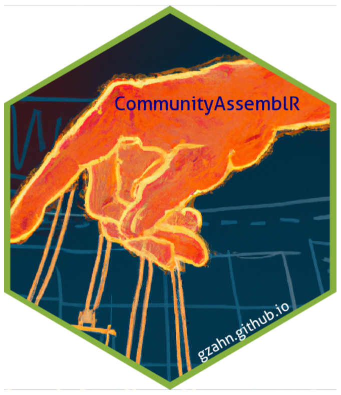


CommunityAssemblR

Package purpose

Microbiome transplantation is a rapidly growing intervention for altering the successional trajectories of macro-communities, conferring disease resistance/alleviation in plants and animals, and for encouraging disturbance/stress resistance in host organisms. It's still a developing and confusing science however, and since microbiomes are almost always transplanted into hosts that already have a resident microbial community, it is difficult to tease apart the various biotic and abiotic factors that determine whether a transplanted microbiome will persist in its new location. Additionally, modeling the process is complex, with potentially hundreds or thousands of predictors and outcomes, many of which will be dependent on each other.

This is a modular set of tools for building and altering community matrices to simulate various community assembly processes that can serve as 'ground truth' for testing transplantation model performance. These functions are intended to be modular and chainable so that a unique community can be progressively built that reflects a variety of selective pressures. This community can then be used to test the detection accuracy of predictive assembly models.

This package is under active development.

Eventual goal: to simulate ecological communities shaped by various assembly pressures for testing model detection of

- Co-occurrence relationships
- Priority effects
- Competitive exclusion
- Antagonism
- Facilitation
- Niche pre-emption
- Keystone taxa
- Abiotic filtering
- Dispersal
- etc.

Example workflow

First we will load a few packages. **CommunityAssemblR** does not currently have any dependencies and I hope to keep it that way, but the other packages here will let us inspect and visualize our communities more easily.

```
# devtools::install_github("gzahn/CommunityAssembleR")
library(CommunityAssemblR)
library(tidyverse)
library(igraph)
library(SpiecEasi)
library(kableExtra)
```

Okay, we can just get started by simulating an even community. We will include 50 taxa and 30 samples, and shoot for 3000 observations per sample, with a standard deviation of 100.

```
comm <- build_even_community(n.taxa = 50, n.samples = 30, n.reads = 3000, taxa.sd = 100)
comm[1:5,1:5] %>% kable() # look at the first 5 rows and columns
```

	taxon_1	taxon_2	taxon_3	taxon_4	taxon_5
sample_1	106	246	0	249	0
sample_2	222	29	174	24	160
sample_3	92	7	91	13	94
sample_4	53	0	75	42	48
sample_5	0	0	139	22	0

This community isn't exactly believable at this point, as it represents a stochastic assembly. However, we can mess with it a bit to make it resemble a more natural microbial community.

First up, we can apply some cooccurrence relationships (implying actual facilitation or inhibition). Here, we are doing 3 rounds of "linkage" where 15 taxa are linked to a hub network, with a fairly strong multiplier `link.scale=5`. Then, we're adding a second round of linkage using different parameters. This demonstrates how functions are stackable in order to customize theoretical assembly processes.

```
comm_linked <- link_taxa_abundances(comm,n.taxa = 15, relationship = "hub",
                                    link.scale = 5, n.links = 3) %>%
               link_taxa_abundances(n.taxa = 30,relationship = "hub",
                                    link.scale = 5, n.links = 2)
```

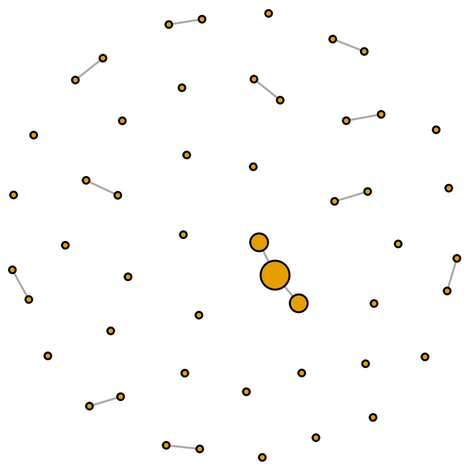
Let's take a look at network plots for both of these communities using SpiecEasi and igraph.

```
# build networks ####

# make SpiecEasi networks
se.params <- list(rep.num=20, ncores=(parallel::detectCores()-1))

# the basic even community
se_comm <- SpiecEasi::spiec.easi(data = comm,
                                 method='mb',
                                 sel.criterion = "bstars",
                                 pulsar.params=se.params)
# altered to have hub taxa relationships
se_linked <- SpiecEasi::spiec.easi(data = comm_linked,
                                   method='mb',
                                   sel.criterion = "bstars",
                                   pulsar.params=se.params)

# convert to igraph objects
comm_igraph <- adj2igraph(getRefit(se_comm))
```

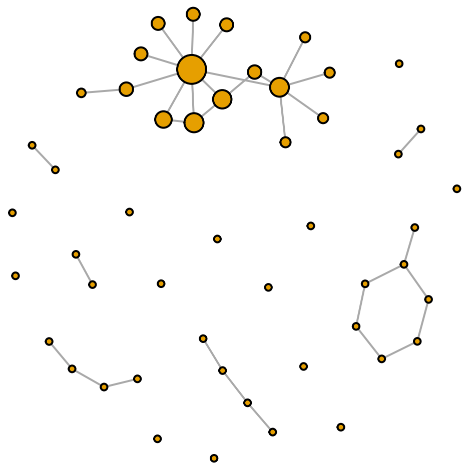
```r
linked_igraph <- adj2igraph(getRefit(se_linked))

# custom scaling function (0,1) that can handle vectors of all zeros
scale01 <- function(x){
  if(max(x) == 0){return(x)} # if all zeros, do nothing
  if(max(x) > 0){return((x - min(x)) / (max(x) - min(x)))} # if some positive, scale as normal
  if(sum(x) < 0){x <- abs(x) # not sure about this
                 scaled <- (x - min(x)) / (max(x) - min(x))
                 return(-scaled)} # if all negative, use absolute values, then replace negative sign
}

# convenience function for plotting hub taxa in networks
plot_hubs <- function(graph,bigpoint=10,littlepoint=3){
  am.coord <- layout.auto(graph)
  art <- articulation_points(graph)
  plot(graph,
       layout = am.coord,
       vertex.size=(scale01(abs(igraph::authority_score(graph)$vector)) * 10)+3,
       vertex.label=NA)
}
```

Now we should be ready for some quick plots

```r
plot_hubs(comm_igraph)
```

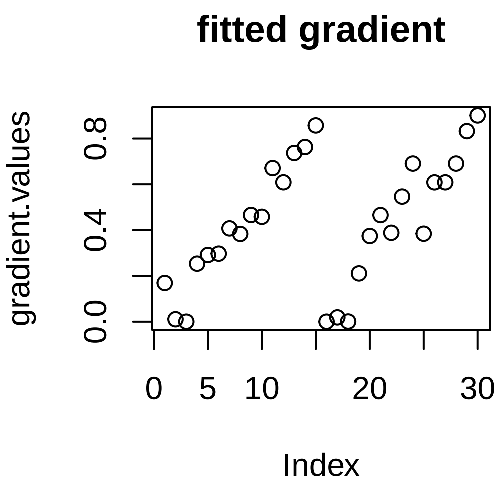


```r
plot_hubs(linked_igraph)
```



We can also add in abiotic filtration along a gradient. We can simluate an environmental gradiane that has a negative impact on 20% of our taxa, and is divided into 2 groups, such as within a replicated experiment.

```r
comm_linked_gradient <- filter_taxa_along_gradient(comm_linked,prop = .2,
                                                   gradient.min = 0,gradient.max = .9,
                                                   gradient.strength = 3,groups = 2,
                                                   association = 'negative',
                                                   show.gradient = TRUE)
```



Now, at last, we've built a random community that matches some theoretical starting position. We can build a donor community that fits the same shape and simulate transplantation under different scenarios. Here, we build a donor community that contains 30 taxa, of which 20% are expected to overlap with resident taxa. The observations per sample will be on the same scale as the orginal (even) community.

```
donor <- build_donor_community(resident.comm = comm,
                               n.transplant.taxa = 30, overlap = .2)
```

And a simulation of a transplantation experiment. . . here, we are forcing 30% of our donor taxa to have antagonistic taxa present in the recipient community. . . we're then introducing the same environmental gradient and letting it apply only to some of the new taxa, since the resident taxa have already been affected by it.

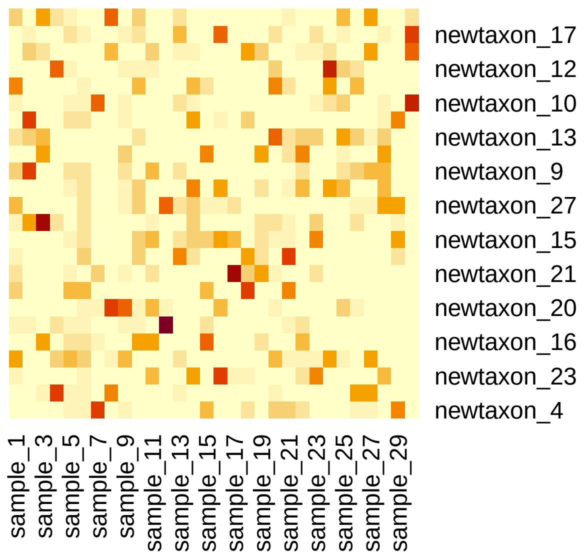
```
transplant <- transplant_w_antagonism(recipient = comm_linked_gradient,
                                      donor = donor,antag.ubiq = .3,
                                      antag.strength = 1) %>%
              filter_taxa_along_gradient(prop = .5,transplant.only = TRUE,
                                      gradient.min = 0,gradient.max = .9,
                                      gradient.strength = 3,groups = 2,
                                      association = 'negative',
                                      show.gradient = FALSE)


transplant[1:5,1:5] %>% kable()
```

	newtaxon_1	newtaxon_4	newtaxon_5	newtaxon_6	newtaxon_8
sample_1	0	0	8	9	59
sample_2	96	0	0	47	0
sample_3	0	0	0	96	0
sample_4	0	0	0	21	41
sample_5	30	10	0	3	34

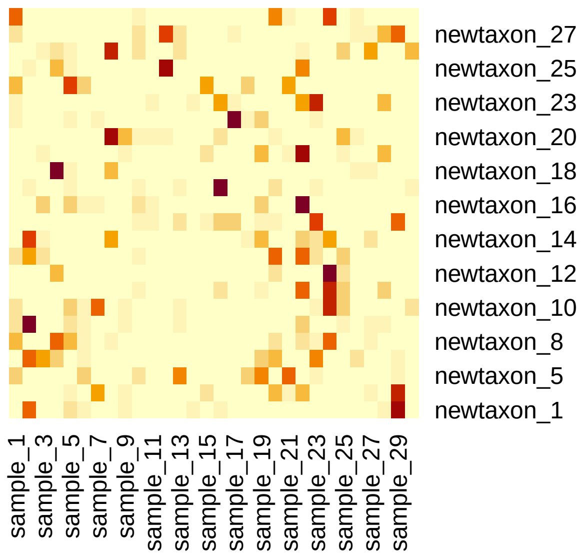
Taxa that are novel in the recipient community are identified with "newtaxon" and now we have completed a simple simulated transplantation experiment with resident antagonism and an inhibitory environmental gradient.

We can do a quick inspection of the communities before and after transplanting:

```r
# convert to relabund
orig_relabund <- vegan::decostand(donor,'total')
new_relabund <- vegan::decostand(transplant,'total')

# novel taxa in the original donor community
heatmap(t(orig_relabund[,grep(colnames(orig_relabund),pattern="newtaxon")]),
        Colv = NA,Rowv=NA)
```

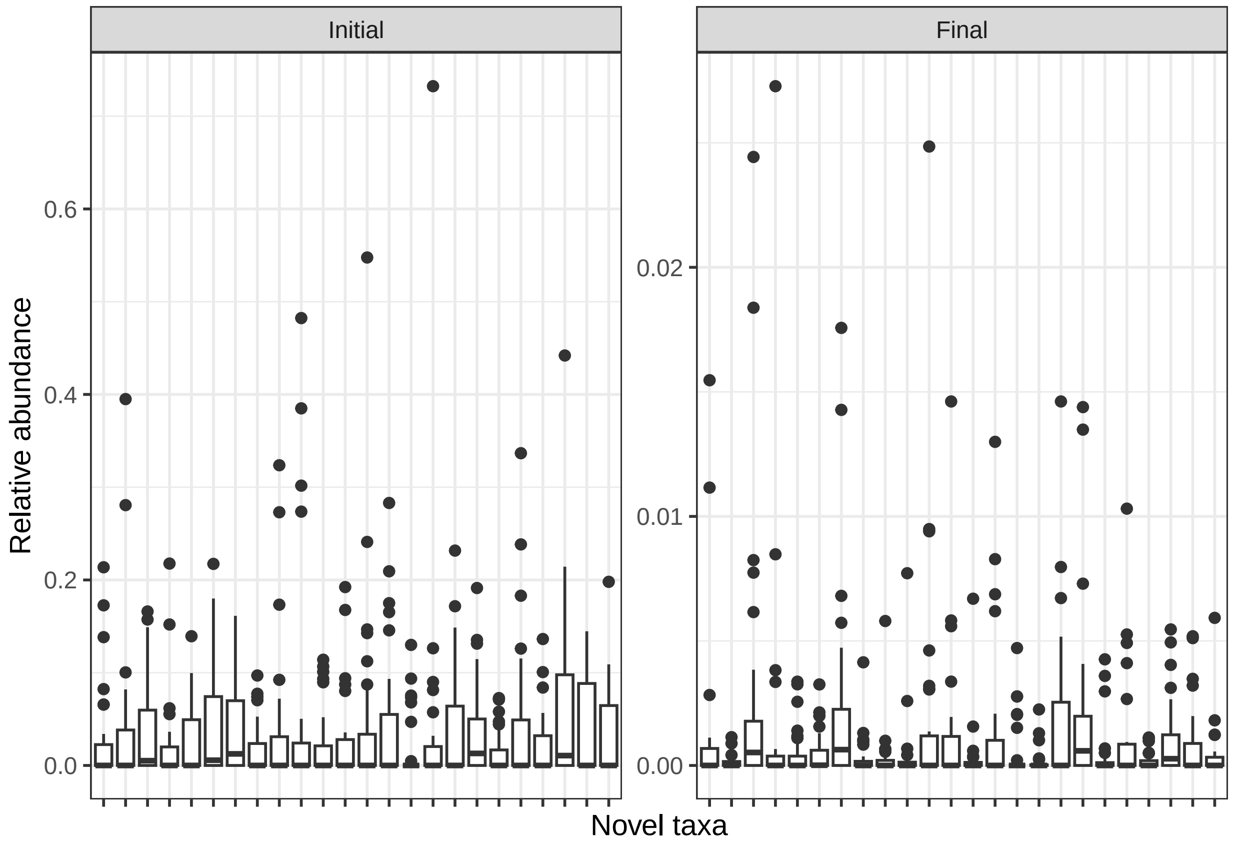


```r
# novel taxa remaining in the new community after transplantation
heatmap(t(new_relabund[,grep(colnames(new_relabund),pattern="newtaxon")]),
        Colv = NA,Rowv=NA)
```



Comparing the novel taxa before and after transplantation:

```r
orig <- (orig_relabund[,grep(colnames(orig_relabund),pattern="newtaxon")]) %>%
  as.data.frame() %>% mutate(group="Initial")
final <- (new_relabund[,grep(colnames(new_relabund),pattern="newtaxon")]) %>%
  as.data.frame() %>% mutate(group="Final")

full_join(orig,final) %>%
  pivot_longer(starts_with("newtaxon")) %>%
  ggplot(aes(x=name,y=value)) +
  geom_boxplot() +
  facet_wrap(~factor(group,levels=c("Initial","Final")),scales = 'free') +
  theme_bw() + theme(axis.text.x = element_blank()) +
  labs(x="Novel taxa",y="Relative abundance")
```


Initial
Final
Relative abundance
Novel taxa