NIFA: Non-negative Independent Factor Analysis disentangles discrete and continuous sources of variation in scRNA-seq data

February 01, 2022

Contents

This walkthrough provides a quick start guide for using **NIFA**. In this walkthrough we start from a publicly available scRNA-seq dataset **SimKumar4easy**. This simulated scRNA-seq dataset is available via the bioconductor package *DuoClustering2018*. We first perform data normalization as described in the manuscript, then we apply **NIFA** and compare the the latent factors with the cell labels.

We first load the data **SimKumar4easy** from the bioconductor package *DuoClustering2018*.

```r
if (!require("DuoClustering2018", character.only = T, warn.conflicts = F,
    quietly = T)) {
    BiocManager::install("DuoClustering2018")
}
library(DuoClustering2018)

library(rsvd)
library(pheatmap)
library(ggpubr)
```

rsvd, *pheatmap* and *ggpubr* are loaded only for the purpose of data normalization and visualization. We also load the **NIFA** package into the current working environment.

```r
if (!require("NIFA", character.only = T, warn.conflicts = F, quietly = T)) {
    devtools::install_github("wgmao/NIFA")
}
library(NIFA)
```

Data Normalization

We incorporate one additional function `normalize_barcode_sums_to_median()` credited to https://github.com/hb-gitified/cellrangerRkit/blob/master/R/normalize.r.

```r
normalize_barcode_sums_to_median <- function(gbm) {
    bc_sums <- colSums(gbm)
    median_sum <- median(bc_sums)
    return(sweep(gbm, 2, median_sum/bc_sums, "*"))
}  #normalize_barcode_sums_to_median
```

We load the dataset **SimKumar4easy** and extract the scRNA-seq matrix (gene-by-cell) and cell labels.

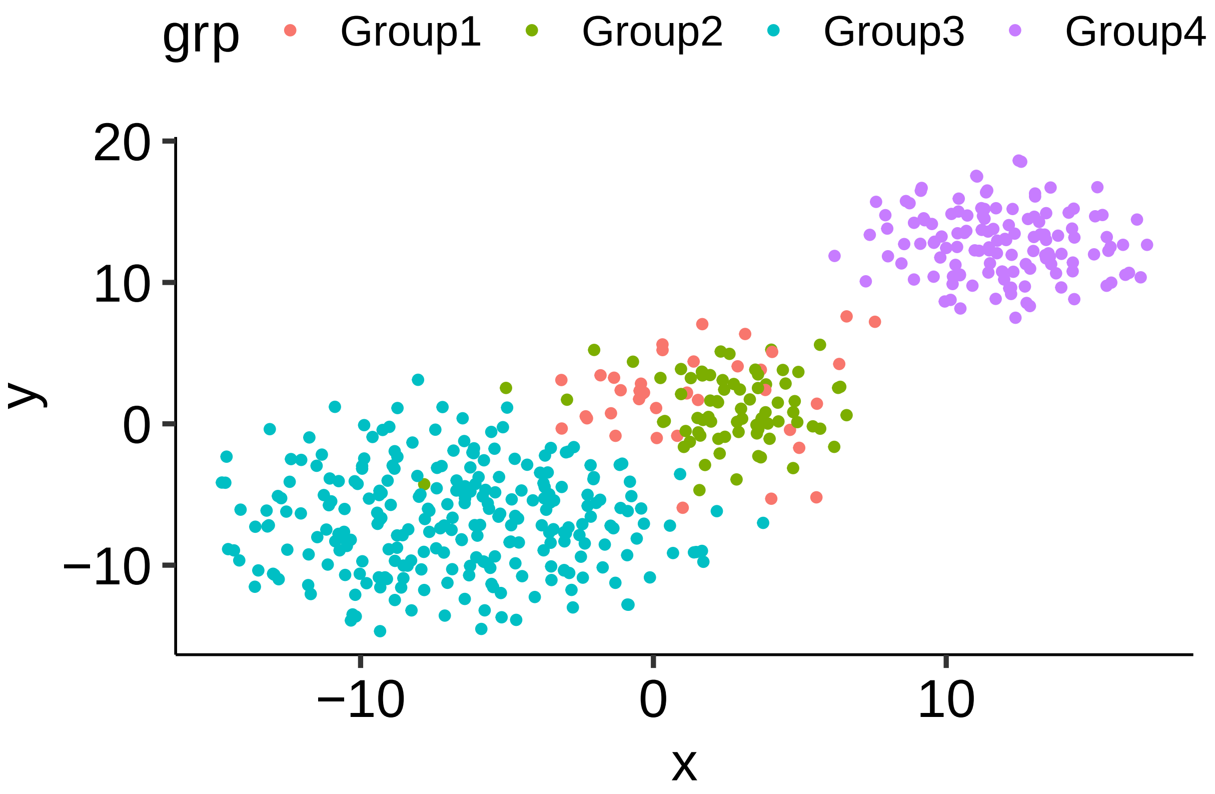
- `rnaseq`: this is the log transformed scRNA-seq matrix

- `celltype`: this encodes the cell labels. There are four types of cells in total, which are Group1, Group2, Group3 and Group4.

```
sce <- sce_full_SimKumar4easy(metadata = F)
rnaseq <- sce@assays$data$logcounts
celltype <- sce$Group
```

We visualize `rnaseq` in the tSNE space and color dots based on `celltype`.

```
dat.plot <- data.frame(x = sce@reducedDims$TSNE[, 1], y = sce@reducedDims$TSNE[,
    2], grp = celltype)
ggplot(dat.plot, aes(x = x, y = y, color = grp)) + geom_point() +
    theme_pubr(base_size = 20)
```



We filter out genes with low-expression levels (missing $>$ 5% of cells)

```
use_genes <- which(apply(rnaseq == 0, 1, sum)/ncol(rnaseq) < 0.95)
rnaseq <- rnaseq[use_genes, ]
```

We then normalize the the total gene expression of each cell to be the median value across all cells using the function `normalize_barcode_sums_to_median()`.

```
rnaseq <- normalize_barcode_sums_to_median(rnaseq)
```

Finally we perform SVD smoothing by replacing `rnaseq` with the SVD approximation of the first 50 decomposed principal components.

```
svdres <- rsvd(tscale(rnaseq), k = 50)
rnaseq.norm <- svdres$u %*% diag(svdres$d[1:50]) %*% t(svdres$v)
rownames(rnaseq.norm) <- rownames(rnaseq)
colnames(rnaseq.norm) <- colnames(rnaseq)
```

Run NIFA

`one_hot_encode()` function is used to convert `celltype` with four cell types into one hot encoding (cell-by-celltype). We can easily compute the correlation between decomposed latent factors and one hot encoding to check the correspondence between latent factors and each cell group.

```r
one_hot_encode <- function(x) {
    nrow <- length(x)
    cat <- names(table(x))
    ncol <- length(cat)
    res <- matrix(0, nrow = nrow, ncol = ncol)
    for (i in 1:ncol) {
        res[which(x == cat[i]), i] <- 1
    }  #for i
    return(res)
}  #one_hot_encode
```

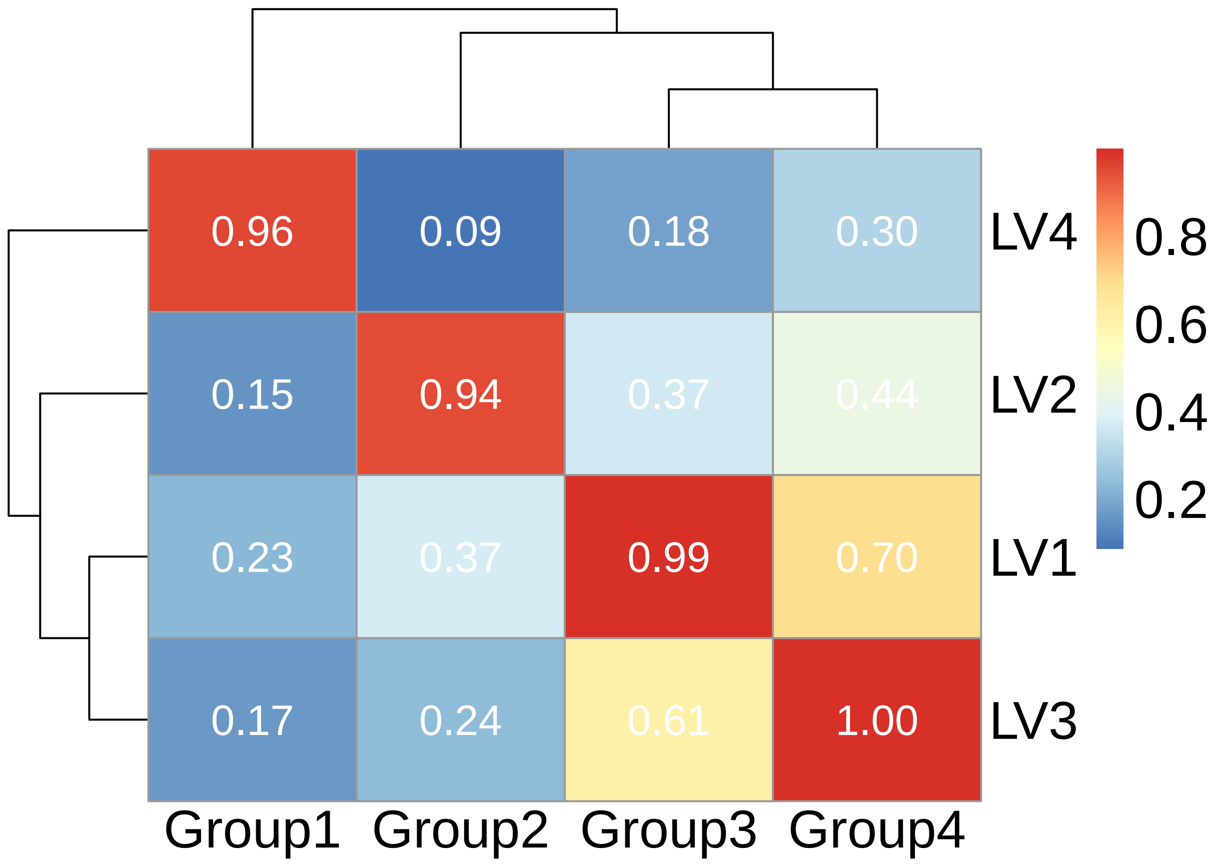
We run *NIFA* by calling `NIFA()` function. Here the number of latent factors is set to be 4 and the number of mixture components associated with each latent factor is also set to be 4. By default, variables are initialized by a simple matrix decomposition with non-negativity constraint on the loadings (`init = "sd", S.init = NULL, A.init = NULL, L1.sd = NULL, L2.sd = NULL`). There are five parameters that are more data-dependent than other parameters.

- `K`: This encodes the number of latent factors. You can set up this parameter by a few runs of experiments. You can also try AIC/BIC criterion on top of SVD decomposition or likelihood/ELBO-based tuning methods.
- `S_threshold`: (default: 6e-5) `NIFA()` monitor the relative changes of decomposed latent factors. This will control the number of iterations before convergence.
- `max.iter`: (default: 1000) This is a hard threshold to control the number of iterations before stopping the variational updates.
- `b_noise_prior`: The empirical recommendation is to set this prior over the noise parameter as `prod(dim(X)*5)`. You can change this parameter or the next one (`beta_expect_flag`) to adapt to your specific scRNA-seq data.
- `beta_expect_flag`: The noise parameter can be set to a fixed value by assigning the desired value to `beta_expect_flag`. Other related parameters will be ignored once `beta_expect_flag` is not `NULL`.

```r
NIFA.res <- NIFA(tscale(rnaseq.norm), K = 4, M = 4, max.iter = 500,
    S_threshold = 6e-05, init = "sd", A.init = NULL, S.init = NULL,
    verbose = F, ref = one_hot_encode(celltype), beta_expect_flag = NULL,
    L1.sd = NULL, L2.sd = NULL, b_noise_prior = prod(dim(rnaseq.norm)) *
        5)
```

We calculate the Pearson correlations between latent factors and one hot encoding of all cell types and we visualize this correspondence using heatmap. There is a one-to-one correspondence between cell types and NIFA latent factors with Pearson correlation > 0.9.

```r
cor.res <- cor(t(NIFA.res$mu_S), one_hot_encode(celltype), method = "p")
rownames(cor.res) <- paste0("LV", 1:nrow(NIFA.res$mu_S))
colnames(cor.res) <- sort(unique(celltype))
pheatmap(abs(cor.res), fontsize = 20, angle_col = "0", display_numbers = T,
    number_color = "white")
```



0.96
0.09
0.18
0.30
LV4
0.15
0.94
0.37
0.44
LV2
0.23
0.37
0.99
0.70
LV1
0.17
0.24
0.61
1.00
LV3
Group1
Group2
Group3
Group4
0.8
0.6
0.4
0.2