Highlighting the features of MUDAN

Jean Fan

2018-02-21

```
library(MUDAN)
```

MUDAN features

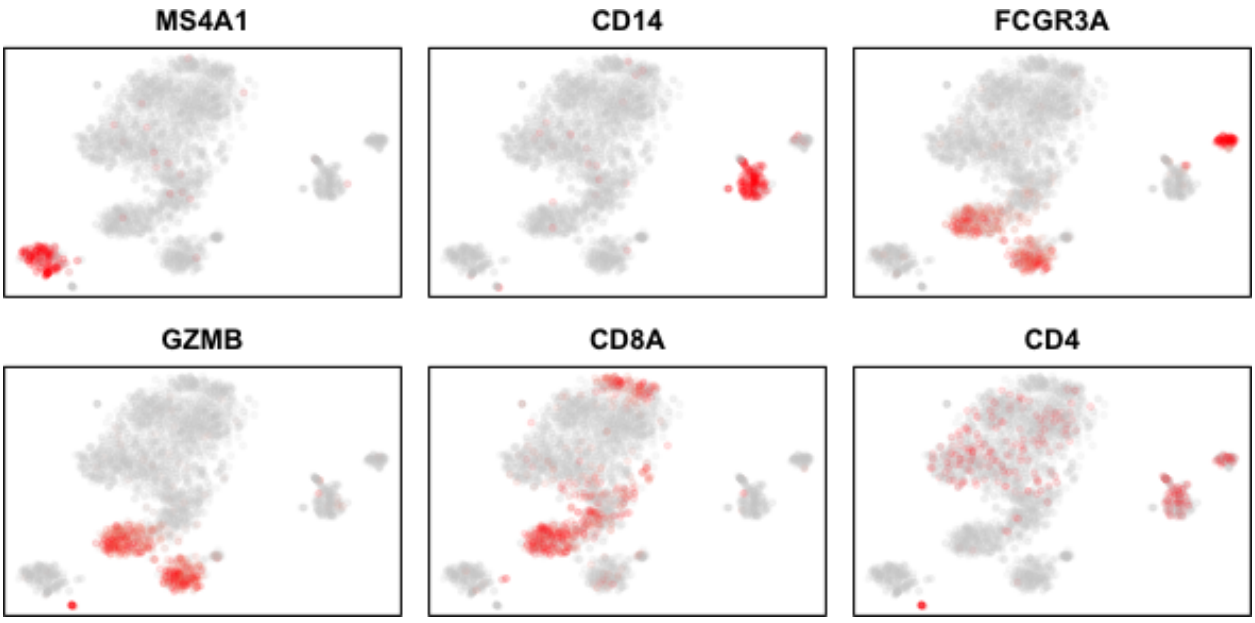
(1) Fast, flexible analysis

```
data(pbmcA) ## load built in 10X pbmcA dataset
## downsample for testing purposes only
#pbmcA <- as.matrix(pbmcA[, 1:1000])
pbmcA <- as.matrix(pbmcA)

start_time <- Sys.time()

## filter out poor genes and cells
cd <- cleanCounts(pbmcA,
                  min.reads = 10,
                  min.detected = 10,
                  verbose=FALSE)
## CPM normalization
mat <- normalizeCounts(cd,
                       verbose=FALSE)
## variance normalize, identify overdispersed genes
matnorm.info <- normalizeVariance(mat,
                                  details=TRUE,
                                  verbose=FALSE)
## log transform
matnorm <- log10(matnorm.info$mat+1)
## 30 PCs on overdispersed genes
pcs <- getPcs(matnorm[matnorm.info$ods,],
              nGenes=length(matnorm.info$ods),
              nPcs=30,
              verbose=FALSE)
## get tSNE embedding on PCs
emb <- Rtsne::Rtsne(pcs,
                    is_distance=FALSE,
                    perplexity=30,
                    num_threads=parallel::detectCores(),
                    verbose=FALSE)$Y
rownames(emb) <- rownames(pcs)

end_time <- Sys.time()
print(paste0("Analysis of ",
             ncol(cd), " cells and ",
```

```
            nrow(cd), " genes took ",
            end_time - start_time, " seconds"))
```

```
## [1] "Analysis of 2896 cells and 10458 genes took 37.917484998703 seconds"
```

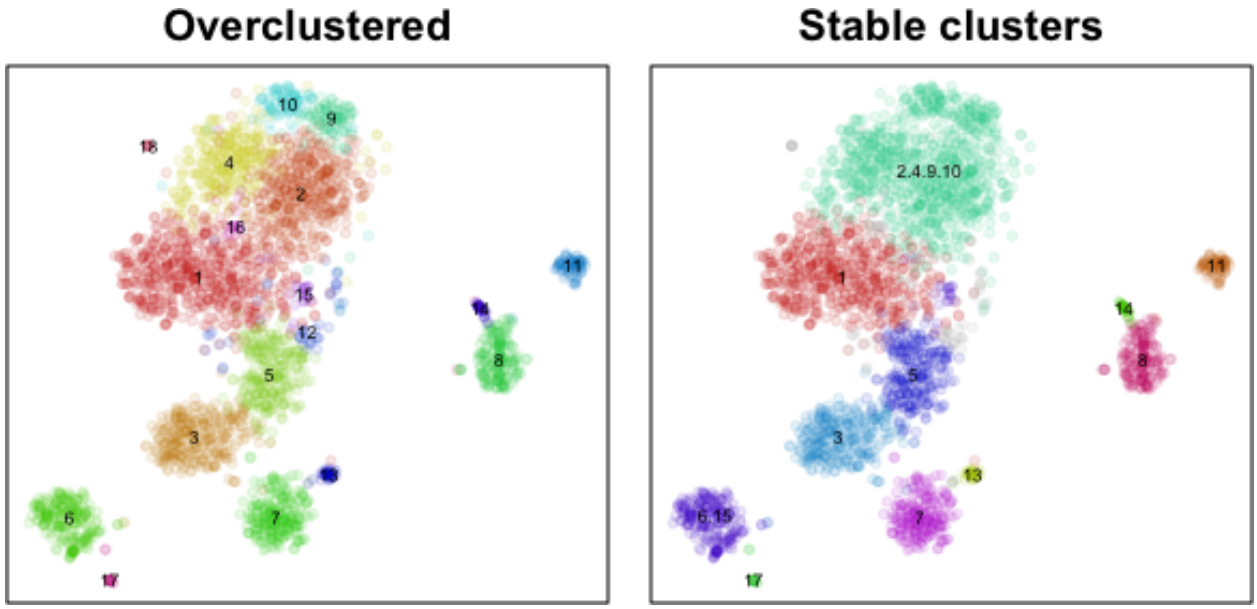
```
## plot expression of marker genes
marker.genes <- c('MS4A1', 'CD14', 'FCGR3A', 'GZMB', 'CD8A', 'CD4')
par(mfrow=c(2,3), mar=c(0.5,0.5,2,0.5))
invisible(lapply(marker.genes, function(g) {
  gcol <- cd[g,] ## plot a gene
  plotEmbedding(emb,
                color=gcol,
                mark.clusters=TRUE,
                main=g, xlab=NA, ylab=NA,
                verbose=FALSE, alpha=0.1)
}))
```



(2) Graph-based subpopulation detection and subpopulation stability analysis

```
## graph-based community detection; over cluster with small k
com <- getComMembership(pcs,
                        k=15, method=igraph::cluster_infomap,
                        verbose=FALSE)

## get stable clusters
stable <- getStableClusters(cd, com, matnorm,
                            min.group.size=10, min.diff.genes=10,
                            z.threshold=1.96,
                            verbose=FALSE, plot=FALSE)

## plot
par(mfrow=c(1,2), mar=c(0.5,0.5,2,0.5))
plotEmbedding(emb, com,
```

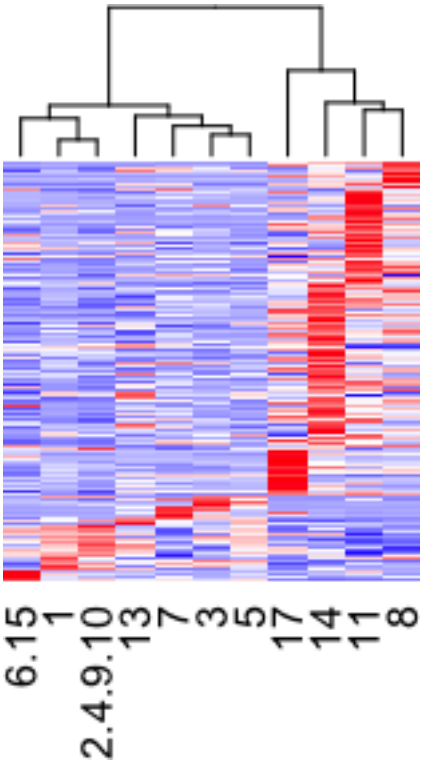
```
              main='Overclustered', xlab=NA, ylab=NA,
              mark.clusters=TRUE, alpha=0.1, mark.cluster.cex=0.5,
              verbose=FALSE) ## plot
plotEmbedding(emb, groups=stable$com,
              main='Stable clusters', xlab=NA, ylab=NA,
              mark.clusters=TRUE, alpha=0.1, mark.cluster.cex=0.5,
              verbose=FALSE)
```




```
## plot significant differential genes as heatmap
dg <- getDifferentialGenes(cd, stable$com)
```

```
## [1] "Running differential expression with 11 clusters ... "
## [1] "Summarizing results ... "
```

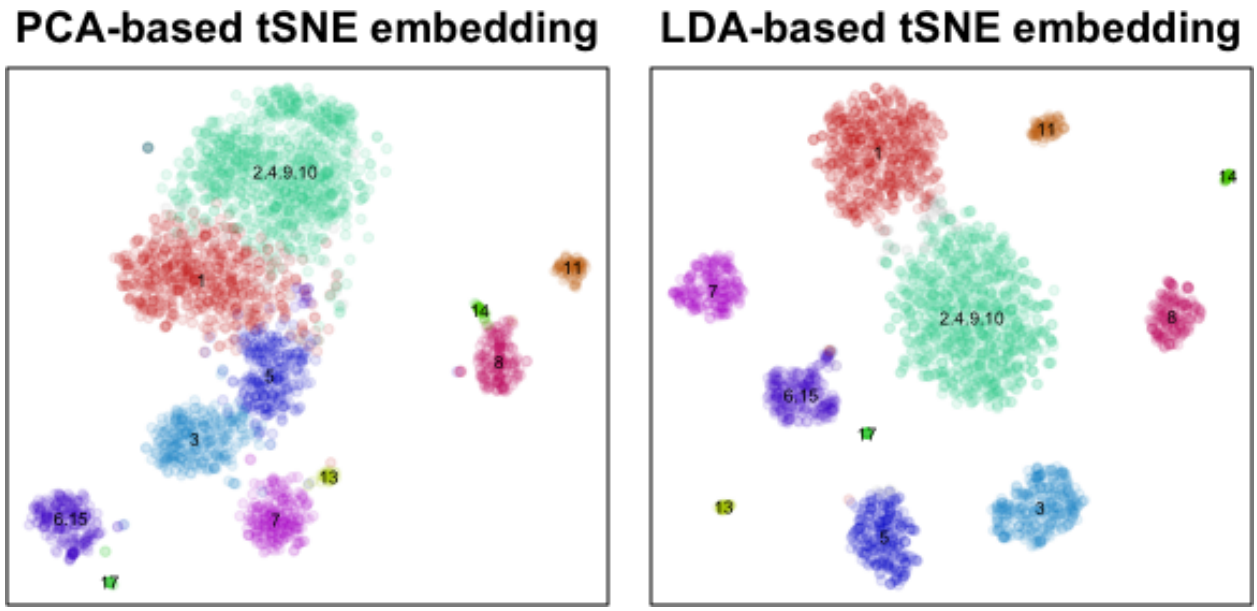
```
dg <- dg[stable$hc$labels[stable$hc$order]]
dg.sig <- unlist(lapply(dg, function(x) {
  x <- x[x$Z>1.96,] ## significant z-score
  x <- x[x$highest,] ## must be highest in this group (optimal markers)
  rownames(x)
}))
dg.sig <- intersect(dg.sig, rownames(stable$mat.summary))
m <- stable$mat.summary[dg.sig,]
heatmap(m,
        Rowv=NA, Colv=as.dendrogram(stable$hc),
        col=colorRampPalette(c("blue", "white", "red"))(100),
        scale="row", trace="none", labRow=NA)
```



(3) LDA-based embedding to optimally separate clusters for visualization

```
## train LDA model
## train on detected significantly differentially expressed genes among stable clusters
genes <- intersect(unique(unlist(stable$pv.sig)), rownames(matnorm))
mn <- matnorm[genes,]
model <- modelLda(mat=mn, com=stable$com,
                  retest=FALSE, verbose=FALSE)
## remember normalization parameters
gsf <- matnorm.info$df$gsf
names(gsf) <- rownames(matnorm.info$df)
ods <- rownames(matnorm.info$mat)[matnorm.info$ods]
## project to LD space
preds <- predict(model, data.frame(t(log10(as.matrix(mat[names(gsf),]*gsf+1)))))
lds <- preds$x
class <- getConfidentPreds(preds$posterior)
emb.lds <- Rtsne::Rtsne(lds,
                        is_distance=FALSE,
                        perplexity=30,
                        num_threads=parallel::detectCores(),
                        verbose=FALSE)$Y
rownames(emb.lds) <- rownames(lds)

## compare
par(mfrow=c(1,2), mar=c(0.5,0.5,2,0.5))
plotEmbedding(emb, groups=class,
              main='PCA-based tSNE embedding', xlab=NA, ylab=NA,
              mark.clusters=TRUE, alpha=0.1, mark.cluster.cex=0.5,
```

```
              verbose=FALSE)
plotEmbedding(emb.lds, groups=class,
              main='LDA-based tSNE embedding', xlab=NA, ylab=NA,
              mark.clusters=TRUE, alpha=0.1, mark.cluster.cex=0.5,
              verbose=FALSE)
```



(4) Projecting new samples into same LD-space to derive common embedding

```
## load new dataset
data(pbmcC)
## downsample for testing purposes only
#pbmcC <- as.matrix(pbmcC[, 1:1000])
pbmcC <- as.matrix(pbmcC)

## combine with old dataset
cds <- list(pbmcA, pbmcC)
genes.int <- Reduce(intersect, lapply(cds, rownames))
cds.filtered <- lapply(cds, function(x) as.matrix(x[genes.int,]))
cds.all <- cleanCounts(do.call(cbind, cds.filtered),
                       min.detected=10, verbose=FALSE)
batch <- factor(unlist(lapply(colnames(cds.all), function(x) {
  strsplit(x, '_')[[1]][4] })))
names(batch) <- colnames(cds.all) ## get sample annotations
mat.all <- normalizeCounts(cds.all, verbose=FALSE)
```

```
## Standard analysis with combined data shows batch effects
matnorm.all.info <- normalizeVariance(mat.all, details=TRUE)
```

```
## [1] "Calculating variance fit ..."
## [1] "Using gam with k=5..."
## [1] "3193 overdispersed genes ... "
```

```
matnorm.all <- log10(matnorm.all.info$mat+1)
pcs.all <- getPcs(matnorm.all[matnorm.all.info$ods,],
```

```
                  nGenes=length(matnorm.all.info$ods),
                  nPcs=30)
```

```
## [1] "Identifying top 30 PCs on 3193 most variable genes ..."
```

```
emb.all <- Rtsne::Rtsne(pcs.all,
                        is_distance=FALSE,
                        perplexity=30,
                        num_threads=parallel::detectCores(),
                        verbose=TRUE)$Y
```

```
## Read the 12388 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 12388
##  - point 10000 of 12388
## Done in 3.81 seconds (sparsity = 0.010862)!
## Learning embedding...
## Iteration 50: error is 98.714079 (50 iterations in 5.90 seconds)
## Iteration 100: error is 89.726453 (50 iterations in 5.97 seconds)
## Iteration 150: error is 84.937318 (50 iterations in 5.10 seconds)
## Iteration 200: error is 84.275499 (50 iterations in 5.00 seconds)
## Iteration 250: error is 83.953748 (50 iterations in 5.13 seconds)
## Iteration 300: error is 3.398162 (50 iterations in 5.01 seconds)
## Iteration 350: error is 3.103727 (50 iterations in 5.13 seconds)
## Iteration 400: error is 2.933542 (50 iterations in 4.85 seconds)
## Iteration 450: error is 2.821362 (50 iterations in 4.88 seconds)
## Iteration 500: error is 2.741088 (50 iterations in 4.85 seconds)
## Iteration 550: error is 2.680463 (50 iterations in 4.80 seconds)
## Iteration 600: error is 2.632896 (50 iterations in 4.85 seconds)
## Iteration 650: error is 2.594513 (50 iterations in 4.90 seconds)
## Iteration 700: error is 2.563290 (50 iterations in 4.89 seconds)
## Iteration 750: error is 2.537414 (50 iterations in 4.96 seconds)
## Iteration 800: error is 2.516102 (50 iterations in 4.92 seconds)
## Iteration 850: error is 2.499303 (50 iterations in 4.93 seconds)
## Iteration 900: error is 2.486108 (50 iterations in 4.97 seconds)
## Iteration 950: error is 2.475667 (50 iterations in 4.97 seconds)
## Iteration 1000: error is 2.467494 (50 iterations in 5.02 seconds)
## Fitting performed in 101.03 seconds.
```

```
rownames(emb.all) <- rownames(pcs.all)
```

```
## Regular batch correction
pcs.all.bc <- t(sva::ComBat(t(pcs.all), batch))
```

```
## 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
```

```
emb.all.bc <- Rtsne::Rtsne(pcs.all.bc,
                           is_distance=FALSE,
```

```
                          perplexity=30,
                          num_threads=parallel::detectCores(),
                          verbose=TRUE)$Y
```

```
## Read the 12388 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 12388
##  - point 10000 of 12388
## Done in 5.72 seconds (sparsity = 0.010990)!
## Learning embedding...
## Iteration 50: error is 98.603771 (50 iterations in 6.84 seconds)
## Iteration 100: error is 90.120164 (50 iterations in 7.86 seconds)
## Iteration 150: error is 86.408855 (50 iterations in 6.14 seconds)
## Iteration 200: error is 85.731332 (50 iterations in 6.07 seconds)
## Iteration 250: error is 85.556583 (50 iterations in 5.99 seconds)
## Iteration 300: error is 3.519356 (50 iterations in 5.62 seconds)
## Iteration 350: error is 3.254636 (50 iterations in 5.25 seconds)
## Iteration 400: error is 3.094237 (50 iterations in 5.37 seconds)
## Iteration 450: error is 2.985761 (50 iterations in 5.27 seconds)
## Iteration 500: error is 2.908874 (50 iterations in 5.31 seconds)
## Iteration 550: error is 2.850914 (50 iterations in 5.30 seconds)
## Iteration 600: error is 2.805019 (50 iterations in 5.44 seconds)
## Iteration 650: error is 2.767830 (50 iterations in 5.49 seconds)
## Iteration 700: error is 2.737341 (50 iterations in 5.47 seconds)
## Iteration 750: error is 2.712397 (50 iterations in 5.44 seconds)
## Iteration 800: error is 2.691978 (50 iterations in 5.46 seconds)
## Iteration 850: error is 2.677492 (50 iterations in 5.54 seconds)
## Iteration 900: error is 2.667429 (50 iterations in 5.54 seconds)
## Iteration 950: error is 2.659926 (50 iterations in 5.56 seconds)
## Iteration 1000: error is 2.654557 (50 iterations in 5.56 seconds)
## Fitting performed in 114.52 seconds.
```

```
rownames(emb.all.bc) <- rownames(pcs.all.bc)
```

```
## MUDAN-based approach
## apply pbmcA's model and gene scaling factors
preds.all <- predictLds(mat.all, model, gsf, verbose=FALSE)
lds.all <- preds.all$x
com.final <- factor(preds.all$class); names(com.final) <- rownames(preds.all$x)
## batch correct within clusters
lds.bc <- clusterBasedBatchCorrect(lds.all, batch, com.final)
```

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

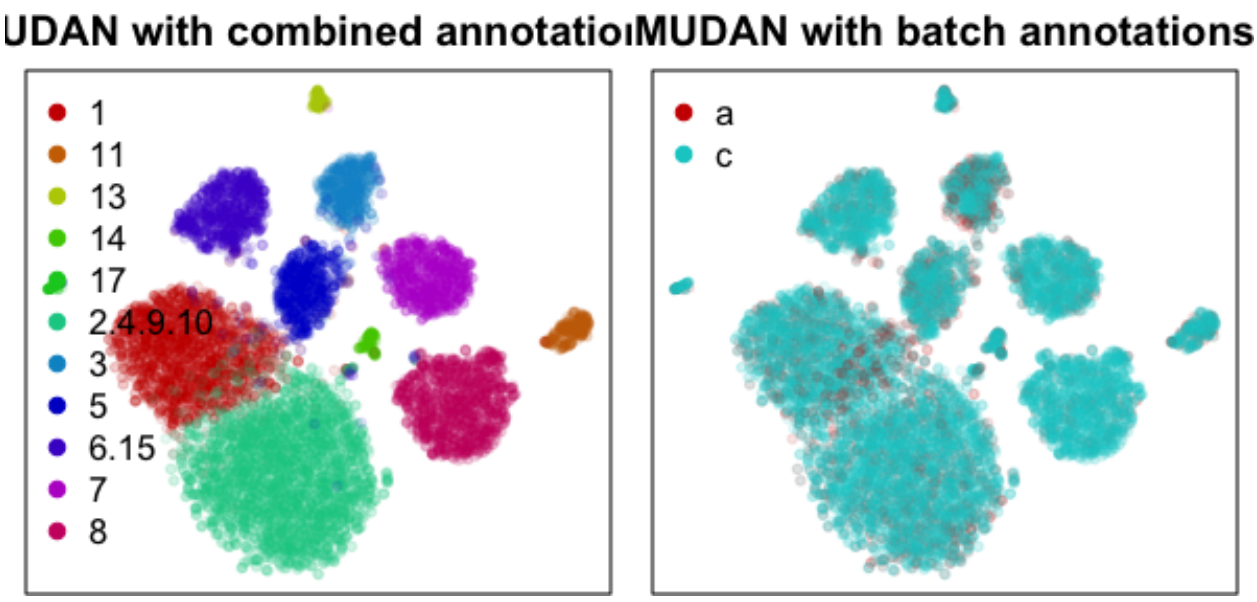
```
## Finding parametric adjustments
## Adjusting the Data
```

```
## get common embedding
emb.lds.bc <- Rtsne::Rtsne(lds.bc,
                           is_distance=FALSE,
                           perplexity=30,
                           num_threads=parallel::detectCores(),
                           verbose=FALSE)$Y
rownames(emb.lds.bc) <- rownames(lds.bc)

## plot
par(mfrow=c(1,2), mar=c(0.5,0.5,2,0.5))
plotEmbedding(emb.lds.bc, com.final,
              main="MUDAN with combined annotations",
              alpha=0.1, show.legend=TRUE, legend.x = "topleft")
```

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

```
plotEmbedding(emb.lds.bc, batch,
              main="MUDAN with batch annotations",
              alpha=0.1, show.legend=TRUE, legend.x = "topleft")
```

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

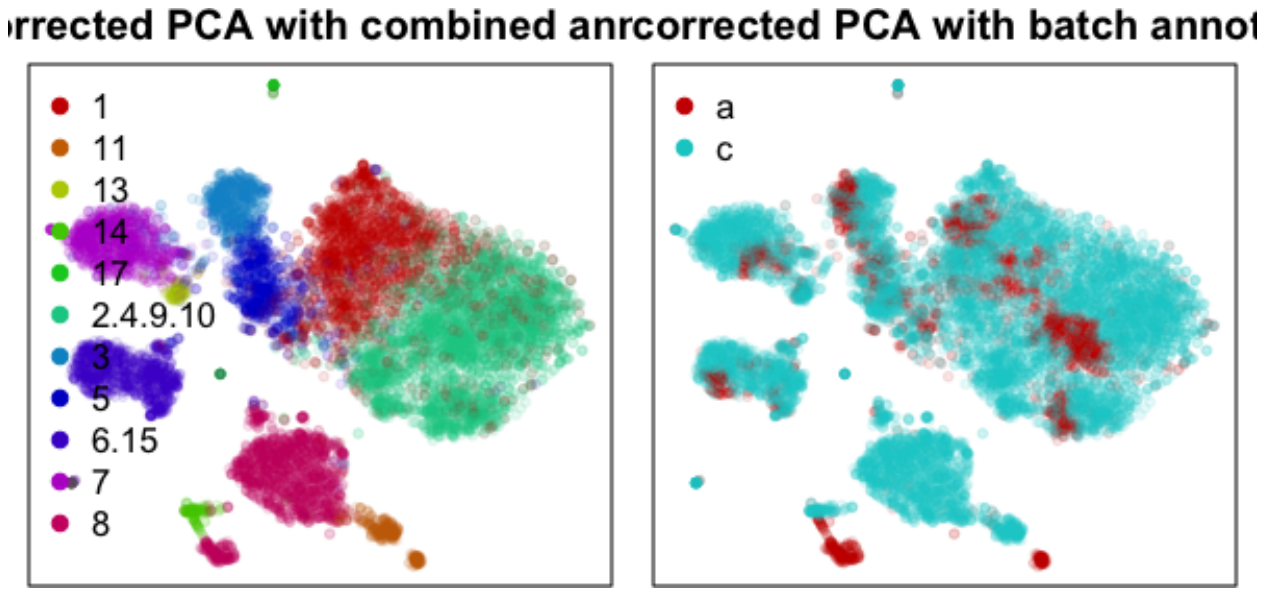



```
## regular batch corrected
plotEmbedding(emb.all.bc, com.final,
              main="Batch-corrected PCA with combined annotations",
              alpha=0.1, show.legend=TRUE, legend.x = "topleft")
```

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

```
plotEmbedding(emb.all.bc, batch,
              main="Batch-corrected PCA with batch annotations",
              alpha=0.1, show.legend=TRUE, legend.x = "topleft")
```

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

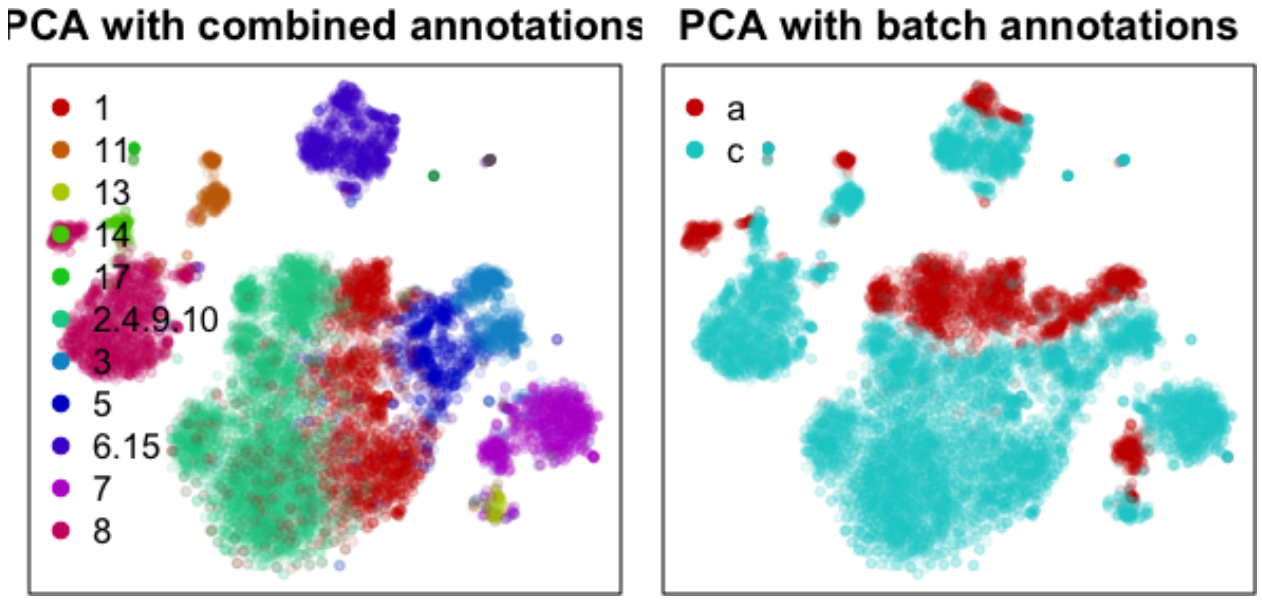



```
## no batch correction
plotEmbedding(emb.all, com.final,
              main="PCA with combined annotations",
              alpha=0.1, show.legend=TRUE, legend.x = "topleft")
```

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

```
plotEmbedding(emb.all, batch,
              main="PCA with batch annotations",
              alpha=0.1, show.legend=TRUE, legend.x = "topleft")
```

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




```
## plot expression of marker genes
par(mfrow=c(2,3), mar=c(0.5,0.5,2,0.5))
invisible(lapply(marker.genes, function(g) {
  gcol <- cds.all[g,] ## plot a gene
```

```
  plotEmbedding(emb.lds.bc, color=gcol,
                main=g, xlab=NA, ylab=NA,
                alpha=0.1,
                verbose=FALSE)
}))
```