Schools detection using the R package acoustic to control Echoview via COM

Martin J. Cox

January 12, 2015

1 Introduction

In this vignette we will use the example acoustic data collected during the KOAS voyage to demonstrate how to carry out schools detection using `acoustic` to control Echoview via COM. We assume the reader has knowledge of the schools detection algorithm implemented in Echoview. If not please visit `http://support.echoview.com/WebHelp/Echoview.htm/` to see the Echoview help file and also ... We also assume the reader is familar with Echoview concepts such as filesets, regions, files and virtual variables.

To run this example, you will need the following files:

1. **Echosounder data:** Simrad EK60 RAW data collected during the KAOS voyage.
2. **An Echoview template file:** An Echoview (.EV) file containing the virtual variables used during data processing and must be copied to the Echoview templates folder.
3. **Echoview calibration file:** The Echoview .ECS file containing the EK60 calibration parameters.
4. **Start and end file regions:** Start and end times of each transect stored as an Echoview region file.

All these files are available from ... and should be downloaded to the data directory specified in the `dd` object. `If you want to reproduce this vignette, you should assign the location of the KAOS example data to dd.`. This vignette requires you to have run the Read Data example first.

First, set the working directory. Echoview requires the full file path to be specified for every file name passed using COM.

```
> dd <- 'c:/Users/Lisa/Desktop/KAOS'
```

2 Loading RAW data into Echoview

In this section we load the necessary packages into the R workspace, open a COM connection between R and Echoview, then populate the Echoview template file with RAW Simrad EK60 data files.

```
> library(acoustic)
```

Open a COM connection between R and Echoview:

```
> EVAppObj <- COMCreate('EchoviewCom.EvApplication')
```

Get a list of the raw files required for each transect:

```
> tF=read.csv(paste(dd,'vignette_file_list_subset.csv', sep = '/'))
> head(tF)
```

```
  transectNumber                          filename
1              1 L0055-D20030115-T171028-EK60.raw
2              1 L0055-D20030115-T182914-EK60.raw
3              2 L0056-D20030118-T000714-EK60.raw
4              3 L0056-D20030118-T131501-EK60.raw
5              3 L0056-D20030118-T143357-EK60.raw
6              3 L0056-D20030118-T155252-EK60.raw
```

```
> tF$filename <- paste(dd,'raw',tF$filename,sep='/')
> uniqueTransect <- unique(tF$transectNumber)
```

3 Add seabed line

Automatic seabed detection performed poorly so the seabed line was manually edited and here we overwrite an existing editable line and replace it with lines saved in the lines directory of the example data.

4 Schools detection

We are now ready to start schools detection. First, we get the ful path and name of all the EV files:

```
> fnEVVec <- list.files(dd,full.name=TRUE,pattern=paste("(", "transect", ").*\\.ev$", sep =
```

Schools detection is run on the 120 kHz 7x7 convolution variable separately for each transect. Using a loop, a single transect is opened, processed and closed before moving on to the next transect. For each transect, the regions definitions and Sv values for each aggregation detected are exported.

```
>   for(i in 1:length(fnEVVec)){
+     EVLog <- NULL
+     message("Processing ",fnEVVec[i])
+     opens <- EVOpenFile(EVAppObj, fileName = fnEVVec[i])
+     EVFile <- opens$EVFile
+     EVLog <- c(EVLog,opens$msg)
+
+     schDet<-EVSchoolsDetect(EVFile = EVFile,
+                             acoVarName = '120 7x7 convolution',
+                             outputRegionClassName = 'aggregations',
+                             deleteExistingRegions = TRUE,
+                             distanceMode = "GPS distance",
+                             maximumHorizontalLink = 15,#m
+                             maximumVerticalLink = 5,#m
+                             minimumCandidateHeight = 1,#m
+                             minimumCandidateLength = 10,#m
+                             minimumSchoolHeight = 2,#m
+                             minimumSchoolLength = 15, #m
+                             dataThreshold = -70)
+     EVLog = c(EVLog, schDet$msg)
+     EVLog = c(EVLog, EVSaveFile(EVFile)$msg)
+
+     #if aggregations were detected, export the data
+     if (schDet$nbrOfDetectedschools > 0) {
+
+     #export region definitions all aggregations
+     regionClass <- EVRegionClassFinder(EVFile, "aggregations")$regionClass
+     EVExportRegionDefByClass(regionClass, paste(dd, "/exported aggregations/aggregation_re
+
+     #export Sv data for 38kHz and 120kHz all aggregations by region
+     EVIntegrationByRegionsExport(EVFile = EVFile, acoVarName = "120 seabed and surface exc
+
+         EVIntegrationByRegionsExport(EVFile = EVFile, acoVarName = "38 seabed and surface
+
+     }
+
+     #close the file
+     EVCloseFile(EVFile = EVFile)
+   }
```

The exported aggregations are now analysed in R. Firstly, aggregations are subsetted to only include krill using the [1.04, 14.75]dB difference window. Krill aggregations are then clustered using the texxtClusterSim library. These steps are run separately for each transect. It is possible to aggregate all krill aggregations data into one .csv file and run the cluster analysis on the entire survey at once, however this is not demonstrated here.

```
> library(clusterSim)
>   for(i in 1:length(fnEVVec)){
+
+ #read in data files - the results of integration by regions for each aggregation
+
+ f038 <- read.csv(paste("C:/Users/Lisa/Desktop/KAOS/exported aggregations/38_aggregation_sv
+ f120 <- read.csv(paste("C:/Users/Lisa/Desktop/KAOS/exported aggregations/120_aggregation_s
+ Sv038 <- cbind.data.frame(Region_name = f038$Region_name, Sv038 = f038$Sv_mean)
+
+ #merge Sv_mean from 38 to 120 kHz data
+ ag <- merge(f120, Sv038, by = 'Region_name')
+
+ #remove NA values
+ ag$Sv_mean[ag$Sv_mean == -999] <- NA
+ ag$ag$Sv038[ag$Sv038 == -999] <- NA
+
+ #calculate dB difference and isolate krill swarms
+ ag$dBdiff <- ag$Sv_mean - ag$Sv038
+ swarms <- subset(as.matrix(ag), ag$dBdiff > 1.02 & ag$dBdiff < 14.75)
+
+ #if no swarms are found, move to the next transect, otherwise run a cluster analysis
+ if (nrow(swarms) == 0) {
+   message("No swarms within difference window of 1.02 - 14.75dB found")
+ } else {
+
+ #select swarm metrics for PCA - there are loads more we could chose, these are just exampl
+ swarms <- swarms[, c("Sv_mean","Sv_max","Sv_min","Corrected_length","Height_mean",
+   "Depth_mean","Corrected_thickness","Corrected_perimeter",
+   "Corrected_area","Image_compactness",
+   "Corrected_mean_amplitude","Coefficient_of_variation",
+   "Horizontal_roughness_coefficient",
+   "Vertical_roughness_coefficient")]
+
+ swarms <- apply(swarms, 2, as.numeric)
+
+ #scale the data
+ scaleSwarm <- scale(swarms)
+
+ #determine number of clusters in scaled krill swarm data using the gap-stat. see:
+ #Tibshirani, R., Walther, G., Hastie, T. (2001), Estimating the number of clusters in a da
+
+ #the code that follows is lifted from the
+ #index.Gap function in clusterSim:
+ # nc - number_of_clusters
+
+ min_nc <- 1
```

```
+ max_nc <- 10
+ if (nrow(swarms) < 10) {
+   max_nc <- nrow(swarms) - 2
+ }
+ min    <- 0
+ clopt  <- NULL
+
+ res <- array(0, c(max_nc - min_nc + 1, 2))
+ res[, 1] <- min_nc:max_nc
+ found <- FALSE
+
+ for (nc in min_nc:max_nc){
+   cl1 <- pam(scaleSwarm, nc, diss = FALSE)
+   cl2 <- pam(scaleSwarm, nc + 1, diss = FALSE)
+   clall <- cbind(cl1$clustering, cl2$clustering)
+   gap <- index.Gap(scaleSwarm, clall, B = 20, method = "pam")
+   res[nc - min_nc + 1, 2] <- diffu <- gap$diffu
+   if ((res[nc-min_nc + 1, 2] >= 0) && (!found)){
+     nc1 <- nc
+     min <- diffu
+     clopt <- cl1$cluster
+     found <- TRUE
+   }
+ }
+ if (found){
+   print(paste("Minimal number of clusters where diffu >= 0 is", nc1, "for diffu = ", round
+ }else{
+   print(paste("Transect", i, "I have not found clustering with diffu>=0", quote = FALSE))
+ }
+ plot(res, type = "p", pch = 0, xlab = "Number of clusters", ylab = "diffu", xaxt = "n")
+ abline(h = 0, untf = FALSE)
+ axis(1, c(min_nc:max_nc))
+ title(paste("Clustering for transect", i, ""))
+
+ swarms <- as.data.frame(swarms)
+ swarms$type <-  c(pam(scaleSwarm, nc1, diss = FALSE)$clustering)
+
+ #print a summary table of the number of swarms assigned to each type
+ t <- table(swarms$type) #krill swarm types determined by PAM)
+ print(t)
+
+ #the following code can print example results summary for depth, height, length by type, h
+ #lapply(split(swarms[, c("Depth_mean" ,'Height_mean','Corrected_length')], swarms$type), s
+ }
+
+ message(paste("Finished classifying aggregations for transect", i))
```

```
+ }

[1] Minimal number of clusters where diffu >= 0 is 9 for diffu =  0.0243

 1  2  3  4  5  6  7  8  9
 8  4  7  1 11  1  3  1  1
[1] Minimal number of clusters where diffu >= 0 is 6 for diffu =  0.0144

 1  2  3  4  5  6
17 46 23  8  8  3
[1] Minimal number of clusters where diffu >= 0 is 3 for diffu =  0.0266

 1  2  3
28  9 27
[1] Minimal number of clusters where diffu >= 0 is 4 for diffu =  0.0113

1 2 3 4
4 2 1 1
[1] Minimal number of clusters where diffu >= 0 is 7 for diffu =  0.0667

1 2 3 4 5 6 7
1 1 1 4 2 1 4
```

This vignette has demonstrated an automated method for performing schools detection in texxtEchoview and then using texxtR to run classification and cluster analysis on the detected aggregations.