rphenoscate: Tutorial 3

Diego S. Porto, Sergei Tarasov, Caleb Charpentier, and SCATE team

August, 2022

In this third tutorial, we will be using *rphenoscate* to build a synthetic phylogenetic character matrix from annotated phenotypes of Ostariophysi fishes available in the Phenoscape Knowledgebase (KB).

Particularly, we will be using the data set of Dillman et al. (2016):10.1111/cla.12127 as a benchmark, which consists of 173 species of Characiform fishes and 463 morphological characters. We will be comparing the 'original' data set (i.e., character matrix retrieved from the original paper) with an 'inferred' data set built from phenotype annotations of the same study.

The aim of this tutorial is to demonstrate that synthetic matrices build with *rphenoscate* retain most of the phylogenetic and semantic properties of standard character matrices thus allowing not only phylogenetic inferences but also semantic inferences with ontology annotated morphological data.

STEP 1. Loading the packages.

First, let's load *rphenoscape* and *rphenoscate*.

```
library("rphenoscape")
library("rphenoscate")
```

Let's load some other packages that might be useful as well. In particular, *TreeTools* allows us to import character matrices in NEXUS format, extract character statements from matrix annotations, and calculate some information metrics from phylogenetic characters. *TreeDist* allows us to calculate some information metrics from phylogenetic trees.

```
library("TreeTools")
library("TreeDist")
library("igraph")
library("ggplot2")
library("viridis")
```

STEP 2. Assembling the data set from a given study.

First, let's retrieve the phylogenetic character matrix from Dillman et al (2016).

```
# Get list of all annotated studies available at Phenoscape KB.
studies <- pk_get_study_list()

# Get a particular study # (Change this part to get a particular study).
study <- studies$id[studies$label == 'Dillman et al. (2016)']

# Get NeXML data.
selected_study <- pk_get_study_xml(study)

# Build the original character matrix.
```

```
char.mat <- RNeXML::get_characters(selected_study[[1]])

# Get rownames and colnames from data set.
row.mat <- rownames(char.mat)
col.mat <- colnames(char.mat)
```

Sometimes, data sets might have rows and/or columns represented as IRIs or character IDS instead of actual human-readable labels. Let's check for that.

```
# Check rownames and colnames from data set.
row.mat[1:3]
```

```
## [1] "Abramites hypselonotus" "Anostomoides laticeps"  "Anostomus anostomus"
```

```
col.mat[1:3]
```

```
## [1] "Alignment of border between frontal and parietal on dorsal surface of cranium"
## [2] "Alignment of dorsal process of fourth epibranchial"
## [3] "Alignment of ectopterygoid"
```

If rownames and/or colnames are IRIs and/or character IDs, then run the code below to extract the original labels.

```
# Get metadata the original character matrix.
selected_study.meta <- pk_get_study_meta(selected_study)

# Get rownames and colnames from data set.
row.mat <- selected_study.meta[[1]]$id_taxa$label[
  match(rownames(char.mat),selected_study.meta[[1]]$id_taxa$otu)]
col.mat <- selected_study.meta[[1]]$id_entities$label[
  match(colnames(char.mat),selected_study.meta[[1]]$id_entities$char)]
```

STEP 3. Getting all phenotype annotations from a given study.

Now, let's retrieve all phenotype annotations from the Phenoscape KB corresponding to characters statements in the Dillman et al. (2016) study. As mentioned before, Phenoscape KB contains expert annotated phenotypes for several studies of Ostariophyisi fishes. In each study, character statements can be represented as a semantic statements using a formal syntax, the Entity-Quality syntax (EQ). Character statements can be annotated with terms from an anatomy ontology (i.e., for the anatomical entities), in this case UBERON, and a phenotype ontology (i.e., for the qualities of entities), in this case PATO, using the software Phenex (Balhoff et al. 2014: dx.doi.org/10.1186/2041-1480-5-45)

```
# Get all phenotype data from the study.
phenotypes <- get_phenotypes(study = study)

# Convert data to a phenotype object (Warning: 10~30 min).
selected_study.obj <- as.phenotype(phenotypes, withTaxa = T)

# Get all character information from the phenotype object.
selected_study.chars <- chars(selected_study.obj)

# Filter characters based on target study.
selected_study.chars.filter <- lapply(selected_study.chars, function(x)
  x[grepl(selected_study.chars$study.id, pattern = study)] )
```

The phenotype annotations retrieved from the Phenoscape KB can be found in the 'selected_study.obj'. Each element of this list contains information for a given phenotype annotation, including phenotype ID,

label, taxonomic distribution, etc. The list 'selected_study.chars.filter' contains information for character and character state data associated with each phenotype annotation, including character and character state labels as in the reference study, in this case, Dillman et al. (2016).

STEP 4. Obtaining mutual exclusivity information.

Now that we have all phenotype annotations from Dillman et al. (2016), we need to assemble them in a character matrix. The core idea here is that each phenotype annotation (i.e, a semantic statement expressed through the EQ syntax) might correspond to an individual character statement about a quality of a taxon or group of taxa. In simple terms, characters in a standard character matrix can be seen as collections of two or more semantic statements. For example, the semantic statements 'anatomical region and (anterior_to some pelvic fin) circular' and 'anatomical region and (anterior_to some pelvic fin) concave' compose the phylogenetic character 'Form of prepelvic region of ventral body surface'. To cluster phenotype annotations into characters, *rphenoscate* parses information from the Phenoscape KB, including the taxonomic distribution of annotations, to classify pairs of phenotypes into exclusivity classes. For more details, refer to the documentation of the 'mutually_exclusive' function. In a nutshell, 'strong_exclusivity' indicate putative pairs of mutually exclusive phenotypes that might correspond to alternative character states of a phylogenetic character. Alternatively, 'strong_compatibility' indicate phenotypes that describe redundant information, therefore, states that should be fused.

Let's obtain the exclusivity classes for all phenotypes annotated in the Dillman et al. (2016).

```
# Determine mutual exclusivity (Warning: 10~30 min).
exclusivity <- mutually_exclusive(phenotypes$id, studies = study)
```

STEP 5. Building characters from phenotype descriptions.

Then, let's use the information on exclusivity classes to extract clusters of semantic statements and build putative characters.

```
# Extract character clusters.
CH <- extract.chars(exclusivity)

# Build characters.
CH.selected_study <- build.chars(CH, selected_study.chars.filter)
```

The object 'CH.selected_study' is a list including information on putative characters inferred from the exclusivity classes obtained in the previous step. 'solved' comprises phenotype annotations that were clustered as characters as inferred from evidence of 'strong_compatibility' whereas 'unsolved' comprises phenotype annotations that could not be assigned unambiguously to a particular character.

```
# Quick checks.
# Solved characters.
CH.selected_study$solved$chars[1:3]
```

```
## $`Form of attachment of hyohyoidei abductores on urohyal`
## [1] "narrow insertion to anteriormost portion of bone only"
## [2] "longitudinally expanded with broad insertion across entire ventral surface and lateral margins
##
## $`Alignment of border between frontal and parietal on dorsal surface of cranium`
## [1] "border aligned nearly directly transversely or with moderate anteromedial alignment"
## [2] "medial portion of border posteromedially aligned, more so in larger individuals"
##
## $`Contact of contralateral parietals along dorsal midline`
## [1] "distinct contact in adults of various sizes"
```

```
## [2] "no contact or very minimal contact anteriorly in large specimens only"
```

```
CH.selected_study$solved$tokens[1:3]
```

```
## $`Form of attachment of hyohyoidei abductores on urohyal`
## [1] "0" "1"
##
## $`Alignment of border between frontal and parietal on dorsal surface of cranium`
## [1] "0" "1"
##
## $`Contact of contralateral parietals along dorsal midline`
## [1] "0" "1"
```

```
# Number of solved characters.
length(CH.selected_study$solved$chars)
```

```
## [1] 422
```

```
# Unsolved characters.
CH.selected_study$unsolved$chars[1:3]
```

```
## $`Form of A3 portion (=pars stegalis) of adductor mandibulae`
## [1] "with extensive origin along lateral surface of mesopterygoid and metapterygoid and broad inserti
##
## $`Form of posterior portion of A2 portion (= pars mallaris) of adductor mandibulae`
## [1] "extends medial of levator arcus palatini with partial origin on hyomandibula"
##
## $`Form of anterior portion of A2 (= pars mallaris) and A3 (=pars stegalis) portions of adductor mandi
## [1] "muscles not dorsally extensive, contact not as in state 1"
```

```
CH.selected_study$unsolved$tokens[1:3]
```

```
## $`Form of A3 portion (=pars stegalis) of adductor mandibulae`
## [1] "0"
##
## $`Form of posterior portion of A2 portion (= pars mallaris) of adductor mandibulae`
## [1] "0"
##
## $`Form of anterior portion of A2 (= pars mallaris) and A3 (=pars stegalis) portions of adductor mandi
## [1] "0"
```

```
# Number of unsolved characters.
length(CH.selected_study$unsolved$chars)
```

```
## [1] 29
```

STEP 6. Building character matrices.

Finally, let's build a synthetic character matrix based on the characters inferred from exclusivity information.

```
# Get set of taxa.
tax <- unique(row.mat)

# Build character matrix for inferred characters.
char.mat.infer <- build.matrix(tax, selected_study.obj,
                               selected_study.chars.filter, CH.selected_study)
```

STEP 7. Phylogenetic inferences with the original and synthetic character matrices.

Now, we can finally compare the original character matrix and the synthetic one to assess how well semantic phenotypes carry the phylogenetic properties of morphological data as observed in standard character matrices.

```
# Copy matrix objects.
m1 <- char.mat
m2 <- char.mat.infer
```

Sometimes, data sets might contain some duplicated rows (taxa) due to taxonomic changes or revisional studies. Let's check for that.

```
# Check for duplicates.
any(duplicated(row.mat))
```

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

If duplicated rows were found, then taxa must be merged and non-matching character states must be coded as polymorphisms. If that is the case, then run the code below.

```
# Get duplicated taxa.
dup <- row.mat[duplicated(row.mat)]

# Merge duplicated taxa.
for(i in 1:length(dup)){

  x <- grep(row.mat, pattern = dup[i])

  y <- apply(m1[x,],2, function(x) paste0(unique(x), collapse = " and "), simplify = F)

  m1[x[1],] <- y
  m1 <- m1[-x[2],]

}
```

Then, let's format taxon labels, character matrices entries, and command blocks to perform Bayesian Inference in MrBayes. First, let's format character matrices.

```
# Prepare data for phylogenetic analyses.

# Reorganize taxon labels (for MrBayes).
tax.lab <- gsub(tax, pattern = "\\?", replacement = "")
tax.lab <- gsub(tax.lab, pattern = "\\(", replacement = "")
tax.lab <- gsub(tax.lab, pattern = "\\)", replacement = "")
tax.lab <- gsub(tax.lab, pattern = " ", replacement = "_")
rownames(m1) <- rownames(m2) <- tax.lab

# Recode polymorphisms from the original matrix (for MrBayes).
m1[is.na(m1)] <- "?"
m1 <- apply(m1, 2, function(x) gsub(x, pattern = "NA", replacement = "?"))
m1 <- apply(m1, 2, function(x) gsub(x, pattern = " and ", replacement = ","))
m1 <- apply(m1, 2, function(x) gsub(x, pattern = "^\\?,", replacement = ""))
m1 <- apply(m1, 2, function(x) gsub(x, pattern = ",\\?$", replacement = ""))
m1 <- apply(m1, 2, function(x) gsub(x, pattern = "^(\\d),", replacement = "(\\1,"))
m1 <- apply(m1, 2, function(x) gsub(x, pattern = ",(\\d)$", replacement = ",\\1)"))
```

Then, set parameters for the MCMC. These can be changed if desired.

```
# Set MCMC parameters (change if desired).
gens = 5000000
runs = 2
chains = 4

# Build MrBayes command block.
block <- readLines("./data/block.nex")
block <- gsub(block, pattern = "&&ngens&&", replacement = format(gens, scientific = F))
block <- gsub(block, pattern = "&&nruns&&", replacement = runs)
block <- gsub(block, pattern = "&&nchains&&", replacement = chains)
block1 <- gsub(block, pattern = "&&filename&&", replacement = "test1")
block2 <- gsub(block, pattern = "&&filename&&", replacement = "test2")

rm(gens, runs, chains)
```

Then, let's export the data.

```
# Create directory to run analyses.
dir.create("mrbayes")

# Export nexus files.
write.nexus.data(as.matrix(m1), format = "standard", interleaved = F,
                 file = "./mrbayes/test1.nex")
write.nexus.data(as.matrix(m2), format = "standard", interleaved = F,
                 file = "./mrbayes/test2.nex")

# Write MrBayes command block.
write(block1, file = "./mrbayes/block1.nex")
write(block2, file = "./mrbayes/block2.nex")
```

STEP 8. Bayesian Inferences on MrBayes.

Run the analyses of 'block1.nex' and 'block2.nex' from within the 'mrbayes' folder.

STEP 9. Evaluating semantic properties of data sets.

In the last two steps of this tutorial we will assess the semantic and (phylogenetic) informational properties of the original and synthetic matrices. First, let's examine the semantic properties of data. For that, we will be using some functions from *rphenoscape*, *TreeTools*, and *TreeDist*. The first property that we are going to examine is the average intra-cluster semantic similarity among phenotypes included in the same character as in the original study and the ones included in the clusters inferred with *rphenoscate*. It is expected that semantic statements describing alternative states of the same character should exhibit overall high average semantic similarity since they refer to the same anatomical entity and describe the same type of phenotype quality (e.g., shape, color). Therefore, the distribution of average intra-cluster semantic similarity values for inferred *rphenoscate* clusters should match closely with that of the original data set.

```
# Create a folder to store pdfs.
dir.create("PDF")

## ASSESSMENT 1: Average intra-phenotype-cluster semantic similarity.

# Get clusters of phenotypes based on data from the original matrix.
x <- match(selected_study.chars.filter$character.label, col.mat)
```

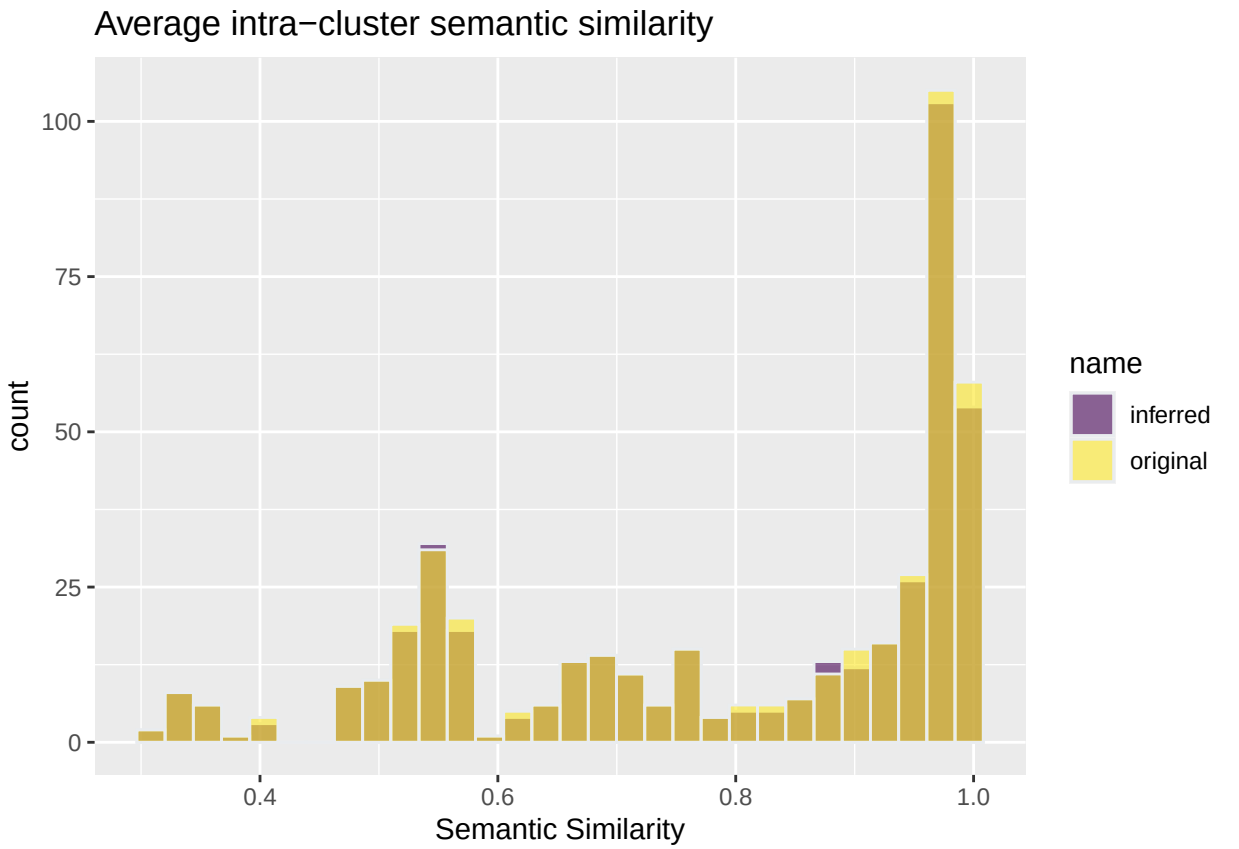
```r
y <- split(selected_study.chars.filter$phenotype.id, f = as.factor(x))

# Calculate average ss among phenotypes in each cluster from the original matrix.
w <- lapply(y, function(x) jaccard_similarity(subsumer_matrix(terms = x)) )
z <- sapply(w, function(x) mean(x[upper.tri(x)]) )
z <- z[!is.na(z)]

# Calculate average ss among phenotypes in each cluster from the inferred matrix.
u <- lapply(lapply(CH.selected_study$solved$clusters, unlist), function(x)
  jaccard_similarity(subsumer_matrix(terms = x)) )
u <- sapply(u, function(x) mean(x[upper.tri(x)]) )
u <- u[!is.na(u)]

# Quick summary of results.
summary(z)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3066  0.5804  0.8996  0.7948  0.9747  0.9945
```

```r
summary(u)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3066  0.5801  0.8906  0.7937  0.9745  0.9945
```

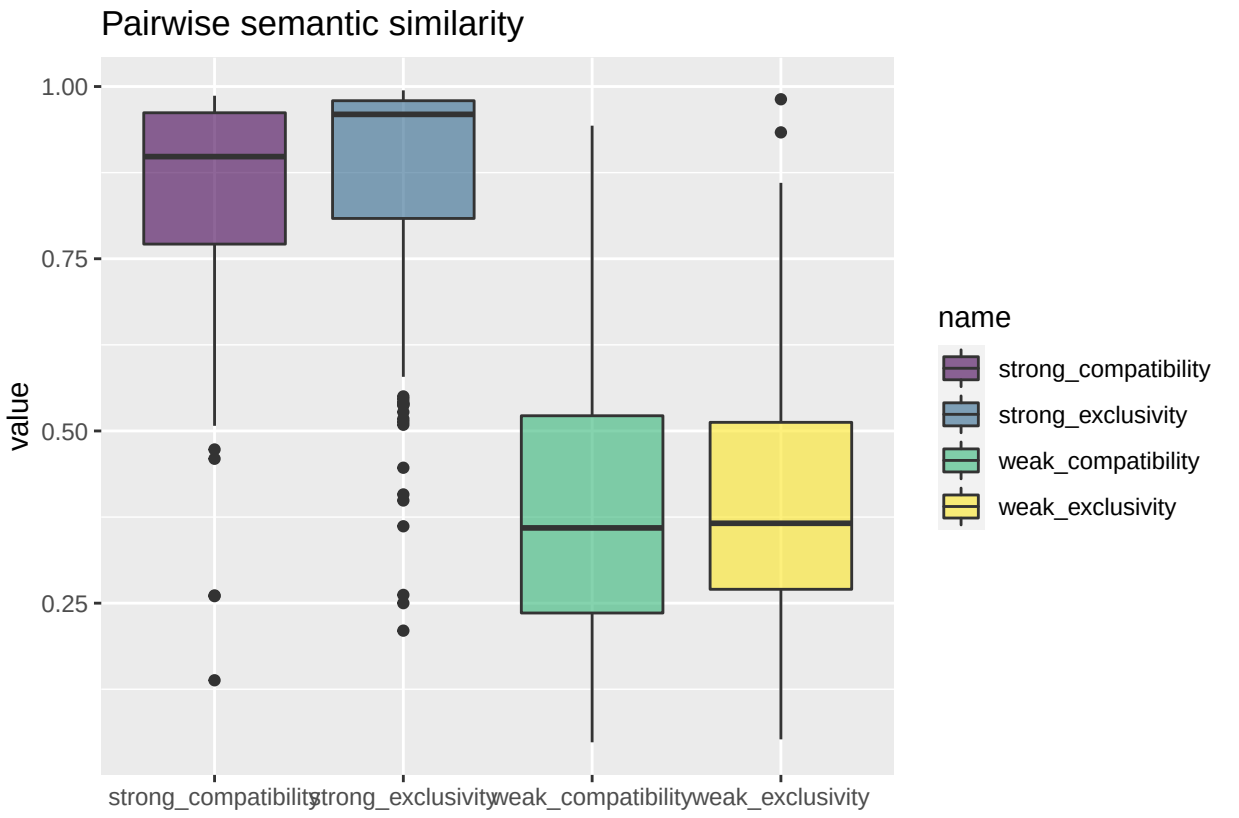
```r
# Build a data.frame.
DF1 <- data.frame(value = c(z,u), name = c(rep("original", length(z)),
                                           rep("inferred", length(u))))

# Make a plot (histogram).
ggplot(data = DF1, aes(x = value, fill = name)) +
  geom_histogram(color = "#e9ecef", alpha = 0.6, position = 'identity') +
  scale_fill_viridis(discrete = TRUE, alpha = 0.6) +
  ggtitle("Average intra-cluster semantic similarity") + xlab("Semantic Similarity")
```



The second property that we are going to examine is the average intra-exclusivity-class semantic similarity for all pairwise comparisons among phenotype statements. The aim here is to evaluate inferred characters from *rphenoscate* regarding the assertion that semantic statements describing alternative states of the same character should exhibit overall high average semantic similarity. As stated previously, the function 'mutually_exclusive' from *rphenoscate* classify pairs of phenotypes into exclusivity classes where 'strong_exclusivity' indicates strong evidence for phenotypes representing alternative states of the same character whereas 'strong_compatibility' indicates strong evidence for phenotypes representing states that are compatible, thus should be fused. Therefore, if groups of phenotypes comprising the same character are expect to have higher semantic similarity, then pairs of phenotypes in the 'strong_exclusivity' and 'strong_compatibility' are expected to have higher semantic similarity as well.

```
## ASSESSMENT 2: Average intra-exclusivity-class semantic similarity.

# lv3: 'inconclusive_evidence' was omitted since no pairs in that class were recovered.

# Extract all phenotypes in each exclusivity class #
lv5 <- exclusivity$dataframe[
  exclusivity$dataframe$mutual_exclusivity == "strong_exclusivity",]
lv4 <- exclusivity$dataframe[
  exclusivity$dataframe$mutual_exclusivity == "weak_exclusivity",]
lv2 <- exclusivity$dataframe[
  exclusivity$dataframe$mutual_exclusivity == "weak_compatibility",]
lv1 <- exclusivity$dataframe[
  exclusivity$dataframe$mutual_exclusivity == "strong_compatibility",]

# Reorder all elements.
lv5 <- lv5[sample(1:dim(lv5)[1],dim(lv5)[1]),]
```

```r
lv4 <- lv4[sample(1:dim(lv4)[1],dim(lv4)[1]),]
lv2 <- lv2[sample(1:dim(lv2)[1],dim(lv2)[1]),]
lv1 <- lv1[sample(1:dim(lv1)[1],dim(lv1)[1]),]

# Get a sample of N phenotype pairs from each exclusivity class (change if necessary).
N <- 100
EX5 <- lv5[sample(1:dim(lv5)[1],N),]
EX4 <- lv4[sample(1:dim(lv4)[1],N),]
EX2 <- lv2[sample(1:dim(lv2)[1],N),]
EX1 <- lv1[sample(1:dim(lv1)[1],N),]

# Create temporary vectors to store values.
s5 <- numeric()
s4 <- numeric()
s2 <- numeric()
s1 <- numeric()
e5 <- character()
e4 <- character()
e2 <- character()
e1 <- character()

# Calculate ss for pairwise comparisons between phenotypes.
for(i in 1:N){

  w5 <- jaccard_similarity(terms = c(EX5$id.1[i], EX5$id.2[i]))
  w4 <- jaccard_similarity(terms = c(EX4$id.1[i], EX4$id.2[i]))
  w2 <- jaccard_similarity(terms = c(EX2$id.1[i], EX2$id.2[i]))
  w1 <- jaccard_similarity(terms = c(EX1$id.1[i], EX1$id.2[i]))

  s5 <- c(s5, w5[upper.tri(w5)])
  s4 <- c(s4, w4[upper.tri(w4)])
  s2 <- c(s2, w2[upper.tri(w2)])
  s1 <- c(s1, w1[upper.tri(w1)])

  e5 <- c(e5, as.character(EX5$mutual_exclusivity[i]))
  e4 <- c(e4, as.character(EX4$mutual_exclusivity[i]))
  e2 <- c(e2, as.character(EX2$mutual_exclusivity[i]))
  e1 <- c(e1, as.character(EX1$mutual_exclusivity[i]))

}

# Build a data.frame.
DF2 <- data.frame(value = c(s5, s4, s2, s1), name = c(e5, e4, e2, e1))

# Make a plot.
ggplot(data = DF2, aes(x = name, y = value, fill = name)) + geom_boxplot() +
  scale_fill_viridis(discrete = TRUE, alpha = 0.6) +
  ggtitle("Pairwise semantic similarity") + xlab("")
```




```
# Clean workspace.
rm(i, lv5, lv4, lv2, lv1, EX5, EX4, EX2, EX1, w5, w4, w2, w1, s5, s4, s2, s1, e5, e4, e2, e1)
```

STEP 10. Evaluating informational properties of data sets.

Now, let's examine the (phylogenetic) informational properties of the data. By that, we mean comparing the phylogenetic information content available for tree inference using the 'original' and 'inferred' character matrices. Systematists often construct character matrices in order to infer the phylogenetic relationships among taxa. In other words,the major goal of assembling character matrices is tree inference. Therefore, assessing the phylogenetic information content and tree topologies inferred from these matrices is a natural way to assess how useful synthetic character matrices can be for phylogeneticts in general.

First, let's measure the cladistic information content sensu Steel and Penny 2006 from all characters in the 'original' and 'inferred' character matrices. For that, we will use a function from *TreeTools* (Smith 2019:doi:10.5281/zenodo.3522725). For more discussions about information theory metrics in phylogenetics and tree space explorations see Steel and Penny (2006), Smith (2020): 10.1093/bioinformatics/btaa614, and Smith (2022):10.1093/sysbio/syab100.

```
## ASSESSMENT 1: Calculate phylogenetic information content from character matrices.
# Copy original character matrix objects.
cinfo1 <- char.mat
cinfo2 <- char.mat.infer

# Reorganize tokens and recode polymorphisms and missings.
cinfo1[is.na(cinfo1)] <- "?"
cinfo1 <- apply(cinfo1, 2, function(x) gsub(x, pattern = "(.) and (.)",
```

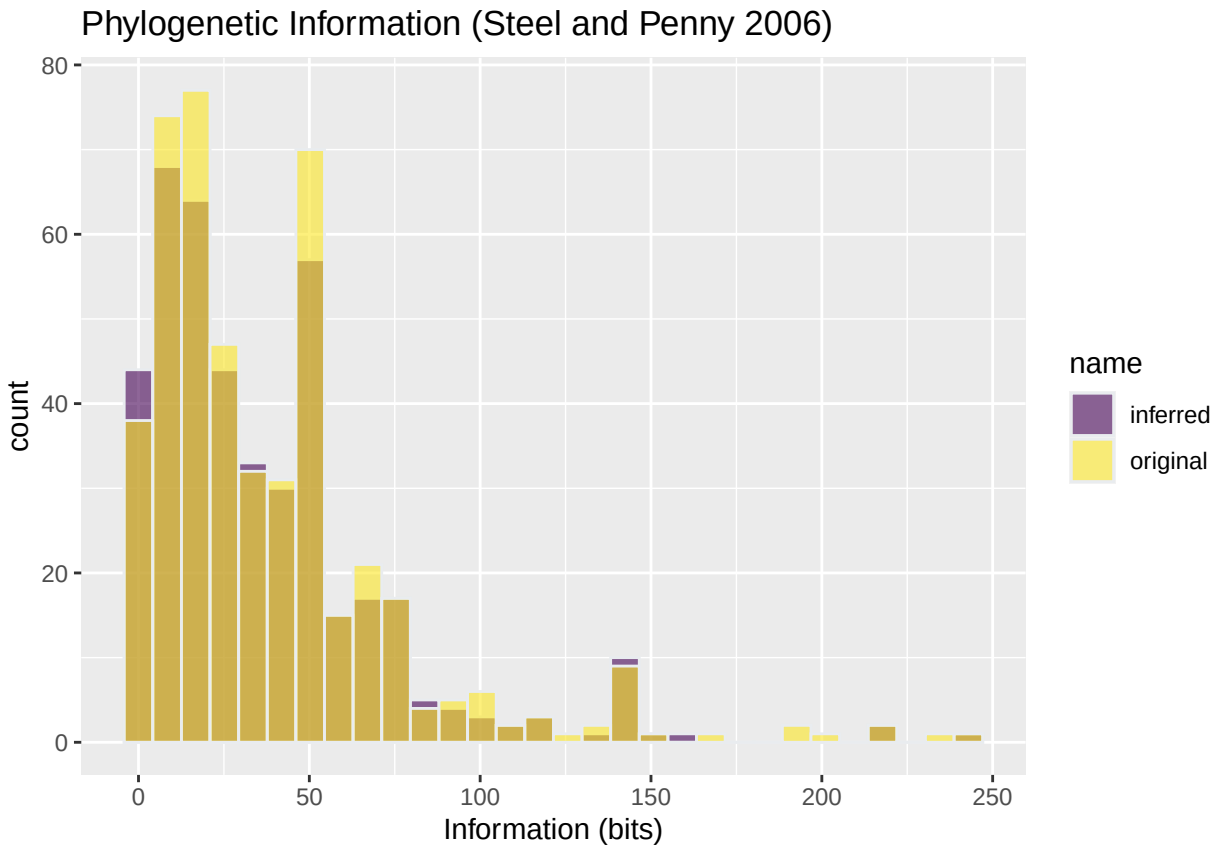
```
                                            replacement = "?"))
cinfo2 <- apply(cinfo2, 2, function(x) gsub(x, pattern = "(.),(.),(.),(.)",
                                            replacement = "?"))
cinfo2 <- apply(cinfo2, 2, function(x) gsub(x, pattern = "(.),(.),(.)",
                                            replacement = "?"))
cinfo2 <- apply(cinfo2, 2, function(x) gsub(x, pattern = "(.),(.)",
                                            replacement = "?"))
cinfo2 <- apply(cinfo2, 2, function(x) gsub(x, pattern = "^\\(",
                                            replacement = ""))
cinfo2 <- apply(cinfo2, 2, function(x) gsub(x, pattern = "\\)$",
                                            replacement = ""))

# Calculate information metrics per-character across the matrix.
cinfo1 <- apply(cinfo1, 2, function(x) CharacterInformation(x) )
cinfo2 <- apply(cinfo2, 2, function(x) CharacterInformation(x) )

# Summary.
summary(cinfo1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   13.27   27.32   38.78   51.84  242.89
```

```
summary(cinfo2)
```

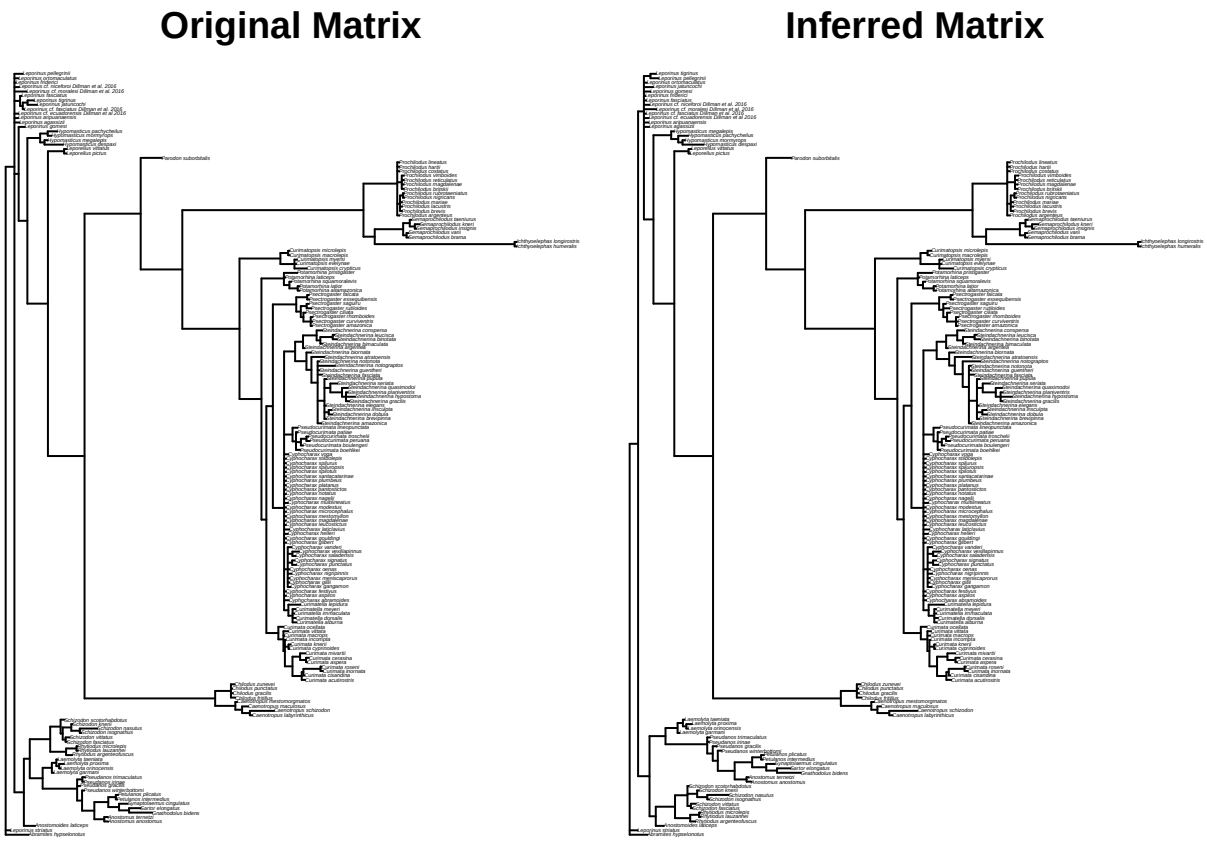
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   11.10   26.43   36.48   50.03  242.89
```

```
# Build data.frame.
DF3 <- data.frame(value = c(cinfo1, cinfo2), name = c(rep("original", length(cinfo1)),
                                                      rep("inferred", length(cinfo2))))

# Plot.
ggplot(data = DF3, aes(x = value, fill = name)) +
  geom_histogram(color = "#e9ecef", alpha = 0.6, position = 'identity') +
  scale_fill_viridis(discrete = TRUE, alpha = 0.6) +
  ggtitle("Phylogenetic Information (Steel and Penny 2006)") + xlab("Information (bits)")
```



Now, let's measure the phylogenetic information from splits in the MJ consensus trees obtained from the analyses of the 'original' and 'inferred' character matrices. For that, we will use functions from *TreeDist* (Smith 2020:doi:10.5281/zenodo.3528123). First, let's import the results from the Bayesian inferences and plot the consensus trees.

```
## ASSESSMENT 2: Calculate phylogenetic information from splits in tree topologies.

# Import consensus results from MrBayes.
tree1 <- read.nexus(file = "./mrbayes/test1.con.tre")
tree2 <- read.nexus(file = "./mrbayes/test2.con.tre")

# Import tree samples #
trees1.r1 <- read.nexus(file = "./mrbayes/test1.run1.t")
trees1.r2 <- read.nexus(file = "./mrbayes/test1.run2.t")
trees2.r1 <- read.nexus(file = "./mrbayes/test2.run1.t")
trees2.r2 <- read.nexus(file = "./mrbayes/test2.run2.t")

par(mfrow = c(1,2), mar = c(0.1,0.1,1.0,0.1))
# Plot consensus tree for the original matrix.
plot.phylo(tree1, cex = 0.2, main = "Original Matrix")

# Plot consensus tree for the 'inferred' matrix.
plot.phylo(tree2, cex = 0.2, main = "Inferred Matrix")
```



And then calculate the information metrics standardized by number of characters.

```
# Calculate total normalized 'intrinsic' phylogenetic information from splits.
paste("Original matrix: ", round(ClusteringInfo(tree1)/length(cinfo1),2))
```

```
## [1] "Original matrix:  10.13"
```

```
paste("Inferred matrix: ", round(ClusteringInfo(tree2)/length(cinfo2),2))
```

```
## [1] "Inferred matrix:  10.98"
```

```
# Calculate total normalized 'extrinsic' phylogenetic information from splits.
paste("Original matrix: ", round(SplitwiseInfo(tree1)/length(cinfo1),2))
```

```
## [1] "Original matrix:  8.86"
```

```
paste("Inferred matrix: ", round(SplitwiseInfo(tree2)/length(cinfo2),2))
```

```
## [1] "Inferred matrix:  9.64"
```

```
# Summaries.
# Splits: 'Intrinsic' phylogenetic information.
summary(ClusteringInfo(tree1, sum = F))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   15.74   15.74   27.44   45.99   54.10  172.96
```

```
summary(ClusteringInfo(tree2, sum = F))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   15.74   15.74   27.44   46.81   55.13  172.96
```

```
# Splits: 'Extrinsic' phylogenetic information.
summary(SplitwiseInfo(tree1, sum = F))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   8.414   8.414  21.308  40.207  49.310 170.463
```

```
summary(SplitwiseInfo(tree2, sum = F))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   8.414   8.414  21.308  41.086  50.374 170.463
```

Finally, let's visualize the posterior tree distributions inferred from both data sets. There are several alternatives for visualizing the posterior tree space (for more details see Smith 2022 for example). Here, let's keep it simple and just calculate the generalized Robinson-Foulds distances (Smith 2020) from tree topologies in the posterior distributions relative to a given MJ consensus tree and plot the resultant histograms. Let's compare each distribution to its own consensus tree and then each distribution with the consensus from the 'original' and 'inferred' data sets.

First, let's extract the post-burnins, join the chains, and organize the data.

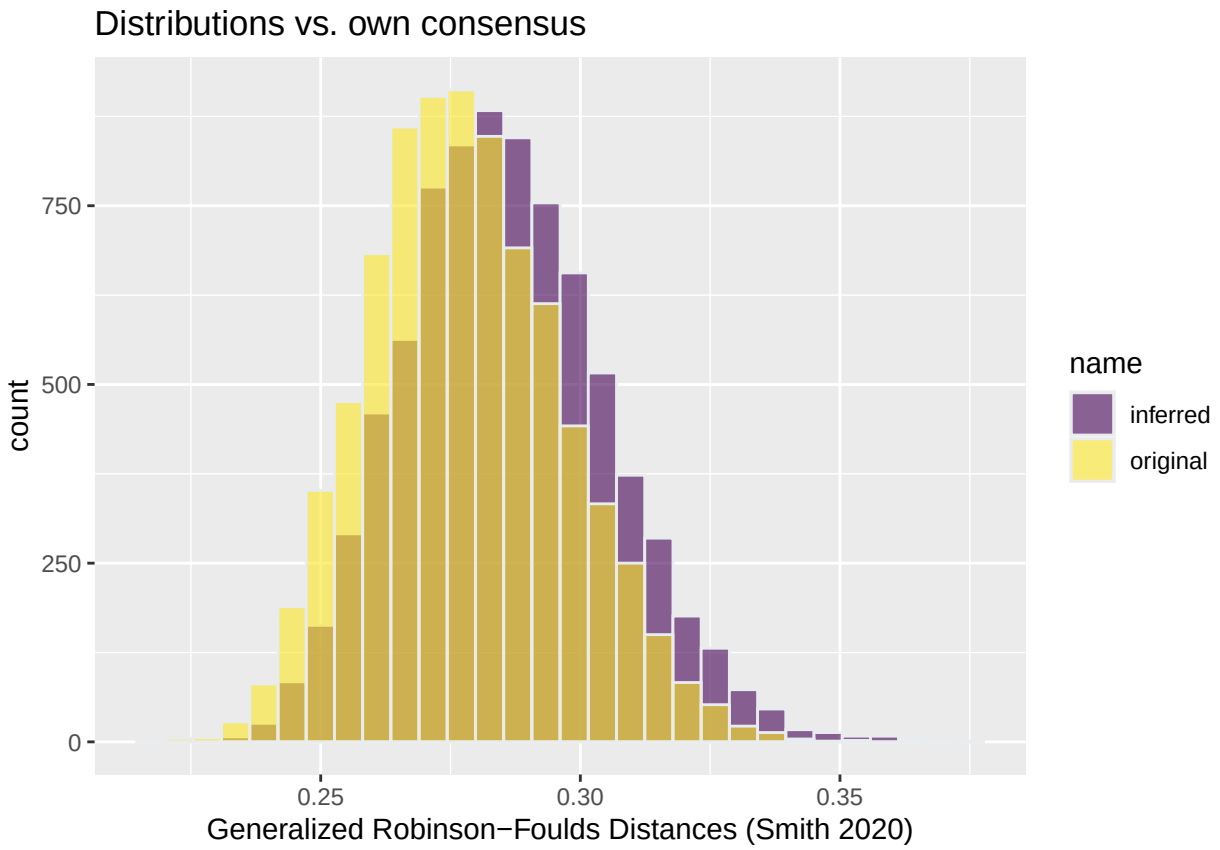
```
## ASSESSMENT 3: Calculate generalize Robinson-Foulds distances among trees.

# Set burnins.
trees1.r1 <- trees1.r1[-c(1:1001)]
trees1.r2 <- trees1.r2[-c(1:1001)]
trees2.r1 <- trees2.r1[-c(1:1001)]
trees2.r2 <- trees2.r2[-c(1:1001)]

# Organize labels.
names(trees1.r1) <- paste0("T", 1:length(trees1.r1), ".R1")
names(trees1.r2) <- paste0("T", 1:length(trees1.r2), ".R2")
names(trees2.r1) <- paste0("T", 1:length(trees2.r1), ".R1")
names(trees2.r2) <- paste0("T", 1:length(trees2.r2), ".R2")

# Join chains.
trees1 <- c(trees1.r1, trees1.r2)
trees2 <- c(trees2.r1, trees2.r2)
```

Then, calculate the generalized RF distances and organize data.

```
# Calculate generalized RF.
# Non-crossed: each consensus vs. its own distribution.
RF1 <- TreeDistance(tree1 = tree1, tree2 = trees1)
RF2 <- TreeDistance(tree1 = tree2, tree2 = trees2)

# Crossed: each consensus vs. the other distribution.
RF1.c <- TreeDistance(tree1 = tree2, tree2 = trees1)
RF2.c <- TreeDistance(tree1 = tree1, tree2 = trees2)

# Build data.frame.
DF4 <- data.frame(value = c(RF1, RF2), name =
                    c(rep("original", length(RF1)), rep("inferred", length(RF2))))
DF5 <- data.frame(value = c(RF1, RF1.c), name =
                    c(rep("original", length(RF1)), rep("crossed", length(RF1.c))))
DF6 <- data.frame(value = c(RF2, RF2.c), name =
                    c(rep("original", length(RF2)), rep("crossed", length(RF2.c))))
```

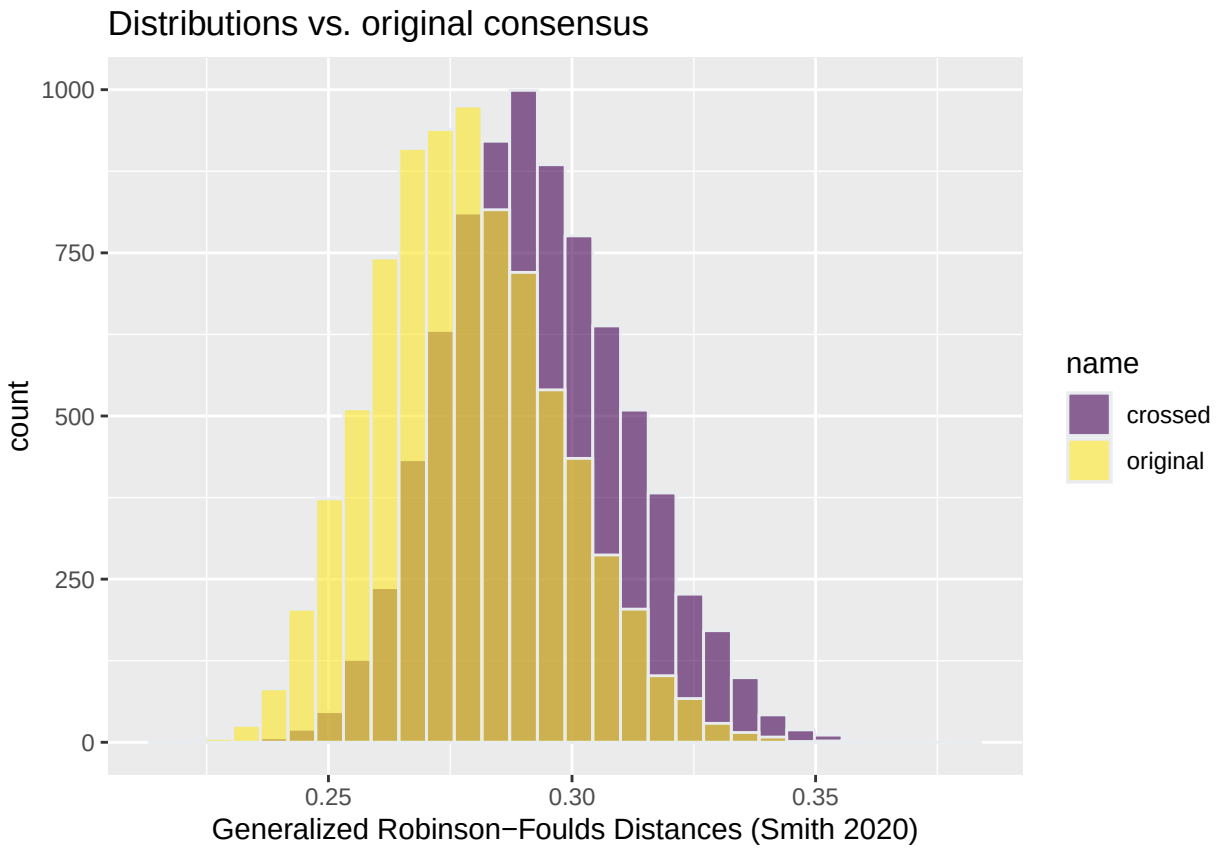
Let's plot the tree distributions compared with their own consensuses.

```
# Plot: non-crossed.
ggplot(data = DF4, aes(x = value, fill = name)) +
  geom_histogram(color = "#e9ecef", alpha = 0.6, position = 'identity') +
  scale_fill_viridis(discrete = TRUE, alpha = 0.6) +
  ggtitle("Distributions vs. own consensus") +
  xlab("Generalized Robinson-Foulds Distances (Smith 2020)")
```



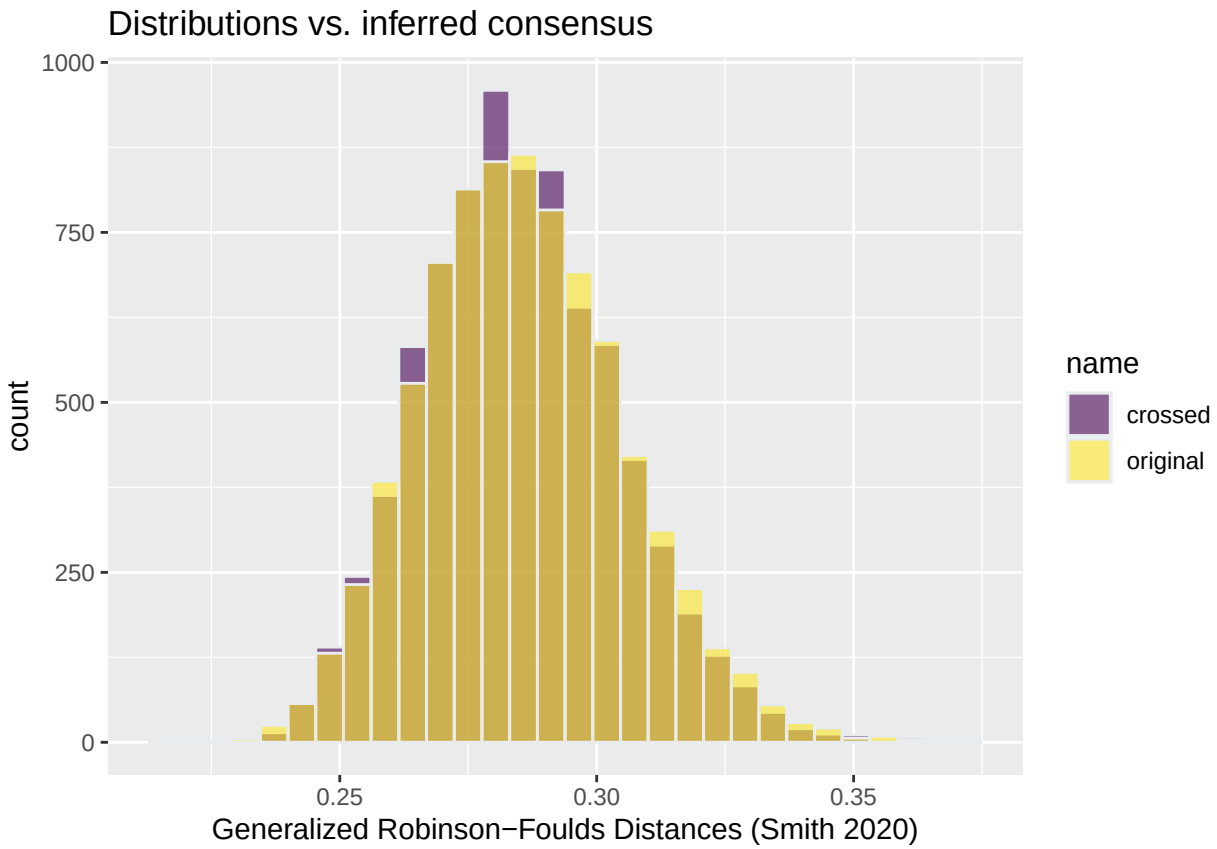
Let's plot the tree distributions compared with the consensus from the 'original' matrix.

```
# Plot: crossed 1.
ggplot(data = DF5, aes(x = value, fill = name)) +
  geom_histogram(color = "#e9ecef", alpha = 0.6, position = 'identity') +
  scale_fill_viridis(discrete = TRUE, alpha = 0.6) +
  ggtitle("Distributions vs. original consensus") +
  xlab("Generalized Robinson-Foulds Distances (Smith 2020)")
```



Let's plot the tree distributions compared with the consensus from the 'inferred' matrix.

```
# Plot: crossed 2.
ggplot(data = DF6, aes(x = value, fill = name)) +
  geom_histogram(color = "#e9ecef", alpha = 0.6, position = 'identity') +
  scale_fill_viridis(discrete = TRUE, alpha = 0.6) +
  ggtitle("Distributions vs. inferred consensus") +
  xlab("Generalized Robinson-Foulds Distances (Smith 2020)")
```


Distributions vs. inferred consensus
1000
750
500
250
0
count
0.25
0.30
0.35
Generalized Robinson−Foulds Distances (Smith 2020)
name
crossed
original