

First, load the following packages. Note, some of the packages are not available on CRAN or BioConductor. They can, however, be installed by using the [drat](#) package.

```
suppressPackageStartupMessages({
  require(bcTSNE)
  require(data.table)
  require(batchelor)
  require(kBET)
  require(splatter)
  require(scater)
  require(Rtsne)
  require(lisi)
  require(harmony)
  require(dlfUtils)
})
## Uncomment to install kBET & lisi
## Note: the packages will require compilation
# if (!require(drat)) {install.packages("drat"); library(drat)}
# drat::addRepo("daynefiler")
# install.packages(c("lisi", "kBET", "harmony", "dlfUtils"))
```

Create simulated single-cell RNA sequencing data using the [splatter](#) package.

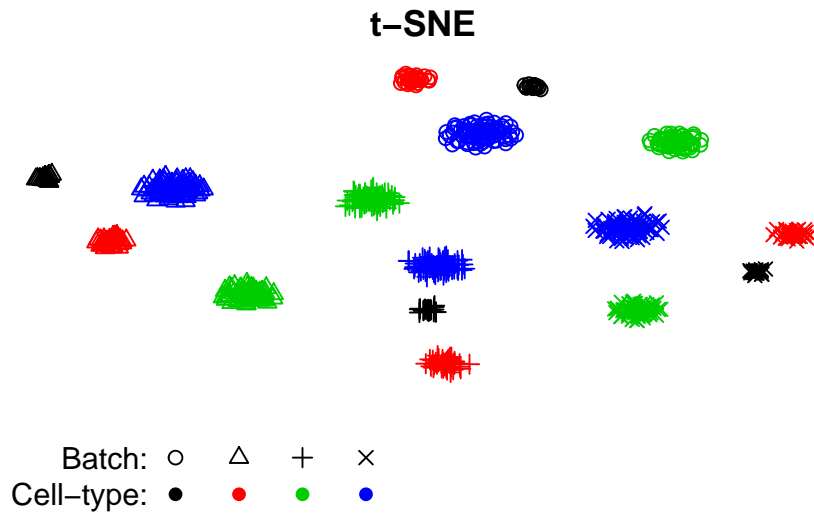
```
p <- newSplatParams(seed = 1234,
  batchCells = rep(200, 4),
  batch.facLoc = 0.2,
  batch.facScale = 0.1,
  group.prob = c(0.1, 0.2, 0.3, 0.4),
  de.facLoc = 0.1,
  de.facScale = 0.4)
sim <- splatSimulate(p, method = "groups", verbose = FALSE)
sizeFactors(sim) <- librarySizeFactors(sim)
sim <- normalize(sim)
sim <- normalize(sim, return_log = FALSE)
assay(sim, "centered") <- t(scale(t(normcounts(sim)),
  center = TRUE,
  scale = FALSE))
Z <- model.matrix(~ -1 + factor(colData(sim)$Batch))
grp <- factor(sim$Group)
bch <- as.integer(factor(sim$Batch))
```

Setup a placeholder for the results:

```
res <- vector(mode = "list", length = 6)
names(res) <- c("btcc", "btlc", "hmlc", "hmcc", "mnn", "tsne")
```

Start by running the regular t-SNE algorithm:

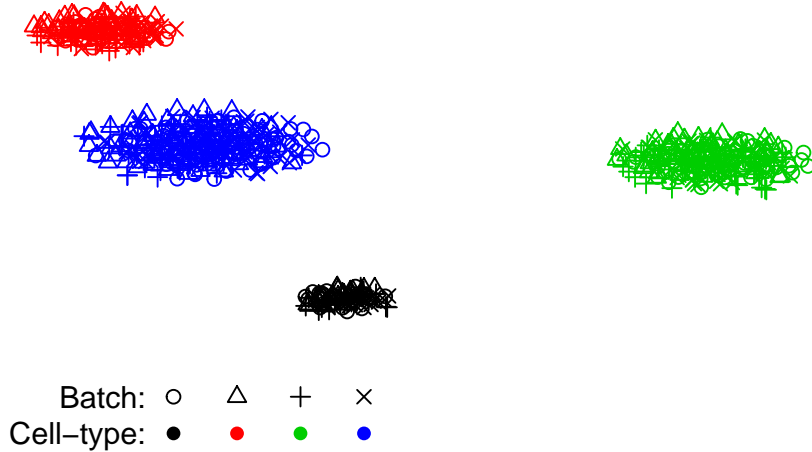
```
set.seed(1234)
res$tsne <- Rtsne(t(assay(sim, "centered")), initial_dims = 50)$Y
pltSimRes(res$tsne, "t-SNE")
```



Run the BC-t-SNE algorithm on centered:

```
set.seed(1234)
res$btcc <- bctsne(t(assay(sim, "centered")), Z, k = 50)$Y
pltSimRes(res$btcc, "bcTSNE-centered")
```

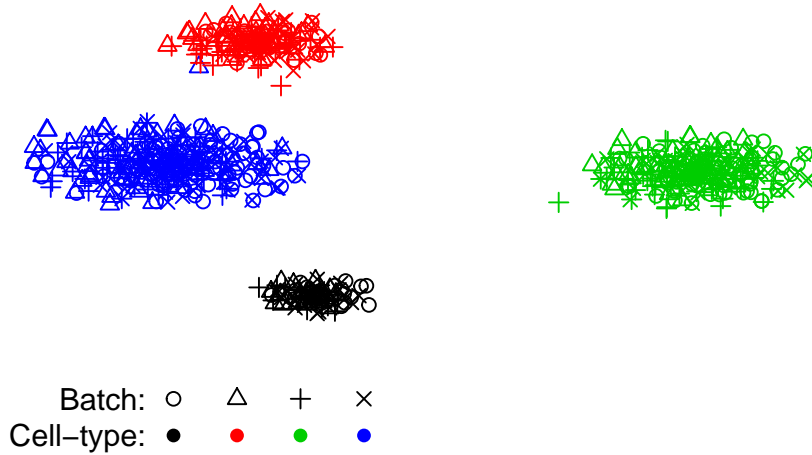
## bcTSNE-centered



Run the BC-t-SNE algorithm on logcounts:

```
set.seed(1234)
res$btlc <- bcttsne(t(logcounts(sim)), Z, k = 50)$Y
pltSimRes(res$btlc, "bcTSNE-logcounts")
```

## bcTSNE-logcounts

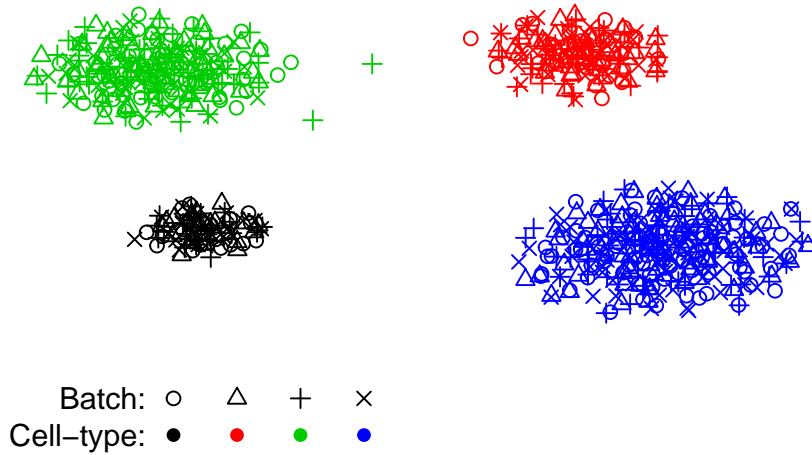


Run the harmony algorithm on the default logcounts:

```
set.seed(1234)
sim <- runPCA(sim, 50, exprs_values = "logcounts")
sim <- RunHarmony(sim, group.by.vars = "Batch")
```

```
res$hmlc = Rtsne(reducedDim(sim, "HARMONY"), pca = FALSE)$Y
pltSimRes(res$hmlc, "Harmony-logcounts")
```

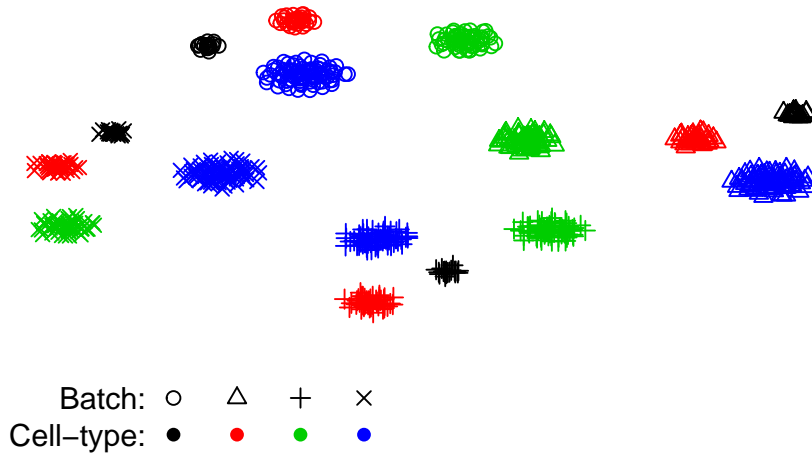
## Harmony-logcounts



Run the harmony algorithm on the same set as bcTSNE, centered:

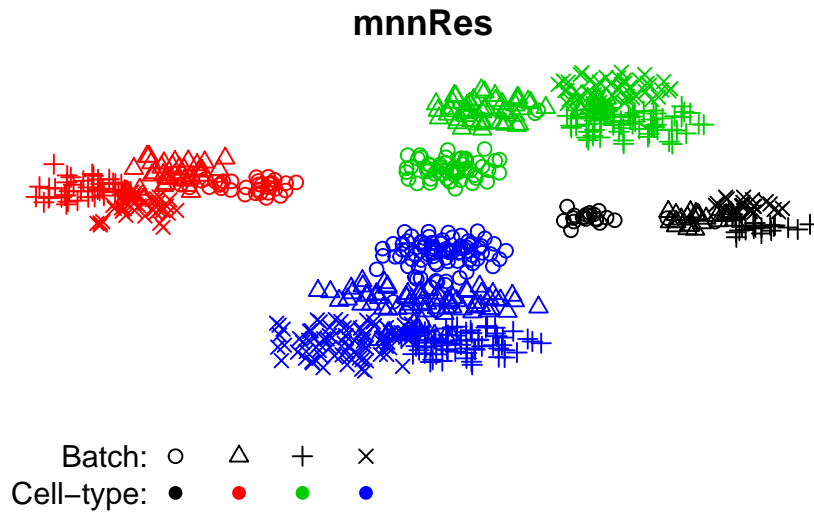
```
set.seed(1234)
sim <- runPCA(sim, 50, exprs_values = "centered")
sim <- RunHarmony(sim, group.by.vars = "Batch")
res$hmc = Rtsne(reducedDim(sim, "HARMONY"), pca = FALSE)$Y
pltSimRes(res$hmc, "Harmony-centered")
```

## Harmony-centered



Run the MNN algorithm:

```
set.seed(1234)
tmp <- mnnCorrect(sim, batch = factor(sim$Batch))
res$mnn <- Rtsne(t(assay(tmp, "corrected")), initial_dims = 50)$Y
rm(tmp)
pltSimRes(res$mnn, "mnnRes")
```



Compare performances:

```
batchList <- list(batch = factor(sim$Batch),
                  cellType = factor(sim$Group))
met <- lapply(res, calcMetrics, bchLst = batchList)

## Warning in max(pcCorr[setsignif]): no non-missing arguments to
## max; returning -Inf
## Warning in max(pcCorr[setsignif]): no non-missing arguments to
## max; returning -Inf
## Warning in max(pcCorr[setsignif]): no non-missing arguments to
## max; returning -Inf
## Warning in max(pcCorr[setsignif]): no non-missing arguments to
## max; returning -Inf

calcMetrics <- function(Y, bchLst) {
  calcSil <- function(x) {
    s <- batch_sil(pca.data = list(x = Y), batch = x, nPCs = 2)
    1 - abs(s)
  }
}
```

		SIL	kBET	iLSIS	PcR
Batch	bcTSNE-centered	0.9794	0.9968	0.7323	0.0000
	bcTSNE-logcounts	0.9815	0.9914	0.7998	0.0000
	Harmony-centered	0.3942	0.8376	0.0000	1.0000
	Harmony-logcounts	0.9666	0.9995	0.8811	0.0000
	MNN	0.9908	0.9945	0.1333	0.0000
	<i>t</i> -SNE	0.5113	0.9627	0.0000	1.0000
Cell type	bcTSNE-centered	0.1686	0.3420	0.0000	1.0000
	bcTSNE-logcounts	0.2020	0.3375	0.0004	1.0000
	Harmony-centered	0.9470	0.4648	0.0061	0.4600
	Harmony-logcounts	0.2369	0.3432	0.0002	1.0000
	MNN	0.3796	0.6227	0.0007	1.0000
	<i>t</i> -SNE	0.9655	0.0531	0.0058	0.4800

```

}
calcKBET <- function(x) {
  kBET(Y, batch = x, do.pca = FALSE, plot = FALSE)$average
}
calcPCA <- function(x) {
  pcRegression(pca.data = prcomp(Y), batch = x, n_top = 2)$pcReg
}
sil <- sapply(bchLst, calcSil)
kbet <- sapply(bchLst, calcKBET)
lisi <- compute_lisi(Y,
  meta_data = as.data.frame(bchLst),
  label_colnames = names(bchLst))
lisi <- colMeans(lisi)
sizes <- sapply(bchLst, function(x) length(unique(x)))
lisi <- (lisi - 1)/(sizes - 1)
pca <- sapply(bchLst, calcPCA)
res <- list(sil = sil, kbet = kbet, lisi = lisi, pca = pca)
do.call(cbind, res)
}

```