assignment_four_radcliffe

Don Radcliffe

3/7/2022

#1

```r
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```r
require(ggordiplots)
```

```
## Loading required package: ggordiplots

## Loading required package: vegan

## Loading required package: permute

## Loading required package: lattice

## This is vegan 2.5-7

## Loading required package: glue
```

```r
require(snakecase)
```

```
## Loading required package: snakecase
```

```r
require(dendextend)
```

```
## Loading required package: dendextend

## Registered S3 method overwritten by 'dendextend':
##   method     from
##   rev.hclust vegan

##
## ---------------------
## Welcome to dendextend version 1.15.2
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## You may ask questions at stackoverflow, use the r and dendextend tags:
##   https://stackoverflow.com/questions/tagged/dendextend
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------

##
## Attaching package: 'dendextend'
## The following object is masked from 'package:permute':
##
##     shuffle

## The following object is masked from 'package:stats':
##
##     cutree
```

```
require(viridis)
```

```
## Loading required package: viridis

## Loading required package: viridisLite
```

```
require(vegan)
require(labdsv)
```

```
## Loading required package: labdsv

## Loading required package: mgcv

## Loading required package: nlme

## This is mgcv 1.8-38. For overview type 'help("mgcv-package")'.

## This is labdsv 2.0-1
## convert existing ordinations with as.dsvord()

##
## Attaching package: 'labdsv'

## The following object is masked from 'package:stats':
##
##     density
```

```
require(tibble)
```

```
## Loading required package: tibble
```

```
require(tidyr)
```

```
## Loading required package: tidyr
```

```
require(stringr)
```

```
## Loading required package: stringr
```

```
require(dplyr)
```

```
## Loading required package: dplyr

##
## Attaching package: 'dplyr'

## The following object is masked from 'package:nlme':
##
##     collapse

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

## The following objects are masked from 'package:base':
##
```

```
##     intersect, setdiff, setequal, union
```

```
require(here)
```

```
## Loading required package: here

## here() starts at C:/ProgramR/sefs502/lecture_1
```

```
oak_raw <- read.csv(here('data/Oak_data_47x216.csv'), stringsAsFactors = TRUE) %>%
  rename('stand' = X)
## Just a few rows and columns to prove it's read in:
head(oak_raw[,1:5])
```

```
##     stand Elev.m LatAppx LongAppx Slope.deg
## 1 Stand01     91 44.4831 123.3618         1
## 2 Stand02    106 44.6613 123.2029         7
## 3 Stand03    152 44.7625 123.2641         6
## 4 Stand04     91 45.0081 123.2233         5
## 5 Stand05    152 45.1381 123.3355        10
## 6 Stand06     91 44.3974 123.3215         5
```

Subsetting out the draw and valley bottom landforms to focus on the ridge and slope landforms.

```
oak_subset <- oak_raw %>%
  filter(Landform == 'Ridge' | Landform == 'Slope') %>%
  column_to_rownames(var = 'stand')
head(oak_subset[,1:5])
```

```
##         Elev.m LatAppx LongAppx Slope.deg AspClass
## Stand01     91 44.4831 123.3618         1 SE.or.NW
## Stand02    106 44.6613 123.2029         7 SE.or.NW
## Stand03    152 44.7625 123.2641         6   S.or.W
## Stand04     91 45.0081 123.2233         5   N.or.E
## Stand05    152 45.1381 123.3355        10   N.or.E
## Stand06     91 44.3974 123.3215         5   N.or.E
```

Necessary adjustments: selecting only the response variables for running the analyses, with stands as the rownames. Subsetting out rare species, to reduce potential for spurious noise caused by species that don't have a major influence on other species, with the commonly used 5% threshold. Relativizing species by maximum, in order to account for having trees, shrubs, and herbs in the same dataset, with different scales of measurement (different than my choice in assignment three).

```
oak_subset_species <- oak_subset %>%
  select(Abgr.s:Zice) %>%
  vegtab(minval = 0.05 * nrow(oak_subset)) %>%
  decostand(method = 'max') %>%
  mutate(across(where(is.numeric), round, 3))
head(oak_subset_species[,1:5])
```

```
##         Quga.t Syal.s Rhdi.s Amal.s Roeg.s
## Stand01  0.687  0.014  0.616  0.045  0.130
## Stand02  0.532  0.167  0.500  0.030  0.087
## Stand03  1.000  0.056  0.360  0.045  0.087
## Stand04  0.532  0.403  0.291  0.343  0.348
## Stand05  0.408  0.292  0.006  0.045  0.130
## Stand06  0.506  0.250  0.151  0.179  0.304
```

Creating a bray-curtis distance matrix because we have compositional data and this is to my knowledge the most commonly used distance measure with compositional data.

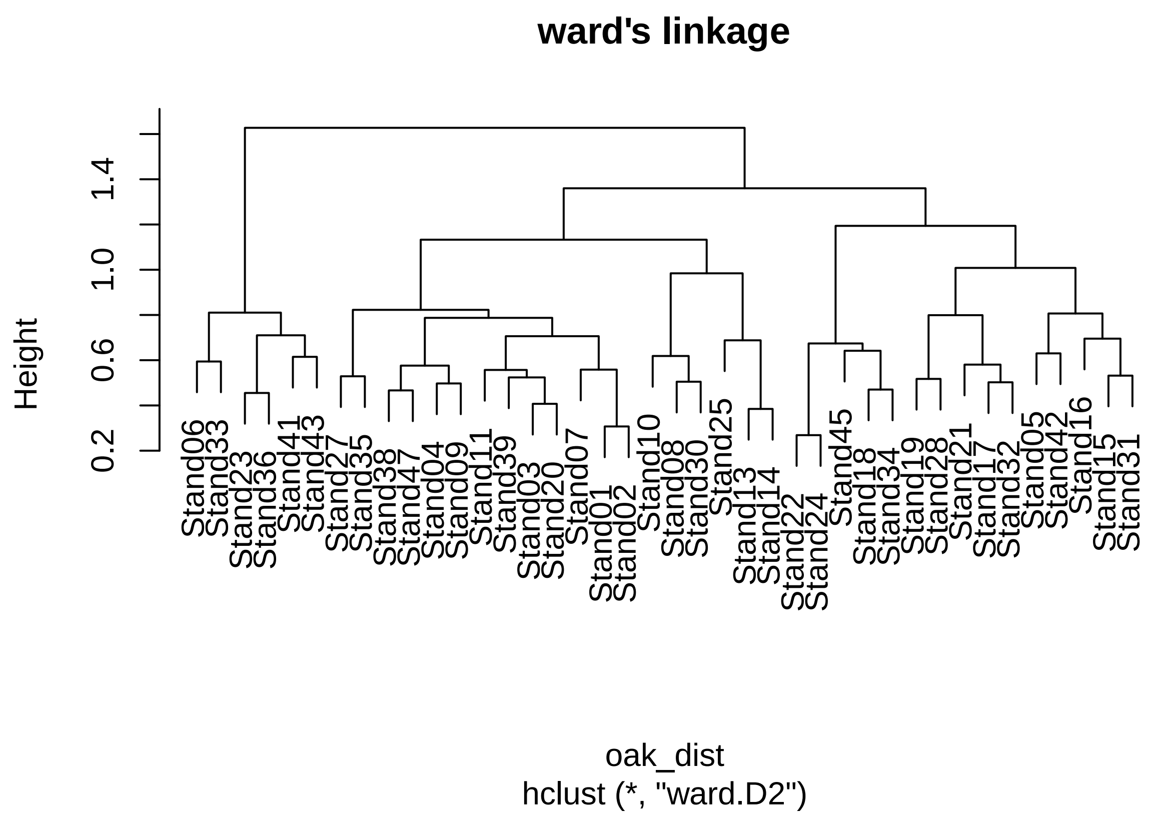
```r
oak_dist <- vegdist(oak_subset_species, method = 'bray')
str(oak_dist)
```

```
##  'dist' num [1:780] 0.307 0.443 0.616 0.846 0.576 ...
##  - attr(*, "Size")= int 40
##  - attr(*, "Labels")= chr [1:40] "Stand01" "Stand02" "Stand03" "Stand04" ...
##  - attr(*, "Diag")= logi FALSE
##  - attr(*, "Upper")= logi FALSE
##  - attr(*, "method")= chr "bray"
##  - attr(*, "call")= language vegdist(x = oak_subset_species, method = "bray")
```

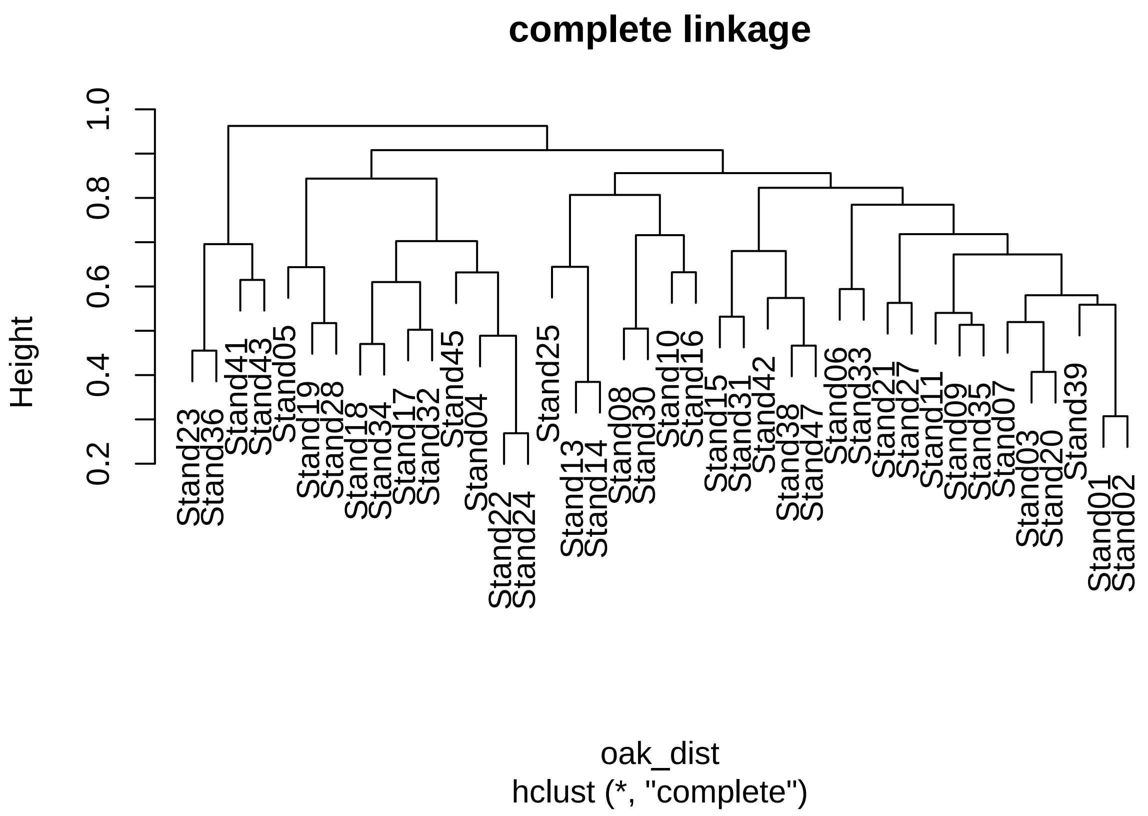
#2

I'm using Ward's linkage because the notes/Singh et al. 2011/McCune & Grace (2002) recommend it. In this context, I'm trying complete linkage as a 'non-recommended' method, just to see how the result may differ, and because both methods seem to me like they should result in relatively evenly dispersed groups.

```r
#Linkage 1: Wards minimum variance method
hclust_ward <- hclust(oak_dist, method = "ward.D2")
plot(hclust_ward, main ="ward's linkage")
```



```r
#Linkage 2: Group average
hclust_complete <- hclust(oak_dist, method = "complete")
plot(hclust_complete, main = 'complete linkage')
```



These methods are both agglomerative: they start with each stand as a separate group and merge groups together. Both methods are hierarchical: they proceed in a series of discrete iterative steps, as opposed to simultaneously. Both methods are polythetic: all variables are used by the clustering algorithm, via the distance matrix.

Ward's method seeks to minimize variance within and between groups. It calculates centroids and the sum of squares of the variance from the centroids for each hypothetical grouping, and then chooses the goruping that results in the least variance. According to the notes, it is not supposed to be used with Bray-Curtis distance, which is what I used, so I'm interested to see how it compares with complete linkage.

Complete linkage joins groups based on the distance between the furthest sample units within each group. Therefore, it will favor joining smaller groups, starting with pairings of single sample units.

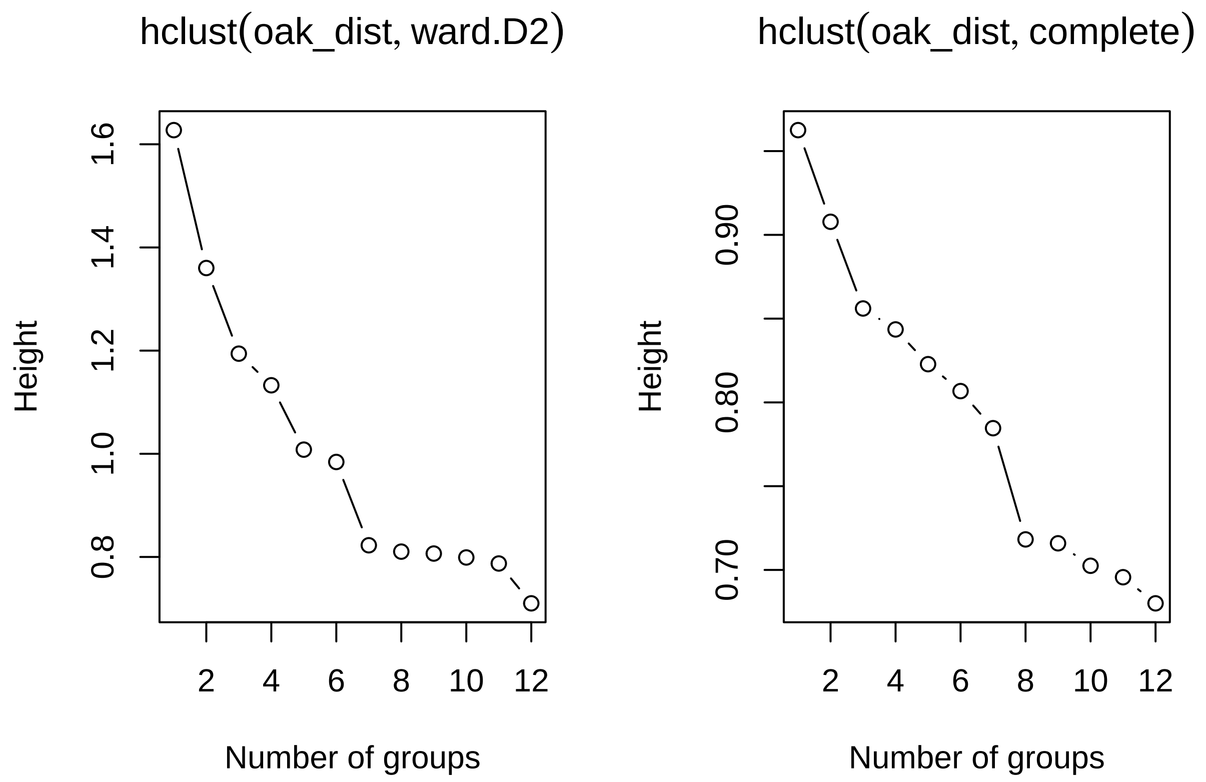
```
scree <- function(hclust_object, max.size = 12) {
  temp <- with(hclust_object, cbind(height, merge))
  colnames(temp) <- c("Height", "JoinsThis", "WithThis")
  plot(x = max.size:1,
       y = temp[(nrow(temp)-(max.size-1)):nrow(temp), 1],
       xlab = "Number of groups", ylab = "Height", main = hclust_object$call, type = "b")
  tail(temp, max.size)
}

par(mfrow = c(1,2))

scree(hclust_ward)
```

```
##          Height JoinsThis WithThis
## [28,] 0.7101006         5       20
## [29,] 0.7873037        17       27
## [30,] 0.7990257        11       18
## [31,] 0.8065980        22       26
## [32,] 0.8103450        19       28
## [33,] 0.8226339        13       29
## [34,] 0.9841162        21       25
## [35,] 1.0080628        30       31
## [36,] 1.1328219        33       34
## [37,] 1.1940918        24       35
## [38,] 1.3601180        36       37
## [39,] 1.6274168        32       38
```

```
scree(hclust_complete)
```



```
##          Height JoinsThis WithThis
## [28,] 0.6800681        14       18
## [29,] 0.6956504         5       22
## [30,] 0.7024918        21       23
## [31,] 0.7158962        10       24
## [32,] 0.7182296        17       27
## [33,] 0.7846109        20       32
## [34,] 0.8067463        26       31
## [35,] 0.8228216        28       33
## [36,] 0.8435453        25       30
## [37,] 0.8560482        34       35
## [38,] 0.9078024        36       37
## [39,] 0.9625066        29       38
```

Seven groups appears to be optimal for Ward's linkage, while eight is better for complete linkage. This is based on the sharp inflection of the lines at these points in each respective scree plot. I'm going to present both with eight groups for the sake of direct comparison.

```r
wards_eight <- cutree(hclust_ward, k = 8)
complete_eight <- cutree(hclust_complete, k = 8)

## Bind these groups to our species dataframe
wards_eight_df <- data.frame(wards_eight)
complete_eight_df <- data.frame(complete_eight)
```

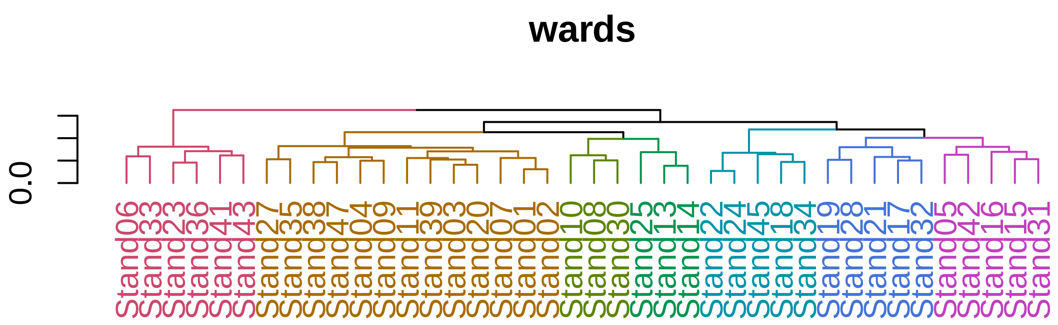
There indeed appears to be differences between groupings of the two linkage methods. Interestingly, ward's has greater heights than complete linkage.

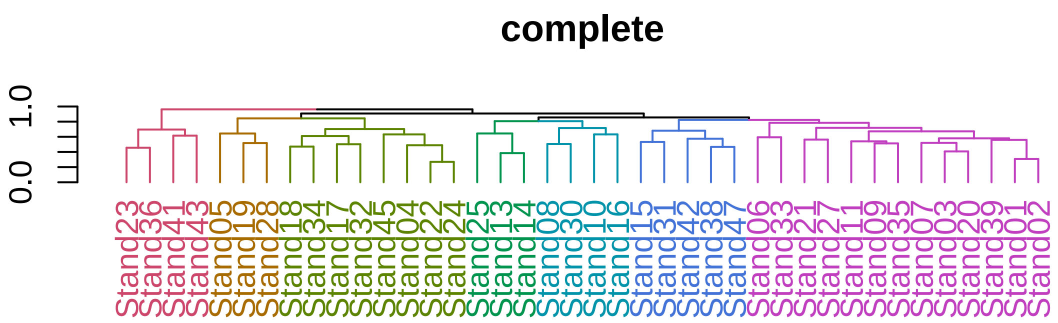
```r
wards_eight_dendrogram <- as.dendrogram(hclust_ward)
complete_eight_dendrogram <- as.dendrogram(hclust_complete)

par(mfrow = c(2,1))

wards_eight_dendrogram %>%
  set("branches_k_color", k = 7) %>%
  set("labels_col", k = 7) %>%
  plot(main = 'wards')

complete_eight_dendrogram %>%
  set("branches_k_color", k = 7) %>%
  set("labels_col", k = 7) %>%
  plot(main = 'complete')
```





The clusters appear to be significantly different according to permanova, whether grouped by wards method or complete linkage.

```
adonis2(oak_subset_species ~ wards_eight, data = wards_eight_df, method = 'bray')
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = oak_subset_species ~ wards_eight, data = wards_eight_df, method = "bray")
##             Df SumOfSqs      R2      F Pr(>F)
## wards_eight  1   0.8842 0.08401 3.4852  0.001 ***
## Residual    38   9.6403 0.91599
## Total       39  10.5245 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
adonis2(oak_subset_species ~ complete_eight, data = complete_eight_df, method = 'bray')
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = oak_subset_species ~ complete_eight, data = complete_eight_df, method = "bray")
##                Df SumOfSqs      R2      F Pr(>F)
## complete_eight  1   0.7450 0.07079 2.8948  0.001 ***
## Residual       38   9.7795 0.92921
## Total          39  10.5245 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

#3

3A a) I'm using autotransform = FALSE, because I've made necessary data adjustments above, including standardization by maximum. I want to maintain more control and understanding of my data transformations.

b) I'm choosing a Bray-Curtis distance because this is commonly used with composition data, and it's what I used for cluster analysis.

c) I'm choosing a maxit of 400, in accordance with recommendations in McCune and Grace (2002)

d) I'm choosing a try of 20, the same as the metaMDS() default, and a trymax of 40, in accordance with McCune and Grace (2002) recommendations.

```r
## Function with an argument for axes
## Run on the dataframe relativized by maximum
run_nmds <- function(axes){
  metaMDS(comm = oak_subset_species,
           k = axes,
            autotransform = FALSE,
            distance = "bray",
            model = "global",
            maxit = 400,
            try = 20, trymax = 40)
}

nmds_three_axes <- run_nmds(axes = 3)
```

```
## Run 0 stress 0.1485504
## Run 1 stress 0.1485971
## ... Procrustes: rmse 0.01428972  max resid 0.06242241
## Run 2 stress 0.1485509
## ... Procrustes: rmse 0.03921212  max resid 0.1577995
## Run 3 stress 0.1429914
## ... New best solution
## ... Procrustes: rmse 0.07294654  max resid 0.2154578
## Run 4 stress 0.145616
## Run 5 stress 0.1492382
## Run 6 stress 0.142991
## ... New best solution
## ... Procrustes: rmse 0.0003910897  max resid 0.001311083
## ... Similar to previous best
## Run 7 stress 0.1485505
## Run 8 stress 0.145616
## Run 9 stress 0.1456161
## Run 10 stress 0.142991
## ... New best solution
## ... Procrustes: rmse 3.993372e-05  max resid 0.000174982
## ... Similar to previous best
## Run 11 stress 0.1429912
## ... Procrustes: rmse 0.0001295082  max resid 0.000590369
## ... Similar to previous best
## Run 12 stress 0.1485494
## Run 13 stress 0.1525516
## Run 14 stress 0.1429909
## ... New best solution
## ... Procrustes: rmse 0.0001009885  max resid 0.0004384476
## ... Similar to previous best
## Run 15 stress 0.1429911
## ... Procrustes: rmse 0.0002405173  max resid 0.0007340852
## ... Similar to previous best
## Run 16 stress 0.1429909
## ... Procrustes: rmse 4.909361e-05  max resid 0.0001283595
## ... Similar to previous best
## Run 17 stress 0.142991
## ... Procrustes: rmse 0.0002427561  max resid 0.0009349434
## ... Similar to previous best
## Run 18 stress 0.1429911
## ... Procrustes: rmse 0.0001873297  max resid 0.0008671652
## ... Similar to previous best
## Run 19 stress 0.1429911
## ... Procrustes: rmse 0.000207359  max resid 0.0009664622
## ... Similar to previous best
## Run 20 stress 0.1429912
## ... Procrustes: rmse 0.0003334117  max resid 0.001279861
## ... Similar to previous best
## *** Solution reached
```

```
nmds_two_axes <- run_nmds(axes = 2)
```

```
## Run 0 stress 0.2000685
## Run 1 stress 0.2051283
## Run 2 stress 0.2093357
## Run 3 stress 0.2116947
## Run 4 stress 0.2149998
## Run 5 stress 0.2000684
## ... New best solution
## ... Procrustes: rmse 0.0001354944  max resid 0.000621655
## ... Similar to previous best
## Run 6 stress 0.2135276
## Run 7 stress 0.2294453
## Run 8 stress 0.217723
## Run 9 stress 0.2275368
## Run 10 stress 0.2097542
## Run 11 stress 0.2114706
## Run 12 stress 0.2097545
## Run 13 stress 0.2167524
## Run 14 stress 0.2154567
## Run 15 stress 0.225115
## Run 16 stress 0.2117551
## Run 17 stress 0.2145137
## Run 18 stress 0.21781
## Run 19 stress 0.2093358
## Run 20 stress 0.2210944
## *** Solution reached
```

```r
nmds_one_axis <- run_nmds(axes = 1)
```

```
## Run 0 stress 0.3157839
## Run 1 stress 0.5522932
## Run 2 stress 0.4711536
## Run 3 stress 0.4971475
## Run 4 stress 0.4004267
## Run 5 stress 0.5467947
## Run 6 stress 0.3170573
## Run 7 stress 0.3910674
## Run 8 stress 0.5626104
## Run 9 stress 0.4741882
## Run 10 stress 0.5017887
## Run 11 stress 0.3666329
## Run 12 stress 0.4945502
## Run 13 stress 0.4297968
## Run 14 stress 0.5574738
## Run 15 stress 0.5055988
## Run 16 stress 0.549088
## Run 17 stress 0.5333844
## Run 18 stress 0.3721859
## Run 19 stress 0.3160092
## ... Procrustes: rmse 0.01766099  max resid 0.04773635
## Run 20 stress 0.4565914
## Run 21 stress 0.4838658
## Run 22 stress 0.4783388
## Run 23 stress 0.4823599
## Run 24 stress 0.4679866
## Run 25 stress 0.4812255
## Run 26 stress 0.5539986
## Run 27 stress 0.4501672
## Run 28 stress 0.318957
## Run 29 stress 0.4585498
## Run 30 stress 0.4669767
## Run 31 stress 0.5616291
## Run 32 stress 0.3185147
## Run 33 stress 0.562283
## Run 34 stress 0.4048218
## Run 35 stress 0.5608938
## Run 36 stress 0.481539
## Run 37 stress 0.3172458
## Run 38 stress 0.5553421
## Run 39 stress 0.5514529
## Run 40 stress 0.4808974
## *** No convergence -- monoMDS stopping criteria:
##      1: stress ratio > sratmax
##     39: scale factor of the gradient < sfgrmin
```

3B We can see from the dataframe produced below how stress increases as the number of axes decrease, although not as dramatically as I may have expected for one axis. Three axes has the least stress (0.143), and the stress of the two axis ordination (0.20) is getting close to being potentially misleading, so I'll use a three axis ordination.

```r
## Build a dataframe with the stress values of each nmds
nmds_stress <- data.frame(
  'axes' = c(3, 2, 1),
  'stress' = c(nmds_three_axes$stress, nmds_two_axes$stress, nmds_one_axis$stress)
)

nmds_stress
```

```
##   axes    stress
## 1    3 0.1429909
## 2    2 0.2000684
## 3    1 0.3157839
```

3C

```
final_nmds <- metaMDS(comm = oak_subset_species,
             k = 3,
             autotransform = FALSE,
             distance = "bray",
             model = "global",
             ## monoMDS is the default, but putting it here to clarify random starts
             engine = c('monoMDS'),
             maxit = 800,
             try = 20, trymax = 40)
```

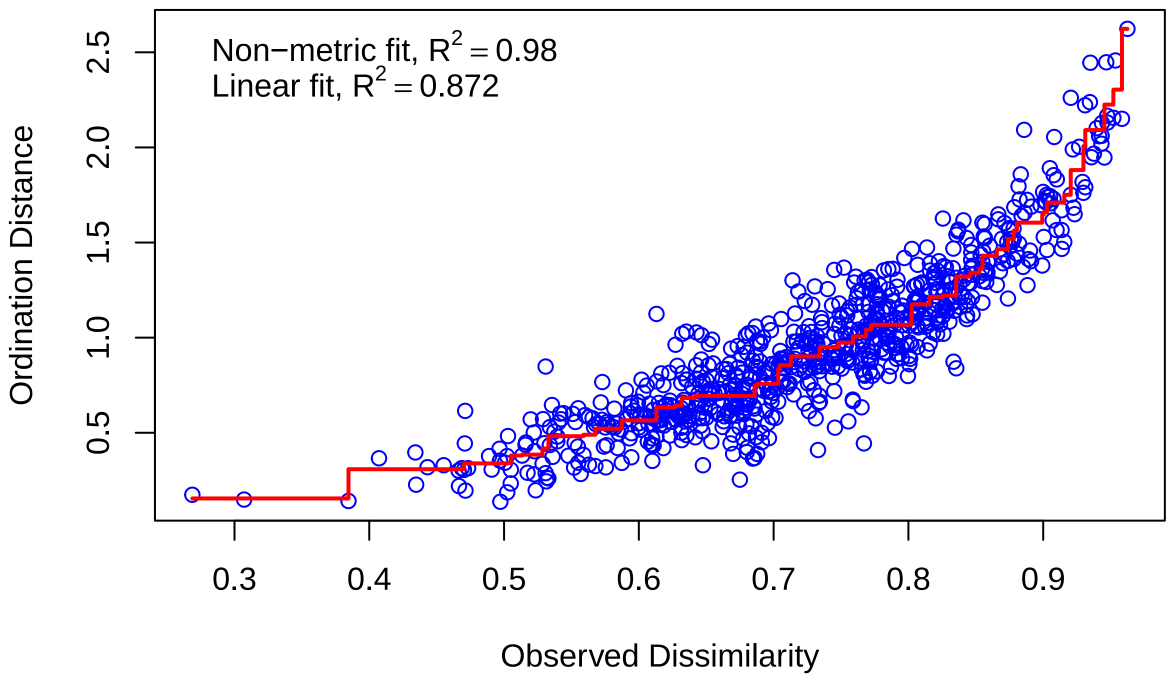
```
## Run 0 stress 0.1485504
## Run 1 stress 0.1485501
## ... New best solution
## ... Procrustes: rmse 0.03907273  max resid 0.1574718
## Run 2 stress 0.1485501
## ... New best solution
## ... Procrustes: rmse 0.001730779  max resid 0.006080533
## ... Similar to previous best
## Run 3 stress 0.1429914
## ... New best solution
## ... Procrustes: rmse 0.08708158  max resid 0.2224533
## Run 4 stress 0.145616
## Run 5 stress 0.1429911
## ... New best solution
## ... Procrustes: rmse 0.0003904515  max resid 0.001148517
## ... Similar to previous best
## Run 6 stress 0.142991
## ... New best solution
## ... Procrustes: rmse 0.0004117721  max resid 0.001259418
## ... Similar to previous best
## Run 7 stress 0.142991
## ... New best solution
## ... Procrustes: rmse 0.0002810638  max resid 0.0008104448
## ... Similar to previous best
## Run 8 stress 0.1491785
## Run 9 stress 0.1429913
## ... Procrustes: rmse 0.0003402123  max resid 0.001244306
## ... Similar to previous best
## Run 10 stress 0.1429911
## ... Procrustes: rmse 0.0001154083  max resid 0.000352422
## ... Similar to previous best
## Run 11 stress 0.1525032
## Run 12 stress 0.1525013
## Run 13 stress 0.1429912
## ... Procrustes: rmse 0.0002921752  max resid 0.0009245662
## ... Similar to previous best
## Run 14 stress 0.1485964
## Run 15 stress 0.1485511
## Run 16 stress 0.142991
## ... Procrustes: rmse 5.402755e-05  max resid 0.0002402324
## ... Similar to previous best
## Run 17 stress 0.145616
## Run 18 stress 0.142991
## ... Procrustes: rmse 0.0001199069  max resid 0.0004734042
## ... Similar to previous best
## Run 19 stress 0.1485499
## Run 20 stress 0.1429909
## ... New best solution
## ... Procrustes: rmse 0.000117168  max resid 0.0005145246
## ... Similar to previous best
## *** Solution reached
```

```
final_nmds$stress
```

```
## [1] 0.1429909
```

3D Both the linear and non-metric fits are above 0.85, which seems quite good to me, although I feel like I always see high r squared values in Shepard plots so I'm not entirely sure of how much weight I should place on high r squared values.

```
stressplot(final_nmds)
```

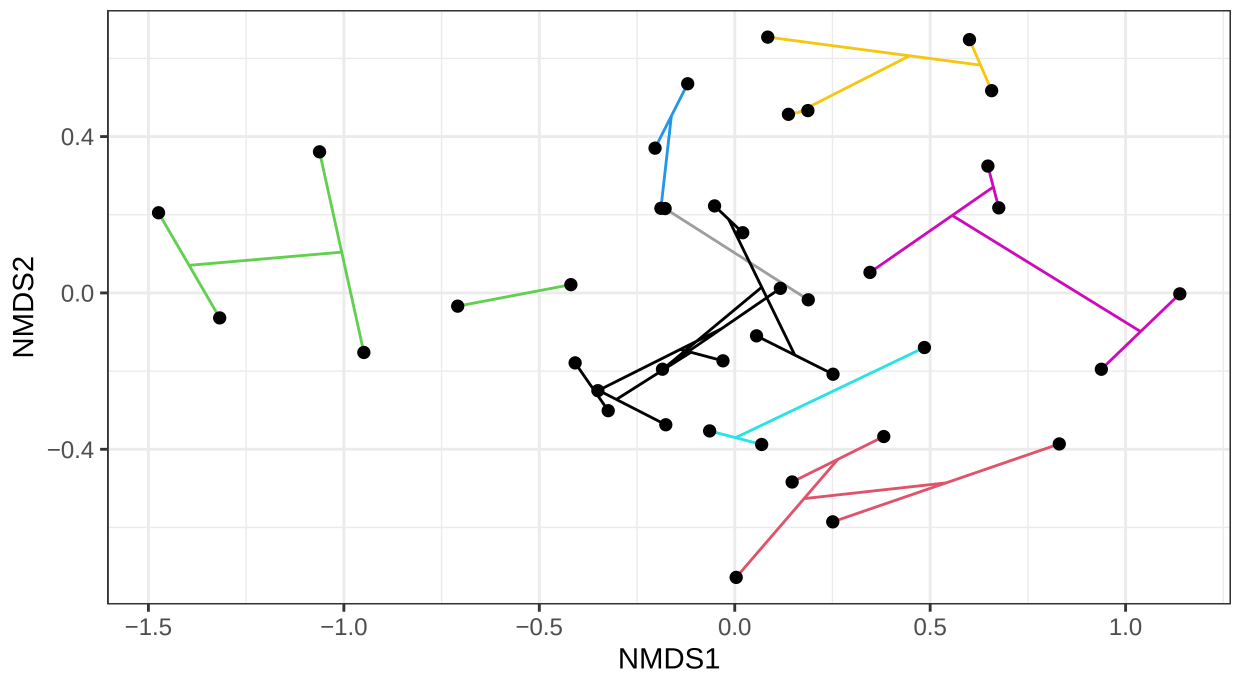


3E Since there were separate questions for the cluster overlay and the final graph, I decided to try gg_ordicluster() for 3E. I think the gg_ordiplot results colored by cluster analysis group look better, however.

I think the overlay of the (ward's distance) clusters supports the highly significant p-value of the PERMANOVA analysis; it appears visually that the cluster analysis did very good job of separating groups in agreement with at least two of three axes in the ordination space.

```
ordicluster <- gg_ordicluster(final_nmds, hclust_ward,
                              prune = 8,
                              col = cutree(hclust_ward, 8),
                              pt.size = 2,
                              plot = FALSE)

ordicluster$plot +
  theme_bw()
```



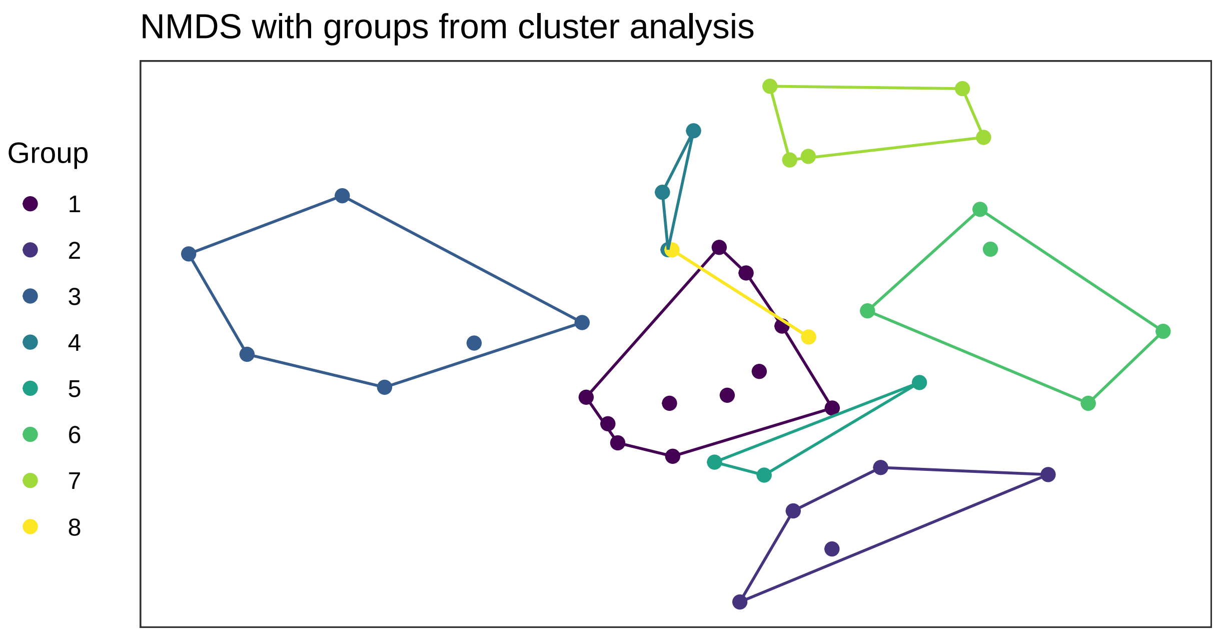
```
ordiplot <- gg_ordiplot(final_nmds, groups = wards_eight_df$wards_eight,
                        hull = TRUE, ellipse = FALSE, spiders = FALSE,
                        plot = FALSE, pt.size = 2)
```

```
## Warning in chol.default(cov, pivot = TRUE): the matrix is either rank-deficient
## or indefinite

## Warning in chol.default(cov, pivot = TRUE): the matrix is either rank-deficient
## or indefinite
```

```
points <- ordiplot$df_ord

ordiplot$plot +
  ggtitle('NMDS with groups from cluster analysis') +
  ## Tried to add characters for landform here, but couldn't get legend for pch to show since I have to
  ## It showed that landform doesn't appear informative of my hclust() groups.
  #geom_point(data = points, aes(x = x, y = y, color = Group), size = 2, pch = oak_subset$Landform, sho
  ## Hard to tell the most adjacent groups from one another
  ## but the group numbers should mean something here so the gradient is nice.
  scale_color_viridis(discrete = TRUE) +
  labs(caption = 'Three axis NMDS of oak dataset, restricted to ridge and slope landforms,
             \n grouped by hierarchical agglomerative clustering with Wards method.
             \n The group size of eight was chosen with a scree plot.') +
  theme_bw() +
  theme(legend.position = 'left') +
  theme(plot.caption = element_text(hjust = 0, lineheight = 0.6)) +
  theme(axis.title = element_blank(),
        axis.ticks = element_blank(),
        axis.text = element_blank(),
        panel.grid = element_blank())
```



Three axis NMDS of oak dataset, restricted to ridge and slope landforms,
grouped by hierarchical agglomerative clustering with Wards method.
The group size of eight was chosen with a scree plot.

```
## To truly make this publication quality, I would ggsave() the graph to jpg,
## but I'm keeping it in the Markdown document for simplicity.
```

3G To visualize the variation in stand-level composition of the oak dataset, I used a Non-Metric Multidimensional Scaling (NMDS) ordination. This was run using the vegan() package in R with the metaMDS() function with the 'monoMDS' engine. The number of axes was chosen based on observing stress values of NMDS analyses run with one, two, and three axes, three was found to be the minimum number of axes needed to achieve a stress value I deemed acceptable (<0.2). 800 iterations were allowed in each local run, with 20 minimum and 40 maximum global iterations. The final ordination showed a stress value of 0.143. For the visualization, I plotted the stand-level coordinates of NMDS1 and NMDS2, and colored the points by groups determined in a hierarchical agglomerative clustering analysis using Ward's method of determining group differences, with group size determined by scree plot.