FishDirection.R

GlennTattersall

Mon Jul 6 11:35:41 2015

```
############ TABLE OF CONTENTS ##############################
# 0.  Libraries - define them all at the top
# 1.  Define Functions
# 2.  Human inputs - manually enter file name as variable f (USER PROVIDES file name)
# 3.  Populate alldata variable from file
# 4a. Centre, Filter/Smooth the x,y detected data
# 4b. Simulate x,y data to check calculations - Remove for actual analysis
# 5.  Clean up the Fish Side Detection
# 6.  Define temperature fish selects and subset by side
# 7.  Movement trajectories using bearing function
# 8.  Plot Fish smooth (x,y) data
# 9.  Plot the trajectory data, polar plot, histograms
#10.  Summaries and final calculations
############ End of TOC ###################################

#0. Libraries required ####
# if missing, type install.packages("packagename") in console
library(plotrix)
library(Thermimage)
library(zoo)
```

```
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(signal)
```

```
##
## Attaching package: 'signal'
##
## The following objects are masked from 'package:stats':
##
##     filter, poly
```

```
library(pastecs)
```

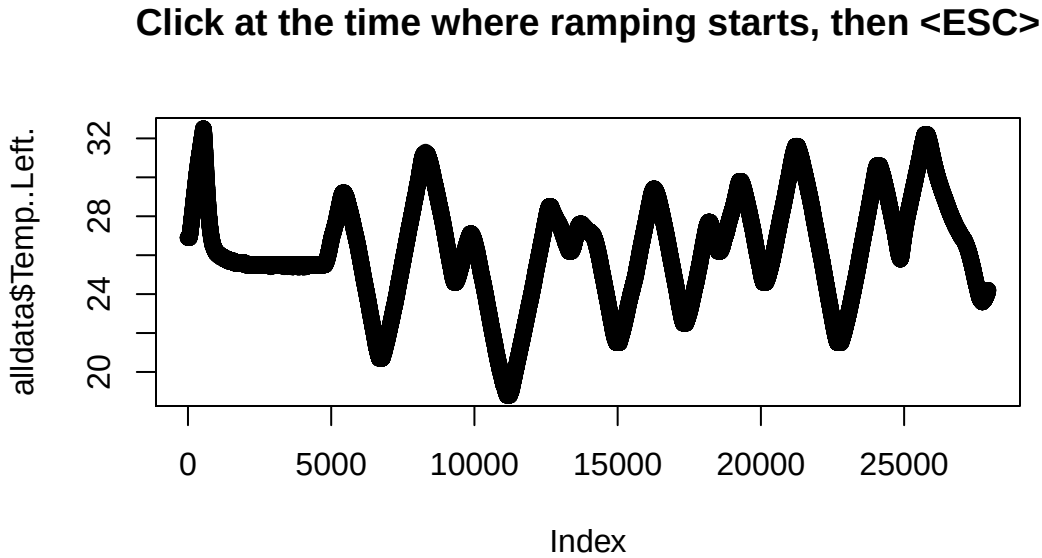
```
## Loading required package: boot
```

```
library(graphics)
library(fields)
```

```
## Loading required package: spam
## Loading required package: grid
## Spam version 1.0-1 (2014-09-09) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.
##
## Attaching package: 'spam'
##
## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve
##
## Loading required package: maps
##
## Attaching package: 'fields'
##
## The following object is masked from 'package:plotrix':
##
##     color.scale
```

```
bias.index.c<-NULL
options(warn=-1)

#1. Define Functions ####
bearing<-function(x,y,zmin=0)
  # this function accepts x and y values of detected coordinates for movement
  # and returns a data frame of four variables: magnitude, theta, change, alpha
  # magnitude and theta define the vectors describing the direction of movement
  # between time points i and i+1.  By necessity, magnitude and theta will be
  # 1 row shorter than the x,y input coordinates
  # change and alpha represent the change in magnitude and direction in the vector
  # of movement
  # zmin is the lowest acceptable level of magnitude required to accept the vector
  # of movement of being realisitc (i.e. zmin is the minimum detectable magnitude)
  # http://stackoverflow.com/questions/2571160/calculate-direction-angle-from-two-vectors
{
  dx<-diff(x)
  dy<-diff(y)
  velocity<-sqrt(dy^2+dx^2)
  # remove really small changes (i.e. zmin=3 units or less)
  dx[which(abs(velocity)<zmin)]<-NA
  dy[which(abs(velocity)<zmin)]<-NA
  velocity[which(abs(velocity)<zmin)]<-NA

  theta<-180*atan2(dy,dx)/pi
  # atan2 will calculate angles with respect to the horizontal.

  alpha<-diff(theta)
  alpha[which(alpha>180)]<-(-360+alpha[which(alpha>180)])
  alpha[which(alpha<(-180))]<-360+alpha[which(alpha<(-180))]
  # alpha is the instantaneous change in theta with respect to index
  # alternatively called the relative angle or turning angle
```

```
  # if the index is in unitary measures of time, alpha becomes degrees/time
  # a +ve alpha corresponds to a left turn, or counter-clockwise turn
  # a -ve alpha corresponds to a right turn, or clock-wise turn
  acceleration<-diff(velocity)
  # change is the magnitude of the relative turn
  alpha<-c(NA,alpha)
  acceleration<-c(NA,acceleration)
  # assign NA to the first value as a placeholder, so that alpha and change
  # align with the n+1 data points
  result<-data.frame(velocity, theta, acceleration, alpha)
}

addgradient <- function()
  {
  # creates the gradient fill colour ramp for smoothScatter plots
  xm <- get('xm', envir = parent.frame(1))
  ym <- get('ym', envir = parent.frame(1))
  z  <- get('dens', envir = parent.frame(1))
  colramp <- get('colramp', parent.frame(1))
  image.plot(xm,ym,z, col = colramp(256), legend.only = T, add =F)
}

spiral<-function(t=seq(1,100,0.1), size=5, direction="left")
{
  if(direction=="right")
  {
    x<-size*t*sin(t)
    y<-size*t*cos(t)
  }
  else if(direction=="left")
  {
    x<-size*t*cos(t)
    y<-size*t*sin(t)
  }
  out<-cbind(x,y)
}

#2. Human Inputs - Filename  ####
mainDir<-"~/Documents/R/MyProjects/ActivityDirection/data"
setwd(mainDir)
l.files<-list.files()
# User input #####
f<-l.files[3]


#3. Populate alldata data.frame ####
alldata<-read.csv(f, header=TRUE, skip=0, row.names=NULL)
par(mfrow=c(1,1), mar=c(10,5,5,5))
plot(alldata$Temp..Left., main="Click at the time where ramping starts, then <ESC>")
title(sub="Only the data following this time point will be analysed", line=4)
```



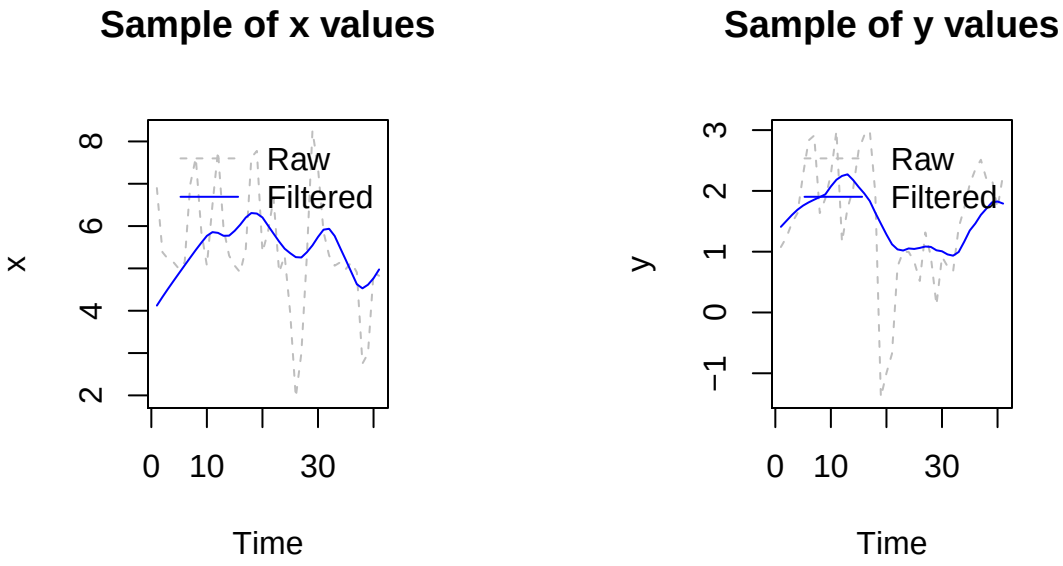

```
oldpar<-par()

timeremove<-4677
#timeremove<-round(mean(locator(1)$x),0)
if(timeremove<1) timeremove<-1
cat("The first", timeremove, "data points will be removed")
```

```
## The first 4677 data points will be removed
```

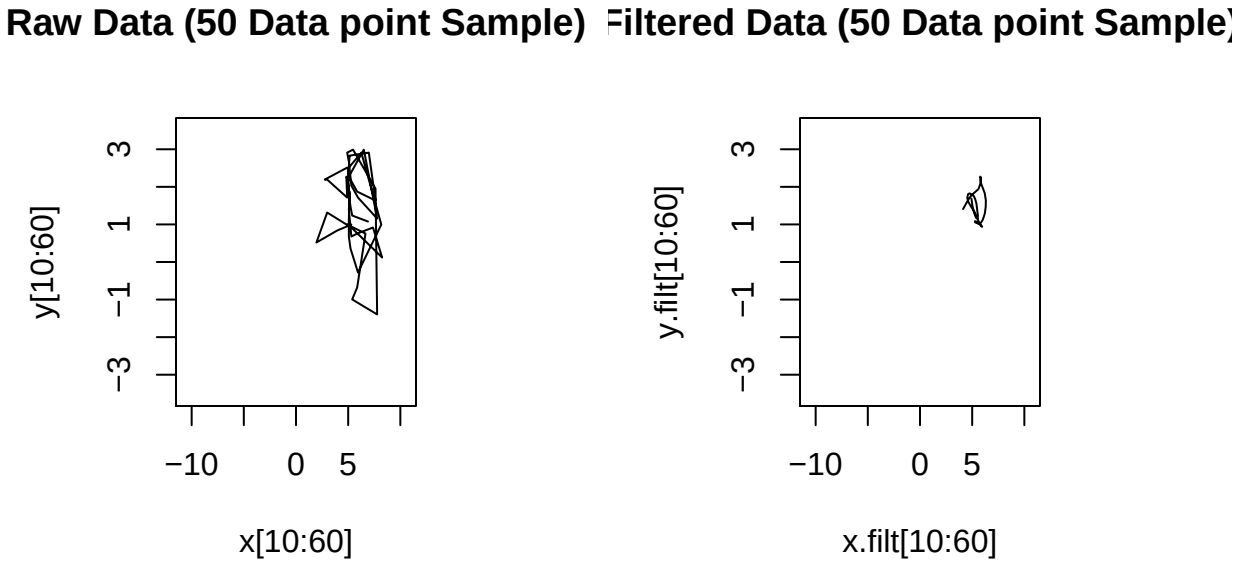
```
cat("\n")
```

```
#  User input #####
alldata<-alldata[-c(1:timeremove),]
#  remove the first hour (i.e. 3600 s) of data.

fit<-lm(100*alldata$Distance.m. ~ alldata$Distance.p.)
b<-fit$coef[1]
m<-fit$coef[2]
# use the data file to ascertain correlation between pixel and cm
colnames(alldata)[5]<-"Temp.Left"
colnames(alldata)[6]<-"Temp.Right"
alldata$X.Pos<-b+m*alldata$X.Pos
alldata$Y.Pos<-b+m*alldata$Y.Pos
alldata<-alldata[-c(8,9,10,11)]
# convert X.Pos and Y.Pos into actual distance units (cm)
```

```
alldata$Real<-as.POSIXct(as.character(alldata$Real), tz="", "%H:%M:%S")
alldata$Elapsed<-as.POSIXct(as.character(alldata$Elapsed), tz="", "%H:%M:%S")
diff.time<-as.integer(alldata$Elapsed-alldata$Elapsed[1])
t<-1:nrow(alldata)
alldata<-data.frame(diff.time, alldata, row.names=NULL)

#4a. Centre, Filter, and Smooth the x and y data ####
alldata$X.Pos<-alldata$X.Pos-(max(alldata$X.Pos,na.rm=TRUE) +
                              min(alldata$X.Pos,na.rm=TRUE))/2
alldata$Y.Pos<-alldata$Y.Pos-(max(alldata$Y.Pos,na.rm=TRUE) +
                              min(alldata$Y.Pos,na.rm=TRUE))/2
half.x<-0
# centre the x data, assuming that the max and min values detected are the extremes

x<-alldata$X.Pos
y<-alldata$Y.Pos
bf<-butter(2, 0.05, type="low")
x.filt1<-filter(bf, x)
y.filt1<-filter(bf, y)
x.filt2<-sgolayfilt(x, 9, 35)
y.filt2<-sgolayfilt(y, 9, 35)
x.filt<-(x.filt1+x.filt2)/2
y.filt<-(y.filt1+y.filt2)/2
# User input ####
# empirically determined filters - may require adjustment

x.filt<-na.locf(x.filt, na.rm=FALSE)
y.filt<-na.locf(y.filt, na.rm=FALSE)
# replace any NA values with previous non-NA value

par(mfrow=c(1,2))
plot(x[10:50], type="l", main="Sample of x values",
     xlab="Time", ylab="x", col="grey", lty=2)
lines(x.filt[10:50], type="l", col="blue")
legend(x="topright", c("Raw", "Filtered"), col=c("grey", "blue"), lty=c(2,1), bty="n")
plot(y[10:50], type="l", main="Sample of y values",
     xlab="Time", ylab="y", col="grey", lty=2)
lines(y.filt[10:50], type="l", col="blue")
legend(x="topright", c("Raw", "Filtered"), col=c("grey", "blue"), lty=c(2,1), bty="n")
```



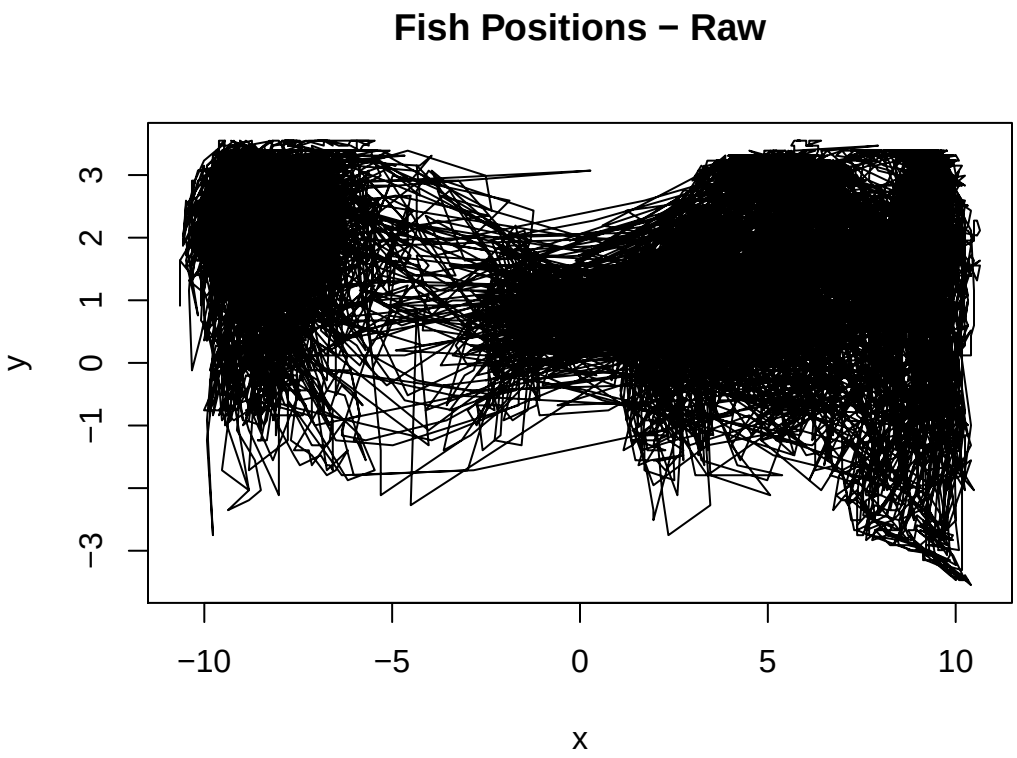

```
# 10 data points shown for illustrative purposes")


alldata<-data.frame(alldata,x.filt,y.filt, row.names=NULL)

plot(x[10:60], y[10:60], main="Raw Data (50 Data point Sample)",
     type='l',  xlim=c(min(x),max(x)), ylim=c(min(y),max(y)))
plot(x.filt[10:60], y.filt[10:60], main="Filtered Data (50 Data point Sample)",
     type='l', xlim=c(min(x),max(x)), ylim=c(min(y),max(y)))
```



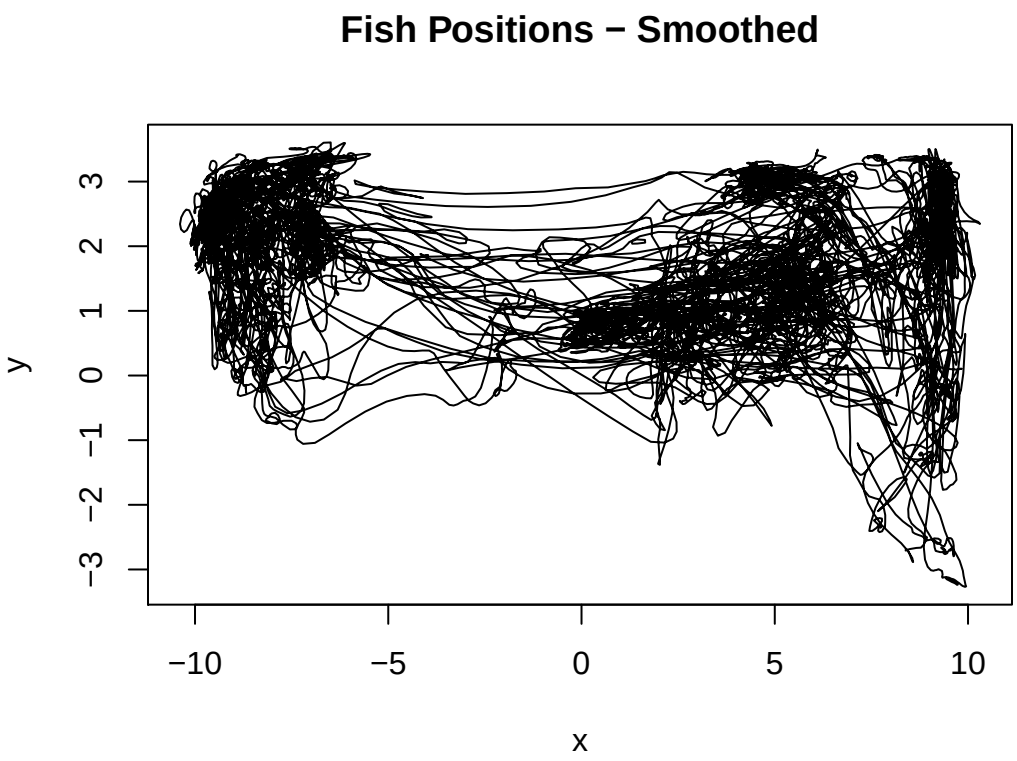

```
#4b. Simulate x,y as a spiral or random #####
rand<-2
# set rand to 1 to run a random sample,
# set rand to 0 to run a spiral,
# set rand to any other number to use imported data
if(rand==1)
{
  #   theta.rand<-runif(nrow(alldata), -1, 1)
  #   mag.rand<-runif(nrow(alldata), 0, 1)
  #   x.rand<-mag.rand*sin(theta.rand)
  #   x.rand<-cumsum(x.rand)
  #   y.rand<-mag.rand*cos(theta.rand)
  #   y.rand<-cumsum(y.rand)
  #
  index.rand<-sample(1:nrow(alldata), nrow(alldata))
  x.rand<-alldata$X.Pos[index.rand]
  y.rand<-alldata$Y.Pos[index.rand]

  alldata$x.filt<-x.rand
  alldata$y.filt<-y.rand
}
if(rand==0)
{
  spir<-spiral(seq(1,nrow(alldata)), 1, "right")
  alldata$x.filt<-spir[,1]
  alldata$y.filt<-spir[,2]
```

```
}

#5.  Clean up Fish.Side ??? detections ####
alldata$Fish.Side<-as.character(alldata$Fish.Side)
q<-which(!alldata$Fish.Side=="???")
if(alldata$Fish.Side[1]=="???") alldata$Fish.Side[1]<-alldata$Fish.Side[q[1]]

for(i in 1:nrow(alldata))
{
  j<-i-1
  #if(alldata$Fish.Side[i]=="???")
  # {
  if(x.filt[i]<half.x)  alldata$Fish.Side[i]<-"LEFT"
  if(x.filt[i]>half.x)  alldata$Fish.Side[i]<-"RIGHT"
  if(x.filt[i]==half.x) alldata$Fish.Side[i]<-alldata$Fish.Side[j]
  #  }
}

alldata$Fish.Side<-as.factor(alldata$Fish.Side)

#6. Define fish temperature ####
slopeleft<-slopeEveryN(alldata$Temp.Left, 30)
sloperight<-slopeEveryN(alldata$Temp.Right, 30)
overallslope<-(slopeleft+sloperight)/2

left.side<-which(alldata$Fish.Side=="LEFT")
right.side<-which(alldata$Fish.Side=="RIGHT")
fish.temp<-1:nrow(alldata) # populate variable fish.temp
fish.temp[left.side]<-alldata$Temp.Left[left.side]
fish.temp[right.side]<-alldata$Temp.Right[right.side]
alldata<-data.frame(alldata,fish.temp, row.names=NULL)

fs<-as.integer(alldata$Fish.Side)-1
# convert Fish.Side character to an integer vector, where Left=0, Right=1
sideswitch<-cbind(1:nrow(alldata), c(0,diff(fs)))
sideswitch<-data.frame(sideswitch)
colnames(sideswitch)<-c("Index","Switch")
# create a indexed matrix that indicates where shuttles occurred
# +1 = moving from left into right (i.e. lower escape)
# - = moving from right into left (i.e. upper escape)
sideswitch<-subset(sideswitch, Switch >0 | Switch<0)
le.index<-sideswitch[which(sideswitch[,2]==1),1]
ue.index<-sideswitch[which(sideswitch[,2]==-1),1]
let<-alldata$Temp.Left[le.index]
uet<-alldata$Temp.Right[ue.index]


#7. Work out movement trajectories using bearing function #####
fish.bearing<-bearing(alldata$x.filt,alldata$y.filt,0)
velocity<-fish.bearing[,1]
velocity<-na.locf(velocity, na.rm=FALSE)
# use na.locf to replace NA values with previous entry
theta<-fish.bearing[,2]
```

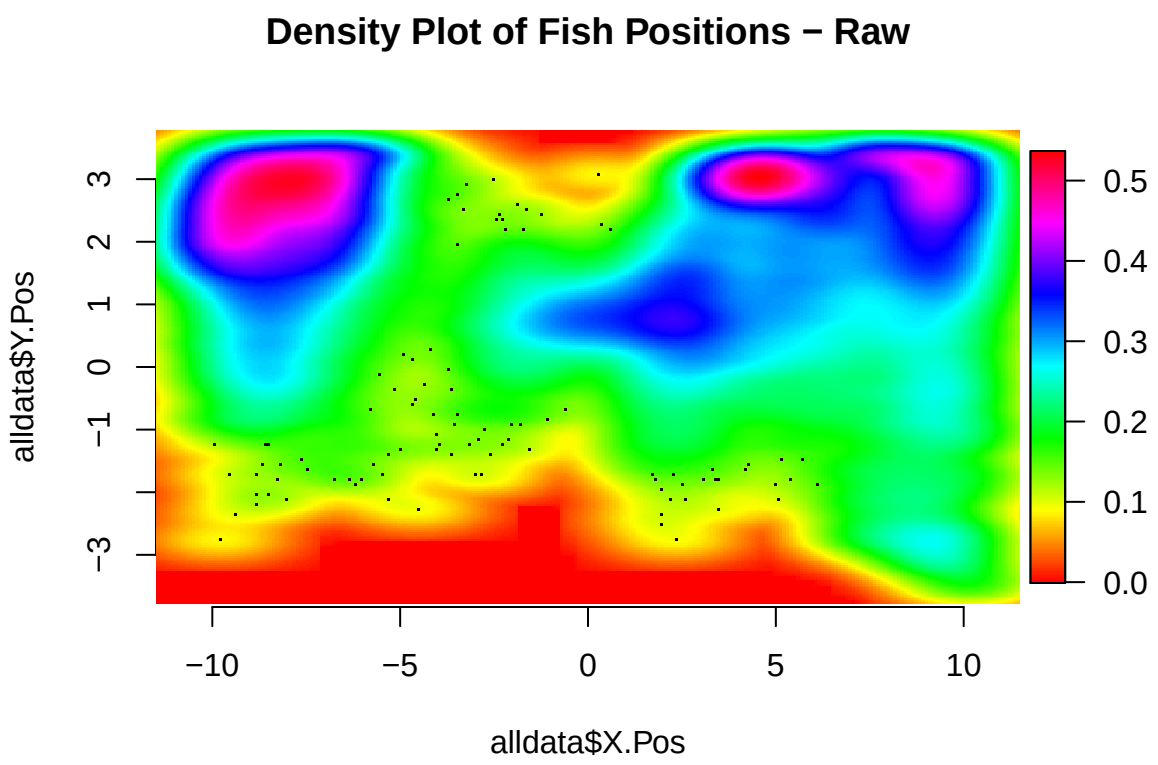
```
theta<-na.locf(theta, na.rm=FALSE)
# use na.locf to replace NA values with previous entry
# this will bestow memory to the direction the fish was last moving
acceleration<-abs(diff(velocity))
alpha<-diff(theta)
# recalculate differences based on NA replaced velocities and thetas
# alphas = 0 will mean a stationary animal
# alphas <> 0 can then be examined for how large or directional they are
alpha[which(alpha>180)]<-(-360+alpha[which(alpha>180)])
alpha[which(alpha<(-180))]<-360+alpha[which(alpha<(-180))]
# convert alpha angles so that 0 to 180 = left, counterclockwise
# and -0 to -180 = right, clockwise

alldata<-data.frame(alldata[-1,], fish.bearing, row.names=NULL)
# remove first row from original dataframe before adding the bearing output
# since the bearing data is based on one less data point

colnames(alldata)<-c("Time.diff",  "Real", "Elapsed", "X.Pos", "Y.Pos", "Temp.Left",
                     "Temp.Right", "Fish.Side", "x.filt", "y.filt", "fish.temp",
                     "velocity", "theta", "acceleration", "alpha")

#8. Plot Fish smoothed (X,Y) data ####
par(mfrow=c(1,1), mar=c(5,5,5,5))
plot(alldata$X.Pos, alldata$Y.Pos, type="l",
     main="Fish Positions - Raw", xlab="x", ylab="y")
```



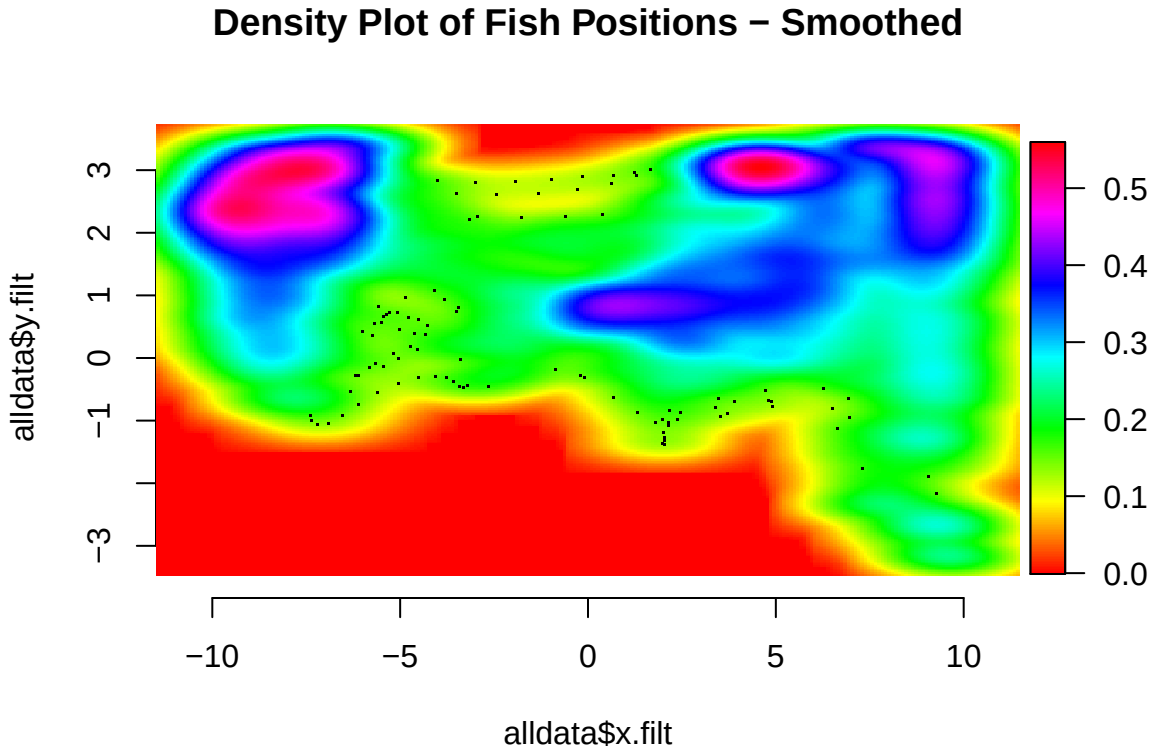
```
plot(alldata$x.filt,alldata$y.filt, type="l",
     main="Fish Positions - Smoothed", xlab="x", ylab="y")
```




```
smooth.palette <- colorRampPalette(blues9[-(1:2)])
#smooth.palette <- heat.colors(7)
smoothScatter(alldata$X.Pos, alldata$Y.Pos, nbin=256, bty="n",
              main="Density Plot of Fish Positions - Raw", colramp=rainbow,
              xlim=c(min(alldata$X.Pos),max(alldata$X.Pos)),
              ylim=c(min(alldata$Y.Pos),max(alldata$Y.Pos)),
              postPlotHook=addgradient)
```

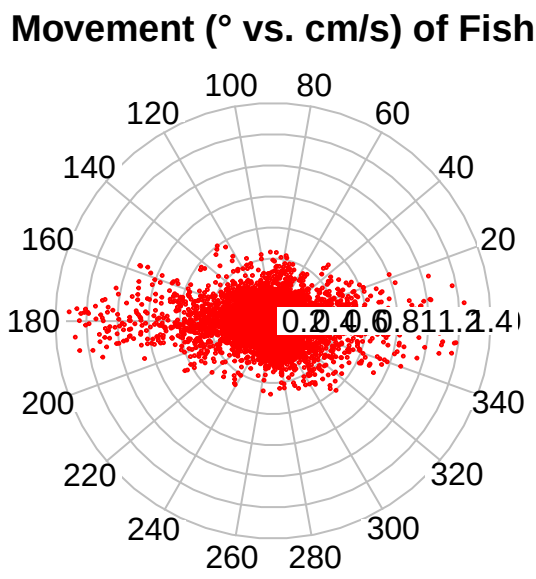


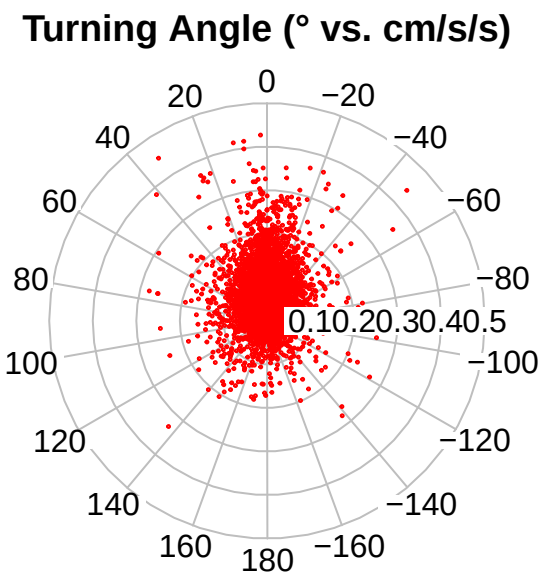

```
smoothScatter(alldata$x.filt, alldata$y.filt, nbin=256, bty="n",
              main="Density Plot of Fish Positions - Smoothed", colramp=rainbow,
              xlim=c(min(alldata$X.Pos),max(alldata$X.Pos)),
              ylim=c(min(alldata$Y.Pos),max(alldata$Y.Pos)),
              postPlotHook=addgradient)
```




```
#
# par(mfrow=c(1,1))
# plot(0, 0, type="n", pch=16, xlim=c(min(alldata$x.filt),max(alldata$x.filt)),
#      ylim=c(min(y.filt),max(y.filt)), main="Detected Positions - Filtered",
#      xlab="X", ylab="Y")
# for (i in 2:nrow(alldata))
# {
#   j<-i-1
#   x<-alldata$x.filt[j:i]
#   y<-alldata$y.filt[j:i]
#   z<-alldata$mag[i]
#   col.point<-"grey"
#   cex.val<-0.1
#
#   if(!is.na(alldata$alpha[i]))
#   {
#     if(alldata$alpha[i]>0)
#     {
#       col.point<-c("blue")
#       cex.val<-1
#     }
#     if(alldata$alpha[i]<0)
#     {
#       col.point<-c("red")
#       cex.val<-1
#     }
```

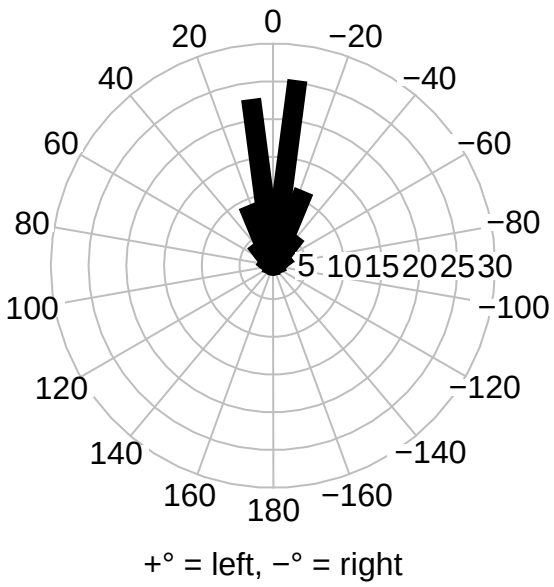
```
#   }
#
#
#   lines(x, y, pch=16, col=col.point)
#   #Sys.sleep(0.25)
# }


#9.  Plot the trajectory data ####
par(mfrow=c(1,2))
polar.plot(velocity, theta, line.col="red", point.col="red", rp.type="s",
           start=0, main=("Movement (° vs. cm/s) of Fish"), cex=0.2)
title(sub="Angle (°) is relative to horizontal = 0°", line=1)
polar.plot(acceleration[-1], alpha[-1], point.col="red", start=90, rp.type="s",
           main="Turning Angle (° vs. cm/s/s)", cex=0.2,
           labels=c(0,20,40,60,80,100,120,140,160,180,-160,-140,-120,-100,
                    -80,-60,-40,-20))
title(sub="+° = left, -° = right", line=1)
```



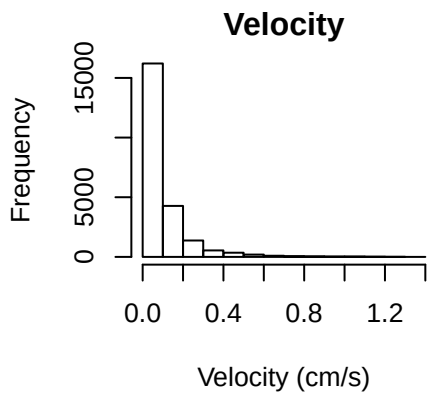


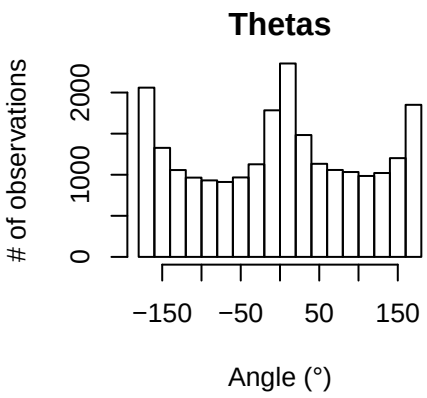

```
bin.labels<-seq(-172.5,172.5,15)
bins<-seq(-180,180,15)

par(mfrow=c(1,1))
freq.alpha<-as.numeric(table(cut(alpha, breaks=bins, labels=bin.labels)))
polar.plot(100*freq.alpha/sum(freq.alpha), bin.labels, line.col="black",
```

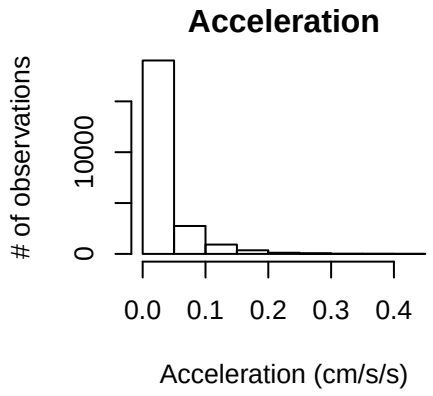
```
            rp.type="r", start=90, lwd=10, lend=1,
            main="Distribution of Alphas (% of observed turning angles)",
            labels=c(0,20,40,60,80,100,120,140,160,180,-160,-140,-120,-100,
                     -80,-60,-40,-20),
            mar=c(5,5,5,5))
title(sub="+° = left, -° = right", line=1)
```

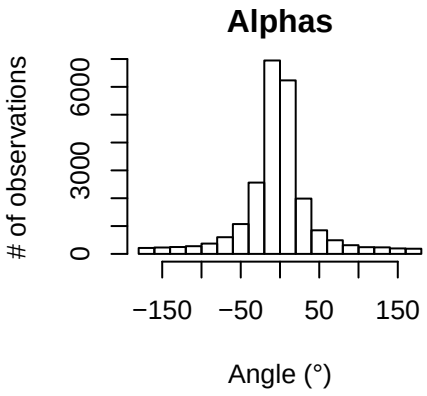



```
par(oldpar)
par(mfrow=c(2,2), mar=c(5,5,2,5))
hist(velocity, main="Velocity", xlab="Velocity (cm/s)")
hist(theta, main="Thetas", xlab="Angle (°)", ylab="# of observations")
hist(acceleration[-1], main="Acceleration", xlab="Acceleration (cm/s/s)",
     ylab="# of observations")
hist(alpha[-1], main="Alphas", xlab="Angle (°)", ylab="# of observations" )
```





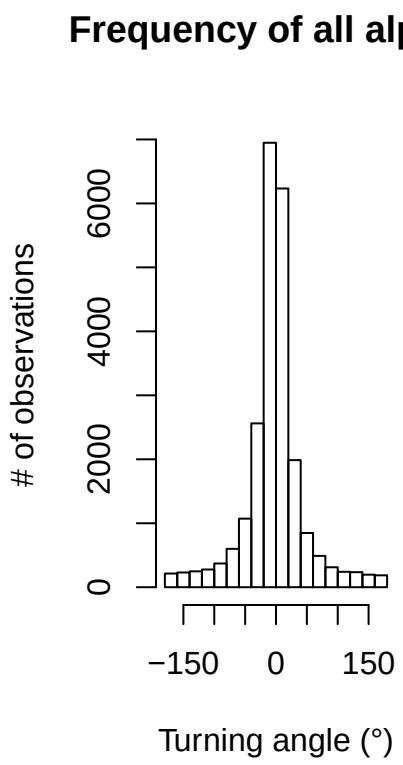


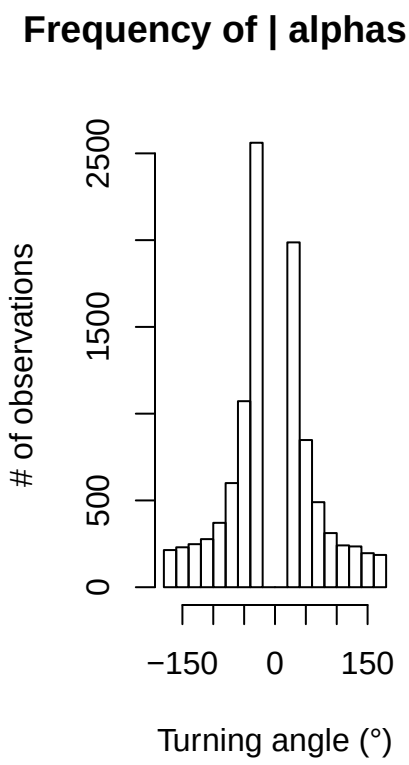



```
mean(alpha, na.rm=TRUE)
```

```
## [1] -2.726136
```

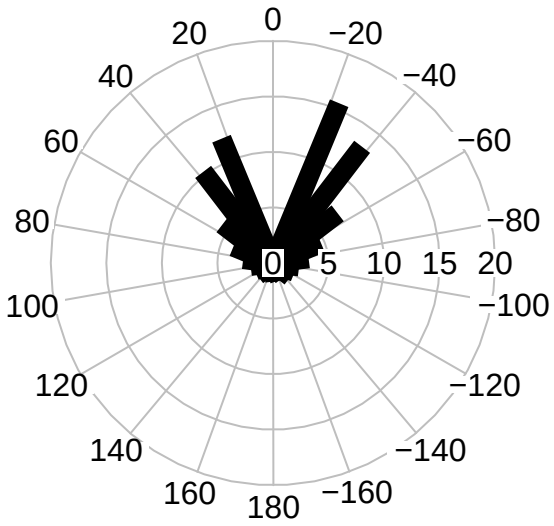
```
par(mfrow=c(1,2), mar=c(5,5,5,5))
hist(na.omit(alpha), main="Frequency of all alphas", xlab="Turning angle (°)",
     ylab="# of observations")
hist(na.omit(alpha[which(abs(alpha)>20)]), main="Frequency of | alphas | > 20°",
     xlab="Turning angle (°)", ylab="# of observations")
```





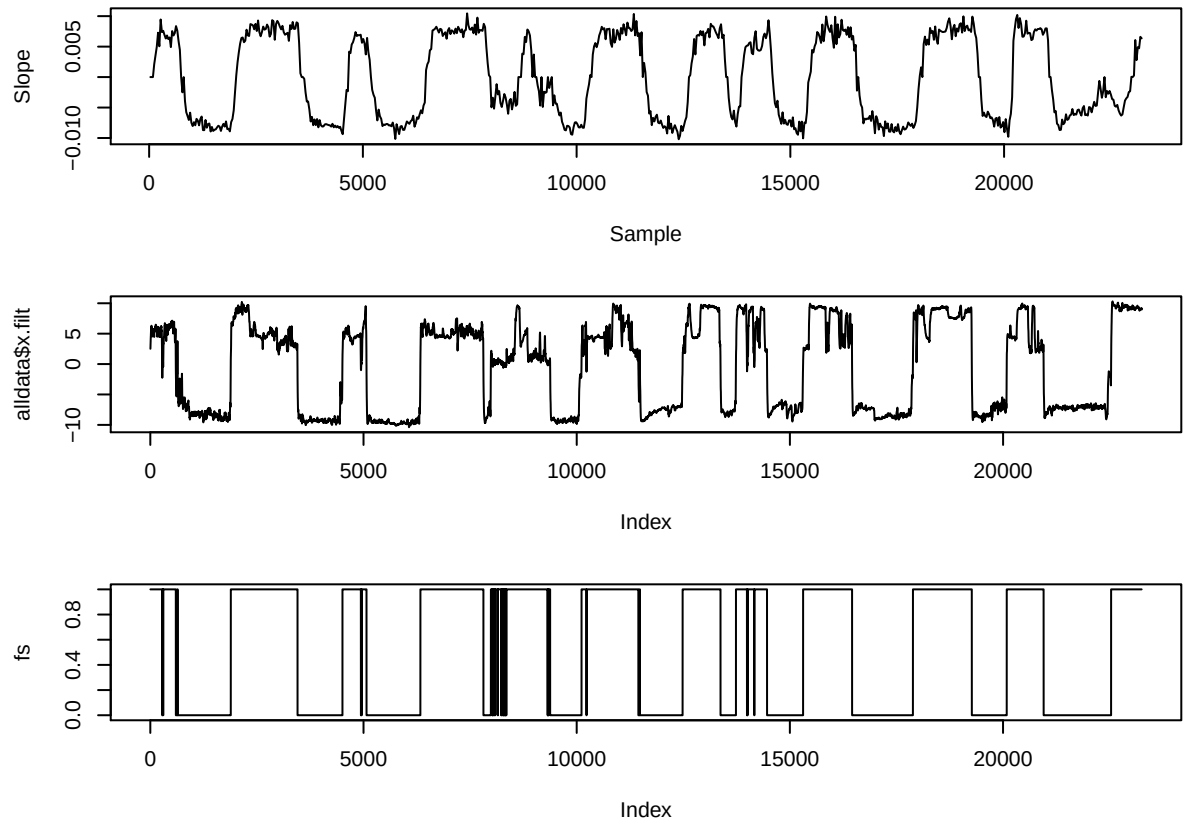

```
par(mfrow=c(1,1))
freq.alpha20<-as.numeric(table(cut(alpha[which(abs(alpha)>20)], breaks=bins,
                                   labels=bin.labels)))
polar.plot(100*freq.alpha20/sum(freq.alpha20), bin.labels, line.col="black",
           rp.type="r", start=90, lwd=10, lend=1,
           main="Distribution of | alphas | > 20° (% of observed turning angles)",
           labels=c(0,20,40,60,80,100,120,140,160,180,-160,-140,-120,-100,
                    -80,-60,-40,-20),
           mar=c(5,5,5,5))
title(sub="+° = left, -° = right", line=1)
```

Distribution of | alphas | > 20° (% of observed turning angles)

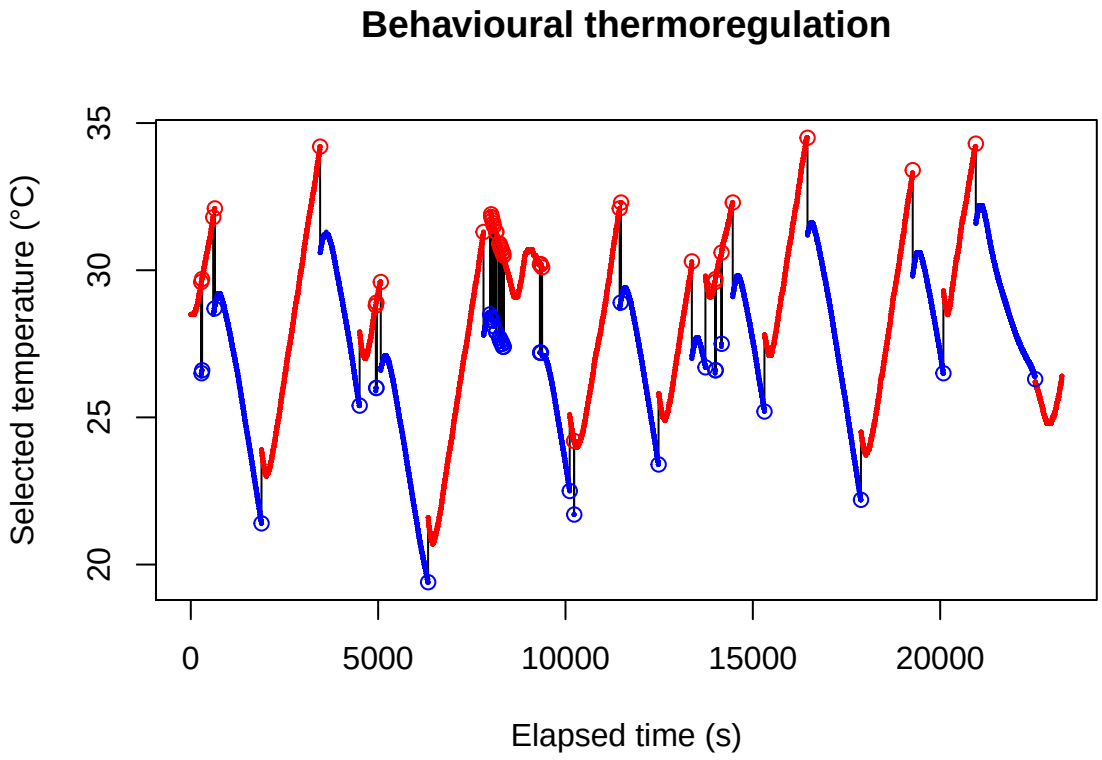


+° = left, −° = right

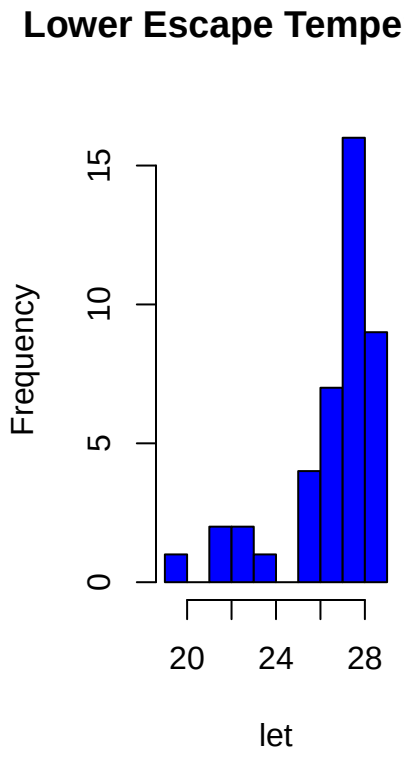
```
#10. Subset data by side ####
pdr<-subset(alldata, Fish.Side=="RIGHT", row.names=NULL)
pdl<-subset(alldata, Fish.Side=="LEFT", row.names=NULL)

par(oldpar)

par(mfrow=c(3,1), mar=c(5,5,1,2))
plot(overallslope, type="l")
plot(alldata$x.filt, type="l")
plot(fs, type="l")
```

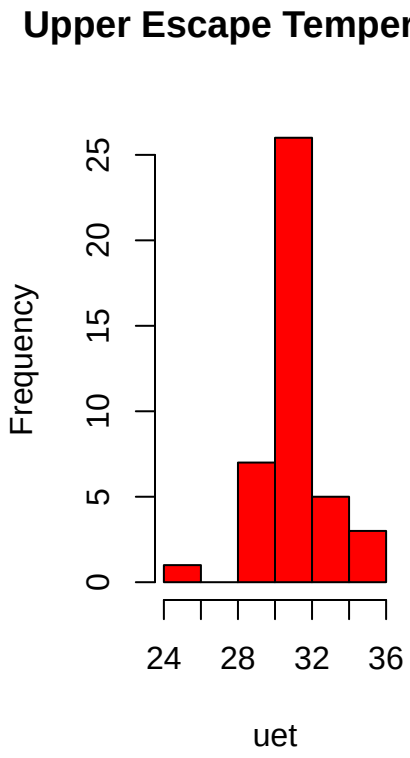



```
par(oldpar)
par(mfrow=c(1,1), mar=c(5,5,5,3))
plot(alldata$Time.diff, alldata$fish.temp, type="l", col="black", main="Behavioural thermoregulation",
     xlab="Elapsed time (s)", ylab="Selected temperature (°C)")
points(pdr$Time.diff, pdr$fish.temp, col="red", cex=0.2)
points(pdl$Time.diff, pdl$fish.temp, col="blue", cex=0.2)
points(alldata$Time.diff[le.index], let, col="blue")
points(alldata$Time.diff[ue.index], uet, col="red")
```




```
par(mfrow=c(1,2), mar=c(5,5,5,5))
hist(let, main="Lower Escape Temperatures", col="blue")
hist(uet, main="Upper Escape Temperatures", col="red")
```






```
stat.desc(let)
```

```
##      nbr.val     nbr.null       nbr.na          min          max
## 4.200000e+01 0.000000e+00 0.000000e+00 1.940000e+01 2.890000e+01
##        range          sum       median         mean      SE.mean
## 9.500000e+00 1.114500e+03 2.730000e+01 2.653571e+01 3.379556e-01
## CI.mean.0.95          var      std.dev     coef.var
## 6.825151e-01 4.796986e+00 2.190202e+00 8.253791e-02
```

```
stat.desc(uet)
```

```
##      nbr.val     nbr.null       nbr.na          min          max
##   42.0000000    0.0000000    0.0000000   24.2000000   34.5000000
##        range          sum       median         mean      SE.mean
##   10.3000000 1299.6000000   30.8000000   30.9428571    0.2623521
## CI.mean.0.95          var      std.dev     coef.var
##    0.5298307    2.8908014    1.7002357    0.0549476
```

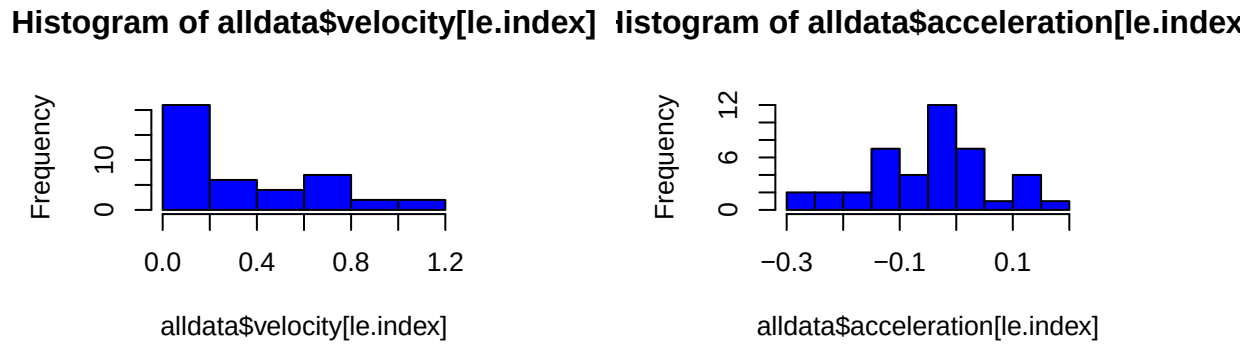
```
mean(let)
```

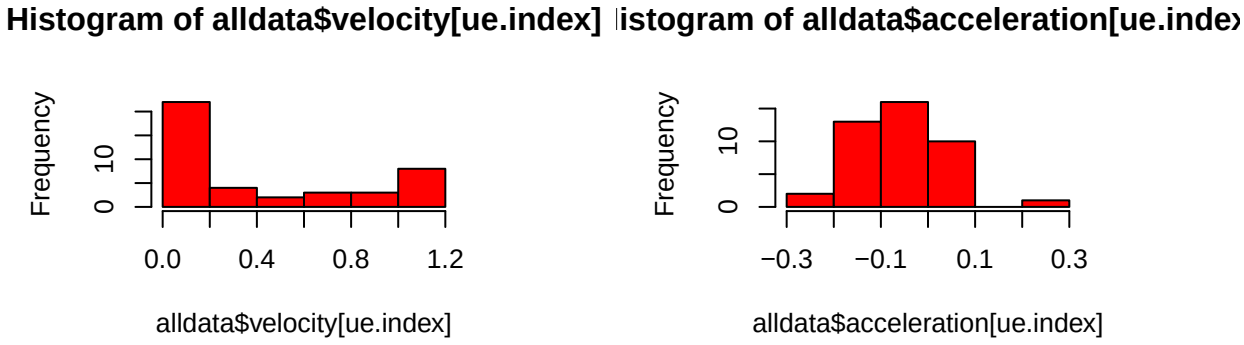
```
## [1] 26.53571
```

```r
mean(uet)
```

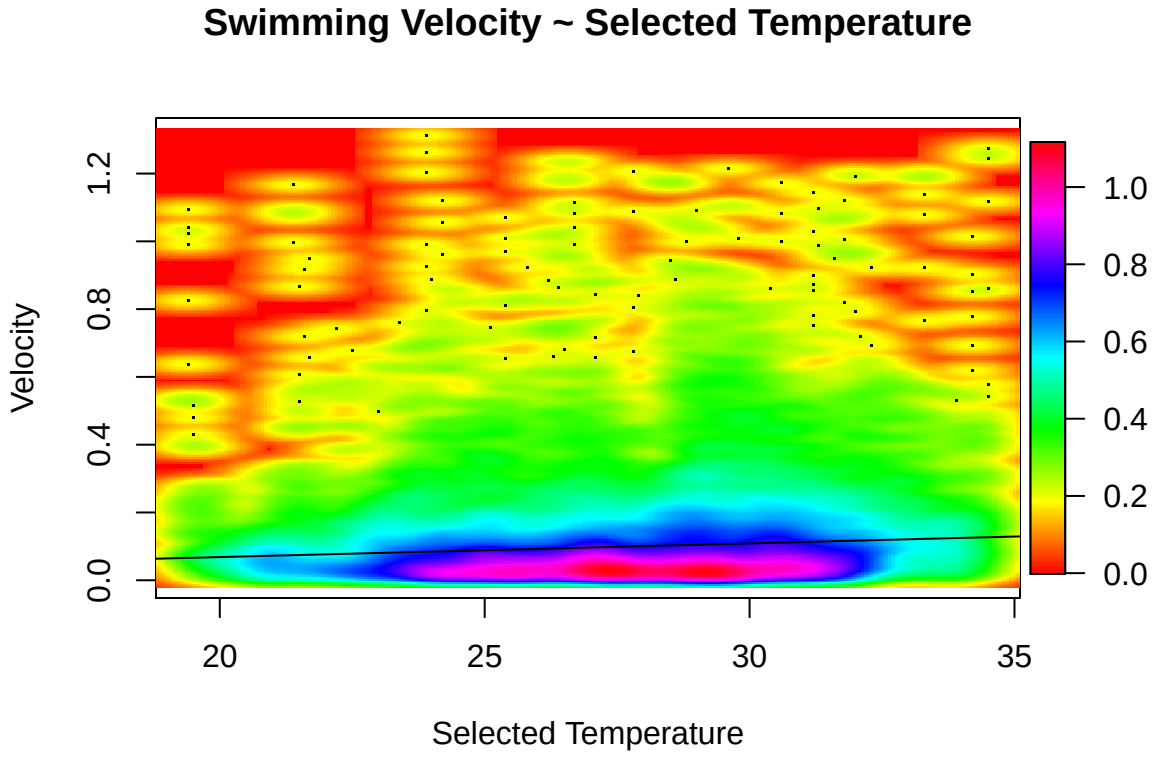
```
## [1] 30.94286
```

```r
par(mfcol=c(2,2), mar=c(5,5,5,5))
hist(alldata$velocity[le.index], col="blue")
hist(alldata$velocity[ue.index], col="red")

hist(alldata$acceleration[le.index], col="blue")
hist(alldata$acceleration[ue.index], col="red")
```

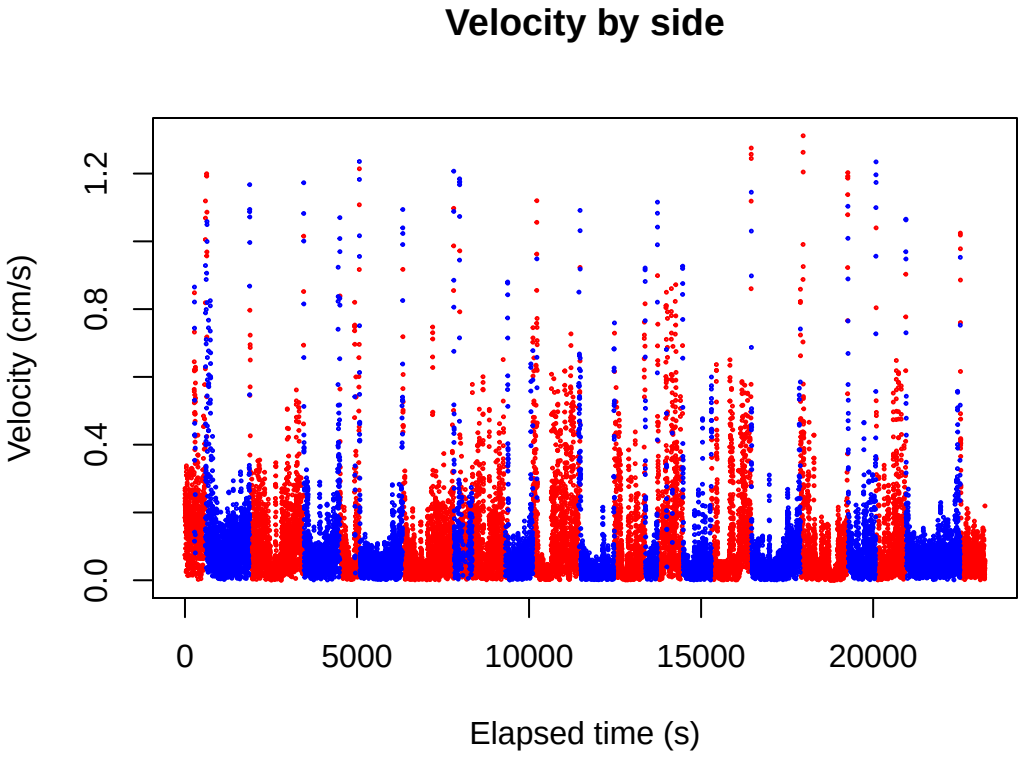




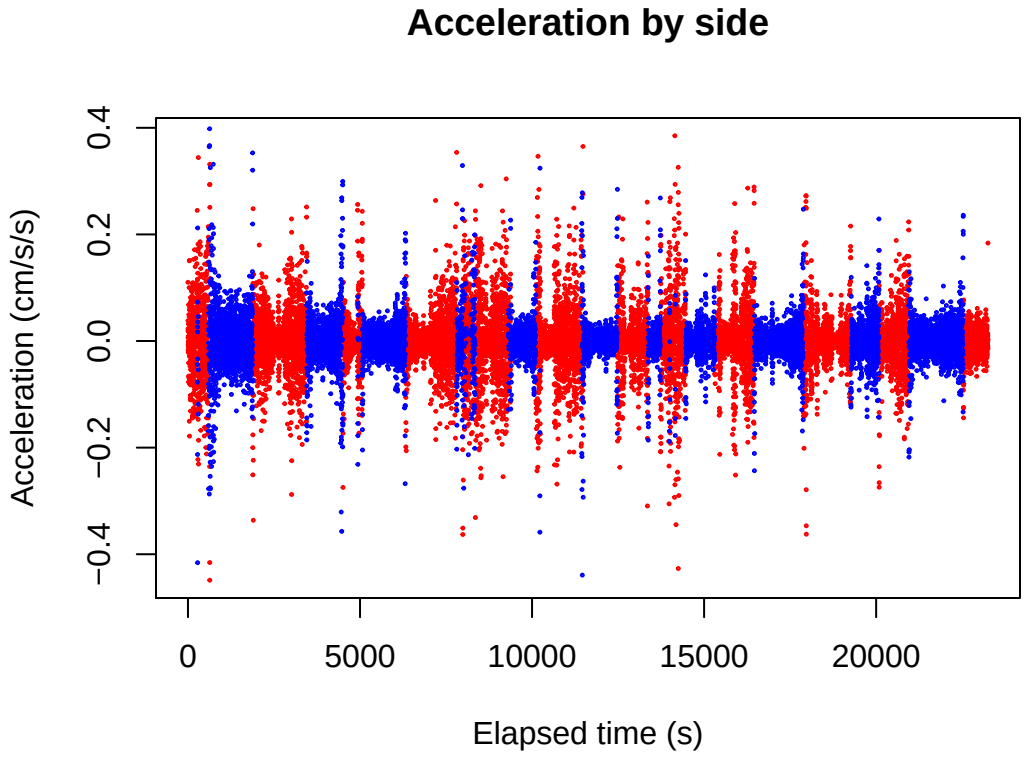

```r
par(mfrow=c(1,1), mar=c(5,5,5,5))
smoothScatter(alldata$fish.temp, alldata$velocity, nbin=256,
              colramp=rainbow,
              main="Swimming Velocity ~ Selected Temperature",
              xlab="Selected Temperature",
              "ylab"="Velocity",
              postPlotHook=addgradient)
fit<-lm(velocity~fish.temp, data=alldata)
abline(coef=fit$coef, xpd=FALSE)
```



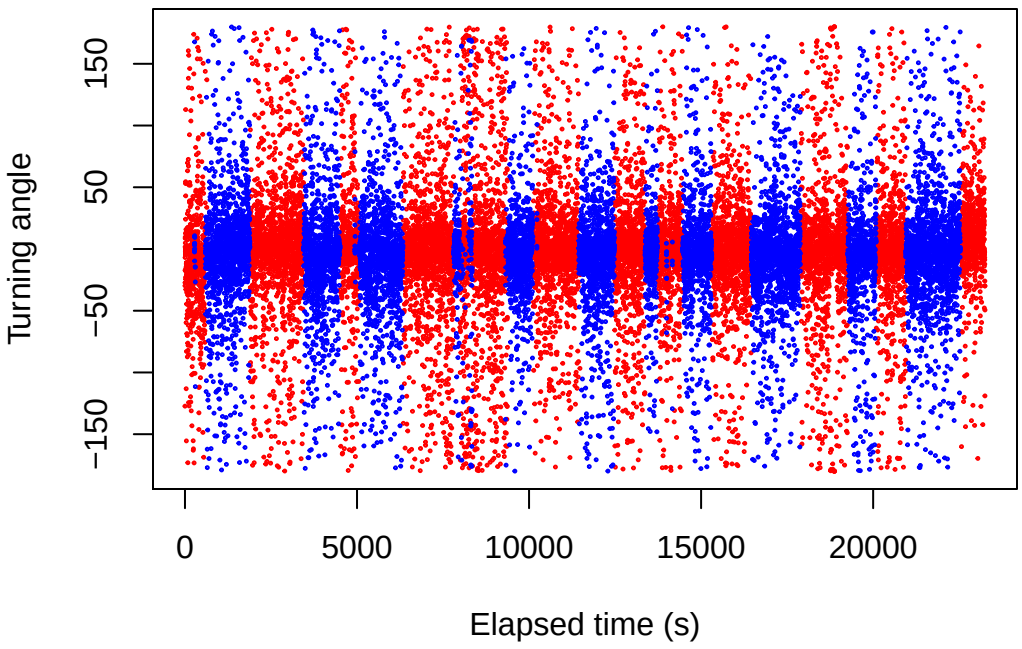

```
#10. Plot trajectory data based on side ####
par(mfrow=c(1,1), mar=c(5,5,5,5))
plot(pdr$Time.diff, pdr$velocity, type="p", col="red", main="Velocity by side",
     xlab="Elapsed time (s)", ylab="Velocity (cm/s)", cex=0.2)
points(pdl$Time.diff, pdl$velocity, col="blue", cex=0.2)
```



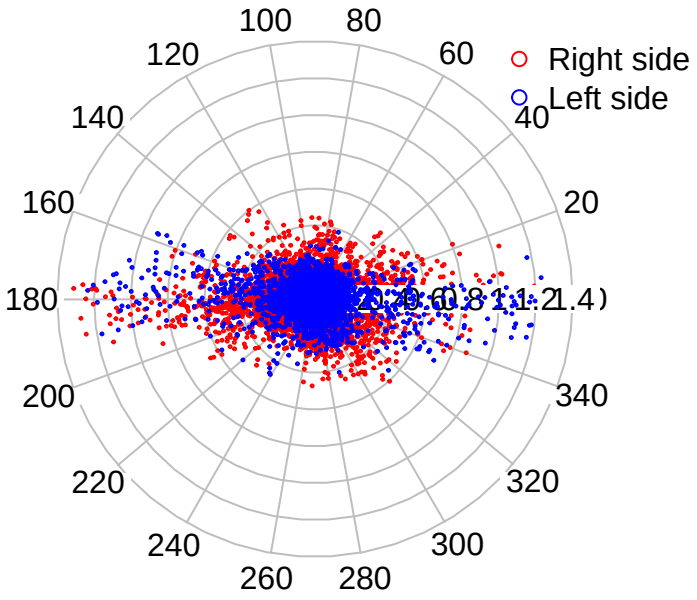

```
plot(pdr$Time.diff, pdr$acceleration, type="p", col="red", main="Acceleration by side",
     xlab="Elapsed time (s)", ylab="Acceleration (cm/s/s)", cex=0.2)
points(pdl$Time.diff, pdl$acceleration, col="blue", cex=0.2)
```



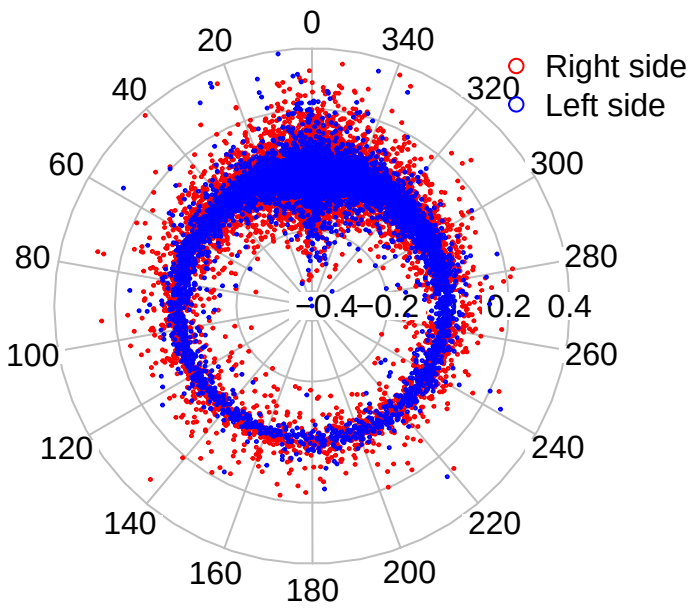

```
plot(pdr$Time.diff, pdr$alpha, type="p", col="red", main="Alpha by side",
     xlab="Elapsed time (s)", ylab="Turning angle", cex=0.2)
points(pdl$Time.diff, pdl$alpha, col="blue", cex=0.2)
```




```
par(mfrow=c(1,1), xpd=TRUE)
polar.plot(pdr$velocity, pdr$theta, point.col="red", rp.type="s", start=0,
           main="Movement vectors (° vs. cm/s)", cex=0.2, mar=c(4,4,4,4))
polar.plot(pdl$velocity, pdl$theta, point.col="blue", cex=0.2, rp.type="s", start=0,
           add=TRUE)
legend(x="topright", c("Right side", "Left side"), col=c("red", "blue"), pch=c(1,1),
       bty="n", inset=c(-0.2,0))
```




```
par(mfrow=c(1,1), xpd=TRUE)
polar.plot(pdr$acceleration[-1], pdr$alpha[-1], point.col="red", start=90, rp.type="s",
           main="Turning vectors (° vs. cm/s/s)", mar=c(4,4,4,4), cex=0.2)
polar.plot(pdl$acceleration[-1], pdl$alpha[-1], point.col="blue", start=90, rp.type="s",
           add=TRUE, cex=0.2)
legend(x="topright", c("Right side", "Left side"), col=c("red", "blue"), pch=c(1,1),
       bty="n", inset=c(-0.2,0))
```




```
par(oldpar)

#11. Summarise and Output
options(scipen=999)
overall.sum<-stat.desc(alldata[,c(11:15)], basic=TRUE, desc=TRUE, norm=FALSE, p=0.95)
overall.sum
```

```
##                   fish.temp         velocity          theta
## nbr.val       23250.00000000 23250.0000000000  23250.0000000
## nbr.null          0.00000000     0.0000000000      0.0000000
## nbr.na            0.00000000     0.0000000000      0.0000000
## min              19.40000000     0.0001584329   -179.9979144
## max              34.50000000     1.3114811299    179.9999480
## range            15.10000000     1.3113226970    359.9978624
## sum          646100.10000000  2322.6743848342 -12782.3481298
## median           28.00000000     0.0579871309      3.7029816
## mean             27.78925161     0.0998999735     -0.5497784
## SE.mean           0.01914993     0.0008500008      0.6994890
## CI.mean.0.95      0.03753513     0.0016660578      1.3710447
## var               8.52623528     0.0167981585  11375.8742445
## std.dev           2.91997179     0.1296077100    106.6577435
## coef.var          0.10507558     1.2973748183   -194.0013299
##                   acceleration          alpha
## nbr.val      23249.000000000000  23249.0000000
## nbr.null         0.000000000000      0.0000000
## nbr.na           1.000000000000      1.0000000
```

```
## min             -0.448731938030   -179.9928659
## max              0.398179200030    179.9868528
## range            0.846911138060    359.9797188
## sum              0.144717934324 -63379.9253486
## median          -0.000198567182     -1.6292487
## mean             0.000006224695     -2.7261355
## SE.mean          0.000333611227      0.3218247
## CI.mean.0.95     0.000653900033      0.6307977
## var              0.002587531180   2407.9258607
## std.dev          0.050867781359     49.0706212
## coef.var      8171.931518599190    -18.0000665
```

```
right.sum<-stat.desc(pdr[,c(11:15)], basic=TRUE, desc=TRUE, norm=FALSE, p=0.95)
right.sum
```

```
##                    fish.temp         velocity          theta
## nbr.val       12520.00000000 12520.0000000000  12520.0000000
## nbr.null          0.00000000     0.0000000000      0.0000000
## nbr.na            0.00000000     0.0000000000      0.0000000
## min              20.70000000     0.0001584329   -179.9979144
## max              34.50000000     1.3114811299    179.9999480
## range            13.80000000     1.3113226970    359.9978624
## sum          353217.30000000  1420.7505676292 -15225.4645767
## median           28.70000000     0.0680217479      2.2786223
## mean             28.21224441     0.1134784798     -1.2160914
## SE.mean           0.02705855     0.0012063328      0.9686677
## CI.mean.0.95      0.05303890     0.0023645975      1.8987373
## var               9.16670444     0.0182195903  11747.7293157
## std.dev           3.02765659     0.1349799626    108.3869426
## coef.var          0.10731711     1.1894763028    -89.1272981
##                  acceleration           alpha
## nbr.val      12519.0000000000  12519.00000000
## nbr.null         0.0000000000      0.00000000
## nbr.na           1.0000000000      1.00000000
## min             -0.4487319380   -179.99286594
## max              0.3850388167    179.98685281
## range            0.8337707548    359.97971876
## sum              3.4989802501 -10090.63367515
## median          -0.0001475119     -0.05215509
## mean             0.0002794936     -0.80602554
## SE.mean          0.0005163069      0.45977995
## CI.mean.0.95     0.0010120408      0.90123927
## var              0.0033372253   2646.48652802
## std.dev          0.0577687227     51.44401353
## coef.var       206.6906890870    -63.82429747
```

```
left.sum<-stat.desc(pdl[,c(11:15)], basic=TRUE, desc=TRUE, norm=FALSE, p=0.95)
left.sum
```

```
##                    fish.temp         velocity         theta
## nbr.val       10730.00000000 10730.0000000000 10730.0000000
## nbr.null          0.00000000     0.0000000000     0.0000000
## nbr.na            0.00000000     0.0000000000     0.0000000
```

```
## min              19.40000000     0.0008225771  -179.9910928
## max              32.20000000     1.2356001412   179.9806344
## range            12.80000000     1.2347775641   359.9717272
## sum          292882.80000000   901.9238172050  2443.1164469
## median           27.50000000     0.0510308753     5.5575319
## mean             27.29569432     0.0840562737     0.2276903
## SE.mean           0.02613196     0.0011694681     1.0098267
## CI.mean.0.95      0.05122348     0.0022923739     1.9794473
## var               7.32729435     0.0146749441 10941.9174794
## std.dev           2.70689755     0.1211401837   104.6036208
## coef.var          0.09916940     1.4411795616   459.4119336
##                 acceleration          alpha
## nbr.val      10730.0000000000  10730.0000000
## nbr.null         0.0000000000      0.0000000
## nbr.na           0.0000000000      0.0000000
## min             -0.4391718919   -179.9059477
## max              0.3981792000    179.8211908
## range            0.8373510919    359.7271385
## sum             -3.3542623158 -53289.2916735
## median          -0.0002381838     -3.8027599
## mean            -0.0003126060     -4.9663832
## SE.mean          0.0003995433      0.4445476
## CI.mean.0.95     0.0007831789      0.8713955
## var              0.0017128822   2120.4898842
## std.dev          0.0413869806     46.0487772
## coef.var      -132.3934326679     -9.2720951
```

```
counterclockwise<-subset(alldata[,11:15], alpha>20)
# subset only those alphas that are >20 degrees into counterclockwise dataframe
clockwise<-subset(alldata[,11:15], alpha<(-20))
# subset only those alphas that are < -20 degrees into clockwise dataframe

n.counterclockwise<-nrow(counterclockwise)
n.clockwise<-nrow(clockwise)

bias.index<-(n.clockwise-n.counterclockwise)/(n.clockwise+n.counterclockwise)
# bias.index is the ratio of the difference in turns to the total counted turns
# -1 = all left, counterclockwise turns
# +1 = all right, clockwise turns
# 0 = no difference in turns
bias.strength<-abs(mean(clockwise$alpha)+mean(counterclockwise$alpha))

cat(paste("The bias index (-1 = counterclockwise, +1 = clockwise) is: \n",
          round(bias.index,4)), "\n")
```

```
## The bias index (-1 = counterclockwise, +1 = clockwise) is:
##  0.1071
```

```
#cat('\n')
cat(paste("The bias strength (|difference in mean ° counterclockwise vs. clockwise|) is: \n",
          round(bias.strength,4), "\n"))
```

```
## The bias strength (|difference in mean ° counterclockwise vs. clockwise|) is:
##  1.8208
```

```
#cat('\n')

#bias.index.c<-c(bias.index.c, bias.index)
```