {
          cat(paste0("\n\n\n- No data available for ",SPs[s]," - ",MS," - ",GSAs[g],"\n"))
        }
      }
}

} # check_CS
```

```{r RCG_summarize_ind_meas, results = 'asis', fig.height=6, fig.width=9}
if (check_CS) {
    cat("\n## Summary of individual measurements\n")
    cat("\nBiological data were analysed to provide a summary table reporting the number of individual by trip for
which biological data have been collected (length, sex, maturity, weight and age)\n")

g=1
for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
      s=1
      for (s in 1:length(SPs)) {
        cat(paste0("\n#### ", SPs[s], "\n"))
        res  <-  suppressMessages(RCG_summarize_ind_meas(data  =  CS,  MS  =  MS,  GSA  =  GSAs[g],  SP  =
SPs[s],verbose=FALSE))
        if (!is.null(res)){
        if (nrow(res)>0){
            cat(paste0("\n_Summary table reporting the number of measurements by trip for each biological variable
for ",SPs[s]," in ",GSAs[g],"_"))
        print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width=TRUE) )
        } else {
            cat(paste0("\n- No data available for : ",SPs[s]," - ",MS," - ",GSAs[g], "\n"))
        }
        } else {
            cat(paste0("\n- No data available for : ",SPs[s]," - ",MS," - ",GSAs[g], "\n"))
        }
      }
}
} # check_Land
```

```{r RCG_check_LFD, results = 'asis'}
if (check_CS) {

  cat("\n## Consistency of LFD by year\n")
  cat("\nThe consistency of length frequency distributions (LFDs) was checked generating a multi-frame plot of the
LFD by year. The Grubbs' test was performed to evaluate whether the minimum and the maximum values of Length class
distribution could be considered as outliers. In case, a table reporting the outliers is returned.\n")

for (g in 1:length(GSAs)) {
      cat(paste0("\n### ", GSAs[g], "\n"))
      s=1
      for (s in 1:length(SPs)) {
        cat(paste0("\n#### ", SPs[s], "\n"))
        suppressMessages(res <- RCG_check_LFD(data=CS, MS = MS, GSA = GSAs[g],SP = SPs[s], min_len = NA, max_len =
NA, verbose = FALSE))
        if (!is.null(res)) {

        cat("\n\n\n")
        cat(paste0("\nPlot of LFDs by year for ",SPs[s]," - ",MS," - ",GSAs[g],"\n"))
print(res[[2]])
        cat("\n\n\n")

if (nrow(res[[1]])>0) {
        cat(paste0("\n_Outliers of the length distribution for ",SPs[s]," in ",GSAs[g],"_\n"))
        print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width=TRUE)
              )
                cat("\n\n\n")
     } else {
         cat("\n", paste0("- No outliers detected in length data"), "\n")
     }
        } else {
          cat(paste0("\n\n\n- No length data available for ",SPs[s]," - ",MS," - ",GSAs[g],"\n"))
        }
```

```
      }
}

} # check_CS
```

```{r RCG_check_LFD_comm_cat, results = 'asis'}
if (check_CS) {

  cat("\n## Consistency of LFD by year and commercial category\n")
  cat("\nThe consistency of length frequency distributions (LFD) was checked generating a multi-frame plot of the LFD by year and commercial size categories.\n")

for (g in 1:length(GSAs)) {
      cat(paste0("\n### ", GSAs[g], "\n"))
      s=1
      for (s in 1:length(SPs)) {
        cat(paste0("\n#### ", SPs[s], "\n"))
        suppressMessages(res <- RCG_check_LFD_comm_cat(data=CS, MS = MS, GSA = GSAs[g],SP = SPs[s]))
        if (!is.null(res)) {

        # cat(paste0("\n_Summary table of the length range by year and commercial size category for ",SPs[s]," in ",GSAs[g],"_\n"))
        #
        # print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width=TRUE) )

        cat("\n\n\n")
        cat(paste0("\nPlot of LFDs by year and commercial size category for ",SPs[s]," - ",MS," - ",GSAs[g],"\n"))
print(res[[2]])
        cat("\n\n\n")

} else {
          cat(paste0("\n\n\n- No data available for ",SPs[s]," - ",MS," - ",GSAs[g],"\n"))
        }
      }
}

} # check_CS
```

```{r RCG_check_mat, results = 'asis',fig.height=9,fig.width=9}
if (check_CS) {

  cat("\n## Consistency of sex and maturity stage\n")
  cat("\nThe consistency of maturity stages by sex and year was qualitatively checked accordingly to the length class.\n")

for (g in 1:length(GSAs)) {
      cat(paste0("\n### ", GSAs[g], "\n"))
      s=1
      for (s in 1:length(SPs)) {
        cat(paste0("\n#### ", SPs[s], "\n"))
        suppressMessages(res <- RCG_check_mat(data=CS, MS = MS, GSA = GSAs[g],SP = SPs[s], verbose = FALSE))
        if (!is.null(res)) {

        cat("\n\n\n")
        cat(paste0("\nPlot of length-maturity by year and sex for ",SPs[s]," - ",MS," - ",GSAs[g],"\n"))
print(res)
        cat("\n\n\n")

} else {
          cat(paste0("\n\n\n- No data available for ",SPs[s]," - ",MS," - ",GSAs[g],"\n"))
        }
      }
}

} # check_CS
```

```{r RCG_check_mat_ogive, results = 'asis'}
if (check_CS) {
  cat("\n## Check of maturity ogive\n")
  cat(paste0("\nThe maturity stages composition was checked estimating the maturity ogives by sex for: ", paste(SPs, collapse = ", "), ". '0', '1' and '2a' were considered as immature stages.\n", sep = ""))

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      # maturity ogive for males individuals
      suppressMessages(res_M <- RCG_check_mat_ogive(data = CS, MS = MS, GSA = GSAs[g], SP = SPs[s], sex = "M", immature_stages = stages_for_immatures, verbose = FALSE))
      # maturity ogive for females individuals
      suppressMessages(res_F <- RCG_check_mat_ogive(data = CS, MS = MS, GSA = GSAs[g], SP = SPs[s], sex = "F", immature_stages = stages_for_immatures, verbose = FALSE))
```

```
      # MALES
      if (!is.null(res_M)) {
        cat("\n\n\n")
        cat(paste0("\nPlot of maturity ogive of males individuals for: ", SPs[s], " - ", MS, " - ", GSAs[g],
"\n"))
print(res_M)
        cat("\n\n\n")
} else {
        cat(paste0("\n\n\n- No males' data available for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
      }

      # FEMALES
      if (!is.null(res_F)) {
        cat("\n\n\n")
        cat(paste0("\nPlot of maturity ogive of males individuals for: ", SPs[s], " - ", MS, " - ", GSAs[g],
"\n"))
        print(res_F)
        cat("\n\n\n")
      } else {
        cat(paste0("\n\n\n- No females' data available for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
      }
    }
  }
} # check_CS
```

```{r RCG_check_lw, results = 'asis'}
if (check_CS) {

  cat("\n## Consistency of length-weight relationship\n")
  cat("\nThe consistency of length-weight relationship was checked generating a multi-frame plot of the LFD by
year and sex.\n")


g=1
for (g in 1:length(GSAs)) {
      cat(paste0("\n### ", GSAs[g], "\n"))
      s=1
      for (s in 1:length(SPs)) {
        cat(paste0("\n#### ", SPs[s], "\n"))
        suppressMessages(res <- RCG_check_lw(data=CS, MS = MS, GSA = GSAs[g],SP = SPs[s], Min = NA, Max = NA,
verbose = FALSE))
        if (!is.null(res)) {

        cat("\n\n\n")
        cat(paste0("\nPlot of length-weight by year and sex for ",SPs[s]," - ",MS," - ",GSAs[g],"\n"))
print(res[[2]])
        cat("\n\n\n")

if (nrow(res[[1]])>0) {
        cat(paste0("\n_Outliers of the lengths distribution for ",SPs[s]," in ",GSAs[g],"_\n"))
        print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width=TRUE) )
                  cat("\n\n\n")
     } else {
         cat("\n", paste0("- No outliers detected in length data"), "\n")
     }

        } else {
          cat(paste0("\n\n\n- No data available for ",SPs[s]," - ",MS," - ",GSAs[g],"\n"))
        }
      }
}

} # check_CS
```

```{r RCG_CS_AL, results = 'asis'}
if (check_CS) {

  cat("\n## Consistency of age-length relationship\n")
  cat("\nRCG detailed data were checked for the consistency of the age-length data. The Grubbs' test was performed
to evaluate whether the maximum value of age distribution could be considered as outliers. In case, a table
reporting the outliers is returned.\n")

for (g in 1:length(GSAs)) {
      cat(paste0("\n### ", GSAs[g], "\n"))
      s=1
      for (s in 1:length(SPs)) {
        cat(paste0("\n#### ", SPs[s], "\n"))
        suppressMessages(res <- RCG_check_AL(data=CS, MS = MS, GSA = GSAs[g],SP = SPs[s], min_age = NA, max_age =
NA))
        if (!is.null(res)) {
        cat("\n\n\n")
        cat(paste0("\nPlot of age-length relationship by year and sex for ",SPs[s]," - ",MS," - ",GSAs[g],"\n"))
print(res[[3]])
        cat("\n\n\n")
```

```
if (nrow(res[[2]])>0) {
  cat("\n\n\n")
  cat(paste0("\n_Table of the age class outliers for ",SPs[s]," in ",GSAs[g],"_\n"))
   print(res[[2]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width=TRUE) )

        cat("\n\n\n")
} else {
    cat("\n", paste0("- No outliers detected records in age data"), "\n")
}
        } else {
          cat(paste0("\n\n\n- No age data available for ",SPs[s]," - ",MS," - ",GSAs[g],"\n"))
        }
      }
}

} # check_CS
```

6. APPENDIX II – MED & BS Rmd script

````
---
title: "| ![](logo.png){width=6in} \\vspace{1in} \n\n\n\n`r format(params$term)` Report\n\n\n\n
  \\vspace{0.1in} "

date: "Compiled on `r format(Sys.time(), '%d/%m/%Y, %H:%M')`"
subtitle: ''

output:
  html_document:
    template: RDBFIS_report_template.html
    toc: true
    toc-title: "Table of contents"
    toc_depth: 5
    number_sections: true
    collapsed: true
    df_print: paged
params:
  term: MEDBS datacall
---
<!-- INITIALIZATION -->

```{r inizialization, include=FALSE}
knitr::opts_chunk$set(
  echo = FALSE,
  message = FALSE,
  warning = FALSE
)
```


```{r libraries, include=FALSE}
# loading needed libraries
lib  <-  c("RDBqc",  "knitr",  "kableExtra",  "dplyr",  "ggplot2",  "rworldmap",  "sp",  "rworldxtra",  "pander",
"data.table", "grDevices", "magrittr", "tictoc", "tidyverse", "fishmethods", "tidyr", "gridExtra")
lapply(lib, require, character.only = TRUE)
```

<!-- USER DEFINED SOURCE FILES -->

```{r setup, include=FALSE}

# reading files
# UNCOMMENT  THE FOLLOWING LINES FOR STAND-ALONE USE
# SELECT THE APPROPRIATE FILE PATHS ON THE LOCAL FOLDERS

Catch <- read.table("./catch.csv", sep = ";", header = TRUE)
Land <- read.table("./landings_gsa18.csv", sep = ";", header = TRUE)
Disc <- read.table("./discards_gsa18.csv", sep = ";", header = TRUE)
ML <- read.table("./ml.csv", sep = ";", header = TRUE)
MA <- read.table("./ma.csv", sep = ";", header = TRUE)
SL <- read.table("./srl.csv", sep = ";", header = TRUE)
SA <- read.table("./sra.csv", sep = ";", header = TRUE)
GP <- read.table("./gp.csv", sep = ";", header = TRUE)
ALK <- read.table("./alk.csv", sep = ";", header = TRUE)

SPs <- c("HKE", "MUT", "DPS")
MS <- "ITA"
GSAs <- "GSA 18"
```


```{r filters, include=FALSE}
# check the presence of user defined filter for species (SPs)
if (exists("SPs")) {
  if (length(SPs) == 1 & is.na(SPs[1])) {
    user_SP <- FALSE
  } else {
    user_SP <- TRUE
  }
} else {
  user_SP <- FALSE
}

# check the presence of user defined filter for member state (MS)
if (exists("MS")) {
  if (length(MS) == 1 & is.na(MS[1])) {
    user_MS <- FALSE
  } else {
    user_MS <- TRUE
  }
} else {
  user_MS <- FALSE
}
````
````

```
# check the presence of user defined filter for sub-area (GSAs)
if (exists("GSAs")) {
  if (length(GSAs) == 1 & is.na(GSAs[1])) {
    user_GSA <- FALSE
  } else {
    user_GSA <- TRUE
  }
} else {
  user_GSA <- FALSE
}

# check the presence of user defined value for 'SOP_threshold'
if (exists("SOP_threshold")) {
  if (length(SOP_threshold) == 1 & is.na(SOP_threshold[1])) {
    SOP_threshold <- 5
  }
} else {
  SOP_threshold <- 5
}

# check the presence of user defined value for 'Rt'
if (exists("Rt")) {
  if (length(Rt) == 1 & is.na(Rt[1])) {
    Rt <- 1
  }
} else {
  Rt <- 1
}
```

```
\newpage
<!-- CATCH -->

```{r check_catch_data, results = 'asis'}

if (exists("Catch")) {
  if (class(Catch) == "data.frame") {
    if (nrow(Catch) > 0) {
      check_Catch <- TRUE
      cat(paste0("\n# Catch table\n"))
      if (!user_SP) {
        SPs <- sort(unique(Catch$species))
      }
      if (!user_MS) {
        MS <- unique(Catch$country)
      }
      if (!user_GSA) {
        GSAs <- sort(unique(Catch$area))
      }
    } else {
      check_Catch <- FALSE
      cat("\n No catch data available \n")
    }
  }
} else {
  check_Catch <- FALSE
  cat("\n No catch data available \n")
}
```

```{r catch_duplicated_records, results = 'asis'}
if (check_Catch) {
  cat("\n## Presence of duplicated records\n")
  cat("\nData included in the catch table were checked for the presence of duplicated records. The result of the
check is here reported:\n")

  duplicates <- Catch[0, ]

  g <- 1
  for (g in 1:length(GSAs)) {
    s <- 1
    for (s in 1:length(SPs)) {
      res <- MEDBS_check_duplicates(data = Catch, type = "c", SP = SPs[s], MS = "ITA", GSA = GSAs[g], verbose =
FALSE)
      if (!is.null(res)) {
        duplicates <- rbind(duplicates, res)
      }
    }
  }

  if (nrow(duplicates) > 0) {
    cat("\n", paste0("- _WARNING_ - ", nrow(duplicates), " replicated record/records in the data"), "\n")
    duplic <- data.frame(id = duplicates[, 1])
    print(duplic %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
    Catch <- Catch[-which(rownames(Catch) %in% as.numeric(rownames(duplicates))), ]
  } else {
```
```

```
    cat("\n", paste0("- No replicated records in Catch data"), "\n")
  }

  cat("\n*Attention*: eventual duplicated records will be removed from catch data for the following analysis \n")
} # check_Catch
```

```{r catch_coverage, results = 'asis',fig.height=6,fig.width=9}

if (check_Catch) {
  cat("\n## Coverage of catch data\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_Catch_coverage(Catch, SP = SPs[s], MS = MS, GSA = GSAs[g]))
      tab1 <- res[[1]]
      tab2 <- res[[2]]

      if (!is.null(res)) {
        suppressMessages(tab1 <- data.frame(tab1 %>% group_by(country, year, area, species) %>% summarise(landings
= sum(landings, na.rm = TRUE))))

        if (nrow(tab1) > 0) {
          cat(paste0("\n_Coverage of landing data for ", SPs[s], " in ", GSAs[g], "_\n"))

          print(tab1 %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))

cat("\n\n\n")
          print(res[[3]])
cat("\n\n\n")
        } else {
          cat("\n\n\n- No data for landings")
        }

        suppressMessages(tab2 <- data.frame(tab2 %>% group_by(country, year, area, species) %>% summarise(discards
= sum(discards, na.rm = TRUE))))
        if (nrow(tab2) > 0) {
          cat(paste0("\n_Coverage of discard data for ", SPs[s], " in ", GSAs[g], "_"))
          print(tab2 %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
cat("\n\n\n")

          print(res[[4]])
cat("\n\n\n")
        } else {
          cat("\n\n\n- No data for discards")
        }
      } else {
        cat(paste0("\n- No Catch data for the selected combination: ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
      }
    }
  }
} # check_Catch
```

```{r catch_SOP, results = 'asis',fig.height=6,fig.width=9}
if (check_Catch) {
  cat("\n## Check of Sum of Products (SOP)\n")
  cat(paste0("\nThe result of the percentage difference of SOP and both landing and discard volumes are reported
in the following table in case the value is greater then ", SOP_threshold, "% (selected threshold value):\n"))

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_SOP(Catch, SP = SPs[s], MS = MS, GSA = GSAs[g], threshold = SOP_threshold,
verbose = FALSE))

      if (nrow(res) > 0) {
        cat(paste0("\n_Critical values of SOP for ", SPs[s], " in ", GSAs[g], "_\n\n"))
        print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      } else {
        cat(paste0("\n_No critical values of SOP detected for ", SPs[s], " in ", GSAs[g], "_\n\n"))
      }
    }
  }
} # check_Catch
```

<!-- LANDING -->

```{r check_landing_data, results = 'asis'}
```

```
if (exists("Land")) {
  if (class(Land) == "data.frame") {
    if (nrow(Land) > 0) {
      check_Land <- TRUE
      cat(paste0("\n# Landing table\n"))
      if (!user_SP) {
        SPs <- sort(unique(Land$species))
      }
      if (!user_MS) {
        MS <- sort(unique(Land$country))
      }
      if (!user_GSA) {
        GSAs <- sort(unique(Land$area))
      }
    } else {
      check_Land <- FALSE
      cat("\n No Landing data available \n")
    }
  }
} else {
  check_Land <- FALSE
  cat("\n No Landing data available \n")
}
```

```{r landings_duplicated_records, results = 'asis'}
if (check_Land) {
  cat("\n## Presence of duplicated records \n")
  cat("\nData included in the landing table were checked for the presence of duplicated records. The result of the
check is here reported:\n")

  duplicates <- Land[0, ]

  for (g in 1:length(GSAs)) {
    s <- 1
    for (s in 1:length(SPs)) {
      # cc <- Catch[Catch$AREA== GSAs[g]&Catch$SPECIES==SPs[s],]
      # Catch <- rbind(Catch,cc[1,])
      res <- MEDBS_check_duplicates(data = Land, type = "l", SP = SPs[s], MS = "ITA", GSA = GSAs[g], verbose =
FALSE)
      if (!is.null(res)) {
        duplicates <- rbind(duplicates, res)
      }
    }
  }

  if (nrow(duplicates) > 0) {
    cat("\n", paste0("- _WARNING_ - ", nrow(duplicates), " replicated record/records in the data"), "\n")
    duplic <- data.frame(id = duplicates[, 1])
    print(duplic %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
    Land <- Land[-which(rownames(Land) %in% as.numeric(rownames(duplicates))), ]
  } else {
    cat("\n", paste0("- No replicated records in Landings data"), "\n")
  }
} # check_Land
```

```{r landing_coverage, results = 'asis',fig.height=6,fig.width=9}
if (check_Land) {
  cat("\n## Coverage of landings data\n")

  g <- 1
  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 3
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_Landing_coverage(Land, SP = SPs[s], MS = MS, GSA = GSAs[g]))
      tab1 <- res[[1]]

      if (!is.null(res)) {
        tab1 <- aggregate(tab1$landings, by = list(tab1$country, tab1$year, tab1$area, tab1$gear, tab1$species),
FUN = "sum")
        colnames(tab1) <- c("country", "year", "area", "gear", "species", "landings")
        if (nrow(tab1) > 0) {
          cat(paste0("\n_Coverage of landing data for ", SPs[s], " in ", GSAs[g], "_"))
          print(tab1 %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
          cat("\n\n\n")
        } else {
          cat("\n\n\n- No data for landings")
}
        print(res[[2]])
        cat("\n\n\n")
} else {
        cat(paste0("\n- No Landings data for the selected combination: ", SPs[s], " - ", MS, " - ", GSAs[g],
"\n"))
      }
```

```
    }
  }
} # check_Land
```

```{r comp_Q_VL_fishery, results = 'asis', fig.height=6, fig.width=9}
if (check_Land) {
  cat("\n## Checks on landing volumes\n")
  cat("\n### Comparison of landing volumes aggregated by quarter and fishery accounting for vessel length\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n#### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n##### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_comp_land_Q_VL_fishery(data = Land, MS = MS, GSA = GSAs[g], SP = SPs[s]))
res <- res[!is.na(res$ratio) & res$ratio > 0, ]
if (nrow(res) > 0) {
        cat(paste0("\n_Comparison of landing volumes aggregated by quarter and fishery reported with and without
vessel length codes for ", SPs[s], " in ", GSAs[g], "_"))
        print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      } else {
        cat(paste0("\n- No landing volumes aggregated by quarter and fishery are reported with and without vessel
codes at the same time for the selected combination: ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
      }
    }
  }
} # check_Land
```

```{r comp_YQ_fishery, results = 'asis', fig.height=6, fig.width=9}
if (check_Land) {
  cat("\n### Comparison of landing volumes aggregated by fishery accounting for time frame\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_comp_land_YQ_fishery(data = Land, MS = MS, GSA = GSAs[g], SP = SPs[s]))
res <- res[!is.na(res$ratio) & res$ratio > 0, ]
if (nrow(res) > 0) {
        cat(paste0("\n_Comparison of landing volumes aggregated by fishery according to the time frame
(quarter/year) for ", SPs[s], " in ", GSAs[g], "_"))
        print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      } else {
        cat(paste0("\n- No landing volumes aggregated by fishery are reported for both the time frames
(quarter/year) for the selected combination: ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
      }
    }
  }
} # check_Land
```

```{r KS_test_land, results = 'asis', fig.height=6, fig.width=9}

if (check_Land) {
  cat("\n## Kolmogorov-Smirnov test\n")
  cat("\nThe Kolmogorov-Smirnov test provides cumulative length distribution plots by fishery and year. The test
is performed on couples of years to assess if their distributions belong to the same population. The table reports
only the cases in which the distribution does not belong to the same population in the given comparison.\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_ks(data = Land, type = "l", SP = SPs[s], MS = MS, GSA = GSAs[g], Rt = Rt))

      if (class(res) == "list" & length(res) > 0) {
        cat("\n\n\n")
        cat(paste0("\nPlot of cumulative length distributions by fishery and year for ", SPs[s], " - ", MS, " - ",
GSAs[g], "\n"))
print(res[[4]])
        cat("\n\n\n")


cat(paste0("\n_Summary table of Kolmogorov-Smirnov test for ", SPs[s], " in ", GSAs[g], "_\n"))
        tab1 <- res[[2]]
        tab1 <- tab1[tab1$Comment == "not belong to same population", ]
        print(tab1 %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      } else {
        cat(paste0("\n- No Kolmogorov-Smirnov test results are available for the selected combination: ", SPs[s],
" - ", MS, " - ", GSAs[g], ".\n"))
      }
    }
  }
```

```
} # check_Land
```

```{r lengthclass_0_land, results = 'asis', fig.height=6, fig.width=9}
if (check_Land) {
  cat("\n## Check the presence of length classes numbers with zeros in landings\n")
  cat("\nThe presence of length classes numbers zero with having weigth > 0 in landings are reported in the following table:\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_lengthclass_0(data = Land, type = "l", MS = MS, GSA = GSAs[g], SP = SPs[s]))
      if (nrow(res) > 0) {
        cat(paste0("\n_Landing records with 0  length class values having weigth > 0 for ", SPs[s], " in ", GSAs[g], "_"))
        print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      } else {
        cat(paste0("\n- No landing records with 0 length class values having weigth > 0 for : ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
      }
    }
  }
} # check_Land
```

```{r length_ind_land, results = 'asis', fig.height=6, fig.width=9}
if (check_Land) {
  cat("\n## Main length size indicators\n")
  cat(paste0("\nThe time series of the main length size indicators (mean and median length) by fishery was estimated and hereafter reported: \n"))

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_length_ind(data = Land, type = "l", MS = MS, GSA = GSAs[g], SP = SPs[s], splines = c(0.2, 0.4, 0.6, 0.8), Xtresholds = c(0.25, 0.5, 0.75)))

      if (class(res) == "list" & length(res) > 0) {
        cat(paste0("\n_Summary table of Main length size indicators for ", SPs[s], " in ", GSAs[g], "_"))
        print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))

        name <- names(res)
        list_name <- strsplit(name, " _ ")
        short_list <- NULL
        n <- 1
        for (n in 1:length(list_name)) {
          short_list[n] <- list_name[[n]][1]
        }
        index <- which(short_list %in% c("MeanLength", "MedianLength"))

        x <- 3
        for (x in index) {
          text <- strsplit(name[[x]], " _ ")[[1]][1]
          cat(paste0("\nPlot of ", short_list[x], " for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
suppressMessages(print(res[[x]]))
          cat("\n\n\n")
}
      } else {
        cat(paste0("\n- No main length size indicators estimated for the selected combination: ", SPs[s], " - ", MS, " - ", GSAs[g], ".\n"))
      }
    }
  }
} # check_Land
```

```{r yr_missing_length_land, results = 'asis', fig.height=6, fig.width=9}
if (check_Land) {
  cat("\n## Check the presence of years with missing length distribution\n")
  cat("\nLanding data were checked to identify the presence of years with missing length distributions. The years with the missing length distribution are reported in the following table (if any):\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_yr_missing_length(data = Land, type = "l", MS = MS, GSA = GSAs[g], SP = SPs[s]))
      if (!is.null(res)) {
        if (nrow(res) > 0) {
```

```
          cat(paste0("\n_Years in which the length distribution is missing in landing data for ", SPs[s], " in ",
GSAs[g], "_"))
          print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
        } else {
          cat(paste0("\n- No years in landing with missing length distribution for : ", SPs[s], " - ", MS, " - ",
GSAs[g], "\n"))
        }
      } else {
        cat("No data available")
      }
    }
  }
} # check_Land
```

```{r weight_0_land, results = 'asis', fig.height=6, fig.width=9}
if (check_Land) {
  cat("\n## Check the presence of landing weights with zeros in landings\n")
  cat("\nThe presence of rows in landing with 0 values in weights having length classes filled in is reported in
the following table:\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- MEDBS_weight_0(data = Land, type = "l", MS = MS, GSA = GSAs[g], SP = SPs[s], verbose = FALSE)
      if (nrow(res) > 0) {
        cat(paste0("\n_Landing records in weight equal to 0 having length classes filled in for ", SPs[s], " in ",
GSAs[g], "_"))
        print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      } else {
        cat(paste0("\n- No landing records in weight equal to 0 having length classes filled in for : ", SPs[s], "
- ", MS, " - ", GSAs[g], "\n"))
      }
    }
  }
} # check_Land
```

```{r weight_minus1_land, results = 'asis', fig.height=6, fig.width=9}
if (check_Land) {
  cat("\n## Check the presence of landing weights with -1 in landings\n")
  cat("\nThe presence of rows in landing with -1 values in weights having length classes filled in is reported in
the following table:\n")

  g <- 2
  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- MEDBS_weight_minus1(data = Land, type = "l", MS = MS, GSA = GSAs[g], SP = SPs[s], verbose = FALSE)
      if (nrow(res) > 0) {
        cat(paste0("\n_Landing records in weight equal to -1 having length classes filled in for ", SPs[s], " in
", GSAs[g], "_"))
        print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      } else {
        cat(paste0("\n- No landing records in weight equal to -1 having length classes filled in for : ", SPs[s],
" - ", MS, " - ", GSAs[g], "\n"))
      }
    }
  }
} # check_Land
```

```{r land_mean_weight_land, results = 'asis', fig.height=6, fig.width=9}
if (check_Land) {
  cat("\n## Mean weight by year, gear and fishery aggregation\n")
  cat(paste0("\n Landing data were checked to assess the consistency of mean species landing weight by year, gear
and fishery.\n"))

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_land_mean_weight(data = Land, MS = MS, GSA = GSAs[g], SP = SPs[s]))

      if (class(res) == "list" & length(res) > 0) {
        if (!is.null(res[[1]])) {
          cat(paste0("\n_Summary table of Main landing weights for ", SPs[s], " in ", GSAs[g], "_"))
          print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
        } else {
          cat(paste0("\n- No main weights estimated for the selected combination: ", SPs[s], " - ", MS, " - ",
GSAs[g], ".\n"))
        }
```

```
        if (length(res) > 1) {
          cat(paste0("\nPlot of landing mean weights for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
suppressMessages(print(res[[2]]))
          cat("\n\n\n")
} else {
          cat(paste0("\n- No main weights estimated for the selected combination: ", SPs[s], " - ", MS, " - ",
GSAs[g], ".\n"))
        }
      } else {
        cat(paste0("\n- No main weights estimated for the selected combination: ", SPs[s], " - ", MS, " - ",
GSAs[g], ".\n"))
      }
    }
  }
} # check_Land
```

```{r plot_landing_ts, results = 'asis', fig.height=6, fig.width=9}
if (check_Land) {
  cat("\n## Total landing time series by quarter\n")
  cat(paste0("\nTo identify possible inconsistencies in landing data, the species time series of landing volumes
is here plotted by quarter.\n"))

  g <- 1
  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_plot_landing_ts(data = Land, MS = MS, GSA = GSAs[g], SP = SPs[s], by =
"quarter"))

      if (length(res) > 0) {
        cat(paste0("\nPlot of total landings for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
print(res)
        cat("\n\n\n")
} else {
        cat(paste0("\n- No total landing time series estimated for the selected combination: ", SPs[s], " - ", MS,
" - ", GSAs[g], ".\n"))
      }
    }
  }
} # check_Land
```

```{r plot_land_vol, results = 'asis', fig.height=6, fig.width=9}
if (check_Land) {
  cat("\n## Total landing time series by gear and fishery\n")
  cat(paste0("\nTo identify possible inconsistencies in landing data, the species time series of landing volumes
is here plotted by fishery and gear.\n"))

  g <- 1
  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_plot_land_vol(data = Land, MS = MS, GSA = GSAs[g], SP = SPs[s]))

      if (length(res) > 0) {
        cat(paste0("\nPlot of total landings for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
suppressMessages(print(res))
        cat("\n\n\n")
} else {
        cat(paste0("\n- No total landing time series estimated for the selected combination: ", SPs[s], " - ", MS,
" - ", GSAs[g], ".\n"))
      }
    }
  }
} # check_Land
```

<!-- DISCARD -->

```{r check_discard_data, results = 'asis'}

if (exists("Disc")) {
  if (class(Disc) == "data.frame") {
    if (nrow(Disc) > 0) {
      check_Disc <- TRUE
      cat(paste0("\n# Discard table\n"))
      if (!user_SP) {
        SPs <- unique(Disc$species)
      }
      if (!user_MS) {
        MS <- unique(Disc$country)
      }
```

```
      if (!user_GSA) {
        GSAs <- unique(Disc$area)
      }
    }
  }
} else {
  check_Disc <- FALSE
  cat("\n No Discard data available \n")
}
```

```{r discards_duplicated_records, results = 'asis'}
if (check_Disc) {
  cat("\n## Presence of duplicated records \n")
  cat("\nData included in the discard table were checked for the presence of duplicated records. The result of the check is here reported:\n")

  duplicates <- Disc[0, ]

  for (g in 1:length(GSAs)) {
    s <- 1
    for (s in 1:length(SPs)) {
      res <- MEDBS_check_duplicates(data = Disc, type = "d", SP = SPs[s], MS = "ITA", GSA = GSAs[g], verbose = FALSE)
      if (!is.null(res)) {
        duplicates <- rbind(duplicates, res)
      }
    }
  }

  if (nrow(duplicates) > 0) {
    cat("\n", paste0("- _WARNING_ - ", nrow(duplicates), " replicated record/records in the data"), "\n")
    duplic <- data.frame(id = duplicates[, 1])
    print(duplic %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
    Disc <- Disc[-which(rownames(Disc) %in% as.numeric(rownames(duplicates))), ]
  } else {
    cat("\n", paste0("- No replicated records in Discard data"), "\n")
  }
} # check_Disc
```

```{r discard_coverage, results = 'asis',fig.height=6,fig.width=9}
if (check_Disc) {
  cat("\n## Coverage of discards data\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_discard_coverage(Disc, SP = SPs[s], MS = MS, GSA = GSAs[g]))
      tab1 <- res[[1]]

      if (!is.null(res)) {
        tab1 <- aggregate(tab1$discards, by = list(tab1$country, tab1$year, tab1$area, tab1$gear, tab1$species), FUN = "sum")
        colnames(tab1) <- c("COUNTRY", "YEAR", "AREA", "GEAR", "SPECIES", "LANDINGS")
        if (nrow(tab1) > 0) {
          cat(paste0("\n_Coverage of landing data for ", SPs[s], " in ", GSAs[g], "_"))
          print(tab1 %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
          cat("\n\n\n")
        } else {
          cat("\n\n\n- No data for landings")
}
        print(res[[2]])
        cat("\n\n\n")
} else {
        cat(paste0("\n- No Landings data for the selected combination: ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
      }
    }
  }
} # check_Land
```

```{r comp_disc_YQ, results = 'asis', fig.height=6, fig.width=9}
if (check_Disc) {
  cat("\n## Comparison between discards in weight by quarter and year\n")
  cat("\nDiscard data by gear were aggragated by quarter and years (-1) and were compared to identify possible inconsistencies.\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_comp_disc_YQ(data = Disc, MS = MS, GSA = GSAs[g], SP = SPs[s]))
```

```
      if (nrow(res) > 0) {
        cat(paste0("\n_Summary table of discard data aggregated by gear and both quarter and year for ", SPs[s], "
in ", GSAs[g], "_"))
        print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      } else {
        cat(paste0("\n No discard data available for: ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
      }
    }
  }
} # check_Disc
```

```{r comp_disc_YQ_fishery, results = 'asis', fig.height=6, fig.width=9}
if (check_Disc) {
  cat("\n## Comparison between discards in weight by quarter and year, accounting for fishery\n")
  cat("\nDiscard data by gear and fishery were aggragated by quarter and yeas and were compared to identify
possible inconsistencies.\n")

  g <- 1
  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_comp_disc_YQ_fishery(data = Disc, MS = MS, GSA = GSAs[g], SP = SPs[s]))
      if (nrow(res) > 0) {
        cat(paste0("\n_Summary table of discard data aggregated by gear, fishery and both quarter and year for ",
SPs[s], " in ", GSAs[g], "_"))
        print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      } else {
        cat(paste0("\n No discard data available for: ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
      }
    }
  }
} # check_Disc
```

```{r disc_mean_weight, results = 'asis', fig.height=6, fig.width=9}
if (check_Disc) {
  cat("\n## Mean weight by year, gear and fishery aggregation\n")
  cat(paste0("\n Discard data were checked to assess the consistency of mean species discard weight by year, gear
and fishery.\n"))

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_disc_mean_weight(data = Disc, MS = MS, GSA = GSAs[g], SP = SPs[s]))
      if (class(res) == "list" & length(res) > 0) {
        if (!is.null(res[[1]])) {
          cat(paste0("\n_Summary table of Main discard weights for ", SPs[s], " in ", GSAs[g], "_"))
          print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
        }
        if (length(res) > 1) {
          cat(paste0("\nPlot of discard mean weights for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
suppressMessages(print(res[[2]]))
          cat("\n\n\n")
} else {
          cat(paste0("\n- No main weights estimated for the selected combination: ", SPs[s], " - ", MS, " - ",
GSAs[g], ".\n"))
        }
      } else {
        cat(paste0("\n- No main weights estimated for the selected combination: ", SPs[s], " - ", MS, " - ",
GSAs[g], ".\n"))
      }
    }
  }
} # check_Disc
```

```{r KS_test_disc, results = 'asis', fig.height=6, fig.width=9}

if (check_Disc) {
  cat("\n## Kolmogorov-Smirnov test\n")
  cat("\nThe Kolmogorov-Smirnov test provides cumulative length distribution plots by fishery and year. The test
is performed on couples of years to assess if their distributions belong to the same population. The summary table
reports only the cases in which the distribution does not belong to the same population in the given
comparison.\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 2
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_ks(data = Disc, type = "d", SP = SPs[s], MS = MS, GSA = GSAs[g], Rt = Rt))
```

```
      if (class(res) == "list" & length(res) > 0) {
        if (!is.null(res[[4]])) {
          cat("\n\n\n")
          cat(paste0("\nPlot of cumulative length distributions by fishery and year for ", SPs[s], " - ", MS, " -
", GSAs[g], "\n"))
          plot(res[[4]])
          cat("\n\n\n")
          if (!is.null(res[[2]])) {
            cat(paste0("\n_Summary table of Kolmogorov-Smirnov test for ", SPs[s], " in ", GSAs[g], "_\n"))
            tab1 <- res[[2]]
            tab1 <- tab1[tab1$Comment == "not belong to same population", ]
            print(tab1 %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
          }
        } else {
          cat(paste0("\n- No Kolmogorov-Smirnov test results are available for the selected combination: ",
SPs[s], " - ", MS, " - ", GSAs[g], ".\n"))
        }
      } else {
        cat(paste0("\n- No Kolmogorov-Smirnov test results are available for the selected combination: ", SPs[s],
" - ", MS, " - ", GSAs[g], ".\n"))
      }
    }
  }
} # check_Disc
```

```{r lengthclass_0_disc, results = 'asis', fig.height=6, fig.width=9}
if (check_Disc) {
  cat("\n## Check the presence of length classes numbers with zeros in discards\n")
  cat("\nThe presence of length classes numbers zero with having weigth > 0 in discards are reported in the
following table:\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_lengthclass_0(data = Disc, type = "d", MS = MS, GSA = GSAs[g], SP = SPs[s]))
      if (nrow(res) > 0) {
        cat(paste0("\nDiscard records with 0  length class values having weigth > 0 for ", SPs[s], " in ",
GSAs[g], "_"))
        print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      } else {
        cat(paste0("\n- No discard records with 0 length class values having weigth > 0 for : ", SPs[s], " - ",
MS, " - ", GSAs[g], "\n"))
      }
    }
  }
} # check_Disc
```

```{r length_ind_disc, results = 'asis', fig.height=6, fig.width=9}
if (check_Disc) {
  cat("\n## Main length size indicators\n")
  cat(paste0("\nThe time series of the main length size indicators (mean and median length) by fishery was
estimated and hereafter reported: \n"))

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_length_ind(data = Disc, type = "d", MS = MS, GSA = GSAs[g], SP = SPs[s],
splines = c(0.2, 0.4, 0.6, 0.8), Xtresholds = c(0.25, 0.5, 0.75)))

      if (class(res) == "list" & length(res) > 0) {
        cat(paste0("\n_Summary table of Main length size indicators for ", SPs[s], " in ", GSAs[g], "_"))
        print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))

        name <- names(res)
        list_name <- strsplit(name, " _ ")
        short_list <- NULL
        n <- 1
        for (n in 1:length(list_name)) {
          short_list[n] <- list_name[[n]][1]
        }
        index <- which(short_list %in% c("MeanLength", "MedianLength"))

        x <- 3
        for (x in index) {
          text <- strsplit(name[[x]], " _ ")[[1]][1]
          cat(paste0("\nPlot of ", short_list[x], " for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
          suppressMessages(plot(res[[x]]))
          cat("\n\n\n")
        }
      } else {
```

```
        cat(paste0("\n- No main length size indicators estimated for the selected combination: ", SPs[s], " - ",
MS, " - ", GSAs[g], ".\n"))
      }
    }
  }
} # check_Disc
```

```{r yr_missing_length_disc, results = 'asis', fig.height=6, fig.width=9}
if (check_Disc) {
  cat("\n## Check the presence of years with missing length distribution\n")
  cat("\nDiscard data were checked to identify the presence of years with missing length distributions. The years
with the missing length distribution are reported in the following table (if any):\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_yr_missing_length(data = Disc, type = "d", MS = MS, GSA = GSAs[g], SP =
SPs[s]))
      if (!is.null(res)) {
        if (nrow(res) > 0) {
          cat(paste0("\n_Years in which the length distribution is missing in discard data for ", SPs[s], " in ",
GSAs[g], "_"))
          print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
        } else {
          cat(paste0("\n- No years in discard with missing length distribution for : ", SPs[s], " - ", MS, " - ",
GSAs[g], "\n"))
        }
      } else {
        cat(paste0("\n- No years in discard with missing length distribution for : ", SPs[s], " - ", MS, " - ",
GSAs[g], "\n"))
      }
    }
  }
} # check_Disc
```

```{r weight_0_disc, results = 'asis', fig.height=6, fig.width=9}
if (check_Disc) {
  cat("\n## Check the presence of discard weights with zeros in discards\n")
  cat("\nThe presence of rows in discards with 0 values in weights having length classes filled in is reported in
the following table:\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- MEDBS_weight_0(data = Disc, type = "d", MS = MS, GSA = GSAs[g], SP = SPs[s], verbose = FALSE)
      if (nrow(res) > 0) {
        cat(paste0("\n_Discard records in weight equal to 0 having length classes filled in for ", SPs[s], " in ",
GSAs[g], "_"))
        print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      } else {
        cat(paste0("\n- No Discard records in weight equal to 0 having length classes filled in for : ", SPs[s], "
- ", MS, " - ", GSAs[g], "\n"))
      }
    }
  }
} # check_Disc
```

```{r weight_minus1_disc, results = 'asis', fig.height=6, fig.width=9}
if (check_Disc) {
  cat("\n## Check the presence of discard weights with -1 in discards\n")
  cat("\nThe presence of rows in discards with -1 values in weights having length classes filled in is reported in
the following table:\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- MEDBS_weight_minus1(data = Disc, type = "d", MS = MS, GSA = GSAs[g], SP = SPs[s], verbose = FALSE)
      if (nrow(res) > 0) {
        cat(paste0("\n_Discard records in weight equal to -1 having length classes filled in for ", SPs[s], " in
", GSAs[g], "_"))
        print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      } else {
        cat(paste0("\n- No discard records in weight equal to -1 having length classes filled in for : ", SPs[s],
" - ", MS, " - ", GSAs[g], "\n"))
      }
    }
  }
} # check_Disc
```

```
```

```{r plot_discard_ts, results = 'asis', fig.height=6, fig.width=9}
if (check_Disc) {
  cat("\n## Total discard time series by quarter\n")
  cat(paste0("\nTo identify possible inconsistencies in discard data, the species time series of discard volumes
is here plotted by quarter.\n"))

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_plot_discard_ts(data = Disc, MS = MS, GSA = GSAs[g], SP = SPs[s], by =
"quarter"))

      if (length(res) > 0) {
        cat(paste0("\nPlot of total discards for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
cat("\n\n")
        print(res)
cat("\n\n\n")
      } else {
        cat(paste0("\n- No total discards time series estimated for the selected combination: ", SPs[s], " - ",
MS, " - ", GSAs[g], ".\n"))
      }
    }
  }
} # check_Disc
```

```{r plot_disc_vol, results = 'asis', fig.height=6, fig.width=9}
if (check_Disc) {
  cat("\n## Total discard time series by gear and fishery\n")
  cat(paste0("\nTo identify possible inconsistencies in discard data, the species time series of discard volumes
is here plotted by fishery and gear.\n"))

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_plot_disc_vol(data = Disc, MS = MS, GSA = GSAs[g], SP = SPs[s]))

      if (length(res) > 0) {
        cat(paste0("\nPlot of total discards for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
cat("\n\n\n")
        suppressMessages(print(res))
cat("\n\n\n")
      } else {
        cat(paste0("\n- No total discard time series estimated for the selected combination: ", SPs[s], " - ", MS,
" - ", GSAs[g], ".\n"))
      }
    }
  }
} # check_Disc
```

<!-- GROWTH PARAMETERS -->

```{r check_GP_data, results = 'asis'}

if (exists("GP")) {
  if (class(GP) == "data.frame") {
    if (nrow(GP) > 0) {
      check_GP <- TRUE
      cat(paste0("\n# Growth parameters (GP) table\n"))
      if (!user_SP) {
        SPs <- unique(GP$species)
      }
      if (!user_MS) {
        MS <- unique(GP$country)
      }
      if (!user_GSA) {
        GSAs <- unique(GP$area)
      }
    }
  }
} else {
  check_GP <- FALSE
  cat("\n No growth parameters (GP) data available \n")
}
```

```{r GP_checks, results = 'asis', fig.height=6, fig.width=9}

if (check_GP) {
  cat("\n## plots of the growth parameter\n")
```

```
  cat("\nData related to growth parameter are used to produce a summary table (counting available VBGF parameters
by YEAR and SEX) and plots of growth curves for the selected combination of species and GSAs by sexes.\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_GP_check(GP, SP = SPs[s], MS = MS, GSA = GSAs[g]))

      if (class(res) == "list" & length(res) > 0) {
        cat(paste0("\n_Summary table of VBGF parameters for ", SPs[s], " in ", GSAs[g], "_"))
        print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))

        name <- names(res)
        list_name <- strsplit(name, " _ ")
        short_list <- NULL
        for (n in 1:length(list_name)) {
          short_list[n] <- list_name[[n]][1]
        }
        index <- which(short_list == "VBGF_cum")

        for (x in index) {
          text <- strsplit(name[[x]], " _ ")[[1]][5]
          cat(paste0("\nPlot of the VBGF parameter for ", SPs[s], " - ", MS, " - ", GSAs[g], " - ", text, "\n"))
print(res[[x]])
          cat("\n\n\n")
}
      } else {
        cat(paste0("\n- No VBGF parameters are available for the selected combination: ", SPs[s], " - ", MS, " -
", GSAs[g], ". (eventual -1 values in VBGF parameters were not considered)\n"))
      }
    }
  }
} # check_GP
```

```{r LW_checks, results = 'asis', fig.height=6, fig.width=9}
if (check_GP) {
  cat("\n## Plots of length-weight (LW) functions\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_LW_check(GP, SP = SPs[s], MS = MS, GSA = GSAs[g]))

      if (class(res) == "list" & length(res) > 0) {
        cat(paste0("\n_Summary table of length-weight parameters for ", SPs[s], " in ", GSAs[g], "_"))
        print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))

        name <- names(res)
        list_name <- strsplit(name, " _ ")
        short_list <- NULL
        for (n in 1:length(list_name)) {
          short_list[n] <- list_name[[n]][1]
        }

        short_list2 <- NULL
        for (n in 1:length(list_name)) {
          short_list2[n] <- list_name[[n]][5]
        }

        index <- which(short_list == "LW_cum" & !is.na(short_list2))

        for (x in index) {
          text <- strsplit(name[[x]], " _ ")[[1]][5]
          cat(paste0("\nPlot of the length-weight parameters for ", SPs[s], " - ", MS, " - ", GSAs[g], " - ",
text, "\n"))
print(res[[x]])
          cat("\n\n\n")
}
      } else {
        cat(paste0("\n- No length-weight parameters are available for the selected combination: ", SPs[s], " - ",
MS, " - ", GSAs[g], ". (eventual -1 values in LW parameters were not considered)\n"))
      }
    }
  }
} # check_GP
```

<!-- MA -->

```{r check_MA_data, results = 'asis'}

if (exists("MA")) {
```

```
  if (class(MA) == "data.frame") {
    if (nrow(MA) > 0) {
      check_MA <- TRUE
      cat(paste0("\n# Maturity at Age (MA) table\n"))
      if (!user_SP) {
        SPs <- unique(MA$species)
      }
      if (!user_MS) {
        MS <- unique(MA$country)
      }
      if (!user_GSA) {
        GSAs <- unique(MA$area)
      }
    }
  }
} else {
  check_MA <- FALSE
  cat("\n No Maturity at Age (MA) data available \n")
}
```

```{r MA_checks, results = 'asis', fig.height=6, fig.width=9}

if (check_MA) {
  cat("\n## Plots of Maturity at Age (MA) data\n")
  g <- 1
  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_MA_check(MA, SP = SPs[s], MS = MS, GSA = GSAs[g]))

      if (class(res) == "list" & length(res) > 0) {
        cat(paste0("\n_Summary table of percentage of matures at age for ", SPs[s], " in ", GSAs[g], "_"))
        print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))

        name <- names(res)
        list_name <- strsplit(name, " _ ")
        short_list <- NULL
        for (n in 1:length(list_name)) {
          short_list[n] <- list_name[[n]][1]
        }

        short_list2 <- NULL
        for (n in 1:length(list_name)) {
          short_list2[n] <- list_name[[n]][5]
        }

        index <- which(short_list == "MA_sex" & !is.na(short_list2))

        for (x in index) {
          text <- strsplit(name[[x]], " _ ")[[1]][5]
          cat(paste0("\nPlot of the percercentage of matures at age for ", SPs[s], " - ", MS, " - ", GSAs[g], " -
", text, "\n"))
print(res[[x]])
          cat("\n\n\n")
}
      } else {
        cat(paste0("\n- No MA data are available for the selected combination: ", SPs[s], " - ", MS, " - ",
GSAs[g], ".\n"))
      }
    }
  }
} # check_MA
```

<!-- ML -->

```{r check_ML_data, results = 'asis'}

if (exists("ML")) {
  if (class(ML) == "data.frame") {
    if (nrow(ML) > 0) {
      check_ML <- TRUE
      cat(paste0("\n# Maturity at Length (ML) table\n"))
      if (!user_SP) {
        SPs <- unique(ML$species)
      }
      if (!user_MS) {
        MS <- unique(ML$country)
      }
      if (!user_GSA) {
        GSAs <- unique(ML$area)
      }
    }
  }
```

```
} else {
  check_ML <- FALSE
  cat("\n No Maturity at Length (ML) data available \n")
}
```

```{r ML_checks, results = 'asis', fig.height=6, fig.width=9}

if (check_ML) {
  cat("\n## Plots of Maturity at Length (ML) data\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_ML_check(ML, SP = SPs[s], MS = MS, GSA = GSAs[g]))

      if (class(res) == "list" & length(res) > 0) {
        cat(paste0("\n_Summary table of percentage of matures at length for ", SPs[s], " in ", GSAs[g], "_"))
        print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))

        name <- names(res)
        list_name <- strsplit(name, " _ ")
        short_list <- NULL
        for (n in 1:length(list_name)) {
          short_list[n] <- list_name[[n]][1]
        }

        short_list2 <- NULL
        for (n in 1:length(list_name)) {
          short_list2[n] <- list_name[[n]][5]
        }

        index <- which(short_list == "ML" & !is.na(short_list2))

        x <- 3
        for (x in index) {
          text <- strsplit(name[[x]], " _ ")[[1]][5]
          cat(paste0("\nPlot of the percercentage of matures at length for ", SPs[s], " - ", MS, " - ", GSAs[g], "
- ", text, "\n"))
print(res[[x]])
          cat("\n\n\n")
}
      } else {
        cat(paste0("\n- No ML data are available for the selected combination: ", SPs[s], " - ", MS, " - ",
GSAs[g], ".\n"))
      }
    }
  }
} # check_ML
```

<!-- SRA -->

```{r check_SRA_data, results = 'asis'}

if (exists("SA")) {
  if (class(SA) == "data.frame") {
    if (nrow(SA) > 0) {
      check_SA <- TRUE
      cat(paste0("\n# Sex Ratio at age (SRA) table\n"))
      if (!user_SP) {
        SPs <- unique(SA$species)
      }
      if (!user_MS) {
        MS <- unique(SA$country)
      }
      if (!user_GSA) {
        GSAs <- unique(SA$area)
      }
    }
  }
} else {
  check_SA <- FALSE
  cat("\n No Sex Ratio at age (SRA) data available \n")
}
```

```{r SA_checks, results = 'asis', fig.height=6, fig.width=9}
if (check_SA) {
  cat("\n## Plots of Sex Ratio at age (SRA) data\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
```

```
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_SA_check(SA, SP = SPs[s], MS = MS, GSA = GSAs[g]))
      if (class(res) == "list" & length(res) > 0) {
        cat(paste0("\n_Summary table of sex ratio at age for ", SPs[s], " in ", GSAs[g], "_"))
        print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))

        cat(paste0("\nPlot of sex ratio at age for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
print(res[[2]])
        cat("\n\n\n")
} else {
        cat(paste0("\n- No SRA data are available for the selected combination: ", SPs[s], " - ", MS, " - ",
GSAs[g], ".\n"))
      }
    }
  }
} # check_SA
```

<!-- SL -->

```{r check_SL_data, results = 'asis'}

if (exists("SL")) {
  if (class(SL) == "data.frame") {
    if (nrow(SL) > 0) {
      check_SL <- TRUE
      cat(paste0("\n# Sex Ratio at length (SRL) table\n"))
      if (!user_SP) {
        SPs <- unique(SL$species)
      }
      if (!user_MS) {
        MS <- unique(SL$country)
      }
      if (!user_GSA) {
        GSAs <- unique(SL$area)
      }
    }
  }
} else {
  check_SL <- FALSE
  cat("\n No Sex Ratio at length (SRL) data available \n")
}
```

```{r SL_checks, results = 'asis', fig.height=6, fig.width=9}

if (check_SL) {
  cat("\n## Plots of Sex Ratio at length (SRL) data\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_SL_check(SL, SP = SPs[s], MS = MS, GSA = GSAs[g]))
      if (class(res) == "list" & length(res) > 0) {
        cat(paste0("\n_Summary table of sex ratio at length for ", SPs[s], " in ", GSAs[g], "_"))
        print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))

        cat(paste0("\nPlot of sex ratio at length for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
print(res[[2]])
        cat("\n\n\n")
} else {
        cat(paste0("\n- No SRL data are available for the selected combination: ", SPs[s], " - ", MS, " - ",
GSAs[g], ".\n"))
      }
    }
  }
} # check_SRL
```

<!-- ALK -->

```{r check_ALK_data, results = 'asis'}

if (exists("ALK")) {
  if (class(ALK) == "data.frame") {
    if (nrow(ALK) > 0) {
      check_ALK <- TRUE
      cat(paste0("\n# Age-Length Keys (ALK) table\n"))
      if (!user_SP) {
        SPs <- unique(ALK$species)
      }
      if (!user_MS) {
        MS <- unique(ALK$country)
      }
      if (!user_GSA) {
```

```
        GSAs <- unique(ALK$area)
      }
    }
  }
} else {
  check_ALK <- FALSE
  cat("\n No Age-Length Keys (ALK) data available \n")
}
```

```{r ALK_checks, results = 'asis', fig.height=6, fig.width=9}

if (check_ALK) {
  cat("\n## Plots of Age-Length Keys (ALK) data\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(MEDBS_ALK(ALK, SP = SPs[s], MS = MS, GSA = GSAs[g]))

      if (class(res) == "list" & length(res) > 0) {
        l <- 1
        for (l in 1:length(res)) {
          sex <- strsplit(names(res[l]), " - ")[[1]][1]
          cat(paste0("\nPlot of ALK for ", SPs[s], " - ", MS, " - ", GSAs[g], " - ", sex, "\n"))
print(res[[l]])
          cat("\n\n\n")
}
      } else {
        cat(paste0("\n- No ALK data are available for the selected combination: ", SPs[s], " - ", MS, " - ",
GSAs[g], ".\n"))
      }
    }
  }
} # check_SRL
```

7. APPENDIX III – FDI Rmd script

````
---
title: "| ![](logo.png){width=6in} \\vspace{1in} \n\n\n\n`r format(params$term)` Report\n\n\n\n
  \\vspace{0.1in} "

date: "Compiled on `r format(Sys.time(), '%d/%m/%Y, %H:%M')`"
subtitle: ''

output:
  html_document:
    template: RDBFIS_report_template.html
    toc: true
    toc-title: "Table of contents"
    toc_depth: 5
    number_sections: true
    collapsed: true
    df_print: paged
params:
  term: FDI datacall
---
<!-- INITIALIZATION -->

```{r initialization, include=FALSE}
knitr::opts_chunk$set(
  echo = FALSE,
  message = FALSE,
  warning = FALSE
)

# loading needed libraries
lib  <-  c("RDBqc",  "knitr",  "kableExtra",  "dplyr",  "ggplot2",  "rworldmap",  "sp",  "rworldxtra",  "pander",
"data.table", "grDevices", "magrittr", "tictoc", "tidyverse", "fishmethods", "tidyr", "gridExtra", "outliers")
lapply(lib, require, character.only = TRUE)
```

<!-- USER DEFINED SOURCE FILES -->

```{r user_setup, include=FALSE}

## reading files
## UNCOMMENT  THE FOLLOWING LINES FOR STAN-ALONE USE
## SELECT THE APPROPRIATE FILE PATHS ON THE LOCAL FOLDERS

tableA <- read.table("./fdi_a_catch.csv", sep = ";", header = TRUE)
tableG <- read.table("./fdi_g_effort.csv", sep = ";", header = TRUE)
tableH <- read.table("./fdi_h_spatial_landings.csv", sep = ";", header = TRUE)
tableI <- read.table("./fdi_i_spatial_effort.csv", sep = ";", header = TRUE)
tableJ <- read.table("./fdi_j_capacity.csv", sep = ";", header = TRUE)

MS <- "ITA"
SPs <- c("DPS")
GSAs <- c("GSA99")
vessel_len <- "COMBINED"
fishtech <- "COMBINED"
metier <- "COMBINED"
```

<!-- DO NOT MODIFY THE FOLLOWING CODE -->

```{r setup, include=FALSE}
# check the presence of user defined filter for member state (MS)
if (exists("MS")) {
  if (length(MS) == 1 & is.na(MS[1])) {
    user_MS <- FALSE
  } else {
    user_MS <- TRUE
  }
} else {
  user_MS <- FALSE
}

# check the presence of user defined filter for sub-area (GSAs)
if (exists("GSAs")) {
  if (length(GSAs) == 1 & is.na(GSAs[1])) {
    user_GSA <- FALSE
  } else {
    user_GSA <- TRUE
  }
} else {
  user_GSA <- FALSE
}

# check the presence of user defined filter for species (SPs)
if (exists("SPs")) {
````
````

```
  if (length(SPs) == 1 & is.na(SPs[1])) {
    user_SP <- FALSE
  } else {
    user_SP <- TRUE
  }
} else {
  user_SP <- FALSE
}

# check the presence of user defined filter for vessel length (vessel_len)
if (exists("vessel_len")) {
  if (length(vessel_len) == 1 & is.na(vessel_len[1])) {
    user_vessel_len <- FALSE
  } else {
    user_vessel_len <- TRUE
  }
} else {
  user_vessel_len <- FALSE
}

# check the presence of user defined filter for fishing technique (fishtech)
if (exists("fishtech")) {
  if (length(fishtech) == 1 & is.na(fishtech[1])) {
    user_fishtech <- FALSE
  } else {
    user_fishtech <- TRUE
  }
} else {
  user_fishtech <- FALSE
}

# check the presence of user defined filter for metier (metier)
if (exists("metier")) {
  if (length(metier) == 1 & is.na(metier[1])) {
    user_metier <- FALSE
  } else {
    user_metier <- TRUE
  }
} else {
  user_metier <- FALSE
}
```

```
\newpage
<!-- Table A catch -->

```{r check_tableA, results = 'asis'}

if (exists("tableA")) {
  if (class(tableA) == "data.frame") {
    if (nrow(tableA) > 0) {
      check_tableA <- TRUE
      cat(paste0("\n# FDI catch data (Table A)\n"))

      if (!user_SP) {
        SPs <- sort(unique(tableA$species))
      }
      if (!user_MS) {
        MS <- unique(tableA$country)
      }
      if (!user_GSA) {
        GSAs <- sort(unique(tableA$sub_region))
      }
      if (!user_vessel_len) {
vessel_len <- sort(unique(tableA$vessel_length))
}
      if (!user_fishtech) {
        fishtech <- sort(unique(tableA$fishing_tech))
      }

      tableA <- tableA[tableA$species %in% SPs &
        tableA$country == MS &
        tableA$sub_region %in% GSAs, ]
    }
  }
} else {
  check_tableA <- FALSE
  cat("\n No FDI catch data available \n")
}
```

```{r FDI_EF_A, results = 'asis'}
if (check_tableA) {
  cat("\n## Check the presence of empty fiels\n")
  cat("\nFDI catch data (table A) were checked for the presence of not allowed empty data. In case empty fields
are detected, a list of fields with the number of empty records is reported.\n")
```
```

```
  res <- suppressMessages(check_EF_FDI_A(data = tableA, verbose = FALSE))

  if (!is.null(res)) {
    res <- res[[1]][which(res[[1]] > 0)]
    if (length(res) > 0) {
      res <- data.frame(Empty_records = res)
      cat("\n\n\n")
      cat(paste0("\n_Table of empty records in FDI catch table (Table A)_\n"))
      print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No empty records detected in FDI catch table A\n"))
    }
  } else {
    cat(paste0("\n\n\n- No data available in FDI catch table A\n"))
  }
} # check_TableA
```

```
```{r FDI_RD_A, results = 'asis'}
if (check_tableA) {
  cat("\n## Check the presence of duplicated records\n")
  cat("\nFDI catch data (table A) were checked for the presence of duplicated records. In particular, it was
checks whether the combination of the first 19 columns generates duplicate records.\n")

  res <- suppressMessages(check_RD_FDI_A(data = tableA, verbose = FALSE))

  if (length(res) > 0) {
    l <- length(res)
    cat(paste0("\n\n\n- ", l, " duplicated record/s was/were detected in the FDI catch table A\n\n\n"))
    df <- tableA[res, which(colnames(tableA) %in% c(
      "country", "year", "quarter", "vessel_length", "fishing_tech", "gear_type",
      "target_assemblage", "mesh_size_range", "metier",
      "domain_discards", "domain_landings", "sub_region",
      "species"
    ))]

    cat("\n\n\n")
    cat(paste0("\n_Table of duplicated records in FDI catch table (Table A)_\n"))
    print(df %>% kable("html") %>% kable_styling(font_size = 11) %>% kable_classic(full_width = TRUE))
    cat("\n\n\n")
  } else {
    cat(paste0("\n\n\n- No duplicated records were detected in the FDI catch table A\n\n\n"))
  }
} # check_TableA
```
```

```
```{r FDI_coverage_A, results = 'asis'}
if (check_tableA) {
  cat("\n## Coverage of FDI catch data by GSA and year\n")
  cat("\nFDI catch data (table A) were analysed to report a summary table of the data coverage in terms of number
of records by country, GSA and year.\n")

  res <- suppressMessages(FDI_coverage(data = tableA, MS, verbose = FALSE))

  if (!is.null(res)) {
    if (nrow(res) > 0) {
      cat("\n\n\n")
      cat(paste0("\n_Summary table of number of records by country, GSA and year (Table A)._\n"))
      print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No data available in FDI catch table A\n"))
    }
  } else {
    cat(paste0("\n\n\n- No data available in FDI catch table A\n"))
  }
} # check_TableA
```
```

```
```{r FDI_cov_tableA, results = 'asis'}
if (check_tableA) {
  cat("\n## Cross-coverage plots of landing and discard data in FDI catch table A\n")
  cat("\nThe FDI catch table was analyzed to return plots for three variables (Total live weight landed (ton),
total value of landings (euro), and total discards (ton)) grouped by year, vessels length, and fishing
techniques.\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 2
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      suppressMessages(res <- FDI_cov_tableA(data = tableA, MS = MS, GSA = GSAs[g], SP = SPs[s], vessel_len =
vessel_len, fishtech = fishtech, verbose = FALSE))

      if (!is.null(res)) {
        cat("\n\n\n")
```
```

```
        cat(paste0("\nPlot of total landing by year for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
suppressMessages(print(res[[3]]))
        cat("\n\n\n")

cat("\n\n\n")
        cat(paste0("\nPlot of total value by year for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
suppressMessages(print(res[[4]]))
        cat("\n\n\n")

cat("\n\n\n")
        cat(paste0("\nPlot of total discard by year for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
suppressMessages(print(res[[5]]))
        cat("\n\n\n")
} else {
        cat(paste0("\n\n\n- No length data available for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
      }
    }
  }
} # check_tableA
```

```{r FDI_disc_coverage, results = 'asis'}
if (check_tableA) {
  cat("\n## Cross-coverage summary table of landing and discard data in FDI catch table A\n")
  cat("\nThe FDI catch table was analyzed to return a summary table reporting the amount and relative percentage
of landings taking into account the availability of discard data by year (discard >0, =0 and =NK).\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      suppressMessages(res <- FDI_disc_coverage(data = tableA, MS = MS, GSA = GSAs[g], SP = SPs[s], verbose =
FALSE))

      if (!is.null(res)) {
        if (nrow(res) > 0) {
          cat(paste0("\n_Summary table landing amount and relative percentage accounting for discard availability
for ", SPs[s], " in ", GSAs[g], "_\n"))

          print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
          cat("\n\n\n")
        } else {
          cat(paste0("\n\n\n- No data available for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
        }
      } else {
        cat(paste0("\n\n\n- No data available for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
      }
    }
  }
} # check_tableA
```

```{r FDI_price_not_null, results = 'asis'}
if (check_tableA) {
  cat("\n## Species value\n")
  cat("\nThe FDI catch table was analyzed to return the estimates of the average price per species and year, that
is further compared with the average price calculated per country (by species). Furthermore, in order to compare
total weight landings and total value landings, the cases with total landings > 0 but landings value = 0 were
identified and reported in a table, were available.\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      suppressMessages(res <- FDI_prices_not_null(data = tableA, MS = MS, GSA = GSAs[g], SP = SPs[s], verbose =
FALSE))

      if (!is.null(res)) {
        if (!is.null(res[[2]])) {
          cat(paste0("\n_Summary table of landings > 0 and landings value = 0 for ", SPs[s], " in ", GSAs[g],
"_\n"))
          print(res[[2]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
          cat("\n\n\n")
        } else {
          cat(paste0("\n\n\n- No data available with landings > 0 and landings value = 0 for ", SPs[s], " - ", MS,
" - ", GSAs[g], "\n"))
        }

        if (!is.null(res[[1]])) {
          cat(paste0("\n_Summary table of avarage prices by year in ", SPs[s], " in ", GSAs[g], "_\n"))
          print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
          cat("\n\n\n")
        } else {
          cat(paste0("\n\n\n- No avarage prices available for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
        }
```

```
      } else {
        cat(paste0("\n\n\n- No data available for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
      }
    }
  }
} # check_tableA
```

```{r FDI_prices_cov, results = 'asis'}
if (check_tableA) {
  cat("\n## Species' prices trend\n")
  cat("\nPlots by species and sub-region were generated to check the trends of the prices time series.\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      suppressMessages(res <- FDI_prices_cov(data = tableA, MS = MS, GSA = GSAs[g], SP = SPs[s], verbose = FALSE))

      if (!is.null(res)) {
        cat("\n\n\n")
        cat(paste0("\nPlot of average price by year for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
suppressMessages(print(res[[3]]))
        cat("\n\n\n")
cat(paste0("\n_Summary table of species' prices by year for ", SPs[s], " in ", GSAs[g], "_\n"))
        print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      } else {
        cat(paste0("\n\n\n- No data available for ", SPs[s], " - ", MS, " - ", GSAs[g], "\n"))
      }
    }
  }
} # check_tableA
```

```
\newpage
<!-- Table G catch -->
```

```{r check_tableG, results = 'asis'}

if (exists("tableG")) {
  if (class(tableG) == "data.frame") {
    if (nrow(tableG) > 0) {
      check_tableG <- TRUE
      cat(paste0("\n# FDI effort data (table G)\n"))

      if (!user_MS) {
        MS <- unique(tableG$country)
      }
      if (!user_GSA) {
GSAs <- unique(unique(tableG$sub_region))
}

      if (!user_fishtech) {
        fishtech <- unique(tableG$fishing_tech)
      }
      if (!user_metier) {
        metier <- unique(tableG$metier)
      }
      tableG <- tableG[tableG$country == MS &
        tableG$sub_region %in% GSAs, ]
    }
  }
} else {
  check_tableG <- FALSE
  cat("\n No FDI catch data available \n")
}
```

```{r FDI_EF_G, results = 'asis'}
if (check_tableG) {
  cat("\n## Check the presence of empty fields\n")
  cat("\nFDI effort data (table G) were checked for the presence of not allowed empty records. In case empty data
are detected, a list of fields with the number of empty records is reported.\n")

  res <- suppressMessages(check_EF_FDI_G(data = tableG, verbose = FALSE))

  if (!is.null(res)) {
    res <- res[[1]][which(res[[1]] > 0)]
    if (length(res) > 0) {
      res <- data.frame(Empty_records = res)
      cat("\n\n\n")
      cat(paste0("\n_Table of empty records in FDI effort table (Table G)_\n"))
      print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No empty records detected in FDI effort table G\n"))
```

```
    }
  } else {
    cat(paste0("\n\n\n- No data available in FDI effort table G\n"))
  }
} # check_TableG
```

```{r FDI_RD_G, results = 'asis'}
if (check_tableG) {
  cat("\n## Check the presence of duplicated records\n")
  cat("\nFDI effort data (table G) were checked for the presence of duplicated records. In particular, it was checks whether the combination of the first 15 columns generates duplicate records.\n")

  res <- suppressMessages(check_RD_FDI_G(data = tableG, verbose = FALSE))

  if (length(res) > 0) {
    l <- length(res)
    cat(paste0("\n\n\n- ", l, " duplicated record/s was/were detected in the FDI effort table G\n\n\n"))
    df <- tableG[res, which(colnames(tableG) %in% c(
      "country", "year", "quarter", "vessel_length", "fishing_tech", "gear_type",
      "target_assemblage", "mesh_size_range", "metier",
      "sub_region", "specon_tech", "deep"
    ))]

    cat("\n\n\n")
    cat(paste0("\n_Table of duplicated records in FDI effort table (Table G)_\n"))
    print(df %>% kable("html") %>% kable_styling(font_size = 11) %>% kable_classic(full_width = TRUE))
    cat("\n\n\n")
  } else {
    cat(paste0("\n\n\n- No duplicated records were detected in the FDI effort table G\n\n\n"))
  }
} # check_TableA
```

```{r FDI_coverage_G, results = 'asis'}
if (check_tableG) {
  cat("\n## Coverage of FDI effort data by GSA and year\n")
  cat("\nFDI effort data (table G) were analysed to report a summary table of the data coverage in terms of number of records by country, GSA and year.\n")

  res <- suppressMessages(FDI_coverage(data = tableG, MS, verbose = FALSE))

  if (!is.null(res)) {
    if (nrow(res) > 0) {
      cat("\n\n\n")
      cat(paste0("\n_Summary table of number of records by country, GSA and year (Table G)._\n"))
      print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No data available in FDI effort table G\n"))
    }
  } else {
    cat(paste0("\n\n\n- No data available in FDI effort table G\n"))
  }
} # check_TableG
```

```{r FDI_cov_tableG, results = 'asis'}
if (check_tableG) {
  cat("\n## Coverage of Fishing effort in FDI table G\n")
  cat("\nThe FDI effort table G was analyzed to return plots for the following 8 variables:
      \n1. Total days at sea;
      \n2. Total Fishing Days;
      \n3. Total kW * days at Sea;
      \n4. total GT * days at sea;
      \n5. Total kW * fishing days;
      \n6. Total GT * fishing days;
      \n7. Hours at Sea;
      \n8. Total kW * hours at sea.\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))

    suppressMessages(res <- FDI_cov_tableG(data = tableG, MS = MS, GSA = GSAs[g], fishtech = fishtech, met = metier, verbose = FALSE))

    if (!is.null(res)) {
      cat("\n\n\n")
      cat(paste0("\nPlot of Total days at sea for ", MS, " - ", GSAs[g], "\n"))
      suppressWarnings(print(res[[3]]))
      cat("\n\n\n")

      cat("\n\n\n")
      cat(paste0("\nPlot of Total Fishing Days for ", MS, " - ", GSAs[g], "\n"))
      suppressWarnings(print(res[[4]]))
      cat("\n\n\n")
```

```
      cat("\n\n\n")
      cat(paste0("\nPlot of Fishing effort in Total kW times days at Sea for ", MS, " - ", GSAs[g], "\n"))
      suppressWarnings(print(res[[5]]))
      cat("\n\n\n")

      cat("\n\n\n")
      cat(paste0("\nPlot of Fishing effort in total GT times days at sea for ", MS, " - ", GSAs[g], "\n"))
      suppressWarnings(print(res[[6]]))
      cat("\n\n\n")

      cat("\n\n\n")
      cat(paste0("\nPlot of Fishing effort in kW times fishing days for ", MS, " - ", GSAs[g], "\n"))
      suppressWarnings(print(res[[7]]))
      cat("\n\n\n")

      cat("\n\n\n")
      cat(paste0("\nPlot of Fishing effort in gross tonnage times fishing days for ", MS, " - ", GSAs[g], "\n"))
      suppressWarnings(print(res[[8]]))
      cat("\n\n\n")

      cat("\n\n\n")
      cat(paste0("\nPlot of Hours at Sea for ", MS, " - ", GSAs[g], "\n"))
      suppressWarnings(print(res[[9]]))
      cat("\n\n\n")

      cat("\n\n\n")
      cat(paste0("\nPlot of kW times hours at sea for ", MS, " - ", GSAs[g], "\n"))
      suppressWarnings(print(res[[10]]))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No data available for ", MS, " - ", GSAs[g], "\n"))
    }
  }
} # check_tableG
```

```
\newpage
<!-- Table H catch -->

```{r check_tableH, results = 'asis'}
if (exists("tableH")) {
  if (class(tableH) == "data.frame") {
    if (nrow(tableH) > 0) {
      check_tableH <- TRUE
      cat(paste0("\n# FDI Landings by rectangle data (table H)\n"))

      if (!user_SP) {
        SPs <- sort(unique(tableH$species))
      }
      if (!user_MS) {
        MS <- unique(tableH$country)
      }
      if (!user_GSA) {
        GSAs <- sort(unique(tableH$sub_region))
      }
      tableH <- tableH[tableH$species %in% SPs &
        tableH$country == MS &
        tableH$sub_region %in% GSAs, ]
    }
  }
} else {
  check_tableH <- FALSE
  cat("\n No FDI Landings by rectangle data available \n")
}
```

```{r FDI_EF_H, results = 'asis'}
if (check_tableH) {
  cat("\n## Check the presence of empty fields\n")
  cat("\nFDI Landings by rectangle data (table H) were checked for the presence of not allowed empty records. In
case empty data are detected, a list of fields with the number of empty records is reported.\n")

  res <- suppressMessages(check_EF_FDI_H(data = tableH, verbose = FALSE))

  if (!is.null(res)) {
    res <- res[[1]][which(res[[1]] > 0)]
    if (length(res) > 0) {
      res <- data.frame(Empty_records = res)
      cat("\n\n\n")
      cat(paste0("\n_Table of empty records in FDI Landings by rectangle table (Table H)_\n"))
      print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No empty records detected in FDI Landings by rectangle table H\n"))
    }
  } else {
    cat(paste0("\n\n\n- No data available in FDI Landings by rectangle table H\n"))
```
```

```
  }
} # check_TableH
```

```{r FDI_RD_H, results = 'asis'}
if (check_tableH) {
  cat("\n## Check the presence of duplicated records\n")
  cat("\nFDI Landings by rectangle data (table H) were checked for the presence of duplicated records. In particular, it was checks whether the combination of the first 20 columns generates duplicate records.\n")

  res <- suppressMessages(check_RD_FDI_H(data = tableH, verbose = FALSE))

  if (length(res) > 0) {
    l <- length(res)
    cat(paste0("\n\n\n- ", l, " duplicated record/s was/were detected in the FDI Landings by rectangle table H\n\n\n"))
    df <- tableH[res, which(colnames(tableH) %in% c(
      "country", "year", "quarter", "vessel_length", "fishing_tech", "gear_type",
      "target_assemblage", "mesh_size_range", "metier",
      "sub_region", "rectangle_type", "rectangle_lat", "rectangle_lon",
      "c_square", "species"
    ))]

    cat("\n\n\n")
    cat(paste0("\n_Table of duplicated records in FDI Landings by rectangle table (Table H)_\n"))
    print(df %>% kable("html") %>% kable_styling(font_size = 11) %>% kable_classic(full_width = TRUE))
    cat("\n\n\n")
  } else {
    cat(paste0("\n\n\n- No duplicated records were detected in the FDI Landings by rectangle table H\n\n\n"))
  }
} # check_TableH
```

```{r FDI_coverage_H, results = 'asis'}
if (check_tableH) {
  cat("\n## Coverage of FDI Landings by rectangle data by GSA and year\n")
  cat("\nFDI Landings by rectangle data (table H) were analysed to report a summary table of the data coverage in terms of number of records by country, GSA and year.\n")

  res <- suppressMessages(FDI_coverage(data = tableH, MS, verbose = FALSE))

  if (!is.null(res)) {
    if (nrow(res) > 0) {
      cat("\n\n\n")
      cat(paste0("\n_Summary table of number of records by country, GSA and year (Table H)._\n"))
      print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No data available in FDI Landings by rectangle table H\n"))
    }
  } else {
    cat(paste0("\n\n\n- No data available in FDI Landings by rectangle table H\n"))
  }
} # check_TableH
```

```{r FDI_checks_spatial_H, results = 'asis'}
if (check_tableH) {
  cat("\n## Consistency of spatial  data in FDI Landings by rectangle table H\n")
  cat("\nFDI Landings by rectangle data (table H) were analysed to report, where available:
      \n1. the records with incorrect combination of NA in the spatial columns 'rectangle_type', 'rectangle_lat', and 'rectangle_lon'
      \n2. the records without any sub_region assignment.\n")

  res <- suppressMessages(FDI_checks_spatial_HI(data = tableH, MS, verbose = FALSE))

  if (!is.null(res)) {
    if (!is.null(res[[1]])) {
      if (nrow(res[[1]])) {
        cat("\n\n\n")
        cat(paste0("\n_Table of incorrect combination of NA in the spatial columns_\n"))
        print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
        cat("\n\n\n")
      } else {
        cat(paste0("\n\n\n- No incorrect combination of NA in the spatial columns\n"))
      }
    } # !is.null(res[[1]])

    if (!is.null(res[[2]])) {
      if (nrow(res[[2]])) {
        cat("\n\n\n")
        cat(paste0("\n_Table of records without sub_region assignment_\n"))
        print(res[[2]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
        cat("\n\n\n")
      } else {
        cat(paste0("\n\n\n- No records without sub_region assignment\n"))
      }
```

```
    } # !is.null(res[[2]])
  } # !is.null(res)
} # check_TableH
```

```{r FDI_check_coord_H, results = 'asis'}
if (check_tableH) {
  cat("\n## Compatibility of the geographical coordinates with rectangle type\n")
  cat("\nFDI Landings by rectangle data (table H) were analysed to report, where available, in which at least one among latitude and longitude is not compatible with the rectangle type.\n")

  res <- suppressMessages(FDI_check_coord(data = tableH, MS, verbose = FALSE))

  if (!is.null(res)) {
    if (nrow(res) > 0) {
      cat("\n\n\n")
      cat(paste0("\n_Table of incorrect combination of NA in the spatial columns_\n"))
      print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No records in which at least one among latitude and longitude is not compatible with the rectangle type\n"))
    }
  } else {
    cat(paste0("\n\n\n- No data available in FDI Landings by rectangle table H\n"))
  }
} # check_TableH
```

\newpage
<!-- Table I catch -->

```{r check_tableI, results = 'asis'}
if (exists("tableI")) {
  if (class(tableI) == "data.frame") {
    if (nrow(tableI) > 0) {
      check_tableI <- TRUE
      cat(paste0("\n# FDI Effort by rectangle data (table I)\n"))

      if (!user_MS) {
        MS <- unique(tableI$country)
      }
      if (!user_GSA) {
        GSAs <- sort(unique(tableI$sub_region))
      }
      tableI <- tableI[tableI$country == MS &
        tableI$sub_region %in% GSAs, ]
    }
  }
} else {
  check_tableI <- FALSE
  cat("\n No FDI Landings by rectangle data available \n")
}
```

```{r FDI_EF_I, results = 'asis'}
if (check_tableI) {
  cat("\n## Check the presence of empty fields\n")
  cat("\nFDI Effort by rectangle data (table I) were checked for the presence of not allowed empty records. In case empty data are detected, a list of fields with the number of empty records is reported.\n")

  res <- suppressMessages(check_EF_FDI_I(data = tableI, verbose = FALSE))

  if (!is.null(res)) {
    res <- res[[1]][which(res[[1]] > 0)]
    if (length(res) > 0) {
      res <- data.frame(Empty_records = res)
      cat("\n\n\n")
      cat(paste0("\n_Table of empty records in FDI Effort by rectangle table (Table I)_\n"))
      print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No empty records detected in FDI Effort by rectangle table I\n"))
    }
  } else {
    cat(paste0("\n\n\n- No data available in FDI Effort by rectangle table I\n"))
  }
} # check_TableI
```

```{r FDI_RD_I, results = 'asis'}
if (check_tableI) {
  cat("\n## Check the presence of duplicated records\n")
  cat("\nFDI Effort by rectangle data (table I) were checked for the presence of duplicated records. In particular, it was checks whether the combination of the first 19 columns generates duplicate records.\n")

  res <- suppressMessages(check_RD_FDI_I(data = tableI, verbose = FALSE))
```

```
  if (length(res) > 0) {
    l <- length(res)
    cat(paste0("\n\n\n- ", l, " duplicated record/s was/were detected in the FDI Effort by rectangle table
I\n\n\n"))
    df <- tableI[res, which(colnames(tableI) %in% c(
      "country", "year", "quarter", "vessel_length", "fishing_tech", "gear_type",
      "target_assemblage", "mesh_size_range", "metier",
      "sub_region", "rectangle_type", "rectangle_lat", "rectangle_lon",
      "c_square"
    ))]

    cat("\n\n\n")
    cat(paste0("\n_Table of duplicated records in FDI Effort by rectangle table (Table I)_\n"))
    print(df %>% kable("html") %>% kable_styling(font_size = 11) %>% kable_classic(full_width = TRUE))
    cat("\n\n\n")
  } else {
    cat(paste0("\n\n\n- No duplicated records were detected in the FDI Effort by rectangle table I\n\n\n"))
  }
} # check_TableI
```

```{r FDI_coverage_I, results = 'asis'}
if (check_tableI) {
  cat("\n## Coverage of FDI Effort by rectangle data by GSA and year\n")
  cat("\nFDI Effort by rectangle data (table I) were analysed to report a summary table of the data coverage in
terms of number of records by country, GSA and year.\n")

  res <- suppressMessages(FDI_coverage(data = tableI, MS, verbose = FALSE))

  if (!is.null(res)) {
    if (nrow(res) > 0) {
      cat("\n\n\n")
      cat(paste0("\n_Summary table of number of records by country, GSA and year (Table I)._\n"))
      print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No data available in FDI Effort by rectangle table I\n"))
    }
  } else {
    cat(paste0("\n\n\n- No data available in FDI Effort by rectangle table I\n"))
  }
} # check_TableI
```

```{r FDI_checks_spatial_I, results = 'asis'}
if (check_tableI) {
  cat("\n## Consistency of spatial  data in FDI Effort by rectangle table I\n")
  cat("\nFDI Effort by rectangle data (table I) were analysed to report, where available:
      \n1. the records with incorrect combination of NA in the spatial columns 'rectangle_type', 'rectangle_lat',
and 'rectangle_lon'
      \n2. the records without any sub_region assignment.\n")

  res <- suppressMessages(FDI_checks_spatial_HI(data = tableI, MS, verbose = FALSE))

  if (!is.null(res)) {
    if (!is.null(res[[1]])) {
      if (nrow(res[[1]])) {
        cat("\n\n\n")
        cat(paste0("\n_Table of incorrect combination of NA in the spatial columns_\n"))
        print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
        cat("\n\n\n")
      } else {
        cat(paste0("\n\n\n- No incorrect combination of NA in the spatial columns\n"))
      }
    } # !is.null(res[[1]])


    if (!is.null(res[[2]])) {
      if (nrow(res[[2]])) {
        cat("\n\n\n")
        cat(paste0("\n_Table of records without sub_region assignment_\n"))
        print(res[[2]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
        cat("\n\n\n")
      } else {
        cat(paste0("\n\n\n- No records without sub_region assignment\n"))
      }
    } # !is.null(res[[2]])
  } # !is.null(res)
} # check_TableI
```

```{r FDI_check_coord_I, results = 'asis'}
if (check_tableI) {
  cat("\n## Compatibility of the geographical coordinates with rectangle type\n")
  cat("\nFDI Effort by rectangle data (table I) were analysed to report, where available, in which at least one
among latitude and longitude is not compatible with the rectangle type.\n")
```

```
  res <- suppressMessages(FDI_check_coord(data = tableI, MS, verbose = FALSE))

  if (!is.null(res)) {
    if (nrow(res) > 0) {
      cat("\n\n\n")
      cat(paste0("\n_Table of incorrect combination of NA in the spatial columns_\n"))
      print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No records in which at least one among latitude and longitude is not compatible with the
rectangle type\n"))
    }
  } else {
    cat(paste0("\n\n\n- No data available in FDI Effort by rectangle table I\n"))
  }
} # check_TableI
```

```
\newpage
<!-- Table J Capacity and fleet segment effort -->

```{r check_tableJ, results = 'asis'}
if (exists("tableJ")) {
  if (class(tableJ) == "data.frame") {
    if (nrow(tableJ) > 0) {
      check_tableJ <- TRUE
      cat(paste0("\n# FDI Capacity and fleet segment effort data (table J)\n"))

      if (!user_MS) {
        MS <- unique(tableJ$country)
      }
      if (!user_GSA) {
        GSAs <- sort(unique(tableJ$principal_sub_region))
      }
      if (!user_vessel_len) {
        vessel_len <- unique(tableJ$vessel_length)
      }
      if (!user_fishtech) {
        fishtech <- unique(tableJ$fishing_tech)
      }

      tableJ <- tableJ[tableJ$country == MS &
        tableJ$principal_sub_region %in% GSAs, ]
    }
  }
} else {
  check_tableJ <- FALSE
  cat("\n No FDI Capacity and fleet segment effort data available \n")
}
```

```{r FDI_EF_J, results = 'asis'}
if (check_tableJ) {
  cat("\n## Check the presence of empty fields\n")
  cat("\nFDI Capacity and fleet segment effort data (table J) were checked for the presence of not allowed empty
records. In case empty data are detected, a list of fields with the number of empty records is reported.\n")

  res <- suppressMessages(check_EF_FDI_J(data = tableJ, verbose = FALSE))
  if (!is.null(res)) {
    res <- res[[1]]
    res <- res[which(res > 0)]
    if (length(res) > 0) {
      res <- data.frame(Empty_records = res)
      cat("\n\n\n")
      cat(paste0("\n_Table of empty records in FDI Capacity and fleet segment effort table (Table J)_\n"))
      print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No empty records detected in FDI Capacity and fleet segment effort table J\n"))
    }
  } else {
    cat(paste0("\n\n\n- No data available in FDI Capacity and fleet segment effort table J\n"))
  }
} # check_TableJ
```

```{r FDI_RD_J, results = 'asis'}
if (check_tableJ) {
  cat("\n## Check the presence of duplicated records\n")
  cat("\nFDI Capacity and fleet segment effort data (table J) were checked for the presence of duplicated records.
In particular, it was checks whether the combination of the first 19 columns generates duplicate records.\n")

  res <- suppressMessages(check_RD_FDI_J(data = tableJ, verbose = FALSE))

  if (length(res) > 0) {
    l <- length(res)
```
```

```
    cat(paste0("\n\n\n- ", l, " duplicated record/s was/were detected in the FDI Capacity and fleet segment effort
table J\n\n\n"))
    df <- tableJ[res, which(colnames(tableJ) %in% c(
      "country", "year", "vessel_length", "fishing_tech", "supra_region",
      "geo_indicator", "principal_sub_region"
    ))]

    cat("\n\n\n")
    cat(paste0("\n_Table of duplicated records in FDI Capacity and fleet segment effort table (Table J)_\n"))
    print(df %>% kable("html") %>% kable_styling(font_size = 11) %>% kable_classic(full_width = TRUE))
    cat("\n\n\n")
  } else {
    cat(paste0("\n\n\n- No duplicated records were detected in the FDI Capacity and fleet segment effort table
J\n\n\n"))
  }
} # check_TableJ
```

```{r FDI_coverage_J, results = 'asis'}
if (check_tableJ) {
  cat("\n## Coverage of FDI Capacity and fleet segment effort data by GSA and year\n")
  cat("\nFDI Capacity and fleet segment effort data (table J) were analysed to report a summary table of the data
coverage in terms of number of records by country, GSA and year.\n")

  res <- suppressMessages(FDI_coverage(data = tableJ, MS, verbose = FALSE))

  if (!is.null(res)) {
    if (nrow(res) > 0) {
      cat("\n\n\n")
      cat(paste0("\n_Summary table of number of records by country, GSA and year (Table J)._\n"))
      print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No data available in FDI Capacity and fleet segment effort table J\n"))
    }
  } else {
    cat(paste0("\n\n\n- No data available in FDI Capacity and fleet segment effort table J\n"))
  }
} # check_TableJ
```

```{r FDI_cov_tableJ, results = 'asis'}
if (check_tableJ) {
  cat("\n## Coverage of FDI Capacity and fleet segment effort table\n")
  cat("\nThe FDI Capacity table was analyzed to return plots for four variables (total trips; total kW; total GT;
total vessels) grouped by year, vessels length, and fishing techniques.\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### ", GSAs[g], "\n"))
    suppressMessages(res <- FDI_cov_tableJ(data = tableJ, MS = MS, GSA = GSAs[g], vessel_len = vessel_len,
fishtech = fishtech, verbose = FALSE))

    if (!is.null(res)) {
      cat("\n\n\n")
      cat(paste0("\nPlot of total trips by year for ", MS, " - ", GSAs[g], "\n"))
print(res[[3]])
      cat("\n\n\n")

cat("\n\n\n")
      cat(paste0("\nPlot of total kW by year for ", MS, " - ", GSAs[g], "\n"))
print(res[[4]])
      cat("\n\n\n")

cat("\n\n\n")
      cat(paste0("\nPlot of total GT by year for ", MS, " - ", GSAs[g], "\n"))
print(res[[5]])
      cat("\n\n\n")

cat("\n\n\n")
      cat(paste0("\nPlot of total vessels by year for ", MS, " - ", GSAs[g], "\n"))
print(res[[6]])
      cat("\n\n\n")
} else {
      cat(paste0("\n\n\n- No data available for ", MS, " - ", GSAs[g], "\n"))
    }
  }
} # check_tableA
```

```{r FDI_vessel_lenth, results = 'asis'}
if (check_tableJ) {
  cat("\n## Consistency of vessel length with vessel length category\n")
  cat("\nThe FDI Capacity and fleet segment effort table (Table J) was analyzed to identify possible
inconsistencies between average vessels length and vessel length category. Two tables are generated. The first one
reports the records in which either vessel length or vessel category or both are not present, if any. The second
table reports the cases in which vessel length does not match vessel length category, if any.\n")
```

```
  suppressMessages(res <- FDI_vessel_lenth(data = tableJ, MS = MS, verbose = FALSE))

  if (!is.null(res)) {
    if (!is.null(res[[1]])) {
      cat(paste0("\n_Summary table of records in which either vessel lengths or vessel categories or both are not
reported._\n"))
      print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No   cases in which either vessel lengths or vessel categories or both are not
reported\n"))
    }
    if (!is.null(res[[2]])) {
      cat(paste0("\n_Summary table of records in which vessel length does not match vessel length category._\n"))
      print(res[[2]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No cases in which vessel length does not match vessel length category.\n"))
    }
  } else {
    cat(paste0("\n\n\n- No data available for ", MS, "\n"))
  }
} # check_tableA
```

<!-- CROSS-CHECKS -->

```{r FDI_fishdays_cov, results = 'asis'}

cat(paste0("\n# Cross-tables checks\n"))

if (check_tableG & check_tableI) {
  cat("\n## Comparison of total fishing days in tables G and I\n")
  cat("\nBoth FDI table G (Effort) and I (Effort by rectangle) were analysed to report the mutual consistency of
the total fishing days and the percentage difference between the two tables.\n")

  res <- suppressMessages(FDI_fishdays_cov(dataG = tableG, dataI = tableI, MS, verbose = FALSE))

  if (!is.null(res)) {
    if (nrow(res) > 0) {
      cat("\n\n\n")
      cat(paste0("\n_Summary table of total fishing days by country, GSA and year (Table G and I)._\n"))
      print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No data available in FDI table G or I\n"))
    }
  } else {
    cat(paste0("\n\n\n- No data available in FDI table G or I for ", MS, "\n"))
  }
} else {
  cat("\nOne or both of tables G and I are missing to perform the comparison on totalfishdays.\n")
} # check_Table G and I
```

```{r FDI_landweight_cov, results = 'asis'}
if (check_tableA & check_tableH) {
  cat("\n## Comparison of landing volumes in tables A and H\n")
  cat("\nBoth FDI table A (Catch) and H (Landings by rectangle) were analysed to report the mutual consistency of
the landing volumes and the percentage difference between the two tables.\n")

  res <- suppressMessages(FDI_landweight_cov(dataA = tableA, dataH = tableH, MS, verbose = FALSE))

  if (!is.null(res)) {
    if (nrow(res) > 0) {
      cat("\n\n\n")
      cat(paste0("\n_Summary table of landing volumes by country, GSA and year (Table A and H)._\n"))
      print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No data available in FDI table A or H\n"))
    }
  } else {
    cat(paste0("\n\n\n- No data available in FDI table A or H for ", MS, "\n"))
  }
} else {
  cat("\nOne or both of tables A and H are missing to perform the comparison on landing volumes\n")
} # check_Table G and I
```

```{r FDI_vessel_numbers, results = 'asis'}
if (check_tableG & check_tableJ) {
  cat("\n## Consistency of number of vessels in FDI table G and J\n")
  cat("\nBoth FDI table G (Effort) and J (Capacity and fleet segment effort) were analysed to report the mutual
consistency of the number of vessels and the percentage difference between the two tables.\n")

  res <- suppressMessages(FDI_vessel_numbers(dataJ = tableJ, dataG = tableG, MS, verbose = FALSE))
```

```
  if (!is.null(res)) {
    if (!is.null(res[[1]])) {
      if (nrow(res[[1]]) > 0) {
        cat("\n\n\n")
        cat(paste0("\n_Summary table of number of vessels by country, GSA and year (Table J and G)._\n"))
        print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
        cat("\n\n\n")
      } else {
        cat(paste0("\n\n\n- No data available in FDI table J or G\n"))
      }
    }

    if (!is.null(res[[2]])) {
      if (nrow(res[[2]]) > 0) {
        cat("\n\n\n")
        cat(paste0("\n_Summary table of cases in which cases the number of vessels > 0 in table J but not existing
in table G._\n"))
        print(res[[2]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
        cat("\n\n\n")
      } else {
        cat(paste0("\n\n\n- No data available in FDI table J or G\n"))
      }
    }
  } else {
    cat(paste0("\n\n\n- No data available in FDI table G or J for ", MS, "\n"))
  }
} else {
  cat("\nOne or both of tables G and J are missing to perform the comparison on the number of vessels\n")
} # check_Table G and I
```

```{r FDI_cross_checks_AG, results = 'asis'}
if (check_tableA & check_tableG) {
  cat("\n## Consistency of landings and effort in tables A and G\n")
  cat("\nBoth FDI table A (Catch) and G (Effort) were analysed to summarise the mutual consistency of the reported
landings and effort data.\n")

  res <- suppressMessages(FDI_cross_checks_AG(data1 = tableA, data2 = tableG, verbose = FALSE))

  if (!is.null(res)) {
    if (nrow(res) > 0) {
      cat("\n\n\n")
      cat(paste0("\n_Summary table of the mutual inconsistency of tables A and G by country, GSA and year._\n"))
      print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No inconsistencies detected between FDI table A and G\n"))
    }
  } else {
    cat(paste0("\n\n\n- No data available in FDI table A or G for ", MS, "\n"))
  }
} else {
  cat("\nOne or both of tables A and G are missing to perform the comparison.\n")
} # check_Table G and I
```

```{r FDI_cross_checks_AH, results = 'asis'}
if (check_tableA & check_tableH) {
  cat("\n## Consistency of landings and spatial landings in table A and H\n")
  cat("\nBoth FDI table A (Catch) and H (Landing by rectangle) were analysed to summarise the mutual consistency
of the reported landings data.\n")

  res <- suppressMessages(FDI_cross_checks_AH(data1 = tableA, data2 = tableH, verbose = FALSE))

  if (!is.null(res)) {
    if (!is.null(res[[1]])) {
      if (nrow(res[[1]]) > 0) {
        cat("\n\n\n")
        cat(paste0("\n_Summary table of inconsistencies detected between table A and table H. Data are reported by
GSA, year, vessel length, fishing technique and gear type._\n"))
        print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
        cat("\n\n\n")
      } else {
        cat(paste0("\n\n\n- No inconsistencies detected between FDI table A and H\n"))
      }
    } #  if (!is.null(res[[1]]))

    if (!is.null(res[[2]])) {
      if (nrow(res[[2]]) > 0) {
        cat("\n\n\n")
        cat(paste0("\n_Summary table of total landings of table A and total spatial landings in table H by year
anc GSA_\n"))
        print(res[[2]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
        cat("\n\n\n")
      } else {
        cat(paste0("\n\n\n- No data available in FDI table A or H\n"))
```

```
      }
    } #  if (!is.null(res[[1]]))
  } else {
    cat(paste0("\n\n\n- No data available in FDI table A or H for ", MS, "\n"))
  }
} else {
  cat("\nOne or both of tables A and H are missing to perform the comparison on landings data\n")
} # check_Table G and I
```

```{r FDI_cross_checks_IG, results = 'asis'}
if (check_tableI & check_tableG) {
  cat("\n## Consistency of effort and spatial effort in tables G and I\n")
  cat("\nBoth FDI table G (Effort) and I (Effort by rectangle) were analysed to summarise the mutual consistency
of the reported effort data.\n")

  res <- suppressMessages(FDI_cross_checks_IG(data1 = tableI, data2 = tableG, verbose = FALSE))

  if (!is.null(res)) {
    if (!is.null(res[[1]])) {
      if (nrow(res[[1]]) > 0) {
        cat("\n\n\n")
        cat(paste0("\n_Summary table of inconsistencies detected between table G and table I. Data are reported by
GSA, year, vessel length, fishing technique and gear type._\n"))
        print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
        cat("\n\n\n")
      } else {
        cat(paste0("\n\n\n- No inconsistencies detected between FDI table G and I\n"))
      }
    } #  if (!is.null(res[[1]]))

    if (!is.null(res[[2]])) {
      if (nrow(res[[2]]) > 0) {
        cat("\n\n\n")
        cat(paste0("\n_Summary table of total effort of table G and total spatial effort in table I by year and
GSA_\n"))
        print(res[[2]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
        cat("\n\n\n")
      } else {
        cat(paste0("\n\n\n- No data available in FDI table G or I\n"))
      }
    } #  if (!is.null(res[[1]]))
  } else {
    cat(paste0("\n\n\n- No data available in FDI table G or I for ", MS, "\n"))
  }
} else {
  cat("\nOne or both of tables G and I are missing to perform the comparison on landings data\n")
} # check_Table G and I
```

8. APPENDIX IV – GFCM Rmd script

````
---
title: "| ![](logo.png){width=6in} \\vspace{1in} \n\n\n\n`r format(params$term)` Report\n\n\n\n
  \\vspace{0.1in} "

date: "Compiled on `r format(Sys.time(), '%d/%m/%Y, %H:%M')`"
subtitle: ''

output:
  html_document:
    template: RDBFIS_report_template.html
    toc: true
    toc-title: "Table of contents"
    toc_depth: 5
    number_sections: true
    collapsed: true
    df_print: paged
params:
  term: GFCM datacall
---
<!-- INITIALIZATION -->

```{r initialization, include=FALSE}
knitr::opts_chunk$set(
  echo = FALSE,
  message = FALSE,
  warning = FALSE
)

# loading needed libraries
lib  <-  c("RDBqc",  "knitr",  "kableExtra",  "dplyr",  "ggplot2",  "rworldmap",  "sp",  "rworldxtra",  "pander",
"data.table", "grDevices", "magrittr", "tictoc", "tidyverse", "fishmethods", "tidyr", "gridExtra", "outliers")
lapply(lib, require, character.only = TRUE)
```

<!-- USER DEFINED SOURCE FILES -->

```{r user_setup, include=FALSE}

# reading files
# UNCOMMENT  THE FOLLOWING LINES FOR STAND-ALONE USE
# SELECT THE APPROPRIATE FILE PATHS ON THE LOCAL FOLDERS

T_ii2 <- read.csv("./dc_dcrf_task_ii2_catch.csv", sep = ";")
T_iii <- read.csv("./dc_dcrf_task_iii_incidental_catch.csv", sep = ";")
T_vii2 <- read.csv("./dc_dcrf_task_vii2_length_data.csv", sep = ";")
T_vii31 <- read.csv("./dc_dcrf_task_vii31_size_1st_matur.csv", sep = ";")
T_vii32 <- read.csv("./dc_dcrf_task_vii32_maturity_data.csv", sep = ";")

MS <- "ITA"
SPs <- c("HKE", "DPS")
GSAs <- c(99)
segments <- "COMBINED"
```

<!-- DO NOT MODIFY THE FOLLOWING CODE -->

```{r setup, include=FALSE}
# check the presence of user defined filter for country (MS)
if (exists("MS")) {
  if (length(MS) == 1 & is.na(MS[1])) {
    user_MS <- FALSE
  } else {
    user_MS <- TRUE
  }
} else {
  user_MS <- FALSE
}

# check the presence of user defined filter for sub-area (GSAs)
if (exists("GSAs")) {
  if (length(GSAs) == 1 & is.na(GSAs[1])) {
    user_GSA <- FALSE
  } else {
    user_GSA <- TRUE
  }
} else {
  user_GSA <- FALSE
}

# check the presence of user defined filter for species (SPs)
if (exists("SPs")) {
  if (length(SPs) == 1 & is.na(SPs[1])) {
    user_SP <- FALSE
````
````

```
  } else {
    user_SP <- TRUE
  }
} else {
  user_SP <- FALSE
}

# check the presence of user defined filter for segments
if (exists("Segments")) {
  if (length(Segments) == 1 & is.na(Segments[1])) {
    user_segments <- FALSE
  } else {
    user_segments <- TRUE
  }
} else {
  user_segments <- FALSE
}
```

```
\newpage
<!-- Task II.2 -->

```{r check_task_ii2, results = 'asis'}

if (exists("T_ii2")) {
  if (class(T_ii2) == "data.frame") {
    if (nrow(T_ii2) > 0) {
      check_task_ii2 <- TRUE
      cat(paste0("\n# GFCM Catch data per species (Table II.2)\n"))

      T_ii2 <- check_gfcm_header(T_ii2, task = "II.2")

      if (!user_SP) {
        SPs <- sort(unique(T_ii2$Species))
      }
      if (!user_MS) {
        MS <- unique(T_ii2$CPC)
      }
      if (!user_GSA) {
GSAs <- sort(unique(T_ii2$GSA))
      }
      if (!user_segments) {
        Segments <- sort(unique(T_ii2$Segment))
      }

      T_ii2 <- T_ii2[T_ii2$Species %in% SPs &
        T_ii2$CPC == MS &
        T_ii2$GSA %in% GSAs, ]
    }
  }
} else {
check_task_ii2 <- FALSE
  cat("\n No GFCM Catch data per species available (Table II.2)\n")
}
```

<!-- duplicated records - Task_II.2 -->

```{r task_ii2_duplicated_records, results = 'asis'}
if (check_task_ii2) {
  cat("\n## Presence of duplicated records\n")
  cat("\nData included in the GFCM Catch table II.2 were checked for the presence of duplicated records. The
result of the check is here reported:\n")

  res <- check_RD_taskII2(data = T_ii2, verbose = FALSE)

  if (!is.null(res) & length(res) > 0) {
    duplicates <- T_ii2[res, ]
    if (nrow(duplicates) > 0) {
      cat("\n", paste0("- _WARNING_ - ", nrow(duplicates), " replicated record/records in the data"), "\n")
      print(T_ii2[res, ] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
T_ii2 <- T_ii2[-res, ]
    } else {
cat("\n", paste0("- No replicated records detected in GFCM Catch table II.2"), "\n")
    }
  } else {
    cat("\n", paste0("- No data available "), "\n")
  }
  cat("\n*Warning*: eventual duplicated records will be removed from data for the following analysis \n")
} # check_task_ii2
```

<!-- empty fields - Task_II.2 -->

```{r task_ii2_empty_fields, results = 'asis'}
if (check_task_ii2) {
```
```

```
  cat("\n## Presence of empty records\n")
  cat("\nData included in the GFCM Catch table II.2 were checked to identify the presence of not allowed empty
data in the given table. In case empty fields are detected a table with the name of the field with the empty
records and the number of records is reported.\n")

res <- suppressMessages(check_EF_taskII2(data = T_ii2, verbose = FALSE))

if (!is.null(res)) {
    res <- res[[1]][which(res[[1]] > 0)]
    if (length(res) > 0) {
      res <- data.frame(Empty_records = res)
      cat("\n\n\n")
      cat(paste0("\n_Table of empty records in GFCM Catch table II.2_\n"))
      print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No empty records detected in GFCM Catch table II.2\n"))
    }
  } else {
    cat(paste0("\n\n\n- No data available in GFCM Catch tableII.2)\n"))
  }
} # check_task_ii2
```

<!-- coverage - Task_II.2 -->

```{r task_ii2_coverage, results = 'asis'}
if (check_task_ii2) {
  cat("\n## Coverage of GFCM Catch table\n")
  cat("\nData included in the GFCM Catch table II.2 were checked to report a summary table of the data coverage in
terms of number of records by year and segment. Plots of both total landing and discard are also provided.\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### GSA ", GSAs[g], "\n"))
    s <- 2
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))

      res <- suppressMessages(GFCM_cov_II2(data = T_ii2, MS = MS, SP = SPs[s], segment = Segments, GSA = GSAs[g],
verbose = FALSE))

      if (!is.null(res)) {
        if (nrow(res[[1]]) > 0) {
          cat("\n\n\n")
          cat(paste0("\n_Summary table of the number of records per segment in GFCM Catch table  II.2 for ",
SPs[s], " - ", MS, " - GSA ", GSAs[g], "_\n"))
          print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
          cat("\n\n\n")
        } else {
          cat(paste0("\n\n\n- No data available\n"))
        }
        if (!is.null(res[[3]])) {
          cat("\n\n\n")
          cat(paste0("\nPlot of total landing coverage for ", SPs[s], " - ", MS, " - GSA ", GSAs[g], "\n"))
suppressMessages(print(res[[3]]))
          cat("\n\n\n")
} else {
          cat(paste0("\n\n\n- No data available\n"))
        }

        if (!is.null(res[[4]])) {
          cat("\n\n\n")
          cat(paste0("\nPlot of total discard coverage for ", SPs[s], " - ", MS, " - GSA ", GSAs[g], "\n"))
suppressMessages(print(res[[4]]))
          cat("\n\n\n")
} else {
          cat(paste0("\n\n\n- No data available\n"))
        }
      } else {
        cat(paste0("\n\n\n- No data available in GFCM Catch tableII.2\n"))
      }
    }
  }
} # check_task_ii2
```

\newpage
<!-- Task III -->

```{r check_task_iii, results = 'asis'}

if (exists("T_iii")) {
  if (class(T_iii) == "data.frame") {
    if (nrow(T_iii) > 0) {
      check_task_iii <- TRUE
      cat(paste0("\n# GFCM Incidental catch of vulnerable species (Table III)\n"))
```

```
      T_iii <- check_gfcm_header(T_iii, task = "III")

      if (!user_SP) {
        SPs <- sort(unique(T_iii$Species))
      }
      if (!user_MS) {
        MS <- unique(T_iii$CPC)
      }
      if (!user_GSA) {
GSAs <- sort(unique(T_iii$GSA))
      }
      if (!user_segments) {
        Segments <- sort(unique(T_iii$Segment))
      }

      T_iii <- T_iii[T_iii$Species %in% SPs &
        T_iii$CPC == MS &
        T_iii$GSA %in% GSAs, ]
    }
  }
} else {
check_task_iii <- FALSE
  cat("\n No dat availablea for GFCM Incidental catch of vulnerable species (Table III)\n")
}
```

<!-- duplicated records - Task_III -->

```{r task_iii_duplicated_records, results = 'asis'}
if (check_task_iii) {
  cat("\n## Presence of duplicated records\n")
  cat("\nData included in the GFCM Incidental catch of vulnerable species (Table III) were checked for the presence of duplicated records. The result of the check is here reported:\n")

  res <- check_RD_taskIII(data = T_iii, verbose = FALSE)

  if (!is.null(res) & length(res) > 0) {
    duplicates <- T_iii[res, ]
    if (nrow(duplicates) > 0) {
      cat("\n", paste0("- _WARNING_ - ", nrow(duplicates), " replicated record/records in the data"), "\n")
      print(T_iii[res, ] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
T_iii <- T_iii[-res, ]
} else {
      cat("\n", paste0("- No replicated records detected in GFCM Incidental catch of vulnerable species (Table III)"), "\n")
    }
  } else {
    cat("\n", paste0("- No data available "), "\n")
  }
  cat("\n*Warming*: eventual duplicated records will be removed from data for the following analysis \n")
} # check_task_iii
```

<!-- empty fields - Task_III -->

```{r task_iii_empty_fields, results = 'asis'}
if (check_task_iii) {
  cat("\n## Presence of empty records\n")
  cat("\nData included in the GFCM Incidental catch of vulnerable species (Table III) were checked to identify the presence of not allowed empty data in the given table. In case empty fields are detected a table with the name of the field with the empty records and the number of records is reported.\n")

res <- suppressMessages(check_EF_taskIII(data = T_iii, verbose = FALSE))

if (!is.null(res)) {
    res <- res[[1]][which(res[[1]] > 0)]
    if (length(res) > 0) {
      res <- data.frame(Empty_records = res)
      cat("\n\n\n")
      cat(paste0("\n_Table of empty records in GFCM Incidental catch of vulnerable species (Table III)_\n"))
      print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No empty records detected in GFCM Incidental catch of vulnerable species (Table III)\n"))
    }
  } else {
    cat(paste0("\n\n\n- No data available in GFCM Incidental catch of vulnerable species (Table III)\n"))
  }
} # check_task_iii
```

<!-- coverage - Task_III -->

```{r task_iii_coverage, results = 'asis'}
if (check_task_iii) {
  cat("\n## Coverage of GFCM Incidental catch of vulnerable species (Table III)\n")
```

```
  cat("\nData included in the GFCM Incidental catch of vulnerable species (Table III) were checked to report a
summary table of the data coverage in terms of number of records by year and segment. Plots of both total landing
and discard are also provided.\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### GSA ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))

      res <- suppressMessages(GFCM_cov_task_iii(data = T_iii, MS = MS, SP = SPs[s], GSA = GSAs[g], verbose =
FALSE))

      if (!is.null(res)) {
        if (nrow(res[[1]]) > 0) {
          cat("\n\n\n")
          cat(paste0("\n_Summary table of the number of individuals caught per source and gear in GFCM Incidental
catch of vulnerable species (Table III) for ", SPs[s], " - ", MS, " - GSA ", GSAs[g], "_\n"))
          print(res[[1]] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
          cat("\n\n\n")
        } else {
          cat(paste0("\n\n\n- No inconsistencies were detected the number of individuals caught.\n"))
        }
        if (!is.null(res[[2]])) {
          cat("\n\n\n")
          cat(paste0("\nPlot of the number of individuals caught time series for ", SPs[s], " - ", MS, " - GSA ",
GSAs[g], "\n"))
suppressMessages(print(res[[2]]))
          cat("\n\n\n")
} else {
          cat(paste0("\n\n\n- No data available\n"))
        }
      } else {
        cat(paste0("\n\n\n- No data available in GFCM Incidental catch of vulnerable species (Table III)\n"))
      }
    }
  }
} # check_task_ii2
```

\newpage
<!-- Task VII.2 -->

```{r check_task_vii2, results = 'asis'}
if (exists("T_vii2")) {
  if (class(T_vii2) == "data.frame") {
    if (nrow(T_vii2) > 0) {
      check_task_vii2 <- TRUE
      cat(paste0("\n# GFCM Biological information (Table VII.2, Length data)\n"))

      T_vii2 <- check_gfcm_header(T_vii2, task = "VII.2")

      if (!user_SP) {
        SPs <- sort(unique(T_vii2$Species))
      }
      if (!user_MS) {
        MS <- unique(T_vii2$CPC)
      }
      if (!user_GSA) {
GSAs <- sort(unique(T_vii2$GSA))
      }

      T_vii2 <- T_vii2[T_vii2$Species %in% SPs &
        T_vii2$CPC == MS &
        T_vii2$GSA %in% GSAs, ]
    }
  }
} else {
check_task_vii2 <- FALSE
  cat("\n No data available for GFCM Biological information (Table VII.2, Length data)\n")
}
```

<!-- duplicated records - Task_VII.2 -->

```{r task_vii2_duplicated_records, results = 'asis'}
if (check_task_vii2) {
  cat("\n## Presence of duplicated records\n")
  cat("\nData included in the GFCM Biological information (Table VII.2, Length data) were checked for the presence
of duplicated records. The result of the check is here reported:\n")

  res <- check_RD_taskVII2(data = T_vii2, verbose = FALSE)

  if (!is.null(res)) {
    if (length(res) > 0) {
      duplicates <- T_vii2[res, ]
      if (nrow(duplicates) > 0) {
```

```
        cat("\n", paste0("- _WARNING_ - ", nrow(duplicates), " replicated record/records in the data"), "\n")
        print(T_vii2[res, ] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width =
TRUE))
T_vii2 <- T_vii2[-res, ]
} else {
        cat("\n", paste0("- No replicated records detected in GFCM Biological information (Table VII.2, Length
data)"), "\n")
      }
    } else {
      cat("\n", paste0("- No replicated records detected in GFCM Biological information (Table VII.2, Length
data)"), "\n")
    }
  } else {
    cat("\n", paste0("- No data available "), "\n")
  }
  cat("\n*Warming*: eventual duplicated records will be removed from data for the following analysis \n")
} # check_task_vii2
```

<!-- empty fields - Task_VII.2 -->

```{r task_vii2_empty_fields, results = 'asis'}
if (check_task_vii2) {
  cat("\n## Presence of empty records\n")
  cat("\nData included in the GFCM Biological information (Table VII.2, Length data) were checked to identify the
presence of not allowed empty data in the given table. In case empty fields are detected a table with the name of
the field with the empty records and the number of records is reported.\n")

res <- suppressMessages(check_EF_taskVII2(data = T_vii2, verbose = FALSE))

if (!is.null(res)) {
    res <- res[[1]][which(res[[1]] > 0)]
    if (length(res) > 0) {
      res <- data.frame(Empty_records = res)
      cat("\n\n\n")
      cat(paste0("\n_Table of empty records in GFCM Biological information (Table VII.2, Length data)_\n"))
      print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No empty records detected in GFCM Biological information (Table VII.2, Length data)
\n"))
    }
  } else {
    cat(paste0("\n\n\n- No data available in GFCM Biological information (Table VII.2, Length data)\n"))
  }
} # check_task_vii2
```

<!-- Lenght-Weight - Task_VII2 -->

```{r task_vii2_LW, results = 'asis',fig.height=9,fig.width=9}
if (check_task_vii2) {
  cat("\n## Consistency of Length-Weight relationship in GFCM Biological information (Table VII.2, Length
data)\n")
  cat("\nData included in the GFCM Biological information (Table VII.2, Length data) were checked to report the
plot of the length-weight relationship of the selected species by year.\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### GSA ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))

      res <- suppressMessages(check_lw_TaskVII.2(data = T_vii2, MS = MS, SP = SPs[s], GSA = GSAs[g], verbose =
FALSE))

      if (!is.null(res)) {
        cat("\n\n\n")
        cat(paste0("\nPlot of the length-weight relationship by year for ", SPs[s], " - ", MS, " - GSA ", GSAs[g],
"\n"))
suppressMessages(print(res))
        cat("\n\n\n")
} else {
        cat(paste0("\n\n\n- No data available in GFCM Biological information (Table VII.2, Length data)\n"))
      }
    }
  }
} # check_task_vii2
```

<!-- LFD - Task_VII2 -->

```{r task_vii2_LFD, results = 'asis',fig.height=9,fig.width=9}
if (check_task_vii2) {
  cat("\n## Consistency of Length Frequency Distribution in GFCM Biological information (Table VII.2, Length
data)\n")
```

```
  cat("\nData included in the GFCM Biological information (Table VII.2, Length data) were checked to report the
plot of the Length Frequency Distribution of the selected species by year.\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### GSA ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))


      res <- suppressMessages(check_ldf_TaskVII.2(data = T_vii2, MS = MS, SP = SPs[s], GSA = GSAs[g], verbose =
FALSE))

      if (!is.null(res)) {
        if (!is.null(res[[1]])) {
          cat("\n\n\n")
          cat(paste0("\nPlot of the Length Frequency Distribution by year and segment for ", SPs[s], " - ", MS, "
- GSA ", GSAs[g], " - Biological sumpling (BS).\n"))
suppressMessages(print(res[[1]]))
          cat("\n\n\n")
} else {
          cat(paste0("\n\n\n- No data available in GFCM Biological information (Table VII.2, Length data)\n"))
        }

        if (!is.null(res[[2]])) {
          cat("\n\n\n")
          cat(paste0("\nPlot of the Length Frequency Distribution  by year  for ", SPs[s], " - ", MS, " - GSA ",
GSAs[g], " - Surveys (SU).\n"))
suppressMessages(print(res[[2]]))
          cat("\n\n\n")
} else {
          cat(paste0("\n\n\n- No biological sampling (BU) data available in GFCM Biological information (Table
VII.2, Length data)\n"))
        }
      } else {
        cat(paste0("\n\n\n- No biological sampling (BU) data available in GFCM Biological information (Table
VII.2, Length data)\n"))
      }
    }
  }
} # check_task_vii2
```

```
\newpage
<!-- Task VII.3.1 -->

```{r check_task_vii31, results = 'asis'}

if (exists("T_vii31")) {
  if (class(T_vii31) == "data.frame") {
    if (nrow(T_vii31) > 0) {
      check_task_vii31 <- TRUE
      cat(paste0("\n# GFCM Size at first maturity (Table VII.3.1)\n"))

      T_vii31 <- check_gfcm_header(T_vii31, task = "VII.3.1")

      if (!user_SP) {
        SPs <- sort(unique(T_vii31$Species))
      }
      if (!user_MS) {
        MS <- unique(T_vii31$CPC)
      }
      if (!user_GSA) {
GSAs <- sort(unique(T_vii31$GSA))
      }

      T_vii31 <- T_vii31[T_vii31$Species %in% SPs &
        T_vii31$CPC == MS &
        T_vii31$GSA %in% GSAs, ]
    }
  }
} else {
check_task_vii31 <- FALSE
  cat("\n No data available for GFCM Size at first maturity (Table VII.3.1)\n")
}
```

```
<!-- duplicated records - Task_VII.2 -->

```{r task_vii31_duplicated_records, results = 'asis'}
if (check_task_vii31) {
  cat("\n## Presence of duplicated records\n")
  cat("\nData included in the GFCM Size at first maturity (Table VII.3.1) were checked for the presence of
duplicated records. The result of the check is here reported:\n")

  res <- check_RD_taskVII31(data = T_vii31, verbose = FALSE)
```

```
  if (!is.null(res)) {
    if (length(res) > 0) {
      duplicates <- T_vii31[res, ]
      if (nrow(duplicates) > 0) {
        cat("\n", paste0("- _WARNING_ - ", nrow(duplicates), " replicated record/records in the data"), "\n")
        print(T_vii31[res, ] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width =
TRUE))
T_vii31 <- T_vii31[-res, ]
} else {
        cat("\n", paste0("- No replicated records detected in GFCM Size at first maturity (Table VII.3.1)"), "\n")
      }
    } else {
      cat("\n", paste0("- No replicated records detected in GFCM Size at first maturity (Table VII.3.1)"), "\n")
    }
  } else {
    cat("\n", paste0("- No data available "), "\n")
  }
  cat("\n*Warming*: eventual duplicated records will be removed from data for the following analysis \n")
} # check_task_vii2
```

<!-- empty fields - Task_VII.3.1 -->

```{r task_vii31_empty_fields, results = 'asis'}
if (check_task_vii31) {
  cat("\n## Presence of empty records\n")
  cat("\nData included in the GFCM Size at first maturity (Table VII.3.1) were checked to identify the presence of
not allowed empty data in the given table. In case empty fields are detected a table with the name of the field
with the empty records and the number of records is reported.\n")

res <- suppressMessages(check_EF_TaskVII31(data = T_vii31, verbose = FALSE))

if (!is.null(res)) {
    res <- res[[1]][which(res[[1]] > 0)]
    if (length(res) > 0) {
      res <- data.frame(Empty_records = res)
      cat("\n\n\n")
      cat(paste0("\n_Table of empty records in GFCM Size at first maturity (Table VII.3.1)\n"))
      print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No empty records detected in GFCM Size at first maturity (Table VII.3.1) \n"))
    }
  } else {
    cat(paste0("\n\n\n- No data available in GFCM Size at first maturity (Table VII.3.1)\n"))
  }
} # check_task_vii2
```

<!-- L50 - Task_VII31 -->

```{r task_vii31_L50, results = 'asis'}
if (check_task_vii31) {
  cat("\n## Consistency of size at first maturity in GFCM Table VII.3.1\n")
  cat("\nData included in the GFCM Size at first maturity (Table VII.3.1) were checked to report the plot of the
Size at first maturity time series of the selected species by sex\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### GSA ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))
      res <- suppressMessages(check_l50_TaskVII.3.1(data = T_vii31, MS = MS, SP = SPs[s], GSA = GSAs[g], verbose =
FALSE))

      if (!is.null(res)) {
        cat("\n\n\n")
        cat(paste0("\nPlot of the size at first maturity by sex for ", SPs[s], " - ", MS, " - GSA ", GSAs[g],
"\n"))
suppressMessages(print(res))
        cat("\n\n\n")
} else {
        cat(paste0("\n\n\n- No data available in GFCM Size at first maturity (Table VII.3.1)\n"))
      }
    }
  }
} # check_task_vii31
```

\newpage
<!-- Task VII.3.2 -->

```{r check_task_vii32, results = 'asis'}

if (exists("T_vii32")) {
  if (class(T_vii32) == "data.frame") {
    if (nrow(T_vii32) > 0) {
```

```
      check_task_vii32 <- TRUE
      cat(paste0("\n# GFCM Biological information (Table VII.3.2, Maturity data)\n"))

      T_vii32 <- check_gfcm_header(T_vii32, task = "VII.3.2")

      if (!user_SP) {
        SPs <- sort(unique(T_vii32$Species))
      }
      if (!user_MS) {
        MS <- unique(T_vii32$CPC)
      }
      if (!user_GSA) {
GSAs <- sort(unique(T_vii32$GSA))
      }

      T_vii32 <- T_vii32[T_vii32$Species %in% SPs &
        T_vii32$CPC == MS &
        T_vii32$GSA %in% GSAs, ]
    }
  }
} else {
check_task_vii32 <- FALSE
  cat("\n No data available for GFCM Biological information (Table VII.3.2, Maturity data)\n")
}
```

<!-- duplicated records - Task_VII.3.2 -->

```{r task_vii32_duplicated_records, results = 'asis'}
if (check_task_vii32) {
  cat("\n## Presence of duplicated records\n")
  cat("\nData included in the GFCM Biological information (Table VII.3.2, Maturity data) were checked for the
presence of duplicated records. The result of the check is here reported:\n")

  res <- check_RD_TaskVII32(data = T_vii32, verbose = FALSE)

  if (!is.null(res)) {
    if (length(res) > 0) {
      duplicates <- T_vii32[res, ]
      if (nrow(duplicates) > 0) {
        cat("\n", paste0("- _WARNING_ - ", nrow(duplicates), " replicated record/records in the data"), "\n")
        print(T_vii32[res, ] %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width =
TRUE))
T_vii32 <- T_vii32[-res, ]
} else {
        cat("\n", paste0("- No replicated records detected in GFCM Biological information (Table VII.3.2, Maturity
data)"), "\n")
      }
    } else {
      cat("\n", paste0("- No replicated records detected in GFCM Biological information (Table VII.3.2, Maturity
data)"), "\n")
    }
  } else {
    cat("\n", paste0("- No data available "), "\n")
  }
  cat("\n*Warming*: eventual duplicated records will be removed from data for the following analysis \n")
} # check_task_vii32
```

<!-- empty fields - Task_VII.3.2 -->

```{r task_vii32_empty_fields, results = 'asis'}
if (check_task_vii32) {
  cat("\n## Presence of empty records\n")
  cat("\nData included in the GFCM Biological information (Table VII.3.2, Maturity data) were checked to identify
the presence of not allowed empty data in the given table. In case empty data are detected a table with the name
of the field with the empty records and the number of records is reported.\n")

res <- suppressMessages(check_EF_TaskVII32(data = T_vii32, verbose = FALSE))

if (!is.null(res)) {
    res <- res[[1]][which(res[[1]] > 0)]
    if (length(res) > 0) {
      res <- data.frame(Empty_records = res)
      cat("\n\n\n")
      cat(paste0("\n_Table of empty records in GFCM Biological information (Table VII.3.2, Maturity data)_\n"))
      print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
      cat("\n\n\n")
    } else {
      cat(paste0("\n\n\n- No empty records detected in GFCM Biological information (Table VII.3.2, Maturity
data)\n"))
    }
  } else {
    cat(paste0("\n\n\n- No data available in GFCM Biological information (Table VII.3.2, Maturity data)\n"))
  }
} # check_task_vii32
```

```
<!-- CatFau - Task_7.3.2 -->

```{r task_vii32_catfau, results = 'asis',fig.height=9,fig.width=9}
if (check_task_vii32) {
  cat("\n## Consistency of sex and maturity stages in GFCM Biological information (Table VII.3.2, Maturity data)\n")
  cat("\nData included in the GFCM Biological information (Table VII.3.2, Maturity data) were checked to identify the possible incorrect codifications of faunistic category according to species and sex.\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### GSA ", GSAs[g], "\n"))
    s <- 2
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))

      res <- suppressMessages(check_species_catfau_TaskVII.3.2(data = T_vii32, species = catfau_check, matsex = sex_mat, MS = MS, SP = SPs[s], GSA = GSAs[g], verbose = FALSE))

      if (!is.null(res)) {
        if (length(res[[1]]) > 0) {
          cat("\n\n\n")
          cat(paste0("\n_Summary table of mismatches between species and faunistic category for ", SPs[s], " - ", MS, " - GSA ", GSAs[g], "_\n"))
          tab <- as.data.frame(res[[1]])
          colnames(tab) <- "unexpected_codes"
          print(tab %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width = TRUE))
          cat("\n\n\n")
        } else {
          cat(paste0("\n\n\n- No mismatches between species and faunistic category were detected in ", SPs[s], " - ", MS, " - GSA ", GSAs[g], ".\n"))
        }
        if (!is.null(res[[2]])) {
          cat("\n\n\n")
          cat(paste0("\nPlot of the length distribution by sex and maturity for ", SPs[s], " - ", MS, " - GSA ", GSAs[g], "\n"))
suppressMessages(print(res[[2]]))
          cat("\n\n\n")
} else {
          cat(paste0("\n\n\n- No maturity data available\n"))
        }
      } else {
        cat(paste0("\n\n\n- No data available in GFCM Biological information (Table VII.3.2, Maturity data)\n"))
      }
    }
  }
} # check_task_vii32
```

<!-- Lenght-Weight - Task_VII32 -->

```{r task_vii32_LW, results = 'asis',fig.height=6,fig.width=9}
if (check_task_vii32) {
  cat("\n## Consistency of Length-Weight relationship in GFCM Biological information (Table VII.3.2, Maturity data)\n")
  cat("\nData included in the GFCM Biological information (Table VII.3.2, Maturity data) were checked to report the plot of the length-weight relationship of the selected species by year.\n")

  for (g in 1:length(GSAs)) {
    cat(paste0("\n### GSA ", GSAs[g], "\n"))
    s <- 1
    for (s in 1:length(SPs)) {
      cat(paste0("\n#### ", SPs[s], "\n"))

      res <- suppressMessages(check_lw_TaskVII.3.2(data = T_vii32, MS = MS, SP = SPs[s], GSA = GSAs[g], verbose = FALSE))

      if (!is.null(res)) {
        cat("\n\n\n")
        cat(paste0("\nPlot of the length-weight relationship by year for ", SPs[s], " - ", MS, " - GSA ", GSAs[g], "\n"))
suppressMessages(print(res))
        cat("\n\n\n")
} else {
        cat(paste0("\n\n\n- No data available in GFCM Biological information (Table VII.3.2, Maturity data)\n"))
      }
    }
  }
} # check_task_vii2
```
```