Normalizing single-cell RNA-seq data using spike-in information

Aaron T. L. Lun[1]***, Davis J. McCarthy***[2,3] ***and John C. Marioni***[1,2,4]

[1]Cancer Research UK Cambridge Institute, Li Ka Shing Centre, Robinson Way, Cambridge CB2 0RE, United Kingdom
[2]EMBL European Bioinformatics Institute, Wellcome Genome Campus, Hinxton, Cambridge CB10 1SD, United Kingdom
[3]St Vincent's Institute of Medical Research, 41 Victoria Parade, Fitzroy, Victoria 3065, Australia
[4]Wellcome Trust Sanger Institute, Wellcome Genome Campus, Hinxton, Cambridge CB10 1SA, United Kingdom

2019-01-09

1 Motivation

Scaling normalization strategies for scRNA-seq data can be broadly divided into two classes. The first class assumes that there exists a subset of genes that are not DE between samples, as described in the previous workflows. The second class uses the fact that the same amount of spike-in RNA was added to each cell (Lun et al. 2017). Differences in the coverage of the spike-in transcripts can only be due to cell-specific biases, e.g., in capture efficiency or sequencing depth. Scaling normalization is then applied to equalize spike-in coverage across cells.

The choice between these two normalization strategies depends on the biology of the cells and the features of interest. If the majority of genes are expected to be DE and there is no reliable

house-keeping set, spike-in normalization may be the only option for removing cell-specific biases. Spike-in normalization should also be used if differences in the total RNA content of individual cells are of interest. In any particular cell, an increase in the amount of endogenous RNA will not increase spike-in coverage (with or without library quantification). Thus, the former will not be represented as part of the bias in the latter, which means that the effects of total RNA content on expression will not be removed upon scaling. With non-DE normalization, an increase in RNA content will systematically increase the expression of all genes in the non-DE subset, such that it will be treated as bias and removed.

2 Setting up the data

2.1 Obtaining the dataset

We demonstrate the use of spike-in normalization on a dataset involving different cell types – namely, mouse embryonic stem cells (mESCs) and mouse embryonic fibroblasts (MEFs) (Islam et al. 2011). The count table was obtained from the NCBI Gene Expression Omnibus (GEO) as a supplementary file using the accession number GSE29087 (http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE29087).

```
library(BiocFileCache)
bfc <- BiocFileCache("raw_data", ask=FALSE)
islam.fname <- bfcrpath(bfc, file.path("ftp://ftp.ncbi.nlm.nih.gov/geo/series",
    "GSE29nnn/GSE29087/suppl/GSE29087_L139_expression_tab.txt.gz"))
```

We load the counts into R, using `colClasses` to speed up `read.table` by pre-defining the type of each column. We also specify the rows corresponding to spike-in transcripts.

```
library(SingleCellExperiment)
counts <- read.table(islam.fname,
    colClasses=c(list("character", NULL, NULL, NULL, NULL, NULL, NULL),
    rep("integer", 96)), skip=6, sep='\t', row.names=1)

is.spike <- grep("SPIKE", rownames(counts))
sce.islam <- SingleCellExperiment(list(counts=as.matrix(counts)))
isSpike(sce.islam, "spike") <- is.spike
dim(sce.islam)
```

```
## [1] 22936    96
```

2.2 Applying quality control

We perform some quality control to remove low-quality cells using the `calculateQCMetrics` function. Outliers are identified within each cell type to avoid issues with systematic differences in the metrics between cell types. The negative control wells do not contain any cells and are useful for quality control (as they *should* manifest as outliers for the various metrics), but need to be removed prior to downstream analysis.

```
library(scater)
sce.islam <- calculateQCMetrics(sce.islam)
sce.islam$grouping <- rep(c("mESC", "MEF", "Neg"), c(48, 44,
4))

libsize.drop <- isOutlier(sce.islam$total_counts, nmads=3, t
ype="lower",
    log=TRUE, batch=sce.islam$grouping)
feature.drop <- isOutlier(sce.islam$total_features_by_count
s, nmads=3, type="lower",
    log=TRUE, batch=sce.islam$grouping)
spike.drop <- isOutlier(sce.islam$pct_counts_spike, nmads=3,
type="higher",
    batch=sce.islam$grouping)

sce.islam <- sce.islam[,!(libsize.drop | feature.drop |
    spike.drop | sce.islam$grouping=="Neg")]
data.frame(ByLibSize=sum(libsize.drop), ByFeature=sum(featur
e.drop),
    BySpike=sum(spike.drop), Remaining=ncol(sce.islam))
```

```
##   ByLibSize ByFeature BySpike Remaining
## 1         4         6      12        77
```

3 Calculating spike-in size factors

We apply the `computeSpikeFactors` method to estimate size factors for all cells. This method computes the total count over all spike-in transcripts in each cell, and calculates size factors to equalize the total spike-in count across cells. Here, we set `general.use=TRUE` as we intend to apply the spike-in factors to all counts.

```
library(scran)
sce.islam <- computeSpikeFactors(sce.islam, general.use=TRU
E)
head(sizeFactors(sce.islam))
```

```
## [1] 1.1486524 1.1274936 0.4285218 1.1011014 0.6646450 1.5
932121
```

```
head(sizeFactors(sce.islam, "spike")) # same as general size
factors.
```

```
## [1] 1.1486524 1.1274936 0.4285218 1.1011014 0.6646450 1.5
932121
```

Running `normalize` will use the spike-in-based size factors to compute normalized log-expression values. Unlike the previous analyses, we do not have to define separate size factors for the spike-in transcripts. This is because the relevant factors are already being used for all genes and spike-in transcripts when `general.use=TRUE`. (The exception is if the experiment uses multiple spike-in sets that behave differently and need to be normalized separately.)

```
sce.islam <- normalize(sce.islam)
```

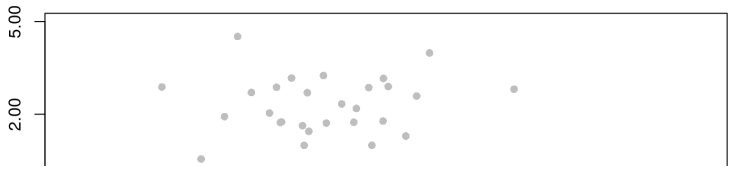
For comparison, we also compute the deconvolution size factors for this data set (Lun, Bach, and Marioni 2016).

```
deconv.sf <- computeSumFactors(sce.islam, sf.out=TRUE, clust
er=sce.islam$grouping)
head(deconv.sf)
```

```
## [1] 0.06424166 0.12172399 0.15547626 0.12876863 0.0568065
1 0.07888561
```

We observe a negative correlation between the two sets of size factors (Figure 1). This is because MEFs contain more endogenous RNA, which reduces the relative spike-in coverage in each library (thereby decreasing the spike-in size factors) but increases the coverage of endogenous genes (thus increasing the deconvolution size factors). If the spike-in size factors were applied to the counts, the expression values in MEFs would be scaled up while expression in mESCs would be scaled down. However, the opposite would occur if deconvolution size factors were used.

```
colours <- c(mESC="red", MEF="grey")
plot(sizeFactors(sce.islam), deconv.sf, col=colours[sce.isla
m$grouping], pch=16,
    log="xy", xlab="Size factor (spike-in)", ylab="Size fact
or (deconvolution)")
legend("bottomleft", col=colours, legend=names(colours), pch
=16)
```



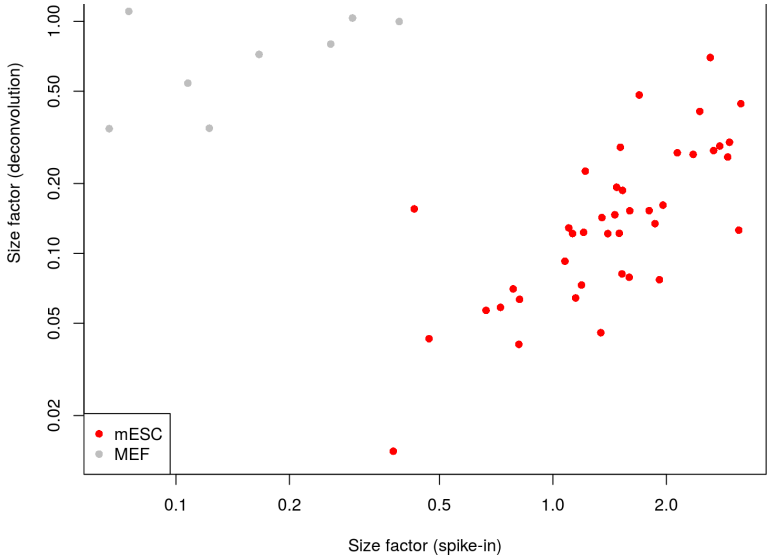


Figure 1: Size factors from spike-in normalization, plotted against the size factors from deconvolution for all cells in the mESC/MEF dataset. Axes are shown on a log-scale, and cells are coloured according to their identity. Deconvolution size factors were computed with small pool sizes owing to the low number of cells of each type.

Whether or not total RNA content is relevant – and thus, the choice of normalization strategy – depends on the biological hypothesis. In the HSC and brain analyses, variability in total RNA across the population was treated as noise and removed by non-DE normalization. This may not always be appropriate if total RNA is associated with a biological difference of interest. For example, Islam et al. (2011) observe a 5-fold difference in total RNA between mESCs and MEFs. Similarly, the total RNA in a cell changes across phases of the cell cycle (Buettner et al. 2015). Spike-in normalization will preserve these differences in total RNA content such that the corresponding biological groups can be easily resolved in downstream analyses.

Comments from Aaron:

- We only use genes with average counts greater than 1 (as specified in `min.mean`) to compute the deconvolution size factors. This avoids problems with discreteness as mentioned in our previous uses of `computeSumFactors`.
- Setting `sf.out=TRUE` will directly return the size factors, rather than a `SingleCellExperiment` object containing those factors. This is more convenient when only the size factors are required for further analysis.

4 Concluding remarks

All software packages used in this workflow are publicly available from the Comprehensive R Archive Network (https://cran.r-project.org (https://cran.r-project.org)) or the Bioconductor project

(http://bioconductor.org (http://bioconductor.org)). The specific version numbers of the packages used are shown below, along with the version of the R installation.

```
sessionInfo()
```

```
## R version 3.5.2 (2018-12-20)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.5 LTS
##
## Matrix products: default
## BLAS: /home/biocbuild/bbs-3.8-bioc/R/lib/libRblas.so
## LAPACK: /home/biocbuild/bbs-3.8-bioc/R/lib/libRlapack.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
##  [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
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices uti
ls     datasets
## [8] methods   base
##
## other attached packages:
##  [1] scRNAseq_1.8.0
##  [2] edgeR_3.24.3
##  [3] Matrix_1.2-15
##  [4] org.Hs.eg.db_3.7.0
##  [5] EnsDb.Hsapiens.v86_2.99.0
##  [6] ensembldb_2.6.3
##  [7] AnnotationFilter_1.6.0
##  [8] DropletUtils_1.2.2
##  [9] pheatmap_1.0.12
## [10] cluster_2.0.7-1
## [11] dynamicTreeCut_1.63-1
## [12] limma_3.38.3
## [13] scran_1.10.2
## [14] scater_1.10.1
## [15] ggplot2_3.1.0
## [16] TxDb.Mmusculus.UCSC.mm10.ensGene_3.4.0
## [17] GenomicFeatures_1.34.1
## [18] org.Mm.eg.db_3.7.0
## [19] AnnotationDbi_1.44.0
## [20] SingleCellExperiment_1.4.1
## [21] SummarizedExperiment_1.12.0
## [22] DelayedArray_0.8.0
## [23] BiocParallel_1.16.5
## [24] matrixStats_0.54.0
## [25] Biobase_2.42.0
## [26] GenomicRanges_1.34.0
## [27] GenomeInfoDb_1.18.1
## [28] IRanges_2.16.0
## [29] S4Vectors_0.20.1
## [30] BiocGenerics_0.28.0
## [31] bindrcpp_0.2.2
## [32] BiocFileCache_1.6.0
```

```
## [33] dbplyr_1.2.2
## [34] knitr_1.21
## [35] BiocStyle_2.10.0
##
## loaded via a namespace (and not attached):
##   [1] readxl_1.2.0             plyr_1.8.4
##   [3] igraph_1.2.2             lazyeval_0.2.1
##   [5] splines_3.5.2            sp_1.3-1
##   [7] digest_0.6.18            htmltools_0.3.6
##   [9] viridis_0.5.1            magrittr_1.5
##  [11] memoise_1.1.0            openxlsx_4.1.0
##  [13] Biostrings_2.50.2        prettyunits_1.0.2
##  [15] colorspace_1.3-2         blob_1.1.1
##  [17] rappdirs_0.3.1           rrcov_1.4-7
##  [19] haven_2.0.0              xfun_0.4
##  [21] dplyr_0.7.8              crayon_1.3.4
##  [23] RCurl_1.95-4.11          bindr_0.1.1
##  [25] survival_2.43-3          zoo_1.8-4
##  [27] glue_1.3.0               gtable_0.2.0
##  [29] zlibbioc_1.28.0          XVector_0.22.0
##  [31] kernlab_0.9-27           car_3.0-2
##  [33] Rhdf5lib_1.4.2           prabclus_2.2-6
##  [35] DEoptimR_1.0-8           HDF5Array_1.10.1
##  [37] abind_1.4-5              VIM_4.7.0
##  [39] scales_1.0.0             sgeostat_1.0-27
##  [41] mvtnorm_1.0-8            DBI_1.0.0
##  [43] GGally_1.4.0             sROC_0.1-2
##  [45] Rcpp_1.0.0               laeken_0.4.6
##  [47] viridisLite_0.3.0        progress_1.2.0
##  [49] foreign_0.8-71           bit_1.1-14
##  [51] mclust_5.4.2             truncnorm_1.0-8
##  [53] vcd_1.4-4                httr_1.4.0
##  [55] fpc_2.1-11.1             RColorBrewer_1.1-2
##  [57] modeltools_0.2-22        NADA_1.6-1
##  [59] flexmix_2.3-14           pkgconfig_2.0.2
##  [61] reshape_0.8.8            XML_3.98-1.16
##  [63] nnet_7.3-12              locfit_1.5-9.1
##  [65] tidyselect_0.2.5         labeling_0.3
##  [67] rlang_0.3.1              reshape2_1.4.3
##  [69] cellranger_1.1.0         munsell_0.5.0
##  [71] tools_3.5.2              RSQLite_2.1.1
##  [73] pls_2.7-0                cvTools_0.3.2
##  [75] evaluate_0.12            stringr_1.3.1
##  [77] yaml_2.2.0               bit64_0.9-7
##  [79] zip_1.0.0                robustbase_0.93-3
##  [81] purrr_0.2.5              biomaRt_2.38.0
##  [83] compiler_3.5.2           beeswarm_0.2.3
##  [85] curl_3.2                 e1071_1.7-0
##  [87] zCompositions_1.1.2      tibble_2.0.0
##  [89] statmod_1.4.30           robCompositions_2.0.9
##  [91] pcaPP_1.9-73             stringi_1.2.4
##  [93] highr_0.7                forcats_0.3.0
##  [95] trimcluster_0.1-2.1      lattice_0.20-38
##  [97] ProtGenerics_1.14.0      pillar_1.3.1
##  [99] BiocManager_1.30.4       lmtest_0.9-36
```

```
## [101] BiocNeighbors_1.0.0      data.table_1.11.8
## [103] cowplot_0.9.4            bitops_1.0-6
## [105] irlba_2.3.2              rtracklayer_1.42.1
## [107] R6_2.3.0                 bookdown_0.9
## [109] KernSmooth_2.23-15       gridExtra_2.3
## [111] rio_0.5.16               vipor_0.4.5
## [113] boot_1.3-20              MASS_7.3-51.1
## [115] assertthat_0.2.0         rhdf5_2.26.2
## [117] withr_2.1.2              GenomicAlignments_1.18.1
## [119] Rsamtools_1.34.0         GenomeInfoDbData_1.2.0
## [121] diptest_0.75-7           hms_0.4.2
## [123] grid_3.5.2               class_7.3-15
## [125] rmarkdown_1.11           DelayedMatrixStats_1.4.0
## [127] carData_3.0-2            mvoutlier_2.0.9
## [129] Rtsne_0.15               ggbeeswarm_0.6.0
```

References

Buettner, F., K. N. Natarajan, F. P. Casale, V. Proserpio, A. Scialdone, F. J. Theis, S. A. Teichmann, J. C. Marioni, and O. Stegle. 2015. “Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells.” *Nat. Biotechnol.* 33 (2):155–60.

Islam, S., U. Kjallquist, A. Moliner, P. Zajac, J. B. Fan, P. Lonnerberg, and S. Linnarsson. 2011. “Characterization of the single-cell transcriptional landscape by highly multiplex RNA-seq.” *Genome Res.* 21 (7):1160–7.

Lun, A. T. L., F. J. Calero-Nieto, L. Haim-Vilmovsky, B. Gottgens, and J. C. Marioni. 2017. “Assessing the reliability of spike-in normalization for analyses of single-cell RNA sequencing data.” *Genome Res.* 27 (11):1795–1806.

Lun, A. T., K. Bach, and J. C. Marioni. 2016. “Pooling across cells to normalize single-cell RNA sequencing data with many zero counts.” *Genome Biol.* 17 (April):75.