SOFTWARE TOOL ARTICLE

BASiCS workflow: a step-by-step analysis of expression variability using single cell RNA sequencing data [version 1; peer review: 3 approved with reservations]

Alan O'Callaghan[1], Nils Eling[2,3], John C. Marioni[4,5], Catalina A. Vallejos[1,6]

[1]MRC Human Genetics Unit, Institute of Genetics & Cancer, University of Edinburgh, Edinburgh, EH4 2XU, UK
[2]Department of Quantitative Biomedicine, University of Zurich, Zürich, CH-8057, Switzerland
[3]Institute for Molecular Health Sciences, ETH Zürich, Zürich, 8093, Switzerland
[4]European Molecular Biology Laboratory, European Bioinformatics Institute, Cambridge, CB10 1SD, UK
[5]Cancer Research UK Cambridge Institute, University of Cambridge, Cambridge, CB2 0RE, UK
[6]The Alan Turing Institute, The Alan Turing Institute, London, NW1 2DB, UK

v1 **First published:** 18 Jan 2022, **11**:59
https://doi.org/10.12688/f1000research.74416.1
Latest published: 07 May 2024, **11**:59
https://doi.org/10.12688/f1000research.74416.2

Abstract

Cell-to-cell gene expression variability is an inherent feature of complex biological systems, such as immunity and development. Single-cell RNA sequencing is a powerful tool to quantify this heterogeneity, but it is prone to strong technical noise. In this article, we describe a step-by-step computational workflow that uses the BASiCS Bioconductor package to robustly quantify expression variability within and between known groups of cells (such as experimental conditions or cell types). BASiCS uses an integrated framework for data normalisation, technical noise quantification and downstream analyses, propagating statistical uncertainty across these steps. Within a single seemingly homogeneous cell population, BASiCS can identify highly variable genes that exhibit strong heterogeneity as well as lowly variable genes with stable expression. BASiCS also uses a probabilistic decision rule to identify changes in expression variability between cell populations, whilst avoiding confounding effects related to differences in technical noise or in overall abundance. Using a publicly available dataset, we guide users through a complete pipeline that includes preliminary steps for quality control, as well as data exploration using the scater and scran Bioconductor packages. The workflow is accompanied by a Docker image that ensures the reproducibility of our results.

Keywords

single-cell RNA sequencing, expression variability, transcriptional noise, differential expression testing, scRNAseq, Bayesian, bioinformatics, heterogeneity

Open Peer Review

Approval Status ? ? ?

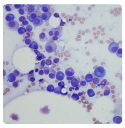
	1	2	3
version 2 (revision) 07 May 2024			
version 1 18 Jan 2022	? view	? view	? view

1. **Oliver M. Crook**, University of Oxford, Oxford, UK
2. **Andrew McDavid**, University of Rochester, Rochester, USA
3. **Michel S Naslavsky**, University of São Paulo, São Paulo, Brazil

Any reports and responses or comments on the article can be found at the end of the article.

This article is included in the Bioinformatics gateway.

This article is included in the Bioconductor gateway.



This article is included in the Cell & Molecular Biology gateway.

This article is included in the RPackage gateway.

Corresponding author: Catalina A. Vallejos (Catalina.vallejos@ed.ac.uk)

Author roles: O'Callaghan A: Conceptualization, Software, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing; **Eling N**: Conceptualization, Methodology, Software, Writing – Original Draft Preparation, Writing – Review & Editing; **Marioni JC**: Project Administration, Supervision, Writing – Review & Editing; **Vallejos CA**: Conceptualization, Project Administration, Software, Supervision, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: C.A.V. is a Chancellor's Fellow funded by the University of Edinburgh. A.OC. was funded by the Chancellor's Fellowship granted to C.A.V. N.E. was funded by the European Molecular Biology Laboratory International PhD Programme. Work by JCM was supported by CRUK (C9545/A29580) and by core support from EMBL
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

How to cite this article: O'Callaghan A, Eling N, Marioni JC and Vallejos CA. **BASiCS workflow: a step-by-step analysis of expression variability using single cell RNA sequencing data [version 1; peer review: 3 approved with reservations]** F1000Research 2022, **11**:59 https://doi.org/10.12688/f1000research.74416.1

First published: 18 Jan 2022, **11**:59 https://doi.org/10.12688/f1000research.74416.1

Introduction

Single-cell RNA-sequencing (scRNA-seq) enables the study of genome-wide cell-to-cell transcriptional heterogeneity that is not captured by bulk experiments.[1–3] On the broadest level, this heterogeneity can reflect the presence of distinct cell subtypes or states. Alternatively, it can be due to gradual changes along biological processes, such as development and differentiation. Several clustering and pseudotime inference tools have been developed to capture these types of heterogeneity.[4,5] However, there is a limited availability of computational tools tailored to study more subtle variability within seemingly homogeneous cell populations. This variability can reflect deterministic or stochastic events that regulate gene expression and, among other settings, has been seen to increase prior to cell fate decisions[6] and during ageing.[7] Transcriptional variability has also been observed to differ from gene to gene and can be conserved across cell types and species.[8]

Stochastic variability within a seemingly homogeneous cell population — often referred to as transcriptional *noise* — can arise from intrinsic and extrinsic sources.[9,10] Extrinsic noise refers to stochastic fluctuations induced by different dynamic cellular states (e.g. cell cycle, metabolism, intra/inter-cellular signalling).[11–13] In contrast, intrinsic noise arises from stochastic effects on biochemical processes such as transcription and translation.[9] Intrinsic noise can be modulated by genetic and epigenetic modifications (such as mutations, histone modifications, CpG island length and nucleosome positioning)[14–16] and usually occurs at the gene level.[9] Cell-to-cell gene expression variability estimates derived from scRNA-seq data capture a combination of these effects, as well as deterministic regulatory mechanisms.[10] Moreover, these variability estimates can also be inflated by the technical noise that is typically observed in scRNA-seq data.[17]

Different strategies have been incorporated into scRNA-seq protocols to control or attenuate technical noise. For example, external RNA spike-in molecules (such as the set introduced by the External RNA Controls Consortium, ERCC[18]) can be added to each cell's lysate in a (theoretically) known fixed quantity. Spike-ins can assist quality control steps,[19] data normalisation[20] and can be used to infer technical noise.[17] Another strategy is to tag individual cDNA molecules using unique molecular identifiers (UMIs) before PCR amplification.[21] Reads that contain the same UMI can be collapsed into a single molecule count, attenuating technical variability associated to cell-to-cell differences in amplification and sequencing depth (these technical biases are not fully removed unless sequencing to saturation[20]). However, despite the benefits associated to the use of spike-ins and UMIs, these are not available for all scRNA-seq protocols.[22]

The Bioconductor package *BASiCS* implements a Bayesian hierarchical framework that accounts for both technical and biological sources of noise in scRNA-seq datasets.[23–25] *BASiCS* jointly performs data normalisation, technical noise quantification and downstream analyses, whilst propagating statistical uncertainty across these steps. These features are implemented within a probabilistic model that builds upon a negative binomial framework, a widely used distribution in the context of bulk and scRNA-seq experiments.[26–28] Critically, *BASiCS* enables the quantification of transcriptional variability within a population of cells, while accounting for the overall mean-variance relationship that typically arises in scRNA-seq data.[29] Furthermore, when available, *BASiCS* can also leverage extrinsic spike-in molecules to aid data normalisation.

This article complements existing scRNA-seq workflows based on the Bioconductor ecosystem (e.g. Refs. 30, 31), providing a detailed framework for transcriptional variability analyses using *BASiCS*. We describe a step-by-step workflow that uses *scater*[19] and *scran*[30] to perform quality control (QC) as well as initial exploratory analyses. Our analysis pipeline includes practical guidance to assess the convergence of the Markov Chain Monte Carlo (MCMC) algorithm that is used to infer model parameters in *BASiCS*, as well as recommendations to interpret and post-process the model outputs. Finally, through a case study in the context of mouse immune cells, we illustrate how *BASiCS* can identify highly and lowly variable genes within a cell population, as well as to compare expression profiles between experimental conditions or cell types.

All source code used to generate the results presented in this article is available in Github and Zenodo.[32] To ensure the reproducibility of this workflow, the analysis environment and all software dependencies are provided as a Docker image.[33] The image can be obtained from Docker Hub.

Methods

Implementation

The *BASiCS* Bioconductor package uses a Bayesian hierarchical framework that borrows information across all genes and cells to robustly quantify transcriptional variability.[34] Similar to the approach adopted in *scran*, *BASiCS* infers cell-specific global scaling normalisation parameters. However, instead of inferring these as a pre-processing step, *BASiCS* uses an integrated approach wherein data normalisation and downstream analyses are performed simultaneously, thereby

propagating statistical uncertainty. To quantify technical noise, the original implementation of *BASiCS* uses information from extrinsic spike-in molecules as control features, but the model has been extended to address situations wherein spike-ins are not available.[29]

BASiCS summarises the expression pattern for each gene through gene-specific *mean* and *over-dispersion* parameters. Mean parameters μ_i quantify the overall expression for each gene i across the cell population under study. In contrast, δ_i captures the excess of variability that is observed with respect to what would be expected in a homogeneous cell population, beyond technical noise. *BASiCS* uses δ_i as a proxy to quantify transcriptional variability. To account for the strong relationship that is typically observed between gene-specific mean expression and over-dispersion estimates, Eling *et al.*[29] introduced a joint prior specification for these parameters. This joint prior has been observed to improve posterior inference when the data is less informative (e.g. small sample size, lowly expressed genes), borrowing information across all genes (and cells) to infer an overall trend that captures the relationship between mean and over-dispersion. The trend is then used to derive gene-specific *residual over-dispersion* parameters ε_i that are not confounded by mean expression. Similar to the DM values implemented in *scran*, these are defined as deviations with respect to the overall trend.

Within a population of cells, *BASiCS* decomposes the total observed variability in expression measurements into technical and biological components.[23] This enables the identification of *highly variable genes* (HVGs) that capture the major sources of heterogeneity within the analysed cells.[17] HVG detection is often used as feature selection, to identify the input set of genes for subsequent analyses. *BASiCS* can also highlight *lowly variable genes* (LVGs) that exhibit stable expression across the population of cells. These may relate to essential cellular functions and can assist the development of new data normalisation or integration strategies.[8]

BASiCS also provides a probabilistic decision rule to perform differential expression analyses between two pre-specified groups of cells.[24,29] While several differential expression tools have been proposed for scRNA-seq data (e.g. Refs. 35, 36), some evidence suggests that these do not generally outperform popular bulk RNA-seq tools.[37] Moreover, most of these methods are only designed to uncover changes in overall expression, ignoring the more complex patterns that can arise at the single cell level.[38] Instead, *BASiCS* embraces the high granularity of scRNA-seq data, uncovering changes in cell-to-cell expression variability that are not confounded by differences in technical noise or in overall expression.

Operation

This step-by-step scRNA-seq workflow is primarily based on the Bioconductor package ecosystem[39] for the R programming language,[40] and as such should run on any major operating system using R ≥ 4.0. A graphical overview is provided in Figure 1 and its main components are described below. The libraries listed below are required for this workflow, all of which can be downloaded from Bioconductor. Alternatively, we provide a Docker image containing all of the software necessary to run *BASiCS* at `https://hub.docker.com/r/alanocallaghan/bocker/`.

```
library("SingleCellExperiment")
library("scater")
library("scran")
library("BASiCS")
```

We use the package *SingleCellExperiment* to convert an input matrix of raw read counts (molecule counts for UMI-based protocols) into a `SingleCellExperiment` object that can also store its associated metadata, such as gene- and cell-specific information. Moreover, when available, the same object can also store read counts for spike-in molecules (see `help("altExp")`). A major advantage of using a `SingleCellExperiment` object as the input for scRNA-seq analyses is the interoperability across a large number of Bioconductor packages.[39]

A critical step in scRNA-seq analyses is QC, removing low quality samples that may distort downstream analyses. In this step, we use QC diagnostics to identify and remove samples that correspond to broken cells, that are empty, or that contain multiple cells.[41] We also typically remove lowly expressed genes that represent less reliable information. The *OSCA* online book provides an extensive overview on important aspects of how to perform QC of scRNA-seq data, including exploratory analyses.[39]

Here, we use the Bioconductor package *scater*[19] to calculate QC metrics for each cell (e.g. total read-count) and gene (e.g. percentage of zeroes across all cells), respectively. We also use the visualisation tools implemented in the *scater* to explore the input dataset and its associated QC diagnostic metrics. For further exploratory data analysis we use the Bioconductor package *scran*.[30] The latter can perform *global scaling* normalisation, calculating cell-specific scaling

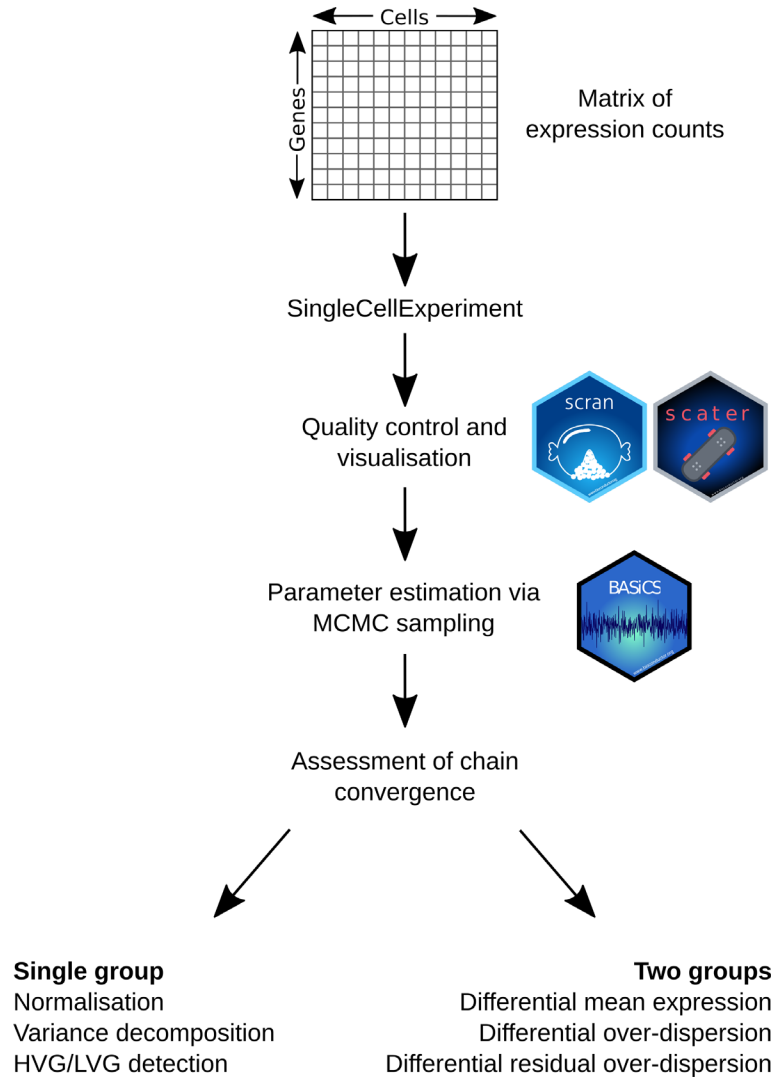


Figure 1. Graphical overview for the single cell RNA sequencing analysis workflow described in this manuscript. Starting from a matrix of expression counts, we use the scater and scran Bioconductor packages to perform QC and initial exploratory analyses. To robustly quantify transcriptional heterogeneity within seemingly homogeneous cell populations, we apply the BASiCS Bioconductor package and illustrate how BASiCS can be used to analyse a single or multiple pre-specified groups of cells.

factors that capture global differences in read-counts across cells (e.g. due to sequencing depth and PCR amplification).[42] To quantify transcriptional variability, *scran* can infer an overall trend between mean expression and the squared coefficent of variation (CV^2) for each gene. Variability estimates that are not confounded by this trend are then obtained via the DM approach.[43] For each gene, these are defined as the distance between CV^2 and a rolling median along the range of mean expression values. DM estimates enable exploratory analyses of transcriptional variability, but a measure of statistical uncertainty is not readily available. As such, gene-specific downstream inference (such as differential variability testing) is precluded.

Use cases

Case study: analysis of naive $CD4^+$ T cells

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.* 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.[7] 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 RNA[18] 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).

Downloading the data

The matrix with raw read counts can be obtained from ArrayExpress under the accession number E-MTAB-4888. In the matrix, column names contain library identifiers and row names display Ensembl gene identifiers.

```
if (!file.exists("downloads/"))
  dir.create("downloads", showWarnings = FALSE)
if (!file.exists("downloads/raw_data.txt")) {
  website <- "https://www.ebi.ac.uk/arrayexpress/files/E-MTAB-4888/"
  file <- "E-MTAB-4888.processed.1.zip"
  download.file(
     paste0(website, file),
     destfile = "downloads/raw_data.txt.zip"
  )
  unzip("downloads/raw_data.txt.zip", exdir = "downloads")
  file.remove("downloads/raw_data.txt.zip")
}
cd4_raw <- read.table("downloads/raw_data.txt", header = TRUE, sep = "\t")
cd4_raw <- as.matrix(cd4_raw)
```

The input matrix contains data for 1,513 cells and 31,181 genes, including 92 ERCC spike-ins.

Information about experimental conditions and other metadata is available under the same accession number.

```
if (!file.exists("downloads/metadata_file.txt")) {
  website <- "https://www.ebi.ac.uk/arrayexpress/files/E-MTAB-4888/"
  file <- "E-MTAB-4888.additional.1.zip"
  download.file(
     paste0(website, file),
     destfile = "downloads/metadata.txt.zip"
  )
  unzip("downloads/metadata.txt.zip", exdir = "downloads")
  file.remove("downloads/metadata.txt.zip")
}
cd4_metadata <- read.table(
  "downloads/metadata_file.txt",
  header = TRUE,
  sep = "\t"
)
## Save sample identifiers as rownames
rownames(cd4_metadata) <- cd4_metadata$X
```

The columns in the metadata file contain library identifiers (`X`), strain information (`Strain`; *Mus musculus castaneus* or *Mus musculus domesticus*), the age of the animals (`Age`; young or old), stimulation state of the cells (`Stimulus`; naive or activated), batch information (`Individuals`; associated to different mice), and cell type information (`Celltype`; via FACS or MACS purification).

Here, we convert the data and metadata described above into a `SingleCellExperiment` object. For this purpose, we first separate the input matrix of expression counts into two matrices associated to intrinsic genes and external spike-ins, respectively. Within the `SingleCellExperiment` object, the latter is stored separately as an *alternative experiment.* For more details on the alternative experiment slot, see `help("altExp")`.

```
## Separate intrinsic from ERCC counts
bio_counts <- cd4_raw[!grepl("ERCC", rownames (cd4_raw)),]
spike_counts <- cd4_raw[grepl("ERCC", rownames (cd4_raw)),]
## Generate the SingleCellExperiment object
sce_cd4_all <- SingleCellExperiment(
  assays = list(counts = bio_counts),
  colData = cd4_metadata[colnames (cd4_raw),]
)
## Add read-counts for spike-ins as an alternative experiment
altExp(sce_cd4_all, "spike-ins") <- SummarizedExperiment(
  assays = list(counts = spike_counts)
)
```

Hereafter, our analysis focuses on naive CD4$^+$ T cells in the presence and absence of stimulation using plate-bound antibodies,+ obtained from young *Mus musculus domesticus* animals, and purified using MACS-based cell sorting. Thus, we subset the `SingleCellExperiment` object to these 146 cells.

```
ind_select <- sce_cd4_all$Strain == "Mus musculus domesticus" &
  sce_cd4_all$Age == "Young" &
  sce_cd4_all$Celltype == "MACS-purified Naive"
sce_naive_active <- sce_cd4_all[, ind_select]
sce_naive_active
## class: SingleCellExperiment
## dim: 31089 146
## metadata(0):
## assays(1): counts
## rownames(31089): ENSMUSG00000000001 ENSMUSG00000000003 …
##   ENSMUSG00000106668 ENSMUSG00000106670
## rowData names(0):
## colnames(146): do6113 do6118 … do6493 do6495
## colData names(6): X Strain … Individuals Celltype
## reducedDimNames(0):
## mainExpName: NULL
## altExpNames(1): spike-ins
```

Annotation

Input data was annotated using Ensembl gene identifiers. To facilitate interpretation, it is often useful to obtain a mapping from Ensembl gene IDs to gene symbols using the BioMart suite (http://www.biomart.org) via the Bioconductor package *biomaRt*.[44] This can also be used to obtain gene-pathways mappings and other metadata (e.g. gene length), useful for performing functional analysis of gene sets identified in downstream analyses.

```
library("biomaRt")
if (!dir.exists("rds")) {
  dir.create("rds", showWarnings = FALSE)
}
if (!file.exists("rds/genenames.rds")) {
  # Initialize mart and dataset
  ensembl <- useEnsembl(
      biomart = "genes",
      version = 104,
      dataset = "mmusculus_gene_ensembl"
  )
  # Select gene ID and gene name
  genenames <- getBM(
      attributes = c("ensembl_gene_id", "external_gene_name", "gene_biotype"),
      mart = ensembl
  )
  rownames(genenames) <- genenames$ensembl_gene_id
  saveRDS(genenames, "rds/genenames.rds")
}
genenames <- readRDS("rds/genenames.rds")
```

We add this information as `rowData` within the `SingleCellExperiment` object created above.

```
## Merge biomaRt annotation
rowdata <- data.frame(ensembl_gene_id = rownames(sce_naive_active))
rowdata <- merge(rowdata, genenames, by = "ensembl_gene_id", all.x = TRUE)
rownames(rowdata) <- rowdata$ensembl_gene_id
## Check if order is correct after merge;
stopifnot(all(rownames(rowdata) == rownames(sce_naive_active)))
## add to the SingleCellExperiment object
rowData(sce_naive_active) <- rowdata
```

For the remaining analysis, we will only focus on the 18,682 protein coding genes that are contained in the data. These are selected below.

```
protein_coding <- which(
  rowData(sce_naive_active)$gene_biotype == "protein_coding"
)
sce_naive_active <- sce_naive_active[protein_coding,]
```

QC and exploratory data analysis

The data available at E-MTAB-4888 have been filtered already to remove poor quality samples. The QC applied in Ref. 7 removed cells with: (i) fewer than 1,000,000 total reads, (ii) less than 20% of reads mapped to endogenous genes, (iii) less than 1,250 or more than 3,000 detected genes and (iv) more than 10% or fewer than 0.5% of reads mapped to mitochondrial genes. We include visualisations of these measures here; we also include another widely used QC diagnostic plot that compares the total number (or fraction) of spike-in counts versus the total number (or fraction) of endogeneous counts. In such a plot, low quality samples are characterised by a high fraction of spike-in counts and a low fraction of endogeneous counts (see Figure 2).

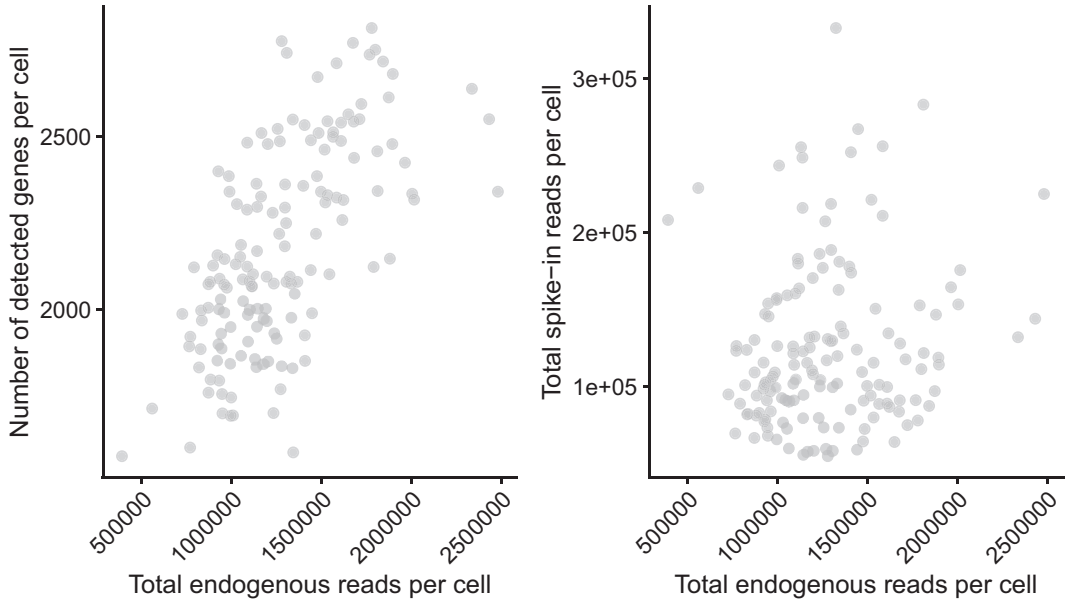


Figure 2. Cell-level quality control metrics. The total number of endogenous read-counts (excludes non-mapped and intronic reads) is plotted against the total number of detected genes (left) and the total number of spike-in read-counts (right).

```
sce_naive_active <- addPerCellQC (sce_naive_active, use_altexps = TRUE)
p_cell_qc1 <- plotColData(
  sce_naive_active,
  x = "sum",
  y = "detected") +
  xlab("Total endogenous reads per cell") +
  ylab("Number of detected genes per cell") +
  theme(axis.text.x = element_text(hjust = 1, angle = 45))
p_cell_qc2 <- plotColData(
  sce_naive_active,
  x = "sum",
  y = "altexps_spike-ins_sum") +
  xlab("Total endogenous reads per cell") +
  ylab("Total spike-in reads per cell") +
  theme(axis.text.x = element_text(hjust = 1, angle = 45))
library("patchwork")
p_cell_qc1 + p_cell_qc2
```

We can also visualise these metrics with respect to cell-level metadata, such as the experimental conditions (active vs unstimulated) and the different mice from which cells were collected (see Figure 3).

```
p_stimulus <- plotColData(
     sce_naive_active,
     x = "sum",
     y = "detected",
     colour_by = "Stimulus"
  ) +
  xlab("Total endogenous reads per cell") +
  ylab("Number of detected genes per cell") +
  theme(
     legend.position = "bottom",
     axis.text.x = element_text(angle = 45, hjust = 1)
  )
p_batch <- plotColData(
     sce_naive_active,
     x = "sum",
     y = "detected",
     colour_by = "Individuals"
  ) +
  xlab("Total endogenous reads per cell") +
  ylab("Number of detected genes per cell") +
  theme(
     legend.position = "bottom",
     axis.text.x = element_text(angle = 45, hjust = 1)
  )
p_stimulus + p_batch
```

To further explore the underlying structure of the data, we perform global scaling normalisation using *scran* and principal component analysis (PCA) of log-transformed normalised expression counts using *scater*. As seen in Figure 4, this analysis suggests the absence of strong batch effects. It should be noted that the estimation of global scaling normalisation factors using *scran* is not strictly necessary in the *BASiCS* workflow. Here, we only use it as part of the exploratory data analysis. Moreover, count-based models for dimensionality reduction (e.g. Refs. 28, 45) could be used as an alternative to PCA, removing the need for log normalisation.

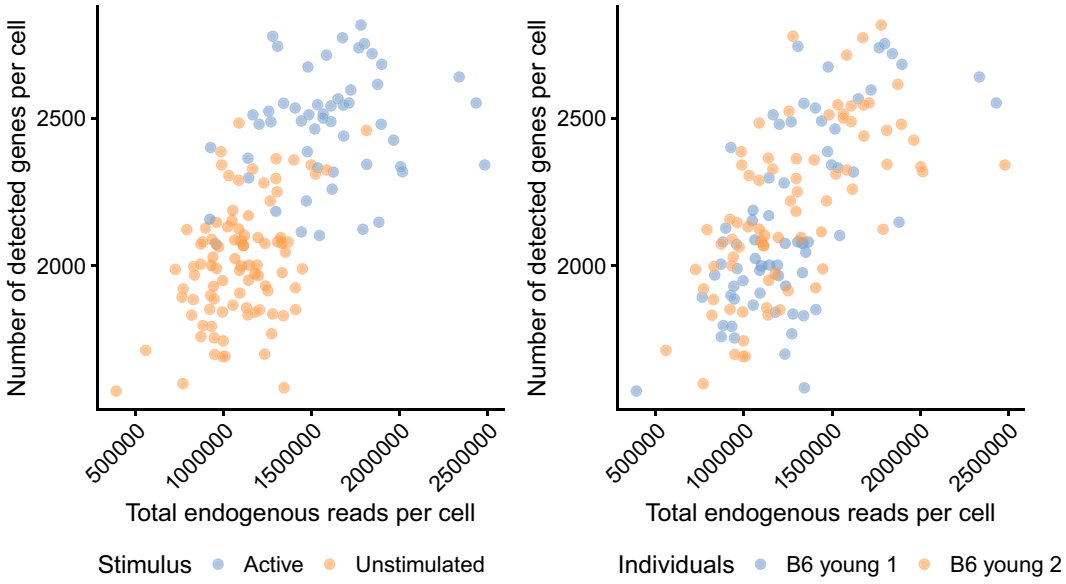


Figure 3. Cell-level quality control metrics according to cell-level metadata. The total number of endogenous reads (excludes non-mapped and intronic reads) is plotted against the total number of detected genes. Colour indicates the experimental condition (left) and animal of origin (right) for each cell.

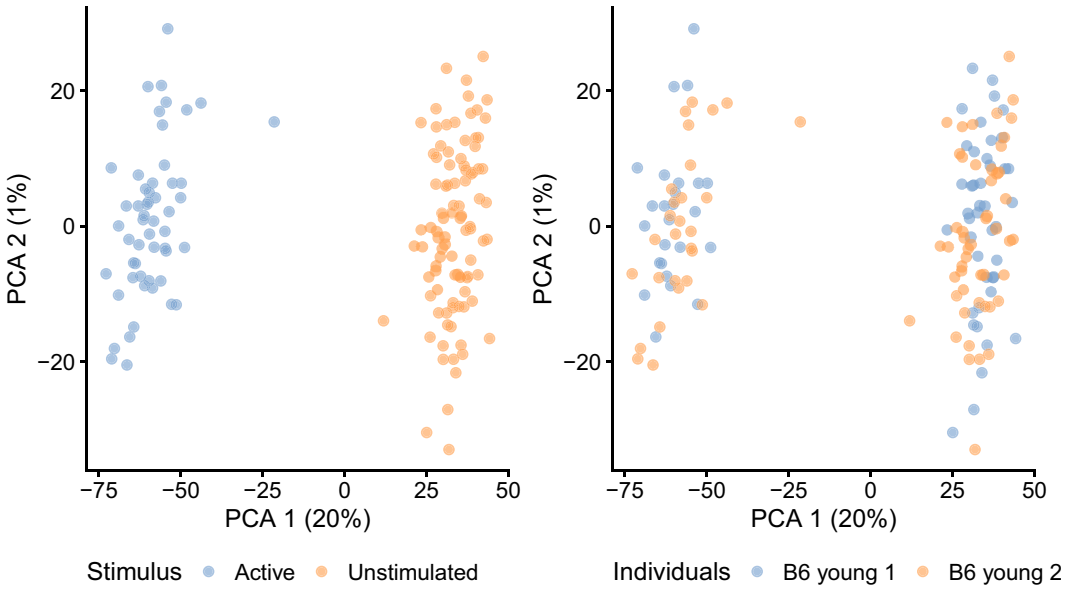


Figure 4. First two principal components of log-transformed expression counts after scran normalisation. Colour indicates the experimental condition (left) and animal of origin (right) for each cell.

```
## Global scaling normalisation + log tranformation + PCA
sce_naive_active <- computeSumFactors(sce_naive_active)
sce_naive_active <- logNormCounts(sce_naive_active)
sce_naive_active <- runPCA(sce_naive_active)
p_stimulus <- plotPCA(sce_naive_active, colour_by = "Stimulus") +
  theme(legend.position = "bottom")
p_batch <- plotPCA(sce_naive_active, colour_by = "Individuals") +
  theme(legend.position = "bottom")
p_stimulus + p_batch
```

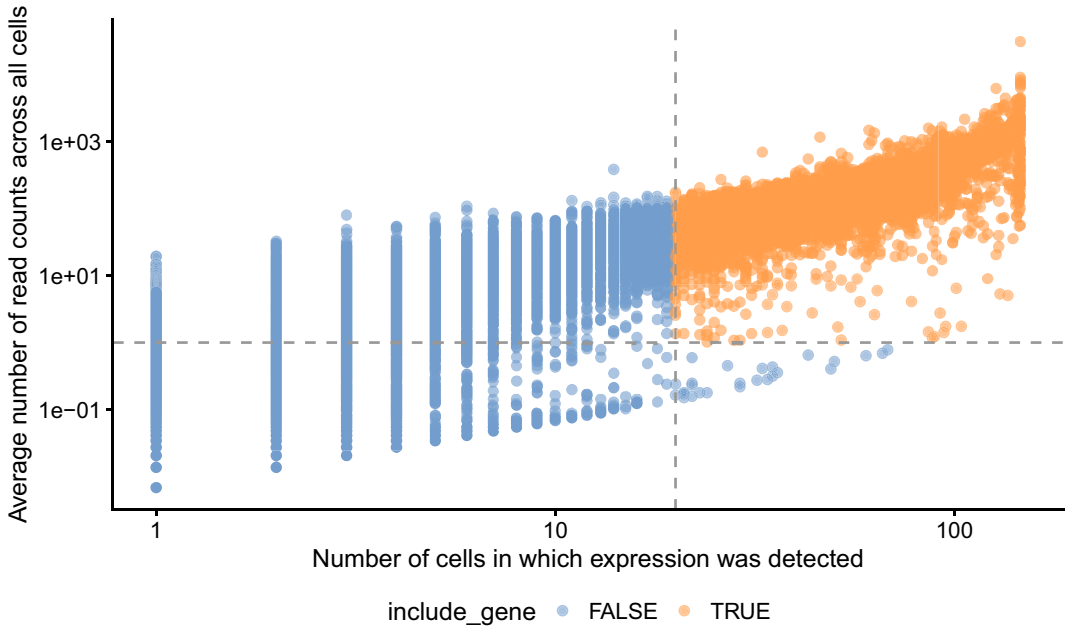


Figure 5. Average read-count for each gene is plotted against the number of cells in which that gene was detected. Dashed grey lines are shown at the thresholds below which genes are removed.

In addition to cell-specific QC, we also recommend a gene filtering step prior to using *BASiCS*. The purpose of this filter is to remove lowly expressed genes that were largely undetected through sequencing, making reliable variability estimates difficult to obtain. Here, we remove genes that are not detected in at least 20 cells across both conditions, or that have an average read count below 1. These thresholds can vary across datasets and should be informed by gene-specific QC metrics such as those shown in Figure 5 as well as prior knowledge about the cell types and conditions being studied, where available.

```
sce_naive_active <- addPerFeatureQC(sce_naive_active, exprs_values = "counts")
## Remove genes with zero total counts across all cells
sce_naive_active <- sce_naive_active[rowData(sce_naive_active)$detected != 0,]
## Transform "detected" into number of cells and define inclusion criteria
rowData(sce_naive_active)$detected_cells <-
  rowData(sce_naive_active)$detected * ncol(sce_naive_active) / 100
detected_threshold <- 20
mean_threshold <- 1
include_gene <- rowData(sce_naive_active)$mean >= mean_threshold &
  rowData(sce_naive_active)$detected_cells >= detected_threshold
rowData(sce_naive_active)$include_gene <- include_gene
plotRowData(
      sce_naive_active,
      x = "detected_cells",
      y = "mean",
      colour_by = "include_gene"
  ) +
  xlab("Number of cells in which expression was detected") +
  ylab("Average number of read counts across all cells") +
  scale_x_log10() +
  scale_y_log10() +
  theme(legend.position = "bottom") +
  geom_vline(
      xintercept = detected_threshold,
      linetype = "dashed",
      col = "grey60"
  ) +
```

```
  geom_hline(
      yintercept = mean_threshold,
      linetype = "dashed",
      col = "grey60"
  )

## Apply gene filter
sce_naive_active <- sce_naive_active[rowData(sce_naive_active)$include_gene,]
```

Subsequently, we also require users to remove spike-in molecules that were not captured through sequencing. We do this separately for naive and active cells.

```
ind_active <- sce_naive_active$Stimulus == "Active"
ind_naive <- sce_naive_active$Stimulus == "Unstimulated"
spikes <- assay(altExp(sce_naive_active))
detected_spikes_active <- rowSums(spikes[, ind_active] > 0) > 0
detected_spikes_naive <- rowSums(spikes[, ind_naive] > 0) > 0
detected_spikes <- detected_spikes_naive & detected_spikes_active
altExp(sce_naive_active) <- altExp(sce_naive_active)[detected_spikes,]
```

The final dataset used in subsequent analyses contains 146 cells, 5171 genes and 49 spike-ins.

Input data for BASiCS

Here, we apply *BASiCS* separately to cells from each experimental condition (93 naive and 53 activated cells). We create separate `SingleCellExperiment` objects for each group of cells.

```
sce_naive <- sce_naive_active[, ind_naive]
sce_active <- sce_naive_active[, ind_active]
```

BASiCS requires these objects to be augmented with extra information in a specific format. If multiple batches of sequenced cells are available (e.g. multiple donors from which cells were extracted, sequencing batches from the same experimental condition), this must be indicated under the `BatchInfo` label as cell-level metadata.

```
colData(sce_naive)$BatchInfo <- colData(sce_naive)$Individuals
colData(sce_active)$BatchInfo <- colData(sce_active)$Individuals
```

If spike-ins will be used to aid data normalisation and technical noise quantification, *BASiCS* also requires the number of spike-in molecules that were added to each well. For each spike-in i, this corresponds to:

$$\mu_i = C_i \times 10^{-18} \times \left(6.022 \times 10^{23}\right) \times V \times D \quad \text{where,}$$

- C_i is the concentration for the spike-in i (measured in $aM\mu l^{-1}$),
- V is the volume added into each well (measure in nl) and
- D is a dilution factor.

The remaining factors in the equation above are unit conversion constants (e.g. from moles to molecules). For the CD4+ T cell data, the authors added a 1:50,000 dilution of the ERCC spike-in mix 1 and a volume of $9nl$ was added into each well. Finally, input concentrations C_i can be downloaded from https://assets.thermofisher.com/TFS-Assets/LSG/manuals.

```
if (!file.exists("downloads/spike_info.txt")) {
  website <- "https://assets.thermofisher.com/TFS-Assets/LSG/manuals/"
  file <- "cms_095046.txt"
  download.file(
      paste0(website, file),
```

```
    destfile = "downloads/spike_info.txt"
  )
}
ERCC_conc <- read.table("downloads/spike_info.txt", sep = "\t", header = TRUE)
```

Based on this information, the calculation above proceeds as follows

```
## Moles per micro litre
ERCC_mmul <- ERCC_conc$concentration.in. Mix.1.attomoles.ul. * (10^(-18))
## Molecule count per microL (1 mole comprises 6.02214076 x 10^{23} molecules)
ERCC_countmul <- ERCC_mmul * (6.02214076 * (10^23))
## Application of the dilution factor (1:50,000)
ERCC_count <- ERCC_countmul / 50000
## Multiplying by the volume added into each well
ERCC_count_final <- ERCC_count * 0.009
```

To update the `sce_naive` and `sce_active` objects, the user must create a `data.frame` whose first column contains the spike-in labels (e.g. ERCC-00130) and whose second column contains the number of molecules calculated above. We add this as row metadata for `altExp (sce_naive)` and `altExp (sce_active)`.

```
SpikeInput <- data.frame(
  Names = ERCC_conc$ERCC.ID,
  count = ERCC_count_final
)
## Exclude spike-ins not included in the input SingleCellExperiment objects
SpikeInput <- SpikeInput[match(rownames(altExp(sce_naive)), SpikeInput$Names),]
## Add as metadata
rowData(altExp(sce_naive)) <- SpikeInput
rowData(altExp(sce_active)) <- SpikeInput
```

Parameter estimation using BASiCS

Parameter estimation is implemented in the `BASiCS_MCMC` function using an adaptive Metropolis within Gibbs algorithm (see section 3 in Ref. 46). The primary inputs for `BASiCS_MCMC` correspond to:

- `Data`: a `SingleCellExperiment` object created as described in the previous sections.
- `N`: the total number of MCMC iterations.
- `Thin`: thining period for output storage (only the `Thin`-th MCMC draw is stored).
- `Burn`: the initial number of MCMC iterations to be discarded.
- `Regression`: if `TRUE` a joint prior is assigned to μ_i and δ_i,[29] and residual over-dispersion values ε_i are inferred. Alternatively, independent log-normal priors are assigned to μ_i and δ_i.[24]
- `WithSpikes`: if `TRUE` information from spike-in molecules is used to aid data normalisation and to quantify technical noise.
- `PriorParam`: Defines the prior hyper-parameters to be used by *BASiCS*. We recommend to use the `BASiCS_PriorParam` function for this purpose. If `PriorMu = "EmpiricalBayes"`, μ_i's are assigned a log-normal prior with gene-specific location hyper-parameters defined via an empirical Bayes framework. Alternatively, if `PriorMu = "default"`, location hyper-parameters are set to be equal 0 for all genes.

As a default, we recommend to use `Regression = TRUE`, as the joint prior introduced by Ref. 29 leads to more stable estimation, particularly for small sample sizes and lowly expressed genes. This approach also enables users to obtain a

measure of transcriptional variability that is not confounded by mean expression. We also recommend to use `PriorMu = "EmpiricalBayes"` as we have observed that an empirical Bayes framework[47] improves estimation performance for sparser datasets. Extra parameters can be used to store the output (`StoreChains`, `StoreDir`, `RunName`) and to monitor the progress of the algorithm (`PrintProgress`).

Here, we run the MCMC sampler separately for naive and activated cells. We use 40,000 iterations (`N`), discarding the initial 20,000 iterations (`Burn`), and saving parameter values only once in each 20 iterations (`Thin`). We recommend this setting as a default choice, as we have observed it to ensure good convergence across multiple datasets. However, fewer iterations may be sufficient for larger and less sparse datasets, and may be more feasible computationally for larger datasets. Practical guidance about MCMC convergence diagnostics is provided in the next section.

```
## MCMC results may vary slightly due to pseudorandom number generation.
## We fix a random seed for exact reproducibility,
## but this is not strictly required
set.seed(42)
chain_naive <- BASiCS_MCMC(
  Data = sce_naive,
  N = 40000,
  Thin = 20,
  Burn = 20000,
  Regression = TRUE,
  WithSpikes = TRUE,
  PriorParam = BASiCS_PriorParam(sce_naive, PriorMu = "EmpiricalBayes"),
  Threads = 4,
  StoreChains = TRUE,
  StoreDir = "rds/",
  RunName = "naive"
)
set.seed(43)
chain_active <- BASiCS_MCMC(
  Data = sce_active,
  N = 40000,
  Thin = 20,
  Burn = 20000,
  Regression = TRUE,
  WithSpikes = TRUE,
  PriorParam = BASiCS_PriorParam(sce_active, PriorMu = "EmpiricalBayes"),
  Threads = 4,
  StoreChains = TRUE,
  StoreDir = "rds/",
  RunName = "active"
)
```

This first of these samplers takes 84 minutes to complete on a 3.4 GHz Intel Core i7 4770k procesor with 32GB RAM, while the second takes 69 minutes. For convenience, these can be obtained online at https://doi.org/10.5281/zenodo.5243265.

```
chains_website <- "https://zenodo.org/record/5243265/files/"
options(timeout = 1000)
if (!file.exists("rds/chain_naive.Rds")) {
  download.file(
      paste0(chains_website, "chain_naive.Rds"),
      destfile = "rds/chain_naive.Rds"
  )
}
if (!file.exists("rds/chain_active.Rds")) {
  download.file(
      paste0(chains_website, "chain_active.Rds"),
```

```
    destfile = "rds/chain_active.Rds"
  )
}
chain_naive <- readRDS("rds/chain_naive.Rds")
chain_active <- readRDS("rds/chain_active.Rds")
```

The output from `BASiCS_MCMC` is a `BASiCS_Chain` object that contains the draws associated to all model parameters. Given that `N` = 40,000, `Thin` = 20 and `Burn` = 20,000, the object contains 1,000 draws for each parameter. These can be accessed using the `displayChainBASiCS` function. For example, the following code displays the first 6 draws for mean expression parameters μ_i associated to the first 3 genes.

```
displayChainBASiCS (chain_naive, Param = "mu")[1:2, 1:3]
##       ENSMUSG00000000001 ENSMUSG00000000088 ENSMUSG00000000131
## [1,]           26.19096           1.759170           35.60811
## [2,]           12.05079           2.564187           35.58258
```

MCMC diagnostics

Before interpreting the estimates generated by *BASiCS*, it is critical to assess the convergence of the MCMC algorithm, i.e. whether the MCMC reached its stationary distribution. If convergence has been achieved, the trace for each parameter should not evolve significantly across iterations, as MCMC draws are expected to be stochastic fluctuations around a horizontal trend once the sampler has converged to its stationary distribution. It is not possible to prove convergence, but multiple graphical and quantitative convergence diagnostics have been proposed to assess the lack of convergence (e.g. Refs. 48, 49). Some advocate the use of multiple MCMC chains using different starting values in order to ensure that the algorithm consistently converges to the same distribution and to avoid convergence to local modes. For *BASiCS*, we have observed that using informed starting values (e.g. based on *scran* normalisation factors) and a sufficiently large value for `N` and `Burn` generally leads to largely consistent estimates across multiple MCMC runs. Hence, the focus of this section is to evaluate quantitative measures of convergence (e.g. Ref. 50) based on a single MCMC chain.

Traceplots can be used to visually assess the history of MCMC iterations for a specific parameter (e.g. Figure 6, left panel). As mentioned above, significant departures from a horizontal trend suggest a lack of convergence. As illustrated in

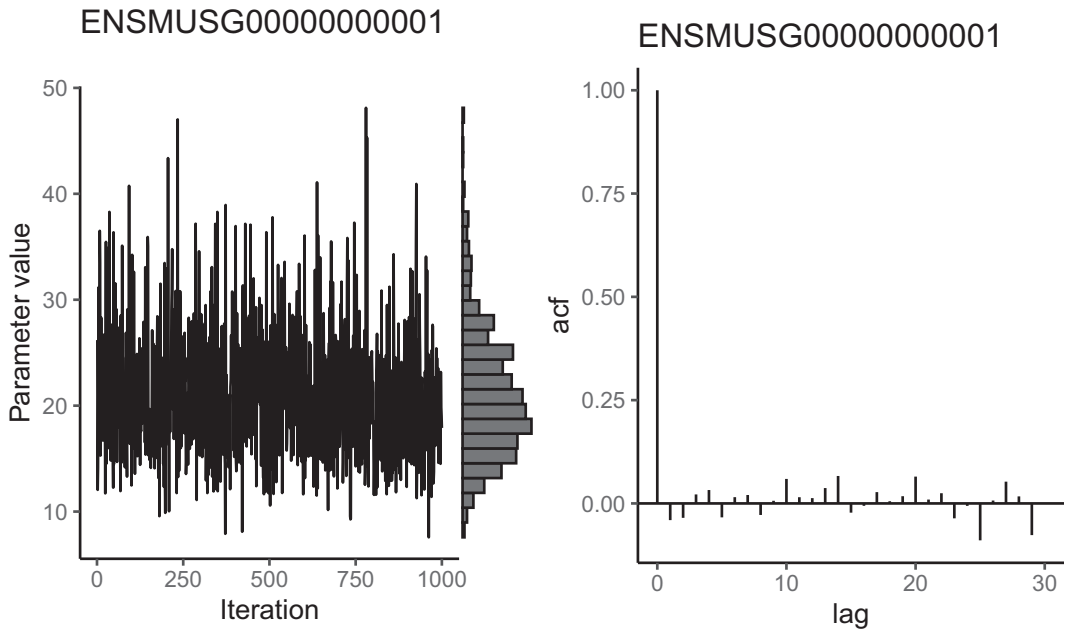


Figure 6. Trace plot, marginal histogram, and autocorrelation function of the posterior distribution of the mean expression parameter for a gene in naive cells following Markov chain Monte Carlo sampling. Trace plots should explore the posterior without getting stuck in one location or drifting over time towards a region of higher density. High autocorrelation indicates that the number of effective independent samples is low. It is good practice to perform this visualisation for many different parameters; here we only show one.

Figure 6, histograms can also be used to display the marginal distribution for each parameter. For *BASiCS*, users should expect these to follow a unimodal distribution. Failure to satisfy these graphical diagnostics suggest that `N` and `Burn` must be increased. Alternatively, more stringent quality control could be applied to the input data, as we have observed that genes with very low counts often suffer from slow convergence.

```
plot (chain_naive, Param = "mu", Gene = 1)
```

As *BASiCS* infers thousands of parameters, it is impractical to assess these diagnostics separately for each parameter. Thus, it is helpful to use numerical diagnostics that can be applied to multiple parameters simultaneously. Here, we illustrate usage for two such metrics focusing on the MCMC chain that was obtained for the naive CD4$^+$T cells (similar results were obtained for activated cells). First, we focus on the diagnostic criterion proposed by Geweke.[50] The latter compares the average of draws obtained during the initial (10% after burn in, by default) and the final part of the chain (50% by default) by calculating Z-scores of the relative difference between the two sets of draws. Large absolute Z-scores suggest that the algorithm has not converged (as a rule of thumb, a threshold at $|Z| < 3$ is often applied). For the naive and activated CD4$^+$ T datasets most Z-scores associated to mean expression parameters μ_i were small in absolute value (see Figure 7A), suggesting that the algorithm has largely converged.

As well as assessing MCMC convergence, it is important to ensure that the MCMC algorithm has efficiently explored the parameter space. For example, the autocorrelation function (e.g. Figure 6, right panel) quantifies the correlation between the chain and its lagged versions. Strong autocorrelation indicates that neighbouring MCMC samples are highly dependent and suggest poor sampling efficiency. The latter may indicate that the MCMC draws do not contain sufficient information to produce accurate posterior estimates. In other words, highly correlated MCMC samplers require more samples to produce the same level of Monte Carlo error for an estimate (defined as the variance of a Monte Carlo estimate across repetitions[51]).

The effective sample size (ESS) is a related measure which represents a proxy for the number of independent draws generated by the MCMC sampler.[52] The latter is defined as:

$$ESS = \frac{N_{tot}}{1 + 2\sum_{k=1}^{\infty} \rho(k)},$$

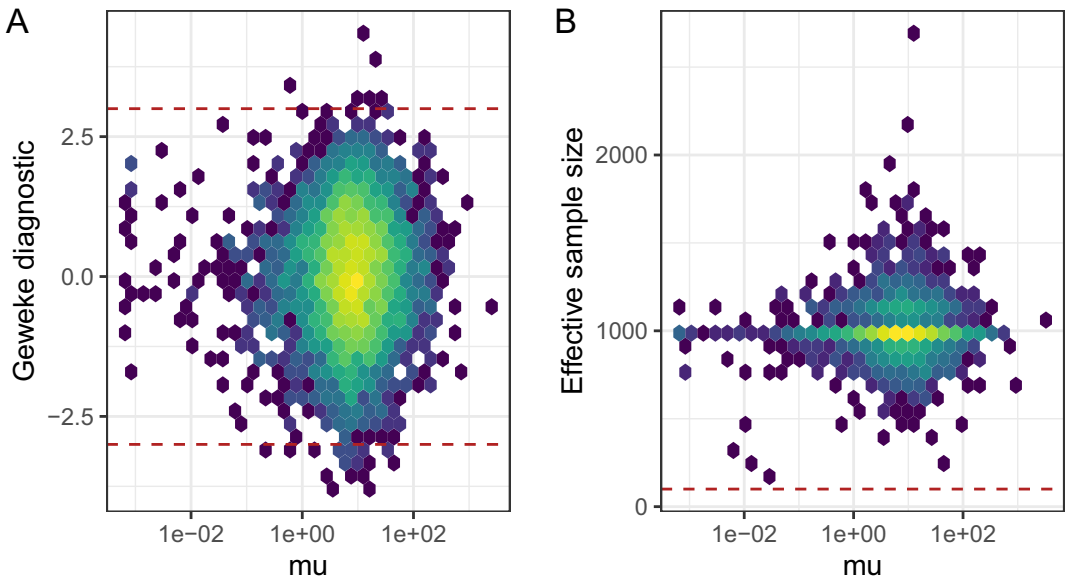


Figure 7. Markov chain Monte Carlo diagnostics for gene-specific mean expression parameters; naive CD4$^+$ T cells. A: Geweke Z-score for mean expression parameters is plotted against mean expression estimates. Dashed lines represent absolute Z-scores of 3, outside of which we advise caution when interpreting results. B: Effective sample size (ESS) is plotted against mean expression estimates. A dashed line shows a threshold of 100, below which we advise caution when interpreting results.

where N_{tot} represents the total number of MCMC draws (after burn-in and thining) and $\rho(k)$ is the autocorrelation at lag k. ESS estimates associated to mean expression parameters for the naive CD4$^+$ T cells are displayed in Figure 7B. Whilst ESS is around 1,000 (N_{tot} in this case) for most genes, we observe low ESS values for a small proportion of genes (primarily lowly expressed genes whose expression was only captured in a small number of cells). As described later in this manuscript, `BASiCS_TestDE` automatically excludes genes with low ESS during differential expression testing (by default a threshold at ESS < 100 is applied). However, if a large number of genes have large Geweke diagnostic values or low effective sample sizes in a certain dataset, then caution should be applied when interpreting the results of the model. These issues can often be addressed by more stringent filtering of genes and cells before performing inference or by increasing the number of iterations.

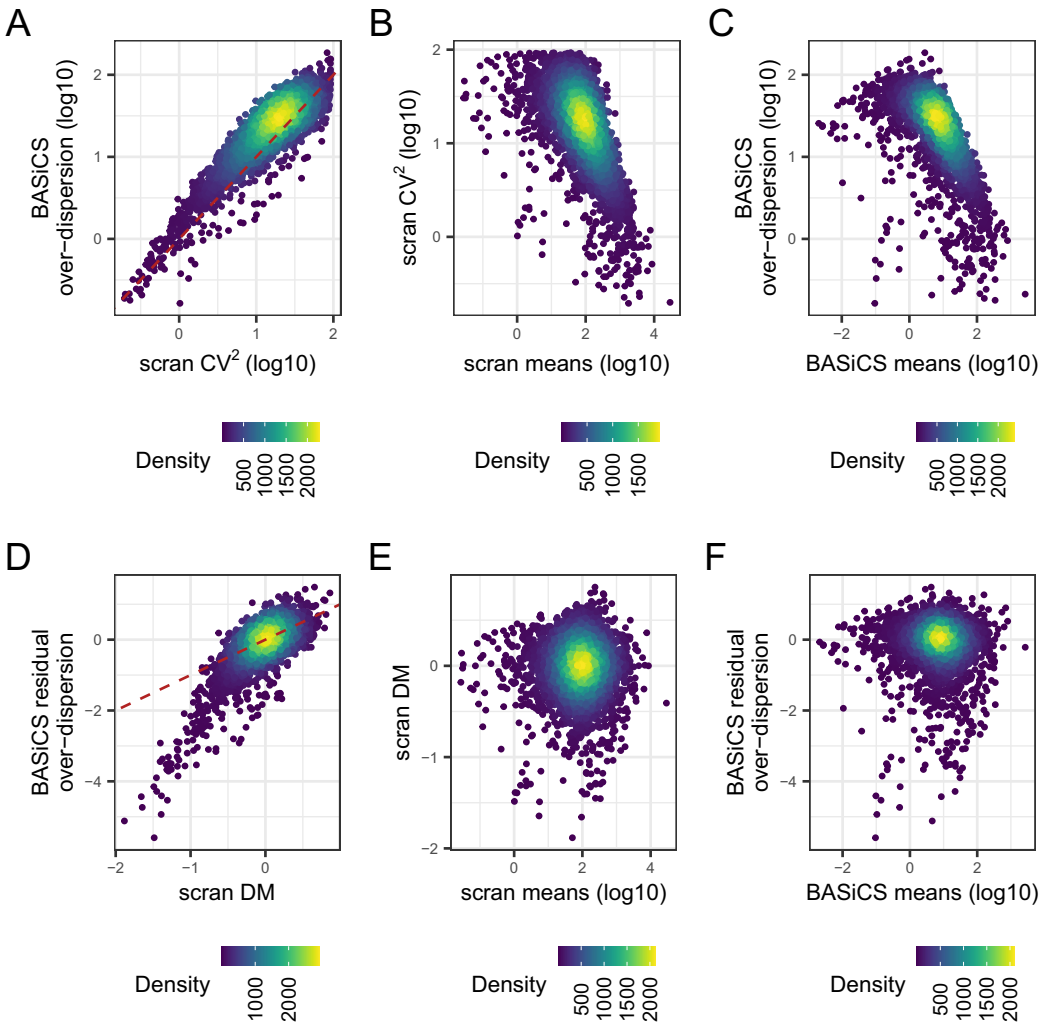


Figure 8. Comparison of gene-specific transcriptional variability estimates and mean expression estimates obtained for each gene using BASiCS and scran. For this analysis, we exclude genes that are not expressed in at least two cells. BASiCS estimates for each gene are defined by the posterior median of the associated parameter. scran estimates for each gene are derived after applying the pooling normalisation strategy proposed by Lun *et al.* Points are coloured according to the local density of genes along the x- and y-axis. A: scran squared CV estimates versus BASiCS estimates for over-dispersion parameters. B: scran estimates for mean expression and the squared CV. C: BASiCS estimates for mean expression and over-dispersion parameters. D: BASiCS estimates for residual over-dispersion parameters versus distance-to-median (DM) values estimated by scran. E: scran estimates for mean expression and DM values. F: BASiCS estimates for mean expression and residual over-dispersion parameters. Dashed red lines in panels A and D represent the line given by x = y.

```
library("coda")
library("ggplot2")
library("viridis")
diag_p1 <- BASiCS_DiagPlot(chain_naive, Measure = "geweke")
diag_p1 <- diag_p1 +
  geom_hline(yintercept = c(-3, 3), col = "firebrick", linetype = "dashed") +
  theme(legend.position = "none")
diag_p2 <- BASiCS_DiagPlot(chain_naive, Measure = "ess")
diag_p2 <- diag_p2 +
  geom_hline(yintercept = 100, col = "firebrick", linetype = "dashed") +
  theme(legend.position = "none")
diag_p1 + diag_p2 + plot_annotation(tag_levels = "A")
```

Quantifying transcriptional variability using BASiCS

Studying gene-level transcriptional variability can provide insights about the regulation of gene expression, and how it relates to the properties of genomic features (e.g. CpG island composition[16]), transcriptional dynamics[53] and aging,[7] among others. The squared coefficient of variation (CV^2) is widely used as a proxy for transcriptional variability. For example, we can obtain CV^2 estimates for each gene using *scran* normalised counts as input. In contrast, *BASiCS* infers transcriptional variability using gene-specific over-dispersion parameters δ_i (see *Methods*). Here, we compare these approaches, focusing on naive $CD4^+$ T cells (repeating this analysis for active cells led to similar results).

As seen in Figure 8A, CV^2 and posterior estimates for δ_i are highly correlated. Moreover, both variability metrics are confounded by differences in mean expression, i.e. highly expressed genes tend to exhibit lower variability (Figure 8B–C). To remove this confounding, *scran* and *BASiCS* derive *residual variability* estimates as deviations with respect to an global mean-variability trend (see *Methods*). These are derived using the DM approach[43] and the residual over-dispersion parameters ε_i defined by Ref. 29, respectively. For the naive $CD4^+$ T cell data, both approaches led to strongly correlated estimates (Figure 8D) and, as expected, neither DM values nor posterior estimates for ε_i are seen to be associated with mean expression (Figure 8E–F). However, unlike the DM method, the integrated approach implemented in *BASiCS* provides a direct measure of statistical uncertainty for these estimates via posterior variance. Note that the `BASiCS_ShowFit` function can be used to generate Figure 8C, but we generated the plot manually to demonstrate how users can extract this information from a `BASiCS_MCMC` object, and for visual consistency with the other panels. For each of the panels in Figure 8, we use the R package *ggpointdensity* to visualise the local density of genes along the axes of mean and variability.

```
library("ggpointdensity")
library("viridis")
## Get BASiCS posterior estimates for mean and variability - naive cells
summary_naive <- Summary(chain_naive)
parameter_df <- data.frame(
  mu = displaySummaryBASiCS(summary_naive, Param = "mu")[, 1],
  delta = displaySummaryBASiCS(summary_naive, Param = "delta")[, 1],
  epsilon = displaySummaryBASiCS(summary_naive, Param = "epsilon")[, 1]
)
## Get scran estimates for mean and variability - naive cells
sce_naive <- logNormCounts(sce_naive, log = FALSE)
parameter_df$mean_scran <- rowMeans(assay(sce_naive, "normcounts"))
parameter_df$cv2_scran <- rowVars(assay(sce_naive, "normcounts")) /
  parameter_df$mean_scran^2
parameter_df$DM <- DM(
  mean = parameter_df$mean_scran,
  cv2 = parameter_df$cv2_scran
)
## Remove genes without counts in > 2 cells - BASiCS estimates not provided
ind_not_na <- !(is.na(parameter_df$epsilon))
plot_params <- list(
  geom_pointdensity(size = 0.6),
  scale_colour_viridis(name = "Density"),
  theme(
```

```
      text = element_text(size = rel(3)),
      legend.position = "bottom",
      legend.text = element_text(angle = 90, size = 8, hjust = 0.5, vjust = 0.5),
      legend.key.size = unit(0.018, "npc")
  )
)
g1 <- ggplot(parameter_df[ind_not_na,], aes (log10(cv2_scran), log10(delta))) +
  plot_params +
  xlab(bquote("scran"~CV^2~"(log10)")) +
  ylab("BASiCS\nover-dispersion (log10)") +
  geom_abline(
      slope = 1,
      intercept = 0,
      colour = "firebrick",
      linetype = "dashed"
  )
g2 <- ggplot(parameter_df[ind_not_na,],
      aes(log10(mean_scran), log10(cv2_scran))
  ) +
  plot_params +
  xlab("scran means (log10)") +
  ylab(bquote("scran"~CV^2~"(log10)"))
g3 <- ggplot (parameter_df[ind_not_na,], aes (log10(mu), log10(delta))) +
  plot_params +
  xlab("BASiCS means (log10)") +
  ylab("BASiCS\nover-dispersion (log10)")
g4 <- ggplot (parameter_df[ind_not_na,], aes (DM, epsilon)) +
  plot_params +
  xlab("scran DM") +
  ylab("BASiCS residual\nover-dispersion") +
  geom_abline(
      slope = 1,
      intercept = 0,
      colour = "firebrick",
      linetype = "dashed"
  )
g5 <- ggplot(parameter_df[ind_not_na,], aes (log10(mean_scran), DM)) +
  plot_params +
  xlab("scran means (log10)") +
  ylab("scran DM")
g6 <- ggplot(parameter_df[ind_not_na,], aes (log10(mu), epsilon)) +
  plot_params +
  xlab("BASiCS means (log10)") +
  ylab("BASiCS residual\nover-dispersion")
(g1 + g2 + g3) / (g4 + g5 + g6) +
  plot_annotation(tag_levels = "A") & theme(plot.tag = element_text(size = 15))
```

HVG/LVG detection using BASiCS

In *BASiCS*, the functions `BASiCS_DetectHVG` and `BASiCS_DetectLVG` can be used to identify genes with substantially high (HVG) or low (LVG) transcriptional variability within a population of cells. If the input `BASiCS_Chain` object was generated by `BASiCS_MCMC` with `Regression = TRUE` (recommended setting), this analysis is based on the posterior distribution obtained for gene-specific residual over-dispersion parameters ε_i (alternatively, the approach introduced by[23] can be used). HVGs are marked as those for which ε_i exceeds a pre-defined threshold with high probability, where the probability cut-off is chosen to match a given expected false discovery rate (EFDR; by default the target EFDR is set to 10%).[54] A similar approach is implemented for LVG detection, but based on whether ε_i is below a pre-specified threshold `EpsilonThreshold`. For example, if the threshold for ε_i is equal to $\log(2)$ (the default), HVGs would be those genes for which the over-dispersion is estimated to be at least two times higher than would be expected given the inferred mean expression level, while LVGs would be those genes for which the residual

over-dispersion is at least two times lower than would be expected on the same basis. In some circumstances, it may be of interest to rank genes and to select those with the highest or the lowest residual over-dispersion, which can be performed using the `PercentileThreshold` parameter; see `help("BASiCS_DetectVG")` for more details.

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

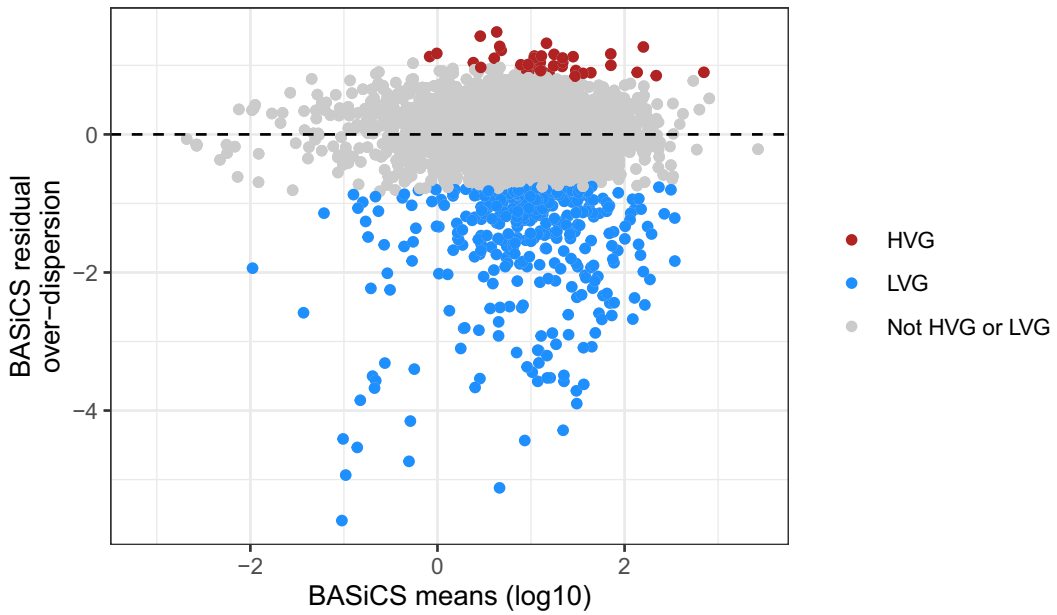


Figure 9. Highly-variable gene (HVG) and lowly-variable gene (LVG) detection using BASiCS. For each gene, BASiCS posterior estimates (posterior medians) associated to mean expression and residual over-dispersion parameters are plotted. Genes are coloured according to HVG/LVG status. Genes that are not expressed in at least two cells are excluded.

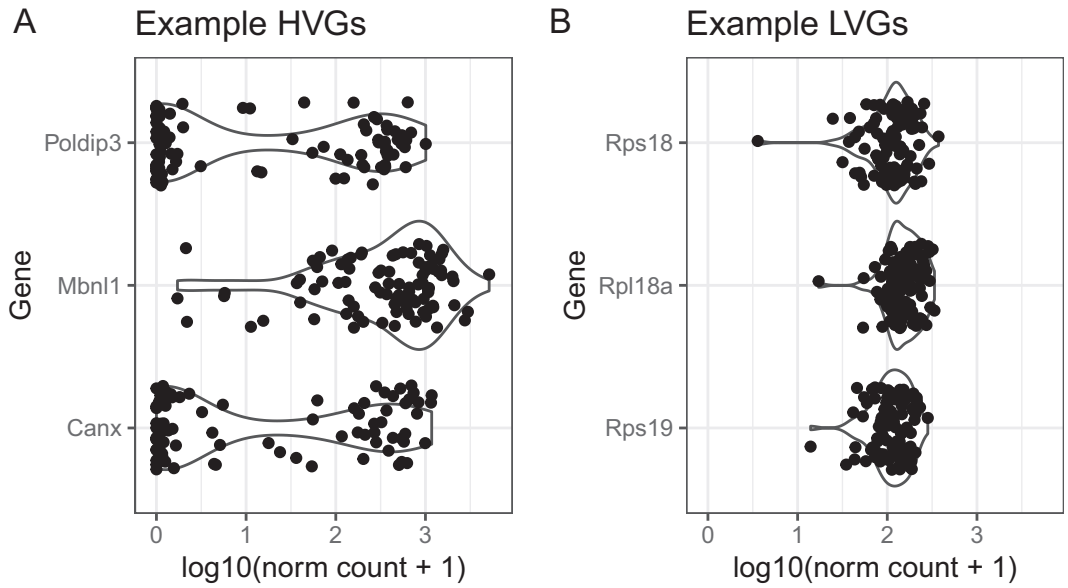


Figure 10. BASiCS denoised counts for example highly-variable gene (HVG) and lowly-variable gene (LVG) with similar overall levels of expression.

For the naive CD4$^+$ T cell data, we obtained *41* HVG and *380* LVG. As shown in Figure 9, these genes are distributed across a wide range of mean expression values. As an illustration, Figure 10 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 10A). Instead, LVG tend to have more narrow and unimodal distributions (Figure 10B).

```
library("reshape2")
## 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),
  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")
```

Differential mean and variability testing using BASiCS

This section highlights the use of *BASiCS* to perform differential expression tests for mean and variability between different pre-specified populations of cells and experimental conditions. Here, we compare the naive $CD4^+$ T cells, analysed in the previous section, to activated $CD4^+$ T cells analysed in the same study.[7] Naive $CD4^+$ T cells were activated for three hours using plate-bound CD3e and CD28 antibodies. T cell activation is linked to strong transcriptional shifts and the up-regulation of lineage specific marker genes, such as Tbx21 and Gata1.[55,56] To generate this data, the authors did not add cytokines, which are needed for T cell differentiation.[57] Therefore, any heterogeneity in the activated cell population does not arise from cells residing in different lineage-specific differentiation states.

Differential expression testing is performed via the `BASiCS_TestDE` function. The main input parameters are

- `Chain1` and `Chain2`: two `BASiCS_Chain` objects created via the `BASiCS_MCMC` function. Each object corresponds to a different pre-specified group of cells.

- `EpsilonM` and `EpsilonR`: introduce a minimum effect size (in a $\log_2$ fold change scale) for the detection of changes in mean or residual over-dispersion, respectively. This enables us to discard small expression changes that are less biologically meaningful. By default, we set these thresholds to be equivalent to a 50% change between the groups. However, different thresholds may be required depending on the context. For example, if most genes show strong differences in mean expression, it can be beneficial to increase the value of `EpsilonM` to focus on strong changes in mean expression.

- `EFDR_M` and `EFDR_R`: define the target EFDR to calibrate the decision rule associated to changes in mean or residual over-dispersion, respectively. Default: 10%.

- `MinESS`: ESS threshold, below which genes will be excluded from the differential expression tests. This is used to increase the robustness of the results, excludes genes for which the sampler explored the parameter space less efficiently (see *MCMC diagnostics* Section). Default: `MinESS` = 100.

```
## Perform differential testing
test_de <- BASiCS_TestDE(
  Chain1 = chain_naive,
  Chain2 = chain_active,
  GroupLabel1 = "Naive",
  GroupLabel2 = "Active",
  EFDR_M = 0.1,
  EFDR_R = 0.1,
  MinESS = 100,
  Plot = FALSE,
  PlotOffset = FALSE
)
table_de_mean <- as.data.frame(
```

```
  test_de,
  Parameter = "Mean",
  Filter = FALSE
)
table_de_resdisp <- as.data.frame(
  test_de,
  Parameter = "ResDisp",
  Filter = FALSE
)
```

After running the test, it is important to visualise the results to facilitate interpretation and to identify systematic patterns among differentially expressed genes. It may also be useful to perform functional enrichment analysis to identify biologically meaningful patterns among these genes. For example, this could be performed using the Bioconductor package *goseq*.[58] We do not perform this here, but a relevant workflow is described by Maksimovic *et al.*[59]

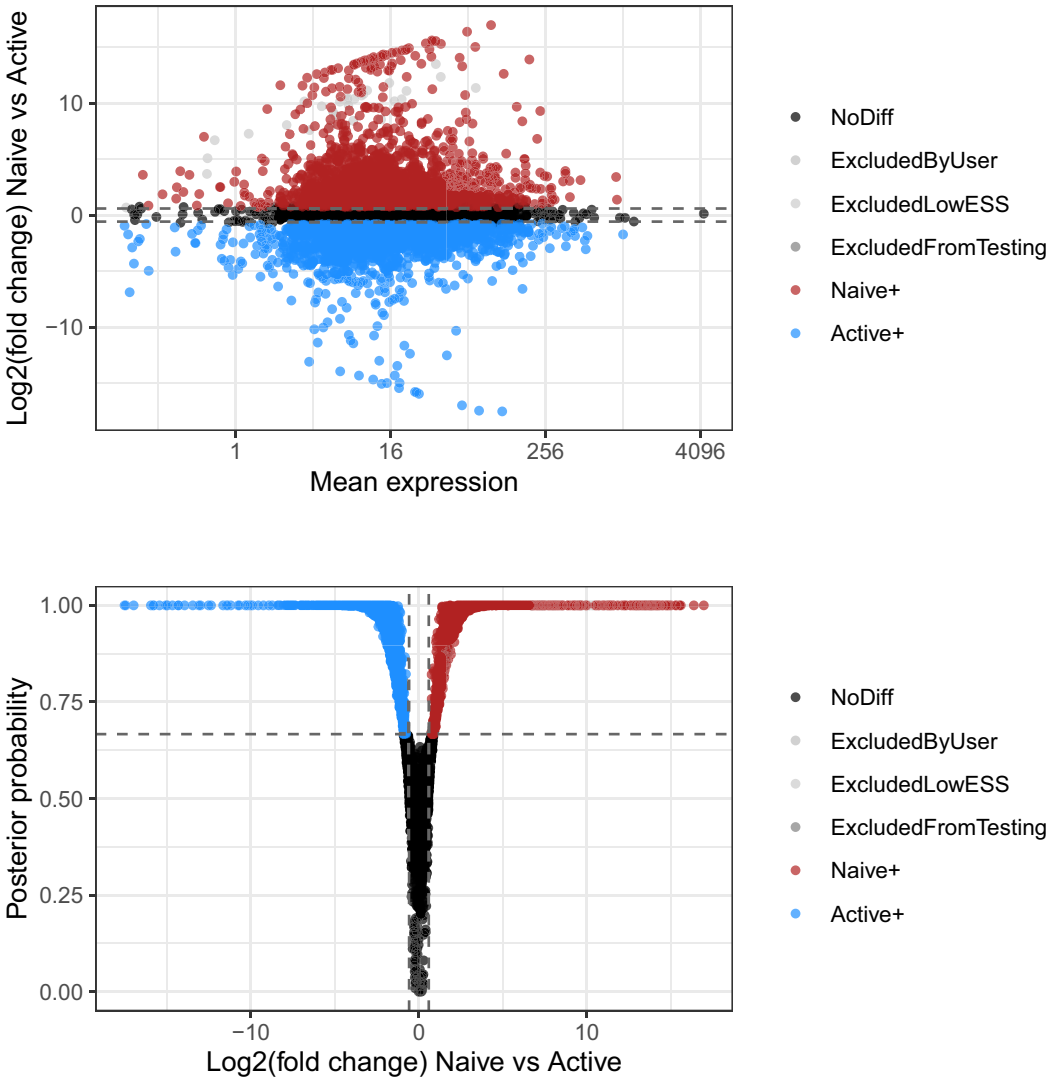


Figure 11. Upper panel presents the mean-difference plot associated to the differential mean expression test between naive and active cells. Log-fold changes of average expression in naive cells relative to active cells are plotted against average expression estimates combined across both groups of cells. 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 first focus on the differential mean expression test. MA-plots (log fold change M versus mean average A) and volcano plots (posterior probability versus log fold change) are popular graphical summaries in this context, and are presented in Figure 11. These can be useful in ensuring that suitable magnitude and confidence thresholds have been chosen. In this instance, it is clear that a large number of genes are differentially expressed between the two conditions, and the selected probability threshold is suitable.

```
p1 <- BASiCS_PlotDE(test_de, Parameters = "Mean", Plots = "MA")
p2 <- BASiCS_PlotDE(test_de, Parameters = "Mean", Plots = "Volcano")
p1 / p2
```

When interpreting the results of differential expression tests, it is useful to visualise expression patterns for differentially expressed genes in order to appraise the significance of the results and guide interpretation. For this purpose, we obtain normalised expression values for each group of cells after correcting for global changes in overall expression between the groups, i.e. a global offset that leads to uniformly higher expression across genes within one group of cells (such step is also internally done by `BASiCS_TestDE`). Among other causes, the latter can be due to overall differences in mRNA content between the groups.

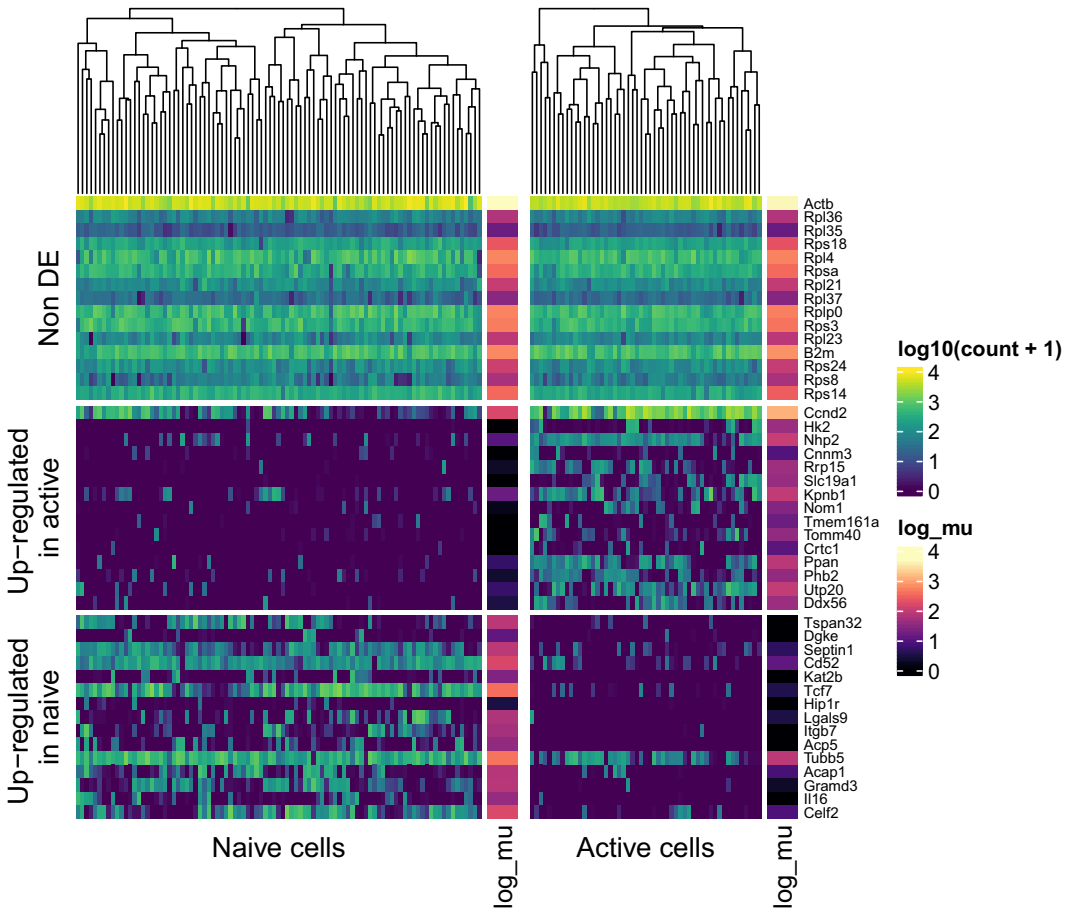


Figure 12. Heatmap displays normalised expression values ($\log_1 0(x+1)$ scale) for naive and active cells. Genes are stratified according to their differential expression status (non differentially expressed; upregulated in naive or active cells). For each group, 15 example genes are shown. These were selected according to the ranking of their associated tail posterior probabilities associated to the differential mean expression test. Colour indicates expression level; colour bars on the right of heatmap segments indicate the inferred mean expression level (in log scale) for each gene in each population.

```
## Calculate a global offset between the groups
offset <- BASiCS_CorrectOffset(chain_naive, chain_active)
offset$Offset
## [1] 0.5987445
## Get offset corrected chain
chain_naive <- offset$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)
```

To visualise expression patterns for multiple genes at once, we use the Bioconductor package *ComplexHeatmap*[60] package, grouping genes according to the result of the differential mean expression test (i.e. up-regulated in naive/active cells or non differentially expressed; see Figure 12). For example, among the non DE group, we observe several genes encoding ribosomal proteins (e.g. *Rps14*). Genes in this family have been previously observed to have stable expression across a wide range of scRNAseq datasets in mouse and human.[8] Such visualisations may aid in the interpretation of such stable or "housekeeping" genes, as well as genes which are up- or down-regulated in each population.

```
library("ComplexHeatmap")
library("circlize")
library("RColorBrewer")
## 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$GeneName)
table_de_selected <- table_de_mean[match_order,]
## Color palette for mu annotation
log_mu_naive <- log10(table_de_selected$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)
```

While several computational tools exist to perform differential mean expression analysis using scRNAseq data,[37] the key focus of *BASiCS* is to perform differential variability testing: identifying changes in transcriptional variability between the groups of cells. To avoid the confounding between mean and over-dispersion, we recommend to use residual over-dispersion parameters ε_i as input to this analysis.

We can now visualise the changes in residual over-dispersion between naive and activated $CD4^+$ T cells in the form of a MA-plot (Figure 13). In this visualisation, the difference between the posterior medians of the residual over-dispersion parameters ε_i are shown on the y-axis. Epsilon values for genes that are not expressed in at least two cells per condition are marked as `NA` and are therefore not displayed.

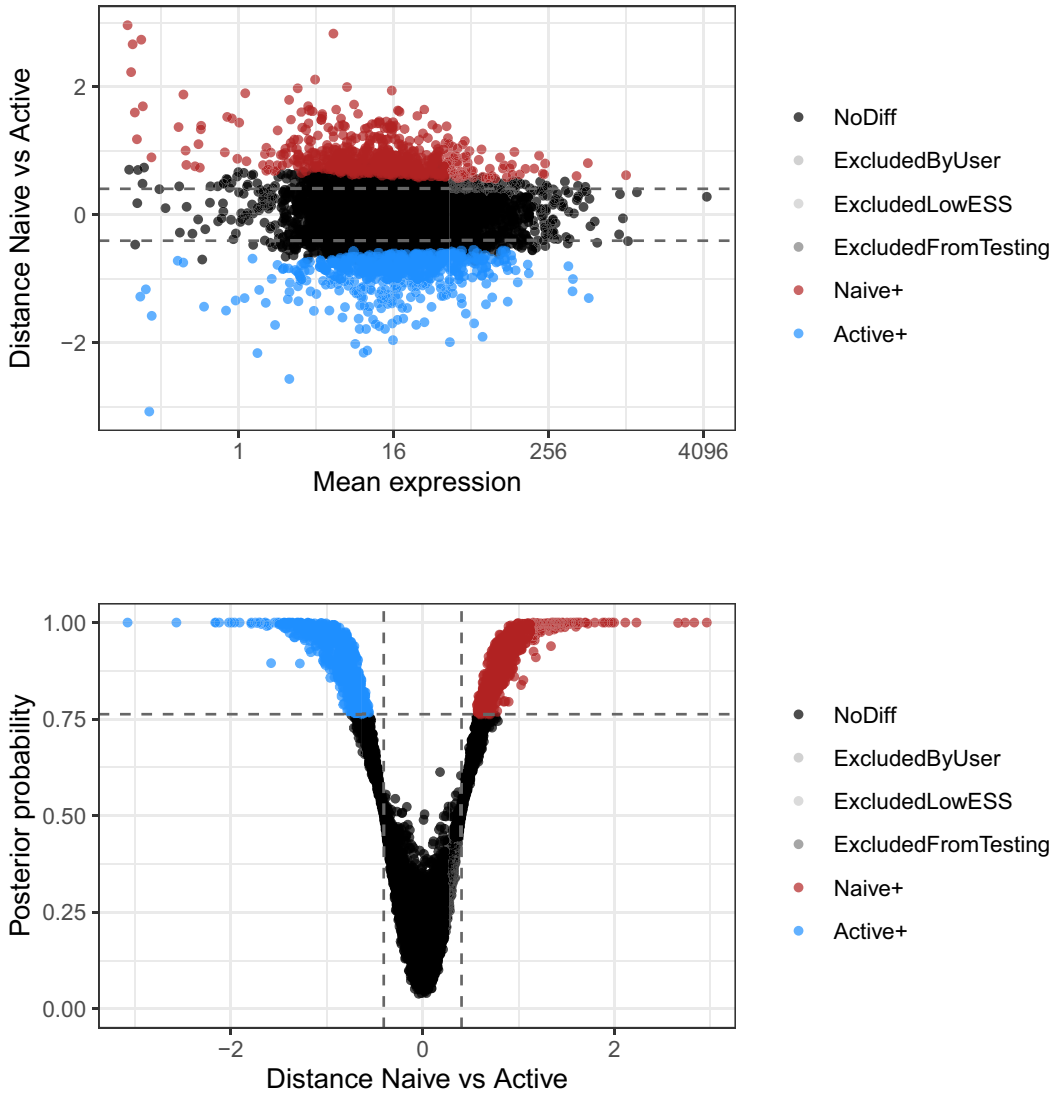


Figure 13. Upper panel presents the mean-difference plot associated to the differential residual over-dispersion test between naive and active cells. Differences of residual over-dispersion in naive cells relative to active cells are plotted against average expression estimates combined across both groups of cells. Bottom panel presents the volcano plot associated to the same test. Differences of residual over-dispersion 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.

```
p1 <- BASiCS_PlotDE(test_de, Parameters = "ResDisp", Plots = "MA")
p2 <- BASiCS_PlotDE(test_de, Parameters = "ResDisp", Plots = "Volcano")
p1 / p2
```

While one could focus on the sets of gene that show significant changes in residual over-dispersion, here we want to highlight how to analyse changes in mean expression in parallel to changes in variability. For this, we will first combine the results of the differential mean expression and the differential residual over-dispersion test. These are independent analyses, given that changes in residual over-dispersion are not confounded by changes in mean expression, as shown in Figure 14.

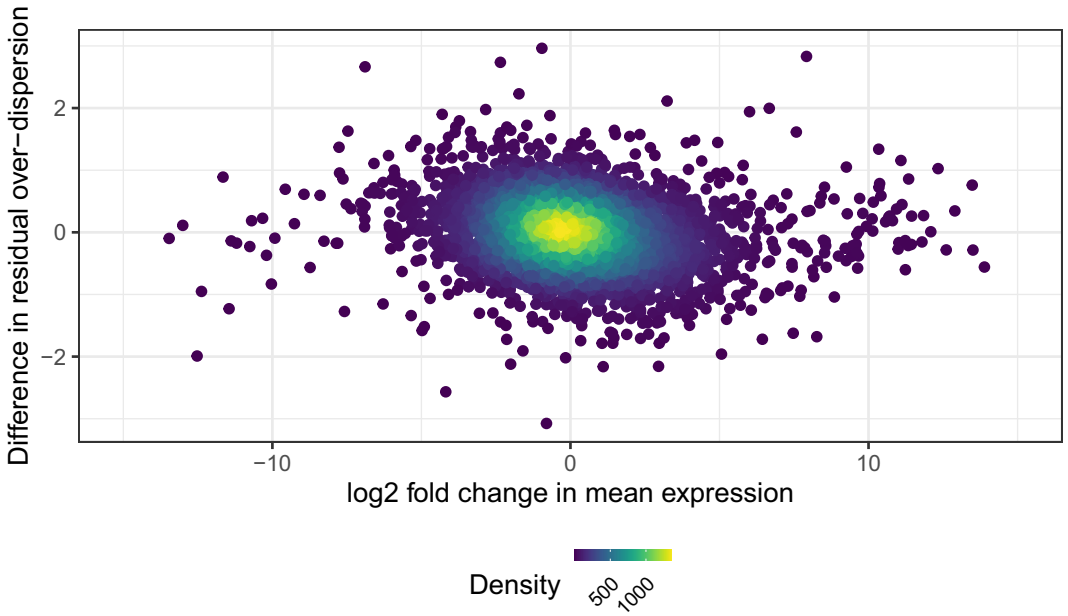


Figure 14. Difference in residual over-dispersion against log2 fold change in mean expression.

```
## combine results of differential mean and residual over-dispersion tests
table_de_combined <- merge(table_de_mean, table_de_resdisp)
## merge with some summary statistics about genes
gene_tests <- data.frame(
  "GeneName" = rownames(dc_naive),
  "ExpPropNaive" = rowMeans(dc_naive > 0),
  "ExpPropActive" = rowMeans(dc_active > 0),
  "nExpPropNaive" = rowSums(dc_naive > 0),
  "nExpPropActive" = rowSums(dc_active > 0)
)
table_combined_genes <- merge(table_de_combined, gene_tests)
## create list of generic plot parameters to use across a few plots
plot_params <- list(
  geom_pointdensity(),
  scale_colour_viridis(name = "Density"),
  theme(
      # text = element_text(size = rel(3)),
      legend.position = "bottom",
      legend.text = element_text(angle = 45, size = 8, hjust = 1, vjust = 1),
      legend.key.size = unit(0.018, "npc")
  )
)
## plot log2FC against difference of residual over-dispersion
ggplot(table_de_combined) +
  aes(MeanLog2FC, ResDispDistance) +
  plot_params +
  xlim(-15, 15) +
  labs(
      x = "log2 fold change in mean expression",
      y = "Difference in residual over-dispersion"
  )
```

While genes with significant changes in residual over-dispersion often have similar levels of mean expression, as seen in Figure 15A and C, they may have a different proportion of zero counts in the two cell populations. Figure 15B and D show that many genes with higher residual over-dispersion in naive cells have a lower proportion of zeros in active cells, and vice versa.

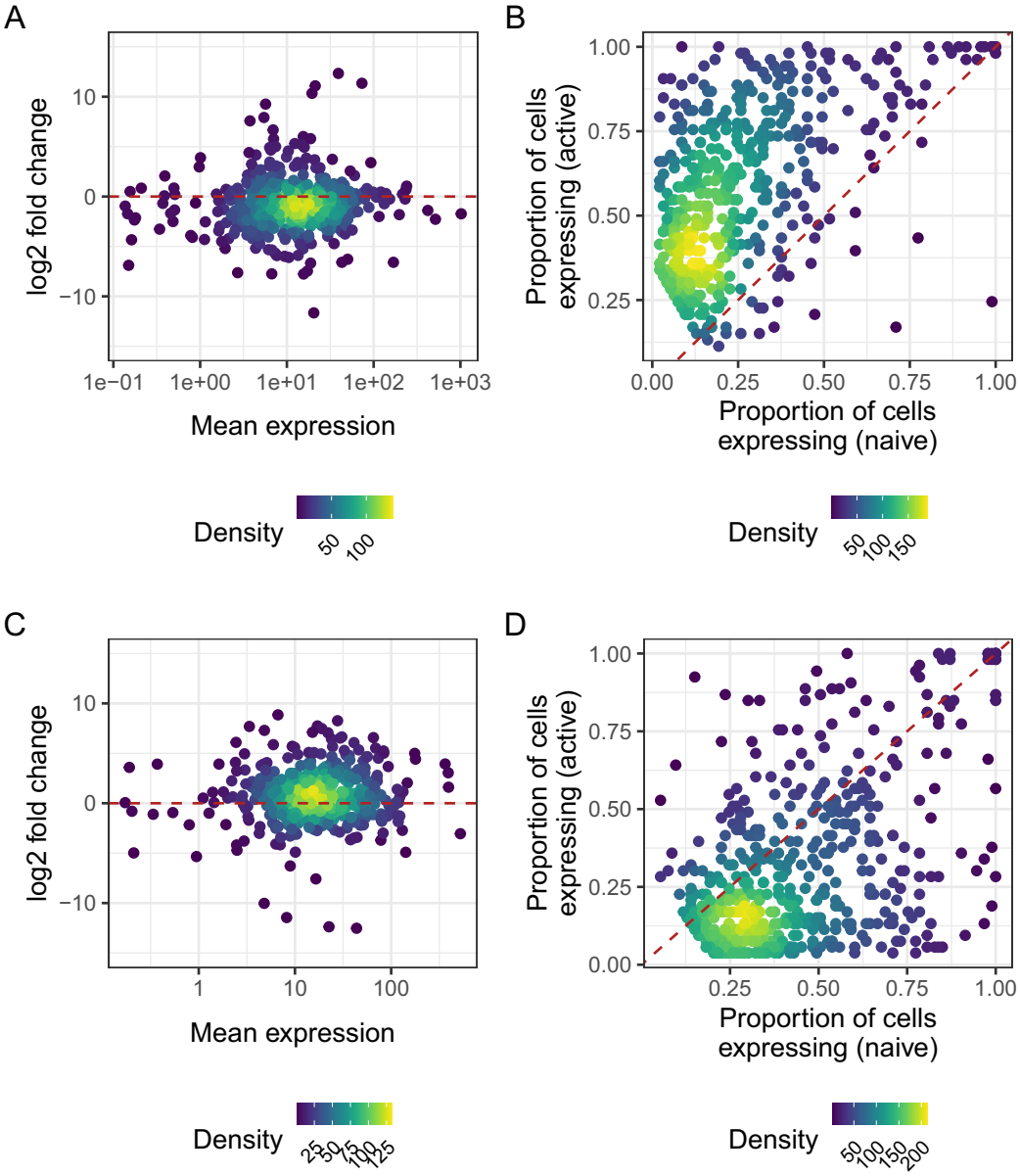


Figure 15. A, C: log2 change in expression against against log mean expression for genes with higher residual over-dispersion in naive (A) cells and active (D) cells. B, D: Proportion of expressed cells for genes, with higher residual over-dispersion in naive cells (B) and active (D) cells. Dashed red lines in panels A and C represent a log fold change of zero, meaning no change in average expression. Dashed red lines in panels B and D represent the line described by y=x, representing equal detection levels in both populations.

```
g1 <- ggplot(
     table_combined_genes[table_combined_genes$ResultDiffResDisp == "Naive+",]
  ) +
  aes(MeanOverall, MeanLog2FC) +
  plot_params +
  ylim(-15, 15) +
  scale_x_log10() +
  labs(x = "Mean expression", y = "log2 fold change") +
  geom_hline(
     yintercept = 0, colour = "firebrick", linetype = "dashed"
  )
g2 <- ggplot(
     table_combined_genes[table_combined_genes$ResultDiffResDisp == "Active+",]
  ) +
  aes(MeanOverall, MeanLog2FC) +
  plot_params +
  ylim(-15, 15) +
  scale_x_log10() +
  labs(x = "Mean expression", y = "log2 fold change") +
  geom_hline(
     yintercept = 0, colour = "firebrick", linetype = "dashed"
  )
g3 <- ggplot(
     table_combined_genes[table_combined_genes$ResultDiffResDisp == "Naive+",]
  ) +
  aes(x = ExpPropNaive, y = ExpPropActive) +
  plot_params +
  labs(
     x = "Proportion of cells\nexpressing (naive)",
     y = "Proportion of cells\nexpressing (active)"
  ) +
  geom_abline(
     slope = 1, intercept = 0, colour = "firebrick", linetype = "dashed"
  )
g4 <- ggplot(
     table_combined_genes[table_combined_genes$ResultDiffResDisp == "Active+",]
  ) +
  aes(x = ExpPropNaive, y = ExpPropActive) +
  plot_params +
  labs(
     x = "Proportion of cells\nexpressing (naive)",
     y = "Proportion of cells\nexpressing (active)"
  ) +
  geom_abline(
     slope = 1, intercept = 0, colour = "firebrick", linetype = "dashed"
  )
(g1 + g3) / (g2 + g4) + plot_annotation(tag_levels = "A")
```

Similarly to analysis of differential expression, it is useful to visualise the results of differential variability tests in order to appraise the quality of the results, and to identify systematic patterns among the genes identified. One useful way to do this is by examining the normalised counts on a gene-by-gene basis. Figure 16 shows denoised counts for genes with significant differences in residual over-dispersion, with each panel showing a different type of expression pattern. Exploration of such patterns is important component of any analysis of differential variability, and should be undertaken with care. Figure 16B and 16D show genes with differing levels of detection in both populations, as well as higher levels of residual over-dispersion in naive or active cells (B and D, respectively). Thus, these genes may represent those with a more bursty expression pattern in one of the cell population. They may also represent genes that are markers of extrinsic variability, for example cell sub-populations that differ in abundance between the cell populations in question. In contrast, Figure 16A and 16C show genes with similar levels of detection in both populations, as well as higher levels of residual

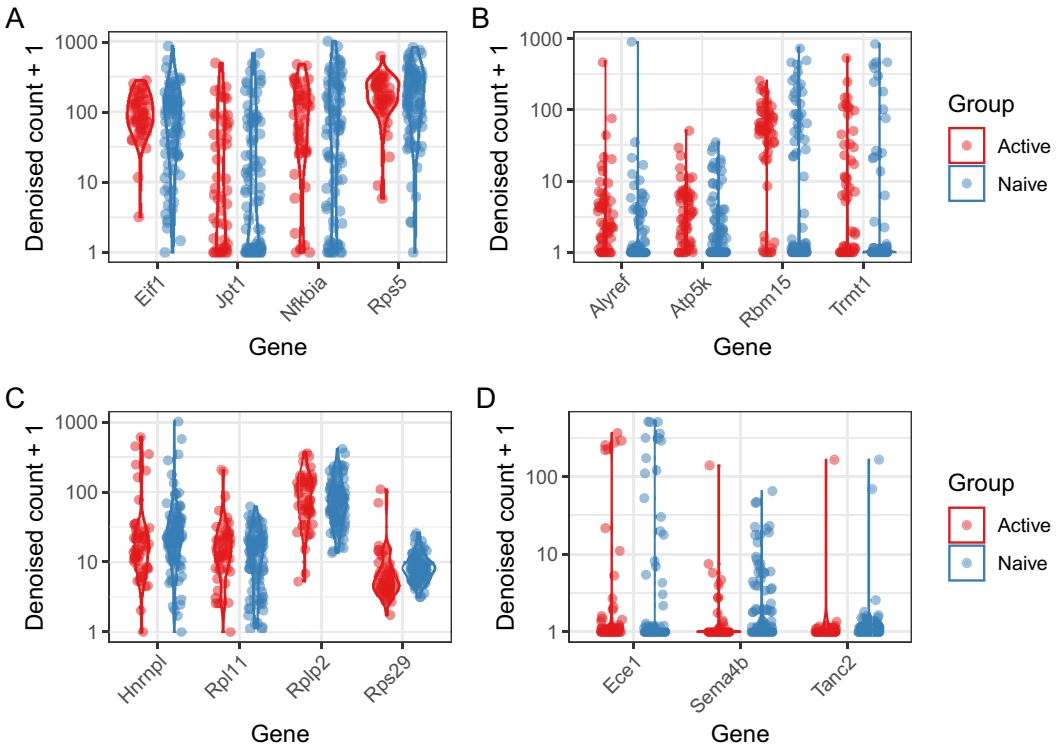


Figure 16. Violin plots of denoised counts. A: Four genes with higher residual over-dispersion in naive cells, and similar levels of detection in active and naive populations. B: Four genes with higher residual over-dispersion in naive cells and different levels of detection in naive and active cells. C: Four genes with higher residual over-dispersion in active cells and similar levels of detection in naive and active cells. D: Three genes with higher residual over-dispersion in active cells and different levels of detection in naive and active cells.

over-dispersion in naive or active cells (A and C, respectively). These cases are likely driven by more tight regulation, rather than transcriptional burst or sub-population structure.

```
## Use the tidyr package to reshape data into a "long" format for ggplot2
library("tidyr")
## Utility function that plots logcounts of a set of genes defined by "ind_vg"
plot_vg <- function(ind_vg) {
  table_combined_genes_vg <- table_combined_genes[ind_vg,]
  ## If more than 4 genes,
  ## pick top 4 ranked by differences in residual over-dispersion
  if (nrow(table_combined_genes_vg) > 4) {
      table_combined_genes_vg <- table_combined_genes_vg[
          order (abs (table_combined_genes_vg$ResDispDistance), decreasing =
TRUE),]
      table_combined_genes_vg <- table_combined_genes_vg[1:4,]
  }
  var_genes <- table_combined_genes_vg$GeneName
  var_naive <- dc_naive[var_genes, , drop = FALSE]
  var_active <- dc_active[var_genes, , drop = FALSE]
  var_df <- rbind(
      data.frame(t(var_naive), Group = "Naive"),
      data.frame(t(var_active), Group = "Active")
  )
  colnames (var_df)[-ncol(var_df)] <- genenames[var_genes, "external_gene_name"]
  var_long_df <- pivot_longer(var_df, cols = colnames(var_df)[-ncol(var_df)])
ggplot (var_long_df, aes(x = name, y = value + 1, colour = Group)) +
```

```
  geom_violin(width = 0.8, position = position_dodge(width = 0.8)) +
  geom_point(
      position = position_jitterdodge(jitter.width = 0.2, dodge.width = 0.8),
      alpha = 0.5
  ) +
  scale_x_discrete(guide = guide_axis(angle = 45)) +
  scale_y_log10() +
  labs(x = "Gene", y = "Denoised count + 1") +
  scale_colour_brewer(palette = "Set1")
}
## genes more variable in naive &
## not differentially expressed &
## with similar levels of non-zero expression &
## expressed in at least 50% of cells within each group
## ie, genes that are probably unimodal but with higher variance in naive cells
ind_vg <- table_combined_genes$ResultDiffResDisp %in% c("Naive+") &
  table_combined_genes$ResultDiffMean %in% c("NoDiff") &
  abs(
      table_combined_genes$ExpPropNaive - table_combined_genes$ExpPropActive
  ) < 0.05 &
  pmin(
      table_combined_genes$ExpPropNaive, table_combined_genes$ExpPropActive
  ) > 0.75
g1 <- plot_vg(ind_vg) +
      theme(legend.position = "none")
## genes more variable in naive &
## not differentially expressed &
## with different levels of non-zero expression &
## expressed in at least 50% of cells within one or more of the groups
## i.e., genes with more bursting pattern of expression
ind_vg <- table_combined_genes$ResultDiffResDisp %in% c("Naive+") &
  table_combined_genes$ResultDiffMean %in% c("NoDiff") &
  abs(
      table_combined_genes$ExpPropNaive - table_combined_genes$ExpPropActive
  ) > 0.40 &
  pmax(
      table_combined_genes$ExpPropNaive, table_combined_genes$ExpPropActive
  ) > 0.75
g2 <- plot_vg(ind_vg)
## genes more variable in Active &
## not differentially expressed &
## with similar levels of non-zero expression &
## expressed in at least 50% of cells within each group
## ie, genes that are probably unimodal but with higher variance
ind_vg <- table_combined_genes$ResultDiffResDisp %in% c("Active+") &
  table_combined_genes$ResultDiffMean %in% c("NoDiff") &
  abs(
      table_combined_genes$ExpPropNaive - table_combined_genes$ExpPropActive
  ) < 0.05 &
  pmin(
      table_combined_genes$ExpPropNaive, table_combined_genes$ExpPropActive
  ) > 0.75
g3 <- plot_vg(ind_vg) +
      theme(legend.position = "none")
## genes more variable in Active &
## not differentially expressed &
## with different levels of dropout &
## expressed in at least 50% of cells within one or more of the groups
## i.e., genes with more bursting pattern of expression
```

```
ind_vg <- table_combined_genes$ResultDiffResDisp %in% c("Active+") &
  table_combined_genes$ResultDiffMean %in% c("NoDiff") &
  abs(
      table_combined_genes$ExpPropNaive - table_combined_genes$ExpPropActive
  ) > 0.20 &
  pmax(
      table_combined_genes$ExpPropNaive, table_combined_genes$ExpPropActive
  ) > 0.50
g4 <- plot_vg(ind_vg)
(g1 + g2) / (g3 + g4) + plot_annotation(tag_levels = "A")
```

Using BASiCS without spike-ins

BASiCS, when using spike-in molecules, uses spike-ins as a reference in order to aid normalisation, based on the assumption that the original quantity of spike-in molecules was approximately equal in each well. Eling *et al.*[29] introduced a novel method of inferring gene expression profiles using *BASiCS* without relying on spike-ins to quantify technical noise. This is useful for droplet-based scRNAseq protocols, given that it is not possible to ensure that each droplet contains a specified quantity of spike-in molecules. In this horizontal integration framework, technical variation is quantified using replication.[61] In the absence of true technical replicates, we assume that population-level characteristics of the cells are replicated using appropriate experimental design. This requires that cells from the same population have been randomly allocated to different batches. Given appropriate experimental design, *BASiCS* assumes that biological effects are shared across batches, while technical variation leads to spurious differences between cells in different batches.

Using *BASiCS* without spike-ins is very similar to using it with spike-ins. We will demonstrate using the naive cells. However, first, we must ensure that a `BatchInfo` field is present in the `SingleCellExperiment` used as input. In this case we use individual of origin as the batch vector.

```
set.seed(42)
chain_naive_nospikes <- BASiCS_MCMC(
  Data = sce_naive,
  PrintProgress = TRUE,
  N = 40000,
  Thin = 20,
  Burn = 20000,
  Regression = TRUE,
  PriorParam = prior_param_naive,
  Threads = 4,
  StoreChains = TRUE,
  StoreDir = "rds/",
  RunName = "naive_nospikes",
  WithSpikes = FALSE
)
```

As before, for convenience we provide a completed version of this chain at https://doi.org/10.5281/zenodo.5243265.

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

The resulting `BASiCS_Chain` object produced using this horizontal integration framework is functionally similar to one produced using the vertical integration framework. It can be used in place of the `BASiCS_Chain` objects produced using the vertical integration approach, as described above.

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 17. 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`.

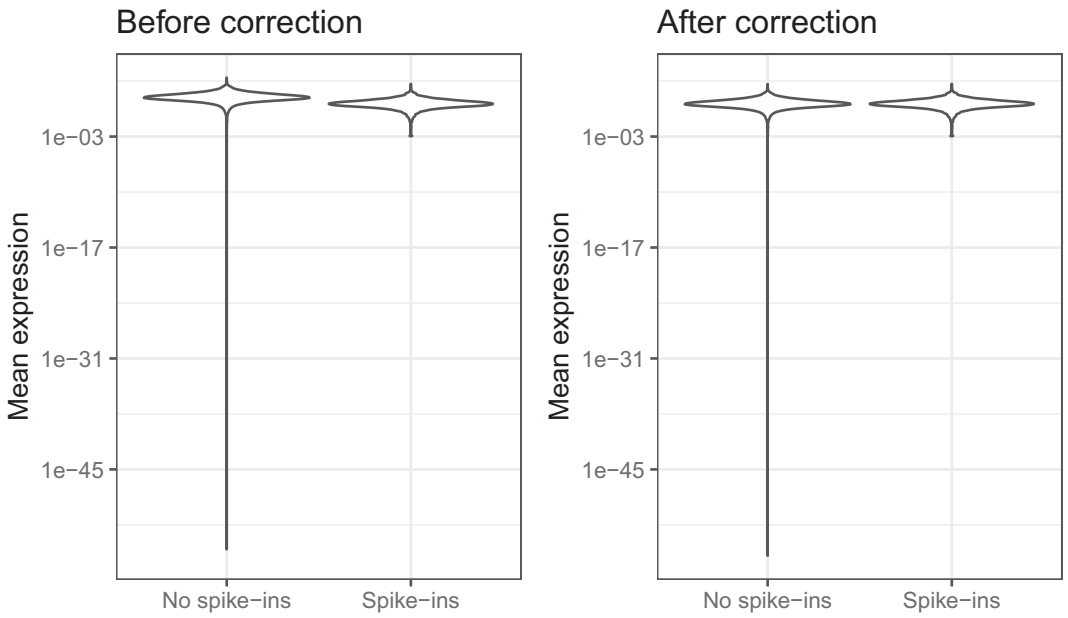


Figure 17. Distribution of mean expression values before and after correcting the global difference in scale.

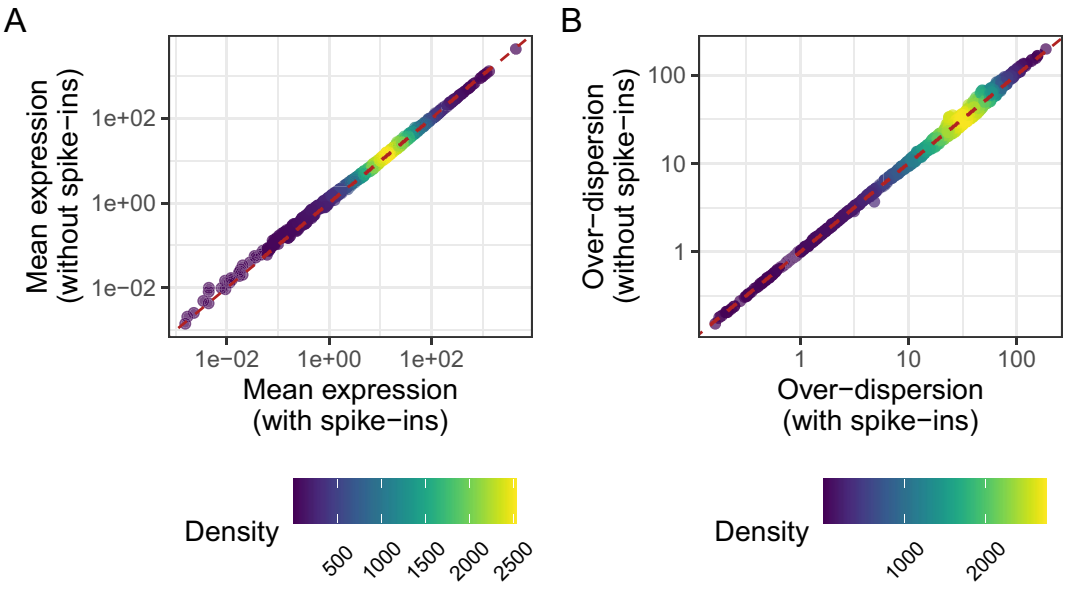


Figure 18. Comparison of point estimates using spike-ins, and the same parameters estimated without using spike-ins for mean expression (A) and over-dispersion (B). A dashed red line indicates the identity line, $y = x$. Genes with zero counts across all cells were excluded from the plot of mean expression parameters.

```
BASiCS_PlotOffset(chain_naive_nospikes, chain_naive,
  GroupLabel1 = "No spike-ins", GroupLabel2 = "Spike-ins",
  Type = "before-after")
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 (2 batches).
## Object stored using BASiCS version: 2.2.1
## 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 Figure 18A and 18B, 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")
```

Discussion

In this article, we have explored the research questions that *BASiCS* seeks to resolve — chiefly, robustly quantifying average and variability in expression in cell populations. We have outlined the appropriate quality control and data visualisation steps to apply when undertaking an analysis using *BASiCS* in order to ensure high quality input data. We have also outlined the steps needed to use *BASiCS* to quantify biological variability, identify highly variable genes, and normalise scRNAseq data from a single population. We have also provided a limited comparison of the results of these analyses using *BASiCS* and the result of similar analyses using *scran*. Furthermore, we have demonstrated functions within *BASiCS* that allow users to ensure the MCMC used in *BASiCS* has converged and produced adequate sample sizes. Finally, we have demonstrated the use of *BASiCS* to robustly identify differentially expressed genes, in terms of mean expression and in terms of biological variability.

Further challenges exist in analysing scRNAseq data.[10,38] For *BASiCS*, the primary challenge currently is computational efficiency. The number of cells profiled in scRNAseq experiments has scaled exponentially since the development of the technology.[62] Given that *BASiCS* requires computationally intensive MCMC sampling to estimate the posterior distribution, it becomes computationally intractable to analyse data from very large numbers of cells. We intend to update this workflow as the field evolves, and as we address the issues and challenges outlined here.

Data availability

ArrayExpress: RNA-seq of coding RNA from single cells [Mus musculus (house mouse)]. Accession number E-MTAB-4888; https://identifiers.org/arrayexpress:E-MTAB-4888.

All data underlying the results are available as part of the article and no additional source data are required.

Software availability

All software used in this workflow is available as part of Bioconductor 3.13 at: `https://bioconductor.org/packages/3.13`.

The source code for BASiCS, along with facilities contributing and reporting bugs is available at: `https://github.com/catavallejos/BASiCS/`.

The source code used for this manuscript is available at: `https://github.com/VallejosGroup/BASiCS-Workflow/`, and as archived source code at time of publication: https://doi.org/10.5281/zenodo.5243265.[32]

License: GPL-2.0

Reproducibility

The following software versions were used throughout this workflow:

- **R version**: R version 4.1.1 (2021-08-10)
- **Bioconductor version**: 3.13

- **R packages**:
  - BASiCS 2.4.0
  - scran 1.20.1
  - scater 1.20.1

Version numbers for all remaining packages are available in the Session Info section.

A Docker image containing all software requirements is available at Docker hub. This image can be downloaded using the command `docker pull alanocallaghan/bocker:0.2.0`.

Session Info

```
sessionInfo()
## R version 4.1.1 (2021-08-10)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.3 LTS
##
## Matrix products: default
## BLAS/LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.8.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8           LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8            LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8        LC_MESSAGES=C
##  [7] LC_PAPER=en_US.UTF-8           LC_NAME=C
##  [9] LC_ADDRESS=C                   LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8     LC_IDENTIFICATION=C
##
## attached base packages:
## [1] grid        parallel stats4 stats graphics grDevices utils
## [8] datasets  methods  base
##
## other attached packages:
##  [1] tidyr_1.1.4                   RColorBrewer_1.1-2
##  [3] circlize_0.4.13               ComplexHeatmap_2.8.0
##  [5] reshape2_1.4.4                ggpointdensity_0.1.0
##  [7] viridis_0.6.1                 viridisLite_0.4.0
##  [9] coda_0.19-4                   patchwork_1.1.1
## [11] biomaRt_2.48.3                BASiCS_2.4.0
## [13] scran_1.20.1                  scater_1.20.1
## [15] scuttle_1.2.1                 SingleCellExperiment_1.14.1
## [17] SummarizedExperiment_1.22.0   Biobase_2.52.0
## [19] GenomicRanges_1.44.0          GenomeInfoDb_1.28.4
## [21] IRanges_2.26.0                S4Vectors_0.30.2
## [23] BiocGenerics_0.38.0           MatrixGenerics_1.4.3
## [25] matrixStats_0.61.0            ggplot2_3.3.5
## [27] knitr_1.36                    BiocStyle_2.20.2
##
## loaded via a namespace (and not attached):
##  [1] BiocFileCache_2.0.0           plyr_1.8.6
##  [3] igraph_1.2.6                  BiocParallel_1.26.2
##  [5] usethis_2.0.1                 digest_0.6.28
##  [7] foreach_1.5.1                 htmltools_0.5.2
##  [9] magick_2.7.3                  fansi_0.5.0
## [11] magrittr_2.0.1                memoise_2.0.0
```

```
## [13] ScaledMatrix_1.0.0              cluster_2.1.2
## [15] doParallel_1.0.16               limma_3.48.3
## [17] Biostrings_2.60.2               prettyunits_1.1.1
## [19] colorspace_2.0-2                blob_1.2.2
## [21] rappdirs_0.3.3                  BiocWorkflowTools_1.18.0
## [23] xfun_0.26                       dplyr_1.0.7
## [25] crayon_1.4.1                    RCurl_1.98-1.5
## [27] hexbin_1.28.2                   iterators_1.0.13
## [29] glue_1.4.2                      gtable_0.3.0
## [31] zlibbioc_1.38.0                 XVector_0.32.0
## [33] GetoptLong_1.0.5                DelayedArray_0.18.0
## [35] BiocSingular_1.8.1              shape_1.4.6
## [37] scales_1.1.1                    DBI_1.1.1
## [39] edgeR_3.34.1                    miniUI_0.1.1.1
## [41] Rcpp_1.0.7                      xtable_1.8-4
## [43] progress_1.2.2                  clue_0.3-59
## [45] dqrng_0.3.0                     bit_4.0.4
## [47] rsvd_1.0.5                      metapod_1.0.0
## [49] httr_1.4.2                      ellipsis_0.3.2
## [51] pkgconfig_2.0.3                 XML_3.99-0.8
## [53] farver_2.1.0                    dbplyr_2.1.1
## [55] locfit_1.5-9.4                  utf8_1.2.2
## [57] tidyselect_1.1.1                labeling_0.4.2
## [59] rlang_0.4.11                    later_1.3.0
## [61] AnnotationDbi_1.54.1            munsell_0.5.0
## [63] tools_4.1.1                     cachem_1.0.6
## [65] generics_0.1.0                  RSQLite_2.2.8
## [67] evaluate_0.14                   stringr_1.4.0
## [69] fastmap_1.1.0                   yaml_2.2.1
## [71] bit64_4.0.5                     fs_1.5.0
## [73] purrr_0.3.4                     KEGGREST_1.32.0
## [75] sparseMatrixStats_1.4.2         mime_0.11
## [77] ggExtra_0.9                     xml2_1.3.2
## [79] compiler_4.1.1                  rstudioapi_0.13
## [81] beeswarm_0.4.0                  filelock_1.0.2
## [83] curl_4.3.2                      png_0.1-7
## [85] tibble_3.1.4                    statmod_1.4.36
## [87] stringi_1.7.4                   lattice_0.20-45
## [89] bluster_1.2.1                   Matrix_1.3-4
## [91] vctrs_0.3.8                     pillar_1.6.3
## [93] lifecycle_1.0.1                 BiocManager_1.30.16
## [95] GlobalOptions_0.1.2             BiocNeighbors_1.10.0
## [97] cowplot_1.1.1                   bitops_1.0-7
## [99] irlba_2.3.3                     httpuv_1.6.3
## [101] R6_2.5.1                       bookdown_0.24
## [103] promises_1.2.0.1               gridExtra_2.3
## [105] vipor_0.4.5                    codetools_0.2-18
## [107] MASS_7.3-54                    assertthat_0.2.1
## [109] rjson_0.2.20                   withr_2.4.2
## [111] GenomeInfoDbData_1.2.6         hms_1.1.1
## [113] beachmat_2.8.1                 rmarkdown_2.11
## [115] DelayedMatrixStats_1.14.3      Cairo_1.5-12.2
## [117] git2r_0.28.0                   shiny_1.7.1
## [119] ggbeeswarm_0.6.0
```

References

1. Stegle O, Teichmann SA, Marioni JC: **Computational and analytical challenges in single-cell transcriptomics.** *Nat. Rev. Genet.* jan 2015; **16**(3): 133–145.
PubMed Abstract | Publisher Full Text

2. Prakadan SM, Shalek AK, Weitz DA: **Scaling by shrinking: empowering single-cell'omics' with microfluidic devices.** *Nat. Rev. Genet.* 2017; **18**(6): 345–361.
PubMed Abstract | Publisher Full Text

3. Patange S, Girvan M, Larson DR: **Single-cell systems biology: Probing the basic unit of information flow.** *Curr. Opin. Syst. Biol.* 2018; **8**: 7–15.
Publisher Full Text

4. Kiselev VY, Andrews TS, Hemberg M: **Challenges in unsupervised clustering of single-cell RNA-seq data.** *Nat. Rev. Genet. 2018.* 2019; **20**: 273–282.
PubMed Abstract | Publisher Full Text | Reference Source

5. Saelens W, Cannoodt R, Todorov H, *et al.*: **A comparison of single-cell trajectory inference methods.** *Nat. Biotechnol.* May 2019; **37**(5): 547–554.
Publisher Full Text

6. Mojtahedi M, Skupin A, Zhou J, *et al.*: **Cell Fate Decision as High-Dimensional Critical State Transition.** *PLoS Biol.* December 2016; **14**(12): e2000640.
PubMed Abstract | Publisher Full Text

7. Martinez-Jimenez CP, Eling N, Chen H-C, *et al.*: **Aging increases cell-to-cell transcriptional variability upon immune stimulation.** *Science.* 2017; **355**: 1433–1436.
PubMed Abstract | Publisher Full Text

8. Lin Y, Ghazanfar S, Strbenac D, *et al.*: **Evaluating stably expressed genes in single cells.** *GigaScience.* September 2019; **8**(9): giz106.
PubMed Abstract | Publisher Full Text

9. Elowitz MB, Levine AJ, Siggia ED, *et al.*: **Stochastic gene expression in a single cell.** *Science.* 2002; **2970**(5584): 1183–1186.
Publisher Full Text | Reference Source

10. Eling N, Morgan MD, Marioni JC: **Challenges in measuring and understanding biological noise.** *Nat. Rev. Genet.* September 2019; **200**(9): 536–548.
Publisher Full Text

11. Zopf CJ, Quinn K, Zeidman J, *et al.*: **Cell-Cycle Dependence of Transcription Dominates Noise in Gene Expression.** *PLoS Comput. Biol.* 2013; **9**(7): 1–12.
Publisher Full Text

12. Iwamoto K, Shindo Y, Takahashi K: **Modeling Cellular Noise Underlying Heterogeneous Cell Responses in the Epidermal Growth Factor Signaling Pathway.** *PLoS Comput. Biol.* 2016; **12**(11): e1005222–18.
PubMed Abstract | Publisher Full Text

13. Kiviet DJ, Nghe P, Walker N, *et al.*: **Stochasticity of metabolism and growth at the single-cell level.** *Nature.* 2014; **514**(7522): 376–379.
PubMed Abstract | Publisher Full Text

14. Eberwine J, Kim J: **Cellular Deconstruction: Finding Meaning in Individual Cell Variation.** *Trends Cell Biol.* 2015; **25**(10): 569–578.
Publisher Full Text

15. Faure AJ, Schmiedel JM, Lehner B: **Systematic Analysis of the Determinants of Gene Expression Noise in Embryonic Stem Cells.** *Cell Systems.* 2017; **5**(5): 471–484.e4.
Publisher Full Text

16. Morgan MD, Marioni JC: **CpG island composition differences are a source of gene expression noise indicative of promoter responsiveness.** *Genome Biol.* 2018; **19**(1): 81–13.
PubMed Abstract | Publisher Full Text

17. Brennecke P, Anders S, Kim JK, *et al.*: **Accounting for technical noise in single-cell RNA-seq experiments.** 2013; 1548-7105.
Reference Source

18. External RNA Controls Consortium: **Proposed methods for testing and selecting the ERCC external RNA controls.** *BMC Genom.* 2005; **6**(1): 150.
PubMed Abstract | Publisher Full Text

19. McCarthy DJ, Campbell KR, Lun ATL, *et al.*: **Scater: Pre-processing, quality control, normalization and visualization of single-cell RNA-seq data in R.** *Bioinformatics.* 2017; **33**(8): 1179–1186.
PubMed Abstract | Publisher Full Text

20. Vallejos CA, Risso D, Scialdone A, *et al.*: **Normalizing single-cell RNA sequencing data: challenges and opportunities.** *Nat. Methods.* 2017; **14**(6): 565–571.
PubMed Abstract | Publisher Full Text

21. Islam S, Zeisel A, Joost S, *et al.*: **Quantitative single-cell RNA-seq with unique molecular identifiers.** *Nat. Methods.* February 2014; **11**(2): 163–166.
Publisher Full Text

22. Haque A, Engel J, Teichmann SA, *et al.*: **A practical guide to single-cell RNA-sequencing for biomedical research and clinical applications.** *Genome Med.* December 2017; **9**(1): 75.
Publisher Full Text

23. Vallejos CA, Marioni JC, Richardson S: **BASiCS: Bayesian analysis of single-cell sequencing data.** *PLoS Comput. Biol.* 2015 a; **11**: e1004333.
PubMed Abstract | Publisher Full Text

24. Vallejos CA, Richardson S, Marioni JC: **Beyond comparisons of means: understanding changes in gene expression at the single-cell level.** *Genome Biol.* 2016; **170**(70).
Publisher Full Text | Reference Source

25. Eling N, Richard AC, Richardson S, *et al.*: **Robust expression variability testing reveals heterogeneous T cell responses.** *bioRxiv.* 2017; 237214.
Publisher Full Text | Reference Source

26. Svensson V: **Droplet scRNA-seq is not zero-inflated.** *Nat. Biotechnol.* February 2020; **38**(2): 147–150.
Publisher Full Text

27. William Townes F: **Review of Probability Distributions for Modeling Count Data.** *arXiv:2001.04343 [stat].* January 2020.

28. William Townes F, Hicks SC, Aryee MJ, *et al.*: **Feature selection and dimension reduction for single-cell RNA-Seq based on a multinomial model.** *Genome Biol.* December 2019; **20**(1): 295.
PubMed Abstract | Publisher Full Text

29. Eling N, Richard AC, Richardson S, *et al.*: **Correcting the Mean-Variance Dependency for Differential Variability Testing Using Single-Cell RNA Sequencing Data.** *Cell Systems.* 2018; **7**(3): 284–294. e12.
PubMed Abstract | Publisher Full Text | Reference Source

30. Lun ATL, McCarthy DJ, Marioni JC: **A step-by-step workflow for basic analyses of single-cell RNA-seq data.** *F1000Res.* 2016 a; **5**(2122).
Publisher Full Text

31. Kim B, Lee E, Kim JK: **Analysis of Technical and Biological Variabilityin Single-Cell RNA Sequencing.** *Computational Methods for Single-Cell Data Analysis.* 2019; volume **1935**. pages 25–43.
Publisher Full Text

32. O`Callaghan A, Eling N, Marioni JC, *et al.*: **BASiCS workflow: a step-by-step analysis of expression variability using single cell RNA sequencing data.** August 2021.
Publisher Full Text

33. Boettiger C: **An introduction to Docker for reproducible research.** *ACM SIGOPS Operating Systems Review.* January 2015; **49**(1): 71–79.
Publisher Full Text

34. Vallejos CA, Marioni JC, Richardson S: **BASiCS: Bayesian analysis of single-cell sequencing data.** *PLoS Comput. Biol.* 2015 b; **11**(6): e1004333.
PubMed Abstract | Publisher Full Text

35. Kharchenko PV, Silberstein L, Scadden DT: **Bayesian approach to single-cell differential expression analysis.** *Nat. Methods.* jul 2014; **11**(7): 740–2.
PubMed Abstract | Publisher Full Text | Reference Source

36. Finak G, McDavid A, Yajima M, *et al.*: **MAST: A flexible statistical framework for assessing transcriptional changes and characterizing heterogeneity in single-cell RNA sequencing data.** *Genome Biol.* December 2015; **16**(1): 278.
PubMed Abstract | Publisher Full Text

37. Soneson C, Robinson MD: **Bias, robustness and scalability in single-cell differential expression analysis.** *Nat. Methods.* April 2018; **150**(4): 255–261.
Publisher Full Text

38. Lähnemann D, Köster J, Szczurek E, *et al.*: **Eleven grand challenges in single-cell data science.** *Genome Biol.* December 2020; **21**(1): 31.
PubMed Abstract | Publisher Full Text

39. Amezquita OA, Carey VJ, Carpp LN, *et al.*: **Orchestrating Single-Cell Analysis with Bioconductor.** *Preprint, Genomics.* March 2019.

40. R Core Team: *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing; 2021.
Reference Source

41. Ilicic T, Kim JK, Kolodziejczyk AA, *et al.*: **Classification of low quality cells from single-cell RNA-seq data.** *Genome Biol.* 2016; **17**(29): 29–15.
PubMed Abstract | Publisher Full Text

42. Lun ATL, Bach K, Marioni JC: **Pooling across cells to normalize single-cell RNA sequencing data with many zero counts.** *Genome Biol.* 2016; **17**(1): 75.
PubMed Abstract | Publisher Full Text

43. Kolodziejczyk AA, Kim JK, Tsang JCH, *et al.*: **Single cell RNA-sequencing of pluripotent states unlocks modular transcriptional variation.** *Cell Stem Cell.* 2015; **17**: 471–485.
PubMed Abstract | Publisher Full Text | Reference Source

44. Durinck S, Spellman PT, Birney E, *et al.*: **Mapping identifiers for the integration of genomic datasets with the R/Bioconductor package biomaRt.** *Nat. Protoc.* August 2009; **4**(8): 1184–1191.
Publisher Full Text

45. Lopez R, Regier J, Cole MB, *et al.*: **Deep generative modeling for single-cell transcriptomics.** *Nat. Methods.* December 2018; **15**(12): 1053–1058.
Publisher Full Text

46. Roberts GO, Rosenthal JS: **Examples of Adaptive MCMC.** *J. Comput. Graph. Stat.* January 2009; **180**(2): 349–367.
Publisher Full Text

47. Casella G: **An Introduction to Empirical Bayes Data Analysis.** *Am. Stat.* May 1985; **39**(2): 83.
Publisher Full Text

48. Cowles MK, Carlin BP: **Markov chain monte carlo convergence diagnostics: A comparative review.** *J. Am. Stat. Assoc.* 1996; **91**(434): 883–904.
Publisher Full Text

49. Brooks SP, Gelman A: **General methods for monitoring convergence of iterative simulations.** *J. Comput. Graph. Stat.* 1998; **7**(4): 434–455.
Publisher Full Text

50. Geweke J, In F: **Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments.** 1995; **4**. 11.

51. Koehler E, Brown E, Sebastien J-PAH: **On the Assessment of Monte Carlo Error in Simulation-Based Statistical Analyses.** *Am. Stat.* May 2009; **630**(2): 155–162.
Publisher Full Text

52. Gelman A, Carlin J, Stern H, *et al.*: *Bayesian Data Analysis.* CRC Press; 2014. ISBN 978-1439840955.

53. Antolović V, Miermont A, Corrigan AM, Chubb JR: **Generation of Single-Cell Transcript Variability by Repression.** *Curr. Biol.* 2017; **27**: 1811–1817.e3.
PubMed Abstract | Publisher Full Text | Reference Source

54. Newton MA: **Detecting differential gene expression with a semiparametric hierarchical mixture method.** *Biostatistics.* April 2004; **50**(2): 155–176.
Publisher Full Text

55. Best JA, Knell J, Yang E, *et al.*: **Transcriptional insights into the CD8$^+$ T cell response to infection and memory T cell formation.** *Nat. Immunol.* April 2013; **140**(4): 404–412.
Publisher Full Text

56. Wenxian F, Ergun A, Lu T, *et al.*: **A multiply redundant genetic switch'locks in' the transcriptional signature of regulatory T cells.** *Nat. Immunol.* October 2012; **13**(10): 972–980.
Publisher Full Text

57. Zhu J, Paul WE: **Peripheral CD4$^+$ T-cell differentiation regulated by networks of cytokines and transcription factors: Transcription factor network in Th cells.** *Immunol. Rev.* November 2010; **238**(1): 247–262.
PubMed Abstract | Publisher Full Text

58. Young MD, Wakefield MJ, Smyth GK, *et al.*: **Gene ontology analysis for RNA-seq: accounting for selection bias.** *Genome Biol.* 2010; **11**.
PubMed Abstract | Publisher Full Text | Reference Source

59. Maksimovic J, Phipson B, Oshlack A: **A cross-package Bioconductor workflow for analysing methylation array data [version 3; peer review: 4 approved].** *F1000Res.* 2020; **5**(1281).
PubMed Abstract | Publisher Full Text

60. Zuguang G, Eils R, Schlesner M: **Complex heatmaps reveal patterns and correlations in multidimensional genomic data.** *Bioinformatics.* 2016; **32**: 2847–2849.
Publisher Full Text

61. Carroll RJ: **Measurement Error in Epidemiologic Studies.** 2005; page 38.

62. Svensson V, Vento-Tormo R, Teichmann SA: **Exponential scaling of single-cell RNA-seq in the past decade.** *Nat. Protoc.* April 2018; **130**(4): 599–604.
Publisher Full Text

Open Peer Review

Current Peer Review Status: ? ? ?

Version 1

Reviewer Report 19 April 2022

https://doi.org/10.5256/f1000research.78169.r128930

?

Michel S Naslavsky
Human Genome and Stem Cell Research Center, University of São Paulo, São Paulo, Brazil

O'Callaghan and colleagues present an approach to analyze single-cell RNA sequencing data in a stepwise manner. This is very useful both to readers and to students engaged in learning applied bioinformatics. As a user (non-developer, non-expert in RNA analysis tool evaluation), my contribution will be from a different perspective from the other reviewers who made excellent and detailed comments on this manuscript.

1) Introduction, 2nd paragraph: “Moreover, these variability estimates can also be inflated by the technical noise that is typically observed in scRNA-seq data”

Comment: Please illustrate with examples of technical noise as presented above for extrinsic and intrinsic noise, e.g. RNA differential degradation, ligation bias, etc.

2) Introduction, 3rd paragraph: “However, despite the benefits associated to the use of spike-ins and UMIs, these are not available for all scRNA-seq protocols.”

Comment: Please provide a reason - something like “due to the very nature of the assay, which isolates library prep from external spike-ins and uses UMIs to map single-cell libraries…” or that spike-ins added to the library after pooling limit the full advantage of using them across single-cell populations.

3) Methods, 2nd paragraph: “Mean parameters *μi* quantify the overall expression for each gene *i* across the cell population under study.”

Comment: Please define cell population as sample (all cells within a scRNA-seq assay) or post-processing UMAP, t-SNE, or similar clustering grouping. I understood the latter, but readers should be certain, and we need to understand which step these parameters refer to.

4) After Figure 5: “These thresholds can vary across datasets and should be informed by gene-specific QC metrics such as those shown in Figure 5 as well as prior knowledge about the cell types

and conditions being studied, where available".

Comment: I understand this is not the intent of this study, but if possible include some rationale behind thresholds for QCing gene exclusion based on types and conditions. Maybe bringing the choice for the example explored here.

5) After spike-in calculation step: "To update the sce_naive and sce_active objects, the user must create a data.frame whose first column contains the spike-in labels (e.g. ERCC-00130) and whose second column contains the number of molecules calculated above. We add this as row metadata for altExp (sce_naive) and altExp (sce_active)."

Comment: By 'update', do the authors mean 'update after normalisation with spike-in'? If so, please specify.

7) MCMC diagnostics

Comment: I am by far not an expert on the analysis, but the authors were very successful in explaining the protocol in a stepwise manner. I would kindly ask if they can add the reason why this is needed: "to ensure that comparisons of gene expression are not random" or something in that direction. Readers will appreciate it.

8) Figures 15 and 16

Comment: These explanations are important to establish a convincing argument for using this workflow. Authors can explore further the 'tight regulation' versus 'transcriptional burst' or 'sub-population structure' scenarios with other examples (either from the literature or running an orthogonal analysis on larger scRNA-seq datasets of CD4+ T cells.

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Population and medical genomics

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 29 Jan 2024

Alan O'Callaghan

O'Callaghan and colleagues present an approach to analyse single-cell RNA sequencing data in a stepwise manner. This is very useful both to readers and to students engaged in learning applied bioinformatics. As a user (non-developer, non-expert in RNA analysis tool evaluation), my contribution will be from a different perspective from the other reviewers who made excellent and detailed comments on this manuscript.

Authors' response:
We are grateful for the positive comment provided by Dr Naslavsky about the usefulness of our workflow. Many thanks for highlighting the issues listed below, we believe that the clarity of our manuscript has substantially improved after addressing these comments. Edits to the main text are highlighted in *italics*.

Reviewer comment:
1) Introduction, 2nd paragraph: "Moreover, these variability estimates can also be inflated by the technical noise that is typically observed in scRNA-seq data"
Comment: Please illustrate with examples of technical noise as presented above for extrinsic and intrinsic noise, e.g. RNA differential degradation, ligation bias, etc.
Authors' response:
We have edited the Introduction to provide additional information about potential sources for technical noise. The following text has been added at the end of the second paragraph: *This technical noise relates to systematic differences between cells that may be introduced by the data generating process (e.g., due to differences in dilution or sequencing depth) [@Vallejos2017].*

Reviewer comment:
2) Introduction, 3rd paragraph: "However, despite the benefits associated to the use of spike-ins and UMIs, these are not available for all scRNA-seq protocols."
Comment: Please provide a reason - something like "due to the very nature of the assay, which isolates library prep from external spike-ins and uses UMIs to map single-cell libraries..." or that spike-ins added to the library after pooling limit the full advantage of using them across single-cell populations.
Authors' response:
We have added the following text to note the difficulties in using UMIs and spike-ins with some assays. For more detail, see https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4823857/. *In particular, spike-ins are of limited use in droplet-based protocols, as spike-ins can only be added to the reagent mixture in a known concentration, and the exact quantity in each droplet necessarily remains unknown [@Bacher2016].*

Reviewer comment:

3) Methods, 2nd paragraph: “Mean parameters μi quantify the overall expression for each gene i across the cell population under study.”
Comment: Please define cell population as sample (all cells within a scRNA-seq assay) or post-processing UMAP, t-SNE, or similar clustering grouping. I understood the latter, but readers should be certain, and we need to understand which step these parameters refer to.
Authors’ response:
In order to clarify this, we have edited the Introduction to describe the experimental designs in which the BASiCS model can be applied:
*The model was initially motivated by supervised experimental designs in which experimental conditions correspond to groups of cells defined *a priori* (e.g. selected cell types obtained through FACS sorting). However, the approach can also be used in cases where the groups of cells of interest are computationally identified through clustering. In such cases, the model does not currently address issues associated to *post-selection inference*, where the same data is analysed twice: first to perform clustering and then to compare expression profiles between the clusters [@Lahnemann2020]; this is an inherent limitation of most differential expression tools.*
Furthermore, we have now included the following text in the Methods, “BASiCS - Bayesian Analysis of Single Cell Sequencing data” subsection:
… within a cell population or populations under study. In this context, cell populations could correspond to groups set a priori by the experimental design (e.g. naive or stimulated CD4+ T cells in [@Martinez-jimenez2017]), or to groups of cells that were computationally identified through clustering.

Reviewer comment:
4) After Figure 5: “These thresholds can vary across datasets and should be informed by gene-specific QC metrics such as those shown in Figure 5 as well as prior knowledge about the cell types and conditions being studied, where available”.
Comment: I understand this is not the intent of this study, but if possible include some rationale behind thresholds for QCing gene exclusion based on types and conditions. Maybe bringing the choice for the example explored here.
Authors’ response:
We have removed the comments about choosing thresholds based on types and conditions, as we feel it does not reflect the true use of these thresholds for our use case. We thank the reviewer for the note, and have replaced this an explanation of our motivation:
This is to ensure a reliable estimate of variability, and is roughly in line with the sample size requirements for the negative binomial distribution outlined in @Lloyd-Smith2007.

Reviewer comment:
5) After spike-in calculation step: “To update the sce_naive and sce_active objects, the user must create a data.frame whose first column contains the spike-in labels (e.g. ERCC-00130) and whose second column contains the number of molecules calculated above. We add this as row metadata for altExp (sce_naive) and altExp (sce_active).”
Comment: By ‘update’, do the authors mean ‘update after normalisation with spike-in’? If so, please specify.
Authors’ response:
Apologies for the confusion. By “update” in this context, we refer to adding the relevant information about spike-ins to the sce_naive and sce_active objects. We note that the spike-

in analysis workflow has been moved to the Zenodo repository.

Reviewer comment:
7) MCMC diagnostics
Comment: I am by far not an expert on the analysis, but the authors were very successful in explaining the protocol in a stepwise manner. I would kindly ask if they can add the reason why this is needed: "to ensure that comparisons of gene expression are not random" or something in that direction. Readers will appreciate it.
Authors' response:
We agree that it is important to explain the rationale behind these diagnostics; we have update the text to reflect this as follows:
Before interpreting the estimates generated by `r Biocpkg("BASiCS")`, it is important to check the performance of the MCMC algorithm used. This algorithm used to characterise the posterior distribution is stochastic by nature, and as such its performance may vary even when used on the same dataset using the same settings. We perform quality control of the MCMC algorithm in order to ensure that the results we observe are the result of properties of the underlying data, rather than being an artefact of the stochastic algorithm used to characterise the data.

Reviewer comment:
8) Figures 15 and 16
Comment: These explanations are important to establish a convincing argument for using this workflow. Authors can explore further the 'tight regulation' versus 'transcriptional burst' or 'sub-population structure' scenarios with other examples (either from the literature or running an orthogonal analysis on larger scRNA-seq datasets of CD4+ T cells.
Authors' response:
The various sources of transcriptional variability are indeed essential to justifying the use of this workflow. We note that previous work has highlighted these different aspects in detail. For example, Eling et al 2018 explored the role of population structure by generating artificially contaminated datasets. Similarly, Martinez-Jimenez 2017 showed the change in transcriptional bursting patterns relating to ageing, where older mice were found to have more stochastic gene expression patterns relative to younger mice. To clarify this point, we have briefly discussed this in the introduction:
For example, @Eling2018 computationally explored the impact of unobserved structural heterogeneity by generating artificially contaminated datasets. In contrast, @Martinez-Jiminenez2017 showed that changes in transcriptional heterogeneity can related to aging, where older mice were shown to exhibit a more "bursting" or stochastic transcriptional pattern relative to younger mice.

Competing Interests: No competing interests were disclosed.

Reviewer Report 01 April 2022

https://doi.org/10.5256/f1000research.78169.r126523

?

Andrew McDavid
Department of Biostatistics and Computational Biology, University of Rochester, Rochester, NY, USA

This is an extensive workflow on applying the authors' BASiCS method to a plate-based single-cell RNAseq data. The authors discuss preparing data downloaded from an external repository, evaluation of QC metrics, interrogation of convergence of a Bayesian model, interpretation of model parameters, and comparative tests in the two sample settings. Code to generate an expansive array of figures is provided.

All said, it's a comprehensive workflow demoing well-documented software. I do have a few concerns about how other users would apply the workflow to their data:

1. The authors demo their method on some naive CD4 cells from inbred mice - really the ideal data set to illustrate a method looking at intrinsic sources of gene heterogeneity or homogeneity. How would prospective users of BASiCS adapt it to more typical datasets from droplet-based scRNAseq, either from samples with more heterogeneity (human) or of less purified cells, where spike-ins are absent and the "horizontal integration" model must be invoked? I am left wondering:
    - a. What sort of QC would be needed?
    - b. How should the data be stratified (into fairly fine-grained cell types on the basis of unsupervised clustering of the expression vector? Though this sort of double-dipping will change the inferential properties of the model).
    - c. How can violations of the Bayesian model be detected, or of unresolved sub-population heterogeneity, indicating more clustering is needed?
    - d. Given that with K populations, we'd have K-times the output from a single run of BASiCS to interpret. How to prioritize or understand it? (Perhaps an approach like muscat uses here would be helpful?[1]).
    - e. What techniques can aid in fitting this computationally-intensive model on larger datasets (e.g. thousands of cells)? Should we downsample cells, or apply more aggressive pre-filtering of genes? I see there's some provision to apply a "divide and conquer strategy" to fit the model on tranches of the data in parallel. How should this be used in practice?

2. The workflow is quite lengthy, enough that I found it hard to follow at times, and that I might be missing the forest for the trees. It would be nice if it could be condensed to emphasize model interpretation. A couple of suggestions on this end:
    - a. The initial vignette about data download and QC could be moved to an appendix, since it's not very specific to the BASiCS model, and there is a lot of boilerplate code, e.g. converting gene symbols. Then we could start from a binary SingleCellExperiment object.
    - b. The plotting code is quite verbose. It would be nice from a usability standpoint to have commonly-used plots included as part of the functionality of the package.
    - c. Is there some way to provide better navigation among sections of the workflow

(this may not be possible via F1000?)? This combined with a paragraph in the introduction ("This case study is organized as follows...section 3.1 describes downloading data from ArrayExpression, 3.2 annotates gene symbols, 3.3 examines QC,...") would make it easier for a reader to find the relevant section of the workflow.
    - d. Or the version of this workflow that's at https://github.com/VallejosGroup/BASiCSWorkflow/ could be rendered and made available on Github, perhaps with some nice markdown features like code-folding and a floating TOC to make it easier to read.

3. The comparison between the model estimates and the scran estimates in Figure 8 does help orient someone who is not completely familiar with the BASiCS model, but it would be nice to have a brief recap of the model in the Methods section (one or two equations), with interpretations, to explain the key parameters being estimated.

Minor comments:

1. I encountered some issues regarding the formatting of the code, which the other reviewer also remarked upon. This may be limited to the html version of the article.
    - a. newlines in `website` strings resulting in mal-formed URL.
    - b. ^ rendered in some odd font that wouldn't execute when I copied a chunk into Rstudio.
    - c. concentration.in..Mix.1.attomoles.ul was missing an `.`

2. The built-in MCMC diagnostics are nice, but I do feel like Rhat (testing within vs between-chain variance, e.g. Vehtari *et al.,* 2021[2]) is maybe what I want most. Is there any provision for running parallel chains?

3. It seemed that when I ran the `BASiCS_DetectHVG` function as specified, I got a couple of warning messages: "The posterior probability threshold chosen via EFDR calibration is too low. Probability threshold automatically set equal to 'ProbThresholdM'."

4. Regarding differential expression testing, could the authors comment on the modeling assumptions about the two "populations" of cells? If I had biological replicates in some conditions (e.g. distinct mice, humans, or plots of stimulated cells), how would they be used here, or does the framework require assuming null inter-replicate variability?

5. Figure 15: Caption says "log2 change in expression against against log mean expression for genes with higher residual over-dispersion in naive (A) cells and active (D) cells", but seems to refer to panels A and C. More broadly, it's not really clear what sort of conclusions we are supposed to draw from this sort of figure.

References

1. Crowell H, Soneson C, Germain P, Calini D, et al.: muscat detects subpopulation-specific state transitions from multi-sample multi-condition single-cell transcriptomics data. *Nature Communications*. 2020; **11** (1). Publisher Full Text
2. Vehtari A, Gelman A, Simpson D, Carpenter B, et al.: Rank-Normalization, Folding, and Localization: An Improved Rˆ for Assessing Convergence of MCMC (with Discussion). *Bayesian Analysis*. 2021; **16** (2). Publisher Full Text

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Partly

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Bioinformatics, biostatistics, single-cell transcriptomics

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 29 Jan 2024

Alan O'Callaghan

We thank the reviewer for their positive assessment of our manuscript, and the detailed comments, which we have used to revise and improve the workflow. Edits to the main text are highlighted in *italics*.

Reviewer comment:
The authors demo their method on some naive CD4 cells from inbred mice - really the ideal data set to illustrate a method looking at intrinsic sources of gene heterogeneity or homogeneity. How would prospective users of BASiCS adapt it to more typical datasets from droplet-based scRNAseq, either from samples with more heterogeneity (human) or of less purified cells, where spike-ins are absent and the "horizontal integration" model must be invoked? I am left wondering:

a. What sort of QC would be needed?

b. How should the data be stratified (into fairly fine-grained cell types on the basis of unsupervised clustering of the expression vector? Though this sort of double-dipping will change the inferential properties of the model).

c. How can violations of the Bayesian model be detected, or of unresolved sub-population heterogeneity, indicating more clustering is needed?

d. Given that with K populations, we'd have K-times the output from a single run of BASiCS to interpret. How to prioritize or understand it? (Perhaps an approach like muscat uses here would be helpful?1).

e. What techniques can aid in fitting this computationally-intensive model on larger datasets (e.g. thousands of cells)? Should we downsample cells, or apply more aggressive pre-filtering of genes? I see there's some provision to apply a "divide and conquer strategy" to fit the model on tranches of the data in parallel. How should this be used in practice?

Authors' response:

1. We thank the reviewer for this note. Due to its popularity, we have therefore modified the workflow to examine droplet-based data instead. The original analysis of CD4 T cells is now provided in an additional repository hosted on Zenodo. Furthermore, to improve the flow of the manuscript, we have removed data pre-processing and quality control to the same Zenodo repository, as we feel that this topic has been covered in detail by previous work (Lun *et al.* 2016, https://f1000research.com/articles/5-2122, Amezquita *et al.* 2019 https://doi.org/10.1038/s41592-019-0654-x).

1. Typical droplet-based datasets exhibit substantial cell-to-cell heterogeneity linked to the presence of distinct cell types or states. This needs to be accounted for before running the BASiCS model. In practice, this means that users should identify similar groups of cells (e.g. cell types) through clustering. As mentioned by Dr McDavid, this introduces a double-dipping effect where the same data is used twice: first to perform clustering and then to compare expression profiles between clusters. This effect is not yet addressed in BASiCS, but we believe it is an important area for future work.

We have now edited the Introduction in order to clarify this:
*The model was initially motivated by supervised experimental designs in which experimental conditions correspond to groups of cells defined *a priori* (e.g. selected cell types obtained through FACS sorting). However, the approach can also be used in cases where the groups of cells of interest are computationally identified through clustering. In such cases, the model does not currently address issues associated to *post-selection inference*, where the same data is analysed twice: first to perform clustering and then to compare expression profiles between the clusters [@Lahnemann2020]; this is an inherent limitation of most differential expression tools.*
Furthermore, the Discussion now highlights the importance of assessing how post-selection inference may affect the performance of BASiCS (which is beyond the scope of this manuscript):
One area for future work is to study how post-selection inference may affect the performance of `r Biocpkg("BASiCS")`. For example, @Neufeld2022 have developed a framework for addressing inflated type I errors when studying differences in means between groups of cells identified via hierarchical clustering, by splitting counts into "train" and "test" sets.

1. Ideally, structural heterogeneity associated with the presence of distinct cell types will be addressed before applying the BASiCS model either experimentally (via cell sorting) or computationally (via clustering). Eling et al, https://doi.org/10.1016/j.cels.2018.06.011, explored a case in which "contamination" was present, with a subpopulation of cells that had a distinct expression profile. We have observed in previous studies that highly variable genes were enriched by

markers of such subpopulations. We have added a brief note explaining this in the introduction:

These residual over-dispersion values can be used to quantify transcriptional heterogeneity without any confounding with mean expression, though it should be noted that high residual over-dispersion may also arise due to structural heterogeneity (e.g. distinct cell sub-types or states).

1. If the data consists of K pre-specified cell groups (e.g. types or states), BASiCS can be run separately within each group (as such, omputation can be trivially parallelised if multiple cores are available). Once the algorithm has been run, the MCMC chains can be post-processed to perform e.g. differential testing.

Currently, our decision rules for differential expression are performed using two groups. Therefore, it is more suitable to focused situations in which a handful of pairwise comparisons (hypothesis) are prioritised a priori. However, in principle the posterior distributions for gene-level parameters could be combined across multiple chains to test other hypotheses, such as an omnibus-type hypothesis (similar to the ANOVA F-test). However, this is outside the scope of this workflow.

1. We recently obtained very promising results about the use of more scalable algorithms for Bayesian inference (e.g. divide-and-conquer MCMC, see Bardenet et al. 2015 https://arxiv.org/abs/1505.02827). We plan to demonstrate the properties and practical applications of computationally scalable inference in a future manuscript (in preparation). We plan to update this workflow once our benchmarks for this new, more scalable, implementation are completed and the updated BASiCS package is available in Bioconductor. However, for the time being, we have added a brief note regarding this.

Reviewer comment:

The workflow is quite lengthy, enough that I found it hard to follow at times, and that I might be missing the forest for the trees. It would be nice if it could be condensed to emphasise model interpretation. A couple of suggestions on this end:

a. The initial vignette about data download and QC could be moved to an appendix, since it's not very specific to the BASiCS model, and there is a lot of boilerplate code, e.g. converting gene symbols. Then we could start from a binary SingleCellExperiment object.

b. The plotting code is quite verbose. It would be nice from a usability standpoint to have commonly-used plots included as part of the functionality of the package.

c. Is there some way to provide better navigation among sections of the workflow (this may not be possible via F1000?)? This combined with a paragraph in the introduction ("This case study is organised as follows...section 3.1 describes downloading data from ArrayExpression, 3.2 annotates gene symbols, 3.3 examines QC,...") would make it easier for a reader to find the relevant section of the workflow.

d. Or the version of this workflow that's at https://github.com/VallejosGroup/BASiCSWorkflow/ could be rendered and made available on Github, perhaps with some nice markdown features like code-folding and a floating TOC to make it easier to read.

Authors' response:

1. Many thanks for this suggestion. We have now moved the data preparation steps to

a separate repository for clarity, as single cell data pre-processing and quality control have been covered in detail by previous work (Lun *et al.* 2016, https://f1000research.com/articles/5-2122, Amezquita *et al.* 2019 https://doi.org/10.1038/s41592-019-0654-x).

2. We agree that the plotting code is quite verbose, and detracts from the flow of the manuscript. We have moved the pre-processing code to a separate repository for clarity, and where possible have simplified the code to improve the reading experience.
3. We have included a short summary of the workflow sections in the introduction, to hopefully aid in navigation.
4. Following peer review, we intend to submit a version of this workflow to Bioconductor as a Bioconductor workflow (https://www.bioconductor.org/packages/release/BiocViews.html#___Workflow) so that it can be regularly built as part of the Bioconductor build system.

Reviewer comment: The comparison between the model estimates and the scran estimates in Figure 8 does help orient someone who is not completely familiar with the BASiCS model, but it would be nice to have a brief recap of the model in the Methods section (one or two equations), with interpretations, to explain the key parameters being estimated.

Authors' response:

Many thanks for this suggestion. We decided to provide a high level description of the model suitable for less technical audiences. However, we agree that further information would help to better interpret the use and output of BASiCS. Therefore, we have now updated the manuscript to include a graphical overview for the BASiCS models and the relevant parameters. This includes a graphical representation of the overall mean/overdispersion trend and how this is used to derive residual overdispersion estimates (as deviations with respect to the overall trend). We hope this helps to clarify the links with the DM estimates provided by scran.

Minor comments:

Reviewer comment:

I encountered some issues regarding the formatting of the code, which the other reviewer also remarked upon. This may be limited to the html version of the article.

a. newlines in `website` strings resulting in mal-formed URL.

b. ^ rendered in some odd font that wouldn't execute when I copied a chunk into Rstudio.

c. concentration.in..Mix.1.attomoles.ul was missing an `.`

Authors' response:

This is a formatting issue which should be resolved now.

Reviewer comment:

The built-in MCMC diagnostics are nice, but I do feel like Rhat (testing within vs between-chain variance, e.g. Vehtari et al., 20212) is maybe what I want most. Is there any provision for running parallel chains?

Authors' response:

Due to the running times of the current MCMC sampler, we have not suggested users to routinely run multiple MCMC chains as part of their analyses. However, throughout the years, we have run BASiCS multiple times in a variety of settings and we have generally observed that different runs of the algorithm converge to similar values. Whilst this does not guarantee that the same would hold for a new dataset, our experience strongly suggests that the proposed diagnostic criteria are sufficient when evaluating MCMC convergence for BASiCS. Nevertheless, we have added support for the Rhat diagnostic criterion, and acknowledge that running multiple chains may be useful and we will aim to implement this in future versions of the BASiCS library. In particular, we recently obtained very promising results about the use of more scalable algorithms for Bayesian inference (e.g. divide-and-conquer MCMC, see Bardenet et al. 2015 https://arxiv.org/abs/1505.0282; manuscript in preparation). This will facilitate running the algorithm multiple times in a timely manner. We plan to update this workflow once our benchmarks for this new, more scalable, implementation are completed and the updated BASiCS package is available in Bioconductor.

Reviewer comment:
It seemed that when I ran the `BASiCS_DetectHVG` function as specified, I got a couple of warning messages: "The posterior probability threshold chosen via EFDR calibration is too low. Probability threshold automatically set equal to 'ProbThresholdM'."
Authors' response:
This message relates to the choice of a probability threshold that controls the EFDR at the chosen level. This is normal behaviour that can be controlled using the ProbThreshold argument, and we have explained it using the following note in the text:
Performing HVG and LVG detection involves identifying a posterior probability threshold that matches a target EFDR. We perform this choice using a grid search in which EFDR is calculated for a range of different probability thresholds between 0.5 and 1. As seen in Figure \@ref(fig:efdr-plot-vg-SM), for the somitic mesoderm cells, this leads to a threshold of `r hvg@ProbThreshold` and `r lvg@ProbThreshold` for HVG and LVG detection, respectively (the value of these thresholds can be extracted from the `ProbThreshold` slot in the `hvg` and `lvg` objects, e.g. using `hvg@ProbThreshold`). For some datasets, the grid search may fail to identify a probability threshold that matches the target EFDR. In such cases, the default minimum probability threshold of $\frac{2}{3}$ is used. This default threshold value can be changed using the `ProbThreshold` to `BASiCS_DetectHVG` and `BASiCS_DetectLVG`.

Reviewer comment:
Regarding differential expression testing, could the authors comment on the modelling assumptions about the two "populations" of cells? If I had biological replicates in some conditions (e.g. distinct mice, humans, or plots of stimulated cells), how would they be used here, or does the framework require assuming null inter-replicate variability?
Authors' response:
In principle, the differential expression testing approach could be used to compare expression profiles between two arbitrary groups of cells. The typical use-case is two cell types or experimental conditions for a given cell type (naive vs stimulated, grown in two different media). Users could potentially apply the model to different biological replicates. In such cases, our over-dispersion parameters will quantify intra-replicate cell-to-cell variability and our differential expression test will quantify systematic differences between

biological replicates. However, this approach does not share information across biological replicates when estimating model parameters (this can be useful in situations where a small number of cells is available per biological replicate, or when the data is sparse). Moreover, this would not scale well to cohort studies (e.g. van der Wijst et al. 2018, https://doi.org/10.1038/s41588-018-0089-9) where multiple biological replicates are present. We have added a brief note about this extension in the Discussion section.
Finally, we also anticipate potential extensions of `r Biocpkg("BASiCS")` to account for the more complex experimental designs. For example, in cohort studies where cells are extracted from multiple individuals, the hierarchical model could be expanded to quantify both for intra- and inter-individual transcriptional variability.

Reviewer comment:
Figure 15: Caption says "log2 change in expression against against log mean expression for genes with higher residual over-dispersion in naive (A) cells and active (D) cells", but seems to refer to panels A and C. More broadly, it's not really clear what sort of conclusions we are supposed to draw from this sort of figure.
Authors' response:
We agree that this figure was somewhat unclear, and we apologise for the mis-labelled panels. We have removed this figure from the revised manuscript as we feel it was not necessary.

Competing Interests: No competing interests were disclosed.

Reviewer Report 08 March 2022

https://doi.org/10.5256/f1000research.78169.r124656

?

Oliver M. Crook
Department of Statistics, University of Oxford, Oxford, UK

Summary:

O'Callaghan and co-authors provide a workflow for the popular method BASiCS and its extensions. This workflow is intended to display the bioconductor infrastructure of BASiCS and provide a template for analysis for others to streamline their data analysis. I find the workflow quite comprehensive and takes me through the analysis points that I would expect to see. First of all, I must confess I am not an expert in processing scRNA data and so I cannot assess whether the choices of filtering, thresholding etc are up-to-date. I will leave this assessment to other reviewers. Here, I wish to focus my review on the communication of sophisticated methods presented, the Bayesian analysis, visualizations and the code. I believe this is an extremely valuable contribution to the scRNA community, the workflow community and applied Bayesians. I would also like to

thank the authors' attention to detail, I found the article well-written and logically structured.

I would encourage you to update your version of R to 4.2 as the next version of Bioconductor will use this.

General comments:

Whilst the method itself is presented technically in other papers, I don't quite get a good picture of the model and its variants. Referring to the other papers I find myself confronted with lots of equations, whilst I can make my way through these details I think the intended users might struggle. The workflow might benefit from a narrative description or recipe that highlights its generative nature. A cartoon or graphic can be particularly insightful.

As good statisticians, we should incorporate experimental design into our models. I think before the data are introduced it would be useful to have a short description of the type of experiments that could be analysed with BASiCS, when BASiCS would be useful over other approaches, when not to apply BASiCS. What sort of data would it be optimal to collect if I wish to analysed it using BASiCS? How valuable are spike-ins for example?

I had to spend quite some time on the code chunks, working out what was going on. I think these could be decorated with useful comments so your intended user is allowed to be as lazy as possible (which is what you want so they will use your package!). This was particularly the case for the pre-processing and parameter estimation.

There are some quite long code chunks that I felt represented fairly repetitive tasks, mostly related to plotting. If you're using ggplot2 then you can write a function which returns a ggplot object with nice defaults and you can explain that it can be customized using the ggplot principles. I think the current style is written to allow this customization but I think it's currently unexplained and perhaps working against you.

In the section of parameter estimation, the discussion on priors is quite brief and quite unclear currently. I'm not quite sure what the "joint prior" is here, I think your intended audience would appreciate a narrative description of these priors. The other choices of independent log Normal is also unclear. The logMean and logSd appear to be set automatically using the data - is that what you mean by Empirical Bayes here? When I dive deeper into changing the priors BASiCS_PriorParam is quite daunting. I think there are some helpful things you can do for your readers. For example, you explain whether the analysis is sensitive to the prior choice and generally how they affect the analysis. You could guide your users through a prior predictive check, in which you simulate parameters from the prior and then fake data from the likelihood. You could then show users how well calibrated their prior choices are. General advice of setting the prior or whether they should just leave these alone would be useful.

I assume that multiple chains are run, are there any MCMC diagnostics based on running parallel chains?

In a Bayesian analysis I expected to be able to manipulate posterior distributions or credible intervals. Maybe I missed it but can we visualize the full posterior distribution for \mu_i vs \delta_i and \mu_i vs \epsilon_i or some other parameters of interest?

I think I haven't quite grasped the "variance decomposition" the authors mention into biological and technical components. This could be \delta_i and \epsilon_i, but I feel I am missing something? I do not yet think the methods section is quite clear on this point.

For the volcano plots the posterior probability is quite bunched up, would this be better on the logit scale? Also It appears the probabilities concentrate around 0.5? Is there an intuitive explanation for this? Indeed, the distribution generated by hist(p2$data$ProbDiffMean) was quite unexpected, are these probabilities calibrated in any way?

You mention computational challenges in the discussion, what is lost by performing maximum likelihood estimation or maximum a posteriori estimation? Perhaps the weighted likelihood bootstrap or variant could be a useful approach, if you have lots of cores.

What are the challenges of applying this to scRNA data with spatial autocorrelation? It is common to now obtain expression data in such a context, does this additional confounding obstruct the model?

Specific comments:

In the first code chunks, in `website <- " https ..." I think there is an initial space which stops this from running correctly.

Remember to include spaces after commas e.g. [a,] rather than [a,], it is much clearer to read.

I think it would be better to load all the packages at the beginning, I had to restart my session several times to install the packages that I realized I needed x% of the way through the workflow.

In figure 4, the axis dimensions of the PCA misrepresents the variance explained. The ratio of axis dimensions should closely match the ratios of the variance explained so that the visualization is faithful to the dimensionality reduction.

Constants used in the package could be saved as global variables, it's very easy to make a typo with these conversions.

In the chunk starting ## Moles per microliter, I had to edit the first line of the code chunk to get it to work. Perhaps it has been malformatted?

\log is a symbol and should always be lower-case

In the abstract, you write "strong technical noise". I don't quite understand what "strong" is adding here.

A number of concepts are left unexplained such as "joint prior", "negative binomial framework", "borrows information", "expected false discovery rate". These are all familiar concepts to someone with advanced statistical knowledge but the casual scRNA reader might find this off putting.

Figure 12 there is a typo in the legend "\log_{1}0 -> \log_{10}".

In Figure 12 the dendrogram is unexplained and I'm not sure what it means in this context.

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: I receive funding from GlaxoSmithKline.

Reviewer Expertise: Biostatistics, Bayesian methodology, R, data-intensive biology

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 29 Jan 2024

Alan O'Callaghan

We would like to thank Dr Crook for such a positive assessment of our manuscript. A detailed point-by-point response for the remaining comments is provided below. Edits to the main text are highlighted in *italics*.

Reviewer comment: I would encourage you to update your version of R to 4.2 as the next version of Bioconductor will use this.
Author's response: We have now updated the workflow to use R version 4.3 which is the current version suggested by Bioconductor.

Reviewer comment: Whilst the method itself is presented technically in other papers, I don't quite get a good picture of the model and its variants. Referring to the other papers I find myself confronted with lots of equations, whilst I can make my way through these details I think the intended users might struggle. The workflow might benefit from a narrative description or recipe that highlights its generative nature. A cartoon or graphic

can be particularly insightful.

Author's response: Many thanks for this suggestion. We have now updated the manuscript to include a graphical overview for the BASiCS model (see Figure 2 in the revised manuscript). This enables us to better introduce the role of different model parameters. In terms of the different variations of the BASiCS model, we provide a high level description for the latest version (as per Eling et al 2018) as this is what we recommend to use as default. We hope this provides a better overview of the methodology whilst ensuring that the workflow is accessible to a wide range of users.

Reviewer comment: As good statisticians, we should incorporate experimental design into our models. I think before the data are introduced it would be useful to have a short description of the type of experiments that could be analysed with BASiCS, when BASiCS would be useful over other approaches, when not to apply BASiCS. What sort of data would it be optimal to collect if I wish to analysed it using BASiCS? How valuable are spike-ins for example?
Author's response: We agree a more explicit description of the experimental design assumed by BASiCS would be useful to potential issues. To address this, we included the following text in the Introduction:
*The model was initially motivated by supervised experimental designs in which experimental conditions correspond to groups of cells defined *a priori* (e.g. selected cell types obtained through FACS sorting). However, the approach can also be used in cases where the groups of cells of interest are computationally identified through clustering. In such cases, the model does not currently address issues associated to *post-selection inference*, where the same data is analysed twice: first to perform clustering and then to compare expression profiles between the clusters [@Lahnemann2020]; this is an inherent limitation of most differential expression tools.*

Furthermore, the Discussion now highlights the importance of assessing how post-selection inference may affect the performance of BASiCS (which is beyond the scope of this manuscript):
One area for future work is to study how post-selection inference may affect the performance of `r Biocpkg("BASiCS")`. For example, @Neufeld2022 have developed a framework for addressing inflated type I errors when studying differences in means between groups of cells identified via hierarchical clustering, by splitting counts into "train" and "test" sets.

Reviewer comment: I had to spend quite some time on the code chunks, working out what was going on. I think these could be decorated with useful comments so your intended user is allowed to be as lazy as possible (which is what you want so they will use your package!). This was particularly the case for the pre-processing and parameter estimation.
Author's response: Indeed, clarity is an important element for this type of analysis workflows. We have now added further comments in the code chunks. We hope this helps to clarify specific steps. Furthermore, we have moved the code dealing with data QC to a separate repository, as the topic has been covered in previous F1000Research workflow articles (e.g. Lun et al 2016, https://f1000research.com/articles/5-2122). We hope these changes have reduced the complexity of the code and improve the flow of our article.

Reviewer comment: There are some quite long code chunks that I felt represented fairly

repetitive tasks, mostly related to plotting. If you're using ggplot2 then you can write a function which returns a ggplot object with nice defaults and you can explain that it can be customised using the ggplot principles. I think the current style is written to allow this customization but I think it's currently unexplained and perhaps working against you.
Author's response: As mentioned above, we have moved the code dealing with data QC to a separate repository, which should reduce the complexity of the code that readers need to process. We have also attempted to reduce the complexity of later plots by splitting them into smaller chunks with better explanation, which we hope aids the readability of the manuscript.

Reviewer comment: In the section of parameter estimation, the discussion on priors is quite brief and quite unclear currently. I'm not quite sure what the "joint prior" is here, I think your intended audience would appreciate a narrative description of these priors. The other choices of independent log Normal is also unclear. The logMean and logSd appear to be set automatically using the data - is that what you mean by Empirical Bayes here? When I dive deeper into changing the priors BASiCS_PriorParam is quite daunting. I think there are some helpful things you can do for your readers. For example, you explain whether the analysis is sensitive to the prior choice and generally how they affect the analysis. You could guide your users through a prior predictive check, in which you simulate parameters from the prior and then fake data from the likelihood. You could then show users how well calibrated their prior choices are. General advice of setting the prior or whether they should just leave these alone would be useful.
Authors' response: We agree that further information/intuition about the priors will improve our manuscript. To address this, we have now extended the model description. As noted above, we have primarily focused on the latest version of the model (Eling et al 2018) as that is the default recommended setting. In particular, we have now provided additional information to clarify the definition of our "joint prior" and the use of an empirical Bayes approach. This includes the bottom right panel in Figure 2 and edits to the "BASiCS - Bayesian Analysis of Single Cell Sequencing data" sub-section within Methods:
*To account for the strong relationship that is typically observed between gene-specific mean expression and over-dispersion estimates, Eling *et al.* [@Eling2018] introduced a *joint prior* specification for these parameters. This joint prior assumes that genes with similar mean expression (μ_i) have similar over-dispersion parameters δ_i. Effectively, this shrinks over-dispersion estimates towards a global trend that captures the relationship between mean and over-dispersion (Figure 2). This improves posterior inference for over-dispersion parameters when the data is less informative (e.g. small sample size, lowly expressed genes) [@Eling2018]. This information-sharing approach is conceptually similar to that performed by @Love2014 and others, where sparse data is pooled to obtain more reliable estimates. The global trend is then used to derive gene-specific *residual over-dispersion* parameters ϵ_i that are not confounded by mean expression. Similar to the DM values implemented in `r Biocpkg("scran")`, these are defined as deviations with respect to the overall trend.*
In terms of the prior hyper-parameter choices, we recommend users to apply the default settings as we have tested these in a variety of contexts. Furthermore, as shown in Vallejos et al (2015), most posterior estimates are robust to different hyper-parameter choices (the exception is for gene-specific parameters for lowly expressed genes, which benefit from the use of the joint prior introduced by Eling et al 2018 https://doi.org/10.1016/j.cels.2018.06.011 and the empirical Bayes framework). Finally, as

shown by Zappia et al (2017) https://doi.org/10.1186/s13059-017-1305-0, the BASiCS model is able to successfully recapitulate the main properties of typical scRNAseq data when used as a generative model (posterior predictive checks). To emphasise this, we have added the following text to the manuscript:
A previous study has shown that BASiCS, when used as a generative model, is well-tuned to capture and recapitulate the main properties of typical scRNAseq data using posterior predictive checks [@Zappia2017].

Reviewer comment: I assume that multiple chains are run, are there any MCMC diagnostics based on running parallel chains?
Authors' response: Due to the running times of the current MCMC sampler, we have not suggested users to routinely run multiple MCMC chains as part of their analyses. However, throughout the years, we have run BASiCS multiple times in a variety of settings and we have generally observed that different runs of the algorithm converge to similar values. Whilst this does not guarantee that the same would hold for a new dataset, our experience strongly suggests that the proposed diagnostic criteria are sufficient when evaluating MCMC convergence for BASiCS. Nevertheless, we acknowledge that running multiple chains may be useful and we will aim to implement this in future versions of the BASiCS library. In particular, we recently obtained very promising results about the use of more scalable algorithms for Bayesian inference (e.g. divide-and-conquer MCMC, see Bardenet et al. 2015 https://arxiv.org/abs/1505.02827; manuscript in preparation). This will facilitate running the algorithm multiple times in a timely manner. We plan to update this workflow once our benchmarks for this new, more scalable, implementation are completed and the updated BASiCS package is available in Bioconductor. We also provide the facility for users to run multiple chains and to manipulate posteriors manually using rstan (https://bioconductor.org/packages/release/bioc/html/BASiCStan.html), although we note that for data of any appreciable size the Hamiltonian Monte Carlo algorithm used by Stan becomes computationally intractable. We have added a brief note to the discussion regarding this, as follows:
*Given that `r Biocpkg("BASiCS")` requires computationally intensive MCMC sampling to estimate the posterior distribution, it becomes computationally intractable to analyse data from very large numbers of cells. Alternative inference schemes such as variational inference for maximum *a posteriori* estimation, or Hamiltonian Monte Carlo may be useful in this context. Advanced users may wish to use the `r Biocpkg("BASiCStan")` package to test these alternative inference schemes. This also provides access to the model diagnostics, facilities for running multiple chains, and posterior summaries provided by Stan [@Carpenter2017], while also being fully compatible with the workflow described in this manuscript via the function `Stan2BASiCS`, that converts the output of the Stan inference procedure to the type of output generated by `BASiCS_MCMC`. However, we note that the Hamiltonian Monte Carlo inference method provided by Stan is more computationally intensive than our default implementation. Furthermore, the faster approximations provided by Stan, namely scalable variational inference and maximum *a posteriori* estimation, are often unstable and less accurate.*

Reviewer comment: In a Bayesian analysis I expected to be able to manipulate posterior distributions or credible intervals. Maybe I missed it but can we visualize the full posterior distribution for \mu_i vs \delta_i and \mu_i vs \epsilon_i or some other parameters of interest?

Authors' response: As the parameter space for the BASiCS model is high-dimensional, it is not practical to visualise the joint posterior distribution posterior of all parameters. However, displaySummaryBASiCS() can be used to obtain point estimates (posterior medians) and the 95% Highest Posterior Density (HPD) intervals for individual parameters. Furthermore, the displayChainBASiCS() function can be used to extract the MCMC chains which can be then manipulated separately by the user. To clarify this, we have added the following sentence to the "Parameter estimation using BASiCS section".
The matrices of MCMC draws can be accessed using the `displayChainBASiCS` function --- this may be useful for estimating and visualising credible intervals using packages like `r CRANpkg("bayesplot")` or `r CRANpkg("tidybayes")`.

Reviewer comment: I think I haven't quite grasped the "variance decomposition" the authors mention into biological and technical components. This could be \delta_i and \epsilon_i, but I feel I am missing something? I do not yet think the methods section is quite clear on this point.
Authors' response: We used the term "variance decomposition" to denote BASiCS' ability to separate the observed variability into a technical and a biological component, which then enables downstream analysis of transcriptional variability: detection of highly and lowly variable genes, and differential variability testing. We have thus removed the explicit use of the term "variance decomposition" from the paper, and we apologise for any confusion this caused.

Reviewer comment: For the volcano plots the posterior probability is quite bunched up, would this be better on the logit scale? Also It appears the probabilities concentrate around 0.5? Is there an intuitive explanation for this? Indeed, the distribution generated by hist(p2$data$ProbDiffMean) was quite unexpected, are these probabilities calibrated in any way?
Authors' response: Thanks for the suggestion. We have changed these plots to use an (adjusted) logit scale, and this does appear to be more informative in most cases. The logit scale is adjusted to avoid undefined values, i.e., logit(0) and logit(1) by adding or subtracting a small value from probabilities with undefined logit values.
In terms of calibration, this is difficult to assess in the absence of ground truth. However, as shown in Vallejos et al (2016), synthetic data examples showed good error rate performance for our differential expression test, provided that a non-zero log-fold change threshold is used (default in the BASiCS_TestDE function is shown by the vertical lines in the volcano plots). This is similar to the approach by McCarthy and Smith (2009) https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2654802/, avoiding the identification of genes with statistically significant differences but whose associated log-fold changes are not biologically significant.

Reviewer comment: You mention computational challenges in the discussion, what is lost by performing maximum likelihood estimation or maximum a posteriori estimation? Perhaps the weighted likelihood bootstrap or variant could be a useful approach, if you have lots of cores.
Authors' response:
We have explored the variational inference algorithm implemented in Stan as a fast method to obtain maximum *a posteriori* point estimates for BASiCS. However, we have generally

observed that inference is less reliable. Moreover, beyond point estimates, it is critical for BASiCS to obtain reliable estimates of posterior uncertainty as these are required for our differential test which relies on tail posterior probabilities. This is particularly problematic when using a mean-field approximation for the posterior, which is what is currently used in our Stan implementation.
As an alternative approach to address these computational challenges, we recently explored more scalable algorithms (e.g. divide-and-conquer MCMC, see Bardenet et al. 2015 https://arxiv.org/abs/1505.02827) to perform inference using the BASiCS model. Our results are promising and a manuscript is in preparation.

Reviewer comment: What are the challenges of applying this to scRNA data with spatial autocorrelation? It is common to now obtain expression data in such a context, does this additional confounding obstruct the model?
Authors' response:
The current BASiCS model assumes that cells are conditionally independent (given gene-specific and other global parameters). As a result our model does not account for the more complex structure that arises in spatial assays. Whilst we have not explored this yet, we hypothesise that ignoring this temporal correlation structure will likely lead to biassed over-dispersion estimates. We have now updated the Discussion to mention that the BASiCS model could be adapted to address more complex correlation structures across cells, such as those introduced by spatial assays and cohort studies in which cells are extracted from multiple donors (e.g. patients):
Finally, we also anticipate potential extensions of `r Biocpkg("BASiCS")` to account for the more complex experimental designs. For example, in cohort studies where cells are extracted from multiple individuals, the hierarchical model could be expanded to quantify both for intra- and inter-individual transcriptional variability. As spatial transcriptomic technologies mature [@Marx2021], Bayesian hierarchical models such as the one implemented in `r Biocpkg("BASiCS")` could also be designed to incorporate spatial (and, potentially, temporal) structure (e.g. [@Aijo2019]). We intend to update this workflow as the field evolves, and as we address the issues and challenges outlined here.

Specific comments:

Reviewer comment:
In the first code chunks, in `website <- " https ..." I think there is an initial space which stops this from running correctly.
Authors' response:
This is a formatting issue which should be resolved now.

Reviewer comment:
Remember to include spaces after commas e.g. [a,] rather than [a,], it is much clearer to read.
Authors' response:
Agreed, thank you for pointing this out; we have addressed this in our code formatting.

Reviewer comment:
I think it would be better to load all the packages at the beginning, I had to restart my

session several times to install the packages that I realized I needed x% of the way through the workflow.

Authors' response:
We initially loaded the packages when they were first used in order to emphasise their role within the analysis pipeline. However, we do acknowledge that some users may prefer to load all the packages from the start, to prevent installation issues during their analysis (note that installation issues are resolved when using the provided Docker image that contains all software requirements). To address this, all libraries are now loaded at the start of the analysis. Explanations for the use of each library are provided in relevant sections.

Reviewer comment:
In figure 4, the axis dimensions of the PCA misrepresents the variance explained. The ratio of axis dimensions should closely match the ratios of the variance explained so that the visualisation is faithful to the dimensionality reduction.

Authors' response:
Thank you for highlighting this. We have now swapped the x- and y-axes such that the larger axis represents the larger component of variation in the data, although the x-axis is still disproportionate given the small amount of variation represented by that PC. However given the difference in scale here (20% vs 1%, we feel that directly proportional scaling would be excessive.

Reviewer comment:
Constants used in the package could be saved as global variables, it's very easy to make a typo with these conversions.

Authors' response:
We have implemented a helper function, BASiCS_CalculateERCC, to handle this conversion process. We note that the spike-in analysis workflow has been moved to the Zenodo repository.

Reviewer comment:
In the chunk starting ## Moles per microliter, I had to edit the first line of the code chunk to get it to work. Perhaps it has been malformatted?

Authors' response:
This is another formatting error and should now be resolved. We note that the spike-in analysis workflow has been moved to the Zenodo repository.

Reviewer comment:
\log is a symbol and should always be lower-case

Authors' response:
This has been resolved.

Reviewer comment:
In the abstract, you write "strong technical noise". I don't quite understand what "strong" is adding here.

Authors' response:
The usage of "strong" aimed to emphasise that the level of technical noise was substantially

higher than what was observed in bulk-level assays (see [3]). To clarify this, we have modified the abstract as follows:
... but it is prone to technical noise, beyond what is observed in bulk-level assays.
Reviewer comment:
A number of concepts are left unexplained such as "joint prior", "negative binomial framework", "borrows information", "expected false discovery rate". These are all familiar concepts to someone with advanced statistical knowledge but the casual scRNA reader might find this off putting.

Authors' response:
This is an excellent suggestion. We have now edited the text to explain these concepts.
To explain the "negative binomial framework" we have added the following text to the Introduction:
The negative binomial distribution is commonly used to model count data when the observed variability exceeds what can be captured by a simpler Poisson model --- this is typically referred to as over-dispersion.
To explain the concept behind the "borrows information" statement, the first paragraph in the "BASiCS - Bayesian Analysis of Single Cell Sequencing data" sub-section within Methods has been edited as follows:
[...] instead of modelling expression patterns separately for each gene, `r Biocpkg("BASiCS")` shares information between all genes to robustly quantify transcriptional variability. For example, as described by [@Eling2018], pooling information across genes with similar mean expression levels helps to obtain more reliable transcriptional variability estimates. The latter is particularly helpful for lowly expressed genes and sparse datasets, where less information is available.

To clarify the definition of our "joint prior" statement, the third paragraph in the "BASiCS - Bayesian Analysis of Single Cell Sequencing data" sub-section within Methods has been edited to include further information:
*To account for the strong relationship that is typically observed between gene-specific mean expression and over-dispersion estimates, Eling *et al.* [@Eling2018] introduced a *joint prior* specification for these parameters. This joint prior assumes that genes with similar mean expression (μ_i) have similar over-dispersion parameters δ_i. Effectively, this shrinks over-dispersion estimates towards a global trend that captures the relationship between mean and over-dispersion (Figure \@ref(fig:basics-schematic)). This improves posterior inference for over-dispersion parameters when the data is less informative (e.g. small sample size, lowly expressed genes) [@Eling2018]. This information-sharing approach is conceptually similar to that performed by @Love2014 and others, where sparse data is pooled to obtain more reliable estimates. The global trend is then used to derive gene-specific *residual over-dispersion* parameters ϵ_i that are not confounded by mean expression. Similar to the DM values implemented in `r Biocpkg("scran")`, these are defined as deviations with respect to the overall trend.*

In the "HVG/LVG detection using BASiCS" sub-section within the case study, we have added the following text to explain the expected false discovery rate:
The expected false discovery rate we use is conceptually similar to false discovery rate control procedures in frequentist statistics, such as the approach of @Benjamini1995, aiming to control

the proportion of false discoveries produced by the procedure.
Reviewer comment:
Figure 12 there is a typo in the legend "\log_{1}0 -> \log_{10}".
Authors' response:
This has been resolved.

Reviewer comment:
In Figure 12 the dendrogram is unexplained and I'm not sure what it means in this context.
Authors' response:
This dendrogram shows the similarity between cells, but it is not required here and therefore has been removed.

Competing Interests: No competing interests were disclosed.