Elaborating on cell-based quality control in single-cell RNA-seq data

***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 Overview

Low-quality cells can often yield misleading results in downstream analyses, by:

- forming their own distinct cluster(s), complicating interpretation of the results. This can be most obviously driven by increased mitochondrial proportions or enrichment for nuclear RNAs after cell damage. However, very small libraries can also form their own clusters due to shifts in the mean upon log-transformation.
- containing genes that appear to be strongly “upregulated” due to the presence of very small size factors. This is most problematic if some transcripts are present at constant

levels in the ambient solution for all cells (i.e., wells or droplets). Small counts will then be greatly inflated upon normalization with these size factors.
- containing genes that appear to be strongly "downregulated" due to the loss of RNA upon cell damage. This seems most pronounced with ribosomal protein genes, though other cytoplasmic transcripts are likely to be affected.
- distorting the characterization of population heterogeneity during variance estimation or principal components analysis. The first few principal components will capture differences in quality rather than biology, reducing the effectiveness of dimensionality reduction. Similarly, genes with the largest variances will be driven by differences between low- and high-quality cells.

As such, we need to remove these cells at the start of the analysis. Recall that we were defining low-quality cells as those with outlier values for various quality control (QC) metrics, using the `isOutlier()` and `calculateQCMetrics()` functions from the *scater (https://bioconductor.org/packages/3.8/scater)* package (McCarthy et al. 2017). Here, we will examine some of the reasoning behind the outlier-based QC in more detail.

2 Assumptions of outlier identification

An outlier-based definition for low-quality cells assumes that most cells are of high quality. This is usually reasonable and can be experimentally supported in some situations by visually checking that the cells are intact, e.g., on the microwell plate. Another assumption is that the QC metrics are independent on the biological state of each cell. This ensures that any outlier values for these metrics are driven by technical factors rather than biological processes. Thus, removing cells based on the metrics will not misrepresent the biology in downstream analyses.

The second assumption is most likely to be violated in highly heterogeneous cell populations. For example, some cell types may naturally have less RNA or express fewer genes than other cell types. Such cell types are more likely to be considered outliers and removed, even if they are of high quality. The use of the MAD mitigates this problem by accounting for biological variability in the QC metrics. A heterogeneous population should have higher variability in the metrics among high-quality cells, increasing the MAD and reducing the chance of incorrectly removing particular cell types (at the cost of reducing power to remove low-quality cells). Nonetheless, filtering based on outliers may not be appropriate in extreme cases where one cell type is very different from the others.

Systematic differences in the QC metrics can be handled to some extent using the `batch=` argument in the `isOutlier()` function. For example, setting `batch` to the plate of origin will identify outliers within each level of `batch`, using plate-specific median and MAD estimates. This is obviously useful for accommodating known differences in experimental processing, e.g., sequencing at different depth or different amounts of added spike-in RNA. We can also include biological factors in `batch`, if those factors could result in systematically fewer expressed genes or lower RNA content. However, this is not applicable in experiments where the factors are not known in advance.

3 Checking for discarded cell types

3.1 In the 416B data set

We can diagnose loss of distinct cell types during QC by looking for differences in gene expression between the discarded and retained cells. To demonstrate, we compute the average count across the discarded and retained pools in the 416B data set.

```
library(SingleCellExperiment)
sce.full.416b <- readRDS("416B_preQC.rds")

library(scater)
lost <- calcAverage(counts(sce.full.416b)[,!sce.full.416b$Pa
ssQC])
kept <- calcAverage(counts(sce.full.416b)[,sce.full.416b$Pas
sQC])
```

If the discarded pool is enriched for a certain cell type, we should observe increased expression of the corresponding marker genes. No systematic upregulation of genes is apparent in the discarded pool in Figure 1, indicating that the QC step did not inadvertently filter out a cell type in the 416B dataset.

```
# Avoid loss of points where either average is zero.
capped.lost <- pmax(lost, min(lost[lost>0]))
capped.kept <- pmax(kept, min(kept[kept>0]))

plot(capped.lost, capped.kept, xlab="Average count (discarde
d)",
    ylab="Average count (retained)", log="xy", pch=16)
is.spike <- isSpike(sce.full.416b)
points(capped.lost[is.spike], capped.kept[is.spike], col="re
d", pch=16)
is.mito <- rowData(sce.full.416b)$is_feature_control_Mt
points(capped.lost[is.mito], capped.kept[is.mito], col="dodg
erblue", pch=16)
```

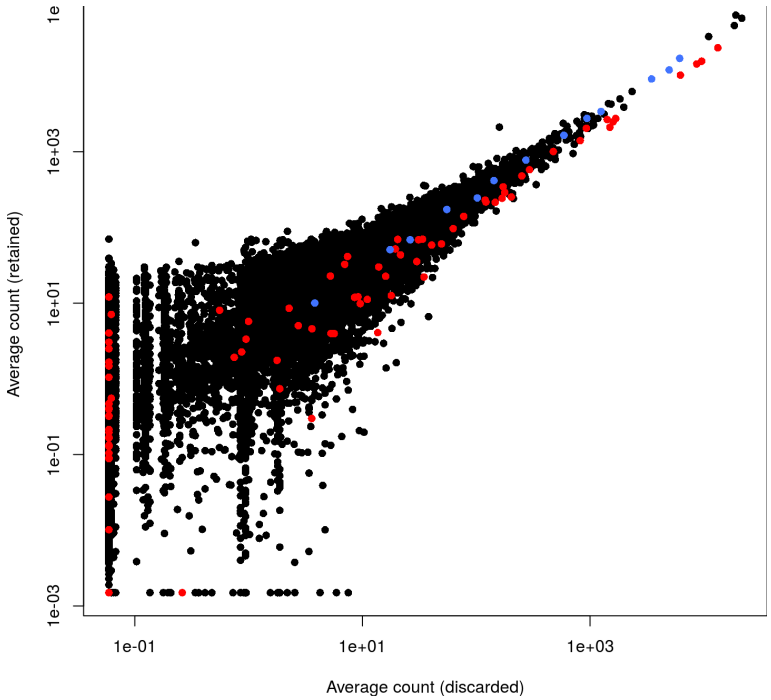

Figure 1: Average counts across all discarded and retained cells in the 416B dataset. Each point represents a gene, with spike-in and mitochondrial transcripts in red and blue respectively.

We examine this more closely by computing log-fold changes between the average counts of the two pools. The `predFC` function stabilizes the log-fold change estimates by adding a prior count to the average of each pool. We only examine the log-fold changes rather than formally testing for differential expression, as we are not interested in penalizing intra-pool heterogeneity.

```
library(edgeR)
coefs <- predFC(cbind(lost, kept), design=cbind(1, c(1,
0)))[,2]
info <- data.frame(logFC=coefs, Lost=lost, Kept=kept,
    row.names=rownames(sce.full.416b))
head(info[order(info$logFC, decreasing=TRUE),], 20)
```

```
##                       logFC      Lost        Kept
## ENSMUSG00000104647 6.844237  7.515034 0.000000000
## Nmur1              6.500909  5.909533 0.000000000
## Retn               6.250501 10.333172 0.196931018
## Fut9               6.096356  4.696897 0.010132032
## ENSMUSG00000102352 6.077614  9.393793 0.206637368
## ENSMUSG00000102379 6.029758  4.244690 0.000000000
## 1700101I11Rik      5.828821  4.483094 0.039199404
## Gm4952             5.698380  6.580862 0.172999108
## ENSMUSG00000106680 5.670156  3.389236 0.005234611
## ENSMUSG00000107955 5.554616  5.268508 0.132532601
## Gramd1c            5.446975  4.435342 0.103783669
## Jph3               5.361082  4.696897 0.139188080
## ENSMUSG00000092418 5.324462  3.395752 0.056931488
## 1700029I15Rik      5.316226  8.199510 0.394588776
## Pih1h3b            5.307439  2.546814 0.000000000
## ENSMUSG00000097176 5.275459  2.541927 0.003772842
## Olfr456            5.093383  2.186536 0.000000000
## ENSMUSG00000103731 5.017303  3.315016 0.107148909
## Klhdc8b            4.933215 19.861036 1.635081878
## ENSMUSG00000082449 4.881422  1.878759 0.000000000
```

Again, no obvious cell type markers are present in the top set of genes upregulated in the discarded pool. The magnitude of the log-fold changes is less important, attributable to imprecision with few cells in the discarded pool. Large log-fold changes can also be driven by enrichment or depletion of mitochondrial, ribosomal protein or nuclear genes upon cell damage.

3.2 In the PBMC data set

For comparison, we consider the PBMC data set in which we previously identified a platelet population (see the previous workflow (https://bioconductor.org/packages/3.8/simpleSingleCell/vignettes/work-3-tenx.html#marker-gene-detection)). Recall that we relied on the use of the `emptyDrops()` method from the *DropletUtils (https://bioconductor.org/packages/3.8/DropletUtils)* package to retain the platelets. In contrast, if we had used a naive threshold on the total unique molecular identifier (UMI) count, we would have removed this population during the cell calling step.

```
sce.pbmc <- readRDS("pbmc_data.rds")
wrong.keep <- sce.pbmc$total_counts >= 1000

lost <- calcAverage(counts(sce.pbmc)[,!wrong.keep])
kept <- calcAverage(counts(sce.pbmc)[,wrong.keep])
```

The presence of a distinct population in the discarded pool manifests in Figure 2 as a shift to the bottom-right for a number of genes. This includes *PF4*, *PPBP* and *SDPR* that are strongly upregulated in the platelets.

```
# Avoid loss of points where either average is zero.
capped.lost <- pmax(lost, min(lost[lost>0]))
capped.kept <- pmax(kept, min(kept[kept>0]))

plot(capped.lost, capped.kept, xlab="Average count (discarde
d)",
    ylab="Average count (retained)", log="xy", pch=16)
platelet <- c("PF4", "PPBP", "SDPR")
points(capped.lost[platelet], capped.kept[platelet], col="or
ange", pch=16)
```

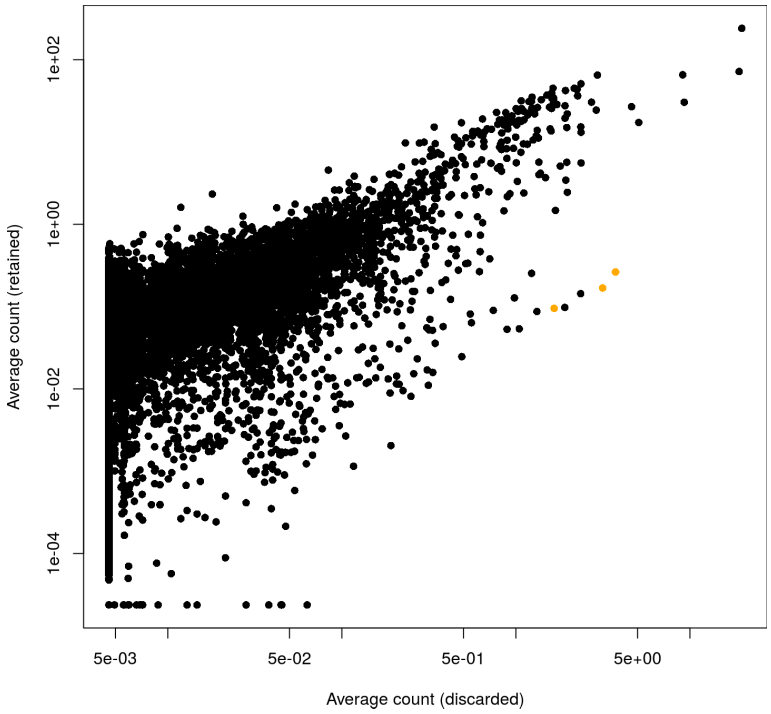

Figure 2: Average counts across all discarded and retained cells in the PBMC dataset, after using a more stringent filter on the total UMI count. Each point represents a gene, with platelet-related genes highlighted in orange.

These platelet-specific genes are also present among the top set of positive log-fold changes.

```
coefs <- predFC(cbind(lost, kept), design=cbind(1, c(1,
0)))[,2]
info <- data.frame(logFC=coefs, Lost=lost, Kept=kept,
    row.names=rownames(sce.pbmc))
head(info[order(info$logFC, decreasing=TRUE),], 20)
```

```
##              logFC      Lost       Kept
## PF4       6.260123 3.1515268 0.16839563
## PPBP      6.193032 3.7362778 0.26331806
## HIST1H2AC 5.937488 2.3556043 0.14358496
## GNG11     5.829776 1.9102028 0.09765613
## SDPR      5.639884 1.6587397 0.09518196
## TUBB1     5.352245 1.3204857 0.08722296
## CLU       5.185106 1.0455393 0.05373447
## ACRBP     4.960898 0.8893300 0.05303073
## NRGN      4.757928 0.9803744 0.12727449
## RGS18     4.638017 1.2305668 0.25363327
## MAP3K7CL  4.526863 0.7415279 0.08998410
## SPARC     4.282294 0.4888558 0.02470512
## MMD       4.250060 0.5552700 0.06362420
## PGRMC1    4.142610 0.5466903 0.08141040
## CMTM5     3.811447 0.3331150 0.01588417
## TSC22D1   3.769813 0.3821595 0.05722460
## HRAT92    3.761280 0.3140556 0.01106262
## GP9       3.739236 0.3436135 0.03580096
## ITGA2B    3.703923 0.3090122 0.01700070
## CTSA      3.635665 0.6181223 0.26569357
```

3.3 Avoiding loss of cell types

If cell types are being incorrectly discarded, the most direct solution is to relax the QC filters by increasing `nmads=` in the `isOutlier()` calls. We can also avoid filtering on metrics that are associated with genuine biological differences between cell types. The most extreme approach would be to not perform any QC filtering at all, thus guaranteeing that all cell types in the data are retained. However, this obviously comes with an increased risk of retaining more low-quality damaged cells. Such cells will cause problems in downstream analyses as discussed above, which motivates the use of a more strict filter (at least on the first pass) in our workflows.

As an aside, it is worth mentioning that the true technical quality of a cell may be correlated with its type. (This differs from a correlation between the cell type and the QC metrics, as the latter are our imperfect proxies for quality.) This can arise if some cell types are not amenable to dissociation or microfluidics handling during the scRNA-seq protocol. In such cases, it is possible to correctly discard an entire cell type during QC if all of its members are damaged. Indeed, concerns over the computational removal of cell types during QC are probably minor compared to losses in the experimental protocol.

4 Alternative approaches to quality control

4.1 Using fixed thresholds

One alternative strategy is to set pre-defined thresholds on each QC metric. For example, we might remove all cells with library sizes below 100000 and numbers of expressed genes below 4000. This avoids any assumptions associated with the use of outliers to identify low-quality cells. However, it generally requires considerable experience to determine appropriate thresholds for each experimental protocol and biological system. For example, thresholds for read count-based data are simply not applicable for UMI-based data, and vice versa. Indeed, even with the same protocol and system, the appropriate threshold can vary from run to run due to the vagaries of RNA capture and sequencing.

4.2 Using PCA-based outliers

Another strategy is to perform a principal components analysis (PCA) based on the quality metrics for each cell, e.g., the total number of reads, the total number of features and the proportion of mitochondrial or spike-in reads. Outliers on a PCA plot may be indicative of low-quality cells that have aberrant technical properties compared to the (presumed) majority of high-quality cells. This is demonstrated below on a brain cell dataset from Tasic et al. (2016), using functions from *scater (https://bioconductor.org/packages/3.8/scater)*.

```
# Obtaining the dataset.
library(scRNAseq)
data(allen)

# Setting up the data.
sce.allen <- as(allen, "SingleCellExperiment")
assayNames(sce.allen) <- "counts"
isSpike(sce.allen, "ERCC") <- grep("ERCC", rownames(sce.alle
n))

# Computing the QC metrics and running PCA.
library(scater)
sce.allen <- calculateQCMetrics(sce.allen)
sce.allen <- runPCA(sce.allen, use_coldata=TRUE, detect_outl
iers=TRUE)
table(sce.allen$outlier)
```

```
##
## FALSE  TRUE
##   374     5
```

Methods like PCA-based outlier detection and support vector machines can provide more power to distinguish low-quality cells from high-quality counterparts (Ilicic et al. 2016). This is because they are able to detect subtle patterns across many quality metrics simultaneously. However, this comes at some cost to interpretability, as the reason for removing a given cell may not

always be obvious. Users interested in the more sophisticated approaches are referred to the *scater (https://bioconductor.org/packages/3.8/scater)* and *cellity (https://bioconductor.org/packages/3.8/cellity)* packages.

4.3 Using the gene expression profiles

For completeness, we note that outliers can also be identified from the gene expression profiles, rather than QC metrics. We consider this to be a risky strategy as it can remove high-quality cells in rare populations. Even if subpopulations are explicitly captured with a mixture model, removal of outlier cells will simply reinforce the existing model. This may be misleading if it understates the biological heterogeneity in each population.

5 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

Ilicic, T., J. K. Kim, A. A. Kołodziejczyk, F. O. Bagger, D. J. McCarthy, J. C. Marioni, and S. A. Teichmann. 2016. “Classification of low quality cells from single-cell RNA-seq data.” *Genome Biol.* 17 (1):29.

McCarthy, D. J., K. R. Campbell, A. T. Lun, and Q. F. Wills. 2017. “Scater: pre-processing, quality control, normalization and visualization of single-cell RNA-seq data in R.” *Bioinformatics* 33 (8):1179–86.

Tasic, B., V. Menon, T. N. Nguyen, T. K. Kim, T. Jarsky, Z. Yao, B. Levi, et al. 2016. “Adult mouse cortical cell taxonomy revealed by single cell transcriptomics.” *Nat. Neurosci.* 19 (2):335–46.