Montserrat iVMS data: results of integration with landings data

Version	Date issued	Prepared by	Reviewers	Date to complete
Draft 1	30/07/17	Dan Edwards, (JNCC)	Alice Doyle (JNCC), Paul Brickle (SAERI), Paul Brewin (SAERI)	05/09/17
Final Draft	06/09/17	Dan Edwards		

Introduction

This report details the production of value maps for fisheries around Montserrat using novel iVMS (inshore vessel monitoring system) technology, using data gathered over the first year of deployment of iVMS hardware. The project is part of a territory to territory partnership between the Falklands Islands Government and the Government of Montserrat, facilitated and assisted by JNCC and funded by the UK government.

iVMS data

iVMS data were collected using a Succorfish M2M SC2 iVMS unit fitted with a solar powered battery charger intended to operate in harsh environments for prolonged periods of time with no maintenance. Units were set to record and report (ping) vessel position, course and speed every 15 minutes via the GSM network, with data being stored and maintained by Succorfish M2M. 7 units were installed on 7 fishing vessels working from Little Bay, Montserrat, and were operational over varying time periods between July 2015 and April 2017.

Landings Data

Landings data were collected via interview by Government of Montserrat staff and was available in the form of a monthly excel sheet covering the period between January 2016 and April 2017 (the iVMS unit installed and operating in 2015 was faulty and the data could not be used, so landings from 2015 were not required). Data were available for a total of 633 fishing trips by 23 vessels.

Method and tools

Data were processed in R using the VMStools package (Hintzen *et al* 2012). The script used is included in Appendix 1.

iVMS data were cleaned to remove impossible locations, speeds and headings, duplicates and pseudo-duplicates (instances where misconfigured iVMS units were reporting more frequently than every minute), points on land, and points where the vessel was in port (using a 200m radius of Little Bay port and moorings). The number of unique days when the vessel was away from port and travelling between 0 and 6 knots (typical fishing speeds) were counted to give an estimate of the total vessel activity over the period when iVMS units were operational.

The date format within the landings data was inconsistent and changed between UK (dd/mm/yyyy) and US (mm/dd/yyyyy) format. This was manually corrected in the original .xlsx workbook where each month's landings were contained on a different sheet and the intended date could be determined. A cleaning routine was then applied to identify and remove any duplicates entries or erroneously large landings values, and a unique ID was generated for each valid landing record. Total values were calculated using a unit value of $8xcd/lb for garfish and ballyhoo, and $10xcd for all other species. Recording of fishing start and end time within the landings data was inconsistent and so could not be used to refine the identification of fishing activity. Instead, it was assumed that all times on a fishing day were potential fishing activity. The landings data were then joined with the iVMS data based on the vessel ID and date (as time could not be used). Each unique day was classified as a "trip" and provided information on the fishing gear used that day. Once iVMS data was linked to a trip and associated gear type, fishing activity could then be separated from vessel transit activity based on speed thresholds, with different speeds being used to identify fishing activity with different gear types. Appropriate speed thresholds were selected through frequency analysis, where peaks in frequency of reported speeds were identified corresponding to different repeated activities (fishing, transiting, hauling/setting gear) which were subsequently validated using local knowledge. Appendix 2 gives the speed thresholds used. The landings for a fishing trip were then distributed equally among the iVMS pings considered to represent fishing activity during that trip.

Data were then aggregated to a 0.01 decimal degree grid for plotting and analysis, and data falling outside of the Montserrat EEZ (which included all gill net activity and all activity by vessel X1004) were removed from further analysis.

Results

Cleaning of iVMS data removed a total of 89,133 pings (Table 1), with the majority of these corresponding to pings in harbour. Data from vessel X1001 were corrupted as the speeds and headings were inconsistently rerecorded and only recorded at certain recurring values, most likely due to a faulty unit. As a result, these data could not be used in the final analysis. All of the data from vessel X1004 that was linked to landings originated outside of the Montserrat EEZ and so was also excluded from further analysis. There were no landings records with corresponding iVMS data for vessel X1006; the vessel had a functional iVMS unit installed for a total of only 24 days, and during that period the analysis of activity suggests that the vessel was only active away from port on 4 days, none of which were surveyed by the fisheries data collectors.

Table 1 iVMS cleaning routine results

	Rows remaining	% of total
total	98965	100
notPossible	98937	99.97
duplicates	98920	99.95
pseudoDuplicates	98722	99.75
OnLand	82047	82.91
InHarbour	9832	9.93
OutOfRegion	9788	9.89

Cleaning of landings data did not remove any records, leaving a total of 633 landings records for 23 vessels.

Table 2 Time coverage of iVMS data, estimated activity, and quantity of data linked to landings

	Start	Finish	Days iVMS operational	Days vessel active	Active days linked to landings	Vessel activity covered
Vessel X1001	2015-07-13	2016-11-20	123	17	0	0%
Vessel X1002	2016-08-26	2017-01-26	87	43	1	2%
Vessel X1003	2016-01-11	2017-01-11	109	69	41	59%
Vessel X1004	2016-09-24	2016-11-14	31	4	2	50%
Vessel X1005	2016-08-26	2016-12-13	91	26	15	58%
Vessel X1006	2016-09-26	2016-11-25	24	4	0	0%
Vessel X1007	2016-09-24	2017-04-22	130	39	26	67%

Table 2 presents the temporal coverage of iVMS data, total days iVMS unit was operating, number of days the vessel was assumed to be active away from port, and the number of fishing trips that had corresponding landings data after all data had been cleaned. iVMS units were operational for a total of **595** days, covering a potential **202** fishing trips. **85** of these trips had associated landings data, representing around **42%** of the estimated vessel effort within the sample.

Figure 1 presents the total value of landings linked to iVMS data by those vessels sampled, and Figure 2 shows landings per unit effort (average landings per fishing trip).

Table 3 Quantity of data collected by gear type

	Number of fishing days	Fishing hrs	% of total landings
Lines	22	183	13
Pot	31	79	12
Seines	28	77	17
Gillnet	3	12	82

The quantity of linked iVMS and landings data by gear type are presented in Table 3.

Appendix 3 contains maps of landings per unit of effort by individual vessel (figure 4) and by gear type (figure 5), as well as total landed values by vessel (figure 6).

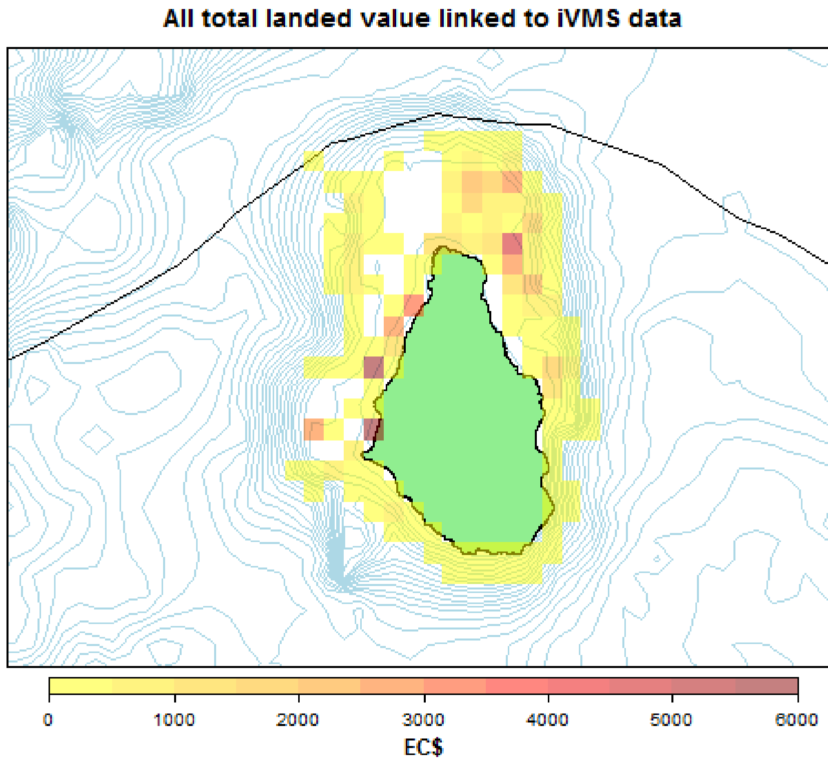

Figure 1 Total landings linked to iVMS

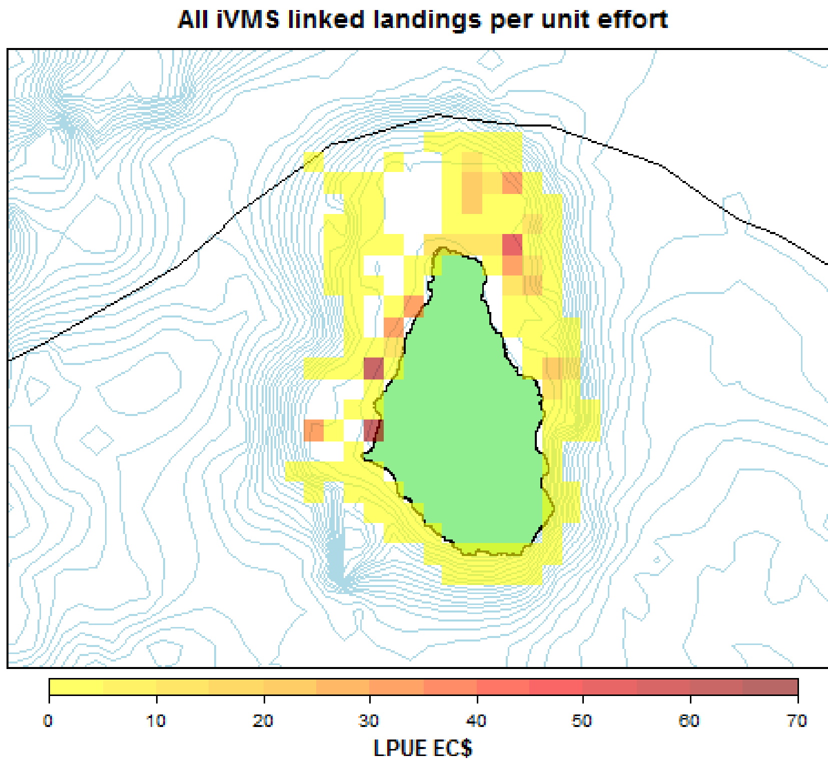

Figure 2 Landings per unit effort

Discussion/Limitations

Community validation
The data presented is a preliminary analysis that has not been validated by the local fishing community. Local knowledge can resolve many artefacts that may be present in the data that may not be identified due to limitations in the methodology, and the outputs of the analysis should not be considered finalised until such a review has been completed. For instance, analysis of the data suggests that the high cell values apparent on the west coast of Montserrat in data from Vessel X1003 and seine net layers (Appendix 3, figures 4 and 5) are valid and are a result of large catches of mainly garfish, however, until validated by the operator/fishing community it remains possible that this could be an artefact cause by, for example, repeatedly using the same area for fish processing.

Furthermore, if the results of an incomplete sample of fishing activity are likely to be extrapolated to the fleet level by marine planners, then it is important that fishers who have not participated in the spatial data collection have opportunity to analyse and comment on preliminary outputs to ensure that key areas supporting their livelihood are also represented somehow within the outputs.

In light of the voluntary nature of the current iVMS data collection programme, it is also important that agreement is sought with the fishing community as to the level of spatial resolution at which the data products are published in order to maintain trust. Too coarse a resolution and the data is of limited value for marine planning, but too fine a resolution may be commercially sensitive and may provide a disincentive for participation. These discussions about the outputs have yet to take place with the fishing community, but are anticipated to occur in early November 2017.

Limited data samples
The values presented are just a sample from those vessels from which both iVMS and landings data had been collected and could be matched. Landings data without corresponding iVMS data and iVMS data without corresponding landings data, have not been used. Due to issue with the iVMS (there were only 4 vessels collecting good quality iVMS data for more than 4 fishing days, and periods of coverage were intermittent) and the Government of Montserrat's fisheries data collection programme (analysis of iVMS suggests only 42% of fishing trips were sampled), the quantity of known activity data is quite limited, and this has influenced the outputs presented in this report. Data from vessel X1002 shows extremely high level of landings per unit effort, however, this is an artefact due to having landings data available for only 1 single fishing trip and values would be expected to fall quite significantly with a longer time series of data as the vessel's footprint increased. This can be addressed by deploying appropriate and effective iVMS hardware, and increasing the frequency that corresponding landings data collected.

A targeted approach towards recording landings from the iVMS enabled vessels should be taken in order to maximise the value returned from the deployment of the iVMS units, and this could be assisted by using the geofence notifications functionality available within the Succorfish software system to send emails to data collectors as vessels approach port after fishing.

The iVMS data set has a poor time coverage and iVMS units worked for inconsistent and intermittent durations, which complicates extrapolating this activity to the fleet level. Furthermore, fishing is a seasonal activity with changes in distribution and abundance of species, especially migratory ones. In order to build an accurate picture of fishing activity across the seasons that can effectively be extrapolated to fleet level then ideally at least 1 year of continuous coverage is required of both iVMS and landings from a well-designed representative sample of vessels.

Quality of sample surveyed
In the absence of an estimate of total fleet effort it is not possible to satisfactorily extrapolate the activities sampled to the fleet level. Without a detailed survey of total effort properly stratified by the different activity types within the fleet it is also unclear how representative the current sample of vessels is. It is noted however that stratification of the sample across fishing gears could be improved, and that not all gear types are covered (there are little and no landings for gill net and spear fishing linked to iVMS data), and the spatial coverage of pot fishing grounds is much less than anticipated and witnessed by the author. Validation with the fishing community should be able to inform an assessment of how well overall activity has been represented.

Landings sampling bias
There is some evidence of a strong bias in sampling of the landings surveyed. Vessel X1007 had landings recorded for 67% of the 39 days when the vessel was active away from port, whereas vessel X1002, apparently more active with 43 days away from port, had only 1 (2%) trip with associated landings data. It is possible that this vessel is actively engaged in tasks other than fishing (goat hunting, for example, or transporting goods), which could lead to an inflated estimate of vessel activity levels, but this is thought to be unlikely at the levels witnessed.

Furthermore, catches are generally estimated in the landings data collection process, and no assessment has been done on the accuracy of the estimates.

Fishing times
In the absence of accurate reporting of fishing start and finish times, it must be assumed that all vessel activity away from port on a day when a landing record has been generated is fishing activity. This may not be the case and the daily footprint of activity may be overestimated. However, it is expected that significantly of this occurring extensively on the same day as fishing activity will be relatively few.

Effective and accurate recording of fishing times as part of the standard landings data collection process would allow for a more refined measure of fishing effort as a standard trip time could be established.

Low confidence in gear type recorded
Multiple gears used in a day are not consistently or effectively recorded in the landings interview, and in many cases just the main gear used that day is reported. This results in an underestimate of the use minor gears that may be used alongside other more active/profitable gears within the same fishing trip. The clearest example of that here is with the gillnet fishery, which fails to appear in the outputs as anticipated. In the few cases where gill net fishing was recorded it was by a single vessel operating outside of the Montserrat EEZ, and so was removed from further analysis.

Landings data interviews do not discriminate between trolling for pelagic species and ground fishing for demersal species, and both are simply recorded as line fishing. As a result, in the current analysis, ocean pelagic landings have been attributed to demersal line fishing grounds and demersal fish landings attributed to open ocean pelagic fishing grounds. It would be possible given additional data processing time to correct for this by separating landings recording ocean pelagic species, applying additional speed filtering (trolling occurs at around 4-7 knots) or spatial filtering based on bathymetry and local knowledge.

Ideally data should be collected on all gears used on a fishing trip and the contribution to landings apportioned to each. However, it is recognised that this can be challenging and impractical to collect when landings are only recorded at the end of the trip and not throughout the day.

Data processing method- splitting daily catches
Daily catches have been divided equally amongst all locations recorded that day where the vessel reported typical fishing speeds. This fails to recognise the reality of heterogeneity across fishing grounds visited in a single day. This is a limitation of the data and could only properly be resolved by fishers collecting catch data at intervals throughout the day.

Data processing method- static gear effort
Identifying fishing effort through vessel activity is a poor method for estimating effort in static gear fisheries, where the quantity of gear and time it is deployed are the key components of fishing effort. However, in the absence of additional data for these parameters then vessel activity is the only available proxy.

Recommendations/Next steps

Increased iVMS coverage
At least one year of continuous data from a well-designed, representative sample of fishing vessels should be collected to ensure an accurate picture of fishing across seasons.

Improved coverage of landings data
Better coverage of vessel landings data is required to maximise the value of the iVMS data collected. It is recommended that steps are taken to increase the sampling frequency of landings from those vessels (improved training and a redesigned data collection framework). This could be facilitated in part by use of the alerts available through the Succorfish being set to automatically notify data collectors when vessels are approaching port.

Improved quality of landings data
The confidence in data collected during landings interviews is low, and it is recommended that steps identified by Brewin (2017) are followed to bring the quality of the data collected up to standard.

Fleet structure and effort
Regular surveys of fleet structure and effort are also recommended in order to allow for a better survey design and deployment of iVMS units across a representative sample of vessels, as well as enabling the effective extrapolation of mapped results (and landings) to the fleet level.

Data management
Landings data are currently digitised and stored only in excel sheet format. This has led to inconsistencies in data entry, particularly regarding date format, and has required some time to correct manually. The government of Montserrat should replace their own data holdings 2016-April 2017 with those corrected as part of this analysis. The use of a properly configured database would reduce some of this data cleaning and processing burden in future and should be adopted.

Static gear measurement
The use of vessel activity is a poor proxy for fishing effort in static gear fisheries. RFID attached to static gear and scanned when deployed and hauled offer the potential to quantify

both the amount of gear and the duration for which it is fished, giving a much more accurate understanding of landings per unit of effort than that presented here. The use of RFID tags and scanners may also offer potential for improving discrimination between effort and catch in the troll fishery for ocean pelagic species.

References

Brewin, P, 2017, Fishery data collection and integration strategy for underpinning sustainable fisheries management in Montserrat, South Atlantic Environmental Research Institute.

Hintzen, N. T., Bastardie, F., Beare, D., Piet, G. J., Ulrich, C., Deporte, N., et al. (2012). VMS tools: Open-source software for the processing, analysis and visualisation of fisheries logbook and VMS data. Fisheries Research, 115–116, 31–43

Appendix 1

```
# Script to extract and process iVMS and landings data for Montserrat
#
# By: Dan Edwards
# Code by: Dan Edwards
# Contact: dan.edwards@jncc.gov.uk
# Date: August 2017
#

# Prep workspace ----------------------------------------------------------

rm(list=ls())

#Load necessary packages and functions- add required packages to vector
necessary <-
c("cluster","data.table","doBy","maps","mapdata","maptools","PBSmapping","s
p", "shapefiles", "rgdal", "raster", "GISTools", "classInt", "svIO",
"R2HTML", "svIDE", "svSocket", "utils", "Matrix", "vmstools", "tidyverse",
"xlsx", "mixtools", "raster", "tableHTML")

installed <- necessary %in% installed.packages()[, 'Package']
if (length(necessary[!installed]) >=1)
  install.packages(necessary[!installed], dep = T)

library(vmstools)
library(Matrix)
library(rgdal)
library(maptools)
library(plyr)
library(tcltk)
library(data.table)
library(ggplot2)
library(raster)
library(dplyr)
library(RColorBrewer)
library(rgeos)
library(tableHTML)

#set wd and create directories
setwd("D:/1_wd")
dir.create("inputData")
dir.create("outputs")
dir.create("working")


# Load shapefiles used ----------------------------------------------------
-----------

#read MNI shapefile to mask raster to EEZ
mniEEZ <- readOGR( dsn= "InputData/shapefiles", "MontserratEEZ" )
#load bathymetery for plot
bathy <- readOGR("InputData/shapefiles", "Bathymetry Contours")
#read in land shapefile
mni <- readOGR("InputData/shapefiles", "mni")


# Load iVMS data ----------------------------------------------------------
-----
```

```
tacsat <- read.csv("InputData/SC2tacsat.csv")
tacsat <- formatTacsat(tacsat)


# CLEAN IVMS DATA ---------------------------------------------------------

#save removed records
remrecsTacsat     <-
matrix(NA,nrow=7,ncol=2,dimnames=list(c("total","notPossible",
"duplicates","pseudoDuplicates","OnLand",
"InHarbour","OutOfRegion"),c("rows","percentage")))
remrecsTacsat["total",] <- c(nrow(tacsat),"100%")

#remove impossible points####
#points not on the globe
idx <- which(abs(tacsat$SI_LATI) > 90 | abs(tacsat$SI_LONG) > 180)

#adding points with heading outside compass range
idx <- unique(c(idx,which(tacsat$SI_HE < 0 | tacsat$SI_HE >  360)))

#add points with unreasonable speeds (>25knots)
idx <- unique(c(idx,which(tacsat$SI_SP > 25)))
length(idx)
if(length(idx)>0) tacsat <- tacsat[-idx,]

remrecsTacsat["notPossible",] <- c(nrow(tacsat),100+round((nrow(tacsat) -
an(remrecsTacsat["total",1]))/an(remrecsTacsat["total",1])*100,2))

remrecsTacsat

#check for duplicates####
#create one date-time stamp
tacsat$SI_DATIM <- as.POSIXct(paste(tacsat$SI_DATE,tacsat$SI_TIME,sep=" "),

                              tz="GMT", format="%d/%m/%Y  %H:%M")
#get records as a string to easily check for duplicates
uniqueTacsat    <-
paste(tacsat$VE_REF,tacsat$SI_LATI,tacsat$SI_LONG,tacsat$SI_DATIM)
tacsat          <- tacsat[!duplicated(uniqueTacsat),] #get rid of the
duplicates
print(nrow(tacsat))

remrecsTacsat["duplicates",] <- c(nrow(tacsat),100+round((nrow(tacsat) -
an(remrecsTacsat["total",1]))/an(remrecsTacsat["total",1])*100,2))

remrecsTacsat

#remove pseudo duplicates ####
#- Remove points which are pseudo duplicates as they have an interval rate
< x minutes- some units were pinging much too fast
intThres <- 0.5    #Minimum difference in time interval in minutes to
prevent pseudo duplicates
intvThres <- 30 # Maximum difference in time interval in minutes to prevent
intervals being too large to be realistic##### Any pings higher than this
are reset to this value #

tacsat          <- sortTacsat(tacsat)
tacsatp         <- intervalTacsat(tacsat,level="vessel",fill.na=T)
```

```
#tacsat          <- tacsatp[which(tacsatp$INTV > intThres |
is.na(tacsatp$INTV)==T),-grep("INTV",colnames(tacsatp))] #doesnt wrtite
intv
tacsat          <- tacsatp[which(tacsatp$INTV > intThres |
is.na(tacsatp$INTV)==T),]

#reset high pings to intvThres
tacsatp$INTV[tacsatp$INTV> intvThres] <- intvThres

#keep all pings above 3min intThres
tacsat <- tacsatp[which(tacsatp$INTV > intThres),]

remrecsTacsat["pseudoDuplicates",] <-
c(nrow(tacsat),100+round((nrow(tacsat) -
an(remrecsTacsat["total",1]))/an(remrecsTacsat["total",1])*100,2))

remrecsTacsat

tacsatBackup <- tacsat
tacsat <- tacsatBackup


# Operational Dates -------------------------------------------------------

#Measure length of installation/operation on vessel ####

tacsat$SI_DATE <- as.Date(tacsat$SI_DATE, format = "%d/%m/%Y")

vessel01 <- range(subset(tacsat, VE_REF == "1001")$SI_DATE)
vessel02 <- range(subset(tacsat, VE_REF == "1002")$SI_DATE)
vessel03 <- range(subset(tacsat, VE_REF == "1003")$SI_DATE)
vessel04 <- range(subset(tacsat, VE_REF == "1004")$SI_DATE)
vessel05 <- range(subset(tacsat, VE_REF == "1005")$SI_DATE)
vessel06 <- range(subset(tacsat, VE_REF == "1006")$SI_DATE)
vessel07 <- range(subset(tacsat, VE_REF == "1007")$SI_DATE)
operational <- data.frame(vessel01, vessel02, vessel03, vessel04, vessel05,
vessel06, vessel07)
operational <- as.data.frame(t(operational))
names(operational) <- c("Start", "Finish")

#Calculate range of operation duration ####
operational$DateRange <- difftime(operational$Finish, operational$Start,
units = "days")
operational$DateRange <- round(operational$DateRange, 0)

# Days working ####
v1 <- length(unique(subset(tacsat, VE_REF == "1001")$SI_DATE))
v2 <- length(unique(subset(tacsat, VE_REF == "1002")$SI_DATE))
v3 <- length(unique(subset(tacsat, VE_REF == "1003")$SI_DATE))
v4 <- length(unique(subset(tacsat, VE_REF == "1004")$SI_DATE))
v5 <- length(unique(subset(tacsat, VE_REF == "1005")$SI_DATE))
v6 <- length(unique(subset(tacsat, VE_REF == "1006")$SI_DATE))
v7 <- length(unique(subset(tacsat, VE_REF == "1007")$SI_DATE))
operational$DaysiVMSOperating <- c(v1,v2,v3,v4,v5,v6,v7)
operational


# Clip Land -------------------------------------------------------

#make class Spatial Polygons for PointOnLand
mni <- as(mni, "SpatialPolygons")
```

```
##Find out prj4string of land shapefile for function below
proj4string(mni)

#index points on land
idx <- pointOnLand(tacsat, mni, proj4string = CRS("+proj=longlat
+datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"))

#save points on land to check if wanted
pol <- tacsat[which(idx == 1),]

#make tacsat points NOT on land
tacsat          <- tacsat[which(idx == 0),]
remrecsTacsat["OnLand",] <- c(nrow(tacsat),100+round((nrow(tacsat) -
an(remrecsTacsat["total",1]))/an(remrecsTacsat["total",1])*100,2))

remrecsTacsat


# Clip harbours -------------------------------------------------------

#Create Harbour Data for Little bay for Point in Harbour Function.
harbour <- c("LittleBayMooring", "LittleBayPier") #names of landing spots
lon <- as.numeric(c("-62.2078", "-62.2059")) #longitudes
lat <- as.numeric(c("16.8031","16.8033"))#latitudes
range <- as.numeric(c("0.2", "0.2"))#range/size of harbour in km
harbours <- data.frame(harbour, lon, lat, range)

#indes points in harbour
idx             <- pointInHarbour(tacsat$SI_LONG,tacsat$SI_LATI,harbours)

#save points in harbour if wanted
pih <- tacsat[which(idx == 1),]

#make tacsat only points not in harbour
tacsat          <- tacsat[which(idx == 0),]
remrecsTacsat["InHarbour",] <- c(nrow(tacsat),100+round((nrow(tacsat) -
an(remrecsTacsat["total",1]))/an(remrecsTacsat["total",1])*100,2))
save(remrecsTacsat,file="working/remrecsTacsat.RData")

#write out points in harbour and on land to check
NotAtSea <- rbind(pih, pol)
write.csv(NotAtSea,"working/OnLandInHarbour.csv")



#some problems with iVMS tracking en route from UK to MNI.  Remove all
pings outside MNI region ####

idx <- which(tacsat$SI_LATI > 17 | tacsat$SI_LATI < 16.5 | tacsat$SI_LONG <
-62.5 | tacsat$SI_LONG > -61.9)


length(idx)
if(length(idx)>0) tacsat <- tacsat[-idx,]

remrecsTacsat["OutOfRegion",] <- c(nrow(tacsat),100+round((nrow(tacsat) -
an(remrecsTacsat["total",1]))/an(remrecsTacsat["total",1])*100,2))
```

```r
remrecsTacsat


save(tacsat, file = "working/tacsat.RData")
tacsatBackup2 <- tacsat
remrecsTacsat
print("Tacsat cleaning is done!")


# Check possible fishing days ---------------------------------------------
--

#Indicatation of total effort?  Select only fishing speeds 1- 6 knots.
Coarse test.  Count unique days
vess1 <- length(unique(subset(tacsat, VE_REF == "1001"  & SI_SP <6 & SI_SP
> 1)$SI_DATE))
vess2 <- length(unique(subset(tacsat, VE_REF == "1002"  & SI_SP <6 & SI_SP
> 1)$SI_DATE))
vess3 <- length(unique(subset(tacsat, VE_REF == "1003"  & SI_SP <6 & SI_SP
> 1)$SI_DATE))
vess4 <- length(unique(subset(tacsat, VE_REF == "1004"  & SI_SP <6 & SI_SP
> 1)$SI_DATE))
vess5 <- length(unique(subset(tacsat, VE_REF == "1005"  & SI_SP <6 & SI_SP
> 1)$SI_DATE))
vess6 <- length(unique(subset(tacsat, VE_REF == "1006"  & SI_SP <6 & SI_SP
> 1)$SI_DATE))
vess7 <- length(unique(subset(tacsat, VE_REF == "1007"  & SI_SP <6 & SI_SP
> 1)$SI_DATE))
UnlinkedDays <- c(vess1, vess2, vess3, vess4, vess5, vess6, vess7)
operational$VesselActive <- UnlinkedDays

#add results to summary table
operational$landingsLinkedDays <- "NA"
operational


print(paste("DaysiVMSworking = ", sum(operational$DaysiVMSOperating), sep =
" "))
print(paste("Possible fishing days =", sum(operational$VesselActive), sep
=" "))


# LANDINGS DATA -----------------------------------------------------------

# require(XLConnect)
# landWB <- loadWorkbook("InputData/landings16_17.xlsx")
# sheet <- getSheets(landWB)
# for (i in 1:length(sheet)){
#   tmp <- read.xlsx("InputData/landings16_17.xlsx", sheetIndex = i,
#                     sheetName = NULL, colIndex = NULL, startRow = NULL,
endRow = NULL,
#                     as.data.frame = TRUE, header = T)
#   if (i == 1) landings <- tmp else landings <- rbind.fill(landings, tmp)
# }
#
# DATES IN LANDINGS DATA VERY INCONSISTENT - UK & US formats and lots of
errors.
# The quickest way to fix this is to inspect and re-enter the data sheet by
sheet in excel
# Write out to a seperate csv for each month/worksheet
```

```r
# Load and combine landings data -----------------------------------------

fileNames <- list.files("InputData/landings")
#combine
landings <- do.call(rbind.fill, lapply(paste("InputData/landings/",
fileNames, sep = ""), read.csv, na.strings=c(""," ","NA")))
#write file for backup
write.csv(landings, file = "working/mergedLandings.csv")

# Rename columns to be eflalo compatibe
landings = landings %>% rename( FT_LHAR = Landing.Site,  VE_NAME =
Vessel.Name, VE_REG = Vessel.Registration.., NO_CREW = Number.of.Crew,
LE_GEAR = Gear.Type, FISHERY = Fishery.Type, FT_DTIME = Start.Time,
FT_LTIME = End.Time, LE_CDAT = Start.Date, LE_EDAT = End.Date, AREA =
Fishing.Area, LE_KG_BLV = Queen.Triggerfish, LE_KG_AQO = Blue.Tang,
LE_KG_AQH = Doctorfish, LE_KG_EEU = Red.Hind, LE_KG_HCZ =
Long.Jawed.Squirrelfish, LE_KG_CFJ = Coney, LE_KG_PWT =
Stoplight.Parrotfish, LE_KG_RSN = Red.Snapper, LE_KG_LJJ = Dog.Snapper,
LE_KG_SNY = Yellowtail.Snapper, LE_KG_LJM = Mahogany.Snapper, LE_KG_LJU =
Blackfin.Snapper, LE_KG_RBY = Rock.Beauty, LE_KG_CDB = Porgy.Jolthead,
LE_KG_SKH = Shark, LE_KG_LJN = Mutton.Snapper, LE_KG_NCY =
Honeycomb.Cowfish, LE_KG_HLC = Caeser.Grunt, LE_KG_GPN = Nassau.Grouper,
LE_KG_UDU = Spotted.Goatfish, LE_KG_KYS = Bermuda.Chub, LE_KG_SLC =
Carribean.Spiny.Lobster, LE_KG_HNU = Black.Margate, LE_KG_ANW =
Queen.Angelfish, LE_KG_GRA = Gray.Angelfish, LE_KG_FRA = French.Angelfish,
LE_KG_BAR = Barracuda, LE_KG_CVJ = Crevalle.Jack, LE_KG_CXR = Bar.Jack,
LE_KG_GAR = Gar, LE_KG_AJK = Comad, LE_KG_LTJ = Silk.Snapper, LE_KG_NXU =
Black.Jack, LE_KG_SNL = Lane.Snapper, LE_KG_HNR = Porkfish.Grunt, LE_KG_LJG
= Mutton.Grouper, LE_KG_RPU = Vermillion.Snapper, LE_KG_LIO = Lionfish,
LE_KG_TUN = Tuna, LE_KG_MAB = Black.Grouper, LE_KG_DOX = Dolphin, LE_KG_WAH
= Wahoo,LE_KG_NBR = Yellow.Jack, LE_KG_USP = Creole.Wrasse, LE_KG_LJI =
Grey.Snapper, LE_KG_HLU = White.Margate, LE_KG_RRU = Rainbow.Runner,
LE_KG_CZT = Ocean.Trigger, LE_KG_EEE = Queen.Snapper, LE_KG_GGG = Graysby,
LE_KG_MWP = Sand.tilefish, LE_KG_WSK = White.Shark, LE_KG_PLM = Palometa,
LE_KG_BOF = Bonefish, LE_KG_SDF = Scrawled.filefish, LE_KG_ASX =
Black.Snapper, LE_KG_TRE = Skip.Jacks, LE_KG_USN = Princess.Parrot,
LE_KG_HTU = Glasseye.Snapper, LE_KG_HLV = French.Grunt, LE_KG_LJP =
Schoolmaster, LE_KG_CER = Mackerel, LE_KG_USU = Blue.Parrot , LE_KG_BZX =
Bonita, LE_KG_BAL = Ballyhoo, LE_KG_NXL = Horseye.Jack, LE_KG_EFD =
Rockhind, LE_KG_MGP = Mango.Snapper, LE_KG_KGF = Kingfish, LE_KG_STP =
Spotlight.Parrotfish, LE_KG_QAF = Queen.Angelfsih, LE_KG_EEO = Bream,
LE_KG_BGI = Bungi, LE_KG_MOA = Moonfish, mackerel_CER = Mackeral,
blueParrot_USU = Blue.parrot, scrwFilefish_SDF = Scrawled.Filefish,
LE_KG_BDR =Spanish.Hogfish , LE_KG_RSY = Redtail.Parrot, LE_KG_BSJ
=Bigeye.Scad..Jacks. , LE_KG_LOB = Lobster, AlmacoComad_AJK =
Almaco.Jack..comad., LE_KG_RGR = Red.Grouper, LE_KG_MOR = Moray, LE_KG_MIX
= Mixed.Fishes, LE_KG_RBP = Redband.Parrotfish, LE_KG_WFF = White.Filefish,
QueenSnapperEEE = Queen.snapper..bream., VermillionSnapper_bungi_RPU_BGI =
Vermillion.Snapper..bungi., Porkfish_HNR = Porkfish, LE_KG_BDN =
Black.Durgon, LE_KG_TAL = Tailor, LE_KG_WSF = Whitespotted.Filefish,
LE_KG_UNI = Unicorn.Filefish, OceanTrigger_CZT = Ocean.Triggerfish,
princess_USN = Princess.Parrotfish, BlueParrot_USU= Blue.Parrotfish,
LE_KG_GAJ= Greater.Amberjack, O_trigger_CZT = Ocean.triggerfish, LE_KG_WGT
= White.Grunt, wahoo_WAH = Wahoo.1, comad_AJK = Almaco.Jack..Comad.)

# consolidate duplicated columns ####
landings$LE_KG_CER <- rowSums(landings[, c("LE_KG_CER", "mackerel_CER")],
na.rm= TRUE)
landings$LE_KG_USU <- rowSums(landings[, c("LE_KG_USU",
"blueParrot_USU","BlueParrot_USU" )], na.rm= TRUE)
```

```
landings$LE_KG_SDF <- rowSums(landings[, c("LE_KG_SDF", "scrwFilefish_SDF"
)], na.rm= TRUE)
landings$LE_KG_AJK <- rowSums(landings[, c("LE_KG_AJK", "AlmacoComad_AJK",
"comad_AJK" )], na.rm= TRUE)
landings$LE_KG_EEE <- rowSums(landings[, c("LE_KG_EEE", "QueenSnapperEEE"
)], na.rm= TRUE)
landings$LE_KG_RPU <- rowSums(landings[, c("LE_KG_RPU",
"VermillionSnapper_bungi_RPU_BGI" )], na.rm= TRUE)
landings$LE_KG_HNR<- rowSums(landings[, c("LE_KG_HNR", "Porkfish_HNR" )],
na.rm= TRUE)
landings$LE_KG_CZT<- rowSums(landings[, c("LE_KG_CZT", "OceanTrigger_CZT",
"O_trigger_CZT"  )], na.rm= TRUE)
landings$LE_KG_USN<- rowSums(landings[, c("LE_KG_USN", "princess_USN" )],
na.rm= TRUE)
landings$LE_KG_USU<- rowSums(landings[, c("LE_KG_USU", "BlueParrot_USU" )],
na.rm= TRUE)
landings$LE_KG_WAH<- rowSums(landings[, c("LE_KG_WAH", "wahoo_WAH" )],
na.rm= TRUE)
landings$NO_CREW <- rowSums(landings[, c("NO_CREW", "Number.Of.Crew")],
na.rm= TRUE)

# drop unused colomns and reorder
landings <- landings[, c("Data.Officer", "Date", "FT_LHAR", "Fisher",
"VE_NAME", "VE_REG", "NO_CREW", "LE_GEAR", "FISHERY", "FT_DTIME",
"FT_LTIME", "LE_CDAT", "LE_EDAT", "AREA","LE_KG_BLV", "LE_KG_AQO",
"LE_KG_AQH", "LE_KG_EEU", "LE_KG_HCZ", "LE_KG_CFJ", "LE_KG_PWT",
"LE_KG_RSN", "LE_KG_LJJ", "LE_KG_SNY", "LE_KG_LJM", "LE_KG_LJU",
"LE_KG_RBY", "LE_KG_CDB", "LE_KG_SKH", "LE_KG_LJN", "LE_KG_NCY",
"LE_KG_HLC", "LE_KG_GPN", "LE_KG_UDU", "LE_KG_KYS", "LE_KG_SLC",
"LE_KG_HNU", "LE_KG_ANW", "LE_KG_GRA", "LE_KG_FRA",
"LE_KG_BAR","LE_KG_CVJ", "LE_KG_CXR", "LE_KG_GAR", "LE_KG_AJK",
"LE_KG_LTJ", "LE_KG_NXU", "LE_KG_SNL", "LE_KG_HNR", "LE_KG_LJG",
"LE_KG_RPU", "LE_KG_LIO", "LE_KG_TUN", "LE_KG_MAB", "LE_KG_DOX",
"LE_KG_WAH","LE_KG_NBR", "LE_KG_USP", "LE_KG_LJI", "LE_KG_HLU",
"LE_KG_RRU", "LE_KG_CZT", "LE_KG_EEE", "LE_KG_GGG", "LE_KG_MWP",
"LE_KG_WSK", "LE_KG_PLM", "LE_KG_BOF", "LE_KG_SDF", "LE_KG_ASX",
"LE_KG_TRE", "LE_KG_USN", "LE_KG_HTU", "LE_KG_HLV", "LE_KG_LJP",
"LE_KG_CER", "LE_KG_USU", "LE_KG_BZX", "LE_KG_BAL", "LE_KG_NXL",
"LE_KG_EFD", "LE_KG_MGP", "LE_KG_KGF", "LE_KG_STP", "LE_KG_QAF",
"LE_KG_EEO", "LE_KG_BGI", "LE_KG_MOA", "LE_KG_BDR", "LE_KG_RSY",
"LE_KG_BSJ", "LE_KG_LOB", "LE_KG_RGR", "LE_KG_MOR", "LE_KG_MIX",
"LE_KG_RBP","LE_KG_WFF", "LE_KG_BDN", "LE_KG_TAL", "LE_KG_WSF",
"LE_KG_UNI", "LE_KG_GAJ",  "LE_KG_WGT")]


# Add in EFLALO columns ####
#create a unique ID for each fishing record
landings$FT_REF <- c(10001:10592)
#Add in some other columns that are part of the EFLALO format and are used
in various VMStools functions
landings$FT_DCOU <- "MNI"
landings$FT_DHAR <- "MSLTB"
landings$FT_LHAR <- "MSLTB"
landings$LE_ID <- landings$FT_REF
landings$LE_SLAT <- "-62.2"
landings$LE_SLON <- "16.8"
landings$LE_ELAT <- "-62.2"
landings$LE_ELON <- "16.8"
landings$LE_MSZ <- "10"
landings$LE_RECT <- "45F05"
landings_LE_DIV <- "IV"
```

```r
landings$VE_LEN <- "10"
landings$VE_KW <- "10"
landings$FT_LCOU <- "MNI"
landings$LE_DIV <- "IV"
landings$LE_KG_ALL <- "0"
landings$LE_EURO_ALL <- "0"
landings$VE_COU <- "MNI"
landings$VE_FLT <- "MNI"
#dummy data above to make vmstools functions work

#date data in landings record isn't very good- lots of entry error.  Just
use single "start date" (LE_CDAT) column for consistency
landings$FT_DDAT <- landings$LE_CDAT
landings$LE_EDAT <- landings$LE_CDAT
landings$FT_LDAT <- landings$LE_CDAT

# change departure and arrival times ####
#(too many blank fields resulted in too mch data being lost.  In absesnce
of departuira and arrival data just assume all activity on that day is
fishing activity )
landings$FT_DTIME <- "00:01"
landings$FT_LTIME <- "23:59"

# give vessels Anon VE_REF ####
#create lookup table of names and anonymous reference
vessNames <- unique(landings$VE_NAME)
iVMSvessNames <- c("Ann Elizabeth", "Daily Bread", "Experience", "In God We
Trust", "Optimum", "Third choice", "Try Me")
iVMSid <- c("1001", "1002", "1003", "1004", "1005", "1006", "1007")
ixx <- as.character(c(1:7))
ids <- as.data.frame(iVMSid, ixx)
ids <- cbind(ids,  iVMSvessNames)

#Populate VE_REF with vesselcode.  match based on name
landings$VE_REF <- "NA"
landings$VE_REF <- ids$iVMSid[match(landings$VE_NAME, ids$iVMSvessNames)]

#Format landings ####
landings <- formatEflalo(landings)
str(landings)

#Create totals columns ####
#Sum landings lbs
landings$LE_KG_ALL <- rowSums(landings[, c("LE_KG_BLV", "LE_KG_AQO",
"LE_KG_AQH", "LE_KG_EEU", "LE_KG_HCZ", "LE_KG_CFJ", "LE_KG_PWT",
"LE_KG_RSN", "LE_KG_LJJ", "LE_KG_SNY", "LE_KG_LJM", "LE_KG_LJU",
"LE_KG_RBY", "LE_KG_CDB", "LE_KG_SKH", "LE_KG_LJN", "LE_KG_NCY",
"LE_KG_HLC", "LE_KG_GPN", "LE_KG_UDU", "LE_KG_KYS", "LE_KG_SLC",
"LE_KG_HNU", "LE_KG_ANW", "LE_KG_GRA", "LE_KG_FRA",
"LE_KG_BAR","LE_KG_CVJ", "LE_KG_CXR", "LE_KG_GAR", "LE_KG_AJK",
"LE_KG_LTJ", "LE_KG_NXU", "LE_KG_SNL", "LE_KG_HNR", "LE_KG_LJG",
"LE_KG_RPU", "LE_KG_LIO", "LE_KG_TUN", "LE_KG_MAB", "LE_KG_DOX",
"LE_KG_WAH","LE_KG_NBR", "LE_KG_USP", "LE_KG_LJI", "LE_KG_HLU",
"LE_KG_RRU", "LE_KG_CZT", "LE_KG_EEE", "LE_KG_GGG", "LE_KG_MWP",
"LE_KG_WSK", "LE_KG_PLM", "LE_KG_BOF", "LE_KG_SDF", "LE_KG_ASX",
"LE_KG_TRE", "LE_KG_USN", "LE_KG_HTU", "LE_KG_HLV", "LE_KG_LJP",
"LE_KG_CER", "LE_KG_USU", "LE_KG_BZX", "LE_KG_BAL", "LE_KG_NXL",
"LE_KG_EFD", "LE_KG_MGP", "LE_KG_KGF", "LE_KG_STP", "LE_KG_QAF",
"LE_KG_EEO", "LE_KG_BGI", "LE_KG_MOA", "LE_KG_BDR", "LE_KG_RSY",
"LE_KG_BSJ", "LE_KG_LOB", "LE_KG_RGR", "LE_KG_MOR", "LE_KG_MIX",
```

```
"LE_KG_RBP","LE_KG_WFF", "LE_KG_BDN", "LE_KG_TAL", "LE_KG_WSF",
"LE_KG_UNI", "LE_KG_GAJ",  "LE_KG_WGT")],na.rm = TRUE)

#Add sum of prime fish $10/lb
landings$lbs_AAA <- rowSums(landings [, c("LE_KG_BLV", "LE_KG_AQO",
"LE_KG_AQH", "LE_KG_EEU", "LE_KG_HCZ", "LE_KG_CFJ", "LE_KG_PWT",
"LE_KG_RSN", "LE_KG_LJJ", "LE_KG_SNY", "LE_KG_LJM", "LE_KG_LJU",
"LE_KG_RBY", "LE_KG_CDB", "LE_KG_SKH", "LE_KG_LJN", "LE_KG_NCY",
"LE_KG_HLC", "LE_KG_GPN", "LE_KG_UDU", "LE_KG_KYS", "LE_KG_SLC",
"LE_KG_HNU", "LE_KG_ANW", "LE_KG_GRA", "LE_KG_FRA", "LE_KG_BAR",
"LE_KG_CVJ", "LE_KG_CXR", "LE_KG_AJK", "LE_KG_LTJ", "LE_KG_NXU",
"LE_KG_SNL", "LE_KG_HNR", "LE_KG_LJG", "LE_KG_RPU", "LE_KG_LIO",
"LE_KG_TUN", "LE_KG_MAB",  "LE_KG_DOX", "LE_KG_WAH", "LE_KG_NBR",
"LE_KG_USP", "LE_KG_LJI", "LE_KG_HLU","LE_KG_RRU", "LE_KG_CZT",
"LE_KG_EEE", "LE_KG_GGG", "LE_KG_MWP", "LE_KG_WSK","LE_KG_PLM","LE_KG_BOF",
"LE_KG_SDF", "LE_KG_ASX", "LE_KG_TRE", "LE_KG_USN", "LE_KG_HTU",
"LE_KG_HLV", "LE_KG_LJP", "LE_KG_CER", "LE_KG_USU", "LE_KG_BZX",
"LE_KG_NXL", "LE_KG_EFD", "LE_KG_MGP", "LE_KG_KGF", "LE_KG_STP",
"LE_KG_QAF","LE_KG_EEO", "LE_KG_BGI","LE_KG_MOA", "LE_KG_BDR", "LE_KG_RSY",
"LE_KG_BSJ", "LE_KG_LOB", "LE_KG_RGR", "LE_KG_MOR", "LE_KG_MIX",
"LE_KG_RBP", "LE_KG_WFF", "LE_KG_BDN", "LE_KG_TAL", "LE_KG_WSF",
"LE_KG_UNI", "LE_KG_GAJ", "LE_KG_WGT")], na.rm = TRUE)

#add sum of small pelagic ($8/lb)
landings$lbs_BBB <- rowSums(landings[, c("LE_KG_GAR", "LE_KG_BAL")], na.rm
= TRUE)

#create value columns (XCD) ####
landings$LE_EURO_AAA <- landings$lbs_AAA * 10
landings$LE_EURO_BBB <- landings$lbs_BBB * 8
landings$LE_EURO_ALL <- rowSums(landings[,c("LE_EURO_AAA", "LE_EURO_BBB")],
na.rm = TRUE)


# Fix issues with gear columns ####
#check gears
landings$LE_GEAR <- as.factor(landings$LE_GEAR)
summary(landings$LE_GEAR)

# FIx problems
landings$LE_GEAR[landings$LE_GEAR == "B.S. Net"] <- "B.S.Net"
landings$LE_GEAR[landings$LE_GEAR == "Spear Fishing"] <- "Spear fishing"
#drop now unused levels
landings$LE_GEAR <- droplevels(landings$LE_GEAR)
# re check
summary(landings$LE_GEAR)

#write out to use in QA
write.csv(landings, "working/landingsQA.csv")

# Re order landings data ####
MNIlandings <- landings[,c("Data.Officer", "VE_REF","VE_NAME","VE_REG",
"VE_LEN", "VE_KW", "VE_COU", "VE_FLT","Fisher", "NO_CREW",
"LE_GEAR","FISHERY", "AREA", "LE_ID", "FT_REF", "FT_DCOU", "FT_DHAR",
"FT_DDAT", "FT_DTIME", "FT_LTIME", "LE_CDAT", "FT_LHAR", "FT_LCOU",
"FT_LDAT", "LE_SLON", "LE_SLAT", "LE_ELON", "LE_ELAT","LE_EDAT", "LE_MSZ",
"LE_RECT", "LE_DIV", "LE_KG_ALL","LE_EURO_ALL" )]

#Now into eflalo Format
eflalo <- formatEflalo(MNIlandings)
```

```
# create date-time for departure and landing
eflalo$FT_DDATIM <- as.POSIXct(paste(eflalo$FT_DDAT,eflalo$FT_DTIME, sep =
" "), tz = "GMT", format = "%d/%m/%Y  %H:%M")
eflalo$FT_LDATIM <- as.POSIXct(paste(eflalo$FT_LDAT,eflalo$FT_LTIME, sep =
" "), tz = "GMT", format = "%d/%m/%Y  %H:%M")


# Clean Eflalo -------------------------------------------------------------

#keep track of removed records
remrecsEflalo     <-
matrix(NA,nrow=2,ncol=2,dimnames=list(c("total","duplicated"),c("rows","per
centage")))
remrecsEflalo["total",] <- c(nrow(eflalo),"100%")

#correct any overly large landings records ####

lanThres      <- 1.5  #Maximum difference in log10-transformed sorted
weights

#order columns #
# Put eflalo in order of 'non kg/eur' columns, then kg columns, then eur
columns
idxkg            <- grep("LE_KG_",colnames(eflalo))
idxeur           <- grep("LE_EURO_",colnames(eflalo))
idxoth           <- which(!(1:ncol(eflalo)) %in% c(idxkg,idxeur))
eflalo           <- eflalo[,c(idxoth,idxkg,idxeur)]

#get the species names
specs            <-
substr(colnames(eflalo[grep("KG",colnames(eflalo))]),7,9)

#Define  maximum allowed catch is per species (errors/outliers)
specBounds       <- lapply(as.list(specs),function(x){
  idx   <-
grep(x,colnames(eflalo))[grep("KG",colnames(eflalo)[grep(x,colnames(eflalo)
)])];
  wgh   <- sort(unique(eflalo[which(eflalo[,idx]>0),idx]));
  difw  <- diff(log10(wgh));
  return(ifelse(any(difw > lanThres),wgh[rev(which(difw <=
lanThres)+1)],ifelse(length(wgh)==0,0,max(wgh,na.rm=T))))})

#Make a list of the species names and the cut-off points / error / outlier
point
specBounds       <- cbind(specs,unlist(specBounds));

#Put these values to zero
specBounds[which(is.na(specBounds[,2])==T),2] <- "0"

#Get the index (column number) of each of the species
idx              <- unlist(lapply(as.list(specs),function(x){
  idx              <-
grep(x,colnames(eflalo))[grep("KG",colnames(eflalo)[grep(x,colnames(eflalo)
)])];
  return(idx)}))

#If landing > cut-off turn it into an 'NA'
warns            <- list()
fixWarns         <- TRUE #set function to auto correct or not
for(iSpec in idx){
  if(length(which(eflalo[,iSpec] > an(specBounds[(iSpec-idx[1]+1),2])))>0){
```

```
    warns[[iSpec]] <- which(eflalo[,iSpec] > an(specBounds[(iSpec-
idx[1]+1),2]))
    if(fixWarns){
      eflalo[which(eflalo[,iSpec] > an(specBounds[(iSpec-
idx[1]+1),2])),iSpec] <- NA
    }
  }
}

#set remaining NAs to zero
for(i in kgeur(colnames(eflalo))) eflalo[which(is.na(eflalo[,i]) == T),i]
<- 0

#- Remove non-unique trip numbers####
eflalo <- eflalo[!duplicated(paste(eflalo$LE_ID,eflalo$LE_CDAT,sep="-")),]
remrecsEflalo["duplicated",] <- c(nrow(eflalo),100+round((nrow(eflalo) -
an(remrecsEflalo["total",1]))/an(remrecsEflalo["total",1])*100,2))

#make a backup
eflaloBackup <- eflalo

save(remrecsEflalo,file="working/remrecsEflalo.RData")
save(eflalo, file ="working/cleanEflalo.RData")
eflaloBackup2 <- eflalo

print("EFLALO Cleaned!")


# Merge iVMS and Landings ---------------------------------------------------
tacsat <- formatTacsat(tacsat)
eflalo <- formatEflalo(eflalo)

tacsatp           <- mergeEflalo2Tacsat(eflalo,tacsat)

#add columns from landings eflalo
tacsatp$LE_GEAR   <- eflalo$LE_GEAR[match(tacsatp$FT_REF,eflalo$FT_REF)]
tacsatp$LE_MSZ    <- eflalo$LE_MSZ[ match(tacsatp$FT_REF,eflalo$FT_REF)]
tacsatp$VE_LEN    <- eflalo$VE_LEN[ match(tacsatp$FT_REF,eflalo$FT_REF)]
tacsatp$VE_KW     <- eflalo$VE_KW[  match(tacsatp$FT_REF,eflalo$FT_REF)]
tacsatp$LE_RECT   <- eflalo$LE_RECT[ match(tacsatp$FT_REF,eflalo$FT_REF)]
tacsatp$LE_MET    <- eflalo$LE_MET[  match(tacsatp$FT_REF,eflalo$FT_REF)]
tacsatp$LE_WIDTH  <- eflalo$LE_WIDTH[match(tacsatp$FT_REF,eflalo$FT_REF)]
tacsatp$VE_FLT    <- eflalo$VE_FLT[  match(tacsatp$FT_REF,eflalo$FT_REF)]
tacsatp$LE_CDAT   <- eflalo$LE_CDAT[ match(tacsatp$FT_REF,eflalo$FT_REF)]
tacsatp$VE_COU    <- eflalo$VE_COU[ match(tacsatp$FT_REF,eflalo$FT_REF)]

#save unmerged data
tacsatpNoMerge       <- subset(tacsatp,FT_REF == 0)
save(tacsatpNoMerge,file= "working/tacsatNotMerged.RData")

#save merged data
tacsatM           <- subset(tacsatp,FT_REF != 0)
save(tacsatM,    file="working/tacsatMerged.RData")

#count how many trips have been merged
NoTrips <- length(unique(tacsatM$FT_REF))
NoTrips ##check number

# Find out number of days linked, by vessel ####
#subset by vessel
vl1 <- subset(tacsatM, VE_REF == "1001")
```

```
vl2 <- subset(tacsatM, VE_REF == "1002")
vl3 <- subset(tacsatM, VE_REF == "1003")
vl4 <- subset(tacsatM, VE_REF == "1004")
vl5 <- subset(tacsatM, VE_REF == "1005")
vl6 <- subset(tacsatM, VE_REF == "1006")
vl7 <- subset(tacsatM, VE_REF == "1007")
vessels <- list(vl1,vl2,vl3,vl4,vl5,vl6,vl7)

#count number of unique trip refs
uniDay <- function (x, column){
  dayz <- length(unique(x[[column]]))
  return(dayz)
}

operational$landingsLinkedDays <-  lapply(vessels, uniDay, "FT_REF")

#calc percent column and add to summary table
operational$Percentage <-
paste(round(c(as.numeric(operational$landingsLinkedDays) /
as.numeric(operational$VesselActive)) * 100), "%", sep = "")
operational <- operational[,c("Start", "Finish", "DaysiVMSOperating",
"VesselActive", "landingsLinkedDays", "Percentage")]
totalPercent <- c(sum(an(operational$landingsLinkedDays)) /
sum(an(operational$VesselActive))) * 100

head(operational)

#view linking summary data ####
print("iVMS and landings data is cleaned and linked!!!!")
remrecsTacsat
remrecsEflalo
operational
print(paste("DaysiVMSworking = ", sum(operational$DaysiVMSOperating), sep =
" "))
print(paste("Possible fishing days =", sum(operational$VesselActive), sep
=" "))
print(paste("Days with corresponding landings data = ",
sum(an(operational$landingsLinkedDays))), sep = "")
print(paste("Percentage of trips with landings data = ",
round(totalPercent), "%"), sep = "")



# IDENTIFY FISHING ACTIVITY -----------------------------------------------

#remove points with NA in important fields
idx                <- which(is.na(tacsatM$VE_REF) == T    |
is.na(tacsatM$SI_LONG) == T | is.na(tacsatM$SI_LATI) == T |
                           is.na(tacsatM$SI_DATIM) == T |
is.na(tacsatM$SI_SP) == T)
if(length(idx)>0) tacsatM         <- tacsatM[-idx,]



tacsatM <- formatTacsat(tacsatM)

# Auto activity Analysis ####
# x11()
# storeScheme   <- activityTacsatAnalyse(tacsatM, units= "year", analyse.by
= "LE_GEAR", identify = "means")
#
```

```
# storeScheme
# #- Define mean values of the peaks and the number of peaks
# storeScheme$means[which(storeScheme$analyse.by == "Pot")]       <- c("-13
-1 0 1 13 ")
# storeScheme$means[which(storeScheme$analyse.by == "Line")]      <- c("-16
-7 0 7 16")
# storeScheme$means[which(storeScheme$analyse.by == "Gill Net")]  <- c("-7
-4 0 4 7")
# storeScheme$means[which(storeScheme$analyse.by == "B.S.Net")]   <- c("-12
-7 0 7 12")
# tacsatM <- formatTacsat(tacsatM)
# acTa                      <- activityTacsat(tacsatM,
units="year",analyse.by="LE_GEAR",
#
storeScheme=storeScheme,plot=FALSE,level="all")
#
# tacsatM$SI_STATE        <- acTa
# tacsatM$ID              <- 1:nrow(tacsatM)

##INSUFFICIENT DATA FOR GOOD AUTO APPROACH


#- Investigate speed pattern through visual inspection of histograms ####
#create speed array for gears
speedarr          <-
as.data.frame(cbind(LE_GEAR=sort(unique(tacsatM$LE_GEAR)),min=NA,max=NA),st
ringsAsFactors=F)
speedarr$min      <- rep(0,nrow(speedarr)) # adjust below
speedarr$max      <- rep(6,nrow(speedarr))

#look for peaks ####
X11()
par(mfrow=c(2,2))
hist(subset(tacsatM, LE_GEAR ==
"Pot")$SI_SP,breaks=30,xlim=c(0,15),xlab="Knots",main="Pots")
hist(subset(tacsatM, LE_GEAR ==
"B.S.Net")$SI_SP,breaks=30,xlim=c(0,15),xlab="Knots",main="Seine")
hist(subset(tacsatM, LE_GEAR ==
"Line")$SI_SP,breaks=30,xlim=c(0,15),xlab="Knots",main="Line")
hist(subset(tacsatM, LE_GEAR == "Gill
Net")$SI_SP,breaks=30,xlim=c(0,15),xlab="Knots",main="Gillnet")

#fill speed array
speedarr[speedarr$LE_GEAR == "Pot",]$max <- 3
speedarr[speedarr$LE_GEAR == "B.S.Net",]$max <- 3
speedarr[speedarr$LE_GEAR == "Line",]$max <- 7
speedarr[speedarr$LE_GEAR == "Gill Net",]$max <- 3

#create .png of histograms ####
png(filename= "outputs/SpeedHistogram.png")
ggplot(data=tacsatM, aes(SI_SP)) +
  geom_histogram(breaks=seq(0, 20, by =0.4),
                 col=1)+
  facet_wrap(~LE_GEAR,ncol=2,scales="free_y")+
  labs(x = "Speed (knots)", y = "Frequency") +
  theme(axis.text.y    = element_text(colour="black"),
        axis.text.x    = element_text(colour="black"),
        axis.title.y   = element_text(size=14),
        axis.title.x   = element_text(size=14),
        panel.background = element_blank(),
        panel.grid.major = element_blank(),
```

```
        panel.grid.minor = element_blank(),
        axis.line = element_line(colour = "black"),
        panel.border = element_rect(colour = "black", fill=NA))
dev.off()

#fill SI_STATE column depending on speeds ####
tacsatM$SI_STATE     <- NA
meteiers              <- unique(tacsatM$LE_GEAR)

for (mm in metiers) {
  tacsatM$SI_STATE[tacsatM$LE_GEAR==mm & tacsatM$SI_SP >=
speedarr[speedarr$LE_GEAR==mm,"min"] & tacsatM$SI_SP <=
speedarr[speedarr$LE_GEAR==mm,"max"]]   <- "f";
}
tacsatM$SI_STATE[is.na(tacsatM$SI_STATE)] <- "s"

#Looking at the data here I think it should also be possible to sperate the
trolling from the ground line fishing.
#speeds of 5-7knots are probably trolling?  Add column to Investiate later-
might be able to seperate
tacsatM$Troll <- "NA"
tacsatM[tacsatM$LE_GEAR == "Line" & tacsatM$SI_SP >=5 & tacsatM$SI_SP <=
7,]$Troll <-   "1"


save(tacsatM, file = "working/tacsatActivity.RData")

tacsatBackup3 <- tacsatM

print("Defining Fishing activity completed!")


# SPLIT LANDINGS AMONGST iVMS PINGS ---------------------------------------

#list columns that are KG or Value
idxkgeur            <- kgeur(colnames(eflalo))
#Create total columns
eflalo$LE_KG_TOT   <- eflalo$LE_KG_ALL
eflalo$LE_EURO_TOT<- eflalo$LE_EURO_ALL
#remove the rest of the KG or value columns
eflalo             <- eflalo[,-idxkgeur]
#Landings data that wasn't merged.
eflaloNM            <- subset(eflalo,!FT_REF %in% unique(tacsatM$FT_REF))
#landings data that was merged
eflaloM             <- subset(eflalo,FT_REF %in% unique(tacsatM$FT_REF))

#set SI_State to binary
tacsatM$SI_STATE[which(tacsatM$SI_STATE != "f")] <- 0
tacsatM$SI_STATE[which(tacsatM$SI_STATE == "f")] <- 1

#Refromat Data and back up.
tacsatM <- formatTacsat(tacsatM)
eflalo <- formatEflalo(eflaloM)
tacsatBackup4 <- tacsatM

#select only fishing pings
tacsatM <- subset(tacsatM, SI_STATE == 1)

#Split landings amongst fishing pings.
```

```
tacsatEflalo  <-
splitAmongPings(tacsat=tacsatM,eflalo=eflalo,variable="all",level="trip",co
nserve=FALSE)

#add columns ####
tacsatEflalo$Csquare  <-
CSquare(tacsatEflalo$SI_LONG,tacsatEflalo$SI_LATI,degrees=0.01)
tacsatEflalo$Year     <- year(tacsatEflalo$SI_DATIM)
tacsatEflalo$Month    <- month(tacsatEflalo$SI_DATIM)

#reorder columns
tacsatEflalo1 <- tacsatEflalo[,c("VE_NAME", "VE_REF","SI_DATE", "SI_TIME",
"SI_LONG", "SI_LATI",  "SI_SP",  "SI_HE", "source", "poll_id",
"received_date", "SI_DATIM", "INTV",  "FT_REF", "LE_GEAR", "LE_CDAT",
"VE_COU", "Troll", "SI_STATE", "LE_KG_TOT", "LE_EURO_TOT" )]

save(tacsatEflalo,file= "working/TacsatEflalo.RData")

print("Dispatching landings completed")


# Create Objects to Map ----------------------------------------------------
#Individual Vessels
ves1te <- subset(tacsatEflalo1, VE_REF == "1001")
ves2te <- subset(tacsatEflalo1, VE_REF == "1002")
ves3te <- subset(tacsatEflalo1, VE_REF == "1003")
ves4te <- subset(tacsatEflalo1, VE_REF == "1004")
ves5te <- subset(tacsatEflalo1, VE_REF == "1005")
ves6te <- subset(tacsatEflalo1, VE_REF == "1006")
ves7te <- subset(tacsatEflalo1, VE_REF == "1007")

vesselsTE <- list(ves1te, ves2te, ves3te, ves4te, ves5te, ves6te, ves7te)

names(vesselsTE) <- c("vs1", "vs2", "vs3", "vs4", "vs5", "vs6", "vs7")

#Gears
FT_Lines <- subset(tacsatEflalo1, LE_GEAR == "Line")
FT_Pot <- subset(tacsatEflalo1, LE_GEAR == "Pot")
FT_Seines <- subset(tacsatEflalo1, LE_GEAR == "B.S.Net")
FT_Gillnet <- subset(tacsatEflalo1, LE_GEAR == "Gill Net")

gearz <- list(FT_Lines, FT_Pot, FT_Seines, FT_Gillnet)


# Make rasters ------------------------------------------------------------

#various parameters for use in making raster and png plot
xmax <- -62
xmin <- -62.42
ymax <- 16.92
ymin <- 16.62
xlim <- c(xmin, xmax)
ylim <- c(ymin, ymax)



#function to make rasters by vessel ####
make_Vess_Raster <- function(ivms){
  vNam = unique(ivms$VE_REF)
  filNam = paste("outputs/", vNam, sep = "")
  lats = data.frame(ivms[,6]) # latitude column
```

```
  longs = data.frame (ivms [,5]) # longditude column
  coords = data.frame (cbind(longs,lats))
  emptyRaster = raster()
  extent(emptyRaster)= as.numeric(paste(c(xlim,ylim), sep=","))
  res(emptyRaster) = 0.01
  #projection(emptyRaster) = CRS("+proj=longlat +datum=WGS84")

  try({
    agg.raster = rasterize(coords, emptyRaster, ivms$LE_EURO_TOT, fun =
"sum")#, background = NA, mask = FALSE, update = FALSE, datatype=
"INT2U")#background = NA or 0
#mask to eez/ukcs
  agg.raster = mask(agg.raster, mniEEZ)
  writeRaster(agg.raster, filename= paste("outputs/", vNam, ".tif", sep =
""), format="GTiff", overwrite=TRUE)
  })
}

lapply(vesselsTE, make_Vess_Raster)

#Function to make LPUE by vessel
LPUE <- function(ivms){
    vNam <- unique(ivms$VE_REF)
    filNam <- paste("outputs/", vNam, sep = "")
    lats <- data.frame(ivms[,6]) # latitude column
    longs <- data.frame (ivms [,5]) # longditude column
    coords <- data.frame (cbind(longs,lats))
    emptyRaster <- raster()
    extent(emptyRaster)<- as.numeric(paste(c(xlim,ylim), sep=","))
    res(emptyRaster) <- 0.01
    #projection(emptyRaster) <- CRS("+proj=longlat +datum=WGS84")

    try({
      agg.raster <- rasterize(coords, emptyRaster, ivms$LE_EURO_TOT, fun =
"sum", background = NA, mask = FALSE, update = FALSE, datatype=
"INT2U")#background = NA or 0
    totEffort = length(unique(ivms$FT_REF))
    agg.raster2 <- calc(agg.raster, fun = function(x){x / totEffort})
    #mask to eez/ukcs
    agg.raster2 <- mask(agg.raster2, mniEEZ)
    writeRaster(agg.raster2, filename= paste("outputs/", vNam, "_LPUE.tif",
sep = ""), format="GTiff", overwrite=TRUE)
    })
}

lapply(vesselsTE, LPUE)

# Function to make rasters by gear ####
make_Gear_Raster <- function(ivms){
  vNam <- unique(ivms$LE_GEAR)
  lats <- data.frame(ivms[,6])
  longs <- data.frame (ivms [,5])
  coords <- data.frame (cbind(longs,lats))
  emptyRaster <- raster()
  extent(emptyRaster)<- as.numeric(paste(c(xlim,ylim), sep=","))
  res(emptyRaster) <- 0.01
  #projection(emptyRaster) <- CRS("+proj=longlat +datum=WGS84")
  filNam <- paste("outputs/", vNam, sep = "")
  try({agg.raster <- rasterize(coords, emptyRaster, ivms$LE_EURO_TOT, fun =
"sum", background = NA, mask = FALSE, update = FALSE, datatype=
"INT2U")#background = NA or 0
```

```
  #mask to eez/ukcs
  agg.raster <- mask(agg.raster, mniEEZ)
  writeRaster(agg.raster, filename= paste(filNam,".tif", sep = ""),
format="GTiff", overwrite=TRUE)
  })
  }

lapply(gearz, make_Gear_Raster)



#Function to make LPUE by vessel
LPUE_gear <- function(ivms){
  vNam <- unique(ivms$LE_GEAR)
  filNam <- paste("outputs/", vNam, sep = "")
  lats <- data.frame(ivms[,6]) # latitude column
  longs <- data.frame (ivms [,5]) # longditude column
  coords <- data.frame (cbind(longs,lats))
  emptyRaster <- raster()
  extent(emptyRaster)<- as.numeric(paste(c(xlim,ylim), sep=","))
  res(emptyRaster) <- 0.01
  #projection(emptyRaster) <- CRS("+proj=longlat +datum=WGS84")

  try({
    agg.raster <- rasterize(coords, emptyRaster, ivms$LE_EURO_TOT, fun =
sum, background = NA, mask = FALSE, update = FALSE, datatype=
"INT2U")#background = NA or 0
  totEffort = length(unique(ivms$FT_REF))
  agg.raster2 <-  calc(agg.raster, fun = function(x){x / totEffort})
  #mask to eez/ukcs
  agg.raster2 <- mask(agg.raster2, mniEEZ)
  writeRaster(agg.raster2, filename= paste("outputs/", vNam, "_LPUE.tif",
sep = ""), format="GTiff", overwrite=TRUE)
  })
}

lapply(gearz, LPUE_gear)


# Total value Raster ####
lats <- data.frame(tacsatEflalo1[,6])
longs <- data.frame (tacsatEflalo1 [,5])
coords <- data.frame (cbind(longs,lats))
emptyRaster <- raster()
extent(emptyRaster)<- as.numeric(paste(c(xlim,ylim), sep=","))
res(emptyRaster) <- 0.01
#projection(emptyRaster) <- CRS("+proj=longlat +datum=WGS84")
TotVraster <- rasterize(coords, emptyRaster, tacsatEflalo1$LE_EURO_TOT, fun
= sum, background = NA, mask = FALSE, update = FALSE, datatype=
"INT2U")#background = NA or 0

#mask to eez/ukcs
TotVraster <- mask(TotVraster, mniEEZ)
writeRaster(TotVraster, filename= "outputs/All.tif", format="GTiff",
overwrite=TRUE)

#Total Value Per Unit Effort ####
totEffort = length(unique(tacsatEflalo1$FT_REF))
TotLPUE <- calc(TotVraster, fun = function(x){x / totEffort})
writeRaster(TotLPUE, filename = "outputs/iVMS_Total_LPUE.tif", format =
"GTiff", overwrite = T)
```

```
print("Aggregated rasters created in output folder!")


# Plots -------------------------------------------------------------------

#clip that bathy shapefile because its taking sooo damn long to plot


b <- bbox(mni)
b[1,1] <- b[1,1]-0.5
b[2,1] <- b[2,1]-0.5
bb <- bbox(t(b))

gClip <- function(shp, bb){
  if(class(bb) == "matrix") b_poly <- as(extent(as.vector(t(bb))),
"SpatialPolygons")
  else b_poly <- as(extent(bb), "SpatialPolygons")
  gIntersection(shp, b_poly, byid = T)
}

bathyClipped <- gClip(bathy, bb)

#object lists for sp.layout
landMap <- list("sp.polygons", mni, fill = "lightgreen", lwd = 0.5, border
= "light grey")
ez <- list("sp.polygons", mniEEZ, fill = "transparent", border = "red", lwd
= 0.5)
bathym <- list("sp.lines", bathyClipped, col = "lightblue", lwd = 0.5,
first = TRUE)   #, first = TRUE
maplist <- list(bathym, ez, landMap)


#Read rasters layers into Raster Stack
layerz <- list.files("outputs", pattern="tif$", full.names=TRUE)
layerss <- stack(layerz)
layerss
#Select only total layers (leave out LPUE and empties(gillnet and vesse))
Totlyrz<- layerss[[c(1,3,7,9, 11)]]

#MMake lpue multi plots for report
layerzLPUE <- list.files("outputs", pattern="LPUE.tif$", full.names=TRUE)
LPUElyrz <- stack(layerzLPUE)
LPUElyrz
#Get rid of empty layers (gillnets and vX1004)
LPUElyrz <- LPUElyrz[[c(-3, -7)]]

names(Totlyrz)
names(LPUElyrz)
names(LPUElyrz) <- c("Vessel_X1002", "Vessel_X1003", "Vessel_X1005",
"Vessel_X1007", "SeineNet", "All", "Lines", "Pots")
# Individual Plots --------------------------------------------------------
x11()
hist(Totlyrz)
#no common scale suitable

#Color Ramp
col <- colorRampPalette(c( "yellow", "orange", "red", "dark red"))
```

```
#remved from first try line below
#Loop through layers and create png maps
for(i in 1:nlayers(Totlyrz)){
  FilNam <- names(Totlyrz[[i]])
  max <- round(max(maxValue((Totlyrz[[i]]))))
  try({png(filename = paste("outputs/", FilNam, ".png", sep =""))
    main = paste( FilNam, " total landed value linked to iVMS data", sep =
"")
    sub = "EC$"
    p <- spplot(Totlyrz[[i]], alpha.regions = 0.5,
                sp.layout= maplist,
                xlim = xlim,
                ylim= ylim,
                margin = F, main = main, sub = sub,
                col.regions = col,
                at = pretty(0:max, 20),
                colorkey = list(space="bottom", width = 0.9, height = 0.9)
    )
    plot(p)

  })
  dev.off()

}

# Multipanel plot All vessel landings -------------------------------------
-------

# Multipanel plot of All vessels total landings on same scale
main = "All iVMS linked landings"
max <- round(max(maxValue(Totlyrz)))
png(filename = "AllLandingsMultiPanel.png")
p <- spplot(Totlyrz, c("X1002", "X1003", "X1005", "X1007"),
            names.attr = c("Vessel X1002", "Vessel X1003", "Vessel X1005",
"Vessel X1007"),
            pretty = TRUE, alpha.regions = 0.6,
            as.table = TRUE,
            sp.layout = maplist,
            xlim = xlim,
            ylim = ylim,
            margin = F,
            main = main,
            sub = sub,
            col.regions = col,
            at = pretty(0:max, 80),
            colorkey = list(space="bottom", width = 0.9, height = 0.9))
plot(p)
dev.off()



# LPUE Plots --------------------------------------------------------------
-

subtitle <- "LPUE EC$"

#function to make single LPUE Maps
make_maps <- function(x){
  FilNam = names(x)
  png(filename = paste("outputs/", FilNam, ".png", sep =""))
```

```
  main = paste( FilNam, " iVMS linked landings value per unit effort", sep
= "")
  p <- spplot(x, pretty = TRUE, alpha.regions = 0.6,
              as.table = TRUE,
              sp.layout = maplist,
              xlim = xlim,
              ylim = ylim,
              margin = F,
              main = main,
              sub = subtitle,
              col.regions = col,
              colorkey = list(space="bottom", width = 1.5, height = 1,
axis.line = list(alpha = 0),labels = list(at = seq(0.5, length(colbreaks) -
0.5), labels = colbreaks))
  )
  plot(p)
  dev.off()

}
#lapply(cutLPUE, make_maps)

#Couldn't get function to work?  Poor practise, but here's a loop instead.
#Color Ramp

for(i in 1 : nlayers(LPUElyrz)){
  FilNam <- names(LPUElyrz[[i]])
  max <- round(maxValue((LPUElyrz[[i]])))
  png(filename = paste("outputs/", FilNam, "_LPUE.png", sep =""))
  main = paste( FilNam, " iVMS linked Landings per Unit Effort", sep = "")
  subtitle = "EC$"
  p <- spplot(LPUElyrz[[i]], alpha.regions = 0.6,
              as.table = TRUE,
              sp.layout= maplist,
              xlim = xlim,
              ylim= ylim,
              margin = F, main = main, sub = subtitle,
              col.regions = col,
              at = pretty(0:max, 20),
              colorkey = list(space="bottom", width = 0.9, height = 0.9))

  plot(p)
  dev.off()
}


# Multi PLots for Report ---------------------------------------------------

#check ranges for scaling
X11()
hist(LPUElyrz)
summary(LPUElyrz)

# Breaks
colbreaks <- c(0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200)

#covert to factor for better scaling of colorRamps
make_Cuts <- function(LPUEx){
  LPUEx <- cut(LPUEx, breaks = c(colbreaks, Inf))
}

cutLPUE <- make_Cuts(LPUElyrz)
```

```
names(cutLPUE)
summary(cutLPUE)
x11()
hist(cutLPUE)

#titles
main <-  "Landings Per Unit Effort"
sub <- "LPUE EC$"

names(cutLPUE)


#write png and plot
#plots of lpue by vessel
png(filename = "outputs/LPUEvessel.png")
p <- spplot(cutLPUE, c("Vessel_X1002", "Vessel_X1003", "Vessel_X1005",
"Vessel_X1007"),
            names.attr = c("Vessel X1002", "Vessel X1003", "Vessel X1005",
"Vessel X1007"),
            alpha.regions = 0.6,
            as.table = TRUE,
            sp.layout = maplist,
            xlim = xlim,
            ylim = ylim,
            margin = F,
            main = main,
            sub = sub,
            col.regions = col,
            at = c(1:14),
            colorkey = list(space="bottom", height = 0.9, width = 0.9,
labels = list(labels = colbreaks, at = seq(0, length(colbreaks) +0.5) ))
)
plot(p)
dev.off()

# plots of lpue by gear type
png(filename = "outputs/LPUEgear.png")
p <- spplot(cutLPUE, c("SeineNet", "Pots", "Lines", "All"),
            names.attr = c("Beach Seine Net", "Pot", "Lines", "All"),
            pretty = TRUE, alpha.regions = 0.6,
            as.table = TRUE,
            sp.layout = maplist,
            xlim = xlim,
            ylim = ylim,
            margin = F,
            main = main,
            sub = sub,
            col.regions = col,
            at = c(1:14),
            colorkey = list(space="bottom", height = 0.9, width = 0.9,
labels = list(labels = colbreaks, at = seq(0, length(colbreaks) +0.5) ))
)
plot(p)
dev.off()



# Write out points shapefile ----------------------------------------------

#write shapefile of points
```

```
lats <- data.frame(tacsatEflalo1[,6])
longs <- data.frame(tacsatEflalo1[,5])
coords <- data.frame(cbind(longs,lats))
pts <- SpatialPoints(coords, proj4string = CRS("+proj=longlat
+ellps=WGS84"))
atts <- SpatialPointsDataFrame(pts, tacsatEflalo1, coords.nrs = numeric(0),
                               proj4string = CRS("+proj=longlat
+ellps=WGS84" ), match.ID = TRUE)
tacsatEflalo1$SI_DATIM <- as.character(tacsatEflalo1$SI_DATIM)
writePointsShape(atts, "outputs/TacsatEfaflo")



# Write tables for report ------------------------------------------------

#Get trip count
trips_count <- function(x) {
  y <- length(unique(x$FT_REF))
  return(y)
}

tripCountgear <- lapply(gearz, trips_count)

#Sum Hrs Fishing
sum_hrs <- function(x){
  round(sum(x$INTV) / 60)
}

sumHrsGear <- lapply(gearz, sum_hrs)

#Percentage of total recorded landings
perc_tot <- function(x){
 a <-  round(sum(x$LE_EURO_TOT) /sum(subset(MNIlandings, LE_GEAR ==
unique(x$LE_GEAR))$LE_EURO_ALL)*100)
 return(a)
}
pTotLan <- lapply(gearz, perc_tot)
pTotLan

CovByGear <- as.data.frame(cbind(tripCountgear, sumHrsGear, pTotLan))

names(CovByGear) <- c("Number of fishing days", "Fishing hrs", "% of total
landings" )

row.names(CovByGear) <- c("Lines", "Pot", "Seines", "Gillnet")

CovMat <- as.matrix(CovByGear)


#write html tables

#data coverage by gear
DataCovGear <- tableHTML(CovMat, width = c(80,80,80,80), theme =
"scientific")
print(DataCovGear, type = "html")

#operation times, trips, landings linked
names(operational) <- c("Start", "Finish", "Days iVMS operational",  "Days
vessel active", "Active days linked to landings", "Vessel activity
covered")
```

```
row.names(operational) <- c("Vessel X1001", "Vessel X1002", "Vessel X1003",
"Vessel X1004", "Vessel X1005", "Vessel X1006", "Vessel X1007")
DataColOp <- tableHTML(operational, width = c(120, 100, 100,60,60,90,60),
theme = "scientific")
print(DataColOp, type = "html")

#tacsat cleaning
remrecsTacsat <- data.frame(remrecsTacsat)
remrecsTacsat$percentage <-
round(as.numeric(as.character(remrecsTacsat$percentage)), digits = 2)

TecRemrecBU <- remrecsTacsat

names(remrecsTacsat) <- c("Rows remaining", "% of total")
remrecsTacsat[1,2] <- 100
remrecsTacsat
TacClean <- tableHTML(remrecsTacsat, theme = "scientific")
print(TacClean, type = "html")

# Speed Array

speedarr[1,1] <- "Beach Seine Net"
rownames(speedarr) <- speedarr$LE_GEAR
speedarr$LE_GEAR <- NULL
speedarr1 <- tableHTML(speedarr, theme = "scientific", headers = c("min
speed", "max speed"), caption = "Fishing speed thresholds used for each
gear analysed" )
print(speedarr1, type = "html")

#Print summaries to console
print("iVMS and landings data is cleaned, linked and aggregated")
print(paste("DaysiVMSworking = ", sum(operational$DaysiVMSOperating), sep =
" "))
print(paste("Possible fishing days =", sum(operational$VesselActive), sep
=" "))
print(paste("Days with corresponding landings data = ",
sum(an(operational$landingsLinkedDays))), sep = "")
```

Appendix 2

Fishing speed threshold identification

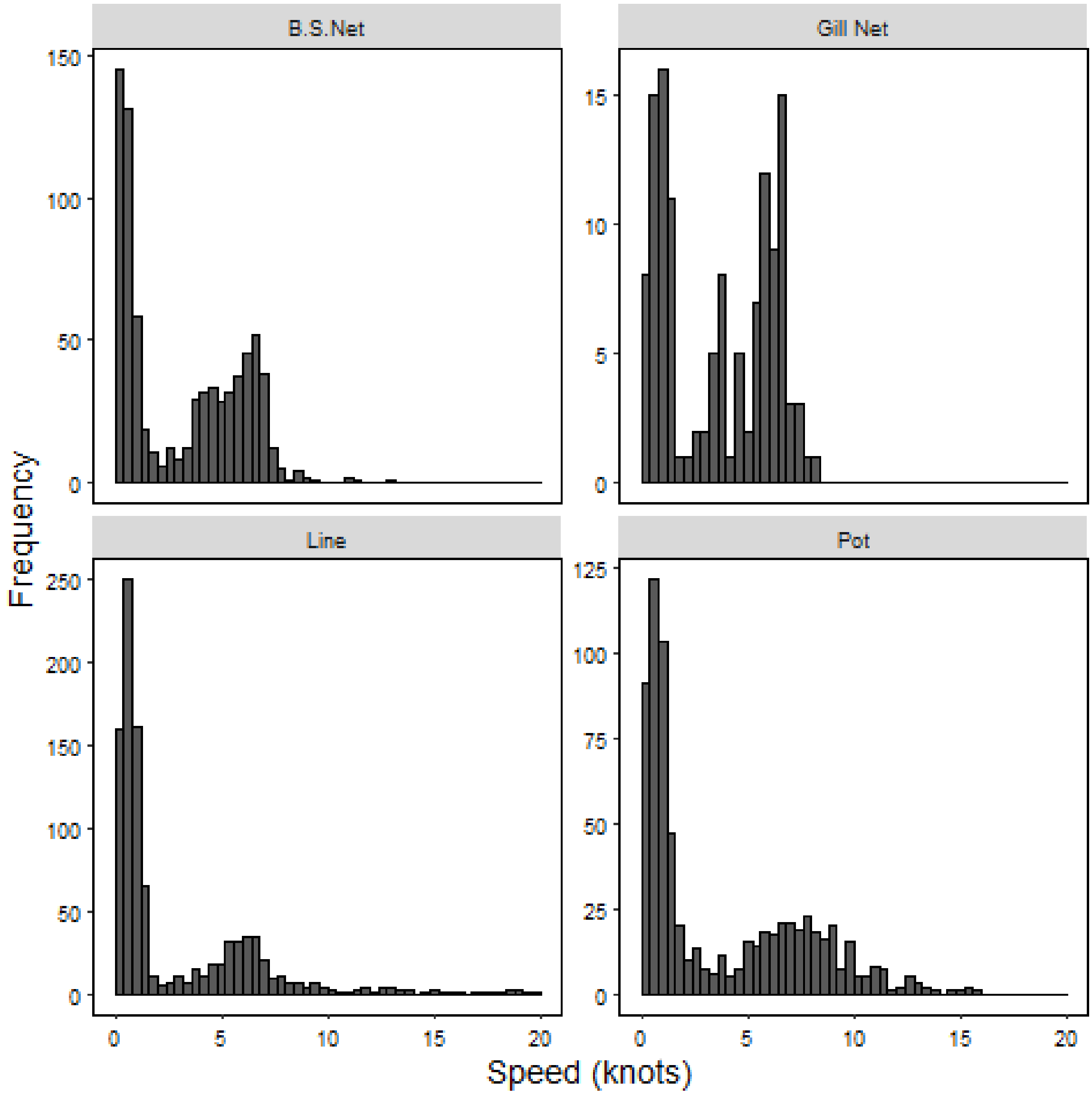

Figure 3 Histogram of speed frequencies for different gear types

Fishing speed thresholds used for each gear analysed

	min speed	max speed
Beach Seine Net	0	3
Gill Net	0	3
Line	0	7
Pot	0	3

Appendix 3

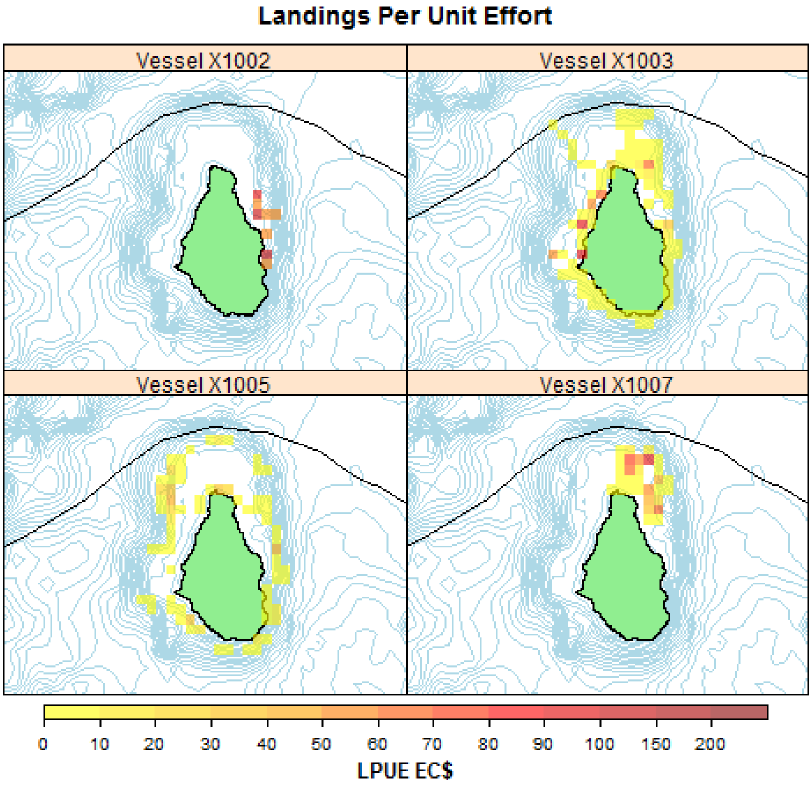

Figure 4 Landings per unit effort (average landed value per day fishing) by vessel

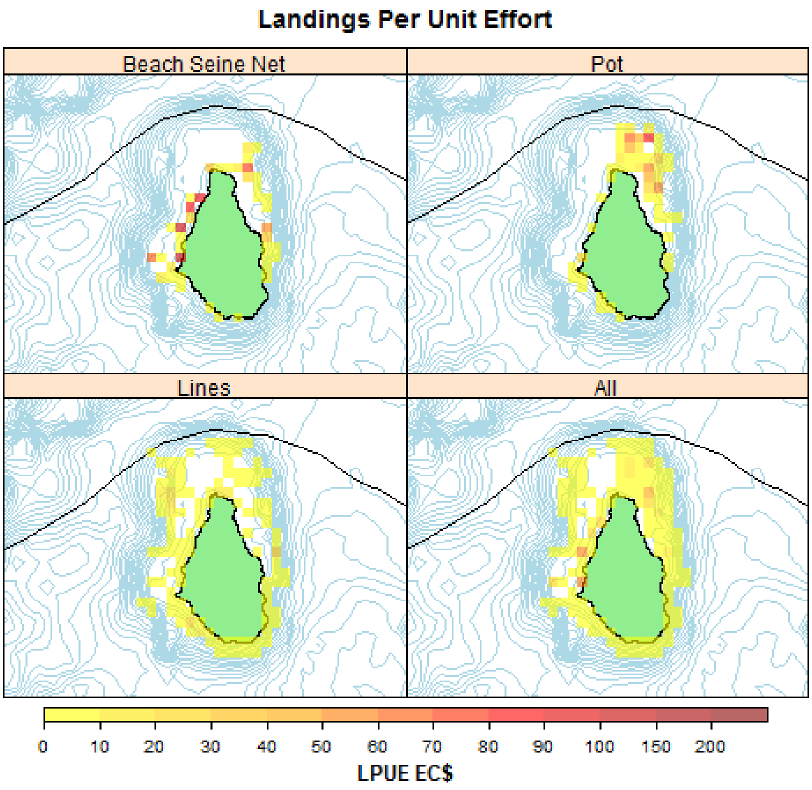

Figure 5 Landings per unit effort (average landed value per day) by gear type.

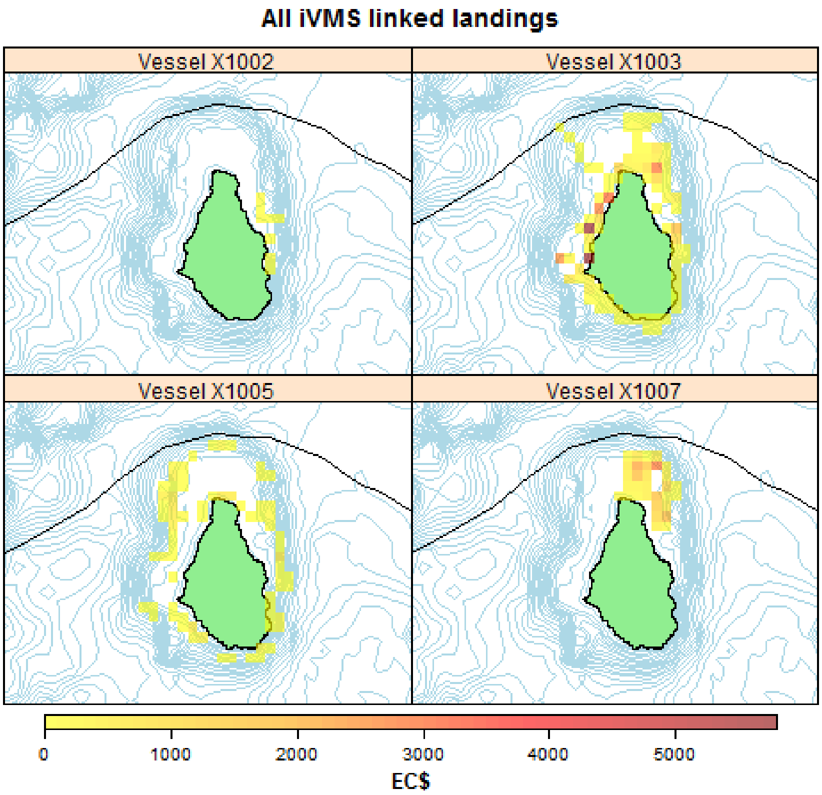

Figure 6 Total landed value of data, by vessel