SOFTWARE TOOL ARTICLE

# REVISED BASiCS workflow: a step-by-step analysis of expression variability using single cell RNA sequencing data [version 2; peer review: 1 approved, 2 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]Institute for Molecular Health Sciences, ETH Zürich, Zürich, 8093, Switzerland
[3]Department of Quantitative Biomedicine, University of Zurich, Zürich, CH-8057, Switzerland
[4]Cancer Research UK Cambridge Institute, University of Cambridge, Cambridge, CB2 0RE, UK
[5]European Molecular Biology Laboratory, European Bioinformatics Institute, Cambridge, CB10 1SD, UK
[6]The Alan Turing Institute, The Alan Turing Institute, London, NW1 2DB, UK

**v2** **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.

**Open Peer Review**

**Approval Status** ✓ ? ?

| | 1 | 2 | 3 |
|---|---|---|---|
| **version 2** (revision) 07 May 2024 | ✓ view | | |
| **version 1** 18 Jan 2022 | ? view | ? view | ? view |

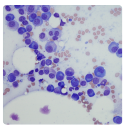
1. **Oliver M. Crook**, University of Oxford, Oxford, UK
2. **Andrew McDavid**, University of Rochester, Rochester, USA
3. **Michel S Naslavsky**, University of São Paulo, São Paulo, Brazil

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

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

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 2; peer review: 1 approved, 2 approved with reservations]** F1000Research 2024, **11**:59 https://doi.org/10.12688/f1000research.74416.2

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

**REVISED** **Amendments from Version 1**

As a result of helpful and insightful comments by the reviewers, the workflow has been substantially simplified and streamlined, while switching focus to more modern, droplet-based scRNAseq data. The code has been simplified wherever possible, removing content covered in greater detail in other documents, while still demonstrating all of the relevant information, tools and techniques for a robust analysis of scRNAseq data using BASiCS. We hope that this updated workflow provides the bioinformatics community with a useful reference for this application.

A detailed description of the changes to figures is given below:

As Figure 1, we have added a schematic describing the main relevant aspects of BASiCS.

Figure 2 is the workflow overview shown in the figure 1 of the previous workflow version.

Figures 3-16 all relate to different data relative to the original workflow. In particular, the previous version showed an analysis of naive and active T cells. In contrast, the updated workflow relates to an analysis of somitic and pre-somitic mesoderm cells.

Figures 2-5 have been removed, as they were not relevant to the flow of the workflow.

Figures 3-5 are equivalent to Figures 6-7 in the previous version, although relating to different underlying data.

Figure 6 shows the criteria used for selection of a posterior probability threshold that satisfies a target expected false discovery rate, and the related expected false negative rate, for selection of highly- and lowly-variable genes using BASiCS.

Figure 7 shows highly- and lowly- variable gene detection using BASiCS, showing posterior estimates of variability against posterior estimates of mean expression.

Figure 8 shows example expression profiles for highly- and lowly-variable genes with similar overall levels of expression.

Figure 9 shows mean-difference and volcano plots of a differential mean expression analysis of somitic vs pre-somitic mesoderm cells.

Figures 10-12 show normalised expression values for somitic and pre-somitic cells, of genes up-regulated in pre-somitic cells (Fig 10), genes up-regulated in somitic cells (Fig 11), and genes neither up- nor down-regulated in either population of cells (Fig 12).

Figure 13 shows mean-difference and volcano plots of a differential expression variability analysis of somitic vs pre-somitic mesoderm cells.

Figure 14 shows differences in residual over-dispersion plotted against changes in mean expression.

**Any further responses from the reviewers can be found at the end of the article**

## 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 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] 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).[18]

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[19]) can be added to each cell's lysate in a (theoretically) known fixed quantity. Spike-ins can assist quality control steps,[20] data normalisation[18] 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).[18] However, despite the benefits associated to the use of spike-ins and UMIs, these are not available for all scRNA-seq protocols.[22] 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.[23]

The Bioconductor package *BASiCS* implements a Bayesian hierarchical framework that accounts for both technical and biological sources of noise in scRNA-seq datasets.[24–26] 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[27]; this is an inherent limitation of most differential expression tools.

*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.[28–31]

The negative binomial distribution is commonly used to model count data when the observed variability differs from what can be captured by a simpler Poisson model — this is typically referred to as over-dispersion. 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.[26] The latter enables the quantification of a gene-level *residual over-dispersion* as a measure of transcriptional noise that is not confounded by differences in gene expression. Furthermore, when available, *BASiCS* can also leverage extrinsic spike-in molecules to aid data normalisation. 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.[32]

This article complements existing scRNA-seq workflows based on the Bioconductor ecosystem (e.g. Refs. 33, 34), providing a detailed framework for transcriptional variability analyses using *BASiCS*. We describe a step-by-step workflow that uses *scater*[20] and *scran*[33] 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.[26] To ensure the reproducibility of this workflow, the analysis environment and all software dependencies are provided as a Docker image.[35] The image can be obtained from Docker Hub, at https://hub.docker.com/r/alanocallaghan/basicsworkflow2020-docker.

## Methods

### Implementation

The *BASiCS* Bioconductor package uses a Bayesian hierarchical model to simultaneously perform data normalisation, technical noise quantification and downstream analyses[24–26] 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 Ref. 7), or to groups of cells that were computationally identified through clustering. Moreover, instead of modelling expression patterns separately for each gene, *BASiCS* shares information between all genes to robustly quantify transcriptional variability. For example, as described (see Software availability section), 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.

A high-level overview for the model implemented in *BASiCS* is shown in Figure 1, and a more detailed description is provided in the ExtendedMethods document in the Zenodo repository for this manuscript (see Data availability). Model parameters are designed to capture different aspects of scRNAseq data. First, *nuisance* parameters are used to capture technical artifacts. This includes cell-specific parameters (indexed by a $j$ subscript) used to perform global scaling normalisation (similar to the size factors in *scran*) and global parameters designed to capture technical over-dispersion (this can also be interpreted as measurement error) that is shared by all genes. Note that, whilst *scran* infers cell-specific size factors as a pre-processing step, *BASiCS* uses an integrated approach wherein data normalisation and downstream analyses are performed simultaneously, thereby propagating statistical uncertainty. Secondly, *BASiCS* summarises expression patterns within the population of cells under study (e.g. a experimental condition or cell type) using two sets of gene-specific parameters (indexed by a $i$ subscript). Mean parameters $\mu_i$ quantify overall expression for each gene $i$. In contrast, $\delta_i$ captures the excess of variability that is observed with respect to what would be expected in a homogeneous cell population, beyond technical noise. The latter is used 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.*[26] introduced a *joint prior* specification for these parameters. This joint prior assumes that genes with similar mean expression ($\mu_i$) have similar over-dispersion parameters $\delta_i$. Effectively, this shrinks over-dispersion estimates towards a global trend that captures the relationship between mean and over-dispersion (Figure 1). This improves posterior inference for over-dispersion parameters when the data is less informative (e.g. small sample size, lowly expressed genes).[26] This information-sharing approach is conceptually similar to that performed by Ref. 28 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 $\varepsilon_i$ 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. These residual over-dispersion values can be used to quantify transcriptional heterogeneity without any confounding with mean expression, though it should be noted that residual over-dispersion may also arise due to structural heterogeneity (e.g. distinct cell sub-types or states). For example,[26] computationally explored the impact of unobserved structural heterogeneity by generating artificially contaminated datasets. In contrast,[7] showed that changes in transcriptional heterogeneity can relate to aging, where older mice were shown to exhibit a more "bursting" or stochastic transcriptional pattern relative to younger mice.

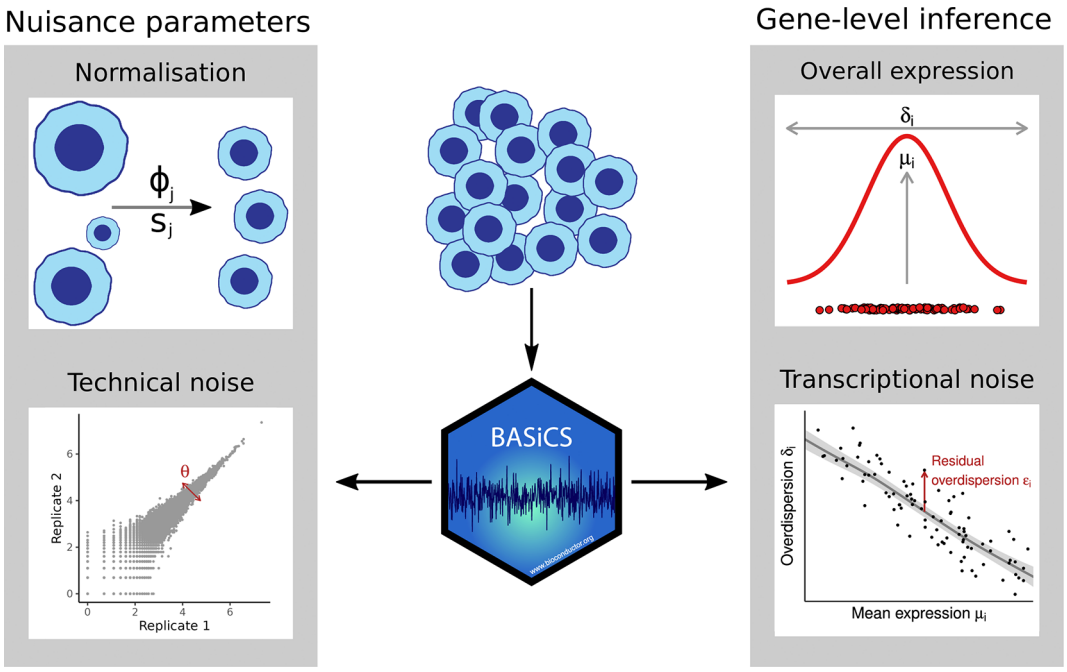



**Figure 1. A schematic representation of the features of the BASiCS model.** BASiCS accounts for cell-to-cell and batch-to-batch variability using nuisance parameters. Accounting for cell-to-cell and batch-to-batch technical variability allows BASiCS to robustly perform gene-level inference of mean and variability. Furthermore, by accounting for the association between mean and variability in scRNAseq, BASiCS can also infer transcriptional noise using the residual variability parameter $\varepsilon_i$.

Additionally, technical variation is quantified using replication.[36] 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. Thus, *BASiCS* models cell-specific size factors $y_i$, and batch-specific technical dispersion $\theta$. It is this version of the model that we focus on here, and that we recommend for most users. Previous versions of the model are available within the package, but are primarily useful for reproducibility purposes or for analysing datasets that contain spike-in genes.

While several differential expression tools have been proposed for scRNA-seq data (e.g. Refs. 37, 38), some evidence suggests that these do not generally outperform popular bulk RNA-seq tools.[39] 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.[27] 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[40] for the R programming language,[41] and as such should run on any major operating system using R $\geq$ 4.0, although results may differ slightly using different Bioconductor and R versions. A graphical overview is provided in 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/basicsworkflow2020-docker.

Here, we load all the libraries that will be used throughout this workflow. First, we load Bioconductor libraries included in Figure 2. We provide further explanation about their function in the following sections.

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

Next, we load other general purpose libraries used for data visualisation — e.g., *ggplot2*), colour scales (e.g., *viridis*, and data manipulation (*tidyr*).

```
# Data visualisation
library("ggplot2")
library("ggpointdensity")
library("patchwork")
library("ComplexHeatmap")
# colour scales
library("viridis")
library("circlize")
library("RColorBrewer")
# data manipulation
library("tidyr")
```

### Input data

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.[40]

A critical step in scRNA-seq analyses is QC, removing low quality samples that may distort downstream analyses. The *OSCA* online book provides an extensive overview on important aspects of how to perform QC of scRNA-seq data, including exploratory analyses.[40] Here, we use QC diagnostics to identify and remove samples that correspond to broken

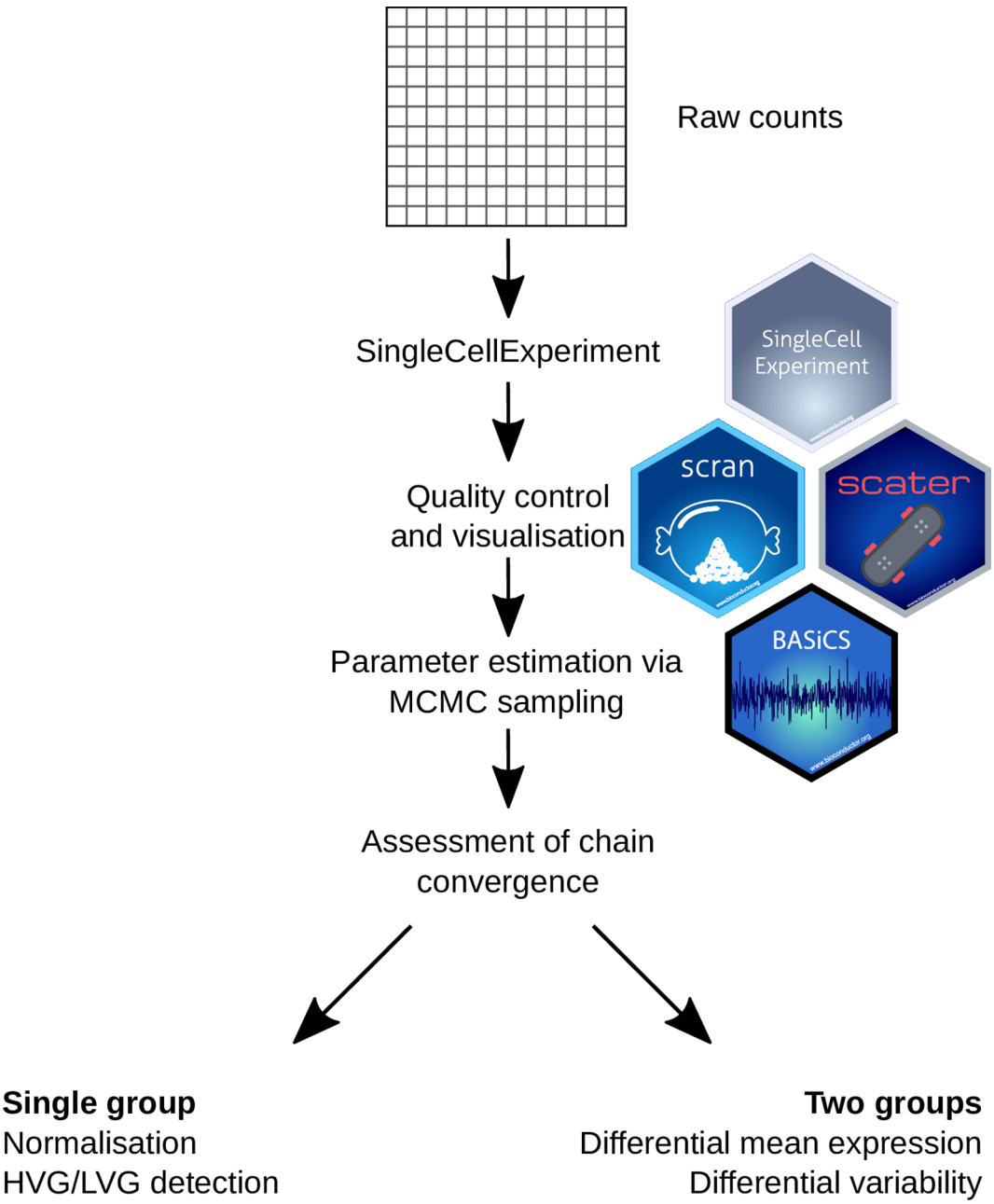



**Figure 2. Graphical overview for the single cell RNA sequencing analysis workflow described in this manuscript.** Starting from a matrix of expression counts, we use the scater and scran Bioconductor packages to perform QC and initial exploratory analyses. To robustly quantify transcriptional heterogeneity 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.

cells, that are empty, or that contain multiple cells.[42] We also remove lowly expressed genes that represent less reliable information.

We recommend the Bioconductor package *scater*[20] 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 recommend the visualisation tools implemented in the *scater* to explore the input dataset and its associated QC diagnostic metrics. To perform further exploratory data analysis we recommend the Bioconductor package *scran*.[33] The latter is used to perform *global scaling* normalisation, calculating cell-specific scaling factors that capture global differences in read-counts across cells (e.g. due to sequencing depth and PCR amplification).[43]

Note that the use of *scran* as a normalisation tool is not a requirement for *BASiCS* (which uses raw read-counts as input). However, we recommend the use of *scran* normalisation as part of the exploratory data analysis. For example, normalised read-counts can be used to visualise the data in a low-dimensional space (e.g. using principal component analysis). This can help to explore the structure of the data, e.g. to examine potential batch effects.

## Use cases

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

The development of droplet-based scRNA-seq[44,45] lead to a large increase in the number of cells that can be profiled per experiment at a fixed cost. With this, large-scale scRNA-seq datasets have been generated to study development across multiple time-points and capturing multiple tissues.[46,47]

To illustrate the use of *BASiCS* in this context, we use droplet-based scRNAseq generated using the 10X microfluidics platform. An example using a dataset generated using a plate-based protocol including spike-ins is provided in the AnalysisCD4T document of the Zenodo repository for this manuscript (see Data availability). In particular, we focus on the data generated by Ref. 46 which contains expression profiles for 20,819 cells collected from E8.25 mouse embryos. These data comprise a highly heterogeneous population of cells with over 20 major cell types. The data is stored under the accession number E-MTAB-6153 on ArrayExpress. The authors also provide additional information summarising the experimental design (animal of source for each cell) and the cluster labels generated by in their analysis. The code required to download these data is provided in the DataPreparationBASiCSWorkflow document of the Zenodo repository for this manuscript (see Data availability).

As explained above, when analysing a heterogeneous population of cells, a clustering analysis is required to identify the cell types that drive this heterogeneity prior to running *BASiCS*. *BASiCS* can then be used to quantify transcriptional variability within each cell type, as well as to characterise differences between cell types. Here, we use the cluster labels generated in the original publication. For illustration purposes, we focus on analysing presomitic and somitic mesoderm cells. This is an arbitrary choice and the steps described in this workflow could be applied to any other pair of cell types.

### Preparing the input data for BASiCS

While quality control and pre-processing is an important topic in scRNAseq analysis, we feel that it has been covered in detail in other articles. For this analysis, we have performed quality control, pre-processing and exploratory data analysis in the DataPreparationBASiCSWorkflow document of the Zenodo repository for this manuscript (see Data availability). The code presented unprocessed raw data and prepares it for use in this workflow. Here, we load the pre-processed data. This consists of two separate SingleCellExperiment data objects: one for presomitic and one for somitic mesoderm cells.

```
# Website where the files are located
files_website <- "https://zenodo.org/record/10251224/files/"
# To avoid timeout issues as we are downloading large files
options(timeout = 1000)
# File download
## The code below uses 'file.exists' to check if files were previously downloaded
## After download, files are then stored in an 'rds' sub-folder
if(!file.exists("rds/sce_sm.Rds")) {
  download.file(
    paste0(files_website, "sce_sm.Rds"),
    destfile = "rds/sce_sm.Rds"
  )
}
if(!file.exists("rds/sce_psm.Rds")) {
  download.file(
    paste0(files_website, "sce_psm.Rds"),
    destfile = "rds/sce_psm.Rds"
  )
}
# Pre-somitic mesoderm cells
sce_psm <- readRDS("rds/sce_psm.Rds")
# Somitic mesoderm cells
sce_sm <- readRDS("rds/sce_sm.Rds")
```

In brief, we have filtered the original data to select presomitic mesoderm (PSM) and somitic mesoderm (SM) cells, excluding a small cluster of cells which the authors identified a potential doublets (where multiple cells are captured in the same droplet). We also exclude cells with more 25,000 UMI-counts. The processed dataset contains 871 pre-somitic and 724 somitic mesoderm cells.

Subsequently, we applied a gene filtering step to remove lowly expressed genes that were largely undetected through sequencing. In particular, we removed genes that are not detected in at least 20 cells across all cells. This is to ensure a reliable estimate of variability, and is roughly in line with the sample size requirements for the negative binomial distribution.[48] We also applied a filter to consider protein-coding genes only. After applying these filters 10862 will be included in our analysis.

Since droplet-based scRNA-seq data are generated without including technical spike-in genes, BASiCS uses measurement error models to quantify technical variation through replication.[37] The different experimental batches used by Ref. 46 will be used for this purpose. In the case of the somitic and pre-somoitic mesoderm cells, two mouse embryos have been used to generate the data. Cells isolated from the first embryo were split into two batches and processed independently. Cells from the second embryo were processed together as a single batch. These three different groups of cells are used as batches. Numbers of cells per batch and cell type are displayed below.

```
# Pre-somitic mesoderm cells
table(sce_psm$sample)
##
## 1.1 1.2   2
## 346 339 186
# Somitic mesoderm cells
table(sce_sm$sample)
##
## 1.1 1.2   2
## 285 284 155
```

To enable BASiCS to use this information, this should be stored as `BatchInfo` within the cell-level metadata.

```
# Pre-somitic mesoderm cells
colData(sce_psm)$BatchInfo <- colData(sce_psm)$sample
# Somitic mesoderm cells
colData(sce_sm)$BatchInfo <- colData(sce_sm)$sample
```

### 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. 49). 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 $\mu_i$ and $\delta_i$,[26] and residual over-dispersion values εi are inferred. Alternatively, independent log-normal priors are assigned to $\mu_i$ and $\delta_i$.[25]
- `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*.

As a default, we recommend to use `Regression = TRUE`, as the joint prior introduced by Ref. 26 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. Advanced users may use the `BASiCS_PriorParam` function to adjust prior hyperparameters. If `PriorMu = "EmpiricalBayes"`, $\mu_i$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. We recommend that users use the defaults for `PriorParam` for most applications. We also recommend to use `PriorMu = "EmpiricalBayes"` as we have observed that an empirical Bayes framework[50] 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 somitic and pre-somitic mesoderm cells. We use 30,000 iterations (`N`), discarding the initial 15,000 iterations as burn-in, (`Burn`), and saving parameter values only once in each 15 iterations (`Thin`). We recommend this setting for large datasets, as it generally produces reliable and reproducible results. However, specific datasets may require a larger number of iterations to achieve convergence. 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_sm <- BASiCS_MCMC(
  Data = sce_sm,
  N = 30000,
  Thin = 15,
  Burn = 15000,
  Regression = TRUE,
  WithSpikes = FALSE,
  PriorParam = BASiCS_PriorParam(sce_sm, PriorMu = "EmpiricalBayes"),
  Threads = 4,
  StoreChains = TRUE,
  StoreDir = "rds/",
  RunName = "SM"
)
set.seed(43)
chain_psm <- BASiCS_MCMC(
  Data = sce_psm,
  N = 30000,
  Thin = 15,
  Burn = 15000,
  Regression = TRUE,
  WithSpikes = FALSE,
  PriorParam = BASiCS_PriorParam(sce_psm, PriorMu = "EmpiricalBayes"),
  Threads = 4,
  StoreChains = TRUE,
  StoreDir = "rds/",
  RunName = "PSM"
)
```

This first of these chains takes 140 minutes to complete on a 3.4 GHz Intel Core i7 4770k procesor with 32GB RAM, while the second takes 159 minutes. For convenience, these can be obtained online at https://doi.org/10.5281/zenodo.10251224.

```
# Website were the files are located
chains_website <- "https://zenodo.org/record/10251224/files/"
# To avoid timeout issues as we are downloading large files
options(timeout = 1000)
# File download
## The code below uses 'file.exists' to check if files were previously downloaded
## After download, files are then stored in an 'rds' sub-folder
if(!file.exists("rds/chain_sm.Rds")) {
  download.file(
    paste0(chains_website, "chain_sm.Rds"),
    destfile = "rds/chain_sm.Rds"
  )
}

if(!file.exists("rds/chain_psm.Rds")) {
  download.file(
    paste0(chains_website, "chain_psm.Rds"),
    destfile = "rds/chain_psm.Rds"
  )
}
## Loads files after download
chain_sm <- readRDS("rds/chain_sm.Rds")
chain_psm <- readRDS("rds/chain_psm.Rds")
```

The output from `BASiCS_MCMC` is a `BASiCS_Chain` object that contains the draws associated to all model parameters. Given that (`N - Burn`) /`Thin` = (30,000 - 15,000) /15 = 1,000 the object contains 1,000 draws for each parameter. 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 *bayesplot* or *tidybayes*. For example, the following code displays the first 2 MCMC draws for mean expression parameters associated to the first 3 genes.

```
displayChainBASiCS(chain_sm, Param = "mu")[1:2, 1:3]
##      ENSMUSG00000025902 ENSMUSG00000033845 ENSMUSG00000025903
## [1,]         0.03320425           3.634436          0.9257323
## [2,]         0.03785246           3.645978          0.9858456
```

## MCMC convergence diagnostics

Before interpreting the estimates generated by *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.

As part of performing quality control of MCMC performance, 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. 51, 52). 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. 53) based on a single MCMC chain.

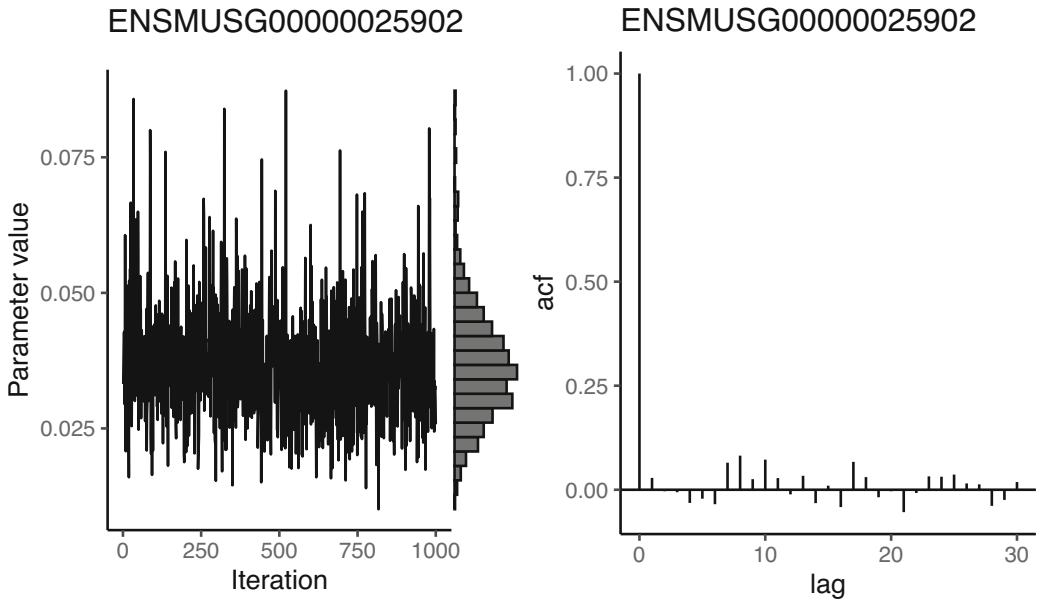



**Figure 3. Trace plot, marginal histogram, and autocorrelation function of the posterior distribution of the mean expression parameter for a gene in somitic mesoderm cells following MCMC sampling.** Trace plots should explore the posterior without getting stuck in one location or drifting over time towards a region of higher density. High autocorrelation indicates that the number of effective independent samples is low. It is good practice to perform these visualisation for many different parameters; here we only show one.

For illustration purposes, here we focus on the chains generated for the somatic mesoderm cells. However, it is important to analyse convergence for all MCMC chains; we have omitted this analysis for pre-somitic mesoderm cells in the present manuscript for brevity, but they can be viewed in the data preparation document (DataPreparationBASiCSWorkflow) in the Zenodo repository for this manuscript (see Data availability).

Traceplots can be used to visually assess the history of MCMC iterations for a specific parameter (e.g., Figure 3 left panel). As mentioned above, significant departures from a horizontal trend suggest a lack of convergence. As illustrated in Figure 3 (right panel), 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_sm, Param = "mu", Gene = 1)
```

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.

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. The function `BASiCS_DiagPlot` can be used to visualise such diagnostic metrics. Here, we illustrate usage for two such metrics focusing on the MCMC chain that was obtained for the somitic mesoderm cells (similar results were obtained for pre-somitic mesoderm cells in the DataPreparationBASiCSWorkflow document of the Zenodo repository (see Data availability)). First, we focus on the diagnostic criterion proposed by Geweke.[53] 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 somitic and pre-somitic mesoderm datasets, most Z-scores associated to mean expression parameters $\mu_i$ were small in absolute value (see Figure 4A), suggesting that the algorithm has largely converged.

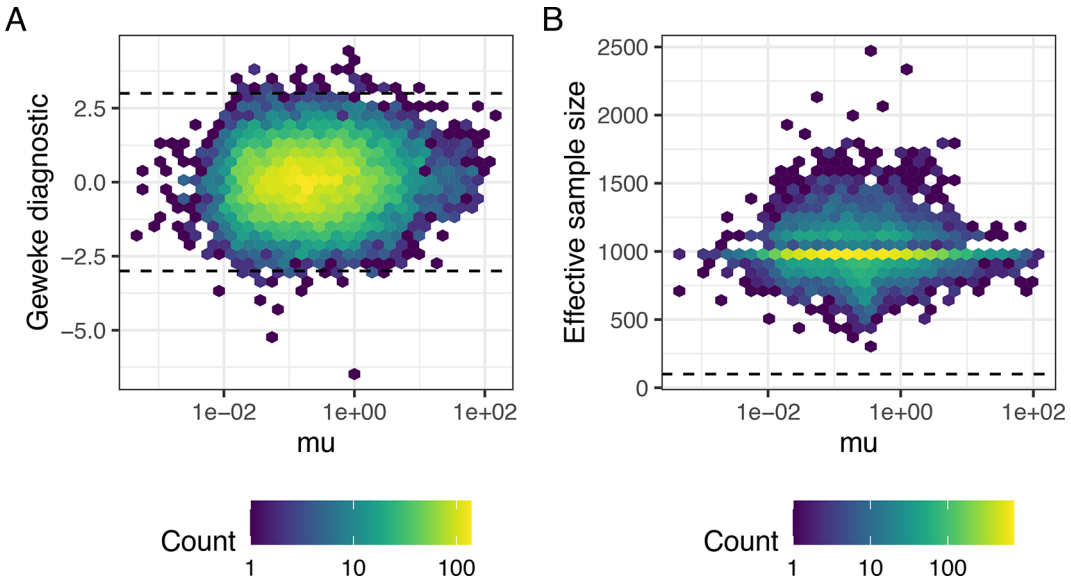



**Figure 4. Markov chain Monte Carlo diagnostics for gene-specific mean expression parameters; somitic mesoderm cells.** A: Geweke Z-score for mean expression parameters is plotted against mean expression estimates. Dashed lines represent absolute Z-scores of 3, outside of which we advise caution when 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.

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 3, 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).[54]

The effective sample size (ESS) is a related measure which represents a proxy for the number of independent draws generated by the MCMC sampler.[55] The latter is defined as:

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

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 somitic mesoderm cells are displayed in Figure 4B. 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. 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.

```
diag_p1 <- BASiCS_DiagPlot(chain_sm, Param = "mu", Measure = "geweke") +
  theme(legend.position = "bottom")
diag_p2 <- BASiCS_DiagPlot(chain_sm, Param = "mu", Measure = "ess") +
  theme(legend.position = "bottom")
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[56] 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 $\delta_i$ (see *Methods*). Figure 5 compares these approaches, focusing on somitic mesoderm cells (repeating this analysis for pre-somitic mesoderm cells led to similar results). For each of the panels in Figure 5, we use the R package *ggpointdensity* to visualise the local density of genes along the axes. Note that the `BASiCS_ShowFit` function can be used to generate Figure 5C, but we generated the plot manually to demonstrate how users can extract this information from a `BASiCS_Chain` object, and for visual consistency with the other panels.

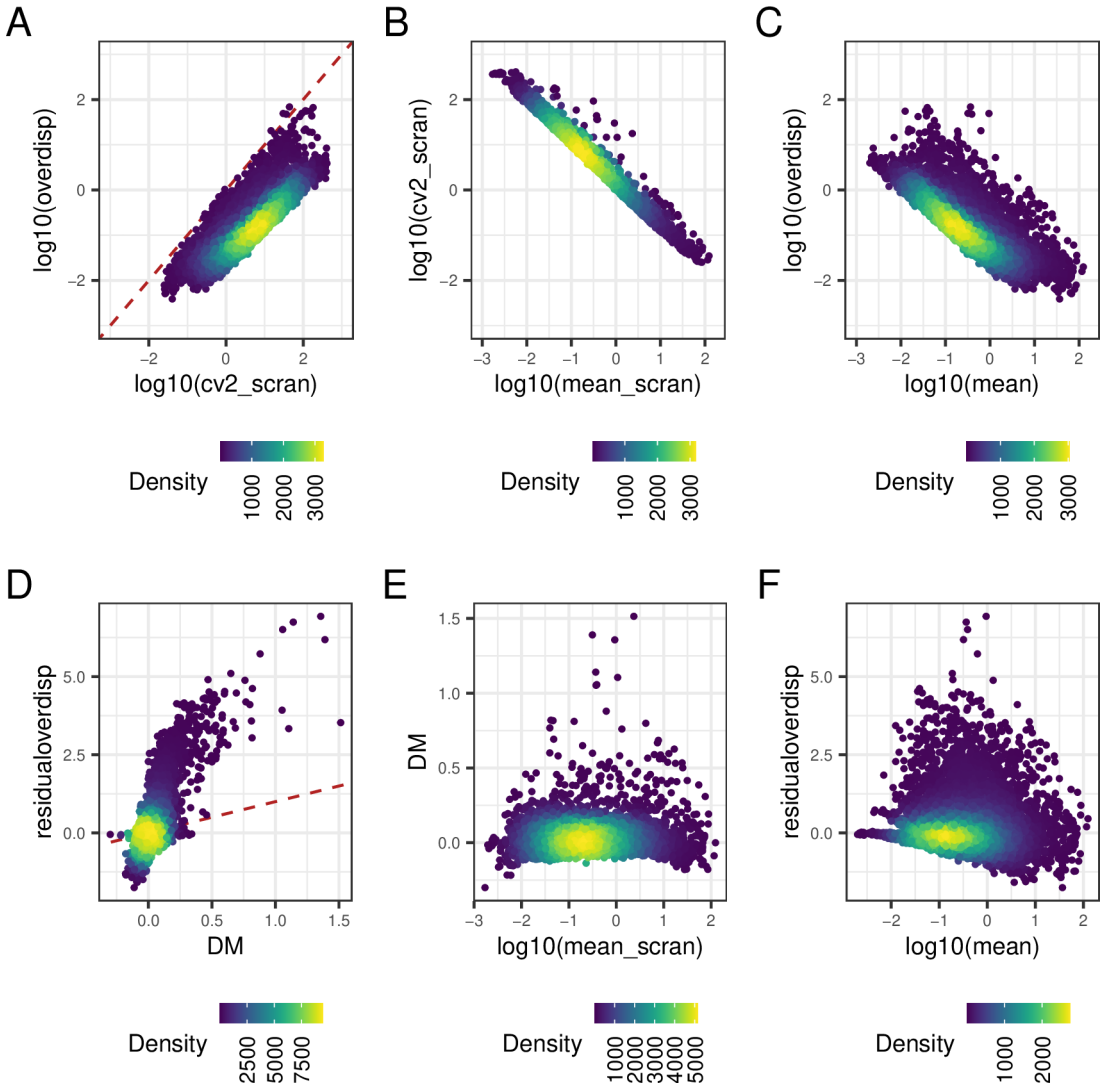



**Figure 5. Comparison of gene-specific transcriptional variability estimates and mean expression estimates obtained for each gene using BASiCS and scran.** For this analysis, we exclude genes that are not expressed in at least 2 cells. BASiCS estimates for each gene are defined by the posterior median 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 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.

As seen in Figure 5A, $CV^2$ and posterior estimates for are highly correlated. Moreover, both variability metrics are confounded by differences in mean expression, i.e. highly expressed genes tend to exhibit lower variability (Figures 5B-C). We note, however, that *scran* infers a more narrow distribution of transcriptional variability estimates for any given mean expression level. To remove the confounding observed in Figures 5B-C, *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[57] and the residual over-dispersion parameters defined by Ref. 26, respectively. For the somitic mesoderm cell data, both approaches led to correlated estimates (Figure 5D, Pearson's correlation equal to 0.75) but the residual variability estimates generated for these data are less consistent than what we have observed in the context of less sparse data (see the AnalysisCD4T document in the Zenodo repository for this manuscript (see Data availability). For the majority of genes (7,697 out of 10,850 captured in at least 3 cells) both approaches led some broadly similar results, inferring the same sign for their residual variability estimates.

```
## Get BASiCS point estimates for mean and variability - SM cells
summary_sm <- Summary(chain_sm)
parameter_df <- data.frame(
  mean = displaySummaryBASiCS(summary_sm, Param = "mu")[, 1],
  overdisp = displaySummaryBASiCS(summary_sm, Param = "delta")[, 1],
  residualoverdisp = displaySummaryBASiCS(summary_sm, Param = "epsilon")[, 1]
)

## Calculate scran size factors - SM cells
sce_sm <- computeSumFactors(sce_sm)
## Get scran estimates for mean and variability - SM cells
sce_sm <- logNormCounts(sce_sm, log = FALSE)
parameter_df$mean_scran <- rowMeans(assay(sce_sm, "normcounts"))
parameter_df$cv2_scran <- rowVars(assay(sce_sm, "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
sel_genes <- !is.na(parameter_df$residualoverdisp)

## The plots below are generated using ggplot2
## We use 'plot_params' to specify the format of
## such plots whilst avoiding code duplication
## The plots can be further customized using ggplot2 principles
## For example, we modify x- and y-axis limits in some of the plots below
plot_params <- list(
  geom_pointdensity(size = 0.6, na.rm = TRUE),
  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[sel_genes,], aes(log10(cv2_scran), log10(overdisp))) +
  xlim(-3, 3) + ylim(-3, 3) +
  geom_abline(
    slope = 1,
    intercept = 0,
    colour = "firebrick",
    linetype = "dashed"
  )
```

```
g2 <- ggplot(parameter_df[sel_genes,]) +
  aes(log10(mean_scran), log10(cv2_scran)) +
  xlim(-3, 2.2) + ylim(-3, 3)

g3 <- ggplot(parameter_df[sel_genes,], aes(log10(mean), log10(overdisp))) +
  xlim(-3, 2.2) + ylim(-3, 3)

g4 <- ggplot(parameter_df[sel_genes,], aes(DM, residualoverdisp)) +
  geom_abline(
    slope = 1,
    intercept = 0,
    colour = "firebrick",
    linetype = "dashed"
  )

g5 <- ggplot(parameter_df[sel_genes,], aes(log10(mean_scran), DM))

g6 <- ggplot(parameter_df[sel_genes,], aes(log10(mean), residualoverdisp))

((g1 + g2 + g3) * plot_params) / ((g4 + g5 + g6) * plot_params) +
   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 $\varepsilon_i$ (alternatively, the approach introduced by Ref. 24 can be used). HVGs are marked as those for which $\varepsilon_i$ 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%).[58] The expected false discovery rate we use is conceptually similar to false discovery rate control procedures in frequentist statistics, such as the approach of Ref. 59, aiming to control the proportion of false discoveries produced by the procedure. A similar approach is implemented for LVG detection, but based on whether $\varepsilon_i$ is below a pre-specified threshold `EpsilonThreshold`. For example, if the threshold for $\varepsilon_i$ is equal to log2 (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_sm, EpsilonThreshold = log(2))
## For HVG detection task:
## the posterior probability threshold chosen via EFDR calibration is too low.
## Probability threshold automatically set equal to 'ProbThreshold'.
## Lowly variable genes
lvg <- BASiCS_DetectLVG(chain_sm, EpsilonThreshold = -log(2))
```

For subsequent visualisation and data exploration, here we merge the output tables generated above and store these as part of the gene-level metadata in the `sce_sm` object.

```
## Merge HVG and LVG tables in a single data frame
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")
)
## 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"
## Store as gene-level metadata
rowData(sce_sm) <- merge(
  rowData(sce_sm), vg_table,
  by.x = "ensembl_gene_id", by.y = "GeneName",
  sort = FALSE
)
```

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 6, for the somitic mesoderm cells, this leads to a threshold of 0.6666667 and 0.8035 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 2/3 is used. This default threshold value can be changed using the `'ProbThreshold'` argument when calling the `'BASiCS_DetectLVG'` and `'BASiCS_DetectHVG'` functions.

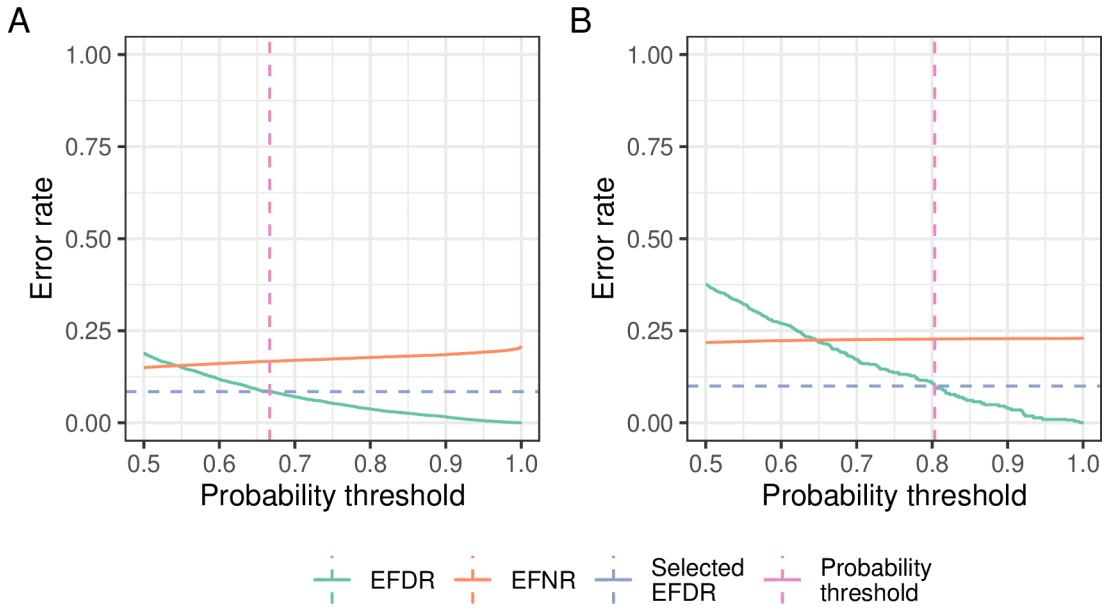



**Figure 6. Lines representing the EFDR and EFNR for a range of posterior probability thresholds for the HVG (A) and LVG (B) detection tasks.** The target EFDR is indicated by dashed horizontal lines, while the chosen posterior probability threshold is indicated by a vertical dashed line.

```
p1 <- BASiCS_PlotVG(hvg, Plot = "Grid")
p2 <- BASiCS_PlotVG(lvg, Plot = "Grid")

p1 + p2 +
  plot_annotation(tag_levels = "A") +
  plot_layout(guides = "collect") &
  theme(legend.position = "bottom")
```

```
plotRowData(sce_sm, x = "Mu", y = "Epsilon", colour_by = "VG") +
  scale_x_log10() +
  geom_hline(yintercept = c(-log(2), 0, log(2)), lty = 2) +
  labs(
    x = "BASiCS means (log10)",
    y = "BASiCS residual\nover-dispersion"
  )
```

For the somitic mesoderm cells data, we obtained *923* HVG and *33* LVG. As shown in Figure 7, these genes are distributed across a wide range of mean expression values. As an illustration, Figure 8 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 and potentially bimodal distribution (Figure 8A). Instead, LVG tend to have more narrow and unimodal distributions (Figure 8B).

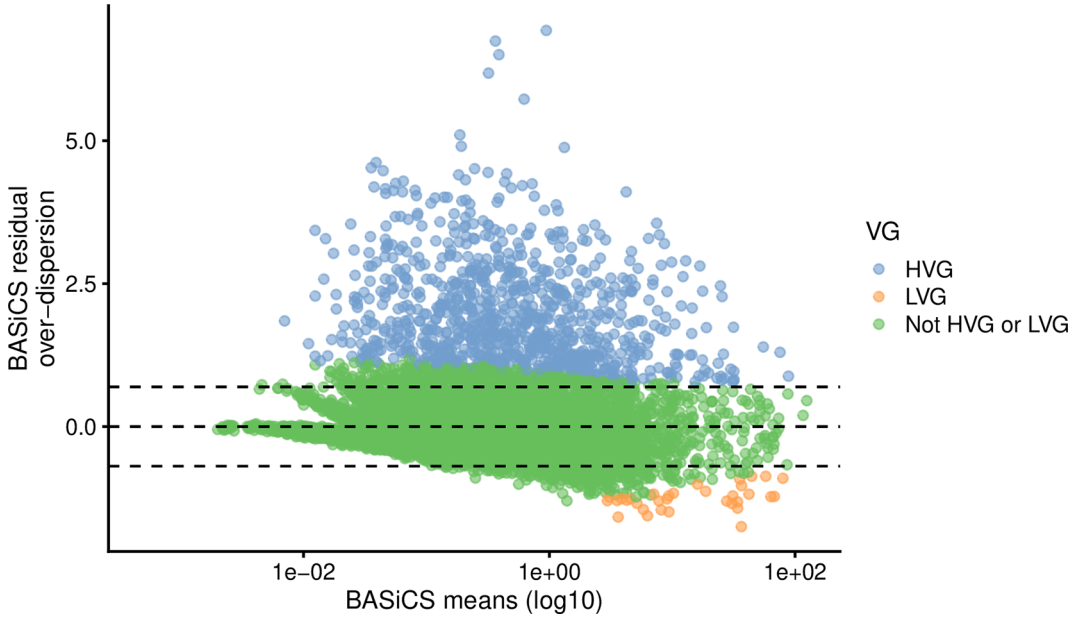



**Figure 7. HVG and LVG detection using BASiCS.** For each gene, BASiCS posterior estimates (posterior medians) associated to mean expression and residual over-dispersion parameters are plotted. Genes are coloured according to HVG/LVG status. Genes that are not expressed in at least 2 cells are excluded.

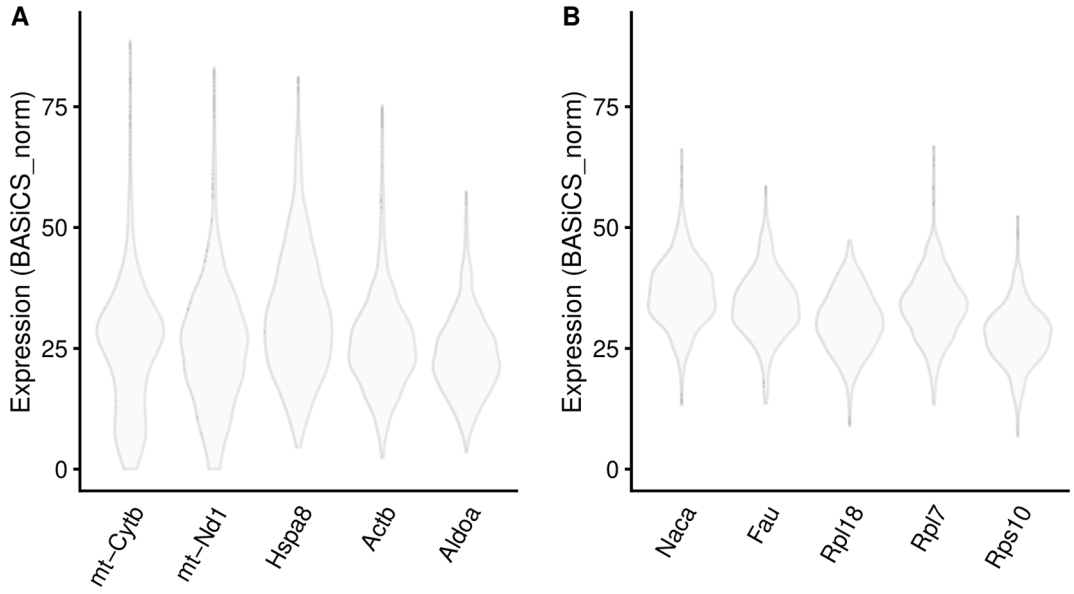



**Figure 8.** **BASiCS denoised counts for example HVG (A) and LVG (B) with similar overall levels of expression.**

```
## Obtain normalised expression values and store them as an assay in sce_sm
assay(sce_sm, "BASiCS_norm") <- BASiCS_DenoisedCounts(
  sce_sm, chain_sm, WithSpikes = FALSE
)

## Select HVG/LVG genes with similar mean expression values
low_exp <- 20
up_exp <- 40
is_mid_exp <- rowData(sce_sm)$Mu > low_exp & rowData(sce_sm)$Mu < up_exp
is_hvg <- is_mid_exp & rowData(sce_sm)$HVG
is_lvg <- is_mid_exp & rowData(sce_sm)$LVG

## Amongst those, order by epsilon and select top 5 HVG and LVG
top_hvg <- order(rowData(sce_sm)$Epsilon[is_hvg], decreasing = TRUE)[1:5]
top_lvg <- order(rowData(sce_sm)$Epsilon[is_lvg], decreasing = FALSE)[1:5]

## Generate violin plots for the selected HVG and LVG
## using scater::plotExpression
plot_hvg <- plotExpression(sce_sm,
    rowData(sce_sm)$external_gene_name[is_hvg][top_hvg],
    exprs_values = "BASiCS_norm",
    swap_rownames = "external_gene_name",
    feature_colours = FALSE,
    point_fun = function(...) list() # disables drawing points
  ) +
  ylim(0, 90)

plot_lvg <- plotExpression(
    sce_sm,
    rowData(sce_sm)$external_gene_name[is_lvg][top_lvg],
    exprs_values = "BASiCS_norm",
    swap_rownames = "external_gene_name",
    feature_colours = FALSE,
    point_fun = function(...) list() # disables drawing points
  ) +
  ylim(0, 90)
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 somitic mesoderm cells, analysed in the previous section, to pre-somitic mesoderm cells analysed in the same study. 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 log2 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`: genes with ESS values below this threshold 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_sm,
  Chain2 = chain_psm,
  GroupLabel1 = "SM",
  GroupLabel2 = "PSM",
  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
)
## combine results of differential mean and residual over-dispersion tests
table_de <- merge(table_de_mean, table_de_resdisp)
## Merge with gene-level metadata to obtain gene names used in figures
table_de <- merge(
  table_de,
  rowData(sce_sm)[, c("ensembl_gene_id", "external_gene_name")],
  by.x = "GeneName", by.y = "ensembl_gene_id"
)
# convert to standard data.frame, not S4 DFrame, for ggplot2
table_de <- as.data.frame(table_de)
```

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*.[60] We do not perform this here, but a relevant workflow is described by Maksimovic *et al.*[61]

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 9. 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.

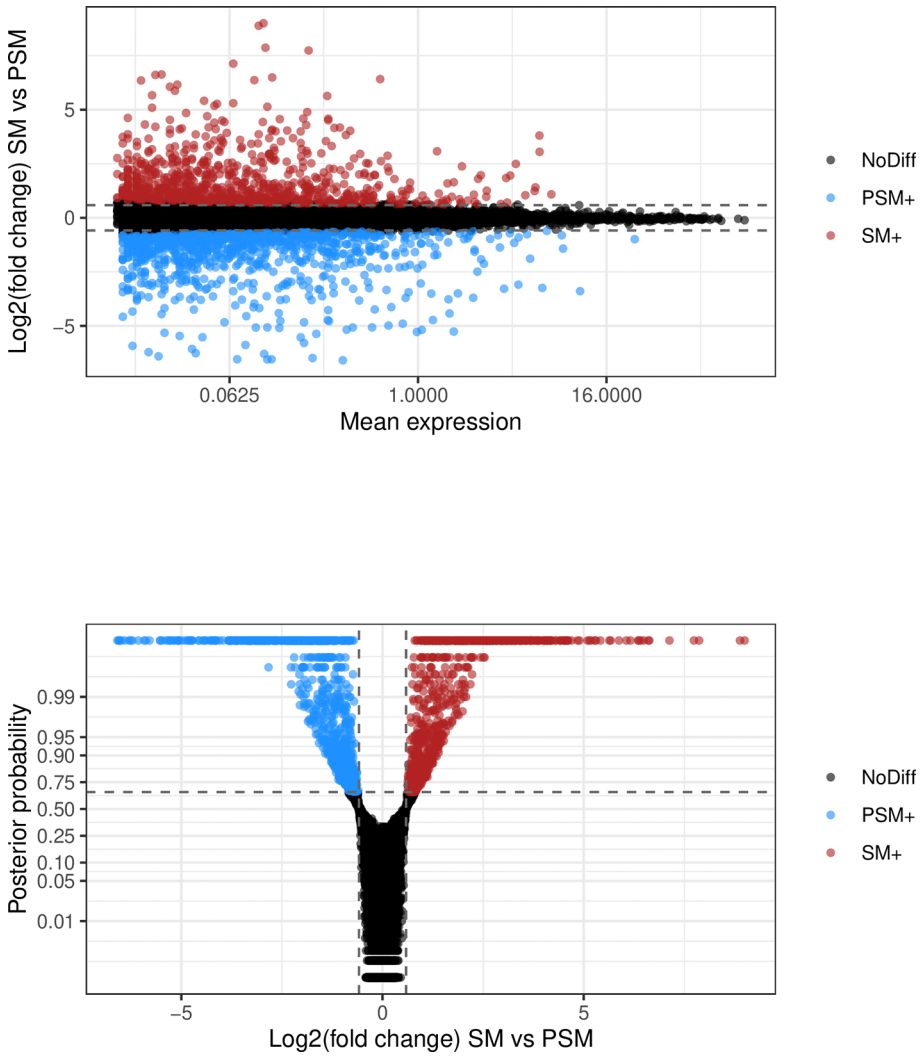



**Figure 9. Upper panel presents the mean-difference plot associated to the differential mean expression test between somitic and pre-somitic cells. Log-fold changes of average expression in somitic cells relative to pre-somitic cells are plotted against average expression estimates combined across both groups of cells. Bottom panel presents the volcano plot associated to the same test. Log-fold changes of average expression in somitic cells relative to pre-somitic mesoderm 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 ESS.**

```
p1 <- BASiCS_PlotDE(test_de, Parameters = "Mean", Plots = "MA")
p2 <- BASiCS_PlotDE(test_de, Parameters = "Mean", Plots = "Volcano",
  TransLogit = TRUE) ## logit-transforms the Y-axis, which can be clearer.
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.

```
## Calculate a global offset between groups, using the SM population as reference
offset <- BASiCS_CorrectOffset(chain_psm, chain_sm)
offset$Offset
## [1] 0.9389425
## Get offset corrected chain
chain_psm <- offset$Chain
## Obtain normalised counts within for the PSM population
## Normalised counts were for the SM population were already obtained
## in the HVG/LVG analysis section
## Obtain normalised expression values and store them as an assay in sce_sm
assay(sce_psm, "BASiCS_norm") <- BASiCS_DenoisedCounts(
  sce_psm, chain_psm,
  WithSpikes = FALSE
)
denoised_counts <- SingleCellExperiment(
  assays = list(
    BASiCS_norm = cbind(
      assay(sce_psm, "BASiCS_norm"),
      assay(sce_sm, "BASiCS_norm")
    )
  ),
  colData = data.frame(
    Celltype = c(
      rep("presomitic", ncol(sce_psm)),
      rep("somitic", ncol(sce_sm))
    )
  )
)
```

To visualise expression patterns for multiple genes at once, we use the Bioconductor package *ComplexHeatmap*[62] package, grouping genes according to the result of the differential mean expression test (i.e. up-regulated in somitic/pre-somitic cells or non differentially expressed; see Figures 10, 11 and 12). For example, among the non DE group, we observe *Cox5a*, a gene essential to mitochondrial function. 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.

To do this in each dataset, we create a few small utility functions, starting with a function to select the genes with highest probability of being differentially expressed, or the genes with smallest probability of being differentially expressed:

```
select_top_n <- function(table, condition, n=15, decreasing=TRUE) {
  ind_condition <- table$ResultDiffMean == condition
  table <- table[ind_condition,]
  ind_diff <- order(table$ProbDiffMean, decreasing = decreasing)[1:n]
  return(table$GeneName[ind_diff])
}
```

Next, we define a function that uses this gene selection to subset the normalised counts for each population:

```
use_select <- function(sce, psm, sce_sm, select) {
  counts_psm <- assay(sce_psm, "BASiCS_norm")[select,]
  counts_sm <- assay(sce_sm, "BASiCS_norm")[select,]
  rownames(counts_psm) <- rownames(counts_sm) <-
    rowData(sce_sm)[select, "external_gene_name"]
  list (psm = counts_psm, sm = counts_sm)
}
```

Next, we define a function that uses the circlize package to create a color scale for the heatmaps:

```
make_colorscale_counts <- function(counts_sm, counts_psm) {
  colorRamp2(
    breaks = seq(0,
      log10(max(c(as.numeric(counts_sm), as.numeric(counts_psm)) + 1)),
      length.out = 20
    ),
    colors = viridis(20)
  )
}
```

Next, we define a function that uses the circlize package to create a color scale for the side annotations displaying the mean expression values in each group:

```
make_colorscale_mu <- function(log_mu_sm, log_mu_psm) {
  colorRamp2(
    breaks = seq(0, max(c(log_mu_sm, log_mu_psm)), length.out = 20),
    colors = viridis(20, option = "A", direction = 1)
  )
}
```

Then, we define a function that uses the ComplexHeatmap package to create a single heatmap, with the normalised counts for a single population, featuring the mean expression values for each gene as a side annotation:

```
make_heatmap <- function(counts, col, log_mu, mu_col, title) {
  Heatmap(
    log10(as.matrix(counts) + 1),
    col = col,
    right_annotation = rowAnnotation(
      'log(mu)' = log_mu,
      col = list('log(mu)' = mu_col)
    ),
    name = "log10(count + 1)",
    column_title = title,
    show_column_names = FALSE,
    cluster_columns = FALSE,
    cluster_rows = FALSE,
    row_names_gp = gpar(fontsize = 6)
  )
}
```

Finally, we define a function that uses the functions we defined so far to create a combined heatmap showing the normalised counts for both populations, with mean expression values as side annotations:

```
plot_heatmap <- function(table, condition, decreasing = TRUE) {
  select <- select_top_n(table_de, condition, decreasing = decreasing)
  counts_sub <- use_select(sce, sce_psm, sce_sm, select)
  counts_sm <- counts_sub$sm
  counts_psm <- counts_sub$psm

  ## Subset table of DE results to extract mean estimates
  match_order <- match(rownames (counts_sm), table_de$external_gene_name)
  table_de_selected <- table_de[match_order,]

  col <- make_colorscale_counts(counts_sm, counts_psm)
  log_mu_sm <- log10(table_de_selected$Mean1)
  log_mu_psm <- log10(table_de_selected$Mean2)

  mu_col <- make_colorscale_mu(log_mu_sm, log_mu_psm)

  h <- make_heatmap(counts_sm, col, log_mu_sm, mu_col, "Somitic cells") +
      make_heatmap(counts_psm, col, log_mu_psm, mu_col, "Pre-somitic cells")
  draw(h)
}
```

First, we show genes up-regulated in somitic cells:

```
plot_heatmap(table_de, "PSM+")
```

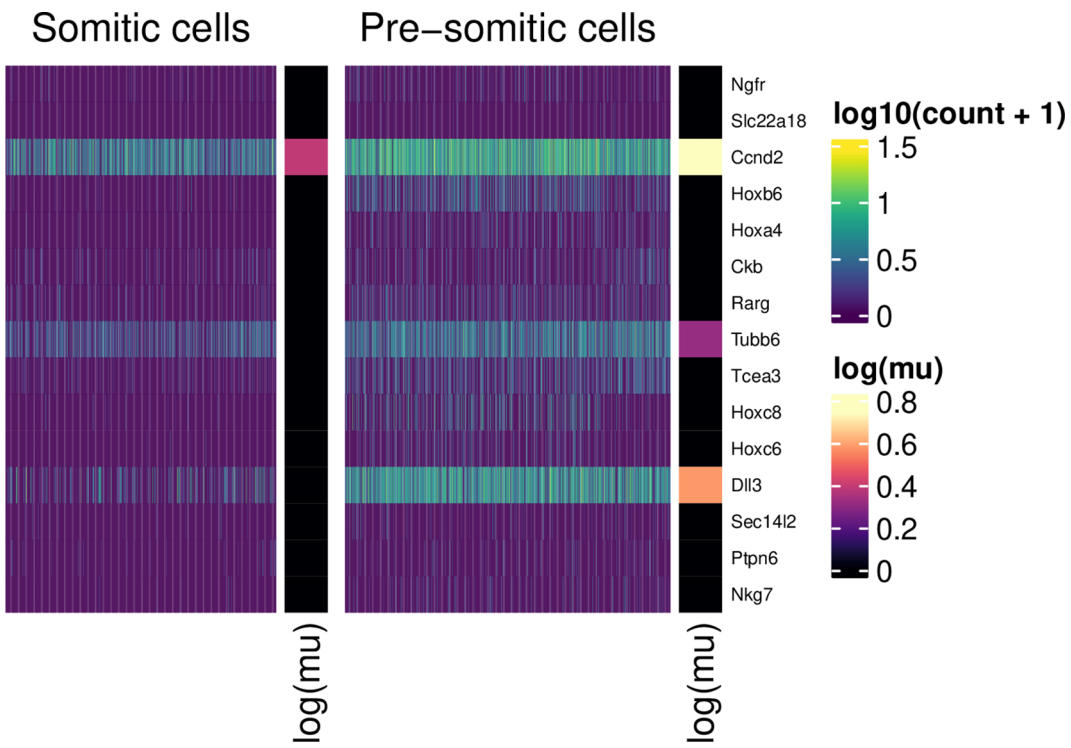



**Figure 10. Normalised expression values (scale) for somitic and pre-somitic cells, showing genes up-regulated in PSM cells.** For each group, 10 example genes are shown. These were selected according to the ranking of their associated tail posterior probabilities associated to the differential mean 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.

We can create a similar plot for genes up-regulated in SM cells:

```
plot_heatmap(table_de, "SM+")
```

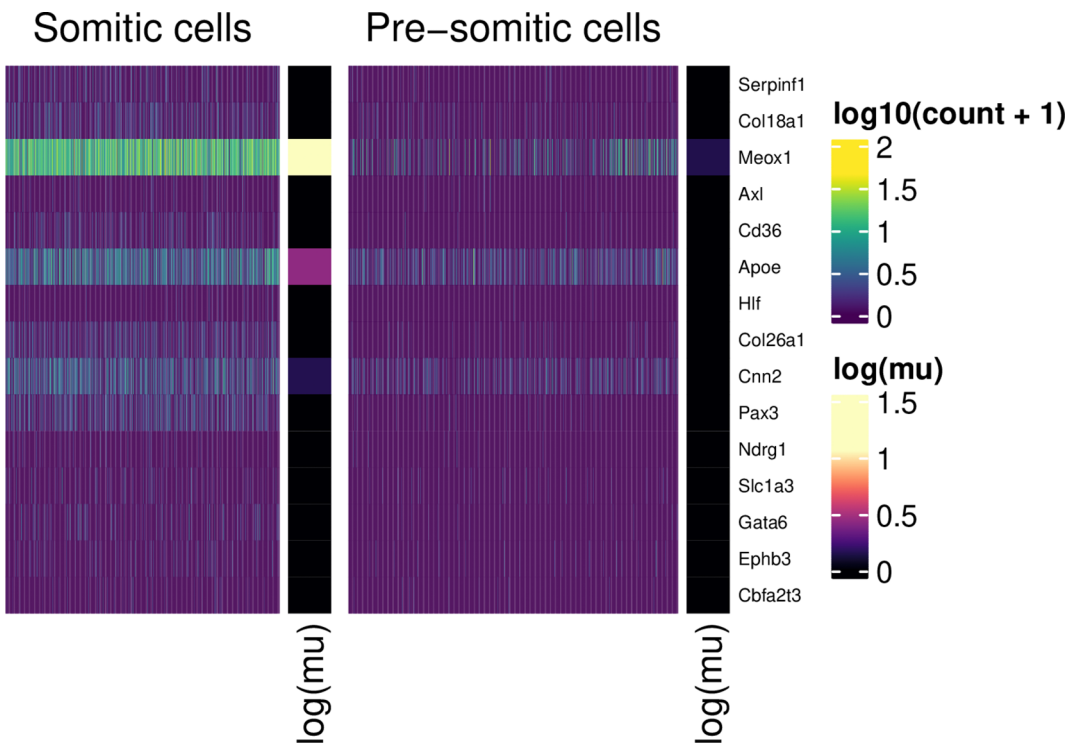



**Figure 11. Normalised expression values (scale) for somitic and pre-somitic cells, showing genes up-regulated in SM cells.** For each group, 10 example genes are shown. These were selected according to the ranking of their associated tail posterior probabilities associated to the differential mean 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.

Finally, we show non-differentially expressed genes:

```
plot_heatmap(table_de, "NoDiff", decreasing = FALSE)
```

While several computational tools exist to perform differential mean expression analysis using scRNAseq data,[39] 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 $\varepsilon_i$ as input to this analysis.

We can now visualise the changes in residual over-dispersion between somitic and pre-somitic mesoderm 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 $\varepsilon_i$ 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.

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

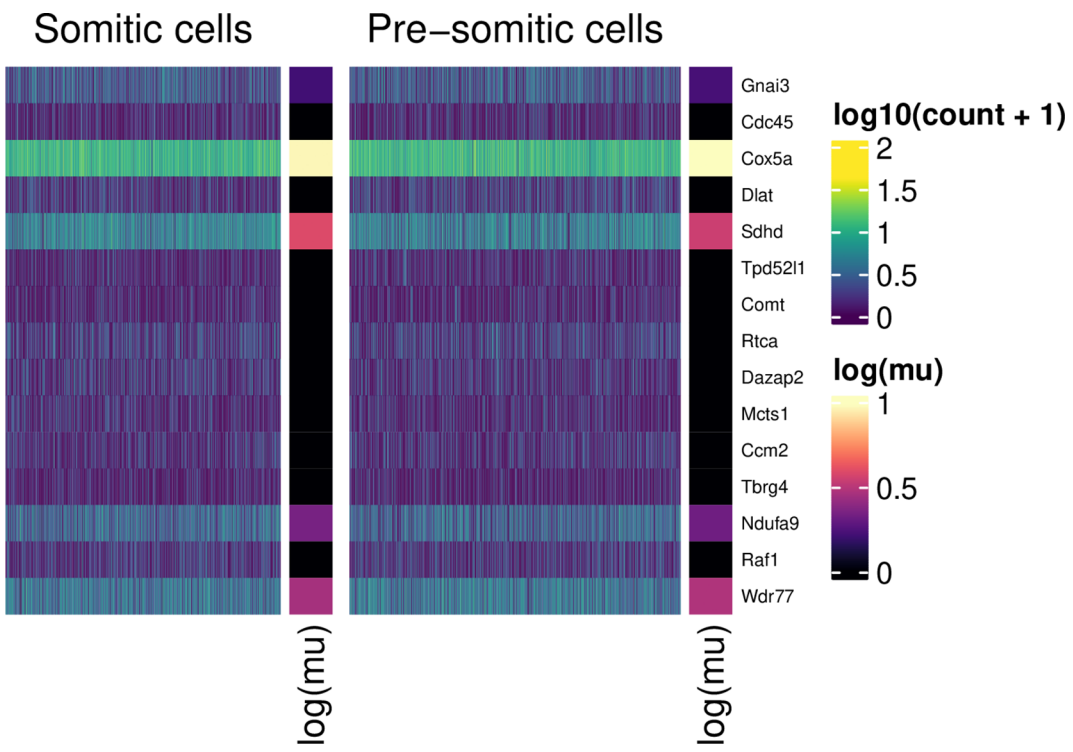



**Figure 12. Normalised expression values (scale) for somitic and pre-somitic cells, showing non-differentially expressed genes.** For each group, 15 example genes are shown. These were selected according to the ranking of their associated tail posterior probabilities associated to the differential mean 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.

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 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 with changes in mean expression, as shown in Figure 14.

```
## create list of generic plot parameters to use across a few plots
plot_params <- list(
  geom_pointdensity(na.rm = TRUE),
  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) +
  aes(MeanLog2FC, ResDispDistance) +
  plot_params +
  xlim(-15, 15) +
  labs(
    x = "log2 fold change in mean expression",
    y = "Difference in residual over-dispersion"
  )
```

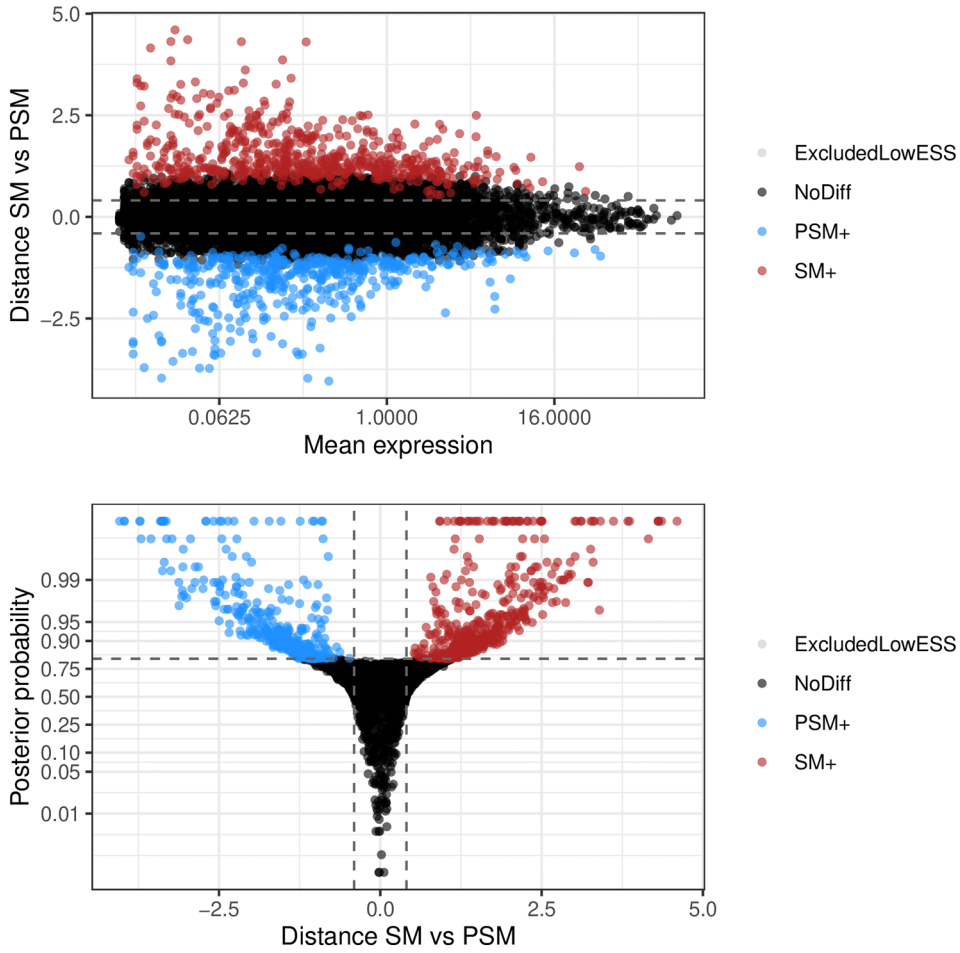



**Figure 13.** **Upper panel presents the mean-difference plot associated to the differential residual over-dispersion test between somitic and pre-somitic cells. Differences of residual over-dispersion in somitic cells relative to pre-somitic mesoderm cells are plotted against average expression estimates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

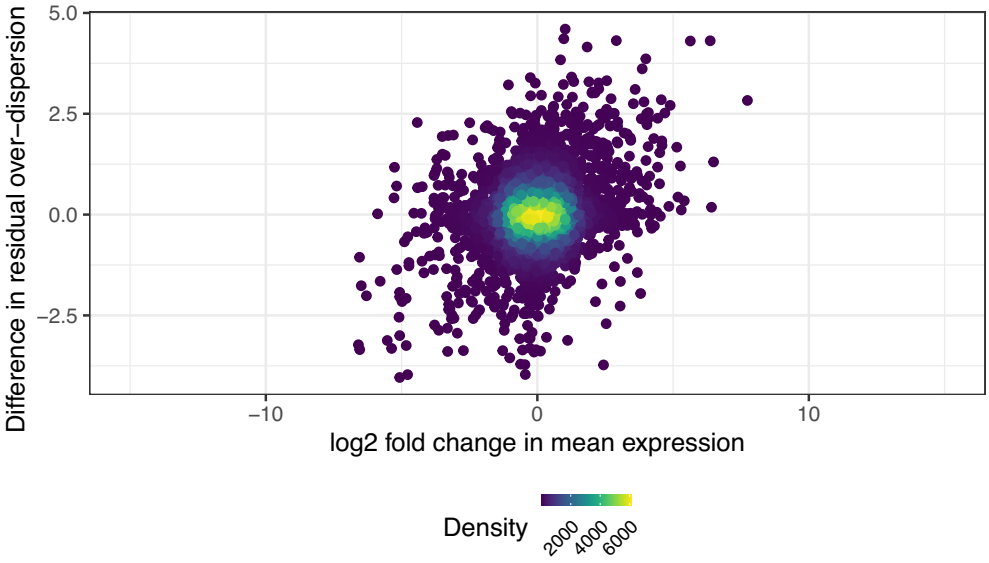



**Figure 14.** **Difference in residual over-dispersion against log2 fold change in mean expression.**

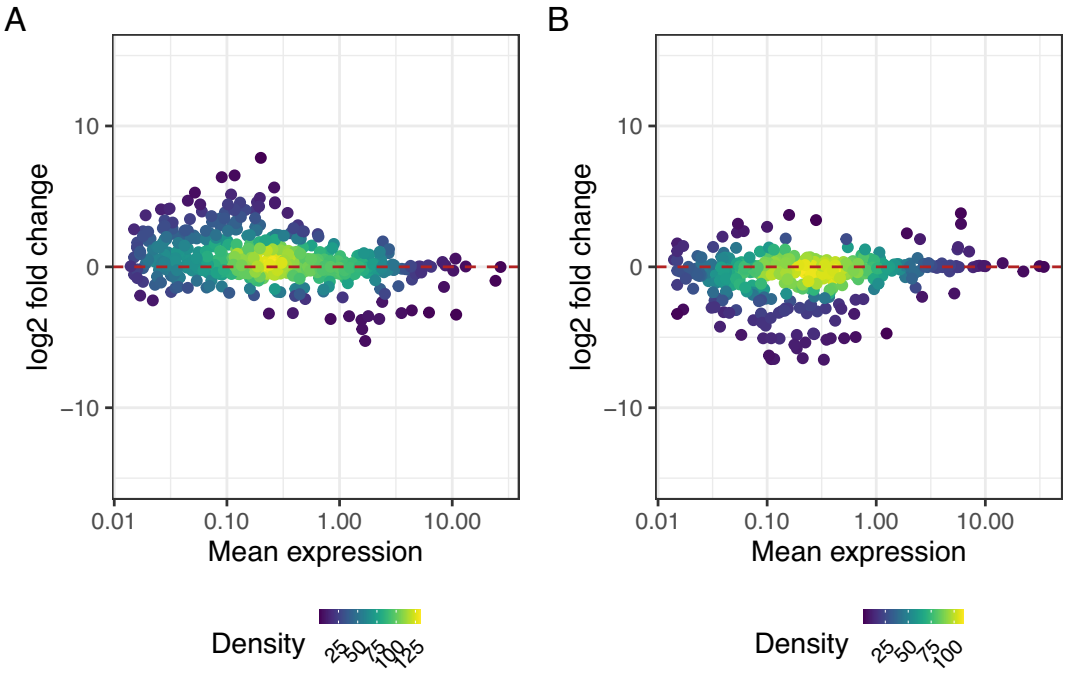



**Figure 15. $log_2$ change in expression against log mean expression for genes with higher residual over-dispersion in somitic (A) cells and pre-somitic mesoderm (B) cells.** Dashed red lines represent a log fold change of zero, meaning no change in average expression.

Genes with significant changes in residual over-dispersion often have similar levels of mean expression, as seen in Figure 15. However, they may have a different proportion of zero counts in the two cell populations, indicating a more bursting expression pattern, or similar proportion of zero counts but more varying expression levels. We will now explore ways to identify genes in each of these categories.

```
## mean expression vs log2FC for genes with higher residual over-dispersion in
## somitic cells
g1 <- ggplot(
    table_de[table_de$ResultDiffResDisp == "SM+",]
  ) +
  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"
  )

## mean expression vs log2FC for genes with higher residual over-dispersion in
## pre-somitic cells
g2 <- ggplot(
    table_de[table_de$ResultDiffResDisp == "PSM+",]
  ) +
  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"
  )
g1 + g2 + 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. To do this, we first add some gene-level summary statistic about the normalised counts to the table_de object, and then define a function that selects the normalised counts for a given gene, and then reshapes them into a form suitable for plotting.

```
## merge with some summary statistics about genes
gene_tests <- data.frame(
  "GeneName" = rownames(sce_sm),
  "ExpPropSM" = rowMeans(assay(sce_sm, "BASiCS_norm") > 0),
  "ExpPropPSM" = rowMeans(assay(sce_psm, "BASiCS_norm") > 0)
)

table_de_genetests <- merge(table_de, gene_tests)
make_vg_df <- function(ind_vg) {
  table_de_vg <- table_de[ind_vg,]
  ## If more than 4 genes,
  ## pick top 4 ranked by differences in residual over-dispersion
  if (nrow(table_de_vg) > 4) {
    table_de_vg <- table_de_vg[
      order(abs(table_de_vg$ResDispDistance), decreasing = TRUE),]
    table_de_vg <- table_de_vg[1:4,]
  }
  var_genes <- table_de_vg$GeneName
  var_sm <- as.matrix(assay(sce_sm, "BASiCS_norm")[var_genes, , drop = FALSE])
  var_psm <- as.matrix(assay(sce_psm, "BASiCS_norm")[var_genes, , drop = FALSE])
  var_df <- rbind(
    data.frame(t(var_sm), Group = "SM"),
    data.frame(t(var_psm), Group = "PSM")
  )
  var_gene_names <- rowData(sce_sm)[var_genes, "external_gene_name"]
  colnames(var_df)[-ncol(var_df)] <- var_gene_names
  pivot_longer(var_df, cols = var_gene_names)
}
```

We can use this function to wrangle the data for a selection of genes we want to visualise. For example, all genes that are identified as differentially variable in somitic cells:

```
df_vg_sm <- make_vg_df(table_de_genetests$ResultDiffResDisp %in% c("SM+"))
head(df_vg_sm)
## # A tibble: 6 × 3
##   Group  name   value
##   <chr>  <chr>  <dbl>
## 1 SM     Krt19  0
## 2 SM     Hesx1  0
## 3 SM     Kdr    0
## 4 SM     Uncx   0
## 5 SM     Krt19  0
## 6 SM     Hesx1  0
```

We can then use this function to plot the normalised counts for genes that fill a number of criteria with respect to mean expression, expression variability, and detection levels, by defining a function that uses the `make_vg_df` function we just defined. This function takes a logical vector and uses it to subset the `table_de`, `sce_sm` and `sce_psm` objects, merges them into a form suitable for plotting, and then plots them.

```
## Utility function that plots logcounts of a set of genes defined by "ind_vg"
plot_vg <- function(ind_vg) {
  df <- make_vg_df(ind_vg)
  ggplot (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),
      shape = 16,
      alpha = 0.2
    ) +
    scale_x_discrete(guide = guide_axis(angle = 45)) +
    scale_y_log10() +
    labs(x = "Gene", y = "Denoised count + 1") +
    scale_colour_brewer(palette = "Set1")
}
```

First, we select genes that are more variable in somitic cells than in pre-somitic mesoderm cells, but which are not differentially expressed. Furthermore, we narrow our selection to genes with similar levels of non-zero expression in both populations, and which are expressed in at least 50% of cells within each population. These criteria are designed to select genes that likely have a largely unimodal expression pattern, simply having higher cell-to-cell variability in somitic cells. We then plot the normalised counts for these genes using the function we defined above, and save it to a variable called g1 for display later.

```
ind_vg_unimodal_sm <- table_de_genetests$ResultDiffResDisp %in% c("SM+") &
  table_de_genetests$ResultDiffMean %in% c("NoDiff") &
  abs(
    table_de_genetests$ExpPropSM - table_de_genetests$ExpPropPSM
  ) < 0.05 &
  pmin(
    table_de_genetests$ExpPropSM, table_de_genetests$ExpPropPSM
  ) > 0.99
g1 <- plot_vg(ind_vg_unimodal_sm) +
    theme (legend.position = "none")
```

Next, we select genes that are more variable in somitic cells than in pre-somitic mesoderm cells, but which are not differentially expressed. Furthermore, we narrow our selection to genes with different levels of non-zero expression in both populations, and which are expressed in at least 50% of cells within one or more of the populations. These are genes that could be described as having a more bursting pattern of expression in somitic cells. We then plot the normalised counts for these genes, and again save the plot object for display later.

```
ind_vg_bursting_sm <- table_de_genetests$ResultDiffResDisp %in% c("SM+") &
  table_de_genetests$ResultDiffMean %in% c("NoDiff") &
  abs(
    table_de_genetests$ExpPropSM - table_de_genetests$ExpPropPSM
  ) > 0.1 &
  pmax(
    table_de_genetests$ExpPropSM, table_de_genetests$ExpPropPSM
  ) > 0.75
g2 <- plot_vg(ind_vg_bursting_sm)
```

Now, we use the same selection criteria, plotting first for genes that are largely unimodal, with a high cell-to-cell variability in pre-somitic cells, and then for genes that have a more bursting expression pattern in pre-somitic cells:

```
ind_vg_unimodal_psm <- table_de_genetests$ResultDiffResDisp %in% c("PSM+") &
  table_de_genetests$ResultDiffMean %in% c("NoDiff") &
  abs(
    table_de_genetests$ExpPropSM - table_de_genetests$ExpPropPSM
  ) < 0.1 &
  pmin(
    table_de_genetests$ExpPropSM, table_de_genetests$ExpPropPSM
  ) > 0.75
g3 <- plot_vg(ind_vg_unimodal_psm) +
    theme(legend.position = "none")

ind_vg_bursting_psm <- table_de_genetests$ResultDiffResDisp %in% c("PSM+") &
  table_de_genetests$ResultDiffMean %in% c("NoDiff") &
  abs(
    table_de_genetests$ExpPropSM - table_de_genetests$ExpPropPSM
  ) > 0.10 &
  pmax(
    table_de_genetests$ExpPropSM, table_de_genetests$ExpPropPSM
  ) > 0.50
g4 <- plot_vg(ind_vg_bursting_psm)
```

Now that we have created four violin plots of the various types of differentially variable gene that we have defined, we can combine the plots together to visualise them simultaneously:

```
(g1 + g2) / (g3 + g4) + plot_annotation(tag_levels = "A")
```

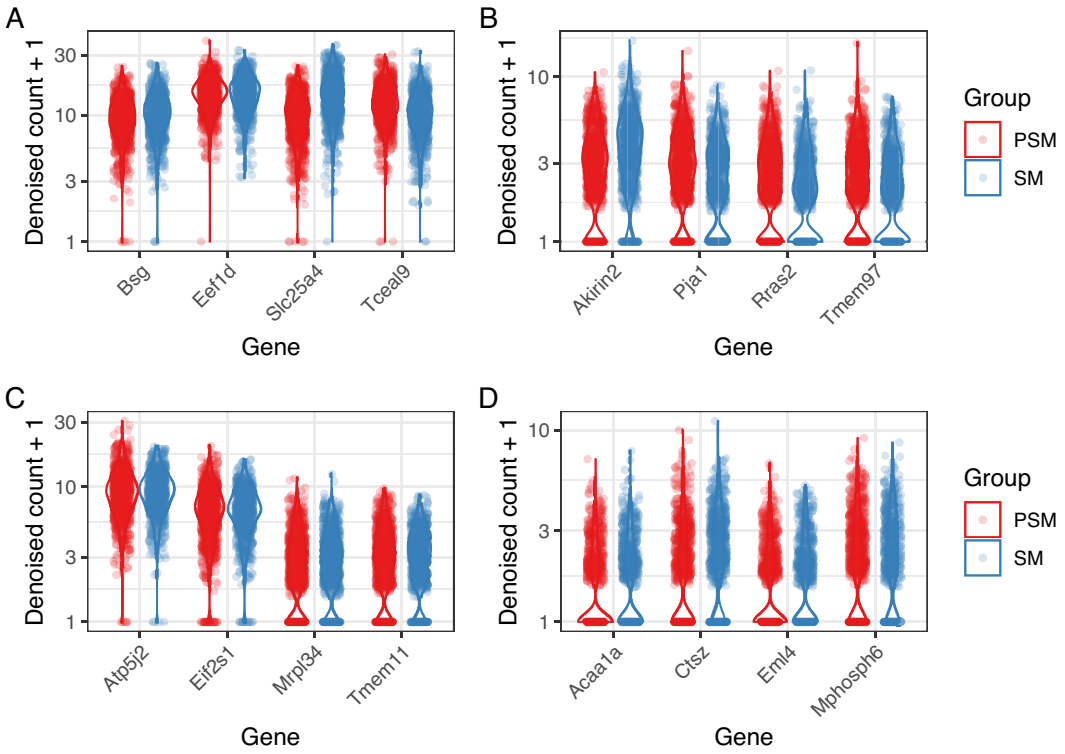



**Figure 16. Violin plots of denoised counts.** A: Four genes with higher residual over-dispersion in somitic cells, and similar levels of detection in pre-somitic and somitic populations. B: Four genes with higher residual over-dispersion in somitic cells and different levels of detection in somitic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

Figure 16 shows denoised counts for genes with significant differences in residual over-dispersion, with each panel showing a different type of expression patterns. 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 somitic or pre-somitic mesoderm 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 over-dispersion in somitic or pre-somitic mesoderm cells (A and C, respectively). These cases are likely driven by more tight regulation, rather than transcriptional burst or sub-population structure.

## 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.[10,27] One area for future work is to study how post-selection inference may affect the performance of *BASiCS*. For example, Ref. 63 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. Another important challenge for *BASiCS* is computational efficiency. The number of cells profiled in scRNAseq experiments has scaled exponentially since the development of the technology.[64] 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. Alternative inference schemes such as variational inference, maximum a posteriori estimation, or Hamiltonian Monte Carlo may be useful in this context. Advanced users may wish to use the `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`,[65] 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 facster approximations provided by Stan, namely scalable variational inference and maximum a posteriori estimation, are often unstable and less accurate.

Finally, we also anticipate potential extensions of *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,[66] Bayesian hierarchical models such as the one implemented in *BASiCS* could also be designed to incorporate spatial (and, potentially, temporal) structure (e.g. Ref. 67). We intend to update this workflow as the field evolves, and as we address the issues and challenges outlined here.

## Data availability

The data used in this article are available for download on EBI ArrayExpress under accession number E-MTAB-4888. The MCMC chains used to generate this article can be found in Zenodo under the DOI 10.5281/zenodo.10251224.

## Software availability

All software used in this workflow is available as part of Bioconductor 3.18 at: https://bioconductor.org/packages/3.18.

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/BASiCSWorkflow/, and as archived source code at time of publication: https://zenodo.org/doi/10.5281/zenodo.5224614.[26]

## Reproducibility

The following software versions were used throughout this workflow:

- **R version**: R version 4.3.2 (2023-10-3)
- **Bioconductor version**: 3.18
- **R packages**:
  - BASiCS 2.14.0
  - scran 1.30.10
  - scater 1.30.1

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/basicsworkflow2020-docker:0.5.3`.

### Session Info

```
sessionInfo()
## R version 4.3.2 (2023-10-31)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 22.04.3 LTS
##
## Matrix products: default
## BLAS: /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
## LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.20.so;
LAPACK version 3.10.0
##
## 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=en_US.UTF-8
##  [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
##
## time zone: Etc/UTC
## tzcode source: system (glibc)
##
## attached base packages:
## [1] grid     stats4    stats     graphics    grDevices utils     datasets
## [8] methods  base
##
## other attached packages:
##  [1] tidyr_1.3.0                      RColorBrewer_1.1-3
##  [3] circlize_0.4.15                  viridis_0.6.4
##  [5] viridisLite_0.4.2                ComplexHeatmap_2.18.0
##  [7] patchwork_1.1.3                  ggpointdensity_0.1.0
##  [9] BASiCS_2.14.0                    scran_1.30.0
## [11] scater_1.30.1                    scuttle_1.12.0
## [13] SingleCellExperiment_1.24.0      SummarizedExperiment_1.32.0
## [15] Biobase_2.62.0                   GenomicRanges_1.54.1
## [17] GenomeInfoDb_1.38.1              IRanges_2.36.0
## [19] S4Vectors_0.40.2                 BiocGenerics_0.48.1
```

```
## [21] MatrixGenerics_1.14.0              matrixStats_1.1.0
##  23] ggplot2_3.4.4                      knitr_1.45
## [25] BiocStyle_2.30.0
##
## loaded via a namespace (and not attached):
##   [1] rstudioapi_0.15.0                  tensorA_0.36.2
##   [3] shape_1.4.6                        magrittr_2.0.3
##   [5] magick_2.8.1                       ggbeeswarm_0.7.2
##   [7] farver_2.1.1                       rmarkdown_2.25
##   [9] fs_1.6.3                           GlobalOptions_0.1.2
##  [11] zlibbioc_1.48.0                    vctrs_0.6.4
##  [13] Cairo_1.6-2                        DelayedMatrixStats_1.24.0
##  [15] RCurl_1.98-1.13                    tinytex_0.48
##  [17] usethis_2.2.2                      htmltools_0.5.6.1
##  [19] S4Arrays_1.2.0                     distributional_0.3.2
##  [21] BiocNeighbors_1.20.0               SparseArray_1.2.2
##  [23] plyr_1.8.9                         igraph_1.5.1
##  [25] mime_0.12                          lifecycle_1.0.3
##  [27] iterators_1.0.14                   pkgconfig_2.0.3
##  [29] rsvd_1.0.5                         Matrix_1.6-1.1
##  [31] R6_2.5.1                           fastmap_1.1.1
##  [33] GenomeInfoDbData_1.2.11            shiny_1.7.5.1
##  [35] clue_0.3-65                        digest_0.6.33
##  [37] colorspace_2.1-0                   dqrng_0.3.2
##  [39] irlba_2.3.5.1                      beachmat_2.18.0
##  [41] labeling_0.4.3                     fansi_1.0.5
##  [43] httr_1.4.7                         abind_1.4-5
##  [45] compiler_4.3.2                     withr_2.5.1
##  [47] doParallel_1.0.17                  backports_1.4.1
##  [49] BiocParallel_1.36.0                hexbin_1.28.3
##  [51] highr_0.10                         MASS_7.3-60
##  [53] DelayedArray_0.28.0                rjson_0.2.21
##  [55] bluster_1.12.0                     tools_4.3.2
##  [57] vipor_0.4.5                        beeswarm_0.4.0
##  [59] httpuv_1.6.12                      glue_1.6.2
##  [61] promises_1.2.1                     checkmate_2.3.0
##  [63] cluster_2.1.4                      reshape2_1.4.4
##  [65] generics_0.1.3                     gtable_0.3.4
##  [67] BiocSingular_1.18.0                ScaledMatrix_1.10.0
##  [69] metapod_1.10.0                     utf8_1.2.4
##  [71] XVector_0.42.0                     stringr_1.5.0
##  [73] ggrepel_0.9.4                      foreach_1.5.2
##  [75] pillar_1.9.0                       ggExtra_0.10.1
##  [77] limma_3.58.1                       posterior_1.5.0
##  [79] later_1.3.1                        BiocWorkflowTools_1.28.0
##  [81] dplyr_1.1.4                        lattice_0.22-5
##  [83] tidyselect_1.2.0                   locfit_1.5-9.8
##  [85] miniUI_0.1.1.1                     git2r_0.33.0
##  [87] gridExtra_2.3                      bookdown_0.36
##  [89] edgeR_4.0.2                        xfun_0.40
##  [91] statmod_1.5.0                      stringi_1.7.12
##  [93] yaml_2.3.7                         evaluate_0.22
##  [95] codetools_0.2-19                   tibble_3.2.1
##  [97] BiocManager_1.30.22                cli_3.6.1
##  [99] xtable_1.8-4                       munsell_0.5.0
## [101] Rcpp_1.0.11                        coda_0.19-4
## [103] png_0.1-8                          parallel_4.3.2
## [105] ellipsis_0.3.2                     assertthat_0.2.1
```

```
## [107] sparseMatrixStats_1.14.0          bitops_1.0-7
## [109] scales_1.3.0                      purrr_1.0.2
## [111] crayon_1.5.2                      GetoptLong_1.0.5
## [113] rlang_1.1.1                       cowplot_1.1.1
```

## 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.
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.
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; **10**(11): 1093–1095.
PubMed Abstract | Publisher Full Text

18. 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 | Free Full Text

19. 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

20. 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

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. Bacher R, Kendziorski C: **Design and computational analysis of single-cell RNA-sequencing experiments.** *Genome Biol.* 2016; **17**(1): 63.
Publisher Full Text

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

25. 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

26. 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

27. 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

28. Love MI, Huber W, Anders S: **Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2.** *Genome Biol.* 2014; **15**(12): 550.
PubMed Abstract | Publisher Full Text | Free Full Text

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

30. Townes WF: **Review of Probability Distributions for Modeling Count Data.** *arXiv:2001.04343 [stat].* January 2020.

31. Townes WF, 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

32. Zappia L, Phipson B, Oshlack A: **Splatter: Simulation of single-cell RNA sequencing data.** *Genome Biol.* 2017; **18**(1): 174.
PubMed Abstract | Publisher Full Text | Free Full Text

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

34. 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

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

36. Carroll RJ: **Measurement Error in Epidemiologic Studies.** Published online2005: 38.
Publisher Full Text

37. Kharchenko PV, Silberstein L, Scadden DT: **Bayesian approach to single-cell differential expression analysis.** *Nat. Methods.* jul 2014; **11**(7): 740–742.
PubMed Abstract | Publisher Full Text | Reference Source

38. 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

39. 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

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

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

42. 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

43. 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

44. Klein AM, Mazutis L, Akartuna I, *et al.*: **Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells.** *Cell.* 2015; **161**(5): 1187–1201.
PubMed Abstract | Publisher Full Text | Free Full Text

45. Macosko EZ, Basu A, Satija R, *et al.*: **Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets.** *Cell.* 2015; **161**(5): 1202–1214.
PubMed Abstract | Publisher Full Text | Free Full Text

46. Ibarra-Soria X, Jawaid W, Pijuan-Sala B, *et al.*: **Defining murine organogenesis at single-cell resolution reveals a role for the leukotriene pathway in regulating blood progenitor formation.** *Nat. Cell Biol.* 2018; **20**(2): 127–134.
PubMed Abstract | Publisher Full Text | Free Full Text

47. Kernfeld EM, Genga RMJ, Neherin K, *et al.*: **A Single-Cell Transcriptomic Atlas of Thymus Organogenesis Resolves Cell Types and Developmental Maturation.** *Immunity.* 2018; **48**: 1258–1270.e6.
Publisher Full Text

48. Lloyd-Smith JO: **Maximum Likelihood Estimation of the Negative Binomial Dispersion Parameter for Highly Overdispersed Data, with Applications to Infectious Diseases.** *PLoS One.* 2007; **2**(2): e180.
PubMed Abstract | Publisher Full Text | Free Full Text

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

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

51. 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

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

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

54. 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

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

56. 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

57. 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

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

59. Benjamini Y, Hochberg Y: **Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing.** *J. R. Stat. Soc., Ser. B, Methodol.* 1995; **57**(1): 289–300.

60. 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

61. 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

62. Gu Z, Eils R, Schlesner M: **Complex heatmaps reveal patterns and correlations in multidimensional genomic data**. *Bioinformatics.* Published online 2016.

63. Neufeld A, Gao LL, Popp J, *et al.*: **Inference after latent variable estimation for single-cell RNA sequencing data.** Published online July2022; **25**: 270–287.
Publisher Full Text | Reference Source

64. 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

65. Carpenter B, Gelman A, Hoffman MD, *et al.*: **Stan: A probabilistic programming language.** *J. Stat. Softw.* 2017; **76**(1): 1–32.
PubMed Abstract | Publisher Full Text | Free Full Text

66. Marx V: **Method of the Year: Spatially resolved transcriptomics.** *Nat. Methods.* 2021; **18**(1): 9–14.
PubMed Abstract | Publisher Full Text

67. Äijö T, Maniatis S, Vickovic S, *et al.*: **Splotch: Robust Estimation of Aligned Spatial Temporal Gene Expression Data.** *Bioinformatics.* 2019.
Publisher Full Text

# Open Peer Review

## Current Peer Review Status: ✔ ? ?

**Version 2**

Reviewer Report 09 May 2024

https://doi.org/10.5256/f1000research.162017.r274782



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

I greatly appreciate the updated article and, from my perspective, is more focused towards its target audience. The work here represent a significant effort on the author's part to ensure their work is reproducible and, importantly, useable to their intended community.

I have a few (extremely minor) comments.

1) In the description of `Thin`, I would suggest something like the following definition: A value r such that the sequence of draws indexed by 1 + rn, for integer values of n, are stored (e.g. reduce the size by roughly 1/r). I think the current definition is incorrect e.g. for a Thin of 4, you wish to store values 1, 5, 9,... ? Not just the 4th? Perhaps I am mis-undertanding

2) I would be a bit more pedantic (sorry!) about \mu_i and \delta_i. You want carefully distinguish different way this could be (mis)-interpreted. The main point is to be really careful about independence between \mu and \delta vs independence between \mu_j and \mu_i. For example, the following options are all possible given some of the descriptions but I think only two (A and C) are what you intend?

A) The *pair* (\mu_i, \delta_i) has a joint prior p(\mu_i, \delta_i) (e.g. there is a single global prior for all i)

B) The *pair**s*** (\mu_i, \delta_i), indexed by i, have joint priors p_i (e.g. each pair has a seperate prior from the same family)

C) independent univariate log-normal priors are assigned to *each of* *mu_i* and to *delta_i*

D) a joint prior is applied over all \mu_i and a joint prior over all \delta_i

3) Increasing the value of `EpsilonM`. I think I bit more of a hint as to what sort of range of values one might consider would be useful.

4) Bayesian inference is quite a nice solution to post-selection inference because you can condition on everything but that requires you to model any clustering process heirachically. I think cut a model would also be appropriate - again more modelling for a another time!

***Competing Interests:*** No competing interests were disclosed.

***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.**

**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 $\mu i$ 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 ($\mu_i$) have similar over-dispersion parameters $\delta_i$. 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 $\epsilon_i$ 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 ($\mu_i$) have similar over-dispersion parameters $\delta_i$. 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 $\epsilon_i$ 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.