Annex: Plumber R API Source Code

Source Code

Source File - DcrfTaskII2Report.Rmd

```
---
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
  task_ii2: !r data.frame(matrix(ncol = 0, nrow = 0))
---
<!-- 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

T_ii2 <- params$task_ii2

# task_ii2 <- read.csv("D:/OneDrive - Coispa Tecnologia & Ricerca S.C.A.R.L/RDB/data/GFCM-DCRF-Tasks_2018-
2019/GFCM-DCRF-Tasks_2018-2019/task_ii2.csv", sep=";") #
# task_ii2 <- task_ii2[task_ii2$GSA =="18",]
# task_ii2$GSA <- "99"

# MS <- "ITA"
# SPs = c("HKE","DPS") # "COMBINED"
# GSAs <- c("18")
# segments = "COMBINED"
```

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

```{r setup, include=FALSE}
# check the presence of user defined filter for country (MS)
user_MS <- FALSE

user_GSA <- FALSE

user_SP <- FALSE

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 Catch)\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 (Table II.2 Catch) available \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 (Table II.2 Catch) 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 (Table II.2 Catch)"), "\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 (Table II.2 Catch) 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 (Table II.2 Catch)_\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 (Table II.2 Catch)\n"))
  }
  } else {
    cat(paste0("\n\n\n- No data available in GFCM Catch table (Table II.2 Catch)\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 (Table II.2 Catch) 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")

g=1
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 (Table II.2
Catch)  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 table (Table II.2 Catch)\n"))
  }
      }
}
} # check_task_ii2
```

Source File - DcrfTaskIIIReport.Rmd

```
---
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: GFCM datacall
  task_iii: !r data.frame(matrix(ncol = 0, nrow = 0))
---
<!-- 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


T_iii <- params$task_iii
# task_iii <- read.csv("D:/OneDrive - Coispa Tecnologia & Ricerca S.C.A.R.L/RDB/data/GFCM-DCRF-Tasks_2018-
2019/GFCM-DCRF-Tasks_2018-2019/task_iii.csv", sep=";")
# task_iii <- task_iii[task_iii$GSA =="18",]
# task_iii$GSA <- "99"

```

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

```{r setup, include=FALSE}
# check the presence of user defined filter for country (MS)
user_MS <- FALSE

user_GSA <- FALSE

user_SP <- FALSE

user_segments <- FALSE

```

\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")

g=1
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
```

Source File - DcrfTaskVII2Report.Rmd

````
---
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
  task_vii2: !r data.frame(matrix(ncol = 0, nrow = 0))
---
<!-- 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

T_vii2 <- params$task_vii2
# task_vii2 <- read.csv("D:/OneDrive - Coispa Tecnologia & Ricerca S.C.A.R.L/RDB/data/GFCM-DCRF-Tasks_2018-
2019/GFCM-DCRF-Tasks_2018-2019/task_vii2.csv", sep=";")
# task_vii2 <- task_vii2[task_vii2$GSA =="18",]
# task_vii2$GSA <- "99"


# MS <- "ITA"
# SPs = c("HKE","DPS") # "COMBINED"
# GSAs <- c("18")
# segments = "COMBINED"
```

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

```{r setup, include=FALSE}
# check the presence of user defined filter for country (MS)
user_MS <- FALSE

user_GSA <- FALSE

user_SP <- FALSE

user_segments <- FALSE

```
````

````
\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")

g=1
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
```

Source File - DcrfTaskVII2Report.Rmd

```
<!-- 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")

g=1
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
```
```

Source File - DcrfTaskVII31Report.Rmd

````
---
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
  task_vii31: !r data.frame(matrix(ncol = 0, nrow = 0))
---
<!-- 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

T_vii31 <- params$task_vii31
# task_vii31 <- read.csv("D:/OneDrive - Coispa Tecnologia & Ricerca S.C.A.R.L/RDB/data/GFCM-DCRF-Tasks_2018
-2019/GFCM-DCRF-Tasks_2018-2019/task_vii31.csv", sep=";")
# task_vii31 <- task_vii31[task_vii31$GSA =="18",]
# task_vii31$GSA <- "99"


# MS <- "ITA"
# SPs = c("HKE","DPS") # "COMBINED"
# GSAs <- c("18")
# segments = "COMBINED"
```

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

```{r setup, include=FALSE}
# check the presence of user defined filter for country (MS)

user_MS <- FALSE

user_GSA <- FALSE

user_SP <- FALSE

user_segments <- FALSE
````
````

````
```

\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 Size at first maturity (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")

g=1
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
```

Source File - DcrfTaskVII32Report.Rmd

````
---
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
  task_vii32: !r data.frame(matrix(ncol = 0, nrow = 0))
---
<!-- 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

T_vii32 <- params$task_vii32
# task_vii32 <- read.csv("D:/OneDrive - Coispa Tecnologia & Ricerca S.C.A.R.L/RDB/data/GFCM-DCRF-Tasks_2018
-2019/GFCM-DCRF-Tasks_2018-2019/task_vii32.csv", sep=";")
# task_vii32 <- task_vii32[task_vii32$GSA =="18",]
# task_vii32$GSA <- "99"

# MS <- "ITA"
# SPs = c("HKE","DPS") # "COMBINED"
# GSAs <- c("18")
# segments = "COMBINED"
```

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

```{r setup, include=FALSE}
# check the presence of user defined filter for country (MS)

user_MS <- FALSE

user_GSA <- FALSE

user_SP <- FALSE

user_segments <- FALSE


```
````

````
\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")

g=1
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")

g=1
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
```

Source File - FdiACatchReport.Rmd

```
---
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
  CATCH: !r data.frame(matrix(ncol = 0, nrow = 0))
---
<!-- 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)

tableA <- params$CATCH
```


\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"))



      SPs <- sort(unique(tableA$species))

      MS  <- unique(tableA$country)

      GSAs <- sort(unique(tableA$sub_region))

      vessel_len <- sort(unique(tableA$vessel_length))

      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 emty 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 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_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 of landing and discard data in FDI catch table\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")


g=1
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 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")
```

```
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 <- 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")


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 <- 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")


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 <- 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 total landing 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
```

Source File - FdiGEffortReport.Rmd

````
---
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
  EFFORT: !r data.frame(matrix(ncol = 0, nrow = 0))
---
<!-- 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)

tableG <- params$EFFORT
```



\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"))

      MS  <- unique(tableG$country)

      GSAs <- sort(unique(tableG$sub_region))

      fishtech <- sort(unique(tableG$fishing_tech))

      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 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_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")


g=1
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
```

Source File - FdiHSpatialLandReport.Rmd

```
---
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
  SPATIAL: !r data.frame(matrix(ncol = 0, nrow = 0))
---
<!-- 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)

tableH <- params$SPATIAL
```


\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"))

      SPs <- sort(unique(tableH$species))

      MS  <- unique(tableH$country)

      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 fiels\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 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_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
```

Source File - FdiHSpatialLandReport.Rmd

```
```

Source File - FdiISpatialFeReport.Rmd

```
---
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
  SPATIALFE: !r data.frame(matrix(ncol = 0, nrow = 0))
---
<!-- 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)

tableI <- params$SPATIALFE
```


\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"))

      MS  <- unique(tableI$country)

      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 fiels\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 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_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
```

Source File - FdiJCapacityReport.Rmd

````
---
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
  CAPACITY: !r data.frame(matrix(ncol = 0, nrow = 0))
---
<!-- 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)

tableJ <- params$CAPACITY
```

\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"))

      MS  <- unique(tableJ$country)

      GSAs <- sort(unique(tableJ$principal_sub_region))

      vessel_len <- sort(unique(tableJ$vessel_length))

      fishtech <- sort(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 fiels\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 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_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 Capacirt 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")


g=1
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_tableJ
```

```{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_tableJ
```

Source File - MedBsAlkReport.Rmd

````
---
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
  ALK: !r data.frame(matrix(ncol = 0, nrow = 0))
---
<!-- 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 STAN-ALONE USE
# SELECT THE APPROPRIATE FILE PATHS ON THE LOCAL FOLDERS

# SOP_threshold <- 5
# Rt <- 1
#
# Catch = Catch_tab_example

ALK <- params$ALK
# Land = Landing_tab_example
# Disc = Discard_tab_example
# ML = ML_tab_example
# MA = MA_tab_example
# SL = SL_tab_example
# SA = SA_tab_example
# GP = GP_tab_example
# ALK = ALK_tab_example

# SPs <- c("HKE")
# MS <- "ITA"
# GSAs <- c("GSA 18","GSA 19")

```


```{r filters, include=FALSE}
````
````

Source File - MedBsAlkReport.Rmd

````
user_SP <- FALSE
user_MS <- FALSE
user_GSA <- FALSE

SOP_threshold <- 5

Rt <- 1


```

\newpage
<!-- 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")
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_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_ALK
```
````
````

Source File - medbscatch.Rmd

````
---
title: "Quality checks DGMARE_MedBS format"
author: "Bitetto Isabella - COISPA, Ioannis Chamodrakas - NKUA"

output: pdf_document
params:
  name_file_Catch: NA
  A_Catch: !r data.frame(matrix(ncol = 0, nrow = 0))

---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
# Quality checks on DGMARE_MedBS datacall formats

## Coverage

### Catch

```{r,warning=FALSE,include=TRUE,echo=FALSE}
library(ggplot2)

# name_file_Catch=  "Catch_DPS9"


#########################################################
# params$A_Catch= read.table(paste(name_file_Catch,".csv",sep=""),sep=",",header=T)

Catch = paste("Checking the file ", params$name_file_Catch,".csv",sep="")

threshold = 5

```
`r Catch`


```{r,warning=FALSE,include=TRUE,echo=FALSE}

Summary_land_wt=aggregate(params$A_Catch[,2:13]$LANDINGS,by=list(params$A_Catch$COUNTRY,
params$A_Catch$YEAR, params$A_Catch$QUARTER, params$A_Catch$VESSEL_LENGTH, params$A_Catch$GEAR,
params$A_Catch$MESH_SIZE_RANGE, params$A_Catch$FISHERY,  params$A_Catch$AREA,params$A_Catch$SPECIES),FUN="
sum")
colnames(Summary_land_wt)=c("COUNTRY", "YEAR", "QUARTER", "VESSEL_LENGTH", "GEAR", "MESH_SIZE_RANGE", "
FISHERY",  "AREA","SPECIES",  "LANDINGS" )

knitr::kable(
  Summary_land_wt[1:nrow(Summary_land_wt),1:ncol(Summary_land_wt)], caption = 'Table 1 - Landing at age at
coverage.'
)



Summary_disc_wt=aggregate(params$A_Catch[,2:13]$DISCARDS,by=list(params$A_Catch$COUNTRY,
params$A_Catch$YEAR, params$A_Catch$QUARTER, params$A_Catch$VESSEL_LENGTH, params$A_Catch$GEAR,
params$A_Catch$MESH_SIZE_RANGE, params$A_Catch$FISHERY,  params$A_Catch$AREA,params$A_Catch$SPECIES),FUN="
sum")
colnames(Summary_disc_wt)=c("COUNTRY", "YEAR", "QUARTER", "VESSEL_LENGTH", "GEAR", "MESH_SIZE_RANGE", "
FISHERY",  "AREA","SPECIES",  "DISCARDS" )

Summary_disc_wt= Summary_disc_wt[-which(round(Summary_disc_wt$DISCARDS,2)==-1),]

knitr::kable(
  Summary_disc_wt[1:nrow(Summary_disc_wt),1:ncol(Summary_disc_wt)], caption = 'Discard at age at coverage.'
)



```
````

```
## Sum of products


```{r,warning=FALSE,include=TRUE,echo=FALSE}

#-----------CHECK CONSISTENCY LANDING-----------------------------------------
Data_call=params$A_Catch

Landing_nb = Data_call[,1:12]
for (i in 1:20){
Landing_nb = cbind(Landing_nb,Data_call[,colnames(Data_call)==paste("AGE_",i-1,"_NO_LANDED",sep="")])
}
Landing_nb = cbind(Landing_nb,Data_call[,colnames(Data_call)=="AGE_20_NO_LANDED"])

colnames(Landing_nb)=c(colnames(Data_call[1:12]),paste("AGE_",c(0:19),"_NO_LANDED",sep=""),"
AGE_20_NO_LANDED")

Landing_nb$Sum= rowSums(Landing_nb[,13:ncol(Landing_nb)])

pos_indices=which(Landing_nb$Sum>0)


# matrice dei pesi medi
Landing_wt = Data_call[,1:12]
for (i in 1:20){
Landing_wt = cbind(Landing_wt,Data_call[,colnames(Data_call)==paste("AGE_",i-1,"_MEAN_WEIGHT_LANDED",sep=""
)])
}
Landing_wt = cbind(Landing_wt,Data_call[,colnames(Data_call)=="AGE_20_MEAN_WEIGHT_LANDED"])

colnames(Landing_wt)=c(colnames(Data_call[1:12]),paste("AGE_",c(0:19),"_MEAN_WEIGHT_LANDED",sep=""),"
AGE_20_MEAN_WEIGHT_LANDED")

Data_call$check_LANDING=""
Data_call$check_DISCARD=""

for (r in pos_indices){

nb = Landing_nb[r,13:(ncol(Landing_nb)-1)]
wt = Landing_wt[r,13:ncol(Landing_wt)]
Prod =  sum(nb*wt)
percentage = (Prod - Landing_wt[r,12])/Landing_wt[r,12]*100
    if (abs(round(percentage,2)) > threshold)
    {
    Data_call$check_LANDING[r] = round(percentage,2)
    }

}


#-----------CHECK CONSISTENCY DISCARD-----------------------------------------

# matrice del discard in numero
Discard_nb = Data_call[,c(1:11,13)]
for (i in 1:20){
Discard_nb = cbind(Discard_nb,Data_call[,colnames(Data_call)==paste("AGE_",i-1,"_NO_DISCARD",sep="")])
}
Discard_nb = cbind(Discard_nb,Data_call[,colnames(Data_call)=="AGE_20_NO_DISCARD"])

colnames(Discard_nb)=c(colnames(Data_call[c(1:11,13)]),paste("AGE_",c(0:19),"_NO_DISCARD",sep=""),"
AGE_20_NO_DISCARD")

Discard_nb$Sum= rowSums(Discard_nb[,13:ncol(Discard_nb)])

pos_indices=which(Discard_nb$Sum>0)

# matrice dei pesi medi
Discard_wt = Data_call[,c(1:11,13)]
for (i in 1:20){
```
```

```
Discard_wt = cbind(Discard_wt,Data_call[,colnames(Data_call)==paste("AGE_",i-1,"_MEAN_WEIGHT_DISCARD",sep=
"")])
}
Discard_wt = cbind(Discard_wt,Data_call[,colnames(Data_call)=="AGE_20_MEAN_WEIGHT_DISCARD"])
colnames(Discard_wt)=c(colnames(Data_call[c(1:11,13)]),paste("AGE_",c(0:19),"_MEAN_WEIGHT_DISCARD",sep=""),
"AGE_20_MEAN_WEIGHT_DISCARD")



for (r in pos_indices){
nb = Discard_nb[r,13:(ncol(Discard_nb)-1)]
wt = Discard_wt[r,13:ncol(Discard_wt)]
Prod =  sum(nb*wt)
percentage = ifelse((Discard_wt[r,12]==0)&(Prod==0),NA,(Prod - Discard_wt[r,12])/Discard_wt[r,12]*100)
    if (!is.na(percentage) & abs(round(percentage,2)) > threshold)
    {
    Data_call$check_DISCARD[r] = round(percentage,2)
    }

}

write.table(Data_call,paste(params$name_file_Catch,"_checked_sum_of_products.csv",sep=""),sep=";",row.names
=F)

error_land=Data_call[Data_call$check_LANDING>=threshold,]
error_disc=Data_call[Data_call$check_DISCARD>=threshold,]

```
Below the records with a difference in the landing in the sum of products >= of the `r threshold` % are
reported.

```{r,warning=FALSE,include=TRUE,echo=FALSE}
if (nrow(error_land)>0)
    {error_land[,c(1,(ncol(error_land)-1))]
} else {
  "No error occurred"
}
```
Below the records with a difference in the discard in the sum of products >= of the `r threshold` % are
reported.

```{r,warning=FALSE,include=TRUE,echo=FALSE}

if (nrow(error_disc)>0)
    {error_disc[,c(1,ncol(error_disc))]
} else {
  "No error occurred"
}

```
```
```

Source File - MedBsCatchReport.Rmd

````
---
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
  A_Catch: !r data.frame(matrix(ncol = 0, nrow = 0))
---
<!-- 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 STAN-ALONE USE
# SELECT THE APPROPRIATE FILE PATHS ON THE LOCAL FOLDERS

SOP_threshold <- 5
Rt <- 1
#
# Catch = Catch_tab_example

Catch <- params$A_Catch
# Land = Landing_tab_example
# Disc = Discard_tab_example
# ML = ML_tab_example
# MA = MA_tab_example
# SL = SL_tab_example
# SA = SA_tab_example
# GP = GP_tab_example
# ALK = ALK_tab_example

# SPs <- c("HKE")
# MS <- "ITA"
# GSAs <- c("GSA 18","GSA 19")

```


```{r filters, include=FALSE}
# check the presence of user defined filter for species (SPs)
````
````

```
user_SP <- FALSE

user_MS <- FALSE

user_GSA <- FALSE

SOP_threshold <- 5

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 <- unique(Catch$species)
      }
      if(!user_MS){
        MS  <- unique(Catch$country)
      }
      if(!user_GSA){
        GSAs <- 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)) {
          # cc <- Catch[Catch$AREA== GSAs[g]&Catch$SPECIES==SPs[s],]
          # Catch <- rbind(Catch,cc[1,])
        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")

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_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"))

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_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],"_"))
        print(res %>% kable("html") %>% kable_styling(font_size = 12) %>% kable_classic(full_width=TRUE) )
        }
      }
}

} # check_Catch

```

Source File - MedBsDiscardsReport.Rmd

````
---
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
  Discards: !r data.frame(matrix(ncol = 0, nrow = 0))
---
<!-- 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 STAN-ALONE USE
# SELECT THE APPROPRIATE FILE PATHS ON THE LOCAL FOLDERS

# SOP_threshold <- 5
# Rt <- 1
#
# Catch = Catch_tab_example

Disc <- params$Discards
# Land = Landing_tab_example
# Disc = Discard_tab_example
# ML = ML_tab_example
# MA = MA_tab_example
# SL = SL_tab_example
# SA = SA_tab_example
# GP = GP_tab_example
# ALK = ALK_tab_example

# SPs <- c("HKE")
# MS <- "ITA"
# GSAs <- c("GSA 18","GSA 19")

```


```{r filters, include=FALSE}
user_SP <- FALSE
````
````

````
user_MS <- FALSE

user_GSA <- FALSE

SOP_threshold <- 5

Rt <- 1

```

\newpage

<!-- 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,]

g=1
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")

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_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")

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(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"))

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_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")

g=1
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",]
                 # rownames(tab1) <- seq(1,nrow(tab1),1)
                 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## Checks 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")

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_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"))

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_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## Checks 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")

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_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## Checks 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")

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_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## Checks 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")

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 = 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"))

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_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"))

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_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
```

Source File - MedBsGpReport.Rmd

````
---
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
  GP: !r data.frame(matrix(ncol = 0, nrow = 0))
---
<!-- 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 STAN-ALONE USE
# SELECT THE APPROPRIATE FILE PATHS ON THE LOCAL FOLDERS

# SOP_threshold <- 5
# Rt <- 1
#
# Catch = Catch_tab_example

GP <- params$GP
# Land = Landing_tab_example
# Disc = Discard_tab_example
# ML = ML_tab_example
# MA = MA_tab_example
# SL = SL_tab_example
# SA = SA_tab_example
# GP = GP_tab_example
# ALK = ALK_tab_example

# SPs <- c("HKE")
# MS <- "ITA"
# GSAs <- c("GSA 18","GSA 19")

```


```{r filters, include=FALSE}
user_SP <- FALSE
````
````

````
user_MS <- FALSE

user_GSA <- FALSE

SOP_threshold <- 5

Rt <- 1

```

\newpage

<!-- 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")


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_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")

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_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
```

Source File - MedBsLandingsReport.Rmd

````
---
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
  Landings: !r data.frame(matrix(ncol = 0, nrow = 0))
---
<!-- 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 STAN-ALONE USE
# SELECT THE APPROPRIATE FILE PATHS ON THE LOCAL FOLDERS

# SOP_threshold <- 5
# Rt <- 1
#
# Catch = Catch_tab_example

Land <- params$Landings
# Land = Landing_tab_example
# Disc = Discard_tab_example
# ML = ML_tab_example
# MA = MA_tab_example
# SL = SL_tab_example
# SA = SA_tab_example
# GP = GP_tab_example
# ALK = ALK_tab_example

# SPs <- c("HKE")
# MS <- "ITA"
# GSAs <- c("GSA 18","GSA 19")

```


```{r filters, include=FALSE}
user_SP <- FALSE
````
````

````
user_MS <- FALSE

user_GSA <- FALSE

SOP_threshold <- 5

Rt <- 1

```

\newpage

<!-- 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,]

g=1
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=1
      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")

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_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")

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_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")

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_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",]
            # rownames(tab1) <- seq(1,nrow(tab1),1)
            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## Checks 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")

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_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"))

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_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## Checks 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")

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_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## Checks 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")

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_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## Checks 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"))

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_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
```
```

Source File - MedBsMaReport.Rmd

````
---
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
  MA: !r data.frame(matrix(ncol = 0, nrow = 0))
---
<!-- 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 STAN-ALONE USE
# SELECT THE APPROPRIATE FILE PATHS ON THE LOCAL FOLDERS

# SOP_threshold <- 5
# Rt <- 1
#
# Catch = Catch_tab_example

MA <- params$MA
# Land = Landing_tab_example
# Disc = Discard_tab_example
# ML = ML_tab_example
# MA = MA_tab_example
# SL = SL_tab_example
# SA = SA_tab_example
# GP = GP_tab_example
# ALK = ALK_tab_example

# SPs <- c("HKE")
# MS <- "ITA"
# GSAs <- c("GSA 18","GSA 19")

```


```{r filters, include=FALSE}
user_SP <- FALSE
````
````

````
user_MS <- FALSE

user_GSA <- FALSE

SOP_threshold <- 5

Rt <- 1

```

\newpage

<!-- 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
```

Source File - MedBsMlReport.Rmd

````
---
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
  ML: !r data.frame(matrix(ncol = 0, nrow = 0))
---
<!-- 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 STAN-ALONE USE
# SELECT THE APPROPRIATE FILE PATHS ON THE LOCAL FOLDERS

# SOP_threshold <- 5
# Rt <- 1
#
# Catch = Catch_tab_example

ML <- params$ML
# Land = Landing_tab_example
# Disc = Discard_tab_example
# ML = ML_tab_example
# MA = MA_tab_example
# SL = SL_tab_example
# SA = SA_tab_example
# GP = GP_tab_example
# ALK = ALK_tab_example

# SPs <- c("HKE")
# MS <- "ITA"
# GSAs <- c("GSA 18","GSA 19")

```


```{r filters, include=FALSE}
user_SP <- FALSE
````
````

````
user_MS <- FALSE

user_GSA <- FALSE

SOP_threshold <- 5

Rt <- 1

```

\newpage

<!-- 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")

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_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
```

Source File - MedBsSraReport.Rmd

````
---
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
  SRA: !r data.frame(matrix(ncol = 0, nrow = 0))
---
<!-- 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 STAN-ALONE USE
# SELECT THE APPROPRIATE FILE PATHS ON THE LOCAL FOLDERS

# SOP_threshold <- 5
# Rt <- 1
#
# Catch = Catch_tab_example

SA <- params$SRA
# Land = Landing_tab_example
# Disc = Discard_tab_example
# ML = ML_tab_example
# MA = MA_tab_example
# SL = SL_tab_example
# SA = SA_tab_example
# GP = GP_tab_example
# ALK = ALK_tab_example

# SPs <- c("HKE")
# MS <- "ITA"
# GSAs <- c("GSA 18","GSA 19")

```


```{r filters, include=FALSE}
user_SP <- FALSE
````
````

````
user_MS <- FALSE

user_GSA <- FALSE

SOP_threshold <- 5

Rt <- 1

```

\newpage

<!-- 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")
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_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
```
````
````

Source File - MedBsSrlReport.Rmd

````
---
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
  SRL: !r data.frame(matrix(ncol = 0, nrow = 0))
---
<!-- 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 STAN-ALONE USE
# SELECT THE APPROPRIATE FILE PATHS ON THE LOCAL FOLDERS

# SOP_threshold <- 5
# Rt <- 1
#
# Catch = Catch_tab_example

SL <- params$SRL
# Land = Landing_tab_example
# Disc = Discard_tab_example
# ML = ML_tab_example
# MA = MA_tab_example
# SL = SL_tab_example
# SA = SA_tab_example
# GP = GP_tab_example
# ALK = ALK_tab_example

# SPs <- c("HKE")
# MS <- "ITA"
# GSAs <- c("GSA 18","GSA 19")

```


```{r filters, include=FALSE}
# check the presence of user defined filter for species (SPs)
````
````

Source File - MedBsSrlReport.Rmd

````
user_SP <- FALSE

user_MS <- FALSE

user_GSA <- FALSE

SOP_threshold <- 5

Rt <- 1

```

\newpage

<!-- 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")

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_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
```
````
````

Source File - plumber.R

```
library(RoME)
library(zip)
library(COSTcore)
library(COSTdbe)
library(COSTeda)
library(RDBprocessing)

#* @get /mean
normalMean <- function(samples=10){
  data <- rnorm(samples)
  mean(data)
}

#* @post /sum
addTwo <- function(a, b){
  as.numeric(a) + as.numeric(b)
}

#* @param f:file
#* @serializer contentType list(type="application/html")
#* @post /medbscatch
function(f) {
  tmp <- tempfile()
  data <- rawToChar(f[[1]])
  text_con <- textConnection(data)
  dataframe <- read.table(text_con,sep=",",header=T)
  names(dataframe) <- tolower(names(dataframe))
  df_name <- names(f)[1]
  rmarkdown::render("MedBsCatchReport.Rmd",tmp,output_format = "html_document",params = list(A_Catch =
dataframe))
  tmpname <- paste(tmp, ".html", sep="")

  readBin(tmpname, "raw", n=file.info(tmpname)$size)
}

#* @param f:file
#* @serializer unboxedJSON
#* @post /medbscatchprocessing
function(f) {
  sampling_data <- rawToChar(f[[1]])
  text_con1 <- textConnection(sampling_data)
  CS <- read.csv(text_con1,sep=",",header=T)
  landings_data <- rawToChar(f[[2]])
  text_con2 <- textConnection(landings_data)
  CL <- read.csv(text_con2,sep=",",header=T)
  costce_data <- rawToChar(f[[3]])
  text_con3 <- textConnection(costce_data)
  CE <- read.csv(text_con3,sep=",",header=T)
  CL$species=CS$species[1]

  res <- CATCH_MEDBS(CS, CL, CE)
  names(res) <- tolower(names(res))
  res
}

#* @param f:file
#* @serializer contentType list(type="application/html")
#* @post /medbsalk
function(f) {
  tmp <- tempfile()
  data <- rawToChar(f[[1]])
  text_con <- textConnection(data)
  dataframe <- read.table(text_con,sep=",",header=T)
  names(dataframe) <- tolower(names(dataframe))
  df_name <- names(f)[1]
  rmarkdown::render("MedBsAlkReport.Rmd",tmp,output_format = "html_document",params = list(ALK = dataframe
))
  tmpname <- paste(tmp, ".html", sep="")

  readBin(tmpname, "raw", n=file.info(tmpname)$size)
```

```
}

#* @param f:file
#* @serializer unboxedJSON
#* @post /medbsalkprocessing
function(f) {
  sampling_data <- rawToChar(f[[1]])
  text_con1 <- textConnection(sampling_data)
  CS <- read.csv(text_con1,sep=",",header=T)
  res <- ALK_MEDBS(CS)
  names(res) <- tolower(names(res))
  res
}

#* @param f:file
#* @serializer contentType list(type="application/html")
#* @post /medbsdiscards
function(f) {
  tmp <- tempfile()
  data <- rawToChar(f[[1]])
  text_con <- textConnection(data)
  dataframe <- read.table(text_con,sep=",",header=T)
  names(dataframe) <- tolower(names(dataframe))
  df_name <- names(f)[1]
  rmarkdown::render("MedBsDiscardsReport.Rmd",tmp,output_format = "html_document",params = list(Discards =
dataframe))
  tmpname <- paste(tmp, ".html", sep="")

  readBin(tmpname, "raw", n=file.info(tmpname)$size)
}

#* @param f:file
#* @serializer unboxedJSON
#* @post /medbsdiscardsprocessing
function(f) {
  sampling_data <- rawToChar(f[[1]])
  text_con1 <- textConnection(sampling_data)
  CS <- read.csv(text_con1,sep=",",header=T)
  landings_data <- rawToChar(f[[2]])
  text_con2 <- textConnection(landings_data)
  CL <- read.csv(text_con2,sep=",",header=T)
  costce_data <- rawToChar(f[[3]])
  text_con3 <- textConnection(costce_data)
  CE <- read.csv(text_con3,sep=",",header=T)

  CL$species=CS$species[1]

  res <- DISC_MEDBS(CS, CL, CE)
  names(res) <- tolower(names(res))
  res
}

#* @param f:file
#* @serializer contentType list(type="application/html")
#* @post /medbslandings
function(f) {
  tmp <- tempfile()
  data <- rawToChar(f[[1]])
  text_con <- textConnection(data)
  dataframe <- read.table(text_con,sep=",",header=T)
  names(dataframe) <- tolower(names(dataframe))
  df_name <- names(f)[1]
  rmarkdown::render("MedBsLandingsReport.Rmd",tmp,output_format = "html_document",params = list(Landings =
dataframe))
  tmpname <- paste(tmp, ".html", sep="")

  readBin(tmpname, "raw", n=file.info(tmpname)$size)
}

#* @param f:file
```

Source File - plumber.R

```r
#* @serializer unboxedJSON
#* @post /medbslandingsprocessing
function(f) {
  sampling_data <- rawToChar(f[[1]])
  text_con1 <- textConnection(sampling_data)
  CS <- read.csv(text_con1,sep=",",header=T)
  landings_data <- rawToChar(f[[2]])
  text_con2 <- textConnection(landings_data)
  CL <- read.csv(text_con2,sep=",",header=T)

  CL$species=CS$species[1]

  res <- LAND_MEDBS(CS, CL)
  names(res) <- tolower(names(res))
  res
}

#* @param f:file
#* @serializer contentType list(type="application/html")
#* @post /medbsma
function(f) {
  tmp <- tempfile()
  data <- rawToChar(f[[1]])
  text_con <- textConnection(data)
  dataframe <- read.table(text_con,sep=",",header=T)
  names(dataframe) <- tolower(names(dataframe))
  df_name <- names(f)[1]
  rmarkdown::render("MedBsMaReport.Rmd",tmp,output_format = "html_document",params = list(MA = dataframe))
  tmpname <- paste(tmp, ".html", sep="")

  readBin(tmpname, "raw", n=file.info(tmpname)$size)
}

#* @param f:file
#* @serializer unboxedJSON
#* @post /medbsmaprocessing
function(f) {
  sampling_data <- rawToChar(f[[1]])
  text_con1 <- textConnection(sampling_data)
  CS <- read.csv(text_con1,sep=",",header=T)
  res <- MA_MEDBS(CS)
  names(res) <- tolower(names(res))
  res
}

#* @param f:file
#* @serializer contentType list(type="application/html")
#* @post /medbsml
function(f) {
  tmp <- tempfile()
  data <- rawToChar(f[[1]])
  text_con <- textConnection(data)
  dataframe <- read.table(text_con,sep=",",header=T)
  names(dataframe) <- tolower(names(dataframe))
  df_name <- names(f)[1]
  rmarkdown::render("MedBsMlReport.Rmd",tmp,output_format = "html_document",params = list(ML = dataframe))
  tmpname <- paste(tmp, ".html", sep="")

  readBin(tmpname, "raw", n=file.info(tmpname)$size)
}

#* @param f:file
#* @serializer unboxedJSON
#* @post /medbsmlprocessing
function(f) {
  sampling_data <- rawToChar(f[[1]])
  text_con1 <- textConnection(sampling_data)
  CS <- read.csv(text_con1,sep=",",header=T)
  res <- ML_MEDBS(CS)
  names(res) <- tolower(names(res))
```

```
  res
}

#* @param f:file
#* @serializer contentType list(type="application/html")
#* @post /medbssra
function(f) {
  tmp <- tempfile()
  data <- rawToChar(f[[1]])
  text_con <- textConnection(data)
  dataframe <- read.table(text_con,sep=",",header=T)
  names(dataframe) <- tolower(names(dataframe))
  df_name <- names(f)[1]
  rmarkdown::render("MedBsSraReport.Rmd",tmp,output_format = "html_document",params = list(SRA = dataframe
))
  tmpname <- paste(tmp, ".html", sep="")

  readBin(tmpname, "raw", n=file.info(tmpname)$size)
}

#* @param f:file
#* @serializer unboxedJSON
#* @post /medbssraprocessing
function(f) {
  sampling_data <- rawToChar(f[[1]])
  text_con1 <- textConnection(sampling_data)
  CS <- read.csv(text_con1,sep=",",header=T)
  res <- SRA_MEDBS(CS)
  names(res) <- tolower(names(res))
  res
}

#* @param f:file
#* @serializer contentType list(type="application/html")
#* @post /medbssrl
function(f) {
  tmp <- tempfile()
  data <- rawToChar(f[[1]])
  text_con <- textConnection(data)
  dataframe <- read.table(text_con,sep=",",header=T)
  names(dataframe) <- tolower(names(dataframe))
  df_name <- names(f)[1]
  rmarkdown::render("MedBsSrlReport.Rmd",tmp,output_format = "html_document",params = list(SRL = dataframe
))
  tmpname <- paste(tmp, ".html", sep="")

  readBin(tmpname, "raw", n=file.info(tmpname)$size)
}

#* @param f:file
#* @serializer unboxedJSON
#* @post /medbssrlprocessing
function(f) {
  sampling_data <- rawToChar(f[[1]])
  text_con1 <- textConnection(sampling_data)
  CS <- read.csv(text_con1,sep=",",header=T)
  res <- SRL_MEDBS(CS)
  names(res) <- tolower(names(res))
  res
}

#* @param f:file
#* @serializer contentType list(type="application/html")
#* @post /medbsgp
function(f) {
  tmp <- tempfile()
  data <- rawToChar(f[[1]])
  text_con <- textConnection(data)
  dataframe <- read.table(text_con,sep=",",header=T)
  names(dataframe) <- tolower(names(dataframe))
```

```
  df_name <- names(f)[1]
  rmarkdown::render("MedBsGpReport.Rmd",tmp,output_format = "html_document",params = list(GP = dataframe))
  tmpname <- paste(tmp, ".html", sep="")

  readBin(tmpname, "raw", n=file.info(tmpname)$size)
}

#* @param f:file
#* @serializer contentType list(type="application/html")
#* @post /fdiacatch
function(f) {
  tmp <- tempfile()
  data <- rawToChar(f[[1]])
  text_con <- textConnection(data)
  dataframe <- read.table(text_con,sep=",",header=T)
  names(dataframe) <- tolower(names(dataframe))
  df_name <- names(f)[1]
  rmarkdown::render("FdiACatchReport.Rmd",tmp,output_format = "html_document",params = list(CATCH =
dataframe))
  tmpname <- paste(tmp, ".html", sep="")

  readBin(tmpname, "raw", n=file.info(tmpname)$size)
}

#* @param f:file
#* @serializer unboxedJSON
#* @post /fdiacatchprocessing
function(f) {
  sampling_data <- rawToChar(f[[1]])
  text_con1 <- textConnection(sampling_data)
  CL <- read.csv(text_con1,sep=",",header=T)
  res <- CATCH_FDI(CL)
  names(res) <- tolower(names(res))
  res
}

#* @param f:file
#* @serializer contentType list(type="application/html")
#* @post /fdigeffort
function(f) {
  tmp <- tempfile()
  data <- rawToChar(f[[1]])
  text_con <- textConnection(data)
  dataframe <- read.table(text_con,sep=",",header=T)
  names(dataframe) <- tolower(names(dataframe))
  df_name <- names(f)[1]
  rmarkdown::render("FdiGEffortReport.Rmd",tmp,output_format = "html_document",params = list(EFFORT =
dataframe))
  tmpname <- paste(tmp, ".html", sep="")

  readBin(tmpname, "raw", n=file.info(tmpname)$size)
}

#* @param f:file
#* @serializer contentType list(type="application/html")
#* @post /fdihspatialland
function(f) {
  tmp <- tempfile()
  data <- rawToChar(f[[1]])
  text_con <- textConnection(data)
  dataframe <- read.table(text_con,sep=",",header=T)
  names(dataframe) <- tolower(names(dataframe))
  df_name <- names(f)[1]
  rmarkdown::render("FdiHSpatialLandReport.Rmd",tmp,output_format = "html_document",params = list(SPATIAL
 = dataframe))
  tmpname <- paste(tmp, ".html", sep="")

  readBin(tmpname, "raw", n=file.info(tmpname)$size)
}
```

```
#* @param f:file
#* @serializer contentType list(type="application/html")
#* @post /fdiispatialfe
function(f) {
  tmp <- tempfile()
  data <- rawToChar(f[[1]])
  text_con <- textConnection(data)
  dataframe <- read.table(text_con,sep=",",header=T)
  names(dataframe) <- tolower(names(dataframe))
  df_name <- names(f)[1]
  rmarkdown::render("FdiISpatialFeReport.Rmd",tmp,output_format = "html_document",params = list(SPATIALFE
 = dataframe))
  tmpname <- paste(tmp, ".html", sep="")

  readBin(tmpname, "raw", n=file.info(tmpname)$size)
}

#* @param f:file
#* @serializer contentType list(type="application/html")
#* @post /fdijcapacity
function(f) {
  tmp <- tempfile()
  data <- rawToChar(f[[1]])
  text_con <- textConnection(data)
  dataframe <- read.table(text_con,sep=",",header=T)
  names(dataframe) <- tolower(names(dataframe))
  df_name <- names(f)[1]
  rmarkdown::render("FdiJCapacityReport.Rmd",tmp,output_format = "html_document",params = list(CAPACITY =
dataframe))
  tmpname <- paste(tmp, ".html", sep="")

  readBin(tmpname, "raw", n=file.info(tmpname)$size)
}

#* @param f:file
#* @serializer contentType list(type="application/html")
#* @post /dcrftaskii2catch
function(f) {
  tmp <- tempfile()
  data <- rawToChar(f[[1]])
  text_con <- textConnection(data)
  dataframe <- read.table(text_con,sep=",",header=T)
  names(dataframe) <- tolower(names(dataframe))
  df_name <- names(f)[1]
  rmarkdown::render("DcrfTaskII2Report.Rmd",tmp,output_format = "html_document",params = list(task_ii2 =
dataframe))
  tmpname <- paste(tmp, ".html", sep="")

  readBin(tmpname, "raw", n=file.info(tmpname)$size)
}

#* @param f:file
#* @serializer unboxedJSON
#* @post /dcrftaskii2catchprocessing
function(f) {
  fdiacatch_data <- rawToChar(f[[1]])
  text_con1 <- textConnection(fdiacatch_data)
  FDIACATCH <- read.csv(text_con1,sep=",",header=T)
  res <- TabII2_GFCM(FDIACATCH)
  names(res) <- tolower(names(res))
  res
}

#* @param f:file
#* @serializer contentType list(type="application/html")
#* @post /dcrftaskiiiincidentalcatch
function(f) {
  tmp <- tempfile()
  data <- rawToChar(f[[1]])
  text_con <- textConnection(data)
```

```
  dataframe <- read.table(text_con,sep=",",header=T)
  names(dataframe) <- tolower(names(dataframe))
  df_name <- names(f)[1]
  rmarkdown::render("DcrfTaskIIIReport.Rmd",tmp,output_format = "html_document",params = list(task_iii =
dataframe))
  tmpname <- paste(tmp, ".html", sep="")

  readBin(tmpname, "raw", n=file.info(tmpname)$size)
}

#* @param f:file
#* @serializer contentType list(type="application/html")
#* @post /dcrftaskvii2lengthdata
function(f) {
  tmp <- tempfile()
  data <- rawToChar(f[[1]])
  text_con <- textConnection(data)
  dataframe <- read.table(text_con,sep=",",header=T)
  names(dataframe) <- tolower(names(dataframe))
  df_name <- names(f)[1]
  rmarkdown::render("DcrfTaskVII2Report.Rmd",tmp,output_format = "html_document",params = list(task_vii2 =
dataframe))
  tmpname <- paste(tmp, ".html", sep="")

  readBin(tmpname, "raw", n=file.info(tmpname)$size)
}

#* @param f:file
#* @serializer contentType list(type="application/html")
#* @post /dcrftaskvii31size1stmatur
function(f) {
  tmp <- tempfile()
  data <- rawToChar(f[[1]])
  text_con <- textConnection(data)
  dataframe <- read.table(text_con,sep=",",header=T)
  names(dataframe) <- tolower(names(dataframe))
  df_name <- names(f)[1]
  rmarkdown::render("DcrfTaskVII31Report.Rmd",tmp,output_format = "html_document",params = list(task_vii31
 = dataframe))
  tmpname <- paste(tmp, ".html", sep="")

  readBin(tmpname, "raw", n=file.info(tmpname)$size)
}

#* @param f:file
#* @serializer unboxedJSON
#* @post /dcrftaskvii31size1stmaturprocessing
function(f) {
  sampling_data <- rawToChar(f[[1]])
  text_con1 <- textConnection(sampling_data)
  CS <- read.csv(text_con1,sep=",",header=T)
  res <- TableVII31(CS)
  names(res) <- tolower(names(res))
  res
}

#* @param f:file
#* @serializer contentType list(type="application/html")
#* @post /dcrftaskvii32maturitydata
function(f) {
  tmp <- tempfile()
  data <- rawToChar(f[[1]])
  text_con <- textConnection(data)
  dataframe <- read.table(text_con,sep=",",header=T)
  names(dataframe) <- tolower(names(dataframe))
  df_name <- names(f)[1]
  rmarkdown::render("DcrfTaskVII32Report.Rmd",tmp,output_format = "html_document",params = list(task_vii32
 = dataframe))
  tmpname <- paste(tmp, ".html", sep="")
```

```
  readBin(tmpname, "raw", n=file.info(tmpname)$size)
}

#* @param f:file
#* @serializer contentType list(type="application/html")
#* @post /rcglandings
function(f) {
  tmp <- tempfile()
  data <- rawToChar(f[[1]])
  text_con <- textConnection(data)
  dataframe <- read.table(text_con,sep=",",header=T)
  names(dataframe) <- tolower(names(dataframe))
  df_name <- names(f)[1]
  rmarkdown::render("RcgLandingsReport.Rmd",tmp,output_format = "html_document",params = list(CL =
dataframe))
  tmpname <- paste(tmp, ".html", sep="")

  readBin(tmpname, "raw", n=file.info(tmpname)$size)
}

#* @param f:file
#* @serializer contentType list(type="application/html")
#* @post /rcgsampling
function(f) {
  tmp <- tempfile()
  data <- rawToChar(f[[1]])
  text_con <- textConnection(data)
  dataframe <- read.table(text_con,sep=",",header=T)
  names(dataframe) <- tolower(names(dataframe))
  df_name <- names(f)[1]
  rmarkdown::render("RcgSamplingReport.Rmd",tmp,output_format = "html_document",params = list(CS =
dataframe))
  tmpname <- paste(tmp, ".html", sep="")

  readBin(tmpname, "raw", n=file.info(tmpname)$size)
}

#* @param f:file
#* @serializer contentType list(type="application/octet-stream")
#* @post /meditsrome
function(f) {
  tmp <- paste0("/home/data/", stringi::stri_rand_strings(1, 7, pattern = "[A-Za-z0-9]"))
  dir.create(tmp)
  dir.create(file.path(tmp, "wd"))
  wd <- file.path(tmp, "wd")
  zipfile <- paste0("/home/data/", stringi::stri_rand_strings(1, 10, pattern = "[A-Za-z0-9]"),".zip")
  tadata <- rawToChar(f[[1]])
  text_con_ta <- textConnection(tadata)
  ta <- read.table(text_con_ta,sep=",",header=T)

  tbdata <- rawToChar(f[[2]])
  text_con_tb <- textConnection(tbdata)
  tb <- read.table(text_con_tb,sep=",",header=T)

  tcdata <- rawToChar(f[[3]])
  text_con_tc <- textConnection(tcdata)
  tc <- read.table(text_con_tc,sep=",",header=T)

  ta <- headers.conversion(ta,"TA")
  tb <- headers.conversion(tb,"TB")
  tc <- headers.conversion(tc,"TC")

  te <- NA
  tl <- NA

  # variables definition
  suffix <- NA
  Stratification= RoME::stratification_scheme
  Ref_list= RoME::TM_list
  DataTargetSpecies=RoME::DataTargetSpecies
```

Source File - plumber.R

```
  Maturity=RoME::Maturity_parameters
  ab_parameters=RoME::LW
  stages=RoME::mat_stages
  assTL=RoME::assTL
  verbose = TRUE
  ################ NON MODIFICARE #############

  RoMEcc(TA=ta,
     TB=tb,
     TC=tc,
     TE=te,
     TL=tl,
     wd=wd,
     suffix=suffix,
     verbose=verbose,
     Stratification=Stratification,
     Ref_list=TM_list,
     DataTargetSpecies=DataTargetSpecies,
     Maturity=Maturity_parameters,
     ab_parameters=ab_parameters,
     stages_list=stages,
     assTL=assTL)

  zip::zip(zipfile, "wd", root = tmp)
  readBin(zipfile, "raw", n=file.info(zipfile)$size)
}
```

Source File - mystyle.css

```
td, th { padding : 6px } th { background-color : black ; color : white; border : 1px solid white; } td {
color : black ; border : 1px solid black }
```

Source File - RcgLandingsReport.Rmd

````
---
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
  CL: !r data.frame(matrix(ncol = 0, nrow = 0))
---
<!-- INITIALIZATION -->

```{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)
```

<!-- USER DEFINED SOURCE FILES -->

```{r setup2, include=FALSE}
# reading files
# UNCOMMENT  THE FOLLOWING LINES FOR STAN-ALONE USE
# SELECT THE APPROPRIATE FILE PATHS ON THE LOCAL FOLDERS
CL <- params$CL
# CS <- read.csv("D:/OneDrive - Coispa Tecnologia & Ricerca S.C.A.R.L/RDB/data/CS_GSA17.csv", sep=";")
# CL <- read.csv("D:/OneDrive - Coispa Tecnologia & Ricerca S.C.A.R.L/RDB/data/CL_GSA17.csv", sep=";")


# SPs <- c("Merluccius merluccius")
# MS <- "ITA"
# GSAs <- c("GSA99")
# stages_for_immatures <- c("0", "1", "2a")
```

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

```{r setup3, include=FALSE}
# check the presence of user defined filter for species (SPs)

user_SP <- FALSE

user_MS <- FALSE

user_GSA <- FALSE

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")


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_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
```

Source File - RcgSamplingReport.Rmd

```
---
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
  CS: !r data.frame(matrix(ncol = 0, nrow = 0))
---
<!-- INITIALIZATION -->

```{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)
```

<!-- USER DEFINED SOURCE FILES -->

```{r setup2, include=FALSE}
# reading files
# UNCOMMENT  THE FOLLOWING LINES FOR STAN-ALONE USE
# SELECT THE APPROPRIATE FILE PATHS ON THE LOCAL FOLDERS
CS <- params$CS
# CS <- read.csv("D:/OneDrive - Coispa Tecnologia & Ricerca S.C.A.R.L/RDB/data/CS_GSA17.csv", sep=";")
# CL <- read.csv("D:/OneDrive - Coispa Tecnologia & Ricerca S.C.A.R.L/RDB/data/CL_GSA17.csv", sep=";")


# SPs <- c("Merluccius merluccius")
# MS <- "ITA"
# GSAs <- c("GSA99")
# stages_for_immatures <- c("0", "1", "2a")
```

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

```{r setup3, include=FALSE}
# check the presence of user defined filter for species (SPs)

user_SP <- FALSE

user_MS <- FALSE

user_GSA <- FALSE

user_stages_for_immatures <- FALSE

```

\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")

g=1
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")


g=1
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")


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_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")


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_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")


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_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
```


stages_for_immatures

```{r RCG_check_mat_ogive, results = 'asis'}
if (check_CS) {
  cat("\n## check_mat_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 = ""))


  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"))
      # 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")
```

```
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_AL(data=CS, MS = MS, GSA = GSAs[g],SP = SPs[s], min_age = NA,
max_age = NA))
        if (!is.null(res)) {

        # cat(paste0("\n_Summary table of length observations by lenght class 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 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
```

Source File - RDBFIS_report_template.html

```
<!DOCTYPE html>

<html$if(lang)$ lang="$lang$" xml:lang="$lang$"$endif$>

<head>

<meta charset="utf-8" />
<meta name="generator" content="pandoc" />
<meta http-equiv="X-UA-Compatible" content="IE=EDGE" />

$if(theme)$
$else$
<meta name="viewport" content="width=device-width, initial-scale=1" />
$endif$

$for(author-meta)$
<meta name="author" content="$author-meta$" />
$endfor$

$if(date-meta)$
<meta name="date" content="$date-meta$" />
$endif$

<title>$if(title-prefix)$$title-prefix$ - $endif$$pagetitle$</title>

$for(header-includes)$
$header-includes$
$endfor$
$if(shiny-prerendered)$
<!-- HEAD_CONTENT -->
$endif$

<style type="text/css">
  code{white-space: pre-wrap;}
  span.smallcaps{font-variant: small-caps;}
  span.underline{text-decoration: underline;}
  div.column{display: inline-block; vertical-align: top; width: 50%;}
  div.hanging-indent{margin-left: 1.5em; text-indent: -1.5em;}
  ul.task-list{list-style: none;}
  $if(quotes)$
  q { quotes: "“" "”" "‘" "’"; }
  $endif$
  $if(displaymath-css)$
  .display.math{display: block; text-align: center; margin: 0.5rem auto;}
  $endif$
</style>

$if(highlightjs)$
<style type="text/css">code{white-space: pre;}</style>
<script type="text/javascript">
if (window.hljs) {
  hljs.configure({languages: []});
  hljs.initHighlightingOnLoad();
  if (document.readyState && document.readyState === "complete") {
    window.setTimeout(function() { hljs.initHighlighting(); }, 0);
  }
}
</script>
$endif$

$if(highlighting-css)$
<style type="text/css">
  code {
    white-space: pre;
  }
  .sourceCode {
    overflow: visible;
  }
</style>
<style type="text/css" data-origin="pandoc">
```

Source File - RDBFIS_report_template.html

```
$highlighting-css$

$if(code_folding)$
.sourceCode .row {
  width: 100%;
}
.sourceCode {
  overflow-x: auto;
}
.code-folding-btn {
  margin-right: -30px;
}
$endif$
</style>
<script>
// apply pandoc div.sourceCode style to pre.sourceCode instead
(function() {
  var sheets = document.styleSheets;
  for (var i = 0; i < sheets.length; i++) {
    if (sheets[i].ownerNode.dataset["origin"] !== "pandoc") continue;
    try { var rules = sheets[i].cssRules; } catch (e) { continue; }
    for (var j = 0; j < rules.length; j++) {
      var rule = rules[j];
      // check if there is a div.sourceCode rule
      if (rule.type !== rule.STYLE_RULE || rule.selectorText !== "div.sourceCode") continue;
      var style = rule.style.cssText;
      // check if color or background-color is set
      if (rule.style.color === '' && rule.style.backgroundColor === '') continue;
      // replace div.sourceCode by a pre.sourceCode rule
      sheets[i].deleteRule(j);
      sheets[i].insertRule('pre.sourceCode{' + style + '}', j);
    }
  }
})();
</script>

$if(toc)$
<div id="$idprefix$TOC">
<h1 class="toc-title">$toc-title$</h1>
$toc$
</div>
$endif$



$endif$

$if(abstract)$
<style type="text/css">
  p.abstract{
    text-align: center;
    font-weight: bold;
  }
  div.abstract{
    margin: auto;
    width: 90%;
  }
</style>
$endif$

$if(source_embed)$
<style type="text/css">
#rmd-source-code {
  display: none;
}
</style>
$endif$

$if(csl-css)$
<style type="text/css">
```

Source File - RDBFIS_report_template.html

```
/* for pandoc --citeproc since 2.11 */
div.csl-bib-body { }
div.csl-entry {
  clear: both;
$if(csl-entry-spacing)$
  margin-bottom: $csl-entry-spacing$;
$endif$
}
.hanging div.csl-entry {
  margin-left:2em;
  text-indent:-2em;
}
div.csl-left-margin {
  min-width:2em;
  float:left;
}
div.csl-right-inline {
  margin-left:2em;
  padding-left:1em;
}
div.csl-indent {
  margin-left: 2em;
}
</style>
$endif$

$for(css)$
<link rel="stylesheet" href="$css$" $if(html5)$$else$type="text/css" $endif$/>
$endfor$



$if(theme)$
<style type = "text/css">
.main-container {
  max-width: 940px;
  margin-left: auto;
  margin-right: auto;
}
img {
  max-width:100%;
}
.tabbed-pane {
  padding-top: 12px;
}
.html-widget {
  margin-bottom: 20px;
}
button.code-folding-btn:focus {
  outline: none;
}
summary {
  display: list-item;
}
pre code {
  padding: 0;
}
</style>

$if(kable-scroll)$
<style type="text/css">
.kable-table {
  border: 1px solid #ccc;
  border-radius: 4px;
  overflow: auto;
  padding-left: 8px;
  padding-right: 8px;
  margin-bottom: 20px;
  max-height: 350px;
}
```

```
.kable-table table {
  margin-bottom: 0px;
}

.kable-table table>thead>tr>th {
  border: none;
  border-bottom: 2px solid #dddddd;
}

.kable-table table>thead {
  background-color: #fff;
}
</style>
$endif$

$if(navbar)$
<style type="text/css">
.dropdown-submenu {
  position: relative;
}
.dropdown-submenu>.dropdown-menu {
  top: 0;
  left: 100%;
  margin-top: -6px;
  margin-left: -1px;
  border-radius: 0 6px 6px 6px;
}
.dropdown-submenu:hover>.dropdown-menu {
  display: block;
}
.dropdown-submenu>a:after {
  display: block;
  content: " ";
  float: right;
  width: 0;
  height: 0;
  border-color: transparent;
  border-style: solid;
  border-width: 5px 0 5px 5px;
  border-left-color: #cccccc;
  margin-top: 5px;
  margin-right: -10px;
}
.dropdown-submenu:hover>a:after {
  border-left-color: #adb5bd;
}
.dropdown-submenu.pull-left {
  float: none;
}
.dropdown-submenu.pull-left>.dropdown-menu {
  left: -100%;
  margin-left: 10px;
  border-radius: 6px 0 6px 6px;
}
</style>

<script type="text/javascript">
// manage active state of menu based on current page
$$(document).ready(function () {
  // active menu anchor
  href = window.location.pathname
  href = href.substr(href.lastIndexOf('/') + 1)
  if (href === "")
    href = "index.html";
  var menuAnchor = $$('a[href="' + href + '"]');

  // mark it active
  menuAnchor.tab('show');
```

```
  // if it's got a parent navbar menu mark it active as well
  menuAnchor.closest('li.dropdown').addClass('active');

  // Navbar adjustments
  var navHeight = $$(".navbar").first().height() + 15;
  var style = document.createElement('style');
  var pt = "padding-top: " + navHeight + "px; ";
  var mt = "margin-top: -" + navHeight + "px; ";
  var css = "";
  // offset scroll position for anchor links (for fixed navbar)
  for (var i = 1; i <= 6; i++) {
    css += ".section h" + i + "{ " + pt + mt + "}\n";
  }
  style.innerHTML = "body {" + pt + "padding-bottom: 40px; }\n" + css;
  document.head.appendChild(style);
});
</script>
$endif$

<!-- tabsets -->

<style type="text/css">
.tabset-dropdown > .nav-tabs {
  display: inline-table;
  max-height: 500px;
  min-height: 44px;
  overflow-y: auto;
  border: 1px solid #ddd;
  border-radius: 4px;
}

.tabset-dropdown > .nav-tabs > li.active:before {
  content: "";
  font-family: 'Glyphicons Halflings';
  display: inline-block;
  padding: 10px;
  border-right: 1px solid #ddd;
}

.tabset-dropdown > .nav-tabs.nav-tabs-open > li.active:before {
  content: "&#xe258;";
  border: none;
}

.tabset-dropdown > .nav-tabs.nav-tabs-open:before {
  content: "";
  font-family: 'Glyphicons Halflings';
  display: inline-block;
  padding: 10px;
  border-right: 1px solid #ddd;
}

.tabset-dropdown > .nav-tabs > li.active {
  display: block;
}

.tabset-dropdown > .nav-tabs > li > a,
.tabset-dropdown > .nav-tabs > li > a:focus,
.tabset-dropdown > .nav-tabs > li > a:hover {
  border: none;
  display: inline-block;
  border-radius: 4px;
  background-color: transparent;
}

.tabset-dropdown > .nav-tabs.nav-tabs-open > li {
  display: block;
  float: none;
}
```

```
.tabset-dropdown > .nav-tabs > li {
  display: none;
}
</style>

<!-- code folding -->
$if(code_menu)$
<style type="text/css">
.code-folding-btn { margin-bottom: 4px; }
</style>
$endif$


$if(toc_float)$

<style type="text/css">

#$idprefix$TOC {
  margin: 25px 0px 20px 0px;
}
@media (max-width: 768px) {
#$idprefix$TOC {
  position: relative;
  width: 100%;
}
}

@media print {
$if(toc_print)$
$else$
#$idprefix$TOC {
  display: none !important;
}
$endif$
.toc-content {
  /* see https://github.com/w3c/csswg-drafts/issues/4434 */
  float: right;
}
}

.toc-content {
  padding-left: 30px;
  padding-right: 40px;
}

div.main-container {
  max-width: 1200px;
}

div.tocify {
  width: 20%;
  max-width: 260px;
  max-height: 85%;
}

@media (min-width: 768px) and (max-width: 991px) {
  div.tocify {
    width: 25%;
  }
}

@media (max-width: 767px) {
  div.tocify {
    width: 100%;
    max-width: none;
  }
}

.tocify ul, .tocify li {
  line-height: 20px;
```

```
}

.tocify-subheader .tocify-item {
  font-size: 0.90em;
}

.tocify .list-group-item {
  border-radius: 0px;
}

$if(toc_collapsed)$
$else$
.tocify-subheader {
  display: inline;
}
.tocify-subheader .tocify-item {
  font-size: 0.95em;
}
$endif$

</style>

$endif$

$endif$

</head>

<body>

$if(theme)$

<div class="container-fluid main-container">

$if(toc_float)$

<!-- setup 3col/9col grid for toc_float and main content  -->
<div class="row">
$if(bs3)$
<div class="col-xs-12 col-sm-4 col-md-3">
$else$
<div class="col-sm-12 col-md-4 col-lg-3">
$endif$
$if(toc-title)$
<h2 id="$idprefix$toc-title">$toc-title$</h2>
$endif$
<div id="$idprefix$TOC" class="tocify">
</div>
</div>

$if(bs3)$
<div class="toc-content col-xs-12 col-sm-8 col-md-9">
$else$
<div class="toc-content col-sm-12 col-md-8 col-lg-9">
$endif$

$endif$

$endif$

$for(include-before)$
$include-before$
$endfor$

$if(theme)$
<div id="$idprefix$header">

$if(code_menu)$
<div class="btn-group pull-right float-right">
<button type="button" class="btn btn-default btn-xs btn-secondary btn-sm dropdown-toggle" data-toggle="
```

Source File - RDBFIS_report_template.html

```
dropdown" aria-haspopup="true" aria-expanded="false"><span>Code</span> <span class="caret"></span></button>
<ul class="dropdown-menu dropdown-menu-right" style="min-width: 50px;">
$if(code_folding)$
<li><a id="rmd-show-all-code" href="#">Show All Code</a></li>
<li><a id="rmd-hide-all-code" href="#">Hide All Code</a></li>
$if(source_embed)$
<li role="separator" class="divider"></li>
$endif$
$endif$
$if(source_embed)$
<li><a id="rmd-download-source" href="#">Download Rmd</a></li>
$endif$
</ul>
</div>

$endif$

$endif$

$if(title)$
<h1 class="title toc-ignore">$title$</h1>
$if(subtitle)$
<h3 class="subtitle">$subtitle$</h3>
$endif$
$for(author)$
$if(author.name)$
<h4 class="author">$author.name$</h4>
$if(author.affiliation)$
<address class="author_afil">
$author.affiliation$<br>$endif$
$if(author.email)$
<a class="author_email" href="mailto:#">$author.email$</a>
</address>
$endif$
$else$
<h4 class="author">$author$</h4>
$endif$
$endfor$
$if(date)$
<h4 class="date">$date$</h4>
$endif$
$if(abstract)$
<div class="abstract">
<p class="abstract">Abstract</p>
$abstract$
</div>
$endif$
$endif$

$if(theme)$
</div>
$endif$

$if(toc_float)$
$else$
$if(toc)$
<div id="$idprefix$TOC">
$if(toc-title)$
<h2 id="$idprefix$toc-title">$toc-title$</h2>
$endif$
$toc$
</div>
$endif$
$endif$

$body$

$for(include-after)$
$include-after$
$endfor$
```

```
$if(theme)$

$if(toc_float)$
</div>
</div>
$endif$

</div>

<script>

// add bootstrap table styles to pandoc tables
function bootstrapStylePandocTables() {
  $$('tr.odd').parent('tbody').parent('table').addClass('table table-condensed');
}
$$(document).ready(function () {
  bootstrapStylePandocTables();
});

$if(kable-scroll)$
$$(document).ready(function () {
  $$('.knitsql-table').addClass('kable-table');
  var container = $$('.kable-table');
  container.each(function() {

    // move the caption out of the table
    var table = $$(this).children('table');
    var caption = table.children('caption').detach();
    caption.insertBefore($$(this)).css('display', 'inherit');
  });
});
$endif$

</script>

<!-- tabsets -->

<script>
$$(document).ready(function () {
  window.buildTabsets("$idprefix$TOC");
});

$$(document).ready(function () {
  $$('.tabset-dropdown > .nav-tabs > li').click(function () {
    $$(this).parent().toggleClass('nav-tabs-open');
  });
});
</script>
$endif$

<!-- code folding -->
$if(code_menu)$
<script>
$$(document).ready(function () {
$if(source_embed)$
  window.initializeSourceEmbed("$source_embed$");
$endif$
$if(code_folding)$
  window.initializeCodeFolding("$code_folding$" === "show");
$endif$
});
</script>
$endif$

$if(toc_float)$
<script>
$$(document).ready(function ()  {

    // temporarily add toc-ignore selector to headers for the consistency with Pandoc
```

```
    $$('.unlisted.unnumbered').addClass('toc-ignore')

    // move toc-ignore selectors from section div to header
    $$('div.section.toc-ignore')
        .removeClass('toc-ignore')
        .children('h1,h2,h3,h4,h5').addClass('toc-ignore');

    // establish options
    var options = {
      selectors: "$toc_selectors$",
      theme: "bootstrap3",
      context: '.toc-content',
      hashGenerator: function (text) {
        return text.replace(/[.\\/?&!#<>]/g, '').replace(/\s/g, '_');
      },
      ignoreSelector: ".toc-ignore",
      scrollTo: 0
    };
$if(toc_collapsed)$
    options.showAndHide = true;
$else$
    options.showAndHide = false;
$endif$
$if(toc_smooth_scroll)$
    options.smoothScroll = true;
$else$
    options.smoothScroll = false;
$endif$

    // tocify
    var toc = $$("#$idprefix$TOC").tocify(options).data("toc-tocify");
});
</script>
$endif$

$if(mathjax-url)$
<!-- dynamically load mathjax for compatibility with self-contained -->
<script>
  (function () {
    var script = document.createElement("script");
    script.type = "text/javascript";
    script.src  = "$mathjax-url$";
    document.getElementsByTagName("head")[0].appendChild(script);
  })();
</script>
$endif$

</body>
</html>
```