NS-Forest version 2

Brian Aevermann[1], Richard Scheuermann[1]

[1]J. Craig Venter Institute

dx.doi.org/10.17504/protocols.io.un7evhn

Human Cell Atlas Method Development Community

Brian Aevermann

ABSTRACT

NS-Forest is an alogrithm that determines the minimum set of genes that are necessary and sufficient to define a cell type cluster derived from single cell RNAseq expression data.

Development and stable releases can be found at :

https://github.com/JCVenterInstitute/NSForest

THIS PROTOCOL ACCOMPANIES THE FOLLOWING PUBLICATION

Aevermann BD, Novotny M, Bakken T, Miller JA, Diehl AD, Osumi-Sutherland D, Lasken RS, Lein ES, Scheuermann RH. Cell type discovery using single-cell transcriptomics: implications for ontological representation. Hum Mol Genet. 2018 May 1;27(R1):R40-R47. doi: 10.1093/hmg/ddy100.

PROTOCOL STATUS

Working

We use this protocol in our group and it is working

Pre-analysis data preparation

1

1. The script is a Jupyter notebook in python 2.7. Required libraries: Numpy, Pandas, Sklearn, graphviz, numexpr

2. Build a Cell by Gene matrix where the values are either normalized or raw count expression values from a single cell RNAseq experiment (tsv or csv formats work by default); the first column should contain gene IDs; the first row should contain cell IDs that can be user defined

3. Add cluster labels as a column whereby each cell is **uniquely** assigned to a cluster; the header for the cluster label column must be "**Clusters**"; the labels themselves must be strings

4. The gene IDs/ symbols must be stripped of " . ", " - ", and "@", or other troublesome characters (replace with "_" or "at")

Input parameters

2

Step 1: Alter read function in first line of code by adding path to your file:
dataFull = pd.read_table("Your/Path/Here", index_col = 0)
Step 2: Dummy columns created and added to matrix
-These binary columns are used for one vs all Random Forest modeling
Step 3: Generates a matrix of cluster median expression values

Step 4: Finds the number of dummy columns (ie. clusters) and prints that to screen as a sanity check

Step 5: Tunable parameters (code below). Change these as needed.

#Random Forest parameters
rfTrees=10000 #Number of trees
threads=1 #Number of threads to use, -1 is the greedy option where it will take all available CPUs/RAM

#Filtering and ranking of genes from random forest parameters

InformativeGenes = 15 #How many top genes from the Random Forest ranked features will be evaluated for binariness
Genes_to_testing = 6 #How many top genes ranked by binary score will be evaluated in permutations by fbeta-score (as the number increases the number of permutation rises exponentially!)

fbeta-score parameters

#setBeta=0.5

Core algorithm description

3 Part 2: Core algorithm

a) The main loop iteratates through the dummy columns one at a time
b) Generates a Random Forest model
c) Extracts and sorts the feature variables by importance (first ranked gene list)
d) Computes Binary score for top 15 genes
e) Re-sorts based upon Binary score, then importance (Second ranked gene list)
f) Top 6 genes are then used to produce individual decision trees to find optimal expression threshold cutoffs
g) The top 6 genes are then permuted using AND logic and the expression threshold cutoffs determined in step f; for each combination of genes, the fbeta-score is computed and stored

Results reporting

4 Part 3: Reporting

There are four reports that are generated by NS-Forest:

1. Binary ranking table: contains binary score and Gini information from the Random Forest model for each gene used in combinations
2. Complete results: the genes, number of markers, and fbeta-score for all tested combinations
3. Top f-beta results (no genes): cluster names and their maximum fbeta-score
4. Top markers results for each clusters (usually multiple): a subset of the complete results; this utilizes a rank that accounts for ties in order to give a more manageable output