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

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

[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]**MRC Human Genetics Unit, Institute of Genetics & Molecular Medicine, University of Edinburgh, Western General Hospital, Crewe Road, Edinburgh, EH4 2XU, UK**
[4]**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

[*]eling@ebi.ac.uk
[†]catalina.vallejos@igmm.ed.ac.uk

Methods

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

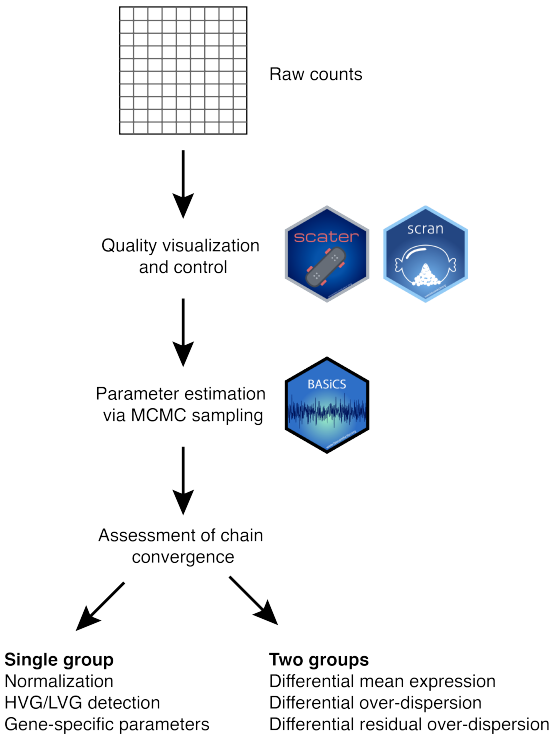

Figure 1. Graphical overview for the scRNA-seq analysis workflow described in this manuscript. Starting from a SingleCellExperiment object, we use the scater and scran Bioconductor packages to perform quality control and initial exploratory analyses. The primary focus of this workflow is to robustly quantify transcriptional heterogeneity within seemingly homogeneous cell populations. For this purpose, we apply the BASiCS Bioconductor package, illustrating how BASiCS can be used to analyse a single or multiple pre-specified groups of cells.

Input data - `SingleCellExperiment`

Here, we use the *SingleCellExperiment* package to convert an input matrix of raw read-counts (molecule counts for UMI-based protocols) into a `SingleCellExperiment` object. Such object can be used to store scRNA-seq data and its associated metadata, such as gene- and cell-specific information. Moreover, when available, the same object can be used to store read-counts for technical spike-in molecules (these can be accessed via the `altExp()` method). A major advantage of using a `SingleCellExperiment` object as the input for scRNA-seq analyses is that it enables interoperability across a large number of Bioconductor packages [1].

Quality control and exploratory analyses - `scater` and `scran`

An critical step in scRNA-seq analyses is to apply quality control (QC) diagnostics, removing low quality samples (wells or droplets, depending on the protocol) that may distort downstream analyses. Among others, QC can help to identify samples that contain broken cells, that are empty or that contain multiple cells [**?**]. Moreover, lowly expressed genes for which less reliable information is available are typically also removed.

The *scater* Bioconductor package was developed to facilitate this process [2]. It can be used to calculate standard QC metrics (e.g. total reads per cell, percentage of mitochondrial reads) and for data visualisation. Here, we primarily use the `calculateQCMetrics` function to calculate cell- and gene-specific QC metrics and the `plotPCA` function to visualise QC metrics, exploring how these interact with available metadata.

The *scran* Bioconductor package offers a variety of functions for low-level scRNA-seq data analysis [3]. While it contains function for doublet detection, and estimation of cell cycle states, we will use it primarily for normalisation in conjunction with the `scater` package [4], and for modelling the mean-variance trend across all genes. The `trendVar` and `decomposeVar` functions will be used to fit a trend between the gene-specific variances and gene-specific mean expression, before decomposing the overall variance into technical and biological components. Furthermore, we will use the `DM` function to calculate the distance of gene-specific squared coefficients of variation (CV^2) to a rolling median along mean expression [5].

Quantifying cell-to-cell transcriptional variability - BASiCS

The *BASiCS* Bioconductor package contains an assembly of functions to estimate and analyse gene- and cell-specific model parameters [6, 7, 8]. `BASiCS` is build upon a hierarchical Baysian framework and as such samples posterior distribution for each model parameter. In mathematical terms, the gene expression count X_{ij} for gene i in cell j is modelled as:

$$\begin{aligned} X_{ij}|\mu_i, \nu_j &\sim \text{Poisson}(\nu_j \mu_i) \\ \nu_j|s_j, \theta &\sim \text{Gamma}(1/\theta, 1/(s_j\theta)) \\ \rho_{ij}|\delta_i &\sim \text{Gamma}(1/\delta_i, 1/\delta_i) \end{aligned}$$

where μ_i explains the gene's mean expression, ν_j the technical effect characterised by the mRNA capture efficiency s_j and the unexplained technical noise parameter θ. Here, ρ_{ij} the biological random effect fluctuating around the gene-specific over-dispersion hyper-parameter δ_i. It is important to note that, in further analyses, μ_i represents the mean expression and δ_i the biological over-dispersion of each gene.

Due to a strong association between mean expression and biological over-dispersion, the model has been extended to correct for such confounding effect. To this end, the prior distribution for the over-dispersion parameters has been changed to:

$$\delta_i|\mu_i \sim \text{log-t}_\eta\left(\text{f}(\mu_i), \sigma^2\right).$$

where $\text{f}(\mu_i)$ describes a smooth regression trend between the over-dispersion and the mean expression parameters. This extension introduced the residual over-dispersion parameters $\epsilon_i = \delta_i - \text{f}(\mu_i)$ that do not scale with mean expression [8].

From a data analysis perspective, the `BASiCS_MCMC` function is the heart of the `BASiCS` package, and can be run in four different settings (see Table 1).

Table 1. Four settings that can be used to run the `BASiCS_MCMC` function.

	No regression	Regression
Using spike-in reads	`WithSpikes = TRUE`	`WithSpikes = TRUE`
	`Regression = FALSE`	`Regression = TRUE`
No spike-ins available	`WithSpikes = FALSE`	`WithSpikes = FALSE`
	`Regression = FALSE`	`Regression = TRUE`

If spike-in counts are availabe and should be used to estimate technical noise, the parameter `WithSpikes` is set to `TRUE` (default). If the regression between over-dispersion and mean expression should be performed, the `Regression` parameters is set to `TRUE` (default). If the user decides to set `Regression = FALSE`, `BASiCS` will not estimate the regression trend, and will not supply the residual over-dispersion parameters ϵ_i.

The `BASiCS_MCMC` function returns a `BASiCS_Chain` object, which can be used for further downstream analyses, many of which are detailed in this workflow. These objects contain draws from Markov chain Monte Carlo (MCMC) samplers, which are used to infer the posterior distribution over the model parameters [9]. Briefly, the posterior distribution quantifies how probable different parameter values are given the observed data. However, before assessing the posterior distribution, we must first ensure that the MCMC sampler has converged to its stationary distribution, and has sampled efficiently from this distribution [10]. If these conditions are not met, then the estimated parameters may be inaccurate. The *coda* CRAN package contains a variety of functions to assess the convergence of a sampled MCMC chain. To use `coda` functions, the individual chains returned by `BASiCS` need to be transformed into a MCMC object that `coda` recognises using the `coda::mcmc` function. `BASiCS` also offers a number of functions to visualise and assess the convergence of MCMC chains. In particular, we will use `BASiCS_EffectiveSize` and `BASiCS_DiagPlot` to calculate and visualise the effective sample size generated by the MCMC samplers.

Other steps [title tbc]

The *goseq* Bioconductor package offers functions to detect the enrichment of gene ontologies (GOs) among user-specified gene sets [11]. Furthermore, `goseq` corrects for gene length biases, which is useful for full length scRNA-seq data as highlighted in the first section. In this workflow, we will use `goseq` to detect GO enrichment among differentially expressed sets of genes.

For downstream analysis, such as GO enrichment analysis or the biological interpretation of individual genes, we need to (i) link each gene's ID to its symbol and (ii) calculate each gene's length. For the first task, the *biomaRt* Bioconductor package annotates a wide range of gene and gene product identifiers [12] by accessing

the BioMart software suite (http://www.biomart.org). We can use `biomaRt` to link the **Mus musculus** gene IDs and to their gene symbols (also referred to as 'gene name'):

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] A. F. M. Smith and G. O. Roberts. Bayesian Computation Via the Gibbs Sampler and Related Markov Chain Monte Carlo Methods. *Journal of the Royal Statistical Society: Series B (Methodological)*, 55(17):3–23, September 1993. ISSN 00359246. doi: 10.1111/j.2517-6161.1993.tb01466.x.

[10] 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, June 1996. ISSN 0162-1459, 1537-274X. doi: 10.1080/01621459.1996.10476956.

[11] Matthew D Young, Matthew J Wakefield, Gordon K Smyth, and Alicia Oshlack. Gene ontology analysis for RNA-seq: accounting for selection bias. *Genome Biology*, 11, 2010. URL http://genomebiology.com/2010/11/2/R14.

[12] Steffen Durinck, Yves Moreau, Arek Kasprzyk, Sean Davis, Bart De Moor, Alvis Brazma, and Wolfgang Huber. BioMart and Bioconductor: A powerful link between biological databases and microarray data analysis. *Bioinformatics*, 21(16):3439–3440, 2005. ISSN 13674803. doi: 10.1093/bioinformatics/bti525.