Supplementary Material 4: Cluster Validation

Measuring and assessing indeterminacy and variation in the morphology-syntax distinction

Sandra Auderset Adam J.R. Tallman

```
# set seed
set.seed(200)
```

Data Preparation

We first produce the simulated data set. We do this by developing four random list of 0s and 1s, a random list of 0s, 1s and 2s and a random list of integers from 0 to 10. The 0s and 1s for the binary variables were replaced ‘y(es)’s and ’n(o)’s to correspond to interruptability/contiguity, fixedness/variability, simplex/complex and freedom/boundedness variables of our data set. The 0s, 1s and 2s were replaced with ’y(es)’, ‘both’ and ‘n(o)’ respectively corresponding to prominence projection / projection deficiency. In our dataset exponence complexity varies from 1 to 10, and we thus create a variable that has a uniform distribution of numbers from 1 to 10 corresponding to exponence complexity.

```
# simulate random data set
FREE <- as.character(sample(c("y", "n"), 100, replace = T))
INTERone <- as.character(sample(c("y", "n"), 100, replace = T))
CODelab <- as.character(sample(c("y", "n"), 100, replace = T))
PMfixed <- as.character(sample(c("y", "n"), 100, replace = T))
PRM <- as.character(sample(c("y", "both", "n"), 100, replace = T))
EXPcomplex <- sample(1:10, 100, replace = T, prob = 10:1)

sim_db <- as_tibble(cbind(FREE, INTERone, CODelab, PMfixed, PRM, EXPcomplex)) %>%
    mutate_if(is.character, as.factor)
glimpse(sim_db)
```

```
## Rows: 100
## Columns: 6
## $ FREE       <fct> n, n, y, n, y, n, n, y, n, n, n, n, y, n, y, n, n, y, n, n,~
## $ INTERone   <fct> y, n, n, y, y, y, y, n, y, y, y, y, y, n, y, y, y, n, y, n,~
## $ CODelab    <fct> y, y, n, y, n, y, y, y, n, y, y, y, y, n, n, y, n, y, n, n,~
## $ PMfixed    <fct> n, n, n, y, n, n, y, n, n, n, n, n, y, n, n, n, n, y, n, y,~
## $ PRM        <fct> both, y, both, both, y, y, both, n, n, both, y, n, y, y, y,~
## $ EXPcomplex <fct> 3, 2, 1, 3, 5, 7, 6, 5, 6, 4, 1, 2, 6, 2, 5, 6, 3, 6, 1, 9,~
```

```
summary(sim_db)
```

```
##  FREE   INTERone CODelab PMfixed   PRM       EXPcomplex
##  n:55   n:43     n:53    n:54    both:24   1      :18
##  y:45   y:57     y:47    y:46    n   :36   6      :17
##                                  y   :40   3      :16
##                                            2      :15
##                                            5      :13
##                                            4      : 7
##                                            (Other):14
```

Next we load data (available at DOI), excluding columns we do not need for the analysis, i.e. we exclude the variables that are summarized in Exponence Complexity and certain meta information about the languages such as affiliation and glottocode.

```
# load data
db <- read_csv(here("SM1_Database.csv")) %>%
    select(Language, PMfixed:EXPcomplex)
```

```
## Rows: 767 Columns: 16
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (11): Language, Glottocode, Family, Morphs, GlossAuthor, ClassAuthorSimp...
## dbl  (5): EXPcomplex, AMsegment, AMsupplN, EXPmultN, MSTfossilN
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
glimpse(db)
```

```
## Rows: 767
## Columns: 7
## $ Language   <chr> "AsheninkaPerene", "AsheninkaPerene", "AsheninkaPerene", "A~
## $ PMfixed    <chr> "y", "y", "y", "y", "y", "y", "y", "y", "y", "y", "y", "y",~
## $ FREE       <chr> "n", "n", "n", "y", "n", "n", "y", "n", "n", "y", "n", "n",~
## $ INTERone   <chr> "y", "y", "y", "n", "n", "y", "y", "y", "y", "y", "y", "y",~
## $ PRM        <chr> "both", "both", "both", "both", "both", "both", "both", "bo~
## $ CODelab    <chr> "y", "y", "y", "y", "y", "y", "y", "y", "n", "y", "y", "y",~
## $ EXPcomplex <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
```

We split the data set by language and select the relevant columns. Then we convert all the character columns to factors to be able to compute the dissimilarity matrices.

```
db_names <- db %>%
    group_by(Language) %>%
    group_keys() %>%
    pull(1)
db_lang <- db %>%
    group_split(Language) %>%
    setNames(db_names) %>%
    map(as_tibble) %>%
    map(., ~(.x %>%
        select(-Language))) %>%
    map(., ~(.x %>%
        mutate_if(is.character, as.factor))) %>%
    drop.levels()
glimpse(db_lang)
```

```
## List of 8
##  $ AsheninkaPerene    : tibble [98 x 6] (S3: tbl_df/tbl/data.frame)
##   ..$ PMfixed   : Factor w/ 2 levels "n","y": 2 2 2 2 2 2 2 2 2 2 ...
##   ..$ FREE      : Factor w/ 2 levels "n","y": 1 1 1 2 1 1 2 1 1 2 ...
##   ..$ INTERone  : Factor w/ 2 levels "n","y": 2 2 2 1 1 2 2 2 2 2 ...
##   ..$ PRM       : Factor w/ 3 levels "both","n","y": 1 1 1 1 1 1 1 1 2 1 ...
##   ..$ CODelab   : Factor w/ 2 levels "n","y": 2 2 2 2 2 2 2 2 1 2 ...
##   ..$ EXPcomplex: num [1:98] 1 1 1 1 1 1 1 1 1 1 ...
## $ Cavineña           : tibble [56 x 6] (S3: tbl_df/tbl/data.frame)
##   ..$ PMfixed   : Factor w/ 2 levels "n","y": 1 1 1 1 2 1 2 2 2 2 ...
##   ..$ FREE      : Factor w/ 2 levels "n","y": 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ INTERone  : Factor w/ 2 levels "n","y": 2 2 2 2 2 2 2 2 2 2 ...
##   ..$ PRM       : Factor w/ 3 levels "both","n","y": 1 1 1 1 2 2 2 2 2 2 ...
##   ..$ CODelab   : Factor w/ 2 levels "n","y": 2 2 2 2 1 1 1 1 1 1 ...
##   ..$ EXPcomplex: num [1:56] 3 1 1 1 3 1 1 1 1 1 ...
## $ CentralAlaskanYupik: tibble [81 x 6] (S3: tbl_df/tbl/data.frame)
##   ..$ PMfixed   : Factor w/ 2 levels "n","y": 1 1 1 1 2 2 2 2 2 2 ...
##   ..$ FREE      : Factor w/ 2 levels "n","y": 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ INTERone  : Factor w/ 2 levels "n","y": 2 2 2 2 1 1 1 1 1 1 ...
##   ..$ PRM       : Factor w/ 2 levels "n","y": 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ CODelab   : Factor w/ 2 levels "n","y": 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ EXPcomplex: num [1:81] 2 1 1 1 1 1 1 1 1 1 ...
## $ Chacobo            : tibble [96 x 6] (S3: tbl_df/tbl/data.frame)
##   ..$ PMfixed   : Factor w/ 2 levels "n","y": 1 1 2 2 2 2 2 2 1 1 ...
##   ..$ FREE      : Factor w/ 2 levels "n","y": 2 1 1 1 1 1 1 1 2 2 ...
##   ..$ INTERone  : Factor w/ 2 levels "n","y": 2 2 2 2 2 2 2 1 2 2 ...
##   ..$ PRM       : Factor w/ 3 levels "both","n","y": 3 1 1 1 1 1 3 2 3 3 ...
##   ..$ CODelab   : Factor w/ 2 levels "n","y": 1 1 1 1 1 1 2 1 2 2 ...
##   ..$ EXPcomplex: num [1:96] 1 1 1 1 1 1 1 3 1 1 ...
## $ Hup                : tibble [65 x 6] (S3: tbl_df/tbl/data.frame)
##   ..$ PMfixed   : Factor w/ 2 levels "n","y": 1 2 2 2 2 2 2 2 2 2 ...
##   ..$ FREE      : Factor w/ 2 levels "n","y": 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ INTERone  : Factor w/ 2 levels "n","y": 2 1 1 2 1 1 1 1 1 1 ...
##   ..$ PRM       : Factor w/ 3 levels "both","n","y": 3 2 2 2 2 2 2 2 2 2 ...
##   ..$ CODelab   : Factor w/ 2 levels "n","y": 1 1 1 1 1 2 2 2 2 2 ...
##   ..$ EXPcomplex: num [1:65] 1 2 1 1 1 1 1 1 1 1 ...
## $ Movima             : tibble [153 x 6] (S3: tbl_df/tbl/data.frame)
##   ..$ PMfixed   : Factor w/ 2 levels "n","y": 1 1 2 2 2 2 2 2 2 2 ...
##   ..$ FREE      : Factor w/ 2 levels "n","y": 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ INTERone  : Factor w/ 2 levels "n","y": 2 2 2 1 1 1 1 1 1 1 ...
##   ..$ PRM       : Factor w/ 3 levels "both","n","y": 3 3 2 1 1 1 1 1 1 1 ...
##   ..$ CODelab   : Factor w/ 2 levels "n","y": 1 1 2 2 1 2 2 2 2 2 ...
##   ..$ EXPcomplex: num [1:153] 1 2 1 1 1 1 1 1 1 1 ...
## $ Puinave            : tibble [99 x 6] (S3: tbl_df/tbl/data.frame)
##   ..$ PMfixed   : Factor w/ 2 levels "n","y": 2 1 2 2 2 2 2 2 2 2 ...
##   ..$ FREE      : Factor w/ 2 levels "n","y": 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ INTERone  : Factor w/ 2 levels "n","y": 2 2 1 1 1 1 1 1 1 1 ...
##   ..$ PRM       : Factor w/ 3 levels "both","n","y": 1 2 2 2 2 2 2 2 2 2 ...
##   ..$ CODelab   : Factor w/ 2 levels "n","y": 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ EXPcomplex: num [1:99] 3 1 1 2 2 1 1 1 1 8 ...
## $ Tariana            : tibble [119 x 6] (S3: tbl_df/tbl/data.frame)
##   ..$ PMfixed   : Factor w/ 2 levels "n","y": 1 1 2 2 2 2 2 2 2 2 ...
##   ..$ FREE      : Factor w/ 2 levels "n","y": 1 1 2 2 2 2 2 2 2 2 ...
##   ..$ INTERone  : Factor w/ 2 levels "n","y": 2 2 2 2 2 2 2 2 2 2 ...
##   ..$ PRM       : Factor w/ 3 levels "both","n","y": 2 2 1 1 1 1 1 1 1 1 ...
##   ..$ CODelab   : Factor w/ 2 levels "n","y": 1 1 2 2 2 2 2 2 2 2 ...
##   ..$ EXPcomplex: num [1:119] 3 5 1 1 1 1 1 1 1 1 ...
```

Creating the (dis)similarity matrices

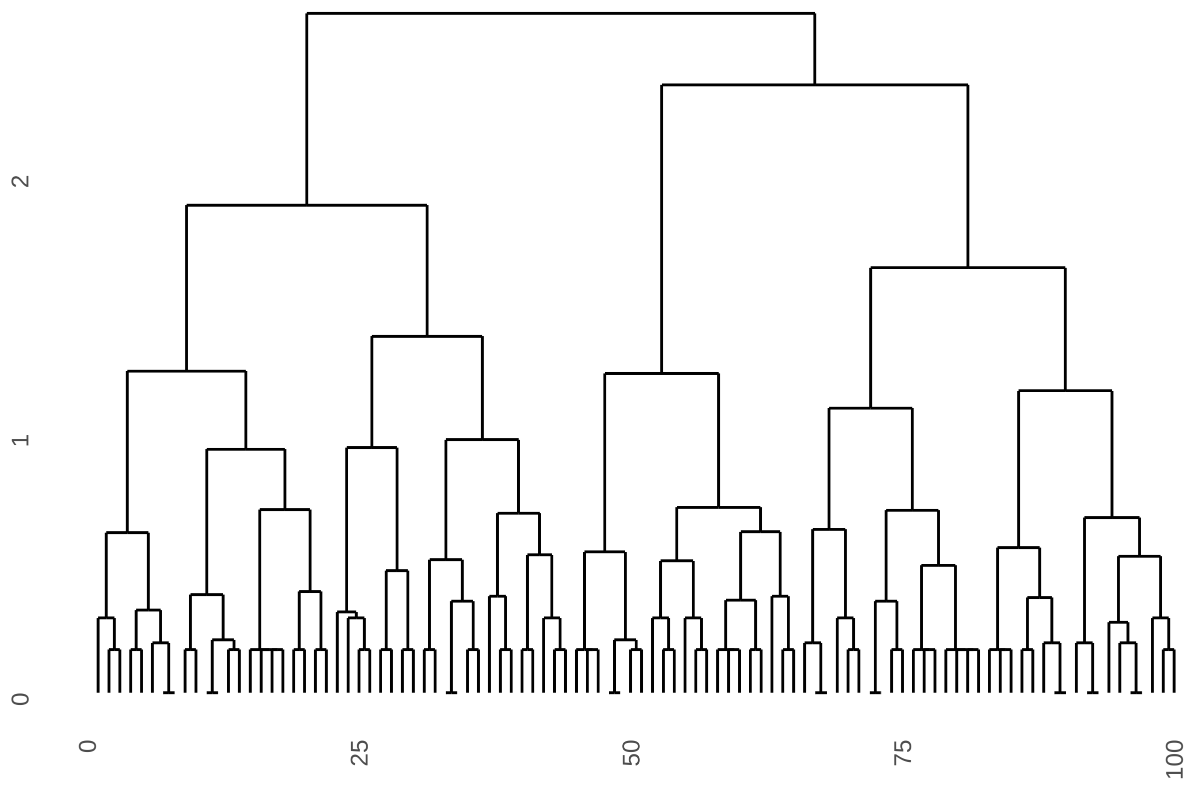
We use the Gower distance appropriate for mixed data types to create the dissimiliarity matrices that will serve as the input for the clustering.

```
db_dist <- lapply(1:length(db_lang), function(d) daisy(db_lang[[d]], metric = "gower"))

# simulated data set
sim_dist <- daisy(sim_db, metric = "gower")
```

Hierarchical clustering per language

We apply hierarchical clustering to the (dis)similarity matrices and plot the results as a dendrogram. We do this furst with the simulated data set and then with the data of the sample.

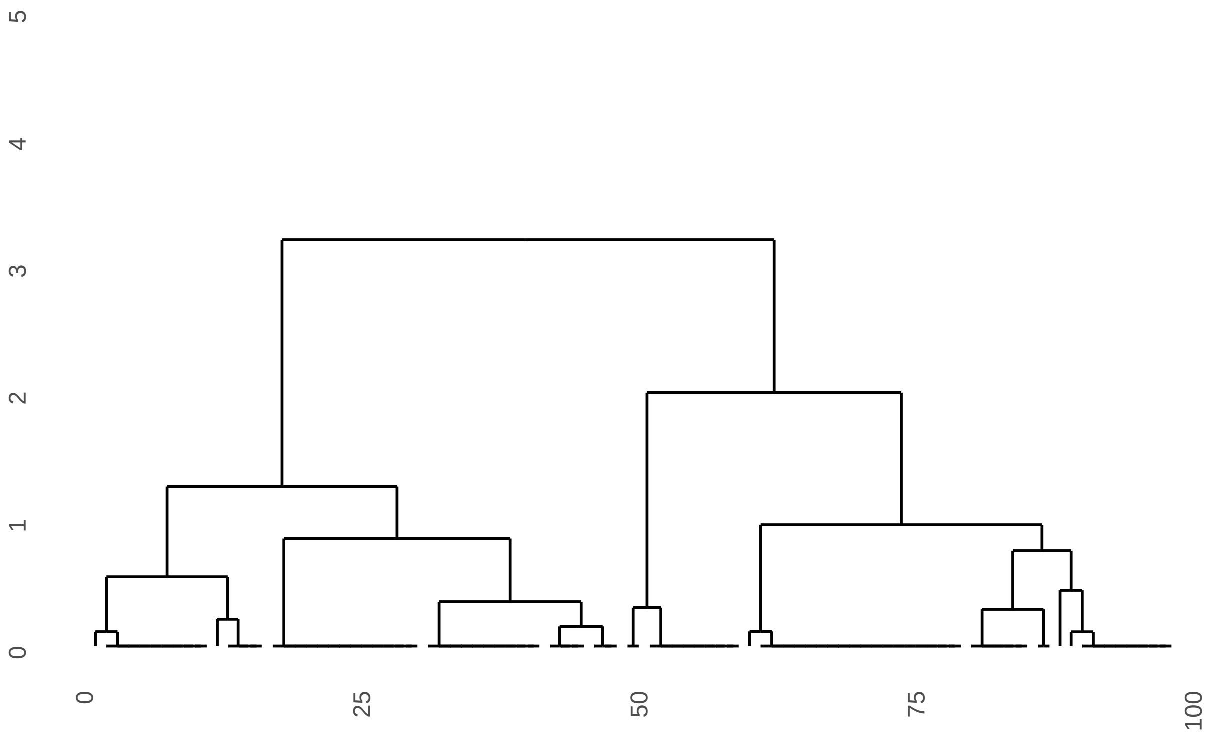
```
# hierarchical clustering
sim_hclust <- hclust(sim_dist, method = "ward.D2")
# plot results
sim_dendro <- ggdendrogram(sim_hclust, leaf_labels = FALSE, labels = FALSE)
sim_dendro
```



```
# hierarchical clustering
db_hclust <- lapply(db_dist, function(c) hclust(c, method = "ward.D2"))
names(db_hclust) <- names(db_lang)
# plot results
db_dendro <- lapply(names(db_hclust), function(p) ggdendrogram(db_hclust[[p]], leaf_labels = FALSE,
    labels = FALSE) + ylim(0, 5) + labs(title = p))
db_dendro
```

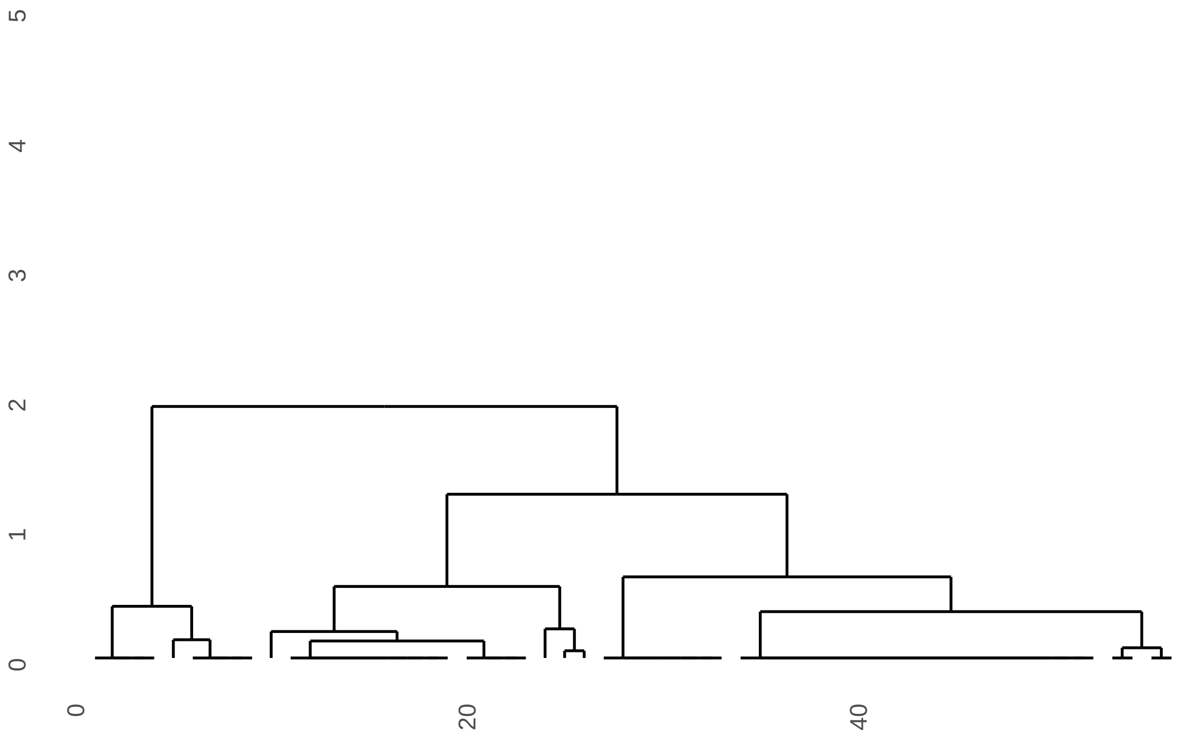
```
## [[1]]
```

AsheninkaPerene

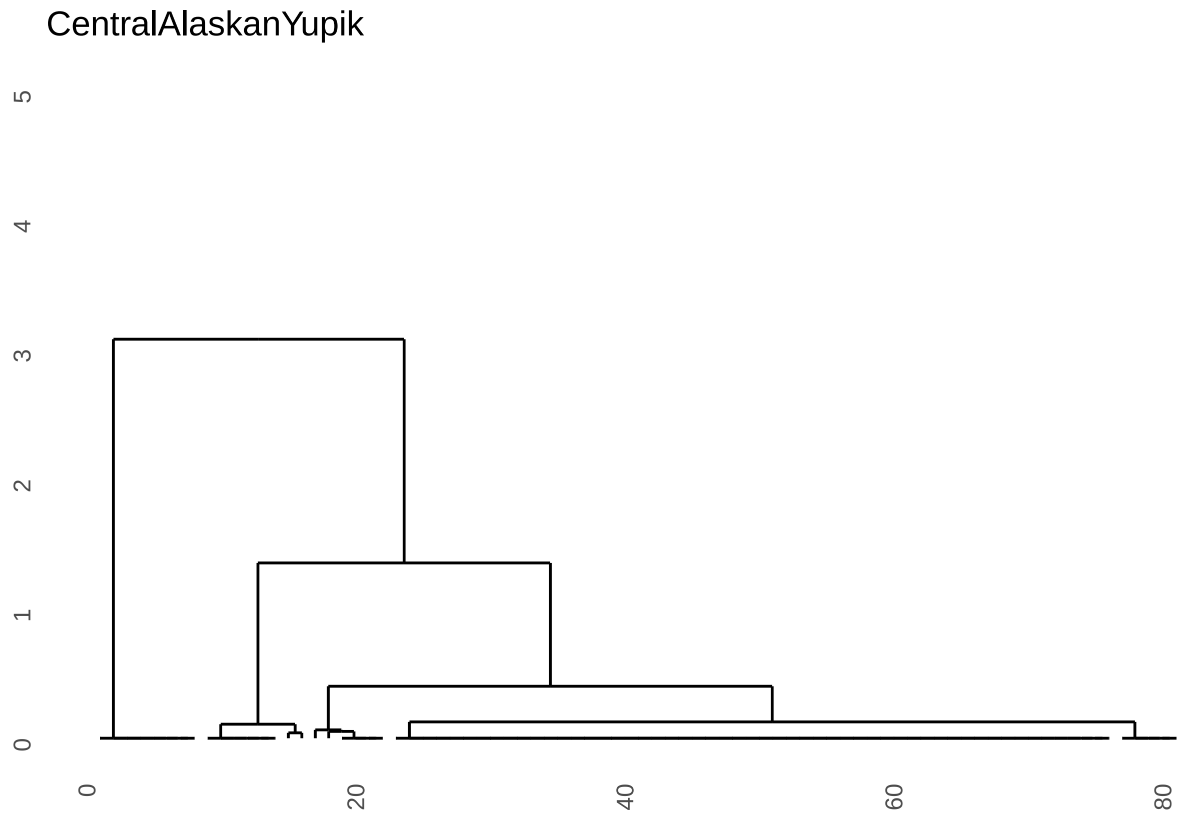


```
##
## [[2]]
```

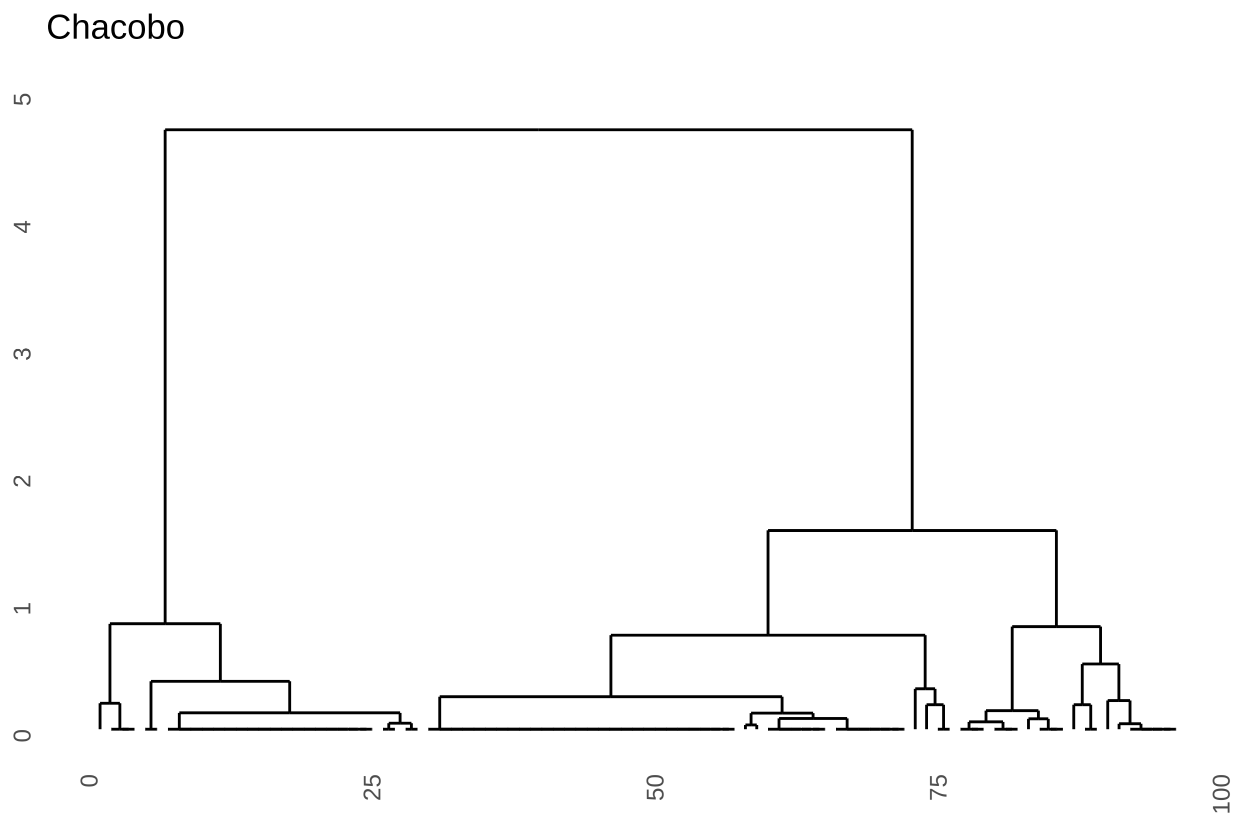
Cavineña



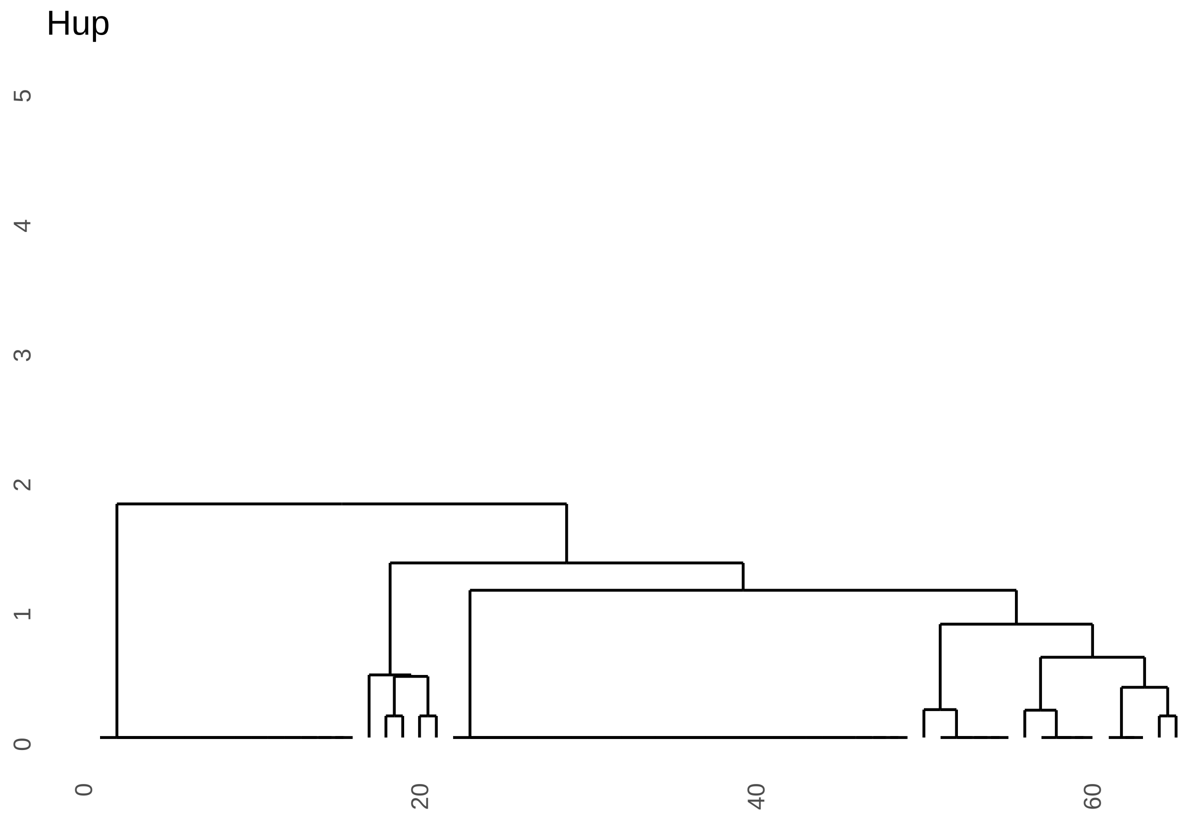
```
##
## [[3]]
```



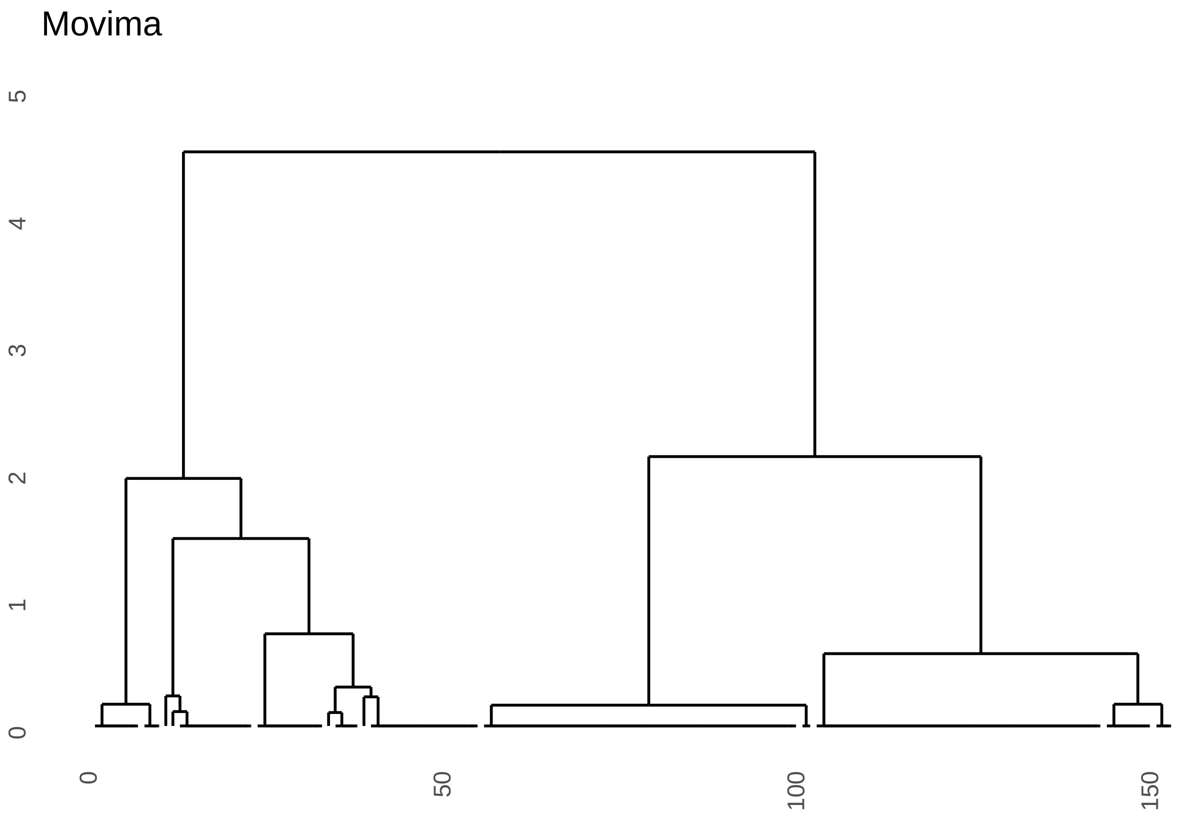
```
##
## [[4]]
```



```
##
## [[5]]
```

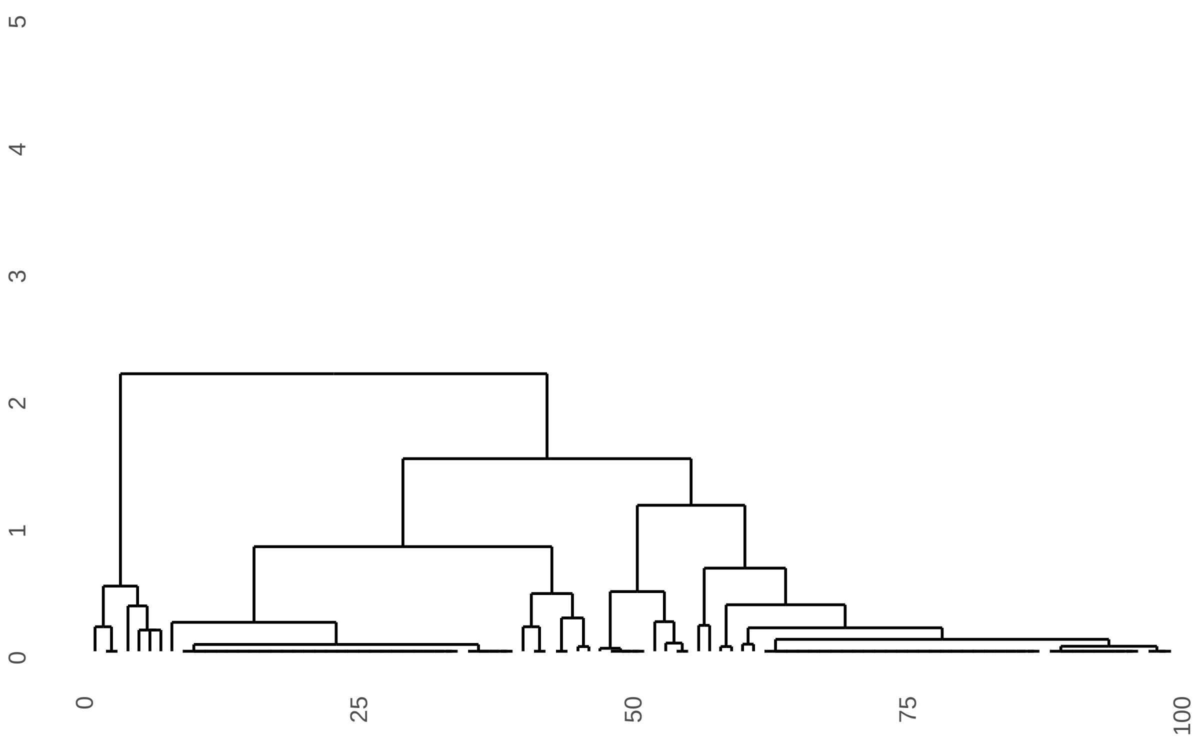


```
##
## [[6]]
```



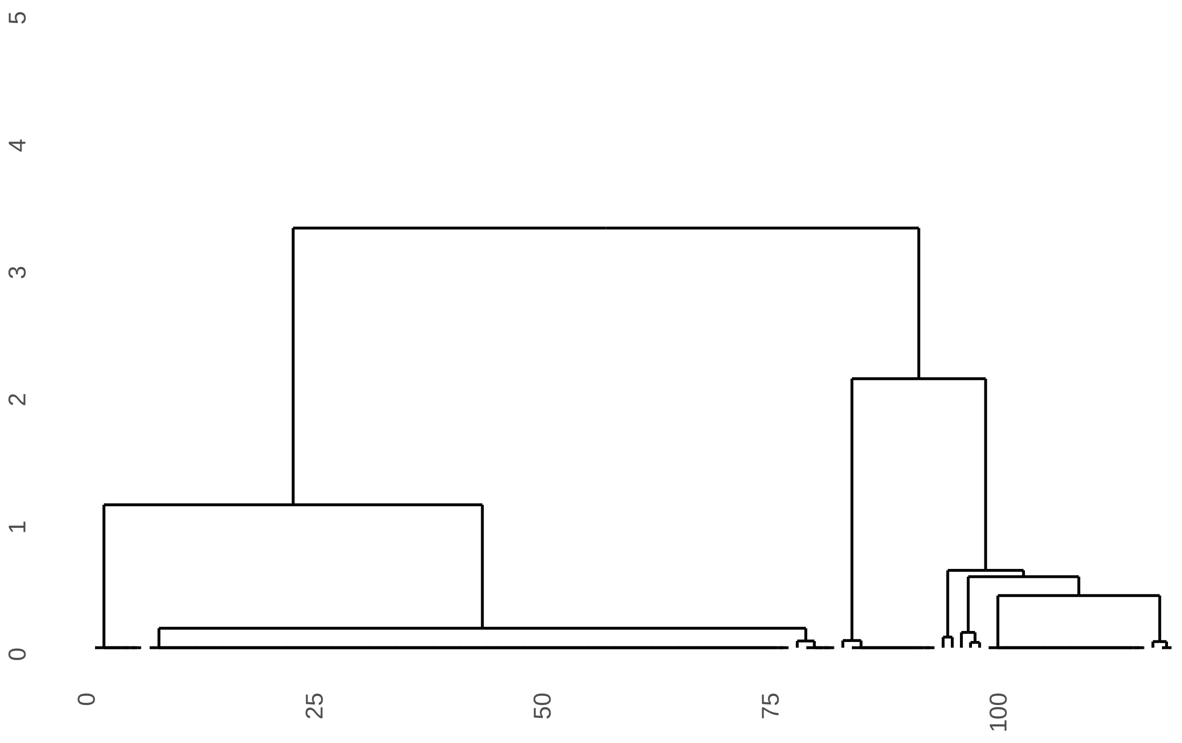
```
##
## [[7]]
```

Puinave



```
##
## [[8]]
```

Tariana



We also calculate the cophenetic coefficient and distance correlation, as well as the height difference between the first and second partition of the cluster. Do do this efficiently, we write a custom function.

```
coph_height <- function(x, y) {
    coph <- cophenetic(x)
    cor <- round(cor(y, coph), 4)
    height <- x$height
    height_diff <- round(height[length(height)] - height[length(height) - 1], 4)
    returnlist <- c(heightdiff = height_diff, coph = cor)
}
```

We apply this to the simulated data set and to the language data.

```
# simulated data
sim_ch <- coph_height(sim_hclust, sim_dist)
sim_ch
```

```
## heightdiff       coph
##     0.2760     0.4955
```

```
# language data
db_ch <- lapply(1:length(db_hclust), function(d) coph_height(db_hclust[[d]], db_dist[[d]]))
names(db_ch) <- names(db_lang)
db_ch
```

```
## $AsheninkaPerene
## heightdiff       coph
##     1.2021     0.7104
##
## $Cavineña
## heightdiff       coph
##     0.6766     0.8470
##
## $CentralAlaskanYupik
## heightdiff       coph
##     1.7257     0.9865
##
## $Chacobo
## heightdiff       coph
##     3.1473     0.9219
##
## $Hup
## heightdiff       coph
##     0.4552     0.7726
##
## $Movima
## heightdiff       coph
##     2.3942     0.8290
##
## $Puinave
## heightdiff       coph
##     0.6677     0.7728
##
## $Tariana
## heightdiff       coph
##     1.1836     0.9353
```

Comparison with simulated languages

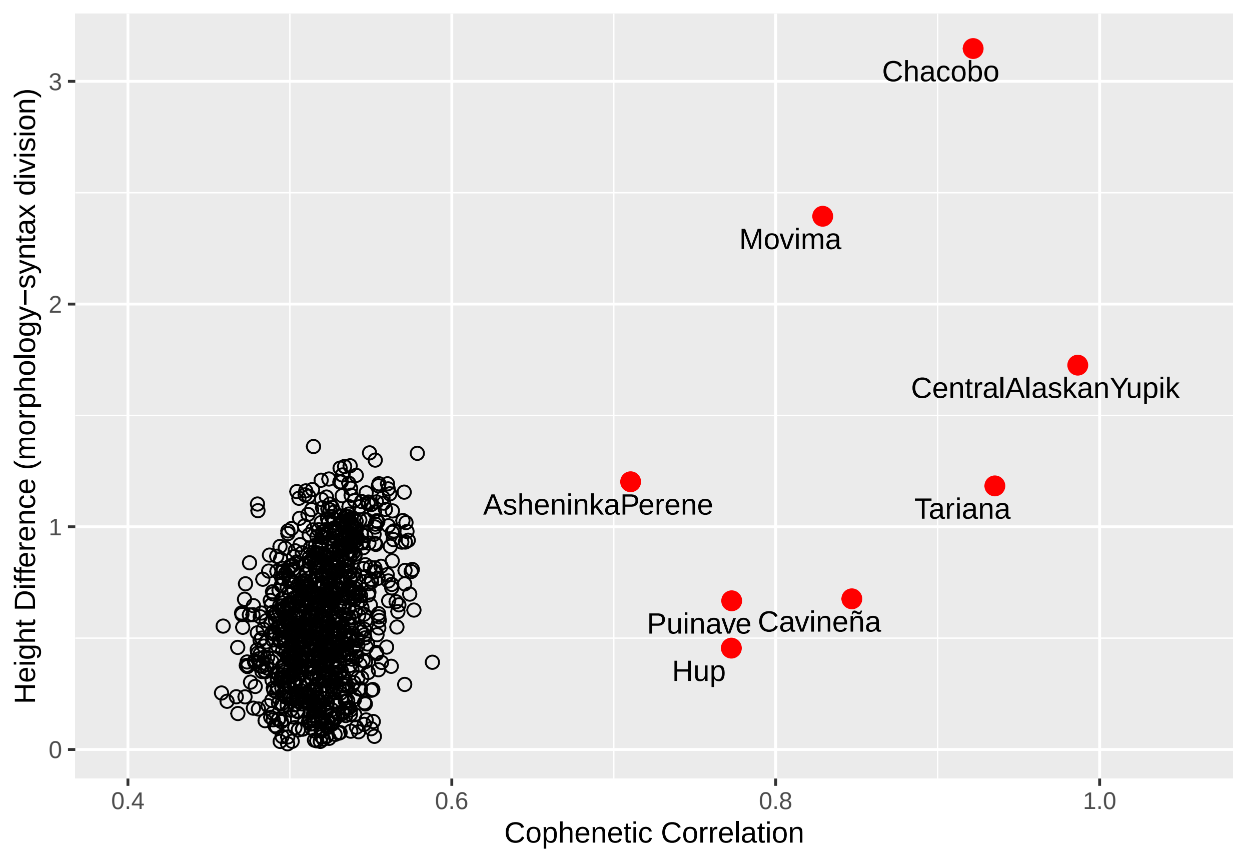
An anonymous reviewer suggests that the languages of our sample should better be compared to many simulated languages rather than one. Thus we simulate 1000 languages, apply hierarchical clustering and calculate cophenetic distance and height difference of the first partition

```r
# set up variables and loop over them
sim1000 <- lapply(1:1000, function(s) {
    FREE <- as.character(sample(c("y", "n"), 100, replace = T))
    INTERone <- as.character(sample(c("y", "n"), 100, replace = T))
    CODelab <- as.character(sample(c("y", "n"), 100, replace = T))
    PMfixed <- as.character(sample(c("y", "n"), 100, replace = T))
    PRM <- as.character(sample(c("y", "both", "n"), 100, replace = T))
    EXPcomplex <- sample(1:10, 100, replace = T, prob = 10:1)
    # combine to df
    sim_db <- as_tibble(cbind(FREE, INTERone, CODelab, PMfixed, PRM, EXPcomplex)) %>%
        mutate_if(is.character, as.factor)
    # compute distances
    sim_dist <- daisy(sim_db, metric = "gower")
    # apply hierarchical clustering
    sim_hclust <- hclust(sim_dist, method = "ward.D2")
    # compute height and cophenetic distances
    sim_ch <- coph_height(sim_hclust, sim_dist)
})
# convert to df
sim1000_df <- as.data.frame(do.call(rbind, sim1000))
# means of simulated languages
summary(sim1000_df)
```

```
##    heightdiff          coph       
##  Min.   :0.0257   Min.   :0.4578  
##  1st Qu.:0.3548   1st Qu.:0.5048  
##  Median :0.5457   Median :0.5195  
##  Mean   :0.5754   Mean   :0.5195  
##  3rd Qu.:0.7884   3rd Qu.:0.5331  
##  Max.   :1.3604   Max.   :0.5880
```

We then plot the cophenetic distances and height differences of the simulated data and the languages of our sample against each other.

```r
# convert language list into df and assign rownames as column
db_ch_df <- as.data.frame(do.call(rbind, db_ch))
db_ch_df <- rownames_to_column(db_ch_df, "Language")

# plot cophenetic distance and height difference
sim_comp <- ggplot(sim1000_df, aes(x = coph, y = heightdiff)) + geom_point(size = 2,
    shape = 1) + geom_point(data = db_ch_df, aes(x = coph, y = heightdiff), color = "red",
    size = 3) + geom_text(data = db_ch_df, aes(x = coph, y = heightdiff, label = Language),
    nudge_x = -0.02, nudge_y = -0.1) + xlim(0.4, 1.05) + labs(x = "Cophenetic Correlation",
    y = "Height Difference (morphology-syntax division)")
sim_comp
```



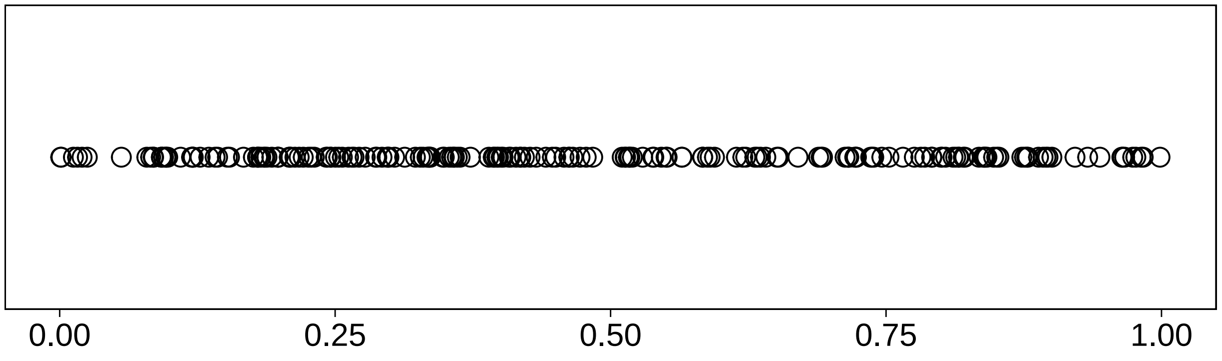
Reproducing Haspelmath's cluster graphic

Below we include the code we used to reproduce the graphics from Haspelmath 2011:64.

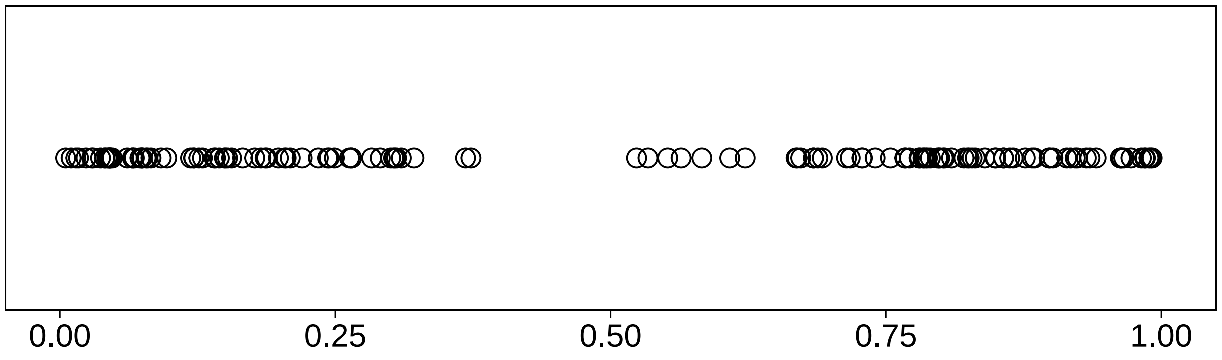
```
# reproducing Haspelmath's graphic set up vectors
noclust <- runif(200, min = 0, max = 1)
clust1 <- rnorm(100, mean = 0.09, sd = 0.15)
clust2 <- rnorm(100, mean = 0.91, sd = 0.15)
# combine into df
noclust_db <- as_tibble(noclust)
clust_db <- as_tibble(clust1, clust2)
# make scatterplot
noclust_plot <- ggplot(noclust_db) + geom_point(aes(x = noclust, y = 1), shape = 1,
    size = 3) + theme_linedraw() + theme(axis.title = element_blank(), axis.text.x = element_text(size =
    axis.text.y = element_blank(), axis.ticks.y = element_blank(), panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(), plot.title = element_text(size = 16)) + xlim(0,
    1) + ylim(0.7, 1.3) + ggtitle("Random distribution")

clust_plot <- ggplot(clust_db) + geom_point(aes(x = clust1, y = 1), shape = 1, size = 3,
    inherit.aes = FALSE) + geom_point(aes(x = clust2, y = 1), shape = 1, size = 3,
    inherit.aes = FALSE) + theme_linedraw() + theme(axis.title = element_blank(),
    axis.text.x = element_text(size = 12), axis.text.y = element_blank(), axis.ticks.y = element_blank(
    panel.grid.major = element_blank(), panel.grid.minor = element_blank(), plot.title = element_text(s
    xlim(0, 1) + ylim(0.7, 1.3) + ggtitle("Two 'clusters' distribution")

hm_plots <- ggarrange(noclust_plot, clust_plot, ncol = 1, nrow = 2)
```

```
hm_plots
```

Random distribution



Two 'clusters' distribution


0.00
0.25
0.50
0.75
1.00