SIA Scripts

10/18/2021

# import require R packages for Stable Isotope Analysis
library(coda)
library(rjags)

## Linked to JAGS 4.2.0

## Loaded modules: basemod,bugs

library(SIBER)

## Loading required package: spatstat

## Loading required package: spatstat.data

## Loading required package: spatstat.geom

## spatstat.geom 2.3-0

## Loading required package: spatstat.core

## Loading required package: nlme

## Loading required package: rpart

## spatstat.core 2.3-0

## Loading required package: spatstat.linnet

## spatstat.linnet 2.3-0

##
## spatstat 2.2-0 (nickname: 'That's not important right now')
## For an introduction to spatstat, type 'beginner'

library(tidyverse)

## Registered S3 method overwritten by 'cli':
## method from
## print.boxx spatstat.geom

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5 v purrr 0.3.4
## v tibble 3.1.5 v dplyr 1.0.7
## v tidyr 1.1.4 v stringr 1.4.0
## v readr 2.0.2 v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --
## x dplyr::collapse() masks nlme::collapse()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag() masks stats::lag()

library(rmarkdown)

# Set working directory
setwd("C:/Users/dlab/Desktop/Dr alys")

# Load the dataset
tedy<-read.csv('rtedi.csv')

tedy2<-read.csv('rtedi2.csv')
# check the number of rows and columns
nrow(tedy)

## [1] 103

ncol(tedy)

## [1] 4

# read the top five rows
head(tedy)

## iso1 iso2 group community
## 1 -15.57 10.26 1 1
## 2 -15.48 10.33 1 1
## 3 -16.29 10.35 1 1
## 4 -16.21 10.44 1 1
## 5 -16.61 10.36 1 1
## 6 -16.54 10.48 1 1

# read the bottom five rows
tail(tedy)

## iso1 iso2 group community
## 98 -18.58 10.73 8 2
## 99 -18.60 10.72 8 2
## 100 -18.98 11.08 8 2
## 101 -18.87 11.00 8 2
## 102 -17.45 11.56 8 2
## 103 -17.45 11.53 8 2

# Check group details
unique(tedy$group)

## [1] 1 2 3 4 5 6 7 8

# Check community details
unique(tedy$community)

## [1] 1 2

# Select groups for community one
siber.data1 = tedy%>%filter(group<=4)
head(siber.data1,3)

## iso1 iso2 group community
## 1 -15.57 10.26 1 1
## 2 -15.48 10.33 1 1
## 3 -16.29 10.35 1 1

nrow(siber.data1)

## [1] 56

# Select groups for community two
siber.data2 = tedy%>%filter(group>=5)
head(siber.data2,3)

## iso1 iso2 group community
## 1 -18.28 12.98 5 2
## 2 -18.23 13.04 5 2
## 3 -19.00 13.01 5 2

nrow(siber.data2)

## [1] 47

# create the siber object
siber.data <- createSiberObject(tedy)
head(siber.data)

## $original.data
## iso1 iso2 group community
## 1 -15.57 10.26 1 1
## 2 -15.48 10.33 1 1
## 3 -16.29 10.35 1 1
## 4 -16.21 10.44 1 1
## 5 -16.61 10.36 1 1
## 6 -16.54 10.48 1 1
## 7 -17.47 11.86 1 1
## 8 -17.46 11.84 1 1
## 9 -19.86 13.18 1 1
## 10 -19.59 12.94 1 1
## 11 -14.15 10.74 1 1
## 12 -13.11 10.16 1 1
## 13 -12.86 9.74 1 1
## 14 -13.95 10.43 1 1
## 15 -17.81 12.04 1 1
## 16 -17.85 12.26 1 1
## 17 -18.47 13.35 2 1
## 18 -18.43 13.32 2 1
## 19 -18.59 13.35 2 1
## 20 -18.48 13.30 2 1
## 21 -18.55 11.99 2 1
## 22 -18.51 11.89 2 1
## 23 -17.08 13.01 2 1
## 24 -17.14 13.00 2 1
## 25 -17.31 12.55 2 1
## 26 -17.33 12.55 2 1
## 27 -17.82 12.65 3 1
## 28 -17.84 12.60 3 1
## 29 -22.18 13.74 3 1
## 30 -22.21 13.79 3 1
## 31 -21.68 12.82 3 1
## 32 -21.68 12.87 3 1
## 33 -21.68 12.99 3 1
## 34 -21.71 12.93 3 1
## 35 -18.54 12.82 3 1
## 36 -18.57 12.79 3 1
## 37 -19.85 12.27 3 1
## 38 -19.86 12.35 3 1
## 39 -19.16 12.34 3 1
## 40 -19.15 12.63 3 1
## 41 -17.70 11.41 4 1
## 42 -17.72 11.37 4 1
## 43 -18.67 10.60 4 1
## 44 -18.55 10.66 4 1
## 45 -16.58 9.12 4 1
## 46 -16.59 9.00 4 1
## 47 -18.58 10.73 4 1
## 48 -18.60 10.72 4 1
## 49 -18.98 11.08 4 1
## 50 -18.87 11.00 4 1
## 51 -17.45 11.56 4 1
## 52 -17.45 11.53 4 1
## 53 -19.57 9.26 4 1
## 54 -19.55 9.30 4 1
## 55 -19.51 9.36 4 1
## 56 -19.42 9.42 4 1
## 57 -18.28 12.98 5 2
## 58 -18.23 13.04 5 2
## 59 -19.00 13.01 5 2
## 60 -18.90 13.00 5 2
## 61 -16.81 13.90 5 2
## 62 -16.83 13.65 5 2
## 63 -16.23 13.08 5 2
## 64 -16.05 12.99 5 2
## 65 -15.63 13.00 5 2
## 66 -15.49 13.23 5 2
## 67 -17.64 13.28 5 2
## 68 -17.72 13.32 5 2
## 69 -18.80 13.25 5 2
## 70 -18.89 13.30 5 2
## 71 -16.50 13.71 5 2
## 72 -16.46 13.80 5 2
## 73 -18.75 13.69 6 2
## 74 -17.73 14.17 6 2
## 75 -18.89 13.85 6 2
## 76 -17.58 13.02 6 2
## 77 -18.37 14.49 6 2
## 78 -18.51 12.99 6 2
## 79 -19.24 13.50 6 2
## 80 -19.44 13.40 6 2
## 81 -15.57 12.39 7 2
## 82 -15.54 12.51 7 2
## 83 -15.18 12.33 7 2
## 84 -15.17 12.48 7 2
## 85 -15.68 12.43 7 2
## 86 -15.61 12.50 7 2
## 87 -15.76 12.46 7 2
## 88 -15.78 12.45 7 2
## 89 -15.55 12.37 7 2
## 90 -15.56 12.36 7 2
## 91 -15.52 12.59 7 2
## 92 -17.70 11.41 8 2
## 93 -17.72 11.37 8 2
## 94 -18.67 10.60 8 2
## 95 -18.55 10.66 8 2
## 96 -16.58 9.12 8 2
## 97 -16.59 9.00 8 2
## 98 -18.58 10.73 8 2
## 99 -18.60 10.72 8 2
## 100 -18.98 11.08 8 2
## 101 -18.87 11.00 8 2
## 102 -17.45 11.56 8 2
## 103 -17.45 11.53 8 2
##
## $all.groups
## [1] "1" "2" "3" "4" "5" "6" "7" "8"
##
## $all.communities
## [1] "1" "2"
##
## $iso.summary
## iso1 iso2
## min -22.21000 9.0000
## max -12.86000 14.4900
## mean -17.75777 12.0265
## median -17.81000 12.4300
##
## $sample.sizes
## 1 2 3 4 5 6 7 8
## 1 16 10 14 16 NA NA NA NA
## 2 NA NA NA NA 16 8 11 12
##
## $raw.data
## $raw.data$`1`
## iso1 iso2 group community
## 1 -15.57 10.26 1 1
## 2 -15.48 10.33 1 1
## 3 -16.29 10.35 1 1
## 4 -16.21 10.44 1 1
## 5 -16.61 10.36 1 1
## 6 -16.54 10.48 1 1
## 7 -17.47 11.86 1 1
## 8 -17.46 11.84 1 1
## 9 -19.86 13.18 1 1
## 10 -19.59 12.94 1 1
## 11 -14.15 10.74 1 1
## 12 -13.11 10.16 1 1
## 13 -12.86 9.74 1 1
## 14 -13.95 10.43 1 1
## 15 -17.81 12.04 1 1
## 16 -17.85 12.26 1 1
## 17 -18.47 13.35 2 1
## 18 -18.43 13.32 2 1
## 19 -18.59 13.35 2 1
## 20 -18.48 13.30 2 1
## 21 -18.55 11.99 2 1
## 22 -18.51 11.89 2 1
## 23 -17.08 13.01 2 1
## 24 -17.14 13.00 2 1
## 25 -17.31 12.55 2 1
## 26 -17.33 12.55 2 1
## 27 -17.82 12.65 3 1
## 28 -17.84 12.60 3 1
## 29 -22.18 13.74 3 1
## 30 -22.21 13.79 3 1
## 31 -21.68 12.82 3 1
## 32 -21.68 12.87 3 1
## 33 -21.68 12.99 3 1
## 34 -21.71 12.93 3 1
## 35 -18.54 12.82 3 1
## 36 -18.57 12.79 3 1
## 37 -19.85 12.27 3 1
## 38 -19.86 12.35 3 1
## 39 -19.16 12.34 3 1
## 40 -19.15 12.63 3 1
## 41 -17.70 11.41 4 1
## 42 -17.72 11.37 4 1
## 43 -18.67 10.60 4 1
## 44 -18.55 10.66 4 1
## 45 -16.58 9.12 4 1
## 46 -16.59 9.00 4 1
## 47 -18.58 10.73 4 1
## 48 -18.60 10.72 4 1
## 49 -18.98 11.08 4 1
## 50 -18.87 11.00 4 1
## 51 -17.45 11.56 4 1
## 52 -17.45 11.53 4 1
## 53 -19.57 9.26 4 1
## 54 -19.55 9.30 4 1
## 55 -19.51 9.36 4 1
## 56 -19.42 9.42 4 1
##
## $raw.data$`2`
## iso1 iso2 group community
## 57 -18.28 12.98 5 2
## 58 -18.23 13.04 5 2
## 59 -19.00 13.01 5 2
## 60 -18.90 13.00 5 2
## 61 -16.81 13.90 5 2
## 62 -16.83 13.65 5 2
## 63 -16.23 13.08 5 2
## 64 -16.05 12.99 5 2
## 65 -15.63 13.00 5 2
## 66 -15.49 13.23 5 2
## 67 -17.64 13.28 5 2
## 68 -17.72 13.32 5 2
## 69 -18.80 13.25 5 2
## 70 -18.89 13.30 5 2
## 71 -16.50 13.71 5 2
## 72 -16.46 13.80 5 2
## 73 -18.75 13.69 6 2
## 74 -17.73 14.17 6 2
## 75 -18.89 13.85 6 2
## 76 -17.58 13.02 6 2
## 77 -18.37 14.49 6 2
## 78 -18.51 12.99 6 2
## 79 -19.24 13.50 6 2
## 80 -19.44 13.40 6 2
## 81 -15.57 12.39 7 2
## 82 -15.54 12.51 7 2
## 83 -15.18 12.33 7 2
## 84 -15.17 12.48 7 2
## 85 -15.68 12.43 7 2
## 86 -15.61 12.50 7 2
## 87 -15.76 12.46 7 2
## 88 -15.78 12.45 7 2
## 89 -15.55 12.37 7 2
## 90 -15.56 12.36 7 2
## 91 -15.52 12.59 7 2
## 92 -17.70 11.41 8 2
## 93 -17.72 11.37 8 2
## 94 -18.67 10.60 8 2
## 95 -18.55 10.66 8 2
## 96 -16.58 9.12 8 2
## 97 -16.59 9.00 8 2
## 98 -18.58 10.73 8 2
## 99 -18.60 10.72 8 2
## 100 -18.98 11.08 8 2
## 101 -18.87 11.00 8 2
## 102 -17.45 11.56 8 2
## 103 -17.45 11.53 8 2

# Create lists of plotting arguments to be passed onwards to each
# of the three plotting functions.
community.hulls.args <- list(col = 1, lty = 1, lwd = 1)
group.ellipses.args <- list(n = 103, p.interval = 0.95, lty = 1, lwd = 2)
group.hulls.args <- list(lty = 2, col = "grey20")

# plotting the graph
par(mfrow=c(1,1))

plotSiberObject(siber.data,
 ax.pad = 2,
 hulls = F, community.hulls.args = community.hulls.args,
 ellipses = T, group.ellipses.args = group.ellipses.args,
 group.hulls = T, group.hulls.args = group.hulls.args,
 bty = "L",
 iso.order = c(1,2),
 xlab = expression({delta}^13\*C~'\u2030'),
 ylab = expression({delta}^15\*N~'\u2030'),
 cex = 0.5
 )

# Calculate sumamry statistics for each group: TA, SEA.B and SEAc
group.ML <- groupMetricsML(siber.data)
print(group.ML)

## 1.1 1.2 1.3 1.4 2.5 2.6 2.7 2.8
## TA 5.716400 1.334400 2.823450 5.017700 2.087250 1.765600 0.09820000 2.480400
## SEA 3.445897 1.177668 1.825339 2.974570 1.176733 1.090238 0.04740530 1.986348
## SEAc 3.692032 1.324877 1.977451 3.187039 1.260785 1.271944 0.05267256 2.184983

# add a legend
legend("topright", colnames(group.ML), pch = c(1,1,1,2,2,2), col = c(1:3, 1:3), lty = 1)

 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

FIT THE BAYESIAN MODEL COMBINED COMMUNITIES

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

# Fit the Bayesian models
# options for running jags
parms <- list()
parms$n.iter <- 2 \* 10^4 # number of iterations to run the model for
parms$n.burnin <- 1 \* 10^3 # discard the first set of values
parms$n.thin <- 10 # thin the posterior by this many
parms$n.chains <- 2 # run this many chains

# define the priors
priors <- list()
priors$R <- 1 \* diag(2)
priors$k <- 2
priors$tau.mu <- 1.0E-3

# fit the ellipses which uses an Inverse Wishart prior
# on the covariance matrix Sigma, and a vague normal prior on the
# means. Fitting is via the JAGS method.
ellipses.posterior <- siberMVN(siber.data, parms, priors)

## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph information:
## Observed stochastic nodes: 16
## Unobserved stochastic nodes: 3
## Total graph size: 32
##
## Initializing model
##
## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph information:
## Observed stochastic nodes: 10
## Unobserved stochastic nodes: 3
## Total graph size: 26
##
## Initializing model
##
## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph information:
## Observed stochastic nodes: 14
## Unobserved stochastic nodes: 3
## Total graph size: 30
##
## Initializing model
##
## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph information:
## Observed stochastic nodes: 16
## Unobserved stochastic nodes: 3
## Total graph size: 32
##
## Initializing model
##
## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph information:
## Observed stochastic nodes: 16
## Unobserved stochastic nodes: 3
## Total graph size: 32
##
## Initializing model
##
## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph information:
## Observed stochastic nodes: 8
## Unobserved stochastic nodes: 3
## Total graph size: 24
##
## Initializing model
##
## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph information:
## Observed stochastic nodes: 11
## Unobserved stochastic nodes: 3
## Total graph size: 27
##
## Initializing model
##
## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph information:
## Observed stochastic nodes: 12
## Unobserved stochastic nodes: 3
## Total graph size: 28
##
## Initializing model

# The posterior estimates of the ellipses for each group can be used to
# calculate the SEA.B for each group.
SEA.B <- siberEllipses(ellipses.posterior)

siberDensityPlot(SEA.B, xticklabels = colnames(group.ML),
 xlab = c("Community | Group"),
 ylab = expression("Standard Ellipse Area " ('\u2030' ^2) ),
 bty = "L",
 las = 1,

 )

# Add red x's for the ML estimated SEA-c
points(1:ncol(SEA.B), group.ML[3,], col="red", pch = "x", lwd = 2)

 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

CRIDIBLE INTERVALS FOR GROUPS IN COMMUNITIES ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

# Calculate some credible intervals
cr.p <- c(0.95, 0.99) # vector of quantiles

# call to hdrcde:hdr using lapply()
SEA.B.credibles <- lapply(
 as.data.frame(SEA.B),
 function(x,...){tmp<-hdrcde::hdr(x)$hdr},
 prob = cr.p)
print(SEA.B.credibles)

## $V1
## [,1] [,2]
## 99% 2.084000 7.619114
## 95% 2.405014 6.527204
## 50% 3.220459 4.544779
##
## $V2
## [,1] [,2]
## 99% 0.4713776 2.753828
## 95% 0.5810819 2.195442
## 50% 0.8560175 1.357551
##
## $V3
## [,1] [,2]
## 99% 0.8696931 3.981792
## 95% 1.0481497 3.178694
## 50% 1.5043614 2.182630
##
## $V4
## [,1] [,2]
## 99% 1.462563 5.994316
## 95% 1.726928 4.856948
## 50% 2.362998 3.387530
##
## $V5
## [,1] [,2]
## 99% 0.5461779 2.330816
## 95% 0.6709878 1.903823
## 50% 0.9580818 1.341809
##
## $V6
## [,1] [,2] [,3] [,4]
## 99% 0.3597913 3.027767 3.232684 3.245001
## 95% 0.4911705 2.169664 NA NA
## 50% 0.7860913 1.282759 NA NA
##
## $V7
## [,1] [,2] [,3] [,4]
## 99% 0.0209145 0.10941064 0.111504 0.1153837
## 95% 0.0247083 0.08581950 NA NA
## 50% 0.0358831 0.05556732 NA NA
##
## $V8
## [,1] [,2]
## 99% 0.8080264 4.635264
## 95% 1.0281589 3.600880
## 50% 1.5603798 2.324593

# do similar to get the modes, taking care to pick up multimodal posterior
# distributions if present
SEA.B.modes <- lapply(
 as.data.frame(SEA.B),
 function(x,...){tmp<-hdrcde::hdr(x)$mode},
 prob = cr.p, all.modes=T)

print(SEA.B.modes)

## $V1
## [1] 3.811842
##
## $V2
## [1] 1.06825
##
## $V3
## [1] 1.761193
##
## $V4
## [1] 2.737442
##
## $V5
## [1] 1.137345
##
## $V6
## [1] 0.9921145
##
## $V7
## [1] 0.04190125
##
## $V8
## [1] 1.908908

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

COMPARING THE POSTERIOR DISTRIBUTIONS

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

## In order to test whether one group’s ellipse is smaller or larger than another,
## we can simply calculate the probability that its posterior distribution is smaller (or larger)

## G1.1.G1.2
Pg1.1\_g1.2 <- sum( SEA.B[,1] < SEA.B[,2] ) / nrow(SEA.B)
print(Pg1.1\_g1.2)

## [1] 0.00225

## G1.1.G1.3
Pg1.1\_g1.3 <- sum( SEA.B[,1] < SEA.B[,3] ) / nrow(SEA.B)
print(Pg1.1\_g1.3)

## [1] 0.02025

## G1.2.G1.3
Pg1.2\_g1.3 <- sum( SEA.B[,2] < SEA.B[,3] ) / nrow(SEA.B)
print(Pg1.2\_g1.3)

## [1] 0.8675

## And then for some of the other pairings:
## G2.5.G2.6
Pg2.5\_g2.6 <- sum( SEA.B[,5] < SEA.B[,6] ) / nrow(SEA.B)
print(Pg2.5\_g2.6)

## [1] 0.441

## G2.5.G2.7
Pg2.5\_g2.7 <- sum( SEA.B[,5] < SEA.B[,7] ) / nrow(SEA.B)
print(Pg2.5\_g2.7)

## [1] 0

## G2.6.G2.7
Pg2.6\_g2.7 <- sum( SEA.B[,6] < SEA.B[,7] ) / nrow(SEA.B)
print(Pg2.6\_g2.7)

## [1] 0

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

COMPARING THE POSTERIOR DISTRIBUTIONS OF SPECIES BETWEEN COMMUNITIES

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

## G1.1.G2.5
Pg1.1\_g2.5 <- sum( SEA.B[,1] < SEA.B[,5] ) / nrow(SEA.B)
print(Pg1.1\_g2.5)

## [1] 5e-04

## G1.2.G2.6
Pg1.2\_g2.6 <- sum( SEA.B[,2] < SEA.B[,6] ) / nrow(SEA.B)
print(Pg1.2\_g2.6)

## [1] 0.449

## G1.3.G2.7
Pg1.3\_g2.7 <- sum( SEA.B[,3] < SEA.B[,7] ) / nrow(SEA.B)
print(Pg1.3\_g2.7)

## [1] 0

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

VISUALIZE TWO COMMUNITIES

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

# create the siber object
siber.data\_2 <- createSiberObject(tedy2)

# Create plots for community 1 & 2
community.hulls.args <- list(col = 1, lty = 1, lwd = 1)
group.ellipses.args <- list(n = 103, p.interval = 0.95, lty = 1, lwd = 2)
group.hulls.args <- list(lty = 2, col = "grey20")

# plotting the graph
par(mfrow=c(1,1))

plotSiberObject(siber.data\_2,
 ax.pad = 2,
 hulls = F, community.hulls.args = community.hulls.args,
 ellipses = T, group.ellipses.args = group.ellipses.args,
 group.hulls = T, group.hulls.args = group.hulls.args,
 bty = "L",
 iso.order = c(1,2),
 xlab = expression({delta}^13\*C~'\u2030'),
 ylab = expression({delta}^15\*N~'\u2030'),
 cex = 0.5)
# Summary statistics for Pangani and Wami
group.summary <- groupMetricsML(siber.data\_2)
print(group.summary)

## 1.1 2.2
## TA 25.048400 14.624750
## SEA 7.543330 5.293235
## SEAc 7.683021 5.410863

legend(x = "topright", # Position
 legend = c("Pangani", "Wami"), # Legend texts
 lty = c(1, 1), # Line types
 col = c(1, 2), # Line colors
 lwd = 1) # Line width

 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Credible Intervals for pangani and Wami community

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

## Calculate Ellipse posterior
ellipses.posterior\_p\_w <- siberMVN(siber.data\_2, parms, priors)

## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph information:
## Observed stochastic nodes: 56
## Unobserved stochastic nodes: 3
## Total graph size: 72
##
## Initializing model
##
## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
## Graph information:
## Observed stochastic nodes: 47
## Unobserved stochastic nodes: 3
## Total graph size: 63
##
## Initializing model

# Calculate sumamry statistics for each group: TA, SEA and SEAc
group.ML\_p\_w <- groupMetricsML(siber.data\_2)
print(group.ML\_p\_w)

## 1.1 2.2
## TA 25.048400 14.624750
## SEA 7.543330 5.293235
## SEAc 7.683021 5.410863

# The posterior estimates of the ellipses for each group can be used to
# calculate the SEA.B for each group.
SEA.B\_p\_w <- siberEllipses(ellipses.posterior\_p\_w)

#SEA.B\_p\_w
# Calculate some credible intervals
cr.p <- c(0.95, 0.99)

# call to hdrcde:hdr using lapply()
SEA.B.credibles\_p\_w <- lapply(
 as.data.frame(SEA.B\_p\_w),
 function(x,...){tmp<-hdrcde::hdr(x)$hdr},
 prob = cr.p)
print(SEA.B.credibles\_p\_w)

## $V1
## [,1] [,2]
## 99% 5.148639 10.876203
## 95% 5.720670 9.795845
## 50% 6.850019 8.238879
##
## $V2
## [,1] [,2]
## 99% 3.561761 7.625513
## 95% 3.872034 7.011560
## 50% 4.713538 5.752615

# do similar to get the modes, taking care to pick up multimodal posterior
# distributions if present
SEA.B.modes\_p\_w <- lapply(
 as.data.frame(SEA.B\_p\_w),
 function(x,...){tmp<-hdrcde::hdr(x)$mode},
 prob = cr.p, all.modes=T)

print(SEA.B.modes\_p\_w)

## $V1
## [1] 7.545965
##
## $V2
## [1] 5.217279

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ Compare probability ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

## G1.1.G1.2
Pr1pr2 <- sum( SEA.B\_p\_w[1,] < SEA.B\_p\_w[2,] ) / nrow(SEA.B\_p\_w)
print(Pr1pr2)

## [1] 0.00025

# Calculate some credible intervals
cr.p <- c(0.95, 0.99)

# call to hdrcde:hdr using lapply()
SEA.B.credibles <- lapply(as.data.frame(SEA.B),function(x,...){tmp<-hdrcde::hdr(x)$hdr},prob = cr.p)

print(SEA.B.credibles)

## $V1
## [,1] [,2]
## 99% 2.083983 7.617883
## 95% 2.404767 6.527204
## 50% 3.220549 4.544831
##
## $V2
## [,1] [,2]
## 99% 0.4712412 2.753828
## 95% 0.5809993 2.195442
## 50% 0.8560688 1.357551
##
## $V3
## [,1] [,2]
## 99% 0.8702318 3.981793
## 95% 1.0482186 3.178694
## 50% 1.5043475 2.182600
##
## $V4
## [,1] [,2]
## 99% 1.462356 5.994316
## 95% 1.726801 4.856951
## 50% 2.363019 3.387530
##
## $V5
## [,1] [,2]
## 99% 0.5459598 2.330816
## 95% 0.6709204 1.903823
## 50% 0.9581544 1.341867
##
## $V6
## [,1] [,2] [,3] [,4]
## 99% 0.3607707 3.026647 3.23269 3.245172
## 95% 0.4916118 2.169672 NA NA
## 50% 0.7860092 1.282700 NA NA
##
## $V7
## [,1] [,2] [,3] [,4]
## 99% 0.02091260 0.10941398 0.1115031 0.1153844
## 95% 0.02470789 0.08581959 NA NA
## 50% 0.03588263 0.05556695 NA NA
##
## $V8
## [,1] [,2]
## 99% 0.8051193 4.635265
## 95% 1.0267631 3.600449
## 50% 1.5608784 2.324853

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

VISUALIZE PANGANI ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

# create the siber object
siber.data\_comm1 <- createSiberObject(siber.data1)

community.hulls.args <- list(col = 1, lty = 1, lwd = 1)
group.ellipses.args <- list(n = 56, p.interval = 0.95, lty = 1, lwd = 2)
group.hulls.args <- list(lty = 2, col = "grey20")
# plot siber.data1 for community 1
# plotting the graph
par(mfrow=c(1,1))

plotSiberObject(siber.data\_comm1,
 ax.pad = 2,
 hulls = F, community.hulls.args = community.hulls.args,
 ellipses = T, group.ellipses.args = group.ellipses.args,
 group.hulls = T, group.hulls.args = group.hulls.args,
 bty = "L",
 iso.order = c(1,2),
 xlab = expression({delta}^13\*C~'\u2030'),
 ylab = expression({delta}^15\*N~'\u2030'),
 cex = 0.5, main = 'Pangani'
 )

#plotGroupEllipses(siber.data, n = 103, p.interval = 0.95,lty = 1, lwd = 2)
# Calculate summary statistics for each group: TA, SEA and SEAc
group.ML2 <- groupMetricsML(siber.data\_comm1)
print(group.ML2)

## 1.1 1.2 1.3 1.4
## TA 5.716400 1.334400 2.823450 5.017700
## SEA 3.445897 1.177668 1.825339 2.974570
## SEAc 3.692032 1.324877 1.977451 3.187039

# add a legend
legend("topright", colnames(group.ML2),
 pch = c(1,1,1,2,2,2), col = c(1:3, 1:3), lty = 1)



++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

OVERLAP BETWEEN ELLIPSE FOR PANGANI

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

## Over lap for G1.1.G1.2
overlap.G1.1.G1.2 <- maxLikOverlap("1.1", "1.2", siber.data\_comm1, p = 0.95)
print(overlap.G1.1.G1.2)

## area.1 area.2 overlap
## 22.105833 7.932625 4.838896

## Over lap for G1.1.G1.3
overlap.G1.1.G1.3 <- maxLikOverlap("1.1", "1.3", siber.data\_comm1, p = 0.95)
print(overlap.G1.1.G1.3)

## area.1 area.2 overlap
## 22.105833 11.839873 7.393863

## Over lap for G1.2.G1.3
overlap.G1.2.G1.3 <- maxLikOverlap("1.2", "1.3", siber.data\_comm1, p = 0.95)
print(overlap.G1.2.G1.3)

## area.1 area.2 overlap
## 7.932625 11.839873 4.630358

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ PROPORTIONS ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

## Proportion for G1.1.G1.2\_area1
prop.of.G1.1.G1.2\_a1 <- as.numeric(overlap.G1.1.G1.2["overlap"] / overlap.G1.1.G1.2["area.1"])
print(prop.of.G1.1.G1.2\_a1)

## [1] 0.2188968

## Proportion for G1.1.G1.2\_area2
prop.of.G1.1.G1.2\_a2 <- as.numeric(overlap.G1.1.G1.2["overlap"] / overlap.G1.1.G1.2["area.2"])
print(prop.of.G1.1.G1.2\_a2)

## [1] 0.6099993

## Proportion for G1.1.G1.3\_area1
prop.of.G1.1.G1.3\_a1 <- as.numeric(overlap.G1.1.G1.3["overlap"] / overlap.G1.1.G1.3["area.1"])
print(prop.of.G1.1.G1.3\_a1)

## [1] 0.3344757

## Proportion for G1.1.G1.3\_area2
prop.of.G1.1.G1.3\_a2 <- as.numeric(overlap.G1.1.G1.3["overlap"] / overlap.G1.1.G1.3["area.2"])
print(prop.of.G1.1.G1.3\_a2)

## [1] 0.6244884

## Proportion for G1.2.G1.3\_area1
prop.of.G1.2.G1.3\_a1 <- as.numeric(overlap.G1.2.G1.3["overlap"] / overlap.G1.2.G1.3["area.1"])
print(prop.of.G1.2.G1.3\_a1)

## [1] 0.5837106

## Proportion for G1.2.G1.3\_area2
prop.of.G1.2.G1.3\_a2 <- as.numeric(overlap.G1.2.G1.3["overlap"] / overlap.G1.2.G1.3["area.2"])
print(prop.of.G1.2.G1.3\_a2)

## [1] 0.3910817

## Both for only G1.1 and G1.2
prop.of.both.G1.1.G1.2 <- as.numeric(overlap.G1.1.G1.2["overlap"] / (overlap.G1.1.G1.2["area.1"] + overlap.G1.1.G1.2["area.2"]))
print(prop.of.both.G1.1.G1.2)

## [1] 0.16109

## Both for only G1.1 and G1.3
prop.of.both.G1.1.G1.3 <- as.numeric(overlap.G1.1.G1.3["overlap"] / (overlap.G1.1.G1.3["area.1"] + overlap.G1.1.G1.3["area.2"]))
print(prop.of.both.G1.1.G1.3)

## [1] 0.2178144

## Both for only G1.2 and G1.3
prop.of.both.G1.2.G1.3 <- as.numeric(overlap.G1.2.G1.3["overlap"] / (overlap.G1.2.G1.3["area.1"] + overlap.G1.2.G1.3["area.2"]))
print(prop.of.both.G1.2.G1.3)

## [1] 0.2341817

## It can be a bit slow to calculate this overlap, so you may want to drop the number of draws if your computer is slow.

## G.1.1 & G1.2
bayes.overlap.G1.1.G1.2 <- bayesianOverlap("1.1", "1.2", ellipses.posterior, draws = 10, p.interval = 0.95,n = 103)
print(bayes.overlap.G1.1.G1.2)

## area1 area2 overlap
## 1 15.99946 10.799977 5.108027
## 2 21.66418 12.597212 6.516258
## 3 19.11112 5.177788 4.340788
## 4 18.54436 4.419311 2.203855
## 5 18.66580 6.037169 3.999379
## 6 34.49718 4.877500 4.227259
## 7 19.46303 7.228742 2.915721
## 8 27.14991 4.872520 4.124329
## 9 26.55946 6.904066 4.570436
## 10 38.14761 6.810520 6.810520

## It can be a bit slow to calculate this overlap, so you may want to drop the number of draws if your computer is slow.
## G.1.1 & G1.3
bayes.overlap.G1.1.G1.3 <- bayesianOverlap("1.1", "1.3", ellipses.posterior, draws = 10, p.interval = 0.95,n = 103)
print(bayes.overlap.G1.1.G1.3)

## area1 area2 overlap
## 1 15.99946 9.708127 3.630793
## 2 21.66418 7.057776 4.426275
## 3 19.11112 11.225948 6.496730
## 4 18.54436 16.008782 4.728687
## 5 18.66580 11.842823 5.900408
## 6 34.49718 13.262218 7.834154
## 7 19.46303 13.278728 6.408305
## 8 27.14991 15.418369 6.298370
## 9 26.55946 11.502706 8.098788
## 10 38.14761 11.586597 8.390977

## It can be a bit slow to calculate this overlap, so you may want to drop the number of draws if your computer is slow.
## G.1.2 & G1.3
bayes.overlap.G1.2.G1.3 <- bayesianOverlap("1.2", "1.3", ellipses.posterior, draws = 10, p.interval = 0.95,n = 103)
print(bayes.overlap.G1.2.G1.3)

## area1 area2 overlap
## 1 10.799977 9.708127 5.011294
## 2 12.597212 7.057776 3.368384
## 3 5.177788 11.225948 3.287585
## 4 4.419311 16.008782 3.590630
## 5 6.037169 11.842823 3.510449
## 6 4.877500 13.262218 1.348287
## 7 7.228742 13.278728 3.120998
## 8 4.872520 15.418369 4.377332
## 9 6.904066 11.502706 4.182087
## 10 6.810520 11.586597 4.082614

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

VISUALIZE PANGANI ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

# create the siber object
siber.data\_comm2 <- createSiberObject(siber.data2)

community.hulls.args <- list(col = 1, lty = 1, lwd = 1)
group.ellipses.args <- list(n = 47, p.interval = 0.95, lty = 1, lwd = 2)
group.hulls.args <- list(lty = 2, col = "grey20")
# plot siber.data1 for community 2
# plotting the graph
par(mfrow=c(1,1))

plotSiberObject(siber.data\_comm2,
 ax.pad = 2,
 hulls = F, community.hulls.args = community.hulls.args,
 ellipses = T, group.ellipses.args = group.ellipses.args,
 group.hulls = T, group.hulls.args = group.hulls.args,
 bty = "L",
 iso.order = c(1,2),
 xlab = expression({delta}^13\*C~'\u2030'),
 ylab = expression({delta}^15\*N~'\u2030'),
 cex = 0.5, main = 'Wami'
 )
#plotGroupEllipses(siber.data, n = 103, p.interval = 0.95,lty = 1, lwd = 2)
# Calculate summary statistics for each group: TA, SEA and SEAc
group.ML3 <- groupMetricsML(siber.data\_comm2)
print(group.ML3)

## 2.5 2.6 2.7 2.8
## TA 2.087250 1.765600 0.09820000 2.480400
## SEA 1.176733 1.090238 0.04740530 1.986348
## SEAc 1.260785 1.271944 0.05267256 2.184983

# add a legend
legend("topright", colnames(group.ML3),
 pch = c(1,1,1,2,2,2), col = c(1:3, 1:3), lty = 1)



++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

OVERLAP BETWEEN ELLIPSE FOR WAMI +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

## Over lap for G2.5.G2.6
overlap.G2.5.G2.6 <- maxLikOverlap("2.5", "2.6", siber.data\_comm2, p = 0.95)
overlap.G2.5.G2.6

## area.1 area.2 overlap
## 7.548880 7.615695 4.147287

## Over lap for G2.5.G2.7
overlap.G2.5.G2.7 <- maxLikOverlap("2.5", "2.7", siber.data\_comm2, p = 0.95)
overlap.G2.5.G2.7

## area.1 area.2 overlap
## 7.548880e+00 3.153739e-01 8.409343e-17

## Over lap for G2.6.G2.7
overlap.G2.6.G2.7 <- maxLikOverlap("2.6", "2.7", siber.data\_comm2, p = 0.95)
overlap.G2.6.G2.7

## area.1 area.2 overlap
## 7.6156945 0.3153739 0.0000000

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Proportion Overlap for Wami

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

## G2.5 & G2.6\_area1
prop.of.G2.5.G2.6\_a1 <- as.numeric(overlap.G2.5.G2.6["overlap"] / overlap.G2.5.G2.6["area.1"])
print(prop.of.G2.5.G2.6\_a1)

## [1] 0.549391

## G2.5 & G2.6\_area2
prop.of.G2.5.G2.6\_a2 <- as.numeric(overlap.G2.5.G2.6["overlap"] / overlap.G2.5.G2.6["area.2"])
print(prop.of.G2.5.G2.6\_a2)

## [1] 0.5445711

## G2.5 & G2.7\_area1
prop.of.G2.5.G2.7\_area1 <- as.numeric(overlap.G2.5.G2.7["overlap"] / overlap.G2.5.G2.7["area.1"])
print(prop.of.G2.5.G2.7\_area1)

## [1] 1.113985e-17

## G2.5 & G2.7\_area2
prop.of.G2.5.G2.7\_area2 <- as.numeric(overlap.G2.5.G2.7["overlap"] / overlap.G2.5.G2.7["area.2"])
print(prop.of.G2.5.G2.7\_area1)

## [1] 1.113985e-17

## G2.6 & G2.7\_area1
prop.of.G2.6.G2.7\_area1 <- as.numeric(overlap.G2.6.G2.7["overlap"] / overlap.G2.6.G2.7["area.1"])
print(prop.of.G2.6.G2.7\_area1)

## [1] 0

## G2.6 & G2.7\_area2
prop.of.G2.6.G2.7\_area2 <- as.numeric(overlap.G2.6.G2.7["overlap"] / overlap.G2.6.G2.7["area.2"])
print(prop.of.G2.6.G2.7\_area1)

## [1] 0

## Both for only 2.5 and 2.6
prop.of.both.G2.5.G2.6 <- as.numeric(overlap.G2.5.G2.6["overlap"] / (overlap.G2.5.G2.6["area.1"] + overlap.G2.5.G2.6["area.2"]))
print(prop.of.both.G2.5.G2.6)

## [1] 0.2734852

## Both for only 2.5 and 2.7
prop.of.both.G2.5.G2.7 <- as.numeric(overlap.G2.5.G2.7["overlap"] / (overlap.G2.5.G2.7["area.1"] + overlap.G2.5.G2.7["area.2"]))
print(prop.of.both.G2.5.G2.7)

## [1] 1.069312e-17

## Both for only 2.6 and 2.7
prop.of.both.G2.6.G2.7 <- as.numeric(overlap.G2.6.G2.7["overlap"] / (overlap.G2.6.G2.7["area.1"] + overlap.G2.6.G2.7["area.2"]))
print(prop.of.both.G2.6.G2.7)

## [1] 0

## It can be a bit slow to calculate this overlap, so you may want to drop the number of draws if your computer is slow.

## For G2.5 & G2.6
bayes.overlap.G2.5.G2.6 <- bayesianOverlap("2.5", "2.6", ellipses.posterior, draws = 10, p.interval = 0.95,n = 103)
print(bayes.overlap.G2.5.G2.6)

## area1 area2 overlap
## 1 8.132823 5.543770 3.097562
## 2 5.603204 6.196616 2.481541
## 3 10.517970 5.402817 5.021762
## 4 8.899977 9.933493 3.817390
## 5 6.628758 4.932589 2.666917
## 6 10.562052 5.064456 4.967223
## 7 7.335877 4.091430 1.771290
## 8 7.319380 20.741067 5.482838
## 9 5.342506 5.279799 3.214402
## 10 6.549197 8.803812 3.201293

## For G2.5 & G2.7
bayes.overlap.G2.5.G2.7 <- bayesianOverlap("2.5", "2.7", ellipses.posterior, draws = 10, p.interval = 0.95,n = 103)
print(bayes.overlap.G2.5.G2.7)

## area1 area2 overlap
## 1 8.132823 0.2405355 1.325152e-01
## 2 5.603204 0.4009854 1.438736e-16
## 3 10.517970 0.1839615 1.839615e-01
## 4 8.899977 0.1629781 6.044427e-17
## 5 6.628758 0.5466086 4.460345e-01
## 6 10.562052 0.2632691 4.839607e-17
## 7 7.335877 0.3585773 8.025633e-02
## 8 7.319380 0.2625385 1.717376e-16
## 9 5.342506 0.3078702 1.516257e-16
## 10 6.549197 0.2605784 6.364267e-17

## For G2.6 & G2.7
bayes.overlap.G2.6.G2.7 <- bayesianOverlap("2.6", "2.7", ellipses.posterior, draws = 10, p.interval = 0.95,n = 103)
print(bayes.overlap.G2.6.G2.7)

## area1 area2 overlap
## 1 5.543770 0.2405355 0.000000e+00
## 2 6.196616 0.4009854 0.000000e+00
## 3 5.402817 0.1839615 0.000000e+00
## 4 9.933493 0.1629781 0.000000e+00
## 5 4.932589 0.5466086 0.000000e+00
## 6 5.064456 0.2632691 0.000000e+00
## 7 4.091430 0.3585773 0.000000e+00
## 8 20.741067 0.2625385 1.257675e-16
## 9 5.279799 0.3078702 0.000000e+00
## 10 8.803812 0.2605784 0.000000e+00

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Overlap of groups from comm1 vs those of comm2

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

## Over lap for G1.1 & G2.5
overlap.G1.1.G2.5 <- maxLikOverlap("1.1", "2.5", siber.data, p = 0.95)
print(overlap.G1.1.G2.5)

## area.1 area.2 overlap
## 22.105833 7.548880 2.508701

## Over lap for G1.2 & G2.6
overlap.G1.2.G2.6 <- maxLikOverlap("1.2", "2.6", siber.data, p = 0.95)
print(overlap.G1.2.G2.6)

## area.1 area.2 overlap
## 7.932625 7.615695 4.545598

## Over lap for G1.3 & G2.7
overlap.G1.3.G2.7 <- maxLikOverlap("1.3", "2.7", siber.data, p = 0.95)
print(overlap.G1.3.G2.7)

## area.1 area.2 overlap
## 1.183987e+01 3.153739e-01 4.336809e-18

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Proportion Overlap for combined communities

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

# Calculate the proportion on overlap
## G1.1.G2.5
prop.of.G1.1.G2.5 <- as.numeric(overlap.G1.1.G2.5["overlap"] / overlap.G1.1.G2.5["area.1"])
print(prop.of.G1.1.G2.5)

## [1] 0.1134859

## G1.1.G2.5
prop.of.G1.1.G2.5 <- as.numeric(overlap.G1.1.G2.5["overlap"] / overlap.G1.1.G2.5["area.2"])
print(prop.of.G1.1.G2.5)

## [1] 0.3323276

## G1.2.G2.6\_area1
prop.of.G1.2.G2.6\_a1 <- as.numeric(overlap.G1.2.G2.6["overlap"] / overlap.G1.2.G2.6["area.1"])
print(prop.of.G1.2.G2.6\_a1)

## [1] 0.5730257

## G1.2.G2.6\_area2
prop.of.G1.2.G2.6\_a2 <- as.numeric(overlap.G1.2.G2.6["overlap"] / overlap.G1.2.G2.6["area.2"])
print(prop.of.G1.2.G2.6\_a2)

## [1] 0.5968724

## G1.3.G2.7\_area1
prop.of.G1.3.G2.7\_a1 <- as.numeric(overlap.G1.3.G2.7["overlap"] / overlap.G1.3.G2.7["area.1"])
print(prop.of.G1.3.G2.7\_a1)

## [1] 3.662885e-19

## G1.3.G2.7\_area2
prop.of.G1.3.G2.7\_a2 <- as.numeric(overlap.G1.3.G2.7["overlap"] / overlap.G1.3.G2.7["area.2"])
print(prop.of.G1.3.G2.7\_a2)

## [1] 1.375132e-17

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Proportion Overlap for combined communities (Overlap for both groups)

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

## Both for only G1.1 and G2.5
prop.of.both.G1.1.G2.5 <- as.numeric(overlap.G1.1.G2.5["overlap"] / (overlap.G1.1.G2.5["area.1"] + overlap.G1.1.G2.5["area.2"]))
print(prop.of.both.G1.1.G2.5)

## [1] 0.08459705

## Both for only G1.2 and G2.6
prop.of.both.G1.2.G2.6 <- as.numeric(overlap.G1.2.G2.6["overlap"] / (overlap.G1.2.G2.6["area.1"] + overlap.G1.2.G2.6["area.2"]))
print(prop.of.both.G1.2.G2.6)

## [1] 0.292353

## Both for only 1.3 and 2.7
prop.of.both.G1.3.G2.7 <- as.numeric(overlap.G1.3.G2.7["overlap"] / (overlap.G1.3.G2.7["area.1"] + overlap.G1.3.G2.7["area.2"]))
print(prop.of.both.G1.3.G2.7)

## [1] 3.567849e-19

## It can be a bit slow to calculate this overlap, so you may want to drop the number of draws if your computer is slow.

## G1.1.G2.5
bayes.overlap.G1.1.G2.5 <- bayesianOverlap("1.1", "2.5", ellipses.posterior, draws = 10, p.interval = 0.95,n = 103)
print(bayes.overlap.G1.1.G2.5)

## area1 area2 overlap
## 1 15.99946 8.132823 0.4864271
## 2 21.66418 5.603204 1.6754520
## 3 19.11112 10.517970 2.2217255
## 4 18.54436 8.899977 0.5155703
## 5 18.66580 6.628758 2.4301892
## 6 34.49718 10.562052 6.4201702
## 7 19.46303 7.335877 1.4009164
## 8 27.14991 7.319380 3.0694066
## 9 26.55946 5.342506 2.0785121
## 10 38.14761 6.549197 4.6819453

## It can be a bit slow to calculate this overlap, so you may want to drop the number of draws if your computer is slow.
## G1.2.G2.6
bayes.overlap.G1.2.G2.6 <- bayesianOverlap("1.2", "2.6", ellipses.posterior, draws = 10, p.interval = 0.95,n = 103)
print(bayes.overlap.G1.2.G2.6)

## area1 area2 overlap
## 1 10.799977 5.543770 3.2840043
## 2 12.597212 6.196616 6.0268640
## 3 5.177788 5.402817 2.2347026
## 4 4.419311 9.933493 0.5259257
## 5 6.037169 4.932589 2.9637807
## 6 4.877500 5.064456 4.3095539
## 7 7.228742 4.091430 1.1736916
## 8 4.872520 20.741067 3.3991928
## 9 6.904066 5.279799 3.8546208
## 10 6.810520 8.803812 3.1047448

## G1.3.G2.7
bayes.overlap.G1.3.G2.7 <- bayesianOverlap("1.3", "2.7", ellipses.posterior, draws = 10, p.interval = 0.95,n = 103)
print(bayes.overlap.G1.3.G2.7)

## area1 area2 overlap
## 1 9.708127 0.2405355 0.000000e+00
## 2 7.057776 0.4009854 0.000000e+00
## 3 11.225948 0.1839615 4.336809e-19
## 4 16.008782 0.1629781 0.000000e+00
## 5 11.842823 0.5466086 6.071532e-18
## 6 13.262218 0.2632691 6.938894e-18
## 7 13.278728 0.3585773 3.469447e-18
## 8 15.418369 0.2625385 2.625385e-01
## 9 11.502706 0.3078702 4.065758e-20
## 10 11.586597 0.2605784 0.000000e+00

# Calculate the various Layman metrics on each of the communities.
community.ML <- communityMetricsML(siber.data)
print(community.ML)

## 1 2
## dY\_range 2.44850000 2.9070833
## dX\_range 3.83723214 3.0255682
## TA 5.02208276 4.2417341
## CD 1.67566121 1.5300689
## MNND 2.16377106 1.7914920
## SDNND 0.01721845 0.6533672

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

EXTRA POSTERIOR MEANS AND VISUALIZE THE FIRST COMMUNITY

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

# extract the posterior means
mu.post <- extractPosteriorMeans(siber.data, ellipses.posterior)

# calculate the corresponding distribution of layman metrics
layman.B <- bayesianLayman(mu.post)

# --------------------------------------
# Visualise the first community
# --------------------------------------
siberDensityPlot(layman.B[[1]], xticklabels = colnames(layman.B[[1]]),
 bty="L", ylim = c(0,20))

# add the ML estimates (if you want). Extract the correct means
# from the appropriate array held within the overall array of means.
comm1.layman.ml <- laymanMetrics(siber.data$ML.mu[[1]][1,1,],
 siber.data$ML.mu[[1]][1,2,]
 )
points(1:6, comm1.layman.ml$metrics, col = "red", pch = "x", lwd = 2)

 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

VISUALIZE THE SECOND COMMUNITY

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

# --------------------------------------
# Visualise the second community
# --------------------------------------
siberDensityPlot(layman.B[[2]], xticklabels = colnames(layman.B[[2]]),
 bty="L", ylim = c(0,20))

# add the ML estimates. (if you want) Extract the correct means
# from the appropriate array held within the overall array of means.
comm2.layman.ml <- laymanMetrics(siber.data$ML.mu[[2]][1,1,],
 siber.data$ML.mu[[2]][1,2,]
)
points(1:6, comm2.layman.ml$metrics, col = "red", pch = "x", lwd = 2)



# --------------------------------------
# Alternatively, pull out TA from both and aggregate them into a
# single matrix using cbind() and plot them together on one graph.
# --------------------------------------

# go back to a 1x1 panel plot
par(mfrow=c(1,1))

siberDensityPlot(cbind(layman.B[[1]][,"TA"], layman.B[[2]][,"TA"]),
 xticklabels = c("Pangani", "Wami"),
 bty="L", ylim = c(0,20),
 las = 1,
 ylab = "TA - Convex Hull Area",
 xlab = "")



TA1\_lt\_TA2 <- sum(layman.B[[1]][,"TA"] < layman.B[[2]][,"TA"]) / length(layman.B[[1]][,"TA"])
print(TA1\_lt\_TA2)

## [1] 0.32975

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

## Overlap Between Ellipses

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

overlap.G1.2.G1.3 <- maxLikOverlap("1.2", "1.3", siber.data, p = 0.95, n =)
overlap.G1.2.G1.3

## area.1 area.2 overlap
## 7.932625 11.839873 4.630358

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++