SCnorm: robust normalization of single-cell RNA-seq data

Rhonda Bacher and Christina Kendziorski

November 23, 2019

Contents

1 Introduction

Normalization is an important first step in analyzing RNA-seq expression data to allow for accurate comparisons of a gene's expression across samples. Typically, normalization methods estimate a scale factor per sample to adjust for differences in the amount of sequencing each sample receives. This is because increases in sequencing typically lead to proportional increases in gene counts. However, in scRNA-seq data sequencing depth does not affect gene counts equally. SCnorm (as detailed in Bacher* and Chu* *et al.*, 2017) is a normalization approach we developed for single-cell RNA-seq data (scRNA-seq). SCnorm groups genes based on their count-depth relationship and within each group applies a quantile regression to estimate scaling factors to remove the effect of sequencing depth from the counts.

If you use SCnorm in published research, please cite:

Bacher R, Chu LF, Leng N, Gasch AP, Thomson J, Stewart R, Newton M, Kendziorski C. SCnorm: robust normalization of single-cell RNA-seq data. Nature Methods. 14 (6), 584-586

(http://www.nature.com/nmeth/journal/v14/n6/full/nmeth.4263.html)

If after reading through this vignette and the FAQ section you have questions or problems using SCnorm, please post them to https://support.bioconductor.org and tag "SCnorm". This will notify the package maintainers and benefit other users.

2 Run SCnorm

Before analysis can proceed, the SCnorm package must be installed.

The easiest way to install SCnorm is through Bioconductor (https://bioconductor.org/packages/SCnorm):

```
if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
BiocManager::install("SCnorm")
```

There are two additional ways to download SCnorm.

The first option is most useful if using a previous version of SCnorm (which can be found at https://github.com/rhondabacher/SCnorm/releases).

The second option is to download the most recent development version of SCnorm. This version may be used with versions of R below 3.4. This version includes use of summarizedExperiment, but parallel computation is maintained using the parallel package.

```
install.packages("SCnorm_x.x.x.tar.gz", repos=NULL, type="source")
#OR
library(devtools)
devtools::install_github("rhondabacher/SCnorm", ref="devel")
```

After successful installation, the package must be loaded into the working space:

```
library(SCnorm)
```

2.1 Required inputs

`Data`: Input to SCnorm may be either of class SummarizedExperiment or a matrix. The expression matrix should be a $G - by - S$ matrix containing the expression values for each gene and each cell, where G is the number of genes and S is the number of cells/samples. Gene names should be assigned as rownames and not a column in the matrix. Estimates of gene expression are typically obtained using RSEM, HTSeq, Cufflinks, Salmon or similar approaches.

The object `ExampleSimSCData` is a simulated single-cell data matrix containing 5,000 rows of genes and 90 columns of cells.

```
data(ExampleSimSCData)
ExampleSimSCData[1:5,1:5]
```

```
##           Cell_1     Cell_2    Cell_3    Cell_4    Cell_5
## Gene_1  3.809447  0.5891684  6.505612 33.052416  0.000000
## Gene_2 21.703998  3.9118515  0.000000  0.000000  5.651665
## Gene_3  2.128070  0.0000000 11.587132 11.352172  5.402953
## Gene_4 22.681505  0.0000000 27.495737  2.571482  8.248906
## Gene_5  0.000000 23.9983434 38.810055  1.244605 12.693561
```

```
str(ExampleSimSCData)
```

```
##  num [1:5000, 1:90] 3.81 21.7 2.13 22.68 0 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:5000] "Gene_1" "Gene_2" "Gene_3" "Gene_4" ...
##   ..$ : chr [1:90] "Cell_1" "Cell_2" "Cell_3" "Cell_4" ...
```

SCnorm now implements the*SingleCellExperiment* class. The input data should be formatted as following, where ExamplesSimSCData is the expression matrix:

```
ExampleSimSCData <- SingleCellExperiment::SingleCellExperiment(assays = list('counts' = ExampleSimSCData))
```

If the data was originally formatted as a *SummarizedExperiment*, then it can be coerced using:

```
ExampleSimSCData <- as(ExampleSimSCData, "SingleCellExperiment")
```

Conditions: The object `Conditions` should be a vector of length S indicating which condition/group each cell belongs to. The order of this vector MUST match the order of the columns in the `Data` matrix.

```
Conditions = rep(c(1), each= 90)
head(Conditions)
```

```
## [1] 1 1 1 1 1 1
```

2.2 SCnorm: Check count-depth relationship

Before normalizing using SCnorm it is advised to check the relationship between expression counts and sequencing depth (referred to as the count-depth relationship) for your data. If all genes have a similar relationship then a global normalization strategy such as median-by-ratio in the DESeq package or TMM in edgeR will also be adequate. However, we find that when the count-depth relationship varies among genes using global scaling strategies leads to poor normalization. In these cases we strongly recommend proceeding with the normalization provided by SCnorm.

The `plotCountDepth` function will estimate the count-depth relationship for all genes. The genes are first divided into ten equally sized groups based on their non-zero median expression. The function will generate a plot of the distribution of slopes for each group. We also recommend checking a variety of filter options in case you find that only genes expressed in very few cells or very low expressors are the main concern.

The function `plotCountDepth` produces a plot as output which may be wrapped around a call to pdf() or png() to save it for future reference. The function also returns the information used for plotting which may be ultilized if a user would like to generate a more customized

figure. The returned object is a list with length equal to the number of conditions and each element of the list is a mtrix containing each genes's assigned group (based on non-zero median expression) and its' estimated count-depth relationship (Slope).

```
pdf("check_exampleData_count-depth_evaluation.pdf", height=5, width=7)
countDeptEst <- plotCountDepth(Data = ExampleSimSCData, Conditions = Conditions,
                FilterCellProportion = .1, NCores=3)
```

```
## Warning in data.table::melt(groupGenes, value.name = "Group"):  The melt generic
in data.table has been passed a integer and will attempt to redirect to the relevant
reshape2 method; please note that reshape2 is deprecated, and this redirection is
now deprecated as well.  To continue using melt methods from reshape2 while both
libraries are attached, e.g.  melt.list, you can prepend the namespace like reshape2::melt(groupGenes).
In the next version, this warning will become an error.
```

```
dev.off()
```

```
## pdf
##   2
```

```
str(countDeptEst)
```

```
## List of 1
##  $ 1:'data.frame': 5000 obs. of  3 variables:
##   ..$ Gene : chr [1:5000] "Gene_1" "Gene_10" "Gene_100" "Gene_1000" ...
##   ..$ Group: int [1:5000] 4 1 3 3 3 4 1 1 2 2 ...
##   ..$ Slope: num [1:5000] 0.07683 -0.00465 0.038 -0.01725 -0.04196 ...
```

```
head(countDeptEst[[1]])
```

```
##        Gene Group        Slope
## 1    Gene_1     4  0.076832005
## 2   Gene_10     1 -0.004653532
## 3  Gene_100     3  0.038000496
## 4 Gene_1000     3 -0.017253843
## 5 Gene_1001     3 -0.041956519
## 6 Gene_1002     4 -0.673548775
```

Since gene expression increases proportionally with sequencing depth, we expect to find the estimated count-depth relationships near 1 for all genes. This is typically true for bulk RNA-seq datasets. However, it does not hold in most single-cell RNA-seq datasets. In this example data, the relationship is quite variable across genes.

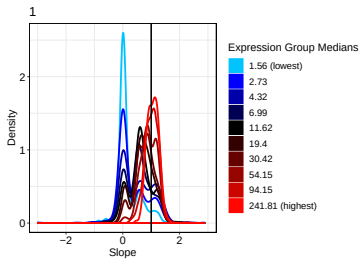


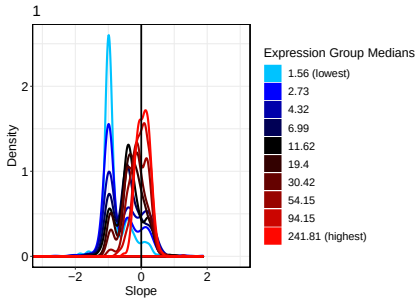
Figure 1: Evaluation of count-depth relationship in un-normalized data

After normalization the count-depth relationship can be estimated on the normalized data, where slopes near zero indicate successful normalization. We can use the function `plotCountDepth` to evaluate normalized data:

```
ExampleSimSCData = SingleCellExperiment::counts(ExampleSimSCData)
# Total Count normalization, Counts Per Million, CPM.
ExampleSimSCData.CPM <- t((t(ExampleSimSCData) / colSums(ExampleSimSCData)) *
                        mean(colSums(ExampleSimSCData)))


countDeptEst.CPM <- plotCountDepth(Data = ExampleSimSCData,
                                   NormalizedData = ExampleSimSCData.CPM,
                                    Conditions = Conditions,
                                    FilterCellProportion = .1, NCores=3)
```

```
## Warning in data.table::melt(groupGenes, value.name = "Group"):  The melt generic
in data.table has been passed a integer and will attempt to redirect to the relevant
reshape2 method; please note that reshape2 is deprecated, and this redirection is
now deprecated as well.  To continue using melt methods from reshape2 while both
libraries are attached, e.g.  melt.list, you can prepend the namespace like reshape2::melt(groupGenes).
In the next version, this warning will become an error.
```




```
str(countDeptEst.CPM)
```

```
## List of 1
##  $ 1:'data.frame': 5000 obs. of  3 variables:
##   ..$ Gene : chr [1:5000] "Gene_1" "Gene_10" "Gene_100" "Gene_1000" ...
##   ..$ Group: int [1:5000] 4 1 3 3 3 4 1 1 2 2 ...
##   ..$ Slope: num [1:5000] -0.923 -1.005 -0.962 -1.017 -1.042 ...
```

```
head(countDeptEst.CPM[[1]])
```

```
##        Gene Group      Slope
## 1    Gene_1     4 -0.9231680
## 2   Gene_10     1 -1.0046535
## 3  Gene_100     3 -0.9619995
## 4 Gene_1000     3 -1.0172538
## 5 Gene_1001     3 -1.0419565
## 6 Gene_1002     4 -1.6735488
```

If normalization is succesful the estimated count-depth relationship should be near zero for all genes. Here the highly expressed genes appear well normalized, while moderate and low expressors have been over-normalized and have negative slopes.

2.3 SCnorm: Normalization

SCnorm will normalize across cells to remove the effect of sequencing depth on the counts. The function `SCnorm` returns: the normalized expression counts, a list of genes which were not considered in the normalization due to filter options, and optionally a matrix of scale factors (if `reportSF = TRUE`).

The default filter for SCnorm only considers genes having at least 10 non-zero expression values. The user may wish to adjust the filter and may do so by changing the value of FilterCellNum. The user may also wish to not normalize very low expressed genes. The parameter `FilterExpression` can be used to exclude genes which have non-zero medians lower than a threshold specified by `FilterExpression` (`FilterExpression`=0 is default).

SCnorm starts at $K = 1$, which normalizes the data assuming all genes should be normalized in one group. The sufficiency of $K = 1$ is evaluated by estimating the normalized count-depth relationships. This is done by splitting genes into 10 groups based on their non-zero unnormalized median expression (equally sized groups) and estimating the mode for each group. If all 10 modes are within .1 of zero, then $K = 1$ is sufficient. If any mode is large than .1 or less than -.1, SCnorm will attempt to normalize assuming $K = 2$ and repeat the group normalizations and evaluation. This continues until all modes are within .1 of zero. (Simulation studies have shown .1 generally works well and is default, but may be adjusted using the parameter `Thresh` in the `SCnorm` function). If `PrintProgressPlots = TRUE` is specified, the progress plots of SCnorm will be created for each value of K tried (default is `FALSE`).

Normalized data can be accessed using the *results()* function.

```
Conditions = rep(c(1), each= 90)
pdf("MyNormalizedData_k_evaluation.pdf")
par(mfrow=c(2,2))
DataNorm <- SCnorm(Data = ExampleSimSCData,
                   Conditions = Conditions,
                   PrintProgressPlots = TRUE,
                   FilterCellNum = 10, K =1,
                   NCores=3, reportSF = TRUE)

## Setting up parallel computation using 3 cores

## SCnorm will normalize assuming 1 is the optimal number of groups.  It is not
advised to set this.

## Gene filter is applied within each condition.

## 0 genes in condition 1 will not be included in the normalization due to
##              the specified filter criteria.

## A list of these genes can be accessed in output,
##    see vignette for example.

## Done!

dev.off()

## pdf
##   2

DataNorm
```

```
## class: SingleCellExperiment
## dim: 5000 90
## metadata(2): ScaleFactors GenesFilteredOut
## assays(2): counts normcounts
## rownames(5000): Gene_1 Gene_2 ... Gene_4999 Gene_5000
## rowData names(0):
## colnames(90): Cell_1 Cell_2 ... Cell_89 Cell_90
## colData names(0):
## reducedDimNames(0):
## spikeNames(0):
## altExpNames(0):
```

```
NormalizedData <- SingleCellExperiment::normcounts(DataNorm)
NormalizedData[1:5,1:5]
```

```
##           Cell_1     Cell_2    Cell_3     Cell_4    Cell_5
## Gene_1  4.927704  0.4627998  5.239220 26.2840393  0.000000
## Gene_2 28.075171  3.0728125  0.000000  0.0000000  5.782488
## Gene_3  2.752762  0.0000000  9.331563  9.0275072  5.528019
## Gene_4 29.339624  0.0000000 22.143373  2.0449014  8.439849
## Gene_5  0.000000 18.8510249 31.255229  0.9897386 12.987387
```

The list of genes not normalized according to the filter specifications may be obtained by:

```
GenesNotNormalized <- results(DataNorm, type="GenesFilteredOut")
str(GenesNotNormalized)
```

```
## List of 1
##  $ GenesFilteredOutGroup1: chr(0)
```

If scale factors should be returned, then they can be accessed using:

```
ScaleFactors <- results(DataNorm, type="ScaleFactors")
str(ScaleFactors)
```

```
##  num [1:5000, 1:90] 0.773 0.773 0.773 0.773 0.773 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:5000] "Gene_1" "Gene_2" "Gene_3" "Gene_4" ...
##   ..$ : chr [1:90] "Cell_1" "Cell_2" "Cell_3" "Cell_4" ...
```

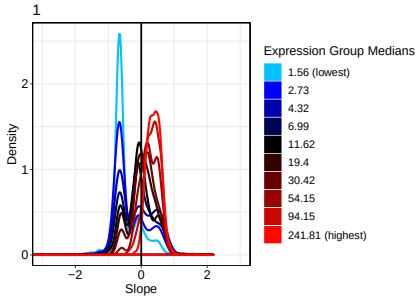
2.4 Evaluate choice of *K*

SCnorm first fits the model for $K = 1$, and sequentially increases K until a satisfactory stopping point is reached. In the example run above, $K = 4$ is chosen, once all 10 slope densities have an absolute slope mode $< .1$.

We can also use the function `plotCountDepth` to make the final plot of the normalized count-depth relationship:

```
countDeptEst.SCNORM <- plotCountDepth(Data = ExampleSimSCData,
                                      NormalizedData = NormalizedData,
                                      Conditions = Conditions,
                                      FilterCellProportion = .1, NCores=3)
```

```
## Warning in data.table::melt(groupGenes, value.name = "Group"):  The melt generic
in data.table has been passed a integer and will attempt to redirect to the relevant
reshape2 method; please note that reshape2 is deprecated, and this redirection is
now deprecated as well.  To continue using melt methods from reshape2 while both
libraries are attached, e.g.  melt.list, you can prepend the namespace like reshape2::melt(groupGenes).
In the next version, this warning will become an error.
```




```
str(countDeptEst.SCNORM)

## List of 1
##  $ 1:'data.frame': 5000 obs. of  3 variables:
##   ..$ Gene : chr [1:5000] "Gene_1" "Gene_10" "Gene_100" "Gene_1000" ...
##   ..$ Group: int [1:5000] 4 1 3 3 3 4 1 1 2 2 ...
##   ..$ Slope: num [1:5000] -0.593 -0.675 -0.597 -0.692 -0.684 ...

head(countDeptEst.SCNORM[[1]])

##        Gene Group      Slope
## 1    Gene_1     4 -0.5927619
## 2   Gene_10     1 -0.6748594
## 3  Gene_100     3 -0.5970708
## 4 Gene_1000     3 -0.6915538
## 5 Gene_1001     3 -0.6835126
## 6 Gene_1002     4 -1.3863016
```

3 SCnorm: Multiple Conditions

When more than one condition is present SCnorm will first normalize each condition independently then apply a scaling procedure between the conditions. In this step the assumption is that most genes are not differentially expressed (DE) between conditions and that any systematic differences in expression across the majority of genes is due to technical biases and should be removed.

Zeros are not included in the estimations of scaling factors across conditions by default, this will make the means across conditions equal when zeros are not considered (which can then used for downstream differential expression analyses with *MAST*, for example). However, if the proportion of zeros is very unequal between conditions and the user plans to use a downstream tool which includes zeros in the model (such as *DESeq* or *edgeR*), then the scaling should be estimated with zeros included in this calculation by using the `useZerosToScale=TRUE` option:

```
Conditions = rep(c(1, 2), each= 90)
DataNorm <- SCnorm(Data = MultiCondData,
                   Conditions = Conditions,
                   PrintProgressPlots = TRUE
                   FilterCellNum = 10,
                   NCores=3,
                   useZerosToScale=TRUE)
```

Generally the definition of condition will be obvious given the experimental setup. If the data are very heterogenous within an experimental setup it may be beneficial to first cluster more similar cells into groups and define these as conditions in SCnorm.

For very large datasets, it might be more reasonable to normalize each condition separately. To do this, first normalize each Condition separate:

```
DataNorm1 <- SCnorm(Data = ExampleSimSCData[,1:45],
                    Conditions = rep("1", 45),
                    PrintProgressPlots = TRUE,
                    FilterCellNum = 10,
                    NCores=3, reportSF = TRUE)


DataNorm2 <- SCnorm(Data = ExampleSimSCData[,46:90],
                   Conditions = rep("2", 45),
                   PrintProgressPlots = TRUE,
                   FilterCellNum = 10,
                   NCores=3, reportSF = TRUE)
```

Then the final scaling step can be performed as follows:

```
normalizedDataSet1 <- results(DataNorm1, type = "NormalizedData")
normalizedDataSet2 <- results(DataNorm2, type = "NormalizedData")

NormList <- list(list(NormData = normalizedDataSet1),
                     list(NormData = normalizedDataSet2))
OrigData <- ExampleSimSCData
Genes <- rownames(ExampleSimSCData)
useSpikes = FALSE
useZerosToScale = FALSE

DataNorm <- scaleNormMultCont(NormData = NormList,
                   OrigData = OrigData,
                   Genes = Genes, useSpikes = useSpikes,
                   useZerosToScale = useZerosToScale)


str(DataNorm$ScaledData)
myNormalizedData <- DataNorm$ScaledData
```

4 SCnorm: UMI data

If the single-cell expression matrix is very sparse (having a lot of zeros) then SCnorm may not be appropriate. Currently, SCnorm will issue a warning if at least one cell/sample has less than 100 non-zero values or total counts below 10,000. SCnorm will fail if the filtering criteria or quality of data leaves less then 100 genes for normalization.

If the data have -many- tied count values consider setting the option `ditherCounts=TRUE` (default is `FALSE`) which will help in estimating the count-depth relationships. This introduces some randomness but results will not change if the command is rerun.

It is highly recommended to check the count-depth relationship before and after normalization. In some cases, it might be desired to adjust the threshold used to decide K. Lowering the threshold may improve results for some datasets (default `Thresh = .1`).

For larger datasets, it may also be desired to increase the speed. One way to do this is to change the parameter `PropToUse`. `PropToUse` controls the proportion of genes nearest to the the overall group mode to use for the group fitting(default is `.25`).

```
checkCountDepth(Data = umiData, Conditions = Conditions,
                FilterCellProportion = .1, FilterExpression = 2)

DataNorm <- SCnorm(Data = umiData, Conditions= Conditions,
                   PrintProgressPlots = TRUE,
                   FilterCellNum = 10,
                   PropToUse = .1,
                   Thresh = .1,
                   ditherCounts = TRUE)
```

5 Spike-ins

SCnorm does not require spike-ins, however if high quality spike-ins are available then they may be use to perform the between condition scaling step. If `useSpikes=TRUE` then only the spike-ins will be used to estimate the scaling factors. If the spike-ins do not span the full range of expression, SCnorm will issue a warning and will use all genes to scale. SCnorm also assumes the spikes-ins are named with the prefix "ERCC-".

```
DataNorm <- SCnorm(Data = MultiCondData, Conditions = Conditions,
                   PrintProgressPlots = TRUE,
                   FilterCellNum = 10, useSpikes=TRUE)
```

6 Within-sample normalization

SCnorm allows correction of gene-specific features prior to the between-sample normalization. We implement the regression based procedure from

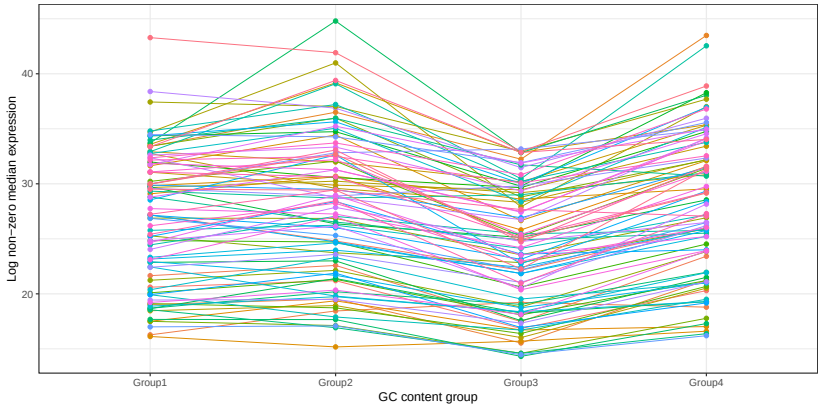
Risso et al., 2011 in *EDASeq*. To use this feature you must set withinSample equal to a vector of gene-specific features, one per gene. This could be anything, but is often GC-content or gene length.

A function to evaluate the extent of bias in expression related to the gene-specific feature is `plotWithinFactor()`. Genes are split into 4 (NumExpressionGroups = 4 is default) equally sized groups based on the gene-specific factor provided. For each cell the median expression of genes in each group isestimated and plot. The function `plotWithinFactor` returns the information used for plotting as a matrix of the expression medians for each group for all cells.

In this example I generate a random vector of gene specific features, this may be the proportion of GC content for example. Each cell receives a different color

```
#Colors each sample:
exampleGC <- runif(dim(ExampleSimSCData)[1], 0, 1)
names(exampleGC) <- rownames(ExampleSimSCData)
withinFactorMatrix <- plotWithinFactor(ExampleSimSCData, withinSample = exampleGC)
```

```
## Warning in data.table::melt(withinCells, id = c("Sample", "Condition")):  The
melt generic in data.table has been passed a data.frame and will attempt to redirect
to the relevant reshape2 method; please note that reshape2 is deprecated, and this
redirection is now deprecated as well.  To continue using melt methods from reshape2
while both libraries are attached, e.g.  melt.list, you can prepend the namespace
like reshape2::melt(withinCells).  In the next version, this warning will become
an error.
```



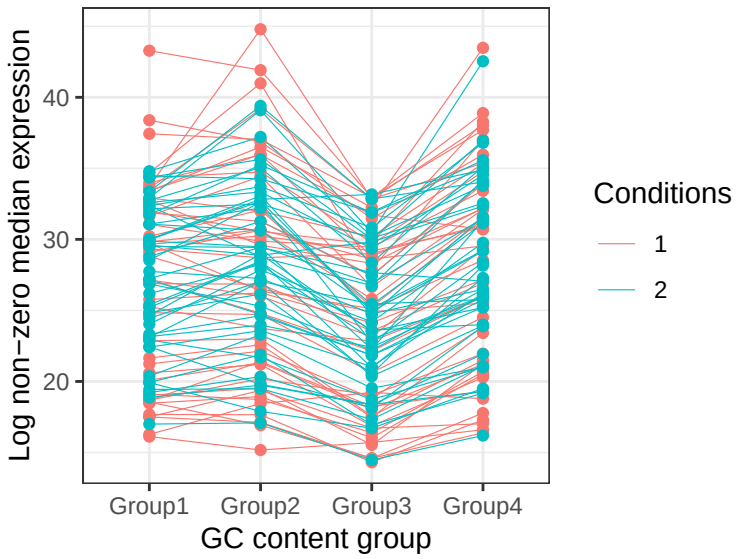

```
head(withinFactorMatrix)
```

```
##        Sample Condition   Group1   Group2   Group3   Group4
## Cell_1 Cell_1         1 20.60408 21.19700 16.85555 20.28997
## Cell_2 Cell_2         1 33.58748 39.17091 32.84120 35.37135
## Cell_3 Cell_3         1 33.42210 35.96874 28.80349 32.16495
## Cell_4 Cell_4         1 33.71427 44.79318 32.83504 38.04689
## Cell_5 Cell_5         1 25.76602 26.24171 24.13147 27.17007
## Cell_6 Cell_6         1 27.14707 26.00680 23.13783 29.35198
```

Specifying a Conditions vector will color each cell according to its associated condition:

```
#Colors samples by Condition:
Conditions <- rep(c(1,2), each=45)
withinFactorMatrix <- plotWithinFactor(ExampleSimSCData, withinSample = exampleGC,
                                       Conditions=Conditions)
```

```
## Warning in data.table::melt(withinCells, id = c("Sample", "Condition")): The
melt generic in data.table has been passed a data.frame and will attempt to redirect
to the relevant reshape2 method; please note that reshape2 is deprecated, and this
redirection is now deprecated as well. To continue using melt methods from reshape2
while both libraries are attached, e.g. melt.list, you can prepend the namespace
like reshape2::melt(withinCells). In the next version, this warning will become
an error.
```




```
head(withinFactorMatrix)
```

```
##        Sample Condition   Group1   Group2   Group3   Group4
## Cell_1 Cell_1         1 20.60408 21.19700 16.85555 20.28997
## Cell_2 Cell_2         1 33.58748 39.17091 32.84120 35.37135
## Cell_3 Cell_3         1 33.42210 35.96874 28.80349 32.16495
## Cell_4 Cell_4         1 33.71427 44.79318 32.83504 38.04689
## Cell_5 Cell_5         1 25.76602 26.24171 24.13147 27.17007
## Cell_6 Cell_6         1 27.14707 26.00680 23.13783 29.35198
```

```
# To run correction use:
DataNorm <- SCnorm(ExampleSimSCData, Conditions,
                   PrintProgressPlots = TRUE,
                   FilterCellNum = 10, withinSample = exampleGC)
```

For additional evaluation on whether to correct for these features or other options for correction, see: Risso, D., Schwartz, K., Sherlock, G. & Dudoit, S. GC-content normalization for RNA-Seq data. BMC Bioinformatics 12, 480 (2011).

7 Session info

Here is the output of sessionInfo on the system on which this document was compiled:

```
  print(sessionInfo())

## R version 3.6.1 (2019-07-05)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows Server 2012 R2 x64 (build 9600)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=C
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] SCnorm_1.8.2
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.3                  lattice_0.20-38
##  [3] snow_0.4-3                  assertthat_0.2.1
##  [5] digest_0.6.22               SingleCellExperiment_1.8.0
##  [7] R6_2.4.1                    GenomeInfoDb_1.22.0
##  [9] plyr_1.8.4                  MatrixModels_0.4-1
## [11] stats4_3.6.1                evaluate_0.14
## [13] ggplot2_3.2.1               highr_0.8
## [15] pillar_1.4.2                zlibbioc_1.32.0
## [17] rlang_0.4.1                 lazyeval_0.2.2
## [19] data.table_1.12.6           SparseM_1.77
## [21] S4Vectors_0.24.0            Matrix_1.2-17
## [23] rmarkdown_1.17              labeling_0.3
## [25] moments_0.14                BiocParallel_1.20.0
## [27] stringr_1.4.0               RCurl_1.95-4.12
## [29] munsell_0.5.0               DelayedArray_0.12.0
## [31] compiler_3.6.1              xfun_0.11
## [33] pkgconfig_2.0.3             BiocGenerics_0.32.0
## [35] htmltools_0.4.0             tidyselect_0.2.5
## [37] SummarizedExperiment_1.16.0 tibble_2.1.3
## [39] GenomeInfoDbData_1.2.2      IRanges_2.20.1
## [41] matrixStats_0.55.0          crayon_1.3.4
## [43] dplyr_0.8.3                 bitops_1.0-6
## [45] grid_3.6.1                  gtable_0.3.0
## [47] lifecycle_0.1.0             magrittr_1.5
## [49] scales_1.1.0                stringi_1.4.3
## [51] farver_2.0.1                XVector_0.26.0
## [53] reshape2_1.4.3              BiocStyle_2.14.0
## [55] forcats_0.4.0               tools_3.6.1
## [57] Biobase_2.46.0              glue_1.3.1
```

```
## [59] purrr_0.3.3                parallel_3.6.1
## [61] yaml_2.2.0                 colorspace_1.4-1
## [63] cluster_2.1.0              BiocManager_1.30.10
## [65] GenomicRanges_1.38.0       knitr_1.26
## [67] quantreg_5.52
```

8 Frequently Asked Questions

Does SCnorm work for all datasets?

No, while SCnorm has proven useful in normalizing many single-cell RNA-seq datasets, it will not be appropriate in every setting. It is important to evaluate normalized data for selected genes as well as overall (e.g. using plotCountDepth).

Can I use SCnorm on your 10X (or very sparse) dataset?

SCnorm is not intended for datasets with more than ~80% zero counts, often K will not converge in these situations. Setting the FilterExpression parameter to 1 or 2 may help, but is not a guarantee. It may also be helpful to use the ditherCounts = TRUE parameter for sparse UMI based data which may contain numerous tied counts (counts of 1 and 2 for example).

Can/Should I use SCnorm on my bulk RNA-seq data?

SCnorm can normalize bulk data. However, SCnorm should not be used to normalize data containing less then 10 cells/samples. Consideration must also be given to what downstream analysis methods will be used. Methods such as DESeq2 or edgeR typically expect a matrix of scaling factors to be supplied, these can to be obtained in SCnorm by setting `reportSF=TRUE` in the SCnorm function.

The count-depth relationship of both unnormalized and normalized bulk RNA-seq data can be evaluated using the `plotCountDepth` function.

How can I speedup SCnorm if I have thousands of cells?

Utilizing multiple cores via the `NCores` parameter is the most effective way.

Splitting the data into smaller clusters may also improve speed. Here is one example on how to cluster and the appropriate way to run `SCnorm`: (This method of clustering single-cells was originally suggested by Lun et al., 2016 in *scran*.)

```
library(dynamicTreeCut)
distM <- as.dist( 1 - cor(BigData, method = 'spearman'))
htree <- hclust(distM, method='ward.D')
clusters <- factor(unname(cutreeDynamic(htree, minClusterSize = 50,
                    method="tree", respectSmallClusters = FALSE)))
names(clusters) <- colnames(BigData)
Conditions = clusters

DataNorm <- SCnorm(Data = BigData,
                   Conditions = Conditions,
                   PrintProgressPlots = TRUE
                   FilterCellNum = 10,
```

```
                    NCores=3, useZerosToScale=TRUE)
```

When should I set the parameter `useZerosToScale=TRUE` ?

This parameter is only used when specifying multiple conditions in SCnorm. The `useZeros ToScale` parameter will determine whether zeros are used in estimating the condition based scaling factors.

Use the default, `useZerosToScale=FALSE`, when planning to use single-cell specific methods such as MAST that separately model continuous and discrete measurements.

If you plan on using an analysis method that explicitly uses zeros in the model like DESeq2, then `useZerosToScale=TRUE` should be used.

SCnorm was unable to converge.

SCnorm will fail if K exceeds 25 groups. It is unlikely that such a large number of groups is appropriate. This might happen if a large proportion of your genes have estimated count-depth relationships very close to zero. Typically this error can be resolved by removing genes with very small counts from the normalization using the parameter `FilterExpression`. Genes having non-zero median expression less than `FilterExpression` will not be scaled during normalization. An appropriate value for `FilterExpression` will be data dependent and decided based on exploring the data. One way that might help decide is to see if there are any natural cutoffs in the non-zero medians:

```
MedExpr <- apply(Data, 1, function(c) median(c[c != 0]))
plot(density(log(MedExpr), na.rm=T))
abline(v=log(c(1,2,3,4,5)))
# might set FilterExpression equal to one of these values.
```