Transcriptional variability analysis of CD4$^+$ T cells using BASiCS

Alan O'Callaghan Nils Eling John C. Marioni Catalina A. Vallejos

Contents

1 Introduction **1**

1.1 Loading pre-run MCMC chains 1

2 Required packages **2**

3 MCMC convergence diagnosics **2**

4 Comparison of parameter estimates with and without spike-ins **3**

5 HVG/LVG detection using BASiCS **5**

6 Differential mean and variability testing using BASiCS **8**

1 Introduction

As a case study, we use scRNA-seq data generated for CD4$^+$ T cells using the C1 Single-Cell Auto Prep System (Fluidigm®). Martinez-Jimenez et al. (2017) profiled naive (hereafter also referred to as unstimulated) and activated (three hours using in vitro antibody stimulation) CD4$^+$ T cells from young and old animals across two mouse strains to study changes in expression variability during ageing and upon immune activation. They extracted naive or effector memory CD4$^+$ T cells from spleens of young or old animals, obtaining purified populations using either magnetic-activated cell sorting (MACS) or fluorescence activated cell sorting (FACS). External ERCC spike-in RNA18 was added to aid the quantification of technical variability across all cells and all experiments were performed in replicates (hereafter also referred to as batches).

1.1 Loading pre-run MCMC chains

Data preparation and MCMC inference for this analysis is described in the accompanying document `DataPreparationCD4T`. Each of the MCMC runs performed is time consuming (a few hours in a typical laptop). Hence, for convenience, we have uploaded our chains to a Zenodo repository. These can be downloaded and used as the input for subsequent analyses.

```
sce_naive <- readRDS("rds/sce_naive.Rds")
sce_active <- readRDS("rds/sce_active.Rds")

chains_website <- "https://git.ecdf.ed.ac.uk/vallejosgroup/basicsworkflow2020/-/raw/master/"
# chains_website <- "https://zenodo.org/record/5243265/files/"
```

```r
options(timeout = 1000)
if (!file.exists("rds/chain_naive.Rds")) {
  download.file(
    paste0(chains_website, "rds/chain_naive.Rds"),
    destfile = "rds/chain_naive.Rds"
  )
}
if (!file.exists("rds/chain_active.Rds")) {
  download.file(
    paste0(chains_website, "rds/chain_active.Rds"),
    destfile = "rds/chain_active.Rds"
  )
}
if (!file.exists("rds/chain_naive_nospikes.Rds")) {
  download.file(
    paste0(chains_website, "rds/chain_naive_nospikes.Rds"),
    destfile = "rds/chain_naive_nospikes.Rds"
  )
}
chain_naive <- readRDS("rds/chain_naive.Rds")
chain_active <- readRDS("rds/chain_active.Rds")
chain_naive_nospikes <- readRDS("rds/chain_naive_nospikes.Rds")
```

2 Required packages

TODO: describe

```r
library("BASiCS")
# Data wrangling
library("reshape2")
# Plotting packages
library("ggplot2")
library("ggpointdensity")
library("ComplexHeatmap")
# Arranging multiple ggplot2 plots
library("patchwork")
# Color scale management
library("circlize")
library("RColorBrewer")
library("viridis")
```

3 MCMC convergence diagnosics

BASiCS uses an adaptive Metropolis-within-Gibbs algorithm (Roberts and Rosenthal 2009) to perform inference. Before interpreting the results, it is important to ensure that the algorithm has converged. There are multiple ways to do this. In principle, we could visualise the history of iterations for each model parameter in order to assess convergence. However, this is impractical as *BASiCS* infers thousands of parameters. Instead, we use the `BASiCS_DiagPlot()` to visualise diagnostic metrics calculated for each parameter. In particular, it can calculate the effective sample size and the Geweke diagnostic criteria (Geweke and In

1995). For simplicity, here we only focus on the MCMC chains obtained for gene-specific mean expression parameters (μ_i) when analysing the naive CD4$^+$ T cells with spike-ins. However, a similar analysis must be applied to all other chains before they are used in downstream analysis.

In this case, we can see positive evidence to support convergence of the MCMC: most Geweke Z-scores are within the dashed lines and most genes have a high effective sample size (above 500).

4 Comparison of parameter estimates with and without spike-ins

Under the horizontal integration approach described above, the scale of mean expression parameters and global scaling factors is not jointly identifiable, in that a global shift in mean expression parameters could be exactly offset by an equivalent shift in cell-specific normalisation parameters. Therefore, the geometric mean of the mean expression parameters is fixed to a constant value. Relative expression level estimates are broadly consistent between the horizontal and vertical integration approaches; however there may be a global difference in mean expression estimates, as shown in Figure 1. It is important to remove this global scale offset before performing comparative analyses. This is performed by default in `BASiCS_TestDE`, but can be performed manually using `BASiCS_CorrectOffset`.

```
BASiCS_PlotOffset(chain_naive_nospikes, chain_naive,
  GroupLabel1 = "No spike-ins", GroupLabel2 = "Spike-ins",
  Type = "before-after")
```

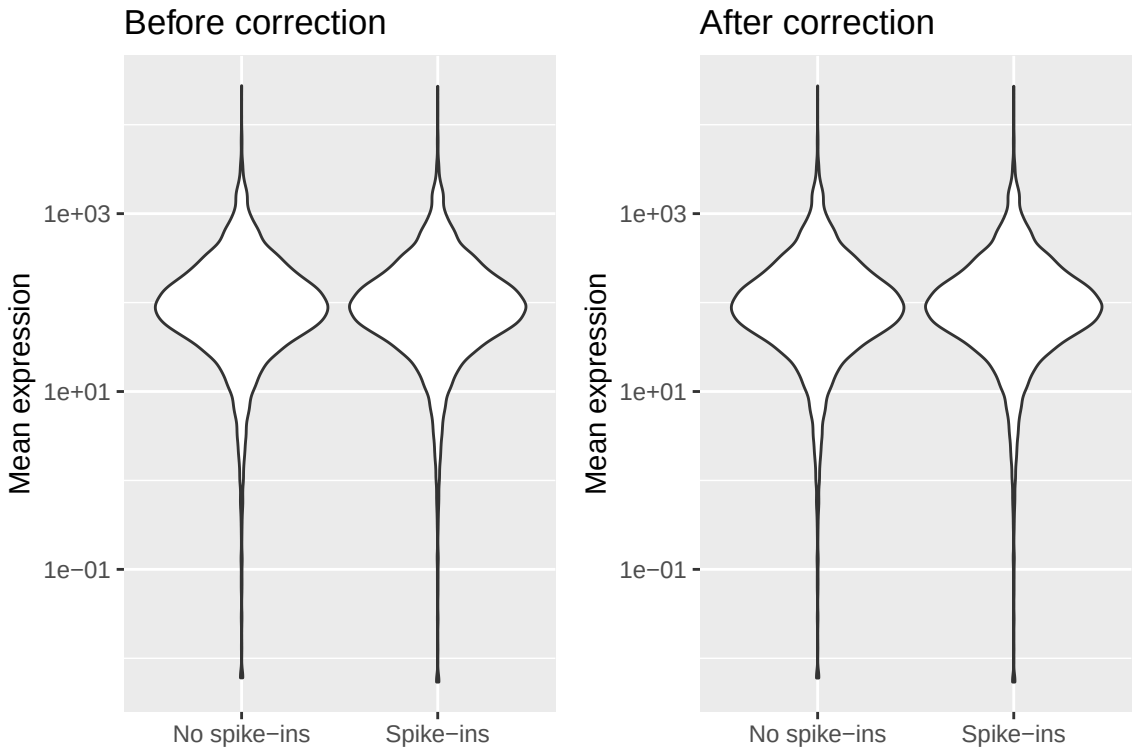


Figure 1: Distribution of mean expression values before and after correcting the global difference in scale.

```
offset <- BASiCS_CorrectOffset(chain_naive_nospikes, chain_naive)
chain_naive_nospikes_offset <- offset$Chain
chain_naive_nospikes_offset
```

```
## An object of class BASiCS_Chain
##  1000 MCMC samples.
##  Dataset contains 5171 biological genes and 93 cells (1 batch).
##  Object stored using BASiCS version:  2.14.0
##  Parameters:  mu delta s nu theta beta sigma2 epsilon RefFreq RBFLocations
```

A number of genes have very low expression estimates in the naive population, due to the fact that they each have zero read counts across the entire naive population; we therefore remove these genes before making a comparison. Following removal of the global offset, the mean expression and over-dispersion estimates obtained from each method are directly comparable. As seen in Figures 2A and 2B, parameter point estimates from the two methods are highly correlated. There is a tail of non-expressed genes with very low mean expression level as inferred without spike-ins, comprising those genes with no measured expression across the entire population.

```
mu_spikes <- displayChainBASiCS(chain_naive)
mu_nospikes <- displayChainBASiCS(chain_naive_nospikes_offset)

# Remove genes with zero counts across all cells and calculate medians
ind_nonzero <- rowSums(counts(sce_naive)) != 0
mu_spikes <- colMedians(mu_spikes[, ind_nonzero])
mu_nospikes <- colMedians(mu_nospikes[, ind_nonzero])

g1 <- ggplot() +
  aes(mu_spikes, mu_nospikes) +
  geom_pointdensity(alpha = 0.7) +
  scale_colour_viridis(name = "Density") +
  scale_x_log10() +
  scale_y_log10() +
  geom_abline(
    colour = "firebrick",
    linetype = "dashed",
    slope = 1,
    intercept = 0
  ) +
  labs(
    x = "Mean expression\n(with spike-ins)",
    y = "Mean expression\n(without spike-ins)"
  ) +
  theme(
    legend.position = "bottom",
    legend.text = element_text(angle = 45, size = 8, hjust = 0.5, vjust = 0.5)
  )

delta_spikes <- displayChainBASiCS(chain_naive, Param = "delta")
delta_nospikes <- displayChainBASiCS(chain_naive_nospikes_offset, Param = "delta")

g2 <- ggplot() +
  aes(colMedians(delta_spikes), colMedians(delta_nospikes)) +
  geom_pointdensity(alpha = 0.7) +
  scale_colour_viridis(name = "Density") +
  scale_x_log10() +
  scale_y_log10() +
  geom_abline(
    colour = "firebrick",
```

```
    linetype = "dashed",
    slope = 1,
    intercept = 0
  ) +
  labs(
    x = "Over-dispersion\n(with spike-ins)",
    y = "Over-dispersion\n(without spike-ins)"
  ) +
  theme(
    legend.position = "bottom",
    legend.text = element_text(angle = 45, size = 8, hjust = 0.5, vjust = 0.5)
  )
g1 + g2 + plot_annotation(tag_levels = "A")
```

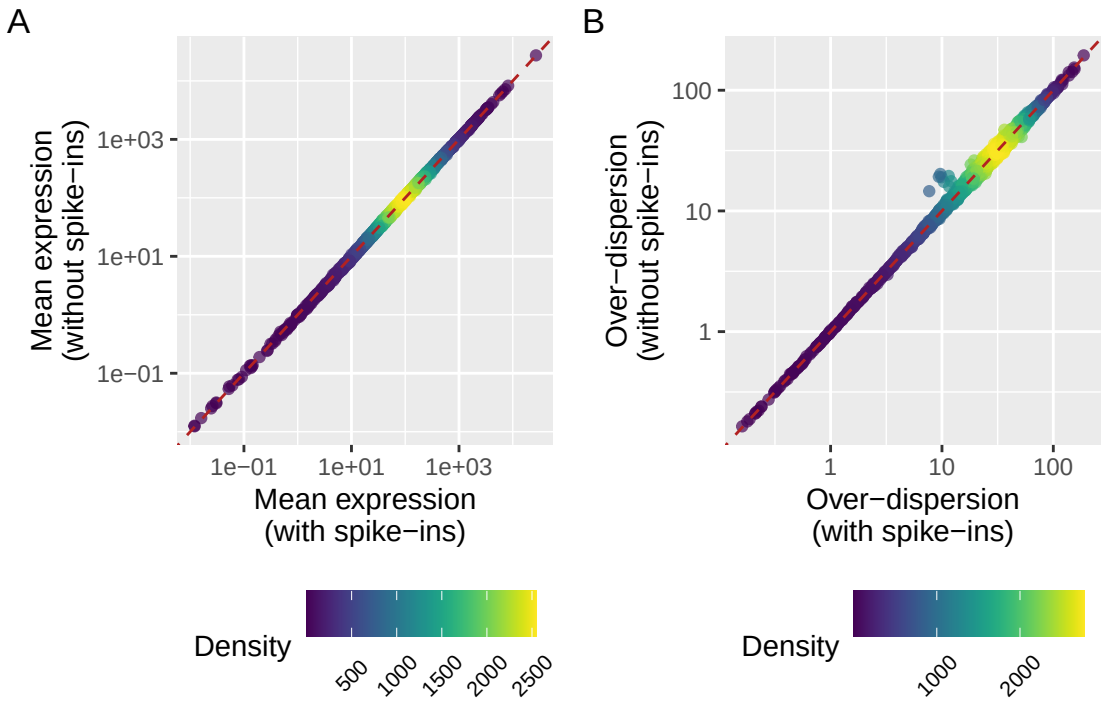


Figure 2: Comparison of point estimates using spike-ins, and the same parameters estimated without using spike-ins for mean expression (A) and over-dispersion (B). A dashed red line indicates the identity line, $y = x$. Genes with zero counts across all cells were excluded from the plot of mean expression parameters.

5 HVG/LVG detection using BASiCS

The functions `BASiCS_DetectHVG` and `BASiCS_DetectLVG` enable the identification of genes with substantially high (HVG) or low (LVG) transcriptional variability within a population of cells.

We label HVGs those genes for which their residual over-dispersion (ϵ_i) exceeds a pre-defined threshold with high probability. Similarly, LVGs are those whose ϵ_i is below a pre-specified threshold with high probability. In both cases, the probability cut-off is chosen using a grid-search that aims to match a given expected

false discovery rate (EFDR; EFDR=10% as default) (Newton 2004). Here, we illustrate this using the naive CD4$^+$ T cells (with spike-ins)

```
## Highly variable genes
hvg <- BASiCS_DetectHVG(chain_naive, EpsilonThreshold = log(2))
## Lowly variable genes
lvg <- BASiCS_DetectLVG(chain_naive, EpsilonThreshold = -log(2))
```

```
## For LVG detection task:
## the posterior probability threshold chosen via EFDR calibration is too low.
## Probability threshold automatically set equal to 'ProbThreshold'.
```

```
## Merge both tables
vg_table <- merge(
  as.data.frame(hvg, Filter = FALSE),
  as.data.frame(lvg, Filter = FALSE),
  all = TRUE
)

## mark genes as highly variable, lowly variable, or not either.
vg_table$VG <- "Not HVG or LVG"
vg_table$VG [vg_table$HVG] <- "HVG"
vg_table$VG [vg_table$LVG] <- "LVG"
ggplot (vg_table) +
  aes(log10(Mu), Epsilon, colour = VG) +
  geom_point() +
  geom_hline(yintercept = 0, lty = 2) +
  labs(
    x = "BASiCS means (log10)",
    y = "BASiCS residual\nover-dispersion"
    ) +
  scale_colour_manual(name = NULL,
                      values = c(
                        "HVG" = "firebrick",
                        "LVG" = "dodgerblue",
                        "Not HVG or LVG" = "grey80"),
                      na.value = "grey80")
```

For the naive CD4$^+$ T cell data, we obtained 75 HVG and rsum(vg_table$VG == "LVG")‘ 380 LVG. As shown in Figure 3, these genes are distributed across a wide range of mean expression values. As an illustration, Figure 4 shows the distribution of normalised expression values for selected HVG and LVG, focusing on examples with similar mean expression levels. As expected, HVG tend to exhibit a wider distribution and potentially bimodal distribution (Figure 4A). Instead, LVG tend to have more narrow and unimodal distributions (Figure 4B).

```
# load file created in DataPreparationCD4T.Rmd
genenames <- readRDS("rds/genenames.Rds")

## Obtain normalised expression values
dc_naive <- BASiCS_DenoisedCounts(sce_naive, chain_naive)
vg_table <- merge(
  as.data.frame(lvg, Filter = FALSE),
  as.data.frame(hvg, Filter = FALSE),
```

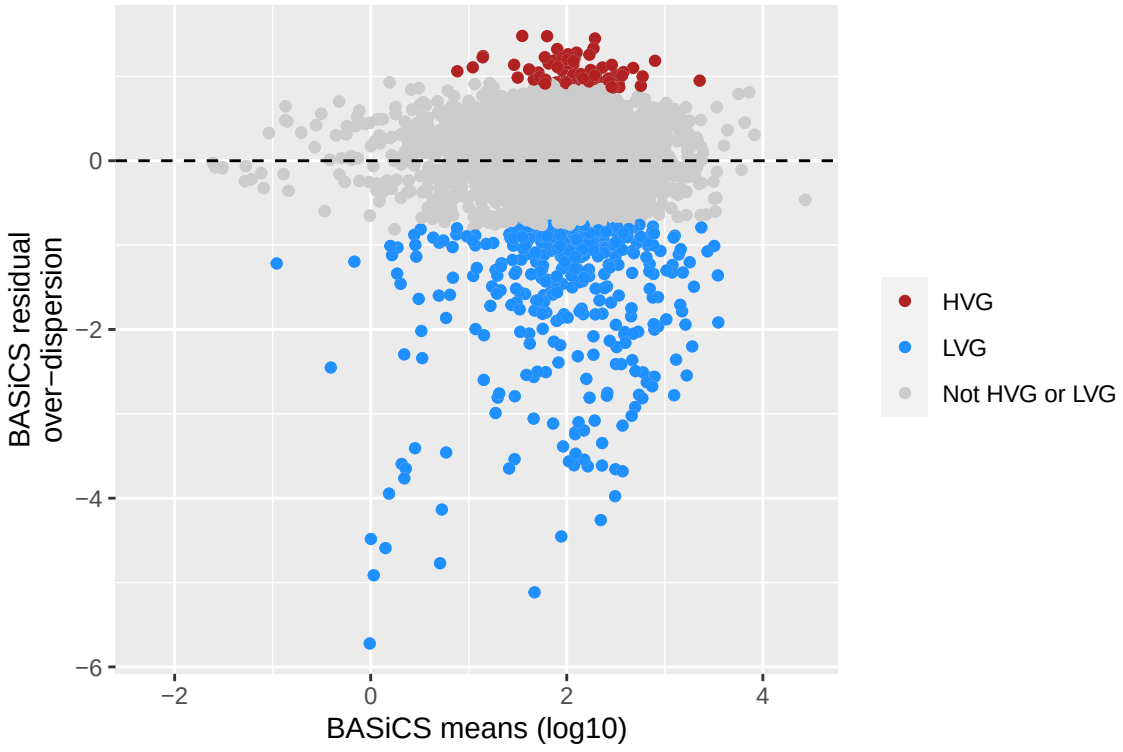


Figure 3: HVG and LVG detection using BASiCS (naive CD4$^+$ T cells, with spike-ins). For each gene, BASiCS posterior estimates (posterior medians) associated to mean expression and residual over-dispersion parameters are plotted. Genes are coloured according to HVG/LVG status. Genes that are not expressed in at least 2 cells are excluded.

```
  by = c("GeneName", "GeneIndex", "Mu", "Delta", "Epsilon"),
  suffixes = c("LVG", "HVG")
)
vg_table <- merge(
  vg_table,
  genenames,
  by.x = "GeneName", by.y = "ensembl_gene_id",
  sort = FALSE
)

## Select HVG/LVG genes with similar mean expression values
low_exp <- 2
up_exp <- 3
is_mid_exp <- log10(vg_table$Mu) > low_exp & log10(vg_table$Mu) < up_exp
hvg_table <- vg_table[which(is_mid_exp & vg_table$HVG),]
lvg_table <- vg_table[which(is_mid_exp & vg_table$LVG),]

## Order by epsilon and select top 3 HVG and LVG within the genes selected above
top_hvg <- order(hvg_table$Epsilon, decreasing = TRUE)[1:3]
top_lvg <- order(lvg_table$Epsilon, decreasing = FALSE)[1:3]
hvg_counts <- log10(t(dc_naive[hvg_table$GeneName[top_hvg],]) + 1)
lvg_counts <- log10(t(dc_naive[lvg_table$GeneName[top_lvg],]) + 1)

## Add genenames
colnames(hvg_counts) <- hvg_table$external_gene_name[top_hvg]
colnames(lvg_counts) <- lvg_table$external_gene_name[top_lvg]
plot_params <- list(
  geom_violin(na.rm = TRUE),
  coord_flip(),
  ylim(-0.05, max(log10(dc_naive + 1))),
  geom_jitter(position = position_jitter(0.3)),
  ylab("log10(norm count + 1)"),
  xlab("Gene")
)
plot_hvg <- ggplot(melt(hvg_counts), aes(x = Var2, y = value)) +
  plot_params + ggtitle("Example HVGs")
plot_lvg <- ggplot(melt(lvg_counts), aes(x = Var2, y = value)) +
  plot_params + ggtitle("Example LVGs")
plot_hvg + plot_lvg + plot_annotation(tag_levels = "A")
```

6 Differential mean and variability testing using BASiCS

Finally, we perform differential testing to identify genes with different expression patterns between naive and stimulated CD4$^+$ T cells. This considers both genes with changes in mean expression and those whose transcriptional variability changes between the naive and stimulated cells. The results are summarised in Fig @ref{fig:de-cd4}.

```
## Perform differential testing
test_de <- BASiCS_TestDE(
  Chain1 = chain_naive,
  Chain2 = chain_active,
```

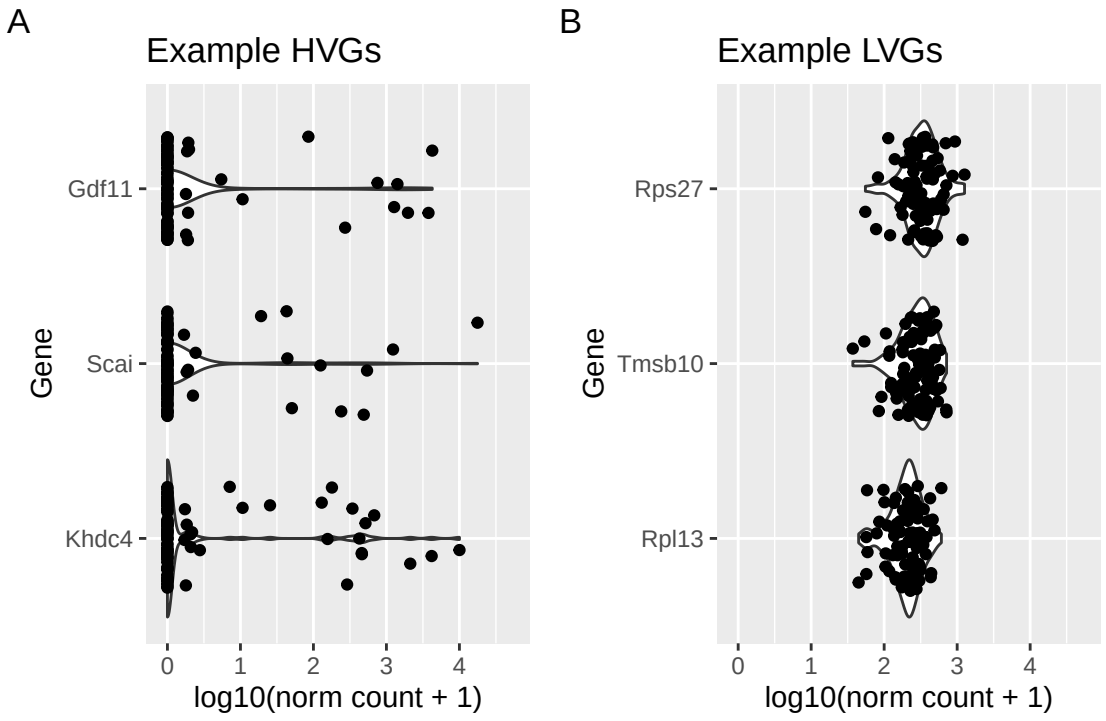


Figure 4: Examples of HVG and LVG within naive CD4+T cells

```
  GroupLabel1 = "Naive",
  GroupLabel2 = "Active",
  EFDR_M = 0.1,
  EFDR_R = 0.1,
  MinESS = 100,
  Plot = FALSE,
  PlotOffset = FALSE
)
p1 <- BASiCS_PlotDE(test_de, Parameters = "Mean", Plots = "MA")
p2 <- BASiCS_PlotDE(test_de, Parameters = "Mean", Plots = "Volcano")
p1 / p2
```

To facilate interpretation, we also visualise expression patterns for differentially expressed genes. To do this, we calculate normalised expression values for each cell and gene, correcting for a global offset between naive and activated cells.

```
offset <- BASiCS_CorrectOffset(chain_naive, chain_active)
offset$Offset
```

```
## [1] 0.7334975
```

```
## Get offset corrected chain
chain_naive <- offset$Chain
## Obtain normalised counts within each group of cells
dc_naive <- BASiCS_DenoisedCounts(sce_naive, chain_naive, WithSpikes = FALSE)
dc_active <- BASiCS_DenoisedCounts(sce_active, chain_active, WithSpikes = FALSE)
```

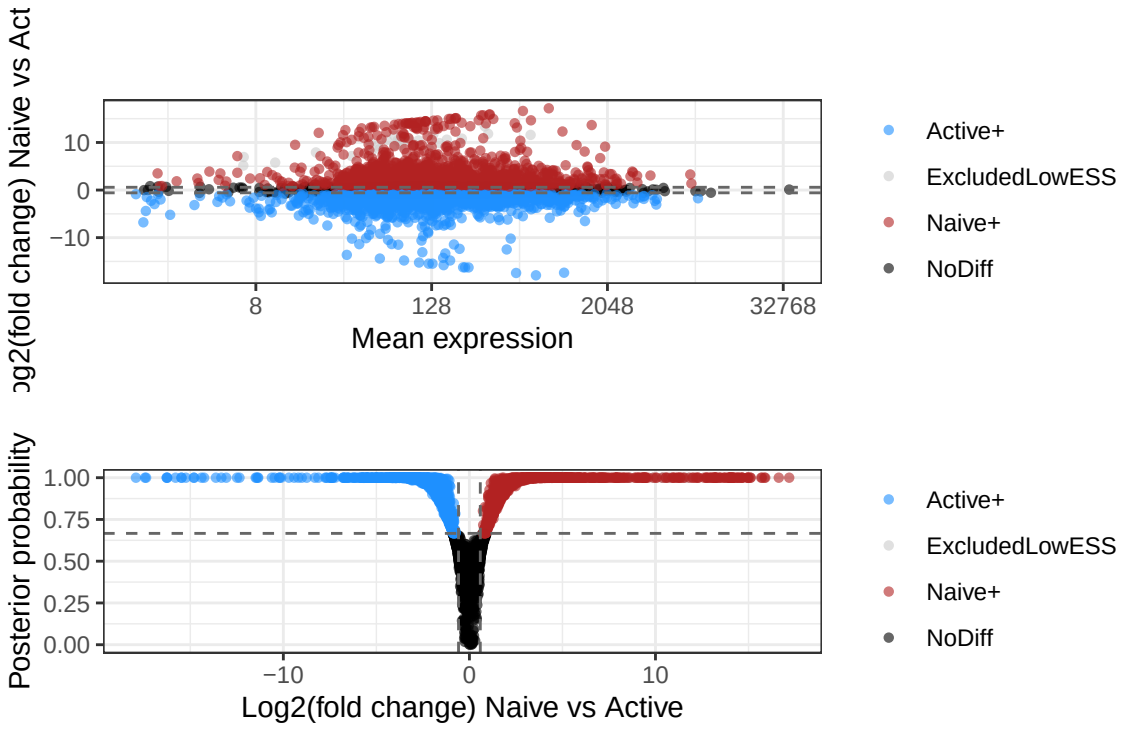
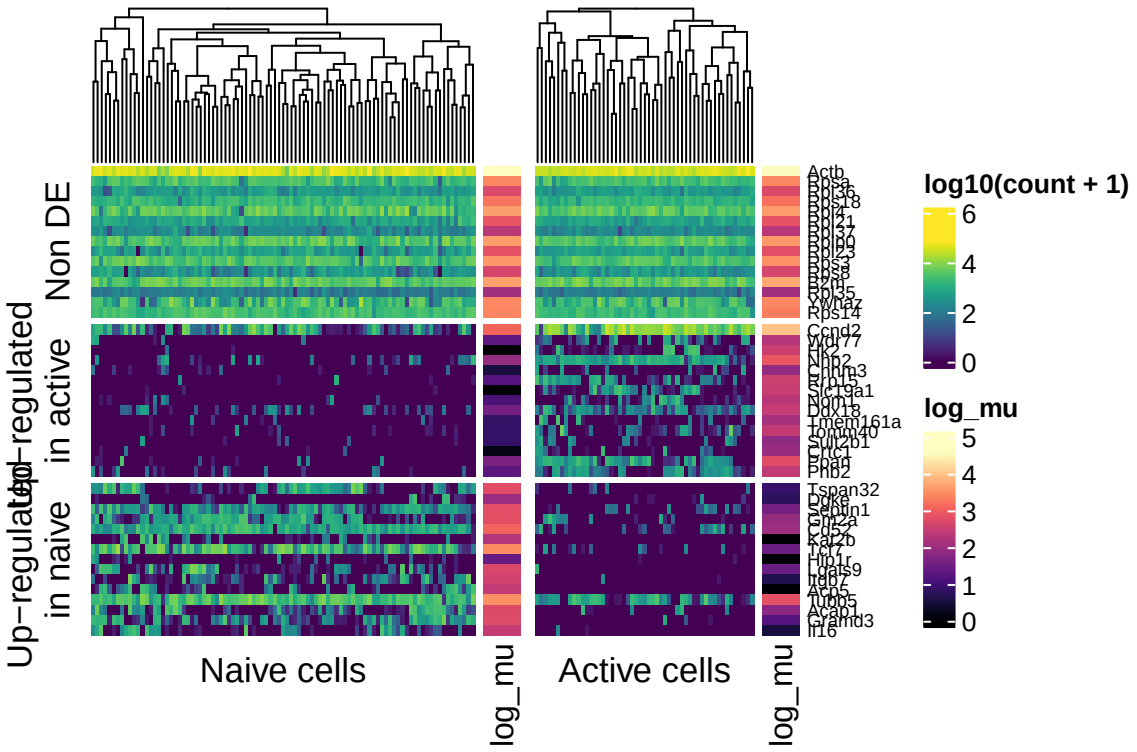


Figure 5: Upper panel presents the mean-difference plot associated to the differential mean expression test between naive and active cells. Log-fold changes of average expression in naive cells relative to active cells are plotted against average expression estimates combined across both groups of cells. Bottom panel presents the volcano plot associated to the same test. Log-fold changes of average expression in naive cells relative to active cells are plotted against their associated tail posterior probabilities. Colour indicates the differential expression status for each gene, including a label to identify genes that were excluded from differential expression test due to low effective sample size.

We then use *ComplexHeatmap* to visualise expression patterns for different groups of genes as defined by the differential expression test above.

```
table_de_mean <- as.data.frame(test_de,
                               Parameter = "Mean",
                               Filter = FALSE)
table_de_resdisp <- as.data.frame(test_de,
                                  Parameter = "ResDisp",
                                  Filter = FALSE)
## Add gene symbol to table
table_de_mean$Symbol <- genenames[table_de_mean$GeneName, 2]
## utility function for selecting genes
select_top_n <- function(table, counts, condition, n=15, decreasing=TRUE) {
  ind_condition <- table$ResultDiffMean == condition
  table <- table[ind_condition,]
  ind_diff <- order(table$ProbDiffMean, decreasing = decreasing)[1:n]
  genes <- table$GeneName[ind_diff]
  counts[genes,]
}
## Active & naive count matrices for genes up-regulated in active cells
act_counts_act <- select_top_n(table_de_mean, dc_active, "Active+")
nai_counts_act <- select_top_n(table_de_mean, dc_naive, "Active+")
## Active & naive count matrices for genes up-regulated in naive cells
act_counts_nai <- select_top_n(table_de_mean, dc_active, "Naive+")
nai_counts_nai <- select_top_n(table_de_mean, dc_naive, "Naive+")
## Active & naive count matrices for genes not differentially expressed
act_counts_nde <- select_top_n(table_de_mean, dc_active, "NoDiff",
                               decreasing = FALSE)
nai_counts_nde <- select_top_n(table_de_mean, dc_naive, "NoDiff",
                               decreasing = FALSE)
## Combine count matrices by cell type
counts_active <- rbind(act_counts_act, act_counts_nai, act_counts_nde)
counts_naive <- rbind(nai_counts_act, nai_counts_nai, nai_counts_nde)
## split heatmaps by gene category
split <- data.frame(
  Upregulated = c(
    rep("Up-regulated \nin active", nrow(act_counts_act)),
    rep("Up-regulated \nin naive", nrow(act_counts_nai)),
    rep("Non DE", nrow(act_counts_nde))
    )
)
syms <- genenames[rownames(counts_active), 2]
fontsize <- 7
## Color palettes using circlize
col <- colorRamp2(
  breaks = seq(0,
               log10(max(c(counts_naive, counts_active) + 1)),
               length.out = 20),
  colors = viridis(20)
)
## Subset table of DE results to extract mean estimates
match_order <- match(rownames(counts_naive), table_de_mean$GeneName)
```

```r
table_de_selected <- table_de_mean[match_order,]
## Color palette for mu annotation
log_mu_naive <- log10(table_de_selected$Mean1)
log_mu_active <- log10(table_de_selected$Mean2)
mu_col <- colorRamp2(
  breaks = seq(0, max(c(log_mu_naive, log_mu_active)), length.out = 20),
  colors = viridis(20, option = "A", direction = 1)
)
Heatmap(
  log10(counts_naive + 1),
  row_labels = syms,
  row_names_gp = gpar(fontsize = fontsize),
  name = "log10(count + 1)",
  column_dend_height = unit(0.2, "npc"),
  column_title_side = "bottom",
  column_title = "Naive cells",
  show_column_names = FALSE,
  cluster_rows = FALSE,
  split = split,
  right_annotation = rowAnnotation(
    log_mu = log_mu_naive,
    col = list(log_mu = mu_col)
    ),
  col = col) +
Heatmap(
  log10(counts_active + 1),
  row_labels = syms,
  column_dend_height = unit(0.2, "npc"),
  row_names_gp = gpar(fontsize = fontsize),
  column_title = "Active cells",
  column_title_side = "bottom",
  show_column_names = FALSE,
  split = split,
  show_heatmap_legend = FALSE,
  right_annotation = rowAnnotation(
    log_mu = log_mu_active,
    col = list(log_mu = mu_col)),
  cluster_rows = FALSE,
  col = col)
```



Geweke, John, and Forthcoming In. 1995. "Evaluating the Accuracy of Sampling-Based Approaches to the Calculation of Posterior Moments" 4 (November).

Martinez-Jimenez, Celia P., Nils Eling, Hung-Chang Chen, Catalina A Vallejos, Aleksandra A Kolodziejczyk, Frances Connor, Lovorka Stojic, et al. 2017. "Aging increases cell-to-cell transcriptional variability upon immune stimulation." *Science* 1436: 1433–36. https://doi.org/10.1126/science.aah4115.

Newton, M. A. 2004. "Detecting Differential Gene Expression with a Semiparametric Hierarchical Mixture Method." *Biostatistics* 5 (2): 155–76. https://doi.org/10.1093/biostatistics/5.2.155.

Roberts, Gareth O., and Jeffrey S. Rosenthal. 2009. "Examples of Adaptive MCMC." *Journal of Computational and Graphical Statistics* 18 (2): 349–67. https://doi.org/10.1198/jcgs.2009.06134.