RANGER-DTL 2.0 Manual

https://compbio.engr.uconn.edu/software/RANGER-DTL/

1. Overview

RANGER-DTL (short for Rapid ANalysis of Gene family Evolution using Reconciliation-DTL) is a software package for inferring gene family evolution by speciation, gene duplication, horizontal gene transfer, and gene loss. The software takes as input a gene tree (rooted or unrooted) and a rooted species tree and reconciles the two by postulating speciation, duplication, transfer, and loss events. RANGER-DTL 2.0 implements many new algorithms that vastly improve upon the capability and functionality of the previous version of RANGER-DTL (version 1, available from http://compbio.mit.edu/ranger-dtl/). Some new capabilities of RANGER-DTL 2.0 include (i) principled handling of unrooted gene trees by considering all possible optimal rootings, instead of just a single arbitrarily chosen optimal rooting, (ii) uniformly random sampling of the space of all optimal reconciliations, making it possible to compute multiple optimal reconciliations and account for the variability in optimal reconciliation scenarios, (iii) handling gene tree uncertainty by collapsing weakly supported gene tree edges and computing and considering all optimal resolutions of the gene tree, and (iv) computing support values for individual DTL event inferences and species mapping assignments while accounting for multiple optimal reconciliations, uncertainty in gene tree rooting, alternative event cost assignments, and even gene tree topological uncertainty. RANGER-DTL 2.0 can efficiently analyze trees with many hundreds or even thousands of taxa, can handle both undated and dated species trees (i.e., both cladograms and chronograms), and allows for the use of distance-dependent transfer costs. Further technical details on the reconciliation model and algorithms used by RANGER-DTL are available in the following papers:

- *Efficient Algorithms for the Reconciliation Problem with Gene Duplication, Horizontal Transfer, and Loss*
  Mukul S. Bansal, Eric J. Alm, and Manolis Kellis.
  ISMB 2012; Bioinformatics 28: i283-i291, 2012.
- *Reconciliation Revisited: Handling Multiple Optima when Reconciling with Duplication, Transfer, and Loss*
  Mukul S. Bansal, Eric J. Alm, and Manolis Kellis.
  Journal of Computational Biology (JCB), 20(10): 738-754, 2013.
  (A preliminary version of this paper appeared in RECOMB 2013.)

- *Exact Algorithms for Duplication-Transfer-Loss Reconciliation with Non-Binary Gene Trees*
  Misagh Kordi and Mukul S. Bansal.
  IEEE/ACM Transactions on Computational Biology and Bioinformatics, 16(4): 1077-1090, 2019.
  (A preliminary version of this paper appeared in ACM-BCB 2016.)
- *On the impact of uncertain gene tree rooting on duplication-transfer-loss reconciliation*
  Soumya Kundu and Mukul S. Bansal.
  BMC Bioinformatics, 19(Suppl 9): 290, 2018.

2. Description of the Software

The RANGER-DTL 2.0 software package consists of 3 core programs that define the core functionality of RANGER-DTL and 5 supplemental programs that provide additional functionality and capability. In addition, we also provide 2 summary scripts to help summarize results across alternative rootings and alternative gene tree topologies and compute support values for the reconciliation. We briefly describe these programs below:

- **Core programs**
  1. *OptRoot*: Takes as input a rooted species tree and an unrooted gene tree and computes all optimal rootings (those that minimize the DTL reconciliation cost) for the gene tree.
  2. *Ranger-DTL*: Takes as input a rooted species tree and rooted gene tree and computes an optimal DTL reconciliation for the gene tree and species tree. In case of multiple optimal reconciliations, the output is an optimal reconciliation sampled uniformly at random among all optimal reconciliations. This program can (and should) be run multiple times on the same species tree and gene tree pair to sample the space of all optimal reconciliations.
  3. *AggregateRanger*: Takes as input a series of output reconciliation files computed using Ranger-DTL on the same gene tree and species tree. The files must be created by executing *Ranger-DTL* (or its *fast* and *dated* variants; see supplementary programs below) on the same input gene tree and species tree multiple times with either the same or different event cost assignments. E.g., One may generate 100 samples each using three different event cost settings, for a total of 300 reconciliation files. These 300 reconciliation files would then be the input for *AggregateRanger*, and the output will be a single combined reconciliation with support values for the inferred events and mappings. Thus, *AggregateRanger* is useful for computing support values for individual events and mappings by accounting for variance due to multiple optimal reconciliations and due to alternative event cost assignments.

- **Supplementary programs**
    4. *OptResolutions-DTL*: Takes as input a rooted gene tree with edge support values (bootstrap or similar) and a rooted species tree, collapses weakly supported edges of the gene tree based on a specified edge support cutoff value, and computes all optimal (i.e., ones with minimum reconciliation cost) binary resolutions of the collapsed (non-binary) gene tree. *OptResolutions-DTL* implements exact algorithms to find optimal resolutions of the collapsed gene tree that minimize the DTL reconciliation cost against the species tree. The software can be used to either compute a single optimal resolution and associated reconciliation cost, or to enumerate all optimal resolutions.
    5. *OptRoot-Dated*: This is the "dated" version of the *OptRoot* program. It requires a fully-dated species tree (i.e., chronogram) and restricts any postulated transfer events to only occur between coexisting species,
    6. *Ranger-DTL-Dated*: This is the "dated" version of the *Ranger-DTL* program. It requires a fully-dated species tree (i.e., chronogram) and restricts any postulated transfer events to only occur between coexisting species
    7. *Ranger-DTL-Fast*: Takes as input a rooted species tree and either a rooted or unrooted gene tree and implements the fastest known algorithm for DTL reconciliation to rapidly compute an optimal reconciliation for even very large trees with thousands of leaves (optimizing over all rootings of the gene tree if it is unrooted). In contrast to *Ranger-DTL*, it cannot sample the space of optimal reconciliations uniformly at random. (It does so non-uniformly). Likewise, in contrast to *OptRoot*, it cannot properly account for multiple optimal rootings of the gene tree (in case of an unrooted gene tree) and simply chooses one optimal rooting at random. This program corresponds to the program ranger-dtl-u from version 1 of the RANGER-DTL package, and is included here to enable those analyses which would be computationally infeasible using *Ranger-DTL.*
    8. *OptRoot-Fast:* The functionality of this program is identical to that of *OptRoot,* except that it does not allow for distance-dependent transfer costs. As with *Ranger-DTL-Fast, OptRoot*-Fast implements the fastest known algorithm to compute all optimal rootings by DTL reconciliation and can be used to rapidly compute all optimal rootings for even very large trees with thousands of leaves.

- **Summary scripts**
    - *SummarizeOptRootings*: This is a python script that aggregates reconciliations across different optimal rootings of the same gene tree and thus helps computes

support values for events and mappings while accounting for multiple gene tree rootings. Specifically, given a species tree and an unrooted gene tree, uses *OptRoot* (or *OptRoot-Dated*) to compute all optimal rootings of the gene tree, reconciles each optimally rooted gene trees multiple times using *Ranger-DTL* (or *Ranger-DTL-Dated*) to sample their optimal reconciliations, and then aggregates the resulting reconciliations into a single consensus reconciliation.

- *SummarizeOptResolutions*: This is a python script that aggregates reconciliations across different optimal resolutions of the same (poorly supported) gene tree and thus helps computes support values for events and mappings while accounting for alternative gene tree topologies. This script automatically calls *OptResolutions-DTL* and *Ranger-DTL* to produce its output.

3. Using the Core and Supplementary Programs

All programs and scripts listed above are designed to work with one another to enable sequential processing pipelines that can account for uncertainty in gene tree rooting or in the gene tree topology, account for multiple optimal reconciliations, and produce an easy-to-interpret aggregated reconciliation output. In the following we describe the input and output formats used by the core programs *OptRoot* and *Ranger-DTL,* and the supplementary programs *OptResolutions-DTL, OptRoot-Dated, Ranger-DTL-Dated, Ranger-DTL-Fast, and OptRoot-Fast*. The aggregation program AggregateRanger requires different input, which we will describe in detail at the end of this section.

Input file format: All core and supplementary programs take as input a single file (specified using the –i command line option) containing first the species tree, followed by a single gene tree. All input trees must be expressed using the Newick format terminated by a semicolon, and they must be fully binary (fully resolved) and rooted. Species names in the species tree must be unique.
E.g., ((speciesA, speciesB), speciesC);

Each leaf in the gene tree must be labeled with the name of the species from which that gene was sampled. If desired, the gene name can be appended to the species name separated by an underscore '_' character. The gene tree may contain any number (zero, one, or more) of homologous genes from the same species.

E.g., (((speciesA_gene1, speciesC_gene1), speciesB_geneX), speciesC_gene2);

and (((speciesA, speciesC), speciesB), speciesC);

are both valid gene tree inputs and, in fact, represent the same gene tree. This gene tree contains one copy of the gene from speciesA and speciesB, and two copies from speciesC.

Specifying event costs: Besides the input trees, all core and supplementary programs must also be given the costs for duplication, transfer, and loss events (specified using –D, –T, and –L options respectively) to be used for the reconciliation. If a cost is not specified, its default value is used instead. Note that each of these three costs must be positive integers. Thus, if one desires to use costs 1, 2.5 and 3.7 for loss, duplication, and transfer, they must use 10, 25, and 37, respectively, instead.

Using distance-dependent transfer costs: The core programs *OptRoot* and *Ranger-DTL* and the supplementary programs *OptRoot-Dated* and *Ranger-DTL-Dated* allow for the use of variable transfer costs. Specifically, they allow for two types of variable transfer costs, invoked using the –type 1 and –type 2 options. The default option is –type 0, which means that the fixed, non-distance-dependent transfer cost is used. When using Type 1 transfer costs the user specifies a threshold and an "additional transfer cost" parameter. If the distance (in the number of edges) between the donor and recipient species for the transfer is less than the threshold, the regular transfer cost is used, and for any distance larger than or equal to the threshold, the (regular transfer cost) + (additional transfer cost) is used. Type 2 variable transfer costs are a generalized version of type 1 costs, in that the transfer cost for a transfer is given by (regular transfer cost) + \floor((distance between donor and recipient) /threshold) * (additional transfer cost).

Sample commands:
./OptRoot.linux -i inputFile.newick -D 3 -T 4 -o outputFile
./Ranger-DTL.linux -i inputFile.newick -o output.reconciliation1

Special instructions for *OptResolutions-DTL*: For the program *OptResolutions-DTL*, each internal node of the gene tree, except for the root, must also have a support value (between 0 and 100 (%)) specified.

E.g., (((speciesA, speciesC)56, speciesB)90, speciesC);

or (((speciesA.gene1, speciesC.gene1)56, speciesB.geneX)90, speciesC.gene2);

In addition, users should also specify a support value cutoff, for collapsing weakly supported branches, using the –B command line option. If a support value cutoff is not specified, then a default value of 80 (%) is used.

Special instructions for all *OptRoot* programs*:* All core and supplementary programs read in trees that are fully binary and rooted. In case of *OptRoot, OptRoot-Dated*, and *OptRoot-Fast*, the input gene tree, though rooted, is automatically assumed to be unrooted. For convenience, *OptRoot*, *OptRoot-Dated*, and *OptRoot-Fast* will also accept unrooted gene trees as input, if they are otherwise binary. E.g. (((a, b), c), d); and ((a,b), c, d); are both valid gene tree inputs for the OptRoot programs and represent the same (unrooted) gene tree.

Special instructions for dated versions (*OptRoot-Dated* and *Ranger-DTL-Dated*): The two supplementary programs OptRoot-Dated and Ranger-DTL-Dated both require the input species tree to be fully dated and restrict any postulated transfer events to only occur between coexisting species.

E.g., ((SpeciesA:1.2, SpeciesB:1.2):0.8, (SpeciesC:1.5, SpeciesD:1.5):0.5);

Note that the species tree must be a chronogram (or ultrametric), with all leaf nodes equidistant from the root node in terms of the species tree branch lengths. The dated programs do not perform branch length error-checking and will work incorrectly if the species tree is not ultrametric. When using the core programs or other non-dated programs, any branch lengths on the species tree will be ignored.

Special instructions for *Ranger-DTL-Fast*: The supplementary program *Ranger-DTL-Fast* can work with both rooted and unrooted gene trees. As with all other programs, the gene tree must still be written in rooted newick format, but users can use the tag [&U] at the beginning of the gene tree to specify that the gene tree should be interpreted as being unrooted (and so rerooted optimally prior to the reconciliation).

E.g. [&U](d,(e,((a,b),c)));

Input for *AggregateRanger:* The core program *AggregateRanger* takes as input a set of reconciliation files computed using any of the Ranger-DTL programs (*Ranger-DTL*, *Ranger-DTL-Dated,* or *Ranger-DTL-Fast*) for the same gene and species tree. In creating these reconciliation files, one may even use different event costs to generate the reconciliations. Each of the reconciliation files must be named using a particular naming scheme: All of the reconciliation files to be aggregated must have the same file name prefix, appended with consecutive numbers starting from 1. For example, if there are 200 reconciliation files to be used in the analysis and the chosen file name prefix is “reconciliation”, then the 100 files must be named reconciliation1, reconciliation2, …, reconciliation200. To have *AggregateRanger* analyze and aggregate these files, one must pass the file name prefix as a command line option to the program; E.g.,

```
./AggregateRanger  path_to_files/reconciliation
```

4. Test files

We provide three simple test files to help illustrate the proper input format for various core and supplementary programs and to help with understanding program output.

- testFile_1.newick is included to illustrate the proper input format for *OptRoot* and *Ranger-DTL* (and also their *fast* variants*)*.
- testFile_2.newick has the same species and gene trees as testFile_1.newick, but the species tree is fully dated (ultrametric). Thus, this file is designed to be used with the dated versions, *OptRoot-Dated* and *Ranger-DTL-Dated.* This file can also be used with the other OptRoot and Ranger-DTL programs.
- testFile_3.newick also uses the same species and gene trees as testFile_1.newick, but the gene tree edges are labeled with support values. This file is thus included to illustrate the proper input format for *OptResolutions-DTL.*

5. Interpreting the output

Executing the included test files should give the user a good idea of what the output for the various programs looks like and how it is to be interpreted. We describe some of the specifics below:

OptRoot, OptRoot-Dated*, and *OptRoot-Fast: All three of these optimal rooting programs have identical output format. The output for the optimal resolutions program consists of a list of optimally rooted gene trees, followed by the reconciliation cost of these optimally rooted gene trees, and the total number of optimal rootings.

As an example, when *OptRoot* is executed on textFile_1.newick using the command `./OptRoot.linux -i TEST_DATA/testFile_1.newick, the output is:`

```
Input file: TEST_DATA/testFile_1.newick
Duplication cost: 2, Transfer cost: 3, Loss cost: 1
Random seed: 996
input species tree of 6 taxa
unrooted input gene tree of 8 taxa
1 input gene trees total
Computing...

 ------------ All Optimal Rootings for Gene Tree 1 (Unrooted) -------------
(d,(a_2,(((e_5,f_100),e_2),((a_1,b),c))));

The minimum reconciliation cost is: 9
The total number of optimal rootings is: 1
time: 0s
```

Ranger-DTL*, *Ranger-DTL-Dated*, and *Ranger-DTL-Fast: All three of these DTL reconciliation programs have identical output format. The reconciliation output begins with the species tree, which is the same as given in the input but for which the internal nodes have been labeled. This is followed by the gene tree, again with internal nodes labeled. (If *Ranger-DTL-Fast* was used and the gene tree was originally unrooted, then an optimally rooted version of that gene tree is output here.) This is followed by the reconciliation itself, which consists of the nodes of the gene tree, output one at a time in post order. Each of these node is labeled either as a leaf node, a speciation node, a duplication node, or a transfer node, and is assigned a mapping to some node from the species tree (this mapping defines the embedding of the gene tree into the species tree). For transfer events, in addition to the mapping (which, in this case, represents the donor species), the recipient species is also specified. Finally, at the end of the reconciliation we output the total reconciliation cost as well as the number of duplication, transfer, and loss events.

As an example, when *Ranger-DTL* is executed on textFile_1.newick using the command `./Ranger-DTL.linux -i TEST_DATA/testFile_1.newick`, the output is:

```
Input file: ../../TEST_DATA/testFile_1.newick
Duplication cost: 2, Transfer cost: 3, Loss cost: 1
Random seed: 1455
input species tree of 6 taxa
rooted input gene tree of 8 taxa
1 input gene trees total
Computing...

 ------------ Reconciliation for Gene Tree 1 (rooted) -------------
Species Tree:
(((a,(b,c)n4)n3,d)n2,(e,f)n5)n1;

Gene Tree:
(((a_1,b)m3,c)m2,((d,a_2)m8,((e_5,f_100)m12,e_2)m11)m7)m1;

Reconciliation:
a_1: Leaf Node
b: Leaf Node
m3 = LCA[a_1, b]: Transfer, Mapping --> b, Recipient --> a
c: Leaf Node
m2 = LCA[a_1, c]: Speciation, Mapping --> n4
d: Leaf Node
a_2: Leaf Node
m8 = LCA[d, a_2]: Speciation, Mapping --> n2
e_5: Leaf Node
f_100: Leaf Node
m12 = LCA[e_5, f_100]: Speciation, Mapping --> n5
e_2: Leaf Node
m11 = LCA[e_5, e_2]: Duplication, Mapping --> n5
```

```
m7 = LCA[d, e_2]: Speciation, Mapping --> n1
m1 = LCA[a_1, e_2]: Duplication, Mapping --> n1

The minimum reconciliation cost is: 12 (Duplications: 2, Transfers: 1, Losses: 5)
Total number of optimal solutions: 10
Total number of candidates for gene birth: 2

time: 0s
```

OptResolutions-DTL: The output for the optimal resolutions program consists of basic statistics for the input gene tree (including maximum out-degree, number of non-binary nodes, and current reconciliation cost), followed by the optimal DTL reconciliation cost (obtained after optimal resolution) and the number of optimal resolutions of the gene tree, and finally a list of all optimal resolutions of the gene tree.

AggregateRanger: The output of *AggregateRanger* is similar to that of *Ranger-DTL*. The output aggregate reconciliation lists each gene tree node in post order (one node per line), and shows the number of times that node was labeled as a speciation, duplication, or transfer, in the input set of sampled reconciliations. Likewise, for each node it shows the most frequent mapping for that node among the input reconciliations, along with the number of times that mapping was seen. The output of *AggregateRanger* thus provides support values for the event type and mapping of each node in the gene tree. These support values are computed based on the fraction (percentage) of sampled reconciliations that support that event assignment or mapping assignment. *AggregateRanger* can be used to aggregate reconciliations and compute support values across multiple optima as well as across different event cost assignments.

Note that *AggregateRanger* does not output the most frequent recipient for transfer nodes. This is for two reasons: First, a node may not be a transfer event across all aggregated reconciliations. And second, even if a node is always labeled as a transfer event, its most frequent mapping may not always be associated with its most frequent recipient. A version of AggregateRanger that also outputs the most frequent recipient is available upon request from Mukul Bansal (mukul.bansal@uconn.edu).

6. Using the summary scripts *SummarizeOptRootings* and *SummarizeOptResolutions*

Input: The input file format for *SummarizeOptRootings* is identical to that of *OptRoot* (or *OptRoot-Dated*) and the input to *SummarizeOptResolutions* is identical to that of *OptResolutions-DTL*.

The command line options available for *SummarizeOptRootings* include all those available for *OptRoo*t. In addition, there are options for controlling the number of sampled optimal reconciliations computed for each optimal rooting (default value 100), and for controlling whether the dated or undated versions of *OptRoot* and *Ranger-DTL* should be used.

Likewise, command line options for *SummarizeOptResolutions* includes those available for *OptResolutions-DTL* and, in addition, includes options for controlling the number of optimal resolutions to be computed (sampled uniformly at random; default value 100), and for the number of optimal sampled optimal reconciliations to be computed for each optimal resolution (default value 100).

Sample commands:

python SummarizeOptRootings.py -i inputFile.newick -o outputFile.results

python SummarizeOptResolutions.py -i inputFile.newick -o outputFile.results

SummarizeOptRootings.py and SummarizeOptResolutions.py have been tested to work with both Python 2 and 3. If either of the scripts throws a "permission denied" error, then please make sure all files have execute permission.

***SummarizeOptRootings* Output:** This script computes and outputs a "consensus" reconciliation, i.e., a reconciliation for the strict consensus tree of all optimal rootings of the gene tree. The output consists of two files: A short output file and a long output file. The short output file is formatted similarly to the output from *AggregateRanger*, except that it only lists nodes of the strict consensus tree of all optimal rootings. This includes all those nodes of the gene tree that are conserved across all the optimal rootings of that gene tree. The long output file is formatted similarly but contains summary reconciliation information for each optimal rooting separately. Thus, if there are three optimal rootings, each conserved gene tree node is listed three times in the long output file, its event assignment and most frequent mapping summarized separately for each rooting. In the short file, these three lines would be further condensed into a single line, with event assignments and most frequent mappings combined across all three optimal rootings. In both the short and long files, the reconciliation is divided into two separate blocks, one consisting of all internal nodes of the strict consensus tree (since they actually represent subtrees conserved across all optimal rootings), and the other consisting of the root of the strict consensus tree (since it does not represent a conserved subtree).

***SummarizeOptResolutions* Output:** *SummarizeOptResolutions* outputs a single file which is also formatted similarly to the output of *AggregateRanger*. This output file only lists those nodes of the gene tree that are present in the non-binary gene tree obtained after collapsing

the weakly supported edges of the input gene tree. This output lists, for each binary and non-binary gene node, the frequencies of the event assignments seen among the sampled reconciliations across the sampled optimal resolutions, as well as the most frequent mapping assignment seen among the sampled reconciliations across all sampled optimal resolutions. The output file also includes basic summary statistics, as well as a list of the sampled optimal resolutions used in the analysis.

7. Notes on using RANGER-DTL 2.0 effectively

The various programs included in the RANGER-DTL 2.0 software package are designed to work together to enable accurate evolutionary analyses.

In most cases, users will only need to use the core programs *OptRoot*, *Ranger-DTL*, and *AggregateRanger* for their analysis. Specifically, if a specific (well supported) rooted gene tree is used, then we recommend using *Ranger-DTL* multiple times (say 100 times with different random seeds) to sample the space of multiple optimal reconciliations and then to aggregate the samples using *AggregateRanger*. If there is uncertainty about the event cost assignments to be used, then we recommend using *Ranger-DTL* multiple times with a few different event cost assignments (for instance, 100 samples each using [D, T, L] costs of [2,3,1], [3,3,1], and [2,4,1], for a total of 300 samples), and then aggregate all samples using *AggregateRanger*. If the gene tree has uncertain rooting or is unrooted, then the program *OptRoot* should be used first to compute all optimal rootings for the gene tree, and then the procedure described above should be used with each optimally rooted gene tree (there will often be only one optimal rooting). If the user is interested in only an aggregate summary of the reconciliation across all optimal rootings then the summary script *SummarizeOptRootings* can be used to automate this computation.

If the gene tree is rooted but not well supported (or has uncertain topology), and bootstrap or other support values are available for the gene tree branches, then the supplementary program *OptResolutions* can be used to collapse weakly supported branches and compute all optimal resolutions of the resulting non-binary gene tree. The procedure outlined in the previous paragraph can then be used for each optimally resolved gene tree (or for a random sample of the optimal resolutions, if there are too many). If the user is interested in only an aggregate summary of the reconciliation across all optimal resolutions then the summary script *SummarizeOptResolutions* can be used to automate this computation. Please note that due to computational complexity the program *OptResolutions* (also *SummarizeOptResolutions*, by extension) is currently only able to handle non-binary gene trees with maximum out-degree up to 8.

If the species tree is fully dated and a dated analysis is desired then the corresponding dated versions *OptRoot-Dated* and *Ranger-DTL-Dated* may be used, though we still recommend using the core, undated, versions. We prefer to use the undated versions because it is often difficult to accurately date species trees and because transfers from extinct or unsampled lineages may violate time constraints.

Finally, the fast versions *OptRoot-Fast* and *Ranger-DTL-Fast* should only be used if the other programs are too slow to be applied to your input trees or if an extremely fast runtime is desired (at the cost of reduced functionality).

Larger dataset and sample Bash script file: A typical genome-scale analysis of gene family evolution consist of a rooted species tree and a large number of rooted or unrooted gene trees. To familiarize users with such datasets and their analysis using RANGER-DTL 2.0, we provide a simulated dataset with a species tree on 100 taxa and a set of 10 gene trees evolved inside that species tree. To facilitate the analysis, we also provide a Bash script to automatically analyze these 10 gene trees, creating 10 aggregate reconciliation files (one for each gene tree). A README.txt file is also included explaining how to use and modify this Bash script. These files are placed inside the "LargerDataSet" folder inside TEST_DATA.

Links to biological datasets: For additional practice and analysis, interested users can easily obtain the biological data sets associated with the following research papers:

LA David, EJ Alm. "Rapid evolutionary innovation during an Archaean Genetic Expansion." Nature, 2010. http://dx.doi.org/10.1038/nature09649.

Szöllősi GJ, Arellano Davín A, Tannier E, Daubin V, Boussau B. "Genome-scale phylogenetic analysis finds extensive gene transfer among fungi." Phil. Trans. R. Soc. B 370: 20140335. http://dx.doi.org/10.1098/rstb.2014.0335.

For the first paper, data can be obtained either by emailing the paper authors or from Mukul Bansal (mukul.bansal@uconn.edu). For the second paper, data is available from the Dryad repository at https://datadryad.org/resource/doi:10.5061/dryad.r2t49

Appendix: List of command line options for the core and supplementary programs

Available command line options for *OptRoot* and *OptRoot-Dated*

```
-i, --input              input file
-o, --output             output file
-D,                      Duplication cost (whole number only, default value 2)
-T,                      Transfer cost (whole number only, default value 3)
-L,                      Loss cost (whole number only, default value 1)
-r, --random             Only output one optimal rooting (instead of enumerating all optimal
                         rootings).
   --type  0|1|2         type of transfer cost to use
                         0 - Fixed transfer cost [default]
                         1 - Simple threshold based variable transfer cost
                         2 - General threshold based variable transfer cost
   --thr <whole number>  threshold value for use with types 1 and 2 (default 10)
   --add <whole number>  additional transfer cost for use with types 1 and 2 above
                         (default value is transfer cost divided by 2, rounded down)
   --seed <whole number> set a user defined random number generator seed.
-q, --quiet              no process output
-v, --version            version number
-h, --help               brief help message
```

Available command line options for *Ranger-DTL* and *Ranger-DTL-Dated*

```
-i, --input              input file
-o, --output             output file
-D,                      Duplication cost (whole number only, default value 2)
-T,                      Transfer cost (whole number only, default value 3)
-L,                      Loss cost (whole number only, default value 1)
   --type  0|1|2         type of transfer cost to use
                         0 - Fixed transfer cost [default]
                         1 - Simple threshold based variable transfer cost
                         2 - General threshold based variable transfer cost
   --thr <whole number>  threshold value for use with types 1 and 2 (default 10)
   --add <whole number>  additional transfer cost for use with types 1 and 2 above
                         (default value is transfer cost divided by 2, rounded down)
   --seed <whole number> set a user defined random number generator seed.
```

-q, --quiet	no process output
-s, --summary	only output summary statistics
-v, --version	version number
-h, --help	brief help message

Available command line options for *OptResolutions-DTL*

-i, --input	Input file
-o, --output	Output file
-D,	Duplication cost (whole number only, default value 2)
-T,	Transfer cost (whole number only, default value 3)
-L,	Loss cost (whole number only, default value 1)
-B	Bootstrap cutoff value (whole number between 0 and 100 only, default value 80)
-N	Maximum number of optimal resolutions to be printed to output file (default value 2,147,483,647). If the actual number of optimal reconciliations is greater than this value, then the specified number of resolutions is sampled uniformly at random from the set of optimal resolutions.
-s, --summary	Only output summary statistics
-q, --quit	Suppress program output. Only output optimized reconciliation cost, number of optimal resolutions, and list of optimal resolutions.
-v, --version	version number
-h, --help	brief help message

Available command line options for *Ranger-DTL-Fast*

-i, --input	input file
-o, --output	output file
-D,	Duplication cost (whole number only, default value 2)
-T,	Transfer cost (whole number only, default value 3)
-L,	Loss cost (whole number only, default value 1)
--seed <whole number>	set a user defined random number generator seed.
-q, --quiet	no process output
-s, --summary	only output summary statistics
-v, --version	version number
-h, --help	brief help message

Available command line options for *OptRoot-Fast*

-i, --input	input file
-o, --output	output file
-D,	Duplication cost (whole number only, default value 2)
-T,	Transfer cost (whole number only, default value 3)
-L,	Loss cost (whole number only, default value 1)
-r, --random	Only output one optimal rooting (instead of enumerating all optimal rootings).
--seed <whole number>	set a user defined random number generator seed.
-q, --quiet	no process output
-v, --version	version number
-h, --help	brief help message