6210 Assignment 2

Nick Werry (with collaboration from Jason Moggridge)

21/10/2020

Setup

```
library(knitr)
opts_chunk$set(tidy = TRUE)
```

Introduction

Cytochrome Oxidase Subunit I (COI) and Sex-Determining Region Y (SRY) are both genes common to mammalian organisms. This study seeks to compare how genetic variation differs between these two genes within the Bos genus. COI is located on the mitochondrial genome, and in Bos taurus has a length of 1545 nucleotides (Genbank: MF925711.1). It's protein product is a fundamental aspect of cytochrome C oxidase, which is involved in cellular energy production. This sequence is commonly used as barcoding region, as it has conserved sites to permit reliable primer binding, but also tolerates the accumulation of genetic variation over time which allows it to be used to predict evolutionary clusters (Pentinsaari et al. 2016). The SRY gene is located on the Y chromosome, and is 1800 nucleotides long in Bos taurus (Genbank: U15569.1). Expression of SRY is responsible for initiating a cascading pathway of male development, giving it a critical role in sex determination. It is believed to act as a transcription factor, modifying expression of other genes to promote male development. Mutations that disrupt SRY function are not well tolerated, as they can result in improper sexual development which makes them unlikely to be passed on to future generations. A consequence of this is that SRY tends to see less sequence divergence than less essential regions (Pamilo and Waugh O'Neill 1997, Cheng et al. 2001). Due to the differences in evolutionary constraints upon these two genes, it is hypothesized that COI will have increased genetic variability, and group into more unique clusters than SRY. It is expected that the strong selective pressure for a functional SRY gene will reduce the substitution frequency as fewer mutations will be passed on through viable offspring, causing the sequence to be more similar across the Bos genus than COI, where substitutions are better tolerated.

Analysis

Load Libraries

Launch all packages needed for analysis.

```
#Load in relevant libraries
library(Biostrings)
library(tidyverse)
library(rentrez)
library(muscle)
```

```
library(DECIPHER)
library(ape)
library(dendextend)
library(ggplot2)
library(ggdendro)
```

Variables

To allow this codeset to be easily adapted for other genes and conditions, commonly adjusted variables are included within this section.

```
#Define genes
# gene1 <- 'COI'
# gene2 <- "SRY"
gene1 <- params$gene1
max1 <- params$upperlen1
min1 <- params$lowerlen1

gene2 <- params$gene2
max2 <- params$upperlen2
min2 <- params$lowerlen2

#Adjust search conditions for each gene
# searchTermGene1 <- sprintf("Bos[ORGN] AND %s[GENE]) AND 550:695[SLEN] NOT (genome[TITL]", gene1)
searchTermGene1 <- sprintf("Bos[ORGN] AND %s[GENE]) AND %s:%s[SLEN] NOT (genome[TITL])", gene1, min1, ma
searchTermGene2 <- sprintf("Bos[ORGN] AND %s[GENE]) AND %s:%s[SLEN] NOT (genome[TITL])", gene2, min2, ma
searchTermGene1
```

```
## [1] "Bos[ORGN] AND COI[GENE]) AND 550:695[SLEN] NOT (genome[TITL])"
```

```
searchTermGene2
```

```
## [1] "Bos[ORGN] AND SRY[GENE]) AND 1300:1500[SLEN] NOT (genome[TITL])"
```

```
##Parameters
#Tolerate X% "N"s in sequence
# missing.data <- 0.05
missing.data <- params$missing

#Limit length of sequence to within X nucleotides of median
# length.var <- 50
length.var <- params$lenvar

#Define model of molecular evolution
# chosen.model <- "TN93"
chosen.model <- params$distmodel

#Define clustering method
# clustering.method <- "single"
clustering.method <- params$clustermethod

#Accept X% sequence variability within a cluster
```

```
# clustering.threshold <- 0.05
clustering.threshold <- params$clusterthreshold

print('selected parameters for analysis:')
```

```
## [1] "selected parameters for analysis:"
```

```
params
```

```
## $gene1
## [1] "COI"
##
## $lowerlen1
## [1] 550
##
## $upperlen1
## [1] 695
##
## $gene2
## [1] "SRY"
##
## $lowerlen2
## [1] 1300
##
## $upperlen2
## [1] 1500
##
## $lenvar
## [1] 50
##
## $missing
## [1] 0.05
##
## $distmodel
## [1] "TN93"
##
## $clustermethod
## [1] "single"
##
## $clusterthreshold
## [1] 0.05
```

The TN93 model of molecular evolution was selected as it incorporates different likelihoods for transversions and transitions, even distinguishing between C/T and A/G transitions. This model accepts unbalanced nucleotide abundance, which is important for this analysis as a mitochondrial gene (high CG content) is being compared with a Y chromosome gene (Tamura and Nei, 1993). Single linkage clustering was performed as it permits more disparate group membership, allowing minor sequence variations within species to still be assigned to the same cluster. A within cluster sequence variability threshold of 0.05 was set. Intra-species variability in COI typically ranges from 2-3%, while intra-genus variability is 10-20%, so threshold of 5% should allow for sequences to cluster by species within the same genus.

Load and Tidy Data

Collect data from NCBI and process it to be suitable for analysis.

```r
#Import Data from NCBI
gene1_search <- entrez_search(db = "nuccore", term = searchTermGene1, retmax = 1000)
gene2_search <- entrez_search(db = "nuccore", term = searchTermGene2, retmax = 1000)

#Prepare data for analysis
gene1_fetch <- entrez_fetch(db = "nuccore", id = gene1_search$ids, rettype = "fasta")
Sys.sleep(1)
gene2_fetch <- entrez_fetch(db = "nuccore", id = gene2_search$ids, rettype = "fasta")

#Write to fasta file
write(gene1_fetch, "gene1_fetch.fasta", sep = "\n")
write(gene2_fetch, "gene2_fetch.fasta", sep = "\n")

#Read FASTA as DNA string (Note: can directly load FASTA files here if collected by other means)
stringSet1 <- readDNAStringSet("gene1_fetch.fasta")
stringSet2 <- readDNAStringSet("gene2_fetch.fasta")

#Format DNA string as dataframe
gene1_df <- data.frame(Name = names(stringSet1), Sequence = paste(stringSet1))
gene2_df <- data.frame(Name = names(stringSet2), Sequence = paste(stringSet2))

#Extract species names, add to new column
gene1_df$Species <- str_extract(gene1_df$Name, "([Bb]os\\s)\\w+")
gene2_df$Species <- str_extract(gene2_df$Name, "([Bb]os\\s)\\w+")

#Filtering to remove "Bos sp," undetermined species
gene1_df <- filter(gene1_df, Species != "Bos sp")
gene2_df <- filter(gene2_df, Species != "Bos sp")

#Extract Sequence IDs, add to new column
gene1_df$ID <- str_extract(gene1_df$Name, "\\w+.[0-9]")
gene2_df$ID <- str_extract(gene2_df$Name, "\\w+.[0-9]")

#Reorder columns to more logical arrangement, remove cluttered Name column
gene1_df <- gene1_df[, c("Species", "ID", "Sequence")]
gene2_df <- gene2_df[, c("Species", "ID", "Sequence")]
```

```r
#Filter to remove missing or low quality sequences
# JM added a column called seqlen to clean up code
gene1_df <- gene1_df %>%
  filter(!is.na(Sequence)) %>%
  mutate(FilteredSequence = str_remove_all(Sequence, "^N+|N+$|-"),
         seqlen = length(FilteredSequence)) %>%
  filter(str_count(FilteredSequence, "N") <= (missing.data*seqlen)) %>%
  filter(seqlen >= median(seqlen) - length.var && seqlen <= median(seqlen) + length.var) %>%
  select(-Sequence)


gene2_df <- gene2_df %>%
  filter(!is.na(Sequence)) %>%
  mutate(FilteredSequence = str_remove_all(Sequence, "^N+|N+$|-"),
         seqlen = length(FilteredSequence)) %>%
  filter(str_count(FilteredSequence, "N") <= (missing.data*seqlen)) %>%
```

```
  filter(seqlen >= median(seqlen) - length.var &&
           seqlen <= median(seqlen) + length.var) %>%
  select(-Sequence)

#Add name identifiers to sequences
names(gene1_df$FilteredSequence) <- gene1_df$ID
names(gene2_df$FilteredSequence) <- gene2_df$ID
```

Checks

Ensure data processing has been executed as intended.

```
#Verify data structure is as expected. Ensure variables set above are logical.

#Should be data.frame and DNAStringSet
sprintf("The %s data was converted to %s, and the filtered sequence is of the class %s.", gene1, class(
```

```
## [1] "The COI data was converted to data.frame, and the filtered sequence is of the class character."
```

```
sprintf("The %s data was converted to %s, and the filtered sequence is of the class %s.", gene2, class(
```

```
## [1] "The SRY data was converted to data.frame, and the filtered sequence is of the class character."
```

```
#See species contributing sequence, should be Bos species only
unique(gene1_df$Species)
```

```
## [1] "Bos taurus"    "Bos grunniens" "Bos indicus"   "Bos frontalis"
## [5] "Bos gaurus"
```

```
unique(gene2_df$Species)
```

```
## [1] "Bos taurus"    "Bos indicus"   "Bos frontalis"
```

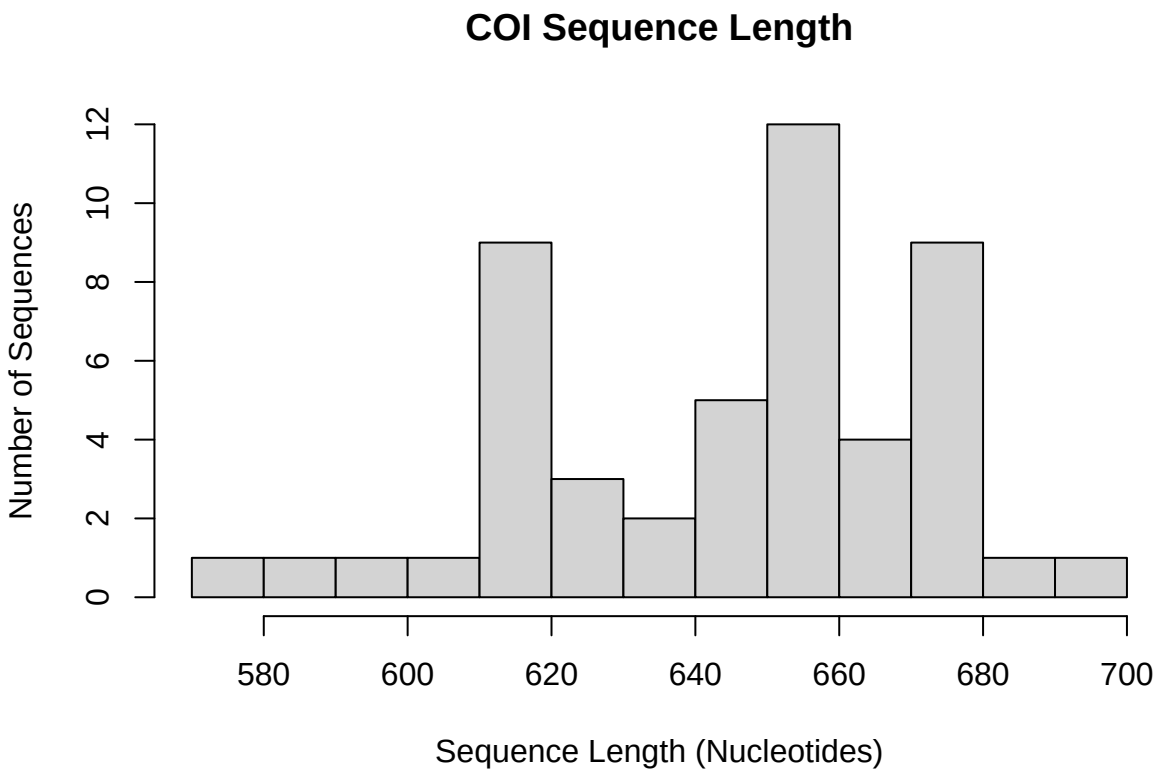
```
#Look for presence of obvious outliers
sprintf("%s data structure:", gene1)
```

```
## [1] "COI data structure:"
```

```
summary(nchar(gene1_df$FilteredSequence))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   573.0   621.0   653.5   644.9   667.0   693.0
```

```
hist(x = nchar(gene1_df$FilteredSequence),
     main = sprintf("%s Sequence Length", gene1),
     xlab = "Sequence Length (Nucleotides)",
     ylab = "Number of Sequences", breaks=10)
```



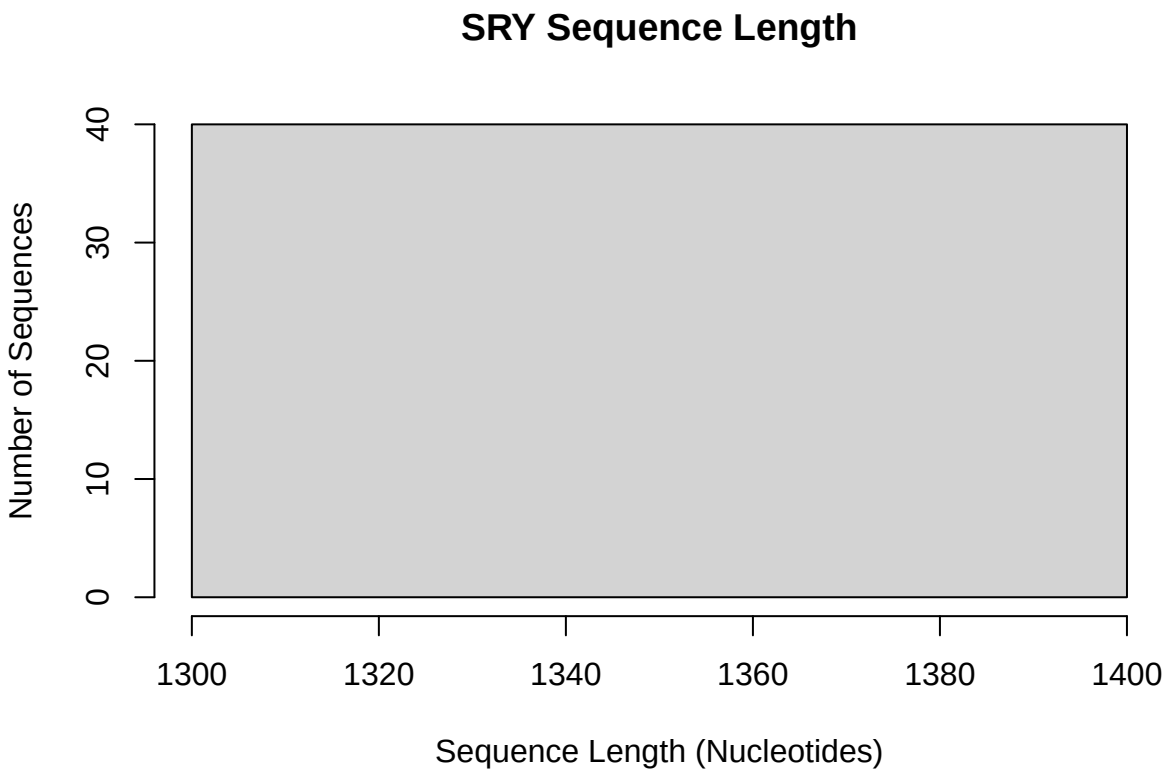

```
sprintf("%s data structure:", gene2)
```

```
## [1] "SRY data structure:"
```

```
summary(nchar(gene2_df$FilteredSequence))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1305    1305    1305    1305    1305    1305
```

```
hist(nchar(as.character(gene2_df$FilteredSequence)),
     main = sprintf("%s Sequence Length", gene2),
     xlab = "Sequence Length (Nucleotides)",
     ylab = "Number of Sequences",
     breaks = 5)
```



Alignment

Align all sequences for Gene1 and Gene2, separately.

```
#Convert Sequence to DNAStringSet format
gene1_df$FilteredSequence <- DNAStringSet(gene1_df$FilteredSequence)
gene2_df$FilteredSequence <- DNAStringSet(gene2_df$FilteredSequence)

#Align sequences
gene1_align <- DNAStringSet(muscle::muscle(gene1_df$FilteredSequence))
gene2_align <- DNAStringSet(muscle::muscle(gene2_df$FilteredSequence))

#Export alignment to named FASTA file
writeXStringSet(gene1_align, file = sprintf("%s_alignment.fas", gene1), format = "fasta")
writeXStringSet(gene2_align, file = sprintf("%s_alignment.fas", gene2), format = "fasta")

#View alignment in browser for QC
BrowseSeqs(gene1_align)
BrowseSeqs(gene2_align)
```

Cluster

Perform clustering to group alignments based on variables selected above.

```
#Convert alignment to DNAbin
gene1_DNAbin <- as.DNAbin(gene1_align)
```

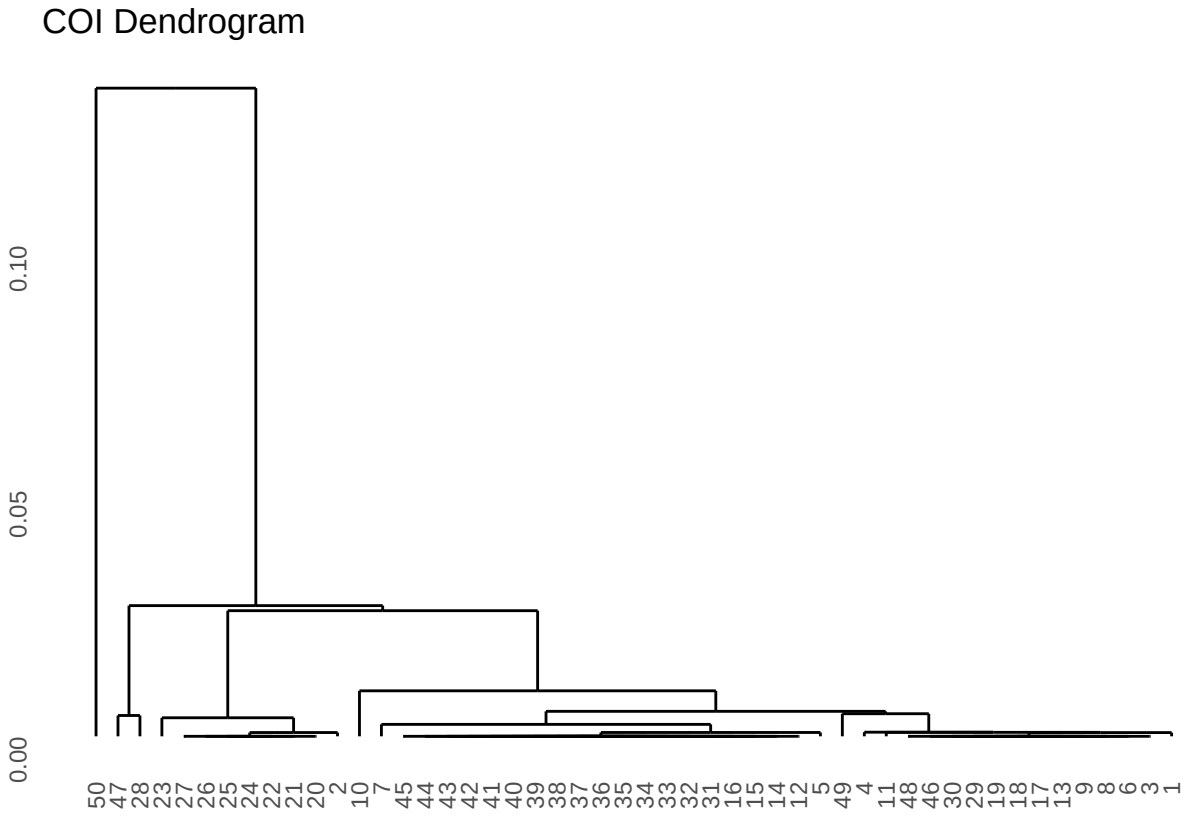
```r
gene2_DNAbin <- as.DNAbin(gene2_align)

#Confirm class is "DNAbin"
sprintf("%s sequences were converted to %s", gene1, class(gene1_DNAbin))
```

```
## [1] "COI sequences were converted to DNAbin"
```

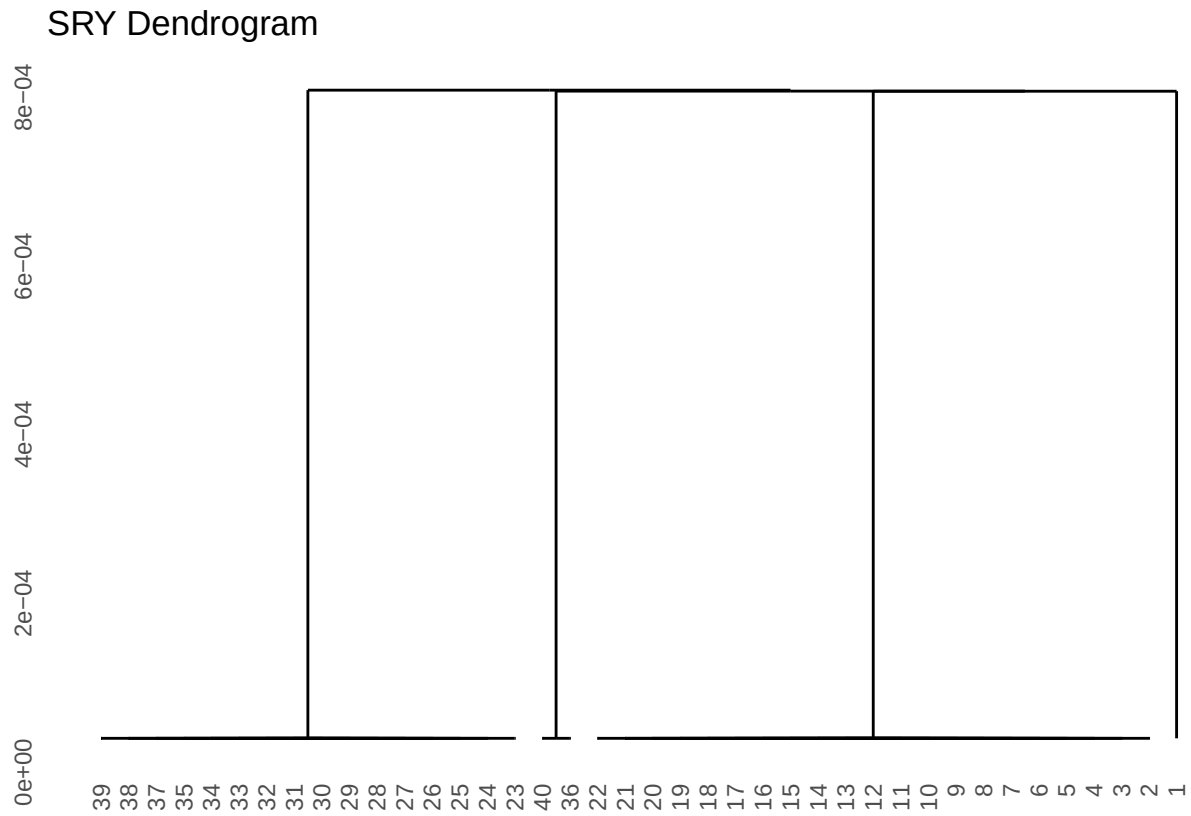
```r
sprintf("%s sequences were converted to %s", gene2, class(gene2_DNAbin))
```

```
## [1] "SRY sequences were converted to DNAbin"
```

```r
#Create distance matrix
gene1_matrix <- dist.dna(gene1_DNAbin, model = chosen.model, as.matrix = TRUE, pairwise.deletion = TRUE)
gene2_matrix <- dist.dna(gene2_DNAbin, model = chosen.model, as.matrix = TRUE, pairwise.deletion = TRUE)

#Cluster and create dendrogram
gene1_clusters <- IdClusters(gene1_matrix, method = clustering.method, cutoff = clustering.threshold, sl
gene2_clusters <- IdClusters(gene2_matrix, method = clustering.method, cutoff = clustering.threshold, sl

#Convert to dend format
gene1_gg <- as.dendrogram(gene1_clusters[[2]])
gene2_gg <- as.dendrogram(gene2_clusters[[2]])

#Plot dendrograms
ggdendrogram(gene1_gg) +
  labs(title = sprintf("%s Dendrogram", gene1), x = "NCBI Accession #", y = "Distance between clusters")
```




```
ggdendrogram(gene2_gg) +
  labs(title = sprintf("%s Dendrogram", gene2), x = "NCBI Accession #", y = "Distance between clusters")
```




```
#Compute the number of clusters identified for each gene
sprintf("%s sequences form %.0f cluster(s).", gene1, length(unique(unlist(gene1_clusters[[1]][1]))))
sprintf("%s sequences form %.0f cluster(s).", gene2, length(unique(unlist(gene2_clusters[[1]][1]))))
```

```
## [1] "COI sequences form 4 cluster(s)."
## [1] "SRY sequences form 1 cluster(s)."
```

View which clusters species were assigned to.

```
#Convert clusters to datafames with cluster and ID columns.
gene1_dfCluster <- as.data.frame(gene1_clusters[1]) %>%
  mutate(ID = row.names(as.data.frame(gene1_clusters[1])))
gene2_dfCluster <- as.data.frame(gene2_clusters[1]) %>%
  mutate(ID = row.names(as.data.frame(gene2_clusters[1])))

#Merge species names and assigned cluster to a single dataframe, filter for unique combinations
gene1_dfSpeciesCluster <- merge(gene1_dfCluster, gene1_df, by = "ID") %>%
  select("Species","cluster") %>%
  group_by(cluster, Species) %>%
  filter(row_number() == 1)
gene2_dfSpeciesCluster <- merge(gene2_dfCluster, gene2_df, by = "ID") %>%
  select("Species","cluster") %>%
  group_by(cluster, Species) %>%
  filter(row_number() == 1)
```

```
#Output species and cluster combinations
sprintf("%s species cluster assignments",gene1)
gene1_dfSpeciesCluster
sprintf("%s species cluster assignments",gene2)
gene2_dfSpeciesCluster
```

```
## [1] "COI species cluster assignments"
## # A tibble: 0 x 2
## # Groups:   cluster, Species [0]
## # ... with 2 variables: Species <chr>, cluster <int>
## [1] "SRY species cluster assignments"
## # A tibble: 0 x 2
## # Groups:   cluster, Species [0]
## # ... with 2 variables: Species <chr>, cluster <int>
```

Results

Clustering analysis of COI and SRY sequence alignments reveals differences in the intra-genus sequence similarity of these genes. In this analysis, COI formed 4 distinct clusters while SRY formed a single cluster, indicating that more variability exists in the COI sequences than SRY. It should be noted that COI data was sourced from 5 species, while SRY data was sourced from 3. Given the strength of the trend observed, however, this is not expected to notably influence the result.

```
#Ratio of species/cluster normalized by number of species
gene1_SC <- (length(unique(gene1_dfSpeciesCluster$Species))/max(gene1_dfSpeciesCluster$cluster))/length
```

```
## Warning in max(gene1_dfSpeciesCluster$cluster): no non-missing arguments to max;
## returning -Inf
```

```
gene2_SC <- (length(unique(gene2_dfSpeciesCluster$Species))/max(gene2_dfSpeciesCluster$cluster))/length
```

```
## Warning in max(gene2_dfSpeciesCluster$cluster): no non-missing arguments to max;
## returning -Inf
```

```
sprintf("%s has a grouping factor of %.2f.", gene1, gene1_SC)
sprintf("%s has a grouping factor of %.2f.", gene2, gene2_SC)
```

```
## [1] "COI has a grouping factor of NaN."
## [1] "SRY has a grouping factor of NaN."
```

To compare how COI and SRY clustering differed, the above "grouping factors" were calculated. These scores take the ratio of species per cluster, then divide by the number of species observed as a means of normalizing the ratio to account for groups with a different number of species. Grouping factor reflects cohesion within the population: a score of 1 indicates that all species belong to the same cluster, while decreasing scores reflects the presence of additional clusters. Given that the grouping factor of COI (0.25) is much lower than that of SRY (1.0), it is apparent that the investigated SRY sequences cluster more closely to each other than the COI sequences do. This result supports the hypothesis that SRY sequences will have less variation than COI.

Discussion

COI is known to tolerate acceptable levels of variation that are used to predict species (Pentinsaari et al. 2016), while SRY is more resistant to nucleotide changes (Cheng et al. 2001). It is logical that this tendency would impact how these sequences cluster, as the difference in selection pressure causes the genes to change at different rates over evolutionary time. Though the TN93 model used in this study compensates for many molecular-level traits, it does not factor in differences in selection pressure for specific genes (Tamura and Nei, 1993). Thus, clustering conditions which distinguish multiple clusters from COI sequence are not sensitive enough to separate SRY sequences from the same genus into more than one cluster, as SRY has not accumulated as many substitutions while the genus diverged. The COI based analysis formed clusters that correlated reasonably closely to species, with a few notable exceptions. Presence of these exceptions justifies why robust barcoding approaches rely on investigation of multiple sites to confirm species, though a single gene can serve as a strong indicator. An interesting note from this investigation is that COI sequences from one species (Bos taurus) were sorted into two clusters: cluster 1 which was exclusive to Bos taurus, and cluster 4 which was shared with Bos indicus. These two species are actually quite close within the Bos genus, and hybrids are capable of producing viable offspring (Hiendler et al 2008), so according the biological species concept they could be considered to be the same species. The two species do exhibit significant morphological differences, however, and are typically maintained as separate populations. Regardless, the overlap in clustering may reflect other underlying genetic similarities between the two. Similarly, Bos frontalis and Bos gaurus were grouped into the same cluster. It is commonly theorized that Bos frontalis is a descendent of Bos gaurus, possibly arising after hybridization with domestic cattle (Kamalakkannan 2020). Given that the domestic species investigated here (Bos taurus and Bos indicus) were not assigned to the same cluster as Bos gaurus and Bos frontalis, this analysis does not support that belief. The trends identified in this report cannot be considered definitive as they are limited due to low sample size. Interesting relationships were identified between the molecular evolution rate of SRY vs COI, and notable phylogenic associations were observed based on COI data, further investigation of additional sequences would be necessary to draw meaningful conclusions.

Literature Resources

Supporting scientific information

Cheng H, Shi H, Zhou R, G Y, Liu L, Liu J, Jiang Y, Kudo T, and Sutou S (2001) Characterization of Bovidae sex determining gene SRY. Genetics Selection Evolution. 33 (6).

Hiendleder S, Lewalski H, and Janke A (2008). Complete mitochondrial genomes of Bos taurus and Bos indicus provide new insights into intraspecies variation, taxonomy, and domestication. Cytogenetic and Genome Research. 120 (1).

Kamalakkannan R Bhavana K, Prabhu VR, Sureshgopi D, Singha HS, and Nagarajan M (2020) The complete mitochondrial genome of Indian gaur, Bos gaurus and its phylogenetic implications. Scientific Reports 10 (11936).

Pamilo P and Waugh O'Neill RJ (1997). Evolution of the Sry Genes. Molecular Biology and Evolution. 14 (1).

Pentinsaari M, Salmela H, Mutanen M, and Roslin T (2016). Molecular evolution of a widely-adopted taxonomic marker (COI) across the animal tree of life. Scientific Reports. 6 (35275).

Tamura K, Nei M (1993). Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. Molecular Biology and Evolution. 10 (3).

Technical Resources

To solve coding problems

Creating strings from variables. Cookbook-r: http://www.cookbook-r.com/Strings/Creating_strings_from_variables/

Introduction to ggdendro. Cran: https://cran.r-project.org/web/packages/ggdendro/vignettes/ggdendro.html

Matching a word after another word in r regex. Stack Overflow: https://stackoverflow.com/questions/34804708/matching-a-word-after-another-word-in-r-regex

Reformat R Source code. Bookdown: https://bookdown.org/yihui/rmarkdown-cookbook/opts-tidy.html

Use sprintf to format floats with no decimal places if integer. Stack Overflow: https://stackoverflow.com/questions/14499579/use-sprintf-to-format-floats-with-no-decimal-places-if-integer