Time calibrated phylogeny (incl. diagnostics)

Beatriz Willink Rachel Blow

May 2020

#StarBEAST2 species trees file output

Building the tree (Fig. 2)

Set your working directory to the parent directory containing all of your MCMC analyses

```
knitr::opts_knit$set(root.dir = '..' )
```

Call "treeio" and "ggtree" packages for reading and writing trees

```
require(treeio)
require(ggtree)
```

Read in the summary tree

```
beast_tree <- read.beast("StarBEAST2/summary.tree")
```

Rename taxa for displaying on tree

```
relabelling <- data.frame("label" = fortify(beast_tree)$label[1:41])
relabelling$label2 <- sub("_", "␣", relabelling$label)
relabelling$label2 <- plyr::revalue(relabelling$label2,
                                    c("Amorphostigma␣armstrongi" = "Ischnura␣
                                       armstrongi",
                                      "Amorphostigma␣sp." = "Ischnura␣sp.",
                                      "Rhodischnura␣nursei" = "Ischnura␣
                                         nursei"))

beast_tree <- full_join(beast_tree, relabelling, by ="label")
```

Call "ggplot2" for additional plotting capability

```
require(ggplot2)
```

Create time-calibrated phylogeny displaying age-estimation uncertainty and posterior estimates

```
tree <- ggtree(beast_tree) +
  coord_cartesian(clip = 'off') +
  geom_tiplab(aes(label = label2), size=3, offset = 1) +
  geom_nodelab(aes(x=x, label=round(posterior, 2)), hjust=1, vjust=-1.2,
     size=2) +
  geom_range("height_0.95_HPD", color = "grey", size = 2, alpha = 0.7) +
  theme_tree2(plot.margin=margin(6, 100, 6, 3))

revts_tree <- revts(tree)
```

Save it as a PDF

```
ggsave("Figures/Fig2.pdf", revts_tree, width = 12, height = 7)
```

#StarBEAST2 log files output

Output from each independent run should be saved into its own subdirectory within the StarBEAST2 directory.

Reading in the data as mcmc objects

Read in your traces and make sure they are named correctly

```
path <- ("StarBEAST2/")
directories <- c("run1/","run2/", "run_from_prior/")
filename <- "starbeast.log"

for (i in directories){
  x = c(path, i, filename)
  assign(paste("run", which(directories==i), sep=""),
         as.data.frame(read.table(paste(x, collapse =""), header=T)))
}
```

Remove any parameters that are not involved with the species tree and give the remaining parameters sensible names

```
attributes(run1)$names
```

```
##   [1] "Sample"
##   [2] "posterior"
##   [3] "likelihood"
##   [4] "prior"
##   [5] "speciescoalescent"
##   [6] "popMean.Species"
##   [7] "branchRatesStdev.Species"
##   [8] "TreeHeight.Species"
##   [9] "TreeHeight.t.16S"
##  [10] "TreeHeight.t.ARG"
##  [11] "TreeHeight.t.COI"
##  [12] "TreeHeight.t.D7"
##  [13] "TreeHeight.t.H3"
##  [14] "TreeLength.Species"
##  [15] "treeLikelihood.16S"
##  [16] "treeLikelihood.ARG"
##  [17] "treeLikelihood.COI"
##  [18] "treeLikelihood.D7"
##  [19] "treeLikelihood.H3"
##  [20] "rateAC.s.16S"
##  [21] "rateAG.s.16S"
##  [22] "rateAT.s.16S"
##  [23] "rateCG.s.16S"
##  [24] "rateCT.s.16S"
##  [25] "rateGT.s.16S"
##  [26] "substmodel.s.16S"
##  [27] "rateAC.s.ARG"
##  [28] "rateAG.s.ARG"
```

```
##  [29] "rateAT.s.ARG"
##  [30] "rateCG.s.ARG"
##  [31] "rateCT.s.ARG"
##  [32] "rateGT.s.ARG"
##  [33] "substmodel.s.ARG"
##  [34] "rateAC.s.COI"
##  [35] "rateAG.s.COI"
##  [36] "rateAT.s.COI"
##  [37] "rateCG.s.COI"
##  [38] "rateCT.s.COI"
##  [39] "rateGT.s.COI"
##  [40] "substmodel.s.COI"
##  [41] "rateAC.s.D7"
##  [42] "rateAG.s.D7"
##  [43] "rateAT.s.D7"
##  [44] "rateCG.s.D7"
##  [45] "rateCT.s.D7"
##  [46] "rateGT.s.D7"
##  [47] "substmodel.s.D7"
##  [48] "rateAC.s.H3"
##  [49] "rateAG.s.H3"
##  [50] "rateAT.s.H3"
##  [51] "rateCG.s.H3"
##  [52] "rateCT.s.H3"
##  [53] "rateGT.s.H3"
##  [54] "substmodel.s.H3"
##  [55] "BMT_ModelIndicator.s.16S"
##  [56] "BMT_ModelIndicator.s.ARG"
##  [57] "BMT_ModelIndicator.s.COI"
##  [58] "BMT_ModelIndicator.s.D7"
##  [59] "BMT_ModelIndicator.s.H3"
##  [60] "BMT_gammaShape.s.16S"
##  [61] "BMT_gammaShape.s.ARG"
##  [62] "BMT_gammaShape.s.COI"
##  [63] "BMT_gammaShape.s.D7"
##  [64] "BMT_gammaShape.s.H3"
##  [65] "BMT_ProportionInvariable.s.16S"
##  [66] "BMT_ProportionInvariable.s.ARG"
##  [67] "BMT_ProportionInvariable.s.COI"
##  [68] "BMT_ProportionInvariable.s.D7"
##  [69] "BMT_ProportionInvariable.s.H3"
##  [70] "hasGammaRates.s.16S"
##  [71] "hasGammaRates.s.ARG"
##  [72] "hasGammaRates.s.COI"
##  [73] "hasGammaRates.s.D7"
##  [74] "hasGammaRates.s.H3"
##  [75] "hasInvariableSites.s.16S"
##  [76] "hasInvariableSites.s.ARG"
##  [77] "hasInvariableSites.s.COI"
##  [78] "hasInvariableSites.s.D7"
##  [79] "hasInvariableSites.s.H3"
##  [80] "ActivePropInvariable.s.16S"
##  [81] "ActivePropInvariable.s.ARG"
##  [82] "ActivePropInvariable.s.COI"
```

```
##  [83] "ActivePropInvariable.s.D7"
##  [84] "ActivePropInvariable.s.H3"
##  [85] "ActiveGammaShape.s.16S"
##  [86] "ActiveGammaShape.s.ARG"
##  [87] "ActiveGammaShape.s.COI"
##  [88] "ActiveGammaShape.s.D7"
##  [89] "ActiveGammaShape.s.H3"
##  [90] "hasEqualFreqs.s.16S"
##  [91] "hasEqualFreqs.s.ARG"
##  [92] "hasEqualFreqs.s.COI"
##  [93] "hasEqualFreqs.s.D7"
##  [94] "hasEqualFreqs.s.H3"
##  [95] "BMT_frequencies.s.16S1"
##  [96] "BMT_frequencies.s.16S2"
##  [97] "BMT_frequencies.s.16S3"
##  [98] "BMT_frequencies.s.16S4"
##  [99] "BMT_frequencies.s.ARG1"
## [100] "BMT_frequencies.s.ARG2"
## [101] "BMT_frequencies.s.ARG3"
## [102] "BMT_frequencies.s.ARG4"
## [103] "BMT_frequencies.s.COI1"
## [104] "BMT_frequencies.s.COI2"
## [105] "BMT_frequencies.s.COI3"
## [106] "BMT_frequencies.s.COI4"
## [107] "BMT_frequencies.s.D71"
## [108] "BMT_frequencies.s.D72"
## [109] "BMT_frequencies.s.D73"
## [110] "BMT_frequencies.s.D74"
## [111] "BMT_frequencies.s.H31"
## [112] "BMT_frequencies.s.H32"
## [113] "BMT_frequencies.s.H33"
## [114] "BMT_frequencies.s.H34"
## [115] "molecularClockRate.c.16S"
## [116] "molecularClockRate.c.ARG"
## [117] "molecularClockRate.c.COI"
## [118] "molecularClockRate.c.D7"
## [119] "molecularClockRate.c.H3"
## [120] "BirthDeathModel.t.Species"
## [121] "netDiversificationRate.t.Species"
## [122] "ExtinctionFraction.t.Species"
## [123] "monophyletic.root.prior."
## [124] "logP.mrca.root.prior.."
## [125] "mrca.age.root.prior."
## [126] "parameter.hyperExponential.mean.root.prior.prior"
## [127] "HyperPrior.hyperExponential.mean.root.prior.prior"
```

```
for (i in 1:length(directories)){
  temp <- eval(parse(text=paste("run", i, "[,c(1:8,␣14,␣
     120:122,124,126,127)]",sep = "")))
  colnames(temp) <- c("sample", "posterior", "likelihood", "prior", "
     coalescent",
                      "population.size", "branch.rates", "root.age",
                      "branch.lengths.summed", "birth.death.model",
                      "diversification.rate", "extinction.fraction", "root
```

```
                          .log",
                        "root.hyper.parameter", "root.hyper.parameter.prior"
                          )
  assign(paste("run", i, sep = ""), temp)
}
```

Set the variables that will be used to create your MCMC object, including 20% post-burnin used when combining posterior species trees.

```
burnin <- floor(0.20 * nrow(run1))
thin <- run1$sample[2] - run1$sample[1]
start <- burnin*thin
end <- max(run1$sample)
```

Call the "coda" package for creating mcmc objects

```
require(coda)
```

Make your mcmc objects and throw away the burnin

```
for (i in 1:length(directories)){
  temp <- eval(parse(text=paste("run", i, "[-c(1:burnin),]",sep = "")))
  assign (paste ("mcmc",i ,sep = ""),
          mcmc(data = temp , start = start, end = end, thin = thin))
}
```

Put them into a list

```
mh.list <-mcmc.list()
for (i in 1:length(directories)){
      temp <- eval(parse(text=paste("mcmc", i, sep = "")))
      mh.list[[i]] <- temp
}
```

##Comparing log files when sampling from the prior and when sampling from the data (Fig. 3)

Reshape data for root age parameter from run 1 (sampling with data) and run 3 (sampling from the prior only)

```
reshaping <- function(x) {
  y <- rbind(t(x[,-1]))
  result <- setNames(as.data.frame.table(y),c("variable","run","value"))
}

for (i in 1:length(directories)){
  temp <- eval(parse(text=paste("run", i,sep = "")))
  reshape <- reshaping(temp)
  reshape$run <- as.factor(rep(i, nrow(reshape)))
  reshape <- reshape[-c(1:burnin),]
  assign(paste("reshape", i, sep = ""), reshape)
}

plot_df <- subset(rbind(reshape1,reshape3), variable == "root.age")
```

Plot the density curves againt each other

```
p <- ggplot(plot_df, aes(x= value, fill = run)) +
  geom_density(alpha = 0.3) +
  scale_fill_manual(values=c("grey0", "goldenrod")) +
  scale_colour_manual(values="grey0") +
  theme_bw(base_size = 12)+
  theme(legend.position = "none", strip.text.x = element_text(size = 6),
        panel.grid.major = element_blank(), panel.grid.minor = element_
           blank())+
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  labs(x="Age␣Estimation", y = "Density")
```

Save figure

```
ggsave("Fig3.pdf", plot = p, device = NULL, path = "Figures",
       scale = 1, width = 7, height = 5, dpi = 300, limitsize = TRUE,
       units = "in")
```

Diagnostics of StarBEAST2 analysis

1) Diagnostics for a single run

Create a new mh.list only including data from runs 1 and 2 (runs with data)

```
mh.list.2 <- mcmc.list()
for (i in 1:(length(directories)-1)){
      temp <- eval(parse(text=paste("mcmc", i, sep = "")))
      mh.list.2[[i]] <- temp
}
```

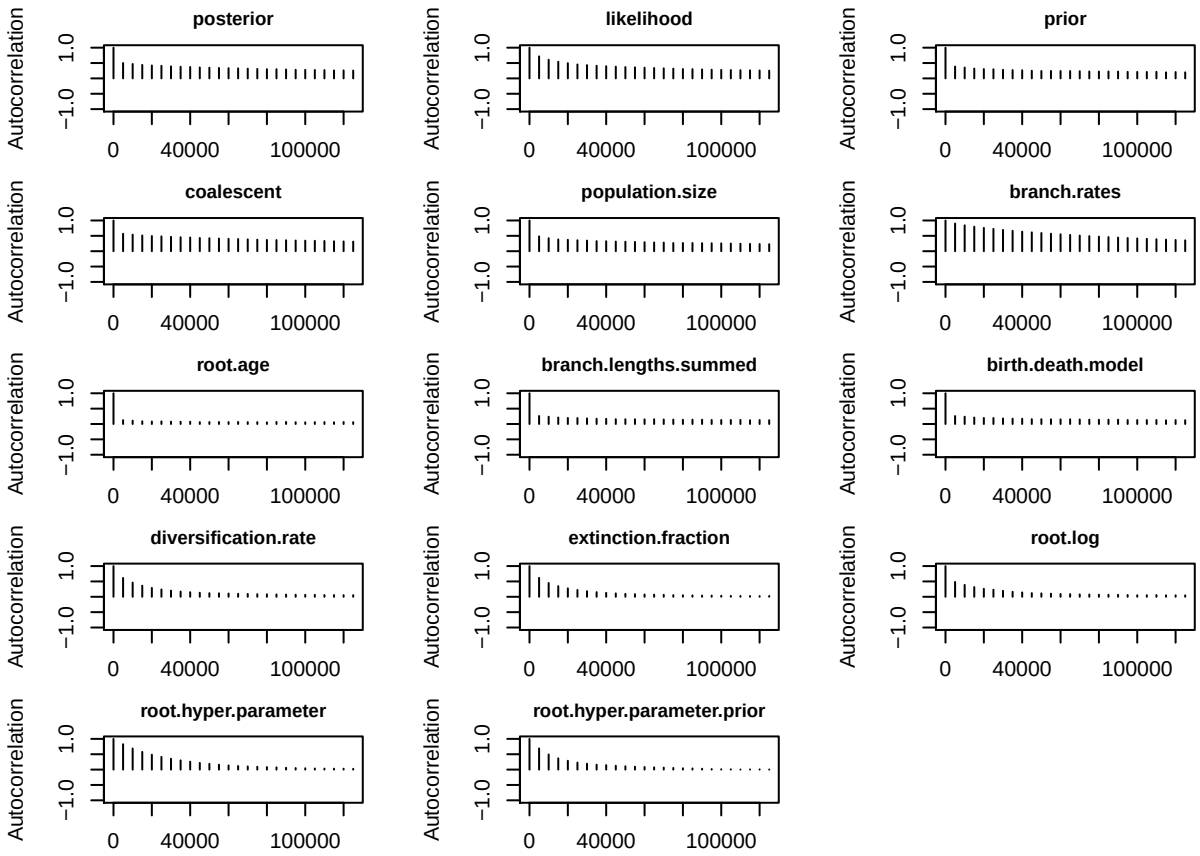
a) Autocorrelation Calculate autocorrelation between draws

```
diag(autocorr(mcmc1)[2, , ])
```

```
##                      sample                   posterior
##                   0.9999063                   0.5000439
##                  likelihood                       prior
##                   0.7182150                   0.3861944
##                  coalescent             population.size
##                   0.5681311                   0.4823333
##                branch.rates                    root.age
##                   0.9000373                   0.1255783
##       branch.lengths.summed           birth.death.model
##                   0.2600304                   0.2590325
##        diversification.rate         extinction.fraction
##                   0.6133483                   0.6181057
##                    root.log        root.hyper.parameter
##                   0.4835762                   0.8261899
## root.hyper.parameter.prior
##                   0.6892516
```

Present autocorrelation in plot form

```
par(mfrow=c(5,3), mar=c(2,4,2,2)+0.1, cex.main = 0.9)
autocorr.plot(mcmc1[,-1], lag.max = 25, auto.layout = F)
```



b) Effective sample size Calculate effective sample size

```
effectiveSize(mcmc1[,-1])
```

```
##                    posterior                   likelihood
##                     736.0322                    764.4830
##                        prior                   coalescent
##                     800.3543                    659.0114
##              population.size                 branch.rates
##                     949.6819                    704.2918
##                     root.age       branch.lengths.summed
##                    3143.0470                   1559.6140
##           birth.death.model        diversification.rate
##                    1585.2244                   3496.3110
##         extinction.fraction                    root.log
##                    4266.7123                   3698.9433
##        root.hyper.parameter root.hyper.parameter.prior
##                    2749.2170                   4230.6742
```

2) Test for convergence of multiple runs

a) Gelman-Rubin statistic Calculate Gelman-Rubin diagnostic of convergence

```
gelman.diag(mh.list.2,confidence = 0.95, transform=TRUE, autoburnin=FALSE,
    multivariate=FALSE)
```

```
## Potential scale reduction factors:
##
##                              Point est. Upper C.I.
## sample                              NaN        NaN
## posterior                             1          1
## likelihood                            1          1
## prior                                 1          1
## coalescent                            1          1
## population.size                       1          1
## branch.rates                          1          1
## root.age                              1          1
## branch.lengths.summed                 1          1
## birth.death.model                     1          1
## diversification.rate                  1          1
## extinction.fraction                   1          1
## root.log                              1          1
## root.hyper.parameter                  1          1
## root.hyper.parameter.prior            1          1
```

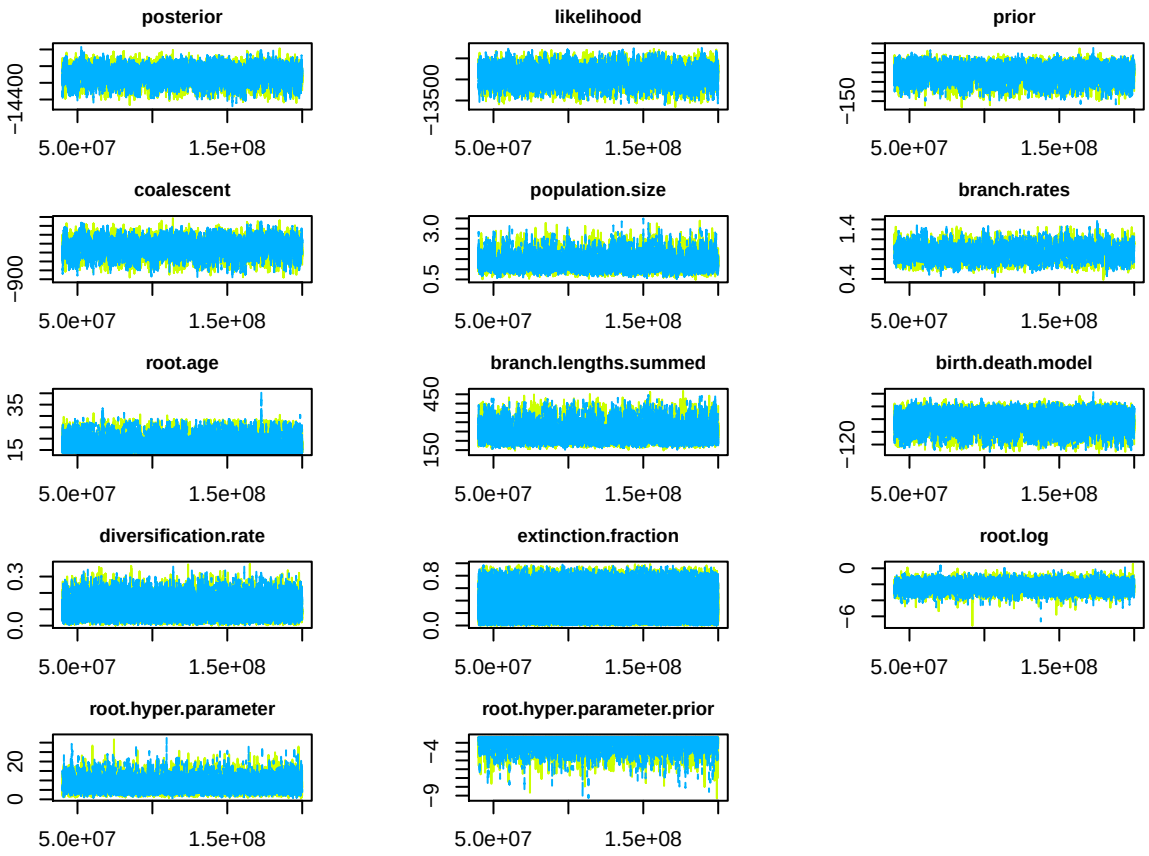
b) Plotting traces Get data into correct format for plotting traces from both runs simultaneously

```
for (i in 2:length(run1)){
  for (j in 1:length(directories)){
    temp <- eval(parse(text=paste("mcmc", j, "[,i]", sep = "")))
    assign(paste("trace_","mcmc", j, colnames(run1[i]), sep=""), temp)
  }
}

temp_list = mcmc.list()
for (i in 2:length(run1)){
  for (j in 1:(length(directories)-1)){
    temp_list[[j]] <-  eval(parse(text=paste("trace_mcmc", j , colnames(
       run1[i]), sep = "")))
    assign(paste("trace_" , colnames(run1[i]), sep = ""), temp_list)
 }
}
```

Plot traces

```
par(mfrow=c(5,3), mar = c(2,4,2,2)+0.1, cex.main = 0.9)
for (i in 2:length(run1)){
  temp <-  eval(parse(text = paste("trace_", colnames(run1[i]), sep = ""))
     )
  traceplot(temp,col = rainbow(3, start=0.2, end=0.9))
  title(main = colnames(run1[i]), ylab = "␣")
}
```



c) Plotting density curves Get data into correct format for plotting density curves of both runs and throw out burnin

```
reshaping <- function(x) {
  y <- rbind(t(x[,-1]))
  result <- setNames(as.data.frame.table(y),c("variable","run","value"))
}

for (i in 1:(length(directories)-1)){
  temp <- eval(parse(text=paste("run", i,sep = "")))
  reshape <- reshaping(temp)
  reshape$run <- as.factor(rep(i, nrow(reshape)))
  reshape <- reshape[-c(1:burnin),]
  assign(paste("reshape", i, sep = ""), reshape)
}

plot_df <- rbind(reshape1,reshape2)
```

Plot by each parameter

```
p <- ggplot(plot_df, aes(x= value, fill = run, colour = run)) +
  geom_density(alpha = 0.1) +
  theme_bw(base_size = 12)+
  theme(legend.position = "none", strip.text.x = element_text(size = 6),
      axis.text.y = element_text(size=5), axis.text.x = element_text(size
         =5),
```

```
        panel.grid.major = element_blank(), panel.grid.minor = element_
            blank())+
  labs(x="Parameter value", y = "Density") +
  facet_wrap(~variable, scales = "free", ncol = 3)

p
```