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 extract sample coordinates of the transect
##
##               (c)  Hendrik Pehlke and Katherina Schimani
##                           last change: 24.05.2019
##
##                               R-version: R-3.5
##
################################################################################
################################################################################


  library("SDraw", lib.loc="~/R/win-library/3.5")
  library("rgeos", lib.loc="~/R/win-library/3.5")
  library("rgdal", lib.loc="~/R/win-library/3.5")
  library("sp", lib.loc="~/R/win-library/3.5")
  library("sf", lib.loc="~/R/win-library/3.5")
  library("svDialogs", lib.loc="~/R/win-library/3.5")


################################################################################
#
#           create functions
#
################################################################################
SegmentSpatialLines <- function(sl, length = 0, n.parts = 0, merge.last
=  FALSE) {
  stopifnot((length > 0 || n.parts >
  0)) id <- 0
  newlines <- list()
  sl <- as(sl, "SpatialLines")
  for (lines in sl@lines) {
    for (line in lines@Lines)
      { crds <- line@coords
      # create segments
      segments <- CreateSegments(coords = crds, length,
      n.parts)  if (merge.last && length(segments) > 1) {
        # in case there is only one segment, merging would result into error

  segments  <-  MergeLast(segments)
```

```
}
      # transform segments to lineslist for SpatialLines
      object for (segment in segments) {
        newlines <- c(newlines, Lines(list(Line(unlist(segment))), ID
= as.character(id)))
        id <- id + 1
      }
    }
  }
  return(SpatialLines(newlines))
}

##############################################################################

CreateSegment <- function(coords, from, to) {
  distance <- 0
  coordsOut <- c()
  biggerThanFrom <- F
  for (i in 1:(nrow(coords) - 1)) {
    d <- sqrt((coords[i, 1] - coords[i + 1, 1])^2 + (coords[i, 2] -
              coords[i +1, 2])^2)
    distance <- distance + d
    if (!biggerThanFrom && (distance > from))
      { w <- 1 - (distance - from)/d
      x <- coords[i, 1] + w * (coords[i + 1, 1] - coords[i,
      1]) y <- coords[i, 2] + w * (coords[i + 1, 2] -
      coords[i, 2]) coordsOut <- rbind(coordsOut, c(x, y))
      biggerThanFrom <- T
    }
    if (biggerThanFrom)
      { if (distance >
      to) {
        w <- 1 - (distance - to)/d
        x <- coords[i, 1] + w * (coords[i + 1, 1] - coords[i,
        1]) y <- coords[i, 2] + w * (coords[i + 1, 2] -
        coords[i, 2]) coordsOut <- rbind(coordsOut, c(x, y))
        break
      }
      coordsOut <- rbind(coordsOut, c(coords[i + 1, 1], coords[i + 1,
                                                               2]))
    }
  }
  return(coordsOut)
}

##############################################################################

CreateSegments <- function(coords, length = 0, n.parts = 0) {
  stopifnot((length > 0 || n.parts > 0))
  # calculate total length
  line total_length <- 0
  for (i in 1:(nrow(coords) - 1)) {
    d <- sqrt((coords[i, 1] - coords[i + 1, 1])^2 + (coords[i, 2] -
coords[i +

1, 2])^2)
    total_length <- total_length + d
  }
```

```
  # calculate stationing of
  segments if (length > 0) {
    stationing <- c(seq(from = 0, to = total_length, by =
length), total_length)
  } else {
    stationing <- c(seq(from = 0, to = total_length, length.out =
                    n.parts), total_length)
  }

  # calculate segments and store the in list
  newlines <- list()
  for (i in 1:(length(stationing) - 1)) {
    newlines[[i]] <- CreateSegment(coords, stationing[i], stationing[i +
                  1])
  }
  return(newlines)
}

################################################################################

MergeLast <- function(lst)
  { l <- length(lst)
  lst[[l - 1]] <- rbind(lst[[l - 1]],
  lst[[l]]) lst <- lst[1:(l - 1)]
  return(lst)
}
################################################################################
#
#          load data
#
################################################################################
if (exists("ROV_transect")) {rm(ROV_transect)}

ROV_transect <-
read.csv("C:/Users/kschiman/Desktop/Bachelorarbeit/05_R/
03_Output_R/all_rov_track_points_with_corrected_Lat_long.csv")

print(unique(ROV_transect$t_name))
ROV_transect <- ROV_transect[ROV_transect$t_name =="ROVAWIpev013",]

# make a subset: Just keep points with ''
unique(ROV_transect$statusCamera)


# check if there are NA's in column "Longitude" or
"Latitude" ROV_transect <-
ROV_transect[!is.na(ROV_transect[,'x']),] ROV_transect <-
ROV_transect[!is.na(ROV_transect[,'y']),]

# select the columns with spatial information: x (Longitude) and
y (Latitude)
x <- c(ROV_transect$x)          #
Longitude y <- c(ROV_transect$y) #
Latitude
```

```
#############################################################################
#
#           create line and change projection
#
#############################################################################

if  (exists("ROV_line"))  {rm(ROV_line)}
ROV_line<-SpatialLines(list(Lines(Line(cbind(x,y)),   ID="a")))
proj4string(ROV_line)  <-  CRS("+init=epsg:4326")

## transform projection into ESRI Projection 102017 - North Pole
Lambert  Azimuthal  Equal  Area
ROV_line  <-  sp::spTransform(ROV_line,CRS("+proj=laea  +lat_0=90  +lon_0=0
             +x_0=0+y_0=0  +ellps=WGS84+datum=WGS84  +units=m  +no_defs"))

#############################################################################
#
#           extract  coordinates  of  reprojected  line
#
#############################################################################

if  (exists(c("xy_coord")))  {rm(xy_coord)}
xy_coord  <-
as.data.frame(coordinates(ROV_line))
str(xy_coord)
names(xy_coord)  <-  c("x_Rechtswert",  "y_Hochwert")
xy_coord$geometry  <-  paste(xy_coord[,1],  "  "
,xy_coord[,2],sep="")

xy_coord$Sonde.m.  <-
ROV_transect$Sonde.m.  xy_coord$Timecode
                    <-
ROV_transect$Timecode xy_coord$heure  <-
ROV_transect$HEURE
#r  =  cbind(xy_coord[,1],  xy_coord[,2])


#############################################################################
#
#     split  lines  according  to  selected  distance  ('interval_length')  and
#     create  new  points
#
#############################################################################

# define  interval  length  (in  meter)
if  (exists("interval_length"))  {rm(interval_length)}
# 'dlg_input' is a function from the package 'svDialogs'
interval_length  <-  5
if  (exists(c("test2")))  {rm(test2)}
test2  <-  SegmentSpatialLines(ROV_line,  length  =  interval_length,  merge.last
=  FALSE)
#gLength(ROV_line)

graphics.off()
plot(ROV_line)
```

```
col = "red"
for (i in 1:length(test2)) {
  col <- ifelse(col == "red", "black", "red")
  #lines(as.matrix(test2[[i]]), col = col, lwd =
  2) lines(test2[i], col = col, lwd = 4)
}

##############################################################################
#
#    create a result table for lines, points and their coordinates and
#    check length of created segment lines
#
##############################################################################

# create an empty result table for lines, points and their coordinates
point_table <- as.data.frame((matrix(NA, nrow= 0, ncol=4)))
names(point_table) <- c("line_nr", "point_nr","x_Rechtswert",
"y_Hochwert")

# now fill the empty resault table with
data a <- 1
for (a in 1:length(test2)){
  # create an empty temporal result table for each run
  temp_table <- as.data.frame((matrix(NA, nrow= 0, ncol=4)))
  # add column names to the temporal table
  names(temp_table) <- c("line_nr", "point_nr","x_Rechtswert",
"y_Hochwert")

  print(a)
  if (exists(c("pp", "nn", "bb"))){rm(pp, nn, bb)}

  # extract all coordinates of the new created line of this
  run pp <- as.data.frame(coordinates(test2)[[a]][[1]])
  # count the rows (="points") of this
  segment nn <- nrow(pp)

  bb <- 1
  for(bb in 1:nn){
    # add line segment number to temporal result table
    temp_table[c(nrow(temp_table)+1),1] <- a
    temp_table[nrow(temp_table),2] <- paste(a,".",bb,sep="")
    # add x and y coordinate (Rechts- und Hochwert) to temporal result
table
    temp_table[nrow(temp_table),3] <- pp[bb,1]
    temp_table[nrow(temp_table),4] <- pp[bb,2]
  }

  # combine the final point table with the temporal result table
  point_table <- rbind(point_table, temp_table)
  rm(pp, nn, bb,
  temp_table) a <- a+1
}
```

```
#############################################################################
#
#     now calculate the length of every new created segment (should be
#     equal to interval_length)
#           Exception: the last segment can be shorter
#
#############################################################################

# test new line segment lengths
if (exists(c("ll", "a"))){rm(ll, a)}

length_list <- c()     # Liste für alle ermittelten

Streckenlängen ll <- c(unique(point_table$line_nr))
a <- 1
for (a in 1:length(ll)){
  print(ll[a])
  if (exists(c("pt_sub", "df","lb", "tmp"))){rm(pt_sub, df, lb, tmp)}
  pt_sub <- point_table[point_table$line_nr==ll[a],]
  lb <- nrow(pt_sub)

  df <- Line(data.frame(pt_sub$x_Rechtswert, pt_sub$y_Hochwert)) %>%
    list() %>%
    Lines(ID = 1) %>%
    list() %>%
    SpatialLines()

  proj4string(df) <- CRS("+proj=laea +lat_0=90 +lon_0=0 +x_0=0 +y_0=0
+ellps=WGS84+datum=WGS84 +units=m +no_defs")
  #gLength(df)

  tmp <- rep(paste("Total length of line ", a, ": ",
round(gLength(df),0), " m", sep=""), lb)

  length_list <- c(length_list, tmp)

  rm(pt_sub, df, lb, tmp)
}

point_table$line_length <-
length_list unique(length_list)
rm(length_list)

#############################################################################
#
#     combine (merge) new point table with the original ROV points to detect
#     the new points
#
#############################################################################

point_table$geometry <- paste(point_table$x_Rechtswert, " ",
point_table$y_Hochwert, sep="")
allpoints <- merge(point_table, xy_coord, by = "geometry", all = TRUE, sort
= FALSE)
allpoints <- merge(xy_coord, point_table,   by = "geometry", all =
TRUE, sort = FALSE)

# Sort by column index [1] then [3]
allpoints <- allpoints[order(allpoints$line_nr ,
```

```
allpoints$heure, allpoints$point_nr ),]
points_ROV_and_segments = allpoints[!duplicated(allpoints$geometry),]
points_ROV_and_segments <- points_ROV_and_segments[-1]
points_ROV_and_segments$projection <- "ESRI Projection 102017 -
North Pole Lambert Azimuthal Equal Area"
names(points_ROV_and_segments)
#points_ROV_and_segments <- points_ROV_and_segments[-c(1,2)]
write.csv(points_ROV_and_segments, file = "C:/Users/kschiman/Desktop/
Bachelorarbeit/05_R/03_Output_R/Ergebnis_split_lines.csv")
```