9 February 2016

Guidelines for the VMSdatacall_2016_proposedWorkflow.r

ICES CIEM
International Council for the Exploration of the Sea
Conseil International pour l'Exploration de la Mer

Part 1

This document is designed to aid analysts streamline the process of extracting VMS data in accordance with requirements of the ICES VMS data call.

Part 1 of the document is provides guidelines for installing all the software necessary for data manipulation and aggregation into the requested format. The software used, R and RStudio, are available as freeware.

The document is designed to aid all users, regardless of their experience using R. The steps listed cover the installation of R and RStudio and detailed information will be provided to cover all stages of the installation process to ensure success. Depending on your skills you might want to jump some of the steps. To ensure consistency across all users we advise installing VMStools version 0.72. Following these steps should enable quick and simple processing of all data.

All the instructions and code below were tested in windows 7 and 8. However, if for any reason something is not working you can contact one of the members of our support team (emails at the end of the document).

Step 1: Installation of R

(Where R is already installed (any 3.x.x 32 bits version) move to step 2)

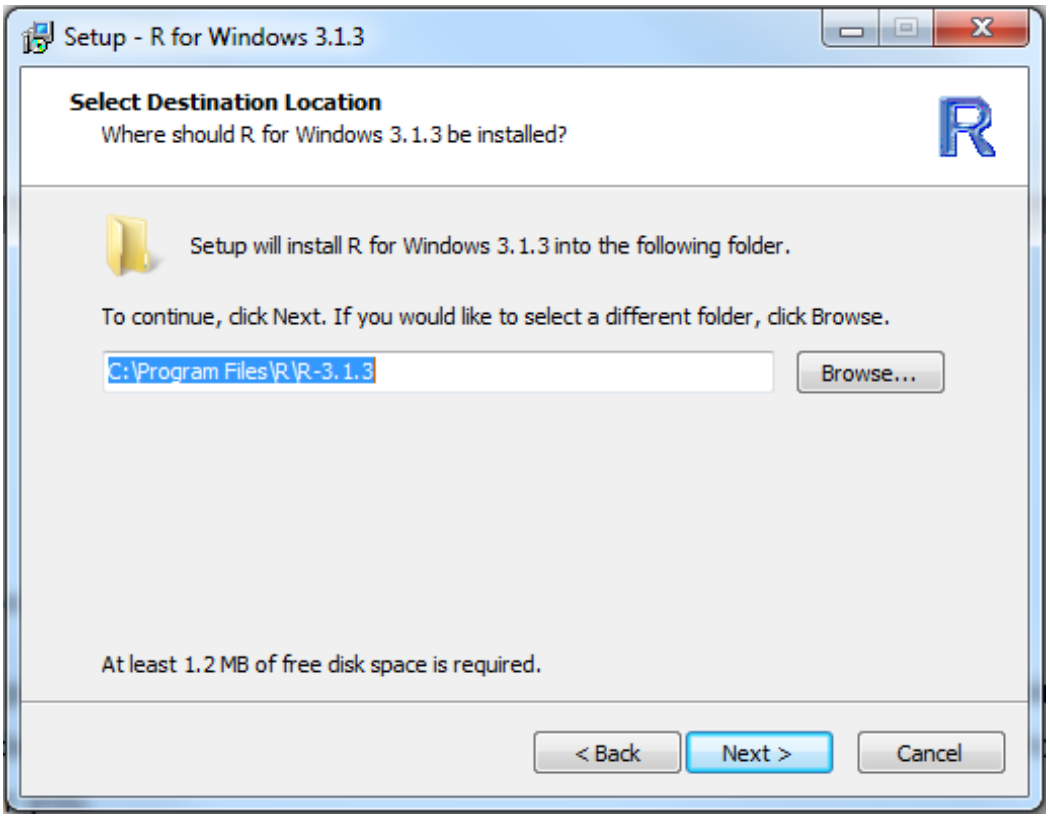
Completion of the first two steps of these guidelines is dependent on the user's computer security setting. In instances where administrator privilege is required then please ask a member of your IT/computer support team to run the first two steps for you.

So let's start by installing R; click on the link below to download R version 3.1.3.

https://cran.rstudio.com/bin/windows/base/old/3.1.3/R-3.1.3-win.exe

Once it is downloaded, double click on the file. Depending on your security settings you might get a pop up security warning asking if you want to Run or Cancel the installation, Click **Run**.

- Select a language (when you select the language, bear in mind that these instructions are in English)
- At this point you should be on the R installation Wizard menu. Just click **Next**
- Here you are presented with the GNU general public license which you are most welcome to read. Click **Next**
- The menu (picture below) will appear and you will be asked to select the destination folder



Rather than accepting the default (C:Files-3.1.3) you should click on the browse button and create a directory C:-3.2.1 this will allow you to install packages without having administrator privileges. Once you have changed the folder just click **Next**

- In the menu "Select components" simply deselect/untick the 64-bit files. Click **Next**
- Click **Next** all the menus until the end of installation, and that's it; R is now installed on the computer.

Step 2: Installation of RStudio

(Where R studio is already installed move to step 3)

If you don't have administrator privilege on your computer you will need to call your IT/computer support to install RStudio. First download RStudio by clicking on the link below:

https://download1.rstudio.org/RStudio-0.99.467.exe

Once downloaded, double click on it. Depending on your security settings you might get a popup security warning asking if you want to Run or Cancel the installation, Click on **Run**. At this point you should be on the RStudio installation Wizard menu. RStudio is

very easy to install so just accept all defaults and click **Next** in all the menus until the end.

That's it Step 2 is complete, RStudiois now installed.

Step 3: Installing vmstools

Next, download a compiled version of vmstools 0.72 by clicking the link below. Make sure you click save rather than open.

https://github.com/nielshintzen/vmstools/releases/download/0.72/vmstools_0.72.zip

Now start Rstudio to install all the necessary R packages that vmstools depends on.

Copy the following text

```
in-
stall.packages(c("cluster","data.table","doBy","maps","mapdata","
maptools","PBSmapping","sp","Matrix","ggplot2"))
```

into the console and press enter. It should start installing all the packages needed. This might take two or three minutes and your console should look like this once it has finished.

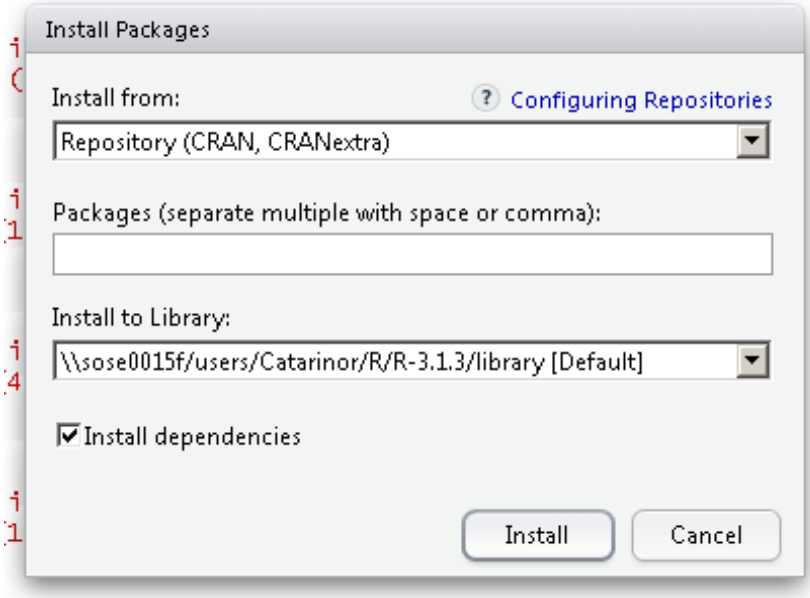
```
trying URL 'http://cran.rstudio.com/bin/windows/contrib/3.1/PBSmapping_2.69.76.zip'
Content type 'application/zip' length 4790638 bytes (4.6 MB)
opened URL
downloaded 4.6 MB

trying URL 'http://cran.rstudio.com/bin/windows/contrib/3.1/sp_1.1-1.zip'
Content type 'application/zip' length 1516713 bytes (1.4 MB)
opened URL
downloaded 1.4 MB

package 'stringi' successfully unpacked and MD5 sums checked
package 'magrittr' successfully unpacked and MD5 sums checked
package 'plyr' successfully unpacked and MD5 sums checked
package 'stringr' successfully unpacked and MD5 sums checked
package 'Rcpp' successfully unpacked and MD5 sums checked
package 'chron' successfully unpacked and MD5 sums checked
package 'reshape2' successfully unpacked and MD5 sums checked
package 'cluster' successfully unpacked and MD5 sums checked
package 'data.table' successfully unpacked and MD5 sums checked
package 'doBy' successfully unpacked and MD5 sums checked
package 'maps' successfully unpacked and MD5 sums checked
package 'mapdata' successfully unpacked and MD5 sums checked
package 'maptools' successfully unpacked and MD5 sums checked
package 'PBSmapping' successfully unpacked and MD5 sums checked
package 'sp' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\catarinor\AppData\Local\Temp\RtmpmcAAg7\downloaded_packages
>
```

To install vmstools click on the Tools tab on the main menu in RStudio. Then select Install Packages. This will trigger a popup menu like the one below. Click on the down arrow in the "Install from:" dialogue box and select the second option "Package Archive File(Zip;tar.gz)" then browse to the vmstools zip file that you just downloaded and press Install.



That's it; you now have all you need to process your data.

Part 2

Part one of these guidelines have guided you through the installation of all the software needed to process your data into the formats specified in the data call. Now, part two will focus on guiding you through the eight steps that comprise the workflow. The aim is to get your data converted into the tables requested by the data call in Annex 1 and 2. .

The work flow was developed in R, the principal objective being to facilitate the submission of data in the specified format by providing all country institutions with a standardised tool for data extraction. This will make data outputs more comparable and easy to work with.

Many of the specific functions that this workflow uses to extract and process VMSVMS and landings data are part of the open-source vmstools package. This package was specifically developed to work with VMS and landings data. If you want to learn more about this tool you will find plenty of information and tutorials in the link below.

https://github.com/nielshintzen/VMStools/wiki

Before initiating the workflow, it is first necessary that you have all VMS and landings data in the tacsat2 and eflalo2 formats respectively. If you need more information about either format you can download a document with the detailed specifications for both in the link below.

https://github.com/nielshintzen/VMStools/releases/download/0.0/Exchange_EFLALO2_v2-1.doc

If you are experiencing difficulties and your data is not yet in the tacsat and/or eflalo format please get in touch with one of the contacts at the bottom of these guidelines. Someone will get back to you and help you to rearrange your data into the specified formats allowing you to move on with the analysis.

The proposed workflow is not a one-size-fits-all solution and there are parts of the script that may need to be adapted to allow for the specific nature of the fisheries data from each country. Throughout this document all parts of code that need adjustment will be highlighted and explained so you should have a good understanding of what is happening at all times.

The code is divided in eight sections and within these there are many blocks. We will explain briefly what each section and block does and its purpose within the code. The idea of the guidelines is not to explain in detail what each line of code does but to give an overview of what is happening at each stage. The script itself is well annotated, so if you are familiar with R and the vmstools package, you probably won't need to follow these guidelines as closely.

Let's begin:

House keeping

Open "Rstudio" and load the workflow. Before making any changes save the script with a different name. This will allow you to quickly refer back to the original code in case anything unexpected happens.

Just a quick note, which may be very obvious for all of those using R, but not so much for someone just trying to follow the guidelines. Anything in the code after a hashtag (#) sign is not code and it won't be read by the program. The # sign is used to add sections and block headers or general annotations. As such, the first line of code starts on line 23.

To make it easier to follow the guidelines we will explain the code, referencing the sections, headers (numbered) and blocks by highlighting them in bold.

- Clear workspace

This code will just clear your work space to allow you to start afresh. Also, the three packages that will be needed to run the code will be loaded into the session. If you followed the instructions in part one these should already be installed and loading them shouldn't be a problem.

#- Settings paths

At this point you need to replace the paths shown in the code with your own ones. The approach used in the code is one of best practice, as everything will be in one main folder "VMSdatacall". This will make it very easy to navigate between folders and to backup. So, for your own convenience it is recommended that you use the default paths as listed in the code. However, it is possible to change these by simply specifying your personal destination folders and defining the chosen path.

#- Settings and specific thresholds

The thresholds here defined will be used later in different processes throughout the code. These will include, data cleaning or definition of vessel state (i.e. fishing/not fishing). The values set for the thresholds are considered to be reasonable and unless there are particularities in your data there shouldn't be a need to change these values.

#- Load OSPAR and HELCOM areas

Before running this block, you will need to download and unzip (to the Polygons folder) two GIS shapefiles that will be used at this stage. The shapefiles are polygons of the OSPAR and HELCOM areas and they will be used to identify the VMS pings and landings within these areas. Links to the files are below:

http://geo.ices.dk/download.php?dataset=ext_ref:helcom_subbasins
http://geo.ices.dk/download.php?dataset=ext_ref:ospar_regions_without_coastline

Once you have unzipped both files into the "Polygons" folder you can then run the code which will load the shapefiles into the session.

1) Load the data

#- 1a) Load VMStools underlying data

This will load into the session support data such as a map of Europe, list of harbours and ICES areas that will be used throughout the code.

#-1b) Looping through the data

The next line of code has a "for" loop which means that the all the code within the loop will run at the same time. This particular loop stretches from line 56 to line 505 leaving only a couple of lines at the end of the code. However, before running the "for" loop there are a few things that need to be changed/ adapted to your case.

In order to ensure that everything is working properly and to have a better understanding of what the code is doing inside the loop we will run one single year as a test. If we are able to run one year of data without coming across errors then we can run the code for all the years at once.

So for the moment we will ignore this block **(#-1b) Looping through the data** and copy the line below into the console:

```
year<-2009
```

#- 1c) load tacsat and eflalo data from file

In your "Data" folder you should have all your tacsat and eflalo files in the .RDATA format. In the code it is expected that your files have the following naming convention "tacsat_ XXXX" i.e. tacsat_2009; tacsat_2010, etc. The same naming convention is applied to the eflalo files. This will allow the code to load the files as they are needed during the "for" loop. Failing to correctly name the files will result in an error.

Since we have just copied **year<- 2009** to the console, when we run this block only the 2009 year data will be loaded.

Now that you have just loaded both tacsat and eflalo for 2009 the next two lines will just change the name of your objects to "tacsat" and "eflalo". Depending on what your objects are called you may need to change the code. (tip: If at any stage you don't remember what the names are just type "ls()" and a list of all objects already loaded will appear). If your object names are not "tacsat" and "eflalo" you will need to adapt the code. However, this is a simple process. Below are two examples for changing objects names in this case the names are "2009Tacsat" and another called "Tac09".

```
tacsat <- get(paste(year,"Tacsat",sep=""))
tacsat <- get(paste("Tac",substr(year,3,4),sep=""))
```

If the objects are already called tacsat and eflalo then, you don't need to run those two lines and you can add a # at the beginning of each line. However, make sure that whatev-

er you have called your objects the naming structure is consistent across all years otherwise the "For" loop won't run.

#- Make sure data is in right format

It just ensures that your files are formatted properly.

#- Take only VMS pings and eflalo records inside OSPAR and HELCOM region

This block of code will identify all VMS pings (tacsat) and landings (eflalo) within the OSPAR and HELCOM regions.

2) Clean the tacsat data

This section will focus on "cleaning" the data in the tacsat file. The information in the tacsat (vms) comes from an electronic system that uses GPS information to collect the data on board the vessel and uses a satellite link to send the data to the database. Despite the reliability of this system conditions at sea are not always the best. There are two main opportunities for errors to occur, when receiving or sending data from the GPS and to the database. The code in this section will look to the most common errors and try to identify all of them. The code will not only delete the errors but also keep a record of what was deleted allowing you to keep track of how much data you have lost due to errors.

#- Keep track of removed points

This section will check for five common types of errors. At each of these checks errors will be removed from the tacsat object. However, the data removed will be kept and saved in the "Results" folder so you can verify the errors. Also, the volume for each of errors for each of the five checks is recorded the "remrecsTacsat" object. This object will tell you percentage wise how much you have lost in relation to the original tacsat object.

#- Remove duplicate records

#- Remove points that cannot be possible

#- Remove points which are pseudo duplicates as they have an interval rate < x minutes

#- Remove points in harbour

#- Remove points on land

All of the above are self-explanatory and each of the five blocks will check for a particular type of error, remove them where they occur and store the removed entries in the "Results" folder and will quantify the number of values removed.

#- Save the remrecsTacsat file

The file is now saved and by typing "remrecsTacsat" into the console you will get an overview of how much data was lost due to errors.

#- Save the cleaned tacsat file

Now you have your file cleaned and saved so no need to repeat the process in future analysis.

3) Clean the eflalo data

This section, like the previous one also focuses on "cleaning" the data. This time the target is the eflalo file. The types of errors are of a different nature but once again the code tries to account for the most commons errors. As in the previous section, the code will keep track of what data has been removed and how much. All these files can be found in the "Results" folder. One should spend a bit of time looking at the data removed as it can be very useful to understand why and where problems occur.

#- Keep track of removed points

The "remrecsEflalo" object will keep you informed of how much data has been removed.

#- Warn for outlying catch records

Basically this block looks for outliers. For each species, it generates a data-driven outlier threshold. If any outliers are found, these will be converted into "NA" values. You can check in the "Results" folder for the files containing all the outliers and you can double check if they are correct or not. If they were correct then you can run the code again but the code will need some adjustments.

#- Remove non-unique trip numbers

#- Remove impossible time stamp records

#- Remove trip starting before 1st Jan

#- Remove trip with overlap with another trip

#- Remove records with arrival date before departure date

The above block headers are self-explanatory and the code in each of the blocks is just identifying those common errors and removing them from the eflalo object.

#- Save the remrecsEflalo file

#- Save the cleaned eflalo file

The "remrecsEflalo" file is saved for future reference. So is the cleaned eflalo file which, like the tacsat, will be ready to use in the future.

4) Merge the tacsat and eflalo data together

In section four we bring the tacsat and eflalo together by merging to create a new object. This will enable us to relate the landings component (eflalo) to the vessel activity i.e. VMS (tacsat).

#- Merge eflalo and tacsat

The files tacsat and eflalo will be combined using some very clever algorithms that use the vessel identifier and date and time in both data sets to relate the landings to the corresponding VMS data for the same trip.

#- Assign gear and length to tacsat. The new object tacsatp is now a merged version of the cleaned tacsat and eflalo objects. However, the new object hasn't inherited all the fields from eflalo due to reasons of processing speed and workability. At this point we will extract some data from the eflalo dataset to populate the corresponding tacsatp fields. The data we are interested in are data that will be used later on in the code to populate the final data tables. Things like gear; kw; metiers, etc.

#- Save not merged tacsat data

Not all vessel activity is associated with fishing events; quite often vessels may be testing equipment or chartered to do jobs other than fishing. So, the merge executed in the previous block only includesVMS data that can be linked to corresponding landings records. As such, it will not be possible to merge all tacsat data for allocation to the tacsatp object. This block will save both the merged and non-merged data into the "Results" folder.

5) Define activity

This is a crucial section, as vessel activity will be defined here. Also, this is the section that needs the most customization for which some knowledge of fisheries activities will be needed. In this section we will try to explain the steps in more detail and incorporate some reproducible examples as well. The first couple of lines in this section will calculate time interval between points. The time values and the interval threshold will be paramount in identifying vessel activity later on.

#- Remove points with NA's in them in critical places

This block gets rid of any rows in the tacsatp for which critical information (vessel reference, latitude, longitude, speed, date and time) is missing. If this data wasn't removed it would most likely lead to errors.

#- Define speed thresholds associated with fishing for gears

The code in this block creates a very useful plot of speed frequency by gear. This plot will be saved in the "Results" folder and before you run any further code you should look closely at the output plot.

The three last lines of this block will create a threshold object. However, your input and knowledge of the relevant fisheries will be needed at this stage. The threshold object will

hold the minimum and maximum speed of fishing for each of the gears (i.e. the minimum and maximum speeds at which the specific fishing activity is thought to occur). To help you with this task you should look at the previous speed frequency plot to help distinguish steaming from fishing events.

By running the third last line in this block you create an object "speedarr" with all the different gears in your data. The second and third lines will fill in column 2 and 3 of the "speedarr" object with the minimum and maximum fishing speeds. These values are set to 1kt and 6kt by default. At this stage you will need to set up the upper and lower limits for each of the gears. Although there are several ways of accomplishing this, we will demonstrate one of them here. In the example below we use a list of 5 gears (DRB; PTB; OTT; GN; FPO) although you are likely to have many more gears in your dataset so you will need to extend the code to accommodate all gears accordingly.

To create our example, copy the code below into the console:

```
speedarr <- as.data.frame(cbind(LE_GEAR=c("DRB", "PTB", "OTT",
"GN", "FPO"),min=NA,max=NA),stringsAsFactors=F)

speedarr

speedarr$min[which(speedarr$LE_GEAR=="DRB")]<- 2   ; speed-
arr$max[which(speedarr$LE_GEAR=="DRB")]<- 5
speedarr$min[which(speedarr$LE_GEAR=="PTB")]<- 2   ; speed-
arr$max[which(speedarr$LE_GEAR=="PTB")]<- 6
speedarr$min[which(speedarr$LE_GEAR=="OTT")]<- 1.5 ; speed-
arr$max[which(speedarr$LE_GEAR=="OTT")]<- 4
speedarr$min[which(speedarr$LE_GEAR=="GN")]<- 0.5  ; speed-
arr$max[which(speedarr$LE_GEAR=="GN")]<- 3
speedarr$min[which(speedarr$LE_GEAR=="FPO")]<- 0.5 ; speed-
arr$max[which(speedarr$LE_GEAR=="FPO")]<- 3

speedarr
```

So, in the example above you can easily see how each line applies to one gear and on the left you have the minimum values on the right the maximum. Make sure when you copy and paste the lines you change the gears and values on both sides.

If your data varies from year to year you might want to check if you have different gears in different years. Before running the full code, you should make sure that all gears are included in the code above.

#- Analyse activity automated for common gears only. Use the speedarr for the other gears

This block allows you to select some gears for which the detection can be done automatically. This is another functionality of VMStools which applies for the most common gears. So, in this block you will need to choose which gears to want to apply auto detection. You can add or delete gears in the first line of code in this block.

The remainder of the code in the block will split the tacsatp object in two depending on whether gears will be detected automatically or whether the thresholds need to be user defined according to the code from the previous block. The remaining lines in the block don't need to be changed.

#- Fill the storeScheme values based on analyses of the pictures

In this block the speed histogram plot created previously will be used once more. You will have to identify the peaks in the plot for the gears for which you want the activity to be automatically detected (bear in mind that the algorithm was developed with trawling in mind).

So, first of all make sure you have a line for each of the gears (see line below) changing the gears and mean speeds accordingly.

```
storeScheme$means[which(storeScheme$analyse.by == "SSC")] <- c("-
9 0 9")
```

Now using the plot, identify where the peaks are and use this to change the code. Make sure you follow the same nomenclature as the example provided. Also, for the algorithm to perform better, we need to create a mirror image of the peaks and with 0 (zero) in the middle .If the number of peaks for a particular gear is greater or less than 5 you will need to add a line (like the one below) with the true number of peaks observed. In the example above there were three peaks -9, 0, 9 so we would need to add the line below to the code.

```
storeScheme$peaks[which(storeScheme$analyse.by == "SSC")]   <- 3
```

The second half of the block, checks the results of the auto detection; if they are not satisfactory the analysis is run once more; this time using fixed peaks. However, in this workflow we will not be using fixed peaks so no need to worry about this.

#- Assign for visually inspected gears a simple speed rule classification

This block deals with all the other gears that are not automatically detected. The code simply applies the upper and lower limits defined previously to define if vessel activity as either steaming or fishing.

#- Combine the two dataset together again

Now that all gears have had their activity defined, the code in this block is just putting it all back together in one object. As usual the object will be saved in the "Results" folder.

6) Dispatch landings of merged eflalo at the ping scale

This section calculates the total daily landings (weight and value) and splits the values equally among the daily fishing pings.

7) Assign c-square, year, month, quarter, area and create table 1

We are at final stage and the code in this block will generate one of the tables requested by the data call (see annex 1). The first part of the code pulls together all the fields needed to create the table. The second part deals with the aggregation by CSquare.

8) Assign year, month, quarter, area and create table 2

We have reached the last section. As in the previous one, the first part of the code will create all the fields needed for table 2 (see annex 2) as requested in the data call. The second part of the code deals with the aggregation into CSquares.

Running the last two lines saves the data into the "Results" folder. Don't forget to check the outputs to make sure that everything is correct.

If you are happy with all the results then you can proceed running the entire code. Make sure you double check the names of the input files, ensuring they follow the convention (as in the example) and run the entire code.

Contacts:

Josefine Egekvist: jsv@aqua.dtu.dk

Niels Hintzen: niels.hintzen@wur.nl

Rui Catarino: r.catarino@marlab.ac.uk

Annex 1: Exchange format for combined VMS and Log book data

Order	Name	Type	Req.	Basic checks	Comments
1	Record type	String	M		Fixed value VE
2	Vessel Flag Country	String	M	Code list	ISO 3166-1 alpha-3 codes. The flag country of the vessel.
3	Year	Integer	M	Code list	1900 to 3000
4	Month	Integer	M	Code list	1 to 12
5	C-square	String	M	Code list	0.05x0.05 degree, C-square reference XXXX:XXX:XXX:X
6	Vessel length category	String	M	Code list	"<12" (if available) "12-15" ">=15"
7	Gear code	String	M	Code list	DCF level 4*
8	Fishing activity category European lvl 6*	String	M	Code list	fishing activity category – it is recommended to submit DCF level 6*
9	Average fishing speed	Decimal numeral	M	1 to 50	Average fishing speed within the aggregation: year, month, c-square, vessel length category, gear code and DCF métier.
10	Fishing hour	Decimal numeral	M	1 to 9999999999	Fishing hour calculated from VMS data.
11	Average Vessel Length overall	Decimal numeral	M	1 to 200	Average vessel length within the aggregation: year, month, c-square, gear code and DCF métier .
12	Average kW	Decimal numeral	M	1 to 9999999999	Average vessel power (kW) within the aggregation: year, month, c-square, gear code and DCF métier .
13	kW*fishing hour	Decimal numeral	M	1 to 9999999999	
14	Tot weight	Decimal numeral	M	1 to 9999999999	Total landings of all species caught. In kg
15	Tot value	Decimal numeral	M	1 to 9999999999	Total value of all species caught. In Euro

M = mandatory. *DCF level = Fishing activity - Metier:

https://datacollection.jrc.ec.europa.eu/wordef/fishing-activity-metier

Annex 2: Exchange format for reporting Log book data

Order	Name	Type	Req.	Basic checks	Comments
1	Record type	String	M		Fixed value LE
2	Vessel Flag Country	String	M	Code list	ISO 3166-1 alpha-3 codes. The flag country of the vessel.
3	Year	Integer	M	Code list	1900 to 3000
4	Month	Integer	M	Code list	1 to 12
5	ICES statistical rectangle	String	M	Code list	Uppercase, e.g. 45F2
6	Gear code	String	M	Code list	DCF level 4*
7	All fishing activity category European lvl 6*	String	M	Code list	All fishing activity category – DCF level 6*
8	Vessel length category	String	M	Code list	Vessel length grouped into: "<12" "12-15" ">=15"
9	VMS enabled category	String	M	Code list	Yes/No
10	FishingDays	Decimal numeral	M	1 to 9999999999	Number of fishing days by ICES rectangle. If a vessel fished in several ICES squares one day, the day will be divided by the number of ICES rectangles.
11	kW*fishing days	Decimal numeral	M	1 to 9999999999	
12	Tot weight	Decimal numeral	M	1 to 9999999999	Total landings of all species caught. In kg
13	Tot value	Decimal numeral	M	1 to 9999999999	Total value of all species caught. In Euro

M = mandatory. *DCF level = Fishing activity - Metier:

https://datacollection.jrc.ec.europa.eu/wordef/fishing-activity-metier