V1075 Monte Carlo

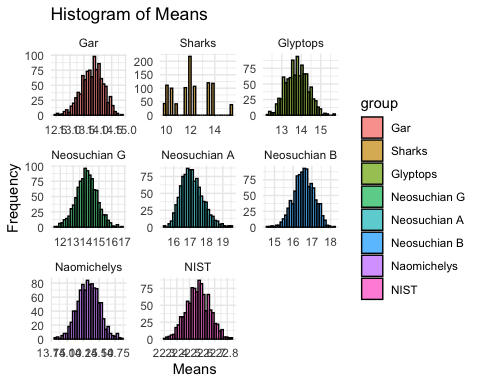
Matthew Allen

December 2023

## Step 1: Load packages and data

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

# Load required packages  
library(dplyr)  
library(purrr)  
library(ggplot2)  
library(RCurl)  
  
# !!! V1075\_MCdata is sourced from script "V1075\_d18O\_wrangle.R"  
  
# local, if needed  
 setwd("/Users/allen/Documents/GitHub/1075\_Vertebrate\_d18Op/Data")  
 V1075\_MCdata <- read.csv("V1075MC\_data.csv")  
 NIST120c <- read.csv("V1075\_NIST120c.csv")  
  
# Set number of Monte Carlo repetitions   
nMCrepetitions <- 1e3  
  
# Subset V1075\_cl by biological group  
gar <- subset(V1075\_MCdata, Taxon == "Lepisosteids")  
shark <- subset(V1075\_MCdata, Taxon == "Hybodonts")  
glyptops <- subset(V1075\_MCdata, Taxon == "Glyptops sp.")  
naomichelys <- subset(V1075\_MCdata, Taxon == "Naomichelys sp.")  
crocG <- subset(V1075\_MCdata, Taxon == "Neosuchian G")  
crocA <- subset(V1075\_MCdata, Taxon == "Neosuchian A")  
crocB <- subset(V1075\_MCdata, Taxon == "Neosuchian B")  
  
# Bootstrapping  
# Resample d18Op for each taxon  
# Take mean of each resample  
# Compile means in data frame  
  
synth\_shark <- data\_frame(num = 1:nMCrepetitions) %>%   
 group\_by(num) %>%   
 mutate(means = mean(sample(shark$d18O, replace = TRUE)))   
  
synth\_gar <- data\_frame(num = 1:nMCrepetitions) %>%   
 group\_by(num) %>%   
 mutate(means = mean(sample(gar$d18O, replace = TRUE)))   
mean(synth\_gar$means)  
  
synth\_glyptops <- data\_frame(num = 1:nMCrepetitions) %>%   
 group\_by(num) %>%   
 mutate(means = mean(sample(glyptops$d18O, replace = TRUE)))   
  
synth\_naomichelys <- data\_frame(num = 1:nMCrepetitions) %>%   
 group\_by(num) %>%   
 mutate(means = mean(sample(naomichelys$d18O, replace = TRUE)))   
  
synth\_crocG <- data\_frame(num = 1:nMCrepetitions) %>%   
 group\_by(num) %>%   
 mutate(means = mean(sample(crocG$d18O, replace = TRUE)))   
  
synth\_crocA <- data\_frame(num = 1:nMCrepetitions) %>%   
 group\_by(num) %>%   
 mutate(means = mean(sample(crocA$d18O, replace = TRUE)))   
  
synth\_crocB <- data\_frame(num = 1:nMCrepetitions) %>%   
 group\_by(num) %>%   
 mutate(means = mean(sample(crocB$d18O, replace = TRUE)))   
  
synth\_NIST <- data\_frame(num = 1:nMCrepetitions) %>%   
 group\_by(num) %>%   
 mutate(means = mean(sample(NIST120c$d.18O.16O, replace = TRUE)))   
  
# Combine data frames into one  
combined\_data <- rbind(  
 data.frame(group = "Gar", values = synth\_gar$means),  
 data.frame(group = "Sharks", values = synth\_shark$means),  
 data.frame(group = "Glyptops", values = synth\_glyptops$means),  
 data.frame(group = "Naomichelys", values = synth\_naomichelys$means),  
 data.frame(group = "Neosuchian G", values = synth\_crocG$means),  
 data.frame(group = "Neosuchian A", values = synth\_crocA$means),  
 data.frame(group = "Neosuchian B", values = synth\_crocB$means),  
 data.frame(group = "NIST", values = synth\_NIST$means)  
)  
  
# Set order of facets  
combined\_data$group <- factor(combined\_data$group, levels = c("Gar", "Sharks", "Glyptops", "Neosuchian G", "Neosuchian A", "Neosuchian B", "Naomichelys", "NIST"))  
  
# Plot histogram of resampled means for each taxon  
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")



# Load the regression models  
  
# Load Barrick regression model (from ~/Documents/GitHub/1075\_Vertebrate\_d18Op/Code/d18Ow\_Proxy\_Regressions.R)  
Barrick\_lm\_model <- readRDS("/Users/allen/Documents/GitHub/1075\_Vertebrate\_d18Op/Data/Barrick\_reg\_lm.rds")  
  
# Load Amiot regression model (from ~/Documents/GitHub/1075\_Vertebrate\_d18Op/Code/d18Ow\_Proxy\_Regressions.R)  
Amiot\_lm\_model <- readRDS("/Users/allen/Documents/GitHub/1075\_Vertebrate\_d18Op/Data/Amiot\_reg\_lm.rds")  
  
# Load Puceat, Longinelli, Nuti (PLN) regression model (from ~/Documents/GitHub/1075\_Vertebrate\_d18Op/Code/d18Ow\_Proxy\_Regressions.R)  
PLN\_lm\_model <- readRDS("/Users/allen/Documents/GitHub/1075\_Vertebrate\_d18Op/Data/PLNd18Op\_reg\_lm.rds")  
  
# Load Tw~Ta transform model (from )  
TwTa\_lm\_model <- readRDS("/Users/allen/Documents/GitHub/1075\_Vertebrate\_d18Op/Data/TwTa\_reg\_lm.rds")  
  
  
summary(Barrick\_lm\_model)  
summary(Amiot\_lm\_model)  
summary(PLN\_lm\_model)

# Croc Water

# Croc Water --------------------------------------------------------------  
  
# Extract the regression coefficients and standard errors  
Amiot\_model\_summary <- summary(Amiot\_lm\_model)  
Amiot\_intercept <- coef(Amiot\_lm\_model)[1]  
Amiot\_slope <- coef(Amiot\_lm\_model)[2]  
Amiot\_intercept\_se <- coef(Amiot\_model\_summary)[1, "Std. Error"] # Standard error for intercept  
Amiot\_slope\_se <- coef(Amiot\_model\_summary)[2, "Std. Error"] # Standard error for slope  
  
cat("Standard Error for Intercept:", Amiot\_intercept\_se, "\n")

## Standard Error for Intercept: 1.076794

cat("Standard Error for Slope:", Amiot\_slope\_se, "\n")

## Standard Error for Slope: 0.06216893

# Simulate regression coefficients for the regression model  
set.seed(123) # For reproducibility  
n\_iterations <- 1e3 # Number of Monte Carlo simulations  
  
# Define   
Amiot\_residual\_sd <- summary(Amiot\_lm\_model)$sigma  
  
# Generate simulated slope and intercept values  
Amiot\_simulated\_slope <- rnorm(n\_iterations, mean = Amiot\_slope, sd = Amiot\_slope\_se)  
Amiot\_simulated\_intercept <- rnorm(n\_iterations, mean = Amiot\_intercept, sd = Amiot\_intercept\_se)  
  
# Monte Carlo simulation for the regression  
# simulate residual error  
Amiot\_residual\_error <- rnorm(n\_iterations, mean = 0, sd = Amiot\_residual\_sd)  
  
# Store Croc G d18Op synth means in vector  
crocG\_synthmeans\_d18Op <- synth\_crocG$means  
mean(crocG\_synthmeans\_d18Op)

## [1] 14.05182

# Expand d18Op values to interact with all simulated parameters  
expanded\_d18Op <- rep(crocG\_synthmeans\_d18Op, each = n\_iterations)  
expanded\_slopes <- rep(Amiot\_simulated\_slope, times = length(crocG\_synthmeans\_d18Op))  
expanded\_intercepts <- rep(Amiot\_simulated\_intercept, times = length(crocG\_synthmeans\_d18Op))  
expanded\_residuals <- rep(Amiot\_residual\_error, times = length(crocG\_synthmeans\_d18Op))  
  
# Calculate water simulations  
crocG\_water\_simulations <- expanded\_slopes \* expanded\_d18Op + expanded\_intercepts + expanded\_residuals  
mean(crocG\_water\_simulations)

## [1] -7.528393

# Sort the simulated water values  
sorted\_crocG\_synthd18Owater <- sort(crocG\_water\_simulations)  
  
# Calculate 95% CI percentiles  
crocwatersynth\_lower\_95\_CI <- quantile(sorted\_crocG\_synthd18Owater, probs = 0.025)  
crocwatersynth\_upper\_95\_CI <- quantile(sorted\_crocG\_synthd18Owater, probs = 0.975)  
  
# Compute the mean of the data  
mean\_crocwater\_synth <- mean(sorted\_crocG\_synthd18Owater)  
  
# Print the results  
cat("Mean crocG d18Owater:", round(mean\_crocwater\_synth, 2), "\n")

## Mean crocG d18Owater: -7.53

cat("95% CI for crocG d18Owater: [", round(crocwatersynth\_lower\_95\_CI, 2), ",", round(crocwatersynth\_upper\_95\_CI, 2), "]\n")

## 95% CI for crocG d18Owater: [ -11.17 , -3.58 ]

library(knitr)  
results <- data.frame(  
 Statistic = c("Mean", "Lower 95% CI", "Upper 95% CI"),  
 Value = c(round(mean\_crocwater\_synth, 2), round(crocwatersynth\_lower\_95\_CI, 2), round(crocwatersynth\_upper\_95\_CI, 2))  
)  
kable(results, caption = "CrocG d18Owater Estimates")

CrocG d18Owater Estimates

|  | Statistic | Value |
| --- | --- | --- |
|  | Mean | -7.53 |
| 2.5% | Lower 95% CI | -11.17 |
| 97.5% | Upper 95% CI | -3.58 |

# Turtle Water

# Extract the regression coefficients and standard errors  
Barrick\_model\_summary <- summary(Barrick\_lm\_model)  
Barrick\_intercept <- coef(Barrick\_lm\_model)[1]  
Barrick\_slope <- coef(Barrick\_lm\_model)[2]  
Barrick\_intercept\_se <- coef(Barrick\_model\_summary)[1, "Std. Error"] # Standard error for intercept  
Barrick\_slope\_se <- coef(Barrick\_model\_summary)[2, "Std. Error"] # Standard error for slope  
  
cat("Standard Error for Intercept:", Barrick\_intercept\_se, "\n")

## Standard Error for Intercept: 0.5848962

cat("Standard Error for Slope:", Barrick\_slope\_se, "\n")

## Standard Error for Slope: 0.03442367

# Simulate regression coefficients for the Barrick regression model  
set.seed(123) # For reproducibility  
n\_iterations <- 1e3 # Number of Monte Carlo simulations  
  
# Define   
Barrick\_residual\_sd <- summary(Barrick\_lm\_model)$sigma  
  
# Generate simulated slope and intercept values  
Barrick\_simulated\_slope <- rnorm(n\_iterations, mean = Barrick\_slope, sd = Barrick\_slope\_se)  
Barrick\_simulated\_intercept <- rnorm(n\_iterations, mean = Barrick\_intercept, sd = Barrick\_intercept\_se)  
  
# Monte Carlo simulation for the regression  
# simulate residual error  
Barrick\_residual\_error <- rnorm(n\_iterations, mean = 0, sd = Barrick\_residual\_sd)  
  
# Store Glyptops d18Op synth means in vector  
glyp\_synthmeans\_d18Op <- synth\_glyptops$means  
mean(glyp\_synthmeans\_d18Op)

## [1] 13.78379

# Expand d18Op values to interact with all simulated parameters  
glyp\_expanded\_d18Op <- rep(glyp\_synthmeans\_d18Op, each = n\_iterations)  
glyp\_expanded\_slopes <- rep(Barrick\_simulated\_slope, times = length(glyp\_synthmeans\_d18Op))  
glyp\_expanded\_intercepts <- rep(Barrick\_simulated\_intercept, times = length(glyp\_synthmeans\_d18Op))  
glyp\_expanded\_residuals <- rep(Barrick\_residual\_error, times = length(glyp\_synthmeans\_d18Op))  
  
# Calculate water simulations  
glyp\_water\_simulations <- glyp\_expanded\_slopes \* glyp\_expanded\_d18Op + glyp\_expanded\_intercepts + glyp\_expanded\_residuals  
mean(glyp\_water\_simulations)

## [1] -8.322535

# Sort the simulated water values  
glyp\_sorted\_synthwater <- sort(glyp\_water\_simulations)  
  
# Calculate 95% CI percentiles  
glypwatersynth\_lower\_95\_CI <- quantile(glyp\_sorted\_synthwater, probs = 0.025)  
glypwatersynth\_upper\_95\_CI <- quantile(glyp\_sorted\_synthwater, probs = 0.975)  
  
# Compute the mean of the middle 95% of the data  
mean\_glyp\_synthwater <- mean(glyp\_sorted\_synthwater)  
  
# Print the results  
cat("Mean Glyptops d18Owater:", round(mean\_glyp\_synthwater, 2), "\n")

## Mean Glyptops d18Owater: -8.32

cat("95% CI for Glyptops d18Owater: [", round(glypwatersynth\_lower\_95\_CI, 2), ",", round(glypwatersynth\_upper\_95\_CI, 2), "]\n")

## 95% CI for Glyptops d18Owater: [ -10.52 , -5.97 ]

# Combine Turtle and Croc Water Estimates

# Define bootstrap iterations  
n\_iterations <- 1e3  
  
# Random sampling and averaging  
bootstrapped\_d18Ow <- replicate(n\_iterations, {  
 sample\_glyp <- sample(glyp\_water\_simulations, size = 1, replace = TRUE)  
 sample\_crocG <- sample(crocG\_water\_simulations, size = 1, replace = TRUE)  
 mean(c(sample\_glyp, sample\_crocG)) # Average of the two  
})  
  
# Analyze combined distribution  
mean\_d18Ow <- mean(bootstrapped\_d18Ow)  
sd\_d18Ow <- sd(bootstrapped\_d18Ow)  
quantiles <- quantile(bootstrapped\_d18Ow, probs = c(0.025, 0.975))  
  
# Print results  
cat("Mean δ¹⁸Ow:", mean\_d18Ow, "\n")

## Mean δ¹⁸Ow: -7.925953

cat("95% CI for δ¹⁸Ow: [", quantiles[1], ",", quantiles[2], "]\n")

## 95% CI for δ¹⁸Ow: [ -10.12694 , -5.524149 ]

# Multi-Taxon Temperature

# Extract the regression coefficients and standard errors  
PLN\_model\_summary <- summary(PLN\_lm\_model)  
PLN\_intercept <- coef(PLN\_lm\_model)[1]  
PLN\_slope <- coef(PLN\_lm\_model)[2]  
PLN\_intercept\_se <- coef(PLN\_model\_summary)[1, "Std. Error"] # Standard error for intercept  
PLN\_slope\_se <- coef(PLN\_model\_summary)[2, "Std. Error"] # Standard error for slope  
  
# Set up the number of Monte Carlo iterations  
set.seed(123)  
n\_iterations <- 1e3  
  
# Define input distributions  
delta\_Op\_distribution <- synth\_gar$means  
delta\_Ow\_distribution <- bootstrapped\_d18Ow  
delta\_ONBS120c\_distribution <- synth\_NIST$means  
  
# Sanity checks (CAN REMOVE LATER)  
mean(delta\_Op\_distribution)

## [1] 13.8007

mean(delta\_Ow\_distribution)

## [1] -7.925953

mean(delta\_ONBS120c\_distribution)

## [1] 22.56711

# Define regression parameters and their uncertainties  
intercept <- PLN\_intercept  
intercept\_sd <- PLN\_intercept\_se  
slope <- PLN\_slope  
slope\_sd <- PLN\_slope\_se  
  
# Simulate regression coefficients  
PLN\_intercept\_simulated <- rnorm(n\_iterations, mean = intercept, sd = intercept\_sd)  
PLN\_slope\_simulated <- rnorm(n\_iterations, mean = slope, sd = slope\_sd)  
  
# Extract residual standard error from temperature regression model  
residual\_sd <- summary(PLN\_lm\_model)$sigma   
  
# Initialize vector to store results  
temperature\_simulations <- numeric(n\_iterations)  
  
# Perform Monte Carlo simulations  
for (i in 1:n\_iterations) {  
 # Sample one value from each input distribution  
 delta\_Op <- sample(delta\_Op\_distribution, 1, replace = TRUE)  
 delta\_Ow <- sample(delta\_Ow\_distribution, 1, replace = TRUE)  
 delta\_ONBS120c <- sample(delta\_ONBS120c\_distribution, 1, replace = TRUE)  
   
 # Simulate regression parameters and residual error  
 intercept\_sim <- rnorm(1, mean = PLN\_intercept, sd = PLN\_intercept\_se)  
 slope\_sim <- rnorm(1, mean = PLN\_slope, sd = PLN\_slope\_se)  
 residual\_error <- rnorm(1, mean = 0, sd = residual\_sd)  
   
 # Calculate temperature for this iteration  
 temperature\_simulations[i] <- intercept\_sim + slope\_sim \* (  
 delta\_Op + (22.6 - delta\_ONBS120c) - delta\_Ow  
 ) + residual\_error  
}  
  
# Summarize the results  
mean\_temperature <- mean(temperature\_simulations)  
lower\_95\_CI <- quantile(temperature\_simulations, probs = 0.025)  
upper\_95\_CI <- quantile(temperature\_simulations, probs = 0.975)  
  
# Display results  
cat("Mean MAWSWT (°C):", mean\_temperature, "\n")

## Mean MAWSWT (°C): 26.29916

cat("95% Confidence Interval (°C): [", lower\_95\_CI, ",", upper\_95\_CI, "]\n")

## 95% Confidence Interval (°C): [ 7.903103 , 43.17998 ]

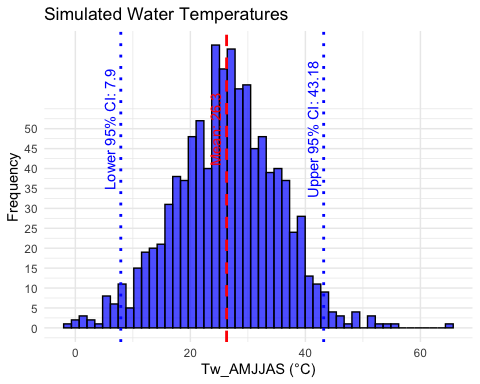
# Summary of temperature simulations  
summary(temperature\_simulations)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -1.366 20.563 26.477 26.299 32.254 65.014

# Standard deviation  
sd\_temp\_sim <- sd(temperature\_simulations)  
  
# Sample size  
n <- length(temperature\_simulations)  
  
# Standard error of the mean  
se\_mean\_temp\_sim <- sd\_temp\_sim / sqrt(n)  
cat("Standard Error of the Mean:", se\_mean\_temp\_sim, "\n")

## Standard Error of the Mean: 0.2851932

# Calculate statistics  
mean\_temp <- mean(temperature\_simulations)  
ci\_lower <- quantile(temperature\_simulations, probs = 0.025)  
ci\_upper <- quantile(temperature\_simulations, probs = 0.975)  
  
# Create the histogram using ggplot2  
library(ggplot2)  
  
ggplot(data.frame(temperature\_simulations), aes(x = temperature\_simulations)) +  
 geom\_histogram(bins = 50, fill = "blue", color = "black", alpha = 0.7) +  
 # Add vertical lines for mean and 95% CI  
 geom\_vline(xintercept = mean\_temp, color = "red", linetype = "dashed", linewidth = 1) +  
 geom\_vline(xintercept = ci\_lower, color = "blue", linetype = "dotted", linewidth = 1) +  
 geom\_vline(xintercept = ci\_upper, color = "blue", linetype = "dotted", linewidth = 1) +  
 # Add labels for the lines  
 annotate("text", x = mean\_temp, y = 50, label = paste("Mean:", round(mean\_temp, 2)),   
 color = "red", angle = 90, vjust = -0.5) +  
 annotate("text", x = ci\_lower, y = 50, label = paste("Lower 95% CI:", round(ci\_lower, 2)),   
 color = "blue", angle = 90, vjust = -0.5) +  
 annotate("text", x = ci\_upper, y = 50, label = paste("Upper 95% CI:", round(ci\_upper, 2)),   
 color = "blue", angle = 90, vjust = -0.5) +  
 # Customize y-axis to have ticks every 5  
 scale\_y\_continuous(breaks = seq(0, 50, by = 5)) +  
 # Add titles and labels  
 labs(title = "Simulated Water Temperatures",  
 x = "Tw\_AMJJAS (°C)",  
 y = "Frequency") +  
 theme\_minimal()



# Water-Air Correction

# Load necessary libraries  
library(ggplot2)  
set.seed(123) # For reproducibility  
  
# Regression coefficients from Tw~Ta model  
# Extract the regression coefficients and standard errors  
TwTa\_model\_summary <- summary(TwTa\_lm\_model)  
TwTa\_intercept <- coef(TwTa\_lm\_model)[1]  
TwTa\_slope <- coef(TwTa\_lm\_model)[2]  
TwTa\_intercept\_se <- coef(TwTa\_model\_summary)[1, "Std. Error"] # Standard error for intercept  
TwTa\_slope\_se <- coef(TwTa\_model\_summary)[2, "Std. Error"] # Standard error for slope  
  
# Extract residual error  
TwTa\_residual\_sd <- summary(TwTa\_lm\_model)$sigma  
  
cat("Standard Error for Intercept:", TwTa\_intercept\_se, "\n")

## Standard Error for Intercept: 2.380387

cat("Standard Error for Slope:", TwTa\_slope\_se, "\n")

## Standard Error for Slope: 0.09243394

# Monte Carlo simulation parameters  
n\_iterations <- 1e3 # Number of Monte Carlo iterations  
  
# Provide distribution of Tw\_AMJJAS values  
Tw\_distribution <- temperature\_simulations  
  
# Monte Carlo simulation incorporating all uncertainties  
Ta\_simulations <- sapply(Tw\_distribution, function(Tw) {  
 TwTa\_simulated\_intercept <- rnorm(n\_iterations, mean = TwTa\_intercept, sd = TwTa\_intercept\_se)  
 TwTa\_simulated\_slope <- rnorm(n\_iterations, mean = TwTa\_slope, sd = TwTa\_slope\_se)  
 TwTa\_residual\_error <- rnorm(n\_iterations, mean = 0, sd = TwTa\_residual\_sd)  
 Tw \* TwTa\_simulated\_slope + TwTa\_simulated\_intercept + TwTa\_residual\_error  
})  
  
# Compute mean and 95% CI for all simulations  
mean\_Ta <- apply(Ta\_simulations, 1, mean)  
lower\_95\_CI <- apply(Ta\_simulations, 1, quantile, probs = 0.025)  
upper\_95\_CI <- apply(Ta\_simulations, 1, quantile, probs = 0.975)  
  
# Overall mean and 95% CI  
overall\_mean\_Ta <- mean(mean\_Ta)  
overall\_lower\_95\_CI <- quantile(unlist(Ta\_simulations), probs = 0.025)  
overall\_upper\_95\_CI <- quantile(unlist(Ta\_simulations), probs = 0.975)  
  
# Display results  
cat("Mean MAWSAT (°C):", overall\_mean\_Ta, "\n")

## Mean MAWSAT (°C): 24.37511

cat("95% Confidence Interval (°C): [", overall\_lower\_95\_CI, ",", overall\_upper\_95\_CI, "]\n")

## 95% Confidence Interval (°C): [ 8.718881 , 40.62746 ]