Albacore Diet Synthesis B

Natasha Hardy

15/04/2021

Prey typologies identified in the diets of albacore tuna

About

Prey typologies text from Ms. Here, we identified key prey typologies or functional groups in albacore tuna diets using hierarchical clustering calculated with divisive algorithms (Jain et al., 1999; Legendre & Legendre, 1998) on a Gower dissimilarity matrix (Gower, 1971) to identify relational structure between mixed ecological traits types: 3 binomial variables and 2 categorical variables for prey species traits (packages included: cluster, vegan and dendextend; code found under 'Albacore_synthesis_b.Rmd'). Our objective in clustering selection (Brock et al., 2008; Charrad et al., 2014; Theodoridis & Koutroumbas, 2006) is to optimise the number of clusters based on (1) maximum differentiation or separation of species between clusters, (2) minimum differentiation of species or compactness within clusters, (3) optimal silhouette width coefficient value as well as Dunny Smith residuals, and (4) evenness or balance of cluster composition (number of species in each cluster). We assessed clusters visually for balance and consistency using cluster dendrograms and trait values that influenced a species' occupancy within a cluster are visualised using heatmaps. The relative position of species to each other in relation to their cluster occupancy, and based on shared or separation of trait values, was visualised using multivariate ordination-based, non-metric multidimensional scaling (nMDS) (Field et al., 1982).

We use code and concepts described by Anastasia Reusova here: https://towardsdatascience.com/hierarchical-clustering-on-categorical-data-in-r-a27e578f2995

And cluster validation concepts further discussed here: https://www.datanovia.com/en/lessons/cluster-validation-statistics-must-know-methods/#silhouette-coefficient

Workspace

```
# Need packages
library(plyr)
library(dplyr)
library(tidyverse)
library(reshape2)
library(factoextra)
library(here)
"%notin%" = Negate("%in%")
here::here()

# Markdown
library(formatR)
```

```
# Graphics
library(ggplot2)
library("PNWColors")
library(viridis)

# Multivariate work
library(vegan)
library(cluster)
library("dendextend")
library(NbClust)
```

PROBABLE prey traits load & cluster algorithms

Prey & traits for cluster

Here we use the cleaned file for prey species traits based on probable life stage consumed by albacore. We need to select species for which we have complete trait information for the selected traits. We obtain 156 taxa with complete trait information for the selected traits noted here.

Below we select variables for use in clustering algorithms. We selected vertical and horizontal habitat association, as well as diel and seasonal migratory traits, the probable life stage consumed and aggregation behaviour. We selected these traits because we hypothesised that these relate to the first level of filtering in the predation process – encountering prey.

```
prey_probable_load = read.csv(here("data/output_data/prey_probable_traits.csv"),
    header = TRUE) %>%
    dplyr::select(-c(X, diel_migrant, refuge, season_migrant, l_max, trophic_level:standard_total,
        energy_density:percent_lipid)) %>%
    dplyr::rename(gregarious = gregarious_primary) %>%
    mutate(diel_migrant_cat = case_when(diel_migrant_cat == "diel_no" ~ "diel (no)",
        diel_migrant_cat == "diel_UN" ~ "diel (unknown)", diel_migrant_cat == "diel_yes" ~
            "diel (yes)"), refuge_cat = case_when(refuge_cat == "refuge_no" ~ "refuge (no)",
        refuge_cat == "refuge_NA" ~ "refuge (unknown)", refuge_cat == "refuge_yes" ~
            "refuge (yes)"), gregarious = case_when(gregarious == "solitary" ~ "solitary",
        gregarious == "pairing" ~ "solitary", gregarious == "shoaling" ~ "schooling",
        gregarious == "schooling" ~ "schooling"), season_cat = case_when(season_cat ==
        "season_no" ~ "season (no)", season_cat == "season_NA" ~ "season (unknown)",
        season_cat == "season_yes" ~ "season (yes)"))
# Note that we are grouping shoaling and schooling as traits due to low
# numbers, as well as solitary and pairing species.

prey_probable = prey_probable_load %>%
    filter(diel_migrant_cat != "diel (unknown)", refuge_cat != "refuge (unknown)",
        season_cat != "season (unknown)") %>%
    drop_na()

summary(prey_probable)
```

```
##   prey_class         prey_order        prey_family         prey_sp
##  Length:156         Length:156         Length:156         Length:156
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

```
##
##
##
##   life_stage        vert_habitat       horz_habitat       diel_migrant_cat
##  Length:156         Length:156         Length:156         Length:156
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##   refuge_cat         season_cat         body_shape         phys_defense
##  Length:156         Length:156         Length:156         Min.   :0.0000
##  Class :character   Class :character   Class :character   1st Qu.:0.0000
##  Mode  :character   Mode  :character   Mode  :character   Median :0.0000
##                                                           Mean   :0.4295
##                                                           3rd Qu.:1.0000
##                                                           Max.   :1.0000
##   transparent     col_disrupt         silver        countershade
##  Min.   :0.000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.000   Median :0.0000   Median :0.0000   Median :0.0000
##  Mean   :0.109   Mean   :0.4551   Mean   :0.4231   Mean   :0.3205
##  3rd Qu.:0.000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##   gregarious            maxFO              maxN             maxM
##  Length:156         Min.   :  0.000   Min.   : 0.000   Min.   : 0.000
##  Class :character   1st Qu.:  1.500   1st Qu.: 0.000   1st Qu.: 0.000
##  Mode  :character   Median :  5.578   Median : 0.100   Median : 0.000
##                     Mean   : 17.620   Mean   : 4.505   Mean   : 5.054
##                     3rd Qu.: 28.200   3rd Qu.: 1.613   3rd Qu.: 2.250
##                     Max.   :100.000   Max.   :78.500   Max.   :95.200
```

Dataframes for analyses:

```
## Row name as column datasets For probable prey traits
prey_probable_row <- prey_probable
dendlabs <- rownames(prey_probable_row)
prey_probable <- cbind(dendlabs, prey_probable_row)

# Probable trait df subsets
probable_species = prey_probable$prey_sp
prey_probable[, 6:17] = lapply(prey_probable[, 6:17], as.factor)
probable_traits = as.data.frame(prey_probable[, c(6:10, 11)])  #12

probable_traits = prey_probable %>%
    dplyr::select(life_stage, vert_habitat, horz_habitat, diel_migrant_cat, season_cat)

str(probable_traits)
```

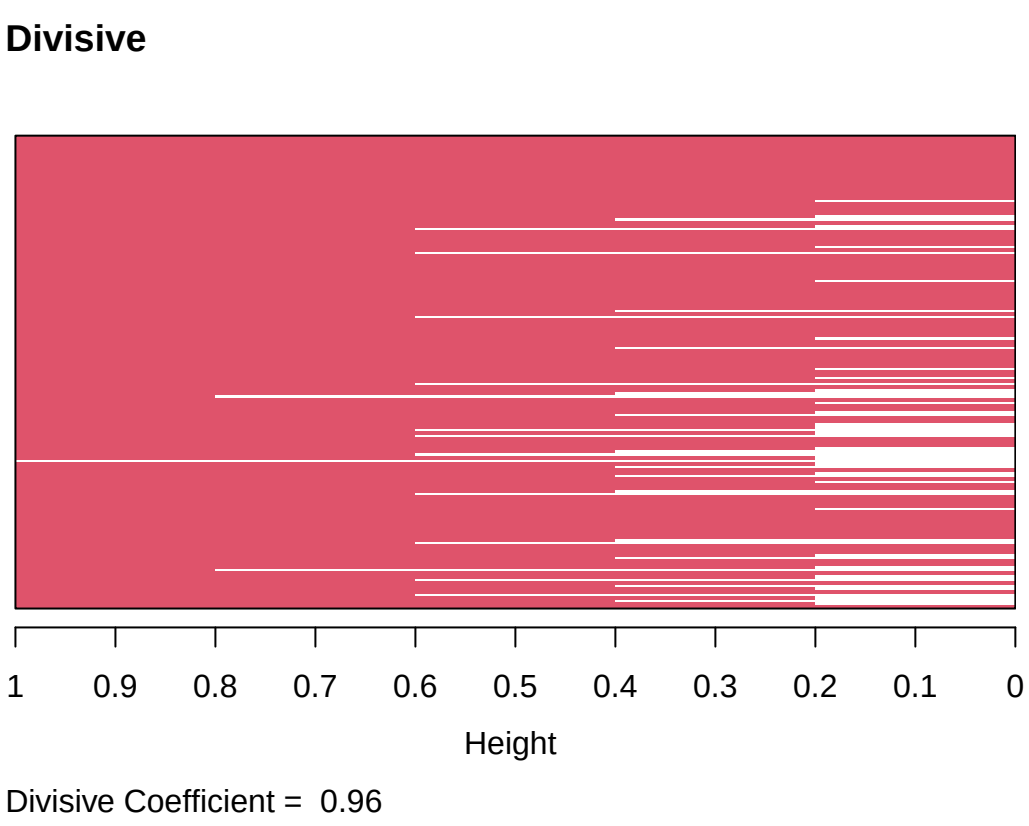
```
## 'data.frame':    156 obs. of  5 variables:
##  $ life_stage      : Factor w/ 3 levels "adult","juvenile",..: 1 1 1 2 2 1 1 1 1 1 ...
##  $ vert_habitat    : Factor w/ 5 levels "bathypelagic",..: 5 2 4 3 4 5 4 4 5 2 ...
##  $ horz_habitat    : Factor w/ 5 levels "coastal","continental shelf",..: 4 1 2 2 2 4 4 4 4 2 ...
```

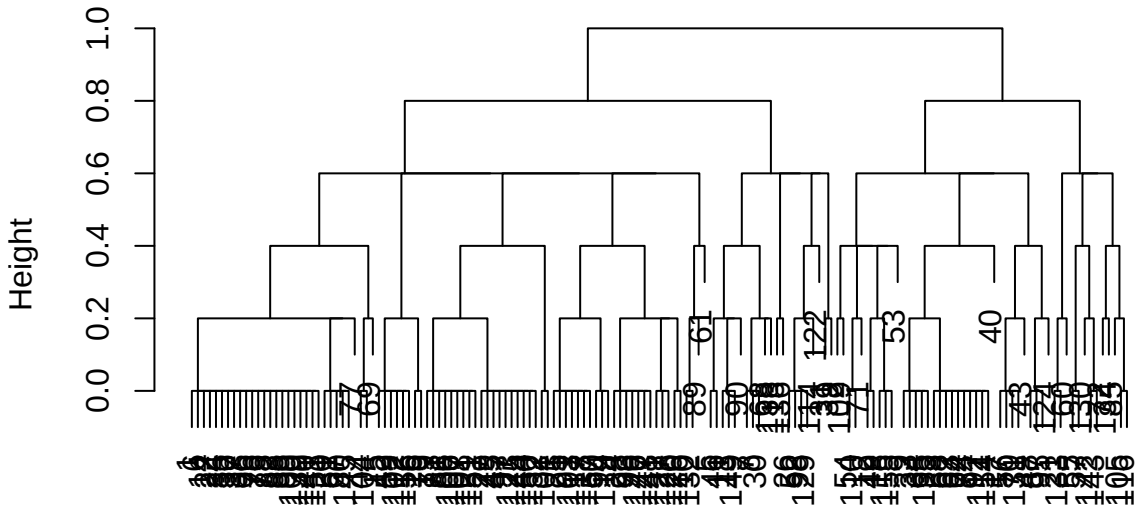
```
##  $ diel_migrant_cat: Factor w/ 2 levels "diel (no)","diel (yes)": 2 2 2 2 1 2 1 1 2 2 ...
##  $ season_cat      : Factor w/ 2 levels "season (no)",..: 2 1 2 2 2 2 2 2 1 2 ...
```

Cluster techniques

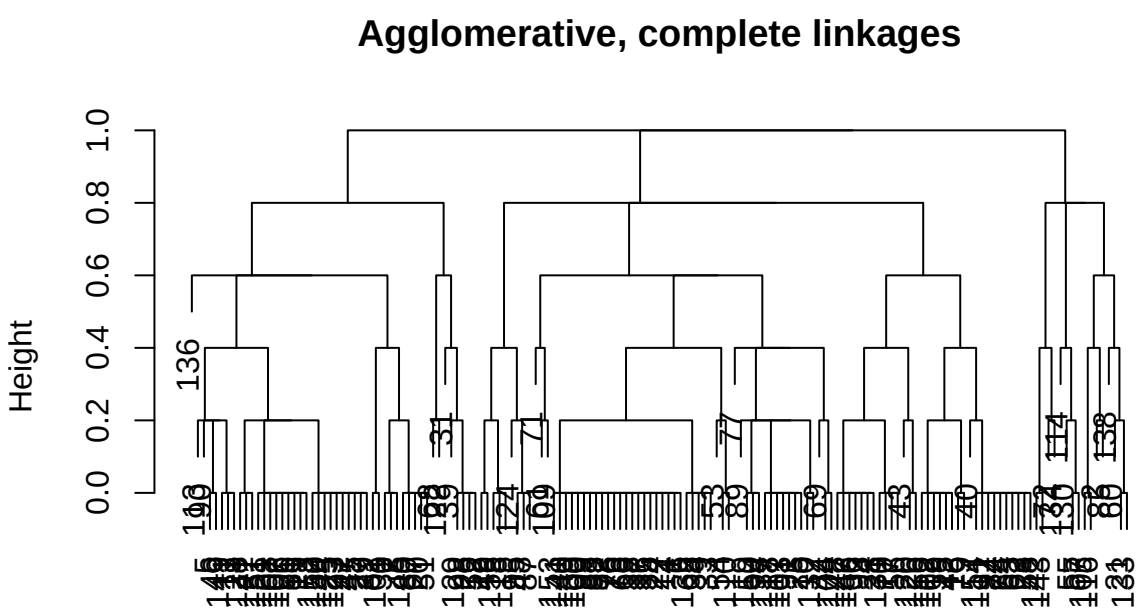
Here we generate a multivariate species-based dissimilarity matrix of the data, and both a hierarchical divisive and agglomerative clustering algorithm in order to select the most appropriate algorithm.

```
#### Distance measure ----
#select just the traits you want to contribute to ordination
prob.gower.dist <- daisy(probable_traits, metric = c("gower"))

#### Divisive cluster ----
prob.divisive.clust <- diana(as.matrix(prob.gower.dist),
                             diss = TRUE, keep.diss = TRUE)
plot(prob.divisive.clust, main = "Divisive")
```

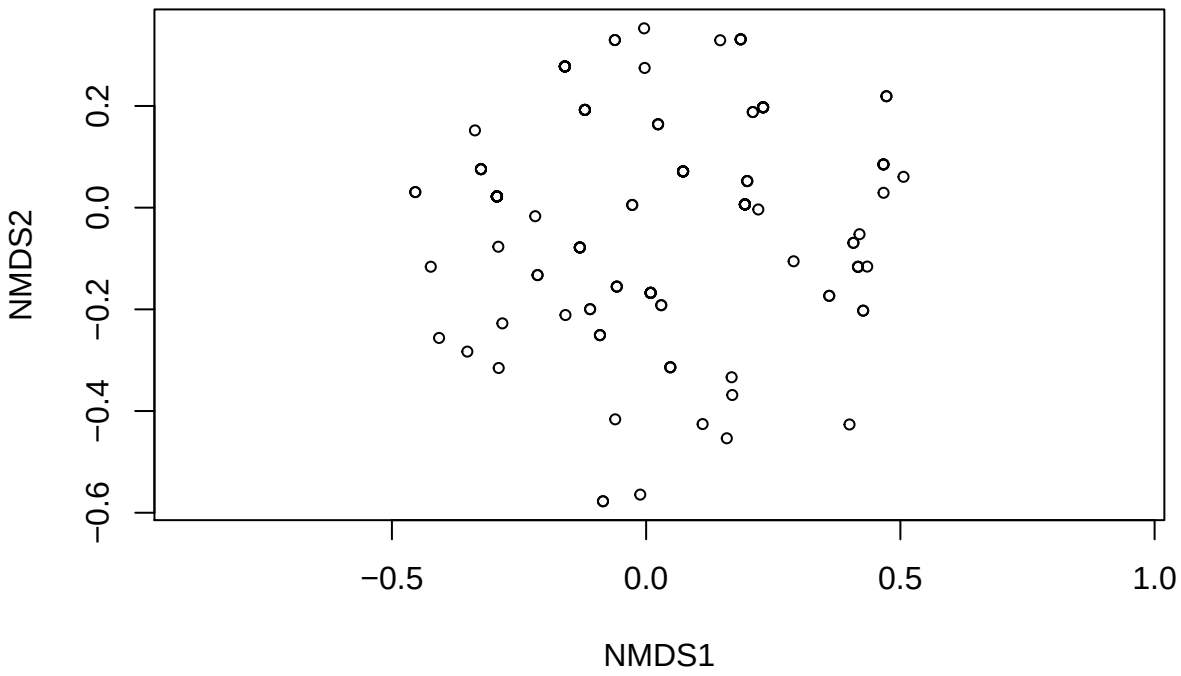





```
#### Agglomerative cluster ----
#Use "average" or "complete" linkage
#ADD NOTE ON AVE VS. COMPLETE LINKAGE
prob.aggl.clustc <- hclust(prob.gower.dist, method = "complete")
plot(prob.aggl.clustc, main = "Agglomerative, complete linkages")
```




```
#### nMDS for dissimilarity - checking the dissimilarity matrix
trait_NMDS_prob <- metaMDS(prob.gower.dist, trymax = 100)
trait_NMDS_prob[["stress"]] #stress = 0.1177472

#Plot -->
plot(trait_NMDS_prob) #plots species as black dots
```




```
#Note that the ordination looks good!
#Need to revisit this and plot in relation to clusters!
```

Cluster Assessment Output

Notes 20-24/07/2020

Ultimately we are aiming for distinct clusters of species, such that the difference within clusters is minimal and between clusters is maximised. Assessing cluster statistical tables, we are consistently observing lower average.within cluster differences using agglomerative clustering compared to divisive algorithms.

Below we are using habitat use + gregarious (as binary) traits for: (i) Divisive (ii) Agglomerative (complete)

```
# Cluster stats comes out as list while it is more convenient to look at it as a table
# This code below will produce a dataframe with observations in columns and variables in row
# Not quite tidy data, which will require a tweak for plotting, but I prefer this view as an output her
library(fpc)
cstats.table <- function(dist, tree, k) {
  clust.assess <- c("cluster.number","n","within.cluster.ss","average.within","average.between",
                    "wb.ratio","dunn2","avg.silwidth")
  clust.size <- c("cluster.size")
  stats.names <- c()
  row.clust <- c()
  output.stats <- matrix(ncol = k, nrow = length(clust.assess))
  cluster.sizes <- matrix(ncol = k, nrow = k)
  for(i in c(1:k)){
```

```
    row.clust[i] <- paste("Cluster-", i, " size")
  }
  for(i in c(2:k)){
    stats.names[i] <- paste("Test", i-1)

    for(j in seq_along(clust.assess)){
      output.stats[j, i] <- unlist(cluster.stats(d = dist, clustering = cutree(tree, k = i))[clust.asses

    }

    for(d in 1:k) {
      cluster.sizes[d, i] <- unlist(cluster.stats(d = dist, clustering = cutree(tree, k = i))[clust.siz
      dim(cluster.sizes[d, i]) <- c(length(cluster.sizes[i]), 1)
      cluster.sizes[d, i]

    }
  }
  output.stats.df <- data.frame(output.stats)
  cluster.sizes <- data.frame(cluster.sizes)
  cluster.sizes[is.na(cluster.sizes)] <- 0
  rows.all <- c(clust.assess, row.clust)
  # rownames(output.stats.df) <- clust.assess
  output <- rbind(output.stats.df, cluster.sizes)[ ,-1]
  colnames(output) <- stats.names[2:k]
  rownames(output) <- rows.all
  is.num <- sapply(output, is.numeric)
  output[is.num] <- lapply(output[is.num], round, 2)
  output
}
```

In the output below, we assess primarily:

(i) Balance between and within clusters == the number of species per cluster and between cluster. We are looking for the method which provides the greatest balance.

(ii) Balance of the lowest 'average.within' and greatest 'average.between' differences between clusters.

(iii) Lower 'dunn2' or dunny smith residual values.

(iv) Higher 'avg.sildwidth' or average silhouette width values.

NOTE: We observe the greatest balance between these cluster validation criteria for the hierarchical divisive clustering algorithm and for k = 7 clusters.

```
#Stats table for divisive method
prob.stats.df.divisive <- cstats.table(prob.gower.dist, prob.divisive.clust, 15)
prob.stats.df.divisive
```

```
##                   Test 1 Test 2 Test 3 Test 4 Test 5 Test 6 Test 7 Test 8
## cluster.number      2.00   3.00   4.00   5.00   6.00   7.00   8.00   9.00
## n                 156.00 156.00 156.00 156.00 156.00 156.00 156.00 156.00
## within.cluster.ss  18.54  14.87  12.66  10.30   9.45   7.73   7.46   6.93
## average.within      0.44   0.39   0.37   0.31   0.29   0.26   0.25   0.24
```

```
## average.between     0.65   0.63   0.63   0.59   0.59   0.58   0.58   0.58
## wb.ratio            0.68   0.62   0.58   0.53   0.50   0.44   0.43   0.42
## dunn2               1.40   1.27   1.35   1.06   0.77   0.77   0.77   0.77
## avg.silwidth        0.32   0.30   0.32   0.30   0.24   0.30   0.28   0.29
## Cluster- 1  size  107.00  86.00  86.00  31.00  31.00  31.00  31.00  31.00
## Cluster- 2  size   49.00  49.00  36.00  36.00  36.00  36.00  36.00  36.00
## Cluster- 3  size    0.00  21.00  21.00  55.00  47.00  26.00  22.00  22.00
## Cluster- 4  size    0.00   0.00  13.00  21.00  21.00  21.00  21.00  11.00
## Cluster- 5  size    0.00   0.00   0.00  13.00   8.00   8.00   4.00   4.00
## Cluster- 6  size    0.00   0.00   0.00   0.00  13.00  21.00   8.00   8.00
## Cluster- 7  size    0.00   0.00   0.00   0.00   0.00  13.00  21.00  21.00
## Cluster- 8  size    0.00   0.00   0.00   0.00   0.00   0.00  13.00  13.00
## Cluster- 9  size    0.00   0.00   0.00   0.00   0.00   0.00   0.00  10.00
## Cluster- 10  size   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
## Cluster- 11  size   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
## Cluster- 12  size   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
## Cluster- 13  size   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
## Cluster- 14  size   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
## Cluster- 15  size   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
##                   Test 9 Test 10 Test 11 Test 12 Test 13 Test 14
## cluster.number     10.00   11.00   12.00   13.00   14.00   15.00
## n                 156.00  156.00  156.00  156.00  156.00  156.00
## within.cluster.ss   6.68    6.50    5.47    4.80    4.56    3.97
## average.within      0.24    0.23    0.21    0.20    0.19    0.18
## average.between     0.58    0.58    0.57    0.57    0.57    0.57
## wb.ratio            0.41    0.40    0.37    0.35    0.34    0.32
## dunn2               0.77    0.77    0.77    0.77    0.76    1.06
## avg.silwidth        0.28    0.28    0.33    0.37    0.37    0.39
## Cluster- 1  size   31.00   31.00   31.00   31.00   31.00   31.00
## Cluster- 2  size   36.00   36.00   11.00   11.00   11.00   11.00
## Cluster- 3  size   22.00   22.00   22.00   22.00   22.00   22.00
## Cluster- 4  size   11.00   11.00   11.00   11.00   11.00   11.00
## Cluster- 5  size    4.00    4.00   25.00   16.00   16.00   16.00
## Cluster- 6  size    8.00    8.00    4.00    4.00    4.00    4.00
## Cluster- 7  size   21.00   21.00    8.00    8.00    8.00    8.00
## Cluster- 8  size   13.00   13.00   21.00    9.00    9.00    9.00
## Cluster- 9  size    8.00    6.00   13.00   21.00   21.00   21.00
## Cluster- 10  size   2.00    2.00    6.00   13.00    3.00    3.00
## Cluster- 11  size   0.00    2.00    2.00    6.00    6.00    6.00
## Cluster- 12  size   0.00    0.00    2.00    2.00    2.00    2.00
## Cluster- 13  size   0.00    0.00    0.00    2.00   10.00    5.00
## Cluster- 14  size   0.00    0.00    0.00    0.00    2.00    5.00
## Cluster- 15  size   0.00    0.00    0.00    0.00    0.00    2.00
```

```
View(prob.stats.df.divisive)
write.csv(prob.stats.df.divisive, here("outputs_figures/clusters/prob.stats.df.divisive.csv"))

#Stats table for agglomerative method
prob.stats.df.aggl <- cstats.table(prob.gower.dist, prob.aggl.clustc, 15)
#complete linkages looks like the most balanced approach
prob.stats.df.aggl
```

```
##                   Test 1 Test 2 Test 3 Test 4 Test 5 Test 6 Test 7 Test 8
## cluster.number      2.00   3.00   4.00   5.00   6.00   7.00   8.00   9.00
```

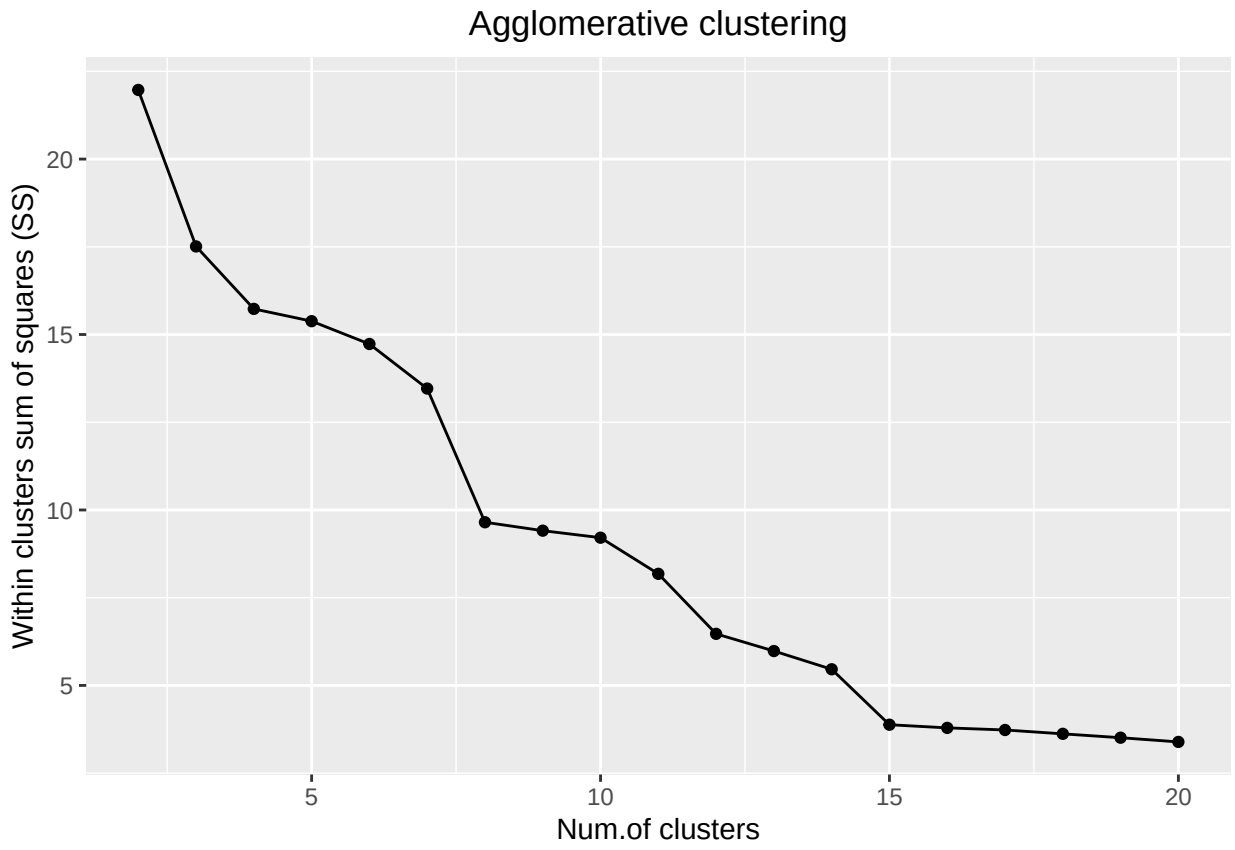
```
## n                 156.00 156.00 156.00 156.00 156.00 156.00 156.00 156.00
## within.cluster.ss  21.97  17.51  15.73  15.38  14.73  13.46   9.65   9.41
## average.within      0.47   0.42   0.40   0.39   0.38   0.36   0.31   0.30
## average.between     0.58   0.61   0.61   0.61   0.61   0.60   0.59   0.59
## wb.ratio            0.82   0.69   0.65   0.64   0.63   0.60   0.52   0.51
## dunn2               1.12   1.18   1.12   1.02   0.88   0.92   0.92   0.83
## avg.silwidth        0.17   0.21   0.19   0.15   0.14   0.15   0.26   0.23
## Cluster- 1  size  108.00  92.00  92.00  92.00  92.00  83.00  50.00  50.00
## Cluster- 2  size   48.00  16.00  16.00  12.00   8.00   8.00   8.00   8.00
## Cluster- 3  size    0.00  48.00  40.00  40.00  40.00  40.00  33.00  33.00
## Cluster- 4  size    0.00   0.00   8.00   8.00   8.00   9.00  40.00  40.00
## Cluster- 5  size    0.00   0.00   0.00   4.00   4.00   8.00   9.00   9.00
## Cluster- 6  size    0.00   0.00   0.00   0.00   4.00   4.00   8.00   6.00
## Cluster- 7  size    0.00   0.00   0.00   0.00   0.00   4.00   4.00   4.00
## Cluster- 8  size    0.00   0.00   0.00   0.00   0.00   0.00   4.00   2.00
## Cluster- 9  size    0.00   0.00   0.00   0.00   0.00   0.00   0.00   4.00
## Cluster- 10  size   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
## Cluster- 11  size   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
## Cluster- 12  size   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
## Cluster- 13  size   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
## Cluster- 14  size   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
## Cluster- 15  size   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
##                   Test 9 Test 10 Test 11 Test 12 Test 13 Test 14
## cluster.number     10.00   11.00   12.00   13.00   14.00   15.00
## n                 156.00  156.00  156.00  156.00  156.00  156.00
## within.cluster.ss   9.21    8.18    6.47    5.98    5.46    3.88
## average.within      0.30    0.28    0.24    0.23    0.22    0.19
## average.between     0.59    0.59    0.58    0.58    0.58    0.57
## wb.ratio            0.50    0.47    0.41    0.40    0.38    0.32
## dunn2               0.83    0.83    0.82    1.15    1.26    1.30
## avg.silwidth        0.19    0.20    0.28    0.29    0.30    0.41
## Cluster- 1  size   50.00   50.00   50.00   50.00   47.00   30.00
## Cluster- 2  size    8.00    8.00    8.00    4.00    4.00    4.00
## Cluster- 3  size   33.00   33.00   14.00   14.00   14.00   17.00
## Cluster- 4  size   39.00   29.00   29.00   29.00   29.00   14.00
## Cluster- 5  size    9.00   10.00   10.00   10.00   10.00   29.00
## Cluster- 6  size    6.00    9.00    9.00    9.00    9.00   10.00
## Cluster- 7  size    4.00    6.00   19.00   19.00   19.00    9.00
## Cluster- 8  size    2.00    4.00    6.00    4.00    4.00   19.00
## Cluster- 9  size    4.00    2.00    4.00    6.00    6.00    4.00
## Cluster- 10  size   1.00    4.00    2.00    4.00    4.00    6.00
## Cluster- 11  size   0.00    1.00    4.00    2.00    3.00    4.00
## Cluster- 12  size   0.00    0.00    1.00    4.00    2.00    3.00
## Cluster- 13  size   0.00    0.00    0.00    1.00    4.00    2.00
## Cluster- 14  size   0.00    0.00    0.00    0.00    1.00    4.00
## Cluster- 15  size   0.00    0.00    0.00    0.00    0.00    1.00
```

```
View(prob.stats.df.aggl)
write.csv(prob.stats.df.aggl, here("outputs_figures/clusters/prob.stats.df.aggl.csv"))
#As per text for adult, average within cluster metric is minimised for ~7-10 clusters, and the average
```

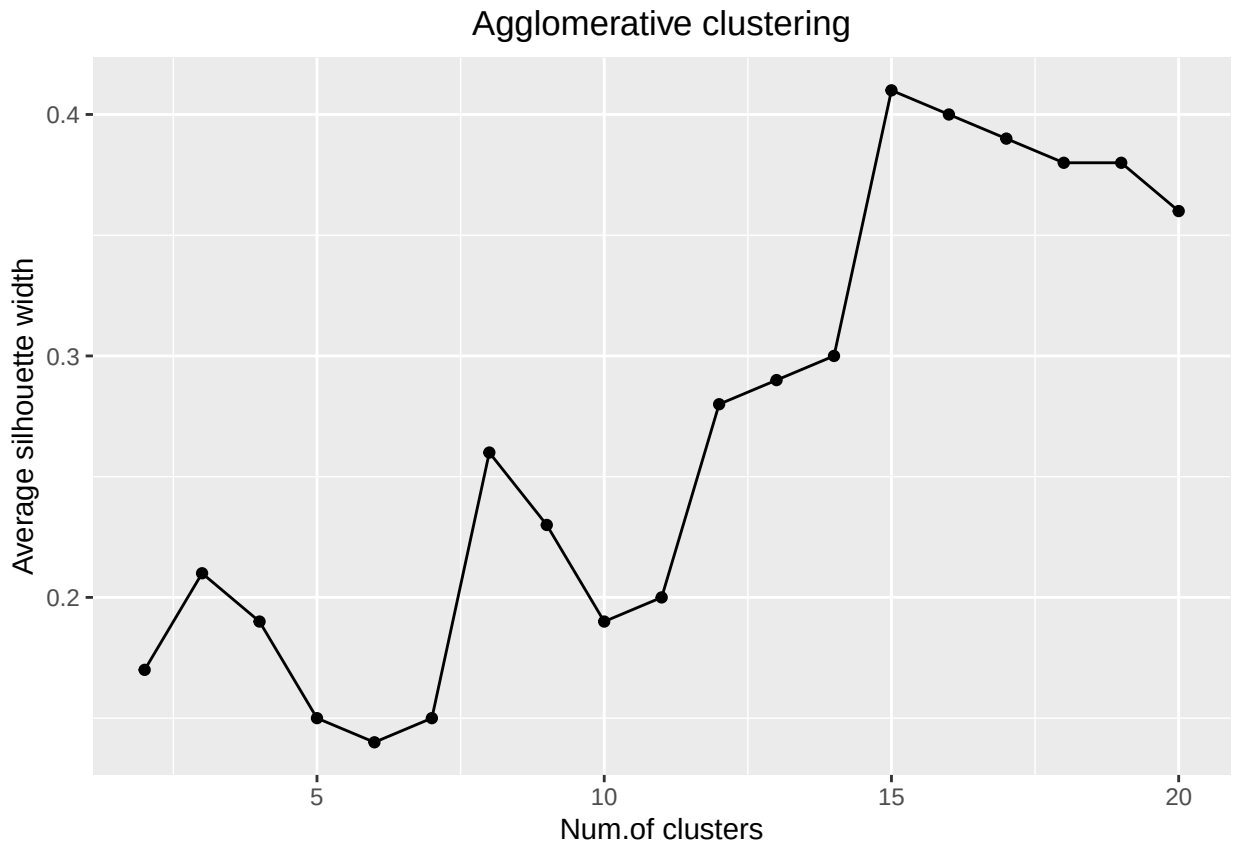
Note Agglomerative – for visualising cluster number selection.

Currently not using the agglomerative output based on stats table output.

```
### Using "Elbow" and "Silhouette" methods to identify the best number of clusters

# Elbow method

# Agglomerative clustering,provides a more ambiguous picture
ggplot(data = data.frame(t(cstats.table(prob.gower.dist, prob.aggl.clustc, 20))),
       aes(x=cluster.number, y=within.cluster.ss)) +
  geom_point()+
  geom_line()+
  ggtitle("Agglomerative clustering") +
  labs(x = "Num.of clusters", y = "Within clusters sum of squares (SS)") +
  theme(plot.title = element_text(hjust = 0.5))
```



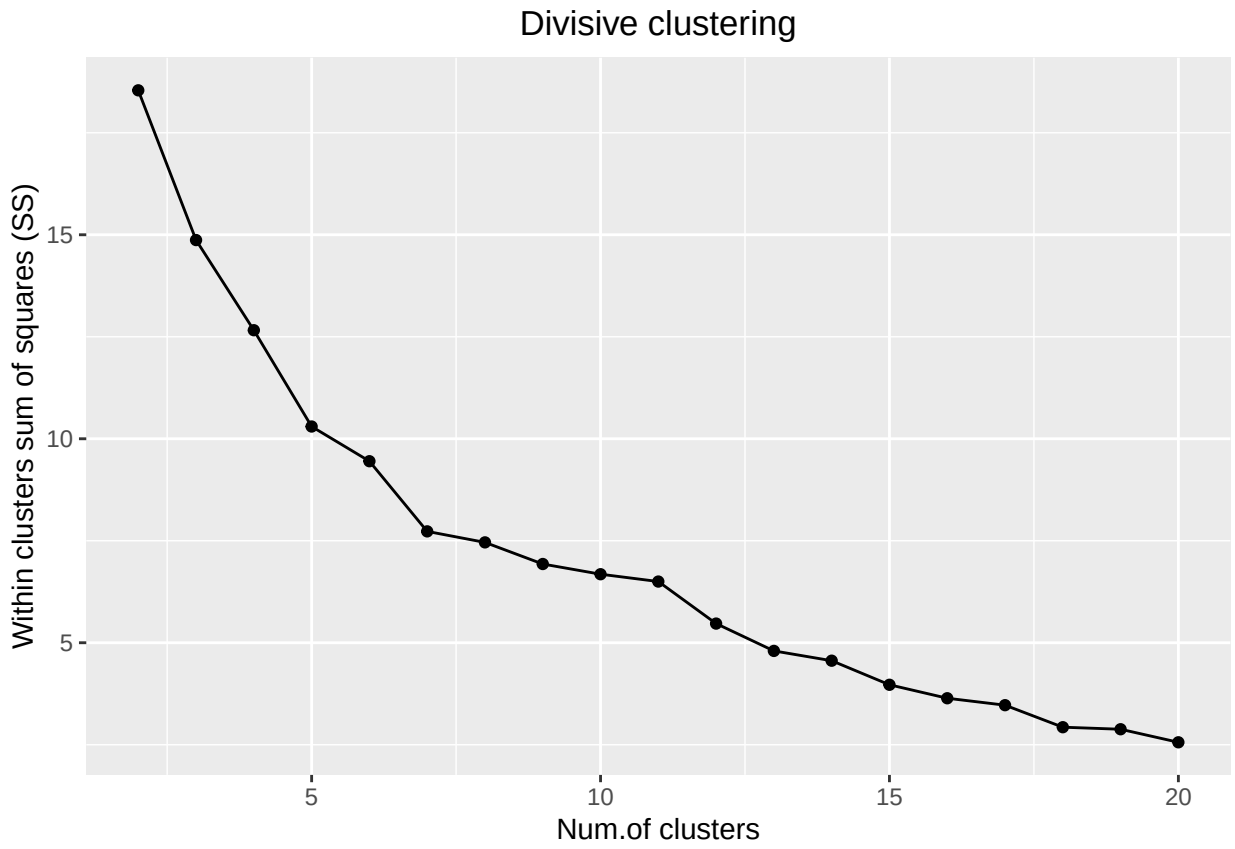

```
## Silhouette
#When it comes to silhouette assessment, the rule is you should choose the number that maximizes the
#silhouette coefficient because you want clusters that are distinctive (far) enough to be considered se

# Agglomerative
ggplot(data = data.frame(t(cstats.table(prob.gower.dist, prob.aggl.clustc, 20))),
       aes(x=cluster.number, y=avg.silwidth)) +
  geom_point()+
  geom_line()+
  ggtitle("Agglomerative clustering") +
  labs(x = "Num.of clusters", y = "Average silhouette width") +
  theme(plot.title = element_text(hjust = 0.5))
```



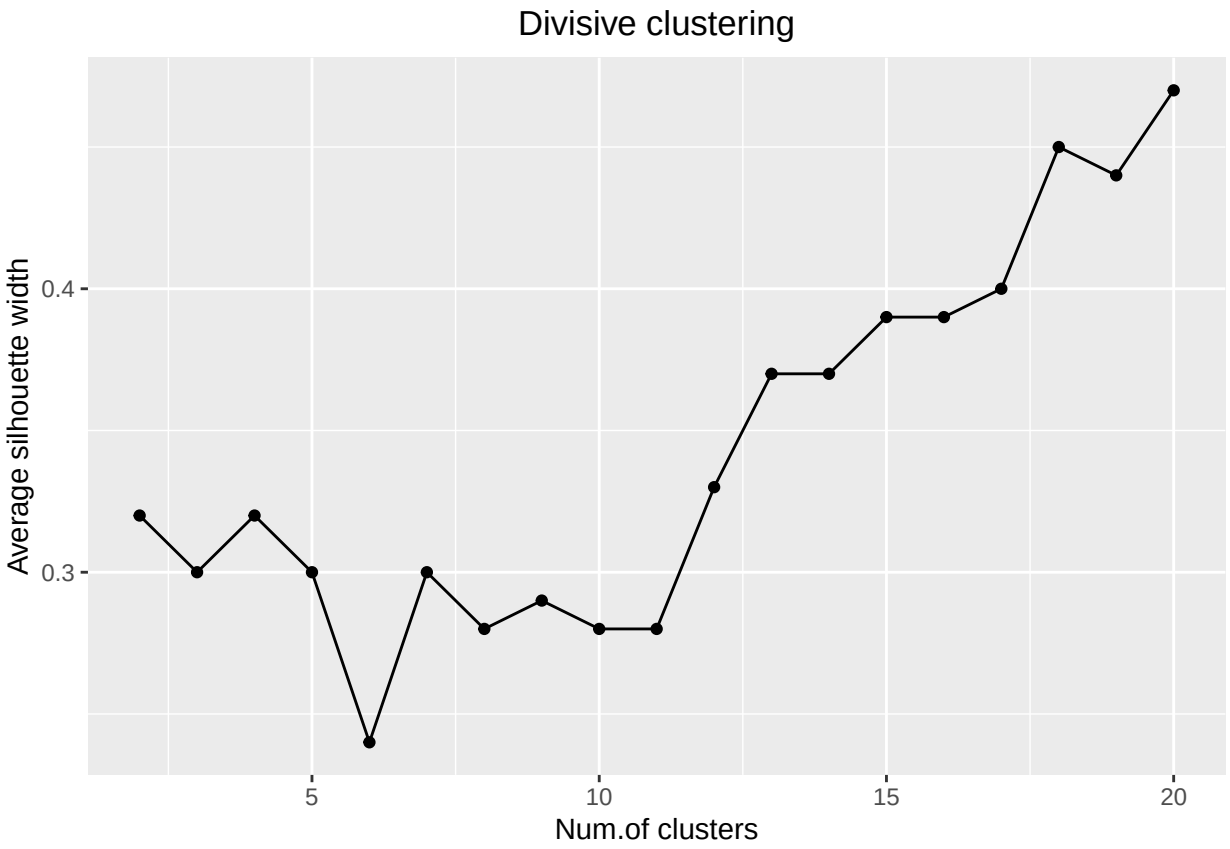
Note Divisive – for visualising cluster number selection.

- Elbow method (20/03/2021) using habitat use + gregarious (binary) + life stage without refuge use inflection at 7
- Silhouette (20/03/2021) using habitat use + gregarious (binary) + life stage without refuge use inflection at 7

```
### Using "Elbow" and "Silhouette" methods to identify the best number of clusters

# Elbow method

# Agglomerative clustering,provides a more ambiguous picture
ggplot(data = data.frame(t(cstats.table(prob.gower.dist, prob.divisive.clust, 20))),
       aes(x=cluster.number, y=within.cluster.ss)) +
  geom_point()+
  geom_line()+
  ggtitle("Divisive clustering") +
  labs(x = "Num.of clusters", y = "Within clusters sum of squares (SS)") +
  theme(plot.title = element_text(hjust = 0.5))
```




```
## Silhouette
#When it comes to silhouette assessment, the rule is you should choose the number that maximizes the
#silhouette coefficient because you want clusters that are distinctive (far) enough to be considered se

# Agglomerative
ggplot(data = data.frame(t(cstats.table(prob.gower.dist, prob.divisive.clust, 20))),
       aes(x=cluster.number, y=avg.silwidth)) +
  geom_point()+
  geom_line()+
  ggtitle("Divisive clustering") +
  labs(x = "Num.of clusters", y = "Average silhouette width") +
  theme(plot.title = element_text(hjust = 0.5))
```



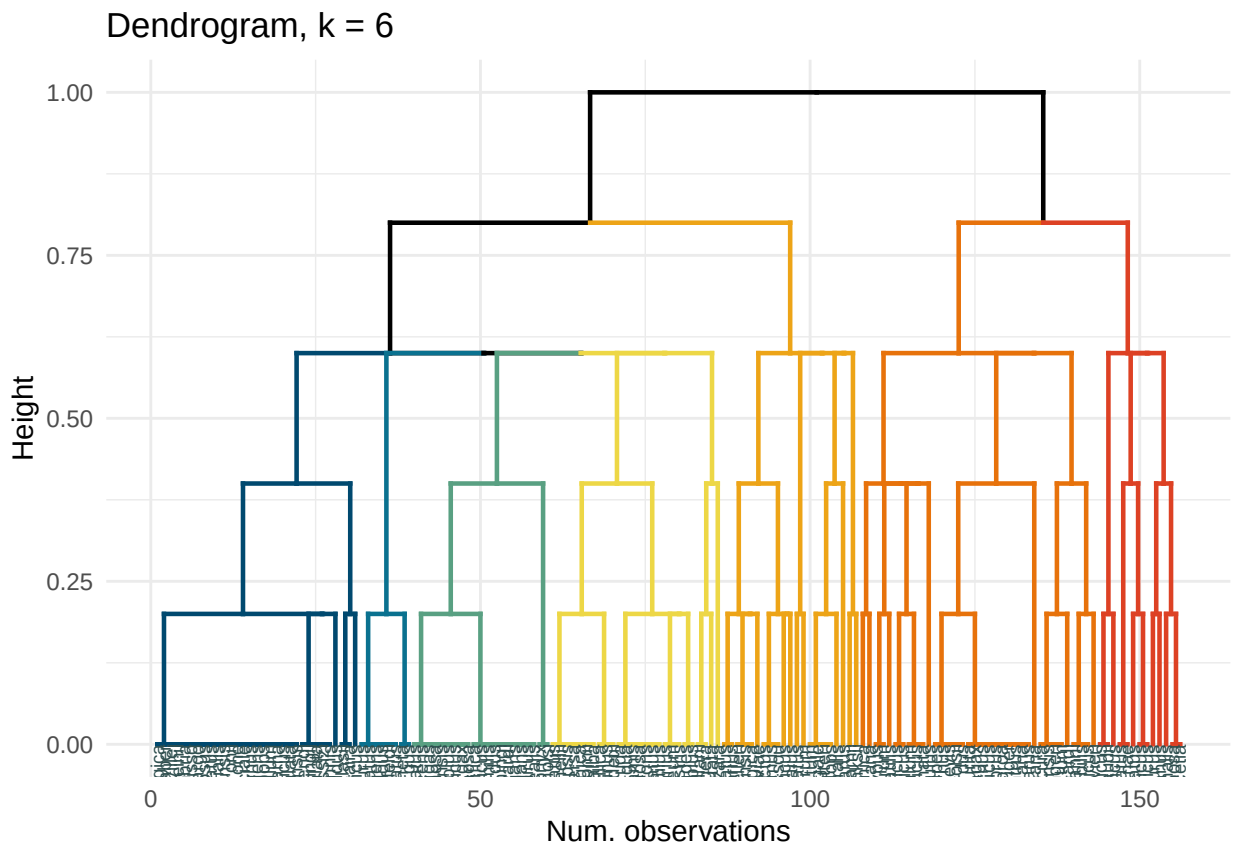
PROBABLE Divisive k = 7

Cluster Dendrograms

Using k = 7 for the habitat association, aggregation behaviour & probable life stage traits.

Horizontal dendrogram

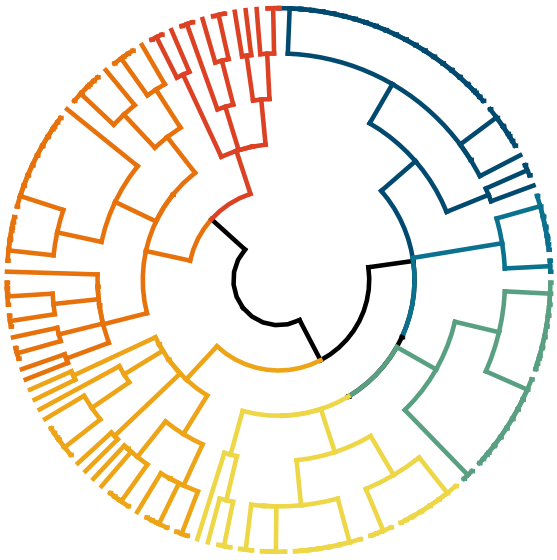
```
#Using agglomerative hierarchical clustering, k = 8
prob.dendro <- as.dendrogram(prob.divisive.clust) #156 species
PNW.pal7 <- pnw_palette(7, name = "Bay", type = "continuous")
###Horizontal dendrogram - Probable traits

#Horizontal cluster illustration version
prob.dendro.col <- prob.dendro %>%
  set("branches_k_color", k = 7, value = PNW.pal7) %>%
  set("branches_lwd", 0.8) %>%
  set("labels", probable_species) %>% #NOT VERY LEGIBLE...
  set("labels_colors",
      value = c("darkslategray")) %>%
  set("labels_cex", 0.5)
prob.ggd1 <- as.ggdend(prob.dendro.col)
prob.dendro.graph <- ggplot(prob.ggd1, theme = theme_minimal()) +
  labs(x = "Num. observations", y = "Height", title = "Dendrogram, k = 6")
prob.dendro.graph
```




```
ggsave(here('outputs_figures/clusters/probable_divis_simple/prob.dendro.horz.k7.png'), plot=prob.dendro
```

Radial dendrogram

```
# Radial plot looks less cluttered (and cooler)
prob.dendro.rad <- ggplot(prob.ggd1, labels = FALSE) +
  scale_y_reverse(expand = c(0.2, 0)) +
  coord_polar(theta="x")
prob.dendro.rad
```



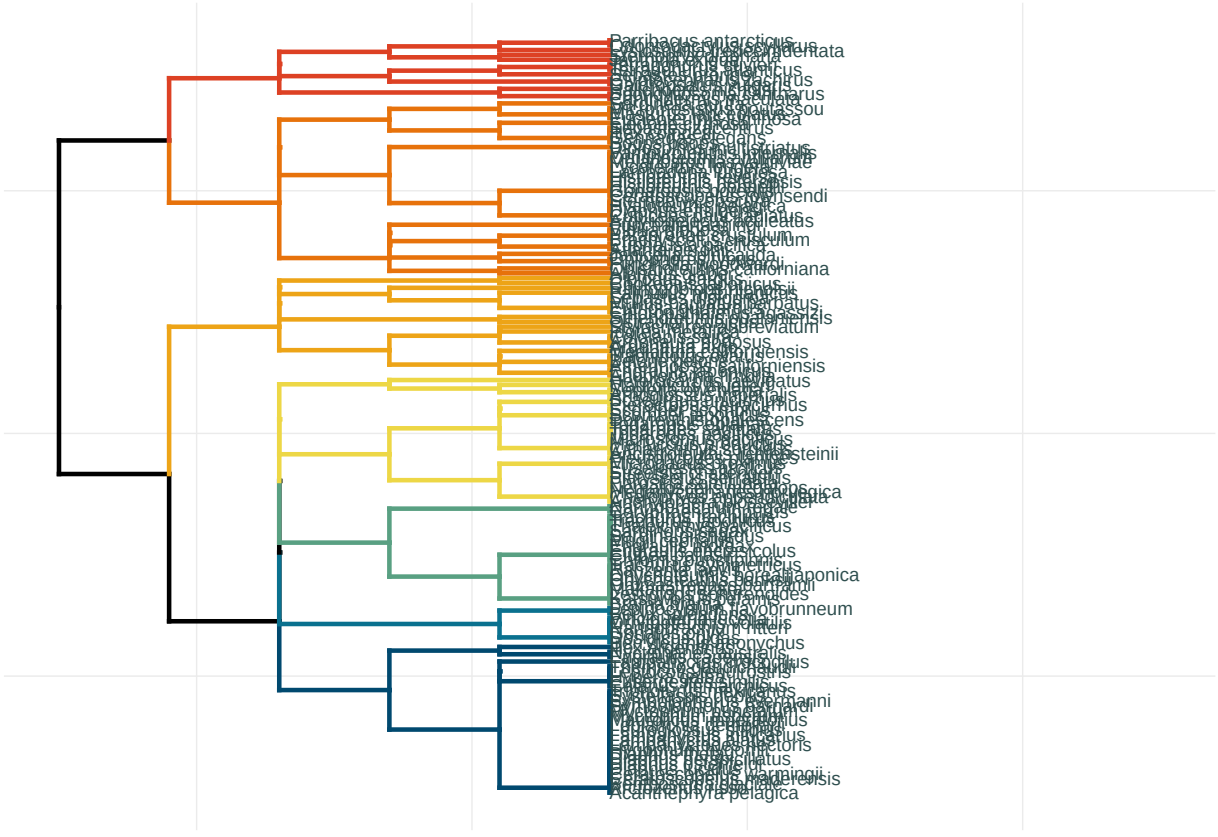
```
#No labels on this one, labels were too cluttered/problems
#Save radial dendrogram for chat

ggsave(here('outputs_figures/clusters/probable_divis_simple/prob.dendro.rad.k7.png'), plot=prob.dendro.r
```

Vertical dendrogram

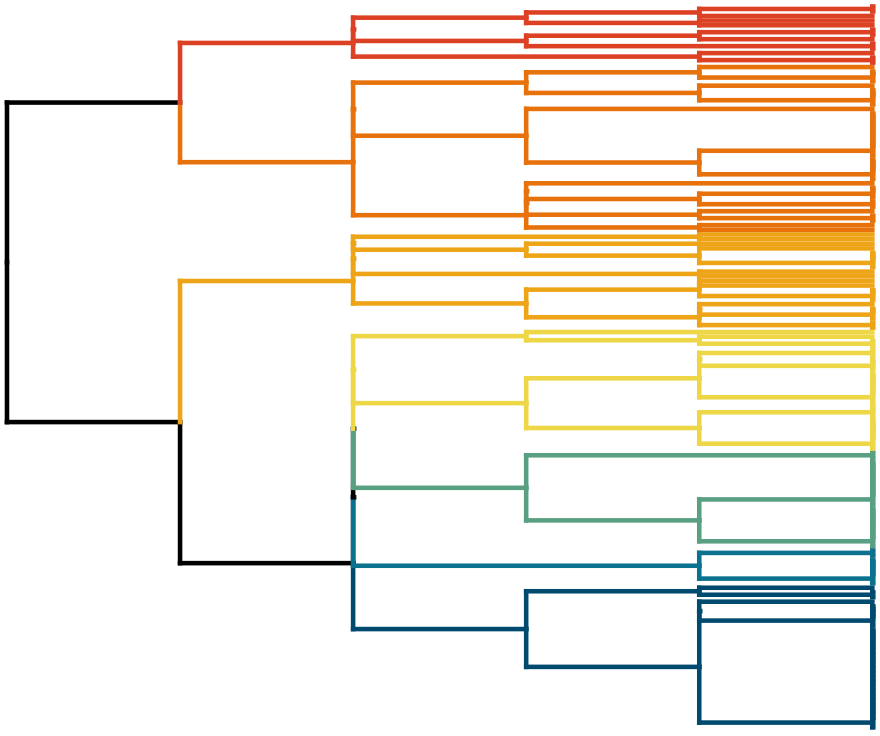
Similar to https://stackoverflow.com/questions/38034663/rotate-labels-for-ggplot-dendrogram

```
# This is a different way to compute hierarchical clustering and cut the tree
#clus <- hcut(mydist, k = 6, hc_func = 'hclust', hc_method = 'ward.D2', graph = FALSE, isdiss = TRUE)

#Below is problematic0
#labels(dend) <- paste0(paste0(rep('', 3), collapse = ''), species0)
#dend <- sort(dend, decreasing = FALSE)
#View(labels(dend))

#Creating df for the dend labels so that we can accurately line them up with the species
dendlabs <- labels(prob.dendro) #Need to create strings of labels to manipulate
dendlabs2 <- as.data.frame(dendlabs) #turn in df

#Join these data so we can relabel the dendrogram
#Use plyr function because it conserves the row order of the left df, which matters for assigning label

dfdend <- join(dendlabs2, prey_probable)
```

```
ggd1 <- ggplot(prob.dendro %>%
                 set("branches_k_color", k = 7, value = PNW.pal7) %>%
                 set("branches_lwd", 0.8) %>%
                 set("labels", dfdend$prey_sp) %>% #NOT VERY LEGIBLE...
                 set("labels_colors",
                     value = c("darkslategray")) %>%
                 set("labels_cex", 0.5),
               theme = theme_minimal(),
               horiz = TRUE)

ggd1 <- ggd1 + theme(panel.grid.major = element_blank(),
                     axis.text = element_blank(),
                     axis.title = element_blank())
ggd1 <- ggd1 + ylim(max(get_branches_heights(prob.dendro)), -1)
ggd1
```




```
ggsave(here('outputs_figures/clusters/probable_divis_simple/prob.dendro.vertlabs.k7.pdf'), plot=ggd1, wi

#With labels removed!!!

prob.dendro.vert <- ggplot(prob.ggd1, horiz = TRUE, labels = FALSE) +
  scale_y_reverse(expand = c(0.2, 0)) #+
#coord_polar(theta="x")
prob.dendro.vert
```



```
#Export as .png
ggsave(here('outputs_figures/clusters/probable_divis_simple/prob.dendro.vert2.k7.png'), plot=prob.dendr
```

Cluster Heatmaps

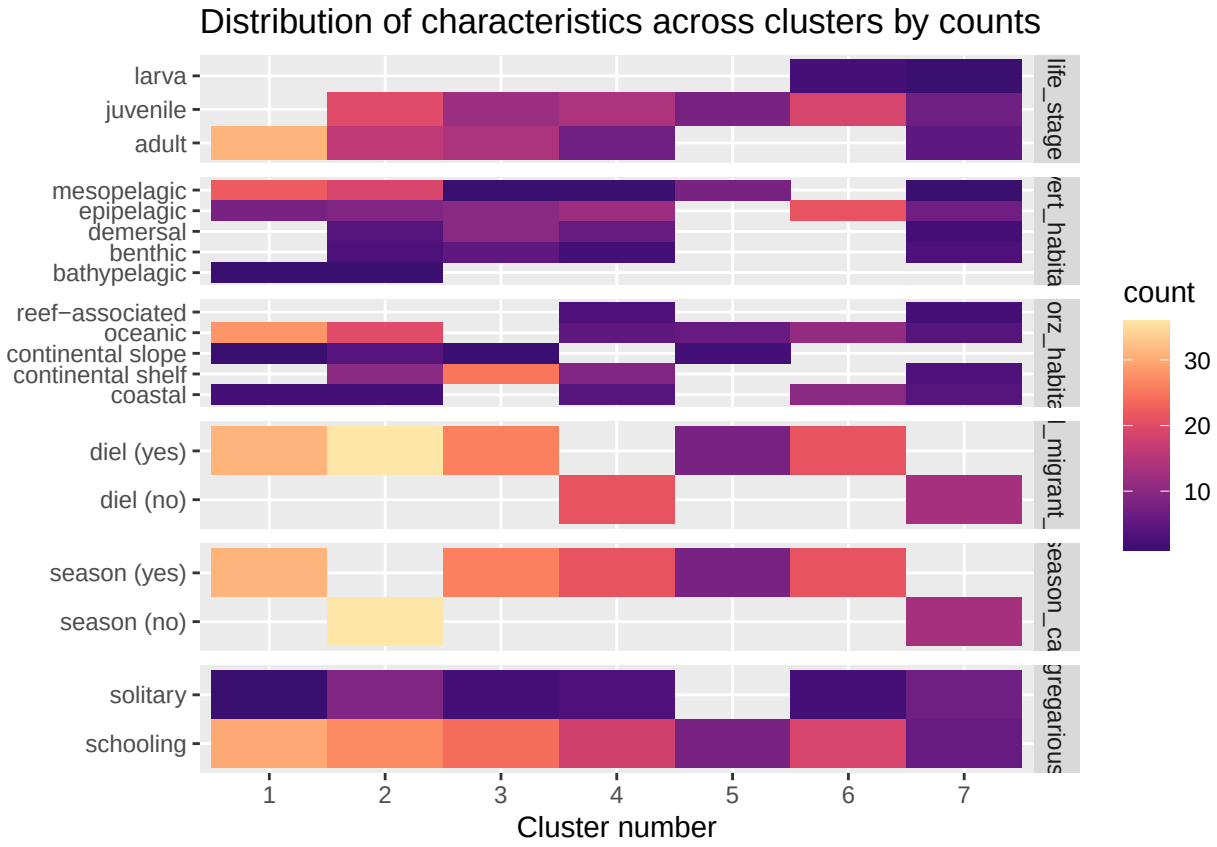
```
#Extract cluster number to trait matrix
prob.clust.num <- cutree(prob.divisive.clust, k = 7)

#we want to bind the original dataset with the cluster numbers such that each species is assigned a clu
#can use whole data or just traits use to just look at unique species clusters in relation to traits
#alb.cl <- cbind(ctraitsO, alb.clust.num)
#OR
prob.prey.cl <- cbind(prey_probable, prob.clust.num)

#View(prob.prey.cl)

#write.csv(prob.prey.cl, here("data/output_data/prob.prey.clusternum_habgreg.divis.k8.csv"))
write.csv(prob.prey.cl, here("outputs_figures/clusters/probable_divis_simple/prob.prey.clusternum_habgr
```

```
# Time for the heatmap
# the 1st step here is to have 1 variable per row
# factors have to be converted to characters in order not to be dropped
```

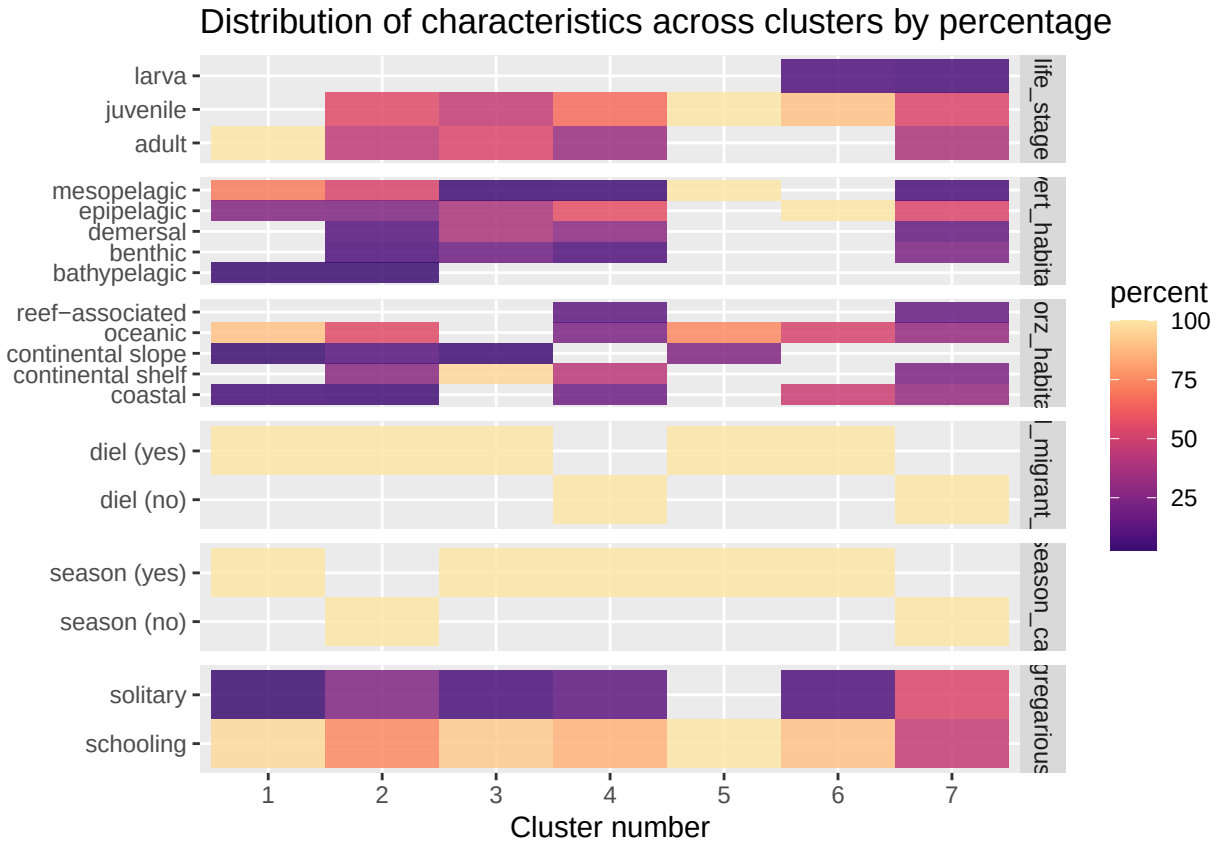
```r
#Note plyr can mess with this!!
#detach("package:plyr", unload=TRUE)

#Create dfs for graphs
prob.clust.long = prob.prey.cl %>%
  dplyr::select(prey_sp, life_stage:season_cat, `gregarious`, `prob.clust.num`, -refuge_cat) %>% #maxFO
  reshape2::melt(id.vars = c("prey_sp", "prob.clust.num"), variable.name = "trait", value.name = "level
  group_by(prob.clust.num, trait, level) %>%
  mutate(count = n_distinct(prey_sp)) %>%
  distinct(prob.clust.num, trait, level, count) %>% #, percent
  group_by(prob.clust.num, trait) %>%
  mutate(percent = count / sum(count)*100) %>%
  arrange(prob.clust.num)

#str(prob.clust.long)

#heatmap.c will be suitable in case you want to go for absolute counts - but it doesn't tell much to my
#problem below involves the values of our data being ordinal, therefore they are not unique
levels(prob.clust.long$trait)
```

```
## [1] "life_stage"       "vert_habitat"     "horz_habitat"     "diel_migrant_cat"
## [5] "season_cat"       "gregarious"
```

```r
#Our data above comes truncated, you would need to truncate the data and re-label clusters depending on
#Example: View(alb.cust.long.q[96:nrow(alb.cust.long.q),])
heatmap.c <- ggplot(prob.clust.long, aes(x = factor(prob.clust.num), y = level)) +
  geom_tile(aes(fill = count))+
  labs(title = "Distribution of characteristics across clusters by counts", x = "Cluster number", y = NU
  scale_fill_viridis(option="magma", begin = 0.2, end = 0.95)+
  facet_grid(trait~. , scales="free_y")
heatmap.c
```




```
ggsave(here('outputs_figures/clusters/probable_divis_simple/prob.dendro.heatcounts.divis.k7.life.pdf'),


heatmap.p <- ggplot(prob.clust.long, aes(x = factor(prob.clust.num), y = factor(level, ordered = T))) +
  geom_tile(aes(fill = percent), alpha = 0.85)+
  labs(title = "Distribution of characteristics across clusters by percentage", x = "Cluster number", y
  #scale_fill_gradient2(low = "darkslategray1", mid = "yellow", high = "turquoise4") +
  scale_fill_viridis(option="magma", begin = 0.2, end = 0.95)+
  facet_grid(trait~., scales="free_y")
heatmap.p
```




```
ggsave(here('outputs_figures/clusters/probable_divis_simple/prob.dendro.heatpercent.divis.k7.life.pdf')

#library(plyr)
```

```
summary(as.factor(prob.prey.cl$prob.clust.num))
```

```
##  1  2  3  4  5  6  7
## 31 36 26 21  8 21 13
```

```
#clusters  1  2  3  4  5  6  7
#         31 36 26 21  8 21 13
```

NMDS for checking on our ordination - divisive

Here we want to visualise species' occupancy of trait-based cluster in multivariate space. Species have been treated as sites and their trait occupancy as 'species' in the ordination and nMDS routines. Thus their position in multivariate space is based off similar and dissimilar trait values. We then overlay their cluster number graphically.

Ordination

```
#### nMDS for dissimilarity
trait_NMDS_prob <- metaMDS(prob.gower.dist, trymax = 1000)  #solution after 541 iterations
trait_NMDS_prob[["stress"]]  #stress = 0.1477077 #reasonable
```

Extract NMDS coordinates and associate with co-variates/grouping factors

```
#Extract NMDS coordinates and associate with co-variates/grouping factors

#Using the scores function from vegan to extract the site scores and convert to a data.frame
data.scores <- as.data.frame(scores(trait_NMDS_prob))  #, "species"

#create a column of site names, from the rownames of data.scores
data.scores$points <- rownames(data.scores)

#bind treatment labels and score values
treatment.scores <- cbind(prob.prey.cl, data.scores)

#Check
#str(treatment.scores)
#Awesome
```

Convex hull calculations

For each ordination and set of grouping variables input to data scores chunk.

```
#Create convex hulls for the space occupied by each cluster value
unique(treatment.scores$prob.clust.num)
```

```
## [1] 1 2 3 4 5 6 7
```

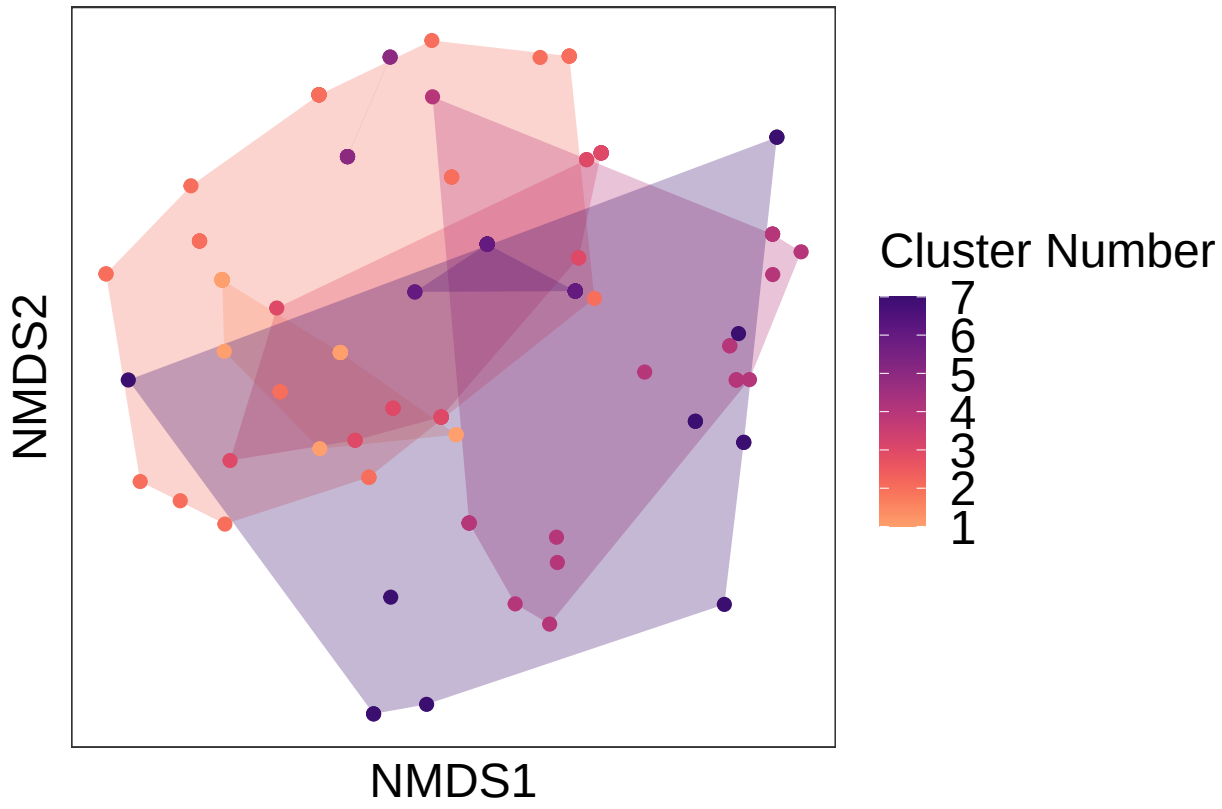
```
length(treatment.scores$prob.clust.num)
```

```
## [1] 156
```

```
#Cluster number - hull loop
clust = as.character(unique(treatment.scores$prob.clust.num))
for(i in 1:length(clust)) {
  temp = clust[i]
  df = treatment.scores[treatment.scores$prob.clust.num == temp, ][chull(treatment.scores[treatment.sco
  assign(paste0('grp.',temp), df)
}

#combine the hull data
hull.data <- rbind(grp.1, grp.2, grp.3, grp.4, grp.5, grp.6, grp.7)

#str(hull.data)
```

NMDS plot

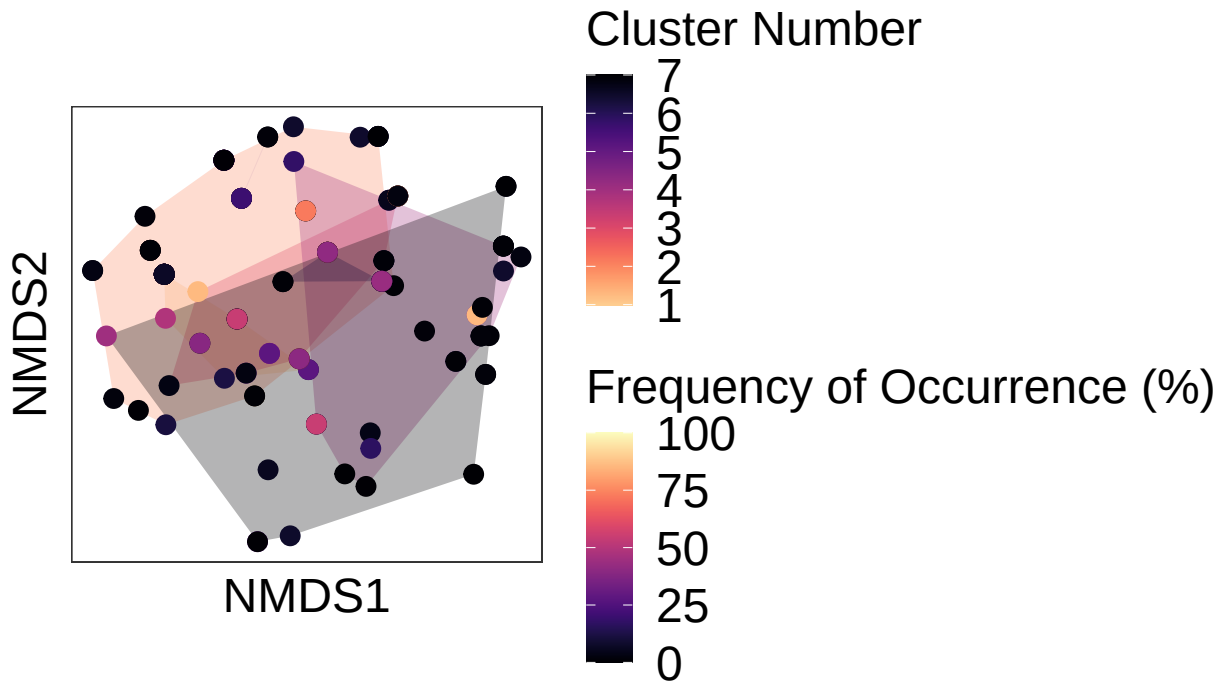
```
#nMDS plot for the assessmenet of global change (yes/no) and the drivers.
#as.integer(unique(treatment.scores$adult.clust.num))

trait_nMDS_prob_fig <- ggplot() +
  geom_polygon(data=hull.data,
               aes(x=NMDS1, y=NMDS2,
                   fill= prob.clust.num,
                   group= prob.clust.num),
               alpha=0.30) + # add the convex hulls
  geom_point(data=treatment.scores,
             aes(x=NMDS1, y=NMDS2, colour= prob.clust.num),
             size=2) + # add the point markers
  coord_equal() +
  theme_bw()  +
  theme(axis.text.x = element_blank(),  # remove x-axis text
        axis.text.y = element_blank(), # remove y-axis text
        axis.ticks = element_blank(),  # remove axis ticks
        axis.title.x = element_text(size=18), # remove x-axis labels
        axis.title.y = element_text(size=18), # remove y-axis labels
        legend.title = element_text(size = 18),
        legend.text = element_text(size = 18),
        legend.justification = c(0,0.5),
        #panel.background = element_rect(fill = "lightgrey"),
        panel.grid.major = element_blank(),  #remove major-grid labels
        panel.grid.minor = element_blank(),  #remove minor-grid labels
        plot.background = element_blank()) +
  scale_colour_viridis(option="magma", begin = 0.8, end = 0.2, name = "Cluster Number") +
  scale_fill_viridis(option="magma", begin = 0.8, end = 0.2, name = "Cluster Number") #+

trait_nMDS_prob_fig
```




```
#ggsave(here("outputs_figures/clusters/probable_divis_simple/trait_nMDS_prob_clusters.png"),
#       plot = trait_nMDS_prob_fig, width = 8, height = 8, dpi = 300)
```

```
trait_nMDS_prob_fo <- ggplot() +
  geom_polygon(data=hull.data,
               aes(x=NMDS1, y=NMDS2,
                   fill= prob.clust.num,
                   group= prob.clust.num),
               alpha=0.30) + # add the convex hulls
  geom_point(data=treatment.scores,
             aes(x=NMDS1, y=NMDS2, colour= maxFO),
             size=3) + # add the point markers
  #geom_text(data=hull.data,
  #          aes(x=NMDS1, y=NMDS2, label = prob.clust.num)) +
  coord_equal() +
  theme_bw()  +
  theme(axis.text.x = element_blank(),  # remove x-axis text
        axis.text.y = element_blank(), # remove y-axis text
        axis.ticks = element_blank(),  # remove axis ticks
        axis.title.x = element_text(size=18), # remove x-axis labels
        axis.title.y = element_text(size=18), # remove y-axis labels
        legend.title = element_text(size = 18),
        legend.text = element_text(size = 18),
        legend.justification = c(0,0.5),
        #panel.background = element_rect(fill = "lightgrey"),
```

```
        panel.grid.major = element_blank(),  #remove major-grid labels
        panel.grid.minor = element_blank(),  #remove minor-grid labels
        plot.background = element_blank()) +
  scale_colour_viridis(option="magma", begin = 0, end = 1, name = "Frequency of Occurrence (%)") +
  scale_fill_viridis(option="magma", begin = 0.9, end = 0, name = "Cluster Number") #+

trait_nMDS_prob_fo
```




```
#ggsave(here("outputs_figures/clusters/trait_nMDS_prob_fo.png"),
#       plot = trait_nMDS_prob_fo, width = 8, height = 8, dpi = 300)
```