Cellbench analysis: pure mixture of three cell lines

Kim-Anh Lê Cao and Luyi Tian

September 11, 2018

Contents

1 Important notes

- need to check the wording with Luyi / Matt regarding the identity of cell types (see in bold in Data section)
- KA to change the colors of the cell lines in expt design
- consider removing the code chunk HGV from seurat
- no perplexity parameter in Seurat t-SNE?
- check but there is no order sur a RunMultiCCA? (there is one for a runCCA which is only for 2 data sets with one set as reference)
- remove unsupervised in scmerge after testing on RNA mix
- kBET on data that are batch removed only? (waiting on Fabian Theis' answer). Clean up kBET code. More details about kBET interpretation. I find the results surprising for kBET. We may have to leave it out. If I understand correctly the high rejection rate means batch effect??
- silhouette custom function has no warning, may need the code to be cleaned up a bit?
- footnote signature and logo?

2 Packages

Install the relevant bioconductor packages to extract the data

Load the relevant libraries

3 Data

3.1 Experimental design

The single cells were sorted and sequenced using different isolation protocols.

The benchmark data includes different layers of technical - 4 sequencing protocol, and biological variability - three cell lines. Briefly, cells from five Human cell lines H2228, H1975, HCC827 from lung tissue (Adenocarcinoma; Non-Small Cell Lung Cancer) are barcoded and pooled in equal amounts. Four different types of 3' end sequencing protocols that span the range of isolation strategies available and the price range: expensive commercial droplet-based capture with Chromium 10X (10X Genomics) and Drop-seq (Dolomite), and cheaper home-brew methods using plate based isolation of cells in microwells with CEL-seq2 (see Figure).

Information of the cell types is as follows. The assignment to each cell type category is performed computationally based on the correlation of the data with bulk RNA from RNA mixOlogy (**I dont understand really what Luyi did regarding this, need to clarify with him later**).

3.2 Load data

Check the dimension of the data

```
## [1] 16468   902
```

```
## [1] 28204   274
```

```
## [1] 15127   225
```

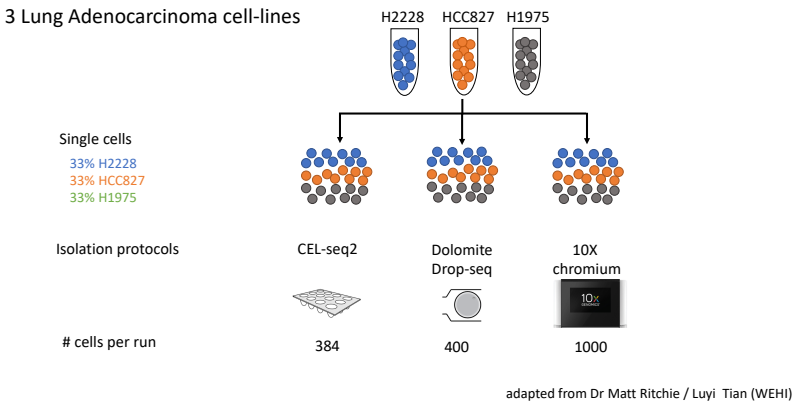


Figure 1: Benchmark experiment design: either a 'pure' cell mixture or a mixture of different amounts of cells, here we focus on the *pure cell mixture* of three cell types

Table 1: Number of cells per cell line type and per protocol

	H1975	H2228	HCC827
Chromium10X	313	315	274
CELseq2	114	81	79
DROPseq	92	65	68

Break down of the number of cells per cell type per protocol:

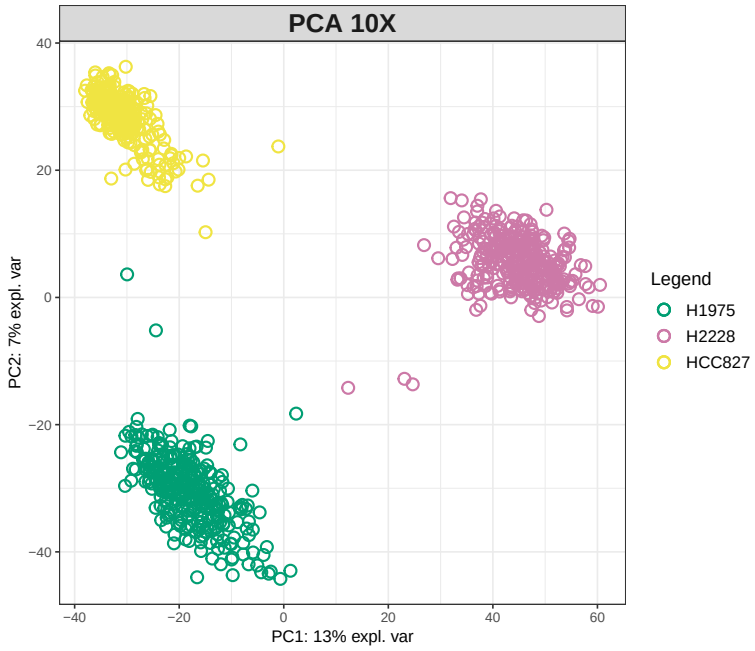
4 Normalisation

5 PCA

5.1 Each protocol independently

We first run a PCA on each data set individually on the log normalised counts:

Colors indicate the cell line type. PCA is unsupervised but we can assign to each cell a group and color.


PCA 10X
PC2: 7% expl. var
PC1: 13% expl. var
Legend
H1975
H2228
HCC827

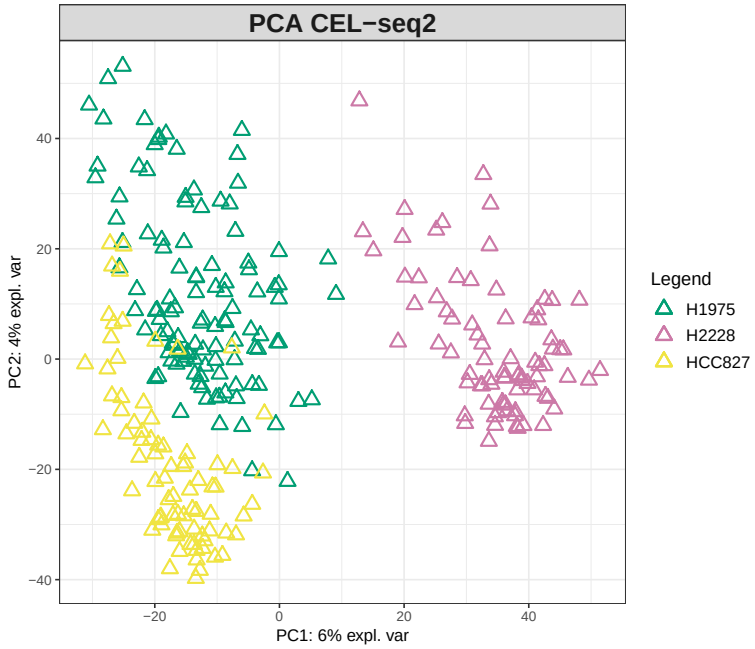
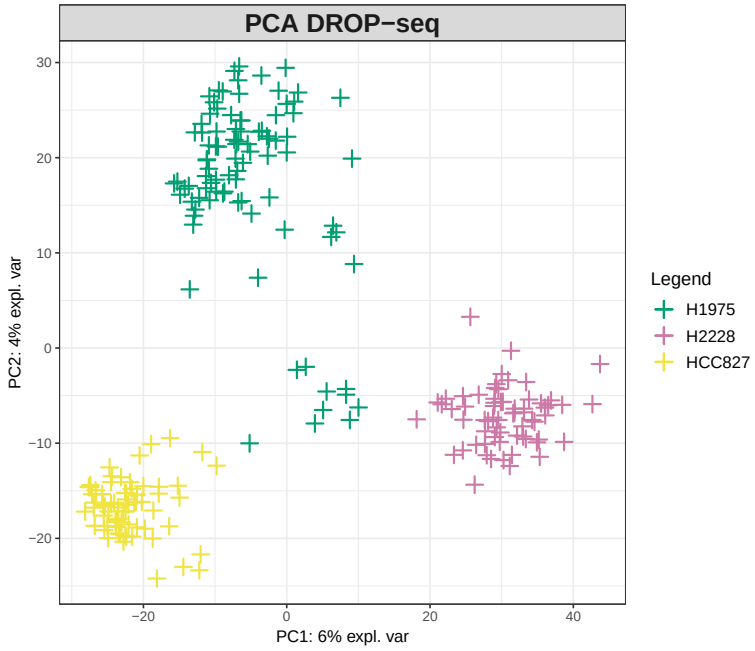

PCA CEL−seq2
PC2: 4% expl. var
PC1: 6% expl. var
Legend
H1975
H2228
HCC827

Table 2: Number of cells per protocol

	x
10X	902
CEL-seq2	274
DROP-seq	225



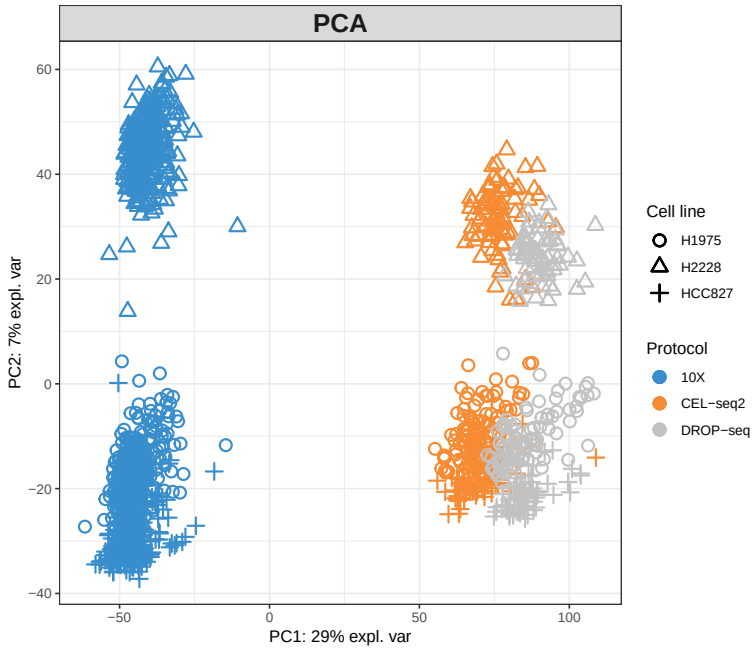
5.2 Data naively combined

We combine the data naively to identify the major sources of variation and visually assess if there is a strong protocol batch effect. See later section for a quantitative assessment.

We first need to extract the common UMI across platform.

We combine the three sets of data naively, and extract the cell line type information and the batch (protocol) information:

PCA on the combined data, using mixOmics. Here the color indicates the protocol:



6 Data integration methods on most variable genes

6.1 Selection of most variable genes

There are different ways to select the most variable genes, we list 3 options here.

6.1.1 FindVariableGenes (Seurat)

First, we need to normalise the data using Seurat. The function calculates a z-score per gene for which the average expression value is divided into 20 bins (by default). See *??FindVariableGenes* for more details.

There are only a small number of variable genes according to this criterion. This is not many, we carry on with other criteria.

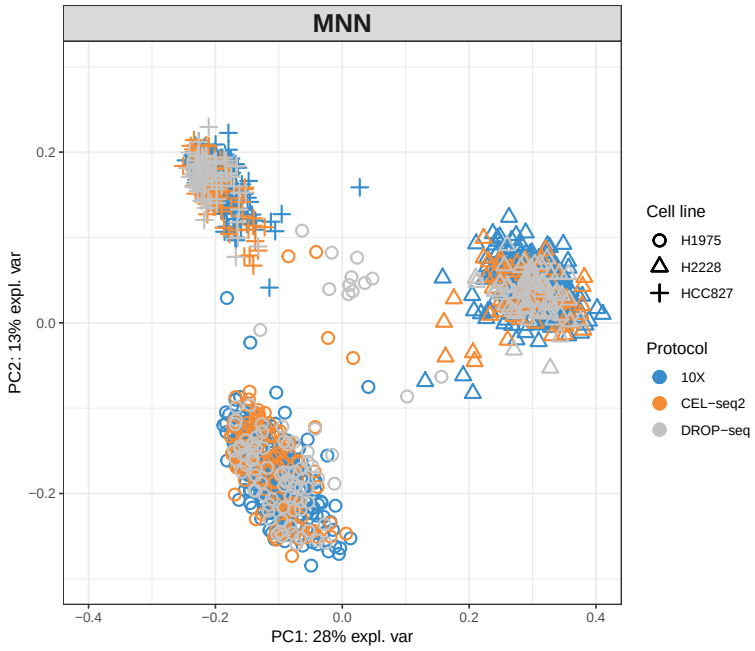
6.1.2 Custom function using decomposeVar (scran)

The decomposeVar function calculates the gene-specific biological and technical variance for an sce object, we then order the top variable genes with a high variance.

Here we obtain a sufficient number of high variable genes (861 genes). We carry on with this list. Note that is it still rather small.

7 MNN correct on HVG

Starting from the highly variable genes identified above



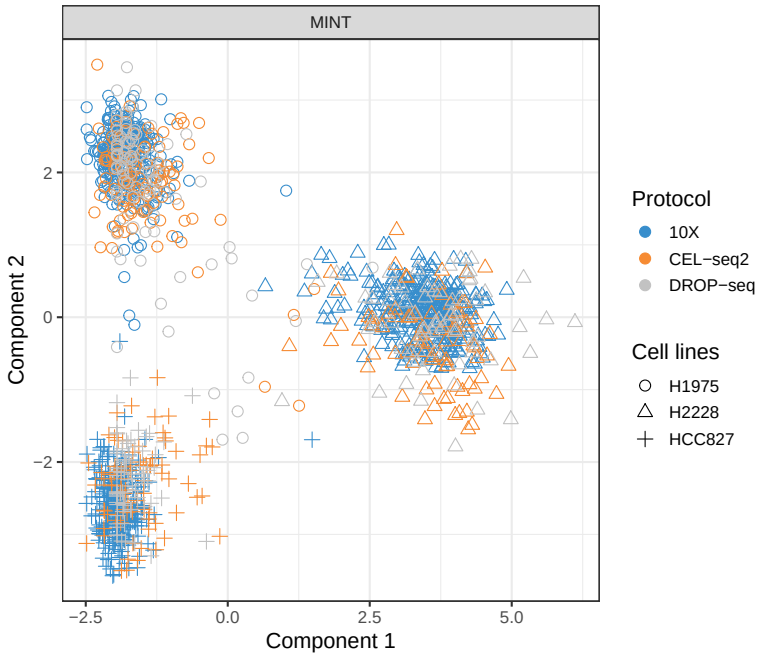
8 sparse MINT with variable selection

With MINT there is not need to select the most variable genes beforehand. We start from the all combined data. The method will select internally the best genes that are agnostic of protocol effect and best discriminate the cell line types.

8.1 Parameters

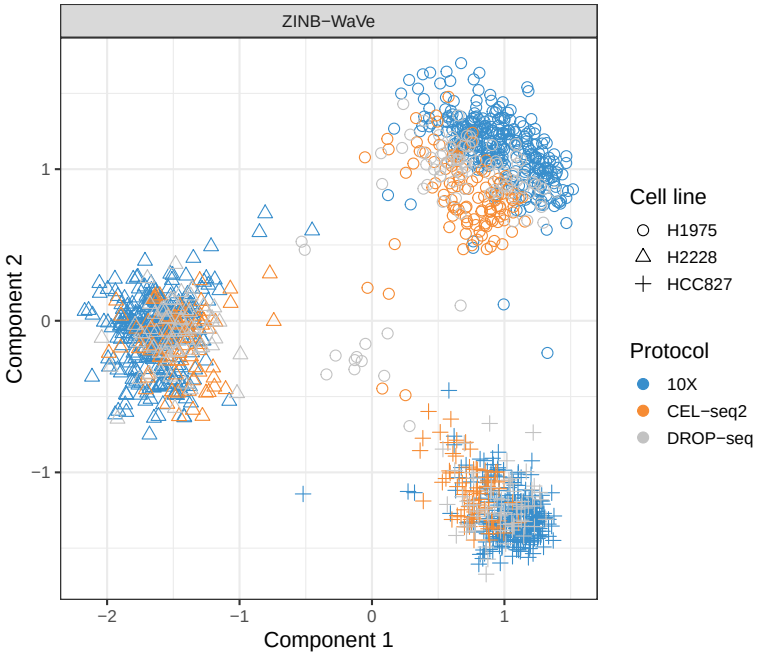
We need to specify the number of variables to select per MINT component. Here we make an arbitrary choice, but we could use the tuning function for an optimal choice, see ??tune.mint.splsda in mixOmics. We provide a full analysis where we explain the tuning step in : https://github.com/AJABADI/MINT_sPLSDA (MINT_Data_Integration). See also another example on microarray data here: http://mixomics.org/mixmint/stemcells-example/.

8.2 MINT variable selection and analysis



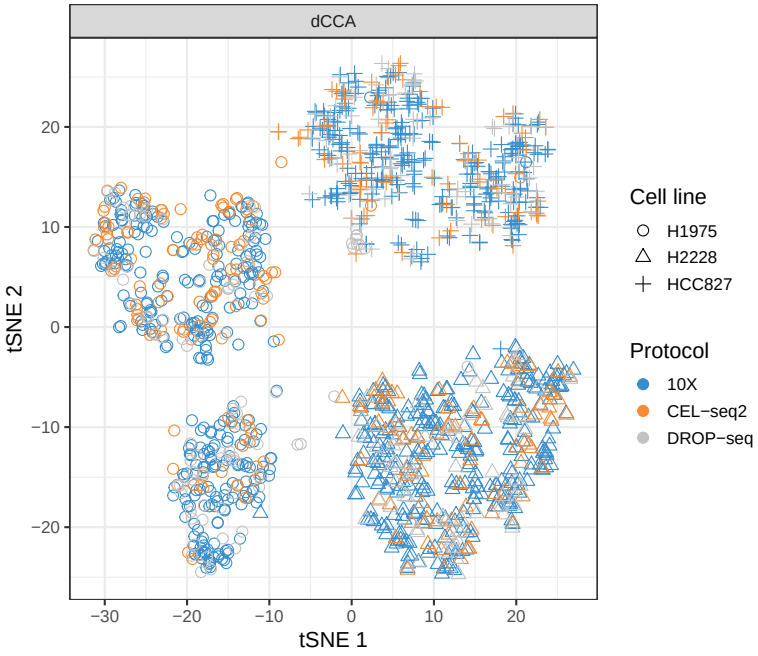
9 ZINB-WaVe

Starting from the highly variable genes identified above. ZINB-WaVe gives us a low-dimensional representation of the data. We can extract the weights from the data to carry on with a differential expression analysis (Van den Berge et al. 2018 and example in https://www.bioconductor.org/packages/devel/bioc/vignettes/zinbwave/inst/doc/intro.html). 'The zinbwave package can be used to compute observational weights to "unlock" bulk RNA-seq tools for single-cell applications, as illustrated in (Van den Berge et al. 2018). Since version 1.1.5, zinbwave computes the observational weights by default. See the man page of zinbwave. The weights are stored in an assay named weights and can be accessed with the following call' (from the ZINB-WaVe vignette).



10 Seurat

We run diagonal CCA from Seurat with 15 components. Visualisation of the reduced dimension is through t-SNE.



11 scMerge

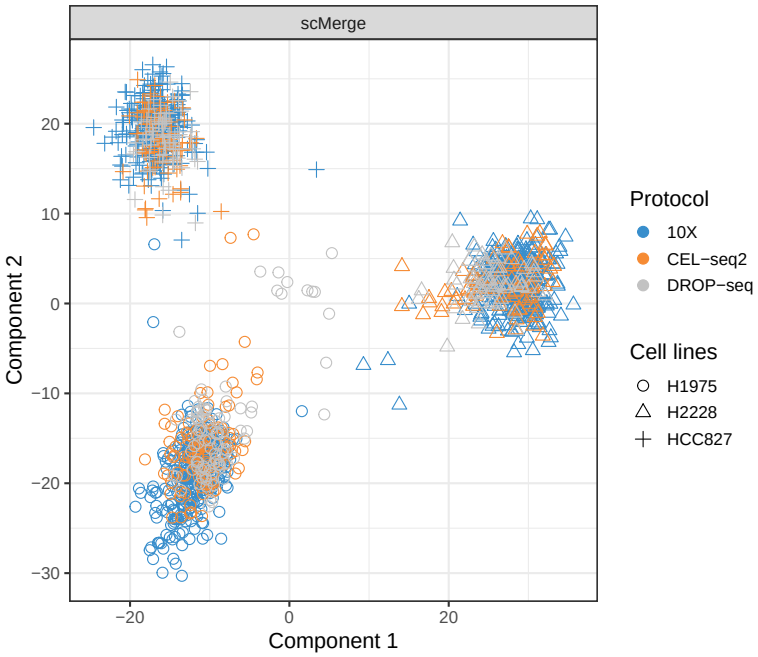
11.1 Identify Stably Expressed Genes

scMerge takes logcounts as input, but we also need to provide the counts to estimate some of the parameters. We identify the SEGs by choosing the most 2000 lowly variable genes per platform and then take the intersection.

We end up with 477 stably expressed genes.

11.2 Unsupervised

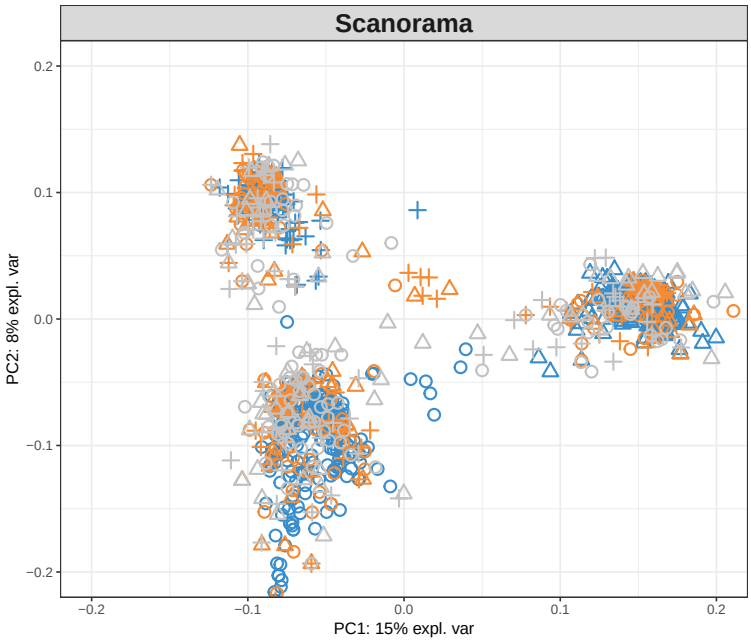
We then run an unsupervised scMerge and run PCA on the resulting data matrix.



11.3 Supervised

12 Scanorama

Scanorama is coded in python, we load the results here of the data matrix and run a PCA. By default the scanorama is run on the most 10,000 HVG



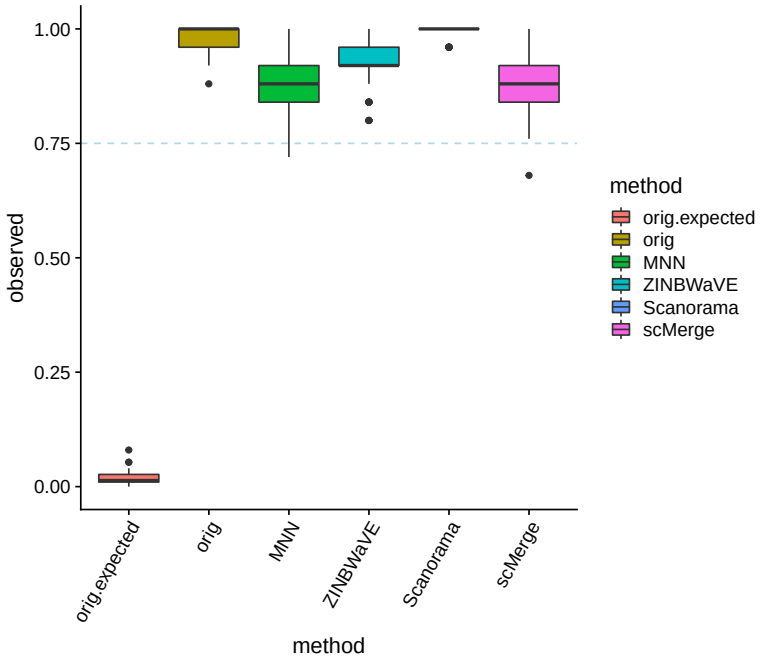
13 Assessment

13.1 kBET evaluation

We run kBET on 25% of the sample size, as advised in the help file. kBET is run on the data matrix resulting from the methods run previously, as well as the original data. We abstrain from running it on the methods that output a reduce dimension (Seurat CCA, MINT).

13.1.1 Summary kBet

We highlight as a horizontal line an acceptance rate of 0.75 (see kBET publication https://www.biorxiv.org/content/biorxiv/early/2017/10/27/200345.full.pdf). For each dataset, kBET returns an overall rejection rate. In our case, a high rejecton rate means that cells are surrounded by samples from the same batch.



13.2 Silhouette width for batch and cell line

To assess the clustering of the data, we use silhouette width, an internal validation metric which is an aggregated measure of how similar an observation is to its own cluster compared its closest neighboring cluster. Here our clusters are already defined, based on either the batch information or the cell type information. The metric ranges from -1 to 1, where higher values indicate a strong cluster.

We calculate the silhouette based on the PCs from PCA for each method that yielded either in a data matrix, or a reduced dimension (in the latter case we calculate the silhouette on those components directly). In our case, a high value for each batch indicate a strong batch effect.

Since we use an Euclidean distance we do not run the Silhouette on Seurat CCA t-SNE components.

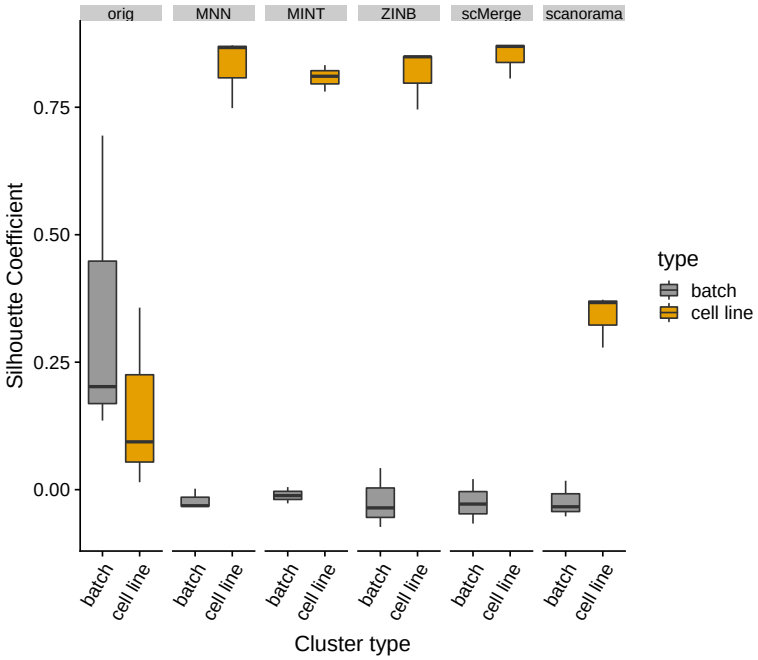
13.2.1 Custom function:

13.2.2 Summary silhouette results

We calculate Silhouette for each method:

Table 3: Summary ARI for components-based methods

	method	ARI
batch	orig	0.75
cell line	orig	0.68
batch	MNN	0.51
cell line	MNN	0.99
batch	MINT	0.51
cell line	MINT	0.99
batch	ZINB	0.51
cell line	ZINB	0.99
batch	scMerge	0.51
cell line	scMerge	0.99
batch	scanorama	0.51
cell line	scanorama	0.75



13.3 ARI

We create a function to calculate the ARI based a distance (euclidean here) and a PAM clustering. The adjusted rand index is then calculate based on the clusters from PAM and the real cluster information (here batch of cell line). A high ARI index with respect to cell line and low with respect to batch ndicates that the method was successful at removing the batch effect whilst retaining the biological information. Here ARI is calculated on component-based methods (from PCA, or directly from ZINB-WaVe or MINT). Seurat CCA was omitted as the reduced representation uses t-SNE.

13.3.1 Custom function

13.3.2 Summary ARI

14 Session information

```
## R version 3.5.0 (2018-04-23)
```

```
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS High Sierra 10.13.3
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_AU.UTF-8/en_AU.UTF-8/en_AU.UTF-8/C/en_AU.UTF-8/en_AU.UTF-8
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] clues_0.5.9                 cluster_2.0.7-1
##  [3] kBET_0.99.5                 scMerge_0.1.8
##  [5] scRNAseq_1.6.0              zinbwave_1.2.0
##  [7] Seurat_2.3.4                Matrix_1.2-14
##  [9] cowplot_0.9.2               mixOmics_6.3.2
## [11] lattice_0.20-35             MASS_7.3-50
## [13] scater_1.8.0                ggplot2_3.0.0
## [15] scran_1.8.2                 SingleCellExperiment_1.2.0
## [17] SummarizedExperiment_1.10.1 DelayedArray_0.6.0
## [19] BiocParallel_1.14.1         matrixStats_0.53.1
## [21] Biobase_2.40.0              GenomicRanges_1.32.3
## [23] GenomeInfoDb_1.16.0         IRanges_2.14.10
## [25] S4Vectors_0.18.3            BiocGenerics_0.26.0
## [27] kableExtra_0.9.0            tictoc_1.0
## [29] knitr_1.20
##
## loaded via a namespace (and not attached):
##   [1] shinydashboard_0.7.0     reticulate_1.9
##   [3] R.utils_2.6.0            tidyselect_0.2.4
##   [5] RSQLite_2.1.1            AnnotationDbi_1.42.1
##   [7] htmlwidgets_1.2          grid_3.5.0
##   [9] trimcluster_0.1-2.1      Rtsne_0.13
##  [11] munsell_0.4.3            codetools_0.2-15
##  [13] ica_1.0-2                statmod_1.4.30
##  [15] DT_0.4                   miniUI_0.1.1.1
##  [17] withr_2.1.2              colorspace_1.3-2
##  [19] pspline_1.0-18           rstudioapi_0.7
##  [21] ROCR_1.0-7               robustbase_0.93-0
##  [23] dtw_1.20-1               gbRd_0.4-11
##  [25] labeling_0.3             Rdpack_0.9-0
##  [27] lars_1.2                 tximport_1.8.0
##  [29] GenomeInfoDbData_1.1.0   bit64_0.9-7
##  [31] rhdf5_2.24.0             rprojroot_1.3-2
##  [33] diptest_0.75-7           R6_2.2.2
##  [35] ggbeeswarm_0.6.0         locfit_1.5-9.1
##  [37] hdf5r_1.0.0              manipulateWidget_0.9.0
##  [39] flexmix_2.3-14           bitops_1.0-6
##  [41] assertthat_0.2.0         promises_1.0.1
##  [43] SDMTools_1.1-221         scales_0.5.0
```

```
##  [45] nnet_7.3-12             beeswarm_0.2.3
##  [47] gtable_0.2.0            rlang_0.2.1
##  [49] genefilter_1.62.0       splines_3.5.0
##  [51] lazyeval_0.2.1          acepack_1.4.1
##  [53] checkmate_1.8.5         rgl_0.99.16
##  [55] yaml_2.1.19             reshape2_1.4.3
##  [57] crosstalk_1.0.0         backports_1.1.2
##  [59] httpuv_1.4.3            Hmisc_4.1-1
##  [61] tools_3.5.0             gplots_3.0.1
##  [63] RColorBrewer_1.1-2      proxy_0.4-22
##  [65] stabledist_0.7-1        dynamicTreeCut_1.63-1
##  [67] ggridges_0.5.0          Rcpp_0.12.17
##  [69] plyr_1.8.4              base64enc_0.1-3
##  [71] zlibbioc_1.26.0         purrr_0.2.5
##  [73] RCurl_1.95-4.10         rpart_4.1-13
##  [75] pbapply_1.3-4           viridis_0.5.1
##  [77] zoo_1.8-2               magrittr_1.5
##  [79] data.table_1.11.4       RSpectra_0.13-1
##  [81] lmtest_0.9-36           RANN_2.6
##  [83] mvtnorm_1.0-8           fitdistrplus_1.0-9
##  [85] gsl_1.9-10.3            hms_0.4.2
##  [87] mime_0.5                evaluate_0.10.1
##  [89] xtable_1.8-2            XML_3.98-1.11
##  [91] mclust_5.4              gridExtra_2.3
##  [93] compiler_3.5.0          ellipse_0.4.1
##  [95] tibble_1.4.2            KernSmooth_2.23-15
##  [97] R.oo_1.22.0             htmltools_0.3.6
##  [99] pcaPP_1.9-73            segmented_0.5-3.0
## [101] corpcor_1.6.9           later_0.7.3
## [103] Formula_1.2-3           snow_0.4-2
## [105] tidyr_0.8.1             DBI_1.0.0
## [107] fpc_2.1-11.1            readr_1.1.1
## [109] R.methodsS3_1.7.1       gdata_2.18.0
## [111] metap_1.0               bindr_0.1.1
## [113] igraph_1.2.2            pkgconfig_2.0.1
## [115] numDeriv_2016.8-1       foreign_0.8-70
## [117] xml2_1.2.0              foreach_1.4.4
## [119] annotate_1.58.0         rARPACK_0.11-0
## [121] vipor_0.4.5             XVector_0.20.0
## [123] bibtex_0.4.2            rvest_0.3.2
## [125] stringr_1.3.1           digest_0.6.15
## [127] copula_0.999-18         tsne_0.1-3
## [129] ADGofTest_0.3           softImpute_1.4
## [131] rmarkdown_1.10          htmlTable_1.12
## [133] edgeR_3.22.2            DelayedMatrixStats_1.2.0
## [135] kernlab_0.9-27          shiny_1.1.0
## [137] gtools_3.5.0            modeltools_0.2-22
## [139] rjson_0.2.20            nlme_3.1-137
## [141] jsonlite_1.5            bindrcpp_0.2.2
## [143] Rhdf5lib_1.2.1          viridisLite_0.3.0
## [145] limma_3.36.1            pillar_1.2.3
## [147] httr_1.3.1              DEoptimR_1.0-8
## [149] survival_2.42-3         glue_1.2.0
## [151] FNN_1.1.2.1             png_0.1-7
```

```
## [153] prabclus_2.2-6           iterators_1.0.9
## [155] glmnet_2.0-16            bit_1.1-14
## [157] mixtools_1.1.0           class_7.3-14
## [159] stringi_1.2.2            blob_1.1.1
## [161] memoise_1.1.0            doSNOW_1.0.16
## [163] latticeExtra_0.6-28      caTools_1.17.1
## [165] dplyr_0.7.6              irlba_2.3.2
## [167] ape_5.1
```