Is species richness mediated by functional and genetic divergence? A global analysis in birds – Analysis workflow

Contents

1 Required libraries

```
# Data handling
library(dplyr)
library(purrr)
library(XML)
library(reshape2)

# Querying Genbank
library(rentrez)

# Handling sequence data
library(seqinr)
library(pegas)

# Spatial data
library(sf)
library(sp)

# Morphology analysis
library(distances)
library(geometry)
library(e1071)
library(dispRity)

# Path analysis
library(phylopath)

# Spatial analysis
library(DHARMa)
library(spaMM)

# Plotting
library(ggplot2)
library(ggthemes)
library(ggspatial)
library(viridis)
library(cowplot)
library(rnaturalearth)
library(rnaturalearthdata)
library(paletteer)
```

2 Shared plotting parameters

Here some common parameters for making plots are called to avoid having to repeat the code throughout the workflow.

```
# Scatterplots
my.theme.scat <- theme(axis.text=element_text(size=14),
        axis.title=element_text(size=14),
        strip.text.x = element_text(size = 14))
```

```r
# Boxplots
my.theme.bp <- theme_bw() +
  theme(axis.text=element_text(size=14),
        axis.title=element_text(size=14),
        strip.text.x = element_text(size = 14),
        legend.title = element_text(size = 14),
        legend.text = element_text(size = 12),
        legend.key.size = unit(1, 'cm'))

#gene.palette <- setNames(object = c("#9b7693",
#                                    "#ebc46e",
#                                    "#8da0cb",
#                                   "#8dd3c7"),
#                            nm = c("CO1",
#                                   "ND2",
#                                   "CYTB",
#                                   "mtDNA"))

gene.palette <- setNames(object = c("#A0C1B8",
                                    "#709fb0",
                                    "#726a95",
                                    "#F8EFB6",
                                    "#85496f",
                                    "#de7952",
                                    "#cfa463"),
                         nm = c("CO1",
                                "ND2",
                                "CYTB",
                                "mtDNA",
                                "MC1R",
                                "RAG1",
                                "EGR1"))
```

3 Morphology analyses

3.1 Global morphology maps

In this section we will generate global maps of the two aspects of morphology quantified in this study – nearest neighbor distance (NND) and convex hull sizes (representing dispersion).

3.1.1 NND scores

To generate these maps we first need to load the community matrix describing presence/absence of species.

```
world.comm.matrix <- read.csv("data/world_community_matrix_raw_no_duplicates.csv")
```

The first two columns of this `data.frame` correspond to the `x` and `y` coordinates of each point, with the remainder being the presence/absence scores for each species at the corresponding point. Let's separate out just the species scores.

```
spp.presence <- world.comm.matrix[,-c(1:2)]

coords <- world.comm.matrix[,1:2]
```

Morphological data for this study is taken from Pigot et al. (2020), who compiled a near exhaustive list of morphological data for birds. Pigot et al. provided PC scores for 9,963 species both for analyses of overall morphology as well as just of the beak. Here, these data are loaded.

```
pigot.all <- read.csv("data/Pigot_etal_morpho.csv")
pigot.beak <- read.csv("data/Pigot_etal_beak.csv")
```

NND scores will be generated for the two datasets respectively using all sampled species, regardless of whether they will be included in subsequent analyses. The `distances` function from the package of the same name will generate a matrix of the pairwise distances between all species.

```
all.nnd <- distances(pigot.all[,-1]) %>% as.matrix
rownames(all.nnd) <- colnames(all.nnd) <- pigot.all$ï..Binomial

beak.nnd <- distances(pigot.beak[,-1]) %>% as.matrix
rownames(beak.nnd) <- colnames(beak.nnd) <- pigot.beak$ï..Binomial
```

Next, we need to load a custom function that will calculate the mean NND score for each point, given the species that are present. We also need to load taxonomic and synonymy data.

```
source("scripts/calculate.mean.nnd.R")

# Jetz et al. (2012)
taxo <- read.csv("data/BLIOCPhyloMasterTax.csv")

# Crouch, in press
syn.dat <- read.csv("data/synonymy_v6.csv")
```

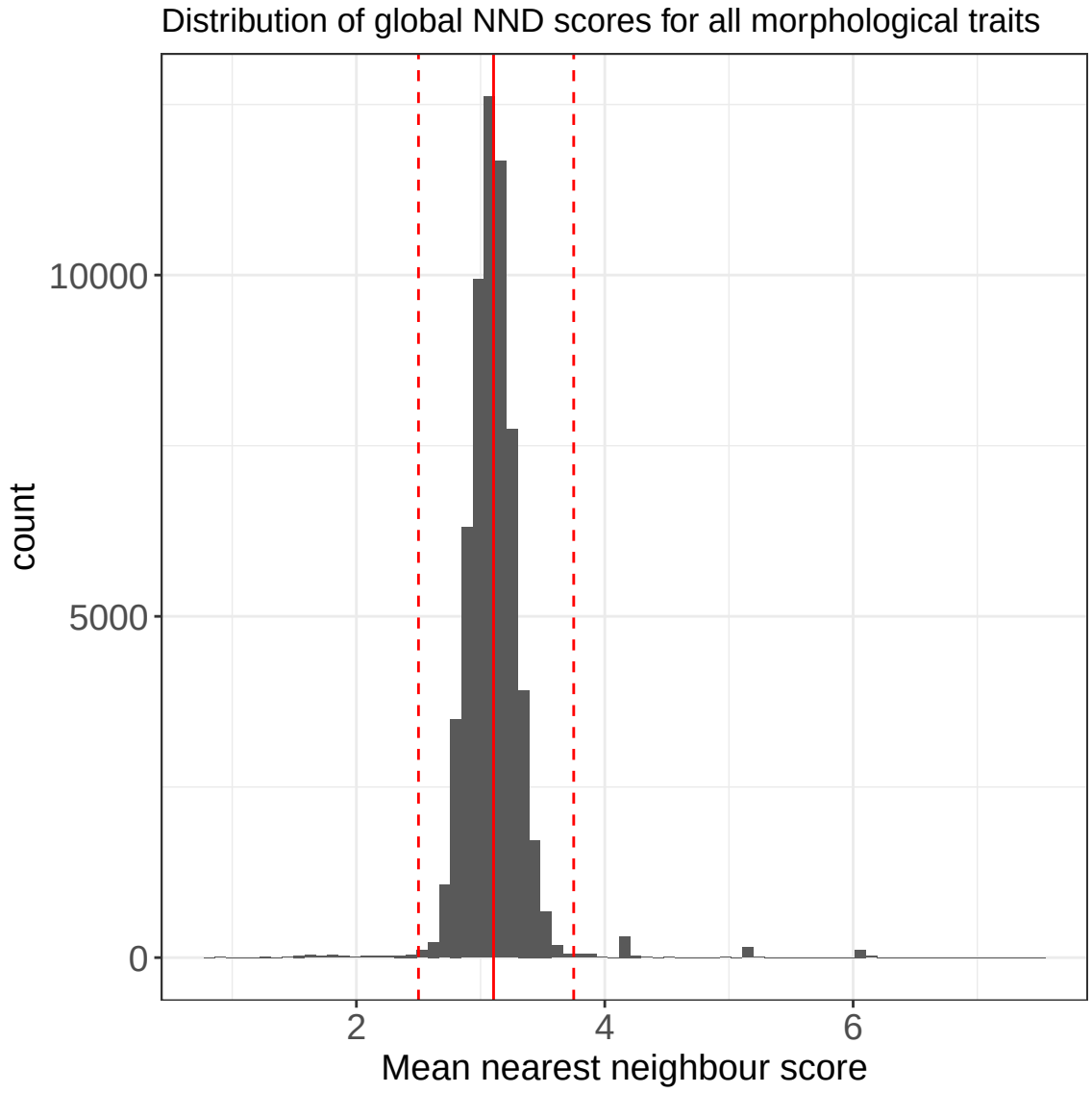
Now the function can be run using the global community matrix.

```
# All traits
global.nnd.scores <- lapply(1:nrow(spp.presence), function(ee){
  calculate.mean.nnd(spp.presence[ee,],
                     all.nnd,
                     syn.dat)
}) %>% unlist

# Beak traits
global.nnd.scores.beak <- lapply(1:nrow(spp.presence), function(ee){
  calculate.mean.nnd(spp.presence[ee,],
                     beak.nnd,
                     syn.dat)
}) %>% unlist
```

The raw values for both the analysis of all traits and of only beak traits are highly clustered around the mean with long tails in both directions.

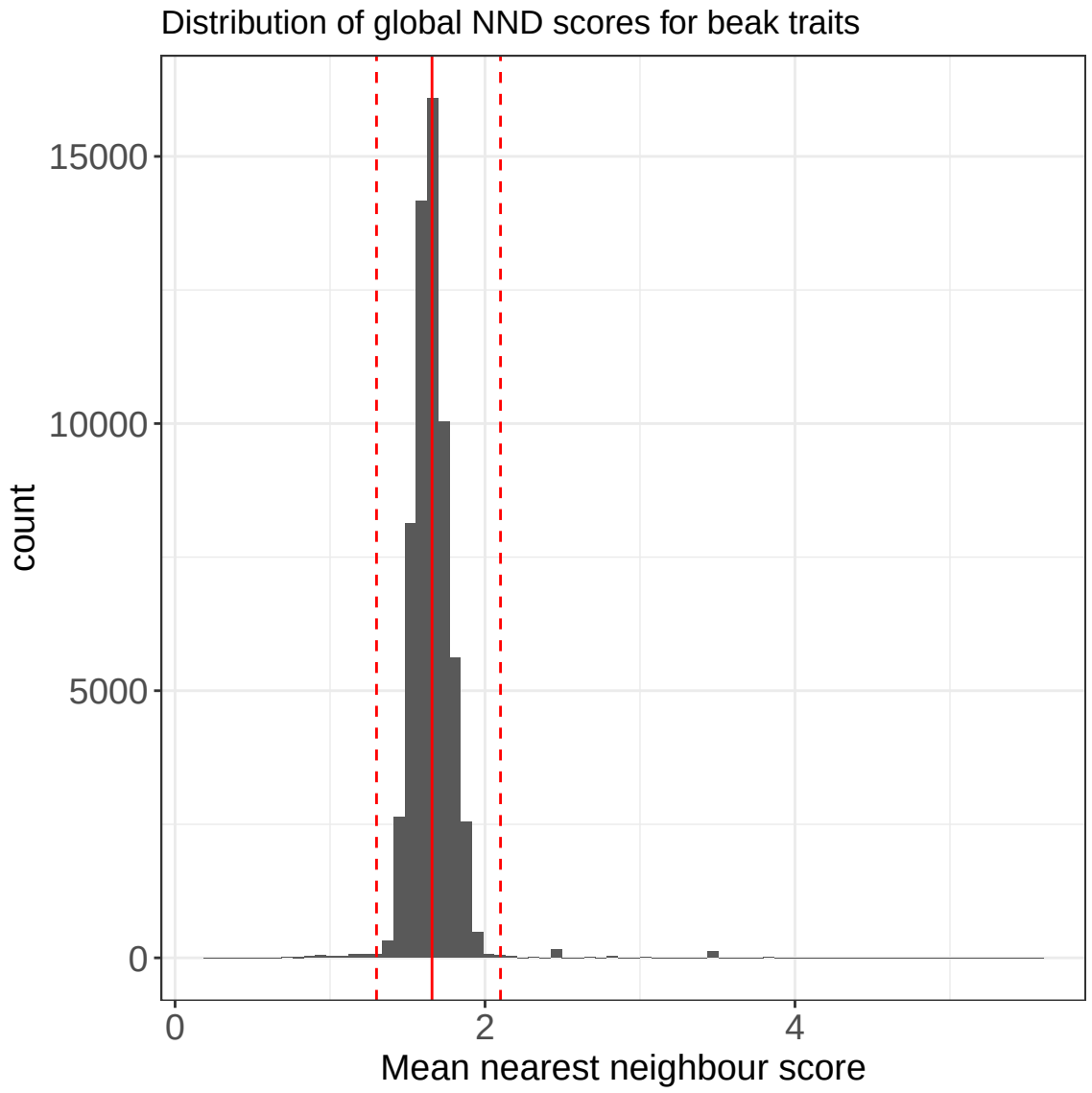
```
ggplot(as.data.frame(global.nnd.scores),
       aes(x = global.nnd.scores)) +
  geom_histogram(bins = 75) +
  my.theme.bp +
  labs(x = "Mean nearest neighbour score") +
  geom_vline(xintercept = mean(global.nnd.scores, na.rm=T),
             color = "red") +
  geom_vline(xintercept = c(2.5, 3.75),
             color = "red",
             linetype = "dashed") +
  ggtitle("Distribution of global NND scores for all morphological traits")
```

```
## Warning: Removed 637 rows containing non-finite values (stat_bin).
```




```
ggplot(as.data.frame(global.nnd.scores.beak),
       aes(x = global.nnd.scores.beak)) +
  geom_histogram(bins = 75) +
  my.theme.bp +
  labs(x = "Mean nearest neighbour score") +
  geom_vline(xintercept = mean(global.nnd.scores.beak, na.rm=T),
             color = "red") +
  geom_vline(xintercept = c(1.3, 2.1),
             color = "red",
             linetype = "dashed") +
  ggtitle("Distribution of global NND scores for beak traits")
```

```
## Warning: Removed 637 rows containing non-finite values (stat_bin).
```



These data will be visualized in combination with information on species diveristy, calculated as the row sums from the world community matrix data. Let's load thoe data, and combine with the NND score results.

```
species.diversity <- readRDS("data/speciesDiversity.RDS")
plot.dat <- cbind(coords, global.nnd.scores, species.diversity)
```

Next, a bivariate color palette needs to be generated that describes the relationship between species diversity and mean NND score. Since this process is going to be called multiple times it has been wrapped into a function to make this workflow easier to follow.

```
source("scripts/make.bivariate.color.palette.df.R")

plot.dat$global.nnd.scores[plot.dat$global.nnd.scores < 2.5] <- NA
plot.dat$global.nnd.scores[plot.dat$global.nnd.scores > 3.75] <- NA
plot.dat.full <- make.bivariate.color.palette.df(plot.dat,
                                                 "species.diversity",
```

```
                                                    "global.nnd.scores")

plot.dat.full <- make.bivariate.color.palette.df(plot.dat,
                                                    species.diversity,
                                                    global.nnd.scores)
```

Now we can use this color palette to create our map, and combine this with a legend showing how the color corresponds to the values of the two variables (species diversity and mean NND).

```
global.map <- ggplot(plot.dat.full, aes(Var1, Var2)) +
  geom_point(aes(color=fill),
             shape = 15,
             size = .35) +
  scale_color_identity()+
  guides(color = "none") +
  coord_sf(xlim = c(-180, 189),
           ylim = c(-60, 83.5),
           expand = FALSE) +
  theme_bw() +
  labs(x = "Longitude",
       y = "Latitude") +
  theme(axis.text=element_text(size=14),
        axis.title=element_text(size=14),
        panel.background = element_rect(fill = "gray45"),
        panel.grid.major = element_line(colour = "gray50"))


inset.figure <- ggplot(plot.dat.full,
                       aes(species.diversity, global.nnd.scores)) +
  geom_point(aes(color = fill),
             size = .5) +
  scale_color_identity() +
  ylim(c(2.5, 3.75)) +
  labs(x = "Species diversity",
       y = "Mean NND score") +
  theme_bw()

png("FigureExport/NNDspeciesDiversity.png",
    width=12,
    height = 9,
    units="in",
    res=500)
ggdraw(global.map) +
  draw_plot(plot = inset.figure,
            x = .06,
            y = .29,
            width = .21,
            height = .25)
dev.off()
```

Now we can repeat the procedure for beak traits.

```
plot.dat.beak.NND <- cbind(coords,
                           global.nnd.scores.beak,
                           species.diversity)

plot.dat.beak.NND$global.nnd.scores.beak[
  plot.dat.beak.NND$global.nnd.scores.beak < 1.3] <- NA

plot.dat.beak.NND$global.nnd.scores.beak[
  plot.dat.beak.NND$global.nnd.scores.beak > 2.1] <- NA

plot.dat.beak.nnd.full <- make.bivariate.color.palette.df(plot.dat.beak.NND,
                                                          "species.diversity",
                                                          "global.nnd.scores.beak")


global.map.beak.nnd <- ggplot(plot.dat.beak.nnd.full,
                              aes(Var1, Var2)) +
  geom_point(aes(color=fill),
             shape = 15,
             size = .35) +
  scale_color_identity()+
  guides(color = "none") +
  coord_sf(xlim = c(-180, 189),
           ylim = c(-60, 83.5),
           expand = FALSE) +
  theme_bw() +
  labs(x = "Longitude",
       y = "Latitude") +
  theme(axis.text=element_text(size=14),
        axis.title=element_text(size=14),
        panel.background = element_rect(fill = "gray45"),
        panel.grid.major = element_line(colour = "gray50"))


inset.figure.beak.nnd <- ggplot(plot.dat.beak.nnd.full,
                        aes(species.diversity, global.nnd.scores.beak)) +
  geom_point(aes(color = fill),
             size = .5) +
  scale_color_identity() +
  ylim(c(1.3, 2.1)) +
  labs(x = "Species diversity",
       y = "Mean NND score") +
  theme_bw()


png("FigureExport/NNDbeakSpeciesDiversity.png",
    width=12,
    height = 9,
    units="in",
    res=500)
ggdraw(global.map.beak.nnd) +
  draw_plot(plot = inset.figure.beak.nnd,
            x = .06,
```

```
            y = .29,
            width = .21,
            height = .25)
dev.off()
```

3.1.2 Hull sizes

We can define a simple function for calculating the convex hull size for species in a geographic area.

```
get.hull.size <- function(x, pc.scores, syn.dat){
    spp.present <- colnames(x)[x==1]
    if(length(spp.present)>3){
    # update synonymy
    for(j in 1:length(spp.present)){
        spp.in <- syn.dat$JetzTip[syn.dat$geo.name == spp.present[j]] %>%
          as.character
        spp.in<- spp.in[!is.na(spp.in)]
        if(length(spp.in)==1){
            spp.present[j] <- spp.in
        }
    }

    target.pc <- pc.scores[pc.scores[,1] %in% spp.present, -1]
    target.size <- convhulln(target.pc[,1:3], "FA")$vol
    } else {
      target.size <- NA
    }
    return(target.size)
}
```

This function can now be applied to the two data sets

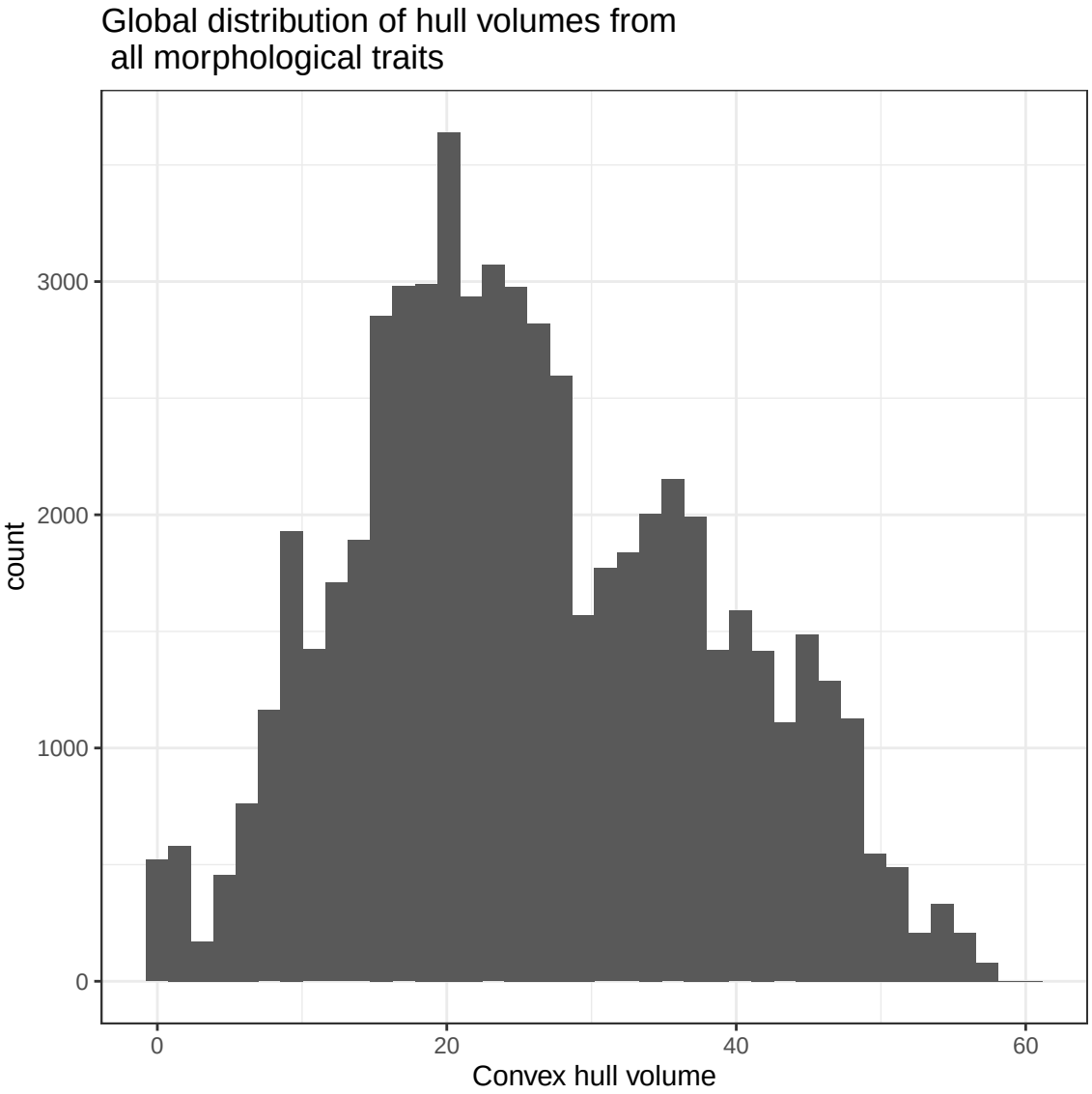
```
global.hull.size <- lapply(1:nrow(spp.presence), function(ee){
  print(paste(ee, "of", nrow(spp.presence)))
  tryCatch({
  get.hull.size(spp.presence[ee,],
                pigot.all,
                syn.dat)
  },
  error = function(cond){
    print("Not enough samples to generate result")
  })
}) %>% unlist
saveRDS(global.hull.size, file="global.hull.size.RDS")

# Beak traits
global.hull.size.beak <- lapply(1:nrow(spp.presence), function(ee){
  print(paste(ee, "of", nrow(spp.presence)))
    tryCatch({
  get.hull.size(spp.presence[ee,],
                pigot.beak,
                syn.dat)
    },
  error = function(cond){
    print("Not enough samples to generate result")
  })
}) %>% unlist
saveRDS(global.hull.size.beak, file="global.hull.size.beak.RDS")
```

```
## Warning in readRDS("data/global.hull.size.RDS") %>% as.numeric(): NAs introduced
## by coercion
```

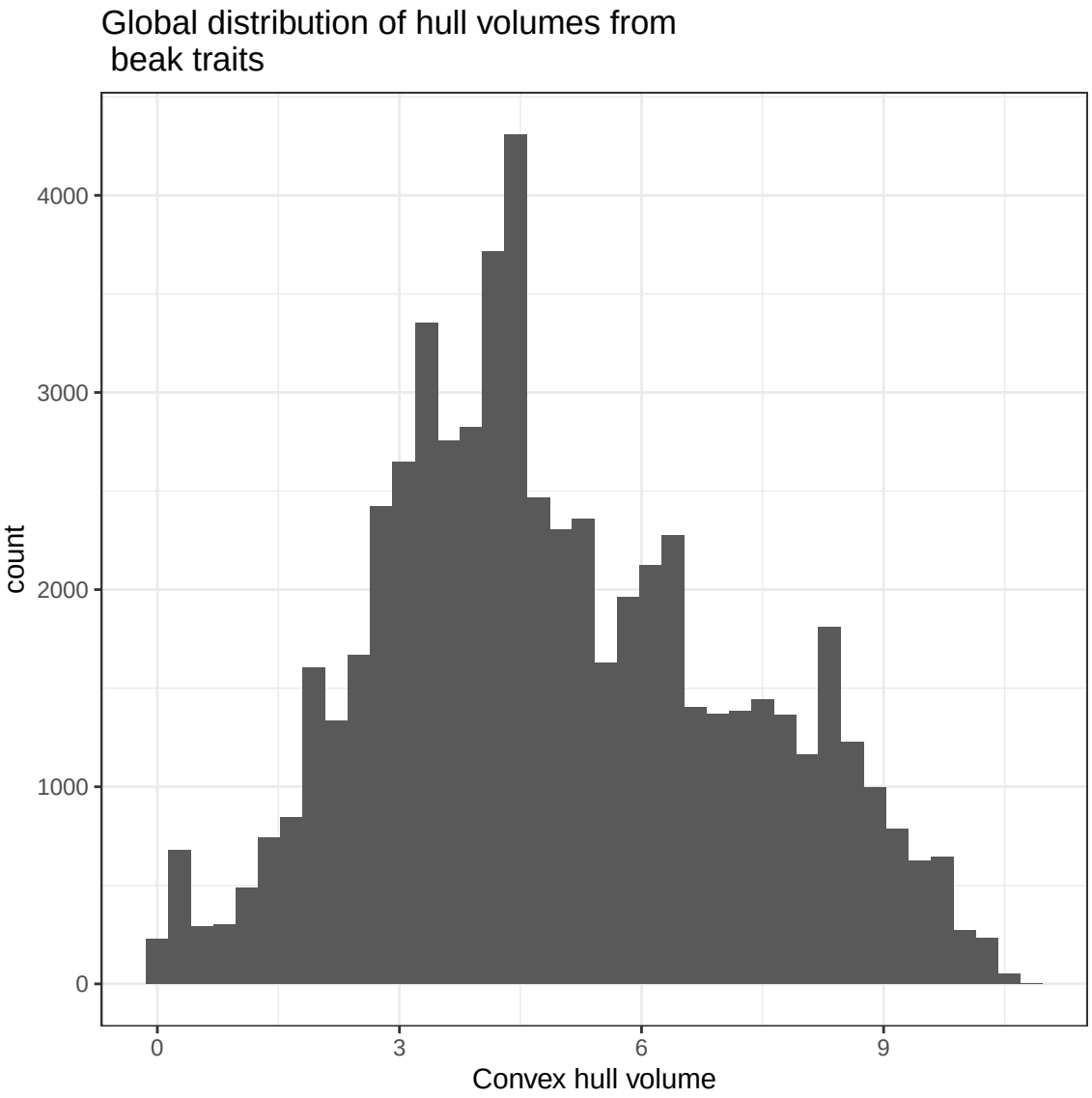
```
## Warning in readRDS("data/global.hull.size.beak.RDS") %>% as.numeric(): NAs
## introduced by coercion
```

Unlike the nearest neighbor scores there are no extreme outliers for the hull size distributions.

```
ggplot(as.data.frame(global.hull.size), aes(x = global.hull.size)) +
  geom_histogram(bins=40)+
  labs(x = "Convex hull volume") +
  theme_bw() +
  ggtitle("Global distribution of hull volumes from\n all morphological traits")
```



```
ggplot(as.data.frame(global.hull.size.beak), aes(x = global.hull.size.beak)) +
  geom_histogram(bins=40)+
  labs(x = "Convex hull volume") +
  theme_bw() +
  ggtitle("Global distribution of hull volumes from\n beak traits")
```



Since there is a much stronger linear relationship between convex hull volume and species diversity we can use a linear color palette for plotting these results.

```
# All traits
plot.dat.hull <- cbind(coords,
                       global.hull.size,
                       species.diversity)

global.map.hull <- ggplot(plot.dat.hull,
                          aes(Var1, Var2,
```

```
                                  color=global.hull.size)) +
  geom_point(shape = 15,
             size = .35) +
  scale_color_paletteer_c("scico::tokyo") +
  guides(color = "none") +
  coord_sf(xlim = c(-180, 189),
           ylim = c(-60, 83.5),
           expand = FALSE) +
  theme_bw() +
  labs(x = "Longitude",
       y = "Latitude") +
  theme(axis.text=element_text(size=14),
        axis.title=element_text(size=14),
        panel.background = element_rect(fill = "gray45"),
        panel.grid.major = element_line(colour = "gray50"))


inset.figure.hull <- ggplot(plot.dat.hull,
                        aes(species.diversity,
                            global.hull.size,
                            color=global.hull.size)) +
  geom_point(size = .5) +
  scale_color_paletteer_c("scico::tokyo") +
  ylim(c(0, 60)) +
  labs(x = "Species diversity",
       y = "Convex hull volume") +
  theme_bw() +
  theme(legend.position = "none")


png("FigureExport/HullSpeciesDiversity.png",
    width=12,
    height = 9,
    units="in",
    res=500)
ggdraw(global.map.hull) +
  draw_plot(plot = inset.figure.hull,
            x = .06,
            y = .29,
            width = .21,
            height = .25)
dev.off()

# Beak traits
plot.dat.hull.beak <- cbind(coords,
                        global.hull.size.beak,
                        species.diversity)

global.map.hull.beak <- ggplot(plot.dat.hull.beak,
                               aes(Var1, Var2,
                                   color=global.hull.size.beak)) +
  geom_point(shape = 15,
             size = .35) +
```

```
  scale_color_paletteer_c("scico::tokyo") +
  guides(color = "none") +
  coord_sf(xlim = c(-180, 189),
           ylim = c(-60, 83.5),
           expand = FALSE) +
  theme_bw() +
  labs(x = "Longitude",
       y = "Latitude") +
  theme(axis.text=element_text(size=14),
        axis.title=element_text(size=14),
        panel.background = element_rect(fill = "gray45"),
        panel.grid.major = element_line(colour = "gray50"))


inset.figure.hull.beak <- ggplot(plot.dat.hull.beak,
                         aes(species.diversity,
                             global.hull.size.beak,
                             color=global.hull.size.beak)) +
  geom_point(size = .5) +
  scale_color_paletteer_c("scico::tokyo") +
  ylim(c(0, 11)) +
  labs(x = "Species diversity",
       y = "Convex hull volume") +
  theme_bw() +
  theme(legend.position = "none")


png("FigureExport/HullBeakSpeciesDiversity.png",
    width=12,
    height = 9,
    units="in",
    res=500)
ggdraw(global.map.hull.beak) +
  draw_plot(plot = inset.figure.hull.beak,
            x = .06,
            y = .29,
            width = .21,
            height = .25)
dev.off()
```

3.2 Generate lists of species in highest diversity areas for each family

To generate these lists we first need to load the spatial data and the R function.

```
# Birdlife International and Natureserve
all.spdf <- readRDS("birdmaps.RDS")

source("scripts/MakeHighestDiversitySpeciesList.R")
```

These analyses will be performed for families with at least five species, so let's generate a vector of the families this

```
fam.table <- table(taxo$BLFamilyLatin)
target.fams <- names(fam.table)[fam.table>=5]
```

Now we can iterate over these families to generate the lists of species names in the highest diversity regions.

```
# Where the results will be stored
maxDiversitySpeciesRes <- vector(mode="list",
                                 length = length(target.fams))
names(maxDiversitySpeciesRes) <- target.fams

# Iterate over target families
for(i in 1:length(target.fams)){
    print(paste(i, "of", length(target.fams)))
    target.species <- taxo$TipLabel[taxo$BLFamilyLatin == target.fams[i]] %>% as.character
    # update synonymy
    for(j in 1:length(target.species)){
        upd.spp <- syn.dat$geo.name[syn.dat$JetzTip == target.species[j]]
        upd.spp <- gsub("_"," ", as.character(upd.spp))
        target.species[j] <- upd.spp
    }

    # subset spatial data
    spdf <- all.spdf[all.spdf$SCINAME %in% target.species,]
  # generate species lists
    maxDiversitySpeciesRes[[i]] <- MakeHighestDiversitySpeciesList(spdf)

}

# save the results
saveRDS(maxDiversitySpeciesRes, file="data/maxDiversitySpeciesRes.RDS")
```

3.3 Compare mean NND scores against random sample

The custom function `compare.nnd` will be used to compare the NND of the target species and a random sample of the same number of species from the same family. This is process is repeated within families if there are identified geographic areas with the same number of maximum species (but with different species composition).

```
source("scripts/compared.nnd.R")

# all morphology
all.morpho.res <- lapply(1:length(maxDiversitySpeciesRes), function(ee){

  fam.res <- lapply(1:length(maxDiversitySpeciesRes[[ee]]), function(ii){
    target.species <- maxDiversitySpeciesRes[[ee]][[ii]]

    # Fix synonyms
    target.species <- gsub(" ", "_", target.species)
    for(jj in 1:length(target.species)){
      new.name <- syn.dat$JetzTip[syn.dat$geo.name==target.species[jj]]
      target.species[jj] <- new.name[!is.na(new.name)]
    }

    family <- taxo$BLFamilyLatin[taxo$TipLabel%in%target.species][1]

    ind.res <- compared.nnd(nnd.matrix = all.nnd,
                            target.species = target.species,
                            family = family,
                            taxo = taxo,
                            num.sim = 1000)

    tibble(family = family,
           family.div = nrow(taxo[taxo$BLFamilyLatin==family,]),
           targetMean = ind.res$targetMean,
           nullMinusTargetMean = mean(ind.res$null.distr) - ind.res$targetMean,
           morphology = "All morphological traits")
  })%>% bind_rows()

}) %>% bind_rows()

# beak traits
beak.morpho.res <- lapply(1:length(maxDiversitySpeciesRes), function(ee){

  fam.res <- lapply(1:length(maxDiversitySpeciesRes[[ee]]), function(ii){
    target.species <- maxDiversitySpeciesRes[[ee]][[ii]]

    # Fix synonyms
    target.species <- gsub(" ", "_", target.species)
    for(jj in 1:length(target.species)){
      new.name <- syn.dat$JetzTip[syn.dat$geo.name==target.species[jj]]
      target.species[jj] <- new.name[!is.na(new.name)]
    }

    family <- taxo$BLFamilyLatin[taxo$TipLabel%in%target.species][1]

    ind.res <- compared.nnd(nnd.matrix = beak.nnd,
                            target.species = target.species,
                            family = family,
                            taxo = taxo,
                            num.sim = 1000)
```

```
    tibble(family = family,
           family.div = nrow(taxo[taxo$BLFamilyLatin==family,]),
           targetMean = ind.res$targetMean,
           nullMinusTargetMean = mean(ind.res$null.distr) - ind.res$targetMean,
           morphology = "Beak traits")
  })%>% bind_rows()

}) %>% bind_rows()

# save
nnd.res <- rbind(all.morpho.res, beak.morpho.res)

write.csv(nnd.res, file="data/nnd.res.csv", row.names = FALSE)
```

3.3.1 Visualize the results

This section plots the relationship between standardized NND scores and clade diversity

```
NNDcladeRichness <- ggplot(nnd.res, aes(x = family.div,
                                        y = targetMean)) +
  geom_point() +
  facet_wrap(~morphology, scales="free") +
  theme_tufte() +
  my.theme.scat +
  geom_rangeframe() +
  scale_x_log10() +
  labs(x = "log(family richness)",
       y = "Mean NND score") +
  geom_smooth(method = "lm")

PackingcladeRichness <- ggplot(nnd.res, aes(x = family.div,
                                        y = nullMinusTargetMean)) +
  geom_point() +
  facet_wrap(~morphology, scales="free") +
  theme_tufte() +
  my.theme.scat +
  geom_rangeframe() +
  scale_x_log10() +
  geom_hline(yintercept = 0, linetype = "dashed") +
  labs(x = "log(family richness)",
       y = "Species packing") +
  geom_smooth(method = "lm")


#png("FigureExport/packingResults.png", width=8, height=8, units="in", res=500)
#gridExtra::grid.arrange(NNDcladeRichness,
#                        PackingcladeRichness,
#                        nrow = 2)
#dev.off()
```

3.3.2 Compare residuals with latitude

Here, we test whether deviation from a 0,0 line in the comparison between family diversity and mean NND scores is explained by geographic information for the families. First, let's load information on species ranges, and summarize at the family level.

```
range.dat <- read.csv("data/RangeData.csv")

range.summarized <- range.dat %>% group_by(Family) %>%
  summarise(meanLatitude = mean(LatitudeCentroid, na.rm=T),
            meanRangeSize = mean(RangeSize, na.rm=T))
colnames(range.summarized)[1] <- "family"
```

Combine these data with the nnd result data and visualize

```
nnd.res.range <- inner_join(nnd.res,
                            range.summarized,
                            by = "family")

nndResidLatitude <- ggplot(nnd.res.range,
                            aes(x = nullMinusTargetMean,
                                y = meanLatitude)) +
  geom_point() +
  facet_wrap(~morphology, scales="free") +
  theme_tufte() +
  my.theme.scat +
  geom_rangeframe() +
  geom_hline(yintercept = 0, linetype = "dashed") +
  labs(x = "Deviation from expectation",
       y = "Mean family latitude") +
  geom_smooth(method = "lm")
```

3.3.3 Path analysis

In this section we explore the relationship between packing in morphospace with family diversity while accounting for these variables co-varying with potentially explanatory variables. First, data on the co-varying factors need to be added to the `data.frame`. Here, data on body size and range size are added.

```
mass.data <- read.csv("data/mass_data.csv")
mass.data$ï..Jetz.species <- gsub(" ","_", mass.data$ï..Jetz.species)

range.size <- readRDS("data/SpeciesRangeSize.RDS")
```

Now these data are added to the existing data, taking the mean value for each family.

```
nnd.res$mass <- as.numeric(NA)
nnd.res$range.size <- as.numeric(NA)

for(kk in 1:nrow(nnd.res)){
  family <- as.character(nnd.res$family[kk])
  tips <- taxo$TipLabel[taxo$BLFamilyLatin==family]
  # Mass data
  nnd.res$mass[kk] <- mass.data$Mass[mass.data$ï..Jetz.species %in% tips] %>% mean
  # Range size
  for(j in 1:length(tips)){
    tips[j] <- syn.dat$geo.name[syn.dat$JetzTip==tips[j]]
  }
  tips <- gsub("_", " ", tips)
  nnd.res$range.size[kk] <- range.size[names(range.size)%in%tips] %>% mean(na.rm=T)
}
```

Next, the mean distance of each species in the dataset to the center of morphological space will be calculated, and family means taken again. These calculations use the Pigot et al. PC data.

```
position.in.morphospace <- function(scores){
  scores <- scores^2
  pos <- scores %>% sum %>% sqrt
  return(pos)
}

spp.distances <- apply(pigot.all[,-1], 1, position.in.morphospace)
names(spp.distances) <- pigot.all$ï..Binomial

spp.distances.beak <- apply(pigot.beak[,-1], 1, position.in.morphospace)
names(spp.distances.beak) <- pigot.beak$ï..Binomial

nnd.res$distanceToCenter <- as.numeric(NA)
for(ee in 1:nrow(nnd.res)){
  family <- as.character(nnd.res$family[ee])
  tips <- taxo$TipLabel[taxo$BLFamilyLatin==family]
  if(nnd.res$morphology[ee]=="All morphological traits"){
    nnd.res$distanceToCenter[ee] <- spp.distances[names(spp.distances)%in%tips] %>%
      mean
  } else {
    nnd.res$distanceToCenter[ee] <- spp.distances.beak[names(spp.distances.beak)%in% tips] %>%
```

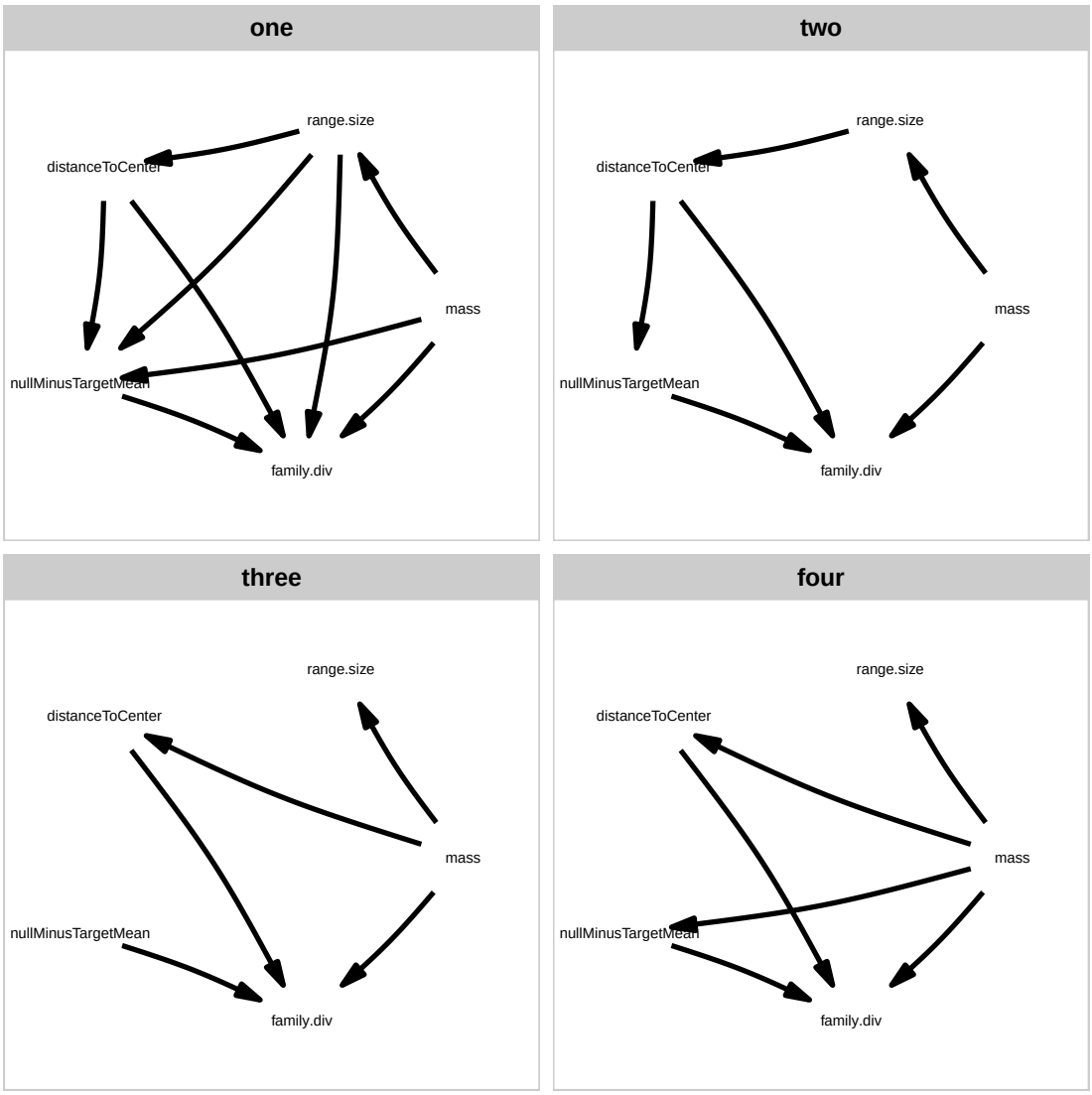
```
      mean
  }
}
```

Next, the models that are to be tested are defined.

```
models <- define_model_set(
  one = c(family.div ~ range.size,
          distanceToCenter ~ range.size,
          nullMinusTargetMean ~ distanceToCenter + range.size + mass),
  two = c(distanceToCenter ~ range.size,
          nullMinusTargetMean ~ distanceToCenter),
  three = c(distanceToCenter ~ mass),
  four = c(distanceToCenter ~ mass,
           nullMinusTargetMean ~ mass),
  .common = c(family.div ~ nullMinusTargetMean +
                mass + distanceToCenter,
              range.size ~ mass)
)
```

Let's take a look what the different models look like

```
plot_model_set(models, text_size = 2)
```



Before the fit of the models can be estimated, the two data types need to be separated.

```
nnd.res.all <- nnd.res[nnd.res$morphology=="All morphological traits",]
nnd.res.beak <- nnd.res[nnd.res$morphology== "Beak traits",]
```

Next, we need to summarize the data where families have repeated entries, and log-transform variables that require it.

```
nnd.res.all <- nnd.res.all %>%
  group_by(family) %>%
  summarize(family.div = log(family.div) %>% mean,
            mass = log(mass) %>% mean,
            nullMinusTargetMean = mean(nullMinusTargetMean),
```

```
            range.size = log(range.size) %>% mean,
            distanceToCenter = log(distanceToCenter) %>% mean) %>% as.data.frame

nnd.res.beak <- nnd.res.beak %>%
  group_by(family) %>%
  summarize(family.div = log(family.div) %>% mean,
            mass = log(mass) %>% mean,
            nullMinusTargetMean = mean(nullMinusTargetMean),
            range.size = log(range.size) %>% mean,
            distanceToCenter = log(distanceToCenter) %>% mean) %>% as.data.frame
```

Finally, we need to load the family-level phylogenetic hypothesis

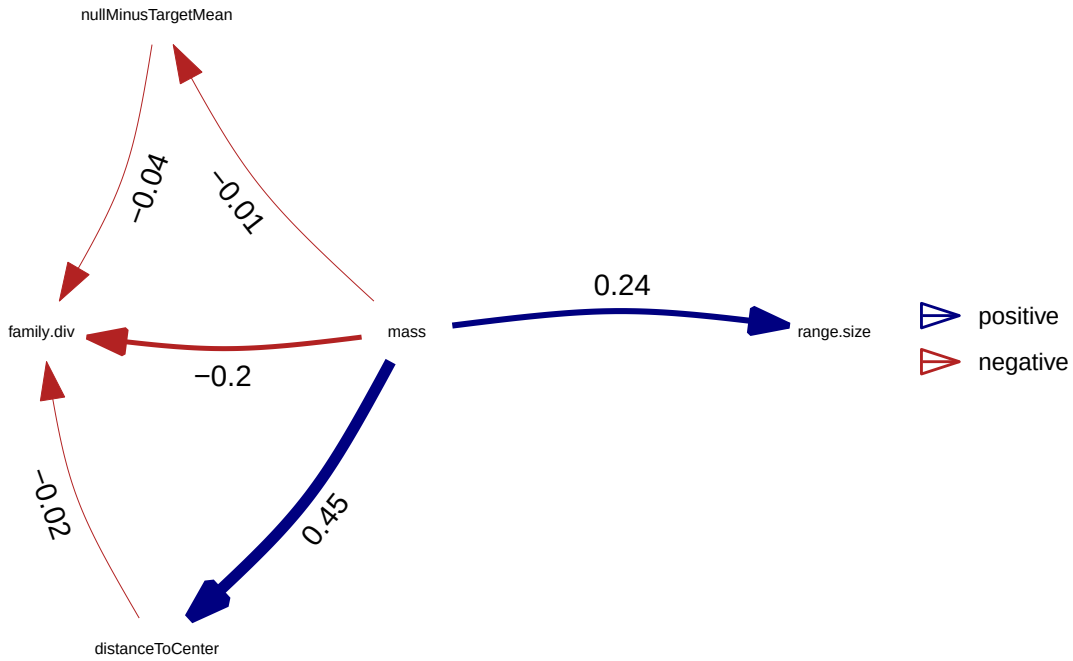
```
fam.phy <- read.tree("data/familyPhylogeny.tre")
```

Now it's possible to estimate the fit of the models to the data.

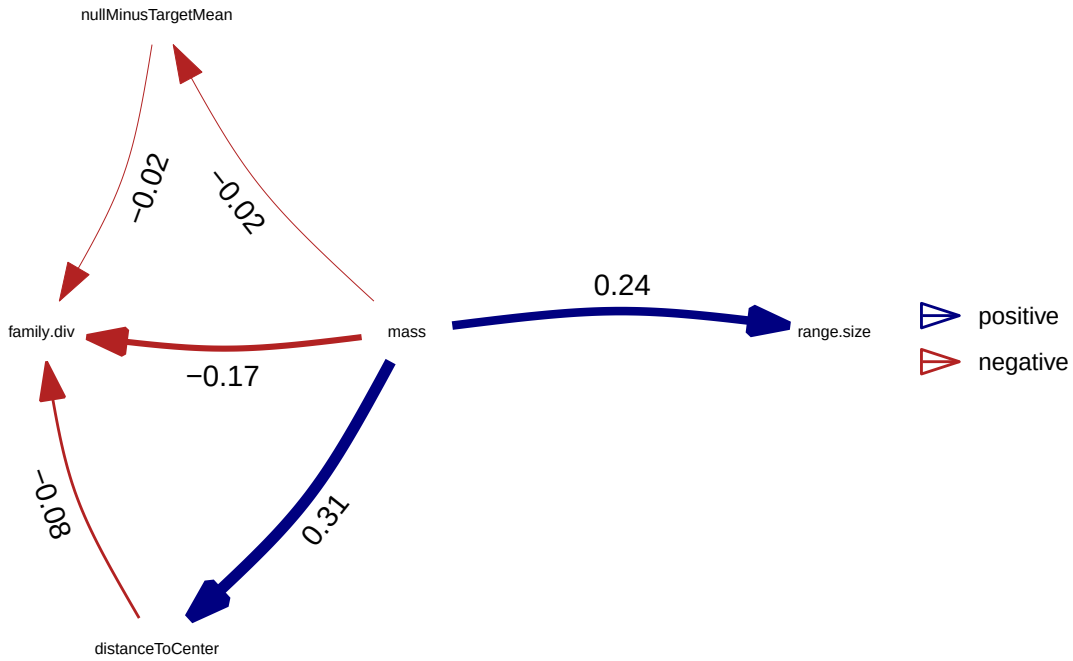
```
rownames(nnd.res.all) <- nnd.res.all$family
rownames(nnd.res.beak) <- nnd.res.beak$family

nnd.data.all <- nnd.res.all[,-1]
nnd.data.beak <- nnd.res.beak[,-1]

result.all <- phylo_path(models,
                     data = nnd.data.all,
                     tree = fam.phy,
                     model = 'lambda')

result.beak <- phylo_path(models,
                     data = nnd.data.beak,
                     tree = fam.phy,
                     model = 'lambda')
```

Perform model averaging and return the final results

```
average_model_full <- average(result.all,
                              avg_method = "full")
plot(average_model_full,
     algorithm = 'mds',
     curvature = 0.1,
     text_size = 2)
```




```
average_model_beak <- average(result.beak,
                              avg_method = "full")
plot(average_model_beak,
     algorithm = 'mds',
     curvature = 0.1,
     text_size = 2)
```


nullMinusTargetMean
−0.02
−0.02
family.div
mass
range.size
0.24
−0.17
−0.08
0.31
distanceToCenter
positive
negative

3.4 Compare hull sizes

This approach is complementary to the NND scores, comparing instead the volume occupied by the species in the maximum diversity area in n-dimensional space to a random sample. This section uses the custom function `compare.hull.size`.

```
source("scripts/compare.hull.size.R")

# all morphology
all.morpho.hull.res <- lapply(1:length(maxDiversitySpeciesRes),
                              function(ee){
  fam.res <- lapply(1:length(maxDiversitySpeciesRes[[ee]]), function(ii){
    target.species <- maxDiversitySpeciesRes[[ee]][[ii]]

    if(length(target.species)>3){

    # Fix synonyms
    target.species <- gsub(" ", "_", target.species)
    for(jj in 1:length(target.species)){
      new.name <- syn.dat$JetzTip[syn.dat$geo.name==target.species[jj]]
      target.species[jj] <- new.name[!is.na(new.name)]
    }

    family <- taxo$BLFamilyLatin[taxo$TipLabel%in%target.species][1]

    ind.res <- compare.hull.size(pc.scores = pigot.all,
                            target.species = target.species,
                            family = family,
                            taxo = taxo,
                            num.sim = 1000)

    tibble(family = family,
           family.div = nrow(taxo[taxo$BLFamilyLatin==family,]),
           targetMean = ind.res$targetVal,
           nullMinusTargetMean = mean(ind.res$null.distr) -
             ind.res$targetVal,
           morphology = "All morphological traits")
    } else {

      tibble(family = taxo$BLFamilyLatin[taxo$TipLabel%in%target.species][1],
             family.div = NA,
             targetMean = NA,
             nullMinusTargetMean = NA)
    }
  })%>% bind_rows()

}) %>% bind_rows()


# beak traits
beak.hull.res <- lapply(1:length(maxDiversitySpeciesRes),
                              function(ee){
  fam.res <- lapply(1:length(maxDiversitySpeciesRes[[ee]]), function(ii){
    target.species <- maxDiversitySpeciesRes[[ee]][[ii]]
```

```
    if(length(target.species)>3){

    # Fix synonyms
    target.species <- gsub(" ", "_", target.species)
    for(jj in 1:length(target.species)){
      new.name <- syn.dat$JetzTip[syn.dat$geo.name==target.species[jj]]
      target.species[jj] <- new.name[!is.na(new.name)]
    }

    family <- taxo$BLFamilyLatin[taxo$TipLabel%in%target.species][1]

    ind.res <- compare.hull.size(pc.scores = pigot.beak,
                            target.species = target.species,
                            family = family,
                            taxo = taxo,
                            num.sim = 1000)

    tibble(family = family,
           family.div = nrow(taxo[taxo$BLFamilyLatin==family,]),
           targetMean = ind.res$targetVal,
           nullMinusTargetMean = mean(ind.res$null.distr) -
             ind.res$targetVal,
           morphology = "Beak traits")
    } else {

      tibble(family = taxo$BLFamilyLatin[taxo$TipLabel%in%target.species][1],
             family.div = NA,
             targetMean = NA,
             nullMinusTargetMean = NA)
    }
  })%>% bind_rows()

}) %>% bind_rows()

# save
hull.res <- rbind(all.morpho.hull.res, beak.hull.res)

write.csv(hull.res, file="data/hull.res.csv", row.names = FALSE)
```

3.4.1 Visualize the results

```
HullCladeRichness <- ggplot(hull.res, aes(x = family.div,
                                          y = targetMean)) +
  geom_point() +
  facet_wrap(~morphology, scales="free") +
  theme_tufte() +
  my.theme.scat +
  geom_rangeframe() +
  scale_x_log10() +
  labs(x = "log(family richness)",
       y = "Hull size") +
```

```
  geom_smooth(method = "lm")

DispersionCladeRichness <- ggplot(hull.res, aes(x = family.div,
                                                y = nullMinusTargetMean)) +
  geom_point() +
  facet_wrap(~morphology, scales="free") +
  theme_tufte() +
  my.theme.scat +
  geom_rangeframe() +
  scale_x_log10() +
  geom_hline(yintercept = 0, linetype = "dashed") +
  labs(x = "log(family richness)",
       y = "Species dispersion") +
  geom_smooth(method = "lm")


png("FigureExport/dispersionResults.png", width=8,
    height=8, units="in", res=500)
gridExtra::grid.arrange(HullCladeRichness,
                        DispersionCladeRichness,
                        nrow = 2)
```

```
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
```

```
dev.off()
```

```
## pdf
##   2
```

```
hull.res.range <- inner_join(hull.res,
                             range.summarized,
                             by = "family")

hullResidLatitude <- ggplot(hull.res.range,
                             aes(x = nullMinusTargetMean,
                                 y = meanLatitude)) +
  geom_point() +
  facet_wrap(~morphology, scales="free") +
  theme_tufte() +
  my.theme.scat +
  geom_rangeframe() +
  geom_hline(yintercept = 0, linetype = "dashed") +
  labs(x = "Deviation from expectation",
       y = "Mean family latitude") +
  geom_smooth(method = "lm")
```

3.4.2 Path analysis

As with the previous path analyses we first need to add in additional information on size and range size.

```
hull.res$mass <- as.numeric(NA)
hull.res$range.size <- as.numeric(NA)

for(kk in 1:nrow(hull.res)){
  family <- as.character(hull.res$family[kk])
  tips <- taxo$TipLabel[taxo$BLFamilyLatin==family]
  # Mass data
  hull.res$mass[kk] <- mass.data$Mass[mass.data$ï..Jetz.species %in% tips] %>% mean
  # Range size
  for(j in 1:length(tips)){
    tips[j] <- syn.dat$geo.name[syn.dat$JetzTip==tips[j]]
  }
  tips <- gsub("_", " ", tips)
  hull.res$range.size[kk] <- range.size[names(range.size)%in%tips] %>% mean(na.rm=T)
}
```

Next, we add in what the convex hull volume for each family is.

```
hull.res$hullVolume <- as.numeric(NA)
for(ee in 1:nrow(hull.res)){
  family <- as.character(hull.res$family[ee])
  tips <- taxo$TipLabel[taxo$BLFamilyLatin==family]
  if(hull.res$morphology[ee]=="All morphological traits"){
    pc.scores <- pigot.all
    pc.subset <- pc.scores[pc.scores$ï..Binomial %in% tips,
                           c("PC1", "PC2", "PC3")]
  } else {
    pc.scores <- pigot.beak
    pc.subset <- pc.scores[pc.scores$ï..Binomial %in% tips,
                           c("Beak_PC1", "Beak_PC2", "Beak_PC3")]
  }
  if(nrow(pc.subset)>3){
    hv <- convhulln(pc.subset[,1:3], "FA")$vol
  } else{
    hv <- NA
  }
  hull.res$hullVolume[ee] <- hv
}
```

We define a similar set of models to the NND scores, but some of the variable names need to be changed.

```
models.hull <- define_model_set(
  one = c(family.div ~ range.size,
          hullVolume ~ range.size,
          nullMinusTargetMean ~ hullVolume + range.size + mass),
  two = c(hullVolume ~ range.size,
          nullMinusTargetMean ~ hullVolume),
  three = c(hullVolume ~ mass),
  four = c(hullVolume ~ mass,
```

```
           nullMinusTargetMean ~ mass),
  .common = c(family.div ~ nullMinusTargetMean +
                mass + hullVolume,
              range.size ~ mass)
)
```

```
hull.res.all <- hull.res[hull.res$morphology=="All morphological traits",]
hull.res.beak <- hull.res[hull.res$morphology== "Beak traits",]
```

Next, as before, we need to summarize the data where families have repeated entries, and log-transform variables where required.

```
hull.res.all <- hull.res.all  %>%
  group_by(family) %>%
  summarize(family.div = log(family.div) %>% mean,
            mass = log(mass) %>% mean,
            nullMinusTargetMean = mean(nullMinusTargetMean),
            range.size = log(range.size) %>% mean,
            hullVolume = log(hullVolume) %>% mean) %>% as.data.frame

hull.res.beak <- hull.res.beak %>%
  group_by(family) %>%
  summarize(family.div = log(family.div) %>% mean,
            mass = log(mass) %>% mean,
            nullMinusTargetMean = mean(nullMinusTargetMean),
            range.size = log(range.size) %>% mean,
            hullVolume = log(hullVolume) %>% mean) %>% as.data.frame
```

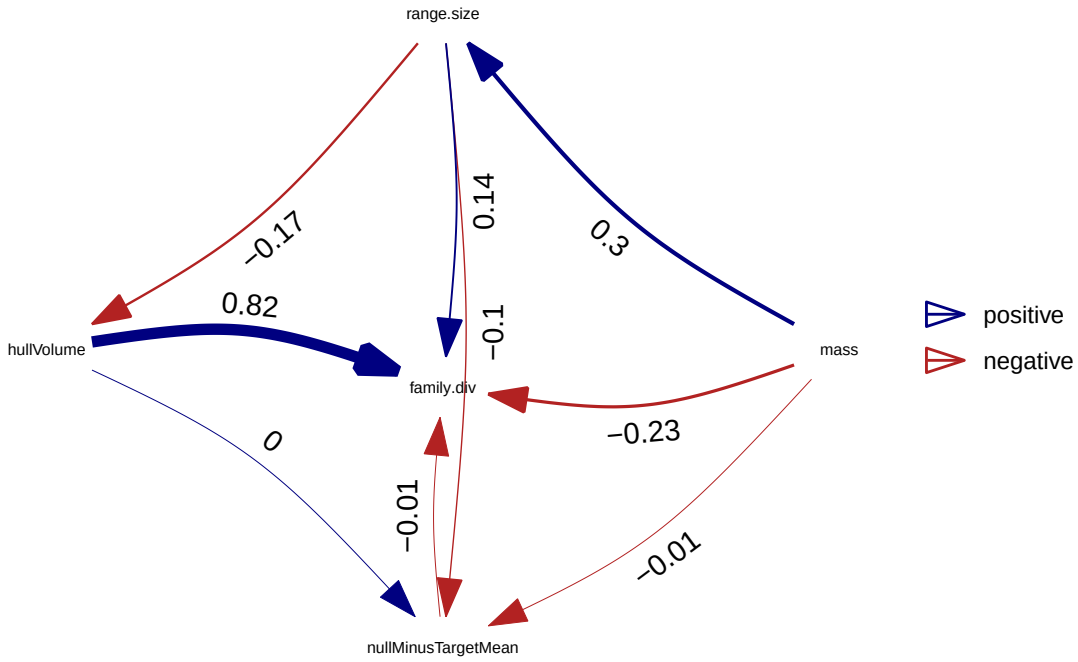
Now it's possible to estimate the fit of the models to the data.

```
rownames(hull.res.all) <- hull.res.all$family
rownames(hull.res.beak) <- hull.res.beak$family

hull.res.all <- hull.res.all[,-1]
hull.res.beak <- hull.res.beak[,-1]

result.all <- phylo_path(models.hull,
                     data = hull.res.all,
                     tree = fam.phy,
                     model = 'lambda')

result.beak <- phylo_path(models.hull,
                     data = hull.res.beak,
                     tree = fam.phy,
                     model = 'lambda')
```

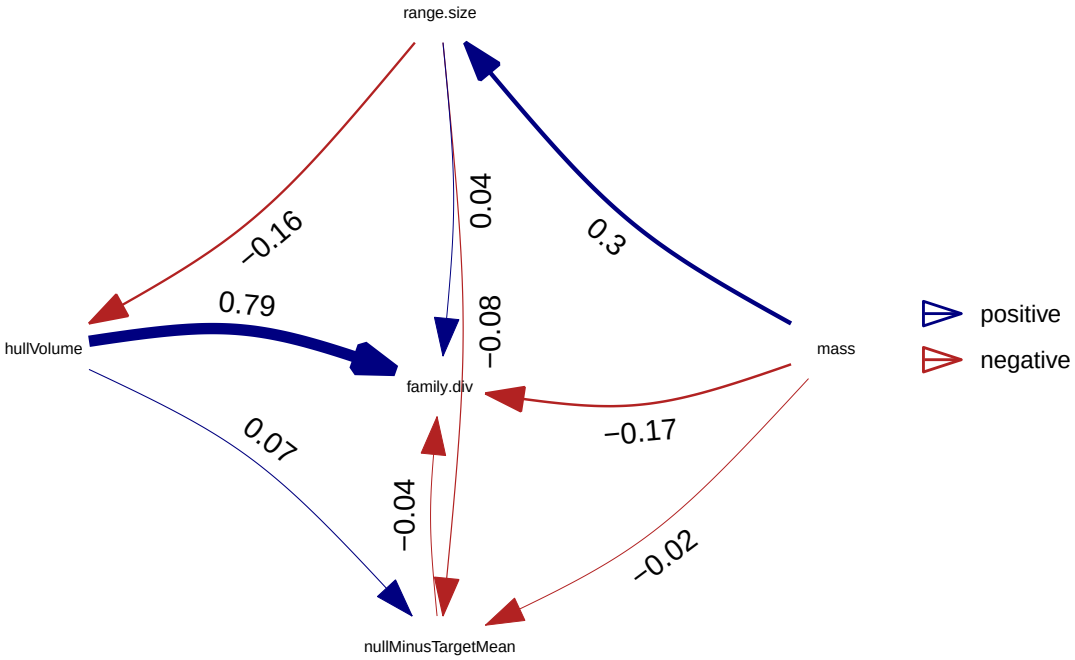
Perform model averaging and return the final results

```
average_model_full <- average(result.all,
                              avg_method = "full")
plot(average_model_full,
     algorithm = 'mds',
```

```
     curvature = 0.1,
     text_size = 2)
```




```
average_model_beak <- average(result.beak,
                              avg_method = "full")
plot(average_model_beak,
     algorithm = 'mds',
     curvature = 0.1,
     text_size = 2)
```


range.size
hullVolume
family.div
mass
nullMinusTargetMean
−0.16
0.79
0.04
−0.08
0.3
0.07
−0.04
−0.17
−0.02
positive
negative

3.5 Compare disparity

Here, we repeat the same procedure as above but calculating the amount of morphological disparity in the highest diversity areas.

```
source("scripts/compare.disparity.R")

# all morphology
all.morpho.disparity.res <- lapply(1:length(maxDiversitySpeciesRes),
                                   function(ee){
  fam.res <- lapply(1:length(maxDiversitySpeciesRes[[ee]]), function(ii){
    target.species <- maxDiversitySpeciesRes[[ee]][[ii]]

    if(length(target.species)>3){

    # Fix synonyms
    target.species <- gsub(" ", "_", target.species)
    for(jj in 1:length(target.species)){
      new.name <- syn.dat$JetzTip[syn.dat$geo.name==target.species[jj]]
      target.species[jj] <- new.name[!is.na(new.name)]
    }

    family <- taxo$BLFamilyLatin[taxo$TipLabel%in%target.species][1]

    ind.res <- compare.disparity(pc.scores = pigot.all,
                                 target.species = target.species,
                                 family = family,
                                 taxo = taxo,
                                 num.sim = 1000)

    tibble(family = family,
           family.div = nrow(taxo[taxo$BLFamilyLatin==family,]),
           targetMean = ind.res$targetVal,
           nullMinusTargetMean = mean(ind.res$null.distr) -
             ind.res$targetVal,
           morphology = "All morphological traits")
    } else {

      tibble(family = taxo$BLFamilyLatin[taxo$TipLabel%in%target.species][1],
             family.div = NA,
             targetMean = NA,
             nullMinusTargetMean = NA)
    }
  })%>% bind_rows()

}) %>% bind_rows()



# beak traits
beak.morpho.disparity.res <- lapply(1:length(maxDiversitySpeciesRes),
                                    function(ee){
  fam.res <- lapply(1:length(maxDiversitySpeciesRes[[ee]]), function(ii){
    target.species <- maxDiversitySpeciesRes[[ee]][[ii]]
```

```
    if(length(target.species)>3){

    # Fix synonyms
    target.species <- gsub(" ", "_", target.species)
    for(jj in 1:length(target.species)){
      new.name <- syn.dat$JetzTip[syn.dat$geo.name==target.species[jj]]
      target.species[jj] <- new.name[!is.na(new.name)]
    }

    family <- taxo$BLFamilyLatin[taxo$TipLabel%in%target.species][1]

    ind.res <- compare.disparity(pc.scores = pigot.beak,
                            target.species = target.species,
                            family = family,
                            taxo = taxo,
                            num.sim = 1000)

    tibble(family = family,
           family.div = nrow(taxo[taxo$BLFamilyLatin==family,]),
           targetMean = ind.res$targetVal,
           nullMinusTargetMean = mean(ind.res$null.distr) -
             ind.res$targetVal,
           morphology = "Beak traits")
    } else {

      tibble(family = taxo$BLFamilyLatin[taxo$TipLabel%in%target.species][1],
             family.div = NA,
             targetMean = NA,
             nullMinusTargetMean = NA)
    }
  })%>% bind_rows()

}) %>% bind_rows()

## Plotting

disparity.res.all <- rbind(all.morpho.disparity.res,
                           beak.morpho.disparity.res)

disparity.res.all <- disparity.res.all[!is.na(disparity.res.all$morphology),]

DisparityCladeRichness <- ggplot(disparity.res.all, aes(x = family.div,
                                                        y = nullMinusTargetMean)) +
  geom_point() +
  facet_wrap(~morphology, scales="free") +
  theme_tufte() +
  my.theme.scat +
  geom_rangeframe() +
  scale_x_log10() +
  geom_hline(yintercept = 0, linetype = "dashed") +
  labs(x = "log(family richness)",
       y = "Null - empirical disparity") +
  geom_smooth(method = "lm")
```

```
png("FigureExport/disparityResults.png", width=6,
    height=5, units="in", res=500)
DisparityCladeRichness
dev.off()
```

4 Genetic analyses

4.1 Compre π between allopatric and sympatric species

4.1.1 Download genetic data and create alignments

These analyses will be performed over the target genes. Complete mtDNA genomes were also downloaded, but this is a much lower sample size. Here the data is downloaded and family-specific alignments created. The `use_history` option is used within `entrez_search` given the large number of sequences downloaded.

```
seq.length <- "500:2000[SLEN]"

cytb.search.term <- "cytb[GENE] OR CYTB[GENE] OR cytB[GENE] AND Aves[ORGN]"

cytb.search.full <- paste(cytb.search.term,
                          seq.length,
                          collapse = " ")

aves_cytb <- entrez_search(db="nuccore",
                          term=cytb.search.full,
                          use_history=TRUE)
```

Now the sequences can be downloaded (in chunks because of the large number)

```
for( seq_start in seq(1,aves_cytb$count,50)){
    recs <- entrez_fetch(db="nuccore", web_history=aves_cytb$web_history,
                         rettype="fasta", retmax=50, retstart=seq_start)
    cat(recs, file="data/aves_cytb.fasta", append=TRUE)
    cat(seq_start+49, "sequences downloaded\r")
}
```

Now repeat the search for *ND2*

```
nd2.search.term <- "ND2[GENE] OR nd2[GENE] AND Aves[ORGN]"

nd2.search.full <- paste(nd2.search.term,
                          seq.length,
                          collapse = " ")

aves_nd2 <- entrez_search(db="nuccore",
                           term=nd2.search.full,
                           use_history=TRUE)
```

Download the *ND2* sequences

```
for( seq_start in seq(1,aves_nd2$count,50)){
    recs <- entrez_fetch(db="nuccore", web_history=aves_nd2$web_history,
                         rettype="fasta", retmax=50, retstart=seq_start)
    cat(recs, file="data/aves_nd2.fasta", append=TRUE)
    cat(seq_start+49, "sequences downloaded\r")
}
```

Search for CO1 sequences

```
co1.search.term <- "CO1[GENE] OR COI[GENE] OR COX1[GENE] Aves[ORGN]"

co1.search.full <- paste(co1.search.term,
                         seq.length,
                         collapse = " ")

aves_co1 <- entrez_search(db="nuccore",
                          term=co1.search.full,
                          use_history=TRUE)
```

Download CO1 sequences

```
for( seq_start in seq(1,aves_co1$count,50)){
    recs <- entrez_fetch(db="nuccore", web_history=aves_co1$web_history,
                         rettype="fasta", retmax=50, retstart=seq_start)
    cat(recs, file="data/aves_co1.fasta", append=TRUE)
    cat(seq_start+49, "sequences downloaded\r")
}
```

Search for mtDNA genomes

```
mt.search.term <- "mitochondrion, complete genome AND Aves[ORGN]"

aves_mt <- entrez_search(db="nuccore",
                          term=mt.search.term,
                          use_history=TRUE)
```

Download

```
for( seq_start in seq(1,aves_mt$count,50)){
    recs <- entrez_fetch(db="nuccore", web_history=aves_mt$web_history,
                         rettype="fasta", retmax=50, retstart=seq_start)
    cat(recs, file="data/aves_mt.fasta", append=TRUE)
    cat(seq_start+49, "sequences downloaded\r")
}
```

Search for RAG1 sequences

```
rag1.search.term <- "RAG1[GENE] OR RAG-1[GENE] OR rag1[GENE] Aves[ORGN]"

rag1.search.full <- paste(rag1.search.term,
                          seq.length,
                          collapse = " ")

aves_rag1 <- entrez_search(db="nuccore",
                          term=rag1.search.full,
                          use_history=TRUE)
```

Download RAG1 sequences

```
for( seq_start in seq(1,aves_rag1$count,50)){
    recs <- entrez_fetch(db="nuccore", web_history=aves_rag1$web_history,
                         rettype="fasta", retmax=50, retstart=seq_start)
    cat(recs, file="data/aves_rag1.fasta", append=TRUE)
    cat(seq_start+49, "sequences downloaded\r")
}
```

Search for MC1R sequences

```
mc1r.search.term <- "MC1R[GENE] OR mc1r[GENE] Aves[ORGN]"

mc1r.search.full <- paste(mc1r.search.term,
                          seq.length,
                          collapse = " ")

aves_mc1r <- entrez_search(db="nuccore",
                           term=mc1r.search.full,
                           use_history=TRUE)
```

Download RAG1 sequences

```
for( seq_start in seq(1,aves_mc1r$count,50)){
    recs <- entrez_fetch(db="nuccore", web_history=aves_mc1r$web_history,
                         rettype="fasta", retmax=50, retstart=seq_start)
    cat(recs, file="data/aves_mc1r.fasta", append=TRUE)
    cat(seq_start+49, "sequences downloaded\r")
}
```

Search for EGR1 sequences

```
egr1.search.term <- "EGR1[GENE] OR EGR-1[GENE] OR egr1[GENE] Aves[ORGN]"

egr1.search.full <- paste(egr1.search.term,
                          seq.length,
                          collapse = " ")

aves_egr1 <- entrez_search(db="nuccore",
                           term=egr1.search.full,
                           use_history=TRUE)
```

Download EGR1 sequences

```
for( seq_start in seq(1,aves_egr1$count,50)){
    recs <- entrez_fetch(db="nuccore", web_history=aves_egr1$web_history,
                         rettype="fasta", retmax=50, retstart=seq_start)
    cat(recs, file="data/aves_egr1.fasta", append=TRUE)
    cat(seq_start+49, "sequences downloaded\r")
}
```

4.1.2 Create family-level alignments

The first stage of creating family-level alignments is parsing the species information in the `.fasta` files. Some of the nuclear genes contain predicted sequences; we want to exclude these as well before proceeding.

```
cytb <- read.fasta("data/aves_cytb.fasta", whole.header = T)
nd2 <- read.fasta("data/aves_nd2.fasta", whole.header = T)
co1 <- read.fasta("data/aves_co1.fasta", whole.header = T)
mt <- read.fasta("data/aves_mt.fasta", whole.header = T)
mc1r <- read.fasta("data/aves_mc1r.fasta", whole.header = T)
rag1 <- read.fasta("data/aves_rag1.fasta", whole.header = T)
egr1 <- read.fasta("data/aves_egr1.fasta", whole.header = T)

mc1r <- mc1r[-grep("PREDICTED", names(mc1r))]
egr1 <- egr1[-grep("PREDICTED", names(egr1))]

parse.fasta.header <- function(x){
  spl.name <- strsplit(x, " ")[[1]][2:3]
  name.out <- paste0(spl.name[1], "_", spl.name[2])
  return(name.out)
}

cytb.species.names <- sapply(names(cytb), parse.fasta.header)
names(cytb.species.names) <- NULL

nd2.species.names <- sapply(names(nd2), parse.fasta.header)
names(nd2.species.names) <- NULL

co1.species.names <- sapply(names(co1), parse.fasta.header)
names(co1.species.names) <- NULL

mt.species.names <- sapply(names(mt), parse.fasta.header)
names(mt.species.names) <- NULL

mc1r.species.names <- sapply(names(mc1r), parse.fasta.header)
names(mc1r.species.names) <- NULL

rag1.species.names <- sapply(names(rag1), parse.fasta.header)
names(rag1.species.names) <- NULL

egr1.species.names <- sapply(names(egr1), parse.fasta.header)
names(egr1) <- NULL
```

A `data.frame` can now be created combining the unique species names with information on their families as well as the number of species sampled. These `data.frame`s will be the basis of all future analyses. Let's begin with the sample size of each species.

```
# Ths species sampled
cytb.species.sampled <- cytb.species.names %>% unique
nd2.species.sampled <- nd2.species.names %>% unique
co1.species.sampled <- co1.species.names %>% unique
mt.species.sampled <- mt.species.names %>% unique
mc1r.species.sampled <- mc1r.species.names %>% unique
rag1.species.sampled <- rag1.species.names %>% unique
```

```
egr1.species.sampled <- egr1.species.names %>% unique

# Create data.frames
cytb.taxonomy <- data.frame(family = as.character(NA),
                            species = cytb.species.sampled,
                            n = NA)
nd2.taxonomy <- data.frame(family = as.character(NA),
                           species = nd2.species.sampled,
                           n = NA)
co1.taxonomy <- data.frame(family = as.character(NA),
                           species = co1.species.sampled,
                           n = NA)
mt.taxonomy <- data.frame(family = as.character(NA),
                          species = mt.species.sampled,
                          n = NA)
mc1r.taxonomy <- data.frame(family = as.character(NA),
                            species = mc1r.species.sampled,
                            n = NA)
rag1.taxonomy <- data.frame(family = as.character(NA),
                            species = rag1.species.sampled,
                            n = NA)
egr1.taxonomy <- data.frame(family = as.character(NA),
                            species = egr1.species.sampled,
                            n = NA)



# Sample size
cytb.n <- lapply(1:length(cytb.species.sampled), function(ee){
  cytb.species.names[cytb.species.names==cytb.species.sampled[ee]] %>%
    length
}) %>% unlist
cytb.taxonomy$n <- cytb.n

nd2.n <- lapply(1:length(nd2.species.sampled), function(ee){
  nd2.species.names[nd2.species.names==nd2.species.sampled[ee]] %>%
    length
}) %>% unlist
nd2.taxonomy$n <- nd2.n

co1.n <- lapply(1:length(co1.species.sampled), function(ee){
  co1.species.names[co1.species.names==co1.species.sampled[ee]] %>%
    length
}) %>% unlist
co1.taxonomy$n <- co1.n

mt.n <- lapply(1:length(mt.species.sampled), function(ee){
  mt.species.names[mt.species.names==mt.species.sampled[ee]] %>%
    length
}) %>% unlist
mt.taxonomy$n <- mt.n

mc1r.n <- lapply(1:length(mc1r.species.sampled), function(ee){
```

```
  mc1r.species.names[mc1r.species.names==mc1r.species.sampled[ee]] %>%
    length
}) %>% unlist
mc1r.taxonomy$n <- mc1r.n

rag1.n <- lapply(1:length(rag1.species.sampled), function(ee){
  rag1.species.names[rag1.species.names==rag1.species.sampled[ee]] %>%
    length
}) %>% unlist
rag1.taxonomy$n <- rag1.n

egr1.n <- lapply(1:length(egr1.species.sampled), function(ee){
  egr1.species.names[egr1.species.names==egr1.species.sampled[ee]] %>%
    length
}) %>% unlist
egr1.taxonomy$n <- egr1.n
```

Now the taxonomic data, which is always fiddly. There's a lot going on in the code below, so let's go over the basic outline. For each gene a loop is used, which checks first whether the species name is in the baseline taxonomic data (from Jetz et al., 2012). If not, it queries whether it is in the synonymy data collated by Crouch (2021). Finally, if not in there it uses `entrez_[x]` to download taxonomic data from NCBI, in conjunction with custom functions for parsing the results (`parse.entrez.taxonomy.R`), to identify additional synonymies. If multiple hits are recovered from the NCBI search, it's probably a subspecies issue in which case the species is excluded. The `synonymy` column will be the equivalent species name in the taxonomic data if that species is not already present.

```
source("scripts/parse.entrez.taxonomy.R")
cytb.taxonomy$Synonymy <- as.character(NA)
nd2.taxonomy$Synonymy <- as.character(NA)
co1.taxonomy$Synonymy <- as.character(NA)
mt.taxonomy$Synonymy <- as.character(NA)
mc1r.taxonomy$Synonymy <- as.character(NA)
rag1.taxonomy$Synonymy <- as.character(NA)
egr1.taxonomy$Synonymy <- as.character(NA)

## cyt b ##
for(j in 1:nrow(cytb.taxonomy)){
  spp <- cytb.taxonomy$species[j]
  in.jetz <- spp %in% taxo$TipLabel
  if(in.jetz==TRUE){
    # Is in base taxonomic data
    cytb.taxonomy$family[j] <- as.character(taxo$BLFamilyLatin[
      taxo$TipLabel==spp])
    # Is in existing synonymy data
  } else if(spp %in% syn.dat$geo.name){
    spp1 <- syn.dat$JetzTip[syn.dat$geo.name==spp]
    spp1 <- spp1[!is.na(spp1)]
    if(spp1 %in% taxo$TipLabel){
          cytb.taxonomy$family[j] <- as.character(taxo$BLFamilyLatin[
      taxo$TipLabel==spp1])
          cytb.taxonomy$Synonymy[j] <- spp1
    }
    # Query NCBI
```

```
  } else {
    entrez.search.res <- entrez_search(db = "taxonomy", term = spp)
    if(length(entrez.search.res$ids)>0){
      entrez.pull <- entrez_fetch(db = "taxonomy",
                                  id=entrez.search.res$ids,
                                  rettype = "xml",
                                  parsed = TRUE)
      synonyms <- find.synonymy(entrez.pull)
      num.synonymies.present <- grep(TRUE,
                                     (synonyms %in% taxo$TipLabel)) %>%
        length
      if(num.synonymies.present == 1){
        syn <- synonyms[synonyms %in% taxo$TipLabel]
        cytb.taxonomy$Synonymy[j] <- syn
        cytb.taxonomy$family[j] <- as.character(taxo$BLFamilyLatin[
          taxo$TipLabel==syn])
      }
    }

  }
}


# nd2
for(j in 1:nrow(nd2.taxonomy)){
  spp <- nd2.taxonomy$species[j]
  in.jetz <- spp %in% taxo$TipLabel
  if(in.jetz==TRUE){
    # Is in base taxonomic data
    nd2.taxonomy$family[j] <- as.character(taxo$BLFamilyLatin[
      taxo$TipLabel==spp])
    # Is in existing synonymy data
  } else if(spp %in% syn.dat$geo.name){
    spp1 <- syn.dat$JetzTip[syn.dat$geo.name==spp]
    spp1 <- spp1[!is.na(spp1)]
    if(spp1 %in% taxo$TipLabel){
          nd2.taxonomy$family[j] <- as.character(taxo$BLFamilyLatin[
      taxo$TipLabel==spp1])
          nd2.taxonomy$Synonymy[j] <- spp1
    }
    # Query NCBI
  } else {
    entrez.search.res <- entrez_search(db = "taxonomy", term = spp)
    if(length(entrez.search.res$ids)>0){
      entrez.pull <- entrez_fetch(db = "taxonomy",
                                  id=entrez.search.res$ids,
                                  rettype = "xml",
                                  parsed = TRUE)
      synonyms <- find.synonymy(entrez.pull)
      num.synonymies.present <- grep(TRUE,
                                     (synonyms %in% taxo$TipLabel)) %>%
        length
      if(num.synonymies.present == 1){
```

```
        syn <- synonyms[synonyms %in% taxo$TipLabel]
        nd2.taxonomy$Synonymy[j] <- syn
        nd2.taxonomy$family[j] <- as.character(taxo$BLFamilyLatin[
          taxo$TipLabel==syn])
      }
    }

  }
}


# co1
for(j in 1:nrow(co1.taxonomy)){
  spp <- co1.taxonomy$species[j]
  in.jetz <- spp %in% taxo$TipLabel
  if(in.jetz==TRUE){
    # Is in base taxonomic data
    co1.taxonomy$family[j] <- as.character(taxo$BLFamilyLatin[
      taxo$TipLabel==spp])
    # Is in existing synonymy data
  } else if(spp %in% syn.dat$geo.name){
    spp1 <- syn.dat$JetzTip[syn.dat$geo.name==spp]
    spp1 <- spp1[!is.na(spp1)]
    if(spp1 %in% taxo$TipLabel){
          co1.taxonomy$family[j] <- as.character(taxo$BLFamilyLatin[
      taxo$TipLabel==spp1])
          co1.taxonomy$Synonymy[j] <- spp1
    }
    # Query NCBI
  } else {
    entrez.search.res <- entrez_search(db = "taxonomy", term = spp)
    if(length(entrez.search.res$ids)>0){
      entrez.pull <- entrez_fetch(db = "taxonomy",
                                  id=entrez.search.res$ids,
                                  rettype = "xml",
                                  parsed = TRUE)
      synonyms <- find.synonymy(entrez.pull)
      num.synonymies.present <- grep(TRUE,
                                     (synonyms %in% taxo$TipLabel)) %>%
        length
      if(num.synonymies.present == 1){
        syn <- synonyms[synonyms %in% taxo$TipLabel]
        co1.taxonomy$Synonymy[j] <- syn
        co1.taxonomy$family[j] <- as.character(taxo$BLFamilyLatin[
          taxo$TipLabel==syn])
      }
    }

  }
}


# MT
```

```
for(j in 1:nrow(mt.taxonomy)){
  spp <- mt.taxonomy$species[j]
  in.jetz <- spp %in% taxo$TipLabel
  if(in.jetz==TRUE){
    # Is in base taxonomic data
    mt.taxonomy$family[j] <- as.character(taxo$BLFamilyLatin[
      taxo$TipLabel==spp])
    # Is in existing synonymy data
  } else if(spp %in% syn.dat$geo.name){
    spp1 <- syn.dat$JetzTip[syn.dat$geo.name==spp]
    spp1 <- spp1[!is.na(spp1)]
    if(spp1 %in% taxo$TipLabel){
          mt.taxonomy$family[j] <- as.character(taxo$BLFamilyLatin[
      taxo$TipLabel==spp1])
          mt.taxonomy$Synonymy[j] <- spp1
    }
    # Query NCBI
  } else {
    entrez.search.res <- entrez_search(db = "taxonomy", term = spp)
    if(length(entrez.search.res$ids)>0){
      entrez.pull <- entrez_fetch(db = "taxonomy",
                                  id=entrez.search.res$ids,
                                  rettype = "xml",
                                  parsed = TRUE)
      synonyms <- find.synonymy(entrez.pull)
      num.synonymies.present <- grep(TRUE,
                                     (synonyms %in% taxo$TipLabel)) %>%
        length
      if(num.synonymies.present == 1){
        syn <- synonyms[synonyms %in% taxo$TipLabel]
        mt.taxonomy$Synonymy[j] <- syn
        mt.taxonomy$family[j] <- as.character(taxo$BLFamilyLatin[
          taxo$TipLabel==syn])
      }
    }

  }
}


# mc1r
for(j in 1:nrow(mc1r.taxonomy)){
  spp <- mc1r.taxonomy$species[j]
  in.jetz <- spp %in% taxo$TipLabel
  if(in.jetz==TRUE){
    # Is in base taxonomic data
    mc1r.taxonomy$family[j] <- as.character(taxo$BLFamilyLatin[
      taxo$TipLabel==spp])
    # Is in existing synonymy data
  } else if(spp %in% syn.dat$geo.name){
    spp1 <- syn.dat$JetzTip[syn.dat$geo.name==spp]
    spp1 <- spp1[!is.na(spp1)]
    if(spp1 %in% taxo$TipLabel){
```

```
          mc1r.taxonomy$family[j] <- as.character(taxo$BLFamilyLatin[
      taxo$TipLabel==spp1])
          mc1r.taxonomy$Synonymy[j] <- spp1
    }
    # Query NCBI
  } else {
    entrez.search.res <- entrez_search(db = "taxonomy", term = spp)
    if(length(entrez.search.res$ids)>0){
      entrez.pull <- entrez_fetch(db = "taxonomy",
                                  id=entrez.search.res$ids,
                                  rettype = "xml",
                                  parsed = TRUE)
      synonyms <- find.synonymy(entrez.pull)
      num.synonymies.present <- grep(TRUE,
                                     (synonyms %in% taxo$TipLabel)) %>%
        length
      if(num.synonymies.present == 1){
        syn <- synonyms[synonyms %in% taxo$TipLabel]
        mc1r.taxonomy$Synonymy[j] <- syn
        mc1r.taxonomy$family[j] <- as.character(taxo$BLFamilyLatin[
          taxo$TipLabel==syn])
      }
    }

  }
}
# rag1
for(j in 1:nrow(rag1.taxonomy)){
  spp <- rag1.taxonomy$species[j]
  in.jetz <- spp %in% taxo$TipLabel
  if(in.jetz==TRUE){
    # Is in base taxonomic data
    rag1.taxonomy$family[j] <- as.character(taxo$BLFamilyLatin[
      taxo$TipLabel==spp])
    # Is in existing synonymy data
  } else if(spp %in% syn.dat$geo.name){
    spp1 <- syn.dat$JetzTip[syn.dat$geo.name==spp]
    spp1 <- spp1[!is.na(spp1)]
    if(spp1 %in% taxo$TipLabel){
          rag1.taxonomy$family[j] <- as.character(taxo$BLFamilyLatin[
      taxo$TipLabel==spp1])
          rag1.taxonomy$Synonymy[j] <- spp1
    }
    # Query NCBI
  } else {
    entrez.search.res <- entrez_search(db = "taxonomy", term = spp)
    if(length(entrez.search.res$ids)>0){
      entrez.pull <- entrez_fetch(db = "taxonomy",
                                  id=entrez.search.res$ids,
                                  rettype = "xml",
                                  parsed = TRUE)
      synonyms <- find.synonymy(entrez.pull)
      num.synonymies.present <- grep(TRUE,
```

```
                                    (synonyms %in% taxo$TipLabel)) %>%
        length
      if(num.synonymies.present == 1){
        syn <- synonyms[synonyms %in% taxo$TipLabel]
        rag1.taxonomy$Synonymy[j] <- syn
        rag1.taxonomy$family[j] <- as.character(taxo$BLFamilyLatin[
          taxo$TipLabel==syn])
      }
    }

  }
}

# egr1
for(j in 1:nrow(egr1.taxonomy)){
  spp <- egr1.taxonomy$species[j]
  in.jetz <- spp %in% taxo$TipLabel
  if(in.jetz==TRUE){
    # Is in base taxonomic data
    egr1.taxonomy$family[j] <- as.character(taxo$BLFamilyLatin[
      taxo$TipLabel==spp])
    # Is in existing synonymy data
  } else if(spp %in% syn.dat$geo.name){
    spp1 <- syn.dat$JetzTip[syn.dat$geo.name==spp]
    spp1 <- spp1[!is.na(spp1)]
    if(spp1 %in% taxo$TipLabel){
          egr1.taxonomy$family[j] <- as.character(taxo$BLFamilyLatin[
      taxo$TipLabel==spp1])
          egr1.taxonomy$Synonymy[j] <- spp1
    }
    # Query NCBI
  } else {
    entrez.search.res <- entrez_search(db = "taxonomy", term = spp)
    if(length(entrez.search.res$ids)>0){
      entrez.pull <- entrez_fetch(db = "taxonomy",
                                  id=entrez.search.res$ids,
                                  rettype = "xml",
                                  parsed = TRUE)
      synonyms <- find.synonymy(entrez.pull)
      num.synonymies.present <- grep(TRUE,
                                     (synonyms %in% taxo$TipLabel)) %>%
        length
      if(num.synonymies.present == 1){
        syn <- synonyms[synonyms %in% taxo$TipLabel]
        egr1.taxonomy$Synonymy[j] <- syn
        egr1.taxonomy$family[j] <- as.character(taxo$BLFamilyLatin[
          taxo$TipLabel==syn])
      }
    }

  }
}
```

Using this higher level taxonomic data, we can now create the alignments.

```
source("scripts/make.family.alignment.R")

## cyt b ##
cytb.families <- cytb.taxonomy$family %>% unique
cytb.families <- cytb.families[!is.na(cytb.families)]

for(p in 1:length(cytb.families)){
  fam <- cytb.families[p]

  # Subset the full alignment for just that family
  fam.align <- make.family.alignment(family = fam,
                                     gene.taxonomy = cytb.taxonomy,
                                     alignment = cytb)

  # Where the alignment will be saved
  dir.string <- paste0("data/familyAlignments/cytb/",
                       fam,
                       ".fasta")

  # export the alignment
  write.fasta(sequences = fam.align,
              names = names(fam.align),
              file.out = dir.string)
}

## nd2 ##
nd2.families <- nd2.taxonomy$family %>% unique
nd2.families <- nd2.families[!is.na(nd2.families)]

for(p in 1:length(nd2.families)){
  fam <- nd2.families[p]
  fam.align <- make.family.alignment(family = fam,
                                     gene.taxonomy = nd2.taxonomy,
                                     alignment = nd2)
  dir.string <- paste0("data/familyAlignments/nd2/",
                       fam,
                       ".fasta")
  write.fasta(sequences = fam.align,
              names = names(fam.align),
              file.out = dir.string)
}

## co1 ##
co1.families <- co1.taxonomy$family %>% unique
co1.families <- co1.families[!is.na(co1.families)]

for(p in 1:length(co1.families)){
  fam <- co1.families[p]
  fam.align <- make.family.alignment(family = fam,
                                     gene.taxonomy = co1.taxonomy,
                                     alignment = co1)
  dir.string <- paste0("data/familyAlignments/co1/",
```

```
                        fam,
                        ".fasta")
  write.fasta(sequences = fam.align,
              names = names(fam.align),
              file.out = dir.string)
}


## mt ##
mt.families <- mt.taxonomy$family %>% unique
mt.families <- mt.families[!is.na(mt.families)]

for(p in 1:length(mt.families)){
  fam <- mt.families[p]
  fam.align <- make.family.alignment(family = fam,
                                     gene.taxonomy = mt.taxonomy,
                                     alignment = mt)
  dir.string <- paste0("data/familyAlignments/mt/",
                        fam,
                        ".fasta")
  write.fasta(sequences = fam.align,
              names = names(fam.align),
              file.out = dir.string)
}


## mc1r ##
mc1r.families <- mc1r.taxonomy$family %>% unique
mc1r.families <- mc1r.families[!is.na(mc1r.families)]

for(p in 1:length(mc1r.families)){
  fam <- mc1r.families[p]
  fam.align <- make.family.alignment(family = fam,
                                     gene.taxonomy = mc1r.taxonomy,
                                     alignment = mc1r)
  dir.string <- paste0("data/familyAlignments/mc1r/",
                        fam,
                        ".fasta")
  write.fasta(sequences = fam.align,
              names = names(fam.align),
              file.out = dir.string)
}


## rag1 ##
rag1.families <- rag1.taxonomy$family %>% unique
rag1.families <- rag1.families[!is.na(rag1.families)]

for(p in 1:length(rag1.families)){
  fam <- rag1.families[p]
  fam.align <- make.family.alignment(family = fam,
                                     gene.taxonomy = rag1.taxonomy,
                                     alignment = rag1)
```

```
  dir.string <- paste0("data/familyAlignments/rag1/",
                       fam,
                       ".fasta")
  write.fasta(sequences = fam.align,
              names = names(fam.align),
              file.out = dir.string)
}


## egr1 ##
egr1.families <- egr1.taxonomy$family %>% unique
egr1.families <- egr1.families[!is.na(egr1.families)]

for(p in 1:length(egr1.families)){
  fam <- egr1.families[p]
  fam.align <- make.family.alignment(family = fam,
                                     gene.taxonomy = egr1.taxonomy,
                                     alignment = egr1)
  dir.string <- paste0("data/familyAlignments/egr1/",
                       fam,
                       ".fasta")
  write.fasta(sequences = fam.align,
              names = names(fam.align),
              file.out = dir.string)
}
```

Each of these family level sequence files now needs to be aligned. Since we probably don't want to call MAFFT hundreds of times individually, we can instead generate a `.sh` file which will have all of the MAFFT calls contained within it, meaning just one file needs to be executed. Let's generate those files using the function `make.mafft.sh`, which just needs to be pointed at a directory, and will automatically generate an `.sh` file which will align all of the `.fasta` files contained within the directory.

```
source("scripts/make.mafft.sh.R")

## cytb ##
make.mafft.sh(dir = "data/familyAlignments/cytb/",
              out.name = "cytbMAFFT")

## nd2 ##
make.mafft.sh(dir = "data/familyAlignments/nd2/",
              out.name = "nd2MAFFT")

## co1 ##
make.mafft.sh(dir = "data/familyAlignments/co1/",
              out.name = "co1MAFFT")

## mt ##
make.mafft.sh(dir = "data/familyAlignments/mt/",
              out.name = "mtMAFFT")

## mc1r ##
make.mafft.sh(dir = "data/familyAlignments/mc1r/",
              out.name = "mc1rMAFFT")
```

```
## rag1 ##
make.mafft.sh(dir = "data/familyAlignments/rag1/",
              out.name = "rag1MAFFT")

## egr1 ##
make.mafft.sh(dir = "data/familyAlignments/egr1/",
              out.name = "egr1MAFFT")
```

If you run a similar workflow and can't execute the `.sh` file, you may need to run (as an example):

```
chmod +x cytbMAFFT.sh
```

to make the file executable, before running

```
./cytbMAFFT.sh
```

Make sure you're in the right directory! These will take a little while to run here given the large number of families.

4.1.3 Calculate species π scores

Species-specific π scores will be calculated using the function `get.species.pi.scores` which again just requires a directory specification, and returns the species π scores from all the files in that directory carrying the abbreviation `*Aligned.fasta`.

```
source("scripts/get.species.pi.scores.R")
# cytb
cytb.pi <- get.species.pi.scores("data/familyAlignments/cytb/")

# nd2
nd2.pi <- get.species.pi.scores("data/familyAlignments/nd2/")

# co1
co1.pi <- get.species.pi.scores("data/familyAlignments/co1/")

# mt
mt.pi <- get.species.pi.scores("data/familyAlignments/mt/")

# mc1r
mc1r.pi <- get.species.pi.scores("data/familyAlignments/mc1r/")

# rag1
rag1.pi <- get.species.pi.scores("data/familyAlignments/rag1/")

# egr1
egr1.pi <- get.species.pi.scores("data/familyAlignments/egr1/")
```

These results can now be combined with the corresponding "taxonomy" files

```
names(cytb.pi) <- gsub(" ","_", names(cytb.pi))
names(nd2.pi) <- gsub(" ","_", names(nd2.pi))
names(co1.pi) <- gsub(" ","_", names(co1.pi))
names(mt.pi) <- gsub(" ", "_", names(mt.pi))
names(mc1r.pi) <- gsub(" ", "_", names(mc1r.pi))
names(rag1.pi) <- gsub(" ", "_", names(rag1.pi))
names(egr1.pi) <- gsub(" ", "_", names(egr1.pi))

cytb.taxonomy$pi <- as.numeric(NA)
nd2.taxonomy$pi <- as.numeric(NA)
co1.taxonomy$pi <- as.numeric(NA)
mt.taxonomy$pi <- as.numeric(NA)
mc1r.taxonomy$pi <- as.numeric(NA)
rag1.taxonomy$pi <- as.numeric(NA)
egr1.taxonomy$pi <- as.numeric(NA)

for(j in 1:length(cytb.pi)){
  cytb.taxonomy$pi[cytb.taxonomy$species==names(cytb.pi)[j]] <- cytb.pi[j]
}

for(j in 1:length(nd2.pi)){
  nd2.taxonomy$pi[nd2.taxonomy$species==names(nd2.pi)[j]] <- nd2.pi[j]
}
```

```
for(j in 1:length(co1.pi)){
  co1.taxonomy$pi[co1.taxonomy$species==names(co1.pi)[j]] <- co1.pi[j]
}

for(j in 1:length(mt.pi)){
  mt.taxonomy$pi[mt.taxonomy$species==names(mt.pi)[j]] <- mt.pi[j]
}

for(j in 1:length(mc1r.pi)){
  mc1r.taxonomy$pi[mc1r.taxonomy$species==names(mc1r.pi)[j]] <- mc1r.pi[j]
}

for(j in 1:length(rag1.pi)){
  rag1.taxonomy$pi[rag1.taxonomy$species==names(rag1.pi)[j]] <- rag1.pi[j]
}

for(j in 1:length(egr1.pi)){
  egr1.taxonomy$pi[egr1.taxonomy$species==names(egr1.pi)[j]] <- egr1.pi[j]
}
```

4.1.4 Integrate allopatry assignments

The sister species assignments of allopatry and sympatry come from a previous study. These data will be collated with the respective "taxonomy files". The allopatry definitions are at the taxonomic level of sister taxa and at the family. First, the data are loaded.

```
### Sister species definitions of allopatry
sister.geo.results <- readRDS("data/geo.results.bird.RDS")

source("scripts/make.terminal.branch.ancestral.state.mod.R")

# species-specific scores
sister.spp.state <- lapply(sister.geo.results,
                           make.terminal.branch.ancestral.state) %>%
  unlist

### Family level definitions of allopatry
family.geo.results <- readRDS("data/bird_terminal_branch_states_2b.RDS") %>%
  bind_rows()
```

Now for some long loops, but essentially they just find the name of each species in the "taxonomy" file and add in the state information.

```
### Species-level allopatry definitions

# cytb
cytb.sister.states <- rep(as.character(NA), nrow(cytb.taxonomy))
for(ee in 1:nrow(cytb.taxonomy)){
  spp <- cytb.taxonomy$species[ee] %>% as.character
  presence.test <- spp %in% names(sister.spp.state)

  if(presence.test==TRUE){
    state <- sister.spp.state[names(sister.spp.state)==spp]
  } else if(cytb.taxonomy$Synonymy[ee]%in% names(sister.spp.state)){
    spp.syn <- cytb.taxonomy$Synonymy[ee]
    state <- sister.spp.state[names(sister.spp.state)==spp.syn]
  } else {
    state <- NA
  }
  cytb.sister.states[ee] <- state
}
cytb.taxonomy$GeographyStateSister <- cytb.sister.states



# nd2
nd2.sister.states <- rep(as.character(NA), nrow(nd2.taxonomy))
for(ee in 1:nrow(nd2.taxonomy)){
  spp <- nd2.taxonomy$species[ee] %>% as.character
  presence.test <- spp %in% names(sister.spp.state)

  if(presence.test==TRUE){
    state <- sister.spp.state[names(sister.spp.state)==spp]
  } else if(nd2.taxonomy$Synonymy[ee]%in% names(sister.spp.state)){
```

```
    spp.syn <- nd2.taxonomy$Synonymy[ee]
    state <- sister.spp.state[names(sister.spp.state)==spp.syn]
  } else {
    state <- NA
  }
  nd2.sister.states[ee] <- state
}
nd2.taxonomy$GeographyStateSister <- nd2.sister.states


# co1
co1.sister.states <- rep(as.character(NA), nrow(co1.taxonomy))
for(ee in 1:nrow(co1.taxonomy)){
  spp <- co1.taxonomy$species[ee] %>% as.character
  presence.test <- spp %in% names(sister.spp.state)

  if(presence.test==TRUE){
    state <- sister.spp.state[names(sister.spp.state)==spp]
  } else if(co1.taxonomy$Synonymy[ee]%in% names(sister.spp.state)){
    spp.syn <- co1.taxonomy$Synonymy[ee]
    state <- sister.spp.state[names(sister.spp.state)==spp.syn]
  } else {
    state <- NA
  }
  co1.sister.states[ee] <- state
}
co1.taxonomy$GeographyStateSister <- co1.sister.states


# mt
mt.sister.states <- rep(as.character(NA), nrow(mt.taxonomy))
for(ee in 1:nrow(mt.taxonomy)){
  spp <- mt.taxonomy$species[ee] %>% as.character
  presence.test <- spp %in% names(sister.spp.state)

  if(presence.test==TRUE){
    state <- sister.spp.state[names(sister.spp.state)==spp]
  } else if(mt.taxonomy$Synonymy[ee]%in% names(sister.spp.state)){
    spp.syn <- mt.taxonomy$Synonymy[ee]
    state <- sister.spp.state[names(sister.spp.state)==spp.syn]
  } else {
    state <- NA
  }
  mt.sister.states[ee] <- state
}
mt.taxonomy$GeographyStateSister <- mt.sister.states


# mc1r
mc1r.sister.states <- rep(as.character(NA), nrow(mc1r.taxonomy))
for(ee in 1:nrow(mc1r.taxonomy)){
  spp <- mc1r.taxonomy$species[ee] %>% as.character
  presence.test <- spp %in% names(sister.spp.state)
```

```
  if(presence.test==TRUE){
    state <- sister.spp.state[names(sister.spp.state)==spp]
  } else if(mc1r.taxonomy$Synonymy[ee]%in% names(sister.spp.state)){
    spp.syn <- mc1r.taxonomy$Synonymy[ee]
    state <- sister.spp.state[names(sister.spp.state)==spp.syn]
  } else {
    state <- NA
  }
  mc1r.sister.states[ee] <- state
}
mc1r.taxonomy$GeographyStateSister <- mc1r.sister.states


# rag1
rag1.sister.states <- rep(as.character(NA), nrow(rag1.taxonomy))
for(ee in 1:nrow(rag1.taxonomy)){
  spp <- rag1.taxonomy$species[ee] %>% as.character
  presence.test <- spp %in% names(sister.spp.state)

  if(presence.test==TRUE){
    state <- sister.spp.state[names(sister.spp.state)==spp]
  } else if(rag1.taxonomy$Synonymy[ee]%in% names(sister.spp.state)){
    spp.syn <- rag1.taxonomy$Synonymy[ee]
    state <- sister.spp.state[names(sister.spp.state)==spp.syn]
  } else {
    state <- NA
  }
  rag1.sister.states[ee] <- state
}
rag1.taxonomy$GeographyStateSister <- rag1.sister.states

# egr1
egr1.sister.states <- rep(as.character(NA), nrow(egr1.taxonomy))
for(ee in 1:nrow(egr1.taxonomy)){
  spp <- egr1.taxonomy$species[ee] %>% as.character
  presence.test <- spp %in% names(sister.spp.state)

  if(presence.test==TRUE){
    state <- sister.spp.state[names(sister.spp.state)==spp]
  } else if(egr1.taxonomy$Synonymy[ee]%in% names(sister.spp.state)){
    spp.syn <- egr1.taxonomy$Synonymy[ee]
    state <- sister.spp.state[names(sister.spp.state)==spp.syn]
  } else {
    state <- NA
  }
  egr1.sister.states[ee] <- state
}
egr1.taxonomy$GeographyStateSister <- egr1.sister.states


### Family level definitions of allopatry

# cytb
```

```
cytb.family.states <- rep(as.numeric(NA), nrow(cytb.taxonomy))
for(ee in 1:nrow(cytb.taxonomy)){
  spp <- cytb.taxonomy$species[ee] %>% as.character
  spp.sub <- gsub("_", " ", spp)
  presence.test <- spp.sub %in% family.geo.results$spp
  if(presence.test==TRUE){
    state <- family.geo.results$allopatry[
      family.geo.results$spp == spp.sub
    ]
  } else if(cytb.taxonomy$Synonymy[ee] %in% family.geo.results$spp){
    spp.syn <- cytb.taxonomy$Synonymy[ee]
    state <- family.geo.results$allopatry[
      family.geo.results$spp == spp.syn
    ]
  } else {
    state <- NA
  }
  cytb.family.states[ee] <- state
}

cytb.family.states[cytb.family.states=="0"] <- "Sympatry"
cytb.family.states[cytb.family.states=="1"] <- "Allopatry"

cytb.taxonomy$GeographyStateFamily <- as.character(cytb.family.states)

# nd2
nd2.family.states <- rep(as.numeric(NA), nrow(nd2.taxonomy))
for(ee in 1:nrow(nd2.taxonomy)){
  spp <- nd2.taxonomy$species[ee] %>% as.character
  spp.sub <- gsub("_", " ", spp)
  presence.test <- spp.sub %in% family.geo.results$spp
  if(presence.test==TRUE){
    state <- family.geo.results$allopatry[
      family.geo.results$spp == spp.sub
    ]
  } else if(nd2.taxonomy$Synonymy[ee] %in% family.geo.results$spp){
    spp.syn <- nd2.taxonomy$Synonymy[ee]
    state <- family.geo.results$allopatry[
      family.geo.results$spp == spp.syn
    ]
  } else {
    state <- NA
  }
  nd2.family.states[ee] <- state
}

nd2.family.states[nd2.family.states=="0"] <- "Sympatry"
nd2.family.states[nd2.family.states=="1"] <- "Allopatry"

nd2.taxonomy$GeographyStateFamily <- as.character(nd2.family.states)

# co1
co1.family.states <- rep(as.numeric(NA), nrow(co1.taxonomy))
```

```
for(ee in 1:nrow(co1.taxonomy)){
  spp <- co1.taxonomy$species[ee] %>% as.character
  spp.sub <- gsub("_", " ", spp)
  presence.test <- spp.sub %in% family.geo.results$spp
  if(presence.test==TRUE){
    state <- family.geo.results$allopatry[
      family.geo.results$spp == spp.sub
    ]
  } else if(co1.taxonomy$Synonymy[ee] %in% family.geo.results$spp){
    spp.syn <- co1.taxonomy$Synonymy[ee]
    state <- family.geo.results$allopatry[
      family.geo.results$spp == spp.syn
    ]
  } else {
    state <- NA
  }
  co1.family.states[ee] <- state
}

co1.family.states[co1.family.states=="0"] <- "Sympatry"
co1.family.states[co1.family.states=="1"] <- "Allopatry"

co1.taxonomy$GeographyStateFamily <- as.character(co1.family.states)



# cytb
mt.family.states <- rep(as.numeric(NA), nrow(mt.taxonomy))
for(ee in 1:nrow(mt.taxonomy)){
  spp <- mt.taxonomy$species[ee] %>% as.character
  spp.sub <- gsub("_", " ", spp)
  presence.test <- spp.sub %in% family.geo.results$spp
  if(presence.test==TRUE){
    state <- family.geo.results$allopatry[
      family.geo.results$spp == spp.sub
    ]
  } else if(mt.taxonomy$Synonymy[ee] %in% family.geo.results$spp){
    spp.syn <- mt.taxonomy$Synonymy[ee]
    state <- family.geo.results$allopatry[
      family.geo.results$spp == spp.syn
    ]
  } else {
    state <- NA
  }
  mt.family.states[ee] <- state
}

mt.family.states[mt.family.states=="0"] <- "Sympatry"
mt.family.states[mt.family.states=="1"] <- "Allopatry"

mt.taxonomy$GeographyStateFamily <- as.character(mt.family.states)

# mc1r
```

```
mc1r.family.states <- rep(as.numeric(NA), nrow(mc1r.taxonomy))
for(ee in 1:nrow(mc1r.taxonomy)){
  spp <- mc1r.taxonomy$species[ee] %>% as.character
  spp.sub <- gsub("_", " ", spp)
  presence.test <- spp.sub %in% family.geo.results$spp
  if(presence.test==TRUE){
    state <- family.geo.results$allopatry[
      family.geo.results$spp == spp.sub
    ]
  } else if(mc1r.taxonomy$Synonymy[ee] %in% family.geo.results$spp){
    spp.syn <- mc1r.taxonomy$Synonymy[ee]
    state <- family.geo.results$allopatry[
      family.geo.results$spp == spp.syn
    ]
  } else {
    state <- NA
  }
  mc1r.family.states[ee] <- state
}

mc1r.family.states[mc1r.family.states=="0"] <- "Sympatry"
mc1r.family.states[mc1r.family.states=="1"] <- "Allopatry"

mc1r.taxonomy$GeographyStateFamily <- as.character(mc1r.family.states)

# rag1
rag1.family.states <- rep(as.numeric(NA), nrow(rag1.taxonomy))
for(ee in 1:nrow(rag1.taxonomy)){
  spp <- rag1.taxonomy$species[ee] %>% as.character
  spp.sub <- gsub("_", " ", spp)
  presence.test <- spp.sub %in% family.geo.results$spp
  if(presence.test==TRUE){
    state <- family.geo.results$allopatry[
      family.geo.results$spp == spp.sub
    ]
  } else if(rag1.taxonomy$Synonymy[ee] %in% family.geo.results$spp){
    spp.syn <- rag1.taxonomy$Synonymy[ee]
    state <- family.geo.results$allopatry[
      family.geo.results$spp == spp.syn
    ]
  } else {
    state <- NA
  }
  rag1.family.states[ee] <- state
}

rag1.family.states[rag1.family.states=="0"] <- "Sympatry"
rag1.family.states[rag1.family.states=="1"] <- "Allopatry"

rag1.taxonomy$GeographyStateFamily <- as.character(rag1.family.states)

# egr1
egr1.family.states <- rep(as.numeric(NA), nrow(egr1.taxonomy))
```

```
for(ee in 1:nrow(egr1.taxonomy)){
  spp <- egr1.taxonomy$species[ee] %>% as.character
  spp.sub <- gsub("_", " ", spp)
  presence.test <- spp.sub %in% family.geo.results$spp
  if(presence.test==TRUE){
    state <- family.geo.results$allopatry[
      family.geo.results$spp == spp.sub
    ]
  } else if(egr1.taxonomy$Synonymy[ee] %in% family.geo.results$spp){
    spp.syn <- egr1.taxonomy$Synonymy[ee]
    state <- family.geo.results$allopatry[
      family.geo.results$spp == spp.syn
    ]
  } else {
    state <- NA
  }
  egr1.family.states[ee] <- state
}

egr1.family.states[egr1.family.states=="0"] <- "Sympatry"
egr1.family.states[egr1.family.states=="1"] <- "Allopatry"

egr1.taxonomy$GeographyStateFamily <- as.character(egr1.family.states)
```

4.1.5 Combine range and body size data

Data on species range sizes and body mass needs to be added to each of the gene `data.frames` to test whether π correlates with these variables. First, let's do range size.

```
range.size <- readRDS("data/SpeciesRangeSize.RDS")

## cytb
spp.range.size <- lapply(1:nrow(cytb.taxonomy), function(ee){
  spp <- cytb.taxonomy$species[ee] %>% as.character
  spp.sub <- gsub("_", " ", spp)
  presence.test <- spp.sub %in% names(range.size)
  if(presence.test==TRUE){
    ind.range.size <- range.size[names(range.size) == spp.sub]
  } else if(spp %in% syn.dat$synonymyJetz){
    spp.syn <- syn.dat$synonymyGeo[
      syn.dat$synonymyJetz==spp
    ]
    spp.syn <- spp.syn[!is.na(spp.syn)][1]
    spp.syn <- gsub("_"," ",spp.syn)
    if(spp.syn %in% names(range.size)){
      ind.range.size <- range.size[names(range.size) == spp.syn]
    } else {
      ind.range.size <- NA
    }
  } else {
    ind.range.size <- NA
  }
```

```
                                                        }) %>% unlist

cytb.taxonomy$rangeSize <- spp.range.size

## nd2
spp.range.size <- lapply(1:nrow(nd2.taxonomy), function(ee){
  spp <- nd2.taxonomy$species[ee] %>% as.character
  spp.sub <- gsub("_", " ", spp)
  presence.test <- spp.sub %in% names(range.size)
  if(presence.test==TRUE){
    ind.range.size <- range.size[names(range.size) == spp.sub]
  } else if(spp %in% syn.dat$synonymyJetz){
    spp.syn <- syn.dat$synonymyGeo[
      syn.dat$synonymyJetz==spp
    ]
    spp.syn <- spp.syn[!is.na(spp.syn)][1]
    spp.syn <- gsub("_"," ",spp.syn)
    if(spp.syn %in% names(range.size)){
      ind.range.size <- range.size[names(range.size) == spp.syn]
    } else {
      ind.range.size <- NA
    }
  } else {
    ind.range.size <- NA
  }

                                                        }) %>% unlist

nd2.taxonomy$rangeSize <- spp.range.size

## co1
spp.range.size <- lapply(1:nrow(co1.taxonomy), function(ee){
  spp <- co1.taxonomy$species[ee] %>% as.character
  spp.sub <- gsub("_", " ", spp)
  presence.test <- spp.sub %in% names(range.size)
  if(presence.test==TRUE){
    ind.range.size <- range.size[names(range.size) == spp.sub]
  } else if(spp %in% syn.dat$synonymyJetz){
    spp.syn <- syn.dat$synonymyGeo[
      syn.dat$synonymyJetz==spp
    ]
    spp.syn <- spp.syn[!is.na(spp.syn)][1]
    spp.syn <- gsub("_"," ",spp.syn)
    if(spp.syn %in% names(range.size)){
      ind.range.size <- range.size[names(range.size) == spp.syn]
    } else {
      ind.range.size <- NA
    }
  } else {
    ind.range.size <- NA
  }

                                                        }) %>% unlist
```

```
co1.taxonomy$rangeSize <- spp.range.size

## mt
spp.range.size <- lapply(1:nrow(mt.taxonomy), function(ee){
  spp <- mt.taxonomy$species[ee] %>% as.character
  spp.sub <- gsub("_", " ", spp)
  presence.test <- spp.sub %in% names(range.size)
  if(presence.test==TRUE){
    ind.range.size <- range.size[names(range.size) == spp.sub]
  } else if(spp %in% syn.dat$synonymyJetz){
    spp.syn <- syn.dat$synonymyGeo[
      syn.dat$synonymyJetz==spp
    ]
    spp.syn <- spp.syn[!is.na(spp.syn)][1]
    spp.syn <- gsub("_"," ",spp.syn)
    if(spp.syn %in% names(range.size)){
      ind.range.size <- range.size[names(range.size) == spp.syn]
    } else {
      ind.range.size <- NA
    }
  } else {
    ind.range.size <- NA
  }

                                                        }) %>% unlist

mt.taxonomy$rangeSize <- spp.range.size


## mc1r
spp.range.size <- lapply(1:nrow(mc1r.taxonomy), function(ee){
  spp <- mc1r.taxonomy$species[ee] %>% as.character
  spp.sub <- gsub("_", " ", spp)
  presence.test <- spp.sub %in% names(range.size)
  if(presence.test==TRUE){
    ind.range.size <- range.size[names(range.size) == spp.sub]
  } else if(spp %in% syn.dat$synonymyJetz){
    spp.syn <- syn.dat$synonymyGeo[
      syn.dat$synonymyJetz==spp
    ]
    spp.syn <- spp.syn[!is.na(spp.syn)][1]
    spp.syn <- gsub("_"," ",spp.syn)
    if(spp.syn %in% names(range.size)){
      ind.range.size <- range.size[names(range.size) == spp.syn]
    } else {
      ind.range.size <- NA
    }
  } else {
    ind.range.size <- NA
  }

                                                        }) %>% unlist
```

```
mc1r.taxonomy$rangeSize <- spp.range.size

## rag1
spp.range.size <- lapply(1:nrow(rag1.taxonomy), function(ee){
  spp <- rag1.taxonomy$species[ee] %>% as.character
  spp.sub <- gsub("_", " ", spp)
  presence.test <- spp.sub %in% names(range.size)
  if(presence.test==TRUE){
    ind.range.size <- range.size[names(range.size) == spp.sub]
  } else if(spp %in% syn.dat$synonymyJetz){
    spp.syn <- syn.dat$synonymyGeo[
      syn.dat$synonymyJetz==spp
    ]
    spp.syn <- spp.syn[!is.na(spp.syn)][1]
    spp.syn <- gsub("_"," ",spp.syn)
    if(spp.syn %in% names(range.size)){
      ind.range.size <- range.size[names(range.size) == spp.syn]
    } else {
      ind.range.size <- NA
    }
  } else {
    ind.range.size <- NA
  }

                                                        }) %>% unlist

rag1.taxonomy$rangeSize <- spp.range.size

## egr1
spp.range.size <- lapply(1:nrow(egr1.taxonomy), function(ee){
  spp <- egr1.taxonomy$species[ee] %>% as.character
  spp.sub <- gsub("_", " ", spp)
  presence.test <- spp.sub %in% names(range.size)
  if(presence.test==TRUE){
    ind.range.size <- range.size[names(range.size) == spp.sub]
  } else if(spp %in% syn.dat$synonymyJetz){
    spp.syn <- syn.dat$synonymyGeo[
      syn.dat$synonymyJetz==spp
    ]
    spp.syn <- spp.syn[!is.na(spp.syn)][1]
    spp.syn <- gsub("_"," ",spp.syn)
    if(spp.syn %in% names(range.size)){
      ind.range.size <- range.size[names(range.size) == spp.syn]
    } else {
      ind.range.size <- NA
    }
  } else {
    ind.range.size <- NA
  }

                                                        }) %>% unlist

egr1.taxonomy$rangeSize <- spp.range.size
```

Now body size

```
mass.data <- read.csv("data/mass_data.csv")
mass.data$ï..Jetz.species <- gsub(" ", "_", mass.data$ï..Jetz.species)

## cytb
spp.mass <- lapply(1:nrow(cytb.taxonomy), function(ee){
  spp <- cytb.taxonomy$species[ee] %>% as.character
  spp.syn <- cytb.taxonomy$species[ee] %>% as.character
  presence.test <- spp %in% mass.data$ï..Jetz.species
  if(presence.test==TRUE){
    ind.range.size <- mass.data$Mass[mass.data$ï..Jetz.species==spp]
  } else if(spp.syn %in% mass.data$ï..Jetz.species){
    ind.range.size <- mass.data$Mass[mass.data$ï..Jetz.species==spp.syn]
  } else {
    ind.range.size <- NA
  }
}) %>% unlist
cytb.taxonomy$speciesMass <- spp.mass

## nd2
spp.mass <- lapply(1:nrow(nd2.taxonomy), function(ee){
  spp <- nd2.taxonomy$species[ee] %>% as.character
  spp.syn <- nd2.taxonomy$species[ee] %>% as.character
  presence.test <- spp %in% mass.data$ï..Jetz.species
  if(presence.test==TRUE){
    ind.range.size <- mass.data$Mass[mass.data$ï..Jetz.species==spp]
  } else if(spp.syn %in% mass.data$ï..Jetz.species){
    ind.range.size <- mass.data$Mass[mass.data$ï..Jetz.species==spp.syn]
  } else {
    ind.range.size <- NA
  }
}) %>% unlist
nd2.taxonomy$speciesMass <- spp.mass

## co1
spp.mass <- lapply(1:nrow(co1.taxonomy), function(ee){
  spp <- co1.taxonomy$species[ee] %>% as.character
  spp.syn <- co1.taxonomy$species[ee] %>% as.character
  presence.test <- spp %in% mass.data$ï..Jetz.species
  if(presence.test==TRUE){
    ind.range.size <- mass.data$Mass[mass.data$ï..Jetz.species==spp]
  } else if(spp.syn %in% mass.data$ï..Jetz.species){
    ind.range.size <- mass.data$Mass[mass.data$ï..Jetz.species==spp.syn]
  } else {
    ind.range.size <- NA
  }
}) %>% unlist
co1.taxonomy$speciesMass <- spp.mass

## mt
spp.mass <- lapply(1:nrow(mt.taxonomy), function(ee){
  spp <- mt.taxonomy$species[ee] %>% as.character
  spp.syn <- mt.taxonomy$species[ee] %>% as.character
```

```
  presence.test <- spp %in% mass.data$ï..Jetz.species
  if(presence.test==TRUE){
    ind.range.size <- mass.data$Mass[mass.data$ï..Jetz.species==spp]
  } else if(spp.syn %in% mass.data$ï..Jetz.species){
    ind.range.size <- mass.data$Mass[mass.data$ï..Jetz.species==spp.syn]
  } else {
    ind.range.size <- NA
  }
}) %>% unlist
mt.taxonomy$speciesMass <- spp.mass

## mc1r
spp.mass <- lapply(1:nrow(mc1r.taxonomy), function(ee){
  spp <- mc1r.taxonomy$species[ee] %>% as.character
  spp.syn <- mc1r.taxonomy$species[ee] %>% as.character
  presence.test <- spp %in% mass.data$ï..Jetz.species
  if(presence.test==TRUE){
    ind.range.size <- mass.data$Mass[mass.data$ï..Jetz.species==spp]
  } else if(spp.syn %in% mass.data$ï..Jetz.species){
    ind.range.size <- mass.data$Mass[mass.data$ï..Jetz.species==spp.syn]
  } else {
    ind.range.size <- NA
  }
}) %>% unlist
mc1r.taxonomy$speciesMass <- spp.mass

## rag1
spp.mass <- lapply(1:nrow(rag1.taxonomy), function(ee){
  spp <- rag1.taxonomy$species[ee] %>% as.character
  spp.syn <- rag1.taxonomy$species[ee] %>% as.character
  presence.test <- spp %in% mass.data$ï..Jetz.species
  if(presence.test==TRUE){
    ind.range.size <- mass.data$Mass[mass.data$ï..Jetz.species==spp]
  } else if(spp.syn %in% mass.data$ï..Jetz.species){
    ind.range.size <- mass.data$Mass[mass.data$ï..Jetz.species==spp.syn]
  } else {
    ind.range.size <- NA
  }
}) %>% unlist
rag1.taxonomy$speciesMass <- spp.mass

## egr1
spp.mass <- lapply(1:nrow(egr1.taxonomy), function(ee){
  spp <- egr1.taxonomy$species[ee] %>% as.character
  spp.syn <- egr1.taxonomy$species[ee] %>% as.character
  presence.test <- spp %in% mass.data$ï..Jetz.species
  if(presence.test==TRUE){
    ind.range.size <- mass.data$Mass[mass.data$ï..Jetz.species==spp]
  } else if(spp.syn %in% mass.data$ï..Jetz.species){
    ind.range.size <- mass.data$Mass[mass.data$ï..Jetz.species==spp.syn]
  } else {
    ind.range.size <- NA
  }
```

```
}) %>% unlist
egr1.taxonomy$speciesMass <- spp.mass
```

See if π correlates with either range size or body mass.

```
cytb.cor <- melt(cytb.taxonomy, id = "pi",
                 measure.vars =c("rangeSize",
                                 "speciesMass"))
cytb.cor$Gene <- "CYTB"

nd2.cor <- melt(nd2.taxonomy, id = "pi",
                measure.vars =c("rangeSize",
                                "speciesMass"))
nd2.cor$Gene <- "ND2"

co1.cor <- melt(co1.taxonomy, id = "pi",
                measure.vars =c("rangeSize",
                                "speciesMass"))
co1.cor$Gene <- "CO1"

mt.cor <- melt(mt.taxonomy, id = "pi",
               measure.vars = c("rangeSize",
                                "speciesMass"))
mt.cor$Gene <- "mtDNA"

mc1r.cor <- melt(mc1r.taxonomy, id = "pi",
                 measure.vars = c("rangeSize",
                                  "speciesMass"))
mc1r.cor$Gene <- "MC1R"

rag1.cor <- melt(rag1.taxonomy, id = "pi",
                 measure.vars = c("rangeSize",
                                  "speciesMass"))
rag1.cor$Gene <- "RAG1"

egr1.cor <- melt(egr1.taxonomy, id = "pi",
                 measure.vars = c("rangeSize",
                                  "speciesMass"))
egr1.cor$Gene <- "EGR1"

cor.plot.dat <- rbind(cytb.cor,
                      nd2.cor,
                      co1.cor,
                      mt.cor,
                      mc1r.cor,
                      rag1.cor,
                      egr1.cor)

x.lab <- expression("log(" * pi * ")")

png("FigureExport/piRangeMass.png", width = 8,
    height=4, units="in", res=500)
ggplot(cor.plot.dat, aes(x = log(pi),
```

```
                    y = log(value),
                    col = Gene)) +
  geom_point(size=.25) +
  my.theme.bp +
  geom_rangeframe() +
  labs(y = "log(variable)",
       x = x.lab) +
  facet_wrap(~variable,
             scales = "free") +
  #scale_color_manual(values = gene.palette) +
  geom_smooth(method="lm")
dev.off()
```

4.1.6 Combine data files are visualize data

First, we load the results of the workflow so far and combine into a single `data.frame`.

```
cytb.taxonomy$gene <- "CYTB"
nd2.taxonomy$gene <- "ND2"
co1.taxonomy$gene <- "CO1"
mt.taxonomy$gene <- "mtDNA"
mc1r.taxonomy$gene <- "MC1R"
rag1.taxonomy$gene <- "RAG1"
egr1.taxonomy$gene <- "EGR1"

plot.dat <- rbind(cytb.taxonomy,
                  nd2.taxonomy,
                  co1.taxonomy,
                  mt.taxonomy,
                  mc1r.taxonomy,
                  rag1.taxonomy)#,
                  #egr1.taxonomy)

plot.dat$gene <- as.factor(plot.dat$gene)
plot.dat$gene <- factor(plot.dat$gene,
                        levels(plot.dat$gene)[c(1,2,5,4,3,6)])
```

We need to add one last piece of information – the major latitudinal region of the species. The script to generate these data is `quantifyRegion.R`. Let's load the results and combine with the data.

```
region.res <- readRDS("data/SpeciesGeographicRegions.RDS")
names(region.res) <- gsub(" ","_", names(region.res))

plot.dat$region <- NA
for(i in 1:nrow(plot.dat)){
    rr <- region.res[names(region.res)==plot.dat$species[i]] %>%
      as.character
    if(length(rr)!=0){
        plot.dat$region[i] <- rr
    }
}
```

These data can also be simplified to visual only those species found in temperate and tropical regions respectively. A plot of all regions will be included in the supplement.

```
plot.dat$regionSimple <- NA

for(i in 1:nrow(plot.dat)){
    if(!is.na(plot.dat$region[i])){
    if(plot.dat$region[i] %in% c("High temperate",
                                 "Low temperate")){
        plot.dat$regionSimple[i] <- "Temperate"
    } else if(plot.dat$region[i] == "Tropical"){
        plot.dat$regionSimple[i] <- "Tropical"
    }
}
}
```

Now we can start making the plots

```
y.lab <- expression("log(" * pi * ")")

piSister<- ggplot(plot.dat[!is.na(plot.dat$GeographyStateSister),],
                          aes(x = GeographyStateSister,
                              y = log(pi),
                              fill = gene)) +
  geom_boxplot() +
  my.theme.bp +
  scale_fill_manual(values = gene.palette) +
  labs(x = "Geographic state",
       y = y.lab,
       fill = "DNA source") +
  ggtitle("Allopatry defined using sister species")


piFamily<- ggplot(plot.dat[!is.na(plot.dat$GeographyStateFamily),],
                          aes(x = GeographyStateFamily,
                              y = log(pi),
                              fill = gene)) +
  geom_boxplot() +
  my.theme.bp +
  scale_fill_manual(values = gene.palette) +
  labs(x = "Geographic state",
       y = y.lab,
       fill = "DNA source")+
  ggtitle("Allopatry defined using confamilial species")



#png("FigureExport/piAllopatry.png", width = 6, height=8,
#    units = "in", res=500)
#gridExtra::grid.arrange(piSister + theme(legend.position = "none"),
#                        piFamily + theme(legend.position="none"),
#                        nrow=2)
#dev.off()
```

```
piRegion <- ggplot(plot.dat[!is.na(plot.dat$regionSimple),],
                   aes(x = factor(regionSimple),
                       y = log(pi),
                       fill = factor(gene))) +
  geom_boxplot() +
  my.theme.bp +
  scale_fill_manual(values = gene.palette) +
  labs(x = "Geographic region",
       y = y.lab,
       fill = "DNA source")+
  ggtitle("Nucleotide diversity by latitudinal region")

piRegionSupp <- ggplot(plot.dat[!is.na(plot.dat$region),],
                   aes(x = region,
                       y = log(pi),
                       fill = gene)) +
  geom_boxplot() +
  my.theme.bp +
  scale_fill_manual(values = gene.palette) +
  labs(x = "Geographic region",
       y = y.lab,
       fill = "DNA source")+
  ggtitle("Nucleotide diversity by latitudinal region")
```

4.2 Compare family π with mean and maximum number of overlapping species

In order to make these comparisons family-specific π estimates are required.

```
cytb.fam.pi <- cytb.taxonomy %>%
  group_by(family) %>%
  summarise(famPi = pi %>% mean(na.rm=T),
            gene = "CYTB")

nd2.fam.pi <- nd2.taxonomy %>%
  group_by(family) %>%
  summarise(famPi = pi %>% mean(na.rm=T),
            gene = "ND2")

co1.fam.pi <- co1.taxonomy %>%
  group_by(family) %>%
  summarise(famPi = pi %>% mean(na.rm=T),
            gene = "CO1")

mt.fam.pi <- mt.taxonomy %>%
  group_by(family) %>%
  summarise(famPi = pi %>% mean(na.rm=T),
            gene="mtDNA")

mc1r.fam.pi <- mc1r.taxonomy %>%
  group_by(family) %>%
  summarise(famPi = pi %>% mean(na.rm=T),
            gene = "MC1R")

rag1.fam.pi <- rag1.taxonomy %>%
  group_by(family) %>%
  summarise(famPi = pi %>% mean(na.rm=T),
            gene = "RAG1")

meanMaxPi <- rbind(cytb.fam.pi,
                   nd2.fam.pi,
                   co1.fam.pi,
                   mt.fam.pi,
                   mc1r.fam.pi,
                   rag1.fam.pi)
```

Next, the data from Crouch (2021) are loaded and added to the data. Only families with at least ten species were quantified by Crouch (2021) so some families will be missing.

```
bird.overlap.res <- read.csv("data/bird.overlaps.csv")

meanMaxPi$meanOverlap <- NA
meanMaxPi$maxOverlap <- NA

for(j in 1:nrow(meanMaxPi)){
  meanVal <-  bird.overlap.res$meanOverlap[
    bird.overlap.res$family == meanMaxPi$family[j]]
  meanVal <- meanVal[!is.na(meanVal)]
  if(length(meanVal)>0){
```

```
    meanMaxPi$meanOverlap[j] <- meanVal
  }
  maxVal <- bird.overlap.res$maxOverlap[
    bird.overlap.res$family == meanMaxPi$family[j]]
  maxVal <- maxVal[!is.na(maxVal)]
  if(length(maxVal)>0){
    meanMaxPi$maxOverlap[j] <- maxVal
  }
}
```

Now we can take a look at the results

```
meanMaxPi$gene <- as.factor(meanMaxPi$gene)
meanMaxPi$gene <- factor(meanMaxPi$gene,
                         levels(meanMaxPi$gene)[c(1,2,5,4,3,6)])

meanOverlapPi <- ggplot(meanMaxPi, aes(x = log(meanOverlap),
                                       y = log(famPi),
                                       col = gene)) +
  geom_point() +
  my.theme.bp +
  geom_rangeframe() +
  labs(x = "log(mean number of overlapping ranges)",
       y = y.lab) +
  scale_color_manual(values = gene.palette) +
  geom_smooth(method="lm")

maxOverlapPi <- ggplot(meanMaxPi, aes(x = log(maxOverlap),
                                       y = log(famPi),
                                       col = gene)) +
  geom_point() +
  my.theme.bp +
  geom_rangeframe() +
  labs(x = "log(maximum number of overlapping ranges)",
       y = y.lab) +
  scale_color_manual(values = gene.palette) +
  geom_smooth(method="lm")


#png("FigureExport/meanMaxPi.png", width = 6, height=8,
#    units = "in", res=500)
#gridExtra::grid.arrange(meanOverlapPi + theme(legend.position = "none"),
#                        maxOverlapPi + theme(legend.position="none"),
#                        nrow=2)
#dev.off()
```

4.3 Difference in π between allopatric and sympatric species

In this section the difference in π between allopatric and sympatric species within a family is compared against family diversity.

```
source("scripts/compare.family.pi.R")

cytb.pi.diff.sister <- compare.family.pi(cytb.taxonomy,
                                         taxo,
                                         "GeographyStateSister")
nd2.pi.diff.sister <- compare.family.pi(nd2.taxonomy,
                                        taxo,
                                        "GeographyStateSister")
co1.pi.diff.sister <- compare.family.pi(co1.taxonomy,
                                        taxo,
                                        "GeographyStateSister")
mt.pi.diff.sister <- compare.family.pi(mt.taxonomy,
                                       taxo,
                                       "GeographyStateSister")
mc1r.pi.diff.sister <- compare.family.pi(mc1r.taxonomy,
                                         taxo,
                                         "GeographyStateSister")
rag1.pi.diff.sister <- compare.family.pi(rag1.taxonomy,
                                         taxo,
                                         "GeographyStateSister")

cytb.pi.diff.family <- compare.family.pi(cytb.taxonomy,
                                         taxo,
                                         "GeographyStateFamily")
nd2.pi.diff.family <- compare.family.pi(nd2.taxonomy,
                                        taxo,
                                        "GeographyStateFamily")
co1.pi.diff.family <- compare.family.pi(co1.taxonomy,
                                        taxo,
                                        "GeographyStateFamily")
mt.pi.diff.family <- compare.family.pi(mt.taxonomy,
                                       taxo,
                                       "GeographyStateFamily")
mc1r.pi.diff.family <- compare.family.pi(mc1r.taxonomy,
                                         taxo,
                                         "GeographyStateFamily")
rag1.pi.diff.family <- compare.family.pi(rag1.taxonomy,
                                         taxo,
                                         "GeographyStateFamily")

cytb.pi.diff.sister$gene <- "CYTB"
nd2.pi.diff.sister$gene <- "ND2"
co1.pi.diff.sister$gene <- "CO1"
mt.pi.diff.sister$gene <- "mtDNA"
mc1r.pi.diff.sister$gene <- "MC1R"
rag1.pi.diff.sister$gene <- "RAG1"

cytb.pi.diff.family$gene <- "CYTB"
nd2.pi.diff.family$gene <- "ND2"
co1.pi.diff.family$gene <- "CO1"
mt.pi.diff.family$gene <- "mtDNA"
mc1r.pi.diff.family$gene <- "MC1R"
rag1.pi.diff.family$gene <- "RAG1"
```

```
pi.diff.sister.full <- rbind(cytb.pi.diff.sister,
                             nd2.pi.diff.sister,
                             co1.pi.diff.sister,
                             mt.pi.diff.sister,
                             mc1r.pi.diff.sister,
                             rag1.pi.diff.sister)

pi.diff.family.full <- rbind(cytb.pi.diff.family,
                             nd2.pi.diff.family,
                             co1.pi.diff.family,
                             mt.pi.diff.family,
                             mc1r.pi.diff.family,
                             rag1.pi.diff.family)

y.lab.diff <- expression("allopatry" ~ pi ~ "- sympatry" ~ pi)

pi.diff.sister <- ggplot(pi.diff.sister.full, aes(x = log(div),
                         y = piDiff,
                         color = gene)) +
    geom_point() +
  theme_bw() +
  my.theme.scat +
  labs(x = "log(family diversity)",
       y = y.lab.diff) +
  scale_color_manual(values = gene.palette) +
  geom_smooth(method="lm") +
  xlim(c(2,6.3))

pi.diff.family <- ggplot(pi.diff.family.full, aes(x = log(div),
                         y = piDiff,
                         color = gene)) +
    geom_point() +
  theme_bw() +
  my.theme.scat +
  labs(x = "log(family diversity)",
       y = y.lab.diff) +
  scale_color_manual(values = gene.palette) +
  geom_smooth(method="lm")+
  xlim(c(2,6.3))


#png("FigureExport/piAllopatryDiv.png", width = 6, height=8,
#    units = "in", res=500)
#gridExtra::grid.arrange(pi.diff.sister +
#                          theme(legend.position = "top") +
#                          ggtitle("Allopatry defined using sister species"),
#                        pi.diff.family +
#                          theme(legend.position="none") +
#                          ggtitle("Allopatry defined using confailial species"),
#                        nrow=2)
#dev.off()
```

4.4 Global π plots

Here, in similar fashion to the global plots of NND scores, we calculate the mean per-grid cell π value for the species present. In an additional step to before, we will also record how many species are sampled at each point. This function is applied to the π calculations from each gene (and mitochondrial genomes).

```
source("scripts/calculate.mean.pi.R")

# CO1
global.co1.scores <- lapply(1:nrow(spp.presence), function(ee){
  calculate.mean.pi(spp.presence[ee,],
                    co1.taxonomy,
                    syn.dat)
}) %>% bind_rows()

# cytb
global.cytb.scores <- lapply(1:nrow(spp.presence), function(ee){
  calculate.mean.pi(spp.presence[ee,],
                    cytb.taxonomy,
                    syn.dat)
}) %>% bind_rows()

# nd2
global.nd2.scores <- lapply(1:nrow(spp.presence), function(ee){
  calculate.mean.pi(spp.presence[ee,],
                    nd2.taxonomy,
                    syn.dat)
}) %>% bind_rows()

# mtDNA
global.mt.scores <- lapply(1:nrow(spp.presence), function(ee){
  calculate.mean.pi(spp.presence[ee,],
                    mt.taxonomy,
                    syn.dat)
}) %>% bind_rows()

# mc1r
global.mc1r.scores <- lapply(1:nrow(spp.presence), function(ee){
  calculate.mean.pi(spp.presence[ee,],
                    mc1r.taxonomy,
                    syn.dat)
}) %>% bind_rows()

# rag1
global.rag1.scores <- lapply(1:nrow(spp.presence), function(ee){
  calculate.mean.pi(spp.presence[ee,],
                    rag1.taxonomy,
                    syn.dat)
}) %>% bind_rows()
```

First, we want to look at how the values compare between the different data sets.

```
pairwise.dat <- data.frame(co1 = log(global.co1.scores$meanPi),
                           cytb = log(global.cytb.scores$meanPi),
                           nd2 = log(global.nd2.scores$meanPi),
                           mt = log(global.mt.scores$meanPi),
                           mc1r = log(global.mc1r.scores$meanPi),
                           rag1 = log(global.rag1.scores$meanPi))

pairwise.dat$co1[is.infinite(pairwise.dat$co1)] <- NA
pairwise.dat$cytb[is.infinite(pairwise.dat$cytb)] <- NA
pairwise.dat$nd2[is.infinite(pairwise.dat$nd2)] <- NA
pairwise.dat$mt[is.infinite(pairwise.dat$mt)] <- NA
pairwise.dat$mc1r[is.infinite(pairwise.dat$mc1r)] <- NA
pairwise.dat$rag1[is.infinite(pairwise.dat$rag1)] <- NA

panel.cor <- function(x, y){
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <- round(cor(x, y, use = "complete.obs"), digits=2)
    txt <- paste0("R = ", r)
    text(0.5, 0.5, txt, cex=2)
}

upper.panel<-function(x, y){
  points(x,y, pch = 19)
}

png("FigureExport/pairwisePlot.png", width=7,
    height = 7, units="in", res=500)
pairs(pairwise.dat,
      lower.panel = panel.cor,
      upper.panel = upper.panel)
dev.off()
```

Each gene will have two plots: (1) global distribution of mean π scores (2) the relationship between π and sample size. For the latter, the process is the same as how the global maps for morphology were generated earlier: a color palette is generated for the data based on the relationship between number of species (here it's sample size of the gene coverage, not total species richness as above) and the map generated with an inset image of the correlation between sample size and mean π. For those plots we can define the objects that will be common across all four.

```
d <- expand.grid(x = seq(0, 1, 0.01), y = seq(0, 1, 0.01)) %>%
  mutate(fill_val = atan(y/x),
         transparency = x + y)

p <- ggplot(d, aes(x, y, fill = fill_val, alpha = transparency)) +
  geom_tile() +
  scale_fill_paletteer_c("scico::berlin") +
  theme_void() +
  theme(legend.position = "none")

pal_d <- ggplot_build(p)$data[[1]] %>%
  select(x, y, fill) %>%
  mutate(x = as.character(x),
```

```
         y = as.character(y))
```

Now we generate the plots for the four genetic data sets. These plots were generated independently and combined in post processing as we already had figured out the dimensions for putting the inset image inside the map and didn't want to do it again.

4.4.1 cytb

```
plot.dat <- data.frame(Var1 = coords$Var1,
                       Var2 = coords$Var2,
                       pi = log(global.cytb.scores$meanPi),
                       n = global.cytb.scores$n)

plot.dat.full <- plot.dat %>%
  mutate(x = as.character(round(scales::rescale(pi), 2)),
         y = as.character(round(scales::rescale(n), 2))) %>%
  left_join(pal_d)


global.map <- ggplot(plot.dat.full, aes(Var1, Var2)) +
  geom_point(aes(color=fill),
             shape = 15,
             size = .35) +
  scale_color_identity()+
  guides(color = "none") +
  coord_sf(xlim = c(-180, 189),
           ylim = c(-60, 83.5),
           expand = FALSE) +
  theme_bw() +
  labs(x = "Longitude",
       y = "Latitude") +
  theme(axis.text=element_text(size=14),
        axis.title=element_text(size=14),
        panel.background = element_rect(fill = "gray45"),
        panel.grid.major = element_line(colour = "gray50"))


inset.figure <- ggplot(plot.dat.full,
                       aes(n, pi)) +
  geom_point(aes(color = fill),
             size = .5) +
  scale_color_identity() +
  labs(x = "Number of sampled species",
       y = expression("log(" * pi * ")")) +
  theme_bw()

png("FigureExport/cytbGlobal.png",
    width=12,
    height = 9,
    units="in",
    res=500)
```

```
ggdraw(global.map + ggtitle("Cytochrome b")) +
  draw_plot(plot = inset.figure,
            x = .06,
            y = .29,
            width = .21,
            height = .25)
dev.off()


# remove extreme tails
plot.dat.full <- plot.dat.full[plot.dat.full$pi < -3 &
                                 plot.dat.full$pi >= -5,]
global.map.pi <- ggplot(plot.dat.full, aes(Var1, Var2)) +
  geom_point(aes(color=pi),
             shape = 15,
             size = .35) +
  scale_color_viridis(option="B")+
  #guides(color = "none") +
  coord_sf(xlim = c(-180, 189),
           ylim = c(-60, 83.5),
           expand = FALSE) +
  theme_bw() +
  labs(x = "Longitude",
       y = "Latitude",
       color = expression("log(" * pi * ")")) +
  theme(axis.text=element_text(size=14),
        axis.title=element_text(size=14),
        panel.background = element_rect(fill = "gray45"),
        panel.grid.major = element_line(colour = "gray50")) +
  ggtitle("Cytochrome b")

png("FigureExport/cytbGlobalPi.png",
    width=12,
    height = 9,
    units="in",
    res=500)
global.map.pi
dev.off()
```

4.4.2 co1

```
plot.dat <- data.frame(Var1 = coords$Var1,
                       Var2 = coords$Var2,
                       pi = log(global.co1.scores$meanPi),
                       n = global.co1.scores$n)

plot.dat.full <- plot.dat %>%
  mutate(x = as.character(round(scales::rescale(pi), 2)),
         y = as.character(round(scales::rescale(n), 2))) %>%
  left_join(pal_d)
```

```
global.map <- ggplot(plot.dat.full, aes(Var1, Var2)) +
  geom_point(aes(color=fill),
             shape = 15,
             size = .35) +
  scale_color_identity()+
  guides(color = "none") +
  coord_sf(xlim = c(-180, 189),
           ylim = c(-60, 83.5),
           expand = FALSE) +
  theme_bw() +
  labs(x = "Longitude",
       y = "Latitude") +
  theme(axis.text=element_text(size=14),
        axis.title=element_text(size=14),
        panel.background = element_rect(fill = "gray45"),
        panel.grid.major = element_line(colour = "gray50"))


inset.figure <- ggplot(plot.dat.full,
                       aes(n, pi)) +
  geom_point(aes(color = fill),
             size = .5) +
  scale_color_identity() +
  labs(x = "Number of sampled species",
       y = expression("log(" * pi * ")")) +
  theme_bw()

png("FigureExport/co1Global.png",
    width=12,
    height = 9,
    units="in",
    res=500)
ggdraw(global.map + ggtitle("Cytochrome c oxidase subunit I")) +
  draw_plot(plot = inset.figure,
            x = .06,
            y = .29,
            width = .21,
            height = .25)
dev.off()


plot.dat.full <- plot.dat.full[plot.dat.full$pi < -2.5 &
                                 plot.dat.full$pi >= -6,]
global.map.pi <- ggplot(plot.dat.full, aes(Var1, Var2)) +
  geom_point(aes(color=pi),
             shape = 15,
             size = .35) +
  scale_color_viridis(option="B")+
  #guides(color = "none") +
  coord_sf(xlim = c(-180, 189),
           ylim = c(-60, 83.5),
           expand = FALSE) +
  theme_bw() +
```

```
  labs(x = "Longitude",
       y = "Latitude",
       color = expression("log(" * pi * ")")) +
  theme(axis.text=element_text(size=14),
        axis.title=element_text(size=14),
        panel.background = element_rect(fill = "gray45"),
        panel.grid.major = element_line(colour = "gray50")) +
  ggtitle("Cytochrome c oxidase subunit I")

png("FigureExport/co1GlobalPi.png",
    width=12,
    height = 9,
    units="in",
    res=500)
global.map.pi
dev.off()
```

4.4.3 nd2

```
plot.dat <- data.frame(Var1 = coords$Var1,
                       Var2 = coords$Var2,
                       pi = log(global.nd2.scores$meanPi),
                       n = global.nd2.scores$n)

plot.dat.full <- plot.dat %>%
  mutate(x = as.character(round(scales::rescale(pi), 2)),
         y = as.character(round(scales::rescale(n), 2))) %>%
  left_join(pal_d)


global.map <- ggplot(plot.dat.full, aes(Var1, Var2)) +
  geom_point(aes(color=fill),
             shape = 15,
             size = .35) +
  scale_color_identity()+
  guides(color = "none") +
  coord_sf(xlim = c(-180, 189),
           ylim = c(-60, 83.5),
           expand = FALSE) +
  theme_bw() +
  labs(x = "Longitude",
       y = "Latitude") +
  theme(axis.text=element_text(size=14),
        axis.title=element_text(size=14),
        panel.background = element_rect(fill = "gray45"),
        panel.grid.major = element_line(colour = "gray50"))


inset.figure <- ggplot(plot.dat.full,
                       aes(n, pi)) +
  geom_point(aes(color = fill),
             size = .5) +
```

```
  scale_color_identity() +
  labs(x = "Number of sampled species",
       y = expression("log(" * pi * ")")) +
  theme_bw()

png("FigureExport/nd2Global.png",
    width=12,
    height = 9,
    units="in",
    res=500)
ggdraw(global.map + ggtitle("NADH dehydrogenase 2")) +
  draw_plot(plot = inset.figure,
            x = .06,
            y = .29,
            width = .21,
            height = .25)
dev.off()


plot.dat.full <- plot.dat.full[plot.dat.full$pi < -2.25 &
                                 plot.dat.full$pi >= -6,]
global.map.pi <- ggplot(plot.dat.full, aes(Var1, Var2)) +
  geom_point(aes(color=pi),
             shape = 15,
             size = .35) +
  scale_color_viridis(option="B")+
  #guides(color = "none") +
  coord_sf(xlim = c(-180, 189),
           ylim = c(-60, 83.5),
           expand = FALSE) +
  theme_bw() +
  labs(x = "Longitude",
       y = "Latitude",
       color = expression("log(" * pi * ")")) +
  theme(axis.text=element_text(size=14),
        axis.title=element_text(size=14),
        panel.background = element_rect(fill = "gray45"),
        panel.grid.major = element_line(colour = "gray50")) +
  ggtitle("NADH dehydrogenase 2")

png("FigureExport/nd2GlobalPi.png",
    width=12,
    height = 9,
    units="in",
    res=500)
global.map.pi
dev.off()
```

4.4.4 mtDNA

```
plot.dat <- data.frame(Var1 = coords$Var1,
                       Var2 = coords$Var2,
```

```
                      pi = log(global.mt.scores$meanPi),
                      n = global.mt.scores$n)

plot.dat.full <- plot.dat %>%
  mutate(x = as.character(round(scales::rescale(pi), 2)),
         y = as.character(round(scales::rescale(n), 2))) %>%
  left_join(pal_d)


global.map <- ggplot(plot.dat.full, aes(Var1, Var2)) +
  geom_point(aes(color=fill),
             shape = 15,
             size = .35) +
  scale_color_identity()+
  guides(color = "none") +
  coord_sf(xlim = c(-180, 189),
           ylim = c(-60, 83.5),
           expand = FALSE) +
  theme_bw() +
  labs(x = "Longitude",
       y = "Latitude") +
  theme(axis.text=element_text(size=14),
        axis.title=element_text(size=14),
        panel.background = element_rect(fill = "gray45"),
        panel.grid.major = element_line(colour = "gray50"))


inset.figure <- ggplot(plot.dat.full,
                       aes(n, pi)) +
  geom_point(aes(color = fill),
             size = .5) +
  scale_color_identity() +
  labs(x = "Number of sampled species",
       y = expression("log(" * pi * ")")) +
  theme_bw()

png("FigureExport/mtGlobal.png",
    width=12,
    height = 9,
    units="in",
    res=500)
ggdraw(global.map + ggtitle("Mitochondrial genomes")) +
  draw_plot(plot = inset.figure,
            x = .06,
            y = .29,
            width = .21,
            height = .25)
dev.off()


plot.dat.full <- plot.dat.full[plot.dat.full$pi < -4 &
                                 plot.dat.full$pi >= -7.5,]
global.map.pi <- ggplot(plot.dat.full, aes(Var1, Var2)) +
```

```
  geom_point(aes(color=pi),
             shape = 15,
             size = .35) +
  scale_color_viridis(option="B")+
  #guides(color = "none") +
  coord_sf(xlim = c(-180, 189),
           ylim = c(-60, 83.5),
           expand = FALSE) +
  theme_bw() +
  labs(x = "Longitude",
       y = "Latitude",
       color = expression("log(" * pi * ")")) +
  theme(axis.text=element_text(size=14),
        axis.title=element_text(size=14),
        panel.background = element_rect(fill = "gray45"),
        panel.grid.major = element_line(colour = "gray50")) +
  ggtitle("Mitochondrial genomes")

png("FigureExport/mtGlobalPi.png",
    width=12,
    height = 9,
    units="in",
    res=500)
global.map.pi
dev.off()
```

4.4.5 mc1r

```
plot.dat <- data.frame(Var1 = coords$Var1,
                       Var2 = coords$Var2,
                       pi = log(global.mc1r.scores$meanPi),
                       n = global.mc1r.scores$n)

plot.dat.full <- plot.dat %>%
  mutate(x = as.character(round(scales::rescale(pi), 2)),
         y = as.character(round(scales::rescale(n), 2))) %>%
  left_join(pal_d)


global.map <- ggplot(plot.dat.full, aes(Var1, Var2)) +
  geom_point(aes(color=fill),
             shape = 15,
             size = .35) +
  scale_color_identity()+
  guides(color = "none") +
  coord_sf(xlim = c(-180, 189),
           ylim = c(-60, 83.5),
           expand = FALSE) +
  theme_bw() +
  labs(x = "Longitude",
       y = "Latitude") +
  theme(axis.text=element_text(size=14),
```

```
        axis.title=element_text(size=14),
        panel.background = element_rect(fill = "gray45"),
        panel.grid.major = element_line(colour = "gray50"))


inset.figure <- ggplot(plot.dat.full,
                       aes(n, pi)) +
  geom_point(aes(color = fill),
             size = .5) +
  scale_color_identity() +
  labs(x = "Number of sampled species",
       y = expression("log(" * pi * ")")) +
  theme_bw()

png("FigureExport/mc1rGlobal.png",
    width=12,
    height = 9,
    units="in",
    res=500)
ggdraw(global.map + ggtitle("MC1R")) +
  draw_plot(plot = inset.figure,
            x = .06,
            y = .29,
            width = .21,
            height = .25)
dev.off()


#plot.dat.full <- plot.dat.full[plot.dat.full$pi < -4 &
#                                 plot.dat.full$pi >= -7.5,]
global.map.pi <- ggplot(plot.dat.full, aes(Var1, Var2)) +
  geom_point(aes(color=pi),
             shape = 15,
             size = .35) +
  scale_color_viridis(option="B")+
  #guides(color = "none") +
  coord_sf(xlim = c(-180, 189),
           ylim = c(-60, 83.5),
           expand = FALSE) +
  theme_bw() +
  labs(x = "Longitude",
       y = "Latitude",
       color = expression("log(" * pi * ")")) +
  theme(axis.text=element_text(size=14),
        axis.title=element_text(size=14),
        panel.background = element_rect(fill = "gray45"),
        panel.grid.major = element_line(colour = "gray50")) +
  ggtitle("MC1R")

png("FigureExport/mc1rGlobalPi.png",
    width=12,
    height = 9,
    units="in",
```

```
    res=500)
global.map.pi
dev.off()
```

4.4.6 rag1

```
plot.dat <- data.frame(Var1 = coords$Var1,
                       Var2 = coords$Var2,
                       pi = log(global.rag1.scores$meanPi),
                       n = global.rag1.scores$n)

plot.dat.full <- plot.dat %>%
  mutate(x = as.character(round(scales::rescale(pi), 2)),
         y = as.character(round(scales::rescale(n), 2))) %>%
  left_join(pal_d)


global.map <- ggplot(plot.dat.full, aes(Var1, Var2)) +
  geom_point(aes(color=fill),
             shape = 15,
             size = .35) +
  scale_color_identity()+
  guides(color = "none") +
  coord_sf(xlim = c(-180, 189),
           ylim = c(-60, 83.5),
           expand = FALSE) +
  theme_bw() +
  labs(x = "Longitude",
       y = "Latitude") +
  theme(axis.text=element_text(size=14),
        axis.title=element_text(size=14),
        panel.background = element_rect(fill = "gray45"),
        panel.grid.major = element_line(colour = "gray50"))


inset.figure <- ggplot(plot.dat.full,
                       aes(n, pi)) +
  geom_point(aes(color = fill),
             size = .5) +
  scale_color_identity() +
  labs(x = "Number of sampled species",
       y = expression("log(" * pi * ")")) +
  theme_bw()

png("FigureExport/rag1Global.png",
    width=12,
    height = 9,
    units="in",
    res=500)
ggdraw(global.map + ggtitle("RAG-1")) +
  draw_plot(plot = inset.figure,
            x = .06,
```

```
            y = .29,
            width = .21,
            height = .25)
dev.off()


plot.dat.full <- plot.dat.full[plot.dat.full$pi >= -4,]
global.map.pi <- ggplot(plot.dat.full, aes(Var1, Var2)) +
  geom_point(aes(color=pi),
             shape = 15,
             size = .35) +
  scale_color_viridis(option="B")+
  #guides(color = "none") +
  coord_sf(xlim = c(-180, 189),
           ylim = c(-60, 83.5),
           expand = FALSE) +
  theme_bw() +
  labs(x = "Longitude",
       y = "Latitude",
       color = expression("log(" * pi * ")")) +
  theme(axis.text=element_text(size=14),
        axis.title=element_text(size=14),
        panel.background = element_rect(fill = "gray45"),
        panel.grid.major = element_line(colour = "gray50")) +
  ggtitle("RAG-1")

png("FigureExport/rag1GlobalPi.png",
    width=12,
    height = 9,
    units="in",
    res=500)
global.map.pi
dev.off()
```

4.5 Spatial regression

First, we fit a non-spatial model to the data, then test for spatial autocorrelation in the residuals. This is combined in a function for simplicity.

```
autocorr.test <- function(x){

  where.missing.dat <- x$meanPi %>% is.na
  where.zero.pi <- x$meanPi == 0

  # Subset the data by those missing data results
  x <- x[where.missing.dat==FALSE & where.zero.pi==FALSE,]

  subset <- seq(1, nrow(x), length.out = 5000)
  x <- x[subset,]

  # Fit non-spatial model
  non_spatial_model <- lm(log(meanPi) ~ log(species.diversity) + log(n),
                          data=x)
  # Store the residuals
  x$resid <- resid(non_spatial_model)

  # Use Moran's I to test for spatial autocorrelation
  resid_sims <- simulateResiduals(non_spatial_model)
  testSpatialAutocorrelation(resid_sims,
                             x = x$Var1,
                             y = x$Var2,
                             plot = FALSE) %>% print
}


autocorr.test(cbind(global.co1.scores,
                    coords,
                    species.diversity))
```

```
##
##  DHARMa Moran's I test for distance-based autocorrelation
##
## data:  resid_sims
## observed = 0.30007785, expected = -0.00020004, sd = 0.00055059, p-value
## < 2.2e-16
## alternative hypothesis: Distance-based autocorrelation
```

```
autocorr.test(cbind(global.nd2.scores,
                    coords,
                    species.diversity))
```

```
##
##  DHARMa Moran's I test for distance-based autocorrelation
##
## data:  resid_sims
## observed = 0.19829329, expected = -0.00020004, sd = 0.00055542, p-value
## < 2.2e-16
## alternative hypothesis: Distance-based autocorrelation
```

```
autocorr.test(cbind(global.cytb.scores,
                    coords,
                    species.diversity))
```

```
##
##  DHARMa Moran's I test for distance-based autocorrelation
##
## data:  resid_sims
## observed = 0.19159993, expected = -0.00020004, sd = 0.00055820, p-value
## < 2.2e-16
## alternative hypothesis: Distance-based autocorrelation
```

```
autocorr.test(cbind(global.mt.scores,
                    coords,
                    species.diversity))
```

```
##
##  DHARMa Moran's I test for distance-based autocorrelation
##
## data:  resid_sims
## observed = 0.22607849, expected = -0.00020004, sd = 0.00055639, p-value
## < 2.2e-16
## alternative hypothesis: Distance-based autocorrelation
```

```
autocorr.test(cbind(global.mc1r.scores,
                    coords,
                    species.diversity))
```

```
##
##  DHARMa Moran's I test for distance-based autocorrelation
##
## data:  resid_sims
## observed = 0.26559883, expected = -0.00020004, sd = 0.00056023, p-value
## < 2.2e-16
## alternative hypothesis: Distance-based autocorrelation
```

```
autocorr.test(cbind(global.rag1.scores,
                    coords,
                    species.diversity))
```

```
##
##  DHARMa Moran's I test for distance-based autocorrelation
##
## data:  resid_sims
## observed = 0.13052734, expected = -0.00020004, sd = 0.00054125, p-value
## < 2.2e-16
## alternative hypothesis: Distance-based autocorrelation
```

Now we make a combined `data.frame` of the data from all the genes and fit the spatial model to the data.

```
# Make a data.frame of the data for analysis
spatial.dat <- rbind(global.nd2.scores,
                     global.co1.scores,
                     global.cytb.scores,
                     global.mt.scores,
                     global.mc1r.scores,
                     global.rag1.scores)

spatial.dat$gene <- c(rep("ND2", nrow(coords)),
                      rep("CO1", nrow(coords)),
                      rep("CYTB", nrow(coords)),
                      rep("MT", nrow(coords)),
                      rep("MC1R", nrow(coords)),
                      rep("RAG1", nrow(coords)))

spatial.dat$species.diversity <- rep(species.diversity, 6)

coords.full <- do.call("rbind", replicate(6, coords, simplify = FALSE))

spatial.dat <- cbind(spatial.dat, coords.full)

spatial.dat$gene <- as.factor(spatial.dat$gene)

# Find where we have missing data for pi scores
where.missing.dat <- spatial.dat$meanPi %>% is.na
where.zero.pi <- spatial.dat$meanPi == 0

# Subset the data by those missing data results
spatial.dat <- spatial.dat[where.missing.dat==FALSE & where.zero.pi==FALSE,]
#coords <- coords[where.missing.dat==FALSE & where.zero.pi==FALSE,]

spatial.dat$meanPi <- log(spatial.dat$meanPi)
spatial.dat$n <- log(spatial.dat$n)
spatial.dat$species.diversity <- log(spatial.dat$species.diversity)

# SubSet
ss <- seq(1, nrow(spatial.dat), length.out = 5000)
spatial.dat <- spatial.dat[ss,]

# Fit spatial model
m_spamm <- fitme(meanPi ~ species.diversity + n + gene +
                   Matern(1 | Var1 + Var2),
                 data = spatial.dat,
                 family = "gaussian",
                 control.HLfit=list(algebra="spprec"))

# Plot the effect of species diversity on pi
new.spatial.dat <- expand.grid(x = 5000, y = 5200,
                      species.diversity = seq(0, 7, length.out = 10),
                      n = seq(1, 6, length.out = 10),
                      gene = factor(c("ND2",
                                      "CO1",
                                      "CYTB",
```

```
                                    "MT",
                                    "MC1R",
                                    "RAG1")))

new.spatial.dat$meanPi <- predict(m_spamm, new.spatial.dat,
                                  re.form=NA) %>% as.numeric
new.spatial.dat$meanPi <- new.spatial.dat$meanPi + mean(c(0,fixef(m_spamm)[2:8]))


# get 95% confidence intervals around predictions
newdat <- cbind(new.spatial.dat, get_intervals(m_spamm,
                                    newdata = new.spatial.dat,
                                    intervals = "fixefVar", re.form = NA) +
                  mean(c(0,fixef(m_spamm)[2:8])))



diversity.pi.plot <- ggplot(spatial.dat, aes(x = species.diversity,
                                             y = meanPi)) +
  geom_point(size=.1) +
  geom_smooth(data = newdat,
              method="lm") +
  # showing trend line with no spatial autocorrelation
  #geom_smooth(data = spatial.dat,
  #            method = "lm",
  #            color = "red") +
  theme_bw() +
  labs(x = "log(species diversity)",
       y = expression("log(" * pi * ")"))

png("FigureExport/diversityPi.png",
    width = 5, height = 5, units="in", res=500)
diversity.pi.plot
dev.off()
```