TreeStats manual

Aasa Feragen, Megan Owen

June 25, 2013

1 Introduction

The TreeStats software package is an implementation of the hypothesis tests and set versions of means, PCA and LDA presented in the paper [1].

The TreeStats software package consists of the following MATLAB functions:

- `set_mean.m` (computes the set mean based on a distance matrix)
- `pca.m` (from a file with input trees and a number of random iterations, computes the best principal geodesic segment and returns its endpoints)
- `lda.m` (from a testset file with input trees and a number of random iterations, computes the best lda segment and returns its endpoints. Given an additional testset file with input trees, a classification accuracy is additionally returned.)
- `hypothesis_tests_sturmmeans.m` (permutation tests for dataset means, variance and spread, where means and variance rely on converged (Sturm) means)
- `hypothesis_tests_setmeans.m` (permutation tests for dataset means, variance and spread, where means and variance rely on set means)
- `treestats_demo.m` (a software demonstration. To run, go to the directory containing this manual and all `.m` files and run `treestats_demo`).

These MATLAB functions rely on the following Java executable files, which can also be used on their own:

- `GTP.jar` This is the tree-space geodesic algorithm by Megan Owen and Scott Provan [3]. The current version as of June 20 2013 is included in the TreeStats software package; however, we recommend the user to check `http://www.unc.edu/depts/stat-or/miscellaneous/provan/treespace/` for an updated version. An instruction manual for running this software independently is also available at this site.
- `SturmMean.jar` This is the SturmMean algorithm by Ezra Miller, Megan Owen and Scott Provan [2]. The current version as of June 20 2013 is included in the TreeStats software package; however, we recomend the user to check `https://cs.uwaterloo.ca/~m2owen/code.html` for an updated version. An instruction manual for running this software independently is also available at this site.
- `PCA.jar` This contains the algorithms related to Principal Component Analysis. An instruction manual can be found in Appendix A.
- `LDA.jar` This contains the algorithms related to Linear Discriminant Analysis. An instruction manual can be found in Appendix B.

- `analysis.jar` This contains a helper program used by the python script `create_comp_matrix.py`, and is treated as a black box.

Moreover, a single python script is used by some of the MATLAB functions:

- `create_comp_matrix.py` (Computes a distance matrix between two sets of trees given by two input text files).

2 Installation

TreeStats is available at `http://image.diku.dk/aasa/software.php`. It requires Java version J2SE 6.0 which can be downloaded for free at `java.sun.com`, and Python, which can be downloaded for free at `python.org`.

The following files are included with the distribution:

- the `.m`, `.py` and `.jar` files listed in Section 1
- the GNU general public license under which this code is released
- this manual `treestats_manual.pdf`

3 Running the software

In general, tree-valued input should consist of `.txt` files containing one tree per line, in Newick format. See `http://evolution.genetics.washington.edu/phylip/newicktree.html` for a description of the Newick format. The trees can also contain vectors on the edges instead of single lengths. See Section 3.2 for details, and see the file `airways_dataset.txt` for an example input file.

For the MATLAB functions, see their preamble for general instructions on running the code, but pay attention to the note on selecting the `epsilon` parameter in Section 3.1.

3.1 Choosing convergence parameters for hypothesis tests, PCA and LDA

Some of the included functions rely on a convergence parameter `epsilon`. The SturmMean.jar uses `epsilon` to determine convergence towards the Fréchet mean, which means `epsilon` needs to be selected when running `hypothesis_tests_sturmmeans.m`. Both LDA and PCA rely on projecting points onto geodesics, which is implemented as an iterative procedure depending on a convergence parameter `epsilon`. Selecting a suitable `epsilon` value depends on the properties of the input trees; too large epsilon gives inaccurate results while too small epsilon gives slow convergence. This is important when running `LDA.jar`, `PCA.jar`, `LDA.m` and `PCA.m`. Suitable choices of `epsilon` depend on the typical norm of a branch attribute. For the examples in this manual, an `epsilon` in the range 10^{-4} to 10^{-6} works well; for the airway examples used in the treestats demo, an `epsilon` of 10^{-4} to 10^{-3} works well.

3.2 Trees with Vectors on their Edges

As described in [1], we can also consider a space of trees in which each split is associated with an m-dimensional vector instead of a single length. To represent trees in this space, replace each edge length in the Newick format with a vector of the form $[a_1\ a_2\ a_3\ ...\ a_m]$, where $a_i \in \mathbb{R}$ for each i. That is, the vector consists of m real numbers separated by single spaces, and enclosed by square brackets.

For example, the tree in Figure 1 should be represented as `((a:[1 1],b:[-1 1]):[2 0],c:[4 -1.5]);` in the input tree file.

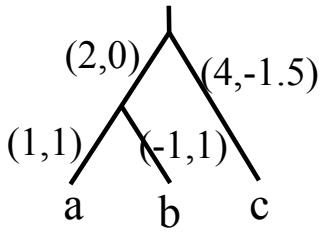

Figure 1: Tree with vectors on its edge instead of lengths.

4 License

References

[1] A. Feragen, M. Owen, J. Petersen, L.H. Thomsen, M.M.W. Wille, A. Dirksen, and M. de Bruijne. Tree-space statistics and approximations for large-scale analysis of anatomical trees. In *Information Processing in Medical Imaging*, 2013.

[2] E. Miller, M. Owen, and S. Provan. Averaging metric phylogenetic trees. *arXiv:1211.7046 [math.MG]*, 2012.

[3] M. Owen and J. S. Provan. A fast algorithm for computing geodesic distances in tree space. *IEEE/ACM Trans. Comp. Biol. Bioinf.*, 8:2–13, 2011.

A Manual for `PCA.jar`

This jar file contains algorithms for related to doing Principal Components Analysis in tree space.

A.1 Running PCA

PCA is run from the command line, using Java. The following command assumes you are in the directory containing `PCA.jar`.

```
> java -jar PCA.jar [options] treefile
```

`treefile` is the name of the file containing the list of trees. For all tree files, the trees should be one per line, in Newick format. See `http://evolution.genetics.washington.edu/phylip/newick tree.html` for a description of the Newick format. The trees can also contain vectors on the

edges instead of single lengths. See Section 3.2 for details.

Options (described below in more detail):

`-a <algorithm>`	select the algorithm to be run. Choices for `<algorithm>` are `random`, `projection_indices`, and `distances_to_geo`.
`-e <epsilon>`	set the value of epsilon used to determine the precision of the tree projections.
`-f <otherTreeFile>`	specify a second tree file.
`-h`	display a summary of the options.
`-i <numIterations>`	max number of iterations for algorithm. Default is 100. `numIterations` must be less than 2 147 483 647.
`-o <outfile>`	store the output in the specified file. Default is output.txt.
`-r`	output the number of orthants that a geodesic passes through.
`-u`	set input trees as unrooted. Default is trees are rooted.

Option `-a` selects which algorithm is run. There are currently three options:

`random`	is used to randomly search for the best fitting geodesic between the given trees. More specifically, at each iteration, this algorithm randomly choses two trees (t_1 and t_2) from the set of input trees, and computes the sum of the squared distances from each input tree to its projection onto the geodesic between t_1 and t_2. The algorithm then prints a line with the following information to the output file: index of tree t_1 in input tree file, index of tree t_2 in input tree file, number of orthants that the geodesic passes through (if `-r` flag is used), sum of squared distance of trees to their projections. Note that the input tree file is indexed starting at 0.
`projection_indices`	computes the projections of the trees in `treefile` onto the geodesic between the two trees in `otherTreeFile`, as specified by the `-f` flag. More specifically, if t_1 is the first tree in `otherTreeFile` and t_2 is the second tree in `otherTreeFile`, then we linearly parametrize the geodesic from t_1 to t_2 by t, such that $t = 0$ at t_1 and $t = 1$ at t_2. Each projection is output as a real number between 0 and 1, representing the value of t at that point on the geodesic.
`distances_to_geo`	calculates the geodesic distances between the trees in the input file (`tree file`) and their projections onto the geodesic between the first two trees in `otherTreeFile`. These geodesic distances are output one per line.

Option `-e` is used to specify the epsilon value that is the maximum error allowed in calculating tree projections. Specifically, epsilon is the maximum geodesic distance allowed between the true projected tree and the approximate tree used.

Option `-f` is used to specify a second tree file, as required by the algorithm options `projection_indices` and `distances_to_geo`.

Option `-h` displays a brief usage method and exits.

Option `-i` specifies the number of iterations for the algorithm option `random`.

Option `-o` is used to specify either the output file or the prefix for the output files if one is generated for each iteration. The default is output.txt.

Option `-r` is used with algorithm option `random` to specify that the output file should also contain the number of orthants that a geodesic passes through. This includes the starting and ending orthants.

Option `-u` is used to indicate that the trees in `treefile` are unrooted. The default is rooted trees. The root acts as an extra leaf that does not appear in the Newick representation, and thus this flag must be used correctly, or the output may be wrong.

A.2 Examples

A.2.1 Example 1: Algorithm `random`

The first example consists of four trees, arranged in tree space as shown in Figure 2. The trees are labelled by their index in the input file `input.txt`, and by their coordinates in tree space. The coordinates correspond to the splits in the following order: $ab|cde$, $abc|de$, $ac|bde$, $bc|ade$.

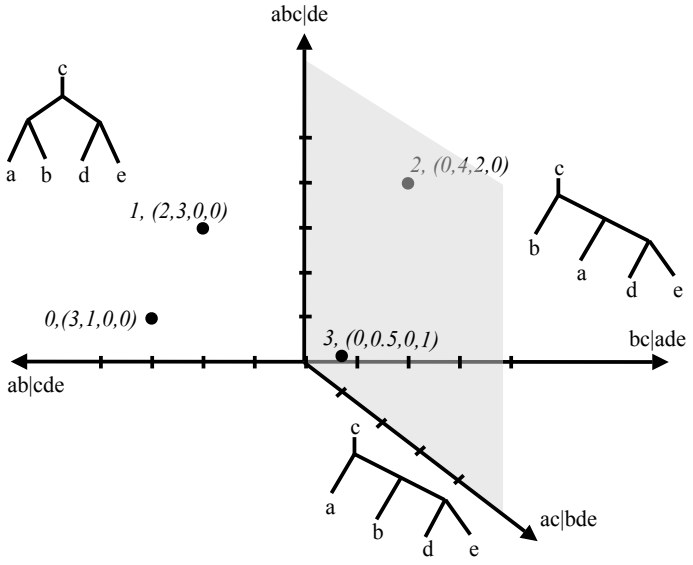

Figure 2: Location of trees in Example 1. The axes are labelled with their corresponding split, and the coordinates represent the splits in the following order: $ab\|cde$, $abc\|de$, $ac\|bde$, $bc\|ade$.

To run this example, put `PCA.jar` into the directory `examples/PCAexample1`. At the command line, go to that directory, and type the following:

```
java -jar PCA.jar -a random -e 0.000001 -i 10 -o output -r -u input.txt
```

Running this command will generate the files `output_0`, `output_1`, ..., `output_9` in the current directory. If we use `outDir/output` instead of `output` in the above command, the files

Index 1	Index 2	Num Orthants	Geo Distance	Score
0	1	1	2.236068	31.45
0	2	2	5.830952	7.625
0	3	2	4.031129	19.836779
1	2	2	4.123106	15.0
1	3	2	3.905125	15.819672
2	3	2	4.609772	15.222222

Table 1: Possible second lines for the output files from Example 1.

would be generated in the subdirectory `outDir`.

Compare `output_0` to `example1_output_0` in the `PCAexample1` subdirectory.

```
Index 1  Index 2  Num Orthants  Geo Distance        Score
3        2        2             4.6097722286464435  15.23343
```

The first line gives the headers for the second line. As the choice of tree endpoints for the geodesic is random, the second line may differ from your file. The possible second lines are given in Table **??**, where the distances have been rounded to 6 decimal places. The first two numbers represent the indices in the input file of the two endpoint trees of the geodesic. The first tree in the file has index 0, the second tree has index 1, etc. The following column, `Num Orthants`, only appears if the `-r` flag is used. It gives the number of orthants that the geodesic passes through, including the beginning and ending ones, but not including the boundary orthants. Here, the 2 indicates that the geodesic starts in one orthant, passes through a boundary orthant, and finishes in a second orthant. The next column gives the geodesic distance between the two selected trees. That is, it gives the length of the geodesic being projected onto. The final column gives the score of the projections, or the sum of squared geodesic distances between the remaining input trees and their projections on the geodesic.

Note that the accuracy of the projections, and hence the score, is related to the epsilon chosen, not to the number of decimals displayed.

A.2.2 Example 2: Algorithm `projection_indices`

The second example consists of same four trees as in Example 1 (Figure 2).

To run this example, put `PCA.jar` into the subdirectory `examples/PCAexample2`. At the command line, go to that subdirectory, and type the following:

```
java -jar PCA.jar -a projection_indices -e 0.000001 -f endpoints.txt -r -u
   input.txt
```

One file will be generated in the current directory, `output.txt`, which contains the indices of the projections of the four trees in `input.txt` onto the geodesic that has the two trees in `endpoints.txt` as its endpoints. Compare `output.txt` to `example2_output.txt` in the `PCAexample2` subdirectory:

```
0.0
1.0
1.0
0.6000000537490502
```

The first two trees in `input.txt` are the same as the two trees in `endpoints.txt`, and so are the endpoints of the geodesic we are projecting onto. Thus they project onto themselves, giving indices 0 and 1, respectively. The third tree in `input.txt` also projects onto the second endpoint, and thus has index 1.0. The final tree, however, projects non-trivially onto the middle of the geodesic.

A.2.3 Example 3: Algorithm `distances_to_geo`

The third example consists of the same four trees as in Example 1 and 2 (Figure 2).

To run this example, put `PCA.jar` into the subdirectory `PCAexample3`. At the command line, go to that directory, and type the following:

```
java -jar PCA.jar -a distances_to_geo -e 0.000001 -f endpoints.txt -r -u
   input.txt
```

One file will be generated in the current directory, `output.txt`, which contains the geodesic distances from the trees `input.txt` to their projections onto geodesic that has the two trees in `endpoints.txt` as its endpoints. Compare `output.txt` to `example3_output.txt` in the `PCAexample3` subdirectory:

```
0.0
0.0
4.123105625617661
3.8013155617496426
```

The first two trees in `input.txt` project to themselves, so those two geodesic distances are 0. The third and fourth trees project non-trivially, and thus have a non-zero geodesic distance.

B Manual for LDA.jar

B.1 Running LDA

LDA is run from the command line, using Java. The following command assumes you are in the directory containing `LDA.jar`.

```
> java -jar LDA.jar [options] treefile
```

`treefile` is the name of the file containing the list of trees. For all tree files, the trees should be one per line, in Newick format. See `http://evolution.genetics.washington.edu/phylip/newicktree.html` for a description of the Newick format. The trees can also contain vectors on the edges instead of single lengths. See Section 3.2 for details.

Options (described below in more detail):

`-a <algorithm>`	select the algorithm to run. Choices for `<algorithm>` are `random`, `projection_indices`, and `distances_to_geo`.
`-e <epsilon>`	set the value of epsilon used to determine the precision of any projections.
`-f <otherTreeFile>`	specify a second tree file.
`-g <otherTreeFile>`	specify a third tree file.
`-h`	display a summary of the options.

`-i <numIterations>`	max number of iterations for algorithm. Default is 100. `numIterations` must be less than 2 147 483 647.
`-m <means>`	specify the means for classification.
`-o <outfile>`	store the output in the specified file. Default is output.txt.
`-u`	set input trees as unrooted. Default is trees are rooted.

Option `-a` selects which algorithm is run. There are currently two options:

`random`	is used to randomly search for the geodesic with the lowest LDA score. More specifically, at each iteration, this algorithm randomly choses two trees (t_1 and t_2) from the third tree file, and computes the score of the projections of the trees in the first tree file (Set 1) and second tree file (Set 2) onto the geodesic with endpoints t_1 and t_2, as described in [1]. The algorithm then prints the following information to an output file: tree t_1, tree t_2, score, geodesic index of the mean of the projections of set 1 (Mean 1), and the geodesic index of the mean of the projections of set 2 (Mean 2).
`classify`	classifies the trees in the tree file by which mean their projection is closer to. The projection is onto the geodesic specified by the endpoints in the second tree file. The classifications ("1" or "2") are written one per line into in the output file, in the same order as the trees in tree file.

Option `-e` is used to specify the epsilon value that is the maximum error allowed in calculating tree projections. Specifically, epsilon is the maximum geodesic distance allowed between the true projected tree and the approximate tree used.

Option `-f` is used to specify a second tree file, as required by the algorithm options `random` and `classify`.

Option `-g` is used to specify a third tree file, as required by the algorithm option `random`.

Option `-h` displays a brief usage method and exits.

Option `-i` specifies the number of iterations for the algorithm option `random`.

Option `-m <means>` specifies the two mean indices for the algorithm option `random`. They should be separated by a single comma, without any spaces.

Option `-o` is used to specify either the output file or the prefix for the output files if one is generated for each iteration. The default is output.txt.

Option `-u` is used to indicate that the trees in `treefile` are unrooted. The default is rooted trees. The root acts as an extra leaf that does not appear in the Newick representation, and thus this flag must be used correctly, or the output may be wrong.

B.2 Examples

B.2.1 Example 1: Algorithm `random`

The first example consists of six trees, arranged in tree space as shown in Figure 3. The coordinates correspond to the splits in the following order: $ab|cde$, $abc|de$, $ac|bde$, $bc|ade$. Trees 1 and 2 are in the file `set1.txt` and trees 3, 4, 5, 6 are in the file `set2.txt`, in that order. The file `endpts.txt` contains trees 1, 2, 3, 4.

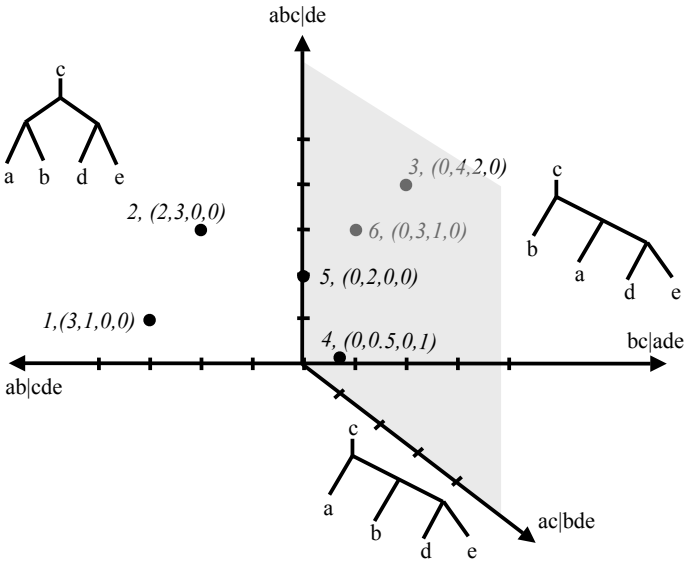

Figure 3: Location of trees in Example 1. The axes are labelled with their corresponding split, and the coordinates represent the splits in the following order: $ab|cde$, $abc|de$, $ac|bde$, $bc|ade$.

To run this example, put `LDA.jar` into the directory `examples/LDAexample1`. At the command line, go to that directory, and type the following:

```
java -jar LDA.jar -a random -e 0.000001 -i 10 -o output -u -f set2.txt -g
   endpts.txt set1.txt
```

Running this command will generate the files `output_0`, `output_1`, ..., `output_9` in the current directory.

Compare `output_0` to `example1_output_0` in the `example1` subdirectory.

```
((a:1,b:1):2,c:1,(d:1,e:1):3);
((a:1,b:1):3,c:1,(d:1,e:1):1);
Score:  3.025718900029112
Mean 1:  0.5
Mean 2:  0.09999998656273751
```

The first two lines are the trees chosen as the endpoints for the geodesic γ. Letting A be the trees in `set1.txt` and B be the trees in `set2.txt`, the third line is the value

$$\frac{d^2(\hat{\mu}(\mathrm{pr}_\gamma(A)), \hat{\mu}(\mathrm{pr}_\gamma(B)))}{\hat{s}^2(\mathrm{pr}_\gamma(A)) + \hat{s}^2(\mathrm{pr}_\gamma(B))}$$

where $\mathrm{pr}_\gamma(A)$ is the projection of the trees in set A onto the geodesic γ, $\mu(\hat{X})$ is the mean of

Endpoint 1	Endpoint 2	Score	Mean 1	Mean 2
1	2	3.025719	0.5	0.9
1	3	348.22839	0.161765	0.668382
1	4	2520.596414	0.092308	0.845192
2	3	133.203547	0.0	0.602941
2	4	363.629338	0.065574	0.6926238
3	4	1.110813	0.666667	0.55

Table 2: Possible output values for Example 1.

the trees in set X, and d is the geodesic distance. This is the same as Equation (4) in [1].

The fourth line in file `example_output_0` is the index on geodesic γ of the mean of the projections of the trees in `set1.txt`, and the fifth line in the file is the index on geodesic γ of the mean of the projections of the trees in `set2.txt`. Possible alternative values are given in Table 2. The numbers have been rounded to 6 decimal places. Additionally, the order of the endpoint trees determine which tree has index 0 and which has index 1, and thus the indices of the mean trees.

Note that the accuracy of the projections, and hence the score, is related to the epsilon chosen, not to the number of decimals displayed.

B.2.2 Example 2: Algorithm `classify`

The second example consists of same six trees as in Example 1 (Figure 3).

To run this example, put `LDA.jar` into the subdirectory `examples/LDAexample2`. At the command line, go to that directory, and type the following:

```
java -jar LDA.jar -a classify -e 0.000001 -f endpoints.txt -r -u input.txt
```

One file will be generated in the current directory, `output.txt`, which contains the indices of the projections of the four trees in `input.txt` onto the geodesic that has the two trees in `endpoints.txt` as its endpoints. Compare `output.txt` to `example2_output.txt` in the `LDAexample2` subdirectory:

```
0.0
1.0
1.0
0.6000000537490502
```

The first two trees in `input.txt` are the same as the two trees in `endpoints.txt`, and so are the endpoints of the geodesic we are projecting onto. Thus they project onto themselves, giving indices 0 and 1, respectively. The third tree in `input.txt` also projects onto the second endpoint, and thus has index 1.0. The final tree, however, projects non-trivially onto the middle of the geodesic.