Multi-sample Unified Discriminant Analysis

Jean Fan

2018-02-27

```
library(MUDAN)

## load data
data(pbmcA)
A <- pbmcA[,1:1000]
## b cells identified from previous analysis
b.cells.pbmcA <- c("frozen_pbmc_donor_a_AAACATTGGCTAAC", "frozen_pbmc_donor_a_AAACCGTGTTACCT",
                   "frozen_pbmc_donor_a_AAACGCACTTCTAC", "frozen_pbmc_donor_a_AAAGACGAATGACC",
                   "frozen_pbmc_donor_a_AAAGGCCTCGAACT", "frozen_pbmc_donor_a_AAAGGCCTTGACCA",
                   "frozen_pbmc_donor_a_AAATCCCTAACCAC", "frozen_pbmc_donor_a_AAATGTTGAGATGA",
                   "frozen_pbmc_donor_a_AAATTCGAATTTCC", "frozen_pbmc_donor_a_AACACGTGCCCTTG",
                   "frozen_pbmc_donor_a_AACACTCTAGCAAA", "frozen_pbmc_donor_a_AACAGAGACTCGCT",
                   "frozen_pbmc_donor_a_AACCACGAAACAGA", "frozen_pbmc_donor_a_AACCACGAATGCTG",
                   "frozen_pbmc_donor_a_AACCGATGTGTGGT", "frozen_pbmc_donor_a_AACCGCCTGTTGTG",
                   "frozen_pbmc_donor_a_AACGCCCTTTCGGA", "frozen_pbmc_donor_a_AACGTGTGGTAGCT",
                   "frozen_pbmc_donor_a_AACTGTCTACCAAC", "frozen_pbmc_donor_a_AAGCAAGAGATAGA",
                   "frozen_pbmc_donor_a_AAGTAGGAAACCGT", "frozen_pbmc_donor_a_AATCCTTGCGAATC",
                   "frozen_pbmc_donor_a_AATCTCACCTCGAA", "frozen_pbmc_donor_a_AATGATACAGCACT",
                   "frozen_pbmc_donor_a_ACACCCTGAAAACG", "frozen_pbmc_donor_a_ACAGGTACTAGCGT",
                   "frozen_pbmc_donor_a_ACAGTCGAATGACC", "frozen_pbmc_donor_a_ACATTCTGGTACCA",
                   "frozen_pbmc_donor_a_ACCATTACTGTTTC", "frozen_pbmc_donor_a_ACCCAAGAGCCTTC",
                   "frozen_pbmc_donor_a_ACCTGGCTATCACG", "frozen_pbmc_donor_a_ACGAACACGAGCTT",
                   "frozen_pbmc_donor_a_ACGATTCTCCTACC", "frozen_pbmc_donor_a_ACGATTCTGTTCTT",
                   "frozen_pbmc_donor_a_ACGCACCTGTCATG", "frozen_pbmc_donor_a_ACGCACCTTTGGCA",
                   "frozen_pbmc_donor_a_ACGCTCACTGACAC", "frozen_pbmc_donor_a_ACGGATTGACGTAC",
                   "frozen_pbmc_donor_a_ACTATCACGGTGGA", "frozen_pbmc_donor_a_AGACACACCCCTTG",
                   "frozen_pbmc_donor_a_AGACTTCTCTGGAT", "frozen_pbmc_donor_a_AGATTAACCCTGAA",
                   "frozen_pbmc_donor_a_AGCATCGACCTTTA", "frozen_pbmc_donor_a_AGGACACTTAACCG",
                   "frozen_pbmc_donor_a_AGGGACGAAAACAG", "frozen_pbmc_donor_a_AGGTTGTGGAGGAC",
                   "frozen_pbmc_donor_a_AGTCACGAAAGATG", "frozen_pbmc_donor_a_AGTGTGACTTGACG",
                   "frozen_pbmc_donor_a_AGTTTAGATCGCCT", "frozen_pbmc_donor_a_AGTTTCACTTGCGA",
                   "frozen_pbmc_donor_a_ATAGGCTGAGTTCG", "frozen_pbmc_donor_a_ATAGGCTGGCGAGA",
                   "frozen_pbmc_donor_a_ATCAACCTGGTGGA", "frozen_pbmc_donor_a_ATCAGGTGGAGATA",
                   "frozen_pbmc_donor_a_ATCCAGGAGCAAGG", "frozen_pbmc_donor_a_ATCCCGTGTGTAGC",
                   "frozen_pbmc_donor_a_ATCCTAACCAGAAA", "frozen_pbmc_donor_a_ATCGCCACGAATCC",
                   "frozen_pbmc_donor_a_ATCTACTGCAGAAA", "frozen_pbmc_donor_a_ATCTCAACGGTTTG",
                   "frozen_pbmc_donor_a_ATGAGCACACCACA", "frozen_pbmc_donor_a_ATGGACACTGGAGG",
                   "frozen_pbmc_donor_a_ATGTACCTGTGTTG", "frozen_pbmc_donor_a_ATGTCACTGTCGAT",
                   "frozen_pbmc_donor_a_ATGTTGCTCGGGAA", "frozen_pbmc_donor_a_ATTAGTGAAGCGGA",
                   "frozen_pbmc_donor_a_ATTCAGCTGTCCTC", "frozen_pbmc_donor_a_ATTCTTCTGAAAGT",
                   "frozen_pbmc_donor_a_ATTTCCGATCAGTG", "frozen_pbmc_donor_a_CAAGGACTACCCAA",
                   "frozen_pbmc_donor_a_CAAGTTCTGTTTCT", "frozen_pbmc_donor_a_CACAGATGCCATAG",
                   "frozen_pbmc_donor_a_CACCACTGTGGTGT", "frozen_pbmc_donor_a_CACCGGGACTTGAG",
                   "frozen_pbmc_donor_a_CACCGTTGTCGTTT", "frozen_pbmc_donor_a_CACTCCGAGTCACA",
                   "frozen_pbmc_donor_a_CACTTATGTTGTGG", "frozen_pbmc_donor_a_CAGACTGAATAAGG",
                   "frozen_pbmc_donor_a_CAGACTGACGACTA")
A <- A[, setdiff(colnames(A), b.cells.pbmcA)]
```

```
dim(A)
```

```
## [1] 13939   921
```

```
data(pbmcB)
B <- pbmcB[,1:1000]
data(pbmcC)
C <- pbmcC[,1:1000]
cds <- list(A, B, C)
names(cds) <- c('A', 'B', 'C')

## combine
k <- 0
vi <- Reduce(intersect, lapply(cds, function(cd) {
  vi <- rowSums(cd)>k
  rownames(cd)[vi]
}))
cds.filter <- lapply(cds, function(cd) cd[vi,])
names(cds.filter)
```

```
## [1] "A" "B" "C"
```

```
sample <- unlist(lapply(1:length(cds.filter), function(i) {
  cd <- cds.filter[[i]]
  n <- names(cds.filter)[i]
  sample <- rep(n, ncol(cd))
  names(sample) <- colnames(cd)
  return(sample)
}))
```

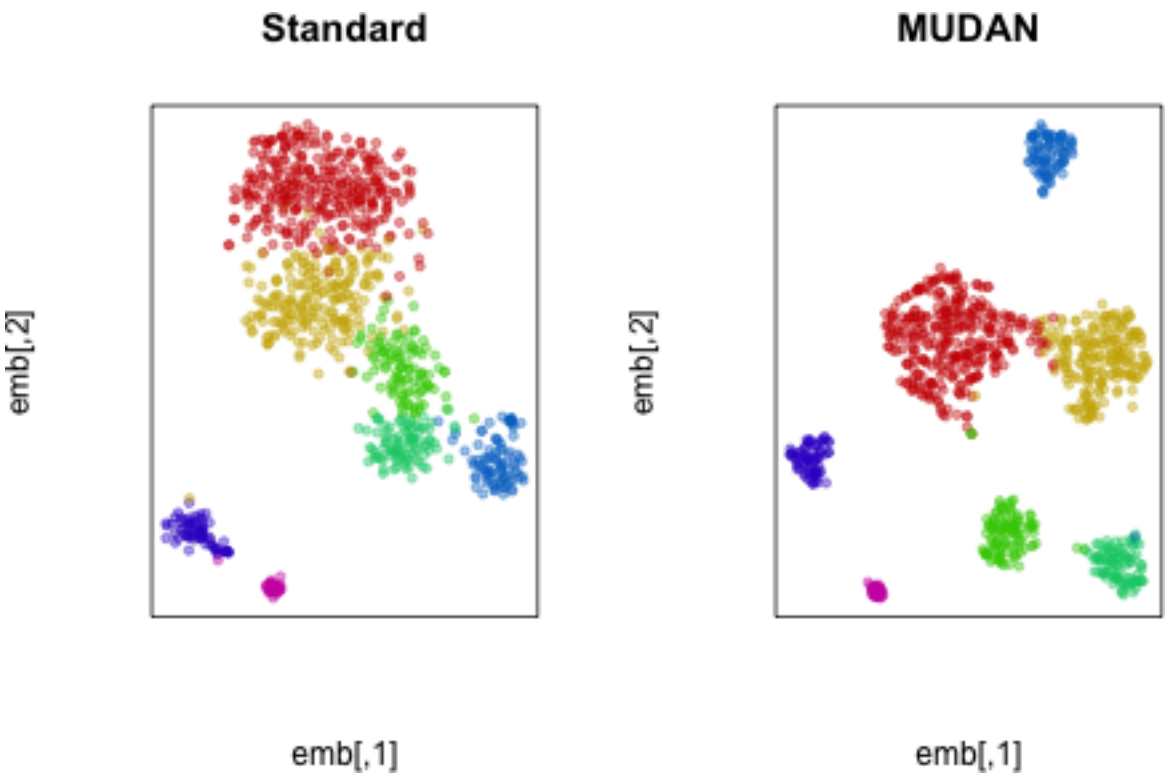
```
## run model for each
getModelInfo <- function(name, cd) {
  myMudanObject <- Mudan$new(name, cd)
  myMudanObject$normalizeCounts()
  myMudanObject$normalizeVariance(plot=FALSE)
  myMudanObject$getPcs(nPcs=30, maxit=1000)
  myMudanObject$getComMembership(communityName='Infomap',
       communityMethod=igraph::cluster_infomap, k=30)
  myMudanObject$getStableClusters(communityName='Infomap')
  myMudanObject$modelCommunity(communityName='Infomap')
  myMudanObject$getMudanEmbedding()
  myMudanObject$getStandardEmbedding()
  par(mfrow=c(1,2))
  myMudanObject$plotEmbedding(communityName='Infomap', embeddingType='PCA',
                              xlab=NULL, ylab=NULL, main='Standard')
  myMudanObject$plotEmbedding(communityName='Infomap', embeddingType='MUDAN',
                              xlab=NULL, ylab=NULL, main='MUDAN')
  return(myMudanObject)
}
models.info <- lapply(seq_along(cds.filter), function(i) {
  getModelInfo(names(cds.filter)[i], cds.filter[[i]])
})
```

```
## [1] "Calculating variance fit ..."
## [1] "Using gam with k=5..."
## [1] "1793 overdispersed genes ... "
```

```
## [1] "Identifying top 30 PCs on 1793 most variable genes ..."
## [1] "finding approximate nearest neighbors ..."
## [1] "calculating clustering ..."
## [1] "graph modularity: 0.625284507025537"
## [1] "identifying cluster membership ..."
## com
##   1   2   3   4   5   6   7
## 322 208 107  93  90  71  30
## [1] "Comparing 3 and 1.2"
## [1] "Running differential expression with 2 clusters ... "
## [1] "Summarizing results ... "
## [1] "Differential genes found: "
## 1.2   3
##  78  26
## [1] "Comparing 1 and 2"
## [1] "Running differential expression with 2 clusters ... "
## [1] "Summarizing results ... "
## [1] "Differential genes found: "
## 1 2
## 4 4
## [1] "Comparing 4 and 5"
## [1] "Running differential expression with 2 clusters ... "
## [1] "Summarizing results ... "
## [1] "Differential genes found: "
## 4 5
## 4 4
## [1] "Comparing 6 and 7"
## [1] "Running differential expression with 2 clusters ... "
## [1] "Summarizing results ... "
## [1] "Differential genes found: "
## 6 7
## 4 4
## [1] "calculating LDA ..."
## Read the 921 x 6 data matrix successfully!
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Normalizing input...
## Building tree...
##  - point 0 of 921
## Done in 0.06 seconds (sparsity = 0.121595)!
## Learning embedding...
## Iteration 50: error is 61.451258 (50 iterations in 0.33 seconds)
## Iteration 100: error is 54.917032 (50 iterations in 0.26 seconds)
## Iteration 150: error is 54.214373 (50 iterations in 0.27 seconds)
## Iteration 200: error is 53.924818 (50 iterations in 0.26 seconds)
## Iteration 250: error is 53.748881 (50 iterations in 0.27 seconds)
## Iteration 300: error is 0.954575 (50 iterations in 0.26 seconds)
## Iteration 350: error is 0.821470 (50 iterations in 0.25 seconds)
## Iteration 400: error is 0.790125 (50 iterations in 0.26 seconds)
## Iteration 450: error is 0.772051 (50 iterations in 0.26 seconds)
## Iteration 500: error is 0.764397 (50 iterations in 0.26 seconds)
## Iteration 550: error is 0.758656 (50 iterations in 0.26 seconds)
## Iteration 600: error is 0.752944 (50 iterations in 0.28 seconds)
## Iteration 650: error is 0.749300 (50 iterations in 0.27 seconds)
```

```
## Iteration 700: error is 0.745859 (50 iterations in 0.27 seconds)
## Iteration 750: error is 0.743528 (50 iterations in 0.29 seconds)
## Iteration 800: error is 0.742100 (50 iterations in 0.28 seconds)
## Iteration 850: error is 0.739686 (50 iterations in 0.27 seconds)
## Iteration 900: error is 0.738461 (50 iterations in 0.27 seconds)
## Iteration 950: error is 0.737291 (50 iterations in 0.28 seconds)
## Iteration 1000: error is 0.735977 (50 iterations in 0.28 seconds)
## Fitting performed in 5.44 seconds.
## Read the 921 x 30 data matrix successfully!
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Normalizing input...
## Building tree...
##  - point 0 of 921
## Done in 0.09 seconds (sparsity = 0.142879)!
## Learning embedding...
## Iteration 50: error is 63.060015 (50 iterations in 0.55 seconds)
## Iteration 100: error is 61.635793 (50 iterations in 0.38 seconds)
## Iteration 150: error is 61.477668 (50 iterations in 0.38 seconds)
## Iteration 200: error is 61.412145 (50 iterations in 0.40 seconds)
## Iteration 250: error is 61.375972 (50 iterations in 0.40 seconds)
## Iteration 300: error is 1.357404 (50 iterations in 0.31 seconds)
## Iteration 350: error is 1.242661 (50 iterations in 0.29 seconds)
## Iteration 400: error is 1.205498 (50 iterations in 0.29 seconds)
## Iteration 450: error is 1.192596 (50 iterations in 0.32 seconds)
## Iteration 500: error is 1.187425 (50 iterations in 0.29 seconds)
## Iteration 550: error is 1.182459 (50 iterations in 0.29 seconds)
## Iteration 600: error is 1.178946 (50 iterations in 0.31 seconds)
## Iteration 650: error is 1.176006 (50 iterations in 0.29 seconds)
## Iteration 700: error is 1.173396 (50 iterations in 0.31 seconds)
## Iteration 750: error is 1.172054 (50 iterations in 0.33 seconds)
## Iteration 800: error is 1.170638 (50 iterations in 0.30 seconds)
## Iteration 850: error is 1.169612 (50 iterations in 0.30 seconds)
## Iteration 900: error is 1.167899 (50 iterations in 0.30 seconds)
## Iteration 950: error is 1.167141 (50 iterations in 0.30 seconds)
## Iteration 1000: error is 1.166483 (50 iterations in 0.35 seconds)
## Fitting performed in 6.68 seconds.
## using provided groups as a factor
```

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

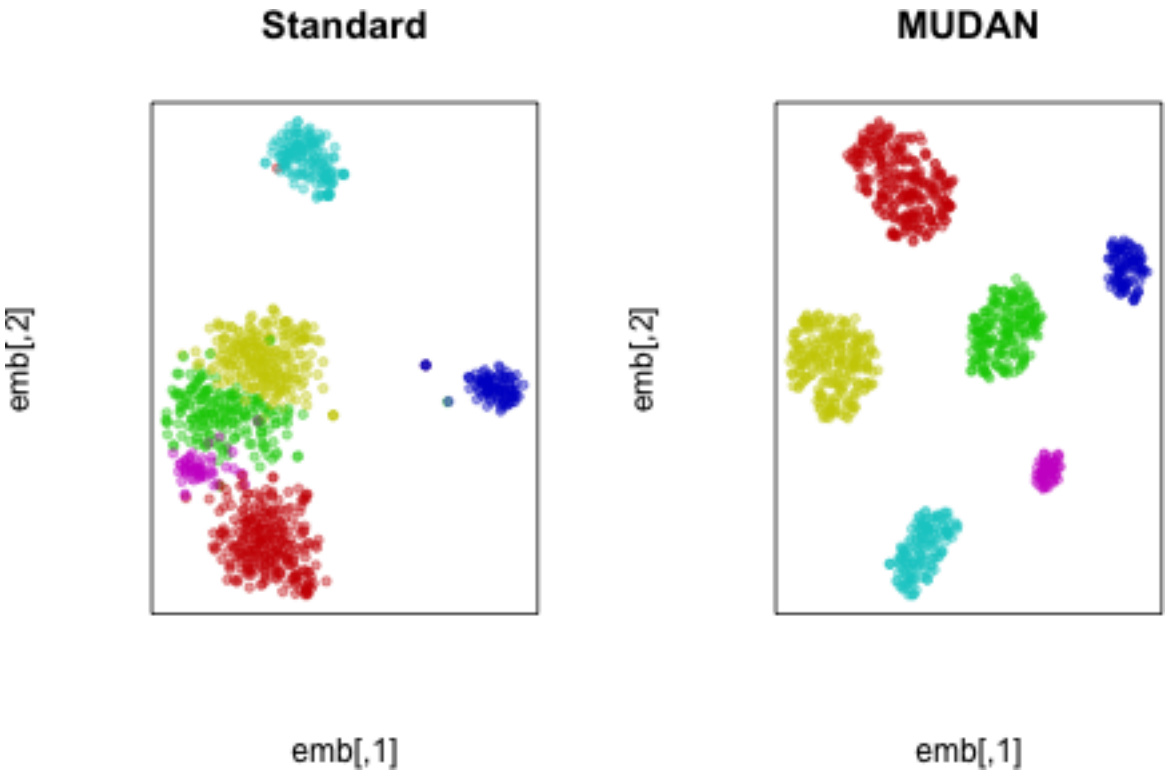



```
## [1] "Calculating variance fit ..."
## [1] "Using gam with k=5..."
## [1] "2023 overdispersed genes ... "
## [1] "Identifying top 30 PCs on 2023 most variable genes ..."
## [1] "finding approximate nearest neighbors ..."
## [1] "calculating clustering ..."
## [1] "graph modularity: 0.643004603751073"
## [1] "identifying cluster membership ..."
## com
##   1   2   3   4   5   6
## 269 243 189 149  94  56
## [1] "Comparing 4 and 5.1.6.2.3"
## [1] "Running differential expression with 2 clusters ... "
## [1] "Summarizing results ... "
## [1] "Differential genes found: "
##         4 5.1.6.2.3
##       359       124
## [1] "Comparing 5 and 1.6.2.3"
## [1] "Running differential expression with 2 clusters ... "
## [1] "Summarizing results ... "
## [1] "Differential genes found: "
## 1.6.2.3       5
##      57      42
## [1] "Comparing 1 and 6.2.3"
## [1] "Running differential expression with 2 clusters ... "
## [1] "Summarizing results ... "
## [1] "Differential genes found: "
##     1 6.2.3
##    95   154
```

```
## [1] "Comparing 6 and 2.3"
## [1] "Running differential expression with 2 clusters ... "
## [1] "Summarizing results ... "
## [1] "Differential genes found: "
## 2.3   6
##  13  21
## [1] "Comparing 2 and 3"
## [1] "Running differential expression with 2 clusters ... "
## [1] "Summarizing results ... "
## [1] "Differential genes found: "
## 2 3
## 4 4
## [1] "calculating LDA ..."
## Read the 1000 x 5 data matrix successfully!
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Normalizing input...
## Building tree...
##  - point 0 of 1000
## Done in 0.07 seconds (sparsity = 0.114056)!
## Learning embedding...
## Iteration 50: error is 59.507010 (50 iterations in 0.33 seconds)
## Iteration 100: error is 53.965820 (50 iterations in 0.32 seconds)
## Iteration 150: error is 53.115706 (50 iterations in 0.32 seconds)
## Iteration 200: error is 52.731841 (50 iterations in 0.31 seconds)
## Iteration 250: error is 52.479507 (50 iterations in 0.30 seconds)
## Iteration 300: error is 1.127210 (50 iterations in 0.29 seconds)
## Iteration 350: error is 0.933009 (50 iterations in 0.27 seconds)
## Iteration 400: error is 0.869174 (50 iterations in 0.27 seconds)
## Iteration 450: error is 0.845957 (50 iterations in 0.29 seconds)
## Iteration 500: error is 0.830810 (50 iterations in 0.29 seconds)
## Iteration 550: error is 0.819192 (50 iterations in 0.29 seconds)
## Iteration 600: error is 0.809856 (50 iterations in 0.30 seconds)
## Iteration 650: error is 0.803243 (50 iterations in 0.34 seconds)
## Iteration 700: error is 0.797107 (50 iterations in 0.30 seconds)
## Iteration 750: error is 0.790482 (50 iterations in 0.28 seconds)
## Iteration 800: error is 0.784121 (50 iterations in 0.29 seconds)
## Iteration 850: error is 0.779846 (50 iterations in 0.29 seconds)
## Iteration 900: error is 0.776954 (50 iterations in 0.29 seconds)
## Iteration 950: error is 0.775011 (50 iterations in 0.27 seconds)
## Iteration 1000: error is 0.773306 (50 iterations in 0.27 seconds)
## Fitting performed in 5.91 seconds.
## Read the 1000 x 30 data matrix successfully!
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Normalizing input...
## Building tree...
##  - point 0 of 1000
## Done in 0.10 seconds (sparsity = 0.134168)!
## Learning embedding...
## Iteration 50: error is 63.229224 (50 iterations in 0.44 seconds)
## Iteration 100: error is 59.867816 (50 iterations in 0.33 seconds)
## Iteration 150: error is 59.631105 (50 iterations in 0.30 seconds)
## Iteration 200: error is 59.554247 (50 iterations in 0.33 seconds)
```

```
## Iteration 250: error is 59.510595 (50 iterations in 0.32 seconds)
## Iteration 300: error is 1.419338 (50 iterations in 0.29 seconds)
## Iteration 350: error is 1.312967 (50 iterations in 0.31 seconds)
## Iteration 400: error is 1.284933 (50 iterations in 0.31 seconds)
## Iteration 450: error is 1.270942 (50 iterations in 0.34 seconds)
## Iteration 500: error is 1.262746 (50 iterations in 0.35 seconds)
## Iteration 550: error is 1.258072 (50 iterations in 0.31 seconds)
## Iteration 600: error is 1.253642 (50 iterations in 0.32 seconds)
## Iteration 650: error is 1.248388 (50 iterations in 0.32 seconds)
## Iteration 700: error is 1.244365 (50 iterations in 0.30 seconds)
## Iteration 750: error is 1.241617 (50 iterations in 0.31 seconds)
## Iteration 800: error is 1.240170 (50 iterations in 0.33 seconds)
## Iteration 850: error is 1.239085 (50 iterations in 0.32 seconds)
## Iteration 900: error is 1.237734 (50 iterations in 0.32 seconds)
## Iteration 950: error is 1.236404 (50 iterations in 0.32 seconds)
## Iteration 1000: error is 1.235131 (50 iterations in 0.32 seconds)
## Fitting performed in 6.48 seconds.
## using provided groups as a factor
```

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

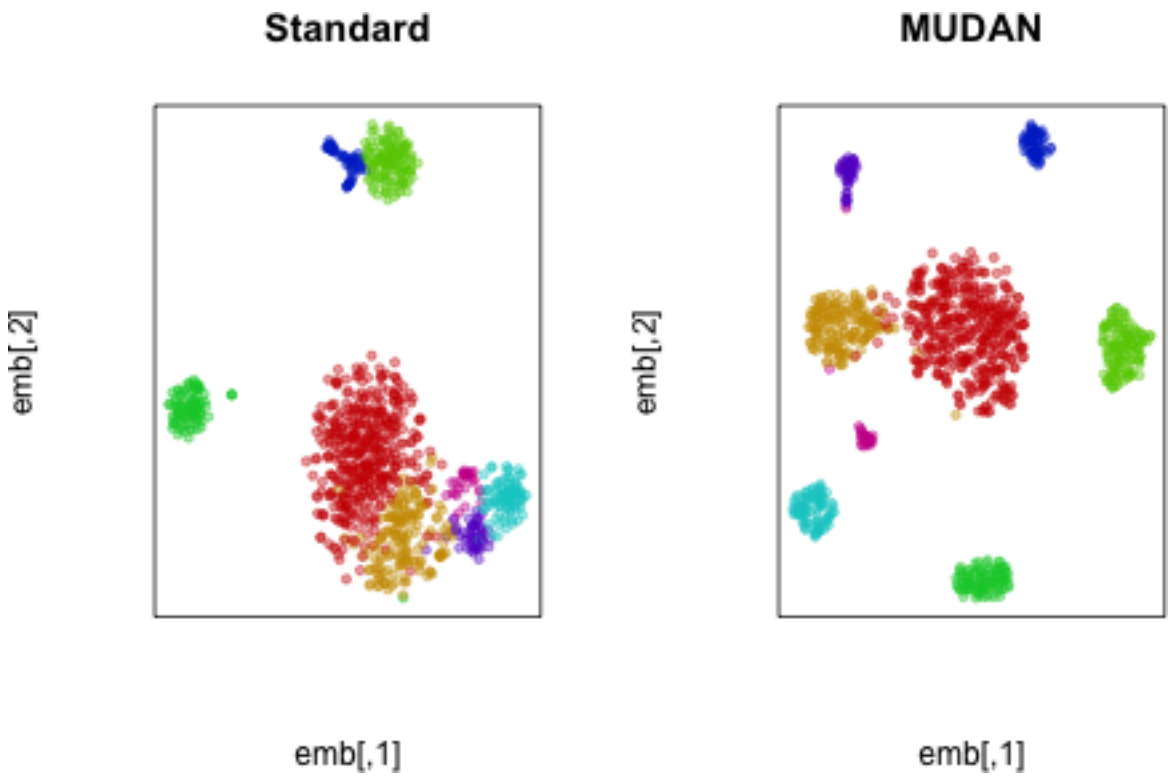



```
## [1] "Calculating variance fit ..."
## [1] "Using gam with k=5..."
## [1] "1783 overdispersed genes ... "
## [1] "Identifying top 30 PCs on 1783 most variable genes ..."
## [1] "finding approximate nearest neighbors ..."
## [1] "calculating clustering ..."
## [1] "graph modularity: 0.586847994635265"
## [1] "identifying cluster membership ..."
## com
```

```
##   1   2   3   4   5   6   7   8
## 398 162 125  97  87  56  46  29
## [1] "Comparing 3 and 6"
## [1] "Running differential expression with 2 clusters ... "
## [1] "Summarizing results ... "
## [1] "Differential genes found: "
## 3 6
## 4 4
## [1] "Comparing 4 and 1.2"
## [1] "Running differential expression with 2 clusters ... "
## [1] "Summarizing results ... "
## [1] "Differential genes found: "
## 1.2   4
##  47  52
## [1] "Comparing 1 and 2"
## [1] "Running differential expression with 2 clusters ... "
## [1] "Summarizing results ... "
## [1] "Differential genes found: "
## 1 2
## 4 4
## [1] "Comparing 8 and 5.7"
## [1] "Running differential expression with 2 clusters ... "
## [1] "Summarizing results ... "
## [1] "Differential genes found: "
## 5.7   8
##   6  18
## [1] "Comparing 5 and 7"
## [1] "Running differential expression with 2 clusters ... "
## [1] "Summarizing results ... "
## [1] "Differential genes found: "
## 5 7
## 4 4
## [1] "calculating LDA ..."
## Read the 1000 x 7 data matrix successfully!
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Normalizing input...
## Building tree...
##  - point 0 of 1000
## Done in 0.07 seconds (sparsity = 0.116116)!
## Learning embedding...
## Iteration 50: error is 62.382026 (50 iterations in 0.37 seconds)
## Iteration 100: error is 56.836647 (50 iterations in 0.31 seconds)
## Iteration 150: error is 56.068522 (50 iterations in 0.30 seconds)
## Iteration 200: error is 55.783584 (50 iterations in 0.31 seconds)
## Iteration 250: error is 55.612223 (50 iterations in 0.30 seconds)
## Iteration 300: error is 1.095148 (50 iterations in 0.30 seconds)
## Iteration 350: error is 0.939942 (50 iterations in 0.29 seconds)
## Iteration 400: error is 0.895714 (50 iterations in 0.28 seconds)
## Iteration 450: error is 0.878509 (50 iterations in 0.29 seconds)
## Iteration 500: error is 0.868358 (50 iterations in 0.31 seconds)
## Iteration 550: error is 0.862187 (50 iterations in 0.27 seconds)
## Iteration 600: error is 0.857630 (50 iterations in 0.28 seconds)
## Iteration 650: error is 0.853631 (50 iterations in 0.28 seconds)
```

```
## Iteration 700: error is 0.850819 (50 iterations in 0.28 seconds)
## Iteration 750: error is 0.848945 (50 iterations in 0.28 seconds)
## Iteration 800: error is 0.845557 (50 iterations in 0.28 seconds)
## Iteration 850: error is 0.843351 (50 iterations in 0.28 seconds)
## Iteration 900: error is 0.841236 (50 iterations in 0.27 seconds)
## Iteration 950: error is 0.839700 (50 iterations in 0.29 seconds)
## Iteration 1000: error is 0.838355 (50 iterations in 0.27 seconds)
## Fitting performed in 5.84 seconds.
## Read the 1000 x 30 data matrix successfully!
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Normalizing input...
## Building tree...
##  - point 0 of 1000
## Done in 0.10 seconds (sparsity = 0.136654)!
## Learning embedding...
## Iteration 50: error is 62.389513 (50 iterations in 0.39 seconds)
## Iteration 100: error is 60.332334 (50 iterations in 0.32 seconds)
## Iteration 150: error is 60.073780 (50 iterations in 0.33 seconds)
## Iteration 200: error is 59.951974 (50 iterations in 0.35 seconds)
## Iteration 250: error is 59.871223 (50 iterations in 0.34 seconds)
## Iteration 300: error is 1.473130 (50 iterations in 0.31 seconds)
## Iteration 350: error is 1.363409 (50 iterations in 0.32 seconds)
## Iteration 400: error is 1.327265 (50 iterations in 0.32 seconds)
## Iteration 450: error is 1.310381 (50 iterations in 0.32 seconds)
## Iteration 500: error is 1.302297 (50 iterations in 0.31 seconds)
## Iteration 550: error is 1.297383 (50 iterations in 0.29 seconds)
## Iteration 600: error is 1.293102 (50 iterations in 0.31 seconds)
## Iteration 650: error is 1.290777 (50 iterations in 0.32 seconds)
## Iteration 700: error is 1.287836 (50 iterations in 0.32 seconds)
## Iteration 750: error is 1.285465 (50 iterations in 0.29 seconds)
## Iteration 800: error is 1.283707 (50 iterations in 0.33 seconds)
## Iteration 850: error is 1.283283 (50 iterations in 0.33 seconds)
## Iteration 900: error is 1.281069 (50 iterations in 0.30 seconds)
## Iteration 950: error is 1.279503 (50 iterations in 0.31 seconds)
## Iteration 1000: error is 1.278670 (50 iterations in 0.30 seconds)
## Fitting performed in 6.40 seconds.
## using provided groups as a factor
```

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




```
## combine all
cds.all <- do.call(cbind, cds.filter)
mat.all <- normalizeCounts(cds.all)
preds <- lapply(models.info, function(modelA.info) {
  predictLds(mat.all, modelA.info$model, modelA.info$gsf)
})
```

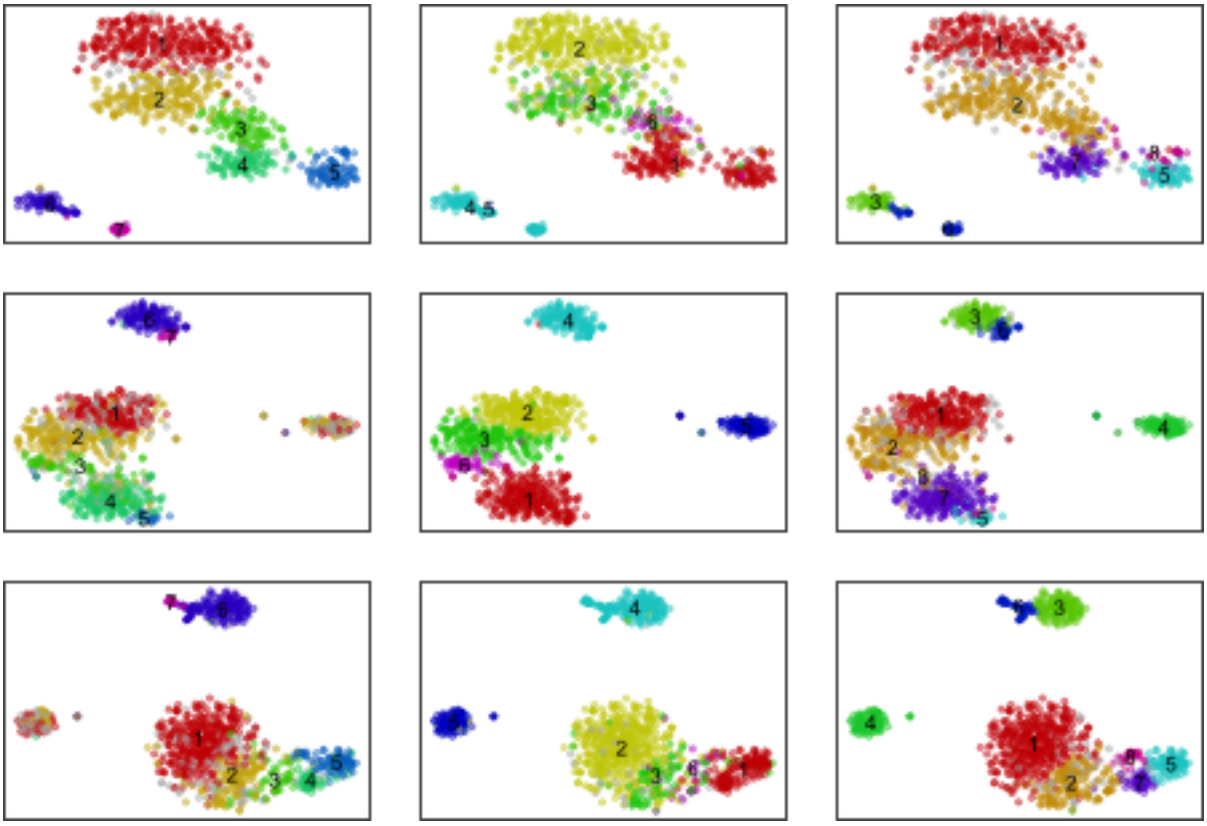
```
## [1] "Percentage of features retained: 1"
## [1] "Percentage of features retained: 1"
## [1] "Percentage of features retained: 1"
```

```
pred.com <- lapply(preds, function(p) {
  getConfidentPreds(p$posterior)
})
names(pred.com) <- names(cds)

## Plot with individual cluster annotations
par(mfrow=c(length(models.info),length(models.info)), mar=rep(1,4))
lapply(1:length(models.info), function(i) {
  lapply(1:length(models.info), function(j) {
    plotEmbedding(models.info[[i]]$emb$PCA, groups=pred.com[[j]], mark.clusters=TRUE, mark.cluster.cex =
  })
})
```

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

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



```
## [[1]]
## [[1]][[1]]
## NULL
##
## [[1]][[2]]
## NULL
##
## [[1]][[3]]
## NULL
##
##
## [[2]]
## [[2]][[1]]
## NULL
##
## [[2]][[2]]
## NULL
##
## [[2]][[3]]
## NULL
##
##
```

```
## [[3]]
## [[3]][[1]]
## NULL
##
## [[3]][[2]]
## NULL
##
## [[3]][[3]]
## NULL
```

```
## Lds
lds.all <- do.call(cbind, lapply(preds, function(p) p$x))

## Combine preds coms (use Ilya's idea of iterative refining)
com.all <- mergePredsList(pred.com=pred.com, min.group.size=10, t=1e-6)
```

```
## [1] "Former entropy:"
## [1] 1.823335
## [1] "New entropy:"
## [1] 2.177441
## [1] "Merge"
## foo
## 1.1 1.2 2.1 2.2 2.3 3.6 4.1 4.2 5.4 5.5 6.6 6.7 7.3 7.4 8.3 8.5
## 711  31  10 377 151 279  52  50  21 144  76  68  29 273  16  41
## [1] "Former entropy:"
## [1] 1.823335
## [1] "New entropy:"
## [1] 2.109892
## [1] "Merge"
## foo
## 1.2 1.3 2.1 2.2 2.3 2.6 3.4 4.5 5.1 6.4 7.1 8.1 8.6
## 805  26  47  87 318  77 277 184 164 144 328  38  16
```

```
## Reassess stability
comA <- getStableClusters(cds.all, com.all, mat.all, min.group.size=10, min.diff.genes=10, z.threshold=
```

```
## [1] "Comparing 3 and 6"
## [1] "Running differential expression with 2 clusters ... "
## [1] "Summarizing results ... "
## [1] "Differential genes found: "
## 3 6
## 4 4
## [1] "Comparing 5 and 7"
## [1] "Running differential expression with 2 clusters ... "
## [1] "Summarizing results ... "
## [1] "Differential genes found: "
## 5 7
## 4 4
## [1] "Comparing 4 and 8.1.2"
## [1] "Running differential expression with 2 clusters ... "
## [1] "Summarizing results ... "
## [1] "Differential genes found: "
##     4 8.1.2
##   129   274
## [1] "Comparing 8 and 1.2"
## [1] "Running differential expression with 2 clusters ... "
```

```
## [1] "Summarizing results ... "
## [1] "Differential genes found: "
## 1.2   8
##  85  70
## [1] "Comparing 1 and 2"
## [1] "Running differential expression with 2 clusters ... "
## [1] "Summarizing results ... "
## [1] "Differential genes found: "
## 1 2
## 4 4
```

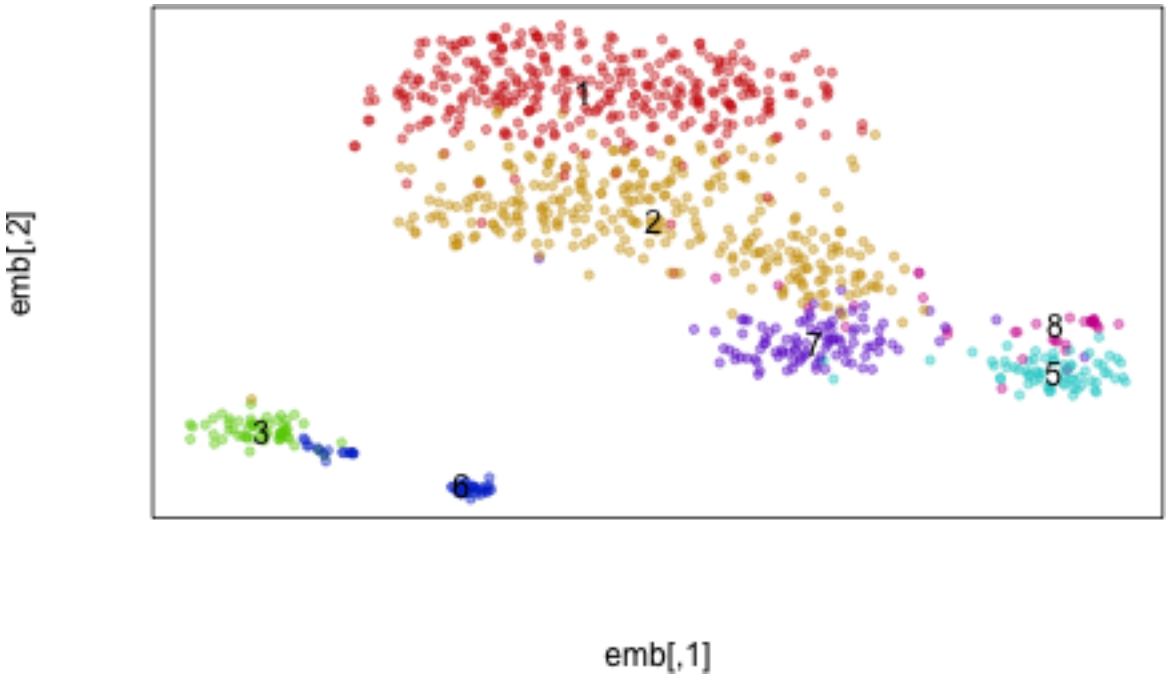
```
com.all.final <- factor(comA$com)
names(com.all.final) <- colnames(cds.all)
length(com.all.final)
```

```
## [1] 2921
```

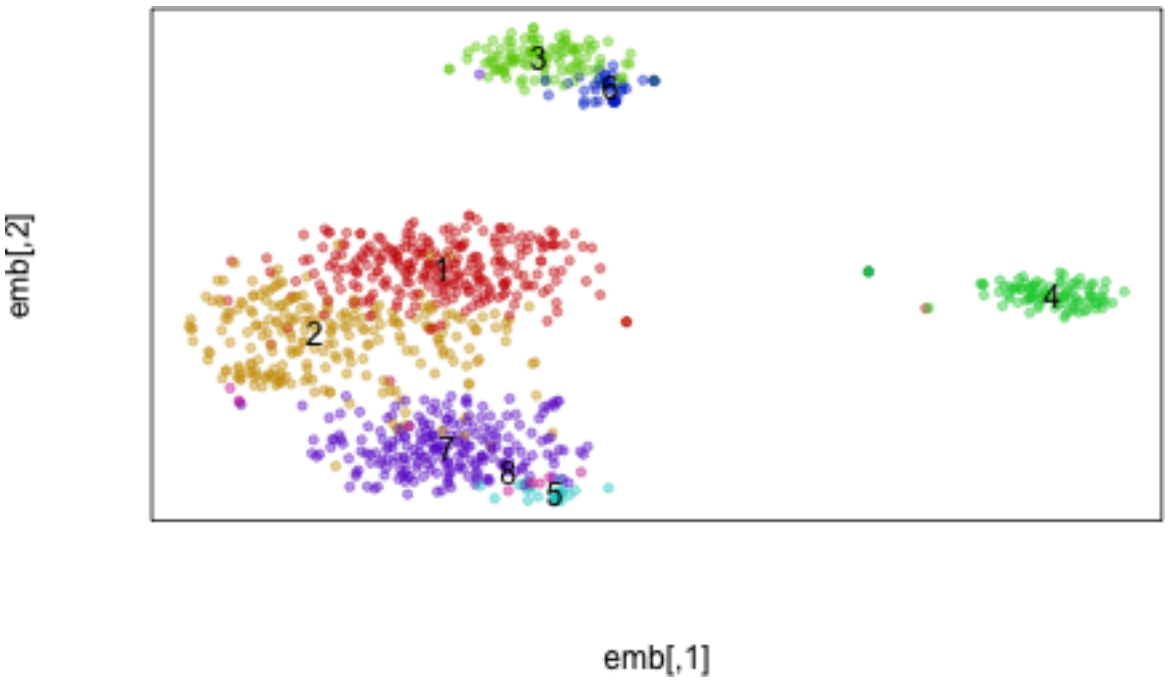
```
## Clean up
com.sub <- na.omit(com.all.final)
df.sub <- data.frame(celltype=com.sub, lds.all[names(com.sub),])
model <- MASS::lda(celltype ~ ., data=df.sub)
model.output <- predict(model, data.frame(lds.all))
com.all.fin <- model.output$class
names(com.all.fin) <- rownames(lds.all)

## Plot with joint clustering
lapply(1:length(models.info), function(i) {
  plotEmbedding(models.info[[i]]$emb$PCA, groups=com.all.fin, mark.clusters=TRUE, mark.cluster.cex = 1)
})
```

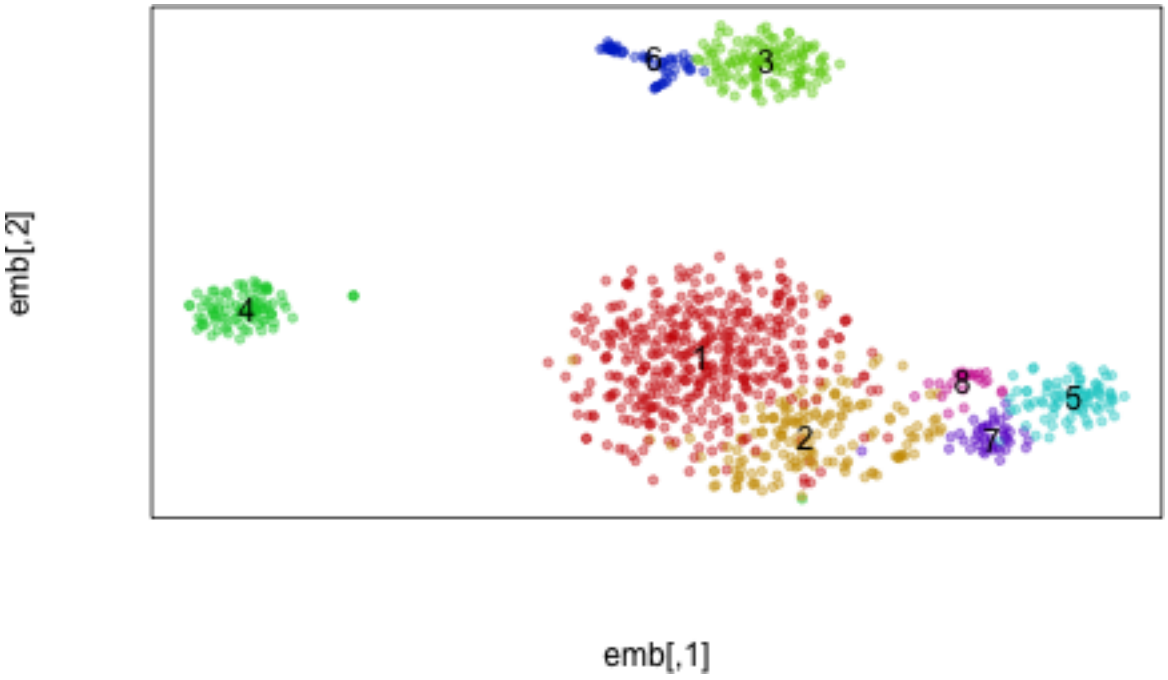
```
## using provided groups as a factor
```




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




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




```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
```

```
## Get common embedding
dim(lds.all)
```

```
## [1] 2921   18
```

```
length(com.all.fin)
```

```
## [1] 2921
```

```
lds.bc <- clusterBasedBatchCorrect(lds.all, sample, com.all.fin, min.group.size=30)
```

```
## Found 3 batches
## Adjusting for 0 covariate(s) or covariate level(s)
## Standardizing Data across genes
## Fitting L/S model and finding priors
## Finding parametric adjustments
## Adjusting the Data
## Found 3 batches
## Adjusting for 0 covariate(s) or covariate level(s)
## Standardizing Data across genes
## Fitting L/S model and finding priors
## Finding parametric adjustments
## Adjusting the Data
## Found 3 batches
## Adjusting for 0 covariate(s) or covariate level(s)
## Standardizing Data across genes
## Fitting L/S model and finding priors
## Finding parametric adjustments
## Adjusting the Data
## Found 2 batches
## Adjusting for 0 covariate(s) or covariate level(s)
## Standardizing Data across genes
## Fitting L/S model and finding priors
## Finding parametric adjustments
## Adjusting the Data
## Found 3 batches
## Adjusting for 0 covariate(s) or covariate level(s)
## Standardizing Data across genes
## Fitting L/S model and finding priors
## Finding parametric adjustments
## Adjusting the Data
## Found 3 batches
## Adjusting for 0 covariate(s) or covariate level(s)
## Standardizing Data across genes
## Fitting L/S model and finding priors
## Finding parametric adjustments
## Adjusting the Data
## Found 3 batches
## Adjusting for 0 covariate(s) or covariate level(s)
## Standardizing Data across genes
## Fitting L/S model and finding priors
## Finding parametric adjustments
## Adjusting the Data
```

```
dim(lds.bc)
```

```
## [1] 2921   18
```

```
emb.lds.bc <- Rtsne::Rtsne(lds.bc,
                           is_distance=FALSE,
                           perplexity=30,
                           num_threads=10,
                           verbose=FALSE)$Y
rownames(emb.lds.bc) <- rownames(lds.bc)

## final plot
par(mfrow=c(3,3), mar=rep(2,4))
## combined embedding with individual predicted cluster annotations
plotEmbedding(emb.lds.bc, groups=pred.com[[1]], mark.clusters=TRUE, mark.cluster.cex = 1)
```

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

```
plotEmbedding(emb.lds.bc, groups=pred.com[[2]], mark.clusters=TRUE, mark.cluster.cex = 1)
```

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

```
plotEmbedding(emb.lds.bc, groups=pred.com[[3]], mark.clusters=TRUE, mark.cluster.cex = 1)
```

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

```
## individual embedding with combined cluster annotations
plotEmbedding(models.info[[1]]$emb$PCA, groups=com.all.fin, mark.clusters=TRUE, mark.cluster.cex = 1)
```

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

```
plotEmbedding(models.info[[2]]$emb$PCA, groups=com.all.fin, mark.clusters=TRUE, mark.cluster.cex = 1)
```

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

```
plotEmbedding(models.info[[3]]$emb$PCA, groups=com.all.fin, mark.clusters=TRUE, mark.cluster.cex = 1)
```

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

```
## combined embedding
plotEmbedding(emb.lds.bc, groups=com.all, mark.clusters=TRUE, mark.cluster.cex = 1)
```

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

```
plotEmbedding(emb.lds.bc, groups=com.all.fin, mark.clusters=TRUE, mark.cluster.cex = 1)
```

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

```
plotEmbedding(emb.lds.bc, groups=sample, mark.clusters=TRUE, mark.cluster.cex = 1)
```

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