Code for ‘Clause-linkage, embeddedness, and nominalizations in Chácobo (Pano)’

Adam J.R. Tallman

2024-01-21

## Packages

library(tidyverse)

library(gdata)

library(dendextend)

library(cluster)  
library(factoextra)

library(ggdendro)

library(phangorn)

library(pacman)  
library(tidyverse)  
library(gridExtra)

library(xtable)

library(stringi)

## Neighbornet

## Loading the file

You can also embed plots, for example:

cl <- read\_csv("/YourPathway/bickelclauselinkageandchacobo3.csv") %>%  
 mutate(across(everything(), trimws)) %>%  
 mutate(across(everything(), stri\_trans\_nfc))

## Rows: 82 Columns: 14  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (14): Language, Label, Marker, ILL-scope, T-scope, Finiteness, ILL-mark,...  
##   
## 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(cl)

## Rows: 82  
## Columns: 14  
## $ Language <chr> "Russian", "Russian", "NewarDolakha", "Nepali", "Nepali", …  
## $ Label <chr> "cvb-adv", "cvb.detached", "chain", "chain", "cvb", "ds-ch…  
## $ Marker <chr> "-a, -v", "-a, -v", "-en", "-era", "-era", "-ga", "-ing", …  
## $ `ILL-scope` <chr> "disjunct", "disjunct", "constraint-free", "constraint-fre…  
## $ `T-scope` <chr> "conjunct", "conjunct", NA, "conjunct", "conjunct", "conju…  
## $ Finiteness <chr> "nonfinite", "nonfinite", "nonfinite", "nonfinite", "nonfi…  
## $ `ILL-mark` <chr> "banned", "banned", NA, "banned", "banned", "banned", "ban…  
## $ `T-mark` <chr> "banned", "banned", "banned", "ok", "ok", "harmonic", "ban…  
## $ Symmetry <chr> "asymmetrical", "asymmetrical", "asymmetrical", "asymmetri…  
## $ WH <chr> "banned", "banned", NA, "ok", "ok", "ok", "banned", "banne…  
## $ Extraction <chr> "banned", "banned", NA, NA, NA, NA, "possible", "banned", …  
## $ FOC <chr> "ok", "ok", "ok", "ok", "ok", NA, "ok", "ok", "ok", NA, NA…  
## $ Position <chr> "flexible-relational", "fixed:pre-main", "flexible-adjacen…  
## $ Layer <chr> "ad-V", "detached", "ad-S", "ad-S", "ad-V", "ad-S", "ad-V"…

This code is used to standardize or clean up the dataframe as its downloaded from Bickel’s paper.

cl <- cl %>%  
 unite(Language:Label, col = "LabelNN", sep = "\_", remove = FALSE, na.rm = TRUE) %>%  
 mutate(LabelNN = str\_replace\_all(LabelNN, ":", ".")) %>%  
 mutate(LabelNN = str\_replace\_all(LabelNN, "/", ".")) %>%  
 mutate(LabelNN = str\_remove\_all(LabelNN, "’")) %>%  
 mutate(LabelNN = stri\_trans\_general(LabelNN, id = "Latin-ASCII")) %>%  
 mutate(across(`ILL-scope`:Layer, as.factor)) %>%  
 arrange(Language, Label)

To use splitstree its important that there are no duplicates. The following code checks for duplicates in the labels.

anyDuplicated(cl$LabelNN)

## [1] 0

## Create distance matrix

The following code is used to create the distance matrix.

cl\_variables <- cl %>%  
 select(`ILL-scope`:Layer)  
cl\_dist <- daisy(cl\_variables, metric = "gower")

After you make the distance matrix it loses the labels associated with the tests, so you have to add them back in with the following code.

cl\_dist <- as.matrix(cl\_dist, labels=TRUE)  
rownames(cl\_dist) <- cl$LabelNN

To make a neighbornet tree with splitstree you have make a nexus file. This is done with the following code.

write.nexus.dist(cl\_dist, file = "/YourPathway/clause\_linkage\_dist.nex")

The following is used to make the distance matrix that is displayed in the paper.

cl\_dist2 <- cl\_dist  
colnames(cl\_dist2) <-cl$LabelNN  
write.csv(cl\_dist2, file = "/YourPathway/distmatrix.csv")

## Cluster analysis

We load the data that has more fine grained variables for Chácobo.

rd <- read.table("/YourPathway/chadependentclauses3.csv", sep=",", header=TRUE)

The following code transforms the variables associated with assymetric extraction into a single continuous variable that varies from 0 to 1.

whv <- select(rd, WH.NP.EXT.MAIN:WH.ADV.EXT.DEP)  
whv[whv=="ok"] <- 2   
whv[whv=="local"] <- 1  
whv[whv=="banned"] <- 0  
whv <- whv %>% mutate\_if(is.character, as.numeric)  
assym.extr <- (rowSums(whv) / max(rowSums((whv))))  
assym.extr

## [1] 0.875 1.000 0.500 1.000 0.875 1.000 0.250 0.875 0.500 1.000 1.000 0.875  
## [13] 0.875

rd$assym.extr <- as.numeric(assym.extr)

We also create a variable for finiteness by aggregating all the variables for specific morphosyntactic markers.

fv <- select(rd, Neg.marked:hueni.marked)  
fv[fv=="ok"] <- 0  
fv[fv=="banned"] <-1  
fv <- fv %>% mutate\_if(is.character, as.numeric)  
nonfiniteness <- rowSums(fv) / max(rowSums((fv)))  
rd$nonfiniteness <- as.numeric(nonfiniteness)

We select the variables

df <- select(rd, referential.function, noun.modifying, ILL.scope, T.scope, Neg.scope, T.mark, WH.NP.EXT.MAIN:WH.ADV.EXT.DEP, nonfiniteness, FOC, Position)  
df[is.na(df)] <- "ok"

We have to assign names to the rows.

varnames <- rd$Label  
rownames(df) <- varnames  
df

## referential.function noun.modifying ILL.scope T.scope  
## concur:sa na no extensible conjunct  
## prior:ds/a na no extensible conjunct  
## quick:ss/a na no extensible conjunct  
## concur:ds/a na no extensible conjunct  
## subseq:ss/a na no extensible conjunct  
## subseq:ds/a na no extensible conjunct  
## interrupt:ss/a na no local conjunct  
## prior:sa na no extensible conjunct  
## nmlz:purp instrumental no local conjunct  
## nmd na yes extensible conjunct  
## nmlz:agt agentive yes extensible conjunct  
## prior:ss na no extensible conjunct  
## concur:ss na no extensible conjunct  
## Neg.scope T.mark WH.NP.EXT.MAIN WH.NP.EXT.DEP WH.ADV.EXT.MAIN  
## concur:sa extensible ok ok local ok  
## prior:ds/a local ok ok ok ok  
## quick:ss/a extensible banned banned banned ok  
## concur:ds/a local ok ok ok ok  
## subseq:ss/a extensible ok ok local ok  
## subseq:ds/a local ok ok ok ok  
## interrupt:ss/a local ok banned local banned  
## prior:sa extensible ok ok local ok  
## nmlz:purp local banned ok banned ok  
## nmd extensible ok ok ok ok  
## nmlz:agt local ok ok ok ok  
## prior:ss extensible ok ok local ok  
## concur:ss extensible ok ok local ok  
## WH.ADV.EXT.DEP nonfiniteness FOC Position  
## concur:sa ok 0.7058824 ok flexible-relational  
## prior:ds/a ok 0.6470588 ok flexible-relational  
## quick:ss/a ok 0.8235294 banned fixed:pre-main  
## concur:ds/a ok 0.6470588 ok flexible-relational  
## subseq:ss/a ok 0.7647059 ok flexible-relational  
## subseq:ds/a ok 0.7647059 ok flexible-relational  
## interrupt:ss/a local 0.7647059 ok fixed:pre-main  
## prior:sa ok 0.7058824 ok flexible-relational  
## nmlz:purp banned 1.0000000 banned flexible-relational  
## nmd ok 0.5294118 ok flexible-relational  
## nmlz:agt ok 0.4705882 ok flexible-relational  
## prior:ss ok 0.7058824 ok flexible-relational  
## concur:ss ok 0.7058824 ok flexible-relational

We make all the variables factors.

df <- df %>% mutate\_if(is.character, as.factor)

We create a distance matrix using gower’s distance metric, which can handle categorical data.

db.dist <- daisy(df, metric="gower")  
db.dist2 <- as.matrix(db.dist, labels=TRUE)

For the paper, we reorder the variables.

col.order <- c("prior:ss", "prior:sa", "concur:ss","concur:sa", "subseq:ss/a", "interrupt:ss/a", "quick:ss/a", "prior:ds/a", "concur:ds/a", "subseq:ds/a", "nmd", "nmlz:agt", "nmlz:purp")  
row.order <- c("prior:ss", "prior:sa", "concur:ss","concur:sa", "subseq:ss/a", "interrupt:ss/a", "quick:ss/a", "prior:ds/a", "concur:ds/a", "subseq:ds/a", "nmd", "nmlz:agt", "nmlz:purp")  
db.dist\_ordered<- db.dist2[row.order, col.order]

We also make a distance matrix for the table.

write.csv(db.dist\_ordered, file = "/YourPathway//distmatrixchacobo.csv")

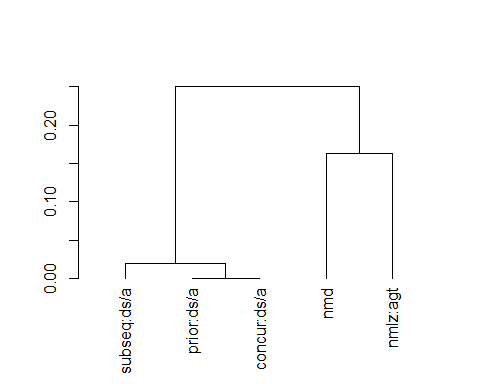
We use the function hclust() to perform a hierarchical cluster analysis.

db.hclust<-hclust(db.dist, method="ward.D2")  
db.hclust

##   
## Call:  
## hclust(d = db.dist, method = "ward.D2")  
##   
## Cluster method : ward.D2   
## Number of objects: 13

The following code is used to create Figure 2 which is a part of the hierarchical tree.

plot(cut(as.dendrogram(db.hclust), h=0.35054060)$lower[[2]])



get\_branches\_heights(as.dendrogram(db.hclust))

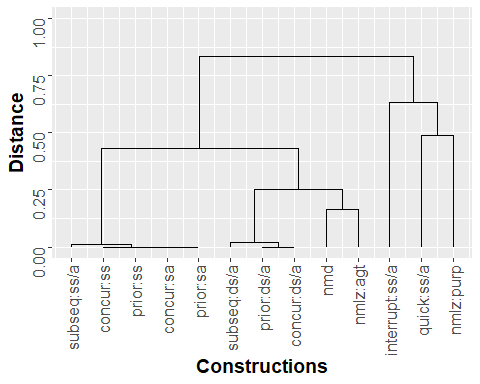
## [1] 0.00000000 0.00000000 0.00000000 0.00000000 0.01081121 0.01973847  
## [7] 0.16239316 0.25054060 0.43104387 0.48717949 0.63376707 0.83265070

The following code creates the whole dendrogram of Figure 3.

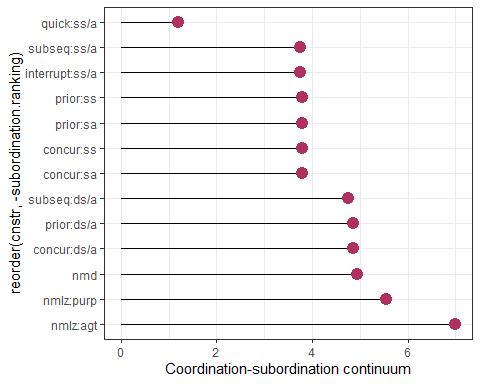
db\_dendro <- ggdendrogram(db.hclust, rotate = FALSE, theme\_dendro = FALSE)+theme(axis.text=element\_text(size=12), axis.title=element\_text(size=14,face="bold"))+  
 xlab("Constructions")+  
 ylab("Distance")+  
 ylim(0,1)

## Scale for y is already present.  
## Adding another scale for y, which will replace the existing scale.

db\_dendro

 The following code is used to make a scale from more subordinate-like to more coordinate-like based on my understanding of how these criteria relate to the distinction. This is needed to create Figure 4.

illscope.i <- ifelse(rd$ILL.scope == "local", 1, 0)  
ref.i <- ifelse(rd$referential.function=="na",0,1)  
noun.modify.i <- ifelse(rd$noun.modifying=="yes", 1, 0)  
neg.scope.i <- ifelse(rd$Neg.scope=="local", 1, 0)   
whv2 <- select(rd, WH.NP.EXT.MAIN, WH.ADV.EXT.MAIN)  
whv2[whv2=="ok"] <- 2   
whv2[whv2=="local"] <- 1  
whv2[whv2=="banned"] <- 0  
whv2 <- whv2 %>% mutate\_if(is.character, as.numeric)  
extr.main.i <- (rowSums(whv2) / max(rowSums((whv2))))  
#atb.extr.i <- ifelse(rd$WH.NP.ATB.EXT == "ok", 0, 1)  
foc.i <- ifelse(rd$FOC == "banned", 0, 1)  
position.i <- ifelse(rd$Position =="fixed:pre-main", 0,1)  
#Nonfiniteness variable  
fv <- select(rd, Neg.marked:hueni.marked)  
fv[fv=="ok"] <- 1  
fv[fv=="banned"] <-0  
fv <- fv %>% mutate\_if(is.character, as.numeric)  
nonfininteness <- rowSums(fv) / max(rowSums((fv)))  
df.i <- data.frame(illscope.i, ref.i, noun.modify.i, neg.scope.i, foc.i, position.i, nonfininteness, extr.main.i)  
df.i2 <- df.i  
df.i2$cnstr <- rd$Label  
df.i2$subordination.ranking <- rowSums(df.i)  
df.i2$cnstr <-factor(df.i2$cnstr)  
df.i2 %>%  
 arrange(subordination.ranking) %>% # First sort by val. This sort the dataframe but NOT the factor levels  
 mutate(name=factor(cnstr, levels=cnstr)) %>% # This trick update the factor levels  
 ggplot(aes(x=reorder(cnstr,-subordination.ranking), y=subordination.ranking)) +  
 ylab("Coordination-subordination continuum")+  
 geom\_segment(aes(xend=name, yend=0)) +  
 geom\_point(size=4, color="maroon") +  
 coord\_flip() +  
 theme\_bw() +  
 ylim(0,7)



## Cluster validation

The following function will give the cophenetic correlation and the height of the first cut.

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)  
}

The paper involves simulations. We need to set the seed for reproduceability.

set.seed(222)

The following code can be used to simulate a single hierarchical cluster.

referential.function <- as.character(sample(df$referential.function, 13, replace=T)) #check  
noun.modifying <- as.character(sample(df$noun.modifying, 13, replace=T)) #check  
ILL.scope <- as.character(sample(df$ILL.scope, 13, replace=T)) #check  
Neg.scope <- as.character(sample(df$Neg.scope, 13, replace=T)) #check  
T.scope <- as.character(sample(df$T.scope, 13, replace=T)) #check  
#ILL.mark <- as.character(sample(df$ILL.mark, 13, replace=T)) #check  
T.mark <- as.character(sample(df$T.mark, 13, replace=T)) #check  
#Symmetry <- as.character(sample(df$Symmetry, 13, replace=T)) #check  
WH.NP.EXT.MAIN <- as.character(sample(df$WH.NP.EXT.MAIN, 13, replace=T)) #check  
WH.NP.EXT.DEP <- as.character(sample(df$WH.NP.EXT.DEP, 13, replace=T)) #check  
WH.ADV.EXT.MAIN <- as.character(sample(df$WH.NP.EXT.MAIN, 13, replace=T)) #check  
WH.ADV.EXT.DEP <- as.character(sample(df$WH.NP.EXT.DEP, 13, replace=T)) #check  
#WH.NP.ATB.EXT <- as.character(sample(df$WH.NP.ATB.EXT, 13, replace=T)) #check  
FOC <- as.character(sample(df$FOC, 13, replace=T)) #check  
#WH.NP.dep <- as.character(sample(df$WH.NP.dep, 13, replace=T))  
Position <- as.character(sample(df$Position, 13, replace=T))  
#Min.left.edge <- as.character(sample(df$Min.left.edge, 13, replace=T))  
#Min.right.edge <- as.character(sample(df$Min.right.edge, 13, replace=T))  
#Max.left.edge <- as.character(sample(df$Max.left.edge, 13, replace=T))  
#Max.right.edge <- as.character(sample(df$Max.right.edge, 13, replace=T))  
#Center.embed.pa <- as.character(sample(df$Center.embed.pa, 13, replace=T))  
#Center.embed.case <- as.character(sample(df$Center.embed.case, 13, replace=T))  
#Neg.marked <- as.character(sample(df$Neg.marked, 13, replace=T)) #check  
Nonfiniteness <- as.character(sample(df$nonfiniteness, 13, replace=T))  
#ya <- as.character(sample(df$ya.mark, 13, replace=T))  
#pao <- as.character(sample(df$pao.marked, 13, replace=T))  
#bahina <- as.character(sample(df$bahina.marked, 13, replace=T))  
#tapi <- as.character(sample(df$tapi.marked, 13, replace=T))  
#yo <- as.character(sample(df$yo.marked, 13, replace=T))  
#shari <- as.character(sample(df$shari.marked, 13, replace=T))  
#hita <- as.character(sample(df$hita.marked, 13, replace=T))  
#yamet <- as.character(sample(df$yamet.marked, 13, replace=T))  
#quiha <- as.character(sample(df$quiha.marked, 13, replace=T))  
#ni <- as.character(sample(df$ni.marked, 13, replace=T))  
#AM <- as.character(sample(df$AM.marked, 13, replace=T))  
#xe <- as.character(sample(df$xe.marked, 13, replace=T))  
#quea <- as.character(sample(df$quea.marked, 13, replace=T))  
#cara <- as.character(sample(df$cara.marked, 13, replace=T))  
#bequi <- as.character(sample(df$bequi.marked, 13, replace=T))  
#roha <- as.character(sample(df$roha.marked, 13, replace=T))  
#tëquën <- as.character(sample(df$tëquën.marked, 13, replace=T))  
#tiarihi <- as.character(sample(df$tiarihi.marked, 13, replace=T))  
#pe <- as.character(sample(df$pe.marked, 13, replace=T))  
#huesti <- as.character(sample(df$huesti.marked, 13, replace=T))  
#rabe <- as.character(sample(df$rabe.marked, 13, replace=T))  
#hueni <- as.character(sample(df$hueni.marked, 13, replace=T))  
sim\_db <- as\_tibble(cbind(FOC, referential.function, noun.modifying, ILL.scope, WH.NP.EXT.DEP, WH.NP.EXT.MAIN, WH.ADV.EXT.DEP, WH.NP.EXT.MAIN, Position, Nonfiniteness, T.scope, T.mark, Neg.scope)) %>%   
 mutate\_if(is.character, as.factor)

## Warning: The `x` argument of `as\_tibble.matrix()` must have unique column names if  
## `.name\_repair` is omitted as of tibble 2.0.0.  
## ℹ Using compatibility `.name\_repair`.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

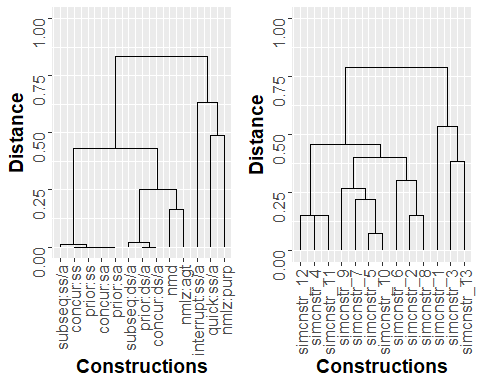
sim\_db <- as.data.frame(sim\_db)  
rownames(sim\_db) <- c("simcnstr\_1", "simcnstr\_2", "simcnstr\_3", "simcnstr\_4", "simcnstr\_5", "simcnstr\_6", "simcnstr\_7", "simcnstr\_8", "simcnstr\_9", "simcnstr\_10", "simcnstr\_11", "simcnstr\_12", "simcnstr\_13")

set.seed(200)  
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")+  
 ylim(0,1)

## Scale for y is already present.  
## Adding another scale for y, which will replace the existing scale.

The following gives you the figure which compares the dendrogram based on the actual data and the dendrogram from simlated data.

grid.arrange(db\_dendro, sim\_dendro, nrow=1)



The following code simulates dendrograms 1000 times. We make finiteness a continuous variable, therefore, all the other individual markers are commented out.

sim1000 <- lapply(1:1000, function(s) {  
 referential.function <- as.character(sample(df$referential.function, 13, replace=T)) #check  
 noun.modifying <- as.character(sample(df$noun.modifying, 13, replace=T)) #check  
 ILL.scope <- as.character(sample(df$ILL.scope, 13, replace=T)) #check  
 Neg.scope <- as.character(sample(df$Neg.scope, 13, replace=T)) #check  
 T.scope <- as.character(sample(df$T.scope, 13, replace=T)) #check  
 #ILL.mark <- as.character(sample(df$ILL.mark, 13, replace=T)) #check  
 T.mark <- as.character(sample(df$T.mark, 13, replace=T)) #check  
 #Symmetry <- as.character(sample(df$Symmetry, 13, replace=T)) #check  
 WH.NP.EXT.MAIN <- as.character(sample(df$WH.NP.EXT.MAIN, 13, replace=T)) #check  
 WH.NP.EXT.DEP <- as.character(sample(df$WH.NP.EXT.DEP, 13, replace=T)) #check  
 WH.ADV.EXT.MAIN <- as.character(sample(df$WH.NP.EXT.MAIN, 13, replace=T)) #check  
 WH.ADV.EXT.DEP <- as.character(sample(df$WH.NP.EXT.DEP, 13, replace=T)) #check  
 #WH.NP.ATB.EXT <- as.character(sample(df$WH.NP.ATB.EXT, 13, replace=T)) #check  
 FOC <- as.character(sample(df$FOC, 13, replace=T)) #check  
 #WH.NP.dep <- as.character(sample(df$WH.NP.dep, 13, replace=T))  
 Position <- as.character(sample(df$Position, 13, replace=T))  
 #Min.left.edge <- as.character(sample(df$Min.left.edge, 13, replace=T))  
 #Min.right.edge <- as.character(sample(df$Min.right.edge, 13, replace=T))  
 #Max.left.edge <- as.character(sample(df$Max.left.edge, 13, replace=T))  
 #Max.right.edge <- as.character(sample(df$Max.right.edge, 13, replace=T))  
 #Center.embed.pa <- as.character(sample(df$Center.embed.pa, 13, replace=T))  
 #Center.embed.case <- as.character(sample(df$Center.embed.case, 13, replace=T))  
 #Neg.marked <- as.character(sample(df$Neg.marked, 13, replace=T)) #check  
 Nonfiniteness <- as.character(sample(df$nonfiniteness, 13, replace=T))  
 #ya <- as.character(sample(df$ya.mark, 13, replace=T))  
 #pao <- as.character(sample(df$pao.marked, 13, replace=T))  
 #bahina <- as.character(sample(df$bahina.marked, 13, replace=T))  
 #tapi <- as.character(sample(df$tapi.marked, 13, replace=T))  
 #yo <- as.character(sample(df$yo.marked, 13, replace=T))  
 #shari <- as.character(sample(df$shari.marked, 13, replace=T))  
 #hita <- as.character(sample(df$hita.marked, 13, replace=T))  
 #yamet <- as.character(sample(df$yamet.marked, 13, replace=T))  
 #quiha <- as.character(sample(df$quiha.marked, 13, replace=T))  
 #ni <- as.character(sample(df$ni.marked, 13, replace=T))  
 #AM <- as.character(sample(df$AM.marked, 13, replace=T))  
 #xe <- as.character(sample(df$xe.marked, 13, replace=T))  
 #quea <- as.character(sample(df$quea.marked, 13, replace=T))  
 #cara <- as.character(sample(df$cara.marked, 13, replace=T))  
 #bequi <- as.character(sample(df$bequi.marked, 13, replace=T))  
 #roha <- as.character(sample(df$roha.marked, 13, replace=T))  
 #tëquën <- as.character(sample(df$tëquën.marked, 13, replace=T))  
 #tiarihi <- as.character(sample(df$tiarihi.marked, 13, replace=T))  
 #pe <- as.character(sample(df$pe.marked, 13, replace=T))  
 #huesti <- as.character(sample(df$huesti.marked, 13, replace=T))  
 #rabe <- as.character(sample(df$rabe.marked, 13, replace=T))  
 #hueni <- as.character(sample(df$hueni.marked, 13, replace=T))  
 # combine to df  
 sim\_db <- as\_tibble(cbind(FOC, referential.function, noun.modifying, ILL.scope, WH.NP.EXT.DEP, WH.NP.EXT.MAIN, WH.ADV.EXT.DEP, WH.NP.EXT.MAIN, Position, Nonfiniteness, T.scope, T.mark, Neg.scope)) %>%   
 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)}  
)

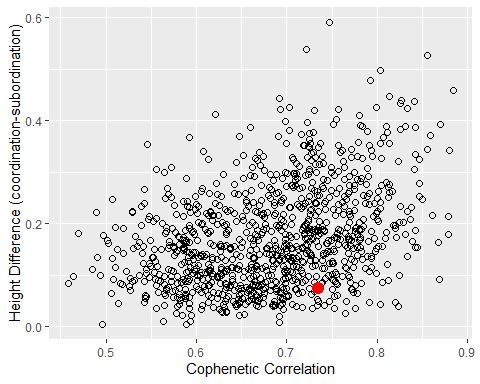
We have to convert it to a database.

# convert to df  
sim1000\_df <- as.data.frame(do.call(rbind, sim1000))  
# means of simulated languages  
summary(sim1000\_df)

## heightdiff coph   
## Min. :0.0041 Min. :0.4586   
## 1st Qu.:0.0962 1st Qu.:0.6175   
## Median :0.1554 Median :0.6872   
## Mean :0.1714 Mean :0.6816   
## 3rd Qu.:0.2310 3rd Qu.:0.7410   
## Max. :0.5910 Max. :0.8842

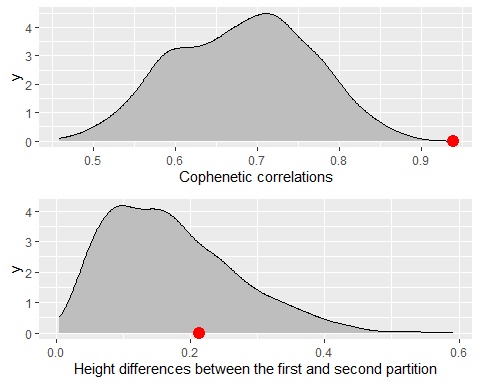
The following code plots cophenetic correlation against the height difference.

sim\_comp <- ggplot(sim1000\_df, aes(x = coph, y = heightdiff)) + geom\_point(size = 2, shape = 1) +  
 geom\_point(aes(x=0.7354,y=0.0744),colour="red", size =4)+  
 labs(x = "Cophenetic Correlation", y = "Height Difference (coordination-subordination)")  
sim\_comp



In the paper density distributions are used instead. The code for Figure 5 is provided below.

dplot1<-ggplot(sim1000\_df, aes(x=coph))+  
 geom\_density(color="black", fill="grey")+  
 geom\_point(aes(x=0.9381, 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=0.2134, y=0),colour="red", size =4)+  
 xlab("Height differences between the first and second partition")  
grid.arrange(dplot1, dplot2, nrow=2)



The numbers in the final paragraphs are calculated as follows.

db.ch<-coph\_height(db.hclust, db.dist)  
cdf.coph <- ecdf(sim1000\_df$coph)  
1- cdf.coph(0.9138)

## [1] 0

cdf.heightdiff <- ecdf(sim1000\_df$heightdiff)  
1-cdf.heightdiff(0.1989)

## [1] 0.334