Mass estimation of extinct taxa and phylogenetic hypothesis both influence analyses of character evolution in a large clade of birds (Telluraves) (Analysis workflow)

Contents

1 Data preparation **2**
1.1 Visualize distribution of the data . . . 3

2 Model selection **4**
2.1 phy.set.main . . . 4
2.2 phy.set.extant . . . 6
2.3 phy.set.constr . . . 6
2.4 phy.set.scale . . . 7

3 Make results table **9**
3.1 Mass distribution through time . . . 10
3.2 Model fit . . . 11
3.3 Morphological variance through time . . . 14

4 An assessment of model selection sensitivity **17**
4.1 Number of extinct taxa required for model selection . . . 17
4.2 Accurate recovery of post-K-Pg constraint model . . . 20

1 Data preparation

Load the required libraries for analysis.

```
library(ape)
library(geiger)
library(phytools)
library(mvMORPH)
library(dplyr)
library(ggplot2)
library(ggthemes)
library(qpcR)
library(viridis)
library(deeptime)
```

Phylogenetic data for these analyses comes from Crouch et al. (2019, MPE). There are four sets of phylogenetic data used. First, is from the principal analysis where no topological constraints were used in the analysis; these data are called `phy.set.main` here. Next, are data where the reconstruction was performed enforcing topological constraints such that the topology matched the most recent molecular estimates; these data are called `phy.set.constr`. Finally, we use data where the divergence estimates have been scaled to be entirely post-K-Pg; these data are called `phy.set.scale` here. Finally, `phy.set.extant` is the same phylogenetic data as `phy.set.main` but with only retaining extant taxa.

```
## Load phylogenetic data ##

phy.set.main <- read.nexus("phylo_set.nex")

phy.set.constr <- read.nexus("phylo_set_constrained.nex")

phy.set.scale <- read.nexus("phylo_set.scale.nex")
phy.set.scale <- lapply(phy.set.scale, multi2di)
class(phy.set.scale) <- "multiPhylo"

phy.set.extant <- read.nexus("phylo_set_extant.nex")
```

In this study, we use three different sets of morphological data. First, where the mean body mass of extant taxa is calculated from all available measurements. Standard error is calculated from all available measurements, and for those species with only one measurement available an error of 0.0345 is assigned. These data are assigned the abbreviation `morpho_full`. Next, we use the same body mass estimates as the first analysis, but where no standard error is assigned to extinct taxa. These data are assigned the abbreviation `no_se`. Finally, we use mass estimates for extinct taxa which were calculated using only the measurements with the highest coefficient of determination from Field et al. (2013). These data are assigned the abbreviation `highest_coef`. In the next section, we define each of these morphology sets, as well as their corresponding standard error vectors.

```
# Load weight data
data <- read.csv("body_weight_age_data_full_sets.csv")

# Define states vectors
full_states <- data$full_mean
names(full_states) <- data$X

no_se_states <- data$no_sampling_error_mean
names(no_se_states) <- data$X

highest_coef_mean <- data$highest_coef_mean
```

```
names(highest_coef_mean) <- data$X

# define standard error of measurements
full_states_se <- data$full_SE
names(full_states_se) <- data$X

no_se_states_se <- data$no_sampling_error_SE
names(no_se_states_se) <- data$X

highest_coef_states_se <- data$highest_coef_SE
names(highest_coef_states_se) <- data$X
```

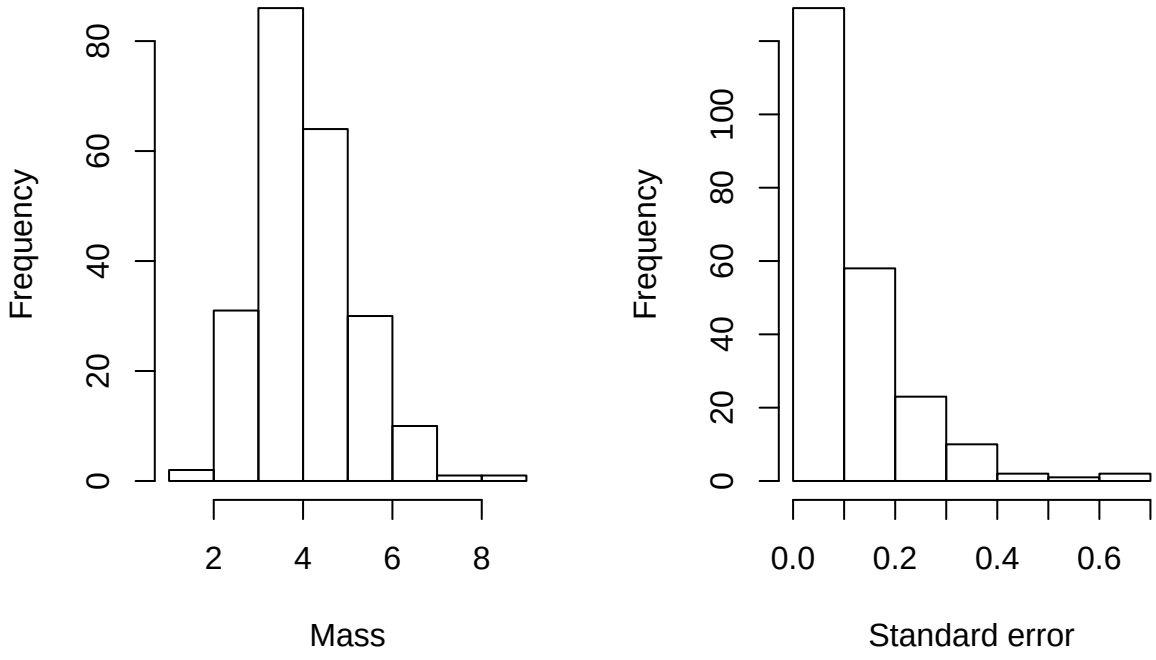
Finally, we define additional vectors containing only extant taxa. This does not need to be repeated for the other two analyses.

```
# Define vectors for analyses of extant taxa
states.extant <- full_states[names(full_states) %in% phy.set.extant[[1]]$tip.label]
states.se.extant <- full_states_se[names(full_states_se) %in% phy.set.extant[[1]]$tip.label]
```

1.1 Visualize distribution of the data

The mean weight data in this study is approximately normally distributed, but the standard error represented by each tip is extremely right skewed. This is because a few tips in the phylogeny represent a large number of extant species (see Table S3).

```
par(mfrow=c(1,2))
hist(full_states, xlab="Mass", main="")
hist(full_states_se, xlab="Standard error", main="")
```



We can also visualize the distribution of the data on a phylogeny sampled from the posterior distribution. Note, we added a background color in post-processing to help accentuate the colors.

```
example.phy <- read.nexus("example.phy.1.nex")

contObj <- contMap(example.phy, full_states, type = "fan", fsize=c(0.0001,1), plot=FALSE)

# Change color palette
n<-length(contObj$cols)
contObj$cols[1:n]<-viridis(n)

# Visualize
# plot(contObj, type="fan", fsize=c(0.5,1))

taxo <- read.csv("tree_taxonomy.csv")

unique.orders <- unique(taxo$taxo)
# Leptosomiformes (Cuckoo roller) is a monotypic order
unique.orders <- unique.orders[!unique.orders %in% c("FOSSIL","LEPTOSOMIFORMES")]

nodes <- vector(mode="numeric", length=length(unique.orders))
text <- vector(mode="character", length=length(unique.orders))

for(i in 1:length(unique.orders)){
  text[i] <- unique.orders[i] %>% as.character
  tips <- taxo[taxo$taxo==unique.orders[i], "tip"]
  nodes[i] <- findMRCA(tree = example.phy, tips = as.character(tips), type = "node")
}

 png("traitOnTree.png", width=8, height=8, units="in", res=500)
plot(contObj, type="fan", fsize=c(0.0000001,1), xlim=c(-160,160))
arc.cladelabels(text=text[1], node=nodes[1])
arc.cladelabels(text=text[2], node=nodes[2])
arc.cladelabels(text=text[3], node=nodes[3])
arc.cladelabels(text=text[4], node=nodes[4])
arc.cladelabels(text=text[5], node=nodes[5])
arc.cladelabels(text=text[6], node=nodes[6])
arc.cladelabels(text=text[7], node=nodes[7], lab.offset=1.12)
dev.off()
```

2 Model selection

The code to fit all of the models tested in this study is wrapped into a function, provided seperately. This code is loaded here:

```
source("fit.all.models.R")
```

We will now run all of the analyses for the three phylogenetic data sets respectively.

2.1 phy.set.main

```
# Principal morphology data set
main.result.full.list <- fit.all.models(multiPhy = phy.set.main,
                                        states = full_states,
```

```
                                    states.se = full_states_se,
                                    num.out = 50,
                                    shift.time = 66)

main.result.full.lnlik <- main.result.full.list[[1]]
write.csv(main.result.full.lnlik,
          file="output/main.result.full.lnlik.csv",
          row.names=FALSE)

main.result.full.aic <- main.result.full.list[[2]]
write.csv(main.result.full.aic,
          file="output/main.result.full.aic.csv",
          row.names=FALSE)

main.full.params <- main.result.full.list[[3]]
saveRDS(main.full.params, file="output/main.full.params.rds")

# No standard error morphology
main.result.no.se.list <- fit.all.models(multiPhy = phy.set.main,
                                    states = no_se_states,
                                    states.se = no_se_states_se,
                                    num.out = 50,
                                    shift.time = 66)

main.result.no.se.lnlik <- main.result.no.se.list[[1]]
write.csv(main.result.no.se.lnlik,
          file="output/main.result.no.se.lnlik.csv",
          row.names=FALSE)

main.result.no.se.aic <- main.result.no.se.list[[2]]
write.csv(main.result.no.se.aic,
          file="output/main.result.no.se.aic.csv",
          row.names=FALSE)

main.no.se.params <- main.result.no.se.list[[3]]
saveRDS(main.no.se.params, file="output/main.no.se.params.rds")

# Highest coefficient of determination
main.result.high.coef.list <- fit.all.models(multiPhy = phy.set.main,
                                    states = highest_coef_mean,
                                    states.se = highest_coef_states_se,
                                    num.out = 50,
                                    shift.time = 66)

main.result.high.coef.lnlik <- main.result.high.coef.list[[1]]
write.csv(main.result.high.coef.lnlik,
          file="output/main.result.high.coef.lnlik.csv",
          row.names=FALSE)

main.result.high.coef.aic <- main.result.high.coef.list[[2]]
write.csv(main.result.high.coef.aic,
          file="output/main.result.high.coef.aic.csv",
          row.names=FALSE)
```

```
main.high.coef.params <- main.result.high.coef.list[[3]]
saveRDS(main.high.coef.params, file="output/main.high.coef.params.rds")
```

2.2 phy.set.extant

Analysis of only extant taxa

```
extant.result.list <- fit.all.models(multiPhy = phy.set.extant,
                                     states = states.extant,
                                     states.se = states.se.extant,
                                     num.out = 50,
                                     shift.time = 66)

extant.result.lnlik <- extant.result.list[[1]]
write.csv(extant.result.lnlik, file="output/extant.result.lnlik.csv", row.names=FALSE)

extant.result.aic <- extant.result.list[[2]]
write.csv(extant.result.aic, file="output/extant.result.aic.csv", row.names=FALSE)

extant.params <- extant.result.list[[3]]
saveRDS(extant.params, file="output/extant.params.rds")
```

2.3 phy.set.constr

Analysis using phylogenetic data with topological constraints

```
# Principal morphology data set
constrain.result.full.list <- fit.all.models(multiPhy = phy.set.constr,
                                     states = full_states,
                                     states.se = full_states_se,
                                     num.out = 50,
                                     shift.time = 66)

constrain.result.full.lnlik <- constrain.result.full.list[[1]]
write.csv(constrain.result.full.lnlik,
          file="output/constrain.result.full.lnlik.csv",
          row.names=FALSE)

constrain.result.full.aic <- constrain.result.full.list[[2]]
write.csv(constrain.result.full.aic,
          file="output/constrain.result.full.aic.csv",
          row.names=FALSE)

constrain.full.params <- constrain.result.full.list[[3]]
saveRDS(constrain.full.params, file="output/constrain.full.params.rds")

# No standard error morphology
constrain.result.no.se.list <- fit.all.models(multiPhy = phy.set.constr,
                                     states = no_se_states,
                                     states.se = no_se_states_se,
                                     num.out = 50,
                                     shift.time = 66)
```

```
constrain.result.no.se.lnlik <- constrain.result.no.se.list[[1]]
write.csv(constrain.result.no.se.lnlik,
          file="output/constrain.result.no.se.lnlik.csv",
          row.names=FALSE)

constrain.result.no.se.aic <- constrain.result.no.se.list[[2]]
write.csv(constrain.result.no.se.aic,
          file="output/constrain.result.no.se.aic.csv",
          row.names=FALSE)

constrain.no.se.params <- constrain.result.no.se.list[[3]]
saveRDS(constrain.no.se.params, file="output/constrain.no.se.params.rds")

# Highest coefficient of determination
constrain.result.high.coef.list <- fit.all.models(multiPhy = phy.set.constr,
                                   states = highest_coef_mean,
                                   states.se = highest_coef_states_se,
                                   num.out = 50,
                                   shift.time = 66)

constrain.result.high.coef.lnlik <- constrain.result.high.coef.list[[1]]
write.csv(constrain.result.high.coef.lnlik,
          file="output/constrain.result.high.coef.lnlik.csv",
          row.names=FALSE)

constrain.result.high.coef.aic <- constrain.result.high.coef.list[[2]]
write.csv(constrain.result.high.coef.aic,
          file="output/constrain.result.high.coef.aic.csv",
          row.names=FALSE)

constrain.high.coef.params <- constrain.result.high.coef.list[[3]]
saveRDS(constrain.high.coef.params,
        file="output/constrain.high.coef.params.rds")
```

2.4 phy.set.scale

Analysis of phylogentic data scaled such that the divergence dates are entirely Cenozoic. The workflow for deriving these phylogenies, as well as the `fossil_treePL` function used, are detailed seperately. Note, the *shift* models are NOT fit to these phylogenetic data.

```
source("fit.Cenozoic.models.R")

# Principal morphology data set
scale.result.full.list <- fit.all.models(multiPhy = phy.set.scale,
                                   states = full_states,
                                   states.se = full_states_se,
                                   num.out = 50,
                                   shift.time = 66)

scale.result.full.lnlik <- scale.result.full.list[[1]]
write.csv(scale.result.full.lnlik,
          file="output/scale.result.full.lnlik.csv",
```

```
          row.names=FALSE)

scale.result.full.aic <- scale.result.full.list[[2]]
write.csv(scale.result.full.aic,
          file="output/scale.result.full.aic.csv",
          row.names=FALSE)

scale.full.params <- scale.result.full.list[[3]]
saveRDS(scale.full.params, file="output/scale.full.params.rds")

# No standard error morphology
scale.result.no.se.list <- fit.all.models(multiPhy = phy.set.scale,
                                  states = no_se_states,
                                  states.se = no_se_states_se,
                                  num.out = 50,
                                  shift.time = 66)

scale.result.no.se.lnlik <- scale.result.no.se.list[[1]]
write.csv(scale.result.no.se.lnlik,
          file="output/scale.result.no.se.lnlik.csv",
          row.names=FALSE)

scale.result.no.se.aic <- scale.result.no.se.list[[2]]
write.csv(scale.result.no.se.aic,
          file="output/scale.result.no.se.aic.csv",
          row.names=FALSE)

scale.no.se.params <- scale.result.no.se.list[[3]]
saveRDS(scale.no.se.params, file="output/scale.no.se.params.rds")

# Highest coefficient of determination
scale.result.high.coef.list <- fit.all.models(multiPhy = phy.set.scale,
                                  states = highest_coef_mean,
                                  states.se = highest_coef_states_se,
                                  num.out = 50,
                                  shift.time = 66)

scale.result.high.coef.lnlik <- scale.result.high.coef.list[[1]]
write.csv(scale.result.high.coef.lnlik,
          file="output/scale.result.high.coef.lnlik.csv",
          row.names=FALSE)

scale.result.high.coef.aic <- scale.result.high.coef.list[[2]]
write.csv(scale.result.high.coef.aic,
          file="output/scale.result.high.coef.aic.csv",
          row.names=FALSE)

scale.high.coef.params <- scale.result.high.coef.list[[3]]
saveRDS(scale.high.coef.params,
        file="output/scale.high.coef.params.rds")
```

3 Make results table

Before we can generate the results table, we first need to load in all of the model fitting results.

```
## Load the results data
main.results.full <- read.csv("output/main.result.full.aic.csv")
main.results.no.se <- read.csv("output/main.result.no.se.aic.csv")
main.results.high.coef <- read.csv("output/main.result.high.coef.aic.csv")

constr.results.full <- read.csv("output/constrain.result.full.aic.csv")
constr.results.no.se <- read.csv("output/constrain.result.no.se.aic.csv")
constr.results.high.coef <- read.csv("output/constrain.result.high.coef.aic.csv")

scale.results.full <- read.csv("output/scale.result.full.aic.csv")
scale.results.no.se <- read.csv("output/scale.result.no.se.aic.csv")
scale.results.high.coef <- read.csv("output/scale.result.high.coef.aic.csv")

extant.results <- read.csv("output/extant.result.aic.no.trend.csv")
```

Now we can use the function *make.results.table* to summarize the model fitting results.

```
# Generate results tables
source("make.results.table.R")

m.r.f <- make.results.table(main.results.full)
m.r.no.se <- make.results.table(main.results.no.se)
m.r.high.coef <- make.results.table(main.results.high.coef)

c.r.f <- make.results.table(constr.results.full)
c.r.no.se <- make.results.table(constr.results.no.se)
c.r.high.coef <- make.results.table(constr.results.high.coef)

# No split models applied to scaled phylogenetic data
wanted.cols <- c ("Brownian.Motion", "OU", "Early.burst", "Trend", "White.noise")

scale.results.full.trim <- scale.results.full[,colnames(scale.results.full) %in% wanted.cols ]
scale.results.no.se.trim <- scale.results.no.se[,colnames(scale.results.no.se) %in% wanted.cols ]
scale.results.high.trim <- scale.results.high.coef[,colnames(scale.results.high.coef) %in% wanted.cols ]


s.r.f <- make.results.table(scale.results.full.trim)
s.r.no.se <- make.results.table(scale.results.no.se.trim)
s.r.high.coef <- make.results.table(scale.results.high.trim)

e.r <- make.results.table(extant.results)
```

Generate figures

3.1 Mass distribution through time

The first figure we will generate visualizes the range in body sizes for extant and extinct taxa through time compared between the two approaches for estimating mass. We manually added a legend in post-processing.

```
no.extant <- data[data$Age != 0,]
extant.only <- data[data$Age == 0,]

age.distr.plot <- ggplot(data, aes(x = Age,
                                   y = full_mean)) +
  #geom_point() +
  labs(x = "Age (Ma)",
       y = "log(mass)") +
  theme_bw() +
    theme(legend.position="top",
          panel.grid=element_blank(),
          axis.text=element_text(size=12),
          axis.title=element_text(size=12)) +
  geom_pointrange(data=extant.only,
                  mapping=aes(x=Age,
                              y=full_mean,
                              ymin = full_mean - full_SE,
                              ymax = full_mean + full_SE),
                  shape = 19,
                  fill = "white",
                  alpha = 0.5,
                  position = 'jitter') +
  geom_pointrange(data=no.extant,
                  mapping=aes(x=Age,
                              y=full_mean,
                              ymin = full_mean - full_SE,
                              ymax = full_mean + full_SE),
                  shape = 19,
                  color = "#fc8d62",
                  alpha = 0.9) +
  geom_pointrange(data=no.extant,
                  mapping=aes(x=Age,
                              y = highest_coef_mean,
                              ymin = highest_coef_mean - highest_coef_SE,
                              ymax = highest_coef_mean + highest_coef_SE),
                  shape = 19,
                  color = "#8da0cb",
                  alpha = 0.9,
                  position = position_jitter(width = 1)) +
  xlim(c(66,0)) +
  geom_vline(xintercept = 66, linetype = "dashed")



png("age_distribution_plot.png", width=10, height=6, units="in", res=500)
gggeo_scale(age.distr.plot,
```

```
            dat="epochs",
            abbrv = FALSE,
            skip=c("Quaternary", "Holocene",
                   "Pleistocene", "Pliocene"))
```

```
## Warning: Removed 78 rows containing missing values (geom_pointrange).
```

```
dev.off()
```

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

3.2 Model fit

First, we will generate a plot showing estimates of model fit for the different analyses. This involves calculating Akaike Weights for each of the `data.frames` of AIC scores.

```
## Calculate Akaike weights
# Simplify the pcR function for our purposes
only.weights <- function(x){
  return(akaike.weights(x)$weights)
}

main.aicw.full <- apply(main.results.full, 1, only.weights)
main.aicw.no.se <- apply(main.results.no.se, 1, only.weights)
main.aicw.high.coef <- apply(main.results.high.coef, 1, only.weights)

constr.aicw.full <- apply(constr.results.full, 1, only.weights)
constr.aicw.no.se <- apply(constr.results.no.se, 1, only.weights)
constr.aicw.high.coef <- apply(constr.results.high.coef, 1, only.weights)

scale.aicw.full <- apply(scale.results.full, 1, only.weights)
scale.aicw.no.se <- apply(scale.results.no.se, 1, only.weights)
scale.aicw.high.coef <- apply(scale.results.high.coef, 1, only.weights)

extant.aicw <- apply(extant.results, 1, only.weights)
```

```
# A utility function to convert the organization of the
# results prior to plotting
source("convert.df.R")


main.aicw.full.df <- main.aicw.full %>% t %>% melt
colnames(main.aicw.full.df) <- c("Itt", "Model", "AICw")

main.a

ggplot(main.aicw.full.df, aes(x=Var1, y=value, fill=Var2)) +
  geom_area(alpha=0.6 , size=1, colour="black") +
  labs(x = "Iteration",
       y = "Akaike weight")

# Perform data transformations
main.aicw.full.df <- convert.df(main.aicw.full,
                                grouping = "main")
```

```
main.aicw.no.se.df <- convert.df(main.aicw.no.se,
                                 grouping = "main")
main.aicw.high.coef.df <- convert.df(main.aicw.high.coef,
                                     grouping = "main")

main.aicw.full.df$morpho <- "All data"
main.aicw.no.se.df$morpho <- "No incorporation of intraspecific error"
main.aicw.high.coef.df$morpho <- "Only measurement with greatest predicitive power"

constr.aicw.full.df <- convert.df(constr.aicw.full,
                              grouping = "constrain")
constr.aicw.no.se.df <- convert.df(constr.aicw.no.se,
                              grouping = "constrain")
constr.aicw.high.coef.df <- convert.df(constr.aicw.high.coef,
                              grouping = "constrain")

constr.aicw.full.df$morpho <- "All data"
constr.aicw.no.se.df$morpho <- "No incorporation of intraspecific error"
constr.aicw.high.coef.df$morpho <- "Only measurement with greatest predicitive power"

scale.aicw.full.df <- convert.df(scale.aicw.full,
                             grouping = "scale")
scale.aicw.no.se.df <- convert.df(scale.aicw.no.se,
                             grouping = "scale")
scale.aicw.high.coef.df <- convert.df(scale.aicw.high.coef,
                             grouping = "scale")

scale.aicw.full.df$morpho <- "All data"
scale.aicw.no.se.df$morpho <- "No incorporation of intraspecific error"
scale.aicw.high.coef.df$morpho <- "Only measurement with greatest predicitive power"

extant.aicw.df <- convert.df(extant.aicw, grouping = "extant")

model.supp.plot.dat <- rbind(main.aicw.full.df,
                             main.aicw.no.se.df,
                             main.aicw.high.coef.df,
                             constr.aicw.full.df,
                             constr.aicw.no.se.df,
                             constr.aicw.high.coef.df,
                             scale.aicw.full.df,
                             scale.aicw.no.se.df,
                             scale.aicw.high.coef.df)

# re-order the factor 'grouping'
model.supp.plot.dat$grouping <- factor(model.supp.plot.dat$grouping,
                                       levels=c("main",
                                                "constrain",
                                                "scale",
                                                "extant"))

# a split BM model was accidentally duplicated, its removed here
v <- model.supp.plot.dat$model=="Shift.Brownian"
model.supp.plot.dat <- model.supp.plot.dat[v==FALSE,]
```

```
#
v <- extant.aicw.df$model=="Shift.Brownian"
extant.aicw.df <- extant.aicw.df[v==FALSE,]

# A text string for the tick labels of the boxplot
x.axis.labels <- c("Ecological\nrelease",
                   "Release \nand radiate",
                   "Ecological \nconstraint",
                   "Radiate \nand constraint",
                   "OU shift",
                   "BM shift",
                   "BM",
                   "OU",
                   "EB",
                   "Trend",
                   "White noise")

## Make the plot

model.supp.plot <- ggplot(model.supp.plot.dat,
                          aes(x=model,
                              y=weights,
                              fill=grouping))+
  geom_boxplot() +
  theme_bw() +
  theme(legend.position="top",
        panel.grid=element_blank(),
        axis.text=element_text(size=14),
        axis.title=element_text(size=14),
        legend.text=element_text(size=16),
        legend.title = element_text(size=16),
        strip.text = element_text(size = 14,
                                  face="bold")) +
  labs(y="Akaike weight", x="Evolutionary model", fill="Analysis") +
  facet_wrap(~morpho, nrow = 3) +
  scale_x_discrete(labels=x.axis.labels)

cols <- colorblind_pal()(5)[2:5]

#png("modelSupport.png", width=15,height=15, units="in", res=500)
model.supp.plot +
  scale_fill_manual(values=cols[1:3]) +
  theme(legend.key.size = unit(3,"line"))
#dev.off()



x.axis.labels.extant <- c("Ecological\nrelease",
                          "Release \nand radiate",
                          "Ecological \nconstraint",
                          "Radiate \nand constraint",
                          "OU shift",
                          "BM shift",
```

```
                          "BM",
                          "OU",
                          "EB",
                          "White noise")

extant.model.supp <- ggplot(extant.aicw.df,
                            aes(x=model,
                                y=weights,
                                fill=grouping))+
  geom_boxplot() +
  theme_bw() +
  theme(legend.position="none",
        panel.grid=element_blank(),
        axis.text=element_text(size=14),
        axis.title=element_text(size=14),
        strip.text = element_text(size = 14,
                                  face="bold")) +
  labs(y="Akaike weight", x="Evolutionary model", fill="Analysis") +
  scale_x_discrete(labels=x.axis.labels.extant)

#png("extant_model_support.png", width = 15, height = 5, units="in", res=500)
extant.model.supp
#dev.off()
```

3.3 Morphological variance through time

In this section we simulate a number of traits under the estimated parameters from a selection of the best fitting models, and calculate the variance in the simulated traits through time to see the timing of diversification in body size.

```
# mvMORPH::mvSIM doesn't retain internal nodes
# phylopars::simtraits doesn't retain internal

source("make.var.through.time.R")

all.sim.res <- vector(mode="list", length=3*50)

where.to.store <- 1


# Simulate traits for a BMshift model

for(i in 1:length(phy.set.main)){

  phy <- phy.set.main[[i]]
  tree.height <- max(nodeHeights(phy))
  shift.point <- tree.height - shift.time
  tree <- make.era.map(phy, c(0, shift.point, tree.height))

  traitData <- sim.rates(tree = tree,
                  sig2 = c(0.003, 0.027),
                  anc = 4.08,
                  nsim = 1,
```

```
                    internal = TRUE)

  data.conv <- make.var.through.time(phy=phy.set.main[[i]],
                                     traitData = traitData)

  # store
  data.store <- cbind(data.conv[,c(1,4)],"BMshift","main")
  colnames(data.store) <- c("age", "variance", "model", "analysis")

  all.sim.res[[where.to.store]] <- data.store

  where.to.store <- where.to.store + 1
}

## extant
for(i in 1:length(phy.set.extant)){
  traitData <- fastBM(tree = phy.set.extant[[i]],
                      a = 4.25,
                    sig2 = 0.019,
                    nsim=1,
                    internal = TRUE)

  data.conv <- make.var.through.time(phy=phy.set.extant[[i]],
                                     traitData = traitData)
  # store
  data.store <- cbind(data.conv[,c(1,4)],"BM","extant")
  colnames(data.store) <- c("age", "variance", "model", "analysis")

  all.sim.res[[where.to.store]] <- data.store

  where.to.store <- where.to.store + 1
}

## constrain

for(i in 1:length(phy.set.constr)){

  phy <- phy.set.constr[[i]]
  tree.height <- max(nodeHeights(phy))
  shift.point <- tree.height - shift.time
  tree <- make.era.map(phy, c(0, shift.point, tree.height))

  traitData <- sim.rates(tree = tree,
                    sig2 = c(0.008, 0.011),
                    anc = 4.09,
                    nsim = 1,
                    internal = TRUE)

  data.conv <- make.var.through.time(phy=phy.set.constr[[i]],
                                     traitData = traitData)

  # store
  data.store <- cbind(data.conv[,c(1,4)],"BMshift","constrain")
```

```
  colnames(data.store) <- c("age", "variance", "model", "analysis")

  all.sim.res[[where.to.store]] <- data.store

  where.to.store <- where.to.store + 1
}

## Combine results
sim.variances <- do.call(rbind, all.sim.res)

saveRDS(sim.variances, file="output/simVariances.RDS")
```

We can now make the plot

```
# re-order the factor 'analysis'
sim.variances$analysis <- factor(sim.variances$analysis,
                                 levels=c("main",
                                          "constrain",
                                          "extant"))
xlimMax <- max(sim.variances$age)

var.plot <- ggplot(sim.variances,
                   aes(x=age,
                       y=variance,
                       color=analysis)) +
  geom_vline(xintercept = 66, linetype="longdash") +
  geom_smooth() +
  theme_bw() +
  xlim(c(70, 0)) +
  theme(legend.position="right",
        panel.grid=element_blank(),
        axis.text=element_text(size=12),
        axis.title=element_text(size=12)) +
  labs(y="Variance", x="Age (Ma)", color="Analysis")

 png("varianceThroughTime.png", width=10, height=6, units="in", res=500)
var.plot + scale_color_colorblind()
 dev.off()
```

4 An assessment of model selection sensitivity

4.1 Number of extinct taxa required for model selection

The best fitting model from the main analysis is a model of radiation followed by post-K-Pg constraint, and an early burst model when only extant taxa are analyzed. In this section we evaluate how many extinct taxa are required for model selection to change. We will sample every 10% of the number of extinct species included, and run each sampling interval 50 times. For these analyses, we will run on a single randomly selected phylogeny from `phy.set.main` and `phy.set.constrain`. At each iteration of the analysis we will record the Akaike weight for the post-K-Pg constraint model.

```
set.seed(8)
s <- sample(seq(1,50,1),1)

single.main <- phy.set.main[[s]]
single.constr <- phy.set.constr[[s]]

# The number of species to include at each iteration
number.of.species.to.include <- c(8,17,26,35,44,53,62,71,80)

# Define vectors of extinct and extant species
source("fossil_species.R")

fossil.spp.present <- fossil.spp[(fossil.spp %in% single.main$tip.label) == TRUE]

# Define vector of living species
extant.species <- single.main$tip.label[!(single.main$tip.label %in% fossil.spp)]


# Define a data.frame to store the results
sample.res <- matrix(NA, ncol=3, nrow=1) %>% as.data.frame
colnames(sample.res) <- c("weight", "propTaxa", "tree")

for(i in 1:length(number.of.species.to.include)){

    for(k in 1:50){

    fossil.sample <- sample(fossil.spp.present,
                            number.of.species.to.include[i],
                            replace=F)

    run.spp <- c(fossil.sample, extant.species)

    run.states <- states[(names(states) %in% run.spp) == TRUE]
    run.se <- states.se[(names(states.se) %in% run.spp) == TRUE]

    EB.model.m <- fitContinuous(phy =  single.main,
                                dat = run.states,
                                SE = run.se,
                                model = "EB")
    EB.model.c <- fitContinuous(phy =  single.constr,
                                dat = run.states,
                                SE = run.se,
                                model = "EB")
```

```
  v <- single.main$tip.label %in% names(run.states)
  drop <- single.main$tip.label[v==FALSE]

  tree.m <- drop.tip(single.main, drop)
  tree.c <- drop.tip(single.constr, drop)

  # Create a simmap phylo
tree.height.m <- max(nodeHeights(single.main))
tree.height.c <- max(nodeHeights(single.constr))

shift.point.m <- tree.height.m - 66
shift.point.c <- tree.height.c - 66

tree.m <- make.era.map(tree.m,
                       c(0,shift.point.m, tree.height.m))
tree.c <- make.era.map(tree.c,
                       c(0,shift.point.c, tree.height.c))

  postConstr.model.m <- mvSHIFT(tree = tree.m,
                                data = run.states,
                                error = run.se,
                                model = "RC")

  postConstr.model.c <- mvSHIFT(tree = tree.c,
                                data = run.states,
                                error = run.se,
                                model = "RC")


  # Model fit
  EB.aicc.m <- EB.model.m$opt$aicc
  EB.aicc.c <- EB.model.c$opt$aicc
RC.aicc.m <- postConstr.model.m$AICc
  RC.aicc.c <- postConstr.model.c$AICc

  main.weights <- akaike.weights(c(RC.aicc.m,
                                   EB.aicc.m))$weights
  constr.weights <- akaike.weights(c(RC.aicc.c,
                                     EB.aicc.m))$weights

  bind <- matrix(NA, ncol=3, nrow=2) %>% as.data.frame
  colnames(bind) <- c("weight", "propTaxa", "tree")

  bind[1,1] <- main.weights[1]
  bind[2,1] <- constr.weights[1]
  bind[1,2] <- bind[2,2] <- length(fossil.sample) / 89
  bind[1,3] <- "main"
  bind[2,3] <- "constrain"

  sample.res <- rbind(sample.res, bind)


  }
```

```
}

# remove the first row of NA's
sample.res <- sample.res[-1,]

write.csv(sample.res, file="results/sample.res.csv", row.names = F)
```

Now, we will summarize the results and visualize the results.

```
## Mean values
mean.vals <- by(sample.res[,1], sample.res[,2:3], mean)

main.mean <- cbind(mean.vals[,2], colnames(mean.vals)[2]) %>% as.data.frame
colnames(main.mean) <- c("weight", "tree")
main.mean$sample.rate <- seq(0.1,0.9,0.1)
rownames(main.mean) <- NULL
main.mean$weight <- as.numeric(levels(main.mean$weight))[main.mean$weight]

constr.mean <- cbind(mean.vals[,1], colnames(mean.vals)[1]) %>% as.data.frame
colnames(constr.mean) <- c("weight", "tree")
constr.mean$sample.rate <- seq(0.1,0.9,0.1)
rownames(constr.mean) <- NULL
constr.mean$weight <- as.numeric(levels(constr.mean$weight))[constr.mean$weight]

all.mean <- rbind(main.mean, constr.mean)


## Standard deviation values

sd.vals <- by(sample.res[,1], sample.res[,2:3], sd)

mean.sd.vals <- sd.vals[,2]
mean.vals.pos <- main.mean$weight + mean.sd.vals
mean.vals.neg <- main.mean$weight - mean.sd.vals

constr.sd.vals <- sd.vals[,1]
constr.vals.pos <- constr.mean$weight + constr.sd.vals
constr.vals.neg <- constr.mean$weight - constr.sd.vals

all.mean$low <- c(mean.vals.neg, constr.vals.neg)
all.mean$high <- c(mean.vals.pos, constr.vals.pos)

## Generate figure
sampling.plot <- ggplot(all.mean, aes(x=sample.rate,
                                      y=weight,
                                      color=tree)) +
  geom_errorbar(data=all.mean, mapping=aes(x=sample.rate,
                      ymin=low,
                      ymax=high,
                      color=tree),
                width=0.01,
                size=1,
                position=position_dodge(width=0.01)) +
  geom_point(position=position_dodge(width=0.01)) +
```

```
  theme_bw() +
  geom_hline(yintercept=0.50, linetype="dashed") +
  labs(x="Proportion extinct species sampled",
       y="Support for post-K-Pg constraint model")+
  #ylim(c(0,1))+
    theme(legend.position="right",
        panel.grid=element_blank(),
        axis.text=element_text(size=12),
        axis.title=element_text(size=12))


#png("samplingPlot.png", width=8, height=6, units="in", res=500)
sampling.plot + scale_color_colorblind()
#dev.off()
```

4.2 Accurate recovery of post-K-Pg constraint model

One potential issue with these data is that, due to the absence of fossils from the Cretaceous and immediately following the K-Pg boundary, the ability of phylogenetic models to correctly infer the underlying process of evolution may be hindered. Specifically, if the deepest branches in the phylogeny conserve sufficient phylogenetic signal an OU process may be mis-identified as a post-K-Pg-constraint process. In this section we test whether simulating under an OU process (with a range of parameters) results in incorrect recovery of a model of post-K-Pg constraint.

First, we simulate traits with increasing strength of the α parameter (strength of selection back to the optimum) under ten different rates of evolution σ^2.

```
alpha.vals <-seq(0.001, 9, length.out=10)

sigma.vals <- seq(0.01, 3, length.out=10)

root.state <- 5

sim.res.data <- matrix(NA, ncol=5,nrow=50*length(alpha.vals)*5) %>% as.data.frame
colnames(sim.res.data) <- c("alpha", "sigma", "postConstrain", "OU", "BM")

itt <- 1

for(j in 1:length(sigma.vals)){
  sigma <- sigma.vals[j]

  for(k in 1:length(alpha.vals)){

    alpha.val <- alpha.vals[k]

    for(i in 1:50){
      tryCatch({
      tree <- phy.set.main[[i]]
      # Simulate
      traits <- mvSIM(tree=tree,
                      nsim=1,
                      model="OU1",
                      param=list(theta=root.state,
```

```
                                sigma=sigma,
                                alpha=alpha.val))

      traits.vector <- traits[,1]
      names(traits.vector) <- rownames(traits)
      # Model fit and store parameters
      sim.res.data[itt,1] <- alpha.val
      sim.res.data[itt,2] <- sigma
      ou.fit <- fitContinuous(tree, traits.vector, model="OU")
      bm.fit <- fitContinuous(tree, traits.vector, model="BM")

      tree.height <- max(nodeHeights(tree))
      shift.point <- tree.height - 66
      simmap.tree <- make.era.map(tree, c(0,shift.point, tree.height))

      shift.fit <- mvSHIFT(simmap.tree, traits.vector, model="EC")

      sim.res.data[itt,3] <- shift.fit$AICc
      sim.res.data[itt,4] <- ou.fit$opt$aicc
      sim.res.data[itt,5] <- bm.fit$opt$aicc

      itt <- itt+1
      write.csv(sim.res.data,file="sim.res.data.csv", row.names = F)
      },error=function(e){})
    }
  }
}
```

Next we calculate the Akaike weight of each iteration of the simulation to estimate model support.

```
only.aicc <- sim.res.data[,3:5]

# shorten to the function from qpcR function to only return 'weights'
simple.weights <- function(x){
  return(akaike.weights(x)$weights)
}

aic.w.scores <- apply(only.aicc, 1, simple.weights) %>% t

sim.plot.dat <- cbind(aic.w.scores, sim.res.data[,1:2])

sim.plot.dat <- rbind(data.frame(weight = aic.w.scores[,1],
                                 alpha = sim.res.data[,1],
                                 sigma = sim.res.data[,2],
                                 model = "postConstrain"),
                      data.frame(weight = aic.w.scores[,2],
                                 alpha = sim.res.data[,1],
                                 sigma = sim.res.data[,2],
                                 model = "OU"),
                      data.frame(weight = aic.w.scores[,3],
                                 alpha = sim.res.data[,1],
                                 sigma = sim.res.data[,2],
                                 model = "BM"))
```

In this document we report the values of all of the values of sigma tested at (10 levels). Each of these values

gives a consistent result:

```
sim.plot <- ggplot(sim.plot.dat, aes(x=alpha, y=weight, color=factor(model))) +
   geom_line(aes(linetype=factor(sigma)))+
#  geom_line() +
  geom_point() +
  theme_bw() +
 # facet_grid(cols = vars(sigma)) +
  theme(legend.position="right",
        panel.grid=element_blank(),
        axis.text=element_text(size=12),
        axis.title=element_text(size=12))

sim.plot
```

For the main main manuscript we report a subset of these results to aid the clarity of the figure. We also perform an additional step to report the mean and standard deviation for each model at each alpha value.

```
# Subset the data by sigma values
sim.plot.dat.subset <- sim.plot.dat

sim.plot.dat.subset$alpha <- round(sim.plot.dat.subset$alpha, 3)
sim.plot.dat.subset$sigma <- round(sim.plot.dat.subset$sigma, 3)

#sim.plot.dat.subset$sigma <- factor(sim.plot.dat.subset$sigma)

sig.vals.keep <- c(0.01, 0.674, 1.339, 2.336, 3)

sim.plot.dat.subset <- sim.plot.dat.subset[sim.plot.dat.subset$sigma%in%sig.vals.keep,]

# Calculate means and standard deviations

# mean.vals <- with(sim.plot.dat.subset,
#              tapply(weight, list("sigma"=sigma,
#                                  "model"=model,
#                                  "alpha"=alpha), mean))

mean.d <- by(sim.plot.dat.subset$weight, sim.plot.dat.subset[,2:4],mean)

sd.d <- by(sim.plot.dat.subset$weight, sim.plot.dat.subset[,2:4],sd)

# A function to summarize the array from this study
source("summarize.array.R")

sumd.data <- summarize.array(mean.d)

sd.data <- summarize.array(sd.d)

bar.data <- cbind(sumd.data, sd.data$variable)
colnames(bar.data)[5] <- "sd"
bar.data$lower <- bar.data$variable - bar.data$sd
bar.data$upper <- bar.data$variable + bar.data$sd
```

Now we create the figure for the main manuscript

```
sim.plot2 <- ggplot(sumd.data,
                    aes(x=alpha, y=variable, color=factor(model))) +
  geom_errorbar(data=bar.data,
                mapping=aes(x=alpha, ymin=upper, ymax=lower, color=factor(model)),
                width=0.1, size=0.5) +
   geom_line(aes(linetype=factor(sigma)))+
#  geom_line() +
  geom_point() +
  theme_bw() +
 # facet_grid(cols = vars(sigma)) +
  theme(legend.position="right",
        panel.grid=element_blank(),
        axis.text=element_text(size=12),
        axis.title=element_text(size=12)) +
  labs(y="Akaike weight", x=expression(alpha),
       linetype=expression(sigma^2),
       color="Model")

## Define a colorblind friendly palette with values
## not used before

cc <- colorblind_pal()(8)

new.cols <- c("gray", cc[6], cc[8])

png("simPlot.png", width=8, height=6,units="in", res=500)
# sim.plot2 + scale_color_colorblind()
sim.plot2 + scale_color_manual(values=new.cols)
dev.off()
```