Supplemental Text 2: Code for Establishing Probable Range of Overlap

#Install packages and load libraries

install.packages("Bchron")

library("Bchron")

# Set working directory

setwd("D:/Folder")

# Define sample space

#Oldest atlatl calendar date (CYBP) [USER ADDED VALUE]

OldAtlDate <- 9539

#Youngest bow calendar date (CYBP) [USER ADDED VALUE]

YoungBowDate <- 0

#Radiocarbon date standard error [USER ADDED VALUE]

StandardRadError <- 50

#How much overlap should we cycle through? [USER ADDED VALUE]

MinOverlap <- 0

MaxOverlap <- 800

OverlapInterval <- 2

#Timing of transition (CYBP) [USER ADDED VALUE]

TransDateSampUnAtl <- 1171

TransDateSampUnBow <- 1169

#Create matrix to store values

ColumnsNeeded <- (MaxOverlap - MinOverlap) / OverlapInterval + 1

AllColNames <- seq(from=MinOverlap, to=MaxOverlap, by=2)

Stat.matrix <- matrix(nrow = 1000, ncol = ColumnsNeeded)

colnames(Stat.matrix) <- AllColNames

RunNum <- 0

#Cycle through different amounts of overlap

for(a in seq(from=MinOverlap, to=MaxOverlap, by=OverlapInterval)) {

 RunNum <- RunNum + 1

 TransDateSampUnAtl <- TransDateSampUnAtl - 0.5\*OverlapInterval

 TransDateSampUnBow <- TransDateSampUnBow + 0.5\*OverlapInterval

 #Prior Ranges (CYBP)

 AtlRange <- c(OldAtlDate:TransDateSampUnAtl)

 BowRange <- c(TransDateSampUnBow:YoungBowDate)

 #Sample sizes:

 #Add sample size iterator here [USER ADDED VALUE]

 g <- 1

 # Interval to iterate through the sequence of sample sizes [USER ADDED VALUE]

 j <- 1

 # Iterate through error calculation h times for sample size = {1, 2, 3, ..., g}

 h <- 1000 #number of times to repeat error stat calculations [USER ADDED VALUE]

 RepVec <- 1:h

 #Create an empty vector to store overlapping error values throughout iterations

 Overlap.h.Iterations <- rep(NA, h)

 #Set up Intcal13 calibration curve

 Lab\_Error <- StandardRadError

 eps = 1e-05

 pathToCalCurves = system.file("data", package = "Bchron")

 calCurveFile <- paste(pathToCalCurves, "/", "intcal13", ".txt.gz", sep = "")

 calCurve = calBP = c14BP = calSd = ageGrid = list()

 calCurve <- as.matrix(read.table(calCurveFile))

 #Calendar years for plotting calibration curve, x values

 calBP <- calCurve[, 1]

 #Radiocarbon years for plotting calibration curve, y values

 c14BP <- calCurve[, 2]

 # Error

 calSd <- calCurve[, 3]

 #Create an age grid that runs from through all radiocarbon years in increments of 1year

 ageGrid <- seq(min(c14BP), max(c14BP), by = 1)

 # Linear interpolation of cal curve

 Intcal13 <- approx(c14BP, calBP, xout = ageGrid)$y

 Intcal13Sd <- approx(c14BP, calSd, xout = ageGrid)$y

 RadVsCal <- data.frame(RCYBP = ageGrid, CalBP=Intcal13)

 #Error

 tau <- sqrt(Lab\_Error^2 + Intcal13Sd)

 #BEGIN ANALYSIS

 # Cycle through sample size (from 1 to g)

 for(n in seq(from=1, to=g, by=j)) {

 # Cycle through error stat calculations for sample size = {1, 2, 3, .., g}

 for(p in RepVec) {

 # Update the user on the progress of runs

 Update <- paste(c(" iteration ", toString(p), " out of ", toString(h), ": ", "for overlap = ", toString(a)),

 collapse = "")

 print(Update, quote = FALSE)

 #Sample atlatl calendar dates (CYBP)

 AtlSample <- sample(AtlRange, size = 42, replace = FALSE, prob = NULL)

 #prob = vector input changing prob of selecting each yr. Could be used to correct for taphonomy.

 #Sample bow calendar dates (CYBP)

 BowSample <- sample(BowRange, size = 29, replace = FALSE, prob = NULL)

 #REVERSE CALIBRATE Samples

 #Atl Samples

 #Create an empty vector to store all R\_Simulate dates

 R\_SimsAtl <- rep(NA, length(AtlSample))

 iterator <- 0

 for(s in AtlSample) {

 iterator <- iterator + 1

 # Calculate density distribution function for the t distribution with degrees of freedom

 dens <- dt((s - Intcal13)/tau, df = 100)

 dens <- dens/sum(dens)

 # Output:

 # ageGrid = probable RCYBP ages for given R\_Simulate date

 # densities = probability density at each position

 out = list(R\_simulate = s, Lab\_Error = Lab\_Error,

 calCurves = "Intcal13", ageGrid = ageGrid[dens > eps], densities = dens[dens > eps])

 names(out) = c("R\_Simulate ", "Lab Error", "Cal Curve", "RCYBP", "Relative Probability")

 class(out) = "BchronCalibratedDates"

 RCYBP\_Sim <- sample(out$RCYBP, 1, prob = out$"Relative Probability")

 R\_SimsAtl[iterator] <- RCYBP\_Sim

 }

 AtlAges <- R\_SimsAtl

 #Bow Samples

 #Create an empty vector to store all R\_Simulate dates

 R\_SimsBow <- rep(NA, length(BowSample))

 iterator <- 0

 for(s in BowSample) {

 iterator <- iterator + 1

 #Calculate density distribution function for the t distribution with degrees of freedom

 dens <- dt((s - Intcal13)/tau, df = 100)

 dens <- dens/sum(dens)

 # Output:

 # ageGrid = probable RCYBP ages for given R\_Simulate date

 # densities = probability density at each position

 out = list(R\_simulate = s, Lab\_Error = Lab\_Error,

 calCurves = "Intcal13", ageGrid = ageGrid[dens > eps], densities = dens[dens > eps])

 names(out) = c("R\_Simulate ", "Lab Error", "Cal Curve", "RCYBP", "Relative Probability")

 class(out) = "BchronCalibratedDates"

 RCYBP\_Sim <- sample(out$RCYBP, 1, prob = out$"Relative Probability")

 R\_SimsBow[iterator] <- RCYBP\_Sim

 }

 BowAges <- R\_SimsBow

 #Re-Calibrate Dates

 #Vectors of radiocarbon standard error

 AtlSds <- mat.or.vec(42, 1)

 AtlSds[1:42] <- StandardRadError

 BowSds <- mat.or.vec(29, 1)

 BowSds[1:29] <- StandardRadError

 #Vector indicating calibration curve

 calCurvesBow <- toString(mat.or.vec(29, 1))

 calCurvesAtl <- toString(mat.or.vec(42, 1))

 calCurvesBow[1:29] <- "intcal13"

 calCurvesAtl[1:42] <- "intcal13"

 #Re-calibrate the dates

 AtlCalib <- BchronCalibrate(ages=AtlAges, ageSds=AtlSds, calCurves=calCurvesAtl)

 BowCalib <- BchronCalibrate(ages=BowAges, ageSds=BowSds, calCurves=calCurvesBow)

 #Determine range of summed probability distribution

 # Create a vector of names of AtlDens$calAges$daten$ageGrid

 ageGridNamesAtl <- toString(mat.or.vec(42,1))

 for (i in seq.int(from=1, to=42)) {

 dateNum <- i

 dateNumStr <- toString(dateNum)

 dateNumTitle <- paste(c("AtlCalib$date", dateNumStr, "$ageGrid"), collapse="")

 ageGridNamesAtl[dateNum] <- dateNumTitle

 }

 SumAtlDates <- rep(NA, 1)

 for (i in seq.int(from=1, to=42)){

 CurrentIteration <- eval(parse(text=ageGridNamesAtl[i]))

 SumAtlDates <- append(CurrentIteration, SumAtlDates)

 }

 #Create a vector of names of BowDens$calAges$daten$ageGrid

 ageGridNamesBow <- toString(mat.or.vec(29,1))

 for (i in seq.int(from=1, to=29)) {

 dateNum <- i

 dateNumStr <- toString(dateNum)

 dateNumTitle <- paste(c("BowCalib$date", dateNumStr, "$ageGrid"), collapse="")

 ageGridNamesBow[dateNum] <- dateNumTitle

 }

 SumBowDates <- rep(NA, 1)

 for (i in seq.int(from=1, to=29)){

 CurrentIteration <- eval(parse(text=ageGridNamesBow[i]))

 SumBowDates <- append(CurrentIteration, SumBowDates)

 }

 #Find the min and max of all age values to give range of summed p distribution

 SumAtlRange <- range(SumAtlDates, na.rm=TRUE)

 SumAtlRangeString <- toString(SumAtlRange)

 print(c("Re-Calibrated Atlatl Date Range (CYBP): ", SumAtlRangeString), quote = FALSE)

 SumBowRange <- range(SumBowDates, na.rm=TRUE)

 SumBowRangeString<- toString(SumBowRange)

 print(c("Re-Calibrated Bow Date Range (CYBP): ", SumBowRangeString), quote = FALSE)

 #Calculate overlap/separation of summed probability distributions

 Overlap <- SumBowRange[2] - SumAtlRange[1]

 CurrentResult <- paste(c("Error: ", toString(Overlap), " calendar years"), collapse = "")

 print(CurrentResult, quote = FALSE)

 #Store overlap/separation for this n, repeat h times for this n

 Overlap.h.Iterations[p] <- Overlap

 }

 #Save data

 Stat.matrix[, RunNum] <- Overlap.h.Iterations

 write.csv(Stat.matrix, file = "Overlap.csv", na="")

 Overlap.h.Iterations <- rep(NA, h)

 }

}

#Show that analysis is complete

print(head(Stat.matrix))

print("...")

print(tail(Stat.matrix))