COSICC_DA_group and COSICC_DA_lineage

Magdalena Strauss

2025-06-06

This tutorial illustrates how to use COSICC_DA_group with SingleCellExperiments and with SeuratObjects.

```
knitr::opts_chunk$set(
  warning = FALSE,
  message = FALSE,
  error = FALSE
)

library(COSICC)
```

```
## Loading required package: SingleCellExperiment

## Loading required package: SummarizedExperiment

## Loading required package: MatrixGenerics

## Loading required package: matrixStats

##
## Attaching package: 'MatrixGenerics'

## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## Loading required package: GenomicRanges

## Loading required package: stats4

## Loading required package: BiocGenerics

##
## Attaching package: 'BiocGenerics'
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, saveRDS, setdiff,
##     table, tapply, union, unique, unsplit, which.max, which.min

## Loading required package: S4Vectors

##
## Attaching package: 'S4Vectors'

## The following object is masked from 'package:utils':
##
##     findMatches

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

## Loading required package: IRanges

## Loading required package: GenomeInfoDb

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

##
## Attaching package: 'Biobase'

## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians

## The following objects are masked from 'package:matrixStats':
##
##     anyMissing, rowMedians

## Loading required package: splatter

## Loading required package: scran

## Loading required package: scuttle

## Loading required package: scater

## Loading required package: ggplot2

## Loading required package: dplyr

##
## Attaching package: 'dplyr'
## The following object is masked from 'package:Biobase':
##
##     combine
```

```
## The following objects are masked from 'package:GenomicRanges':
##
##     intersect, setdiff, union
```

```
## The following object is masked from 'package:GenomeInfoDb':
##
##     intersect
```

```
## The following objects are masked from 'package:IRanges':
##
##     collapse, desc, intersect, setdiff, slice, union
```

```
## The following objects are masked from 'package:S4Vectors':
##
##     first, intersect, rename, setdiff, setequal, union
```

```
## The following objects are masked from 'package:BiocGenerics':
##
##     combine, intersect, setdiff, union
```

```
## The following object is masked from 'package:matrixStats':
##
##     count
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
## Loading required package: ggrepel
```

```
## Loading required package: scales
```

```
## Loading required package: ggthemes
```

```
## Loading required package: destiny
```

```
##
## Attaching package: 'destiny'
```

```
## The following object is masked from 'package:SummarizedExperiment':
##
##     distance
```

```
## The following object is masked from 'package:GenomicRanges':
##
##     distance
```

```
## The following object is masked from 'package:IRanges':
##
##     distance
```

COSICC_DA_group for SingleCellExperiments

First, we simulate a chimera-style data set, where cells have a fluorescent marker tdTomato or not.

```
sim_data_sce <- simulate_sce(seed = 10)
```

The function above simulated a knockout and a wild-type chimera data set.

```
sce_knockout <- sim_data_sce$case
sce_WT <- sim_data_sce$control
```

Let’s have a look at the simulated data sets:

```
colData(sce_knockout)
```

```
## DataFrame with 3894 rows and 5 columns
##                  Cell       Batch celltype ExpLibSize    tdTomato
##           <character> <character> <factor>  <numeric> <character>
## Cell1355    Cell1355      Batch1  Group8     57934.8         pos
## Cell136      Cell136      Batch1  Group8     63484.6         pos
## Cell4675    Cell4675      Batch1  Group9     68405.4         pos
## Cell1033    Cell1033      Batch1  Group9     58145.4         pos
## Cell4670    Cell4670      Batch1  Group10    65819.3         pos
## ...              ...         ...      ...        ...         ...
## Cell1623    Cell1623      Batch1   Group4    75971.8         neg
## Cell3078    Cell3078      Batch1   Group4    60108.4         neg
## Cell943      Cell943      Batch1   Group6    57467.0         neg
## Cell4234    Cell4234      Batch1   Group4    53096.3         neg
## Cell4793    Cell4793      Batch1   Group7    55155.8         neg
```

```
colData(sce_WT)
```

```
## DataFrame with 5000 rows and 5 columns
##                  Cell       Batch celltype ExpLibSize    tdTomato
##           <character> <character> <factor>  <numeric> <character>
## Cell1           Cell1      Batch1   Group1    66621.0         neg
## Cell2           Cell2      Batch1   Group9    72881.1         pos
## Cell3           Cell3      Batch1   Group4    59400.4         neg
## Cell4           Cell4      Batch1   Group3    92392.0         pos
## Cell5           Cell5      Batch1   Group8    43452.9         neg
## ...              ...         ...      ...        ...         ...
## Cell4996    Cell4996      Batch1   Group6    56514.5         neg
## Cell4997    Cell4997      Batch1   Group5    61676.2         neg
## Cell4998    Cell4998      Batch1   Group9    30729.4         neg
## Cell4999    Cell4999      Batch1   Group8    72140.7         pos
## Cell5000    Cell5000      Batch1   Group6    57851.1         neg
```

The tdTomato coloumn contains the values “pos” and “neg”.

To use COSICC_DA_group we need to rename them to TRUE and FALSE.

```
sce_WT$tdTomato <- sce_WT$tdTomato == "pos"
sce_knockout$tdTomato <- sce_knockout$tdTomato == "pos"
```

Furthermore, we need to rename the colData.

```
names(colData(sce_WT))[names(colData(sce_WT))== "tdTomato"] <- "marked"
names(colData(sce_WT))[names(colData(sce_WT)) == "celltype"] <- "cell_type"
names(colData(sce_knockout))[names(colData(sce_knockout))== "tdTomato"] <- "marked"
names(colData(sce_knockout))[names(colData(sce_knockout)) == "celltype"] <- "cell_type"
```

We also make sure that the cells from the WT and knockout data sets have different names.

```
colnames(sce_WT) <- paste0(colnames(sce_WT),"_WT")
colnames(sce_knockout) <- paste0(colnames(sce_knockout),"_knockout")
```

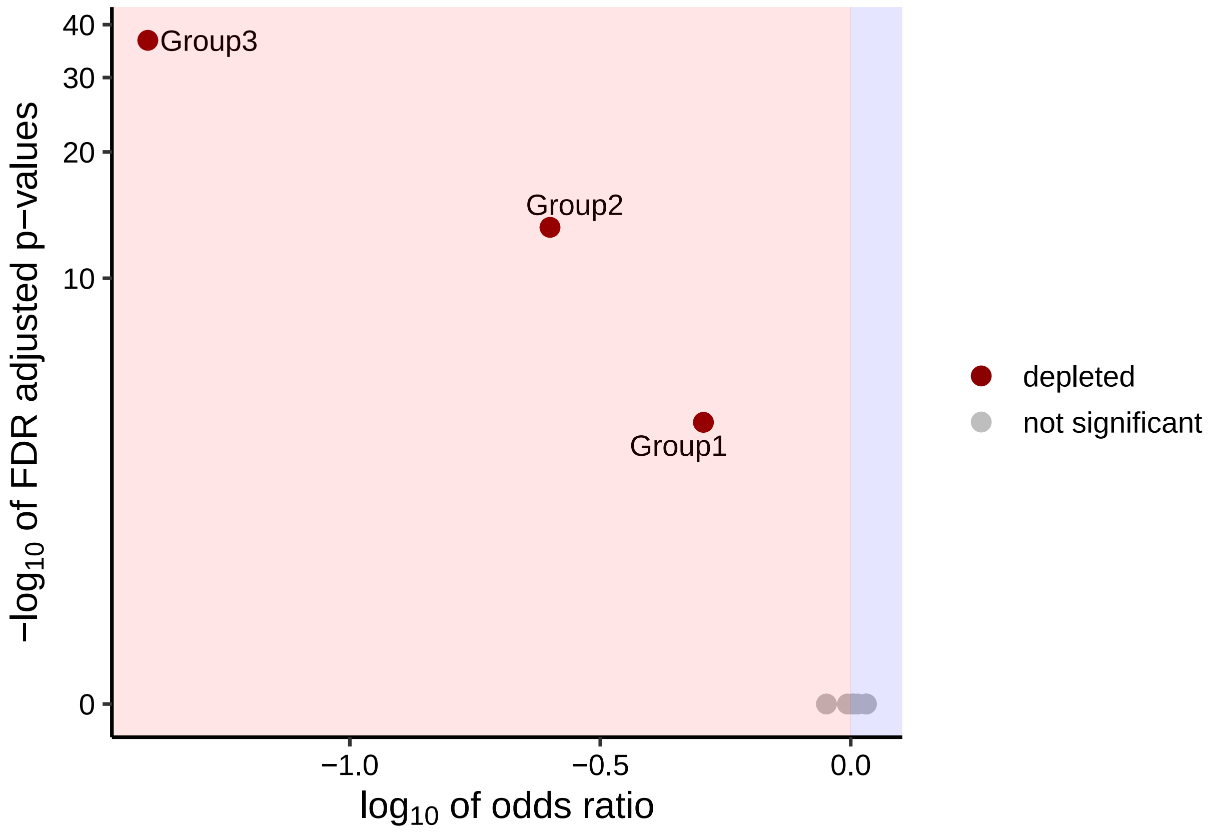
Now the data look as follows:

```
head(colData(sce_WT))
```

```
## DataFrame with 6 rows and 5 columns
##                  Cell       Batch cell_type ExpLibSize    marked
##           <character> <character>  <factor>  <numeric> <logical>
## Cell1_WT        Cell1      Batch1    Group1    66621.0     FALSE
## Cell2_WT        Cell2      Batch1    Group9    72881.1      TRUE
## Cell3_WT        Cell3      Batch1    Group4    59400.4     FALSE
## Cell4_WT        Cell4      Batch1    Group3    92392.0      TRUE
## Cell5_WT        Cell5      Batch1    Group8    43452.9     FALSE
## Cell6_WT        Cell6      Batch1    Group1    73600.7     FALSE
```

We can now COSICC_DA_group to identify depletion and/or enrichment of cell types for the tdTomato positive cells in the knockout chimeras.

```
DA_result_sce <- COSICC_DA_group(
    sce_case=sce_knockout,
    sce_control=sce_WT
)
```



The figure above illustrates the cell types Group1, Group2 and Group3 are depleted for the tdTomato positive group in the knockout data set compared to the wild-type data set.

The output of the function COSICC_DA_group is a data frame with the following columns.

Column Name	Description
`cell_type`	Cell type name.
`FDR`	FDR computed using the Benjamini-Hochberg method.
`odds_ratio`	Odds ratio of enrichment/depletion for each group of cells or cell type.
`sig`	Significance status: "enriched", "depleted", or "not significant".

Below we print the output.

```
DA_result_sce
```

```
##    cell_type     p_values odds_ratio             sig
## 3     Group3 1.815058e-37 0.03950769        depleted
## 2     Group2 5.769775e-14 0.25097447        depleted
## 1     Group1 3.974107e-05 0.50807159        depleted
## 4     Group4 1.000000e+00 1.07278126 not significant
## 5     Group5 1.000000e+00 1.07482222 not significant
## 6     Group6 1.000000e+00 1.01275506 not significant
## 7     Group7 1.000000e+00 0.89440486 not significant
## 8     Group8 1.000000e+00 1.03276313 not significant
## 9     Group9 1.000000e+00 1.00808883 not significant
## 10   Group10 1.000000e+00 0.98486236 not significant
```

COSICC_DA_group for SeuratObject

First, we simulate a chimera-style data set, where cells have a fluorescent marker tdTomato or not.

```
library(Seurat)
sim_data_seurat <- simulate_seurat(seed = 10)
seurat_knockout <- sim_data_seurat$case
seurat_WT <- sim_data_seurat$control
```

Now we create SingleCellExperiments.

```
sce_WT <- SingleCellExperiment(assays=list(counts=seurat_WT@assays$RNA),colData=seurat_WT@meta.data)
sce_knockout <- SingleCellExperiment(assays=list(counts=seurat_knockout@assays$RNA),colData=seurat_knoc
```

Now you can use COSICC_DA_group as described in the section on COSICC_DA_group for SingleCellExperiments above.

COSICC_DA_lineage

To illustrate COSICC_DA_lineage, we simulate lineage scores. In application in development, these scores might be scores indicating probabilities of a cells turning into each lineage. We use example data from the package.

```
data(package="COSICC")
```

This shows that the package contains the following example data sets:

lineage_scores
sce_DA_lineage_case
sce_DA_lineage_control

The scores look as follows:

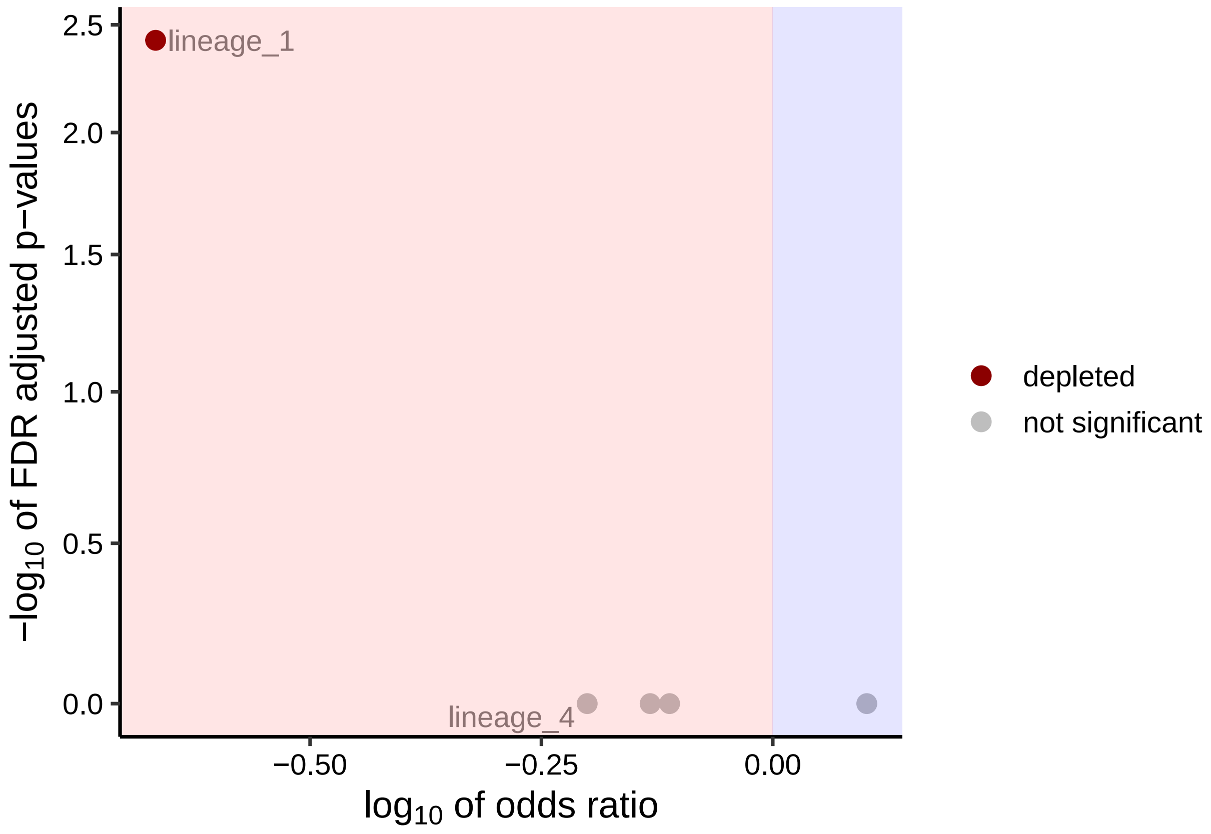
```
head(lineage_scores )
```

```
##                 lineage_1    lineage_2   lineage_3   lineage_4 lineage_5
## cell_151_case 0.00000000 0.0010299588 0.042756294 0.115318029 0.0000000
## cell_152_case 0.83263916 0.0693115699 0.000000000 0.000000000 0.0000000
## cell_153_case 0.07710486 0.0000360329 0.000000000 0.000000000 0.1835377
## cell_154_case 0.65712779 0.0589009524 0.000000000 0.000000000 0.1355032
## cell_155_case 0.13271436 0.0000000000 0.003163341 0.132757013 0.0000000
## cell_156_case 0.09525577 0.0649880589 0.000000000 0.007613369 0.0000000
##                          id
## cell_151_case cell_151_case
## cell_152_case cell_152_case
## cell_153_case cell_153_case
## cell_154_case cell_154_case
## cell_155_case cell_155_case
## cell_156_case cell_156_case
```

Note that one of the columns is called id and contains the cell names.

We can use the lineage scores and SingleCellExperiments as input to COSICC_DA_lineage.

```
lineage_result <- COSICC_DA_lineage(sce_DA_lineage_case,sce_DA_lineage_control,lineage_scores)
```



The output looks as follows:

```
head(lineage_result)
```

```
##     lineage    p_values odds_ratio             sig
## 1 lineage_1 0.003764327  0.2152840        depleted
## 2 lineage_2 1.000000000  0.7370509 not significant
## 3 lineage_3 1.000000000  1.2637089 not significant
## 4 lineage_4 1.000000000  0.6301841 not significant
## 5 lineage_5 1.000000000  0.7735475 not significant
```