Nestedness_and_Reproductive_Success

Isabelle Nollette, Sydney James

2025-12-08

#INTRODUCTION It is widely understood that plant-pollinator interactions are a key driver of community assembly and species establishment, as greater reproductive success is known to be positively correlated to effective pollination, increasing overall ecosystem productivity Thus, knowing which of these interactions will promote positive species coexistence by enhancing reproductive success is critical to ensuring population persistence in the face of an ever-changing environment (Moeller (2005)). This is especially important in the context of Wetland prairies, which are a critical part of the Oregon ecosystem. Not only are they biodiversity hotspots, but they also improve water quality, recharge groundwater, reduce flood severity, and support breeding and migrating wildlife. However, almost all remaining Willamette Valley prairies have been degraded by human pollution, invasive species, and altered water regimes. Thus, the need for restoration is unquestionably important. The role of pollinators is critical to the reproductive success of plant communities, and has delicate margins in doing so. Thus, research and data analysis such as this project can strengthen existing knowledge on community assembly and pollinator community composition, better informing specific restoration efforts, to prioritise the most productive community assemblages and have a meaningful impact on degraded landscapes like the wet prairies in the Willamette Valley.

Existing literature in the field of community assembly has focused on how direct interactions among plants (e.g. competition for resources) and abiotic factors (e.g. soil moisture) influence their persistence in a particular habitat (Sargent and Ackerly (2008)). Moreover, studies predicting how community and species characteristics such as niche overlaps, specialisation, trait divergence and distribution, and phylogenetics influence patterns of species co-occurrence and persistence within a community are not uncommon (Moeller (2005), Ponisio, Gaiarsa, and Kremen (2017), Sargent and Ackerly (2008), Slingsby and Verboom (2006)). However, this research has little consideration for the role of plant-animal interactions in the greater community context, particularly that of the local pollinator networks as a potential habitat filter and driver of plant fitness. Instead, focus is often on the scale of direct plant-animal interactions.

Many of these studies explore the dynamics of these plant-pollinator interactions as a driver of species richness (Memtsas et al. (2022)), where the objective is often community resilience to an external stressor. Such resiliance is usually correlated to the structure of specialists and generalists of both plants and pollinators within a community. Abundance of generalist pollinators is often high (Moeller (2005)) and advances the transfer of pollen between plants; however, some degree of specialisation is necessary for successful pollination for flowers requiring conspecific pollen (Brosi). This characteristic of community structure is usually defined as the level of nestedness - the tendency for specialists to interact with subsets of the species that generalists interact with. It is thought that a perfectly nested community is the most resilient (Brosi (2016)). Highly specialised systems like the fiscus trees and their wasp pollinators experience a trade-off between high reproductive success and low resilience, therefore experiencing a low level of nestedness.

Thus, our analysis targets this gap in more depth, by investigating the reproductive success as an output of nestedness. We aim to grow scienfitic understanding about the trade-off between resilience and reproductive success as it promotes community population persistence in different community assemblies.

Through deeper analysis of this existing literature, we can gain an understanding of current plant-pollinator network dynamics, where specialist pollinators are often found in greater abundance yet with lower diversity, and the inverse is true for generalist Moeller (2005)). These dynamics can also be understood as a metric of nestedness, which encompasses the specialisation of plants and pollinators as a vector of species richness, and therefore can add to our understanding of the drivers behind reproductive success in these plant-pollinator networks. Additionally, we know that plant species with similar flowering phenologies and floral morphologies may compete for pollination, consequently making pollinator effectiveness and occurrence a driver of reproductive success in communities with congeners (Bell, Karron, and Mitchell (2005)). There is also a wealth of research that outlines the two potential mechanisms of competition for pollination, as pollinator preference and improper pollen transfer, where the latter occurs when heterospecific pollen is deposited on stigmas of one or both competitors. Both of these mechanisms are important factors in plant-pollinator networks as they can be confounding on reproductive success, and limit the extent to which congeners can beneficially coexist (Bell, Karron, and Mitchell (2005)).

Surveys of floral visitor communities across large geographic ranges to examine the network of pollinators and their contribution to community assembly are frequent and provide a strong framework for the methods of this study. Differently, the data for this project allows for the examination of pollination effectiveness at a smaller spatial scale, and focuses analysis in communities native to a defined ecoregion to enhance targeted restoration efforts.

Understanding research done by Moeller (2005) and Brosi (2016) on the structure of specialisation in pollinator communities visiting focal plants across a wide geographic range is fundamental to the formation of our hypothesis. Building upon the understanding that the difference in pollinator communities between plant communities is due to the proportion of specialists versus generalists, rather than diversity, we form our first hypothesis: We predict that focal plant reproductive success will be greater in communities with less floral overlap and a more specialised pollinator community structure, due to reduced pollen interference and resource competition. These communities are predicted to be the least nested. Conversely, we predict that communities with more congeners to focal plants will attract greater overall abundance of pollinators; however, reproductive success and pollinator biodiversity will be lower. These communities are predicted to be the most nested. Therefore, the optimal, most nested plant assemblage for restoration will feature limited plant overlap but high pollinator sharing.

#DATA CLEANING The data sets we used in this analysis were collected by members of the Hallett and Ponsio Labs at the University of Oregon at the Holiday Farm site from 2020 to 2024. In order to ensure our data was clean we used a linear regression, created from the relationship between bloom count and seed count of plants that were properly bagged to ensure seeds were collected, to generate seed count data based on the mass of the inflorescence of samples that were improperly bagged and therefore did not have counted seeds.

```
library(bipartite)
```

```
## Loading required package: sna
```

```
## Loading required package: statnet.common
```

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

```
## The following objects are masked from 'package:base':
##
##     attr, order, replace
```

```
## Loading required package: network
```

```
##
## 'network' 1.19.0 (2024-12-08), part of the Statnet Project
## * 'news(package="network")' for changes since last version
## * 'citation("network")' for citation information
## * 'https://statnet.org' for help, support, and other information
## sna: Tools for Social Network Analysis
## Version 2.8 created on 2024-09-07.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##  For citation information, type citation("sna").
##  Type help(package="sna") to get started.
```

```
## Loading required package: vegan
```

```
## Loading required package: permute
```

```
##  This is bipartite 2.23.
##  For latest changes see versionlog in ?"bipartite-package". For citation see: citat
ion("bipartite").
##  Have a nice time plotting and analysing two-mode networks.
```

```
##
## Attaching package: 'bipartite'
```

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

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(tibble)
MICGRA1 <- read.csv("/Users/sydneyjames/OneDrive_UniversityofOregon/final_community_as
sembly-SJIN/data/ MICGRA.csv")
#MICGRA1 <- read.csv("/Users/isabellenollette/Desktop/final_community_assembly-SJIN/da
ta/ MICGRA.csv")

## remove unuseful columns
MICGRA <- subset(MICGRA1, select = -c(S, Herbivory, X, Herbivory.1, X.1, ONotes, CNote
s, Opods, Cpods))

## remove mono plots
MICGRA <- subset(MICGRA, M != 0)

## turn any blooms = 0 into NAs
MICGRA$Oblooms[MICGRA$Oblooms == 0] <- NA
MICGRA$Cblooms[MICGRA$Cblooms == 0] <- NA

## remove any NAs
MICGRA <- MICGRA %>%
  drop_na(Oseeds, Oblooms, Cseeds, Cblooms)

## find seeds per bloom and add column
MICGRA <- MICGRA %>%
  mutate(OSeeds_Per_bloom = round(Oseeds/Oblooms, digits = 2)) %>%
  mutate(CSeeds_Per_bloom = round(Cseeds/Cblooms, digits = 2))

## rename columns to match pollinator data
MICGRA <- MICGRA %>%
  rename("Stand" = St,
         "Seed_Mix" = M,
         "Treatment" = genSpec,
         "PlantGenusSpecies" = Sp)

GILCAP1 <- read.csv("/Users/sydneyjames/OneDrive_UniversityofOregon/final_community_as
sembly-SJIN/data/GILCAP.csv")
#GILCAP1 <- read.csv("/Users/isabellenollette/Desktop/final_community_assembly-SJIN/da
ta/GILCAP.csv")

## remove unuseful columns
GILCAP <- subset(GILCAP1, select = -c(S, Herbivory, OflwrsGood, OflwrsBad, OseedsBad,
OseedsGood, Omass, Herbivory.1, CflwrsGood, CflwrsBad, CseedsBad, CseedsGood, Cmass, n
otesOpen, notesClosed, Opods, Cpods, X))

## remove mono plots
GILCAP <- subset(GILCAP, M != 0)

## turn any blooms = 0 into NAs
GILCAP$Oblooms[GILCAP$Oblooms == 0] <- NA
GILCAP$Cblooms[GILCAP$Cblooms == 0] <- NA

## remove any NAs
GILCAP <- GILCAP %>%
  drop_na(OseedsTotal, Oblooms, CseedsTotal, Cblooms)

## find seeds per bloom and add column
```

```
GILCAP <- GILCAP %>%
  mutate(OSeeds_Per_bloom = round(OseedsTotal/Oblooms, digits = 2)) %>%
  mutate(CSeeds_Per_bloom = round(CseedsTotal/Cblooms, digits = 2))

## rename columns to match pollinator data
GILCAP <- GILCAP %>%
  rename("Stand" = St,
         "Seed_Mix" = M,
         "Treatment" = genSpec,
         "PlantGenusSpecies" = Sp)

GLTCCOLLIGRA1 <- read.csv("/Users/sydneyjames/OneDrive_UniversityofOregon/final_commun
ity_assembly-SJIN/data/GLTCCOLLIGRA.csv")
#GLTCCOLLIGRA1 <- read.csv("/Users/isabellenollette/Desktop/final_community_assembly-S
JIN/data/GLTCCOLLIGRA.csv")

## remove unuseful columns
GLTCCOLLIGRA <- subset(GLTCCOLLIGRA1, select = -c(S, OseedsGood, OseedsBad, Ocomplet
e., OemptyInflorMass, CseedsGood, CseedsBad, Ccomplete., CemptyInflorMass, notesOpen,
notesClosed, X, X.1, X.2, X.3, X.4, X.5))

## remove mono plots
GLTCCOLLIGRA <- subset(GLTCCOLLIGRA, M != 0)

## turn any blooms = 0 into NAs
GLTCCOLLIGRA$Oblooms[GLTCCOLLIGRA$Oblooms == 0] <- NA
GLTCCOLLIGRA$Cblooms[GLTCCOLLIGRA$Cblooms == 0] <- NA

## remove any NAs
GLTCCOLLIGRA <- GLTCCOLLIGRA %>%
  drop_na(OseedsTotal, Oblooms, CseedsTotal, Cblooms)

## find seeds per bloom and add column
GLTCCOLLIGRA <- GLTCCOLLIGRA %>%
  mutate(OSeeds_Per_bloom = round(OseedsTotal/Oblooms, digits = 2)) %>%
  mutate(CSeeds_Per_bloom = round(CseedsTotal/Cblooms, digits = 2))

## rename columns to match pollinator data
GLTCCOLLIGRA <- GLTCCOLLIGRA %>%
  rename("Stand" = St,
         "Seed_Mix" = M,
         "Treatment" = genSpec,
         "PlantGenusSpecies" = Sp)

#GLTC_MADGRA <- read.csv("/Users/isabellenollette/Desktop/final_community_assembly-SJI
N/data/GLTCMADGRA.csv")
GLTC_MADGRA <- read.csv("/Users/sydneyjames/OneDrive_UniversityofOregon/final_communit
y_assembly-SJIN/data/GLTCMADGRA.csv")

GLTC_MADGRA <- subset(GLTC_MADGRA,
                      select = -c(S, lostSeeds.O, Oflorets, lostSeeds.C, Cflorets, not
esOpen, notesClosed, notesOther))

## removing mono plots
GLTC_MADGRA <- subset(GLTC_MADGRA, M != 0)
#changing ""  and 0 to NAs
GLTC_MADGRA[GLTC_MADGRA == ""| GLTC_MADGRA == 0] <- NA
```

```
#dropped NAs
GLTC_MADGRA <- GLTC_MADGRA %>%
  drop_na(Oblooms, Cblooms)

#making the headers match holiday data
GLTC_MADGRA <- GLTC_MADGRA %>%
  rename("Stand" = St,
         "Seed_Mix" = M,
         "Treatment" = genSpec,
         "PlantGenusSpecies" = Sp)

## making new seeds column
GLTC_MADGRA <- GLTC_MADGRA %>%
  mutate(OSeeds =
           ifelse((!is.na(OseedsGood) & OseedsGood >= 0) | (!is.na(Oseeds.Bad) & Oseed
s.Bad >=0), rowSums(cbind(OseedsGood, Oseeds.Bad), na.rm = TRUE),
                  ifelse(!is.na(OseedsGood) | !is.na(Oseeds.Bad),
                         ifelse(!is.na(OseedsGood), OseedsGood, Oseeds.Bad),
                         ifelse(!is.na(OseedsTotal), OseedsTotal,
                                NA))))
GLTC_MADGRA <- GLTC_MADGRA %>%
  mutate(CSeeds =
           ifelse((!is.na(CseedsGood) & CseedsGood >= 0) | (!is.na(CseedsBad) & Cseeds
Bad >=0), rowSums(cbind(CseedsGood, CseedsBad), na.rm = TRUE),
                  ifelse(!is.na(CseedsGood) | !is.na(CseedsBad),
                         ifelse(!is.na(CseedsGood), CseedsGood, CseedsBad),
                         ifelse(!is.na(CseedsTotal), CseedsTotal,
                                NA))))

##dropping old seed columns
GLTC_MADGRA <- subset(GLTC_MADGRA,
                      select = -c(OseedsGood, Oseeds.Bad, OseedsTotal, CseedsGood, Cse
edsBad, CseedsTotal))

##Linear regression for seed mass to fill in empty seed values
## FOR OPEN subset for data rows that have both total seeds and mass:
OSeed_mass <- subset(GLTC_MADGRA, OSeeds !=0 & Omass != 0)
## linear reg:
model <- lm(OSeed_mass$OSeeds~OSeed_mass$Omass)
Ob <- model$coefficients[1]
Om <- model$coefficients[2]
##subset data to get rows that have mass but don't have seeds
OnoSeeds_yesMass <- subset(GLTC_MADGRA, (OSeeds == 0 | is.na(OSeeds)) & Omass != 0)
## get the mass column for the regression
Ox <- OnoSeeds_yesMass$Omass
Oy <- Om*Ox+Ob
##fill in empty rows
OnoSeeds_yesMass$OSeeds <- Oy

## FOR CLOSED subset for data rows that have both total seeds and mass:
CSeed_mass <- subset(GLTC_MADGRA, CSeeds !=0 & Cmass != 0)
## linear reg:
model <- lm(CSeed_mass$CSeeds~CSeed_mass$Cmass)
Cb <- model$coefficients[1]
Cm <- model$coefficients[2]
##subset data to get rows that have mass but don't have seeds
CnoSeeds_yesMass <- subset(GLTC_MADGRA, (CSeeds == 0 | is.na(CSeeds)) & Cmass != 0)
```

```
## get the mass column for the regression
Cx <- CnoSeeds_yesMass$Cmass
Cy <- Cm*Cx+Cb
##fill in empty rows
CnoSeeds_yesMass$CSeeds <- Cy

cleaned_GLTC_MADGRA <- GLTC_MADGRA
CnoSeeds_yesMass.small <- CnoSeeds_yesMass %>% select(Cmass, CSeeds)
OnoSeeds_yesMass.small <- OnoSeeds_yesMass %>% select(Omass, OSeeds)

## Join OSeeds values
cleaned_GLTC_MADGRA <- cleaned_GLTC_MADGRA %>%
  left_join(OnoSeeds_yesMass.small, by = "Omass", suffix = c("", ".new")) %>%
  mutate(OSeeds = ifelse(is.na(OSeeds), OSeeds.new, OSeeds)) %>%
  select(-OSeeds.new)

## Join CSeeds values
cleaned_GLTC_MADGRA <- cleaned_GLTC_MADGRA %>%
  left_join(CnoSeeds_yesMass.small, by = "Cmass", suffix = c("", ".new")) %>%
  mutate(CSeeds = ifelse(is.na(CSeeds), CSeeds.new, CSeeds)) %>%
  select(-CSeeds.new)

## adding seeds per bloom
cleaned_GLTC_MADGRA <- cleaned_GLTC_MADGRA %>%
  mutate(OSeeds_Per_Bloom = OSeeds/Oblooms,
         CSeeds_Per_Bloom = CSeeds/Cblooms)
#drop remaining NAs
cleaned_GLTC_MADGRA <- cleaned_GLTC_MADGRA %>%
  drop_na(OSeeds, CSeeds) %>%
  subset(select = -c(Omass, Cmass))

#WWCLAAMO <- read.csv("/Users/isabellenollette/Desktop/final_community_assembly-SJIN/d
ata/WWCLAAMO.csv")
WWCLAAMO <- read.csv("/Users/sydneyjames/OneDrive_UniversityofOregon/final_community_a
ssembly-SJIN/data/WWCLAAMO.csv")

WWCLAAMO <- subset(WWCLAAMO,
                   select = -c(S, notesOpen, notesClosed))

#removing mono plots
WWCLAAMO <- subset(WWCLAAMO, M != 0)
#changing ""  and 0 to NAs
WWCLAAMO[WWCLAAMO == ""| WWCLAAMO == 0] <- NA
#dropped NAs
WWCLAAMO <- WWCLAAMO %>%
  drop_na(Oblooms, Cblooms)

#making the headers match holiday data
WWCLAAMO <- WWCLAAMO %>%
  rename("Stand" = St,
         "Seed_Mix" = M,
         "Treatment" = genSpec,
         "PlantGenusSpecies" = Sp)

#making new seeds column
WWCLAAMO <- WWCLAAMO %>%
  mutate(OSeeds =
           ifelse((!is.na(OseedsGood) & OseedsGood >= 0) | (!is.na(OseedsBad) & Oseeds
Bad >=0), rowSums(cbind(OseedsGood, OseedsBad), na.rm = TRUE),
                  ifelse(!is.na(OseedsGood) | !is.na(OseedsBad),
                         ifelse(!is.na(OseedsGood), OseedsGood, OseedsBad),
                         ifelse(!is.na(OseedsTotal), OseedsTotal,
                                NA))))
WWCLAAMO <- WWCLAAMO %>%
  mutate(CSeeds =
           ifelse((!is.na(CseedsGood) & CseedsGood >= 0) | (!is.na(CseedsBad) & Cseeds
Bad >=0), rowSums(cbind(CseedsGood, CseedsBad), na.rm = TRUE),
                  ifelse(!is.na(CseedsGood) | !is.na(CseedsBad),
                         ifelse(!is.na(CseedsGood), CseedsGood, CseedsBad),
                         ifelse(!is.na(CseedsTotal), CseedsTotal,
                                NA))))

#dropping old seed columns
WWCLAAMO <- subset(WWCLAAMO,
                      select = -c(OseedsGood, OseedsBad, OseedsTotal, CseedsGood, Csee
dsBad, CseedsTotal))
##Linear regression for seed mass:
## FOR OPEN subset for data rows that have both total seeds and mass:
OSeed_mass <- subset(WWCLAAMO, OSeeds !=0 & Omass != 0)
## linear reg:
model <- lm(OSeed_mass$OSeeds~OSeed_mass$Omass)
Ob <- model$coefficients[1]
Om <- model$coefficients[2]
##subset data to get rows that have mass but dont have seeds
OnoSeeds_yesMass <- subset(WWCLAAMO, OSeeds == 0 | is.na(OSeeds) & Omass != 0)
## get the mass column for the regression
Ox <- OnoSeeds_yesMass$Omass
Oy <- Om*Ox+Ob
##fill in empty rows
OnoSeeds_yesMass$OSeeds <- Oy

## FOR CLOSED subset for data rows that have both total seeds and mass:
CSeed_mass <- subset(WWCLAAMO, CSeeds !=0 & Cmass != 0)
## linear reg:
model <- lm(CSeed_mass$CSeeds~CSeed_mass$Cmass)
Cb <- model$coefficients[1]
Cm <- model$coefficients[2]
##subset data to get rows that have mass but dont have seeds
CnoSeeds_yesMass <- subset(WWCLAAMO, CSeeds == 0 | is.na(CSeeds) & Cmass != 0)
## get the mass column for the regression
Cx <- CnoSeeds_yesMass$Cmass
Cy <- Cm*Cx+Cb
##fill in empty rows
CnoSeeds_yesMass$CSeeds <- Cy

cleaned_WWCLAAMO <- WWCLAAMO
CnoSeeds_yesMass.small <- CnoSeeds_yesMass %>% select(Cmass, CSeeds)
OnoSeeds_yesMass.small <- OnoSeeds_yesMass %>% select(Omass, OSeeds)

# Join OSeeds values
cleaned_WWCLAAMO <- cleaned_WWCLAAMO %>%
  left_join(OnoSeeds_yesMass.small, by = "Omass", suffix = c("", ".new")) %>%
  mutate(OSeeds = ifelse(is.na(OSeeds), OSeeds.new, OSeeds)) %>%
  select(-OSeeds.new)
```

```
# Join CSeeds values
cleaned_WWCLAAMO <- cleaned_WWCLAAMO %>%
  left_join(CnoSeeds_yesMass.small, by = "Cmass", suffix = c("", ".new")) %>%
  mutate(CSeeds = ifelse(is.na(CSeeds), CSeeds.new, CSeeds)) %>%
  select(-CSeeds.new)

#adding seeds per bloom
cleaned_WWCLAAMO <- cleaned_WWCLAAMO %>%
  mutate(OSeeds_Per_Bloom = OSeeds/Oblooms,
         CSeeds_Per_Bloom = CSeeds/Cblooms)
#drop remaining NAs
cleaned_WWCLAAMO <- cleaned_WWCLAAMO %>%
  drop_na(OSeeds, CSeeds) %>%
  subset(select = -c(Omass, Cmass))

#WWCLAPUR <- read.csv("/Users/isabellenollette/Desktop/final_community_assembly-SJIN/d
ata/WWCLAPUR.csv")
WWCLAPUR <- read.csv("/Users/sydneyjames/OneDrive_UniversityofOregon/final_community_a
ssembly-SJIN/data/WWCLAPUR.csv")

WWCLAPUR <- subset(WWCLAPUR,
                      select = -c(S, notes, notesClosed))

#removing mono plots
WWCLAPUR <- subset(WWCLAPUR, M != 0)
#changing ""  and 0 to NAs
WWCLAPUR[WWCLAPUR == ""| WWCLAPUR == 0] <- NA
#dropped NAs
WWCLAPUR <- WWCLAPUR %>%
  drop_na(Oblooms, Cblooms)

#making the headers match holiday data
WWCLAPUR <- WWCLAPUR %>%
  rename("Stand" = St,
         "Seed_Mix" = M,
         "Treatment" = genSpec,
         "PlantGenusSpecies" = Sp)

#making new seeds column
WWCLAPUR <- WWCLAPUR %>%
  mutate(OSeeds =
           ifelse((!is.na(OseedsGood) & OseedsGood >= 0) | (!is.na(OseedsBad) & Oseeds
Bad >=0), rowSums(cbind(OseedsGood, OseedsBad), na.rm = TRUE),
                  ifelse(!is.na(OseedsGood) | !is.na(OseedsBad),
                         ifelse(!is.na(OseedsGood), OseedsGood, OseedsBad),
                         ifelse(!is.na(OseedsTotal), OseedsTotal,
                                NA))))
WWCLAPUR <- WWCLAPUR %>%
  mutate(CSeeds =
           ifelse((!is.na(CseedsGood) & CseedsGood >= 0) | (!is.na(CseedsBad) & Cseeds
Bad >=0), rowSums(cbind(CseedsGood, CseedsBad), na.rm = TRUE),
                  ifelse(!is.na(CseedsGood) | !is.na(CseedsBad),
                         ifelse(!is.na(CseedsGood), CseedsGood, CseedsBad),
                         ifelse(!is.na(CseedsTotal), CseedsTotal,
                                NA))))
```

```
#dropping old seed columns
cleaned_WWCLAPUR <- subset(WWCLAPUR,
                      select = -c(OseedsGood, OseedsBad, OseedsTotal, CseedsGood, Csee
dsBad, CseedsTotal))

#adding seeds per bloom
cleaned_WWCLAPUR <- cleaned_WWCLAPUR %>%
  mutate(OSeeds_Per_Bloom = OSeeds/Oblooms,
         CSeeds_Per_Bloom = CSeeds/Cblooms)
#drop remaining NAs
cleaned_WWCLAPUR <- cleaned_WWCLAPUR %>%
  drop_na(OSeeds, CSeeds) %>%
  subset(select = -c(Omass, Cmass))

#WWCOLLIGRA <- read.csv("/Users/isabellenollette/Desktop/final_community_assembly-SJI
N/data/WWCOLLIGRA.csv")
WWCOLLIGRA <- read.csv("/Users/sydneyjames/OneDrive_UniversityofOregon/final_community
_assembly-SJIN/data/WWCOLLIGRA.csv")

WWCOLLIGRA <- subset(WWCOLLIGRA,
                      select = -c(S, Ocomplete., Ccomplete., notesOpen, notesClosed, 1
9,20,21,22,23,24))  #remove unuseful columns

#removing mono plots
WWCOLLIGRA <- subset(WWCOLLIGRA, M != 0)
#changing ""  and 0 to NAs
WWCOLLIGRA[WWCOLLIGRA == ""| WWCOLLIGRA == 0] <- NA
#dropped NAs
WWCOLLIGRA <- WWCOLLIGRA %>%
  drop_na(Oblooms, Cblooms)

#making the headers match holiday data
WWCOLLIGRA <- WWCOLLIGRA %>%
  rename("Stand" = St,
          "Seed_Mix" = M,
          "Treatment" = genSpec,
          "PlantGenusSpecies" = Sp,
         "Omass" = OemptyInflorMass,
         "Cmass" = CemptyInflorMass)

#making new seeds column
WWCOLLIGRA <- WWCOLLIGRA %>%
  mutate(OSeeds =
           ifelse((!is.na(OseedsGood) & OseedsGood >= 0) | (!is.na(OseedsBad) & Oseeds
Bad >=0), rowSums(cbind(OseedsGood, OseedsBad), na.rm = TRUE),
                  ifelse(!is.na(OseedsGood) | !is.na(OseedsBad),
                         ifelse(!is.na(OseedsGood), OseedsGood, OseedsBad),
                                NA)))
WWCOLLIGRA <- WWCOLLIGRA %>%
  mutate(CSeeds =
           ifelse((!is.na(CseedsGood) & CseedsGood >= 0) | (!is.na(CseedsBad) & Cseeds
Bad >=0), rowSums(cbind(CseedsGood, CseedsBad), na.rm = TRUE),
                  ifelse(!is.na(CseedsGood) | !is.na(CseedsBad),
                         ifelse(!is.na(CseedsGood), CseedsGood, CseedsBad),
                                NA)))

#dropping old seed columns
```

```
WWCOLLIGRA <- subset(WWCOLLIGRA,
                      select = -c(OseedsGood, OseedsBad, CseedsGood, CseedsBad))
##Linear regression for seed mass to fill in missing values
## FOR OPEN subset for data rows that have both total seeds and mass:
OSeed_mass <- subset(WWCOLLIGRA, OSeeds !=0 & Omass != 0)
## linear reg:
model <- lm(OSeed_mass$OSeeds~OSeed_mass$Omass)
Ob <- model$coefficients[1]
Om <- model$coefficients[2]
##subset data to get rows that have mass but dont have seeds
OnoSeeds_yesMass <- subset(WWCOLLIGRA, OSeeds == 0 | is.na(OSeeds) & Omass != 0)
## get the mass column for the regression
Ox <- OnoSeeds_yesMass$Omass
Oy <- Om*Ox+Ob
##fill in empty rows
OnoSeeds_yesMass$OSeeds <- Oy

## FOR CLOSED subset for data rows that have both total seeds and mass:
CSeed_mass <- subset(WWCOLLIGRA, CSeeds !=0 & Cmass != 0)
## linear reg:
model <- lm(CSeed_mass$CSeeds~CSeed_mass$Cmass)
Cb <- model$coefficients[1]
Cm <- model$coefficients[2]
##subset data to get rows that have mass but dont have seeds
CnoSeeds_yesMass <- subset(WWCOLLIGRA, CSeeds == 0 | is.na(CSeeds) & Cmass != 0)
## get the mass column for the regression
Cx <- CnoSeeds_yesMass$Cmass
Cy <- Cm*Cx+Cb
##fill in empty rows
CnoSeeds_yesMass$CSeeds <- Cy

cleaned_WWCOLLIGRA <- WWCOLLIGRA
CnoSeeds_yesMass.small <- CnoSeeds_yesMass %>% select(Cmass, CSeeds)
OnoSeeds_yesMass.small <- OnoSeeds_yesMass %>% select(Omass, OSeeds)

# Join OSeeds values
cleaned_WWCOLLIGRA <- cleaned_WWCOLLIGRA %>%
  left_join(OnoSeeds_yesMass.small, by = "Omass", suffix = c("", ".new")) %>%
  mutate(OSeeds = ifelse(is.na(OSeeds), OSeeds.new, OSeeds)) %>%
  select(-OSeeds.new)

# Join CSeeds values
cleaned_WWCOLLIGRA <- cleaned_WWCOLLIGRA %>%
  left_join(CnoSeeds_yesMass.small, by = "Cmass", suffix = c("", ".new")) %>%
  mutate(CSeeds = ifelse(is.na(CSeeds), CSeeds.new, CSeeds)) %>%
  select(-CSeeds.new)

#adding seeds per bloom
cleaned_WWCOLLIGRA <- cleaned_WWCOLLIGRA %>%
  mutate(OSeeds_Per_Bloom = OSeeds/Oblooms,
         CSeeds_Per_Bloom = CSeeds/Cblooms)
#drop remaining NAs
cleaned_WWCOLLIGRA <- cleaned_WWCOLLIGRA %>%
  drop_na(OSeeds, CSeeds) %>%
  subset(select = -c(Omass, Cmass))

#WWMADGRA <- read.csv("/Users/isabellenollette/Desktop/final_community_assembly-SJIN/d
ata/WWMADGRA.csv")
WWMADGRA <-read.csv("/Users/sydneyjames/OneDrive_UniversityofOregon/final_community_as
sembly-SJIN/data/WWMADGRA.csv")

WWMADGRA <- subset(WWMADGRA, M != 0)       #removing mono plots

WWMADGRA <- WWMADGRA %>%
  rename("Stand" = St,
         "Seed_Mix" = M,
         "Treatment" = genSpec,
         "PlantGenusSpecies" = Sp)   #making the headers correct

#making new seeds column
WWMADGRA <- WWMADGRA %>%
  mutate(OSeeds =
           ifelse((!is.na(OseedsGood) & OseedsGood >= 0) | (!is.na(Oseeds.Bad) & Oseed
s.Bad >=0), rowSums(cbind(OseedsGood, Oseeds.Bad), na.rm = TRUE),
                  ifelse(!is.na(OseedsGood) | !is.na(Oseeds.Bad),
                         ifelse(!is.na(OseedsGood), OseedsGood, Oseeds.Bad),
                                NA)))
WWMADGRA <- WWMADGRA %>%
  mutate(CSeeds =
           ifelse((!is.na(CseedsGood) & CseedsGood >= 0) | (!is.na(CseedsBad) & Cseeds
Bad >=0), rowSums(cbind(CseedsGood, CseedsBad), na.rm = TRUE),
                  ifelse(!is.na(CseedsGood) | !is.na(CseedsBad),
                         ifelse(!is.na(CseedsGood), CseedsGood, CseedsBad),
                                NA)))

#dropping old seed columns
WWMADGRA <- subset(WWMADGRA,
                      select = -c(OseedsGood, Oseeds.Bad, CseedsGood, CseedsBad))
##Linear regression for seed mass:
## FOR OPEN subset for data rows that have both total seeds and mass:
OSeed_mass <- subset(WWMADGRA, OSeeds !=0 & Omass != 0)
## linear reg:
model <- lm(OSeed_mass$OSeeds~OSeed_mass$Omass)
Ob <- model$coefficients[1]
Om <- model$coefficients[2]
##subset data to get rows that have mass but dont have seeds
OnoSeeds_yesMass <- subset(WWMADGRA, OSeeds == 0 | is.na(OSeeds) & Omass != 0)
## get the mass column for the regression
Ox <- as.numeric(OnoSeeds_yesMass$Omass)
```

```
## Warning: NAs introduced by coercion
```

```
Oy <- Om*Ox+Ob
##fill in empty rows
OnoSeeds_yesMass$OSeeds <- Oy

## FOR CLOSED subset for data rows that have both total seeds and mass:
CSeed_mass <- subset(WWMADGRA, CSeeds !=0 & Cmass != 0)
## linear reg:
model <- lm(CSeed_mass$CSeeds~CSeed_mass$Cmass)
Cb <- model$coefficients[1]
Cm <- model$coefficients[2]
##subset data to get rows that have mass but dont have seeds
CnoSeeds_yesMass <- subset(WWMADGRA, CSeeds == 0 | is.na(CSeeds) & Cmass != 0)
## get the mass column for the regression
Cx <- CnoSeeds_yesMass$Cmass
Cy <- Cm*Cx+Cb
##fill in empty rows
CnoSeeds_yesMass$CSeeds <- Cy
```

We then combined seed set data for each focal species into a single seed data set, where only good seed values were kept and totaled from both open (pollinated) and closed (unpollinated) flowers.

```
cleaned_WWMADGRA <- WWMADGRA
CnoSeeds_yesMass.small <- CnoSeeds_yesMass %>% select(Cmass, CSeeds)
OnoSeeds_yesMass.small <- OnoSeeds_yesMass %>% select(Omass, OSeeds)

# Join OSeeds values
cleaned_WWMADGRA <- cleaned_WWMADGRA %>%
  left_join(OnoSeeds_yesMass.small, by = "Omass", suffix = c("", ".new")) %>%
  mutate(OSeeds = ifelse(is.na(OSeeds), OSeeds.new, OSeeds)) %>%
  select(-OSeeds.new)
```

```
## Warning in left_join(., OnoSeeds_yesMass.small, by = "Omass", suffix = c("", : Dete
cted an unexpected many-to-many relationship between `x` and `y`.
## ℹ Row 6 of `x` matches multiple rows in `y`.
## ℹ Row 4 of `y` matches multiple rows in `x`.
## ℹ If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
# Join CSeeds values
cleaned_WWMADGRA <- cleaned_WWMADGRA %>%
  left_join(CnoSeeds_yesMass.small, by = "Cmass", suffix = c("", ".new")) %>%
  mutate(CSeeds = ifelse(is.na(CSeeds), CSeeds.new, CSeeds)) %>%
  select(-CSeeds.new)

#adding seeds per bloom
cleaned_WWMADGRA <- cleaned_WWMADGRA %>%
  mutate(OSeeds_Per_Bloom = OSeeds/as.numeric(Oblooms),
         CSeeds_Per_Bloom = CSeeds/Cblooms)
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `OSeeds_Per_Bloom = OSeeds/as.numeric(Oblooms)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
#drop remaining NAs
cleaned_WWMADGRA <- cleaned_WWMADGRA %>%
  drop_na(OSeeds, CSeeds) %>%
  subset(select = -c(Omass, Cmass))

GILCAP_cleaned <- GILCAP %>%
  rename(OSeeds = OseedsTotal,
         CSeeds = CseedsTotal,
         OSeeds_Per_Bloom = OSeeds_Per_bloom,
         CSeeds_Per_Bloom = CSeeds_Per_bloom)

GLTCCOLLIGRA_cleaned <- GLTCCOLLIGRA %>%
  rename(OSeeds = OseedsTotal,
         CSeeds = CseedsTotal,
         OSeeds_Per_Bloom = OSeeds_Per_bloom,
         CSeeds_Per_Bloom = CSeeds_Per_bloom)

MICGRA_cleaned <- MICGRA %>%
  rename(OSeeds = Oseeds,
         CSeeds = Cseeds,
         OSeeds_Per_Bloom = OSeeds_Per_bloom,
         CSeeds_Per_Bloom = CSeeds_Per_bloom)



All_Cleaned_Data <- rbind(GILCAP_cleaned, MICGRA_cleaned,GLTCCOLLIGRA_cleaned, cleaned
_GLTC_MADGRA,cleaned_WWCLAAMO, cleaned_WWCLAPUR, cleaned_WWCOLLIGRA)
write.csv(All_Cleaned_Data, "/Users/sydneyjames/OneDrive_UniversityofOregon/final_comm
unity_assembly-SJIN/data/All_Cleaned_Data")
```

Our pollinator interaction data was provided by the Ponisio lab and included two data sets, one which featured vegetation data and one which provided the pollinator interactions for each plant. We filtered the entire pollinator interaction data set to only include one year of post-fire data (2024) to best match our seed data timeline, and joined these data frames together by each stand, mix, and collection date combination to create a list of adjacency matrices containing all possible plant pollinator interactions.

```
veg <- read.csv("/Users/sydneyjames/OneDrive_UniversityofOregon/final_community_assemb
ly-SJIN/data/veg.csv")
ints <- read.csv("/Users/sydneyjames/OneDrive_UniversityofOregon/final_community_assem
bly-SJIN/data/Holiday_PP_Interaction.csv")
 #veg <- read.csv("/Users/isabellenollette/Desktop/final_community_assembly-SJIN/data/
veg.csv")
 #ints <- read.csv("/Users/isabellenollette/Desktop/final_community_assembly-SJIN/dat
a/Holiday_PP_Interaction.csv")

veg_data <- veg %>%
  separate(col = Stand, into = c("Stand", "Loc"), sep = ":") %>%
  subset(Stand = c("604","605","607","611","700","702","704","705")) %>%
  subset(Site == 6) %>%
  subset(Transect != "Unenhanced") %>%
  subset(select = c("Stand", "Date", "Transect", "PlantGenusSpecies"))
```

```
## Warning: In subset.data.frame(., Stand = c("604", "605", "607", "611", "700",
##     "702", "704", "705")) :
##  extra argument 'Stand' will be disregarded
```

```
ints1 <- ints %>%
  filter(Year == 2024) %>%
  separate(col = Stand, into = c("Year1", "Stand"), sep = "_") %>%
  separate(col = Stand, into = c("Stand", "Site"), sep = ":") %>%
  filter(Stand == "604" | Stand == "605" | Stand == "607" | Stand == "611" | Stand ==
"700" | Stand ==     "702" | Stand == "704" | Stand == "705")
```

```
## Warning: Expected 2 pieces. Additional pieces discarded in 267 rows [1, 2, 3, 4, 5,
6,
## 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...].
```

```
both_data1 <- ints1 %>%
  inner_join(veg_data, by = "Stand", relationship = "many-to-many") %>%
  subset(select = -c(X, Year1, Site, Year, DoyStart)) %>%
  rename("Pollinator" = GenusSpecies, "Plant" = PlantGenusSpecies.y)

#Remove extra plant data
both_data1$PlantGenusSpecies.x <- NULL
#merge stand and sampling round
both_data1$Standsr <- paste(both_data1$Stand, both_data1$Transect, both_data1$Date, se
p = "_")

#Split all data into each stand
Stand_df <- split(both_data1, both_data1$Standsr, drop = FALSE)

# collect all Plants and Pollinator names:
all_plants <- unique(unlist(lapply(Stand_df, function(df) df$Plant)))
all_pollinators <- unique(unlist(lapply(Stand_df, function(df) df$Pollinator)))

# Make each community into adjacency matrix stored in a list - for loop retaining no o
f rows/col

WHOLE_LIST <- vector("list", length = length(Stand_df))
for(i in seq_along(Stand_df)){
 ThisMix <- Stand_df[[i]] %>%
  ungroup() %>%
  complete(Plant = all_plants, Pollinator = all_pollinators, fill = list(interactions
= 0)) %>%
  group_by(Pollinator, Plant) %>%
  summarize(interactions = n(), .groups = "drop") %>%
  pivot_wider(names_from = Pollinator, values_from = interactions, values_fill = 0) %
>%
  column_to_rownames( var = "Plant")
  ThisMix[is.na(ThisMix)] <- 0
  ThisMix[ThisMix == 1] <- 0
  WHOLE_LIST[[i]] <- ThisMix
  names(WHOLE_LIST) <- names(Stand_df)
}

nested_values <- vector(length = length(WHOLE_LIST))
for(i in seq_along(WHOLE_LIST)){
  this.mat <- as.matrix(WHOLE_LIST[[i]])
  empty(this.mat, count = FALSE)
  rownames(this.mat) <- rownames(WHOLE_LIST[[i]])
  nodf <- networklevel(this.mat, index = "NODF")
  nested_values[[i]] <- round(nodf, digits = 3)
}
```

```
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
## Warning in networklevel(this.mat, index = "NODF"): Web is really too small to
## calculate any reasonable index. You will get the values nonetheless, but I
## wouldn't put any faith in them!
```

```
nested_values_st <- paste(names(WHOLE_LIST), nested_values, sep = "_")
head(nested_values_st)
```

```
## [1] "604_Mix1_6/20/24_56.522" "604_Mix1_7/31/24_56.757"
## [3] "604_Mix2_5/23/24_56.41"  "604_Mix2_6/20/24_54.902"
## [5] "604_Mix2_7/31/24_NA"     "604_Mix3_5/23/24_NA"
```

```
saveRDS(WHOLE_LIST, file = "WHOLE_LIST.rds")


#WHOLE_LIST <- readRDS("/Users/isabellenollette/Desktop/final_community_assembly-SJIN/data/WHOLE_LIST.rds")
WHOLE_LIST <- readRDS("/Users/sydneyjames/OneDrive_UniversityofOregon/final_community_assembly-SJIN/data/WHOLE_LIST.rds")
```

#DATA MANIPULATION Using the bipartide package, we calculated network-level weighted nestedness for each plant pollinator interaction matrix . In addition, we used the function oecosimu which creates a null model, and compares each interaction matrix to give us a scaled comparable value to represent nestedness when investigating the relationship to reproductive productivity. Reproductive productivity was represented by the pollinator benefit of each plant, which was calculated by subtracting the seed set of open flowers from that of the closed flowers, indicating the reproductive benefit each flower gained from pollination. These values were averaged for each stand and mix combination, and for each mix across all stands, resulting in a final data frame containing standardized nestedness and pollinator benefit for each stand and mix combination. As Dr Ponisio taught us, we used clear commenting and chunks in our code to ensure the reproducibility of our analyses and cleaning, especially as beginner data scientists. This not only makes it easier for readers to understand the purpose of specific chunks, but really helped when we were working on different sections of the data, and needed to combine our work.

```
# set.seed(1)
# # vegan's oecosimu framework with quasiswap (fixed row & column sums)
# null_fun <- function(x) vegan::nestedtemp(x, index = "NODF", weighted = TRUE)$statis
tic
#
# #Null model simulation
# standnull_nested_values <- lapply(WHOLE_LIST, function(nullmodel) {
#   oecosimu(nullmodel, null_fun, method = "r2dtable", nsimul = 100)
# })
# names(standnull_nested_values) = names(WHOLE_LIST)
# #tranforming the list into a data frame containg the name data as well as null stand
ardized nestedness
# nested_vec <- sapply(standnull_nested_values, function(x) unname(x$statistic))
# nestedness_values_df <- tibble(
#   name = names(nested_vec),
#   null_nestedness = unlist(nested_vec)
# ) %>%
#   separate(name, into = c("stand", "mix", "date"), sep = "_", remove = TRUE)

#preparing seed data for joining
seed_count <- read.csv("/Users/sydneyjames/OneDrive_UniversityofOregon/final_community
_assembly-SJIN/data/All_Cleaned_Data")
#seed_count <- read.csv("/Users/isabellenollette/Desktop/final_community_assembly-SJI
N/data/All_Cleaned_Data")

# rename mixes to match pollinator/data
seed_count$Seed_Mix <- as.character(seed_count$Seed_Mix)
seed_count$Seed_Mix[seed_count$Seed_Mix == 1] <- "Mix1"
seed_count$Seed_Mix[seed_count$Seed_Mix == 2] <- "Mix2"
seed_count$Seed_Mix[seed_count$Seed_Mix == 3] <- "Mix3"
seed_count$Seed_Mix[seed_count$Seed_Mix == 4] <- "Mix4"
seed_count$Seed_Mix[seed_count$Seed_Mix == 5] <- "Mix5"
seed_count$Seed_Mix[seed_count$Seed_Mix == 6] <- "Mix6"

# rename plants to match pollinator/data (run each row separtely nd it works even if r
ed)
seed_count$PlantGenusSpecies[seed_count$PlantGenusSpecies == "CLAAMO"] <- "Clarkia amo
ena"
seed_count$PlantGenusSpecies[seed_count$PlantGenusSpecies ==  "CLAPUR" ] <- "Clarkia pu
rpurea"
seed_count$PlantGenusSpecies[seed_count$PlantGenusSpecies == "GILCAP"] <- "Gilia capit
ata"
seed_count$PlantGenusSpecies[seed_count$PlantGenusSpecies == "COLLIGRA"] <- "Collinsia
grandiflora"
seed_count$PlantGenusSpecies[seed_count$PlantGenusSpecies == "MADGRA"] <- "Madia graci
lis"

# rename columns and remove uneeded ones
seed_count1 <- seed_count %>%
  rename("Transect" = Seed_Mix, "Plant" = PlantGenusSpecies) %>%
  subset(select = -c(X, Treatment)) %>%
  filter(Plant != "MICGRA")

# paste stand and mix and split seed data by these combos
seed_count1$Standsr1 <- paste(seed_count1$Stand, seed_count1$Transect, sep = "_")
seed_count_open <- seed_count1 %>%
  group_by(Standsr1) %>%
  summarise(av_OSeeds_Per_Bloom = mean(OSeeds_Per_Bloom))

seed_count_closed <- seed_count1 %>%
  group_by(Standsr1) %>%
  summarise(av_CSeeds_Per_Bloom = mean(CSeeds_Per_Bloom))

#try averaging by mix
 seed_count_open_MIX <- seed_count1 %>%
   group_by(Transect) %>%
   summarise(av_OSeeds_Per_Bloom = mean(OSeeds_Per_Bloom))
 seed_count_closed_MIX <- seed_count1 %>%
   group_by(Transect) %>%
   summarise(av_CSeeds_Per_Bloom = mean(CSeeds_Per_Bloom))

##combine nestedness, standardized nestedness, seed data, and pollinator data
# separtate vector into unique stand and mix and create data frame then
df_nested <- tibble(
  name = nested_values_st) %>%
  separate(name, into = c("stand", "mix", "date", "nestedness"), sep = "_", remove = T
RUE)

# join null to actual
df_all <- df_nested %>%
inner_join(nestedness_values_df)
```

```
## Error: object 'nestedness_values_df' not found
```

```
# find average for every stand and mix combo by averaging nestedness of different date
s
nestedness_av_final <- df_all %>%
  group_by(stand, mix) %>%
  mutate(nestedness = as.numeric(nestedness)) %>%
  summarise(
    av_nestedness_actual = mean(nestedness, na.rm = TRUE),
    av_nestedness_null = mean(null_nestedness))
```

```
## Error: object 'df_all' not found
```

```
nestedness_av_final$Standsr1 <- paste(nestedness_av_final$stand, nestedness_av_final$m
ix, sep = "_")
```

```
## Error: object 'nestedness_av_final' not found
```

```
nestedness_av_final$stand <- NULL
```

```
## Error: object 'nestedness_av_final' not found
```

```
nestedness_av_final$mix <- NULL
```

```
## Error: object 'nestedness_av_final' not found
```

```
nestedness_av_final <- nestedness_av_final %>%
  filter(Standsr1 != "605_Mix2") # this row was ommitted in the seed data and thus is
extra
```

```
## Error: object 'nestedness_av_final' not found
```

```
write.csv(nestedness_av_final, "nestedness_av_final.csv", row.names = FALSE)
```

```
## Error in eval(expr, p): object 'nestedness_av_final' not found
```

```
av_nestedness_final <- read.csv("/Users/sydneyjames/OneDrive_UniversityofOregon/final_
community_assembly-SJIN/data/nestedness_av_final.csv")
#av_nestedness_final <- read.csv("/Users/isabellenollette/Desktop/final_community_asse
mbly-SJIN/data/nestedness_av_final.csv")

nestedness_av_final_byMIX <- #read.csv("/Users/isabellenollette/Desktop/final_communit
y_assembly-SJIN/data/nestedness_av_final_byMIX.csv")
read.csv("/Users/sydneyjames/OneDrive_UniversityofOregon/final_community_assembly-SJI
N/data/nestedness_av_final_byMIX.csv")

# join open seeds data and closed seed data
final_df1 <- right_join(av_nestedness_final, seed_count_open, by = "Standsr1")
final_df2 <- right_join(final_df1, seed_count_closed, by = "Standsr1")

# reorder columns for clarity
final_df <- final_df2 %>%
  select(Standsr1, av_OSeeds_Per_Bloom, av_CSeeds_Per_Bloom, av_nestedness_actual, av_
nestedness_null)

write.csv(final_df, "final_df", row.names = FALSE)

#doing the same thing with the by mix data

names(seed_count_open_MIX)[names(seed_count_open_MIX) == "Transect"] <- "mix"
names(seed_count_closed_MIX)[names(seed_count_closed_MIX) == "Transect"] <- "mix"
# join open seeds data and closed seed data
final_df1MIX <- right_join(nestedness_av_final_byMIX, seed_count_open_MIX, by = "mix")
final_df2MIX <- right_join(final_df1MIX, seed_count_closed_MIX, by = "mix")

# reorder columns for clarity
final_dfMIX <- final_df2MIX %>%
  select(mix, av_OSeeds_Per_Bloom, av_CSeeds_Per_Bloom, av_nestedness_actual, av_neste
dness_null)

write.csv(final_dfMIX, "final_dfMIX.csv", row.names = FALSE)
```

#ANALYSIS/HYPOTHESIS TESTING To evaluate the relationship between reproductive productivity and nestedness, we graphed our found values for each stand and mix combination with nestedness on the x axis and seed set on the y axis in scatter plot, and used a fitted linear model to find the line of best fit. The slope of this line represents the found relationship between nestedness and seed set, and our p-value tells us how accurately our line of best fit represents this relationship. We also used ANOVA to give us further information on the variation between mixes and how that may affect this relationship.

```
#final_data <- read.csv("/Users/isabellenollette/Desktop/final_community_assembly-SJI
N/data/final_df")
final_data <- read.csv("/Users/sydneyjames/OneDrive_UniversityofOregon/final_community
_assembly-SJIN/data/final_df")



final_data2 <- final_data %>%
  separate(Standsr1, into = c("Stand", "Mix"), sep = "_") %>%
  group_by(Mix) %>%
  mutate(
    PollinationBenefit = av_OSeeds_Per_Bloom - av_CSeeds_Per_Bloom
  )

lm_POLLINATORBENEFITvNESTEDNESS <- lm(PollinationBenefit ~ av_nestedness_null * Mix, d
ata = final_data2)
summary(lm_POLLINATORBENEFITvNESTEDNESS)
```

```
##
## Call:
## lm(formula = PollinationBenefit ~ av_nestedness_null * Mix, data = final_data2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -32.442  -4.723  -0.883   1.542  73.166
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  10.040     13.660   0.735    0.467
## av_nestedness_null           14.264     23.281   0.613    0.544
## MixMix2                     -14.412     22.579  -0.638    0.527
## MixMix3                      -2.450     19.833  -0.124    0.902
## MixMix4                     -12.940     18.601  -0.696    0.491
## MixMix5                     -10.262     19.244  -0.533    0.597
## MixMix6                      25.073     19.787   1.267    0.213
## av_nestedness_null:MixMix2  -10.436     35.798  -0.292    0.772
## av_nestedness_null:MixMix3  -13.854     31.142  -0.445    0.659
## av_nestedness_null:MixMix4   -0.765     35.310  -0.022    0.983
## av_nestedness_null:MixMix5  -13.815     28.512  -0.485    0.631
## av_nestedness_null:MixMix6  -42.736     32.914  -1.298    0.203
##
## Residual standard error: 21.37 on 35 degrees of freedom
## Multiple R-squared:  0.2124, Adjusted R-squared:  -0.03516
## F-statistic: 0.858 on 11 and 35 DF,  p-value: 0.5869
```

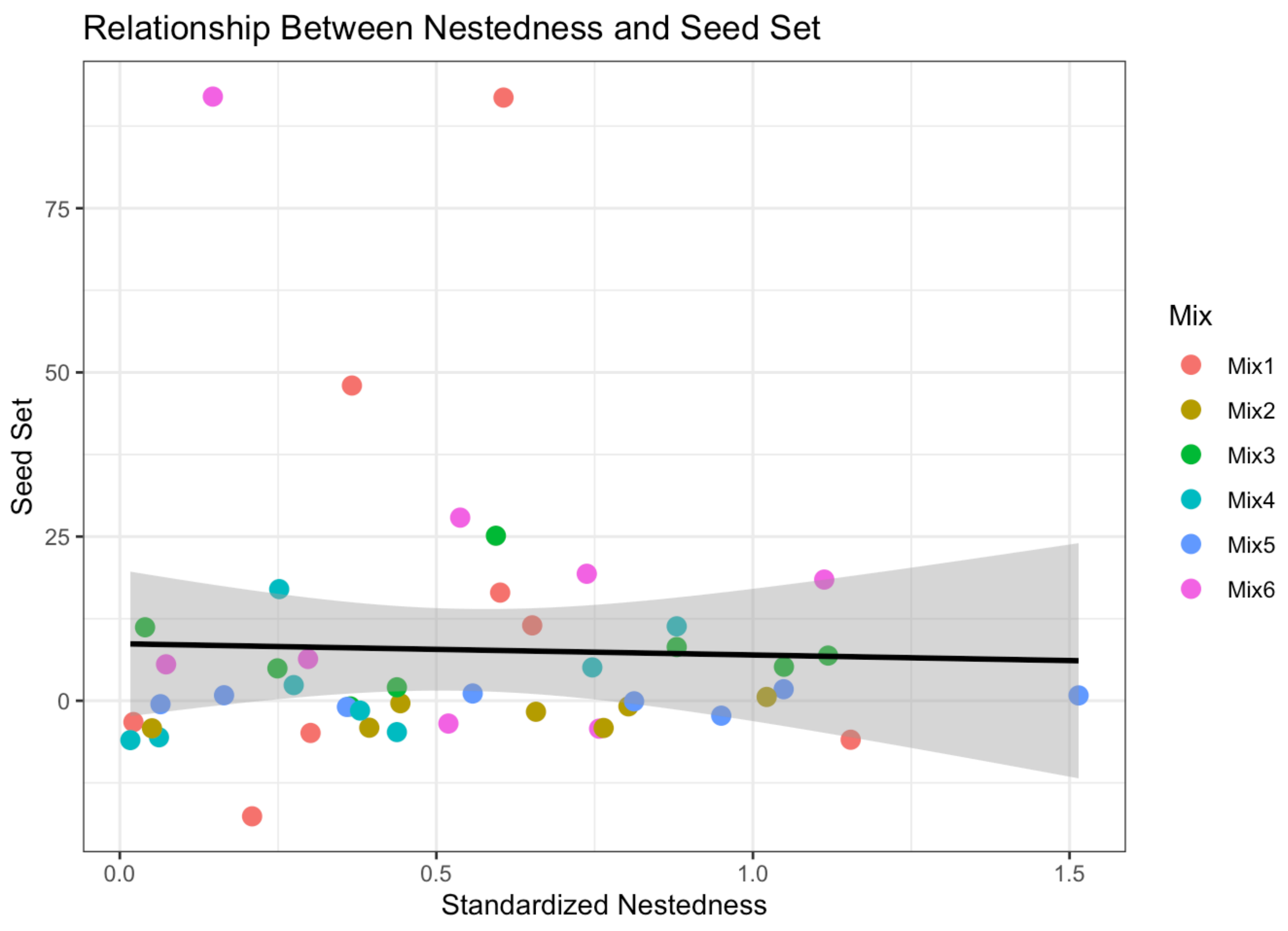
```
anova(lm_POLLINATORBENEFITvNESTEDNESS)
```

```
## Analysis of Variance Table
##
## Response: PollinationBenefit
##                        Df  Sum Sq Mean Sq F value Pr(>F)
## av_nestedness_null      1    18.0   17.95  0.0393 0.8440
## Mix                     5  3308.3  661.67  1.4492 0.2313
## av_nestedness_null:Mix  5   982.7  196.54  0.4305 0.8242
## Residuals              35 15980.1  456.58
```

#VISUALISATIONS The final combined data was separated into stand and mix, and pollinator benefit was calculated for each of these unique combinations. Pollinator benefit = (average number of open seeds per bloom) - (average number of closed seeds per bloom). Nestedness was then plotted against pollinator benefit and labelled as seed set using the ggplot2 package, and the data were put through an ANOVA analysis to summaries key statistical figures of this relationship. The same method was conducted for a second visualisation; however, the data were grouped by mix, and pollinator benefit was calculated for each mix across all stands. Our final visualisation comprises of only the mixes with the highest and lowest values of nestedness and seed set to see if a trend was observed. In each visualisation, mix was identified by color.

```
library(ggplot2)
ggplot(final_data2, aes(x = av_nestedness_null, y = PollinationBenefit, color = Mix))
+
  geom_point(size = 3) +
  geom_smooth(method = "lm", se = TRUE, color = "black") +
  theme_bw() +
  labs(
    title = "Relationship Between Nestedness and Seed Set",
    x = "Standardized Nestedness",
    y = "Seed Set"
  )
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

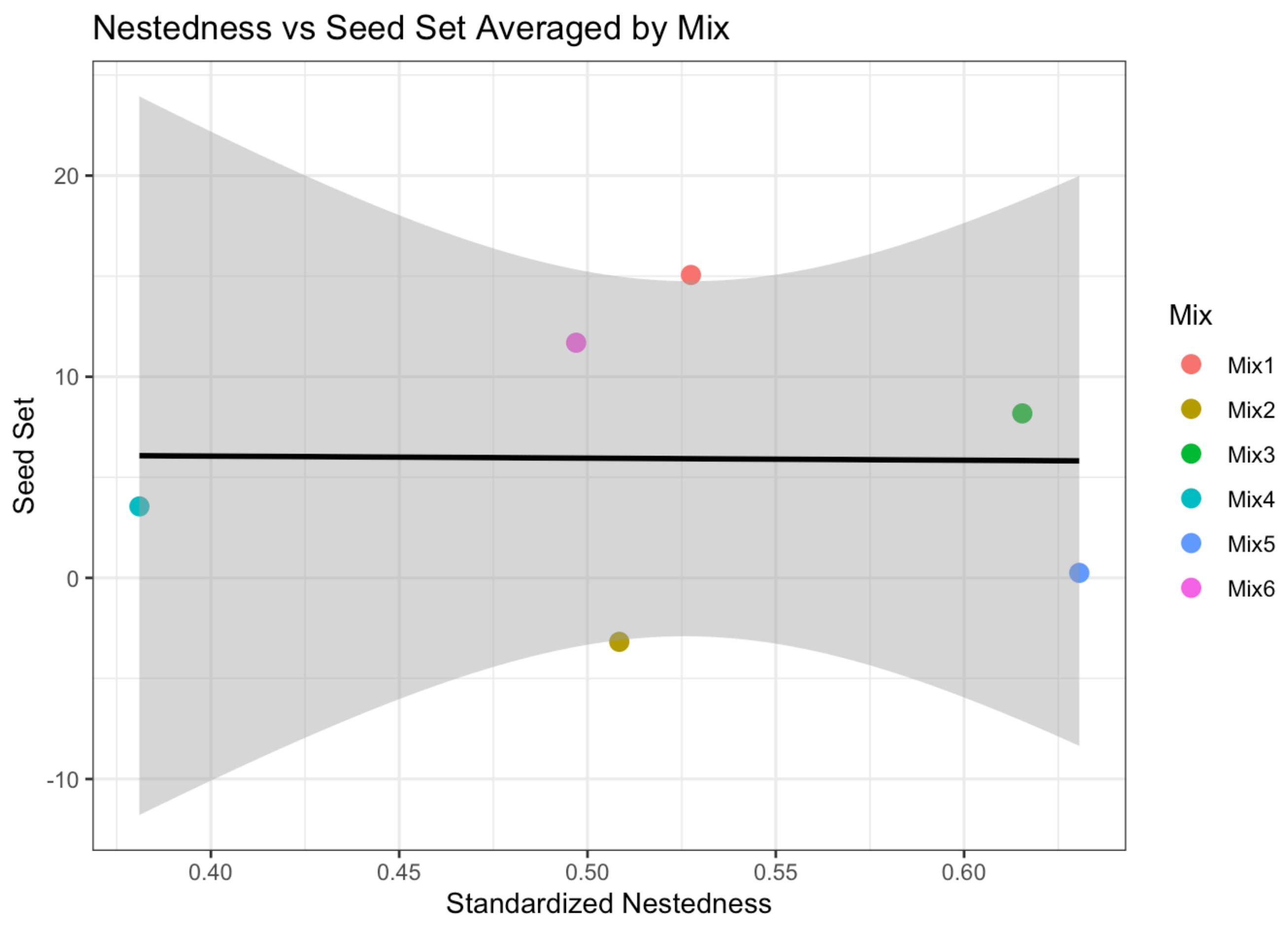


```
ggsave("nestednessVseedset_bymix.png", width = 6, height = 4, dpi = 300)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
#final_dfMIX <- read.csv("/Users/isabellenollette/Desktop/final_community_assembly-SJI
N/data/final_dfMIX.csv")
final_dfMIX <- read.csv("/Users/sydneyjames/OneDrive_UniversityofOregon/final_communit
y_assembly-SJIN/data/final_dfMIX.csv")

final_dfMIX <- final_dfMIX %>%
  mutate(
    PollinationBenefit = av_OSeeds_Per_Bloom - av_CSeeds_Per_Bloom,
    Mix = mix
  )

ggplot(final_dfMIX, aes(x = av_nestedness_null, y = PollinationBenefit, color = Mix))
+
  geom_point(size = 3) +
  geom_smooth(method = "lm", se = TRUE, color = "black") +
  theme_bw() +
  labs(
    title = "Nestedness vs Seed Set Averaged by Mix",
    x = "Standardized Nestedness",
    y = "Seed Set"
  )
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

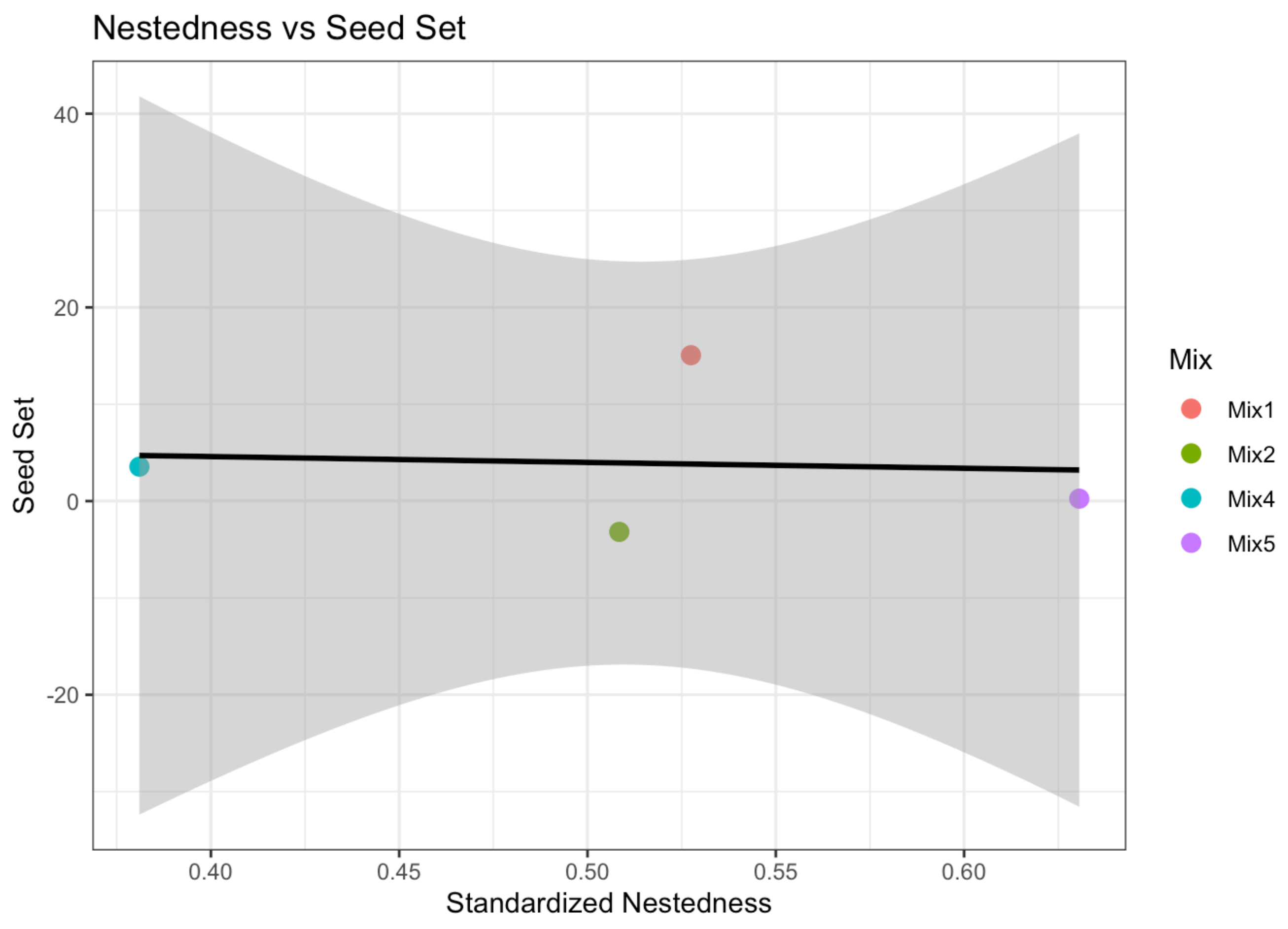


```
ggsave("nestednessVseedset.png", width = 6, height = 4, dpi = 300)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
# just plot mix 4 and 5 (highest and lowest nestedness)

final_dfMIX_edit <- final_dfMIX %>%
  mutate(
    PollinationBenefit = av_OSeeds_Per_Bloom - av_CSeeds_Per_Bloom,
    Mix = mix
  ) %>%
  filter(mix == "Mix4" | mix == "Mix5" | mix == "Mix1" | mix == "Mix2")

ggplot(final_dfMIX_edit, aes(x = av_nestedness_null, y = PollinationBenefit, color = M
ix)) +
  geom_point(size = 3) +
  geom_smooth(method = "lm", se = TRUE, color = "black") +
  theme_bw() +
  labs(
    title = "Nestedness vs Seed Set",
    x = "Standardized Nestedness",
    y = "Seed Set"
  )
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

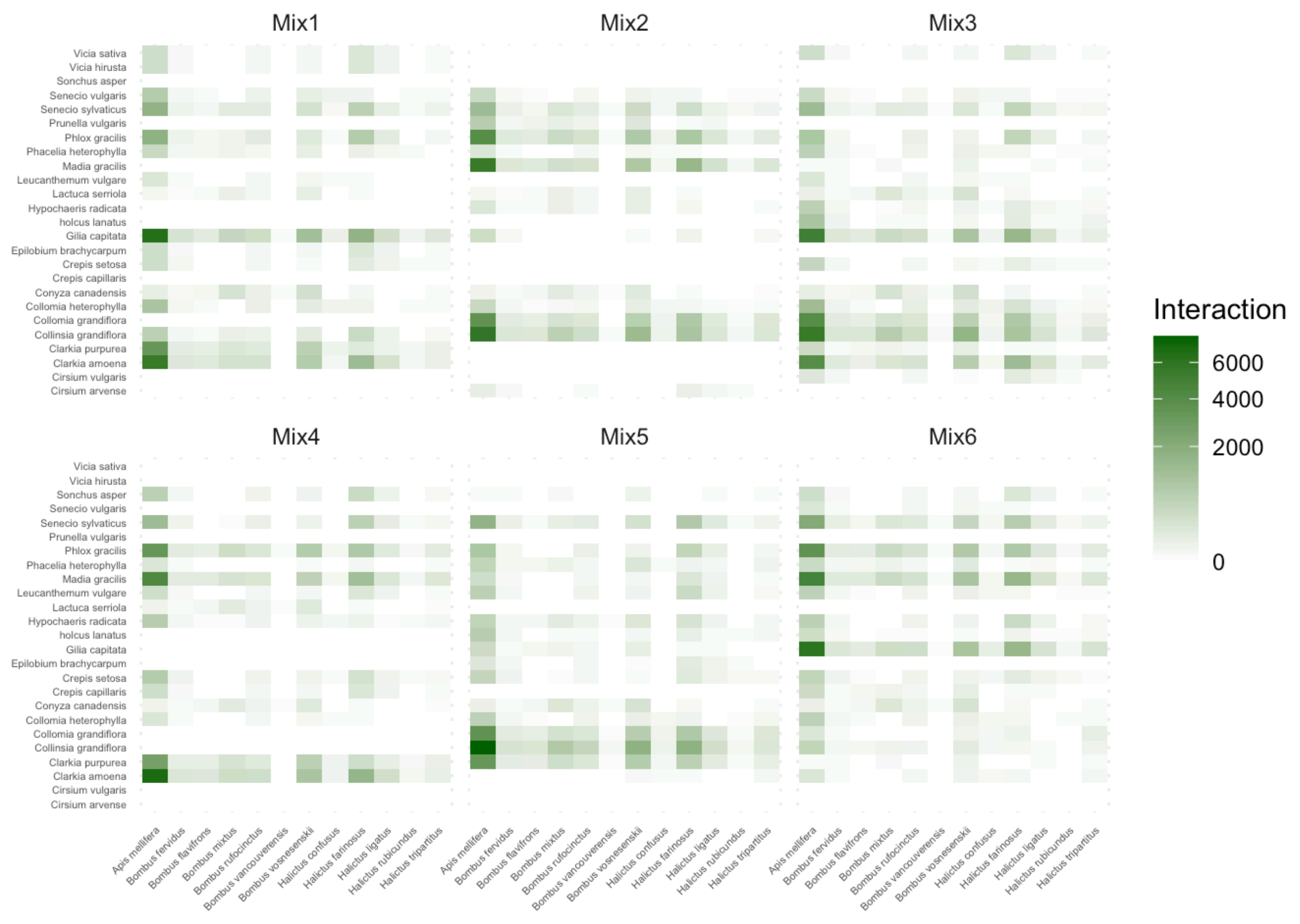


```
ggsave("nestednessVseedset_bymix.png", width = 6, height = 4, dpi = 300)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

We also created a heat map using our list of plant pollinator adjacency matrices to create a visual representation of nesting patterns within each mix.

```
#WHOLE_LIST <- readRDS("/Users/isabellenollette/Desktop/final_community_assembly-SJIN/
data/WHOLE_LIST.rds")
WHOLE_LIST <- readRDS("/Users/sydneyjames/OneDrive_UniversityofOregon/final_community_
assembly-SJIN/data/WHOLE_LIST.rds")

library(stringr)
library(ggplot2)
library(purrr)

mix_vector <- str_split(names(WHOLE_LIST), "_", simplify = TRUE)[,2]
list_by_mix <- split(WHOLE_LIST, mix_vector)

agg_by_mix <- lapply(list_by_mix, function(sublist) {
  Reduce("+", sublist)   # adding adjacency matrices together
})

plot_all_mixes_heatmap <- function(mat_df) {
  long_all <- map2_dfr(mat_df, names(mat_df), ~ .x %>%
                         rownames_to_column("Plant") %>%
                         pivot_longer(
                           cols = -Plant,
                           names_to = "Pollinator",
                           values_to = "value"
                         ) %>%
                         mutate(Mix = .y))

  ggplot(long_all, aes(x = Pollinator, y = Plant, fill = value)) +
    geom_tile() +
    scale_fill_gradient(low = "white", high = "darkgreen", trans = scales::sqrt_trans
()) +
    labs(fill = "Interaction") +
    facet_wrap(~Mix, ncol = 3) +   # adjust ncol to control layout
    theme_minimal() +
    theme(
      axis.text.x = element_text(angle = 45, hjust = 1, size = 4),
      axis.text.y = element_text(size = 4),
      axis.title = element_blank()
    )
}

plot_all_mixes_heatmap(agg_by_mix)
```



```
ggsave("heatmap_bymix.png", width = 6, height = 4, dpi = 300)
```

#CONCLUSION It was found that there is no statistically significant relationship between community nestedness and reproductive output as indicated by the p-values. Specifically, it was expected that mix 1 and 2 would produce opposite levels of nestedness, however, as evident in figure 3, their nestedness values are almost equal, despite high and low reproductive success, respectively. Our first hypothesis that higher specialized communities will be the least nested was not supported. If this hypothesis was supported, the results would illustrate a negative relationship between seed set and nestedness, however as no such trend was observed, we accept our null hypothesis, representing a lack of correlation between our variables: nestedness and reproductive success.Our second hypothesis that greater abundance of generalist pollinators will inhibit reproductive success yet promote community nestedness was also unsupported. While high nestedness was observed in Mix 1, the community assembly with only generalists, an extremely high p-value of These results may be confounded by: Competition as it relates to co-existence theory: where the distribution of biological resources had a greater effect on seed set than pollination Human error during sampling: data were very inconsistent between stands, mixes and plants, impacting precision Human error during sample processing: data manipulation and results at the detriment to accuracy of data sets which were incomplete and inconsistently labelled. Differences in sorting good seeds and bad seeds in sample processing: our analysis used only good seeds in counting total seed set, however determining good and bad seeds is up to the counter's discretion, thus allowing room for individual biases. Nestedness precision: our data analysis gave us a nestedness value for the overall community of each stand, mix, and date combination. However our seed data is more precise providing us with reproductive productivity for each plant in each stand mix combination. Running a different calculation to give us each plants contribution to nestedness would give us a more precise value to compare to seed set.Scaling by bloom: Another factor in our final data could have been our lack of consideration of bloom count for each plant in our seed comparisons. each species has a different number of blooms and we did not take that into consideration and scale our seed data which may have made our data across species less comparable. Resilience: Mix 5 was found to have the highest nestedness and therefore, by the standards of Brosi (2006), may be the most resilient community in the face of a changing climate in temperate regions and therefore the most sustainable for restoration efforts. Conversely, mix 4 was found to have the lowest nestedness. By these same standards, the resilience of this community may be low, and planting this particular assembly could be detrimental to restoration efforts. Despite being unable to support any hypothesis or prove a relationship exists between reproductive success and output, there is an inferred difference between community assemblages in their reproductive productivity, hence supporting existing literature and knowledge of community ecology that habitat filters exist and there are abiotic and biotic factors that impact the establishment of communites. Thus, our third and final hypothesis predicting optimal community assembly can not be inferred with statistical significance, and more, long-term research may need to be done to find a relationship. Alternatively, a correlation between nestedness and reproductive output may not exist in degraded wetland prairie ecosystems.

FINAL THOUGHTS This project was hard. Before starting Data Camp tutorials in the weeks leading up to this class, neither of us had ever coded before, and we were instantly inundated with this new language from the get-go. However, like every new language, the best way to learn is immersion, and boy, did we do that. Hardly making the weekly lab submission deadline was hard enough, and signing ourselves up to take on an unpublished, raw data set was an even larger feat, one we had no idea would make us lose our minds. There is no need to explain the challenges, as those are clear in our code, but we do want to explain, and more so show gratitude, for the things we learned, as upcoming scientists in both natural and data sciences: -Label your samples - when collecting samples in the field, ensure that everybody collecting is consistently labelling their envelopes/bags, and each sample has the maximum amount of information on it. This is the metadata of the sample, and while some of it may seem redundant at the time of collection, you never know who will use it and when it will be used. If there are multiple formats of the samples, such as digital and physical copies, ensure they are exactly the same. Here, it is also critical to label all columns, sample IDs, dates and species names, for example, in the EXACT same way (seeds and Seeds are different!). This also applies to renaming columns while coding, and ensuring all collaborators are using a consistent style. -Fill in NAs - there will always be gaps in data, especially when collecting plants and pollinators who have proven to be unreliable and inconsistent. Sometimes flowers don’t bloom, or stems get eaten, but that information is useless if there is no explanation in the data to explain why there is an “NA”. If this information is available, those analysing the data can use their understanding of the context to either remove these values, turn them to zeros, or literally anything else that is more helpful than an anonymous NA. -Use #s for organization - like we mentioned before, commenting before each section or important line of code really helped us through the collaborative coding process. We were able to troubleshoot efficiently, replicate code from different sections, and understand our analysis multiple days later. But what’s more, is the use of ## or even ###, in addition to #, which we used as a tool during the initial stages of coding, to separate sections and future intentions with our code, or just create a practical outline for the workflow if we didn’t have time to actually start coding. -Github is a b***** - as we struggled through for loops and ggplot, we arguably struggled more with Github. We learned the importance of using separate rmd files to write code, and combining them later, rather than working on the same document. Don’t work on the same file at the same time. This pull push method makes a lot of sense for people working with temporal and geographical, however, while we sat next to eachother in the science library, using one computer was the most effective way to collaboratively code. We also found outstanding evidence that the commit message is really important, and while it seems tedious, it can recover hours of lost work.

These final thoughts are just a little extra optional reading to cap off our project; however, we want to emphasize how much of a learning curve we both had in this class both inside and outside of the prescribed course content. We learned how to be better scientists, better teammates and better critical thinkers. We learned the importance of grit and persistence even in the face of difficulty as well as how to ask the right questions and the importance of seeking help, and of course we learned how gratifying and rewarding it can be to struggle with your code and find success even in nonsignificant results. We wanted to thank both our professor Lauren Ponisio and our GE Rebecca Hayes for all of their patience, support, and guildance throughout this project and we surely will think of your both fondly whenever we use our data science skills in the future.

Bell, John M., Jeffrey D. Karron, and Randall J. Mitchell. 2005. “INTERSPECIFIC COMPETITION FOR POLLINATION LOWERS SEED PRODUCTION AND OUTCROSSING IN MIMULUS RINGENS.” *Ecology* 86 (3): 762–71. https://doi.org/10.1890/04-0694 (https://doi.org/10.1890/04-0694).

Brosi, Berry J. 2016. “Pollinator Specialization: From the Individual to the Community.” *New Phytologist* 210 (4): 1190–94. https://doi.org/10.1111/nph.13951 (https://doi.org/10.1111/nph.13951).

Memtsas, G. I., M. Lazarina, S. P. Sgardelis, T. Petanidou, and A. S. Kallimanis. 2022. “What Plant–Pollinator Network Structure Tells Us about the Mechanisms Underlying the Bidirectional Biodiversity Productivity Relationship?” *Basic and Applied Ecology* 63 (September): 49–58. https://doi.org/10.1016/j.baae.2022.05.006 (https://doi.org/10.1016/j.baae.2022.05.006).

Moeller, David A. 2005. “Pollinator Community Structure and Sources of Spatial Variation in Plant?pollinator Interactions in Clarkia Xantiana Ssp. Xantiana.” *Oecologia* 142 (1): 28–37. https://doi.org/10.1007/s00442-004-1693-1 (https://doi.org/10.1007/s00442-004-1693-1).

Ponisio, Lauren C., Marilia P. Gaiarsa, and Claire Kremen. 2017. “Opportunistic Attachment Assembles Plant–Pollinator Networks.” Edited by Dominique Gravel. *Ecology Letters* 20 (10): 1261–72. https://doi.org/10.1111/ele.12821 (https://doi.org/10.1111/ele.12821).

Sargent, Risa D., and David D. Ackerly. 2008. “Plant–Pollinator Interactions and the Assembly of Plant Communities.” *Trends in Ecology & Evolution* 23 (3): 123–30. https://doi.org/10.1016/j.tree.2007.11.003 (https://doi.org/10.1016/j.tree.2007.11.003).

Slingsby, Jasper A., and G. Anthony Verboom. 2006. “Phylogenetic Relatedness Limits Co-occurrence at Fine Spatial Scales: Evidence from the Schoenoid Sedges (Cyperaceae: Schoeneae) of the Cape Floristic Region, South Africa.” *The American Naturalist* 168 (1): 14–27. https://doi.org/10.1086/505158 (https://doi.org/10.1086/505158).