R Notebook

Prepare the Workspace

```
knitr::opts_chunk$set(warning=FALSE, message=FALSE, error = FALSE) # rmd options
rm(list = ls()); invisible(gc()) # cleaning
```

Control Block

```
# f, f, 0.25 looks nice.
allowmigrants <- F # OPTIONS: T, F
allowsympatry <- F # OPTIONS: T, F
minoverperc <- 0 # remove pairs that do not have thermal overlap (anagenesis)
costvar <- "ele"
```

Packages & Prefs

```
options(scipen = 999) # turn off scientific notation
'%notin%' <- Negate('%in%')
require(ggplot2) # load packages
require(GGally)
require(viridis)
require(caper)
library(dplyr)
library(stringr)
library(maps)
library(ape)
library(EnvStats)
library(forecast)
require(nlme)
require(geodist)
require(letsR)
require(spdep)
require(spatialreg)
require(rnaturalearth)
require(rnaturalearthdata)
require(rgeos)
require(sf)
require(rgdal)
require(raster)
require(lwgeom)
require(EnvStats)
require(psych)
require(phylolm)

world <- ne_coastline(scale = "medium", returnclass = "sf")
load("/Users/boterolab1/Box Sync/CB_VF_Shared/Dry_Lab/Projects/JMPH/PREP/PAM/Data/LonLat_BirdPAM_raster


vlog <- function(x){
```

```
   log( x + abs(min( x , na.rm = T)) + 1)
}

lonlat2UTM = function(lonlat) {
  utm = (floor((lonlat[1] + 180) / 6) %% 60) + 1
  if(lonlat[2] > 0) {
    utm + 32600
  } else{
    utm + 32700
  }
}

# BoxCox function
myBCtransform <- function(myvector) {
  # shift scale to positive numbers and identify optimal lambda for box-cox transformation
  mylambda <- boxcox(as.numeric(myvector)-min(as.numeric(myvector))+1, optimize = T)$lambda

  # transform
  myvector <- scale(boxcoxTransform(as.numeric(myvector)-min(as.numeric(myvector))+1, mylambda))
  return (scale(myvector))
}

setwd("~/Box Sync/CB_VF_Shared/Dry_Lab/Projects/JMPH/Analyze_Processed_Cluster_Outputs/Data")
exclusion <- read.csv(file = "exclusion_nonsimpatric_nonmigrant.csv")
exclusion$realm1red[is.na(exclusion$realm1red)] <- "NA" # NA is north america not R's NA value. fix.
exclusion$realm2green[is.na(exclusion$realm2green)] <- "NA"
```

Load Main Data

```
# main dataframe ---------------------------------
setwd("~/Box Sync/CB_VF_Shared/Dry_Lab/Projects/JMPH/Process_Cluster_Outputs/Data")
load(file =  "Pair_Barrier_Data_FEB2021.rdata")
mydata <- mypairdata; rm(mypairdata)
rownames(mydata) <- mydata$Species.1
# mydatahold <- mydata
```

initial masks

```
# migration ---
if(allowmigrants == F){
  mydata <- mydata[which(mydata$Migration == 1.0),]
}

# patry
if(allowsympatry == F){
  mydata <- mydata[which(mydata[,paste0(costvar, "_c0")] > 0 ),] # doesnt matter if you use ele, mat, v
}
```

sort

```
# mydata$uniquePairId == exclusion$mydata.uniquePairId
# x <- mydata[which(mydata$Species.1 == "Apteryx_owenii"),]
mydata <- mydata[order(match(mydata$uniquePairId,exclusion$mydata.uniquePairId)), ]
# y <- mydata[which(mydata$Species.1 == "Apteryx_owenii"),]
# sum(y!=x, na.rm = T) # checks.
# head(mydata)
```

```
# mydata$uniquePairId == exclusion$mydata.uniquePairId
# rm(x,y)
```

```
# # Basic range maps for all pairs (no paths)
# wdPAM <- "/Users/boterolab1/Box Sync/CB_VF_Shared/Dry_Lab/Projects/JMPH/PREP/PAM/Data"
# setwd(wdPAM); load("cbPAM.rdata")
# setwd("~/Box Sync/CB_VF_Shared/Dry_Lab/Projects/JMPH/Process_Cluster_Outputs/Data")
# pdf("pairmaps2.pdf", width = 19, height = 9.25)
# for (i in 1:nrow(mydata)){
# x <- cbPAM[,c("Longitude(x)","Latitude(y)",mydata$Species.1bl[i])]
# x <- as.data.frame(x[x[,3] == 1,])
# if(ncol(x) == 1) {
#   x <- t(x)
#   colnames(x) <- c("lon", "lat", "pres")
#   x <- as.data.frame(x)
# } else {
#   colnames(x) <- c("lon", "lat", "pres")
# }
#
# y <- cbPAM[,c("Longitude(x)","Latitude(y)",mydata$Species.2bl[i])]
# y <- as.data.frame(y[y[,3] == 1,])
# if(ncol(y) == 1) {
#   y <- t(y)
#   colnames(y) <- c("lon", "lat", "pres")
#   y <- as.data.frame(y)
#
# } else {
#   colnames(y) <- c("lon", "lat", "pres")
# }
# z <- ggplot(world)+
#     geom_sf() +
#     geom_point(data = x, aes(y=lat, x=lon), color = "red") +
#     geom_point(data = y, aes(y=lat, x=lon), color = "green") +
#     theme_bw() +
#     ggtitle(i)
# print(z)
# }
# print(i)
# dev.off()
```

```
mydata_exclusion <- cbind(mydata, exclusion)
save(mydata_exclusion, file = "mydata_exclusion.rdata")
mydata$realm1 <- exclusion$realm1red
mydata$realm2 <- exclusion$realm2green
mydata$realm <- paste0(mydata$realm1, mydata$realm2)
table(mydata$realm)
```

```
##
## AAAA AAIM ATAT ATIM ATPA IMAA IMAT IMIM IMNT NANA NTAA NTNA NTNT OCOC PAAT PANA
##   30    1   45    3    1    3    3   27    1    4    1    1  135    2    3    2
## PAPA
##    6
```

```
mydata$realm[mydata$realm == "AAIM"]; mydata$realm[mydata$realm == "IMAA"] <-  "AAIM"
```

```
## [1] "AAIM"
```

```
mydata$realm[mydata$realm == "ATIM"]; mydata$realm[mydata$realm == "IMAT"] <-  "ATIM"
```

```
## [1] "ATIM" "ATIM" "ATIM"
```

```
mydata$realm[mydata$realm == "ATPA"]; mydata$realm[mydata$realm == "PAAT"] <-  "ATPA"
```

```
## [1] "ATPA"
```

```
mydata$realm[mydata$realm == "IMNT"]; mydata$realm[mydata$realm == "NTIM"] <-  "IMNT"
```

```
## [1] "IMNT"
```

```
mydata$realm[mydata$realm == "NTAA"]; mydata$realm[mydata$realm == "AANT"] <-  "NTAA"
```

```
## [1] "NTAA"
```

```
mydata$realm[mydata$realm == "NTNA"]; mydata$realm[mydata$realm == "NANT"] <-  "NTNA"
```

```
## [1] "NTNA"
```

```
mydata$realm[mydata$realm == "PANA"]; mydata$realm[mydata$realm == "NAPA"] <-  "PANA"
```

```
## [1] "PANA" "PANA"
```

```
mydata$realm <- as.factor(mydata$realm); mydata$realm <- relevel(mydata$realm, "NTNT")

mydata$landgap <- as.logical(exclusion$island)

mydata$cosmopolitan <- as.logical(exclusion$cosmopolitan)

mydata$new.old <- as.logical(exclusion$new.old)

rm(exclusion, mydata_exclusion)
```

Impose masks & do calcuations

```
# filter cosmopolitan and new/old world species (there are relatively few after imposing previous masks
mydata <- mydata[which(mydata$cosmopolitan == FALSE & mydata$new.old == FALSE),]

# dependent variable: elevational barrier size ---
mydata$cost <- mydata[, paste0(costvar, "_c25")]

# data filtering -----------------------
#  thermal overlap ---
mydata$MAT_overlap <- mydata[,paste0("MAT", "_ov_perc_smrnge")]
mydata <- mydata[mydata$MAT_overlap > minoverperc,]

# update sort order --------------------
mydata$sortorder <- seq(1:nrow(mydata))

# longitude ----------------------------
mydata$lon <- mydata[,paste0("lon_mean_pair_", costvar, "_c25")]

# latitude -----------------------------
mydata$lat <- mydata[,paste0("lat_mean_pair_", costvar, "_c25")]
```

```r
# temperature breadth ------------------
mydata$tas_breadth <- mydata$tas_range # mean(mean(sp1 annual tas range -- one value per cell), mean(sp

# mean annual temperature ---------------
mydata$tas_position <- mydata$tas_mean # mean(mean(sp1 annual tas mean  -- one value per cell), mean(sp

# precipitation breadth ----------------
mydata$pcp_breadth <- mydata$pcp_range # mean(mean(sp1 annual pcp range  -- one value per cell), mean(s

# precipitation breadth ----------------
mydata$pcp_position <- mydata$pcp_mean # mean(mean(sp1 annual pcp mean  -- one value per cell), mean(sp

# distance -----------------------------
mydata$distance <- mydata[,paste0("centroid_distance_",costvar,"_c25")]

# mountain mass ------------------------
mtns <- readOGR(dsn="~/Box Sync/CB_VF_Shared/Dry_Lab/Projects/JMPH/Other_Input_Data/GMBA", layer="GMBA
mtns <- st_as_sf(mtns)
buffer_ranges<-c(seq(from = 0, to = 200, by  = 50)[-1], seq(from=0, to = 3000, by = 250)[-1]) * 1000
bufferdata <- matrix(nrow = nrow(mydata), ncol = length(buffer_ranges))
for(i in 1:nrow(mydata)){
  my_centroid <- st_geometry(st_sfc(st_point(c(mydata$lon[i],mydata$lat[i]))))
  st_crs(my_centroid) <- crs(LonLat_BirdPAM_raster)
  EPSG_2_UTM <- as.numeric(lonlat2UTM(my_centroid[[1]]))
  # To see the UTM #st_crs(EPSG_2_UTM)$proj4string
  my_centroid_proj = st_transform(st_as_sf(my_centroid), EPSG_2_UTM)
  for(j in 1:length(buffer_ranges)){
    my_centroid_with_b_km_buffer <- st_buffer(my_centroid_proj,dist=buffer_ranges[j])
    mtns_in_buffer_b_km <- st_intersection(st_make_valid(sf::st_transform(my_centroid_with_b_km_buffer,
    area_in_buffer_b_km <- sum(as.numeric(st_area(mtns_in_buffer_b_km)))
    bufferdata[i, j] <- area_in_buffer_b_km
  }
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
```

```
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 31
## [1] 32
## [1] 33
## [1] 34
## [1] 35
## [1] 36
## [1] 37
## [1] 38
## [1] 39
## [1] 40
## [1] 41
## [1] 42
## [1] 43
## [1] 44
## [1] 45
## [1] 46
## [1] 47
## [1] 48
## [1] 49
## [1] 50
## [1] 51
## [1] 52
## [1] 53
## [1] 54
## [1] 55
## [1] 56
## [1] 57
## [1] 58
## [1] 59
## [1] 60
## [1] 61
## [1] 62
## [1] 63
## [1] 64
## [1] 65
## [1] 66
## [1] 67
## [1] 68
## [1] 69
## [1] 70
## [1] 71
## [1] 72
```

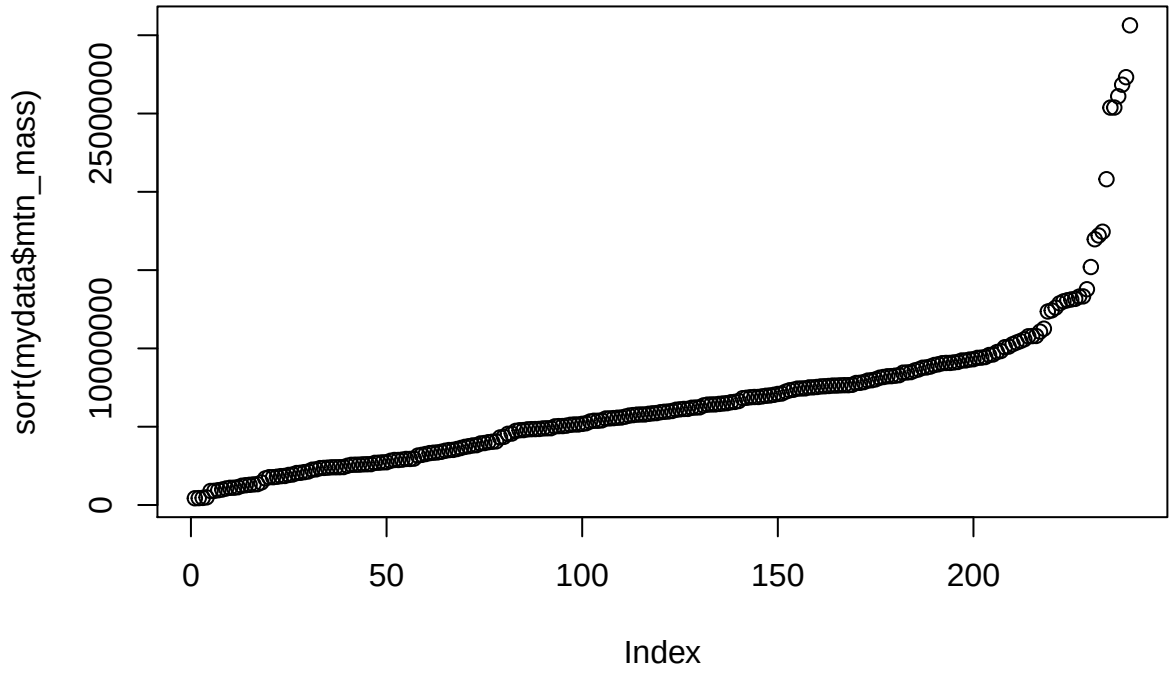
```
## [1] 73
## [1] 74
## [1] 75
## [1] 76
## [1] 77
## [1] 78
## [1] 79
## [1] 80
## [1] 81
## [1] 82
## [1] 83
## [1] 84
## [1] 85
## [1] 86
## [1] 87
## [1] 88
## [1] 89
## [1] 90
## [1] 91
## [1] 92
## [1] 93
## [1] 94
## [1] 95
## [1] 96
## [1] 97
## [1] 98
## [1] 99
## [1] 100
## [1] 101
## [1] 102
## [1] 103
## [1] 104
## [1] 105
## [1] 106
## [1] 107
## [1] 108
## [1] 109
## [1] 110
## [1] 111
## [1] 112
## [1] 113
## [1] 114
## [1] 115
## [1] 116
## [1] 117
## [1] 118
## [1] 119
## [1] 120
## [1] 121
## [1] 122
## [1] 123
## [1] 124
## [1] 125
## [1] 126
```

```
## [1] 127
## [1] 128
## [1] 129
## [1] 130
## [1] 131
## [1] 132
## [1] 133
## [1] 134
## [1] 135
## [1] 136
## [1] 137
## [1] 138
## [1] 139
## [1] 140
## [1] 141
## [1] 142
## [1] 143
## [1] 144
## [1] 145
## [1] 146
## [1] 147
## [1] 148
## [1] 149
## [1] 150
## [1] 151
## [1] 152
## [1] 153
## [1] 154
## [1] 155
## [1] 156
## [1] 157
## [1] 158
## [1] 159
## [1] 160
## [1] 161
## [1] 162
## [1] 163
## [1] 164
## [1] 165
## [1] 166
## [1] 167
## [1] 168
## [1] 169
## [1] 170
## [1] 171
## [1] 172
## [1] 173
## [1] 174
## [1] 175
## [1] 176
## [1] 177
## [1] 178
## [1] 179
## [1] 180
```

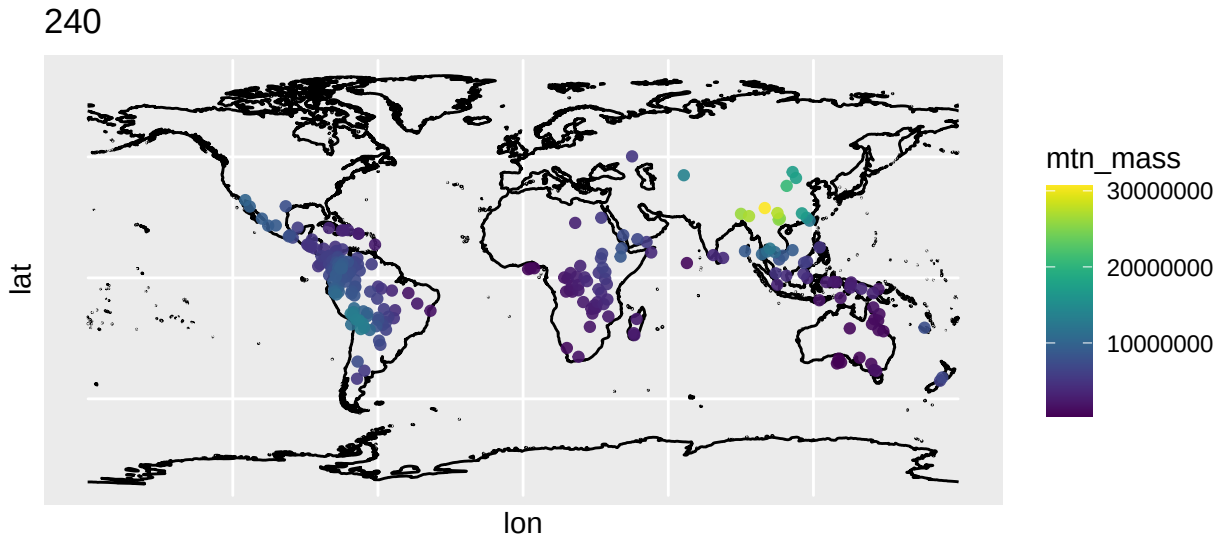
```
## [1] 181
## [1] 182
## [1] 183
## [1] 184
## [1] 185
## [1] 186
## [1] 187
## [1] 188
## [1] 189
## [1] 190
## [1] 191
## [1] 192
## [1] 193
## [1] 194
## [1] 195
## [1] 196
## [1] 197
## [1] 198
## [1] 199
## [1] 200
## [1] 201
## [1] 202
## [1] 203
## [1] 204
## [1] 205
## [1] 206
## [1] 207
## [1] 208
## [1] 209
## [1] 210
## [1] 211
## [1] 212
## [1] 213
## [1] 214
## [1] 215
## [1] 216
## [1] 217
## [1] 218
## [1] 219
## [1] 220
## [1] 221
## [1] 222
## [1] 223
## [1] 224
## [1] 225
## [1] 226
## [1] 227
## [1] 228
## [1] 229
## [1] 230
## [1] 231
## [1] 232
## [1] 233
## [1] 234
```

```
## [1] 235
## [1] 236
## [1] 237
## [1] 238
## [1] 239
## [1] 240
```

```
bufferdatahold <- bufferdata
for(i in 1:nrow(bufferdata)) {
  bufferdata[i,] <- bufferdata[i,] / buffer_ranges
}
mydata$mtn_mass <- rowSums(bufferdata)
plot(sort(mydata$mtn_mass))
```




```
x <- ggplot(world)+
  geom_sf() +
  geom_point(data = mydata[order(mydata[, "mtn_mass"], decreasing = F),], aes(y=lat, x=lon, color = mtn
  ggtitle(i)+scale_color_viridis()
print(x)
```




```
# for troubleshooting and checks.
# plot(mtns$geometry)
# plot(my_centroid, add =T)
# plot(st_make_valid((sf::st_transform(my_centroid_with_b_km_buffer, crs=st_crs(mtns)))), add = T)
# probcoords <- st_geometry(st_sfc(st_point(c(8.80132870725873,18.1810491982364))))
# plot(probcoords, add = T, col = "red")

# water buffering ----------------------
wtr <- read_sf("~/Box Sync/CB_VF_Shared/Dry_Lab/Projects/JMPH/Other_Input_Data/ne_50m_ocean/ne_50m_ocean
buffer_ranges<-c(seq(from = 0, to = 200, by  = 50)[-1], seq(from=0, to = 3000, by = 250)[-1]) * 1000
bufferdataw <- matrix(nrow = nrow(mydata), ncol = length(buffer_ranges))
for(i in 1:nrow(mydata)){
  my_centroid <- st_geometry(st_sfc(st_point(c(mydata$lon[i],mydata$lat[i]))))
  st_crs(my_centroid) <- crs(LonLat_BirdPAM_raster)
  EPSG_2_UTM <- as.numeric(lonlat2UTM(my_centroid[[1]]))
  # To see the UTM #st_crs(EPSG_2_UTM)$proj4string
  my_centroid_proj = st_transform(st_as_sf(my_centroid), EPSG_2_UTM)
  for(j in 1:length(buffer_ranges)){
    my_centroid_with_b_km_buffer <- st_buffer(my_centroid_proj,dist=buffer_ranges[j])
    wtr_in_buffer_b_km <- st_intersection(st_make_valid(sf::st_transform(my_centroid_with_b_km_buffer,
    area_in_buffer_b_km <- sum(as.numeric(st_area(wtr_in_buffer_b_km)))
    bufferdataw[i, j] <- area_in_buffer_b_km
  }
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
```

```
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 31
## [1] 32
## [1] 33
## [1] 34
## [1] 35
## [1] 36
## [1] 37
## [1] 38
## [1] 39
## [1] 40
## [1] 41
## [1] 42
## [1] 43
## [1] 44
## [1] 45
## [1] 46
## [1] 47
## [1] 48
## [1] 49
## [1] 50
## [1] 51
## [1] 52
## [1] 53
## [1] 54
## [1] 55
## [1] 56
## [1] 57
## [1] 58
## [1] 59
## [1] 60
## [1] 61
## [1] 62
## [1] 63
## [1] 64
## [1] 65
```

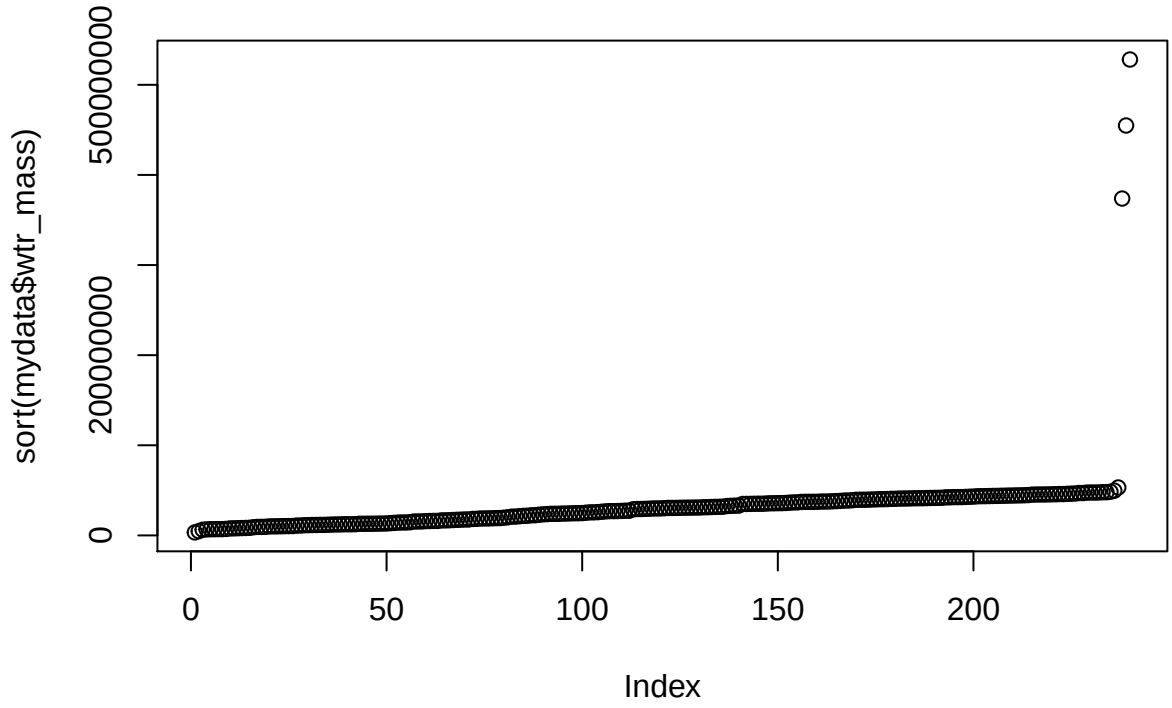
```
## [1] 66
## [1] 67
## [1] 68
## [1] 69
## [1] 70
## [1] 71
## [1] 72
## [1] 73
## [1] 74
## [1] 75
## [1] 76
## [1] 77
## [1] 78
## [1] 79
## [1] 80
## [1] 81
## [1] 82
## [1] 83
## [1] 84
## [1] 85
## [1] 86
## [1] 87
## [1] 88
## [1] 89
## [1] 90
## [1] 91
## [1] 92
## [1] 93
## [1] 94
## [1] 95
## [1] 96
## [1] 97
## [1] 98
## [1] 99
## [1] 100
## [1] 101
## [1] 102
## [1] 103
## [1] 104
## [1] 105
## [1] 106
## [1] 107
## [1] 108
## [1] 109
## [1] 110
## [1] 111
## [1] 112
## [1] 113
## [1] 114
## [1] 115
## [1] 116
## [1] 117
## [1] 118
## [1] 119
```

```
## [1] 120
## [1] 121
## [1] 122
## [1] 123
## [1] 124
## [1] 125
## [1] 126
## [1] 127
## [1] 128
## [1] 129
## [1] 130
## [1] 131
## [1] 132
## [1] 133
## [1] 134
## [1] 135
## [1] 136
## [1] 137
## [1] 138
## [1] 139
## [1] 140
## [1] 141
## [1] 142
## [1] 143
## [1] 144
## [1] 145
## [1] 146
## [1] 147
## [1] 148
## [1] 149
## [1] 150
## [1] 151
## [1] 152
## [1] 153
## [1] 154
## [1] 155
## [1] 156
## [1] 157
## [1] 158
## [1] 159
## [1] 160
## [1] 161
## [1] 162
## [1] 163
## [1] 164
## [1] 165
## [1] 166
## [1] 167
## [1] 168
## [1] 169
## [1] 170
## [1] 171
## [1] 172
## [1] 173
```

```
## [1] 174
## [1] 175
## [1] 176
## [1] 177
## [1] 178
## [1] 179
## [1] 180
## [1] 181
## [1] 182
## [1] 183
## [1] 184
## [1] 185
## [1] 186
## [1] 187
## [1] 188
## [1] 189
## [1] 190
## [1] 191
## [1] 192
## [1] 193
## [1] 194
## [1] 195
## [1] 196
## [1] 197
## [1] 198
## [1] 199
## [1] 200
## [1] 201
## [1] 202
## [1] 203
## [1] 204
## [1] 205
## [1] 206
## [1] 207
## [1] 208
## [1] 209
## [1] 210
## [1] 211
## [1] 212
## [1] 213
## [1] 214
## [1] 215
## [1] 216
## [1] 217
## [1] 218
## [1] 219
## [1] 220
## [1] 221
## [1] 222
## [1] 223
## [1] 224
## [1] 225
## [1] 226
## [1] 227
```

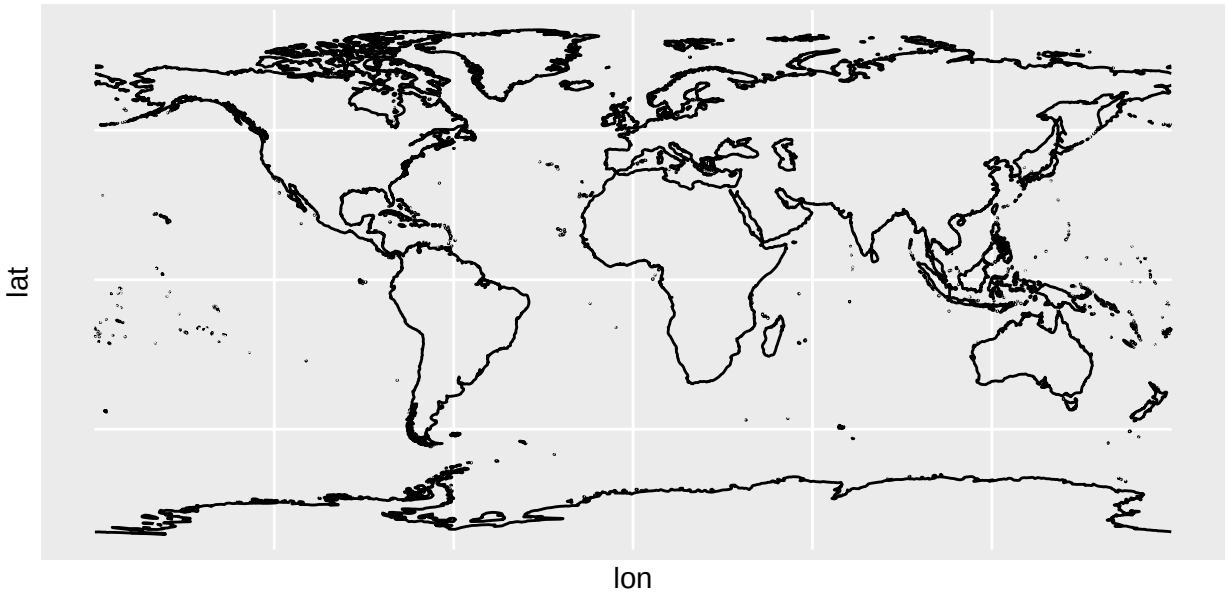
```
## [1] 228
## [1] 229
## [1] 230
## [1] 231
## [1] 232
## [1] 233
## [1] 234
## [1] 235
## [1] 236
## [1] 237
## [1] 238
## [1] 239
## [1] 240
```

```
bufferdatawhold <- bufferdataw
for(i in 1:nrow(bufferdataw)) {
  bufferdataw[i,] <- bufferdataw[i,] / buffer_ranges
}
mydata$wtr_mass <- rowSums(bufferdataw)
plot(sort(mydata$wtr_mass))
```

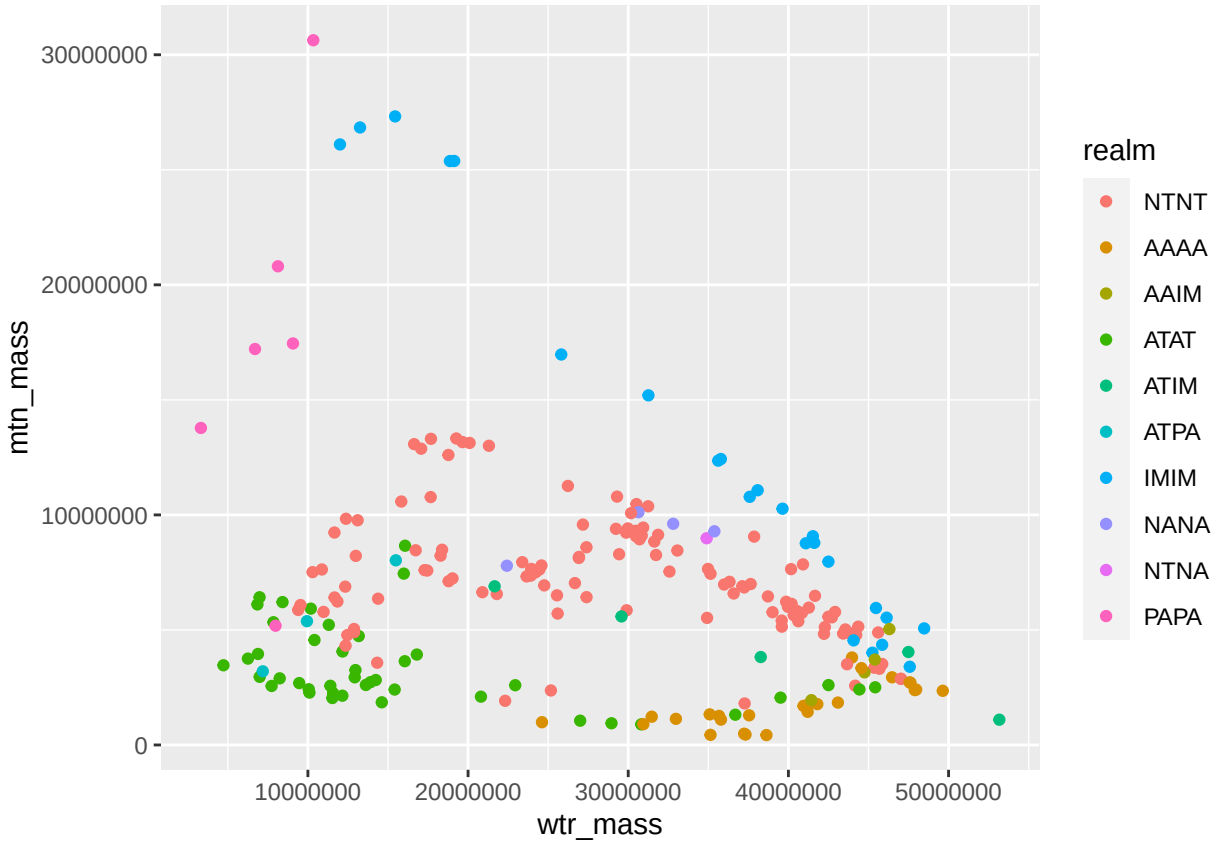



```
plotdata <- mydata[order(mydata[, "wtr_mass"], decreasing = F),]
x <- ggplot(world)+
  geom_sf() +
  geom_point(data = plotdata[which(plotdata$wtr_mass_new < 370000000),], aes(y=lat, x=lon, color = wtr_
  ggtitle(i)+scale_color_viridis()
print(x)
```

240




```
ggplot(mydata[mydata$wtr_mass < 370000000,],aes(x=wtr_mass, y = mtn_mass, color =realm))+
  geom_point()
```



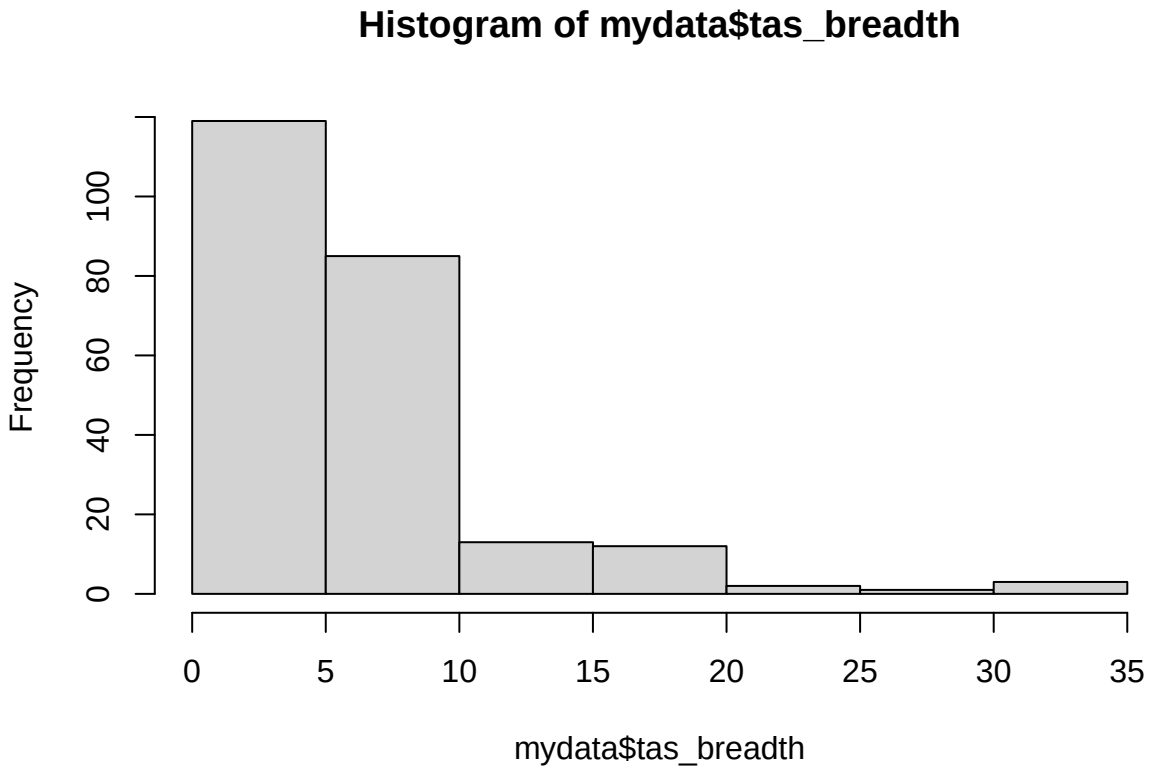

```
# dispersal ability ---------------------
dispab <- read.csv("~/Box Sync/CB_VF_Shared/Dry_Lab/Projects/JMPH/Other_Input_Data/Bird Hand-Wing Index/
mydata$dispersal_ability <- NA
for (i in 1:nrow(mydata)){
```

```
  dispab_sp1 <- dispab$HWI[dispab$IUCN.name == mydata$Species.1bl[i]]
  dispab_sp2 <- dispab$HWI[dispab$IUCN.name == mydata$Species.2bl[i]]
  dispab_pair <- mean(c(dispab_sp1, dispab_sp2), na.rm = T)
  mydata$dispersal_ability[i] <- dispab_pair
  rm(dispab_sp1, dispab_sp2, dispab_pair)
}
mydata <- mydata[!is.nan(mydata$dispersal_ability),]

# pair age -----------------------------
mydata$pair_age <- mydata$Pair.age..MY.

# length of boundary -------------------
mydata$boundary_length <- mydata$boundary_length_ele_c25

# retain cols of interest only.
mydata <- mydata[,c("uniquePairId", "Species.1", "Species.2", "Species.1bl", "Species.2bl", "cost", "la
                    "tas_breadth","tas_position", "pcp_breadth","pcp_position", "mtn_mass", "wtr_mass",
                    "dispersal_ability", "pair_age", "distance", "boundary_length", "MAT_overlap", "rea
                    "landgap")]
gc()
```

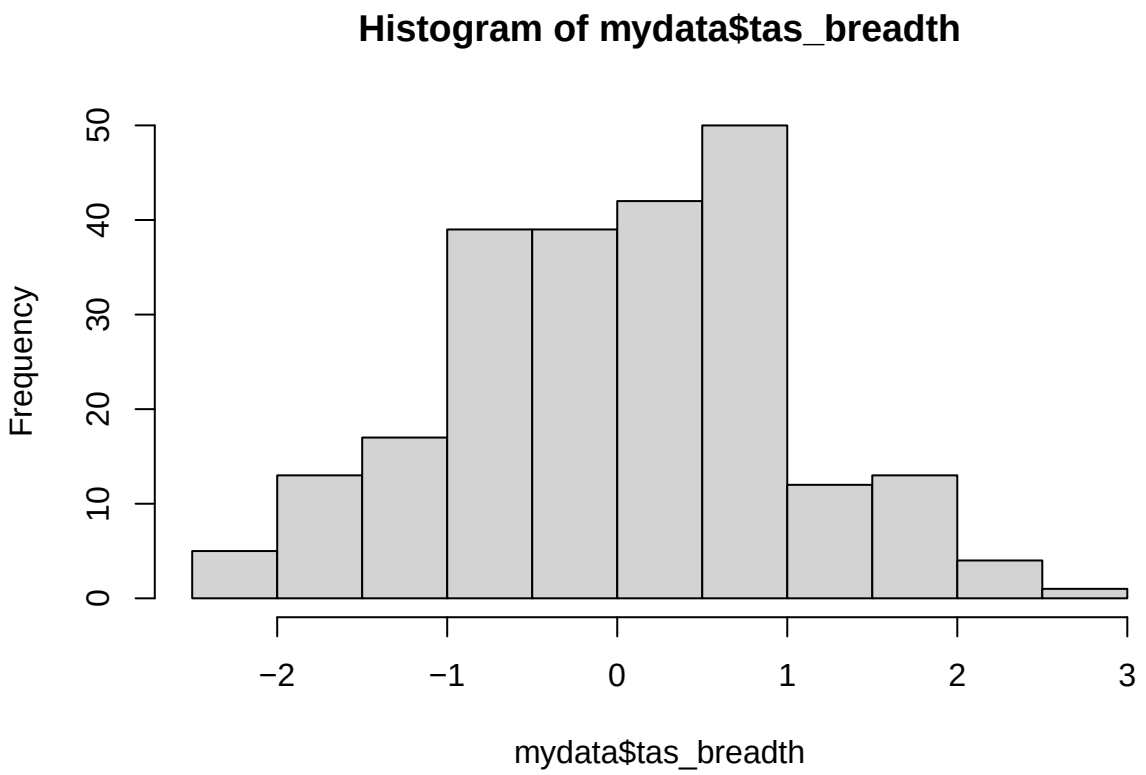
```
##            used  (Mb) gc trigger  (Mb) limit (Mb) max used  (Mb)
## Ncells  4337026 231.7    8226360 439.4         NA  7048029 376.5
## Vcells  8911268  68.0   15325476 117.0      16384 15325476 117.0
```

PREP FOR PCA

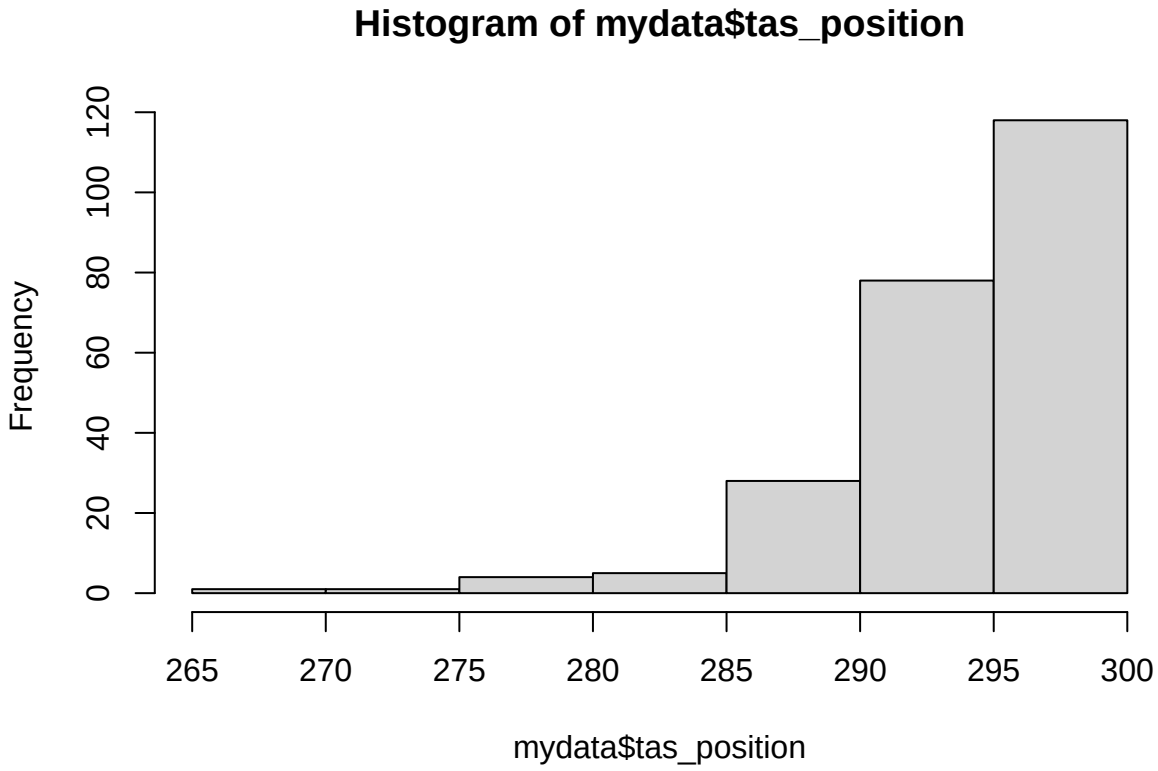
```
#######################################
# prepare raw predictors for PCA
hist(mydata$tas_breadth)
```



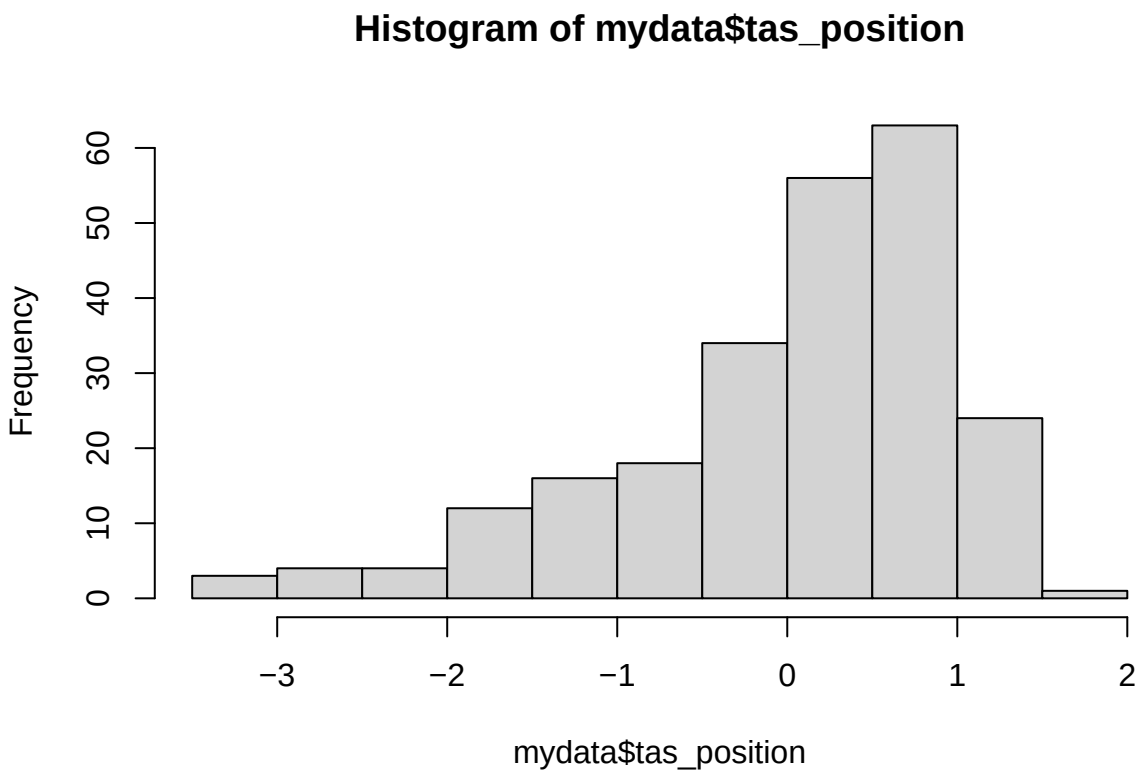
```
mydata$tas_breadth <- scale(myBCtransform(mydata$tas_breadth))
hist(mydata$tas_breadth)
```



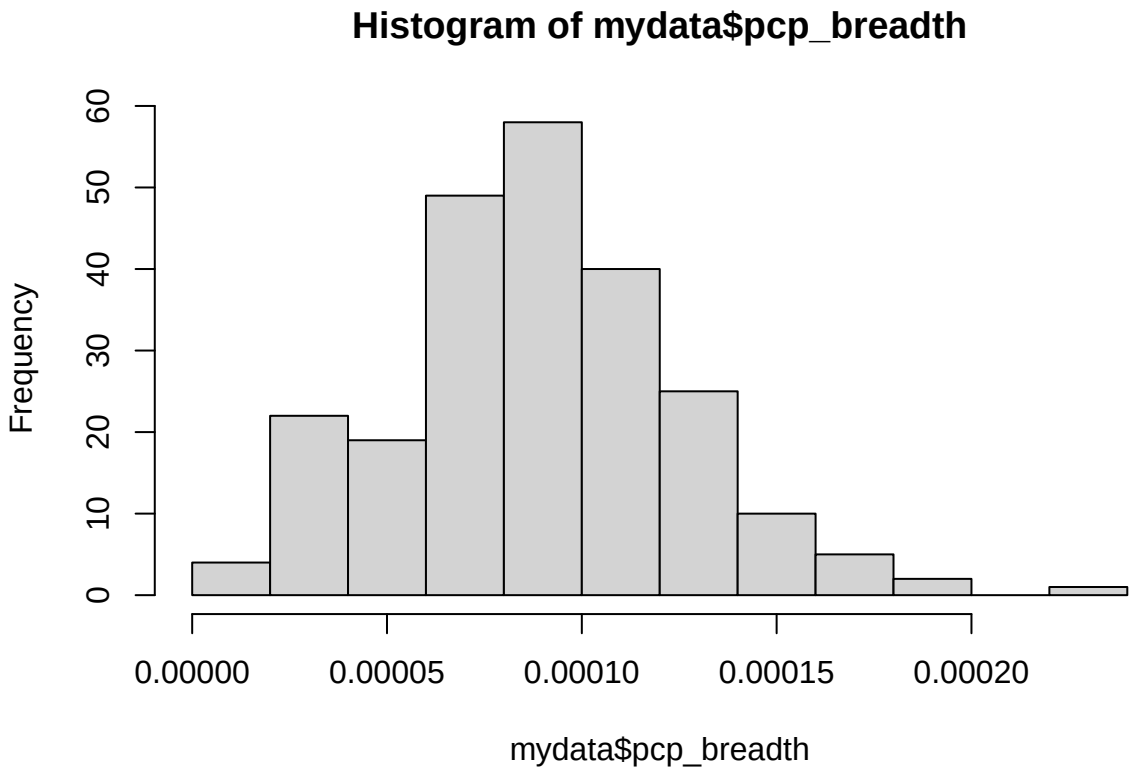

```
hist(mydata$tas_position)
```



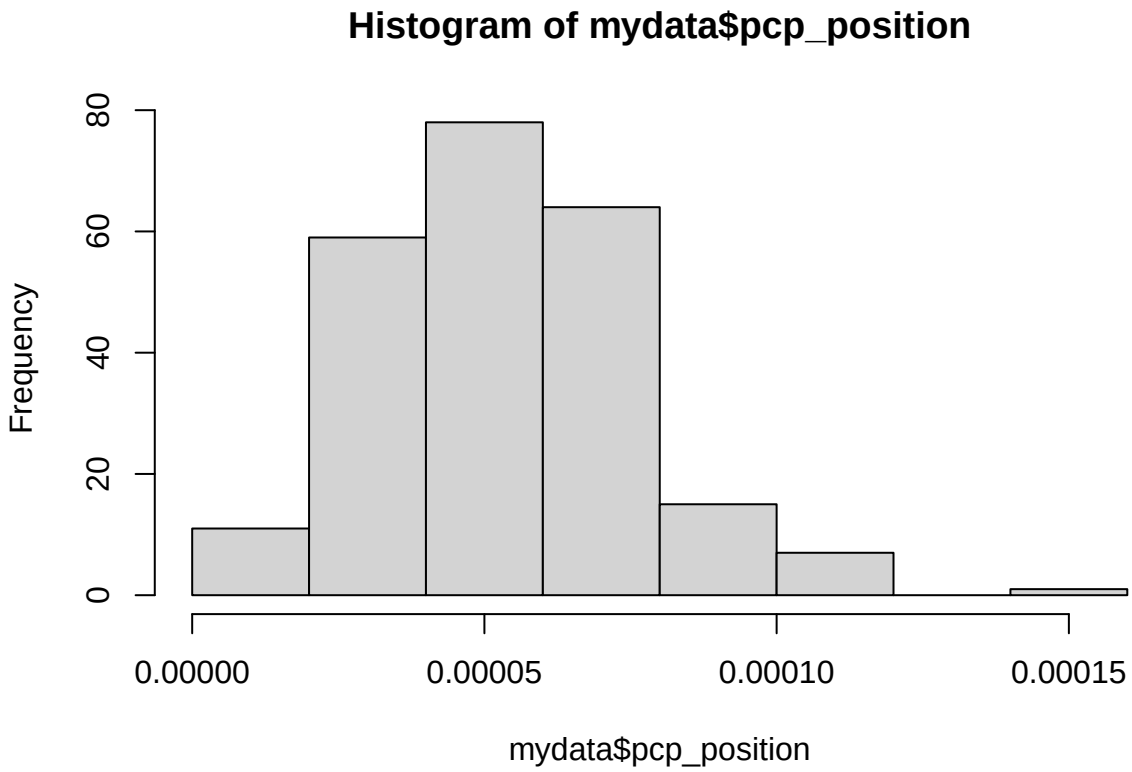
```
mydata$tas_position <- scale(myBCtransform(mydata$tas_position))
hist(mydata$tas_position)
```



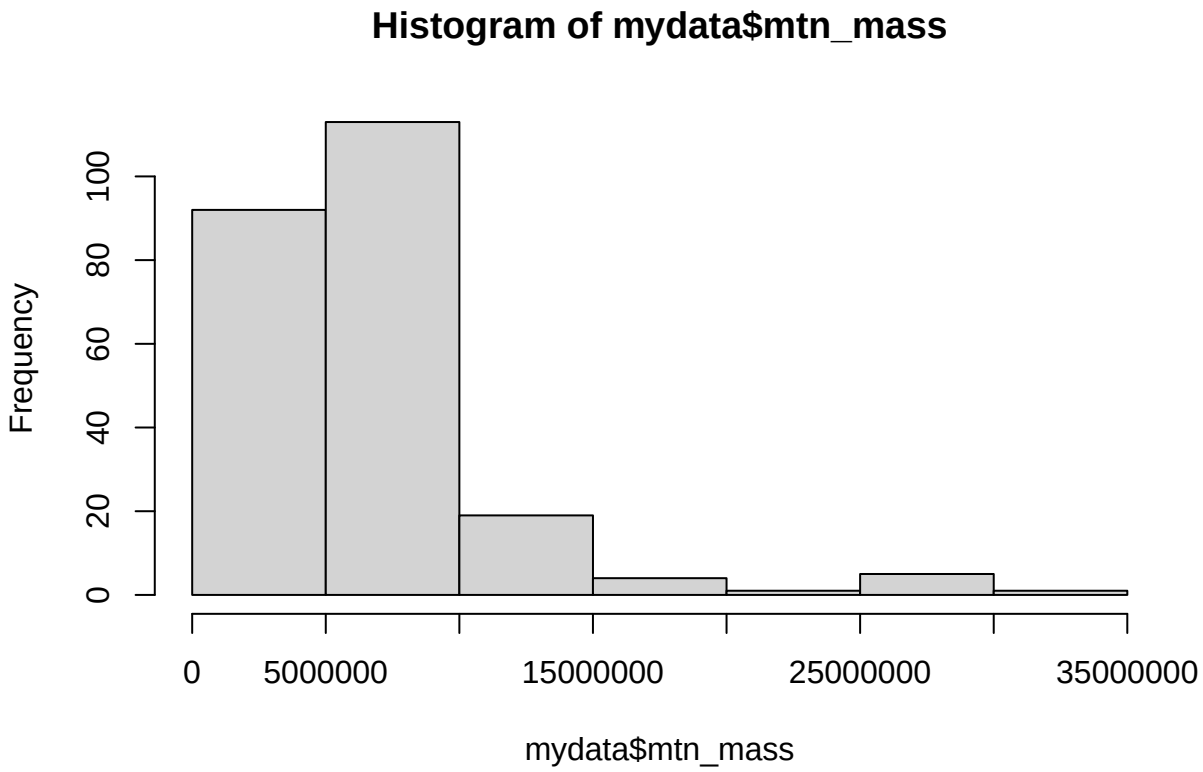

```
hist(mydata$pcp_breadth)
```



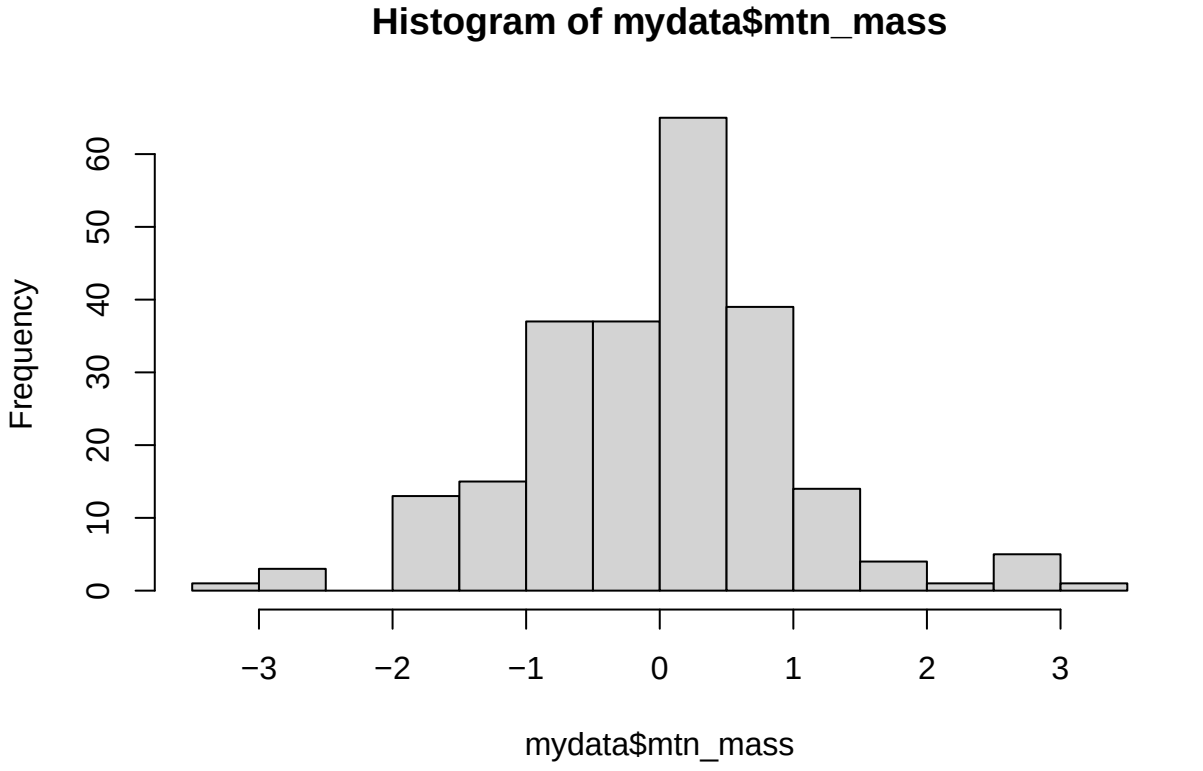
```
mydata$pcp_breadth <- scale(mydata$pcp_breadth)

hist(mydata$pcp_position)
```



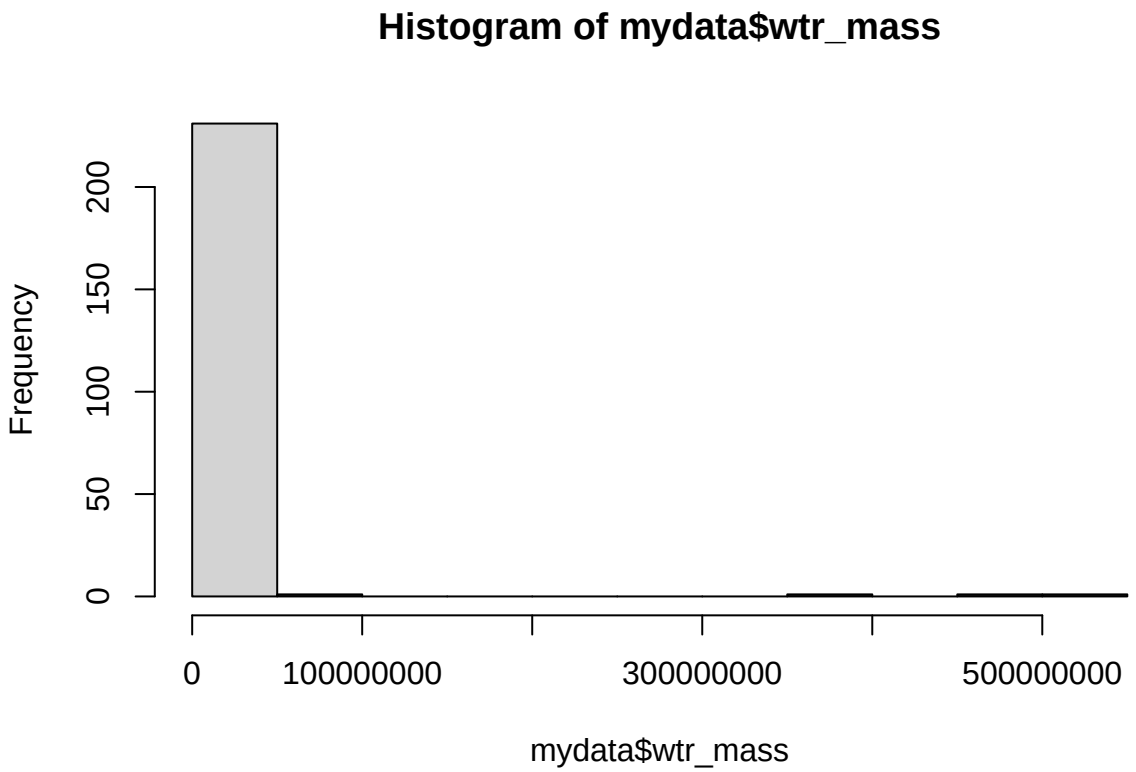

```
mydata$pcp_position <- scale(mydata$pcp_position)

hist(mydata$mtn_mass)
```



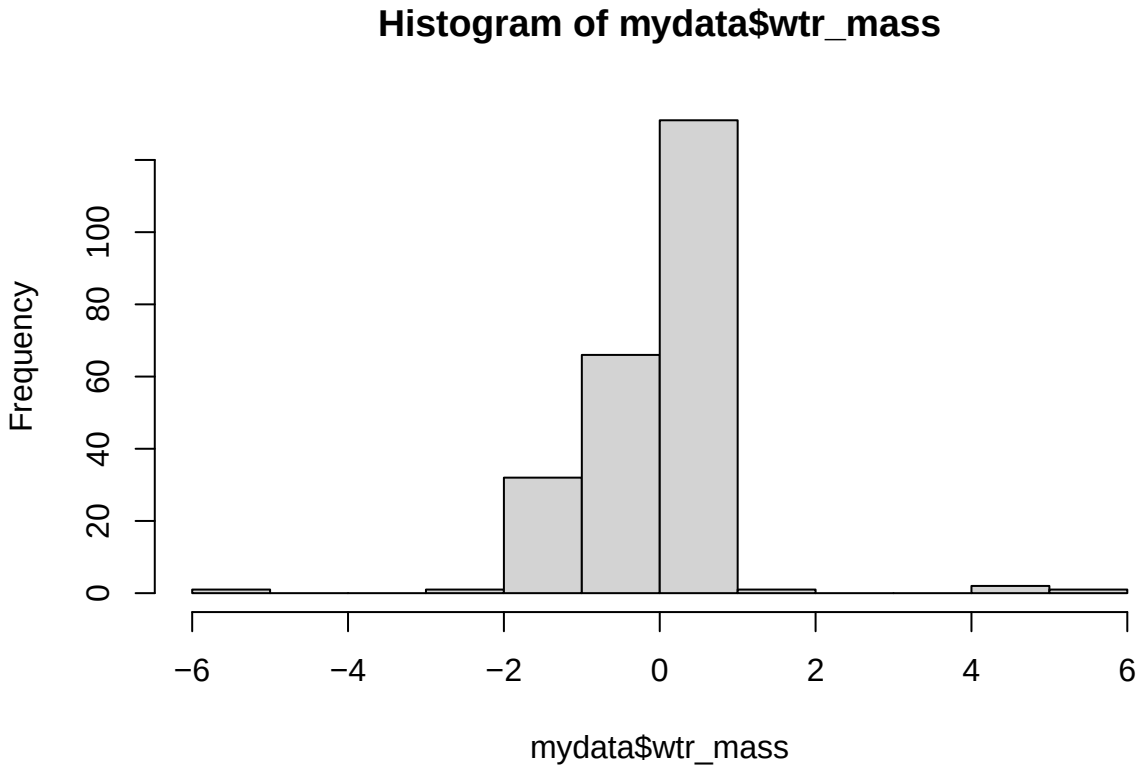

```
mydata$mtn_mass <- scale(myBCtransform(mydata$mtn_mass))
hist(mydata$mtn_mass)
```



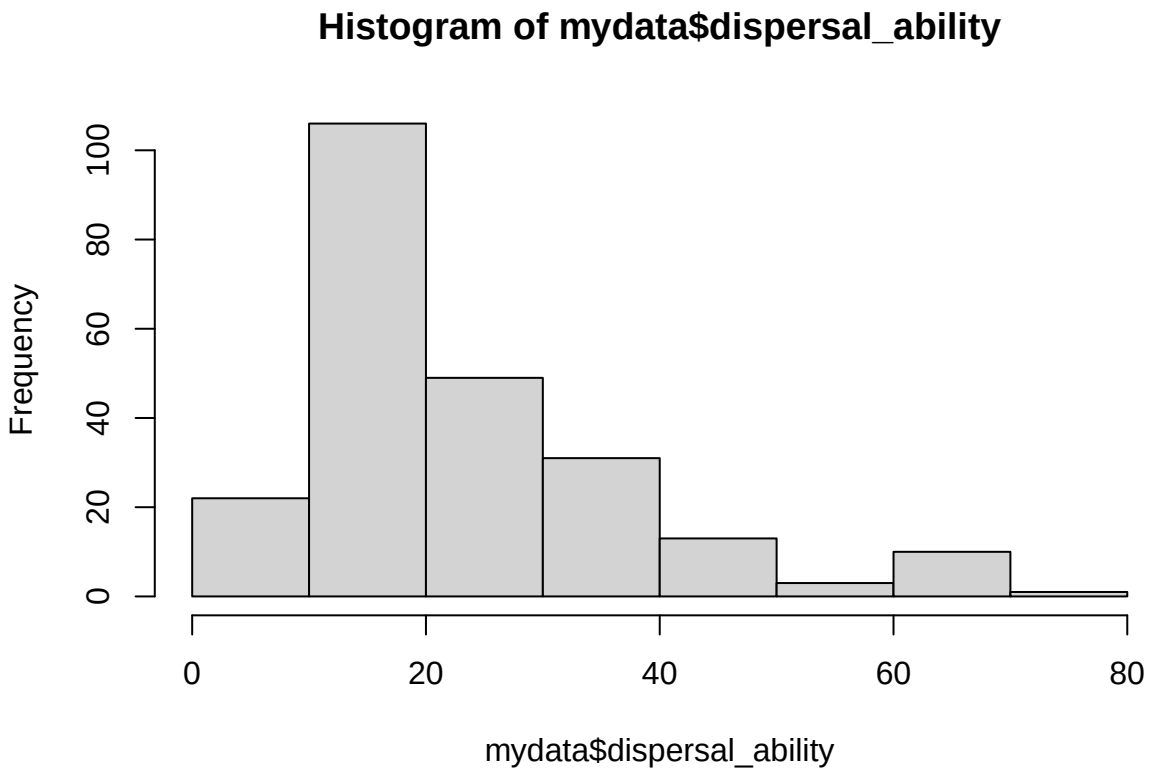
```
hist(mydata$wtr_mass)
```



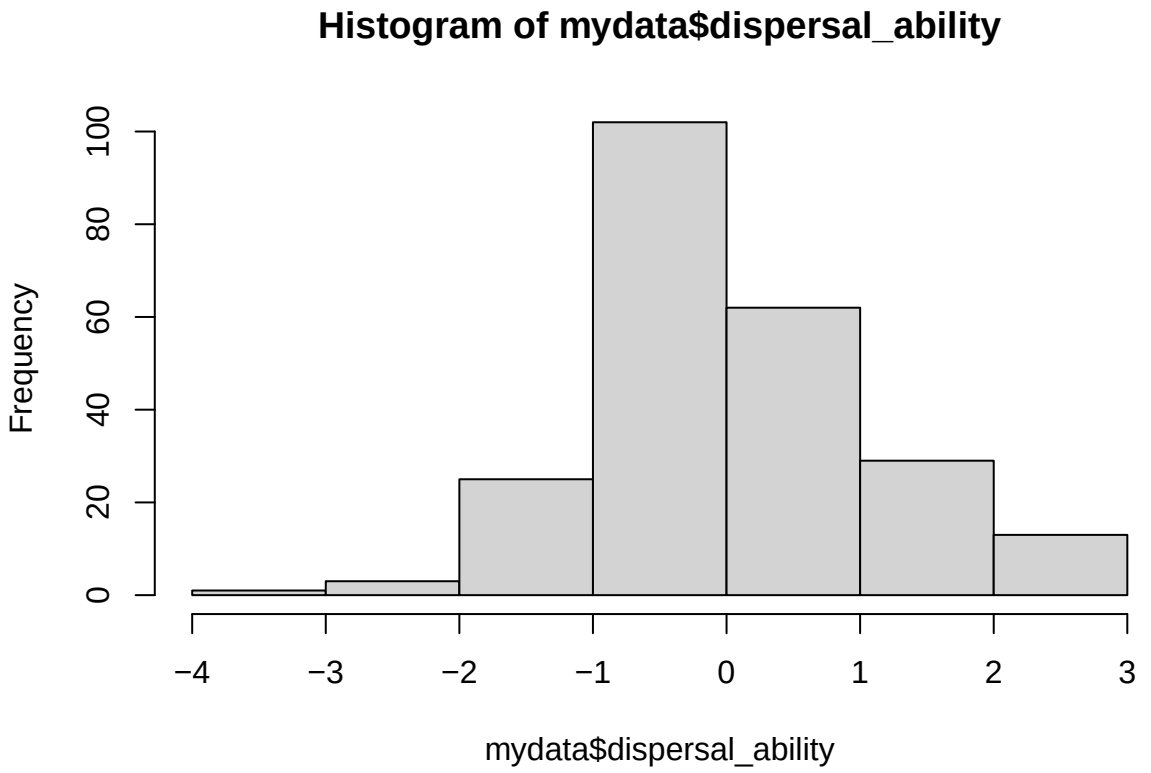

```
mydata$wtr_mass <- scale(myBCtransform(mydata$wtr_mass))
hist(mydata$wtr_mass)
```



```r
hist(mydata$dispersal_ability)
```

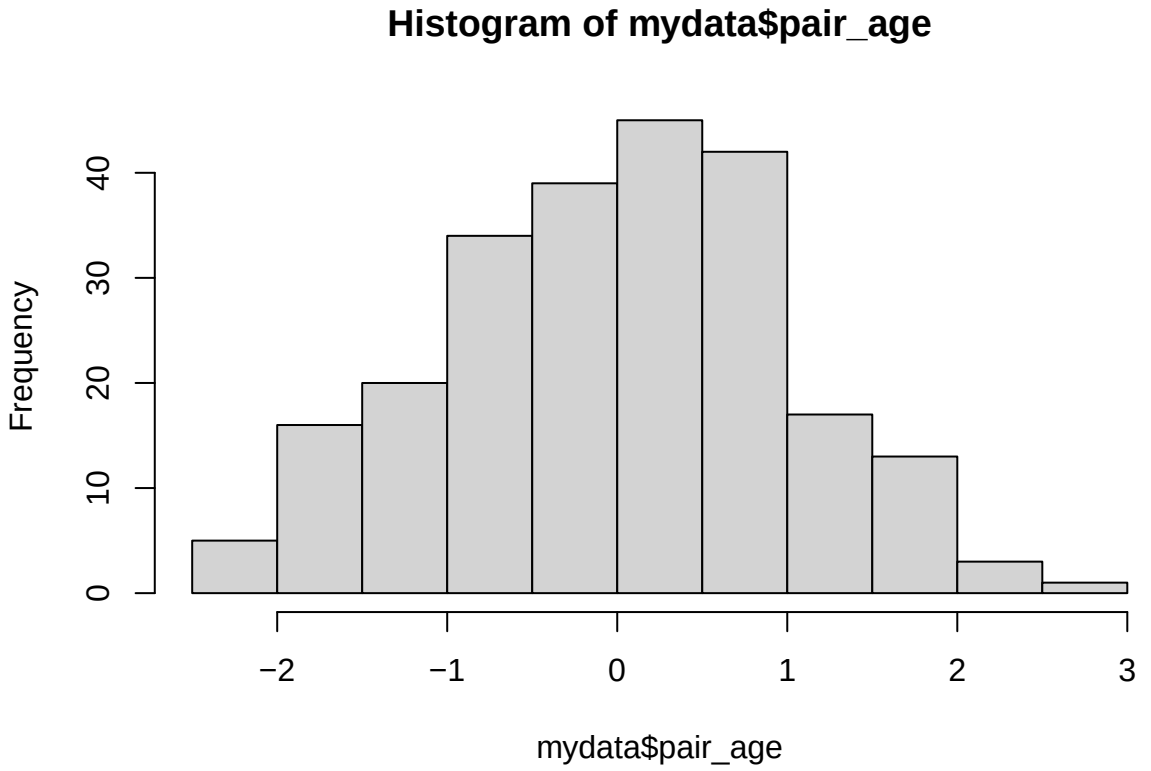



```r
mydata$dispersal_ability <- scale(myBCtransform(mydata$dispersal_ability))
hist(mydata$dispersal_ability)
```

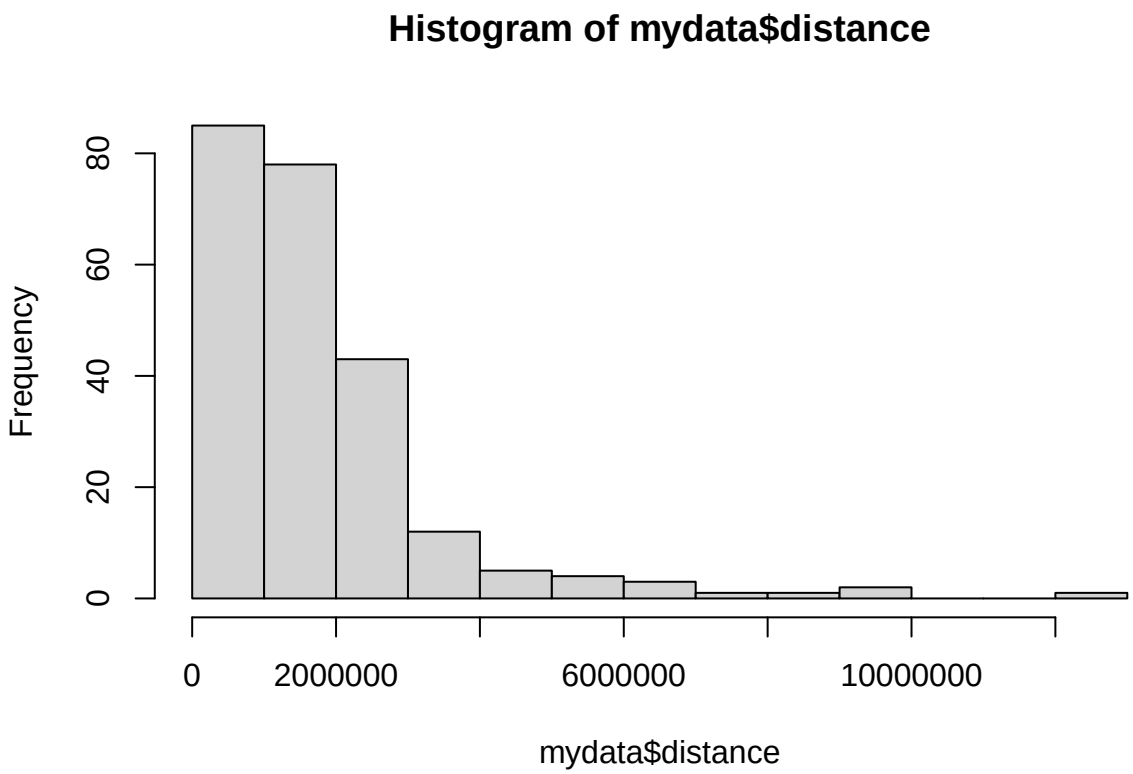


```r
hist(mydata$pair_age)
```

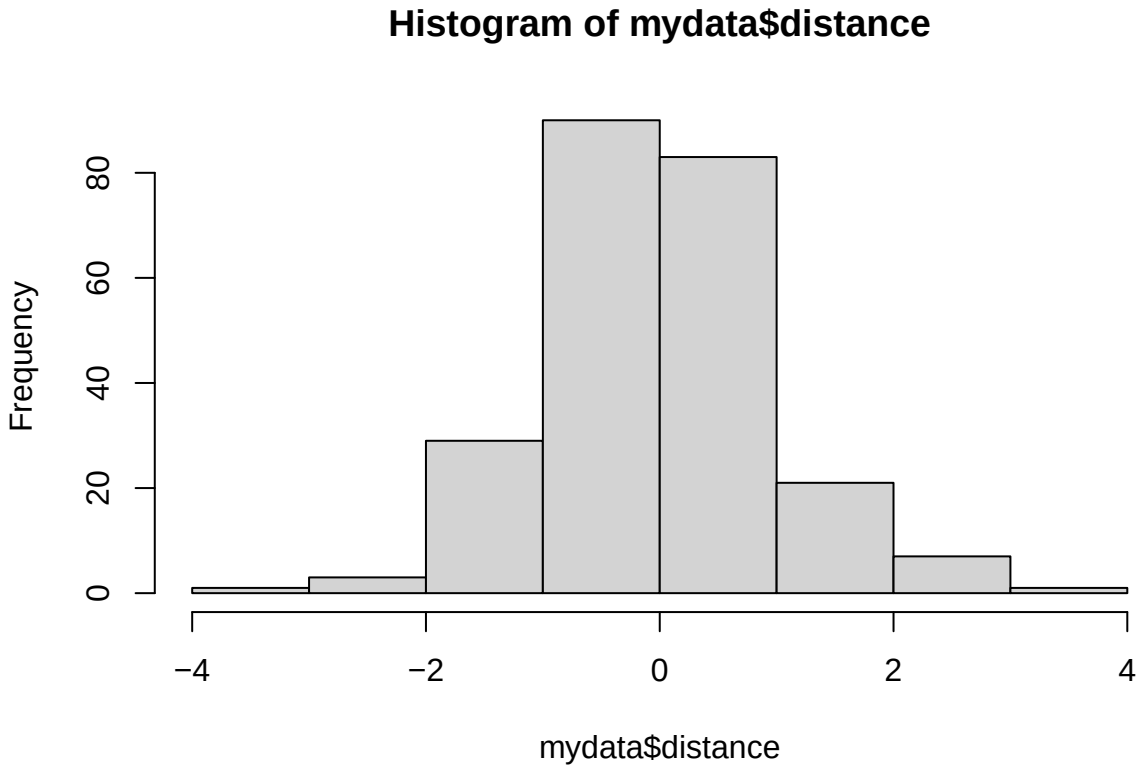

```r
mydata$pair_age <- scale(myBCtransform(mydata$pair_age))
hist(mydata$pair_age)
```



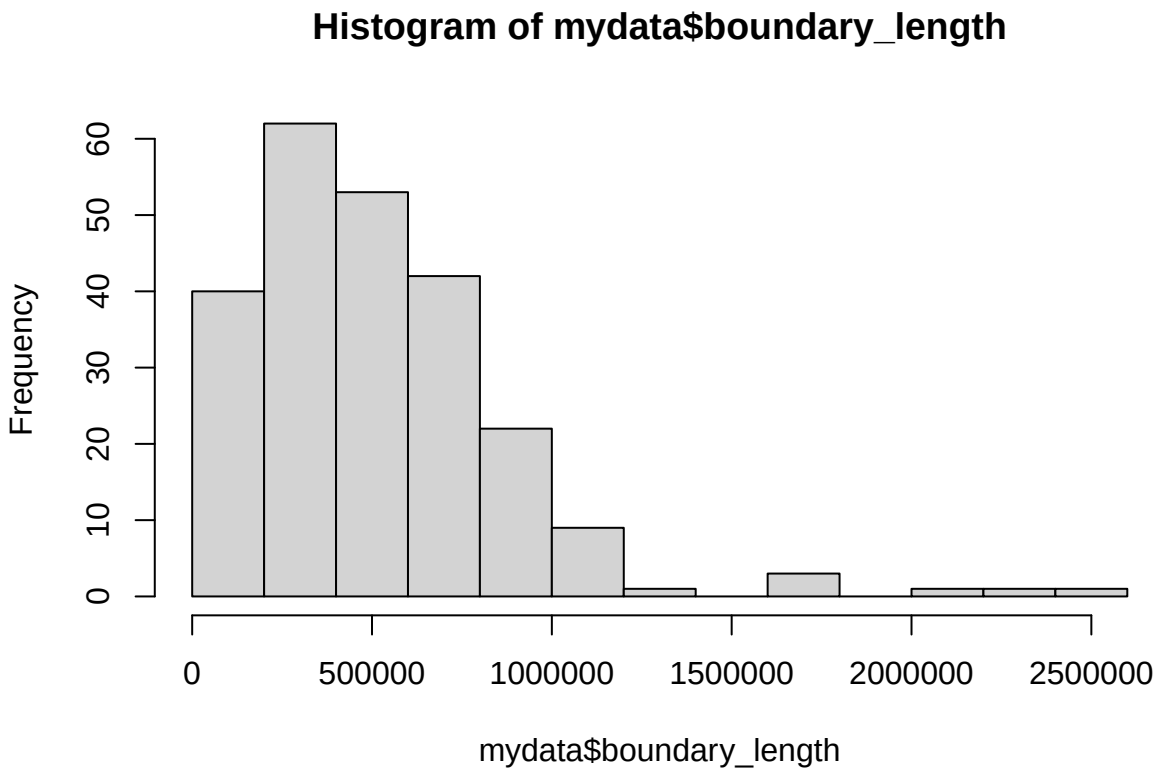
```
hist(mydata$distance)
```



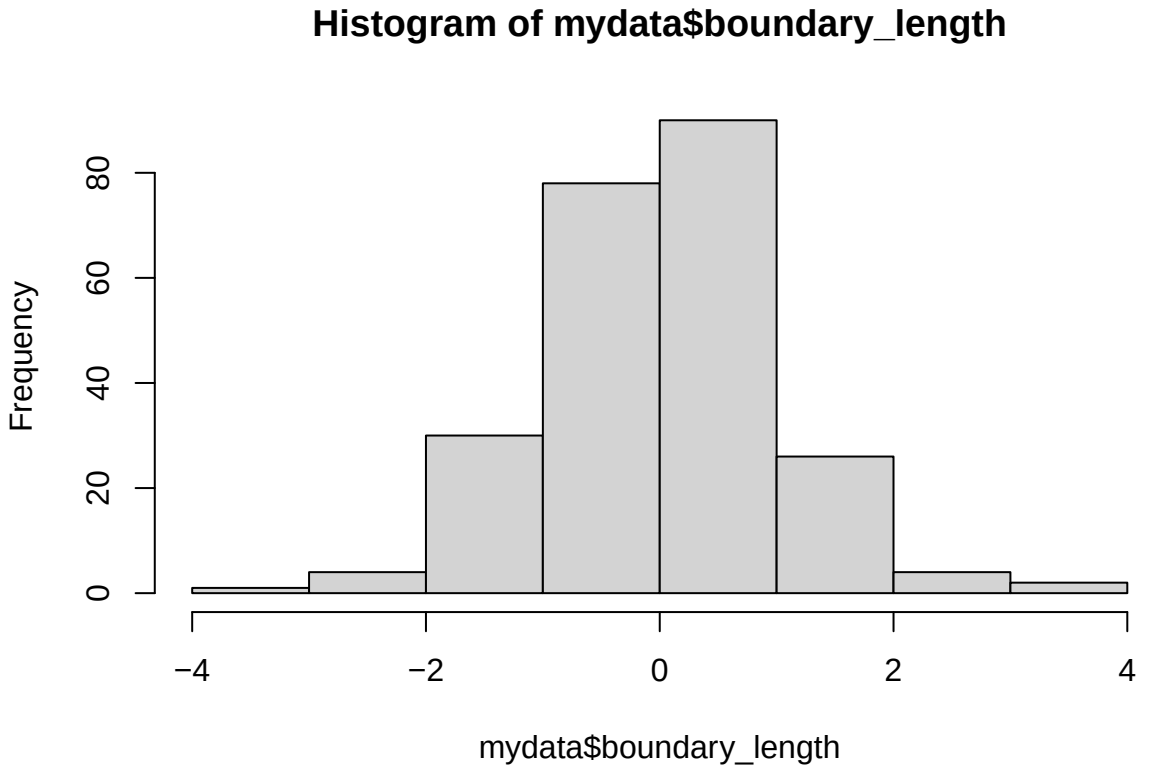

```
mydata$distance <- scale(myBCtransform(mydata$distance))
hist(mydata$distance)
```



```
hist(mydata$boundary_length)
```




```
mydata$boundary_length <- scale(myBCtransform(mydata$boundary_length))
hist(mydata$boundary_length)
```



```
#####################################
# Now run PCA of raw predictors
myPCA <- principal(mydata[,c('tas_breadth', 'tas_position', 'pcp_breadth',
                             'pcp_position', 'mtn_mass', 'wtr_mass',
                             'dispersal_ability', 'pair_age','distance',
                             'boundary_length')], nfactors = 9, rotate = "none")
myPCA$loadings
```

```
##
## Loadings:
##                   PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8
## tas_breadth        0.817  0.286 -0.344 -0.101  0.115
## tas_position      -0.338 -0.659  0.402  0.153 -0.245         0.415
## pcp_breadth       -0.666  0.485  0.416 -0.137
## pcp_position      -0.764  0.239  0.405         0.212 -0.151 -0.229  0.127
## mtn_mass          -0.123  0.790                0.246  0.103  0.514
## wtr_mass          -0.387 -0.466 -0.363  0.223  0.604 -0.232  0.109
## dispersal_ability  0.100  0.189  0.182  0.868  0.137  0.358 -0.127
## pair_age           0.228 -0.423  0.400 -0.491  0.375  0.473
## distance           0.613         0.584         0.185 -0.356        -0.344
## boundary_length    0.620  0.109  0.633  0.133        -0.147  0.116  0.378
##                   PC9
## tas_breadth        0.262
## tas_position       0.118
## pcp_breadth        0.326
## pcp_position      -0.110
## mtn_mass          -0.107
## wtr_mass           0.107
## dispersal_ability
## pair_age
## distance
## boundary_length
##
##                  PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8   PC9
## SS loadings    2.795 1.878 1.686 1.126 0.739 0.590 0.543 0.298 0.224
## Proportion Var 0.279 0.188 0.169 0.113 0.074 0.059 0.054 0.030 0.022
## Cumulative Var 0.279 0.467 0.636 0.748 0.822 0.881 0.936 0.965 0.988
```

```
# looks like the first four components properly load all the variables in the set
# at least once (I label them for now simply using the highest loaded variable)

mydata$PC1.tasbreadth <- myPCA$scores[,'PC1']
mydata$PC2.mtnmass <- myPCA$scores[,'PC2']
mydata$PC3.boundarylength <- myPCA$scores[,'PC3']
mydata$PC4.dispersalability <- myPCA$scores[,'PC4']
```

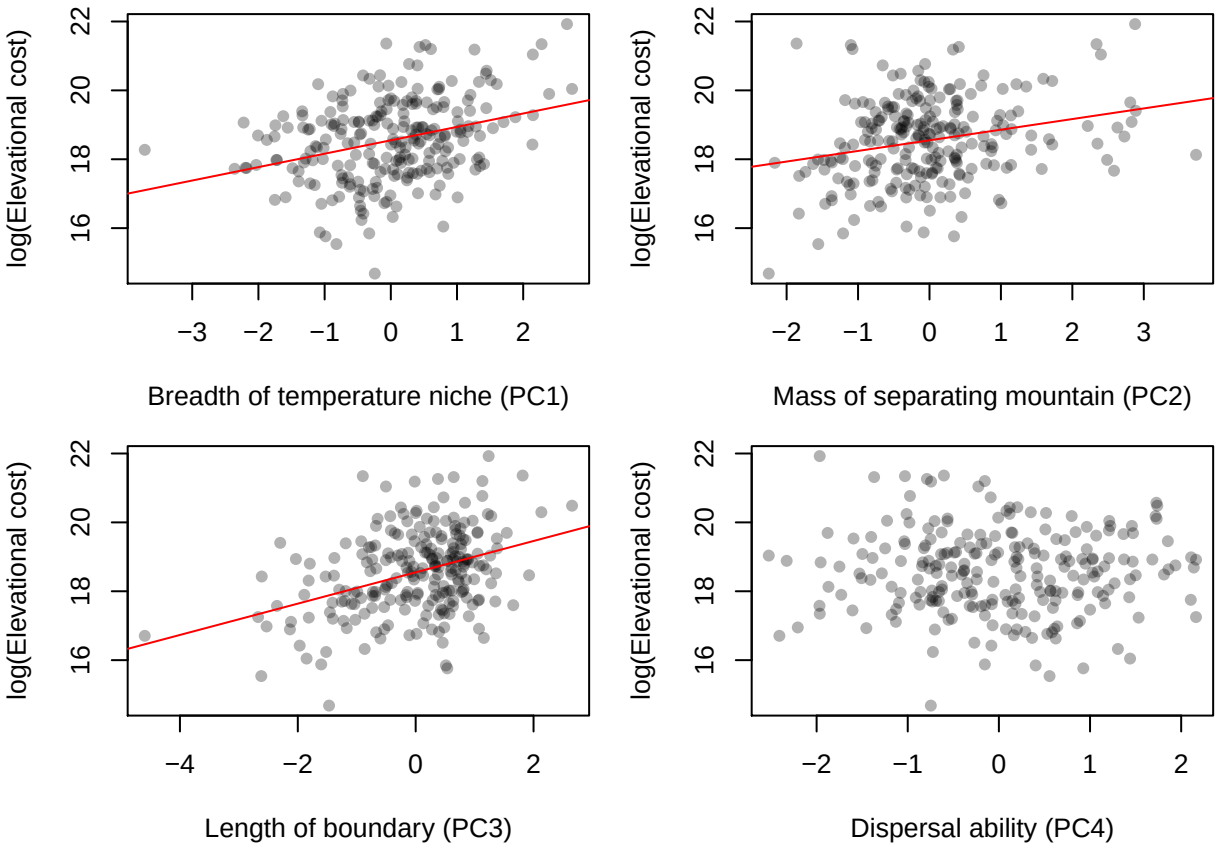
```
#####################################
# now we can run the pGLS
# load phylogenetic hypotheses
load("~/Box Sync/CB_VF_Shared/Dry_Lab/Projects/JMPH/Other_Input_Data/BirdTrees/BirdTrees.Rdata")
# first we deal with phylogenetic non-independence by using the tip of one of the
# species in each pair as the placement of the pair in the tree
CHK <- geiger::name.check(trees[[1]], mydata, data.names = mydata$Species.1)

# change the 'size' to iterate through different topology options
```

```
All.mods <- list()
counter <- 1
for (i in sample(x = 1:length(trees), size = 1)) {
  i <- 1
  mytree <- drop.tip(trees[[i]], CHK$tree_not_data)

  mymod <- phylolm(I(log(cost)) ~ PC1.tasbreadth + PC2.mtnmass +
                     PC3.boundarylength + PC4.dispersalability,
                   phy = mytree, model = 'lambda', data = mydata)

  All.mods[[counter]] <- mymod
  counter <- counter+1
}

# still need to figure out how to summarize across trees (talk to Angela about it
# as she and I have already discussed how to do this). For now...

summary (All.mods[[1]])
```

```
##
## Call:
## phylolm(formula = I(log(cost)) ~ PC1.tasbreadth + PC2.mtnmass +
##     PC3.boundarylength + PC4.dispersalability, data = mydata,
##     phy = mytree, model = "lambda")
##
##    AIC logLik
##  681.1 -333.5
##
## Raw residuals:
##      Min       1Q   Median       3Q      Max
## -2.73216 -0.60811 -0.09543  0.74091  2.57661
##
## Mean tip height: 104.1374
## Parameter estimate(s) using ML:
## lambda : 0.0000001
## sigma2: 0.009609122
##
## Coefficients:
##                       Estimate    StdErr  t.value               p.value
## (Intercept)          18.550271  0.065960 281.2338 < 0.00000000000000022 ***
## PC1.tasbreadth        0.389256  0.066101   5.8888       0.0000000136908 ***
## PC2.mtnmass           0.308573  0.066101   4.6682       0.0000051630607 ***
## PC3.boundarylength    0.454677  0.066101   6.8785       0.0000000000566 ***
## PC4.dispersalability -0.022863  0.066101  -0.3459                0.7298
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-squared: 0.3112    Adjusted R-squared: 0.2992
##
## Note: p-values and R-squared are conditional on lambda=0.0000001.
```

```
layout(matrix(1:4, 2, 2, byrow = T))

par(mar = c(4.1,4.1,1,1))
```

```
plot(mydata$PC1.tasbreadth, log(mydata$cost), pch = 16, col = rgb(0,0,0,0.3),
     xlab = 'Breadth of temperature niche (PC1)', ylab = 'log(Elevational cost)')
abline(a=mymod$coefficients['(Intercept)'],
       b = mymod$coefficients['PC1.tasbreadth'],
       col = 'red')

plot(mydata$PC2.mtnmass, log(mydata$cost), pch = 16, col = rgb(0,0,0,0.3),
     xlab = 'Mass of separating mountain (PC2)', ylab = 'log(Elevational cost)')
abline(a=mymod$coefficients['(Intercept)'],
       b = mymod$coefficients['PC2.mtnmass'],
       col = 'red')

plot(mydata$PC3.boundarylength, log(mydata$cost), pch = 16, col = rgb(0,0,0,0.3),
     xlab = 'Length of boundary (PC3)', ylab = 'log(Elevational cost)')
abline(a=mymod$coefficients['(Intercept)'],
       b = mymod$coefficients['PC3.boundarylength'],
       col = 'red')

plot(mydata$PC4.dispersalability, log(mydata$cost), pch = 16, col = rgb(0,0,0,0.3),
     xlab = 'Dispersal ability (PC4)', ylab = 'log(Elevational cost)')
```



load phylo and prune

```
# phylo ----------------------------------------
setwd("~/Box Sync/CB_VF_Shared/Dry_Lab/Projects/JMPH/Other_Input_Data/BirdTrees")
load(file = "BirdTrees.Rdata")
tree <- trees[[1]]; rm(trees) # pick tree (VF GET TREES FROM COONEY!!! and use MCC tree.) -- currently
```

```
tree <- drop.tip(tree, tree$tip.label[which(tree$tip.label %notin% mydata$Species.1)]) # initial name m
mydata <- mydata[which(mydata$Species.1 %in% tree$tip.label),]
mydata <- mydata[match(tree$tip.label, mydata$Species.1),]
sum(mydata$Species.1 != tree$tip.label) # sorted (but still specify form below for safety)
```

```
## [1] 0
```

Neighbors for sptial analysis.

```
distm <- mydata[,c("lon", "lat")]
distm <- geodist(distm, measure = "geodesic")
rownames(distm) <- rownames(mydata)
colnames(distm) <- rownames(mydata)
basecols <- ncol(mydata)

# neightbors
coords<-cbind(mydata$lon, mydata$lat); coords<-as.matrix(coords) ; row.names(coords)<-rownames(mydata)
k1 <- knn2nb(knearneigh(coords, longlat = T))
nb<- dnearneigh(coords,row.names = row.names(coords), d1=0,d2=max(unlist(nbdists(k1, coords, longlat =
```

Sensitivity Analyses

```
# 1 Pair age (all v. < 8mya (end of uplift of Andes))
# 2 Distance (all v. < 1500*1000) (1500 / 110 = ~ 22 degrees)
# 3 MAT_overlap (> 0% v. > 75% (more restrictive == more conservative for this measure.))
# 4 landgap (all v. nogap) *ALL GAPS ARE < 110km (two water grid cells marked as land for having >50% l
```