Supplement to: Video survey of deep benthic macroalgae and macroalgal detritus along a glacial Arctic fjord: Kongsfjorden (Spitsbergen)

Katherina Schimani[1], Katharina Zacher[1], Kerstin Jerosch[1], Hendrik Pehlke[1], Christian Wiencke[1], Inka Bartsch[1]

[1] Alfred-Wegener-Institute Helmholtz-Center for Polar and Marine Research, Am Handelshafen 12, 27570 Bremerhaven, Germany

Corresponding author: Katherina Schimani, k.schimani@bo.berlin
Current address: Botanischer Garten und Botanisches Museum Berlin, Freie Universität Berlin, Königin-Luise-Str. 6-8, 14195 Berlin, Germany

```
################################################################################
################################################################################
##
##        R-program to load and manipulate the ROV--transect files
##
##                (c)  Hendrik Pehlke and Katherina Schimani
##                             last change: 24.05.2019
##
##                                 R-version: R-3.5
##
################################################################################
################################################################################

################################################################################
##      CONTENT
################################################################################

##  Step 1: Installation and loading of R packages
##  Step 2: creation of necessary functions
##  Step 3: Setting of work directories
##  Step 4: Loading of ROV transect files
##  Step 5: Correction of the geographic coordinates (Longitude, Latitude)
##  Step 6: Conversion into SpatialPointDataFrame
##  Step 7: Conversion of the x-y points to a line


################################################################################
##      Step 1: Installation and loading of R packages

rm(list=ls())          # Removes all files in the working (global)
environment
cat("\014")            # clear console
graphics.off()         # clear all plots
options(scipen=999)    # disable scientific notation

# unload all loaded packages
lapply(paste('package:',names(sessionInfo()$otherPkgs),sep=""),
       detach,character.only=TRUE,unload=TRUE)

#  install.packages(openxlsx)
library(openxlsx,   lib.loc="~/R/win-library/3.5")

#  install.packages(sp)
library(sp,   lib.loc="~/R/win-library/3.5")

#  install.packages(maptools)
library(maptools,   lib.loc="~/R/win-library/3.5")
```

```
#  install.packages(leaflet)
library(leaflet,  lib.loc="~/R/win-library/3.5")

#  install.packages(tools)
library(tools,  lib.loc="~/R/win-library/3.5")

#  install.packages(rgdal)
library(rgdal,  lib.loc="~/R/win-library/3.5")


##############################################################################
##     Step 2: creation of necessary functions

# Create function to convert XY coordinates to lines in R

# creation of a function containing 5 variables, two of them are set to
NULL
points_to_line <- function(data, long, lat, id_field = NULL, sort_field =
NULL) {
  # Convert to SpatialPointsDataFrame

  coordinates(data) <- c(long, lat)

  # If there is a sort field...
  if (!is.null(sort_field)) {
    if (!is.null(id_field)) {

      data <- data[order(data[[id_field]], data[[sort_field]]), ]
    } else {
      data <- data[order(data[[sort_field]]), ]
    }
  }

  # If there is only one path...
  if (is.null(id_field)) {

    # spatiallines= function to create lines
    lines <- SpatialLines(list(Lines(list(Line(data)), "id")))

    return(lines)

    # Now, if we have multiple lines...
  } else if (!is.null(id_field)) {

    # Split into a list by ID field
    paths <- sp::split(data, data[[id_field]])

    sp_lines <- SpatialLines(list(Lines(list(Line(paths[[1]])), "line1")))

    # I like for loops, what can I say...
    for (p in 2:length(paths)) {
      id <- paste0("line", as.character(p))

      l <- SpatialLines(list(Lines(list(Line(paths[[p]])), id)))
      # sprbind connects objects
      sp_lines <- spRbind(sp_lines, l)
    }
    return(sp_lines)
  }
}
```

```
################################################################################
##     Step 3: Setting of work directories

# now choose the directory, where the ROV-transect files are located
# new variable: data.path ti set directory
data.path <- choose.dir(default = "C:\\Users\\kschiman\\Desktop\
\Bachelorarbeit\\03_ROV  Protokolle\\02_Bereinigt",
                        caption = "Select folder with the transect files")

# new variable: storage_path to set storage directory
storage_path <- choose.dir(default = "C:\\Users\\kschiman\\Desktop\
\Bachelorarbeit\\03_ROV Protokolle\\03_Output R",
                           caption = "Select folder to save results")

################################################################################
##     Step 4: Loading of ROV transect files

# change working directory to 'data.path'
setwd(data.path)
txt.list <- dir(pattern="txt")
xlsx.list <- dir(pattern="xlsx")

a <- 1

for (a in 1:length(xlsx.list)){
  print(a)
    name <- tools::file_path_sans_ext(xlsx.list[a])
  # file_path_sans_ext returns the file paths without extensions
  print(name)

  if (exists("temp")) {rm(temp)}
  temp <- read.delim(txt.list[a],
                     stringsAsFactors=FALSE,
                     na.strings="")

  temp <- temp[rowSums(is.na(temp)) != ncol(temp),]

  if (exists("Test")) {rm(Test)}
  Test <- read.xlsx(xlsx.list[a],
                    sheet = 1,
                    startRow = 1,
                    colNames = TRUE,
                    rowNames = FALSE,
                    detectDates = FALSE,
                    skipEmptyRows = TRUE,
                    skipEmptyCols = FALSE,
                    rows = NULL,
                    cols = c(1:16),
                    check.names = FALSE,
                    namedRegion = NULL,
                    na.strings = "NA",
                    fillMergedCells = FALSE)
  # drop all rows containing 'NA'
  # Test <- Test[complete.cases(Test), ]     # just complete rows

  Test[,c(1:2)] <- NA
  Test[,c(1:2)] <- temp[,c(1:2)]
  Test$Timecode <- temp$Timecode
```

```
  # Test$DATE <- c(Test$DATE[1])
  # convert column 'DATE' to a date value in R
  # Test$DATE <- as.Date(Test$DATE, format='%d %m %Y')
  # convert column "HEURE" to a time value in R
 # Test$HEURE <- format(as.POSIXct((as.numeric(Test$HEURE)) *
 #  86400, origin = "1970-01-01",tz = "UTC"),"%H:%M:%S")

  Test <- Test[complete.cases(Test), ]

  # now give assign the data frame 'Test' to a new unique name
  # Assign a value to a name in an environment
  assign(paste("track",name,sep="_"),
         Test)
  c.names <- colnames(Test)
  # cleaning
  the wokspace
  rm(name,
  Test)

  }
rm(a)

# create a list containing all track data frmae names in the workspace
# ls lists all data in WORKSPACE, which have "track" in
# their name  track.list <- ls(pattern = "track_")

# Creation of an excel file containing all tracks
# with 2 new colloums: name of track and sequenz for sorting
# the data of  one track

all.tracks <- as.data.frame(mget(track.list[1]))
# as.data.freme: Functions to change if an object is a data
frame, or  coerce it if possible, mget: Search by name for
more objects
# Funktion names: get or set
worksheet names  names(all.tracks) <-
c.names
all.tracks$t_name  <-
c(tools::file_path_sans_ext(xlsx.list[1]))
all.tracks$sort_field <- seq.int(nrow(all.tracks))

a <- 2
for (a in 2:length(track.list)){

  if (exists("Temp")) {rm(Temp)}
  Temp <-
  as.data.frame(mget(track.list[a]))
  names(Temp) <- c.names
  Temp$t_name <- c(tools::file_path_sans_ext(xlsx.list[a]))
  Temp$sort_field <- seq.int(nrow(Temp))

  # combined all.tracks is created from all all.tracks
  und Temp  all.tracks <- rbind(all.tracks, Temp)
  rm(Temp)
}
rm(a)
```

```
################################################################################
#### Step 5: Correction of the geographic coordinates (Longitude, Latitude)

# show all column names
names(all.tracks)
if (exists("df_01")) {rm(df_01)}
df_01 <- all.tracks

# show head of data
head(df_01)

# check coordinates correct them and and recalculate them to decimal degree
# (e.g. 78.2456°)
if  (exists("Long_Lat_df_01_sub")){rm(Long_Lat_df_01_sub)}

# select the columns with (still incorrect) Latitude and Longitude values,
# selection gets the name mm
mm <- select.list(c(names(all.tracks)),
                  #preselect = c(names(df_data_shp_EPSG_4326)),
                  multiple = TRUE,
                  title = "Select Lat/Long data",
                  graphics = TRUE)

# make a new data frame just with the (still incorrect) Latitude and
# Longitude columns
Lat_Long_df_01_sub <- all.tracks[c(mm)]
rm(mm)

# create two new columns to store the results in
# two new coloums are generated and set NA
Lat_Long_df_01_sub$Lat_Degree_Dezi_Min <- NA
Lat_Long_df_01_sub$Long_Degree_Dezi_Min <- NA

a <- 1
for (a in 1:nrow(Lat_Long_df_01_sub)){

  if (exists(c("test1", "test2"))) {rm(test1, test2)}

  test1 <-
as.numeric(substr(as.character(Lat_Long_df_01_sub$Latitude[a]),start = 1,
stop = 2))
  test2 <-
as.numeric(substr(as.character(Lat_Long_df_01_sub$Latitude[a]),start = 3,
stop = 100))/60
  Lat_Long_df_01_sub$Lat_Degree_Dezi_Min[a] <- test1+test2
  rm(test1, test2)

# in case of Longitude: first select just the first two digits (= degree, e.g.
# 78) and store # them as number, then select the rest of the digits (=
# minutes, e.g. 56.242) and divide the value with 60 to calculate from
# minutes to decimal degrees. Then add the two values
```

```
  test1 <-
as.numeric(substr(as.character(Lat_Long_df_01_sub$Longitude[a]),start  =  1,
stop = 2))
  test2 <-
as.numeric(substr(as.character(Lat_Long_df_01_sub$Longitude[a]),start  =  3,
stop = 100))/60
  Lat_Long_df_01_sub$Long_Degree_Dezi_Min[a]  <-  test1+test2
  rm(test1,  test2)
}
rm(a)

# now add column with transect name ("t_name") and the column which defines
# the order of the
# points in each transect "sort_field" from "all.tracks"

Lat_Long_df_01_sub$t_name  <-  all.tracks$t_name
Lat_Long_df_01_sub$sort_field  <-  all.tracks$sort_field

# just keep the columns with the recalculated Lat/ Long values, the t_names
# and the sort id number
# colums 1 and 2 with incorrect coordinates are not taken
Lat_Long_df_01_sub  <-  Lat_Long_df_01_sub[,c(3,4,5,6)]

#############################################################################
##  Step 6: Conversion into SpatialPointDataFrame
# #         library(sp)

# if there are still some NAs in the Latitude or Longitude data, then drop
# those rows
df  <-  Lat_Long_df_01_sub[complete.cases(Lat_Long_df_01_sub),  ]

# now just save the Longitude and Latitude columns in 'spatial_points'
spatial_points  <-  df[,c(2:1)]

# rename those columns with "x" and "y"
names(spatial_points)  <-  c("x",  "y")

coordinates(spatial_points)  <-  ~x+y
plot(spatial_points,  pch=19,  col=  "red")

#rm(df,  Lat_Long_df_01_sub)
#############################################################################
##  Step 7: Convertion of the x-y points to a line

# nn <- c(unique(df$t_name))
#
# if (exists(c("temp", "t_lines"))) {rm(temp, t_lines)}
# temp <- df[df$t_name==nn[c(1:2)],]
t_lines <- points_to_line(data = df,
                          long = "Long_Degree_Dezi_Min",
                          lat = "Lat_Degree_Dezi_Min",
                          id_field = "t_name",
                          sort_field = "sort_field"
                          )
plot(t_lines)
leaflet(data = t_lines) %>%
```

```
  addTiles()  %>%
  addPolylines()

# convert to SpatialLinesDataFrame as the first step to export the lines
# as a shapefile
SLDF  =  SpatialLinesDataFrame(t_lines,
                               data.frame(name  =
c(tools::file_path_sans_ext(xlsx.list)),
                                          row.names  =  c("line1",
                                                          "line2",
                                                          "line3",
                                                          "line4",
                                                          "line5",
                                                          "line6")))

as.data.frame(SLDF)

# EPSG strings
latlong  =  "+init=epsg:4326"
proj4string(SLDF)  =  CRS(latlong)

SLDF$length_m  <-  round(SpatialLinesLengths(SLDF)*1000,0)
as.data.frame(SLDF)


leaflet(data  =  SLDF)  %>%
  addTiles()  %>%
  addPolylines()

rgdal::writeOGR(SLDF,
                dsn=storage_path  ,
                layer="ROV_tracks",
                driver="ESRI  Shapefile",
                overwrite_layer=TRUE)

# save "all_tracks" as csv-file
setwd(storage_path)

# now add the corrected lon/Lat columns to "spatial_points"
test  <-  as.data.frame(spatial_points,  xy=TRUE)
test  <-  cbind(test,  all.tracks)

# export of the data in a csv file
write.csv(test,  file  =  "all_rov_track_points_with_corrected_Lat_long.csv",
sep="\t",  row.names=FALSE)
```