Indices based on rotated principal components

Luciano L.M. De Benedictis

2026-01-07

```
# setup ------------------------------------------------------------------

library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.6
## v forcats   1.0.1     v stringr   1.6.0
## v ggplot2   4.0.1     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.2
## v purrr     1.2.0
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(GET)
library(vegan)
```

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

```
library(psych)
```

```
##
## Attaching package: 'psych'
##
## The following object is masked from 'package:vegan':
##
##     pca
##
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
resampled <- readRDS("transect_resampled.rds")
speciestraits <- readRDS("traits_imputation.rds")
source("0.functions.R")

indicesaxes <- vector(mode = "list")
curvesaxes <- vector(mode = "list")
```

```r
# transform traits ------------------------------------------------------

data <- speciestraits |>
  select(species, where(is.numeric)) |>
  select(-n) |>
  mutate(across(where(is.numeric),  ~ boxcox(.))) |>
  as.data.frame()
```

```
## Box-Cox transform with lambda = 0.1
## Box-Cox transform with lambda = -0.5
## Box-Cox transform with lambda = -0.4
## Box-Cox transform with lambda = 0.3
## Box-Cox transform with lambda = 0.2
## Box-Cox transform with lambda = 0
## Box-Cox transform with lambda = 0.8
## Box-Cox transform with lambda = 1.8
## Box-Cox transform with lambda = -0.2
## Box-Cox transform with lambda = 0
```

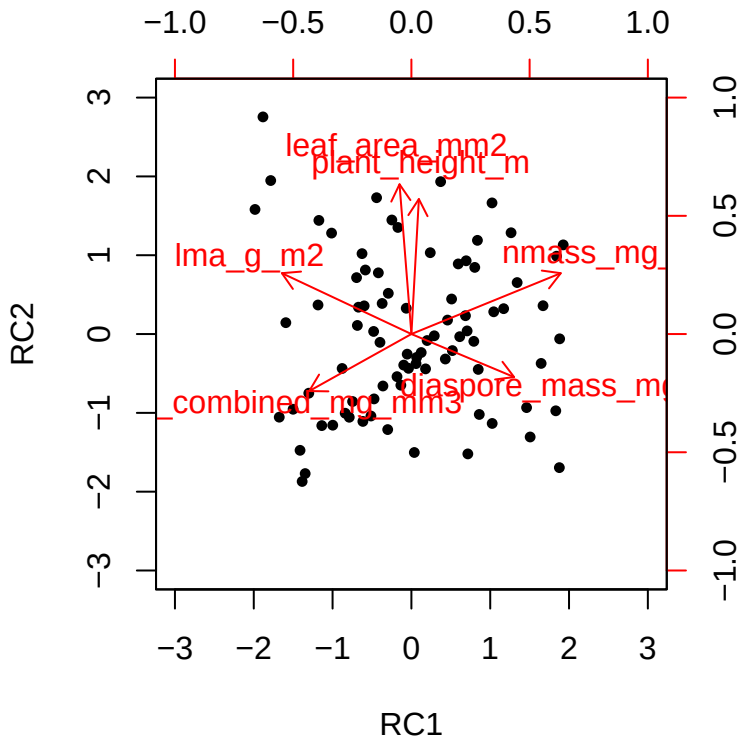
```r
# check dimensions ------------------------------------------------------

PCA_above <- data |>
  select(2:7) |>
  psych::principal(nfactors = 2, rotate = "varimax")

PCA_above
```

```
## Principal Components Analysis
## Call: psych::principal(r = select(data, 2:7), nfactors = 2, rotate = "varimax")
## Standardized loadings (pattern matrix) based upon correlation matrix
##                       RC1   RC2   h2   u2 com
## leaf_area_mm2       -0.06  0.79 0.63 0.37 1.0
## nmass_mg_g           0.79  0.32 0.73 0.27 1.3
## lma_g_m2            -0.68  0.32 0.57 0.43 1.4
## plant_height_m       0.04  0.71 0.51 0.49 1.0
## diaspore_mass_mg     0.54 -0.23 0.35 0.65 1.3
## ssd_combined_mg_mm3 -0.54 -0.30 0.38 0.62 1.6
##
##                        RC1  RC2
## SS loadings           1.68 1.49
## Proportion Var        0.28 0.25
## Cumulative Var        0.28 0.53
## Proportion Explained  0.53 0.47
## Cumulative Proportion 0.53 1.00
##
## Mean item complexity =  1.3
## Test of the hypothesis that 2 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0.17
##  with the empirical chi square  67.8  with prob <  6.6e-14
##
## Fit based upon off diagonal values = 0.38
```

```
png('plots/PCA_above.png', bg = "white", width = 7, height = 7, units = "in",
    res = 1000)
biplot(PCA_above, main = NULL)
dev.off()
```

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

```
biplot(PCA_above, main = NULL)
```




```
data <- cbind(data, PCA_above$scores) |>
  rename(above1 = RC1, above2 = RC2)

PCA_clonal <- data |>
  select(8:11) |>
  psych::principal(nfactors = 2, rotate = "varimax")

PCA_clonal
```

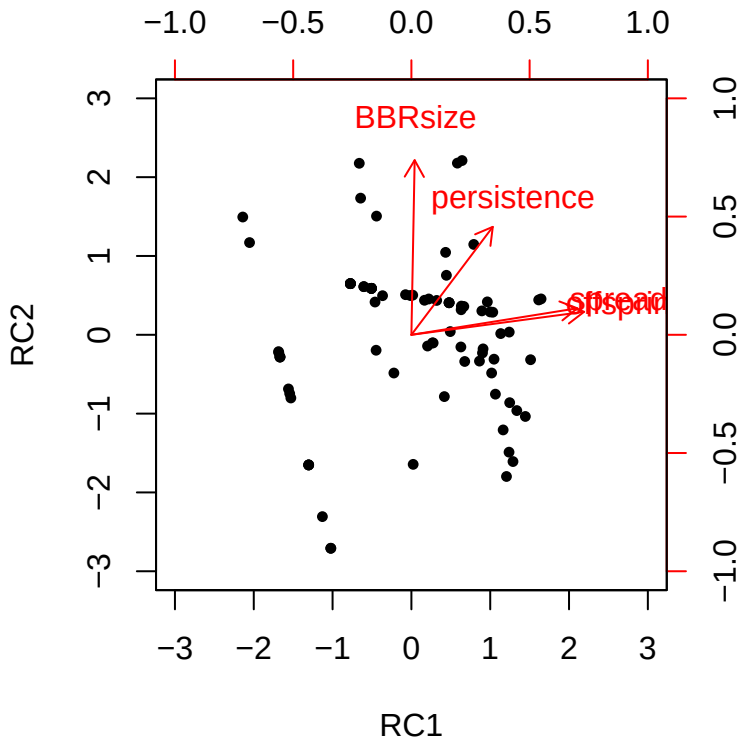
```
## Principal Components Analysis
## Call: psych::principal(r = select(data, 8:11), nfactors = 2, rotate = "varimax")
## Standardized loadings (pattern matrix) based upon correlation matrix
##              RC1  RC2   h2   u2 com
## BBRsize     0.02 0.92 0.85 0.15 1.0
```

```
## persistence 0.43 0.57 0.51 0.49 1.9
## offspring   0.91 0.12 0.85 0.15 1.0
## spread      0.88 0.14 0.79 0.21 1.1
##
##                        RC1  RC2
## SS loadings           1.79 1.21
## Proportion Var        0.45 0.30
## Cumulative Var        0.45 0.75
## Proportion Explained  0.60 0.40
## Cumulative Proportion 0.60 1.00
##
## Mean item complexity =  1.2
## Test of the hypothesis that 2 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0.15
##  with the empirical chi square  21.43  with prob <  NA
##
## Fit based upon off diagonal values = 0.85
```

```
png('plots/PCA_clonal.png', bg = "white", width = 7, height = 7, units = "in",
    res = 1000)
biplot(PCA_clonal, main = NULL)
dev.off()
```

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

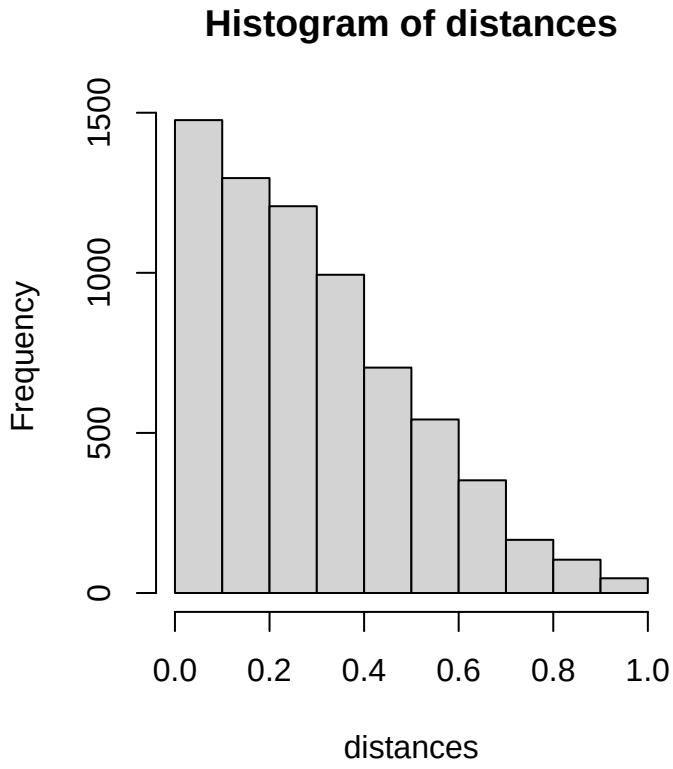
```
biplot(PCA_clonal, main = NULL)
```




```
data <- cbind(data, PCA_clonal$scores) |>
  rename(clonal1 = RC1, clonal2 = RC2)


# aboveground-------------------------------------------------------------

## aboveground RC1 -----
### calculate distances ------------------------------------------------

distances <- data |>
  select(above1) |>
  dist(method = "euclidean", diag = T, upper = T) |>
  as.matrix()

#scaling to 0-1

distances <- distances / max(distances)

#double checking the squared euclidean property

ade4::is.euclid(as.dist(sqrt(distances)))
```

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

```
#check distance distribution

hist(distances)
```




```
#pairs with distance 0

as.data.frame(as.table(distances)) |>
  filter(Freq == 0 & Var1 != Var2)
```

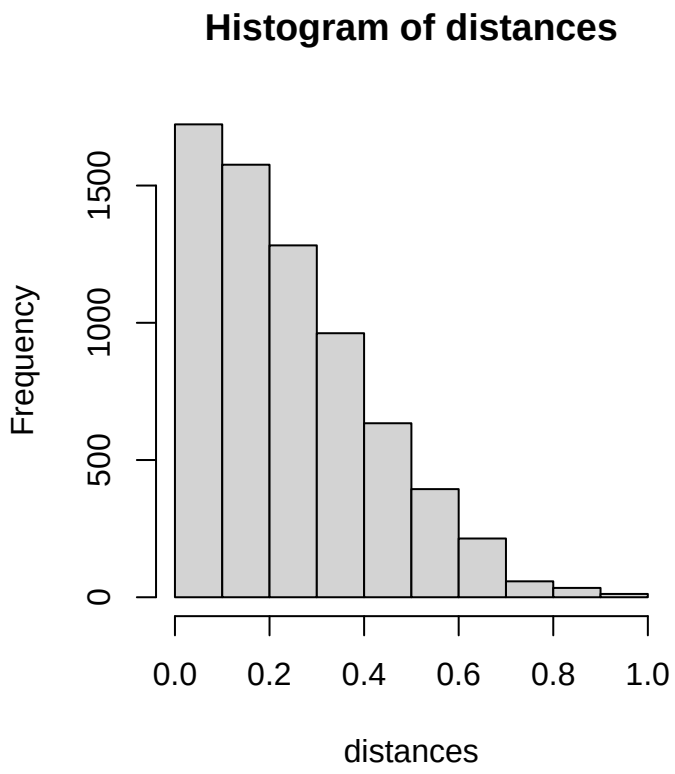
```
## [1] Var1 Var2 Freq
## <0 rows> (or 0-length row.names)
```

```
### calculate indices ------------------------------------------------

species <- data$species

raotrait <- lapply(resampled, function (x)
  lapply(x, function (y) qdecomp(y, distances, species)) |>
    bind_rows(.id = "step") |>
    mutate(step = as.numeric(step))
) |>
  bind_rows(.id = "site")

#scale steps in units of area
```

```
raotrait <- raotrait |>
  mutate(step = step / 100)

#add to list
indicesaxes[[1]] <- raotrait
names(indicesaxes)[1] <- "above1"

### create curve set ------------------------------------------------------

curvesets <- vector(mode = "list", length = sum(map_lgl(raotrait, is.numeric))-1)

for (i in seq_along(curvesets)){
  column <- colnames(raotrait)[i+2]
  curves <- raotrait |>
    select(site, step, {{column}}) |>
    pivot_wider(names_from = site, values_from = 3) |>
    filter(if_all(where(is.numeric), ~ !is.na(.)))
  curvesets[[i]] <- curve_set(obs = as.data.frame(curves[-1]), r = curves[[1]])
  names(curvesets)[i] <- column
}

curvesaxes[[1]] <- curvesets
names(curvesaxes)[1] <- "above1"

## aboveground RC2 -----
### calculate distances ---------------------------------------------------

distances <- data |>
  select(above2) |>
  dist(method = "euclidean", diag = T, upper = T) |>
  as.matrix()

#scaling to 0-1

distances <- distances / max(distances)

#double checking the squared euclidean property

ade4::is.euclid(as.dist(sqrt(distances)))
```

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

```
#check distance distribution

hist(distances)
```




```
#pairs with distance 0

as.data.frame(as.table(distances)) |>
  filter(Freq == 0 & Var1 != Var2)
```

```
## [1] Var1 Var2 Freq
## <0 rows> (or 0-length row.names)
```

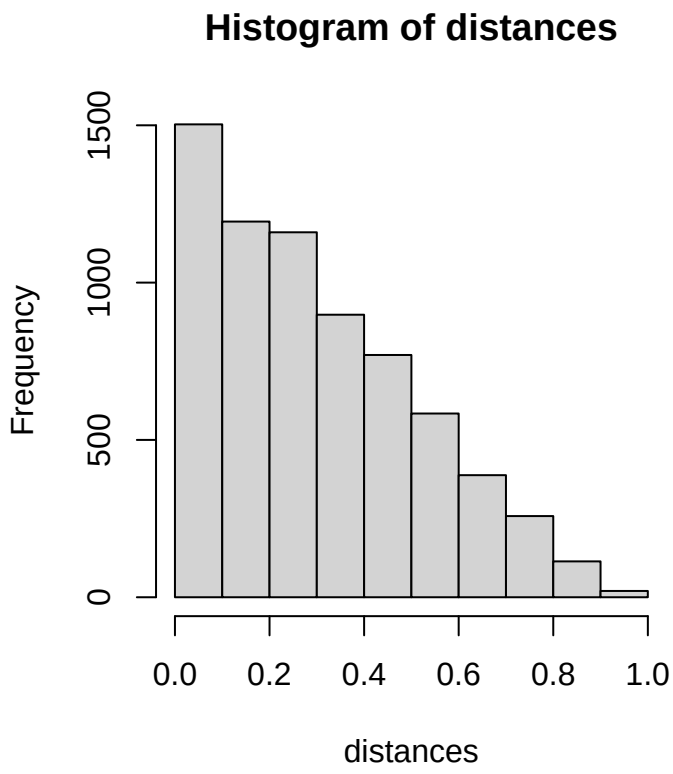
```
### calculate indices ----------------------------------------------------

species <- data$species

raotrait <- lapply(resampled, function (x)
  lapply(x, function (y) qdecomp(y, distances, species)) |>
    bind_rows(.id = "step") |>
    mutate(step = as.numeric(step))
) |>
  bind_rows(.id = "site")

#scale steps in units of area

raotrait <- raotrait |>
  mutate(step = step / 100)

#add to list
```

```r
indicesaxes[[2]] <- raotrait
names(indicesaxes)[2] <- "above2"

### create curve set ------------------------------------------------------

curvesets <- vector(mode = "list", length = sum(map_lgl(raotrait, is.numeric))-1)

for (i in seq_along(curvesets)){
  column <- colnames(raotrait)[i+2]
  curves <- raotrait |>
    select(site, step, {{column}}) |>
    pivot_wider(names_from = site, values_from = 3) |>
    filter(if_all(where(is.numeric), ~ !is.na(.)))
  curvesets[[i]] <- curve_set(obs = as.data.frame(curves[-1]), r = curves[[1]])
  names(curvesets)[i] <- column
}

curvesaxes[[2]] <- curvesets
names(curvesaxes)[2] <- "above2"

# clonal ------------------------------------------------------------------

## clonal RC1 -----
### calculate distances ---------------------------------------------------

distances <- data |>
  select(clonal1) |>
  dist(method = "euclidean", diag = T, upper = T) |>
  as.matrix()

#scaling to 0-1

distances <- distances / max(distances)

#double checking the squared euclidean property

ade4::is.euclid(as.dist(sqrt(distances)))
```

```
## Warning in ade4::is.euclid(as.dist(sqrt(distances))): Zero distance(s)
```

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

```r
#check distance distribution

hist(distances)
```




```
#pairs with distance 0

as.data.frame(as.table(distances)) |>
  filter(Freq == 0 & Var1 != Var2)
```

```
##    Var1 Var2 Freq
## 1    10    1    0
## 2    29    1    0
## 3    44    1    0
## 4    49    1    0
## 5    67    1    0
## 6     1   10    0
## 7    29   10    0
## 8    44   10    0
## 9    49   10    0
## 10   67   10    0
## 11   19   15    0
## 12   37   15    0
## 13   34   17    0
## 14   75   17    0
## 15   15   19    0
## 16   37   19    0
## 17   48   28    0
## 18   66   28    0
## 19    1   29    0
```

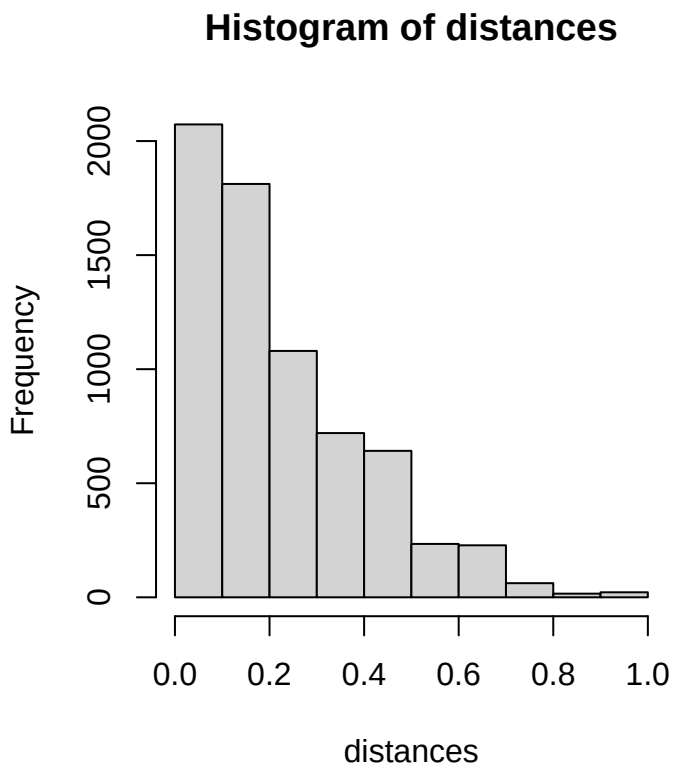
```
## 20   10   29    0
## 21   44   29    0
## 22   49   29    0
## 23   67   29    0
## 24   35   31    0
## 25   36   31    0
## 26   40   31    0
## 27   77   31    0
## 28   17   34    0
## 29   75   34    0
## 30   31   35    0
## 31   36   35    0
## 32   40   35    0
## 33   77   35    0
## 34   31   36    0
## 35   35   36    0
## 36   40   36    0
## 37   77   36    0
## 38   15   37    0
## 39   19   37    0
## 40   61   39    0
## 41   31   40    0
## 42   35   40    0
## 43   36   40    0
## 44   77   40    0
## 45    1   44    0
## 46   10   44    0
## 47   29   44    0
## 48   49   44    0
## 49   67   44    0
## 50   28   48    0
## 51   66   48    0
## 52    1   49    0
## 53   10   49    0
## 54   29   49    0
## 55   44   49    0
## 56   67   49    0
## 57   39   61    0
## 58   28   66    0
## 59   48   66    0
## 60    1   67    0
## 61   10   67    0
## 62   29   67    0
## 63   44   67    0
## 64   49   67    0
## 65   17   75    0
## 66   34   75    0
## 67   31   77    0
## 68   35   77    0
## 69   36   77    0
## 70   40   77    0
## 71   82   80    0
## 72   80   82    0
```

```r
### calculate indices ------------------------------------------------------

species <- data$species

raotrait <- lapply(resampled, function (x)
  lapply(x, function (y) qdecomp(y, distances, species)) |>
    bind_rows(.id = "step") |>
    mutate(step = as.numeric(step))
) |>
  bind_rows(.id = "site")

#scale steps in units of area

raotrait <- raotrait |>
  mutate(step = step / 100)

#add to list
indicesaxes[[3]] <- raotrait
names(indicesaxes)[3] <- "clonal1"

### create curve set ------------------------------------------------------

curvesets <- vector(mode = "list", length = sum(map_lgl(raotrait, is.numeric))-1)

for (i in seq_along(curvesets)){
  column <- colnames(raotrait)[i+2]
  curves <- raotrait |>
    select(site, step, {{column}}) |>
    pivot_wider(names_from = site, values_from = 3) |>
    filter(if_all(where(is.numeric), ~ !is.na(.)))
  curvesets[[i]] <- curve_set(obs = as.data.frame(curves[-1]), r = curves[[1]])
  names(curvesets)[i] <- column
}

curvesaxes[[3]] <- curvesets
names(curvesaxes)[3] <- "clonal1"

## clonal RC2 -----
### calculate distances ------------------------------------------------------

distances <- data |>
  select(clonal2) |>
  dist(method = "euclidean", diag = T, upper = T) |>
  as.matrix()

#scaling to 0-1

distances <- distances / max(distances)

#double checking the squared euclidean property

ade4::is.euclid(as.dist(sqrt(distances)))
```

```
## Warning in ade4::is.euclid(as.dist(sqrt(distances))): Zero distance(s)
```

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

```
#check distance distribution

hist(distances)
```




```
#pairs with distance 0

as.data.frame(as.table(distances)) |>
  filter(Freq == 0 & Var1 != Var2)
```

```
##    Var1 Var2 Freq
## 1    10    1    0
## 2    29    1    0
## 3    44    1    0
## 4    49    1    0
## 5    67    1    0
## 6     1   10    0
## 7    29   10    0
## 8    44   10    0
## 9    49   10    0
## 10   67   10    0
## 11   19   15    0
```

```
## 12   37   15    0
## 13   34   17    0
## 14   75   17    0
## 15   15   19    0
## 16   37   19    0
## 17   48   28    0
## 18   66   28    0
## 19    1   29    0
## 20   10   29    0
## 21   44   29    0
## 22   49   29    0
## 23   67   29    0
## 24   35   31    0
## 25   36   31    0
## 26   40   31    0
## 27   77   31    0
## 28   17   34    0
## 29   75   34    0
## 30   31   35    0
## 31   36   35    0
## 32   40   35    0
## 33   77   35    0
## 34   31   36    0
## 35   35   36    0
## 36   40   36    0
## 37   77   36    0
## 38   15   37    0
## 39   19   37    0
## 40   61   39    0
## 41   31   40    0
## 42   35   40    0
## 43   36   40    0
## 44   77   40    0
## 45    1   44    0
## 46   10   44    0
## 47   29   44    0
## 48   49   44    0
## 49   67   44    0
## 50   28   48    0
## 51   66   48    0
## 52    1   49    0
## 53   10   49    0
## 54   29   49    0
## 55   44   49    0
## 56   67   49    0
## 57   39   61    0
## 58   28   66    0
## 59   48   66    0
## 60    1   67    0
## 61   10   67    0
## 62   29   67    0
## 63   44   67    0
## 64   49   67    0
## 65   17   75    0
```

```
## 66   34   75    0
## 67   31   77    0
## 68   35   77    0
## 69   36   77    0
## 70   40   77    0
## 71   82   80    0
## 72   80   82    0
```

```
### calculate indices ------------------------------------------------------

species <- data$species

raotrait <- lapply(resampled, function (x)
  lapply(x, function (y) qdecomp(y, distances, species)) |>
    bind_rows(.id = "step") |>
    mutate(step = as.numeric(step))
) |>
  bind_rows(.id = "site")

#scale steps in units of area

raotrait <- raotrait |>
  mutate(step = step / 100)

#add to list
indicesaxes[[4]] <- raotrait
names(indicesaxes)[4] <- "clonal2"

### create curve set -------------------------------------------------------

curvesets <- vector(mode = "list", length = sum(map_lgl(raotrait, is.numeric))-1)

for (i in seq_along(curvesets)){
  column <- colnames(raotrait)[i+2]
  curves <- raotrait |>
    select(site, step, {{column}}) |>
    pivot_wider(names_from = site, values_from = 3) |>
    filter(if_all(where(is.numeric), ~ !is.na(.)))
  curvesets[[i]] <- curve_set(obs = as.data.frame(curves[-1]), r = curves[[1]])
  names(curvesets)[i] <- column
}

curvesaxes[[4]] <- curvesets
names(curvesaxes)[4] <- "clonal2"


# save results ------------------------------------------------------------

saveRDS(indicesaxes, "indices_axes.rds")
saveRDS(curvesaxes, "curves_axes.rds")
```