Supplemental Information for Bangal et al: Flock species richness influences node importance and modularity in mixed-species flock networks

script by Daizaburo Shizuka

updated 06/04/21

Contents

This is the supplemental materials for Bangal, Sridhar, Shizuka, Vander Meiden & Shanker: Flock-species richness influences node importance and modularity in mixed-species flock networks. This document is intended to fully replicate the analyses and figures contained in the publication.

Step 1: Create a global dataset

Required Packages:

```
library(igraph)
library(EcoSimR)
library(tidyverse)
library(assortnet)
```

store default graphics parameters

```
def.par <- par(no.readonly = TRUE)
knitr::opts_chunk$set(tidy.opts=list(width.cutoff=50), tidy=TRUE)
```

First, we will load all of the data and then create a list of all species that are seen at least once in the dataset.

```
dat10 = read.csv("India-Anshi-2010.csv")
dat11 = read.csv("India-Anshi-2011.csv")
dat17 = read.csv("India-Anshi-2017.csv")

all.ids = unique(c(as.character(dat10$species), as.character(dat11$species),
    as.character(dat17$species)))
all.ids
```

```
##  [1] "ad"          "bbfly"       "bcrt"        "bd"          "bhcs"
##  [6] "bho"         "blbird"      "blbul"       "bltit"       "bnm"
## [11] "bwfs"        "cmflmbk"     "ctb"         "ctstar"      "dfb"
## [16] "fbb"         "forwag"      "gfc"         "ghb"         "glw"
## [21] "gori"        "grflmbk"     "grtd"        "hsw"         "iora"
## [26] "lblw"        "lwdshk"      "lynpe"       "malprk"      "mgh"
## [31] "mph"         "mwt"         "oht"         "parfly"      "pic"
## [36] "quak"        "rtb"         "rtf"         "scim"        "scmin"
## [41] "spbab"       "sphun"       "trog"        "verfly"      "vfn"
## [46] "vhp"         "wbbf"        "wbt"         "wcb"         "wclw"
## [51] "wheye"       "ybb"         "bbcuckoo"    "blwdpk"      "junglefowl"
## [56] "magrob"      "rwb"         "tickelfly"   "blyrw"       "bno"
## [61] "hillmy"      "jbab"        "php"         "smmin"
```

Now we will use a match function to fill in the global matrix with ALL POSSIBLE birds as rows. This will create a list of three matrices (one for each year), all with same number of rows but with different numbers of columns (corresponding to how many flocks seen in a year).

```
m.list = lapply(list(dat10, dat11, dat17), function(x) {
    m = x[match(all.ids, x$species), -1]
    rownames(m) = all.ids
    m
})
```

Now we can use `do.call()` to combine this into a single matrix and convert NAs to 0s.

We eliminate low-occurrence species and also drop 2-spp flocks that had those species in them (see Methods).

```
big.m = do.call("cbind", m.list)
big.m[is.na(big.m)] = 0
#'big.m' is the individual-by-group matrix for the entire dataset
big.m = as.matrix(big.m)

# dropping species that occur in less than 1% (6 or
# less than 6) flocks from the 'big.m'
big.m = big.m[-which(rowSums(big.m) < 7), ]

# drop the flocks which now only have one species
big.m = big.m[, -which(colSums(big.m) == 1)]
```

Step 2: Construct networks using custom function

I have made two functions:

(1) `construct_raw_net()` makes the raw co-occurrence network with association indices being simply the number of times pair of species seen together divided by the number of flocks considered in the dataset

(2) `filter_by_sim()` makes a network whose edges are filtered by comparing the raw number of co-occurrences to the expected number of co-occurrences based on matrix permutations. Here, we use "sim5" method from the EcosimR package. This method keeps column totals (flock sizes) constant, and species are randomly drawn based on their relative frequency in the dataset (rowsums/sum of matrix). Z-scores are calculated as $\frac{O-\mu}{\sigma}$ where O = observed co-occurrence, μ=mean of expected co-occurrences, and σ=sd of expected co-occurrences. Default threshold for Z-score ratained in the network is 1.96, but this can be set in the function.

2.1 Create raw networks

Here's the function to make raw networks.

```r
construct_raw_net = function(dat, set.min = 1, set.max = 2) {
    # filter data to flock sizes as specified by
    # set.min and set.max
    emp.dat = dat[, which(colSums(dat) <= set.max &
        colSums(dat) >= set.min)]

    # remove species that aren't seen in this reduced
    # dataset
    emp.dat = emp.dat[which(rowSums(emp.dat) > 0),
        ]

    # generate raw co-occurrence matrix by multiplying
    # by transpose.
    emp.mat = emp.dat %*% t(emp.dat)

    # set diagonal to 0
    diag(emp.mat) = 0

    # divide by number of flocks in dataset
    ai.raw = emp.mat/ncol(emp.dat)

    # generate network
    g.raw = graph_from_adjacency_matrix(ai.raw, "undirected",
        weighted = T, diag = F)
    g.raw  #output
}
```

We can use this function to create ‘raw networks’ from all observations (`gall.raw`), flocks of 1-2 spp (`g2.raw`), 3-5spp (`g5.raw`), 6-10spp (`g10.raw`), and 11-22spp (`g22.raw`).

```r
gall.raw = construct_raw_net(big.m, set.min = 1, set.max = 22)

g2.raw = construct_raw_net(big.m, set.min = 1, set.max = 2)

g5.raw = construct_raw_net(big.m, set.min = 3, set.max = 5)

g10.raw = construct_raw_net(big.m, set.min = 6, set.max = 10)

g22.raw = construct_raw_net(big.m, set.min = 11, set.max = 22)
```

2.1 Create raw networks

Here’s the function to make filtered networks using sim5 as the null model:

```r
filter_by_sim = function(dat, set.min = 1, set.max = 2,
    iterations = 1000, z.threshold = 1.96) {
    times = iterations
    emp.dat = dat[, which(colSums(dat) <= set.max &
        colSums(dat) >= set.min)]
    emp.dat = emp.dat[which(rowSums(emp.dat) > 0),
        ]
    emp.mat = emp.dat %*% t(emp.dat)
    rand.list = lapply(1:times, function(x) sim5(emp.dat))
    rand.mats = lapply(rand.list, function(x) {
        a = x %*% t(x)
        diag(a) = 0
        a
    })
    rand.array = simplify2array(rand.mats)
    means = apply(rand.array, c(1, 2), mean)
    sds = apply(rand.array, c(1, 2), sd)
    ai = (emp.mat - means)/sds
    ai[which(ai < z.threshold)] = 0
    ai[which(is.nan(ai))] = 0
    ai[which(ai > 0)] = 1
    g = graph_from_adjacency_matrix(ai, "undirected",
        diag = F)
    g
}
```

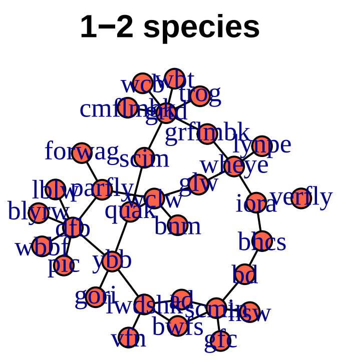
Demo the filtered networks:

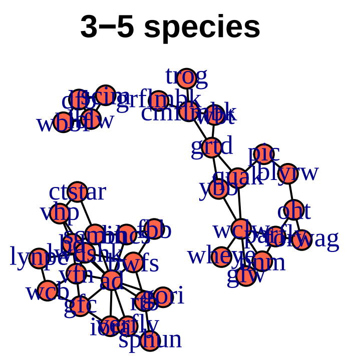
Create 4 networks using data from 2 species flocks, up to 5 species flocks, up to 10 species flocks, and up to 22 species flocks. Note that I am using a Z-score of 1 as the threshold for keeping/dropping edges. I just picked this because it seemed to generate networks that somewhat resemble what Priti had sent earlier, but this is flexible. You should think of what kind of threshold you want to set.

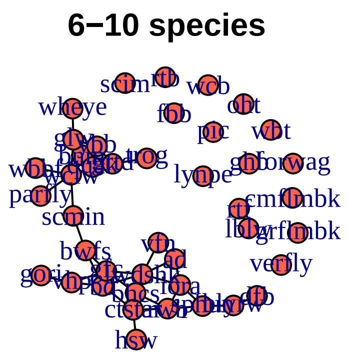
Priti: I've modified the chunck below to change the z value and also to create networks of non-cumulatively increasing flock sizes.

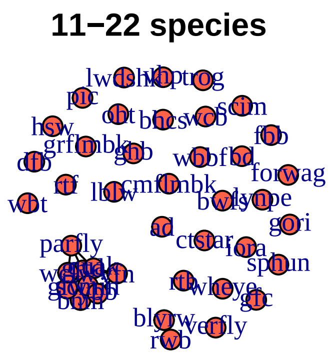
```
g2 = filter_by_sim(big.m, set.min = 1, set.max = 2,
    iterations = 1000, z.threshold = 1.96)

g5 = filter_by_sim(big.m, set.min = 3, set.max = 5,
    iterations = 1000, z.threshold = 1.96)

g10 = filter_by_sim(big.m, set.min = 6, set.max = 10,
    iterations = 1000, z.threshold = 1.96)

g22 = filter_by_sim(big.m, set.min = 11, set.max = 22,
    iterations = 1000, z.threshold = 1.96)

gall = filter_by_sim(big.m, set.min = 1, set.max = 22,
    iterations = 1000, z.threshold = 1.96)
```

```
layout(matrix(c(1, 2, 3, 4), nrow = 2, byrow = T))
par(mar = c(2, 2, 2, 2))
plot(g2, vertex.color = "tomato", edge.color = "black",
    main = "1-2 species")
plot(g5, vertex.color = "tomato", edge.color = "black",
    main = "3-5 species")
plot(g10, vertex.color = "tomato", edge.color = "black",
    main = "6-10 species")
plot(g22, vertex.color = "tomato", edge.color = "black",
    main = "11-22 species")
```









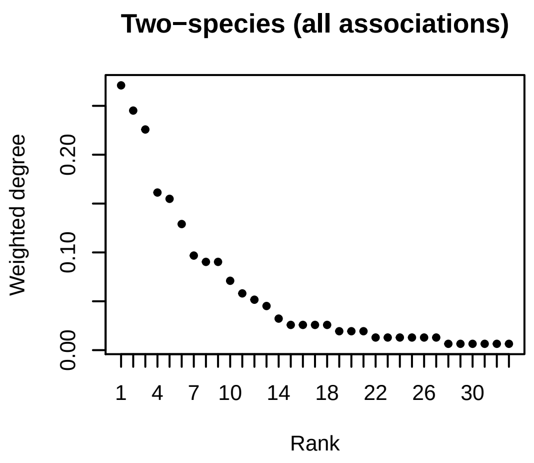
```r
par(def.par)
```

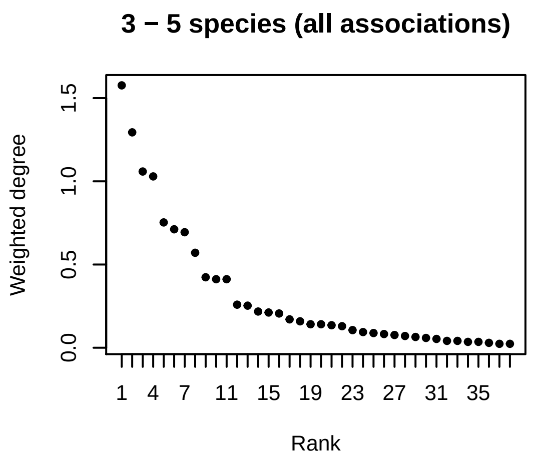
Step 3: Calculate centrality of species in each network (Table 1)

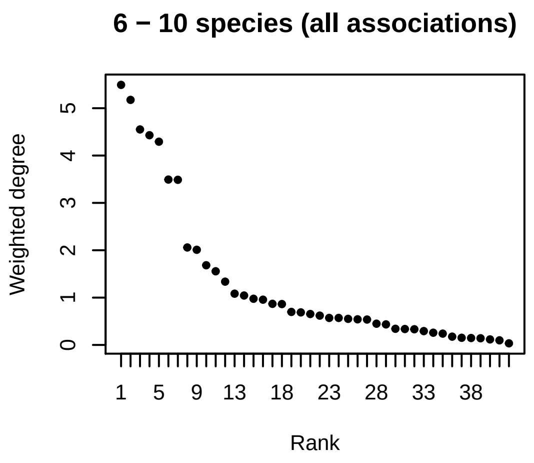
3.1 Visualize the distribution of weighted degree (a.k.a. node strength)

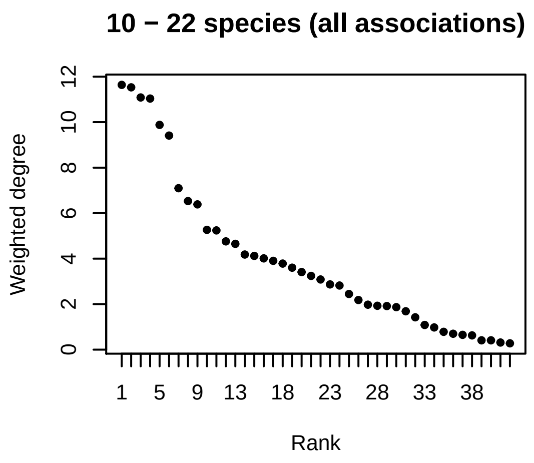
Here we plot the rank weighted degree plots for all the above networks (Supplemental Figures S1-4)

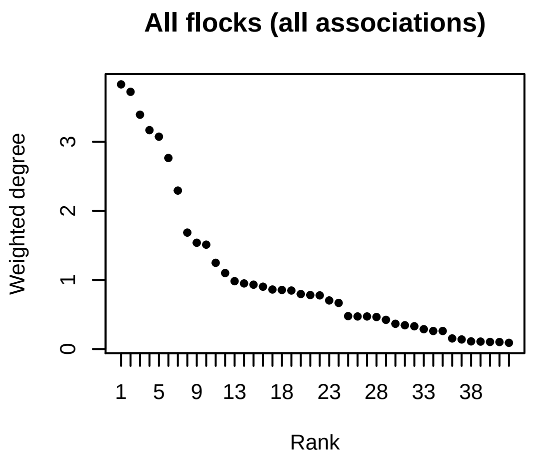
```r
layout(matrix(c(1, 2, 3, 4, 5, 6), nrow = 3, byrow = T))
plot(sort(strength(g2.raw), decreasing = TRUE), xaxp = c(1,
    length(sort(strength(g2.raw))), (length(sort(strength(g2.raw))) -
        1)), xlab = "Rank", ylab = "Weighted degree",
    pch = 20, main = "Two-species (all associations)")

plot(sort(strength(g5.raw), decreasing = TRUE), xaxp = c(1,
    length(sort(strength(g5.raw))), (length(sort(strength(g5.raw))) -
        1)), xlab = "Rank", ylab = "Weighted degree",
    pch = 20, main = "3 - 5 species (all associations)")

plot(sort(strength(g10.raw), decreasing = TRUE), xaxp = c(1,
    length(sort(strength(g10.raw))), (length(sort(strength(g10.raw))) -
        1)), xlab = "Rank", ylab = "Weighted degree",
    pch = 20, main = "6 - 10 species (all associations)")

plot(sort(strength(g22.raw), decreasing = TRUE), xaxp = c(1,
    length(sort(strength(g22.raw))), (length(sort(strength(g22.raw))) -
        1)), xlab = "Rank", ylab = "Weighted degree",
    pch = 20, main = "10 - 22 species (all associations)")

plot(sort(strength(gall.raw), decreasing = TRUE), xaxp = c(1,
    length(sort(strength(gall.raw))), (length(sort(strength(gall.raw))) -
        1)), xlab = "Rank", ylab = "Weighted degree",
    pch = 20, main = "All flocks (all associations)")

par(def.par)
```











Let’s store weighted degree info for each species across flock size categories.

```r
wd2 <- as.data.frame(sort(strength(g2.raw), decreasing = TRUE))
wd2 <- rownames_to_column(wd2)
colnames(wd2) <- c("species", "WD_2")
wd2$wd2.rank = rank(-wd2$WD_2)

wd5 <- as.data.frame(sort(strength(g5.raw), decreasing = TRUE))
wd5 <- rownames_to_column(wd5)
colnames(wd5) <- c("species", "WD_5")
wd5$wd5.rank = rank(-wd5$WD_5)

wd10 <- as.data.frame(sort(strength(g10.raw), decreasing = TRUE))
wd10 <- rownames_to_column(wd10)
colnames(wd10) <- c("species", "WD_10")
wd10$wd10.rank = rank(-wd10$WD_10)

wd22 <- as.data.frame(sort(strength(g22.raw), decreasing = TRUE))
wd22 <- rownames_to_column(wd22)
colnames(wd22) <- c("species", "WD_22")
wd22$wd22.rank = rank(-wd22$WD_22)

wdall <- as.data.frame(sort(strength(gall.raw), decreasing = TRUE))
wdall <- rownames_to_column(wdall)
colnames(wdall) <- c("species", "WD_all")
wdall$wdall.rank = rank(-wdall$WD_all)

raw_WD <- Reduce(function(x, y) merge(x = x, y = y,
    by = "species", all.x = TRUE, all.y = TRUE), list(wd2,
    wd5, wd10, wd22, wdall))

head(raw_WD[order(raw_WD$wd2.rank), ])
```

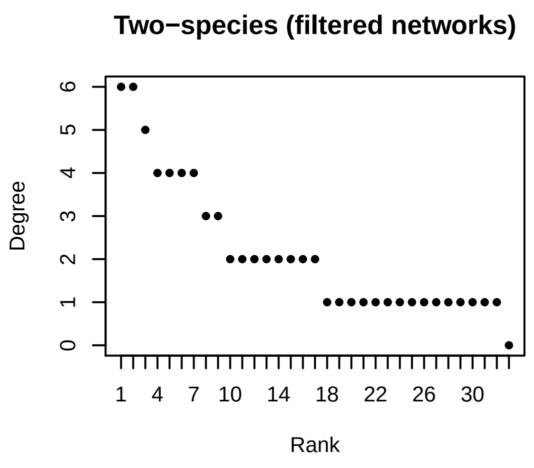
```
##    species      WD_2 wd2.rank      WD_5 wd5.rank     WD_10 wd10.rank     WD_22
## 31   scmin 0.2709677        1 0.6941176        7 3.4878049         7  9.879518
## 40    wclw 0.2451613        2 1.5764706        1 5.4926829         1 11.036145
## 26    quak 0.2258065        3 1.2941176        2 5.1756098         2 11.638554
## 1       ad 0.1612903        4 0.7117647        6 1.6829268        10  5.240964
## 41   wheye 0.1548387        5 0.4117647       11 2.0097561         9  5.265060
## 9      dfb 0.1290323        6 0.4235294        9 0.9560976        16  3.409639
##    wd22.rank    WD_all wdall.rank
## 31         5 2.7650897          6
## 40         4 3.8303426          1
## 26         1 3.7226754          2
## 1         11 1.5106036         10
## 41        10 1.5383361          9
## 9         20 0.9314845         15
```

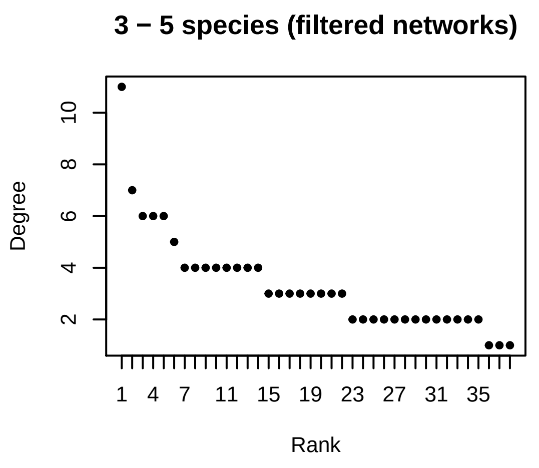
```r
write.csv(raw_WD, "Raw_net_WD.csv")
```

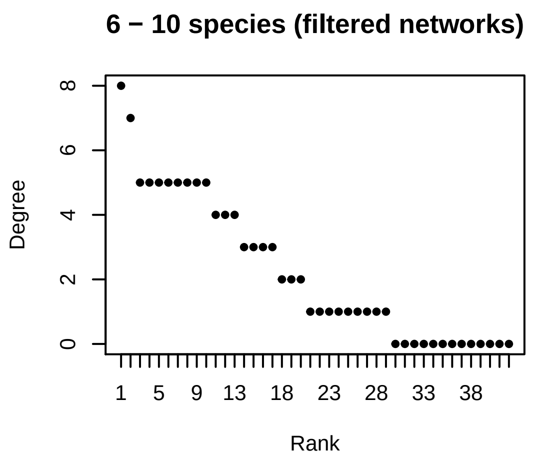
We can do the same with the filtered networks

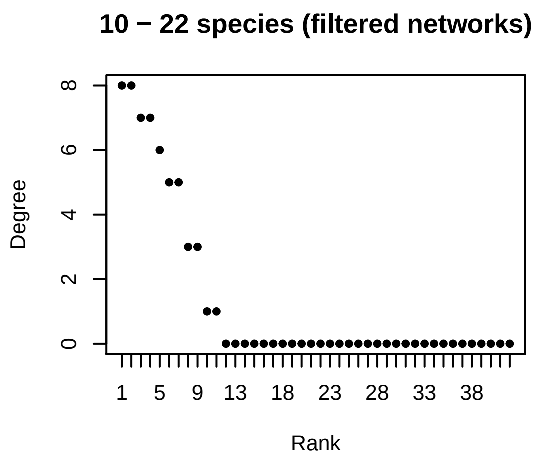
```r
layout(matrix(c(1, 2, 3, 4, 5, 6), nrow = 3, byrow = T))
plot(sort(degree(g2), decreasing = TRUE), xaxp = c(1,
    length(sort(strength(g2))), (length(sort(strength(g2))) -
        1)), xlab = "Rank", ylab = "Degree", pch = 20,
    main = "Two-species (filtered networks)")
```

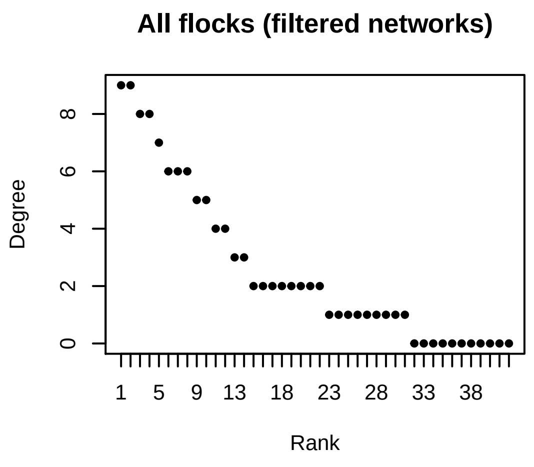
```r
plot(sort(degree(g5), decreasing = TRUE), xaxp = c(1,
    length(sort(strength(g5))), (length(sort(strength(g5))) -
        1)), xlab = "Rank", ylab = "Degree", pch = 20,
    main = "3 - 5 species (filtered networks)")

plot(sort(degree(g10), decreasing = TRUE), xaxp = c(1,
    length(sort(strength(g10))), (length(sort(strength(g10))) -
        1)), xlab = "Rank", ylab = "Degree", pch = 20,
    main = "6 - 10 species (filtered networks)")

plot(sort(degree(g22), decreasing = TRUE), xaxp = c(1,
    length(sort(strength(g22))), (length(sort(strength(g22))) -
        1)), xlab = "Rank", ylab = "Degree", pch = 20,
    main = "10 - 22 species (filtered networks)")

plot(sort(degree(gall), decreasing = TRUE), xaxp = c(1,
    length(sort(strength(gall))), (length(sort(strength(gall))) -
        1)), xlab = "Rank", ylab = "Degree", pch = 20,
    main = "All flocks (filtered networks)")

par(def.par)
```











Let’s store weighted degree info for each species across flock size categories.

```
wd2 <- as.data.frame(sort(degree(g2), decreasing = TRUE))
wd2 <- rownames_to_column(wd2)
colnames(wd2) <- c("species", "WD_2")
wd2$wd2.rank = rank(-wd2$WD_2)

wd5 <- as.data.frame(sort(degree(g5), decreasing = TRUE))
```

```
wd5 <- rownames_to_column(wd5)
colnames(wd5) <- c("species", "WD_5")
wd5$wd5.rank = rank(-wd5$WD_5)

wd10 <- as.data.frame(sort(degree(g10), decreasing = TRUE))
wd10 <- rownames_to_column(wd10)
colnames(wd10) <- c("species", "WD_10")
wd10$wd10.rank = rank(-wd10$WD_10)

wd22 <- as.data.frame(sort(degree(g22), decreasing = TRUE))
wd22 <- rownames_to_column(wd22)
colnames(wd22) <- c("species", "WD_22")
wd22$wd22.rank = rank(-wd22$WD_22)

wdall <- as.data.frame(sort(degree(gall), decreasing = TRUE))
wdall <- rownames_to_column(wdall)
colnames(wdall) <- c("species", "WD_all")
wdall$wdall.rank = rank(-wdall$WD_all)

filtered_WD <- Reduce(function(x, y) merge(x = x, y = y,
    by = "species", all.x = TRUE, all.y = TRUE), list(wd2,
    wd5, wd10, wd22, wdall))

head(filtered_WD[order(filtered_WD$wd2.rank), ])
```

```
##    species WD_2 wd2.rank WD_5 wd5.rank WD_10 wd10.rank WD_22 wd22.rank WD_all
## 9      dfb    6      1.5    3     18.5     1      25.0     0      27.0      2
## 17    grtd    6      1.5    4     10.5     5       6.5     8       1.5      9
## 31   scmin    5      3.0    6      4.0     2      19.0     6       5.0      8
## 21  lwdshk    4      5.5    7      2.0     7       2.0     0      27.0      4
## 40    wclw    4      5.5    6      4.0     8       1.0     7       3.5      8
## 41   wheye    4      5.5    1     37.0     1      25.0     0      27.0      1
##    wdall.rank
## 9        18.5
## 17        1.5
## 31        3.5
## 21       11.5
## 40        3.5
## 41       27.0
```

```
write.csv(filtered_WD, "filtered_net_WD.csv")
```

Step 3: Community detection and modularity of networks

First, here is the filtered 2-species flock network with nodes colored by community. We used to filtered network, and applied the Louvain method community detection. The community detection method uses the edge weights by default.

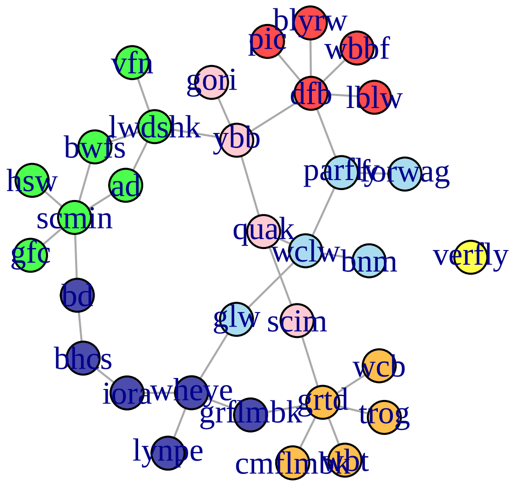
```
com.g2 = cluster_louvain(g2)  #run community detection
com.g2
```

```
## IGRAPH clustering multi level, groups: 7, mod: 0.6
## + groups:
##   $`1`
##   [1] "dfb"   "lblw"  "pic"   "wbbf"  "blyrw"
```

```
##
## $`2`
## [1] "cmflmbk" "grtd"    "trog"    "wbt"     "wcb"
##
## $`3`
## [1] "verfly"
##
## $`4`
## + ... omitted several groups/vertices
```

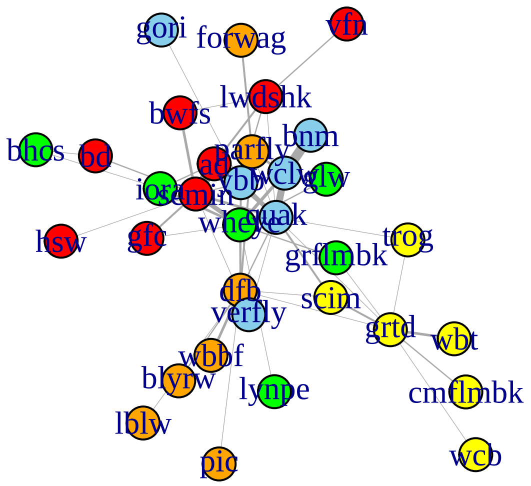
Priti:modified 2 chunk below to retain vertex labels. Also changed the vertex color below to adjust tansparency so that node labels are visible.

```
g2.membership = membership(com.g2)  #get community membership from g2
color.pal = c("red", "orange", "yellow", "green", "skyblue",
    "darkblue", "lightpink")  #set up color palette
V(g2)$g2com = g2.membership
V(g2)$g2com.col = color.pal[g2.membership]  #assign color to g2 network
plot(g2, vertex.color = adjustcolor(V(g2)$g2com.col,
    alpha.f = 0.7), edge.width = E(g2)$weight)
```



We can do this for the raw 2-species network as well:

```
com.g2.raw = cluster_louvain(g2.raw)  #run community detection
g2.raw.membership = membership(com.g2.raw)  #get community membership from g2
color.pal = c("red", "orange", "yellow", "green", "skyblue")  #set up color palette
V(g2.raw)$g2com = g2.raw.membership
V(g2.raw)$g2com.col = color.pal[g2.raw.membership]  #assign color to g2 network
plot(g2.raw, vertex.color = V(g2.raw)$g2com.col, edge.width = E(g2.raw)$weight *
    50)
```



```r
modularity(com.g2.raw)
```

```
## [1] 0.4175442
```

Let's calculate modularity for all versions of the networks (***note: this is not included in the manuscript***). We can make this routine faster if we organize the different networks into two lists (raw networks and filtered networks).

```r
rawnets = list(g2.raw, g5.raw, g10.raw, g22.raw)
filtnets = list(g2, g5, g10, g22)
gnames = c("2 species", "3-5 species", "6-10 species",
    "11-22 species")

for (i in 1:4) {
    com = cluster_louvain(rawnets[[i]])
    print(paste(gnames[i], " (raw)", " = ", round(modularity(com),
        3), sep = ""))
}
```

```
## [1] "2 species (raw) = 0.418"
## [1] "3-5 species (raw) = 0.212"
## [1] "6-10 species (raw) = 0.075"
## [1] "11-22 species (raw) = 0.015"
```

```r
for (i in 1:4) {
    com = cluster_louvain(filtnets[[i]])
    print(paste(gnames[i], " (filtered)", " = ", round(modularity(com),
        3), sep = ""))
}
```

```
## [1] "2 species (filtered) = 0.599"
## [1] "3-5 species (filtered) = 0.594"
## [1] "6-10 species (filtered) = 0.551"
## [1] "11-22 species (filtered) = 0.071"
```

Step 4: Measuring assortment by community based on clusters identified in 2-spp flocks

One major question we ask is how the discrete clusters, or 'network communities' that we detect in 2-species flocks–i.e., species pair associations–are reflected networks built from larger, more speciose flocks. In other words, to what extent do we lose our ability to detect clustered groups of species when we focus on larger mixed-species flocks?

To ask this question, we can determine how well the network is assorted based on the community membership identified in 2-spp flocks using an assortment coefficient. We can use the community assignments of nodes in 2-spp flocks to measure the assortment coefficient. The advantages of this method compared to simply comparing modularity values across networks is that (a) even if two networks with the same nodes have the same modularity value, it does not mean the nodes are assigned the same way, and (b) assortment coefficient conveniently ranges from 0 to 1 and in many ways are easier to compare across networks.

This is a modified use of what we used in Shizuka & Farine (2016) to show how we can measure the robustness of community assignments (this pdf should be open access).

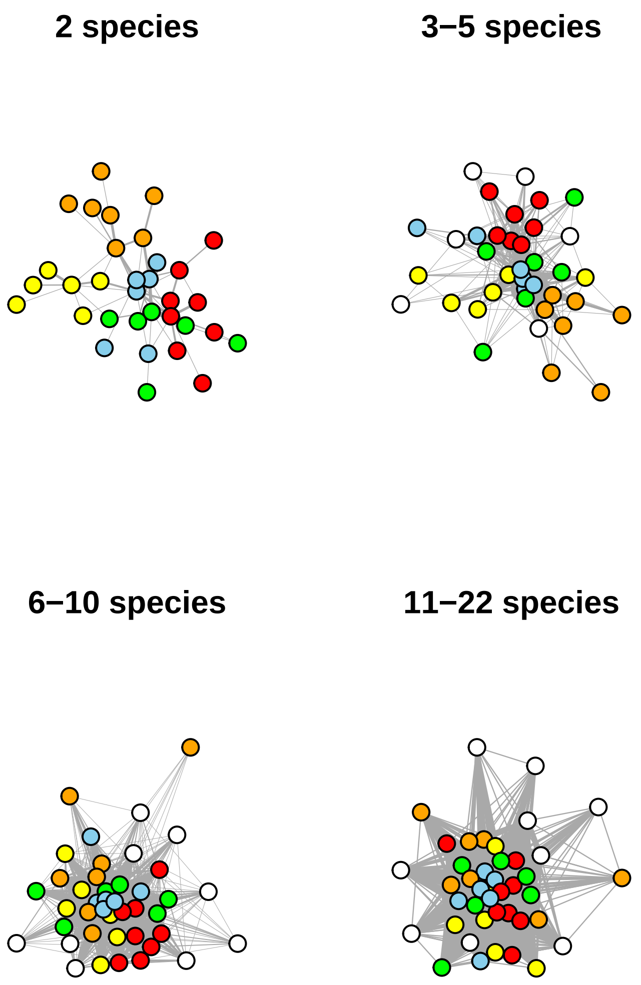
To do this, we first have to restrict the networks for all classes to the same set of nodes (i.e., nodes found in the 2-spp flock networks).

###4.1 Assign community assignments from 2-species flocks (i.e., clusters of pairwise species associations) as node attributes for other networks

As a first step, for the 3-5 spp, 6-10 spp and 11-22 spp flocks, we will create a node attribute that indicates the community assignments of those species in the 2-spp flocks. For the species that do not appear in the 2-spp flocks, we assign them as "white".

First, with the raw networks:

```r
# we use the `rawnets` and `gnames` objects from
# above

for (i in 1:4) {
    V(rawnets[[i]])$g2com = V(rawnets[[1]])$g2com[match(V(rawnets[[i]])$name,
        V(rawnets[[i]])$name)]
    V(rawnets[[i]])$g2com.col = V(rawnets[[1]])$g2com.col[match(V(rawnets[[i]])$name,
        V(rawnets[[1]])$name)]
    V(rawnets[[i]])$g2com.col[is.na(V(rawnets[[i]])$g2com.col)] = "white"
}


layout(matrix(c(1, 2, 3, 4), nrow = 2, byrow = T))
par(mar = c(2, 2, 2, 2))
for (i in 1:4) {
    plot(rawnets[[i]], vertex.color = V(rawnets[[i]])$g2com.col,
        vertex.label = "", main = gnames[i], edge.width = E(rawnets[[i]])$weight *
            50)
}
```



We'll save the version for the manuscript as a pdf.

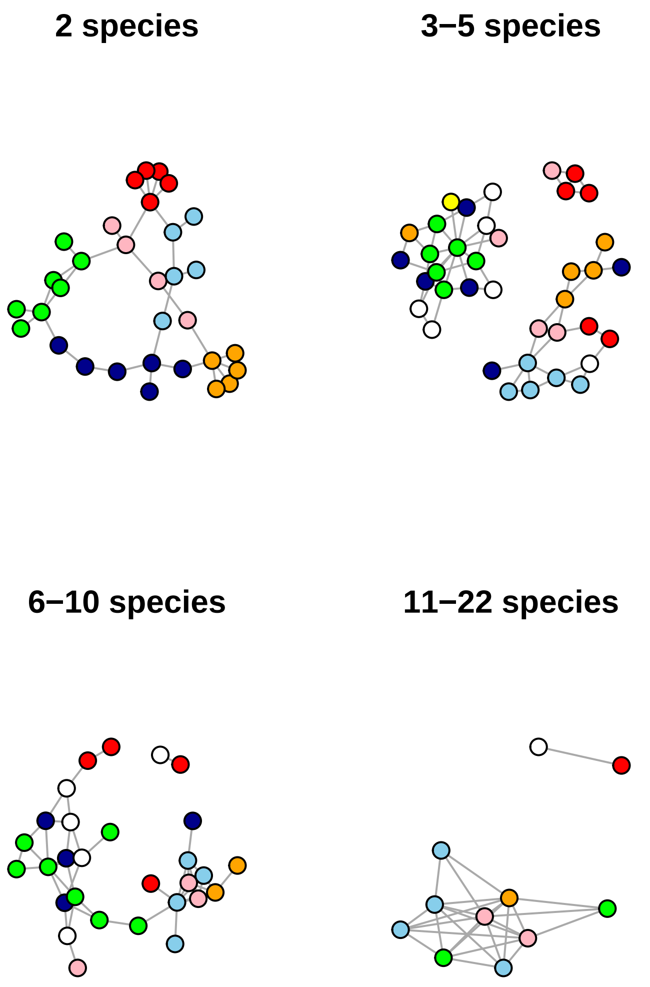
```
png("MixedSpeciesFlockNetwork_v4_DS_files/Figure1A.png",
    height = 1500, width = 400)
layout(matrix(c(1, 2, 3, 4), nrow = 4, byrow = T))
par(mar = c(2, 2, 2, 2))
for (i in 1:4) {
    plot(rawnets[[i]], vertex.color = V(rawnets[[i]])$g2com.col,
        vertex.label = "", edge.width = E(rawnets[[i]])$weight *
            50, edge.color = gray(0.1))
}
dev.off()
```

```
## pdf
##   2
```

```
par(def.par)
```

Now, with the filtered networks. Here, we remove the 'isolate' nodes–i.e., the species that are not connected any other species after edges have been filtered.

```
# we use the `filtnets` and `gnames` objects from
# above

for (i in 1:4) {
    V(filtnets[[i]])$g2com = V(filtnets[[1]])$g2com[match(V(filtnets[[i]])$name,
        V(filtnets[[i]])$name)]
    V(filtnets[[i]])$g2com.col = V(filtnets[[1]])$g2com.col[match(V(filtnets[[i]])$name,
        V(filtnets[[1]])$name)]
    V(filtnets[[i]])$g2com.col[is.na(V(filtnets[[i]])$g2com.col)] = "white"
}

layout(matrix(c(1, 2, 3, 4), nrow = 2, byrow = T))
par(mar = c(2, 2, 2, 2))
for (i in 1:4) {
    g.filt = delete.vertices(filtnets[[i]], which(degree(filtnets[[i]]) ==
        0))
    plot(g.filt, vertex.color = V(g.filt)$g2com.col,
        vertex.label = "", main = gnames[i], edge.width = E(g.filt)$weight/2)
}
```



```
png("MixedSpeciesFlockNetwork_v4_DS_files/Figure1B.png",
    height = 1500, width = 400)
layout(matrix(c(1, 2, 3, 4), nrow = 4, byrow = T))
par(mar = c(2, 2, 2, 2))
for (i in 1:4) {
    g.filt = delete.vertices(filtnets[[i]], which(degree(filtnets[[i]]) ==
        0))
    plot(g.filt, vertex.color = V(g.filt)$g2com.col,
        vertex.label = "", edge.color = gray(0.1))
}
dev.off()
```

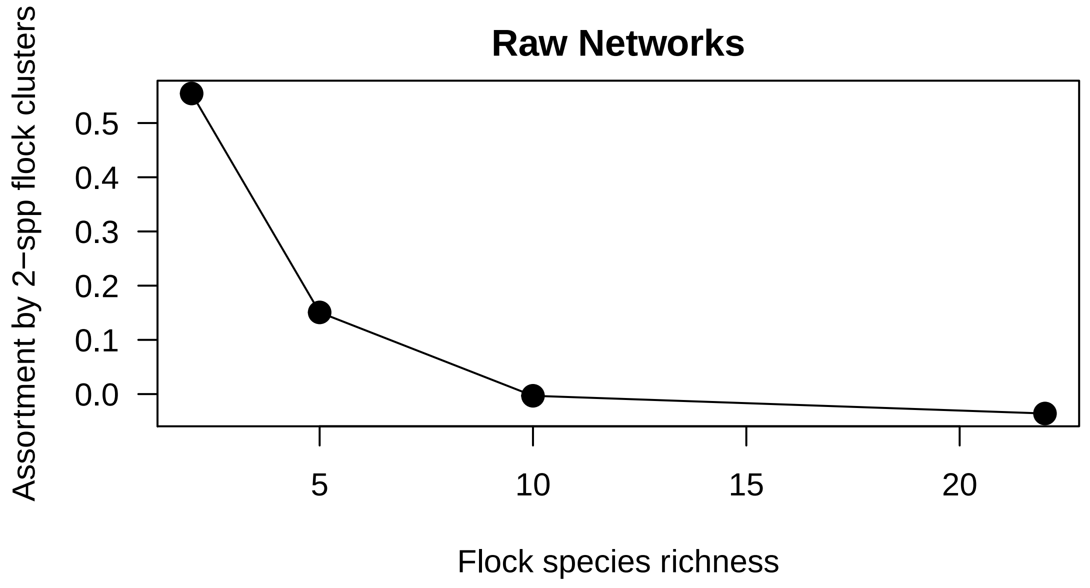
```
## pdf
##   2
```

```
par(def.par)
```

4.2 Calculate assortment of nodes based on their clusters in 2-species flocks

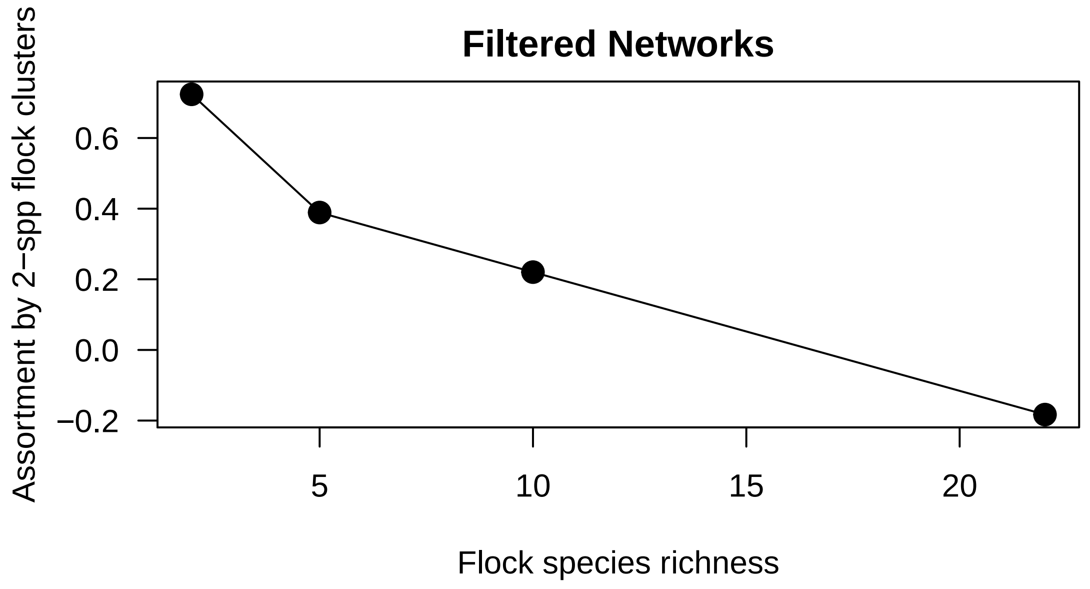
Now, we will calculate the assortment coefficient based on community assignment in 2-specues flocks. To do this, we first have to remove the “white” nodes. We then extract the adjacency matrix and use this in the `assortment.discrete` function in assortnet.

```
rs.raw = vector(length = 4)
for (i in 1:4) {
    g.raw = delete.vertices(rawnets[[i]], which(V(rawnets[[i]])$g2com.col ==
        "white"))
    adj.raw = as_adjacency_matrix(g.raw, attr = "weight",
        sparse = F)
    rs.raw[i] = assortment.discrete(adj.raw, V(g.raw)$g2com.col)$r
}

par(mar = c(4, 5, 2, 1))
plot(c(2, 5, 10, 22), rs.raw, type = "o", pch = 19,
    cex = 1.5, ylab = "Assortment by 2-spp flock clusters",
    xlab = "Flock species richness", las = 1, main = "Raw Networks")
```



Do this same thing for filtered networks

```
rs.filt = vector(length = 4)
for (i in 1:4) {
    g.filt = delete.vertices(filtnets[[i]], which(V(filtnets[[i]])$g2com.col ==
        "white"))
    adj.filt = as_adjacency_matrix(g.filt, sparse = F)
    rs.filt[i] = assortment.discrete(adj.filt, V(g.filt)$g2com.col)$r
}

par(mar = c(4, 5, 2, 1))
plot(c(2, 5, 10, 22), rs.filt, type = "o", pch = 19,
    cex = 1.5, ylab = "Assortment by 2-spp flock clusters",
    xlab = "Flock species richness", las = 1, main = "Filtered Networks")
```



4.3 Calculate 95% CI from null model of assortment

Now that we have our empirical values of assortment by the communities identified in pairwise species associations, we want to statistically test whether these patterns are different from what we expect under a null model. Here, we use a **node-label permutation** procedure to generate the null expected range of assortment values we expect. A node-label permutation swaps node attributes (in this case, the community assignments from the 2-spp flocks) across nodes while preserving the rest of the network structure (i.e., we don't move any edges) and then recalculates the assortment coefficient. We do this 1,000 times to generate a confidence interval for the assortment coefficient under this null hypothesis. This is a fairly standard method for null hypothesis testing for assortment patterns.

First, we implement the node-label permutation procedure for one network (the 2-spp raw network).

```
r2.g2.raw_rand = sapply(1:1000, function(x) {
    g.raw = delete.vertices(rawnets[[i]], which(V(rawnets[[i]])$g2com.col ==
        "white"))
    assortment.discrete(as_adjacency_matrix(g.raw,
        attr = "weight", sparse = F), sample(V(g.raw)$g2com,
        length(V(rawnets[[1]])), replace = F))$r
})
mean(r2.g2.raw_rand)
```

```
## [1] 0.02255673
```

```
r2.g2_ci = quantile(r2.g2.raw_rand, probs = c(0.025,
    0.975))
r2.g2_ci
```

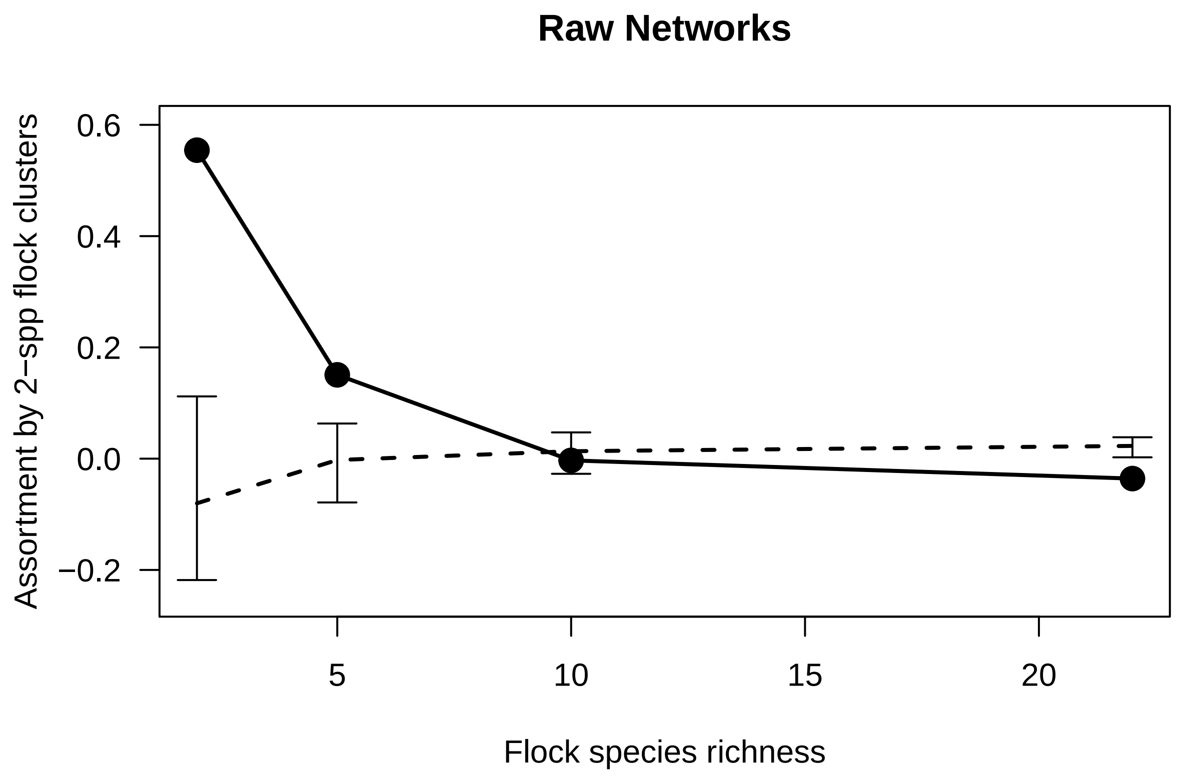
```
##        2.5%       97.5%
## 0.002322437 0.037951665
```

Now we can do it for all 4 raw networks (2 species, 3-5 species, 6-10 species, 11-22 species).

```
r.raw_rand = list()
for (i in 1:4) {
    g.raw = delete.vertices(rawnets[[i]], which(V(rawnets[[i]])$g2com.col ==
        "white"))
    r.raw_rand[[i]] = sapply(1:1000, function(x) {
        assortment.discrete(as_adjacency_matrix(g.raw,
            attr = "weight", sparse = F), sample(V(g.raw)$g2com,
            length(V(g.raw)), replace = F))$r
    })
}

ci.raw = sapply(r.raw_rand, function(x) quantile(x,
    probs = c(0.025, 0.975)))
ci.raw
```

```
##             [,1]        [,2]        [,3]        [,4]
## 2.5%  -0.2182115 -0.07871892 -0.02721867 0.002445453
## 97.5%  0.1119073  0.06316719  0.04722932 0.038479415
```

```
plot(c(2, 5, 10, 22), rs.raw, type = "o", lwd = 2,
    ylim = c(-0.25, 0.6), pch = 19, cex = 1.5, ylab = "Assortment by 2-spp flock clusters",
    xlab = "Flock species richness", las = 1, main = "Raw Networks")
arrows(2, ci.raw[1, 1], 2, ci.raw[2, 1], length = 0.1,
    angle = 90, code = 3)
arrows(5, ci.raw[1, 2], 5, ci.raw[2, 2], length = 0.1,
    angle = 90, code = 3)
arrows(10, ci.raw[1, 3], 10, ci.raw[2, 3], length = 0.1,
    angle = 90, code = 3)
arrows(22, ci.raw[1, 4], 22, ci.raw[2, 4], length = 0.1,
    angle = 90, code = 3)
points(c(2, 5, 10, 22), sapply(r.raw_rand, mean), type = "l",
    lty = 2, lwd = 2)
```

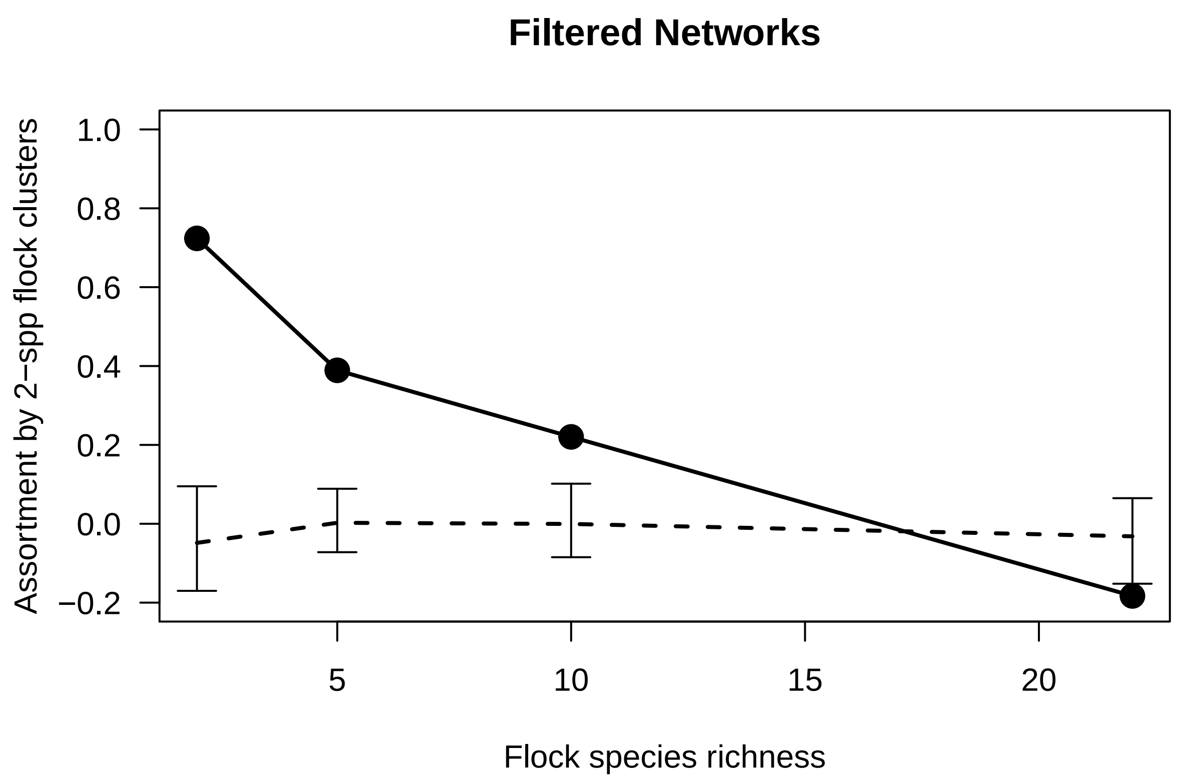


```
# test for positive assortment
p2.raw.positive = (length(which(r.raw_rand[[1]] >=
    rs.raw[[1]])) + 1)/1001
p5.raw.positive = (length(which(r.raw_rand[[2]] >=
    rs.raw[[2]])) + 1)/1001
p10.raw.positive = (length(which(r.raw_rand[[3]] >=
    rs.raw[[3]])) + 1)/1001
p22.raw.positive = (length(which(r.raw_rand[[4]] >=
    rs.raw[[4]])) + 1)/1001

# test for negative assortment
p2.raw.negative = (length(which(r.raw_rand[[1]] <=
    rs.raw[[1]])) + 1)/1001
p5.raw.negative = (length(which(r.raw_rand[[2]] <=
    rs.raw[[2]])) + 1)/1001
p10.raw.negative = (length(which(r.raw_rand[[3]] <=
    rs.raw[[3]])) + 1)/1001
p22.raw.negative = (length(which(r.raw_rand[[4]] <=
    rs.raw[[4]])) + 1)/1001

p.data.raw = data.frame(species_richness = c("2", "3-5",
    "6-10", "11-22"), network_type = rep("raw network",
    4), p_positive = round(c(p2.raw.positive, p5.raw.positive,
    p10.raw.positive, p22.raw.positive), 3), p_negative = round(c(p2.raw.negative,
    p5.raw.negative, p10.raw.negative, p22.raw.negative),
    3))

knitr::kable(p.data.raw)
```

species_richness	network_type	p_positive	p_negative
2	raw network	0.001	1.000
3-5	raw network	0.001	1.000
6-10	raw network	0.817	0.184
11-22	raw network	1.000	0.001

```r
r.filt_rand = list()
for (i in 1:4) {
    g.filt = delete.vertices(filtnets[[i]], which(V(filtnets[[i]])$g2com.col ==
        "white"))
    r.filt_rand[[i]] = sapply(1:1000, function(x) {
        assortment.discrete(as_adjacency_matrix(g.filt,
            sparse = F), sample(V(g.filt)$g2com, length(V(g.filt)),
            replace = F))$r
    })
}

ci.filt = sapply(r.filt_rand, function(x) quantile(x,
    probs = c(0.025, 0.975)))
```

```r
plot(c(2, 5, 10, 22), rs.filt, type = "o", lwd = 2,
    ylim = c(-0.2, 1), pch = 19, cex = 1.5, ylab = "Assortment by 2-spp flock clusters",
    xlab = "Flock species richness", las = 1, main = "Filtered Networks")
arrows(2, ci.filt[1, 1], 2, ci.filt[2, 1], length = 0.1,
    angle = 90, code = 3)
arrows(5, ci.filt[1, 2], 5, ci.filt[2, 2], length = 0.1,
    angle = 90, code = 3)
arrows(10, ci.filt[1, 3], 10, ci.filt[2, 3], length = 0.1,
    angle = 90, code = 3)
arrows(22, ci.filt[1, 4], 22, ci.filt[2, 4], length = 0.1,
    angle = 90, code = 3)
points(c(2, 5, 10, 22), sapply(r.filt_rand, mean),
    type = "l", lty = 2, lwd = 2)
```



```
# test for positive assortment
p2.filt.positive = (length(which(r.filt_rand[[1]] >=
    rs.filt[[1]])) + 1)/1001
p5.filt.positive = (length(which(r.filt_rand[[2]] >=
    rs.filt[[2]])) + 1)/1001
p10.filt.positive = (length(which(r.filt_rand[[3]] >=
    rs.filt[[3]])) + 1)/1001
p22.filt.positive = (length(which(r.filt_rand[[4]] >=
    rs.filt[[4]])) + 1)/1001

# test for negative assortment
p2.filt.negative = (length(which(r.filt_rand[[1]] <=
    rs.filt[[1]])) + 1)/1001
p5.filt.negative = (length(which(r.filt_rand[[2]] <=
    rs.filt[[2]])) + 1)/1001
p10.filt.negative = (length(which(r.filt_rand[[3]] <=
    rs.filt[[3]])) + 1)/1001
p22.filt.negative = (length(which(r.filt_rand[[4]] <=
    rs.filt[[4]])) + 1)/1001

p.data.filt = data.frame(species_richness = c("2",
    "3-5", "6-10", "11-22"), network_type = rep("filtered network",
    4), p_positive = round(c(p2.filt.positive, p5.filt.positive,
    p10.filt.positive, p22.filt.positive), 3), p_negative = round(c(p2.filt.negative,
    p5.filt.negative, p10.filt.negative, p22.filt.negative),
    3))

p.data.all = left_join(p.data.raw, p.data.filt, by = "species_richness")
```

```r
knitr::kable(p.data.all)
```

species_richness	network_type.x	p_positive.x	p_negative.x	network_type.y	p_positive.y	p_negative.y
2	raw network	0.001	1.000	filtered network	0.001	1.000
3-5	raw network	0.001	1.000	filtered network	0.001	1.000
6-10	raw network	0.817	0.184	filtered network	0.001	1.000
11-22	raw network	1.000	0.001	filtered network	0.998	0.003

Save pdf version for pub

```r
pdf("MixedSpeciesFlockNetwork_v4_DS_files/Figure2B.pdf",
    width = 6, height = 5)
plot(c(2, 5, 10, 22), rs.raw, type = "o", ylim = c(-0.25,
    0.75), pch = 19, cex = 1.5, ylab = "Assortment by 2-spp flock clusters",
    xlab = "Flock species richness", las = 1, main = "Raw Networks")
arrows(2, ci.raw[1, 1], 2, ci.raw[2, 1], length = 0.1,
    angle = 90, code = 3)
arrows(5, ci.raw[1, 2], 5, ci.raw[2, 2], length = 0.1,
    angle = 90, code = 3)
arrows(10, ci.raw[1, 3], 10, ci.raw[2, 3], length = 0.1,
    angle = 90, code = 3)
arrows(22, ci.raw[1, 4], 22, ci.raw[2, 4], length = 0.1,
    angle = 90, code = 3)
points(c(2, 5, 10, 22), sapply(r.raw_rand, mean), type = "l",
    lty = 2)
dev.off()
```

```
## pdf
##   2
```

```r
par(def.par)
```

Save pdf version for pub

```r
pdf("MixedSpeciesFlockNetwork_v4_DS_files/Figure2A.pdf",
    width = 6, height = 5)
plot(c(2, 5, 10, 22), rs.filt, type = "o", ylim = c(-0.25,
    0.75), pch = 19, cex = 1.5, ylab = "Assortment by 2-spp flock clusters",
    xlab = "Flock species richness", las = 1, main = "Filtered Networks")
arrows(2, ci.filt[1, 1], 2, ci.filt[2, 1], length = 0.1,
    angle = 90, code = 3)
arrows(5, ci.filt[1, 2], 5, ci.filt[2, 2], length = 0.1,
    angle = 90, code = 3)
arrows(10, ci.filt[1, 3], 10, ci.filt[2, 3], length = 0.1,
    angle = 90, code = 3)
arrows(22, ci.filt[1, 4], 22, ci.filt[2, 4], length = 0.1,
    angle = 90, code = 3)
points(c(2, 5, 10, 22), sapply(r.filt_rand, mean),
    type = "l", lty = 2, lwd = 2)
dev.off()
```

```
## pdf
##   2
```

```
par(def.par)
```