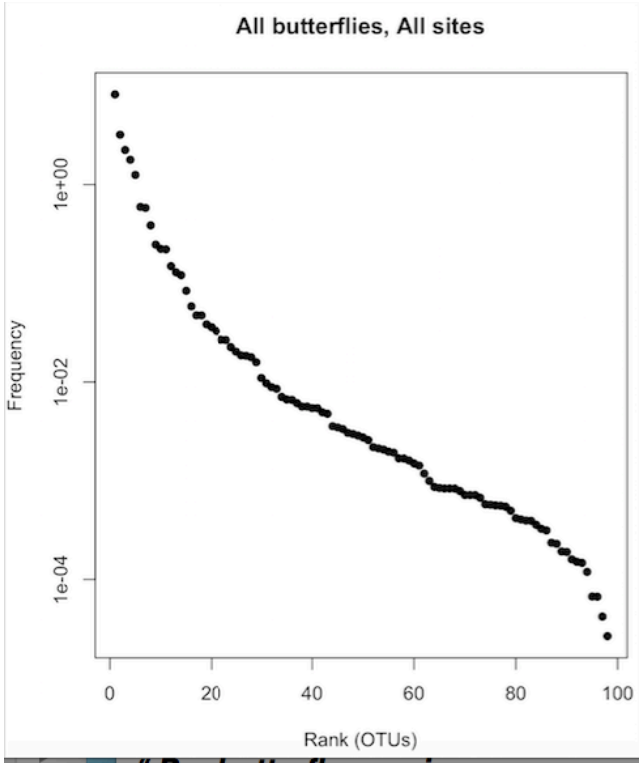
```
library(RADanalysis)
library(BiodiversityR)
library(codyn) # Explore this one…

# Function to normalize data
normalize <- function(x) {
return ((x - min(x)) / (max(x) - min(x)))
}

data=read_xlsx("/Users/Daniel/Desktop/BES project 2014/BES analysis 2020/
data1.xlsx") # prueba con Brown argus
data=as.data.frame(data)
data_original=data
data=data[,-c(1,2,3,4)]
datasites=data[1]
data[1]=datasites
#data = setNames(data.frame(t(data[,-1])), data[,1]) # Transpose a dataframe and
keep first column as column names
data_old = data %>% filter(str_detect(site, "old")) # filters old sites
data_new = data %>% filter(str_detect(site, "new")) # filters new sites
olddata = setNames(data.frame(t(data_old[,-1])), data_old[,1]) # Transpose a
dataframe and keep first column as column names
newdata = setNames(data.frame(t(data_new[,-1])), data_new[,1])
nrow(olddata)-length(which(rowSums(olddata)==0)) # Number of OTUs in old range:
55 - 90
(nrow(olddata)-length(which(rowSums(olddata)==0)))/nrow(olddata) # Same in
percentage
nrow(newdata)-length(which(rowSums(newdata)==0)) # Number of OTUs in new
range: 53 - 82
(nrow(newdata)-length(which(rowSums(newdata)==0)))/nrow(newdata) # Same in
percentage

q=colSums(data[,-1])
plot(rev(sort(q)),log="y",pch=16,xlab="Rank (OTUs)",ylab="Frequency",main="All
butterflies, All sites")
```



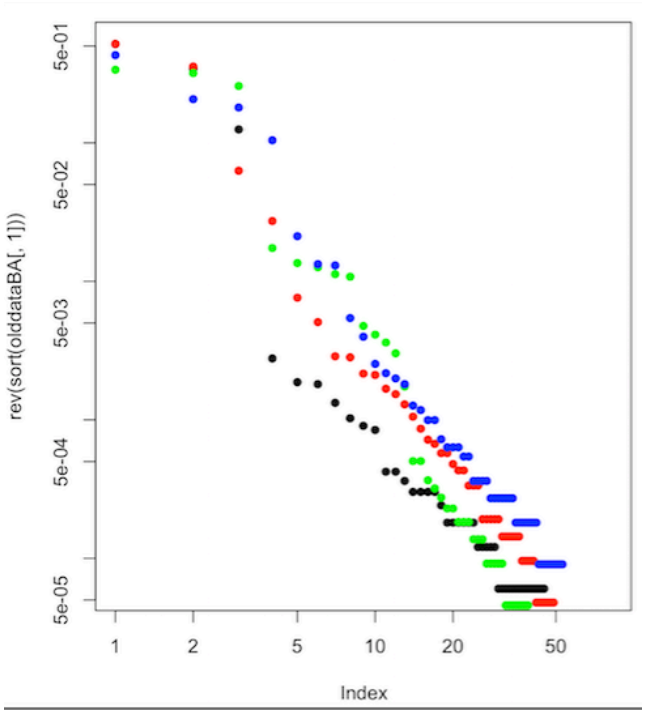
 # Per butterfly species

dataBA=filter(data_original,butterfly=="BrownArgus")
dataBA=dataBA[,-c(1,2,3,4)]
datasites=dataBA[1]
dataBA[1]=datasites
dataBA=dataBA[,-(which(colSums(dataBA[,-1])==0)+1)] # We consider only OTUs reported in the butterfly's range (old+new), not OTUs from other butterflies
dataBA_old = dataBA %>% filter(str_detect(site, "old")) # filters old sites
dataBA_new = dataBA %>% filter(str_detect(site, "new")) # filters new sites
olddataBA = setNames(data.frame(t(dataBA_old[,-1])), dataBA_old[,1]) # Transpose a dataframe and keep first column as column names
newdataBA = setNames(data.frame(t(dataBA_new[,-1])), dataBA_new[,1])
#nrow(olddataBA)-length(which(rowSums(olddataBA)==0)) # Number of OTUs in old range: 47 - 73
#(nrow(olddataBA)-length(which(rowSums(olddataBA)==0)))/nrow(olddataBA) # Same in percentage
#nrow(newdataBA)-length(which(rowSums(newdataBA)==0)) # Number of OTUs in new range: 44 - 72
#(nrow(newdataBA)-length(which(rowSums(newdataBA)==0)))/nrow(newdataBA) # Same in percentage
dataSW=filter(data_original,butterfly=="SpeckledWood")
dataSW=dataSW[,-c(1,2,3,4)]
datasites=dataSW[1]
dataSW[1]=datasites
dataSW=dataSW[,-(which(colSums(dataSW[,-1])==0)+1)] # We consider only OTUs reported in the butterfly's range (old+new), not OTUs from other butterflies
dataSW_old = dataSW %>% filter(str_detect(site, "old")) # filters old sites
dataSW_new = dataSW %>% filter(str_detect(site, "new")) # filters new sites
olddataSW = setNames(data.frame(t(dataSW_old[,-1])), dataSW_old[,1]) #

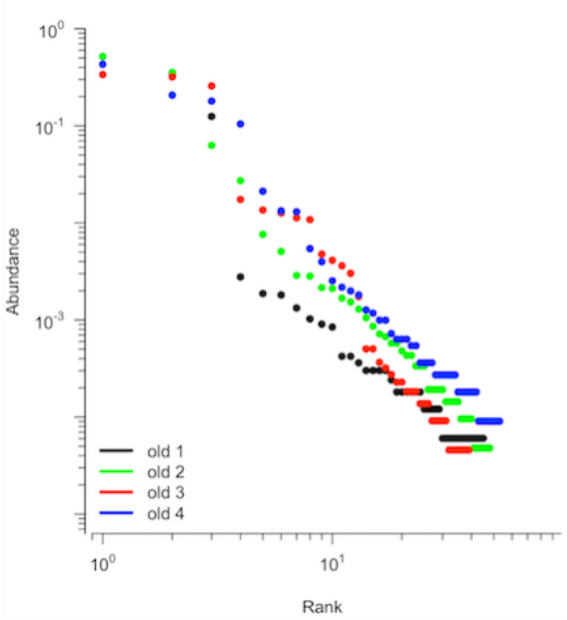
```
Transpose a dataframe and keep first column as column names
newdataSW = setNames(data.frame(t(dataSW_new[,-1])), dataSW_new[,1])
#nrow(olddataSW)-length(which(rowSums(olddataSW)==0)) # Number of OTUs in
old range: 54
#(nrow(olddataSW)-length(which(rowSums(olddataSW)==0)))/nrow(olddataSW) #
Same in percentage
#nrow(newdataSW)-length(which(rowSums(newdataSW)==0)) # Number of OTUs in
new range: 53
#(nrow(newdataSW)-length(which(rowSums(newdataSW)==0)))/nrow(newdataSW)
# Same in percentage
dataSSS=filter(data_original,butterfly=="SSSkipper")
dataSSS=dataSSS[,-c(1,2,3,4)]
datasites=dataSSS[1]
dataSSS[1]=datasites
dataSSS=dataSSS[,-(which(colSums(dataSSS[,-1])==0)+1)] # We consider only
OTUs reported in the butterfly's range (old+new), not OTUs from other butterflies
dataSSS_old = dataSSS %>% filter(str_detect(site, "old")) # filters old sites
dataSSS_new = dataSSS %>% filter(str_detect(site, "new")) # filters new sites
olddataSSS = setNames(data.frame(t(dataSSS_old[,-1])), dataSSS_old[,1]) #
Transpose a dataframe and keep first column as column names
newdataSSS = setNames(data.frame(t(dataSSS_new[,-1])), dataSSS_new[,1])
#nrow(olddataSSS)-length(which(rowSums(olddataSSS)==0)) # Number of OTUs in
old range: 52
#(nrow(olddataSSS)-length(which(rowSums(olddataSSS)==0)))/nrow(olddataSSS) #
Same in percentage
#nrow(newdataSSS)-length(which(rowSums(newdataSSS)==0)) # Number of OTUs
in new range: 29
#(nrow(newdataSSS)-length(which(rowSums(newdataSSS)==0)))/
nrow(newdataSSS) # Same in percentage

  ######

#par(mfrow=c(2,2)) # Plotting 4 old x new sites
plot(rev(sort(olddataBA[,1])),log="y",pch=16,xlab="Rank",ylab="Abundance")
points(rev(sort(olddataBA[,2])),log="y",pch=16,col="red")
points(rev(sort(olddataBA[,3])),log="y",pch=16,col="green")
points(rev(sort(olddataBA[,4])),log="y",pch=16,col="blue")
points(rev(sort(newdataBA[,1])),log="y",pch=4)
points(rev(sort(newdataBA[,2])),log="y",pch=4,col="red")
points(rev(sort(newdataBA[,4])),log="y",pch=4,col="green")
points(rev(sort(newdataBA[,5])),log="y",pch=4,col="blue")
```




```
# https://cran.r-project.org/web/packages/RADanalysis/vignettes/RADanalysis_intro.html
rads <- olddataBA
#plot original rads
line_cols <- c("black","green","red","blue")
#to specify different stages of subjects
sample_classes <- c(1,2,3,4)
plot(1,col = "white", xlim=c(1,60), ylim=c(0.00001,1),log  = "y",axes = F,xlab =
"Rank",ylab = "Abundance",main = "")
sfsmisc::eaxis(side = 1,at = c(1,10,20,30,50,100))
sfsmisc::eaxis(side = 2,at = c(0.00001,0.001,0.01,0.1,1))
for(i in 1:ncol(rads)){
    temp <- sort(rads[-1,i],decreasing = TRUE)
    temp <- temp[temp>0]
    points(x=temp,lwd = 2,col = line_cols[sample_classes[i]], pch=16)
}
legend("topright",bty = "n",legend = c("old 1","old 2","old 3","old 4"),col =
line_cols,lwd = 3)
```



# Different fits
mod <- radfit(olddataBA[,1])
mod
plot(mod)

#Zipf fits: the smaller the absolute value of the parameter, the more even the distribution of abundances in the community. **In gral, parameters are higher in expanded margins —> more uneven distributions in expanded ranges. This info must be taken together with the next section on RADs**

Brown argus
old 1: 0.67
old 2: 0.66
old 3: 0.54
old 4: 0.55
mean old = 0.605
new 1: 0.75
new 2: 0.58
new 3: 0.54
new 4: 0.56
new 5: 0.51
mean new: 0.6

Speckled wood
old 5: 0.62
old 6: 0.62
old 7: 0.58
mean old: 0.61
new 6: 0.50
new 7: 0.81
new 8: 0.70
mean new: 0.67

Silver-spotted skipper
old 8: 0.56
old 9: 0.47
old 10: 0.48
mean old: 0.50
new 9: 0.59
new 10: 0.62
mean new: 0.605

######################

BROWN ARGUS

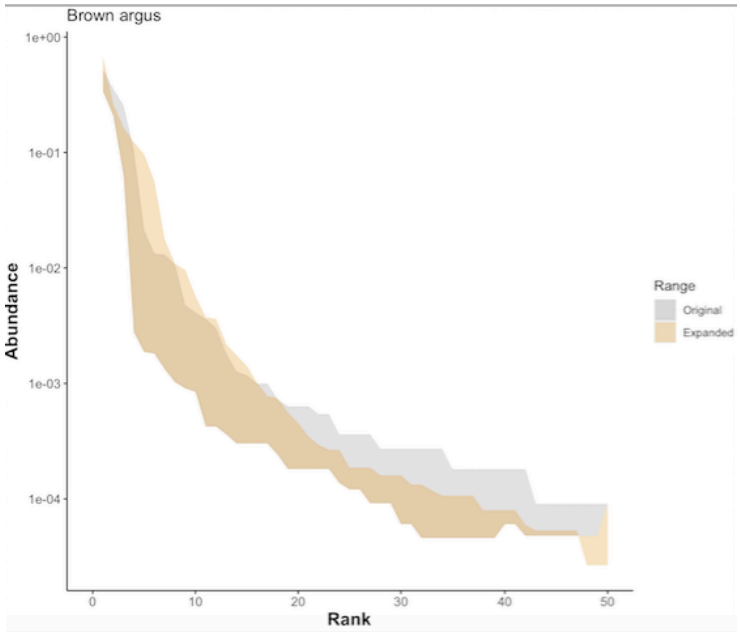
```
# We sort the datasets in decreasing rank-abundance order:
olddataBA2=matrix(nrow=nrow(olddataBA),ncol=ncol(olddataBA))
colnames(olddataBA2)=c("old1","old2","old3","old4")
newdataBA2=matrix(nrow=nrow(newdataBA),ncol=ncol(newdataBA))
colnames(newdataBA2)=c("new1","new2","new3","new4","new5")
for(i in 1:ncol(olddataBA)) {
        olddataBA2[,i] <- sort(olddataBA[,i],decreasing = TRUE)
        }
for(i in 1:ncol(newdataBA)) {
        newdataBA2[,i] <- sort(newdataBA[,i],decreasing = TRUE)
        }
newdataBA2=newdataBA2[,-c(3)]# Remove BA new site #3

# We calculate min and max values to have the range for the ribbon plots:
# olddataBA and newdataBA have the same number of rows
oldBA=matrix(nrow=nrow(olddataBA2),ncol=4)
colnames(oldBA)=c("OTU","min","max","range")
newBA=matrix (nrow=nrow(newdataBA2),ncol=4)
colnames(newBA)=c("OTU","min","max","range")

for(i in 1:nrow(olddataBA2))
        {
        oldmax=max(olddataBA2[i,])
        x=olddataBA2[i,]
        oldmin=min(x[which(x>0)])
        newmax=max(newdataBA2[i,])
        y=newdataBA2[i,]
        newmin=min(x[which(x>0)])
        oldBA[i,1]=i
        oldBA[i,2]=oldmin
        oldBA[i,3]=oldmax
        oldBA[i,4]=1 # OLD RANGE
        newBA[i,1]=i
        newBA[i,2]=newmin
        newBA[i,3]=newmax
        newBA[i,4]=2 # NEW RANGE
```

```
        }
BA=rbind(oldBA,newBA)
BA=as.data.frame(BA)
BA[sapply(BA, is.infinite)] <- NA

ggplot(BA) + geom_ribbon(aes(x=OTU, ymin=min, ymax=max, fill=factor(range)),
alpha=0.3) +
theme(axis.text=element_text(size=10),axis.title=element_text(size=14,face="bold"))
+ theme(panel.grid.major = element_blank(), panel.grid.minor =
element_blank(),panel.background = element_blank(), axis.line =
element_line(colour = "black")) + labs(x="Rank",y="Abundance",title="Brown argus")
+ labs(color = "OTU") + scale_fill_manual(values=c("#999999", "#E69F00"),
name="Range", breaks=c(1,2), labels=c("Original","Expanded")) + scale_y_log10() +
xlim(0,50)
```



SPECKLED WOOD

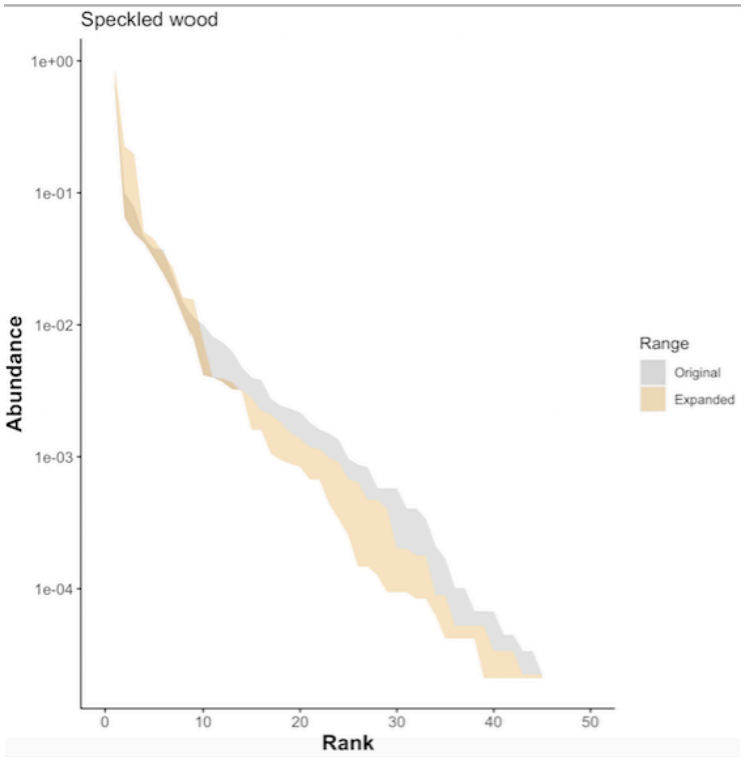
```
# We sort the datasets in decreasing rank-abundance order:
olddataSW2=matrix(nrow=nrow(olddataSW),ncol=ncol(olddataSW))
colnames(olddataSW2)=c("old5","old6","old7")
newdataSW2=matrix(nrow=nrow(newdataSW),ncol=ncol(newdataSW))
colnames(newdataSW2)=c("new6","new7","new8")
for(i in 1:ncol(olddataSW)) {
        olddataSW2[,i] <- sort(olddataSW[,i],decreasing = TRUE)
        }
for(i in 1:ncol(newdataSW)) {
        newdataSW2[,i] <- sort(newdataSW[,i],decreasing = TRUE)
        }

# We calculate min and max values to have the range for the ribbon plots:
# olddataSW and newdataSW have the same number of rows
```

```
oldSW=matrix(nrow=nrow(olddataSW2),ncol=4)
colnames(oldSW)=c("OTU","min","max","range")
newSW=matrix (nrow=nrow(newdataSW2),ncol=4)
colnames(newSW)=c("OTU","min","max","range")

for(i in 1:nrow(olddataSW2))
	{
	oldmax=max(olddataSW2[i,])
	x=olddataSW2[i,]
	oldmin=min(x[which(x>0)])
	newmax=max(newdataSW2[i,])
	y=newdataSW2[i,]
	newmin=min(x[which(x>0)])
	oldSW[i,1]=i
	oldSW[i,2]=oldmin
	oldSW[i,3]=oldmax
	oldSW[i,4]=1 # OLD RANGE
	newSW[i,1]=i
	newSW[i,2]=newmin
	newSW[i,3]=newmax
	newSW[i,4]=2 # NEW RANGE
	}
SW=rbind(oldSW,newSW)
SW=as.data.frame(SW)
SW[sapply(SW, is.infinite)] <- NA

ggplot(SW) + geom_ribbon(aes(x=OTU, ymin=min, ymax=max, fill=factor(range)),
alpha=0.3) +
theme(axis.text=element_text(size=10),axis.title=element_text(size=14,face="bold"))
+ theme(panel.grid.major = element_blank(), panel.grid.minor =
element_blank(),panel.background = element_blank(), axis.line =
element_line(colour = "black")) + labs(x="Rank",y="Abundance",title="Speckled
wood") + labs(color = "OTU") + scale_fill_manual(values=c("#999999", "#E69F00"),
name="Range", breaks=c(1,2), labels=c("Original","Expanded")) + scale_y_log10() +
xlim(0,50)
```



SILVER-SPOTTED SKIPPER

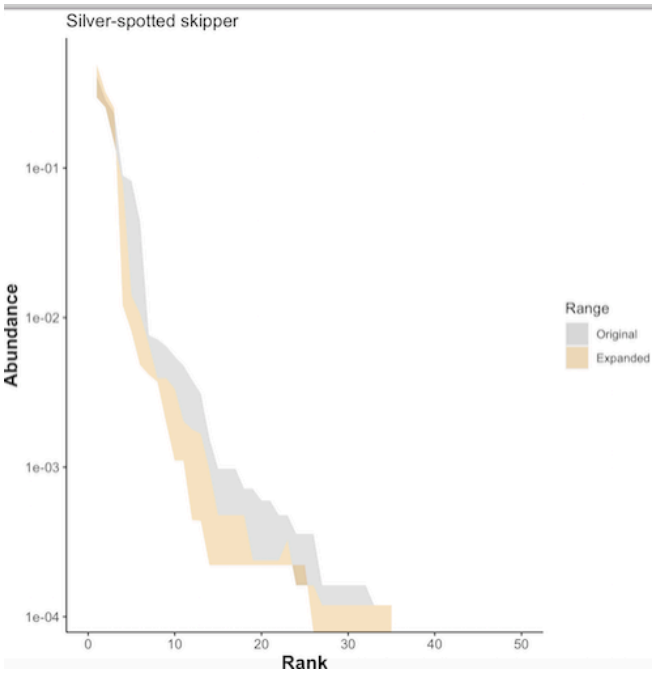
```
# We sort the datasets in decreasing rank-abundance order:
olddataSSS2=matrix(nrow=nrow(olddataSSS),ncol=ncol(olddataSSS))
colnames(olddataSSS2)=c("old8","old9","old10")
newdataSSS2=matrix(nrow=nrow(newdataSSS),ncol=ncol(newdataSSS))
colnames(newdataSSS2)=c("new9","new10")
for(i in 1:ncol(olddataSSS)) {
        olddataSSS2[,i] <- sort(olddataSSS[,i],decreasing = TRUE)
        }
for(i in 1:ncol(newdataSSS)) {
        newdataSSS2[,i] <- sort(newdataSSS[,i],decreasing = TRUE)
        }

# We calculate min and max values to have the range for the ribbon plots:
# olddataSSS and newdataSSS have the same number of rows
oldSSS=matrix(nrow=nrow(olddataSSS2),ncol=4)
colnames(oldSSS)=c("OTU","min","max","range")
newSSS=matrix (nrow=nrow(newdataSSS2),ncol=4)
colnames(newSSS)=c("OTU","min","max","range")

for(i in 1:nrow(olddataSSS2))
        {
        oldmax=max(olddataSSS2[i,])
        x=olddataSSS2[i,]
        oldmin=min(x[which(x>0)])
        newmax=max(newdataSSS2[i,])
        y=newdataSSS2[i,]
        newmin=min(x[which(x>0)])
```

```
        oldSSS[i,1]=i
        oldSSS[i,2]=oldmin
        oldSSS[i,3]=oldmax
        oldSSS[i,4]=1 # OLD RANGE
        newSSS[i,1]=i
        newSSS[i,2]=newmin
        newSSS[i,3]=newmax
        newSSS[i,4]=2 # NEW RANGE
        }
SSS=rbind(oldSSS,newSSS)
SSS=as.data.frame(SSS)
SSS[sapply(SSS, is.infinite)] <- NA

ggplot(SSS) + geom_ribbon(aes(x=OTU, ymin=min, ymax=max, fill=factor(range)),
alpha=0.3) +
theme(axis.text=element_text(size=10),axis.title=element_text(size=14,face="bold"))
+ theme(panel.grid.major = element_blank(), panel.grid.minor =
element_blank(),panel.background = element_blank(), axis.line =
element_line(colour = "black")) + labs(x="Rank",y="Abundance",title="Silver-spotted
skipper") + labs(color = "OTU") + scale_fill_manual(values=c("#999999",
"#E69F00"), name="Range", breaks=c(1,2), labels=c("Original","Expanded")) +
scale_y_log10() + xlim(0,50)
```

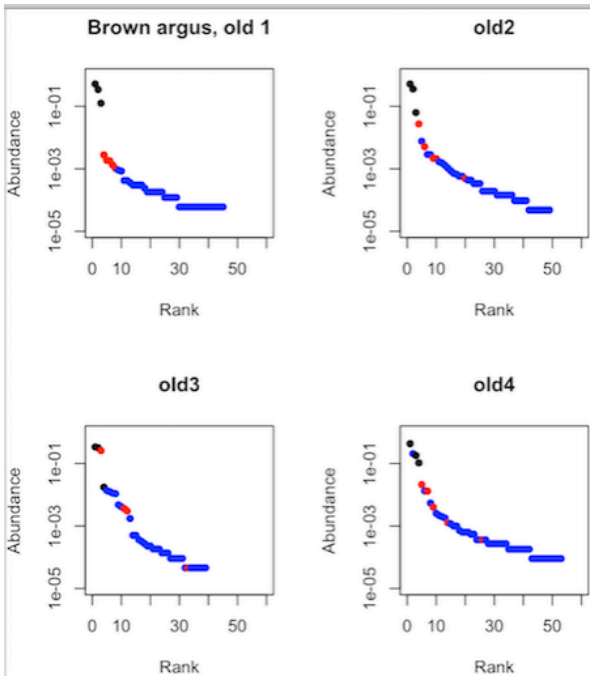


 # CHECK WHAT OTUs ARE MOST ABUNDANT: most species in new range are already in the old range —> Abundant species in old range tend to also be abundant in the new range

Using colours for high, intermediate, low OTU abundances

```
# Brown argus old #1
q1=as.data.frame(cbind(gsub("OTU", "",
rownames(olddataBA)),olddataBA[,1],rep(0,nrow(olddataBA))))
colnames(q1)=c("OTU","old1","cat")
q1$OTU=as.numeric(as.character(q1$OTU))
q1$old1=as.numeric(as.character(q1$old1))
q1$cat=as.numeric(as.character(q1$cat))
q1=q1 %>% arrange(desc(old1))
for (i in c(1:nrow(q1))) {
        if (q1[i,2]>0.01) {
                q1[i,3]="black"
                }
        if (q1[i,2]<=0.01 && q1[i,2]>0.001) {
                q1[i,3]="blue"
                }
        if (q1[i,2]<0.001) {
                q1[i,3]="red"
                }
        }
# Brown argus old #2
q2=as.data.frame(cbind(gsub("OTU", "",
rownames(olddataBA)),olddataBA[,2],rep(0,nrow(olddataBA))))
colnames(q2)=c("OTU","old2","cat")
q2$OTU=as.numeric(as.character(q2$OTU))
q2$old2=as.numeric(as.character(q2$old2))
q2$cat=as.numeric(as.character(q2$cat))
q2$cat=(q1 %>% arrange(OTU))[3]$cat
q2=q2 %>% arrange(desc(old2))
# Brown argus old #3
q3=as.data.frame(cbind(gsub("OTU", "",
rownames(olddataBA)),olddataBA[,3],rep(0,nrow(olddataBA))))
colnames(q3)=c("OTU","old3","cat")
q3$OTU=as.numeric(as.character(q3$OTU))
q3$old3=as.numeric(as.character(q3$old3))
q3$cat=as.numeric(as.character(q3$cat))
q3$cat=(q1 %>% arrange(OTU))[3]$cat
q3=q3 %>% arrange(desc(old3))
# Brown argus old #4
q4=as.data.frame(cbind(gsub("OTU", "",
rownames(olddataBA)),olddataBA[,4],rep(0,nrow(olddataBA))))
colnames(q4)=c("OTU","old4","cat")
q4$OTU=as.numeric(as.character(q4$OTU))
q4$old4=as.numeric(as.character(q4$old4))
q4$cat=as.numeric(as.character(q4$cat))
q4$cat=(q1 %>% arrange(OTU))[3]$cat
q4=q4 %>% arrange(desc(old4))
# Plotting
par(mfrow=c(2,2))
plot(rev(sort(q1[,2])),log="y",xlim=c(0,60),ylim=c(0.00001,1),col=q1$cat,pch=16,xlab
="Rank",ylab ="Abundance",main="Brown argus, old 1")
```

```
plot(rev(sort(q2[,2])),log="y",xlim=c(0,60),ylim=c(0.00001,1),col=q2$cat,pch=16,xlab
="Rank",ylab ="Abundance",main="old2")
plot(rev(sort(q3[,2])),log="y",xlim=c(0,60),ylim=c(0.00001,1),col=q3$cat,pch=16,xlab
="Rank",ylab ="Abundance",main="old3")
plot(rev(sort(q4[,2])),log="y",xlim=c(0,60),ylim=c(0.00001,1),col=q4$cat,pch=16,xlab
="Rank",ylab ="Abundance",main="old4")
```



RUN THE FOLLOWING CODE TO CHANGE ABUNDANCE CRITERIA IN THE ANALYSIS OF CORE COMMUNITITES!!!!!! (ALSO THE PREVIOSI LINES MUST BE RUN)

 # Same as above, but pooling ranges (old sites together, new sites together) and comparing old Vs new

 # Brown argus OLD SITES

```
olddataBAall=as.data.frame(rowSums(olddataBA))
newdataBAall=as.data.frame(rowSums(newdataBA))
olddataBAall=as.data.frame(cbind(gsub("OTU", "",
rownames(olddataBAall)),olddataBAall[,1],rep(0,nrow(olddataBAall))))
colnames(olddataBAall)=c("OTU","old","cat")
olddataBAall$OTU=as.numeric(as.character(olddataBAall$OTU))
olddataBAall$old=as.numeric(as.character(olddataBAall$old))
olddataBAall$old=normalize(olddataBAall$old) # We normalize data to constraint
values to {0,1}
olddataBAall$cat=as.numeric(as.character(olddataBAall$cat))
olddataBAall=olddataBAall %>% arrange(desc(old))
for (i in c(1:nrow(olddataBAall))) {
        if (olddataBAall[i,2]>0.01) {
                olddataBAall[i,3]="black"
                }
```

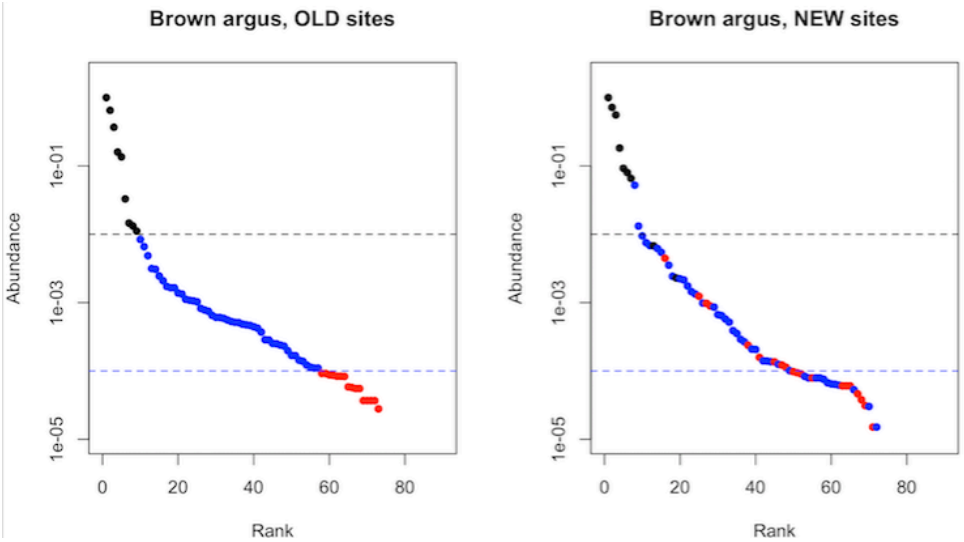
```
        if (olddataBAall[i,2]<=0.01 && olddataBAall[i,2]>0.0001) {
                olddataBAall[i,3]="blue"
                }
        if (olddataBAall[i,2]<=0.0001) {
                olddataBAall[i,3]="red"
                }
        }
```

 # Brown argus NEW SITES

```
newdataBAall=as.data.frame(rowSums(newdataBA))
newdataBAall=as.data.frame(rowSums(newdataBA))
newdataBAall=as.data.frame(cbind(gsub("OTU", "",
rownames(newdataBAall)),newdataBAall[,1],rep(0,nrow(newdataBAall))))
colnames(newdataBAall)=c("OTU","new","cat")
newdataBAall$OTU=as.numeric(as.character(newdataBAall$OTU))
newdataBAall$new=as.numeric(as.character(newdataBAall$new))
newdataBAall$new=normalize(newdataBAall$new) # We normalize data to
constraint values to {0,1}
newdataBAall$cat=as.numeric(as.character(newdataBAall$cat))
newdataBAall$cat=(olddataBAall %>% arrange(OTU))[3]$cat
newdataBAall=newdataBAall %>% arrange(desc(new))
```

 # Plotting

```
par(mfrow=c(1,2))
plot(rev(sort(olddataBAall[,2])),log="y",xlim=c(0,90),ylim=c(0.00001,2),col=olddataBA
all$cat,pch=16,xlab="Rank",ylab ="Abundance",main="Brown argus \nOLD sites")
abline(h=0.01,col="black",lty=2)
abline(h=0.0001,col="blue",lty=2)
text(55,0.3,"High",cex=0.8)
text(65,0.001,"Intermediate",cex=0.8)
text(15,0.000025,"Low",cex=0.8)
plot(rev(sort(newdataBAall[,2])),log="y",xlim=c(0,90),ylim=c(0.00001,2),col=newdata
BAall$cat,pch=16,xlab="Rank",ylab ="Abundance",main="Brown argus \nNEW
sites")
abline(h=0.01,col="black",lty=2)
abline(h=0.0001,col="blue",lty=2)
text(55,0.3,"High",cex=0.8)
text(65,0.001,"Intermediate",cex=0.8)
text(15,0.000025,"Low",cex=0.8)
```



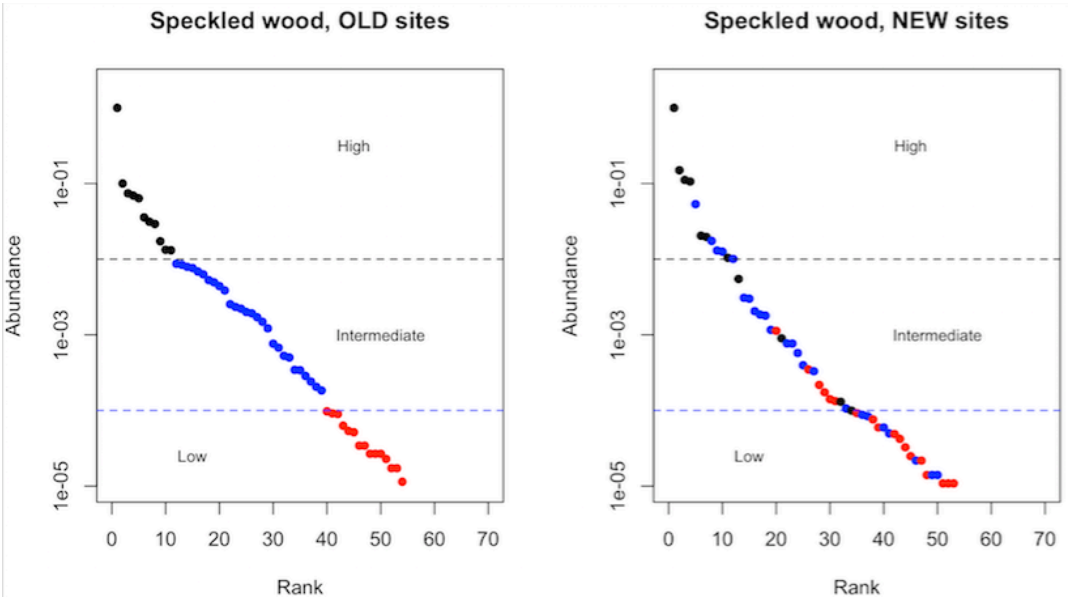
 # Speckled wood OLD SITES

```
olddataSWall=as.data.frame(rowSums(olddataSW))
newdataSWall=as.data.frame(rowSums(newdataSW))
olddataSWall=as.data.frame(cbind(gsub("OTU", "",
rownames(olddataSWall)),olddataSWall[,1],rep(0,nrow(olddataSWall))))
colnames(olddataSWall)=c("OTU","old","cat")
olddataSWall$OTU=as.numeric(as.character(olddataSWall$OTU))
olddataSWall$old=as.numeric(as.character(olddataSWall$old))
olddataSWall$old=normalize(olddataSWall$old) # We normalize data to constraint
values to {0,1}
olddataSWall$cat=as.numeric(as.character(olddataSWall$cat))
olddataSWall=olddataSWall %>% arrange(desc(old))
for (i in c(1:nrow(olddataSWall))) {
    if (olddataSWall[i,2]>0.01) {
        olddataSWall[i,3]="black"
        }
    if (olddataSWall[i,2]<=0.01 && olddataSWall[i,2]>0.0001) {
        olddataSWall[i,3]="blue"
        }
    if (olddataSWall[i,2]<=0.0001) {
        olddataSWall[i,3]="red"
        }
    }
```

 # Speckled wood NEW SITES

```
newdataSWall=as.data.frame(rowSums(newdataSW))
newdataSWall=as.data.frame(rowSums(newdataSW))
newdataSWall=as.data.frame(cbind(gsub("OTU", "",
rownames(newdataSWall)),newdataSWall[,1],rep(0,nrow(newdataSWall))))
colnames(newdataSWall)=c("OTU","new","cat")
newdataSWall$OTU=as.numeric(as.character(newdataSWall$OTU))
newdataSWall$new=as.numeric(as.character(newdataSWall$new))
newdataSWall$new=normalize(newdataSWall$new) # We normalize data to
constraint values to {0,1}
newdataSWall$cat=as.numeric(as.character(newdataSWall$cat))
newdataSWall$cat=(olddataSWall %>% arrange(OTU))[3]$cat
```

```
newdataSWall=newdataSWall %>% arrange(desc(new))
# Plotting
par(mfrow=c(1,2))
plot(rev(sort(olddataSWall[,2])),log="y",xlim=c(0,70),ylim=c(0.00001,2),col=olddataS
Wall$cat,pch=16,xlab="Rank",ylab ="Abundance",main="Speckled wood \nOLD
sites")
abline(h=0.01,col="black",lty=2)
abline(h=0.0001,col="blue",lty=2)
text(45,0.3,"High",cex=0.8)
text(50,0.001,"Intermediate",cex=0.8)
text(15,0.000025,"Low",cex=0.8)
plot(rev(sort(newdataSWall[,2])),log="y",xlim=c(0,70),ylim=c(0.00001,2),col=newdata
SWall$cat,pch=16,xlab="Rank",ylab ="Abundance",main="Speckled wood \nNEW
sites")
abline(h=0.01,col="black",lty=2)
abline(h=0.0001,col="blue",lty=2)
text(45,0.3,"High",cex=0.8)
text(50,0.001,"Intermediate",cex=0.8)
text(15,0.000025,"Low",cex=0.8)
```



 # Silver-spotted skipper OLD SITES

```
olddataSSSall=as.data.frame(rowSums(olddataSSS))
newdataSSSall=as.data.frame(rowSums(newdataSSS))
olddataSSSall=as.data.frame(cbind(gsub("OTU", "",
rownames(olddataSSSall)),olddataSSSall[,1],rep(0,nrow(olddataSSSall))))
colnames(olddataSSSall)=c("OTU","old","cat")
olddataSSSall$OTU=as.numeric(as.character(olddataSSSall$OTU))
olddataSSSall$old=as.numeric(as.character(olddataSSSall$old))
olddataSSSall$old=normalize(olddataSSSall$old) # We normalize data to constraint
values to {0,1}
olddataSSSall$cat=as.numeric(as.character(olddataSSSall$cat))
olddataSSSall=olddataSSSall %>% arrange(desc(old))
# WE CHANGE CRITERIA TO ESTABLISH CATEGORIES AS THE LOW
```

ABUNDANCE OTUs WITH THE CRITERIA ABOVE DOES NOT FIND ANY OTU

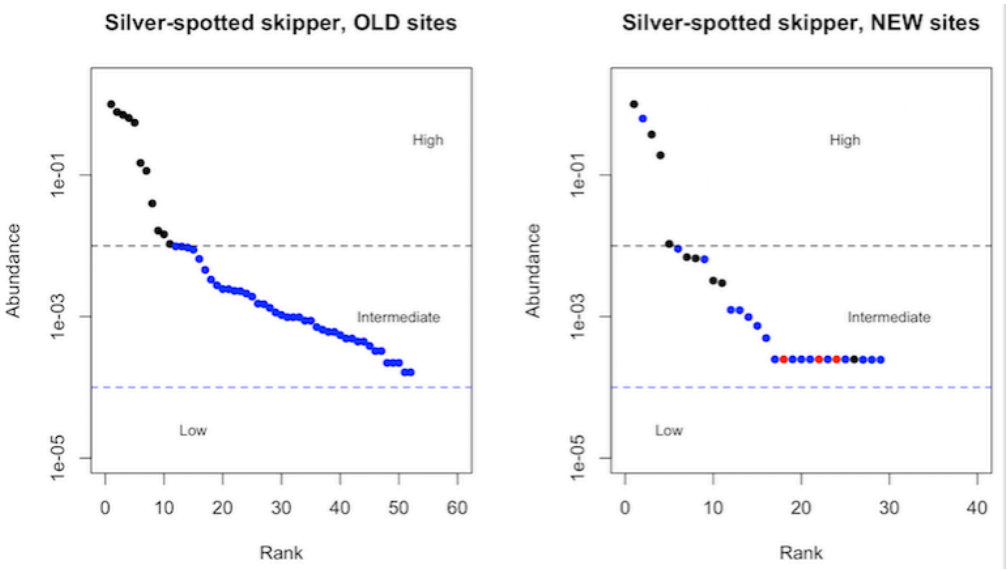
```
for (i in c(1:nrow(olddataSSSall))) {
	if (olddataSSSall[i,2]>0.01) {
		olddataSSSall[i,3]="black"
		}
	if (olddataSSSall[i,2]<=0.01 && olddataSSSall[i,2]>0.001) {
		olddataSSSall[i,3]="blue"
		}
	if (olddataSSSall[i,2]<=0.001) {
		olddataSSSall[i,3]="red"
		}
	}
```

 # Silver-spotted skipper NEW SITES

```
newdataSSSall=as.data.frame(rowSums(newdataSSS))
newdataSSSall=as.data.frame(rowSums(newdataSSS))
newdataSSSall=as.data.frame(cbind(gsub("OTU", "",
rownames(newdataSSSall)),newdataSSSall[,1],rep(0,nrow(newdataSSSall))))
colnames(newdataSSSall)=c("OTU","new","cat")
newdataSSSall$OTU=as.numeric(as.character(newdataSSSall$OTU))
newdataSSSall$new=as.numeric(as.character(newdataSSSall$new))
newdataSSSall$new=normalize(newdataSSSall$new) # We normalize data to
constraint values to {0,1}
newdataSSSall$cat=as.numeric(as.character(newdataSSSall$cat))
newdataSSSall$cat=(olddataSSSall %>% arrange(OTU))[3]$cat
newdataSSSall=newdataSSSall %>% arrange(desc(new))
```

 # Plotting

```
par(mfrow=c(1,2))
plot(rev(sort(olddataSSSall[,2])),log="y",xlim=c(0,60),ylim=c(0.0001,2),col=olddataS
SSall$cat,pch=16,xlab="Rank",ylab ="Abundance",main="Silver-spotted skipper
\nOLD sites")
abline(h=0.01,col="black",lty=2)
abline(h=0.001,col="blue",lty=2)
text(40,0.3,"High",cex=0.8)
text(40,0.005,"Intermediate",cex=0.8)
text(15,0.00025,"Low",cex=0.8)
plot(rev(sort(newdataSSSall[,2])),log="y",xlim=c(0,40),ylim=c(0.0001,2),col=newdata
SSSall$cat,pch=16,xlab="Rank",ylab ="Abundance",main="Silver-spotted skipper
\nNEW sites")
abline(h=0.01,col="black",lty=2)
abline(h=0.001,col="blue",lty=2)
text(25,0.3,"High",cex=0.8)
text(30,0.005,"Intermediate",cex=0.8)
text(5,0.00025,"Low",cex=0.8)
```



##
##
oldBA=cbind(rownames(olddataBA),olddataBA)
oldBA %>% arrange(desc(old1)) # 60,62,67
oldBA %>% arrange(desc(old2)) # 60,67,62
oldBA %>% arrange(desc(old3)) # 67,60,84,62
oldBA %>% arrange(desc(old4)) # 67,39,62,60
newBA=cbind(rownames(newdataBA),newdataBA)
newBA %>% arrange(desc(new1)) # 67,60,62
newBA %>% arrange(desc(new2)) # 67,60,62,84
newBA %>% arrange(desc(new3)) # 67,62,60
newBA %>% arrange(desc(new4)) # 60,39,18(this is exclusive to new range!),67,84,62
newBA %>% arrange(desc(new5)) # 62,67,84,79,60
oldSW=cbind(rownames(olddataSW),olddataSW)
oldSW %>% arrange(desc(old5)) # 67,90,60
oldSW %>% arrange(desc(old6)) # 67,18,79,60
oldSW %>% arrange(desc(old7)) # 67,60,79,84
newSW=cbind(rownames(newdataSW),newdataSW)
newSW %>% arrange(desc(new6)) # 67,79,88,84
newSW %>% arrange(desc(new7)) # 67,79,84,88
newSW %>% arrange(desc(new8)) # 67,84,47,79
oldSSS=cbind(rownames(olddataSSS),olddataSSS)
oldSSS %>% arrange(desc(old8)) # 79,78,67,60
oldSSS %>% arrange(desc(old9)) # 84,79,60,106(exclusive of old range),67,62
oldSSS %>% arrange(desc(old10)) # 67,78,60,84,62,79
newSSS=cbind(rownames(newdataSSS),newdataSSS)
newSSS %>% arrange(desc(new9)) # 79,67,81
newSSS %>% arrange(desc(new10)) # 79,81,60,67
##
##

Beta-diversity analysis: comparing old Vs new sites for each

butterfly

```
# We use betadiver(), which calculates beta diversity indeces of Koleff et al (JAE 2003, Table 1).
# Below, we select Whittaker's index

# BROWN ARGUS
olddataBAall_ab1=filter(olddataBAall,cat=="black")
olddataBAall_ab2=filter(olddataBAall,cat=="blue")
olddataBAall_ab3=filter(olddataBAall,cat=="red")
# For new sites, we need to locate most, intermediate and least abundant species. BEfore we use eye comparisons to see if there's changes in the RADs between old and new ranges. Now we want to do a beta-diversity analysis
for (i in c(1:nrow(newdataBAall))) {
	if (newdataBAall[i,2]>0.01) {
		newdataBAall[i,3]="black"
		}
	if (newdataBAall[i,2]<=0.01 && newdataBAall[i,2]>0.0001) {
		newdataBAall[i,3]="blue"
		}
	if (newdataBAall[i,2]<=0.0001) {
		newdataBAall[i,3]="red"
		}
	}
newdataBAall_ab1=filter(newdataBAall,cat=="black")
newdataBAall_ab2=filter(newdataBAall,cat=="blue")
newdataBAall_ab3=filter(newdataBAall,cat=="red")
# Most abundant species
BA_ab1=cbind(olddataBAall_ab1[,1:2],newdataBAall_ab1[1:2])
BA_ab1 <- full_join(olddataBAall_ab1[,1:2],newdataBAall_ab1[1:2], by = c('OTU'))
BA_ab1[is.na(BA_ab1)] <- 0
BA_ab1=as.data.frame(t(BA_ab1))
colnames(BA_ab1)=BA_ab1[1,]
BA_ab1[,1]=rownames(BA_ab1)
BA_ab1=BA_ab1[-1,]
BA_ab1.beta=betadiver(BA_ab1,method="w") #0.22
# Intermediate abundant species
BA_ab2=cbind(olddataBAall_ab2[,1:2],newdataBAall_ab2[1:2])
BA_ab2 <- full_join(olddataBAall_ab2[,1:2],newdataBAall_ab2[1:2], by = c('OTU'))
BA_ab2[is.na(BA_ab2)] <- 0
BA_ab2=as.data.frame(t(BA_ab2))
colnames(BA_ab2)=BA_ab2[1,]
BA_ab2[,1]=rownames(BA_ab2)
BA_ab2=BA_ab2[-1,]
BA_ab2.beta=betadiver(BA_ab2,method="w") #0.34
# High abundant species
BA_ab3=cbind(olddataBAall_ab3[,1:2],newdataBAall_ab3[1:2])
BA_ab3 <- full_join(olddataBAall_ab3[,1:2],newdataBAall_ab3[1:2], by = c('OTU'))
BA_ab3[is.na(BA_ab3)] <- 0
BA_ab3=as.data.frame(t(BA_ab3))
colnames(BA_ab3)=BA_ab3[1,]
```

```
BA_ab3[,1]=rownames(BA_ab3)
BA_ab3=BA_ab3[-1,]
BA_ab3.beta=betadiver(BA_ab3,method="w") #0.75

# SPECKLED WOOD
olddataSWall_ab1=filter(olddataSWall,cat=="black")
olddataSWall_ab2=filter(olddataSWall,cat=="blue")
olddataSWall_ab3=filter(olddataSWall,cat=="red")
# For new sites, we need to locate most, intermediate and least abundant species. BEfore we use eye comparisons to see if there's changes in the RADs between old and new ranges. Now we want to do a beta-diversity analysis
for (i in c(1:nrow(newdataSWall))) {
	if (newdataSWall[i,2]>0.01) {
		newdataSWall[i,3]="black"
		}
	if (newdataSWall[i,2]<=0.01 && newdataSWall[i,2]>0.0001) {
		newdataSWall[i,3]="blue"
		}
	if (newdataSWall[i,2]<=0.0001) {
		newdataSWall[i,3]="red"
		}
	}
newdataSWall_ab1=filter(newdataSWall,cat=="black")
newdataSWall_ab2=filter(newdataSWall,cat=="blue")
newdataSWall_ab3=filter(newdataSWall,cat=="red")
# Most abundant species
SW_ab1=cbind(olddataSWall_ab1[,1:2],newdataSWall_ab1[1:2])
SW_ab1 <- full_join(olddataSWall_ab1[,1:2],newdataSWall_ab1[1:2], by = c('OTU'))
SW_ab1[is.na(SW_ab1)] <- 0
SW_ab1=as.data.frame(t(SW_ab1))
colnames(SW_ab1)=SW_ab1[1,]
SW_ab1[,1]=rownames(SW_ab1)
SW_ab1=SW_ab1[-1,]
SW_ab1.beta=betadiver(SW_ab1,method="w") #0.39
# Intermediate abundant species
SW_ab2=cbind(olddataSWall_ab2[,1:2],newdataSWall_ab2[1:2])
SW_ab2 <- full_join(olddataSWall_ab2[,1:2],newdataSWall_ab2[1:2], by = c('OTU'))
SW_ab2[is.na(SW_ab2)] <- 0
SW_ab2=as.data.frame(t(SW_ab2))
colnames(SW_ab2)=SW_ab2[1,]
SW_ab2[,1]=rownames(SW_ab2)
SW_ab2=SW_ab2[-1,]
SW_ab2.beta=betadiver(SW_ab2,method="w") #0.48
# High abundant species
SW_ab3=cbind(olddataSWall_ab3[,1:2],newdataSWall_ab3[1:2])
SW_ab3 <- full_join(olddataSWall_ab3[,1:2],newdataSWall_ab3[1:2], by = c('OTU'))
SW_ab3[is.na(SW_ab3)] <- 0
SW_ab3=as.data.frame(t(SW_ab3))
colnames(SW_ab3)=SW_ab3[1,]
SW_ab3[,1]=rownames(SW_ab3)
```

```
SW_ab3=SW_ab3[-1,]
SW_ab3.beta=betadiver(SW_ab3,method="w") #0.71

# SILVER-SPOTTED SKIPPER
olddataSSSall_ab1=filter(olddataSSSall,cat=="black")
olddataSSSall_ab2=filter(olddataSSSall,cat=="blue")
olddataSSSall_ab3=filter(olddataSSSall,cat=="red")
# For new sites, we need to locate most, intermediate and least abundant species. BEfore we use eye comparisons to see if there's changes in the RADs between old and new ranges. Now we want to do a beta-diversity analysis
for (i in c(1:nrow(newdataSSSall))) {
	if (newdataSSSall[i,2]>0.01) {
		newdataSSSall[i,3]="black"
		}
	if (newdataSSSall[i,2]<=0.01 && newdataSSSall[i,2]>0.001) {
		newdataSSSall[i,3]="blue"
		}
	if (newdataSSSall[i,2]<=0.001) {
		newdataSSSall[i,3]="red"
		}
	}
newdataSSSall_ab1=filter(newdataSSSall,cat=="black")
newdataSSSall_ab2=filter(newdataSSSall,cat=="blue")
newdataSSSall_ab3=filter(newdataSSSall,cat=="red")
# Most abundant species
SSS_ab1=cbind(olddataSSSall_ab1[,1:2],newdataSSSall_ab1[1:2])
SSS_ab1 <- full_join(olddataSSSall_ab1[,1:2],newdataSSSall_ab1[1:2], by =
c('OTU'))
SSS_ab1[is.na(SSS_ab1)] <- 0
SSS_ab1=as.data.frame(t(SSS_ab1))
colnames(SSS_ab1)=SSS_ab1[1,]
SSS_ab1[,1]=rownames(SSS_ab1)
SSS_ab1=SSS_ab1[-1,]
SSS_ab1.beta=betadiver(SSS_ab1,method="w") #0.5
# Intermediate abundant species
SSS_ab2=cbind(olddataSSSall_ab2[,1:2],newdataSSSall_ab2[1:2])
SSS_ab2 <- full_join(olddataSSSall_ab2[,1:2],newdataSSSall_ab2[1:2], by =
c('OTU'))
SSS_ab2[is.na(SSS_ab2)] <- 0
SSS_ab2=as.data.frame(t(SSS_ab2))
colnames(SSS_ab2)=SSS_ab2[1,]
SSS_ab2[,1]=rownames(SSS_ab2)
SSS_ab2=SSS_ab2[-1,]
SSS_ab2.beta=betadiver(SSS_ab2,method="w") #0.79
# High abundant species
SSS_ab3=cbind(olddataSSSall_ab3[,1:2],newdataSSSall_ab3[1:2])
SSS_ab3 <- full_join(olddataSSSall_ab3[,1:2],newdataSSSall_ab3[1:2], by =
c('OTU'))
SSS_ab3[is.na(SSS_ab3)] <- 0
SSS_ab3=as.data.frame(t(SSS_ab3))
```

```
colnames(SSS_ab3)=SSS_ab3[1,]
SSS_ab3[,1]=rownames(SSS_ab3)
SSS_ab3=SSS_ab3[-1,]
SSS_ab3.beta=betadiver(SSS_ab3,method="w") #0.69
```