Introduction to PhyloAcc-C tutorial (version 0.1)

Patrick Gemmell

Introduction

Genomes contain conserved non-coding elements (CNEs) that are important to biological function. But which elements relate to which traits, and how? PhyloAcc-C answers this question by estimating how molecular substitution rates and the rate of change of a continuous trait covary.

Input to PhyloAcc-C is a multiple sequence alignment of a CNE, continuous trait data observed in extant species, and a background phylogeny and substitution process. Gibbs sampling is used to assign latent conservation states (background, conserved, accelerated) to lineages. These hidden states are associated with molecular and phenotypic change by rate multipliers. Covariation in rates of molecular and trait evolution is assessed using the posterior ratio of trait variance multipliers $\beta_3 : \beta_2$.

In this tutorial, the PhyloAcc-C approach will be illustrated using publicly available mammalian nucleotide alignments and lifespan data.

N.B. PhyloAcc-C has been implemented using a mixture of R and C++. The upside of using C++ is good performance. The downside is that to use PhyloAcc-C you will need a development environment installed on your computer. In future we will provide binary packages.

More information on the PhyloAcc family of methods is available at: https://phyloacc.github.io.

Quick start using real data

Let's work through a basic PhyloAcc-C analysis using the data files supplied with this tutorial document. As well as having installed PhyloAcc-C, you will also need to have installed the `ape` package, a useful tool for working with phylogenetic data in R.

First we must specify the relationship between some mammalian species. This is done by loading a tree using `ape`. The tree is the same as that used by Hu et al. (2019); the branch lengths capture the relationship between loci under neutral evolution.

```r
# A fixed phylogeny.
tree = ape::read.tree("mammal_tree.txt")
```

As well as a phylogeny, PhyloAcc-C requires details on the molecular substitution process sequences undergo. Let's supply a rate matrix and stationary distribution, again using data from Hu et al. (2019).

```r
# A fixed rate matrix.
Q <- matrix(
  c(-1.075162,  0.186970,  0.696268,  0.191923,
     0.181082, -0.873473,  0.255492,  0.436899,
     0.674340,  0.255493, -1.164645,  0.234813,
     0.191924,  0.451108,  0.242449, -0.885481),
  nrow=4, ncol=4)

# The associated fixed stationary distribution.
PI <- c(0.246000, 0.254000, 0.254000, 0.246000)
```

The aim of PhyloAcc-C is to relate the phenotypic evolution of a trait to the molecular evolution of a CNE. Therefore, we will load some trait data. In this case we use data on mammalian size and lifespan, taken from the SI of Kowalczyk et al. (2020).

I added a column `recode` to the data I downloaded and used it to make sure the names in the trait data file match the names in the tree exactly. PhyloAcc-C expects the trait of interest to be in a column `Y`, and species name to be in a column `NAME`; Kowalczyk et al. (2020) actually studied two traits, so we will set `Y` to refer to the column containing trait ELL, or "exceptionally long-lived given body size".

```
# Mammalian trait data from Kowalczyk et al. (2020).
trait_tbl = with(read.csv("mammal_trait.tsv", sep="\t"), data.frame(NAME=recode, Y=PC2))
```

Finally, we will load some sequence data, in this case a CNE from chromosome 17. Again, we use helpful functions from the `ape` package.

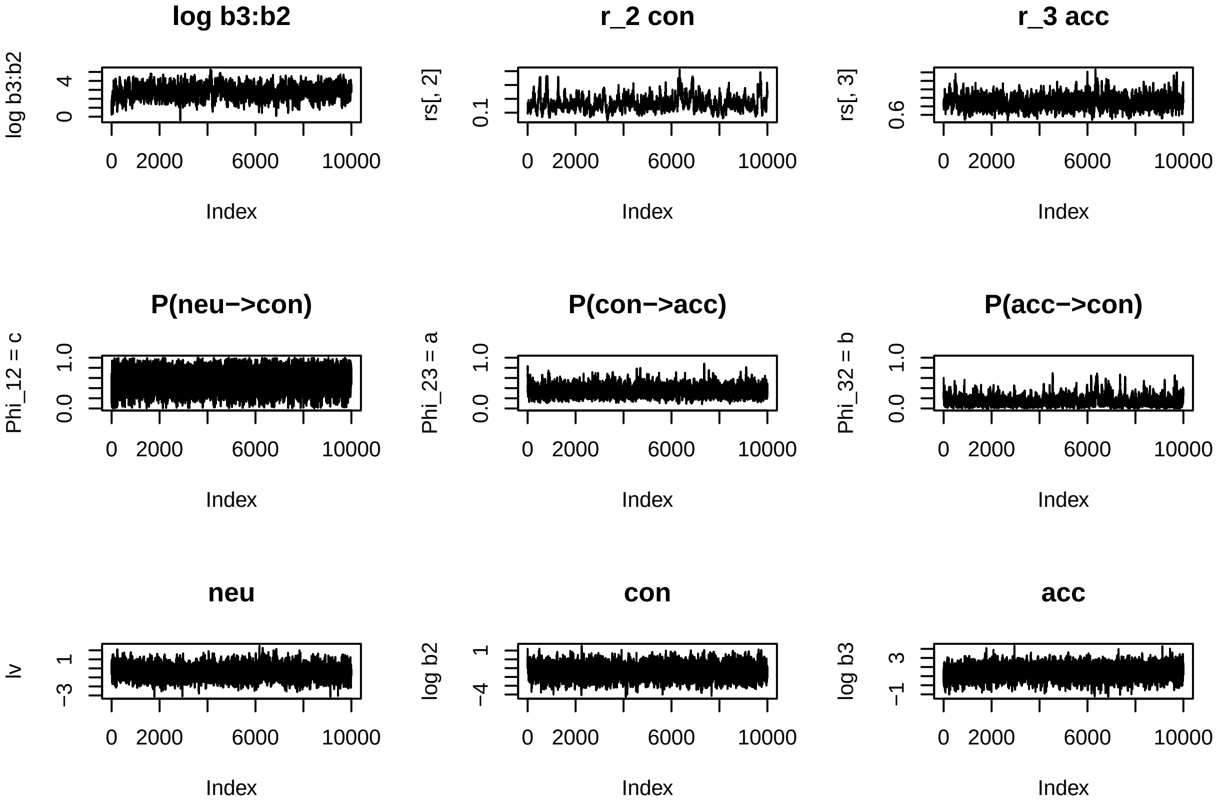
```
# Load a multiple sequence alignment.
alignment = ape::read.dna("VCE138247.fna", format="fasta", as.matrix=T, as.character=T)
```

Now we are ready to run the PhyloAcc-C model by using the command `run_model`. The command has a lot of parameters (to get information on the commands in the PhyloAcc-C package type `?pac3`) but we can take advantage of some defaults in this case. Note that we pass in the data we just loaded, and also that we ask for 10,000 iterations. We also set a Beta uniform prior on transition probabilities between different latent conservation states (`Phi_prior`).

```
set.seed(7)
mcmc_out = pac3::run_model(tree=tree, alignment=alignment,
                           trait_tbl=trait_tbl,
                           transition_matrix=Q, stationary_distro=PI,
                           n_iter=10000,
                           Phi_prior=c(1.0, 1.0, 1.0, 1.0, 1.0, 1.0))
```

After some time the `mcmc_out` variable will be assigned the result of running the model. First it is a good idea to plot the trace to see if the model has mixed. (For a basic tutorial on Bayesian modelling using MCMC, I like Lunn et al. (2013), but a Google search will bring up lots of free online resources.)

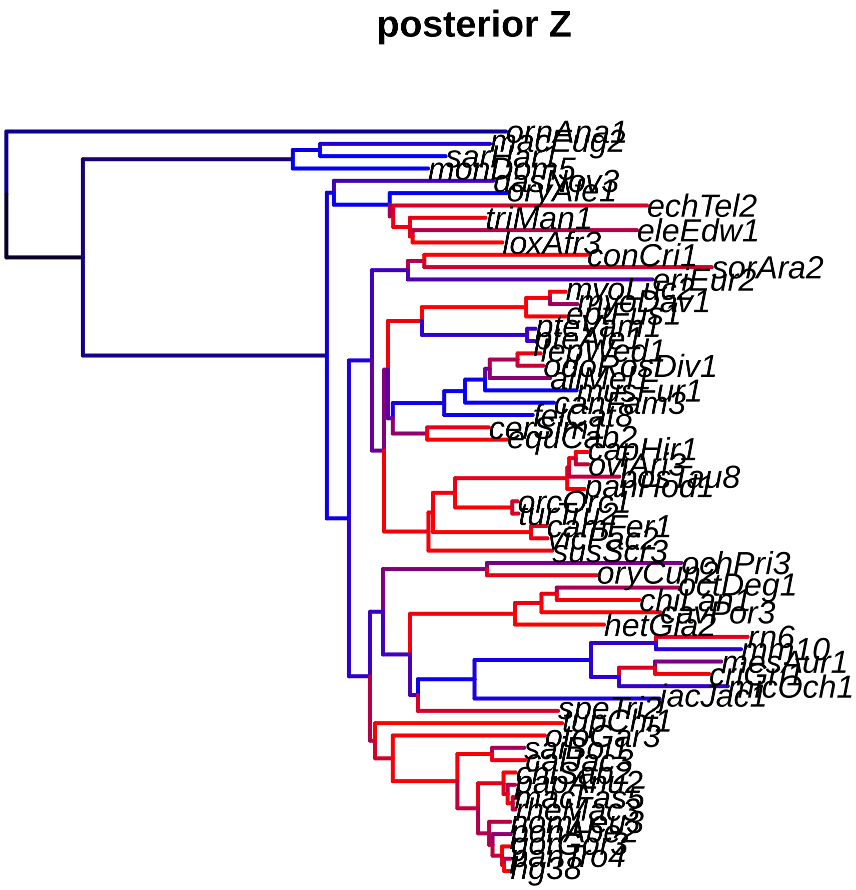
```
pac3::plot_mixing(mcmc_out)
```



It looks like r_3 is mixing well but r_2 seems to mix more slowly; it would be good to run the chain for longer, or to run multiple chains, but let's go to the next step for demonstration purposes.

One thing we might be interested in is the CNE specific latent conservation states across the tree. Command `plot_z` displays a high posterior probability of acceleration as red, conservation as blue, and background/neutral as black.

```
pac3::plot_z(mcmc_out, drop_n=5000)
```



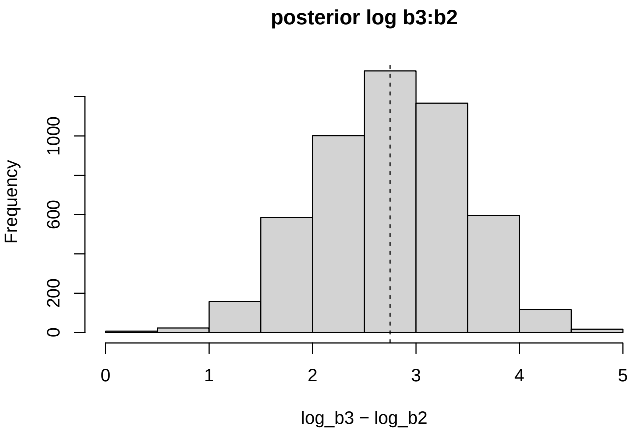
Note that the `plot_z` command is using the trace output from the MCMC implicitly but that you can access these values directly. There is a `z_trace`, an `r_trace`, a `Phi_trace`, and a `y_trace`, covering model parameters of interest. For example, to access samples directly related to trait movement look at `mcmc_out$mcmc_trace$v_trace`. The last few samples are:

```
tail(mcmc_out$mcmc_trace$v_trace)
```

```
##                 log.v  log.beta2 log.beta3
##  [9995,] -0.4038984 -1.3208397  2.165732
##  [9996,] -0.1523930 -2.0872757  1.168345
##  [9997,] -0.7801980 -1.1163548  1.716989
##  [9998,] -0.3006057 -1.6603054  1.291640
##  [9999,] -0.6368271 -0.9920936  1.873598
## [10000,] -0.5408662 -1.7616650  1.781058
```

The last plot we make will tell us about what we are really interested in: how does the rate of evolution of the ELL trait covary with the substitution rate of CNE VCE138247? PhyloAcc-C answers this question by reporting the posterior ratio $\log \beta_3 - \log \beta_2$; a histogram can be made using `plot_v`.

```
pac3::plot_v(mcmc_out, drop_n=5000)
```



In this case we can see the ratio is positive, so that when the element evolves quickly, so too does the trait. This can also be seen by looking at the trace directly.

```
vs = mcmc_out$mcmc_trace$v_trace[-(1:5000),]
quantile(vs[,3]-vs[,2], probs=c(.05, .5, .95))
```

```
##       5%      50%      95% 
## 1.590286 2.777239 3.837008
```

An alternative scenario would have been a negative log ratio, which would suggest the opposite, that when the trait evolves quickly the element tends to be conserved. Of course, most CNEs in the genome have a log ratio that is centred on 0, whereby there is no strong relationship either way.

Multiple chains

To improve confidence in MCMC output it is a good idea to run multiple independent chains. This example shows how to run three independent chains, combine their results, and apply the Gelman and Rubin convergence diagnostic. To perform these steps you will need the `coda` package. For more information on the diagnostic refer to the coda manual (type `?coda::gelman.diag`).

```
# Run the PhyloAcc-C model 3 times using the same data and priors as above.
set.seed(42)
n_runs = 3
mcmc_outs = replicate(n_runs,
                      pac3::run_model(tree=tree, alignment=alignment,
                                      trait_tbl=trait_tbl,
                                      transition_matrix=Q, stationary_distro=PI,
                                      n_iter=10000,
                                      Phi_prior=c(1.0, 1.0, 1.0, 1.0, 1.0, 1.0)),
                      simplify=F)

# A helper function to pack up the traces into the format coda likes.
bundle_trace = function(mcmc_out) {
  mcmc_trace = mcmc_out$mcmc_trace
  coda::as.mcmc(cbind(mcmc_trace$r_trace[,-1], # r_1 is always 1, so exclude
                      mcmc_trace$v_trace,
                      mcmc_trace$Phi_trace))
}

# A list of coda objects to diagnose.
mcmc_list = coda::as.mcmc.list(lapply(mcmc_outs, bundle_trace), start=1000)
```

```
# A common MCMC diagnostic.
coda::gelman.diag(mcmc_list)
```

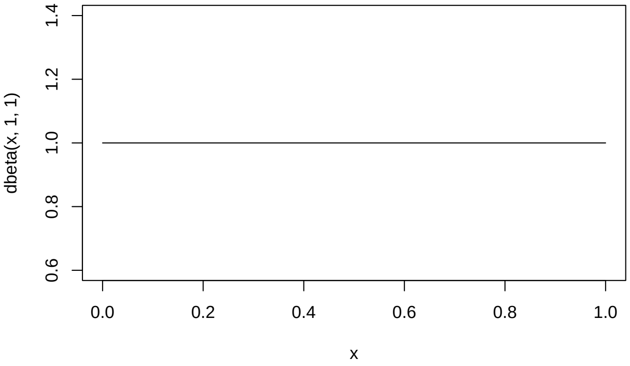
```
## Potential scale reduction factors:
##
##            Point est. Upper C.I.
## r2               1.02       1.04
## r3               1.00       1.01
## log.v            1.00       1.00
## log.beta2        1.00       1.01
## log.beta3        1.00       1.00
## phi.nc           1.00       1.00
## phi.ca           1.00       1.01
## phi.ac           1.02       1.02
##
## Multivariate psrf
##
## 1.01
```

Prior selection and simulation

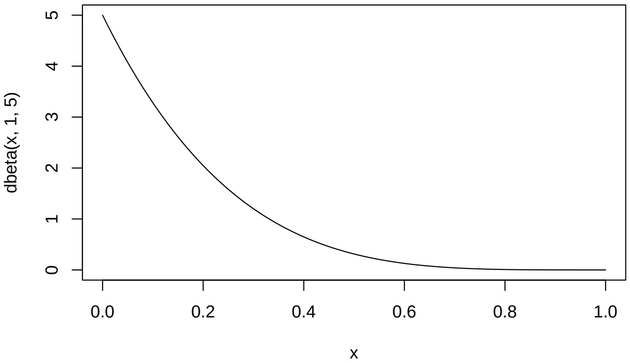
Simulation is useful for several reasons. For example, before running PhyloAcc-C on many elements, which will be time consuming, it is a good idea to see if the model will perform on the phylogeny you have, given the patterns of evolution you expect to see. Simulation can also help you understand the PhyloAcc-C priors.

Let's think about how to set priors. Transitions between latent conservation states are controlled by per-branch switching probabilities and managed using beta priors. You might set uniform priors on switching probabilities or perhaps be skeptical that conservation states will switch often. A useful way to understand your prior choice is to plot it with the `curve` function.

```
curve(dbeta(x,1,1)) # Any probability of switching is a priori equally likely.
```



```
curve(dbeta(x,1,5)) # Expect high switching probabilities to be rare.
```



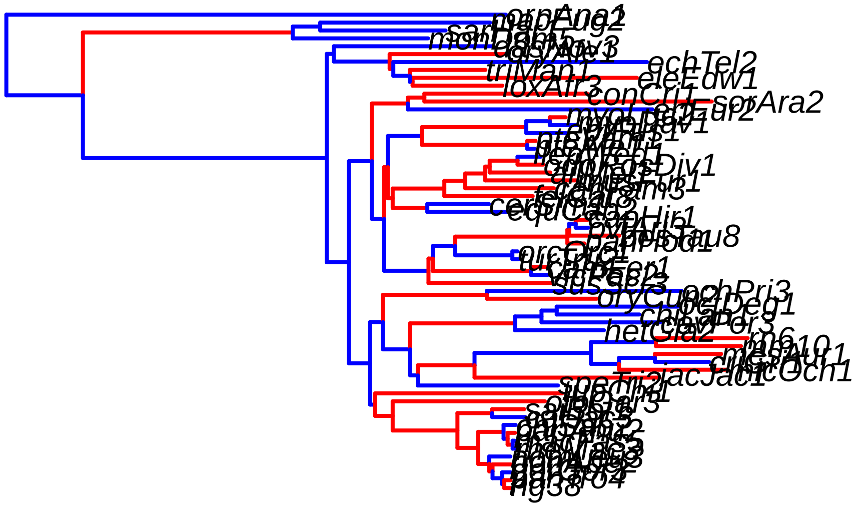
After selecting switching probabilities, you can bundle them into a single list as follows.

```
# Beta priors are put on z switches via Phi matrix:
# 1 1 is uniform, 1 5 is probably 1/3 or less, etc
# neutral -> conserved subs rate
prior_nc_a = 1 # c : conserved in paper
prior_nc_b = 1
# conserved -> accelerated subs rate
prior_ca_a = 1 # a : accelerate in paper
prior_ca_b = 1
# accelerated -> conserved subs rate
prior_ac_a = 1 # b : back in paper
prior_ac_b = 1

# pack them up
prior_Phi = c(prior_nc_a, prior_nc_b,
              prior_ca_a, prior_ca_b,
              prior_ac_a, prior_ac_b)
```

You will also need to specify a prior on the conservation state of the branch leading to the root node.

```
# What state is the root in? A categorical prior.
prior_ZR = c(.5,.5, 0) # In this case, fifty-fifty background or conserved.
```

At this point we have specified the Markov process that assigns latent conservation states to branches on the tree. We can investigate the consequences of our priors by simulation. I would encourage you to run the following code several times to get a feel for the model you are specifying. Each time you run it you will get a plot of your phylogeny where the branches are coloured red (accelerated), blue (conserved), or black (background).

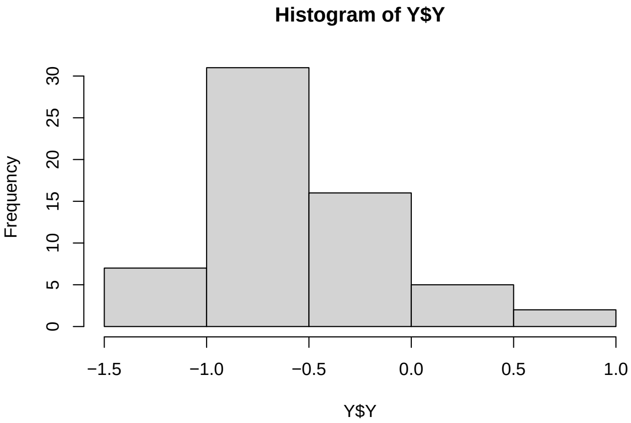


The conservation states can be extracted from the list returned by `sim_z`, and then passed on to `sim_y` for simulating trait movement and `sim_X` for simulating nucleotide alignments.

```
# We can retrieve the names of the background, conserved, and accelerated branches.
nb = names(Z)[Z==1] # background/neutral
cb = names(Z)[Z==2] # conserved
ab = names(Z)[Z==3] # accelerated
```

Let's think about trait evolution. In this case we will simulate with some fixed parameters, whereby the baseline variance rate σ^2 is 1, the trait evolves at one-fifth that rate on conserved branches ($\beta_2 = 0.2$), and twice that rate on accelerated ones ($\beta_3 = 2$). (This is a large effect!) We can simulate the evolution of the trait under these parameters by passing them, along with our conservation states, to `sim_y`.

```
param_v  = 1
param_b2 = 0.2
param_b3 = 2

Y = pac3::sim_y(tree, neu_branches=nb, con_branches=cb, acc_branches=ab,
                v=param_v, b2=param_b2, b3=param_b3)
hist(Y$Y)
```

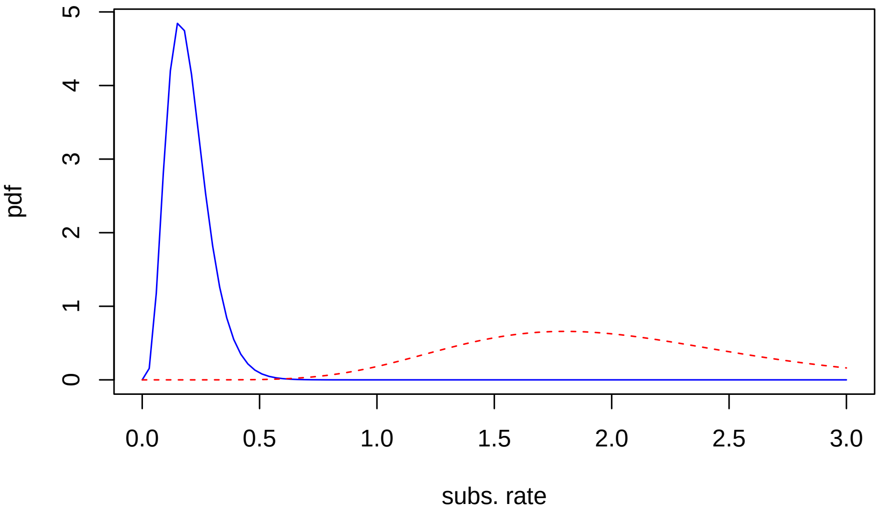


Parameters σ^2, β_2, and β_3 control variance, which must be positive, and their priors are specified on the log scale. For the tutorial data, I think the following values are reasonable priors; they put effect sizes of 10x (greater or smaller) at a chance of about 1/50.

```
# Mean and variance for log baseline sigma^2.
prior_lv_u  = 0.0  # log baseline trait movement on background (z=1) branches
prior_lv_sd = 2.0

# Mean and variance for log beta_2.
prior_lb2_u  = 0.0 # log multiplier for conserved (z=2) branches
prior_lb2_sd = 1.0

# Mean and variance for log beta_3.
prior_lb3_u  = 0.0 # log multiplier for accelerated (z=3) branches
prior_lb3_sd = 1.0

# Mean and variance for ancestral value of the trait.
prior_root_u  = 0.0 # ancestral trait at root
prior_root_sd = 1.0
```

To simulate an alignment under the PhyloAcc-C model we need priors on nucleotide substitution rate multipliers r_2 and r_3. For the mammal tree we are using, values based on an analysis of mammalian alignments by Hu et al. (2019) are sensible. Priors on nucleotide substitution rates are set using a gamma distribution.

```
cprior_a =  5.0; # gamma hyperprior on con rate
cprior_b =  0.04;

nprior_a = 10.0; # gamma hyperprior on acc rate
nprior_b =  0.2;

# This is what the priors like.
curve(dgamma(x, 5, 1/0.04), from=0, to=3, col="blue", xlab="subs. rate", ylab="pdf")
curve(dgamma(x,10, 1/0.20), add=T, lty=2, col="red")
```



Simulation is performed using the `sim_X` function. As with the trait simulation, we need to pass information on latent conservation states. In this example I select a conserved rate multiplier of 0.2 and an accelerated rate multiplier or 1.2. These are not atypical under the gamma priors.

```
# We can simulate an alignment too.
X_res = pac3::sim_X(PI=PI, Q=Q, tree=tree,
                    neu_branches=nb, con_branches=cb, acc_branches=ab,
                    con_m=0.2, # r_2 in the paper
```

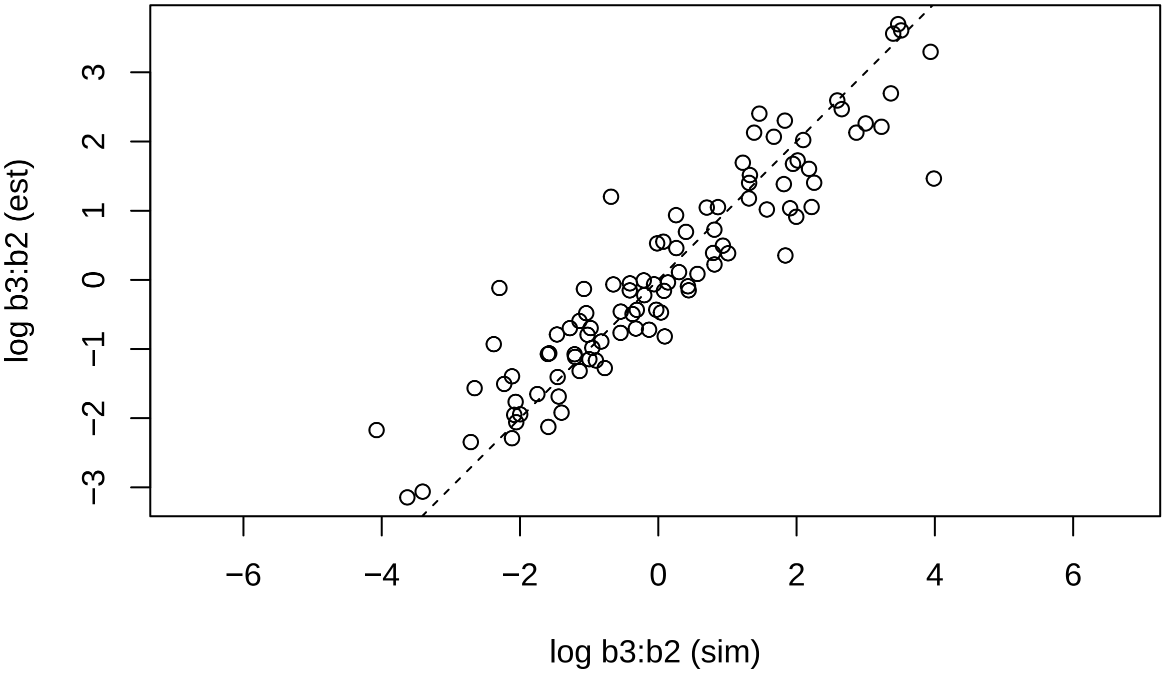
```
                         acc_m=1.2, # r_3 in the paper
                         S=200)     # simulate 200 sites
```

Using the above techniques, you may want to simulate many replicate alignments, traits, and latent conservation states to explore candidate priors for your own use of the software. You can use the replicates to check the model can potentially generate the kind of data you have.

Lastly, to run the model on simulated data, you can straightforwardly pass in the results of the simulations to the `run_model` function like so:

```
mcmc_out = pac3::run_model(tree=tree, alignment=X,
                           trait_tbl=Y,
                           transition_matrix=Q, stationary_distro=PI,
                           n_iter=10000,
                           Phi_prior=prior_Phi)
```

I ran 100 simulations, picking parameters from the above priors, with the following results. On the x axis are the "true" simulated ratios $\log \beta_3 - \log \beta_2$ and on the y axis are the values estimated using `run_model`. The relationship is not perfect, but reasonable enough to warrant studying real data, I think.



A final note. Using simulations tells you how well inference works when data is generated from the model. Data such as lifespan is not generated from the model but from some unknown real-world process. Therefore performance on simulated data is ultimately meaningful only to the extent you believe the model captures the salient aspects of the real-world.

Bayes factors

Think of PhyloAcc-C as a torch (flashlight) that might highlight CNEs of interest with respect to a given trait. After running PhyloAcc-C on a set of, say, 1000 elements, how would you determine which are most interesting?

There are several approaches to this question, but one answer is to use Bayes factors to see how much more likely data is under a "full model" (M1), linking trait evolution to molecular evolution, versus a null model (M0), where there is no systematic link. You could then follow-up on the elements with the highest Bayes factors.

In the full model, variance multipliers β_2 and β_3 are free to vary, whereas in the restricted null model $\beta_2 = \beta_3 = 1$, and the trait always evolves at a background rate regardless of latent conservation states. The null hypothesis is nested inside the full hypothesis, and the priors on β_2 and β_3 are common to all candidate elements. Therefore the Bayes factors can be estimated using the ratio of prior to posterior density of $(\log\beta_2, \log\beta_3)$ at point $(0, 0)$.

This point hypothesis test can be conducted on a per-element basis by using a density estimation package and the traces produced by PhyloAcc-C. There are more than a dozen such packages, but we will use `sm` here.

Simulation can help illustrate the idea. Let's first simulate latent conservation states and an alignment.

```r
# A fixed tree.
tree = ape::read.tree("mammal_tree.txt")

# A fixed rate matrix.
Q <- matrix(
  c(-1.075162,  0.186970,  0.696268,  0.191923,
     0.181082, -0.873473,  0.255492,  0.436899,
     0.674340,  0.255493, -1.164645,  0.234813,
     0.191924,  0.451108,  0.242449, -0.885481),
  nrow=4, ncol=4)

# The associated fixed stationary distribution.
PI <- c(0.246000, 0.254000, 0.254000, 0.246000)

set.seed(1215)

# Simulated latent conservation states.
Z = pac3::sim_z(tree, Z_R_prior=c(1, 0, 0), P_nc=0.5, P_ca=0.2, P_ac=0.2)

nb = names(Z)[Z==1]
cb = names(Z)[Z==2]
ab = names(Z)[Z==3]

# Simulated alignment.
X = pac3::sim_X(PI=PI, Q=Q, tree=tree,
                neu_branches=nb, con_branches=cb, acc_branches=ab,
                con_m=0.2, acc_m=2.0, S=200)
```

Now let's simulate two sets of trait data. The first set, T0, is drawn without regard to the conservation state of the branch. The second set, T1, is drawn so that the trait moves 10 times as quickly on accelerated branches as conserved branches (log b3:b2 is about 2.3). We run the model twice; the inputs are identical except for the trait data.

```r
# Simulated trait data under the null model M0.
T0 = pac3::sim_y(tree, neu_branches=nb, con_branches=cb, acc_branches=ab,
                 v=1, b2=1, b3=1, yR=0) # v = v*b2 = v*b3

# Simulated trait data under the full model M1.
T1 = pac3::sim_y(tree, neu_branches=nb, con_branches=cb, acc_branches=ab,
                 v=1, b2=0.2, b3=2, yR=0) # b3 : b2 = 10

mcmc_M0 = pac3::run_model(tree=tree, alignment=X,
                          trait_tbl=T0, # M0 data
                          transition_matrix=Q, stationary_distro=PI,
                          n_iter=10000,
                          prior_lb2=c(0.0, 1.0),
                          prior_lb3=c(0.0, 1.0),
                          Phi_prior=prior_Phi)

mcmc_M1 = pac3::run_model(tree=tree, alignment=X,
                          trait_tbl=T1, # M1 data
                          transition_matrix=Q, stationary_distro=PI,
                          n_iter=10000,
                          prior_lb2=c(0.0, 1.0),
                          prior_lb3=c(0.0, 1.0),
                          Phi_prior=prior_Phi)
```

To calculate Bayes factors let's introduce a helper function. It takes a trace, estimates the posterior density of $(\log\beta_2, \log\beta_3)$ at $(0, 0)$, and uses this to estimate a Bayes factor, given our Normal$(0, 1)$ priors.

```
calculate_BF = function(mcmc_out) {
  # Get the trace of log b2 and log b3, discarding first 1000 rows as burn-in.
  v_trace = mcmc_out$mcmc_trace$v_trace[-(1:1000),]
  # Ask sm for a density estimate at (0, 0).
  d00 = sm::sm.density(v_trace[,2:3], model="Normal", eval.grid=F,
                   eval.points=matrix(data=c(0,0), byrow=T, nrow=1, ncol=2),
                   display="none")
  true_null = 1/(2*pi) # The prior density of (log b2, log b3) at (0,0).
  true_null/d00$estimate # Bayes factor is ratio of densities at (0,0)
}

calculate_BF(mcmc_M0)
```

```
## [1] 0.6802666
```

```
calculate_BF(mcmc_M1)
```

```
## [1] 104.7438
```

For the inference using the data in `mcmc_M0` the Bayes factor is less than one, indicating we should increase our belief that the rates of phenotypic and molecular evolution *are not* linked. For the inference using the data in `mcmc_M1`, a Bayes factor of 105 suggests we should drastically increase our belief that the rates of phenotypic and molecular evolution are linked. Elements like the latter should be rare candidates for followup when real world data is studied. (For reference, I found well under 100 such elements out of hundreds of thousands I studied in mammals.)

Note that Bayes factors quantify how much "better" M1 is than M0, a relative measure. A "very bad" fit will have a higher Bayes factor than a "atrocious" fit. Even in the best of circumstances, neither M1 or M0 will capture evolution exactly. For this reason, and others, I recommend you see Bayes factors as a useful heuristic for ranking elements rather than specific values you take too seriously.

In general it is sensible to include as much outside knowledge as you can when interpreting the result of running PhyloAcc-C on a collection of elements. For example, if an element with a high Bayes factor has two dozen separate acceleration events you might question whether the model's assumption of a single accelerated rate category makes sense. Or, if the alignment of an element with a high Bayes factor looks bad, you should worry about the uncertainty of the latent rate categories that are derived from it.

Approachable information on Bayes factors can be found in the tutorial by Wagenmakers et al. (2010).

References

Hu, Z., Sackton, T. B., Edwards, S. V., and Liu, J. S. (2019). Bayesian detection of convergent rate changes of conserved noncoding elements on phylogenetic trees, Molecular biology and evolution, 36(5), 1086–1100.

Kowalczyk, A., Partha, R., Clark, N. L., & Chikina, M. (2020). Pan-mammalian analysis of molecular constraints underlying extended lifespan. Elife, 9, e51089.

Lunn, D., Jackson, C., Best, N., Thomas, A., & Spiegelhalter, D. (2013). The BUGS book. A practical introduction to Bayesian analysis, Chapman Hall, London.

Wagenmakers, E. J., Lodewyckx, T., Kuriyal, H., & Grasman, R. (2010). Bayesian hypothesis testing for psychologists: A tutorial on the Savage–Dickey method. Cognitive psychology, 60(3), 158-189.