Vol. 12 (2019) *Acta Physica Polonica B Proceedings Supplement* No 1

# MODELLING TRAIT-DEPENDENT SPECIATION WITH APPROXIMATE BAYESIAN COMPUTATION*

Krzysztof Bartoszek†

Department of Computer and Information Science, Linköping University, Sweden

Pietro Liò‡

Computer Laboratory, University of Cambridge, Cambridge, United Kingdom

*(Received December 31, 2018)*

Phylogeny is the field of modelling the temporal discrete dynamics of speciation. Complex models can nowadays be studied using the Approximate Bayesian Computation approach which avoids likelihood calculations. The field's progression is hampered by the lack of robust software to estimate the numerous parameters of the speciation process. In this work, we present an `R` package, **pcmabc**, publicly available on CRAN, based on Approximate Bayesian Computations, that implements three novel phylogenetic algorithms for trait-dependent speciation modelling. Our phylogenetic comparative methodology takes into account both the simulated traits and phylogeny, attempting to estimate the parameters of the processes generating the phenotype and the trait. The user is not restricted to a predefined set of models and can specify a variety of evolutionary and branching models. We illustrate the software with a simulation-reestimation study focused around the branching Ornstein–Uhlenbeck process, where the branching rate depends non-linearly on the value of the driving Ornstein–Uhlenbeck process. Included in this work is a tutorial on how to use the software.

DOI:10.5506/APhysPolBSupp.12.25

## 1. Introduction

The relationship between genotypes and phenotypes originates from a very complex spatial and temporal, non-linear dynamical system. Due to the quick and microscopic dynamics, parts of the developmental process are

* Presented at the Summer Solstice 2018 Conference on Discrete Models of Complex Systems, Gdańsk, Poland, June 25–27, 2018.

† Corresponding author: `krzysztof.bartoszek@liu.se`, `krzbar@protonmail.ch`

‡ `pl219@cam.ac.uk`

usually hidden from direct observation. Data on adult animals are rarely collected in a continuous manner. The characteristics of a phenotype depend on both genetic and environmental factors so the genetically identical individuals could have a different phenotype expressively due to the subtle dependency on environmental (exposomes) factors.

Phylogenetics occupies an extremely important position in the Modern Synthesis and it is central to most areas of biology. It bridges population genetics [1], genomics and cancer research [2]. Furthermore, there is hope that it may assist in discovering relationships between complex diseases [3].

In the last decade, the area of phylogenetics has been tremendously exposed to novel statistical and computational models previously adopted only in theoretical studies. Bayesian methodologies are now at the core of phylogenetic comparative methods (PCM — the study of phenotypic data on the between-species level, the trait measurements are dependent through the species' common evolutionary history) and are used to evaluate macroevolutionary hypotheses of phenotypic evolution under distinct evolutionary processes in a phylogenetic context.

In particular, Approximate Bayesian Computation (ABC) is a powerful methodology to estimate the posterior distributions of model parameters without evaluating the (usually computationally very costly) likelihood function [4, 5]. That property has widened the domains of application of Bayesian models (see *e.g.* [6] for recent developments of ABC approaches for evolutionary biology) and has offered interesting challenges in parameters' estimation tasks. This leads to an interesting consideration: biology, in particular DNA sequence analysis, has been central for the development of the ABC methodology [7], in return, biology represents a rich application domain for Bayesian analysis which could probably inspire further theoretical developments.

Another key factor for these new Bayesian theoretical frameworks has been the statistical computing language `R` [8] that is central to a community of scientists who have developed a wide range of tools and functions for phylogenetic comparative analyses. The development of software tools (see, for instance, `https://cran.r-project.org/web/views/Phylogenetics.html`) has grown in parallel with the perception of how the use of newly developed tools would facilitate the understanding of statistical and biological concepts, produce successful instances of phylogenetic inference and bridge the gap between mathematicians and biologists. Therefore, the large existing variety of tools reflects and accommodates the current rich interdisciplinary and multidisciplinary field of phylogenetic studies. To a beginner, phylogenetics as a field, represents the intersection of interesting questions, great data and powerful tools.

With the above considerations in mind, here we would like to address trait-dependent speciation models by means of phylogeny-based evolutionary dynamics. The vastity and complexity of the research theme is attacked by means of a general, although focused, tool based on powerful ABC approach. The beginner is accurately guided by a tutorial and a description of the package. The `R` code found in the tutorial is available from the authors on request.

The paper has the following structure. In the next section, 2, we introduce the **pcmabc** `R` package and the algorithms for simulation of traits and trees. In particular, in Section 2, we describe three novel phylogenetic algorithms required by the ABC inference procedure. Then, in Section 3, we describe a simulation study to evaluate whether the ABC inference package can capture any signal on the trait-dependent speciation process, based only on the contemporary sample and phylogeny. The tutorial in Section 4 is self-contained although it uses the same Ornstein–Uhlenbeck (OU) process as in Section 3. We end the paper with Section 5 which summarizes the possibilities of the software, its limitations and possible directions of future development. Although we leave to the reader the choice of the order, we are delighted to report that the tool is not only serving the scope of studying by simulations the theory on evolution of traits or rapid prototyping the implementation of new hypotheses. It embeds a large generality towards a class of problems at the core of today's theoretical advancements in phylogenetics. Therefore, we aim in the future at collecting and presenting in a website the parameters and the biological results obtained using this methodology and software.

To facilitate reading, we have adopted the following fonts for the computer code. Programming language names are written in typewriter font (*e.g.* `R`), `R` package names are written in bold (*e.g.* **pcmabc**) and inline code is written in italicized typewriter font (*e.g.* `x<-1`).

The package is publicly available on CRAN and can be downloaded from `https://cran.r-project.org/web/packages/pcmabc/`

## 2. The pcmabc R package

### *2.1. The ABC algorithm*

Obtaining the exact likelihood for a phylogenetic comparative sample is possible only in special cases. This is essentially only for normal models [13, 14] or discrete state Markov chains [15, 16]. Hence, for these types of models, a pleiphora of estimation packages are available (*e.g.* [11, 17–23]). However, beyond these types of models, development has yet to take off (however, see [24–28]).

The issue at heart is that the likelihood cannot be obtained exactly. Hence, alternative methods need to be employed, numerically solving an ODE system [29–31] or ABC (*e.g.* [32–34] and see a review in [35]).

The **pcmabc** package implements an ABC algorithm that allows for estimation of parameters of an arbitrary Markov model of trait-dependent speciation (see also Figs. 1 and 2). Let $\Theta$ denote the set of model parameters (known ones and those to be estimated) and $d(\cdot,\cdot)$ the distance between two phylogenetic comparative data sets (tip data and phylogeny). The general structure of the ABC algorithm in the PCM context is also described in [35]. The parameter point estimate is the inverse distance weighted average. We take the inverse distance to take into account how close the simulated sample resembles the original under the given parameter set.

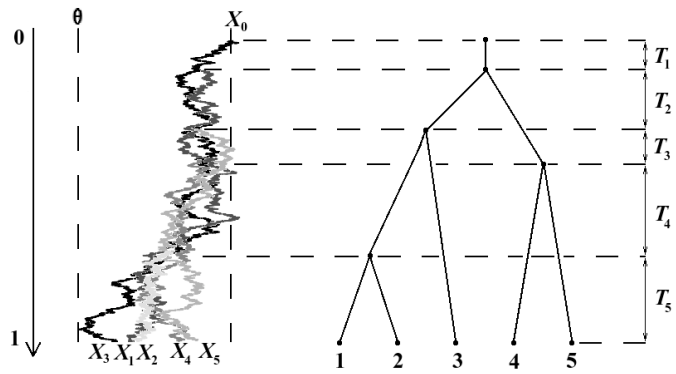


Fig. 1. On the left: a branching OU process simulated on a realization of a tree with $n = 5$ tips using the **TreeSim** [9, 10] and **mvSLOUCH** [11] `R` packages. Parameters used are: $\alpha = 1$, $\sigma = 1$, $X_0 - \theta = 2$, after the tree height was scaled to height 1. On the right: the species tree disregarding the trait values supplied with the notation for the inter-speciation times. Reprinted by permission of the Applied Probability Trust. First published in *J. Appl. Probab.* **52(4)**. Copyright ©Applied Probability Trust 2015.

The ABC inference algorithm is invoked as

```
PCM_ABC( phyltree , phenotypedata , par0 , phenotype.model
    , fbirth , fdeath , X0, step , abcsteps , eps ,
fupdate , typeprintprogress , tree.fixed , dist_method)
```

The first three parameters are "standard" ones, *phyltree* is the phylogeny in **ape**'s [36] *phylo* format, *phenotypedata* is a matrix of trait measurements (rows are the different tips) and *par0* are the initial starting parameters. As described in Section 2.3, the ABC algorithm's distance function treats the phylogeny and trait data separately, hence there is no need for the order of

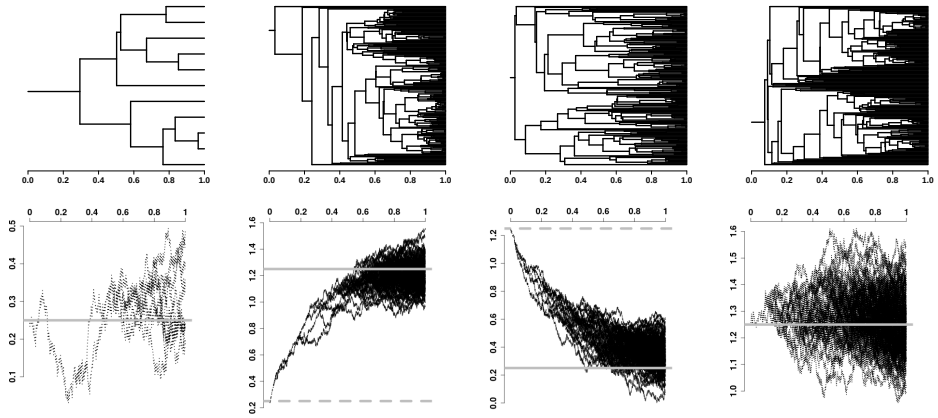

Fig. 2. Examples of phylogenies (top row) and trait trajectories (second row) generated by trait-dependent speciation models. The phenotype follows an OU process, $\mathrm{d}X_\mathrm{t} = -3(X_\mathrm{t} - \psi)\mathrm{d}t + 0.25\mathrm{d}B_\mathrm{t}$ and the birth rate is $10 \times |\sin(X_\mathrm{t})|$. The choice of the birth rate is for illustrative and not biological purposes, *i.e.* we want speciation dynamics to visibly follow trait dynamics. It is important for the reader to remember that the phylogenies and traits are simulated jointly and not that the trait is simulated conditional on the tree, as is the case with most PCM simulations. Columns from left to right: $(X_0 = 0.25, \psi = 0.25)$, $(X_0 = 0.25, \psi = 1.25)$, $(X_0 = 1.25, \psi = 0.25)$, $(X_0 = 1.25, \psi = 1.25)$. The simulations were done by the presented here **pcmabc** package. The trajectories are plotted using the function `pcmabc::draw_phylproc()`, but without the axes, these have to be manually added by the user. Note that the vertical axis of the trait trajectories are not to scale. The height of all trees is 1. The stationary mean (solid) and ancestral (dashed) values are marked with gray lines. If there is only one line, it is because they are equal.

rows to correspond to the tips' order. The initial parameters object, `par0`, is a list of named lists. The first list corresponds to the parameters of the phenotypic evolution process, the second to the speciation dynamics, and the third to the extinction dynamics. Inside each list, the user can indicate which parameters are to be optimized over, which treated as fixed and which are to be positive. If the user would like some further modifications or transformations of the parametrization that is optimized over, they will need to provide their own simulation, birth–death and parameter update functions. The next parameter, `phenotype.model`, is a user-provided function that models the evolution of the phenotype (see Section 2.2), `fbirth` and `fdeath` are user-provided birth and death rate functions (see Section 2.2), `X0` is the root state, `step` is the step size of the simulation (see Section 2.2),

*abcsteps* is the number of parameter draws of the ABC algorithm, *eps* is the acceptance threshold for the parameters (see Alg. 1), *fupdate* is the parameter update function (see Section 2.4), *typeprintprogress* is a parameter that indicates what sort of summary should be printed out by the inference algorithm during its progress. The user is free to provide their own function here. Then, the parameter *tree.fixed* is a logical variable if the phylogeny's branching dynamics depend on the phenotype or not (see Alg. 1), and lastly, *dist_method* is a vector of two entries indicating the distance measure between the trait data and phylogenetic trees, respectively (see Section 2.3).

**Algorithm 1 pcmabc** ABC algorithm, *PCM_ABC()*

**input**: A phylogenetic tree $\mathcal{T}^{(n)}$ with $n$ tips and phenotypic observations for each tip
**output**: Point estimates of parameters with posterior distribution
Set $\Theta_1 := \Theta_{\text{initial}}$, $\mathcal{P} := \emptyset$, $\mathcal{R} = \emptyset$, invtotaldist:= 0, $j = 1$
**for** $i = 1$ to abcsteps **do**
    accept = FALSE
    **if** tree fixed **then**
        Simulate phenotypic data on tree according to Alg. 2 ▷ Section 2.2, *simulate_phenotype_on_tree()*
    **else**
        Simulate phenotypic data and tree according to Alg. 3 ▷ Section 2.2, *simulate_phylproc()*
    **end if**
    $\rho := d$(simulated data,observed data) ▷ Section 2.3
    **if** $\rho < \epsilon$ **then**
        $\mathcal{P}[j] := \Theta_i$, $\mathcal{R}[j] = \rho$
        accept = TRUE
        invtotaldist := invtotaldist$+1/\rho$
        $j++$
    **end if**
    $\Theta_{i+1} :=$ ParameterProposal($\Theta_i$,accept) ▷ Section 2.4
**end for**

$$\hat{\Theta} := \left( \sum_j \mathcal{P}[j]/\mathcal{R}[j] \right) /\text{invtotaldist}$$

**return** $(\hat{\Theta}, \mathcal{P}, \mathcal{R})$

### *2.2. Simulating the phylogeny and trait(s)*

At the core of an ABC method is the ability to simulate data under the model given parameter proposals. In our situation, we have two possibilities to consider. Firstly, the tree is assumed fixed, *i.e.* the trait value does not affect the branching dynamics (Alg. 2) and, secondly, the branching rates depend on the phenotype (Alg. 3).

**Algorithm 2** Trait simulation algorithm, *simulate_phenotype_on_tree()*

**input**: A phylogenetic tree $\mathcal{T}^{(n)}$ with $n$ tips, a Markovian model of trait evolution and trait value at root, $x_0$
**output**: Simulated values of the trait along the whole tree
**procedure** SIMULATE_ON_TREE(tree, simulation_model, $x_0$)
    $X_{\mathcal{T}^{(n)}} := list()$
    **for** daughter branch $i$ of root of tree **do**
        $X_i :=$ simulate trajectory on $i^{\text{th}}$ branch
        $X_{\mathcal{T}^{(n)}}[i] :=$ join($X_i$,SIMULATE_ON_TREE(subtree from $i^{\text{th}}$ daughter node, simulation_model, $X_i$[end]))
    **end for**
    **return** joined entries of $X_{\mathcal{T}^{(n)}}$
**end procedure**
**return** SIMULATE_ON_TREE( $\mathcal{T}^{(n)}$, simulation_model, $x_0$)

The first situation is a standard one and we just mention it for completeness' sake. At the root, one starts at the initial value (user provided). Then, along the branch, of length $t$, one simulates the phenotype according to the provided simulation procedure (conditional on the candidate parameters).

The user provides their own simulation function defined as

```
f_user_trait_simul<-function(time, params, X0, step)
```

where *time* is the duration of the simulation, *params* is a named list of model parameters that are interpreted inside the simulation function, *X0* is the initial trait value and *step* is the simulation's step size. The output of the function is to be the trajectory of the trait starting from *X0* for time *time* with values at points $i \cdot$*step*. The output object is a matrix. In the first row, there are the time instances (starting from 0, in steps of size *step*, correction for the actual time from the root is done later by the package), and in the next rows, the trait values at the time instances. It is important to point out that the trait can be of any dimension.

**Algorithm 3** Trait and tree simulation algorithm, *simulate_phylproc()*

**input**: Tree height, number of tips, a Markovian model of trait evolution and trait value at root, $x_0$
**output**: Tree and simulated values of the trait along the whole tree
**procedure** SIMULATE_ON_TREE(height, simulation_model, $x_0$)
    $X(t) :=$ simulate trait trajectory on lineage with length height
    Calculate the birth rate $\lambda(t)$ as a function of $X(t)$
    Calculate the death rate $\mu(t)$ as a function of $X(t)$
    $\Lambda = \max \lambda(t)$
    Decompose $\lambda(t) = \Lambda p_\lambda(t)$
    Simulate a Poisson process for time height and rate $\Lambda$
    Accept events from the Poisson process with probability $p_\lambda(t)$ ▷ these will be branching events
    $\mathcal{M} = \max \mu(t)$
    Decompose $\mu(t) = \mathcal{M} p_\mu(t)$
    Simulate a Poisson process for time height and rate $\mathcal{M}$
    Accept events from the Poisson process with probability $p_\mu(t)$ ▷ these will be extinction events
    Check if a death event is along the lineage. If so, remove all branching events following it.
    **for** each accepted branching event $i$ **do**
        $\mathcal{T}_i :=$SIMULATE_ON_TREE(height decreased by time of node $i$, simulation_model, trait value at node $i$)
    **end for**
    **return** joined $\mathcal{T}_i$ and $X(t)$
**end procedure**
**return** SIMULATE_ON_TREE(Tree height, simulation_model, $x_0$)

Special support is provided for trait simulation by the **yuima** package [37], through the *simulate_sde_on_branch()* function. The call is

```
simulate_sde_on_branch(branch.length, model.yuima, X0,
    step)
```

where *model.yuima* replaces *params* and defines the stochastic differential equation that should be used to model the trait. The *model.yuima* parameter is the output of the function *yuima::setModel()*. Details how to construct it can be found in *yuima::setModel()*'s manual pages and also in *simulate_sde_on_branch()*'s manual page there is an example how to set it.

After simulating along the branch, a speciation event occurs. Independent evolution is then repeated as above along all the daughter lineages. The starting condition for these is the value at the parental node. More concisely, we describe this in Alg. 2. Algorithm 2 is invoked by calling

```
simulate_phenotype_on_tree(phyltree, fsimulphenotype,
    simul.params, X0, step)
```

*phyltree* is the phylogeny, *X0* the root state, *step* the step size. The function to simulate the phenotype is passed through *fsimulphenotype* and the named list of parameters passed to it through *simul.params*. To simulate the phenotype through the *simulate_sde_on_branch()* function, one sets *fsimulphenotype="sde.yuima"*.

The second situation is more involved. Branching dynamics are trait-dependent so a straightforward simulation, by time steps of some size, would be very computationally heavy. Hence, we employ a variation of the rejection sampling algorithm for the Inhomogeneous Poisson process (Proposition p. 32, [38]). Inside Alg. 3, we did not write how one uses the number of tips. This is a very technical detail in the implementation. The simulation will terminate if it reaches the maximum number of tips (contemporary or also including extinct, the user decides which to provide). However, there is no guarantee that a tree with this number of tips will actually be simulated. If there are death events, then the process may die out before the value is reached. Alternatively, too few birth events get simulated and, again, the desired number is not reached.

The simulation function is more involved, then in the case of simulation on a fixed tree,

```
simulate_phylproc(tree.height, simul.params, X0,
    fbirth, fdeath, fbirth.params, fdeath.params,
    fsimulphenotype, n.contemporary, n.tips.total, step)
```

Some parameters are self-explanatory, *tree.height* is the height of the tree, *X0* the root state, *step* the simulations step size, *n.contemporary* the number of contemporary tips, *n.tips.total* total number of tips, including extinct ones. As before, *fsimulphenotype* is the function to simulate the trait but if it equals *"sde.yuima"*, then *simulate_sde_on_branch()* is invoked. Parameters of trait simulation function are passed as a named list in *simul.params*.

The user additionally provides the birth and death rates (the latter can be passed as *NULL* meaning that it is a pure-birth process) as functions. The birth rate function is provided through the *fbirth* parameter and the death function through the *fdeath* parameter. The parameters of the two rate functions are passed through the named lists *fbirth.params* and

*fdeath.params*. Both can be *NULL*. It is up to the user to make sure that the functions return positive values. Some rate functions are provided inside the package, namely *rate_const* (constant rate, with a possibility to switch to a different value if the first trait variable exceeds some value) and *rate_id* (equals the value of the first trait or linear transformation of it, see manual for options).

The first parameter of the birth and death function has to be the trait vector. The user should remember that the first entry of the trait vector is time (counted from the start of the lineage, *i.e.* from the speciation event where the lineage appeared). Hence, some sort of time heterogeneity is possible. The second parameter is the named list of rate function parameters. Both rate functions should return a single non–negative real number.

It is worth pointing out that the package has also basic graphic capabilities. The function *draw_phylproc()* takes the output of the function *simulate_phylproc()* and draws the trajectory of the trait (as can be seen in Fig. 2). If one wants the tree (in *phylo* format), then one can access it through the field *tree* of *simulate_phylproc()* output. It is good to know that the tree has a couple of extra fields with respect to the usual fields of the *phylo* format tree. In particular, it has the field *tree.height* which stores the height of the tree (time from origin to contemporary tips) and *node.heights* which stores the time from each node to the origin of the tree.

### *2.3. The summary statistics and distance measure*

A key element of an ABC algorithm is the choice of the summary statistic for the simulated/observed sample and the distance function between the observed and simulated statistics. In our situation, the sample consists of two components — the phylogeny and observed trait values.

The **pcmabc** package offers various possibilities in this respect. We describe the ones that through numerical experiments seemed to work best. It turned out that looking at the sample mean and variance of the trait values was the best option. Statistics based on tips' means and variances have already been considered in the PCM setting [32–34]. However, the situation is different in the previous two cases.

In [34], terminal lineages corresponding to higher taxonomic levels is considered. Each such tip contains a number of species for which phylogenetic relationships might not be resolved. Then, the differences between the means and variances of the trait measurements from species inside each higher order tip are taken. Earlier, in a similar spirit, just the variances inside clades were compared [32]. Similarly, in [33], the mean and variance of the difference between tip measurements are calculated.

Our situation is different as we do not consider a fixed tree but a random one, so we will not have a correspondence between the tips. On the other hand, even with a fixed tree, there are multiple possible symmetries, from the perspective of the trait process, in the tip labellings. Take, for example, the simplest case of a cherry (*i.e.* two tips stemming from a single ancestral node). Then, as trait evolution is independent following speciation, one cannot distinguish between the two tips, unless there is some additional information, like branch specific parameters. Without such, taking the difference between tips by the original labels might not be the optimal choice.

The distance between trees is actually more involved. It seems that the weighted and normalized Robinson–Foulds metric [39, 40] implemented in the **phangorn** [41] package as `wRF.dist()` with `normalize=TRUE` seemed to work the most effectively in our experiments. These are the default distance functions. However, it should be pointed out that the simulation-based tests were performed using OU models of trait evolution. These are normal processes and hence all information is stored in the mean and variance. Should the trait evolve as a non-normal process other distance measures could be more appropriate.

### *2.4. Proposing parameters*

The standard ABC algorithm draws parameter proposals from the prior distribution (see the ABC steps description in [35]). In our situation, due to the complexity of the sample, this would have resulted in nearly all parameter proposals being rejected. Hence, we employed a hybrid proposal algorithm that attempts to explore in detail the parameter space around "good" proposals.

If the previous parameter set was rejected, the inference algorithm samples each parameter uniformly from the interval $[-10, 10]$. Such a restriction was chosen not to have extreme parameter values. If this restriction is problematic, a user may provide their own proposal function as described below. If the previous parameter set was accepted, then each new parameter is the previous parameter modified by a mean-zero normal deviation (with user specified standard deviation). As a result, we cannot say that we obtain a sample from the posterior distribution as a usual ABC algorithm would result in. Nevertheless, this method seems to result in decent parameter estimates as presented in Section 3. The user is allowed to specify which parameters are to be positive (transformation by taking the exponential).

However, the user is also free to specify their own parameter update function. The function has to handle the call

```
f_user_update(par, par0, baccept, ABCdist, phenotypedata,
    phyltree)
```

where *par* is the list (as described in Section 2.1) of parameters from the previous step, *par0* are the initial parameters provided to *PCM_ABC()* (see Section 2.1), *baccept* is a logical variable indicating if the *par* parameters were accepted (*TRUE*) or not (*FALSE*), *ABCdist* is the distance between the observed and simulated (under *par*) data, *phenotypedata* is the original data matrix of trait measurements and *phyltree* is the original phylogenetic tree.

In particular, this means that it is possible to change the inference to a usual ABC algorithm with independent proposals from the prior. The user-defined function just samples from the prior ignoring all information on the previously considered parameter set.

## 3. Proof of concept simulation study

We performed a simple simulation study to evaluate whether the ABC inference package can capture any signal on the trait-dependent speciation process, based only on the contemporary sample and phylogeny. This is not a trivial question, as it is not clear what exactly is estimable in such a setup. The trait and branching dynamics interact with each other with many potential masking effects (see Fig. 3 for example plots on how to present the tree's dynamics). In the PCM context, such masking effects occur in even simpler setups (*e.g.* [42]). Hence, we aim at a proof of concept study to evaluate the potential usefulness of the inference algorithm.

We simulate a univariate trait that follows an OU process, defined as

$$\mathrm{d}X_t = -\alpha(X_t - \psi)\mathrm{d}t + \sigma \mathrm{d}B_t,$$

```
simulate_OU_sde<-function(time,params,X0,step){
    A <- c(paste("(-",params$a11,")*(x1-(",params$psi1
        ,"))",sep=""))
    S <- matrix( params$s11, 1,1)
    yuima.1d <- yuima::setModel(drift = A, diffusion =
        S, state.variable=c("x1"),solve.variable=c("x1
        ") )
    simulate_sde_on_branch(time,yuima.1d,X0,step)
}
```

with parameters

```
true_sde.params<-list(a11=1,s11=1,psi1=0, positivevars
    =c(TRUE,TRUE,FALSE), abcstepsd=rep(0.5,3))
```

The reader can also see how one indicates positive parameters *positivevars* and the standard deviation for the Gaussian update of proposed parameters *abcstepsd*. We assume a pure birth process, with speciation rate function

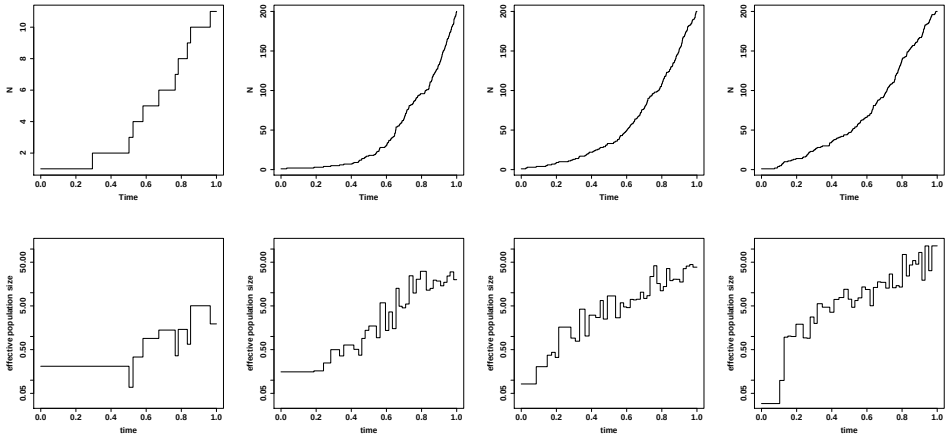


Fig. 3. Lineage through time and skyline plots corresponding to the phylogenetic trees of Fig. 2. In the first row, we provide the lineage through time plots, **ape***::ltt.plot()*, and in the second row, the skyline plots, **ape***::skyline()* with *epsilon=-log(0.95)/3*. The choice of *epsilon* is motivated by how quickly ancestral effects are lost. The parameter *epsilon* controls the temporal structure in the population data [1]. From our perspective, we can think of it as controlling how long the coalescent rate will be approximately constant. The expectation of the OU process after time $t$ equals $e^{-3t}X_0 + (1 - e^{-3t})\theta$. Hence, we may ask how much time is needed to pass for the expectation of the trait to lose a "significant" amount of the ancestral value. If by a "significant" amount of loss we take losing 5% of the ancestral value, then we obtain the time to lose this as $-\log(0.95)/3$ (*cf.* with phylogenetic half-life [12]).

```
fbirth_rate_constrained<-function(x,params,...){
    x<-x[2]
    params$scale/(1+exp(-x))
}
```

with parameters

```
true_birth.params<-list(scale=5,abcstepsd=0.5,
  positivevars=c(TRUE,TRUE),fixed=c(FALSE))
```

We chose the OU model for the phenotype because it is the current gold standard in PCMs for modelling adaptive evolution. Its dynamics are very well-understood — after an initial "burn-in period", the trait will stabilize around its stationary distribution. Furthermore, for a constant rate birth–death, the process "memory effects" have been intensively studied (*e.g.* [43–45]). Hence, in our setup, we should expect the birth rate to oscillate in a controlled manner, after each lineage reaches its stationary distribution.

However, as the *scale=5* parameter is significantly greater than $\alpha = 1$, we are in the fast branching (or slow adaptation) regime. This induces strong correlations (through remembered ancestral effects) between evolving lineages, making estimation more difficult and in a way introducing less straightforward process dynamics.

The speciation rate takes values between (0,*scale*). With negative trait values, it decreases to 0, with positive, increases up to *scale*. As the trait follows an OU process with stationary mean 0, it will oscillate around 0. Hence, the birth rate should oscillate around *scale*/2. The main question of interest is if the inference procedure will be able to identify the value of *scale* and also of the OU process's parameters. The reader can also see how one indicates parameters which are not to be optimized over. The logical values in the field *fixed* tell this to the inference procedure. If it were *TRUE*, then *scale* would never be changed from its initial value.

We follow a simple procedure of simulating data under the model and then calling *PCM_ABC()* to recover parameters given the simulated phylogeny and contemporary trait measurements. We take *abcsteps=1000*, number of tips of the phylogeny 200, simulation step size 0.001, the distance between phylogenies is taken as the Robinson–Foulds metric and the distance between the traits compares the sample mean and variances. We take *eps=0.2* as the parameter acceptance cut-off when comparing the summary statistics. Furthermore, we used the default setting that the inference algorithm samples each new parameter proposal after rejection, uniformly from the interval $[-10, 10]$. This however is on the scale used in estimation, so if a parameter is assumed positive ($\alpha$, $\sigma$, *scale*), then $[-10, 10]$ are bounds on the log scale. On the real scale, these translate to $[e^{-10}, e^{10}]$. Furthermore, we bounded the *scale* parameter (on its actual scale) by 10, *i.e.* *scale*$\in [e^{-10}, 10]$.

For the fixed tree simulation part, we did not use **pcmabc**'s functionality. This was to avoid any potential bias and see how **pcmabc** can recover parameters from a completely independent of its code simulation. We simulated the phylogeny using the **TreeSim** package

```
phyltree<-TreeSim::sim.bd.taxa(numbsim=1,n=200,lambda
    =1,mu=0)[[1]]
```

and then the OU-evolving traits using the **mvSLOUCH** package

```
OUOUparameters<-list(vY0=matrix(0,1,1),A=matrix(1,1,1)
    ,mPsi=matrix(0,1,1),Syy=matrix(1,1,1))
OUOUdata<-mvSLOUCH::simulOUCHProcPhylTree(phyltree,
    OUOUparameters)
```

We perform 190 repeats of the simulate and recover parameters procedure. We recover parameters for the situation where the tree is assumed fixed (here only OU parameters can be recovered) and where the tree is assumed to be non-fixed (we can also recover `scale`). We present the summary of the results of the simulation study in Table I, and in Figs. 4, 5 the histograms of the estimated values.

TABLE I

Mean and variance of parameter estimates for 190 repeats of the simulate and re-estimate procedure. It is important to notice that for the fixed tree case, 36 of the repeats resulted in an error (and no estimate), while for the non-fixed tree case, 53. The sample's standard deviation is abbreviated as sd. We also present estimates of the composite parameter $v_y = \sigma^2/(2\alpha)$, the stationary variance of the OU process.

| | | Fixed tree | | Non-fixed tree | |
|---|---|---|---|---|---|
| Parameter | True value | Mean | sd | Mean | sd |
| $\alpha$ | 1 | 2.354 | 1.999 | 1.9738 | 1.585 |
| $\sigma$ | 1 | 1.33 | 0.627 | 1.473 | 0.887 |
| $v_y$ | 0.5 | 0.523 | 0.269 | 0.831 | 1.085 |
| $\psi$ | 0 | 0.010 | 0.313 | −0.803 | 4.261 |
| `scale` | 5 | — | — | 6.331 | 5.374 |

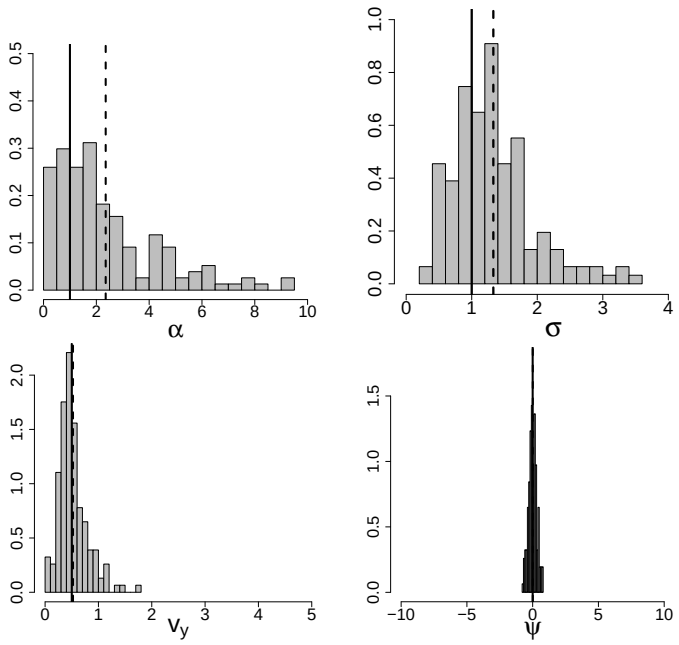


Fig. 4. Histograms of the parameter estimates in the fixed tree case, top left: $\alpha$, top right: $\sigma$, bottom left: $\psi$, bottom right: $v_y$. The solid gray line is the true value, the dashed is the mean of the estimates.

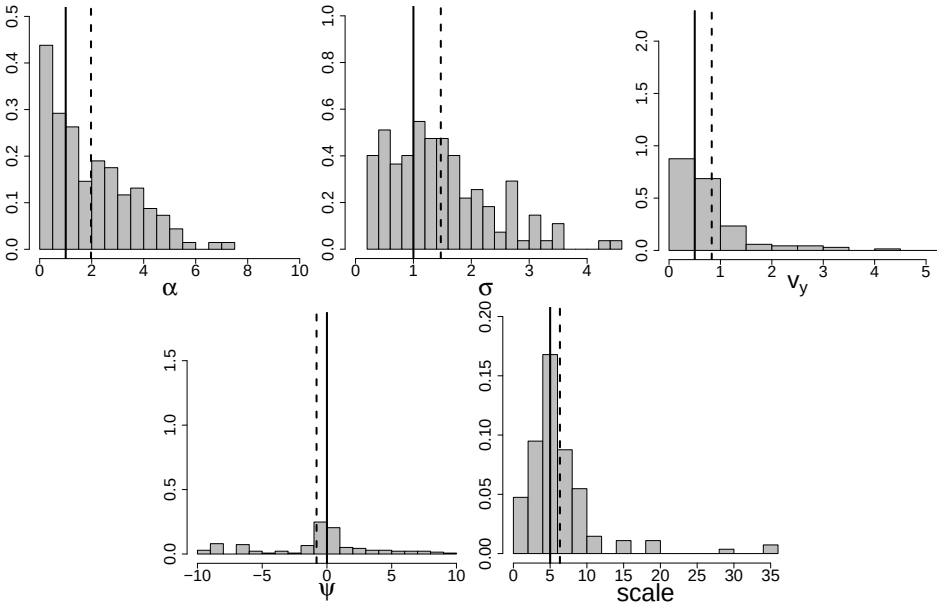


Fig. 5. Histograms of the parameter estimates in the fixed tree case, top left: $\alpha$, top center: $\sigma$, top right: $v_y$, bottom left: $\psi$, bottom right: `scale`. The solid gray line is the true value, the dashed is the mean of the estimates.

All simulations were done in `R` version 3.4.2 running on an openSUSE 42.3 (x86_64) box. A major handicap of the study were the long running times. About two weeks were required to obtain 20 repeats of the simulation-estimation procedure (with 1000 parameter proposals).

The results of the simulation-estimation are, on the one hand, not surprising. The parameters $\alpha$, $\sigma$ are not easy to estimate, as was indicated in [46]. The parameters $\psi$, $v_y$ (OU's stationary mean and variance) are much easier to estimate, as they should be close to the sample average and variance respectively (when the tree is fixed, *e.g.* [45]).

On the other hand, what is optimistic in the fixed tree case, is that these parameter estimates are close to the true ones. Previous studies were performed for likelihood-based inference methods. Our ABC approach does not use the likelihood and, hence, is at a serious disadvantage. Furthermore, only 1000 ABC steps are done (including the rejected proposals), due to running times. Despite this, the estimated parameter values are in a reasonable range.

When one looks at the more interesting situation, where the tree is not taken to be fixed, one can also be optimistic. The key parameter, `scale` seems estimated not too far away from its true value. Furthermore, a slight

improvement in the value of $\alpha$ is visible. The marked deterioration of the $\psi$ parameter is surprising and warrants a more in-depth analysis. One possible direction of further development is to develop (as the interface allows this) hybrid ways of parameter proposals. For example, for $\psi$ and $v_y$, the sample mean and variance values could be used more directly.

## 4. pcmabc: an R tutorial

One first needs to install the package from *e.g.* by

```
install.packages("pcmabc")
```

and begins working with it by loading it

```
library(pcmabc)
```

Then, one needs to read-in the trait observations and tree. As this is project specific, we will rather simulate it using **pcmabc**'s capabilities. We first define the trait simulation function (for the same OU process as we considered in Section 3)

```
simulate_OU_sde<-function(time,params,X0,step){
    A <- c(paste("(-",params$a11,")*(x1-(",params$psi1
        ,"))",sep=""))
    S <- matrix(params$s11, 1,1)
    yuima.1d <- yuima::setModel(drift = A, diffusion =
        S, state.variable=c("x1"),solve.variable=c("x1
        ") )
    simulate_sde_on_branch(time,yuima.1d,X0,step)
}
```

and then the birth rate function (again the same as in Section 3)

```
fbirth_rate_constrained<-function(x,params,...){
    x<-x[2]
    params$scale/(1+exp(-x))
}
```

It is important to have the `...` (*i.e.* the three dots following ***params*** in the above code snippet) in the function's interface. This is because when called in the package, other parameters can be passed to it for generality, even though the user's implementation will not require them. Having defined the functions, we define the parameters under which we want to simulate

```
true_sde.params<-list(a11=1,s11=1,psi1=0)
true_birth.params<-list(scale=5)
numtips<-200
tree_height<-max(15,log(numtips))
step<-0.001
```

If we had assumed a non-zero extinction rate, then its definition and parameters would be handled in exactly the same way. With all of this, we can simulate our trait-dependent speciation process

```
simres<-simulate_phylproc(tree_height,simul.params=
    true_sde.params,X0=X0,fbirth=fbirth_rate_
    constrained,fdeath=NULL,fbirth.params=true_birth.
    params,fdeath.params=NULL,fsimulphenotype=simulate_
    OU_sde,n.contemporary=numtips,n.tips.total=100*
    numtips,step=step)
```

It is worth commenting on three function parameters here, *tree_height*, *n.contemporary* and *num.tips.total*. The package first simulates the backbone lineage of length equal to *tree_height*. Then, it will start to simulate lineages coming out of the backbone lineage. The simulation needs a stopping condition, it will either be that all birth events have taken place or *n.contemporary* tips (tips at height *tree_height*) are generated, or *num.tips.total* are generated. The latter is for the situation when extinction is present. The value of *num.tips.total* will be the total number of tips, contemporary and extinct. Here, it is just given for illustrative purposes. The value of *step* is the simulation step size. After simulation, if one wants to plot the trait trajectory over the tree, one may use

```
draw_phylproc(simres)
```

The figure is a bare drawing of the trait's evolution on the tree. Any plot components like axes have to be added manually by the user. The tree can also be plotted, using *e.g.* **ape**'s plotting capabilities

```
plot(simres$tree,show.tip.label=FALSE,root.edge=TRUE)
```

In order to estimate parameters, we need a phylogenetic tree and a matrix with tip measurements. The phylogeny has to be in the *phylo* format. We can recover them from the simulated object as

```
phyltree<-simres$tree
phenotypedata<-get_phylogenetic_sample(simres)
```

The package provides the functionality to recover the tip measurements using an inbuilt function *get_phylogenetic_sample()*. We are now ready to perform the ABC inference, using some random starting parameters and choosing the distance calculation methods

```
sde.params<-list(a11=5*runif(1),s11=5*runif(1),psi1=0,
    positivevars=c(TRUE,TRUE,FALSE),abcstepsd=rep
    (0.1,3))
birth.params<-list(scale=4+5*runif(1),maxval=10,
    abcstepsd=0.5,positivevars=c(TRUE,TRUE),fixed=c(
    FALSE,TRUE))
par0<-list(phenotype.model.params=sde.params,birth.
    params=birth.params)
X0<-c(0)
tree_dist<-"wRFnorm.dist"
data_dist<-"variancemean"
eps<-0.2
abcsteps<-1000
step<-0.001
ABCres<-PCM_ABC(phyltree=phyltree,phenotypedata=
    phenotypedata,par0=par0,phenotype.model=simulate_OU
    _sde,fbirth=fbirth,fdeath=NULL,X0=X0,step=step,
    abcsteps=abcsteps,eps=eps,tree.fixed=FALSE,dist_
    method=c(data_dist,tree_dist))}
```

If we wanted to assume a fixed tree (*i.e.* the phenotype does not affect speciation), we would set *tree.fixed=FALSE* and *e.g.* *tree.dist<-NA*.

Afterwards (for the above setup, it takes a bit under one day in R 3.4.2 for openSUSE 42.3 (x86_64) on a 3.50GHz processor), we need to extract the estimated parameters. The field *ABCres$param.estimate* is a list with two lists (one if *tree.fixed=FALSE* or three if extinction present). The first list *phenotype.model.params* contains the point estimates of the trait evolution's process's parameters and the second list *birth.params* the point estimates of the speciation rate's parameters. If extinction is present, a third list *death.params* with the point estimates of the extinction rate's parameters will be present. The point estimates are calculated as described in Alg. 1 as a weighted, by the inverse distances, average of the accepted parameter values. The field *ABCres$all.accepted* contains all the accepted parameters, including the distance from the observed data, field *distance.from.data* for each accepted parameter set and also the inverse of the distance, field *inv.distance.from.data*. In the output object of *PCM_ABC*, there are also two further fields: *sum.dists.from.data* the sum of the distances from the observed data for all accepted parameters and *sum.inv.dists.from.data* the sum of the inverses of these distances.

## 5. Conclusions

### *5.1. The possibilities of the software*

The **pcmabc** package is designed for maximum flexibility from the user's perspective. It will be easiest to provide the full call of the function and discuss the more involved components. This we did in the short tutorial in Section 4.

One could say that some of the parameters are extremely technical and maybe should be hidden from the user. However, as a lot concerning the probabilistic/statistical properties of these models is not clear at this stage, the user should have the possibility to experiment. Such experimentation should lead to better understanding of the underlying properties of the system. In turn, this will allow us to know what is the best choice of these parameters (and to make them the default ones).

The package is extremely flexible and hence should be attractive for various types of studies. Coupled with a model selection procedure, it will allow for comparing different models of evolution and hence, asking questions about the system under study. The basic question one will want to ask is whether the speciation dynamics depend on the trait under study or not. For this, one can try a constant birth–death rate function, time-dependent (but trait-independent) speciation dynamics and trait-dependent speciation dynamics. Treating time as a "trait", one may study if the speciation dynamics increase, decrease with time or maybe exhibit some sort of periodic behaviour. Similarly with trait-dependent dynamics — are the speciation rates monotonic w.r.t. the trait, is there a carrying capacity, periodicity or maybe some more complicated dynamics.

### *5.2. Some limitations*

The one main feature that is lacking in the package at the moment is the possibility to implement punctuated equilibrium models (*e.g.* [47, 48]), *i.e.* jumps at speciation points, or more generally, including direct feedback from the speciation dynamics into the trait dynamics. This is as for computational efficiency, we first simulate the whole lineage and only afterwards, through the rejection sampling algorithm, simulate the points of speciation on this lineage. Hence, if the speciation event is meant to have an effect on the already simulated lineage (*e.g.* through a jump), the already simulated trait data following the speciation point becomes invalid. And this, in turn, could invalidate the thinning from the rejection sampling algorithm and hence, the simulated speciation event. Therefore, to include such models, a different simulation algorithm has to be developed.

Having written the above caveat, our approach is still extremely general and it is important to point out that it allows for another, biologically very relevant, type of speciation driven evolution. Namely, the simulation algorithm has an inbuilt concept of a "spine" (or in other words main) lineage. Speciation driven dynamics, like jumps at speciation points, can take place at the start of new lineages. This is consistent with the idea that a new lineage (species) broke off because of some dramatic event, sudden jump in the trait.

Even though this is not directly evident from the user interface, the package can easily handle time-heterogeneous models. What one passes to the simulation functions is the trait value at the start of the branch or the trait value at the potential branching event time. Then the simulation procedure evolves taking (from the package's perspective) 0 as the starting time of the branch. However, one can have time (from the root) as one of the elements of the trait vector and then this can be used appropriately in the definition of the transition simulation procedures and birth–death rate functions. Therefore, one can immediately recognize that one can include other dummy "control" traits, *e.g.* environments, (geological) epochs, *etc.* All that needs to be remembered is that all user-defined functions have to appropriately treat these dimensions. Furthermore, the package allows for defining fixed (*i.e.* not estimated) trait and speciation dynamics parameters, providing immense flexibility.

### *5.3. Directions of development*

Despite the generality of the package, there is a lot of space for further development both in theoretical and implementation directions. A more involved and detailed study is required to know what are the optimal inference settings, what distance measures should be used. For specific models, one should ask what parameters are estimable.

From the perspective of the implementation, speciation-dependent trait evolution is missing. For effectiveness' sake, full lineages are simulated and only then are speciation events marked on them. This implies that on the "main" lineage-dependent speciation-dependent trait evolution cannot take place. Hence, new simulation algorithms have to be developed that will not discard everything after the first time the branching influences the phenotype.

We are grateful to the anonymous reviewer for their comments which have significantly improved the manuscript. K.B.'s research is supported by the Swedish Research Council's (Vetenskapsrådet) grant No. 2017–04951.

## REFERENCES


[1] K. Strimmer, O.G. Pybus, *Mol. Biol. Evol.* **18**, 2298 (2001).

[2] A. Sottoriva *et al.*, *Proc. Natl. Acad. Sci. USA* **110**, 4009 (2013).

[3] H. Xiao, K. Bartoszek, P. Liò, *BMC Bioinformatics* **19**, 439 (2018).

[4] P.J. Diggle, R.J. Gratton, *J. R. Stat. Soc. B* **46**, 193 (1984).

[5] D.B. Rubin, *Ann. Stat.* **12**, 1151 (1984).

[6] J. Lintusaari *et al.*, *Syst. Biol.* **66**, e66 (2017).

[7] S. Tavaré, D.J. Balding, R.C. Griffiths, P. Donnelly, *Genetics* **145**, 505 (1997).

[8] R Core Team, *R: A Language and Environment for Statistical Computing*, The R Foundation for Statistical Computing, Vienna, Austria, 2013.

[9] T. Stadler, *J. Theor. Biol.* **261**, 58 (2009).

[10] T. Stadler, *Syst. Biol* **60**, 676 (2011).

[11] K. Bartoszek *et al.*, *J. Theor. Biol.* **314**, 204 (2012).

[12] T.F. Hansen, *Evolution* **51**, 1341 (1997).

[13] V. Mitov, K. Bartoszek, G. Asimomitis, T. Stadler, `arXiv:1809.09014 [q-bio.PE]`.

[14] H. Morlon *et al.*, *Meth. Ecol. Evol.* **7**, 589 (2016).

[15] J. Felsenstein, *Syst. Biol.* **22**, 240 (1973).

[16] J. Felsenstein, *J. Mol. Evol.* **17**, 368 (1981).

[17] M.A. Butler, A.A. King, *Am. Nat.* **164**, 683 (2004).

[18] J. Clavel, G. Escarguel, G. Merceron, *Meth. Ecol. Evol.* **6**, 1311 (2015).

[19] R.G. FitzJohn, *Meth. Ecol. Evol.* **3**, 1084 (2012).

[20] E.W. Goolsby, J. Bruggeman, C. Ané, *Meth. Ecol. Evol.* **8**, 22 (2017).

[21] T.F. Hansen, J. Pienaar, S.H. Orzack, *Evolution* **62**, 1965 (2008).

[22] L.J. Harmon *et al.*, *Bioinformatics* **24**, 129 (2008).

[23] M. Manceau, A. Lambert, H. Morlon, *Syst. Biol.* **66**, 551 (2017).

[24] K. Bartoszek, in: Proceedings of the Eighteenth National Conference on Applications of Mathematics in Biology and Medicine, Krynica Morska 2012, p. 25.

[25] S.P. Blomberg, `bioRxiv`, DOI:10.1101/067363.

[26] F.C. Boucher *et al.*, *Syst. Biol.* **67**, 304 (2018).

[27] P. Duchen *et al.*, *Syst. Biol.* **66**, 950 (2017).

[28] M.J. Landis, J.G. Schraiber, M. Liang, *Syst. Biol.* **62**, 193 (2013).

[29] R.G. FitzJohn, *Syst. Biol.* **59**, 619 (2010).

[30] E.E. Goldberg, L.T. Lancaster, R.H. Ree, *Syst. Biol.* **60**, 451 (2011).

[31] W.P. Maddison, P.E. Midford, S.P. Otto, *Syst. Biol.* **56**, 701 (2007).

[32] F. Bokma, *Syst. Biol.* **59**, 602 (2010).

[33] D.-C. Jhwueng, `arXiv:1808.05878 [stat.ME]`.

[34] G.J. Slater *et al.*, *Evolution* **66**, 752 (2012).

[35] N. Kutsukake, H. Innan, in: *Modern Phylogenetic Comparative Methods and Their Application in Evolutionary Biology*, L.Z. Garamszegi (Ed.), Springer, Berlin–Heidelberg 2014, p. 409.

[36] E. Paradis, K. Schliep, *Bioinformatics*, DOI:10.1093/bioinformatics/bty633.

[37] A. Brouste *et al.*, *J. Stat. Soft.* **57**, 1 (2014).

[38] S.M. Ross, *Simulation*, Elsevier AP, 2006.

[39] D.F. Robinson, L.R. Foulds, in: *Combinatorial Mathematics VI*, Lect. Notes Math. vol. 748, A.F. Horadam, W.D. Wallis (Eds.), Springer, Berlin–Heidelberg 1979, p. 119.

[40] D.F. Robinson, L.R. Foulds, *Math. Biosci.* **53**, 131 (1981).

[41] K.P. Schliep, *Bioinformatics* **27**, 592 (2011).

[42] K. Bartoszek, S. Glémin, I. Kaj, M. Lascoux, *J. Theor. Biol.* **429**, 35 (2017).

[43] R. Adamczak, P. Miłoś, *Elect. J. Probab.* **20**, 1 (2015) [`arXiv:1111.4559 [math.PR]`].

[44] C. Ané, L.S.T. Ho, S. Roch, *J. Math. Biol.* **74**, 355 (2017).

[45] K. Bartoszek, S. Sagitov, *J. App. Prob.* **52**, 1115 (2015).

[46] C.E. Cressler, M.A. Butler, A.A. King, *Syst. Biol.* **64**, 953 (2015).

[47] K. Bartoszek, *Math. Biosci.* **254**, 42 (2014).

[48] F. Bokma, *J. Evol. Biol.* **15**, 1048 (2002).