R tutorial

iapesca, a R-package for manipulating and interpretating high resolution geospatial data from fishing vessels

RBE - Hisseo

Ifremer • Fisheries Information System • Julien Rodriguez

30 janvier 2023

- R source for this tutorial available on https://gitlab.ifremer.fr/iapesca/r-tutorial_iapesca

- This package provides methods for cleaning, detecting fishing trips, resampling positions at a constant time interval, calculating covariates describing boat trajectories. It also offers usefuls functions for optimizing hyperparameters for machine-learning models, creating behavioral diagnosis or analyzing the data as far as geocomputation methods used to analyze passive gears with appropriated fishing effort metrics.

- These developments have been carried out through the IAPESCA and DELMOGES programs and inspired and guided by the methods and experiences shared during the ICES WKSSFEO in Lisbon in December 2021.

1 - Description and settings

1.1 Purpose

- The purpose of this package is to provide generic methods for qualifying geolocation data regardless of the source Using high-resolution geolocation data for fishing vessels, the methods provided aim at improving the calculation of vessel fishing effort and proposing metrics more suited for passive gears. The applications have been developed for fishing vessels operating nets with following metrics : fishing time, nets length and soaking time. They could easily be used or extended for active gears or other passive gears.
- Different methods and tools are proposed, in particular for :
  - Data preparation : cleaning, fishing trips detection, resampling, linking to vmstools
  - The calculation of covariates used to implement machine-learning models (iapesca project)
  - Machine-learning methods : behavioral classifications, novelty detection, optimization of random forest models, etc. . ..
  - Geo-computation methods allowing the detection and consolidation of passive gears and the detection of setting events by using buffers and scoring.
  - The calculation of gear metrics for passive gears.
- This work has greatly benefited from the contributions of Mathieu Woillez and François Danhiez (Cap Gemini) within the framework of the Iapesca project, of Martial Laurans for his expertise and his help in preparing the dataset provided in the package and, of course, of the contributions of the ICES WKSSFGEO, relying heavily on guidance and codes made available by participants.

1.2 Configuration

- This package and R-codes were developped using R version 4.1.3 (2022-03-10) – “One Push-Up” and tested with R version 4.2.2 running under Windows 10 x64
- Run on DELL Precision 7550 with Intel(R) Core(TM) i7-10875H CPU processor

1.3 Installing iapesca package

- Package source available on https://gitlab.ifremer.fr/iapesca/r-packages_iapesca
- Define a R project or set the directory to link to the folder downloaded from Gitlab
- Run following devtools commands to prepare the package from source :
  devtools : :build()
  devtools : :document()
- Install using either the command *devtools : :install()* or the compressed file created “iapesca_x.x.tar.gz”

- Load the package using the command :

```
library(iapesca)
```

- View the functions in the package (only the first objects shown)

```
ls(getNamespace("iapesca"), all.names=TRUE) %>% head
```

```
[1] ".__NAMESPACE__."            ".__S3MethodsTable__."       ".packageName"               "Backgroun
```

- Test some functions by running the examples provided in documentation

```
?novelty_svm
```

- If an error message appears, restart RStudio or your R session and reload the package running *library(iapesca)*

1.4 Machine-learning models used in this tutorial

- Download the "ModelsNets_Recopesca0900.rds" object using the link provided
- Copy and paste this object in the folder /data defined by the path *paste0(root, "/data")*

1.5 Some useful functions used in this tutorial not provided in the package

```
# plot_tree: plots a decision tree from a CART model calibrated using rpart package

plot_tree <- function(mod.CART){

    if(inherits(mod.CART, "rpart")){

      oldw <- getOption("warn")
      options(warn = -1)

      rpart.plot::prp(
        mod.CART,
        type = 2,
        extra = 1,
        split.cex = 1.5,
        cex = 0.9,
        box.palette = "YlGnBl")

      options(warn=oldw)

    }else{

      warning("Object must be a rpart::rpart output")

      }

}
```

```r
# multi_ggplots: plots many ggplots using the same window

multi_ggplots <- function(...,
                          plotlist=NULL,
                          cols) {

    plots <- c(list(...), plotlist)

    numPlots <- length(plots)

    plotCols <- cols
    plotRows <- ceiling(numPlots/plotCols)

    grid::grid.newpage()
    grid::pushViewport(grid::viewport(layout = grid::grid.layout(plotRows, plotCols)))
    vplayout <- function(x, y){
      grid::viewport(layout.pos.row = x, layout.pos.col = y)
    }

    for (i in 1:numPlots) {
      curRow = ceiling(i/plotCols)
      curCol = (i-1) %% plotCols + 1
      print(plots[[i]], vp = vplayout(curRow, curCol))
    }

}

# Calc_Acc: calculates model accuracy from an equilibrated contingency table

Calc_Acc <- function(cont.tab){

  acc <- round(100 * sum(diag(cont.tab))/sum(cont.tab), digits = 2)
  return(acc)

}

# plot_FE: creates y~x plots to compare fishing effort metrics

plot_FE <- function(FE.true,
                    FE.pred,
                    FE.met){

   rg <- range(FE.true[, FE.met], FE.pred[, FE.met], na.rm = TRUE)

   plot(FE.pred[, FE.met] ~ FE.true[, FE.met],
        pch = "+",
        ylim = rg,
        xlim = rg,
        xlab = "True operations",
        ylab = "Predicted operations",
        main = FE.met )
```

```
   abline(a = 0, b = 1, lty =2)

   text( x = FE.true[, FE.met],
         y = FE.pred[, FE.met] + as.numeric(diff(rg))/10,
         labels = FE.true$FISHING_TRIP_FK, cex  = 0.7)

}

# print_table: print tables in pdf document

print_table <- function(tab,
                        latex.options = c("HOLD_position", "scale_down"),
                        pos = "center"){

    row.names(tab) <- NULL
    tab %>%
    knitr::kable(format = "latex",
    format.args = list(big.mark = " ")) %>%
    kableExtra::kable_styling(latex_options = latex.options,
                              position = pos)

}
```

2 - iapesca features

- iapesca package provides 50 functions, many of them being helpers nested in umbrella functions

2.1 data

- Data from the French Fisheries Information System (FIS, https://sih.ifremer.fr/), the Ifremer permanent, operational and multidisciplinary national monitoring network for the observation of marine resources and their uses : positions and harbours.
- *positions* Geolocation information from a fishing vessel operating nets. This dataset is a subset of Ifremer, RECOPESCA database for which fishing operations have been identified and validated with the fisherman. This database has been anonymized : identity have been changed and spatial geolocation randomly translated. A data frame with 1,937 rows and 11 columns.
  - VESSEL_FK : Fishing vessel identifier, character
  - FISHING_TRIP_FK : Fishing trip identifier, character
  - DATE_TIME : timestamp, character coercible to POSIXct
  - turn : turning angle in radians, numeric calculated using EMbC package in Clustering_BivariateBinary function
  - speed : speed in knots, numeric calculated using EMbC package in Clustering_BivariateBinary function
  - hdg : heading in radians, numeric calculated using EMbC package in Clustering_BivariateBinary function
  - setting : fishing gear identifier for setting operations, a character string with NAs when the vessel isn't fishing
  - hauling : fishing gear identifier for hauling operations, a character string with NAs when the vessel isn't fishing
  - FishingOperation : a character string describing the fishing activity for each position
  - LONGITUDE : longitude of the the translated position in WGS84 CRS
  - LATITUDE : latitude of the the translated position in WGS84 CRS
- *harbours* Anonymized harbour table (French fleet file) from FIS database stored in Harmonie. A sf object with 1 row and 7 columns. It has been translated using the same constant as position so that the harbour fits the positions.

2.2 R-helpers

- *Char2Time* Converts a character string to valid date-time format in the desired time zone
- *CleanMemory* Cleans memory and temporary folder
- *CreateFormula* Creates a formula from character vectors
- *Harmonize_Ids* Returns unique identifiers having constant length
- *Pivot_Table* Creates different columns from a categorical variable
- *set_0nbr* Returns a character string with zeros depending on object size and ID.length

2.3 Geodata toolbox

- *BackgroundMapWithCountries* Download open tiles using geodata package to create a background with country limits
- *coord2sf* Spatializes coordinates : creates multipoint, linestring or polygons as sf objects from coordinates stored as numeric.

- *CustomizedProjectedCRS* Creates a customized planar crs for reprojecting the data in a cartesian system
- *df2sfp* Converts a data.frame to a sf multipoints object
- *lonlat2UTM* From Robin Lovelace, Geocomputation with R, see https://geocompr.robinlovelace.net/reproj-geo-data.html. Calculates the UTM EPSG code associated with any point on the planet
- *PathStudyBoundaries* Creates path boundaries with country or coastline limits defined as a polygon or linestring object. If country.ISO3 is defined, imports country boundaries tiles using geodata package and removes intersection with inland geometries
- *Pos2Path* Creates a linear path with linestrings from positions
- *SfBbox* Retrieves the bbox of a spatial object as a spatial polygon
- *sfp2df* Converts a sf multipoints object to a data.frame
- *Translate_Positions* Translate geographic positions by x and y after reprojecting the data in a cartesian system

2.4 Data-management tools

- *CalcThresholds* Calculates thresholds for values considering their distribution should be unimodal and gaussian, function is used internally in *Calc_NetsThresholds*
- *CleanSkewness* Removes the values that make a distribution skewed. At each step values over a threshold defined as min/max +/- range(values)/100 are dropped.
- *CleanSpuriousSpeeds* Calculates speed and cleans the spurious positions, a user-defined threshold being set for speed. The positions are removed iteratively until the computed speeds remain below the threshold.
- *CleanSpuriousValues* Cleans the data supposing its distribution should be unimodal and gaussian. Test wether the distribution is unimodal, if it isn't, the function will try to retrieve the dominant mode. Skewness is eventually corrected.
- *Cluster_Distributions* Clusters the values from the distribution they may belong
- *Detect_FishingTrips* Defines the fishing trip from positions. The function first cleans the positions and calculates speed then defines the fishing trips. 3 methods are available : it may either use a buffer with harbours or coasline and eventually adds a decision rule based on speed threshold.
- *Detect_Modes* Detects the possibles modes in values
- *Resample_Traj* Resamples linearily the positions with a constant timelapse between them. This function has to be applied by fishing trip, vessel and fishingtrip columns being identified as VESSEL_FK and FISHING_TRIP_FK.
- *Traj_Desc* Summarizes and sorts vessels and fishing trips by fishing trip starts

2.5 Movement features computation

- *CalcAcceleration* Calculates acceleration between two positions. By convention the acceleration is calculated between the position and the previous one Return the original sf object with a column "Acceleration" in m/s2.
- *CalcDist* Computes distances between positions in nautic miles, **by convention the path is defined between the position and the previous one**. Two methods have been implemented : "sf" uses great circle distance calculations with sf : :st_distance function for longlat coordinates, "nn2" uses RANN : :nn2 function for positions reprojected with cartesian coordinates. Function to be applied by fishing trip.
- *CalcFeatures* Calculates additional features describing movements to be used with machine-learning models : acceleration, proximity index, jerk, bearing rate, speed change, straigthness, sinuosity, turning angle, speed and direction change. if an harbour object is provided another covariate "PointInharbour" is added. A moving window may be provided

- *CalcHeading* Calculates heading between two positions in degrees
- *CalcSpeed* Calculates speed between two positions. By convention the speed is calculated between the position and the previous one. Returns the original sf object with a column "DIFFTIME.secs" in seconds, "SPEED.kn" in knots and eventually "DISTANCE.nm" in nautic miles if CalcDist hasn't been applied before
- *CalcTurningAngle* Calculates the turning angle between two positions relative to the previous one (in degrees). This function has to be applied by fishing trip, vessel and fishingtrip columns being identified as VESSEL_FK and FISHING_TRIP_FK
- *Process_TripPositions* An umbrella function to process the positions by fishing trips : Calculates speed and cleans the spurious positions, eventually redefines the fishing trips or resamples the positions. At last the heading is calculated and a linear path with linestrings is optionnaly retrieved from positions

2.6 Machine-learning & clustering

- *Clustering_BivariateBinary* Performs gaussian binary clustering applied to fishing trips using EMbC package
- *Clustering_KMeans* Creates and chooses the optimal number of classes to cluster the data using Mac-Queen's K-means algorithm
- *ContiguousSegments* Identifies contiguous segments from covariates, eg sequences of consecutive points which have the same combination of covariates
- *Detect_BehaviourChanges* Defines a non-supervised behavioural clustering based on speed, heading and turning angle. This function has to be applied by fishing trip
- *novelty_svm* Performs one-class svm for novelty detection
- *tune_RF* Automatic optimization of hyperparameters used in a random forest model (mtry and min.node.size). Two designs of nb.param X nb.param are tested using Out-of-Bag error estimate. The full design is then predicted from a random-forest model.

2.7 Geocomputation tools

- *Calc_NetsThresholds* Calculates plausible thresholds for speed, length and heading and, if available, soaking time
- *Consolidate_Nets* Consolidates Nets, output of Create_Nets function using thresholds on length, speed and heading This function must be applied by fishing trip
- *Create_Nets* Creates nets from positions flagged by an operation. Each net is identified as a spatial linestring with a unique identifier, the length, average heading and speed, hauling start and end. This function must be applied by fishing trip
- *Create_NetsByBoat* Creates nets from positions flagged by an operation. This function uses Create_Nets and Calc_NetsThresholds, it must be applied by fishing vessel for many fishing trips for the consolidation process to be relevant. This process may be parallelized by Fishing trip using parallelize argument if dedicated packages are available
- *Retrieve_Gear* Retrieves the gear identifier from Nets, output of Create_Nets function
- *Retrieve_SettingOperations* Retrieve setting operations from the positions using the nets. Retrieves the gear identifier from Nets, output of Create_Nets function then identifies the related setting events. Defines new columns in Nets objects to identify the setting : Fishing trip identifier, start and end and soaking time in hours
- *Set_NetThresholds* seeks for the best combination of arguments qt and iter to be used in Calc_NetsThresholds. Tests iteratively designs of values for qt and iter and selects their combination that maximizes the decay of thresholds.

2.8 Maps & plots

- *plot_Feature* plots boat paths described by a quantitative variable and optionally by a qualitative variable or the fishing operations
- *plot_Paths* maps boat paths described by a quantitative variable and optionally by a qualitative variable or the fishing operations
- *plot_Trip* plots boat paths described by a quantitative variable and the fishing operations and the evolution of features in time for the same trip see ?plot_Paths and ?plot_Feature

2.9 Links to vmstools

- *Positions2Tacsat* Converts fishing vessels positions to tacsat format used in vmstools. Positions are cleaned from duplicates and spurious positions based on speed. Speed and heading are calculated in knots and degrees
- *Sacrois2VmstoolsEflaloFormat* **under construction, not available yet** Function for converting French FIS consolidated logbooks format (SACROIS) to eflalo used in vsmtools.

3 - Case study : qualification of the positions of a vessel operating nets with sensors

3.1 Context

3.1.1 RECOPESCA program

Faced to the lack of data to accurately assess the spatial distribution of catches and fishing effort and for the environmental characterization of the fishing area, Ifremer has since 2005 implemented a project called 'Recopesca'. It is based on a sample of voluntary fishing vessels equipped with sensors coupled to on-board GPS monitors recording positions at least every fifteen minutes. The data gathered are used as a benchmark dataset for various research programs regarding to the use of geolocation information to monitor fisheries.

3.1.2 Issues when using sensors for passive gears

- The main difficulty related to the use of sensors for passive gear relies in the decorrelation between vessel and gear activities. On top of that, only some gears may be equipped, and the sensor gives sparse information on immersion/emersion but not the entire fishing operation. As a consequence, the information related to fishing activity can be described for less than 2% of the positions.

- Let's use the example provided in iapesca package to simulate a dataset qualified using sensors :

- Load the dataset provided in the package

```
data(positions)
```

- Set the seed for repetition of random processes (results of this tutorial will change if the seed is modified)

```
seed <- 221111
set.seed(seed)
```

- Summarize Fishing trips and events using *Traj_Desc* function

```
Traj_Desc(positions) %>%
  print_table(latex.options = c("HOLD_position"))
```

VESSEL_FK	FISHING_TRIP_FK	Start.FT	End.FT
NAVIRE_0075	18483451	2021-12-23 06 :55 :23	2021-12-24 07 :56 :17
NAVIRE_0075	18483452	2021-12-26 03 :00 :53	2021-12-27 12 :31 :47
NAVIRE_0075	18483453	2021-12-27 18 :46 :47	2021-12-29 21 :46 :47
NAVIRE_0075	18529291	2022-01-09 10 :43 :18	2022-01-13 18 :28 :23
NAVIRE_0075	18569373	2022-01-13 19 :43 :17	2022-01-16 12 :49 :23
NAVIRE_0075	18569374	2022-01-16 14 :19 :23	2022-01-17 10 :19 :23
NAVIRE_0075	18569376	2022-01-22 19 :46 :42	2022-01-27 17 :47 :41
NAVIRE_0075	18569377	2022-01-27 18 :02 :41	2022-01-30 11 :17 :41

```
Fop.desc <- Traj_Desc(positions, add.group = c("hauling", "setting"))

Fop.desc %>%
  head %>%
  print_table
```

VESSEL_FK	FISHING_TRIP_FK	hauling	setting	Start.FT	End.FT
NAVIRE_0075	18483451	NA	NA	2021-12-23 06 :55 :23	2021-12-23 14 :56 :17
NAVIRE_0075	18483451	NA	18483452_001	2021-12-23 15 :11 :14	2021-12-23 15 :41 :23
NAVIRE_0075	18483451	NA	18483452_005	2021-12-23 15 :56 :17	2021-12-23 16 :26 :14
NAVIRE_0075	18483451	NA	NA	2021-12-23 16 :41 :23	2021-12-23 20 :41 :48
NAVIRE_0075	18483451	NA	18483452_002	2021-12-23 20 :56 :42	2021-12-23 21 :26 :44
NAVIRE_0075	18483451	NA	NA	2021-12-23 21 :41 :23	2021-12-23 21 :41 :23

```
FishingEvents <- unique(Fop.desc$hauling)
FishingTrips <- unique(Fop.desc$FISHING_TRIP_FK)
```

- Select randomly one gear by fishing trip associated to the sensor and keep only one position qualified related to its immersion or emersion.

```
sel.gears <- do.call(c, lapply(FishingTrips, function(Ft){

   gears.FT <- Fop.desc$hauling[ Fop.desc$FISHING_TRIP_FK %in% Ft]
   gears.FT <- gears.FT [!is.na(gears.FT)]
   selected.net <- NULL

   if(length(gears.FT) > 0){
     selected.net <- sample(gears.FT, 1)
   }

}))

haul.index <- lapply(sel.gears, function(x) { which(positions$hauling %in% x)})
haul.ind2rm <- lapply(haul.index, function(x) { sample(x, length(x) -1 )})

set.index <- lapply(sel.gears, function(x) { which(positions$setting %in% x)})
set.ind2rm <- lapply(set.index, function(x) { sample(x, length(x) -1 )})
```

- Create a new variable named “Fop.sensor” from “FishingOperation” which compilates fishing informations from sensor only

```
positions$Fop.sensor <- as.character(positions$FishingOperation)

# Remove information from gears not equiped by sensors
sel.gears2rm <- !(positions$hauling %in% sel.gears | positions$setting %in% sel.gears)
positions$Fop.sensor[ sel.gears2rm ] <- NA

# Remove not selected positions from gears equiped by sensors
positions$Fop.sensor[do.call(c, haul.ind2rm)] <- NA
positions$Fop.sensor[do.call(c, set.ind2rm)] <- NA
```

- Fishing operation : 1937 positions qualified in the dataset

```
with(positions, table(FishingOperation))
```

```
FishingOperation
   Hauling NotFishing    Setting
       451       1368        118
```

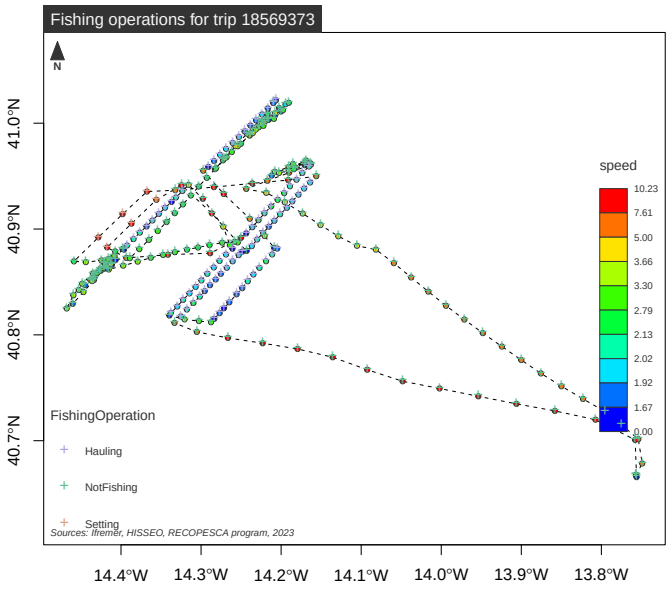
- Information retrieved from the sensors : **14 positions qualified only representing 0.7% of the whole dataset**

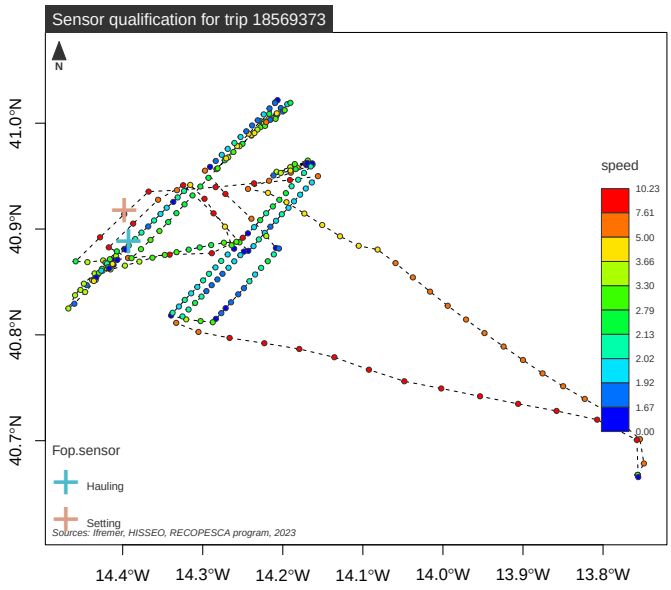
```
with(positions, table(FishingOperation, Fop.sensor))
```

```
                Fop.sensor
FishingOperation Hauling Setting
      Hauling          7       0
      NotFishing       0       0
      Setting          0       7
```

- Map the resulting paths for the trip starting on January 13th 2022 using the *plot_Paths* function
- The trip is selected and a sf object "pos.trip" is created from positions using *df2sfp* function

```
par(mfrow=c(1, 2))

trip <- FishingTrips[5]
pos.trip <- df2sfp(positions[ positions$FISHING_TRIP_FK %in% trip, ])

plot_Paths(pos.trip,
           quanti = "speed",
           FishingOp = "FishingOperation",
           main = sprintf("Fishing operations for trip %s", trip),
           cex.FoP = 1,
           Create.BgMap = FALSE)

plot_Paths(pos.trip,
            quanti = "speed",
            FishingOp = "Fop.sensor",
            main = sprintf("Sensor qualification for trip %s", trip),
            cex.FoP = 3,
            Create.BgMap = FALSE)
```



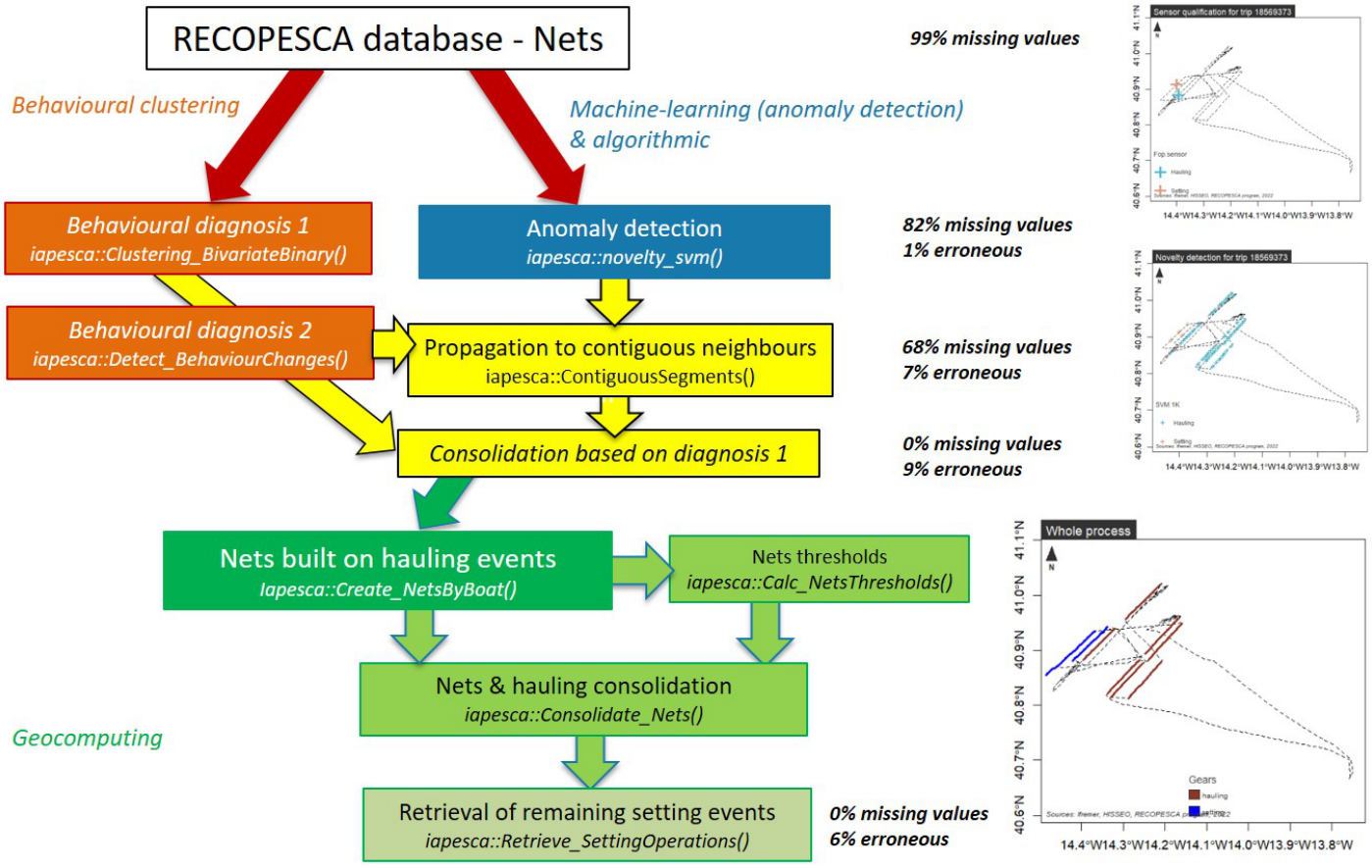


3.1.3 An algorithmic chain to qualify the positions using sensors

- An algorithmic chain for interpreting the sparse information was developed to post-qualify the dataset. It involves iterative steps combining behavioral description of the paths, anomaly detection, propagation and consolidation procedures. Geo-computing is finally used to consolidate the identified gears and to retrieve the setting operations.

- This process allowed the qualification of 95% of the positions using RECOPESCA dataset for vessels operating nets.

- Due to the process, an uncertainty remains regarding the qualification of the positions.



3.2 Behavioural annotation using binary clustering

- Performs gaussian binary clustering applied to fishing trips. The function *Clustering_BivariateBinary* uses EMbC : :stbc, a specific constructor for movement ecology purposes providing a bivariate speed/turn clustering.
- It returns the original object with new columns named “turn”, “speed”, “hdg” and “BinClust”
- This function must be applied by fishing trips.

```
positions <- do.call(rbind, lapply(FishingTrips, function(trip){

  pos.trip <- positions[ positions$FISHING_TRIP_FK %in% trip, ]
  binclust <- Clustering_BivariateBinary(pos.trip)
```

```
  return(binclust)

}))

head(positions)
```

```
print_table(
    setNames(
      doBy::summary_by(positions,
                  CreateFormula(c("speed", "turn"), "BinClust"),
                  FUN = function(x){ round(mean(x), digits = 2)}
                  ),
          c("BinClust", "mn.speed", "mn.turn")),
    latex.options = c("HOLD_position")
    )
```

BinClust	mn.speed	mn.turn
1	2.38	0.05
2	2.12	1.86
3	6.90	0.04
4	6.43	1.49
5	0.00	0.00

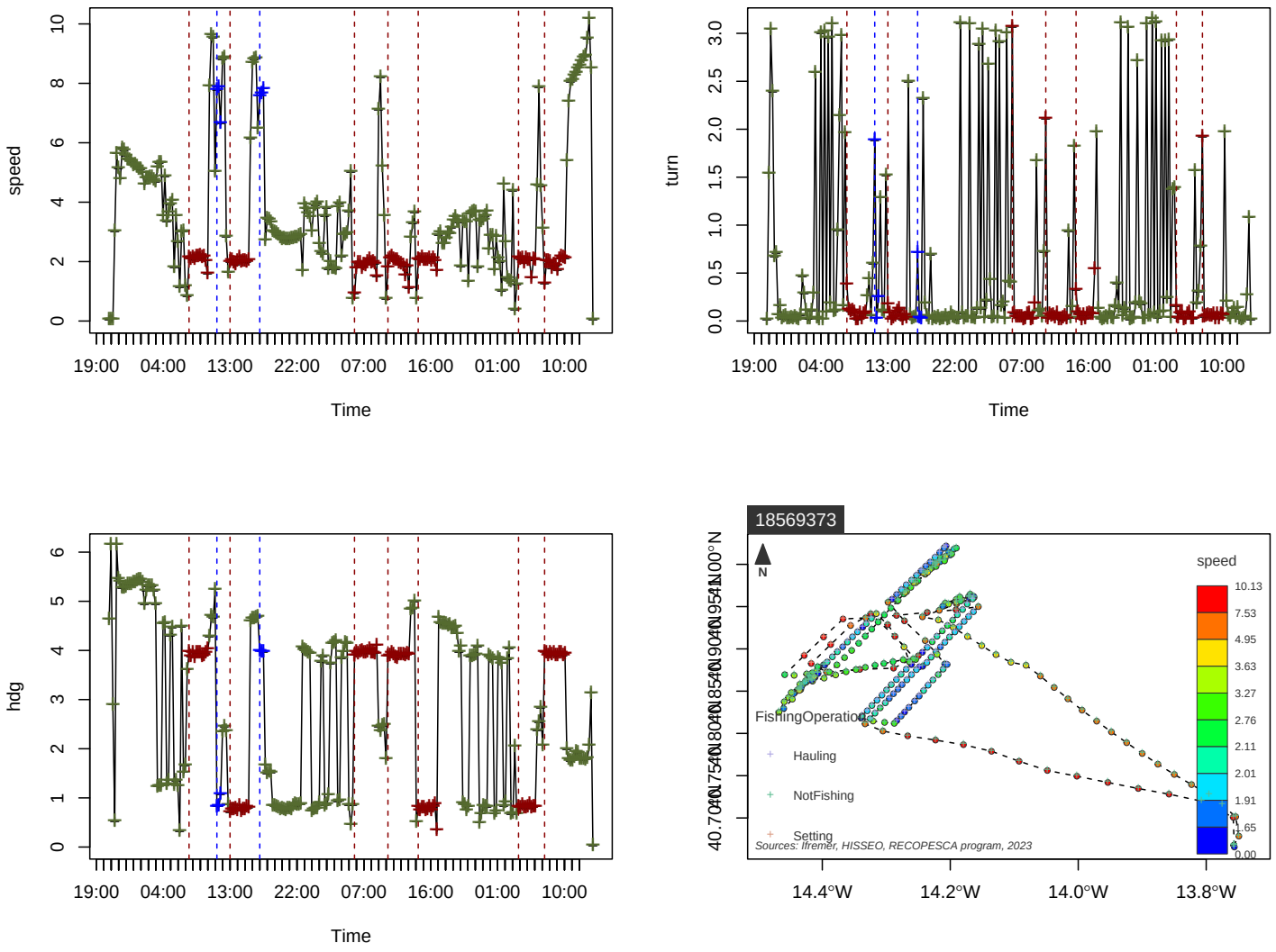
- The cluster 1 describes straight paths with low speed, 2 turning paths, 3 straight paths with high speed and 4 high speed and turning paths.

```
with(positions, table(BinClust, FishingOperation))
```

```
        FishingOperation
BinClust Hauling NotFishing Setting
       1     423        562       2
       2      27        361       6
       3       0        330      83
       4       1        107      27
       5       0          8       0
```

- BinClust 1 may be associated to hauling, and eventually 2
- BinClust 3 to setting, and eventually 4
- This relation is not exclusive : not fishing modality may also belong to these class !
- Map the paths and plot the features calculated by *Clustering_BivariateBinary* function in relation to fishing operations for the trip starting on January 13th 2022 using the *plot_Trip* function
- The trip is selected and a sf object "pos.trip" is created from positions using *df2sfp* function

```
trip <- FishingTrips[5]
pos.trip <- df2sfp(positions[ positions$FISHING_TRIP_FK %in% trip, ])

plot_Trip(pos.trip,
          col.features = c("speed", "turn", "hdg"),
          col.FishingOp = "FishingOperation",
          quanti = "speed")
```



- 9 fishing operations for this fishing trip : 7 nets are hauled and 2 are set.

3.3 Novelty detection

- This step uses one-class svm method for novelty detection implemented in *novelty_svm*. This method requires e1071 : :svm function with argument type = "one-classification".
- Novelty detection methods can be used to predict anomalies such as spams or fraud detection. Other methods could be used like isolation forest, elliptic envelope or local outlier factor algorithms.
- In our case, these methods are interesting because we try to detect similar events from sparse and partial data (no information is available for not fishing events).
- From the 7 known events by fishing operation, two models will be built using turn, speed and heading as covariates (features calculated using the *Clustering_BivariateBinary* function). The parameter "nu" in the function could be optimized.

```
hauling.svm <- novelty_svm(data = positions,
                           category = "Hauling",
                           y = "Fop.sensor",
                           covar = c( "turn" ,  "speed", "hdg"),
                           pred = TRUE,
                           nu = 0.1)
```

```
setting.svm <- novelty_svm(data = positions,
                           category = "Setting",
                           y = "Fop.sensor",
                           covar = c( "turn" ,  "speed", "hdg"),
                           pred = TRUE,
                           nu = 0.1)
```

- Results for hauling.svm model

```
hauling.svm$svm.model
```

```

Call:
svm.default(x = train.dat, y = NULL, scale = FALSE, type = "one-classification", kernel = kernel, gam


Parameters:
   SVM-Type:  one-classification
 SVM-Kernel:  radial
      gamma:  0.3333333
         nu:  0.1

Number of Support Vectors:  4
```

```
table(positions$Fop.sensor, hauling.svm$pred)
```

```
          FALSE TRUE
  Hauling     1    6
  Setting     7    0
```

```
table(positions$FishingOperation, hauling.svm$pred)
```

```
             FALSE TRUE
  Hauling      152  299
  NotFishing  1355   13
  Setting      118    0
```

- 312 positions may be qualified as hauling events on which 13 only are false positives. 1 on 7 positions aren't recognized by the model as "hauling".
- Results for setting.svm model

```
setting.svm$svm.model
```

```

Call:
svm.default(x = train.dat, y = NULL, scale = FALSE, type = "one-classification", kernel = kernel, gam
```

```
Parameters:
   SVM-Type:  one-classification
 SVM-Kernel:  radial
      gamma:  0.3333333
         nu:  0.1

Number of Support Vectors:  5
```

```
table(positions$Fop.sensor, setting.svm$pred)
```

```
          FALSE TRUE
  Hauling     7    0
  Setting     2    5
```

```
table(positions$FishingOperation, setting.svm$pred)
```

```
             FALSE TRUE
  Hauling      451    0
  NotFishing  1363    5
  Setting       85   33
```

- 38 positions may be qualified as setting events on which 5 only are false positives. 2 on 7 positions aren't recognized by the model as "setting".
- Save the novelty detection models results in a new variable named "SVM.1K"

```
positions$SVM.1K <- rep(NA, nrow(positions))
positions$SVM.1K[hauling.svm$pred] <- "Hauling"
positions$SVM.1K[setting.svm$pred] <- "Setting"
```

- Add the FALSE negatives from sensors

```
sel.missing <- is.na(positions$SVM.1K) & !is.na(positions$Fop.sensor)
positions$SVM.1K[ sel.missing ] <- positions$Fop.sensor[ sel.missing ]
```

- View results of novelty detection step

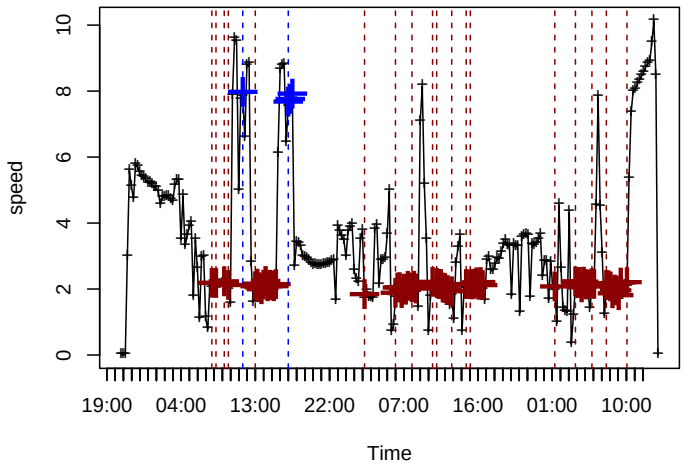
```
with(positions, table(FishingOperation, SVM.1K))
```

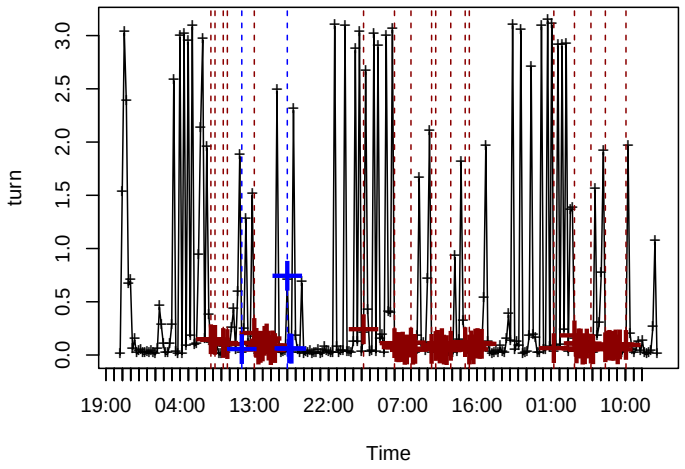
```
                SVM.1K
FishingOperation Hauling Setting
      Hauling        300       0
      NotFishing      13       5
      Setting          0      35
```

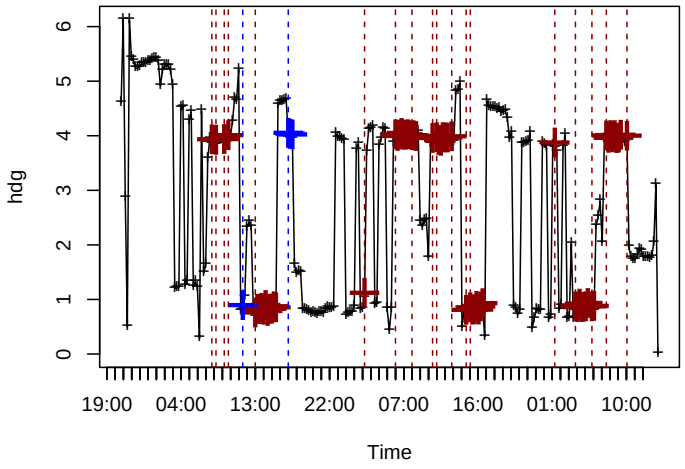
- Information retrieved from novelty detection step : **353 positions qualified representing 18.2% of the whole dataset**, 5.1% of them being erroneous.

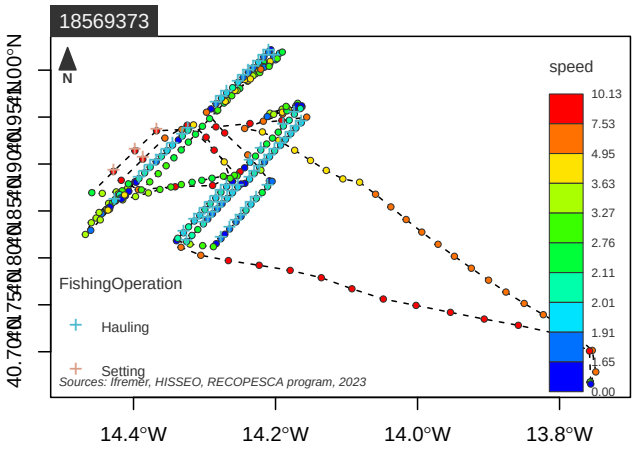
- Map the paths and plot the features in relation to fishing events detected by novelty detection for the trip starting on January 13th 2022 using the *plot_Trip* function
- The trip is selected and a sf object "pos.trip" is created from positions using *df2sfp* function

```
trip <- FishingTrips[5]
pos.trip <- df2sfp(positions[ positions$FISHING_TRIP_FK %in% trip, ])

plot_Trip(pos.trip,
          col.features = c("speed", "turn", "hdg"),
          col.FishingOp = "SVM.1K",
          quanti = "speed")
```









3.4 Propagation using a non supervized behavioural diagnosis

3.4.1 Use of *Detect_BehaviourChanges* function

- This function defines a non-supervised behavioural clustering based on speed, heading and turning angle. This function has to be applied by fishing trip.
- In object "trip.path", it returns the original sf object with "Clust.Pass", the value of behavioural cluster and "Pass.number" identifying contiguous positions with similar direction and another object named "ClustDesc", a data.frame with clusters characteristics.

- Example for the trip starting on January 13th 2022 : 18569373
- The trip is selected and a sf object "pos.trip" was previsouly created from positions using *df2sfp* function

```
pos.localbehav <- Detect_BehaviourChanges(pos.trip)$trip.path
pos.localbehav %>% class
```

```
[1] "sf"         "data.frame"
```

```
pos.localbehav %>% colnames
```

```
 [1] "VESSEL_FK"        "FISHING_TRIP_FK"  "DATE_TIME"        "setting"          "hauling"          "
[10] "hdg"              "BinClust"         "SVM.1K"           "DISTANCE.nm"      "DIFFTIME.secs"    "
[19] "Clust.Pass"       "geometry"
```

- View the interaction between Clust.Pass and fishing operations

```
with(pos.localbehav, table(FishingOperation, Clust.Pass))
```

```
                Clust.Pass
FishingOperation  1  2  3  4  5
      Hauling    32  0 45  3  0
      NotFishing 45 18 67 15 30
      Setting     2  1  3  0  0
```

- As for binary clustering, this relation is not exclusive.
- The "Pass.number" variable identifying contiguous points having the same direction and "Clust.Pass", their behaviour, we may guess, using the interaction between these variables that, if one point is qualified as a fishing operation its neighbours are also related to the same fishing operation.
- We first create a new variable "Segments" combining "Pass.number" and "Clust.Pass" using function *ContiguousSegments*. This function identifies contiguous segments from covariates.

```
pos.localbehav <- ContiguousSegments(data = pos.localbehav,
                                     covar = c("Pass.number", "Clust.Pass"))
pos.localbehav %>% colnames
```

```
 [1] "VESSEL_FK"        "FISHING_TRIP_FK"  "DATE_TIME"        "setting"          "hauling"          "
[10] "hdg"              "BinClust"         "SVM.1K"           "DISTANCE.nm"      "DIFFTIME.secs"    "
[19] "Clust.Pass"       "Segments"         "geometry"
```

```
pos.localbehav[1:20, 18:20] %>%
  print_table(latex.options = c("HOLD_position"))
```

Pass.number	Clust.Pass	Segments	geometry
01	3	1	POINT (-13.75669 40.66761)
02	2	2	POINT (-13.7567 40.6676)
03	4	3	POINT (-13.7567 40.66763)
04	2	4	POINT (-13.7567 40.66762)
05	1	5	POINT (-13.74899 40.67847)
06	3	6	POINT (-13.7539 40.70142)
06	3	6	POINT (-13.77511 40.7154)
06	3	6	POINT (-13.7956 40.72757)
06	3	6	POINT (-13.82287 40.73946)
06	3	6	POINT (-13.84985 40.75145)
06	3	6	POINT (-13.87551 40.76359)
06	3	6	POINT (-13.89999 40.77622)
06	3	6	POINT (-13.92423 40.78897)
06	3	6	POINT (-13.94808 40.80168)
06	3	6	POINT (-13.97112 40.81448)
06	3	6	POINT (-13.994 40.82748)
06	3	6	POINT (-14.01607 40.84086)
06	3	6	POINT (-14.03758 40.85429)
06	3	6	POINT (-14.05915 40.8678)
06	3	6	POINT (-14.0816 40.88059)

- From the assumption used for propagation, we may identify fishing segments :

```
FishingSegs <- unique(pos.localbehav$Segments[!is.na(pos.localbehav$SVM.1K)])
tab.op <- with(pos.localbehav[!is.na(pos.localbehav$SVM.1K), ], table(Segments, SVM.1K))
trans.codes <- data.frame(FishingSegs = FishingSegs,
                          Fop = colnames(tab.op)[apply(tab.op, 1, which.max)])
trans.codes %>%
  print_table(latex.options = c("HOLD_position"))
```

FishingSegs	Fop
22	Hauling
25	Setting
28	Hauling
31	Setting
42	Hauling
52	Hauling
56	Hauling
59	Hauling
73	Hauling
82	Hauling
86	Hauling

- 11 fishing segments detected that might be linked to different fishing operations
- Having the same behaviours, the points belonging to these segments are linked to related fishing operation.
- A new variable “Fop.pred” is created to store these informations.

```
pos.localbehav$Fop.pred <- pos.localbehav$Segments
pos.localbehav$Fop.pred[ !pos.localbehav$Fop.pred %in% FishingSegs] <- NA
pos.localbehav$Fop.pred <- factor(pos.localbehav$Fop.pred)
levels(pos.localbehav$Fop.pred) <- trans.codes$Fop
pos.localbehav$Fop.pred <- as.character(pos.localbehav$Fop.pred)
with(pos.localbehav, table(FishingOperation, SVM.1K))
```

```
                SVM.1K
FishingOperation Hauling Setting
      Hauling         59       0
      NotFishing       2       0
      Setting          0       4
```

```
with(pos.localbehav, table(FishingOperation, Fop.pred))
```

```
                Fop.pred
FishingOperation Hauling Setting
      Hauling         77       0
      NotFishing       7       0
      Setting          0       5
```

- Using this propagation based on behavioural diagnosis, 19 new positions have been qualified with 5 new erroneous qualifications due the propagation of initial erroneous hauling events.

3.4.2 Application of the propagation to the whole dataset

- The *Detect_BehaviourChanges* function is applied to the whole dataset by Fishing trip

```
Propagation_fun <- function(trip){

        pos.trip <- df2sfp(positions[ positions$FISHING_TRIP_FK %in% trip, ])
        pos.localbehav <- ContiguousSegments(
                            Detect_BehaviourChanges(pos.trip)$trip.path,
                            covar = c("Pass.number", "Clust.Pass"))

        FishingSegs <- unique(pos.localbehav$Segments[!is.na(pos.localbehav$SVM.1K)])

        tab.op <- with(pos.localbehav[!is.na(pos.localbehav$SVM.1K), ],
                       table(Segments, SVM.1K))
        trans.codes <- data.frame(FishingSegs = FishingSegs,
                                  Fop = colnames(tab.op)[apply(tab.op, 1, which.max)])

        pos.localbehav$Fop.pred <- pos.localbehav$Segments
        pos.localbehav$Fop.pred[ !pos.localbehav$Fop.pred %in% FishingSegs] <- NA
        pos.localbehav$Fop.pred <- factor(pos.localbehav$Fop.pred )
        levels(pos.localbehav$Fop.pred) <- trans.codes$Fop
        pos.localbehav$Fop.pred <- as.character(pos.localbehav$Fop.pred)

        return(pos.localbehav)
}
```

```
pos.sf <- do.call(rbind,
                  lapply(FishingTrips,
                         Propagation_fun))

with(pos.sf, table(FishingOperation, Fop.pred))
```

```
                Fop.pred
FishingOperation Hauling Setting
      Hauling        433       0
      NotFishing     103      35
      Setting          0      57
```

3.5 Consolidation using binary clustering behavioural diagnosis

- BinClust results may be used to remove some False positives :

```
pos.sf$Fop.pred [pos.sf$Fop.pred %in% "Setting" & !pos.sf$BinClust %in% c(3, 4)] <- NA
pos.sf$Fop.pred [pos.sf$Fop.pred %in% "Hauling" & !pos.sf$BinClust %in% c(1, 2)] <- NA

with(pos.sf, table(FishingOperation, Fop.pred))
```

```
                Fop.pred
FishingOperation Hauling Setting
      Hauling        433       0
      NotFishing      82      15
      Setting          0      55
```

- "NotFishing" category can now be affected

```
pos.sf$Fop.pred [is.na(pos.sf$Fop.pred )] <- "NotFishing"
```

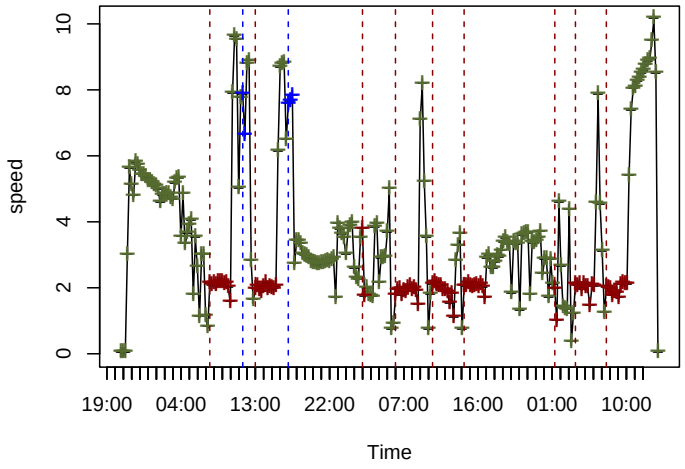
- Results of the qualification :

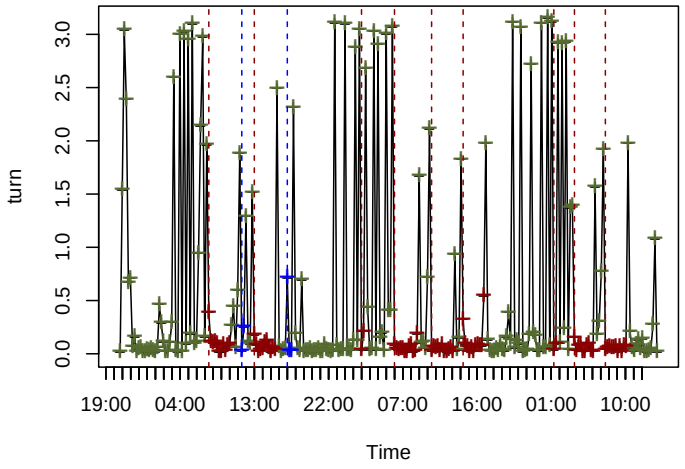
```
tab.Fopres <- with(pos.sf, table(FishingOperation, Fop.pred))
GobalAccuracy <- Calc_Acc(tab.Fopres)
tab.Fopres
```

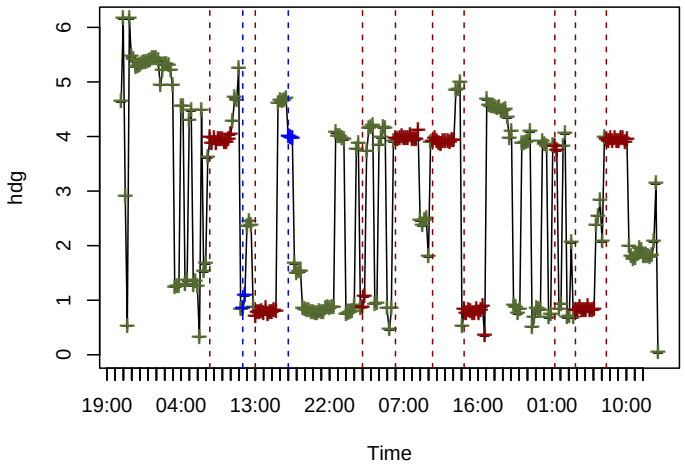
```
                Fop.pred
FishingOperation Hauling NotFishing Setting
      Hauling        433         18       0
      NotFishing      82       1271      15
      Setting          0         63      55
```

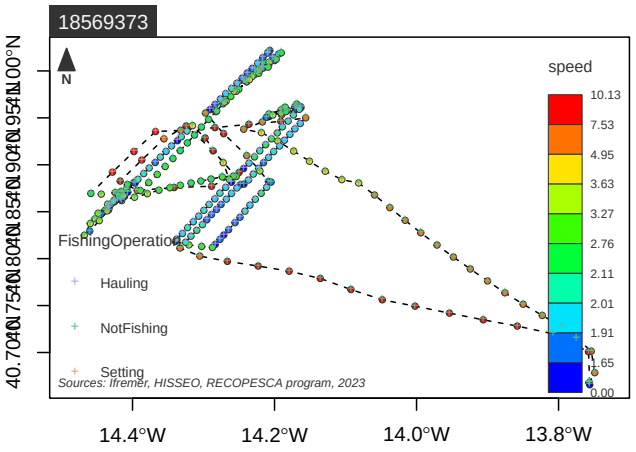
- Information retrieved from novelty detection step and use of behavioural diagnosis : **whole dataset qualified**, with 9.19% of the positions being erroneous.
- Map the paths and plot the features in relation to fishing events detected for the trip starting on January 13th 2022 using the *plot_Trip* function

```
pos.trip <- pos.sf[ pos.sf$FISHING_TRIP_FK %in% trip, ]

plot_Trip(pos.trip,
          col.features = c("speed", "turn", "hdg"),
          col.FishingOp = "Fop.pred",
          quanti = "speed")
```









3.6 Consolidation using geocomputation

- Nets are created from hauling events using the umbrella function *Create_NetsByBoat*. This function is applied by trip and the process may be parallelized.
- The propagation process involving the use of *Detect_BehaviourChanges* the function used when the method "Use.BehaviourChanges" is chosen, this argument has been set to FALSE. If set to TRUE, nets are built from the combination of hauling sequences and "Clust.Pass" variable, the non-supervized behavioural diagnosis.
- A consolidation process is applied internally using the *Consolidate_Nets* function. It uses statistics on nets dimension, heading and speed to remove sequences that built from false positives detection. If these statistics are not available, their are built using the observed values from created nets. This method is used when argument "Auto.ThreshHolds.Detection" is set to TRUE. In this case arguments "qt", defining the gaussian quantile for retrieving thresholds has been set to 0.98 and "iter" the number of iterations to "1"

(see arguments of *Calc_NetsThresholds*). If left unprovided these arguments are set using *Set_NetThresholds* function which tests iteratively designs of values and choose their combination that maximizes the decay of thresholds.

- From the thresholds defined using these arguments, nets can be removed, or corrected (paste or split).
- The "parallelize" option is set to TRUE, packages parallel, foreach and doParallel must have been installed. If left unprovided "nCores" argument is set to half the available cores on the computer.

```
Nets.pred <- Create_NetsByBoat(traj = pos.sf,
                               Col.Fop = "Fop.pred",
                               Fop.category = "Hauling",
                               Use.BehaviourChanges = FALSE,
                               Auto.ThreshHolds.Detection = TRUE,
                               qt = 0.98, iter = 1,
                               parallelize = TRUE,
                               verbose = TRUE
                              )
```

- The object "Nets.pred" contains "Nets" a list of spatial linestring objects of class sf, each of them being the output of Create_Nets function applied to the fishing trips.

```
Nets.pred$Nets %>% names
```

```
[1] "18483452" "18483453" "18529291" "18569373" "18569374" "18569376" "18569377"
```

```
Nets.pred$Nets %>% class
```

```
[1] "list"
```

```
Nets.pred$Nets[[1]] %>% class
```

```
[1] "sf"         "data.frame"
```

```
Nets.pred$Nets[[1]] %>%
  sf::st_set_geometry(NULL) %>%
  print_table
```

Net	length	heading	speed	hauling.start	hauling.end	its	FISHING_TRIP_FK
18483452_01	10 543.377 [m]	47.28780	1.938857	2021-12-26 06 :24 :38	2021-12-26 09 :15 :56	0	18483452
18483452_02	10 135.940 [m]	226.79877	2.093160	2021-12-26 11 :54 :13	2021-12-26 14 :30 :38	0	18483452
18483452_03	9 590.553 [m]	227.34743	1.865010	2021-12-27 01 :47 :04	2021-12-27 04 :30 :55	0	18483452
18483452_04	9 412.996 [m]	47.96498	1.821342	2021-12-27 06 :32 :05	2021-12-27 09 :15 :56	0	18483452

- 40 nets are detected for these 7 fishing trips with hauling events, representing a total of 417443 meters
- The second object "Nets.Thresh" is a list of thresholds, output of Calc_NetsThresholds function
- Nets consolidation is built from these thresholds.

```
Nets.pred$Nets.Thresh
```

```
$LimVal.Speed
[1] 1.216852 3.020310

$LimVal.Length
[1]  4319.147 15055.832

$LimVal.Heading
          [,1]      [,2]
[1,]  32.03766  56.07869
[2,] 220.80288 232.87585

$LimVal.SoakTime
NULL
```

- Real nets are built from hauling events identified in "FishingOperation" column

```
Nets.real <- Create_NetsByBoat(traj = pos.sf,
                               Col.Fop = "FishingOperation",
                               Fop.category = "Hauling",
                               Use.BehaviourChanges = FALSE,
                               Auto.ThreshHolds.Detection = TRUE,
                               qt = 0.999, iter = 1,
                               parallelize = TRUE,
                               verbose = TRUE
)
```

- 39 nets are detected for these 7 fishing trips with hauling events, representing a total of 394811 meters
- Following thresholds being set :

```
Nets.real$Nets.Thresh
```

```
$LimVal.Speed
[1] 1.678635 2.202641

$LimVal.Length
[1]  8889.193 11865.144

$LimVal.Heading
          [,1]      [,2]
[1,]  40.95072  50.13776
[2,] 220.74699 232.31793

$LimVal.SoakTime
NULL
```

3.7 Retrieval of remaining setting events using geocomputation

- As setting events are less characterizable than hauling, in particular because they are carried out at higher speeds involving fewer positions, the geocomputation can help in identifying these sequences.
- The umbrella function "Retrieve_SettingOperations" has been built for that purpose. This function returns two objects :
  - nets, a spatial linestring object of class sf with additional columns to describe setting events : setting trip, start and end and soaking time.
  - traj, the initial positions as sf object with new columns identifying hauling and setting operations with their gear identifier.
- Buffers and scoring are used to retrieve the setting events. Based on a fully qualified dataset, they have been set to 100 m for hauling events and 330 m for setting events up to argument "buffer.max" (DEFAULT 550 m) if no candidates for setting are identified.
- If the argument "remove.orphans" is set to TRUE, nets for which no related setting events have been found are removed.
- Setting events are retrieved from predicted and qualified fishing operations.

```
Setting.Pred <- Retrieve_SettingOperations(traj = pos.sf,
                                           ls.nets = Nets.pred$Nets,
                                           Col.Fop = "Fop.pred",
                                           Setting.category = "Setting",
                                           Hauling.category = "Hauling",
                                           remove.orphans = TRUE)

Setting.Real <- Retrieve_SettingOperations(traj = pos.sf,
                                           ls.nets = Nets.real$Nets,
                                           Col.Fop = "FishingOperation",
                                           Setting.category = "Setting",
                                           Hauling.category = "Hauling")
```

```
names(Setting.Pred)
```

```
[1] "nets" "traj"
```

```
Setting.Pred %>% str
```

```
List of 2
 $ nets:Classes 'sf' and 'data.frame':  40 obs. of  13 variables:
  ..$ Net              : chr [1:40] "18483452_01" "18483452_02" "18483452_03" "18483452_04" ...
  ..$ length           : Units: [m] num [1:40] 10543 10136 9591 9413 10273 ...
  ..$ heading          : num [1:40] 47.3 226.8 227.3 48 46 ...
  ..$ speed            : num [1:40] 1.94 2.09 1.87 1.82 1.77 ...
  ..$ hauling.start    : chr [1:40] "2021-12-26 06:24:38" "2021-12-26 11:54:13" "2021-12-27 01:47:04
  ..$ hauling.end      : chr [1:40] "2021-12-26 09:15:56" "2021-12-26 14:30:38" "2021-12-27 04:30:55
  ..$ its              : int [1:40] 0 0 0 0 0 0 0 0 0 0 ...
  ..$ FT_Hauling       : chr [1:40] "18483452" "18483452" "18483452" "18483452" ...
  ..$ setting.start    : chr [1:40] "2021-12-23 15:11:14" "2021-12-23 21:11:42" "2021-12-26 18:01:47
  ..$ setting.end      : chr [1:40] "2021-12-23 15:41:23" "2021-12-23 21:41:23" "2021-12-26 18:46:48
  ..$ FT_Setting       : chr [1:40] "18483451" "18483451" "18483452" "18483451" ...
```

```
  ..$ soaking.time.hours: num [1:40] 67.1 66.3 11.5 90.3 103.3 ...
  ..$ geometry          :sfc_LINESTRING of length 40; first list element:  'XY' num [1:24, 1:2] -14.2
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:24] "1" "2" "3" "4" ...
  .. .. ..$ : chr [1:2] "X" "Y"
  ..- attr(*, "sf_column")= chr "geometry"
  ..- attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA NA NA NA ...
  .. ..- attr(*, "names")= chr [1:12] "Net" "length" "heading" "speed" ...
 $ traj:Classes 'sf' and 'data.frame':  1937 obs. of  23 variables:
  ..$ VESSEL_FK      : chr [1:1937] "NAVIRE_0075" "NAVIRE_0075" "NAVIRE_0075" "NAVIRE_0075" ...
  ..$ FISHING_TRIP_FK: chr [1:1937] "18483451" "18483451" "18483451" "18483451" ...
  ..$ DATE_TIME      : chr [1:1937] "2021-12-23 06:55:23" "2021-12-23 06:55:44" "2021-12-23 07:11:17"
  ..$ setting        : chr [1:1937] NA NA NA NA ...
  ..$ hauling        : chr [1:1937] NA NA NA NA ...
  ..$ Fop.sensor     : chr [1:1937] NA NA NA NA ...
  ..$ turn           : num [1:1937] 0 0.261 1.881 1.392 1.051 ...
  ..$ speed          : num [1:1937] 0.18357 0.00588 0.31833 5.71063 8.16105 ...
  ..$ hdg            : num [1:1937] 3.614 3.875 1.994 0.602 5.835 ...
  ..$ BinClust       : num [1:1937] 1 1 2 4 4 4 3 3 3 3 ...
  ..$ SVM.1K         : chr [1:1937] NA NA NA NA ...
  ..$ DISTANCE.nm    : num [1:1937] 0 0.00107 0.00152 0.07949 1.42131 ...
  ..$ DIFFTIME.secs  : num [1:1937] 0 21 933 900 897 909 894 900 900 897 ...
  ..$ SPEED.kn       : num [1:1937] 0 0.18337 0.00588 0.31797 5.70425 ...
  ..$ HEADING.deg    : num [1:1937] 243.7 228.7 336.7 56.2 116.6 ...
  ..$ TURN.deg       : num [1:1937] 0 15 108 79.6 60.4 ...
  ..$ Pass.number    : chr [1:1937] "01" "01" "02" "03" ...
  ..$ Clust.Pass     : chr [1:1937] "4" "4" "5" "1" ...
  ..$ Segments       : num [1:1937] 1 1 2 3 4 5 5 5 5 5 ...
  ..$ Fop.pred       : chr [1:1937] "NotFishing" "NotFishing" "NotFishing" "NotFishing" ...
  ..$ Hauling_GearId : chr [1:1937] NA NA NA NA ...
  ..$ Setting_GearId : chr [1:1937] NA NA NA NA ...
  ..$ geometry       :sfc_POINT of length 1937; first list element:  'XY' num [1:2] -13.8 40.7
  ..- attr(*, "sf_column")= chr "geometry"
  ..- attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA NA NA NA ...
  .. ..- attr(*, "names")= chr [1:22] "VESSEL_FK" "FISHING_TRIP_FK" "DATE_TIME" "setting" ...
```

```
Nets.pred <- Setting.Pred$nets
Nets.real <- Setting.Real$nets
```

- Using the function *Calc_NetsThresholds*, new thresholds for soaking times can now be defined

```
Threshs <- Calc_NetsThresholds(Nets.pred, qt = 0.98, iter = 1)

Nets.pred %>%
  sf::st_set_geometry(NULL) %>%
  summary
```

```
     Net                length          heading          speed       hauling.start      hauling.end
 Length:40          Min.   : 8889   Min.   : 36.05   Min.   :1.531   Length:40          Length:40
 Class :character   1st Qu.: 9643   1st Qu.: 45.91   1st Qu.:1.859   Class :character   Class :charac
 Mode  :character   Median :10101   Median :223.24   Median :1.927   Mode  :character   Mode  :charac
                    Mean   :10436   Mean   :140.22   Mean   :1.943
```

```
                  3rd Qu.:10725   3rd Qu.:226.41   3rd Qu.:2.034
                  Max.   :13428   Max.   :230.03   Max.   :2.380
 FT_Setting        soaking.time.hours
Length:40          Min.   :  4.133
Class :character   1st Qu.: 58.843
Mode  :character   Median : 67.747
                   Mean   : 61.454
                   3rd Qu.: 70.531
                   Max.   :110.098
```

```
Threshs
```

```
$LimVal.Speed
[1] 1.531469 2.379840

$LimVal.Length
[1]  8889.193 12741.755

$LimVal.Heading
          [,1]      [,2]
[1,]  40.96202  50.38158
[2,] 222.51528 229.29220

$LimVal.SoakTime
[1]  20.03274 110.09833
```

```
Nets.real %>%
  sf::st_set_geometry(NULL) %>%
  summary
```

```
    Net                length         heading          speed       hauling.start      hauling.end
Length:39          Min.   : 8889   Min.   : 41.98   Min.   :1.679   Length:39          Length:39
Class :character   1st Qu.: 9642   1st Qu.: 45.43   1st Qu.:1.871   Class :character   Class :charac
Mode  :character   Median :10133   Median :222.39   Median :1.939   Mode  :character   Mode  :charac
                   Mean   :10123   Mean   :138.40   Mean   :1.952
                   3rd Qu.:10480   3rd Qu.:226.61   3rd Qu.:2.056
                   Max.   :11523   Max.   :230.07   Max.   :2.203
 FT_Setting        soaking.time.hours
Length:39          Min.   : 39.13
Class :character   1st Qu.: 64.69
Mode  :character   Median : 68.22
                   Mean   : 68.97
                   3rd Qu.: 71.94
                   Max.   :110.09
```

- Predicted values from sensors can now be updated in a new column named "Fop.predGeo"

```
pos.sf$Fop.predGeo <- rep(NA, nrow(pos.sf))
pos.sf$Fop.predGeo[ !is.na( Setting.Pred$traj$Hauling_GearId)] <- "Hauling"
pos.sf$Fop.predGeo[ !is.na( Setting.Pred$traj$Setting_GearId)] <- "Setting"
pos.sf$Fop.predGeo[ is.na(pos.sf$Fop.predGeo)] <- "NotFishing"
```

```
with(pos.sf, table(FishingOperation, Fop.pred))
```

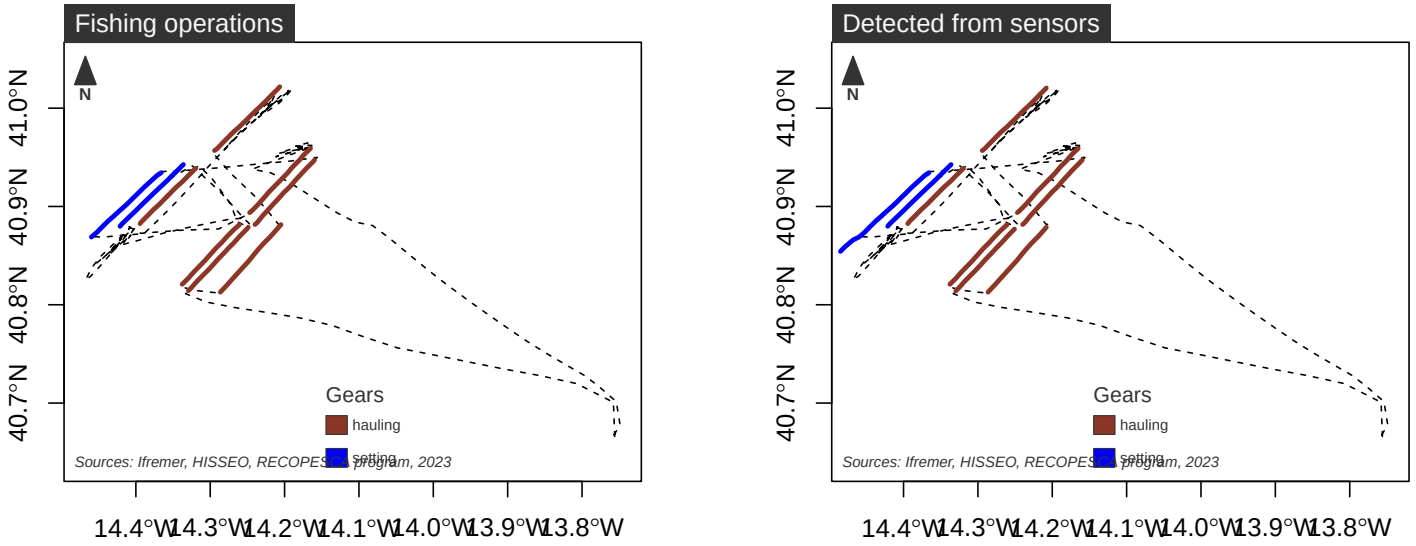
```
                Fop.pred
FishingOperation Hauling NotFishing Setting
      Hauling        433         18       0
      NotFishing      82       1271      15
      Setting          0         63      55
```

```
with(pos.sf, table(FishingOperation, Fop.predGeo))
```

```
                Fop.predGeo
FishingOperation Hauling NotFishing Setting
      Hauling        433         18       0
      NotFishing      39       1296      33
      Setting          0         27      91
```

- This geocomputation process increases the accuracy of qualification from 90.81 % to 93.96 %.
- It reduces the rate of FALSE positives and increases by 65% the number of true setting operations detected.
- Map the gears created for fishing trip 18569373 using *plot_Paths* function.

```
par(mfrow=c(1, 2))

trip <- FishingTrips[5]
pos.trip <- pos.sf[ pos.sf$FISHING_TRIP_FK %in% trip, ]

plot_Paths(pos.trip,
           FishingOp = "FishingOperation",
           quanti = NULL,
           main = "Fishing operations",
           gears.set = Nets.real[ Nets.real$FT_Setting %in% trip, ],
           gears.hauled = Nets.real[ Nets.real$FT_Hauling %in% trip, ],
           Create.BgMap = FALSE)

plot_Paths(pos.trip,
           FishingOp = "Fop.predGeo",
           quanti = NULL,
           main = "Detected from sensors",
           gears.set = Nets.pred[ Nets.pred$FT_Setting %in% trip, ],
           gears.hauled = Nets.pred[ Nets.pred$FT_Hauling %in% trip, ],
           Create.BgMap = FALSE)
```



- Using thresholds on soaking time, the consolidation could be improved. It wasn't included by default in *Create_NetsByBoat* for the calculation to remain reasonable (Nets have to be created two times when using the *Consolidate_Nets* methods within this function).
- In order to build a clean dataset for qualification, this step could be included defining thresholds using nested iapesca functions :

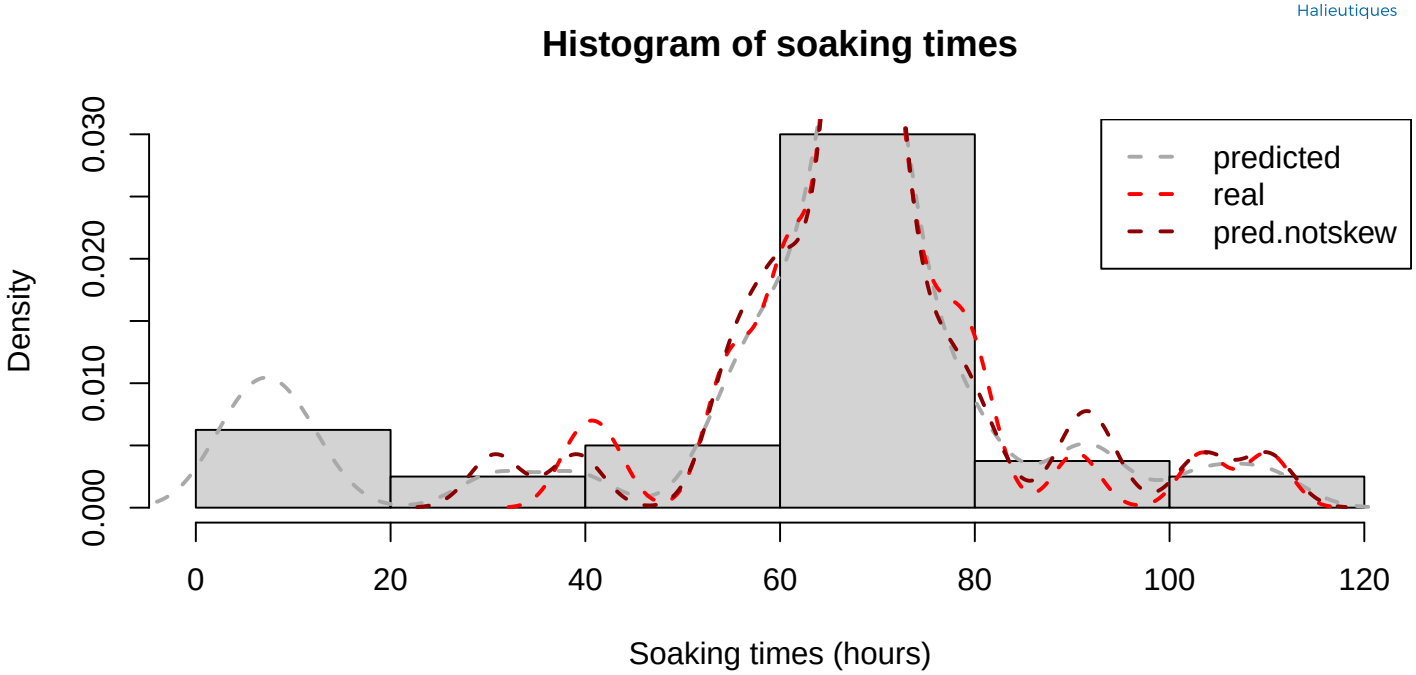
```
summary(CleanSpuriousValues(Nets.pred$soaking.time.hours))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  54.24   66.32   68.23   67.41   69.63   81.10      11
```

```
summary(CleanSkewness(Nets.pred$soaking.time.hours))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  30.75   63.85   68.24   69.18   71.94  110.10       5
```

```
is.skew <- is.na(CleanSkewness(Nets.pred$soaking.time.hours))

hist(Nets.pred$soaking.time.hours, freq=FALSE,
     main = "Histogram of soaking times", xlab = "Soaking times (hours)")
lines(density(Nets.pred$soaking.time.hours), col = "darkgray", lty = 2, lwd = 2)
lines(density(Nets.real$soaking.time.hours), col = "red", lty = 2, lwd = 2)
lines(density(Nets.pred$soaking.time.hours[!is.skew]), col = "darkred", lty = 2, lwd = 2)
legend(x = "topright", legend= c("predicted", "real", "pred.notskew"), lty = 2,
       col = c("darkgray", "red", "darkred"), lwd = 2)
```

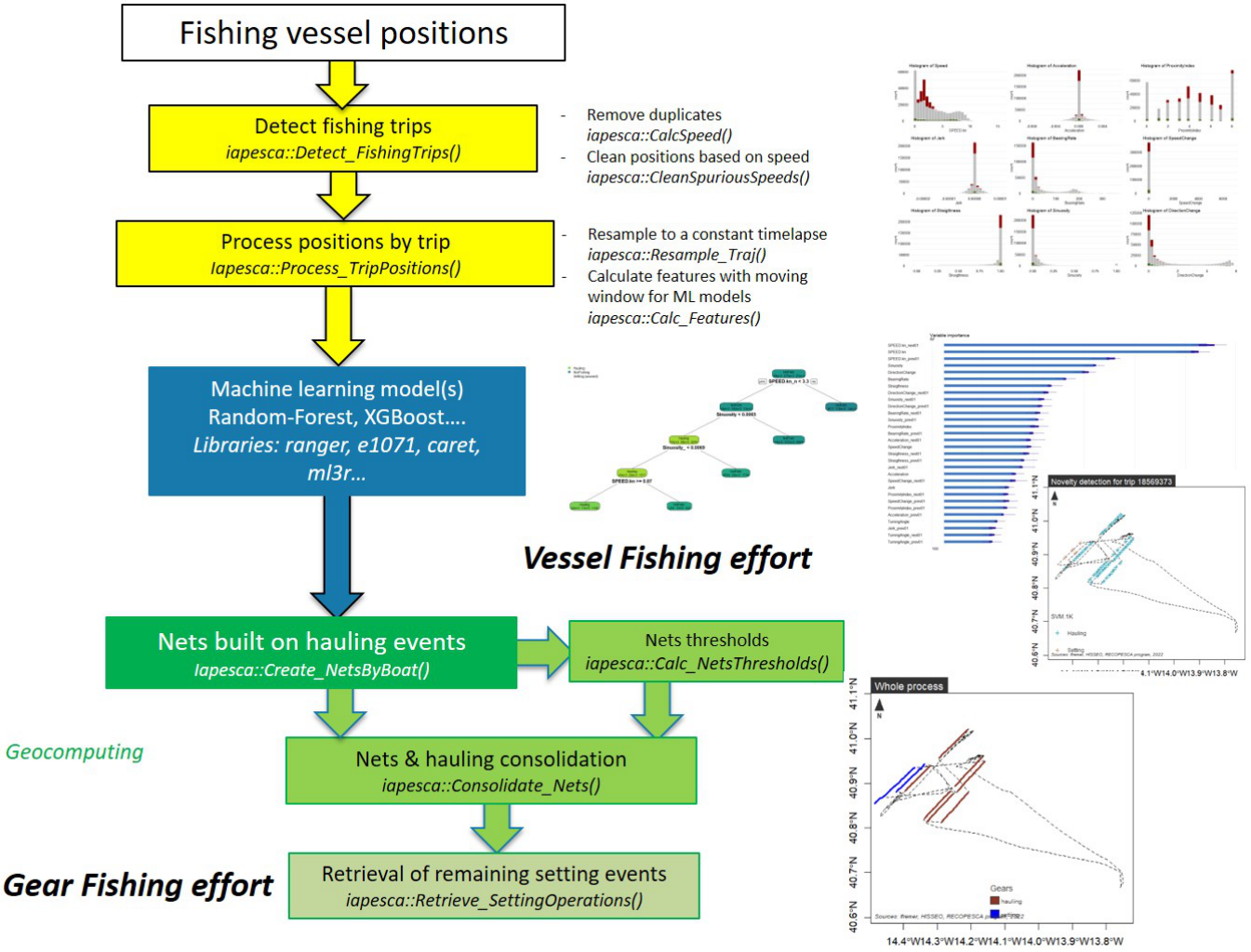


- A basic control is provided in *Retrieve_SettingOperations* with the argument tol.soaktime defined by default at 240 hours. The function searchs for possible setting during the last 3 fishing trips. These parameters could be made adjustable to include more control when detecting setting events.

4 - Case study : assesment of vessel and passive gears fishing efforts using a machine-learning model and geocomputation

4.1 Process to qualify the positions using models and geocomputation

- The figure below presents the global process and associated functions/packages when using machine-learning models for qualifying fishing vessels positions.
- The geocomputation process have been developped specifically for retrieving fishing efforts for passive gears. The method and functions are similar to the one used in the case study presented before when qualifying positions using sensors.



4.2 Loading and viewing the machine-learning models

- Load the machine-learning models stored in "ModelsNets_Recopesca0900.rds" saved in your data path defined by sprintf("%s/data",root)
- The models were calibrated on a dataset resampled to 15 minutes qualified using sensors or video :
  - 37 french fishing vessels operating nets for 2852 fishing trips
  - 405 884 positions with 9125 hauling operations identified.

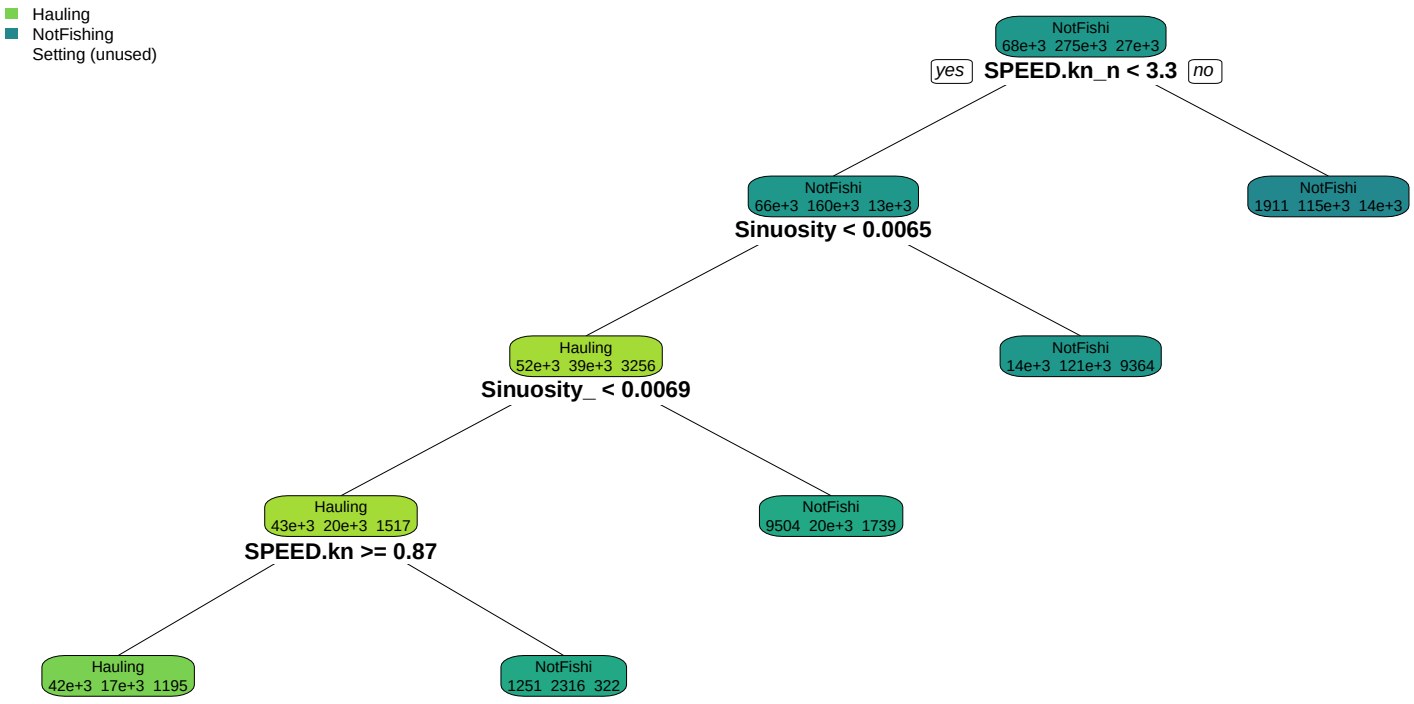
```
mod.nets.900 <- readRDS(
        file = sprintf("%s/ModelsNets_Recopesca0900.rds", fld.data)
        )
mod.nets.900 %>% names
```

```
[1] "optim.rf"  "mod.rf"    "mod.rpart" "probs.rf"
```

- This is a list with four objects :
  - optim.rf, hyperparameters for the random-forest model calculated using *tune_RF* function
  - mod.rf, a random-forest model, object of class "ranger"
  - mod.rpart, a CART model, single tree object of class "rpart"
  - probs.rf, a vector, thresholds of probabilities used to qualify the positions using the random-forest model. These thresholds have been defined to increase the number of fishing operations detected, false positives being then cleaned up using the geocomputation consolidation process.

Decision rules of the CART model

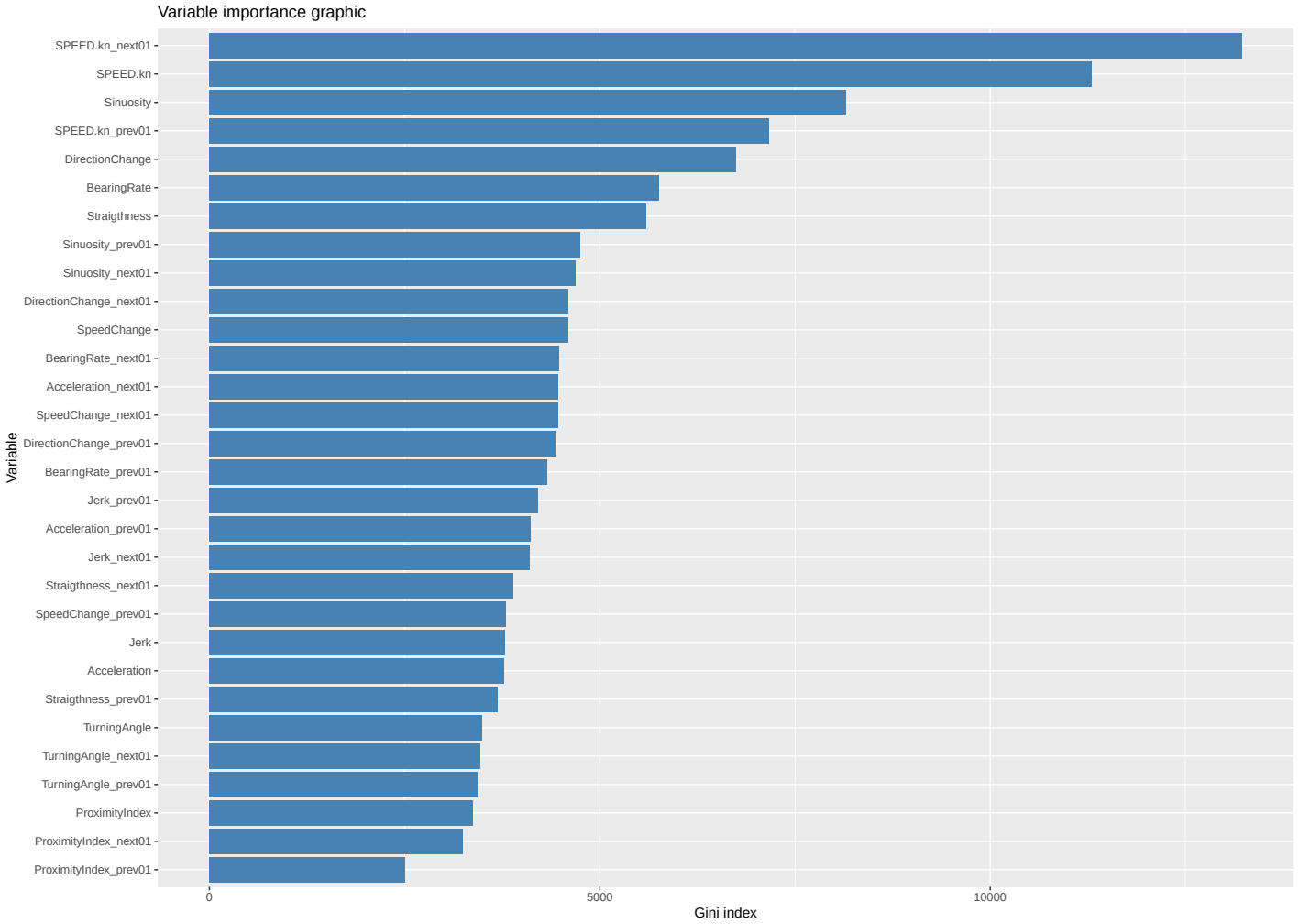
```
plot_tree(mod.nets.900$mod.rpart)
```



- Decision rules for setting events can't be defined with this single tree.
- Hauling events are defined when :
  - the speed for next position is below 3.3 knots while it remains superior to 0.87 knots for the actual position
  - the sinuosity is below 0.0065 and below 0.0069 for the next position.

Variable importance of the random-forest model

```
rf.ImpVar <- mod.nets.900$mod.rf$variable.importance %>% sort(decreasing = TRUE)
df.ImpVar <- data.frame(Variable = names(rf.ImpVar),
                        VarImp = rf.ImpVar)

ggplot(data = df.ImpVar, aes(x = reorder(Variable, VarImp), y = VarImp))+
  geom_bar(stat = "identity", fill = "steelblue")+
  coord_flip()+
  labs(title =  "Variable importance graphic", y = "Gini index", x = "Variable")
```



- Speed related features remain the most important ones for taking the decisions but also features describing sinuosity or direction changes.
- **To avoid over-fitting and for the models to remain robust when applied to boats operating in different fishing areas, covariates like direction, bearing, longitude and latitudes were not used directly even if they are used in the calculation of operational covariates.**

4.3 Processing the positions with features calculation

Preparation of the dataset

- Reload the *positions* dataset provided in iapesca package

```
data(positions)
```

- **This dataset wasn't used for calibrating the different models provided.**
- The data.frame is converted to a sf multipoints object using *df2sfp* function

```
pos.sf <- df2sfp(positions, coords = c("LONGITUDE", "LATITUDE"))
```

- Fishing trips must have been defined for processing the positions. If not, use the *Detect_FishingTrips* function, different methods being provided :
  - "Harbours", uses a spatial object either points or polygons of class sf to describe the harbours
  - "CoastLine", uses a spatial line objet of class sf describing the coastline. If set together to NULL with harbours and DetectCoastLine is TRUE, a coastline will be downloaded using geodata package with nested function *PathStudyBoundaries*.
  - a speed threshold may be additionnaly defined using argument SpeedLimit.
  - the default buffer (buf.size.start) is set to 2000 meters up to 2 nautic miles until an intersection is found with a minimum stop threshold in harbour or along the coast defined by the "min.stop" argument (DEFAULT = 3600 seconds).
- A summary of fishing trips is created using *Traj_Desc* function

```
trajDesc <- Traj_Desc(pos.sf)
FishingTrips <- trajDesc$FISHING_TRIP_FK
```

Processing the positions for the fishing trip starting on January 13th

- The *Process_TripPositions* is an umbrella function wrapping different operations on positions :
  - the duplicates are eventually removed when using the function *CalcSpeed* and the positions are eventually cleaned based on speed values using the *CleanSpuriousSpeeds* function.
  - fishing trips may eventually be reset if an harbour spatial object is provided (*Detect_FishingTrips*)
  - the positions may be resampled using the function *Resample_Traj* when filling the "resampling" argument in seconds
  - features may be calculated using the function *Calc_Features* when setting "CalcFeatures" argument to TRUE. A movingWindow may be defined using the argument "movingWindow" (set to 1 by default)
- Example for the trip starting on January 13th 2022 : 18569373
- The trip is selected and the function is applied to the path using following options :
  - MaxSpeed = 25, argument for *CleanSpuriousSpeeds* function, speed threshold in nautic miles for a position to be regarded as spurious.
  - resampling = 900, argument for *Resample_Traj* function, timelaps for resampling in seconds.
  - keep.var = "FishingOperation", argument for *Resample_Traj* function to keep the column in the resampled path.
  - create.paths = FALSE, option to return the positions as a linear path using *Pos2Path* function desactivated.
  - columns.ref = c("VESSEL_FK", "FISHING_TRIP_FK"), character string describing the columns with vessel and fishing trips identifiers

- CalcFeatures = TRUE, calculates additional features for machine-learning models using *CalcFeatures* function
- movingWindow = 1, number of neighbours to consider when calculating the moving window using *CalcFeatures* function

```
trip <- FishingTrips[5]

pos.trip <- Process_TripPositions(trip.path = pos.sf[ pos.sf$FISHING_TRIP_FK %in% trip, ],
                                  MaxSpeed = 25,
                                  resampling = 900,
                                  keep.var = "FishingOperation",
                                  create.paths = FALSE,
                                  columns.ref = c("VESSEL_FK", "FISHING_TRIP_FK"),
                                  CalcFeatures = TRUE,
                                  movingWindow = 1)
pos.trip %>% names
```

```
[1] "trip.path" "traj"
```

```
pos.trip$trip.path %>% colnames
```

```
 [1] "DATE_TIME"              "VESSEL_FK"              "FISHING_TRIP_FK"        "FishingOperation"
 [8] "HEADING.deg"            "Acceleration"           "ProximityIndex"         "Jerk"
[15] "Straigthness"           "Sinuosity"              "TurningAngle"           "DirectionChange"
[22] "Jerk_prev01"            "Bearing_prev01"         "BearingRate_prev01"     "SpeedChange_prev01"
[29] "DirectionChange_prev01" "SPEED.kn_next01"        "Acceleration_next01"    "ProximityIndex_next0
[36] "SpeedChange_next01"     "Straigthness_next01"    "Sinuosity_next01"       "TurningAngle_next01"
```

- A comparison of "speed" computed when using the *Clustering_BivariateBinary* and "SPEED.kn" computed by *CalcSpeed* function.

```
pos.sf$speed[ pos.sf$FISHING_TRIP_FK %in% trip] %>% head
```

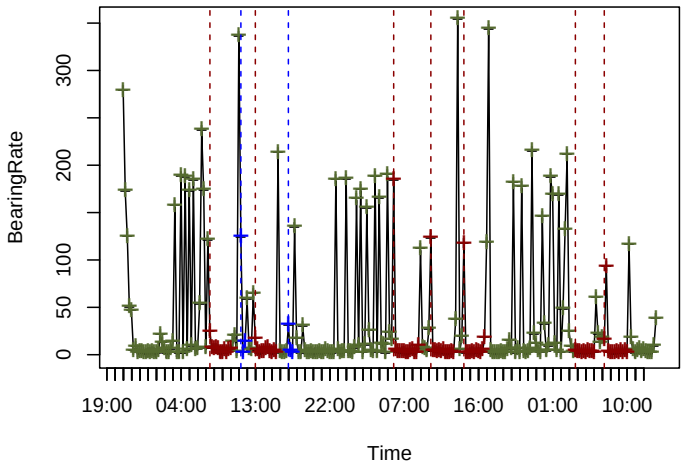
```
[1] 0.0009517128 0.0067395330 0.0029241483 2.9887933520 5.6381009273 5.1385326225
```

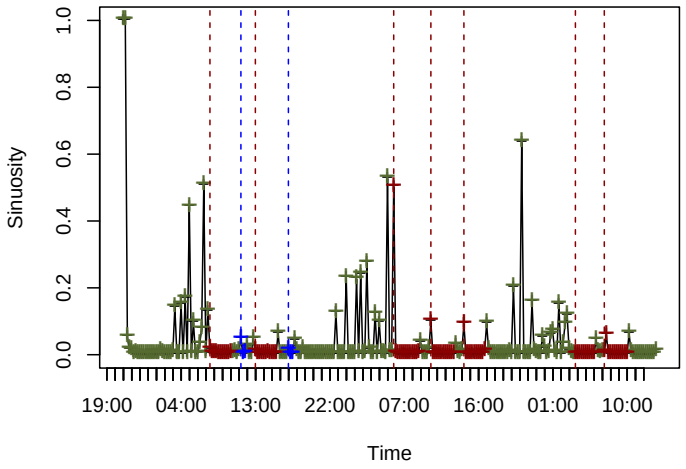
```
pos.trip$trip.path$SPEED.kn %>% head
```

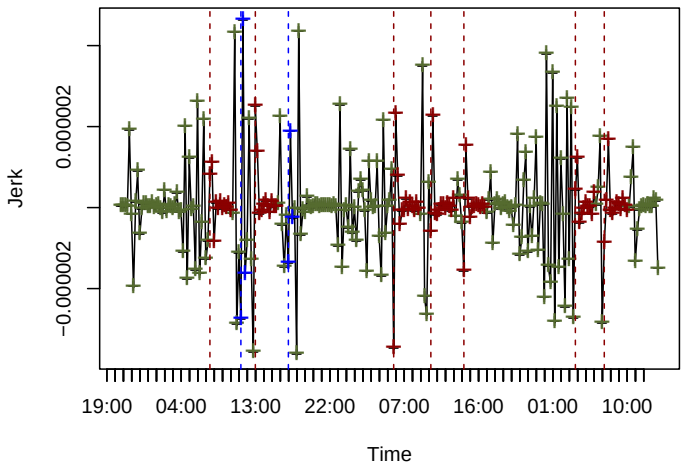
```
[1] 0.0000000000 0.0009415852 0.0066731843 0.0028950227 2.9598814163 5.5826631157
```

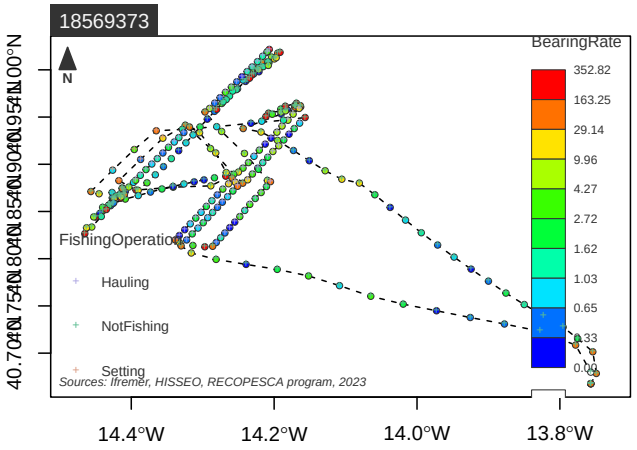
- A shift appears between these computations, it is explained by the convention used when computing the distances :
  - it is defined between the position and the previous one when using *CalcDist*, speed in position 1 equals 0
  - it is defined between the position and the next one when applying *Clustering_BivariateBinary* which uses EMbC functions, speed in last position equals 0.
- Features and their relations towards fishing operations can be viewed using *plot_Feature* or *plot_Trip* functions in iapesca.

```
plot_Trip(pos.trip$trip.path,
          quanti = "BearingRate",
          cex.FoP = 0.6,
          col.features = c("BearingRate" , "Sinuosity", "Jerk"),
          col.FishingOp = "FishingOperation")
```









- The *Process_TripPositions* function is then applied to the whole dataset by Fishing trip

```
posForMl <- do.call(rbind,
                lapply(trajDesc$FISHING_TRIP_FK,
                       function(trip){
                         Process_TripPositions(
                             trip.path = pos.sf[ pos.sf$FISHING_TRIP_FK %in% trip, ],
                             MaxSpeed = 25,
                             resampling = 900,
                             keep.var = "FishingOperation",
                             create.paths = FALSE,
                             columns.ref = c("VESSEL_FK", "FISHING_TRIP_FK"),
                             CalcFeatures = TRUE,
```

```
                                movingWindow = 1)$trip.path
                              }))
```

Distribution of the values of the covariates used in the models and their relations to fishing operations

- Define the covariates and their identifiers in posForMl columns.
- Define an object identifying the lines whithout missing values : "nnai"

```
covar <- c( "SPEED.kn", "Acceleration", "ProximityIndex",
            "Jerk",  "BearingRate", "SpeedChange",
            "Straigthness", "Sinuosity", "TurningAngle",
            "DirectionChange")
col.index <- sapply(1:length(covar),
                    function(k){ grep(colnames(posForMl), pattern = covar[k]) })
covar.cols <- colnames(posForMl)[col.index]
covar.cols %>% print
```

```
 [1] "SPEED.kn"               "SPEED.kn_prev01"        "SPEED.kn_next01"        "Acceleration"
 [8] "ProximityIndex_prev01"  "ProximityIndex_next01"  "Jerk"                   "Jerk_prev01"
[15] "BearingRate_next01"     "SpeedChange"            "SpeedChange_prev01"     "SpeedChange_next01"
[22] "Sinuosity"              "Sinuosity_prev01"       "Sinuosity_next01"       "TurningAngle"
[29] "DirectionChange_prev01" "DirectionChange_next01"
```

```
nnai <- apply(posForMl[, covar.cols], 1, function(x){ !anyNA(x) })

nnai %>% summary
```

```
   Mode   FALSE    TRUE
logical      32    1902
```

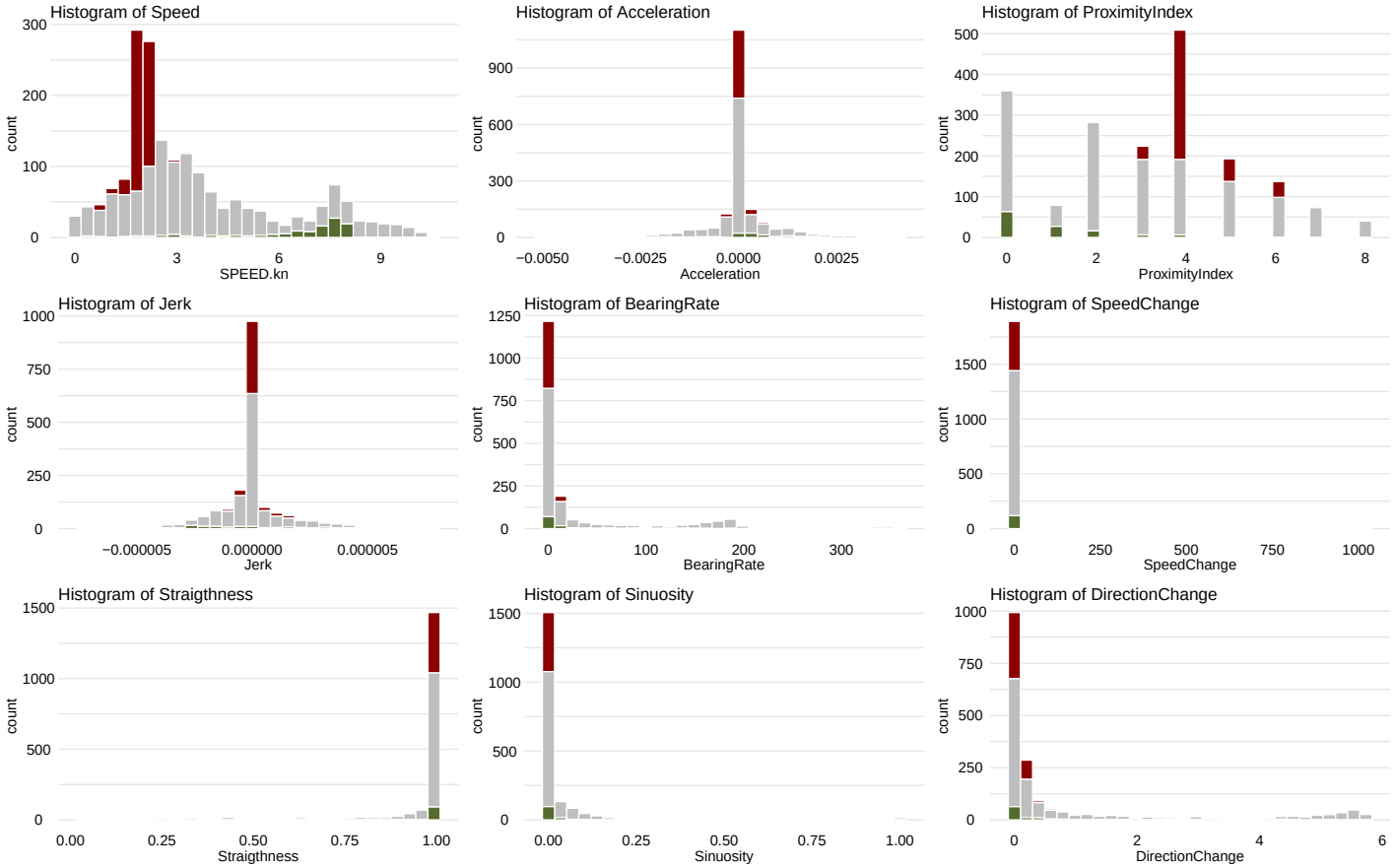
- Creates ggplot graphics

```
plot.spd <- ggplot(posForMl[nnai, ], aes(SPEED.kn, fill = FishingOperation)) +
  geom_histogram(color = "white") +
  labs(title = "Histogram of Speed") +
  DALEX::theme_ema() +
  scale_fill_manual("", values = c("darkred",  "grey",  "darkolivegreen"))

plot.acc <- ggplot(posForMl[nnai, ], aes(Acceleration, fill = FishingOperation)) +
  geom_histogram(color = "white") +
  ggtitle("Histogram of Acceleration") +
  DALEX::theme_ema() +
  scale_fill_manual("", values = c("darkred",  "grey",  "darkolivegreen"))

plot.pri <- ggplot(posForMl[nnai, ], aes(ProximityIndex, fill = FishingOperation)) +
  geom_histogram(color = "white") +
  ggtitle("Histogram of ProximityIndex") +
  DALEX::theme_ema() +
  scale_fill_manual("", values = c("darkred",  "grey",  "darkolivegreen"))
```

```
plot.jrk <- ggplot(posForMl[nnai, ], aes(Jerk, fill = FishingOperation)) +
  geom_histogram(color = "white") +
  ggtitle("Histogram of Jerk") +
  DALEX::theme_ema() +
  scale_fill_manual("", values = c("darkred",  "grey",  "darkolivegreen"))

plot.bra <- ggplot(posForMl[nnai, ], aes(BearingRate, fill = FishingOperation)) +
  geom_histogram(color = "white") +
  ggtitle("Histogram of BearingRate") +
  DALEX::theme_ema() +
  scale_fill_manual("", values = c("darkred",  "grey",  "darkolivegreen"))

plot.spc <- ggplot(posForMl[nnai, ], aes(SpeedChange, fill = FishingOperation)) +
  geom_histogram(color = "white") +
  ggtitle("Histogram of SpeedChange") +
  DALEX::theme_ema() +
  scale_fill_manual("", values = c("darkred",  "grey",  "darkolivegreen"))

plot.str <- ggplot(posForMl[nnai, ], aes(Straigthness, fill = FishingOperation)) +
  geom_histogram(color = "white") +
  ggtitle("Histogram of Straigthness") +
  DALEX::theme_ema() +
  scale_fill_manual("", values = c("darkred",  "grey",  "darkolivegreen"))

plot.sin <- ggplot(posForMl[nnai, ], aes(Sinuosity, fill = FishingOperation)) +
  geom_histogram(color = "white") +
  ggtitle("Histogram of Sinuosity") +
  DALEX::theme_ema() +
  scale_fill_manual("", values = c("darkred",  "grey",  "darkolivegreen"))

plot.dir <- ggplot(posForMl[nnai, ], aes(DirectionChange, fill = FishingOperation)) +
  geom_histogram(color = "white") +
  ggtitle("Histogram of DirectionChange") +
  DALEX::theme_ema() +
  scale_fill_manual("", values = c("darkred",  "grey",  "darkolivegreen"))
```

- Distribution of the covariates and the position of setting (green) and hauling (red) values.

```
multi_ggplots(plot.spd, plot.acc, plot.pri,
              plot.jrk, plot.bra, plot.spc,
              plot.str, plot.sin, plot.dir,
              cols = 3)
```



4.4 Prediction of fishing events using the random-forest model

- The random-forest model in "mod.nets.900" is used to set predictions of fishing events.
- For setting the predictions, a data.frame object without missing values for covariates is created using "sfp2df(posForMl[nnai,])" command.

```
pred.rf <- predict(mod.nets.900$mod.rf, data = sfp2df(posForMl[nnai, ]))
pred.rf$predictions %>% head
```

```
     Hauling NotFishing Setting
[1,]   0.006      0.986   0.008
[2,]   0.002      0.974   0.024
[3,]   0.000      0.979   0.021
[4,]   0.000      0.960   0.040
[5,]   0.000      0.986   0.014
[6,]   0.000      1.000   0.000
```

- A new variable named "Fop.RF" is created to store the predictions results

```
posForMl$Fop.RF <- rep(NA, nrow(posForMl))
posForMl$Fop.RF[nnai] <- colnames(pred.rf$predictions)[apply(pred.rf$predictions, 1, which.max)]

tab.rf <- with(posForMl, table(FishingOperation, Fop.RF))
acc.rf <- Calc_Acc(tab.rf)
```

- Random-forest provides an accuracy of 83.33 % with following results by fishing events :

```
tab.rf
```

```
                Fop.RF
FishingOperation Hauling NotFishing Setting
      Hauling        418         33       0
      NotFishing     167       1166       0
      Setting          0        117       1
```

- Almost all the setting events remain undetected.
- User-defined probabilities have been set for optimizing the probability of detecting fishing events. They are stored in "pred.rf$predictions" object containing the probability threshold for a position to belong to one class.
- These probability thresholds may be used to improve the number of setting events detected :

```
mod.nets.900$probs.rf
```

```
   Hauling NotFishing    Setting
        31         NA         21
```

- Using these probabilities thresholds, setting events predictions can be reset.

```
sel.newSettings <- pred.rf$predictions[, "Setting"] > mod.nets.900$probs.rf["Setting"]/100
sel.newSettings <- sel.newSettings & !posForMl$Fop.RF[nnai] %in% "Hauling"
posForMl$Fop.RF[nnai] [sel.newSettings] <- "Setting"

tab.rf <- with(posForMl, table(FishingOperation, Fop.RF))
acc.rf <- Calc_Acc(tab.rf)
```

- The overall accuracy increases to 85.54 % with following results by fishing events :

```
tab.rf
```

```
                Fop.RF
FishingOperation Hauling NotFishing Setting
      Hauling        418         33       0
      NotFishing     167       1134      32
      Setting          0         43      75
```

- Probability thresholds having been voluntarily reduced to optimize the detection of fishing events, the number of false positives concerning these events is necessarily higher.
- A great number of setting events remain undetected (36%), most of them will be retrieved by geocomputation process implemented in *Retrieve_SettingOperations* function

4.5 Building and consolidation of fishing gears using geocomputation

- The process and functions involved as far as their arguments have been previously described in part 3.6 of this tutorial.
- In this case, the method "Use.BehaviourChanges" is chosen : nets are built from the combination of hauling sequences and "Clust.Pass" variable, the non-supervized behavioural diagnosis implemented in *Detect_BehaviourChanges*.
- For the consolidation, the method "Auto.ThreshHolds.Detection" has been selected with "qt" and "iter" arguments left unprovided. They will be set using *Set_NetThresholds* which tests iteratively designs of values and choose their combination that maximizes the decay of thresholds. This process involving random simulations, different results can be expected when the process is rerun if the argument "seed" is not defined.
- From the thresholds defined using these arguments, nets can be removed, or corrected (paste or split).

- The "parallelize" option is set to TRUE, packages parallel, foreach and doParallel must have been installed. If left unprovided "nCores" argument is set to half the available cores on the computer.

```
Nets.rfMod <- Create_NetsByBoat(traj = posForMl,
                                Col.Fop = "Fop.RF",
                                Fop.category = "Hauling",
                                Use.BehaviourChanges = TRUE,
                                Auto.ThreshHolds.Detection = TRUE,
                                seed = seed,
                                parallelize = TRUE,
                                verbose = TRUE
                               )
```

- The object "Nets.pred" contains "Nets" a list of spatial linestring objects of class sf, each of them being the output of Create_Nets function applied to the fishing trips.

```
Nets.rfMod$Nets %>% names
```

```
[1] "18483452" "18483453" "18529291" "18569373" "18569374" "18569376" "18569377"
```

```
Nets.rfMod$Nets[[1]] %>%
  sf::st_set_geometry(NULL) %>%
    print_table
```

Net	length	heading	speed	hauling.start	hauling.end	its	FISHING_TRIP_FK
18483452_01	9 987.505 [m]	46.97879	2.043995	2021-12-26 06 :38 :23	2021-12-26 09 :15 :53	0	18483452
18483452_02	5 240.702 [m]	226.10015	2.030032	2021-12-26 11 :53 :23	2021-12-26 13 :15 :53	0	18483452
18483452_03	9 633.829 [m]	227.47720	1.862079	2021-12-27 01 :45 :53	2021-12-27 04 :30 :53	0	18483452
18483452_04	9 440.489 [m]	47.87464	1.815323	2021-12-27 06 :30 :53	2021-12-27 09 :15 :53	0	18483452

- 43 nets are detected for 7 fishing trips with hauling events, representing a total of 323596 meters
- The second object "Nets.Thresh" is a list of thresholds, output of Calc_NetsThresholds function

- Nets consolidation is built from these thresholds.

```
Nets.rfMod$Nets.Thresh
```

```
$LimVal.Speed
[1] 1.615294 2.484906

$LimVal.Length
[1] 4915.587 9997.726

$LimVal.Heading
          [,1]      [,2]
[1,]  25.96133  50.22237
[2,] 192.06030 231.62310

$LimVal.SoakTime
NULL
```

- Real nets were built previously in part 3.6 from hauling events identified in "FishingOperation" column.
- 39 nets were detected for these 7 fishing trips with hauling events, representing a total of 394811 meters
- **It appears that the thresholds automatically detected for the individual net lengths are underestimated, leading to an underestimation of the total length deployed when applying the consolidation process.**
- **When using the automated consolidation process, it is advised to use more fishing trips to define more robust statistics**
- If they are known, user-defined thresholds can be set inside the function using the arguments : "length.lim", "av.speed.lim", "heading.lim".
- In the case of this boat, we know that the nets have a length of 11 kms each. When using a temporal resolution of 15 minutes, we may expect an underestimation of the length by the methods, we then define thresholds for lengths ranging from 8,000 m to 13,000 m

```
Nets.rfMod <- Create_NetsByBoat(traj = posForMl,
                                Col.Fop = "Fop.RF",
                                Fop.category = "Hauling",
                                Use.BehaviourChanges = FALSE,
                                Auto.ThreshHolds.Detection = TRUE,
                                length.lim = c(8000, 13000),
                                seed = seed,
                                parallelize = TRUE,
                                verbose = TRUE
                               )
```

- 44 nets are detected for 8 fishing trips with hauling events, representing a total of 448607 meters.

4.6 Retrieval of remaining setting events using geocomputation

- As setting events are less characterizable than hauling, in particular because they are carried out at higher speeds involving fewer positions, the geocomputation can help in identifying these sequences.

- The umbrella function "Retrieve_SettingOperations" has been built for that purpose. This function returns two objects :

- nets, a spatial linestring object of class sf with additional columns to describe setting events : setting trip, start and end and soaking time.
- traj, the initial positions as sf object with new columns identifying hauling and setting operations with their gear identifier.

- Buffers and scoring are used to retrieve the setting events. Based on a fully qualified dataset, they have been set to 100 m for hauling events and 330 m for setting events up to argument "buffer.max" (DEFAULT 550 m) if no candidates for setting are identified.

- If the argument "remove.orphans" is set to TRUE, nets for which no related setting events have been found are removed.

```
Setting.PredRF <- Retrieve_SettingOperations(traj = posForMl,
                                             ls.nets = Nets.rfMod$Nets,
                                             Col.Fop = "Fop.RF",
                                             Setting.category = "Setting",
                                             Hauling.category = "Hauling",
                                             remove.orphans = FALSE)
```

```
names(Setting.PredRF)
```

```
[1] "nets" "traj"
```

```
Setting.PredRF %>% str
```

```
List of 2
 $ nets:Classes 'sf' and 'data.frame':  44 obs. of  13 variables:
  ..$ Net               : chr [1:44] "18483451_01" "18483452_01" "18483452_02" "18483452_03" ...
  ..$ length            : Units: [m] num [1:44] 12925 9988 10148 8753 9634 ...
  ..$ heading           : num [1:44] 236 47 227 225 227 ...
  ..$ speed             : num [1:44] 2.49 2.04 2.07 2.66 1.86 ...
  ..$ hauling.start     : chr [1:44] "2021-12-23 16:10:23" "2021-12-26 06:38:23" "2021-12-26 11:53:23
  ..$ hauling.end       : chr [1:44] "2021-12-23 19:10:23" "2021-12-26 09:15:53" "2021-12-26 14:30:53
  ..$ its               : int [1:44] 0 0 0 0 0 0 0 0 1 ...
  ..$ FT_Hauling        : chr [1:44] "18483451" "18483452" "18483452" "18483452" ...
  ..$ setting.start     : chr [1:44] "2021-12-23 11:10:23" "2021-12-23 15:10:23" "2021-12-23 21:10:23
  ..$ setting.end       : chr [1:44] "2021-12-23 11:55:23" "2021-12-23 15:40:23" "2021-12-23 21:40:23
  ..$ FT_Setting        : chr [1:44] "18483451" "18483451" "18483451" "18483451" ...
  ..$ soaking.time.hours: num [1:44] 9 67.1 66.3 70.5 79.6 ...
  ..$ geometry          :sfc_LINESTRING of length 44; first list element:  'XY' num [1:25, 1:2] -14.2
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:25] "1" "2" "3" "4" ...
  .. .. ..$ : chr [1:2] "X" "Y"
  ..- attr(*, "sf_column")= chr "geometry"
  ..- attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA NA NA NA ...
  .. ..- attr(*, "names")= chr [1:12] "Net" "length" "heading" "speed" ...
 $ traj:Classes 'sf' and 'data.frame':  1934 obs. of  43 variables:
  ..$ DATE_TIME             : POSIXct[1:1934], format: "2021-12-23 06:55:23" "2021-12-23 07:10:23" "2
  ..$ VESSEL_FK             : chr [1:1934] "NAVIRE_0075" "NAVIRE_0075" "NAVIRE_0075" "NAVIRE_0075" ..
  ..$ FISHING_TRIP_FK       : chr [1:1934] "18483451" "18483451" "18483451" "18483451" ...
  ..$ DISTANCE.nm           : num [1:1934] 0 0.00248 0.0747 1.34136 1.96392 ...
```

```
  ..$ DIFFTIME.secs         : num [1:1934] 0 900 900 900 900 900 900 900 900 900 ...
  ..$ SPEED.kn              : num [1:1934] 0 0.00994 0.29878 5.36542 7.85568 ...
  ..$ HEADING.deg           : num [1:1934] 235 337 56 115 135 ...
  ..$ Acceleration          : num [1:1934] 0 0.00000568 0.00016511 0.00289611 0.00142345 ...
  ..$ ProximityIndex        : num [1:1934] 2 2 2 0 0 0 0 0 0 0 ...
  ..$ Jerk                  : num [1:1934] 0 0.00000000631 0.00000017714 0.00000303445 -0.0000016363
  ..$ Bearing               : num [1:1934] -134.2 -18.4 46.8 120.7 142.7 ...
  ..$ BearingRate           : num [1:1934] NA 115.8 65.2 74 21.9 ...
  ..$ SpeedChange           : num [1:1934] NA 29.0738 16.9575 0.4641 0.0835 ...
  ..$ Straigthness          : num [1:1934] NA 0.962 0.958 0.877 0.984 ...
  ..$ Sinuosity             : num [1:1934] NA 0.15198 0.03163 0.02004 0.00608 ...
  ..$ TurningAngle          : num [1:1934] NA 101.5 79.4 58.5 20.3 ...
  ..$ DirectionChange       : num [1:1934] NA 3.231 2.528 1.863 0.647 ...
  ..$ SPEED.kn_prev01       : num [1:1934] NA 0 0.00994 0.29878 5.36542 ...
  ..$ Acceleration_prev01   : num [1:1934] NA 0 0.00000568 0.00016511 0.00289611 ...
  ..$ ProximityIndex_prev01 : num [1:1934] NA 2 2 2 0 0 0 0 0 0 ...
  ..$ Jerk_prev01           : num [1:1934] NA 0 0.00000000631 0.00000017714 0.00000303445 ...
  ..$ Bearing_prev01        : num [1:1934] NA -134.2 -18.4 46.8 120.7 ...
  ..$ BearingRate_prev01    : num [1:1934] NA NA 115.8 65.2 74 ...
  ..$ SpeedChange_prev01    : num [1:1934] NA NA 29.074 16.958 0.464 ...
  ..$ Straigthness_prev01   : num [1:1934] NA NA 0.962 0.958 0.877 ...
  ..$ Sinuosity_prev01      : num [1:1934] NA NA 0.152 0.0316 0.02 ...
  ..$ TurningAngle_prev01   : num [1:1934] NA NA 101.5 79.4 58.5 ...
  ..$ DirectionChange_prev01: num [1:1934] NA NA 3.23 2.53 1.86 ...
  ..$ SPEED.kn_next01       : num [1:1934] 0.00994 0.29878 5.36542 7.85568 7.19984 ...
  ..$ Acceleration_next01   : num [1:1934] 0.00000568 0.00016511 0.00289611 0.00142345 -0.00037489 ..
  ..$ ProximityIndex_next01 : num [1:1934] 2 2 0 0 0 0 0 0 0 0 ...
  ..$ Jerk_next01           : num [1:1934] 0.00000000631 0.00000017714 0.00000303445 -0.0000016363 -0
  ..$ Bearing_next01        : num [1:1934] -18.4 46.8 120.7 142.7 149.2 ...
  ..$ BearingRate_next01    : num [1:1934] 115.81 65.17 73.97 21.92 6.53 ...
  ..$ SpeedChange_next01    : num [1:1934] 29.0738 16.9575 0.4641 0.0835 0.0143 ...
  ..$ Straigthness_next01   : num [1:1934] 0.962 0.958 0.877 0.984 0.998 ...
  ..$ Sinuosity_next01      : num [1:1934] 0.15198 0.03163 0.02004 0.00608 0.00212 ...
  ..$ TurningAngle_next01   : num [1:1934] 101.51 79.43 58.53 20.34 6.97 ...
  ..$ DirectionChange_next01: num [1:1934] 3.231 2.528 1.863 0.647 0.222 ...
  ..$ Fop.RF                : chr [1:1934] NA NA "NotFishing" "NotFishing" ...
  ..$ Hauling_GearId        : chr [1:1934] NA NA NA NA ...
  ..$ Setting_GearId        : chr [1:1934] NA NA NA NA ...
  ..$ geometry              :sfc_POINT of length 1934; first list element:  'XY' num [1:2] -13.8 40.7
  ..- attr(*, "sf_column")= chr "geometry"
  ..- attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA NA NA NA ...
  .. ..- attr(*, "names")= chr [1:42] "DATE_TIME" "VESSEL_FK" "FISHING_TRIP_FK" "DISTANCE.nm" ...
```

```
Nets.rfMod <- Setting.PredRF$nets
```

- Predicted values consolidated by the geocomputation process can now be updated in a new column named "Fop.RfGeo"

```
posForMl$Fop.RfGeo <- rep(NA, nrow(posForMl))
posForMl$Fop.RfGeo[ !is.na( Setting.PredRF$traj$Hauling_GearId)] <- "Hauling"
posForMl$Fop.RfGeo[ !is.na( Setting.PredRF$traj$Setting_GearId)] <- "Setting"
posForMl$Fop.RfGeo[ is.na(posForMl$Fop.RfGeo)] <- "NotFishing"
```

```
with(posForMl, table(FishingOperation, Fop.RF))
```

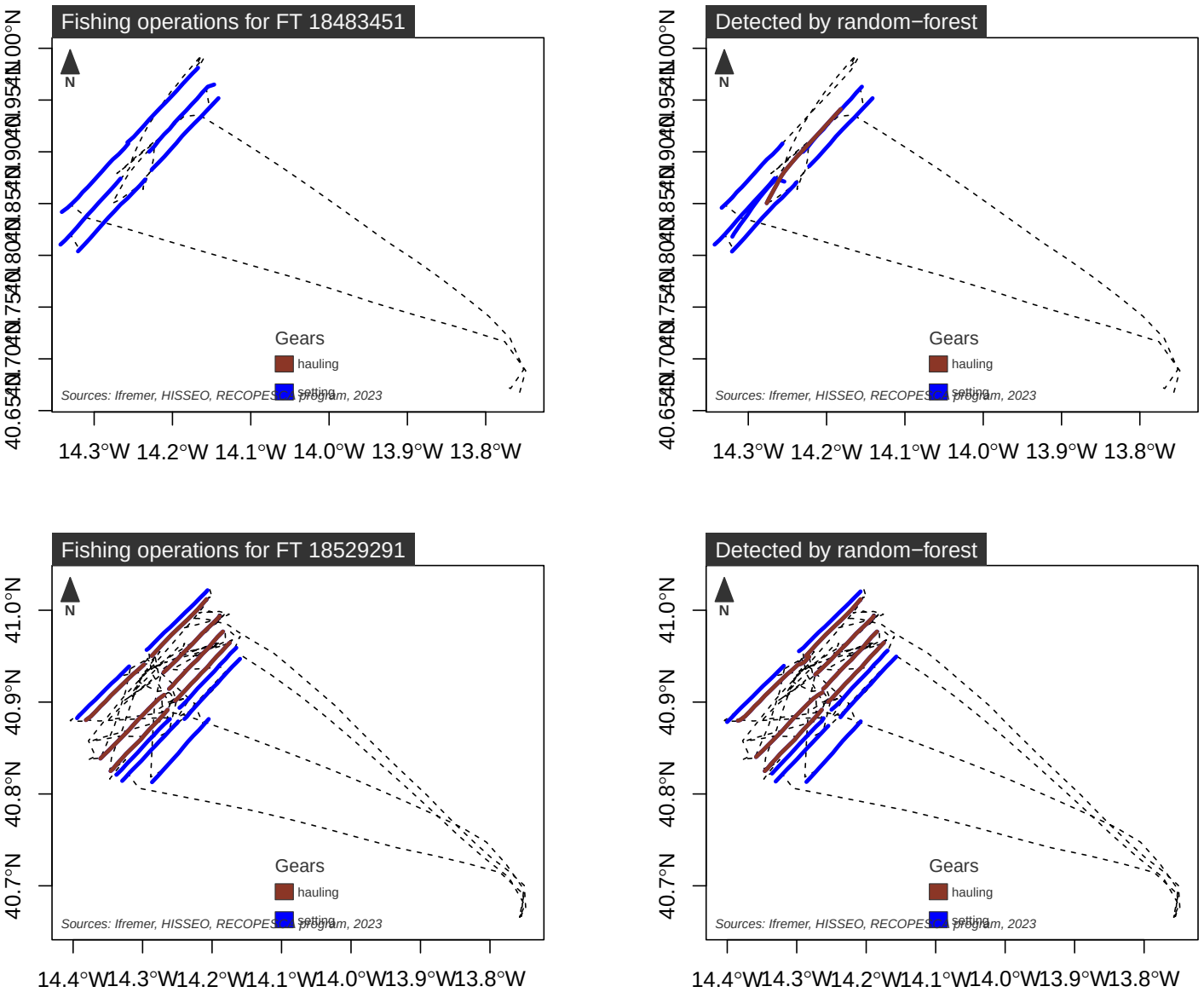
```
                Fop.RF
FishingOperation Hauling NotFishing Setting
      Hauling        418         33       0
      NotFishing     167       1134      32
      Setting          0         43      75
```

```
with(posForMl, table(FishingOperation, Fop.RfGeo))
```

```
                Fop.RfGeo
FishingOperation Hauling NotFishing Setting
      Hauling        417         34       0
      NotFishing      55       1276      34
      Setting          0         14     104
```

- This geocomputation process increases the accuracy of qualification from 85.54 % to **92.92 %**.
- It reduces the rate of false positives by 55.28 % and increases by 38.67% the number of true setting operations detected.
- **Without user-defined thresholds for net lengths, overall accuracy also increased to 89.5 %, but the number of true hauling events was underestimated by 24.4 % instead of 7.54 % when using user-defined thresholds for net lengths.**

```
par(mfrow=c(1, 2))

trip <- FishingTrips[1]
pos.trip <- posForMl[ posForMl$FISHING_TRIP_FK %in% trip, ]

plot_Paths(pos.trip,
           FishingOp = "FishingOperation",
           quanti = NULL,
           main = sprintf("Fishing operations for FT %s", trip),
           gears.set = Nets.real[ Nets.real$FT_Setting %in% trip, ],
           gears.hauled = Nets.real[ Nets.real$FT_Hauling %in% trip, ],
           Create.BgMap = FALSE)

plot_Paths(pos.trip,
           FishingOp = "Fop.RfGeo",
           quanti = NULL,
           main = "Detected by random-forest",
           gears.set = Nets.rfMod[ Nets.rfMod$FT_Setting %in% trip, ],
           gears.hauled = Nets.rfMod[ Nets.rfMod$FT_Hauling %in% trip, ],
           Create.BgMap = FALSE)
```



4.7 Calculation of fishing effort metrics

4.7.1 Fishing vessel effort : from "days at sea" to "fishing hours"

- A function is created to retrieve following vessel fishing effort metrics :
  - Days at sea
  - Hours at sea
  - Fishing days
  - Fishing hours

```
Calc_VesselFishingEffort <- function(pos,
                                     Col.Fop,
                                     NotFop.category = "NotFishing"){

  trajDesc.Fop <- Traj_Desc(pos, add.group = Col.Fop)
  trajDesc.Fop$Duration <- difftime(trajDesc.Fop$End.FT,
                                    trajDesc.Fop$Start.FT,
```

```
                                  units = "hours")

fun_Date <- function(x){

    c(min = as.character(min(Char2Time(as.character(x)))),
      max = as.character(max(Char2Time(as.character(x))))
    )

}

trajDesc.FT <- as.data.frame(
                      aggregate(trajDesc.Fop[, c("Start.FT", "End.FT")],
                         by = list(VESSEL_FK = trajDesc.Fop$VESSEL_FK,
                                   FISHING_TRIP_FK=  trajDesc.Fop$FISHING_TRIP_FK),
                         FUN = fun_Date)
)

trajDesc.FT$Start.FT <- Char2Time(trajDesc.FT$Start.FT[,1])
trajDesc.FT$End.FT <- Char2Time(trajDesc.FT$End.FT[,2])
trajDesc.FT$DaysAtSea <- as.numeric(substr(trajDesc.FT$End.FT, 9, 10))
                          - as.numeric(substr(trajDesc.FT$Start.FT, 9,10)) + 1
trajDesc.FT$HoursAtSea <- difftime(trajDesc.FT$End.FT,
                                   trajDesc.FT$Start.FT,
                                   units = "hours")

trajDesc.Fop <- trajDesc.Fop[ !trajDesc.Fop[, Col.Fop] %in% NotFop.category, ]
FishingDays <- as.data.frame(
                      aggregate(trajDesc.Fop[, c("Start.FT", "End.FT")],
                         by = list(VESSEL_FK = trajDesc.Fop$VESSEL_FK,
                                   FISHING_TRIP_FK=  trajDesc.Fop$FISHING_TRIP_FK),
                         FUN = fun_Date)
)
FishingDays$Start.FT <- Char2Time(FishingDays$Start.FT[,1])
FishingDays$End.FT <- Char2Time(FishingDays$End.FT[,2])
FishingDays$FishingDays <- as.numeric(substr(FishingDays$End.FT, 9,10))
                           - as.numeric(substr(FishingDays$Start.FT, 9,10)) + 1

FishingHours <- as.data.frame(
                      aggregate(trajDesc.Fop[, c("Duration")],
                         by = list(VESSEL_FK = trajDesc.Fop$VESSEL_FK,
                                   FISHING_TRIP_FK=  trajDesc.Fop$FISHING_TRIP_FK),
                         FUN = sum)
)

colnames(FishingHours) [ncol(FishingHours)] <- "FishingHours"

FishingEffort <- merge(FishingDays,
                      FishingHours,
                      by = c("VESSEL_FK", "FISHING_TRIP_FK"))

VesselFE <- merge(trajDesc.FT,
                  FishingEffort[, !colnames(FishingEffort) %in% c("Start.FT", "End.FT")],
```

```
                    by = c("VESSEL_FK", "FISHING_TRIP_FK")
                    )
  VesselFE$HoursAtSea <- as.numeric(VesselFE$HoursAtSea)
  VesselFE$FishingHours <- as.numeric(VesselFE$FishingHours)

 return(VesselFE)

}
```

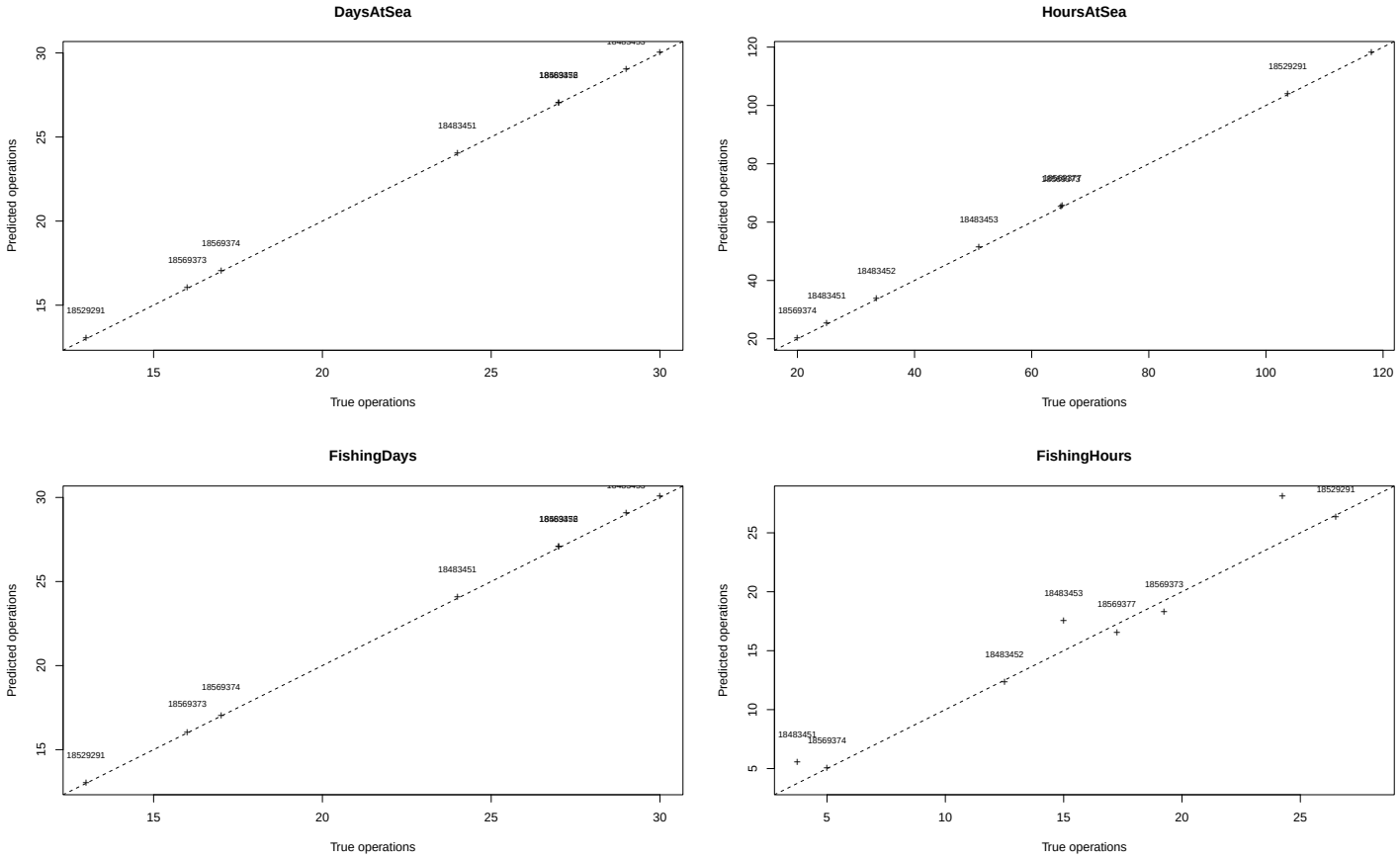
- Calculation of predicted and true vessel fishing effort metrics

```
vesselFE.true <- Calc_VesselFishingEffort(posForMl, "FishingOperation")
vesselFE.RfGeo <- Calc_VesselFishingEffort(posForMl, "Fop.RfGeo")

vesselFE.true %>%
  head %>%
  print_table
```

VESSEL_FK	FISHING_TRIP_FK	Start.FT	End.FT	DaysAtSea	HoursAtSea	FishingDays	FishingHours
NAVIRE_0075	18483451	2021-12-23 06 :55 :23	2021-12-24 07 :55 :23	24	25.00	24	3.75
NAVIRE_0075	18483452	2021-12-26 03 :00 :53	2021-12-27 12 :30 :53	27	33.50	27	12.50
NAVIRE_0075	18483453	2021-12-27 18 :46 :47	2021-12-29 21 :46 :47	29	51.00	29	15.00
NAVIRE_0075	18529291	2022-01-09 10 :43 :18	2022-01-13 18 :28 :18	13	103.75	13	26.50
NAVIRE_0075	18569373	2022-01-13 19 :43 :17	2022-01-16 12 :43 :17	16	65.00	16	19.25
NAVIRE_0075	18569374	2022-01-16 14 :19 :23	2022-01-17 10 :19 :23	17	20.00	17	5.00

- Comparison of computed fishing effort metrics by fishing trips

```
  par(mfrow = c(2,2))

  plot_FE(vesselFE.true, vesselFE.RfGeo, "DaysAtSea")
  plot_FE(vesselFE.true, vesselFE.RfGeo, "HoursAtSea")
  plot_FE(vesselFE.true, vesselFE.RfGeo, "FishingDays")
  plot_FE(vesselFE.true, vesselFE.RfGeo,"FishingHours")
```



- Logically, no changes are observed regarding days or hours at sea defined by the fishing trips annotation. For these eight fishing trips :
  - on average, 22.875 days at sea for a total of 183
  - on average 60.2 hours at sea for a total of 481.5
- No differences are also observed between the Fishing days when they are computed from true or predicted operations :
  - on average 22.9 fishing days for a total of 183
- Small differences appear regarding the computation of fishing hours :
  - on average 15.4 hours fishing for a total of 123.5 when the true operations are used
  - on average 16.2 hours fishing for a total of 129.2 when the predicted operations are used
- Using the predicted operations leads to a small bias when estimating the fishing hours by 4.7 %.

4.7.2 Total length of soaked/hauled nets

- These metrics can be easily extracted from “nets” objects
- A function is created to retrieve following gear fishing effort metrics by fishing trip :
  - soaked nets length and number
  - hauled nets length and number
  - Statistics on soaking times for hauled nets (mean and median)

```
Calc_GearFishingEffort <- function(nets, pos){

  trajDesc <- Traj_Desc(pos)

  stats_nets <- function(x){
                return(c(total.lg  = sum(as.numeric(x), na.rm = TRUE),
                         nb.nets  = length(x)))
  }

  stats_skTime <- function(x){
                return(c(mn.sk  = mean(as.numeric(x), na.rm = TRUE),
                         md.sk = stats::median(as.numeric(x), na.rm = TRUE)))
  }

  Lg.Nets.soaked <-  as.data.frame(
                      aggregate(nets$length,
                        by = list(nets$FT_Setting),
                        FUN = stats_nets))

  colnames(Lg.Nets.soaked)[1] <- "FISHING_TRIP_FK"
  Lg.Nets.soaked$lgNets.soaked <- Lg.Nets.soaked$x[,1]
  Lg.Nets.soaked$nbNets.soaked <- Lg.Nets.soaked$x[,2]
  Lg.Nets.soaked <- Lg.Nets.soaked[, c("FISHING_TRIP_FK", "lgNets.soaked",  "nbNets.soaked")]

  Lg.Nets.hauled <-  as.data.frame(
                      aggregate(nets$length,
                        by = list(nets$FT_Hauling),
                        FUN = stats_nets))

  colnames(Lg.Nets.hauled)[1] <- "FISHING_TRIP_FK"
  Lg.Nets.hauled$lgNets.hauled <- Lg.Nets.hauled$x[,1]
  Lg.Nets.hauled$nbNets.hauled <- Lg.Nets.hauled$x[,2]
  Lg.Nets.hauled <- Lg.Nets.hauled[, c("FISHING_TRIP_FK", "lgNets.hauled",  "nbNets.hauled")]

  Lg.Nets <- merge(
              merge(trajDesc,
                    Lg.Nets.soaked,
                    by = "FISHING_TRIP_FK",
                    all.x = TRUE),
              Lg.Nets.hauled,
              by = "FISHING_TRIP_FK",
              all.x = TRUE)


  SKT.Nets.hauled <-  as.data.frame(
                        aggregate(nets$soaking.time.hours,
                          by = list(nets$FT_Hauling),
                          FUN = stats_skTime))

  colnames(SKT.Nets.hauled)[1] <- "FISHING_TRIP_FK"
  SKT.Nets.hauled$mean.SKT <- SKT.Nets.hauled$x[,1]
  SKT.Nets.hauled$median.SKT<- SKT.Nets.hauled$x[,2]
```

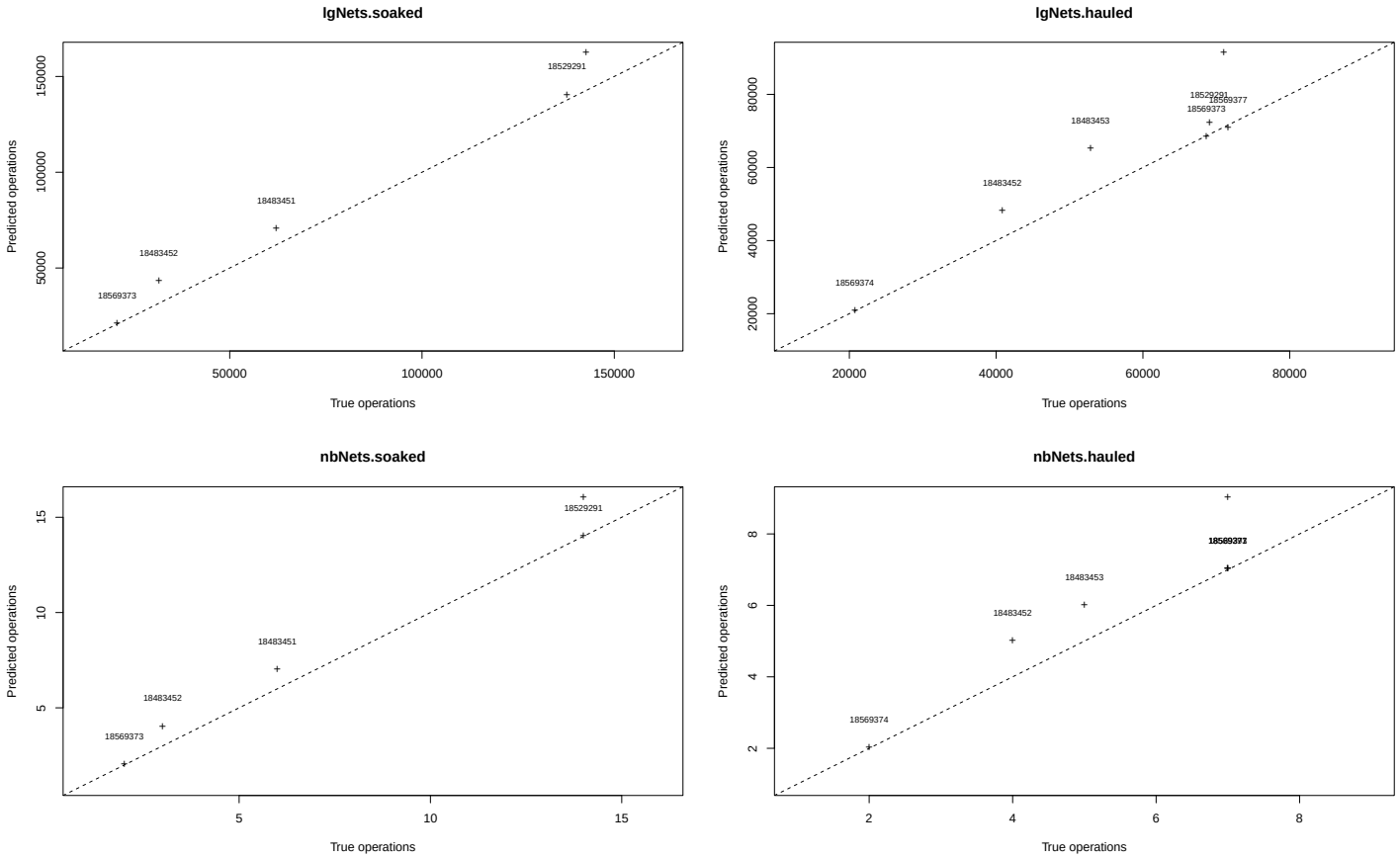
```
  SKT.Nets.hauled <- SKT.Nets.hauled[, c("FISHING_TRIP_FK", "mean.SKT",  "median.SKT")]

  SKT.Nets <- merge(trajDesc,
                    SKT.Nets.hauled,
                    by = "FISHING_TRIP_FK",
                    all.x = TRUE)

  GearFE <- merge(Lg.Nets,
                  SKT.Nets[, !colnames(SKT.Nets) %in% c("Start.FT", "End.FT")],
                  by = c("VESSEL_FK", "FISHING_TRIP_FK"))

  return(GearFE)
}
```

- Calculation of predicted and true gear fishing effort metrics

```
gearFE.true <- Calc_GearFishingEffort(Nets.real, posForMl)
gearFE.RfGeo <- Calc_GearFishingEffort(Nets.rfMod, posForMl)

gearFE.true %>%
    head %>%
    print_table
```

VESSEL_FK	FISHING_TRIP_FK	Start.FT	End.FT	lgNets.soaked	nbNets.soaked	lgNets.hauled	nbNets.hauled	mean.SKT	median.SKT
NAVIRE_0075	18483451	2021-12-23 06 :55 :23	2021-12-24 07 :55 :23	62 142.75	6	NA	NA	NA	NA
NAVIRE_0075	18483452	2021-12-26 03 :00 :53	2021-12-27 12 :30 :53	31 569.92	3	40 840.13	4	75.97979	73.32778
NAVIRE_0075	18483453	2021-12-27 18 :46 :47	2021-12-29 21 :46 :47	NA	NA	52 872.53	5	82.31567	73.75167
NAVIRE_0075	18529291	2022-01-09 10 :43 :18	2022-01-13 18 :28 :18	137 712.92	14	69 070.47	7	58.46393	60.75028
NAVIRE_0075	18569373	2022-01-13 19 :43 :17	2022-01-16 12 :43 :17	20 759.69	2	68 642.45	7	70.56948	69.37972
NAVIRE_0075	18569374	2022-01-16 14 :19 :23	2022-01-17 10 :19 :23	NA	NA	20 759.69	2	64.21764	64.21764

- Comparison of computed number and length of nets deployed by fishing trip



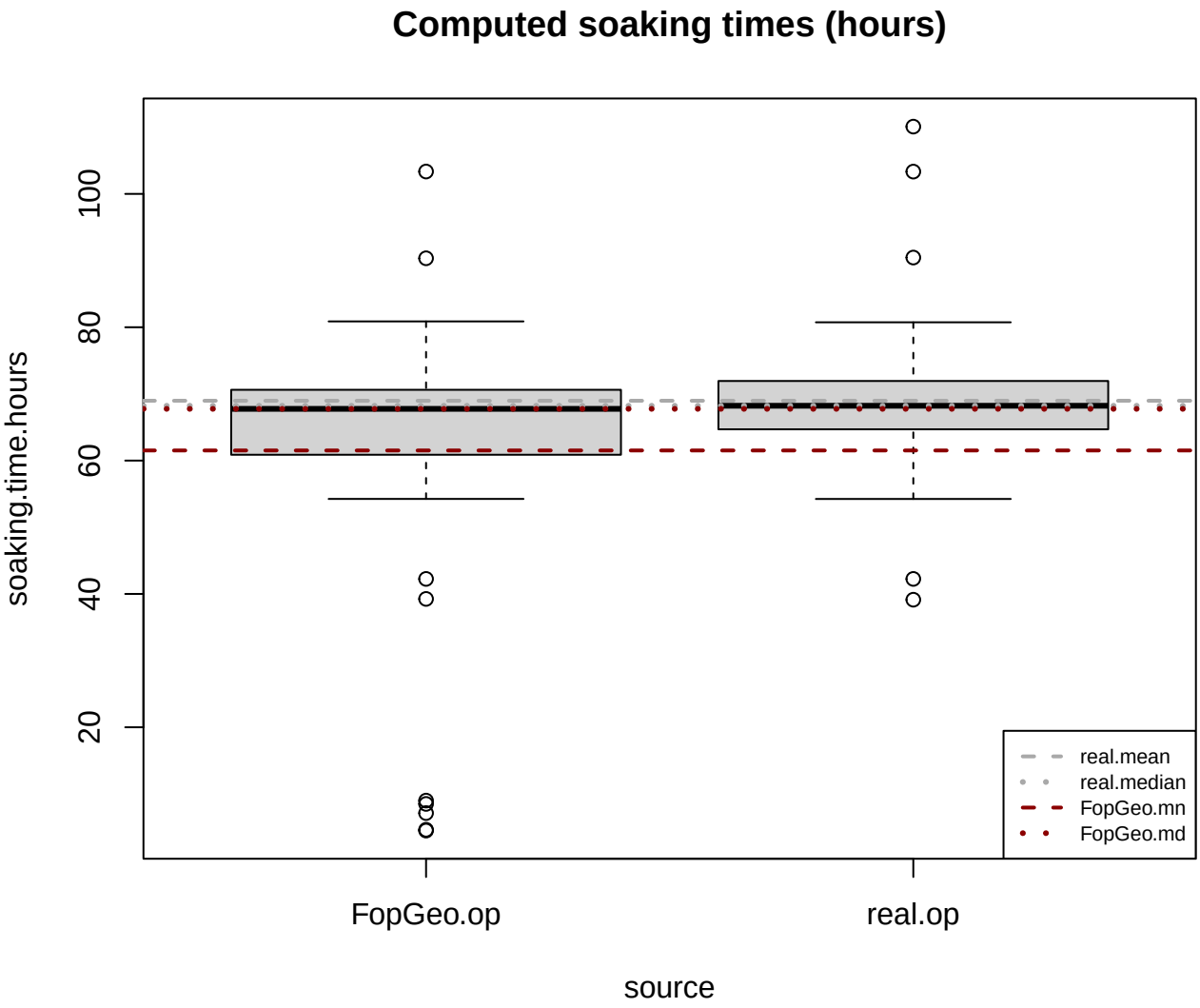
- Regarding the number of nets deployed :
  - a total of 39 nets soaked and 39 hauled
  - a total of 44 nets soaked and 44 hauled when the predicted operations are used
- Regarding the length of nets deployed :
  - a total of 394811 meters of nets soaked and 394811 meters of nets hauled
  - a total of 448607 meters of nets soaked and 448607 meters of nets hauled when the predicted operations are used

4.7.3 Soaking times

- Comparison of computed soaking times by gear

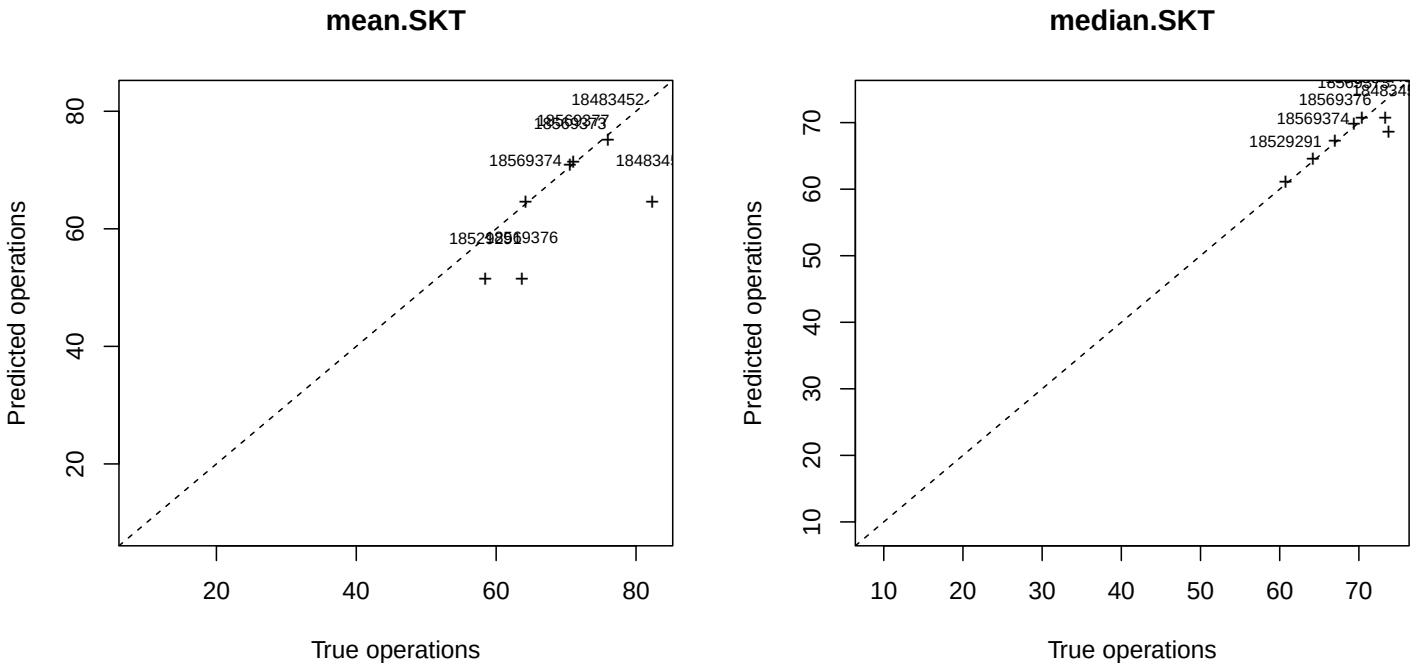
```
stats.sk <- st_set_geometry(Nets.real, NULL)
stats.sk$source <- "real.op"
pred.sk <- st_set_geometry(Nets.rfMod, NULL)
pred.sk$source <- "FopGeo.op"
stats.sk <- rbind(stats.sk, pred.sk)

boxplot(soaking.time.hours ~ source,
        stats.sk, main = "Computed soaking times (hours)")
abline(h = mean(Nets.real$soaking.time.hours, na.rm = TRUE),
       lty = 2, lwd = 2, col = "darkgray")
abline(h = mean(Nets.rfMod$soaking.time.hours, na.rm = TRUE),
       lty = 2, lwd = 2, col = "darkred")
abline(h = median(Nets.real$soaking.time.hours, na.rm = TRUE),
```

```
       lty = 3, lwd = 3, col = "darkgray")
abline(h = median(Nets.rfMod$soaking.time.hours, na.rm = TRUE),
       lty = 3, lwd = 3, col = "darkred")
legend(x = "bottomright", legend = c("real.mean", "real.median", "FopGeo.mn", "FopGeo.md"),
       cex = 0.7, lty = c(2,3,2,3), lwd = c(2,3,2,3),
       col = c(rep("darkgray",2), rep("darkred",2)))
```



- Using the predicted operations may lead to an underestimation of soaking times due to 5 operations for which the computed soaking time is below 10 hours :
  - using the averaged values it is underestimated by 10.8%
  - the median appears to be a better metric driving to an underestimation of 0.7% only.
- Comparison of computed soaking times for hauled nets by fishing trip

```
par(mfrow = c(1,2))

plot_FE(gearFE.true, gearFE.RfGeo, "mean.SKT")
plot_FE(gearFE.true, gearFE.RfGeo, "median.SKT")
```



- Using thresholds on soaking time, the consolidation could be improved. It wasn't included by default in *Create_NetsByBoat* for the calculation to remain reasonable (Nets have to be created two times when using the *Consolidate_Nets* methods within this function).
- In order to build a clean dataset for qualification, this step could be included in the function *Retrieve_SettingOperations* by defining thresholds using nested iapesca functions :

```
summary(Nets.rfMod$soaking.time.hours)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   4.50   60.94   67.76   61.52   70.54  103.36
```

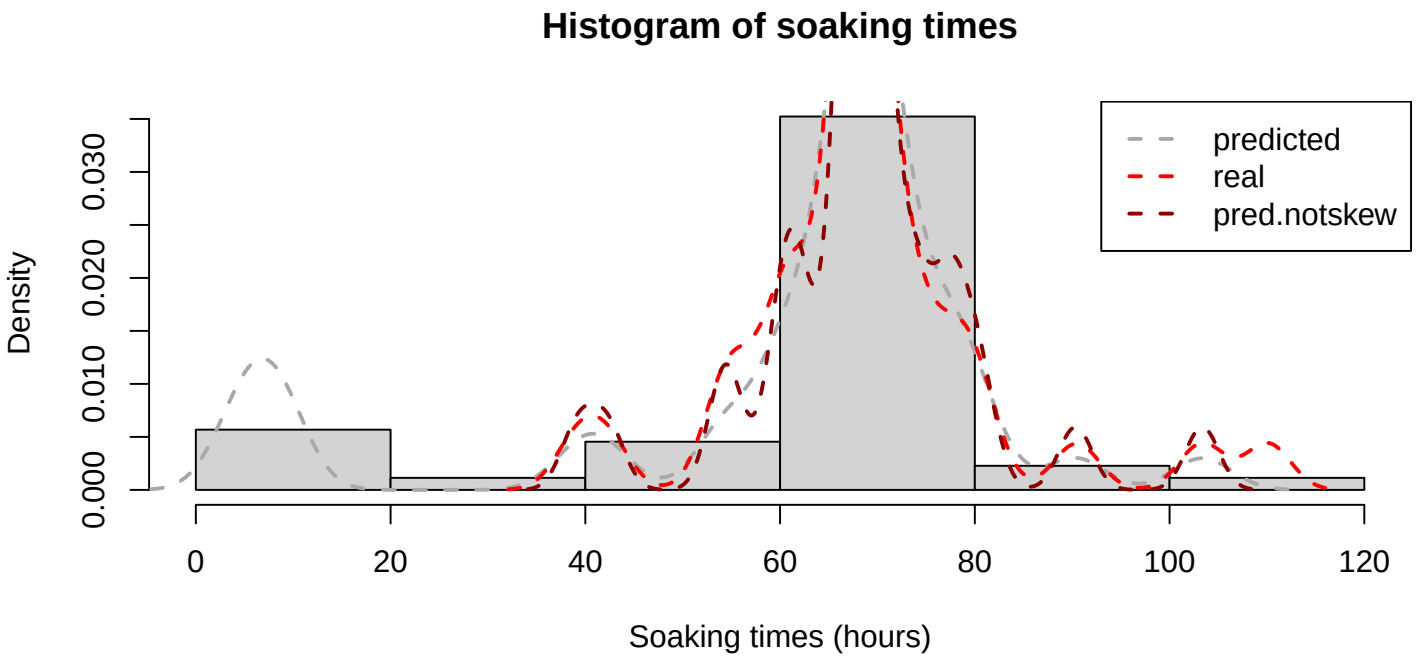
```
summary(CleanSpuriousValues(Nets.rfMod$soaking.time.hours))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  59.60   67.07   68.01   67.73   69.57   74.00      16
```

```
summary(CleanSkewness(Nets.rfMod$soaking.time.hours))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  39.25   66.67   68.50   68.54   72.02  103.36       5
```

```
is.skew <- is.na(CleanSkewness(Nets.rfMod$soaking.time.hours))

hist(Nets.rfMod$soaking.time.hours, freq=FALSE,
     main = "Histogram of soaking times", xlab = "Soaking times (hours)")
lines(density(Nets.rfMod$soaking.time.hours), col = "darkgray", lty = 2, lwd = 2)
lines(density(Nets.real$soaking.time.hours), col = "red", lty = 2, lwd = 2)
lines(density(Nets.rfMod$soaking.time.hours[!is.skew]), col = "darkred", lty = 2, lwd = 2)
legend(x = "topright", legend= c("predicted", "real", "pred.notskew"), lty = 2,
       col = c("darkgray", "red", "darkred"), lwd = 2)
```



- A basic control is provided in *Retrieve_SettingOperations* with the argument tol.soaktime defined by default at 240 hours. The function searchs for possible setting during the last 3 fishing trips. These parameters could be made adjustable to include more control when detecting setting events.

5 - Developments for iapesca v1 (2023)

- Test the robustness of the package on larger datasets with higher temporal resolution and eventually improve process execution speed.
- Implement user-defined buffers for hauling and setting in *Retrieve_SettingOperations* function and computation of setting speeds.
- Include a consolidation process based on computed soaking times thresholds when retrieving setting operations : "Auto.ThreshHolds.Detection" method added in *Retrieve_SettingOperations* function. (Split the function, using the argument "tol.soaktime" as a vector of length two).
- Functions for computing vessel and gears fishing efforts metrics : fishing days/hours, length and soaking times using positions and/or gear objects.
- Functions for rasterizing fishing effort metrics from positions and/or gear objects using user-defined C.squares.
- Functions for mapping different fishing effort metrics.
- Function for converting French FIS consolidated logbooks format (SACROIS) to eflalo used in vsmtools.