Individual Sample Analysis

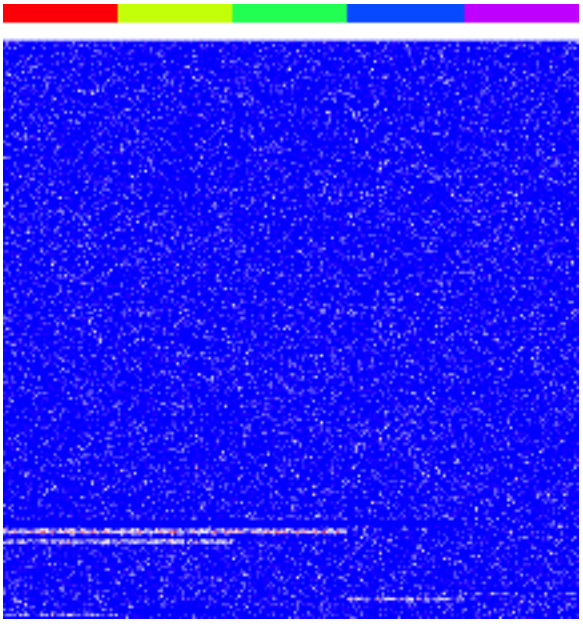
Jean Fan

2017-11-25

To demonstrate the utility of `MUDAN` for analysis of an individual sample, we will use simulated data to show how poor signal can be strengthened by graph-based community detection.

Let's begin by simulating some data where we have 5 different cell types / groups.

```
library(MUDAN)

## simulate data
simulate.data <- function(G=5, N=100, M=1000, initmean=0, initvar=10, upreg=1, upregvar=15, ng=10, ng2=1
  set.seed(seed)
  mat <- matrix(rnorm(N*M*G, initmean, initvar), M, N*G)
  rownames(mat) <- paste0('gene', 1:M)
  colnames(mat) <- paste0('cell', 1:(N*G))
  group <- factor(sapply(1:G, function(x) rep(paste0('group', x), N)))
  names(group) <- colnames(mat)

  ## unique diff genes
  diff <- lapply(1:G, function(x) {
    diff <- rownames(mat)[(((x-1)*ng)+1):(((x-1)*ng)+ng)]
    mat[diff, group==paste0('group', x)] <<- mat[diff, group==paste0('group', x)] + rnorm(ng, upreg, upr
    return(diff)
  })
  names(diff) <- paste0('group', 1:G)

  ## shared subpops
  diff2 <- lapply(2:(G-1), function(x) {
    y <- x+G
    diff <- rownames(mat)[(((y-1)*ng2)+1):(((y-1)*ng2)+ng2)]
    mat[diff, group %in% paste0("group", 1:x)] <<- mat[diff, group %in% paste0("group", 1:x)] + rnorm(ng
    return(diff)
  })

  mat[mat<0] <- 0
  mat <- round(mat)

  if(plot) {
    heatmap(mat, Rowv=NA, Colv=NA, col=colorRampPalette(c('blue', 'white', 'red'))(100), scale="none", (
  }

  return(list(mat=mat, group=group))
}

data <- simulate.data()
```



First, we will use a conventional PCA dimensionality approach followed by tSNE. What we get is a fuzzy blob.

```
myMudanObject <- Mudan$new("sim", data$mat, ncores=4)
```

```
## [1] "Filtering matrix with 500 cells and 1000 genes ..."
## [1] "Resulting matrix has 500 cells and 1000 genes"
```

```
myMudanObject$libSizeNormalize()
```

```
## [1] "Normalizing matrix with 500 cells and 1000 genes"
```

```
myMudanObject$varianceNormalize(plot=FALSE)
```

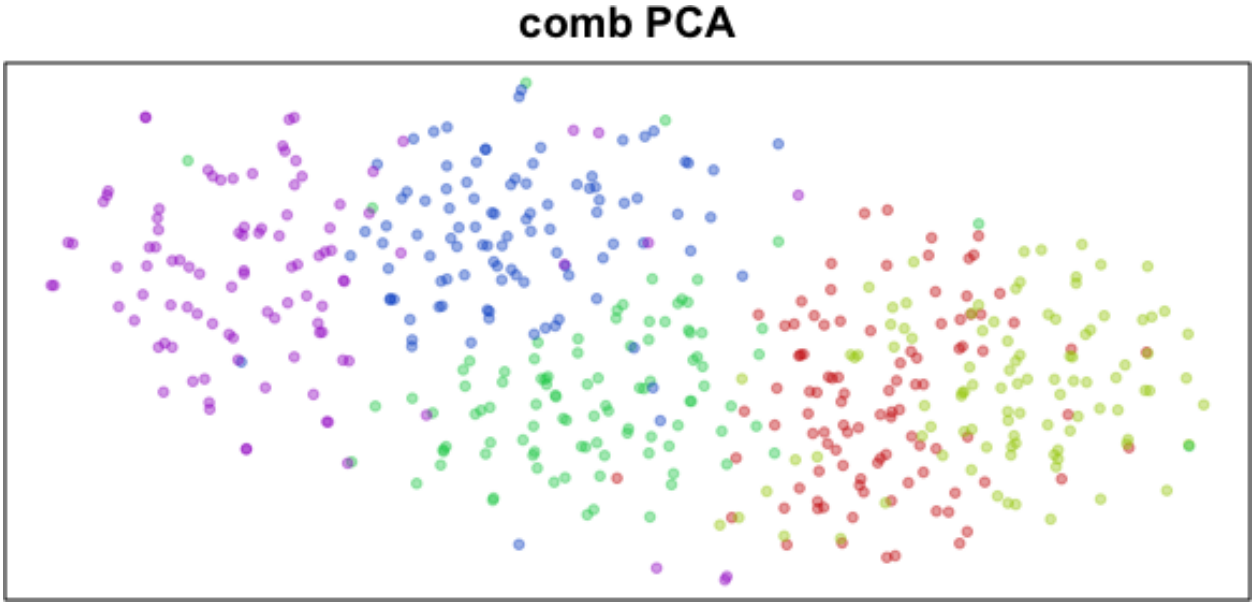
```
## [1] "Calculating variance fit ..."
## [1] "Using gam with k=5..."
## [1] "3 overdispersed genes ... "
```

```
myMudanObject$dimensionalityReduction(nGenes = 1000, nPcs = 30, maxit=1000)
```

```
## [1] "Identifying top 30 PCs on 1000 most variable genes ..."
```

```
myMudanObject$getStandardEmbedding(plot=FALSE)

## PCA-based approach
plotEmbedding(myMudanObject$emb[['PCA']], groups=data$group, main="comb PCA")
```

```
## using provided groups as a factor
```



Now we will use various graph-based community detection approaches to identify potential groups and the corresponding lower-dimensional space that can capture this group-separability. We will run tSNE on these lower-dimensional spaces corresponding to each set of identified groups.

```
## community detection
myMudanObject$communityDetection(reductionType='pcs', communityName="Walktrap5", communityMethod=igraph
```

```
## [1] "finding approximate nearest neighbors ..."
## [1] "calculating clustering ..."
## [1] "graph modularity: 0.503677070140839"
## [1] "identifying cluster membership ..."
## com
##   1   2   3   4   5   6   7   8
## 110 132  49  61  34  52  54   8
```

```
myMudanObject$communityDetection(reductionType='pcs', communityName="Walktrap10", communityMethod=igrapl
```

```
## [1] "finding approximate nearest neighbors ..."
## [1] "calculating clustering ..."
## [1] "graph modularity: 0.432157695293427"
## [1] "identifying cluster membership ..."
## com
##   1   2   3   4
## 193  99 116  92
```

```
myMudanObject$communityDetection(reductionType='pcs', communityName="Walktrap20", communityMethod=igrapl
```

```
## [1] "finding approximate nearest neighbors ..."
## [1] "calculating clustering ..."
## [1] "graph modularity: 0.377821356058121"
## [1] "identifying cluster membership ..."
## com
##   1   2   3   4
## 201 114  87  98
```

```
## use all communities detected
myMudanObject$modelCommunity()
```

```
## [1] "calculating LDA ..."
## [1] "LDA prediction accuracy ..."
##
## FALSE  TRUE
##    16   484
## [1] "calculating LDA ..."
## [1] "LDA prediction accuracy ..."
##
## FALSE  TRUE
##    17   483
## [1] "calculating LDA ..."
## [1] "LDA prediction accuracy ..."
##
## FALSE  TRUE
##    18   482
```

```
myMudanObject$getMudanEmbedding(plot=FALSE)
```

```
## [1] "Running LDA models ..."
## [1] "Running Rtsne with perplexity 30"
## Read the 500 x 13 data matrix successfully!
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Normalizing input...
## Building tree...
##  - point 0 of 500
## Done in 0.03 seconds (sparsity = 0.221336)!
## Learning embedding...
## Iteration 50: error is 55.515618 (50 iterations in 0.16 seconds)
## Iteration 100: error is 53.732980 (50 iterations in 0.15 seconds)
## Iteration 150: error is 53.615481 (50 iterations in 0.14 seconds)
## Iteration 200: error is 53.610243 (50 iterations in 0.15 seconds)
## Iteration 250: error is 53.602528 (50 iterations in 0.14 seconds)
## Iteration 300: error is 0.797772 (50 iterations in 0.14 seconds)
## Iteration 350: error is 0.745621 (50 iterations in 0.14 seconds)
## Iteration 400: error is 0.729199 (50 iterations in 0.14 seconds)
## Iteration 450: error is 0.723564 (50 iterations in 0.14 seconds)
## Iteration 500: error is 0.721542 (50 iterations in 0.14 seconds)
## Iteration 550: error is 0.720105 (50 iterations in 0.13 seconds)
## Iteration 600: error is 0.718806 (50 iterations in 0.14 seconds)
## Iteration 650: error is 0.718362 (50 iterations in 0.14 seconds)
## Iteration 700: error is 0.716923 (50 iterations in 0.13 seconds)
## Iteration 750: error is 0.716214 (50 iterations in 0.14 seconds)
## Iteration 800: error is 0.715992 (50 iterations in 0.14 seconds)
## Iteration 850: error is 0.715333 (50 iterations in 0.15 seconds)
## Iteration 900: error is 0.714791 (50 iterations in 0.16 seconds)
## Iteration 950: error is 0.714298 (50 iterations in 0.15 seconds)
## Iteration 1000: error is 0.713053 (50 iterations in 0.15 seconds)
## Fitting performed in 2.88 seconds.
```

```
plotEmbedding(myMudanObject$emb[['MUDAN']], groups=data$group, main="MUDAN")
```

```
## using provided groups as a factor
```

We are able to much more clearly visually identify our original groups, even without prior knowledge of the groups.

In theory, if we wanted to simply visualize known groups, we can also provide those group labels to identify a corresponding lower-dimensional space that can capture this group-separability.

```
myMudanObjectb <- Mudan$new("biased", data$mat, ncores=4)
```

```
## [1] "Filtering matrix with 500 cells and 1000 genes ..."
## [1] "Resulting matrix has 500 cells and 1000 genes"
```

```
myMudanObjectb$libSizeNormalize()
```

```
## [1] "Normalizing matrix with 500 cells and 1000 genes"
```

```
myMudanObjectb$varianceNormalize(plot=FALSE)
```

```
## [1] "Calculating variance fit ..."
## [1] "Using gam with k=5..."
## [1] "3 overdispersed genes ... "
```

```
myMudanObjectb$dimensionalityReduction(nGenes = 1000, nPcs = 30, maxit=1000)
```

```
## [1] "Identifying top 30 PCs on 1000 most variable genes ..."
```

```
## use known groups rather than unbiased detection from graph-based approaches
myMudanObjectb$modelCommunity(groups=data$group)
```

```
## [1] "calculating LDA ..."
## [1] "LDA prediction accuracy ..."
##
## TRUE
##  500
```

```
myMudanObjectb$getMudanEmbedding(plot=FALSE)
```

```
## [1] "Running LDA models ..."
## [1] "Running Rtsne with perplexity 30"
## Read the 500 x 4 data matrix successfully!
```

```
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Normalizing input...
## Building tree...
##  - point 0 of 500
## Done in 0.04 seconds (sparsity = 0.192768)!
## Learning embedding...
## Iteration 50: error is 49.045983 (50 iterations in 0.18 seconds)
## Iteration 100: error is 45.534837 (50 iterations in 0.17 seconds)
## Iteration 150: error is 44.814200 (50 iterations in 0.15 seconds)
## Iteration 200: error is 44.470875 (50 iterations in 0.24 seconds)
## Iteration 250: error is 44.261395 (50 iterations in 0.18 seconds)
## Iteration 300: error is 0.624804 (50 iterations in 0.15 seconds)
## Iteration 350: error is 0.522839 (50 iterations in 0.25 seconds)
## Iteration 400: error is 0.492094 (50 iterations in 0.15 seconds)
## Iteration 450: error is 0.471754 (50 iterations in 0.15 seconds)
## Iteration 500: error is 0.460690 (50 iterations in 0.14 seconds)
## Iteration 550: error is 0.452535 (50 iterations in 0.15 seconds)
## Iteration 600: error is 0.448243 (50 iterations in 0.15 seconds)
## Iteration 650: error is 0.445512 (50 iterations in 0.13 seconds)
## Iteration 700: error is 0.443549 (50 iterations in 0.13 seconds)
## Iteration 750: error is 0.442603 (50 iterations in 0.12 seconds)
## Iteration 800: error is 0.441130 (50 iterations in 0.13 seconds)
## Iteration 850: error is 0.438845 (50 iterations in 0.13 seconds)
## Iteration 900: error is 0.437468 (50 iterations in 0.13 seconds)
## Iteration 950: error is 0.435759 (50 iterations in 0.14 seconds)
## Iteration 1000: error is 0.434216 (50 iterations in 0.14 seconds)
## Fitting performed in 3.11 seconds.
```

```
plotEmbedding(myMudanObjectb$emb[['MUDAN']], groups=data$group, main="biased MUDAN")
```

```
## using provided groups as a factor
```