SPADE: Spanning Tree Progression of Density Normalized Events

Michael Linderman, Erin Simonds, Zach Bjornson, Peng Qiu

April 16, 2015

michael.d.linderman@gmail.com

Contents

1 Licensing

SPADE is licensed under the GPL-2 (or later) and you are free to use and redistribute this software under the terms of that license. However, we ask you to cite the following paper if you use this software for publication.

1. Qiu, P., Simonds, E. F., Bendall, S. C., Gibbs, K. D., Bruggner, R., Linderman, M. D., Sachs, K., Nolan, G. P., Plevritis, S. K. (2011). Extracting a cellular hierarchy from high-dimensional cytometry data with SPADE. *Nature Biotechnology* 29, 886-891 (2011).

2 Overview

Recent advances in flow cytometry, specifically mass cytometry Bendall et al. (2011), enable simultaneous single-cell measurement of 30+ surface surface intracellular proteins. With such tools it is possible to measure, in a single experiment, enough markers to identify and compare functional immune activities across nearly all cell types in the human hematopoietic lineage. However practical approaches to analyze and visualize data at this scale were not previously available.

SPADE Qiu et al. (2011), spanning tree progression of density normalized events, is an algorithm to organize cells into hierarchies of related phenotypes. These hierarchies, or tree structures, facilitate visualization of developmental lineages, identification of rare cell types, and comparison of functional markers across cell types and stimuli.

This package provides a performant implementation of SPADE, and an accompanying plugin for Cytoscape that provides a GUI for visualizing SPADE trees in the context of the underlying cytometry data.

3 Installation

SPADE is distributed via bioconductor and can be installed via the `biocLite` script. The **SPADE** source package should build on any platform with a modern C++ compiler (and on Windows with `Rtools`). On Linux/Unix/OSX platforms with a C++ compiler that supports OpenMP, **SPADE** will be built with support for multicore and shared memory multi-processor systems. By default, OpenMP is disabled when building on Windows platforms due to difficulties with the pthreads compatibility DLLs.

Those users wishing to enable OpenMP support on Windows will need to modify `src/Makevars.win` to include the following compiler flags:

```
PKG_CXXFLAGS=-fopenmp
PKG_LIBS=-mthreads -lgomp -lpthread
```

and ensure that the necessary pthreads compatibility libraries are installed. These libraries can be downloaded for 32-bit Windows installations at `http://sourceware.org/pthreads-win32`.

SPADE can be used as a stand-alone package, or with an accompanying Cytoscape plugin termed "CytoSPADE". Cytoscape is a free and open-source network visualization tool that can be downloaded at `http://cytoscape.org`. After installing Cytoscape, the CytoSPADE plugin can be installed via the `SPADE.installPlugin` function included in the **SPADE** package. It takes a single argument with the path the Cytoscape install directory, for example on OSX,

```
SPADE.installPlugin("/Applications/Cytoscape_v2.8.1/")
```

would install the plugin.

Once installed, CytoSPADE can be invoked from Cytoscape's Plugins menu. The plugin helps users setup SPADE analyses, and once those analyses are completed, to interactively view the trees produced by SPADE in the context of the underlying flow cytometry data. Figure 3 shows a screenshot of the plugin in use. The plugin links the traditional biaxial plot, familiar to flow cytometry practioners, with the trees

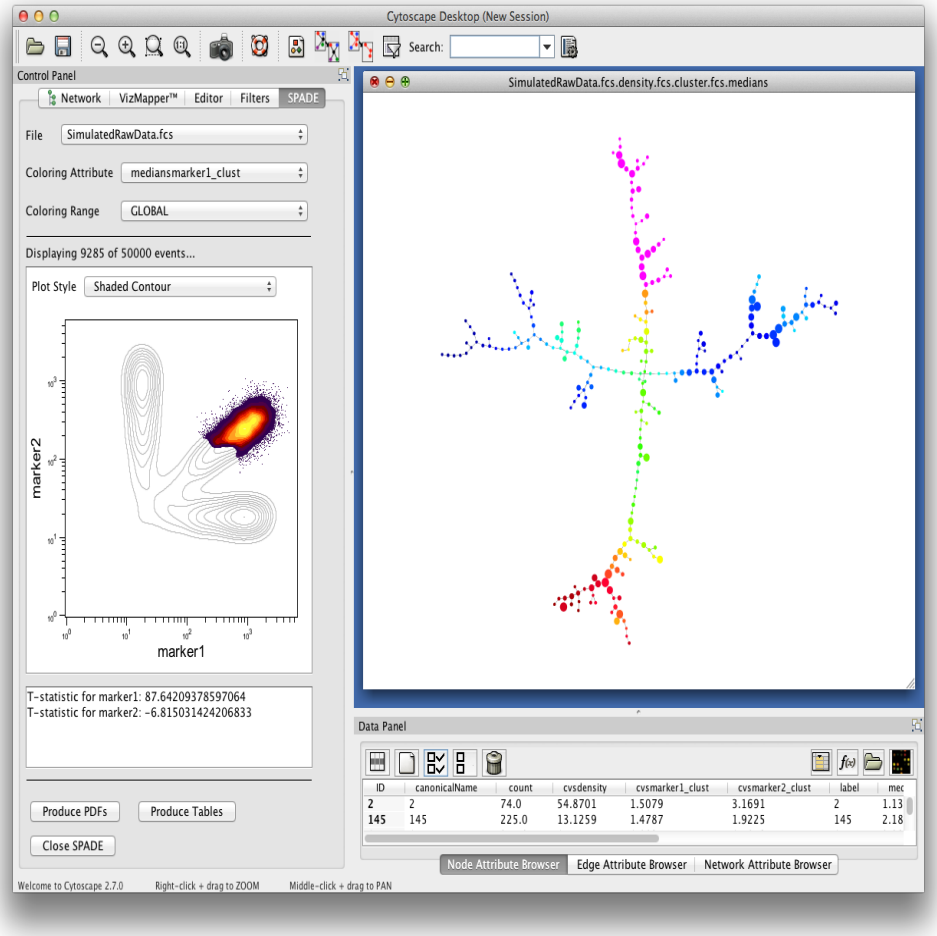


Figure 1: Screenshot of CytoSPADE Cytoscape plugin. Note that the biaxial plot (left) highlights those cells belonging to the nodes selected in the tree view (right).

produced by **SPADE**. Selecting nodes in the tree view, for instance, will show just the cells that belong to those nodes in the biaxial plot. Note that to use the **SPADE** R package from within Cytoscape `Rscript` must be in your path.

4 Example

To demonstrate the functionality of **SPADE**, we use `SimulatedRawData.fcs`, a simple two-parameter synthetic dataset included in the `extdata` directory of the package.

```
> library(spade)
```

```
> data_file_path = system.file(file.path("extdata","SimulatedRawData.fcs"), package = "spade")
```

The example data, shown in Figure 4, has two parameters with (in `arcsinh` space) a very apparent "trident" structure. We would like **SPADE** to be able to recover that structure from the data, and in particular capture the rare population at the "base" of the trident and the transitional populations.

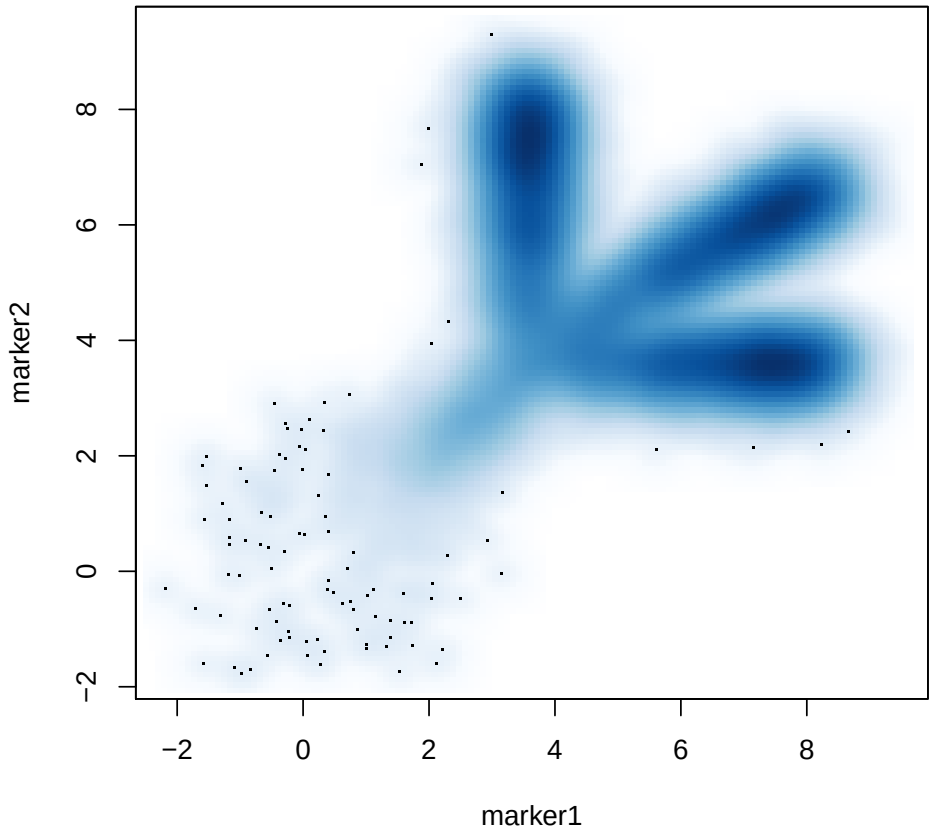


Figure 2: Biaxial plot of the two markers in the sample FCS file transformed into `arcsinh` space. Note the varying population densities, and in particular the rare population where both markers are "low".

The typical entry point to **SPADE** is the driver function. The driver orchestrates the four steps of the SPADE workflow:

1. per-FCS file density dependent downsampling to create uniform cellular density and better capture rare populations
2. across-FCS files unsupervised agglomerative hierarchical clustering to identify distinct sub-populations
3. build a minimum spanning tree that links those clusters
4. per-FCS files upsampling to assign cells to those clusters

and provides useful defaults for **SPADE's** many control parameters.

Basic use of the driver with the sample FCS file would look like the following. In this example we are clustering on the two parameters `marker1` and `marker2` shown in Figure 4 and are transforming the data with the `arcsinh` transform supplied in the **flowCore** package. Numerous other transforms are provided by **flowCore**.

More typically one clusters on a subset of the parameters, such as just surface markers, and then annotates the tree with medians and fold change values for the other, potentially functional, markers. Also more typically, one works with many FCS files. The driver will also accept a vector of files, or a directory containing the FCS files to be analyzed (the latter is the most common usage).

```
> output_dir <- tempdir()
> transforms <- c(`marker1`=flowCore::arcsinhTransform(a=0, b=0.2), `marker2`=flowCore::arcsinhTransform
> SPADE.driver(data_file_path, out_dir=output_dir, transforms=transforms, cluster_cols=c("marker1","mark

  Estimated downsampling-I progress:  0% ...
  Estimated downsampling-I progress: 100% ...
```

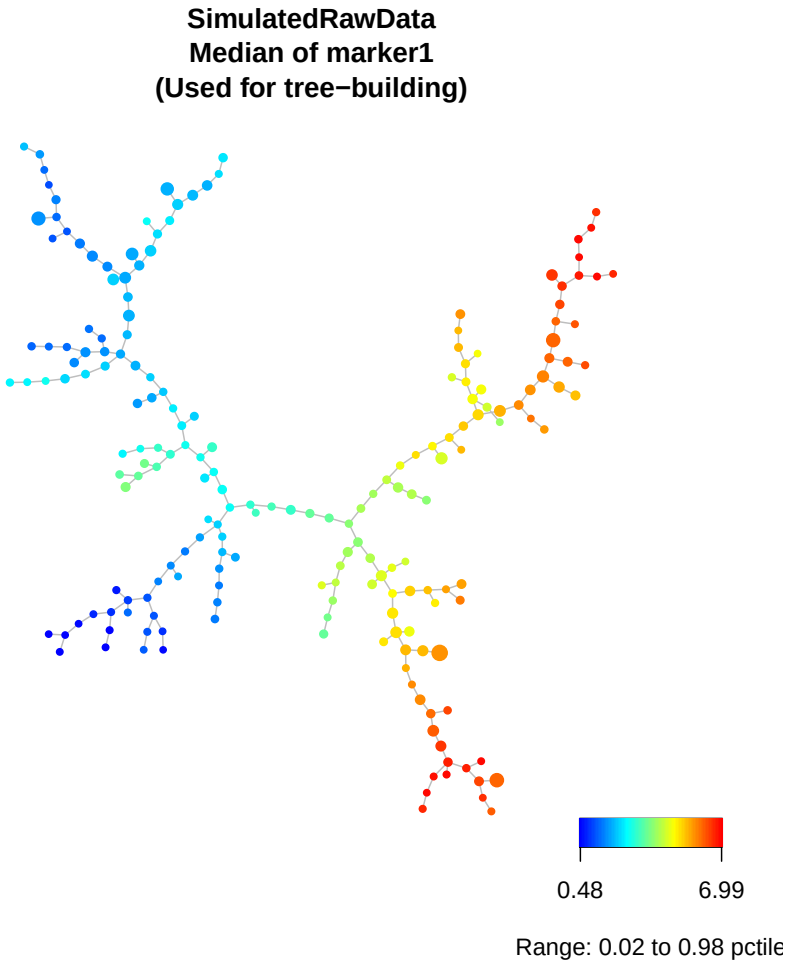
SPADE produces a number of files in the `output` directory including annotated tree structures for each FCS file. These trees are saved as GML files that can be opened by a number of graph visualization packages, including Cytoscape.

```
> grep("^libloc",dir(output_dir),invert=TRUE,value=TRUE)

 [1] "SimulatedRawData.fcs.density.fcs"
 [2] "SimulatedRawData.fcs.density.fcs.cluster.fcs"
 [3] "SimulatedRawData.fcs.density.fcs.cluster.fcs.anno.Rsave"
 [4] "SimulatedRawData.fcs.density.fcs.cluster.fcs.medians.gml"
 [5] "SimulatedRawData.fcs.downsample.fcs"
 [6] "clusters.fcs"
 [7] "clusters.table"
 [8] "global_boundaries.table"
 [9] "layout.table"
[10] "merge_order.txt"
[11] "mst.gml"
[12] "tables"
```

It is often helpful to visualize the results directly. **SPADE** includes functions for plotting all possible trees as PDFs.

```
> mst_graph <- igraph::read.graph(paste(output_dir,"mst.gml",sep=.Platform$file.sep),format="gml")
> layout    <- read.table(paste(output_dir,"layout.table",sep=.Platform$file.sep))
> SPADE.plot.trees(mst_graph, output_dir, out_dir=paste(output_dir,"pdf",sep=.Platform$file.sep), layout
```

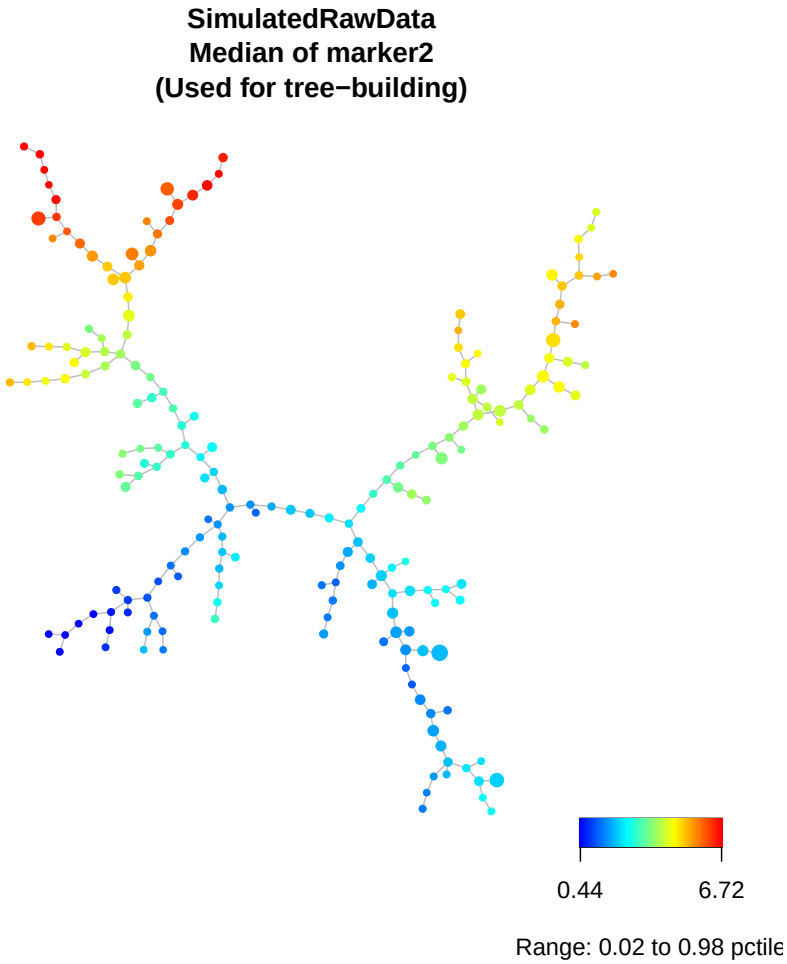


The number of plots is the product of the number of FCS files and number of annotated markers, i.e., there is a plot for each FCS file and each annotation parameters (median or fold change). In this simple case there is only one file, and thus we cannot compute any fold changes. We can however look at per-cluster medians of `marker1` and `marker2` (that is the median value for that marker for all cells assigned to a particular cluster).

Although not immediately obvious, after looking for a moment we can see the "trident" structure in these plots. Specifically we see the three branches where one or both of `marker1` and `marker2` are "high" and the rare populate where both markers are "low". Further we can see the transitional populations between the base and tips of the trident.

This simple example is intended to show how **SPADE** can successfully recover the underlying "structure" of our data – the "trident". In the immunological context, we seek to recover the structure of differentiation of immune cells, and in particular the hematopoietic lineage. We exploit the fact that "parents" and "children" in that differentiation process are close to each other in surface marker space, and further, are linked to together by transitional populations. By over-clustering and then linking those clusters we can successfully recover the structure of the differentiation process and then use that structure as a scaffolding on which to overlay phenotypic data of interest.

In this case we are viewing the median value of the two surface markers overlaid on the entire structure of the data. Additional markers, or fold-change of markers, can be viewed similarly.


SimulatedRawData
Median of marker2
(Used for tree−building)
0.44
6.72
Range: 0.02 to 0.98 pctile

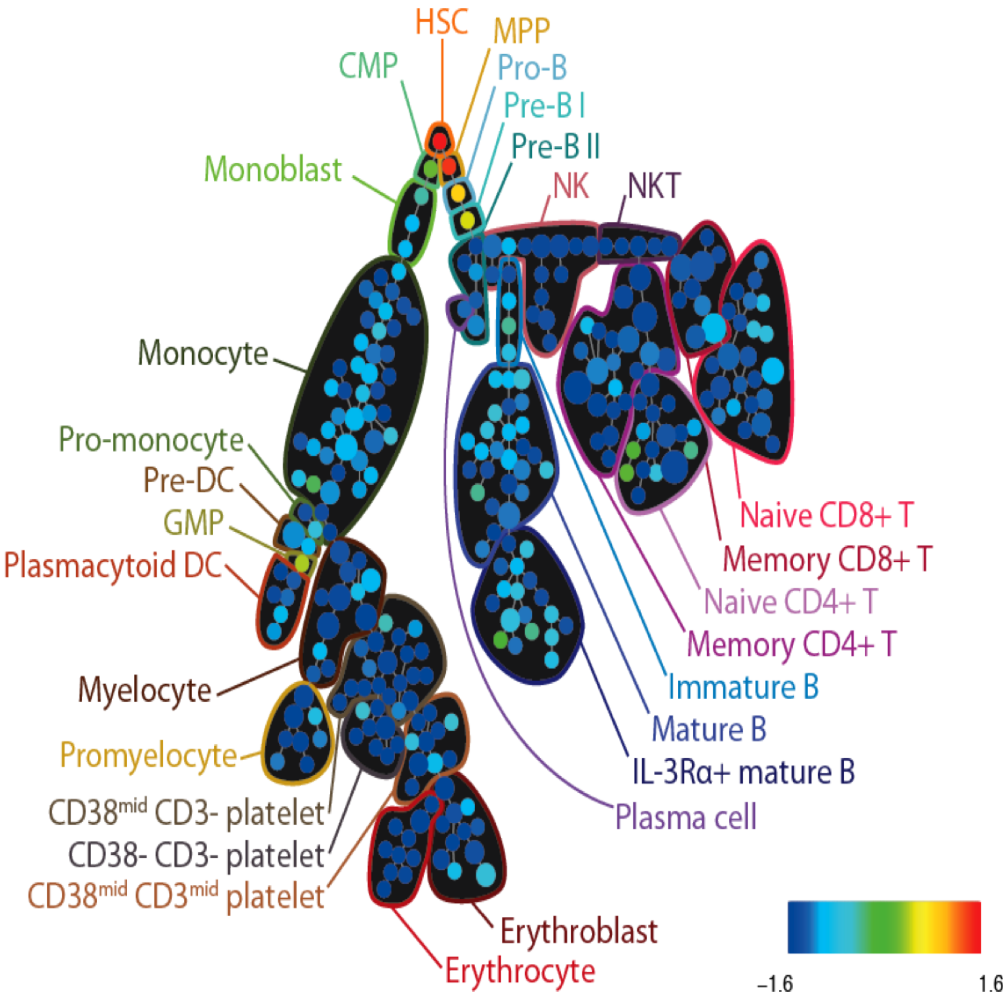


Figure 3: SPADE tree annotated with CD34 expression in healthy human bone marrow (PBMC) samples analyzed via mass cytometry. Adapted from Bendall et al. (2011).

As a more complex example, Figure 4 shows the result from an actual SPADE analysis of 30+ parameter flow cytometry data collected with the CyToF mass cytometer performed for Bendall et al. (2011) (that is not generated as part of this vignette). Cells were clustered using 13 surface markers. Putative cell populations were annotated manually and the nodes moved on the page to better resemble textbook drawings of the hematopoietic lineage. However, the trees themselves, that is the nodes and edges, were produced automatically and unsupervised by the SPADE algorithm. The tree in Figure 4 is annotated with median expression of CD34, which as expected is "high" for hematopoietic stem cells (HSCs). This tree shows how SPADE can recover the structure of hematopoietic lineage, including rare populations such as HSCs, and how that structure can be used as a scaffold on which additional data is overlaid.

This example workflow gives a brief introduction to using **SPADE**. The interested user is pointed to Qiu et al. (2011), and in particular it supplemental material, as well as Bendall et al. (2011) for more detail about the SPADE algorithm and how it can be used to analyze flow cytometry data. Additional documentation for SPADE (including a wiki), the source code, bug tracker and other material can be found at `http://cytospade.org`.

References

Sean C. Bendall, Erin F. Simonds, Peng Qiu, El-ad D. Amir, Peter O. Krutzik, Rachel Finck, Robert V. Bruggner, Rachel Melamed, Angelica Trejo, Olga I. Ornatsky, Robert S. Balderas, Sylvia K. Plevritis, Karen Sachs, Dana Peer, Scott D. Tanner, and Garry P. Nolan. Single-cell mass cytometry of differential immune and drug responses across a human hematopoietic continuum. *Science*, 332(6030):687–696, 2011. doi: 10.1126/science.1198704. URL `http://www.sciencemag.org/content/332/6030/687.abstract`.

P. Qiu, E. F. Simonds, S. C. Bendall, K. D. Gibbs, R. V. Bruggner, M. D. Linderman, K. Sachs, G. P. Nolan, and S. K. Plevritis. Extracting a cellular hierarchy from high-dimensional cytometry data with spade. *Nat Biotechnol*, 29(10):886–91, 2011. URL `http://www.ncbi.nlm.nih.gov/pubmed/21964415`.