V1075 Monte Carlo

Matthew Allen

December 2023

Introduction

Here I demonstrate how one might run a Monte Carlo on δ^{18}O-derived temperature proxies in order to estimate uncertainty.

In this example we apply a dual-taxon temperature proxy sensu Fricke and Wing (2007). We'll use δ^{18}O data that I generated from the Albian (mid-Cretaceous) Oklahoma Museum of Natural History V1075 vertebrate microfossil assemblage from the Cloverly Formation of Montana.

You can replace this input data with your data of interest. You'll need δ^{18}O data from an aquatic turtle and a freshwater fish, both from the same assemblage. You will also need the δ^{18}O of the NIST120c standards that were run with your samples. The turtle and fish .csv file will need to be formatted with mean δ^{18}O for each specimen, as well as the eco_type/taxon of each specimen.

Step 1: Load packages and data

!!! I want to source the data directly from GitHub, but Rmarkdown is giving me trouble !!!

```
# Packages
packages <- c("dplyr", "purrr", "ggplot2", "Rcurl")

# install.packages(packages)

library(dplyr)
library(purrr)
library(ggplot2)
# library(Rcurl)

# Data
    # Call d18Op data from GitHub, cleaned and averaged per specimen
(V1075_BySpec.csv)
      # samples_URL <-
getURL("https://raw.githubusercontent.com/mattgeo1990/1075_Vertebrate_d18Op/m
ain/Data/V1075_GarTurtle")
      # V1075_GarTurtle <- read.csv(text = samples_URL)

    # Call NIST120c data from Github, compiled from 2 TC-EA runs
      # standards_URL <-
```

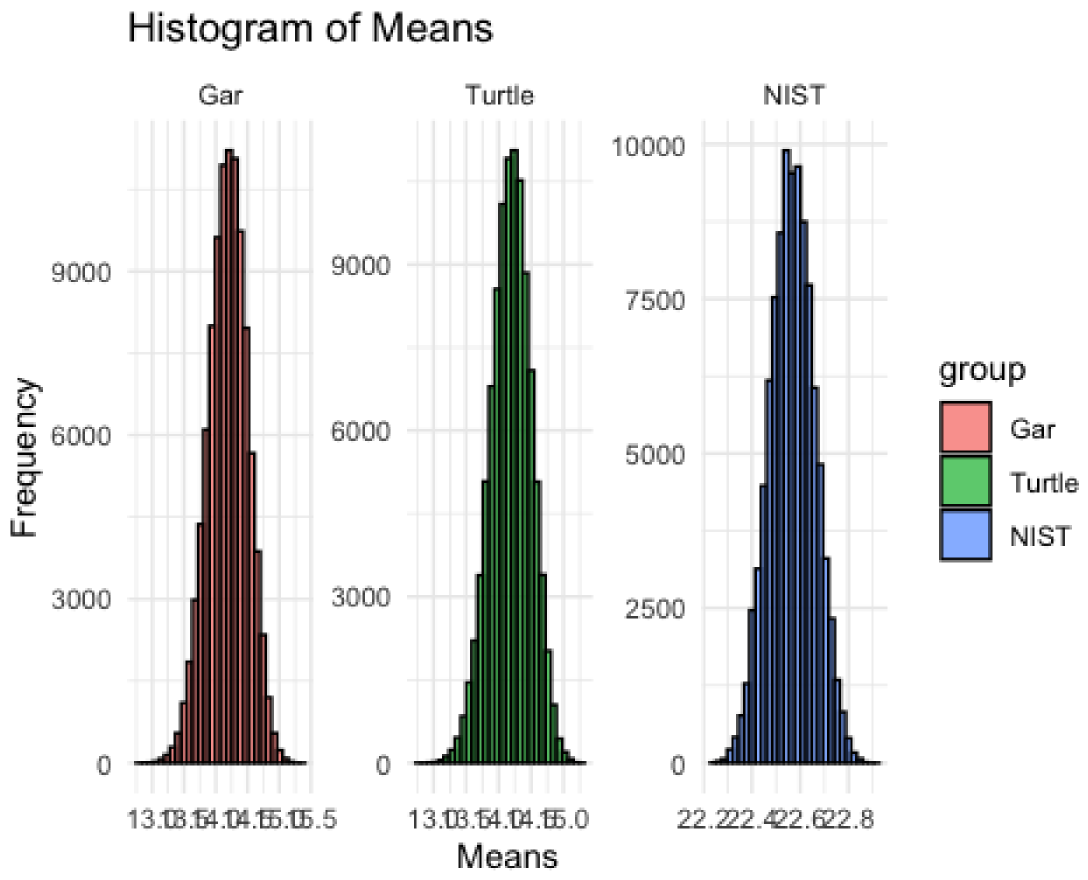
```
"https://raw.githubusercontent.com/mattgeo1990/1075_Vertebrate_d18Op/main/Dat
a/V1075_NIST120c.csv"
        # NIST120c <- read.csv(standards_URL)

#
setwd("/Users/allen/Documents/GitHub/1075_Vertebrate_d18Op/Data")
V1075_GarTurtle <- read.csv("V1075_GarTurtle.csv")
NIST120c <- read.csv("V1075_NIST120c.csv")
```

Step 2: Resampling the isotope data

Here we are resampling with replacement and taking the mean of each resample.

Let's plot a histogram to look at our resample means.

```
# Set number of Monte Carlo repetitions
  nMCrepetitions <- 1e5

# Subset V1075_GarTurtle into gar and turtle matrices
  gar <- subset(V1075_GarTurtle, eco_type = "Fish")
  turtle <- subset(V1075_GarTurtle, eco_type = "Aquatic Turtle")

# resample

  synth_gar <- data_frame(num = 1:nMCrepetitions) %>%
  group_by(num) %>%
  mutate(means = mean(sample(gar$d18O, replace = TRUE)))

  synth_turtle <- data_frame(num = 1:nMCrepetitions) %>%
    group_by(num) %>%
    mutate(means = mean(sample(turtle$d18O, replace = TRUE)))

  synth_NIST <- data_frame(num = 1:nMCrepetitions) %>%
    group_by(num) %>%
    mutate(means = mean(sample(NIST120c$d.18O.16O, replace = TRUE)))

  # Plot histograms of each distribution
    # Combine data frames into one
      combined_data <- rbind(
        data.frame(group = "Gar", values = synth_gar$means),
        data.frame(group = "Turtle", values = synth_turtle$means),
        data.frame(group = "NIST", values = synth_NIST$means)
      )

    # Set order of facets
      combined_data$group <- factor(combined_data$group, levels = c("Gar",
"Turtle", "NIST"))

    # Plotting with ggplot2 using facet_wrap
      ggplot(combined_data, aes(x = values, fill = group)) +
        geom_histogram(position = "identity", alpha = 0.7, bins = 30, color =
"black") +
        labs(title = "Histogram of Means",
             x = "Means",
             y = "Frequency") +
        theme_minimal() +
        facet_wrap(~group, scales = "free")
```

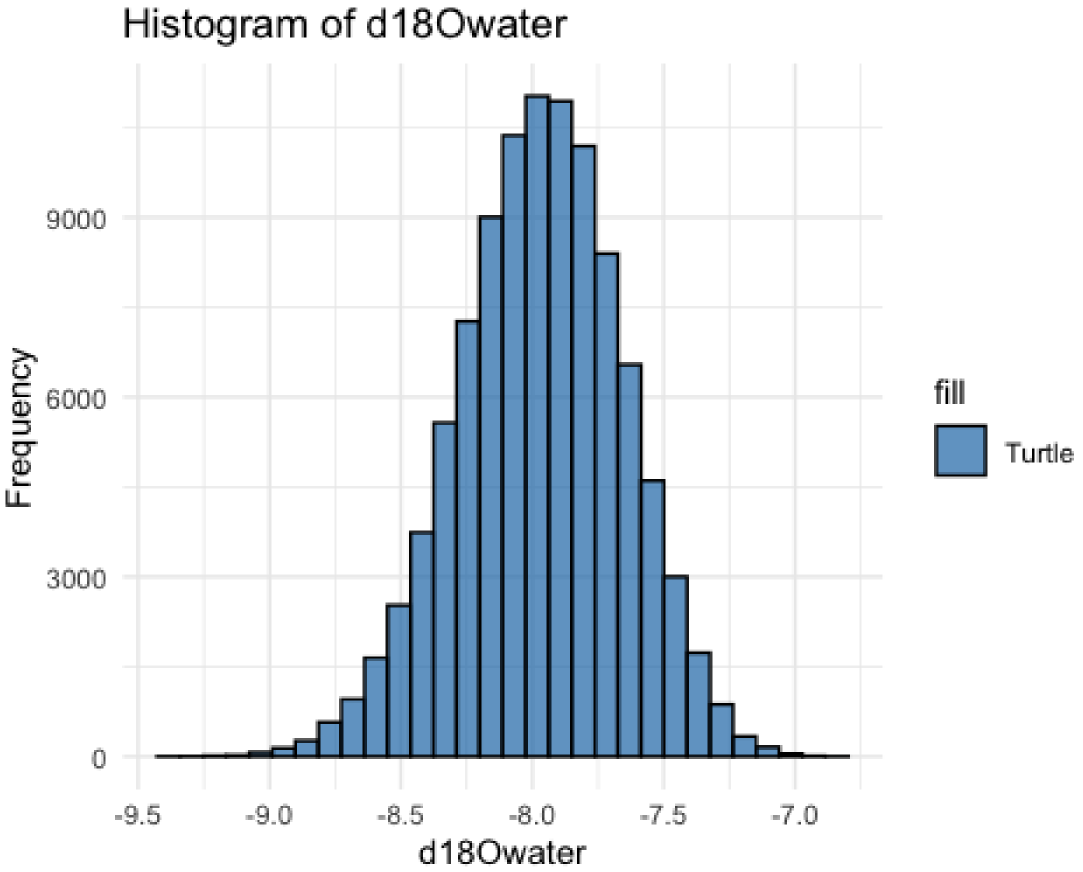


Step 3: Calculate water oxygen isotope values from simulated turtle samples

Note that we are only using data from an aquatic turtle genus Glyptops spp. It is important to choose a turtle taxon that is thought to have been mostly or entirely aquatic in habit.

We'll do this using the relationship between environmental water and turtle bone defined by Barrick et al. (1999).

```
# calculate water

  # create function
    turtlewater <- function(synthmeans_turtle){
      1.01 *(synthmeans_turtle) - 22.3 #Barrick et al. (1999)
    }
  # run turtlewater on synth_turtle
    synth_turtle <- synth_turtle %>%
      mutate(d18Owater = turtlewater(means))

# Plot histogram of d18Owater
  ggplot(synth_turtle, aes(x = d18Owater, fill = "Turtle")) +
    geom_histogram(position = "identity", alpha = 0.7, bins = 30, color =
"black") +
    labs(title = "Histogram of d18Owater",
         x = "d18Owater",
         y = "Frequency") +
    theme_minimal() +
    scale_fill_manual(values = c("#1f78b4"))
```



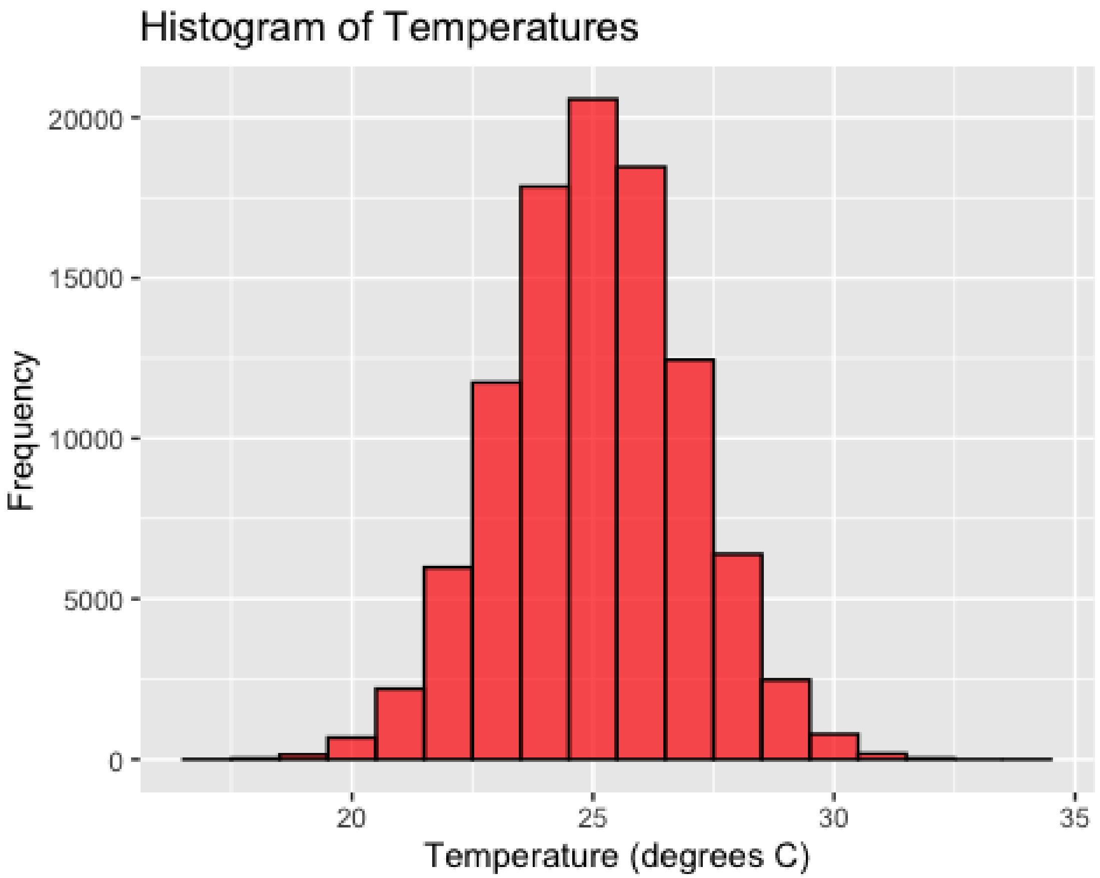
```
  # CI, mean of turtle d18Owater
    CIsetup_water <- sort(synth_turtle$d18Owater)

    # Calculate the lower and upper percentiles for the middle 95%
    lower_percentile_water <- quantile(CIsetup_water, 0.025)
    upper_percentile_water <- quantile(CIsetup_water, 0.975)

    # Subset the middle 95% of the data
    subset_CIsetup_water <- CIsetup_water[CIsetup_water >=
lower_percentile_water & CIsetup_water <= upper_percentile_water]

    # Take mean
    mean_turtlewater <- round(mean(subset_CIsetup_water), 1)
    lowCI_water <- round(abs(mean_turtlewater - lower_percentile_water), 1)
    highCI_water <- round(abs(mean_turtlewater - upper_percentile_water), 1)
    cat(mean_turtlewater, "+", highCI_water, "/", "-", lowCI_water)
```

```
## -8 + 0.6 / - 0.6
```

Step 4: Calculate temperatures

We'll use the relationship between $\delta^{18}O_{phosphate}$ and temperature defined originally by Longinelli & Nuti (1973) and refined by Puc'eat et al. (2010)

```
# calculate temps
    # define temp function
    TempFun <- function(d18Ofish, NISTmean, d18Owater) {
      temp <- 118.7 - 4.22*((d18Ofish  +(22.6 - NISTmean)) - d18Owater)
    }

    # run TempFun over all the synthetic means
    synth_temps <- TempFun(d18Ofish = synth_gar$means,
                           NISTmean = synth_NIST$means,
                           d18Owater = synth_turtle$d18Owater)

# Plot the temperatures
  ggplot(data = data.frame(temperature = synth_temps), aes(x = temperature))
+
        geom_histogram(binwidth = 1, fill = c("#ff0000"), color = "black",
alpha = 0.7) +
        labs(title = "Histogram of Temperatures",
             x = "Temperature (degrees C)",
             y = "Frequency")
```



Step 5: Confidence Intervals

Finally, we'll construct confidence intervals around the mean of our temperature estimates.

```
# Construct confidence intervals
    CIsetup <- sort(synth_temps)

# Calculate the lower and upper percentiles for the middle 95%
    lower_percentile <- quantile(CIsetup, 0.025)
    upper_percentile <- quantile(CIsetup, 0.975)

# Subset the middle 95% of the data
    subset_CIsetup <- CIsetup[CIsetup >= lower_percentile & CIsetup <=
upper_percentile]

# Take mean
    mean_temp <- round(mean(subset_CIsetup), 1)
    lowCI <- round(abs(mean_temp - lower_percentile), 1)
    highCI <- round(abs(mean_temp - upper_percentile), 1)
    cat(mean_temp, "+", highCI, "/", "-", lowCI)

## 25 + 3.8 / - 3.7
```