Calculate population level metrics

Kevin Healy

16 Feb 2019

This document outlines the calculation of the life history metrics associated with the Axis of demography paper Healy et al. 2019.

Packages and Data

```
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
```

```
## Loading required package: rgl
```

```
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 3.3
```

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

```
library(ineq)
```

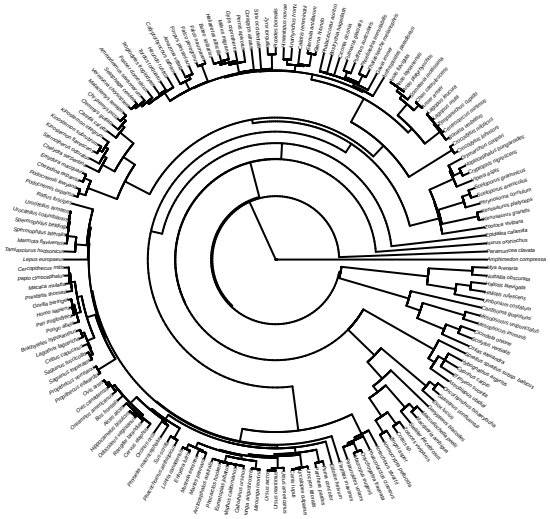
The data is from a version of the COMADRE database released with this paper. This may have been updated since so check out the on-line database for updates (https://www.compadre-db.org/Data/Comadre)

```
load("COMADRE_v_2.0.0.9.RData")

source("Demography_functions.R")
```

Load in the distribution of trees that was created in the Phylogeny_construction script and lets plot the first tree in the distribution.

```
com_tree <- read.tree("COMADRE_100_phylo.tre")
plot(com_tree[[1]], cex = 0.2, type = "fan")
```



Next we load in the trait data collated from the literature, see the supplementary methods for further description of these data.

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

Let also add some human demography from Keyfitz and Flieger 1968, 1971, 1990.

```
load("keyfitz_Agam_100x100.Rdata")
```

Data Subsetting

Next we subset the compadre dataset so that only matrices are kept for populations that are not experimental manipulated, can be divided into fecundity and growth elements, are recorded annually, and have at least 2 years. We used both pooled and mean matrices from Comadre (See Salguero-Gómez et al. 2016 for details on COMADRE data)

```
###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 ergodicity, primitivity and irreducibility

```
is_ergodic <- vector()
is_primitive <- vector()
is_irreducible <- vector()

#is_post_rep gives true if repo is > 0 for final column,
#hence false means its post reproductive
is_post_rep <- vector()

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)
}
```

```
test.frame <- data.frame(is_ergodic,is_primitive,is_irreducible,is_post_rep)

#keep describes the matrix, by number, that passed the above tests
#We use the number so we can refer back to information in the COMADRE object
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 lets get some info on all the matrices we are using.

```
##loop through the keep list which 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


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

}
```

Metric calculation

First we need to calculate a few things in order make further metric calculations. First we need a final age point. Here we calculate the age when 95% an 99% of a cohort is dead for each population matrix

```
##age (year) when 99% of cohort are dead
surv_99 <- vector()
##age (year) when 95% of cohort are dead
surv_95 <- vector()
```

```
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]

    #if the matrix dimension is larger then surv_99 we use that as the
    #lifespan cutoff instead.
    min_max[i] <- max(surv_99[i],matrix_size[i])

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

Next we calculate life tables from each of the matrix model in order to calculate metrics related to the spread of reproduction and survival. We use the makeLifeTable function from the Mage github repository (https://github.com/jonesor/compadreDB/tree/master/Mage)

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

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

    ####calculate the lxmx curve
    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")})
}
```

We use the `lifeTimeRepEvents` and `meanLifeExpectancy` functions from the `Mage` github repository (https://github.com/jonesor/compadreDB/tree/master/Mage), to calculate age at first reproduction, Mean life expectancy, and Life expectancy conditional on reaching sexual maturity,

Age at first reproduction

```
life_time_La <- vector()

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

```
  tryCatch({

    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")})
}
```

Mean life expectancy

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

        mean_life_expect[i] <-  meanLifeExpectancy(matU = combMat[[keep[i]]]$matU,
                                                   startLife= 1)

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

Life expectancy conditional on reaching sexual maturity

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

    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")})
}
```

Generation Time

For generation tiome we used `generation.time` function from the `popbio` package.

```
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 reporduction rate

We calculate the mean reproductive rate of a population both at its stable state distribution, and also using the raw matrix with no corrections.

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

     ### mean reproduction rate at stable state
    mean_repo_rate_stable_state[i] <- meanRepo(matA = combMat[[keep[i]]]$matA,
                                               matF = combMat[[keep[i]]]$matF)

     ### mean reproduction rate not at stable state
    mean_repo_rate[i] <- mean(combMat[[keep[i]]]$matF[1,combMat[[keep[i]]]$matF[1,] > 0])


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

Spread of reproduction

Using life tables we calculate the distribution of the lxmx curve using the Gini index function `Gini` from the `ineq` package. We also calculate the standard deviation of the curve.

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

gini_prop <-vector()

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

     ### mean reproduction 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")})
}
```

Spread of mortality

Using life tables we calculate the spread of mortality using the lx curves. To standardise for lx curve length we fit each curve with a spline and calculate the standard deviation of the distribution of mortality `fx`, which is the proportion of the cohort which die in each time step. We also calculate the probability of an individual reaching the age of sexual maturity.

```
sbins <- 100
```

```
surv_sd <- vector()
fx_curve <- list()
La_sbins <- vector()
prop_la <- vector()
spline_curve <- list()


##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)

        prop_la[i]<- spline_curve[[i]][La_sbins[i]]

      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")})
}
```

Clean up for export.

We put all the measures into a table

```
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,
                        prop_la)
```

```
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"]
prop_la_Dropped_species <- ind_vital[is.na(ind_vital[,c("prop_la")]),
                                "species_ind_full"]
```

Now we 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 information to create a single combined dataset.

```
mass_g <- vector()
mode_of_life <- vector()
repo_output <- vector()
met_rate <- vector()
IUCN <- vector()



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



      mode_of_life[k] <-   as.vector(trophic_data[trophic_data$species
                        == as.vector(ind_vital[k,"species_ind_full"]),
                          "mode_of_life"])

      mass_g[k] <-   as.vector(trophic_data[trophic_data$species
                        == as.vector(ind_vital[k,"species_ind_full"]),
                          "body_mass_g"])

      repo_output[k] <-   as.vector(trophic_data[trophic_data$species
                        == as.vector(ind_vital[k,"species_ind_full"]),
                          "mass_specific_reproductive_output_g_g"])

      met_rate[k] <-   as.vector(trophic_data[trophic_data$species
                        == as.vector(ind_vital[k,"species_ind_full"]),
                          "mass_specific_metabolic_rate_w_kg"])


      IUCN[k] <-   as.vector(trophic_data[trophic_data$species
                        == as.vector(ind_vital[k,"species_ind_full"]),
```

```
                          "IUCN_statues"])


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

Stick them all together

```
pop_vital <- data.frame(ind_vital,
                        mass_g,
                        mode_of_life,
                        repo_output,
                        met_rate,
                        IUCN,
                        min_max
                         )
```

Human data

Due to the format of the human data we calculated all the metrics separately for humans and ten combined them into a single dataset at the end.

```
#Spline bins

sbins <- 100

#lets calculate for Ireland
Irish_rows_1960 <- as.numeric(rownames(keyfitz$countryyear[keyfitz$countryyear$country
                                    == "Ireland" & keyfitz$countryyear$year > 1960,]))
Irish_rows_1920 <- as.numeric(rownames(keyfitz$countryyear[keyfitz$countryyear$country
                                    == "Ireland" & keyfitz$countryyear$year < 1960,]))

#lets calculate for Sweden
Sweden_rows_a60 <- as.numeric(rownames(keyfitz$countryyear[keyfitz$countryyear$country
                                    == "Sweden" & keyfitz$countryyear$year > 1960,]))
Sweden_rows_1850_19 <- as.numeric(rownames(keyfitz$countryyear[keyfitz$countryyear$country
                                   == "Sweden" & keyfitz$countryyear$year > 1850
                                               & keyfitz$countryyear$year < 1901,]))

#lets calculate for Japan
Japan_rows_b60 <- as.numeric(rownames(keyfitz$countryyear[keyfitz$countryyear$country
                                  == "Japan" & keyfitz$countryyear$year < 1960,]))
Japan_rows_a60 <- as.numeric(rownames(keyfitz$countryyear[keyfitz$countryyear$country
                                  == "Japan" & keyfitz$countryyear$year > 1960,]))



##average the pops to make mean matrices
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_a60")

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 cohort are dead
h_surv_99 <- vector()
##age (year) when 95% of cohort 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 reproduction rate raw
h_mean_repo_rate <- vector()
##mean reproduction 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()

prop_h_la<- vector()

fx_h_curve <- list()
```

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

##La_h_sbins converts 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 matrices into U F and C matrices
    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]

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

    ##life expectancy conditional on reaching reproduction
    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 reproduction rate at stable state distribution
  h_mean_repo_rate_stable_state[i] <- meanRepo(matA = A_hum,
                                               matF = F_hum)

  ### mean reproduction rate not at stable state distribution
  h_mean_repo_rate[i] <- mean(F_hum[1,])



  ####calulculate the lxmx curve
  #I use h_surv_99 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 = sbins)[1])


  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)


  prop_h_la[i]<- h_spline_curve[[i]][La_h_sbins[i]]


  ###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")})
}


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,
                          prop_h_la
                         )
```

Combine the human and animal data together.

```
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,
                          prop_la = prop_h_la,

                          mass_g =  rep(trophic_data[trophic_data$species
                                    == "Homo_sapiens",
                                    "body_mass_g"],
                                    length(pop_country_year)),

                          mode_of_life = rep(trophic_data[trophic_data$species
                                     == "Homo_sapiens",
                                     "mode_of_life"],
                                     length(pop_country_year)),

                          repo_output = rep(trophic_data[trophic_data$species
                                     == "Homo_sapiens",
                                     "mass_specific_reproductive_output_g_g"],
```

```
                                    length(pop_country_year)),

                          met_rate = rep(trophic_data[trophic_data$species
                                      == "Homo_sapiens",
                                      "mass_specific_metabolic_rate_w_kg"],
                                      length(pop_country_year)),

                          IUCN = rep(trophic_data[trophic_data$species
                                      == "Homo_sapiens","IUCN_statues"],
                                     length(pop_country_year)),
                          min_max = h_surv_99)

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

Now we can clean the data before exporting it

First lets add another column relating to whether 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"
```

Some of the species population data has clear errors in them. Here we remove the most obvious ones and give the reason why.

```
pop_data <- full_m_data

##Spratt have lifespans greater then 100 years according the matrices so were removed.
pop_data_sr <- pop_data[pop_data$species != "Sprattus_sprattus",]

##The Northern muriqui also gives much longer lifespans (>105) than any record.
pop_data_sr2 <- pop_data_sr[pop_data_sr$species != "Brachyteles_hypoxanthus",]

##Scolytus_ventralis is coded in as annual but is clearly a seasonal species.
pop_data_sr3 <- pop_data_sr2[pop_data_sr2$species != "Scolytus_ventralis",]

##The study for Enhydra_lutris is a simulation based study so we remove it.
pop_data_sr4 <- pop_data_sr3[pop_data_sr3$species != "Enhydra_lutris",]

##Problems with the repoductive elements.
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 we remove any infs of Na's in the pop metrics

```
##remove infinities
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)
```

Now we match up the data to the phylogeny and make a multiphylo object.

```
##add a column 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
```

We also make a series of separate datasets and matching phylogenies for each of the additional analysis of 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")
```