Mass Consuming Miniature Meanings

Analysing the Carved Stones of Neolithic Bornholm

Laura Ahlqvist [*] Christian S. Hoggard [†] Rune Iversen [‡] Ditte Kofod [§]
Otto Nielsen [¶] Finn Ole S. Nielsen [‖] Niels N.Johannsen [**]

December 2020

Abstract

A unique artefact assemblage recovered at the causewayed enclosure of Vasagård on the Baltic island of Bornholm may provide insights into ritual life and underlying patterns of cognition and transmission among the Neolithic population that used this site. Here, more than 400 so-called 'sun stones' have emerged – small tablets of shale, sandstone and water-rolled pebbles that have been engraved with a range of motifs. One prevalent motif consists of a circle with radiating lines very similar to what present-day humans would produce if asked to draw a sun, hence the archaeological name; however, a range of other motifs also appear on the stones. All of the engravings are based on a relatively small repertoire and yet, no two stones are identical as the repeated elements are combined in different constellations, creating substantial variability within the material. An explanation for the seeming dichotomy between the normativity that directs the choice of motif, on the one hand, and the relative freedom in its execution, on the other, may potentially be found in the ways that cultural transmission and cognitive processes structured the manufacturing and use of the stones. Drawing on a range of interdisciplinary approaches, we explore the roles of imitation, emulation and active teaching, and how these connect to reproducibility and memorability, in an attempt to understand the apparent combination of structuring and idiosyncrasy. This leads to a number of observations as well as suggestions for further research on these enigmatic stones.

Introduction

This is a R Markdown documented associated with the article *Mass Consuming Miniature Meanings: Analysing the Carved Stones of Neolithic Bornholm*, original research submitted to **PLoS ONE**. This Markdown document details the analytical procedure used throughout the article, from data importing and the packages used, to the multivariate framework underpinning the article's findings.

A copy of all files (including visualisations and tree data) can also be found on **GitHub** and the **Open Science Framework**.

Package requirements

11 packages are required:
- **ape v.5.4-1** (https://cran.r-project.org/web/packages/ape/index.html)

[*]Institution, example@example.org
[†]Department of Archaeology and Anthropology, University of Southampton, C.S.Hoggard@soton.ac.uk
[‡]Institution3, example@example.org
[§]Institution4, example@example.org
[¶]Institution5, example@example.org
[‖]Institution6, example@example.org
[**]Institution7, example@example.org

- **FactoMineR v.2.3** (https://cran.r-project.org/web/packages/FactoMineR/index.html)
- **factoextra v.1.0.7** (https://cran.r-project.org/web/packages/factoextra/index.html)
- **ggraph v.2.0.4** (https://cran.r-project.org/web/packages/ggraph/index.html)
- **ggtree b.2.2.0** (https://bioconductor.org/packages/release/bioc/html/ggtree.html)
- **igraph v.1.2.6** (https://cran.r-project.org/web/packages/igraph/index.html)
- **magrittr v.1.5** (https://cran.r-project.org/web/packages/magrittr/index.html)
- **patchwork v.1.1.0** (https://cran.r-project.org/web/packages/patchwork/index.html) - **phangorn v.2.5.5** (https://cran.r-project.org/web/packages/phangorn/index.html)
- **rio v.0.5.16** (https://cran.r-project.org/web/packages/rio/index.html)
- **Rphylip v.0.1-23** (https://cran.r-project.org/web/packages/Rphylip/index.html) - **tidyverse v.1.3.0** (https://cran.r-project.org/web/packages/tidyverse/index.html)

Note: to install ggtree from Biconductor you will first need to install **BiocManager**. In this script **BiocManager v. 1.30.10* is used.

Files and Program Requirements

For this markdown three different .csv files are required:

- **Ahlqvist_et_al_2020_1.csv** [traits only (structure: dichotomous)]
- **Ahlqvist_et_al_2020_2.csv** [traits and shape (structure: dichotomous)]
- **Ahlqvist_et_al_2020_3.csv** [traits, shape, size and raw material (structure: dichotomous)]

These can be stored in a local repository and brought into the R Environment, or downloaded as individual .rds objects. To import and tidy, prior to analysis perform the following code:

```
dataset_one <- read_csv("Ahlqvist_et_al_2020_1.csv")
dataset_one %<>% mutate_at(c(names(dataset_one)), funs(factor(.))) %>%
    column_to_rownames("id")

dataset_two <- read_csv("Ahlqvist_et_al_2020_2.csv")
dataset_two %<>% mutate_at(c(names(dataset_two)), funs(factor(.))) %>%
    column_to_rownames("id")

dataset_three <- read_csv("Ahlqvist_et_al_2020_3.csv")
dataset_three %<>% mutate_at(c(names(dataset_three)), funs(factor(.))) %>%
    column_to_rownames("id")
```

Or to download straight into R from GitHub (preferred):

```
dataset_one <- rio::import("https://github.com/CSHoggard/-sun-stones/raw/main/dataset_one.rds")
dataset_two <- rio::import("https://github.com/CSHoggard/-sun-stones/raw/main/dataset_two.rds")
dataset_three <- rio::import("https://github.com/CSHoggard/-sun-stones/raw/main/dataset_three.rds")
```

Each dataset will be subject to the same three analytical and exploratory procedures, specifically multiple correspondence analysis (MCA), network analysis and the tree-building exercises. In testing the underlying structure of variation in the artefacts, a combination of all examples and unique examples (artefacts with a distinct dichotomous configuration) will be pursued. The following transformation arguments are therefore required:

```
dataset_one_unique <- dataset_one %>% unique()

dataset_two_clean <- dataset_two %>% select(-shape)

dataset_two_unique <- dataset_two_clean %>% unique()

dataset_three_clean <- dataset_three %>% select(-gcontext, -context,
```

```
    -shape, -material, -size)

dataset_three_unique <- dataset_three_clean %>% unique()
```

Note: As Rphylip is a nearly comprehensive R interface for the PHYLIP phylogeny methods program package, one must first install PHYLIP: https://evolution.genetics.washington.edu/phylip.html. The phylogenetic trees are included as stand-alone files (for those not wishing to download additional programs).

Multiple Correspondence Analysis (MCA)

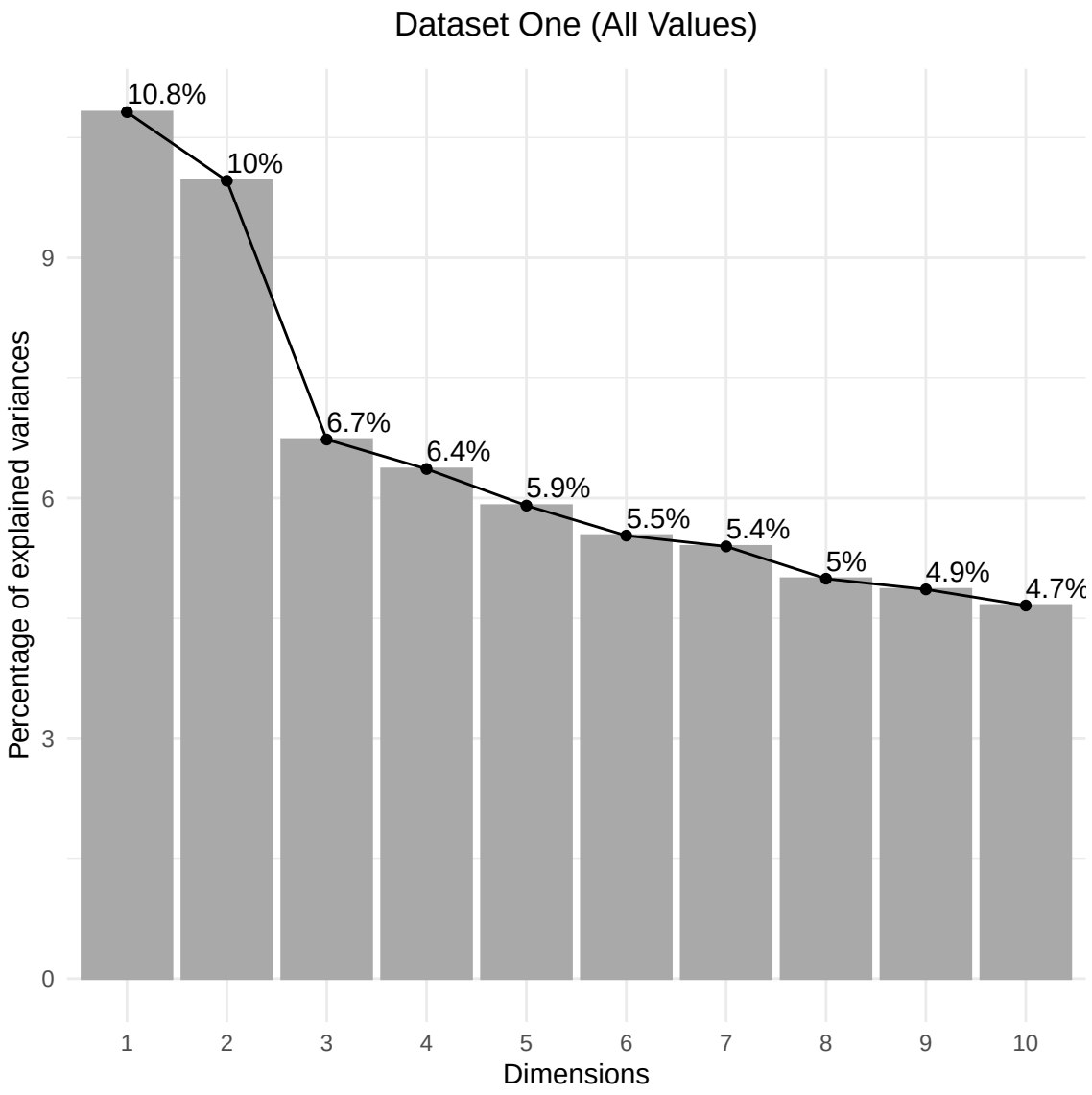
Multiple correspondence analysis (MCA henceforth) is an unsupervised multivariable procedure for analysing non-linear interactions and the interaction between several categorical dependent variables (Abdi and Valentin, 2007). Used widely among archaeologists (Ringrose, 1992; Macheridis, 2017; Asouti et al. 2018; Leplongeon, 2020), MCA is an extension of correspondence analysis (CA) and can be viewed as a generalisation of principal component analysis (PCA) for categorical and binary (dichotomous) data instead of continuous data. In being a dimensionality reduction strategy, MCA is designed to produce new synthetic dimensions comprising of different contributions from each variables. And similarly to CA, proximities (distances) are meaningful only between points from the same dataset (i.e. rows with rows and columns with columns). However, in its ability to explore the relationship between a large number of variables, MCA is of greater benefit here. As noted by Abdi and Valentin (2007), equivalent methodologies include optimal scaling, appropriate scoring, dual scaling, homogeneity analysis and scalogram analysis. For a thorough exploration of the data, a greater number of dimensions are retained rather than the default five. To create the MCA class object the `FactoMineR::MCA()` argument is used:

```
mcadataone <- MCA(dataset_one, ncp = 10, graph = FALSE)
mcadataoneu <- MCA(dataset_one_unique, ncp = 10, graph = FALSE)
mcadatatwo <- MCA(dataset_two_clean, ncp = 10, graph = FALSE)
mcadatatwou <- MCA(dataset_two_unique, ncp = 10, graph = FALSE)
mcadatathree <- MCA(dataset_three_clean, ncp = 10, graph = FALSE)
mcadatathreeu <- MCA(dataset_three_unique, ncp = 10, graph = FALSE)
```

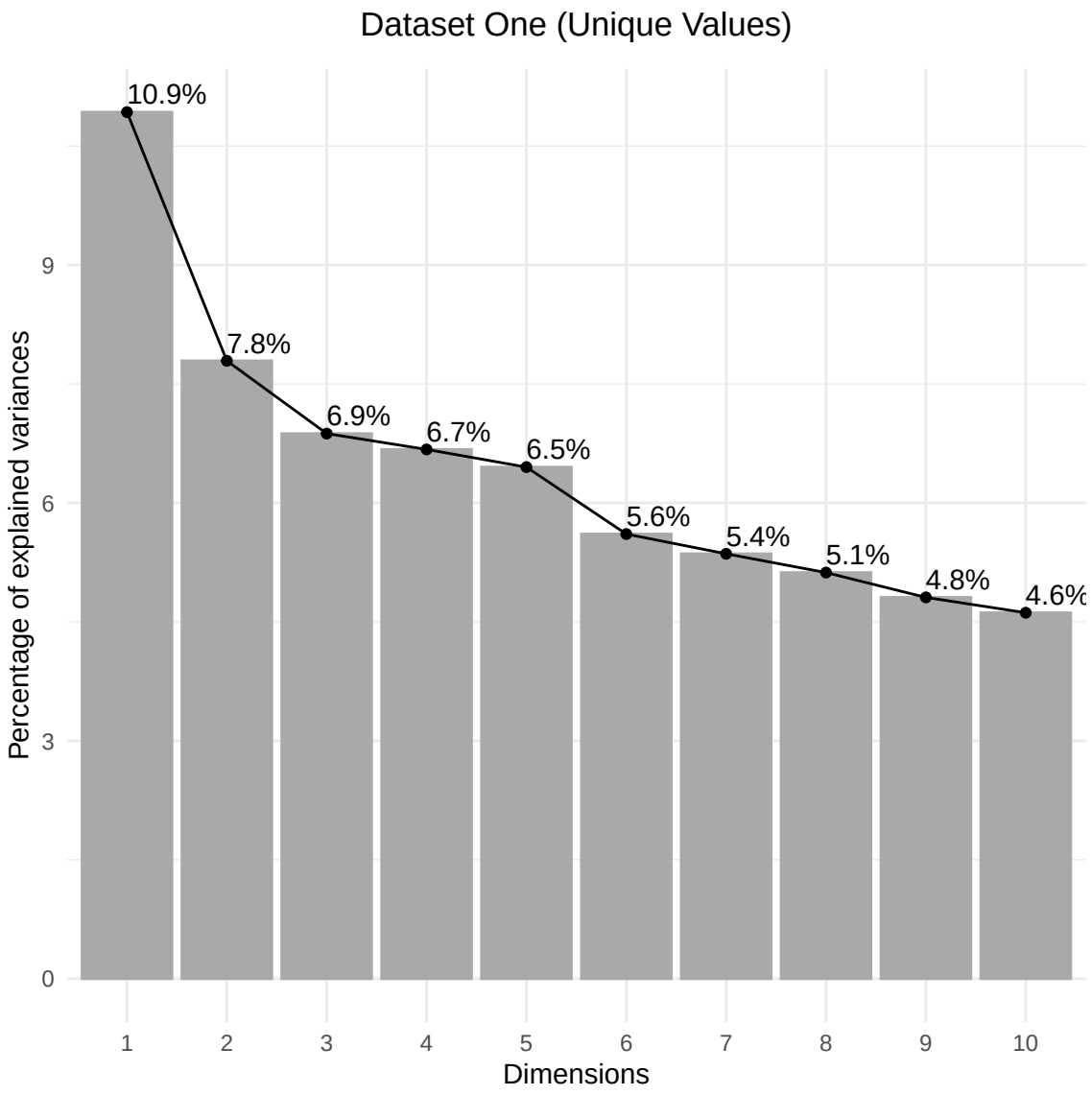
We can extract the eigenvalues (the percentage of explained variance in a given dimension) for each MCA class object through the `factoextra::get_eig()` function and visualise these eigenvalues through a screeplot, using the `factoextra::fviz_screeplot()` function:

```
get_eig(mcadataone)
get_eig(mcadataoneu)
get_eig(mcadatatwo)
get_eig(mcadatatwou)
get_eig(mcadatathree)
get_eig(mcadatathreeu)
```

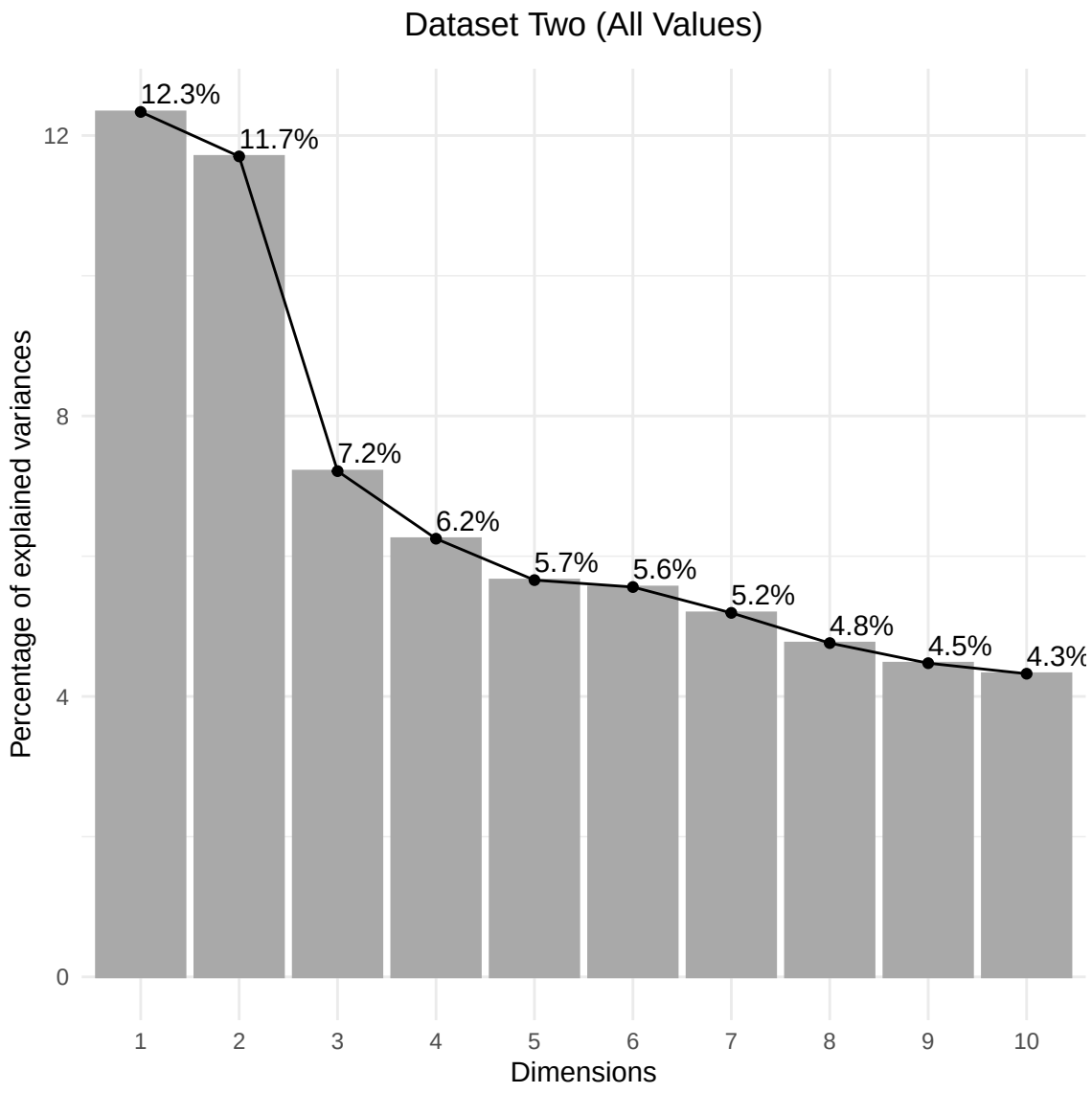
```
fviz_screeplot(mcadataone, addlabels = TRUE, barcolor = "#a9a9a9",
    barfill = "#a9a9a9") + theme_minimal() + labs(title = "Dataset One (All Values)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```



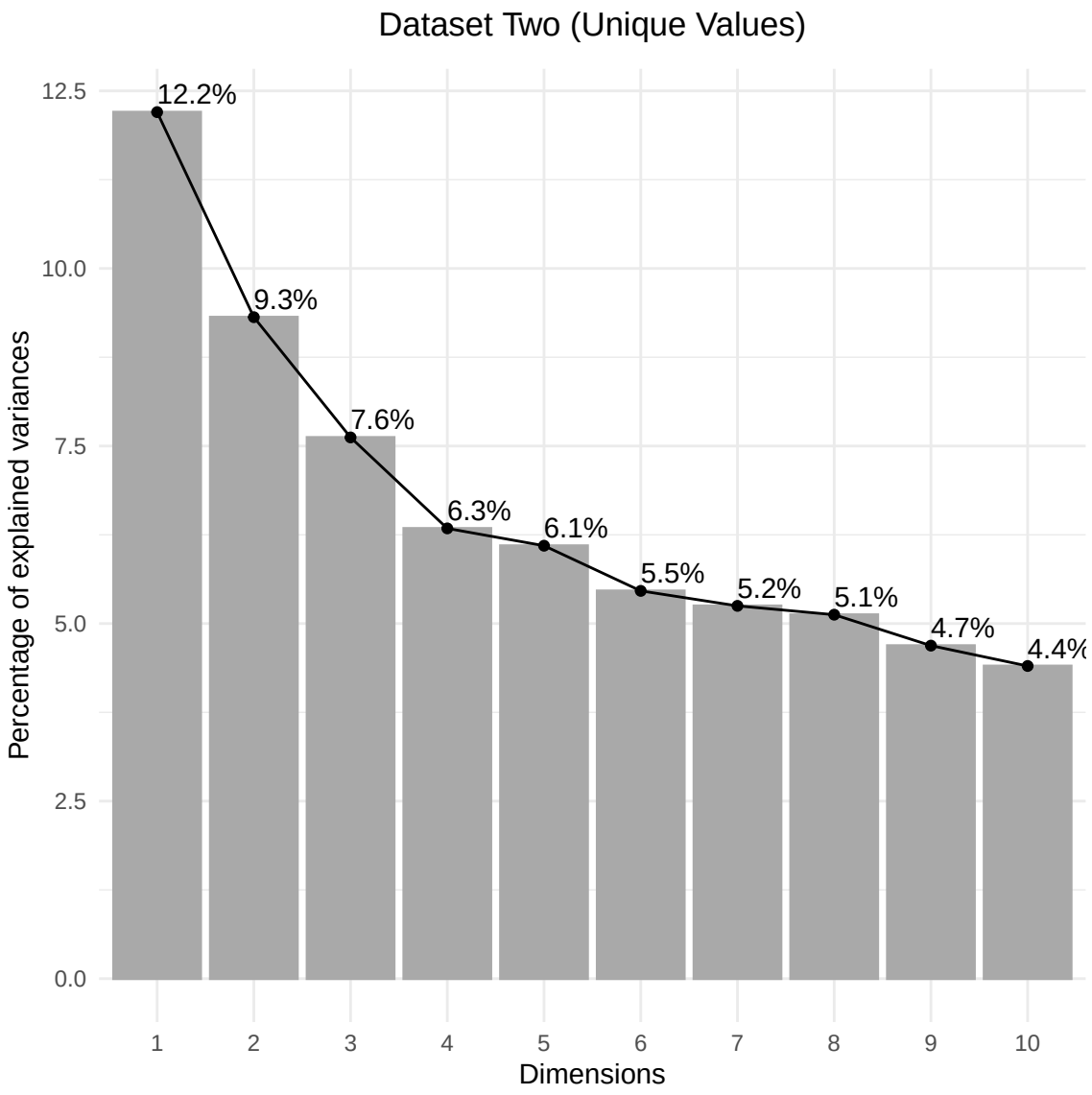

```
fviz_screeplot(mcadataoneu, addlabels = TRUE, barcolor = "#a9a9a9",
    barfill = "#a9a9a9") + theme_minimal() + labs(title = "Dataset One (Unique Values)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```



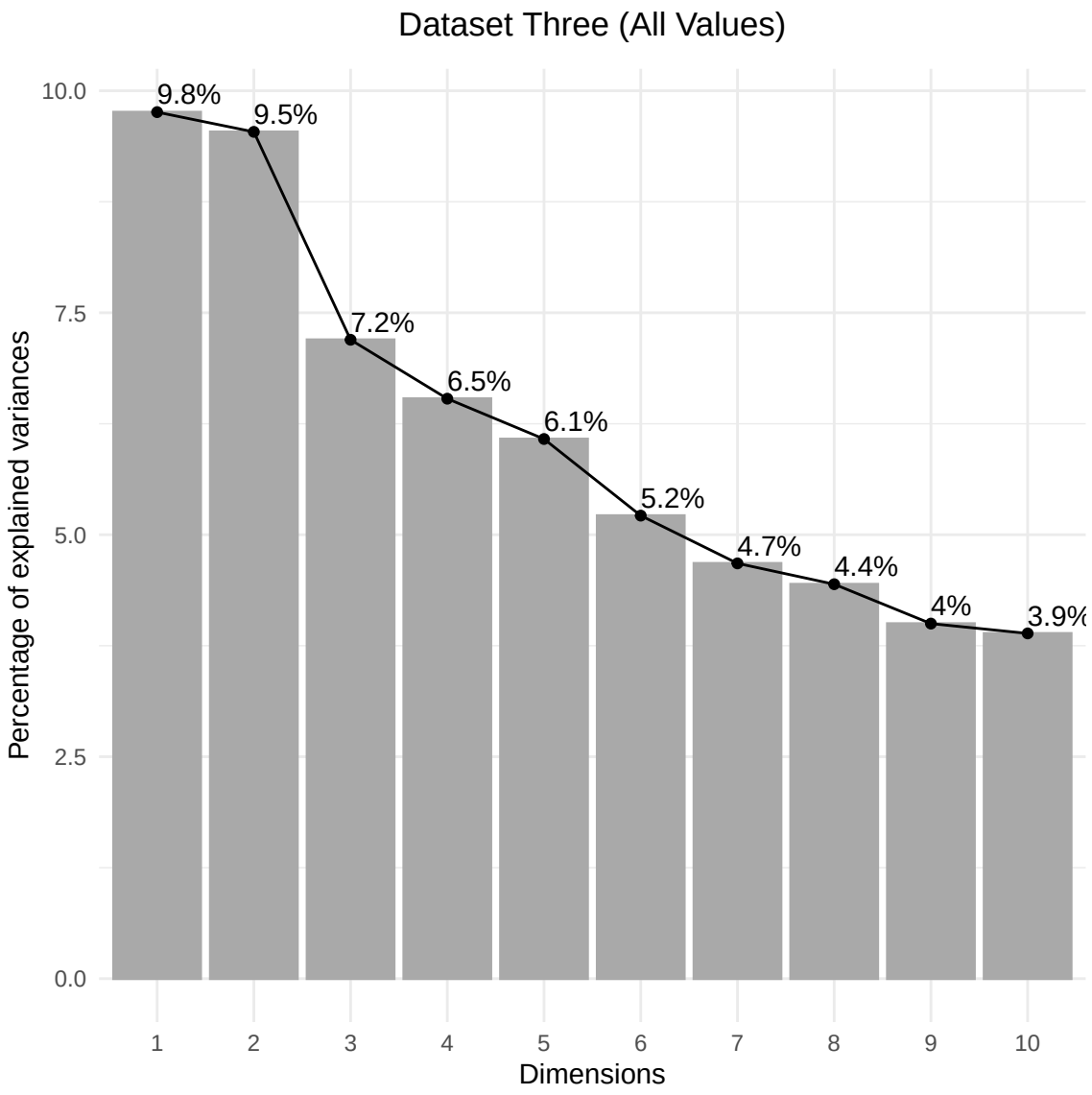

```
fviz_screeplot(mcadatatwo, addlabels = TRUE, barcolor = "#a9a9a9",
    barfill = "#a9a9a9") + theme_minimal() + labs(title = "Dataset Two (All Values)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```



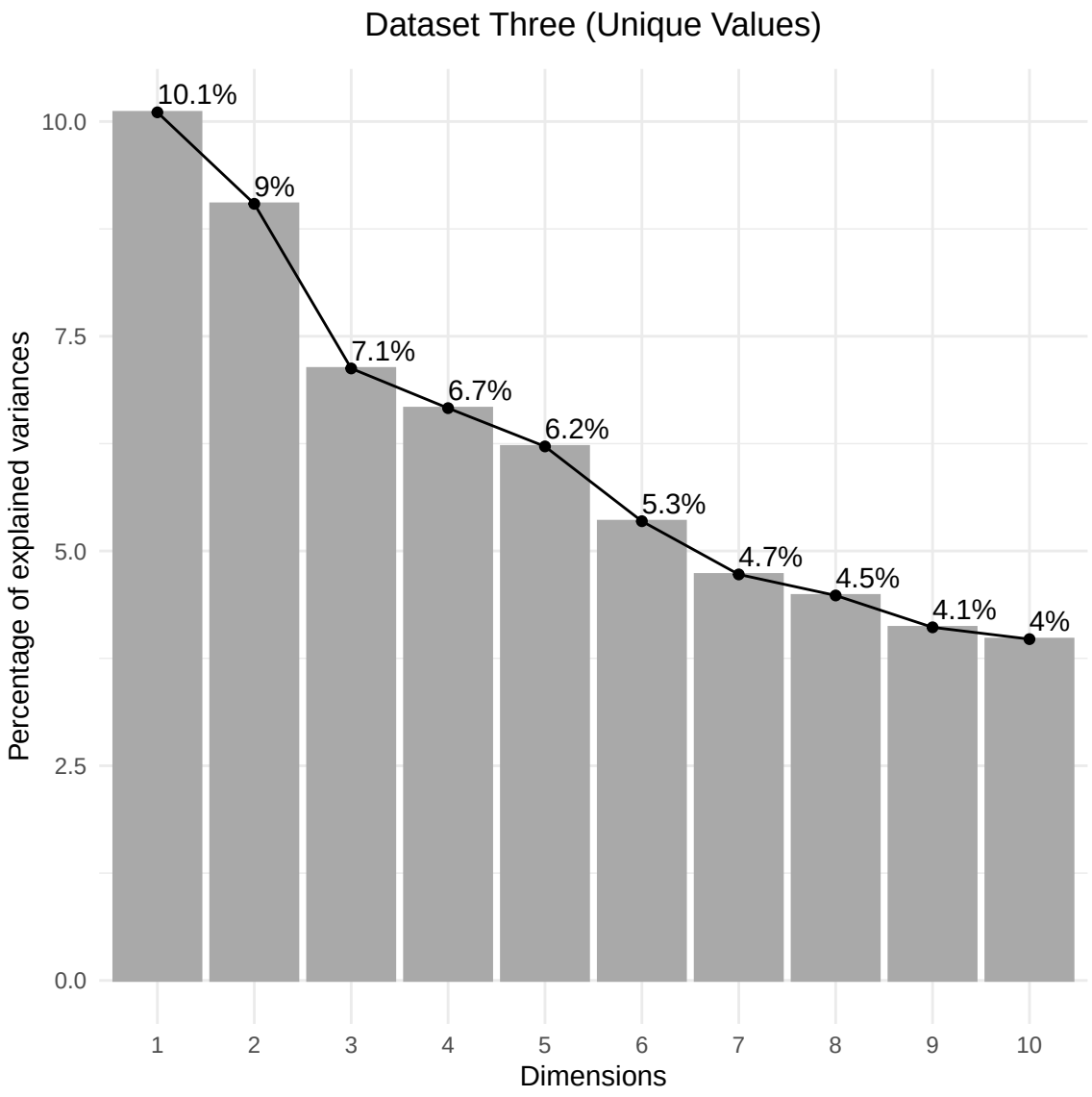

```
fviz_screeplot(mcadatatwou, addlabels = TRUE, barcolor = "#a9a9a9",
    barfill = "#a9a9a9") + theme_minimal() + labs(title = "Dataset Two (Unique Values)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```




```
fviz_screeplot(mcadatathree, addlabels = TRUE, barcolor = "#a9a9a9",
    barfill = "#a9a9a9") + theme_minimal() + labs(title = "Dataset Three (All Values)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```

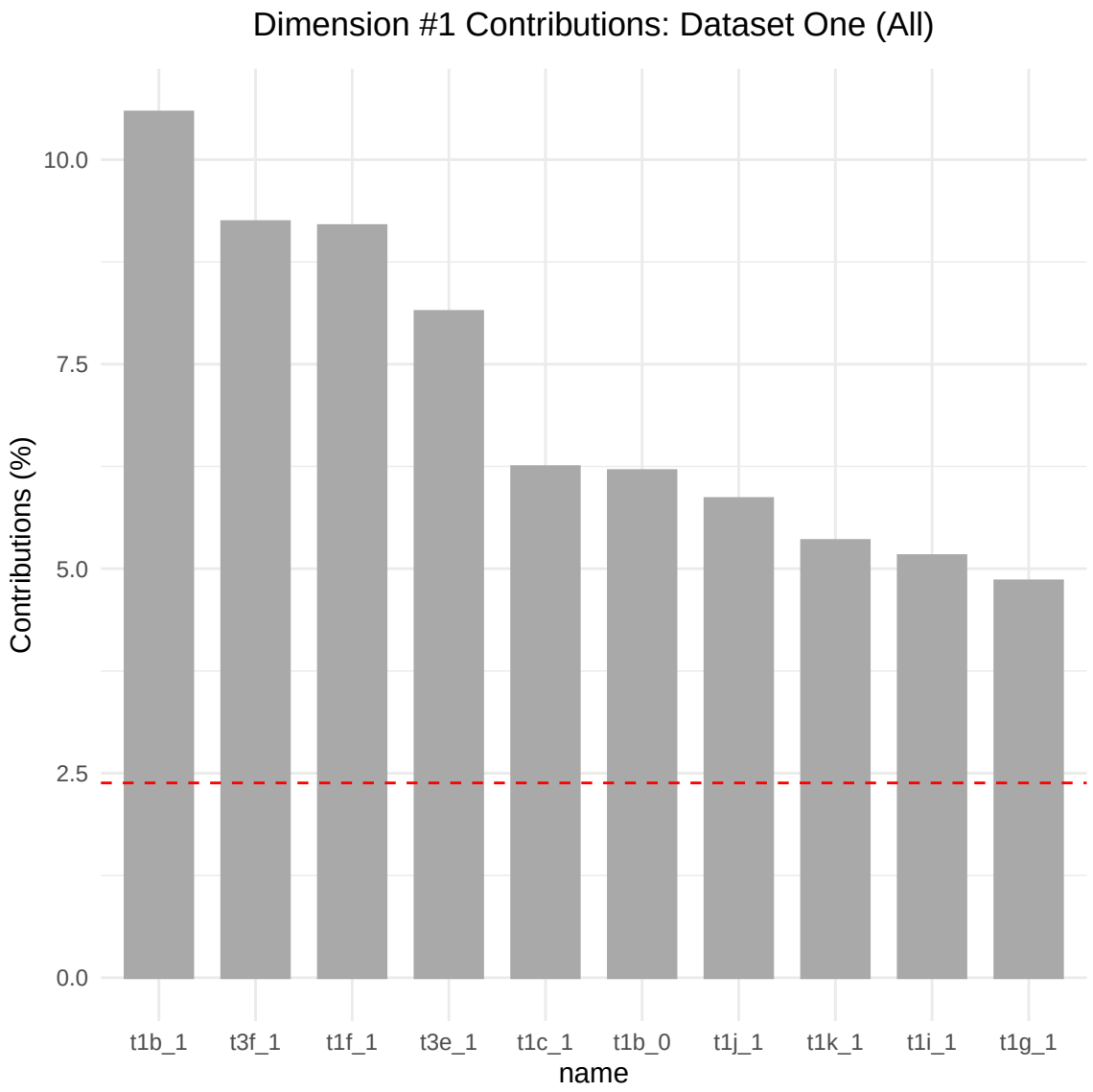



```
fviz_screeplot(mcadatathreeu, addlabels = TRUE, barcolor = "#a9a9a9",
    barfill = "#a9a9a9") + theme_minimal() + labs(title = "Dataset Three (Unique Values)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```

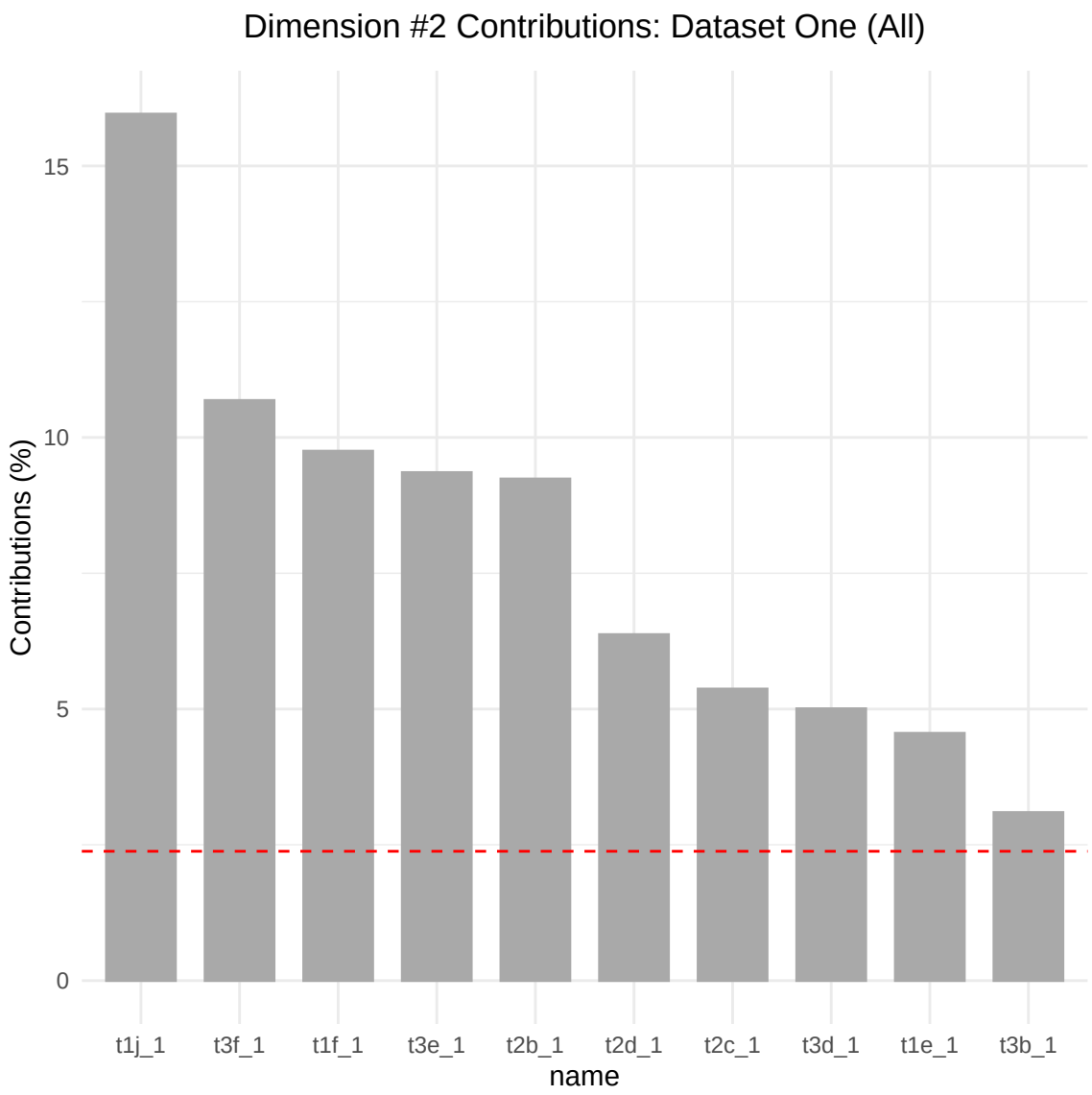


Using the `base::Summary()` function we can call the object and inspect its properties. Alternatively, we can extract the information for each variable (for each dataset) using the `factoextra::get_mca()` function. This will provide the coordinates for the categories ("coord"), the contributions of categories in each dimension ("contrib") and the squared cosine values for each category ("cos2"). The contribution of particular variables to PC1 and PC2 can now be called through the `factoextra::fviz_contrib` function, with the variables specific in the `choice` argument. For example, in the first dataset (containing all values), the presence of traits 1b, 3f, 1f, and 3e (t1b_1, t3f_1, t1f_1 and t3e_1) contribute highest to the first dimension (the largest source of hypothetical variation). For the purposes of this .rmd, this is repeated for the first and second axis of all three datasets (and the complete and unique configurations).

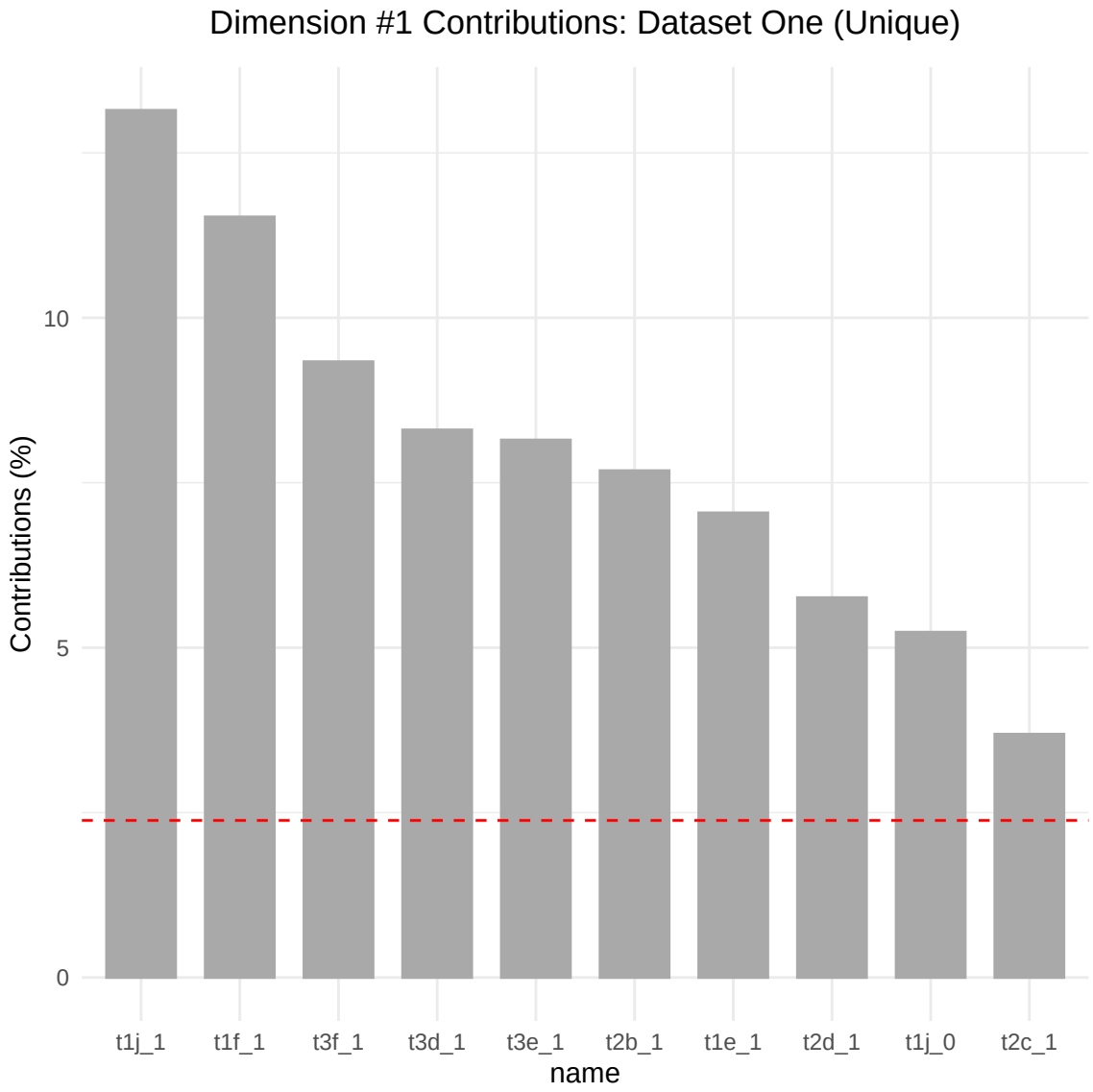
```
fviz_contrib(mcadataone, choice = "var", axes = 1, top = 10,
    color = "#a9a9a9", fill = "#a9a9a9") + theme_minimal() +
    labs(title = "Dimension #1 Contributions: Dataset One (All)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```



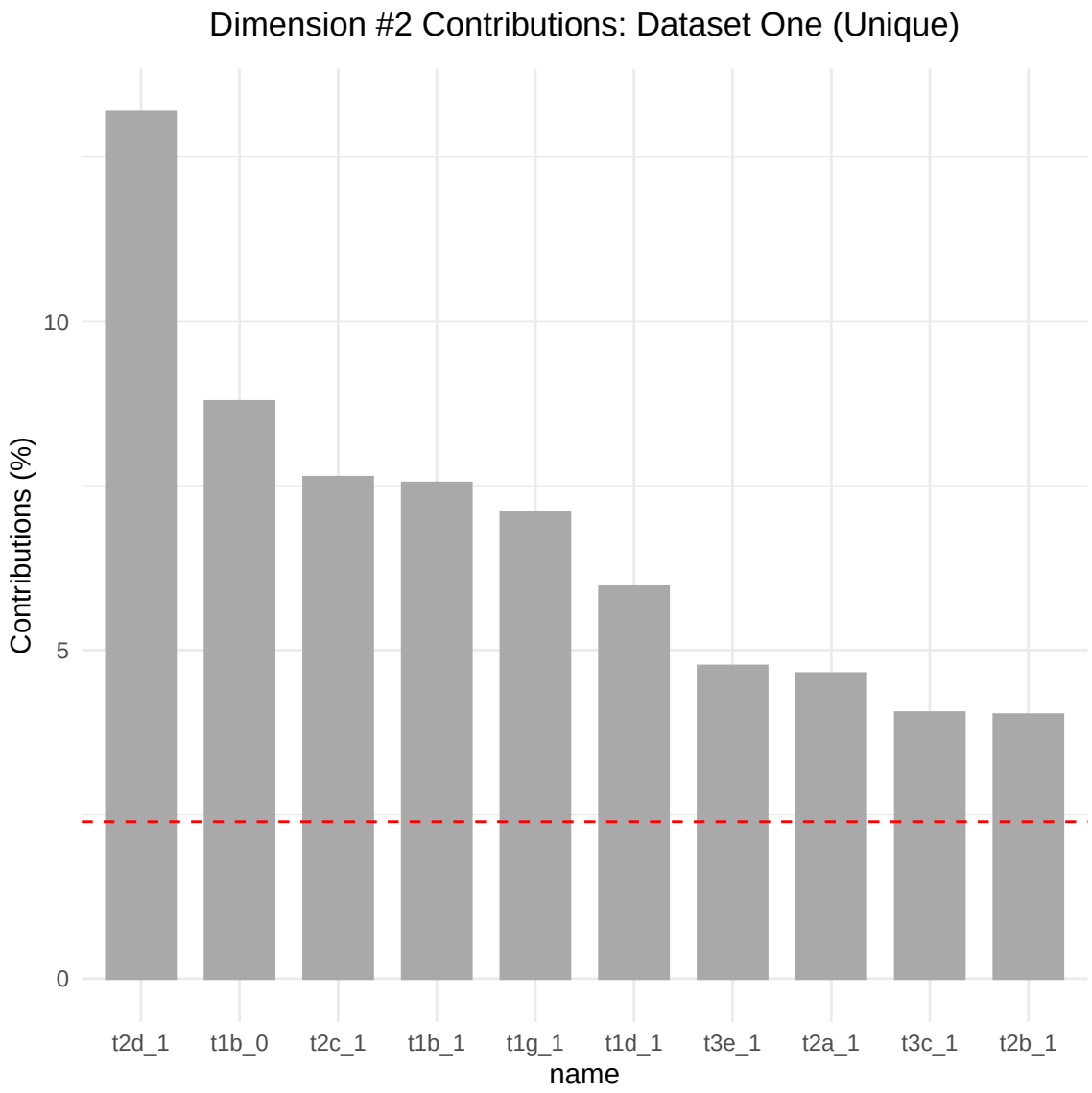

```
fviz_contrib(mcadataone, choice = "var", axes = 2, top = 10,
    color = "#a9a9a9", fill = "#a9a9a9") + theme_minimal() +
    labs(title = "Dimension #2 Contributions: Dataset One (All)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```



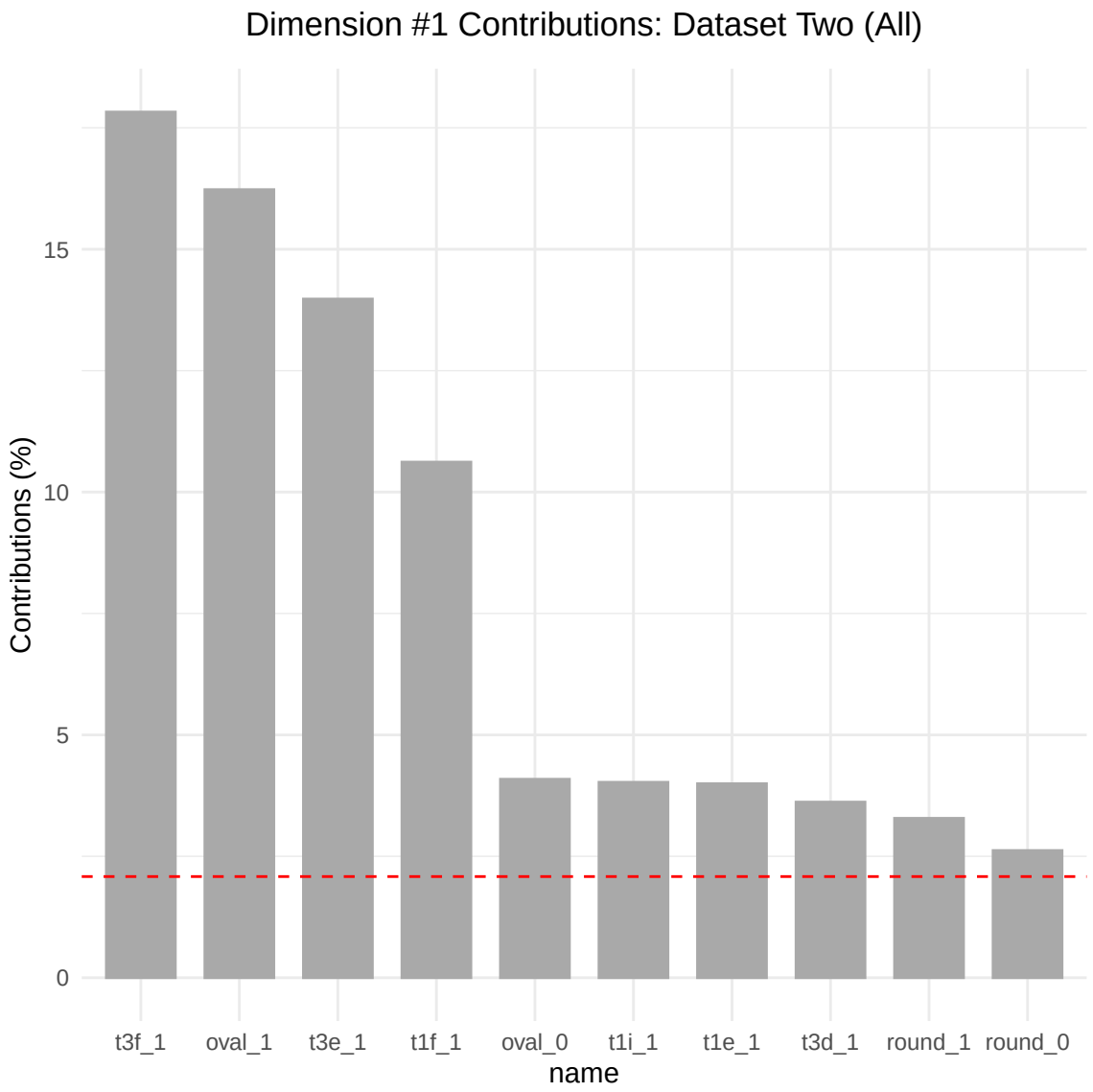

```
fviz_contrib(mcadataoneu, choice = "var", axes = 1, top = 10,
    color = "#a9a9a9", fill = "#a9a9a9") + theme_minimal() +
    labs(title = "Dimension #1 Contributions: Dataset One (Unique)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```



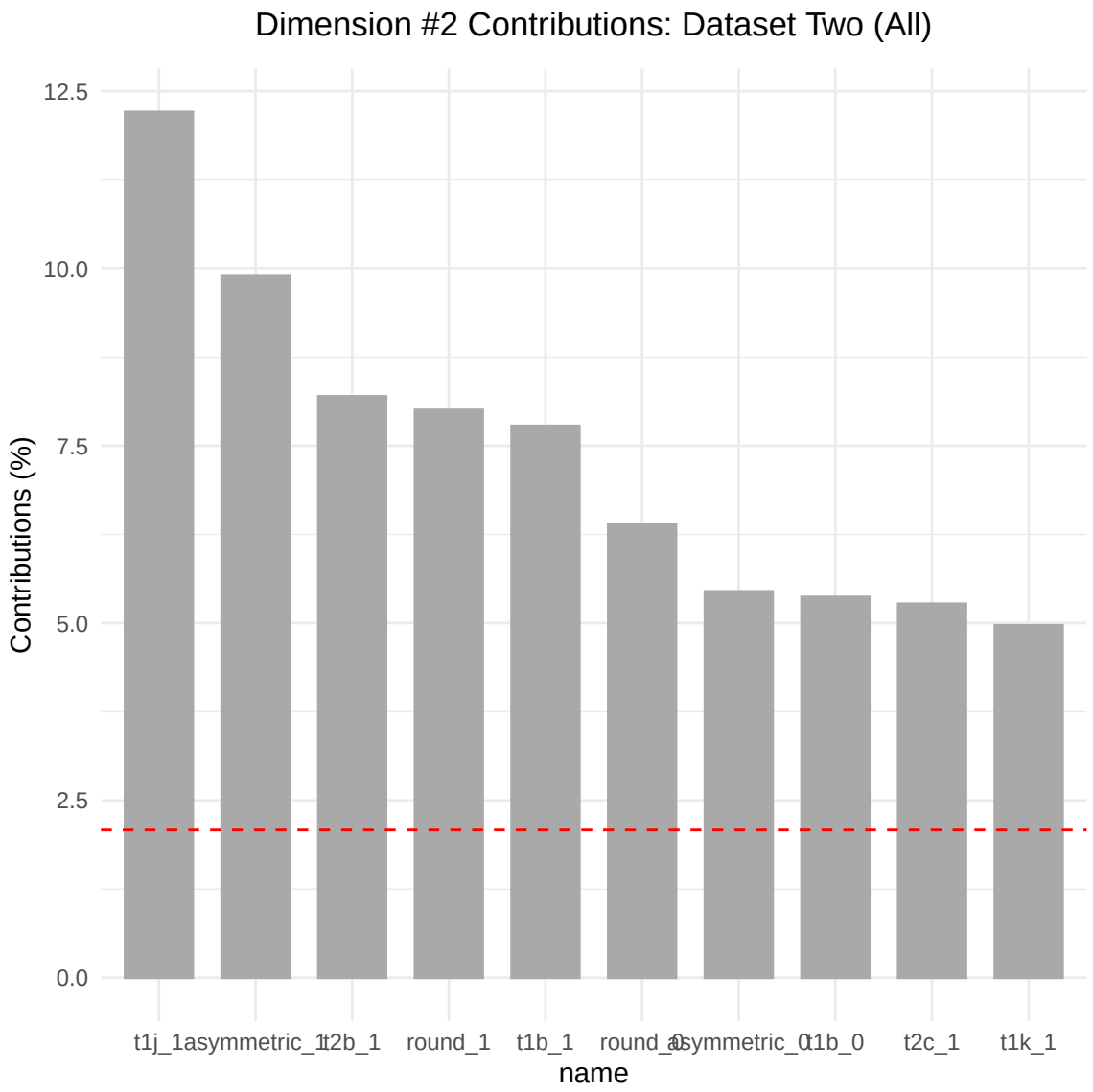

```
fviz_contrib(mcadataoneu, choice = "var", axes = 2, top = 10,
    color = "#a9a9a9", fill = "#a9a9a9") + theme_minimal() +
    labs(title = "Dimension #2 Contributions: Dataset One (Unique)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```



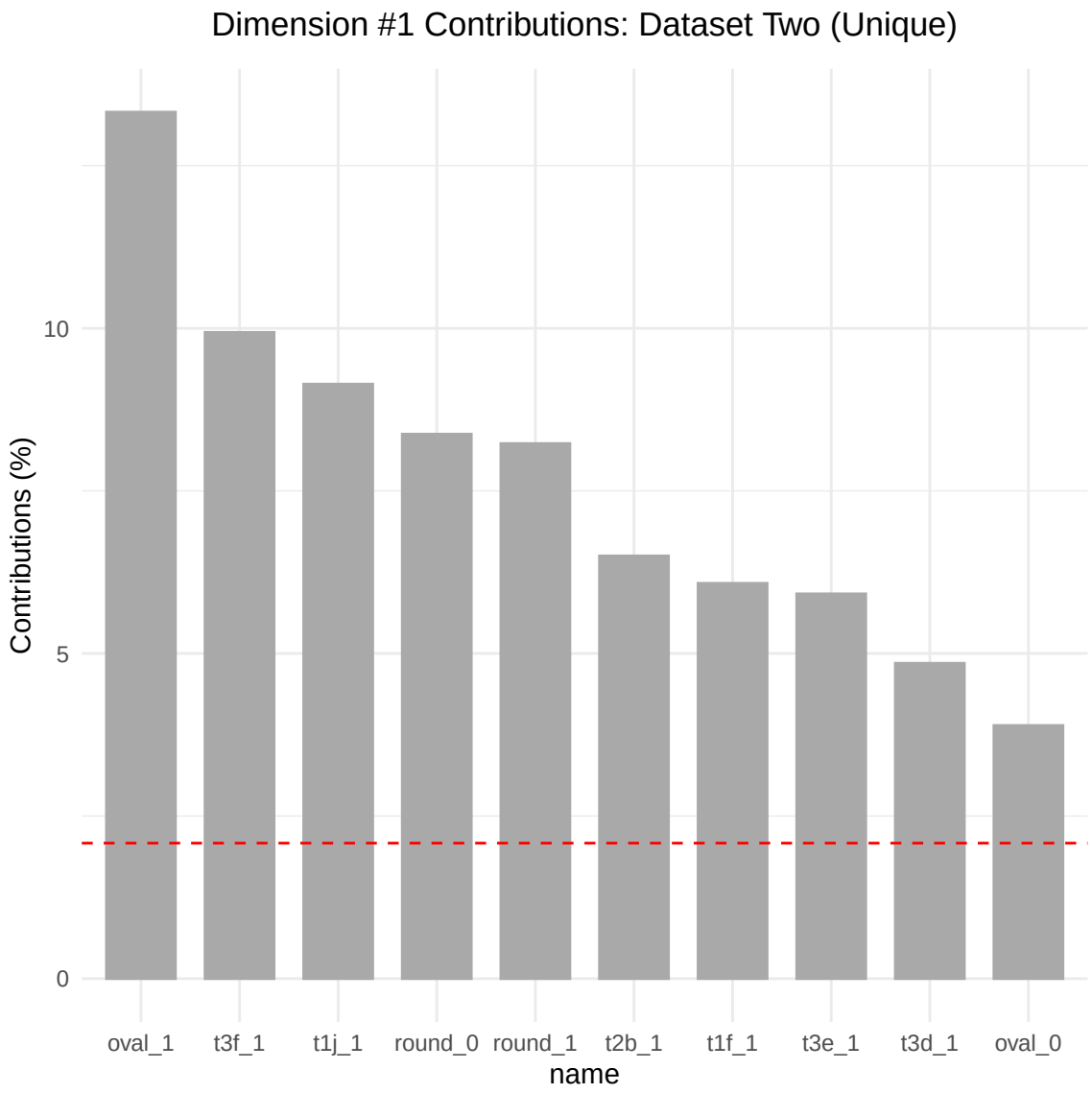

```
fviz_contrib(mcadatatwo, choice = "var", axes = 1, top = 10,
    color = "#a9a9a9", fill = "#a9a9a9") + theme_minimal() +
    labs(title = "Dimension #1 Contributions: Dataset Two (All)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```



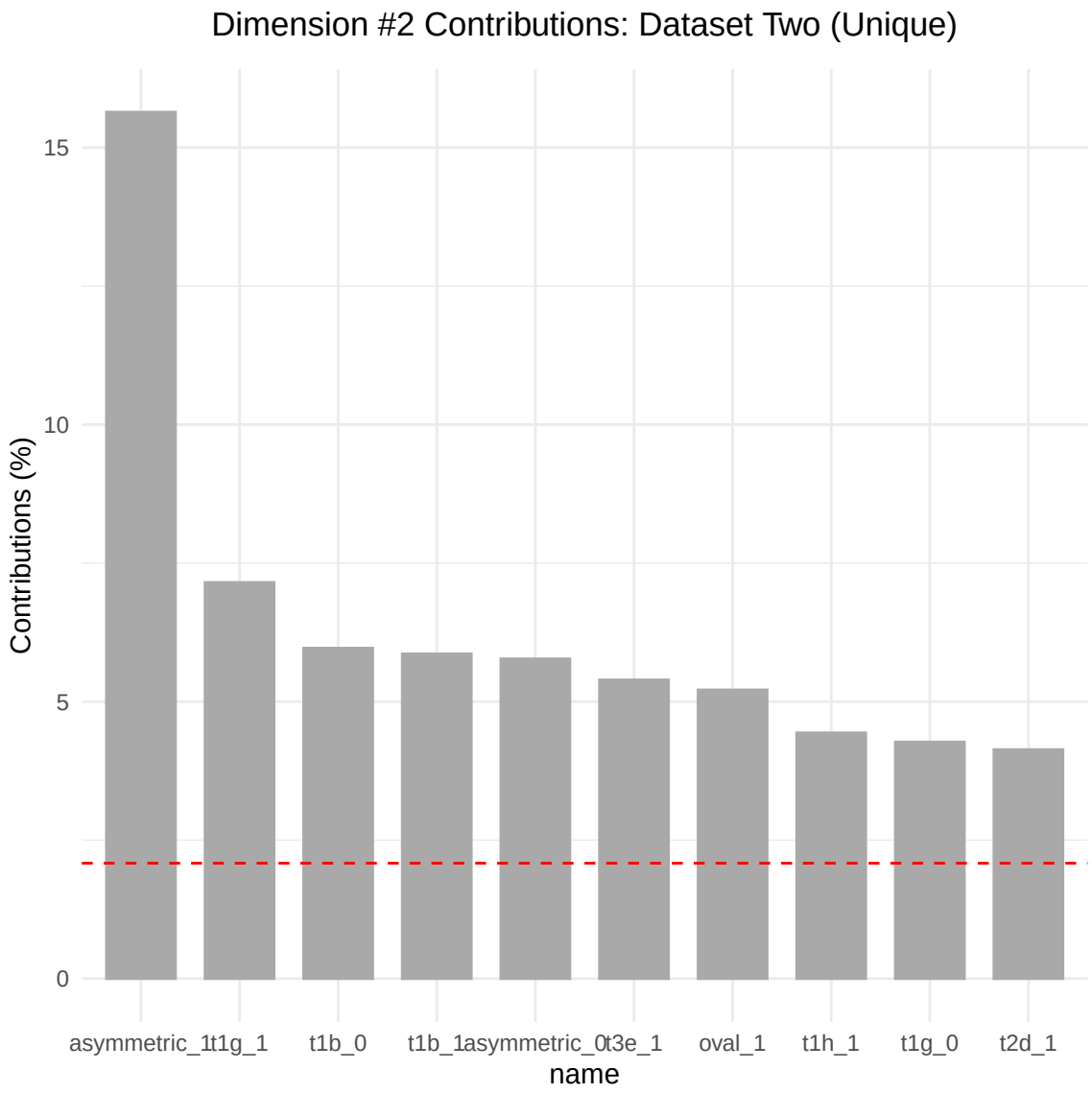

```
fviz_contrib(mcadatatwo, choice = "var", axes = 2, top = 10,
    color = "#a9a9a9", fill = "#a9a9a9") + theme_minimal() +
    labs(title = "Dimension #2 Contributions: Dataset Two (All)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```



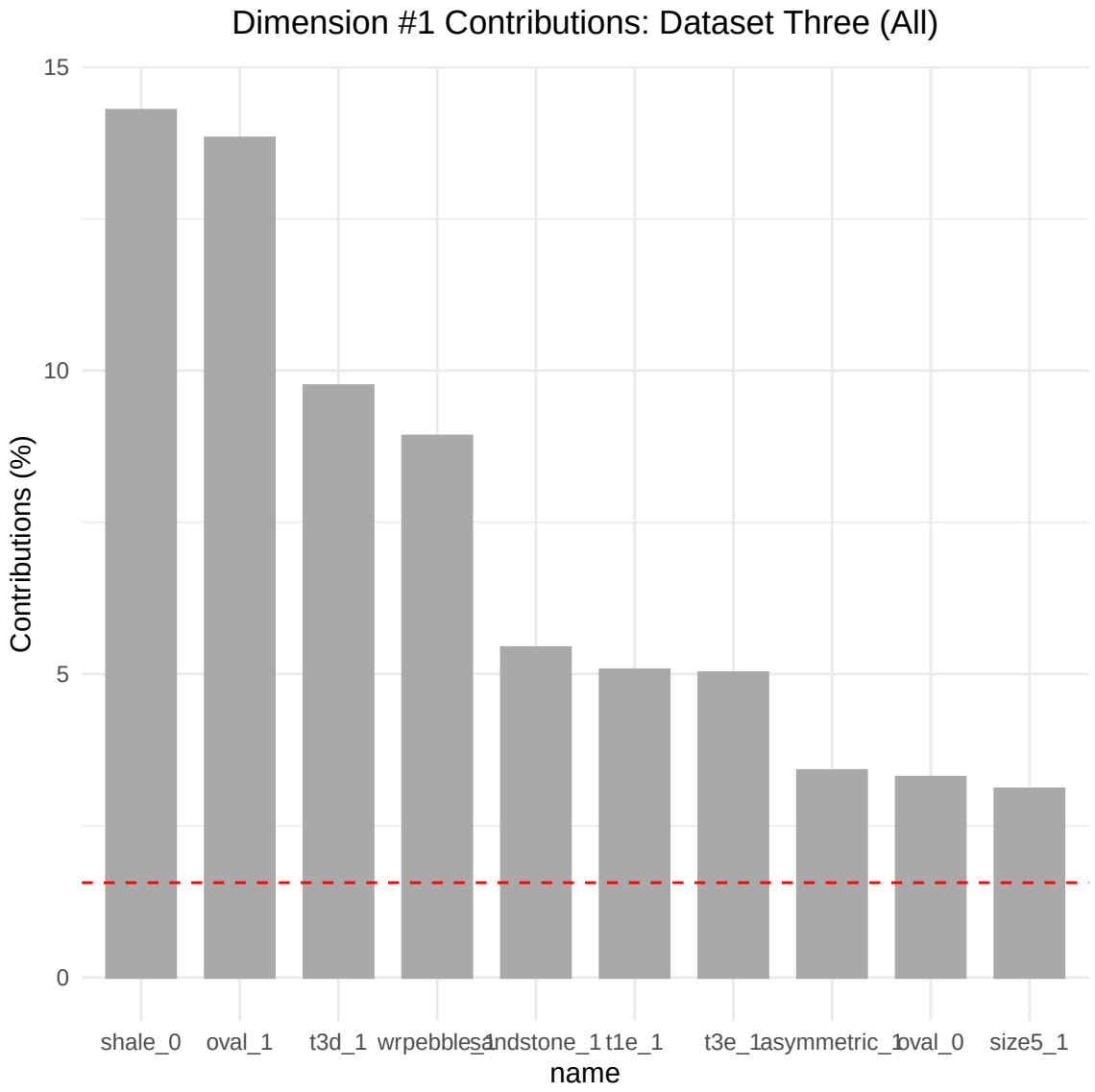

```
fviz_contrib(mcadatatwou, choice = "var", axes = 1, top = 10,
    color = "#a9a9a9", fill = "#a9a9a9") + theme_minimal() +
    labs(title = "Dimension #1 Contributions: Dataset Two (Unique)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```



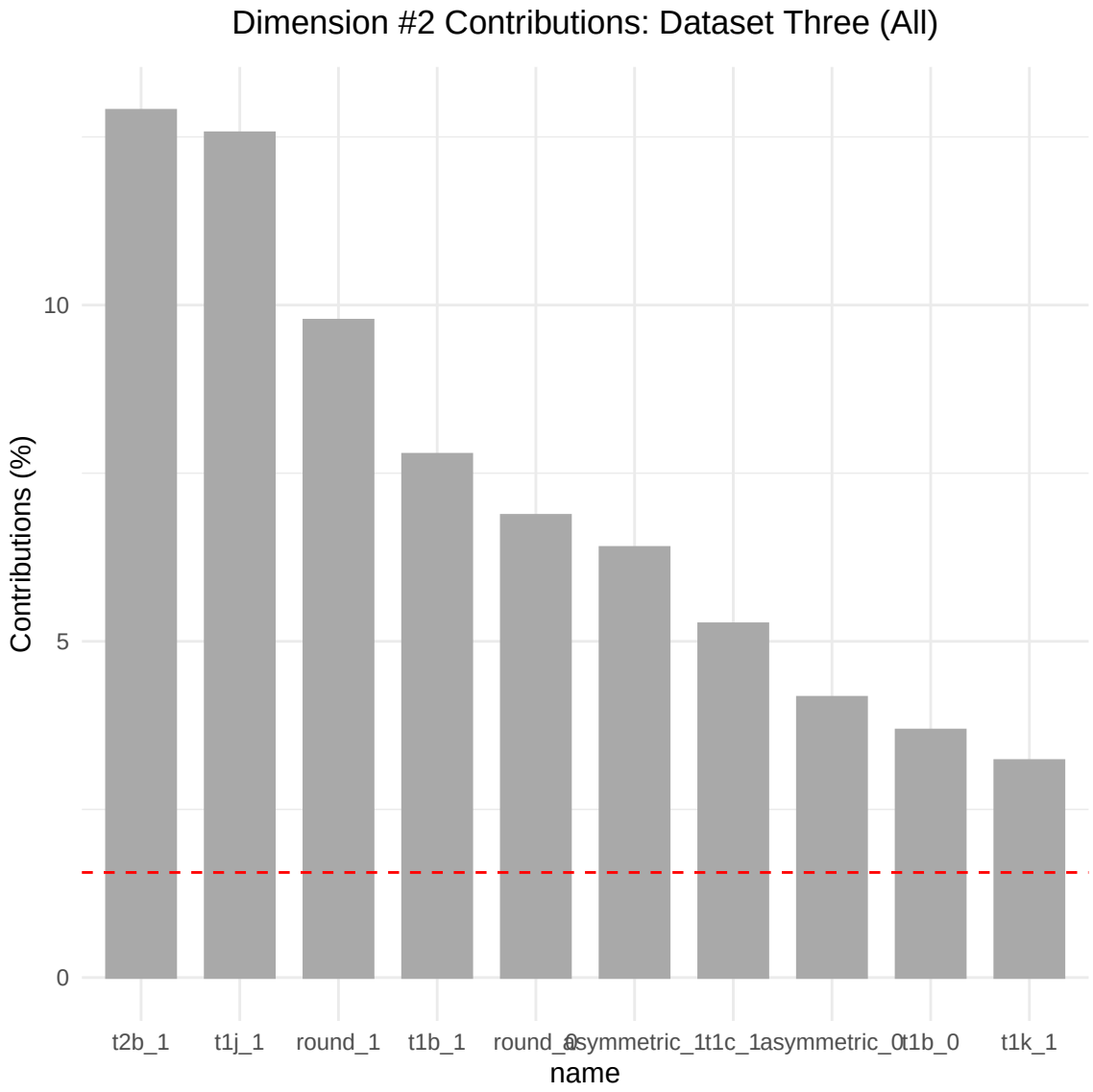

```
fviz_contrib(mcadatatwou, choice = "var", axes = 2, top = 10,
    color = "#a9a9a9", fill = "#a9a9a9") + theme_minimal() +
    labs(title = "Dimension #2 Contributions: Dataset Two (Unique)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```



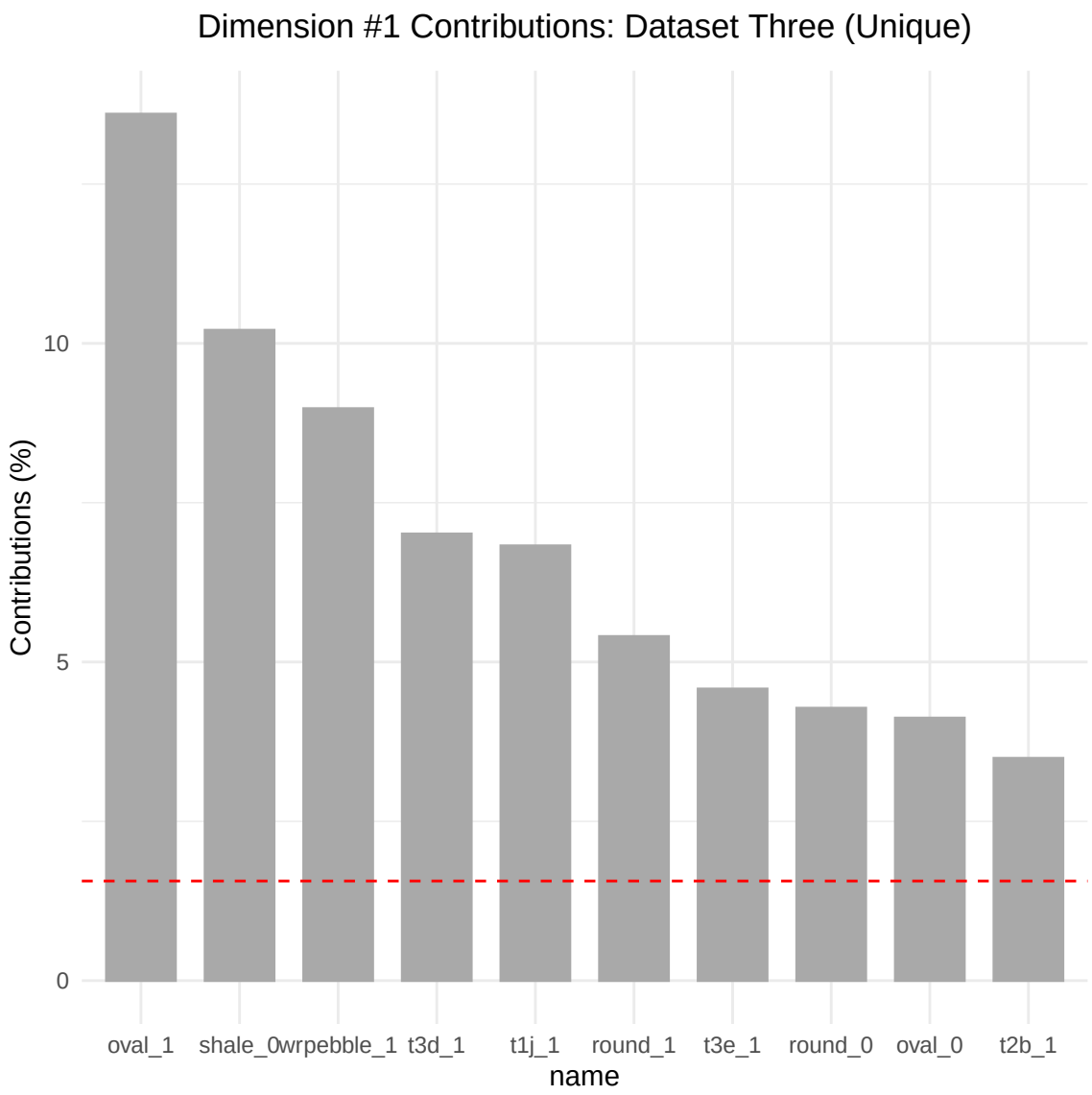

```
fviz_contrib(mcadatathree, choice = "var", axes = 1, top = 10,
    color = "#a9a9a9", fill = "#a9a9a9") + theme_minimal() +
    labs(title = "Dimension #1 Contributions: Dataset Three (All)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```



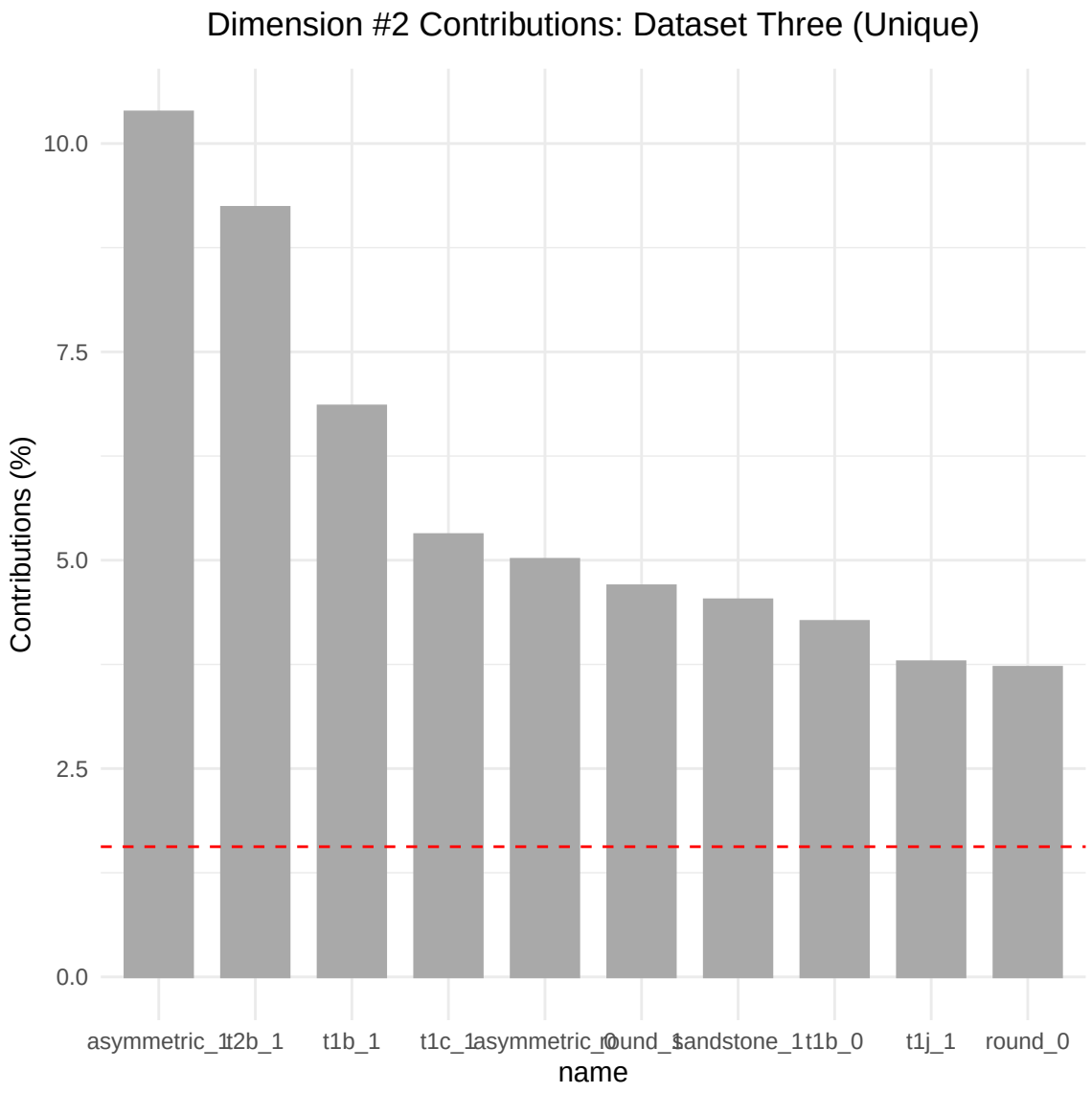

```
fviz_contrib(mcadatathree, choice = "var", axes = 2, top = 10,
    color = "#a9a9a9", fill = "#a9a9a9") + theme_minimal() +
    labs(title = "Dimension #2 Contributions: Dataset Three (All)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```




```
fviz_contrib(mcadatathreeu, choice = "var", axes = 1, top = 10,
    color = "#a9a9a9", fill = "#a9a9a9") + theme_minimal() +
    labs(title = "Dimension #1 Contributions: Dataset Three (Unique)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```

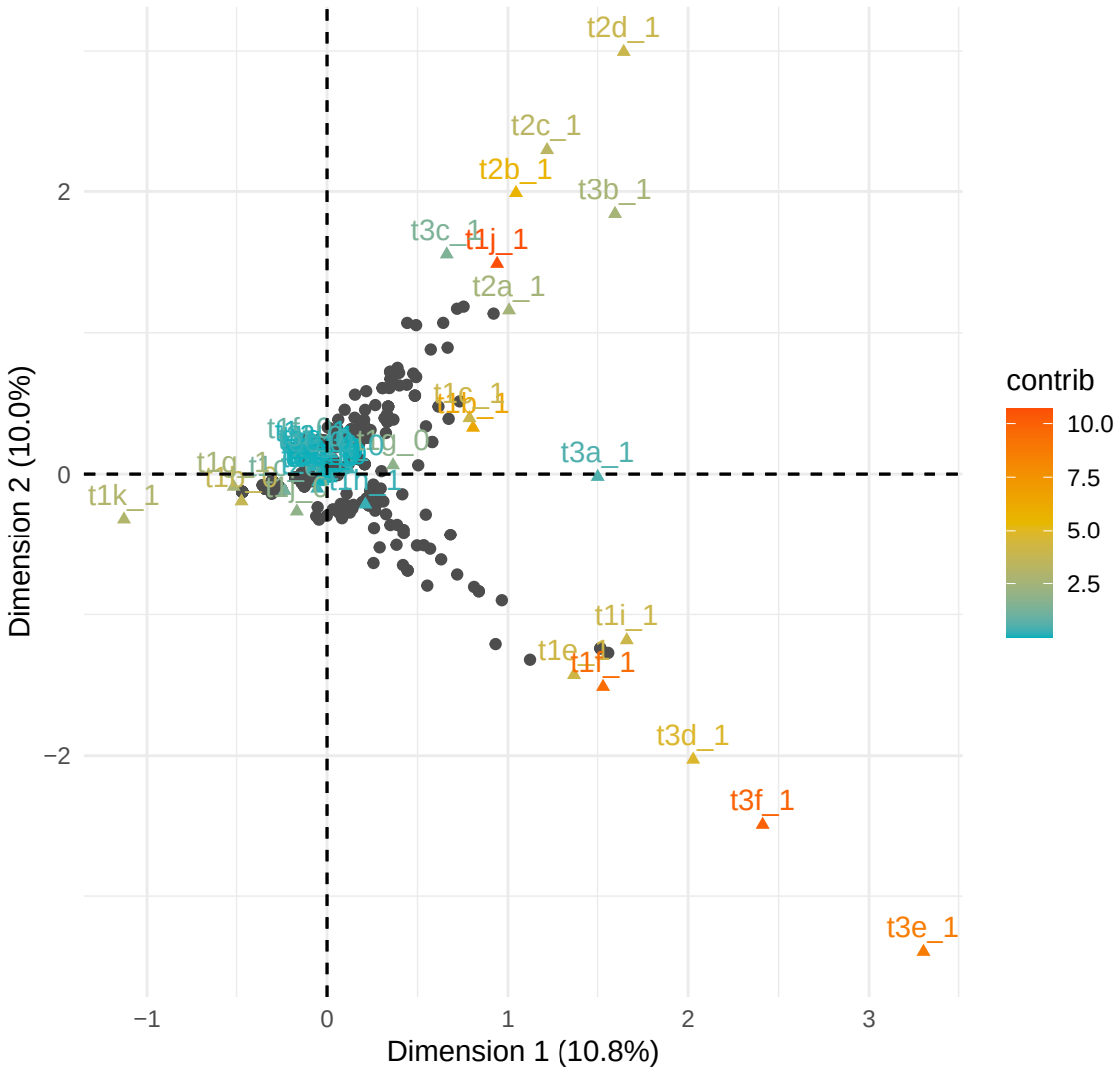



```
fviz_contrib(mcadatathreeu, choice = "var", axes = 2, top = 10,
    color = "#a9a9a9", fill = "#a9a9a9") + theme_minimal() +
    labs(title = "Dimension #2 Contributions: Dataset Three (Unique)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```

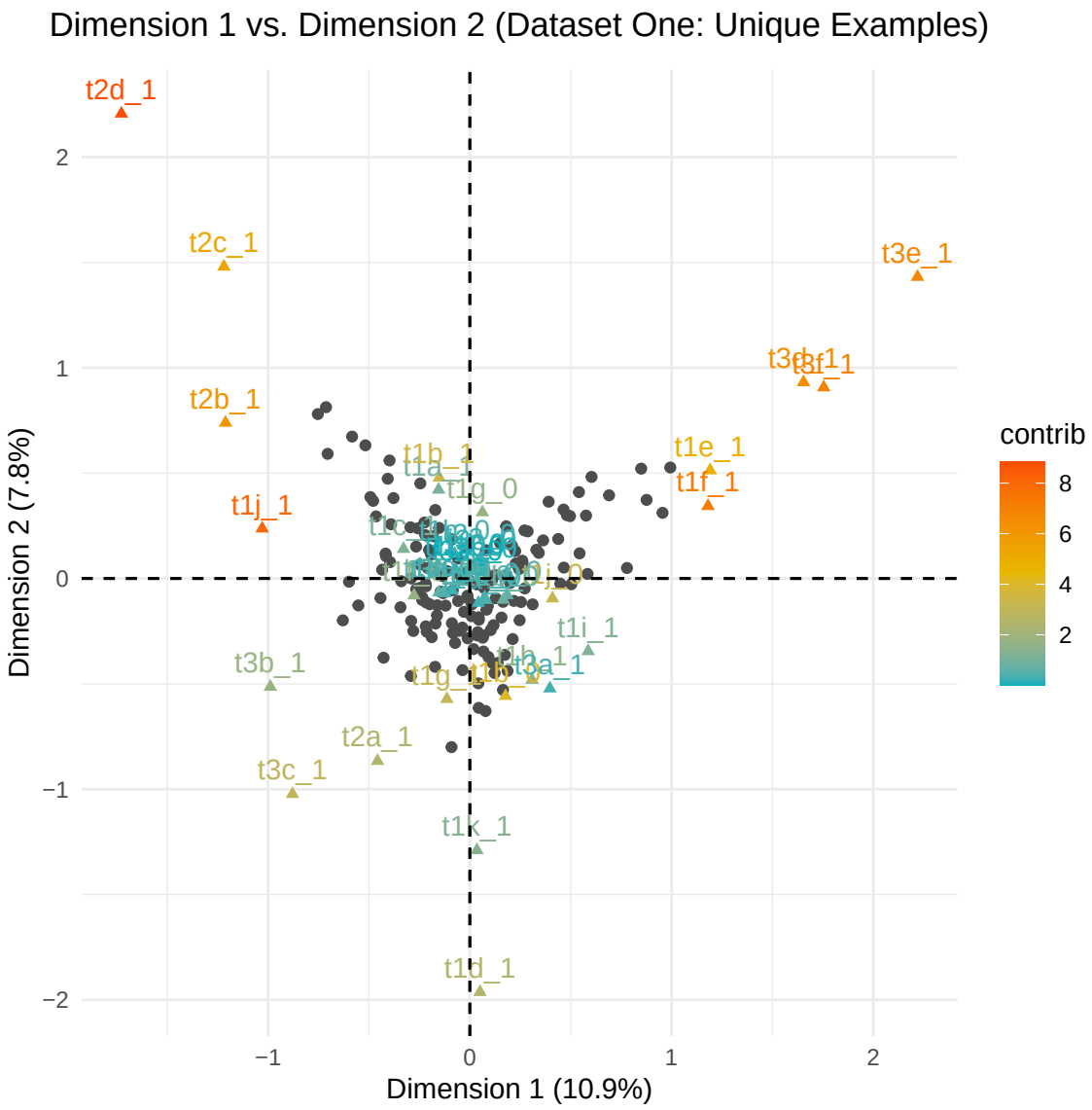


With the different contributions of each dimension understood, and the percentage variance as explained by each dimension established, we can now begin to construct a bivariate graph, displaying instances (artefacts) and contributions, and understand the main changes in artefact design. Contribution graphics, with artefacts mapped can be produced through the `factoextra::fviz_mca` function, with the variables coloured for their respective contributions:

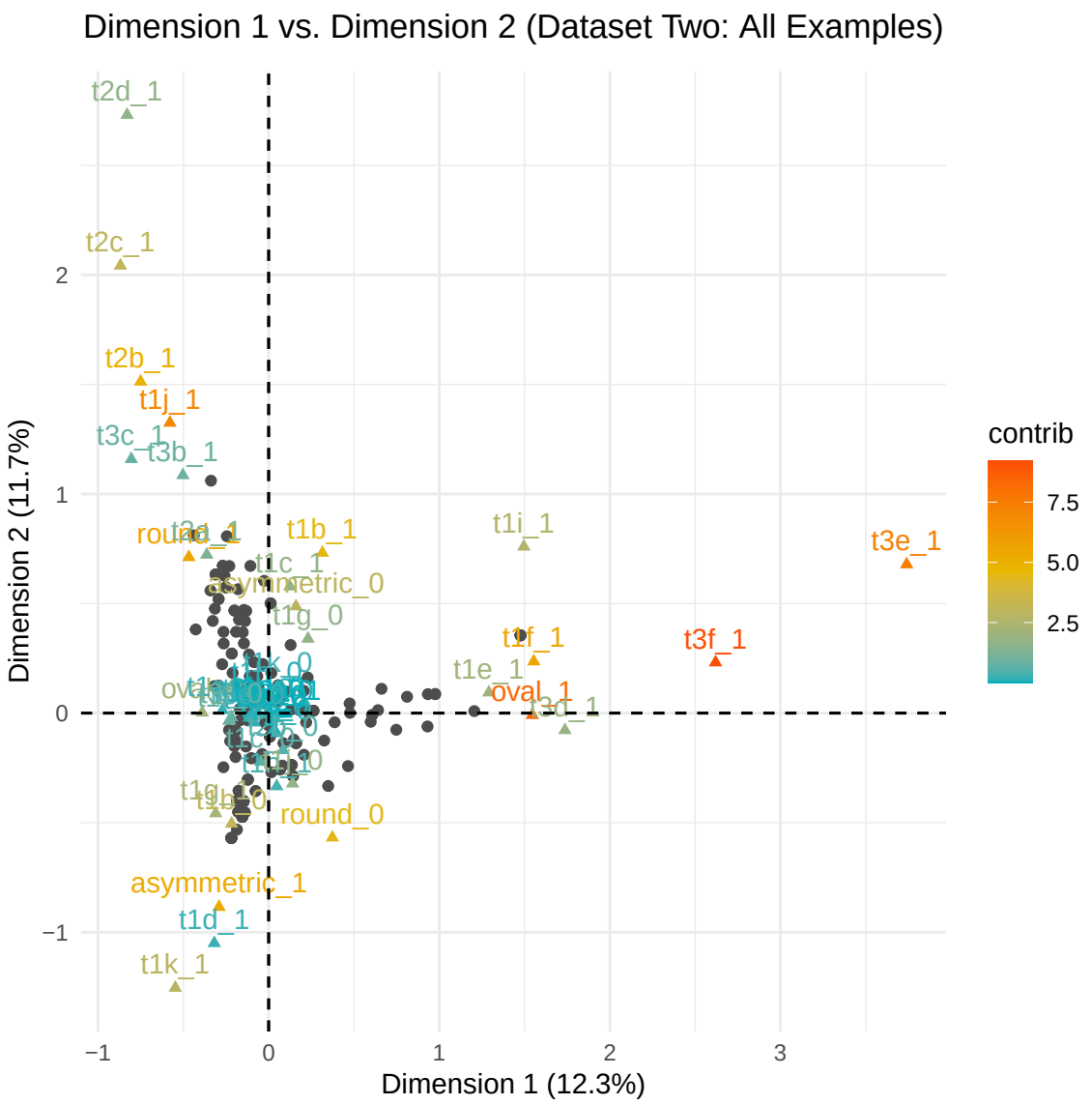
```
fviz_mca(mcadataone, col.var = "contrib", label = "var", gradient.cols = c("#00AFBB",
    "#E7B800", "#FC4E07"), col.ind = "grey30", pointsize = 1.5) +
    labs(title = "Dimension 1 vs. Dimension 2 (Dataset One: All Examples)",
        x = "Dimension 1 (10.8%)", y = "Dimension 2 (10.0%)") +
    theme_minimal() + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



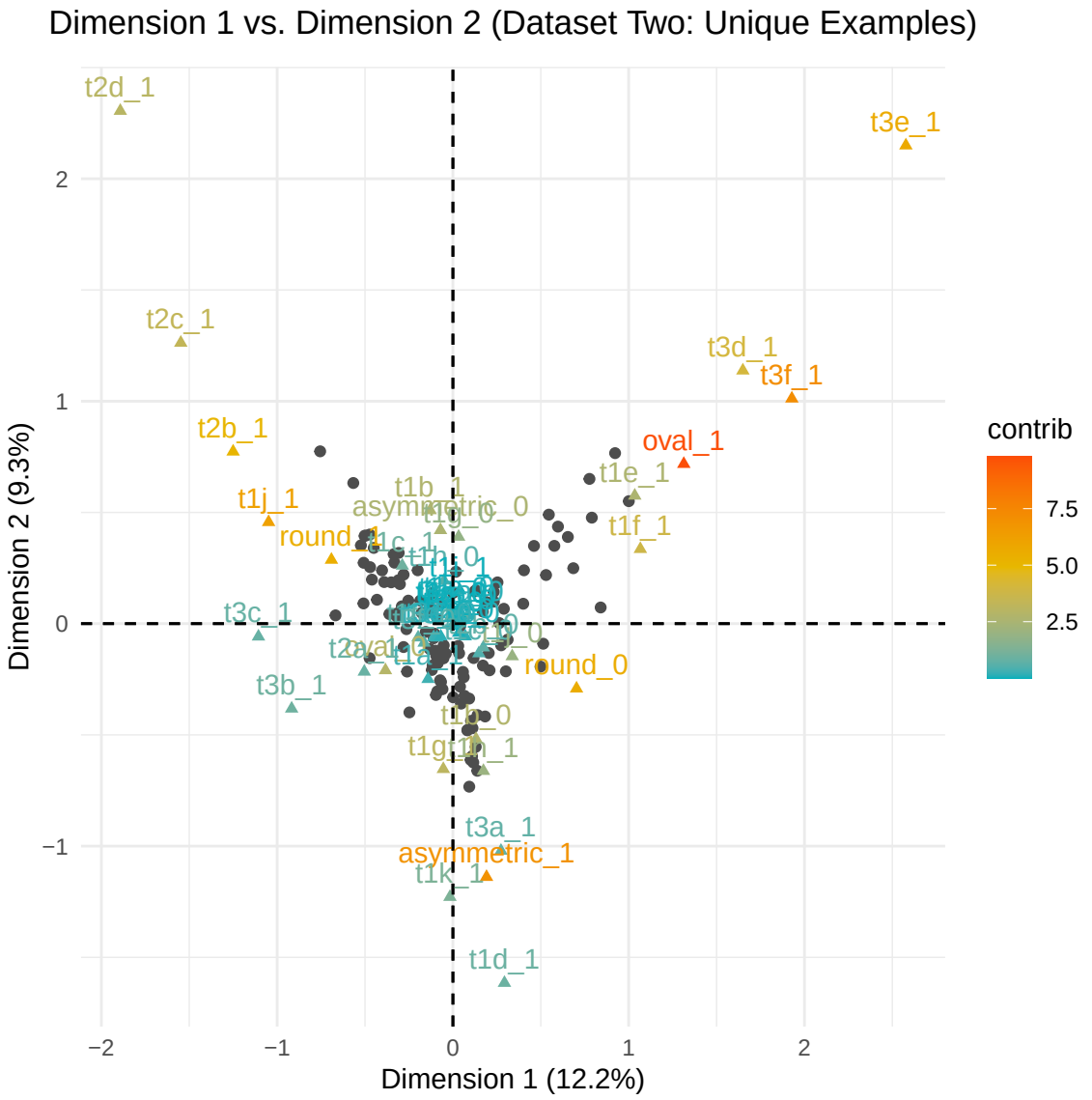

```
fviz_mca(mcadataoneu, col.var = "contrib", label = "var", gradient.cols = c("#00AFBB",
    "#E7B800", "#FC4E07"), col.ind = "grey30", pointsize = 1.5) +
    labs(title = "Dimension 1 vs. Dimension 2 (Dataset One: Unique Examples)",
        x = "Dimension 1 (10.9%)", y = "Dimension 2 (7.8%)") +
    theme_minimal() + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



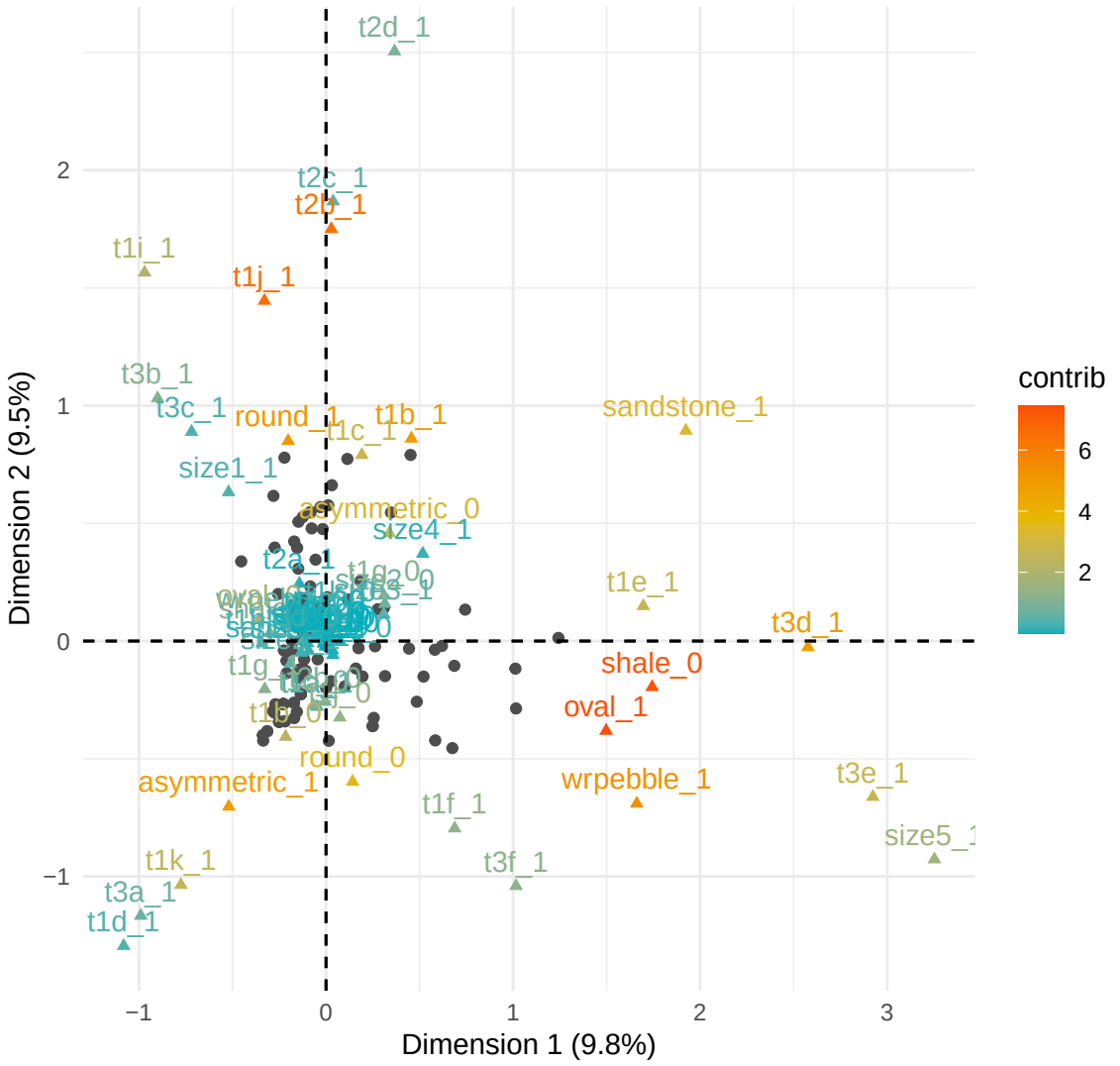

```
fviz_mca(mcadatatwo, col.var = "contrib", label = "var", gradient.cols = c("#00AFBB",
    "#E7B800", "#FC4E07"), col.ind = "grey30", pointsize = 1.5) +
    labs(title = "Dimension 1 vs. Dimension 2 (Dataset Two: All Examples)",
        x = "Dimension 1 (12.3%)", y = "Dimension 2 (11.7%)") +
    theme_minimal() + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



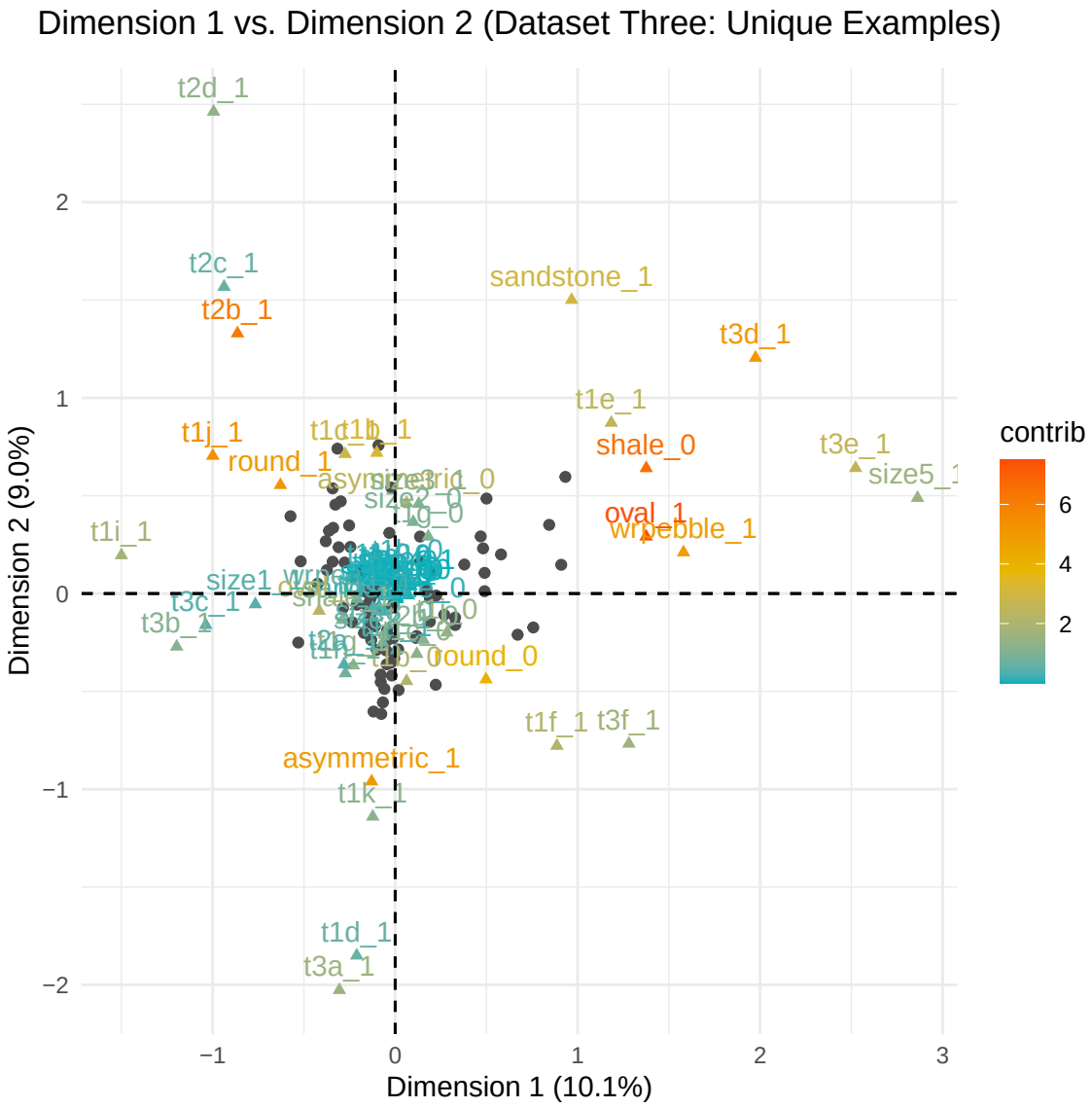

```
fviz_mca(mcadatatwou, col.var = "contrib", label = "var", gradient.cols = c("#00AFBB",
    "#E7B800", "#FC4E07"), col.ind = "grey30", pointsize = 1.5) +
    labs(title = "Dimension 1 vs. Dimension 2 (Dataset Two: Unique Examples)",
        x = "Dimension 1 (12.2%)", y = "Dimension 2 (9.3%)") +
    theme_minimal() + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```




```
fviz_mca(mcadatathree, col.var = "contrib", label = "var", gradient.cols = c("#00AFBB",
    "#E7B800", "#FC4E07"), col.ind = "grey30", pointsize = 1.5) +
    labs(title = "Dimension 1 vs. Dimension 2 (Dataset Three: All Examples)",
        x = "Dimension 1 (9.8%)", y = "Dimension 2 (9.5%)") +
    theme_minimal() + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```




```
fviz_mca(mcadatathreeu, col.var = "contrib", label = "var", gradient.cols = c("#00AFBB",
    "#E7B800", "#FC4E07"), col.ind = "grey30", pointsize = 1.5) +
    labs(title = "Dimension 1 vs. Dimension 2 (Dataset Three: Unique Examples)",
        x = "Dimension 1 (10.1%)", y = "Dimension 2 (9.0%)") +
    theme_minimal() + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



Network Analysis

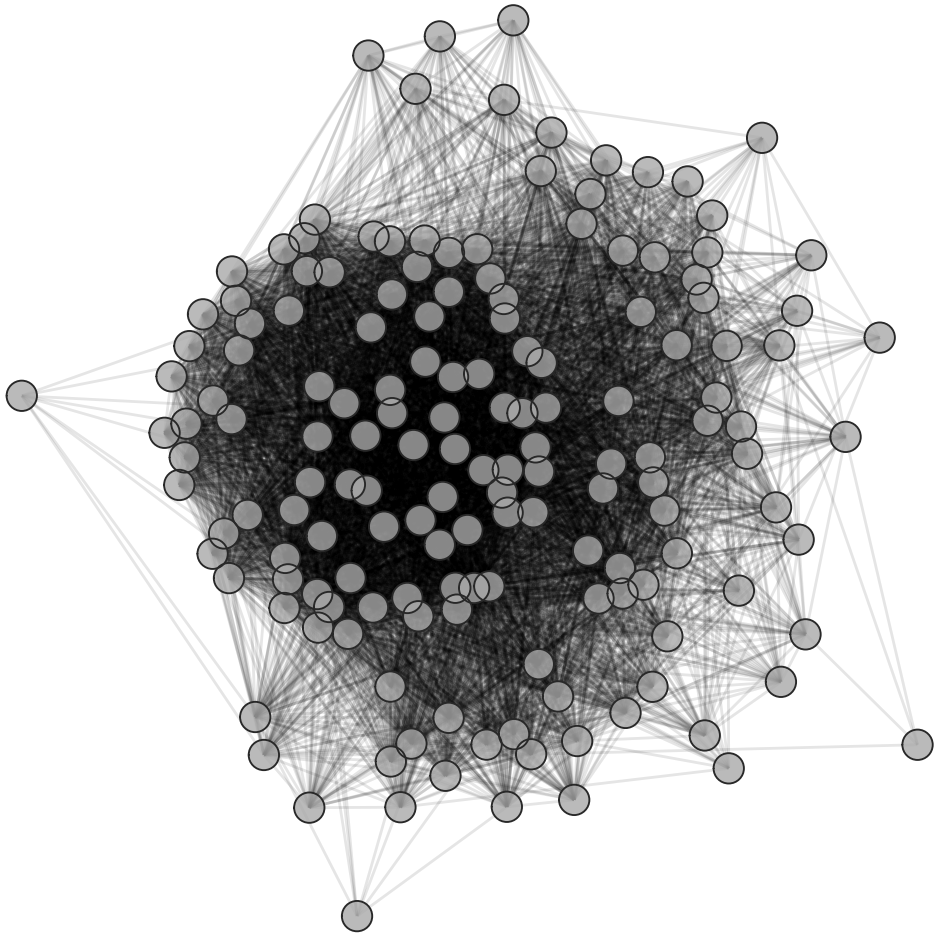
To further explore the underlying structure of similarity and dissimilarity between artefacts, and the different aspects of sun-stone design, tools from network science, now commonly adopted by archaeologists (Brughmans, 2010; 2013; Knappett, 2014; Östborn, 2014; Collar et al. 2015; Brughmans and Peeples, 2017), were used. A bipartite igraph object was first created, using the dichtomous values and the `igraph::graph_from_incidence_matrix` and `igraph::bipartite.projection` function. A bipartite graph (sensu Asratian et al. 1998) is a graph whose vertices can be divided into two independent sets; in this example, the procedure will result in two separate 'colourable' graphs - one on the betweenness and similarity in overall design for each artefact, and one focused on the betweenness and similarity of motifs. **Note: these procedures are only applicable to datasets with unique values**.

```
network_one <- graph_from_incidence_matrix(dataset_one_unique)
network_two <- graph_from_incidence_matrix(dataset_two_unique)
network_three <- graph_from_incidence_matrix(dataset_three_unique)
```

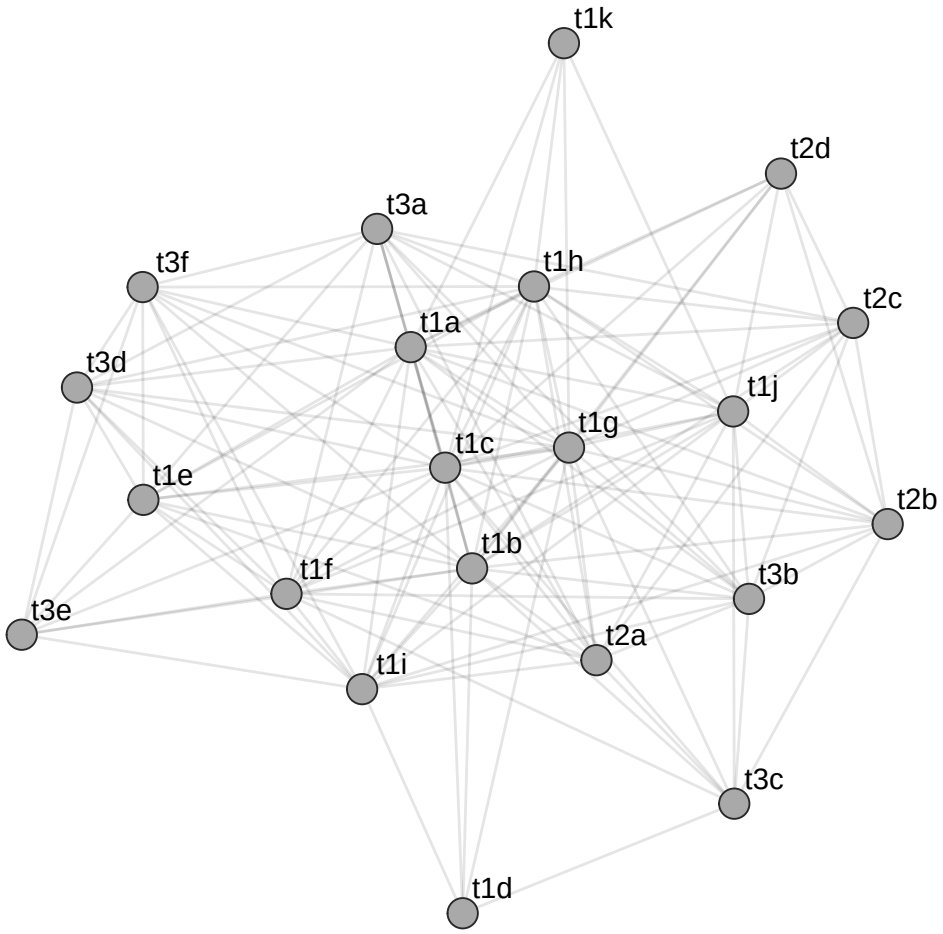
```
network_one_bp <- bipartite.projection(network_one)
network_two_bp <- bipartite.projection(network_two)
network_three_bp <- bipartite.projection(network_three)
```

With the visualisation tools and algorithms in both `igraph` and `ggraph` we can now visualise the structure in both sets of graphs within the bipartite projection:

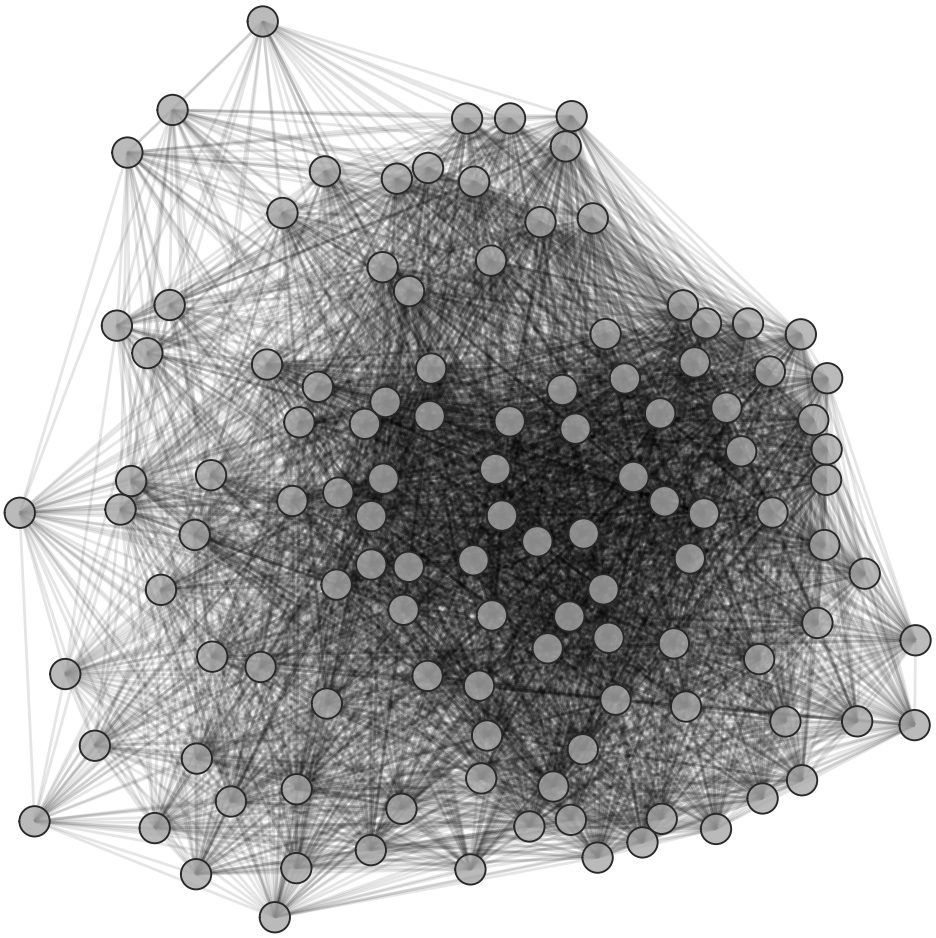
```
ggraph(network_one_bp$proj1, layout = "stress") + geom_edge_link(alpha = 0.1) +
    geom_node_point(colour = "#a9a9a9", alpha = 0.8, shape = 16,
        size = 5) + geom_node_point(shape = 1, size = 5, colour = "#282828") +
    theme_graph(background = NA)
```



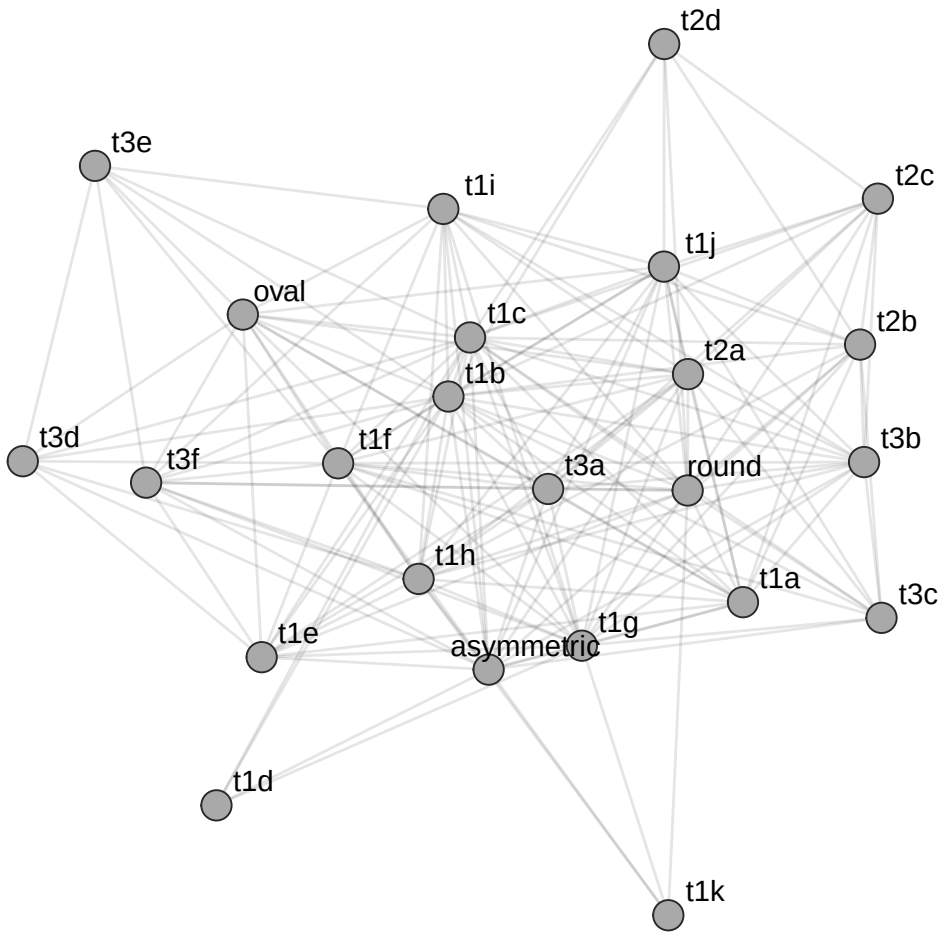
```
ggraph(network_one_bp$proj2, layout = "stress") + geom_edge_link(alpha = 0.1) +
    geom_node_point(colour = "#a9a9a9", shape = 16, size = 5) +
    geom_node_point(shape = 1, size = 5, colour = "#282828") +
    geom_node_text(aes(label = name), nudge_x = 0.08, nudge_y = 0.08) +
    theme_graph(background = NA)
```



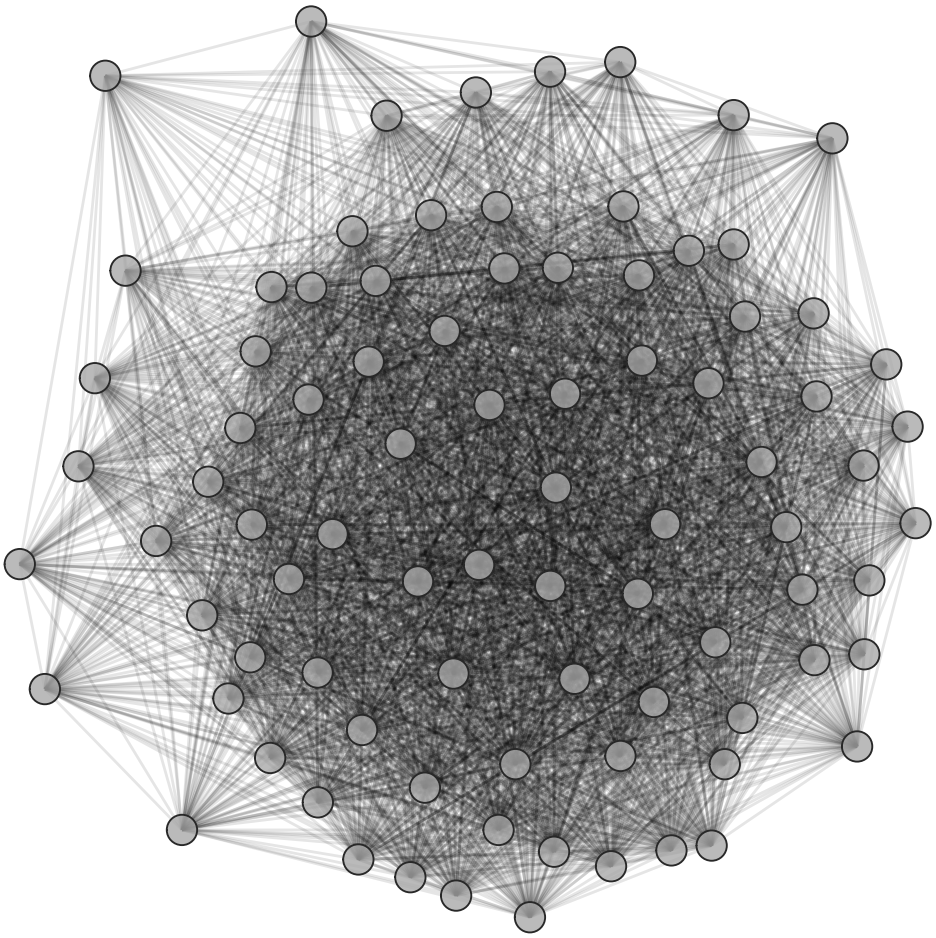

```
ggraph(network_two_bp$proj1, layout = "stress") + geom_edge_link(alpha = 0.1) +
    geom_node_point(colour = "#a9a9a9", alpha = 0.8, shape = 16,
        size = 5) + geom_node_point(shape = 1, size = 5, colour = "#282828") +
    theme_graph(background = NA)
```



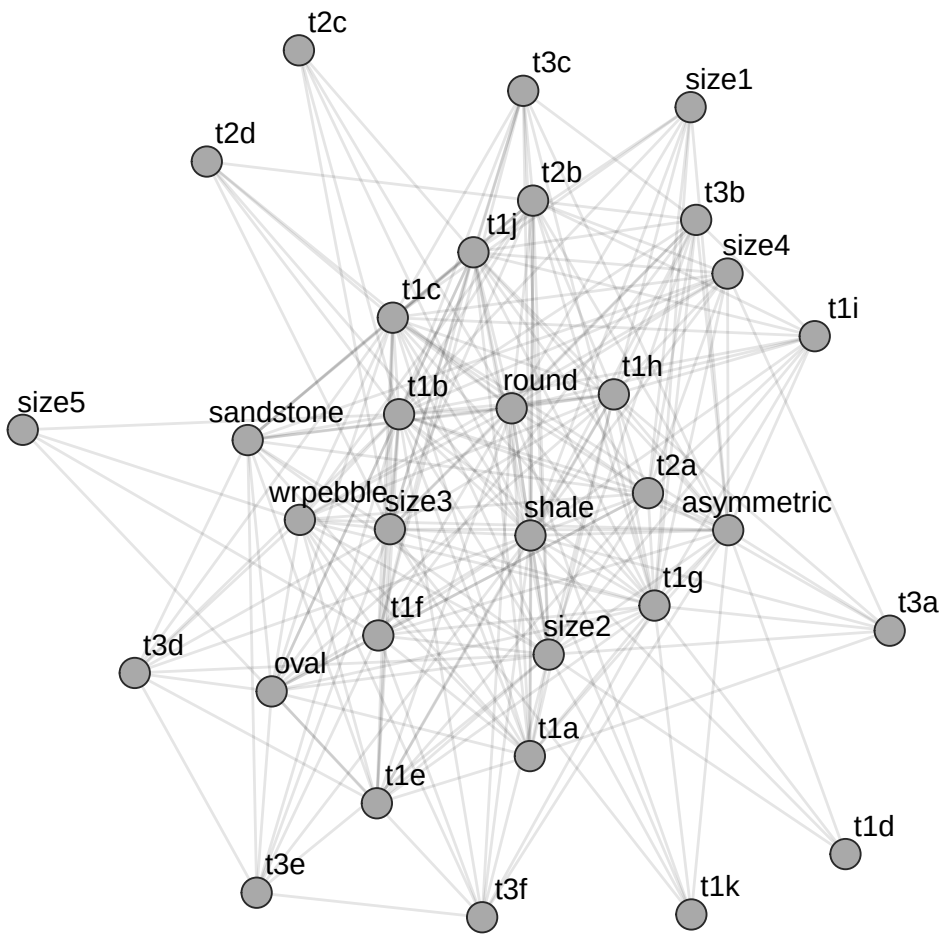
```
ggraph(network_two_bp$proj2, layout = "stress") + geom_edge_link(alpha = 0.1) +
    geom_node_point(colour = "#a9a9a9", shape = 16, size = 5) +
    geom_node_point(shape = 1, size = 5, colour = "#282828") +
    geom_node_text(aes(label = name), nudge_x = 0.1, nudge_y = 0.08) +
    theme_graph(background = NA)
```




```
ggraph(network_three_bp$proj1, layout = "stress") + geom_edge_link(alpha = 0.1) +
    geom_node_point(colour = "#a9a9a9", alpha = 0.8, shape = 16,
        size = 5) + geom_node_point(shape = 1, size = 5, colour = "#282828") +
    theme_graph(background = NA)
```



```
ggraph(network_three_bp$proj2, layout = "stress") + geom_edge_link(alpha = 0.1) +
    geom_node_point(colour = "#a9a9a9", shape = 16, size = 5) +
    geom_node_point(shape = 1, size = 5, colour = "#282828") +
    geom_node_text(aes(label = name), nudge_x = 0.1, nudge_y = 0.08) +
    theme_graph(background = NA)
```



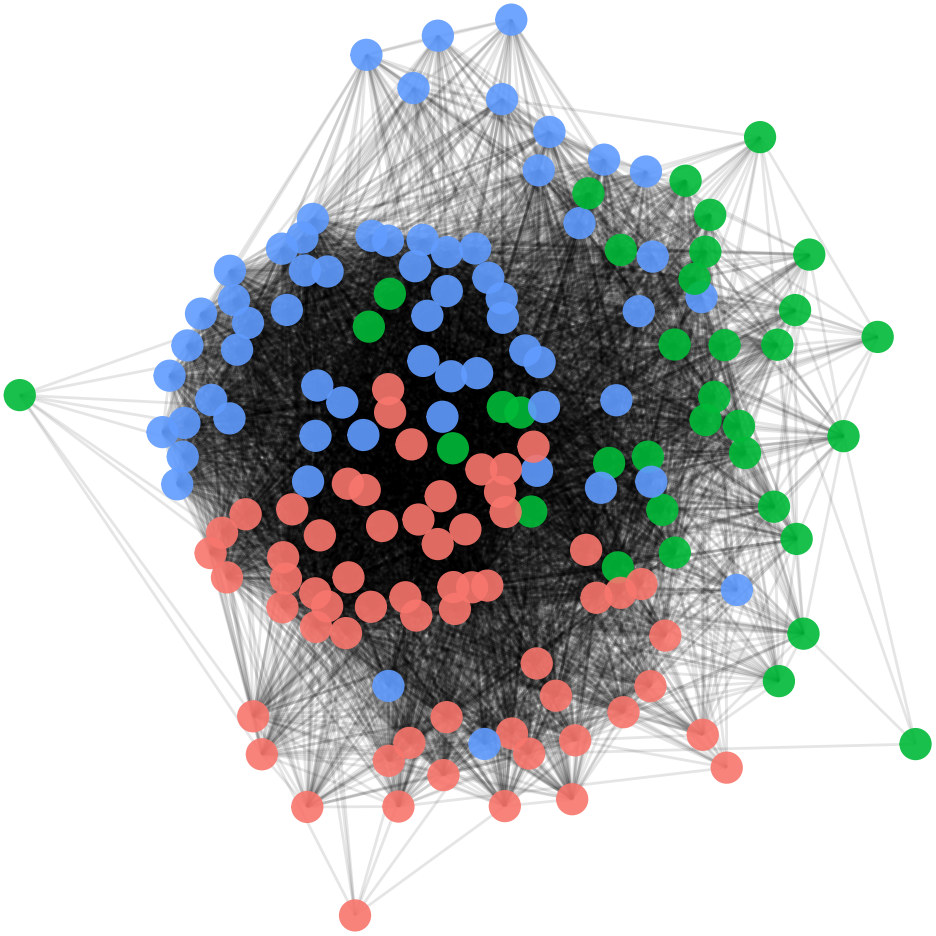
Note: the stress algorithm adopted here is a modified form of the Kamada-Kawai ('kk') layout, with a different optimisation strategy.

In order to identify community structure, that is to say nodes of a network which can be categorised into (potentially overlapping) sets of nodes such that each set of nodes is densely connected internally, an investigation of modularity optimisation was undertaken, using a hierarchical agglomeration algorithm *sensu* Clauset et al. (2004). This is performed in igraph using the `igraph::cluster_fast_greedy` function, and is executed on each projection (two) of each dataset (three). Note: the number of communities is not dictated by the user, but rather the result of the algorithm.

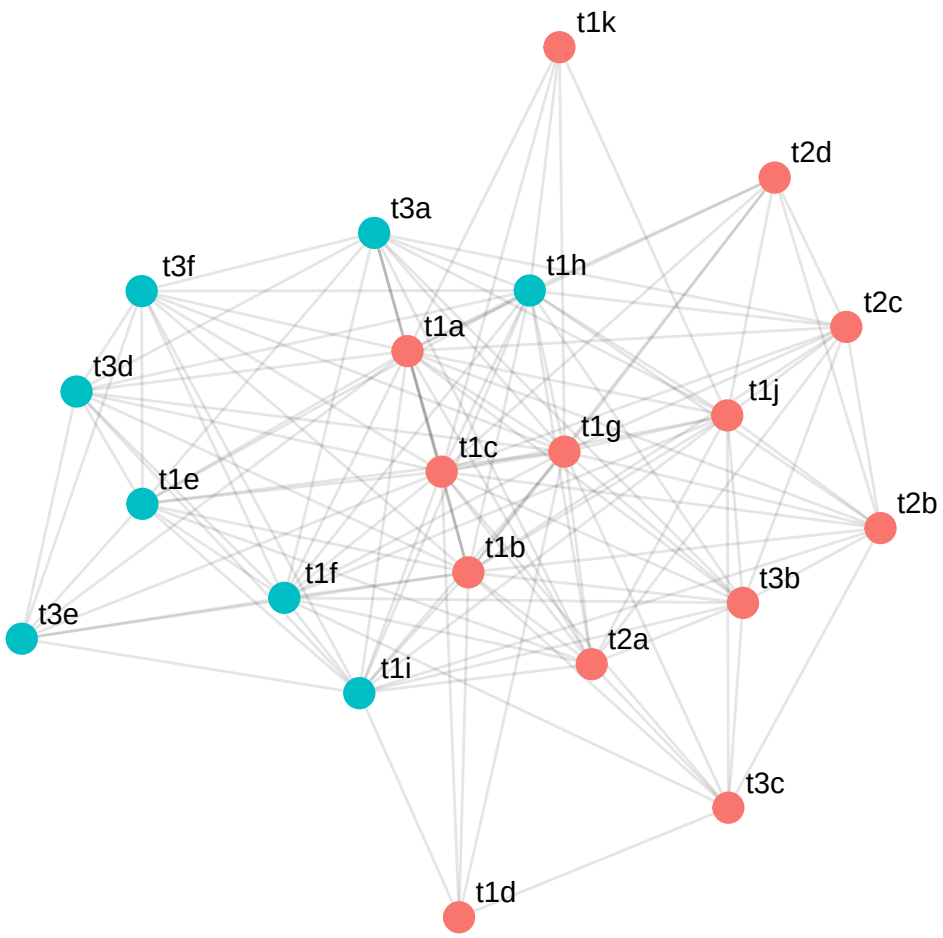
```
mod.network_1p1 <- cluster_fast_greedy(network_one_bp$proj1)
mod.network_1p2 <- cluster_fast_greedy(network_one_bp$proj2)
mod.network_2p1 <- cluster_fast_greedy(network_two_bp$proj1)
mod.network_2p2 <- cluster_fast_greedy(network_two_bp$proj2)
mod.network_3p1 <- cluster_fast_greedy(network_three_bp$proj1)
mod.network_3p2 <- cluster_fast_greedy(network_three_bp$proj2)
```

We can then project these clusters using similar ggraph and igraph functions:

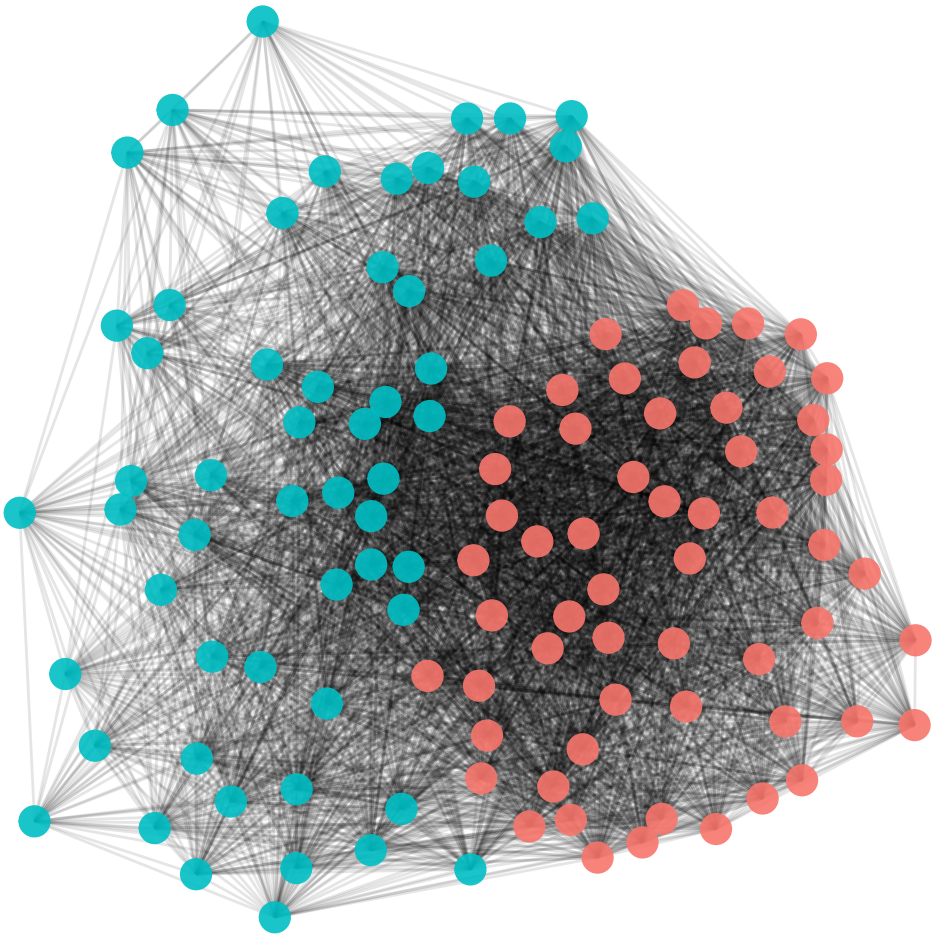
```
ggraph(network_one_bp$proj1, layout = "stress") + geom_edge_link(alpha = 0.1) +
    geom_node_point(aes(colour = as.factor(mod.network_1p1$membership)),
        alpha = 0.9, size = 5) + theme_graph(background = NA) +
    theme(legend.position = "none")
```



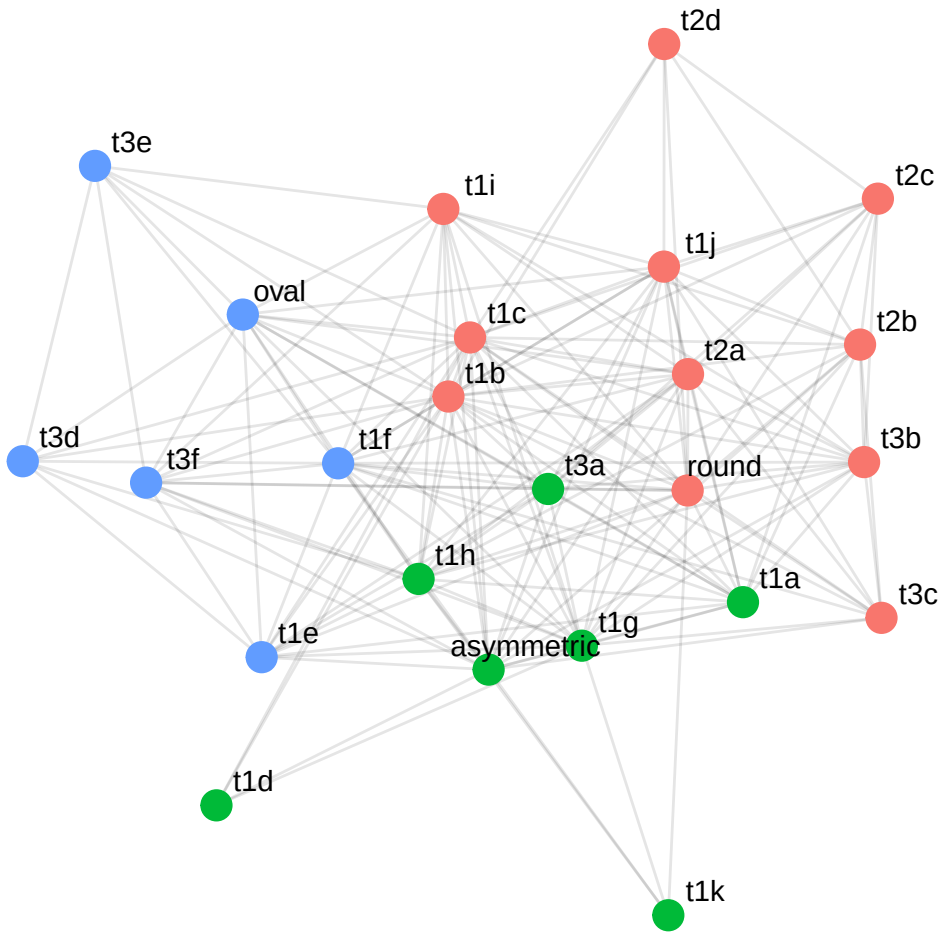
```
ggraph(network_one_bp$proj2, layout = "stress") + geom_edge_link(alpha = 0.1) +
    geom_node_point(aes(colour = as.factor(mod.network_1p2$membership)),
        size = 5) + geom_node_text(aes(label = name), nudge_x = 0.1,
    nudge_y = 0.08) + theme_graph(background = NA) + theme(legend.position = "none")
```



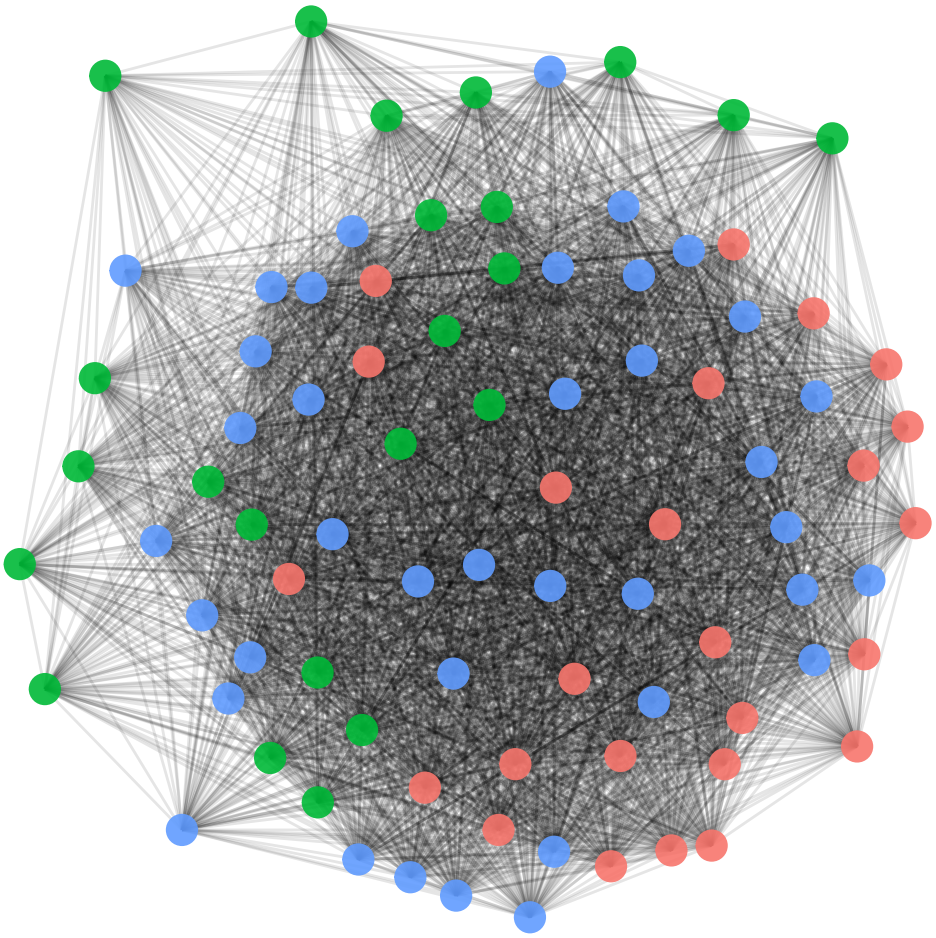

```
ggraph(network_two_bp$proj1, layout = "stress") + geom_edge_link(alpha = 0.1) +
    geom_node_point(aes(colour = as.factor(mod.network_2p1$membership)),
        alpha = 0.9, size = 5) + theme_graph(background = NA) +
    theme(legend.position = "none")
```



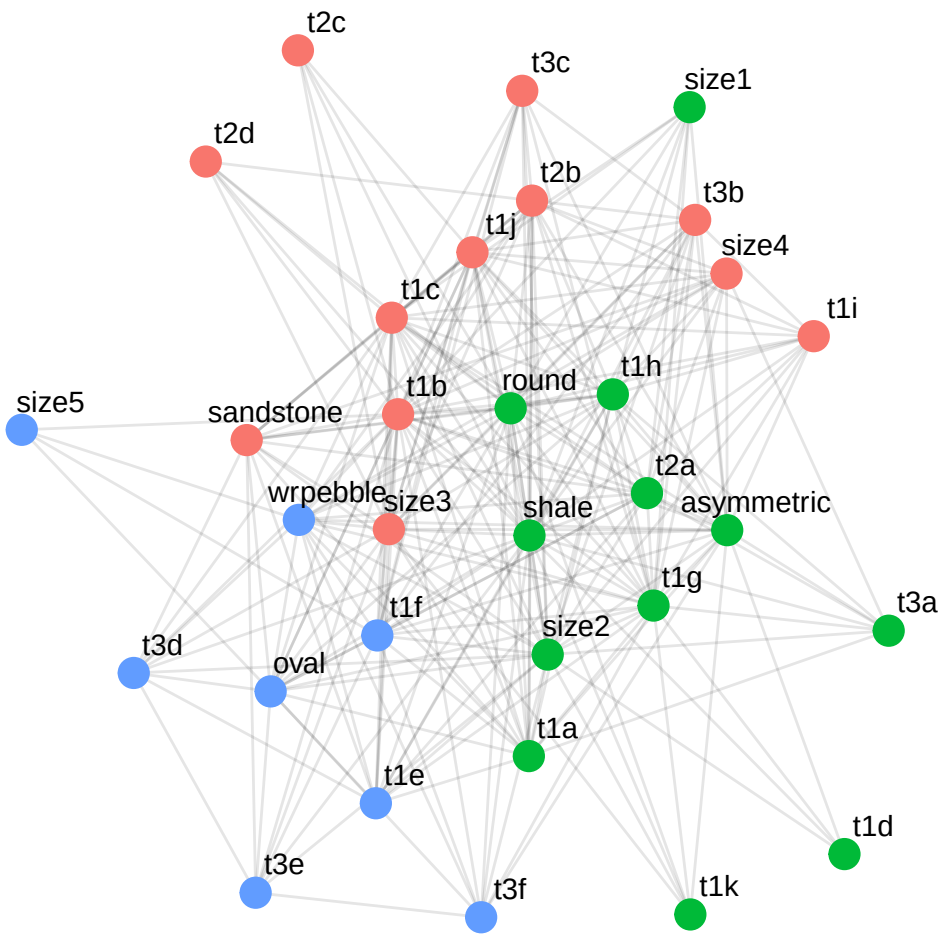
```
ggraph(network_two_bp$proj2, layout = "stress") + geom_edge_link(alpha = 0.1) +
    geom_node_point(aes(colour = as.factor(mod.network_2p2$membership)),
        size = 5) + geom_node_text(aes(label = name), nudge_x = 0.1,
    nudge_y = 0.08) + theme_graph(background = NA) + theme(legend.position = "none")
```




```
ggraph(network_three_bp$proj1, layout = "stress") + geom_edge_link(alpha = 0.1) +
    geom_node_point(aes(colour = as.factor(mod.network_3p1$membership)),
        alpha = 0.9, size = 5) + theme_graph(background = NA) +
    theme(legend.position = "none")
```



```
ggraph(network_three_bp$proj2, layout = "stress") + geom_edge_link(alpha = 0.1) +
    geom_node_point(aes(colour = as.factor(mod.network_3p2$membership)),
        size = 5) + geom_node_text(aes(label = name), nudge_x = 0.1,
    nudge_y = 0.08) + theme_graph(background = NA) + theme(legend.position = "none")
```



Tree-Building Exercises

For the last three decades evolutionary archaeologists have adapted micro-evolutionary (population genetic) and macro-evolutionary (phylogenetic) methods to better understand the nature of change in material culture (O'Brien and Lyman, 2000 Bettinger, 2008; Shennan, 2008, 2009; Riede et al. 2019; Ivanovaite et al. 2019). Through this notion of culture as a multi-generational system of information transmission *sensu* Boyd and Richerson (1985) past communities of practice, estimations of contact and mixing have been investigated, and the nature of transmission hypothesised (O'Brien et al. 2008; Collard and Shennan, 2008; Riede 2011).

To fully explore the underlying structure of variation within the sun-stones, and to hypothesise the underwritten system of information transmission across the production of the sun-stones, a variety of different tree-building exercises are explored here - they can be here categorised into **agglomerative hierarchical clustering** and **parsimony-based phylogenetic** approaches. The first set of methodologies provide an underlying basis of variation, similar to the methodologies above, using measures calculated from distance matrices. The second set of methodologies are more computationally intensive and use the same distance matrices to determine the hypothesised evolutionary relationships between the artefacts through the assumption that the relationships with the smallest number of character changes is most likely to be correct, or to

quote Edwards and Cavalli-Sforza (1963: 553): *"the most plausible estimate of the evolutionary tree is that which invokes the minimum net amount of evolution."* While the latter methodologies have been grounded in systematic biology for some time, applications of parsimony-based phylogenetic analyses are more recent (O'Brien et al. 2001; Mace et al. 2005; Lipo et al. 2006).

Many variants of agglomerative hierarchical clustering exist, three common methods often utilised by archaeologists (Bouby et al. 2013; D'Huy, 2013; Podani and Morrison, 2017; Sutikna et al. 2018) are performed here: **neighbour joining (NJ)** *sensu* Saitou and Nei (1987) and Studier and Keppler (1988), **unweighted pair group method with arithmetic mean (UPGMA)** *sensu* Sneath and Sokal (1973) and **weighted pair group method with arithmetic mean (WPGMA)** *sensu* Sokal (1958).

In each of the three agglomerative methods, distance matrices must be constructed. This is done using `base::dist` for the complete dataset and for unique values:

```
distmatrixone <- dist(dataset_one, method = "binary")
distmatrixoneunique <- dist(dataset_one_unique, method = "binary")
distmatrixtwo <- dist(dataset_two_clean, method = "binary")
distmatrixtwounique <- dist(dataset_two_unique, method = "binary")
distmatrixthree <- dist(dataset_three_clean, method = "binary")
distmatrixthreeunique <- dist(dataset_three_unique, method = "binary")
```

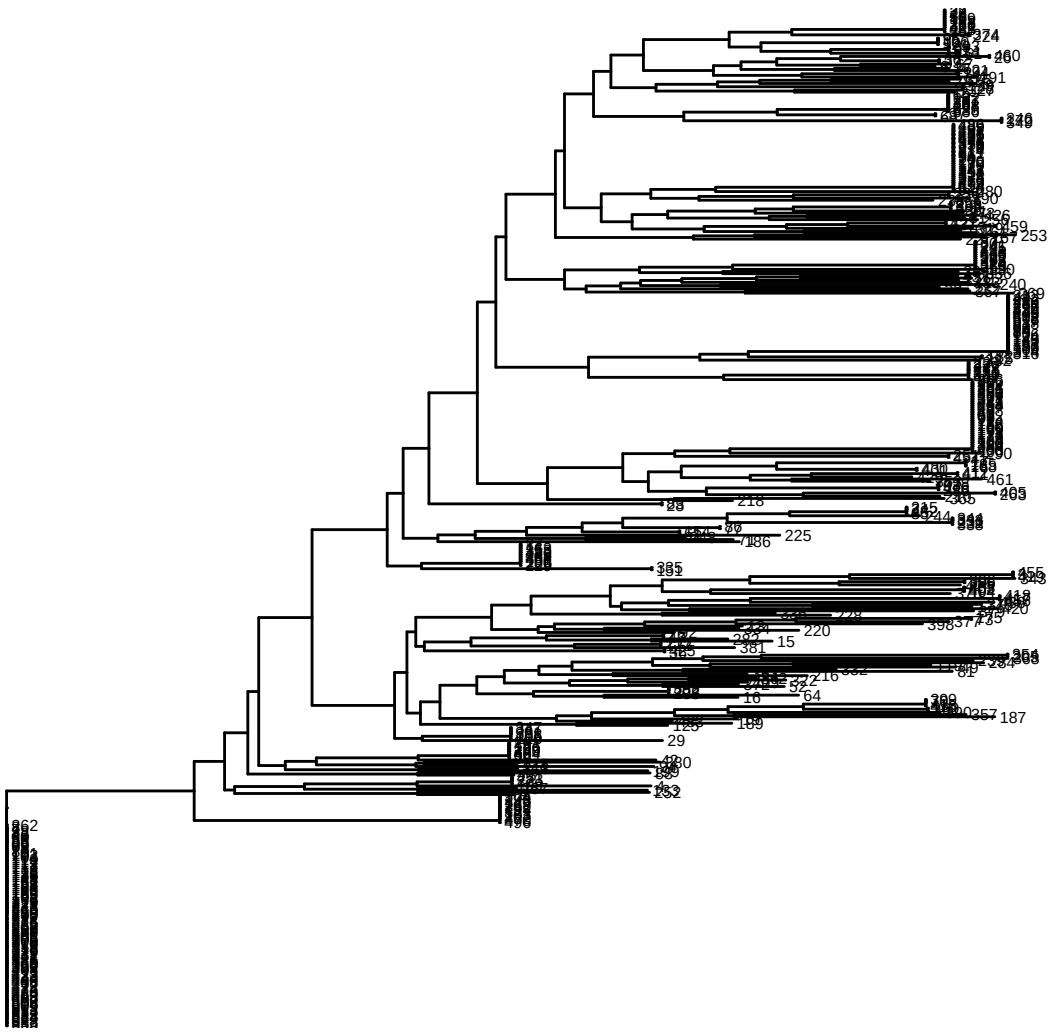
Neighbour Joining (NJ)

Using the distance matrices above, we can use the `ape::nj()` function to generate the estimated trees.

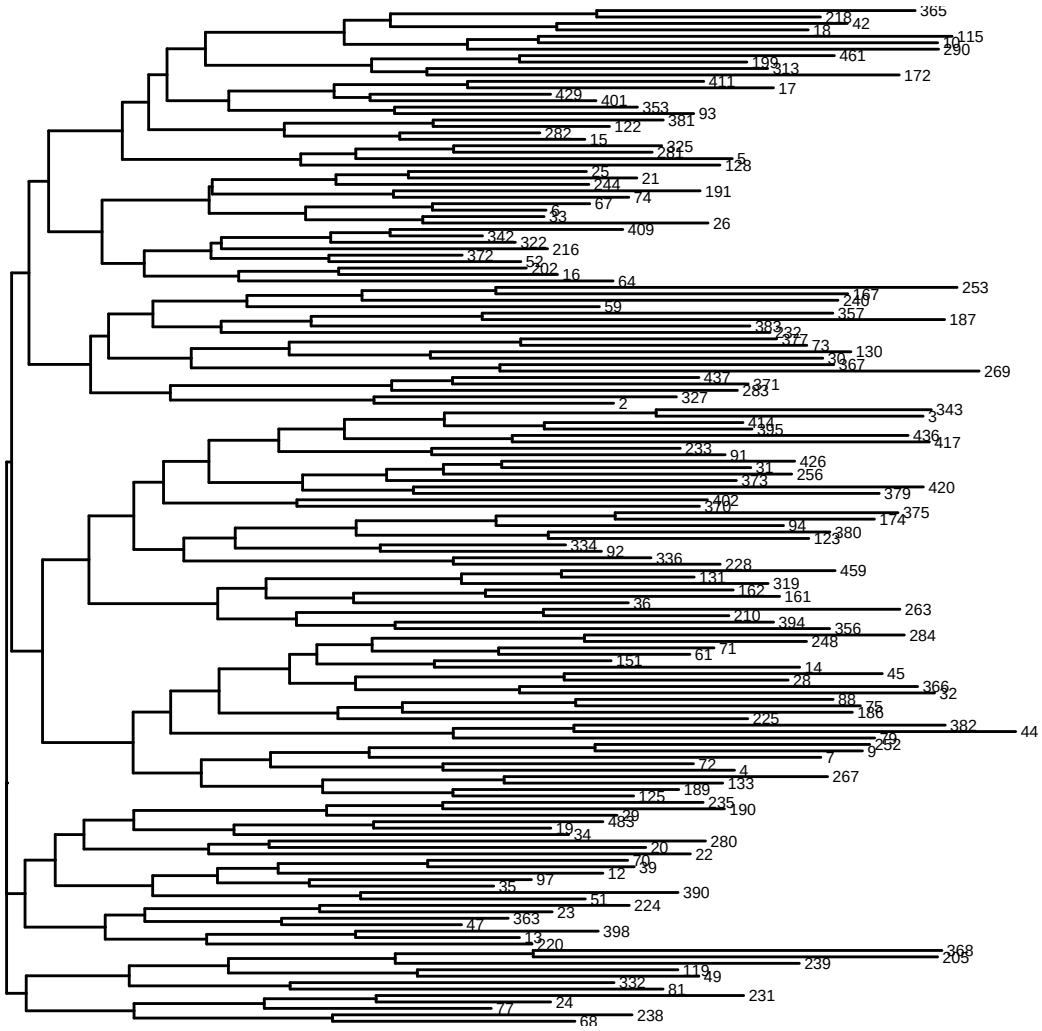
```
treenjone <- nj(distmatrixone)
treenjoneu <- nj(distmatrixoneunique)
treenjtwo <- nj(distmatrixtwo)
treenjtwou <- nj(distmatrixtwounique)
treenjthree <- nj(distmatrixthree)
treenjthreeu <- nj(distmatrixthreeunique)
```

We can then visualise these trees using the tools in `ggtree`. Each variant (including factors where appropriate are visualised here). Note: metadata objects are created to facilitate the visualisation of groups in `ggtree`.

```
ggtree(treenjone, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("Neighbour Joining\n(Dataset One: All Values)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```

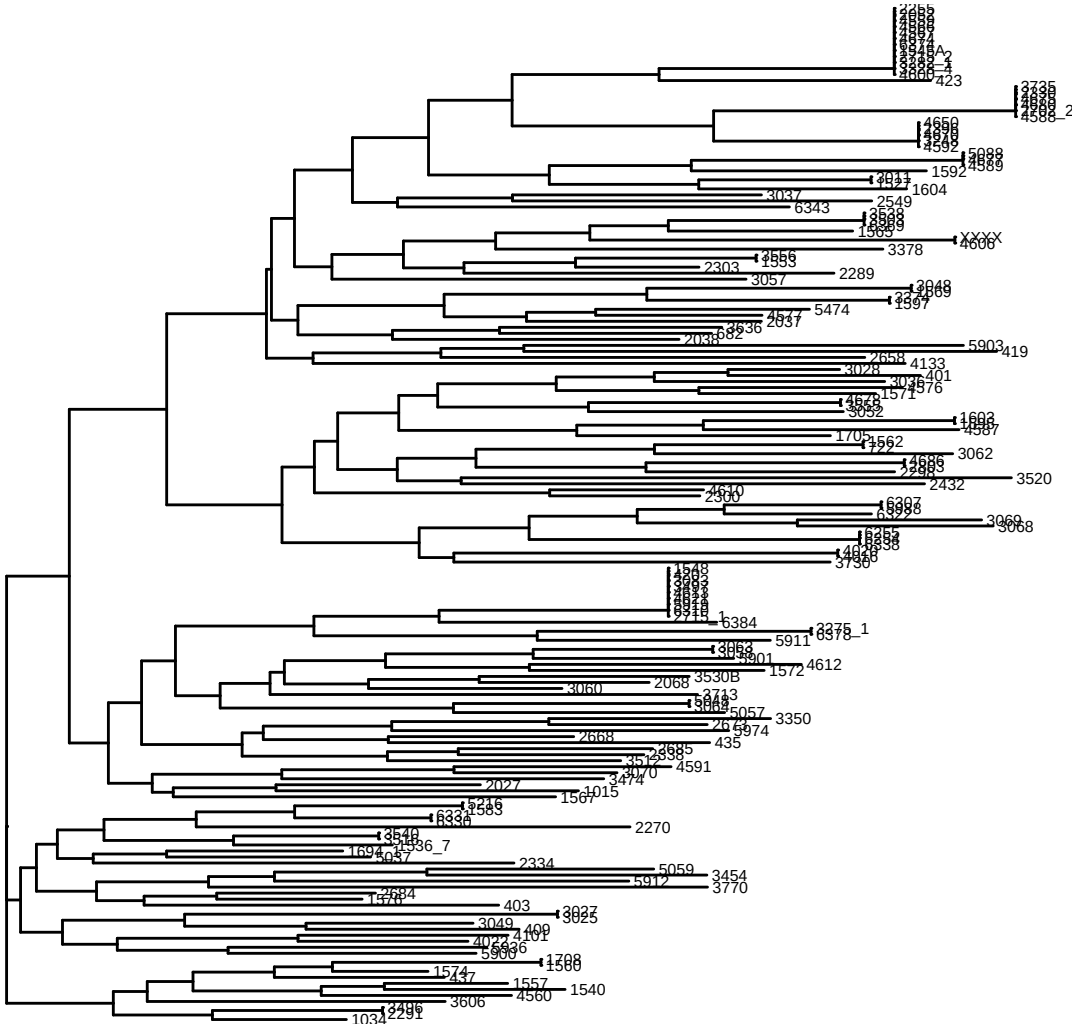



```
ggtree(treenjoneu, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("Neighbour Joining\n(Dataset One: Unique Values)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```

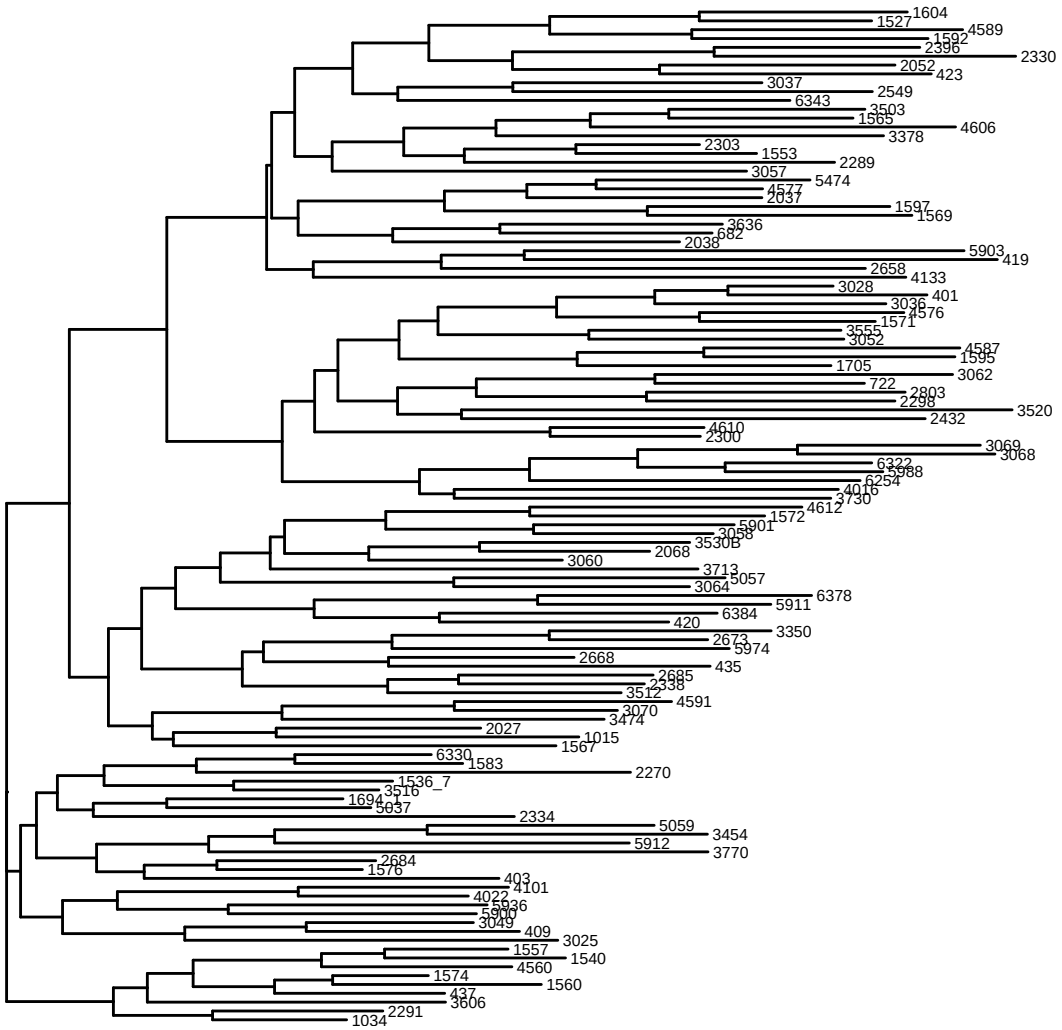



```
ggtree(treenjtwo, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("Neighbour Joining\n(Dataset Two: All Values)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```

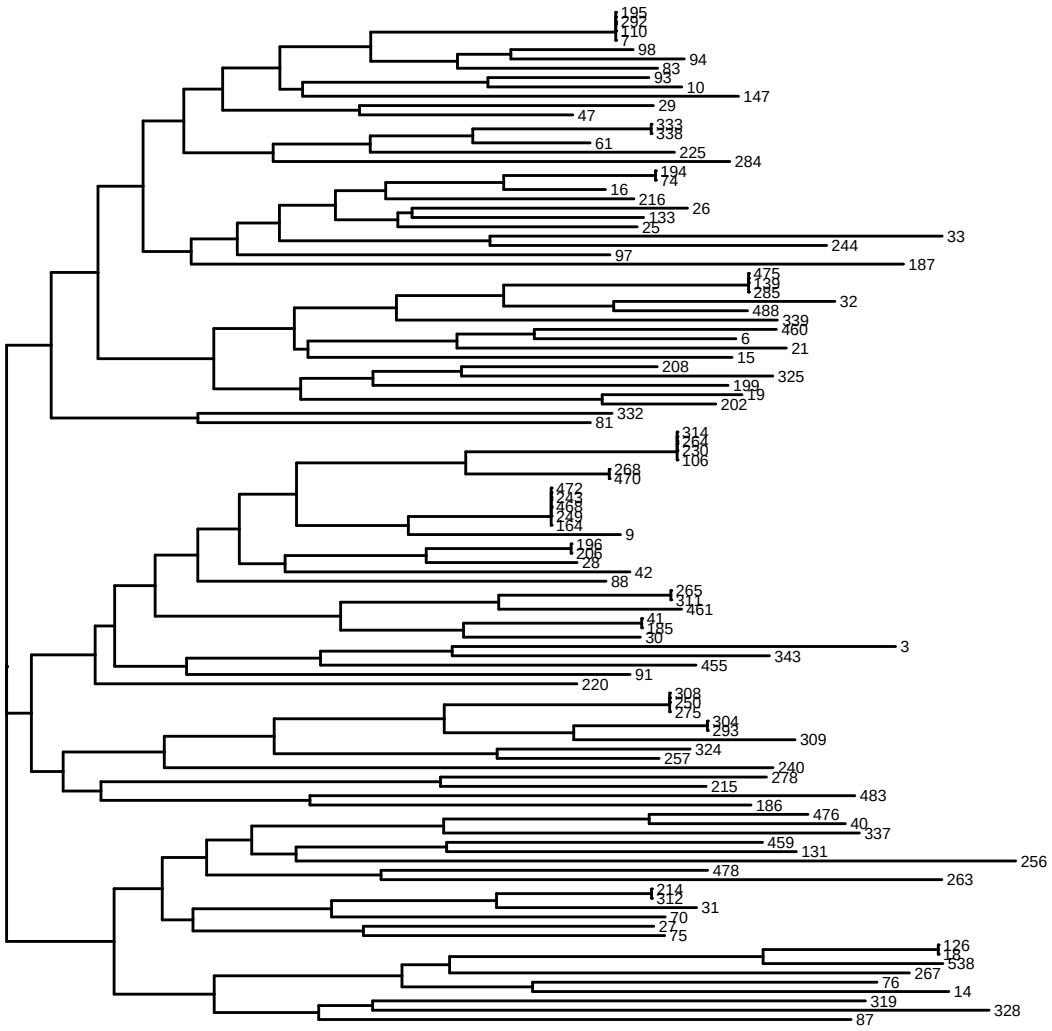
Neighbour Joining
(Dataset Two: All Values)



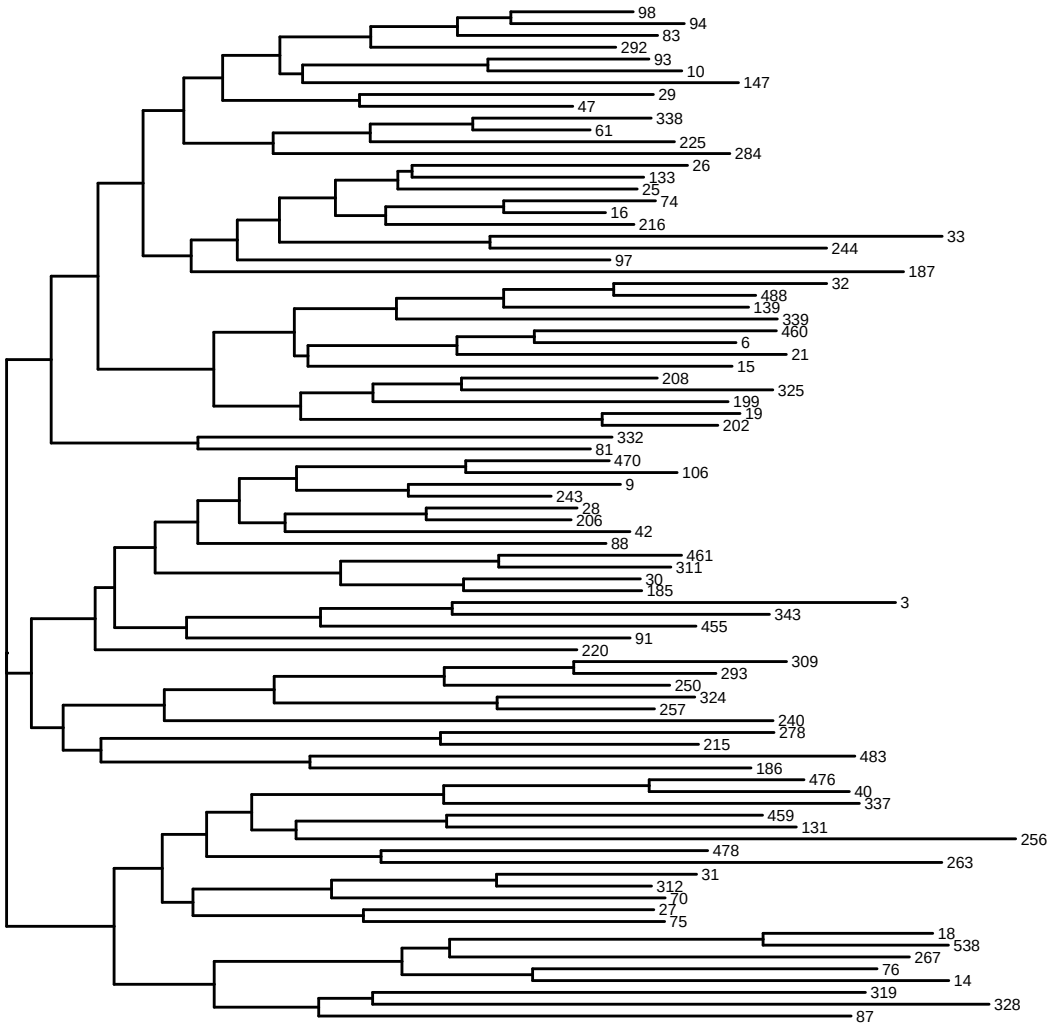

```
ggtree(treenjtwou, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("Neighbour Joining\n(Dataset Two: Unique Values)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```




```
ggtree(treenjthree, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("Neighbour Joining\n(Dataset Three: All Values)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```

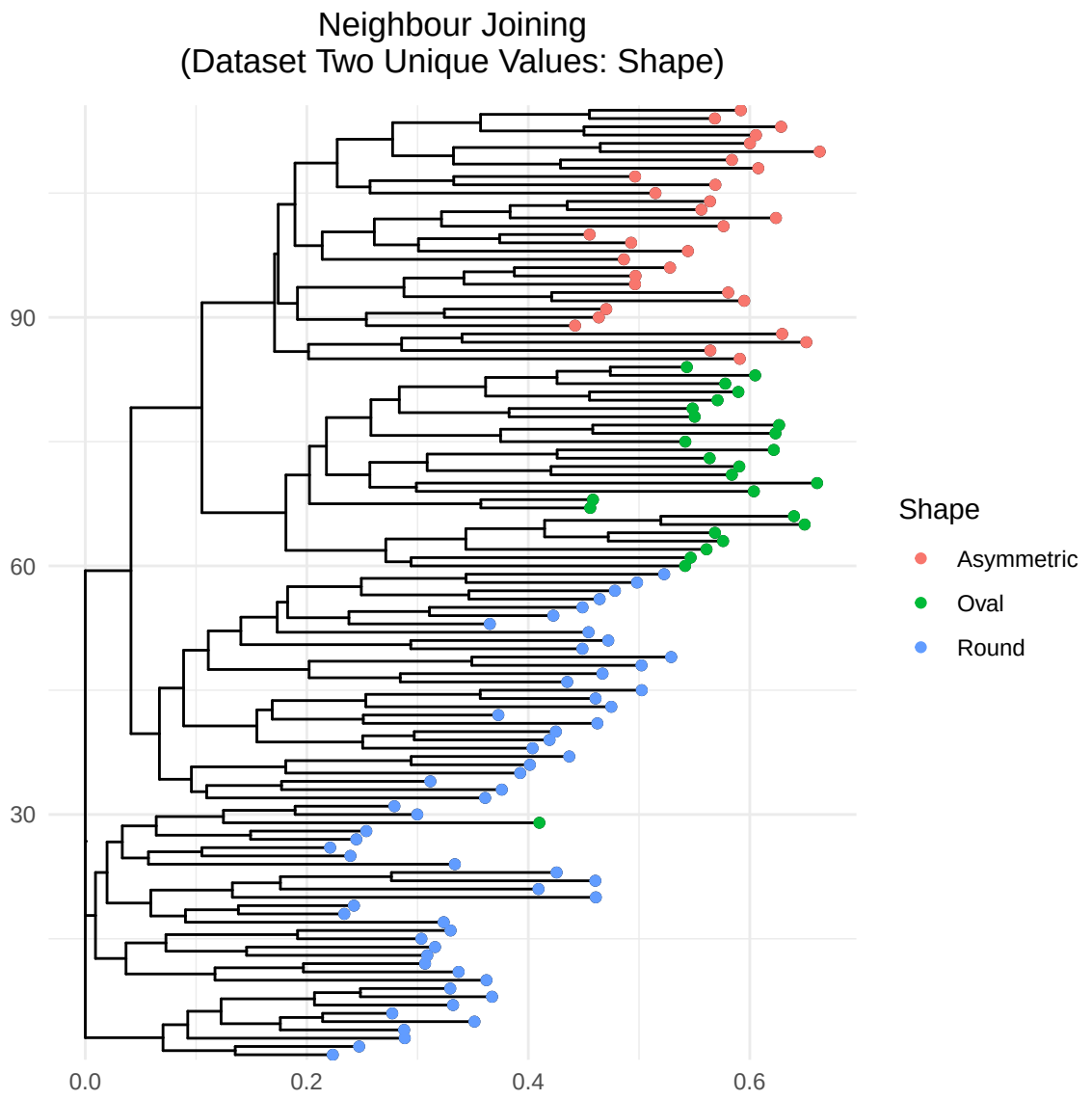



```
ggtree(treenjthreeu, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("Neighbour Joining\n(Dataset Three: Unique Values)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```

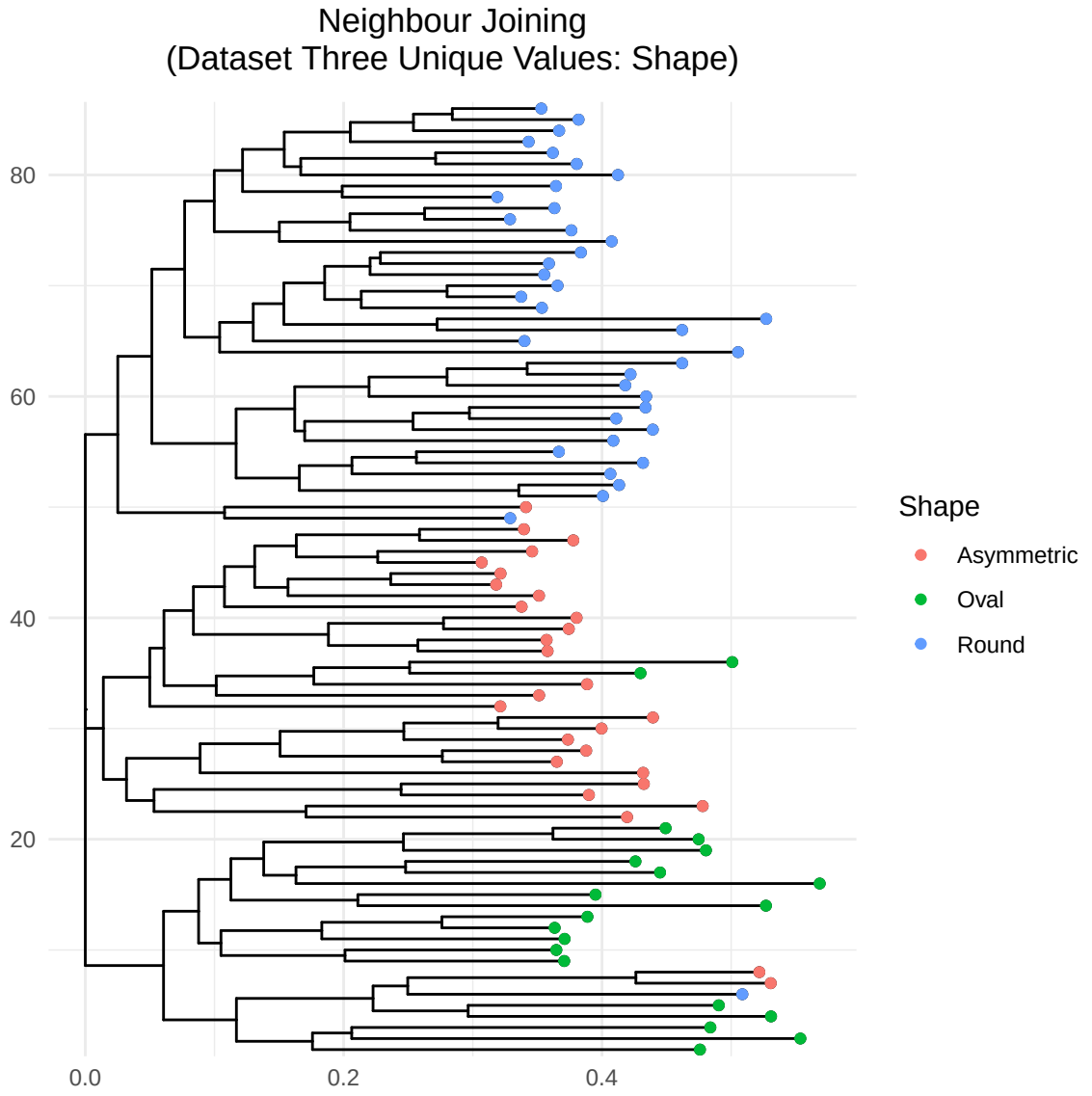



```
dataset_two_full_meta <- dataset_two %>% select(shape) %>% rownames_to_column(var = "id") %>%
    rename(tip.label = id)

dataset_two_unique_meta <- dataset_two %>% unique() %>% select(shape) %>%
    rownames_to_column(var = "id") %>% rename(tip.label = id)

dataset_three_full_meta <- dataset_three %>% select(shape, material,
    size, gcontext, context) %>% rownames_to_column(var = "id") %>%
    rename(tip.label = id)

dataset_three_unique_meta <- dataset_three %>% unique() %>% select(shape,
    material, size, gcontext, context) %>% rownames_to_column(var = "id") %>%
    rename(tip.label = id)

g1 <- ggtree(treenjtwou, layout = "rectangular") + geom_tippoint(size = 1.5)

g2 <- ggtree(treenjthreeu, layout = "rectangular") + geom_tippoint(size = 1.5)
```

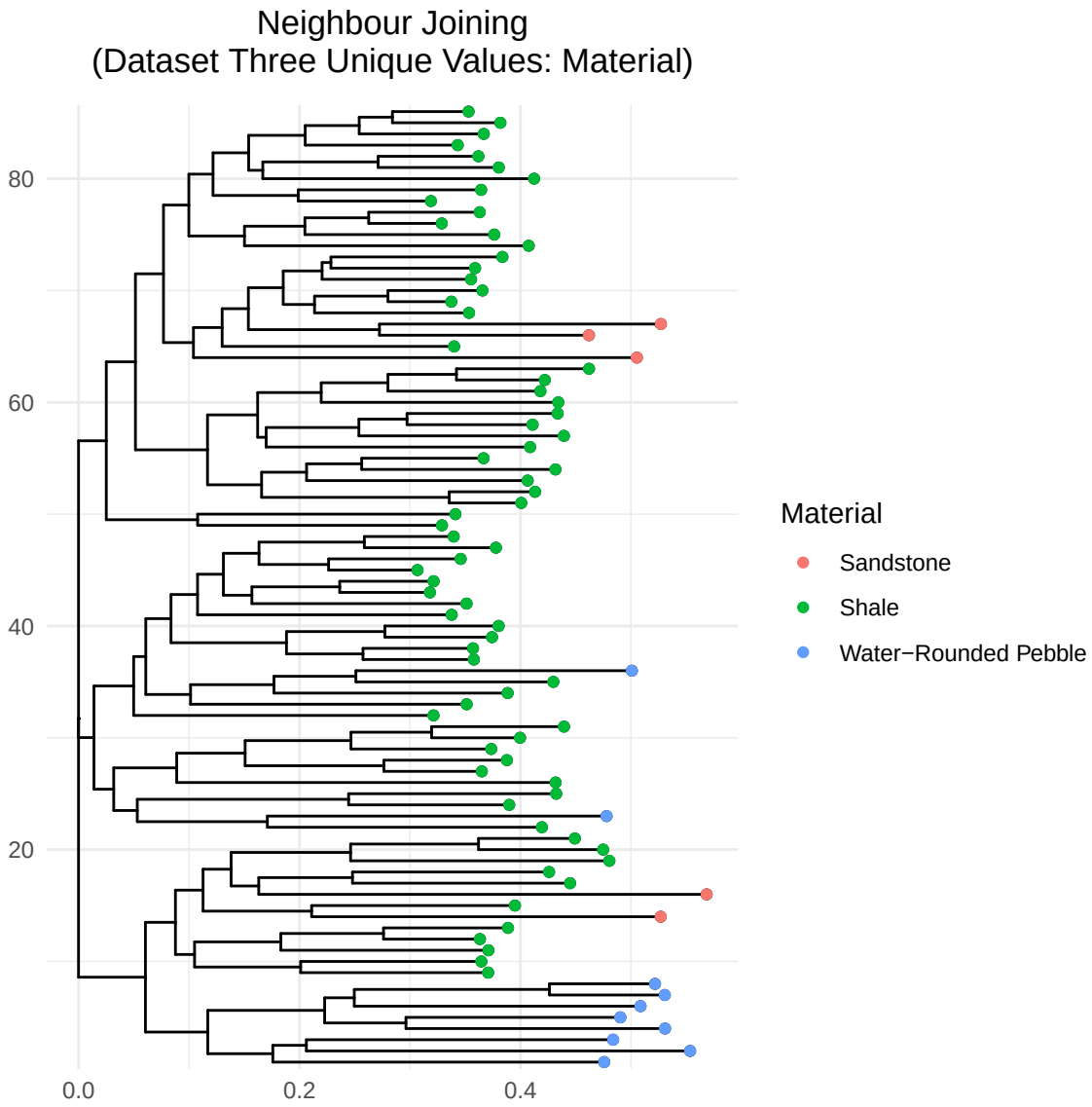
```
g1 %<+% dataset_two_unique_meta + geom_tippoint(aes(colour = shape)) +
    labs(colour = "Shape") + ggtitle("Neighbour Joining\n(Dataset Two Unique Values: Shape)") +
    scale_colour_discrete(name = "Shape", labels = c("Asymmetric",
        "Oval", "Round")) + theme_minimal() + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



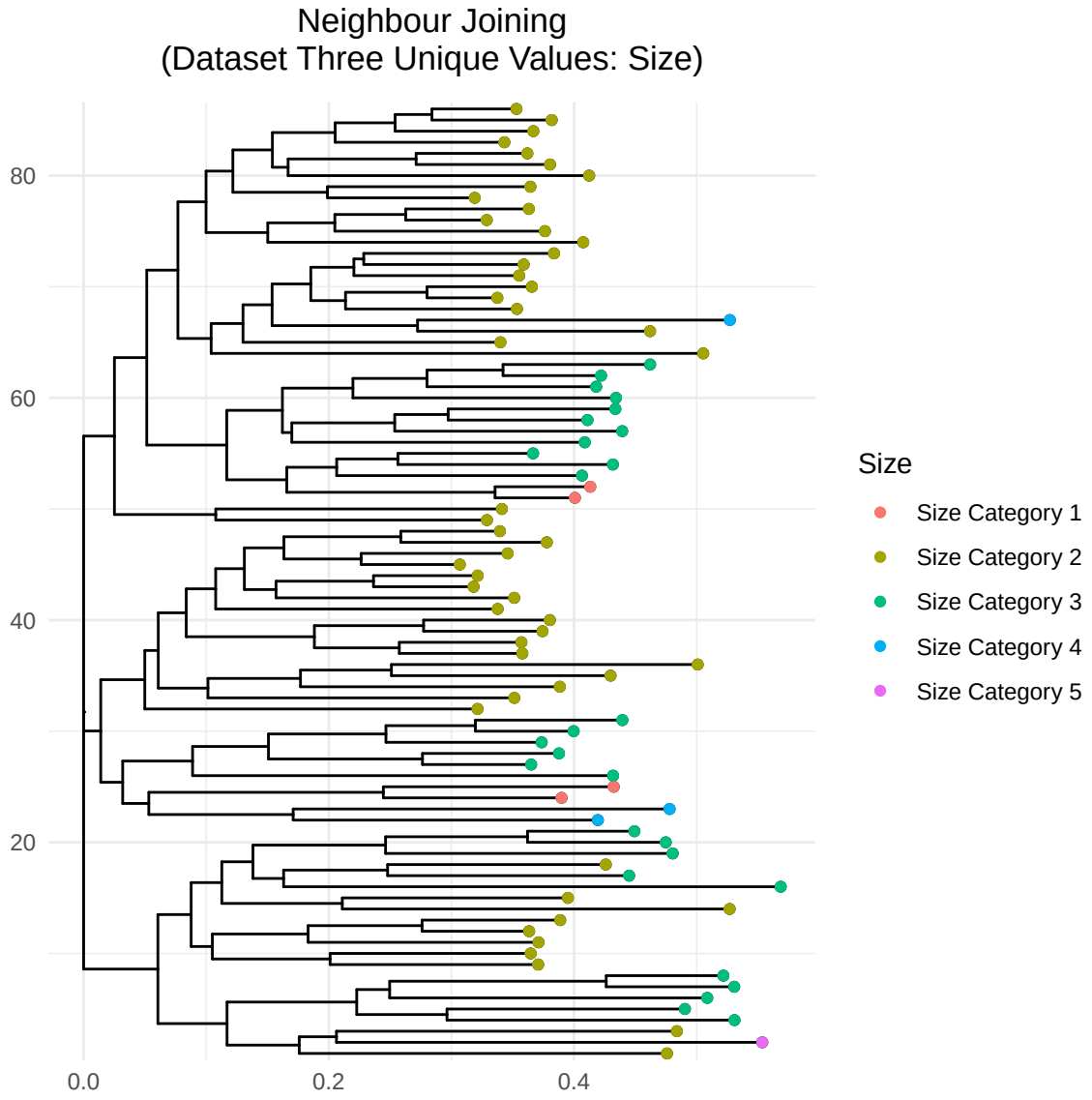

```
g2 %<+% dataset_three_unique_meta + geom_tippoint(aes(colour = shape)) +
    labs(colour = "Shape") + ggtitle("Neighbour Joining\n(Dataset Three Unique Values: Shape)") +
    scale_colour_discrete(name = "Shape", labels = c("Asymmetric",
        "Oval", "Round")) + theme_minimal() + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



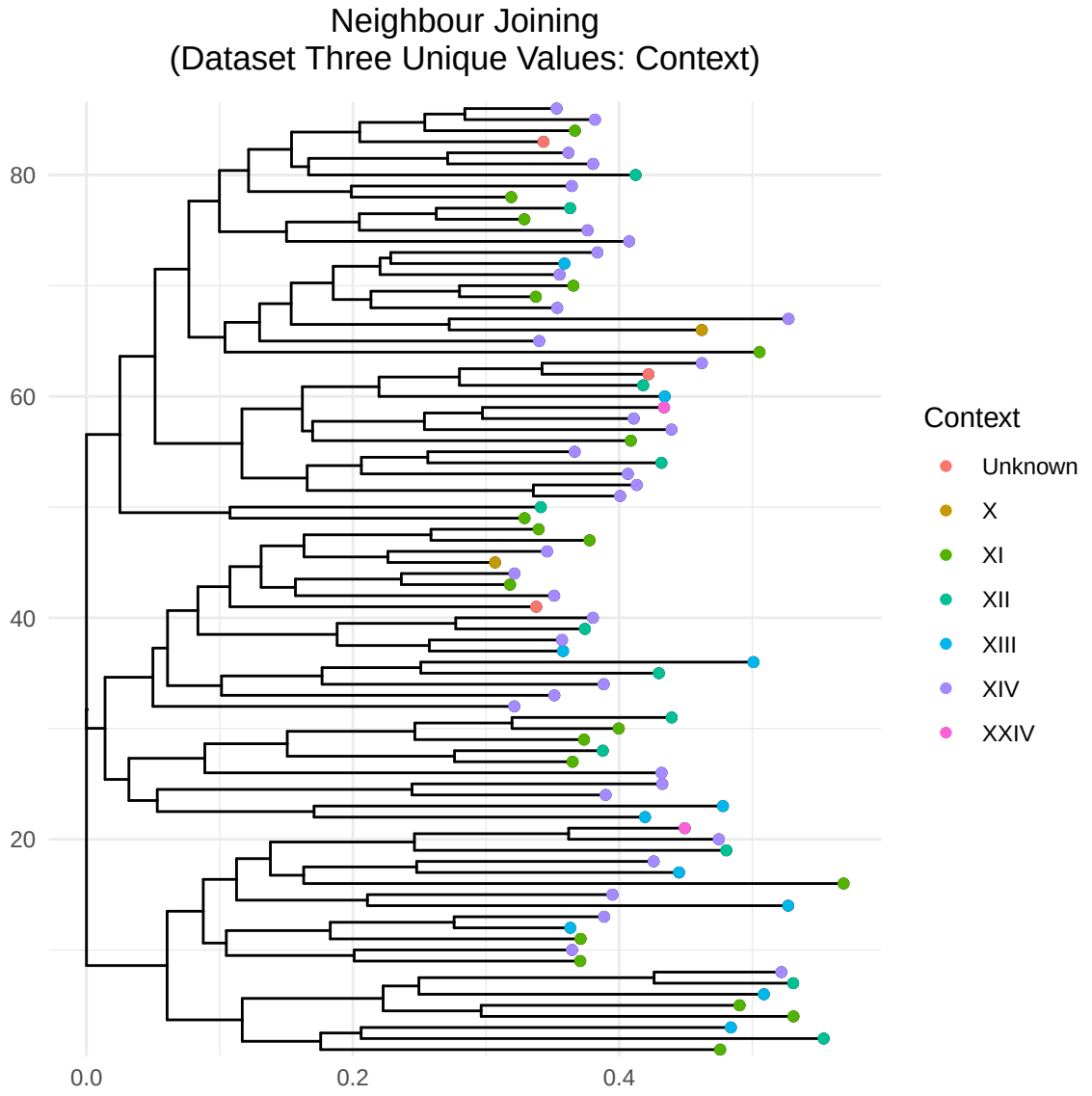

```
g2 %<+% dataset_three_unique_meta + geom_tippoint(aes(colour = material)) +
    labs(colour = "Material") + ggtitle("Neighbour Joining\n(Dataset Three Unique Values: Material)") +
    scale_colour_discrete(name = "Material", labels = c("Sandstone",
        "Shale", "Water-Rounded Pebble")) + theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```




```
g2 %<+% dataset_three_unique_meta + geom_tippoint(aes(colour = size)) +
    labs(colour = "Size") + ggtitle("Neighbour Joining\n(Dataset Three Unique Values: Size)") +
    scale_colour_discrete(name = "Size", labels = c("Size Category 1",
        "Size Category 2", "Size Category 3", "Size Category 4",
        "Size Category 5")) + theme_minimal() + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```




```
g2 %<+% dataset_three_unique_meta + geom_tippoint(aes(colour = gcontext)) +
    labs(colour = "Context") + ggtitle("Neighbour Joining\n(Dataset Three Unique Values: Context)") +
    scale_colour_discrete(name = "Context", labels = c("Unknown",
        "X", "XI", "XII", "XIII", "XIV", "XXIV")) + theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```



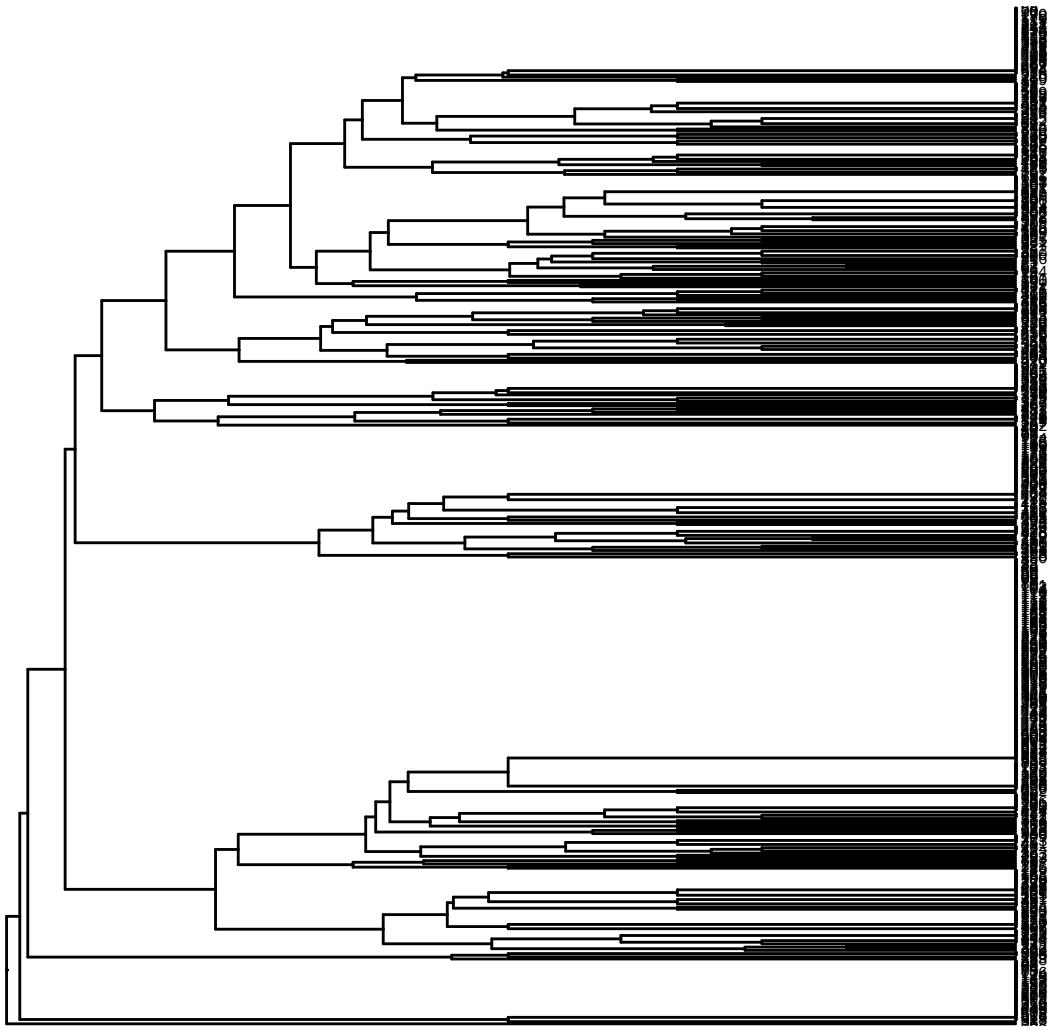
Unweighted Pair Group Method with Arithmetic Mean (UPGMA)

Using the same distance matrices above, we can use the `phangorn::upgma()` function to generate the estimated trees.

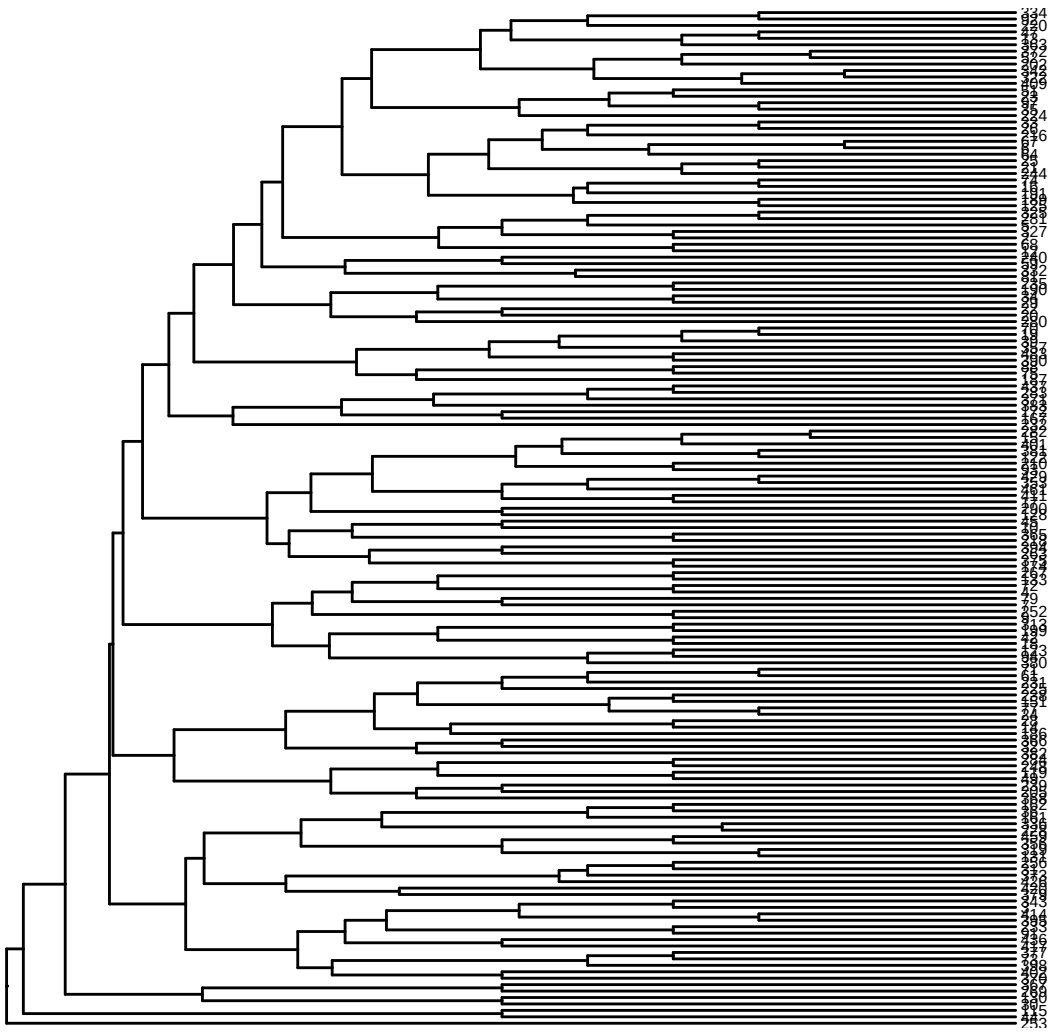
```
treeupgmaone <- upgma(distmatrixone)
treeupgmaoneu <- upgma(distmatrixoneunique)
treeupgmatwo <- upgma(distmatrixtwo)
treeupgmatwou <- upgma(distmatrixtwounique)
treeupgmathree <- upgma(distmatrixthree)
treeupgmathreeu <- upgma(distmatrixthreeunique)
```

The visualisations in `ggtree` are used again to visualise these new trees.

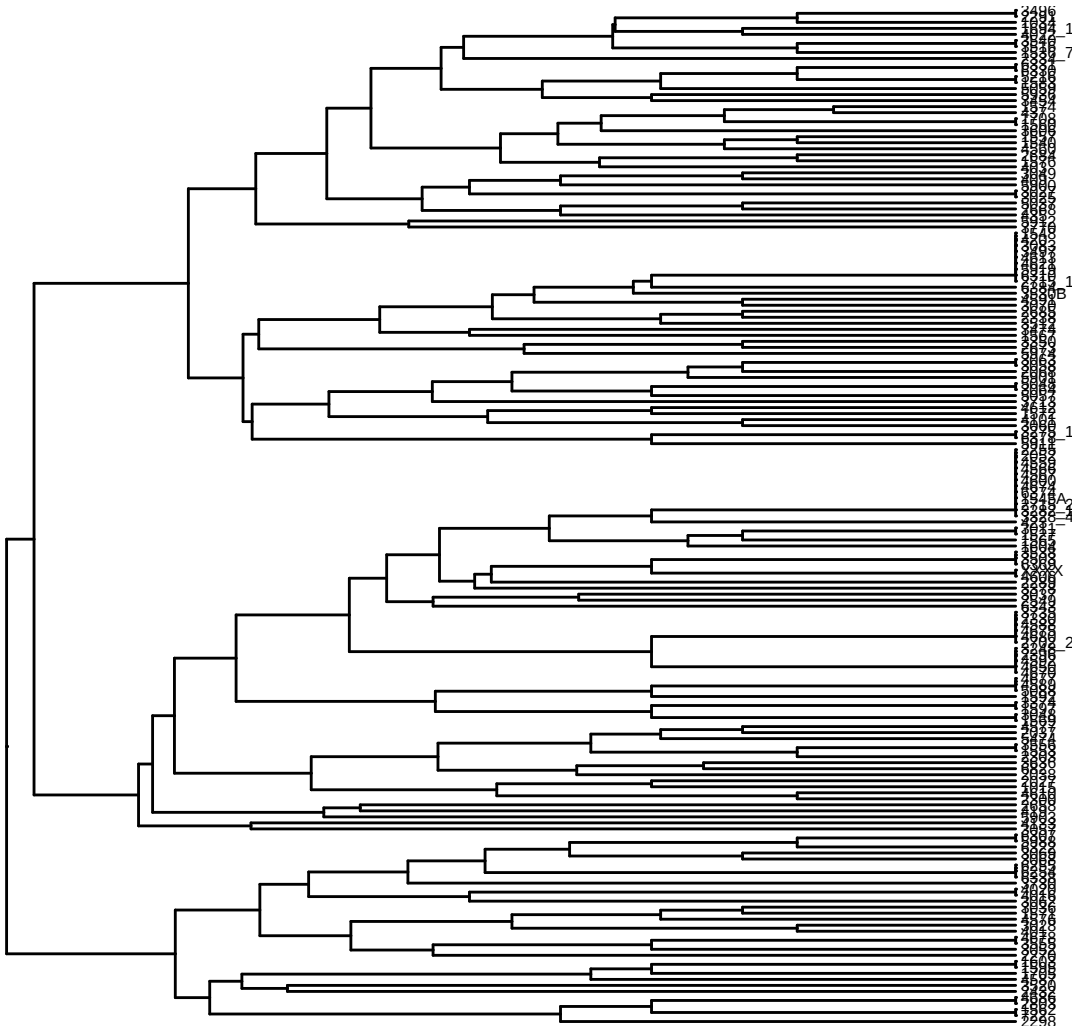
```
ggtree(treeupgmaone, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("UPGMA\n(Dataset One: All Values)") + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



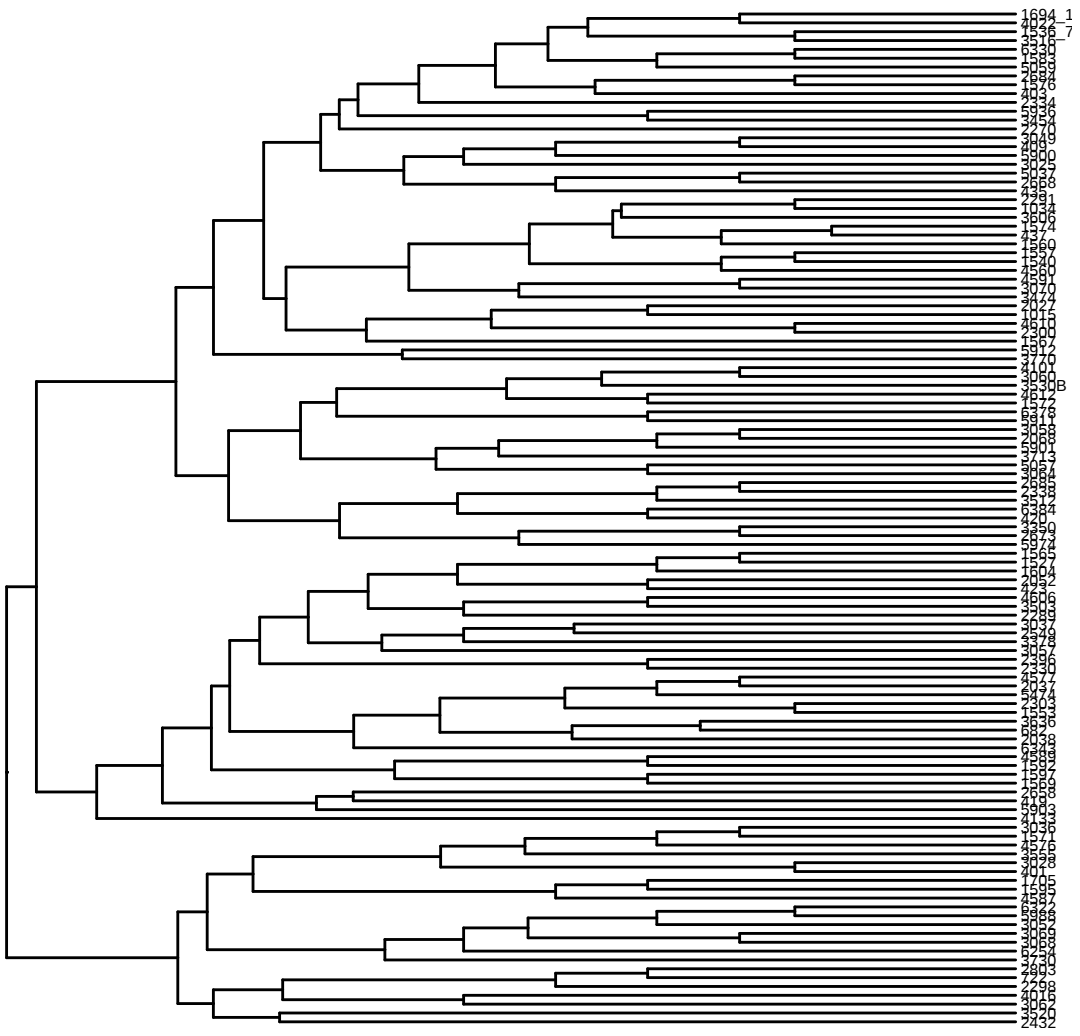

```
ggtree(treeupgmaoneu, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("UPGMA\n(Dataset One: Unique Values)") + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



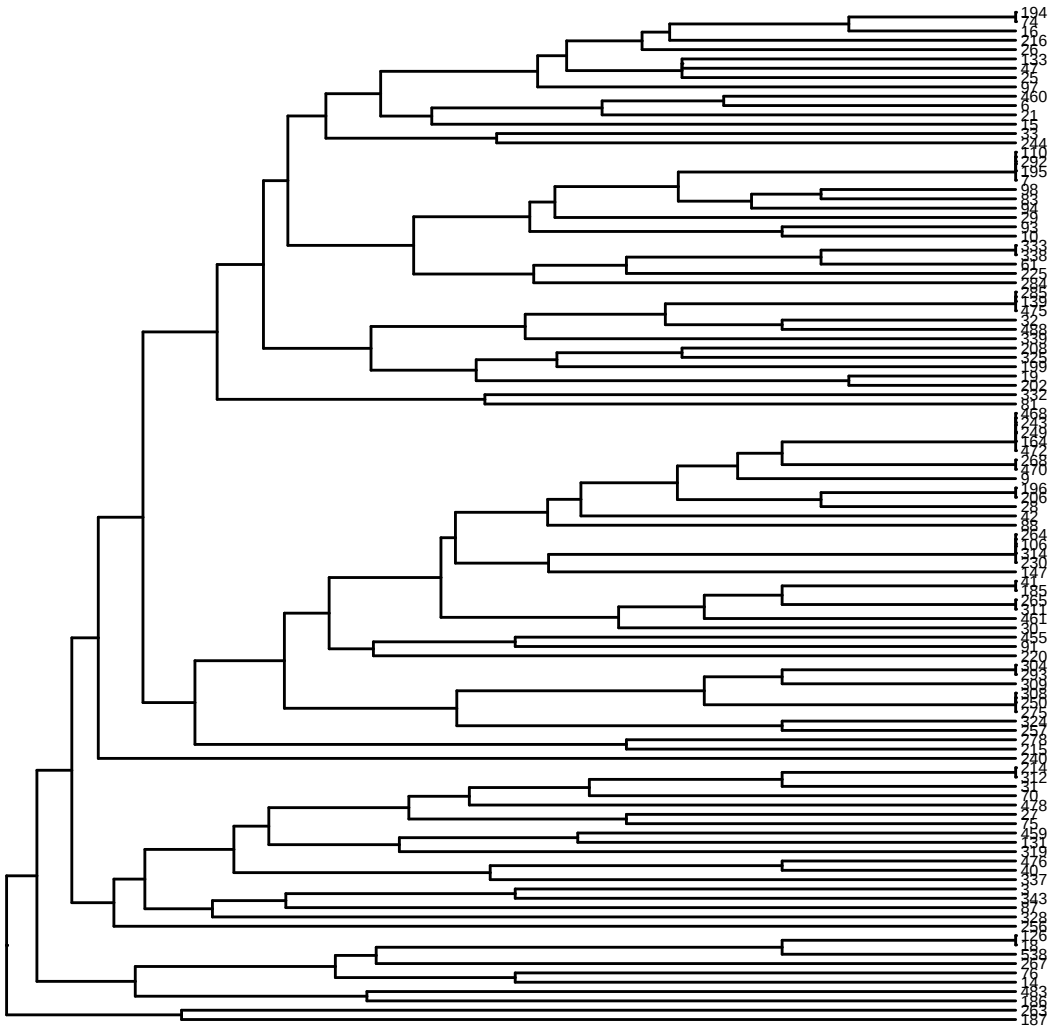

```
ggtree(treeupgmatwo, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("UPGMA\n(Dataset Two: All Values)") + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



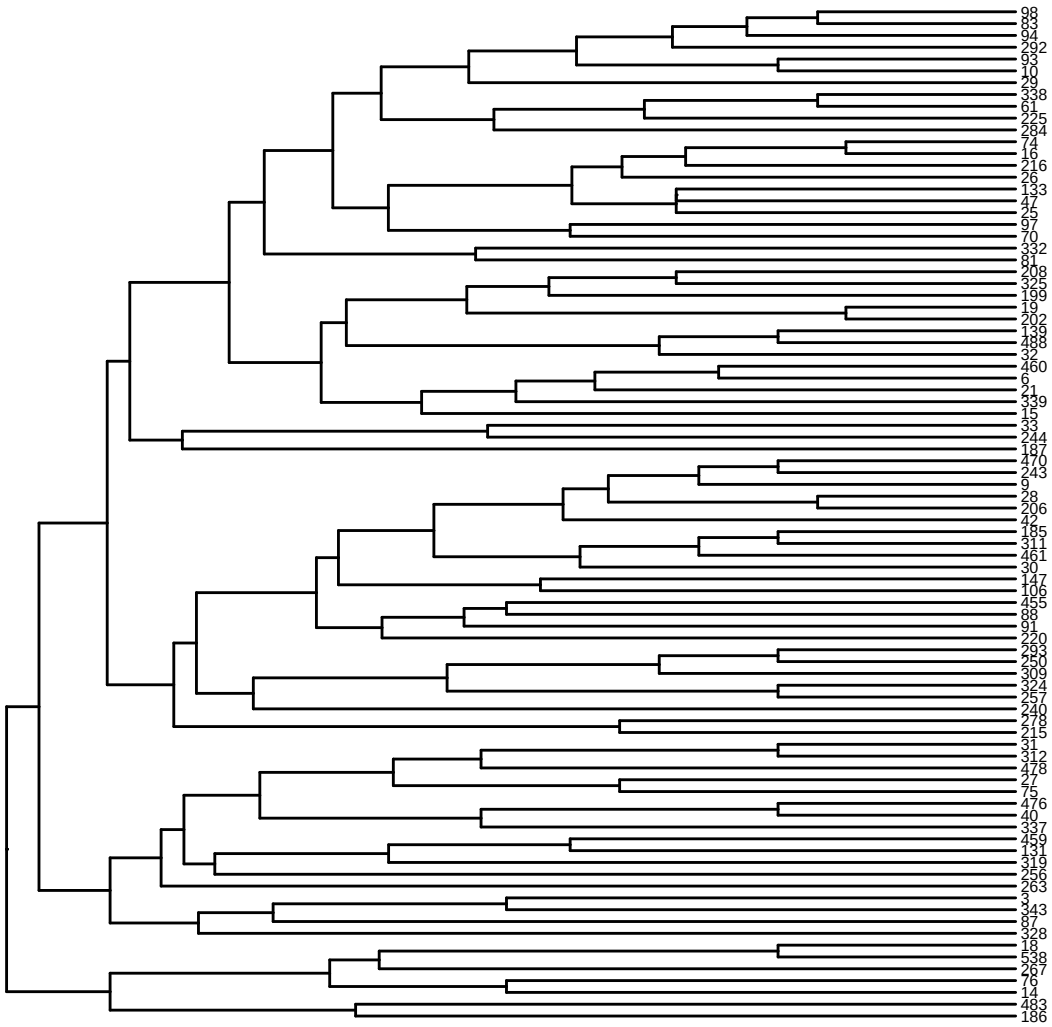

```
ggtree(treeupgmatwou, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("UPGMA\n(Dataset Two: Unique Values)") + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



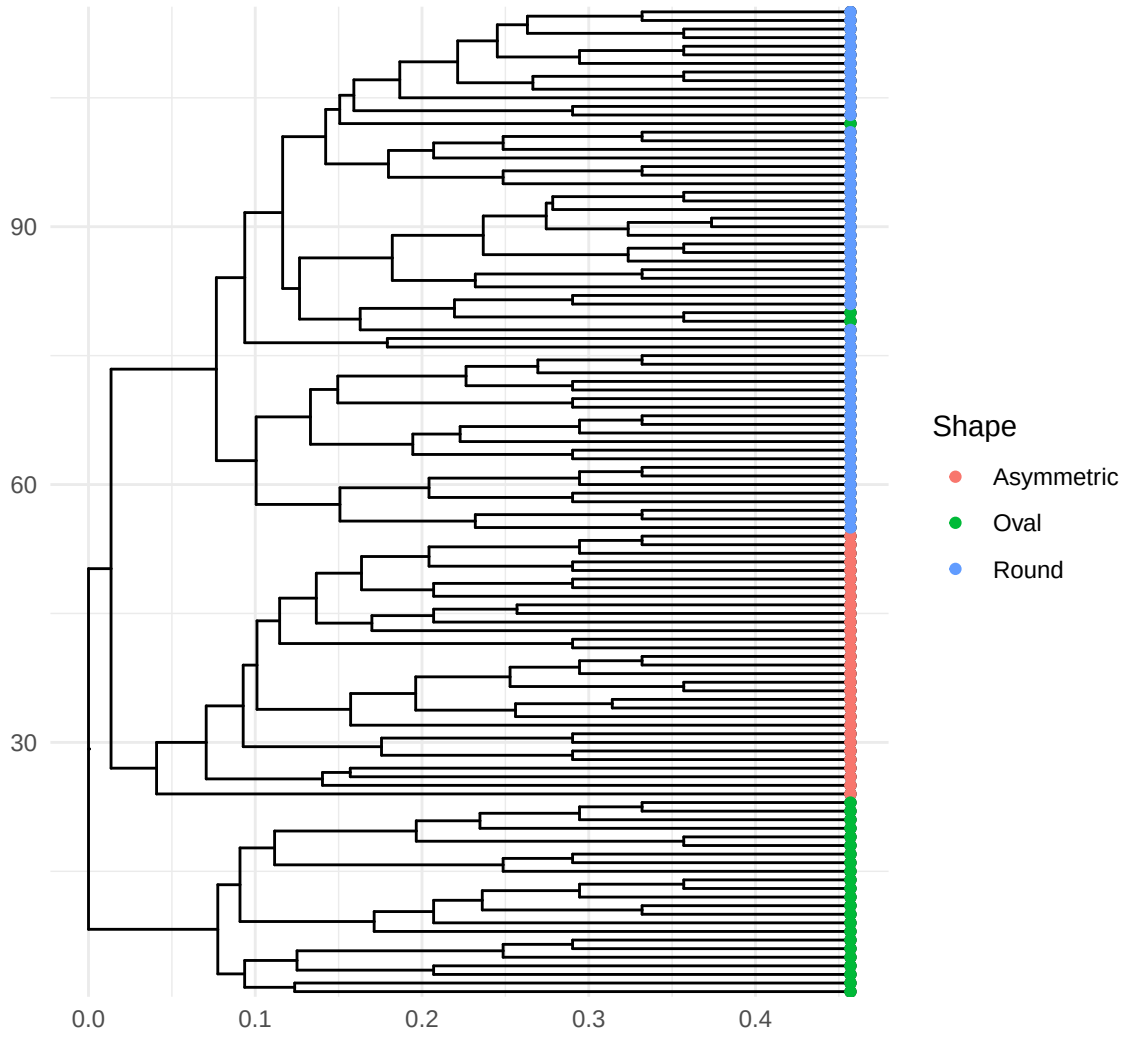

```
ggtree(treeupgmathree, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("UPGMA\n(Dataset Three: All Values)") + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



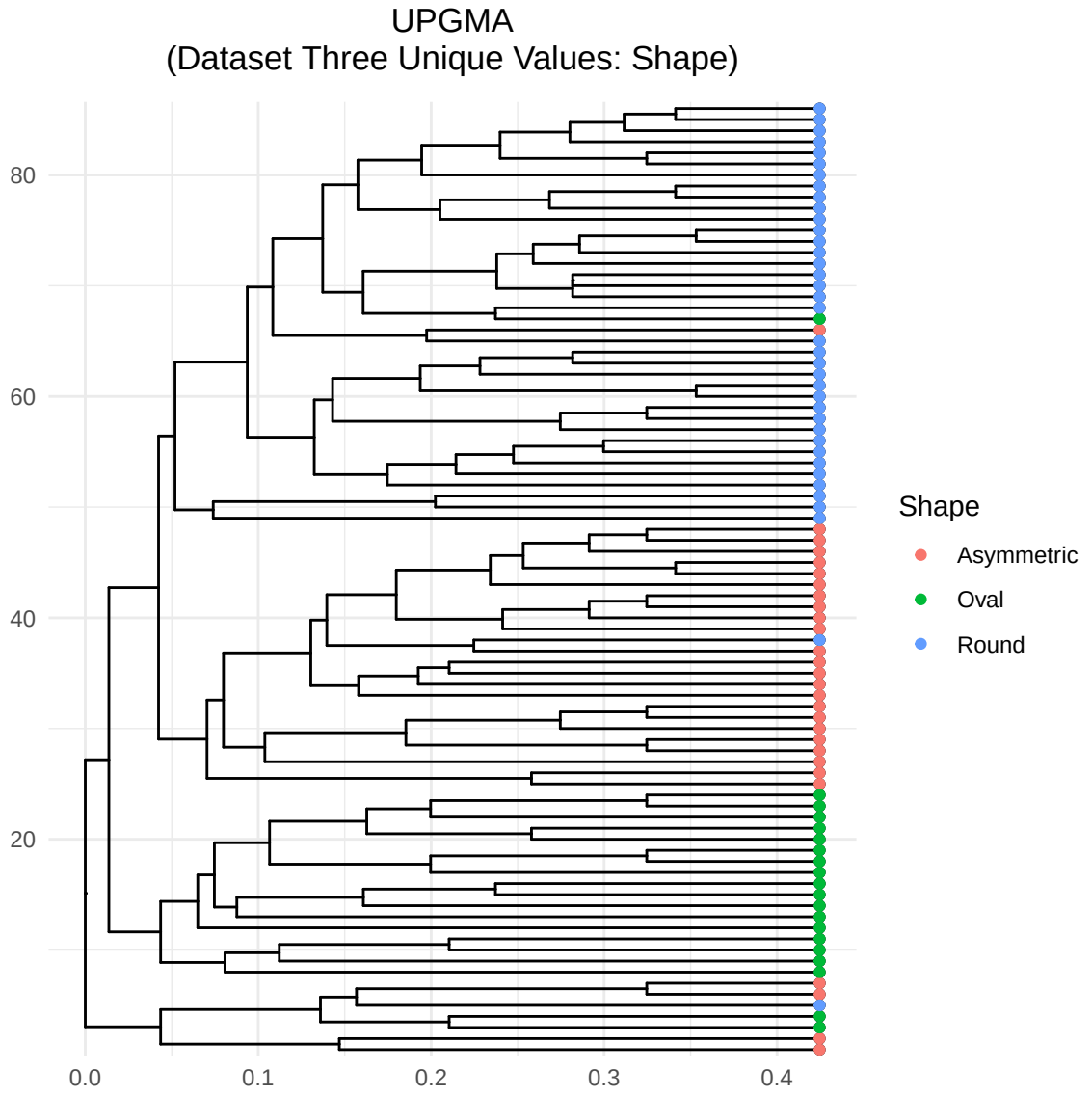

```
ggtree(treeupgmathreeu, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("UPGMA\n(Dataset Three: Unique Values)") + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



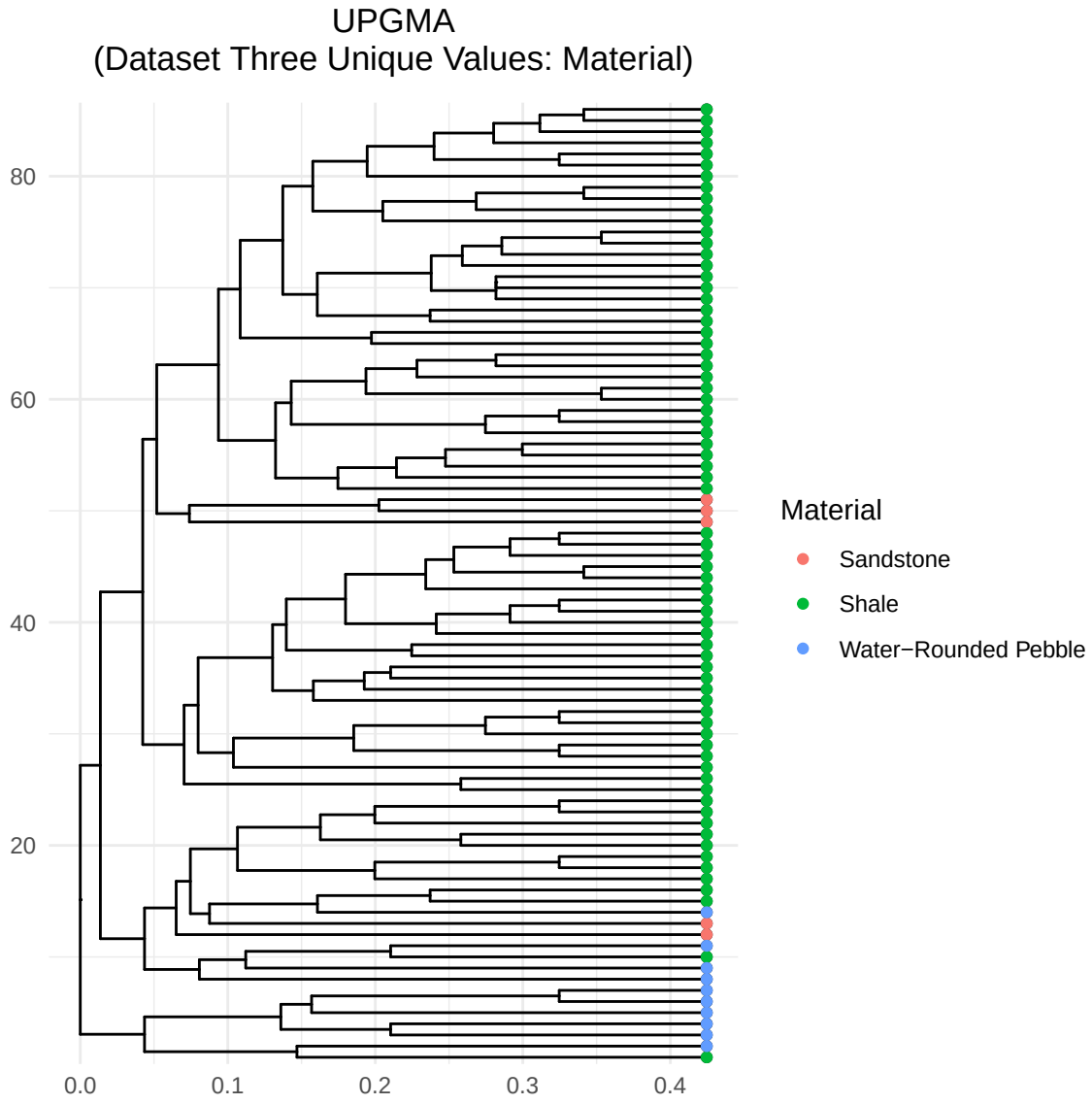

```
g3 <- ggtree(treeupgmatwou, layout = "rectangular") + geom_tippoint(size = 1.5)

g4 <- ggtree(treeupgmathreeu, layout = "rectangular") + geom_tippoint(size = 1.5)

g3 %<+% dataset_two_unique_meta + geom_tippoint(aes(colour = shape)) +
    labs(colour = "Shape") + ggtitle("UPGMA\n(Dataset Two Unique Values: Shape)") +
    scale_colour_discrete(name = "Shape", labels = c("Asymmetric",
        "Oval", "Round")) + theme_minimal() + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



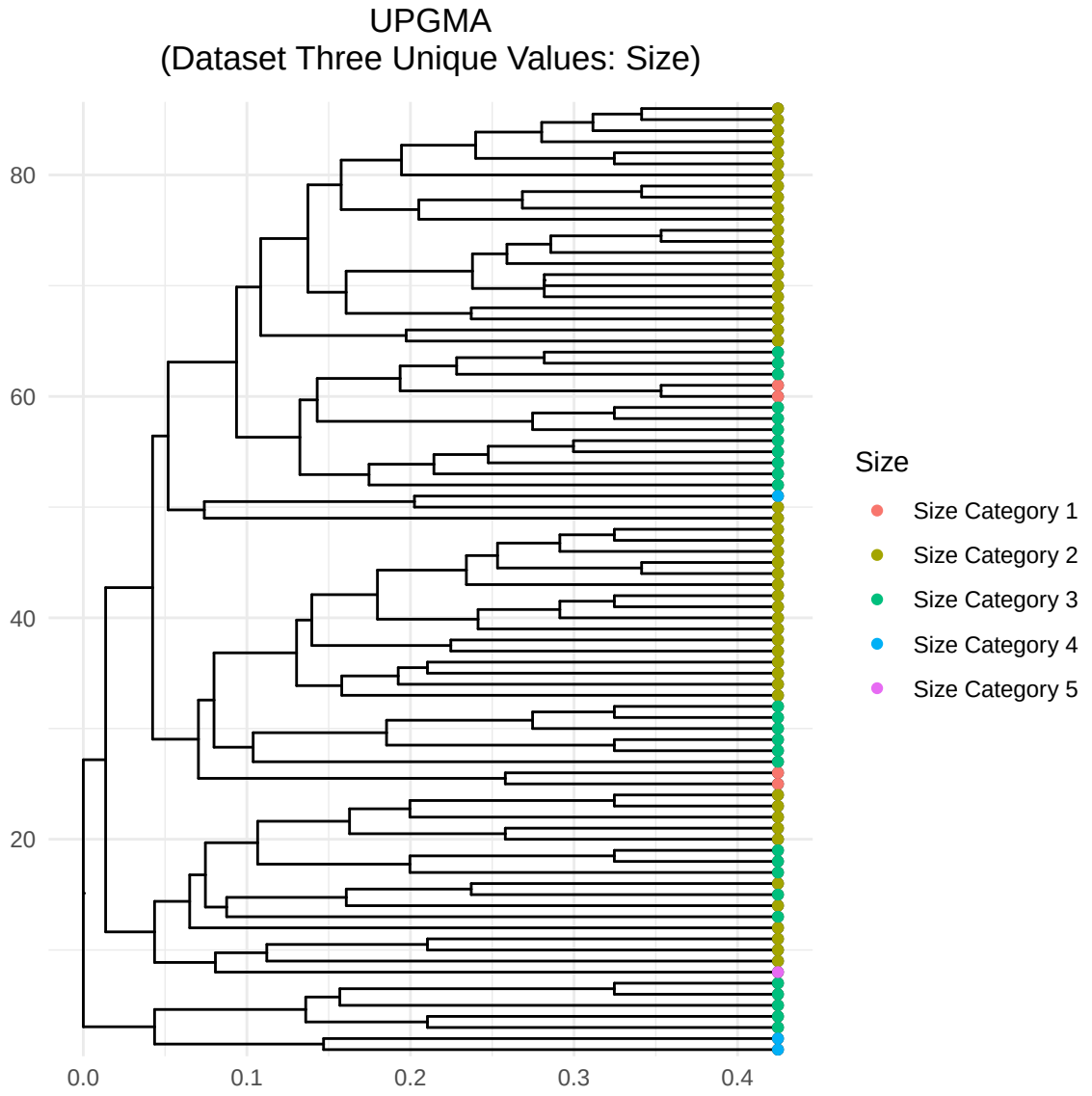

```
g4 %<+% dataset_three_unique_meta + geom_tippoint(aes(colour = shape)) +
    labs(colour = "Shape") + ggtitle("UPGMA\n(Dataset Three Unique Values: Shape)") +
    scale_colour_discrete(name = "Shape", labels = c("Asymmetric",
        "Oval", "Round")) + theme_minimal() + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



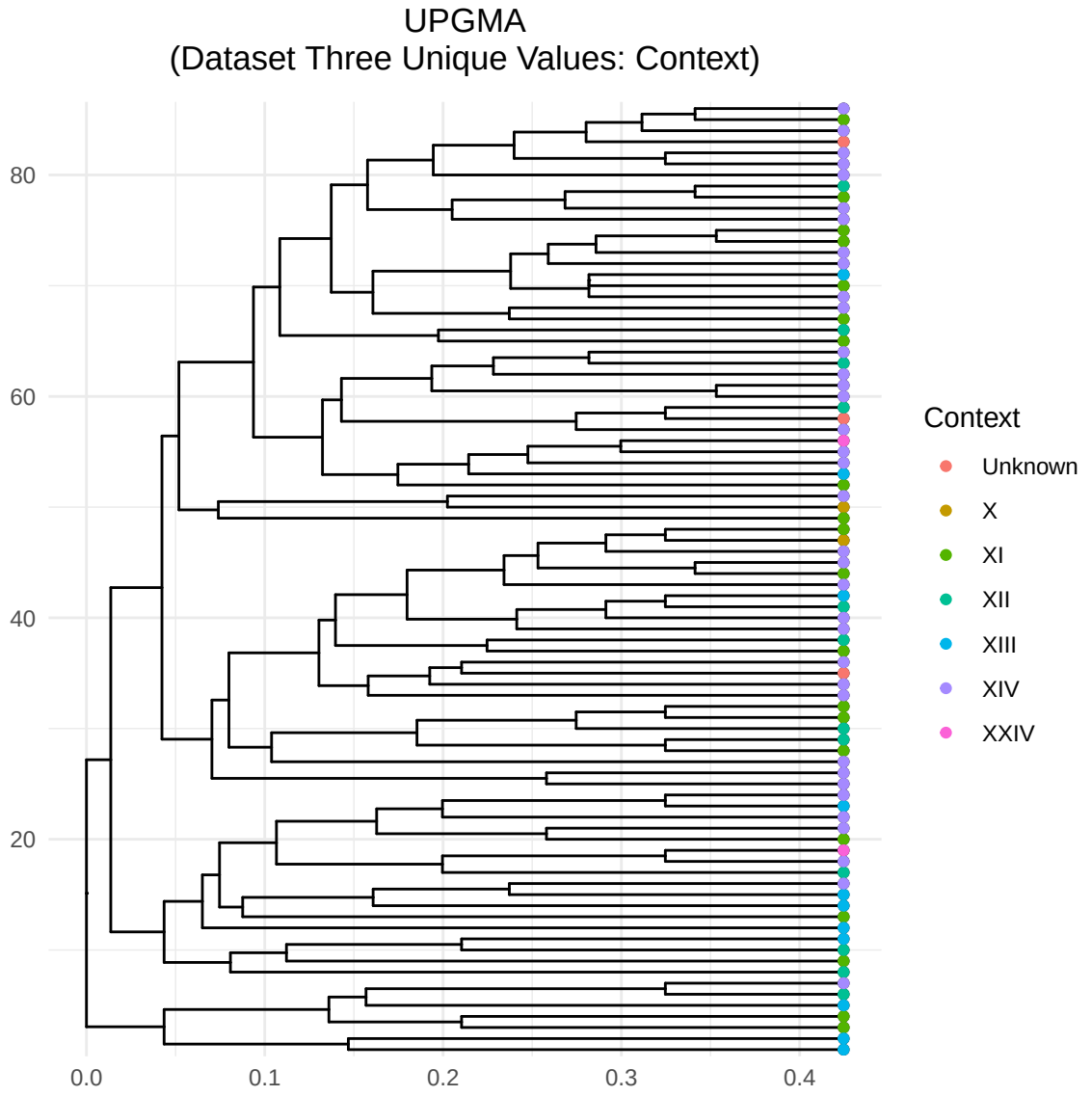

```
g4 %<+% dataset_three_unique_meta + geom_tippoint(aes(colour = material)) +
    labs(colour = "Material") + ggtitle("UPGMA\n(Dataset Three Unique Values: Material)") +
    scale_colour_discrete(name = "Material", labels = c("Sandstone",
        "Shale", "Water-Rounded Pebble")) + theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```




```
g4 %<+% dataset_three_unique_meta + geom_tippoint(aes(colour = size)) +
    labs(colour = "Size") + ggtitle("UPGMA\n(Dataset Three Unique Values: Size)") +
    scale_colour_discrete(name = "Size", labels = c("Size Category 1",
        "Size Category 2", "Size Category 3", "Size Category 4",
        "Size Category 5")) + theme_minimal() + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```




```
g4 %<+% dataset_three_unique_meta + geom_tippoint(aes(colour = gcontext)) +
    labs(colour = "Context") + ggtitle("UPGMA\n(Dataset Three Unique Values: Context)") +
    scale_colour_discrete(name = "Context", labels = c("Unknown",
        "X", "XI", "XII", "XIII", "XIV", "XXIV")) + theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```



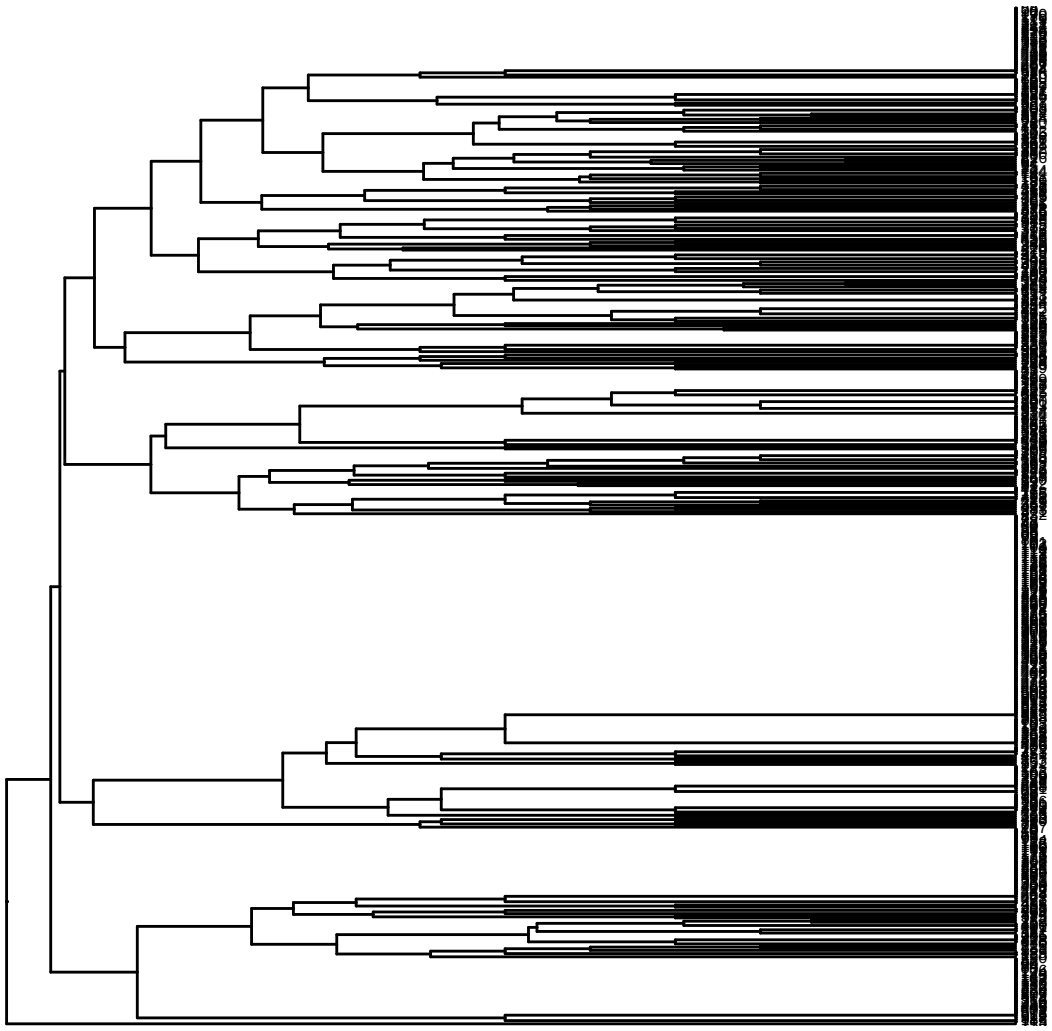
Weighted Pair Group Method with Arithmetic Mean (WPGMA)

As above, we can use the functions in `phangorm`, specifically `phangorn::upgma()` to generate the estimated trees.

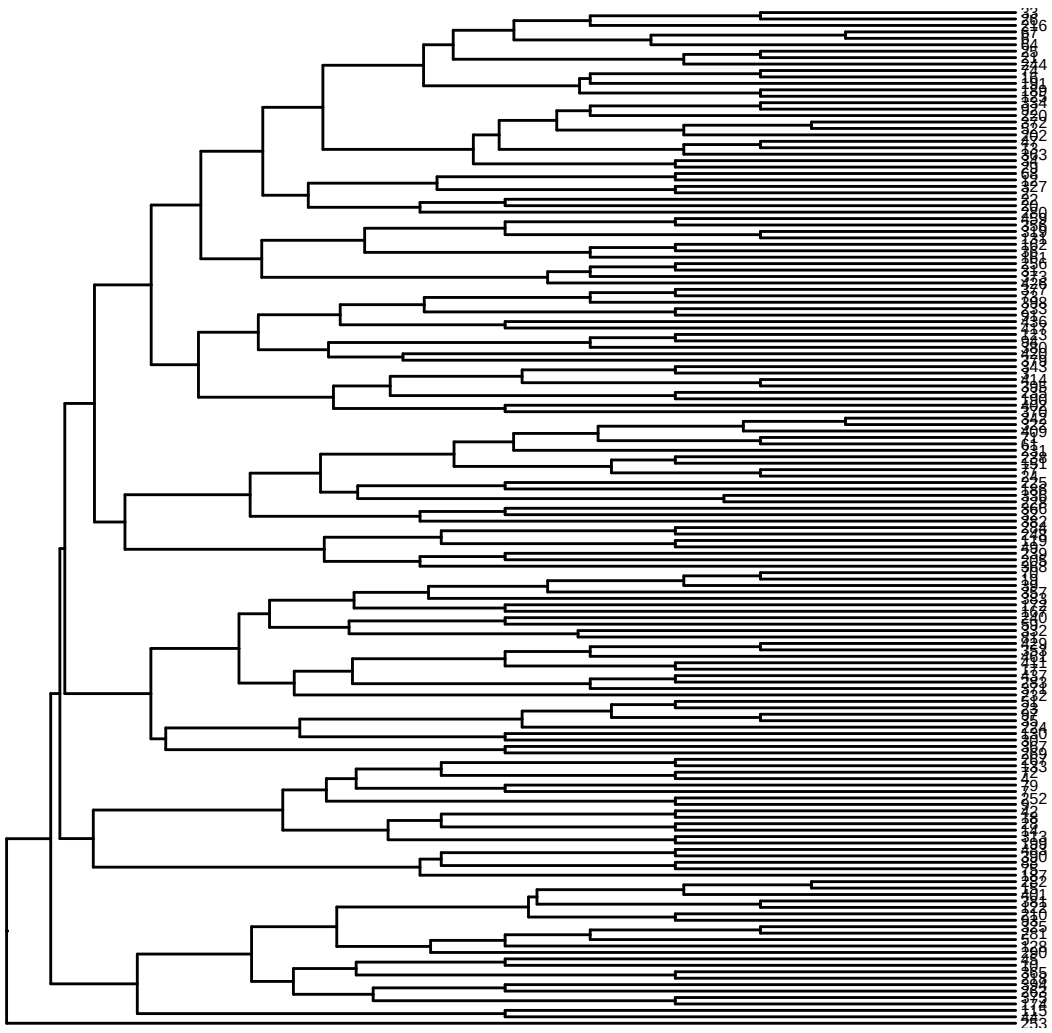
```
treewpgmaone <- wpgma(distmatrixone)
treewpgmaoneu <- wpgma(distmatrixoneunique)
treewpgmatwo <- wpgma(distmatrixtwo)
treewpgmatwou <- wpgma(distmatrixtwounique)
treewpgmathree <- wpgma(distmatrixthree)
treewpgmathreeu <- wpgma(distmatrixthreeunique)
```

The visualisation tools in `ggtree` are used again to visualise these new trees.

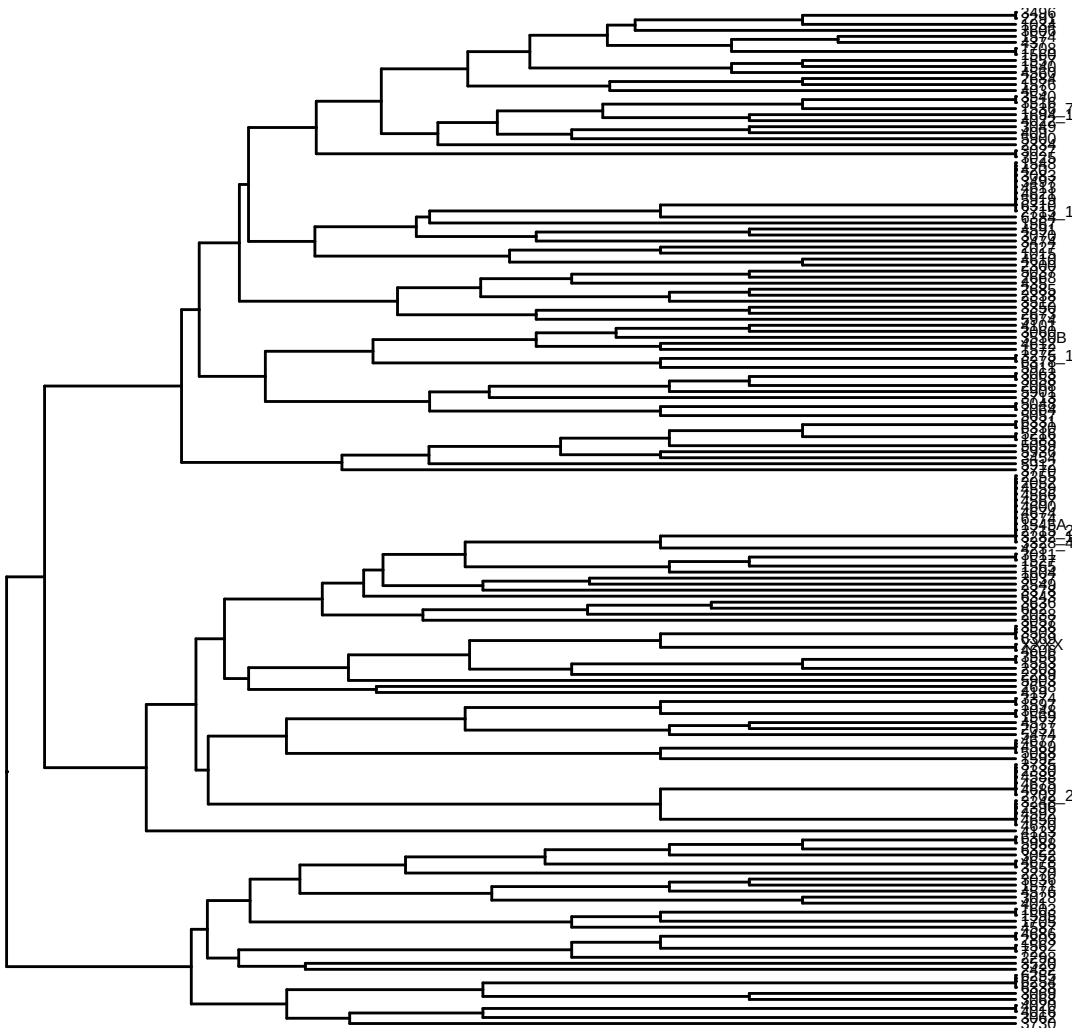
```
ggtree(treewpgmaone, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("WPGMA\n(Dataset One: All Values)") + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



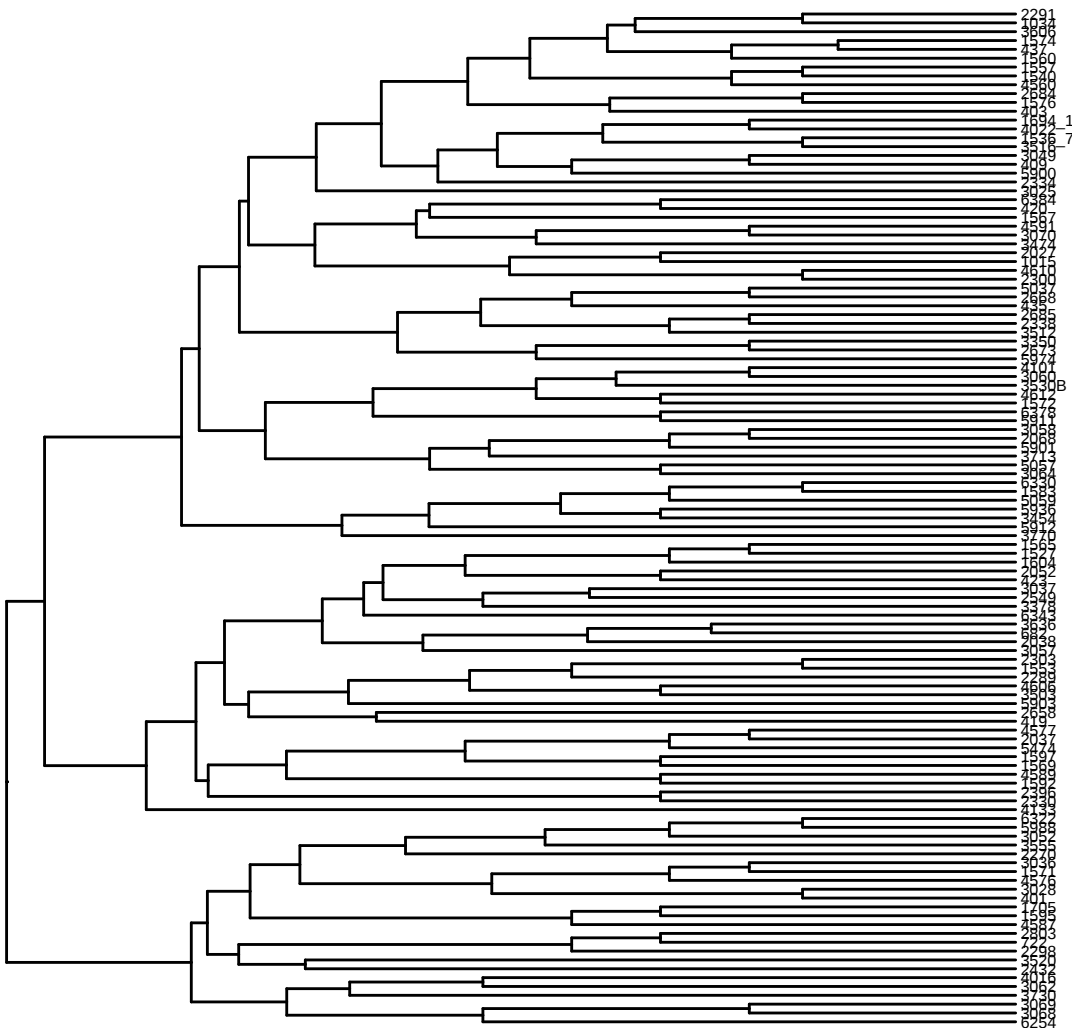

```
ggtree(treewpgmaoneu, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("WPGMA\n(Dataset One: Unique Values)") + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



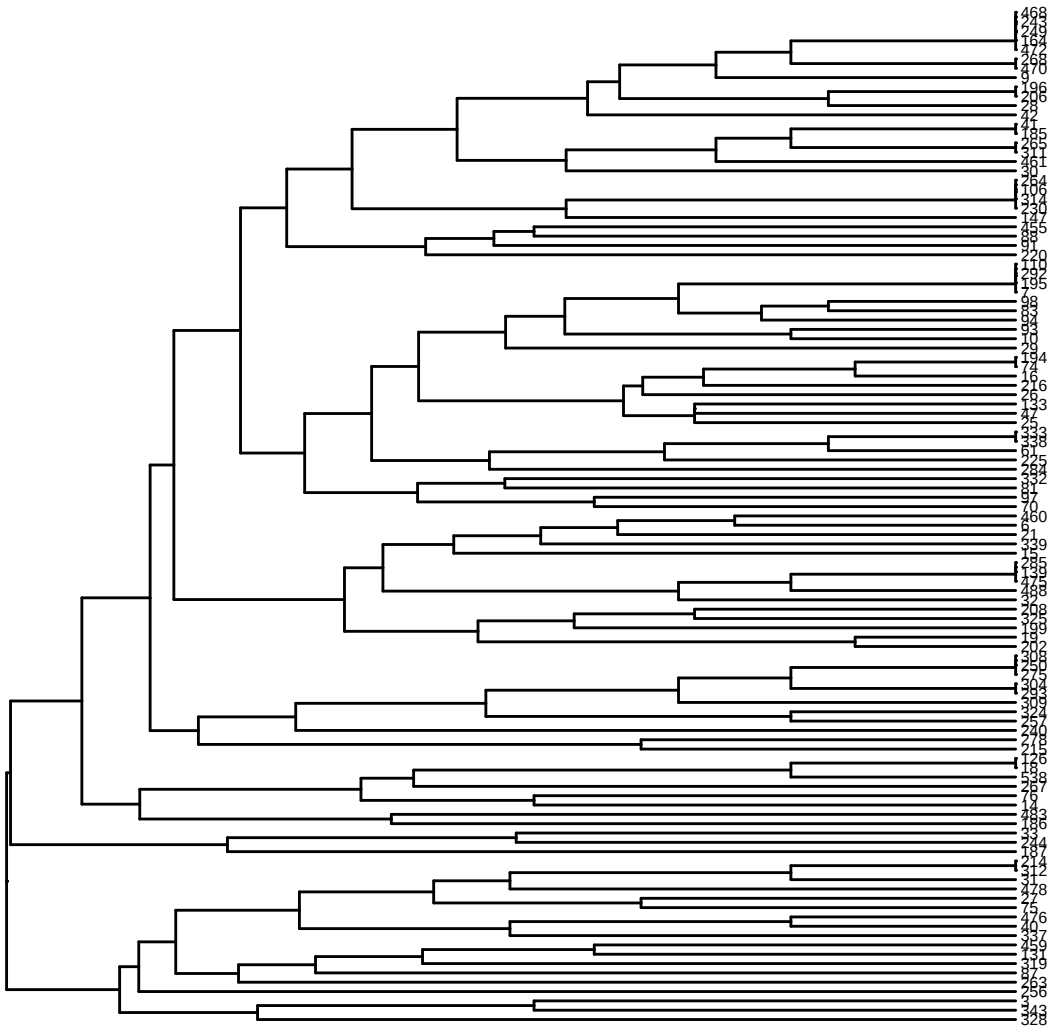

```
ggtree(treewpgmatwo, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("WPGMA\n(Dataset Two: All Values)") + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



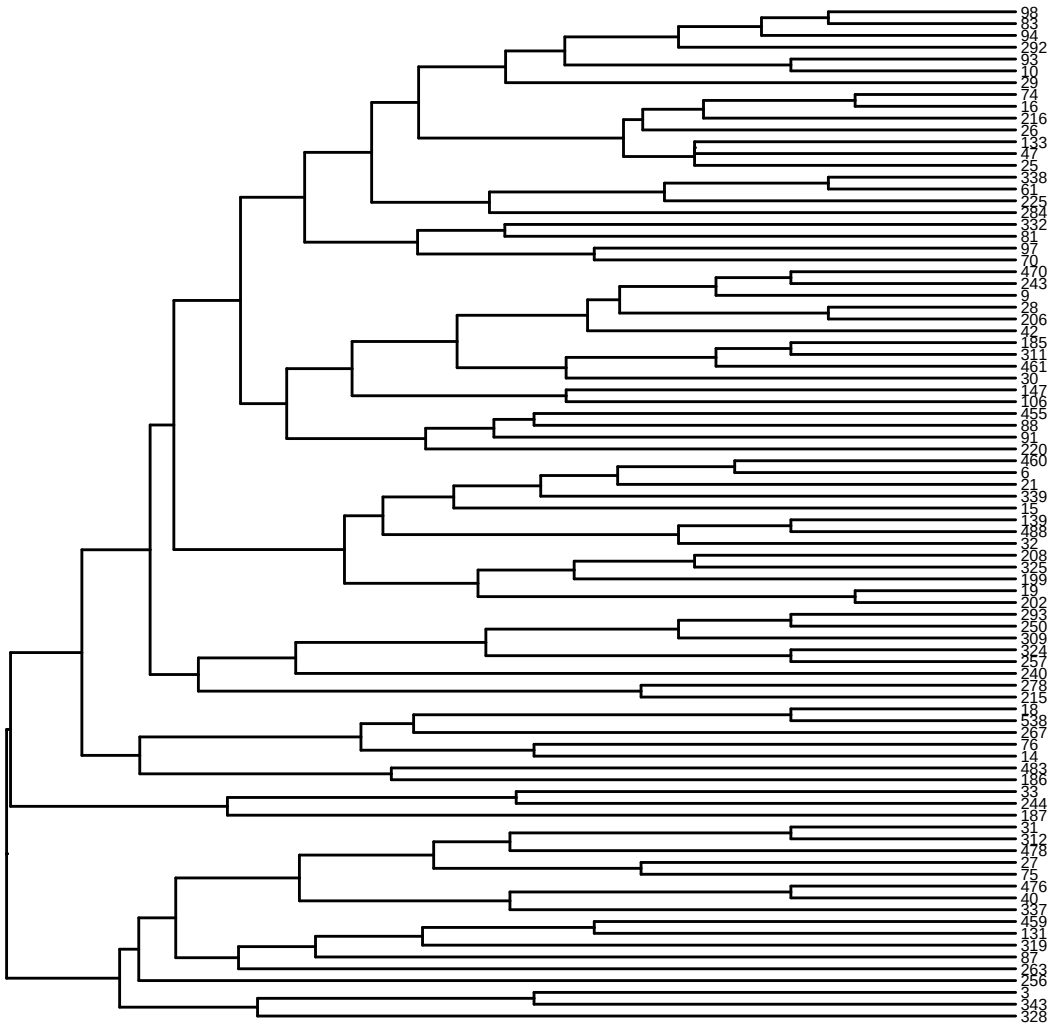

```
ggtree(treewpgmatwou, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("WPGMA\n(Dataset Two: Unique Values)") + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



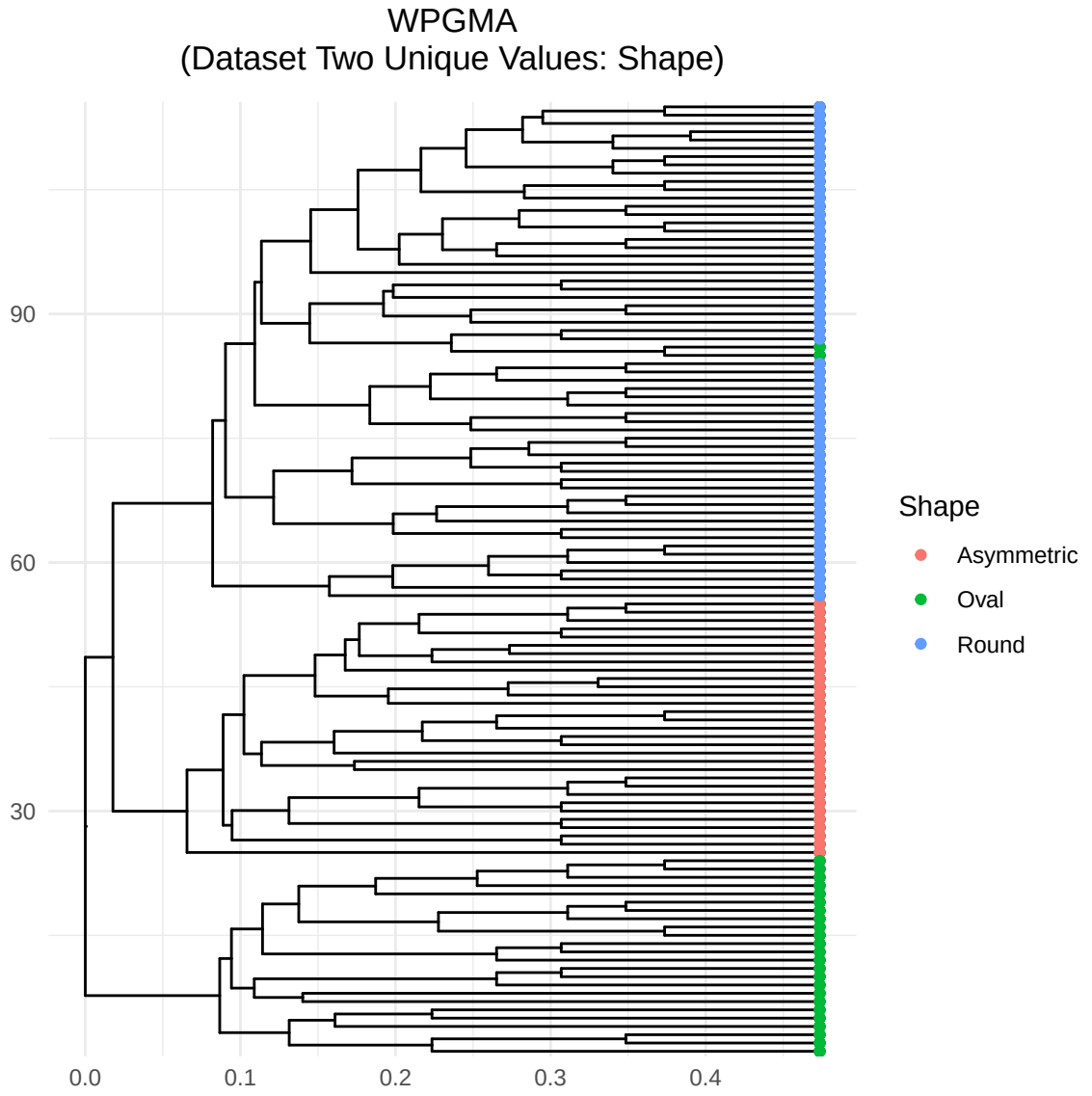

```
ggtree(treewpgmathree, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("WPGMA\n(Dataset Three: All Values)") + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



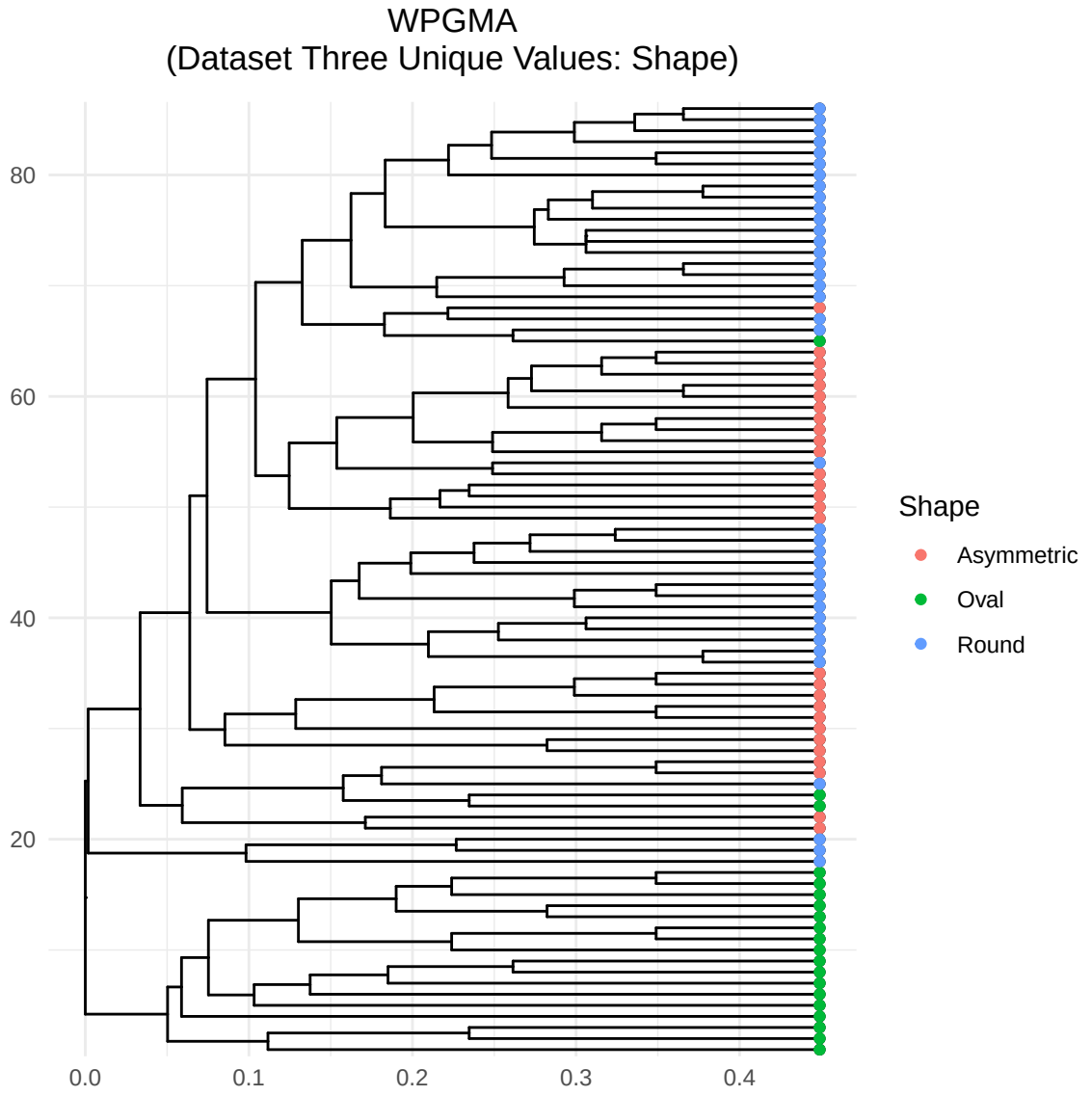

```
ggtree(treewpgmathreeu, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("WPGMA\n(Dataset Three: Unique Values)") + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



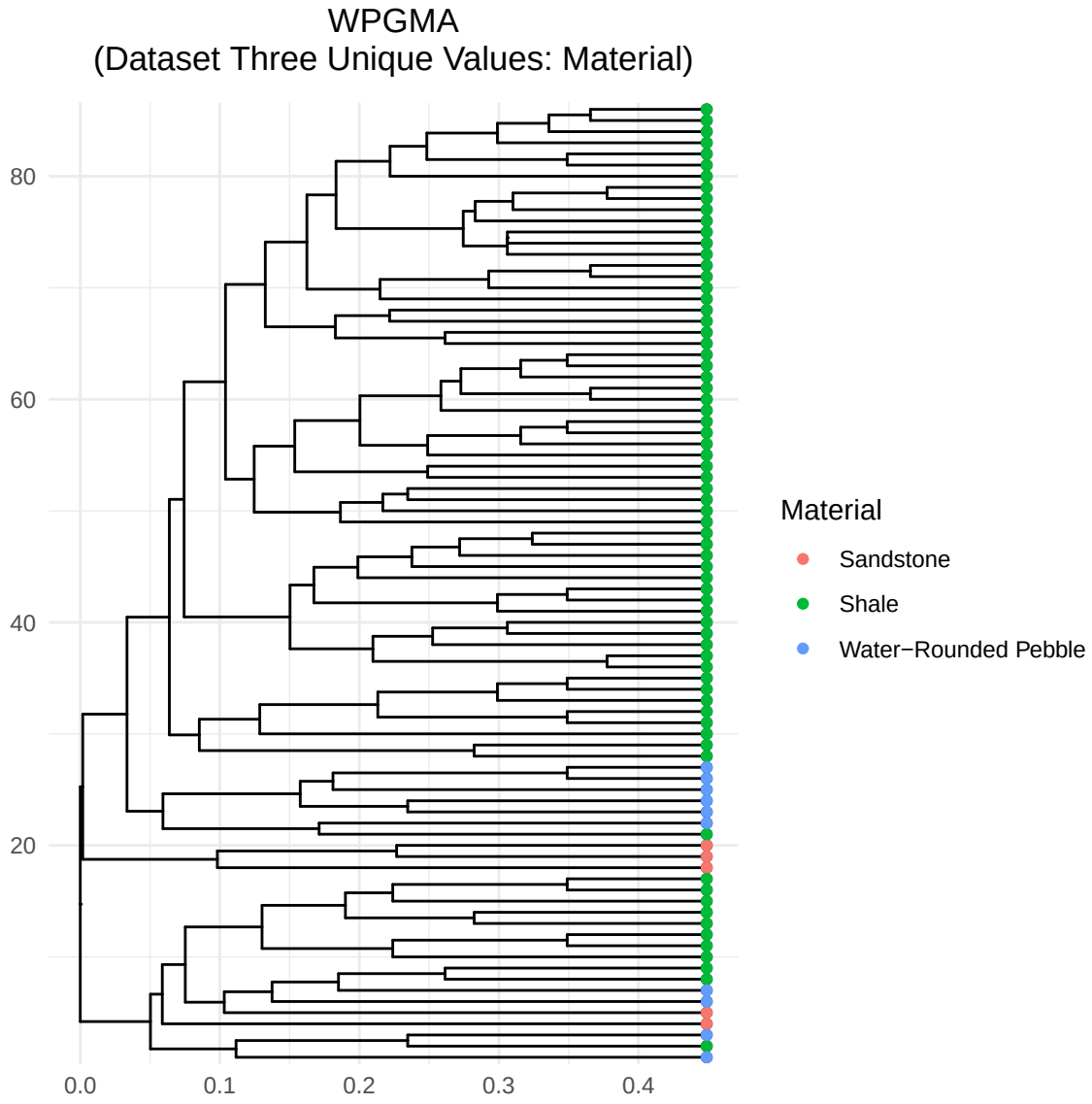

```
g5 <- ggtree(treewpgmatwou, layout = "rectangular") + geom_tippoint(size = 1.5)

g6 <- ggtree(treewpgmathreeu, layout = "rectangular") + geom_tippoint(size = 1.5)

g5 %<+% dataset_two_unique_meta + geom_tippoint(aes(colour = shape)) +
    labs(colour = "Shape") + ggtitle("WPGMA\n(Dataset Two Unique Values: Shape)") +
    scale_colour_discrete(name = "Shape", labels = c("Asymmetric",
        "Oval", "Round")) + theme_minimal() + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



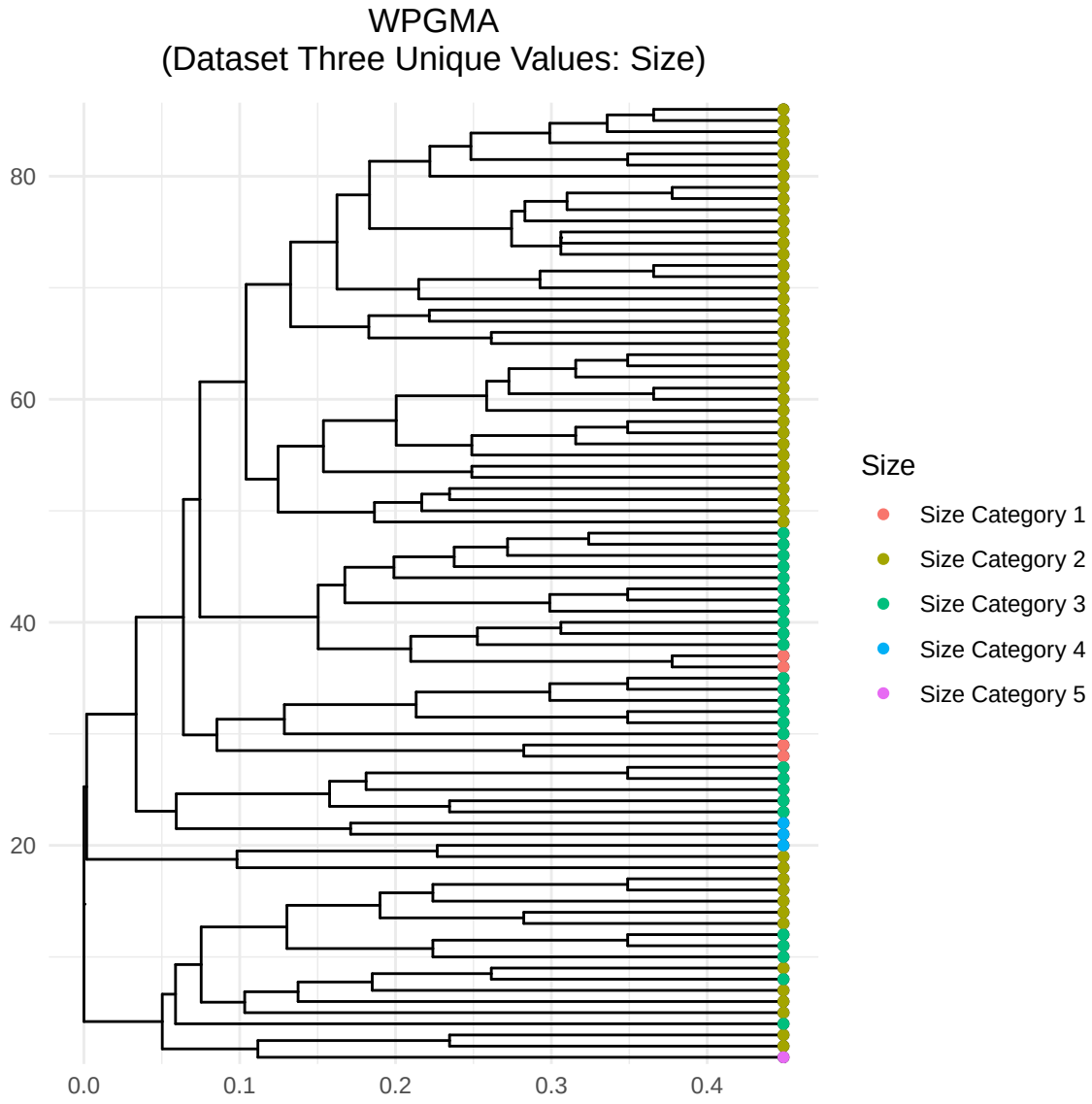

```
g6 %<+% dataset_three_unique_meta + geom_tippoint(aes(colour = shape)) +
    labs(colour = "Shape") + ggtitle("WPGMA\n(Dataset Three Unique Values: Shape)") +
    scale_colour_discrete(name = "Shape", labels = c("Asymmetric",
        "Oval", "Round")) + theme_minimal() + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



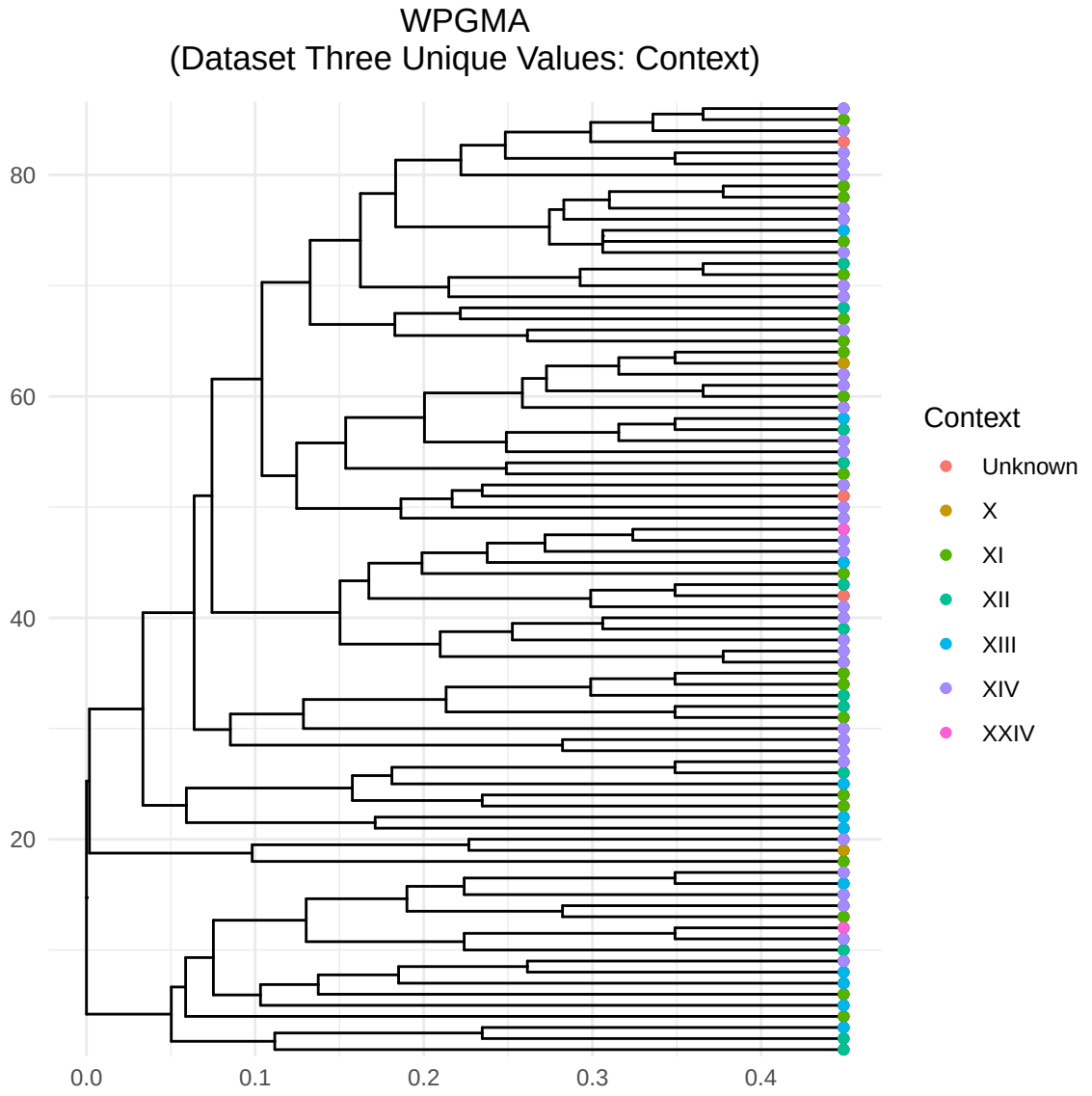

```
g6 %<+% dataset_three_unique_meta + geom_tippoint(aes(colour = material)) +
    labs(colour = "Material") + ggtitle("WPGMA\n(Dataset Three Unique Values: Material)") +
    scale_colour_discrete(name = "Material", labels = c("Sandstone",
        "Shale", "Water-Rounded Pebble")) + theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```




```
g6 %<+% dataset_three_unique_meta + geom_tippoint(aes(colour = size)) +
    labs(colour = "Size") + ggtitle("WPGMA\n(Dataset Three Unique Values: Size)") +
    scale_colour_discrete(name = "Size", labels = c("Size Category 1",
        "Size Category 2", "Size Category 3", "Size Category 4",
        "Size Category 5")) + theme_minimal() + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```




```
g6 %<+% dataset_three_unique_meta + geom_tippoint(aes(colour = gcontext)) +
    labs(colour = "Context") + ggtitle("WPGMA\n(Dataset Three Unique Values: Context)") +
    scale_colour_discrete(name = "Context", labels = c("Unknown",
        "X", "XI", "XII", "XIII", "XIV", "XXIV")) + theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```



Parsimony-Based Methodologies

In constructing a scenario which demonstrates the possible underwritten system of information transmission across the production of the sun-stones, two different parsimony-based methodologies are here explored. In each instance, the character states are binary two-state character traits (absence or presence of an attribute).

Following the first numerical phylogenies produced by parsimony (Cavalli-Sforza and Edwards, 1965) and Edward's and Cavalli-Sforza's (1963) declaration, different methods describing different phylogenetic patterns are available. The two explored here are:

- **Wagner Parsimony** (Kluge and Farris, 1969): The most prominent and common parsimony principle. Wagner parsimony assumes an **unknown ancestral state**, with **reversible character states** e.g. state 0 to state 1 and state 1 to state 0. **Equal (or similar) transitional rates are also assumed**. Commonly exploited for two-state character traits (multistate scenarios are also possible), and is most naturally modeled via continuous-time Markov chain (CTMC) transitions (Suchard et al. 2001).
- **Dollo Parsimony** (Le Quesne, 1974; Farris, 1977): This scenario assumes that **derived characters**

(sun-stone patterns) can appear only once within a tree but can be lost multiple times, unlike Wagner parsimony which allows reversibility from state 0 to state 1 . Dollo parsimony is traditionally used for binary data, however stochastic models which allow for multi-state characters are now developed (Alekseyenko et al. 2008).

Particular phylogenetic characters are going to be better modeled by one of the parsimony models, for example a highly mutable character pertains more to Wagner parsimony, whereas in scenarios where the loss of a character is a more common event e.g. Morphological characters (Gould, 1970) the Dollo parsimony principle may be more applicable. As we cannot assume the conditions in which these sun-stones were created, and the ancestral origins of the sun-stones, both methods are pursued here.

In this methodology, the two most prominent parsimony methodologies are adopted. As the ancestral states are assumed unknown, the **Camin-Sokal Parsimony** (Camin and Sokal, 1965) model is not adopted here. Similarly, as multi-state characters are not explored here then the **Fitch-Margoliash criterion** (Fitch and Margoliash, 1967) is not pursued here.

To perform the two parsimony methods, the `Rphylip::Rmix` and `Rphylip::Rdollop` functions were used. In each instance, the package default number of random addition replicates were used (n = 10).

Wagner Parsimony

In detail, the Wagner method for each iteration of the dataset was as follows:

```
dataset_one_matrix_unique <- as.matrix(dataset_one_unique)
wagneronetreeunique <- Rmix(dataset_one_matrix_unique, method = "Wagner",
    random.addition = 10, global = FALSE)
write.tree(wagneronetreeunique, "dataset_one/wagner/wagner_unique.tree")

dataset_two_matrix_unique <- as.matrix(dataset_two_unique)
wagnertwotreeunique <- Rmix(datasettwomatrixunique, method = "Wagner",
    random.addition = 10, global = FALSE)
write.tree(wagnertwotreeunique, "dataset_two/wagner/wagner_unique.tree")

dataset_three_matrix_unique <- as.matrix(dataset_three_unique)
wagnerthreetreeunique <- Rmix(datasetthreematrixunique, method = "Wagner",
    random.addition = 10, global = FALSE)
write.tree(wagnerthreetreeunique, "dataset_three/wagner/wagner_unique.tree")
```

The trees then can be read into the R Environment through the `ggtree:read.tree` argument:

```
wagneronetreeunique <- read.tree("dataset_one/wagner/wagner_unique.tree")
wagnertwotreeunique <- read.tree("dataset_two/wagner/wagner_unique.tree")
wagnerthreetreeunique <- read.tree("dataset_three/wagner/wagner_unique.tree")
```

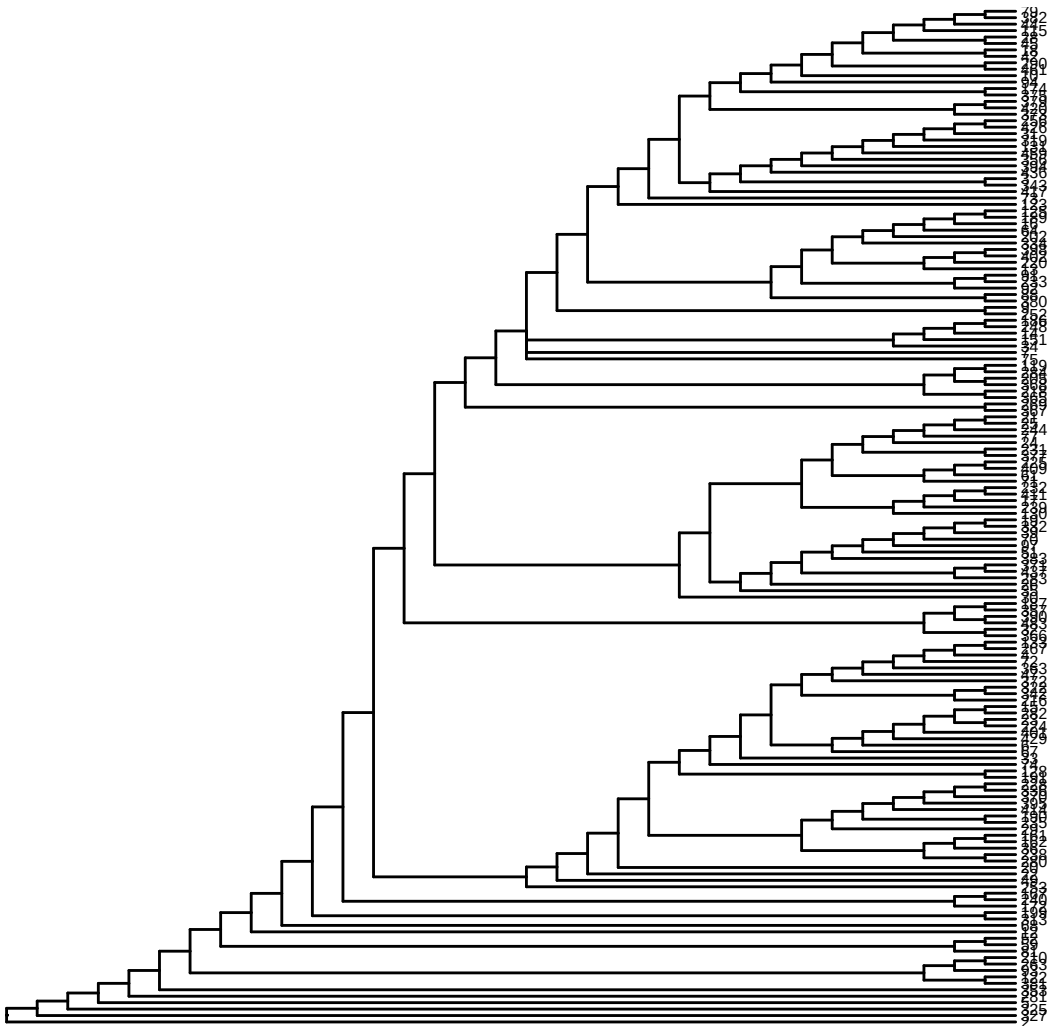
Or alternatively through the import functions in the `rio` package (preferred):

```
wagneronetreeunique <- import("https://github.com/CSHoggard/-sun-stones/raw/main/wagneronetreeunique.rds
wagnertwotreeunique <- import("https://github.com/CSHoggard/-sun-stones/raw/main/wagnertwotreeunique.rds
wagnerthreetreeunique <- import("https://github.com/CSHoggard/-sun-stones/raw/main/wagnerthreetreeuniqu
```

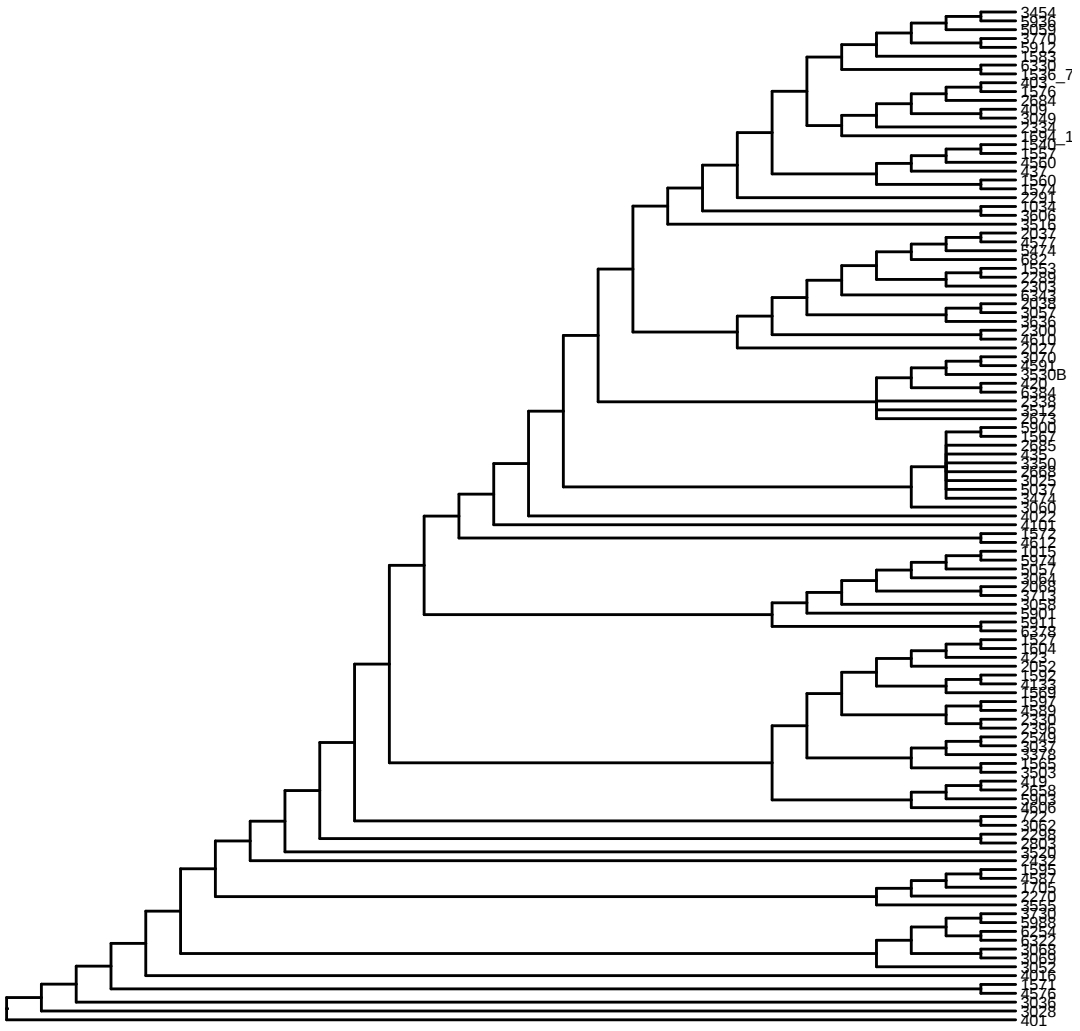
These can then be visualised through the `ggtree` visualisation tools, with consensus trees computer through the following code:

```
consensusonewagner <- consensus(wagneronetreeunique, p = 0.5,
    check.labels = TRUE)
consensustwowagner <- consensus(wagnertwotreeunique, p = 0.5,
    check.labels = FALSE)
consensusthreewagner <- consensus(wagnerthreetreeunique, p = 0.5,
```

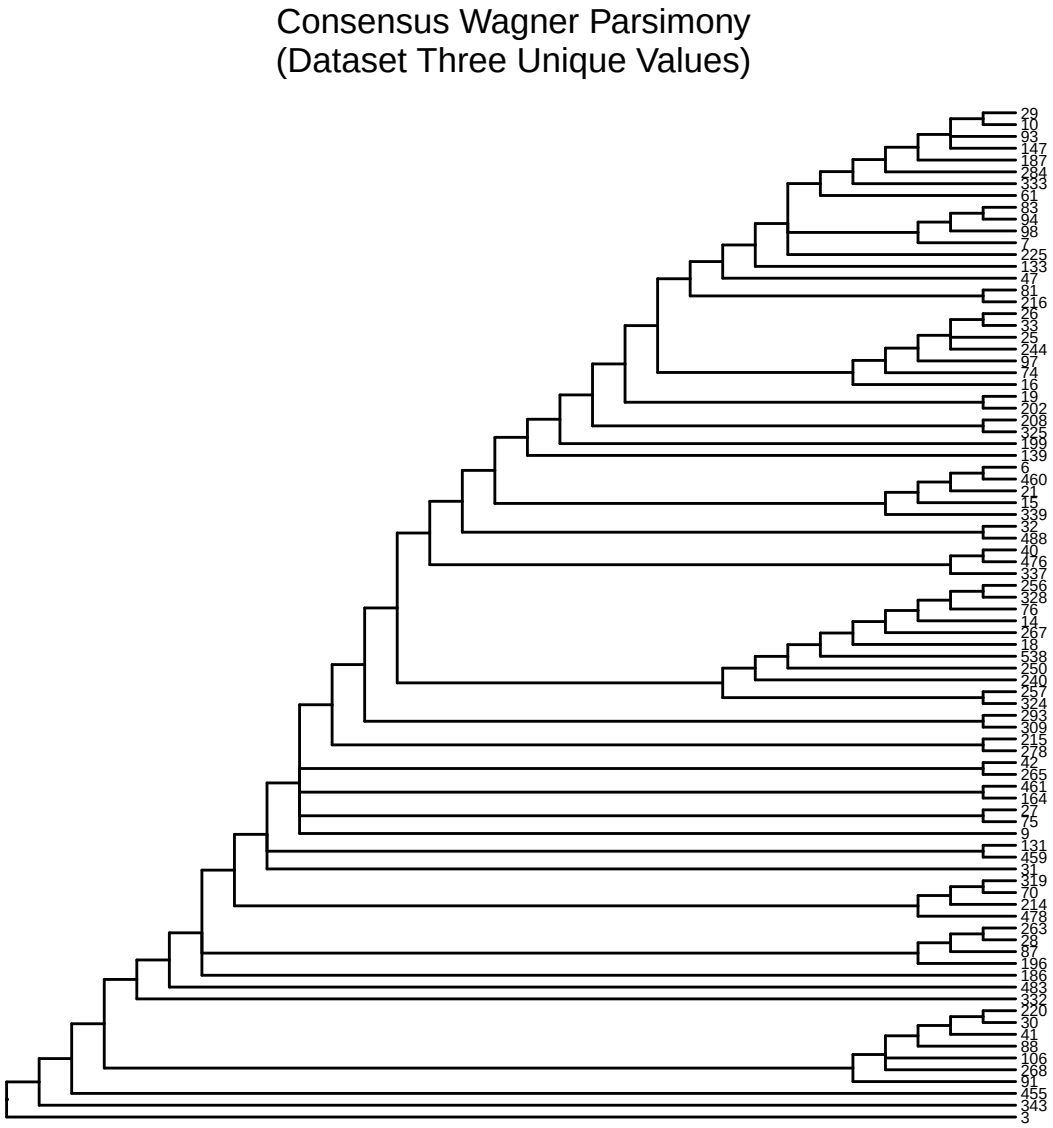
```
    check.labels = FALSE)

ggtree(consensusonewagner, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("Consensus Wagner Parsimony\n(Dataset One Unique Values)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```




```
ggtree(consensustwowagner, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("Consensus Wagner Parsimony\n(Dataset Two Unique Values)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```

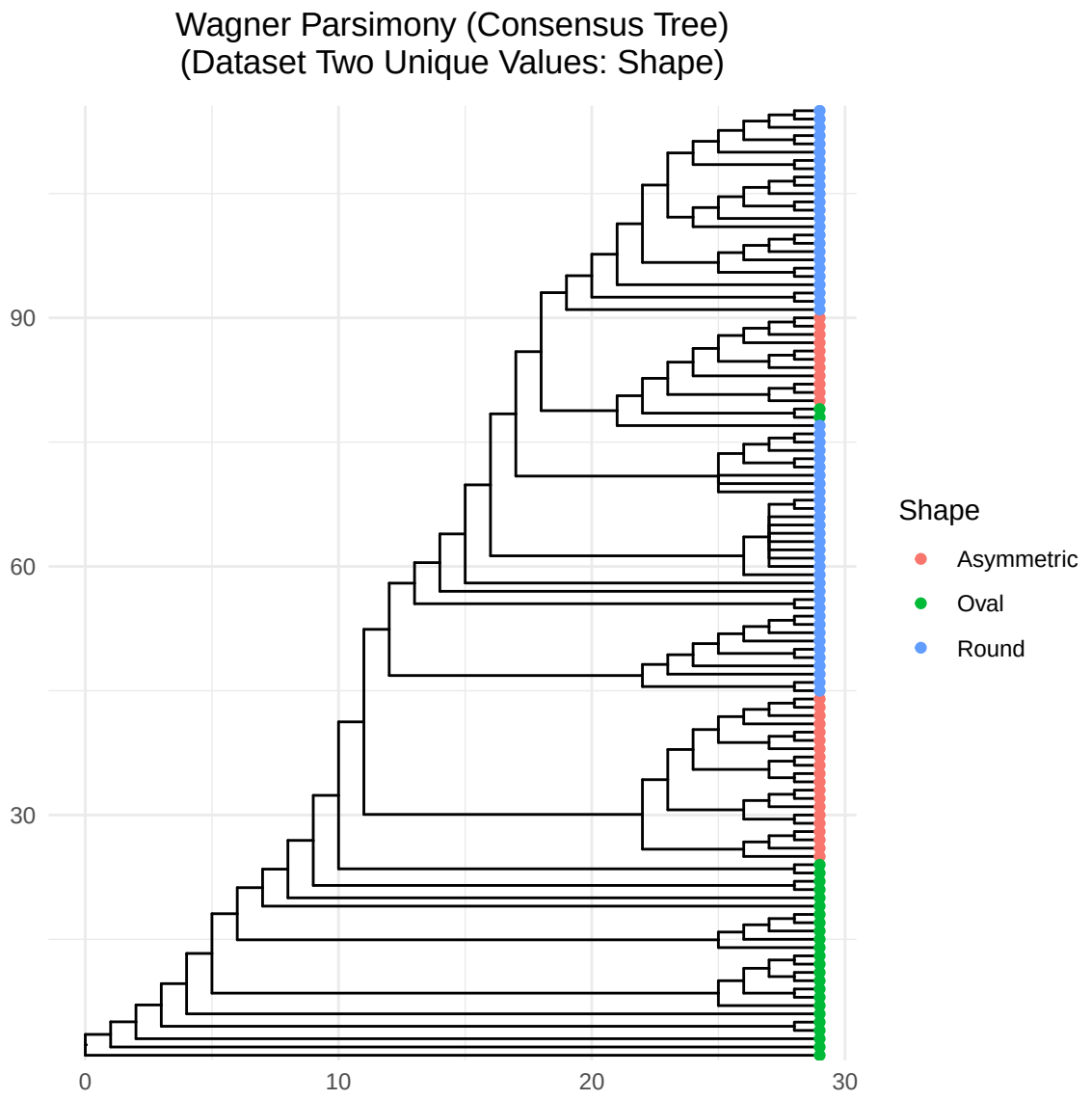



```
ggtree(consensusthreewagner, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("Consensus Wagner Parsimony\n(Dataset Three Unique Values)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```

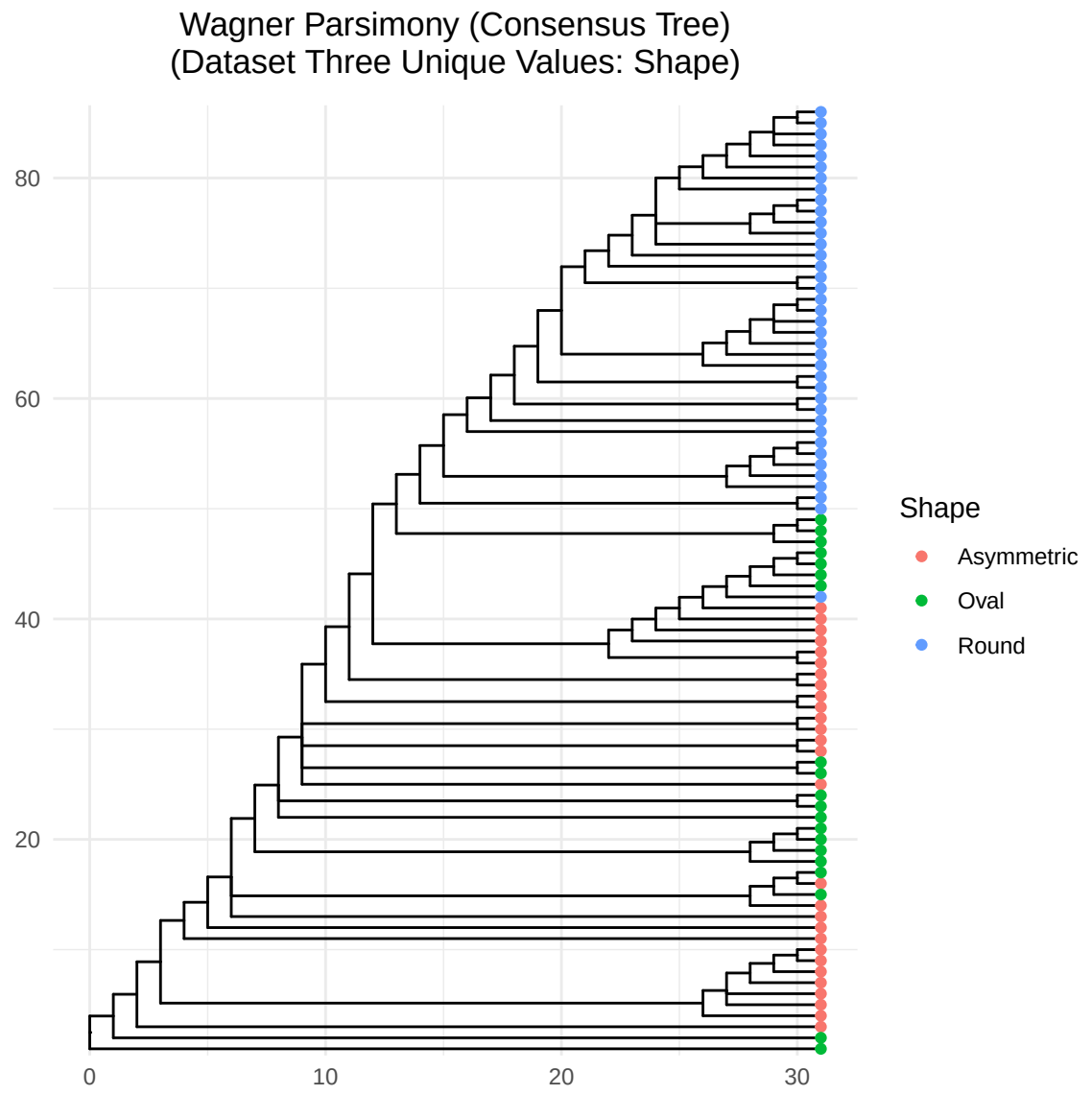


Factors such as shape, material, size and context can now be visualised:

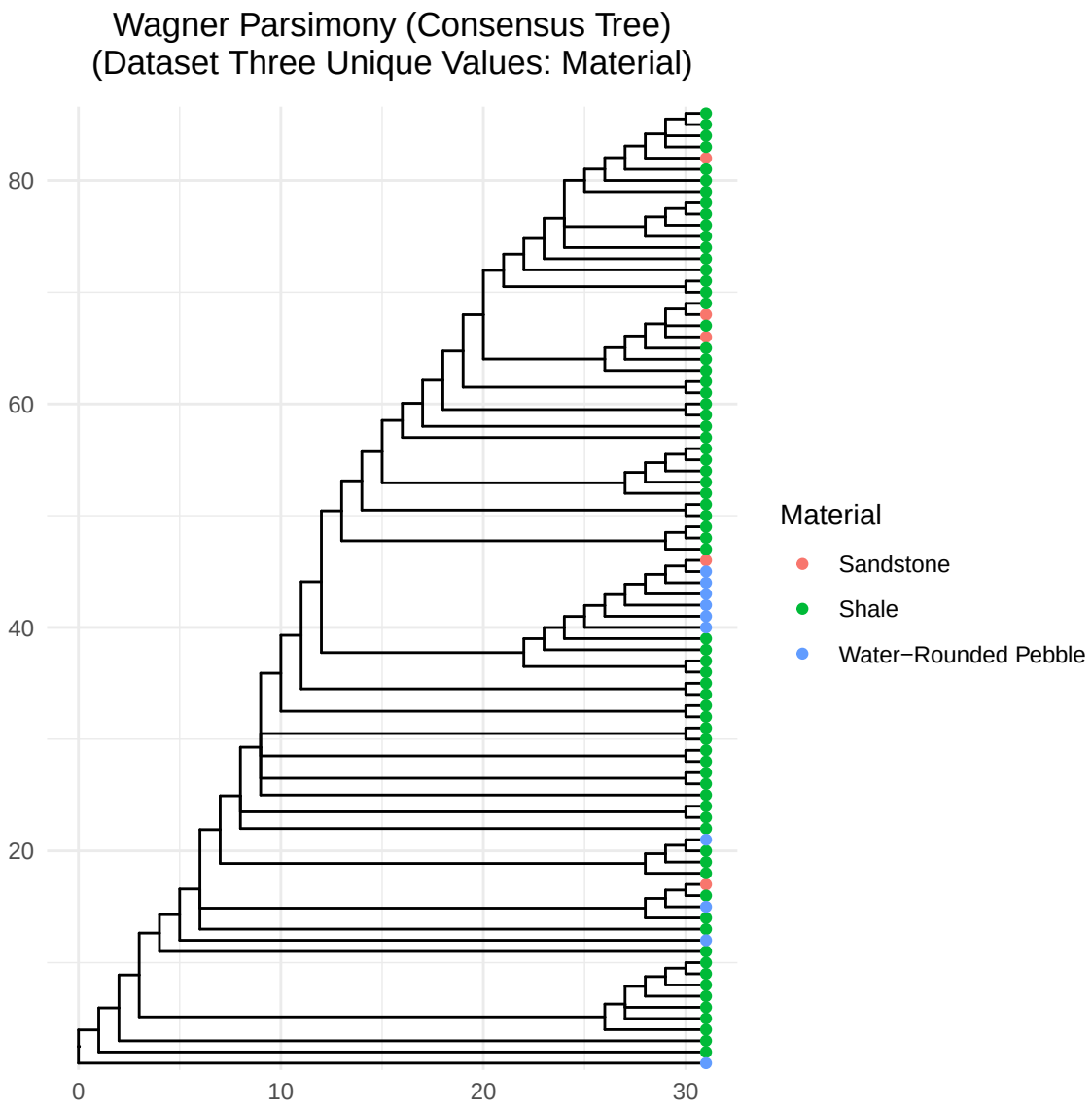
```
g7 <- ggtree(consensustwowagner, layout = "rectangular")
g8 <- ggtree(consensusthreewagner, layout = "rectangular")

g7 %<+% dataset_two_unique_meta + geom_tippoint(aes(colour = shape)) +
    labs(colour = "Shape") + ggtitle("Wagner Parsimony (Consensus Tree)\n(Dataset Two Unique Values: Sha
    scale_colour_discrete(name = "Shape", labels = c("Asymmetric",
        "Oval", "Round")) + theme_minimal() + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



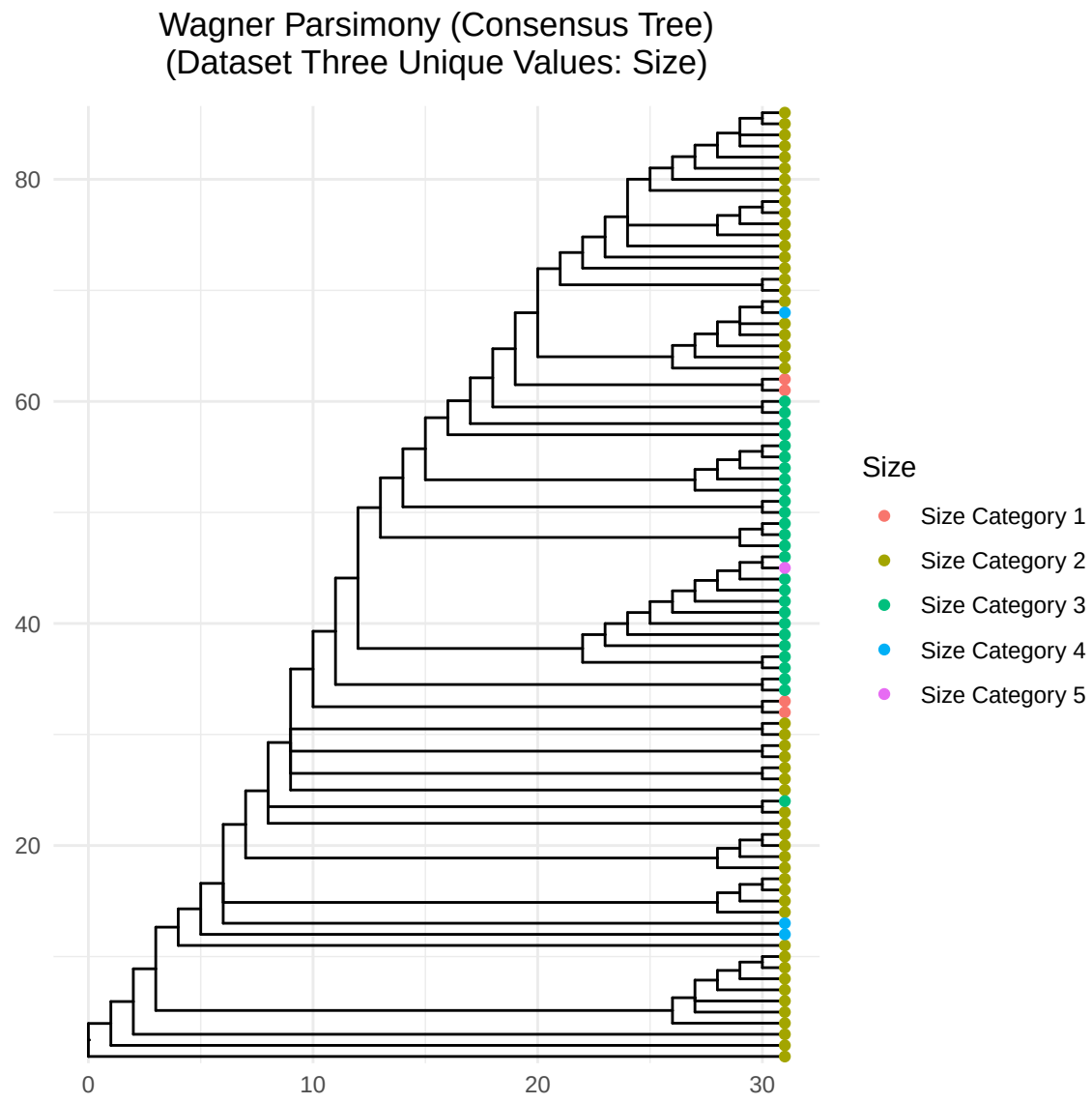

```
g8 %<+% dataset_three_unique_meta + geom_tippoint(aes(colour = shape)) +
    labs(colour = "Shape") + ggtitle("Wagner Parsimony (Consensus Tree)\n(Dataset Three Unique Values: S
    scale_colour_discrete(name = "Shape", labels = c("Asymmetric",
        "Oval", "Round")) + theme_minimal() + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



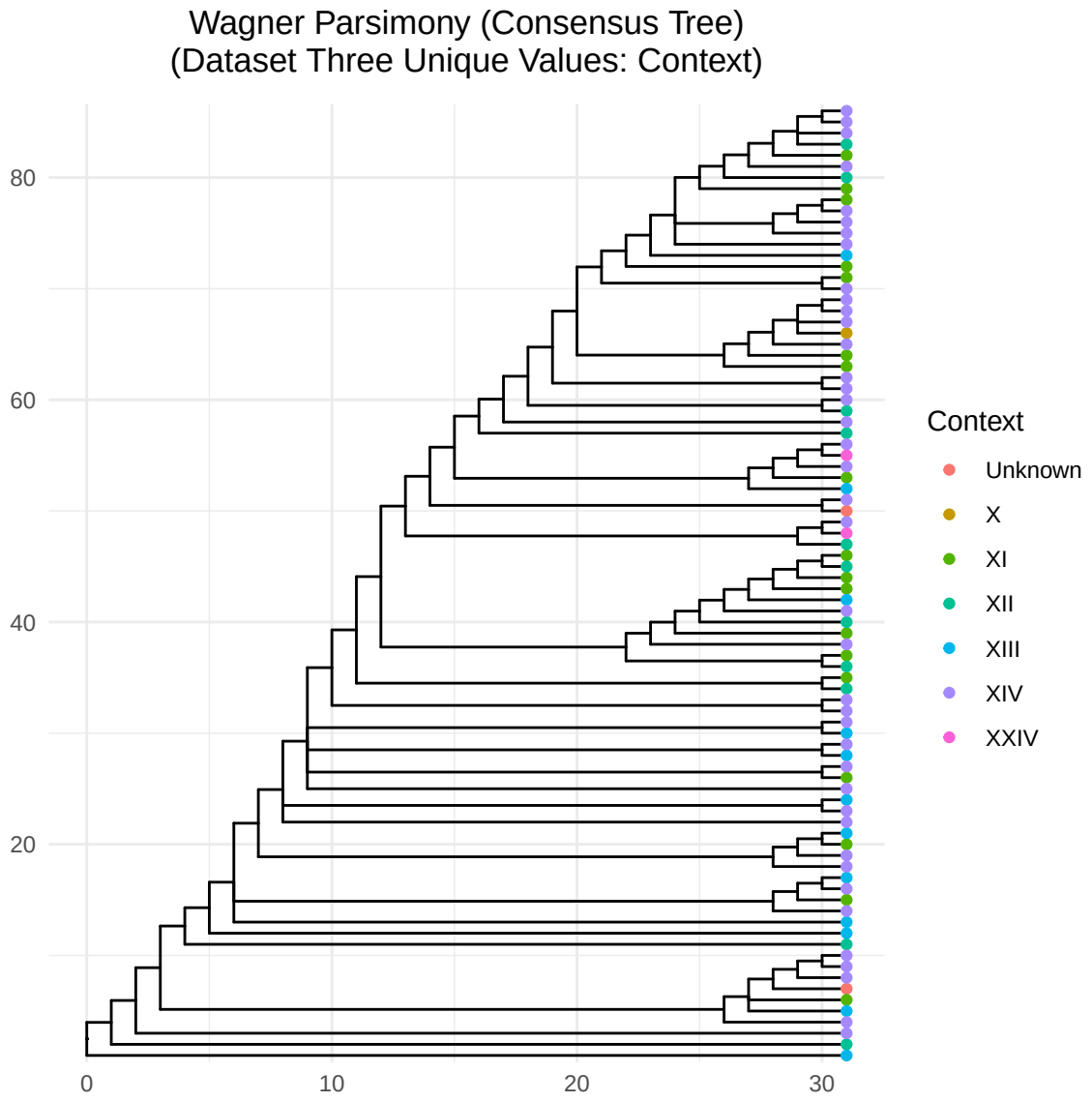

```
g8 %<+% dataset_three_unique_meta + geom_tippoint(aes(colour = material)) +
    labs(colour = "Material") + ggtitle("Wagner Parsimony (Consensus Tree)\n(Dataset Three Unique Values
    scale_colour_discrete(name = "Material", labels = c("Sandstone",
        "Shale", "Water-Rounded Pebble")) + theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```




```
g8 %<+% dataset_three_unique_meta + geom_tippoint(aes(colour = size)) +
    labs(colour = "Size") + ggtitle("Wagner Parsimony (Consensus Tree)\n(Dataset Three Unique Values: S
    scale_colour_discrete(name = "Size", labels = c("Size Category 1",
        "Size Category 2", "Size Category 3", "Size Category 4",
        "Size Category 5")) + theme_minimal() + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```




```
g8 %<+% dataset_three_unique_meta + geom_tippoint(aes(colour = gcontext)) +
    labs(colour = "Context") + ggtitle("Wagner Parsimony (Consensus Tree)\n(Dataset Three Unique Values
    scale_colour_discrete(name = "Context", labels = c("Unknown",
        "X", "XI", "XII", "XIII", "XIV", "XXIV")) + theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```



Dollo Parsimony

In detail, the Dollo method for each iteration of the dataset was as follows:

```
dolloonetreeunique <- Rdollop(dataset_one_matrix_unique, method = "dollo",
    random.addition = 10, global = FALSE)
write.tree(dolloonetreeunique, "dataset_one/dollo/dollo_unique.tree")

dollotwotreeunique <- Rdollop(datasettwomatrixunique, method = "dollo",
    random.addition = 10, global = FALSE)
write.tree(dollotwotreeunique, "dataset_two/dollo/dollo_unique.tree")

dollothreetreeunique <- Rdollop(datasetthreematrixunique, method = "dollo",
    random.addition = 10, global = FALSE)
write.tree(dollothreetreeunique, "dataset_three/dollo/dollo_unique.tree")
```

Note: the matrices were produced in the previous parsimony method:

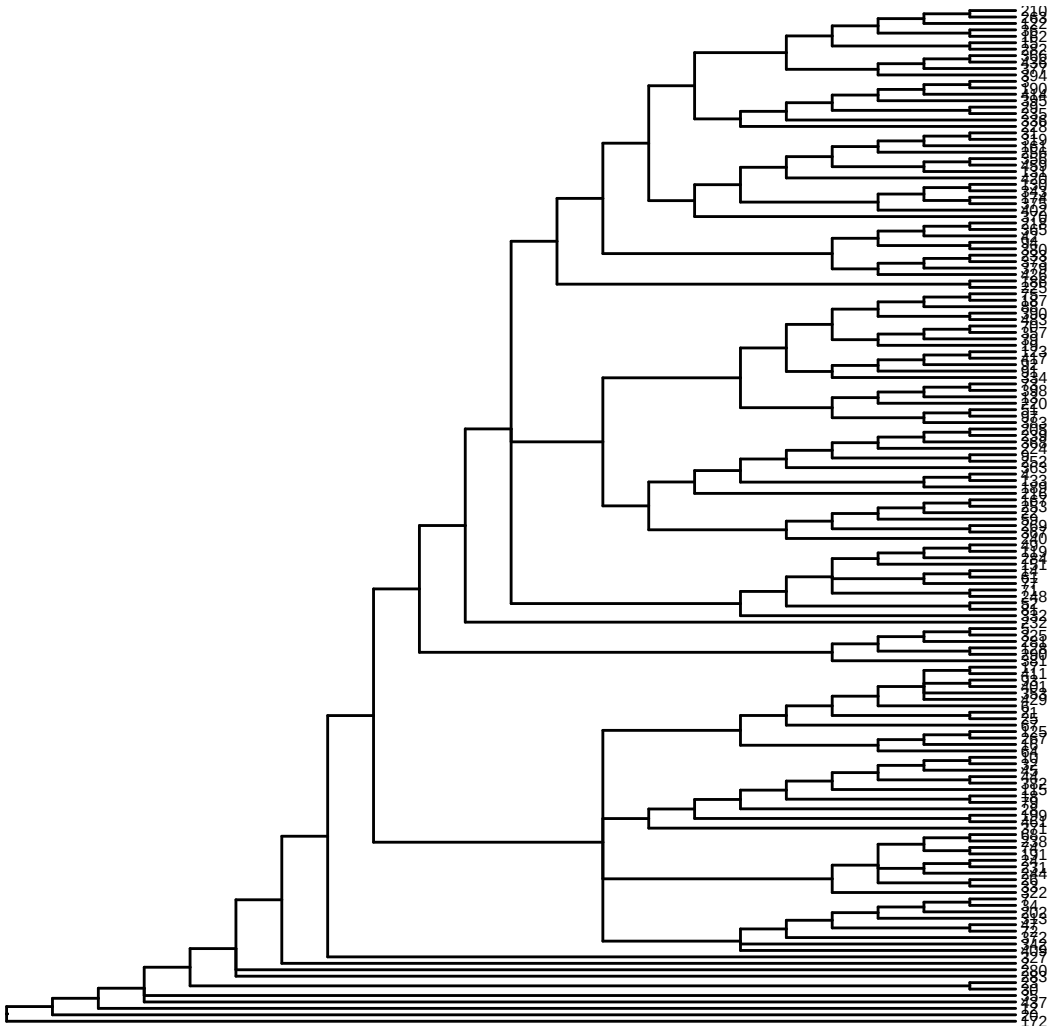
```
dolloonetreeunique <- read.tree("dataset_one/dollo/dollo_unique.tree")
dollotwotreeunique <- read.tree("dataset_two/dollo/dollo_unique.tree")
dollothreetreeunique <- read.tree("dataset_three/dollo/dollo_unique.tree")
```

Or alternatively through the import functions in the `rio` package (preferred):

```
dolloonetreeunique <- import("https://github.com/CSHoggard/-sun-stones/raw/main/dolloonetreeunique.rds")
dollotwotreeunique <- import("https://github.com/CSHoggard/-sun-stones/raw/main/dollotwotreeunique.rds")
dollothreetreeunique <- import("https://github.com/CSHoggard/-sun-stones/raw/main/dollothreetreeunique.r
```

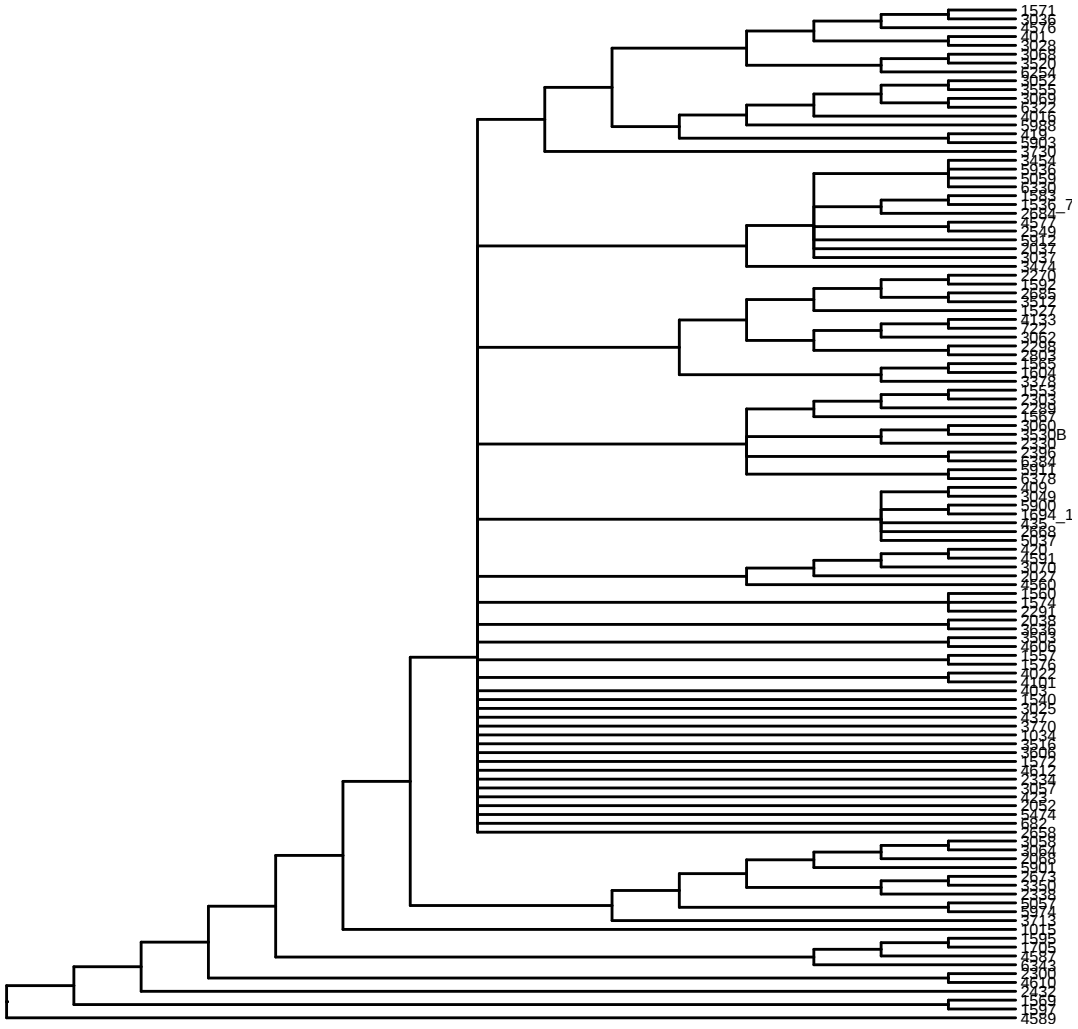
These can then be visualised through the `ggtree` visualisation tools, with consensus trees computer through the following code:

```
consensusonedollo <- consensus(dolloonetreeunique, p = 0.5, check.labels = TRUE)
consensustwodollo <- consensus(dollotwotreeunique, p = 0.5, check.labels = FALSE)
consensusthreedollo <- consensus(dollothreetreeunique, p = 0.5,
    check.labels = FALSE)

ggtree(consensusonedollo, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("Dollo Parsimony (Consensus Tree)\n(Dataset One: Unique Values)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```

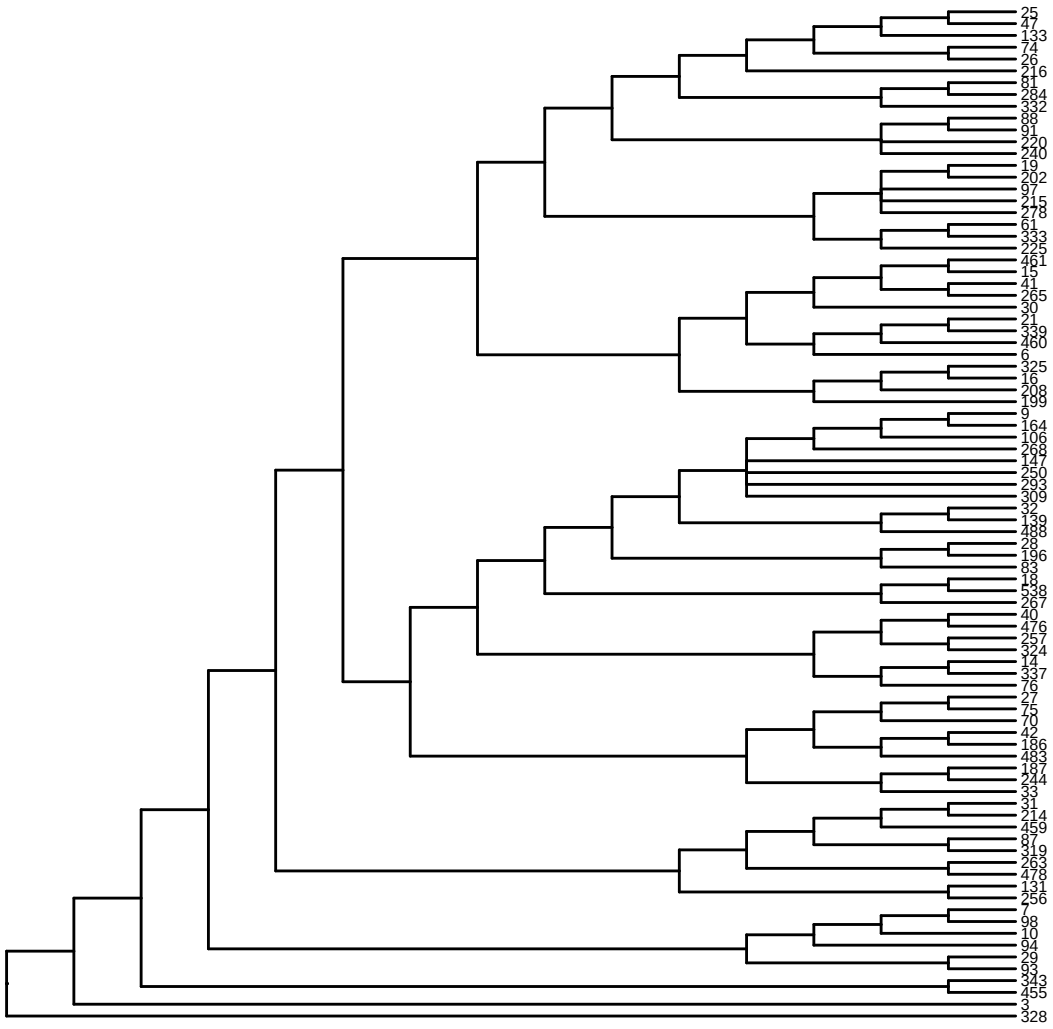



```
ggtree(consensustwodollo, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("Dollo Parsimony (Consensus Tree)\n(Dataset Two: Unique Values)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```

Dollo Parsimony (Consensus Tree)
(Dataset Two: Unique Values)

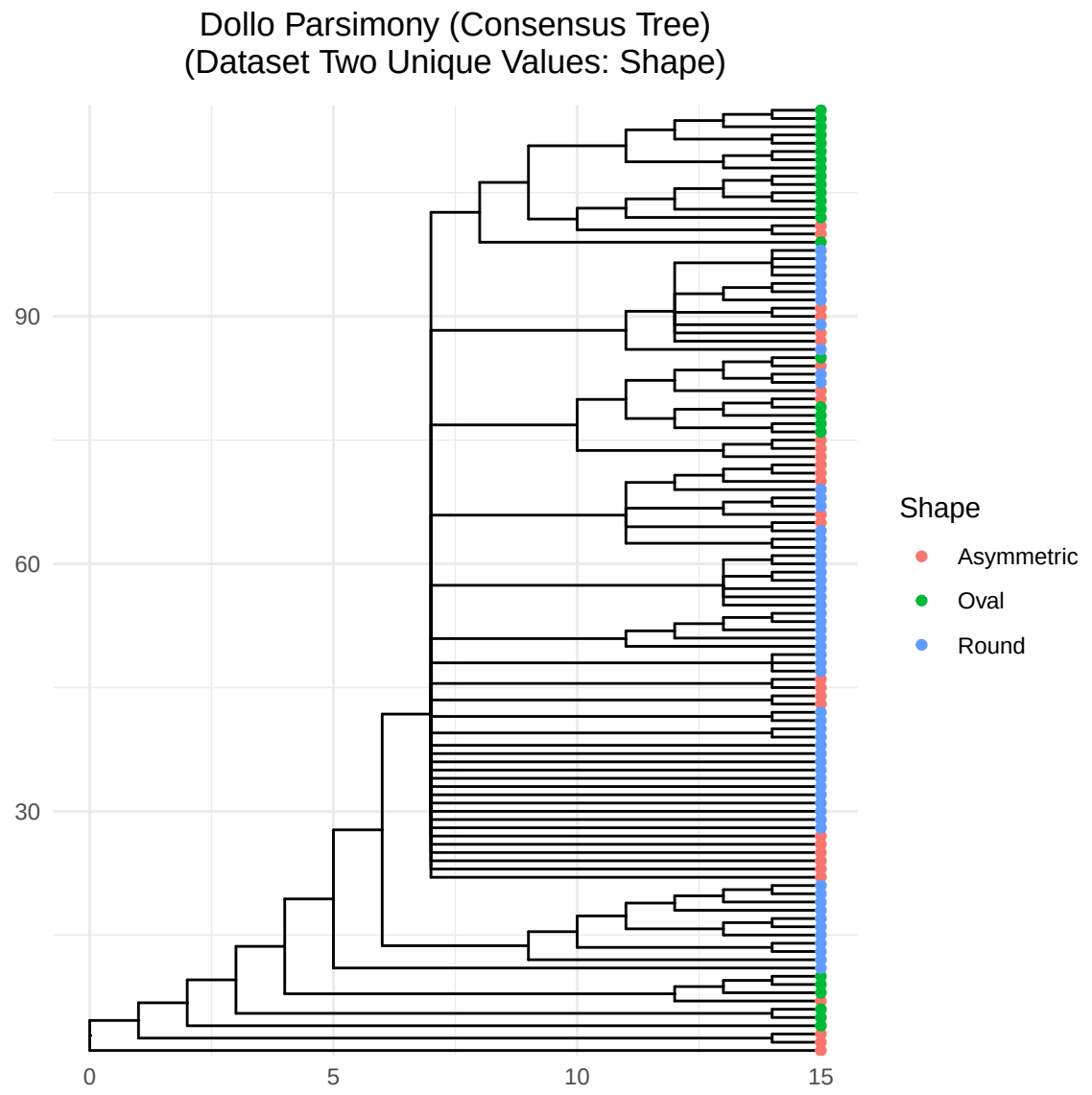



```
ggtree(consensusthreedollo, layout = "rectangular") + geom_tiplab(size = 2) +
    ggtitle("Dollo Parsimony (Consensus Tree)\n(Dataset Three: Unique Values)") +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```

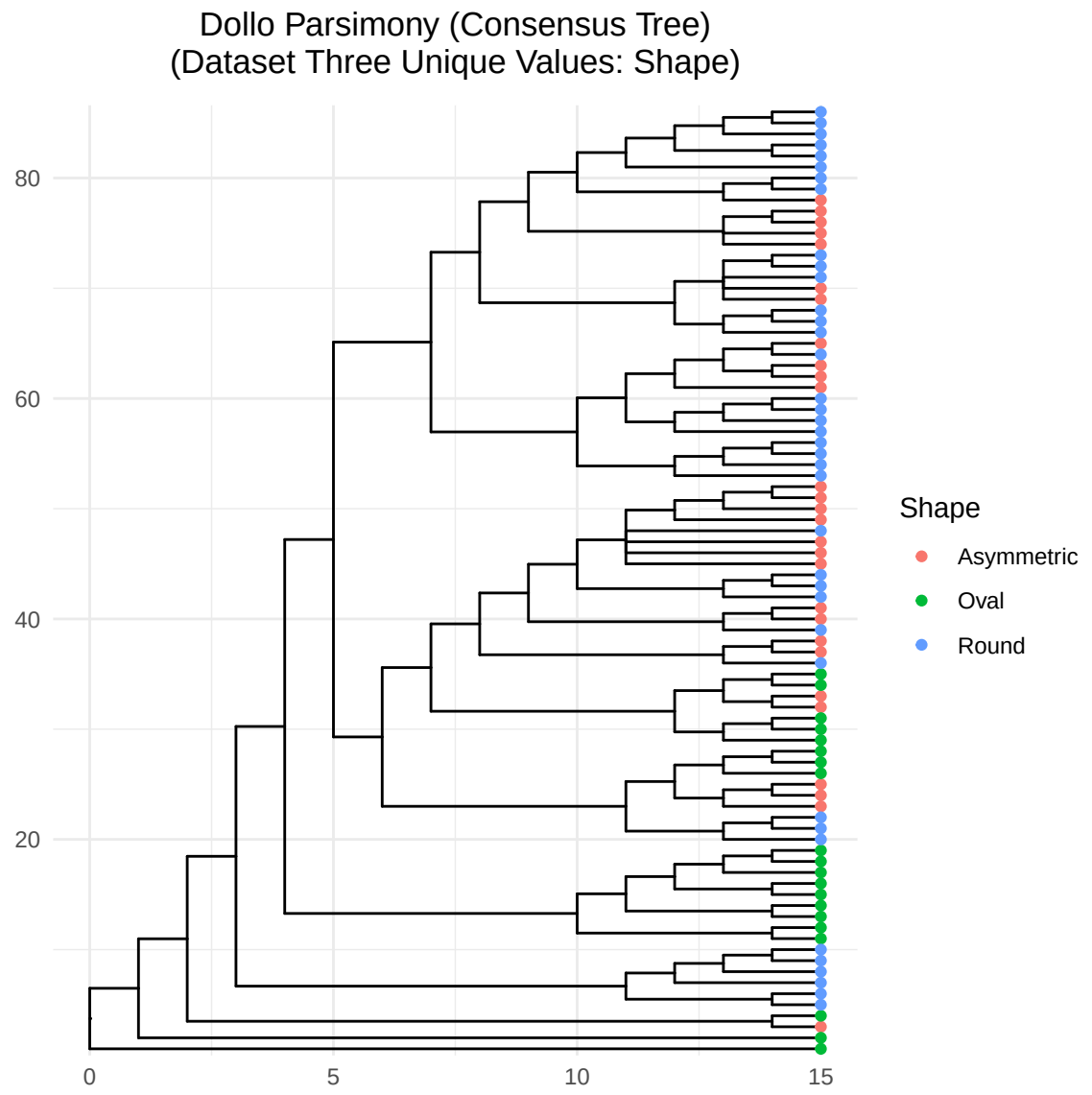


Factors such as shape, material, size and context can now be visualised:

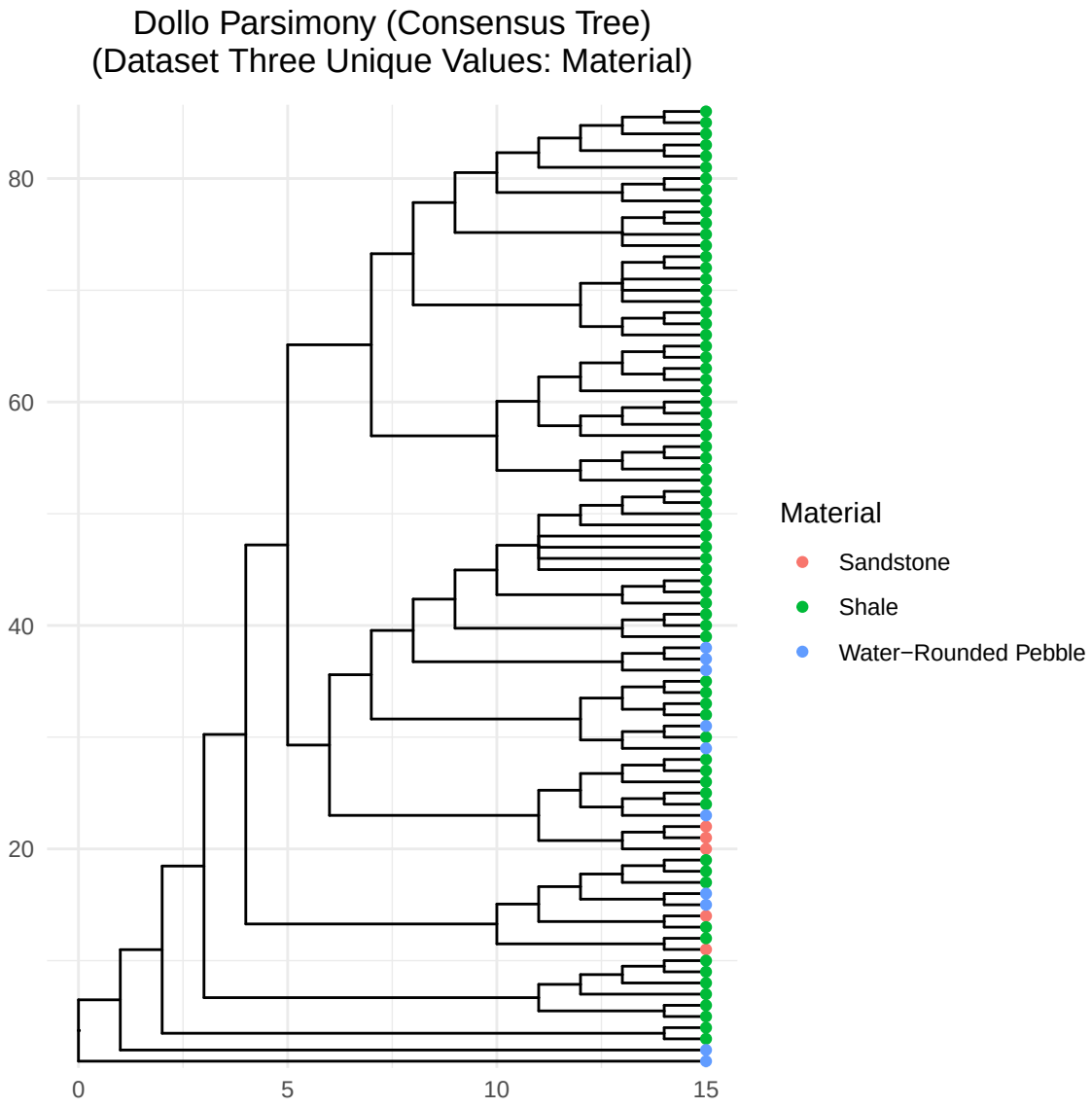
```
g9 <- ggtree(consensustwodollo, layout = "rectangular")
g10 <- ggtree(consensusthreedollo, layout = "rectangular")

g9 %<+% dataset_two_unique_meta + geom_tippoint(aes(colour = shape)) +
    labs(colour = "Shape") + ggtitle("Dollo Parsimony (Consensus Tree)\n(Dataset Two Unique Values: Shap
    scale_colour_discrete(name = "Shape", labels = c("Asymmetric",
        "Oval", "Round")) + theme_minimal() + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



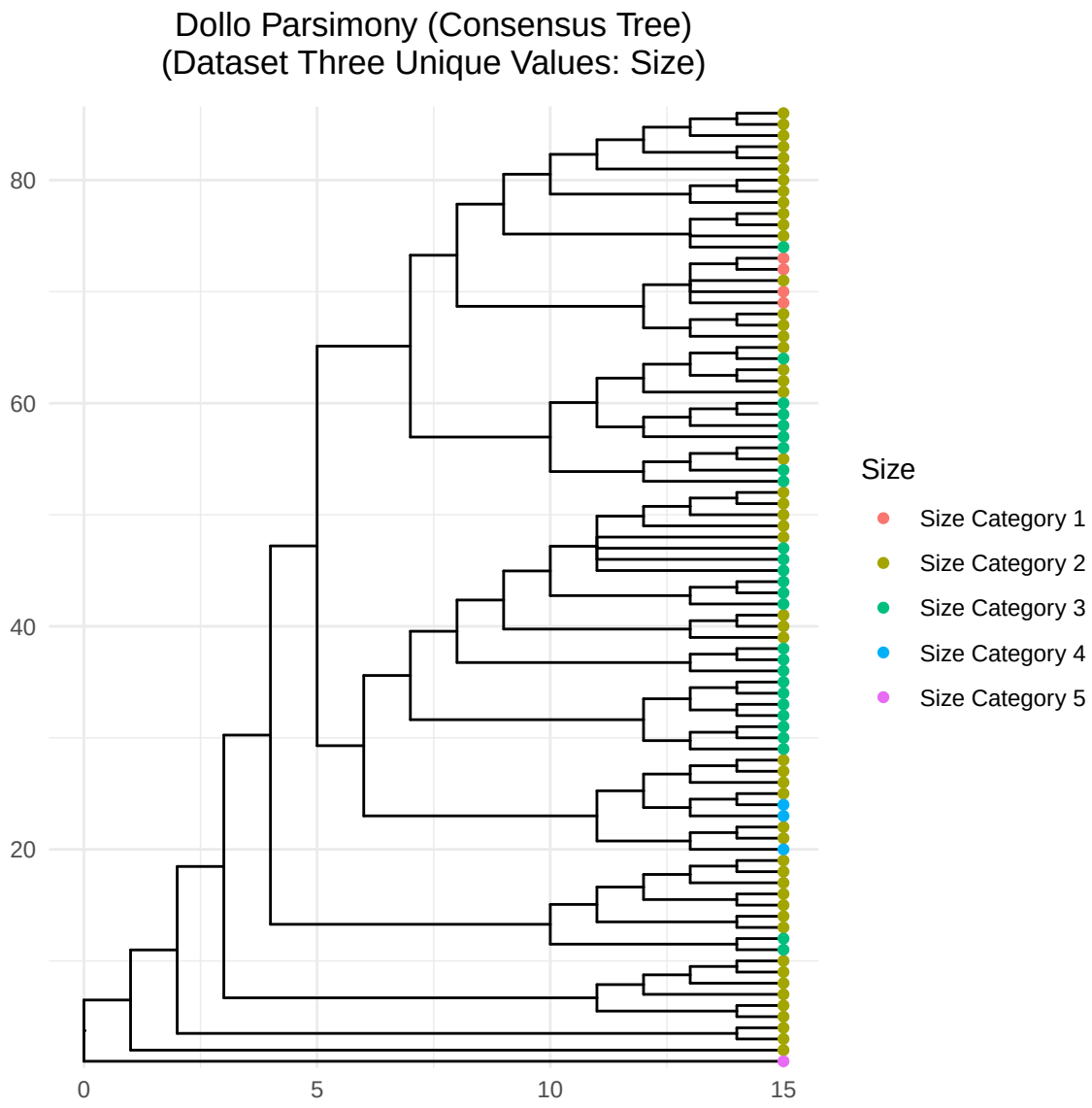

```
g10 %<+% dataset_three_unique_meta + geom_tippoint(aes(colour = shape)) +
    labs(colour = "Shape") + ggtitle("Dollo Parsimony (Consensus Tree)\n(Dataset Three Unique Values: Sh
    scale_colour_discrete(name = "Shape", labels = c("Asymmetric",
        "Oval", "Round")) + theme_minimal() + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```



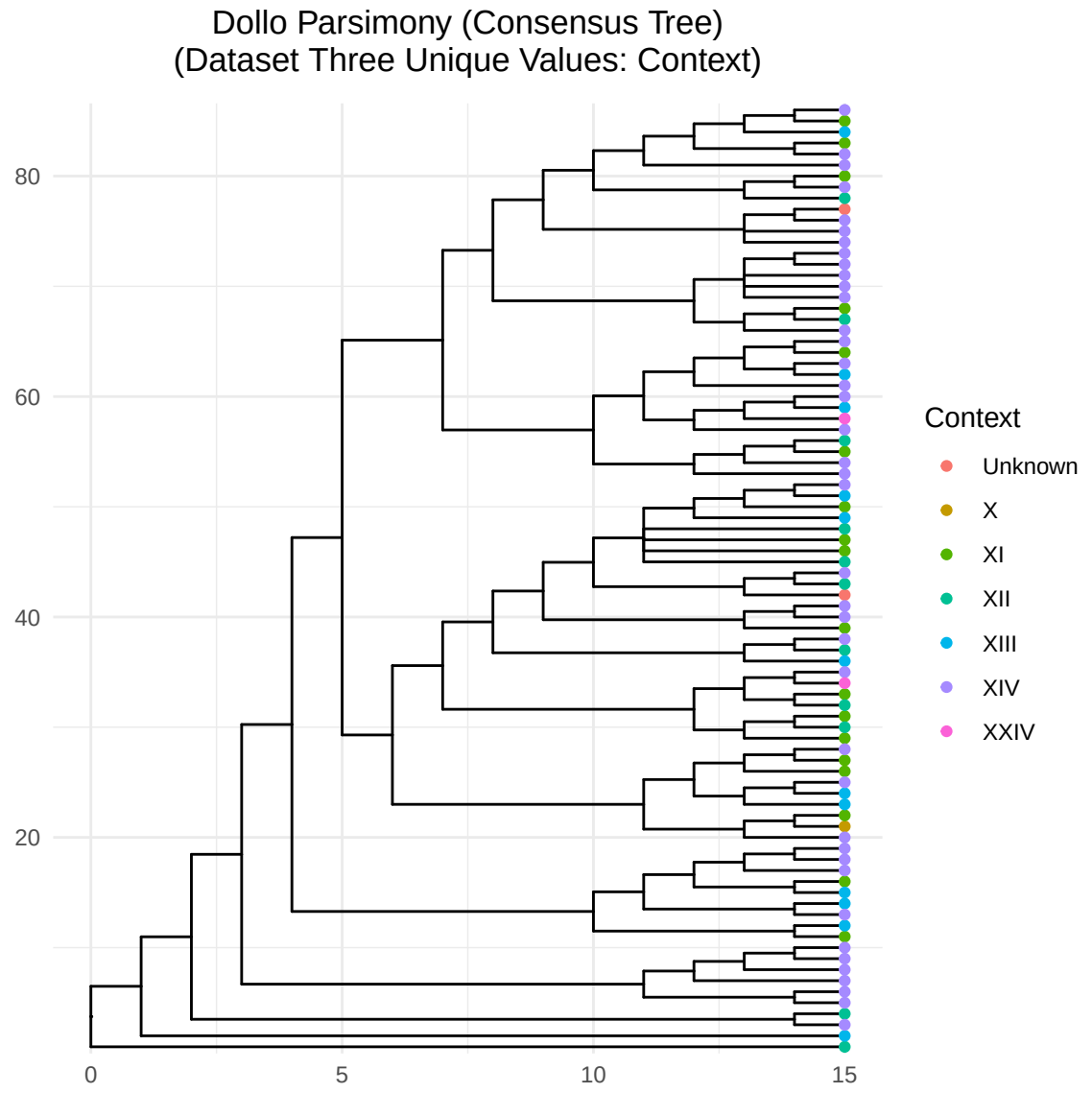

```
g10 %<+% dataset_three_unique_meta + geom_tippoint(aes(colour = material)) +
    labs(colour = "Material") + ggtitle("Dollo Parsimony (Consensus Tree)\n(Dataset Three Unique Values
    scale_colour_discrete(name = "Material", labels = c("Sandstone",
        "Shale", "Water-Rounded Pebble")) + theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```




```
g10 %<+% dataset_three_unique_meta + geom_tippoint(aes(colour = size)) +
    labs(colour = "Size") + ggtitle("Dollo Parsimony (Consensus Tree)\n(Dataset Three Unique Values: Si
    scale_colour_discrete(name = "Size", labels = c("Size Category 1",
        "Size Category 2", "Size Category 3", "Size Category 4",
        "Size Category 5")) + theme_minimal() + theme(plot.title = element_text(hjust = 0.5,
    margin = margin(0, 0, 10, 0)))
```




```
g10 %<+% dataset_three_unique_meta + geom_tippoint(aes(colour = gcontext)) +
    labs(colour = "Context") + ggtitle("Dollo Parsimony (Consensus Tree)\n(Dataset Three Unique Values:
    scale_colour_discrete(name = "Context", labels = c("Unknown",
        "X", "XI", "XII", "XIII", "XIV", "XXIV")) + theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))
```



Parsimony Tree Comparisons

We can summarise the differences between the two parsimony methods in the following code:

```
comparePhylo(consensusonewagner, consensusonedollo)
```

```
## => Comparing consensusonewagner with consensusonedollo.
## Both trees have the same number of tips: 158.
## Both trees have the same tip labels.
## Trees have different numbers of nodes: 155 and 147.
## Both trees are rooted.
## Both trees are not ultrametric.
## 146 clades in consensusonewagner not in consensusonedollo.
## 138 clades in consensusonedollo not in consensusonewagner.
```

```
comparePhylo(consensustwowagner, consensustwodollo)
```

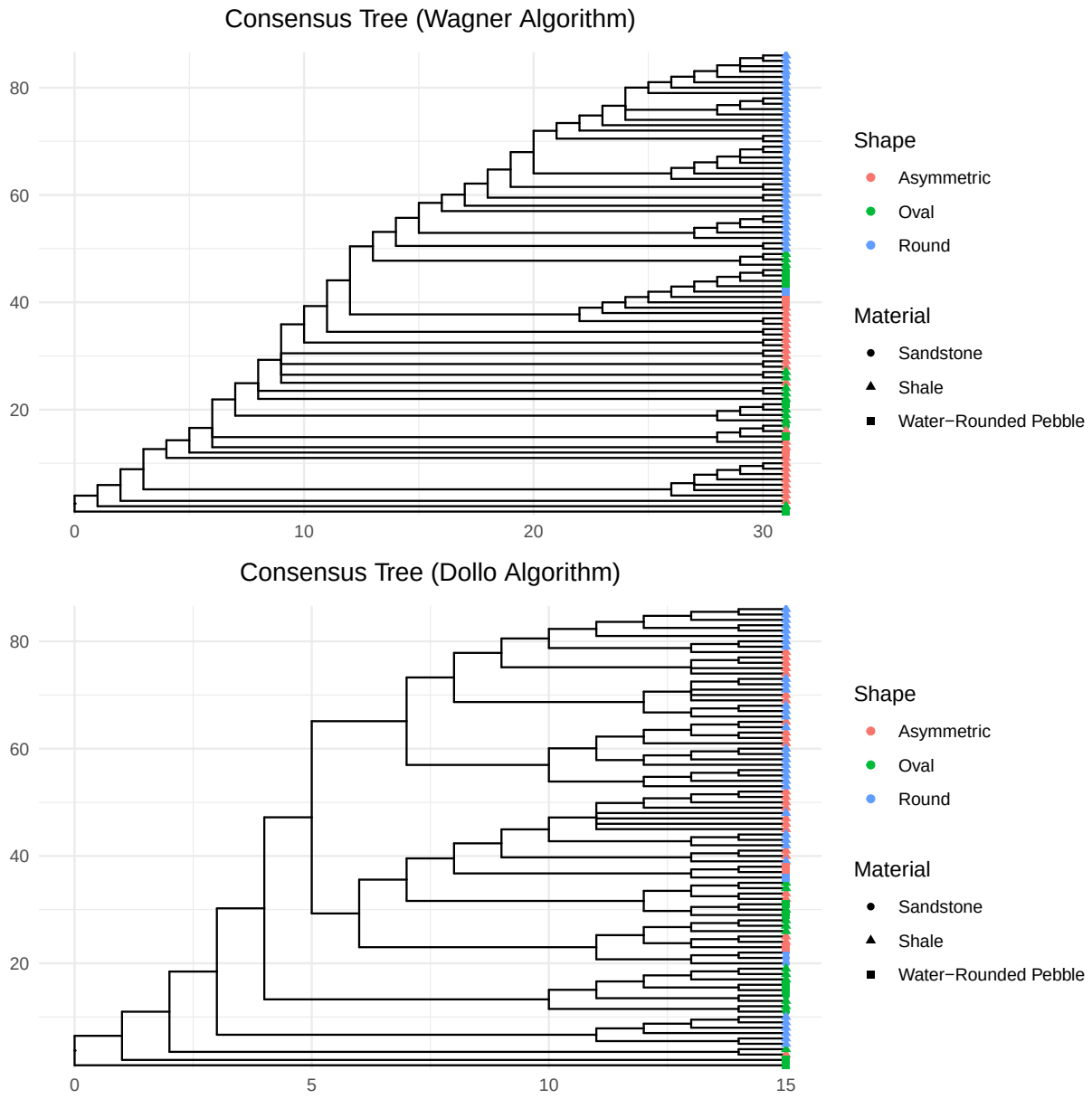
```
## => Comparing consensustwowagner with consensustwodollo.
```

```
## Both trees have the same number of tips: 115.
## Both trees have the same tip labels.
## Trees have different numbers of nodes: 106 and 76.
## Both trees are rooted.
## Both trees are not ultrametric.
## 99 clades in consensustwowagner not in consensustwodollo.
## 69 clades in consensustwodollo not in consensustwowagner.
```

```
comparePhylo(consensusthreewagner, consensusthreedollo)
```

```
## => Comparing consensusthreewagner with consensusthreedollo.
## Both trees have the same number of tips: 86.
## Both trees have the same tip labels.
## Trees have different numbers of nodes: 76 and 79.
## Both trees are rooted.
## Both trees are not ultrametric.
## 71 clades in consensusthreewagner not in consensusthreedollo.
## 74 clades in consensusthreedollo not in consensusthreewagner.
```

```
cg1 <- g8 %<+% dataset_three_unique_meta + geom_tippoint(aes(shape = material,
    colour = shape)) + ggtitle("Consensus Tree (Wagner Algorithm)") +
    scale_shape_discrete(name = "Material", labels = c("Sandstone",
        "Shale", "Water-Rounded Pebble")) + scale_colour_discrete(name = "Shape",
    labels = c("Asymmetric", "Oval", "Round")) + theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))

cg2 <- g10 %<+% dataset_three_unique_meta + geom_tippoint(aes(shape = material,
    colour = shape)) + ggtitle("Consensus Tree (Dollo Algorithm)") +
    scale_shape_discrete(name = "Material", labels = c("Sandstone",
        "Shale", "Water-Rounded Pebble")) + scale_colour_discrete(name = "Shape",
    labels = c("Asymmetric", "Oval", "Round")) + theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5, margin = margin(0,
        0, 10, 0)))

(cg1/cg2)
```


Consensus Tree (Wagner Algorithm)
80
60
40
20
0
10
20
30
Shape
Asymmetric
Oval
Round
Material
Sandstone
Shale
Water−Rounded Pebble
Consensus Tree (Dollo Algorithm)
80
60
40
20
0
5
10
15
Shape
Asymmetric
Oval
Round
Material
Sandstone
Shale
Water−Rounded Pebble