Problem Set 2

Steven Cao

10/19/2019

Part 1, Problem 1

```
# Generate vectors
p <- c(1,2)
q <- c(3,4)

# Calculate Manhattan distance between the p point-pair and the q point-pair
vector_absoluteDifferences = abs(p-q)
manhattan_distance = sum(vector_absoluteDifferences)

# Calculate Canberra distance between the p point-pair and the q point-pair
vector_absoluteDifferences = abs(p-q) # i.e. the numerator
vector_totalFeatureMagnitudes = abs(p) + abs(q) # i.e. the denominator
canberra_distance = sum(vector_absoluteDifferences) / sum(vector_totalFeatureMagnitudes)

# Calculate Euclidean distance between the p point-pair and the q point-pair
vector_differences = p-q # get "a" and "b", as per "a^2 + b^2 = c^2"
c_squared = vector_differences^2 # i.e. the vector is {a^2, b^2}
c_squared = sum(c_squared) # compress from vector to its equivalent scalar
euclidean_distance = sqrt(c_squared) # get c, which is the euclidean distance

manhattan_distance # output is 4
```

```
## [1] 4
```

```
canberra_distance # output is 0.4
```

```
## [1] 0.4
```

```
euclidean_distance # output is 2.8284
```

```
## [1] 2.828427
```

Part 1, Problem 2

```
# Combine the point-pair vectors into a matrix
matrix_points <- rbind(p,q)

# Check distances
check_manhattan_distance = dist(matrix_points, method = "manhattan")
check_canberra_distance = dist(matrix_points, method = "canberra")
check_euclidean_distance = dist(matrix_points, method = "euclidean")

check_manhattan_distance # output is 4
```

```
##   p
## q 4
```

```
check_canberra_distance # output is 0.833
```

```
##           p
## q 0.8333333
```

```
check_euclidean_distance # output is 2.83
```

```
##          p
## q 2.828427
```

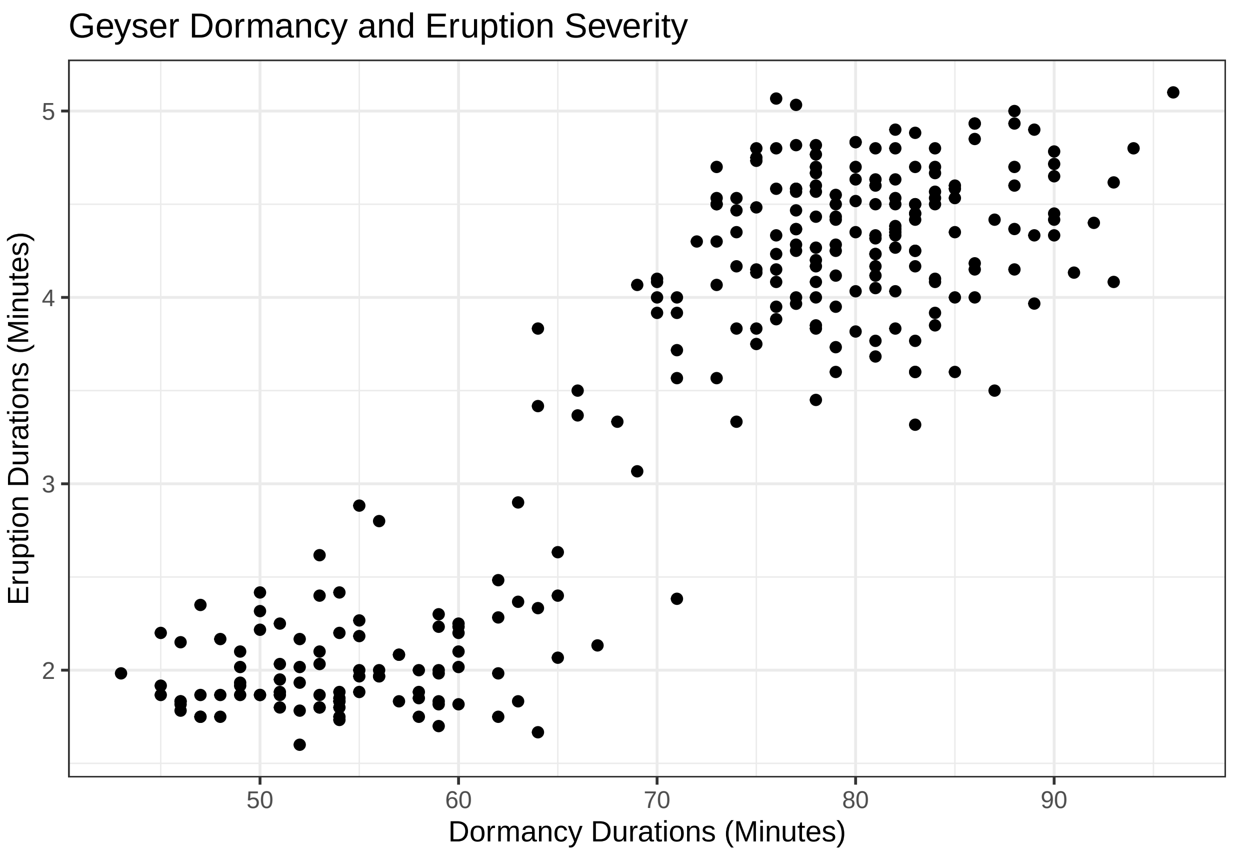
My initial answer for the canberra distanced (0.4) was not correct, compared to the actual answer (0.833). The mistake I made in the code was that I summed the components of vectors individually before dividing, rather than doing element-wise division first and then summing the result. (I.e. I did "sum(vector)/sum(vector) rather than sum(vector/vector)") The answers for the other two measures of distance were correct.

Part 1, Problem 3

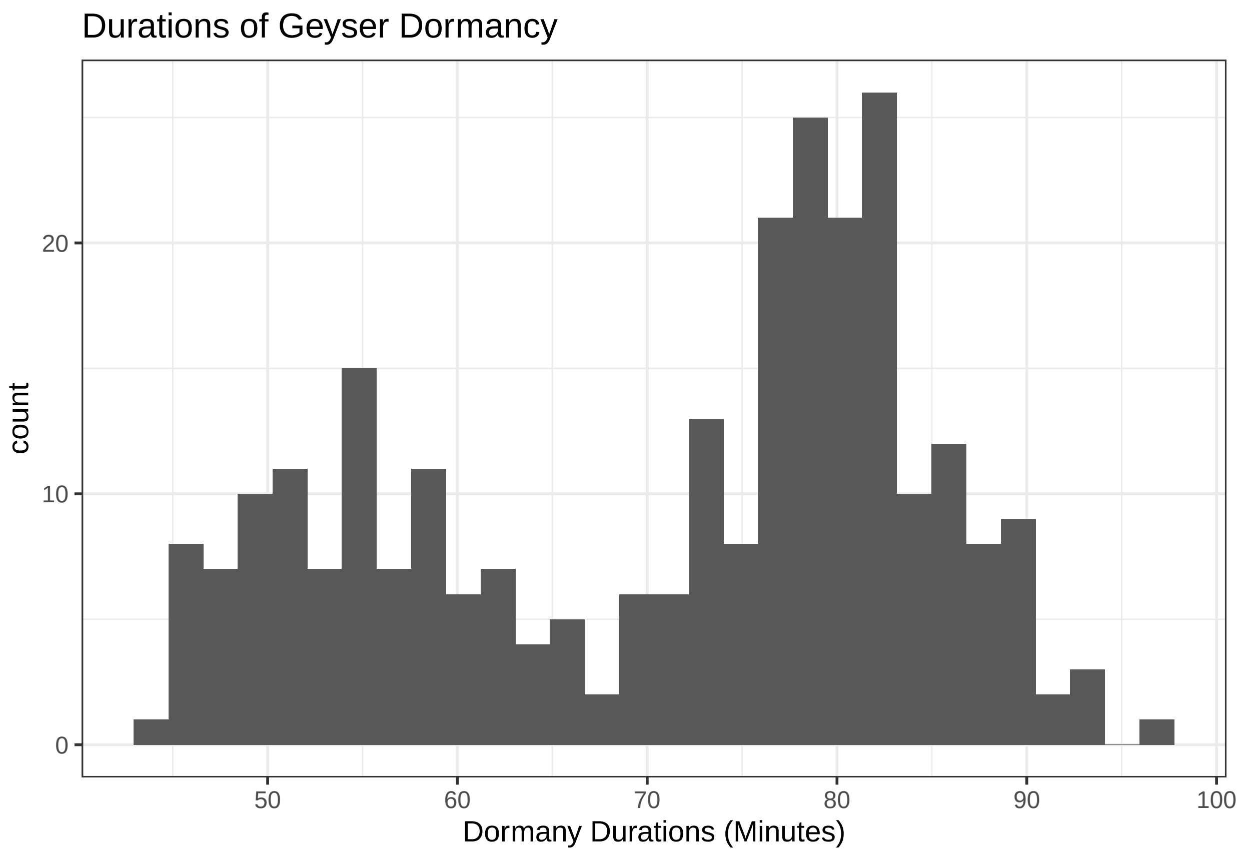
The Euclidean distance is the most natural way to measure distance, whether the dimensions are spatial or represent other features. With this measure of distance, we can capture "interactive effects" between variables, which is in contrast to Manhattan distance. With Manhattan distance, every 1 "unit" of difference regardless of variable type contributes equally to how different two data points are. With Euclidean distance, 10 units of difference in a single variable is considered more distant than 10 units of difference across two variables. Last but not least, the Canberra distance seems to drown out differences based on the magnitudes of the data points themselves. If we take two points and transpose them on the plane, the distance relative to each other (i.e. Euclidean or Manhattan) will remain the same, but the Canberra distance will not - the focus seems to be on the proportion of change relative to the magnitude of the data points themselves. These varying natures of the different kinds of distances would likely be significant depending on the kind of measurements one is performing, along with the domain and nature of the data themselves (e.g. Canberra could be useful for the sensitive detection of changes in faint signals).

Part 1, Problem 4

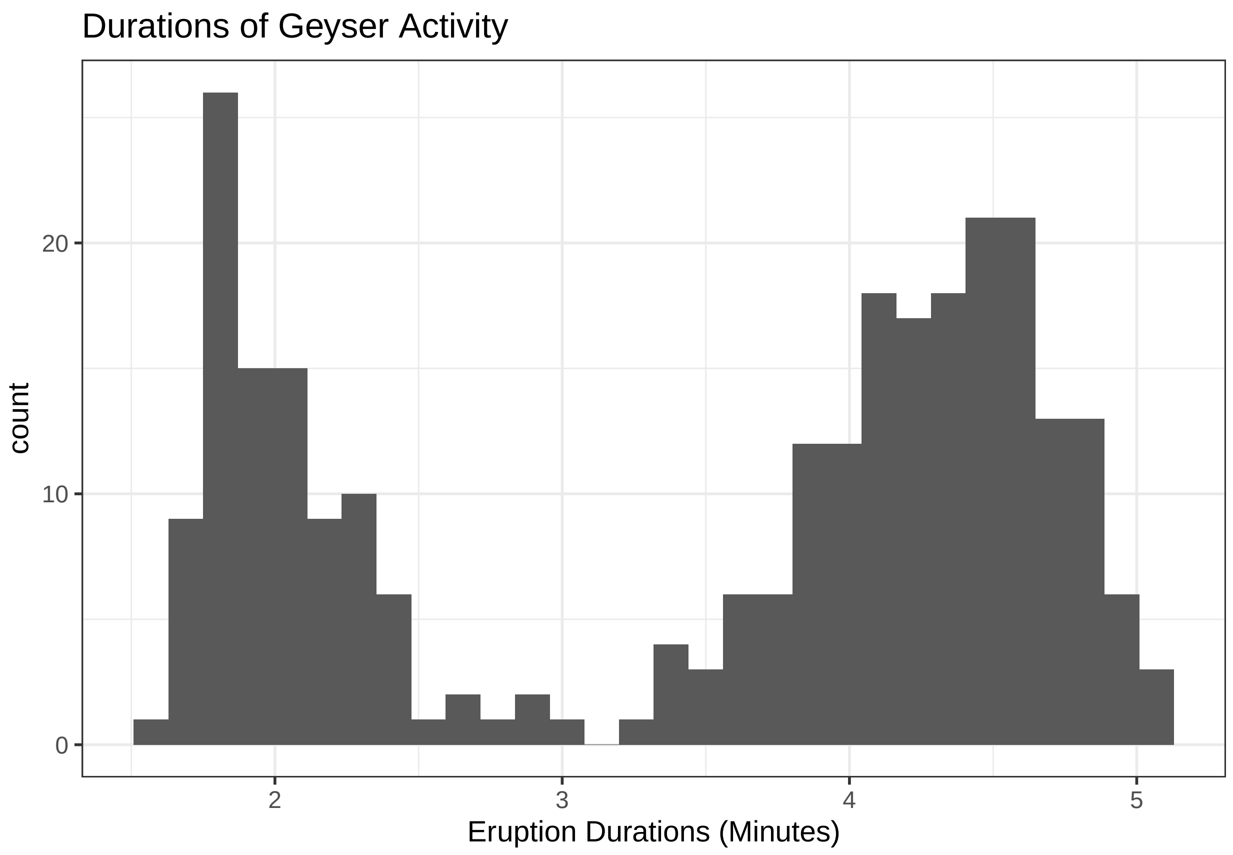
```
## Skim summary statistics
##  n obs: 272
##  n variables: 2
##
## -- Variable type:numeric ------------------------------------------------------------------------
##   variable missing complete   n  mean    sd   p0   p25 p50   p75 p100
##  eruptions       0      272 272  3.49  1.14  1.6  2.16   4  4.45  5.1
##    waiting       0      272 272 70.9  13.59 43   58     76 82    96
##      hist
##  <U+2587><U+2583><U+2581><U+2581><U+2582><U+2585><U+2587><U+2583>
##  <U+2582><U+2585><U+2583><U+2582><U+2585><U+2587><U+2586><U+2582>
```



```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



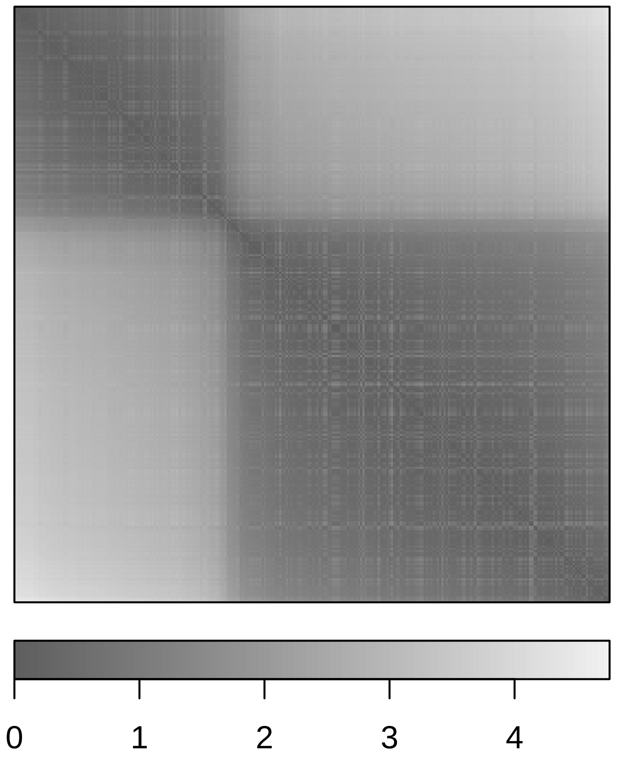
The main takeaways from these plots seems to be that: there are two clusters of data (in the lower-left and upper-right corners, with the upper-right cluster being bigger), and there definitely seems to be a correlation between the duration of dormancy and the duration of eruption.

Part 1, Problem 5

```
# We are assuming that "dissimilarity matrix" and "distance matrix" means the same thing.
# A distance matrix takes a point in some n-dimensional feature space, and then finds the distance
# between that point and all other points. So naturally enough, the output is an n-by-n matrix.

# Standardise (i.e. scale) the data
dataStandardised_geyser_OldFaithful = scale(data_geyser_OldFaithful)

# Get all possible pairwise distances in the form of a distance matrix
distanceMatrix_geyser_OldFaithful = dist(dataStandardised_geyser_OldFaithful, method = "euclidean")

# Quicker way of doing it
# distanceMatrix_geyser_OldFaithful = daisy(data_geyser_OldFaithful, metric = "euclidean", stand = TRUE
```

Part 1, Problem 6

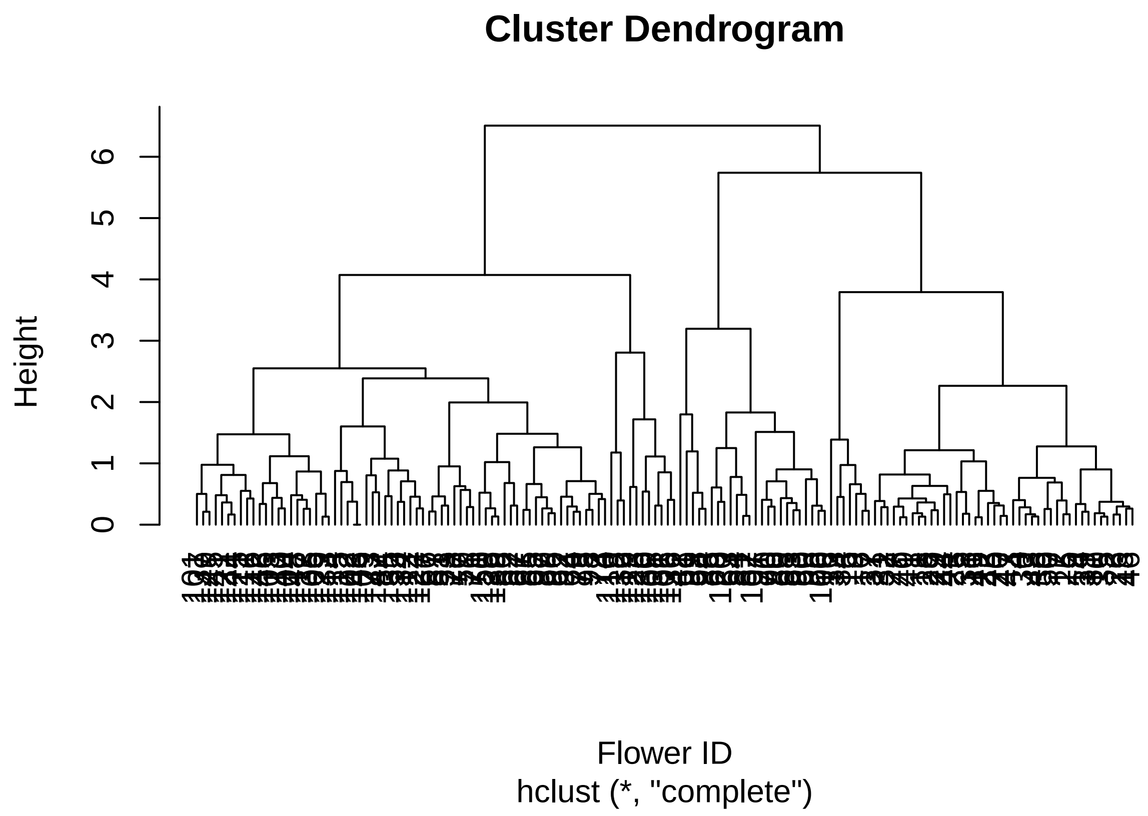
```
## Registered S3 method overwritten by 'seriation':
##   method         from
##   reorder.hclust gclus
```



Part 1, Problem 7

```
# Load the Iris data, scrub the dataset of qualitative variables,
# then perform steps needed to yield a distance matrix.
# For reference, it measures 4+1 variables, all containing 150 observations
distanceMatrix_flowers_Iris <- iris %>%
  dplyr::select( -Species ) %>% # exclude species
  scale() %>% # standardise variables
  dist(method = "euclidean") # and finally calculate the distance matrix
```

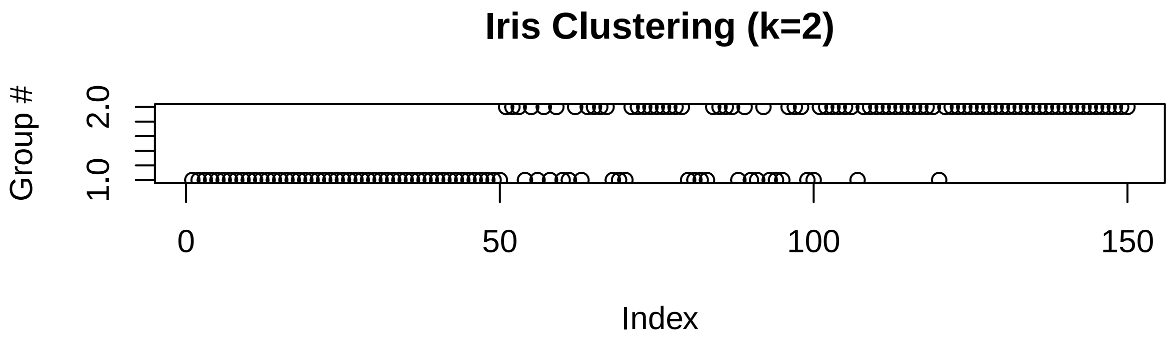
Part 1, Problem 8

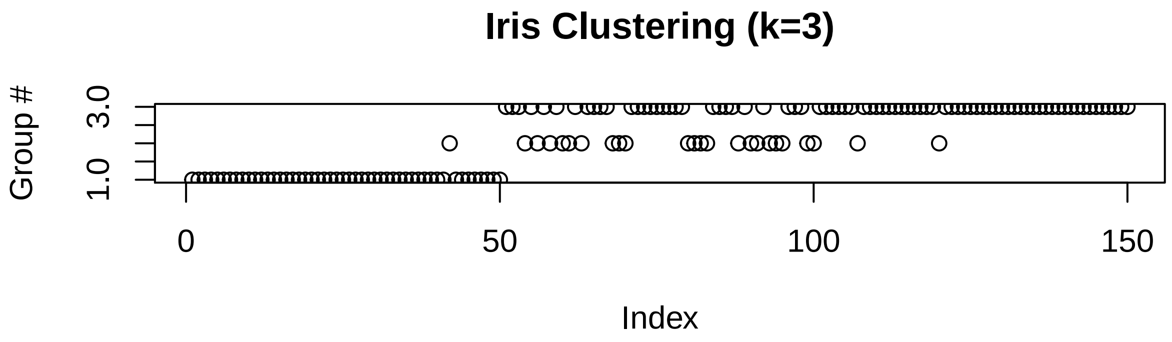


There is a lot of “vague clustering” that goes on “for awhile” (i.e. until around height = 3) where the data points finally coalesce into a handful of major groupings. The final groupings that correspond with “the true nature of the data” (i.e. there being three species) happens at around height = 4, and it is also shown (somewhat) that two of the species resemble each other (the two on the right of the dendrogram) more than the third species (on the left).

Part 1, Problem 9

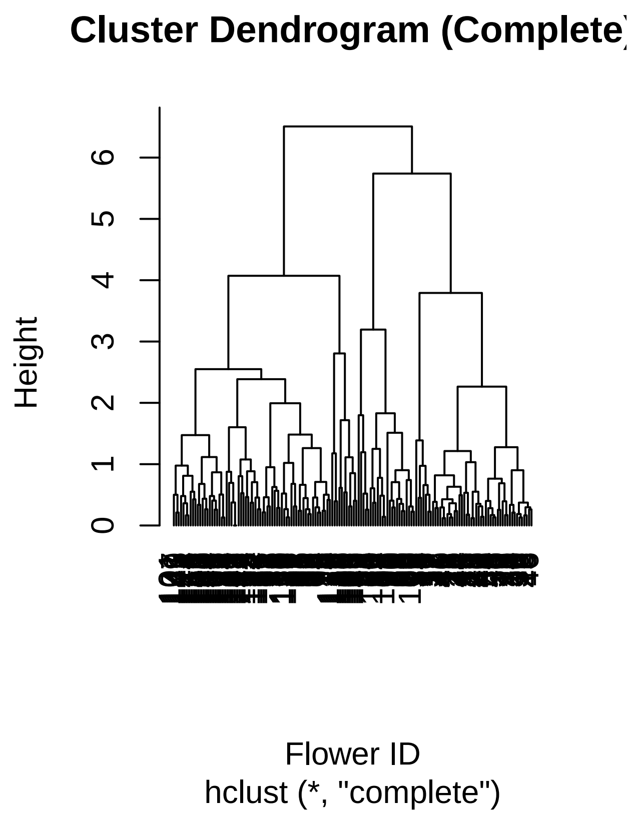
```
##
## ---------------------
## Welcome to dendextend version 1.12.0
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## Or contact: <tal.galili@gmail.com>
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------

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

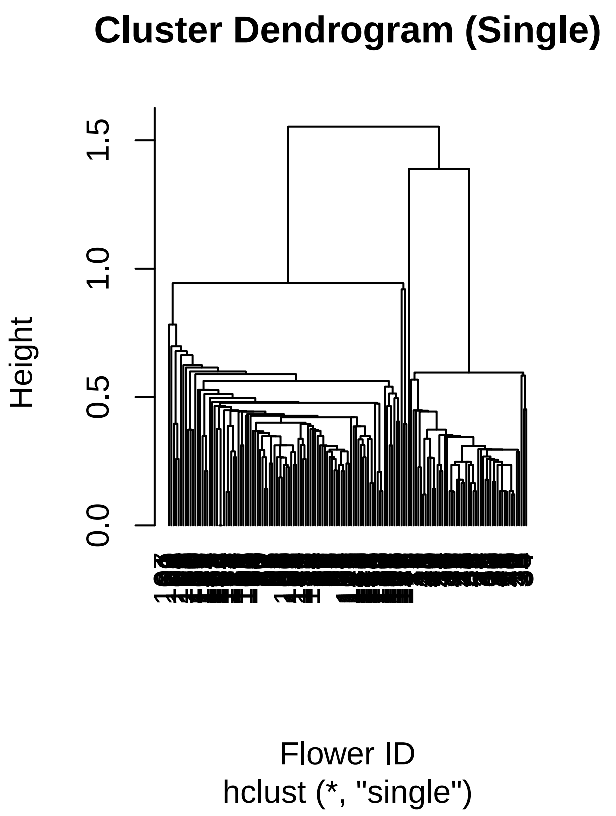




When k=2, there is a high density of data points in "group 1" for the first 50 observations, and in "group 2" for the last 50 observations, with the middle 50 observations being less dense and split half-and-half between the two groups. When k=3, the first and last 50 observations are basically the same, but all the observations from the middle 50 that originally belonged in the first group is now placed into an "intermediate group". So k=3 seems to have a better, more accurate resolution than k=2 (which is expected given the presence of 3 different species).

Part 1, Problem 10





Single-linkage seems to have an overall lesser height (1.5 versus 6) than the complete-linkage dendrogram. That makes sense, considering how single-linkage looks for shortest distances between points, as opposed to complete-linkage, which is the opposite. Likewise, clustering happens very rapidly, and in a short, bursty phase with single-linkage, as opposed to complete-linkage where clustering happens more gradually and less "energetically" over time. What is interesting to note, however, is that despite their differences, both settle at 3 clusters and at 2 clusters at around the same proportion of their total height (i.e. 2/3 of their total height, which is 1.0/1.5 in the single-linkage and 4.0/6.0 in the complete-linkage).

Part 2, Problem 1

a. The first two things I would consider are the aims of the research at hand and the nature of the data. If the research is intended to build and test hypotheses, for example, then we would be imposing and testing models onto the data to see goodness-of-fit rather than trying to create clusters that would already have maximal fit. As for the nature of the data, clustering helps parse out subtle patterns in otherwise messy, noisy, and complicated data. If the data is going to be fairly simple and relatively straightforward, a visualisation would usually suffice in place of clustering techniques. Last but not least, if we are not sampling data from what may be "numerous populations, each with their own eccentricities" (an example being sampling US Citizens which will inadvertently pull from all kinds of subpopulations, like Republicans and Democrats), then there may not be very much reason to suppose any kind of heterogeneity (i.e. disposition for clustering) in the data.
b. The three main techniques are informal visualisation (making a lattice plot comparing every combination of features from the total feature space and looking for clustering in each subplot), creating an ODI plot (and looking for dark, well-defined squares), and mathematically via sparse sampling (checking that the Hopkins test statistic is greater than 0.5).

c. Because of the ambiguiuty that accompanies exploratory data analysis, there is no one magic bullet for any portion of this enterprise. Every technique leaves off somewhere, where another technique can potentially pick up on. If one is dealing with a relatively low feature space, creating a lattice of plots can generally suffice, albeit the ODI is more sensitive to clustering (it is easier to discern squares than clusters) while the Hopkins statistic is more "precise" (it is harder to say which of two similar squares is darker, or which of two similar clusters is more clustery, whereas numbers remain unambiguous to compare). Even though I mention the mathematical route being more "precise", interpreting a number without much context is more difficult than looking at a scatterplot. If any of these methods suggest any semblance of clusterability, that alone might be enough to motivate an attempt in the hopes of finding even the subtlest patterns.

d. If there is little/no support for clusterability, but there is a strong theoretical reason (although I don't mean to insinuate hypothesis-testing) to suppose that there could be clusters (patterns), then such a preconception could allude to subtle patterns hiding in the data. Otherwise, even without such a preconception, it could still be worth going ahead with clustering methods if the project has no other constraints and if the effort required to do so is minimal. (Part of EDA is that it is precisely not known what one will get in advance, and so will be open to surprises.)

Part 2, Problem 2

(Article for reference can be found here: https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/meet.14504301312) a. The purpose of the research was to find naturalistic boundaries (beginnings and ends) to internet browsing sessions (use of AOL search engine). They collected user IDs, and the time in between search engine queries associated with each user ID. These data points were then clustered based not directly on the absolute pause time (because it would be better for it to be user-unique) but on the "z-scores" (very crudely speaking) of the user's pause times. Pause times that amounted to low "z-scores" were clustered together, and apart from pause times that amounted to high "z-scores". The clustering technique, because it iterates through all pause times in order of "z-score", will also be able to detect when there is a sharp jump in "z-score", and it is this jump which marks the boundaries of a user's search session. b. They mention using a variant of HAC, so their overall process is likely different, although they also omit many details which were explicated in class. No mention was made of testing for clusterability. Distance calculation is rather straightforward (there is only one feature, so Euclidean/Manhattan amount to the same thing). Linkage seems to be average-based. The number of clusters is set to k=1, which isn't so important because their use of HAC seems to focus more on taking advantage of "large jumps during the search of the feature space" than anything else. So despite whatever differences/omissions there may be, I don't think it changes my opinions of the paper. c. Their theoretical framework mentions three relatively-independent variables which each contribute to the demarcation of search sessions, but they only test one variable (which is the duration of pause between search engine queues). Even though they explicitly mention testing each variable independently and use the theoretical framework as a justification for that, I still think it would very much be worthwhile to look for clustering in a higher feature space, i.e. by combining and measuring for the three variables all at once.