Reconstructing the evolution of developmental sequences in R: analysis of a small data set using pgi2

Luke B Harrison

August 7, 2023

Contents

---

1 Introduction

The PGi2 package implements the PGi algorithm described in Harrison and Larsson [1] for the analysis of the evolution of developmental sequences. Readers are referred to that article for a more in-depth discussion of the theoretical concerns. This vignette will focus on the application of pgi to the analysis of a small dataset of developmental sequences published by Velhagen (1997).

2 Basic Concepts

Conducting a PGi analysis in Rusing the pgi2 consists of several steps.
1. The raw data: 1) ranked developmental sequences and a 2) phylogenetic tree are read into R using the function `pgi.read.nexus()` to create an "empty" `pgi.tree` class data structure (or alternatively for advanced users, this can be contructed manually)
2. The PGi algorithm is executed multiple times with the `pgi()` function on the "empty" `pgi.tree` data structure to infer ancestral developmental sequences and sequence heterochronies. Consensus trees are produced for each run of the algorithm.
3. If more than one run was conducted (always recomended), a "superconsensus" of the individual consensus trees is performed with `pgi.supercon()`
4. The results are plotted using `plot.pgi.tree()`

3 Worked example: Velhagen's (1997) data set

3.1 Background

Velhagen [2] published a data set of developmental sequences of thamnophiine snakes. In this data set, there are 6 taxa, for which the sequence of ossification of 5 cranial bones is determined. A phylogeny is provided with the sequences. To be used in PGi, these sequences need to be specified in ranked developmental sequence format to yield a matrix of ranks for each bone. Note: developmental sequences may have simultaenity - that is events may occure simulatenously and thus a single sequence may have fewer ranks than events.

Velhagen reported the developmental seqeunce in terms of the order of the ossification of the bones: B = basioccipital; M = maxilla; N = nasal; Q = quadrate; S = supratemporal.

$Nerodiasipedon$:	$BMQS, N$
$Nerodiataxispilota$:	MS, B, N, Q
$Storeriadekayi$:	M, BS, Q, N
$Storeriaoccipitomaculata$:	M, B, NS, Q
$Thamnophisproximus$:	B, S, M, Q, N
$Thamnophisradix$:	S, BM, NQ

To simply exceution, PGi will store sequences as numerics starting with 1, i.e. 1 = B, 2 = M, 3 = N, 4 = Q, 5= S. Therefore the these raw sequences would be represented in PGi as:

$N.sipedon$:	$[1, 2, 4, 5], 3$
$N.taxispilota$:	$[2, 5], 1, 3, 4$
$S.dekayi$:	$[2, 5], 1, 4, 3$
$S.occipitomaculata$:	$2, 1, [3, 5], 4$
$T.proximus$:	$1, 5, 2, 4, 3$
$T.radix$:	$5, [1, 2], [3, 4]$

Internally, PGi also uses ranked developmental seqeunces and this is the input format. For Velhagen's data set, Translating these into ranked developmental sequence (with events in the order stated above) yields the following:

	B, M, N, Q, S
$N.sipedon$:	$1, 1, 2, 1, 1$
$N.taxispilota$:	$2, 1, 3, 4, 1$
$S.dekayi$:	$2, 1, 4, 3, 1$
$S.occipitomaculata$:	$2, 1, 3, 4, 3$
$T.proximus$:	$1, 3, 5, 4, 2$
$T.radix$:	$2, 2, 3, 3, 1$

Note: the pgi2 includes a utility function that can translate developmental sequences into ranked developmental sequences: `seq.to.rds()` and vice-versa `rds.to.seq()`, please refer to the man pages of those functions `?seq.to.rds` and `?rds.to.seq` for details.

A NEXUS formatted file of this data set (as ranked developmental seqeunces) and the accompanying phylogenetic tree is included in the PGi package, and is exerpted below as an example.

```
#NEXUS

BEGIN TAXA;
        TITLE Untitled_Taxa_Block;
        DIMENSIONS NTAX=6;
        TAXLABELS
                tproximus tradix sdekayi soccipitomaculata nsipedon ntaxispilota
        ;

END;


BEGIN CHARACTERS;
        TITLE  Untitled_Character_Matrix;
        DIMENSIONS  NCHAR=5;
        FORMAT DATATYPE = STANDARD GAP = - MISSING = ? SYMBOLS = "  0 1 2 3 4 5";
        MATRIX
        tproximus          13542
        tradix             22331
        sdekayi            21431
        soccipitomaculata  21343
        nsipedon           11211
        ntaxispilota       21341

;

END;

BEGIN TREES;
        Title Untitled_Tree_Block;
        LINK Taxa = Untitled_Taxa_Block;
        TRANSLATE
                1 tproximus,
                2 tradix,
                3 sdekayi,
                4 soccipitomaculata,
                5 nsipedon,
                6 ntaxispilota;
        TREE tree = ((4,3),(((1,2),5),6));

END;
```

3.2 Reading developmental sequence data into R

The NEXUS file is read by pgi.read.nexus(), alternatively, the data structure is also available in the PGi2 package using data(velhagen).

```
> library(pgi2)
> velhagen <- pgi.read.nexus("Velhagen1997.nex")
```

Note, please ensure the Velhagen1997.nex file is in the current R working directory or a full path name will need to be specified. Alternatively exceuting `pgi.read.nexus(interactive=TRUE)` will open an interactive dialogue to find the file. Now that the NEXUS file has been read into the velhagen `pgi.tree` object we can verify it was correctly read by using the `summary()` and `plot()` functions:

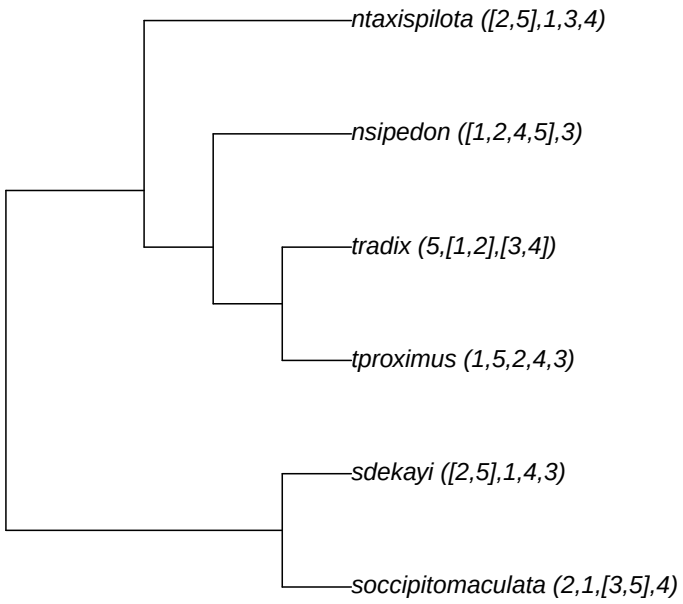
```
> summary(velhagen)
```

```
**PGi data object**
Contains a phylogenetic tree topology and sequence data
Number of taxa: 6
Number of nodes: 5
Sequence Length: 5

*Topology*
((soccipitomaculata,sdekayi),(((tproximus,tradix),nsipedon),ntaxispilota));

*Data* (First two sequences only, override with print.all.seqs=TRUE)
soccipitomaculata:     2,1,[3,5],4
sdekayi:               [2,5],1,4,3
```

```
> plot(velhagen, show.tip.seq = TRUE)
```



The velhagen data structure now contains the ranked developmental sequences and phylogenetic tree in an "empty" `pgi.tree` object. Further details on graphical options can be found in the manual for the `plot.pgi.tree()` function. Note: that output from the `summary.pgi.tree` and `plot.pgi.tree` functions display the sequences as RAW developmental sequences and not as ranked developmental sequences.

3.3 Executing the PGi algorithm

This sections assumes readers are familiar with the description of the PGi algorithm in Harrison and Larsson [1]. The pgi() function controls overall execution of the genetic algoirthm, and the

inference of ancestral developmental sequences and the generation of consensus trees of sequence heterochronies. Several parameters must be specified. First an "empty" pgi.tree must be first argument. Next, the number of runs and whether any are conducted in parallel using the R packages foreach and doMC is specified by nruns, where the first number is the number of runs in series and the second, the number of runs to be executed in parallel in each series run; thus the total number of runs is the product of these numbers, or 8 in the example below. Note PGi does not parallelize execution more than this. The inference parameters (for the genetic algorithm) are provided in the info.params list: here, 100 cycles of selection, exceuted 100 times, with 100 intermediate sequences retained at each internal node a specified. The edit cost function is `parsimov`, and the consensus parameters are as show:

```
> velhagen.con.trees <- pgi(velhagen, nruns = c(4, 2), inf.params = list(heuristic = "pgi",
+     cycles = 100, replicates = 100, ret.anc.seq = 100, edit.cost.func = "parsimov",
+     simultaneity = TRUE), con.params = list(con.type = "semi-exhaustive",
+     semi.ex.con.max.n = 5000, edit.cost.func = "parsimov"), verbosity = 1)

Overriding inf parameters in tree with the provided list
Overriding con parameters in tree with the provided list
Overiding nruns with the explicitely defined value
Executing PGi, with 8 independent computation[s], total
Running PGi in parallel mode: running 2 simultaenous computations a total of 4 time[s].
WARNING: if your computer has fewer than 2 processor cores, this will do more harm than good
[[1]]
[1] "Initializing parallel processing, core 1 of 2 \n"

[[2]]
[1] "Initializing parallel processing, core 2 of 2 \n"

Running inference with the pgi heuristic. Replicates/cycle=100 Cycles=100 and a retained matrix size of 100
Will display overall (per run) progress bar

  |
  |                                                                      |   0%
  |
  |==============                                                        |  20%
  |
  |============================                                          |  40%
  |
  |==========================================                            |  60%
  |
  |========================================================              |  80%
  |
  |======================================================================| 100%
Selected semi-exhaustive pseudoconsensus.

Semi Exhaustive consensus: maximum number of total per branch transversals = 5000

  |
  |                                                                      |   0%
  |
  |==================================                                    |  50%
  |
  |======================================================================| 100%
Completed cycle 1 out of 4  cycles in series
[[1]]
[1] "Initializing parallel processing, core 1 of 2 \n"

[[2]]
[1] "Initializing parallel processing, core 2 of 2 \n"

Running inference with the pgi heuristic. Replicates/cycle=100 Cycles=100 and a retained matrix size of 100
Will display overall (per run) progress bar

  |
  |                                                                      |   0%
  |
  |==============                                                        |  20%
  |
  |============================                                          |  40%
  |
  |==========================================                            |  60%
```

```
  |
  |=======================================================              |  80%
  |
  |======================================================================| 100%
Selected semi-exhaustive pseudoconsensus.

Semi Exhaustive consensus: maximum number of total per branch transversals = 5000

  |
  |                                                                      |   0%
  |
  |==================================                                    |  50%
  |
  |======================================================================| 100%
Completed cycle 2 out of 4  cycles in series
[[1]]
[1] "Initializing parallel processing, core 1 of 2 \n"

[[2]]
[1] "Initializing parallel processing, core 2 of 2 \n"

Running inference with the pgi heuristic. Replicates/cycle=100 Cycles=100 and a retained matrix size of 100
Will display overall (per run) progress bar

  |
  |                                                                      |   0%
  |
  |==============                                                        |  20%
  |
  |============================                                          |  40%
  |
  |==========================================                            |  60%
  |
  |=========================================================             |  80%
  |
  |======================================================================| 100%
Selected semi-exhaustive pseudoconsensus.

Semi Exhaustive consensus: maximum number of total per branch transversals = 5000

  |
  |                                                                      |   0%
  |
  |==================================                                    |  50%
  |
  |======================================================================| 100%
Completed cycle 3 out of 4  cycles in series
[[1]]
[1] "Initializing parallel processing, core 1 of 2 \n"

[[2]]
[1] "Initializing parallel processing, core 2 of 2 \n"

Running inference with the pgi heuristic. Replicates/cycle=100 Cycles=100 and a retained matrix size of 100
Will display overall (per run) progress bar

  |
  |                                                                      |   0%
  |
  |==============                                                        |  20%
  |
  |============================                                          |  40%
  |
  |==========================================                            |  60%
  |
  |=========================================================             |  80%
  |
  |======================================================================| 100%
Selected semi-exhaustive pseudoconsensus.

Semi Exhaustive consensus: maximum number of total per branch transversals = 5000

  |
```

```
  |                                                                      |   0%
  |
  |==================================                                    |  50%
  |
  |======================================================================| 100%
Completed cycle 4 out of 4  cycles in series

Results
Minimum tree length = 8, mean tree length across replicates = 8
Executed cleanly
```

3.4 Examining the initial results

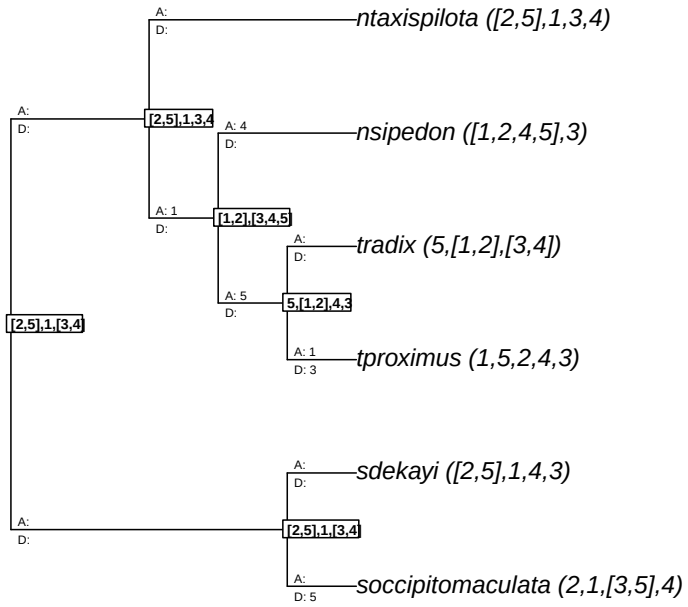
The results of the 8 indepedent runs are stored in a `multi.pgi.tree` data object, this is essentially an R `list` of `pgi.tree` objects. Generic methods `summary()` and `plot` are available for the `multi.pgi.tree` class.

```
> summary(velhagen.con.trees)

8  PGi Data objects
Status of the objects:  consensus,consensus,consensus,consensus,consensus,consensus,consensus,consensus
Tree lengths recorded in the PGi Objects:  8,8,8,8,8,8,8,8
```

Execution has yielded 8 consesus trees, each with tree lengths of 8 sequence heterochronies. Individual trees and estimated sequence heterochronies, and ancestral sequences can be visualized either in a single figure by using `plot(velhagen.con.trees)` or invidiually using `plot(velhagen.con.trees[[N]])`.

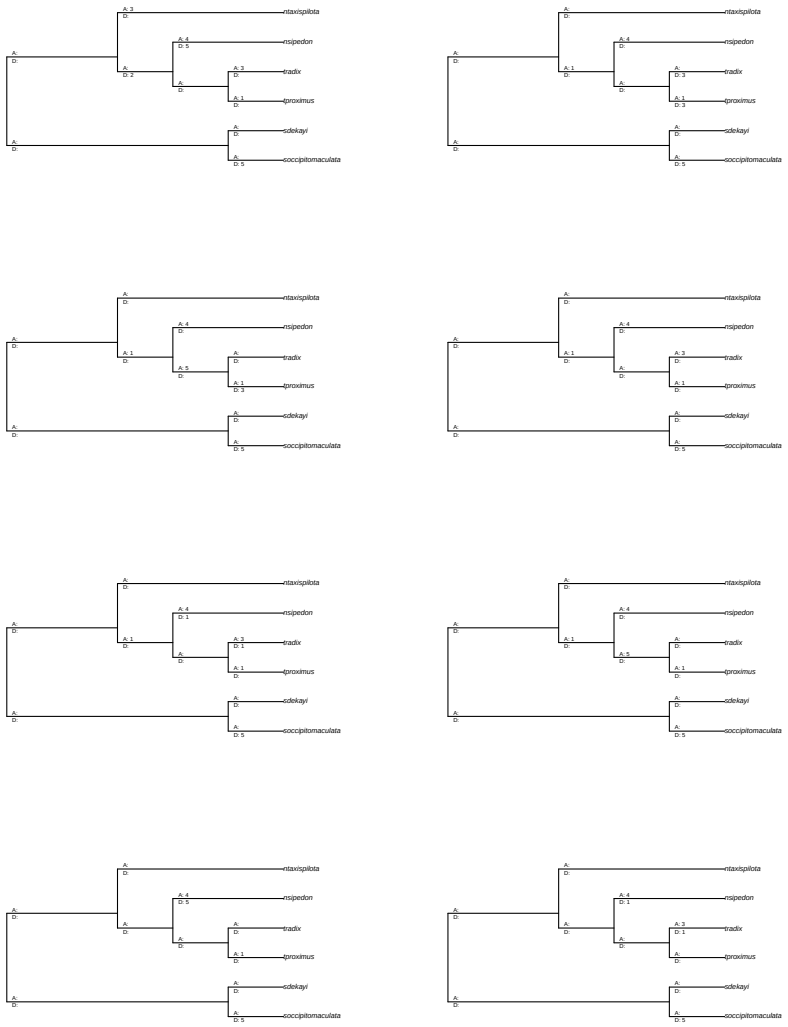
```
> plot(velhagen.con.trees[[3]], show.tip.seq = TRUE, show.anc.seq = TRUE)
```



For example, this is the consensus solution from the 3rd of 8 independent PGi runs, showing ancestral sequences, sequence heterochronies, and tip sequences.

```
> plot(velhagen.con.trees, show.tip.seq = FALSE, show.anc.seq = FALSE)

Creating multi.pgi.tree plot using a plotting matrix of x = 4 and y = 2
```



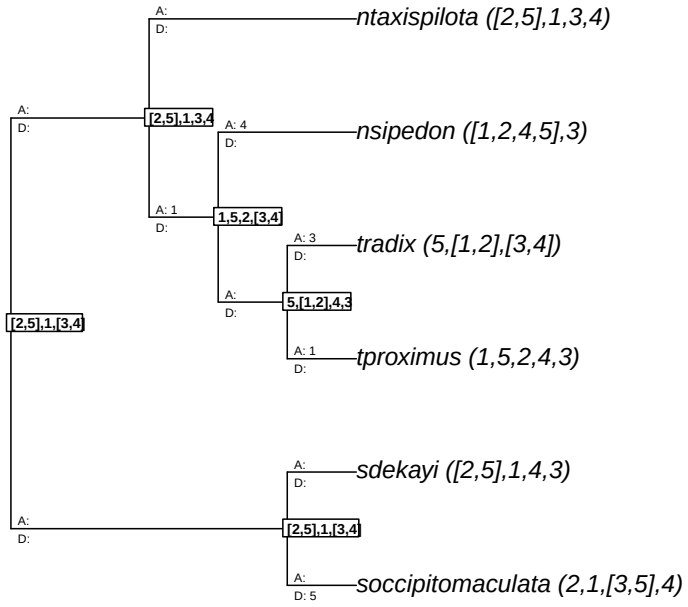
This figure, using the `plot()` method on the `multi.pgi.tree` object yields a multi panel figure of all 8 indepdent runs

3.5 Generating a superconsensus

The invidiual consensus results can be summarized using a superconsensus tree. This tree is generated using the `pgi.supercon` function, which expects a `multi.pgi.tree` data object with at least 2 conensus trees and will output a `pgi.tree` data object of the type "superconsensus". By default, the `pgi.supercon` function will create a superconsensus of all consensus solutions of the shortest tree length (in terms of sequence heterochronies). This behaviours can be modified using the `tol` parameter, see `?pgi.supercon` for more details.

```
> velhagen.supercon <- pgi.supercon(velhagen.con.trees, tol = 0,
+     verbosity = 1)
Calculating superconsensus using those trees with a tree length <=  8
```

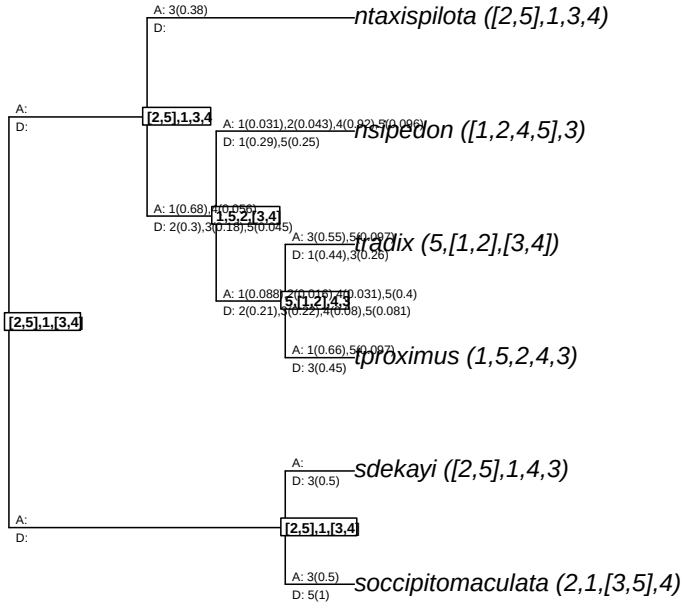
```
> plot(velhagen.supercon, show.tip.seq = TRUE, show.anc.seq = TRUE)
```



The above figure represents the final output of the PGi algorithm for this data set. Further output options are available in the plotting function, see `?plot.pgi.tree` for more details. For example, support values for each heterochrony, which represent the fraction of total solutions containing that specific sequence heterochrony can be visualized, with the `print.support=TRUE` option. By default, only sequence heterochronies present in at least 50% of solutions examined will be printed, but this behaviour can be adjusted using the parameter `seq.het.thres`. In the following example, all sequence heterochronies are printed with their support values and all estimated consensus ancestral sequences are also printed along with the raw developmental sequences.

```
> plot(velhagen.supercon, seq.het.thres = 0, print.support = TRUE,
+     show.anc.seq = TRUE, show.tip.seq = TRUE)

Sequence heterochrony threshold is 0, automatically printing support values
```



Note: that PGi uses the excellent `ape::plot.phylo` to actually do the figure plotting. There is a limit to what can be automatically specified for readability of the figure and some manual editting will likely be required. Additional parameters to `plot.pgi()` will automatically be passed to `ape::plot.phylo()`

References

[1] Luke B Harrison and Hans CE Larsson. Estimating evolution of temporal sequence changes: A practical approach to inferring ancestral developmental sequences and sequence heterochrony. *Systematic Biology*, 57:378–387, 2008.

[2] William A Velhagen. Analyzing developmental sequences using sequence units. *Systematic Biology*, 46(1):204–210, 1997.