Post-Analysis Evaluation of Simulations for TreEvo

David Bapst

August 13, 2019

First thing first, what directory do we want to get data files from?

```
dir <- "~//treevo_paper//analyses_cluster_fast//"
```

Load these, then remove all empty runs so we only analyze output for runs that we have results for.

```
## TreEvo Version Used: 0.21.0
```

```
## ape Version Used: 5.3
```

Effective Sample Size

This function calculates Effective Sample Size (ESS) on results.

Performs the best when results are from multiple runs.

And then let's look at ESS:

```
ESS
```

```
## $An_Emp_Bound
##             1.2
## starting_1  226
## intrinsic_1 226
## intrinsic_2 226
##
## $An_Emp_BrownMotion
##                  1.2
## starting_1  259.0000
## intrinsic_1 330.3551
##
## $An_Emp_TimeReg
##                    1.2
## starting_1   15.36522
## intrinsic_1 146.61490
##
## $Aq_Emp_3Opt2Bound
##                    1.2
## starting_1   7.224600
## intrinsic_1  1.574292
## intrinsic_2 13.966218
## intrinsic_3  1.862767
## intrinsic_4 12.008233
## intrinsic_5 30.419311
## intrinsic_6 33.480387
## intrinsic_7 13.660456
## intrinsic_8 11.778192
##
## $Aq_Emp_BrownMotion
```

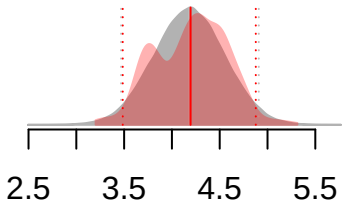
```
##                   1.2
## starting_1  160.51778
## intrinsic_1  39.28863
```

plotUnivariatePosteriorVsPrior

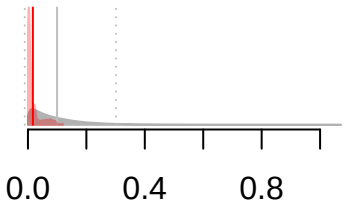
This function plots priors versus their posteriors - this will be useful for runs with bad prior on BM.

Note that the following code will skip parameters whose posterior distributions are highly discontinuous, suggesting complex multimodal distributions that are not best considered via smoothed density kernals.

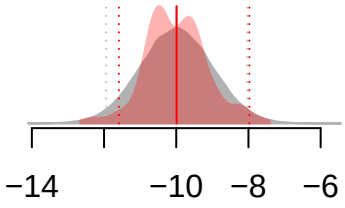
```
##                                 jobName
## "An_Emp_Bound_tree_1_trait_1_08-07-19"
## [1] "Run 1"
```



starting_1

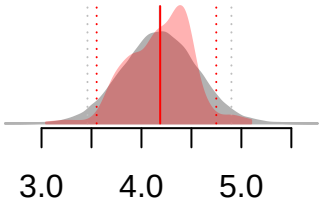


intrinsic_1

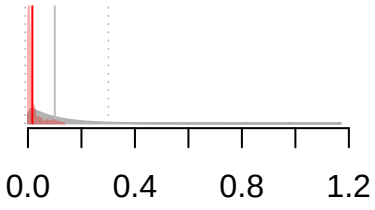


intrinsic_2

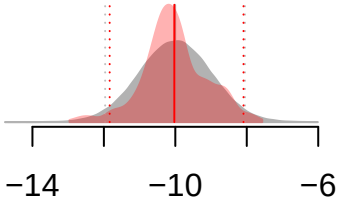
```
##                                 jobName
## "An_Emp_Bound_tree_1_trait_1_08-07-19"
## [1] "Run 2"
```



starting_1

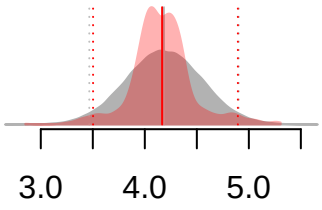


intrinsic_1

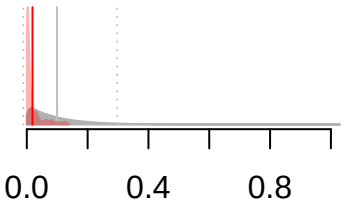


intrinsic_2

```
##                                       jobName
## "An_Emp_BrownMotion_tree_1_trait_1_08-09-19"
## [1] "Run 1"
```

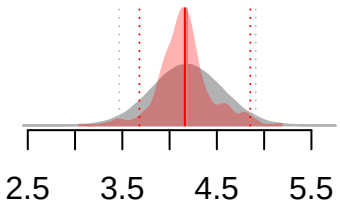


starting_1

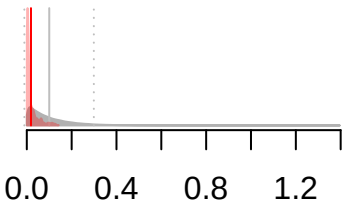


intrinsic_1

```
##                                       jobName
## "An_Emp_BrownMotion_tree_1_trait_1_08-09-19"
## [1] "Run 2"
```

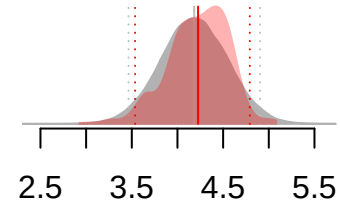


starting_1

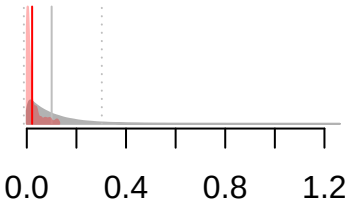


intrinsic_1

```
##                                     jobName
## "An_Emp_TimeReg_tree_1_trait_1_08-09-19"
## [1] "Run 1"
```

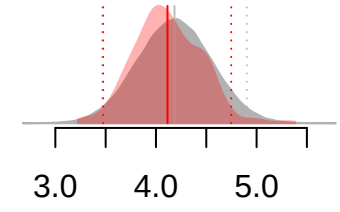


starting_1

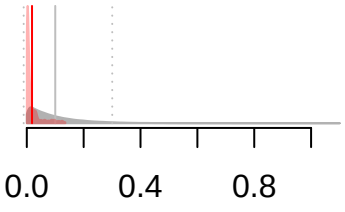


intrinsic_1

```
##                                     jobName
## "An_Emp_TimeReg_tree_1_trait_1_08-09-19"
## [1] "Run 2"
```

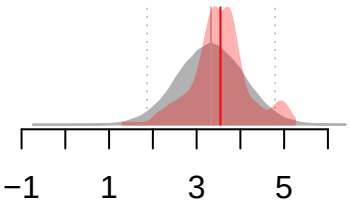


starting_1

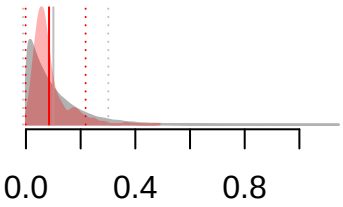


intrinsic_1

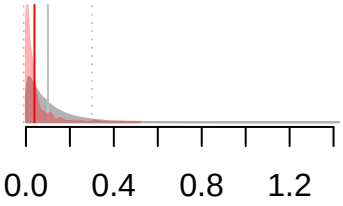
```
##                                       jobName
## "Aq_Emp_3Opt2Bound_tree_1_trait_1_08-09-19"
## [1] "Run 1"
```



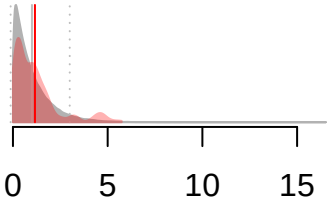
starting_1



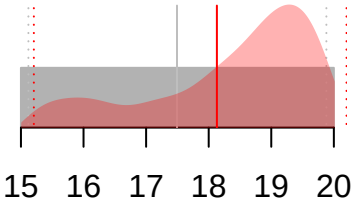
intrinsic_1



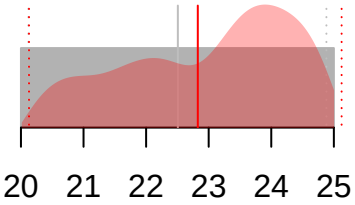
intrinsic_2



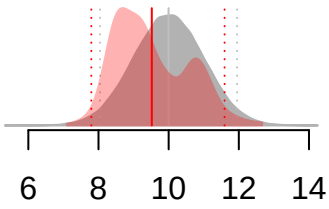
intrinsic_3



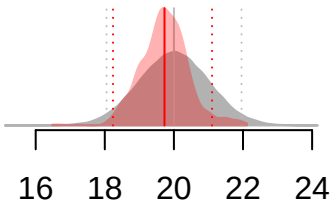
intrinsic_4



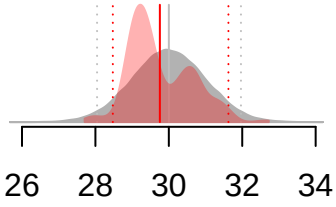
intrinsic_5



intrinsic_6

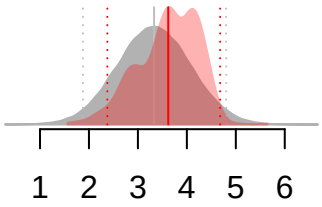


intrinsic_7

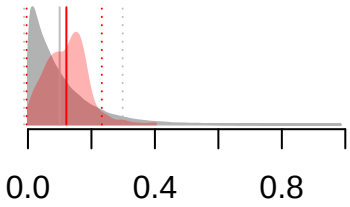


intrinsic_8

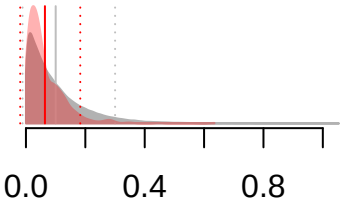
```
##                                       jobName
## "Aq_Emp_3Opt2Bound_tree_1_trait_1_08-09-19"
## [1] "Run 2"
```



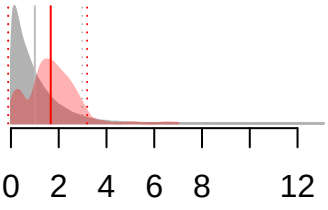
starting_1



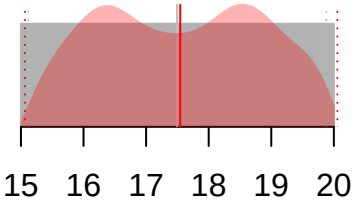
intrinsic_1



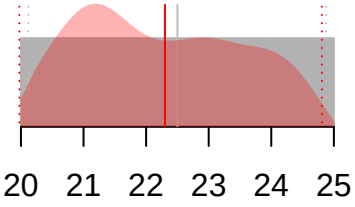
intrinsic_2



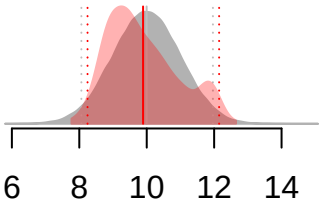
intrinsic_3



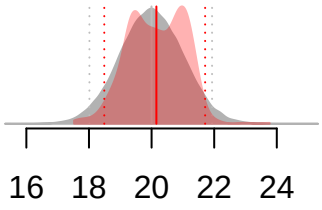
intrinsic_4



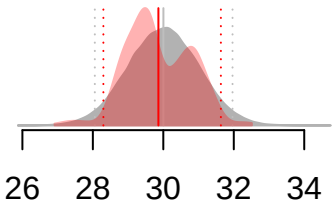
intrinsic_5



intrinsic_6

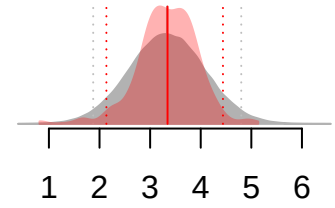


intrinsic_7

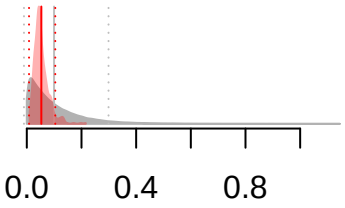


intrinsic_8

```
##                                       jobName
## "Aq_Emp_BrownMotion_tree_1_trait_1_08-09-19"
## [1] "Run 1"
```

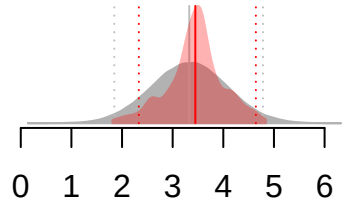


starting_1

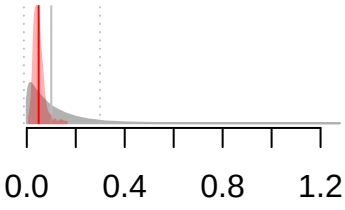


intrinsic_1

```
##                                       jobName
## "Aq_Emp_BrownMotion_tree_1_trait_1_08-09-19"
## [1] "Run 2"
```



starting_1



intrinsic_1

Because we did multiple runs, the most useful way to look at these plots is compare parameter estimates from different runs and see if there is convergence. In this case, we can see runs with poor ESS do not look like they have converged well on the same suite of parameter estimates.

The function `highestPostDens` returns for the weighted mean, standard deviation, upper and lower highest posterior density (HPD) for each free parameter in posterior. This probably isn't very useful when we can plot the distributions and compare them, like above.

```
highestPostDens(results$particleDataFrame, percent=0.95, returnData=FALSE)
```

plotABC_3D

This function plots posterior density distribution for each generation in a three-dimensional plot window. Unfortunately, due to `gpclib` not being available on Windows machines, it isn't available for the author of this document at this very moment.

```
for(i in 1:length(analysisOutput)){
  nTree <- length(analysisOutput[[i]])
  for(j in nTree){
    nTrait <- length(analysisOutput[[i]][[j]])
    for (k in nTrait){
      nRunsFound <- length(analysisOutput[[i]][[j]][[k]])
      for(l in 1:nRunsFound){
        analysisFound <- analysisOutput[[i]][[j]][[k]][[l]]
        #
        print(analysisFound$input.data["jobName",])
        #
        whichNonFixedPriors <- which(sapply(
          analysisFound$priorList,
          function(x) x$fun != "fixed"
          ))
        #
        nPar <- length(whichNonFixedPriors)
        #
        for(m in 1:nPar){

          plotABC_3D(
            particleDataFrame = results[[1]]$particleDataFrame,
            parameter = 6 + m,
            show.particles = "none",
            plot.parent = FALSE,
            realParam = FALSE,
            realParamValues = NA
            )

          }
        }
      }
    }
  }
```

Methods for comparing Results of Analyses Based on Simulated Data To True Values

plotPosteriors

For each free parameter in the posterior, a plot is made of the distribution of values estimate in the last generation. This can also be used to visually compare against true (generating) parameter values in a simulation.

```
plotPosteriors(particleDataFrame=resultsBM$particleDataFrame,
   priorsMat=resultsBM$PriorMatrix)
```

testMultivarOutlierHDR

This tests if an 'outlier' (some sample) is within a multivariate cloud of particles at some alpha

Very useful for testing if the generating parameters are within the particles for a simulation for dependant analyses - not so useful for indep analyses though!

```
particleMatrix <- NA
generatingParams <- NA

testMultivarOutlierHDR(
  dataMatrix = particleMatrix,
  outlier = generatingParams,
  alpha = 0.8,
  pca = TRUE
  )
```

NOTES

Notes from conversation with Peter Smits (05-09-18)

So I'm doing approximate bayesian computation and the question is, what do I want to show to the reader

I want to show posterior parameter estimates from real data, and show that they are very different from parameter estimates made under other models, or under the same model but with simulated data, for scenarios with a small number of models.

To show this, I want to make the same series of posterior predictive checks and demonstrate how your prefered model better recapitulates the data it was fit to.

ECDF

ECDF is the empirical cumulative distribution function, also known as the ranked order accumulation curve, available as the function `ecdf` in R.

ECDF is a cool way of summarizing the entire dataset graphically.

Basic question: How well does simulations under a fit model reproduce ecdf or the density of the original data? If your model does a better job of doing that, then it is straight up a better model. It also goes beyond parameter estimates and towards the model describing the data

Bayesian analysis wants to describe more than just the expected value. it is greedy and wants to describe the whole posterior. The posterior predictive distribution describes all data sets that are consistent with the model, given the original input information. If the PPD doesn't look like the empirical data, then the model is not describing your data.

More notes, from 06-21-18

The general sketch is (a) sample particles from the posterior, simulate under this set of parameters N times, and compare the original ECDF for each parameter to the simulated.

A different other idea:

a) First, draw parameter estimates from some posterior particle, simulate under those parameters
b) Then test if 'true' generating parameters are actually within the 95% HDF of the simulated posterior

This approach deals with how we don't really understand how adequate the models are for giving unbiased estimates of parameters.

checkAdequacy

sixAnalysesTwoModels

One could imagine writing a function that just takes arguments: tree, params, etc. and returns results for a particular comparison between two models (possible function names `checkAdequacy` or `sixAnalysesTwoModels`). Basic idea would be you have a dataset as input, two models of interest, and you would then follow this up with six corresponding analyses.

a) Fit model A and model B to real data
b) Simulate under model A and model B fitted parameters from the posterior
c) Then fit both model A and B to both sets of simulated data.

In this case, model A would generally be some simple 'null' model that we wish to compare against, such as BM.