Social Associations

Kyra Bankhead

2023-03-02

In this markdown I will:

1. Subset data to easily visualize analytically steps. Then, create an association matrix from the gambit of the group assumption using the simple-ratio index function.
2. Form repeated permutations from the true association matrix to create a null distribution of their coefficients of variations (CV). This will determine if the true CV ranges outside of what is expected by chance associations.
3. Calculate the average SRI index within different combinations of HI status pairings.

PART 1: *Social Association Matrix*

Association Matrices will have an annual temporal resolution.

To ensure enough information is available to analyze individual association preference, I have made sure that each association matrix only indludes individuals that were seen at least 10 times in a year.

```
# Set working directory here
setwd("../data")

# Load all necessary packages
require(asnipe) # get_group_by_individual--Damien Farine
require(assocInd) # Could do permutatioNP
require(vegan)
require(doParallel) # Run multiple cores for faster computing
library(ggplot2)
require(foreach)
library(reshape2) # For graphing
library(gridExtra) # To combine plots

##############################################################################
# PART 1: Social Association Matrix ----------------------------------------------

# Read in & combine files
firstgen_data <- read.csv("firstgen_data.csv")
secondgen_data <- read.csv("secondgen_data.csv")
orig_data <- rbind(firstgen_data, secondgen_data)
```

```r
orig_data <- subset(orig_data, subset=c(orig_data$Code != "None"))

# Make date into a date class
orig_data$Date <- as.Date(as.character(orig_data$Date), format="%d-%b-%y")
orig_data$Year <- as.numeric(format(orig_data$Date, format = "%Y"))

# Match ID to sex, age and human data
ILV <- read.csv("Individual_Level_Variables.csv")
# human_data <- read.csv("human_dolphin_data.csv")

## Sex
ID_sex <- setNames(ILV$Sex, ILV$Alias)
orig_data$Sex <- ID_sex[orig_data$Code]
## Age
ID_birth <- setNames(ILV$BirthYear, ILV$Alias)
orig_data$Birth <- ID_birth[orig_data$Code]
orig_data$Age <- as.numeric(as.character(orig_data$Year)) - as.numeric(as.character(orig_data$Birth))

# Get rid of any data with no location data
orig_data <- orig_data[!is.na(orig_data$StartLat) & !is.na(orig_data$StartLon),]
sample_data <- subset(orig_data, subset=c(orig_data$StartLat != 999))

# Now split up data 7 years before and after HAB
sample_data <- sample_data[sample_data$Year >= 1998 & sample_data$Year <= 2011,]

# Get rid of data with no sex or age data
sample_sexage_data <- sample_data[!is.na(sample_data$Sex) & !is.na(sample_data$Age),]

# Make a list of split years per dataframe
split_years <- function (sample_data) {

  ## Sort the data by Year
  sample_data <- sample_data[order(sample_data$Year), ]
  ## Create a column indicating the group (1 or 2) based on the midpoint
  sample_data$Group <- ifelse(sample_data$Year < median(sample_data$Year), 1, 2)
  ## Split the data into two groups based on the 'Group' column
  list_splityears <- split(sample_data, sample_data$Group)

  # Eliminate IDs with less than 5 locations
   updated_list_years <- list()  # Initialize an empty list to store the updated datasets

    for (i in seq_along(list_splityears)) {
      ID <- unique(list_splityears[[i]]$Code)
      obs_vect <- numeric(length(ID))

      for (j in seq_along(ID)) {
        obs_vect[j] <- sum(list_splityears[[i]]$Code == ID[j])
      }

      sub <- data.frame(ID = ID, obs_vect = obs_vect)
      sub <- subset(sub, subset = obs_vect > 10)

      updated_list_years[[i]] <- subset(list_splityears[[i]], Code %in% sub$ID)
```

```
    }
   return(updated_list_years)
}

list_years <- split_years(sample_data)
list_years_sexage <- split_years(sample_sexage_data)

# Make an overlapping dataset
## Get unique codes from both lists
codes_list1 <- unique(list_years[[1]]$Code)
codes_list2 <- unique(list_years[[2]]$Code)
## Find the common codes
common_codes <- intersect(codes_list1, codes_list2)
## Subset the data frames based on the common codes
list_years_ovrlap <- lapply(list_years, function(df) {
  df[df$Code %in% common_codes, ]
})
```

Create gambit of the group index

```
# Calculate Gambit of the group
create_gbi <- function(list_years) {
gbi <- list()
group_data <- list()
for (i in seq_along(list_years)) {

  # Group each individual by date and sighting
  group_data[[i]] <- cbind(list_years[[i]][,c("Date","Sighting","Code","Year")])
  group_data[[i]]$Group <- cumsum(!duplicated(group_data[[i]][1:2])) # Create sequential group # by dat
  group_data[[i]] <- cbind(group_data[[i]][,3:5]) # Subset ID and group #

  # Gambit of the group index
  gbi[[i]] <- get_group_by_individual(group_data[[i]][,c("Code", "Group")], data_format = "individuals"
}
          return(gbi)
                                      }

gbi <- create_gbi(list_years)
gbi_sexage <- create_gbi(list_years_sexage)
gbi_ovrlap <- create_gbi(list_years_ovrlap)
```

Calculate simple-ratio index with group by individual matrix above.

```
# Create association matrix
create_nxn <- function(list_years, gbi) {
source("../code/functions.R") # SRI & null permutation
n.cores <- detectCores()
system.time({
  registerDoParallel(n.cores)
```

```
nxn <- list()
for (i in seq_along(list_years)) {
  nxn[[i]] <- as.matrix(SRI.func(gbi[[i]]))
}
# End parallel processing
stopImplicitCluster()
})
return(nxn)
}

nxn <- create_nxn(list_years, gbi)
nxn_sexage <- create_nxn(list_years_sexage, gbi_sexage)
nxn_ovrlap <- create_nxn(list_years_ovrlap, gbi_ovrlap)
```

PART 2: *Permutations*

Create 1000 repeated matrices from true matrix to form null distribution

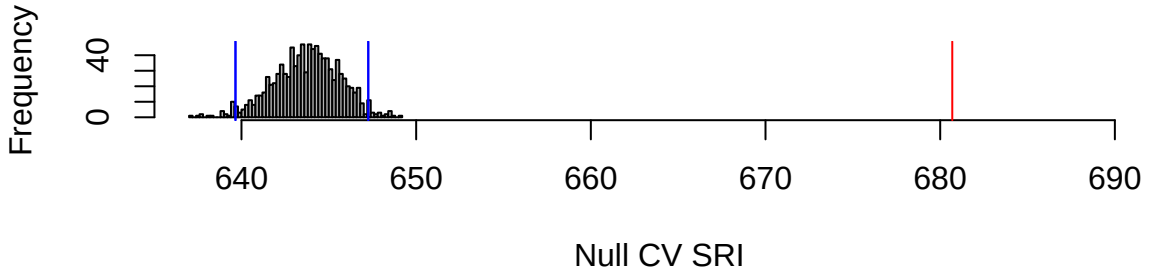
Permute association within samples to test long-term associations and control for gregariousness.

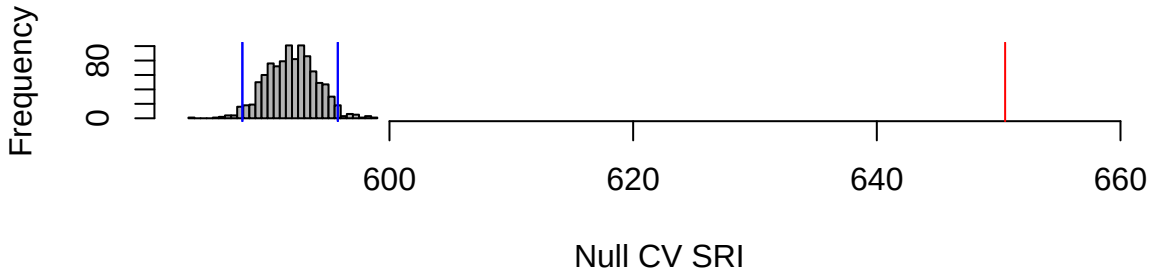
Form CV distribution: True association index coefficient of variation

CV = (SD/mean)*100

```
# Read in null cv values for one year
cv_null <- readRDS("../data/cv_years.RData")
## Remove NAs, if any
# cv_null = cv_null[!is.na(cv_null)]

# Calculate the CV of the observation association data
# CV = (SD/mean)*100
cv_obs <- lapply(nxn, function (df) {(sd(df) / mean(df)) * 100})  # Very high CV = unexpectedly
# high or low association indices in the empirical distribution

# Calculate 95% confidence interval, in a two-tailed test
cv_ci = lapply(cv_null, function (df) {quantile(df, probs=c(0.025, 0.975), type=2)})

# Check whether pattern of connections is non-random
par(mfrow=c(2, 1))

# Create a list to store the histograms
hist_cvs <- list()

# Create histograms for each element in cv_null
for (i in seq_along(cv_null)) {
  hist_cvs[[i]] <- hist(cv_null[[i]],
                        breaks=50,
                        xlim = c(min(cv_null[[i]]), max(cv_obs[[i]] + 10)),
                        col='grey70',
                        main = NULL,
                        xlab="Null CV SRI")
```

```
  # Add lines for empirical CV, 2.5% CI, and 97.5% CI
  abline(v= cv_obs[[i]], col="red")
  abline(v= cv_ci[[i]], col="blue")
  abline(v= cv_ci[[i]], col="blue")
}
```






```
#' This shows whether there are more preferred/avoided
#' relationships than we would expect at random
```

We can reject the null hypothesis that individuals associate at random and conclude that there is evidence that associations are different from what we would expect by chance. Since the CV(TAI) is lower than the other CV, the associations are lower than expected.

PART 3: *SRI Within HI Status Pairs*

Assign HI Status to each individual within the SRI matrix

```
# Read in different behavior's data frames from "GLMM.R"
IDbehav_BG <- readRDS("../data/IDbehav_BG.RData")
IDbehav_SD <- readRDS("../data/IDbehav_SD.RData")
IDbehav_FG <- readRDS("../data/IDbehav_FG.RData")
```

```r
# Get unique behavior assignments
status <- function(IDbehav, HI, NonHI){
  lapply(seq_along(IDbehav), function(i) {
    IDbehav[[i]]$Stat <- ifelse(IDbehav[[i]]$HI > 0, HI, NonHI)
    df <- IDbehav[[i]][, c('Code', 'Stat')]
    df
    })
}

## Match each individual with it's behavior
BG <- status(IDbehav_BG, "BG", "NBG")
SD <- status(IDbehav_SD, "SD", "NSD")
FG <- status(IDbehav_FG, "FG", "NFG")

# Replace individuals in the matrix with their assigned behavior
replace_ID_with_HI <- function(sri_matrix, ID_HI_df) {
  # Create vector that matches IDs to their stat
  id_to_stat <- setNames(ID_HI_df$Stat, ID_HI_df$Code)

  # Replace each ID with stat in row and column names
  row_names <- id_to_stat[rownames(sri_matrix)]
  col_names <- id_to_stat[colnames(sri_matrix)]

  # Create the replaced matrix
  replaced_matrix <- sri_matrix

  # Assign row and column names with behavioral states
  dimnames(replaced_matrix) <- list(row_names, col_names)
  return(replaced_matrix)
}

# Make a replaced nxn for each behavior
BG_nxn <- lapply(seq_along(nxn), function(i) {
  replace_ID_with_HI(nxn[[i]], BG[[i]])
})

SD_nxn <- lapply(seq_along(nxn), function(i) {
  replace_ID_with_HI(nxn[[i]], SD[[i]])
})

FG_nxn <- lapply(seq_along(nxn), function(i) {
  replace_ID_with_HI(nxn[[i]], FG[[i]])
})
```

Get an average SRI for each category of pairing

Step 1: Create a matrix for each category of HI status

```r
## Step 1: Create a matrix for each category of stat

is_NBG <- is_BG <- list()
```

```r
for (i in seq_along(BG_nxn)) {
  is_NBG[[i]] <- rownames(BG_nxn[[i]]) == "NBG"
  is_BG[[i]] <- rownames(BG_nxn[[i]]) == "BG"
}

is_NFG <- is_FG <- list()
for (i in seq_along(FG_nxn)) {
  is_NFG[[i]] <- rownames(FG_nxn[[i]]) == "NFG"
  is_FG[[i]] <- rownames(FG_nxn[[i]]) == "FG"
}

is_NSD <- is_SD <- list()
for (i in seq_along(SD_nxn)) {
  is_NSD[[i]] <- rownames(SD_nxn[[i]]) == "NSD"
  is_SD[[i]] <- rownames(SD_nxn[[i]]) == "SD"
}
```

Step 2: Extract the HI status combinations

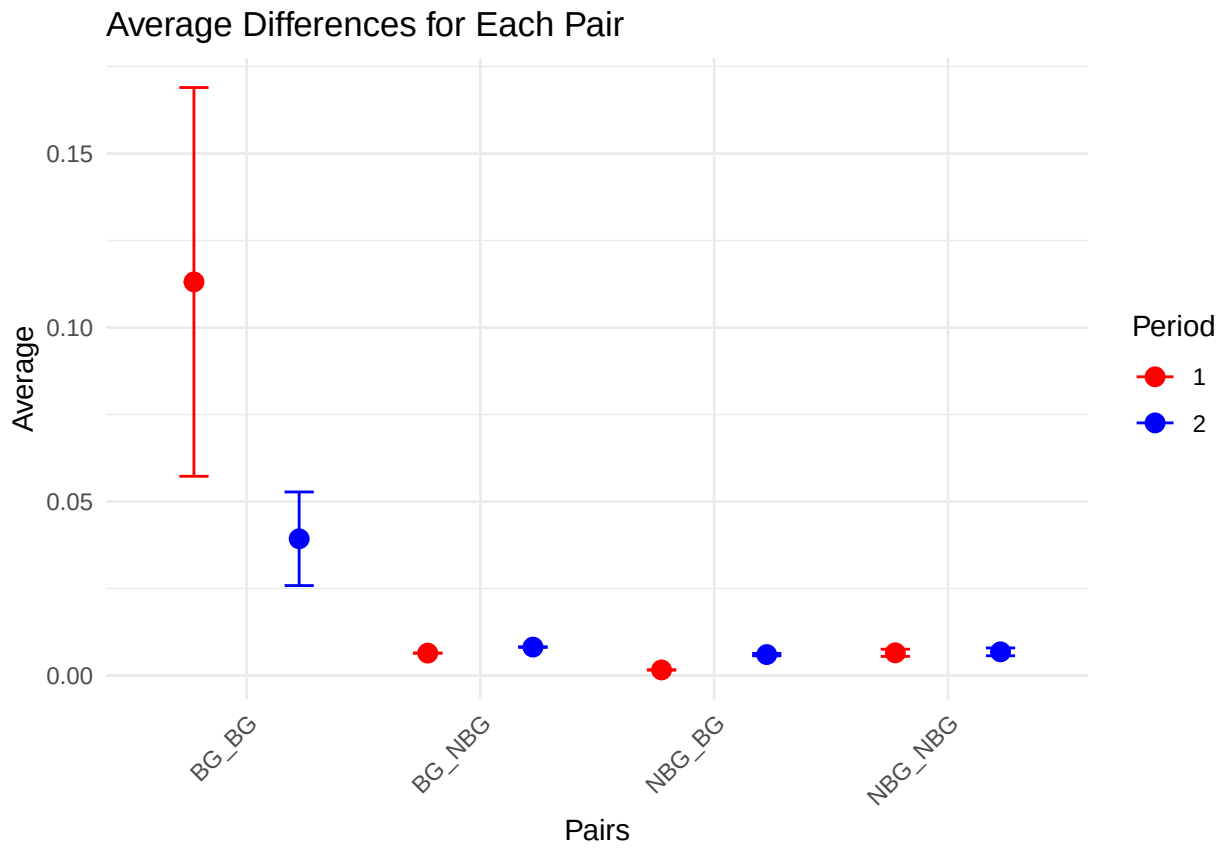
```r
## Step 2: Extract the combinations
### Function to extract combinations
extract_combs <- function(HI_nxn, is_row, is_col) {
  combs <- lapply(seq_along(HI_nxn), function(i) {
    HI_nxn[[i]][is_row[[i]], is_col[[i]]]
  })
  return(combs)
}

#### Apply for each stat comb
NBG_NBG <- extract_combs(BG_nxn, is_NBG, is_NBG)
NBG_BG <- extract_combs(BG_nxn, is_NBG, is_BG)
BG_NBG <- extract_combs(BG_nxn, is_BG, is_NBG)
BG_BG <- extract_combs(BG_nxn, is_BG, is_BG)

NFG_NFG <- extract_combs(FG_nxn, is_NFG, is_NFG)
NFG_FG <- extract_combs(FG_nxn, is_NFG, is_FG)
FG_NFG <- extract_combs(FG_nxn, is_FG, is_NFG)
FG_FG <- extract_combs(FG_nxn, is_FG, is_FG)

NSD_NSD <- extract_combs(SD_nxn, is_NSD, is_NSD)
NSD_SD <- extract_combs(SD_nxn, is_NSD, is_SD)
SD_NSD <- extract_combs(SD_nxn, is_SD, is_NSD)
SD_SD <- extract_combs(SD_nxn, is_SD, is_SD)
```

Step 3: Calculate the average of non-diagonal elements in the pairing sub-matrices

```r
## Step 3: Calculate the average of non-diagonal elements in the pairing sub-matrices

### Function to calculate avg
avg_comb <- function(a, b, c, d) {
```

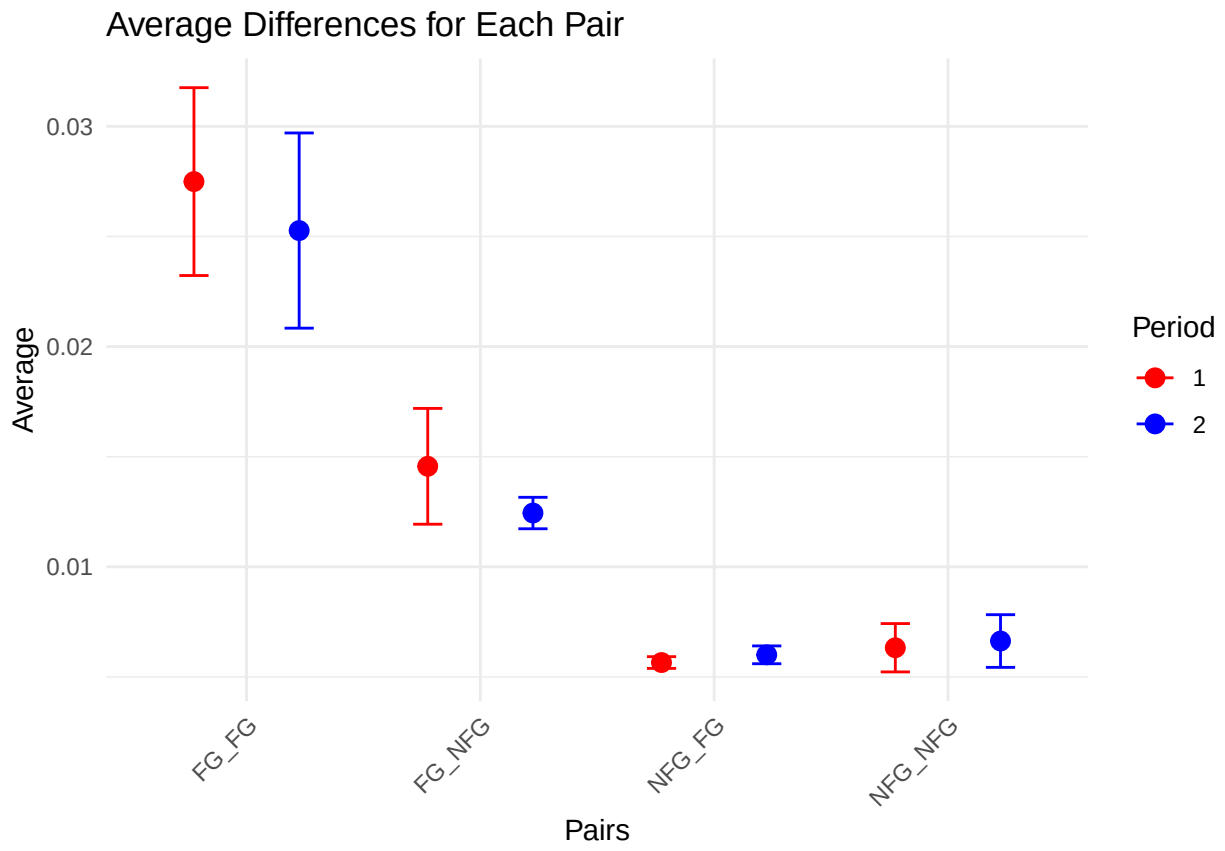
```r
  avg_a <- lapply(seq_along(a), function(i) {
    mean(a[[i]][lower.tri(a[[i]])])
  })
  var_a <- lapply(seq_along(a), function(i) {
    var(a[[i]][lower.tri(a[[i]])])
  })
  name_a <- lapply(a, function(df) {
    paste(unique(rownames(df)), unique(colnames(df)), sep = "_")
  })
  avg_b <- lapply(seq_along(b), function(i) {
    mean(b[[i]][lower.tri(b[[i]])])
  })
  var_b <- lapply(seq_along(b), function(i) {
    var(b[[i]][lower.tri(b[[i]])])
  })
  name_b <- lapply(b, function(df) {
    paste(unique(rownames(df)), unique(colnames(df)), sep = "_")
  })
  avg_c <- lapply(seq_along(c), function(i) {
    mean(c[[i]][lower.tri(c[[i]])])
  })
  var_c <- lapply(seq_along(c), function(i) {
    var(c[[i]][lower.tri(c[[i]])])
  })
  name_c <- lapply(c, function(df) {
    paste(unique(rownames(df)), unique(colnames(df)), sep = "_")
  })
  avg_d <- lapply(seq_along(d), function(i) {
    mean(d[[i]][lower.tri(d[[i]])])
  })
  var_d <- lapply(seq_along(d), function(i) {
    var(d[[i]][lower.tri(d[[i]])])
  })
  name_d <- lapply(d, function(df) {
    paste(unique(rownames(df)), unique(colnames(df)), sep = "_")
  })
  avgvar_df <- data.frame(
    Period = rep(c(1, 2), 4),
    Category = c(name_a[[1]], name_a[[2]],
                 name_b[[1]], name_b[[2]],
                 name_c[[1]], name_c[[2]],
                 name_d[[1]], name_d[[2]]),
    Average = c(unlist(avg_a)[[1]], unlist(avg_a)[[2]],
                unlist(avg_b)[[1]], unlist(avg_b)[[2]],
                unlist(avg_c)[[1]], unlist(avg_c)[[2]],
                unlist(avg_d)[[1]], unlist(avg_d)[[2]]),
    Variation = c(unlist(var_a)[[1]], unlist(var_a)[[2]],
                  unlist(var_b)[[1]], unlist(var_b)[[2]],
                  unlist(var_c)[[1]], unlist(var_c)[[2]],
                  unlist(var_d)[[1]], unlist(var_d)[[2]]))
  return(avgvar_df)
}
```

```r
# Begging
avg_BG <- avg_comb(NBG_NBG, NBG_BG, BG_NBG, BG_BG)

# Plot the differences
# Convert Period to a factor for distinct colors
avg_BG$Period <- factor(avg_BG$Period)

# Define your own colors for each period
period_colors <- c("red", "blue")  # Add more colors if needed

ggplot(avg_BG, aes(x = Category, y = Average, fill = Period, color = Period)) +
  geom_point(position = position_dodge(width = 0.9), size = 3, stat = "identity") +
  geom_errorbar(aes(ymin = Average - Variation, ymax = Average + Variation),
                position = position_dodge(width = 0.9), width = 0.25) +
  labs(x = "Pairs", y = "Average") +
  ggtitle("Average Differences for Each Pair") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_fill_manual(values = period_colors) +
  scale_color_manual(values = period_colors)
```




```r
# Foraging around fixed gear
avg_FG <- avg_comb(NFG_NFG, NFG_FG, FG_NFG, FG_FG)

# Plot the differences
```

```r
# Convert Period to a factor for distinct colors
avg_FG$Period <- factor(avg_FG$Period)

# Define your own colors for each period
period_colors <- c("red", "blue")  # Add more colors if needed

ggplot(avg_FG, aes(x = Category, y = Average, fill = Period, color = Period)) +
  geom_point(position = position_dodge(width = 0.9), size = 3, stat = "identity") +
  geom_errorbar(aes(ymin = Average - Variation, ymax = Average + Variation),
                position = position_dodge(width = 0.9), width = 0.25) +
  labs(x = "Pairs", y = "Average") +
  ggtitle("Average Differences for Each Pair") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_fill_manual(values = period_colors) +
  scale_color_manual(values = period_colors)
```




```r
# Scavenging/Depredation
avg_SD <- avg_comb(NSD_NSD, NSD_SD, SD_NSD, SD_SD)

# Plot the differences
# Convert Period to a factor for distinct colors
avg_SD$Period <- factor(avg_SD$Period)

# Define your own colors for each period
```

```r
period_colors <- c("red", "blue")  # Add more colors if needed

ggplot(avg_SD, aes(x = Category, y = Average, fill = Period, color = Period)) +
  geom_point(position = position_dodge(width = 0.9), size = 3, stat = "identity") +
  geom_errorbar(aes(ymin = Average - Variation, ymax = Average + Variation),
                position = position_dodge(width = 0.9), width = 0.25) +
  labs(x = "Pairs", y = "Average") +
  ggtitle("Average Differences for Each Pair") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_fill_manual(values = period_colors) +
  scale_color_manual(values = period_colors)
```