**For phylogenetical signal analyses**

dades <- read.csv("DATA.csv", sep=";")

attach(dades)

library("phytools")

tree<-read.tree("PhytoPhylo.tre")

class(tree)="phylo"

tree <- drop.tip(tree,tip=tree$tip.label[!tree$tip.label %in% dades[,2]])

tree$node.label<- 1:length(tree$node.label)

sps <- dades$phylogeny

sps[sps %in% tree$tip.label] # Tinc

sps[!sps %in% tree$tip.label] # No Tinc

tree <- drop.tip(tree, tip=tree$tip.label[!tree$tip.label %in% sps])

class(tree) <- "phylo"

tree<- chronos(tree, lambda = 0.5, model = "correlated")

row.names(dades) <- dades$phylogeny

results <- matrix(0, 3, 42)

for (i in 3:44) {

results[1, i-2] <- colnames(dades)[i]

var <- dades[,i]

names(var) <- row.names(dades)

lambda <- phylosig(tree, x= var, method="lambda", test=TRUE, nsim=100000, se=NULL, start=NULL, control=list())

results[2, i-2] <- lambda$lambda

results[3, i-2] <- lambda$P

}

write.csv(results, file = "Phylosignal results.csv")

**For Plot phylo-tree with colours**

dades <- read.csv("DATA.csv", sep=";")

library("phytools")

tree<-read.tree("PhytoPhylo.tre")

class(tree)="phylo"

for (i in 1:231){

tree$tip.label <- sub(pattern = as.character(dades[i,2]),replacement = paste(" ",dades[i,1]," "), x= tree$tip.label)

}

sps <- paste(" ",dades$species," ")

sps[sps %in% tree$tip.label]

sps[!sps %in% tree$tip.label]

tree <- drop.tip(tree, tip=tree$tip.label[!tree$tip.label %in% sps])

class(tree) <- "phylo"

chronos(tree, lambda=0.5, model = "correlated")

x <-dades$N

names(x) <- sps

sp <- sps

sp <- sp[sp %in% tree$tip.label]

x <- x[sp]

library("RColorBrewer")

paleta<-brewer.pal(9, "RdYlBu")

obj <- contMap(tree, x)

obj<-setMap(obj, colors=paleta)

tiff(file = "Figure.tiff", width = 5000, height = 5000, units = "px", res = 350)

plot(obj,type="fan", cex=0.5)

dev.off()

**For PCA figure of soils**

library(openxlsx)

library(ggplot2)

Func <- read.xlsx("D:/Users/h.vallicrosa/Desktop/Art. Nínxol Biogeoquímic/Ninxol Biogeoquímic/Sòl/Sòls6e.xlsx")

x <- Func[,36]

y <- Func[,37]

sp <- Func[,3]

subordre <- Func[,34]

ordre <- Func[,35]

###Vectors

Names <- c("N","P","K","Ca","Mg","S","N:P","N:K","N:Ca","N:Mg","N:S","P:K","P:Ca","P:Mg","P:S","K:Ca","K:Mg","K:S","Ca:Mg","Ca:S","Mg:S")

F1i <- c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)

F1f <- c(0.062647,0.496972,-0.001200,-0.720393,-0.618248,-0.324116,-0.409596,-0.028919,0.782985,0.600063,0.319442,0.201616,0.833209,0.768185,0.557541,0.728667,0.614248,0.190497,-0.211233,-0.615485,-0.468495)

F2i <- c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)

F2f <- c(-0.283409,0.132709,0.262363,-0.194736,0.261515,-0.377699,-0.341392,-0.554097,0.004799,-0.621262,0.240356,-0.231377,0.208649,-0.379275,0.516742,0.304283,-0.301581,0.684221,-0.572117,0.172346,0.678864)

Vectors <- data.frame(Names, F1i, F1f, F2i, F2f)

colnames(Vectors)= c("Noms", "F1i", "F1f", "F2i", "F2f")

df <- data.frame(x, y, sp, subordre, ordre)

qplot(x,y, data=df, color=as.factor(ordre))

###Script

centroids <- aggregate(cbind(x,y)~class,df,mean)

ggplot(df,aes(x,y,color=factor(class))) +

geom\_point(size=3)+ geom\_point(data=centroids,size=5)

gg <- merge(df,aggregate(cbind(mean.x=x,mean.y=y)~class,df,mean),by="class")

ggplot(gg, aes(x,y,color=factor(class)))+geom\_point(size=3)+

geom\_point(aes(x=mean.x,y=mean.y),size=5)+

geom\_segment(aes(x=mean.x, y=mean.y, xend=x, yend=y))

centroids <- aggregate(cbind(x,y)~class,df,mean)

f <- function(z)sd(z)/sqrt(length(z)) # function to calculate std.err

se <- aggregate(cbind(se.x=x,se.y=y)~class,df,f)

centroids <- merge(centroids,se, by="class") # add std.err column to centroids

ggplot(gg, aes(x,y,color=factor(class)))+

geom\_point(size=3)+

geom\_point(data=centroids, size=5)+

geom\_errorbar(data=centroids,aes(ymin=y-se.y,ymax=y+se.y),width=0.1)+

geom\_errorbarh(data=centroids,aes(xmin=x-se.x,xmax=x+se.x),height=0.1)

####TUNING

centroids <- aggregate(cbind(x,y)~ordre,df,mean)

ggplot(df,aes(x,y,color=factor(ordre))) +

geom\_point(size=3)+ geom\_point(data=centroids,size=5)

gg <- merge(df,aggregate(cbind(mean.x=x,mean.y=y)~ordre,df,mean),by="ordre")

ggplot(gg, aes(x,y,color=factor(ordre)))+geom\_point(size=3)+

geom\_point(aes(x=mean.x,y=mean.y),size=5)+

geom\_segment(aes(x=mean.x, y=mean.y, xend=x, yend=y))

require(ggrepel)

centroids <- aggregate(cbind(x,y)~ordre,df,mean)

#f <- function(z) qt(0.025,df=length(z)-1, lower.tail=F)\* sd(z)/sqrt(length(z)) #95%confident

f <- function(z)sd(z)/sqrt(length(z)) # function to calculate std.err

se <- aggregate(cbind(se.x=x,se.y=y)~ordre,df,f)

centroids <- merge(centroids,se, by="ordre") # add std.err column to centroids

gg <- merge(df,aggregate(cbind(mean.x=x,mean.y=y)~ordre,df,mean),by="ordre")

ggplot(gg, aes(x,y,color=factor(ordre)))+

geom\_segment(data = Vectors, aes(xend = Vectors[ ,3], yend=Vectors[ ,5]),

x=Vectors[,2], y=Vectors[,4], colour="gray",

arrow=arrow(angle=25, length=unit(0.25, "cm")), linetype= 1)+

geom\_point(data=centroids, size=2)+

geom\_errorbar(data=centroids,aes(ymin=y-se.y,ymax=y+se.y),width=0.03)+

geom\_errorbarh(data=centroids,aes(xmin=x-se.x,xmax=x+se.x),height=0.03)+

geom\_text\_repel(data=centroids, aes(label=ordre, vjust=3.5, hjust=1.3, size=6), segment.color = 'transparent')+

geom\_text\_repel(data= Vectors, aes(x = F1f, y = F2f,label=Noms, vjust=2, hjust=0, size=6), color = 'black', segment.color = 'transparent')+

theme(panel.background = element\_blank(), axis.line = element\_line(colour = "black"))+

theme(legend.position = "none")+

theme(axis.text=element\_text(size=14))+

theme(panel.border = element\_rect(fill=NA, colour = "black", size=1))+

labs(x="PC1 (27.26%)", y="PC2 (15.53%)")+

theme(axis.title.x = element\_text(size=16),axis.title.y = element\_text(size=16))

**For Europe maps of species distribution**

library(openxlsx)

library(rgdal)

library(raster)

library(rgeos)

library(sf)

library(tidyverse)

library(gdata)

library(rworldmap)

library(ggmap)

library(maptools)

library(maps)

library(rworldxtra)

DataPoints <- read.xlsx("D:/Users/h.vallicrosa/Desktop/Art. Nínxol Biogeoquímic/Distribució/EspeciesEuropa.xlsx", sheet=3)

A.alba <- shapefile("D:/Users/h.vallicrosa/Desktop/Art. Nínxol Biogeoquímic/Distribució/Mapes/Abies alba/Abies\_alba\_EUFORGEN.shp")

#A.alba <- as(A.alba, 'SpatialPolygons')

F.sylvatica <- shapefile("D:/Users/h.vallicrosa/Desktop/Art. Nínxol Biogeoquímic/Distribució/Mapes/Fagus sylvatica/Fagus\_sylvatica\_EUFORGEN.shp")

F.sylvatica <- gBuffer(F.sylvatica, width = 0)

F.sylvatica <- as(F.sylvatica, "SpatialPolygonsDataFrame")

L.decidua <- shapefile("D:/Users/h.vallicrosa/Desktop/Art. Nínxol Biogeoquímic/Distribució/Mapes/Larix decidua/Larix\_decidua\_EUFORGEN.shp")

L.decidua <- gBuffer(L.decidua, width = 0)

L.decidua <- as(L.decidua, "SpatialPolygonsDataFrame")

P.abies <- shapefile("D:/Users/h.vallicrosa/Desktop/Art. Nínxol Biogeoquímic/Distribució/Mapes/Picea abies/Picea\_abies\_EUFORGEN.shp")

P.abies <- gBuffer(P.abies, width = 0)

P.abies <- as(P.abies, "SpatialPolygonsDataFrame")

P.sylvestris <- shapefile("D:/Users/h.vallicrosa/Desktop/Art. Nínxol Biogeoquímic/Distribució/Mapes/Pinus sylvestris/Pinus\_sylvestris\_EUFORGEN.shp")

P.sylvestris <- gBuffer(P.sylvestris, width = 0)

P.sylvestris <- as(P.sylvestris, "SpatialPolygonsDataFrame")

Q.petraea <- shapefile("D:/Users/h.vallicrosa/Desktop/Art. Nínxol Biogeoquímic/Distribució/Mapes/Quercus petraea/Quercus\_petraea\_EUFORGEN.shp")

Q.petraea <- gBuffer(Q.petraea, width = 0)

Q.petraea <- as(Q.petraea, "SpatialPolygonsDataFrame")

Q.robur <- shapefile("D:/Users/h.vallicrosa/Desktop/Art. Nínxol Biogeoquímic/Distribució/Mapes/Quercus robur/Quercus\_robur\_EUFORGEN.shp")

Q.robur <- gBuffer(Q.robur, width = 0)

Q.robur <- as(Q.robur, "SpatialPolygonsDataFrame")

##### Creació data frames per espècies #####

abiesalba <- DataPoints[1:483,]

Lat <- (abiesalba[,6])

Long <- (abiesalba[,7])

LatLongabiesalba <- matrix(c(Long, Lat), ncol = 2)

fagussylvatica <- DataPoints[569:2163,]

Lat <- (fagussylvatica[,6])

Long <- (fagussylvatica[,7])

LatLongfagussylvatica <- matrix(c(Long, Lat), ncol = 2)

larixdecidua <- DataPoints[2195:2227,]

Lat <- (larixdecidua[,6])

Long <- (larixdecidua[,7])

LatLonglarixdecidua <- matrix(c(Long, Lat), ncol = 2)

piceaabies <- DataPoints[2228:8614,]

Lat <- (piceaabies[,6])

Long <- (piceaabies[,7])

LatLongpiceaabies <- matrix(c(Long, Lat), ncol = 2)

pinussylvestris <- DataPoints[10297:17117,]

Lat <- (pinussylvestris[,6])

Long <- (pinussylvestris[,7])

LatLongpinussylvestris <- matrix(c(Long, Lat), ncol = 2)

quercuspetraea <- DataPoints[17261:17780,]

Lat <- (quercuspetraea[,6])

Long <- (quercuspetraea[,7])

LatLongquercuspetraea <- matrix(c(Long, Lat), ncol = 2)

quercusrobur <- DataPoints[17781:18389,]

Lat <- (quercusrobur[,6])

Long <- (quercusrobur[,7])

LatLongquercusrobur <- matrix(c(Long, Lat), ncol = 2)

################################## Reperesentacions gràfiques ###########################

plot(A.alba)

points(abiesalba$Long, abiesalba$Lat, add=TRUE, col="blue")

#### Obtenir les interseccions

albacembra <- crop(A.alba, P.cembra)

Latlong <- SpatialPoints(LatLongabiesalba, proj4string = CRS(proj4string(albacembra)))

Aalba2 <- over(Latlong, albacembra)

library("RColorBrewer")

brewer.pal(12, "Paired")

display.brewer.pal(12, "Paired")

### Picea abies vs Quercus robur

tiff("Pa&Qr.tiff", res=100, height=800, width = 1100)

newmap <- getMap(resolution = "high")

plot(newmap, main="Picea abies VS Quercus robur", xlim = c(-20, 49), ylim = c(35, 71), asp = 1)

abiesrobur <- crop(P.abies,Q.robur)

plot(P.abies, add=TRUE, col="#A6CEE3")

plot(Q.robur, add=TRUE, col="#FFFF99")

plot(abiesrobur, add=TRUE, col="#B2DF8A")

points(piceaabies$Long, piceaabies$Lat, pch=20, col="#B15928", cex=1)

points(quercusrobur$Long, quercusrobur$Lat, pch=20, col="#33A02C", cex=1)

dev.off()

# ### Abies alba vs Picea abies

# newmap <- getMap(resolution = "high")

# plot(newmap, main="Abies alba VS Picea abies", xlim = c(-20, 59), ylim = c(35, 71), asp = 1)

# albaabies <- crop(A.alba, P.abies)

# plot(A.alba, add=TRUE, col="red")

# plot(P.abies, add=TRUE, col="yellow")

# plot(abiesrobur, add=TRUE, col="orange")

# points(abiesalba$Long, abiesalba$Lat, col="blue")

# points(piceaabies$Long, piceaabies$Lat, col="green")

# ### Quercus petraea vs Quercus robur

# newmap <- getMap(resolution = "high")

# plot(newmap, main="Quercus petraea VS Quercus robur", xlim = c(-20, 59), ylim = c(35, 71), asp = 1)

# petraearobur <- crop(Q.petraea, Q.robur)

# plot(Q.petraea, add=TRUE, col="red")

# plot(Q.robur, add=TRUE, col="yellow")

# plot(petraearobur, add=TRUE, col="orange")

# points(quercuspetraea$Long, quercuspetraea$Lat, col="blue")

# points(quercusrobur$Long, quercusrobur$Lat, col="green")

# ### Fagus sylvativa VS Larix decidua

# newmap <- getMap(resolution = "high")

# plot(newmap, main="Fagus sylvatica VS Larix decidua", xlim = c(-20, 59), ylim = c(35, 71), asp = 1)

# sylvaticadecidua <- crop(F.sylvatica, L.decidua)

# plot(F.sylvatica, add=TRUE, col="red")

# plot(L.decidua, add=TRUE, col="yellow")

# plot(sylvaticadecidua, add=TRUE, col="orange")

# points(fagussylvatica$Long, fagussylvatica$Lat, col="blue")

# points(larixdecidua$Long, larixdecidua$Lat, col="green")

# ### Fagus sylvatica VS Abies alba

# newmap <- getMap(resolution = "high")

# plot(newmap, main="Fagus sylvatica VS Abies alba", xlim = c(-15, 49), ylim = c(39, 71), asp = 1)

# sylvaticadecidua <- crop(F.sylvatica, A.alba)

# plot(F.sylvatica, add=TRUE, col="red")

# plot(A.alba, add=TRUE, col="yellow")

# plot(sylvaticadecidua, add=TRUE, col="orange")

# points(fagussylvatica$Long, fagussylvatica$Lat, col="blue")

# points(abiesalba$Long, abiesalba$Lat, col="green")

### Fagus sylvatica vs Pinus sylvestris

tiff("Fs&Ps.tiff", res=100, height=800, width = 1100)

newmap <- getMap(resolution = "high")

plot(newmap, main="Fagus sylvatica VS Pinus sylvestris", xlim = c(-20, 59), ylim = c(35, 71), asp = 1)

sylvaticasilvestris <- crop(F.sylvatica, P.sylvestris)

plot(F.sylvatica, add=TRUE, col="#A6CEE3")

plot(P.sylvestris, add=TRUE, col="#FFFF99")

plot(sylvaticasilvestris, add=TRUE, col="#B2DF8A")

points(fagussylvatica$Long, fagussylvatica$Lat, pch=20, col="#B15928", cex=1)

points(pinussylvestris$Long, pinussylvestris$Lat, pch=20, col="#33A02C", cex=1)

dev.off()

### Abies alba vs Quercus petraea

tiff("Aa&Qp.tiff", res=100, height=800, width = 1100)

newmap <- getMap(resolution = "high")

plot(newmap, main="Abies alba VS Quercus petraea", xlim = c(-20, 59), ylim = c(35, 71), asp = 1)

albapetraea <- crop(A.alba, Q.petraea)

plot(A.alba, add=TRUE, col="#A6CEE3")

plot(Q.petraea, add=TRUE, col="#FFFF99")

plot(albapetraea, add=TRUE, col="#B2DF8A")

points(abiesalba$Long, abiesalba$Lat, pch=20, col="#B15928", cex=1)

points(quercuspetraea$Long, quercuspetraea$Lat, pch=20, col="#33A02C", cex=1)

dev.off()

# ### Abies alba vs Quercus robur

# newmap <- getMap(resolution = "high")

# plot(newmap, main="Abies alba VS Quercus robur", xlim = c(-20, 59), ylim = c(35, 71), asp = 1)

# albarobur <- crop(A.alba, Q.robur)

# plot(A.alba, add=TRUE, col="red")

# plot(Q.robur, add=TRUE, col="yellow")

# plot(albarobur, add=TRUE, col="orange")

# points(abiesalba$Long, abiesalba$Lat, col="blue")

# points(quercusrobur$Long, quercusrobur$Lat, col="green")

# ### Picea abies vs Fagus sylvatica

# newmap <- getMap(resolution = "high")

# plot(newmap, main="Picea abies VS Fagus sylvatica", xlim = c(-20, 59), ylim = c(35, 71), asp = 1)

# abiessylvatica <- crop(P.abies, F.sylvatica)

# plot(P.abies, add=TRUE, col="red")

# plot(F.sylvatica, add=TRUE, col="yellow")

# plot(abiessylvatica, add=TRUE, col="orange")

# points(piceaabies$Long, piceaabies$Lat, col="blue")

# points(fagussylvatica$Long, fagussylvatica$Lat, col="green")

# ### Pinus sylvestris vs Quercus petraea

# newmap <- getMap(resolution = "high")

# plot(newmap, main="Pinus sylvestris vs Quercus petraea", xlim = c(-20, 59), ylim = c(35, 71), asp = 1)

# sylvestrispetraea <- crop(P.sylvestris, Q.petraea)

# plot(P.sylvestris, add=TRUE, col="red")

# plot(Q.petraea, add=TRUE, col="yellow")

# plot(sylvestrispetraea, add=TRUE, col="orange")

# points(pinussylvestris$Long, pinussylvestris$Lat, col="blue")

# points(quercuspetraea$Long, quercuspetraea$Lat, col="green")

**For soil models (effect of soil order)**

library(openxlsx)

library(nlme)

library(MuMIn)

ModClima <- read.xlsx("D:/Users/h.vallicrosa/Desktop/Art. Nínxol Biogeoquímic/Ninxol Biogeoquímic/Models/VarClim6esp3.xlsx")

ModClim<-na.exclude(ModClima)

###### N

n <-lme(log(N) ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(n)

hist(resid(n))

r.squaredGLMM(n)

anova.lme(n, type = "marginal", adjustSigma = F)

n2 <- lm(N~Ordre+SPECIE, data=ModClim)

hist(resid(n2))

summary(n2)

anova(n2)

###### P

p <-lme(log(P) ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(p)

hist(resid(p))

r.squaredGLMM(p)

anova.lme(p, type = "marginal", adjustSigma = F)

p2 <- lm(log(P)~Ordre+SPECIE, data=ModClim)

hist(resid(p2))

summary(p2)

anova(p2)

###### K

k <-lme(log(K) ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(k)

hist(resid(k))

r.squaredGLMM(k)

anova.lme(k, type = "marginal", adjustSigma = F)

k2 <- lm(K~Ordre+SPECIE, data=ModClim)

hist(resid(k2))

summary(k2)

anova(k2)

###### Ca

ca <-lme(log(Ca) ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(ca)

hist(resid(ca))

r.squaredGLMM(ca)

anova.lme(ca, type = "marginal", adjustSigma = F)

ca2 <- lm(Ca~Ordre+SPECIE, data=ModClim)

hist(resid(ca2))

summary(ca2)

anova(ca2)

###### Mg

mg <-lme(log(Mg) ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(mg)

hist(resid(mg))

r.squaredGLMM(mg)

anova.lme(mg, type = "marginal", adjustSigma = F)

mg2 <- lm(Mg~Ordre+SPECIE, data=ModClim)

hist(resid(mg2))

summary(mg2)

anova(mg2)

###### S

s <-lme(log(S) ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(s)

hist(resid(s))

r.squaredGLMM(s)

anova.lme(s, type = "marginal", adjustSigma = F)

s2 <- lm(S~Ordre+SPECIE, data=ModClim)

hist(resid(s2))

summary(s2)

anova(s2)

###### NP

np <-lme(log(NP) ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(np)

hist(resid(np))

r.squaredGLMM(np)

anova.lme(np, type = "marginal", adjustSigma = F)

np2 <- lm(sqrt(NP)~Ordre+SPECIE, data=ModClim)

hist(resid(np2))

summary(np2)

anova(np2)

###### NK

nk <-lme(log(NK) ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(nk)

hist(resid(nk))

r.squaredGLMM(nk)

anova.lme(nk, type = "marginal", adjustSigma = F)

nk2 <- lm(log(NK)~Ordre+SPECIE, data=ModClim)

hist(resid(nk2))

summary(nk2)

anova(nk2)

###### NCa

nca <-lme(sqrt(NCa) ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(nca)

hist(resid(nca))

r.squaredGLMM(nca)

anova.lme(nca, type = "marginal", adjustSigma = F)

nca2 <- lm(log(NCa)~Ordre+SPECIE, data=ModClim)

hist(resid(nca2))

summary(nca2)

anova(nca2)

###### NMg

nmg <-lme(log(NMg) ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(nmg)

hist(resid(nmg))

r.squaredGLMM(nmg)

anova.lme(nmg, type = "marginal", adjustSigma = F)

nmg2 <- lm(log(NMg)~Ordre+SPECIE, data=ModClim)

hist(resid(nmg2))

summary(nmg2)

anova(nmg2)

###### NS

ns <-lme(log(NS) ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(ns)

hist(resid(ns))

r.squaredGLMM(ns)

anova.lme(ns, type = "marginal", adjustSigma = F)

ns2 <- lm(log(NS)~Ordre+SPECIE, data=ModClim)

hist(resid(ns2))

summary(ns2)

anova(ns2)

###### PK

pk <-lme(log(PK) ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(pk)

hist(resid(pk))

r.squaredGLMM(pk)

anova.lme(pk, type = "marginal", adjustSigma = F)

pk2 <- lm(log(PK)~Ordre+SPECIE, data=ModClim)

hist(resid(pk2))

summary(pk2)

anova(pk2)

###### PCa

pca <-lme(sqrt(PCa) ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(pca)

hist(resid(pca))

r.squaredGLMM(pca)

anova.lme(pca, type = "marginal", adjustSigma = F)

pca2 <- lm(log(PCa)~Ordre+SPECIE, data=ModClim)

hist(resid(pca2))

summary(pca2)

anova(pca2)

###### PMg

pmg <-lme(log(PMg) ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(pmg)

hist(resid(pmg))

r.squaredGLMM(pmg)

anova.lme(pmg, type = "marginal", adjustSigma = F)

pmg2 <- lm(log(PMg)~Ordre+SPECIE, data=ModClim)

hist(resid(pmg2))

summary(pmg2)

anova(pmg2)

###### PS

ps <-lme(log(PS) ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(ps)

hist(resid(ps))

r.squaredGLMM(ps)

anova.lme(ps, type = "marginal", adjustSigma = F)

ps2 <- lm(log(PS)~Ordre+SPECIE, data=ModClim)

hist(resid(ps2))

summary(ps2)

anova(ps2)

###### KCa

kca <-lme(sqrt(KCa) ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(kca)

hist(resid(kca))

r.squaredGLMM(kca)

anova.lme(kca, type = "marginal", adjustSigma = F)

kca2 <- lm(log(KCa)~Ordre+SPECIE, data=ModClim)

hist(resid(kca2))

summary(kca2)

anova(kca2)

###### KMg

kmg <-lme(log(KMg) ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(kmg)

hist(resid(kmg))

r.squaredGLMM(kmg)

anova.lme(kmg, type = "marginal", adjustSigma = F)

kmg2 <- lm(log(KMg)~Ordre+SPECIE, data=ModClim)

hist(resid(kmg2))

summary(kmg2)

anova(kmg2)

###### KS

ks <-lme(log(KS) ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(ks)

hist(resid(ks))

r.squaredGLMM(ks)

anova.lme(ks, type = "marginal", adjustSigma = F)

ks2 <- lm(log(KS)~Ordre+SPECIE, data=ModClim)

hist(resid(ks2))

summary(ks2)

anova(ks2)

###### CaMg

camg <-lme(log(CaMg) ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(camg)

hist(resid(camg))

r.squaredGLMM(camg)

anova.lme(camg, type = "marginal", adjustSigma = F)

camg2 <- lm(log(CaMg)~Ordre+SPECIE, data=ModClim)

hist(resid(camg2))

summary(camg2)

anova(camg2)

###### CaS

cas <-lme(log(CaS) ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(cas)

hist(resid(cas))

r.squaredGLMM(cas)

anova.lme(cas, type = "marginal", adjustSigma = F)

cas2 <- lm(CaS~Ordre+SPECIE, data=ModClim)

hist(resid(cas2))

summary(cas2)

anova(cas2)

###### MgS

mgs <-lme(sqrt(MgS) ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(mgs)

hist(resid(mgs))

r.squaredGLMM(mgs)

anova.lme(mgs, type = "marginal", adjustSigma = F)

mgs2 <- lm(sqrt(MgS)~Ordre+SPECIE, data=ModClim)

hist(resid(mgs2))

summary(mgs2)

anova(mgs2)

###### Score 1

s1 <-lme(Score.1 ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(s1)

hist(resid(s1))

r.squaredGLMM(s1)

anova.lme(s1, type = "marginal", adjustSigma = F)

s12 <- lm(log(Score.1+9)~Ordre+SPECIE, data=ModClim)

hist(resid(s12))

summary(s12)

anova(s12)

###### Score 2

s2 <-lme(Score.2 ~ Ordre, random=~1|SPECIE, data=ModClim )

summary(s2)

hist(resid(s2))

r.squaredGLMM(s2)

anova.lme(s2, type = "marginal", adjustSigma = F)

s22 <- lm(Score.2~Ordre+SPECIE, data=ModClim)

hist(resid(s22))

summary(s22)

anova(s22)

**For Bayesian phylogenetic mixed models with climate, N deposition and soil variables as independent fixed factors.**

#### Sardans et al., Nature Ecology & Evolution ####

# Bayesian analyses #

# Load functions and packages first

#### Load data ####

dades <- read.csv(file="./VarClim6esp3.csv", sep=";")

library("phytools")

tree<-read.tree("./PhytoPhylo.tre")

class(tree)="phylo"

dades$sps <- as.character("NA")

for (i in 1:nrow(dades)){

a <- strsplit(x = as.character(dades$SPECIE[i]), split = " ")

dades$sps [i] <- paste(a[[1]][1], a[[1]][2], sep="\_")

}

# chose species in db

unique(dades$sps)[unique(dades$sps) %in% tree$tip.label] # Tinc

unique(dades$sps)[!unique(dades$sps) %in% tree$tip.label] # No Tinc

tree<- compute.brlen(tree, power=0.5)

is.ultrametric(tree) #Yes!

tree.pr <- drop.tip(tree,tip=tree$tip.label[!tree$tip.label %in% unique(dades$sps)])

tree.pr$node.label<- 1:length(tree.pr$node.label)

subdata <- dades[which(dades$sps %in%tree.pr$tip.label),]

unique(subdata$sps)[!unique(subdata$sps) %in% tree.pr$tip.label] # No Tinc

subdata$animal <- subdata$sps

subdata <- subdata[,c("Lat", "Long", "N", "P", "K", "Ca", "Mg", "S", "NP", "NK", "NCa", "NMg", "NS", "PK", "PCa", "PMg",

"PS", "KCa", "KMg", "KS", "CaMg", "CaS", "MgS",

"TminAvg","VapAvc","NDep","MAT","SradAvg","MAP","Isothermality","DiurnalRange","sps","animal", "Score.1", "Score.2", "Score.3", "Ordre", "Subordre")]

subdata <- na.omit(subdata)

unique(subdata$sps)[!unique(subdata$sps) %in% tree.pr$tip.label] # No Tinc

subdata$Ordre <- droplevels(subdata$Ordre)

table(subdata$Ordre)

#### Load ICC functions ####

## functions for MCMCglmm

# From Smith et al. (2016) â??Risk and resilience:

# variations in magnesium in echinoid skeletal calciteâ??.

random\_intercepts\_prior <- function(n, V=1, nu=0.002){

R\_=list(R1=list(V=V, nu=nu))

G\_=replicate(n,list(V=V, nu=nu), simplify=FALSE)

names(G\_) <-paste("G", 1:n, sep="")

return(list(R=R\_, G=G\_))

}

DIC <- function(...){

vals <-round(sapply(list(...), "[[", "DIC"),2)

paste0(vals, " (",round(vals-min(vals),2), ")")

}

R2 <- function(mod){

fixed\_eff <-colMeans(mod$Sol)

fixed\_var\_comp <-var(as.vector(fixed\_eff %\*%t(mod$X)))

all\_randoms <-colMeans(mod$VCV)

residual <- all\_randoms[["units"]]

random\_var\_comp <-sum(all\_randoms) - residual

R2 <- (fixed\_var\_comp + random\_var\_comp)/(sum(all\_randoms) + fixed\_var\_comp)

round(R2,3)

}

R2.many <- function(...) sapply(list(...), R2)

estimate\_and\_cis <- function(mod\_summary\_element){

apply(mod\_summary\_element, 1, function(x) do.call(sprintf,c("%.3f (%.3f -- %.3f)",as.list(x[1:3])) ))}

extract\_all\_estimates <- function(mod){

summ <-summary(mod)

list( fixed =estimate\_and\_cis(summ$solutions),

random =estimate\_and\_cis(summ$Gcovariances),

residual =estimate\_and\_cis(summ$Rcovariances))

}

extract\_one\_estimate <- function(mod\_ests, param, type){

if(param %in%names(mod\_ests[[type]])){

return(mod\_ests[[type]][[param]])

}

"-"

}

extract\_estimate <- function (mod\_ests, param, type){

sapply(mod\_ests, extract\_one\_estimate, param=param, type=type)

}

estimate\_table <- function(..., to\_include=NULL){

estimates <-lapply(list(...), extract\_all\_estimates)

if(is.null(to\_include)){ #Use every variable if none are specified

types <-c("fixed", "random", "residual")

all\_vars <-lapply(types, function(type)

unique(unlist(lapply(estimates, function(x)names(x[[type]])))))

oo\_include <-cbind(unlist(all\_vars),rep(types,sapply(all\_vars, length)))

}

res <-apply(to\_include, 1, function(x) extract\_estimate(estimates, x[1], x[2]))

colnames(res) <- to\_include[,1]

as.data.frame(res)

}

summary\_table <- function(names, to\_include, ... ){

data.frame(model= names,

DIC =DIC(...),

R2 =R2.many(...),

estimate\_table(..., to\_include=to\_include))

}

# Maspon's functions for ICC on MCMCglmm and brms

icc<- function(x, ...) UseMethod("icc")

icc.brmsfit<- function(x){

pars<- brms::parnames(x)

varRandom<- grep("^(sd\_|sigma)", pars, value=TRUE)

varRandom<- paste0(varRandom, "^2")

varTotal<- paste(varRandom, collapse=" + ") # total variance

h<- paste0(varRandom, " / (", varTotal, ") = 0")

res<- brms::hypothesis(x, h, class=NULL)

rownames(res$hypothesis)<- gsub("\\/\\(.\*\\) = 0", "/total\_variance = 0", rownames(res$hypothesis))

return(res)

}

icc.mcmc<- function(x){

totalVar<- rowSums(x) # Total variance

varsVar<- coda::mcmc(apply(x, MARGIN=2, function(x) x / totalVar)) ## but see pÃ g. 50 vignette("CourseNotes", package="MCMCglmm")

return (varsVar)

}

icc.MCMCglmm<- function(x){

return (icc.mcmc(x$VCV))

}

icc.default<- icc.mcmc

#### Start analyses ####

library(ape)

library(MCMCglmm)

library(MuMIn)

MCMCglmm.updateable<- updateable(MCMCglmm)

subdata$animal <- as.factor(subdata$animal)

subdata$sps <- as.factor(subdata$sps)

Nnitt=1000000

Nthin=100

Nburnin=2000

# estimating fixed and random effects

# from: https://stat.ethz.ch/pipermail/r-sig-mixed-models/2015q3/023861.html

# by the guy of the marginal and conditional R2's of MuMIn

# MCMCglmm (it is probably better to get a posterior distribuiton of R2

# rather than getting each varaince component - we do this below as an

# alternative)

rsq.mcmc <- function(mmF){ # for 6 predictors + intercept, species and phylogeny

# mmF <- mod2

mFixed <- mean(mmF$Sol[,2]) \* mmF$X[, 2] + mean(mmF$Sol[, 3]) \* mmF$X[, 3] + mean(mmF$Sol[ ,4]) \* mmF$X[, 4] +

mean(mmF$Sol[ ,5]) \* mmF$X[, 5] + mean(mmF$Sol[ ,6]) \* mmF$X[, 6] + mean(mmF$Sol[ ,7]) \* mmF$X[, 7]

# Calculation of the variance in fitted values

mVarF<- var(mFixed)

# MCMCglmm - marginal

mVarF/(mVarF+sum(apply(mmF$VCV,2,mean)))

# alternative with crebile intervals

vmVarF<-numeric(nrow(mmF$VCV))

for(i in 1:nrow(mmF$VCV)){

Var<-var(as.vector(mmF$Sol[i,] %\*% t(mmF$X)))

vmVarF[i]<-Var

}

R2m<-vmVarF/(vmVarF+mmF$VCV[,1]+mmF$VCV[,2]+mmF$VCV[,3]) # remove the last if only 1 random

# R2m<-vmVarF/(vmVarF+mmF$VCV[,1]+mmF$VCV[,2]) #

mean(R2m)

# posterior.mode(R2m)

# HPDinterval(R2m)

# MCMCglmm - conditional

# (mVarF+sum(apply(mmF$VCV,2,mean)[-3]))/(mVarF+sum(apply(mmF$VCV,2,mean))) # if only 1 random, [-2] (below)

# (mVarF+sum(apply(mmF$VCV,2,mean)[-2]))/(mVarF+sum(apply(mmF$VCV,2,mean))) # if only 1 random, [-2]

# alternative with crebile intervals

R2c<-(vmVarF+mmF$VCV[,1]+mmF$VCV[,2])/(vmVarF+mmF$VCV[,1]+mmF$VCV[,2]+mmF$VCV[,3]) # if only 1 random, remove the second

# R2c<-(vmVarF+mmF$VCV[,1])/(vmVarF+mmF$VCV[,1]+mmF$VCV[,2]) #

# mean(R2c)

# posterior.mode(R2c)

# HPDinterval(R2c)

return(data.frame(R2m=mean(R2m), R2c=mean(R2c)))

}

ccmods <- function(mod){

# effecte especies

species <- mod$VCV[, "sps"]/(mod$VCV[, "animal"] + mod$VCV[, "sps"] + mod$VCV[, "units"]) # posterior distribution of the plot effect, proportion of variance explained by the random effect

# effectiveSize(species)

# posterior.mode(species)

# median(species);mean(species)

# HPDinterval(species)

# plot(species)

# efecte animal

phylo <- mod$VCV[, "animal"]/(mod$VCV[, "animal"] + mod$VCV[, "sps"] + mod$VCV[, "units"]) # posterior distribution of the phylo effect, proportion of variance explained by the random effect

# effectiveSize(phylo)

# posterior.mode(phylo)

# median(phylo);mean(phylo)

# HPDinterval(phylo)

# plot(phylo)

# efecte units

units <- mod$VCV[, "units"]/(mod$VCV[, "animal"] + mod$VCV[, "sps"] + mod$VCV[, "units"]) # posterior distribution of the units effect, proportion of variance explained by the random effect

# effectiveSize(units)

# posterior.mode(units)

# median(units);mean(units)

# HPDinterval(units)

# plot(units)

return(data.frame(phylo=mean(phylo), species=mean(species), units=mean(units)))

}

# Ordre is soil

Nnitt=1050000

Nthin=200

Nburnin=2000

mod.n <- MCMCglmm.updateable(scale(log(N)) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.n, file="./rev2\_mod.n.Rdata")

a <- Sys.time()

mod.p <- MCMCglmm.updateable(scale(log(P)) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.p, file="./rev2\_mod.p.Rdata")

mod.k <- MCMCglmm.updateable(scale(log(K)) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.k, file="./rev2\_mod.k.Rdata")

mod.ca <- MCMCglmm.updateable(scale(log(Ca)) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.ca, file="./rev2\_mod.ca.Rdata")

mod.mg <- MCMCglmm.updateable(scale(log(Mg)) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.mg, file="./rev2\_mod.mg.Rdata")

mod.s <- MCMCglmm.updateable(scale(log(S)) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.s, file="./rev2\_mod.s.Rdata")

mod.np <- MCMCglmm.updateable(scale(log(NP)) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.np, file="./rev2\_mod.np.Rdata")

mod.nk <- MCMCglmm.updateable(scale(log(NK)) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.nk, file="./rev2\_mod.nk.Rdata")

mod.nca <- MCMCglmm.updateable(scale(log(NCa)) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.nca, file="./rev2\_mod.nca.Rdata")

Sys.time() - a

# check

rsq.mcmc(mod.nca)

ccmods(mod.nca)

# autocorr.diag(mod.nca$Sol)

# autocorr.diag(mod.nca$VCV)

effectiveSize(mod.nca$Sol)

effectiveSize(mod.nca$VCV)

a <- Sys.time()

mod.nmg <- MCMCglmm.updateable(scale(log(NMg)) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.nmg, file="./rev2\_mod.nmg.Rdata")

mod.ns <- MCMCglmm.updateable(scale(log(NS)) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.ns, file="./rev2\_mod.ns.Rdata")

mod.pk <- MCMCglmm.updateable(scale(log(PK)) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.pk, file="./rev2\_mod.pk.Rdata")

mod.pca <- MCMCglmm.updateable(scale(log(PCa)) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.pca, file="./rev2\_mod.pca.Rdata")

mod.pmg <- MCMCglmm.updateable(scale(log(PMg)) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.pmg, file="./rev2\_mod.pmg.Rdata")

mod.ps <- MCMCglmm.updateable(scale(log(NS)) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.ps, file="./rev2\_mod.ps.Rdata")

mod.kca <- MCMCglmm.updateable(scale(log(KCa)) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.kca, file="./rev2\_mod.kca.Rdata")

mod.kmg <- MCMCglmm.updateable(scale(log(KMg)) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.kmg, file="./rev2\_mod.kmg.Rdata")

mod.ks <- MCMCglmm.updateable(scale(log(KS)) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.ks, file="./rev2\_mod.ks.Rdata")

mod.camg <- MCMCglmm.updateable(scale(log(CaMg)) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.camg, file="./rev2\_mod.camg.Rdata")

mod.cas <- MCMCglmm.updateable(scale(log(CaS)) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.cas, file="./rev2\_mod.cas.Rdata")

mod.mgS <- MCMCglmm.updateable(scale(log(MgS)) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.mgS, file="./rev2\_mod.mgS.Rdata")

mod.scr1 <- MCMCglmm.updateable(scale(Score.1) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.scr1, file="./rev2\_mod.scr1.Rdata")

mod.scr2 <- MCMCglmm.updateable(scale(Score.2) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.scr2, file="./rev2\_mod.scr2.Rdata")

mod.scr3 <- MCMCglmm.updateable(scale(Score.3) ~ scale(MAT) + scale(MAP) + scale(VapAvc) + scale(NDep) + scale(SradAvg) + scale(DiurnalRange), random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(mod.scr3, file="./rev2\_mod.scr3.Rdata")

Sys.time() - a

rsq.mcmc(mod.scr3)

ccmods(mod.scr3)

# autocorr.diag(mod.nmg$Sol)

# autocorr.diag(mod.nmg$VCV)

effectiveSize(mod.scr3$Sol)

effectiveSize(mod.scr3$VCV)

load(file="./rev2\_modsoil.n.Rdata")

# Soil models

a <- Sys.time()

modsoil.n <- MCMCglmm.updateable(scale(log(N)) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.n, file="./rev2\_modsoil.n.Rdata")

modsoil.p <- MCMCglmm.updateable(scale(log(P)) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.p, file="./rev2\_modsoil.p.Rdata")

modsoil.k <- MCMCglmm.updateable(scale(log(K)) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.k, file="./rev2\_modsoil.k.Rdata")

modsoil.ca <- MCMCglmm.updateable(scale(log(Ca)) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.ca, file="./rev2\_modsoil.ca.Rdata")

modsoil.mg <- MCMCglmm.updateable(scale(log(Mg)) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.mg, file="./rev2\_modsoil.mg.Rdata")

modsoil.s <- MCMCglmm.updateable(scale(log(S)) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.s, file="./rev2\_modsoil.s.Rdata")

modsoil.np <- MCMCglmm.updateable(scale(log(NP)) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.np, file="./rev2\_modsoil.np.Rdata")

modsoil.nk <- MCMCglmm.updateable(scale(log(NK)) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.nk, file="./rev2\_modsoil.nk.Rdata")

modsoil.nca <- MCMCglmm.updateable(scale(log(NCa)) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.nca, file="./rev2\_modsoil.nca.Rdata")

modsoil.nmg <- MCMCglmm.updateable(scale(log(NMg)) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.nmg, file="./rev2\_modsoil.nmg.Rdata")

modsoil.ns <- MCMCglmm.updateable(scale(log(NS)) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.ns, file="./rev2\_modsoil.ns.Rdata")

modsoil.pk <- MCMCglmm.updateable(scale(log(PK)) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.pk, file="./rev2\_modsoil.pk.Rdata")

modsoil.pca <- MCMCglmm.updateable(scale(log(PCa)) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.pca, file="./rev2\_modsoil.pca.Rdata")

modsoil.pmg <- MCMCglmm.updateable(scale(log(PMg)) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.pmg, file="./rev2\_modsoil.pmg.Rdata")

modsoil.ps <- MCMCglmm.updateable(scale(log(NS)) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.ps, file="./rev2\_modsoil.ps.Rdata")

modsoil.kca <- MCMCglmm.updateable(scale(log(KCa)) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.kca, file="./rev2\_modsoil.kca.Rdata")

modsoil.kmg <- MCMCglmm.updateable(scale(log(KMg)) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.kmg, file="./rev2\_modsoil.kmg.Rdata")

modsoil.ks <- MCMCglmm.updateable(scale(log(KS)) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.ks, file="./rev2\_modsoil.ks.Rdata")

modsoil.camg <- MCMCglmm.updateable(scale(log(CaMg)) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.camg, file="./rev2\_modsoil.camg.Rdata")

modsoil.cas <- MCMCglmm.updateable(scale(log(CaS)) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.cas, file="./rev2\_modsoil.cas.Rdata")

modsoil.mgS <- MCMCglmm.updateable(scale(log(MgS)) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.mgS, file="./rev2\_modsoil.mgS.Rdata")

modsoil.scr1 <- MCMCglmm.updateable(scale(Score.1) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.scr1, file="./rev2\_modsoil.scr1.Rdata")

modsoil.scr2 <- MCMCglmm.updateable(scale(Score.2) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.scr2, file="./rev2\_modsoil.scr2.Rdata")

modsoil.scr3 <- MCMCglmm.updateable(scale(Score.3) ~ Ordre, random=~animal+sps, data= subdata, pedigree=tree.pr, nodes='TIPS', family="gaussian", verbose=T, prior=random\_intercepts\_prior(2), nitt=Nnitt, burnin=Nburnin, thin=Nthin)

save(modsoil.scr3, file="./rev2\_modsoil.scr3.Rdata")

Sys.time() - a

rsq.mcmc(modsoil.n)

ccmods(modsoil.n)

heidel.diag(modsoil.n$VCV)

heidel.diag(modsoil.n$Sol)

autocorr.diag(modsoil.n$Sol)

autocorr.diag(modsoil.n$VCV)

effectiveSize(modsoil.n$Sol)

effectiveSize(modsoil.n$VCV)

summary(modsoil.n)

# post hoc soil models #

library (emmeans)

library (MCMCglmm)

summary(modsoil.scr1)

emmeans(as.mcmc(modsoil.scr1), pairwise ~ Ordre, data= subdata, adjust="bonferroni") # pots canviar bonferroni per tukey etc... fes help(emmeans) per veure les opcions

emmeans(modsoil.scr1, scale(Score.1) ~ Ordre, random=~animal+sps, data= subdata, adjust="bonferroni")

modsoil.scr3 %>%

emmeans( ~ Ordre, data = subdata) %>%

contrast(method = "pairwise") # %>%

# gather\_emmeans\_draws() # %>%

# ggplot(aes(x = .value, y = contrast)) +

# stat\_halfeyeh()