SUNPLIN USER MANUAL

SUNPLIN (Simulation with Uncertainty for Phylogenetic Investigations) expands trees by inserting the provided phylogenetic uncertain taxa (PUT) randomly into the partially known tree and calculates the corresponding patristic distance matrices. The results can be easily imported into a statistical software and used to conduct large scale phylogenetic comparative simulations that take uncertainty into account.

Sunplin can be used as a standalone program, a shared object in the R system environment, or as a web service. The standalone version is the option of choice for large scale simulations since it has the best performance and outputs all the data directly into local files. The shared object in R can also output the resulting trees in a file, but it produces the distance matrices one at a time in order to avoid spending too much memory. Finally, the web service is a more user-friend option, requiring no compilation and with easy access trough the web. However, the data produced can be large (especially the distance matrices) and the download time can take long. The installation and the usage of these version of Sunplin are described next.

1. Standalone version

The standalone version requires a C++ compiler. If the g++ compiler is available, for example, the following command can be use to compile the source-code.

```
> g++ sunplin.cpp -o sunplin
```

This generates the executable file named sunplin that can be invoked as:

```
sunplin ( expd <f1> <f2> <n> <m> | dist <f3> ) [-l <f4>]
```

The vertical bar indicates that sunplin can be used to randomly expand a tree (expd) or to calculate distance matrices (dist).

To expand a tree, a newick formatted tree (f1) and the species to be inserted (f2 - plain text) must be provided. In addition, the number of trees to generate (n) and the expansion method (m - (1) node-based, (2) branch-based) are required. The file f2 must contain pairs <PUT> <MDCC>, one per line, where PUT is the phylogenetic uncertain taxa and MDCC is the most derived consensus clade from which point the PUT will be randomly inserted.

To calculate the distance matrix, a newick formatted tree (f3) must be provided. If a nexus formatted file (containing several trees) is used, all corresponding distance matrices are calculated.

In both cases, the output file name can be specified (f4). If omitted, default names are used.

Examples:

```
./sunplin expd Hum_146_spp.tree Hum_158_spp.puts 1000 1
```

Expands the Hum_146_spp.tree provided by inserting the species specified in Hum_158_spp.puts. The process is repeated 1000 times, using the node-based method, and the resulting trees are stored in a nexus formatted file named Hum_158_spp-expd.nex

```
./sunplin expd Hum_146_spp.tree Hum_158_spp.puts 100 2 -l Hum_304-spp
```

Expands the Hum_146_spp.tree provided by inserting the species specified in Hum_158_spp.puts. The process is repeated 100 times, using the branch-based method, and the resulting trees are stored in a file named Hum_304-spp.nex

```
./sunplin dist Hum_146_spp.tree
```

Calculates the distance matrix of the Hum_146_spp.tree provided. The resulting matrix is stored in a file named Hum_146_spp-dist.xls

```
./sunplin dist Hum_304-spp.nex -l Hum_304-spp_dist
```

Calculates the distance matrices of all trees provided in Hum_304-spp.nex. The resulting matrices are stored in a file named Hum_304-spp_dist.xls.

2. R shared object version

In order to be able to use Sunplin in the R system environment, first an R shared object must be created using the following command, that compiles the C++ source-code into an R loadable library. Note: Windows users must have Rtools (http://cran.r-project.org/bin/windows/Rtools/) installed.

```
> R CMD SHLIB sunplin-r.cpp --output=sunplin.spn
```

Then, start up R from the terminal and copy the command below.

```
dyn.load("sunplin.spn")
```

Once the library has been successfully loaded, the following functions can be used to expand trees and calculate distance matrices respectively.

```
.Call("expd", "<f1>", "<f2>", <n>, <m>)

.Call("dist", "<f3>|<t1>", “tree”|”file”)
```

To expand a tree, a newick formatted tree (f1) and the species to be inserted (f2 - plain text) must be provided. In addition, the number of trees to generate (n) and the expansion method (m - (1) node-based, (2) branch-based) are required. The file f2 must contain pairs <PUT> <MDCC>, one per line, where PUT is the phylogenetic uncertain taxa and MDCC is the most derived consensus clade from which point the PUT will be randomly inserted

To calculate the distance matrix, a newick formatted tree (f3) or a tree (t1) - 'phylo' type - must be provided. The third parameter indicates whether f3 or t1 is used.

Instead of calling the .Call functions directly, we created wrapper functions that allows the user to call the C++ code with minimal hassle. These functions, along with supporting functions are stored in the file "sunplin-functions.r", and described at the end of this document. Thus the following functions can be used to expand trees and to calculate distance matrices respectively.

```
sunplin.expd("<f1>", "<f2>", <n>, <m>)

sunplin.dist("<f3>|<t1>", tree|file)
```

The main difference is that the first parameter was hidden and it is now part of the function name.

Examples:

Before trying out the examples below, the Sunplin functions must be loaded with the following command, that also loads the Sunplin library (sunplin.spn) and the ape package.

```
source("sunplin-functions.r")

trees <- sunplin.expd("Hum_146_spp.tree","Hum_158_spp.puts", 100, 2)
tree10 <- sunplin.n.tree(trees, 10, "tree")
tree10nk <- sunplin.tree2newick( tree10 )
```

Expands the Hum_146_spp.tree provided by inserting the species specified in Hum_158_spp.puts. The process is repeated 100 times, using the branch-based method, and the resulting trees are stored in an object of class "multiPhylo" from the ape package.

```
mat1 <- sunplin.dist("Hum_146_spp.tree", "file")
```

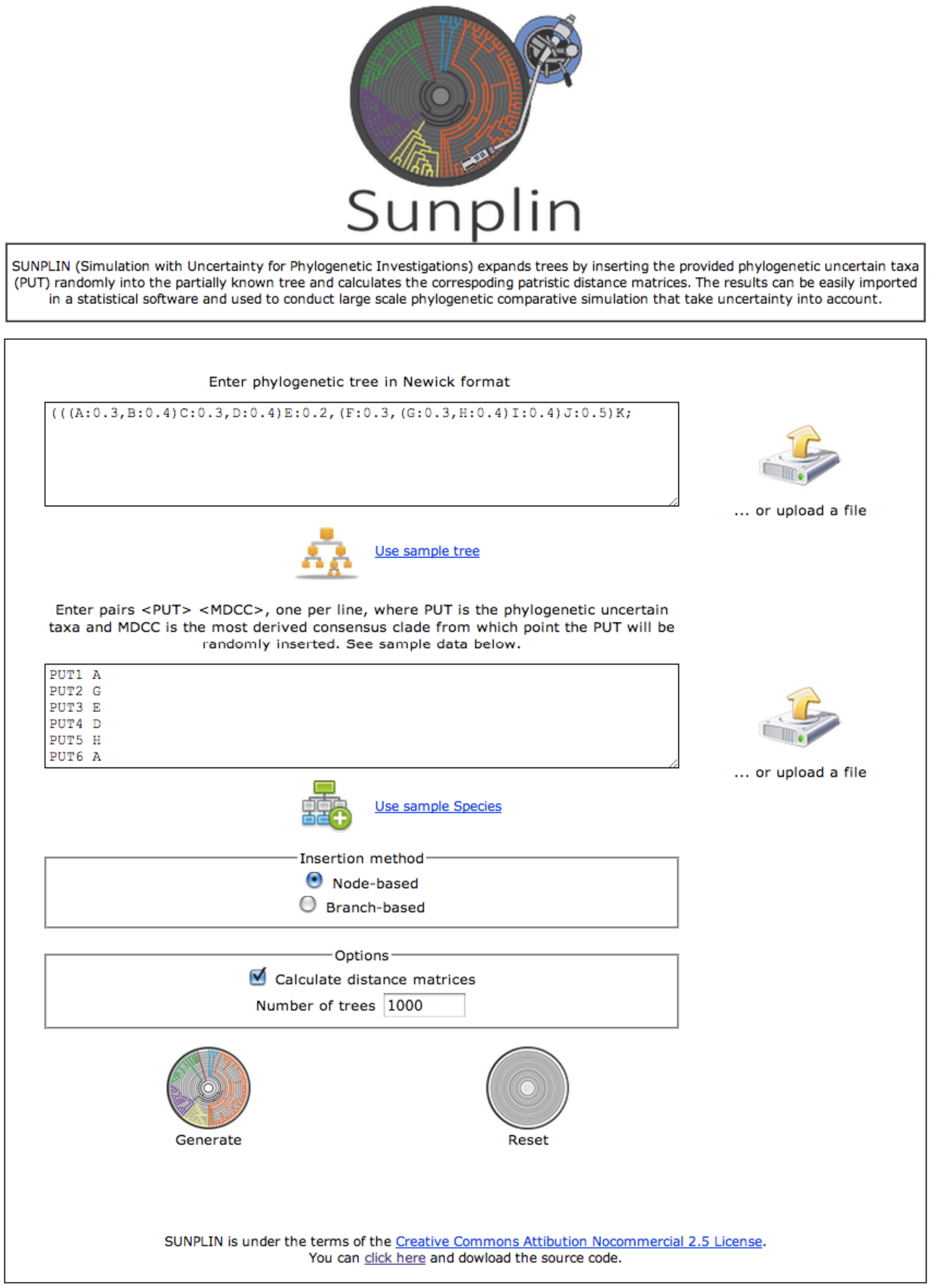
Calculates the distance matrix of the tree provided in the file Hum_146_spp.tree. The resulting matrix mat1 is an object of class "matrix".

```
mat10 <- sunplin.dist(tree10nk, "tree" )
```

Calculates the distance matrix of the tree provided in the character object tree10nk (tree in newick format). The resulting matrix mat10 is an object of class "matrix".

3. Web service version

Users can use Sunplin from any machine with a web browser, without having to compile and install the program. The Sunplin web service provides an easy to use interface and can be accessed at purl.oclc.org/NET/sunplin/ . The main page of the site is shown in figure 1.



SUNPLIN (Simulation with Uncertainty for Phylogenetic Investigations) expands trees by inserting the provided phylogenetic uncertain taxa (PUT) randomly into the partially known tree and calculates the correspoding patristic distance matrices. The results can be easily imported in a statistical software and used to conduct large scale phylogenetic comparative simulation that take uncertainty into account.

Enter phylogenetic tree in Newick format

```
(((A:0.3,B:0.4)C:0.3,D:0.4)E:0.2,(F:0.3,(G:0.3,H:0.4)I:0.4)J:0.5)K;
```

... or upload a file

Use sample tree

Enter pairs <PUT> <MDCC>, one per line, where PUT is the phylogenetic uncertain taxa and MDCC is the most derived consensus clade from which point the PUT will be randomly inserted. See sample data below.

```
PUT1 A
PUT2 G
PUT3 E
PUT4 D
PUT5 H
PUT6 A
```

... or upload a file

Use sample Species

Insertion method

● Node-based

○ Branch-based

Options

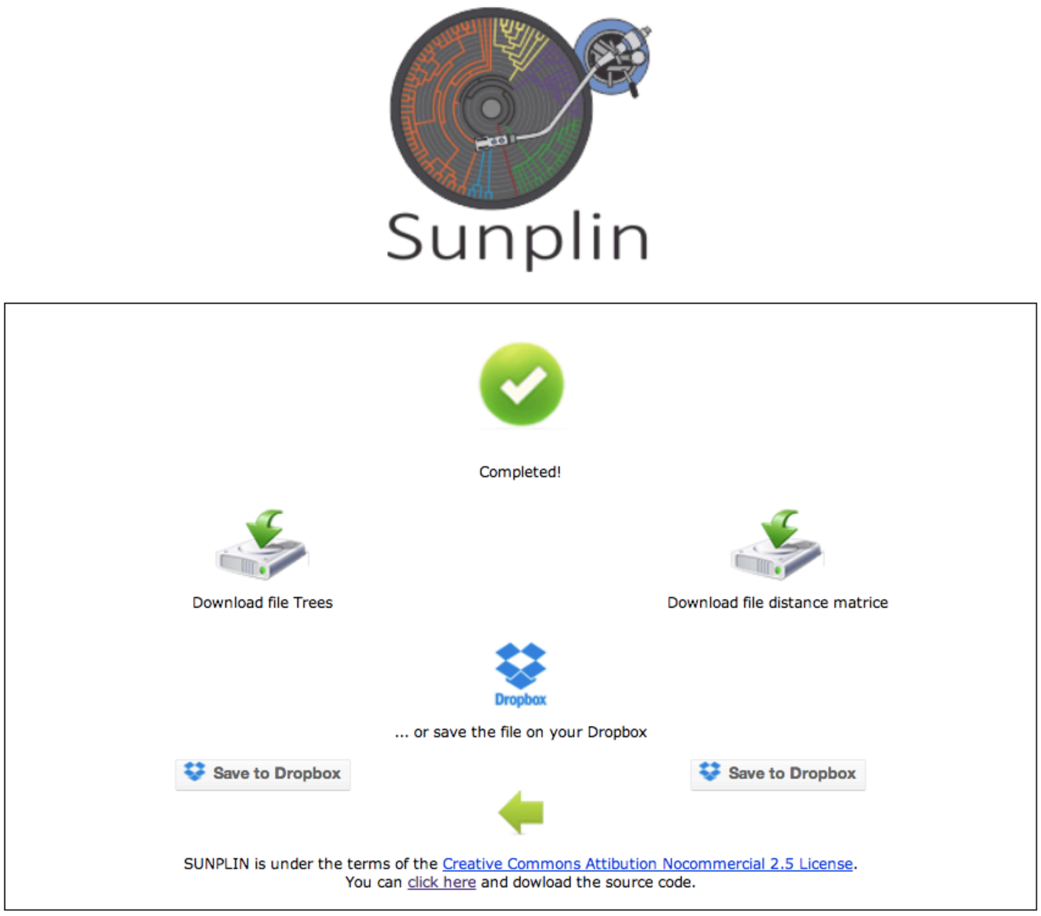
✓ Calculate distance matrices

Number of trees 1000

SUNPLIN is under the terms of the Creative Commons Attibution Nocommercial 2.5 License.
You can click here and dowload the source code.

Figure 1 – Main page of the Sunplin web site

The site allows the user to enter a tree (copying and pasting or through a file) or to use an example containing a small tree coded in the Newick format. Next, a file with phylogenetic uncertain taxa (PUT) data, or the provided sample data, have to be loaded. This information must be provided in pairs <PUT> <MDCC>, one per line. Each PUT must be followed by a space and an MDCC, that is, the most derived consensus clade from which point the PUT will be randomly inserted. The user has to choose the insertion method (node-based or branch-based), to inform the number of expanded trees, and to select whether to compute the distance matrices or not. Figure 2 shows the results page after the information shown in Figure 1 has been submitted.



Completed!

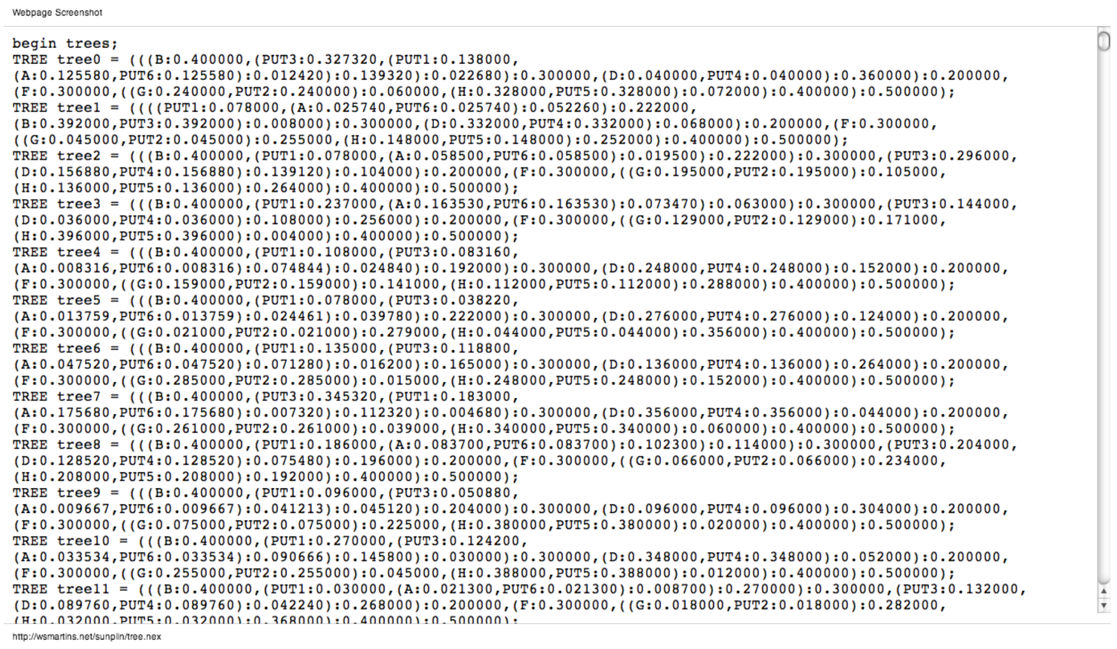
... or save the file on your Dropbox

SUNPLIN is under the terms of the Creative Commons Attibution Nocommercial 2.5 License.
You can click here and dowload the source code.

Figure 2 – Results page after submission

The user can download both the expanded trees generated (Nexus format) and the distance matrices calculated (Excel's xls format). Since these files (especially the distance matrices) can be large, the site provides the option of saving the files directly into a Dropbox folder. This option requires a Dropbox account and the login information. Figures 3 and 4 show fragments of the files produced by the website.

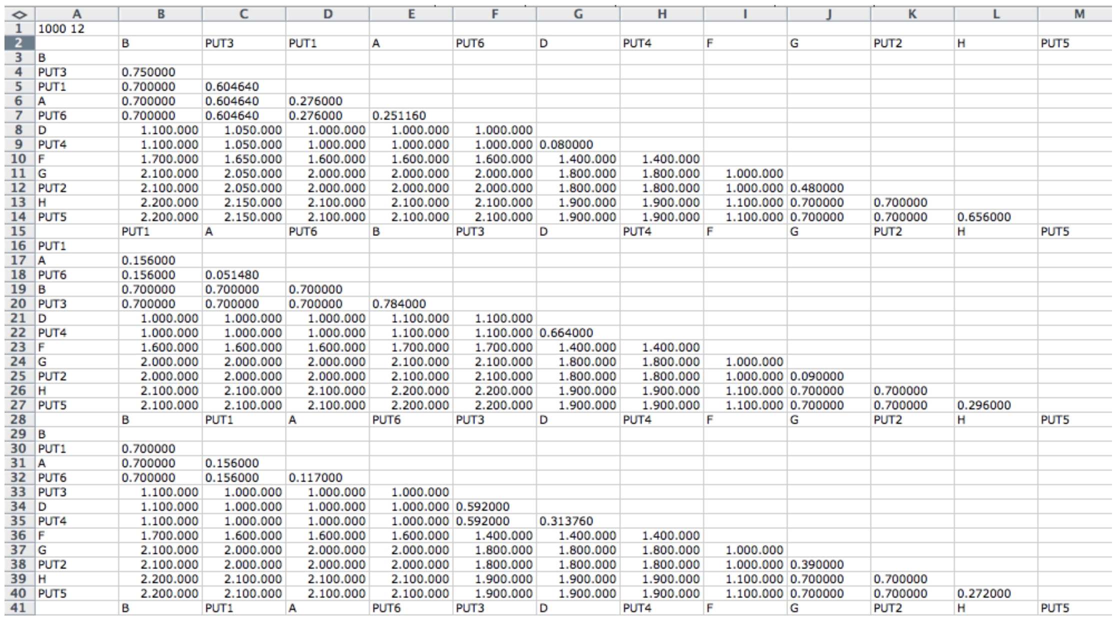
The R functions included in the file "sunplin-functions.r" (see descriptions at the end of this document), can be used to easily import the expanded trees and distance matrices into the R system.


Webpage Screenshot

```
begin trees;
TREE tree0 = (((B:0.400000,(PUT3:0.327320,(PUT1:0.138000,
(A:0.125580,PUT6:0.125580):0.012420):0.139320):0.022680):0.300000,(D:0.040000,PUT4:0.040000):0.360000):0.200000,
(F:0.300000,((G:0.240000,PUT2:0.240000):0.060000,(H:0.328000,PUT5:0.328000):0.072000):0.400000):0.500000);
TREE tree1 = ((((PUT1:0.078000,(A:0.025740,PUT6:0.025740):0.052260):0.222000,
(B:0.392000,PUT3:0.392000):0.008000):0.300000,(D:0.332000,PUT4:0.332000):0.068000):0.200000,(F:0.300000,
((G:0.045000,PUT2:0.045000):0.255000,(H:0.148000,PUT5:0.148000):0.252000):0.400000):0.500000);
TREE tree2 = (((B:0.400000,(PUT1:0.078000,(A:0.058500,PUT6:0.058500):0.019500):0.222000):0.300000,(PUT3:0.296000,
(D:0.156880,PUT4:0.156880):0.139120):0.104000):0.200000,(F:0.300000,((G:0.195000,PUT2:0.195000):0.105000,
(H:0.136000,PUT5:0.136000):0.264000):0.400000):0.500000);
TREE tree3 = (((B:0.400000,(PUT1:0.237000,(A:0.163530,PUT6:0.163530):0.073470):0.063000):0.300000,(PUT3:0.144000,
(D:0.036000,PUT4:0.036000):0.108000):0.256000):0.200000,(F:0.300000,((G:0.129000,PUT2:0.129000):0.171000,
(H:0.396000,PUT5:0.396000):0.004000):0.400000):0.500000);
TREE tree4 = (((B:0.400000,(PUT1:0.108000,(PUT3:0.083160,
(A:0.008316,PUT6:0.008316):0.074844):0.024840):0.192000):0.300000,(D:0.248000,PUT4:0.248000):0.152000):0.200000,
(F:0.300000,((G:0.159000,PUT2:0.159000):0.141000,(H:0.112000,PUT5:0.112000):0.288000):0.400000):0.500000);
TREE tree5 = (((B:0.400000,(PUT1:0.078000,(PUT3:0.038220,
(A:0.013759,PUT6:0.013759):0.024461):0.039780):0.222000):0.300000,(D:0.276000,PUT4:0.276000):0.124000):0.200000,
(F:0.300000,((G:0.021000,PUT2:0.021000):0.279000,(H:0.044000,PUT5:0.044000):0.356000):0.400000):0.500000);
TREE tree6 = (((B:0.400000,(PUT1:0.135000,(PUT3:0.118800,
(A:0.047520,PUT6:0.047520):0.071280):0.016200):0.165000):0.300000,(D:0.136000,PUT4:0.136000):0.264000):0.200000,
(F:0.300000,((G:0.285000,PUT2:0.285000):0.015000,(H:0.248000,PUT5:0.248000):0.152000):0.400000):0.500000);
TREE tree7 = (((B:0.400000,(PUT3:0.345320,(PUT1:0.183000,
(A:0.175680,PUT6:0.175680):0.007320):0.112320):0.004680):0.300000,(D:0.356000,PUT4:0.356000):0.044000):0.200000,
(F:0.300000,((G:0.261000,PUT2:0.261000):0.039000,(H:0.340000,PUT5:0.340000):0.060000):0.400000):0.500000);
TREE tree8 = (((B:0.400000,(PUT1:0.186000,(A:0.083700,PUT6:0.083700):0.102300):0.114000):0.300000,(PUT3:0.204000,
(D:0.128520,PUT4:0.128520):0.075480):0.196000):0.200000,(F:0.300000,((G:0.066000,PUT2:0.066000):0.234000,
(H:0.208000,PUT5:0.208000):0.192000):0.400000):0.500000);
TREE tree9 = (((B:0.400000,(PUT1:0.096000,(PUT3:0.050880,
(A:0.009667,PUT6:0.009667):0.041213):0.045120):0.204000):0.300000,(D:0.096000,PUT4:0.096000):0.304000):0.200000,
(F:0.300000,((G:0.075000,PUT2:0.075000):0.225000,(H:0.380000,PUT5:0.380000):0.020000):0.400000):0.500000);
TREE tree10 = (((B:0.400000,(PUT1:0.270000,(PUT3:0.124200,
(A:0.033534,PUT6:0.033534):0.090666):0.145800):0.030000):0.300000,(D:0.348000,PUT4:0.348000):0.052000):0.200000,
(F:0.300000,((G:0.255000,PUT2:0.255000):0.045000,(H:0.388000,PUT5:0.388000):0.012000):0.400000):0.500000);
TREE tree11 = (((B:0.400000,(PUT1:0.030000,(A:0.021300,PUT6:0.021300):0.008700):0.270000):0.300000,(PUT3:0.132000,
(D:0.089760,PUT4:0.089760):0.042240):0.268000):0.200000,(F:0.300000,((G:0.018000,PUT2:0.018000):0.282000,
(H:0.032000,PUT5:0.032000):0.368000):0.400000):0.500000);
```

http://wsmartins.net/sunplin/tree.nex

Figure 3 – Fragment of the file containing the expanded trees



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	1000 12												
2		B	PUT3	PUT1	A	PUT6	D	PUT4	F	G	PUT2	H	PUT5
3	B												
4	PUT3	0.750000											
5	PUT1	0.700000	0.604640										
6	A	0.700000	0.604640	0.276000									
7	PUT6	0.700000	0.604640	0.276000	0.251160								
8	D	1.100.000	1.050.000	1.000.000	1.000.000	1.000.000							
9	PUT4	1.100.000	1.050.000	1.000.000	1.000.000	1.000.000	0.080000						
10	F	1.700.000	1.650.000	1.600.000	1.600.000	1.600.000	1.400.000	1.400.000					
11	G	2.100.000	2.050.000	2.000.000	2.000.000	2.000.000	1.800.000	1.800.000	1.000.000				
12	PUT2	2.100.000	2.050.000	2.000.000	2.000.000	2.000.000	1.800.000	1.800.000	1.000.000	0.480000			
13	H	2.200.000	2.150.000	2.100.000	2.100.000	2.100.000	1.900.000	1.900.000	1.100.000	0.700000	0.700000		
14	PUT5	2.200.000	2.150.000	2.100.000	2.100.000	2.100.000	1.900.000	1.900.000	1.100.000	0.700000	0.700000	0.656000	
15		PUT1	A	PUT6	B	PUT3	D	PUT4	F	G	PUT2	H	PUT5
16	PUT1												
17	A	0.156000											
18	PUT6	0.156000	0.051480										
19	B	0.700000	0.700000	0.700000									
20	PUT3	0.700000	0.700000	0.700000	0.784000								
21	D	1.000.000	1.000.000	1.000.000	1.100.000	1.100.000							
22	PUT4	1.000.000	1.000.000	1.000.000	1.100.000	1.100.000	0.664000						
23	F	1.600.000	1.600.000	1.600.000	1.700.000	1.700.000	1.400.000	1.400.000					
24	G	2.000.000	2.000.000	2.000.000	2.100.000	2.100.000	1.800.000	1.800.000	1.000.000				
25	PUT2	2.000.000	2.000.000	2.000.000	2.100.000	2.100.000	1.800.000	1.800.000	1.000.000	0.090000			
26	H	2.100.000	2.100.000	2.100.000	2.200.000	2.200.000	1.900.000	1.900.000	1.100.000	0.700000	0.700000		
27	PUT5	2.100.000	2.100.000	2.100.000	2.200.000	2.200.000	1.900.000	1.900.000	1.100.000	0.700000	0.700000	0.296000	
28		B	PUT1	A	PUT6	PUT3	D	PUT4	F	G	PUT2	H	PUT5
29	B												
30	PUT1	0.700000											
31	A	0.700000	0.156000										
32	PUT6	0.700000	0.156000	0.117000									
33	PUT3	1.100.000	1.000.000	1.000.000	1.000.000								
34	D	1.100.000	1.000.000	1.000.000	1.000.000	0.592000							
35	PUT4	1.100.000	1.000.000	1.000.000	1.000.000	0.592000	0.313760						
36	F	1.700.000	1.600.000	1.600.000	1.600.000	1.400.000	1.400.000	1.400.000					
37	G	2.100.000	2.000.000	2.000.000	2.000.000	1.800.000	1.800.000	1.800.000	1.000.000				
38	PUT2	2.100.000	2.000.000	2.000.000	2.000.000	1.800.000	1.800.000	1.800.000	1.000.000	0.390000			
39	H	2.200.000	2.100.000	2.100.000	2.100.000	1.900.000	1.900.000	1.900.000	1.100.000	0.700000	0.700000		
40	PUT5	2.200.000	2.100.000	2.100.000	2.100.000	1.900.000	1.900.000	1.900.000	1.100.000	0.700000	0.700000	0.272000	
41		B	PUT1	A	PUT6	PUT3	D	PUT4	F	G	PUT2	H	PUT5

Figure 4 – Fragment of the file containing the distance matrices

Examples:

Before trying out the examples below, the Sunplin functions must be loaded with the following command,that also loads the Sunplin library (sunplin.spn) and the ape package.

```
source("sunplin-functions.r")
```

Start up R from the directory (folder) containing the files downloaded from the site.

```
mytrees <- sunplin.read.nexus("tree.nex")
```

Get all trees of file “tree.nex” and store into a list of trees (multiPhylo object) named mytrees. Individual trees can be accessed by using mytrees[[i]].

```
tree10 <- sunplin.n.tree(“tree.nex”, 10, “file”)
```

Get the tenth tree of file “tree.nex” and store into tree10 (phylo object).

```
mat10 <- sunplin.n.mat( "dist.xls", 10 )
```

Get the tenth matrix and store into mat10 (matrix object)

4. Supporting R function

The R functions bellow are defined in the sunplin-functions.r file and can be used to call Sunplin from within the R environment and to import Sunplin results into the R environment. Use source("sunplin-functions.r") to load these function into the R environment.

sunplin.read.nexus (localFile)
Reads localFile and creates a multiPhylo object (list of trees) containing its trees.

sunplin.tree2newick (Tree)
Transforms a phylo object Tree into a character object containg Tree in newick format.

sunplin.n.tree (fileOrText, nth, typeInput)
Reads a list of trees from fileOrText and returns the nth tree. If typeInput is "tree", fileOrText is a multiPhylo object, otherwise it is a file.

sunplin.n.mat (fileName, nth)
Reads a list of distance matrices from fileName and returns the nth distance matrix.

sunplin.expd (fileTree, filePuts, numTree, method)
Reads a tree from fileTree and returns numTree expanded trees using the specified insertion method. filePuts must contain pairs <PUT> <MDCC>, one per line, where PUT is the phylogenetic uncertain taxa and MDCC is the most derived consensus clade from which point the PUT will be randomly inserted.

sunplin.dist (fileOrText, typeInput)
Reads a tree from fileOrText and returns its distance matrix. If typeInput is "tree", fileOrText is a phylo object. If typeInput is "file", fileOrText is a newick file.

Contents of the sunplin-functions.r file:

```
library(ape)

sunplin.read.nexus <- function ( localFile ) {
	return ( read.nexus(localFile) );
}

sunplin.tree2newick <- function( Tree ){
	return (write.tree(Tree))
}

sunplin.n.tree <- function ( fileOrText, nth, typeInput ){
	if( typeInput == "tree" ){
		write(fileOrText, "temp", sep = "\n")
		fileOrText = "temp"
	}
	ret <- strsplit(scan(fileOrText, what = character(), skip = nth, nlines = 1 ), " ")[[4]]
	return (read.tree(text=ret))
}

sunplin.n.mat <- function ( fileName, nth ){
	ret <- scan(fileName, what = character(), nlines = 1)
	if( is.numeric(ret[1]) == TRUE ) qtdNodes <- as.numeric(ret[1])
	else qtdNodes <- as.numeric(ret[2])
	ret <- scan(fileName, what = character(), sep = '\n', skip = (((nth-1)*(qtdNodes+1) + 1)),
nlines =(qtdNodes) + 1)
	ret <- strsplit(ret,"\t")
	mat <- round(lower.tri(matrix(1,qtdNodes+1,qtdNodes+1)))
	for(i in 1 : qtdNodes+1 )
		mat[1,i] <- ret[[1]][i]
	for(i in 2 : qtdNodes+1){
		for(j in 1 : i-1 ){
			mat[i,j] <- ret[[i]][j]
		}
	}
	return (mat)
}

sunplin.expd <- function( fileTree, filePuts, numTree, method ){
	dyn.load("sunplin.spn")
	return (.Call("expd", fileTree, filePuts, numTree, method))
}

sunplin.dist <- function( fileOrText, typeInput ){
	dyn.load("sunplin.spn")
	return (.Call("dist", fileOrText, typeInput))
}
```