# TreeTop: Installation and tutorial

## Table of Contents

This document contains information about the installation and usage of TreeTop. [add paper details]

## Installation

Installing TreeTop is simple: copy the directory titled *TreeTop* to somewhere on your path. If you wish to reproduce some of the examples shown in the paper, please also copy the folder *treetop_data* to some location (does not have to be in the path).

## Overview

To run TreeTop, there are three functions:

- `treetop_pre_run`: Diagnostic function to examine the distributions of markers across the input files, and to use mutual information to help determine any markers which could be excluded.
- `treetop`: Runs treetop non-recursively. `parpool` should be called before running. Outputs produced are: TreeTop layout annotated by input (and extra) markers, to assist with understanding what processes are taking place in the data; TreeTop layout annotated by the input files at each reference node; distribution of the density values calculated for each reference node (as a diagnostic for determining best value of sigma to use); distribution of branching scores over the data, including whether any are significant, plus the branches and branching point identified by TreeTop; profiles of markers as the identifed branches are traversed through the branching point.
- `treetop_recursive`: Runs treetop recursively, so that any branches identified in the whole dataset are then analysed by TreeTop for further branching, and so on. `parpool` should be called before running. Outputs produced are: TreeTop layout annotated by all branches and sub-branches, plus *p*-values of the associated branching points.

## Inputs to treetop: `input_struct`

Data for input into TreeTop is specified via the `input_struct` object. The required fields of `input_struct` are as follows:

- `data_dir` String defining path to directory where input files are stored
- `output_dir` String defining path to directory for outputs. This path has the form */parent_folder/output_name*, and that *parent_folder* already exists. The *output_name* subdirectory is then created, and *output_name* is used as a label for TreeTop intermediate and output files.

- `used_markers` Cell array of markers to be used for this run. All must be present in the input files.

One of the following two fields must be defined:

- `filenames` Cell array of fcs filenames, defining input files. All input files should contain the same set of marker names.
- `mat_file` Inputs to TreeTop can alternatively be given via a single mat file, which should contain a struct called `all_struct`, with the fields `all_data`, `all_markers` and `all_labels`. `all_data` is a matrix of marker values, *n_cells* * *n_markers*. `all_markers` is a cell array of strings containing marker names, of length *n_markers*. `all_labels` is a cell array of strings, containing file labels for every cell, of length *n_cells*.

Additional possible inputs are:

- `extra_markers` Cell array of additional markers which are not used in the run, but are of interest, for example for validation. All must be present in the input files.
- `file_annot` Cell array of shorthand labels for fcs filenames. If omitted, values in filenames are used.

To run the following code, please enter the location of the `treetop_data` directory, and where you would like outputs to be saved.

```
% define directory for data
data_dir  = 'X:/Packages/treetop package/treetop_data';
% define directory for outputs
output_dir  = 'X:/Packages/treetop package';

% check these
if ~exist(data_dir, 'dir') | isempty(output_dir)
 error('please define valid data and output directories')
end

% define input_struct
input_struct  = struct( ...
 'data_dir',     fullfile(data_dir, 'thymus'), ...
 'output_dir',    fullfile(output_dir, 'thymus'), ...
 'used_markers',   {{'DC02', 'DC03', 'DC04'}}, ...
 'extra_markers',
 {{'CD27', 'CD4', 'CD5', 'CD127', 'CD44', 'CD69', 'CD117', 'CD62L', 'CD24', 'CD3',

  'CD8', 'CD25', 'TCRb', 'BCL11b', 'CD11b', 'CD11c', 'CD161', 'CD19', 'CD38', 'CD4

  'CD90', 'Foxp3', 'GATA3', 'IA', 'Notch1', 'Notch3', 'RORg', 'Runx1', 'TCRgd', 'k
 'filenames',    {{'wishbone thymus 2.fcs'}}, ...
 'file_annot',    {{'thymus'}}, ...
 'used_cofactor',   0, ...
 'extra_cofactor',   0 ...
 );
```

## Inputs to treetop: `options_struct`

Options for running TreeTop are specified via the `options_struct` object. The fields of `options_struct` are as follows:

- `sigma` Bandwidth used for calculating cell density values. Once TreeTop has been run, the png file *output_name* density distribution.png can be used as a diagnostic to check whether the value of sigma is appropriate.

The following options are optional

- `metric_name` Distance metric to be used. Valid options are 'L1', 'L2', 'angle'; if omitted, default value is 'L1'.
- `n_ref_cells` Number of reference nodes used by TreeTop; if omitted, default value is 200.
- `n_trees` Number of trees sampled by TreeTop; if omitted, default value is 1000.
- `outlier` Quantile of density distribution below which a cell is regarded as an outlier and excluded; if omitted, default value is 0.01.
- `threshold` Quantile of density distribution above which a cell is regarded as high density, and subject to downsampling; if omitted, default value is 0.5.
- `used_cofactor` Cofactor for calculating arcsinh for `used_markers`. Value of 0 results in no arcsinh transformation being performed. If omitted, default value is 5. (See https://my.vanderbilt.edu/irish-lab/protocols/scales-and-transformation/ for use of cofactor.)
- `extra_cofactor` Cofactor for calculating arcsinh for `extra_markers`. Value of 0 results in no arcsinh transformation being performed. If omitted, default value is 5.
- `file_ext` Specifies format of output image files. Valid values are 'png', 'eps' and 'pdf'; if omitted, default value is 'png'.

```
% define options_struct
options_struct  = struct( ...
 'metric_name',    'L1', ...
 'n_ref_cells',   200, ...
 'n_trees',    1000, ...
 'outlier',   0.01, ...
 'threshold',    0.2, ...
 'sigma',   1e-4, ...
 'file_ext',    'png' ...
 );
```

## Before running TreeTop

Before running TreeTop, we can first examine the distributions of the input markers, and information shared between them.

```
treetop_pre_run(input_struct, options_struct)
```

## Running TreeTop

To run TreeTop, we first initialize a pool, then call `treetop`.

```
pool_obj = gcp('nocreate');
if numel(pool_obj) == 0
 parpool(4)
end
treetop(input_struct, options_struct)
```

## Running TreeTop recursively

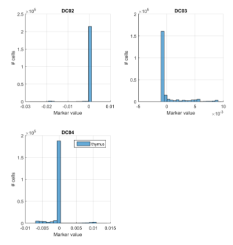
If a branch has been identified at the top level, then users might want also to run TreeTop recursively. In the case of the thymus data, no further branch points are identified.

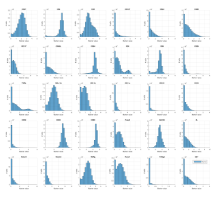
```
treetop_recursive(input_struct, options_struct)
```

## Interpreting TreeTop results

TreeTop produces plots to help you interpret your data. Many of these are based on the TreeTop layout, which is the output of a force-directed graph layout algorithm using the ensemble of trees as an input.

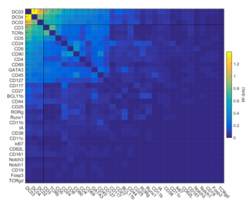
**Outputs from `treetop_pre_run`:**

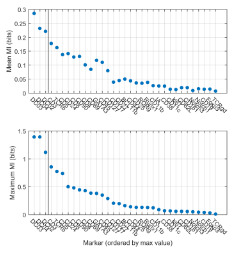




- *[output_name] used marginals.png*
- *[output_name] extra marginals.png*

Plots of marginal distributions of all markers, split by input file.

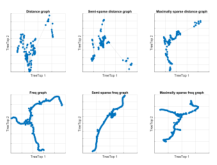




- *[output_name] MI matrix.png*
- *[output_name] max MI.png*

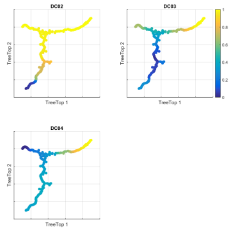
Heatmap of mutual information (MI) between each pair of markers, and also of the maximum MI observed across all other markers. High MI between two markers indicates that they share information, and therefore might be involved in the same process. For each marker, the maximum MI with all other markers is shown; markers with low maximum MI share little information with any other marker, and can be considered for exclusion to improve signal in the data.

**Outputs from `treetop`:**



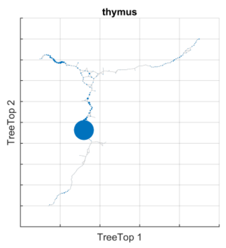
- *[output_name] all layouts.png*

This shows all the possible graph layouts which could be used. This can be helpful if the graph for the default layout is excessively sparse.



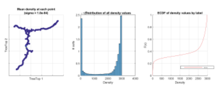
- *[output_name] used marker values.png*
- *[output_name] extra marker values.png*

TreeTop layout annotated by input (and extra) markers. This can assist with understanding what processes are taking place in the data. Each subplot shows all reference nodes, coloured by the mean value of that subplot's marker across all cells allocated to this reference node. The values are scaled to be from 0 to 1. Only the colours vary between the subplots. The edges connecting the points are derived from the ensemble of trees learned by TreeTop, and show the stronger connections between reference nodes.



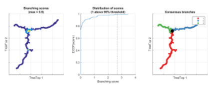
- *[output_name] all samples.png*
- *[output_name] largest sample.png*

TreeTop layout annotated by the input files. The labelling of the files is not used as an input to TreeTop. The *all samples* file shows a separate subplot for each input file. The layout and edges are as for the marker values plots. The size of each reference node is proportional to the number of cells from a given input file allocated to that node. Nodes with zero cells from a file allocated to them are omitted. The 'largest sample' file annotates each node with the file which has the largest proportion at that node. So for example, if there are two files, and 2% of file 1 is allocated to a node, and 1% of file 2 is allocated there, then this node would be coloured as for file 1. Note that file 2 could have more cells allocated to the node in absolute numbers. (Note that if there is only one sample, only the *all samples* file is produced.)



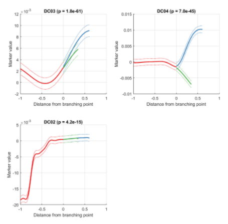
- *[output_name] density distribution.png*

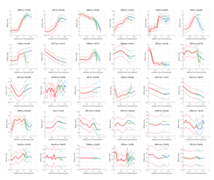
The first subplot shows the TreeTop layout annotated by the mean density value at each reference node. The second subplot shows the distribution of all density values calculated. This can be used as a diagnostic for determining appropriate values of sigma to use: if the distribution is concentrated at 0, then the bandwidth (`sigma`) is too small; if the distribution is concentrated at 1, then the bandwidth is too large, and the neighbourhood of every cell is effectively the whole dataset.



- *[output_name] branching outputs.png*

The first subplot shows the TreeTop layout annotated by the branching score calculated for each reference node. The second subplot shows the empirical cumulative distribution function of the values displayed in subplot 1. The vertical dotted line shows the 95th percentile for the corresponding null distribution. Any values to the right of this line are significant branching points. Subplot 3 shows the branches associated with the point with the highest branching score, and the branch point itself (in black). Subplot 4 compares the branches identified with the labels from the input files; the total of each column is 1. The normalized mutual information (NMI) is shown above; this is 1 when each branch matches to one and only one input file.

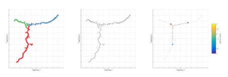




- *[output_name] used marker profiles.png*
- *[output_name] extra marker profiles.png*

These plots show the evolution of each marker as identified branches are traversed through the branching point. The *x* axis shows the mean distance from the branch point to a given point, over the ensemble of trees. The *y* axis shows the marker value at each point on the branch. This line is a Gaussian process fit to the marker values for each reference node, using the mean tree distance as the predictor. The dotted lines show the standard deviation of the Gaussian process around the mean (the solid line). The branch with the largest number of points is put on the left hand side (i.e. arbitrarily given negative distance from the branching point).

**Outputs from `treetop_recursive`:**



- *[output_name] recursive branches.png*

These plots show all branches and sub-branches identified by recursive application of TreeTop. Subplot 1 shows TreeTop layout annotated by all sub-branches, with distinct colours for each sub-branch; subplot 2 shows the same information, however annotated with the branch structure, e.g. a point annotated as 1_2_1 corresponds to sub-sub-branch 1 of sub-branch 2 of branch 1. Subplot 3 shows the individual branch points resulting in each sub-branch. connected according to how they were identifed. The colour corresponds to -log10(*p*), where *p* is the *p*-value for a given branch point.

# Replicating outputs from TreeTop paper

If you have downloaded the folder *treetop_data*, you can reproduce the results shown in the paper. To do this you need to specify a value for `run_switch`, as follows:

1. T cell thymic maturation data, preprocessed with diffusion maps
2. Healthy human bone marrow, mass cytometry data
3. Hierarchically branching synthetic data
4. Linear synthetic data
5. Gaussian synthetic data
6. Circular synthetic data
7. Triangular synthetic data
8. B cell maturation
9. Healthy human bone marrow, scRNAseq data

```
run_switch  = 2;
treetop_example_runs(data_dir, output_dir, run_switch)
```

*Published with MATLAB® R2015b*