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.

```
pkgLoaded <- suppressPackageStartupMessages({
  c(require(bcTSNE),
    require(data.table),
    require(batchelor),
    require(kBET),
    require(splatter),
    require(scater),
    require(Rtsne),
    require(lisi),
    require(harmony),
    require(dlfUtils),
    require(xtable))
})
pkgLoaded <- all(pkgLoaded)
## 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.

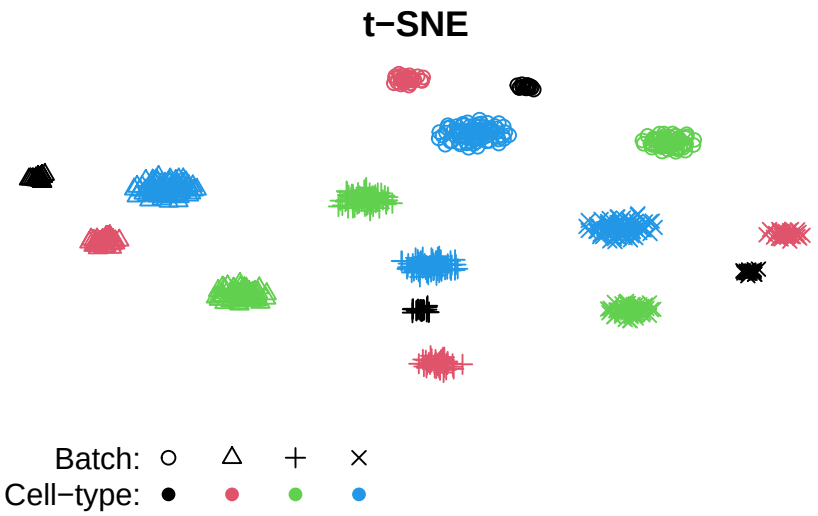
```
if (pkgLoaded) {
  p <- newSplatParams(seed = 1234,
                      batchCells = rep(200, 4),
                      batch.facLoc = 0.2,
                      batch.facScale = 0.1,
                      group.prob = c(0.1, 0.2, 0.3, 0.4),
                      de.facLoc = 0.1,
                      de.facScale = 0.4)
  sim <- splatSimulate(p, method = "groups", verbose = FALSE)
  sizeFactors(sim) <- librarySizeFactors(sim)
  sim <- logNormCounts(sim)
  sim <- logNormCounts(sim, 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")
```

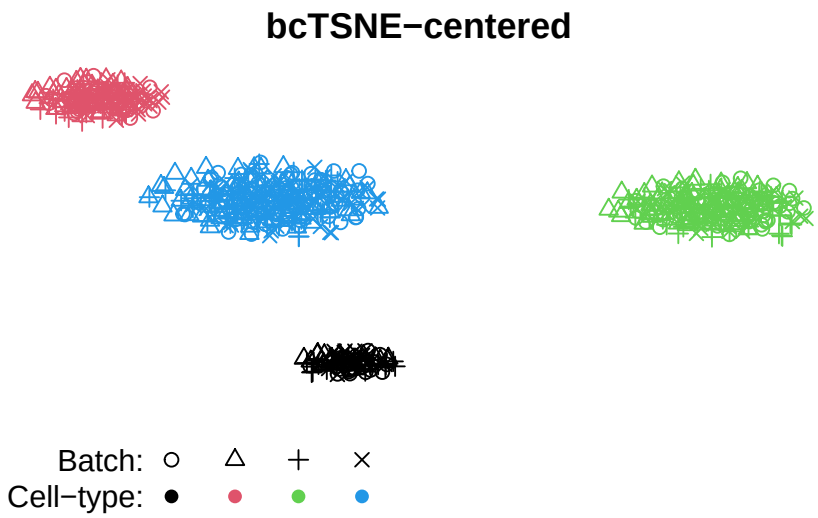
Start by running the regular t-SNE algorithm:

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



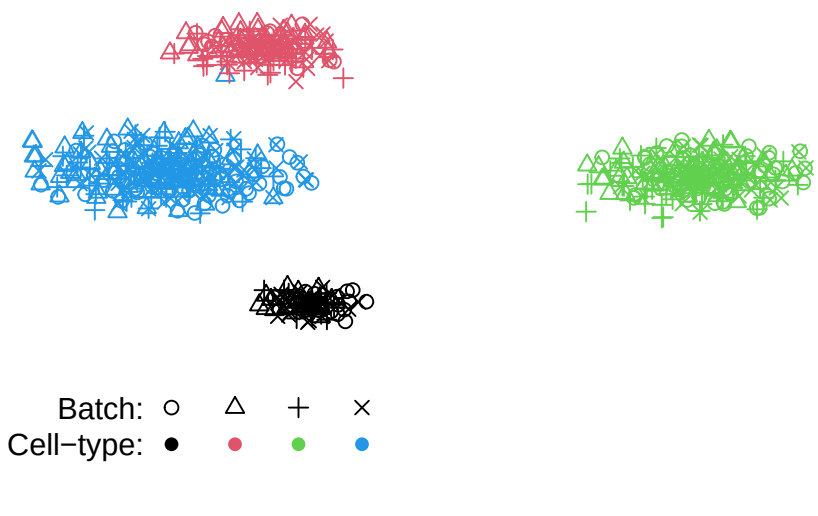
Run the BC-t-SNE algorithm on `centered`:

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



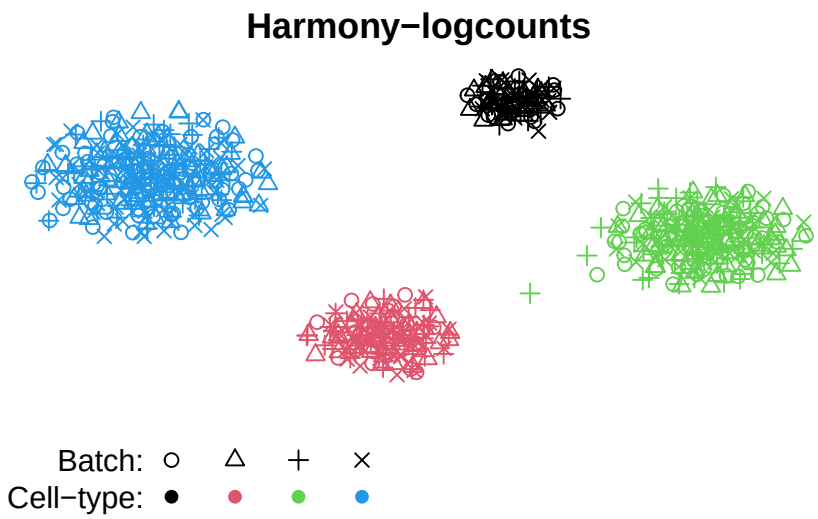
Run the BC-t-SNE algorithm on `logcounts`:

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



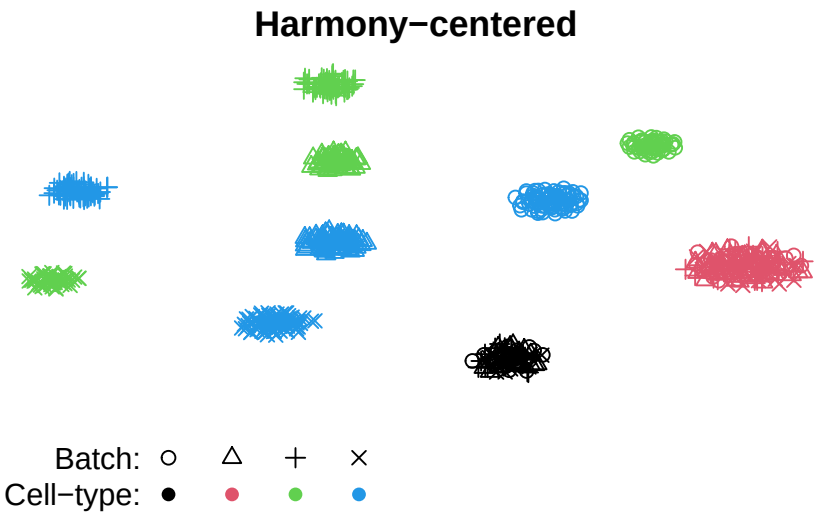
Run the harmony algorithm on the default `logcounts`:

```
if (pkgLoaded) {
  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")
}
```



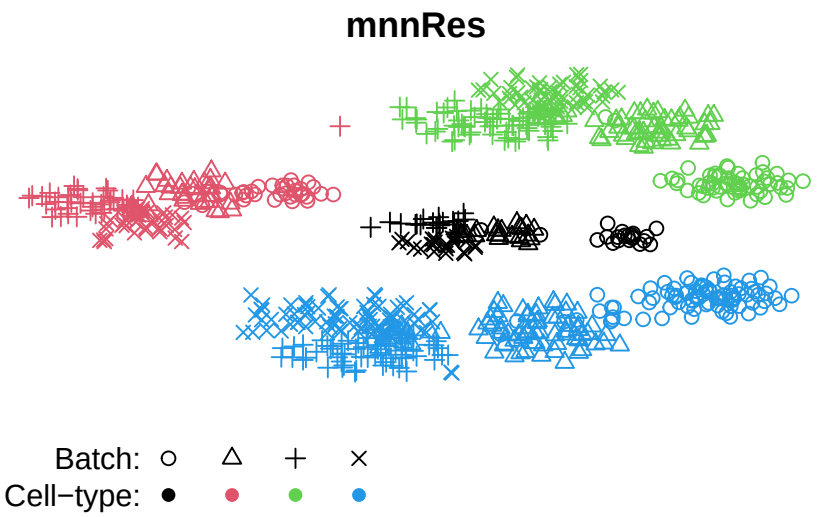
Run the harmony algorithm on the same set as bcTSNE, `centered`:

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



Run the MNN algorithm:

```
if (pkgLoaded) {
  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:

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

```
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in max(pcCorr[setsignif]):  no non-missing arguments to
max; returning -Inf
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
```

```
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in max(pcCorr[setsignif]):  no non-missing arguments to
max; returning -Inf
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in max(pcCorr[setsignif]):  no non-missing arguments to
max; returning -Inf
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
```

```
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
## Warning in if (class(knn) == "list") {:  the condition has length
> 1 and only the first element will be used
```

```
if (pkgLoaded) {
  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	0.9806	0.9977	0.7602	0.0000
	bcTSNE-logcounts	0.9811	0.9946	0.7871	0.0000
	Harmony-centered	0.8522	0.6696	0.2575	1.0000
	Harmony-logcounts	0.9682	0.9964	0.8765	0.0000
	MNN	0.9568	0.9634	0.1383	0.2820
	t-SNE	0.5113	0.9627	0.0000	1.0000
Cell type	bcTSNE-centered	0.1884	0.3327	0.0000	1.0000
	bcTSNE-logcounts	0.2056	0.3357	0.0004	1.0000
	Harmony-centered	0.6624	0.6098	0.0000	1.0000
	Harmony-logcounts	0.2245	0.3500	0.0004	1.0000
	MNN	0.4786	0.9068	0.0006	1.0000
	t-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)
}
}
```