Supplementary Material for SHI, PEI-JIAN, ZI-HUA ZHAO, HARDEV S. SANDHU, CANG HUI, XING-YUAN MEN, FENG GE, AND BAI-LIAN LI−**An Optimization Approach to Estimating Ground-Dwelling Arthropod Densities by the Two-Circle Method.** Florida Entomologist 97(2) (June, 2014) at <http://purl.fcla.edu/fcla/entomologist/browse>

**PART 1. USAGE OF ATTACHED R FUNCTIONS**

The reader should be aware that in Part 2 of this supplementary material details are provided concerning 3 R functions for estimating the parameters and their confidence intervals in the TCM. The first of these R functions is *optim.tcm*, which was used to estimate the parameters of equation 1 in the text based on the *optim* function; the second is the *bca.tcm* function, which was used to calculate the standard error and confidence interval of estimated *r* or *D*; and the third is the *sample.validity* function, which was used to check the validity of sample size for each distance class and number of distance classes for the estimation of population density. The *optim.tcm* function is actually a candidate procedure of the *tcm* function in Zhao et al. (2013). In addition, we also provided the *tc.points* function, which can be found in Zhao et al. (2013).

How to Make Use of the TCM:

Firstly, the users need install the R statistical software (<http://www.r-project.org/>), and then two additional R packages, "Hmisc" package (<http://cran.r-project.org/web/packages/Hmisc/index.html>) and the "bootstrap" package (<http://cran.r-project.org/web/packages/bootstrap/index.html>) are need to be installed.

 Secondly, the users need to copy all R scripts from S2 to R Console to use these functions.

(i) The following R scripts show the usage of the *optim.tcm* function:

dis <- c(0, 1.5, 2, 2.5)

N <- c(5.733, 9.733, 10.600, 11.267)

optim.tcm( dis, N, r.range = seq(1, 2, len = 5), D.range = seq(1, 2, len = 5), fig.opt = TRUE )

Here, dis is an array used to store distances between paired pitfall traps; N is an array used to store the observed number of arthropods caught in paired traps. Values can be replaced by real data from the users. r.range and D.range represent the initial value ranges of effective trapping radius (*r*) and population density (*D*) for the optimization. fig.opt is an option to choose to plot the figure.

(ii) The following R scripts show the usage of the *bca.tcm* function:

dis <- c(0, 1.5, 2, 2.5)

N <- c(5.733, 9.733, 10.600, 11.267)

bca.tcm( dis, N, nboot = 2000, goal.par = "D", CI = 0.95, r.range = seq(1, 2, len = 5), D.range = seq(1, 2, len = 5) )

Here, dis is an array used to store distances between paired pitfall traps; N is an array used to store the observed number of arthropods caught in paired traps. nboot is the number of bootstrap replications for performing BC*a* method. goal.par is a parameter option of "D" representing the population density, and of "r" representing the effective trapping radius. CI represents the goal confidence interval, e.g., 0.95 denotes 95% CI. r.range and D.range represent the initial value ranges of effective trapping radius (*r*) and population density (*D*) for the optimization. By carrying out the above command, we can obtain the standard error of a goal parameter and its 95% CI endpoints.

(iii) The following R scripts show the usage of the *sample.validity* function:

D.real <- c(1, 1.5, 2)

dis.design <- seq(0, 3, len = 4)

sample.validity( CV = 0.15, n.given = 15, dis.given = dis.design, r.given = 1.5, D.given = D.real, nboot = 2000, CI = 0.95, fig.opt = TRUE )

Here, CV represents the coefficient of variation in population density *D*. n.given represents the number of replications per distance class. dis.given is an array of given distances between paired pitfall traps. r.given represents a given effective trapping radius, which is only permitted to be a single number. D.given is an array of given population densities, which can be a single number. nboot is the number of bootstrap replications for performing BC*a* method. CI represents the goal confidence interval, e.g., 0.95 denotes 95% CI. fig.opt is an option to choose to plot the figure of *y* = *x* for comparing the given and predicted densities.

 In the above three functions, the first two are generally applied to the data, whereas the third is only used to check the validity of sample size and distance gradient for simulation studies.

**PART 2**. **R FUNCTIONS FOR ESTIMATING THE PARAMETERS AND THEIR CONFIDENCE INTERVALS IN THE TCM**

optim.tcm <- function(Distance, N, r.range = seq(1, 2, len = 5), D.range = seq(1, 2, len = 5), fig.opt = TRUE) {

 # optim.tcm function

 # Codes were developed in early February 2013

 # ( email: peijianshi@gmail.com )

 arcccos <- function(x){

 x[ which( x > 1) ] <- 1

 return(acos(x))

 }

 tc.points <- function(P, d){

 r <- P[1]

 D <- P[2]

 (2 \* pi \* r^2 - 2 \* (r^2 \* arcccos( d/(2\*r) ) - d / 2 \* r \* sin(arcccos( d/(2\*r) ))) ) \* D

 }

 tc.points.convergence <- function(x){

 r <- x[1]

 D <- x[2]

 expected.value <- (2 \* pi \* r^2 - 2 \* (r^2 \* arcccos( Distance/(2\*r) )

 - Distance / 2 \* r \* sin(arcccos( Distance/(2\*r)))) ) \* D

 abs(sum( (N - expected.value)^2 ))

 }

 KeyMatrix <- matrix(NA, length(r.range)\*length(D.range), 3)

 counter <-0

 for (i in r.range){

 for (j in D.range){

 counter <- counter + 1

 Result <- optim(c(i, j), tc.points.convergence)

 goal.r <- Result$par[1]

 goal.D <- Result$par[2]

 rss <- Result$value

 KeyMatrix[counter, ] <- c(goal.r, goal.D, rss)

 }

 }

 M <- KeyMatrix[KeyMatrix[, 3] == min(KeyMatrix[, 3])]

 M <- matrix(M, ncol = ncol(KeyMatrix))

 if (nrow(M)>1){

 index1 <- sample(1:nrow(M), replace=TRUE)[1]

 M <- M[index1,]

 }

 M <- M

 rss <- M[3]

 Rsquare <- 1 - sum((tc.points(M[1:2], Distance) - N)^2)/sum((N - mean(N))^2)

 if (fig.opt == "TRUE" | fig.opt == "T"){

 library("splines")

 library("survival")

 library("Hmisc")

 Dis.Ave <- tapply(Distance, Distance, mean)

 N.Ave <- tapply(N, Distance, mean)

 N.Sd <- tapply(N, Distance, sd)

 N.Sd[is.na(N.Sd)] <- 0

 x.fit <- seq(0, (1 + 1/20) \* max(Distance), len = 1000)

 y.fit <- tc.points(M[1:2], x.fit)

 #tiff(file = "Fitting.tif", width=72, height=72, units='mm',res=1200,compression='lzw',pointsize=5)

 par(mar=c(5,5,1,1))

 plot(x.fit, y.fit, type = "l", ylim = c( 0, max(N.Ave + N.Sd) ), col="grey40", lwd = 2, cex.lab = 1.5,

 cex.axis = 1.5, xlab = expression(bold("Distance apart (m)")),

 ylab=expression(bold("Number of individuals caught in paired traps")) )

 errbar(Dis.Ave, N.Ave, N.Ave - N.Sd, N.Ave + N.Sd, add=TRUE, pch=16, cex = 2, lwd=1)

 #dev.off()

 }

 list(r = M[1], D = M[2], RSS = rss, Rsquare = Rsquare)

}

bca.tcm <- function( Distance, N, nboot=2000, goal.par="D", CI=0.95, r.range=seq(1, 2, len=5), D.range=seq(1, 2, len=5) ){

 # bca.tcm function

 # Codes were developed in late June 2013

 # ( email: peijianshi@gmail.com )

 library(bootstrap)

 alpha0 <- c( (1-CI)/2, 1-(1-CI)/2 )

 xdata <- rbind(Distance, N)

 xdata <- t(xdata)

 xdata <- matrix(xdata, ncol=2)

 theta <- function(x, xdata){

 res1 <- optim.tcm( xdata[x, 1], xdata[x, 2], r.range = r.range, D.range = D.range, fig.opt = F )

 if (goal.par == "D")

 return(res1$D)

 if (goal.par == "r")

 return(res1$r)

 }

 res2 <- jackknife(1:nrow(xdata), theta, xdata)

 res3 <- bcanon(1:nrow(xdata), nboot, theta, xdata, alpha = c(alpha0[1], alpha0[2]))

 if (goal.par == "D")

 temp <- list(D.se = res2$jack.se, D.LCI = res3$confpoints[1,2][[1]], D.UCI = res3$confpoints[2,2][[1]])

 if (goal.par == "r")

 temp <- list(r.se = res2$jack.se, r.LCI = res3$confpoints[1,2][[1]], r.UCI = res3$confpoints[2,2][[1]])

 temp

 }

tc.points <- function(P, d){

 # tc.points function

 # Codes were developed in late October 2012

 # This function can be found in Zhao et al. (2013)

 # ( email: peijianshi@gmail.com )

 arcccos <- function(x){

 x[ which( x > 1) ] <- 1

 return(acos(x))

 }

 r <- P[1]

 D <- P[2]

 (2 \* pi \* r^2 - 2 \* (r^2 \* arcccos( d/(2\*r) ) - d / 2 \* r \* sin(arcccos( d/(2\*r) ))) ) \* D

}

sample.validity <- function( CV = 0.10, n.given = 30, dis.given = seq(0,3, by=1), r.given = 1.5, D.given = seq(1, 2, by=0.01), nboot=2000, CI=0.95, fig.opt = TRUE ){

 # sample.validity function

 # Codes were developed in late June 2013

 # ( email: peijianshi@gmail.com )

 library(bootstrap)

 alpha0 <- c( (1 - CI)/2, 1-(1-CI)/2 )

 if( length(r.given)>1 ){

 print("r.given should be only a single value.")

 break

 }

 if(nboot > 0 ){

 if( length(D.given)>1 )

 print( paste("There are totally ", length(D.given), " groups of circles.") )

 if( length(D.given) == 1 )

 print( paste("There is ", length(D.given), " group of circles.") )

 cat("\n")

 }

 Sd.D.arr <- CV \* D.given

 D.pre <- NA

 D.sd <- NA

 D.LCI <- NA

 D.UCI <- NA

 for (i in 1:length(D.given)){

 x <- rep(dis.given, n.given)

 y <- c()

 D.prac.temp <- c()

 for (j in 1:n.given){

 D.temp <- rnorm( 1, mean = D.given[i], sd = Sd.D.arr[i] )

 y <- c( y, tc.points( c(r.given, D.temp), dis.given ) )

 }

 theta <- function(x, xdata){

 res <- optim.tcm( xdata[x, 1], xdata[x, 2], r.range = r.given, D.range = D.given[i], fig.opt = F )

 return(res$D)

 }

 res2 <- optim.tcm( x, y, r.range = r.given, D.range = D.given[i], fig.opt = F )

 D.pre[i] <- res2$D

 if(nboot > 0 ){

 print( paste("It is Group ", i, " with the number of circles = ", nboot + n.given\*length(dis.given)) )

 cat("\n")

 xdata <- rbind(x, y)

 xdata <- t(xdata)

 xdata <- matrix(xdata, ncol=2)

 res3 <- jackknife(1:nrow(xdata), theta, xdata)

 res4 <- bcanon(1:nrow(xdata), nboot, theta, xdata, alpha=c(alpha0[1], alpha0[2]))

 D.sd[i] <- res3$jack.se

 D.LCI[i] <- res4$confpoints[1,2]

 D.UCI[i] <- res4$confpoints[2,2]

 }

 }

 if( length(D.given) > 2 )

 R2 <- cor(D.given, D.pre)^2

 if( length(D.given) < 3 )

 R2 <- NA

 if (fig.opt == "TRUE" | fig.opt == "T"){

 #tiff(file = "Comparison.tif", width=72, height=72, units='mm',res=600,compression='lzw',pointsize=5)

 par(mar=c(5,5,1,1))

 plot(D.given, D.pre, xlab=expression( bold( paste("Known density ( ", m^{"-2"}, ")") ) ), ylab=expression( bold( paste("Estimated density ( ", m^{"-2"}, ")") ) ),

 cex.lab=1.5, cex.axis=1.5, cex=1.5, xlim=c(0.5,2.5), ylim=c(0.5,2.5))

 abline(0, 1, col=1)

 #dev.off()

 }

 list(D.pre = D.pre, D.sd = D.sd, Rsquare=R2, D.LCI=D.LCI, D.UCI=D.UCI)

 }