BASiCS workflow: a step-by-step analysis of expression variability using single cell RNA sequencing data

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

[1]**European Molecular Biology Laboratory, European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SD, UK**
[2]**Cancer Research UK Cambridge Institute, University of Cambridge, Li Ka Shing Centre, Cambridge, CB2 0RE, UK**
[3]**Department of Quantitative Biomedicine, University of Zurich, Winterthurerstrasse 190, CH-8057, Zurich, Switzerland**
[4]**MRC Human Genetics Unit, Institute of Genetics & Molecular Medicine, University of Edinburgh, Western General Hospital, Crewe Road, Edinburgh, EH4 2XU, UK**
[5]**The Alan Turing Institute, British Library, 96 Euston Road, London, NW1 2DB, UK**

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 which 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, whilst 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 two publicly available datasets, we guide users through a complete pipeline which includes preliminary steps for quality control as well as data exploration using the scater and scran Bioconductor packages. Data for the first case study was generated using the Fluidigm@ C1 system, in which extrinsic spike-in RNA molecules were added as a control. The second dataset was generated using a droplet-based system, for which spike-in RNA is not available. This analysis provides an example, in which differential variability testing reveals insights regarding a possible early cell fate commitment process. The workflow is accompanied by a Docker image that ensures the reproducibility of our results.

Keywords

Single-cell RNA sequencing, expression variability, transcriptional noise, differential expression testing

[*]a.b.o'callaghan@sms.ed.ac.uk
[†]catalina.vallejos@igmm.ed.ac.uk

Introduction

Single-cell RNA-sequencing (scRNA-seq) enables the study of genome-wide transcriptional heterogeneity in cell populations that is not captured by bulk experiments [1, 2, 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 methods have been developed to characterise these types of heterogeneity [4, 5]. However, there is a limited availability of computational tools tailored to study more subtle variability within seemingly homogeneous cell populations. This variability can reflect deterministic or stochastic events that regulate gene expression and, among others, has been reported to increase prior to cell fate decisions [6] as well as during ageing [7].

This article complements existing scRNA-seq workflows based on the Bioconductor ecosystem (e.g. [8, 9]), providing a detailed framework for transcriptional variability analyses. Firstly, we briefly discuss the sources of variability that arise in scRNA-seq data and the strategies that have been designed to control or attenuate technical noise in these assays. Subsequently, we describe a step-by-step workflow which uses *scater* [10] and *scran* [8] to perform quality control (QC) as well as initial exploratory analyses. To robustly quantify transcriptional variability we use *BASiCS* [11, 12, 13] — a Bayesian hierarchical framework that jointly performs data normalisation, technical noise quantification and downstream analyses, whilst propagating statistical uncertainty across these steps. 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 as well as recommendations to interpret and post-process the model outputs. Finally, through a case study in the context of immune cells, we illustrate how *BASiCS* can be used to 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 on Github. To ensure the reproducibility of this workflow, the analysis environment and all software dependencies are provided as a Docker image [14]. The image can be obtained from Docker Hub.

Sources of variability in scRNA-seq data

Stochastic variability within a seemingly homogeneous cell population — often referred to as transcriptional *noise* — can arise from intrinsic and extrinsic sources [15, 16]. Extrinsic noise refers to stochastic fluctuations induced by different dynamic cellular states (e.g. cell cycle, metabolism, intra/inter-cellular signalling) [17, 18, 19]. In contrast, intrinsic noise arises from stochastic effects on biochemical processes such as transcription and translation [15]. Intrinsic noise can be modulated by genetic and epigenetic modifications (such as mutations, histone modifications, CpG island length and nucleosome positioning) [20, 21, 22] and usually occurs at the gene level [15]. Cell-to-cell gene expression variability estimates derived from scRNA-seq data capture a combination of these effects, as well as deterministic regulatory mechanisms [16]. Moreover, these variability estimates can also be inflated by the technical noise that is typically observed in scRNA-seq data [23].

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 [24]) can be added to each cell's lysate in a (theoretically) known fixed quantity. Spike-ins can assist quality control steps [10], data normalisation [25] and can be used to infer technical noise [23]. Another strategy is to tag individual cDNA molecules using unique molecular identifiers (UMIs) before PCR amplification [26]. 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 [25]). However, despite the benefits associated to the use of spike-ins and UMIs, these are not available for all scRNA-seq protocols [27].

Methods

This step-by-step scRNA-seq workflow is primarily based on the Bioconductor package ecosystem [28]. A graphical overview is provided in Figure 1 and its main components are described below.

Input data

```
library("SingleCellExperiment")
```

We use *SingleCellExperiment* to convert an input matrix of raw read-counts (molecule counts for UMI-based protocols) into a `SingleCellExperiment` object which 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 `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 [28].

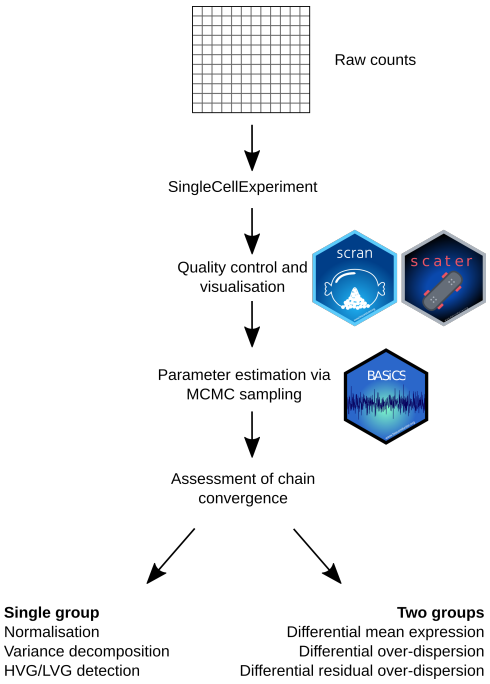

Figure 1. Graphical overview for the scRNA-seq analysis workflow described in this manuscript. Starting from a matrix of expression counts, we use the scater and scran Bioconductor packages to perform QC and initial exploratory analyses. To robustly quantify transcriptional heterogeneity within seemingly 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.

Quality control and exploratory analysis

```
library("scater")
library("scran")
```

An critical step in scRNA-seq analyses is QC, removing low quality samples that may distort downstream analyses. Among others, QC diagnostics can help to identify samples that contain broken cells, that are empty or that contain multiple cells [29]. Moreover, lowly expressed genes for which less reliable information is available are typically also removed. The *OSCA* online book provides an extensive overview on important aspects of how to perform QC of scRNA-seq data, including exploratory analyses [28].

Here, we use the *scater* package [10] to calculate QC metrics for each cell (e.g. total read-count) and gene (e.g. percentage of zeroes across all cells), respectively. Moreover, we use the visualisation tools implemented in *scater* to explore the input dataset and its associated QC diagnostic metrics. For further data exploration we use the *scran* package [8]. *scran* can 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) [30]. Moreover, *scran* enables exploratory analyses of transcriptional variability. For example, it can be used to infer an overall trend between mean expression and the squared coefficent of variation (CV^2) for each gene. To derive variability estimates that are not confounded by this overall trend, *scran* also defines gene-specific DM (distance to the mean) estimates as the distance between CV^2 and a rolling median along the range of mean expression values [31]. DM estimates enable exploratory analyses of cell-to-cell heterogeneity, but a measure of uncertainty is not readily available. As such, gene-specific downstream inference (e.g. differential variability testing) is precluded.

BASiCS - Bayesian Analysis of Single Cell Sequencing data

```
library("BASiCS")
```

The *BASiCS* package uses a Bayesian hierarchical framework that borrows information across all genes and cells to robustly quantify transcriptional variability [32]. Similar to the approach adopted in *scran*, *BASiCS* infers cell-specific global scaling normalisation parameters. However, instead of inferring these as a pre-processing step, *BASiCS* uses an integrated approach in which data normalisation and downstream analyses

are performed simultaneously, thereby propagating statistical uncertainty. To quantify technical noise, the original implementation of *BASiCS* uses information from extrinsic spike-in molecules as control features, but the model has been extended to address situations in which spike-ins are not available [33].

BASiCS summarises expression patterns through gene-specific *mean* (μ_i) and *over-dispersion* (δ_i) parameters. Mean parameters μ_i quantify the overall expression for each gene i across the population of cells under study. In contrast, δ_i captures the excess of variability that is observed with respect to what would be expected in a homogeneous cell population, after taking into account technical noise. This 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.* [33] recently introduced a joint prior specification for these parameters. This joint prior formulation has been observed to improve posterior inference when the data is less informative (e.g. small sample size, lowly expressed genes) and can be used to derive gene-specific *residual over-dispersion* parameters ϵ_i that are not confounded by mean expression. Similar to DM values implemented in *scran*, these are defined as deviations with respect to an overall regression trend that captures the relationship between mean and over-dispersion values.

Within a population of cells, *BASiCS* decomposes the total observed variability in expression measurements into technical and biological components [11]. This enables the identification of *highly variable genes* (HVGs) that capture the major sources of heterogeneity within the analysed cells [23]. HVG detection is often used as feature selection, to identify the input set of genes for subsequent analyses. *BASiCS* can also highlight *lowly variable genes* (LVGs) that exhibit stable expression across the population of cells. These may relate to essential cellular functions and can assist the development of new data normalisation or integration strategies [34].

BASiCS also provides a probabilistic decision rule to perform differential expression analyses between two (or more) pre-specified groups of cells [12, 33]. Whilst several differential expression tools have been proposed for scRNA-seq data (e.g. [35, 36]), some evidence suggests that these do not generally outperform popular bulk RNA-seq tools [37]. Moreover, most of these methods are only designed to uncover changes in overall expression, ignoring the more complex patterns that can arise at the single cell level [38]. Instead, *BASiCS* embraces the high granularity of scRNA-seq data, uncovering changes in cell-to-cell expression variability that are not confounded by differences in technical noise or in overall expression.

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

As a case study, we use scRNA-seq data generated for CD4$^+$ T cells using the C1 Single-Cell Auto Prep System (Fluidigm®). Martinez-Jimenez *et al.* profiled naive (hereafter also referred to as unstimulated) and activated (3 hours using *in vitro* antibody stimulation) CD4$^+$ T cells from young and old animals across two mouse strains to study changes in expression variability during ageing and upon immune activation [7]. They extracted naive or effector memory CD4$^+$ T cells from spleens of young or old animals, obtaining purified populations using either magnetic-activated cell sorting (MACS) or fluorescence activated cell sorting (FACS). External ERCC spike-in RNA [24] was added to aid the quantification of technical variability across all cells and all experiments were performed in replicates (also referred to as batches) to control for batch effects.

Obtaining the data

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

```
if (!file.exists("downloads/raw_data.txt")) {
  website <- "https://www.ebi.ac.uk/arrayexpress/files/E-MTAB-4888/"
  file <- "E-MTAB-4888.processed.1.zip"
  destfile <- "downloads/raw_data.txt.zip"
  download.file(
    paste0(website, file),
    destfile = destfile
  )
  unzip("downloads/raw_data.txt.zip", exdir = "downloads")
  file.remove("downloads/raw_data.txt.zip")
}

# Read in raw data
CD4_raw <- read.table("downloads/raw_data.txt", header = TRUE, sep = "\t")
CD4_raw <- as.matrix(CD4_raw)
```

The input matrix contains data for 1,513 cells and 31,181 genes (including 92 ERCC spike-ins). Information about experimental conditions and other metadata is available under the same accession number.

```
if (!file.exists("downloads/metadata_file.txt")) {
  website <- "https://www.ebi.ac.uk/arrayexpress/files/E-MTAB-4888"
  file <- "E-MTAB-4888.additional.1.zip"
  download.file(
    paste0(website, file),
    destfile = "downloads/metadata.txt.zip"
  )
  unzip("downloads/metadata.txt.zip", exdir = "downloads")
  file.remove("downloads/metadata.txt.zip")
}

CD4_metadata <- read.table(
  "downloads/metadata_file.txt",
  header = TRUE,
  sep = "\t"
)

# Save sample identifiers as rownames
rownames(CD4_metadata) <- CD4_metadata$X
```

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

The data and metadata described above are then converted into a `SingleCellExperiment` object. For this purpose, we first separate the input matrix of expression counts into separate matrices associated to intrinsic genes and external spike-ins. Within the `SingleCellExperiment` object, the latter is stored separately through as an alternative experiment (see `?altExp`).

```
# Separate intrinsic from ERCC counts
bio_counts <- CD4_raw[!grepl("ERCC", rownames(CD4_raw)), ]
spike_counts <- CD4_raw[grepl("ERCC", rownames(CD4_raw)), ]
# Generate the SingleCellExperiment object
sce_CD4_all <- SingleCellExperiment(
  assays = list(counts = as.matrix(bio_counts)),
  colData = CD4_metadata[colnames(CD4_raw), ]
)
# Add read-counts for spike-ins as an alternative experiment
altExp(sce_CD4_all, "spike-ins") <- SummarizedExperiment(
  assays = list(counts = spike_counts)
)
```

Hereafter our analysis focuses on naive and activated $CD4^+$ T cells obtained from young *Mus musculus domesticus* animals, purified using MACS-based cell sorting. Here, we extract these 146 samples.

```
ind_select <- sce_CD4_all$Strain == "Mus musculus domesticus" &
  sce_CD4_all$Age == "Young" &
  sce_CD4_all$Celltype == "MACS-purified Naive"
sce_naive_active <- sce_CD4_all[, ind_select]
sce_naive_active
```

```
## class: SingleCellExperiment
## dim: 31089 146
## metadata(0):
## assays(1): counts
## rownames(31089): ENSMUSG00000000001 ENSMUSG00000000003 ...
##   ENSMUSG00000106668 ENSMUSG00000106670
## rowData names(0):
## colnames(146): do6113 do6118 ... do6493 do6495
## colData names(6): X Strain ... Individuals Celltype
## reducedDimNames(0):
## altExpNames(1): spike-ins
```

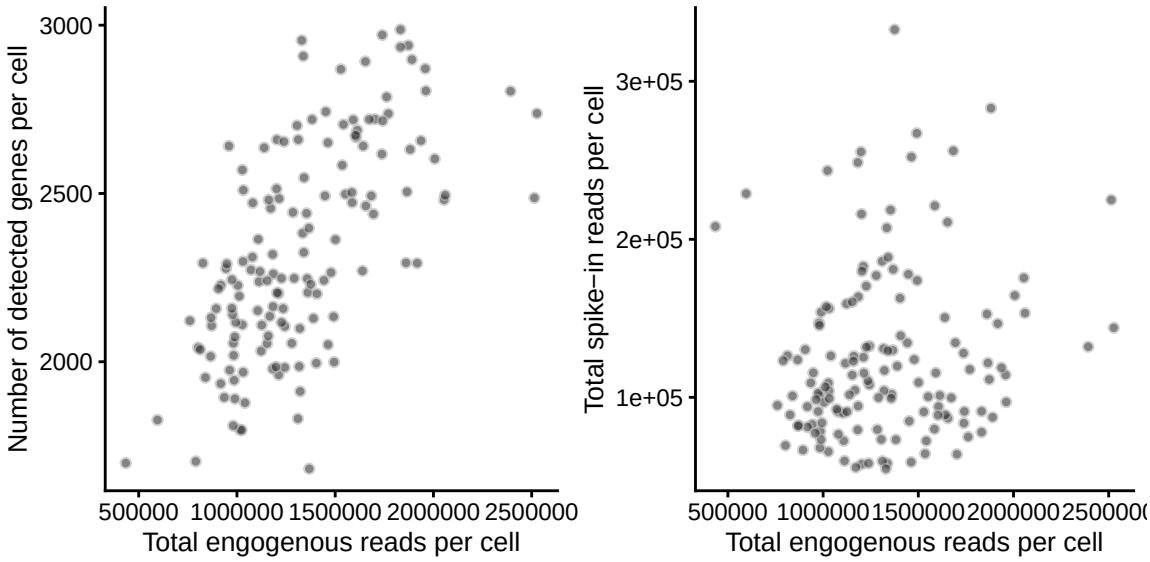

Figure 2. Cell-level QC metrics. The total number of endogenous read-counts (excludes non-mapped and intronic reads) is plotted against the total number of detected genes (left) and the total number of spike-in read-counts (right).

QC and exploratory analysis

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

```
# Calculate and plot per cell QC metrics
sce_naive_active <- addPerCellQC(sce_naive_active, use_altexps = TRUE)
p_cellQC1 <- plotColData(
  sce_naive_active,
  x = "sum",
  y = "detected") +
  xlab("Total engogenous reads per cell") +
  ylab("Number of detected genes per cell")
p_cellQC2 <- plotColData(
  sce_naive_active,
  x = "sum",
  y = "altexps_spike-ins_sum") +
  xlab("Total engogenous reads per cell") +
  ylab("Total spike-in reads per cell")
multiplot(p_cellQC1, p_cellQC2, cols = 2)
```

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

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

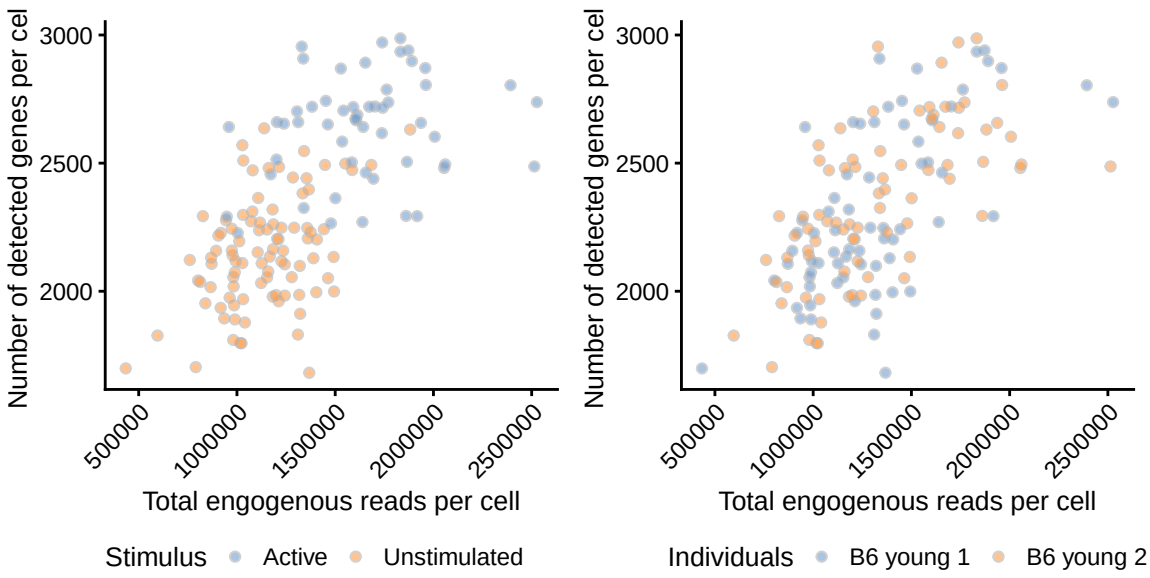

Figure 3. Cell-level QC metrics according to cell-level metadata. The total number of endogenous reads (excludes non-mapped and intronic reads) is plotted against the total number of detected genes. Colour indicates the experimental condition (left) and animal of origin (right) for each cell.

```
    sce_naive_active,
    x = "sum",
    y = "detected",
    colour_by = "Individuals") +
  xlab("Total engogenous reads per cell") +
  ylab("Number of detected genes per cell") +
  theme(
    legend.position = "bottom",
    axis.text.x = element_text(angle = 45, hjust = 1))
multiplot(p_stimulus, p_batch, cols = 2)
```

To further explore the underlying structure of the data, we compute global scaling normalisation factors using *scran* and perform a principal component analysis (PCA) of log-transformed normalised expression counts using *scater*. As seen in Figure 4, this analysis suggests the absence of strong batch effects.

```
# Global scaling normalisation
sce_naive_active <- computeSumFactors(sce_naive_active)
sce_naive_active <- logNormCounts(sce_naive_active)
# PCA
sce_naive_active <- runPCA(sce_naive_active)
p_stimulus <- plotPCA(sce_naive_active, colour_by = "Stimulus") +
  theme(legend.position = "bottom")
p_batch <- plotPCA(sce_naive_active, colour_by = "Individuals") +
  theme(legend.position = "bottom")
multiplot(p_stimulus, p_batch, cols = 2)
```

In addition to cell-specific QC, we also recommend to apply a gene filtering step prior to the use of *BASiCS*. The purpose of this filter is to remove lowly expressed genes that were largely undetected through sequencing and for which variability estimates are less reliable. Here, we remove all genes that are not detected in at least 5 cells across both experimental conditions or for which their average read count is below 1. These thresholds can vary across datasets and might be informed by gene-specific QC metrics such as those shown in 5.

```
# Calculate per gene QC metrics
sce_naive_active <- addPerFeatureQC(sce_naive_active, exprs_values = "counts")
# Remove genes with zero total counts across all cells
sce_naive_active <- sce_naive_active[rowData(sce_naive_active)$detected != 0, ]
# Transform 'detected' metadata into number of cells
rowData(sce_naive_active)$detected_cells <-
  rowData(sce_naive_active)$detected * ncol(sce_naive_active) / 100
```

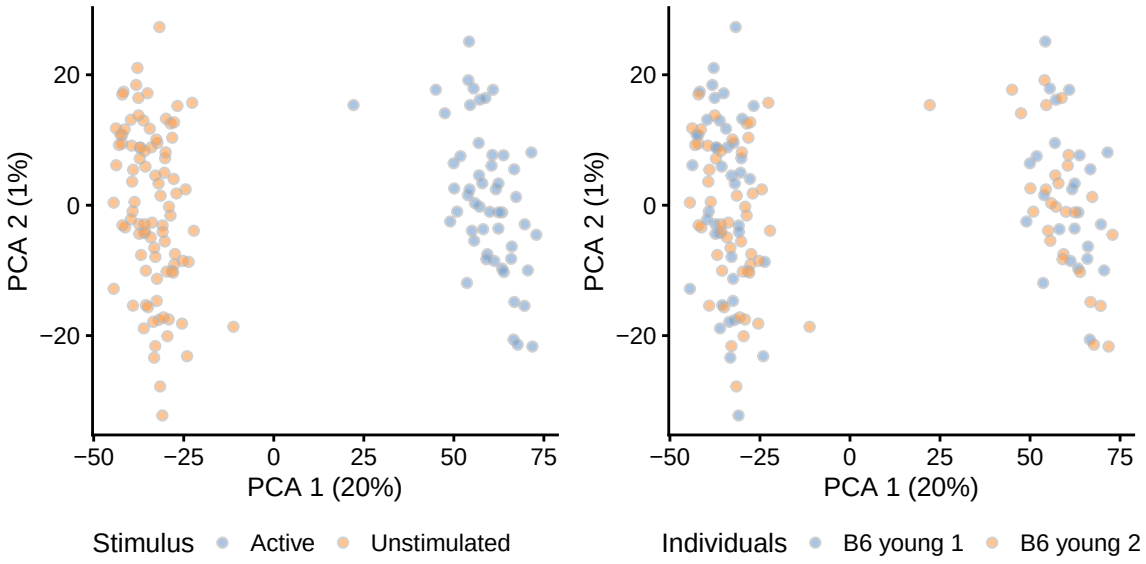

Figure 4. First two principal components of log-transformed expression counts after scran normalisation. Colour indicates the experimental condition (left) and animal of origin (right) for each cell.

```
# Define inclusion criteria
rowData(sce_naive_active)$include_gene <- rowData(sce_naive_active)$mean >= 1 &
  rowData(sce_naive_active)$detected_cells >= 5

plotRowData(
    sce_naive_active,
    x = "detected_cells",
    y = "mean",
    colour_by = "include_gene") +
  xlab("Total engogenous reads per cell") +
  ylab("Number of detected genes per cell") +
  scale_x_log10() +
  scale_y_log10() +
  theme(
    legend.position = "bottom",
    axis.text.x = element_text(angle = 45, hjust = 1)) +
geom_vline(xintercept = 5, linetype = "dashed", col = "grey60") +
geom_hline(yintercept = 1, linetype = "dashed", col = "grey60")
```

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

Subsequently we also recommend to remove spike-in molecules that were not captured through sequencing.

```
detected_spikes <- rowSums(assay(altExp(sce_naive_active)) > 0) > 0
altExp(sce_naive_active) <- altExp(sce_naive_active)[detected_spikes, ]
```

The final dataset used in subsequent analyses contains 146 cells, 8953 genes and 58 spike-ins.

BASiCS analysis - input data

Here, we apply the *BASiCS* model separately to cells from each experimental condition (93 naive and 53 activated cells). Separate `SingleCellExperiment` objects are created for each group of cells.

```
sce_naive <- sce_naive_active[, sce_naive_active$Stimulus == "Unstimulated"]
sce_active <- sce_naive_active[, sce_naive_active$Stimulus == "Active"]
```

BASiCS requires the user to update these objects with additional information, using a specific format. Firstly, if multiple batches of sequenced cells are available (e.g. multiple donors from which cells were extracted or

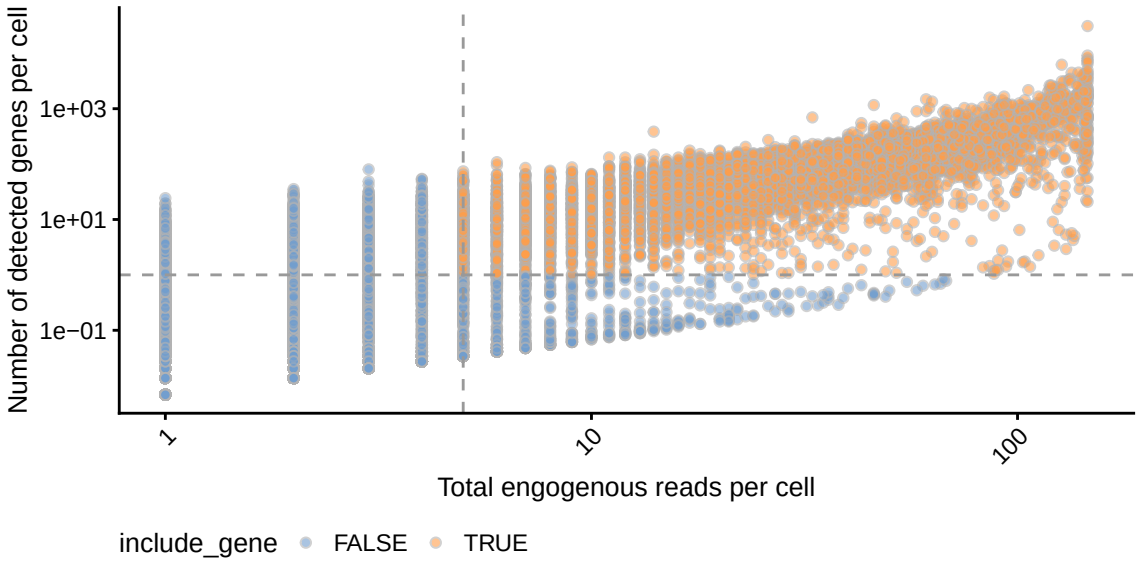

Figure 5. Average read-count for each gene is plotted against the number of cells in which that gene was detected. Dashed grey lines are shown at the thresholds below which genes are removed.

multiple sequencing batches from the same experimental condition), this information must be included under the `BatchInfo` label as part of the cell-level metadata.

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

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

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

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

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

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

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

Based on this information, the calculation above proceeds as follows

```
# Moles per micro litre
ERCC_mmul <- ERCC_conc$concentration.in.Mix.1..attomoles.ul. * (10^(-18))
# Molecule count per micro litre (1 mole comprises 6.02214076 x 10^{23} molecules)
ERCC_countmul <- ERCC_mmul * (6.02214076 * (10^23))
```

```
# Application of the dilution factor (1:50,000)
ERCC_count <- ERCC_countmul / 50000
# Multiplying by the volume added into each well
ERCC_count_final <- ERCC_count * 0.009
```

To update the `sce_naive` and `sce_active` objects, the user must create a `data.frame` whose first column contains the labels associated to the spike-in molecule (e.g. ERCC-00130) and whose second column contains the input number of molecules calculated above. This information is added as metadata for `altExp(sce_naive)` and `altExp(sce_active)`, respectively.

```
SpikeInput <- data.frame(
  Names = ERCC_conc$ERCC.ID,
  count = ERCC_count_final
)
# Exclude spike-ins not included in the input SingleCellExperiment objects
SpikeInput <- subset(SpikeInput, Names %in% rownames(altExp(sce_naive)))

metadata(sce_naive)$SpikeInput <- SpikeInput
metadata(sce_active)$SpikeInput <- SpikeInput
```

BASiCS analysis - parameter estimation

Parameter estimation is implemented in the `BASiCS_MCMC` function using an adaptive Metropolis within Gibbs algorithm [39]. 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 will be stored).
- `Burn`: the initial number of MCMC iterations to be discarded.
- `Regression`: if `TRUE` a join prior is assigned to μ_i and δ_i [33], and residual over-dispersion values ϵ_i are inferred. Alternatively, independent log-normal priors are assigned to μ_i and δ_i [12].
- `WithSpikes`: if `TRUE` information from spike-in molecules is used to aid data normalisation and to quantify technical noise.

As a default, we recommend to use `Regression = TRUE` as we have observed that the joint prior introduced by Eling *et al.* leads to improved inference for small sample sizes and lowly expressed genes. Moreover, the joint prior formulation enables users to obtain a measure of transcriptional variability that is not confounded by mean expression. Additional optional parameters can be used to store the generated output (`StoreChains`, `StoreDir`, `RunName`) and to monitor the progress of the algorithm (`PrintProgress`).

Here, we run the MCMC sampler separately for naive and activated cells. We use 40,000 iterations, discarding the initial 20,000 iterations. We recommend this setting as a default choice, as we have observed it to ensure good convergence for the algorithm. However, for large datasets and less sparse datasets, a lower number of iterations may be sufficient. Practical guidance about convergence diagnostics is provided in the next section.

```
MCMC_naive <- BASiCS_MCMC(
  Data = sce_naive,
  N = 40000,
  Thin = 20,
  Burn = 20000,
  Regression = TRUE,
  WithSpikes = TRUE,
  StoreChains = TRUE,
  StoreDir = "rds/",
  RunName = "naive"
)

MCMC_active <- BASiCS_MCMC(
  Data = sce_active,
  N = 40000,
```

```
  Thin = 20,
  Burn = 20000,
  Regression = TRUE,
  WithSpikes = TRUE,
  StoreChains = TRUE,
  StoreDir = "rds/",
  RunName = "active"
)
```

This sampler runs for 167 minutes on a 1.4 GHz Intel Core i5 procesor with 4GB RAM and produces a `BASiCS_Chain` data object. For comparison, this sampler runs for 97 minutes on a 3.4 GHz Intel Core i7 procesor with 16GB RAM. For convenience, the MCMC chain can be obtained online at https://git.ecdf.ed.ac.uk/vallejosgroup/basicsworkflow2020.

```
if (!file.exists("rds/MCMC_naive.rds")) {
  website <- "https://git.ecdf.ed.ac.uk/vallejosgroup/basicsworkflow2020/raw/master/"
  file <- "MCMC_naive.rds"
  download.file(
    paste0(website, file),
    destfile = "rds/MCMC_naive.rds"
  )
}
if (!file.exists("rds/MCMC_active.rds")) {
  website <- "https://git.ecdf.ed.ac.uk/vallejosgroup/basicsworkflow2020/raw/master/"
  file <- "MCMC_active.rds"
  download.file(
    paste0(website, file),
    destfile = "rds/MCMC_active.rds"
  )
}
MCMC_naive <- readRDS("rds/MCMC_naive.rds")
MCMC_active <- readRDS("rds/MCMC_active.rds")
```

The output from `BASiCS_MCMC` is a `BASiCS_Chain` object which contains a list of matrices whose rows store the MCMC draws for each set of parameters (e.g. μ_i's). These can be accessed using the `displayChainBASiCS` function. For example, the following code displays the first 2 MCMC draws for mean expression parameters μ_i associated to the first 3 genes.

```
displayChainBASiCS(MCMC_naive, Param = "mu")[1:2, 1:3]
```

```
##      ENSMUSG00000000001 ENSMUSG00000000056 ENSMUSG00000000085
## [1,]           9.291713           1.823314           1.237160
## [2,]          15.078529           2.126944           1.494165
```

BASiCS analysis - convergence diagnostics

Before we perform downstream analyses, it is critical to assess the convergence of the MCMC algorithm. Multiple graphical and quantitative convergence diagnostics have been proposed (e.g. [40, 41]). For example, traceplots can be used to visualise the history of posterior draws generated by the algorithm for a specific parameter (e.g. Figure 6, left panel). If the algorithm converged, draws are expected to be stochastic fluctuations around a horizontal trend. As illustrated in Figure 6, histograms can also be used to display the marginal distribution for each parameter. Users should expect these to follow a unimodal distribution. Failure to satisfy these graphical diagnostics suggest that `N` and `Burn` must be increased.

Here, we highlight two ways of assessing the convergence of the MCMC sampler by (i) plotting trace plots, sample densities and autocorrelation, and (ii) plotting the the effective sample size across multiple parameters. Trace plots show the sampled parameter values over time. A chain is likely to have converged if, after burn-in, its trace plot shows a stable mean, with samples drawn around that mean, and without long periods without movement. Sample density plots show the marginal distribution of that parameter. A chain is likely to have converged when the sample density (in form of a histogram) shows a unimodal distribution. The autocorrelation of an MCMC chain is defined as the Pearson correlation between the chain and time-delayed versions of the chain. The difference in time-points is referred to as 'lag'. This helps to assess whether a chain has sampled efficiently from its stationary distribution. High autocorrelation indicates that the obtained samples are not independent, and indicates that we may not have enough information about the posterior

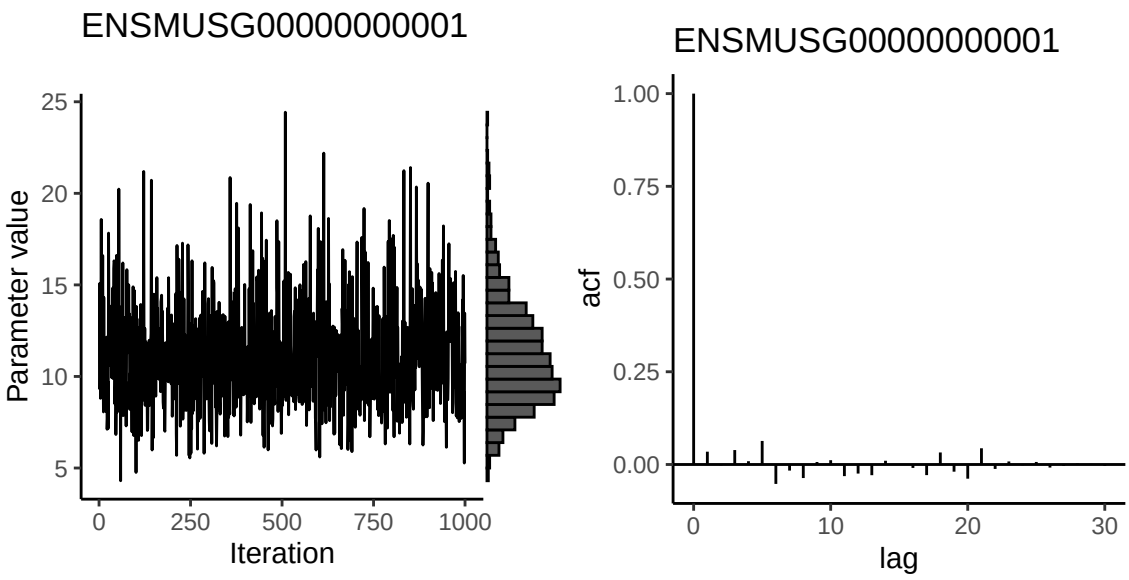

Figure 6. Trace plot, marginal histogram, and autocorrelation function for a gene in naive cells following MCMC sampling. Trace plots should explore the posterior well, without getting stuck in one location or drifting over time towards a region of higher density. High autocorrelation indicates that 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.

distribution of the parameters. The chain is likely to be sampling efficiently if the autocorrelation (except for lag = 1) is small (e.g. < 0.25), as it indicates that the stored samples are largely independent.

Effective sample size is a measure of the number of independent samples generated for a model parameter [42]. Simply, it is defined as the number of samples taken relative to the total autocorrelation. More formally, it is defined as follows:

$$\text{ESS} = \frac{n}{1 + 2\sum_{k=1}^{\infty} \rho(k)}$$

where n is the number of samples and $\rho(k)$ is the autocorrelation at lag k. We can visualise this parameter by plotting histograms of the effective sample size over all genes, and by plotting effective sample size against mean expression or over-dispersion for all genes.

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

```
library("ggplot2")
```

References

[1] Oliver Stegle, Sarah a. Teichmann, and John C. Marioni. Computational and analytical challenges in single-cell transcriptomics. *Nature Reviews Genetics*, 16(3):133–145, jan 2015. ISSN 1471-0056. doi: 10.1038/nrg3833. URL http://www.nature.com/doifinder/10.1038/nrg3833.

[2] Sanjay M. Prakadan, Alex K. Shalek, and David A. Weitz. Scaling by shrinking: empowering single-cell 'omics' with microfluidic devices. *Nature Reviews Genetics*, 18(6):345–361, 2017. ISSN 1471-0056. doi: 10.1038/nrg.2017.15. URL http://www.nature.com/doifinder/10.1038/nrg.2017.15.

[3] Simona Patange, Michelle Girvan, and Daniel R. Larson. Single-cell systems biology: Probing the basic unit of information flow. *Current Opinion in Systems Biology*, 8:7–15, 2018. ISSN 24523100. doi: 10.1016/j.coisb.2017.11.011. URL https://doi.org/10.1016/j.coisb.2017.11.011.

[4] Vladimir Yu Kiselev, Tallulah S. Andrews, and Martin Hemberg. Challenges in unsupervised clustering of single-cell RNA-seq data. *Nature Reviews Genetics 2018*, 2019. ISSN 1471-0064. doi: 10.1038/s41576-018-0088-9. URL https://www.nature.com/articles/s41576-018-0088-9?utm{_}source=feedburner{&}utm{_}medium=feed{&}utm{_}campaign=Feed{%}3A+nrg{%}2Frss{%}2Fcurrent+{%}28Nature+Reviews+Genetics+-+Issue{%}29.

[5] Wouter Saelens, Robrecht Cannoodt, Helena Todorov, and Yvan Saeys. A comparison of single-cell trajectory inference methods. *Nature Biotechnology*, 37(5):547–554, May 2019. ISSN 1087-0156, 1546-1696. doi: 10.1038/s41587-019-0071-9.

[6] Mitra Mojtahedi, Alexander Skupin, Joseph Zhou, Ivan G. Castaño, Rebecca Y. Y. Leong-Quong, Hannah Chang, Kalliopi Trachana, Alessandro Giuliani, and Sui Huang. Cell Fate Decision as High-Dimensional Critical State Transition. *PLOS Biology*, 14(12):e2000640, December 2016. ISSN 1545-7885. doi: 10.1371/journal.pbio.2000640.

[7] Celia P. Martinez-Jimenez, Nils Eling, Hung-Chang Chen, Catalina A Vallejos, Aleksandra A Kolodziejczyk, Frances Connor, Lovorka Stojic, Timothy F Rayner, Michael J T Stubbington, Sarah A Teichmann, Maike de la Roche, John C Marioni, and Duncan T Odom. Aging increases cell-to-cell transcriptional variability upon immune stimulation. *Science*, 1436:1433–1436, 2017. doi: 10.1126/science.aah4115.

[8] Aaron T. L. Lun, Davis J. McCarthy, and John C. Marioni. A step-by-step workflow for basic analyses of single-cell RNA-seq data. *F1000Research*, 5(2122), 2016. ISSN 2046-1402. doi: 10.12688/f1000research.9501.1.

[9] Beomseok Kim, Eunmin Lee, and Jong Kyoung Kim. Analysis of Technical and Biological Variabilityin Single-Cell RNA Sequencing. In *Computational Methods for Single-Cell Data Analysis*, volume 1935, pages 25–43. 2019. ISBN 978-1-4939-9056-6. doi: 10.1007/978-1-4939-9057-3. URL http://www.ncbi.nlm.nih.gov/pubmed/30758827{%}0Ahttp://link.springer.com/10.1007/978-1-4939-9057-3{_}12{%}0Ahttp://link.springer.com/10.1007/978-1-4939-9057-3.

[10] Davis J. McCarthy, Kieran R. Campbell, Aaron T.L. Lun, and Quin F. Wills. Scater: Pre-processing, quality control, normalization and visualization of single-cell RNA-seq data in R. *Bioinformatics*, 33(8):1179–1186, 2017. ISSN 14602059. doi: 10.1093/bioinformatics/btw777.

[11] Catalina A. Vallejos, John C. Marioni, and Sylvia Richardson. BASiCS: Bayesian analysis of single-cell sequencing data. *PLOS Computational Biology*, 11:e1004333, 2015. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1004333. URL http://dx.plos.org/10.1371/journal.pcbi.1004333.

[12] Catalina A Vallejos, Sylvia Richardson, and John C Marioni. Beyond comparisons of means: understanding changes in gene expression at the single-cell level. *Genome Biology*, 17(70), 2016. doi: 10.1101/035949. URL http://biorxiv.org/content/early/2016/01/05/035949.abstract.

[13] Nils Eling, Arianne C. Richard, Sylvia Richardson, John C. Marioni, and Catalina A. Vallejos. Robust expression variability testing reveals heterogeneous T cell responses. *bioRxiv*, page 237214, 2017. doi: 10.1101/237214. URL https://www.biorxiv.org/content/early/2017/12/21/237214.full.pdf+html.

[14] Carl Boettiger. An introduction to Docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1): 71–79, January 2015. ISSN 0163-5980. doi: 10.1145/2723872.2723882.

[15] Michael B Elowitz, Arnold J Levine, Eric D Siggia, and Peter S Swain. Stochastic gene expression in a single cell. *Science*, 297(5584):1183–1186, 2002. ISSN 1095-9203. doi: 10.1126/science.1070919. URL http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve{&}db=PubMed{&}dopt=Citation{&}list{_}uids=12183631.

[16] Nils Eling, Michael D. Morgan, and John C. Marioni. Challenges in measuring and understanding biological noise. *Nature Reviews Genetics*, 20(9):536–548, September 2019. ISSN 1471-0056, 1471-0064. doi: 10.1038/s41576-019-0130-6.

[17] Cristopher J. Zopf, Katie Quinn, Joshua Zeidman, and Narendra Maheshri. Cell-Cycle Dependence of Transcription Dominates Noise in Gene Expression. *PLoS Computational Biology*, 9(7):1–12, 2013. ISSN 1553734X. doi: 10.1371/journal.pcbi.1003161.

[18] Kazunari Iwamoto, Yuki Shindo, and Koichi Takahashi. Modeling Cellular Noise Underlying Heterogeneous Cell Responses in the Epidermal Growth Factor Signaling Pathway. *PLoS Computational Biology*, 12(11):1–18, 2016. ISSN 15537358. doi: 10.1371/journal.pcbi.1005222.

[19] Daniel J. Kiviet, Philippe Nghe, Noreen Walker, Sarah Boulineau, Vanda Sunderlikova, and Sander J. Tans. Stochasticity of metabolism and growth at the single-cell level. *Nature*, 514(7522):376–379, 2014. ISSN 0028-0836. doi: 10.1038/nature13582. URL http://www.nature.com/doifinder/10.1038/nature13582.

[20] James Eberwine and Junhyong Kim. Cellular Deconstruction: Finding Meaning in Individual Cell Variation. *Trends in Cell Biology*, 25(10):569–578, 2015. ISSN 18793088. doi: 10.1016/j.tcb.2015.07.004. URL http://dx.doi.org/10.1016/j.tcb.2015.07.004.

[21] Andre J. Faure, Jörn M. Schmiedel, and Ben Lehner. Systematic Analysis of the Determinants of Gene Expression Noise in Embryonic Stem Cells. *Cell Systems*, 5(5):471–484, 2017. ISSN 24054720. doi: 10.1016/j.cels.2017.10.003.

[22] Michael D. Morgan and John C. Marioni. CpG island composition differences are a source of gene expression noise indicative of promoter responsiveness. *Genome Biology*, 19(1):1–13, 2018. ISSN 1474760X. doi: 10.1186/s13059-018-1461-x.

[23] Philip Brennecke, Simon Anders, Jong Kyoung Kim, Aleksandra A Kołodziejczyk, Xiuwei Zhang, Valentina Proserpio, Bianka Baying, Vladimir Benes, Sarah a Teichmann, John C Marioni, and Marcus G Heisler. Accounting for technical noise in single-cell RNA-seq experiments., 2013. ISSN 1548-7105. URL http://www.ncbi.nlm.nih.gov/pubmed/24056876.

[24] External RNA Controls Consortium. Proposed methods for testing and selecting the ERCC external RNA controls. *BMC Genomics*, 6(June 2004):150, 2005. ISSN 1471-2164. doi: 10.1186/1471-2164-6-150.

[25] Catalina A Vallejos, Davide Risso, Antonio Scialdone, Sandrine Dudoit, and John C Marioni. Normalizing single-cell RNA sequencing data: challenges and opportunities. *Nature Methods*, 14(6):565–571, 2017. ISSN 1548-7091. doi: 10.1038/nmeth.4292. URL http://www.nature.com/doifinder/10.1038/nmeth.4292.

[26] Saiful Islam, Amit Zeisel, Simon Joost, Gioele La Manno, Pawel Zajac, Maria Kasper, Peter Lönnerberg, and Sten Linnarsson. Quantitative single-cell RNA-seq with unique molecular identifiers. *Nature Methods*, 11(2):163–166, February 2014. ISSN 1548-7091, 1548-7105. doi: 10.1038/nmeth.2772.

[27] Ashraful Haque, Jessica Engel, Sarah A. Teichmann, and Tapio Lönnberg. A practical guide to single-cell RNA-sequencing for biomedical research and clinical applications. *Genome Medicine*, 9(1):75, December 2017. ISSN 1756-994X. doi: 10.1186/s13073-017-0467-4.

[28] Robert A. Amezquita, Vince J. Carey, Lindsay N. Carpp, Ludwig Geistlinger, Aaron T. L. Lun, Federico Marini, Kevin Rue-Albrecht, Davide Risso, Charlotte Soneson, Levi Waldron, Hervé Pagès, Mike Smith, Wolfgang Huber, Martin Morgan, Raphael Gottardo, and Stephanie C. Hicks. Orchestrating Single-Cell Analysis with Bioconductor. Preprint, Genomics, March 2019.

[29] Tomislav Ilicic, Jong Kyoung Kim, Aleksandra A Kolodziejczyk, Frederik Otzen Bagger, Davis James Mccarthy, John C Marioni, and Sarah A Teichmann. Classification of low quality cells from single-cell RNA-seq data. *Genome Biology*, 17(29):1–15, 2016. ISSN 1474-760X. doi: 10.1186/s13059-016-0888-1. URL http://dx.doi.org/10.1186/s13059-016-0888-1.

[30] Aaron T. L. Lun, Karsten Bach, and John C. Marioni. Pooling across cells to normalize single-cell RNA sequencing data with many zero counts. *Genome Biology*, 17(1):75, 2016. ISSN 1474-760X. doi: 10.1186/s13059-016-0947-7. URL http://genomebiology.biomedcentral.com/articles/10.1186/s13059-016-0947-7.

[31] Aleksandra A. Kolodziejczyk, Jong Kyoung Kim, Jason C.H. Tsang, Tomislav Ilicic, Johan Henriksson, Kedar N. Natarajan, Alex C. Tuck, Xuefei Gao, Marc Bühler, Pentao Liu, John C. Marioni, and Sarah A. Teichmann. Single cell RNA-sequencing of pluripotent states unlocks modular transcriptional variation. *Cell Stem Cell*, 17:471–485, 2015. ISSN 19345909. doi: 10.1016/j.stem.2015.09.011. URL http://linkinghub.elsevier.com/retrieve/pii/S193459091500418X.

[32] Catalina A. Vallejos, John C. Marioni, and Sylvia Richardson. BASiCS: Bayesian analysis of single-cell sequencing data. *PLOS Computational Biology*, 11(6):e1004333, 2015. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1004333. URL http://dx.plos.org/10.1371/journal.pcbi.1004333.

[33] Nils Eling, Arianne C. Richard, Sylvia Richardson, John C. Marioni, and Catalina A. Vallejos. Correcting the Mean-Variance Dependency for Differential Variability Testing Using Single-Cell RNA Sequencing Data. *Cell Systems*, 7(3):1–11, 2018. ISSN 24054712. doi: 10.1016/j.cels.2018.06.011. URL https://linkinghub.elsevier.com/retrieve/pii/S2405471218302783.

[34] Yingxin Lin, Shila Ghazanfar, Dario Strbenac, Andy Wang, Ellis Patrick, David M Lin, Terence Speed, Jean Y H Yang, and Pengyi Yang. Evaluating stably expressed genes in single cells. *GigaScience*, 8(9):giz106, September 2019. ISSN 2047-217X. doi: 10.1093/gigascience/giz106.

[35] Peter V Kharchenko, Lev Silberstein, and David T Scadden. Bayesian approach to single-cell differential expression analysis. *Nature methods*, 11(7):740–2, jul 2014. ISSN 1548-7105. doi: 10.1038/nmeth.2967. URL http://www.ncbi.nlm.nih.gov/pubmed/24836921.

[36] Greg Finak, Andrew McDavid, Masanao Yajima, Jingyuan Deng, Vivian Gersuk, Alex K. Shalek, Chloe K. Slichter, Hannah W. Miller, M. Juliana McElrath, Martin Prlic, Peter S. Linsley, and Raphael Gottardo. MAST: A flexible statistical framework for assessing transcriptional changes and characterizing heterogeneity in single-cell RNA sequencing data. *Genome Biology*, 16(1):278, December 2015. ISSN 1474-760X. doi: 10.1186/s13059-015-0844-5.

[37] Charlotte Soneson and Mark D Robinson. Bias, robustness and scalability in single-cell differential expression analysis. *Nature Methods*, 15(4):255–261, April 2018. ISSN 1548-7091, 1548-7105. doi: 10.1038/nmeth.4612.

[38] David Lähnemann, Johannes Köster, Ewa Szczurek, Davis J. McCarthy, Stephanie C. Hicks, Mark D. Robinson, Catalina A. Vallejos, Kieran R. Campbell, Niko Beerenwinkel, Ahmed Mahfouz, Luca Pinello, Pavel Skums, Alexandros Stamatakis, Camille Stephan-Otto Attolini, Samuel Aparicio, Jasmijn Baaijens, Marleen Balvert, Buys de Barbanson, Antonio Cappuccio, Giacomo Corleone, Bas E. Dutilh, Maria Florescu, Victor Guryev, Rens Holmer, Katharina Jahn, Thamar Jessurun Lobo, Emma M. Keizer, Indu Khatri, Szymon M. Kielbasa, Jan O. Korbel, Alexey M. Kozlov, Tzu-Hao Kuo, Boudewijn P.F. Lelieveldt, Ion I. Mandoiu, John C. Marioni, Tobias Marschall, Felix Mölder, Amir Niknejad, Lukasz Raczkowski, Marcel Reinders, Jeroen de Ridder, Antoine-Emmanuel Saliba, Antonios Somarakis, Oliver Stegle, Fabian J. Theis, Huan Yang, Alex Zelikovsky, Alice C. McHardy, Benjamin J. Raphael, Sohrab P. Shah, and Alexander Schönhuth. Eleven grand challenges in single-cell data science. *Genome Biology*, 21(1):31, December 2020. ISSN 1474-760X. doi: 10.1186/s13059-020-1926-6.

[39] Gareth O. Roberts and Jeffrey S. Rosenthal. Examples of Adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349–367, January 2009. ISSN 1061-8600, 1537-2715. doi: 10.1198/jcgs.2009.06134.

[40] Mary Kathryn Cowles and Bradley P. Carlin. Markov chain monte carlo convergence diagnostics: A comparative review. *Journal of the American Statistical Association*, 91(434):883–904, 1996. doi: 10.1080/01621459.1996.10476956. URL https://amstat.tandfonline.com/doi/abs/10.1080/01621459.1996.10476956.

[41] Stephen P. Brooks and Andrew Gelman. General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, 7(4):434–455, 1998. doi: 10.1080/10618600.1998.10474787. URL https://amstat.tandfonline.com/doi/abs/10.1080/10618600.1998.10474787.

[42] Andrew Gelman, John Carlin, Hal Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian Data Analysis*. CRC Press, 2014. ISBN 978-1439840955.