R Markdown with simulation example

Daniel Caetano and Tiago Quental

5/16/2022

This Markdown document will show how to install and use the "buddPhy" package for R. This package allows for simulation of multiple scenarios of budding speciation for both discrete and continuous traits. The package also offers functions for plotting the budding history of lineages and the distribution of rates of trait evolution under several scenarios of lineage-age dependent rates, budding and cladogenetic events associated with asymmetric speciation. The package is currently available through github.

Package installation

First install the package dependencies:

```
install.packages("viridisLite")
install.packages("ape")
install.packages("expm")
install.packages("FossilSim")
install.packages("ratematrix")
install.packages("phytools")
```

The package can be installed from the github repository.

```
library( devtools )
install_github(repo = "Caetanods/buddPhy")
```

```
library( buddPhy )
```

Generating a phylogeny

The first step is to obtain a phylogenetic tree. `buddPhy` will trace the evolutionary history of the lineages through the branches of an existent phylogeny, but it will not generate a new one. This is useful because the user can produce lineage histories based on their empirical phylogenetic tree without the need to modify it in anyway.

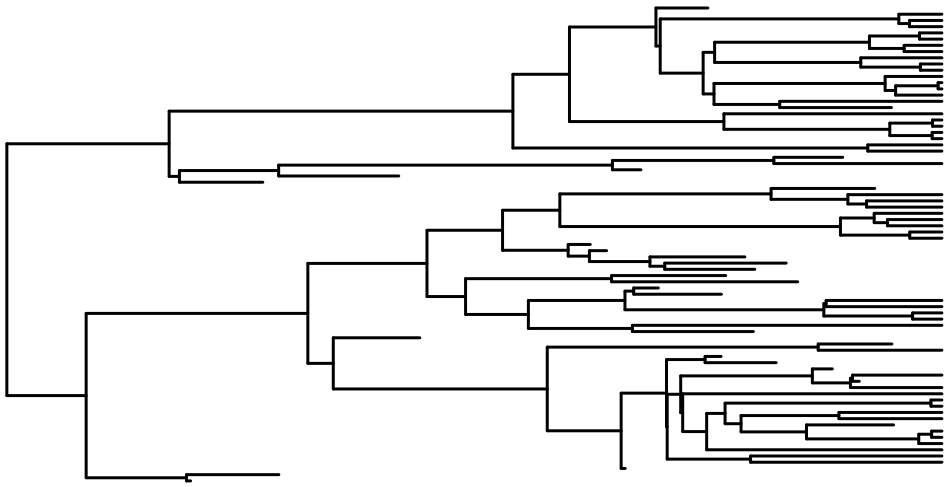
If a phylogenetic tree is not available, then one can be simulated using a variety of packages. Here we are using `TreeSim`, but note that other packages such as `geiger`, `phytools`, and `ape` could be used as well.

```
library( TreeSim )
```

```
## Loading required package: ape
```

```
## Loading required package: geiger
```

```
phy <- sim.bd.taxa(n = 50, numbsim = 1, lambda = 0.02, mu = 0.01, complete = TRUE)[[1]]
plot.phylo(x = phy, show.tip.label = FALSE, edge.width = 1.5)
```



Simulating discrete trait histories

`buddPhy` will generate the history of the lineages together with the history of a discrete or a continuous trait. Let's start by simulating a discrete trait. For this first we need to create a Q matrix with the rates of transition between states:

```
Q <- rbind(c(-0.06, 0.03, 0.03), c(0.03, -0.06, 0.03), c(0.03, 0.03, -0.06))
colnames(Q) <- rownames(Q) <- 1:3
```

We will use this matrix to generate the history for 3 states. Note that we have symmetric evolution. Now we can use the `buddPhy::sim_Mk_budding_exp` to generate the trait history:

```
sim <- sim_Mk_budding_exp(tree = phy, Q = Q, budding_prob = 0.25, change_rate = 0.2
                    , cladogenetic_change = "prob", cladogenetic_prob = 0.5
                    , cladogenetic_state = "other")
```

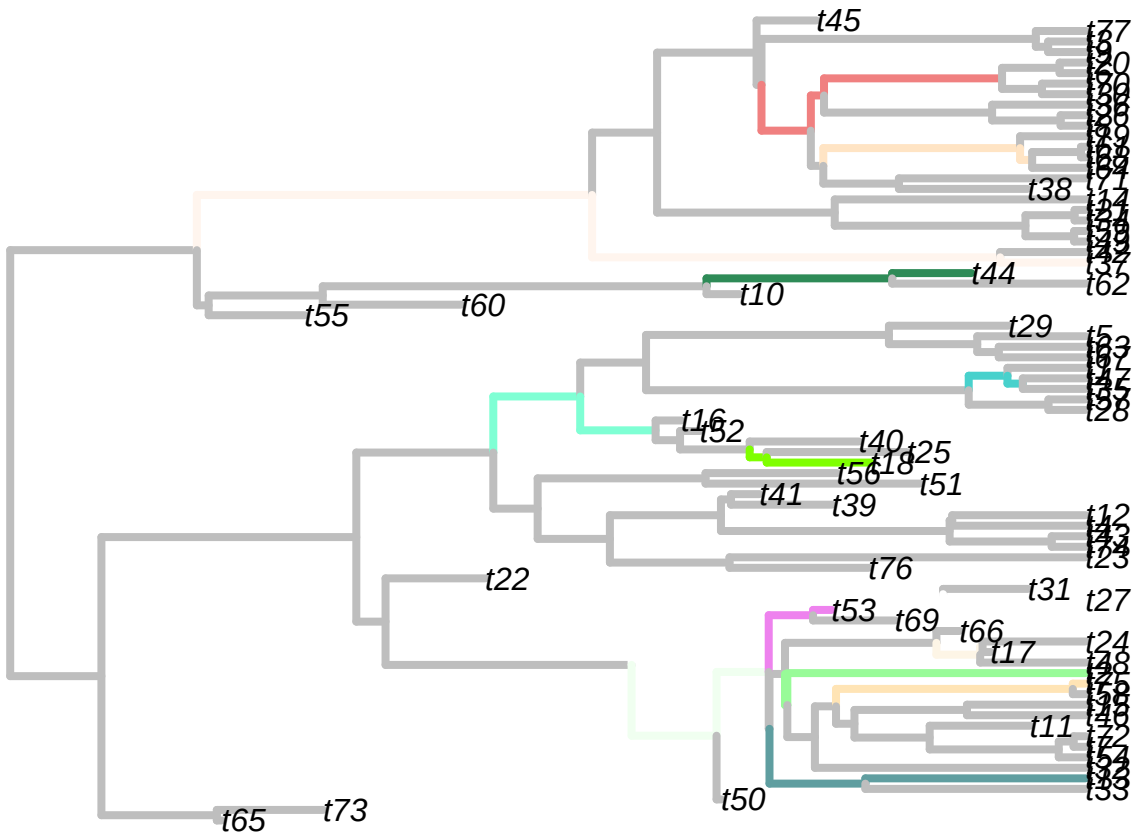
The line of code above is controlling many aspects of the simulation. `budding_prob` controls the probability that a budding event will happen at a given node of the tree. `change_rate` is the mean value of the exponential decay function which reduces rates of trait evolution in function of lineage-age. In this model the older lineages show slower rates of trait evolution than the new lineages. `cladogenetic_change`

is the type of trait change that happens at the event of budding. Here the user can choose between making a cladogenetic change ("prob" or "flat") or nothing special at the nodes (i.e., anagenetic evolution only). `cladogenetic_prob` is the probability that a cladogenetic event will happen when a node is assigned to be associated with a budding speciation event. Setting this parameter to 0.5 means that an event of cladogenetic change will only happen about half of the times that a budding event takes place. Finally, the `cladogenetic_state` parameter controls which state can appear after an event of budding. Some of the options are "other" (a state distinct from the state of the progenitor lineage) and "any" (any state, including the same state of the progenitor lineage).

Please refer to the help page for the `sim_Mk_budding_exp` function for more information about the options.

After performing the simulation we can plot the history of the lineages. Note that lineages with distinct colors are different. Colored branches indicate progenitor lineages which produced one or more daughter lineages through budding. The collection of colors are randomized.

```
plot_mother_lineages(sim_obj = sim, background_color = "gray", edge_width = 4)
```



We can also summary the history of the internal nodes and indicate which of the nodes were budding speciation events:

```
budd_hist <- buddPhy::get_Budding_History(sim = sim)
## The list of budding nodes:
budd_hist$budd_nodes
```

```
##  [1]  98 118 149 109 130 111 124  88 105 136 120  89 134 137 110 114
```

```
## A logic vector that can be used to perform further analyses:
budd_hist$budd_edges
```

```
##   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE
##  [13] FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [25] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE  TRUE
##  [37] FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE
##  [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE
##  [73]  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
##  [85]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [97] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
## [109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [121] FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE
## [133] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [145] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

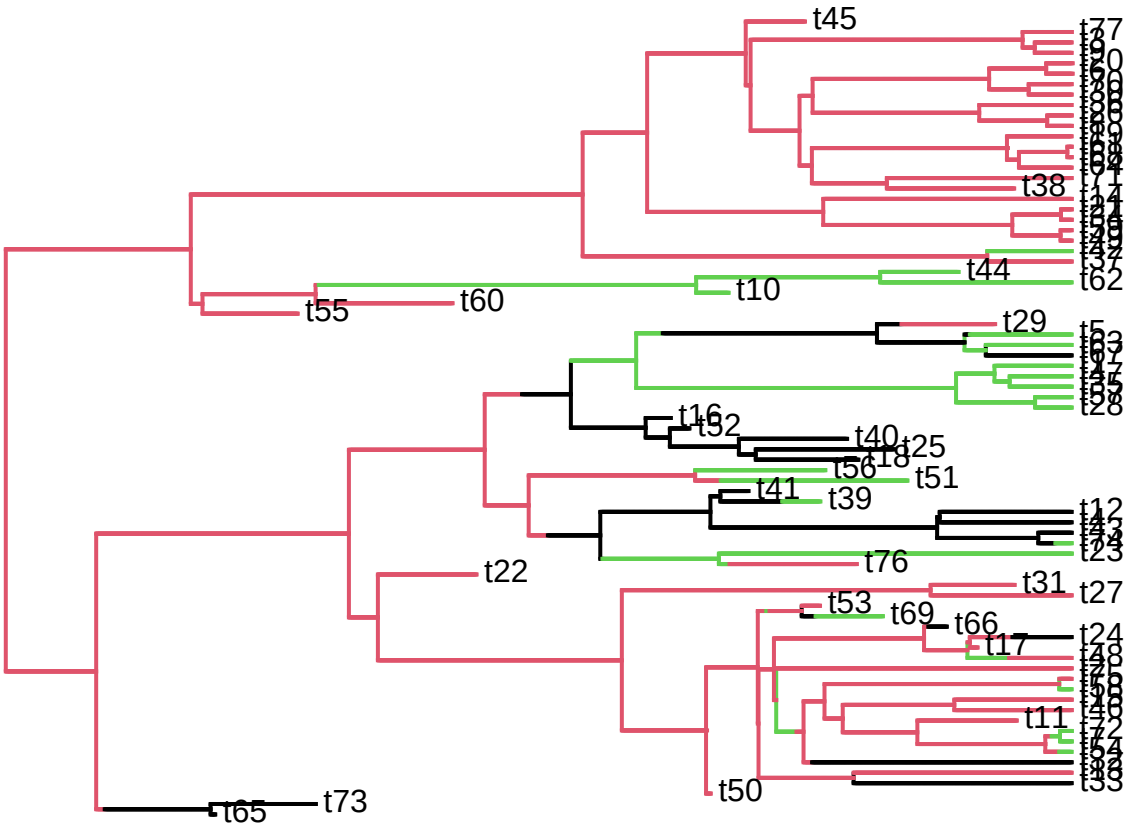
We can also plot the history of the traits. This is similar to a stochastic map, but here we are showing the true history of the trait and not an estimate of ancestral states.

```
library( phytools )
```

```
## Loading required package: maps
```

```
## The sim object is a list with many elements, including a stochastic map
##      with the true history of the simulation.
plotSimmap(tree = sim$simmap)
```

```
## no colors provided. using the following legend:
##         1         2         3
##   "black" "#DF536B" "#61D04F"
```



Of course, we can also extract the states for each tip of the tree:

```
sim$tip_state
```

```
##  t5 t21  t4 t72 t67 t54 t61 t19 t26 t24 t58  t2 t20 t37 t13 t34  t1 t35 t71 t14
##   3   2   1   3   1   3   2   2   2   1   3   2   2   2   2   2   3   3   2   2
## t23 t33 t77 t43 t36 t70 t57  t7 t46 t63  t9 t64 t47 t68 t59 t48 t15 t12 t49 t28
##   3   1   2   1   2   2   3   3   2   3   2   2   3   2   2   2   2   1   2   3
## t74 t27 t75  t3  t6 t30  t8 t62 t42 t32 t11 t31 t38 t29 t17 t44 t66 t51 t25 t69
##   3   2   2   2   2   2   2   3   3   1   2   2   2   2   2   3   1   3   1   3
## t18 t76 t40 t56 t39 t53 t45 t41 t10 t50 t52 t16 t22 t60 t73 t55 t65
##   1   2   1   3   3   2   2   1   3   2   1   1   2   2   1   2   1
## Levels: 1 2 3
```

Plotting the rates of trait evolution

In `buddPhy`, the age of the lineages influence the rate of trait evolution. Thus, rates of trait evolution vary across the tree. This is not a rate-homogeneous model of trait evolution. The distribution of rates will vary in function of the exponential decay factor and the probabilities of budding speciation.

We can easily plot the variation in the rates of trait evolution and also extract this information to make further analyses.

```
## The function will produce a phylogenetic tree in a simmap format and also
##      information for the colors.
```

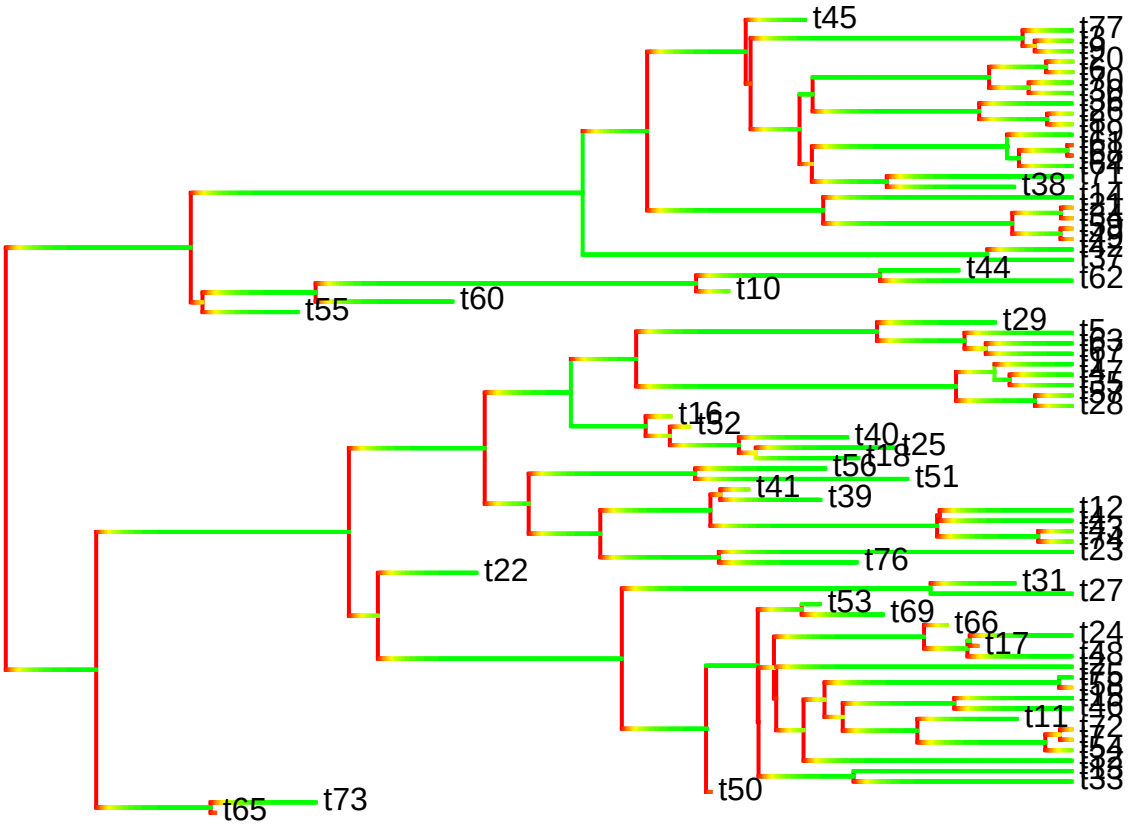
```
rate_tree <- make_scaler_simmap(sims = sim, ncat = 20)
class( rate_tree$simmap )
```

```
## [1] "phylo"  "simmap"
```

```
head( rate_tree$col_vec )
```

```
##         A         B         C         D         E         F
## "#00FF00" "#1AFF00" "#35FF00" "#50FF00" "#6BFF00" "#86FF00"
```

```
## Then we can use the plotSimmap function to produce the desired figure:
plotSimmap(tree = rate_tree$simmap, colors = rate_tree$col_vec)
```



In the figure above red colors indicate the fastest rate, which is equal to the original Q matrix used for the simulation. Then colors approaching green are slower rates of trait evolution. The decay function works by reducing the original rate of trait evolution by a factor.

We can check the actual values for the slowdown factor. These are also produced by the same function used above and stored as an element of the list:

```
rate_tree$legend_matrix
```

```
##          from         to label     col
## 1  0.00000000 0.04909519     A #00FF00
```

```
## 2  0.04909519 0.09819039     B #1AFF00
## 3  0.09819039 0.14728558     C #35FF00
## 4  0.14728558 0.19638078     D #50FF00
## 5  0.19638078 0.24547597     E #6BFF00
## 6  0.24547597 0.29457117     F #86FF00
## 7  0.29457117 0.34366636     G #A1FF00
## 8  0.34366636 0.39276156     H #BBFF00
## 9  0.39276156 0.44185675     I #D6FF00
## 10 0.44185675 0.49095195     J #F1FF00
## 11 0.49095195 0.54004714     K #FFF100
## 12 0.54004714 0.58914233     L #FFD600
## 13 0.58914233 0.63823753     M #FFBB00
## 14 0.63823753 0.68733272     N #FFA100
## 15 0.68733272 0.73642792     O #FF8600
## 16 0.73642792 0.78552311     P #FF6B00
## 17 0.78552311 0.83461831     Q #FF5000
## 18 0.83461831 0.88371350     R #FF3500
## 19 0.88371350 0.93280870     S #FF1A00
## 20 0.93280870 1.00000000     T #FF0000
```

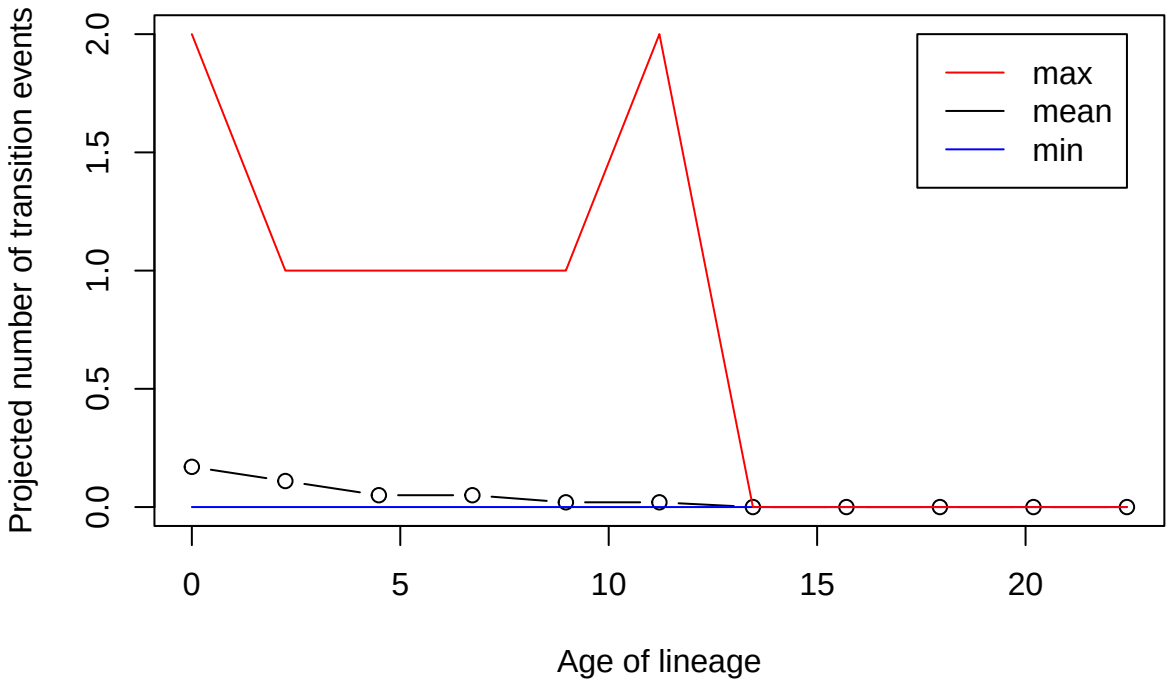
```
## The columns "from" and "to" have the limits for the categories associated with
##      each color on column "col". The "label" column is just a sequence of letters
##      to help plotting the information.
## The values of "from" and "to" are rate scalers. So the true transition rate for
##      each of these categories are:
ft_rate <- data.frame(from_rate = rate_tree$legend_matrix$from * Q[1,2]
                      , to_rate = rate_tree$legend_matrix$to * Q[1,2])
ft_rate
```

```
##      from_rate     to_rate
## 1  0.000000000 0.001472856
## 2  0.001472856 0.002945712
## 3  0.002945712 0.004418568
## 4  0.004418568 0.005891423
## 5  0.005891423 0.007364279
## 6  0.007364279 0.008837135
## 7  0.008837135 0.010309991
## 8  0.010309991 0.011782847
## 9  0.011782847 0.013255703
## 10 0.013255703 0.014728558
## 11 0.014728558 0.016201414
## 12 0.016201414 0.017674270
## 13 0.017674270 0.019147126
## 14 0.019147126 0.020619982
## 15 0.020619982 0.022092838
## 16 0.022092838 0.023565693
## 17 0.023565693 0.025038549
## 18 0.025038549 0.026511405
## 19 0.026511405 0.027984261
## 20 0.027984261 0.030000000
```

We can investigate the pattern created by the slowdown function by plotting the scaling factor in function of the age of the lineage. Below we will use the average branch length as the age of the lineage as a point of reference.

```
mean_br <- mean(phy$edge.length)
rate_decay <- simulate_rate_decay(change_rate = 0.2, start_rate = Q
                                  , time_elapsed = mean_br, plot = TRUE
                                  , nsims = 100)
```

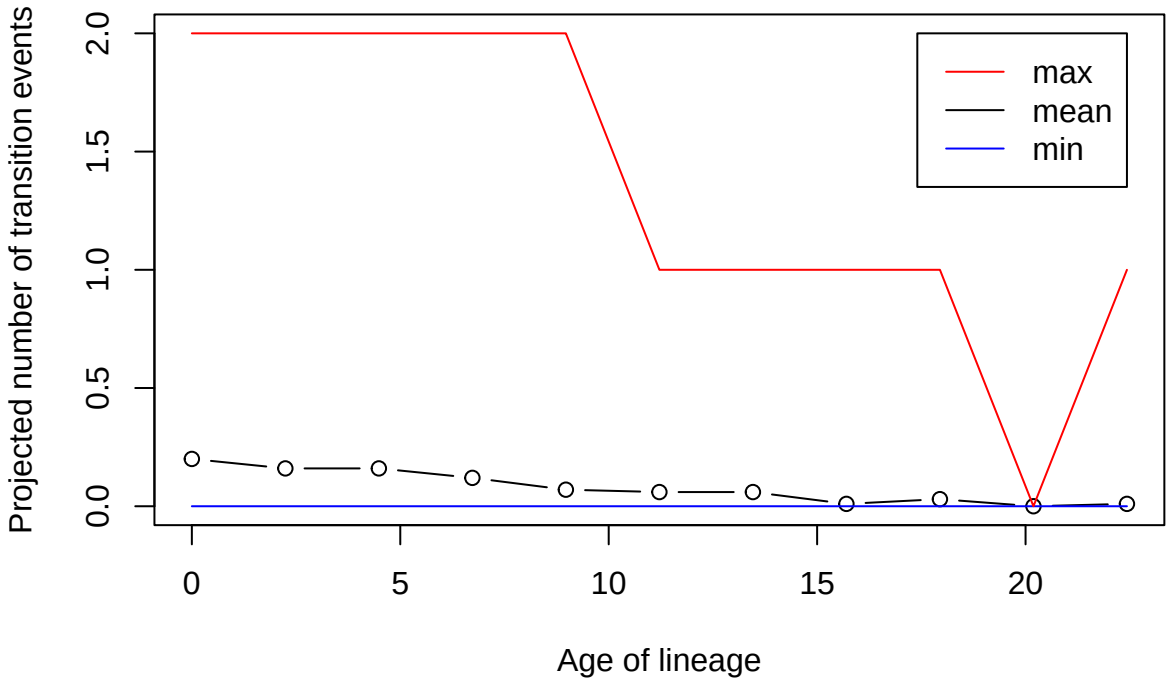
```
## [1] "Projecting expected number of transitions after 22.4324713275105 time units using a MK model."
```



The figure above shows how the expected number of transitions will decrease over time. If we change the `change_rate` we can control how fast this slowdown happens:

```
slow_rate_decay <- simulate_rate_decay(change_rate = 0.1, start_rate = Q
                                       , time_elapsed = mean_br, plot = TRUE
                                       , nsims = 100)
```

```
## [1] "Projecting expected number of transitions after 22.4324713275105 time units using a MK model."
```



Other functions in buddPhy

This was just a quick example of the type of functions that buddPhy has. Here are a list of the functions in the package. Please refer to their help pages for instructions of usage and what type of thing can be done with them.

```
## List of the functions and their parameters:
lsf.str("package:buddPhy")
```

```
## drop_fossils_track_node : function (phy)
## get_Budding_History : function (sim)
## make_phylip_data : function (data, states, file = "biogeobears_data.txt")
## make_scaler_simmap : function (sims, ncat = 10, low_cut, high_cut)
## mcmc_mk_budd_exp : function (phy, trait, k, prior_beta_Q = c(2, 5), prior_beta_budd = c(2,
##     5), prior_exp_rate = 1, sample_prob = c(0.25, 0.25, 0.25, 0.25), Q_prop = 0.2,
##     budd_prop = 0.2, rate_prop = 0.2, gen = 100)
## optim_base_rate : function (sim, target_mean_rate)
## plot_cont_sim : function (sim_obj, edge_width = 1.5, bg = "white", jump_col = "coral",
##     jump_width = 1, main = "", legend = TRUE)
## plot_mother_lineages : function (sim_obj, background_color = "gray", edge_width = 3, no_margin = TRU
## sim_BM_budding_exp : function (tree, sigma, anc = 0, budding_prob = 0, budding_mother = NA,
##     change_rate = 2, decay_fn = TRUE, jump_size = NA)
## sim_BM_trace_history : function (sim_BM, sigma, anc = 0, change_rate = 2, decay_fn = TRUE, jump_size
##     jump_type = "fixed", jump_all = FALSE)
## sim_Mk_budding_exp : function (tree, Q, anc = NULL, budding_prob = 0, budding_mother = NA, change_ra
```

```
##     decay_fn = TRUE, cladogenetic_change = "none", cladogenetic_state = "any",
##     cladogenetic_prob = 1)
## sim_Mk_trace_history : function (sim_MK, Q, anc = NULL, change_rate = 2, decay_fn = TRUE, cladogeneti
##     cladogenetic_state = "any", cladogenetic_prob = 1)
## simulate_rate_decay : function (change_rate, start_rate = NULL, time_elapsed, plot = TRUE, npoints =
##     nsims = 1000)
## tree_rescale_exp : function (change_rate = 0.05, budding_prob = 0.3, tree, chunk_fraction = 0.01,
##     decay_fn = TRUE)
```

Some highlights are `sim_Mk_trace_history` which can be used to simulate trait histories while holding the history of the lineages constant. Other interesting functions are `sim_BM_budding_exp` and `sim_BM_trace_history` which work similarly to the example above, but simulate a continuous trait under Brownian motion model.