CellScore 1.27.0: Evaluation of Cell Identity

Nancy Mah, Katerina Taškova
nancy.l.mah@googlemail.com, katerina@tashkova.org

November 25, 2024

Contents

1 Introduction

The **CellScore** package contains functions to evaluate the cell identity of a test sample undergoing a cell transition, given a starting (donor) cell type and a desired target cell type. The evaluation is based upon a scoring system, which uses a set of standard samples of known cell types as the reference set. It combines the benefits of two metrics, cosine similarity of expression profiles and fractions of expressed cell type specific genes, into a single score for the cell identity, called CellScore. The CellScore evaluation has been carried out on a large set of microarray data from one platform (Affymetrix Human Genome U133 Plus 2.0). In principle, the method could be applied to any expression dataset, provided that there are a sufficient number of standard samples and that the data are properly normalized (to account for study-specific and platform-specific effects).

2 Installation

This vignette assumes that you have already installed `R` ($\geq$ 4.5.0) and that you have basic working knowledge of `R`. You will additionally need to install the core Bioconductor packages if these have not already been installed:

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

To complete this tutorial, you will to install the **hgu133plus2CellScore** and **CellScore** packages from the Bioconductor repository:

```
BiocManager::install(c("hgu133plus2CellScore", "CellScore"))
```

3 Data preparation for analysis with CellScore

In order to use the package functionality, you will need normalized expression data from both the reference samples (from standard cell types) and from the test samples (from engineered cell types). A pre-selected and formatted dataset of reference samples from the Affymetrix Human Genome U133 Plus 2.0 platform is provided as a separate data package (**hgu133plus2CellScore**). The reference dataset (stored as `eset.std`) includes 837 samples covering over 100 tissues or distinct cell types. To illustrate the usage of the package, a test dataset is also distributed with the package.

For an explicit description of the format of the data and how to generate your own reference data, please see the Appendix A.

4 Example analysis with CellScore

In this section, we include a full analysis on the provided test dataset to demonstrate the functionality of **CellScore**. The typical workflow includes the following steps:

1. *Prepare the data*: A formatted expression dataset from reference cell types profiled with a specific Affymetrix platform is provided as a data package. You will also need to prepare the expression profiles of your test samples in the same way as the reference data, and provide an additional table with the information about the cell transitions that should be evaluated (described in Appendix A).

2. *Calculate the on/off score*: The first metric of CellScore is based on present/absent probe calls between donor and target cell types.

3. *Calculate the cosine similarity*: The second metric of CellScore is the cosine similarity between the standard and the test cell types, based on the normalized expression data.

4. *Generate the CellScore*: This calculates the overall cell identity score of a given test sample and a given cell transition.

5. *Generate the CellScore report*: The PDF report contains plots and heatmaps of the CellScore metrics to help you visualize the progress/direction of the evaluated cell transitions.

4.1 Load the data

Load the packages and standard dataset:

```
library(Biobase)
library(CellScore)
library(hgu133plus2CellScore) # loads eset.std
```

For this vignette, you will need the data from two external files distributed with the **CellScore** package source:

- `eset48.RData`: An `R` data file that stores `eset48`, an *ExpressionSet* object with the normalized gene expression values of 48 samples from engineered (derived) cells representing several cell transitions.

- `cell_change_test.tsv`: A tab-delimited text file with the information on the cell transitions that should be evaluated; each row depicts a single cell transition.

These data can be loaded with the code below, which also combines the gene expression values of the test samples (`eset48`) with the ones of the standard reference samples (`eset.std`) in one single *ExpressionSet* object `eset`.

```
## Locate the external data files in the CellScore package
rdata.path <- system.file("extdata", "eset48.RData",
                          package="CellScore")
tsvdata.path <- system.file("extdata", "cell_change_test.tsv",
                            package="CellScore")

if (file.exists(rdata.path) && file.exists(tsvdata.path)) {
    ## Load the normalized expressions of 48 test samples
    load(rdata.path)
```

```
    ## Import the cell change info for the loaded test samples
    cell.change <- read.delim(file=tsvdata.path, sep="\t",
                              header=TRUE, stringsAsFactors=FALSE)
    print("Content of cell.change")
    print(cell.change)

    ## Combine the standards and the test data
    eset <- combine(eset.std, eset48)
    print("dim(eset) returns")
    print(dim(eset))
}
```

```
## [1] "Content of cell.change"
##   start     test target
## 1   FIB  iPS-FIB    ESC
## 2   FIB piPS-FIB    ESC
## 3   KER  iPS-KER    ESC
## [1] "dim(eset) returns"
## Features  Samples
##    19851      885
```

The particular examples covered in this tutorial (`cell.change`) include two cell transitions producing three different derived cell types: induced pluripotent stem cells derived from fibroblasts or keratinocytes (iPS-FIB and iPS-KER, respectively), and partially induced pluripotent stem cells from fibroblast (piPS-FIB).

4.2 Calculate the on/off score

The first CellScore metric we will calculate is the on/off score. This score is based on the fraction of donor markers lost and the fraction of target markers gained by the derived cells in a given cell transition. It can be calculated on a sample level (for each sample individually) or on a group level (for each derived cell type, aggregating the information across all samples). The function call below calculates the on/off scores for each derived cell type listed in `cell.change`. The outcome is a list of two data frames.

```
group.OnOff <- OnOff(eset, cell.change, out.put="marker.list")
summary(group.OnOff)
```

```
##         Length Class      Mode
## scores  10     data.frame list
## markers  9     data.frame list
```

In this example, we see that as a group, the iPS-KER samples have the best on/off score (high donor marker loss and high target marker gain).

```
head(group.OnOff$scores)
```

```
##   start target     test markers.start markers.target
## 1   FIB    ESC  iPS-FIB           162            388
## 2   FIB    ESC piPS-FIB           162            388
## 3   KER    ESC  iPS-KER           262            440
```

```
##   start.mkrs.in.test target.mkrs.in.test loss.start.mkrs
## 1                  0                 108       1.0000000
## 2                 38                  29       0.7654321
## 3                  1                 301       0.9961832
##   gain.target.mkrs OnOffScore
## 1       0.27835052  1.2783505
## 2       0.07474227  0.8401744
## 3       0.68409091  1.6802741
```

The OnOff() function also outputs a data frame of marker genes used for the calculation of the on/off score. A data frame of the marker genes can be found here:

```
head(group.OnOff$markers)
```

```
##       comparison group    probe_id    median platform_id gene_symbol
## 1.94    FIB->ESC   FIB   226950_at 0.4008768      GPL570      ACVRL1
## 1.623   FIB->ESC   FIB   207510_at 0.4080080      GPL570      BDKRB1
## 1.624   FIB->ESC   FIB   205870_at 0.4056214      GPL570      BDKRB2
## 1.649   FIB->ESC   FIB   202701_at 0.6121189      GPL570        BMP1
## 1.683   FIB->ESC   FIB   205715_at 0.2365783      GPL570        BST1
## 1.837   FIB->ESC   FIB 209310_s_at 0.3739509      GPL570       CASP4
##                                          gene_name entrezgene_id
## 1.94                activin A receptor type II-like 1            94
## 1.623                          bradykinin receptor B1           623
## 1.624                          bradykinin receptor B2           624
## 1.649                    bone morphogenetic protein 1           649
## 1.683               bone marrow stromal cell antigen 1           683
## 1.837 caspase 4, apoptosis-related cysteine peptidase           837
##        feature_id
## 1.94    226950_at
## 1.623   207510_at
## 1.624   205870_at
## 1.649   202701_at
## 1.683   205715_at
## 1.837 209310_s_at
```

Next, we calculate the on/off score for all samples in the expression dataset. This is provided as a separate step since it can take some time to calculate, depending on how many samples there are in the `eset` object. In this case, it was fast and took a few seconds of CPU time on an Intel Core i7-5600U CPU @ 2.60GHz.

```
individ.OnOff <- OnOff(eset, cell.change, out.put="individual")
```

The calculations are done and it is a good idea to save the calculated scores:

```
save(file="OnOffscore.RData", individ.OnOff, group.OnOff)
```

Things to check at this point: Looking at the on/off score of individual samples that make up the standards, check if there are any clear outliers. For example, these outliers may be caused by wrong annotation or modifications to the cell line that make them unsuitable as standards. We can either

eliminate these samples form the analysis, or keep them but score them for different cell transitions. For this purpose, the values of the `category` column in `eset@phenoData@data` (the *phenotype data frame*, described in Appendix A) can be set to "NA" or "unknown". If the category is "NA", the corresponding sample will not be scored. If the category is "unknown", the corresponding sample will be scored for all available cell transitions specified in `cell.change`.

For example, look at all samples that are embryonic stem cells (ESC) in the transition from FIB to ESC:

```
sel.transition <- individ.OnOff$scores$start == "FIB" &
                  individ.OnOff$scores$target == "ESC"
sel.esc <- grepl("embryonic stem cell", individ.OnOff$scores$cell_type)
onoff.sel <- individ.OnOff$scores[sel.esc & sel.transition,
                                  c(4,6,7,12,13,19,20,21)]
```

Ideally, standard ESC samples in the transition from FIB to ESC should have an on/off score close to 2. However, the scores of the ESC cells ranges from 1.44 to 1.98:

```
summary(onoff.sel$OnOffScore)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.445   1.893   1.931   1.908   1.946   1.982
```

These samples are clear outliers and therefore already have been removed from the ESC standards by assigning them to the category "test".

```
onoff.sel[order(onoff.sel$OnOffScore)[1:4],]
```

```
##                                          cell_type category
## 1.GPL570.GSE12583.GSM315621  embryonic stem cell line     test
## 1.GPL570.GSE12583.GSM315622  embryonic stem cell line     test
## 1.GPL570.GSE23583.GSM579901       embryonic stem cell standard
## 1.GPL570.GSE50868.GSM1231391 embryonic stem cell line standard
##                              general_cell_type start target
## 1.GPL570.GSE12583.GSM315621              ESC.x   FIB    ESC
## 1.GPL570.GSE12583.GSM315622              ESC.x   FIB    ESC
## 1.GPL570.GSE23583.GSM579901                ESC   FIB    ESC
## 1.GPL570.GSE50868.GSM1231391               ESC   FIB    ESC
##                              loss.start.mkrs gain.target.mkrs
## 1.GPL570.GSE12583.GSM315621       0.5246914        0.9201031
## 1.GPL570.GSE12583.GSM315622       0.5987654        0.9046392
## 1.GPL570.GSE23583.GSM579901       0.8148148        0.9871134
## 1.GPL570.GSE50868.GSM1231391      0.8888889        0.9381443
##                              OnOffScore
## 1.GPL570.GSE12583.GSM315621    1.444794
## 1.GPL570.GSE12583.GSM315622    1.503405
## 1.GPL570.GSE23583.GSM579901    1.801928
## 1.GPL570.GSE50868.GSM1231391   1.827033
```

Finally, you can plot the group-wise on/off scores in a pyramid barplot (see Figure 1). The function `BarplotOnOff()` also returns the data (a list of two data frames) used for making the barplot:

```
barplot.out <- BarplotOnOff(eset, group.OnOff$scores)
barplot.out
```

```
## $GroupComparisonsForPlot
##   start target     test markers.start markers.target
## 2   FIB    ESC piPS-FIB           162            388
## 1   FIB    ESC  iPS-FIB           162            388
## 3   KER    ESC  iPS-KER           262            440
##   start.mkrs.in.test target.mkrs.in.test loss.start.mkrs
## 2                 38                  29       0.7654321
## 1                  0                 108       1.0000000
## 3                  1                 301       0.9961832
##   gain.target.mkrs OnOffScore
## 2       0.07474227  0.8401744
## 1       0.27835052  1.2783505
## 3       0.68409091  1.6802741
##
## $OnOffBarplotData
##   position    markers       markername     tags
## 1    Below 0.76543210  loss.start.mkrs piPS-FIB
## 2    Below 1.00000000  loss.start.mkrs  iPS-FIB
## 3    Below 0.99618321  loss.start.mkrs  iPS-KER
## 4    Above 0.07474227 gain.target.mkrs piPS-FIB
## 5    Above 0.27835052 gain.target.mkrs  iPS-FIB
## 6    Above 0.68409091 gain.target.mkrs  iPS-KER
```

The plot in Figure 1 can be easily saved in a PDF format as follows.

```
pdf(file="GroupOnOffScore_Barplot.pdf")
barplot.out <- BarplotOnOff(eset, group.OnOff$scores)
dev.off()
```

4.3 Calculate the cosine similarity

The second metric for cell scoring is the cosine similarity that can be calculated with the function `CosineSimScore()`. First, the expression matrix (containing only standard samples) is filtered for the most variable probes/genes, as defined by the interquartile range (IQR). By default we keep only the top 10% of the genes ranked by IQR, and this cutoff can be changed with the function argument `iqr.cutoff`. Then the mean centroid of the expression values for each standard cell type (defined by `eset$general_cell_type`) is calculated. Finally, the cosine similarity is calculated between each centroid and sample.

Note that this function can take few minutes to calculate, depending on how many samples there are in the `eset` object. So, start the execution and go for a coffee break ...

```
tmp.time <- system.time(cs <- CosineSimScore(eset, cell.change,
                                             iqr.cutoff=0.1))
tmp.time
```

```
##    user  system elapsed
##  39.709   3.474  43.194
```

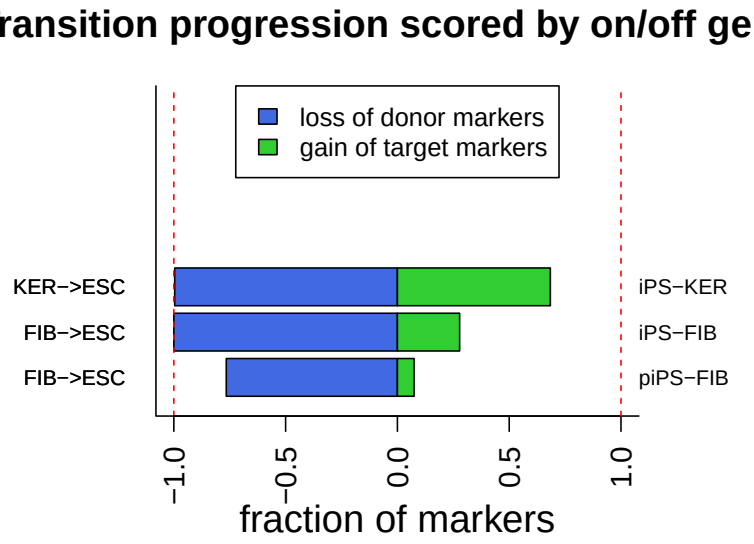

Figure 1: Pyramid barplot of on/off scores. The horizontal barplot shows the fraction of donor markers lost (blue) and the fraction of target markers gained (green). Each bar shows the results of one cell transition (left margin) for a particular derived cell type (right margin). The longer the coloured bars, the more successful the transition.

In our case, the calculation took approximately 40 seconds of CPU time on an Intel Core i7-5600U CPU @ 2.60GHz. The result of the calculations is a list of five data objects:

- `cs$pdataSub`: A data frame with the phenotype of the reference samples.
- `cs$esetSub.IQR`: An *ExpressionSet* object with the gene-filtered expression profiles of the reference samples.
- `cs$cosine.general.groups`: A matrix with the values of cosine similarity between all general groups defined by `eset$general_cell_type`
- `cs$cosine.subgroups`: A matrix with the values of cosine similarity between all sub groups defined by `eset$sub_cell_type1`
- `cs$cosine.samples`: A matrix with the values of cosine Similarity between all samples, general groups and subgroups

Now we can visualize the cosine similarity values in a heatmap. The function `PlotCosineSimHeatmap()` will generate a PDF file. For example, we can plot the cosine similarity between

- the mean centroid of all general cell types, annotated by `eset$general_cell_type`

```
PlotCosineSimHeatmap(cs$cosine.general.groups, "general groups",
                     width=20, height=20, x=-20, y=3)
```

- the mean centroid of all subgroup cell types, annotated by `eset$sub_cell_type1`

```
PlotCosineSimHeatmap(cs$cosine.subgroups, "subgroups",
                     width=14, height=14, x=-14, y=3)
```

- all samples and subgroups; note that this plot can be enormous, depending on the number of samples, so you need to adjust the page size to make it viewable.

```
PlotCosineSimHeatmap(cs$cosine.samples, "samples",
                     width=50, height=50, x=-50, y=10)
```

Rather than plotting the cosine similarity for all samples and subgroups, you can pick which standards and test samples to plot. For example, plot the general groups (fibroblast and embryonic stem cells) for the transition from FIB to ESC and the relevant test cells for this cell transition.

```
## Get the names (IDs) of the sample and their description
samples.cs <- colnames(cs$cosine.samples)
samples.eset <- sampleNames(eset)

## Select the samples of interest and their corresponding score
sel.ips <- eset$category == "test" &
           eset$sub_cell_type1 %in% c("piPS-FIB", "iPS-FIB")
sel <- samples.cs %in% c(c("FIB", "ESC"), samples.eset[sel.ips])
cs.sel <- cs$cosine.samples[sel, sel]

## Rename columns/rownames to more descriptive labels
## as cs.sel is a symetric matrix, these are identical
ids <- match(colnames(cs.sel), samples.eset)
ids.na <- is.na(ids)
ids.rest <- na.omit(ids)
new.colnames <- c(colnames(cs.sel)[ids.na],
                  paste(eset$sub_cell_type1[ids.rest],
                        eset$sample_id[ids.rest],
                        sep="_")
                  )
colnames(cs.sel) <- rownames(cs.sel) <- new.colnames

## Plot the heatmap
PlotCosineSimHeatmap(cs.sel, "piPS", width=10, height=10, x=-10, y=3)
```

Notice that in the last plot, many of the partial iPS cell lines are more similar to FIB rather than to ESC, which is to be expected (see Figure 2)

As an additional sanity check, we can perform a Principal Component Analysis (PCA) and plot the data in the space of the the first two principal components. For example, the PCA plot of the standard reference samples (used to

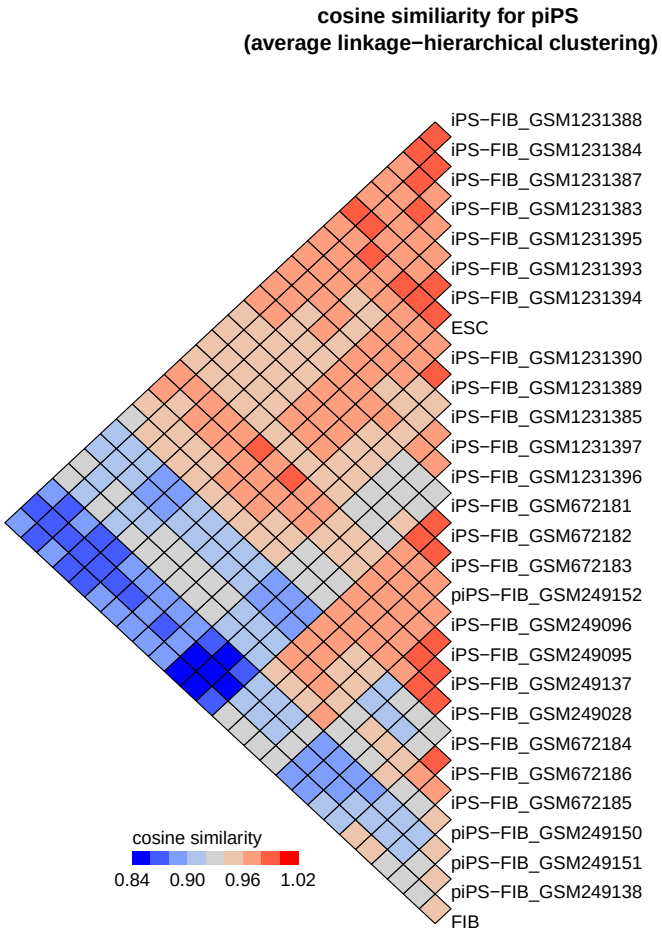

Figure 2: Heatmap of cosine similarity. The triangular heatmap shows the cosine similarity calculated between the centroids of the donor (fibroblasts; FIB) and desired target cells (embryonic stem cells; ESC). In addition, it shows the similarity between the individual samples of two cell transitions, that is iPS-FIB and piPS-FIB.

calculate the cosine similarity) should show that the similar cell types cluster closer together. The `PcaStandards()` function can be used to generate such a plot. The same allows the samples to be colored according to different properties of the samples:

- experiment ID

```
PcaStandards(cs$pdataSub$experiment_id, "Experiment ID",
             cs$esetSub.IQR)
```

- general cell type (see Figure 3):

```
PcaStandards(cs$pdataSub$general_cell_type, "General Cell Type",
             cs$esetSub.IQR)
```

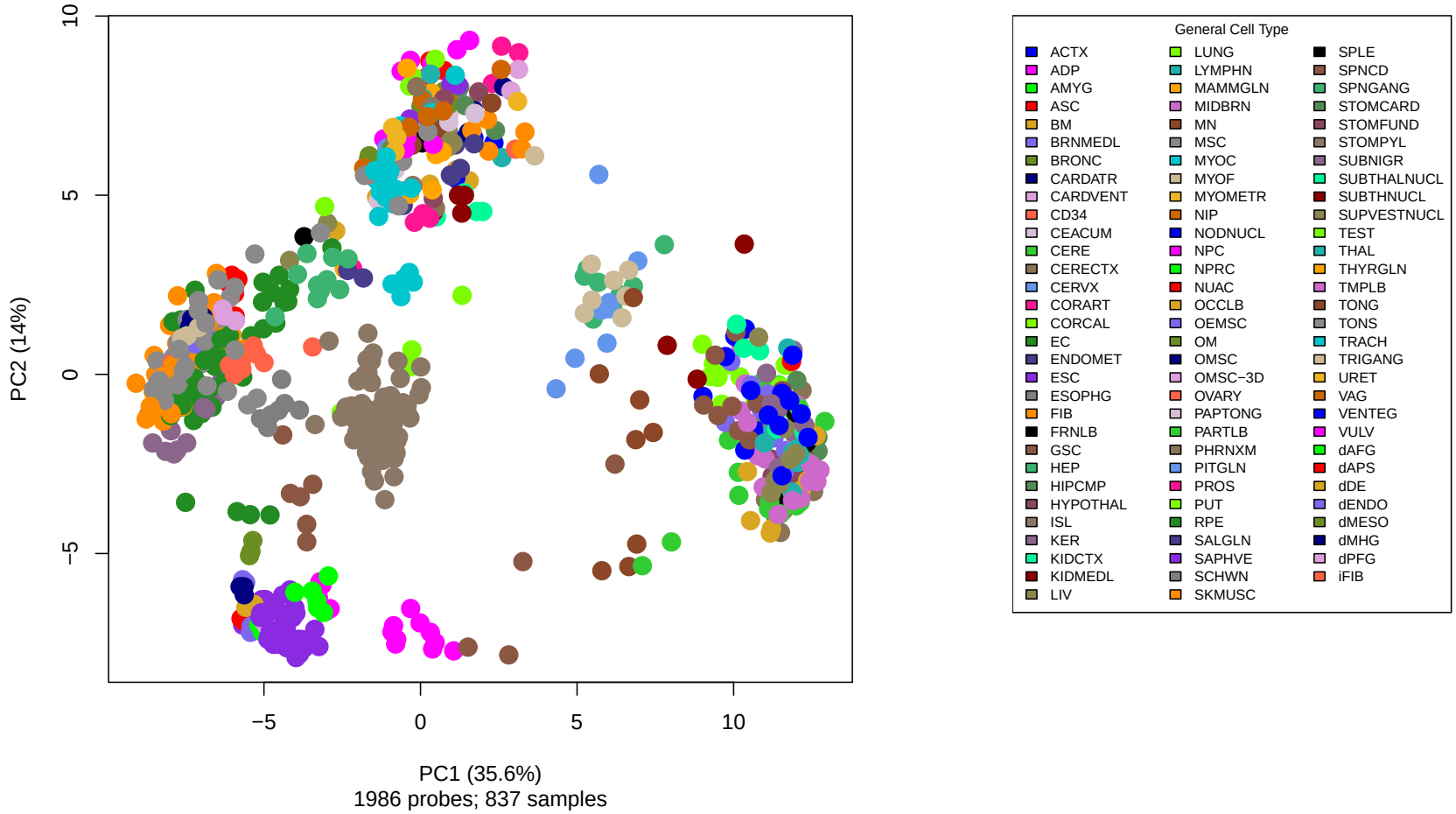

Figure 3: Principal component analysis of the reference dataset. The reference samples cover a wide range of tissues and cell types from many studies and are the basis for comparison in the CellScore method. The analysis was applied on the expression matrix corresponidng to the most variable genes (as defined by the IQR cutoff). The plot, based on the first two prinicipal components, shows that similar cell types tend to cluster together. Brain tissue clusters on the right side, pluripotent and multipotent stem cells at the bottom, and somatic cell types on the left and upper clusters.

To identify specific sample (points) on the PCA plot you could plot the figure including a text label for each sample. Samples with missing label ("NA") will have no text annotation on the plot:

```
pdf(file="StandardSamples_PCA_Labels.pdf", width=28, height=14)
PcaStandards(cs$pdataSub$general_cell_type, "General Cell Type",
             cs$esetSub.IQR)
PcaStandards(cs$pdataSub$general_cell_type, "General Cell Type",
             cs$esetSub.IQR,
             text.label=cs$pdataSub$general_cell_type)

dev.off()
```

4.4 Generate the CellScore

Now that we've calculated the on/off scores and cosine similarities, it's straightforward to calculate the CellScores for every sample in the expression matrix.

```
cellscore <- CellScore(data=eset, transitions=cell.change, scores.onoff=individ.OnOff$scores,
                       scores.cosine=cs$cosine.samples)
```

Finally, we can save all the scores and data in one file for later manipulations.

```
save(file="VignetteResults.RData",
     eset,                          # the combined expression dataset
     group.OnOff, individ.OnOff,    # the on/off score values
     cs,                            # the cosine similaritiy values
     cellscore                      # the CellScore values
     )
```

4.5 Generate CellScore reports

In the final step, we will generate some diagnostic plots and save all the plots in a summary report outputted to a PDF file.

4.5.1 Scatter plot of donor-like and target-like scores

In the scatter plot of all test samples, you can quickly identify samples that have not completely transitioned to the desired cell type. Ideally, samples with good transition to the target cell type are located in the upper left-hand corner. Samples with poor transition will retain more donor-like expression and tend to be located on the right-hand side of the plot (see Figure 4):

The following will generate a two-paneled plot.

```
ScatterplotCellScoreComponents(cellscore, cell.change, FALSE)
```

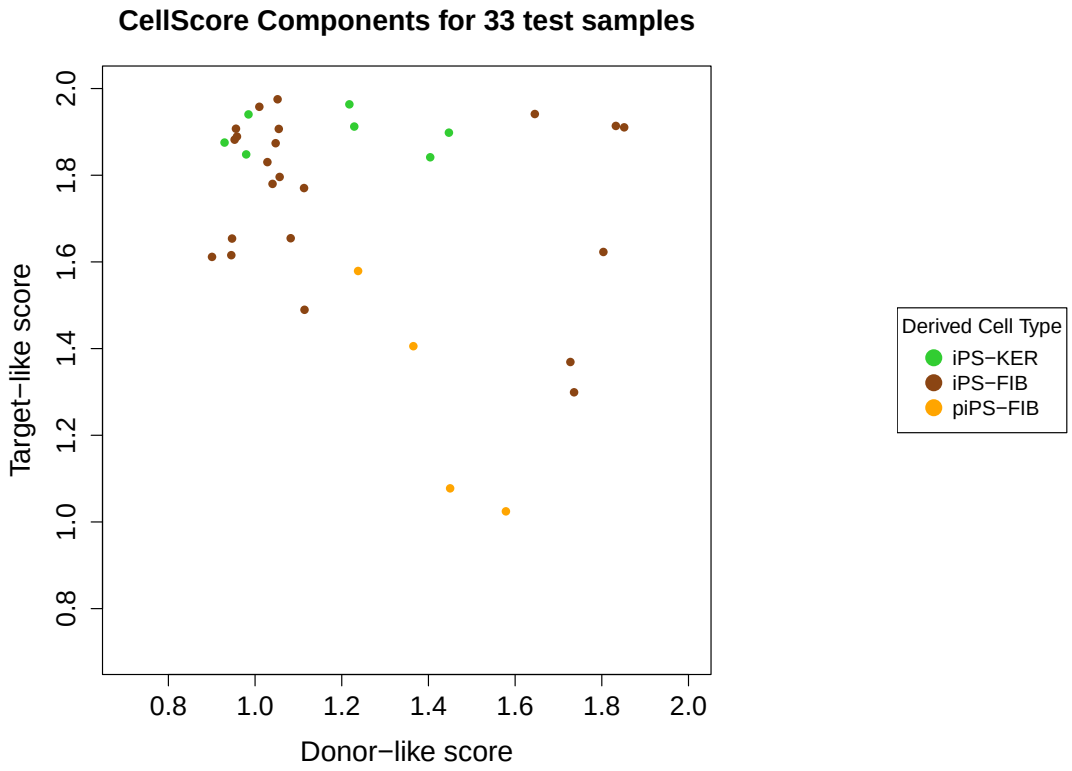

Figure 4: Scatter plot of donor-like and target-like scores. Derived cell types with the most successful transition have low donor-like score and high target-like scores and should cluster in the upper left-hand corner. In this example, the partially reprogrammed iPS cells (piPS-FIB) show gradual transition to their desired target cell type. A few iPS-FIB samples retain high donor-like scores, indicating unusual properties of these lines.

If there are too many points to plot, you could also choose which transitions to plot. For example, the code below will only plot the partial iPS cells:

```
ScatterplotCellScoreComponents(cellscore, cell.change[2,], FALSE)
```

4.5.2 Boxplot of CellScore values

The `BoxplotCellScore` function plots an overview of the CellScore values across all the test samples, grouped by subgroup. The minimum score is about -1.2, and the maximum score is about 1.2 (see Figure 5):

```
BoxplotCellScore(cellscore, cell.change)
```

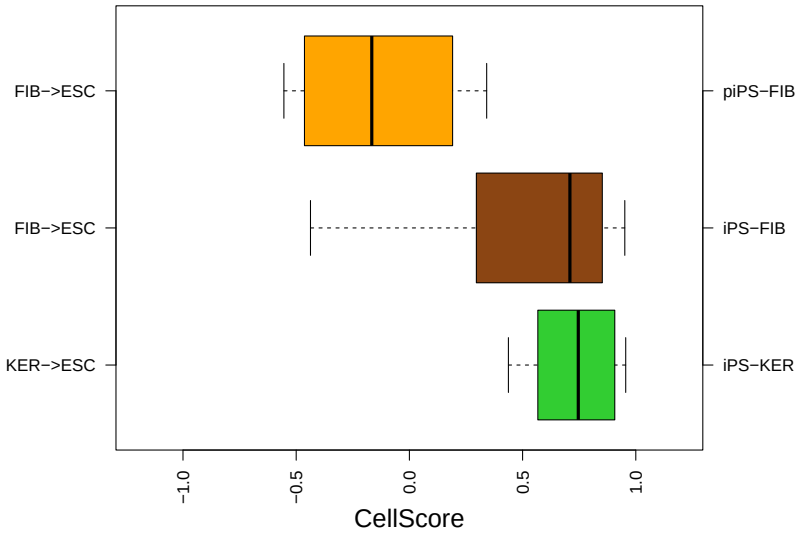

Figure 5: Boxplot of CellScore values by subgroups.

4.5.3 Rug plot of CellScore values

The `RugplotCellScore()` function generates a rug plot of the CellScore values by experiment, and the samples will be colored by a secondary property from the *phenotype data frame*. In this case, we choose to color the samples by the values in the `transition_induction_method` column. (Figure 6).

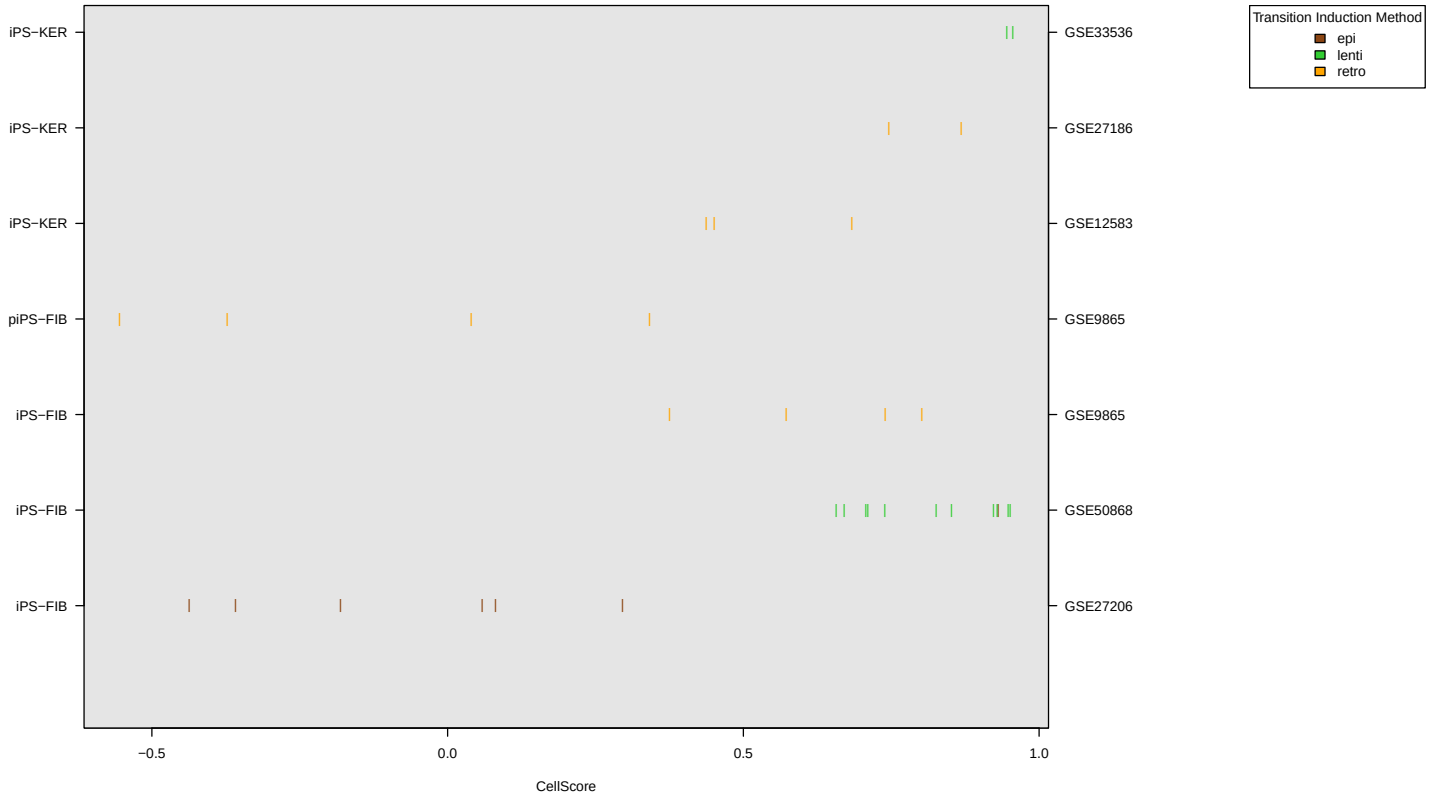

Figure 6: Rug plot of CellScore values. The CellScore values are plotted for each derived cell type (left margin) within a study (GEO accession numbers in left margin). Each test sample is represented by a vertical line and colored by its transition induction method.

4.5.4 The summary report

Finally, the last plotting function will generate a CellScore report for each study and cell transition that has been defined in the `cell.change` data frame.

As the report consists of many pages, it is best to simply plot everything to a single PDF file.

```
pdf(file="CellScoreReport_PerTransition.pdf", width=7, height=11)
CellScoreReport(cellscore, cell.change, group.OnOff$markers, eset)
dev.off()
```

The report is composed of several plots, including

- *scatter plot of the donor- and target-like scores*: Test samples, along with the standards for donor and target, are shown on the scatter plot for easy comparison (see Figure 7).
- *density plot of the CellScore values*: The CellScore values of the test samples should be located somewhere between the CellScore values for the donor and target standards (see Figure 8, upper panel).
- *rug plot of the CellScore values*: This plot has a closer look at the test samples and the target standards. Dashed vertical lines indicate the number of standard deviations from the mean target CellScore (see Figure 8, lower panel).
- *heatmap of the donor and target markers*: It gives a quick overview of the number of marker genes being expressed above detection level (as defined by the present calls in the *calls matrix*) in the test samples (see Figure 9).

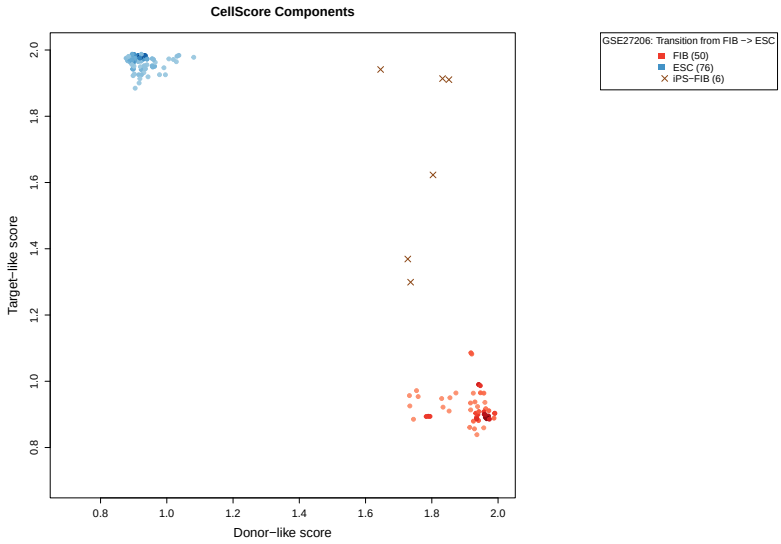

Figure 7: Scatter plot of CellScore components. The first plot of the CellScore report shows a scatter plot of the donor-like and target-like scores of the donor standard (in this case, fibroblasts (FIB); red) and target standard (embryonic stem cell (ESC); blue), as well as the derived cell types (induced pluripotent stem cells from fibroblasts (iPS-FIB); brown crosses). The number of samples from each group is indicated in parentheses in the figure legend.

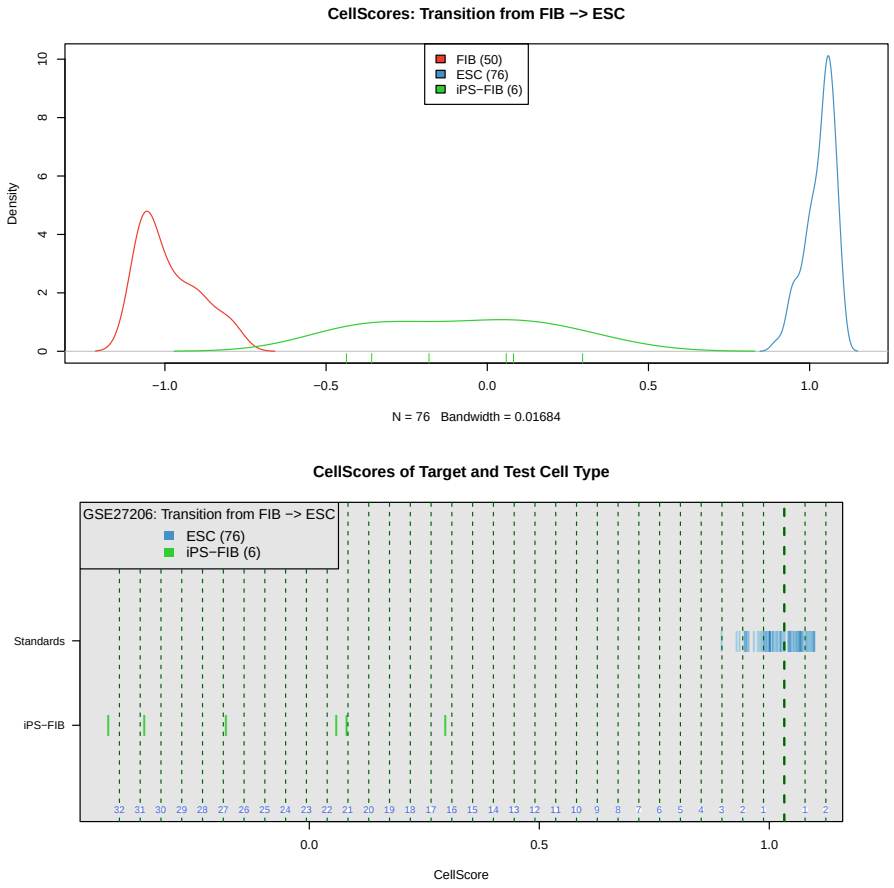

Figure 8: Density and rug plots of CellScore values. The CellScore values are shown as a density plot (upper panel) and as a rug plot (lower panel). The rug plot displays the test cells (iPS-FIB) in relation to the desired target cell type (ESC). Vertical dashed lines indicate the number of standard deviations away from the mean CellScore (bold vertical dashed line) of the target cell type.

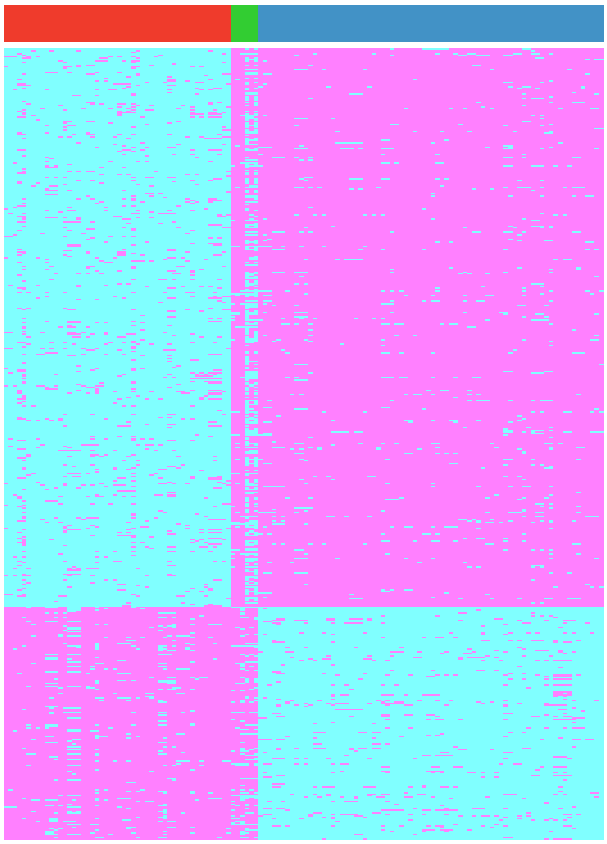

Figure 9: Heatmap of donor and target markers. Fibroblasts (red) are the donor cells, and embryonic stem cells are the desired target cells (blue). The test cells are induced pluripotent stem cells from fibroblasts (iPS-FIB; green).

4.6 R session information

```
sessionInfo()

## R Under development (unstable) (2024-11-20 r87352)
## Platform: aarch64-apple-darwin20
## Running under: macOS Ventura 13.7.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRlapack.dylib;  LAPAC
##
## locale:
## [1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/New_York
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods
## [7] base
##
## other attached packages:
## [1] hgu133plus2CellScore_1.27.0 CellScore_1.27.0
## [3] knitr_1.49                  Biobase_2.67.0
## [5] BiocGenerics_0.53.3         generics_0.1.3
##
## loaded via a namespace (and not attached):
##  [1] Matrix_1.7-1                jsonlite_1.8.9
##  [3] compiler_4.5.0              highr_0.11
##  [5] crayon_1.5.3                gtools_3.9.5
##  [7] SummarizedExperiment_1.37.0 GenomicRanges_1.59.1
##  [9] bitops_1.0-9                IRanges_2.41.1
## [11] lattice_0.22-6              R6_2.5.1
## [13] XVector_0.47.0              SnowballC_0.7.1
## [15] S4Arrays_1.7.1              GenomeInfoDb_1.43.1
## [17] DelayedArray_0.33.2         MatrixGenerics_1.19.0
## [19] GenomeInfoDbData_1.2.13     RColorBrewer_1.1-3
## [21] xfun_0.49                   caTools_1.18.3
## [23] SparseArray_1.7.2           zlibbioc_1.53.0
## [25] grid_4.5.0                  S4Vectors_0.45.2
## [27] squash_1.0.9                KernSmooth_2.23-24
## [29] evaluate_1.0.1              abind_1.4-8
## [31] stats4_4.5.0                lsa_0.73.3
## [33] httr_1.4.7                  matrixStats_1.4.1
## [35] tools_4.5.0                 UCSC.utils_1.3.0
## [37] gplots_3.2.0
```

Appendices

A Specifications for the input datasets

A.1 Reference dataset

If you have test samples from another microarray platform, you may want to build your own reference data object to suit your own needs. In this case you have to select your own reference samples. This can be as extensive as including many cell types and biological replicates, or as little as only providing some biological replicates for standard (donor and target) and test cell types. You must have at least three samples in each standard (donor/target), and at least one sample from a test cell type. The method will perform better if there are more samples, given that the quality of the samples is reasonable. The advantage of having more samples is to capture a robust signal, which should be representative for all samples of the same comparison. Limiting samples to only a handful of replicates runs the risk of focussing on study-specific effects.

The expression data should be transformed into an object of the *ExpressionSet* class (defined in the Biconductor package **Biobase**). For in-depth description of the *ExpressionSet* class please refer to the **Biobase** package docmentation. In the following, we will describe only the relevant data structures for **CellScore**.

1. *Calls matrix* with probe IDs in rows, samples in columns; 1=present, 0=absent. The probe IDs should be located in the rownames of the matrix.

2. *Normalized expression matrix* with probe IDs in rows, samples in columns. The probe IDs should be located in the rownames of the matrix. This matrix MUST have the same dimensions and row order as the *calls matrix*.

3. *Annotation data frame* for the probe IDs in the *calls matrix*. The same probe IDs used in the *calls matrix* MUST be used as the rownames of this data frame, and their order should match the order of the *calls matrix*. There should be no duplicate probe IDs. This data frame must contain the following five columns:

   - `probe_id`
   - `platform_id`
   - `gene_symbol`
   - `gene_name`
   - `entrezgene_id`

   Other columns will be ignored.

4. *Phenotype data frame* with information about the samples in the expression matrix. Samples are in rows and columns are attributes of the sample. The rownames of the data frame must be the sample IDs and must exactly match the column names of the *calls matrix* and *normalized expression matrix*. This data frame must contain the following columns:

- `experiment_id`: This should be a unique identifier for an experiment, for example a GEO experiment ID or an ArrayExpress experiment ID.
- `sample_id`: Sample IDs should be unique, such as the GSM accession numbers from GEO.
- `platform_id`: Use a unique ID for each platform.
- `cell_type`: Use a short text description here.
- `category`: Each sample must be assigned to one of "standard" / "test" / "NA"
- `general_cell_type`: Use an abbreviation to label the general cell type. For example, FIB for fibroblast.
- `donor_tissue`: If the sample is a derived cell type, then enter the abbreviation for the donor cell type. If the sample is standard cell type, then enter the donor tissue from which it was isolated. Otherwise, enter "NA".
- `sub_cell_type1`: The sub-cell type is a compound term of the general cell type and its donor tissue.

Additional columns are optional. The properties in these columns could be used to color the individual samples in the rug plots (Figure 6). For example, the samples could be colored by

- `transition_induction_method`, the method used to engineer the derived cell types, or
- `donor_cell_body_location`, the anatomical area from which the donor cell was taken.

A.2 Test dataset

The test dataset refers to the set of test samples, that is the samples of experimentally derived cell that we would like to evaluate by means of CellScore. Technically, every sample not included in the reference dataset could be supplied as a test sample. The expression profiles of the test samples should be normalized in the same manner as the reference dataset, and stored in an *ExpressionSet* object in the same manner as the reference dataset. The order of the genes (i.e. probes in the rows) in both (reference and test) data objects must be the same.

A.3 Input expression matrix for CellScore functions

The *ExpressionSet* objects for the standards and the test samples should just be combined to yield one *ExpressionSet* object.

A.4 Input table of cell transitions for CellScore functions

A separate data frame should define the cell transitions to be evaluated, based on the donor and target cell type. This data frame must contain the following columns:

- `start`: the donor cell type, using the same abbreviations as in the `general_cell_type` column of the *phenotype data frame*.
- `test`: the engineered cell type, using the same abbreviations as in the `sub_cell_type1` column of the *phenotype data frame*.
- `target`: the target cell type, using the same abbreviations as in the `general_cell_type` column of the *phenotype data frame*.