Temporal Resolution

Kyra Bankhead

2023-07-24

In this markdown I will:

1. Divide the data into different resolutions.
2. Calculate null and empirical Whittaker Dissimilarity Index between time period lengths.

Read in Data and Create a Function to Count the Number of Individuals in Disignated Time Resolution

```
# Set working directory here
setwd("C:/Users/bankh/My_Repos/Dolphins/data")

## load all necessary packages
library(vegan)
# Run multiple cores for faster computing
require(doParallel)
require(parallel)
library(sfsmisc, verbose=F)

# Read in file and add months
sample_data <- read.csv("sample_data.csv")

# Get all unique Code values in the entire sample_data
all_codes <- unique(sample_data$Code)

# Create a function that counts the IDs in each element
count_instances <- function(df) {
  code_counts <- table(df$Code)
  code_counts <- code_counts[match(all_codes, names(code_counts))]
  code_counts[is.na(code_counts)] <- 0
  return(code_counts)
}
```

Divide Resolutions from Lowest to Highest Scale

```
# -------------------- 22 sets of 1 year increments----------------------------
# Make a list of only 1 year per dataframe
list_years <- split(sample_data, sample_data$Year)
```

```
# Apply the count_instances function to each year
instances_per_year <- lapply(list_years, count_instances)
# Convert the list of counts to a data frame
p1y <- do.call(rbind, instances_per_year)
# Transforming into binary matrices
p1y <- as.matrix(p1y); p1y[which(p1y>=1)] = 1; p1y[which(p1y<1)] = 0


# -------------------- 11 sets of 2 year increments----------------------------
# Make a list of 2 years per dataframe
sample_data$TwoYearIncrement <- cut(sample_data$Year, breaks = seq(min(sample_data$Year), max(sample_dat
list_twoyears <- split(sample_data, sample_data$TwoYearIncrement)
# Apply the count_instances function to each two years
instances_per_twoyear <- lapply(list_twoyears, count_instances)
# Convert the list of counts to a data frame
p2y <- do.call(rbind, instances_per_twoyear)
# Transforming into binary matrices
p2y <- as.matrix(p2y); p2y[which(p2y>=1)] = 1; p2y[which(p2y<1)] = 0


# -------------------- 7 sets of 3 year increments----------------------------
# Make a list of 3 years per dataframe
sample_data$ThreeYearIncrement <- cut(sample_data$Year, breaks = seq(min(sample_data$Year), max(sample_d
list_threeyears <- split(sample_data, sample_data$ThreeYearIncrement)
# Apply the count_instances function to each two years
instances_per_threeyear <- lapply(list_threeyears, count_instances)
# Convert the list of counts to a data frame
p3y <- do.call(rbind, instances_per_threeyear)
# Transforming into binary matrices
p3y <- as.matrix(p3y); p3y[which(p3y>=1)] = 1; p3y[which(p3y<1)] = 0


# -------------------- 6 sets of 4 year increments----------------------------
# Make a list of 4 years per dataframe
sample_data$FourYearIncrement <- cut(sample_data$Year, breaks = seq(min(sample_data$Year), max(sample_da
list_fouryears <- split(sample_data, sample_data$FourYearIncrement)
# Apply the count_instances function to each two years
instances_per_fouryear <- lapply(list_fouryears, count_instances)
# Convert the list of counts to a data frame
p4y <- do.call(rbind, instances_per_fouryear)
# Transforming into binary matrices
p4y <- as.matrix(p4y); p4y[which(p4y>=1)] = 1; p4y[which(p4y<1)] = 0


# -------------------- 4 sets of 5 year increments----------------------------
# Make a list of 5 years per dataframe
sample_data$FiveYearIncrement <- cut(sample_data$Year, breaks = seq(min(sample_data$Year), max(sample_da
list_fiveyears <- split(sample_data, sample_data$FiveYearIncrement)
# Apply the count_instances function to each two years
instances_per_fiveyear <- lapply(list_fiveyears, count_instances)
# Convert the list of counts to a data frame
p5y <- do.call(rbind, instances_per_fiveyear)
# Transforming into binary matrices
```

```
p5y <- as.matrix(p5y); p5y[which(p5y>=1)] = 1; p5y[which(p5y<1)] = 0


# -------------------- 4 sets of 6 year increments---------------------------
# Make a list of 6 years per dataframe
sample_data$SixYearIncrement <- cut(sample_data$Year, breaks = seq(min(sample_data$Year), max(sample_da
list_sixyears <- split(sample_data, sample_data$SixYearIncrement)
# Apply the count_instances function to each two years
instances_per_sixyear <- lapply(list_sixyears, count_instances)
# Convert the list of counts to a data frame
p6y <- do.call(rbind, instances_per_sixyear)
# Transforming into binary matrices
p6y <- as.matrix(p6y); p6y[which(p6y>=1)] = 1; p6y[which(p6y<1)] = 0


# -------------------- 3 sets of 7 year increments---------------------------
# Make a list of 7 years per dataframe
sample_data$SevenYearIncrement <- cut(sample_data$Year, breaks = seq(min(sample_data$Year), max(sample_
list_sevenyears <- split(sample_data, sample_data$SevenYearIncrement)
# Apply the count_instances function to each two years
instances_per_sevenyear <- lapply(list_sevenyears, count_instances)
# Convert the list of counts to a data frame
p7y <- do.call(rbind, instances_per_sevenyear)
# Transforming into binary matrices
p7y <- as.matrix(p7y); p7y[which(p7y>=1)] = 1; p7y[which(p7y<1)] = 0


# -------------------- 3 sets of 8 year increments---------------------------
# Make a list of 8 years per dataframe
sample_data$EightYearIncrement <- cut(sample_data$Year, breaks = seq(min(sample_data$Year), max(sample_
list_eightyears <- split(sample_data, sample_data$EightYearIncrement)
# Apply the count_instances function to each two years
instances_per_eightyear <- lapply(list_eightyears, count_instances)
# Convert the list of counts to a data frame
p8y <- do.call(rbind, instances_per_eightyear)
# Transforming into binary matrices
p8y <- as.matrix(p8y); p8y[which(p8y>=1)] = 1; p8y[which(p8y<1)] = 0
```

Calculate Null and Emperical Whittaker Dissimilarity Index between Time Period Lengths

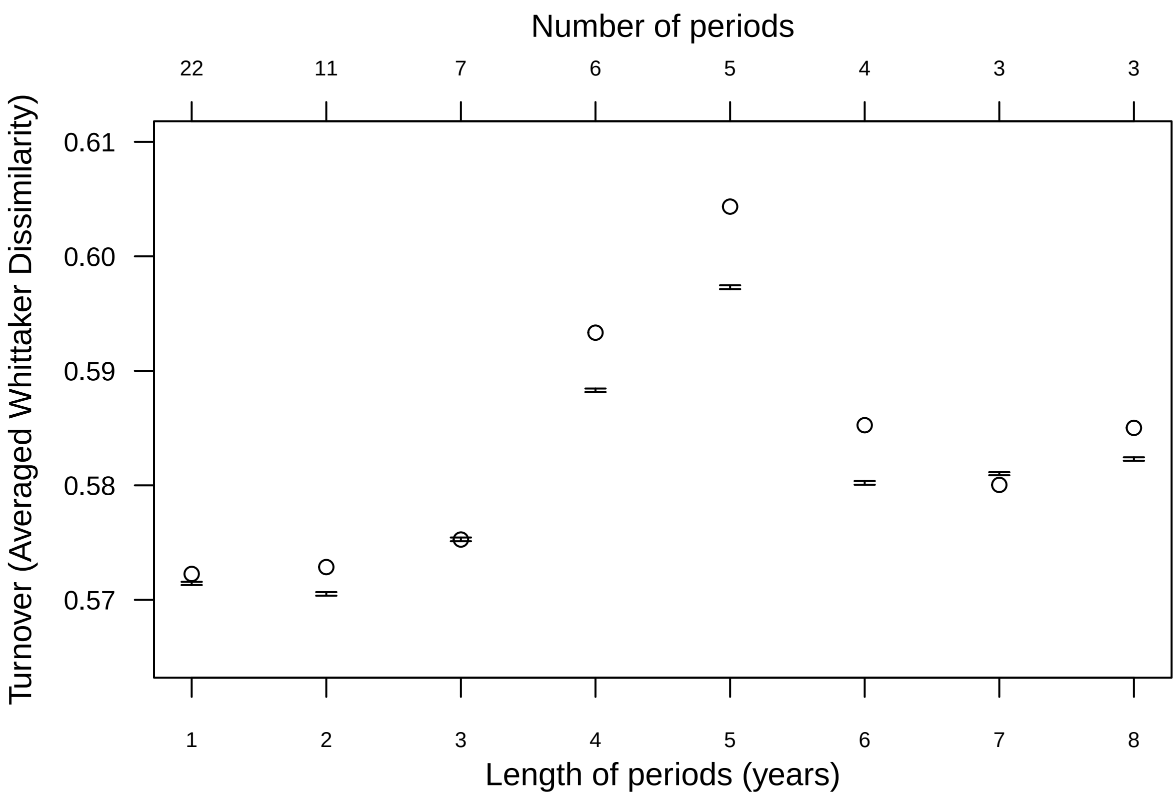
```
source("../code/functions.R") # WDI & WDI permutation

# Turn over results
t1 = turnover_w(data = p1y, iter = 1000, subseq=F, plot=FALSE)
t2 = turnover_w(data = p2y, iter = 1000, subseq=F, plot=FALSE)
t3 = turnover_w(data = p3y, iter = 1000, subseq=F, plot=FALSE)
t4 = turnover_w(data = p4y, iter = 1000, subseq=F, plot=FALSE)
t5 = turnover_w(data = p5y, iter = 1000, subseq=F, plot=FALSE)
t6 = turnover_w(data = p6y, iter = 1000, subseq=F, plot=FALSE)
t7 = turnover_w(data = p7y, iter = 1000, subseq=F, plot=FALSE)
```

```
t8 = turnover_w(data = p8y, iter = 1000, subseq=F, plot=FALSE)

all = rbind(t1, t2, t3, t4, t5, t6, t7, t8)
all = cbind(c(1, 2, 3, 4, 5, 6, 7, 8), all)

par(mar=c(4,5,4,1))
# Plot the final results. Whisker represent 95%CI generated by the null model. X-axis represent the num
errbar(x=c(1, 2, 3, 4, 5, 6, 7, 8), y=all[,2], all[,4], all[,5], ylab="Turnover (Averaged Whittaker Dis
       pch=1, cap=0.02, xaxt='n', xlab="", las=1, cex=1.0, ylim=c(0.565,0.61), xlim=c(1,8), cex.axis=0.
axis(1, at=c(1, 2, 3, 4, 5, 6, 7, 8),las=1, cex.axis=0.7)
mtext(side = 1, "Length of periods (years)", line = 2, font = 1)
axis(3, at=c(1, 2, 3, 4, 5, 6, 7, 8),las=1, labels=c(22, 11, 7, 6, 5, 4, 3, 3), cex.axis=0.7)
mtext(side = 3, "Number of periods", line = 2, font = 1)
```




```
# Print final results
all
```

```
##            Empirical        SD    2.5%CI   97.5%CI
## Turnover 1 0.5722610 0.1412982 0.5712920 0.5715716
## Turnover 2 0.5728612 0.1346759 0.5703612 0.5706761
## Turnover 3 0.5752652 0.1295006 0.5751203 0.5754465
## Turnover 4 0.5933349 0.1256835 0.5881474 0.5884570
## Turnover 5 0.6043450 0.1244430 0.5971372 0.5974714
## Turnover 6 0.5852543 0.1230493 0.5800557 0.5803758
## Turnover 7 0.5800392 0.1086852 0.5808827 0.5811472
```

```
## Turnover 8 0.5850189 0.1171334 0.5821494 0.5824490
```