Highlighting the features of MUDAN

Jean Fan

2018-02-21

```
library(MUDAN)
```

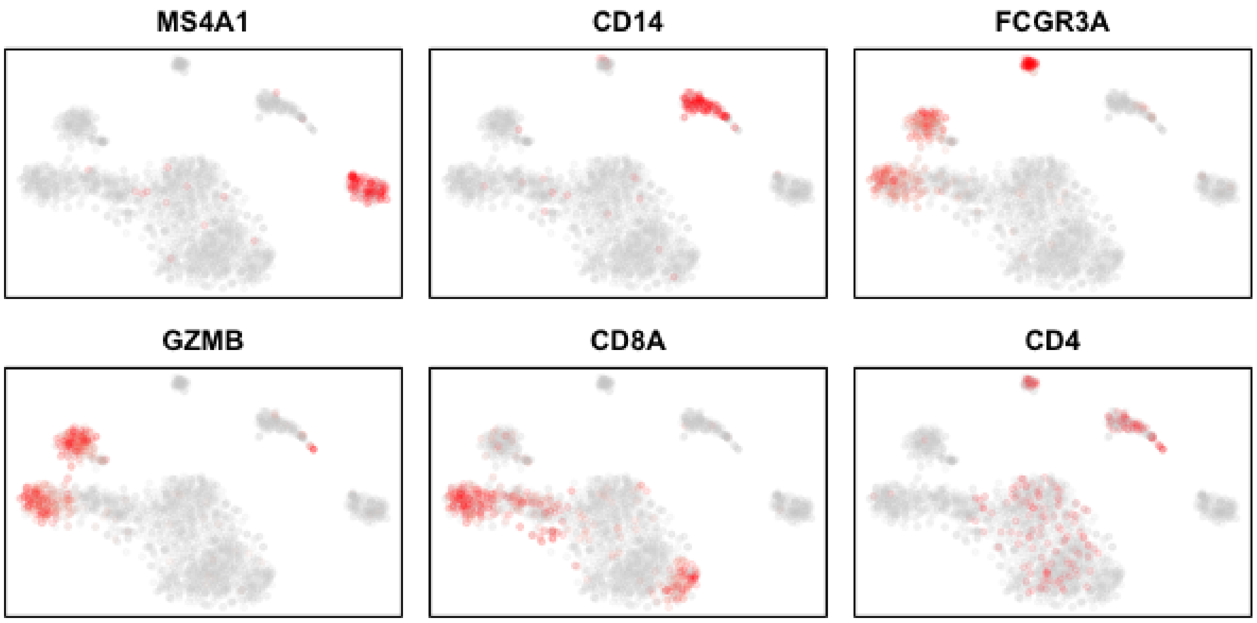
MUDAN features

(1) Fast, flexible analysis

```
data(pbmcA) ## load built in 10X pbmcA dataset
## downsample for testing purposes only
pbmcA <- as.matrix(pbmcA[, 1:2000])

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 2000 cells and 9632 genes took 26.1660299301147 seconds"
```

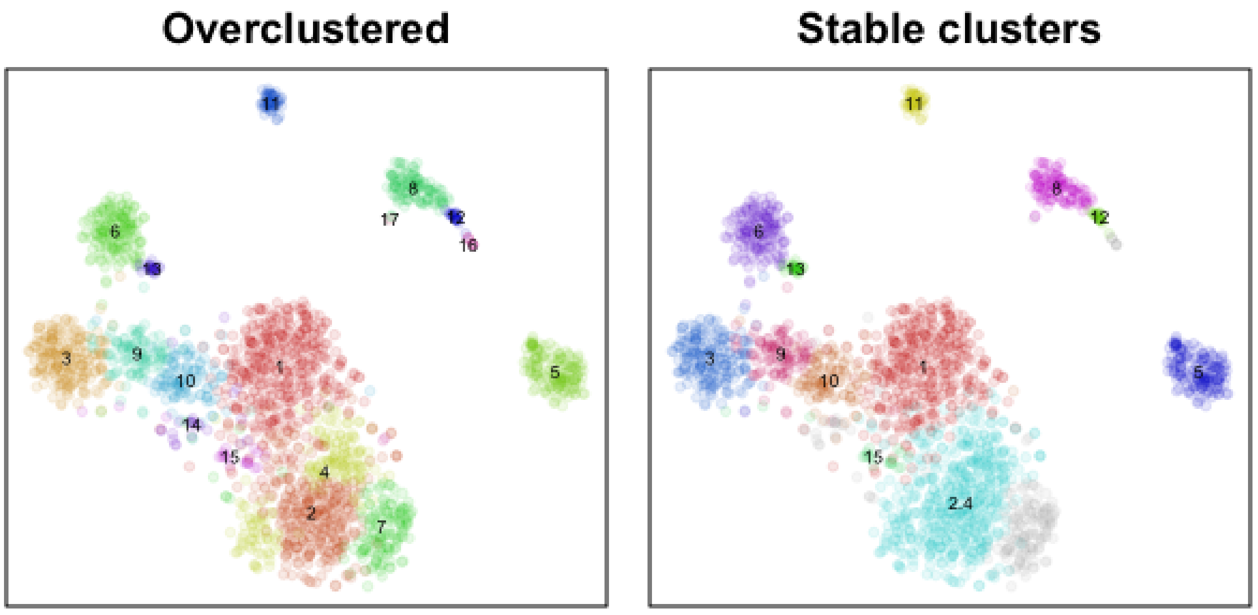
```
## plot expression of marker genes
marker.genes <- c('MS4A1', 'CD14', 'FCGR3A', 'GZMB', 'CD8A', 'CD4')
par(mfrow=c(2,3), mar=c(0.5,0.5,2,0.5))
invisible(lapply(marker.genes, function(g) {
  gcol <- cd[g,] ## plot a gene
  plotEmbedding(emb,
                color=gcol,
                mark.clusters=TRUE,
                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=5,
                            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
```

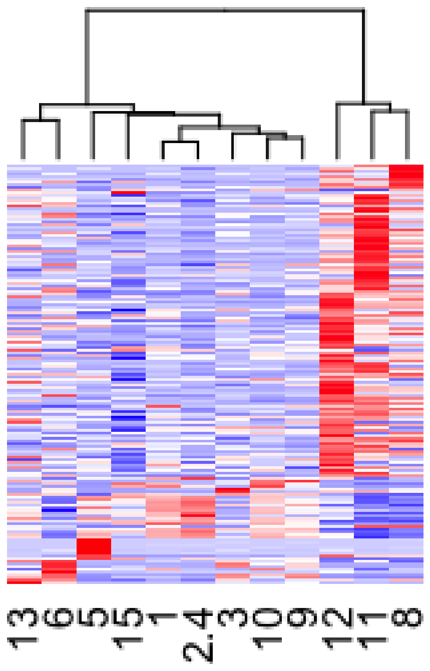
```
plotEmbedding(emb, groups=stable$com,
              main='Stable clusters', xlab=NA, ylab=NA,
              mark.clusters=TRUE, alpha=0.1, mark.cluster.cex=0.5,
              verbose=FALSE)
```



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

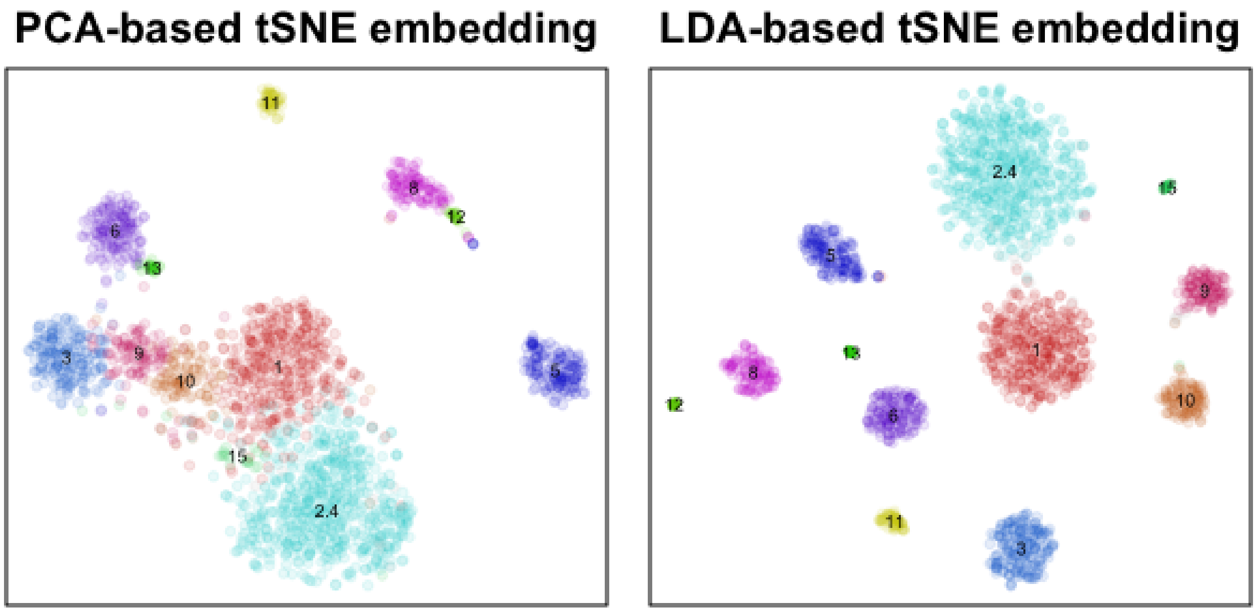
```
## [1] "Running differential expression with 12 clusters ... "
## [1] "Summarizing results ... "
```

```
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[, 1:2000])

## 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] "2331 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 2331 most variable genes ..."
```

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

```
## Read the 4000 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 4000
## Done in 1.00 seconds (sparsity = 0.033238)!
## Learning embedding...
## Iteration 50: error is 84.828501 (50 iterations in 2.79 seconds)
## Iteration 100: error is 74.017368 (50 iterations in 2.59 seconds)
## Iteration 150: error is 73.029449 (50 iterations in 2.51 seconds)
## Iteration 200: error is 72.749898 (50 iterations in 2.26 seconds)
## Iteration 250: error is 72.618879 (50 iterations in 1.97 seconds)
## Iteration 300: error is 2.315962 (50 iterations in 1.72 seconds)
## Iteration 350: error is 2.074260 (50 iterations in 1.59 seconds)
## Iteration 400: error is 1.959037 (50 iterations in 1.62 seconds)
## Iteration 450: error is 1.890128 (50 iterations in 1.68 seconds)
## Iteration 500: error is 1.846732 (50 iterations in 1.74 seconds)
## Iteration 550: error is 1.821407 (50 iterations in 1.73 seconds)
## Iteration 600: error is 1.807193 (50 iterations in 1.76 seconds)
## Iteration 650: error is 1.799401 (50 iterations in 1.96 seconds)
## Iteration 700: error is 1.792757 (50 iterations in 1.99 seconds)
## Iteration 750: error is 1.787139 (50 iterations in 1.97 seconds)
## Iteration 800: error is 1.782517 (50 iterations in 1.91 seconds)
## Iteration 850: error is 1.777445 (50 iterations in 1.93 seconds)
## Iteration 900: error is 1.772900 (50 iterations in 1.86 seconds)
## Iteration 950: error is 1.768764 (50 iterations in 1.82 seconds)
## Iteration 1000: error is 1.765191 (50 iterations in 1.84 seconds)
## Fitting performed in 39.24 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 4000 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 4000
## Done in 3.10 seconds (sparsity = 0.033578)!
## Learning embedding...
## Iteration 50: error is 84.597406 (50 iterations in 5.27 seconds)
## Iteration 100: error is 75.313812 (50 iterations in 4.78 seconds)
## Iteration 150: error is 74.699065 (50 iterations in 4.12 seconds)
## Iteration 200: error is 74.478029 (50 iterations in 3.94 seconds)
## Iteration 250: error is 74.376454 (50 iterations in 3.46 seconds)
## Iteration 300: error is 2.560859 (50 iterations in 3.16 seconds)
## Iteration 350: error is 2.287897 (50 iterations in 3.21 seconds)
## Iteration 400: error is 2.165065 (50 iterations in 3.70 seconds)
## Iteration 450: error is 2.098761 (50 iterations in 4.26 seconds)
## Iteration 500: error is 2.056923 (50 iterations in 4.33 seconds)
## Iteration 550: error is 2.028765 (50 iterations in 3.89 seconds)
## Iteration 600: error is 2.010886 (50 iterations in 2.92 seconds)
## Iteration 650: error is 2.001480 (50 iterations in 2.15 seconds)
## Iteration 700: error is 1.997738 (50 iterations in 1.86 seconds)
## Iteration 750: error is 1.994211 (50 iterations in 1.84 seconds)
## Iteration 800: error is 1.989984 (50 iterations in 1.83 seconds)
## Iteration 850: error is 1.985299 (50 iterations in 1.91 seconds)
## Iteration 900: error is 1.980493 (50 iterations in 2.08 seconds)
## Iteration 950: error is 1.975449 (50 iterations in 2.01 seconds)
## Iteration 1000: error is 1.974561 (50 iterations in 1.94 seconds)
## Fitting performed in 62.66 seconds.
```

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

```r
## 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)
```

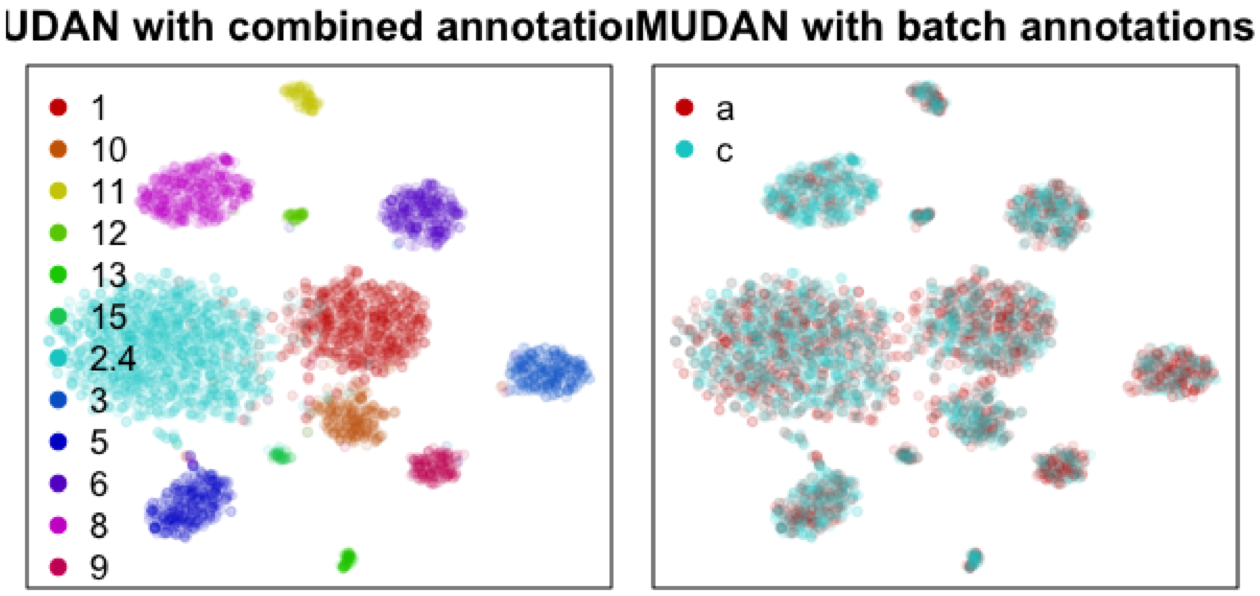
```
## Found 2 batches
## Adjusting for 0 covariate(s) or covariate level(s)
## Standardizing Data across genes
## Fitting L/S model and finding priors
## Finding parametric adjustments
## Adjusting the Data
## Found 2 batches
## Adjusting for 0 covariate(s) or covariate level(s)
## Standardizing Data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

```r
## 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
```

```r
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
```

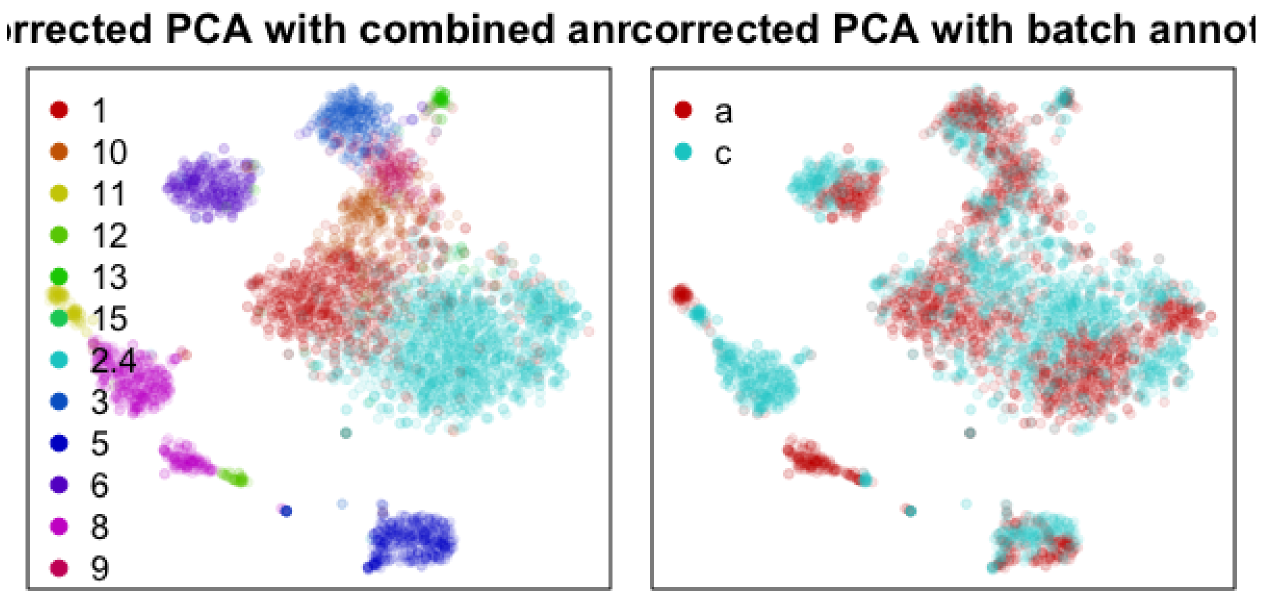


```r
## regular batch corrected
plotEmbedding(emb.all.bc, com.final,
              main="Batch-corrected PCA with combined annotations",
              alpha=0.1, show.legend=TRUE, legend.x = "topleft")
```

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

```r
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
```

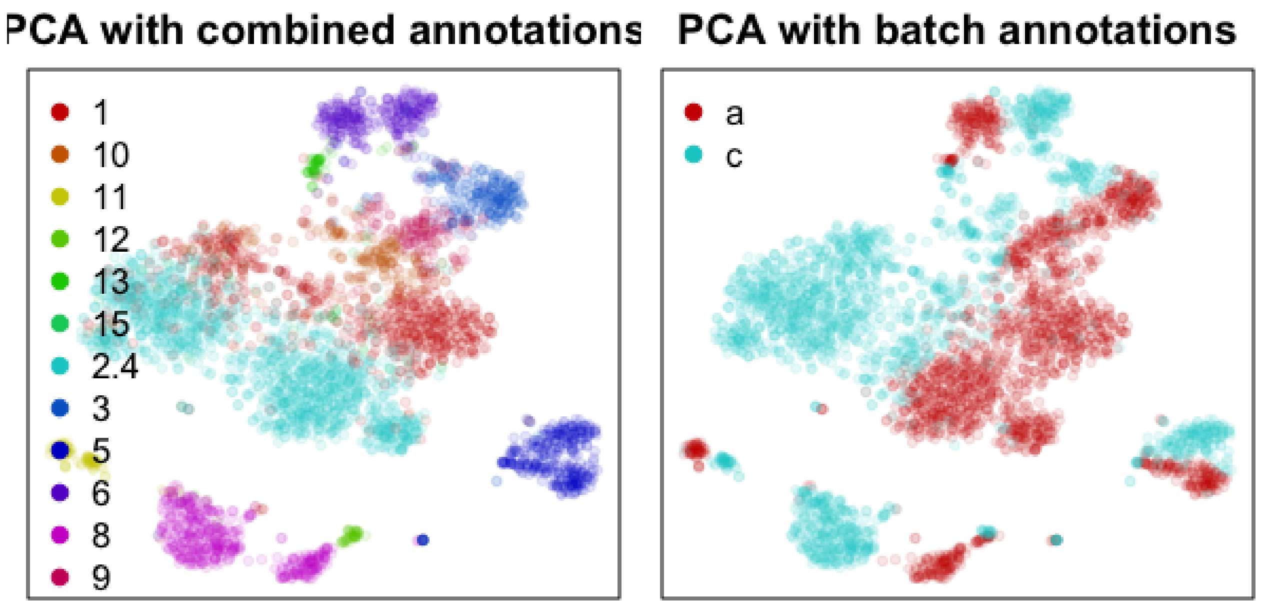


```
## no batch correction
plotEmbedding(emb.all, com.final,
              main="PCA with combined annotations",
              alpha=0.1, show.legend=TRUE, legend.x = "topleft")
```

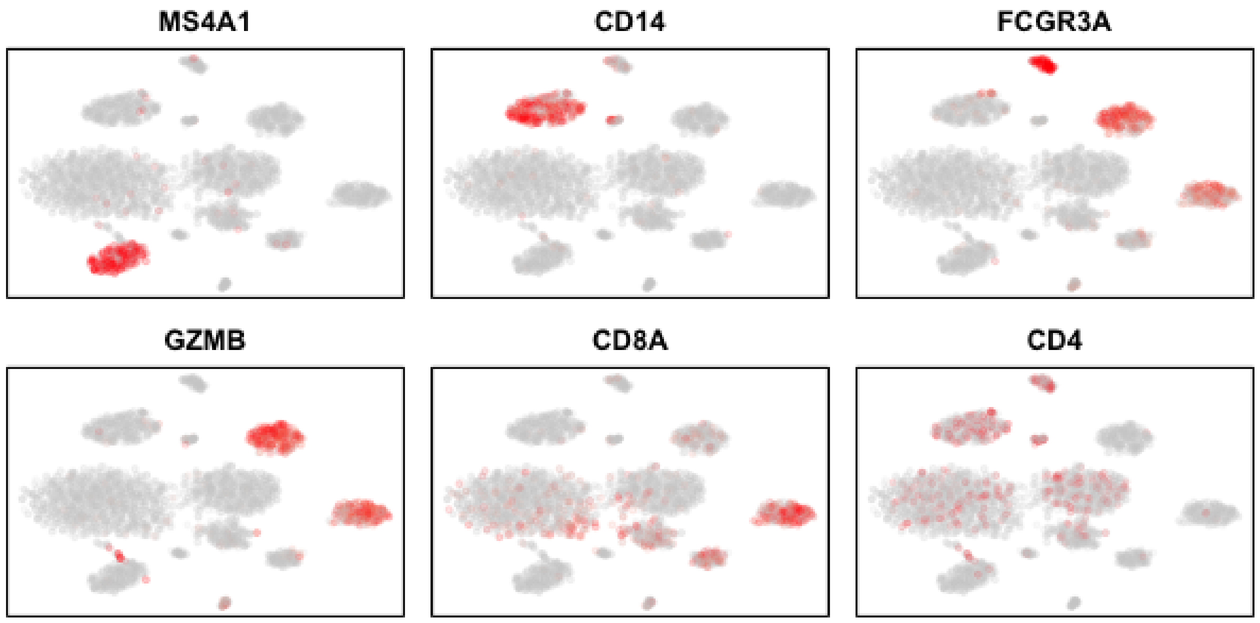
```
## using provided groups as a factor
```

```
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)
}))
```


MS4A1
CD14
FCGR3A
GZMB
CD8A
CD4