**Null and regression models in ecology**

Guillaume Latombe

During the last practical, you learned how to compute different ecological patterns in R. In particular, you computed patterns of alpha and beta diversity. However, as we saw during the lecture, beta diversity values may simply be the result of how widespread some species are and how rich sites are. During this practical, you will use null models based on permutation algorithms, along with regression models, to explore and interpret alpha and beta diversity values for established alien species in French Polynesia (Society archipelago) and in Hawai’i, and explore what drives species richness and turnover in these two archipelagos.

Before starting the practical, load the following packages (and install them if they are not on your machine yet):

library(vegan)

library(gam)

library(scam)

library(gdm)

library(car)

library(betapart)

library(tidyverse)

library(zetadiv)

**Null models: permutation algorithms**

I already created the data frames you will need for these analyses, and saved them as an RData file. You can load them by calling the following function:

load("islands\_Soc\_Haw\_null\_models\_practical.RData")

First, you need to compute beta diversity values for the two archipelagos. Apply function beta.pair() to the dat.soc.pa data frame. This will generate a list, with the Sorensen, Simpson, and nestedness beta diversity values stored in each element of the list as a distance matrix format (you will get one value for each pair of islands). Compute the average beta diversity value across islands for each index.

We now need to assess if these values are different from what we can expect from chance alone. To do so, we will use permutation algorithm to generate beta diversity values under randomness. Look at the help file for the permatfull() function. For the Society archipelago, variable m will be dat.soc.pa. We will only do 99 permutations to keep computation times reasonable (but in practice, it would be better to use 999 permutations). Set the corresponding parameter to 99. We have presence-absence matrices, so we need to define mtype="prab".

Finally, we need to define if we want to impose constraints on the presence-absence matrix during the permutations. We can either impose no constraint at all (fixedmar = "none"), fix row sums (fixedmar = "rows"), fix column sums (fixedmar = "columns"), or both (fixedmar = "both"). What does fixing rows or column sums mean in ecological terms? Which option should we choose?

Compute these permutations for all four options, and store them in lists named dat.soc.pa.perm.none, dat.soc.pa.perm.row, dat.soc.pa.perm.col and dat.soc.pa.perm.both (the last one will take a couple of minutes). Take a moment to explore the structure of these list (what data they contain).

You can load the lists for Hawai’i by calling load("permutations\_hawaii.RData") (I did it before because it take 15 minutes).

We now need to compute beta diversity indices for each permutated matrix, and then compute the average for the beta diversity values corresponding to all pairs of islands. You can do so using the code below:

beta.mean <- function(dat){ ##this function computes the average for Sorensen and Simpson beta diversity for a givent site-by-species matrix dat

beta.dat <- beta.pair(dat)

return(c(mean(beta.dat$beta.sor),mean(beta.dat$beta.sim)))

}

beta.rand.soc.none <- data.frame(matrix(unlist(lapply(dat.soc.pa.perm.none$perm,beta.mean)),99,2,byrow = TRUE)) ## this applies the beta.mean() function above to each permutated site-by-species matrix

names(beta.rand.soc.none) <- c("Sorensen","Simpson") ## rename the columns of the data frame

beta.rand.soc.row <- data.frame(matrix(unlist(lapply(dat.soc.pa.perm.row$perm,beta.mean)),99,2,byrow = TRUE))

names(beta.rand.soc.row) <- c("Sorensen","Simpson")

beta.rand.soc.col <- data.frame(matrix(unlist(lapply(dat.soc.pa.perm.col$perm,beta.mean)),99,2,byrow = TRUE))

names(beta.rand.soc.col) <- c("Sorensen","Simpson")

beta.rand.soc.both <- data.frame(matrix(unlist(lapply(dat.soc.pa.perm.both$perm,beta.mean)),99,2,byrow = TRUE))

names(beta.rand.soc.both) <- c("Sorensen","Simpson")

Take some time to understand how the lapply(), unlist(), matrix() and data.frame() functions work.

We can now plot the distributions under the different permutation algorithms, along with the observed beta diversity values for the Sorensen and Simpson dissimilarity index, and the 5% and 95% quantiles:

par(mfrow=c(2,4))

hist(beta.rand.haw.none$Sorensen,breaks=seq(0,1,0.025),main="None fixed",xlab="Sorensen")

abline(v=quantile(beta.rand.haw.none$Sorensen,0.025),col="blue")

abline(v=quantile(beta.rand.haw.none$Sorensen,0.0975),col="blue")

abline(v=beta.haw.obs[1],col="red")

hist(beta.rand.haw.row$Sorensen,breaks=seq(0,1,0.025),main="Rows fixed",xlab="Sorensen")

abline(v=quantile(beta.rand.haw.row$Sorensen,0.025),col="blue")

abline(v=quantile(beta.rand.haw.row$Sorensen,0.0975),col="blue")

abline(v=beta.haw.obs[1],col="red")

hist(beta.rand.haw.col$Sorensen,breaks=seq(0,1,0.025),main="Columns fixed",xlab="Sorensen")

abline(v=quantile(beta.rand.haw.col$Sorensen,0.025),col="blue")

abline(v=quantile(beta.rand.haw.col$Sorensen,0.0975),col="blue")

abline(v=beta.haw.obs[1],col="red")

hist(beta.rand.haw.both$Sorensen,breaks=seq(0,1,0.025),main="Both fixed",xlab="Sorensen")

abline(v=quantile(beta.rand.haw.both$Sorensen,0.025),col="blue")

abline(v=quantile(beta.rand.haw.both$Sorensen,0.0975),col="blue")

abline(v=beta.haw.obs[1],col="red")

hist(beta.rand.haw.none$Simpson,breaks=seq(0,1,0.025),main="",xlab="Simpson")

abline(v=quantile(beta.rand.haw.none$Simpson,0.025),col="blue")

abline(v=quantile(beta.rand.haw.none$Simpson,0.0975),col="blue")

abline(v=beta.haw.obs[2],col="red")

hist(beta.rand.haw.row$Simpson,breaks=seq(0,1,0.025),main="",xlab="Simpson")

abline(v=quantile(beta.rand.haw.row$Simpson,0.025),col="blue")

abline(v=quantile(beta.rand.haw.row$Simpson,0.0975),col="blue")

abline(v=beta.haw.obs[2],col="red")

hist(beta.rand.haw.col$Simpson,breaks=seq(0,1,0.025),main="",xlab="Simpson")

abline(v=quantile(beta.rand.haw.col$Simpson,0.025),col="blue")

abline(v=quantile(beta.rand.haw.col$Simpson,0.0975),col="blue")

abline(v=beta.haw.obs[2],col="red")

hist(beta.rand.haw.both$Simpson,breaks=seq(0,1,0.025),main="",xlab="Simpson")

abline(v=quantile(beta.rand.haw.both$Simpson,0.025),col="blue")

abline(v=quantile(beta.rand.haw.both$Simpson,0.0975),col="blue")

abline(v=beta.haw.obs[2],col="red")

Discuss the figures. Can you explain why the histograms are different? What can we say about species turnover in this system? Play with the function quantile(), and see if turnover would be significantly different than expected by chance for a one-tailed test.

Do the same for Hawai’i.

**Regression models on species richness:**

Here we will load data for the whole of French Polynesia (or at least the islands for which I could extract environmental data).

load("islands\_FP\_regression\_models\_practical.RData")

***Generalised Linear Models***

Compute alpha diversity for dat.FP.pa, and store it as a new column in islands.pred.FP. Complete the following code:

islands.pred.FP$alpha <- ...

We will apply a GLM to explain alpha diversity as a function of environmental factors in islands.pred.FP. Which factors can you use? Look at the correlation between them and discuss.

Select some environmental factors, and compute the GLMs. Since we have count data, we will use a Poisson family. Complete the following code:

mod.glm.FP <- glm(alpha~...,data=islands.pred.FP,family = poisson())

We can inspect the model outputs using the following code:

summary(mod.glm.FP)

We can also look at how the correlation between predictors affect the output using the variance inflation factor (vif). We won’t get into details here, but as a rule of thumb, you want the vif values to be below 10. Otherwise, you need to drop some predictors with high vif values. One usually does this one predictor at a time, and re-assess the vif.

vif(mod.glm.FP)

You can also check different combinations of predictors, and see if more complex models actually explain better the response, using the Akaike Information Criterion (AIC). The AIC is a relative value that can only be used to compare models calibrated on the same data, but for different combinations of predictors and response variables. In short, you only keep a more complex model if its AIC is lower than the AIC of a simpler model by a margin of 2.

Try some combinations of predictors. Do not just use random combinations, but build your models based on different hypotheses.

AIC(mod.glm.FP)

Finally, we can check how well the GLMs fit the data using the following code:

cor(islands.pred.FP$alpha,predict(mod.glm.FP,type="response"))

What do the model outputs tell you?

***Generalised Additive Models***

We will see another type of model, more complex: Generalised Additive Models (GAM). A GAM is a linear model with a key difference when compared to Generalised Linear Models: a GAM is allowed to learn non-linear features. GAMs relax the restriction that the relationship must be a simple weighted sum, and instead assume that the outcome can be modelled by a sum of arbitrary functions of each feature. To do this, we simply replace beta coefficients from Linear Regression with a flexible function which allows nonlinear relationships. Don’t worry about the maths, the idea is to illustrate some important concepts when fitting statistical models. Here, you will only be able to use one predictor, due to the number of islands and the complexity of such a model. For island area, the code is the following:

mod.gam.FP <- gam(alpha~s(elev\_max)+…,data=islands.pred.FP,family = poisson(),method = "REML")

the s() function allows to fit the non-linear relationships. These are called “splines”.

As before, let’s see how well the GAM fits the data:

cor(islands.pred.FP$alpha,predict(mod.gam.FP,type="response"))

What does it tell you?

Let’s plot the model outputs:

##model performance

cor(islands.pred.FP$alpha,predict(mod.glm.FP,type="response"))

cor(islands.pred.FP$alpha,predict(mod.gam.FP,type="response"))

cor(islands.pred.FP$alpha,predict(mod.gam.FP2,type="response"))

##splines

par(mfrow=c(1,2))

plot(mod.gam.FP)

points(islands.pred.FP$IslandArea,log(islands.pred.FP$alpha)-mod.gam.FP$coefficients[1])

What can you tell about the models?

Let’s smooth the splines, by reducing their complexity, and fit another set of models:

mod.gam.FP2 <- gam(alpha~s(elev\_max,k=4)+…,data=islands.pred.FP,family = poisson(),method = "REML")

cor(islands.pred.FP$alpha,predict(mod.gam.FP2,type="response"))

par(mfrow=c(2,4))

plot(mod.gam.FP,ylim=c(-20,20),residuals = T,pch=1)

plot(mod.gam.FP2,ylim=c(-20,20),residuals = T,pch=1)

What difference does it make?

***Transform the original data***

Look at the values for island areas, and how they are distributed? Can this cause some issue? We usually prefer predictor values to be distributed as uniformly as possible to avoid biases. A typical transformation to do is the log transformation. Let’s log-transform elevation and island area and redo the same analyses. How would you interpret the results?

##GLMs

mod.glm.FP <- glm(alpha~log(elev\_max)+log(IslandArea)+…,data=islands.pred.FP,family = poisson())

##GAMs

mod.gam.FP <- gam(alpha~s(log(elev\_max))+s(log(IslandArea))+…,data=islands.pred.FP,family = poisson(),method = "REML")

##Smoother GAMs

mod.gam.FP2 <- gam(alpha~s(log(elev\_max),k=4)+s(log(IslandArea),k=4)+…,data=islands.pred.FP,family = poisson(),method = "REML")

summary(mod.glm.FP)

summary(mod.gam.FP)

summary(mod.gam.FP2)

##model performance

cor(islands.pred.FP$alpha,predict(mod.glm.FP,type="response"))

cor(islands.pred.FP$alpha,predict(mod.gam.FP,type="response"))

cor(islands.pred.FP$alpha,predict(mod.gam.FP2,type="response"))

par(mfrow=c(1,3))

plot(islands.pred.FP$alpha,predict(mod.glm.FP,type="response"),type="p")

plot(islands.pred.FP$alpha,predict(mod.gam.FP,type="response"),type="p")

plot(islands.pred.FP$alpha,predict(mod.gam.FP2,type="response"),type="p")

##Plot GAM splines

par(mfrow=c(2,4))

plot(mod.gam.FP,ylim=c(-20,20),residuals = T,pch=1)

plot(mod.gam.FP2,ylim=c(-20,20),residuals = T,pch=1)

**Regression models on species turnover:**

Here we will examine if environmental factors can explain species turnover, computed using Sorensen and Simpson dissimilarity indices. We will use Generalised Dissimilarity Models (GDMs), discussed in class. Based on the outputs of the null models and the richness analyses, can you make some predictions about the results you will get?

Let’s compute a GDM for beta diversity. We will use the zetadiv package, using zeta diversity for order 2, which corresponds to beta diversity (see Bonus section below about why we use zetadiv).

Here, since we use all combinations of islands, we will have more data points than the number of islands, so we will use all the predictors (although in practice it would be advisable to do more variable selection). The number of possible combinations of 2 out of 14 islands (for the Society archipelago) is given by:

choose(14,2)

and you can see these combinations by typing:

combn(14,2)

First, we need to load the island coordinates, to incorporate distance in the models:

load("island\_coordinates\_FP.RData")

Let’s compute the GDMs for Sorensen and Simpson, for French Polynesia:

##GDMs

GDM.FP.Sor <- Zeta.msgdm(data.spec = dat.FP.pa, data.env = islands.pred.FP[2:7], xy = FP.coords[2:3],order=2,reg.type = "ispline",method.glm = "glm.fit.cons",cons = -1, cons.inter = -1,distance.type="ortho",normalize="Sorensen",family = binomial("log"))

GDM.FP.Sim <- Zeta.msgdm(data.spec = dat.FP.pa, data.env = islands.pred.FP[2:7], xy = FP.coords[2:3],order=2,reg.type = "ispline",method.glm = "glm.fit.cons",cons = -1, cons.inter = -1,distance.type="ortho",normalize="Simpson",family = binomial("log"))

# GDM.FP.Sor <- Zeta.msgdm(data.spec = dat.FP.pa, data.env = islands.pred.FP[2:7],order=2,reg.type = "ispline",method.glm = "glm.fit.cons",cons = -1, cons.inter = -1,distance.type="ortho",normalize="Sorensen",family = binomial("log"))

# GDM.FP.Sim <- Zeta.msgdm(data.spec = dat.FP.pa, data.env = islands.pred.FP[2:7],order=2,reg.type = "ispline",method.glm = "glm.fit.cons",cons = -1, cons.inter = -1,distance.type="ortho",normalize="Simpson",family = binomial("log"))

Let’s see how well the predicted beta diversity values fit the observed values:

##Model performance

Dat.FP.Sor.pred <- Predict.msgdm(model.msgdm = GDM.FP.Sor$model, reg.type = "ispline", newdata = GDM.FP.Sor$predictors)

cor(GDM.FP.Sor$val,Dat.FP.Sor.pred)^2

Dat.FP.Sim.pred <- Predict.msgdm(model.msgdm = GDM.FP.Sor$model, reg.type = "ispline", newdata = GDM.FP.Sor$predictors)

cor(GDM.FP.Sim$val,Dat.FP.Sim.pred)^2

Let’s plot the predicted vs observed values, and the I-splines generated by the GDMs. The left column is for Sorensen, and the right column is for Simpson:

##Plot outputs

par(mfrow=c(2,2))

plot(GDM.FP.Sor$val,Dat.FP.Sor.pred,pch=20,col="blue")

lines(sort(GDM.FP.Sor$val),predict(lm(Dat.FP.Sor.pred~GDM.FP.Sor$val))[order(GDM.FP.Sor$val)])

plot(GDM.FP.Sim$val,Dat.FP.Sim.pred,pch=20,col="blue")

lines(sort(GDM.FP.Sim$val),predict(lm(Dat.FP.Sim.pred~GDM.FP.Sim$val))[order(GDM.FP.Sim$val)])

Plot.ispline(msgdm=GDM.FP.Sor,data.env = islands.pred.FP[2:7],distance=TRUE)

Plot.ispline(msgdm=GDM.FP.Sim,data.env = islands.pred.FP[2:7],distance=TRUE)

Do the outputs correspond to the predictions you made at the beginning of this section?

Do the same after log-transforming elevation and area. How does it change the splines?

islands.pred.FP2 <- islands.pred.FP

islands.pred.FP2$elev\_max <- log(islands.pred.FP2$elev\_max)

islands.pred.FP2$IslandArea <- log(islands.pred.FP2$IslandArea)

GDM.FP.Sor2 <- Zeta.msgdm(data.spec = dat.FP.pa, data.env = islands.pred.FP2[2:7], xy = FP.coords[2:3],order=2,reg.type = "ispline",method.glm = "glm.fit.cons",cons = -1, cons.inter = -1,distance.type="ortho",normalize="Sorensen",family = binomial("log"))

GDM.FP.Sim2 <- Zeta.msgdm(data.spec = dat.FP.pa, data.env = islands.pred.FP2[2:7], xy = FP.coords[2:3],order=2,reg.type = "ispline",method.glm = "glm.fit.cons",cons = -1, cons.inter = 91,distance.type="ortho",normalize="Simpson",family = binomial("log"))

Dat.FP.Sor.pred2 <- Predict.msgdm(model.msgdm = GDM.FP.Sor2$model, reg.type = "ispline", newdata = GDM.FP.Sor2$predictors)

cor(GDM.FP.Sor2$val,Dat.FP.Sor.pred2)^2

Dat.FP.Sim.pred2 <- Predict.msgdm(model.msgdm = GDM.FP.Sor2$model, reg.type = "ispline", newdata = GDM.FP.Sor2$predictors)

cor(GDM.FP.Sim2$val,Dat.FP.Sim.pred2)^2

par(mfrow=c(2,2))

plot(GDM.FP.Sor2$val,Dat.FP.Sor.pred2,pch=20,col="blue")

lines(sort(GDM.FP.Sor2$val),predict(lm(Dat.FP.Sor.pred2~GDM.FP.Sor2$val))[order(GDM.FP.Sor2$val)])

plot(GDM.FP.Sim2$val,Dat.FP.Sim.pred2,pch=20,col="blue")

lines(sort(GDM.FP.Sim2$val),predict(lm(Dat.FP.Sim.pred2~GDM.FP.Sim2$val))[order(GDM.FP.Sim2$val)])

Plot.ispline(msgdm=GDM.FP.Sor2,data.env = islands.pred.FP2[2:7],distance=TRUE)

Plot.ispline(msgdm=GDM.FP.Sim2,data.env = islands.pred.FP2[2:7],distance=TRUE)

**Bonus:**

You can also use the gdm package to compute the GDM. It is much quicker than zetadiv for large datasets, but it is a bit more tricky to format the data, and it does not allow to use Simpson diversity (which, as we saw, is important). Here is the code to do so:

dat.FP.pa2 <- cbind(row.names(dat.FP.pa),dat.FP.pa)

names(dat.FP.pa2)[1] <- "island"

dat.FP.pa.long <- dat.FP.pa2 %>%

pivot\_longer(cols=!island,

names\_to = "Species",

values\_to = "Presence")

dat.FP.pa.long <- dat.FP.pa.long[-which(dat.FP.pa.long$Presence==0),1:2]

dat.FP.pa.long$IslandArea <- numeric(nrow(dat.FP.pa.long))

dat.FP.pa.long$elev\_max <- numeric(nrow(dat.FP.pa.long))

dat.FP.pa.long$temp.mean <- numeric(nrow(dat.FP.pa.long))

dat.FP.pa.long$temp.seas <- numeric(nrow(dat.FP.pa.long))

dat.FP.pa.long$precan <- numeric(nrow(dat.FP.pa.long))

dat.FP.pa.long$prec.seas <- numeric(nrow(dat.FP.pa.long))

dat.FP.pa.long$long <- numeric(nrow(dat.FP.pa.long))

dat.FP.pa.long$lat <- numeric(nrow(dat.FP.pa.long))

for(i in row.names(dat.FP.pa)){

toto <- islands.pred.FP[which(islands.pred.FP$Name\_USGSO==i),2:7]

dat.FP.pa.long[which(dat.FP.pa.long$island==i),3:8] <- toto[1:6]

toto <- FP.coords[which(FP.coords$island==i),2:3]

dat.FP.pa.long[which(dat.FP.pa.long$island==i),9:10] <- toto

}

dat.FP.pa.long <- as.data.frame(dat.FP.pa.long)

head(dat.FP.pa.long)

sppData.FP <- dat.FP.pa.long[c(1,2,9,10)]

envTab.FP <- dat.FP.pa.long[c(1,3:10)]

#envTab.FP <- dat.FP.pa.long[c(1,4)]

sitePairTab.FP <- formatsitepair(sppData.FP,2,XColumn="long",YColumn="lat",sppColumn="Species", siteColumn="island",predData=envTab.FP)

gdmTabMod.FP<-gdm(sitePairTab.FP,geo=TRUE)

summary(gdmTabMod.FP)

plot(gdmTabMod.FP)