Highlighting the features of MUDAN

Jean Fan

2018-02-22

```
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 28.401407957077 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)
}))
```

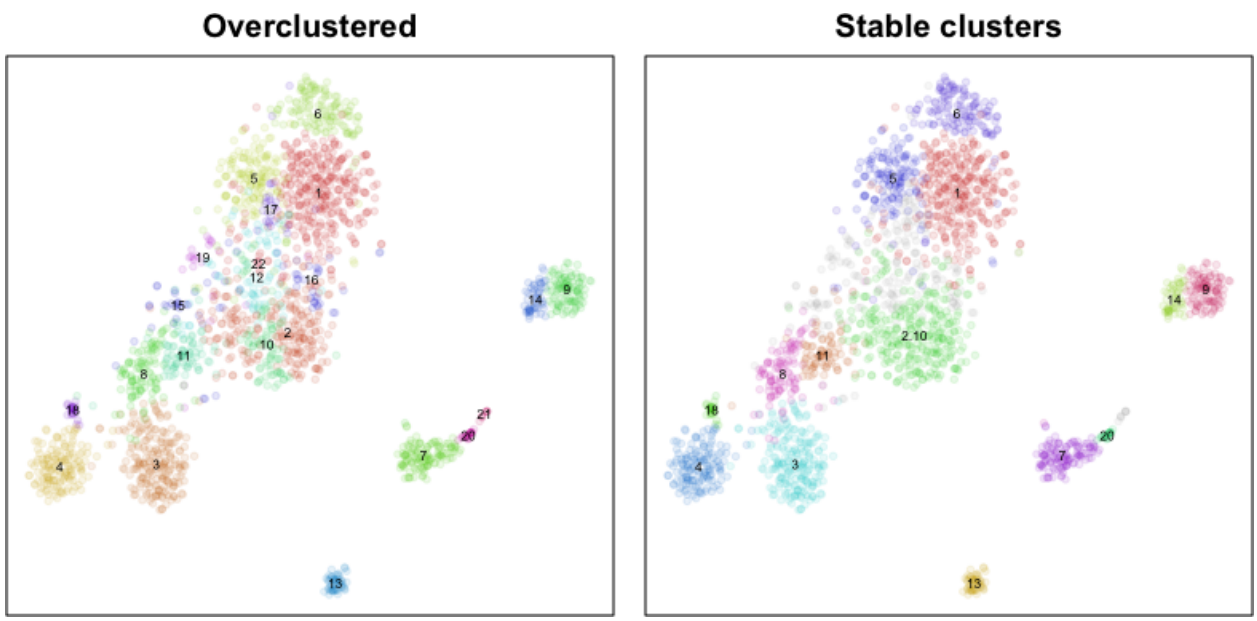
(2) Graph-based subpopulation detection and subpopulation stability analysis

```
## graph-based community detection; over cluster with small k
com <- getComMembership(pcs,
                        k=10, 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
```

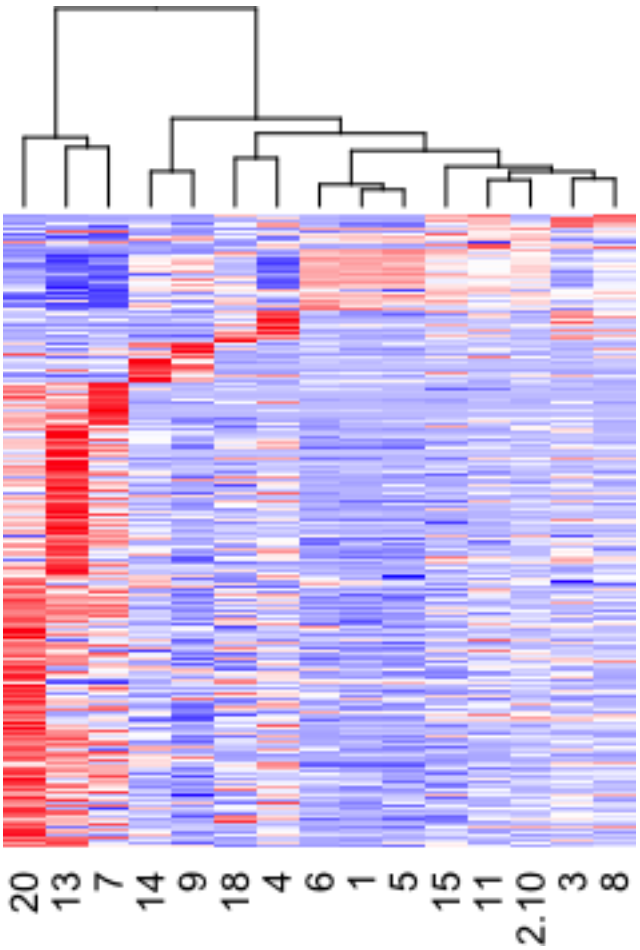
```
plotEmbedding(emb, groups=stable$com,
              main='Stable clusters', xlab=NA, ylab=NA,
              mark.clusters=TRUE, alpha=0.1, mark.cluster.cex=0.5,
              verbose=FALSE)
```




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

```
## [1] "Running differential expression with 14 clusters ... "
## [1] "Summarizing results ... "
```

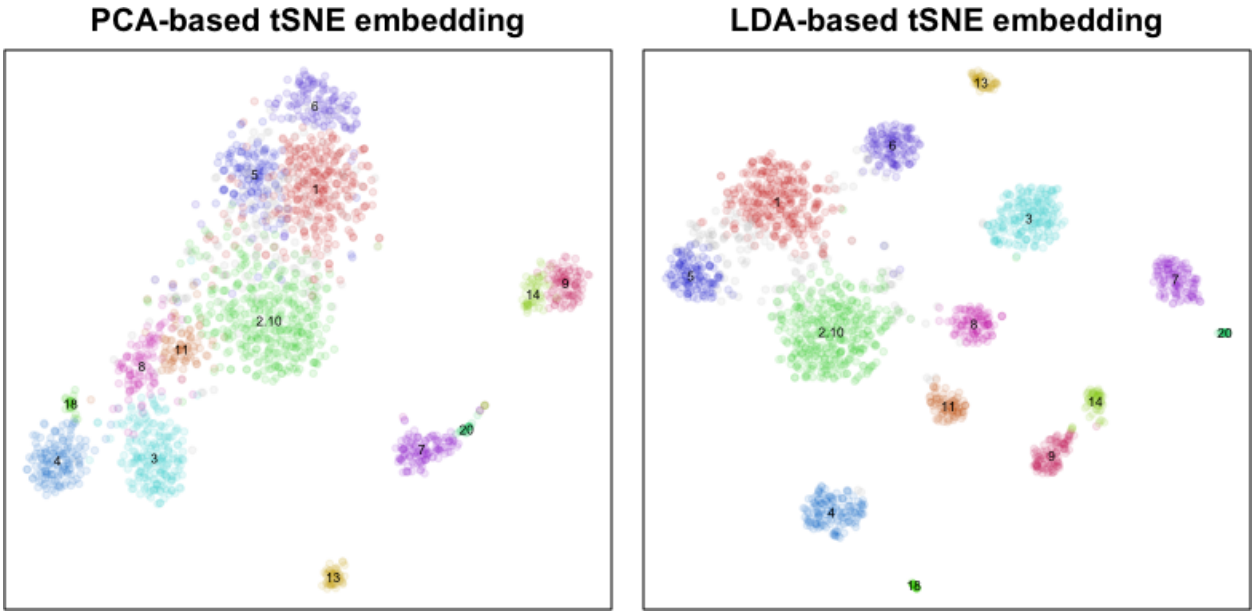
```
dg <- dg[stable$hc$labels[stable$hc$order]]
dg.sig <- unlist(lapply(dg, function(x) {
  x <- x[x$Z>1.96,] ## significant z-score
  x <- x[x$highest,] ## must be highest in this group (optimal markers)
  rownames(x)
}))
dg.sig <- intersect(dg.sig, rownames(stable$mat.summary))
m <- stable$mat.summary[dg.sig,]
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=FALSE)$Y
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=FALSE)$Y
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
## 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
```

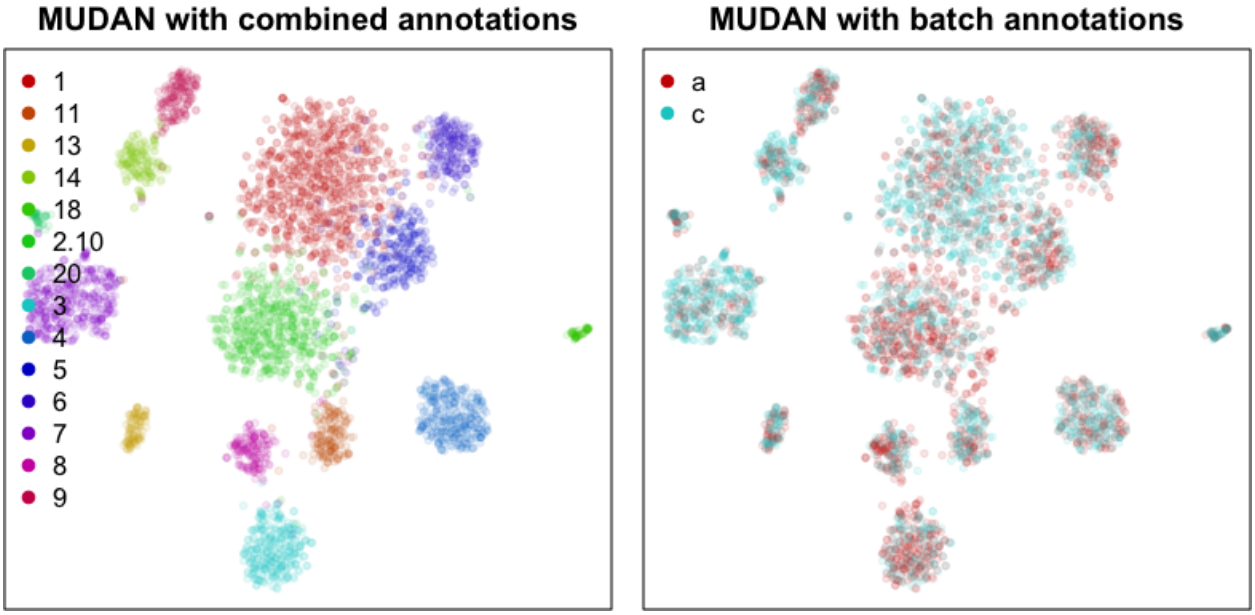
```
## get common embedding
emb.lds.bc <- Rtsne::Rtsne(lds.bc,
                           is_distance=FALSE,
                           perplexity=30,
                           num_threads=parallel::detectCores(),
                           verbose=FALSE)$Y
rownames(emb.lds.bc) <- rownames(lds.bc)

## 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
```




```
## 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)
}))
```

When no batch correction, shows obvious batch differences.

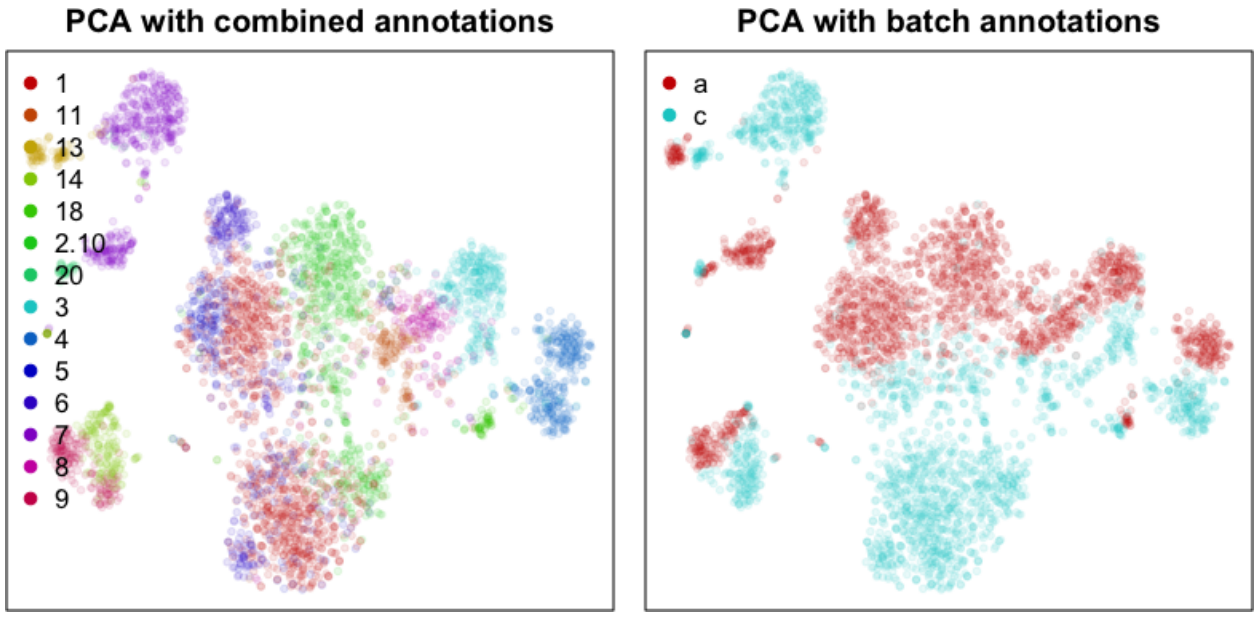
```
par(mfrow=c(1,2), mar=c(0.5,0.5,2,0.5))
## no batch correction
plotEmbedding(emb.all, com.final,
```

```
              main="PCA with combined annotations",
              alpha=0.1, show.legend=TRUE, legend.x = "topleft")
```

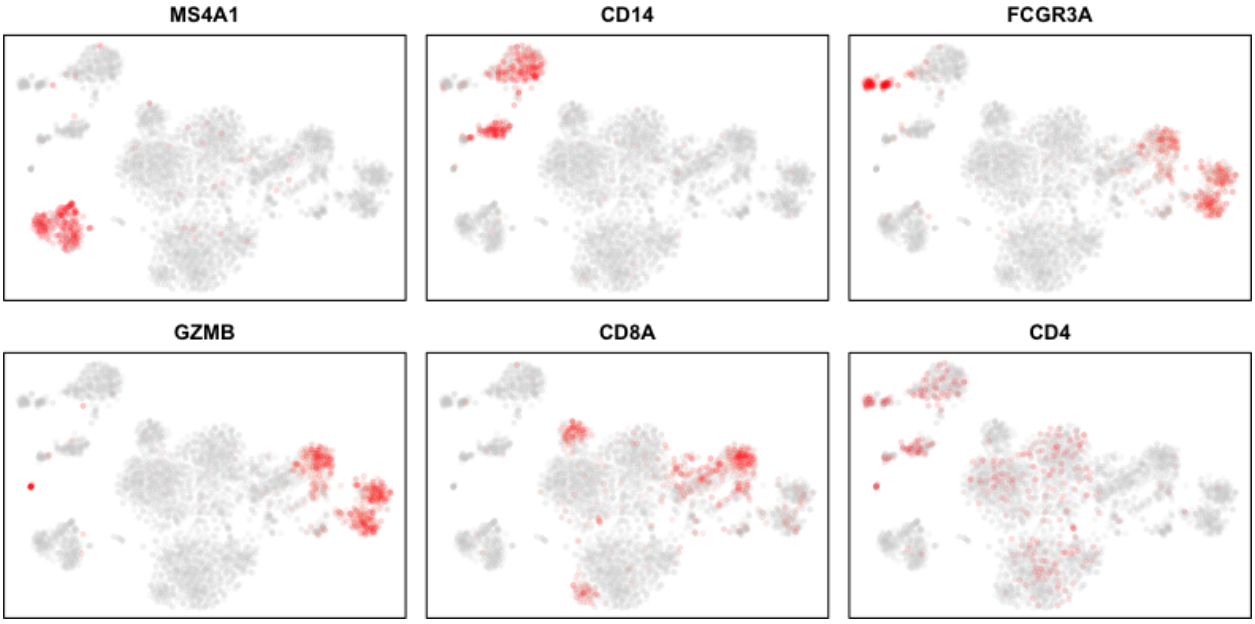
```
## using provided groups as a factor
```

```
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.all, color=gcol,
                main=g, xlab=NA, ylab=NA,
                alpha=0.1,
                verbose=FALSE)
}))
```



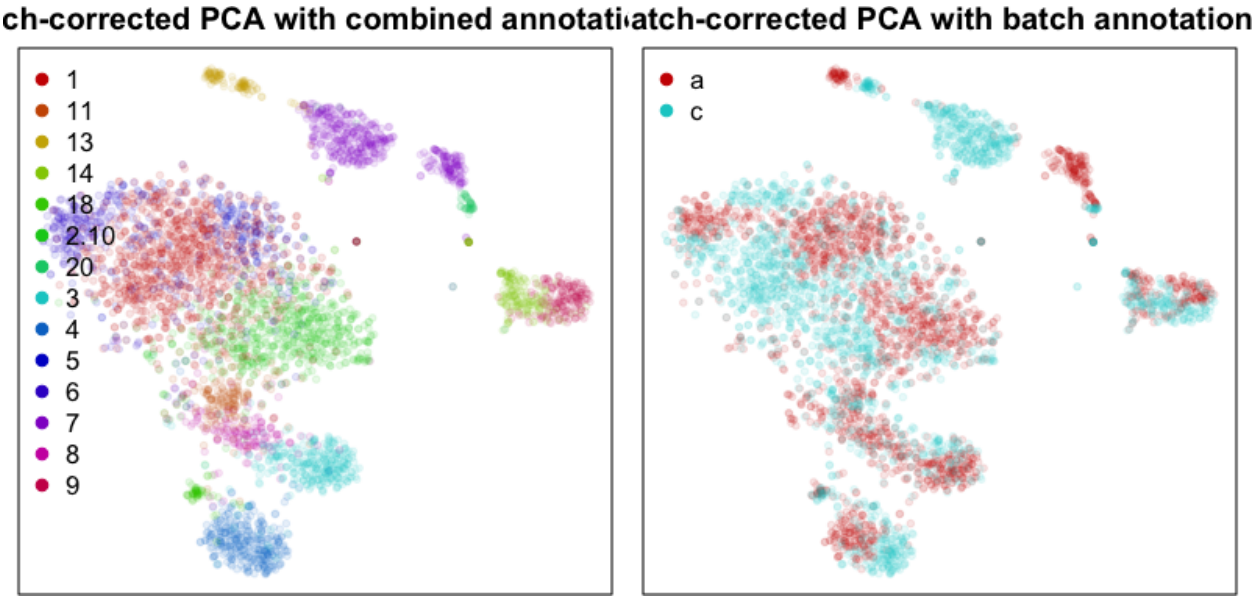
Regular batch correction is still insufficient, especially for smaller subpopulations.

```
par(mfrow=c(1,2), mar=c(0.5,0.5,2,0.5))
## 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
```



```
## 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.all.bc, color=gcol,
                main=g, xlab=NA, ylab=NA,
                alpha=0.1,
                verbose=FALSE)
}))
```

(5) Reference annotation

```
## load sorted data with known cell-type annotations from sorting
data(referenceCounts)
data(referenceAnnot)

## standard analysis to see how sorted annotations compare with unbiased analysis
mat.reference <- normalizeCounts(referenceCounts)
```

```
## [1] "Normalizing matrix with 2140 cells and 8863 genes"
```

```
matnorm.info.reference <- normalizeVariance(mat.reference,
                                    details=TRUE,
                                    verbose=FALSE)
matnorm.reference <- log10(matnorm.info.reference$mat+1)
pcs.reference <- getPcs(matnorm.reference[matnorm.info.reference$ods,],
               nGenes=length(matnorm.info.reference$ods),
               nPcs=30,
               verbose=FALSE)
emb.reference <- Rtsne::Rtsne(pcs.reference,
                     is_distance=FALSE,
                     perplexity=30,
                     num_threads=parallel::detectCores(),
                     verbose=FALSE)$Y
rownames(emb.reference) <- rownames(pcs.reference)
```

```
## get marker genes
dg.reference <- getDifferentialGenes(referenceCounts, referenceAnnot)
```

```
## [1] "Running differential expression with 9 clusters ... "
## [1] "Summarizing results ... "
```

```
dg.reference.sig <- lapply(dg.reference, function(x) {
  x <- na.omit(x)
  x <- x[x$Z>3,]
  x <- x[x$highest,]
  rownames(x)
})

## train LDA using known annotations from sorting
model.reference <- modelLda(matnorm.reference[unlist(dg.reference.sig),], referenceAnnot, retest=TRUE)
```

```
## [1] "calculating LDA ..."
## [1] "LDA prediction accuracy ..."
##
## TRUE
## 2140
```

```
## apply classifier to new data
gsf <- matnorm.info.reference$df$gsf
names(gsf) <- rownames(matnorm.info.reference$df)
reference.pred <- predictLds(mat.all, model.reference, gsf)
```

```
## [1] "Percentage of features retained: 0.986799052239648"
```

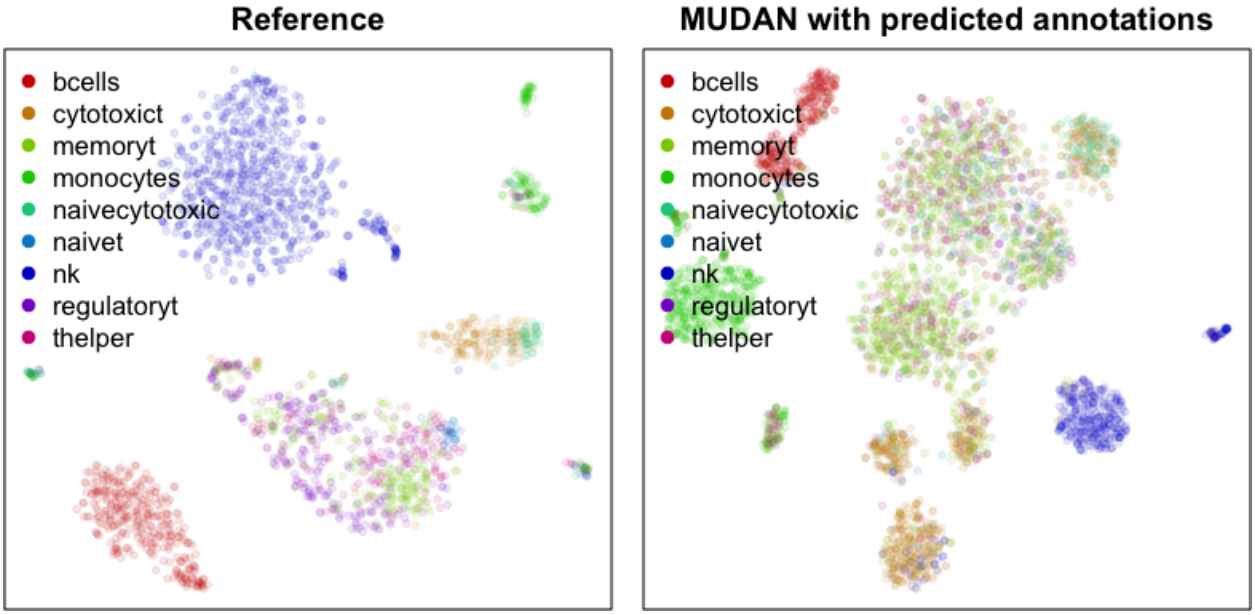
```
reference.com <- getConfidentPreds(reference.pred$posterior)

## plot
par(mfrow=c(1,2), mar=c(0.5,0.5,2,0.5))
plotEmbedding(emb.reference, referenceAnnot,
              main="Reference",
              alpha=0.1, xlab=NA, ylab=NA,
              show.legend=TRUE, legend.x = "topleft")
```

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

```
plotEmbedding(emb.lds.bc, reference.com,
              main="MUDAN with predicted annotations",
              alpha=0.1, xlab=NA, ylab=NA,
              show.legend=TRUE, legend.x = "topleft")
```

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


Reference
bcells
cytotoxict
memoryt
monocytes
naivecytotoxic
naivet
nk
regulatoryt
thelper
MUDAN with predicted annotations
bcells
cytotoxict
memoryt
monocytes
naivecytotoxic
naivet
nk
regulatoryt
thelper