# The FIt-SNE FlowJo Plugin

## Introduction

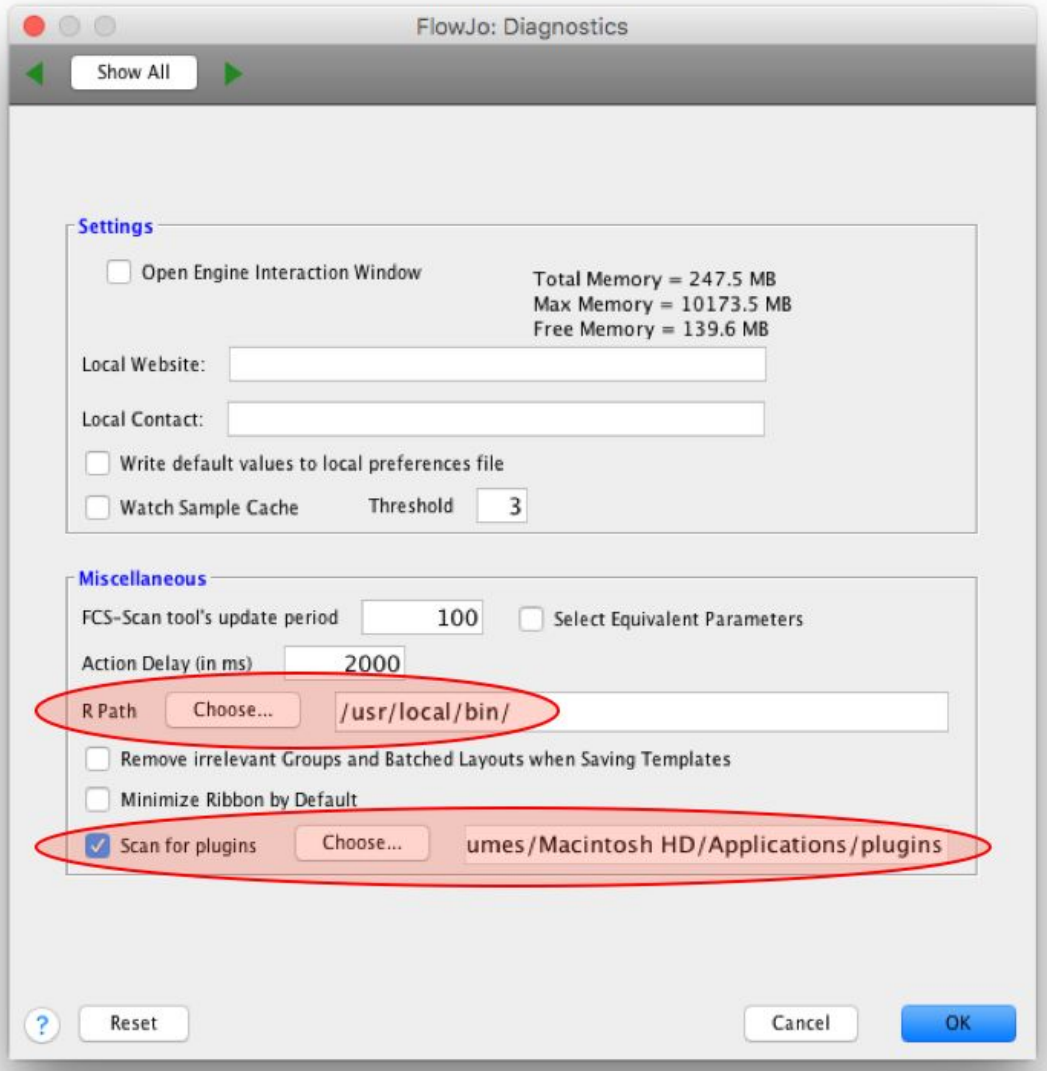
t-SNE is a well known and useful method for dimensionality reduction and visualization of flow cytometry and single cell sequencing data. However the original implementation scaled very poorly for large datasets. This has been somewhat addressed by the Barnes-Hut implementation of t-SNE, which made t-SNE applicable to typical flow cytometric data. Still, downsampling was often required since Barnes-Hut t-SNE does not scale well to hundreds of thousands or millions of high dimensional data-points. Recently, Linderman et. al developed a new Fast Fourier Transform-accelerated Interpolation-based t-SNE (FIt-SNE), which further accelerates the computation of t-SNE. In addition, Linderman et. al introduced the option of calculating input similarities in high dimensions using multi-threaded approximate nearest neighbors instead of a typical vantage point tree implementation. Fun fact: this is done using a library developed by Spotify to identify and suggest songs that listeners may like based on what they are commonly listening to. Finally, an option of a late exaggeration was added tp allows for easier identification of clusters in the resulting t-SNE embeddings.

The FIt-SNE algorithm has been implemented as a plugin compatible with both FlowJo and SeqGeq analysis programs. As a result, this plugin will produce new derived parameter(s) capturing the resulting low dimensional t-SNE embeddings.

## Installation

In order to install the FIt-SNE plugin, you will need to

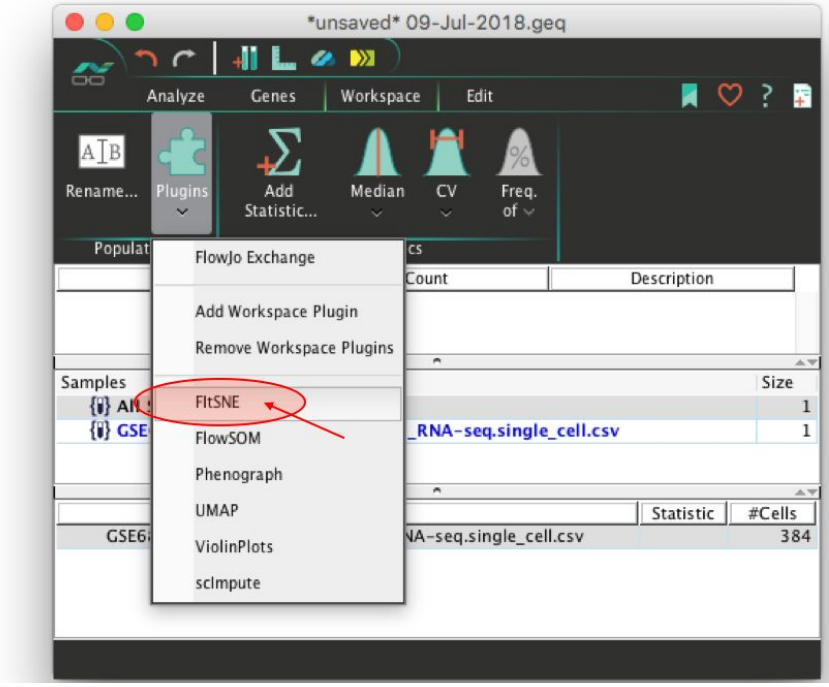
1. Place the plugin jar file in your Plugins folder, and direct FlowJo or SeqGeq to that folder using the Diagnostics section of the preference.
2. Make sure you have R installed and the R path is specified in the R Path field of the Diagnostics section of the preferences; see docs.flowjo.com/d2/plugins/installing-plugins/ for more details.
3. Restart the (FlowJo or SeqGeq) application to pick up the new plugin.

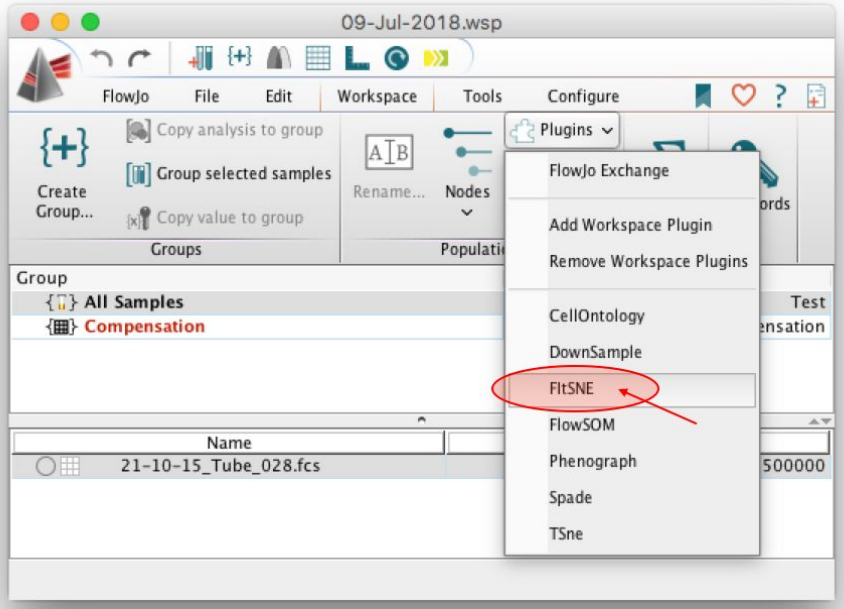



# Usage

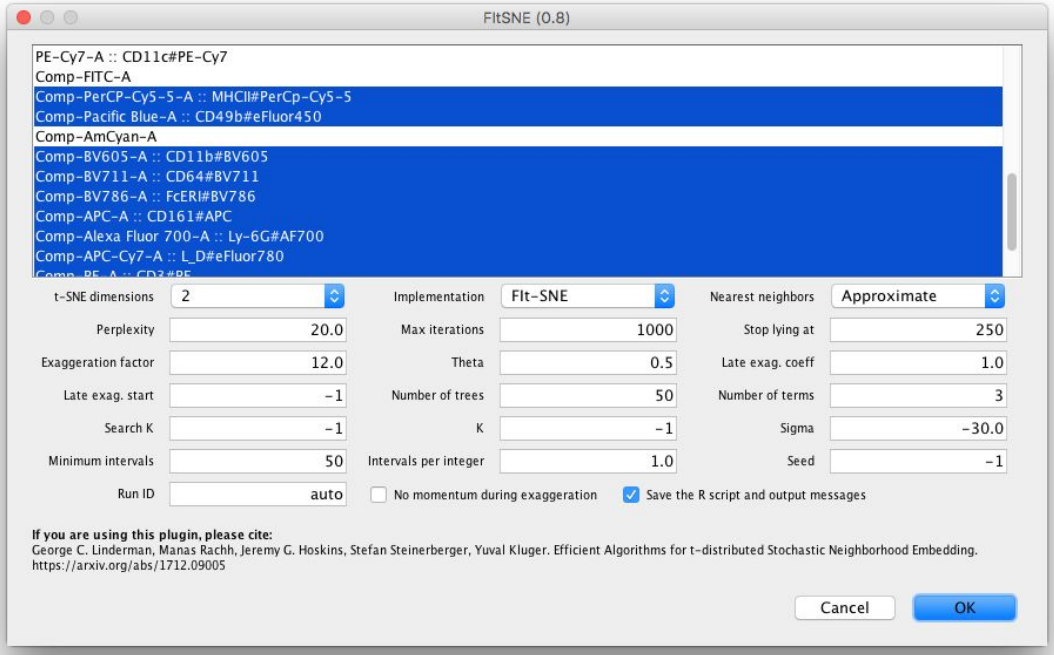
To run the FitSNE plugin on a population (or sample),

1. Select the population of interest within the workspace
2. Go to the Workspace tab and select the **FitSNE** option from within the Plugins dropdown there. Note that plugin will be unavailable (greyed out) if no population is selected.




3. Select the parameters and settings that you would like to use to run the FIt-SNE algorithm within the resulting plugin dialog. Those settings are described further below.




4. Once the plugin has finished calculating the dimensionally reduced parameters they will become available within the data matrix as a set of derived or analytic parameters which are meant to describe the variance in populations over the whole latent space of N-Dim parameters, from the raw data matrix

# Input parameter description

To find out more regarding what each setting in FIt-SNE adjusts, mouse over those options within the plugin interface. Please note that several of those settings are rather advanced and in order to use those effectively, you may have to review the FIt-SNE paper to get an understanding of the algorithm.

## Parameter selector

Which input parameters, such as FCS channels in FlowJo, principal components or genes or transcripts in SeqGeq, do you want to run the algorithm on.

## t-SNE dimensions

You can request the algorithm to calculate 1 or 2 dimensional embedding, i.e., it will generate 1 or 2 derived parameters; 2 by default.

## Implementation

You can select between FIt-SNE and Barnes-Hut tSNE. FIt-SNE is the default; Barnes-Hut tSNE is provided only for comparison.

## Nearest neighbors

How should one calculate the nearest neighbors that each data point (cell) is attracted to? This can be done either by using an approximate nearest neighbors implementation using the Spotify Annoy library (https://github.com/spotify/annoy) or using an exact nearest neighbors implementation with vantage point trees. If the input dimensionality (=number of selected parameters) is relatively small (e.g.., flow cytometry input, or a “few” principal components from single cell RNA seq data) then the exact implementation should be almost as fast as the approximate one, but for high-dimensional (~30+) inputs, the approximate nearest neighbors will perform significantly faster.

## Perplexity

Perplexity is a measure for information that is defined as 2 to the power of the Shannon entropy. Same as in regular t-SNE, the perplexity may be viewed as a knob that sets the number of effective nearest neighbors that each cell is being attracted by (while also being repulsed by all of the neighbors). In practical terms, if you are getting a strange ‘ball’ with uniformly distributed points, it usually indicates that your perplexity way too high because all points want to be equidistant.
If perplexity is set to a negative value then K (kernel width; number of nearest neighbors) and sigma (bandwidth) parameters will be used instead of perplexity.

## Max iterations

The maximum number of iterations that the algorithm will perform.

## Stop lying at

Is the Iteration after which the early exaggeration stops and perplexities are no longer exaggerated by the exaggeration factor, i.e., this value controls the length of the early exaggeration phase. Increasing the early exaggeration may be useful for large numbers of events; however, be careful not to increase both the exaggeration and exaggeration factor too much, or nearest neighbors calculations may tend to over-cluster.

## Exaggeration factor

Exaggeration factor used to multiply the attractive forces with during the early exaggeration phase of the calculation. This also controls how tight natural clusters in the original space are in the embedded space and how much space will be between them. If you think about clustered data, the purpose of early exaggeration is to make it so that the cells corresponding to a cluster can all "find each other" despite the repulsion from so many other cells.

## Theta

Theta controls the estimation of forces for the Barnes Hut algorithm (it is only applicable to the Barnes Hut implementation). Smaller values will lead to more precise (and computationally more expensive implementation); a value of 0 will effectively turn off Barnes Hut and fall back to the original “precise” t-SNE implementation. See also jheer.github.io/barnes-hut/ for an explanation of how the choice of theta effects the algorithm.

## Late exag. coeff

Late exaggeration coefficient used to multiply the attractive forces with during the late exaggeration phase, i.e., after the late exaggeration start. Late exaggeration will typically cause the clusters in the resulting embedded space to become tighter and better separated from one another, but you may start losing local structures within those clusters.

## Late exag. start

This is the Iteration at which the late exaggeration starts and attractive forces are again exaggerated, this time by the late exaggeration coefficient. The default value of -1 will disable the late exaggeration.

## Number of trees

Number of trees used in the Spotify Annoy library to perform approximate nearest neighbor search. This affects the build time and the size of the index used for nearest neighbor search; a larger value will give more accurate nearest neighbors, but will consume more memory and take more time. See github.com/spotify/annoy for more details.

## Search K

Search K used in the Spotify Annoy library to perform approximate nearest neighbor search. A value of -1 will set this automatically. This also affects the nearest neighbor search performance; a larger value will give more accurate results, but will take longer time to return. See github.com/spotify/annoy for more details.

## Number of terms

FIt-SNE uses an interpolation scheme to approximate the repulsive forces at each step of gradient descent. This parameter corresponds to the number of interpolation nodes. Increasing the number of nodes should increase the accuracy, but this effect will quickly saturate and the Runge phenomenon (en.wikipedia.org/wiki/Runge%27s_phenomenon) takes over because it is an equispaced interpolation. For this reason, it is not recommended to adjust this parameter. Instead, the accuracy of the approximation may be increased by changing the grid using the "minimum intervals" and "intervals per integer" sections.

## Minimum intervals

The minimum number of intervals and the intervals per integer adjust the fineness of the grid used in the FIt-SNE interpolation scheme. The Minimum intervals describes the smallest number of subintervals in which to split each dimension of the t-SNE space when calculating the FIt-SNE. However, as the space increases in size (t-SNE typically expands to be on the order of -100 to 100 by -100 to 100), we need to maintain the fineness of the grid. This is where the Intervals per integer prevail. The number of subintervals in each dimension at any given iteration is the larger of (i) the minimum number of intervals, and (ii) the number of intervals per integer times the current size of the t-SNE grid.

For example, let's say you set the minimum number of intervals to 45 and the intervals per integer to 1.2. Let's say the current t-SNE grid spans from -10 to 10 by -10 to 10, which is a grid of 20 times 20 integers. 20 times 1.2 = 24, which is less than 45 and therefore, the values of 45 (minimum number of intervals) will be used to determine the number of subintervals for FIt-SNE interpolation. Let's say that later, t-SNE has already expanded to -20 to 20 by -20 to 20, which is a grid of 40 times 40. 40 times 1.2 = 48, which is larger than 45 and therefore, the value of 48 will be used to determine the number of subintervals for FIt-SNE interpolation.

## Intervals per integer

The intervals per integer and the minimum number of intervals adjust the fineness of the grid used in the FIt-SNE interpolation scheme. The Minimum intervals describes the smallest number of subintervals in which to split each dimension of the t-SNE space when calculating the FIt-SNE. However, as the space increases in size (t-SNE typically expands to be on the order of -100 to 100 by -100 to 100), we need to maintain the fineness of the grid. This is where the Intervals per integer prevail. The number of subintervals in each dimension at any given iteration is the larger of (i) the minimum number of intervals, and (ii) the number of intervals per integer times the current size of the t-SNE grid. See also the example provided above.

## K

Number of nearest neighbors to control the kernel width; a value of -1 will auto set this parameter. Note that this parameter will be ignored unless perplexity is set to a negative value, i.e., the kernel width can be controlled either by perplexity, or set manually by K and sigma. Same as perplexity, this may be viewed as a knob that sets the number of effective nearest neighbors that each cell is being attracted by.

## Sigma

The bandwidth (standard deviation for Gaussian kernel). This will be ignored unless perplexity is set to a negative value, i.e., the kernel width can be controlled either by perplexity, or set manually by K and sigma. Same as perplexity, this may be viewed as a knob that sets the number of effective nearest neighbors that each cell is being attracted by.

## Seed

FIt-SNE is a stochastic algorithm and as such, slightly different results will be produced every time you run it unless you set seed to a positive value. The default of -1 will set the seed randomly.

## Run ID

An identifier used to create the derived parameter names. The “auto” value will create names encoding most of the input settings, which ensures that they will be unique, but they are also somewhat long due to the number of all the different options provided. You may want to choose your own Run ID to shorten the names.

## No momentum during exaggeration

If checked, then there will be no momentum/gains until after the early exaggeration phase is completed, i.e., a regular gradient descent will be used to optimize the cost function. Gradient descent with momentum is an optimization of a gradient descent algorithm that relies on exponentially decreasing weighted averages applied to older data points and thus typically minimizes oscillation of the gradient descent. You can read more at datalya.com/blog/2017/gradient-descent-with-momentum but this checkbox should have minimal effect on the results.

## Save the R script and output messages

Would you like to save the resulting R script and output messages for debugging and troubleshooting purposes? We suggest that you do so.

## Notes and suggestions

Similar to regular t-SNE, this plugin will typically run well on a condensed matrix of parameters either from flow cytometry directly, or in the case of scRNA-seq data, on principal components. Large numbers of events within the population of interest will be more difficult to cluster than small event numbers. If necessary, the downsample platform can be used to reduce the number of events within a population, in an unbiased manner.

You may also consider reviewing this distill.pub/2016/misread-tsne/ guide to help you understand the different t-SNE parameters.

# Contact

If you have any questions, concerns, or other feedback related to this plugin, please reach out to FlowJo technical support specialists by emailing to techsupport@flowjo.com.

# References:


1. Linderman, G. et al. Efficient Algorithms for t-distributed Stochastic Neighborhood Embedding. *arXiv preprint arXiv:1712.09005.* (2017), https://arxiv.org/abs/1712.09005
2. Linderman, G and Steinerberger, S. Clustering with t-SNE, provably. *arXiv preprint arXiv:1706.02582.* (2017), https://arxiv.org/abs/1706.02582