```r
## ----style, echo=FALSE, results='hide', message=FALSE--------------------
library(BiocStyle)
library(knitr)
opts_chunk$set(error=FALSE, message=FALSE, warning=FALSE)
opts_chunk$set(fig.width=7, fig.height=7)
opts_chunk$set(dpi=300, dev="png", dev.args=list(pointsize=15))
options(bitmapType="cairo", width=100)

# Setting single-core unless explicitly specified otherwise.
library(BiocParallel)
register(SerialParam())

# Deciding whether we want to re-download everything or not.
on.bioc <- TRUE

## ---- message=FALSE, echo=FALSE, results='hide'--------------------------
library(Rtsne)
library(mvoutlier)
library(destiny)

## ---- eval=on.bioc, echo=FALSE, results='hide'---------------------------
all.urls <- c("https://www.ncbi.nlm.nih.gov/geo/download/?
acc=GSE61533&format=file&file=GSE61533%5FHTSEQ%5Fcount%5Fresults%2Exls%2Egz",

              "https://www.ncbi.nlm.nih.gov/geo/download/?
acc=GSE29087&format=file&file=GSE29087%5FL139%5Fexpression%5Ftab%2Etxt%2Egz",

              "https://storage.googleapis.com/linnarsson-lab-www-
blobs/blobs/cortex/expression_mRNA_17-Aug-2014.txt",
              "https://storage.googleapis.com/linnarsson-lab-www-
blobs/blobs/cortex/expression_mito_17-Aug-2014.txt",
              "https://storage.googleapis.com/linnarsson-lab-www-
blobs/blobs/cortex/expression_spikes_17-Aug-2014.txt",
              "http://www.ebi.ac.uk/teichmann-
srv/espresso/static/counttable_es.csv",
              "http://www.nature.com/nbt/journal/v33/n2/extref/nbt.3102-
S7.xlsx")
all.basenames <- basename(all.urls)
all.basenames[1] <- "GSE61533_HTSEQ_count_results.xls.gz"
all.basenames[2] <- "GSE29087_L139_expression_tab.txt.gz"
all.modes <- rep("w", length(all.urls))
all.modes[!grepl("(txt|csv)$", all.basenames)] <- "wb"
for (x in seq_along(all.urls)) {
  download.file(all.urls[x], all.basenames[x], mode=all.modes[x])
}

## ---- results='asis', eval=on.bioc, echo=FALSE---------------------------
cat("***Note:*** *to cite this article, please refer to
http://f1000research.com/articles/5-2122/v1 for instructions.*")

## ----------------------------------------------------------------------
library(R.utils)
gunzip("GSE61533_HTSEQ_count_results.xls.gz", remove=FALSE, overwrite=TRUE)
```

```
library(readxl)
all.counts <- as.data.frame(read_excel('GSE61533_HTSEQ_count_results.xls',
sheet=1))
rownames(all.counts) <- all.counts$ID
all.counts <- all.counts[,-1]
library(scater)
sce <- newSCESet(countData=all.counts)
dim(sce)

## ---------------------------------------------------------------------------
is.spike <- grepl("^ERCC", rownames(sce))
is.mito <- grepl("^mt-", rownames(sce))

## ---------------------------------------------------------------------------
sce <- calculateQCMetrics(sce, feature_controls=list(ERCC=is.spike,
Mt=is.mito))
head(colnames(pData(sce)))

## ---------------------------------------------------------------------------
library(scran)
setSpike(sce) <- "ERCC"

## ----libplothsc, fig.height=6, fig.width=12, fig.cap="**Figure 1:**
Histograms of library sizes (left) and number of expressed genes (right) for
all cells in the HSC dataset."----
par(mfrow=c(1,2))
hist(sce$total_counts/1e6, xlab="Library sizes (millions)", main="",
     breaks=20, col="grey80", ylab="Number of cells")
hist(sce$total_features, xlab="Number of expressed genes", main="",
     breaks=20, col="grey80", ylab="Number of cells")

## ---------------------------------------------------------------------------
libsize.drop <- isOutlier(sce$total_counts, nmads=3, type="lower", log=TRUE)
feature.drop <- isOutlier(sce$total_features, nmads=3, type="lower",
log=TRUE)

## ----controlplothsc, fig.width=12, fig.height=6, fig.cap="**Figure 2:**
Histogram of the proportion of reads mapped to mitochondrial genes (left) or
spike-in transcripts (right) across all cells in the HSC dataset."----
par(mfrow=c(1,2))
hist(sce$pct_counts_feature_controls_Mt, xlab="Mitochondrial proportion
(%)",
     ylab="Number of cells", breaks=20, main="", col="grey80")
hist(sce$pct_counts_feature_controls_ERCC, xlab="ERCC proportion (%)",
     ylab="Number of cells", breaks=20, main="", col="grey80")

## ---------------------------------------------------------------------------
mito.drop <- isOutlier(sce$pct_counts_feature_controls_Mt, nmads=3,
type="higher")
spike.drop <- isOutlier(sce$pct_counts_feature_controls_ERCC, nmads=3,
type="higher")

## ---------------------------------------------------------------------------
```

```
sce <- sce[,!(libsize.drop | feature.drop | mito.drop | spike.drop)]
data.frame(ByLibSize=sum(libsize.drop), ByFeature=sum(feature.drop),
           ByMito=sum(mito.drop), BySpike=sum(spike.drop),
Remaining=ncol(sce))

## ----pcaqualplothsc, fig.cap="**Figure 3:** PCA plot for cells in the HSC
dataset, constructed using quality metrics. The first and second components
are shown on each axis, along with the percentage of total variance
explained by each component. Bars represent the coordinates of the cells on
each axis."----
fontsize <- theme(axis.text=element_text(size=12),
axis.title=element_text(size=16))
plotPCA(sce, pca_data_input="pdata") + fontsize

## ---- echo=FALSE, results="hide", message=FALSE--------------------------
set.seed(100)

## ----phaseplothsc, message=FALSE, fig.cap="**Figure 4:** Cell cycle phase
scores from applying the pair-based classifier on the HSC dataset, where
each point represents a cell."----
mm.pairs <- readRDS(system.file("exdata", "mouse_cycle_markers.rds",
package="scran"))
library(org.Mm.eg.db)
anno <- select(org.Mm.eg.db, keys=rownames(sce), keytype="SYMBOL",
column="ENSEMBL")
ensembl <- anno$ENSEMBL[match(rownames(sce), anno$SYMBOL)]
assignments <- cyclone(sce, mm.pairs, gene.names=ensembl)
plot(assignments$score$G1, assignments$score$G2M, xlab="G1 score",
ylab="G2/M score", pch=16)

## ------------------------------------------------------------------------
sce <- sce[,assignments$phases=="G1"]

## ------------------------------------------------------------------------
ave.counts <- calcAverage(sce)
keep <- ave.counts >= 1
sum(keep)

## ----abhisthsc, fig.cap="**Figure 5:** Histogram of log-average counts for
all genes in the HSC dataset. The filter threshold is represented by the
blue line."----
hist(log10(ave.counts), breaks=100, main="", col="grey80",
     xlab=expression(Log[10]~"average count"))
abline(v=log10(1), col="blue", lwd=2, lty=2)

## ----topgenehsc, fig.height=9, fig.width=6, fig.cap="**Figure 6:**
Percentage of total counts assigned to the top 50 most highly-abundant
features in the HSC dataset. For each feature, each bar represents the
percentage assigned to that feature for a single cell, while the circle
represents the average across all cells. Bars are coloured by the total
number of expressed features in each cell, while circles are coloured
according to whether the feature is labelled as a control feature."----
plotQC(sce, type = "highest-expression", n=50) + fontsize
```

```
## ---------------------------------------------------------------------------
numcells <- nexprs(sce, byrow=TRUE)
alt.keep <- numcells >= 10
sum(alt.keep)

## ----geneplothsc, fig.cap="**Figure 7:** Number of expressing cells
against the log-mean expression for each gene in the HSC dataset. Spike-in
transcripts are highlighted in red."----
smoothScatter(log10(ave.counts), numcells, xlab=expression(Log[10]~"average
count"),
              ylab="Number of expressing cells")
is.ercc <- isSpike(sce, type="ERCC")
points(log10(ave.counts[is.ercc]), numcells[is.ercc], col="red", pch=16,
cex=0.5)

## ---------------------------------------------------------------------------
sce <- sce[keep,]

## ---- warning=FALSE---------------------------------------------------------
sce <- computeSumFactors(sce, sizes=seq(20, 80, 5))
summary(sizeFactors(sce))

## ----normplothsc, fig.cap="**Figure 8:** Size factors from deconvolution,
plotted against library sizes for all cells in the HSC dataset. Axes are
shown on a log-scale."----
plot(sizeFactors(sce), sce$total_counts/1e6, log="xy",
     ylab="Library size (millions)", xlab="Size factor")

## ---------------------------------------------------------------------------
sce <- computeSpikeFactors(sce, type="ERCC", general.use=FALSE)

## ---------------------------------------------------------------------------
sce <- normalize(sce)

## ----explvarplothsc, fig.cap="**Figure 9:** Density plot of the percentage
of variance explained by the (log-transformed) total spike-in counts across
all genes in the HSC dataset. For each gene, the percentage of the variance
of the normalized log-expression values across cells that is explained by
each factor is calculated. Each curve corresponds to one factor and
represents the distribution of percentages across all genes."----
plotExplanatoryVariables(sce, variables=c("counts_feature_controls_ERCC",

"log10_counts_feature_controls_ERCC")) + fontsize

## ---------------------------------------------------------------------------
var.fit <- trendVar(sce, trend="loess", use.spikes=FALSE, span=0.2)
var.out <- decomposeVar(sce, var.fit)

## ----hvgplothsc, fig.cap="**Figure 10:** Variance of normalized log-
expression values for each gene in the HSC dataset, plotted against the mean
log-expression. The blue line represents the mean-dependent trend fitted to
the variances of the endogenous genes. Variance estimates for spike-in
```

```
transcripts are highlighted in red."----
plot(var.out$mean, var.out$total, pch=16, cex=0.6, xlab="Mean log-
expression",
     ylab="Variance of log-expression")
o <- order(var.out$mean)
lines(var.out$mean[o], var.out$tech[o], col="dodgerblue", lwd=2)
cur.spike <- isSpike(sce)
points(var.out$mean[cur.spike], var.out$total[cur.spike], col="red", pch=16)

## ------------------------------------------------------------------------
hvg.out <- var.out[which(var.out$FDR <= 0.05 & var.out$bio >= 0.5),]
hvg.out <- hvg.out[order(hvg.out$bio, decreasing=TRUE),]
nrow(hvg.out)
write.table(file="hsc_hvg.tsv", hvg.out, sep="\t", quote=FALSE,
col.names=NA)
head(hvg.out)

## ----hvgvioplothsc, fig.cap="**Figure 11:** Violin plots of normalized
log-expression values for the top 10 HVGs in the HSC dataset. Each point
represents the log-expression value in a single cell."----
plotExpression(sce, rownames(hvg.out)[1:10]) + fontsize

## ------------------------------------------------------------------------
set.seed(100)
var.cor <- correlatePairs(sce, subset.row=rownames(hvg.out))
write.table(file="hsc_cor.tsv", var.cor, sep="\t", quote=FALSE,
row.names=FALSE)
head(var.cor)

## ------------------------------------------------------------------------
sig.cor <- var.cor$FDR <= 0.05
summary(sig.cor)

## ---- message=FALSE------------------------------------------------------
library(RBGL)
g <- ftM2graphNEL(cbind(var.cor$gene1, var.cor$gene2)[sig.cor,],
                  W=NULL, V=NULL, edgemode="undirected")
cl <- highlyConnSG(g)$clusters
cl <- cl[order(lengths(cl), decreasing=TRUE)]
head(cl)

## ------------------------------------------------------------------------
var.cor <- correlatePairs(sce, subset.row=rownames(hvg.out), per.gene=TRUE)
head(var.cor)
sig.cor <- var.cor$FDR <= 0.05
summary(sig.cor)

## ----heatmaphsc, message=FALSE, fig.width=7, fig.height=7,
fig.cap="**Figure 12:** Heatmap of mean-centred normalized log-expression
values for correlated HVGs in the HSC dataset. Dendrograms are formed by
hierarchical clustering on the Euclidean distances between genes (row) or
cells (column)."----
chosen <- var.cor$gene[sig.cor]
```

```
norm.exprs <- exprs(sce)[chosen,,drop=FALSE]
heat.vals <- norm.exprs - rowMeans(norm.exprs)
library(gplots)
heat.out <- heatmap.2(heat.vals, col=bluered, symbreak=TRUE, trace='none',
cexRow=0.6)

## ----pcaplothsc, fig.cap="**Figure 13:** PCA plot constructed from
normalized log-expression values of correlated HVGs, where each point
represents a cell in the HSC dataset. First and second components are shown,
along with the percentage of variance explained. Bars represent the
coordinates of the cells on each axis. Each cell is coloured according to
its total number of expressed features."----
plotPCA(sce, exprs_values="exprs", colour_by="total_features",
        feature_set=chosen) + fontsize

## ----tsneplothsc, fig.cap="**Figure 14:** _t_-SNE plots constructed from
normalized log-expression values of correlated HVGs, using a range of
perplexity values. In each plot, each point represents a cell in the HSC
dataset. Bars represent the coordinates of the cells on each axis. Each cell
is coloured according to its total number of expressed features.",
fig.width=12, fig.height=6----
set.seed(100)
out5 <- plotTSNE(sce, exprs_values="exprs", perplexity=5,
colour_by="total_features",
                 feature_set=chosen) + fontsize + ggtitle("Perplexity = 5")
out10 <- plotTSNE(sce, exprs_values="exprs", perplexity=10,
colour_by="total_features",
                  feature_set=chosen) + fontsize + ggtitle("Perplexity =
10")
out20 <- plotTSNE(sce, exprs_values="exprs", perplexity=20,
colour_by="total_features",
                  feature_set=chosen) + fontsize + ggtitle("Perplexity =
20")
multiplot(out5, out10, out20, cols=3)

## ------------------------------------------------------------------------
saveRDS(file="hsc_data.rds", sce)

## ---- echo=FALSE, results='hide'-----------------------------------------
rm(sce, all.counts)
gc()

## ------------------------------------------------------------------------
readFormat <- function(infile) {
  # First column is empty.
  metadata <- read.delim(infile, stringsAsFactors=FALSE, header=FALSE,
nrow=10)[,-1]
  rownames(metadata) <- metadata[,1]
  metadata <- metadata[,-1]
  metadata <- as.data.frame(t(metadata))
  # First column after row names is some useless filler.
  counts <- read.delim(infile, stringsAsFactors=FALSE, header=FALSE,
row.names=1, skip=11)[,-1]
```

```
  counts <- as.matrix(counts)
  return(list(metadata=metadata, counts=counts))
}

## ---------------------------------------------------------------------------
endo.data <- readFormat("expression_mRNA_17-Aug-2014.txt")
spike.data <- readFormat("expression_spikes_17-Aug-2014.txt")
mito.data <- readFormat("expression_mito_17-Aug-2014.txt")

## ---------------------------------------------------------------------------
m <- match(endo.data$metadata$cell_id, mito.data$metadata$cell_id)
mito.data$metadata <- mito.data$metadata[m,]
mito.data$counts <- mito.data$counts[,m]

## ---- echo=FALSE---------------------------------------------------------------
stopifnot(identical(endo.data$metadata$cell_id,
spike.data$metadata$cell_id)) # should be the same.
stopifnot(all(endo.data$metadata$cell_id==mito.data$metadata$cell_id)) #
should now be the same.

## ---------------------------------------------------------------------------
raw.names <- sub("_loc[0-9]+$", "", rownames(endo.data$counts))
new.counts <- rowsum(endo.data$counts, group=raw.names, reorder=FALSE)
endo.data$counts <- new.counts

## ---------------------------------------------------------------------------
all.counts <- rbind(endo.data$counts, mito.data$counts, spike.data$counts)
metadata <- AnnotatedDataFrame(endo.data$metadata)
sce <- newSCESet(countData=all.counts, phenoData=metadata)
dim(sce)

## ---------------------------------------------------------------------------
nrows <- c(nrow(endo.data$counts), nrow(mito.data$counts),
nrow(spike.data$counts))
is.spike <- rep(c(FALSE, FALSE, TRUE), nrows)
is.mito <- rep(c(FALSE, TRUE, FALSE), nrows)

## ---- echo=FALSE, results='hide'-------------------------------------------
# Save some memory.
rm(mito.data, endo.data, spike.data, new.counts)
gc()

## ---------------------------------------------------------------------------
sce <- calculateQCMetrics(sce, feature_controls=list(Spike=is.spike,
Mt=is.mito))
setSpike(sce) <- "Spike"

## ----libplotbrain, fig.width=12, fig.height=6, fig.cap="**Figure 15:**
Histograms of library sizes (left) and number of expressed genes (right) for
all cells in the brain dataset."----
par(mfrow=c(1,2))
hist(sce$total_counts/1e3, xlab="Library sizes (thousands)", main="",
     breaks=20, col="grey80", ylab="Number of cells")
```

```
hist(sce$total_features, xlab="Number of expressed genes", main="",
     breaks=20, col="grey80", ylab="Number of cells")

## ----controlplotbrain, fig.width=12, fig.height=6, fig.cap="**Figure 16:**
Histogram of the proportion of UMIs assigned to mitochondrial genes (left)
or spike-in transcripts (right) across all cells in the brain dataset."----
par(mfrow=c(1,2))
hist(sce$pct_counts_feature_controls_Mt, xlab="Mitochondrial proportion
(%)",
     ylab="Number of cells", breaks=20, main="", col="grey80")
hist(sce$pct_counts_feature_controls_Spike, xlab="ERCC proportion (%)",
     ylab="Number of cells", breaks=20, main="", col="grey80")

## ------------------------------------------------------------------------
libsize.drop <- isOutlier(sce$total_counts, nmads=3, type="lower", log=TRUE)
feature.drop <- isOutlier(sce$total_features, nmads=3, type="lower",
log=TRUE)
mito.drop <- isOutlier(sce$pct_counts_feature_controls_Mt, nmads=3,
type="higher")
spike.drop <- isOutlier(sce$pct_counts_feature_controls_Spike, nmads=3,
type="higher")

## ------------------------------------------------------------------------
sce <- sce[,!(libsize.drop | feature.drop | spike.drop | mito.drop)]
data.frame(ByLibSize=sum(libsize.drop), ByFeature=sum(feature.drop),
           ByMito=sum(mito.drop), BySpike=sum(spike.drop),
Remaining=ncol(sce))

## ----echo=FALSE, results='hide'------------------------------------------
gc()

## ---- echo=FALSE, results='hide', message=FALSE--------------------------
mm.pairs <- readRDS(system.file("exdata", "mouse_cycle_markers.rds",
package="scran"))
library(org.Mm.eg.db)

## ----phaseplotbrain, message=FALSE, fig.cap="**Figure 17:** Cell cycle
phase scores from applying the pair-based classifier on the brain dataset,
where each point represents a cell."----
anno <- select(org.Mm.eg.db, keys=rownames(sce), keytype="SYMBOL",
column="ENSEMBL")
ensembl <- anno$ENSEMBL[match(rownames(sce), anno$SYMBOL)]
assignments <- cyclone(sce, mm.pairs, gene.names=ensembl)
plot(assignments$score$G1, assignments$score$G2M, xlab="G1 score",
ylab="G2/M score", pch=16)

## ----echo=FALSE, results='hide'------------------------------------------
gc()

## ------------------------------------------------------------------------
ave.counts <- calcAverage(sce)
keep <- ave.counts >= 0.1
```

```
## ----abhistbrain, fig.cap="**Figure 18:** Histogram of log-average counts
for all genes in the brain dataset. The filter threshold is represented by
the blue line."----
hist(log10(ave.counts), breaks=100, main="", col="grey",
     xlab=expression(Log[10]~"average count"))
abline(v=log10(0.1), col="blue", lwd=2, lty=2)

## ------------------------------------------------------------------------
sce <- sce[keep,]
nrow(sce)

## ----echo=FALSE, results='hide'------------------------------------------
gc()

## ------------------------------------------------------------------------
sce <- sce[!fData(sce)$is_feature_control_Mt,]

## ----echo=FALSE, results='hide'------------------------------------------
gc()

## ------------------------------------------------------------------------
clusters <- quickCluster(sce)
sce <- computeSumFactors(sce, cluster=clusters)
summary(sizeFactors(sce))

## ----echo=FALSE, results='hide'------------------------------------------
gc()

## ----normplotbrain, fig.cap="**Figure 19:** Size factors from
deconvolution, plotted against library sizes for all cells in the brain
dataset. Axes are shown on a log-scale."----
plot(sizeFactors(sce), sce$total_counts/1e3, log="xy",
     ylab="Library size (thousands)", xlab="Size factor")

## ------------------------------------------------------------------------
sce <- computeSpikeFactors(sce, type="Spike", general.use=FALSE)

## ------------------------------------------------------------------------
sce <- normalize(sce)

## ---- echo=FALSE, results='hide', message=FALSE--------------------------
fontsize <- theme(axis.text=element_text(size=12),
axis.title=element_text(size=16))

## ----explvarplotbrain, fig.cap="**Figure 20:** Density plot of the
percentage of variance explained by each factor across all genes in the
brain dataset. For each gene, the percentage of the variance of the
normalized log-expression values that is explained by the (log-transformed)
total spike-in counts, the sex or age of the mouse, or the tissue of origin
is calculated. Each curve corresponds to one factor and represents the
distribution of percentages across all genes."----
plotExplanatoryVariables(sce, variables=c("counts_feature_controls_Spike",
"log10_counts_feature_controls_Spike", "sex", "tissue", "age")) + fontsize

## ---------------------------------------------------------------------------
design <- model.matrix(~sce$sex)

## ---------------------------------------------------------------------------
var.fit.des <- trendVar(sce, trend="loess", span=0.4, design=design)
var.out.des <- decomposeVar(sce, var.fit.des)
hvg.out.des <- var.out.des[which(var.out.des$FDR <= 0.05 & var.out.des$bio
>= 0.5),]
nrow(hvg.out.des)

## ---------------------------------------------------------------------------
collected <- list()
for (block in levels(sce$sex)) {
  cur.sce <- sce[,sce$sex==block]
  cur.sce <- normalize(cur.sce)
  var.fit <- trendVar(cur.sce, trend="loess", span=0.4)
  collected[[block]] <- decomposeVar(cur.sce, var.fit)
}

## ---- echo=FALSE, results='hide', message=FALSE--------------------------
rm(cur.sce)
gc()

## ----hvgplotbrain, fig.width=12, fig.height=6, fig.cap="**Figure 21:**
Variance of normalized log-expression values against the mean for each gene,
calculated across all cells from male (left) or female mice (right). In each
plot, the red line represents the mean-dependent trend in the technical
variance of the spike-in transcripts (also highlighted as red points)."----
par(mfrow=c(1,2))
for (block in c("1", "-1")) {
  var.out <- collected[[block]]
  plot(var.out$mean, var.out$total, pch=16, cex=0.6, xlab="Mean log-
expression",
       ylab="Variance of log-expression", main=ifelse(block=="1", "Male",
"Female"))
  points(var.out$mean[isSpike(sce)], var.out$total[isSpike(sce)], col="red",
pch=16)
  o <- order(var.out$mean)
  lines(var.out$mean[o], var.out$tech[o], col="red", lwd=2)
}

## ---------------------------------------------------------------------------
var.out <- do.call(combineVar, collected)
hvg.out <- var.out[which(var.out$FDR <= 0.05 & var.out$bio >= 0.5),]
hvg.out <- hvg.out[order(hvg.out$bio, decreasing=TRUE),]
nrow(hvg.out)
write.table(file="brain_hvg.tsv", hvg.out, sep="\t", quote=FALSE,
col.names=NA)
head(hvg.out)

## ----hvgvioplotbrain, fig.cap="**Figure 22:** Violin plots of normalized
```

```
log-expression values for the top 10 HVGs in the brain dataset. For each
gene, each point represents the log-expression value for an individual
cell."----
plotExpression(sce, rownames(hvg.out)[1:10], alpha=0.05, jitter="jitter") +
fontsize

## ------------------------------------------------------------------------
set.seed(100)
var.cor <- correlatePairs(sce, design=design, subset.row=rownames(hvg.out),
per.gene=TRUE)
write.table(file="brain_cor.tsv", var.cor, sep="\t", quote=FALSE,
row.names=FALSE)
head(var.cor)
sig.cor <- var.cor$FDR <= 0.05
summary(sig.cor)

## ------------------------------------------------------------------------
library(limma)
adj.exprs <- exprs(sce)
adj.exprs <- removeBatchEffect(adj.exprs, batch=sce$sex)
norm_exprs(sce) <- adj.exprs

## ----tsneplotbrain, fig.cap="**Figure 23:** _t_-SNE plots constructed from
the normalized and corrected log-expression values of correlated HVGs for
cells in the brain dataset. Each point represents a cell and is coloured
according to its expression of the top HVG (left) or _Mog_ (right).",
fig.width=15, fig.height=6----
chosen <- var.cor$gene[sig.cor]
top.hvg <- rownames(hvg.out)[1]
tsne1 <- plotTSNE(sce, exprs_values="norm_exprs", colour_by=top.hvg,
                  perplexity=10, rand_seed=100, feature_set=chosen) +
fontsize
tsne2 <- plotTSNE(sce, exprs_values="norm_exprs", colour_by="Mog",
                  perplexity=10, rand_seed=100, feature_set=chosen) +
fontsize
multiplot(tsne1, tsne2, cols=2)

## ----echo=FALSE, results='hide'-------------------------------------------
rm(var.cor)
gc()

## ----pcaplotbrain, fig.cap="**Figure 24:** PCA plots constructed from the
normalized and corrected log-expression values of correlated HVGs for cells
in the brain dataset. Each point represents a cell and is coloured according
to its expression of the top HVG (left) or _Mog_ (right).", fig.width=15,
fig.height=6----
pca1 <- plotPCA(sce, exprs_values="norm_exprs", colour_by=top.hvg) +
fontsize
pca2 <- plotPCA(sce, exprs_values="norm_exprs", colour_by="Mog") + fontsize
multiplot(pca1, pca2, cols=2)

## ------------------------------------------------------------------------
chosen.exprs <- norm_exprs(sce)[chosen,]
```

```
my.dist <- dist(t(chosen.exprs))
my.tree <- hclust(my.dist, method="ward.D2")

## ------------------------------------------------------------------------
library(dynamicTreeCut)
my.clusters <- unname(cutreeDynamic(my.tree, distM=as.matrix(my.dist),
verbose=0))

## ----heatmapbrain, message=FALSE, fig.width=7, fig.height=10,
fig.cap="**Figure 25:** Heatmap of mean-centred normalized and corrected
log-expression values for correlated HVGs in the brain dataset. Dendrograms
are formed by hierarchical clustering on the Euclidean distances between
genes (row) or cells (column). Column colours represent the cluster to which
each cell is assigned after a dynamic tree cut. Heatmap colours are capped
at a maximum absolute log-fold change of 5."----
heat.vals <- chosen.exprs - rowMeans(chosen.exprs)
clust.col <- rainbow(max(my.clusters))
heatmap.2(heat.vals, col=bluered, symbreak=TRUE, trace='none', cexRow=0.3,
          ColSideColors=clust.col[my.clusters], Colv=as.dendrogram(my.tree),
          breaks=seq(-5, 5, length.out=21))

## ---- results='hide'--------------------------------------------------------
pdf("brain_heat.pdf", width=10, height=100) # Large 'height' to show all
gene names.
heatmap.2(heat.vals, col=bluered, symbreak=TRUE, trace='none', cexRow=0.3,
          ColSideColors=clust.col[my.clusters], Colv=as.dendrogram(my.tree),
          lwid=c(1, 4), lhei=c(1, 50), # Avoid having a very large colour
key.
          breaks=seq(-5, 5, length.out=21))
dev.off()

## ------------------------------------------------------------------------
# Using the technical components computed with a design matrix.
pcs <- denoisePCA(sce, design=design, technical=var.fit.des$trend,
subset.row=chosen)
dim(reducedDimension(pcs)) # Cluster on this instead of t(chosen.exprs)

## ----echo=FALSE, results='hide'--------------------------------------------
gc()

## ------------------------------------------------------------------------
markers <- findMarkers(sce, my.clusters, design=design)

## ---- echo=FALSE, results="hide"--------------------------------------------
old.digits <- options()$digits
options(digits=3)

## ------------------------------------------------------------------------
marker.set <- markers[["1"]]
head(marker.set, 10)

## ---- echo=FALSE, results="hide"--------------------------------------------
options(digits=old.digits)
```

```
## ------------------------------------------------------------------------
write.table(marker.set, file="brain_marker_1.tsv", sep="\t", quote=FALSE,
col.names=NA)

## ----heatmapmarkerbrain, fig.cap="**Figure 26:** Heatmap of mean-centred
normalized and corrected log-expression values for the top set of markers
for cluster 1 in the brain dataset. Column colours represent the cluster to
which each cell is assigned, as indicated by the legend."----
top.markers <- marker.set$Gene[marker.set$Top <= 10]
top.exprs <- norm_exprs(sce)[top.markers,,drop=FALSE]
heat.vals <- top.exprs - rowMeans(top.exprs)
heatmap.2(heat.vals, col=bluered, symbreak=TRUE, trace='none', cexRow=0.6,
          ColSideColors=clust.col[my.clusters], Colv=as.dendrogram(my.tree),
dendrogram='none')
legend("bottomleft", col=clust.col, legend=sort(unique(my.clusters)),
pch=16)

## ------------------------------------------------------------------------
library(edgeR)
y <- convertTo(sce, type="edgeR")

## ------------------------------------------------------------------------
saveRDS(file="brain_data.rds", sce)

## ---- echo=FALSE, results='hide'------------------------------------------
gc()

## ------------------------------------------------------------------------
counts <- read.table("GSE29087_L139_expression_tab.txt.gz",
colClasses=c(list("character",

NULL, NULL, NULL, NULL, NULL, NULL), rep("integer", 96)), skip=6, sep='\t',
row.names=1)
sce <- newSCESet(countData=counts)
sce$grouping <- rep(c("mESC", "MEF", "Neg"), c(48, 44, 4))
sce <- sce[,sce$grouping!="Neg"] # Removing negative control wells.
sce <- calculateQCMetrics(sce, feature_controls=list(spike=grep("SPIKE",
rownames(counts))))
setSpike(sce) <- "spike"

## ------------------------------------------------------------------------
sce <- computeSpikeFactors(sce, general.use=TRUE)

## ------------------------------------------------------------------------
sce <- normalize(sce)

## ----normplotspikemef, fig.cap="**Figure 27:** Size factors from spike-in
normalization, plotted against the size factors from deconvolution for all
cells in the mESC/MEF dataset. Axes are shown on a log-scale, and cells are
coloured according to their identity. Deconvolution size factors were
computed with small pool sizes owing to the low number of cells of each
type."----
```

```
colours <- c(mESC="red", MEF="grey")
deconv.sf <- computeSumFactors(sce, sf.out=TRUE, cluster=sce$grouping,
sizes=1:4*10)
plot(sizeFactors(sce), deconv.sf, col=colours[sce$grouping], pch=16,
log="xy",
     xlab="Size factor (spike-in)", ylab="Size factor (deconvolution)")
legend("bottomleft", col=colours, legend=names(colours), pch=16)

## ------------------------------------------------------------------------
incoming <- as.data.frame(read_excel("nbt.3102-S7.xlsx", sheet=1))
rownames(incoming) <- incoming[,1]
incoming <- incoming[,-1]
incoming <- incoming[,!duplicated(colnames(incoming))] # Remove duplicated
genes.
sce <- newSCESet(exprsData=t(incoming))

## ---- echo=FALSE, results='hide', message=FALSE-------------------------
mm.pairs <- readRDS(system.file("exdata", "mouse_cycle_markers.rds",
package="scran"))
library(org.Mm.eg.db)
fontsize <- theme(axis.text=element_text(size=12),
axis.title=element_text(size=16))
set.seed(100)

## ----phaseplotth2, message=FALSE, fig.cap="**Figure 28:** Cell cycle phase
scores from applying the pair-based classifier on the T~H~2 dataset, where
each point represents a cell."----
anno <- select(org.Mm.eg.db, keys=rownames(sce), keytype="SYMBOL",
column="ENSEMBL")
ensembl <- anno$ENSEMBL[match(rownames(sce), anno$SYMBOL)]
assignments <- cyclone(sce, mm.pairs, gene.names=ensembl, assay="exprs")
plot(assignments$score$G1, assignments$score$G2M, xlab="G1 score",
ylab="G2/M score", pch=16)

## ------------------------------------------------------------------------
design <- model.matrix(~ G1 + G2M, assignments$score)
fit.block <- trendVar(sce, use.spikes=NA, trend="loess", design=design)
dec.block <- decomposeVar(sce, fit.block)

## ----pcaplotth2, fig.width=12, fig.height=6, fig.cap="**Figure 29:** PCA
plots before (left) and after (right) removal of the cell cycle effect in
the T~H~2 dataset. Each cell is represented by a point with colour and size
determined by the G1 and G2/M scores, respectively. Only HVGs were used to
construct each plot."----
# Finding HVGs without blocking on phase score.
fit <- trendVar(sce, use.spikes=NA, trend="loess")
dec <- decomposeVar(sce, fit)
top.hvgs <- which(dec$FDR <= 0.05 & dec$bio >= 0.5)
sce$G1score <- assignments$score$G1
sce$G2Mscore <- assignments$score$G2M
out <- plotPCA(sce, feature_set=top.hvgs, colour_by="G1score",
size_by="G2Mscore") +
  fontsize + ggtitle("Before removal")
```

```
# Using HVGs after blocking on the phase score.
top.hvgs2 <- which(dec.block$FDR <= 0.05 & dec.block$bio >= 0.5)
norm_exprs(sce) <- removeBatchEffect(exprs(sce),
covariates=assignments$score[,c("G1", "G2M")])
out2 <- plotPCA(sce, exprs_values="norm_exprs", feature_set=top.hvgs2,
colour_by="G1score",
                size_by="G2Mscore") + fontsize + ggtitle("After removal")
multiplot(out, out2, cols=2)

## ----diffusionth2, fig.cap="**Figure 30:** A diffusion map for the T~H~2
dataset, where each cell is coloured by its expression of _Gata3_."----
plotDiffusionMap(sce, exprs_values="norm_exprs", colour_by="Gata3") +
fontsize

## ---- echo=FALSE, results='hide'-----------------------------------------
saveRDS(file="th2_data.rds", sce)
gc()

## ------------------------------------------------------------------------
incoming <- read.table("counttable_es.csv", header=TRUE, row.names=1)
my.ids <- rownames(incoming)
anno <- select(org.Mm.eg.db, keys=my.ids, keytype="ENSEMBL",
column="SYMBOL")
anno <- anno[match(my.ids, anno$ENSEMBL),]
head(anno)

## ------------------------------------------------------------------------
library(TxDb.Mmusculus.UCSC.mm10.ensGene)
location <- select(TxDb.Mmusculus.UCSC.mm10.ensGene, keys=my.ids,
                   column="CDSCHROM", keytype="GENEID")
location <- location[match(my.ids, location$GENEID),]
is.mito <- location$CDSCHROM == "chrM" & !is.na(location$CDSCHROM)
sum(is.mito)

## ------------------------------------------------------------------------
is.spike <- grepl("^ERCC", my.ids)
sum(is.spike)

## ------------------------------------------------------------------------
anno <- anno[,-1,drop=FALSE]
rownames(anno) <- my.ids
sce <- newSCESet(countData=incoming, featureData=AnnotatedDataFrame(anno))
sce <- calculateQCMetrics(sce, feature_controls=list(ERCC=is.spike))
setSpike(sce) <- "ERCC"

## ------------------------------------------------------------------------
sce <- sce[grepl("ENSMUS", rownames(sce)) | isSpike(sce),]
dim(sce)

## ---- echo=FALSE, results='hide'-----------------------------------------
saveRDS(file="mesc_data.rds", sce)
gc()
```

```
## ---- echo=FALSE, results='asis'-----------------------------------------
if (!on.bioc) {
  cat("Users can install all required packages and execute the workflow by
following the instructions at
https://www.bioconductor.org/help/workflows/simpleSingleCell.\n")
}
cat("The workflow takes less than an hour to run on a desktop computer with
8 GB of memory.\n")

## ------------------------------------------------------------------------
sessionInfo()

## ---- eval=on.bioc, echo=FALSE, results='hide'---------------------------
unlink(all.basenames)
unlink("GSE61533_HTSEQ_count_results.xls")
```