R code 2023-02-07 (Clustering and using simulations for inference)

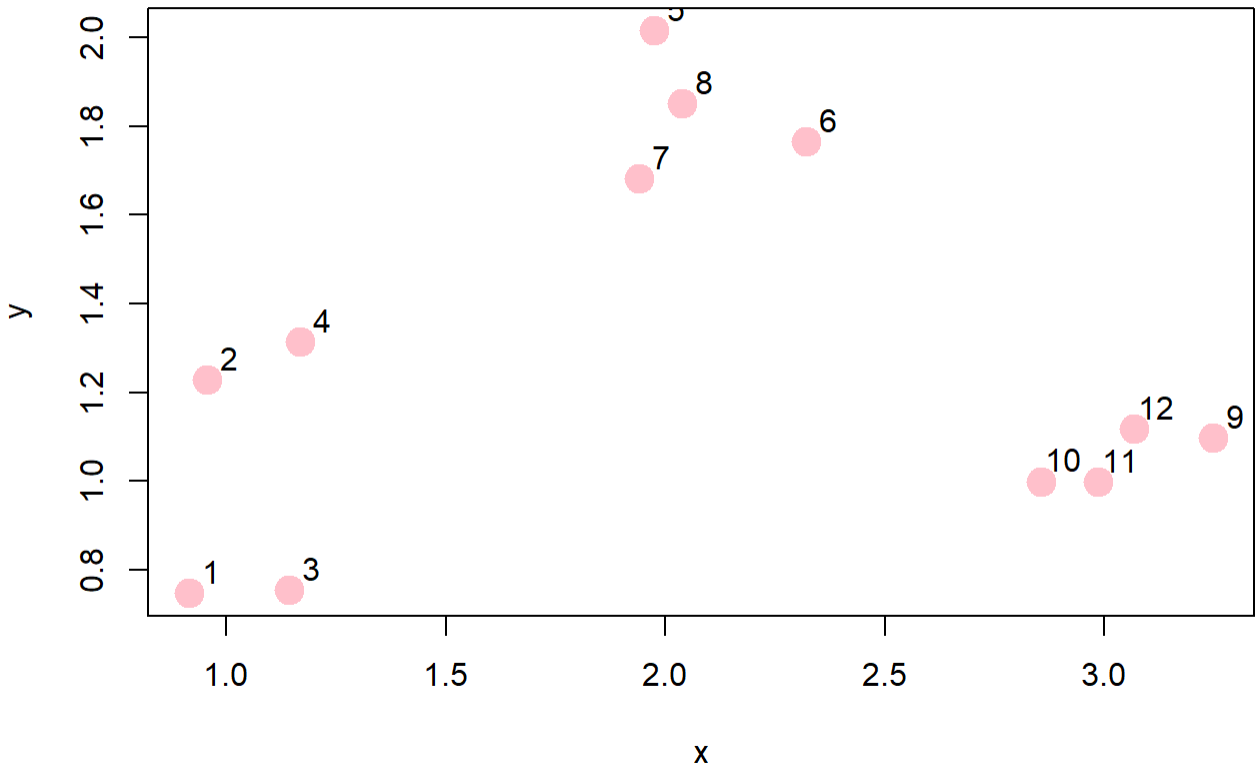
Adam Tallman

2023-02-06

Hierarchical cluster

Let's simulate some data.

```
set.seed(4321)
x <- rnorm(12, rep(1:3, each = 4), 0.2)
y <- rnorm(12, rep(c(1, 2, 1), each = 4), 0.2)
plot(x, y, col = "pink", pch = 19, cex = 2)
text(x + 0.05, y + 0.05, labels = as.character(1:12))
```



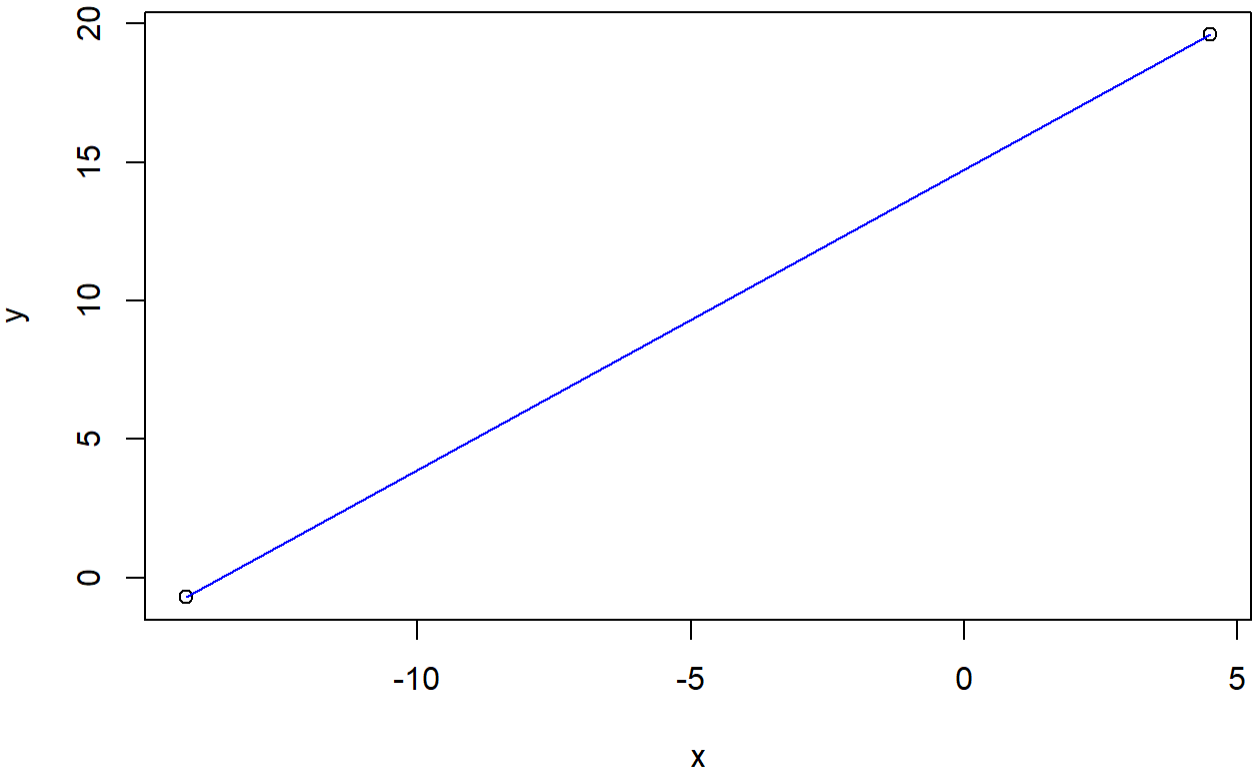
Measuring the distance between all the points

Clustering methods start with calculating the distance between every point - here's different methods for doing this. But here's some code for doing it.

Euclidean distance is probably familiar to everyone. It is just the distance on a cartesian plane.

\[\large d_{euc}(x,y) = \sqrt{\sum_{i=1}^n(x_i - y_i)^2 } \] If you had two variables the distance is just a line drawn between them.

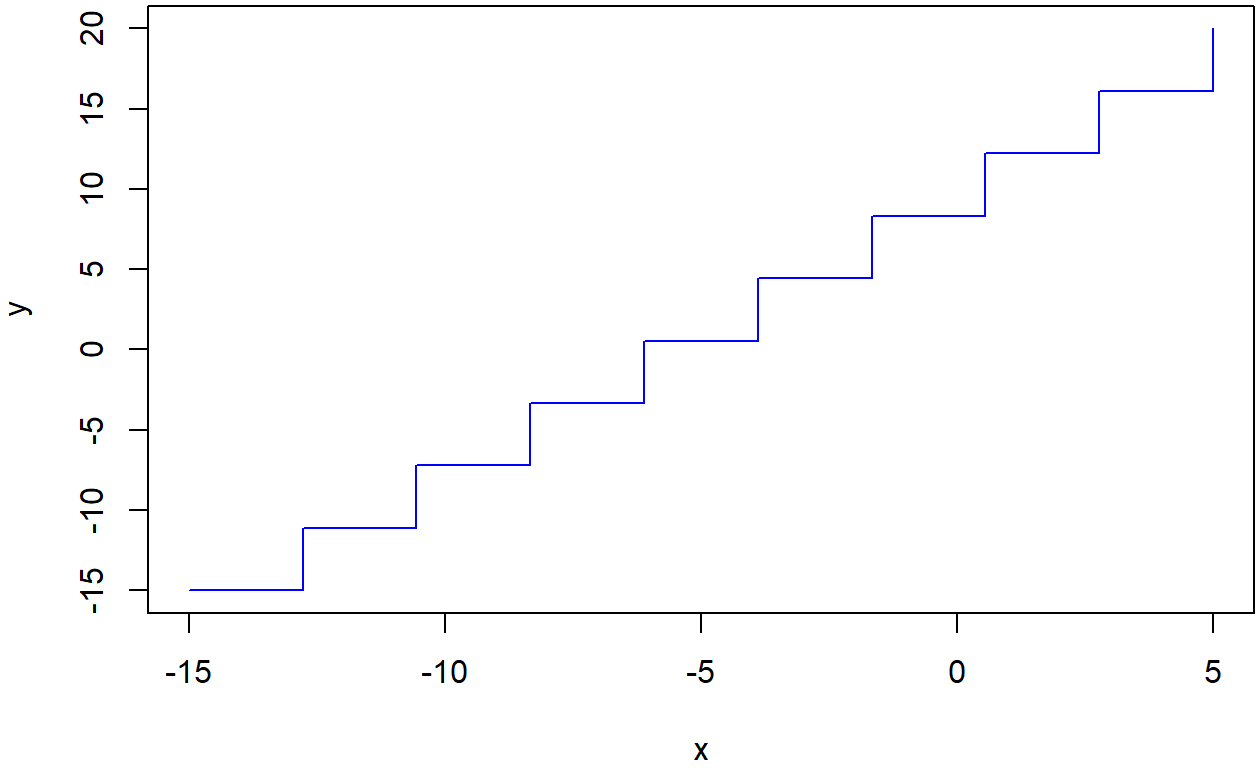
```
set.seed(1122)
x <- rnorm(2)*10
y <- rnorm(2)*10
plot(x,y)
segments(x[1], y[1], x[2], y[2], col= 'blue')
```



Manhattan distance in based

\[\large d_{man}(x,y) = \sum_{i=1}^n|(x_i-y_i)| \]

```
f = function(x) {
  floor(x)
}
x = seq(-15, 5, length.out = 10)
y = seq(-15, 20, length.out = 10)
plot(x, y, type = 's',col="blue")
```



There's also a lot of other measures. In most linguistic application we are dealing with categorical data. "Gower" is the most used method now because it can handle mixed data, where some data are categorical and some are continuous. *p* is the number of features. For each feature *k* = 1, ….,*p* a score s is provided *s*.

\[d_{Gower}(x_i,x_j) = \frac{\sum^p_{k=1}s_{ijk}}{p} \]

```
set.seed(4321)
x <- rnorm(12, rep(1:3, each = 4), 0.2)
y <- rnorm(12, rep(c(1, 2, 1), each = 4), 0.2)
dataFrame <- data.frame(x=x, y=y)
dist(dataFrame, method="euclidean")
```

```
##           1         2         3         4         5         6         7
## 2  0.4818060
## 3  0.2290071 0.5083847
## 4  0.6210186 0.2300065 0.5595680
## 5  1.6516473 1.2874484 1.5084603 1.0675884
## 6  1.7363507 1.4683435 1.5515341 1.2383578 0.4279533
## 7  1.3877613 1.0850453 1.2223180 0.8553386 0.3340584 0.3901532
## 8  1.5749957 1.2500236 1.4145687 1.0225625 0.1765242 0.2953299 0.1949666
## 9  2.3594401 2.2966216 2.1320837 2.0912494 1.5701360 1.1421832 1.4327423
## 10 1.9578445 1.9146662 1.7299970 1.7171607 1.3448566 0.9336432 1.1423308
## 11 2.0869817 2.0442189 1.8589258 1.8455955 1.4345148 1.0147559 1.2498504
## 12 2.1859159 2.1164159 1.9592779 1.9106631 1.4145273 0.9877883 1.2612721
##            8         9        10        11
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9  1.4247104
## 10 1.1798333 0.4037896
## 11 1.2743264 0.2795230 0.1302977
## 12 1.2632629 0.1806167 0.2437852 0.1458997
```

```
rdistxy <- as.matrix(dist(dataFrame, method="euclidean"))
```

We have to remove the diagonal from consideration.

```
diag(rdistxy)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12
##  0  0  0  0  0  0  0  0  0  0  0  0
```

```
diag(rdistxy) <- diag(rdistxy) + 100000
```

The (bottom-up) hierarchical cluster algorithm

It looks for the closest data points first.

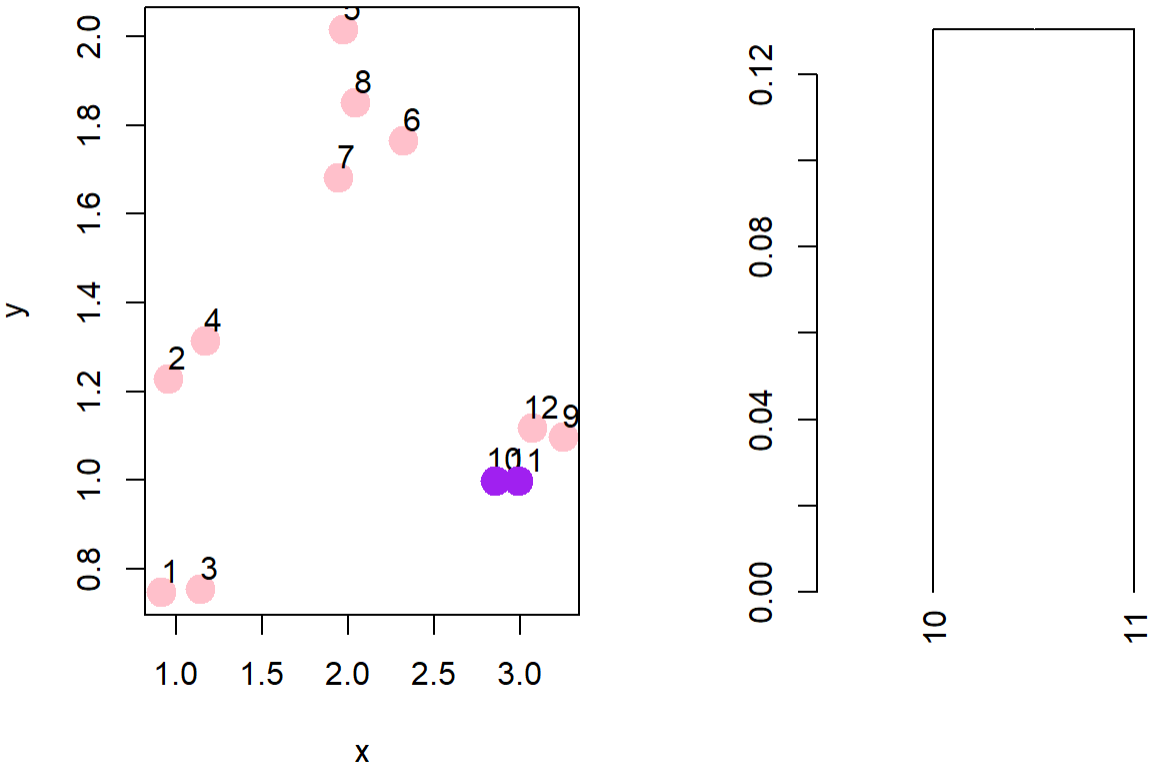
```
ind <- which(rdistxy == min(rdistxy), arr.ind = TRUE)
ind
```

```
##    row col
## 11  11  10
## 10  10  11
```

```
par(mfrow = c(1, 2))
```

The following plot shows you how the hierarchical algorithm makes its first cluster.

```
par(mfrow = c(1, 2))
plot(x, y, col = "pink", pch = 19, cex = 2)
text(x + 0.05, y + 0.05, labels = as.character(1:12))
points(x[ind[1, ]], y[ind[1, ]], col = "purple", pch = 19, cex = 2)
hcluster <- dist(dataFrame)
dendro <- as.dendrogram(hclust(hcluster))
plot(cut(dendro, h = 0.4)$lower[[3]])
```



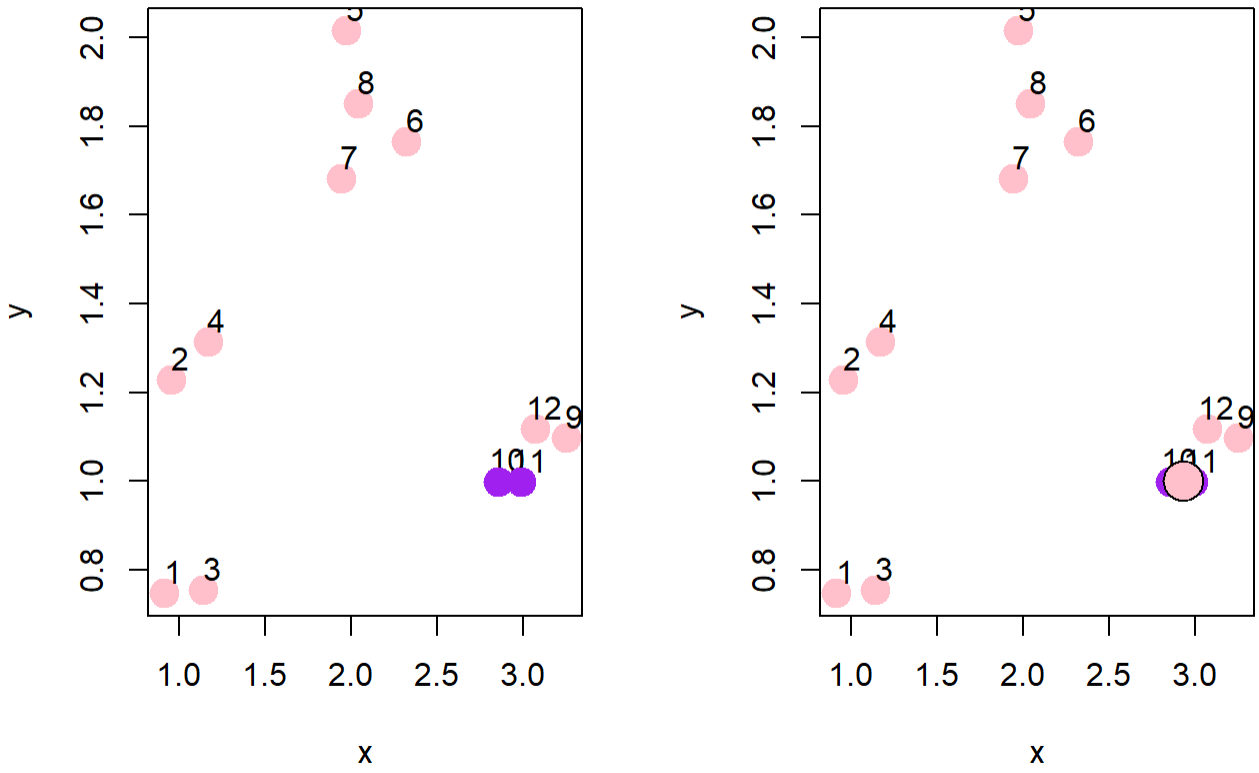
Then it groups 10 and 11 into the same point - the height (0.12) is the distance between the combined points. After this the algorithm would find the next closest points.

```
nextmin <- rdistxy[order(rdistxy)][7]
ind2 <- which(rdistxy == nextmin,arr.ind=TRUE)
ind2
```

```
##    row col
## 12  12   9
## 9    9  12
```

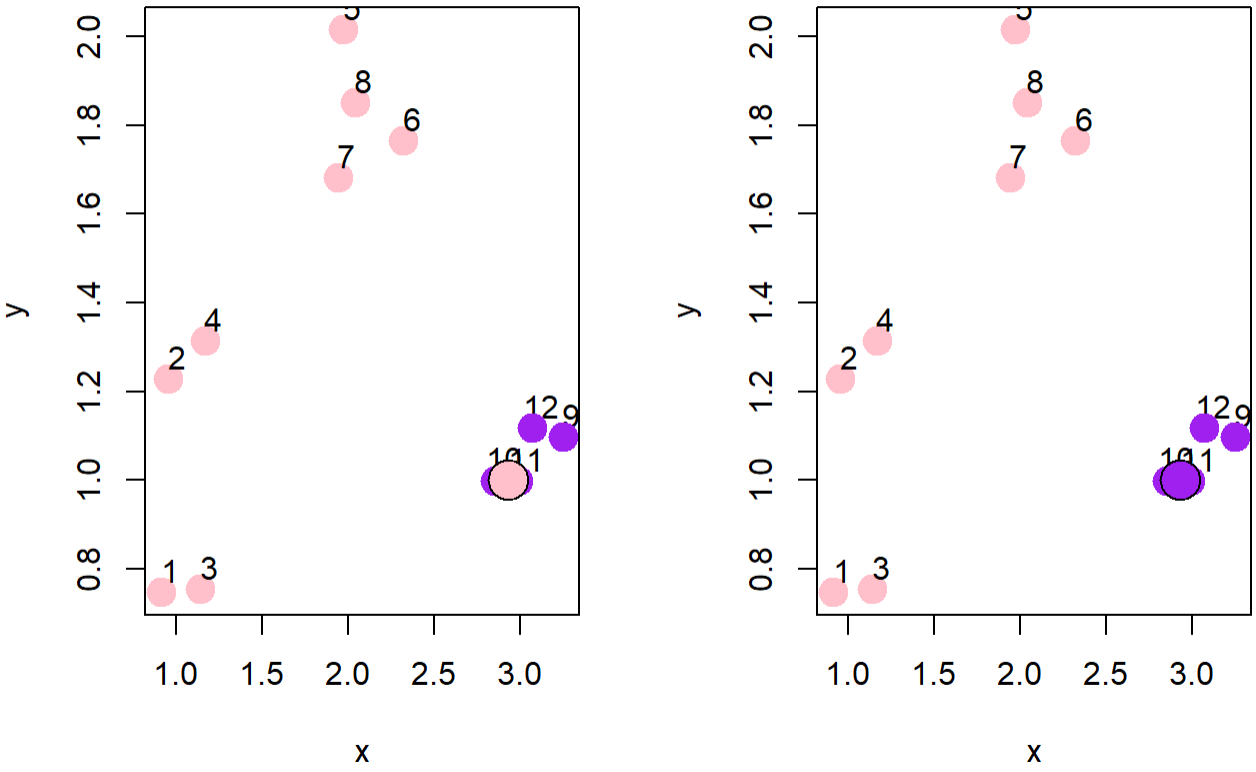
``Agglomeration" means grouping two points into one and then we find the next point.

```
par(mfrow = c(1, 2))
plot(x, y, col = "pink", pch = 19, cex = 2)
text(x + 0.05, y + 0.05, labels = as.character(1:12))
points(x[ind[1, ]], y[ind[1, ]], col = "purple", pch = 19, cex = 2)

plot(x, y, col = "pink", pch = 19, cex = 2)
text(x + 0.05, y + 0.05, labels = as.character(1:12))
points(x[ind[1, ]], y[ind[1, ]], col = "purple", pch = 19, cex = 2)
symbols(x=c(2.93), y=c(1), circles=0.12, add=T, inches=F, pch=20, bg="pink")
```

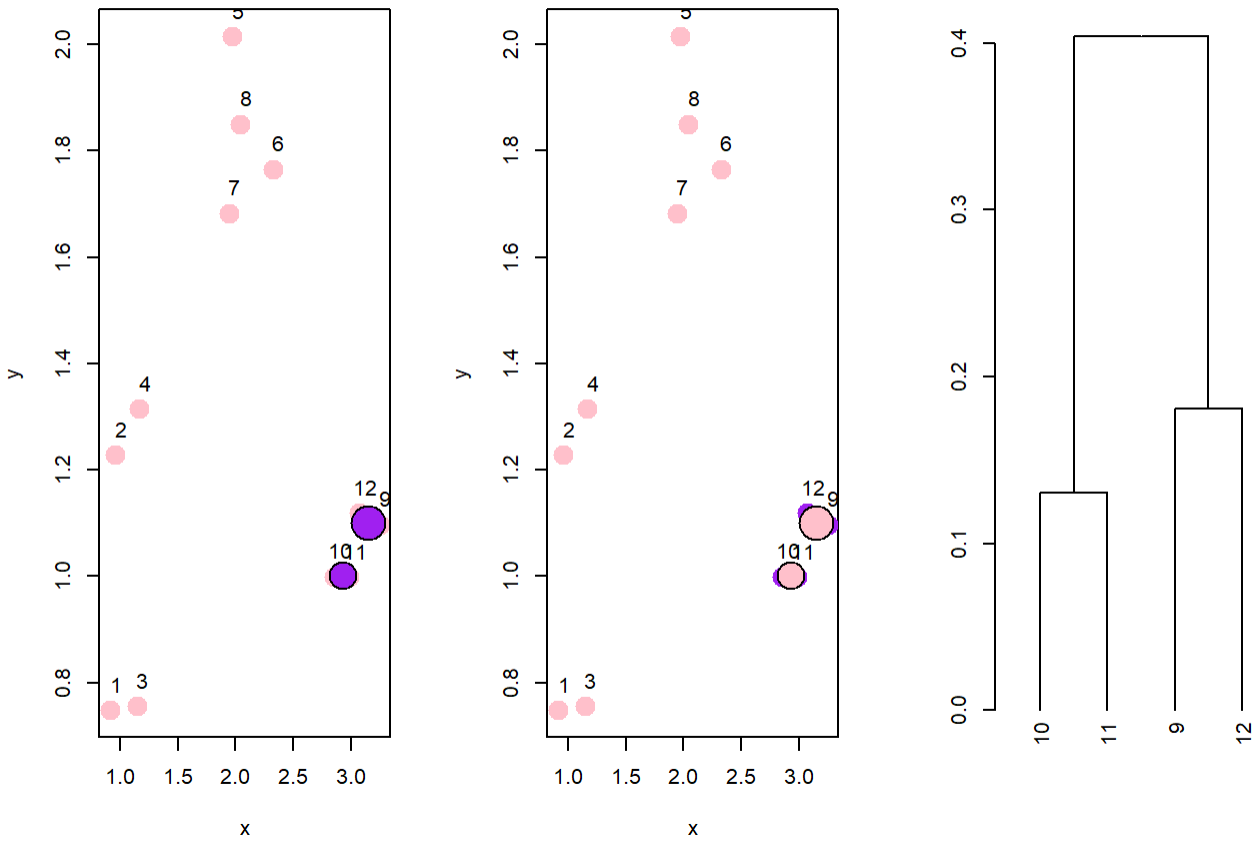


So the next agglomeration would look like this:

```
par(mfrow = c(1, 2))

plot(x, y, col = "pink", pch = 19, cex = 2)
text(x + 0.05, y + 0.05, labels = as.character(1:12))
points(x[ind[1, ]], y[ind[1, ]], col = "purple", pch = 19, cex = 2)
symbols(x=c(2.93), y=c(1), circles=0.12, add=T, inches=F, pch=20, bg="pink")
points(x[ind2[1, ]], y[ind2[1, ]], col = "purple", pch = 19, bg = "purple", cex=2)

plot(x, y, col = "pink", pch = 19, cex = 2)
text(x + 0.05, y + 0.05, labels = as.character(1:12))
points(x[ind[1, ]], y[ind[1, ]], col = "purple", pch = 19, cex = 2)
symbols(x=c(2.93), y=c(1), circles=0.12, add=T, inches=F, pch=20, bg="purple")
points(x[ind2[1, ]], y[ind2[1, ]], col = "purple", pch = 19, bg = "purple", cex=2)
```

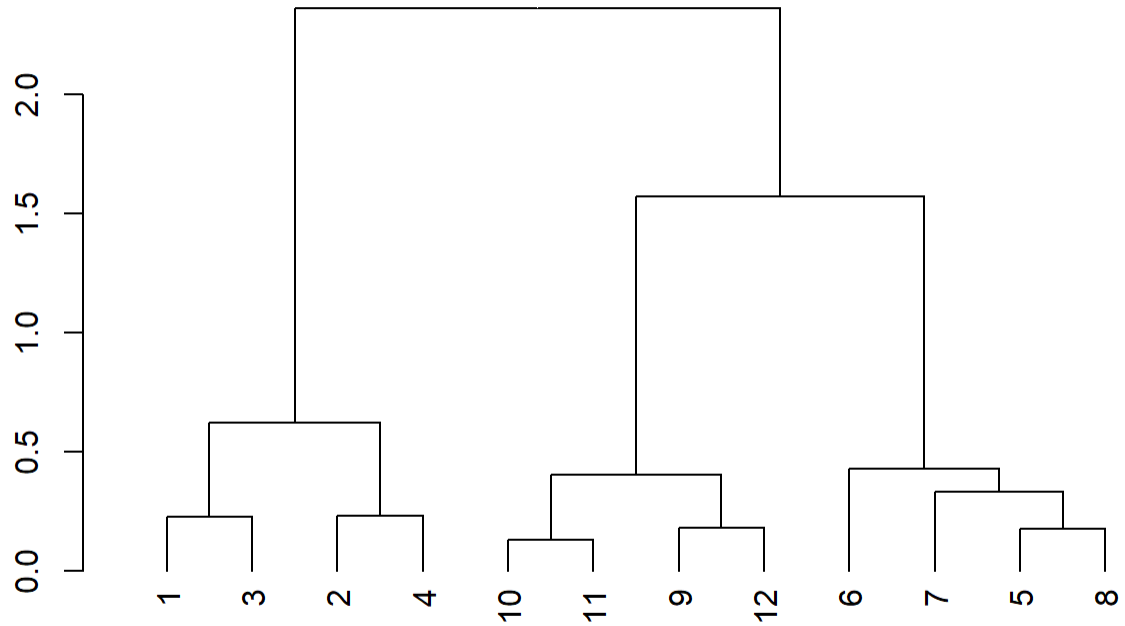


```
par(mfrow = c(1, 3))

plot(x, y, col = "pink", pch = 19, cex = 2)
text(x + 0.05, y + 0.05, labels = as.character(1:12))
symbols(x=c(2.93), y=c(1), circles=0.12, add=T, inches=F, pch=20, bg="purple")
symbols(x=c(3.15), y=c(1.1), circles=0.15, add=T, inches=F, pch=20, bg="purple", cex=2)

plot(x, y, col = "pink", pch = 19, cex = 2)
text(x + 0.05, y + 0.05, labels = as.character(1:12))
points(x[ind[1, ]], y[ind[1, ]], col = "purple", pch = 19, cex = 2)
symbols(x=c(2.93), y=c(1), circles=0.12, add=T, inches=F, pch=20, bg="pink")
points(x[ind2[1, ]], y[ind2[1, ]], col = "purple", pch = 19, bg = "purple", cex=2)
symbols(x=c(3.15), y=c(1.1), circles=0.15, add=T, inches=F, pch=20, bg="pink", cex=2)

hcluster <- dist(dataFrame)
clusters <- hclust(hcluster)

dendro <- as.dendrogram(clusters)
plot(cut(dendro, h = 0.5)$lower[[3]])
```



It keeps going until it clusters all of the points.

```
par(mfrow = c(1, 1))

plot(dendro)
```



Clause-linkage and the division between coordination and subordination

Although you are taught in traditional grammar class that there is some sort of discrete division between coordination and subordination, in fact, there seem to be lots of intermediate cases. There are no sufficient and necessary criteria for distinguishing between coordination and subordination.

Do coordination and subordination cluster into groups according to different criteria for distinguishing the two clause types?

```
cl <- read.csv("/Users/Adan Tallman/Desktop/clause_linkage.csv", header=TRUE)
```

```
head(cl)
```

Language	Label	ILL_scope	T_scope	Finiteness	ILL_mark	T_mark
<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>

	Language <chr>	Label <chr>	ILL_scope <chr>	T_scope <chr>	Finiteness <chr>	ILL_mark <chr>	T_mark <chr>
1	Amele	BUT	local	local	finite	banned	ok
2	Amele	CHAIN	conjunct	conjunct	nonfinite	banned	banned
3	Amele	COND	disjunct	extensible	any	ok	ok
4	Amele	OR	conjunct	local	finite	ok	ok
5	Amele	PURP	disjunct	extensible	any	banned	ok
6	Belhare	AND	constraint-free	local	finite	ok	ok

6 rows | 1-8 of 14 columns

We are going to name the variables based on the language and their conjunction gloss and/or construction name

```
cl <- cl %>% mutate_if(is_character, as.factor)
cl<- cl%>%unite("Name", 1:2, remove=TRUE)
head(cl)
```

	Name <chr>	ILL_scope <fct>	T_scope <fct>	Finiteness <fct>	ILL_mark <fct>	T_mark <fct>
1	Amele _BUT	local	local	finite	banned	ok
2	Amele _CHAIN	conjunct	conjunct	nonfinite	banned	banned
3	Amele _COND	disjunct	extensible	any	ok	ok
4	Amele _OR	conjunct	local	finite	ok	ok
5	Amele _PURP	disjunct	extensible	any	banned	ok
6	Belhare _AND	constraint-free	local	finite	ok	ok

6 rows | 1-7 of 13 columns

You need to make sure these are the row names.

```
rownames(cl) <- cl$Name
cl <- cl %>% mutate_if(is_character, as.factor)
head(cl)
```

	Name <fct>	ILL_scope <fct>	T_scope <fct>	Finiteness <fct>	ILL_m… <fct>	T_m… <fct>
Amele _BUT	Amele _BUT	local	local	finite	banned	ok
Amele _CHAIN	Amele _CHAIN	conjunct	conjunct	nonfinite	banned	banned
Amele _COND	Amele _COND	disjunct	extensible	any	ok	ok

	Name <fct>	ILL_scope <fct>	T_scope <fct>	Finiteness <fct>	ILL_m… <fct>	T_m… <fct>
Amele _OR	Amele _OR	conjunct	local	finite	ok	ok
Amele _PURP	Amele _PURP	disjunct	extensible	any	banned	ok
Belhare _AND	Belhare _AND	constraint-free	local	finite	ok	ok

6 rows | 1-7 of 13 columns

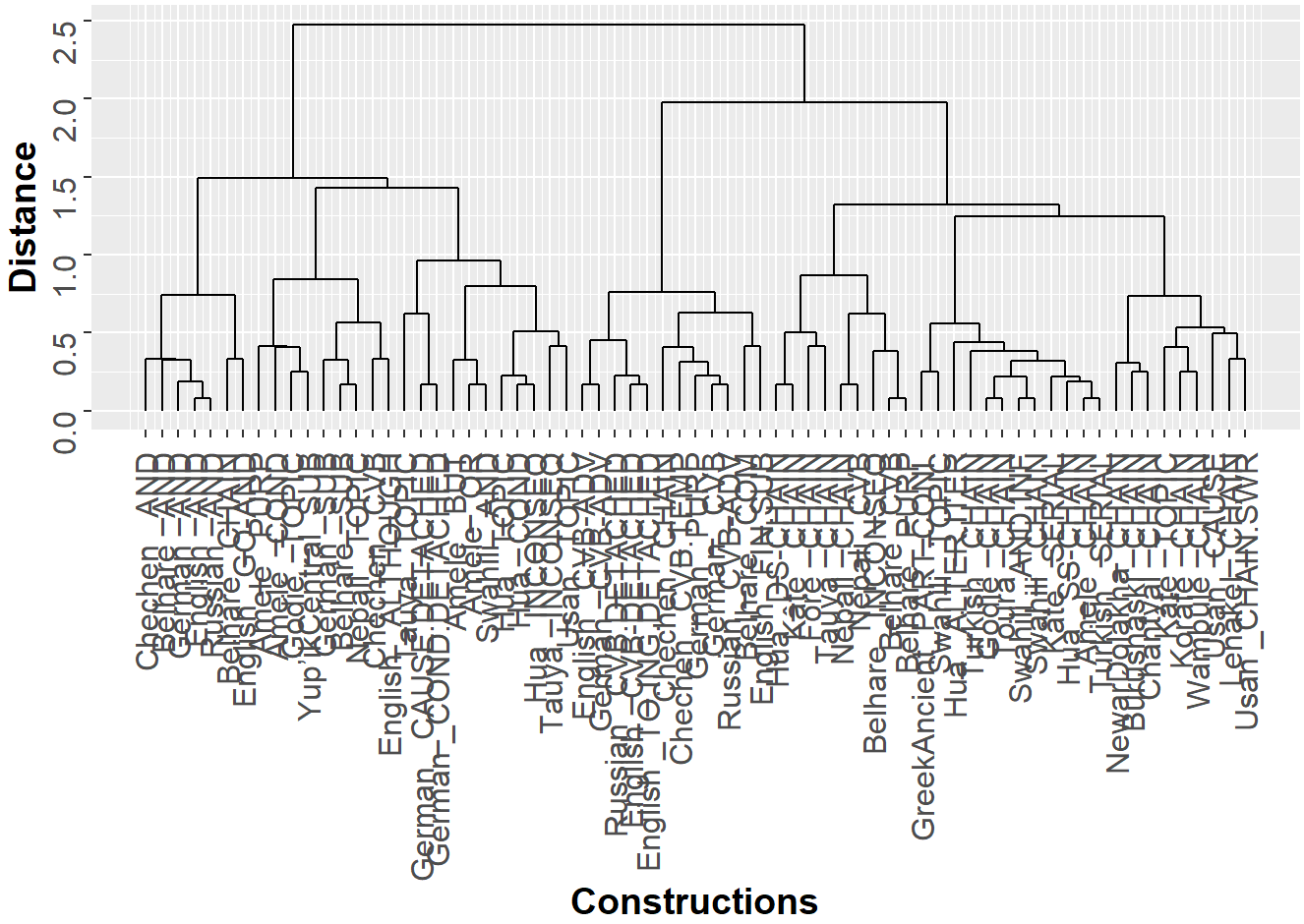
Then we create a distance matrix. We'll have to use gower, because we have categorical data.

```
db.dist <- daisy(cl, metric="gower")
```

Create the dendrogram

```
db.hclust<-hclust(db.dist, method="ward.D2")
db_dendro <- ggdendrogram(db.hclust, rotate = FALSE, theme_dendro = FALSE)+theme(axis.text=el
ement_text(size=12),
        axis.title=element_text(size=14,face="bold"))+
        xlab("Constructions")+
        ylab("Distance")
```

```
db_dendro
```



Using a random forest on the hierarchical cluster

How do we know which variables are the most important? Well we could use the hierarchical cluster to classify elements into group 1 and 2 and then use classification trees.

There's a function called cutree() that can do this for us.

```
cutree(db.hclust, k = 2)
```

```
##              Amele _BUT             Amele _CHAIN              Amele _COND
##                       1                        2                       1
##               Amele _OR              Amele _PURP             Belhare _AND
##                       1                        1                       1
##           Belhare _CHAIN             Belhare _COM             Belhare _CVB
##                       1                        2                       2
##        Belhare _INCONSEQ            Belhare _PURP             Belhare _SUB
##                       2                        2                       1
##        Burúshaski _CHAIN          Chantyal _CHAIN             Chechen _AND
##                       2                        2                       1
##          Chechen _CHAIN             Chechen _CVB         Chechen _CVB.TEMP
##                       2                        1                       2
##        English _ALTHOUGH             English _AND          English _CVB-ADV
##                       1                        1                       2
##    English _CVB-DETACHED          English _FIN.SUB           English _GO-AND
##                       2                        2                       1
## English _TO.ING.DETACHED              Fore _CHAIN              German _AND
##                       2                        2                       1
##   German _CAUSE.DETACHED     German _COND.DETACHED              German _CVB
##                       1                        1                       2
##          German _CVB-ADV             German _PURP             German _SUB
##                       2                        2                       1
##             Godié _CHAIN             Godié _TOPIC   GreekAncient_PART.CONI.
##                       2                        1                       2
##          Hua _ALTER.ITER                Hua _COND            Hua _DS-CHAIN
##                       2                        1                       2
##            Hua _INCONSEQ            Hua _SS-CHAIN               Hua _TOPIC
##                       1                        2                       1
##             Kâte _CHAIN             Kâte _SERIAL             Kâte _TOPIC
##                       2                        2                       2
##           Korafe _CHAIN           Lenakel _CHAIN            Nepali _CHAIN
##                       2                        2                       2
##             Nepali _CVB            Nepali _TOPIC        NewarDolakha_CHAIN
##                       2                        1                       2
##            Russian _AND         Russian _CVB-ADV     Russian _CVB.DETACHED
##                       1                        2                       2
##            Swahili _AND         Swahili _AND.INF            Swahili _CHAIN
##                       1                        2                       2
##           Swahili _TOPIC             Tauya _CHAIN           Tauya _INCONSEQ
##                       2                        2                       1
##             Tauya _TOPIC             Toura _CHAIN            Turkish _CHAIN
##                       1                        2                       2
##          Turkish _SERIAL              Usan _CAUSE           Usan _CHAIN.SWR
##                       2                        2                       2
##              Usan _TOPIC           Wambule _CHAIN         Yup’ikCentral_SUB
##                       1                        2                       1
```

```
clusters2 <- as.data.frame(cutree(db.hclust, k = 2))
clusters2$Name <- cl$Name
colnames(clusters2)[1] <- "Clusters_partition1"
 head(clusters2)
```

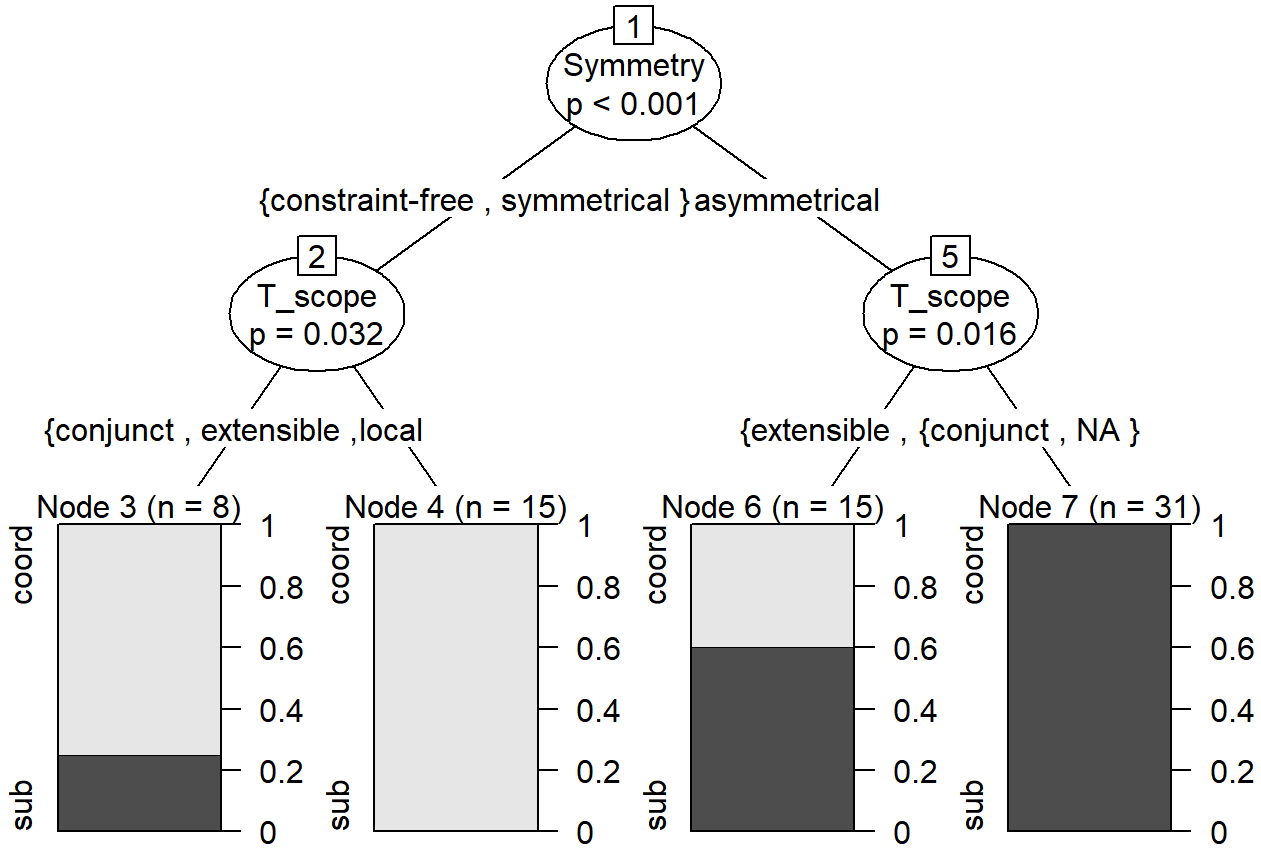
	Clusters_partition1	Name
	<int>	<fct>
Amele _BUT	1	Amele _BUT
Amele _CHAIN	2	Amele _CHAIN
Amele _COND	1	Amele _COND
Amele _OR	1	Amele _OR
Amele _PURP	1	Amele _PURP
Belhare _AND	1	Belhare _AND

6 rows

```
cl2 <- merge(cl, clusters2, by="Name")
summary(cl2)
```

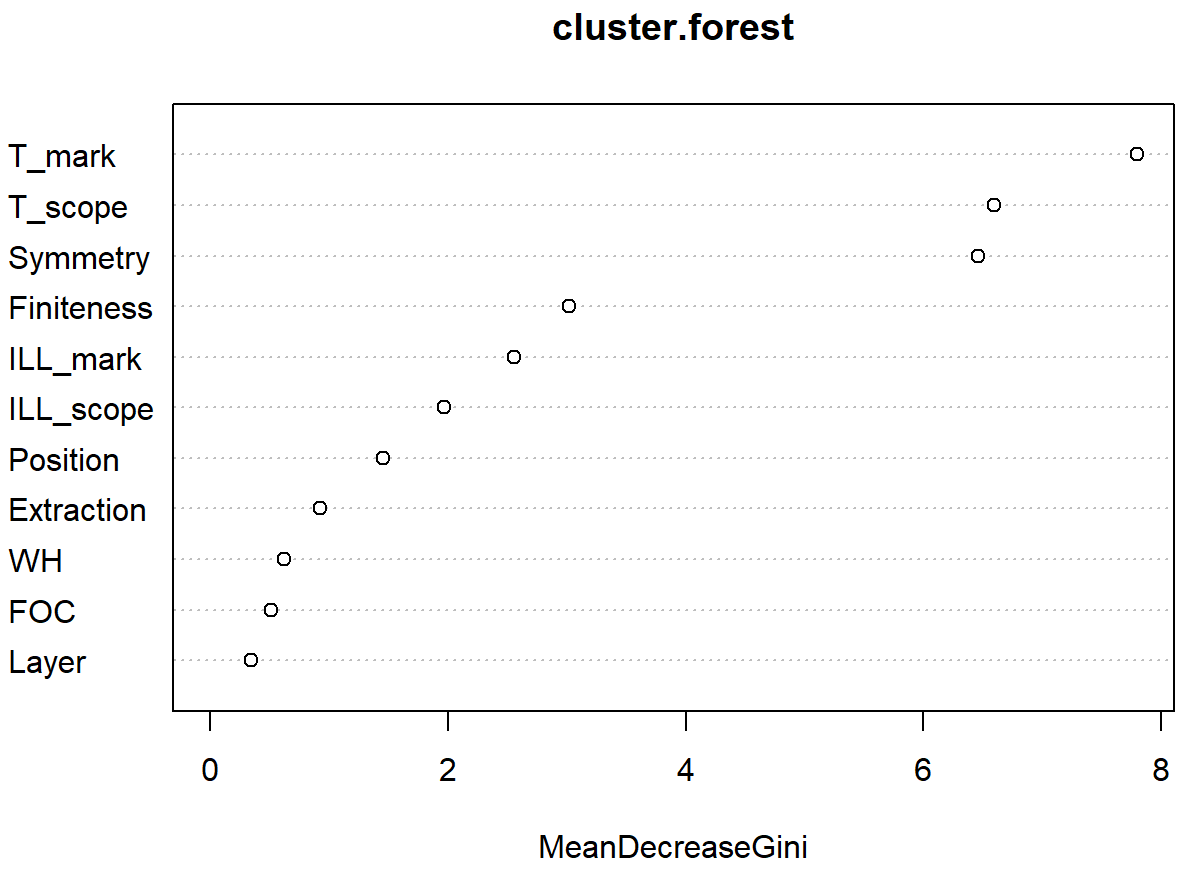
```
##             Name               ILL_scope         T_scope        Finiteness
##  Amele _BUT   : 1   conjunct        :14   conjunct   :32   any       : 4
##  Amele _CHAIN : 1   constraint-free :17   extensible :11   finite    :22
##  Amele _COND  : 1   disjunct        :21   local      :23   NA        : 1
##  Amele _OR    : 1   extensible      : 2   NA         : 3   nonfinite :42
##  Amele _PURP  : 1   local           :14
##  Belhare _AND : 1   NA              : 1
##  (Other)      :63
##       ILL_mark        T_mark               Symmetry        WH
##  banned   :39   banned   :30   asymmetrical    :46   banned :16
##  harmonic : 2   harmonic : 5   constraint-free : 9   NA     :29
##  NA       : 8   NA       : 2   symmetrical     :14   ok     :24
##  ok       :20   ok       :32
##
##
##
##      Extraction      FOC                     Position        Layer
##  banned   :28   banned : 5   fixed:post-main     : 6   ad-S     :55
##  NA       :36   NA     :35   fixed:pre-main      :20   ad-V     : 8
##  possible : 5   ok     :29   flexible-adjacent   :24   detached : 6
##                              flexible-relational :17
##                              flxed:pre-main      : 1
##                              NA                  : 1
##
##  Clusters_partition1
##  Min.   :1.000
##  1st Qu.:1.000
##  Median :2.000
##  Mean   :1.609
##  3rd Qu.:2.000
##  Max.   :2.000
##
```

```
cl2$Clusters_partition1 <- as.factor(cl2$Clusters_partition1)
cl2$Cluster <- as.factor(ifelse(cl2$Clusters_partition1 == 1, "coord", "sub"))
cluster.tree <- ctree(Cluster~ILL_scope+T_scope+Finiteness+ILL_mark+T_mark+Symmetry+WH+Extrac
tion+FOC+Position+Layer, data=cl2)
plot(cluster.tree)
```




```
cluster.forest<- randomForest(Cluster~ILL_scope+T_scope+Finiteness+ILL_mark+T_mark+Symmetry+WH+Extraction+FOC+Position+Layer, data=cl2,  controls=cforest_unbiased(ntree=100, mtry=2))
```

```
varImpPlot(cluster.forest)
```




```
cluster.forest
```

```
##
## Call:
##  randomForest(formula = Cluster ~ ILL_scope + T_scope + Finiteness +      ILL_mark + T_mar
k + Symmetry + WH + Extraction + FOC + Position +      Layer, data = cl2, controls = cforest_
unbiased(ntree = 100,      mtry = 2))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 7.25%
## Confusion matrix:
##       coord sub class.error
## coord    24   3  0.11111111
## sub       2  40  0.04761905
```

Programming in R and simulating a null hypothesis

We don't really know whether our partition into two groups is a meaningful partition. Notice that in inferential statistics we have some way of knowing whether the result the model spits back is better than `chance". how do we overcome the` clustering tendency problem?"

One way of doing this would be to try to simulate a ``random" distribution and see how different it is from the results of the actual data.

```
head(cl)
```

	Name <fct>	ILL_scope <fct>	T_scope <fct>	Finiteness <fct>	ILL_m… <fct>	T_m… <fct>
Amele _BUT	Amele _BUT	local	local	finite	banned	ok
Amele _CHAIN	Amele _CHAIN	conjunct	conjunct	nonfinite	banned	banned
Amele _COND	Amele _COND	disjunct	extensible	any	ok	ok
Amele _OR	Amele _OR	conjunct	local	finite	ok	ok
Amele _PURP	Amele _PURP	disjunct	extensible	any	banned	ok
Belhare _AND	Belhare _AND	constraint-free	local	finite	ok	ok

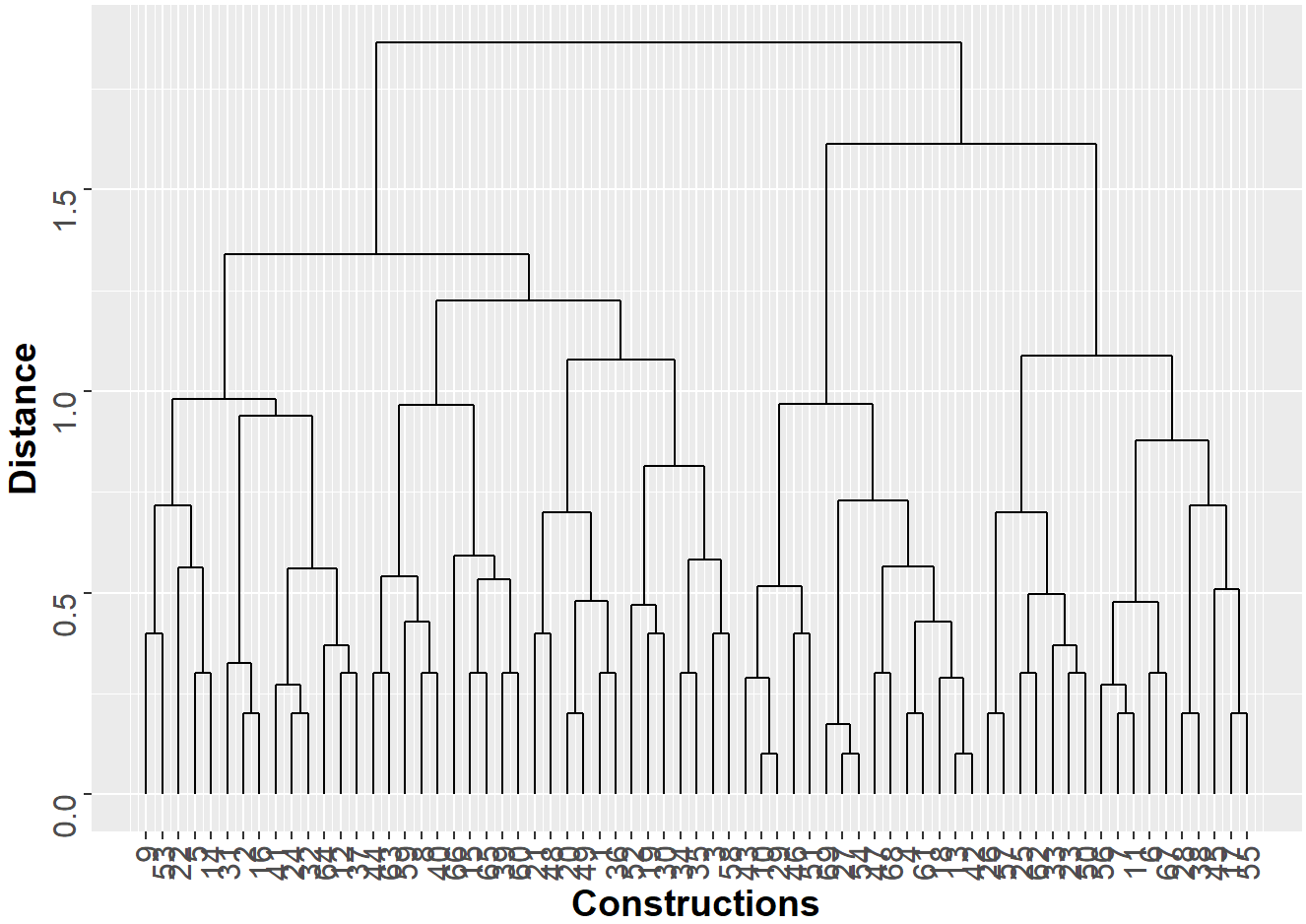
6 rows | 1-7 of 13 columns

```
set.seed(1234)
  ILL_scope_sim <- as.character(sample(cl$ILL_scope, 69, replace=T))
T_scope_sim <- as.character(sample(cl$T_scope, 69, replace=T))
Finiteness_sim <- as.character(sample(cl$Finiteness, 69, replace=T))
ILL_mark_sim <- as.character(sample(cl$ILL_mark, 69, replace=T))
T_mark_sim <- as.character(sample(cl$T_mark, 69, replace=T))
WH_sim <- as.character(sample(cl$WH, 69, replace=T))
Extraction_sim <- as.character(sample(cl$Extraction, 69, replace=T))
FOC_sim <- as.character(sample(cl$FOC, 69, replace=T))
Position_sim <- as.character(sample(cl$Position, 69, replace=T))
Layer_sim <- as.character(sample(cl$Layer, 69, replace=T))
sim_db <- as_tibble(cbind(ILL_scope_sim, T_scope_sim, Finiteness_sim, ILL_mark_sim, T_mark_si
m, WH_sim, Extraction_sim, FOC_sim, Position_sim, Layer_sim)) %>% mutate_if(is.character, as.
factor)
```

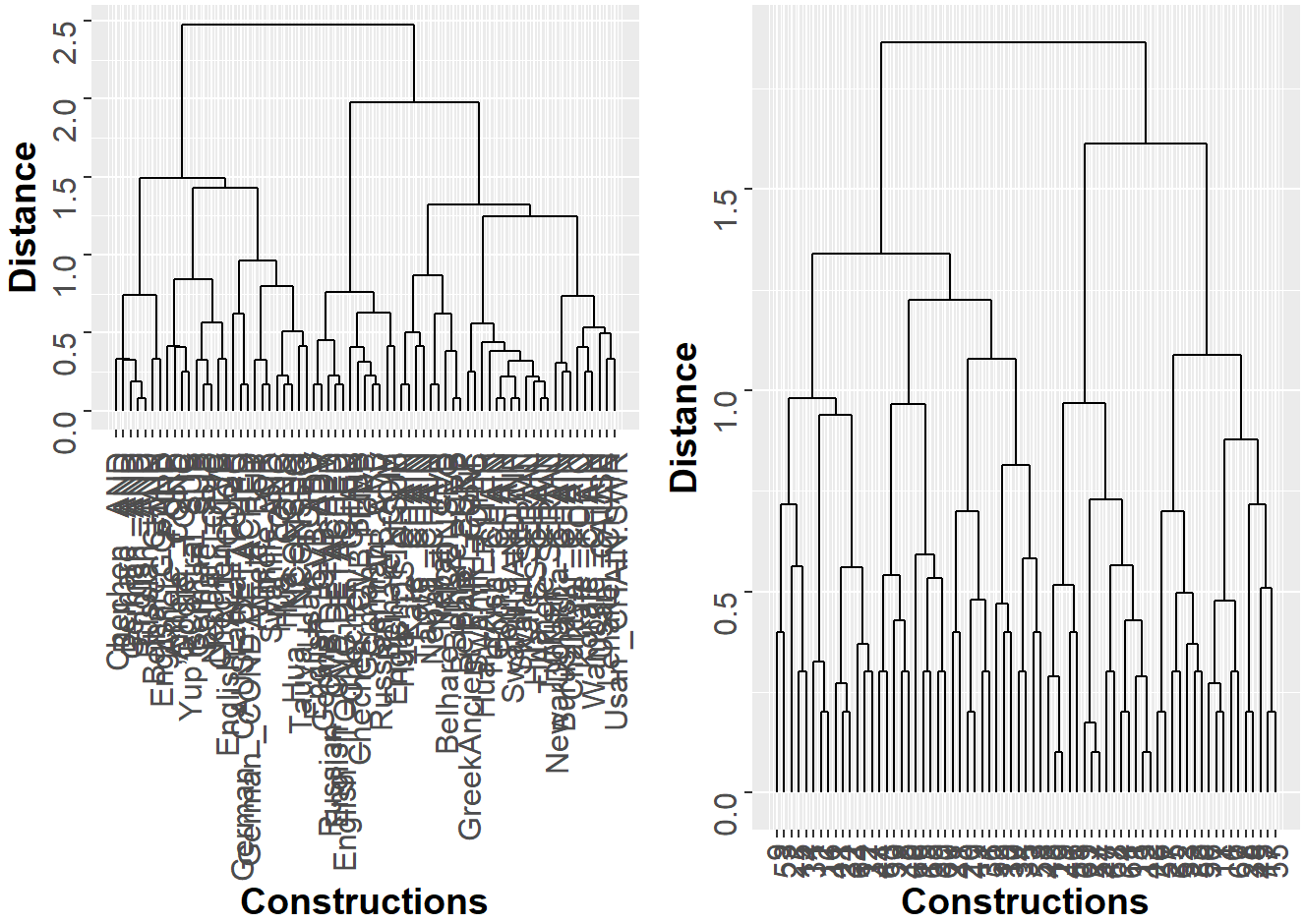
We can make a dendrogram for our simulated data.

```
sim_dist <- daisy(sim_db, metric = "gower")
sim_hclust <- hclust(sim_dist, method="ward.D2")
# plot results
sim_dendro<- ggdendrogram(sim_hclust, rotate = FALSE, theme_dendro =FALSE)+
  theme(axis.text=element_text(size=12),
        axis.title=element_text(size=14,face="bold"))+
        xlab("Constructions")+
        ylab("Distance")
```

```
sim_dendro
```




```
grid.arrange(db_dendro, sim_dendro, nrow=1)
```



Of interest for us is the relative height difference between the first and second partition of the data. This can stand as a proxy for how strong the dichotomous split into coordinate and subordinate constructions is.

Making functions in R (cophenetic correlation and the height difference)

What we want is a function that tells us how well the classification is. for this we can use the cophonetic correlation and a measurement of how strong the first partition is. Let's make a function that does this.

In R you can make a function with the function, function(). For instance.

```
function.1 <- function(x){
  x^2 +1
}
```

```
function.1(5)
```

```
## [1] 26
```

The following would make a function that spits out our cophenetic correlation and a measurement of the first height difference.

```
coph_height <- function(x, y){
  coph <- cophenetic(x)
  cor <- round(cor(y, coph), 4)
  height <- x$height
  rel_height_diff <- round(height[length(height)]-height[length(height)-1], 4) / max(x$heigh
t)
  returnlist <- c("heightdiff" = rel_height_diff, "coph" = cor)
}
```

Here is the cophenetic correlation and the height difference.

```
sim_ch<-coph_height(sim_hclust, sim_dist)
sim_ch
```

```
## heightdiff       coph
##  0.1344581  0.4055000
```

```
db.ch<-coph_height(db.hclust, db.dist)
db.ch
```

```
## heightdiff       coph
##  0.2003889  0.5888000
```

So the split between two groups is stronger than chance, maybe but the cophenetic variation is much lower.

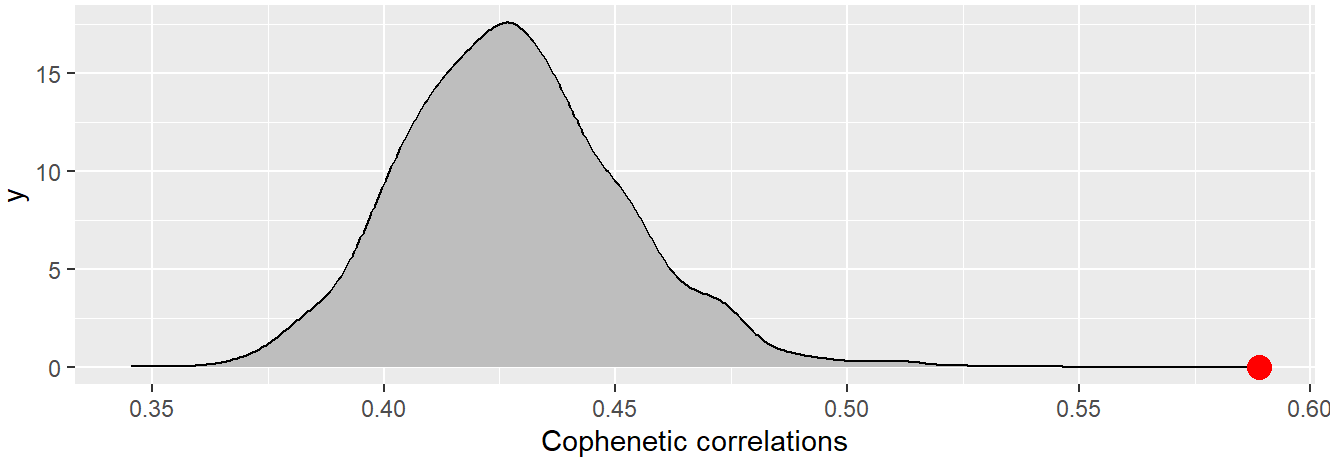
However, we only did one simulation.

```
sim1000 <- lapply(1:1000, function(s) {
  ILL_scope_sim <- as.character(sample(cl$ILL_scope, 69, replace=T))
T_scope_sim <- as.character(sample(cl$T_scope, 69, replace=T))
Finiteness_sim <- as.character(sample(cl$Finiteness, 69, replace=T))
ILL_mark_sim <- as.character(sample(cl$ILL_mark, 69, replace=T))
T_mark_sim <- as.character(sample(cl$T_mark, 69, replace=T))
WH_sim <- as.character(sample(cl$WH, 69, replace=T))
Extraction_sim <- as.character(sample(cl$Extraction, 69, replace=T))
FOC_sim <- as.character(sample(cl$FOC, 69, replace=T))
Position_sim <- as.character(sample(cl$Position, 69, replace=T))
Layer_sim <- as.character(sample(cl$Layer, 69, replace=T))
sim_db <- as_tibble(ILL_scope_sim, T_scope_sim, Finiteness_sim, ILL_mark_sim, T_mark_sim, WH_
sim, Extraction_sim, FOC_sim, Position_sim, Layer_sim)
sim_db <- as_tibble(cbind(ILL_scope_sim, T_scope_sim, Finiteness_sim, ILL_mark_sim, T_mark_si
m, WH_sim, Extraction_sim, FOC_sim, Position_sim, Layer_sim)) %>% mutate_if(is.character, as.
factor)
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.   :0.003518   Min.   :0.3455
##  1st Qu.:0.085749   1st Qu.:0.4117
##  Median :0.124970   Median :0.4268
##  Mean   :0.132479   Mean   :0.4279
##  3rd Qu.:0.175331   3rd Qu.:0.4420
##  Max.   :0.321508   Max.   :0.5402
```

```
dplot1<-ggplot(sim1000_df, aes(x=coph))+
  geom_density(color="black", fill="grey")+
  geom_point(aes(x=0.5888000, y=0),colour="red", size =4)+
  xlab("Cophenetic correlations")
dplot2<- ggplot(sim1000_df, aes(x=heightdiff))+
  geom_density(color="black", fill="grey")+
  geom_point(aes(x=0.2003889, y=0),colour="red", size =4)+
  xlab("Height differences between the first and second partition")
grid.arrange(dplot1, dplot2, nrow=2)
```


15
10
5
0
y
0.35
0.40
0.45
0.50
0.55
0.60
Cophenetic correlations


6
4
2
0
y
0.0
0.1
0.2
0.3
Height differences between the first and second partition