Marsupial brain evo

Orlin T

28/07/2020

knitr::opts\_chunk$set(  
 warning = TRUE, # show warnings  
 message = TRUE, # show messages  
 error = TRUE, # do not interrupt generation in case of errors,  
 echo = TRUE # show R code  
)

#Load up data and tree

##Convert vars

##Imputations

data\_mice <- data  
  
data\_mice$Names <- NULL  
data\_mice$Species <- NULL  
data\_mice$Order <- NULL  
data\_mice$Family <- NULL  
data\_mice$Common.Name <- NULL  
data\_mice$Status..Endangered..Vulnerable..Common..Abundant..Rare..or.sparse...Declining..Limited. <- NULL  
data\_mice$Dimorphism <- NULL  
data\_mice$ased <- NULL  
data\_mice$BodyM <- NULL  
data\_mice$BodyF <- NULL  
  
  
data\_mice$BodyN <- log(data\_mice$BodyN)  
data\_mice$Brain <- log(data\_mice$Brain)  
data\_mice$Weaning.age <- log(data\_mice$Weaning.age)  
data\_mice$Litter.size <- log (data\_mice$Litter.size)  
data\_mice$HR <- log(data\_mice$HR)  
data\_mice$Population.density <- log(data\_mice$Population.density)  
data\_mice$FMR.Riek <- log (data\_mice$FMR.Riek)  
  
  
#Plot missing data  
  
pdf(file="missing.pdf",width=9,height=12)  
aggr\_plot <- aggr(data\_mice, col=c('navyblue','red'), numbers=TRUE, sortVars=TRUE, labels=names(data\_mice), cex.axis=.5, gap=4, ylab=c("Histogram of missing data","Pattern"))  
  
aggr\_plot  
  
  
dev.off()  
  
  
#Impute  
#Phyomice  
  
prec <- precomputePsi(tree)  
  
imp <- mice(data\_mice, m=5, meth = c("phpmm", "phpmm", "", "", "phpmm", "phpmm", "phpmm", "phpmm", "phpmm", "phpmm", "phpmm", "phpmm", "phpmm","phpmm","phpmm","phpmm", "phpmm", "phpmm"), psi = prec$psi, psiinv = prec$psiinv, maxit = 2)  
  
pred <- imp$predictorMatrix   
pred[, "FMR.Riek"] <- 0  
pred[, "Torpor"] <- 0  
pred[, "Play"] <- 0  
pred[, "Population.density"] <- 0  
pred[, "HR"] <- 0  
  
imp <- mice(data\_mice, m=5, pred = pred, meth = c("phpmm", "phpmm", "", "", "phpmm", "phpmm", "phpmm", "phpmm", "phpmm", "phpmm", "phpmm", "phpmm", "phpmm","phpmm","phpmm","phpmm", "phpmm", "phpmm"), psi = prec$psi, psiinv = prec$psiinv, maxit = 2000)  
  
save(imp, file = "imp2000.RData")  
  
plot(imp)  
stripplot(imp, pch = 20, cex = 1.2)  
  
complete(imp, 2)  
  
load("./imp25x500.Rdata")

#MCMCGlmm

#assign all the imputed datasets do dataX(num) object  
for(imputedsets in 1: length(imp$imp[[1]])) {  
 assign(paste0("dataX",imputedsets), complete(imp, imputedsets))  
 assign(paste0("dataX",imputedsets), cbind(Names = data[[1]], get(paste0("dataX",imputedsets))))  
}  
  
  
  
#Check for branch lenghts of 0 and add 0.01% of the median if this is the case  
  
tree$edge.length[which(tree$edge.length == 0)] <- 0.01\*median(tree$edge.length[-which(tree$edge.length == 0)])  
tree <- force.ultrametric(tree,method="extend")  
  
  
#Define models for MulTree  
  
formula\_dev <- Brain ~ Weaning.age + Litter.size + BodyN  
formula\_soc <- Brain ~ Group.living + Parental.care + Mating.system + Population.density + BodyN  
formula\_env <- Brain ~ DiurnalityN + Shelter.safety + Arboreality + Diet + HR + BodyN  
formula\_ori <- Brain ~ Origin \* BodyN  
formula\_vul <- Brain ~ Status \* BodyN  
formula\_tor <- Brain ~ Torpor \* BodyN  
formula\_pla <- Brain ~ Play \* BodyN  
formula\_fmr <- Brain ~ FMR.Riek \* BodyN  
  
  
#Parameters for the MCMC  
  
# Number of interations  
nitt <- 10  
# Length of burnin -> what to ignore (the initial N ot iterations)  
burnin <- 2  
# Amount of thinning -> lenght of step (sampling rate)  
thin <- 2  
  
  
#Creating mulTree objects  
  
##Creates mulTree objects = to N imputed datasets -> length(imp$imp[[1]])  
##mulTree\_data1 <- as.mulTree(data = data1, tree = tree, taxa = "Names")  
  
for(imputedsets in 1:length(imp$imp[[1]])) {  
 assign(paste("mulTree\_data",imputedsets,sep=""), as.mulTree(data = get(paste("dataX",imputedsets, sep="")), tree = tree, taxa = "Names"))  
}  
  
##removes the dataX files so they are all within the mulTree objects  
rm(list=ls(pattern="dataX"))  
  
  
#Parameter set up with priors  
  
## The formula will be set within run.mulTree  
## mul\_formula <- formula\_X  
  
# The MCMC parameters (iterations, thining, burnin)  
mul\_parameters <- c(nitt, thin, burnin)  
# The MCMCglmm priors  
mul\_priors <- list(R = list(V = 1, nu = 0.002),  
 G = list(G1 = list(V = 1, nu = 0.002)))  
  
#Run MCMCglmm on the number of imputed datasets -> length(imp$imp[[1]]) + over all formulas (check the names)  
#set formula names manually, to match the label after the \_ in the formulas above  
  
#mulTree(mulTree.data = mulTree\_data1, formula = mul\_formula, priors = mul\_priors,  
#parameters = mul\_parameters, output = "./MCMCmodels/model1", ESS = 1000,  
#chains = 2)  
  
  
  
for(imputedsets in 1 : length(imp$imp[[1]])) {  
 for (form in c("dev", "soc", "env", "ori", "vul", "tor", "pla", "fmr"))  
 mulTree(mulTree.data = get(paste0("mulTree\_data",imputedsets)), formula = get(paste0("formula\_", form)), priors = mul\_priors,  
 parameters = mul\_parameters, output = paste0("./MCMCmodels/model\_", form, "/" , "model", imputedsets), ESS = 1000, chains = 2)  
 }  
  
  
#Extracting solutions from 2 chains per model \*<<imputedsets>> and combining in a list  
  
  
#Set WD to the model of interest  
#Load imp if not loaded, as it is used for the count of the number of imputed sets  
#setwd("./MCMCmodels")  
  
#Reading all models + all chains (if more chains are run, add manually here)  
#setwd("C:/Users/uqotodor\_local/Dropbox/05. Github/Marsupial-brain-evo/MCMCmodels/model\_dev")  
setwd("C:/Users/uqotodor\_local/Dropbox/05. Github/Marsupial-brain-evo/MCMCmodels/model\_ori3")  
list() -> raw\_models\_chain1 -> raw\_models\_chain2  
for(imputedsets in 1: length(imp$imp[[1]])) {  
 raw\_models\_chain1[[imputedsets]] <- read.mulTree(paste0("model",imputedsets,"-tree1\_chain1"), model = TRUE)  
 raw\_models\_chain2[[imputedsets]] <- read.mulTree(paste0("model",imputedsets,"-tree1\_chain2"), model = TRUE)  
}  
  
raw\_models\_allchains <- c(raw\_models\_chain1, raw\_models\_chain2)  
  
## Get Sol  
# @param model this should be a single model (e.g. one chain output from mcmcglmm)  
# @return the Sol vector from one single chain  
get.Sol <- function(model) {return(model$Sol)}  
  
## Get summary of Sol  
# @param model this should be a single model (e.g. one chain output from mcmcglmm)  
# @return the summary table of the Sol  
get.summary.Sol <- function(model) {return(summary(model$Sol))}  
  
## Turn into dataframe and get statistics  
## obtain Variance (SD^2) and bind to the list  
## remove unnecessary columns  
# @param The output of get.summary.Sol (the summary of Sol)  
# @return dataframe with statistics  
get.stat.Sol <- function(summarySol) {  
 output <- as.data.frame(summarySol$statistics)  
 output$Var <- (output$SD)^2  
 output$SD <- NULL  
 output$`Naive SE` <- NULL  
 output$`Time-series SE` <- NULL  
 return(output)  
}  
  
## Run the functions and get \*results\*  
all\_Sol <- lapply(raw\_models\_allchains, get.Sol)  
all\_summaries <- lapply(raw\_models\_allchains, get.summary.Sol)  
results <- lapply(all\_summaries, get.stat.Sol)  
  
#setwd("./")  
  
##Combining solutions for density plots + percentages  
  
#using list.rbind from rlist  
solX <- as.data.frame(list.rbind(all\_Sol))  
  
##Neat up the names!  
  
names(solX)  
names(solX) <- c("Intercept", "Var1", "Var2", "VarN")  
  
  
  
#Plot density plots  
  
#plot in pdf  
  
  
pdf(file="./MCMCmodels/density.pdf")  
par( mfrow = c( 3, 3 ))  
 for(column in 1 : c(ncol(solX))) {  
 percentage\_above\_zero <- length(which(solX[,column] >= 0))/length(solX[,column])  
hdr.den(solX[,column], main = names(solX[column]), sub = paste0(round(percentage\_above\_zero\*100,2), "% above zero"), prob = c(50, 95, 99))  
abline(v = 0, lty = 1)   
 }  
dev.off()

##Pooling the results using the Rubin’s rule ##Make sure to adjust parameters!

# Using the posteriors collated in \*results\*  
  
mbar <- function (x, col=1) { # function to calculate average parameter estimates and average variances from the results list  
 vals <- rowMeans(matrix(unlist(lapply(x, function (z) z[, col])),  
 nrow=dim(x[[1]])[1], ncol=length(x)))  
 names(vals) <- rownames(x[[1]])  
 vals  
}  
  
Bm <- function (x) {  
 Qmbar <- mbar(x)  
 Qvals <- matrix(unlist(lapply(x, function (z) z[,1])),  
 ncol=length(x), nrow=dim(x[[1]])[1])  
 QQ <- apply(Qvals, 2, function (x) x - Qmbar)  
 apply(QQ, 1, function (z) (z %\*% z)/(length(z)-1))  
}  
  
Tm <- function (x) {  
 Umbar <- mbar(x, col=2)  
 Umbar + (1 + 1/length(x)) \* Bm(x)  
}  
  
lambda <- function (nu) {  
 (nu + 1)/(nu + 3)  
}  
  
vm <- function (Bm, Tm, m=5){  
 gammahat <- (1+1/m) \* sum(Bm/Tm)/length(Bm)  
 (m-1)/(gammahat^2)  
}  
  
  
# Now we do the analysis:  
  
  
#m - imputed sets \* chains  
#n - number of species  
#k - number of params as N of cols in solutions  
  
m <- (length(imp$imp[[1]]))\*2  
n <- 176  
k <- ncol(solX)  
## m = number of imputed datasets  
## n = number of observations  
## k = number of parameters  
  
Bm1 <- Bm(results)  
Tm1 <- Tm(results) # total variance  
  
# calculation of the degrees of freedom for t-tests of parameters  
  
vhatobs <- lambda(n-k)\*(n-k)\*(1- (1+1/m) \* sum(Bm1/Tm1)/length(Bm1))  
vm1 <- vm(Bm1, Tm1)  
vmtilde <- 1/(1/vm1+1/vhatobs)  
  
Qmbar <- mbar(results) # mean parameter estimates  
  
WaldT <- Qmbar/sqrt(Tm1)  
upperCI <- Qmbar + sqrt(Tm1) \* qt(.95, vmtilde)  
lowerCI <- Qmbar - sqrt(Tm1) \* qt(.95, vmtilde)  
  
tTable <- cbind(Qmbar, SE=sqrt(Tm1), WaldT,  
 df=vmtilde, p=2\*(1-pt(abs(WaldT),  
 vmtilde)), lowerCI, upperCI)

##DIC calculation

#Getting DICs and averaging (possibly other ICs)  
  
get.DIC <- function(model) {return(model$DIC)}  
  
model\_DICs <- lapply(raw\_models\_allchains, get.DIC)   
model\_DICs <- as.data.frame(model\_DICs)  
DICnames <- c(paste0("Model", 1:ncol(model\_DICs)))  
colnames(model\_DICs) <- DICnames  
DICs <- mean(t(model\_DICs))  
DICs

##H calculation

#Calculating average H for all models  
  
get.H <- function(model) {return((var(model$VCV[,"animal"]))/  
+ (var(model$VCV[,"animal"]) + var(model$VCV[,"units"])))  
 }  
  
Hs <- lapply(raw\_models\_allchains, get.H)  
Hs <- as.data.frame(Hs)  
mean(t(Hs))

##Plotting 2: Export text and plot output

#Change row names  
row.names(tTable)  
row.names(tTable) <- names(solX)  
  
#Export a csv table  
#col.names = NA offsets the header with 1  
write.table(as.matrix(tTable), "./MCMCmodels/analysis-output.csv", sep = ",", col.names = NA, row.names = TRUE)  
  
#Export txt with posteriors and Hs and DICs  
sink('./MCMCmodels/analysis-output.txt')  
#options(width=10000) stops word wrapping  
options(width=10000)  
#print pooled results  
tTable  
#print Hs  
print("Mean H")  
mean(t(Hs))  
#print DICs  
print("Mean DIC")  
DICs  
sink()  
  
#Convert data to res.plot  
res.plot <- as.data.frame(tTable)  
  
#res.plot$upperCI <- NULL  
#res.plot$lowerCI <- NULL  
res.plot$p <- NULL  
res.plot$df <- NULL  
res.plot$WaldT <- NULL  
  
#generate pdf with the model  
pdf(file="./MCMCmodels/model.pdf",width=6,height=4)  
p <- ggplot(  
 res.plot,   
 aes(x = Qmbar, y = fct\_relevel(row.names(res.plot), "VarN", "Intercept", after = Inf), xmin = lowerCI, xmax = upperCI)) +  
 geom\_point(aes(color = row.names(res.plot))) +  
 geom\_errorbarh(aes(color = row.names(res.plot)), height=0.05)+  
 geom\_vline(xintercept = 0, color = "red", linetype="longdash", size=0.5) +  
 theme\_light()  
p$labels$colour <- "Model parameters"  
p$labels$x <- "Posterior estimate + 95% CI"  
p$labels$y <- "Model parameters"  
p  
  
#generates bayesplot of the model  
color\_scheme\_set("brightblue")  
mcmc\_intervals(t(res.plot))  
  
dev.off()  
  
#optional  
mcmc\_areas(t(res.plot))

##Missingness analysis

library(naniar)  
library(visdat)  
  
#visualisation of missingness  
vis\_dat(data)  
vis\_miss(data)  
  
  
ggplot(data,   
 aes(x = Brain,   
 y = Litter.size)) +   
 geom\_miss\_point()  
  
gg\_miss\_var(data) + theme\_bw() + labs(y = "N missing values")  
  
gg\_miss\_var(data, facet = Origin)  
  
# creating shadow matrix for NA !NA comparison  
data1 <- as\_shadow(data)  
aq\_shadow <- bind\_shadow(data)  
aq\_nab <- nabular(data)  
  
library(dplyr)  
  
data %>%  
 bind\_shadow() %>%  
 group\_by(Litter.size\_NA) %>%  
 summarise\_at(.vars = "Brain",  
 .funs = c("mean", "sd", "var", "min", "max"),  
 na.rm = TRUE)  
  
  
ggplot(aq\_shadow,  
 aes(x = Brain,  
 colour = Litter.size\_NA)) +   
 geom\_density()  
  
 data %>%  
 bind\_shadow() %>%  
 ggplot(aes(x = Brain,  
 fill = Litter.size\_NA)) +  
 geom\_histogram()  
   
   
   
   
 data %>%  
 add\_prop\_miss() %>%  
 head()  
   
library(rpart)  
library(rpart.plot)  
   
 data %>%  
 add\_prop\_miss() %>%  
 rpart(prop\_miss\_all ~ ., data = .) %>%  
 prp(type = 4, extra = 101, prefix = "Prop. Miss = ")  
   
   
#Phylogenetic signal in missing data (remove edge labels first)  
   
#convert the datatset to 0 for NAs and 1s for present data   
test <- phylo.d(dataNA, tree, Names, VARIABLE.NAME, permut = 1000, rnd.bias=NULL)  
summary(test)  
plot(test)