# scAlign Tutorial

*Nelson Johansen, Gerald Quon*

*2019-05-02*

## Introduction

This tutorial provides a guided alignment for two groups of cells from cellbench RNA mixture experiments. In this tutorial we demonstrate the unsupervised alignment strategy of `scAlign` described in Johansen et al, 2018 along with typical analysis utilizing the aligned dataset, and show how `scAlign` can identify and match cell types across platforms without using the labels as input.

### Alignment goals

The following is a walkthrough of a typical alignment problem for `scAlign` and has been designed to provide an overview of data preprocessing, alignment and finally analysis in our joint embedding space. Here, our primary goals include:

1. Learning a low-dimensional cell state space in which cells group by function and type, regardless of condition (platform).
2. Accurately labeling old cells with cell cycle and cell type information using only the young cell annotations.

### Installation

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

```
Guide to installing python and tensorflow on different operating systems.

On Windows:
Download Python 3.6.8. Note, newer versions of Python (e.g. 3.7) cannot use TensorFlow at this time.
Make sure pip is included in the installation.
Open Windows Command Prompt, or PowerShell.
Navigate to your Python installation directory (for Windows 10, the default seems to be C:\Users\userid
Run .\pip install --upgrade tensorflow

On Ubuntu:
sudo apt update
sudo apt install python3-dev python3-pip
pip3 install --user --upgrade tensorflow # install in $HOME
python3 -c "import tensorflow as tf; tf.enable_eager_execution(); print(tf.reduce_sum(tf.random_normal(

On MacOS (homebrew):
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
export PATH="/usr/local/bin:/usr/local/sbin:$PATH"
brew update
brew install python # Python 3
pip3 install --user --upgrade tensorflow # install in $HOME
```

```
python3 -c "import tensorflow as tf; tf.enable_eager_execution(); print(tf.reduce_sum(tf.random_normal(

Further details at: https://www.tensorflow.org/install
```

## Data setup

The gene count matrices used for this tutorial are hosted on the cellbench github: here.

First, we load in the normalized cellbench data. The data was normalized following the procedures defined on the cellbench github.

```
library(scAlign)
library(SingleCellExperiment)
library(ggplot2)

## Load in cellbench data
data("cellbench", package = "scAlign", envir = environment())

## Extract RNA mixture cell types
mix.types = unlist(lapply(strsplit(colnames(cellbench), "-"), "[[", 2))

## Extract Platform
batch = c(rep("CEL", length(which(!grepl("sortseq", colnames(cellbench)) == TRUE))),
          rep("SORT", length(which(grepl("sortseq", colnames(cellbench)) == TRUE))))
```

## scAlign setup

The general design of scAlign's makes it agnostic to the input RNA-seq data representation. Thus, the input data can either be gene-level counts, transformations of those gene level counts or a preliminary step of dimensionality reduction such as canonical correlates or principal component scores. Here we create the `scAlign` object from the previously normalized cellbench data and perform CCA on the unaligned data.

```
## Create SCE objects to pass into scAlignCreateObject
youngMouseSCE <- SingleCellExperiment(
    assays = list(scale.data = cellbench[,batch=='CEL'])
)

oldMouseSCE <- SingleCellExperiment(
    assays = list(scale.data = cellbench[,batch=='SORT'])
)

## Build the scAlign class object and compute PCs
scAlignCB = scAlignCreateObject(sce.objects = list("CEL"=youngMouseSCE,
                                                   "SORT"=oldMouseSCE),
                                labels = list(mix.types[batch=='CEL'],
                                              mix.types[batch=='SORT']),
                                data.use="scale.data",
                                pca.reduce = FALSE,
                                cca.reduce = TRUE,
                                ccs.compute = 5,
                                project.name = "scAlign_cellbench")
```

```
## [1] "Computing CCA using Seurat."
```

```
## Centering and scaling data matrix
## Centering and scaling data matrix
```

```
## Running CCA
```

```
## Merging objects
```

### Alignment of cellbench RNAmixture

Now we align the cell populations from both protocols. `scAlign` returns a low-dimensional joint embedding space where the effect of platform is removed allowing us to use the complete dataset for downstream analyses such as clustering or differential expression.

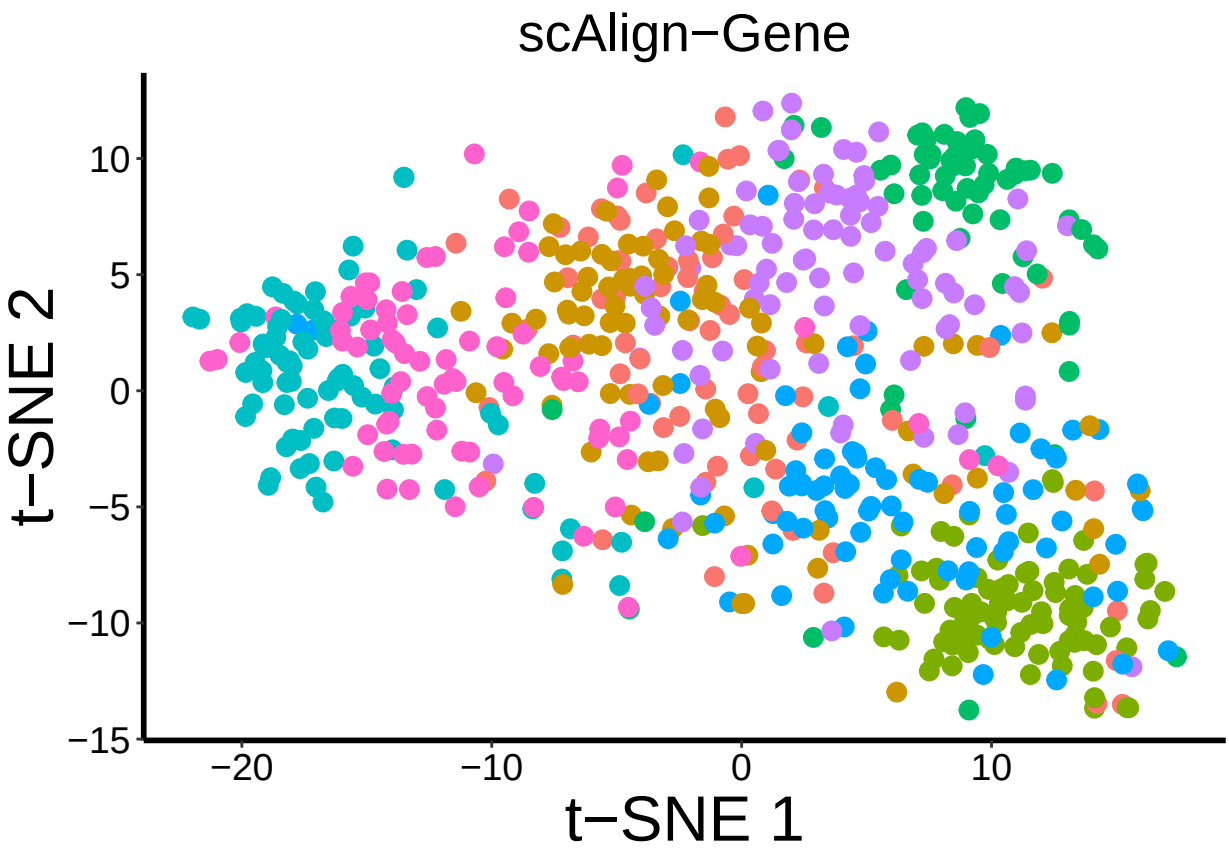
```
## Run scAlign with all_genes
scAlignCB = scAlign(scAlignCB,
                    options=scAlignOptions(steps=1000,
                                           log.every=1000,
                                           norm=TRUE,
                                           early.stop=TRUE),
                    encoder.data="scale.data",
                    supervised='none',
                    run.encoder=TRUE,
                    run.decoder=FALSE,
                    log.dir=file.path('~/models_temp','gene_input'),
                    device="CPU")
```

```
## [1] "============== Step 1/3: Encoder training ==============="
## [1] "Graph construction"
## [1] "Adding source walker loss"
## [1] "Adding target walker loss"
## [1] "Done random initialization"
## [1] "Step: 1    Loss: 68.2143    Alignment: 0.8687"
## [1] "Step: 100    Loss: 49.1458    Alignment: 0.7286"
## [1] "Step: 200    Loss: 39.5291    Alignment: 0.6807"
## [1] "Step: 300    Loss: 32.5206    Alignment: 0.7588"
## [1] "Step: 400    Loss: 27.3723    Alignment: 0.8568"
## [1] "Step: 500    Loss: 26.1896    Alignment: 0.867"
## [1] "Step: 600    Loss: 24.1721    Alignment: 0.8655"
## [1] "Step: 700    Loss: 24.0994    Alignment: 0.8614"
## [1] "Step: 800    Loss: 24.0894    Alignment: 0.8478"
## [1] "Step: 900    Loss: 24.1644    Alignment: 0.8362"
## [1] "Step: 1000    Loss: 24.7685    Alignment: 0.8355"
```

```
# ## Additional run of scAlign with CCA
# scAlignCB = scAlign(scAlignCB,
#                     options=scAlignOptions(steps=1000,
#                                            log.every=1000,
#                                            norm=TRUE,
#                                            early.stop=TRUE),
#                     encoder.data="CCA",
#                     supervised='none',
#                     run.encoder=TRUE,
#                     run.decoder=FALSE,
#                     log.dir=file.path('~/models','cca_input'),
#                     device="CPU")
```

```
## Plot aligned data in tSNE space, when the data was processed in three different ways:
##   1) either using the original gene inputs,
##   2) after CCA dimensionality reduction for preprocessing.
##   Cells here are colored by input labels

set.seed(5678)
gene_plot = PlotTSNE(scAlignCB,
                     "ALIGNED-GENE",
                     title="scAlign-Gene",
                     perplexity=30)
## Show plot
gene_plot
```




```
# cca_plot = PlotTSNE(scAlignCB,
#                     "ALIGNED-CCA",
#                     title="scAlign-CCA",
#                     perplexity=30)
#
# multi_plot_labels = grid.arrange(gene_plot, cca_plot, nrow = 1)
```

## Session Info

```
sessionInfo()
```

```
## R version 3.6.0 (2019-04-26)
```

```
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 18.04.2 LTS
##
## Matrix products: default
## BLAS:   /home/biocbuild/bbs-3.9-bioc/R/lib/libRblas.so
## LAPACK: /home/biocbuild/bbs-3.9-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 utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] ggplot2_3.1.1               SingleCellExperiment_1.6.0
##  [3] SummarizedExperiment_1.14.0 DelayedArray_0.10.0
##  [5] BiocParallel_1.18.0         matrixStats_0.54.0
##  [7] Biobase_2.44.0              GenomicRanges_1.36.0
##  [9] GenomeInfoDb_1.20.0         IRanges_2.18.0
## [11] S4Vectors_0.22.0            BiocGenerics_0.30.0
## [13] scAlign_1.0.0
##
## loaded via a namespace (and not attached):
##   [1] Seurat_3.0.0           Rtsne_0.15             colorspace_1.4-1
##   [4] ggridges_0.5.1         XVector_0.24.0         base64enc_0.1-3
##   [7] listenv_0.7.0          npsurv_0.4-0           ggrepel_0.8.0
##  [10] codetools_0.2-16       splines_3.6.0          R.methodsS3_1.7.1
##  [13] lsei_1.2-0             knitr_1.22             config_0.3
##  [16] jsonlite_1.6           ica_1.0-2              cluster_2.0.9
##  [19] png_0.1-7              R.oo_1.22.0            tfruns_1.4
##  [22] sctransform_0.2.0      compiler_3.6.0         httr_1.4.0
##  [25] assertthat_0.2.1       Matrix_1.2-17          lazyeval_0.2.2
##  [28] htmltools_0.3.6        tools_3.6.0            rsvd_1.0.0
##  [31] igraph_1.2.4.1         gtable_0.3.0           glue_1.3.1
##  [34] GenomeInfoDbData_1.2.1 RANN_2.6.1             reshape2_1.4.3
##  [37] dplyr_0.8.0.1          Rcpp_1.0.1             gdata_2.18.0
##  [40] ape_5.3                nlme_3.1-139           gbRd_0.4-11
##  [43] lmtest_0.9-37          xfun_0.6               stringr_1.4.0
##  [46] globals_0.12.4         irlba_2.3.3            gtools_3.8.1
##  [49] future_1.12.0          MASS_7.3-51.4          zlibbioc_1.30.0
##  [52] zoo_1.8-5              scales_1.0.0           RColorBrewer_1.1-2
##  [55] yaml_2.2.0             reticulate_1.12        pbapply_1.4-0
##  [58] gridExtra_2.3          stringi_1.4.3          tensorflow_1.13.1
##  [61] caTools_1.17.1.2       bibtex_0.4.2           Rdpack_0.11-0
##  [64] SDMTools_1.1-221.1     rlang_0.3.4            pkgconfig_2.0.2
##  [67] bitops_1.0-6           evaluate_0.13          lattice_0.20-38
##  [70] ROCR_1.0-7             purrr_0.3.2            labeling_0.3
##  [73] htmlwidgets_1.3        cowplot_0.9.4          tidyselect_0.2.5
```

```
##  [76] plyr_1.8.4            magrittr_1.5          R6_2.4.0
##  [79] gplots_3.0.1.1        pillar_1.3.1          whisker_0.3-2
##  [82] withr_2.1.2           fitdistrplus_1.0-14   survival_2.44-1.1
##  [85] RCurl_1.95-4.12       tibble_2.1.1          future.apply_1.2.0
##  [88] tsne_0.1-3            crayon_1.3.4          KernSmooth_2.23-15
##  [91] plotly_4.9.0          rmarkdown_1.12        grid_3.6.0
##  [94] data.table_1.12.2     FNN_1.1.3             metap_1.1
##  [97] digest_0.6.18         tidyr_0.8.3           R.utils_2.8.0
## [100] munsell_0.5.0         viridisLite_0.3.0
```