Time calibrated pyhlogeny (incl. diagnostics)

Beatriz Willink Rachel Blow

Contents

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, 15, 6, 3)) +
  geom_cladelabel(node=43, label = "Clade I", offset = 5) +
  geom_cladelabel(node=69, label= "Clade II", offset = 5) +
  geom_cladelabel(node=53, label= "Clade III", offset = 5) +
  geom_cladelabel(node=56, label= "Clade IV", offset = 5)

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_Rates.s.16S1"
##  [61] "BMT_Rates.s.16S2"
##  [62] "BMT_Rates.s.16S3"
##  [63] "BMT_Rates.s.16S4"
##  [64] "BMT_Rates.s.16S5"
##  [65] "BMT_Rates.s.16S6"
##  [66] "BMT_Rates.s.ARG1"
##  [67] "BMT_Rates.s.ARG2"
##  [68] "BMT_Rates.s.ARG3"
##  [69] "BMT_Rates.s.ARG4"
##  [70] "BMT_Rates.s.ARG5"
##  [71] "BMT_Rates.s.ARG6"
##  [72] "BMT_Rates.s.COI1"
##  [73] "BMT_Rates.s.COI2"
##  [74] "BMT_Rates.s.COI3"
##  [75] "BMT_Rates.s.COI4"
##  [76] "BMT_Rates.s.COI5"
##  [77] "BMT_Rates.s.COI6"
##  [78] "BMT_Rates.s.D71"
##  [79] "BMT_Rates.s.D72"
##  [80] "BMT_Rates.s.D73"
##  [81] "BMT_Rates.s.D74"
##  [82] "BMT_Rates.s.D75"
##  [83] "BMT_Rates.s.D76"
##  [84] "BMT_Rates.s.H31"
##  [85] "BMT_Rates.s.H32"
##  [86] "BMT_Rates.s.H33"
##  [87] "BMT_Rates.s.H34"
##  [88] "BMT_Rates.s.H35"
##  [89] "BMT_Rates.s.H36"
##  [90] "BMT_gammaShape.s.16S"
##  [91] "BMT_gammaShape.s.ARG"
##  [92] "BMT_gammaShape.s.COI"
##  [93] "BMT_gammaShape.s.D7"
##  [94] "BMT_gammaShape.s.H3"
##  [95] "BMT_ProportionInvariable.s.16S"
##  [96] "BMT_ProportionInvariable.s.ARG"
##  [97] "BMT_ProportionInvariable.s.COI"
##  [98] "BMT_ProportionInvariable.s.D7"
##  [99] "BMT_ProportionInvariable.s.H3"
## [100] "hasGammaRates.s.16S"
## [101] "hasGammaRates.s.ARG"
## [102] "hasGammaRates.s.COI"
## [103] "hasGammaRates.s.D7"
```

```
## [104] "hasGammaRates.s.H3"
## [105] "hasInvariableSites.s.16S"
## [106] "hasInvariableSites.s.ARG"
## [107] "hasInvariableSites.s.COI"
## [108] "hasInvariableSites.s.D7"
## [109] "hasInvariableSites.s.H3"
## [110] "ActivePropInvariable.s.16S"
## [111] "ActivePropInvariable.s.ARG"
## [112] "ActivePropInvariable.s.COI"
## [113] "ActivePropInvariable.s.D7"
## [114] "ActivePropInvariable.s.H3"
## [115] "ActiveGammaShape.s.16S"
## [116] "ActiveGammaShape.s.ARG"
## [117] "ActiveGammaShape.s.COI"
## [118] "ActiveGammaShape.s.D7"
## [119] "ActiveGammaShape.s.H3"
## [120] "hasEqualFreqs.s.16S"
## [121] "hasEqualFreqs.s.ARG"
## [122] "hasEqualFreqs.s.COI"
## [123] "hasEqualFreqs.s.D7"
## [124] "hasEqualFreqs.s.H3"
## [125] "BMT_frequencies.s.16S1"
## [126] "BMT_frequencies.s.16S2"
## [127] "BMT_frequencies.s.16S3"
## [128] "BMT_frequencies.s.16S4"
## [129] "BMT_frequencies.s.ARG1"
## [130] "BMT_frequencies.s.ARG2"
## [131] "BMT_frequencies.s.ARG3"
## [132] "BMT_frequencies.s.ARG4"
## [133] "BMT_frequencies.s.COI1"
## [134] "BMT_frequencies.s.COI2"
## [135] "BMT_frequencies.s.COI3"
## [136] "BMT_frequencies.s.COI4"
## [137] "BMT_frequencies.s.D71"
## [138] "BMT_frequencies.s.D72"
## [139] "BMT_frequencies.s.D73"
## [140] "BMT_frequencies.s.D74"
## [141] "BMT_frequencies.s.H31"
## [142] "BMT_frequencies.s.H32"
## [143] "BMT_frequencies.s.H33"
## [144] "BMT_frequencies.s.H34"
## [145] "molecularClockRate.c.16S"
## [146] "molecularClockRate.c.ARG"
## [147] "molecularClockRate.c.COI"
## [148] "molecularClockRate.c.D7"
## [149] "molecularClockRate.c.H3"
## [150] "BirthDeathModel.t.Species"
## [151] "netDiversificationRate.t.Species"
## [152] "ExtinctionFraction.t.Species"
## [153] "logP.mrca.root.prior.."
## [154] "mrcatime.root.prior."
## [155] "parameter.hyperExponential.mean.root.prior.prior"
## [156] "HyperPrior.hyperExponential.mean.root.prior.prior"
```

```
for (i in 1:length(directories)){
  temp <- eval(parse(text=paste("run", i, "[,c(1:8,14,150:153,155:156)]",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

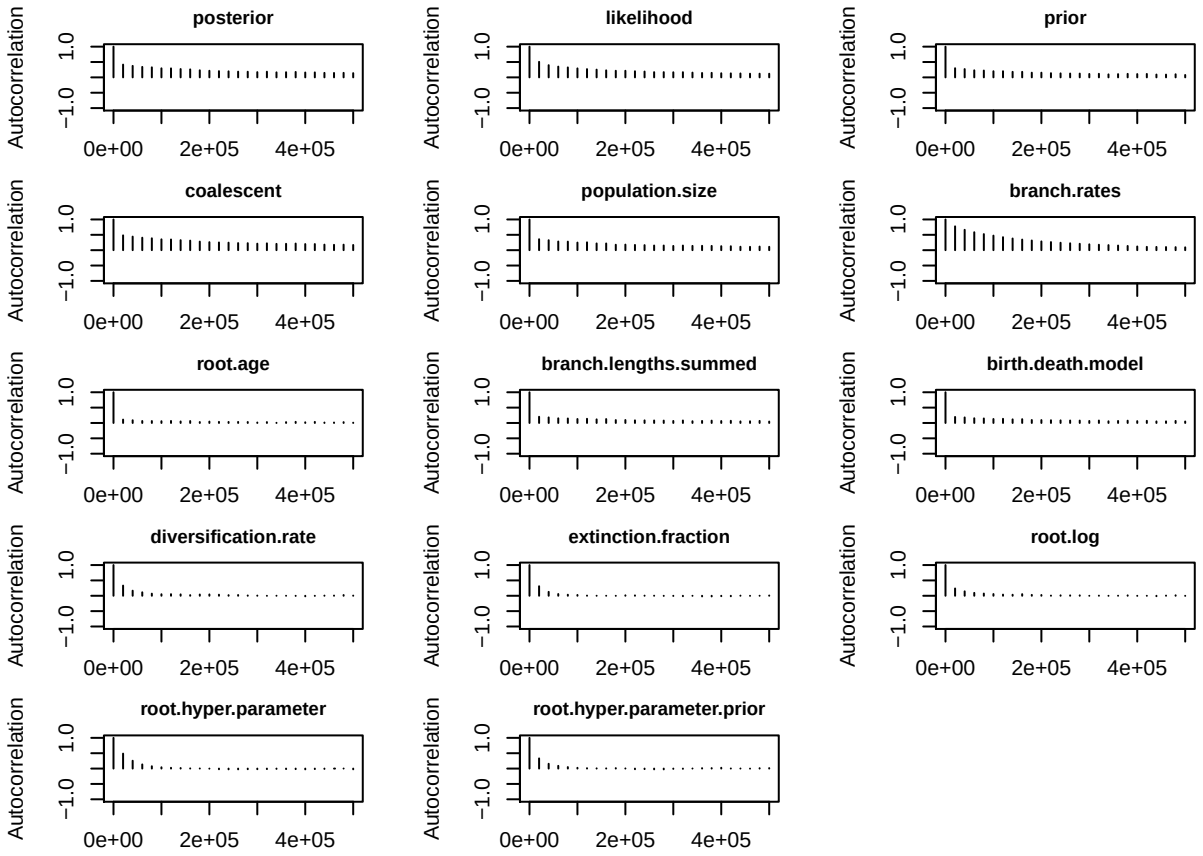
```
diag(autocorr(mcmc1)[2, , ])
```

```
##                    sample                 posterior
##                 0.9996250                 0.4157919
##                likelihood                     prior
##                 0.5059585                 0.2979133
##                coalescent           population.size
```

```
##                 0.4823087                 0.3565740
##             branch.rates                  root.age
##                 0.7760449                 0.1098280
##      branch.lengths.summed         birth.death.model
##                 0.2020148                 0.1964479
##       diversification.rate       extinction.fraction
##                 0.3313919                 0.3109229
##                  root.log      root.hyper.parameter
##                 0.2380908                 0.4868924
## root.hyper.parameter.prior
##                 0.3317132
```

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
```

```
##                   393.8220                   565.2079
##                      prior                  coalescent
##                   636.0853                   316.9500
##            population.size                branch.rates
##                   401.9005                   537.4080
##                   root.age      branch.lengths.summed
##                  3122.9389                   987.4064
##          birth.death.model       diversification.rate
##                   996.7976                  2579.2317
##        extinction.fraction                   root.log
##                  3925.2359                  2989.5049
##       root.hyper.parameter root.hyper.parameter.prior
##                  2633.7426                  3433.4249
```

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                          1.00       1.00
## likelihood                         1.15       1.16
## prior                              1.00       1.00
## coalescent                         1.00       1.00
## population.size                    1.00       1.00
## branch.rates                       1.00       1.00
## root.age                           1.00       1.00
## branch.lengths.summed              1.00       1.00
## birth.death.model                  1.00       1.00
## diversification.rate               1.00       1.00
## extinction.fraction                1.00       1.00
## root.log                           1.00       1.00
## root.hyper.parameter               1.00       1.00
## root.hyper.parameter.prior         1.00       1.00
```

b) Plotting traces

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

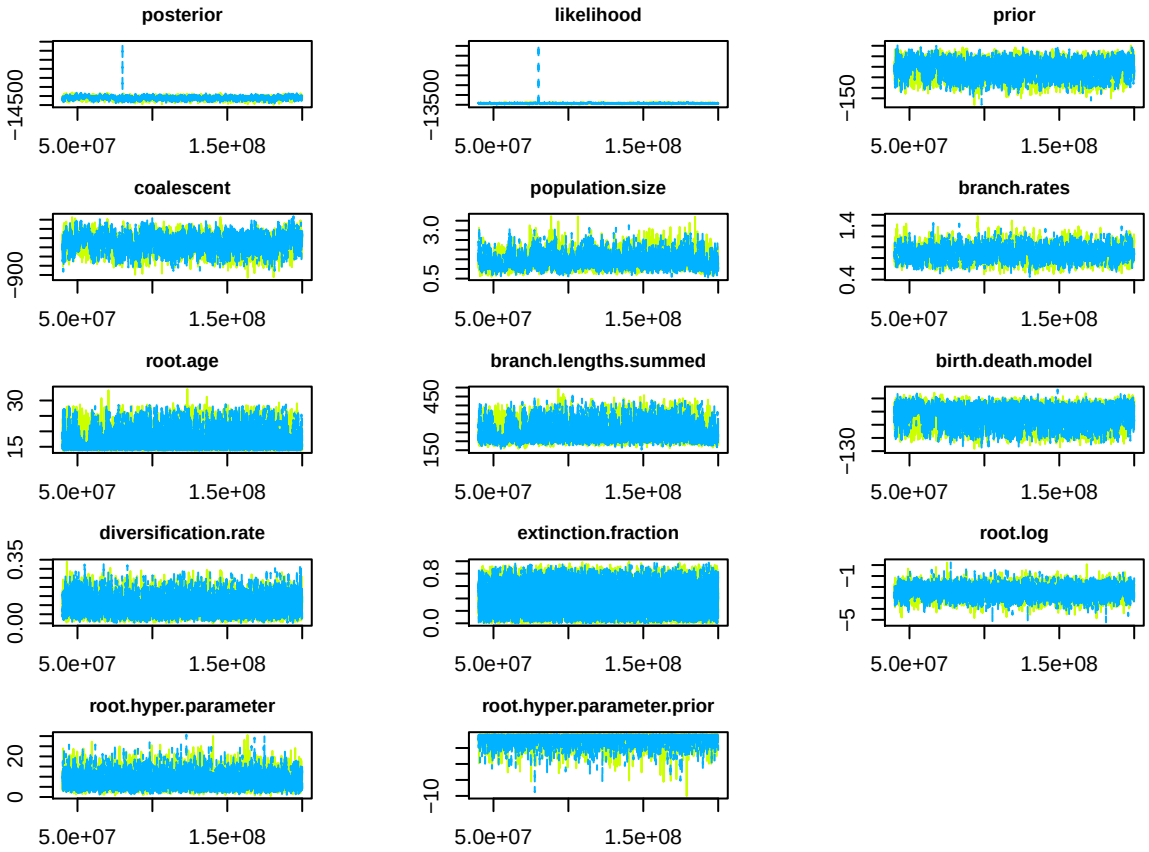
```
for (i in 2:length(run1)){
  for (j in 1:length(directories)){
    temp <- eval(parse(text=paste("mcmc", j, "[,i]", sep = "")))
    assign(paste("trace_","mcmc", j, colnames(run1[i]), sep=""), temp)
  }
}

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
```