Fast Likelihood Calculation for Multivariate Gaussian Phylogenetic Models with Shifts

Supplementary Material

Venelin Mitov[a,b,*], Krzysztof Bartoszek[c], Georgios Asimomitis[a], Tanja Stadler[a,b]

[a]*Department of Biosystems Science and Engineering, Eidgenössische Technische Hochschule Zürich, Basel, Switzerland*
[b]*Swiss Institute of Bioinformatics, Lausanne, Switzerland*
[c]*Department of Computer and Information Science, Linköping University, Linköping, Sweden*

[*]Corresponding author
Email addresses: `vmitov@gmail.com` (Venelin Mitov), `krzysztof.bartoszek@liu.se`,`krzbar@protonmail.ch` (Krzysztof Bartoszek), `georgios.asimomitis@gmail.com` (Georgios Asimomitis), `tanja.stadler@bsse.ethz.ch` (Tanja Stadler)

Contents

Appendix A. The PCMBase R package

The **PCMBase** package takes advantage of the fact that the quadratic polynomial representation of the likelihood function is valid for all models in the $\mathcal{G}_{LInv}$ family. Hence, once the analytical integration over the internal nodes has been implemented, the addition of a new $\mathcal{G}_{LInv}$ model to the framework boils down to defining the transition density in terms of the functions $\vec{\omega}$, $\mathbf{\Phi}$ and $\mathbf{V}$ (Def. 1). **PCMBase** implements this idea, based on the concept of inheritance between programming modules: Eqs. (2), (9), (10), (11) are implemented in a base module called "GaussianPCM", which is abstract with respect to $\vec{\omega}$, $\mathbf{\Phi}$ and $\mathbf{V}$ (Fig. S1). These functions are provided in inheriting modules definable for each $\mathcal{G}_{LInv}$ model. This hierarchical design is presented in Fig. S1.

*Appendix A.1. Extending **PCMBase***

Extending the **PCMBase** functionality can be achieved in two ways:

1. **Adding a new model**. It is possible to write a new module inheriting from the module "GaussianPCM" and implementing its own version of the functions $\vec{\omega}$, $\mathbf{\Phi}$ and $\mathbf{V}$;

2. **Adding a parameterization**. It is possible to restrict or apply a transformation to some of the parameters of an already defined model (Fig. S1).

Appendix A.2. Using the package

Figure S2 shows the runtime objects and use-cases currently implemented in the **PCMBase** package. Once the modules for the models of interest have been implemented, the **PCMBase** package can be used to:

- Creating a model object. The end-user function for creating a model object is `PCM()`. A model object represents an S3 object, that is, a named list with members corresponding to the model parameters, such as `H`, `Sigma_x` and `Sigmae_x`, and a class attribute equalling the model type, e.g. `BM` or `OU`.

- Simulating the evolution of a set of continuous traits along a tree, according to a model. The user level function for trait simulation is `PCMSim()`. Based on the S3 class of its model argument `PCMSim()` invokes an appropriate specification of the S3 generic function `PCMCond()`, which creates a random sampler from the trait distribution at the end of a branch, given the model, the branch length and the trait values at the beginning of the branch. Starting from a user specified trait value at the root (parameter `X0`), the simulation proceeds as a pre-order traversal of the tree, i.e. first the daughter nodes of the root node are assigned random trait values drawn from their corresponding conditional distributions, then their daughters are assigned and so on, until all tip nodes in the tree have been assigned a random trait value.

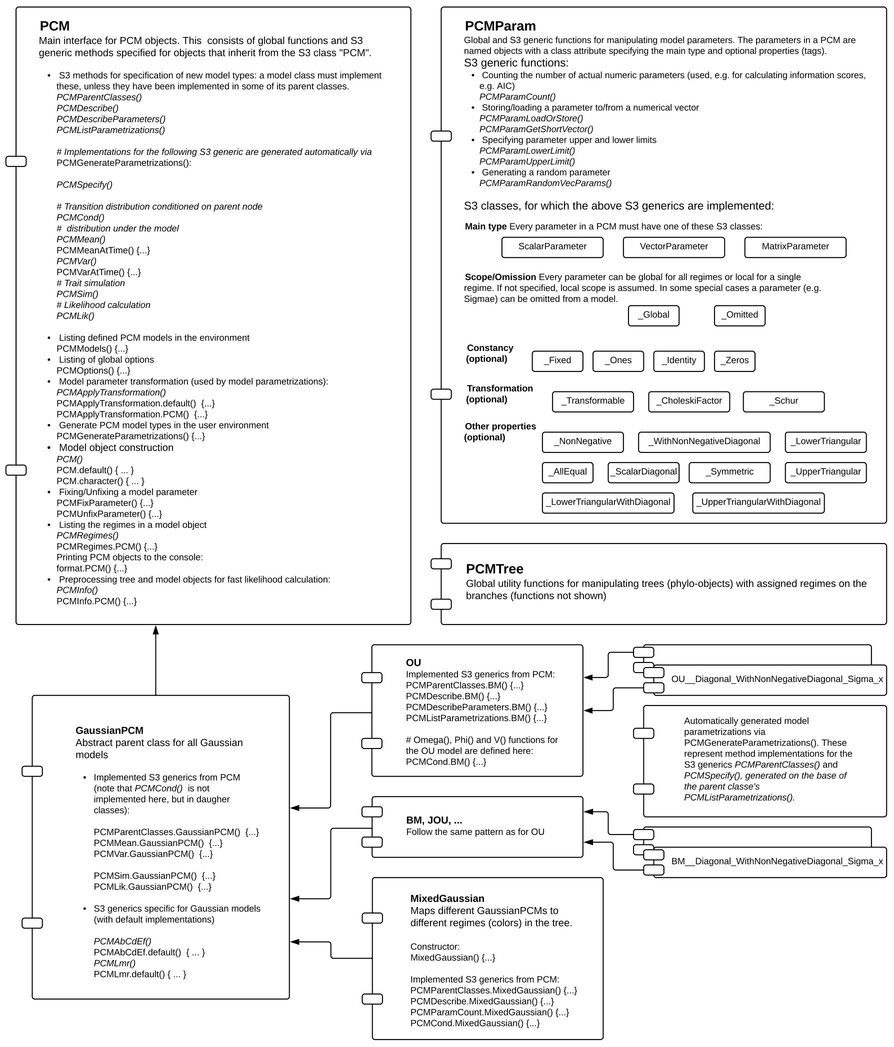
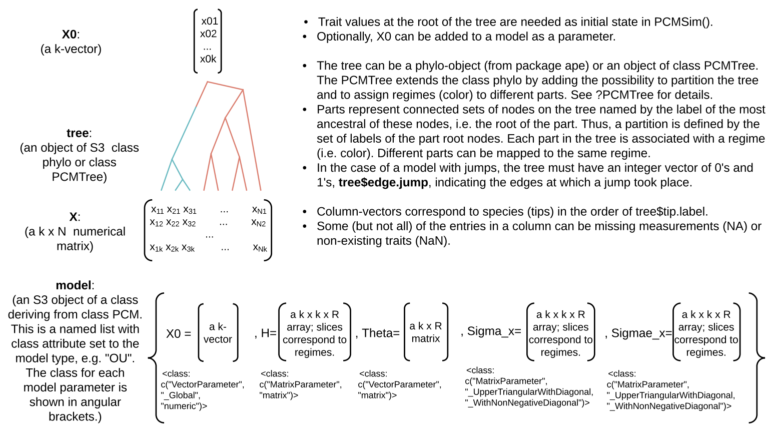

PCM
Main interface for PCM objects. This consists of global functions and S3 generic methods specified for objects that inherit from the S3 class "PCM".
S3 methods for specification of new model types: a model class must implement these, unless they have been implemented in some of its parent classes.
PCMParentClasses()
PCMDescribe()
PCMDescribeParameters()
PCMListParametrizations()
# Implementations for the following S3 generic are generated automatically via PCMGenerateParametrizations():
PCMSpecify()
# Transition distribution conditioned on parent node
PCMCond()
# distribution under the model
PCMMean()
PCMMeanAtTime() {...}
PCMVar()
PCMVarAtTime() {...}
# Trait simulation
PCMSim()
# Likelihood calculation
PCMLik()
Listing defined PCM models in the environment
PCMModels() {...}
Listing of global options
PCMOptions() {...}
Model parameter transformation (used by model parametrizations):
PCMApplyTransformation()
PCMApplyTransformation.default() {...}
PCMApplyTransformation.PCM() {...}
Generate PCM model types in the user environment
PCMGenerateParametrizations() {...}
Model object construction
PCM()
PCM.default() { ... }
PCM.character() { ... }
Fixing/Unfixing a model parameter
PCMFixParameter() {...}
PCMUnfixParameter() {...}
Listing the regimes in a model object
PCMRegimes()
PCMRegimes.PCM() {...}
Printing PCM objects to the console:
format.PCM() {...}
Preprocessing tree and model objects for fast likelihood calculation:
PCMInfo()
PCMInfo.PCM() {...}
PCMParam
Global and S3 generic functions for manipulating model parameters. The parameters in a PCM are named objects with a class attribute specifying the main type and optional properties (tags).
S3 generic functions:
Counting the number of actual numeric parameters (used, e.g. for calculating information scores, e.g. AIC)
PCMParamCount()
Storing/loading a parameter to/from a numerical vector
PCMParamLoadOrStore()
PCMParamGetShortVector()
Specifying parameter upper and lower limits
PCMParamLowerLimit()
PCMParamUpperLimit()
Generating a random parameter
PCMParamRandomVecParams()
S3 classes, for which the above S3 generics are implemented:
Main type Every parameter in a PCM must have one of these S3 classes:
ScalarParameter
VectorParameter
MatrixParameter
Scope/Omission Every parameter can be global for all regimes or local for a single regime. If not specified, local scope is assumed. In some special cases a parameter (e.g. Sigmae) can be omitted from a model.
_Global
_Omitted
Constancy (optional)
_Fixed
_Ones
_Identity
_Zeros
Transformation (optional)
_Transformable
_CholeskiFactor
_Schur
Other properties (optional)
_NonNegative
_WithNonNegativeDiagonal
_LowerTriangular
_AllEqual
_ScalarDiagonal
_Symmetric
_UpperTriangular
_LowerTriangularWithDiagonal
_UpperTriangularWithDiagonal
PCMTree
Global utility functions for manipulating trees (phylo-objects) with assigned regimes on the branches (functions not shown)
OU
Implemented S3 generics from PCM:
PCMParentClasses.BM() {...}
PCMDescribe.BM() {...}
PCMDescribeParameters.BM() {...}
PCMListParametrizations.BM() {...}
# Omega(), Phi() and V() functions for the OU model are defined here:
PCMCond.BM() {...}
OU__Diagonal_WithNonNegativeDiagonal_Sigma_x
Automatically generated model parametrizations via PCMGenerateParametrizations(). These represent method implementations for the S3 generics PCMParentClasses() and PCMSpecify(), generated on the base of the parent classe's PCMListParametrizations().
GaussianPCM
Abstract parent class for all Gaussian models
Implemented S3 generics from PCM (note that PCMCond() is not implemented here, but in daugher classes):
PCMParentClasses.GaussianPCM() {...}
PCMMean.GaussianPCM() {...}
PCMVar.GaussianPCM() {...}
PCMSim.GaussianPCM() {...}
PCMLik.GaussianPCM() {...}
S3 generics specific for Gaussian models (with default implementations)
PCMAbCdEf()
PCMAbCdEf.default() { ... }
PCMLmr()
PCMLmr.default() { ... }
BM, JOU, ...
Follow the same pattern as for OU
BM__Diagonal_WithNonNegativeDiagonal_Sigma_x
MixedGaussian
Maps different GaussianPCMs to different regimes (colors) in the tree.
Constructor:
MixedGaussian() {...}
Implemented S3 generics from PCM:
PCMParentClasses.MixedGaussian() {...}
PCMDescribe.MixedGaussian() {...}
PCMParamCount.MixedGaussian() {...}
PCMCond.MixedGaussian() {...}

Figure S1: **An overview of the PCMBase package.** Each box represents a module. The modules "PCM", "PCMParam" and "PCMTree" define the end-user interface. In particular, the module "PCM" defines the interface for adding model extensions. Function names written in *italic* style denote S3 generic declarations. These functions can be defined or overwritten by inheriting modules, to provide model-specific behavior. The module "GaussianPCM" implements the pruning-wise likelihood calculation. The functions $\vec{\omega}$, $\mathbf{\Phi}$ and $\mathbf{V}$ for each model within the framework must be implemented in specifications of the S3 generic function "PCMCond". It is possible to define parametrizations restricting particular model parameters, e.g. forcing a matrix parameter to be a diagonal matrix.


X0:
(a k-vector)
x01
x02
...
x0k
tree:
(an object of S3 class phylo or class PCMTree)
X:
(a k x N numerical matrix)
x_{11} x_{21} x_{31} ... x_{N1}
x_{12} x_{22} x_{32} ... x_{N2}
...
x_{1k} x_{2k} x_{3k} ... x_{Nk}
model:
(an S3 object of a class deriving from class PCM. This is a named list with class attribute set to the model type, e.g. "OU". The class for each model parameter is shown in angular brackets.)
X0 = a k-vector
<class: c("VectorParameter", "_Global", "numeric")>
, H= a k x k x R array; slices correspond to regimes.
<class: c("MatrixParameter", "matrix")>
, Theta= a k x R matrix
<class: c("VectorParameter", "matrix")>
, Sigma_x= a k x k x R array; slices correspond to regimes.
<class: c("MatrixParameter", "_UpperTriangularWithDiagonal, "_WithNonNegativeDiagonal")>
, Sigmae_x= a k x k x R array; slices correspond to regimes.
<class: c("MatrixParameter", "_UpperTriangularWithDiagonal, "_WithNonNegativeDiagonal")>

- Trait values at the root of the tree are needed as initial state in PCMSim().
- Optionally, X0 can be added to a model as a parameter.

- The tree can be a phylo-object (from package ape) or an object of class PCMTree. The PCMTree extends the class phylo by adding the possibility to partition the tree and to assign regimes (color) to different parts. See ?PCMTree for details.
- Parts represent connected sets of nodes on the tree named by the label of the most ancestral of these nodes, i.e. the root of the part. Thus, a partition is defined by the set of labels of the part root nodes. Each part in the tree is associated with a regime (i.e. color). Different parts can be mapped to the same regime.
- In the case of a model with jumps, the tree must have an integer vector of 0's and 1's, **tree$edge.jump**, indicating the edges at which a jump took place.

- Column-vectors correspond to species (tips) in the order of tree$tip.label.
- Some (but not all) of the entries in a column can be missing measurements (NA) or non-existing traits (NaN).

- **Showing the PCMBase runtime options:**

```
PCMOptions()
```

- **Showing the currently available model types**

```
PCMModels()
```

- **Creating a PCM object for 2 traits and 3 regimes**

```
# This will create a model object with default parameter values
model <- PCM("OU", k=2, regimes = 1:3)
print(model)
```

- **Setting parameter values manually**

```
# The brackets in model$X0[ ]  below are important
model$X0[ ] <- c(0.2, 1.3)
# regime 1:
model$H[,, 1] <- matrix(c(1, 0, 0, 1), 2, 2)
model$Sigma_x[,, 1] <- matrix(c(1, 0, 0.6, 1), 2, 2)
# regime 2:
model$H[,, 2] <- matrix(c(2, 0, 0, 1), 2, 2)
model$Sigma_x[,, 2] <- matrix(c(1, 0, 0.6, 1), 2, 2)
# regime 3:
model$H[,, 3] <- matrix(c(0.12 0, 0, 0.2), 2, 2)
model$Sigma_x[,, 3] <- matrix(c(1, 0, 0.6, 1), 2, 2)

print(model)

# CAUTION: Do not assign entire parameters directly, because
# this will eraze the parameter's attributes including its class.
# The following code is wrong:
model$X0 <- c(0.2, 1.3)
```

- **Setting random parameter values**

```
# First, we generate a random parameter vector for the model.
# The following code generates a random parameter vector for
# the model  drawing from a uniform distribution between the
# model's lower and upper limits:
vecParamValues <- PCMParamRandomVecParams(model)
print(vecParamValues)
# then, we load the generated vector into the model
PCMParamLoadOrStore(model, vecParamValues, 0, load=TRUE)
print(model)
```

- **Simulating a tree with regimes (using the phytools package)**

```
# make results reproducible
set.seed(20)

R <- 3; N <- 200;
# rate mtrix of transition from one regime to another
Q <- matrix(c(-0.1, 0, 0.1, 0, 0.01, -0.01, 0.1, 0, -0.1),
         R, R)
colnames(Q) <- rownames(Q) <- letters[1:R]
tree <- phytools::pbtree(n=N, scale=1, b = 1, d = 0.4)
tree.abc <- phytools::sim.history(tree, Q, anc='a')
tree.abc.s <- phytools::map.to.singleton(tree.abc)
PCMTreeSetRegimesForEdges(tree = tree.abc.s,  regimes =
                 match(names(tree.abc.s$edge.length), letters[1:R]))
PCMTreePlot(tree.abc.s, layout="fan")  # requires ggtree R-package
```

- **Simulating trait evolution**

```
# The following code generates a k x M matrix, where M is the total
# number of nodes in the tree:
data <- PCMSim( tree.abc.s, model, X0=model$X0 )
# The first columns correspond to the tips in the order of tree$tip.label
X <- data[ , 1:PCMTreeNumTips(tree.abc.s)]
PCMPlotTraitData2D(X, tree.abc.s)
```

- **One likelihood calculation on a given model**

```
loglik <- PCMLik(X, tree.abc.s, model)
```

- **Multiple likelihood calculations on different model parameters**

```
# The first N columns correspond to the tips
X <- data[ , 1:PCMTreeNumTips(tree.abc.s)]
if( require(PCMBaseCpp) == FALSE ){
  # using default R-implementation (PCMLmr.default):
  metaInfo <- PCMInfo(X, tree.abc.s, modelOU)
} else {
   # using fast (parallel) C++ implementation (PCMBaseCpp):
  metaInfo <- PCMInfoCpp(X, tree.abc.s, modelOU)
}

# main loop (e.g. during model inference on given data)
while( NOT DONE ) {
  loglik <- PCMLik(X, tree.abc.s, modelOU, metaI = metaInfo)
  model <- propose.next.model(model)   # user-defined function
}
```

Figure S2: **Using the PCMBase package** The main runtime objects are depicted on the top of the figure, followed by coding examples for the specific use-cases. Further coding examples can be found in the package online documentation `https://venelin.github.io/PCMBase/`.

- Calculating the (log–)likelihood of a model, given a tree and trait values at its tips. The user level function for likelihood calculation is `PCMLik()`. This function is implemented in the "GaussianPCM" module and inherited by all of its daughter modules. The calculation proceeds in four steps:

  1. Initially, the model-specific functions $\vec{\omega}$, $\mathbf{\Phi}$ and $\mathbf{V}$ are calculated based on the model parameters $\vec{\Theta}$ and the branch lengths t_i (note that this operation does not need the trait values to be present at any tip or internal node in the tree).
  2. Then, the coefficients $\mathbf{A}_i$, $\vec{b}_i$, $\mathbf{C}_i$, $\vec{d}_i$, $\mathbf{E}_i$ and f_i are calculated for each internal and tip node in the tree based on the values $\vec{\omega}$, $\mathbf{\Phi}$ and $\mathbf{V}$ calculated in the previous step. This calculation is done in the function `PCMAbCdEf()` within the module "GaussianPCM" which, again, is inherited by all model modules (see Fig. S1).
  3. Next, the coefficients $\mathbf{L}_i$, $\vec{m}_i$, r_i are calculated based on the trait values at the tips, the values of $\mathbf{A}_i$, $\vec{b}_i$, $\mathbf{C}_i$, $\vec{d}_i$, $\mathbf{E}_i$ and f_i calculated in the previous step, and the recursive procedure described in Section 2.4, Eqs. (9), (10) and (11).
  4. Finally, the likelihood value is calculated using the formula

     $$\ell(\vec{\Theta}) = pdf(\mathbf{X}|\vec{x}_0, \mathbb{T}, \tilde{\mathbf{\Theta}}) = \exp\left(\vec{x}_0^T \mathbf{L}_0 \vec{x}_0 + \vec{x}_0^T \vec{m}_0 + r_0\right), \tag{S1}$$

     where $\vec{\Theta}$ denotes the set of model parameters and $\vec{x}_0$ is treated either as a parameter (specified as a member `X0` in the model object) or as the point that maximizes the above equation given by the formula:

     $$\vec{x}_0 = -0.5 L_0^{-1} \vec{m}_0. \tag{S2}$$

Appendix A.3. Parameter restrictions

Due to constraints arising from their mathematical formulation or to facilitate the biological interpretation, the parameters of Gaussian models are often subject to restrictions. For example, the $k \times k$ matrix parameter $\mathbf{\Sigma}$ of a BM or an OU process is always symmetric positive-definite; the selection strength matrix $\mathbf{H}$ of an OU process is usually restricted to be at least semi-positive-definite (but not necessarily symmetric), in order to limit the interpretation to cases of stabilizing selection rather than cases of repulsion. Wherever possible, such restrictions of the parameter space should be governed by biological and not by mathematical or computational reasons. As we show in Section 4.1, our OU implementation supports a variety of parametrizations, ranging from the most general cases allowing for singular or negative-definite matrices to limiting cases of strictly positive-definite, symmetric or diagonal matrices. Such parameter restrictions can pose a challenge, during a parameter inference procedure, since many of the proposed parameter values may violate some of the imposed restrictions. This can be overcome through transformation of the parameters before calculating the functions $\vec{\omega}$, $\mathbf{\Phi}$ and $\mathbf{V}$. The **PCMBase** parametrization API (`https://venelin.github.io/PCMBase/articles/PCMParam.html`) allows to specify that a parameter is transformable by adding an `S3` class `"_Transformable"` to its `"class"` attribute. This indicates that, before calculating $\vec{\omega}$, $\mathbf{\Phi}$ and $\mathbf{V}$, the generic function `PCMApplyTransformation()` is to be called on this parameter. After the transformation, the parameter preserves its name but its `S3` class `"_Transformable"` is replaced by `"_Transformed"` and the other `S3` classes of the parameter may change as well (for examples, see the subsections below and the on-line tutorial at `https://venelin.github.io/PCMBase/articles/PCMParam.html#transformation`). The purpose of such parameter is to perform the inference search, e.g. likelihood optimization, in an unrestricted space of transformable parameter values, while guaranteeing that the transformed parameters satisfy all imposed restrictions. Similar techniques, some described briefly below, have been used in other OU implementations, e.g. (Clavel et al., 2015; Bartoszek, 2011).

Appendix A.3.1. Restricting a matrix to be symmetric semi-positive-definite

To ensure that a matrix parameter $\mathbf{Z}$ is symmetric semi-positive-definite, it is sufficient to specify the `S3` classes `c("MatrixParameter", "_CholeskyFactor", "_Transformable")` for this parameter. This indicates that, $\mathbf{Z}$ will be stored as an upper triangular matrix (Cholesky factor) with non-negative elements on the diagonal, and that, before calculating $\vec{\omega}$, $\mathbf{\Phi}$ and $\mathbf{V}$, it will be transformed as

$$\mathbf{Z}_{\_Transformed} = \mathbf{Z}^T_{\_Transformable}\ \mathbf{Z}_{\_Transformable} \tag{S3}$$

After the transformation, the `S3` class attribute of $\mathbf{Z}$ is set to `c("MatrixParameter", "_SemiPositiveDefinite", "_Transformed")`. An `R`-code for this example is provided in the package web-site at `https://venelin.github.io/PCMBase/articles/PCMParam.html#transformation`. In some situations, for example, when cross-checking likelihood values between different implementations, the transformed version $\mathbf{Z}_{\_Transformed}$ may be known *a-priori* and may need to be stored in a PCM model as a Cholesky factor. In this case the matrix $\mathbf{Z}_{\_Transformable}$ can be obtained using the `R`-function `chol()`, namely via `Z <- chol(Z_Transformed)`.

Treatment of the matrix parameters $\mathbf{\Sigma}$, $\mathbf{\Sigma}_e$, $\mathbf{\Sigma}_J$ and the measurement error variance `VE`. The rate-matrix parameter $\mathbf{\Sigma}$ of an OU process (Section 4), the parameter $\mathbf{\Sigma}_e$ denoting the non-phylogenetic component of the variance covariance for a tip (Section 3.3), the matrix $\mathbf{\Sigma}_J$ denoting the variance-covariance matrix of a normal jump of the trait value (Sections 3.5 and 4.2), and the measurement error variance $\mathbf{V}_{err,i}$ for a tip i (Section 3.3) are all symmetric positive-definite matrices. Due to historical reasons, **PCMBase** follows a different convention for these terms. In particular, $\mathbf{\Sigma}$, $\mathbf{\Sigma}_e$, $\mathbf{\Sigma}_J$ are parametrized as `Sigma_x`, `Sigmae_x` and `Sigmaj_x`, all of which are upper-triangular matrices and

$$\begin{array}{rcl} \mathbf{\Sigma} & = & \texttt{Sigma\_x}\ \texttt{Sigma\_x}^T \\ \mathbf{\Sigma}_e & = & \texttt{Sigmae\_x}\ \texttt{Sigmae\_x}^T \\ \mathbf{\Sigma}_J & = & \texttt{Sigmaj\_x}\ \texttt{Sigmaj\_x}^T. \end{array} \tag{S4}$$

Analogically, the terms $\mathbf{V}_{err,i}$ (which are not model parameters but fixed values for each tip i) are specified as

$$\mathbf{V}_{err,i} = \texttt{SE[,,i]}\ \texttt{SE[,,i]}^T, \tag{S5}$$

where `SE[,,i]` is, again, an upper triangular matrix. It is important to note that the decompositions in Eqs. S4 and S5 are analogical, yet different, from the Cholesky decomposition (Eq. S3), respectively, the `R` `chol()` function. The **PCMBase** function `UpperTriFactor()` can be used to decompose a symmetric positive-definite matrix $\mathbf{\Sigma}$ according to Eqs. S4 and S5 (see Fig. S12 for an example).

Appendix A.3.2. Optional restrictions on the symmetry and the eigenvalues of a matrix parameter

If both, the positive-definiteness and the symmetry of a matrix parameter are optional, e.g. the matrix $\mathbf{H}$ of an OU process, such restrictions can be imposed through the Schur decomposition as discussed previously by Clavel et al. (2015). Specifically, we parametrize $\mathbf{H}$ as a $k \times k$-dimensional matrix $\mathbf{H}_{\_Transformable}$ with `S3` classes `c("MatrixParameter", "_Schur", "_Transformed")` as follows:

- the upper triangle of $\mathbf{H}_{\_Transformable}$, excluding the diagonal, specifies $k(k-1)/2$ rotation angles for Givens rotations (Golub and Van Loan, 2013) to obtain a $k \times k$-dimensional orthoganal matrix $\mathbf{Q}$;
- the lower triangle of $\mathbf{H}_{\_Transformable}$ including the diagonal defines a $k \times k$ triangular matrix $\mathbf{T}$.

Then, $\mathbf{H}_{\_Transformed}$ is obtained from $\mathbf{Q}$ and $\mathbf{T}$ as follows (Clavel et al., 2015):

$$\mathbf{H}_{\_Transformed} = \mathbf{Q}\mathbf{T}^T\mathbf{Q}^T \tag{S6}$$

The matrix $\mathbf{H}_{\_Transformed}$ calculated in this way has all of its eigenvalues equal to the elements on the diagonal of $\mathbf{H}_{\_Transformable}$ (Clavel et al., 2015). Thus, by restricting the diagonal of $\mathbf{H}_{\_Transformable}$ to non-negative or strictly positive values, we guarantee that $\mathbf{H}_{\_Transformed}$ will have all of its eigenvalues non-negative or strictly positive (hence, guaranteeing semi- or strict positive definiteness). Further, if $\mathbf{H}_{\_Transformed}$ is diagonal, then so is the matrix $\mathbf{H}_{\_Transformed}$; if $\mathbf{H}_{\_Transformable}$ is upper triangular, then $\mathbf{T}$ is diagonal and the resulting matrix $\mathbf{H}_{\_Transformed}$ is symmetric. Finally, if $\mathbf{H}_{\_Transformable}$ is neither diagonal nor triangular, then the resulting matrix $\mathbf{H}_{\_Transformed}$ is asymmetric.

Appendix A.4. Singular branches

Despite the generality, speed and easiness of use of the package the user has to be aware of a potential pitfall. Theorem 1 and the proof of Thm. 2 indicate a numerical weakness of our method. If a branch ending at node i is extremely short, then the associated with it variance–covariance matrix, $\mathbf{V}_i$, can be computationally singular. Hence, calculating its inverse, a necessary step to obtain the likelihood, will not be possible. **PCMBase** catches such an error and returns it, pointing to the offending node. As an alternative, it is possible to tolerate such an error: if the branch is shorter than a user-specified threshold (runtime options `PCMBase.Skip.Singular` and `PCMBase.Threshold.Skip.Singular`), the whole branch can be treated as a 0–length branch and skipped during the likelihood calculation.

Appendix A.5. Package availability and license

The **PCMBase**-package is licensed under the General Public Licence (GPL) version 3.0. The stable version of the package is published on CRAN (`https://cran.r-project.org/package=PCMBase`), while the most recent version and documentation are accessible from `https://github.com/venelin/PCMBase`.

*Appendix A.6. Parallel likelihood calculation with the **PCMBaseCpp** add–in*

For faster likelihood calculation, it is possible to use multiple processor cores to perform the calculation of $\vec{\omega}$, $\mathbf{\Phi}$, $\mathbf{V}$, $\mathbf{A}_i$, $\vec{b}_i$, $\mathbf{C}_i$, $\vec{d}_i$, $\mathbf{E}_i$ and f_i in parallel. This is possible, given the fact that these coefficients depend solely on the model parameters and on the branch lengths in the tree, see, e.g. Eqs. (18) and (20). The calculation of the coefficients $\mathbf{L}_i$, $\vec{m}_i$, r_i is not fully parallelizable but can be divided in parallelizable steps (generations) using a parallel post–order traversal algorithm (Mitov and Stadler, 2019). We implemented this idea in the accompanying package **PCMBaseCpp**, built on top of the **Armadillo** template library for linear algebra (Sanderson and Curtin, 2016), the **Rcpp** package for seamless `R` and `C++` integration (Eddelbuettel, 2013) and the **SPLITT** library for parallel tree traversal (Mitov and Stadler, 2019).

We compared the performance of the multivariate serial and parallel **PCMBase** implementation against other univariate and multivariate implementations in a separate work (Mitov and Stadler, 2019). As shown in (Mitov and Stadler, 2019), on contemporary multi-core CPUs, the parallel **PCMBaseCpp** implementation can speed up the likelihood calculation up to an order of magnitude starting with 2 traits and trees of 100 to 10'000 tips. For moderate numbers of traits, the use of **PCMBaseCpp** as a C++ back-end is highly recommended, even if not using multi-core parallelization, because serial `C++` code execution is nearly 100 times faster than the equivalent implementation written in `R` (see also Appendix D). For bigger numbers of traits (e.g. $k > 100$), the two implementations will tend to equalize in execution time, since most of the computation will be performed by the underlying linear algebra libraries, i.e. **Armadillo** (Sanderson and Curtin, 2016), **LAPACK** (Anderson et al., 1999) and **BLAS** (Blackford et al., 2002).

The **PCMBaseCpp**-package is licensed under the General Public Licence (GPL) version 3.0. The package code is accessible from `https://github.com/venelin/PCMBaseCpp`.

Appendix B. Pruning log-likelihood calculation for an example 3-trait mixed Gaussian phylogenetic model

To illustrate the pruning log-likelihood calculation algorithm, we show step-by-step how the log-likelihood of a $\mathcal{G}_{LInv}$ model is calculated, given a tree and trait data associated with its tips. In particular, we consider three variants of the example tree and data shown on Fig. 1. In all variants, the measured trait values at the tips of the tree are the same as depicted on Fig. 1, the only difference being in the specification of the active coordinate vectors $\vec{k}_i$ at the internal and root nodes. The three variants are specified as follows:

1. The second trait does not exist for the tips 1, 2 and 3 and has not existed for their ancestral nodes 6, 8 and 9 (see Fig. 1). This corresponds to a biological scenario where a given morphological feature (e.g. a tail), is assumed or known to have been present for the ancestral species at the root of the tree but, subsequently, has disappeared in one of its subclades. By convention, in **PCMBase**, this is specified using `NaN`'s to indicate non-existing traits for the tips. At the time of parsing the data, **PCMBase** propagates this non-existence to the ancestral nodes until reaching a node with at least one descendant, for which the trait has existed (i.e. is either a measured finite value or a `NA`).

2. Everything is set as in the first variant with the only difference that, this time, the second trait is assumed to be non-existing also at the root of the tree. This corresponds to a biological scenario where a given morphological feature has not been present for the ancestral species at the root of the tree but, subsequently, has appeared in one of its subclades. When using **PCMBase**, such custom setting can be specified either by amending by hand the member `pc` of the list returned from the function `PCMInfo` (code example in Fig. S5) or by setting the runtime option `PCMBase.PCMPresentCoordinates` to a user-defined custom function.

3. in the third variant, it is assumed that all three traits have existed for all ancestral nodes and the missing trait values for some of the tips are due to lacking measurements. Following, the convention in **PCMBase**, this can be specified by denoting all missing measurements by `NA`s (i.e. not available, see also Fig. S5)).

The code-listing in Fig. S3 shows the R-code for defining the example tree and trait data:

```
library(ape);
library(PCMBase);

# Non-ultrametric phylogenetic tree of 5 tips in both examples:
treeNewick <- "((5:0.8,4:1.8)7:1.5,(((3:0.8,2:1.6)6:0.7)8:0.6,1:2.6)9:0.9)0;"
tree <- PCMTree(read.tree(text = treeNewick))
# Partitioning the tree in two parts and assign the regimes:
PCMTreeSetPartRegimes(tree, part.regime = c(‘6‘=2), setPartition = TRUE, inplace = TRUE)

# Trait-data:
X <- cbind(
  c(0.3, NaN, 1.4),
  c(0.1, NaN, NA),
  c(0.2, NaN, 1.2),
  c(NA, 0.2, 0.2),
  c(NA, 1.2, 0.4))

colnames(X) <- as.character(1:5)
```

Figure S3: **R-code for defining the example tree and the trait data. `NA` values indicate missing measurements; `NaN` values indicate non-existing traits. This tree and data are shown on Fig. 1.**

Next, we define a 2-regime mixed Gaussian phylogenetic model with an OU process assigned to regime 1 and a BM process assigned to regime 2. The R-code for the model definition is shown on Fig. S4 (see also Table S1).

```
model.OU.BM <- MixedGaussian(
  k = nrow(X1),
  modelTypes = c(
    BM =
"BM__Omitted_X0__UpperTriangularWithDiagonal_WithNonNegativeDiagonal_Sigma_x__Omitted_Sigmae_x",
    OU =
"OU__Omitted_X0__H__Theta__UpperTriangularWithDiagonal_WithNonNegativeDiagonal_Sigma_x__Omitted_Sigmae_x"
  ),
  mapping = c(2, 1),
  Sigmae_x = structure(
    0,
    class = c("MatrixParameter", "_Omitted",
              description = "upper triangular factor of the non-phylogenetic variance-covariance")))

model.OU.BM <- PCMApplyTransformation(model.OU.BM)

# We specify a vector of NAs for the root (X0), to indicate that the log-likelihood
# should be calculated for the optimum value of X0.
model.OU.BM[["X0"]][] <- c(NA, NA, NA)

# selection strength matrix for the OU process
model.OU.BM$`1`$H[,,1] <- cbind(
  c(.1, -.7, .6),
  c(1.3, 2.2, -1.4),
  c(0.8, 0.2, 0.9))
# long term optimum vector for the OU process:
model.OU.BM$`1`$Theta[] <- c(1.3, -.5, .2)

# random drift matrix for the OU process
model.OU.BM$`1`$Sigma_x[,,1] <- cbind(
  c(1, 0, 0),
  c(1.0, 0.5, 0),
  c(0.3, -.8, 1))

# random drift matrix for the BM process
model.OU.BM$`2`$Sigma_x[,,1] <- cbind(
  c(0.8, 0, 0),
  c(1, 0.3, 0),
  c(0.4, 0.5, 0.3))

# This command generates a table of the model parameters in latex format:
print(
  PCMTable(model.OU.BM, removeUntransformed = FALSE),
  xtable = TRUE, type="latex")
```

Figure S4: **R-code defining a 2-regime MGPM.** A summary of the model parameters is shown on Table S1.

regime	type	$X0$	$\mathbf{H}$	$\vec{\Theta}$	$\mathbf{\Sigma}_x$	$\mathbf{\Sigma}$
:global:	NA	$\begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix}$				
1	OU		$\begin{bmatrix} 0.10 & 1.30 & 0.80 \\ -0.70 & 2.20 & 0.20 \\ 0.60 & -1.40 & 0.90 \end{bmatrix}$	$\begin{bmatrix} 1.30 \\ -0.50 \\ 0.20 \end{bmatrix}$	$\begin{bmatrix} 1.00 & 1.00 & 0.30 \\ 0.00 & 0.50 & -0.80 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$	$\begin{bmatrix} 2.09 & 0.26 & 0.30 \\ 0.26 & 0.89 & -0.80 \\ 0.30 & -0.80 & 1.00 \end{bmatrix}$
2	BM				$\begin{bmatrix} 0.80 & 1.00 & 0.40 \\ 0.00 & 0.30 & 0.50 \\ 0.00 & 0.00 & 0.30 \end{bmatrix}$	$\begin{bmatrix} 1.80 & 0.50 & 0.12 \\ 0.50 & 0.34 & 0.15 \\ 0.12 & 0.15 & 0.09 \end{bmatrix}$

Table S1: **A 3-trait mixed Gaussian phylogenetic model with two regimes.** The global-scope parameter $\vec{X}_0$ specifies the trait vector at the root. This is set to a vector of NAs (denoted by dots) to indicate that the likelihood of the model should be calculated at the optimal value of $\vec{X}_0$, i.e. the root trait vector that maximizes the likelihood (see. Eqs. S1 and S2). The other parameters have local scope for each of the two regimes in the model. The regime 1 covering the black branches of the tree (Fig. 1) represents an OU-process; the regime 2 covering the orange branches of the tree represents a BM-process.

To calculate the log-likelihood value, we call the **PCMBase** function `PCMLik` (Fig. S5).

```
# Variant 1:
PCMLik(X[, tree$tip.label], tree, model.OU.BM)
[1] -11.92
attr(,"X0")
[1]  9.566 -6.349 15.254

# Variant 2: First we call the function PCMInfo to obtain a meta-information object.
metaI.variant2 <- PCMInfo(X[, tree$tip.label], tree, model.OU.BM)
# Then, we manually change the vector of present coordinates for the root node.
# The pc-matrix is a k x M matrix of logical values, each column corresponding
# to a node. The active coordinates are indicated by the TRUE entries.
# To prevent assigning to the wrong column in the pc-table, we first assign
# the node-labels as column nanmes.
colnames(metaI.variant2$pc) <- PCMTreeGetLabels(tree)
metaI.variant2$pc[, "0"] <- c(TRUE, FALSE, TRUE)
# After the change, the pc-matrix looks like this:
metaI.variant2[["pc"]]
         5     4     3     2     1     0    7     9     8     6
[1,] FALSE FALSE  TRUE  TRUE  TRUE  TRUE TRUE  TRUE  TRUE  TRUE
[2,]  TRUE  TRUE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE
[3,]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE TRUE  TRUE  TRUE  TRUE
# And the log-likelihood value is:
PCMLik(X[, tree$tip.label], tree, model.OU.BM, metaI = metaI.variant2)
[1] -12.17
attr(,"X0")
[1] 10.611    NaN  8.747

# Variant 3: We set all NaN values in X to NA, to indicate that these are
# missing measurements
X3 <- X
X3[is.nan(X3)] <- NA_real_
PCMLik(X3[, tree$tip.label], tree, model.OU.BM)
[1] -10.71
attr(,"X0")
[1]  15.99  18.34 -11.95
```

Figure S5: **Log-likelihood of the model `MGPM.OU.BM` for the three variants.** The **PCMBase** function `PCMLik` calculates the log-likelihood of a model for a given tree and trait data. The attribute “X0” for the log-likelihood value is the estimated root trait vector maximizing the log-likelihood for the given values of the other model parameters (see Eq. S2).

In the following paragraphs, we show step-by-step how the log-likelihood values in Fig. S5 are calculated. To that end, we use the **PCMBase** function `PCMLikTrace`, which returns a `data.table` with the values of $\vec{\omega}$, $\mathbf{\Phi}$, $\mathbf{V}$, $\mathbf{A}$, $\vec{b}$, $\mathbf{C}$, $\vec{d}$, $\mathbf{E}$, f, $\mathbf{L}$, $\vec{m}$ and r used in the log-likelihood calculation. The values of the above terms for the three example variants are written in tables S2-S10. We validate these values through R-code snippets corresponding to the equations in the text (Figs. S6-S11).

Step 1. Calculating $\vec{\omega}$, $\mathbf{\Phi}$ and $\mathbf{V}$ for each tip or internal node. The equations for calculating the terms $\vec{\omega}$, $\mathbf{\Phi}$ and $\mathbf{V}$ for each tip or internal node i are specific for the model type. In the case of regime 1, the model is an OU process, so we use Eq. 18 and Eq. 20, Thm. 4. The R-code on Fig. S6 shows the calculation for tip 1. In the case of regime 2, the model is a BM process, so we use Eq.

21, Corollary 1 of Thm. 4. The R-code on Fig. S7 shows the calculation for tip 2. The resulting values for the terms $\vec{\omega}$, $\mathbf{\Phi}$, $\mathbf{V}$ and $\mathbf{V}^{-1}$ for all tips and internal nodes for each of the three variants are written in tables S2, S3 and S4, respectively. We notice that, for calculating the term for each node, only data associated with the branch leading to that node is needed (in this case, the branch's length and regime). Hence, this step can be performed in parallel for all nodes in the tree.

```
# OU parameters:
H <- model.OU.BM$`1`$H[,, 1]
theta <- model.OU.BM$`1`$Theta[, 1]
Sigma <- model.OU.BM$`1`$Sigma_x[,, 1] %*% t(model.OU.BM$`1`$Sigma_x[,, 1])

# Eigenvalues of H: these can be complex numbers
lambda <- eigen(H)$values

# Matrix of eigenvectors of H: again, these can be complex
P <- eigen(H)$vectors
P_1 <- solve(P)

# vectors of active coordinates:
pc <- PCMInfo(X[, tree$tip.label], tree, model.OU.BM)$pc

# length of the branch leading to tip 1 (2.6):
t1 <- PCMTreeDtNodes(tree)[endNodeLab == "1", endTime - startTime]

# active coordinates for tip 1 and its parent:
k1 <- pc[, match("1", PCMTreeGetLabels(tree))]
k9 <- pc[, match("9", PCMTreeGetLabels(tree))]

# k x k matrix formed from the pairs of lambda-values and t1 (see Eq. 19):
LambdaMat <- matrix(0, 3, 3)
for(i in 1:3)
  for(j in 1:3)
     LambdaMat[i,j] <- 1/(lambda[i]+lambda[j]) * (1-exp(-(lambda[i]+lambda[j]) * t1))

# omega, Phi, V for tip 1 (see Eq. 20):
print(omega1 <- (diag(1, 3, 3)[k1, ] - expm::expm(-H*t1)[k1, ]) %*% theta[])
     [,1]
[1,] 0.62
[2,] 0.34
print(Phi1 <- expm::expm(-H*t1)[k1, k9])
      [,1]  [,2]
[1,]  0.38 -0.35
[2,] -0.06  0.13
print(V1 <- (P %*% (LambdaMat * (P_1 %*% Sigma %*% t(P_1))) %*% t(P))[k1, k1])
          [,1]      [,2]
[1,]  2.216+0i -0.075+0i
[2,] -0.075-0i  0.258-0i
# Notice that for V1, all complex part are 0.
```

Figure S6: **Calculating $\vec{\omega}$, $\mathbf{\Phi}$ and $\mathbf{V}$ for a node in an OU regime.** The R-code uses the base function `eigen` for the eigendecomposition of the selection strength matrix $\mathbf{H}$, the function `solve` for matrix inversion and the function `expm` from the R-package **expm** for matrix exponentiation.

```
# BM parameter:
Sigma <- model.OU.BM$`2`$Sigma_x[,,1] %*% t(model.OU.BM$`2`$Sigma_x[,,1])

# vectors of active coordinates:
pc <- PCMInfo(X[, tree$tip.label], tree, model.OU.BM)$pc

# length of the branch leading to tip 2 (1.6):
t2 <- PCMTreeDtNodes(tree)[endNodeLab == "2", endTime - startTime]

# active coordinates for tip 1 and its parent:
k2 <- pc[, match("2", PCMTreeGetLabels(tree))]
k6 <- pc[, match("6", PCMTreeGetLabels(tree))]

# omega, Phi, V for tip 1:
print(omega2 <- as.matrix(rep(0, 3)[k2]))
     [,1]
[1,]    0
print(Phi2 <- as.matrix(diag(1, 3, 3)[k2, k6]))
     [,1]
[1,]    1
[2,]    0
print(V2 <- as.matrix((t2*Sigma)[k2, k2]))
     [,1]
[1,] 2.88
```

Figure S7: **Calculating $\vec{\omega}$, $\mathbf{\Phi}$ and $\mathbf{V}$ for a node in a BM regime.**

j	i	t_i	$\vec{k}_i$	$\vec{\omega}_i$	$\mathbf{\Phi}_i$	$\mathbf{V}_i$	$\mathbf{V}_i^{-1}$
6	2	$[1.60]$	$[1]$	$\begin{bmatrix}0.00\\ .\\ .\end{bmatrix}$	$\begin{bmatrix}1.00 & . & 0.00\\ . & . & .\\ . & . & .\end{bmatrix}$	$\begin{bmatrix}2.88 & . & .\\ . & . & .\\ . & . & .\end{bmatrix}$	$\begin{bmatrix}0.35 & . & .\\ . & . & .\\ . & . & .\end{bmatrix}$
6	3	$[0.80]$	$\begin{bmatrix}1\\ 3\end{bmatrix}$	$\begin{bmatrix}0.00\\ .\\ 0.00\end{bmatrix}$	$\begin{bmatrix}1.00 & . & 0.00\\ . & . & .\\ 0.00 & . & 1.00\end{bmatrix}$	$\begin{bmatrix}1.44 & . & 0.10\\ . & . & .\\ 0.10 & . & 0.07\end{bmatrix}$	$\begin{bmatrix}0.76 & . & -1.02\\ . & . & .\\ -1.02 & . & 15.24\end{bmatrix}$
8	6	$[0.70]$	$\begin{bmatrix}1\\ 3\end{bmatrix}$	$\begin{bmatrix}0.00\\ .\\ 0.00\end{bmatrix}$	$\begin{bmatrix}1.00 & . & 0.00\\ . & . & .\\ 0.00 & . & 1.00\end{bmatrix}$	$\begin{bmatrix}1.26 & . & 0.08\\ . & . & .\\ 0.08 & . & 0.06\end{bmatrix}$	$\begin{bmatrix}0.87 & . & -1.16\\ . & . & .\\ -1.16 & . & 17.42\end{bmatrix}$
9	1	$[2.60]$	$\begin{bmatrix}1\\ 3\end{bmatrix}$	$\begin{bmatrix}0.62\\ .\\ 0.34\end{bmatrix}$	$\begin{bmatrix}0.38 & . & -0.35\\ . & . & .\\ -0.06 & . & 0.13\end{bmatrix}$	$\begin{bmatrix}2.22 & . & -0.07\\ . & . & .\\ -0.07 & . & 0.26\end{bmatrix}$	$\begin{bmatrix}0.46 & . & 0.13\\ . & . & .\\ 0.13 & . & 3.92\end{bmatrix}$
9	8	$[0.60]$	$\begin{bmatrix}1\\ 3\end{bmatrix}$	$\begin{bmatrix}-0.04\\ .\\ 0.49\end{bmatrix}$	$\begin{bmatrix}0.89 & . & -0.32\\ . & . & .\\ -0.17 & . & 0.60\end{bmatrix}$	$\begin{bmatrix}1.00 & . & 0.06\\ . & . & .\\ 0.06 & . & 0.21\end{bmatrix}$	$\begin{bmatrix}1.01 & . & -0.26\\ . & . & .\\ -0.26 & . & 4.77\end{bmatrix}$
7	4	$[1.80]$	$\begin{bmatrix}2\\ 3\end{bmatrix}$	$\begin{bmatrix}.\\ -0.86\\ 0.44\end{bmatrix}$	$\begin{bmatrix}. & . & .\\ 0.22 & -0.21 & -0.16\\ -0.11 & 0.29 & 0.24\end{bmatrix}$	$\begin{bmatrix}. & . & .\\ . & 0.37 & -0.21\\ . & -0.21 & 0.25\end{bmatrix}$	$\begin{bmatrix}. & . & .\\ . & 5.16 & 4.30\\ . & 4.30 & 7.58\end{bmatrix}$
7	5	$[0.80]$	$\begin{bmatrix}2\\ 3\end{bmatrix}$	$\begin{bmatrix}.\\ -0.78\\ 0.53\end{bmatrix}$	$\begin{bmatrix}. & . & .\\ 0.25 & 0.03 & -0.12\\ -0.17 & 0.42 & 0.51\end{bmatrix}$	$\begin{bmatrix}. & . & .\\ . & 0.27 & -0.18\\ . & -0.18 & 0.22\end{bmatrix}$	$\begin{bmatrix}. & . & .\\ . & 7.78 & 6.15\\ . & 6.15 & 9.30\end{bmatrix}$
0	9	$[0.90]$	$\begin{bmatrix}1\\ 3\end{bmatrix}$	$\begin{bmatrix}0.02\\ .\\ 0.53\end{bmatrix}$	$\begin{bmatrix}0.81 & -0.61 & -0.39\\ . & . & .\\ -0.17 & 0.42 & 0.47\end{bmatrix}$	$\begin{bmatrix}1.35 & . & 0.03\\ . & . & .\\ 0.03 & . & 0.23\end{bmatrix}$	$\begin{bmatrix}0.74 & . & -0.11\\ . & . & .\\ -0.11 & . & 4.38\end{bmatrix}$
0	7	$[1.50]$	$\begin{bmatrix}1\\ 2\\ 3\end{bmatrix}$	$\begin{bmatrix}0.22\\ -0.88\\ 0.49\end{bmatrix}$	$\begin{bmatrix}0.64 & -0.67 & -0.43\\ 0.24 & -0.18 & -0.16\\ -0.13 & 0.34 & 0.30\end{bmatrix}$	$\begin{bmatrix}1.82 & 0.45 & -0.02\\ 0.45 & 0.34 & -0.20\\ -0.02 & -0.20 & 0.25\end{bmatrix}$	$\begin{bmatrix}1.29 & -3.11 & -2.44\\ -3.11 & 13.06 & 10.44\\ -2.44 & 10.44 & 12.43\end{bmatrix}$
ND	0	ND	$\begin{bmatrix}1\\ 2\\ 3\end{bmatrix}$	ND	ND	ND	ND

Table S2: **Values of t_i, $\vec{k}_i$, $\vec{\omega}_i$, $\mathbf{\Phi}_i$, $\mathbf{V}_i$ and $\mathbf{V}_i^{-1}$ for the example tree and data, variant 1.**Each row corresponds to a node (column i). j: parent node of i; ND: not defined; ".": inactive value (at inactive coordinates).

j	i	t_i	$\vec{k}_i$	$\vec{\omega}_i$	$\mathbf{\Phi}_i$	$\mathbf{V}_i$	$\mathbf{V}_i^{-1}$
6	2	$[1.60]$	$[1]$	$\begin{bmatrix}0.00\\.\\.\end{bmatrix}$	$\begin{bmatrix}1.00 & . & 0.00\\. & . & .\\. & . & .\end{bmatrix}$	$\begin{bmatrix}2.88 & . & .\\. & . & .\\. & . & .\end{bmatrix}$	$\begin{bmatrix}0.35 & . & .\\. & . & .\\. & . & .\end{bmatrix}$
6	3	$[0.80]$	$\begin{bmatrix}1\\3\end{bmatrix}$	$\begin{bmatrix}0.00\\.\\0.00\end{bmatrix}$	$\begin{bmatrix}1.00 & . & 0.00\\. & . & .\\0.00 & . & 1.00\end{bmatrix}$	$\begin{bmatrix}1.44 & . & 0.10\\. & . & .\\0.10 & . & 0.07\end{bmatrix}$	$\begin{bmatrix}0.76 & . & -1.02\\. & . & .\\-1.02 & . & 15.24\end{bmatrix}$
8	6	$[0.70]$	$\begin{bmatrix}1\\3\end{bmatrix}$	$\begin{bmatrix}0.00\\.\\0.00\end{bmatrix}$	$\begin{bmatrix}1.00 & . & 0.00\\. & . & .\\0.00 & . & 1.00\end{bmatrix}$	$\begin{bmatrix}1.26 & . & 0.08\\. & . & .\\0.08 & . & 0.06\end{bmatrix}$	$\begin{bmatrix}0.87 & . & -1.16\\. & . & .\\-1.16 & . & 17.42\end{bmatrix}$
9	1	$[2.60]$	$\begin{bmatrix}1\\3\end{bmatrix}$	$\begin{bmatrix}0.62\\.\\0.34\end{bmatrix}$	$\begin{bmatrix}0.38 & . & -0.35\\. & . & .\\-0.06 & . & 0.13\end{bmatrix}$	$\begin{bmatrix}2.22 & . & -0.07\\. & . & .\\-0.07 & . & 0.26\end{bmatrix}$	$\begin{bmatrix}0.46 & . & 0.13\\. & . & .\\0.13 & . & 3.92\end{bmatrix}$
9	8	$[0.60]$	$\begin{bmatrix}1\\3\end{bmatrix}$	$\begin{bmatrix}-0.04\\.\\0.49\end{bmatrix}$	$\begin{bmatrix}0.89 & . & -0.32\\. & . & .\\-0.17 & . & 0.60\end{bmatrix}$	$\begin{bmatrix}1.00 & . & 0.06\\. & . & .\\0.06 & . & 0.21\end{bmatrix}$	$\begin{bmatrix}1.01 & . & -0.26\\. & . & .\\-0.26 & . & 4.77\end{bmatrix}$
7	4	$[1.80]$	$\begin{bmatrix}2\\3\end{bmatrix}$	$\begin{bmatrix}.\\-0.86\\0.44\end{bmatrix}$	$\begin{bmatrix}. & . & .\\0.22 & -0.21 & -0.16\\-0.11 & 0.29 & 0.24\end{bmatrix}$	$\begin{bmatrix}. & . & .\\. & 0.37 & -0.21\\. & -0.21 & 0.25\end{bmatrix}$	$\begin{bmatrix}. & . & .\\. & 5.16 & 4.30\\. & 4.30 & 7.58\end{bmatrix}$
7	5	$[0.80]$	$\begin{bmatrix}2\\3\end{bmatrix}$	$\begin{bmatrix}.\\-0.78\\0.53\end{bmatrix}$	$\begin{bmatrix}. & . & .\\0.25 & 0.03 & -0.12\\-0.17 & 0.42 & 0.51\end{bmatrix}$	$\begin{bmatrix}. & . & .\\. & 0.27 & -0.18\\. & -0.18 & 0.22\end{bmatrix}$	$\begin{bmatrix}. & . & .\\. & 7.78 & 6.15\\. & 6.15 & 9.30\end{bmatrix}$
0	9	$[0.90]$	$\begin{bmatrix}1\\3\end{bmatrix}$	$\begin{bmatrix}0.02\\.\\0.53\end{bmatrix}$	$\begin{bmatrix}0.81 & . & -0.39\\. & . & .\\-0.17 & . & 0.47\end{bmatrix}$	$\begin{bmatrix}1.35 & . & 0.03\\. & . & .\\0.03 & . & 0.23\end{bmatrix}$	$\begin{bmatrix}0.74 & . & -0.11\\. & . & .\\-0.11 & . & 4.38\end{bmatrix}$
0	7	$[1.50]$	$\begin{bmatrix}1\\2\\3\end{bmatrix}$	$\begin{bmatrix}0.22\\-0.88\\0.49\end{bmatrix}$	$\begin{bmatrix}0.64 & . & -0.43\\0.24 & . & -0.16\\-0.13 & . & 0.30\end{bmatrix}$	$\begin{bmatrix}1.82 & 0.45 & -0.02\\0.45 & 0.34 & -0.20\\-0.02 & -0.20 & 0.25\end{bmatrix}$	$\begin{bmatrix}1.29 & -3.11 & -2.44\\-3.11 & 13.06 & 10.44\\-2.44 & 10.44 & 12.43\end{bmatrix}$
ND	0	ND	$\begin{bmatrix}1\\3\end{bmatrix}$	ND	ND	ND	ND

Table S3: **Values of t_i, $\vec{k}_i$, $\vec{\omega}_i$, $\mathbf{\Phi}_i$, $\mathbf{V}_i$ and $\mathbf{V}_i^{-1}$ for the example tree and data, variant 2.**Each row corresponds to a node (column i). j: parent node of i; ND: not defined; ".": inactive value (at inactive coordinates).

j	i	t_i	$\vec{k}_i$	$\vec{\omega}_i$	$\mathbf{\Phi}_i$	$\mathbf{V}_i$	$\mathbf{V}_i^{-1}$
6	2	$[1.60]$	$[1]$	$\begin{bmatrix}0.00\\ .\\ .\end{bmatrix}$	$\begin{bmatrix}1.00 & 0.00 & 0.00\\ . & . & .\\ . & . & .\end{bmatrix}$	$\begin{bmatrix}2.88 & . & .\\ . & . & .\\ . & . & .\end{bmatrix}$	$\begin{bmatrix}0.35 & . & .\\ . & . & .\\ . & . & .\end{bmatrix}$
6	3	$[0.80]$	$\begin{bmatrix}1\\3\end{bmatrix}$	$\begin{bmatrix}0.00\\ .\\ 0.00\end{bmatrix}$	$\begin{bmatrix}1.00 & 0.00 & 0.00\\ . & . & .\\ 0.00 & 0.00 & 1.00\end{bmatrix}$	$\begin{bmatrix}1.44 & . & 0.10\\ . & . & .\\ 0.10 & . & 0.07\end{bmatrix}$	$\begin{bmatrix}0.76 & . & -1.02\\ . & . & .\\ -1.02 & . & 15.24\end{bmatrix}$
8	6	$[0.70]$	$\begin{bmatrix}1\\2\\3\end{bmatrix}$	$\begin{bmatrix}0.00\\ 0.00\\ 0.00\end{bmatrix}$	$\begin{bmatrix}1.00 & 0.00 & 0.00\\ 0.00 & 1.00 & 0.00\\ 0.00 & 0.00 & 1.00\end{bmatrix}$	$\begin{bmatrix}1.26 & 0.35 & 0.08\\ 0.35 & 0.24 & 0.10\\ 0.08 & 0.10 & 0.06\end{bmatrix}$	$\begin{bmatrix}2.23 & -7.44 & 9.42\\ -7.44 & 40.67 & -57.87\\ 9.42 & -57.87 & 99.76\end{bmatrix}$
9	1	$[2.60]$	$\begin{bmatrix}1\\3\end{bmatrix}$	$\begin{bmatrix}0.62\\ .\\ 0.34\end{bmatrix}$	$\begin{bmatrix}0.38 & -0.52 & -0.35\\ . & . & .\\ -0.06 & 0.17 & 0.13\end{bmatrix}$	$\begin{bmatrix}2.22 & . & -0.07\\ . & . & .\\ -0.07 & . & 0.26\end{bmatrix}$	$\begin{bmatrix}0.46 & . & 0.13\\ . & . & .\\ 0.13 & . & 3.92\end{bmatrix}$
9	8	$[0.60]$	$\begin{bmatrix}1\\2\\3\end{bmatrix}$	$\begin{bmatrix}-0.04\\ -0.69\\ 0.49\end{bmatrix}$	$\begin{bmatrix}0.89 & -0.51 & -0.32\\ 0.23 & 0.17 & -0.10\\ -0.17 & 0.40 & 0.60\end{bmatrix}$	$\begin{bmatrix}1.00 & 0.19 & 0.06\\ 0.19 & 0.24 & -0.17\\ 0.06 & -0.17 & 0.21\end{bmatrix}$	$\begin{bmatrix}2.09 & -4.60 & -4.16\\ -4.60 & 19.50 & 16.55\\ -4.16 & 16.55 & 18.82\end{bmatrix}$
7	4	$[1.80]$	$\begin{bmatrix}2\\3\end{bmatrix}$	$\begin{bmatrix}.\\ -0.86\\ 0.44\end{bmatrix}$	$\begin{bmatrix}. & . & .\\ 0.22 & -0.21 & -0.16\\ -0.11 & 0.29 & 0.24\end{bmatrix}$	$\begin{bmatrix}. & . & .\\ . & 0.37 & -0.21\\ . & -0.21 & 0.25\end{bmatrix}$	$\begin{bmatrix}. & . & .\\ . & 5.16 & 4.30\\ . & 4.30 & 7.58\end{bmatrix}$
7	5	$[0.80]$	$\begin{bmatrix}2\\3\end{bmatrix}$	$\begin{bmatrix}.\\ -0.78\\ 0.53\end{bmatrix}$	$\begin{bmatrix}. & . & .\\ 0.25 & 0.03 & -0.12\\ -0.17 & 0.42 & 0.51\end{bmatrix}$	$\begin{bmatrix}. & . & .\\ . & 0.27 & -0.18\\ . & -0.18 & 0.22\end{bmatrix}$	$\begin{bmatrix}. & . & .\\ . & 7.78 & 6.15\\ . & 6.15 & 9.30\end{bmatrix}$
0	9	$[0.90]$	$\begin{bmatrix}1\\2\\3\end{bmatrix}$	$\begin{bmatrix}0.02\\ -0.81\\ 0.53\end{bmatrix}$	$\begin{bmatrix}0.81 & -0.61 & -0.39\\ 0.25 & -0.02 & -0.13\\ -0.17 & 0.42 & 0.47\end{bmatrix}$	$\begin{bmatrix}1.35 & 0.29 & 0.03\\ 0.29 & 0.28 & -0.18\\ 0.03 & -0.18 & 0.23\end{bmatrix}$	$\begin{bmatrix}1.60 & -3.66 & -3.14\\ -3.66 & 15.62 & 12.92\\ -3.14 & 12.92 & 15.08\end{bmatrix}$
0	7	$[1.50]$	$\begin{bmatrix}1\\2\\3\end{bmatrix}$	$\begin{bmatrix}0.22\\ -0.88\\ 0.49\end{bmatrix}$	$\begin{bmatrix}0.64 & -0.67 & -0.43\\ 0.24 & -0.18 & -0.16\\ -0.13 & 0.34 & 0.30\end{bmatrix}$	$\begin{bmatrix}1.82 & 0.45 & -0.02\\ 0.45 & 0.34 & -0.20\\ -0.02 & -0.20 & 0.25\end{bmatrix}$	$\begin{bmatrix}1.29 & -3.11 & -2.44\\ -3.11 & 13.06 & 10.44\\ -2.44 & 10.44 & 12.43\end{bmatrix}$
ND	0	ND	$\begin{bmatrix}1\\2\\3\end{bmatrix}$	ND	ND	ND	ND

Table S4: **Values of t_i, $\vec{k}_i$, $\vec{\omega}_i$, $\mathbf{\Phi}_i$, $\mathbf{V}_i$ and $\mathbf{V}_i^{-1}$ for the example tree and data, variant 3.**Each row corresponds to a node (column i). j: parent node of i; ND: not defined; ".": inactive value (at inactive coordinates).

Step 2. Calculating $\mathbf{A}$*,* $\vec{b}$*,* $\mathbf{C}$*,* $\vec{d}$*,* $\mathbf{E}$ *and* f *for each tip or internal node.* To calculate the terms $\mathbf{A}$, $\vec{b}$, $\mathbf{C}$, $\vec{d}$, $\mathbf{E}$ and f for each tip or internal node i, we use Eq. 2, Thm. 1. The R-code on Fig. S8 shows the calculation for tip 1. The resulting values for all tips and internal nodes for each of the three variants are written in tables S5, S6 and S7, respectively. We notice that, for calculating the terms for each node, only the corresponding values of $\vec{\omega}$, $\mathbf{\Phi}$ and $\mathbf{V}$ are used. Hence, this step can be performed in parallel after or together with step 1 for all nodes in the tree.

```
# For tip 1. We directly apply Eq. 2, Thm 1:
# We can safely use the real part of V1 (all imaginary parts are 0):
print(V1)
          [,1]      [,2]
[1,]  2.216+0i -0.075+0i
[2,] -0.075-0i  0.258-0i
V1 <- Re(V1)
V1_1 <- solve(V1)

print(A1 <- -0.5*V1_1)
       [,1]   [,2]
[1,] -0.228 -0.066
[2,] -0.066 -1.958
print(E1 <- t(Phi1) %*% V1_1)
      [,1]  [,2]
[1,]  0.16 -0.19
[2,] -0.14  0.45
print(b1 <- V1_1 %*% omega1)
     [,1]
[1,] 0.33
[2,] 1.41
print(C1 <- -0.5 * E1 %*% Phi1)
       [,1]   [,2]
[1,] -0.037  0.040
[2,]  0.040 -0.053
print(d1 <- -E1 %*% omega1)
       [,1]
[1,] -0.038
[2,] -0.065
print(f1 <- -0.5 * (t(omega1) %*% V1_1 %*% omega1 + sum(k1)*log(2*pi) + log(det(V1))))
     [,1]
[1,] -1.9
```

Figure S8: **Calculating $\mathbf{A}$, $\vec{b}$, $\mathbf{C}$, $\vec{d}$, $\mathbf{E}$ and f for a tip or an internal node.** The values shown are for tip 1, variant 1 but the same code is relevant for all nodes and all variants.

j	i	$\vec{k}_i$	$\mathbf{A}_i$	$\vec{b}_i$	$\mathbf{C}_i$	$\vec{d}_i$	$\mathbf{E}_i$	f_i
6	2	$\begin{bmatrix}1\end{bmatrix}$	$\begin{bmatrix}-0.17 & . & . \\ . & . & . \\ . & . & .\end{bmatrix}$	$\begin{bmatrix}0.00 \\ . \\ .\end{bmatrix}$	$\begin{bmatrix}-0.17 & . & -0.00 \\ . & . & . \\ -0.00 & . & -0.00\end{bmatrix}$	$\begin{bmatrix}-0.00 \\ . \\ -0.00\end{bmatrix}$	$\begin{bmatrix}0.35 & . & . \\ . & . & . \\ 0.00 & . & .\end{bmatrix}$	$\begin{bmatrix}-1.45\end{bmatrix}$
6	3	$\begin{bmatrix}1 \\ 3\end{bmatrix}$	$\begin{bmatrix}-0.38 & . & 0.51 \\ . & . & . \\ 0.51 & . & -7.62\end{bmatrix}$	$\begin{bmatrix}0.00 \\ . \\ 0.00\end{bmatrix}$	$\begin{bmatrix}-0.38 & . & 0.51 \\ . & . & . \\ 0.51 & . & -7.62\end{bmatrix}$	$\begin{bmatrix}0.00 \\ . \\ 0.00\end{bmatrix}$	$\begin{bmatrix}0.76 & . & -1.02 \\ . & . & . \\ -1.02 & . & 15.24\end{bmatrix}$	$\begin{bmatrix}-0.66\end{bmatrix}$
8	6	$\begin{bmatrix}1 \\ 3\end{bmatrix}$	$\begin{bmatrix}-0.44 & . & 0.58 \\ . & . & . \\ 0.58 & . & -8.71\end{bmatrix}$	$\begin{bmatrix}0.00 \\ . \\ 0.00\end{bmatrix}$	$\begin{bmatrix}-0.44 & . & 0.58 \\ . & . & . \\ 0.58 & . & -8.71\end{bmatrix}$	$\begin{bmatrix}0.00 \\ . \\ 0.00\end{bmatrix}$	$\begin{bmatrix}0.87 & . & -1.16 \\ . & . & . \\ -1.16 & . & 17.42\end{bmatrix}$	$\begin{bmatrix}-0.52\end{bmatrix}$
9	1	$\begin{bmatrix}1 \\ 3\end{bmatrix}$	$\begin{bmatrix}-0.23 & . & -0.07 \\ . & . & . \\ -0.07 & . & -1.96\end{bmatrix}$	$\begin{bmatrix}0.33 \\ . \\ 1.41\end{bmatrix}$	$\begin{bmatrix}-0.04 & . & 0.04 \\ . & . & . \\ 0.04 & . & -0.05\end{bmatrix}$	$\begin{bmatrix}-0.04 \\ . \\ -0.07\end{bmatrix}$	$\begin{bmatrix}0.16 & . & -0.19 \\ . & . & . \\ -0.14 & . & 0.45\end{bmatrix}$	$\begin{bmatrix}-1.89\end{bmatrix}$
9	8	$\begin{bmatrix}1 \\ 3\end{bmatrix}$	$\begin{bmatrix}-0.51 & . & 0.13 \\ . & . & . \\ 0.13 & . & -2.39\end{bmatrix}$	$\begin{bmatrix}-0.17 \\ . \\ 2.37\end{bmatrix}$	$\begin{bmatrix}-0.50 & . & 0.46 \\ . & . & . \\ 0.46 & . & -0.96\end{bmatrix}$	$\begin{bmatrix}0.54 \\ . \\ -1.48\end{bmatrix}$	$\begin{bmatrix}0.94 & . & -1.02 \\ . & . & . \\ -0.48 & . & 2.95\end{bmatrix}$	$\begin{bmatrix}-1.65\end{bmatrix}$
7	4	$\begin{bmatrix}2 \\ 3\end{bmatrix}$	$\begin{bmatrix}. & . & . \\ . & -2.58 & -2.15 \\ . & -2.15 & -3.79\end{bmatrix}$	$\begin{bmatrix}. \\ -2.54 \\ -0.35\end{bmatrix}$	$\begin{bmatrix}-0.07 & 0.05 & 0.04 \\ 0.05 & -0.17 & -0.14 \\ 0.04 & -0.14 & -0.11\end{bmatrix}$	$\begin{bmatrix}0.53 \\ -0.43 \\ -0.32\end{bmatrix}$	$\begin{bmatrix}. & 0.67 & 0.11 \\ . & 0.17 & 1.31 \\ . & 0.19 & 1.10\end{bmatrix}$	$\begin{bmatrix}-1.34\end{bmatrix}$
7	5	$\begin{bmatrix}2 \\ 3\end{bmatrix}$	$\begin{bmatrix}. & . & . \\ . & -3.89 & -3.07 \\ . & -3.07 & -4.65\end{bmatrix}$	$\begin{bmatrix}. \\ -2.85 \\ 0.10\end{bmatrix}$	$\begin{bmatrix}-0.12 & -0.00 & 0.07 \\ -0.00 & -0.89 & -0.87 \\ 0.07 & -0.87 & -0.89\end{bmatrix}$	$\begin{bmatrix}0.73 \\ 0.05 \\ -0.40\end{bmatrix}$	$\begin{bmatrix}. & 0.88 & -0.06 \\ . & 2.81 & 4.07 \\ . & 2.19 & 4.00\end{bmatrix}$	$\begin{bmatrix}-1.21\end{bmatrix}$
0	9	$\begin{bmatrix}1 \\ 3\end{bmatrix}$	$\begin{bmatrix}-0.37 & . & 0.06 \\ . & . & . \\ 0.06 & . & -2.19\end{bmatrix}$	$\begin{bmatrix}-0.04 \\ . \\ 2.34\end{bmatrix}$	$\begin{bmatrix}-0.32 & 0.36 & 0.32 \\ 0.36 & -0.55 & -0.55 \\ 0.32 & -0.55 & -0.57\end{bmatrix}$	$\begin{bmatrix}0.43 \\ -1.00 \\ -1.12\end{bmatrix}$	$\begin{bmatrix}0.62 & . & -0.83 \\ -0.50 & . & 1.89 \\ -0.34 & . & 2.11\end{bmatrix}$	$\begin{bmatrix}-1.87\end{bmatrix}$
0	7	$\begin{bmatrix}1 \\ 2 \\ 3\end{bmatrix}$	$\begin{bmatrix}-0.64 & 1.55 & 1.22 \\ 1.55 & -6.53 & -5.22 \\ 1.22 & -5.22 & -6.22\end{bmatrix}$	$\begin{bmatrix}1.82 \\ -7.04 \\ -3.63\end{bmatrix}$	$\begin{bmatrix}-0.15 & 0.23 & 0.17 \\ 0.23 & -0.76 & -0.58 \\ 0.17 & -0.58 & -0.44\end{bmatrix}$	$\begin{bmatrix}0.06 \\ 1.16 \\ 0.75\end{bmatrix}$	$\begin{bmatrix}0.40 & -0.22 & -0.70 \\ -1.13 & 3.25 & 3.98 \\ -0.79 & 2.38 & 3.09\end{bmatrix}$	$\begin{bmatrix}-3.47\end{bmatrix}$
ND	0	$\begin{bmatrix}1 \\ 2 \\ 3\end{bmatrix}$	ND	ND	ND	ND	ND	ND

Table S5: **Values of A, $\vec{b}$, C, $\vec{d}$, E and f for the example, variant 1.** For the notation, see also legend for Fig. S2.

j	i	$\vec{k}_i$	$\mathbf{A}_i$	$\vec{b}_i$	$\mathbf{C}_i$	$\vec{d}_i$	$\mathbf{E}_i$	f_i
6	2	$\begin{bmatrix}1\end{bmatrix}$	$\begin{bmatrix}-0.17 & . & . \\ . & . & . \\ . & . & .\end{bmatrix}$	$\begin{bmatrix}0.00 \\ . \\ .\end{bmatrix}$	$\begin{bmatrix}-0.17 & . & -0.00 \\ . & . & . \\ -0.00 & . & -0.00\end{bmatrix}$	$\begin{bmatrix}-0.00 \\ . \\ -0.00\end{bmatrix}$	$\begin{bmatrix}0.35 & . & . \\ . & . & . \\ 0.00 & . & .\end{bmatrix}$	$\begin{bmatrix}-1.45\end{bmatrix}$
6	3	$\begin{bmatrix}1 \\ 3\end{bmatrix}$	$\begin{bmatrix}-0.38 & . & 0.51 \\ . & . & . \\ 0.51 & . & -7.62\end{bmatrix}$	$\begin{bmatrix}0.00 \\ . \\ 0.00\end{bmatrix}$	$\begin{bmatrix}-0.38 & . & 0.51 \\ . & . & . \\ 0.51 & . & -7.62\end{bmatrix}$	$\begin{bmatrix}0.00 \\ . \\ 0.00\end{bmatrix}$	$\begin{bmatrix}0.76 & . & -1.02 \\ . & . & . \\ -1.02 & . & 15.24\end{bmatrix}$	$\begin{bmatrix}-0.66\end{bmatrix}$
8	6	$\begin{bmatrix}1 \\ 3\end{bmatrix}$	$\begin{bmatrix}-0.44 & . & 0.58 \\ . & . & . \\ 0.58 & . & -8.71\end{bmatrix}$	$\begin{bmatrix}0.00 \\ . \\ 0.00\end{bmatrix}$	$\begin{bmatrix}-0.44 & . & 0.58 \\ . & . & . \\ 0.58 & . & -8.71\end{bmatrix}$	$\begin{bmatrix}0.00 \\ . \\ 0.00\end{bmatrix}$	$\begin{bmatrix}0.87 & . & -1.16 \\ . & . & . \\ -1.16 & . & 17.42\end{bmatrix}$	$\begin{bmatrix}-0.52\end{bmatrix}$
9	1	$\begin{bmatrix}1 \\ 3\end{bmatrix}$	$\begin{bmatrix}-0.23 & . & -0.07 \\ . & . & . \\ -0.07 & . & -1.96\end{bmatrix}$	$\begin{bmatrix}0.33 \\ . \\ 1.41\end{bmatrix}$	$\begin{bmatrix}-0.04 & . & 0.04 \\ . & . & . \\ 0.04 & . & -0.05\end{bmatrix}$	$\begin{bmatrix}-0.04 \\ . \\ -0.07\end{bmatrix}$	$\begin{bmatrix}0.16 & . & -0.19 \\ . & . & . \\ -0.14 & . & 0.45\end{bmatrix}$	$\begin{bmatrix}-1.89\end{bmatrix}$
9	8	$\begin{bmatrix}1 \\ 3\end{bmatrix}$	$\begin{bmatrix}-0.51 & . & 0.13 \\ . & . & . \\ 0.13 & . & -2.39\end{bmatrix}$	$\begin{bmatrix}-0.17 \\ . \\ 2.37\end{bmatrix}$	$\begin{bmatrix}-0.50 & . & 0.46 \\ . & . & . \\ 0.46 & . & -0.96\end{bmatrix}$	$\begin{bmatrix}0.54 \\ . \\ -1.48\end{bmatrix}$	$\begin{bmatrix}0.94 & . & -1.02 \\ . & . & . \\ -0.48 & . & 2.95\end{bmatrix}$	$\begin{bmatrix}-1.65\end{bmatrix}$
7	4	$\begin{bmatrix}2 \\ 3\end{bmatrix}$	$\begin{bmatrix}. & . & . \\ . & -2.58 & -2.15 \\ . & -2.15 & -3.79\end{bmatrix}$	$\begin{bmatrix}. \\ -2.54 \\ -0.35\end{bmatrix}$	$\begin{bmatrix}-0.07 & 0.05 & 0.04 \\ 0.05 & -0.17 & -0.14 \\ 0.04 & -0.14 & -0.11\end{bmatrix}$	$\begin{bmatrix}0.53 \\ -0.43 \\ -0.32\end{bmatrix}$	$\begin{bmatrix}. & 0.67 & 0.11 \\ . & 0.17 & 1.31 \\ . & 0.19 & 1.10\end{bmatrix}$	$\begin{bmatrix}-1.34\end{bmatrix}$
7	5	$\begin{bmatrix}2 \\ 3\end{bmatrix}$	$\begin{bmatrix}. & . & . \\ . & -3.89 & -3.07 \\ . & -3.07 & -4.65\end{bmatrix}$	$\begin{bmatrix}. \\ -2.85 \\ 0.10\end{bmatrix}$	$\begin{bmatrix}-0.12 & -0.00 & 0.07 \\ -0.00 & -0.89 & -0.87 \\ 0.07 & -0.87 & -0.89\end{bmatrix}$	$\begin{bmatrix}0.73 \\ 0.05 \\ -0.40\end{bmatrix}$	$\begin{bmatrix}. & 0.88 & -0.06 \\ . & 2.81 & 4.07 \\ . & 2.19 & 4.00\end{bmatrix}$	$\begin{bmatrix}-1.21\end{bmatrix}$
0	9	$\begin{bmatrix}1 \\ 3\end{bmatrix}$	$\begin{bmatrix}-0.37 & . & 0.06 \\ . & . & . \\ 0.06 & . & -2.19\end{bmatrix}$	$\begin{bmatrix}-0.04 \\ . \\ 2.34\end{bmatrix}$	$\begin{bmatrix}-0.32 & . & 0.32 \\ . & . & . \\ 0.32 & . & -0.57\end{bmatrix}$	$\begin{bmatrix}0.43 \\ . \\ -1.12\end{bmatrix}$	$\begin{bmatrix}0.62 & . & -0.83 \\ . & . & . \\ -0.34 & . & 2.11\end{bmatrix}$	$\begin{bmatrix}-1.87\end{bmatrix}$
0	7	$\begin{bmatrix}1 \\ 2 \\ 3\end{bmatrix}$	$\begin{bmatrix}-0.64 & 1.55 & 1.22 \\ 1.55 & -6.53 & -5.22 \\ 1.22 & -5.22 & -6.22\end{bmatrix}$	$\begin{bmatrix}1.82 \\ -7.04 \\ -3.63\end{bmatrix}$	$\begin{bmatrix}-0.15 & . & 0.17 \\ . & . & . \\ 0.17 & . & -0.44\end{bmatrix}$	$\begin{bmatrix}0.06 \\ . \\ 0.75\end{bmatrix}$	$\begin{bmatrix}0.40 & -0.22 & -0.70 \\ . & . & . \\ -0.79 & 2.38 & 3.09\end{bmatrix}$	$\begin{bmatrix}-3.47\end{bmatrix}$
ND	0	$\begin{bmatrix}1 \\ 3\end{bmatrix}$	ND	ND	ND	ND	ND	ND

Table S6: **Values of A, $\vec{b}$, C, $\vec{d}$, E and f for the example, variant 2.** For the notation, see also legend for Fig. S2.

j	i	$\vec{k}_i$	$\mathbf{A}_i$	$\vec{b}_i$	$\mathbf{C}_i$	$\vec{d}_i$	$\mathbf{E}_i$	f_i
6	2	$\begin{bmatrix}1\end{bmatrix}$	$\begin{bmatrix}-0.17 & . & . \\ . & . & . \\ . & . & .\end{bmatrix}$	$\begin{bmatrix}0.00 \\ . \\ .\end{bmatrix}$	$\begin{bmatrix}-0.17 & -0.00 & -0.00 \\ -0.00 & -0.00 & -0.00 \\ -0.00 & -0.00 & -0.00\end{bmatrix}$	$\begin{bmatrix}-0.00 \\ -0.00 \\ -0.00\end{bmatrix}$	$\begin{bmatrix}0.35 & . & . \\ 0.00 & . & . \\ 0.00 & . & .\end{bmatrix}$	$\begin{bmatrix}-1.45\end{bmatrix}$
6	3	$\begin{bmatrix}1 \\ 3\end{bmatrix}$	$\begin{bmatrix}-0.38 & . & 0.51 \\ . & . & . \\ 0.51 & . & -7.62\end{bmatrix}$	$\begin{bmatrix}0.00 \\ . \\ 0.00\end{bmatrix}$	$\begin{bmatrix}-0.38 & -0.00 & 0.51 \\ -0.00 & -0.00 & -0.00 \\ 0.51 & -0.00 & -7.62\end{bmatrix}$	$\begin{bmatrix}0.00 \\ 0.00 \\ 0.00\end{bmatrix}$	$\begin{bmatrix}0.76 & . & -1.02 \\ 0.00 & . & 0.00 \\ -1.02 & . & 15.24\end{bmatrix}$	$\begin{bmatrix}-0.66\end{bmatrix}$
8	6	$\begin{bmatrix}1 \\ 2 \\ 3\end{bmatrix}$	$\begin{bmatrix}-1.12 & 3.72 & -4.71 \\ 3.72 & -20.34 & 28.94 \\ -4.71 & 28.94 & -49.88\end{bmatrix}$	$\begin{bmatrix}0.00 \\ 0.00 \\ 0.00\end{bmatrix}$	$\begin{bmatrix}-1.12 & 3.72 & -4.71 \\ 3.72 & -20.34 & 28.94 \\ -4.71 & 28.94 & -49.88\end{bmatrix}$	$\begin{bmatrix}0.00 \\ 0.00 \\ 0.00\end{bmatrix}$	$\begin{bmatrix}2.23 & -7.44 & 9.42 \\ -7.44 & 40.67 & -57.87 \\ 9.42 & -57.87 & 99.76\end{bmatrix}$	$\begin{bmatrix}0.41\end{bmatrix}$
9	1	$\begin{bmatrix}1 \\ 3\end{bmatrix}$	$\begin{bmatrix}-0.23 & . & -0.07 \\ . & . & . \\ -0.07 & . & -1.96\end{bmatrix}$	$\begin{bmatrix}0.33 \\ . \\ 1.41\end{bmatrix}$	$\begin{bmatrix}-0.04 & 0.06 & 0.04 \\ 0.06 & -0.11 & -0.08 \\ 0.04 & -0.08 & -0.05\end{bmatrix}$	$\begin{bmatrix}-0.04 \\ -0.07 \\ -0.07\end{bmatrix}$	$\begin{bmatrix}0.16 & . & -0.19 \\ -0.22 & . & 0.61 \\ -0.14 & . & 0.45\end{bmatrix}$	$\begin{bmatrix}-1.89\end{bmatrix}$
9	8	$\begin{bmatrix}1 \\ 2 \\ 3\end{bmatrix}$	$\begin{bmatrix}-1.05 & 2.30 & 2.08 \\ 2.30 & -9.75 & -8.28 \\ 2.08 & -8.28 & -9.41\end{bmatrix}$	$\begin{bmatrix}1.04 \\ -5.12 \\ -1.98\end{bmatrix}$	$\begin{bmatrix}-0.65 & 1.19 & 1.04 \\ 1.19 & -4.36 & -3.66 \\ 1.04 & -3.66 & -3.26\end{bmatrix}$	$\begin{bmatrix}-0.08 \\ 2.17 \\ 1.01\end{bmatrix}$	$\begin{bmatrix}1.50 & -2.38 & -3.04 \\ -3.49 & 12.17 & 12.36 \\ -2.71 & 9.46 & 10.98\end{bmatrix}$	$\begin{bmatrix}-1.75\end{bmatrix}$
7	4	$\begin{bmatrix}2 \\ 3\end{bmatrix}$	$\begin{bmatrix}. & . & . \\ . & -2.58 & -2.15 \\ . & -2.15 & -3.79\end{bmatrix}$	$\begin{bmatrix}. \\ -2.54 \\ -0.35\end{bmatrix}$	$\begin{bmatrix}-0.07 & 0.05 & 0.04 \\ 0.05 & -0.17 & -0.14 \\ 0.04 & -0.14 & -0.11\end{bmatrix}$	$\begin{bmatrix}0.53 \\ -0.43 \\ -0.32\end{bmatrix}$	$\begin{bmatrix}. & 0.67 & 0.11 \\ . & 0.17 & 1.31 \\ . & 0.19 & 1.10\end{bmatrix}$	$\begin{bmatrix}-1.34\end{bmatrix}$
7	5	$\begin{bmatrix}2 \\ 3\end{bmatrix}$	$\begin{bmatrix}. & . & . \\ . & -3.89 & -3.07 \\ . & -3.07 & -4.65\end{bmatrix}$	$\begin{bmatrix}. \\ -2.85 \\ 0.10\end{bmatrix}$	$\begin{bmatrix}-0.12 & -0.00 & 0.07 \\ -0.00 & -0.89 & -0.87 \\ 0.07 & -0.87 & -0.89\end{bmatrix}$	$\begin{bmatrix}0.73 \\ 0.05 \\ -0.40\end{bmatrix}$	$\begin{bmatrix}. & 0.88 & -0.06 \\ . & 2.81 & 4.07 \\ . & 2.19 & 4.00\end{bmatrix}$	$\begin{bmatrix}-1.21\end{bmatrix}$
0	9	$\begin{bmatrix}1 \\ 2 \\ 3\end{bmatrix}$	$\begin{bmatrix}-0.80 & 1.83 & 1.57 \\ 1.83 & -7.81 & -6.46 \\ 1.57 & -6.46 & -7.54\end{bmatrix}$	$\begin{bmatrix}1.34 \\ -5.90 \\ -2.54\end{bmatrix}$	$\begin{bmatrix}-0.36 & 0.64 & 0.52 \\ 0.64 & -2.26 & -1.83 \\ 0.52 & -1.83 & -1.53\end{bmatrix}$	$\begin{bmatrix}-0.01 \\ 1.76 \\ 0.96\end{bmatrix}$	$\begin{bmatrix}0.89 & -1.17 & -1.80 \\ -2.22 & 7.31 & 7.94 \\ -1.63 & 5.50 & 6.66\end{bmatrix}$	$\begin{bmatrix}-2.53\end{bmatrix}$
0	7	$\begin{bmatrix}1 \\ 2 \\ 3\end{bmatrix}$	$\begin{bmatrix}-0.64 & 1.55 & 1.22 \\ 1.55 & -6.53 & -5.22 \\ 1.22 & -5.22 & -6.22\end{bmatrix}$	$\begin{bmatrix}1.82 \\ -7.04 \\ -3.63\end{bmatrix}$	$\begin{bmatrix}-0.15 & 0.23 & 0.17 \\ 0.23 & -0.76 & -0.58 \\ 0.17 & -0.58 & -0.44\end{bmatrix}$	$\begin{bmatrix}0.06 \\ 1.16 \\ 0.75\end{bmatrix}$	$\begin{bmatrix}0.40 & -0.22 & -0.70 \\ -1.13 & 3.25 & 3.98 \\ -0.79 & 2.38 & 3.09\end{bmatrix}$	$\begin{bmatrix}-3.47\end{bmatrix}$
ND	0	$\begin{bmatrix}1 \\ 2 \\ 3\end{bmatrix}$	ND	ND	ND	ND	ND	ND

Table S7: **Values of A, $\vec{b}$, C, $\vec{d}$, E and f for the example, variant 3.** For the notation, see also legend for Fig. S2.

Step 3. Calculating $\mathbf{L}$*,* $\vec{m}$ *and* r *for each internal node and the root of the tree.* We follow Eq. 10 and Eq. 11, Thm. 2 to calculate the terms $\mathbf{L}$, $\vec{m}$ and r for each internal node and the root of the tree. To be more explicit, let's denote the summand terms corresponding to the j's daughter node i in Eq. 10 and Eq. 11 as follows.

$$\begin{aligned}
\mathbf{L}_{ji} &= \begin{cases} \mathbf{C}_i & \text{if } i \text{ is a tip} \\ \mathbf{C}_i - (1/4)\mathbf{E}_i \left(\mathbf{A}_i + \mathbf{L}_i\right)^{-1} \mathbf{E}_i^T & \text{otherwise;} \end{cases} \\
\vec{m}_{ji} &= \begin{cases} \vec{d}_i + \mathbf{E}_i \vec{x}_i & \text{if } i \text{ is a tip} \\ \vec{d}_i - (1/2)\mathbf{E}_i \left(\mathbf{A}_i + \mathbf{L}_i\right)^{-1} \left(\vec{b}_i + \vec{m}_i\right) & \text{otherwise;} \end{cases} \\
r_{ji} &= \begin{cases} \vec{x}_i^T \mathbf{A}_i \vec{x}_i + \vec{x}_i^T \vec{b}_i + f_i & \text{if } i \text{ is a tip} \\ f_i + r_i + (k_i/2)\log(2\pi) & \\ \quad -(1/2)\log(|(-2)\left(\mathbf{A}_i + \mathbf{L}_i\right)|) & \\ \quad -(1/4)\left(\vec{b}_i + \vec{m}_i\right)^T \left(\mathbf{A}_i + \mathbf{L}_i\right)^{-1} \left(\vec{b}_i + \vec{m}_i\right) & \text{otherwise.} \end{cases}
\end{aligned} \tag{S3}$$

We notice that the terms $\mathbf{L}_{ji}$, $\vec{m}_{ji}$ and r_{ji} are defined for every tip or internal node i but not for the root 0. For an internal node i with parent j, the terms $\mathbf{L}_{ji}$, $\vec{m}_{ji}$ and r_{ji} depend on i's $\mathbf{L}_i$, $\vec{m}_i$ and r_i. For any internal or root node j, the terms $\mathbf{L}_j$, $\vec{m}_j$ and r_j are calculated as the sums of the corresponding terms $\mathbf{L}_{ji}$, $\vec{m}_{ji}$ and r_{ji} for each $i \in Desc(j)$. Therefore, to obtain the root terms $\mathbf{L}_0$, $\vec{m}_0$ and r_0, we perform a postorder traversal of the tree (i.e. pruning). The R-code snippets on Figs. S9 and S10 show how this postorder calculation is done for a part of the tree, using the trait values in variant 1. The resulting values for $\mathbf{L}_{ji}$, $\vec{m}_{ji}$, r_{ji}, $\mathbf{L}_j$, $\vec{m}_j$, and r_j are listed in Tables S8, S9 and S10.

```
# For tip 2 with parent node 6, we use the following terms stored in Table S5:
A2 <- matrix(-0.17)
b2 <- 0.0
C2 <- rbind(c(-0.17, 0),
              c(0, 0))
d2 <- c(0.0, 0.0)
E2 <- matrix(c(0.35, 0), nrow = 2, ncol = 1)
f2 <- -1.45
k2 <- 1

# Now we apply Eq. S3:
print(L62 <- C2)
      [,1] [,2]
[1,] -0.17    0
[2,]  0.00    0
print(m62 <- d2 + E2 %*% X[k2, "2", drop = FALSE])
         2
[1,] 0.035
[2,] 0.000
print(r62 <- t(X[k2, "2", drop = FALSE]) %*% A2 %*% X[k2, "2", drop = FALSE] +
          t(X[k2, "2", drop = FALSE]) %*% b2 + f2)
        2
2 -1.4517

# For tip 3 with parent node 6, applying Eq. S3, we obtain (see Table S8):
L63 <- rbind(c(-0.38, 0.51),
               c(0.51, -7.62))
m63 <- c(-1.07, 18.09)
r63 <- -11.41

# Now, we sum the terms L6i, m6i and r6i over all daughters of 6 (i) to obtain:
print(L6 <- L62 + L63)
      [,1]  [,2]
[1,] -0.55  0.51
[2,]  0.51 -7.62
print(m6 <- m62 + m63)
          2
[1,] -1.035
[2,] 18.090
print(r6 <- r62 + r63)
        2
2 -12.862
```

Figure S9: **Postorder tree traversal for calculating $\mathbf{L}_0$, $\vec{m}_0$, and r_0. I: Calculating $\mathbf{L}_{ji}$, $\vec{m}_{ji}$, and r_{ji} for tip nodes in example variant 1.**

```
# Using Eq. S3, we obtain L86, m86, r86, using the values for A,b,C,d,E,f in Table S5:
A6 <- rbind(c(-0.44, 0.58),
              c(0.58, -8.71))
b6 <- c(0.0, 0.0)
C6 <- rbind(c(-0.44, 0.58),
              c(0.58, -8.71))
d6 <- c(0.0, 0.0)
E6 <- rbind(c(0.87, -1.16),
              c(-1.16, 17.42))
f6 <- -0.52
k6 <- c(1, 3)

print(L86 <- C6 - (1/4)*E6 %*% solve(A6 + L6) %*% t(E6))
         [,1]     [,2]
[1,] -0.24819  0.27124
[2,]  0.27124 -4.06431
print(m86 <- d6 - (1/2)*E6 %*% solve(A6 + L6) %*% (b6+m6))
            2
[1,] -0.56799
[2,]  9.64900
print(r86 <- f6+r6+(length(k6)/2)*log(2*pi) -
          (1/2)*log(det(-2*(A6+L6))) -
          (1/4)*t(b6+m6) %*% solve(A6+L6) %*% (b6+m6))
        2
2 -8.5723

# Because 8 is a singleton node, we immediately obtain L8, m8, r8:
L8 <- L86; m8 <- m86; r8 <- r86;
```

Figure S10: **Postorder tree traversal for calculating $\mathbf{L}_0$, $\vec{m}_0$, and r_0. II: Calculating $\mathbf{L}_{ji}$, $\vec{m}_{ji}$, and r_{ji} for an internal node in example variant 1.**

j	i	$\vec{x}_i$	$\vec{k}_i$	$\mathbf{L}_i$	$\vec{m}_i$	r_i	$\mathbf{L}_{ji}$	$\vec{m}_{ji}$	r_{ji}
6	2	$\begin{bmatrix}0.1\\NaN\\NA\end{bmatrix}$	$\begin{bmatrix}1\end{bmatrix}$	ND	ND	ND	$\begin{bmatrix}-0.174 & . & -0.000\\. & . & .\\-0.000 & . & -0.000\end{bmatrix}$	$\begin{bmatrix}0.035\\.\\0.000\end{bmatrix}$	$\begin{bmatrix}-1.450\end{bmatrix}$
6	3	$\begin{bmatrix}0.2\\NaN\\1.2\end{bmatrix}$	$\begin{bmatrix}1\\3\end{bmatrix}$	ND	ND	ND	$\begin{bmatrix}-0.381 & . & 0.508\\. & . & .\\0.508 & . & -7.622\end{bmatrix}$	$\begin{bmatrix}-1.067\\.\\18.089\end{bmatrix}$	$\begin{bmatrix}-11.405\end{bmatrix}$
8	6	$\begin{bmatrix}NA\\NaN\\NA\end{bmatrix}$	$\begin{bmatrix}1\\3\end{bmatrix}$	$\begin{bmatrix}-0.555 & . & 0.508\\. & . & .\\0.508 & . & -7.622\end{bmatrix}$	$\begin{bmatrix}-1.032\\.\\18.089\end{bmatrix}$	$\begin{bmatrix}-12.855\end{bmatrix}$	$\begin{bmatrix}-0.243 & . & 0.271\\. & . & .\\0.271 & . & -4.065\end{bmatrix}$	$\begin{bmatrix}-0.568\\.\\9.648\end{bmatrix}$	$\begin{bmatrix}-8.571\end{bmatrix}$
9	1	$\begin{bmatrix}0.3\\NaN\\1.4\end{bmatrix}$	$\begin{bmatrix}1\\3\end{bmatrix}$	ND	ND	ND	$\begin{bmatrix}-0.037 & . & 0.040\\. & . & .\\0.040 & . & -0.053\end{bmatrix}$	$\begin{bmatrix}-0.249\\.\\0.520\end{bmatrix}$	$\begin{bmatrix}-3.735\end{bmatrix}$
9	8	$\begin{bmatrix}NA\\NaN\\NA\end{bmatrix}$	$\begin{bmatrix}1\\3\end{bmatrix}$	$\begin{bmatrix}-0.243 & . & 0.271\\. & . & .\\0.271 & . & -4.065\end{bmatrix}$	$\begin{bmatrix}-0.568\\.\\9.648\end{bmatrix}$	$\begin{bmatrix}-8.571\end{bmatrix}$	$\begin{bmatrix}-0.195 & . & 0.250\\. & . & .\\0.250 & . & -0.594\end{bmatrix}$	$\begin{bmatrix}-0.402\\.\\1.267\end{bmatrix}$	$\begin{bmatrix}-4.248\end{bmatrix}$
7	4	$\begin{bmatrix}NA\\0.2\\0.2\end{bmatrix}$	$\begin{bmatrix}2\\3\end{bmatrix}$	ND	ND	ND	$\begin{bmatrix}-0.068 & 0.054 & 0.040\\0.054 & -0.174 & -0.141\\0.040 & -0.141 & -0.114\end{bmatrix}$	$\begin{bmatrix}0.683\\-0.138\\-0.066\end{bmatrix}$	$\begin{bmatrix}-2.349\end{bmatrix}$
7	5	$\begin{bmatrix}NA\\1.2\\0.4\end{bmatrix}$	$\begin{bmatrix}2\\3\end{bmatrix}$	ND	ND	ND	$\begin{bmatrix}-0.115 & -0.001 & 0.070\\-0.001 & -0.893 & -0.869\\0.070 & -0.869 & -0.890\end{bmatrix}$	$\begin{bmatrix}1.762\\5.043\\3.834\end{bmatrix}$	$\begin{bmatrix}-13.880\end{bmatrix}$
0	9	$\begin{bmatrix}NA\\NaN\\NA\end{bmatrix}$	$\begin{bmatrix}1\\3\end{bmatrix}$	$\begin{bmatrix}-0.232 & . & 0.291\\. & . & .\\0.291 & . & -0.647\end{bmatrix}$	$\begin{bmatrix}-0.651\\.\\1.786\end{bmatrix}$	$\begin{bmatrix}-7.983\end{bmatrix}$	$\begin{bmatrix}-0.140 & 0.162 & 0.143\\0.162 & -0.200 & -0.183\\0.143 & -0.183 & -0.170\end{bmatrix}$	$\begin{bmatrix}-0.262\\0.422\\0.429\end{bmatrix}$	$\begin{bmatrix}-7.430\end{bmatrix}$
0	7	$\begin{bmatrix}NA\\NA\\NA\end{bmatrix}$	$\begin{bmatrix}1\\2\\3\end{bmatrix}$	$\begin{bmatrix}-0.183 & 0.053 & 0.110\\0.053 & -1.066 & -1.010\\0.110 & -1.010 & -1.004\end{bmatrix}$	$\begin{bmatrix}2.444\\4.906\\3.769\end{bmatrix}$	$\begin{bmatrix}-16.230\end{bmatrix}$	$\begin{bmatrix}-0.052 & 0.052 & 0.035\\0.052 & -0.113 & -0.082\\0.035 & -0.082 & -0.060\end{bmatrix}$	$\begin{bmatrix}1.222\\-0.396\\-0.174\end{bmatrix}$	$\begin{bmatrix}-10.947\end{bmatrix}$
ND	0	$\begin{bmatrix}NA\\NA\\NA\end{bmatrix}$	$\begin{bmatrix}1\\2\\3\end{bmatrix}$	$\begin{bmatrix}-0.192 & 0.214 & 0.178\\0.214 & -0.313 & -0.265\\0.178 & -0.265 & -0.230\end{bmatrix}$	$\begin{bmatrix}0.960\\0.026\\0.255\end{bmatrix}$	$\begin{bmatrix}-18.377\end{bmatrix}$	ND	ND	ND

Table S8: **Results from the postorder calculation of L, $\vec{m}$, r for example variant 1.**

j	i	$\vec{x}_i$	$\vec{k}_i$	$\mathbf{L}_i$	$\vec{m}_i$	r_i	$\mathbf{L}_{ji}$	$\vec{m}_{ji}$	r_{ji}
6	2	$\begin{bmatrix}0.1\\NaN\\NA\end{bmatrix}$	$\begin{bmatrix}1\end{bmatrix}$	ND	ND	ND	$\begin{bmatrix}-0.174 & . & -0.000\\. & . & .\\-0.000 & . & -0.000\end{bmatrix}$	$\begin{bmatrix}0.035\\.\\0.000\end{bmatrix}$	$\begin{bmatrix}-1.450\end{bmatrix}$
6	3	$\begin{bmatrix}0.2\\NaN\\1.2\end{bmatrix}$	$\begin{bmatrix}1\\3\end{bmatrix}$	ND	ND	ND	$\begin{bmatrix}-0.381 & . & 0.508\\. & . & .\\0.508 & . & -7.622\end{bmatrix}$	$\begin{bmatrix}-1.067\\.\\18.089\end{bmatrix}$	$\begin{bmatrix}-11.405\end{bmatrix}$
8	6	$\begin{bmatrix}NA\\NaN\\NA\end{bmatrix}$	$\begin{bmatrix}1\\3\end{bmatrix}$	$\begin{bmatrix}-0.555 & . & 0.508\\. & . & .\\0.508 & . & -7.622\end{bmatrix}$	$\begin{bmatrix}-1.032\\.\\18.089\end{bmatrix}$	$\begin{bmatrix}-12.855\end{bmatrix}$	$\begin{bmatrix}-0.243 & . & 0.271\\. & . & .\\0.271 & . & -4.065\end{bmatrix}$	$\begin{bmatrix}-0.568\\.\\9.648\end{bmatrix}$	$\begin{bmatrix}-8.571\end{bmatrix}$
9	1	$\begin{bmatrix}0.3\\NaN\\1.4\end{bmatrix}$	$\begin{bmatrix}1\\3\end{bmatrix}$	ND	ND	ND	$\begin{bmatrix}-0.037 & . & 0.040\\. & . & .\\0.040 & . & -0.053\end{bmatrix}$	$\begin{bmatrix}-0.249\\.\\0.520\end{bmatrix}$	$\begin{bmatrix}-3.735\end{bmatrix}$
9	8	$\begin{bmatrix}NA\\NaN\\NA\end{bmatrix}$	$\begin{bmatrix}1\\3\end{bmatrix}$	$\begin{bmatrix}-0.243 & . & 0.271\\. & . & .\\0.271 & . & -4.065\end{bmatrix}$	$\begin{bmatrix}-0.568\\.\\9.648\end{bmatrix}$	$\begin{bmatrix}-8.571\end{bmatrix}$	$\begin{bmatrix}-0.195 & . & 0.250\\. & . & .\\0.250 & . & -0.594\end{bmatrix}$	$\begin{bmatrix}-0.402\\.\\1.267\end{bmatrix}$	$\begin{bmatrix}-4.248\end{bmatrix}$
7	4	$\begin{bmatrix}NA\\0.2\\0.2\end{bmatrix}$	$\begin{bmatrix}2\\3\end{bmatrix}$	ND	ND	ND	$\begin{bmatrix}-0.068 & 0.054 & 0.040\\0.054 & -0.174 & -0.141\\0.040 & -0.141 & -0.114\end{bmatrix}$	$\begin{bmatrix}0.683\\-0.138\\-0.066\end{bmatrix}$	$\begin{bmatrix}-2.349\end{bmatrix}$
7	5	$\begin{bmatrix}NA\\1.2\\0.4\end{bmatrix}$	$\begin{bmatrix}2\\3\end{bmatrix}$	ND	ND	ND	$\begin{bmatrix}-0.115 & -0.001 & 0.070\\-0.001 & -0.893 & -0.869\\0.070 & -0.869 & -0.890\end{bmatrix}$	$\begin{bmatrix}1.762\\5.043\\3.834\end{bmatrix}$	$\begin{bmatrix}-13.880\end{bmatrix}$
0	9	$\begin{bmatrix}NA\\NaN\\NA\end{bmatrix}$	$\begin{bmatrix}1\\3\end{bmatrix}$	$\begin{bmatrix}-0.232 & . & 0.291\\. & . & .\\0.291 & . & -0.647\end{bmatrix}$	$\begin{bmatrix}-0.651\\.\\1.786\end{bmatrix}$	$\begin{bmatrix}-7.983\end{bmatrix}$	$\begin{bmatrix}-0.140 & . & 0.143\\. & . & .\\0.143 & . & -0.170\end{bmatrix}$	$\begin{bmatrix}-0.262\\.\\0.429\end{bmatrix}$	$\begin{bmatrix}-7.430\end{bmatrix}$
0	7	$\begin{bmatrix}NA\\NA\\NA\end{bmatrix}$	$\begin{bmatrix}1\\2\\3\end{bmatrix}$	$\begin{bmatrix}-0.183 & 0.053 & 0.110\\0.053 & -1.066 & -1.010\\0.110 & -1.010 & -1.004\end{bmatrix}$	$\begin{bmatrix}2.444\\4.906\\3.769\end{bmatrix}$	$\begin{bmatrix}-16.230\end{bmatrix}$	$\begin{bmatrix}-0.052 & 0.053 & 0.035\\0.053 & -1.066 & -1.010\\0.035 & -1.010 & -0.060\end{bmatrix}$	$\begin{bmatrix}1.222\\4.906\\-0.174\end{bmatrix}$	$\begin{bmatrix}-10.947\end{bmatrix}$
ND	0	$\begin{bmatrix}NA\\NaN\\NA\end{bmatrix}$	$\begin{bmatrix}1\\3\end{bmatrix}$	$\begin{bmatrix}-0.192 & . & 0.178\\. & . & .\\0.178 & . & -0.230\end{bmatrix}$	$\begin{bmatrix}0.960\\.\\0.255\end{bmatrix}$	$\begin{bmatrix}-18.377\end{bmatrix}$	ND	ND	ND

Table S9: **Results from the postorder calculation of L, $\vec{m}$, r for example variant 2.**

j	i	$\vec{x}_i$	$\vec{k}_i$	$\mathbf{L}_i$	$\vec{m}_i$	r_i	$\mathbf{L}_{ji}$	$\vec{m}_{ji}$	r_{ji}
6	2	$\begin{bmatrix}0.1\\NA\\NA\end{bmatrix}$	$\begin{bmatrix}1\end{bmatrix}$	ND	ND	ND	$\begin{bmatrix}-0.174&-0.000&-0.000\\-0.000&-0.000&-0.000\\-0.000&-0.000&-0.000\end{bmatrix}$	$\begin{bmatrix}0.035\\0.000\\0.000\end{bmatrix}$	$\begin{bmatrix}-1.450\end{bmatrix}$
6	3	$\begin{bmatrix}0.2\\NA\\1.2\end{bmatrix}$	$\begin{bmatrix}1\\3\end{bmatrix}$	ND	ND	ND	$\begin{bmatrix}-0.381&-0.000&0.508\\-0.000&-0.000&-0.000\\0.508&-0.000&-7.622\end{bmatrix}$	$\begin{bmatrix}-1.067\\0.000\\18.089\end{bmatrix}$	$\begin{bmatrix}-11.405\end{bmatrix}$
8	6	$\begin{bmatrix}NA\\NA\\NA\end{bmatrix}$	$\begin{bmatrix}1\\2\\3\end{bmatrix}$	$\begin{bmatrix}-0.555&0.000&0.508\\0.000&0.000&0.000\\0.508&0.000&-7.622\end{bmatrix}$	$\begin{bmatrix}-1.032\\0.000\\18.089\end{bmatrix}$	$\begin{bmatrix}-12.855\end{bmatrix}$	$\begin{bmatrix}-0.243&-0.000&0.271\\-0.000&0.000&-0.000\\0.271&-0.000&-4.065\end{bmatrix}$	$\begin{bmatrix}-0.568\\-0.000\\9.648\end{bmatrix}$	$\begin{bmatrix}-8.571\end{bmatrix}$
9	1	$\begin{bmatrix}0.3\\NA\\1.4\end{bmatrix}$	$\begin{bmatrix}1\\3\end{bmatrix}$	ND	ND	ND	$\begin{bmatrix}-0.037&0.059&0.040\\0.059&-0.109&-0.076\\0.040&-0.076&-0.053\end{bmatrix}$	$\begin{bmatrix}-0.249\\0.711\\0.520\end{bmatrix}$	$\begin{bmatrix}-3.735\end{bmatrix}$
9	8	$\begin{bmatrix}NA\\NA\\NA\end{bmatrix}$	$\begin{bmatrix}1\\2\\3\end{bmatrix}$	$\begin{bmatrix}-0.243&-0.000&0.271\\-0.000&0.000&-0.000\\0.271&-0.000&-4.065\end{bmatrix}$	$\begin{bmatrix}-0.568\\-0.000\\9.648\end{bmatrix}$	$\begin{bmatrix}-8.571\end{bmatrix}$	$\begin{bmatrix}-0.195&0.213&0.250\\0.213&-0.318&-0.426\\0.250&-0.426&-0.594\end{bmatrix}$	$\begin{bmatrix}-0.402\\0.860\\1.267\end{bmatrix}$	$\begin{bmatrix}-4.248\end{bmatrix}$
7	4	$\begin{bmatrix}NA\\0.2\\0.2\end{bmatrix}$	$\begin{bmatrix}2\\3\end{bmatrix}$	ND	ND	ND	$\begin{bmatrix}-0.068&0.054&0.040\\0.054&-0.174&-0.141\\0.040&-0.141&-0.114\end{bmatrix}$	$\begin{bmatrix}0.683\\-0.138\\-0.066\end{bmatrix}$	$\begin{bmatrix}-2.349\end{bmatrix}$
7	5	$\begin{bmatrix}NA\\1.2\\0.4\end{bmatrix}$	$\begin{bmatrix}2\\3\end{bmatrix}$	ND	ND	ND	$\begin{bmatrix}-0.115&-0.001&0.070\\-0.001&-0.893&-0.869\\0.070&-0.869&-0.890\end{bmatrix}$	$\begin{bmatrix}1.762\\5.043\\3.834\end{bmatrix}$	$\begin{bmatrix}-13.880\end{bmatrix}$
0	9	$\begin{bmatrix}NA\\NA\\NA\end{bmatrix}$	$\begin{bmatrix}1\\2\\3\end{bmatrix}$	$\begin{bmatrix}-0.232&0.272&0.291\\0.272&-0.427&-0.502\\0.291&-0.502&-0.647\end{bmatrix}$	$\begin{bmatrix}-0.651\\1.571\\1.786\end{bmatrix}$	$\begin{bmatrix}-7.983\end{bmatrix}$	$\begin{bmatrix}-0.089&0.140&0.106\\0.140&-0.258&-0.204\\0.106&-0.204&-0.163\end{bmatrix}$	$\begin{bmatrix}-0.394\\1.044\\0.833\end{bmatrix}$	$\begin{bmatrix}-8.380\end{bmatrix}$
0	7	$\begin{bmatrix}NA\\NA\\NA\end{bmatrix}$	$\begin{bmatrix}1\\2\\3\end{bmatrix}$	$\begin{bmatrix}-0.183&0.053&0.110\\0.053&-1.066&-1.010\\0.110&-1.010&-1.004\end{bmatrix}$	$\begin{bmatrix}2.444\\4.906\\3.769\end{bmatrix}$	$\begin{bmatrix}-16.230\end{bmatrix}$	$\begin{bmatrix}-0.052&0.052&0.035\\0.052&-0.113&-0.082\\0.035&-0.082&-0.060\end{bmatrix}$	$\begin{bmatrix}1.222\\-0.396\\-0.174\end{bmatrix}$	$\begin{bmatrix}-10.947\end{bmatrix}$
ND	0	$\begin{bmatrix}NA\\NA\\NA\end{bmatrix}$	$\begin{bmatrix}1\\2\\3\end{bmatrix}$	$\begin{bmatrix}-0.141&0.192&0.141\\0.192&-0.372&-0.286\\0.141&-0.286&-0.223\end{bmatrix}$	$\begin{bmatrix}0.827\\0.648\\0.659\end{bmatrix}$	$\begin{bmatrix}-19.328\end{bmatrix}$	ND	ND	ND

Table S10: **Results from the postorder calculation of L, $\vec{m}$, r for example variant 3.**

Step 4. Calculating the log-likelihood value. Once the terms $\mathbf{L}_0$, $\vec{m}_0$, and r_0 have been obtained, the log-likelihood is calculated using Eq. S1. If no value has been specified for the root vector $\vec{x}_0$, the log-likelihood is calculated at the optimal value of $\vec{x}_0$ obtained via Eq. S2. The R-code on Fig. S11 shows how these calculations are done for the three variants.

The log-likelihood values differ between Variants 1, 2 and 3. We notice that both, the log-likelihood values and the estimated trait vectors of the root differ between the three variants (Figs. S5 and S11). These differences are due to the different assumptions of trait existence at the internal nodes. In the case of variants 1 and 2, the sole difference is in the assumed non-existence of trait 2 at the root for variant 2. This implies that, in variant 2, the root vector $\vec{x}_0$ has two elements, and the matrix $\mathbf{\Phi}_i$ for its direct descendants $i \in \{7, 9\}$ has two columns (see Condition 2.a in Dfn. 1 and Tables S2 and S3). This is equivalent to setting the second column of the 3×3 matrices $\mathbf{\Phi}_7$ and $\mathbf{\Phi}_9$ to zero so that the effect of the second trait value at the root on the trait expectations at its descendants is cancelled. As a result, the terms $\mathbf{L}_0$, $\vec{m}_0$, $\hat{\vec{x}}_0$ and the log-likelihood differ between variants 1 and 2. In variant 3, the likelihood represents the marginal probability density function of the observed trait values, assuming that all traits have existed at all internal nodes. Hence, again, we observe a different log-likelihood value with respect to variants 1 and 2. These examples reveal the assumptions of trait existence at the root and the internal nodes of the tree represent an essential part of the trait model, with "essential" meaning that these assumptions affect the dimensionality and the value of the expected trait vector and the terms $\vec{\omega}_i$, $\mathbf{\Phi}_i$ and $\mathbf{V}_i$ for particular internal and/or tip nodes in the tree. The **PCMBase** package allows to explicitly specify such trait existence assumptions for each node in the tree. This enables incorporating knowledge about the trait existence from the fossil record into the phylogenetic model.

```
# Variant 1 (using the values of L0, m0 and r0 from Table S8):
L0 <- rbind(c(-0.192, 0.214, 0.178),
              c(0.214, -0.313, -0.265),
              c(0.178, -0.265, -0.230))
m0 <- c(0.96, 0.026, 0.255)
r0 <- -18.377
# Use Eq. S2 to estimate the optimal X0 and Eq. S1 to calculate the log-likelihood:
print(t(x0Hat <- -0.5*solve(L0) %*% m0))
       [,1]    [,2]   [,3]
[1,] 9.6409 -6.2514 15.218
print(ll0 <- t(x0Hat) %*% L0 %*% x0Hat + t(x0Hat) %*% m0 + r0)
       [,1]
[1,] -11.89

# Variant 2 (using the values of L0, m0 and r0 from Table S9):
L0 <- rbind(c(-0.192, 0.178),   c(0.178, -0.230))
m0 <- c(0.96, 0.255)
r0 <- -18.377
print(t(x0Hat <- -0.5*solve(L0) %*% m0))
       [,1]   [,2]
[1,] 10.668 8.8105
print(ll0 <- t(x0Hat) %*% L0 %*% x0Hat + t(x0Hat) %*% m0 + r0)
        [,1]
[1,] -12.133

# Variant 3 (for maximal precision, use L0, m0 and r0 returned from calling PCMLikTrace):
traceTable3 <- PCMLikTrace(X3[, tree$tip.label], tree, model.OU.BM)
setkey(traceTable3, i)
print(L0 <- traceTable3[list("0")]$L_i[[1]])
         [,1]     [,2]     [,3]
[1,] -0.14084  0.19195  0.14072
[2,]  0.19195 -0.37153 -0.28617
[3,]  0.14072 -0.28617 -0.22328
print(m0 <- traceTable3[list("0")]$m_i[[1]])
[1] 0.82729 0.64841 0.65898
print(r0 <- traceTable3[list("0")]$r_i[[1]])
[1] -19.328
print(t(x0Hat <- -0.5*solve(L0) %*% m0))
      [,1]   [,2]    [,3]
[1,] 15.99 18.342 -11.955
print(ll0 <- t(x0Hat) %*% L0 %*% x0Hat + t(x0Hat) %*% m0 + r0)
        [,1]
[1,] -10.706
```

Figure S11: **Calculating the log-likelihood for variants 1, 2 and 3.** Small differences with the log-likelihood values in Fig. S5 are due to rounding error in Tables S8 and S9.

Appendix C. Technical correctness

Validating the technical correctness is an important but often neglected step in the development of likelihood calculation software. This step is particularly relevant for complex multivariate models, because logical errors can occur in many levels, such as the mathematical equations for the different terms involved in the likelihood, the programming code implementing these equations, the code responsible for the tree traversal, the parametrization of the model and the preprocessing of the input data. These logical errors add up to numerical errors caused by limited floating point precision, which can be extremely hard to identify. Ultimately, these errors lead to wrong likelihood values, false parameter inference and wrong analysis. All these concerns motivate for a systematic approach of testing the correctness of the software. We implemented two types of tests for validating the correctness described in the following sections.

Appendix C.1. Comparison with existing tools

As a quick validation test, we compared the likelihood calculation with the `R`-package **mvMORPH** (Clavel et al., 2015). Figure S12 shows the `R`-code, in which a phylogenetic tree and bi-variate trait data are simulated using an "OUBM" model with a shift occurring 3 Ma ago simultaneously on all branches of the tree. To simulate the tree and data, we used the `R`-packages **ape** and **mvMORPH**. Then, the log-likelihood values of the simulated model are calculated using the packages **mvMORPH** and **PCMBase**. This test confirmed an exact match between the likelihood values calculated by the two packages.

Appendix C.2. Simulation-based validation

While the validation approach described in the previous section is suitable for quick testing during development, it is limited to the types of models already implemented within the existing software, e.g. **mvMORPH**. To overcome this limitation, we implemented a technical correctness test of the three models currently implemented in **PCMBase** using the method of posterior quantiles proposed by Cook et al. (2006). The posterior quantiles method (Alg. S1) is a simulation based approach. It employs the fact that, for a fixed prior distribution of the model parameters, the sample of posterior quantiles of any model parameter, θ is uniform (see e.g. Cook et al., 2006; Mitov and Stadler, 2019, for details). Thus, any deviation from uniformity of the posterior quantile sample for any of the model parameters indicates the presence of an error, either in the simulation software, or in the likelihood calculator used to generate the posterior samples. In this way, the posterior quantile method allows to validate at once the simulation and the likelihood calculation functionality of **PCMBase**.

Algorithm S1 Posterior quantiles method

```
1: Sample "true" parameters $\vec{\Theta}$ from the prior;
2: Simulate random data, $\mathbf{X}_{\vec{\Theta}}$, under the model specified by $\vec{\Theta}$;
3: Generate a sample $S_\theta$ from the posterior distribution $P_\theta = P(\theta|\mathbf{X}_{\vec{\Theta}})$;
4: Calculate the empirical quantile of the "true" $\theta$ in $S_\theta$;
```

We used a fixed non–ultrametric tree of $N = 515$ tips with two regimes "a" and "b". The tree was generated using the functions `pbtree()` and `sim.history()` from the package **phytools** (Revell, 2011). We implemented the posterior quantile test using the **BayesValidate** `R`–package (Cook et al., 2006). For each model we set a parametrization and a prior distribution as follows:

- BM
  3 parameters: $\vec{\Theta}_{BM} = [\Sigma_{11}, \Sigma_{12}, \Sigma_{e,11}]$, such that

$$\mathbf{\Sigma}_a = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12} & \Sigma_{11} \end{pmatrix}, \mathbf{\Sigma}_b = \begin{pmatrix} \Sigma_{11} & 0 \\ 0 & \Sigma_{11} \end{pmatrix}, \mathbf{\Sigma}_{e,a} = \mathbf{\Sigma}_{e,b} = \begin{pmatrix} \Sigma_{e,11} & 0 \\ 0 & \Sigma_{e,11} \end{pmatrix}$$

prior: $\Sigma_{11} \sim \text{Exp}(1)$, $\Sigma_{12} \sim \mathcal{U}(-0.9\Sigma_{11},\, 0.9\Sigma_{11})$, $\Sigma_{e,11} \sim \text{Exp}(10)$.

- OU
  8 parameters: $\vec{\Theta}_{OU} = [\vec{\Theta}_{BM}, \theta_{b,1}, \theta_{b,2}, H_{b,11}, H_{b,12}, H_{b,22}]$, such that $\mathbf{\Sigma}_a$, $\mathbf{\Sigma}_b$, $\mathbf{\Sigma}_{e,a}$ and $\mathbf{\Sigma}_{e,b}$ are defined as for BM and

  $$\vec{\theta}_a = \begin{pmatrix} 0 \\ 0 \end{pmatrix},\ \vec{\theta}_b = \begin{pmatrix} \theta_{b,1} \\ \theta_{b,2} \end{pmatrix},\ \mathbf{H}_a = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix},\ \mathbf{H}_b = \begin{pmatrix} H_{b,11} & H_{b,12} \\ H_{b,12} & H_{b,11} \end{pmatrix}$$

  prior: for parameters in $\vec{\Theta}_{BM}$ the same prior has been used as for the BM model; for the new parameters, the prior has been set as $\theta_{b,1} \sim \mathcal{N}(1, .25)$, $\theta_{b,2} \sim \mathcal{N}(2, .5)$, $H_{b,11} \sim \text{Exp}(1)$, $H_{b,22} \sim \text{Exp}(1)$, $H_{b,12} \sim \mathcal{U}(-0.9\sqrt{H_{b,11}H_{b,22}},\, 0.9\sqrt{H_{b,11}H_{b,22}})$.

- JOU
  9 parameters: $\vec{\Theta}_{JOU} = [\vec{\Theta}_{OU}, \Sigma_{j,11}]$, such that $\mathbf{\Sigma}_a$, $\mathbf{\Sigma}_b$, $\mathbf{\Sigma}_{e,a}$, $\mathbf{\Sigma}_{e,b}$, $\vec{\theta}_a$, $\vec{\theta}_b$, $\mathbf{H}_a$, $\mathbf{H}_b$ are defined as for OU and

  $$\vec{\mu}_{j,a} = \begin{pmatrix} -\theta_{b,1} \\ -\theta_{b,2} \end{pmatrix},\ \vec{\mu}_{j,b} = \begin{pmatrix} \theta_{b,1} \\ \theta_{b,2} \end{pmatrix},\ \mathbf{\Sigma}_{j,a} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},\ \mathbf{\Sigma}_{j,b} = \begin{pmatrix} \Sigma_{j,11} & 0 \\ 0 & \Sigma_{j,11} \end{pmatrix}$$

  prior: for parameters in $\vec{\Theta}_{OU}$ the same prior has been used as for the OU model; for the new parameter, the prior has been set as $\Sigma_{j,11} \sim \text{Exp}(10)$.

For each, model, we ran the function `validate()` from the **BayesValidate** package, setting the number of replications to 48. The results are summarized in Fig. S13. All Bonferroni adjusted p–values of the absolute Z_θ statistics were above 0.2, showing that the posterior quantiles did not deviate from uniformity (see Cook et al., 2006, for details on Z_θ statistic).

```
> library(mvMORPH)
> library(PCMBase)
> library(PCMBaseCpp)
>
> set.seed(1, kind = "Mersenne-Twister", normal.kind = "Inversion")
>
> # Generating a random tree
> tree <- ape::rtree(50)
>
> # Providing a tree whith the shift mapped on
> tot <- max(nodeHeights(tree))
> age <- tot-3    # The shift occured 3 Ma ago
> tree <- make.era.map(tree, c(0, age))
>
> # Convert the tree with mapped regimes to a PCMTree object
> pcmTree <- PCMTree(map.to.singleton(tree))
> PCMTreeSetRegimesForEdges(pcmTree, names(pcmTree[["edge.length"]]))
>
> # Plotting the trees for illustration that they have the same regimes
> #plotSimmap(tree,fsize=0.6,node.numbers=FALSE,lwd=3, pts=FALSE)
> #PCMTreePlot(pcmTree)
>
> # Simulating trait evolution using the mvMORPH package
> alpha<-matrix(c(1,0.1,0,2),2)
> sigma<-matrix(c(.1,.1,0,.1),2)
> theta<-c(2,3)
>
> data<-mvSIM(tree, param=list(
+   sigma=sigma, alpha=alpha, ntraits=2, theta=theta,
+   names_traits=c("head.size","mouth.size")), model="OUBM", nsim=1)
>
> # Create a log-likelihood calculation function for an OUBM model using mvMORPH:
> llmvMORPH <- mvSHIFT(
+   tree, data, model = "OUBM", optimization = "fixed")[["llik"]]
>
> # Calculating the log-likelihood value of the parameters using mvMORPH:
> llmvMORPH(vecParams <- c(alpha[lower.tri(alpha, diag = TRUE)],
+                                              sigma[lower.tri(sigma, diag = TRUE)],
+                                        theta = theta), root.mle = FALSE)
[1] -515.6049
>
> # Create a PCM model object using PCMBase. For simplicity, we use here a 2-regime
# OU model. Alternatively, we could have used a mixed Gaussian model with an OU
# and a BM regime.
> pcmOUBM <- PCM("OU", k = 2, regimes = c("1", "2"))
>
> # Specify the parameter values for the model
> pcmOUBM[["H"]][,,1] <- alpha %*% t(alpha)
> pcmOUBM[["Sigma_x"]][,,1]<- pcmOUBM[["Sigma_x"]][,,2]<-UpperTriFactor(sigma %*% t(sigma))
> pcmOUBM[["Theta"]][,1] <- theta
> pcmOUBM[["X0"]][] <- theta
>
> # Calculate the log-likelihood value using PCMBaseCpp (faster execution):
> metaICpp <- PCMInfoCpp(t(data), pcmTree, pcmOUBM)
> PCMLik(t(data), pcmTree, pcmOUBM, metaI = metaICpp)
[1] -515.6049
attr(,"X0")
[1] 2 3
> # The values are matchine up to numerical error
> all.equal(target = llmvMORPH(vecParams, root.mle = FALSE),
+           current = PCMLik(t(data), pcmTree, pcmOUBM), check.attributes = FALSE)
[1] TRUE
> all.equal(target = llmvMORPH(vecParams, root.mle = FALSE),
+           current = PCMLik(t(data), pcmTree, pcmOUBM, metaI = metaICpp),
+           check.attributes = FALSE)
[1] TRUE
```

Figure S12: **Comparing the log-likelihood values obtained via PCMBase, PCMBaseCpp, and mvMORPH.**

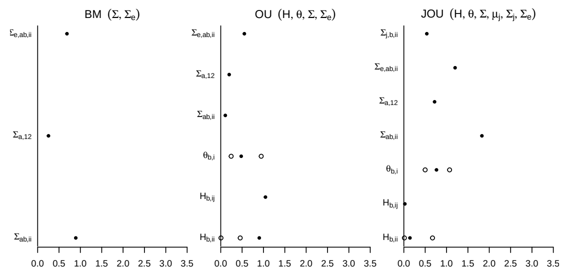


Figure S13: **Absolute Z_θ–statistics for the three posterior quantile tests.** The Z_θ statistic is described by Cook et al. (2006). High values indicate deviation from uniformity of the posterior quantile distribution for an individual model parameter (circles) or a batch of several model parameters (bullets). The reported values, smaller than 3 for all parameters, had insignificant p-values as well as Bonferroni–adjusted p–values. This means that the posterior quantile samples for the different model parameters do not significantly deviate from uniform distributions, therefore, confirming the technical correctness of the implementation. The plots were generated using the package **BayesValidate** (Cook et al., 2006).

Appendix D. Likelihood calculation time

In this section we compare the likelihood calculation time of the **PCMBaseCpp** `R`-package with currently existing multivariate PCM software and we evaluate the time complexity of the calculation in terms of number of regimes, tips and traits.

Appendix D.1. Comparison with existing tools

We implemented a benchmark comparing the likelihood calculation times for the `R`-packages **mvMORPH** and **PCMBaseCpp**. The comparison was done on trees of $N \in \{25, 50, 100, 150, 200, 250\}$ and simulated data for $k \in \{1, 2, 4, 8\}$ traits. Similar to section Appendix C.1, we used an "OUBM" model with a shift occurring 3 Ma ago simultaneously on all branches of the tree. The `R`-code for the benchmark is available in the file "data-raw/CompareLLTimes_mvMORPH.R" available from the **PCMBaseCpp** github repository (`https://github.com/venelin/PCMBaseCpp`). The log-likelihood calculation times for this benchmark are shown in Table S11. **PCMBaseCpp** outperformed **mvMORPH** in all tested cases. Furthermore, the relative difference in likelihood calculation time was increasing with N and k. This is explained by the higher power (in terms of N) complexity of the likelihood calculation algorithm used in **mvMORPH**.

k	Tool	N:	25	50	100	150	200	250
1	**mvMORPH**		0.33	0.37	0.60	1.00	2.10	2.40
1	**PCMBaseCpp**		0.27	0.26	0.40	0.40	0.40	0.30
2	**mvMORPH**		0.50	0.78	2.10	4.70	7.90	13.00
2	**PCMBaseCpp**		0.48	0.71	1.10	1.70	2.00	2.50
4	**mvMORPH**		1.09	3.56	11.90	33.50	78.90	166.00
4	**PCMBaseCpp**		0.76	1.25	2.20	3.20	3.90	4.80
8	**mvMORPH**		8.28	31.32	100.20	412.00	591.10	1422.60
8	**PCMBaseCpp**		2.23	3.68	7.40	10.20	13.20	18.10

Table S11: **Comparison of the log-likelihood calculation times between the R-packages mvMORPH (Clavel et al., 2015) and PCMBaseCpp.** All calculation times are in milliseconds. The benchmark was executed on an personal computer running a 2.3GHz Intel(R) Core i7 processor with 4 cores.

Appendix D.2. Evaluating the computational complexity

In this section, we evaluate the time complexity of the likelihood calculation algorithm for $\mathcal{G}_{LInv}$ models. Let the number of traits k is fixed, R denotes the number of regimes in a $\mathcal{G}_{LInv}$ model of a fixed type, such as OU, and $E = M - 1$ denotes the number of edges (branches) in the tree. The likelihood calculation consists of operations executed just once (e.g. transformations of global model parameters (see Section Appendix A.3), estimating the root value (Eq. S2) and calculating the final likelihood (Eq. S1)), operations executed once for each regime (e.g. transformations of local parameters, see Section Appendix A.3) and operations executed once for each edge (e.g. calculations of the model-specific terms $\vec{\omega}$, $\mathbf{\Phi}$, and the generic terms $\mathbf{V}$, $\mathbf{A}$, ..., f, $\mathbf{L}_{ji}$, $\vec{m}_{ji}$ and r_{ji}, see Appendix B). Hence, the likelihood calculation time can be expressed as the linear function

$$T = \alpha + \beta R + \gamma E + \epsilon, \tag{S4}$$

where α, β and γ are constants, and ϵ is a small value including time measurement error and random fluctuations in the computation time. In the case of a mixed Gaussian (MG) model with more than one type the likelihood calculation time can be expressed as the linear function

$$T = \alpha + \sum_{m \in \mathcal{M}} (\beta_m R_m + \gamma_m E_m) + \epsilon, \tag{S5}$$

where $\mathcal{M} \subset \mathcal{G}_{LInv}$ denotes the finite set of model types used in the MG, R_m denotes the number of regimes with assigned model type m, E_m denotes the number of edges with assigned model m, β_m and γ_m are the constants β and γ specific for the model m, and ϵ is specified as above.

As an example, we estimate the constants α, β and γ for the BM and OU models with shifts for up to 32 traits. Then, we use these constants to predict the likelihood calculation time of an MG model over these two model types. We use the function `BenchmarkRvsCpp` from the **PCMBaseCpp** package. This function performs a likelihood calculation benchmark comparing the `R`-implementation in the **PCMBase** package versus a multivariate and a univariate (for $k = 1$ only) implementation of the above model types in the **PCMBaseCpp** package. The benchmark trees and model type mappings for the MG model are summarized in Table S12.

N	E	R	R_{BM}	R_{OU}	E_{BM}	E_{OU}
10	18	2	1	1	15	3
100	198	4	2	2	128	70
1000	1998	11	5	6	903	1095
10000	19998	11	5	6	8627	11371

Table S12: **Summary of the trees and mixed Gaussian models used for measuring the likelihood calculation times of the PCMBaseCpp package.** Each row summarizes one tree and a mixed Gaussian model with BM and OU model types mapped to its regimes. N: number of tips; E: number of edges; R: number of regimes.

We ran the benchmark on a personal computer (PC) running OS X on an Intel(R) Core(TM) i7-4850HQ CPU @ 2.30 GHz, after compiling the **PCMBaseCpp** package using the complier `clang++` version 8. The parallel OpenMP option was disabled during the run, so that only serial execution on one processor core was measured (for a report of parallel likelihood calculation times, see Mitov and Stadler, 2019). We omit reporting the likelihood calculation times of the `R`-implementation noting that these were between one and three orders of magnitude worse than the times of the `C++` implementations (likelihood calculation times for up to 16 traits for this `R`-implementation have been reported in Mitov and Stadler (2019)). For each case represented by a trait number $k \in \{1, 2, 4, 8, 16, 32\}$, a number of tips $N \in \{10, 100, 1000, 10000\}$, a model type $\Updownarrow \in \{\text{BM, OU, MG}\}$ and implementation $\mathcal{I} \in \{\text{multivariate, univariate}\}$, we measured the the likelihood calculation time excluding the initial transformation of the model parameters (in this case, the Schur transformation for the parameter $\mathbf{H}$ of the OU process, see Section Appendix A.3) and the total likelihood calculation time including this transformation (Table S13).

k	N	R	—BM	**BM**	—OU	**OU**	—MG	**MG**	—BM1	**BM1**	—OU1	**OU1**	—MG1	**MG1**
1	10	2	0.3	**0.3**	0.4	**0.5**	0.5	**0.7**	0.3	**0.2**	0.3	**0.4**	0.5	**0.7**
1	100	4	0.7	**0.7**	0.9	**1.0**	1.2	**1.4**	0.2	**0.2**	0.3	**0.5**	0.6	**0.9**
1	1000	11	4.9	**4.9**	5.8	**6.3**	6.5	**7.3**	0.4	**0.5**	0.4	**0.9**	1.4	**2.1**
1	10000	11	53.6	**53.6**	62.4	**62.9**	60.1	**61.3**	2.5	**2.6**	3.2	**3.3**	4.1	**4.4**
2	10	2	0.3	**0.3**	0.4	**0.7**	0.5	**0.8**						
2	100	4	0.9	**0.9**	1.1	**1.6**	1.3	**1.8**						
2	1000	11	6.8	**6.8**	8.5	**9.5**	8.8	**10.1**						
2	10000	11	71.5	**73.0**	87.2	**90.5**	82.7	**86.6**						
4	10	2	0.4	**0.4**	0.5	**1.3**	0.6	**1.0**						
4	100	4	1.7	**1.6**	2.0	**3.4**	2.2	**3.1**						
4	1000	11	14.3	**14.4**	18.1	**21.1**	17.3	**20.0**						
4	10000	11	143.8	**145.5**	179.4	**185.0**	165.3	**168.2**						
8	10	2	0.8	**0.7**	1.0	**3.6**	1.0	**2.4**						
8	100	4	4.9	**4.8**	6.6	**12.1**	6.2	**9.0**						
8	1000	11	48.0	**48.7**	64.1	**78.8**	57.9	**66.3**						
8	10000	11	487.3	**482.1**	630.5	**648.4**	568.4	**582.9**						
16	10	2	1.4	**1.3**	2.6	**17.8**	1.9	**9.3**						
16	100	4	11.8	**11.3**	22.2	**52.8**	15.7	**31.5**						
16	1000	11	112.7	**112.1**	210.2	**288.4**	162.0	**205.3**						
16	10000	11	1106.2	**1110.7**	2097.6	**2184.3**	1559.9	**1600.5**						
32	10	2	4.6	**4.7**	11.0	**95.7**	6.9	**55.5**						
32	100	4	49.5	**49.6**	104.8	**280.1**	71.2	**157.4**						
32	1000	11	495.8	**495.1**	1028.8	**1498.6**	766.3	**1023.1**						
32	10000	11	4940.1	**4932.6**	10314.3	**10749.6**	7559.7	**7827.8**						

Table S13: **Likelihood calculation time (in milliseconds) for C++-implementations of the OU, BM and MG models.** k: number of traits; N: number of tips; R: number of regimes. The other columns are aranged in pairs for each implementation: column titles prefixed with a "—" (normal style) denote the likelihood calculation time excluding the time for parameter transformation; column titles in bold style denote the total likelihood calculation time including the time for parameter transformation. BM, OU, MG: multivariate BM, OU and Mixed Gaussian implementations; BM1, OU1, MG1: univariate BM, OU and Mixed Gaussian implementations. Note that, in the case of the BM model, there is no parameter transformation, and, due to measurement error, it is possible for the left values to slightly exceed the right values. The benchmark was executed on an personal computer running a 2.3GHz Intel(R) Core i7 processor with 4 cores.

As estimates of α, β and γ for the BM and OU model types we use the intercept and the two slopes of an ordinary least squares (OLS) regression of the likelihood calculation time on R and E (Table S14).

k	Constant	—BM	**BM**	—OU	**OU**	—BM1	**BM1**	—OU1	**OU1**
1	α	0.463143	**0.480222**	0.607168	**0.580315**	0.238996	**0.181692**	0.333010	**0.398032**
	β	-0.086901	**-0.092838**	-0.101753	**-0.050035**	-0.007816	**0.002341**	-0.016528	**0.021193**
	γ	0.002705	**0.002705**	0.003143	**0.003144**	0.000120	**0.000120**	0.000150	**0.000133**
2	α	0.420806	**0.450530**	0.463269	**0.687829**				
	β	-0.071651	**-0.084791**	-0.062516	**-0.010216**				
	γ	0.003594	**0.003677**	0.004369	**0.004499**				
4	α	0.382812	**0.334448**	0.385721	**0.693049**				
	β	-0.045994	**-0.039606**	-0.023892	**0.206972**				
	γ	0.007197	**0.007281**	0.008965	**0.009102**				
8	α	0.605562	**0.036232**	0.108701	**0.223263**				
	β	-0.129859	**0.052239**	0.103291	**1.395566**				
	γ	0.024409	**0.024074**	0.031466	**0.031642**				
16	α	-0.038683	**-0.043790**	1.135699	**3.325943**				
	β	0.217806	**0.114518**	-0.028951	**6.816576**				
	γ	0.055200	**0.055480**	0.104852	**0.105312**				
32	α	-0.372107	**-0.234173**	3.433175	**4.174764**				
	β	0.262973	**0.252746**	-0.453319	**42.592297**				
	γ	0.246904	**0.246525**	0.515841	**0.513894**				

Table S14: **Estimated values of α, β and γ from the likelihood calculation times in Table S13.** All values are in milliseconds. Due to very small values of γ in the case of $k = 1$, all values have been rounded to the sixth decimal digit. The column titles denoting the different model implementations have the same meaning as in Table S13. The benchmark was executed on an personal computer running a 2.3GHz Intel(R) Core i7 processor with 4 cores.

Now, we can plug the values from Tables S12 and S14 in Eq. S5 to predict the likelihood calculation time for the MG model[1]. For example in the case $k = 8$ and $N = 1000$, we obtain

$$T = 0.223263+0.052239*5+0.024074*903+1.395566*6+0.031642*1095 = 65.24. \tag{S6}$$

We notice that the predicted value of 65.24 ms does not deviate substantially from the measured time of 66.3 ms (Table S13, row for $k = 8$, $N = 1000$, column MG right (bold style) value).

Appendix D.2.1. Final remarks

Based on the likelihood calculation benchmark, there are several noteworthy observations formulated in the following paragraphs.

For $k = 1$, univariate implementations are considerably faster than multivariate implementations. In particular for $k = 1$ and $N \geq 1000$, the likelihood calculation times for the BM1, OU1 and MG1 implementations are nearly an order of magnitude faster (Table S13). This is because all complex $k \times k$ matrix operations (e.g. matrix multiplication and inversion) are reduced to univariate arithmetic operations (e.g. numeric multiplication and division).

For small N and big k, the relative cost of parameter transformation can be very high. In particular, notice that the total likelihood calculation time for the OU model in the cases of $k \geq 16$ and $N = 10$ is more than an order of magnitude higher than the corresponding BM time (values in bold style in Table S13). Conversely, the time excluding the parameter transformation for the OU model is only about two times bigger. In part, this is due to the fact that, in the current **PCMBase/PCMBaseCpp** implementation, the parameter transformation is done at the level of **PCMBase** (in `R`). In such cases, it can be beneficial to implement the parameter transformation at the level of the C++ backend.

The calculation time for the OU model grows asymptotically faster with k than the time for the BM model. Considering the times excluding the parameter transformation (values in normal style in Table S13), we notice that for $k = 2$ the time for the OU model is only about 1.16 times bigger than the BM time. Conversely, for $k = 32$ the OU time is already more than 2 times bigger than the BM time. This difference in growth rate with k is explained by the presence of a matrix exponentiation operation in the calculation of the terms $\vec{\omega}$ and $\mathbf{\Phi}$ for each edge (Eq. 18).

References

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.,

[1]For the constant α, we take the biggest of the constants associated with each model type.

1999. LAPACK Users' Guide. Society for Industrial and Applied Mathematics, Philadelphia, PA. third edition.

Bartoszek, K., 2011. Multivariate Aspects of Phylogenetic Comparative Methods. Licentiate Thesis, University of Gothenburg.

Blackford, L.S., Petitet, A., Pozo, R., Remington, K., Whaley, R.C., Demmel, J., Dongarra, J., Duff, I., Hammarling, S., Henry, G., et al., 2002. An updated set of basic linear algebra subprograms (blas). ACM Transactions on Mathematical Software 28, 135–151.

Clavel, J., Escarguel, G., Merceron, G., 2015. mvmorph: an R package for fitting multivariate evolutionary models to morphometric data. Methods in Ecology and Evolution 6, 1311–1319.

Cook, S.R., Gelman, A., Rubin, D.B., 2006. Validation of software for Bayesian models using posterior quantiles. Journal of Computational and Graphical Statistics 15, 675–692.

Eddelbuettel, D., 2013. Seamless R and C++ Integration with Rcpp. Springer Science & Business Media, New York, NY.

Golub, G.H., Van Loan, C.F., 2013. Matrix Computations. The Johns Hopkins University Press, Baltimore.

Mitov, V., Stadler, T., 2019. Parallel likelihood calculation for phylogenetic comparative models: The splitt c++ library. Methods in Ecology and Evolution 10, 493–506.

Revell, L.J., 2011. phytools: an R package for phylogenetic comparative biology (and other things). Methods in Ecology and Evolution 3, 217–223.

Sanderson, C., Curtin, R., 2016. Armadillo: a template-based C++ library for linear algebra. Journal of Open Source Software 1.