Calculate population level metrics

Kevin Healy

16 Feb 2019

This document outlines the calulation of the metrics associated with the Axis of demography paper Healy et al. Beyound the fast slow continuum.

```
library(popbio)
library(popdemo)
```

```
## Welcome to popdemo! This is version 1.3-0
## Use ?popdemo for an intro, or browseVignettes('popdemo') for vignettes
## Citation for popdemo is here: doi.org/10.1111/j.2041-210X.2012.00222.x
## Development and legacy versions are here: github.com/iainmstott/popdemo
```

```
library(ape)
library(caper)
```

```
## Loading required package: MASS
```

```
## Loading required package: mvtnorm
```

```
library(phytools)
```

```
## Loading required package: maps
```

```
library(MCMCglmm)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following object is masked from 'package:phytools':
##
##     expm
```

```
## Loading required package: coda
```

```
library(mulTree)
```

```
## Loading required package: hdrcde
```

```
## This is hdrcde 3.2
```

```
## Loading required package: snow
```

```
library(ineq)
library(pspline) #is this still relavent?

#devtools to get the Mage package
library(devtools)
##there seems to be a bug in there code to downlaod at the moment
#install_github("jonesor/compadreDB/Mage")
```

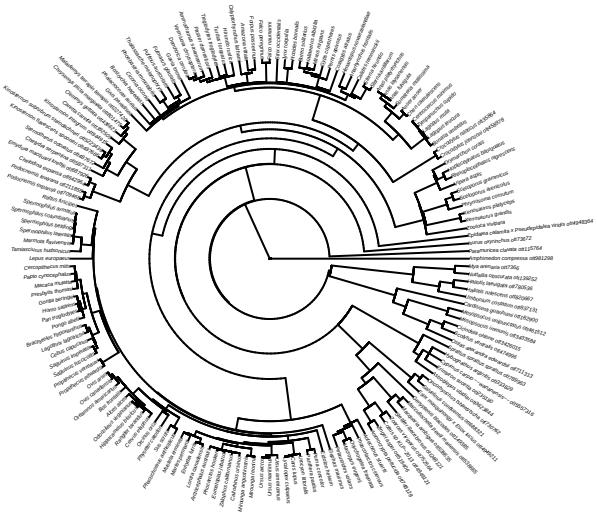
The data is from a version of the COMADRE database released with this paper. This may have been updated since so check out for more species or corrections to certain species in the datset here (give website link)

```
load("COMADRE_vX.RData")

source("Demography_functions.R")
```

Load in the phylogeny and plot the first tree in the distribution

```
com_tree <- read.tree("COMADRE_100_phylo_feb_2019.tre")
plot(com_tree[[1]], cex = 0.2, type = "fan")
```



Next we load in the trait data collated from the litrature

```
trophic_data <- read.csv("Trophic_data_June16_2017.csv",
                         sep = ",", header = T)
```

Next we subset the compadre dataset so that only matrices are kept for populations that are not experimentaly manipulated, can be divided into fecundity and growth elements, are recodred annually, and have at least 2 years.

```
###Subset entries form comadre with the below restictions for pooled and mean matrices.
pooled_Metadata <- (subset(comadre$metadata,
                           MatrixComposite == "Pooled"
                           & MatrixDimension >= 2
                           & StudyDuration >= 2
                           & MatrixSplit == "Divided"
                           & MatrixFec == "Yes"
                           & MatrixTreatment == "Unmanipulated"
                           & AnnualPeriodicity == "1"
                           & SurvivalIssue<=1
))

mean_Metadata <- (subset(comadre$metadata,
                         MatrixComposite == "Mean"
                         & MatrixDimension >= 2
                         & StudyDuration >= 2
                         & MatrixSplit == "Divided"
                         & MatrixFec == "Yes"
                         & MatrixTreatment == "Unmanipulated"
                         & AnnualPeriodicity == "1"
                         & SurvivalIssue<=1
```

```
))


###stick them together
combined_data <- rbind(pooled_Metadata, mean_Metadata)

###pull out the matching rows
keep_first <- as.numeric(rownames(combined_data))

##use these rows to pull out the matrices
combMat <- comadre$mat[keep_first]

#and associated matrix Class data
combmatrixClass <- comadre$matrixClass[keep_first]

##and set up a vector of the species
species_list <- data.frame(species =(combined_data$SpeciesAccepted),
                           class = (combined_data$Class),
                           phyla = (combined_data$Phylum))


#pull out the unique species
species_list_u <- unique(species_list$species)
```

Now check all matrices for egordicity, primativity and irriducablity

```
is_ergodic <- vector()
is_primitive <- vector()
is_irreducible <- vector()
is_post_rep <- vector() ##this gives true if repo is > 0 for final colume, hence false means its post r
all_true <- vector()

for(i in 1:length(keep_first)){
  tryCatch({
    is_ergodic[i] <- isErgodic(combMat[[i]]$matA)
    is_primitive[i] <- isPrimitive(combMat[[i]]$matA)
    is_irreducible[i] <- isIrreducible(combMat[[i]]$matA)
    is_post_rep[i] <- is.matrix_post_rep(combMat[[i]]$matA)
  }, error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
  all_true[i] <- all(c(is_ergodic[i],is_primitive[i],is_irreducible[i]) == TRUE)
}
```

```
## ERROR : infinite or missing values in 'x'
## ERROR : infinite or missing values in 'x'
## ERROR : infinite or missing values in 'x'
## ERROR : infinite or missing values in 'x'
## ERROR : infinite or missing values in 'x'
## ERROR : infinite or missing values in 'x'
## ERROR : infinite or missing values in 'x'
## ERROR : infinite or missing values in 'x'
## ERROR : infinite or missing values in 'x'
## ERROR : infinite or missing values in 'x'
```

```
test.frame <- data.frame(is_ergodic,is_primitive,is_irreducible,is_post_rep)
keep <- which(all_true == TRUE)
```

```
discard_species <- which(all_true == FALSE)
discard_species_names  <- unique(combined_data[discard_species,]$SpeciesAccepted)

clean_species <- combined_data[keep,]$SpeciesAccepted
```

Now to calulate each of the metrics. First lets get some info on all the matrices we are using.

```
##loop through keep whaich are all the matces that passed the above tests.
##species names
species_ind_full <- vector()
##taxa name
taxa_name <- vector()
##population_name
pop_mat_name <- vector()
###name of the first stage in the matrix
first_stage <- vector()
##matrix dimesnion
matrix_size <- vector()


for(i in 1:length(keep)){

##species names
species_ind_full[i] <-  combined_data[keep[i],]$SpeciesAccepted

##taxa name
taxa_name[i] <- as.vector(combined_data[keep[i],]$Class)

##population_name
pop_mat_name[i] <- combined_data[keep[i],]$MatrixPopulation

###name of the first stage in the matrix
first_stage[i] <- as.vector(combmatrixClass[keep[i]][[1]][1,2])

##matrix dimesnion
matrix_size[i] <- dim(combMat[[keep[i]]]$matA)[1]

}

first_stage <- unlist(first_stage)
```

Now lets calulate age at first reporduction

```
##Age at first reproduction
life_time_La <- vector()

for(i in 1:length(keep)){
  tryCatch({

    ##### age at first reproduction
    life_time_La[i] <-  lifeTimeRepEvents(matU =   combMat[[keep[i]]]$matU,
                                          matF =  combMat[[keep[i]]]$matF,
                                          startLife = 1)$La

    },error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
```

```
}
```

```
## ERROR : missing value where TRUE/FALSE needed
## ERROR : Lapack routine dgesv: system is exactly singular: U[17,17] = 0
## ERROR : matF contains only 0 values
```

Now lets calulate the age when 95% an 99% of the mature population is alive

```
##age (year) when 99% of chohort are dead
surv_99 <- vector()
##age (year) when 95% of chohort are dead
surv_95 <- vector()
#a resonable minimum maximum which is the hist of matrix size, 99% survival or QSD
min_max <- vector()

for(i in 1:length(keep)){
  tryCatch({


    ##### age when 95% are dead
    surv_95[i] <- exceptionalLife(combMat[[keep[i]]]$matU, startLife = 1)[1]

    ##### age when 99% are dead

    surv_99[i] <- which(log10(makeLifeTable(
                    matU = combMat[[keep[i]]]$matU,
                    matF =  combMat[[keep[i]]]$matF,
                    matC =  combMat[[keep[i]]]$matC,
              startLife = 1, nSteps = 1000)$lx*1000) < 0)[2]


    min_max[i] <- max(surv_99[i],matrix_size[i])
   #min_max[i] <- surv_99[i]

    },error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
}
```

```
## Warning in min(which(lifespanLeftover < 5)): no non-missing arguments to
## min; returning Inf
```

```
## Warning in min(which(lifespanLeftover < 1)): no non-missing arguments to
## min; returning Inf
```

```
## Warning in makeLifeTable(matU = combMat[[keep[i]]]$matU, matF =
## combMat[[keep[i]]]$matF, : matF contains only 0 values
```

Now lets calulate some lifetables

```
###life table
lxmx_curve <- list()

for(i in 1:length(keep)){
  tryCatch({

    ####calulculate the lxmx curve
    #I use max lifespan as the cut off point
    lxmx_curve[[i]] <-  makeLifeTable(matU =  combMat[[keep[i]]]$matU,
```

```
                                matF =  combMat[[keep[i]]]$matF,
                                matC =  combMat[[keep[i]]]$matC,
                                startLife = 1, nSteps = min_max[i])

  },error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
}
```

```
## ERROR : negative length vectors are not allowed
```

```
## Warning in makeLifeTable(matU = combMat[[keep[i]]]$matU, matF =
## combMat[[keep[i]]]$matF, : matF contains only 0 values
```

Mean life expectancy

```
mean_life_expect <- vector()
for(i in 1:length(keep)){
  tryCatch({
    ##### mean life expectancy
    mean_life_expect[i] <-  meanLifeExpectancy(matU = combMat[[keep[i]]]$matU,
                                               startLife= 1)

        },error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
}
```

```
## ERROR : Lapack routine dgesv: system is exactly singular: U[17,17] = 0
```

Life expectancy conditional on reaching sexual maturity

```
M_rep_lif_exp <- vector()
for(i in 1:length(keep)){
  tryCatch({

  ##life expectancy conditional on reaching reporduction
    M_rep_lif_exp[i] <- lifeTimeRepEvents(matU =  combMat[[keep[i]]]$matU,
                                          matF =  combMat[[keep[i]]]$matF,
                                          startLife = 1)$meanRepLifeExpectancy

        },error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
}
```

```
## ERROR : missing value where TRUE/FALSE needed
## ERROR : Lapack routine dgesv: system is exactly singular: U[17,17] = 0
## ERROR : matF contains only 0 values
```

Generation Time

```
gen_time <- vector()

for(i in 1:length(keep)){
  tryCatch({

    ##### generation time from popbio package
    gen_time[i]  <- generation.time(A =  combMat[[keep[i]]]$matA)

        },error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
}
```

Mean repo rate

```
mean_repo_rate <- vector()
mean_repo_rate_not_fixed <- vector()
mean_repo_rate_stable_state <- vector()
for(i in 1:length(keep)){
  tryCatch({

     ### mean reporduction rate
    mean_repo_rate_stable_state[i] <- meanRepo(matA = combMat[[keep[i]]]$matA,
                                               matF = combMat[[keep[i]]]$matF)

    mean_repo_rate[i] <- mean(combMat[[keep[i]]]$matF[1,combMat[[keep[i]]]$matF[1,] > 0])



        },error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
}
```

Gini and mx reporduction

```
gini <- vector()
mxlx_mean <- vector()
mxlxsd <- vector()
mxsd <- vector()

gini_prop <-vector()

for(i in 1:length(keep)){
  tryCatch({

     ### mean reporduction rate
       gini[i] <- Gini(lxmx_curve[[i]]$lx*lxmx_curve[[i]]$mx, corr= F)

       mxlxsd[i] <- sd(lxmx_curve[[i]]$lx*lxmx_curve[[i]]$mx)

        },error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
}
```

standard diviation of survival

```
sbins <- 100

surv_sd <- vector()
fx_curve <- list()
La_sbins <- vector()

spline_curve <- list()
##age at which 10% of the population are alive


##age at which 90% of the population are alive


##calulate f(x) age at death
for(i in 1:length(keep)){
  tryCatch({
```

```
    spline_curve[[i]] <-  unlist(lx_spline(lxmx_curve[[i]]$lx, lxmx_curve[[i]]$x, bins = sbins)[1])

      La_sbins[i] <- round((life_time_La[i]/length(lxmx_curve[[i]]$lx))*sbins)

      for(z in 1:c(length(spline_curve[[i]])-1)){

    if(z == 1){fx_curve[i][1] <- 1 - c(spline_curve[[i]][z+1])}

    else{ fx_curve[[i]][z] <- c(spline_curve[[i]][z]) - spline_curve[[i]][z+1]

  }
      }

       surv_sd[i] <- sd(fx_curve[[i]], na.rm = TRUE)


        },error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
}
```

```
## Warning in max(lx): no non-missing arguments to max; returning -Inf
```

```
## Warning in is.na(lx): is.na() applied to non-(list or vector) of type
## 'NULL'
```

```
## Warning in max(x): no non-missing arguments to max; returning -Inf
```

```
## ERROR : 'to' must be a finite number
```

And put them all into a table

```
###some errors here could be cleaned up

ind_vital <- data.frame(species_ind_full,
                        taxa_name,
                        pop_mat_name,
                        mean_life_expect,
                        life_time_La,
                        mean_repo_rate,
                        mean_repo_rate_stable_state,
                        gen_time,
                        M_rep_lif_exp,
                        matrix_size,
                        surv_95,
                        surv_99,
                        gini,
                        mxlxsd,
                        surv_sd)

La_Dropped_species <- ind_vital[is.na(ind_vital[,c("life_time_La")]),
                                "species_ind_full"]
Mrepo_Dropped_species <- ind_vital[is.na(ind_vital[,c("mean_repo_rate")]),
                                "species_ind_full"]
gen_time_Dropped_species <- ind_vital[is.na(ind_vital[,c("gen_time")]),
                                "species_ind_full"]
M_rep_lif_exp_Dropped_species <- ind_vital[is.na(ind_vital[,c("M_rep_lif_exp")]),
                                "species_ind_full"]
```

```
surv_99_Dropped_species <- ind_vital[is.na(ind_vital[,c("surv_99")]),
                                     "species_ind_full"]
surv_95_Dropped_species <- ind_vital[is.na(ind_vital[,c("surv_95")]),
                                     "species_ind_full"]
```

Now just remove the sub_species part of the names

```
ind_vital[,"species_ind_full"]  <- gsub(" subsp.*","", ind_vital[,"species_ind_full"])
ind_vital[,"species_ind_full"]  <- gsub(" ","_", ind_vital[,"species_ind_full"])
```

Next we add in the external infor such as habitat etc.

```
##trophic data
mass_g <- vector()
parental_care <- vector()
repo_type <- vector()
mobility <- vector()
trophic_level <- vector()
habitat_ecology <- vector()
habitat <-  vector()
repo_size_g <- vector()


for(k in 1:(length(ind_vital$species_ind_full))){
  tryCatch({



      parental_care[k] <-   as.vector(na.omit(trophic_data[trophic_data$species  == as.vector(ind_vital

      repo_type[k] <-   as.vector(na.omit(trophic_data[trophic_data$species  == as.vector(ind_vital[k,"

      mobility[k] <-   as.vector(na.omit(trophic_data[trophic_data$species  == as.vector(ind_vital[k,"sp

      mass_g[k] <-   (na.omit(trophic_data[trophic_data$species  == as.vector(ind_vital[k,"species_ind_

      trophic_level[k] <-   as.vector(na.omit(trophic_data[trophic_data$species  == as.vector(ind_vital

      habitat_ecology[k] <-  as.vector(na.omit(trophic_data[trophic_data$species  == as.vector(ind_vita

      habitat[k] <-  as.vector(na.omit(trophic_data[trophic_data$species  == as.vector(ind_vital[k,"spe


      },error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
}
```

```
## Warning in parental_care[k] <- as.vector(na.omit(trophic_data[trophic_data
## $species == : number of items to replace is not a multiple of replacement
## length
```

```
## Warning in repo_type[k] <- as.vector(na.omit(trophic_data[trophic_data
## $species == : number of items to replace is not a multiple of replacement
## length
```

```
## Warning in mobility[k] <- as.vector(na.omit(trophic_data[trophic_data
## $species == : number of items to replace is not a multiple of replacement
```

```
## length

## Warning in trophic_level[k] <- as.vector(na.omit(trophic_data[trophic_data
## $species == : number of items to replace is not a multiple of replacement
## length

## Warning in habitat_ecology[k] <-
## as.vector(na.omit(trophic_data[trophic_data$species == : number of items to
## replace is not a multiple of replacement length

## Warning in habitat[k] <- as.vector(na.omit(trophic_data[trophic_data
## $species == : number of items to replace is not a multiple of replacement
## length

## Warning in parental_care[k] <- as.vector(na.omit(trophic_data[trophic_data
## $species == : number of items to replace is not a multiple of replacement
## length

## Warning in repo_type[k] <- as.vector(na.omit(trophic_data[trophic_data
## $species == : number of items to replace is not a multiple of replacement
## length

## Warning in mobility[k] <- as.vector(na.omit(trophic_data[trophic_data
## $species == : number of items to replace is not a multiple of replacement
## length

## Warning in trophic_level[k] <- as.vector(na.omit(trophic_data[trophic_data
## $species == : number of items to replace is not a multiple of replacement
## length

## Warning in habitat_ecology[k] <-
## as.vector(na.omit(trophic_data[trophic_data$species == : number of items to
## replace is not a multiple of replacement length

## Warning in habitat[k] <- as.vector(na.omit(trophic_data[trophic_data
## $species == : number of items to replace is not a multiple of replacement
## length

## Warning in parental_care[k] <- as.vector(na.omit(trophic_data[trophic_data
## $species == : number of items to replace is not a multiple of replacement
## length

## Warning in repo_type[k] <- as.vector(na.omit(trophic_data[trophic_data
## $species == : number of items to replace is not a multiple of replacement
## length

## Warning in mobility[k] <- as.vector(na.omit(trophic_data[trophic_data
## $species == : number of items to replace is not a multiple of replacement
## length

## Warning in trophic_level[k] <- as.vector(na.omit(trophic_data[trophic_data
## $species == : number of items to replace is not a multiple of replacement
## length

## Warning in habitat_ecology[k] <-
## as.vector(na.omit(trophic_data[trophic_data$species == : number of items to
## replace is not a multiple of replacement length

## Warning in habitat[k] <- as.vector(na.omit(trophic_data[trophic_data
## $species == : number of items to replace is not a multiple of replacement
## length
```

Stick them all together

```
pop_vital <- data.frame(ind_vital,
                        mass_g,
                        parental_care,
                        repo_type,
                        mobility,
                        trophic_level,
                        habitat_ecology,
                        habitat,
                        min_max
                        )
```

Let also add some human demography from other sources external to COMADRE

```
#load("PPMs Eusrostat 100x100.Rdata")
load("keyfitz_Agam_100x100.Rdata")
```

lets just calulate the whole lot together

```
#Spline bins

sbins <- 100

##I need to come up with a better breack down for these
#lets calculate for Ireland
Irish_rows_1960 <- as.numeric(rownames(keyfitz$countryyear[keyfitz$countryyear$country == "Ireland" & k
Irish_rows_1920 <- as.numeric(rownames(keyfitz$countryyear[keyfitz$countryyear$country == "Ireland" & k

#lets calculate for Sweden as Finland didnt work
Sweden_rows_a60 <- as.numeric(rownames(keyfitz$countryyear[keyfitz$countryyear$country == "Sweden" & key
Sweden_rows_1850_19 <- as.numeric(rownames(keyfitz$countryyear[keyfitz$countryyear$country == "Sweden" &

#lets calculate for Japan
Japan_rows_b60 <- as.numeric(rownames(keyfitz$countryyear[keyfitz$countryyear$country == "Japan" & keyfi
Japan_rows_a60 <- as.numeric(rownames(keyfitz$countryyear[keyfitz$countryyear$country == "Japan" & keyfi


Irish_rows<- as.numeric(rownames(keyfitz$countryyear[keyfitz$countryyear$country == "Ireland",]))

##average the pops
Irish_mat_1960 <- keyfitz$Agam100[Irish_rows_1960]
Irish_mat_1920 <- keyfitz$Agam100[Irish_rows_1920]
Sweden_rows_1850_19 <- keyfitz$Agam100[Sweden_rows_1850_19]
Sweden_rows_a60 <- keyfitz$Agam100[Sweden_rows_a60]
japan_mat_b60 <- keyfitz$Agam100[Japan_rows_b60]
japan_mat_a60 <- keyfitz$Agam100[Japan_rows_a60]

Irish_1960 <- meanMatrix(Irish_mat_1960)
Irish_1920 <- meanMatrix(Irish_mat_1920)
Sweden_1850_19 <- meanMatrix(Sweden_rows_1850_19)
Sweden_a60 <- meanMatrix(Sweden_rows_a60)
japan_b60 <- meanMatrix(japan_mat_b60)
japan_a60 <- meanMatrix(japan_mat_a60)
```

```
##you need to put this in the right order
human_pop <- c("Homo_Irish1920" , "Homo_Irish1960", "Sweden_1850_19","Sweden_a60", "japan_b60", "japan_
human_mats <- list(Irish_1920, Irish_1960, Sweden_1850_19, Sweden_a60, japan_b60, japan_a60)

##pop names and year
pop_country_year <- vector()
##mean life expectancy
h_mean_life_expect <- vector()
##age (year) when 99% of chohort are dead
h_surv_99 <- vector()
##age (year) when 95% of chohort are dead
h_surv_95 <- vector()
##life expectancy contingent on entering reproduction
h_M_rep_lif_exp <- vector()
##Age at first reproduction
h_life_time_La <- vector()
#generation time
h_gen_time <- vector()
##mean reporductive rate raw
h_mean_repo_rate <- vector()
##mean reporductive rate
h_mean_repo_rate_stable_state <- vector()
#matrix dimension size
h_matrix_size <- vector()
h_gini <- vector()
h_mxlxsd <- vector()
h_surv_sd <- vector()

fx_h_curve <- list()

h_lxmx_curve_list <- list()
h_spline_curve <- list()

##convert the age of maturity to the right point on the spline sampled curve
La_h_sbins <- vector()



for(i in 1:length(human_mats)){
  tryCatch({

###we first need to decompose the A matri into U F and C
    A_hum <-  human_mats[[i]]
    U_hum <- A_hum
    U_hum[1,] <- 0
    F_hum <- matrix(0,dim(A_hum)[1],dim(A_hum)[1])
    F_hum[1,] <- A_hum[1,]
    C_hum <- matrix(0,dim(A_hum)[1],dim(A_hum)[1])


    ##country and year
    pop_country_year[i] <- human_pop[i]
```

```
##matrix dimesnion
h_matrix_size[i] <- dim(A_hum)[1]


##### mean life expectancy
h_mean_life_expect[i] <-    meanLifeExpectancy(matU = U_hum,
                                        startLife= 1)
##### age when 95% are dead
h_surv_95[i] <- exceptionalLife(U_hum, startLife=1)[1]
##### age when 99% are dead
h_surv_99[i] <- which(log10(makeLifeTable(
                matU =  U_hum,
                matF =  F_hum,
                matC =  C_hum,
          startLife = 1, nSteps = 1000)$lx*1000) < 0)[2]

##### time to first reproduction
h_life_time_La[i] <-    lifeTimeRepEvents(matU = U_hum,
                                    matF = F_hum,
                                    startLife = 1)$La

##life expectancy conditional on reaching reporduction
h_M_rep_lif_exp[i] <- lifeTimeRepEvents(matU =  U_hum,
                                    matF =  F_hum,
                                    startLife = 1)$meanRepLifeExpectancy


##### generation time from popbio package
h_gen_time[i]  <- generation.time(A =  A_hum)

### mean reporduction rate
h_mean_repo_rate_stable_state[i] <- meanRepo(matA = A_hum,
                                         matF = F_hum)

h_mean_repo_rate[i] <- mean(F_hum[1,])



####calulculate the lxmx curve
#I use max lifespan as the cut off point
h_lxmx_curve <-  makeLifeTable(matU =  U_hum,
                            matF =  F_hum,
                            matC =  C_hum,
                            startLife = 1,
                            nSteps = h_surv_99[i])

h_lxmx_curve_list[[i]] <-  makeLifeTable(matU =  U_hum,
                            matF =  F_hum,
                            matC =  C_hum,
                            startLife = 1,
                            nSteps = h_surv_99[i])

h_spline_curve[[i]] <-  unlist(lx_spline(h_lxmx_curve_list[[i]]$lx, h_lxmx_curve_list[[i]]$x, bins =
```

```
    La_h_sbins[i] <- round((h_life_time_La[i]/h_surv_99[i])*sbins)

    h_gini[i] <- Gini(h_lxmx_curve$lx*h_lxmx_curve$mx)

    h_mxlxsd[i] <- sd(h_lxmx_curve$lx*h_lxmx_curve$mx)



    ###f(x) for humans

             for(z in 1:c(length(h_spline_curve[[i]])-1)){

     if(z == 1){fx_h_curve[i][1] <- 1 - c(h_spline_curve[[i]][z+1])}

     else{ fx_h_curve[[i]][z] <- c(h_spline_curve[[i]][z]) - h_spline_curve[[i]][z+1]

   }
     }



    h_surv_sd[i] <- sd(fx_h_curve[[i]], na.rm = TRUE)


    },error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
}


###some errors here could be cleaned up
h_ind_vital <- data.frame(
                        pop_country_year,
                        h_mean_life_expect,
                        h_life_time_La,
                        h_mean_repo_rate_stable_state,
                        h_mean_repo_rate,
                        h_gen_time,
                        h_gini,
                        h_M_rep_lif_exp,
                        h_matrix_size,
                        h_surv_95,
                        h_surv_99,
                        h_surv_sd,
                        h_mxlxsd
                       )
```

```
human_dem <- data.frame(species_ind_full = rep("Homo_sapiens", length(human_pop)),
                        taxa_name =  rep("Mammalia" ,length(pop_country_year)),
                        pop_mat_name = pop_country_year,
                        mean_life_expect = h_mean_life_expect,
                        life_time_La = h_life_time_La,
                        mean_repo_rate = h_mean_repo_rate,
```

```
                          mean_repo_rate_stable_state = h_mean_repo_rate_stable_state,
                          gen_time = h_gen_time,
                          M_rep_lif_exp = h_M_rep_lif_exp,
                          matrix_size = h_matrix_size,
                          surv_95 = h_surv_99,
                          surv_99 = h_surv_95,
                          gini = h_gini,
                          mxlxsd = h_mxlxsd,
                          surv_sd = h_surv_sd,
                          mass_g =  rep(trophic_data[trophic_data$species  == "Homo_sapiens","mass_g"],le
                          parental_care = rep(trophic_data[trophic_data$species  == "Homo_sapiens","paren
                          repo_type = rep(trophic_data[trophic_data$species  == "Homo_sapiens","repo_type
                          mobility = rep(trophic_data[trophic_data$species  == "Homo_sapiens","mobility"]
                          trophic_level = rep(trophic_data[trophic_data$species  == "Homo_sapiens","troph
                          habitat_ecology = rep(trophic_data[trophic_data$species  == "Homo_sapiens","hab
                          habitat = rep(trophic_data[trophic_data$species  == "Homo_sapiens","habitat"],l
                          min_max = h_surv_99)

full_m_data <- rbind(pop_vital,human_dem)
```

We can drop some variables

```
full_m_data <- full_m_data[,
                           c("species_ind_full",
                             "taxa_name",
                             "pop_mat_name",
                             "mean_life_expect",
                             "life_time_La",
                             "mean_repo_rate",
                             "mean_repo_rate_stable_state",
                             "gen_time",
                             "M_rep_lif_exp",
                             "matrix_size",
                             "surv_95",
                             "surv_99",
                             "gini",
                             "mxlxsd",
                             "surv_sd",
                             "mass_g",
                             "parental_care",
                             "repo_type",
                             "mobility",
                             "trophic_level",
                             "habitat_ecology",
                             "habitat",
                             "min_max")
                             ]
```

Now we can clean the data before expoting it

First lets add another column relating to whther the species is warm or cold blooded

```
full_m_data$met_type <- as.vector(full_m_data$taxa)
full_m_data[full_m_data$taxa == "Aves"|full_m_data$taxa == "Mammalia" ,"met_type"] <- "endo"
full_m_data[!(full_m_data$taxa == "Aves"|full_m_data$taxa == "Mammalia") ,"met_type"] <- "ecto"
```

Remove biologically crazy species

```
pop_data <- full_m_data

####remonve these problem species as they clearly have a problem
##Spratt have lifesspans >100 which clearly way off any sensible biology
pop_data_sr <- pop_data[pop_data$species != "Sprattus_sprattus",]
##This stupid monkey thinks it can live for >105
pop_data_sr2 <- pop_data_sr[pop_data_sr$species != "Brachyteles_hypoxanthus",]
##Scolytus_ventralis is coded in as annual but its a seasonal species
pop_data_sr3 <- pop_data_sr2[pop_data_sr2$species != "Scolytus_ventralis",]

####remove Enhydra_lutris its a simulation based study
##This gives an Na at the moment, check at the end to see if it still does, in which case remove this li
pop_data_sr4 <- pop_data_sr3[pop_data_sr3$species != "Enhydra_lutris",]

pop_data_sr5 <- pop_data_sr4[pop_data_sr3$species != "Somateria_mollissima",]


##remove the population colunm
drops <- c("pop_mat_name")
pop_data_nopop <- pop_data_sr5[,!(names(pop_data_sr5) %in% drops)]

#rename the species_ind_full col name to "species"
colnames(pop_data_nopop)[1] <- "species"
```

Now remove any infs of Na's in the pop metrics

```
##remove infinaties
pop_data_infr  <- do.call(data.frame,lapply(pop_data_nopop, function(x) replace(x, is.infinite(x),NA)))

full_infna_data <- na.omit(pop_data_infr)
```

```
##add a colume to idenfy pgls match up
phylo_match_data <- data.frame(full_infna_data, animal = full_infna_data$species)

pgls_unique_matched <- data.frame(species = unique(phylo_match_data$species),
                                  species_pgls = unique(phylo_match_data$animal))


axis_trees <- list()
for(i in 1:(length(com_tree))){
tree_ren <- com_tree[[i]]

#tree match
axis_comp<- comparative.data(phy = tree_ren,
                             data = pgls_unique_matched,
                             names.col = "species_pgls" , force.root = TRUE)
axis_trees[[i]] <- axis_comp$phy
}

class(axis_trees) <- "multiPhylo"

axis_comp$dropped$unmatched.rows
```

```
##  [1] "Cyprinus_carpio"           "Esox_lucius"
```

```
##  [3] "Maccullochella_peelii"     "Macquaria_ambigua"
##  [5] "Oncorhynchus_tshawytscha"  "Salvelinus_confluentus"
##  [7] "Thalassarche_melanophris"  "Haliotis_rufescens"
##  [9] "Physeter_macrocephalus"    "Urocitellus_armatus"
## [11] "Urocitellus_columbianus"   "Kinosternon_flavescens"
## [13] "Kinosternon_integrum"      "Ammocrypta_pellucida"
## [15] "Astroblepus_ubidiai"       "Genypterus_blacodes"
## [17] "Stellifer_illecebrosus"    "Zingel_asper"
## [19] "Paramuricea_clavata"       "Anthropoides_paradiseus"
## [21] "Setophaga_cerulea"         "Sternula_antillarum"
## [23] "Mya_arenaria"              "Nuttallia_obscurata"
## [25] "Amphimedon_compressa"      "Isurus_oxyrinchus"
## [27] "Haliotis_laevigata"        "Umbonium_costatum"
## [29] "Chelodina_expansa"         "Chelydra_serpentina"
## [31] "Chrysemys_picta"           "Clemmys_guttata"
## [33] "Crocodylus_johnsoni"       "Cryptophis_nigrescens"
## [35] "Drymarchon_couperi"        "Emydura_macquarii"
## [37] "Hoplocephalus_bungaroides" "Kinosternon_subrubrum"
## [39] "Malaclemys_terrapin"       "Podocnemis_lewyana"
## [41] "Podocnemis_expansa"
```

Lets also prun down the tree to subset for mammals, aves, endotherms and ectotherms.

```
##subset the data
phylo_match_mammal  <- phylo_match_data[phylo_match_data$taxa_name == "Mammalia",]
pgls_unique_mammal <- data.frame(species = unique(phylo_match_mammal$species),
                                 species_pgls = unique(phylo_match_mammal$animal))


phylo_match_aves  <- phylo_match_data[phylo_match_data$taxa_name == "Aves",]
pgls_unique_aves <- data.frame(species = unique(phylo_match_aves$species),
                                 species_pgls = unique(phylo_match_aves$animal))

phylo_match_endo  <- phylo_match_data[phylo_match_data$met_type == "endo",]
pgls_unique_endo <- data.frame(species = unique(phylo_match_endo$species),
                                 species_pgls = unique(phylo_match_endo$animal))

phylo_match_ecto  <- phylo_match_data[phylo_match_data$met_type == "ecto",]
pgls_unique_ecto <- data.frame(species = unique(phylo_match_ecto$species),
                                 species_pgls = unique(phylo_match_ecto$animal))


mam_trees <- list()
aves_trees <- list()
endo_trees <- list()
ecto_trees <- list()

for(i in 1:(length(com_tree))){
tree_ren <- com_tree[[i]]

#tree match mammals
mam_comp<- comparative.data(phy = tree_ren,
                            data = pgls_unique_mammal,
                            names.col = "species_pgls" , force.root = TRUE)
mam_trees[[i]] <- mam_comp$phy
```

```
#aves
aves_comp<- comparative.data(phy = tree_ren,
                             data = pgls_unique_aves,
                             names.col = "species_pgls" , force.root = TRUE)
aves_trees[[i]] <- aves_comp$phy

#endo
endo_comp<- comparative.data(phy = tree_ren,
                             data = pgls_unique_endo,
                             names.col = "species_pgls" , force.root = TRUE)
endo_trees[[i]] <- endo_comp$phy

#ecto
ecto_comp<- comparative.data(phy = tree_ren,
                             data = pgls_unique_ecto,
                             names.col = "species_pgls" , force.root = TRUE)
ecto_trees[[i]] <- ecto_comp$phy


}

class(mam_trees) <- "multiPhylo"
class(aves_trees) <- "multiPhylo"
class(endo_trees) <- "multiPhylo"
class(ecto_trees) <- "multiPhylo"
```

now lets write all our new data and phylogenies out

```
write.csv(phylo_match_data, file = "axis_analysis_data.csv", row.names = FALSE)
write.csv(phylo_match_mammal, file = "mammal_analysis_data.csv", row.names = FALSE)
write.csv(phylo_match_aves, file = "aves_analysis_data.csv", row.names = FALSE)
write.csv(phylo_match_endo, file = "endo_analysis_data.csv", row.names = FALSE)
write.csv(phylo_match_ecto, file = "ecto_analysis_data.csv", row.names = FALSE)

write.tree(axis_trees, file = "axis_analysis_phylo.tre")
write.tree(mam_trees, file = "mam_analysis_phylo.tre")
write.tree(aves_trees, file = "aves_analysis_phylo.tre")
write.tree(endo_trees, file = "endo_analysis_phylo.tre")
write.tree(ecto_trees, file = "ecto_analysis_phylo.tre")
```