Test hierarchy

Load libraries

```
library(dplyr)
library(Skillings.Mack)
library(PMCMR)
library(ggplot2)
library(DyaDA)
library(stringdist)
```

Note that the package `DyaDA` is still in development, and will need to be installed from github:

```
#install.packages("devtools")
#library(devtools)
#install_github("DLEIVA/DyaDA")
#library(DyaDA)
```

Load data

```
d = read.csv("../data/DiversityIndices_ND.csv", stringsAsFactors = F)

domain.order = c("colour","shape",'sound','touch','taste','smell')

d$domain = factor(d$domain, levels=domain.order, labels=c("Color","Shape","Sound","Touch","Taste","Smel

# Note spellingo f Yucatec / Yucatek
d$Language = factor(d$Language,
                    levels =
   rev(c("English", "Farsi", "Turkish",
         "Dogul Dom", "Siwu", "Cantonese", "Lao",
         "Malay", "Semai", "Kilivila", "Mian",
         "Yeli Dnye", "Umpila", "Tzeltal",
         "Yucatec", "Zapotec", "Yurakare",
         "ASL", "BSL", "Kata Kolok")),
                    labels =
   rev(c("English", "Farsi", "Turkish",
         "Dogul Dom", "Siwu", "Cantonese", "Lao",
         "Malay", "Semai", "Kilivila", "Mian",
         "Yélî Dnye", "Umpila", "Tzeltal",
         "Yucatec", "Zapotec", "Yurakare",
         "ASL", "BSL", "Kata Kolok")))

labels= data.frame(Language=as.character(levels(d$Language)))
labels$x = 1:nrow(labels)
```

Summarise diversity index by domain and language:

```
d2 = d %>% group_by(Language,domain) %>%
  summarise(m=mean(simpson.diversityIndex, na.rm=T),
            upper=mean(simpson.diversityIndex)+sd(simpson.diversityIndex),
            lower=mean(simpson.diversityIndex)-sd(simpson.diversityIndex))
```

```
## Warning: package 'bindrcpp' was built under R version 3.3.2
```

```
d2$rank = unlist(tapply(d2$m,d2$Language,rank))
```

Analyse hierarchy

There's clearly no strict hierarchy. here is a table each domain's rank number within each language. If there as a strict hierarchy, then each domain would have a single rank number:

```
table(d2$domain,d2$rank)
```

```
##
##          1  2  3  4  5  6
##   Color  0  0  1  7 11  1
##   Shape  2  1  5  8  3  1
##   Sound  3  4  6  3  0  0
##   Touch  2  9  5  1  1  2
##   Taste  0  0  3  1  5  9
##   Smell 13  6  0  0  0  1
```

Calculate median codability.

```
meanCodability = sapply(rev(as.character(unique(d2$Language))),function(l){
  d2x = d2[as.character(d2$Language)==l,]
  mx = d2x$m
  names(mx) = as.character(d2x$domain)
  mx[as.character(unique(d2$domain))]
})
meanCodability = t(meanCodability)
```

Number of unique rank orders

Some languages have no data for some domains, so need to check whether each language is consistent with every other language. The functions below calculate the number of unique compatible rank orders.

```
matchCompatibleij = function(i,j,data){
  # Take two row indices and check if the
  # ranks of the vectors of the rows in data
  # are in a compatible order.
  # Rows can include missing data.
  ix = data[i,]
  jx = data[j,]
  complete = !(is.na(ix)|is.na(jx))
  all(order(ix[complete])==order(jx[complete]))
}
matchCompatible <- Vectorize(matchCompatibleij, vectorize.args=list("i","j"))

compatibleMatches = function(data){
  # Compare all individuals to all others,
  # checking if each is compatible
  m = outer(1:nrow(data),1:nrow(data),matchCompatible,data=data)
  rownames(m) = rownames(data)
  colnames(m) = rownames(data)
  return(m)
}

matches = compatibleMatches(meanCodability)
```

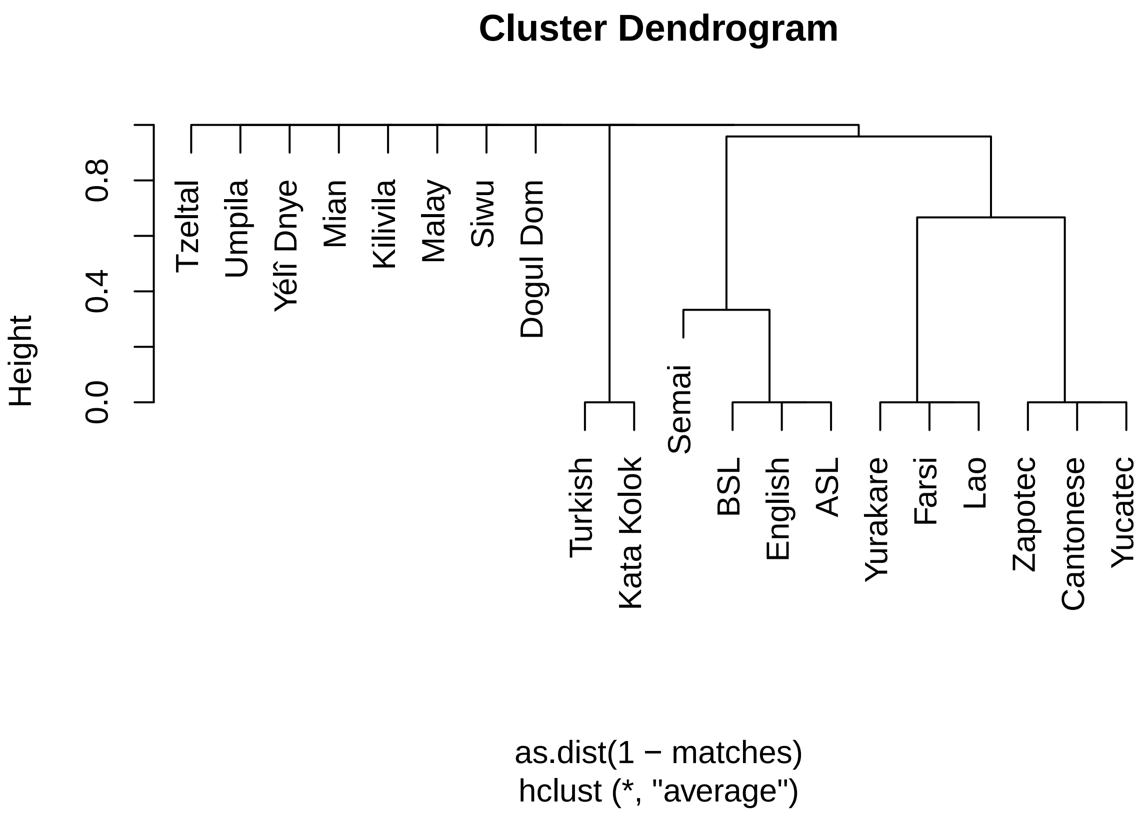
Languages with unique rankings:

```
names(which(rowSums(matches)==1))
```

```
## [1] "Dogul Dom" "Siwu"      "Malay"     "Kilivila"  "Mian"      "Yélî Dnye"
## [7] "Umpila"    "Tzeltal"
```

Use hierarchical clustering to identify unique rankings:

```
hc = hclust(as.dist(1-matches),method = "average")
plot(hc)
```



Number of unique rankings:

```
trueUniqueRanks = length(unique(cutree(hc,h = 0)))
```

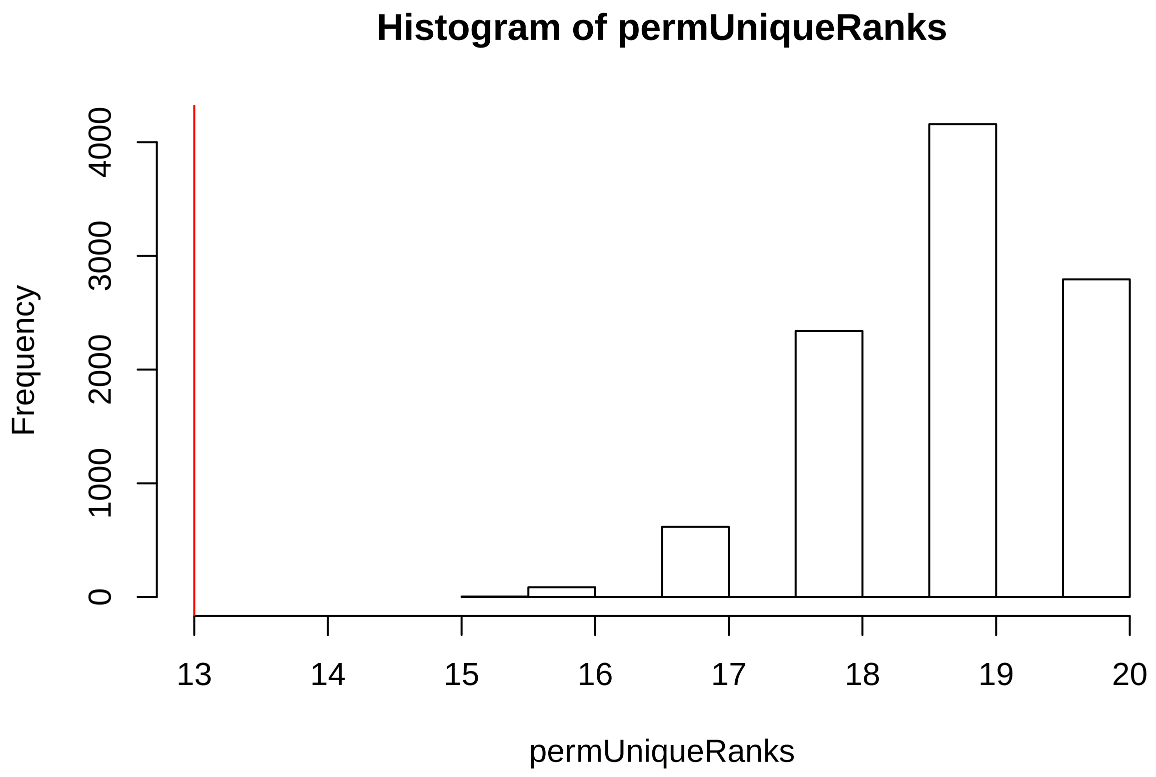
There are 13 unique rankings. We can calculate the probability of seeing this many rankings due to chance. However, an exact solution is difficult because of the missing data. Instead, we can just use permutation to sample the space of possibilities.

```
permWithinRows = function(m){
  # Permute values within rows,
  # keeping NAs in place
  n = colnames(m)
  m = t(apply(m,1,function(X){
    if(sum(is.na(X))>0){
      return(append(sample(X[!is.na(X)]),
                    NA,which(is.na(X))-1))
    } else{
      return(sample(X))
    }
  }))
  colnames(m) = n
  return(m)
}

permRanks = function(){
  matches = compatibleMatches(permWithinRows(meanCodability))
  hc = hclust(as.dist(1-matches),method = "average")
  length(unique(cutree(hc,h = 0)))
}
```

```
permUniqueRanks = replicate(10000,permRanks())
```

Compare true number of unique ranks to the permuted distribution:

```
hist(permUniqueRanks,
     xlim=range(c(permUniqueRanks,trueUniqueRanks)))
abline(v=trueUniqueRanks,col=2)
```



```
p = sum(permUniqueRanks<=trueUniqueRanks)/length(permUniqueRanks)
z = (trueUniqueRanks-mean(permUniqueRanks))/sd(permUniqueRanks)
```

The observed number of unique compatible ranks is less than would be expected by chance (z = -6.4586029, $p < 0.001$).

Aristotelian hierarchy

How compatible are the rank orderings with the Aristotelian hierarchy? This question is complicated by the fact that some languages do not have data for some domains, so each langauge needs to be assessed only comparing to those domains where data is available. The domains of colour and shape are also collapsed under the category of "Sight".

We're using Spearman's footrule to measure distance between rankings.

```
aristotelian.order =
c("Sight","Sight", "Sound", "Touch","Taste","Smell")

# Convert domains to senses (collapse colour and shape to sight)
meanCodability.Senses = meanCodability
colnames(meanCodability.Senses) = c("Sight","Sight","Touch",'Taste',"Smell","Sound")


spearmanFootrule = function(v1,v2){
  # Version of Spearmans' footrule that can handle
  # Multiple tied ranks
  sum(sapply(seq_along(v1), function(i) min(abs(i - (which(v2 == v1[i]))))))
}

spearmanFootruleFromAristotelanOrder = function(X){
  rk = names(sort(X,decreasing = T,na.last = NA))
  # filter aristotelian.order list
  ao = aristotelian.order[aristotelian.order %in% rk]
  rk.rank = as.numeric(factor(rk,levels=unique(ao)))
  ao.rank = as.numeric(factor(ao,levels=unique(ao)))
  spearmanFootrule(rk.rank,ao.rank)
}

# Compare each language to the Aristotelian hierarchy
trueNumEdits = apply(meanCodability.Senses,1,spearmanFootruleFromAristotelanOrder)
t(t(trueNumEdits))
```

```
##             [,1]
## English        2
## Farsi          9
## Turkish        9
## Dogul Dom     13
## Siwu          13
## Cantonese      7
## Lao            9
## Malay          4
## Semai          2
## Kilivila       7
## Mian           4
## Yélî Dnye     11
## Umpila        15
## Tzeltal        7
## Yucatec        7
## Zapotec        7
## Yurakare       5
## ASL            2
## BSL            2
## Kata Kolok     7
```

```
mean(trueNumEdits)
```

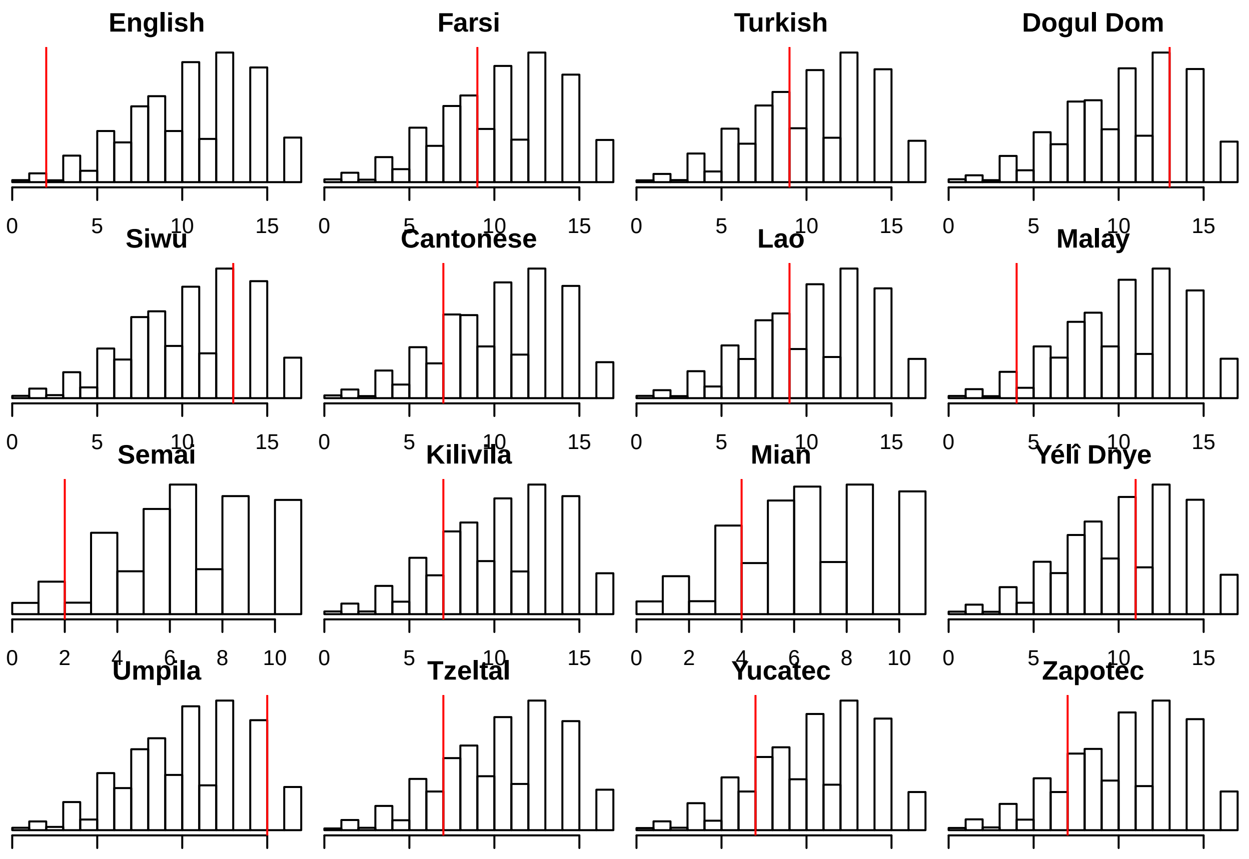
```
## [1] 7.1
```

Get expected Spearman's footrule for each langage when permuting codability scores within languages (maintaining missing domain data).

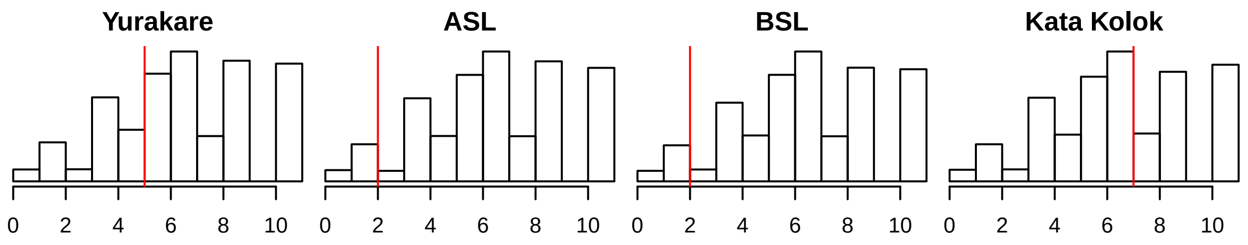
```
set.seed(1290)
permutedNumEdits = replicate(10000,
    apply(permWithinRows(meanCodability.Senses),1,
      spearmanFootruleFromAristotelanOrder))
```

Analyse the difference:

```
par(mfrow=c(4,4),mar=c(1,0,2,0))
for(i in 1:length(trueNumEdits)){
  hist(permutedNumEdits[i,],
     xlim=range(c(permutedNumEdits[i,],trueNumEdits[i])),
     main=names(trueNumEdits)[i],
     yaxt='n')
abline(v=trueNumEdits[i],col=2)
}
```



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



```
p = sapply(1:length(trueNumEdits), function(i){
  sum(permutedNumEdits[i,]<=trueNumEdits[i])/ncol(permutedNumEdits)
})
names(p) = names(trueNumEdits)
```

Table of p-values of each language, showing the probaility of the permuted distances being smaller or equal to the true distance:

```
t(t(sort(p)))
```

```
##                [,1]
## English    0.0134
## Malay      0.0497
## Semai      0.0633
## BSL        0.0675
## ASL        0.0688
## Tzeltal    0.1757
## Cantonese  0.1765
## Yucatec    0.1791
## Zapotec    0.1803
## Kilivila   0.1839
## Mian       0.2033
## Yurakare   0.2759
## Farsi      0.3841
## Turkish    0.3857
## Lao        0.3875
## Yélî Dnye  0.5979
## Kata Kolok 0.6083
## Siwu       0.8040
## Dogul Dom  0.8071
## Umpila     0.9481
```

Note that these p-values do not ajust for multiple comparisons.

Skillings Mack test

This is similar to the Friedman test, but allows missing data.

```r
# Add the missing data back in
d4 = as.data.frame(d2[,c("Language",'domain','m')])
missing_data= rbind(
           c(Language="Yurakare",domain="Sound",m=NA),
           c("ASL","Sound",NA),
           c("BSL","Sound",NA),
           c("Kata Kolok","Sound",NA),
           c("Semai","Taste",NA),
           c("Mian","Taste",NA))
d4 = rbind(d4, missing_data)
d4$m = as.numeric(d4$m)

sm = capture.output(Ski.Mack(y = d4$m,
         groups = d4$domain,
         blocks = d4$Language,
         simulate.p.value = T,
         B = 10000))

print(sm[1:4])
```

```
## [1] ""
## [2] "Skillings-Mack Statistic =  32.927402 , p-value =  4e-06 "
## [3] "Note: the p-value is based on the chi-squared distribution with d.f. =  5 "
## [4] "Based on B =  10000 , Simulated p-value =  0.000000 "
```

The p-value is small, so we can reject the null hypothesis that the distribution of mean codability of domains are is the same across langauges. That is, there are some statistical hierarchies in the data.

We can now run post-hoc tests on pairs of domains to discover the hierarchies. I'm using the Nemenyi test, following Demšar (2006).

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. Journal of Machine learning research, 7(Jan), 1-30.

```r
ph = posthoc.friedman.nemenyi.test(
      y = d4$m,
      groups = d4$domain,
      blocks = d4$Language)
ph
```

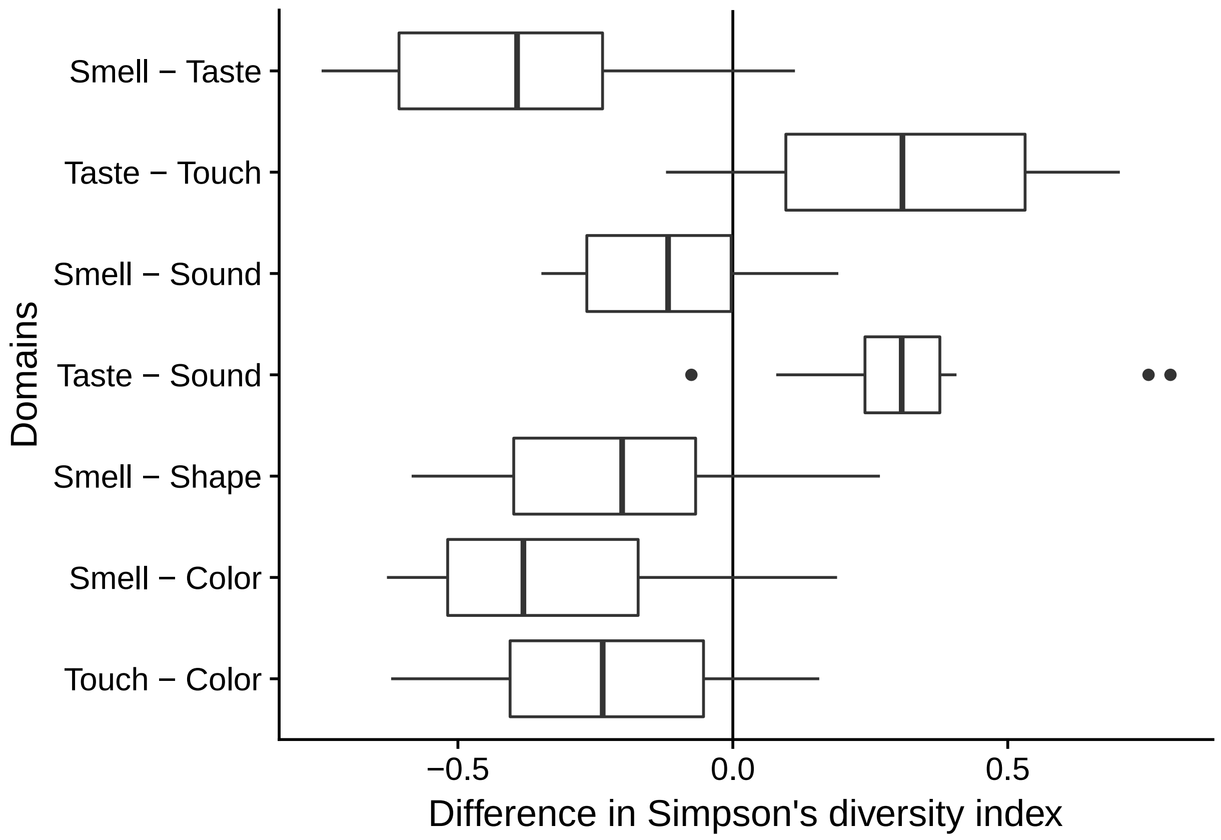
```
##
##  Pairwise comparisons using Nemenyi multiple comparison test
##              with q approximation for unreplicated blocked data
##
## data:  d4$m , d4$domain and d4$Language
##
##       Color   Shape   Sound   Touch   Taste
## Shape 0.53819 -       -       -       -
## Sound 0.20136 0.99163 -       -       -
## Touch 0.02837 0.75555 0.97398 -       -
## Taste 0.91341 0.07430 0.01258 0.00071 -
## Smell 3.8e-06 0.00702 0.04674 0.28044 1.0e-08
##
```

```
## P value adjustment method: none
```

Significant pairs:

```
sig.threshold = 0.05

sig = which(ph$p.value<sig.threshold,arr.ind = T, useNames = F)
sig[,1] = rownames(ph$p.value)[sig[,1]]
sig[,2] = colnames(ph$p.value)[as.numeric(sig[,2])]
```

```
diff = sapply(1:nrow(sig), function(i){
  meanCodability[,sig[i,1]] -
    meanCodability[,sig[i,2]]
})

cnt = sapply(1:nrow(sig), function(i){
  max(sum(meanCodability[,sig[i,1]] >
        meanCodability[,sig[i,2]],na.rm=T) ,
      sum(meanCodability[,sig[i,1]] <
        meanCodability[,sig[i,2]],na.rm=T))
})
cnt.poss = apply(meanCodability,2,function(X){sum(!is.na(X))})

colnames(diff) = paste(sig[,1],"-",sig[,2])

ggplot(as.data.frame.table(diff),
       aes(x=Var2,y=Freq)) +
  geom_hline(yintercept = 0) +
  geom_boxplot() +
  ylab("Difference in Simpson's diversity index") +
  xlab("Domains") +
  coord_flip()
```



Summarise the patterns alongside the statistics:

```
mdiff = colMeans(diff,na.rm = T)
direction = c(">","<")[1+(mdiff>0)]
ruleLables = sapply(1:nrow(sig), function(i){
  if(mdiff[i]>0){
    return(paste(sig[i,2],"<",sig[i,1]))
  } else{
    return(paste(sig[i,1],"<",sig[i,2]))
  }})
ruleP = ph$p.value[ph$p.value<sig.threshold & !is.na(ph$p.value)]
ruleP = signif(ruleP,2)
ruleP[ruleP<0.001] = "< 0.001"
paste(ruleLables,"(p =",ruleP,",",
      cnt,"/",cnt.poss,"languages)")
```

```
## [1] "Touch < Color (p = 0.028 , 17 / 20 languages)"
## [2] "Smell < Color (p = < 0.001 , 19 / 20 languages)"
## [3] "Smell < Shape (p = 0.007 , 18 / 20 languages)"
## [4] "Sound < Taste (p = 0.013 , 13 / 18 languages)"
## [5] "Smell < Sound (p = 0.047 , 12 / 20 languages)"
## [6] "Touch < Taste (p = < 0.001 , 16 / 16 languages)"
## [7] "Smell < Taste (p = < 0.001 , 17 / 20 languages)"
```

Linearity test

We use algorithms from ethology for converting a matrix of wins and losses in dyadic interactions into a consistent linear hierarchy. That is, we treat domains as "individuals" and see in how many languages a domain "dominantes" (more codable than) another.

Build a dominance matrix:

```
domain.labels = c("Color","Shape","Sound","Touch","Taste","Smell")

m = apply(expand.grid(domain.labels,domain.labels),1, function(X){
  #gt = d2[d2$domain==X[1],]$m > d2[d2$domain==X[2],]$m
  gt = meanCodability[,X[1]]>meanCodability[,X[2]]
  sum(gt,na.rm=T)
})
m = matrix(m,nrow=length(domain.labels), ncol=length(domain.labels))
rownames(m) = domain.labels
colnames(m) = domain.labels
m
```

```
##       Color Shape Sound Touch Taste Smell
## Color     0    16    16    17     4    19
## Shape     4     0    12    14     4    18
## Sound     0     4     0     8     1    12
## Touch     3     6     8     0     2    17
## Taste    14    14    13    16     0    17
## Smell     1     2     4     3     1     0
```

In the table above, e.g. Colour is more codable than Smell in 19 languages.

Run the linearity test:

```
set.seed(3298)
linear.hierarchy.test(m)
```

```
##
## Linearity test: Landau's h
## ==========================
##
##
##  Number of individuals: 6  ( 15 dyads)
##  Total number of interactions: 270
##
##  Descriptive table of dyadic relationships:
##             Frequency Percentage
## Unknown             0    0.00000
## Tied                1    6.66667
## One-way             1    6.66667
## Two-way            14   93.33333
## Significant        11   73.33333
##
##  Landau's h index: 0.971429
##  E(h): 0.428571
##  Var(h): 0.083333
##  Improved Landau's h: 0.971429
##  Number of randomizations: 9999
##  P-value (right): 1e-04
##  P-value (left): 1
##
##
## Linearity test: Kendall's rho
## ==============================
##
##  Number of circular dyads: 0.25
##  Expected number of circular dyads: 5
##  Maximum number of circular dyads: 8
##  Kendall's rho index: 0.96875
## Cannot computed since n<10.
```

The statistic is significant, suggesting that there is some consistency to the dominance ranking. Remember, this is just based on 'wins' being bigger than 'losses', it doesn't take into account the significance of the differences.

Find the most consistent order using the I&SI algorithm:

```
# Needed to fix one of the methods
source("DyaDA_ISIMethod2.R")
isi = ISI.method2(m,names = rownames(m))
isi
```

```
##
## Ordering a dominance matrix: I&SI Method
##
##  Call:
##  ISI.method2(X = m, names = rownames(m))
##
##  Initial order:  Color Shape Sound Touch Taste Smell
##
##       Color Shape Sound Touch Taste Smell
## Color     .    16    16    17     4    19
## Shape     4     .    12    14     4    18
## Sound     .     4     .     8     1    12
## Touch     3     6     8     .     2    17
## Taste    14    14    13    16     .    17
## Smell     1     2     4     3     1     .
##
##  Initial number of inconsistencies:  4
##  Initial strength of incosistencies:  4
##
##  Inconsistent dyads and strength of inconsistencies:
##   Individual.i Individual.j Strength.Inconsistency
## 1       Ind. 1       Ind. 5                      4
## 2       Ind. 2       Ind. 5                      3
## 3       Ind. 3       Ind. 5                      2
## 4       Ind. 4       Ind. 5                      1
##
## ***********************************************************
##  Final order:  Taste Color Shape Touch Sound Smell
##
##       Taste Color Shape Touch Sound Smell
## Taste     .    14    14    16    13    17
## Color     4     .    16    17    16    19
## Shape     4     4     .    14    12    18
## Touch     2     3     6     .     8    17
## Sound     1     .     4     8     .    12
## Smell     1     1     2     3     4     .
##
##  Final number of inconsistencies:  0
##  Final strength of inconsistencies:  0
##
##  Inconsistent dyads and strength of inconsistencies:
## [1] NA
```

The linear order most consistent with the domain ranking within languages is (according to the I&SI algorithm):

```
best.order= rownames(isi$dataFinal)
paste(best.order,collapse =">")
```

```
## [1] "Taste>Color>Shape>Touch>Sound>Smell"
```

Again, you can't interpret this as "Touch is significantly more codable than sound", it's just an order that is consistent with each domain being more codable in more languages than the last.

```
actual.orders = apply(meanCodability,1,function(X){
  x = rev(colnames(meanCodability)[order(X)])
  all(x == best.order[best.order %in% x])
           })
sum(actual.orders)
```

```
## [1] 0
```

```
which(actual.orders)
```

```
## named integer(0)
```

There are no languages that are compatible with this order, so it's a poor approximation.