flowSim introduction

Sebastiano Montante

2023-02-09

This document describes the usage of the main flowSim functions and reports examples of typical flowSim scripts.

Introduction

flowSim is a tool designed to visualize, detect and remove highly redundant information in large FCM training sets, increasing the performance of ML algorithms by reducing overfitting while at the same time decreasing computational time for training. The tool performs near duplicate image detection (NDD) by combining community detection algorithms with the density analysis of the marker expression values.

Installation

The easiest way to install flowSim is directly from github:

```r
library(devtools)
devtools::install_github("semontante/flowSim")
```

The user can also download the package locally and install it from the package folder:

```r
install("path/to/local_dir")
```

Input

The input of flowSim is a directory containing the bivariate marker expression of the images under analysis. In particular, the marker expression of each image must be reported in a csv file whose first and second column report the expression of, respectively, the first and second marker. Here's an example of a correctly formatted csv file:

```
#>   PE.CF594.A FITC.A
#> 1      2.985  3.137
#> 2      2.433  2.752
#> 3      2.942  2.914
#> 4      2.874  3.121
#> 5      1.805  2.119
#> 6      1.925  2.712
#> 7      0.871  2.560
#> 8      2.782  2.836
#> 9      2.645  2.265
```

Note: If the input csv contains more than two columns, the other columns are ignored. For example, if the csv files contain a third column with events membership, this third column will be ignored.

Identification and removal of similar images

There are two main functions to consider in the flowSim package:

1. V1 mode: get_similarity_all_plots() for small datasets (< 500 files).
2. V2 mode: get_similarity_all_plots_v2() for large datasets.

Both functions calculate the pairwise distance matrix of all input files under analysis. They have similar arguments, see help(get_similarity_all_plots_v2) or help(get_similarity_all_plots) to look at the list of possible options.

```r
# Get the directory path where the csv files are stored.

fpath <- system.file("extdata", package="flowSim")

# get similarity scores matrix for all files (V1 mode, < 500 files).
library(parallel)
library(sm)
library(utils)
out<-get_similarity_all_plots(path_dir = fpath)

# Alternatively, it is possible to feed the list of paths (only V1 mode).
paths_plots<-list.files(path = fpath,full.names = T)
out<-get_similarity_all_plots(paths_plots = paths_plots)

# get similarity scores matrix for all files (V2 mode, > 500 files).
library(parallel)
library(sm)
library(doSNOW)
library(foreach)
library(Rtsne)
library(anocva)
library(ggplot2)
library(utils)
library(stats)
out<-get_similarity_all_plots_v2(path_dir = fpath)

# In both modes the user can provide the indices of the paths of the files to analyze.
# It is possible to get these indices by using a dedicated flowSim function
# or by using the user's own code.

inds_selected_files<-get_inds_files_selected(path_dir = fpath,
                                             files_selected = files_to_analyze)

# Where files_to_analyze = vector of files names to analyze.
out<-get_similarity_all_plots_v2(n_samples=inds_selected_files,path_dir = fpath)

# Analyze only 2nd and 3rd file path: based on list.files(fpath,full.names=T)
out<-get_similarity_all_plots_v2(n_samples=c(2,3),path_dir = fpath)
```

Note: get_similarity_all_plots_v2 is designed to be used with large datasets. https://github.com/semontante/flowSim_test_data contains large datasets to test the V2 mode of flowSim.

Based on this distance matrix, a network of nodes is generated (each node is a file under analysis). A community detection algorithm is applied to identify the clusters of similar files:

```r
# Execute igraph clustering.
library(igraph)
library(RColorBrewer)
# louvain clustering (default).
list_output<-gen_igraph_network(similarity_output = out,
                                method_clust = "louvain")
igraph_network<-list_output$igraph_network
partition_vec<-list_output$partition
# generate visnetwork.
library(visNetwork)
set.seed(1234) # set seed to get same groups colors.
visnetdata<-gen_visnetwork_data(igraph_network)
# heterogeneity score (optional).
library(igraph)
heterogeneity_score<-get_heterogeneity_score(igraph_network,partition_vec=partition_vec,
                                             visnetdata = visnetdata)
```

The user can optionally visualize the network:

```r
# plot full network
library(visNetwork)
plot_visnet(visnetdata)
```

The final step is to filter out the similar files and export the final results (the selected files for each cluster). The user has 3 options, it is possible: 1. To get a dataframe reporting the names of the selected files for each group. 2. To export the bivariate plots of selected files for each group. 3. To export the bivariate plots of all files of all groups.

```r
library(sm)
library(parallel)
library(igraph)
library(visNetwork)
library(RColorBrewer)
library(utils)
library(stats)
library(Rtsne)
library(ggplot2)
library(doSNOW)
library(foreach)
library(anocva)
library(graphics)
# The directory, the files paths and the features dataset are needed.

fpath <- system.file("extdata", package="flowSim")
paths_plots<-list.files(path = fpath,full.names = T)
df_features<-import_df_features(paths_plots = paths_plots)

# Execution of the similarity generation step, igraph grouping
# and the visnetwork generation step.
```

```r
out<-get_similarity_all_plots(path_dir = fpath)
list_output<-gen_igraph_network(similarity_output = out,
                                method_clust = "louvain")
igraph_network<-list_output$igraph_network
partition_vec<-list_output$partition
set.seed(1234) # set seed to get same groups colors.
visnetdata<-gen_visnetwork_data(igraph_network)

# (1) get a dataframe reporting the names of the selected files for each group.
info_exporting<-exports_filtered_plots(visnetdata = visnetdata,
                                       df_features = df_features)

# (2) export the bivariate plots of selected files for each group.

df_features<-import_df_features(paths_plots = paths_plots)
info_exporting<-exports_filtered_plots(visnetdata = visnetdata,df_features = df_features,
                                       path_expr_data = fpath,
                                       path_output = "path/to/output_dir")

# Note: the output of V2 mode contains already the df_features.
out<-get_similarity_all_plots_v2(path_dir = fpath) # V2 mode execution.
info_exporting<-exports_filtered_plots(visnetdata = visnetdata,
                                       df_features = out$df_features)

# (3) export the bivariate plots of all files of all groups.
exports_plots(visnetdata = visnetdata,path_expr_data = fpath,
              path_output = "path/to/output_dir")
```

Note: The directory ("fpath""") can contain all files even if the similarity analysis was performed on a subset of files (no need to generate a subset of the directory).

In (2) and (3) the output directory will contain a folder for each group of files.

Visualization options

There are 3 different types of network visualizations. The user can visualize all the input files organized in clusters and all the isolated files ("full network visualization"):

```r
library(visNetwork)
# plot full network.
plot_visnet(visnetdata)
# plot full network setting the size of nodes.
plot_visnet(visnetdata,size_nodes=25)
```

The user can also visualize only a partial area of the network ("partial network visualization"). The isolated files (black nodes) can also be removed. This is particularly useful for large networks.

```r
# plot network without the isolated files.
plot_visnet(visnetdata,remove_single_files = T) # \donotrun.
# plot only specific groups.
plot_visnet(visnetdata,select_group = c("1"))
```

Finally, for very large networks (> 4000 files), it is not possible to plot the full network at once. It is useful in this case to plot a contracted version of the network ("contracted network visualization"), where each node is a group and each edge is proportional to the number of connections between the groups.

```
# generate contracted data.
visnetdata_contracted<-gen_visnetwork_data(igraph_network,contract_net = T) # \donotrun.
# plot contracted network.
plot_visnet(visnetdata_contracted) # \donotrun.
```

Note: Some visualizations options (like the contracted network visualization) are designed to be used with large datasets. https://github.com/semontante/flowSim_test_data contains large datasets to test more complex flowSim visualization options.

Small vs large datasets, parameters tuning

The V1 algorithm is designed to work for small datasets (< 500 files) and it is also the most accurate version.

```
# Get the directory path where the csv files are stored.
fpath <- system.file("extdata", package="flowSim")
# get similarity scores matrix for all files (V1 mode, < 500 files).
# with default parameters.
library(parallel)
library(sm)
library(utils)
out<-get_similarity_all_plots(path_dir = fpath)
```

There are several parameters the user can set to personalize the similarity output. The tolerance parameter influences the number of clusters estimated by the algorithm. A lower tolerance threshold tends to generate a lower number of clusters (intra-cluster differences are more tolerated), while a higher tolerance threshold tends to generate more clusters (intra-cluster differences are less tolerated). The default tolerance threshold for the V1 mode is 0.6.

```
# Set tolerance to 0.7. Default is 0.6.
out<-get_similarity_all_plots(path_dir = fpath, thr_score = 0.7)
```

It is also possible to set the number of cores to use:

```
# Use 4 cores.
out<-get_similarity_all_plots(path_dir = fpath, n_cores=4)
```

The V2 algorithm is designed to work with large datasets. It is faster, it consumes less memory, but it is also less accurate. It has the same parameters of V1 and an additional parameter to control the level of fragmentation of the dataset. As the fragmentation parameter increases, the time of execution and memory load decreases, but the accuracy also decreases. Up to 2000 files the fragmentation parameter is automatically estimated, above 2000 files it needs to be manually set. The default tolerance threshold for the V2 algorithm is 0.9.

```
# dataset broken in 5 fragments.
out<-get_similarity_all_plots_v2(path_dir = fpath, n_batches = 5)
# dataset broken in 100 fragments.
out<-get_similarity_all_plots_v2(path_dir = fpath, n_batches = 100)
```

It is possible to increase the accuracy of the V2 algorithm, by performing several cycles of execution. Each cycle is performed on the files selected from the previous cycle. In this way, the residual homogeneity generated from the previous cycles is progressively reduced.

```r
# Execution of flowSim in V2 mode.
out<-get_similarity_all_plots_v2(path_dir = fpath)

list_output<-gen_igraph_network(similarity_output = out,method_clust = "louvain")
igraph_network<-list_output$igraph_network
partition_vec<-list_output$partition
set.seed(1234) # to get same groups colors.
visnetdata<-gen_visnetwork_data(igraph_network)
info_exporting<-exports_filtered_plots(visnetdata = visnetdata,
                                       df_features = out$df_features)

# The selected files will be the files of the next cycle.
files_new_cycle<-unlist(strsplit(info_exporting$files_filtered,";"))
inds_selected_files<-get_inds_files_selected(path_dir = fpath,
                                             files_selected = files_new_cycle)

# Execution of flowSim on the selected files of the previous cycles.
out<-get_similarity_all_plots_v2(n_samples=inds_selected_files,
                                 path_dir = fpath)
```

Both V1 mode and V2 mode have the same exporting step, which can be regulated by the filtering parameter. The filtering parameter controls how many files are selected from each group. The default value is 2.1. By decreasing this parameter, the number of files selected from each group increases.

```r
# default threshold: 2.1
info_exporting<-exports_filtered_plots(visnetdata = visnetdata,df_features = df_features)
# Filtering threshold set to 1.2. More patterns preserved.
info_exporting<-exports_filtered_plots(visnetdata = visnetdata,df_features = df_features,
                                       filter_thr=1.2)
```