Queiroz et al. 2020

Marco Mello

10/6/2020

Supplement to the paper Queiroz *et al.* (2020, Biotropica).

Ecological Synthesis Lab (SintECO).

Authors: Joel A. Queiroz, Ugo M. Diniz, Diego P. Vázquez, Zelma M. Quirino, Francisco A.R. Santos, Marco A.R. Mello, Isabel C. Machado.

See README for further info.

This tutorial is aimed at facilitating full reproductibility of the analyses and figures presented in our paper.

Summary

Set the stage

Set the working directory:

```
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))
```

Delete all previous objects:

```
rm(list= ls())
```

Load the required packages:

```
library(igraph)
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```
library(bipartite)
```

```
## Loading required package: vegan

## Loading required package: permute

##
## Attaching package: 'permute'

## The following object is masked from 'package:igraph':
##
##     permute

## Loading required package: lattice

## This is vegan 2.5-6

##
## Attaching package: 'vegan'

## The following object is masked from 'package:igraph':
##
##     diversity

## Loading required package: sna

## Loading required package: statnet.common

##
## Attaching package: 'statnet.common'

## The following object is masked from 'package:base':
##
##     order

## Loading required package: network

## network: Classes for Relational Data
## Version 1.16.0 created on 2019-11-30.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##                     Mark S. Handcock, University of California -- Los Angeles
##                     David R. Hunter, Penn State University
##                     Martina Morris, University of Washington
##                     Skye Bender-deMoll, University of Washington
##  For citation information, type citation("network").
##  Type help("network-package") to get started.
```

```
##
## Attaching package: 'network'

## The following objects are masked from 'package:igraph':
##
##     %c%, %s%, add.edges, add.vertices, delete.edges, delete.vertices,
##     get.edge.attribute, get.edges, get.vertex.attribute, is.bipartite,
##     is.directed, list.edge.attributes, list.vertex.attributes,
##     set.edge.attribute, set.vertex.attribute

## sna: Tools for Social Network Analysis
## Version 2.5 created on 2019-12-09.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##  For citation information, type citation("sna").
##  Type help(package="sna") to get started.

##
## Attaching package: 'sna'

## The following objects are masked from 'package:igraph':
##
##     betweenness, bonpow, closeness, components, degree, dyad.census,
##     evcent, hierarchy, is.connected, neighborhood, triad.census

##  This is bipartite 2.15.
##  For latest changes see versionlog in ?"bipartite-package". For citation see: citation("bipartite").
##  Have a nice time plotting and analysing two-mode networks.

##
## Attaching package: 'bipartite'

## The following object is masked from 'package:vegan':
##
##     nullmodel

## The following object is masked from 'package:igraph':
##
##     strength

library(Rmisc)


## Loading required package: plyr

##
## Attaching package: 'plyr'

## The following object is masked from 'package:bipartite':
##
##     empty

## The following object is masked from 'package:network':
##
##     is.discrete
```

```
library(vegan)
library(gdata)
```

```
## gdata: read.xls support for 'XLS' (Excel 97-2004) files ENABLED.

##

## gdata: read.xls support for 'XLSX' (Excel 2007+) files ENABLED.

##
## Attaching package: 'gdata'

## The following object is masked from 'package:stats':
##
##     nobs

## The following object is masked from 'package:utils':
##
##     object.size

## The following object is masked from 'package:base':
##
##     startsWith
```

```
library(ggplot2)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:gdata':
##
##     combine
```

```
library(grid)
```

Load the custom-made functions:

```
source("RestNullModel.R")
source("PosteriorProb.R")
source("MyDiamond.R")
```

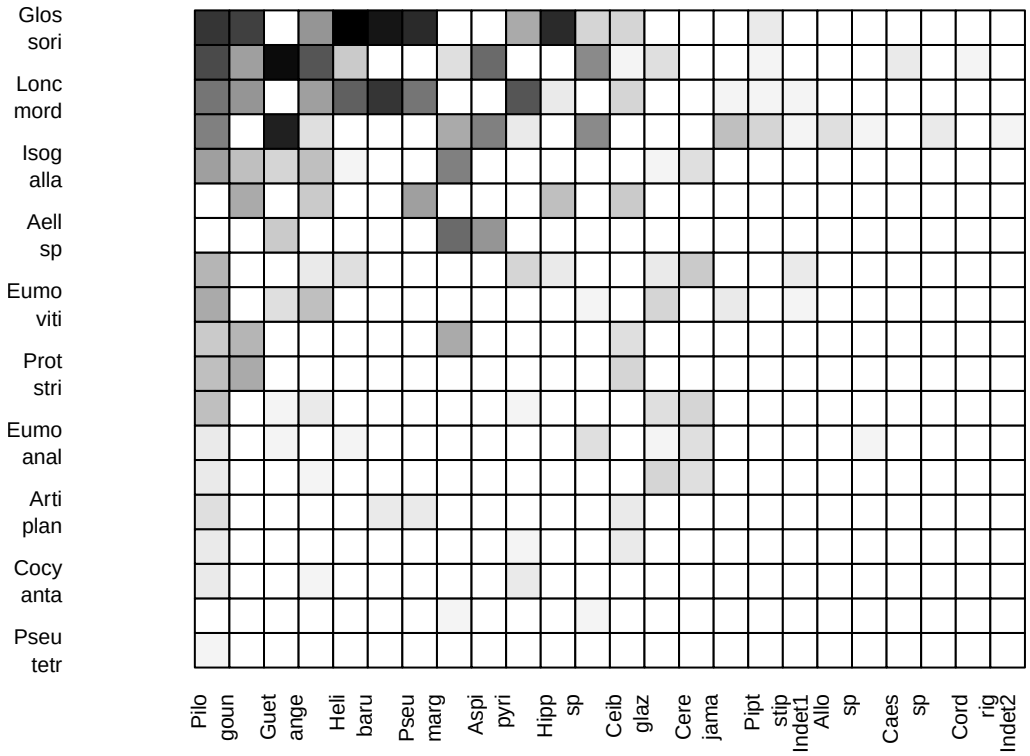
Process the network

Import the network:

```
data <- as.matrix(read.delim("data/network.txt", row.names=1))
```

Plot the matrix to have a first impression about the network's structure:

```
visweb(data)
```



Convert the network to igraph format:

```
data2 <- graph_from_incidence_matrix(data, directed = F, weighted = TRUE)
```

Inform which nodes represent which taxonomic groups:

```
V(data2)$set[1:nrow(data)] = c("Moths", "Moths", "Bats",    "Bats", "Moths",    "Moths",
                       "Moths", "Moths",    "Moths",    "Bats", "Bats", "Moths",
                       "Moths", "Moths",    "Moths",    "Moths",    "Moths",
                       "Moths", "Moths")
```

```
## Warning in vattrs[[name]][index] <- value: número de itens para para substituir
## não é um múltiplo do comprimento do substituto
```

```
V(data2)$set[(nrow(data)+1):(nrow(data)+ncol(data))] = "Plants"
```

Figure 1

This figure was made in Photoshop as a panel of photos taken in the field. Therefore, it is not reproducible by code

Figure 2

This is the graph that represents the nocturnal pollination network studied.

The original graph was made in Gephi, so it is reproducible only by pseudocode. Here we describe the steps needed to reproduce it

In the next section we provide also code to plot a similar graph in R.

Pseudocode to reproduce Figure 2 in Gephi

1. Prepare the data that will be fed to Gephi:

    a. "vertices.csv" (folder "figure2") containing vertex data. Each row corresponds to a vertex, and each column, a descriptor. Contains the columns "Id" (vertex ID), "Label" (certex code), "timeset" (left blank), "Polygon" (the shape of the vertex, the number represents the number of sides of the polygon) and "syndrome" (a discreet variable representing chiropterophily, sphingophily, other syndromes, unidentified species or pollinators).

    b. "edges.csv" (folder "figure2") containing edge data. Each row corresponds to an interaction. Contain the columns "Source" and "Target" - the two interacting species – "Type" (directed or undirected), "Id", "label" (left blank), "timeset" (left blank), and "weight" (interaction frequency).

2. Open Gephi, click "Tools" in the upper left corner and select "Plugins". Search for the plugin "Polygon Shaped Nodes" and install it. When successfully installed, restart Gephi.

3. Head to Data Laboratory and select "Import Spreadsheet". Import first the "vertices.csv" file as a Nodes table, selecting comma as a separator. Click 'next'. Make sure both "Polygon" and "Syndrome" are marked as integers, then click "Finish". Select Graph Type "undirected" and click "OK".

4. Import the "edges.csv" file as an Edges table, selecting comma as a separator. Click "next". Make sure that "Weight" is marked as an integer and click "finish". Mark "Append to existing workplace" to associate the edges with the vertices already imported.

5. Head to Overview. In the Appearance tap on the left side of the window, click "Nodes", select "Partition", and choose "Syndrome" as partition. Click the squares to manually tweak node color by syndrome. Note that pollinators do not fit into syndromes and are represented by zeros.

6. Still in Nodes, switch to node size (icon containing growing circles), select "Unique", and change node size according to preference. In the "Edge" section, click "Unique" and then click the square to choose edge color. Click "Apply" below to make changes effective.

7. In the central part of the Overview window, change node position according to preference. The positions intended for our work are such that modules (identified by the network analysis in R) are visually clear.

8. Head to Preview. In Presets, select "Default Straight". In the Node Label section, check "Show labels" and choose label font and size. In the Edges section, check "Show edges", select "original" in Color, and uncheck "Curved". Any other setting in the Preview window may be changed according to preference.

9. Export file in preferred format. Module polygons were drawn in an image editing software, to which SVG format is optimal.

Code to reproduce Figure 2 in R

We are going to use the same igraph object prepared before:

```
data2
```

```
## IGRAPH 06d36e8 UNWB 43 118 --
## + attr: type (v/l), name (v/c), set (v/c), weight (e/n)
## + edges from 06d36e8 (vertex names):
##  [1] Call_gris--Pilo_goun Call_gris--Bauh_chei Call_gris--Ipom_marc
##  [4] Call_gris--Guet_ange Call_gris--Ambu_cear Call_gris--Crot_sp
##  [7] Call_gris--Indet1    Call_gris--Pipt_stip Call_gris--Aspi_pyri
## [10] Call_gris--Comb_sp   Call_gris--Anad_colu Call_gris--Allo_sp
## [13] Call_gris--Schi_bras Call_gris--Indet2    Erin_ello--Pilo_goun
## [16] Erin_ello--Bauh_chei Erin_ello--Ceib_glaz Erin_ello--Ench_spec
## [19] Erin_ello--Guet_ange Erin_ello--Toco_form Erin_ello--Ambu_cear
## [22] Erin_ello--Heli_baru Erin_ello--Crot_sp   Erin_ello--Pipt_stip
## + ... omitted several edges
```

Set an energy-minimization layout:

```
lay1 <- layout_nicely(data2)
```

Set edge mode and width:

```
E(data2)$arrow.mode = 0
E(data2)$width = E(data2)$weight/5+1
```

Import the "diamond" vertex shape:

```
source("MyDiamond.R")
```

Set vertex shapes:

```
V(data2)$shape = V(data2)$set
V(data2)$shape = gsub("Bats","diamond",V(data2)$shape)
V(data2)$shape = gsub("Moths","square",V(data2)$shape)
V(data2)$shape = gsub("Plants","circle",V(data2)$shape)
```

Calculate the Louvain modularity (resolution = 1.0).

At this resolution, this algorithm gives very similar results as DIRT_LPA+ (Beckett modularity). Nevertheless, by running this analysis with the *igraph* package, it is easier to plot the graph with clouds representing the modules. We did not find a similar solution using outputs from the *bipartite* package.

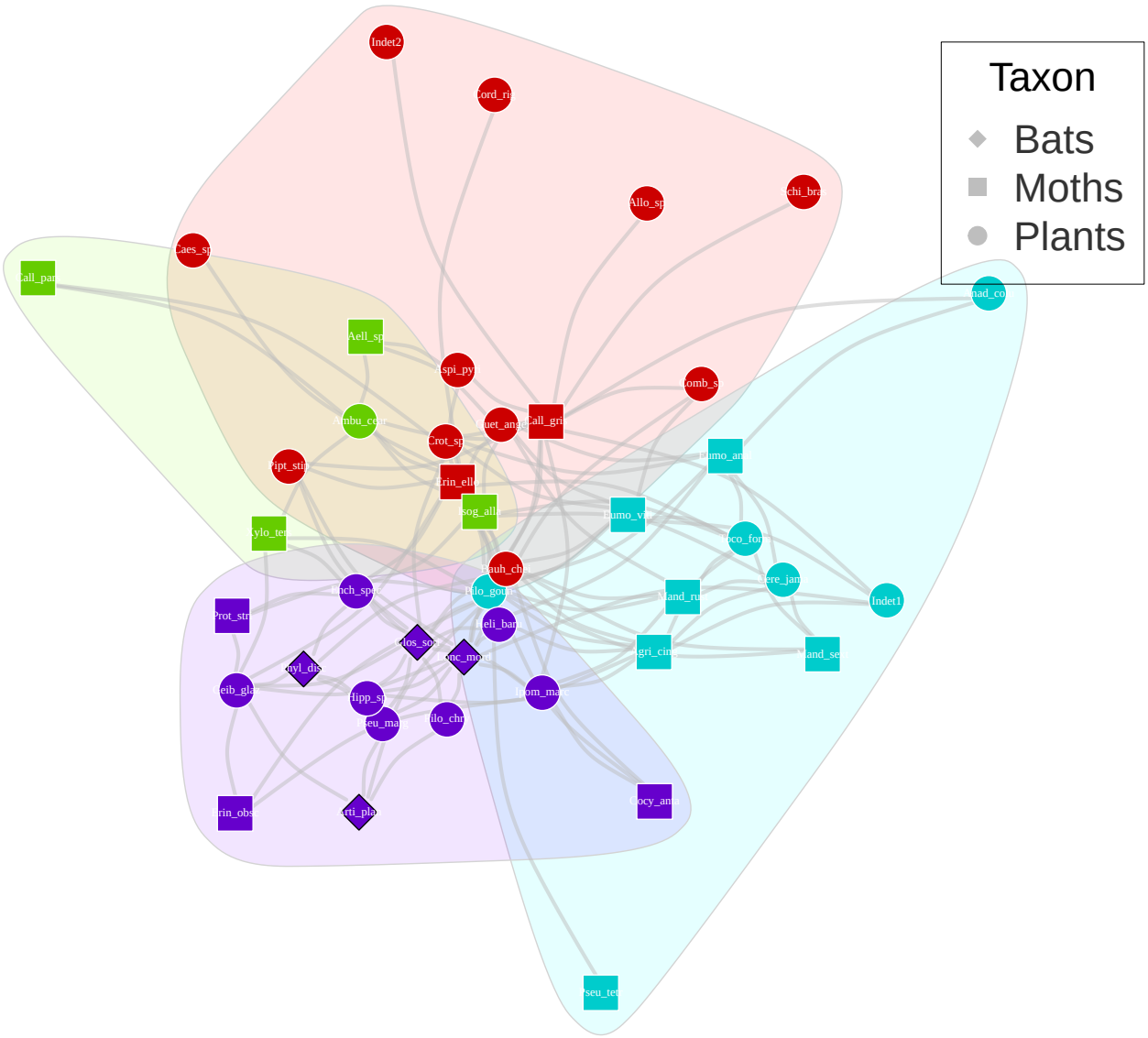
```
data2.lou = cluster_louvain(data2)
```

Set node and cloud colors by modularity:

```
colrs <- rainbow(length(data2.lou), alpha = 1.0, s = 1, v = 0.8)
V(data2)$color <- colrs[data2.lou$membership]
clouds = rainbow(length(data2.lou), alpha = 0.1)
```

Plot Figure 2:

```
par(mfrow=c(1,1),mar=c(1,1,1,5))
plot(data2.lou,
     data2,
     col = V(data2)$color,
     mark.border="lightgrey",
     mark.col=clouds,
     vertex.size=7.5,
     vertex.label=V(data2)$name,
     vertex.label.color="white",
     vertex.label.cex=.6,
     edge.color = adjustcolor("grey", alpha.f = .5),
     edge.curved=0.3,
     edge.width = 3,
     layout=lay1)
legend(x = 0.9,y = 1.0, legend = c("Bats", "Moths", "Plants"),
       pch = c(18,15,19),  title="Taxon",
       text.col = "gray20", title.col = "black",
       box.lwd = 0, cex = 2, col=c("grey", "grey", "grey"))
```




```
par(mfrow=c(1,1))
```

Figure S2

Here we are going to reproduce the topological analysis of the nocturnal networks. It takes several steps.

In the end, we are going to reproduce Figure S2, provided in the supplementary material, which represents the compound topology of the network.

First, set the number of permutations to be used in all null model analyses.

Consider that this kind of analysis is very resource-consuming. So have in mind your comoputer's power and memory, before setting this value.

```
permutations <- 10
```

Generate randomized matrices using the Vázquez null model:

```
nulls <- nullmodel(data, N=permutations, method="vaznull")
```

Modularity

Calculate modularity (DIRT_LPA+) for the observed network:

```
Mod <- computeModules(data, method = "Beckett")
```

Extract module membership:

```
Part <- bipartite::module2constraints(Mod)
row.Part <- Part[1:nrow(data)]
col.Part <- Part[(nrow(data)+1):(nrow(data)+ncol(data))]
```

Calculate modularity for the randomized networks:

```
nullmod <- sapply(nulls, computeModules, method = "Beckett")
modnull <- sapply(nullmod, function(x) x@likelihood)
(Mod@likelihood - mean(modnull))/sd(modnull) # Z value
```

```
## [1] 15.9225
```

```
Mod.sig <- sum(modnull>(Mod@likelihood)) / length(modnull) # p value
```

Now let us plot the observed value against the distribution of randomized values.

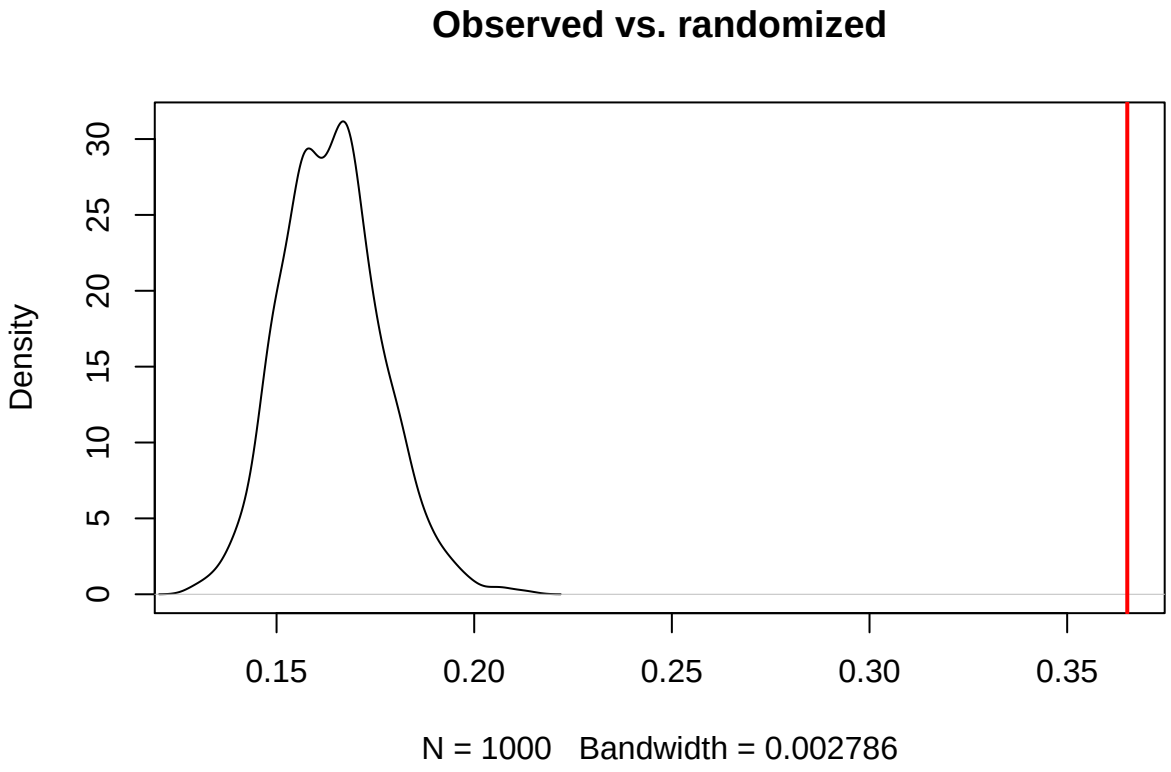
If the observed value (red line) falls much **higher** than the randomized values (black curve), it means that the topology in question might be a good explanation for the structure of the network, as it is much higher than expected by chance.

If the observed value falls much **lower** then than randomized values, it means that the topology in question is probably a poor explanation for the structure of the networks, as it is much lower than expected by chance.

Nevertheless, keep in mind that, on the one hand, the score of a network for a given topological metric is one of its intrinsic properties. On the other hand, the p-value estimated using a null model is a different property. Therefore, it is meaningless to think black-and-white in terms of the network being nested or not, modular or not, specialized or not. Those properties are continuous and intrinsic. The chance of a particula score having emerged by chance is a another story.

Plot the curve:

```
plot(density(modnull), main="Observed vs. randomized",
     xlim=c(min((Mod@likelihood), min(modnull)),
            max((Mod@likelihood), max(modnull))))
abline(v=Mod@likelihood, col="red", lwd=2, xlab="")
```



Estimate the p-values:

```
Mod@likelihood #observed
```

```
## [1] 0.365208
```

```
mean(modnull) #randomized mean
```

```
## [1] 0.1639436
```

```
sd(modnull) #randomized SD
```

```
## [1] 0.01264025
```

```
(Mod@likelihood - mean(modnull))/sd(modnull) # Z-value
```

```
## [1] 15.9225
```

```
sum(modnull>(Mod@likelihood)) / length(modnull) #randomized > observed
```

```
## [1] 0
```

```
sum(modnull<(Mod@likelihood)) / length(modnull) #randomized < observed
```

```
## [1] 1
```

Specialization

Calculate specialization (H2') for the observed network:

```
Spec <- networklevel(data, index="H2")
```

Calculate specialization for the randomized networks:

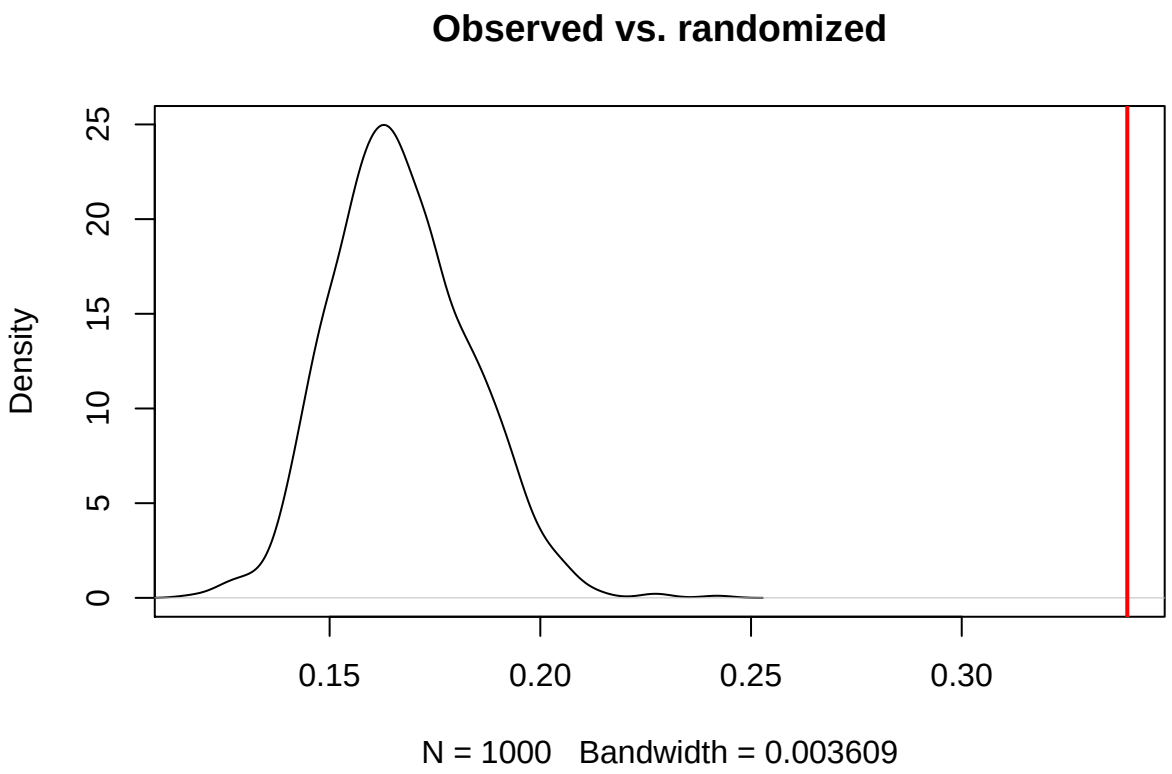
```
randomized.Spec <- unlist(sapply(nulls, networklevel, index="H2"))
(Spec - mean(randomized.Spec))/sd(randomized.Spec) # Z value
```

```
##       H2
## 10.57605
```

```
Spec.sig <- sum(randomized.Spec>Spec)/length(randomized.Spec) # p value
```

Plot the observed value against the distribution of randomized values:

```
plot(density(randomized.Spec), main="Observed vs. randomized",
     xlim=c(min((Spec), min(randomized.Spec)),
            max((Spec), max(randomized.Spec))))
abline(v=Spec, col="red", lwd=2, xlab="")
```



Estimate the p-values:

```
Spec #observed
```

```
##        H2
## 0.3392862
```

```
mean(randomized.Spec) #randomized mean
```

```
## [1] 0.1670043
```

```
sd(randomized.Spec) #randomized SD
```

```
## [1] 0.01628981
```

```
(Spec - mean(randomized.Spec))/sd(randomized.Spec) # Z-value
```

```
##       H2
## 10.57605
```

```
sum(randomized.Spec>(Spec)) / length(randomized.Spec) #randomized > observed
```

```
## [1] 0
```

```
sum(randomized.Spec<(Spec)) / length(randomized.Spec) #randomized < observed
```

```
## [1] 1
```

Nestedness

#Calculate nestedness (WNODF) for the observed network:

```
Nest <- networklevel(data, index="weighted NODF")
```

Calculate nestedness for the randomized networks:

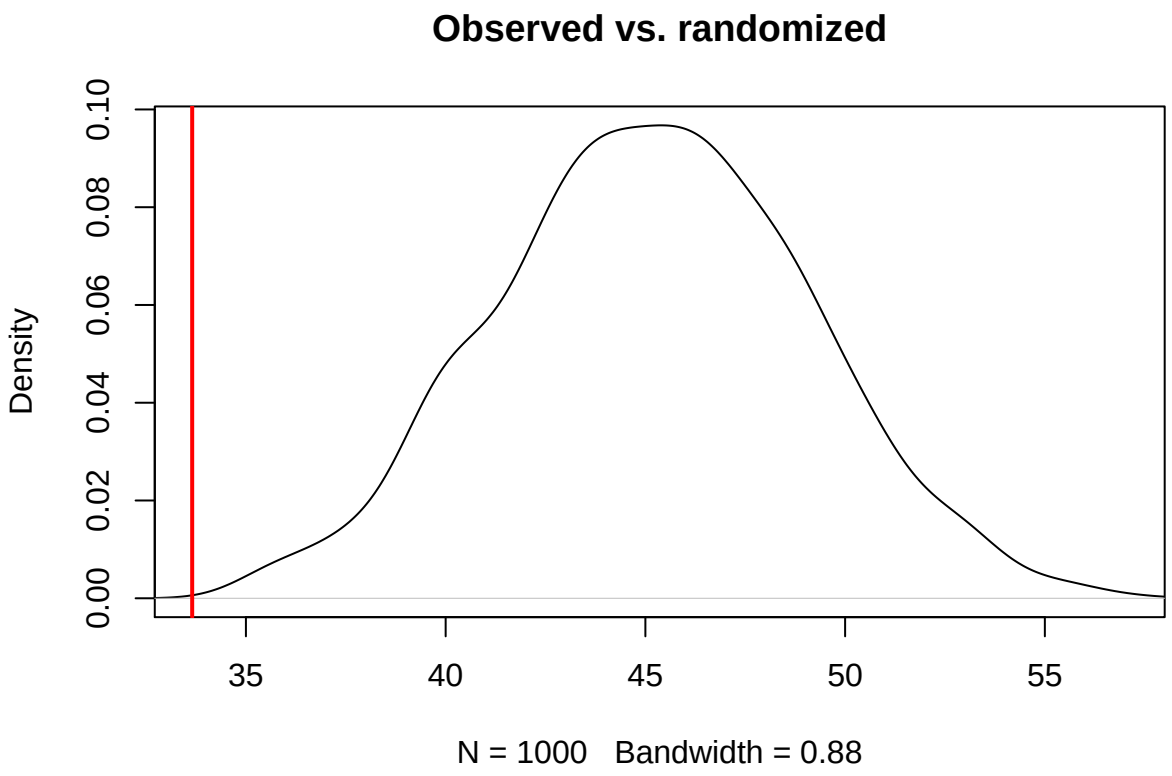
```
randomized.Nest <- unlist(sapply(nulls, networklevel, index="weighted NODF"))
(Nest - mean(randomized.Nest))/sd(randomized.Nest) # Z value
```

```
## weighted NODF
##     -2.960621
```

```
Nest.sig <- sum(randomized.Nest>Nest)/length(randomized.Nest) # p value
```

Plot the observed value against the distribution of randomized values:

```
plot(density(randomized.Nest), main="Observed vs. randomized",
     xlim=c(min((Nest), min(randomized.Nest)),
            max((Nest), max(randomized.Nest))))
abline(v=Nest, col="red", lwd=2, xlab="")
```



Estimate the p-values:

```
Nest #observed
```

```
## weighted NODF
##       33.6547
```

```
mean(randomized.Nest) #randomized mean
```

```
## [1] 45.17913
```

```
sd(randomized.Nest) #randomized SD
```

```
## [1] 3.892572
```

```
(Nest - mean(randomized.Nest))/sd(randomized.Nest) # Z-value
```

```
## weighted NODF
##     -2.960621
```

```
sum(randomized.Nest>(Nest)) / length(randomized.Nest) #randomized > observed
```

```
## [1] 1
```

```r
sum(randomized.Nest<(Nest)) / length(randomized.Nest) #randomized < observed
```

```
## [1] 0
```

Compound topology

Calculate compound nestedness (using WNODA) for the observed network:

```r
obs.com <- unlist(bipartite::nest.smdm(x = data,
                                       constraints = Part, #Input the modular structured recovered from
                                       weighted = T, #By considering the edge weights, you are choosing
                                       decreasing = "abund"))
```

Calculate constrained interaction probabilities considering the network's modular structure:

```r
Pij <- PosteriorProb(M = data,
                     R.partitions = row.Part, C.partitions = col.Part, #Input the modular structured re
                     Prior.Pij = "degreeprob", #Choose the null model
                     Conditional.level = "modules") #Choose the kind of constraints
```

Generate randomized networks with the restricted model, considering the interaction probabilities calculated before:

```r
nulls.com <- RestNullModel(M = data,
                           Pij.Prob = Pij, #Recover the probabilities calculated in the previous comman
                           Numbernulls = permutations, #This step may take long, so start experimenting
                           Print.null = F,
                           allow.degeneration = F, #Choose whether you allow orphan rows and columns to
                           return.nonrm.species = F,
                           connectance = T, byarea = T,
                           R.partitions = row.Part,
                           C.partitions = col.Part)
```

Calculate compound nestedness for the randomized networks:

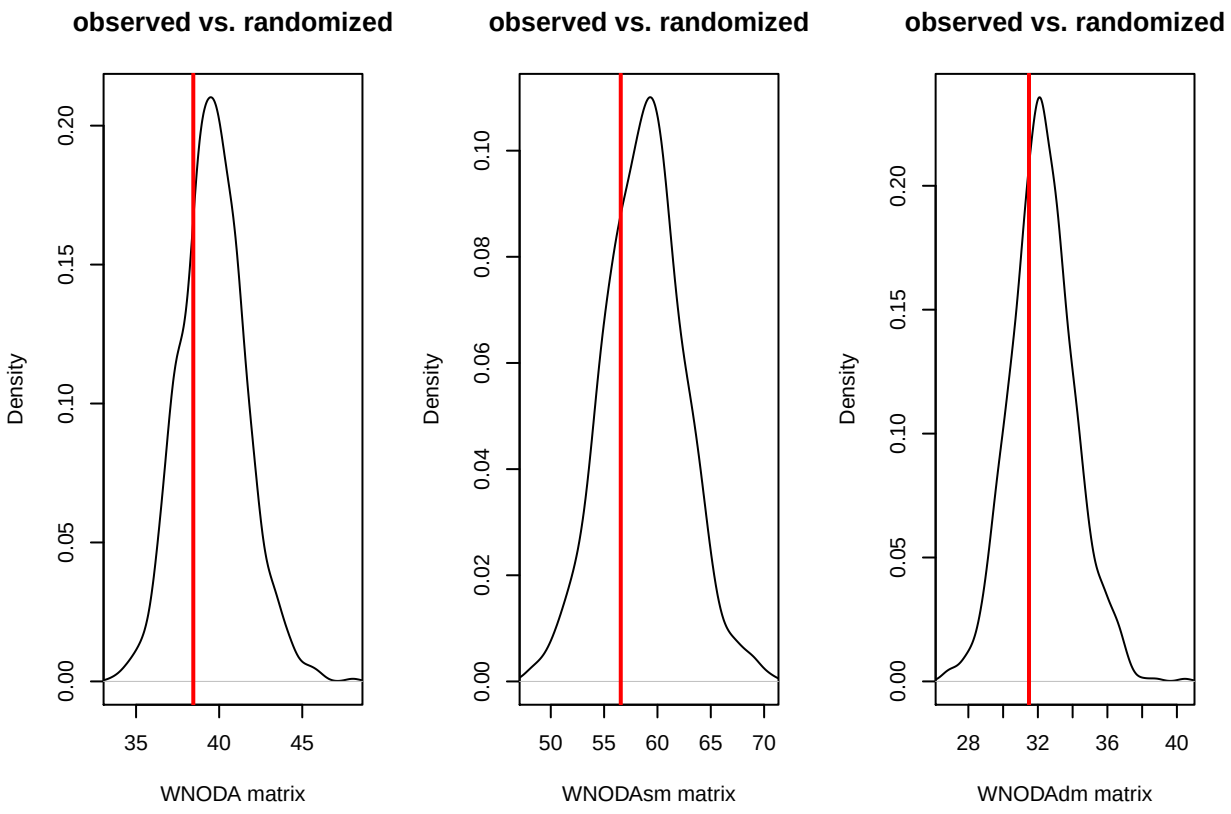
```r
rest.nest <- nest.smdm(data,
                       constraints = Part,
                       weighted = T,
                       decreasing = "abund",
                       sort = T)

unlist(rest.nest)
```

```
##        WNODArow        WNODAcol     WNODAmatrix    WNODA_SM_row    WNODA_DM_row
##        42.09696        36.18075        38.44400        59.70653        36.18125
##    WNODA_SM_col    WNODA_DM_col WNODA_SM_matrix WNODA_DM_matrix
##        54.88777        28.41015        56.55879        31.48972
```

```
null.com <- sapply(nulls.com,
                   function(x) bipartite::nest.smdm(x = x,
                                                    constraints = Part,
                                                    weighted = T,
                                                    decreasing = "abund"))
WNODA.null.com <- unlist(null.com[3,])
WNODAsm.null.com <- unlist(null.com[8,])
WNODAdm.null.com <- unlist(null.com[9,])
```

Plot the observed nestedness value against the distribution of randomized values:

```
par(mfrow = c(1,3))

plot(density(WNODA.null.com), xlim=c(min(obs.com[3], min(WNODA.null.com)),
                                     max(obs.com[3], max(WNODA.null.com))),
     main="observed vs. randomized", xlab = "WNODA matrix")
abline(v=obs.com[3], col="red", lwd=2)

plot(density(WNODAsm.null.com), xlim=c(min(obs.com[8], min(WNODAsm.null.com)),
                                       max(obs.com[8], max(WNODAsm.null.com))),
     main="observed vs. randomized", xlab = "WNODAsm matrix")
abline(v=obs.com[8], col="red", lwd=2)

plot(density(WNODAdm.null.com), xlim=c(min(obs.com[9], min(WNODAdm.null.com)),
                                       max(obs.com[9], max(WNODAdm.null.com))),
     main="observed vs. randomized", xlab = "WNODAdm matrix")
abline(v=obs.com[9], col="red", lwd=2)
```



```
par(mfrow = c(1,1))
```

Estimate the p-values:

Nestedness in the entire network:

```
praw.WNODA <- sum(WNODA.null.com>obs.com[3]) / length(WNODA.null.com)
p.WNODA <- ifelse(praw.WNODA > 0.5, 1- praw.WNODA, praw.WNODA)    # P-value
```

Nestedness within the modules:

```
praw.WNODAsm <- sum(WNODAsm.null.com>obs.com[8]) / length(WNODAsm.null.com)
p.WNODAsm <- ifelse(praw.WNODAsm > 0.5, 1- praw.WNODAsm, praw.WNODAsm)    # P-value
```

Nestedness between the modules:

```
praw.WNODAdm <- sum(WNODAdm.null.com>obs.com[9]) / length(WNODAdm.null.com)
p.WNODAdm <- ifelse(praw.WNODAdm > 0.5, 1- praw.WNODAdm, praw.WNODAdm)    # P-value
```

Plot the compound topology: Figure S2. Now we have come to it. All previous topological analyses were needed to make this final analysis.

Sort the matrix in a way that facilitates visualizing its compound topology:

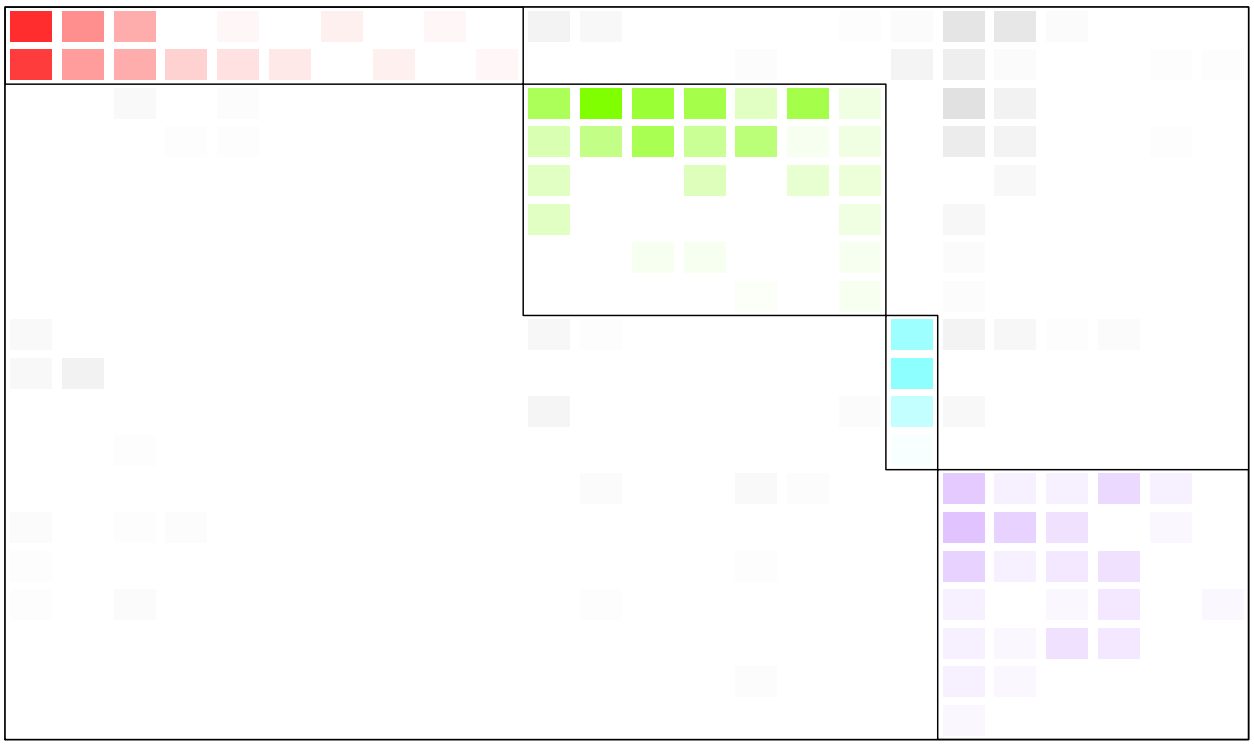
```
data.comp <- bipartite::sortmatrix(matrix = data, topology = "compound",
                                   sort_by = "weights",
                                   row_partitions = row.Part,
                                   col_partitions = col.Part)
```

Assign colors to the modules:

```
modcol <- rainbow((length(unique(Part))), alpha=1, s = 1, v = 1)
```

Plot Figure S2:

```
plotmatrix(data.comp$matrix,
           row_partitions = data.comp$row_partitions,
           col_partitions = data.comp$col_partitions,
           border = T,
           binary = F,
           modules_colors = modcol,
           within_color = modcol,
           between_color = "lightgrey")
```



Summary of the topological results

The network has 19 rows and 24 columns.

The network's specialization (H2) is 0.34, P = 0.

The network's modularity (DIRT_LPA+) is 0.37, P = 0, and it contains 4 modules.

The network's nestedness (WNODF) is 0.34, P = 1.

The network shows the following scores of nestedness (WNODA):

Entire network = 0.38, P = 0.26.

Between the modules = 0.31, P = 0.33.

Within the modules = 0.57, P = 0.28.

Figure 3

All analyses in this section are focused on the nodes, and not on the entire network.

They are run in steps and then compiled to produce the panel of Figure 3.

Calculate specialization (d'):

```
d <- specieslevel(data,index="d")
dplants <- d$`higher level`
```

Calculate betweenness centrality (BC):

```
BC <- specieslevel(data, index="betweenness")
BCplants <- BC$higher
```

Calculate normalized degree (nk):

```
ND <-ND(data, normalised=T)
NDplants <- ND$higher
```

Compare centrality metrics by pollination syndrome.

Import the data:

```
plants <- read.xls("data/plants.xlsx", h=T) # reading compiled spreadsheet with species & metrics class
```

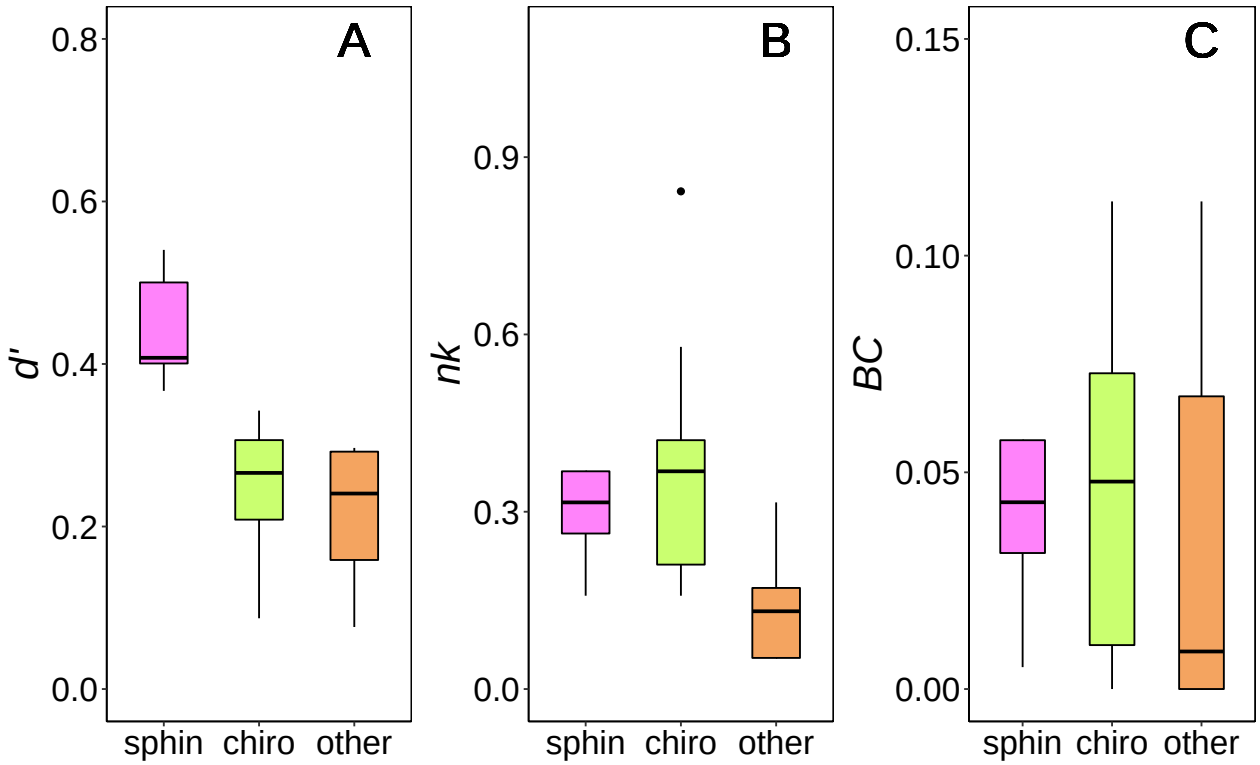
Change the reference level for the GLMs:

```
ord <-  ordered(plants$Guild, levels = c("sphin", "chiro", "other"))
```

Plot Figure 3:

```
theme_set(theme_gray(base_size = 24))

pd <- ggplot(plants, aes(x=ord, y=d, fill=Guild)) +
  ylab("d'")+ xlab("")+ ylim(0, 0.8) +
  scale_fill_manual(values=c("darkolivegreen1", "sandybrown", "orchid1"))+
  geom_boxplot(width=0.5, color="black") +
  theme_classic() +
  theme(panel.border = element_rect(colour = "black", fill=NA, size=.5) ,
        axis.title.y = element_text(color="black", face ="italic", size =23),
        axis.text= element_text(color="black", size=19),
        legend.position = "none") +
   geom_text(x="other", y=0.8, label="A", size = 10)

pnk <- ggplot(plants, aes(x=ord, y=nk, fill=Guild)) +
  ylab("nk")+ xlab("")+ ylim(0, 1.1) +
  scale_fill_manual(values=c("darkolivegreen1", "sandybrown", "orchid1"))+
  geom_boxplot(width=0.5, color="black") +
  theme_classic() +
  theme(panel.border = element_rect(colour = "black", fill=NA, size=.5) ,
        axis.title.y = element_text(color="black", face ="italic", size =23),
        axis.text= element_text(color="black", size=19),
        legend.position = "none") +
  geom_text(x="other", y=1.1, label="B", size = 10)

pBC <- ggplot(plants, aes(x=ord, y=bc, fill=Guild)) +
  ylab("BC")+ xlab("")+ ylim(0, 0.15) +
  scale_fill_manual(values=c("darkolivegreen1", "sandybrown", "orchid1"))+
  geom_boxplot(width=0.5, color="black") +
  theme_classic() +
  theme(panel.border = element_rect(colour = "black", fill=NA, size=.5) ,
```

```
        axis.title.y = element_text(color="black", face ="italic", size =23),
        axis.text= element_text(color="black", size=19),
        legend.position = "none") +
  geom_text(x="other", y=0.15, label="C", size = 10)

grid.arrange(pd, pnk, pBC,
             ncol=3,
             vp=viewport(width=1.0, height=0.9))
```



GLMs to compare centrality by pollination syndrome

Prepare the data:

```
#table(plants$Guild)
#plants$Guild <- relevel(plants$Guild, ref="chiro") #changing reference level for GLMs
#plants$Guild <- relevel(plants$Guild, ref="sphin")
#plants$Guild <- relevel(plants$Guild, ref="other")
```

d':

```
glmd <- glm(plants$d ~ plants$Guild, family=quasibinomial("logit"))
summary(glmd)
```

```
##
## Call:
## glm(formula = plants$d ~ plants$Guild, family = quasibinomial("logit"))
```

```
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.39392  -0.08322   0.04888   0.15974   0.23511
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -1.1626     0.1588  -7.320  6.1e-07 ***
## plants$Guildother  -0.1409     0.2364  -0.596  0.55809
## plants$Guildsphin   0.9344     0.2421   3.860  0.00106 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasibinomial family taken to be 0.04119193)
##
##     Null deviance: 1.72876  on 21  degrees of freedom
## Residual deviance: 0.86102  on 19  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 4
```

```
glm_d <- anova(glmd, test = "Chisq")
glm_d
```

```
## Analysis of Deviance Table
##
## Model: quasibinomial, link: logit
##
## Response: plants$d
##
## Terms added sequentially (first to last)
##
##
##              Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                            21    1.72876
## plants$Guild  2  0.86774        19    0.86102 2.664e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

BC:

```
glmbc <- glm(plants$bc ~ plants$Guild, family=quasibinomial("logit"))
summary(glmbc)
```

```
##
## Call:
## glm(formula = plants$bc ~ plants$Guild, family = quasibinomial("logit"))
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.32025  -0.25659  -0.02493   0.09635   0.33539
##
```

```
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -2.9447     0.3179  -9.264 1.78e-08 ***
## plants$Guildother  -0.3615     0.5092  -0.710    0.486
## plants$Guildsphin  -0.2632     0.5764  -0.457    0.653
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasibinomial family taken to be 0.04318113)
##
##     Null deviance: 0.98081  on 21  degrees of freedom
## Residual deviance: 0.95685  on 19  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 6
```

```
glm_bc <- anova(glmbc, test = "Chisq")
glm_bc
```

```
## Analysis of Deviance Table
##
## Model: quasibinomial, link: logit
##
## Response: plants$bc
##
## Terms added sequentially (first to last)
##
##
##              Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## NULL                             21    0.98081
## plants$Guild  2 0.023963        19    0.95685   0.7577
```

nk:

```
glmnk <- glm(plants$nk ~ plants$Guild, family=quasibinomial("logit"))
summary(glmnk)
```

```
##
## Call:
## glm(formula = plants$nk ~ plants$Guild, family = quasibinomial("logit"))
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.49970  -0.27872  -0.03614   0.13659   0.94078
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -0.4643     0.2330  -1.992  0.06090 .
## plants$Guildother  -1.3664     0.4194  -3.258  0.00414 **
## plants$Guildsphin  -0.4082     0.4072  -1.003  0.32869
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for quasibinomial family taken to be 0.1158454)
##
##     Null deviance: 3.6428  on 21  degrees of freedom
## Residual deviance: 2.2518  on 19  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 4
```

```
glm_k <- anova(glmnk, test = "Chisq")
glm_k
```

```
## Analysis of Deviance Table
##
## Model: quasibinomial, link: logit
##
## Response: plants$nk
##
## Terms added sequentially (first to last)
##
##
##              Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## NULL                             21     3.6428
## plants$Guild  2   1.3911        19     2.2518 0.002469 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 4

Import the morphology data

```
morph_plants<-read.xls("data/morph_pla.xlsx", h=T)
morph_pol <- read.xls("data/morph_pol.xlsx", h=T)
```

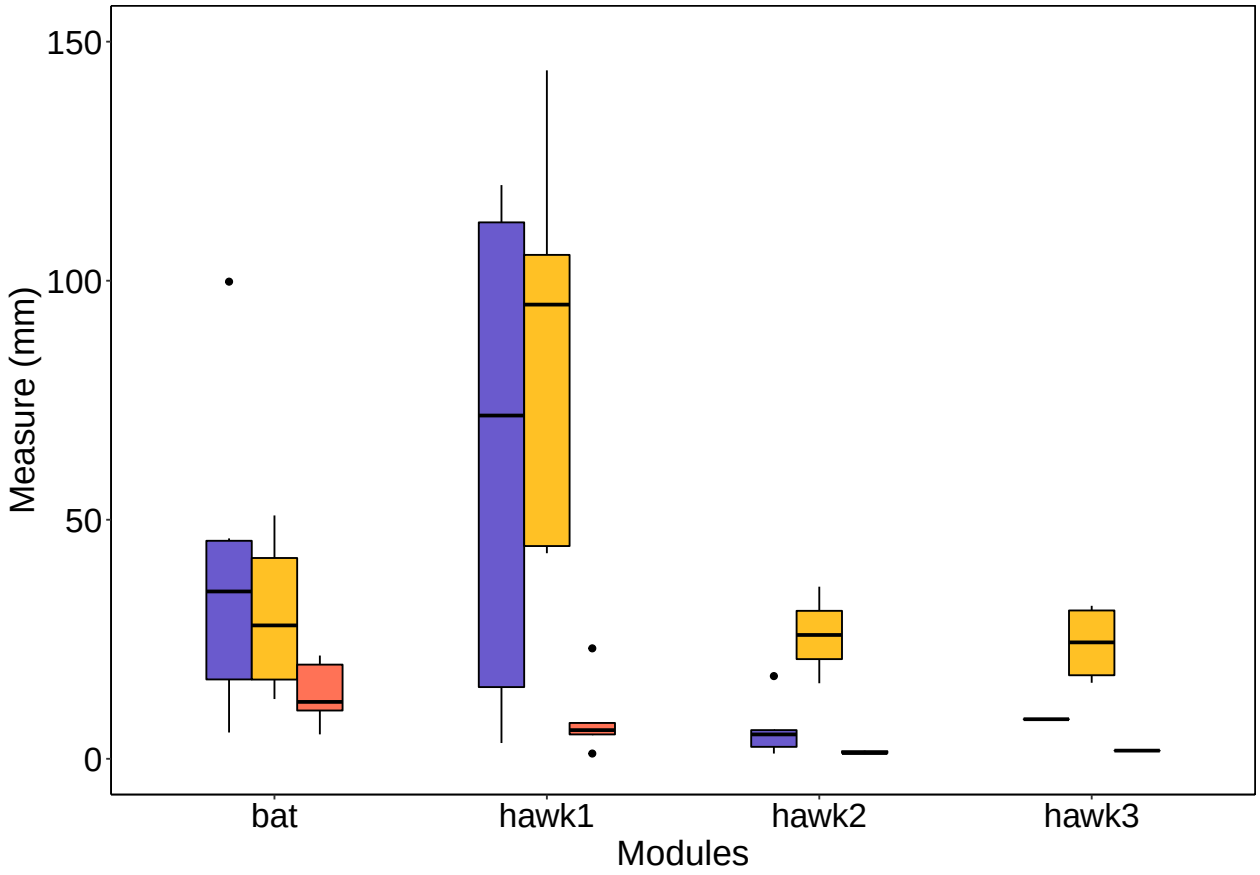
Change the reference level for the GLMs:

```
#morph_plants$module <- relevel(morph_plants$module, ref="bat")
#morph_pol$module <- relevel(morph_pol$module, ref="hawk1")
```

Plot Figure 4:

```
ggmorph<-read.xls("data/morph_graph.xlsx",h=T)

ggplot(ggmorph, aes(x=module, y=measure, fill=variable)) +
  ylab("Measure (mm)")+ xlab("Modules")+ ylim(0, 150) +
  scale_fill_manual(values=c("slateblue", "goldenrod1", "coral1"))+
  geom_boxplot(width=0.5, color="black", position = position_dodge(width=0.5)) +
  theme_classic() +
  theme(panel.border = element_rect(colour = "black", fill=NA, size=.5) ,
        axis.title.y = element_text(color="black", size =20),
```

```
        axis.title.x = element_text(color="black", size =20),
        axis.text= element_text(color="black", size=19), legend.position = "none")
```



GLMs to compare morphometry by module

Pollinator tongues:

```
glm_pol <- glm(morph_pol$length_pol~morph_pol$module, family=gaussian())
summary(glm_pol)
```

```
##
## Call:
## glm(formula = morph_pol$length_pol ~ morph_pol$module, family = gaussian())
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -40.114  -13.442   -1.283   14.301   60.886
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)               29.783     11.137   2.674  0.01733 *
## morph_pol$modulehawk1     53.331     15.177   3.514  0.00313 **
## morph_pol$modulehawk2     -3.883     22.273  -0.174  0.86392
## morph_pol$modulehawk3     -5.633     17.609  -0.320  0.75344
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 744.1431)
##
##     Null deviance: 25036  on 18  degrees of freedom
## Residual deviance: 11162  on 15  degrees of freedom
## AIC: 185.06
##
## Number of Fisher Scoring iterations: 2
```

```
anova(glm_pol, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model: gaussian, link: identity
##
## Response: morph_pol$length_pol
##
## Terms added sequentially (first to last)
##
##
##                  Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                                18      25036
## morph_pol$module  3    13874        15      11162 0.0003239 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Floral width (w) and length (l):

```
glm_pla_l <- glm(morph_plants$length_pla~morph_plants$module, family=gaussian())
glm_pla_w <- glm(morph_plants$width_pla~morph_plants$module, family=gaussian())
summary(glm_pla_l)
```

```
##
## Call:
## glm(formula = morph_plants$length_pla ~ morph_plants$module,
##     family = gaussian())
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -61.160  -14.011   -0.850    8.049   61.986
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  37.81      13.45   2.812   0.0138 *
## morph_plants$modulehawk1     26.65      20.83   1.279   0.2217
## morph_plants$modulehawk2    -31.41      20.83  -1.508   0.1538
## morph_plants$modulehawk3    -29.53      38.04  -0.777   0.4504
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for gaussian family taken to be 1265.831)
##
##     Null deviance: 26920  on 17  degrees of freedom
## Residual deviance: 17722  on 14  degrees of freedom
## AIC: 185.14
##
## Number of Fisher Scoring iterations: 2
```

```
anova(glm_pla_l, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model: gaussian, link: identity
##
## Response: morph_plants$length_pla
##
## Terms added sequentially (first to last)
##
##
##                     Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## NULL                                    17      26920
## morph_plants$module  3   9198.2        14      17722  0.06387 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(glm_pla_w)
```

```
##
## Call:
## glm(formula = morph_plants$width_pla ~ morph_plants$module, family = gaussian())
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -8.920  -2.553  -0.240   0.335  14.540
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 14.030      2.328   6.028  3.1e-05 ***
## morph_plants$modulehawk1    -5.470      3.606  -1.517   0.1515
## morph_plants$modulehawk2   -12.690      3.606  -3.519   0.0034 **
## morph_plants$modulehawk3   -12.320      6.584  -1.871   0.0823 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 37.9257)
##
##     Null deviance: 1046.76  on 17  degrees of freedom
## Residual deviance:  530.96  on 14  degrees of freedom
## AIC: 122
##
## Number of Fisher Scoring iterations: 2
```

```r
anova(glm_pla_w, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model: gaussian, link: identity
##
## Response: morph_plants$width_pla
##
## Terms added sequentially (first to last)
##
##
##                     Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## NULL                                    17    1046.76
## morph_plants$module  3    515.8        14     530.96 0.003503 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

SFigure S1

Load the interaction data for the Chao1 estimator:

```r
sampbat<- read.xls("data/sampbat.xlsx", h=T)
estimateR(sampbat, index =c("chao"))
```

```
##               [,1]
## S.obs    13.000000
## S.chao1  14.000000
## se.chao1  2.283481
## S.ACE    14.877551
## se.ACE    1.415216
```

```r
str(sampbat)
```

```
## 'data.frame':    1 obs. of  13 variables:
##  $ Pilo_.goun: int 40
##  $ Bauh_chei : int 24
##  $ Ceib_glaz : int 15
##  $ Ench_spec : int 40
##  $ Ipom_marc : int 26
##  $ Heli_baru : int 50
##  $ Crot_sp   : int 4
##  $ Pseu_marg : int 49
##  $ Indet1    : int 1
##  $ Liri_sp   : int 32
##  $ Pipt_stip : int 3
##  $ Pilo_chry : int 52
##  $ Comb_sp   : int 1
```

```r
samphawk <- read.xls("data/samphawk.xlsx", h=T)
estimateR(samphawk, index =c("chao"))
```

```
##                [,1]
## S.obs    22.0000000
## S.chao1  22.2000000
## se.chao1  0.6195203
## S.ACE    23.3031903
## se.ACE    2.4385542
```

```
str(samphawk)
```

```
## 'data.frame':    1 obs. of  22 variables:
##  $ Pilo_.goun: int 83
##  $ Bauh_chei : int 39
##  $ Ceib_glaz : int 10
##  $ Ench_spec : int 30
##  $ Ipom_marc : int 10
##  $ Guet_ange : int 68
##  $ Toco_form : int 18
##  $ Ambu_cear : int 48
##  $ Heli_baru : int 10
##  $ Crot_sp   : int 27
##  $ Cere_jama : int 18
##  $ Indet1    : int 4
##  $ Liri_sp   : int 2
##  $ Pipt_stip : int 5
##  $ Aspi_pyri : int 38
##  $ Comb_sp   : int 8
##  $ Anad_colu : int 2
##  $ Alib_sp   : int 1
##  $ Allo_sp   : int 3
##  $ Caes_sp   : int 2
##  $ Schi_bras : int 2
##  $ Indet2    : int 1
```

Load the interaction data for drawing the rarefaction curve:

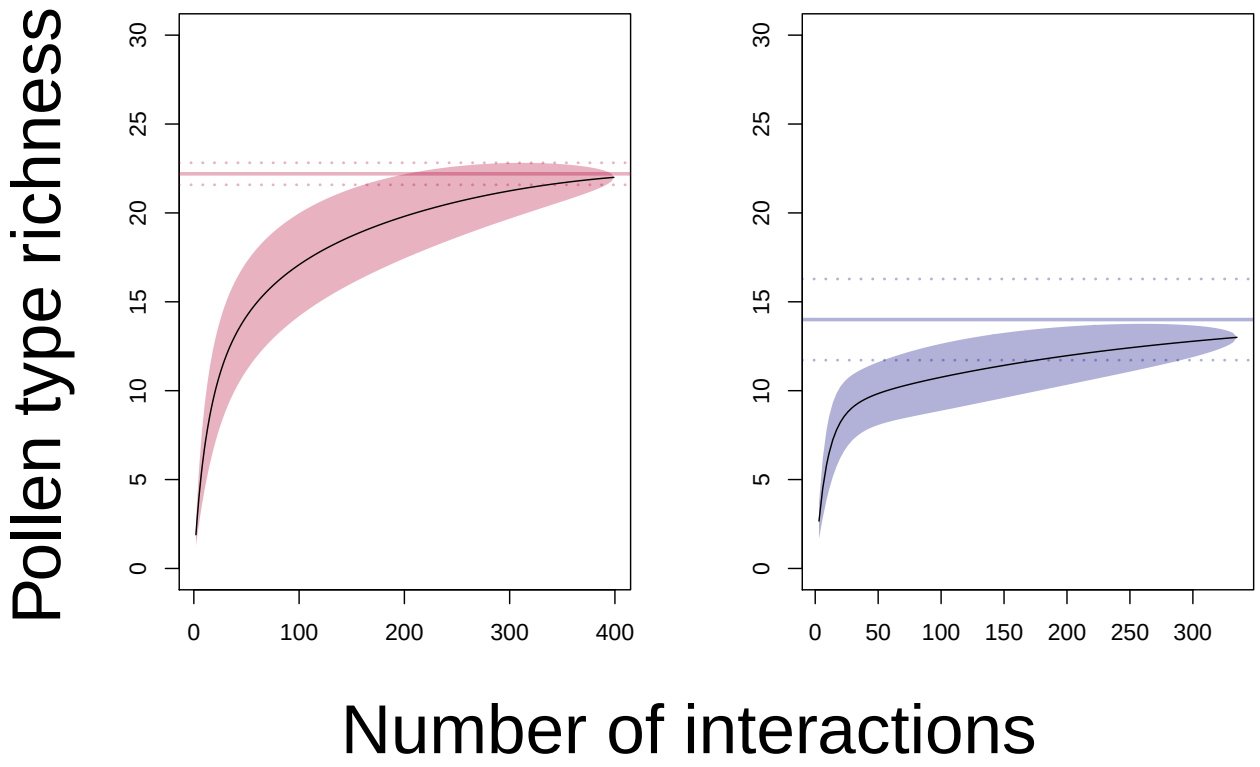
```
sampling_bats <- read.xls("data/sampling_bats.xlsx", h=T)
curve_bat<- specaccum(sampling_bats, method="rarefaction")
```

```
sampling_hawkmoths <- read.xls("data/sampling_hawkmoths.xlsx", h=T)
curve_hawk<- specaccum(sampling_hawkmoths, method="rarefaction")
```

Plot Figure S1:

```
par(mfrow=c(1,2), oma=c(5,5,0,0))

plot(curve_hawk, ci.type = "poly", xvar = "individuals", ci.lty=0, ylab = NA,
     xlab = NA,
     ci.col=rgb(0.7, 0, 0.2, 0.3), ylim=c(0,30))
abline(h=22.2, lty=1, col=rgb(0.7, 0, 0.2, 0.3), lwd=2.5)
abline(h=(22.2+0.6195203), lty=3, col=rgb(0.7, 0, 0.2, 0.3), lwd=2)
```

```
abline(h=(22.2-0.6195203), lty=3, col=rgb(0.7, 0, 0.2, 0.3), lwd=2)

plot(curve_bat, ci.type = "poly", xvar = "individuals", ci.lty=0,
     ci.col=rgb(0, 0, 0.5, 0.3), ylim=c(0,30), ylab=NA, xlab=NA)
abline(h=14, lty=1, col=rgb(0, 0, 0.5, 0.3), lwd=2.5)
abline(h=(14-2.283481), lty=3, col=rgb(0, 0, 0.5, 0.3), lwd=2)
abline(h=(14+2.283481), lty=3, col=rgb(0, 0, 0.5, 0.3), lwd=2)

mtext("Number of interactions", side = 1, cex = 3, outer = T)
mtext("Pollen type richness", side = 2, cex = 3, outer = T)
```




```
par(mfrow=c(1,1))
```