

# Whale Wave: Habitat Use Data Prep

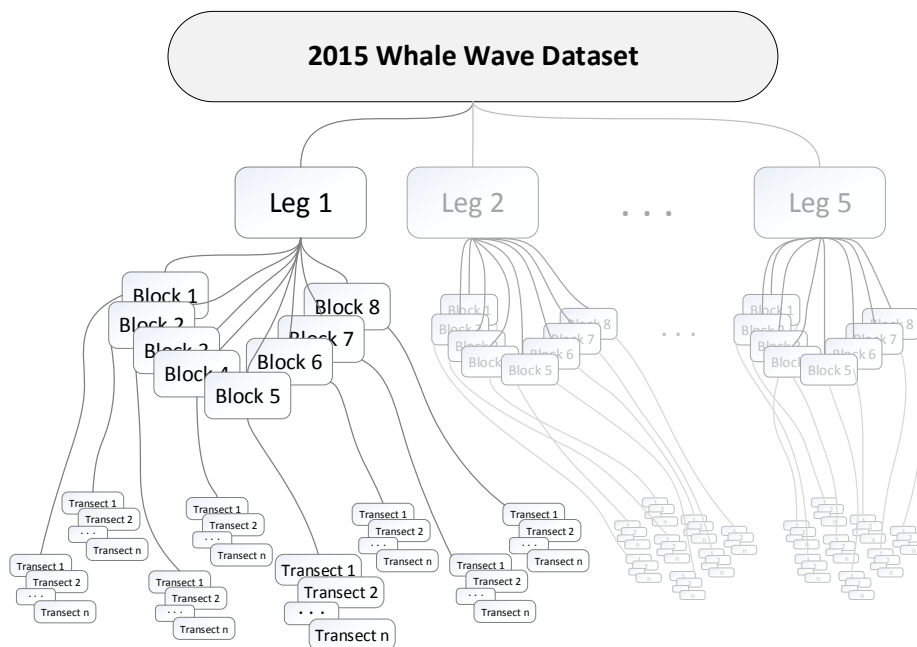
February 27, 2016

## Contents

<b>1</b>	<b>Data collection</b>	<b>2</b>
<b>2</b>	<b>Prepare backscatter transects</b>	<b>2</b>
2.1	Decide upon backscatter transect length . . . . .	2
2.2	Bin backscatter into transects . . . . .	2
2.3	Create list of summary data for transect . . . . .	4
<b>3</b>	<b>Calculate backscatter metrics</b>	<b>6</b>
<b>4</b>	<b>Environmental variables</b>	<b>8</b>
4.1	Calculate SST for each transect . . . . .	8
4.2	Calculate distance to nearest inlet . . . . .	10
4.3	Calculate distance from Ocean . . . . .	11
<b>5</b>	<b>Count Humpbacks</b>	<b>13</b>
<b>6</b>	<b>Count Effort (km)</b>	<b>15</b>
<b>7</b>	<b>Combine lists into final data file</b>	<b>17</b>
<b>8</b>	<b>Visualize</b>	<b>20</b>
8.1	Profile view of unbinned data . . . . .	20
8.2	Map view of transect bins . . . . .	21

# 1 Data collection

5 monthly surveys, summer 2015.



## 2 Prepare backscatter transects

### 2.1 Decide upon backscatter transect length

Investigating autocorrelation in the backscatter data. The following RnW file explains my process, and includes paths to all files and data used to do so:

```
> # setwd(paste(wd,"Dropbox/Bangarang Docs/R Bangarang/echo-process/bsm/scale",sep=""))
> # BSM-scale.RnW
```

### 2.2 Bin backscatter into transects

I chose a `bin.size = 5km`. I then ran the following function, which creates a folder of echo .csv's, named "5", in "Dropbox/Bangarang Docs/Data/2015/ECHO15/TRN-packaged-bins/". The function draws on echo sounder files from this folder path: "Dropbox/Bangarang Docs/Data/2015/ECHO15/TRN-packaged-csv".

```
> setwd(paste(wd,"Dropbox/Bangarang Docs/R Bangarang/2014/echo",sep=""))
> source("echo-trn-bins.R")
> #echo.bin(bin.size=5,min.error=0,max.error=20)
> echo.bin
```

```
function (bin.size, min.error = 0, max.error = 20, min.rows = 150,
  wd = "/Users/eric.keen/")
```

```

{
  writedir <- paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECH015/TRN-packaged-bins/",
    bin.size, sep = "")
  readdir <- paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECH015/TRN-packaged-filter-db-csv",
    sep = "")
  library(PBSmapping)
  data(nepacLLhigh)
  setwd(paste(wd, "Dropbox/Bangarang Docs/R Bangarang/2014/plotkfs",
    sep = ""))
  source("plotkfs.R")
  library(swfsMisc)
  setwd(readdir)
  lf <- list.files()
  bindcount <- vector()
  for (i in 1:length(lf)) {
    filename <- lf[i]
    setwd(readdir)
    echo <- read.csv(filename, stringsAsFactors = FALSE)
    echo$X <- as.numeric(echo$X)
    echo$Y <- as.numeric(echo$Y)
    echo <- echo[!is.na(echo$X) & !is.na(echo$Y), ]
    lat1 <- echo$Y[1:(nrow(echo) - 1)]
    lat2 <- echo$Y[2:nrow(echo)]
    lon1 <- echo$X[1:(nrow(echo) - 1)]
    lon2 <- echo$X[2:nrow(echo)]
    jumps <- data.frame(lat1, lon1, lat2, lon2)
    d <- apply(jumps, 1, function(row) {
      di <- as.numeric(row)
      dist <- distance(di[1], di[2], di[3], di[4], units = "nm",
        method = "haversine")
      return(dist)
    })
    d <- d * 1.85
    cumd <- cumsum(d)
    totd <- sum(d)
    dbreaks <- seq(0, totd, by = bin.size)
    n.bin <- length(dbreaks)
    starts <- ends <- vector()
    if (n.bin <= 1) {
      starts <- 1
      ends <- nrow(echo)
    }
    else {
      starts <- vector()
      for (bi in dbreaks) {
        starti <- which.min(abs(bi - cumd))
        starts <- c(starts, starti)
      }
      ends <- c(starts[2:length(starts)], nrow(echo))
    }
    cues <- data.frame(starts, ends)
    for (j in 1:nrow(cues)) {
      subrows <- cues$starts[j]:cues$ends[j]
      bind <- sum(d[subrows])
    }
  }
}

```

```

subecho <- echo[subrows, ]
fnsuffix <- subecho$date[1]
fnsuffix <- substr(fnsuffix, 12, 19)
fnsuffix <- gsub(":", "", fnsuffix)
binname <- paste0(gsub(".csv", "", filename), "-",
  fnsuffix, ".csv")
if (is.finite(bind) & bind >= min.error & bind <=
  max.error & nrow(subecho) >= min.rows) {
  setwd(writedir)
  write.csv(subecho, file = binname, quote = FALSE,
    row.names = FALSE)
  bindcount <- c(bindcount, bind)
  print(paste0("Bin ", j, ": ", binname, " (nrow=",
    nrow(subecho), ", L=", round(bind, digits = 3),
    ") stored away!!!"))
}
}
}
}
}
> setwd(curdir)

```

## 2.3 Create list of summary data for transect

The next few steps rely on functions in this R file:

```

> setwd(paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECHO15/TRN-bins-docs", sep=""))
> source("trn-bins-list.R")
> setwd(curdir)

```

To summarize data for the backscatter transects, I ran the following function. This function makes no changes to the folder of echo .csv's. The function takes the .csv files in the correct folder (found by the bin.size I feed the function) and determines the trackline length (km) of the file, the centroid coordinates, and the distance of the centroid from the ocean (using the set of oceandistance functions). It outputs this list as a .csv named "trn-bins-2015-5.csv" in "Data/2015/ECHO15/TRN-packaged-bins/list". The idea there is that if I try a bunch of different backscatter transect lengths, all the summary lists for all lengths will be in the same folder, as will all HW counts for all lengths, etc.

```

> #bin.list(bin.size="5", ODI=TRUE)
> bin.list

function (bin.size, ODI = TRUE, wd = "/Users/eric.keen/")
{
  setwd(paste(wd, "Dropbox/Bangarang Docs/R Bangarang/2014/plotkfs",
    sep = ""))
  source("plotkfs.R")
  library(swfscMisc)
  readdir <- paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECHO15/TRN-packaged-bins/",
    bin.size, sep = "")
  setwd(readdir)
  filename <- list.files()
  year <- as.numeric(substr(filename, 5, 8))
  leg <- as.numeric(substr(filename, 10, 10))
  block <- substr(filename, 12, 14)
  day <- substr(filename, 16, 23)
  jul <- as.numeric(julian(as.Date(day, format = "%Y%m%d"),

```

```

    origin = as.Date("2015-01-01"))
binlist <- data.frame(filename, year, day, jul, leg, block)
X <- Y <- pings.lo <- pings.hi <- dist <- rep(NA, times = nrow(binlist))
for (f in 1:length(filename)) {
  setwd(readdir)
  echo <- read.csv(filename[f], stringsAsFactors = FALSE)
  if (nrow(echo) > 5) {
    center <- nrow(echo)/2
    X[f] <- echo$X[center]
    Y[f] <- echo$Y[center]
    pings.lo[f] <- nrow(echo[echo$freq == 1, ])
    pings.hi[f] <- nrow(echo[echo$freq == 2, ])
    lat1 <- echo$Y[1:(nrow(echo) - 1)]
    lat2 <- echo$Y[2:nrow(echo)]
    lon1 <- echo$X[1:(nrow(echo) - 1)]
    lon2 <- echo$X[2:nrow(echo)]
    jumps <- data.frame(lat1, lon1, lat2, lon2)
    d <- apply(jumps, 1, function(row) {
      di <- as.numeric(row)
      dist <- distance(di[1], di[2], di[3], di[4],
        units = "nm", method = "haversine")
      return(dist)
    })
    d <- d * 1.85
    dist[f] <- sum(d)
  }
  print(filename[f])
}
binlist$X <- X
binlist$Y <- Y
binlist$pings.lo <- pings.lo
binlist$pings.hi <- pings.hi
binlist$km <- dist
binlist$size <- rep(bin.size, times = nrow(binlist))
if (ODI) {
  setwd(paste(wd, "Dropbox/Bangarang Docs/R Bangarang/oceandistance",
    sep = ""))
  source("oceandistance.R")
  access.pts <- list(c(-128.95, 53.56))
  access.names <- c("Dewdney", "Middle Caamano", "S Middle Caamano",
    "W Rennison")
  od <- rep(NA, times = nrow(binlist))
  for (i in 1:nrow(binlist)) {
    x1 <- as.numeric(binlist$X[i])
    y1 <- as.numeric(binlist$Y[i])
    if (!any(c(is.na(x1), is.na(y1)))) {
      odops <- vector()
      for (j in 1:length(access.pts)) {
        print(paste0("Measuring echo bin ", binlist$filename[i],
          " to access pt at ", access.names[j], "..."))
        ap <- access.pts[[j]]
        x2 <- ap[1]
        y2 <- ap[2]
        if (j == 1) {

```

```

        topplot <- TRUE
      }
      else {
        topplot <- FALSE
      }
      odj <- ocean.distance(x1, y1, x2, y2, plot.route = topplot,
        wd = wd)
      odops <- c(odops, odj)
    }
    odops <- odops[!is.na(odops)]
    odi <- which.min(odops)
    od[i] <- round(odops[odi], digits = 4)
    print(paste0("Minimum ocean distance for echo bin ",
      binlist$filename[i], " = ", round(odops[odi],
        digits = 3), "km!!!"))
  }
}
}
else {
  od <- rep(NA, times = nrow(binlist))
}
binlist$OD <- od
setwd(paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECHO15/TRN-packaged-bins/list",
  sep = ""))
newname <- paste0("trn-bins-2015-", bin.size, ".csv")
write.csv(binlist, file = newname, quote = FALSE, row.names = FALSE)
print(paste0("Done creating echo bin list for bin size: ",
  bin.size, "km!!!!"))
}

```

### 3 Calculate backscatter metrics

To calculate BSMs, I had to source an additional file of functions:

```

> setwd(paste(wd, "Dropbox/Bangarang Docs/R Bangarang/echo-process/bsm", sep = ""))
> source("bsm-bin-15.R")
> setwd(curdir)

```

This file actually sources another file of basic building blocks for the BSM functions:

```

> setwd(paste(wd, "Dropbox/Bangarang Docs/R Bangarang/echo-process/bsm", sep = ""))
> source("bsm-basic-15.R")
> setwd(curdir)

```

The "bsm-bin-15.R" file offers two approaches to calculating BSMs. The first, which is what I ended up using, takes the .echo csv's in the bin.size folder and calculates the final, formal BSMs for each file, saving the results to a dataframe then to a .csv (of the same name, as above, but in this folder: "Data/2015/ECHO15/TRN-packaged-bins/bsm"). This approach relies on the following function. The Boolean "neighbors" option gives you the choice of calculating nearest Neighbor/Dispersion metrics, which are extremely time consuming (10+ hours of processing time for 2015). I ended up not using the neighbor metric anyways (it was too abstract and probably not relevant), so I usually run this function with "neighbors=FALSE".

```

> #bin.bsm(bin.size="5",neighbors=FALSE)
> bin.bsm

```

```

function (bin.size, wd = "/Users/eric.keen/")
{
  library(swfscMisc)
  setwd(paste(wd, "Dropbox/Bangarang Docs/R Bangarang/echo-process/bsm",
    sep = ""))
  source("bsm-basic-15.R")
  bindir <- paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECH015/TRN-packaged-bins/",
    bin.size, sep = "")
  setwd(bindir)
  bins <- list.files()
  bin.bsm.proc <- function(bin) {
    binnumbers <- bsm.bin.numbers(bin)
    pings <- binnumbers[[1]]
    all <- binnumbers[[2]]
    biota <- all[all > 0]
    water <- binnumbers[[3]]
    totpx <- binnumbers[[4]]
    sumpx <- binnumbers[[5]]
    I <- bsm.bin.integral(sumpx, pings)
    M <- bsm.bin.mnpx(sumpx, totpx)
    pb <- bsm.bin.pb(all)
    D <- vert.dispersion(bin)
    C <- patchiness(bin, quantile.range = 0.5, window.hor = 10,
      window.ver = 20, s = 1, random = FALSE, toplot = FALSE)
    results <- data.frame(I, M, C, D)
    return(results)
  }
  bsm.mr <- data.frame()
  for (f in 1:length(bins)) {
    filename <- bins[f]
    setwd(bindir)
    bin <- read.csv(filename, stringsAsFactors = FALSE)
    hi <- bin[bin$freq == 2, ]
    lo <- bin[bin$freq == 1, ]
    hibsms <- bin.bsm.proc(hi)
    names(hibsms) <- paste0(names(hibsms), ".hi")
    lobsms <- bin.bsm.proc(lo)
    names(lobsms) <- paste0(names(lobsms), ".lo")
    bsmf <- cbind(filename, hibsms, lobsms)
    bsm.mr <- rbind(bsm.mr, bsmf)
    print(bsmf)
  }
  setwd(paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECH015/TRN-packaged-bins/bsm",
    sep = ""))
  newname <- paste0("trn-bins-2015-", bin.size, ".csv")
  write.csv(bsm.mr, file = newname, quote = FALSE, row.names = FALSE)
  print(paste0("Done calculating BSMs for bin size: ", bin.size,
    "km!!!!"))
}

```

## 4 Environmental variables

### 4.1 Calculate SST for each transect

To calculate mean SST for each transect, I ran this function, which outputs a .csv of the same name as above but in a different folder: "Data/2015/ECHO15/TRN-packaged-bins/sst".

```
> #bin.sst(bin.size="5")
> bin.sst

function (bin.size, wd = "/Users/eric.keen/")
{
  setwd(paste(wd, "Dropbox/Bangarang Docs/R Bangarang/2014",
    sep = ""))
  source("B14_Functions.R")
  setwd(paste(wd, "Dropbox/Bangarang Docs/R Bangarang/2014/plotkfs",
    sep = ""))
  source("plotkfs.R")
  setwd(paste(wd, "Dropbox/Bangarang Docs/R Bangarang/oceandistance",
    sep = ""))
  source("oceandistance.R")
  library(swfscMisc)
  setwd(paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECHO15/TRN-packaged-bins/list",
    sep = ""))
  binlist <- read.csv(paste0("trn-bins-2015-", bin.size, ".csv"),
    stringsAsFactors = FALSE)
  SST <- SSS <- rep(NA, times = nrow(binlist))
  for (i in 1:nrow(binlist)) {
    day <- binlist$day[i]
    dist <- binlist$km[i]
    legi <- binlist$leg[i]
    tstart <- substr(binlist$filename[i], 25, 30)
    tstart <- paste0(substr(day, 1, 4), "-", substr(day,
      5, 6), "-", substr(day, 7, 8), " ", substr(tstart,
      1, 2), ":", substr(tstart, 3, 4), ":", substr(tstart,
      3, 4))
    tstart <- strptime(tstart)
    rub <- datefilter14(day, day, yr = "2015", wd = wd)
    istart <- which.min(abs(difftime(tstart, rub$date, units = "secs")))
    rub <- rub[istart:nrow(rub), ]
    if (nrow(rub) > 1) {
      lat1 <- rub$lat[1:(nrow(rub) - 1)]
      lat2 <- rub$lat[2:nrow(rub)]
      lon1 <- rub$long[1:(nrow(rub) - 1)]
      lon2 <- rub$long[2:nrow(rub)]
      jumps <- data.frame(lat1, lon1, lat2, lon2)
      d <- apply(jumps, 1, function(row) {
        di <- as.numeric(row)
        dist <- distance(di[1], di[2], di[3], di[4],
          units = "nm", method = "haversine")
        return(dist)
      })
      d <- d * 1.85
      rub$dist <- c(d, NA)
    }
  }
}
```

```

    rub <- rub[!is.na(rub$dist), ]
    rub$cumsum <- cumsum(rub$dist)
    if (nrow(rub) > 1) {
      iend <- which(rub$cumsum >= dist)[1]
      if (dist >= rub$cumsum[nrow(rub)]) {
        iend <- nrow(rub)
      }
      rub <- rub[1:iend, ]
      ssti <- mean(as.numeric(rub$tsg.temp), na.rm = TRUE)
      sssi <- mean(as.numeric(rub$tsg.sal), na.rm = TRUE)
    }
  }
  if (is.na(ssti) | is.na(sssi)) {
    gapfill <- tsg.fill(X = binlist$X[i], Y = binlist$Y[i],
      leg = legi, wd = wd)
    ssti <- gapfill[1]
    sssi <- gapfill[2]
  }
  SST[i] <- ssti
  SSS[i] <- sssi
  print(paste0("SST and SSS: Bin size ", bin.size, " - ",
    binlist$filename[i]))
}
binlist$sst <- SST
binlist$sss <- SSS
setwd(paste0(wd, "Dropbox/Bangarang Docs/Data/2015/ECH015/TRN-packaged-bins/sst",
  sep = ""))
newname <- paste0("trn-bins-2015-", bin.size, ".csv")
write.csv(binlist, file = newname, quote = FALSE, row.names = FALSE)
print(paste0("Done creating SST means for bin size: ", bin.size,
  "km!!!!"))
}

```

This function relies on a child function to fill in TSG data gaps using interpolated data:

```

> tsg.fill

function (X, Y, leg, wd = "/Users/eric.keen/")
{
  library(swfscMisc)
  writefile <- paste0(wd, "Dropbox/Bangarang Docs/Data/", yr,
    "/stations/interp-csv")
  setwd(writefile)
  sfilename <- paste0("2015-", leg, "-TSG-sss.csv")
  tfilename <- paste0("2015-", leg, "-TSG-sst.csv")
  sst <- read.csv(tfilename, stringsAsFactors = FALSE)
  sss <- read.csv(sfilename, stringsAsFactors = FALSE)
  Xmin <- X - 0.01
  Xmax <- X + 0.01
  Ymin <- Y - 0.01
  Ymax <- Y + 0.01
  sst <- sst[which(sst$X <= Xmax & sst$X >= Xmin & sst$Y <=
    Ymax & sst$Y >= Ymin), ]
  sss <- sss[which(sss$X <= Xmax & sss$X >= Xmin & sss$Y <=
    Ymax & sss$Y >= Ymin), ]
}

```

```

if (nrow(sst) == 0 | nrow(sss) == 0) {
  SST <- SSS <- NA
}
else {
  d <- vector()
  for (j in 1:nrow(sst)) {
    lat <- sst$Y[j]
    lon <- sst$X[j]
    di <- distance(lat1 = Y, lon1 = X, lat2 = lat, lon2 = lon,
      method = "Vincenty", units = "km")
    d <- c(d, di)
  }
  mind <- which.min(d)
  SST <- sst$Z[mind]
  SSS <- sss$Z[mind]
}
print("Missing TSG reading gap-filled by interpolated dataset!")
return(c(SST, SSS))
}

```

## 4.2 Calculate distance to nearest inlet

```

> #inletD(bin.size="5")
> inletD

function (bin.size, wd = "/Users/eric.keen/")
{
  setwd(paste(wd, "Dropbox/Bangarang Docs/R Bangarang/2014/plotkfs",
    sep = ""))
  source("plotkfs.R")
  library(swfscMisc)
  setwd(paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECHO15/TRN-packaged-bins/list",
    sep = ""))
  binlist <- read.csv(paste0("trn-bins-2015-", bin.size, ".csv"),
    stringsAsFactors = FALSE)
  setwd(paste(wd, "Dropbox/Bangarang Docs/R Bangarang/oceandistance",
    sep = ""))
  source("oceandistance.R")
  access.pts <- list(c(-129.152526855469, 52.8832199310379),
    c(-129.057769775391, 53.2142569416679), c(-128.862762451172,
    53.4733352741244), c(-128.95, 53.56))
  access.names <- c("Surf", "Cornwall", "Bishop", "Devastation")
  od <- rep(NA, times = nrow(binlist))
  for (i in 1:nrow(binlist)) {
    x1 <- as.numeric(binlist$X[i])
    y1 <- as.numeric(binlist$Y[i])
    if (!any(c(is.na(x1), is.na(y1)))) {
      odops <- vector()
      for (j in 1:length(access.pts)) {
        print(paste0("Measuring echo bin ", binlist$filename[i],
          " to access pt at ", access.names[j], "..."))
        ap <- access.pts[[j]]
        x2 <- ap[1]
        y2 <- ap[2]

```

```

        if (j == 1) {
            topplot <- TRUE
        }
        else {
            topplot <- FALSE
        }
        odj <- ocean.distance(x1, y1, x2, y2, plot.route = topplot,
            wd = wd)
        odops <- c(odops, odj)
    }
    odops <- odops[!is.na(odops)]
    odi <- which.min(odops)
    od[i] <- round(odops[odi], digits = 4)
    print(paste0("Minimum ocean distance for echo bin ",
        binlist$filename[i], " = ", round(odops[odi],
            digits = 3), "km!!!"))
}
}
binlist$inlet <- od
setwd(paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECH015/TRN-packaged-bins/inlet",
    sep = ""))
newname <- paste0("trn-bins-2015-", bin.size, ".csv")
write.csv(binlist, file = newname, quote = FALSE, row.names = FALSE)
print(paste0("Done creating inlet distance list for bin size: ",
    bin.size, "km!!!!"))
}

```

### 4.3 Calculate distance from Ocean

```

> #bin.ocean(bin.size="5")
> bin.ocean

function (bin.size, wd = "/Users/eric.keen/")
{
    setwd(paste(wd, "Dropbox/Bangarang Docs/R Bangarang/2014/plotkfs",
        sep = ""))
    source("plotkfs.R")
    library(swfscMisc)
    readdir <- paste0(wd, "Dropbox/Bangarang Docs/Data/", yr,
        "/stations/interp-csv")
    writedir <- paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECH015/TRN-packaged-bins/ocean",
        sep = "")
    setwd(paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECH015/TRN-packaged-bins/list",
        sep = ""))
    binlist <- read.csv(paste0("trn-bins-2015-", bin.size, ".csv"),
        stringsAsFactors = FALSE)
    chla <- secchi <- rep(NA, times = nrow(binlist))
    for (i in 1:nrow(binlist)) {
        X <- as.numeric(binlist$X[i])
        Y <- as.numeric(binlist$Y[i])
        legi <- as.numeric(binlist$leg[i])
        chlai <- secchii <- NA
        if (legi != 3) {
            setwd(readdir)

```

```

chladf <- read.csv(paste0("2015-", legi, "-Chla-chl.csv"),
  stringsAsFactors = FALSE)
secchidf <- read.csv(paste0("2015-", legi, "-Secchi-secchi.csv"),
  stringsAsFactors = FALSE)
Xmin <- X - 0.01
Xmax <- X + 0.01
Ymin <- Y - 0.01
Ymax <- Y + 0.01
chladf <- chladf[which(chladf$X <= Xmax & chladf$X >=
  Xmin & chladf$Y <= Ymax & chladf$Y >= Ymin),
]
secchidf <- secchidf[which(secchidf$X <= Xmax & secchidf$X >=
  Xmin & secchidf$Y <= Ymax & secchidf$Y >= Ymin),
]
if (nrow(chladf) != 0 & nrow(secchidf) != 0) {
  d <- vector()
  for (j in 1:nrow(chladf)) {
    lat <- chladf$Y[j]
    lon <- chladf$X[j]
    di <- distance(lat1 = Y, lon1 = X, lat2 = lat,
      lon2 = lon, method = "Vincenty", units = "km")
    d <- c(d, di)
  }
  mind <- which.min(d)
  chlai <- chladf$Z[mind]
  d <- vector()
  for (j in 1:nrow(secchidf)) {
    lat <- secchidf$Y[j]
    lon <- secchidf$X[j]
    di <- distance(lat1 = Y, lon1 = X, lat2 = lat,
      lon2 = lon, method = "Vincenty", units = "km")
    d <- c(d, di)
  }
  mind <- which.min(d)
  secchii <- secchidf$Z[mind]
  chla[i] <- chlai
  secchi[i] <- secchii
}
}
print(paste0("Leg ", legi, ": Chla = ", chlai, " | Euphotic depth = ",
  secchii))
}
binlist$chla <- chla
binlist$secchi <- secchi
setwd(paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECH015/TRN-packaged-bins/ocean",
  sep = ""))
newname <- paste0("trn-bins-2015-", bin.size, ".csv")
write.csv(binlist, file = newname, quote = FALSE, row.names = FALSE)
print(paste0("Done creating oceanography list for bin size: ",
  bin.size, "km!!!!"))
}

```

## 5 Count Humpbacks

I used the following function to count humpbacks in a given radius around the centroids of each transect. I would often "batch-count" multiple radii at once, to save time, by supplying a vector of radii for the radius.vec input. The function saves a file of the same name as above but in a different folder: "Data/2015/ECHO15/TRN-packaged-bins/hw". This function has a nested function that actually references processed sightings summaries from each day of effort in 2015. This function filters out all whales that did not occur on the same day as the transect file and that were not seen while on transect effort (so only effort 1 and 2 whales were kept). It then draws a square around the centroid with a width and height of 2\*radius, to filter out whales further. It then calculates the Euclidean distance from whale to centroid to make sure that it is actually within the radius.

```
> radii <- 2.5 # seq(1,7,by=.25)
> #bin.hw(bin.size="5",radius.vec=radii)
> bin.hw

function (bin.size, radius.vec = 5, wd = "/Users/eric.keen/")
{
  setwd(paste(wd, "Dropbox/Bangarang Docs/R Bangarang/2014/plotkfs",
    sep = ""))
  source("plotkfs.R")
  setwd(paste(wd, "Dropbox/Bangarang Docs/R Bangarang/oceandistance",
    sep = ""))
  source("oceandistance.R")
  library(swfsMisc)
  setwd(paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECHO15/TRN-packaged-bins/list",
    sep = ""))
  binlist <- read.csv(paste0("trn-bins-2015-", bin.size, ".csv"),
    stringsAsFactors = FALSE)
  filename <- binlist$filename
  setwd(paste(wd, "Dropbox/Bangarang Docs/Data/2015/SIW15/SIWproc",
    sep = ""))
  lf <- list.files()
  siw <- data.frame()
  for (i in 1:length(lf)) {
    siw15 <- read.csv(lf[i], stringsAsFactors = FALSE)
    siw <- rbind(siw, siw15)
  }
  siw <- siw[!is.na(as.numeric(siw$X)) & !is.na(as.numeric(siw$Y)),
    ]
  whale.count <- function(X, Y, block, radius.km = c(2), leg,
    spp = "HW", eff = c(1, 2), siw) {
    siw <- siw[siw$circ == leg & siw$eff %in% eff & siw$sp ==
      spp & siw$block == block, ]
    lat1 <- as.numeric(Y)
    lon1 <- as.numeric(X)
    radius <- max(radius.km)
    right <- as.numeric(destination(Y, X, brng = 90, distance = radius,
      units = "km", type = "Vincenty")[2])
    left <- as.numeric(destination(Y, X, brng = 270, distance = radius,
      units = "km", type = "Vincenty")[2])
    top <- as.numeric(destination(Y, X, brng = 0, distance = radius,
      units = "km", type = "Vincenty")[1])
    bot <- as.numeric(destination(Y, X, brng = 180, distance = radius,
      units = "km", type = "Vincenty")[1])
```

```

subsiw <- siw[as.numeric(siw$X) >= left, ]
subsiw <- subsiw[as.numeric(subsiw$X) <= right, ]
subsiw <- subsiw[as.numeric(subsiw$Y) <= top, ]
subsiw <- subsiw[as.numeric(subsiw$Y) >= bot, ]
if (nrow(subsiw) > 0) {
  print(paste0("Leg: ", leg, " - Calculating Euclidean distance to candidate whales..."))
  dvec <- vector()
  grpvec <- vector()
  for (j in 1:nrow(subsiw)) {
    lat2 <- as.numeric(subsiw$Y[j])
    lon2 <- as.numeric(subsiw$X[j])
    grpj <- subsiw$grp.best[j]
    if (is.na(grpj)) {
      grpj <- 1
    }
    if (grpj > 8) {
      grpj <- 8
    }
    odj <- distance(lat1, lon1, lat2, lon2, units = "km",
      method = "Vincenty")
    dvec <- c(dvec, odj)
    grpvec <- c(grpvec, grpj)
  }
  hwcounts <- vector()
  for (k in 1:length(radius.km)) {
    din <- which(dvec <= radius.km[k])
    nhwi <- sum(grpvec[din])
    hwcounts <- c(hwcounts, nhwi)
  }
}
else {
  hwcounts <- rep(0, times = length(radius.km))
}
print(paste0("Leg ", leg, " whales within radius.vec of echo center: ",
  paste(hwcounts, collapse = "/")))
return(hwcounts)
}
for (r in 1:length(radius.vec)) {
  colname <- paste0("hw", radius.vec[r])
  tempvec <- rep(NA, times = nrow(binlist))
  for (i in 1:nrow(binlist)) {
    Xi <- binlist$X[i]
    Yi <- binlist$Y[i]
    blocki <- binlist$block[i]
    legi <- binlist$leg[i]
    if (!any(c(is.na(Xi), is.na(Yi)))) {
      hwi <- whale.count(Xi, Yi, block = blocki, radius.km = radius.vec[r],
        leg = legi, siw = siw)
      tempvec[i] <- hwi
    }
  }
}
binlist$newcol <- tempvec
names(binlist)[ncol(binlist)] <- colname
}

```

```

setwd(paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECH015/TRN-packaged-bins/hw",
            sep = ""))
newname <- paste0("trn-bins-2015-", bin.size, ".csv")
write.csv(binlist, file = newname, quote = FALSE, row.names = FALSE)
print(paste0("Done creating HW counts for bin size: ", bin.size,
            "km!!!!"))
}

```

## 6 Count Effort (km)

I used the following function to total up the kilometers of trackline occurring within a given radius of each transect. I used the same batch counting technique as in the humpbacks. This function involves reading in all RUB dailies from 2015, which can be time consuming. Similar to the humpback function, there is a nested function that reduces the giant RUB dataframe to only effort 1 and 2 occurring on the same day as the backscatter transect. Then it calculates the distance between rows in the subset RUB dataframe, removes all distances that are above 50m apart (such entries would have to correspond to breaks in effort and would artificially inflate km trackline if they weren't removed), and totals up the distance occurring within a square surrounding the centroid with length and width of  $2*r$ . The buffer input option was never used, but it could be used to simply increase all the radii provided by the same amount. The function outputs a .csv of the same name as above but in a different folder.

```

> radii <- 2.5 # seq(1,7,by=.25)
> #bin.eff(bin.size="5",radius.vec=radii,buffer=0)
> bin.eff

function (bin.size, radius.vec = 1:20, buffer = 0, wd = "/Users/eric.keen/")
{
  setwd(paste(wd, "Dropbox/Bangarang Docs/R Bangarang/2014/plotkfs",
            sep = ""))
  source("plotkfs.R")
  library(swfscMisc)
  setwd(paste(wd, "Dropbox/Bangarang Docs/R Bangarang/2014",
            sep = ""))
  source("B14_Functions.R")
  setwd(paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECH015/TRN-packaged-bins/list",
            sep = ""))
  binlist <- read.csv(paste0("trn-bins-2015-", bin.size, ".csv"),
                    stringsAsFactors = FALSE)
  filename <- binlist$filename
  rub <- datefilter14(datemin = 20150527, datemax = 20150924,
                    yr = "2015", wd = wd)
  rub.backup <- rub
  X <- as.numeric(rub$long)
  Y <- as.numeric(rub$lat)
  leg <- as.numeric(rub$circ)
  eff <- as.numeric(rub$eff)
  data <- data.frame(X, Y, leg, eff)
  data$block <- rub$block
  data <- data[!is.na(data$X) & !is.na(data$Y) & !is.na(data$leg) &
            !is.na(data$eff), ]
  lat1 <- data$Y[1:(nrow(data) - 1)]
  lat2 <- data$Y[2:nrow(data)]
  lon1 <- data$X[1:(nrow(data) - 1)]

```

```

lon2 <- data$X[2:nrow(data)]
distdf <- data.frame(lat1, lon1, lat2, lon2)
d <- apply(distdf, 1, function(row) {
  row <- as.numeric(row)
  distance(row[1], row[2], row[3], row[4], units = "nm",
    method = "haversine")
})
d <- d * 1.85
d <- c(d, NA)
data$d <- d
data <- data[1:(nrow(data) - 1), ]
data <- data[data$d < 0.1, ]
eff.count <- function(X, Y, radius.km = c(2), leg, eff = c(1,
  2), buffer = 0, trackline) {
  trackline <- trackline[trackline$leg == leg & trackline$eff %in%
    eff, ]
  Y <- as.numeric(Y)
  X <- as.numeric(X)
  effkm <- rep(NA, times = length(radius.km))
  for (k in 1:length(radius.km)) {
    radius <- radius.km[k] + buffer
    right <- as.numeric(destination(Y, X, brng = 90,
      distance = radius, units = "km", type = "Vincenty")[2])
    left <- as.numeric(destination(Y, X, brng = 270,
      distance = radius, units = "km", type = "Vincenty")[2])
    top <- as.numeric(destination(Y, X, brng = 0, distance = radius,
      units = "km", type = "Vincenty")[1])
    bottom <- as.numeric(destination(Y, X, brng = 180,
      distance = radius, units = "km", type = "Vincenty")[1])
    subrub <- trackline[trackline$X >= left, ]
    subrub <- subrub[subrub$X <= right, ]
    subrub <- subrub[subrub$Y <= top, ]
    subrub <- subrub[subrub$Y >= bottom, ]
    effk <- 0
    if (nrow(subrub) > 0) {
      effk <- sum(subrub$d, na.rm = TRUE)
    }
    effkm[k] <- effk
  }
  print(paste0("Leg ", leg, " effort (km) within radius ",
    radius, " of echo center: ", paste(effkm, collapse = "/")))
  return(effkm)
}
for (r in 1:length(radius.vec)) {
  colname <- paste0("ef", radius.vec[r])
  tempvec <- rep(NA, times = nrow(binlist))
  for (i in 1:nrow(binlist)) {
    Xi <- binlist$X[i]
    Yi <- binlist$Y[i]
    blocki <- binlist$block[i]
    legi <- binlist$leg[i]
    if (!any(c(is.na(Xi), is.na(Yi)))) {
      efi <- eff.count(Xi, Yi, radius.km = radius.vec[r],
        leg = legi, buffer = buffer, trackline = data)
    }
  }
}

```

```

        tempvec[i] <- efi
      }
    }
    binlist$newcol <- tempvec
    names(binlist)[ncol(binlist)] <- colname
  }
  setwd(paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECHO15/TRN-packaged-bins/eff",
    sep = ""))
  newname <- paste0("trn-bins-2015-", bin.size, ".csv")
  write.csv(binlist, file = newname, quote = FALSE, row.names = FALSE)
  print(paste0("Done creating EFF counts for bin size: ", bin.size,
    "km!!!!"))
}

```

## 7 Combine lists into final data file

This function, sourced from the "trn-bins-list.R" file with the rest of the listing functions, combines all the lists (Summary, SST, HW, EFF and BSM) into a single dataframe and saves it to a .csv (same name as above, "trn-bins-2015-5.csv") but in a different folderpath: "Data/2015/ECHO15/TRN-bins-docs". This is the final dataset. The "use.echo.eff" option gives me the option of replacing RUB effort with the calculated distance of the echosounder trackline. I did NOT use this option in creating the final dataset.

I then copied this .csv file into this Sweave RnW's working folder.

```

> #bin.combine(bin.size="5",pre=FALSE,hw=TRUE,eff=TRUE,use.echo.eff=FALSE,bsm=TRUE)
> bin.combine

function (bin.size, sst = TRUE, hw = TRUE, eff = TRUE, eff.correct = FALSE,
  bsm = TRUE, D.correct = TRUE, M.correct = TRUE, C.correct = TRUE,
  inlet = TRUE, ocean = TRUE, wd = "/Users/eric.keen/")
{
  setwd(paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECHO15/TRN-packaged-bins/list/",
    sep = ""))
  binlist <- read.csv(paste0("trn-bins-2015-", bin.size, ".csv"),
    stringsAsFactors = FALSE)
  if (hw) {
    setwd(paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECHO15/TRN-packaged-bins/hw/",
      sep = ""))
    hw <- read.csv(paste0("trn-bins-2015-", bin.size, ".csv"),
      stringsAsFactors = FALSE)
    cols <- which(substr(names(hw), 1, 2) == "hw")
    hw <- hw[, cols]
    binlist <- cbind(binlist, hw)
  }
  if (eff) {
    setwd(paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECHO15/TRN-packaged-bins/eff/",
      sep = ""))
    eff <- read.csv(paste0("trn-bins-2015-", bin.size, ".csv"),
      stringsAsFactors = FALSE)
    cols <- which(substr(names(eff), 1, 2) == "ef")
    eff <- eff[, cols]
    if (eff.correct) {
      efflo <- which(eff < 2)
      for (i in 1:length(eff)) {

```

```

        if (eff[i] < 2) {
          eff[i] <- binlist$km[i]
        }
      }
      print(paste0("Oddly low effort replaced with echo km in rows: ",
        paste(efflo, collapse = "/")))
    }
    binlist <- cbind(binlist, eff)
  }
  if (bsm) {
    setwd(paste(wd, "Dropbox/Bangarang Docs/R Bangarang/echo-process/bsm",
      sep = ""))
    source("bsm-bin-15.R")
    setwd(paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECH015/TRN-packaged-bins/bsm/",
      sep = ""))
    bsms <- read.csv(paste0("trn-bins-2015-", bin.size, ".csv"),
      stringsAsFactors = FALSE)
    bsms$filename <- NULL
    if (D.correct) {
      maxlo <- max(bsms$D.lo, na.rm = TRUE)
      maxhi <- max(bsms$D.hi, na.rm = TRUE)
      lona <- which(is.na(bsms$D.lo))
      hina <- which(is.na(bsms$D.hi))
      bsms$D.lo[lona] <- maxlo
      bsms$D.hi[hina] <- maxhi
      print(paste0("Lo freq Dispersion corrected with max value in rows: ",
        paste(lona, collapse = "/")))
      print(paste0("Hi freq Dispersion corrected with max value in rows: ",
        paste(hina, collapse = "/")))
    }
    if (M.correct) {
      lona <- which(is.na(bsms$M.lo))
      hina <- which(is.na(bsms$M.hi))
      bsms$M.lo[lona] <- 125
      bsms$M.hi[hina] <- 125
      print(paste0("Lo freq Mn Intensity corrected with 125s in rows: ",
        paste(lona, collapse = "/")))
      print(paste0("Hi freq Mn Intensity corrected with 125s in rows: ",
        paste(hina, collapse = "/")))
    }
    if (C.correct) {
      lona <- which(is.na(bsms$C.lo))
      hina <- which(is.na(bsms$C.hi))
      bsms$C.lo[lona] <- 0
      bsms$C.hi[hina] <- 0
      print(paste0("Lo freq Patchiness corrected with 0s in rows: ",
        paste(lona, collapse = "/")))
      print(paste0("Hi freq Patchiness corrected with 0s in rows: ",
        paste(hina, collapse = "/")))
    }
    binlist <- cbind(binlist, bsms)
  }
  if (sst) {
    setwd(paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECH015/TRN-packaged-bins/sst/",

```

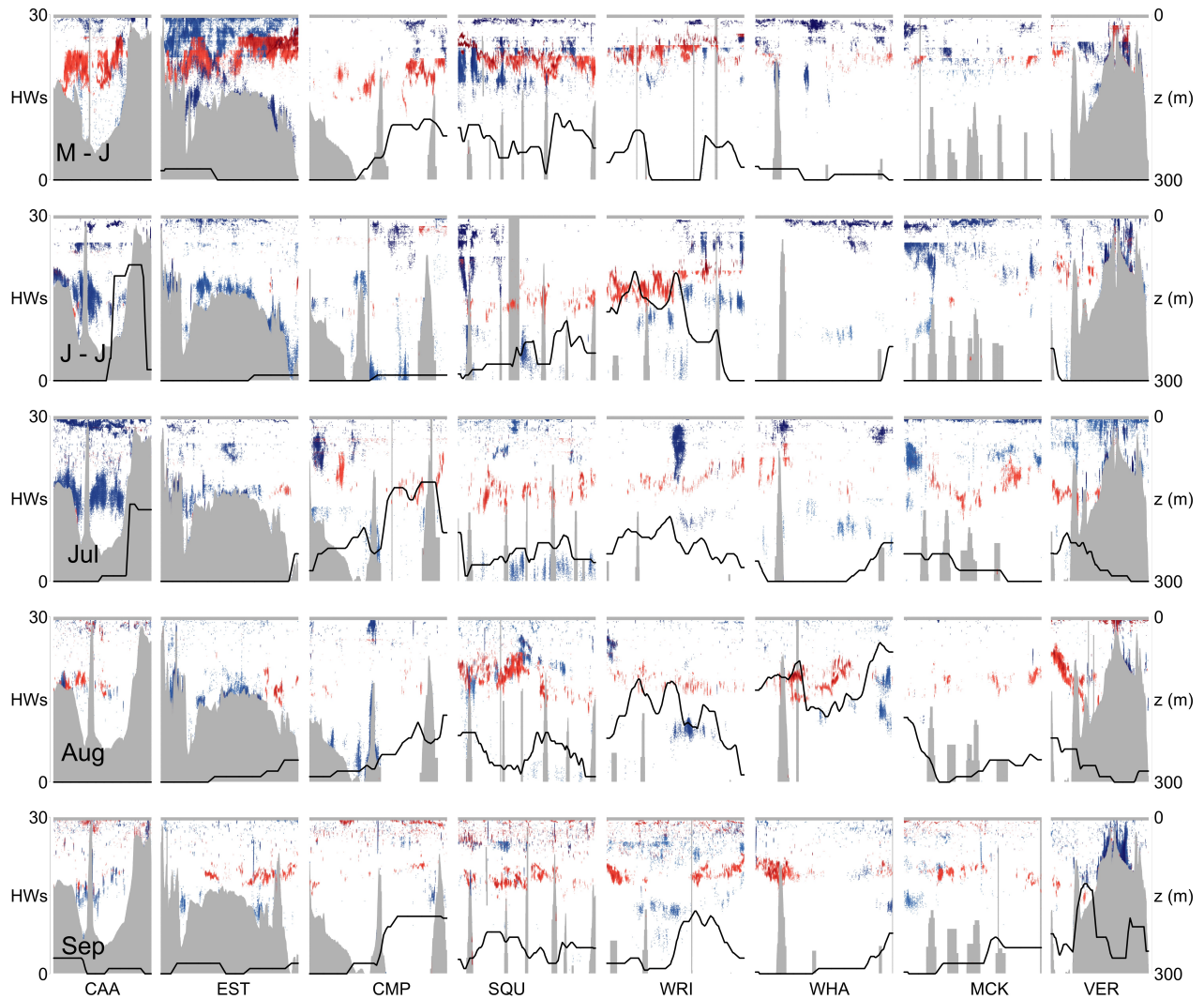
```

        sep = ""))
    sst <- read.csv(paste0("trn-bins-2015-", bin.size, ".csv"),
        stringsAsFactors = FALSE)
    sst <- sst[, 14:ncol(sst)]
    binlist <- cbind(binlist, sst)
}
if (inlet) {
    setwd(paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECH015/TRN-packaged-bins/inlet/",
        sep = ""))
    inlet <- read.csv(paste0("trn-bins-2015-", bin.size,
        ".csv"), stringsAsFactors = FALSE)
    inlet <- inlet[, 14:ncol(inlet)]
    binlist <- cbind(binlist, inlet)
}
if (ocean) {
    setwd(paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECH015/TRN-packaged-bins/ocean/",
        sep = ""))
    ocean <- read.csv(paste0("trn-bins-2015-", bin.size,
        ".csv"), stringsAsFactors = FALSE)
    ocean <- ocean[, 14:ncol(ocean)]
    binlist <- cbind(binlist, ocean)
}
final <- binlist
setwd(paste(wd, "Dropbox/Bangarang Docs/Data/2015/ECH015/TRN-bins-docs",
    sep = ""))
newname <- paste0("trn-bins-2015-", bin.size, ".csv")
write.csv(final, file = newname, quote = FALSE, row.names = FALSE)
}

```

## 8 Visualize

### 8.1 Profile view of unbinned data

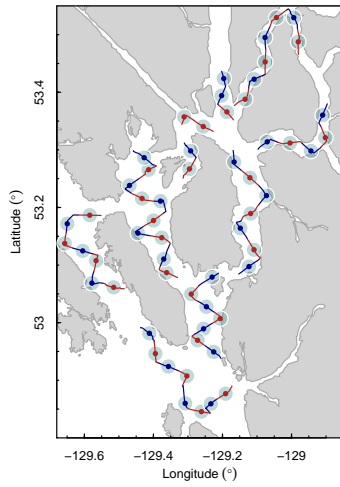


Humpback association with prey field in 5 monthly surveys (2015). Profiles of each survey consist of geographic blocks arranged roughly from offshore (left) to inshore (right). Profile background is acoustic backscatter (blue = fish-like, 33kHz; red = krill-like, 200kHz) from the surface (*profile top*) down to 300m.

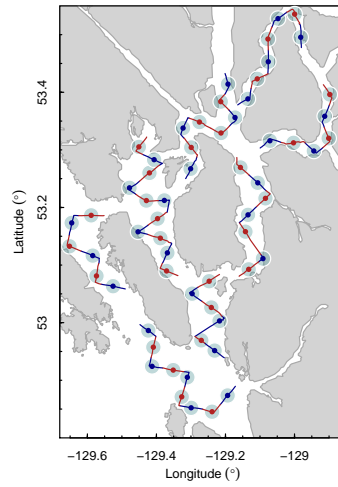
Color intensity indicates backscatter strength (deeper/darker colors = stronger backscatter). White is regions of water that did not register above a threshold backscatter level. Grey is seafloor and manually removed self-noise. Superimposed black line is a running mean (approx. 500m window size) of humpback whales seen within 3km of trackline.

## 8.2 Map view of transect bins

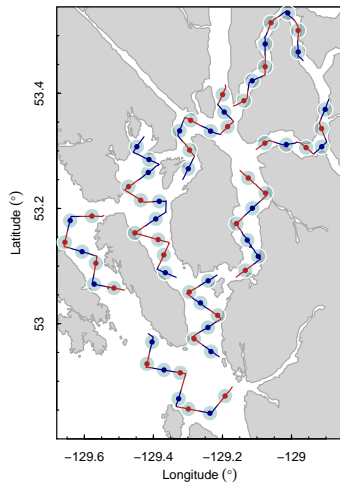
**June**



**July**



**Aug**



**Sep**

