Demonstrating the Utility of MUDAN

Jean Fan

2017-11-23

To demonstrate the utility of `MUDAN`, we will analyze two 10X PBMC datasets from Zheng et al.

The data are already provided to you as counts matrices. We will load and subsample the data.

```
library(MUDAN)
data("pbmcA")
data("pbmcB")
print(dim(pbmcA))
```

```
## NULL
```

```
print(dim(pbmcB))
```

```
## [1] 15325  7765
```

```
set.seed(0)
## subsample cells for smaller dataset to run faster
vi <- sample(colnames(pbmcA), 2000)
pbmcA <- pbmcA[, vi]
vi <- sample(colnames(pbmcB), 2000)
pbmcB <- pbmcB[, vi]
```

First, we will combine the two datasets without batch correction and perform regular dimensionality reduction with PCA and visualization with tSNE.

```
## filter and combine
v1 <- rownames(pbmcA)[rowSums(pbmcA)>0]
v2 <- rownames(pbmcB)[rowSums(pbmcB)>0]
genes.int <- intersect(v1, v2)
length(genes.int)
```

```
## [1] 13003
```

```
cd <- cbind(pbmcA[genes.int,], pbmcB[genes.int,])
group <- factor(colnames(cd) %in% colnames(pbmcA), labels=c('pbmcA', 'pbmcB'))
names(group) <- colnames(cd)

## see separation by batch if we cluster all together
myMudanObject <- Mudan$new("comb", cd, ncores=4)
```

```
## [1] "Filtering matrix with 4000 cells and 13003 genes ..."
## [1] "Resulting matrix has 3701 cells and 12988 genes"
```

```
myMudanObject$libSizeNormalize()
```

```
## [1] "Normalizing matrix with 3701 cells and 12988 genes"
```

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

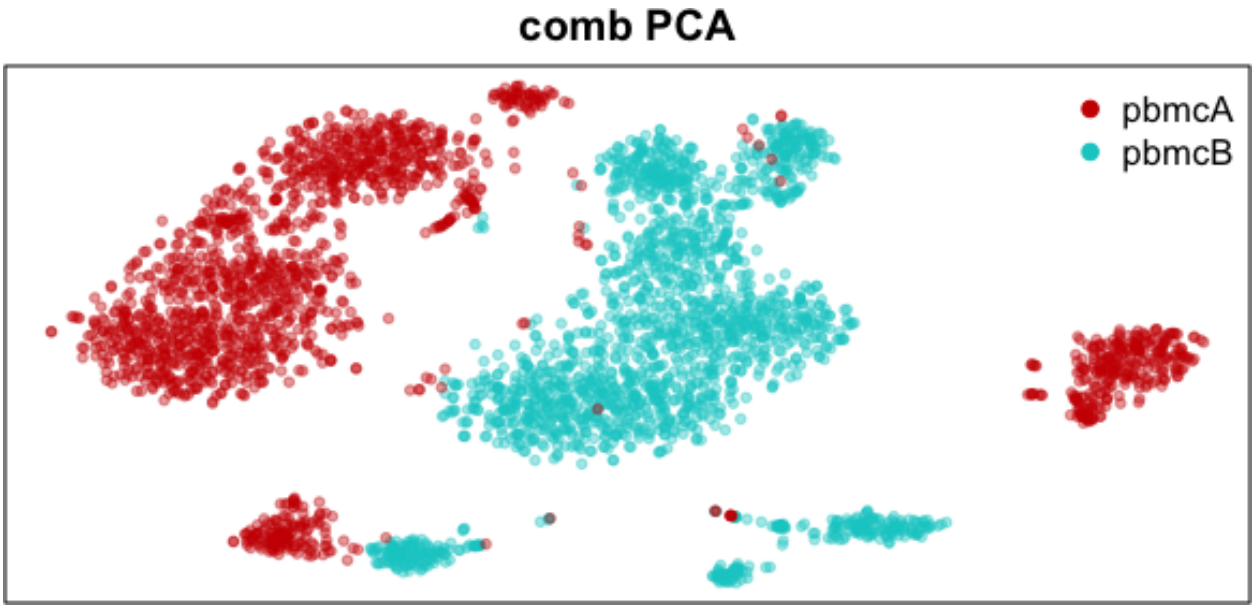
```
## [1] "Calculating variance fit ..."
## [1] "Using gam with k=5..."
## [1] "2613 overdispersed genes ... "
```

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

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

```
myMudanObject$getStandardEmbedding(plot=FALSE)
plotEmbedding(myMudanObject$emb[['PCA']], groups=group, main="comb PCA", show.legend=TRUE)
```

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



Indeed, we see prominent batch/patient differences. Traditionally, we would normalize out batch specific differences with batch correction using as using `combat`.

```
## apply combat batch correction
library(sva)
myMudanObjectbc <- Mudan$new("combbc", cd, ncores=4)
```

```
## [1] "Filtering matrix with 4000 cells and 13003 genes ..."
## [1] "Resulting matrix has 3701 cells and 12988 genes"
```

```
myMudanObjectbc$libSizeNormalize()
```

```
## [1] "Normalizing matrix with 3701 cells and 12988 genes"
```

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

```
## [1] "Calculating variance fit ..."
## [1] "Using gam with k=5..."
## [1] "2613 overdispersed genes ... "
```

```
cdbc <- sva::ComBat(dat=as.matrix(myMudanObjectbc$matnorm), batch=group[colnames(myMudanObjectbc$matnorm
```

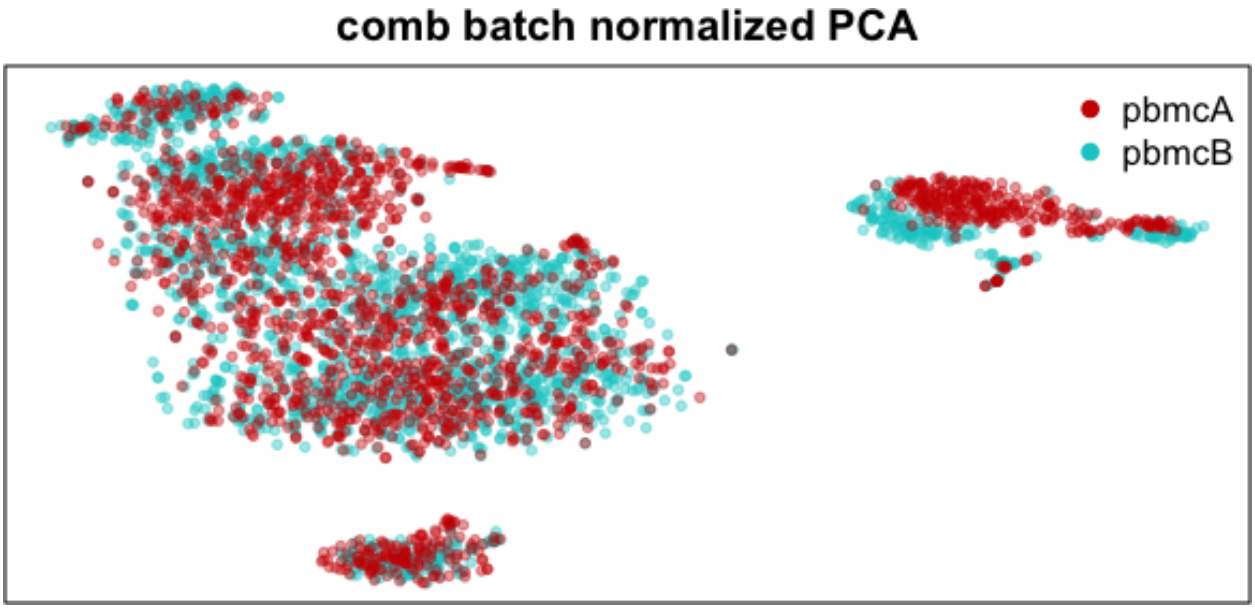
```
## Found 2 batches
## Adjusting for 0 covariate(s) or covariate level(s)
## Standardizing Data across genes
## Fitting L/S model and finding priors
## Finding parametric adjustments
## Adjusting the Data
```

```
myMudanObjectbc$matnorm <- cdbc
myMudanObjectbc$dimensionalityReduction(nGenes = 1000, nPcs = 30, maxit=1000)
```

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

```
myMudanObjectbc$getStandardEmbedding(plot=FALSE)
plotEmbedding(myMudanObjectbc$emb[['PCA']], groups=group, main="comb batch normalized PCA", show.legend=
```

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



However, we will show later that this batch correction "over-corrects", removing true biological signal and obscures our ability to identify the appropriate cell subtypes.

Let's first analyze each dataset separately.

```
## analyze pbmcA by itself
myMudanObject1 <- Mudan$new("pbmcA", pbmcA, ncores=4)
```

```
## [1] "Filtering matrix with 2000 cells and 13939 genes ..."
## [1] "Resulting matrix has 1928 cells and 13246 genes"
```

```
myMudanObject1$libSizeNormalize()
```

```
## [1] "Normalizing matrix with 1928 cells and 13246 genes"
```

```
myMudanObject1$varianceNormalize(plot=FALSE)
```

```
## [1] "Calculating variance fit ..."
## [1] "Using gam with k=5..."
## [1] "2482 overdispersed genes ... "
```

```
myMudanObject1$dimensionalityReduction(nGenes = 1000, nPcs = 30, maxit=1000)
```

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

```
myMudanObject1$getStandardEmbedding(plot=FALSE)
myMudanObject1$communityDetection(reductionType='pcs', communityName="Infomap", communityMethod=igraph:
```

```
## [1] "finding approximate nearest neighbors ..."
## [1] "calculating clustering ..."
```

```
## [1] "graph modularity: 0.622781014403065"
## [1] "identifying cluster membership ..."
## com
##   1   2   3   4   5   6   7   8   9
## 621 408 205 189 161 130 122  53  39
```

```
myMudanObject1$modelCommunity(communityName="Infomap")
```

```
## [1] "calculating LDA ..."
## [1] "LDA prediction accuracy ..."
##
## FALSE  TRUE
##     2  1926
```

```
myMudanObject1$getMudanEmbedding(plot=FALSE)
```

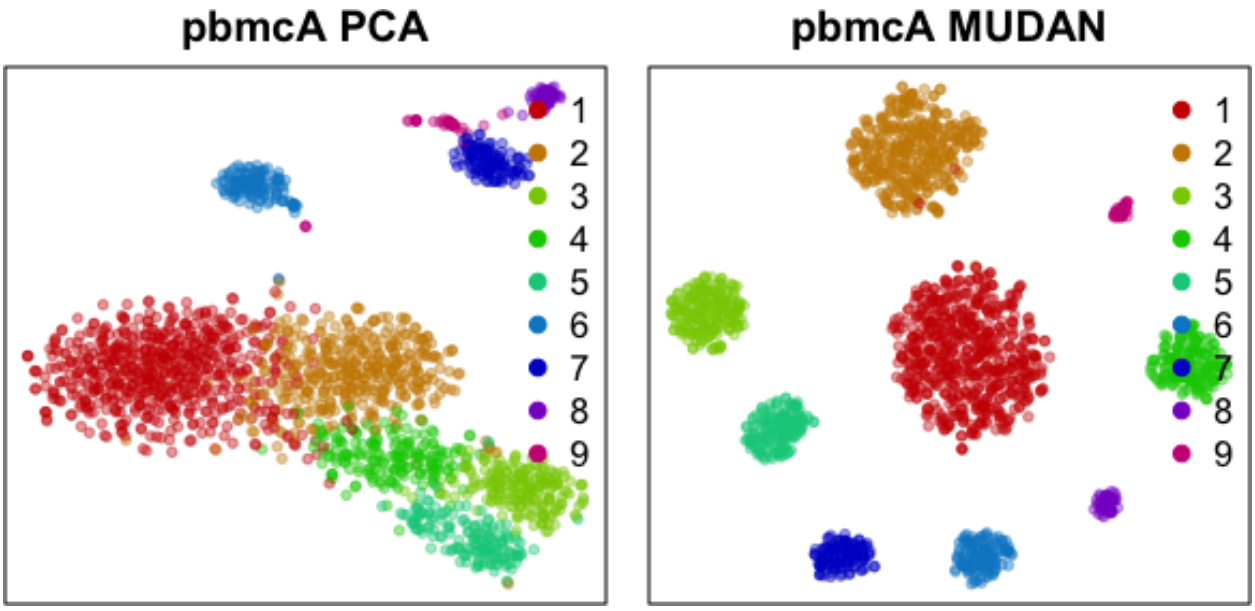
```
## [1] "Running LDA models ..."
## [1] "Running Rtsne with perplexity 30"
## Read the 1928 x 8 data matrix successfully!
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Normalizing input...
## Building tree...
##  - point 0 of 1928
## Done in 0.17 seconds (sparsity = 0.062285)!
## Learning embedding...
## Iteration 50: error is 69.736790 (50 iterations in 0.97 seconds)
## Iteration 100: error is 61.072022 (50 iterations in 0.76 seconds)
## Iteration 150: error is 59.965073 (50 iterations in 0.85 seconds)
## Iteration 200: error is 59.497018 (50 iterations in 0.92 seconds)
## Iteration 250: error is 59.225505 (50 iterations in 0.82 seconds)
## Iteration 300: error is 1.722273 (50 iterations in 0.73 seconds)
## Iteration 350: error is 1.498755 (50 iterations in 0.69 seconds)
## Iteration 400: error is 1.390400 (50 iterations in 0.81 seconds)
## Iteration 450: error is 1.339452 (50 iterations in 0.84 seconds)
## Iteration 500: error is 1.316452 (50 iterations in 0.75 seconds)
## Iteration 550: error is 1.302469 (50 iterations in 0.85 seconds)
## Iteration 600: error is 1.291751 (50 iterations in 0.74 seconds)
## Iteration 650: error is 1.282878 (50 iterations in 0.75 seconds)
## Iteration 700: error is 1.275738 (50 iterations in 0.87 seconds)
## Iteration 750: error is 1.269143 (50 iterations in 0.79 seconds)
## Iteration 800: error is 1.264489 (50 iterations in 0.74 seconds)
## Iteration 850: error is 1.260463 (50 iterations in 0.87 seconds)
## Iteration 900: error is 1.256168 (50 iterations in 0.76 seconds)
## Iteration 950: error is 1.251852 (50 iterations in 1.01 seconds)
## Iteration 1000: error is 1.249351 (50 iterations in 0.85 seconds)
## Fitting performed in 16.37 seconds.
```

```
par(mfrow=c(1,2))
myMudanObject1$plot(reductionType='pcs', communityName="Infomap", embeddingType="PCA", main='pbmcA PCA'
```

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

```
myMudanObject1$plot(reductionType='pcs', communityName="Infomap", embeddingType="MUDAN", main='pbmcA MU
```

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




```
## analyze pbmcB by itself
myMudanObject2 <- Mudan$new("pbmcB", pbmcB, ncores=4)
```

```
## [1] "Filtering matrix with 2000 cells and 15325 genes ..."
## [1] "Resulting matrix has 1777 cells and 12751 genes"
```

```
myMudanObject2$libSizeNormalize()
```

```
## [1] "Normalizing matrix with 1777 cells and 12751 genes"
```

```
myMudanObject2$varianceNormalize(plot=FALSE)
```

```
## [1] "Calculating variance fit ..."
## [1] "Using gam with k=5..."
## [1] "2238 overdispersed genes ... "
```

```
myMudanObject2$dimensionalityReduction(nGenes = 1000, nPcs = 30, maxit=1000)
```

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

```
myMudanObject2$getStandardEmbedding(plot=FALSE)
myMudanObject2$communityDetection(reductionType='pcs', communityName="Infomap", communityMethod=igraph:
```

```
## [1] "finding approximate nearest neighbors ..."
## [1] "calculating clustering ..."
## [1] "graph modularity: 0.636461655211348"
## [1] "identifying cluster membership ..."
## com
##   1   2   3   4   5   6   7   8   9
## 476 415 353 218 140  60  64  35  16
```

```
myMudanObject2$modelCommunity(communityName="Infomap")
```

```
## [1] "calculating LDA ..."
## [1] "LDA prediction accuracy ..."
##
## FALSE  TRUE
##     2  1775
```

```
myMudanObject2$getMudanEmbedding(plot=FALSE)
```

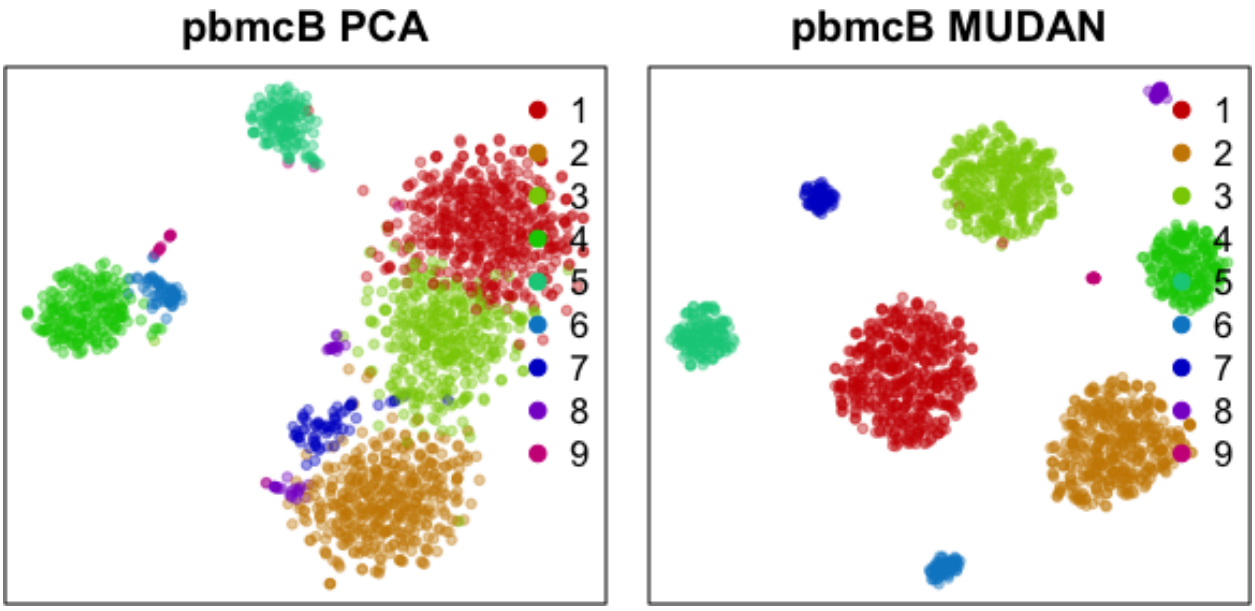
```
## [1] "Running LDA models ..."
## [1] "Running Rtsne with perplexity 30"
## Read the 1777 x 8 data matrix successfully!
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Normalizing input...
## Building tree...
##  - point 0 of 1777
## Done in 0.15 seconds (sparsity = 0.069111)!
## Learning embedding...
## Iteration 50: error is 67.379531 (50 iterations in 0.83 seconds)
## Iteration 100: error is 60.114044 (50 iterations in 0.67 seconds)
## Iteration 150: error is 59.159200 (50 iterations in 0.73 seconds)
## Iteration 200: error is 58.738566 (50 iterations in 0.67 seconds)
## Iteration 250: error is 58.490979 (50 iterations in 0.71 seconds)
## Iteration 300: error is 1.729501 (50 iterations in 0.64 seconds)
## Iteration 350: error is 1.515921 (50 iterations in 0.61 seconds)
## Iteration 400: error is 1.415461 (50 iterations in 0.62 seconds)
## Iteration 450: error is 1.363509 (50 iterations in 0.62 seconds)
## Iteration 500: error is 1.342554 (50 iterations in 0.64 seconds)
## Iteration 550: error is 1.327606 (50 iterations in 0.62 seconds)
## Iteration 600: error is 1.316966 (50 iterations in 0.63 seconds)
## Iteration 650: error is 1.306467 (50 iterations in 0.64 seconds)
## Iteration 700: error is 1.297730 (50 iterations in 0.65 seconds)
## Iteration 750: error is 1.291023 (50 iterations in 0.67 seconds)
## Iteration 800: error is 1.284961 (50 iterations in 0.64 seconds)
## Iteration 850: error is 1.280630 (50 iterations in 0.63 seconds)
## Iteration 900: error is 1.277302 (50 iterations in 0.65 seconds)
## Iteration 950: error is 1.274689 (50 iterations in 0.67 seconds)
## Iteration 1000: error is 1.271647 (50 iterations in 0.65 seconds)
## Fitting performed in 13.19 seconds.
```

```
par(mfrow=c(1,2))
myMudanObject2$plot(reductionType='pcs', communityName="Infomap", embeddingType="PCA", main='pbmcB PCA'
```

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

```
myMudanObject2$plot(reductionType='pcs', communityName="Infomap", embeddingType="MUDAN", main='pbmcB MU
```

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



Now, we will learn from each dataset analyzed individually and project the combined dataset.

```
## project and combine (need to clean up into function)
v1 <- rownames(pbmcA)[rowSums(pbmcA)>30]
v2 <- rownames(pbmcB)[rowSums(pbmcB)>30]
genes.int <- intersect(v1, v2)
length(genes.int)
```

```
## [1] 5990
```

```
m1 <- as.matrix(myMudanObject1$mat[genes.int,])
m2 <- as.matrix(myMudanObject2$mat[genes.int,])
mall <- cbind(m1, m2)
combat <- sva::ComBat(dat=mall, batch=group[colnames(mall)]) ## remove batch effect
```

```
## 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
```

```
m1c <- combat[, colnames(m1)]
m2c <- combat[, colnames(m2)]
## maximize axis of separation
nGenes <- 500
vargenes <- getVariableGenes(myMudanObject1$matnorm[genes.int,], nGenes)
model1 <- modelLda(mat=m1c[vargenes,], com=myMudanObject1$com[['pcs']][['Infomap']])
```

```
## [1] "calculating LDA ..."
## [1] "LDA prediction accuracy ..."
##
## FALSE  TRUE
##    48  1880
```

```
vargenes <- getVariableGenes(myMudanObject2$matnorm[genes.int,], nGenes)
model2 <- modelLda(mat=m2c[vargenes,], com=myMudanObject2$com[['pcs']][['Infomap']])
```

```
## [1] "calculating LDA ..."
## [1] "LDA prediction accuracy ..."
##
## FALSE  TRUE
##    38  1739
```

```
## project
lds1 <- predict(model1, data.frame(t(combat)))$x
lds2 <- predict(model2, data.frame(t(combat)))$x
## get embedding on projections
reduction <- cbind(lds1, lds2)
emb <- Rtsne::Rtsne(reduction, is_distance=FALSE, perplexity=60, verbose=TRUE, num_threads=2)$Y
```

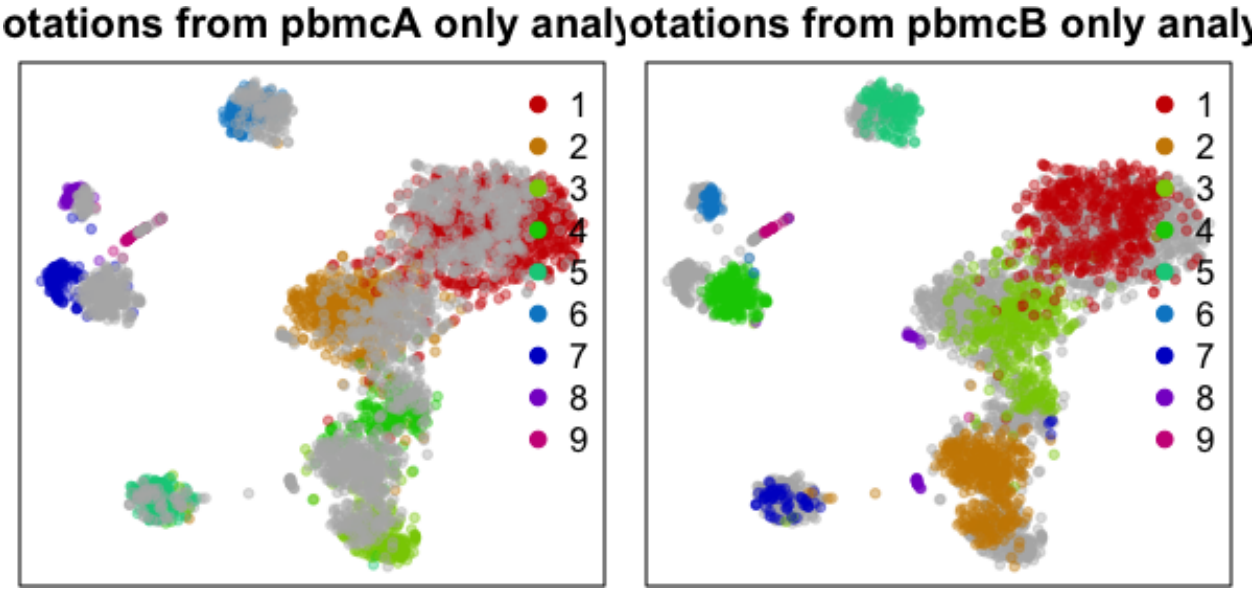
```
## Read the 3705 x 16 data matrix successfully!
## Using no_dims = 2, perplexity = 60.000000, and theta = 0.500000
## Computing input similarities...
## Normalizing input...
## Building tree...
##  - point 0 of 3705
## Done in 1.01 seconds (sparsity = 0.066306)!
## Learning embedding...
## Iteration 50: error is 76.291638 (50 iterations in 2.15 seconds)
## Iteration 100: error is 64.643037 (50 iterations in 2.01 seconds)
## Iteration 150: error is 63.724403 (50 iterations in 1.95 seconds)
## Iteration 200: error is 63.457379 (50 iterations in 1.85 seconds)
## Iteration 250: error is 63.320726 (50 iterations in 1.82 seconds)
## Iteration 300: error is 1.871757 (50 iterations in 1.79 seconds)
## Iteration 350: error is 1.673055 (50 iterations in 1.75 seconds)
## Iteration 400: error is 1.586690 (50 iterations in 1.67 seconds)
## Iteration 450: error is 1.540213 (50 iterations in 1.67 seconds)
## Iteration 500: error is 1.513341 (50 iterations in 1.70 seconds)
## Iteration 550: error is 1.497984 (50 iterations in 1.69 seconds)
## Iteration 600: error is 1.486267 (50 iterations in 1.77 seconds)
## Iteration 650: error is 1.478722 (50 iterations in 1.67 seconds)
## Iteration 700: error is 1.473960 (50 iterations in 1.67 seconds)
## Iteration 750: error is 1.469328 (50 iterations in 1.66 seconds)
## Iteration 800: error is 1.465053 (50 iterations in 1.66 seconds)
## Iteration 850: error is 1.461626 (50 iterations in 1.68 seconds)
## Iteration 900: error is 1.458666 (50 iterations in 1.66 seconds)
## Iteration 950: error is 1.456300 (50 iterations in 1.72 seconds)
## Iteration 1000: error is 1.454106 (50 iterations in 1.63 seconds)
## Fitting performed in 35.18 seconds.
```

```
rownames(emb) <- rownames(reduction)
## look at new embedding with old annotations
par(mfrow=c(1,2))
plotEmbedding(emb, groups=myMudanObject1$com[['pcs']][['Infomap']], show.legend=TRUE, main='annotations
```

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

```
plotEmbedding(emb, groups=myMudanObject2$com[['pcs']][['Infomap']], show.legend=TRUE, main='annotations
```

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



We can get new annotations from the projection and compare with our previous results.

```
## get new annotation
com <- getComMembership(t(reduction), k=100, method=igraph::cluster_infomap)
```

```
## [1] "finding approximate nearest neighbors ..."
## [1] "calculating clustering ..."
## [1] "graph modularity: 0.72818369052139"
## [1] "identifying cluster membership ..."
## com
##    1    2    3    4    5    6    7    8    9   10   11
## 1069  717  385  381  304  277  235  220  114    2    1
```

```
par(mfrow=c(1,3))
plotEmbedding(emb, groups=com)
```

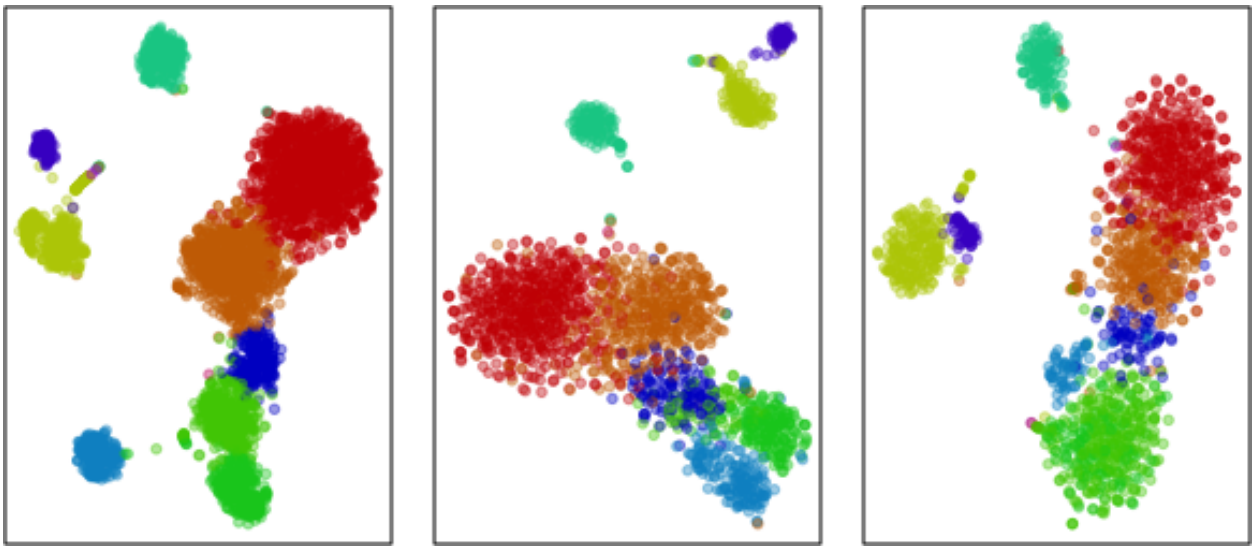
```
## using provided groups as a factor
```

```
plotEmbedding(myMudanObject1$emb[['PCA']], groups=com[rownames(myMudanObject1$emb[['PCA']])])
```

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

```
plotEmbedding(myMudanObject2$emb[['PCA']], groups=com[rownames(myMudanObject2$emb[['PCA']])])
```

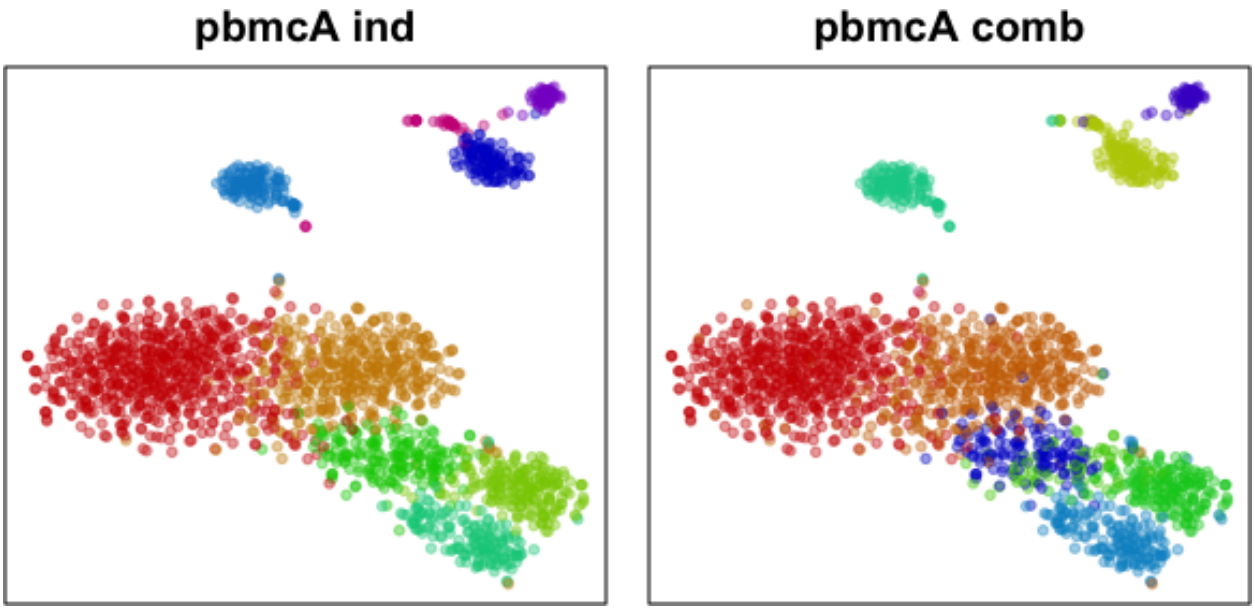
```
## using provided groups as a factor
```



```
## plot all together
par(mfrow=c(1,2))
myMudanObject1$plot(reductionType='pcs', communityName="Infomap", embeddingType="PCA", main='pbmcA ind'
```

```
## using provided groups as a factor
plotEmbedding(myMudanObject1$emb[['PCA']], groups=com[rownames(myMudanObject1$emb[['PCA']])], main='pbm
```

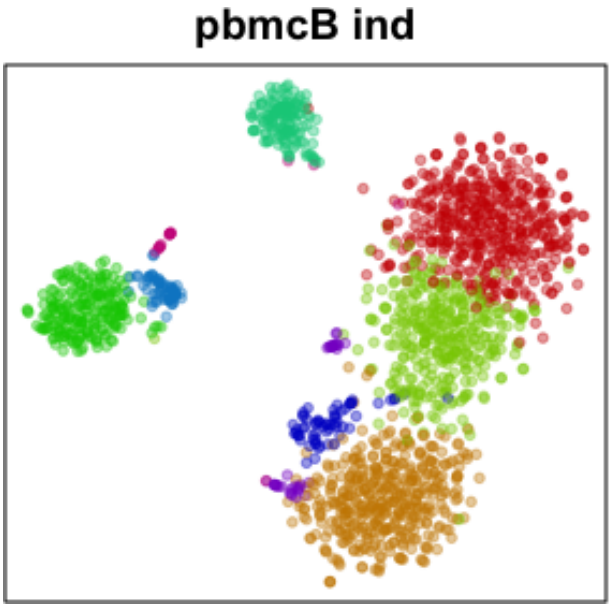
```
## using provided groups as a factor
```

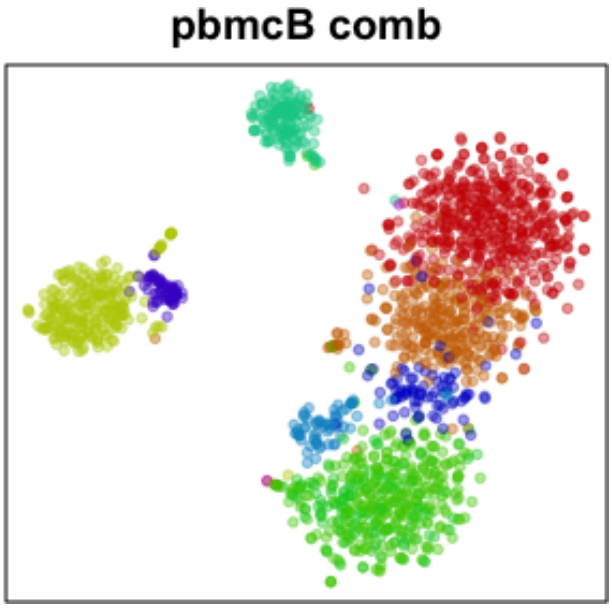



```
myMudanObject2$plot(reductionType='pcs', communityName="Infomap", embeddingType="PCA", main='pbmcB ind'
```

```
## using provided groups as a factor
plotEmbedding(myMudanObject2$emb[['PCA']], groups=com[rownames(myMudanObject2$emb[['PCA']])], main='pbm
```

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



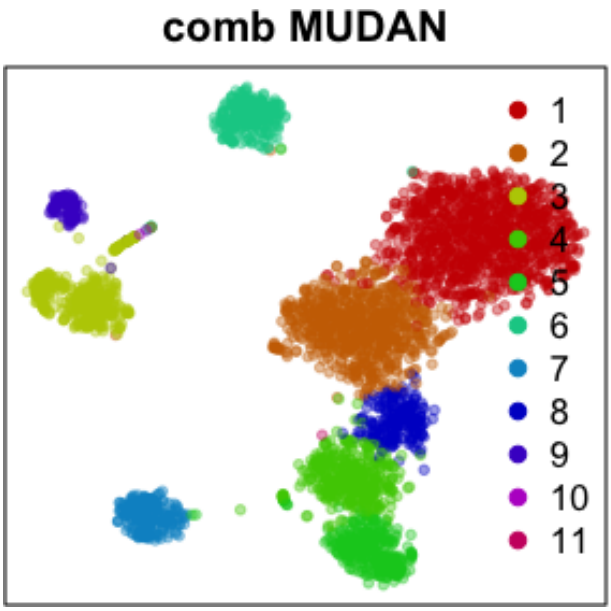


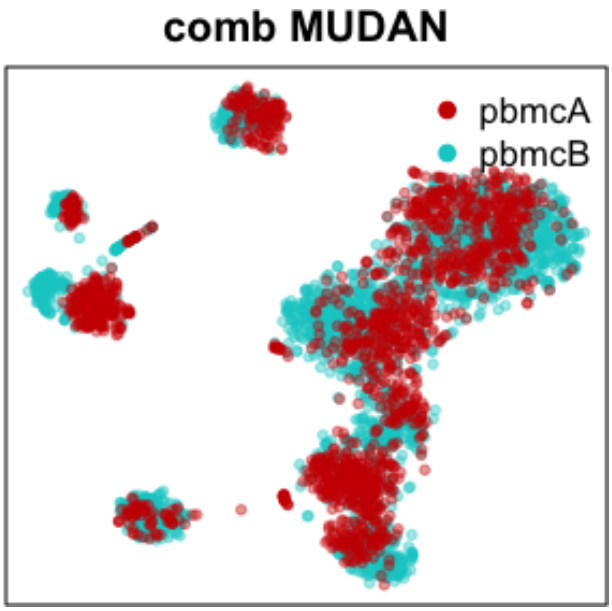

```
## some groups more prominent in pbmcB now separated in pbmcA and vice versa
par(mfrow=c(1,2))
plotEmbedding(emb, groups=com, main="comb MUDAN", show.legend=TRUE)
```

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

```
plotEmbedding(emb, groups=group, main="comb MUDAN", show.legend=TRUE)
```

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



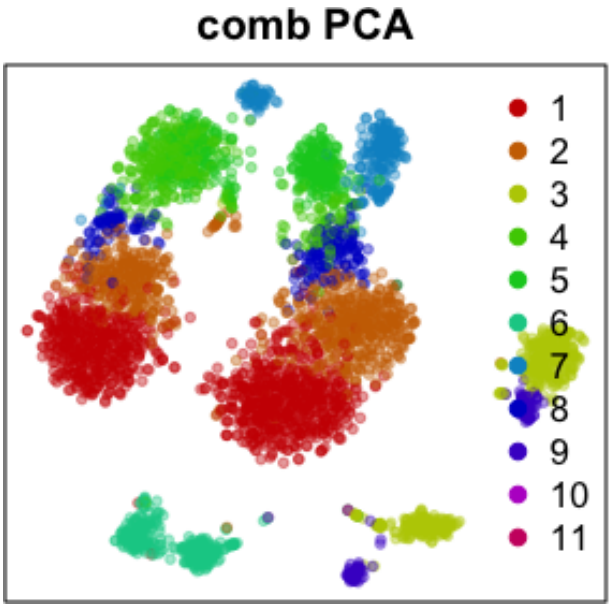


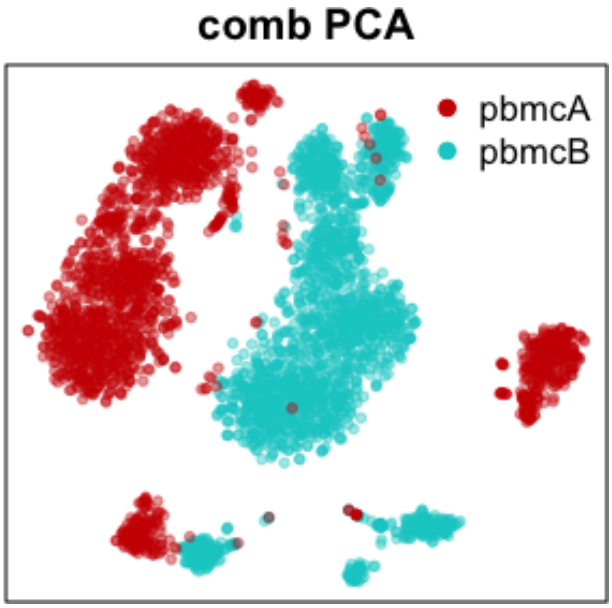

```
plotEmbedding(myMudanObject$emb[['PCA']], groups=com, main="comb PCA", show.legend=TRUE)
```

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

```
plotEmbedding(myMudanObject$emb[['PCA']], groups=group, main="comb PCA", show.legend=TRUE)
```

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



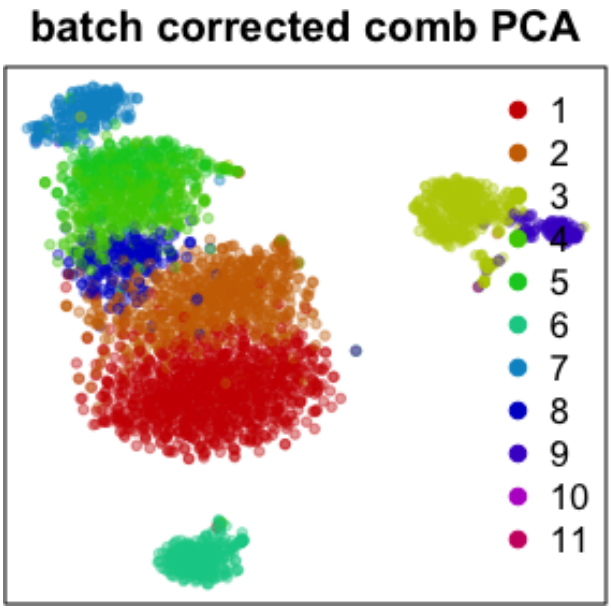


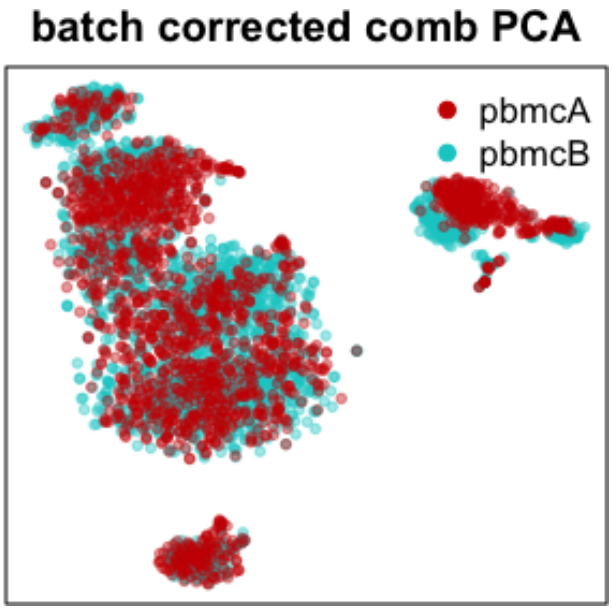

```
plotEmbedding(myMudanObjectbc$emb[['PCA']], groups=com, main="batch corrected comb PCA", show.legend=TRU
```

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

```
plotEmbedding(myMudanObjectbc$emb[['PCA']], groups=group, main="batch corrected comb PCA", show.legend=
```

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





Replot with shuffled colors.

```
par(mfrow=c(1,3))
plotEmbedding(emb, groups=com, main="comb MUDAN", shuffle.colors=TRUE)
```

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

```
plotEmbedding(myMudanObject$emb[['PCA']], groups=com, main="comb PCA", shuffle.colors=TRUE)
```

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

```
plotEmbedding(myMudanObjectbc$emb[['PCA']], groups=com, main="batch corrected comb PCA", shuffle.colors=
```

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