# R code for Maiasaura growth curves.

# Publication:

# Woodward, H.N., Freedman, E.A., Farlow, J.O., and Horner, J.R. 2015. Maiasaura, an exemplar model for dinosaur population biology: a statistically robust assessment of growth dynamics and survivorship. Paleobiology.

# This code has been annotated and slightly modified from the code of Cooper et al. (2008), kindly provided by Andrew Lee (personal communication, 2013).

# If citing this code in the future, please cite both this paper (Woodward et al., 2015) and Cooper et al. (2008).

# Model equation used in Cooper et al. (2008) has here been updated to the model equation used in Lee and O'Connor (2013).

# Data files should be saved as .csv in the following format: (Maiasaura tibia 33 used as example.)

# lagged present

# 2.5 17.76

# 17.76 22.16

# 22.16 25.13

# 25.13 25.86

# 25.86 26.58

# In the above example, the first column represents the tibial circumference at the beginning of year 1 (hatching), 2, 3, 4, and 5.

# The second column has these values shifted to represent the tibial circumference at the end of each year (LAG measurements).

# Each bone used must have 3 or more lines of arrested growth (LAGs).

# If you would like to save this code as a .R file, copy and paste this text into a new window in R, then save.

################## Instructions - detailed ##################

# Section 1 fits multiple growth curve models to size measurement data from a single bone (longitudinal sampling).

# Section 1 must be re-run separately for each bone.

# Tabulate the results for each model type and each bone in Excel or other spreadsheet.

# For each model type, record the results of A, K, residual standard error, and AICc for each bone.

# Calculate the mean value for A, K, residual standard error, and AICc.

# Calculate I from these mean values using equation 5 of Lee and O'Connor (2013):

# I = -(1/K)\*ln(-((m-1)\*A^(1-m))/(A^(1-m)-Ao^(1-m))) where Ao is the hatchling size.

# Select model type with lowest AICc value.

# Using the equation y=A\*((1+exp(-K)\*((x/A)^(1-m)-1))^(1/(1-m)), insert the shape parameter m and mean values of A and K, and the value x of the size measurement at the beginning of the time interval.

# Example: tibial circumference at hatching = 2.5 cm.

# Use x = 2.5 in the equation to calculate the size at the end of year 1.

# Use the result to calculate size at the end of year 2.

# Repeat for the number of years preserved in the bone with the most lines of arrested growth (LAGs).

# This produces the age vs size data for the mean growth line.

# Use this mean data in Section 2 to calculate 95% confidence intervals around the mean line.

##### Annotations are provided throughout the code to help indicate where model names or values need to be changed each time the analysis is run. Instructions are indicated with 5 "#" symbols. #####

# A "#" symbol at the start of a line deactivates that line of code.

# Remove a "#" to activate that line of code.

################## Instructions - summary ##################

# After you have made the appropriate modifications for your data, copy and paste section 1 into R and press the "enter" key.

# Repeat for each bone.

# Tabulate results in Excel or similar program and do mean calculations described above.

# Make appropriate modifications to Section 2, copy, paste into R, and press the "enter" key.

#########################################################

########### Section 1 ############

# Initialize

library(stats) # load the stats library

library(nlme) # load the nlme library

library(graphics) # load the graphics library

alpha=0.05 # CI coverage rate is 100(1 - alpha)%

B=2000 # number of bootstrap samples

set.seed(1001) # for repeatibility

# Set working directory

setwd("/Volumes/Macintosh HD/Users/My Computer/Desktop") ##### Change for your computer #####

# If there is error and can't change directory, then try putting all files in exact same location as this R code file.

#-------------------------------------------------------------

# initialization function for special sigmoidal equations

#-------------------------------------------------------------

sigmoidalInit <- function(mCall, LHS, data){

xy <- sortedXyData(mCall[["x"]], LHS, data)

if (nrow(xy)<3){

stop("Too few distinct input values to fit to model")

}

A <- max(abs(xy[,"y"]))

K <- (max(xy[,"y"])-min(xy[,"y"]))/(max(xy[,"x"])-min(xy[,"x"]))

value <- c(A, K)

names(value) <- mCall[c("A", "K")]

value

}

#-------------------------------------------------------------

# initialization function for linear and exponential equations

#-------------------------------------------------------------

linearInit <- function(mCall, LHS, data){

xy <- sortedXyData(mCall[["x"]], LHS, data)

if (nrow(xy)<3){

stop("Too few distinct input values to fit to model")

}

K <- (max(xy[,"y"])-min(xy[,"y"]))/(max(xy[,"x"])-min(xy[,"x"]))

value <- c(K)

names(value) <- mCall[c("K")]

value

}

#------------------------------------------------------------

###### Load data file for individual bone. #####

#------------------------------------------------------------

##### Each bone should have its own csv file name. #####

##### Enter the correct file name here each time you run section 1. #####

histodata = read.csv("maia33.csv", header=T)

q <- length(histodata$present)

#--------------------------------------------

# Fit data to the monomolecular difference eq m=0

#--------------------------------------------

monomol <- selfStart(~A\*((1+exp(-K)\*((x/A)^(1-0)-1))^(1/(1-0))),

initial=sigmoidalInit, parameters=c("A", "K"))

getInitial(present ~ monomol(lagged, A, K), histodata)

monomol.nls <- nls(present ~ monomol(lagged, A, K), data=histodata)

summary(monomol.nls)

logLik(monomol.nls)

AIC(monomol.nls, k=log(nrow(histodata)))

#--------------------------------------------

# Fit data to the von Bertalanffy difference eq m=2/3

#--------------------------------------------

vonbert <- selfStart(~A\*((1+exp(-K)\*((x/A)^(1-(2/3))-1))^(1/(1-(2/3)))),

initial=sigmoidalInit, parameters=c("A", "K"))

getInitial(present ~ vonbert(lagged, A, K), histodata)

vonbert.nls <- nls(present ~ vonbert(lagged, A, K), data=histodata)

summary(vonbert.nls)

logLik(vonbert.nls)

AIC(vonbert.nls, k=log(nrow(histodata)))

#---------------------------------------

# Fit data to the Gompertz difference eq m=1.001

#---------------------------------------

gompertz <- selfStart(~A\*((1+exp(-K)\*((x/A)^(1-1.001)-1))^(1/(1-1.001))),

initial=sigmoidalInit, parameters=c("A", "K"))

getInitial(present ~ gompertz(lagged, A, K), histodata)

gompertz.nls <- nls(present ~ gompertz(lagged, A, K), data=histodata)

summary(gompertz.nls)

logLik(gompertz.nls)

AIC(gompertz.nls, k=log(nrow(histodata)))

#---------------------------------------

# Fit data to the logistic difference eq m=2 (2012)

#---------------------------------------

logistic <- selfStart(~A\*((1+exp(-K)\*((x/A)^(1-2)-1))^(1/(1-2))),

initial=sigmoidalInit, parameters=c("A", "K"))

getInitial(present ~ logistic(lagged, A, K), histodata)

logistic.nls <- nls(present ~ logistic(lagged, A, K), data=histodata)

summary(logistic.nls)

logLik(logistic.nls)

AIC(logistic.nls, k=log(nrow(histodata)))

#---------------------------------------

# Fit data to the linear difference eq

#---------------------------------------

linear <- selfStart(~x\*exp(K/x), initial=linearInit, parameters=c("K"))

getInitial(present ~ linear(lagged, K), histodata)

linear.nls <- nls(present ~ linear(lagged, K), data=histodata)

summary(linear.nls)

logLik(linear.nls)

AIC(linear.nls, k=log(nrow(histodata)))

########### End of Section 1 ############

#########################################################

########### Section 2 ############

#--------------------------------------------------------------------------

# Section 2 will produce 95% confidence intervals using model-based (parametric) bootstrapping coefficients of best model.

# Best model is determined by manually comparing AICc results for model with lowest value from Section 1.

# Using mean values from Section 1, produce data table for mean curve by

# using the equation y=A\*((1+exp(-K)\*((x/A)^(1-m)-1))^(1/(1-m))

# Insert the shape parameter m and mean values of A and K, and the value x of the size measurement at the beginning of the time interval.

# Repeat for each year, up to total age of oldest individual in sample.

# Save as csv in same format as the other csv data files.

#--------------------------------------------------------------------------

# Ensure that definitions for A and K are still initialized.

#-------------------------------------------------------------

# initialization function for special sigmoidal equations

#-------------------------------------------------------------

sigmoidalInit <- function(mCall, LHS, data){

xy <- sortedXyData(mCall[["x"]], LHS, data)

if (nrow(xy)<3){

stop("Too few distinct input values to fit to model")

}

A <- max(abs(xy[,"y"]))

K <- (max(xy[,"y"])-min(xy[,"y"]))/(max(xy[,"x"])-min(xy[,"x"]))

value <- c(A, K)

names(value) <- mCall[c("A", "K")]

value

}

#-------------------------------------------------------------

# initialization function for linear and exponential equations

#-------------------------------------------------------------

linearInit <- function(mCall, LHS, data){

xy <- sortedXyData(mCall[["x"]], LHS, data)

if (nrow(xy)<3){

stop("Too few distinct input values to fit to model")

}

K <- (max(xy[,"y"])-min(xy[,"y"]))/(max(xy[,"x"])-min(xy[,"x"]))

value <- c(K)

names(value) <- mCall[c("K")]

value

}

#-------------------------------------------------------------

# Load the data table and A and K values for the mean curve.

#-------------------------------------------------------------

histodata = read.csv("maiamean.csv", header=T) ##### Enter your mean csv file name here. #####

q <- length(histodata$present)

# Create A.res and K.res with number of columns B

B=2000 # number of bootstrap samples

A <- 29.591126 ##### Manually enter your mean A used to construct mean data set. #####

K <- 0.799560 ##### Manually enter your mean K used to construct mean data set. #####

A.res <- matrix(nrow=1, ncol=B)

K.res <- matrix(nrow=1, ncol=B)

A.res[1]<-A

K.res[1]<-K

#--------------------------------------------

# Initialize the model equation

#--------------------------------------------

##### Monomolecular is used here as an example. If monomolecular was not your best model, then be sure to change the model type every single time it is mentioned in the rest of Section 2. #####

monomol <- selfStart(~A\*((1+exp(-K)\*((x/A)^(1-0)-1))^(1/(1-0))), initial=sigmoidalInit, parameters=c("A", "K"))

#vonbert <- selfStart(~A\*((1+exp(-K)\*((x/A)^(1-(2/3))-1))^(1/(1-(2/3)))), initial=sigmoidalInit, parameters=c("A", "K"))

#gompertz <- selfStart(~A\*((1+exp(-K)\*((x/A)^(1-1.001)-1))^(1/(1-1.001))), initial=sigmoidalInit, parameters=c("A", "K"))

#logistic <- selfStart(~A\*((1+exp(-K)\*((x/A)^(1-2)-1))^(1/(1-2))), initial=sigmoidalInit, parameters=c("A", "K"))

#-------------------------------------------------------------------

# Residuals

#-------------------------------------------------------------------

# Residuals add small amounts of error to each data point. The bootstrapping will do this 2000 times.

# Calculate the mean error from the earlier results:

# When running the monomol model on each actual bone in Section 1, you recorded its residual standard error.

# Calculate the mean of these errors in Excel.

##### Will need to manually alter this value each time (currently 0.709) #####

mrse= 0.709 ##### Change to your value for mean residual standard error. #####

resid.res <- matrix(rnorm(B\*q,0,mrse),nrow=q,ncol=B)

# q is the length of "histodata", which varies with each input bone file

# B is the number of bootstrap replicates.

# 0 is used as the mean of the residual error distribution

# 0.709 (or whatever your mean error is) is used as the standard deviation of the residual error distribution

#-------------------------------------------------------------------

#--------------------------------------------

# Bootstrap loop

#--------------------------------------------

for (i in 2:B){ # Will repeat for 2000 replicates

histodata.res <- histodata

# This will shock/jitter the initial point of each specimen with a random residual

# Later points also get shocked in loop below with resid.res

# If any of these rnorm values are below the negative value of hatchling size, it will create errors.

# I.e., if hatchling tibia circumference is 2.5 cm, then an rnorm value lower than -2.5 added to it will result in a hatchling size below 0, causing an error.

# Problem should be rare and random for most length measurements. Try running this loop again, may get no errors.

# For body mass, hatchling mass may be much smaller than the mean residual standard error.

# In this case, manually alter the line below, replacing "mrse" with a smaller value.

# 1/10 or 1/100 of mrse, whichever is needed to ensure that the randomly selected normal distribution values are

# less than hatchling size.

# This will only affect the shocking of the hatchling size; all later sizes will get normal error.

# Adding random residual value to each data point

histodata.res$lagged[1] <- histodata.res$lagged[1] + rnorm(1,0,mrse) #### replace mrse with smaller value if needed

# Model-based resampling of mean data table (1:q) (number of rows in data)

# Creates duplicate data table with slightly different values

for (j in 1:q){

histodata.res$present[j] <-

##### Pick your model name, remove "#" to activate; add "#" to deactivate others. #####

# monomolecular version

(A.res[1]\*((1+exp(-K.res[1])\*((histodata.res$lagged[j]/A.res[1])^(1-0)-1))^(1/(1-0)))+resid.res[j,i])

# vonbert version

# (A.res[1]\*((1+exp(-K.res[1])\*((histodata.res$lagged[j]/A.res[1])^(1-(2/3))-1))^(1/(1-(2/3))))+resid.res[j,i])

# gompertz version

# (A.res[1]\*((1+exp(-K.res[1])\*((histodata.res$lagged[j]/A.res[1])^(1-(1.001))-1))^(1/(1-(1.001))))+resid.res[j,i])

# logistic version

# (A.res[1]\*((1+exp(-K.res[1])\*((histodata.res$lagged[j]/A.res[1])^(1-(2))-1))^(1/(1-(2))))+resid.res[j,i])

if (histodata.res$present[j] < histodata.res$lagged[j]){

histodata.res$present[j] <- histodata.res$lagged[j]

}

if (j < q) histodata.res$lagged[j+1] <- histodata.res$present[j]

}

##### Pick your model name, remove "#" to activate; add "#" to deactivate others. #####

getInitial(present ~ monomol(lagged, A, K), histodata.res)

#getInitial(present ~ vonbert(lagged, A, K), histodata.res)

#getInitial(present ~ gompertz(lagged, A, K), histodata.res)

#getInitial(present ~ logistic(lagged, A, K), histodata.res)

# Now it takes the shocked/jittered bootstrap data and runs a model to get the A K for this iteration

##### Pick your model name, remove "#" to activate; add "#" to deactivate others. #####

monomol.nls.res <- nls(present ~ monomol(lagged, A, K), data=histodata.res)

A.res[i] <- coef(monomol.nls.res)[1]

K.res[i] <- coef(monomol.nls.res)[2]

# vonbert.nls.res <- nls(present ~ vonbert(lagged, A, K), data=histodata.res)

# A.res[i] <- coef(vonbert.nls.res)[1]

# K.res[i] <- coef(vonbert.nls.res)[2]

# gompertz.nls.res <- nls(present ~ gompertz(lagged, A, K), data=histodata.res)

# A.res[i] <- coef(gompertz.nls.res)[1]

# K.res[i] <- coef(gompertz.nls.res)[2]

# logistic.nls.res <- nls(present ~ logistic(lagged, A, K), data=histodata.res)

# A.res[i] <- coef(logistic.nls.res)[1]

# K.res[i] <- coef(logistic.nls.res)[2]

}

################# End of bootstrap loop #################

# So the bootstrapping made 2000 copies of the data that were shocked with a random residual

# and then the monomolecular (or other) model was run on the resulting data, producing

# 2000 A and K values that were then saved in A.res and K.res.

#---------------------------------------------------------------------

# Calculating growth values: this will extract the mean asymptotic circumference, growth rate constant, and

# calculate the MGR, RGR, iGR and duration to 95% size

#---------------------------------------------------------------------

# Use if there are NAs (Use these two lines of code to remove NAs if you don't want to do it by hand. We did not have any NAs to get rid of.)

#A.res <- A.res[,-which(apply(A.res,2,function(x)all(is.na(x))))]

#K.res <- K.res[,-which(apply(K.res,2,function(x)all(is.na(x))))]

#---------------------------------------------------

##### Adjust hatching size here. Can use any units as long as they are consistent for entire analysis. #####

A0 <- 2.5 # Maiasaura hatchling tibia circumference in cm

#A0 <- 7.5 # Maiasaura hatchling tibia length in cm

#A0 <- 0.544366 # Maiasaura hatchling body length in m

#A0 <- 2 # Maiasaura hatchling body mass in kg

##### Adjust model type here #####

m <- 0 # m parameter for the monomolecular model

#m <- 2/3 # m parameter for the vonbert model

#m <- 1.001 # m parameter for the Gompertz model

#m <- 2 # m parameter for the logistic model

# Note that in R, the command "log" computes a natural log by default (command "ln" in Excel).

# This form of the I equation is from Cooper et al. (2008), and differs visually from the form in Lee and O'Connor (2013), but they are mathematically equivalent.

I.res <- (1/K.res)\*log((1/(m-1))\*((A0/A.res)^(1-m)-1))

duration.res <- I.res + 3/K.res

MGR.res <- A.res\*K.res\*m^(m/(1-m))

time.75A <- log((0.75^(1-m)-1)/(m-1))\*(-1/K.res)+I.res #Age at 75% grown

circum\_RM.res <- A.res\*exp(-exp(-K.res\*(time.75A-I.res)))

RGR.res <- vector(mode="numeric",length(K.res))

iGR.res <- vector(mode="numeric",length(K.res))

for (i in 1:length(K.res)){

A <- A.res[i]

K <- K.res[i]

I <- I.res[i]

time <- time.75A[i]

g <- deriv(~ A\*exp(-exp(-K\*(time-I))),"time")

RGR.res[i] <- attr(eval(g),"gradient")/circum\_RM.res[i]

iGR.res[i] <- attr(eval(g),"gradient")

}

#-----------------------------------

# calculate 95% confidence intervals

#-----------------------------------

alpha=0.05 # CI coverage rate is 100(1 - alpha)%

A.sort <- sort(A.res)

K.sort <- sort(K.res)

I.sort <- sort(I.res)

duration.sort <- sort(duration.res)

MGR.sort <- sort(MGR.res)

RGR.sort <- sort(RGR.res)

iGR.sort <- sort(iGR.res)

time.75A.sort <- sort(time.75A)

A.lo <- A.sort[floor((alpha/2)\*length(A.res))]

K.lo <- K.sort[floor((alpha/2)\*length(K.res))]

I.lo <- I.sort[floor((alpha/2)\*length(I.res))]

duration.lo <- duration.sort[floor((alpha/2)\*length(duration.res))]

MGR.lo <- MGR.sort[floor((alpha/2)\*length(MGR.res))]

RGR.lo <- RGR.sort[floor((alpha/2)\*length(RGR.res))]

iGR.lo <- iGR.sort[floor((alpha/2)\*length(iGR.res))]

time.75A.lo <- time.75A.sort[floor((alpha/2)\*length(time.75A))]

A.up <- A.sort[ceiling((1-alpha/2)\*length(A.res))]

K.up <- K.sort[ceiling((1-alpha/2)\*length(K.res))]

I.up <- I.sort[ceiling((1-alpha/2)\*length(I.res))]

duration.up <- duration.sort[ceiling((1-alpha/2)\*length(duration.res))]

MGR.up <- MGR.sort[ceiling((1-alpha/2)\*length(MGR.res))]

RGR.up <- RGR.sort[ceiling((1-alpha/2)\*length(RGR.res))]

iGR.up <- iGR.sort[ceiling((1-alpha/2)\*length(iGR.res))]

time.75A.up <- time.75A.sort[ceiling((1-alpha/2)\*length(time.75A))]

A.ci <- c(A.lo, A.up)

K.ci <- c(K.lo, K.up)

I.ci <- c(I.lo, I.up)

duration.ci <- c(duration.lo, duration.up)

MGR.ci <- c(MGR.lo, MGR.up)

RGR.ci <- c(RGR.lo, RGR.up)

iGR.ci <- c(iGR.lo, iGR.up)

time.75A.ci <- c(time.75A.lo, time.75A.up)

"Asymptotic bone circumference:"

A.res[1]

A.ci

"growth rate constant:"

K.res[1]

K.ci

"Age at inflection:"

I.res[1]

I.ci

"duration to 95% asymptotic size"

duration.res[1]

duration.ci

"maximum growth rate:"

MGR.res[1]

MGR.ci

"relative growth rate:"

RGR.res[1]

RGR.ci

"instantaneous growth rate:"

iGR.res[1]

iGR.ci

"time to 75 percent size"

time.75A[1]

time.75A.ci

# If you run Section 2 a second time, you should get the same results for the mean data, but slightly different results for the lower and upper 95% confidence intervals due to the random sampling of the bootstrapping.

#--------------------------------------------------------------------

# Export bootstrap data into .CSV files for future use if desired.

#--------------------------------------------------------------------

write(A.res, "A\_fem\_res.csv", sep=",")

write(K.res, "K\_fem\_res.csv", sep=",")

write(I.res, "I\_fem\_res.csv", sep=",")

write(duration.res, "duration\_fem\_res.csv", sep=",")

write(MGR.res, "MGR\_fem\_res.csv", sep=",")

write(RGR.res, "RGR\_fem\_res.csv", sep=",")

write(iGR.res, "iGR\_fem\_res.csv", sep=",")

write(time.75A, "time\_75A\_fem.csv", sep=",")

#-----------------------------------------------------------

# Plot graph of mean curve and 95% confidence interval

#-----------------------------------------------------------

##### Change y-axis label for your measurement type #####

plot(1, 1, type="n", ylab="Tibial circumference (cm)", xlab="Age (yr)",

xlim=c(0,q), ylim=c(0,A.ci[2]), las=1, family="serif")

curve(A.res[1]\*exp(-exp(-K.res[1]\*(x-I.res[1]))), 0, 15, 30, lwd=2, add=T)

time.range <- c(0:15)

predict.res <- matrix(nrow=length(time.range), ncol=length(A.res))

predict.lo <- c(0:15)

predict.up <- c(0:15)

for (i in 1:length(time.range)){

predict.res[i,] <- A.res\*exp(-exp(-K.res\*(time.range[i]-I.res)))

predict.lo[i] <- sort(predict.res[i,])[floor((alpha/2)\*length(A.res))]

predict.up[i] <- sort(predict.res[i,])[ceiling((1-alpha/2)\*length(A.res))]

}

points(time.range, predict.lo, type="l", lty=2, lwd=1)

points(time.range, predict.up, type="l", lty=2, lwd=1)

########### End of Section 2 ############